



# USER'S MANUAL

---

O K I A R M - B A S E D M I C R O C O N T R O L L E R P R O D U C T S

## ML674001/ML675001 Series

---

January 31, 2004

*advantage*<sup>™</sup>  
microcontrollers

# Oki Semiconductor



# Preface

This Development Specification contains hardware and software specifications of Oki Electric's ML674001 Series/ML675001 Series 32-bit microcontroller. The manuals shown below are also available, and should be consulted a necessary.

## **ARM Architecture Reference Manual**

- Description of ARM instruction set architecture

## **ARM7TDMI Data Sheet**

- Description of ARM7TDMI instruction set
- Description of ARM7TDMI operation

The above documents are published by ARM Corporation.  
Please ensure that you refer to the latest versions.

# Notation

This manual uses the following notational conventions.

Type	Notation	Meaning
■ Numerals	<i>0xnn</i>	Hexadecimal number
■ Address	<i>0xnnnn_nnnn</i>	Hexadecimal number (It's means '0xnnnnnnnn')
■ Units	word, WORD	1 word = 32 bits
	byte, BYTE	1 byte = 8 bits
	M (mega-)	$10^6$
	K (Kilo-)	$2^{10} = 1024$
	k (kilo-)	$10^3 = 1000$
	m (milli-)	$10^{-3}$
	μ (micro)	$10^{-6}$
	n (nano-)	$10^{-9}$
	s	second(s)
■ Terms	"H" level	The VIH or VOH voltage level stipulated in the Electrical Characteristics as the voltage high signal level
	"L" level	The VIL or VOL voltage level stipulated in the Electrical Characteristics as the voltage low signal level

## Table of Contents

### Chapter 1 Introduction

1.1 Features.....	1-1
1.2 Functional Blocks.....	1-4
1.2.1 ML674001 series Block Diagram.....	1-4
1.2.2 ML675001 series Block Diagram.....	1-5
1.3 Pins.....	1-6
1.3.1 Pin Layout.....	1-6
1.3.1.1 LFBGA.....	1-6
1.3.1.2 LQFP.....	1-7
1.3.2 Pin List.....	1-8
1.3.3 Pin Descriptions.....	1-12
1.3.4 Pin States.....	1-16
1.3.5 Pin Structure and Treatment.....	1-18
1.3.6 Treatment of Unused Pins.....	1-20

### Chapter 2 CPU

2.1 Overview.....	2-1
2.2 CPU Operation States.....	2-1
2.2.1 State Transitions.....	2-1
2.3 Address Space.....	2-1
2.4 Memory Format.....	2-2
2.5 Instruction Length.....	2-2
2.6 Data Types.....	2-2
2.7 Processor Modes.....	2-2
2.8 Registers.....	2-3
2.8.1 ARM State Registers.....	2-3
2.8.2 THUMB State Registers.....	2-5
2.8.3 Relationships Between ARM and THUMB State Registers.....	2-6
2.8.4 Accessing Upper Registers from THUMB State.....	2-6
2.9 Program Status Registers.....	2-7
2.9.1 Condition Code Flags.....	2-7
2.9.2 Control Bits.....	2-7
2.9.3 Reserved Bits.....	2-8
2.10 Instruction Set Features.....	2-9
2.10.1 ARM Instruction Set.....	2-9
2.10.2 THUMB Instruction Set.....	2-9
2.11 Addressing Modes.....	2-10
2.11.1 Load/Store Instructions.....	2-10
2.11.2 Multiple Load/Store Instructions.....	2-10
2.12 Exceptions.....	2-11
2.12.1 Switching to Exception Handler.....	2-11
2.12.2 Returning from Exception Handler.....	2-11
2.12.3 Summary of Exception Switching.....	2-12
2.12.4 FIQ.....	2-12
2.12.5 IRQ.....	2-12
2.12.6 Aborts.....	2-13
2.12.7 Software Interrupts.....	2-13
2.12.8 Undefined Instructions.....	2-13
2.12.9 Exception Vectors.....	2-14
2.12.10 Exception Priority Order.....	2-14
2.13 Resets.....	2-15

---

## Chapter 3 Address Mapping

---

3.1 Overview .....	3-1
3.1.1 Pin List.....	3-1
3.1.2 Register List.....	3-1
3.2 Address Map.....	3-2
3.2.1 ML674001 series address map.....	3-2
3.2.2 ML675001 series address map.....	3-3
3.3 Register Descriptions.....	3-4
3.3.1 Remap Control Register (RMPCON) .....	3-4
3.3.2 ROM Select Register (ROMSEL).....	3-5
3.4 Remap Description .....	3-6
3.4.1 Remap Setting.....	3-6
3.4.2 Boot Control .....	3-7
3.4.3 Notes of Address map.....	3-7
3.4.4 Remap Setting Example.....	3-8

---

## Chapter 4 Chip Configuration

---

4.1 Overview .....	4-1
4.1.1 Pin List.....	4-1
4.2 Pin Description .....	4-1

---

## Chapter 5 Clock Generator

---

5.1 Overview .....	5-1
5.1.1 Components .....	5-1
5.1.2 Pin List.....	5-2
5.1.3 PLL Clock frequency setup (ML675001 Series Only. *) .....	5-2
5.2 Sample Crystal Connections.....	5-2

---

## Chapter 6 Reset Control

---

6.1 Overview .....	6-1
6.1.1 Pin List.....	6-1
6.2 Reset Types.....	6-1
6.2.1 External Reset Input.....	6-1
6.2.2 Watchdog Timer Overflow .....	6-1
6.3 Operational Description.....	6-2

---

## Chapter 7 Power Management

---

7.1 Overview .....	7-1
7.2 Power Management Functions.....	7-1
7.2.1 Register List.....	7-3
7.3 Register Descriptions.....	7-4
7.3.1 Block Clock Control Register (BCKCTL).....	7-4
7.3.2 Clock Stop Register (CLKSTP).....	7-7
7.3.3 Clock Gear Control Register (CGBCNT0).....	7-9
7.3.4 Clock Wait Register (CKWT) .....	7-10
7.3.5 Stopping Clock Signals to Functional Blocks.....	7-11
7.3.6 Clock Gear .....	7-11
7.3.7 HALT Mode .....	7-12
7.3.8 STANDBY Mode .....	7-12
7.4 Using Power Management with DRAM.....	7-13
7.4.1 Activating Self Refresh Operation.....	7-13
7.4.2 Deactivating Self Refresh Operation .....	7-13

---

## Chapter 8 Interrupt Controller

8.1 Overview .....	8-1
8.1.1 Components .....	8-2
8.1.2 Pin List .....	8-3
8.1.3 Register List .....	8-3
8.2 Interrupt Sources .....	8-4
8.2.1 External Fast Interrupt (EFIQ_N) .....	8-4
8.2.2 External Interrupts (EXINT[n]) .....	8-4
8.2.3 Internal Interrupts (IRQn) .....	8-4
8.2.4 Interrupt Source List .....	8-5
8.3 Interrupt Levels .....	8-6
8.4 Register Descriptions .....	8-7
8.4.1 IRQ Register (IRQ) .....	8-7
8.4.2 Software Interrupt Register (IRQS) .....	8-8
8.4.3 FIQ Register (FIQ) .....	8-9
8.4.4 FIQRAW Register (FIQRAW) .....	8-10
8.4.5 FIQ Enable Register (FIQEN) .....	8-11
8.4.6 IRQ Number Register (IRN) .....	8-12
8.4.7 Current Interrupt Level Register (CIL) .....	8-13
8.4.8 Interrupt Level Control Register 0 (ILC0) .....	8-14
8.4.9 Interrupt Level Control Register 1 (ILC1) .....	8-16
8.4.10 Current Interrupt Level Clear Register (CILCL) .....	8-18
8.4.11 Current Interrupt Level Encode Register (CILE) .....	8-19
8.4.12 IRQ Clear Register (IRCL) .....	8-20
8.4.13 IRQA Register (IRQA) .....	8-21
8.4.14 IRQ Detection Mode Setting Register (IDM) .....	8-23
8.4.15 Interrupt Level Control Register (ILC) .....	8-24
8.4.16 Register Settings for Interrupt Sources .....	8-26
8.5 Description of Operation .....	8-27
8.5.1 External Fast Interrupt (EFIQ_N) .....	8-27
8.5.2 External and Internal Interrupts (IRQn) .....	8-28
8.5.3 Nested Interrupts and Re-Entrant Interrupt Service Routines .....	8-30
8.5.4 Important Notes on Interrupts .....	8-31
8.5.5 Waking from HALT and STANDBY Modes .....	8-32
8.5.6 Error Response .....	8-33
8.5.7 Interrupt Response Times .....	8-33
8.6 Interrupt Acceptance Timing Charts .....	8-34
8.6.1 FIQ Interrupt Timing Chart .....	8-34
8.6.2 IRQ Interrupt Timing Chart (nIR0 to nIR15) .....	8-34
8.6.3 IRQ Interrupt Timing Chart (nIR16 to nIR31) .....	8-36

## Chapter 9 Cache Memory

9.1 Overview .....	9-1
9.1.1 Configuration .....	9-1
9.1.2 List of Control Registers .....	9-2
9.2 Description of Control Registers .....	9-2
9.2.1 Cache Lock Control Register (CON) .....	9-2
9.2.2 Cacheable Register (CACHE) .....	9-4
9.2.3 FLUSH Register (FLUSH) .....	9-5
9.3 Description of Operations .....	9-6
9.3.1 Initialization of Cache Memory .....	9-6
9.3.2 Cacheable/Non-cacheable Setting .....	9-6
9.3.3 Description of operations .....	9-6

9.3.4	Lock Function .....	9-6
9.3.5	Load Mode .....	9-7
9.3.6	Flushing Function .....	9-7
9.4	Precautions in Use .....	9-8
9.4.1	Precautions when Using DMA Transfer .....	9-8
9.4.2	Precautions in Remapping .....	9-8
9.5	Examples of Setting .....	9-9
9.5.1	Example of Cache Memory Initialization .....	9-9
9.5.2	Example of Cache Memory Flushing Requiring Data Write Back .....	9-9
9.5.3	Example of Lock Setting Procedure .....	9-10
9.6	Typical Operation Timings .....	9-12

## Chapter 10 Built-In Memory

10.1	Overview .....	10-1
10.2	Built-In SRAM .....	10-1
10.3	Built-In FLASH ROM .....	10-1

## Chapter 11 External Memory Controller

11.1	Overview .....	11-1
11.1.1	Pin List .....	11-1
11.1.2	Register List .....	11-2
11.2	Register Descriptions .....	11-3
11.2.1	Bus Width Control Register (BWC) .....	11-3
11.2.2	External I/O bank 2/3 Bus Width Control Register (IO23BWC) (*1: ML675001 Series only) .....	11-5
11.2.3	External ROM Access Control Register (ROMAC) .....	11-6
11.2.4	External SRAM Access Control Register (RAMAC) .....	11-8
11.2.5	External I/O Bank 0/1 Access Control Register (IO01AC) .....	11-10
11.2.6	External I/O Bank 2/3 Access Control Register (IO23ACX *1, IO23ACY *2) .....	11-12
11.2.7	DRAM Bus Width Control Register (DBWC) .....	11-14
11.2.8	DRAM Control Register (DRMC) .....	11-15
11.2.9	DRAM Characteristics Control Register (DRPC) .....	11-17
11.2.10	SDRAM Mode Register (SDMD) .....	11-18
11.2.11	DRAM Command Register (DCMD) .....	11-20
11.2.12	DRAM Refresh Cycle Control Register 0 (RFSH0) .....	11-21
11.2.13	DRAM Refresh Cycle Control Register 1 (RFSH1) .....	11-22
11.2.14	DRAM Power Down Control Register (RDWC) .....	11-24
11.3	Operational Description .....	11-25
11.3.1	Bus Width .....	11-25
11.3.2	ROM/SRAM Control .....	11-25
11.3.3	I/O Banks Control .....	11-26
11.3.4	DRAM Control .....	11-27
11.3.5	Access Timing Parameters for DRAM .....	11-31
11.4	Access Timing .....	11-34
11.4.1	Accessing External Devices .....	11-34
11.4.1.1	External ROM/RAM Access .....	11-34
11.4.1.2	External I/O Bank Access .....	11-35
11.4.1.3	EDO DRAM Access .....	11-37
11.4.1.4	SDRAM Access .....	11-39
11.5	DRAM Power Management .....	11-42
11.6	Sample External Memory Connections .....	11-43
11.6.1	Connecting ROM .....	11-44
11.6.2	Connecting SRAM .....	11-46
11.6.3	Connecting EDO DRAM .....	11-48
11.6.4	Connecting SDRAM .....	11-50

## Chapter 12 Direct Memory Access Controller (DMAC)

12.1 Overview.....	12-1
12.1.1 Components .....	12-2
12.1.2 Pin List.....	12-4
12.1.3 Register List.....	12-4
12.2 Register Descriptions.....	12-5
12.2.1 DMA Mode Register (DMAMOD) .....	12-5
12.2.2 DMA Status Register (DMASTA).....	12-6
12.2.3 DMA Transfer Complete Status Register (DMAINT).....	12-7
12.2.4 DMA Channel Mask Registers (DMACMSK0 and DMACMSK1).....	12-9
12.2.5 DMA Transfer Mode Registers (DMACTMOD0 and DMACTMOD1) .....	12-10
12.2.6 DMA Transfer Source Address Registers (DMACSA0 and DMACSA1).....	12-12
12.2.7 DMA Transfer Destination Address Registers (DMACDA0 and DMACDA1) .....	12-13
12.2.8 DMA Transfer Count Registers (DMACSI0 and DMACSI1).....	12-14
12.2.9 DMA Transfer Complete Status Clear Registers (DMACCINT0 and DMACCINT1) .....	12-15
12.3 Operational Description.....	12-16
12.3.1 DMA Transfer Modes.....	12-16
12.3.2 DMA Request Sources.....	12-16
12.3.3 Starting a DMA Transfer .....	12-17
12.3.4 Ending a DMA Transfer .....	12-18
12.3.5 DMA Channel Priority.....	12-20
12.3.6 Important Usage Notes .....	12-21
12.4 DMA Transfer Timing.....	12-22
12.4.1 Starting a Transfer .....	12-22
12.4.2 Transfer Timing .....	12-23

## Chapter 13 GPIO

13.1 Overview.....	13-1
13.1.1 Components .....	13-2
13.1.2 Pin List.....	13-3
13.1.3 Register List.....	13-4
13.2 Register Descriptions.....	13-5
13.2.1 Port Output Registers (GPPOA, GPPOB, GPPOC, GPPOD and GPPOE) .....	13-5
13.2.2 Port Input Registers (GPPIA,GPPIB, GPPIC,GPPID, and GPPIE).....	13-5
13.2.3 Port Mode Registers (GPPMA,GPPMB, GPPMC, GPPMD and GPPME).....	13-6
13.2.4 Port Interrupt Enable Registers (GPIEA,GPIEB, GPIEC,GPIED and GPIEE) .....	13-7
13.2.5 Port Interrupt Polarity Register (GPIPA,GPIPB,GPIPC,GPIPD and GPIPE) .....	13-8
13.2.6 Port Interrupt Status Registers (GPISA,GPISB, GPISC,GPISD and GPISE) .....	13-9
13.2.7 Port Function Select Register (GPCTL) .....	13-10
13.3 Description of Operation .....	13-14
13.3.1 Interrupt Requests .....	13-14
13.3.2 Primary/Secondary function configuration .....	13-15

## Chapter 14 Watchdog Timer (WDT)

14.1 Overview.....	14-1
14.1.1 Components .....	14-1
14.1.2 Register List.....	14-1
14.2 Register Descriptions.....	14-2
14.2.1 Watchdog Timer Control Register (WDTCON).....	14-2
14.2.2 Time Base Counter Control Register (WDTBCON) .....	14-3
14.2.3 Status Register (WDSTAT).....	14-5
14.3 Description of Operation .....	14-6
14.3.1 Operation Modes.....	14-6



14.3.2	Interval Timer Operation .....	14-6
14.3.3	Watchdog Timer Operation .....	14-6
14.3.4	Starting Timer .....	14-6

## Chapter 15 Timers

---

15.1	Overview.....	15-1
15.1.1	Components .....	15-1
15.1.2	Register List.....	15-3
15.2	Register Descriptions.....	15-4
15.2.1	System Timer Enable Register (TMEN).....	15-4
15.2.2	System Timer Reload Register (TMRLR).....	15-5
15.2.3	System Timer Overflow Register (TMOVFR) .....	15-6
15.2.4	Timer Control Registers (TIMECNTL0 to TIMECNTL5).....	15-7
15.2.5	Timer Base Registers (TIMEBASE0 to TIMEBASE5).....	15-9
15.2.6	Timer Counter Register (TIMECNT0 to TIMECNT5).....	15-10
15.2.7	Timer Compare Registers (TIMECMP0 to TIMECMP5) .....	15-11
15.2.8	Timer Status Registers (TIMESTAT0 to TIMESTAT5) .....	15-12
15.3	Description of Operation .....	15-13
15.3.1	System Timer .....	15-13
15.3.2	Auto Reload Timers.....	15-14
15.3.3	Specifying Clock and Starting Auto Reload Timers.....	15-15

## Chapter 16. PWM Generator

---

16.1	Overview.....	16-1
16.1.1	Components .....	16-1
16.1.2	Pin List.....	16-2
16.1.3	Register List.....	16-2
16.2	Register Descriptions.....	16-3
16.2.1	PWM Registers (PWR0 and PWR1) .....	16-3
16.2.2	PWM Period Registers (PWCY0 and PWCY1) .....	16-4
16.2.3	PWM Counter (PWC0 and PWC1) .....	16-5
16.2.4	PWM Control Registers (PWCON0 and PWCON1).....	16-6
16.2.5	PWM Interrupt Status Register (PWINTSTS).....	16-7
16.3	Description of Operation .....	16-8
16.3.1	PWM Operation.....	16-8
16.3.2	Timing Examples.....	16-8

## Chapter 17 SIO

---

17.1	Overview.....	17-1
17.1.1	Components .....	17-1
17.1.2	Pin List.....	17-2
17.1.3	Control Register List.....	17-2
17.2	Control Register Descriptions.....	17-3
17.2.1	Transfer Buffer Register (SIOBUF) .....	17-3
17.2.2	SIO Status Register (SIOSTA) .....	17-4
17.2.3	SIO Control Register (SIOCON) .....	17-6
17.2.4	Baud Rate Control Register (SIOBCN).....	17-8
17.2.5	Baud Rate Timer Register (SIOBT).....	17-9
17.2.6	SIO test control Register (SIOTCN).....	17-10
17.3	Description of Operation .....	17-11
17.3.1	Transmitting Data .....	17-11
17.3.2	Receiving Data.....	17-11
17.3.3	Generating Baud Rate Clock .....	17-12
17.3.4	Receive Interrupts.....	17-12

17.3.5 Transmit Interrupts .....	17-13
17.4 Important Usage Notes .....	17-14

## Chapter 18 UART with FIFO(16byte)

18.1 Overview.....	18-1
18.1.1 Components .....	18-2
18.1.2 Pins .....	18-2
18.1.3 Register List.....	18-3
18.2 Register Descriptions.....	18-4
18.2.1 Receiver Buffer Register (UARTBR) .....	18-4
18.2.2 Transmitter Holding Register (UARTTHR) .....	18-5
18.2.3 Interrupt Enable Register (UARTIER) .....	18-6
18.2.4 Interrupt Identification Register (UARTIIR) .....	18-7
18.2.5 FIFO Control Register (UARTFCR) .....	18-9
18.2.6 Line Control Register (UARTLCR) .....	18-11
18.2.7 Modem Control Register (UARTMCR) .....	18-13
18.2.8 Line Status Register (UARTLSR) .....	18-15
18.2.9 Modem Status Register (UARTMSR) .....	18-18
18.2.10 Scratch Register (UARTSCR) .....	18-20
18.2.11 Divisor Latch (LSB) (UARTDLL) .....	18-21
18.2.12 Divisor Latch (MSB) (UARTDLM).....	18-22
18.3 Description of Operation .....	18-23
18.3.1 Transmitting Data .....	18-23
18.3.2 Receiving Data.....	18-24
18.3.3 Generating Baud Rate Clock .....	18-26
18.3.4 Buffered Operation .....	18-27
18.3.5 Queue Polled Mode .....	18-28
18.3.6 Error Status .....	18-29
18.3.7 Setup Procedure .....	18-30

## Chapter 19 Synchronous SIO

19.1 Overview .....	19-1
19.1.1 Configuration .....	19-1
19.1.2 List of Pins.....	19-2
19.1.3 List of Registers .....	19-2
19.2 Registers .....	19-3
19.2.1 Synchronous SIO Transmit/Receive Buffer Register (SSIOBUF) .....	19-3
19.2.2 Synchronous SIO Status Register (SSIOST) .....	19-4
19.2.3 Synchronous SIO Interrupt Request Register (SSIOINT) .....	19-5
19.2.4 Synchronous SIO Interrupt Enable Register (SSIOINTEN).....	19-6
19.2.5 Synchronous SIO Control Register (SSIOCON) .....	19-7
19.2.6 Synchronous SIO Test Control Register (SSIoTSCON).....	19-8
19.3 Operations.....	19-9
19.3.1 Master Mode/Slave Mode.....	19-9
19.3.2 Transmit Operation .....	19-9
19.3.3 Receive Operation.....	19-11
19.3.4 Interrupt Signal .....	19-13

## Chapter 20 I2C

20.1 Overview .....	20-1
20.1.1 Configuration .....	20-1
20.1.2 List of Pins.....	20-2
20.1.3 List of Registers .....	20-2
20.2 Registers .....	20-3

20.2.1	I2C Bus Control Register (I2CCON).....	20-3
20.2.2	I2C Bus Slave Address Mode Register (I2CSAD) .....	20-5
20.2.3	I2C Bus Transfer Speed Register (I2CCLR) .....	20-6
20.2.4	I2C Bus Status Register (I2CSR).....	20-7
20.2.5	I2C Bus Interrupt Request Register (I2CIR).....	20-8
20.2.6	I2C Bus Interrupt Mask Register (I2CIMR) .....	20-9
20.2.7	I2C Bus Transmit/Receive Data Register (I2CDR) .....	20-10
20.2.8	I2C Bus Transfer Speed Counter (I2CBC) .....	20-11
20.3	Operations.....	20-12
20.3.1	Transmit operation (transfer of 1 byte from master to slave, in 7-bit address mode) .....	20-12
20.3.2	Receive operation (transfer of 1 byte from slave to master, in 7-bit address mode).....	20-13
20.3.3	Transmit operation (transfer of 2 or more bytes from master to slave, in 7-bit address mode) .....	20-14
20.3.4	Receive operation (transfer of 2 byte or more from slave to master, in 7-bit address mode) .....	20-15
20.3.5	Restart sequence transmit operation .....	20-16
20.3.6	To receive 1-byte of data from another slave device after transmitting 1-byte of data:.....	20-16
20.3.7	Start byte transmit operation .....	20-17
20.3.8	7-bit Address Mode and 10-bit Address Mode .....	20-19

## Chapter 21 Analog-to-Digital Converter

21.1	Overview.....	21-1
21.1.1	Components .....	21-1
21.1.2	Pin List.....	21-2
21.1.3	Control Register List.....	21-2
21.2	Control Register Descriptions.....	21-3
21.2.1	Analog-to-Digital Converter Control Register 0 (ADCON0).....	21-3
21.2.2	Analog-to-Digital Converter Control Register 1 (ADCON1).....	21-5
21.2.3	Analog-to-Digital Converter Control Register 2 (ADCON2).....	21-6
21.2.4	Analog-to-Digital Converter Interrupt Control Register (ADINT).....	21-7
21.2.5	Analog-to-Digital Converter Forced Interrupt Register (ADFINT).....	21-9
21.2.6	Analog-to-Digital Converter Result Registers (ADR0 to ADR3).....	21-10
21.3	Operational Description .....	21-11
21.3.1	Scan Mode .....	21-11
21.3.2	Select Mode .....	21-11

## Chapter 22 Built-In Flash Memory

22.1	Overview.....	22-1
22.1.1	Block Diagram.....	22-2
22.2	Flash Memory Programming .....	22-3
22.2.1	General Description of Flash Memory Programming.....	22-3
22.2.2	Flash Memory Programming Method Using the JTAG Debug Function .....	22-3
22.2.3	Flash Memory Programming Method Using the Built-in Boot Program.....	22-5
22.3	Built-in Boot Program .....	22-7
22.3.1	Functional Description.....	22-7
22.3.2	Operating Procedure .....	22-7
22.3.3	Operating Environment.....	22-7
22.4	SDP Command Sequence Control for Built-In Flash Memory.....	22-8
22.4.1	Operational Description .....	22-8
22.4.2	Command Entries .....	22-9
22.4.3	Read/Reset (Software Reset) .....	22-9
22.4.4	Erase .....	22-9
22.4.5	Program.....	22-9
22.4.6	Protect.....	22-9
22.4.7	Protect Cancel.....	22-9
22.4.8	Product Identification/Software ID.....	22-11

22.4.9 Verify Protect.....	22-11
22.4.10 Hardware Reset.....	22-11
22.4.11 Detecting the End of an Erase or Program Cycle .....	22-12

## Chapter 23 JTAG

---

23.1 Overview.....	23-1
23.1.1 Configuration .....	23-1
23.1.2 Pin List.....	23-2
23.2 On-Board Debug Function.....	23-3
23.2.1 Necessary Conditions .....	23-3
23.2.2 Connections .....	23-3
23.3 Boundary Scan Function.....	23-4
23.3.1 Boundary Scan Control Circuit.....	23-5
23.3.2 Registers .....	23-6
23.3.3 TAP Controller .....	23-7
23.3.4 Instructions .....	23-8

## Chapter 24 Electrical Characteristics

---

24.1 Absolute Maximum Ratings .....	24-1
24.2 Operating Conditions .....	24-2
24.3 Electrical Characteristics .....	24-3
24.3.1 DC Characteristics for ML674001 Series .....	24-3
24.3.2 DC Characteristics for ML675001 Series .....	24-4
24.4 AC Characteristics .....	24-5
24.4.1 AC Characteristics for ML674001 Series .....	24-5
24.4.2 AC Characteristics for ML675001 Series .....	24-14
24.4.3 Timing Charts .....	24-23
24.5 Analog-to-Digital Converter Characteristics .....	24-48

## Appendixes

---

Appendix A. Register List.....	A-1
Appendix B. Package Dimensions .....	A-6

## Revision History

---

Revision History .....	R-1
------------------------	-----

## *Chapter 1*

# **Introduction**

---

## Chapter 1 Introduction

### 1.1 Features

This high-performance CMOS 32-bit micro-controller combines the 32-bit ARM7TDMI™ core, a RISC CPU developed by Advanced RISC Machines Limited (ARM), with a DMA controller, serial ports, PWM generator, analog-to-digital converter, 16-bit timers, and other peripheral functions on a single LSI.

In addition to 32-bit data processing, this LSI includes internal RAM and onboard peripherals that make it ideal for such embedded control applications as PC peripherals and communication terminals.

Finally, there is a built-in external memory controller for directly connecting ROM, SRAM, SDRAM, other memory types, and peripheral devices.

The following is a list of features.

- CPU
  - 32-bit RISC CPU (ARM7TDMI)
  - Built-in 8KB unified cache (ML675001 series only)
  - Little endian byte order
  - Operating frequency:
    - ML674001 series :1 MHz to 33 MHz
    - ML675001 series :1 MHz to 60 MHz
  - Instruction set: Free switching between a highly dense 32-bit instruction set and a 16-bit subset offering higher object code efficiency
  - General-purpose registers: 32-bit × 31
  - Barrel shifter: Simultaneous ALU and barrel shift operations in the same instruction
  - Multiplier (32-bit × 8-bit)
  - JTAG interface for debugging
- Built-in Memory
  - SRAM 32Kbytes (8K x 32bits), 1 cycle access
  - FLASH memory
    - ML674001: ROM-less version
    - ML67Q4002: 256Kbytes (128K x 16bits)
    - ML67Q4003: 512Kbytes (256K x 16bits)
    - ML675001: ROM-less version
    - ML67Q5002: 256Kbytes (128K x 16bits)
    - ML67Q5003: 512Kbytes (256K x 16bits)
- Interrupt Controller
  - One fast interrupt (FIQ) source (external)
  - 27 interrupt (IRQ) sources (23 internal and 4 external)
  - Independent masking for each FIQ and IRQ source
  - Independent interrupt priority level settings for each IRQ source
  - Priority control blocking IRQ requests with priority levels at or below those for interrupt requests currently being processed
  - Choice of level or edge sensing for external IRQ sources EXINT0 to EXINT3.
- Timers
  - One 16-bit system timer
  - Six 16-bit auto reload timers
    - Independent clock settings for each timer
    - Independent choice of one shot or interval timer operation for each timer
  - Maximum period: 30 ms or more
- Watchdog Timer
  - One 16-bit timer
  - Choice of interval or watchdog timer operation
  - Choice of interrupt or reset upon overflow
  - Maximum period: 200 ms or more

- GPIO
  - Four 8-bit ports
  - One 10-bit port
  - Individual settings for pin I/O direction
  - Individual settings for pin interrupt requests
- PWM
  - Two outputs with 16-bit resolution
  - Maximum period; 30 ms or more
- Analog-to-Digital Converter
  - Four channels of 10-bit resolution, each using consecutive comparison
  - Sample and hold function
  - Choice of scan or select operation
  - Conversion time:
    - ML674001 series :5  $\mu$ s to 25  $\mu$ s
    - ML675001 series :2  $\mu$ s to 25  $\mu$ s
- DMA Controller
  - Two channels
  - Choice of fixed or round robin mode for channel priority order
  - Choice of cycle-steal or burst mode for requesting bus access
  - Choice of software or external DMA transfer requests
  - Maximum transfer count: 65,536
  - Data transfer sizes: 8-, 16-, and 32-bit
- External Memory Controller
  - ROM access
    - Supports 16-bit devices
    - Supports Flash memory
  - SRAM access
    - Supports 16-bit devices
    - Supports Asynchronous SRAM
  - DRAM access
    - Supports EDO DRAM and SDRAM
    - Supports 16-bit and 8-bit (only possible for EDO DRAM) devices
    - Supports distributed CAS before RAS (CBR) refresh
  - External I/O banks access
    - Four banks of external I/O space
    - Supports 8- and 16-bit devices
    - Supports external WAIT input signal (XWAIT)
- SIO
  - Full duplex asynchronous operation
  - Built-in baud rate generator
- Synchronous SIO
  - Choice of Master or Slave operation
  - Choice of LSB or MSB fast operation
- I2C
  - Master mode only
  - Supports fast mode (400Kbps) , standard mode (100Kbps)
- UART
  - 16550A-compatible asynchronous communications
  - 16-byte FIFO each for transmit and receive operations
  - Full duplex collision operation
  - Built-in baud rate generator
- Clock Signal
  - Connects to crystal
    - ML674001 series : 16 MHz to 33 MHz
    - ML675001 series : 5MHz to 14 MHz
  - Also supports direct external clock input

- Power Management
  - STANDBY mode: Stop clock in software
  - HALT mode: Stop clock signals to CPU and other key components in software
  - Clock gear: clock change is dynamically possible in the division ratio of clock input frequency.
    - ML674001 series : 1/1, 1/2, 1/4, 1/8, or 1/16
    - ML675001 series : 1/1, 1/2, 1/4, 1/8, 1/16, or 1/32
  - Functional blocks: Stop clock signals to individual function blocks



## 1.2 Functional Blocks

### 1.2.1 ML674001 series Block Diagram

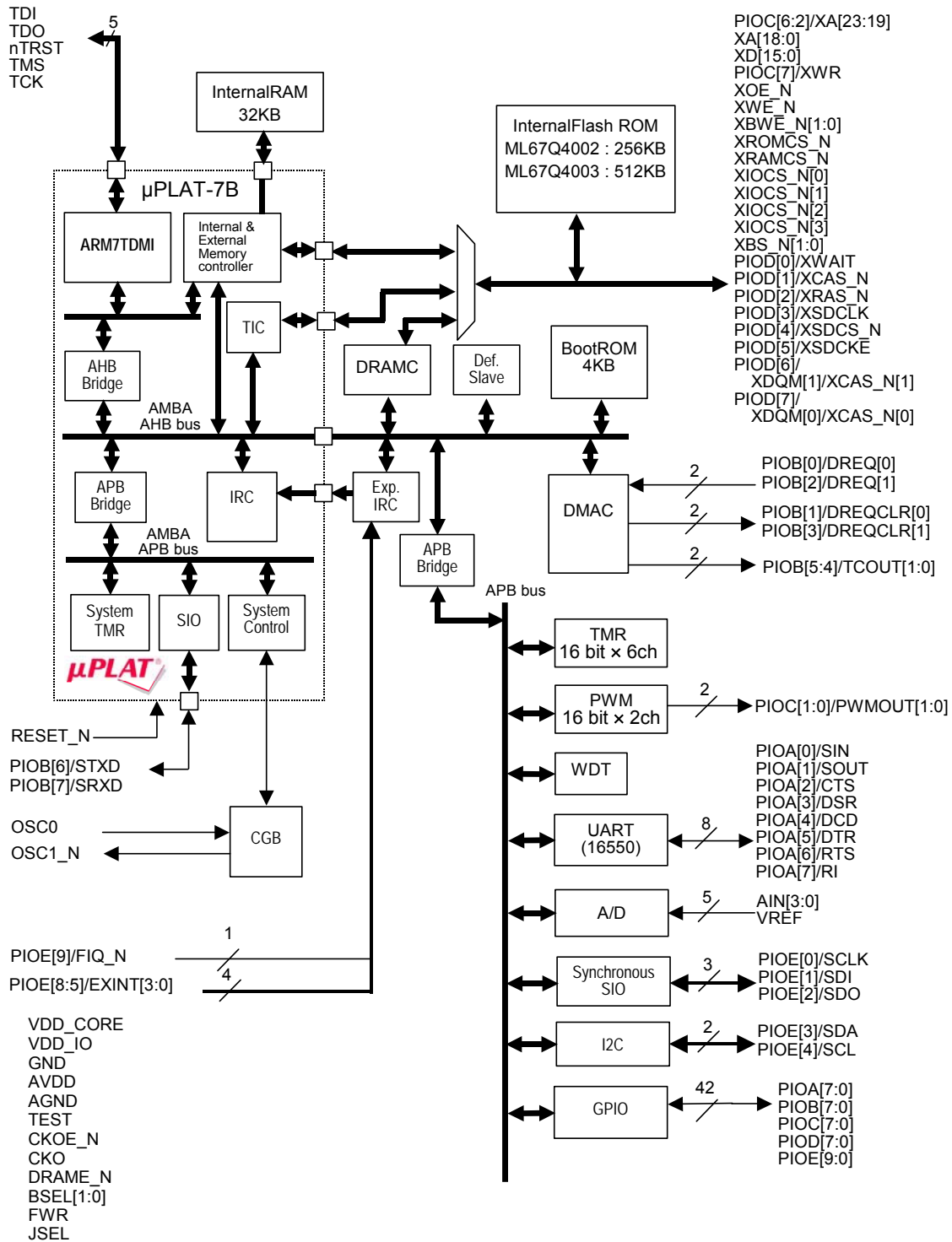


Figure 1.1 ML674001 series Block Diagram

1.2.2 ML675001 series Block Diagram

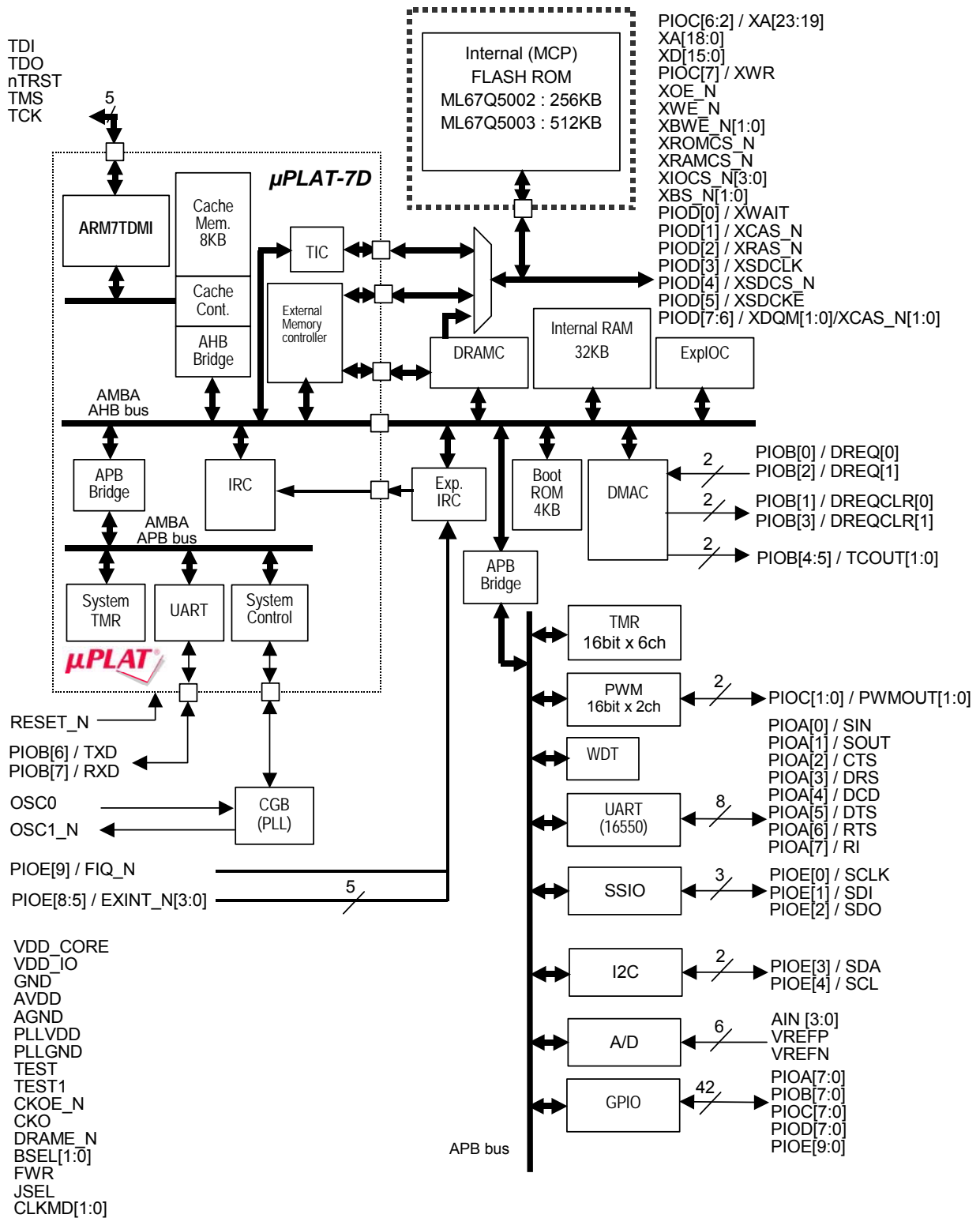


Figure 1.2 ML675001 series Block Diagram

### 1.3 Pins

#### 1.3.1 Pin Layout

##### 1.3.1.1 LFBGA

	13	12	11	10	9	8	7	6	5	4	3	2	1
N	PIOD[6]/XDQM[1]	XIOCS_N[3]	XIOCS_N[1]	XRAMC_S_N	XBWE_N[0]	XOE_N	PIOC[4]/XA[21]	XA[16]	XA[14]	XA[11]	XA[9]	XA[7]	XA[6]
M	PIOD[7]/XDQM[0]	XIOCS_N[2]	XIOCS_N[0]	XWE_N	PIOC[7]/XWR	PIOC[6]/XA[23]	PIOC[2]/XA[19]	XA[17]	XA[15]	XA[13]	XA[10]	XA[4]	XA[5]
L	PIOB[1]/DREQC_LR[0]	PIOB[2]/DREQ[1]	PIOB[0]/DREQ[0]	XROMC_S_N	XBWE_N[1]	PIOC[5]/XA[22]	PIOC[3]/XA[20]	XA[18]	XA[12]	VDD_IO	XA[8]	XA[2]	GND
K	PIOB[3]/DREQC_LR[1]	PIOB[5]/TCOUT[1]	VDD_IO	GND	VDD_IO	VDD_CORE	VDD_IO	GND	GND	XA[3]	XA[0]	XD[13]	XA[1]
J	PIOC[0]/PWMOUT[0]	GND	PIOB[4]/TCOUT[0]	PIOC[1]/PWMOUT[1]	<b>144pin LFBGA (TOP VIEW)</b>					VDD_IO	XD[15]	XD[11]	XD[14]
H	XBS_N[0]	XBS_N[1]	PIOD[0]/XWAIT	VDD_CORE						VDD_CORE	XD[10]	NC	XD[12]
G	PIOD[2]/XCRAS_N	PIOD[1]/XCAS_N	VDD_IO	GND						VDD_IO	XD[8]	NC, CLKMD1	XD[9]
F	BSEL[1]	PIOD[5]/XSDCKE	PIOD[3]/XSDCLK	PIOD[4]/XSDCS_N						GND	XD[7]	XD[6]	XD[5]
E	PIOE[7]/EXINT[2]	BSEL[0]	PIOE[8]/EXINT[3]	PIOE[5]/EXINT[0]						GND	XD[2]	NC, CLKMD0*	XD[4]
D	PIOE[0]/SCLK	PIOE[6]/EXINT[1]	PIOE[9]/EFIQ_N	PIOE[2]/SDO	OSC1_N	PIOA[1]/SOUT	AIN[0]	NC, VREFN*	VDD_IO	GND	VDD_IO	XD[3]	XD[1]
C	TDI	PIOE[1]/SDI	CKO	TMS	CKOE_N	AVDD	AIN[1]	AIN[3]	VDD_CORE	PIOA[5]/DTR	FWR	XD[0]	RESET_N
B	nTRST	TDO	TCK	GND	VDD_IO	PIOA[0]/SIN	VREF, VREFP*	AGND	GND	PIOA[3]/DSR	PIOA[7]/RI	PIOE[4]/SCL	PIOB[7]/SRXD
A	NC, PLLVDD*	NC, PLLGND*	JSEL	DRAME_N	OSC0	TEST	AIN[2]	PIOA[2]/CTS	PIOA[4]/DCD	PIOA[6]/RTS	PIOE[3]/SDA	PIOB[6]/STXD	NC, TEST1*

Notes: NC Pins is not electrically connected inside PKG.

\*A1: NC (ML674001Series), TEST1(ML675001Series)

\*E2: NC(ML674001Series), CLKMD0(ML675001Series)

\*G2: NC (ML674001Series), CLKMD1(ML675001Series)

\*A13: NC (ML674001Series), PLLVDD(ML675001Series)

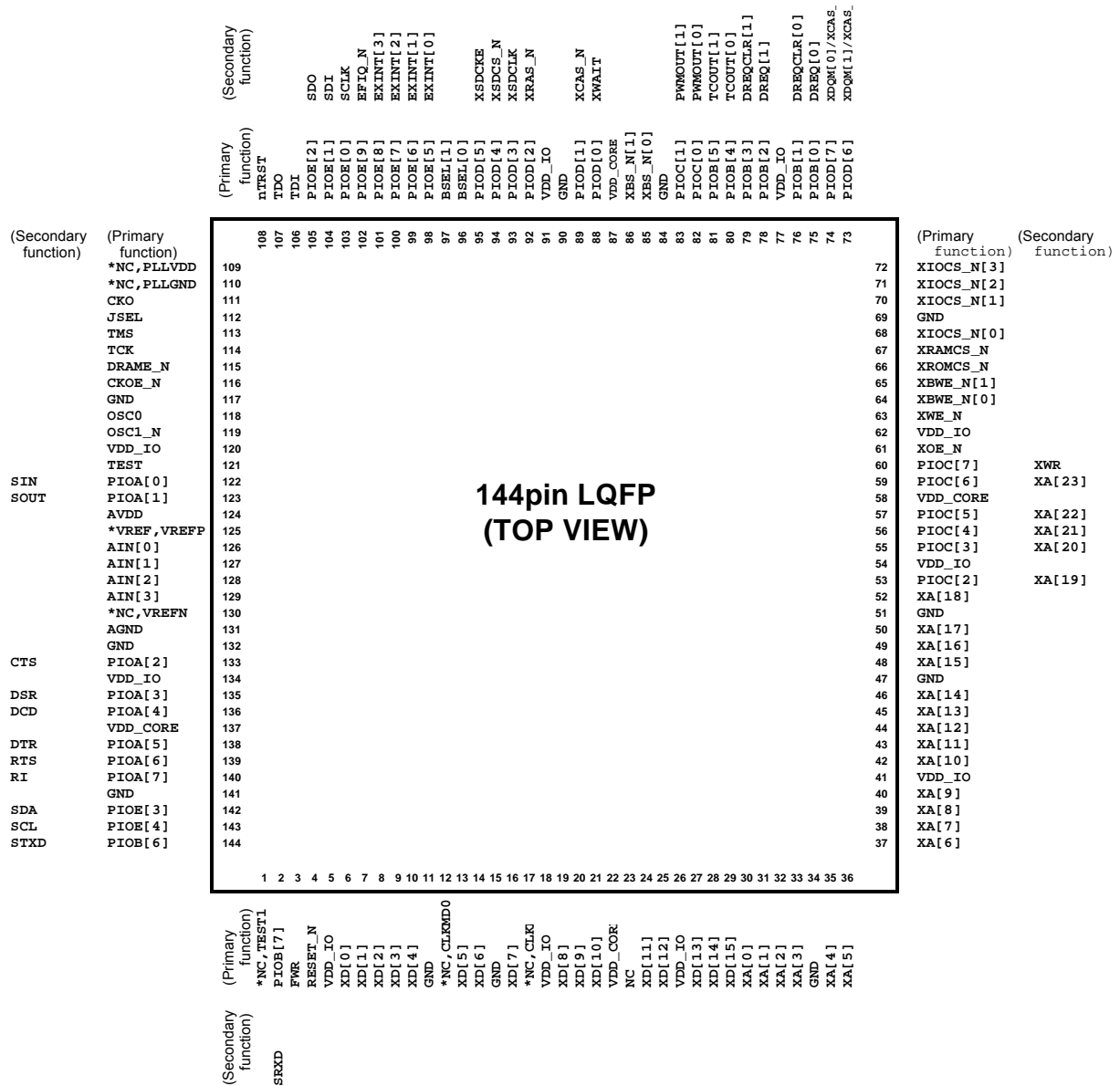
\*A12: NC (ML674001Series), PLLGND(ML675001Series)

\*B7: VREF (ML674001Series), VREFP(ML675001Series)

\*D6: NC (ML674001Series), VREFN(ML675001Series)

**Figure 1.3 Pin Layout(LFBGA)**

1.3.1.2 LQFP



Notes: NC Pins is not electrically connected inside PKG.

- \*1: NC (ML674001Series), TEST1(ML675001Series)
- \*12: NC(ML674001Series), CLKMDO(ML675001Series)
- \*17: NC (ML674001Series), CLKMD1(ML675001Series)
- \*109: NC (ML674001Series), PLLVDD(ML675001Series)
- \*110: NC (ML674001Series), PLLGND(ML675001Series)
- \*125: VREF (ML674001Series), VREFP(ML675001Series)
- \*130: NC (ML674001Series), VREFN(ML675001Series)

Figure 1.4 Pin Layout(LQFP)

## 1.3.2 Pin List

Pin		Primary Function			Secondary Function		
LQFP	BGA	Symbol	I/O	Description	Symbol	I/O	Description
1	A1	NC	—	NC (ML674001Series)	—	—	
		TEST1	I	TEST Mode (ML675001Series)	—	—	
2	B1	PIOB[7]	I/O	General port (with interrupt function)	SRXD	I	SIO receive signal
3	C3	FWR	I	TEST Mode	—	—	
4	C1	RESET_N	I	Reset input	—	—	
5	D3	VDD_IO	VDD	IO power supply	—	—	
6	C2	XD[0]	I/O	External data bus	—	—	
7	D1	XD[1]	I/O	External data bus	—	—	
8	E3	XD[2]	I/O	External data bus	—	—	
9	D2	XD[3]	I/O	External data bus	—	—	
10	E1	XD[4]	I/O	External data bus	—	—	
11	E4	GND	GND	GND	—	—	
12	E2	NC	—	NC (ML674001Series)	—	—	
		CLKMD0	I	Clock mode input (ML675001Series)	—	—	
13	F1	XD[5]	I/O	External data bus	—	—	
14	F2	XD[6]	I/O	External data bus	—	—	
15	F4	GND	GND	GND	—	—	
16	F3	XD[7]	I/O	External data bus	—	—	
17	G2	NC	—	NC (ML674001Series)	—	—	
		CLKMD1	I	Clock mode input (ML675001Series)	—	—	
18	G4	VDD_IO	VDD	I/O power supply	—	—	
19	G3	XD[8]	I/O	External data bus	—	—	
20	G1	XD[9]	I/O	External data bus	—	—	
21	H3	XD[10]	I/O	External data bus	—	—	
22	H4	VDD_CORE	VDD	CORE power supply	—	—	
23	H2	NC	—	NC	—	—	
24	J2	XD[11]	I/O	External data bus	—	—	
25	H1	XD[12]	I/O	External data bus	—	—	
26	J4	VDD_IO	VDD	I/O power supply	—	—	
27	K2	XD[13]	I/O	External data bus	—	—	
28	J1	XD[14]	I/O	External data bus	—	—	
29	J3	XD[15]	I/O	External data bus	—	—	
30	K3	XA[0]	O	External address output	—	—	
31	K1	XA[1]	O	External address output	—	—	
32	L2	XA[2]	O	External address output	—	—	
33	K4	XA[3]	O	External address output	—	—	
34	L1	GND	GND	GND	—	—	
35	M2	XA[4]	O	External address output	—	—	
36	M1	XA[5]	O	External address output	—	—	
37	N1	XA[6]	O	External address output	—	—	
38	N2	XA[7]	O	External address output	—	—	
39	L3	XA[8]	O	External address output	—	—	
40	N3	XA[9]	O	External address output	—	—	

Pin		Primary Function			Secondary Function		
LQFP	BGA	Symbol	I/O	Description	Symbol	I/O	Description
41	L4	VDD_IO	VDD	I/O power supply	—	—	
42	M3	XA[10]	O	External address output	—	—	
43	N4	XA[11]	O	External address output	—	—	
44	L5	XA[12]	O	External address output	—	—	
45	M4	XA[13]	O	External address output	—	—	
46	N5	XA[14]	O	External address output	—	—	
47	K5	GND	GND	GND	—	—	
48	M5	XA[15]	O	External address output	—	—	
49	N6	XA[16]	O	External address output	—	—	
50	M6	XA[17]	O	External address output	—	—	
51	K6	GND	GND	GND	—	—	
52	L6	XA[18]	O	External address output	—	—	
53	M7	PIOC[2]	I/O	General port (with interrupt function)	XA[19]	O	External address output
54	K7	VDD_IO	VDD	I/O power supply	—	—	
55	L7	PIOC[3]	I/O	General port (with interrupt function)	XA[20]	O	External address output
56	N7	PIOC[4]	I/O	General port (with interrupt function)	XA[21]	O	External address output
57	L8	PIOC[5]	I/O	General port (with interrupt function)	XA[22]	O	External address output
58	K8	VDD_CORE	VDD	CORE power supply	—	—	
59	M8	PIOC[6]	I/O	General port (with interrupt function)	XA[23]	O	External address output
60	M9	PIOC[7]	I/O	General port (with interrupt function)	XWR	O	Transfer direction of external bus
61	N8	XOE_N	O	Output enable (excluding SDRAM)	—	—	
62	K9	VDD_IO	VDD	I/O power supply	—	—	
63	M10	XWE_N	O	Write enable	—	—	
64	N9	XBWE_N[0]	O	Byte write enable (LSB)	—	—	
65	L9	XBWE_N[1]	O	Byte write enable (MSB)	—	—	
66	L10	XROMCS_N	O	External ROM chip select	—	—	
67	N10	XRAMCS_N	O	External RAM chip select	—	—	
68	M11	XIOCS_N[0]	O	IO chip select 0	—	—	
69	K10	GND	GND	GND	—	—	
70	N11	XIOCS_N[1]	O	IO chip select 1	—	—	
71	M12	XIOCS_N[2]	O	IO chip select 2	—	—	
72	N12	XIOCS_N[3]	O	IO chip select 3	—	—	
73	N13	PIOD[6]	I/O	General port (with interrupt function)	XDQM[1]/XCA S_N[1]	O	INPUT/OUTPUT mask/CAS (MSB)
74	M13	PIOD[7]	I/O	General port (with interrupt function)	XDQM[0]/XCA S_N[0]	O	INPUT/OUTPUT mask/CAS (LSB)
75	L11	PIOB[0]	I/O	General port (with interrupt function)	DREQ[0]	I	DMA request signal (CH0)
76	L13	PIOB[1]	I/O	General port (with interrupt function)	DREQCLR[0]	O	DREQ Clear Signal (CH0)
77	K11	VDD_IO	VDD	I/O power supply	—	—	
78	L12	PIOB[2]	I/O	General port (with interrupt function)	DREQ[1]	I	DMA request signal (CH1)
79	K13	PIOB[3]	I/O	General port (with interrupt function)	DREQCLR[1]	O	DREQ Clear Signal (CH1)
80	J11	PIOB[4]	I/O	General port (with interrupt function)	TCOUT[0]	O	DMAC Terminal Count (CH0)
81	K12	PIOB[5]	I/O	General port (with interrupt function)	TCOUT[1]	O	DMAC Terminal Count

Pin		Primary Function			Secondary Function		
LQFP	BGA	Symbol	I/O	Description	Symbol	I/O	Description
							(CH1)
82	J13	PIOC[0]	I/O	General port (with interrupt function)	PWMOUT[0]	O	PWM output (CH0)
83	J10	PIOC[1]	I/O	General port (with interrupt function)	PWMOUT[1]	O	PWM output (CH1)
84	J12	GND	GND	GND	—	—	
85	H13	XBS_N[0]	O	External bus byte select (LSB)	—	—	
86	H12	XBS_N[1]	O	External bus byte select (MSB)	—	—	
87	H10	VDD_CORE	VDD	CORE power supply	—	—	
88	H11	PIOD[0]	I/O	General port (with interrupt function)	XWAIT	I	Wait input signal for I/O Bank 0/1/2/3
89	G12	PIOD[1]	I/O	General port (with interrupt function)	XCAS_N	O	Column address strobe (SDRAM)
90	G10	GND	GND	GND	—	—	
91	G11	VDD_IO	VDD	I/O power supply	—	—	
92	G13	PIOD[2]	I/O	General port (with interrupt function)	XRAS_N	O	Row address strobe (SDRAM/EDO-DRAM)
93	F11	PIOD[3]	I/O	General port (with interrupt function)	XSDCLK	O	Clock for SDRAM
94	F10	PIOD[4]	I/O	General port (with interrupt function)	XSDCS_N	O	Chip select for SDRAM
95	F12	PIOD[5]	I/O	General port (with interrupt function)	XSDCKE	O	Clock enable (SDRAM)
96	E12	BSEL[0]	I	Select boot device	—	—	
97	F13	BSEL[1]	I	Select boot device	—	—	
98	E10	PIOE[5]	I/O	General port (with interrupt function)	EXINT[0]	I	Interrupt input
99	D12	PIOE[6]	I/O	General port (with interrupt function)	EXINT[1]	I	Interrupt input
100	E13	PIOE[7]	I/O	General port (with interrupt function)	EXINT[2]	I	Interrupt input
101	E11	PIOE[8]	I/O	General port (with interrupt function)	EXINT[3]	I	Interrupt input
102	D11	PIOE[9]	I/O	General port (with interrupt function)	EFIQ_N	I	FIQ input
103	D13	PIOE[0]	I/O	General port (with interrupt function)	SCLK	I/O	SSIO clock
104	C12	PIOE[1]	I/O	General port (with interrupt function)	SDI	I	SSIO Serial Data In
105	D10	PIOE[2]	I/O	General port (with interrupt function)	SDO	O	SSIO Serial Data Out
106	C13	TDI	I	JTAG Data Input	—	—	
107	B12	TDO	O	JTAG data out	—	—	
108	B13	nTRST	I	JTAG reset	—	—	
109	A13	NC	—	NC (ML674001Series)	—	—	
		PLLVD	VDD	PLL power supply (ML675001Series)	—	—	
110	A12	NC	—	NC (ML674001Series)	—	—	
		PLLGND	GND	PLL GND (ML675001Series)	—	—	
111	C11	CKO	O	Clock output	—	—	
112	A11	JSEL	I	JTAG select	—	—	
113	C10	TMS	I	JTAG mode select	—	—	
114	B11	TCK	I	JTAG clock	—	—	
115	A10	DRAME_N	I	DRAM enable	—	—	
116	C9	CKOE_N	I	Clock out enable	—	—	
117	B10	GND	GND	GND	—	—	
118	A9	OSC0	I	Oscillation input pin	—	—	
119	D9	OSC1_N	O	Oscillation output pin	—	—	
120	B9	VDD_IO	VDD	IO power supply	—	—	

Pin		Primary Function			Secondary Function		
LQFP	BGA	Symbol	I/O	Description	Symbol	I/O	Description
121	A8	TEST	I	Test mode input	—	—	
122	B8	PIOA[0]	I/O	General port (with interrupt function)	SIN	I	UART Serial Data In
123	D8	PIOA[1]	I/O	General port (with interrupt function)	SOUT	O	UART Serial Data Out
124	C8	AVDD	VDD	A/D CONVERTER power supply	—	—	
125	B7	VREF	I	A/D CONVERTER Reference voltage (ML674001Series)	—	—	
		VREFP	I	A/D CONVERTER Reference voltage (ML675001Series)	—	—	
126	D7	AIN[0]	I	A/D CONVERTER analog input port	—	—	
127	C7	AIN[1]	I	A/D CONVERTER analog input port	—	—	
128	A7	AIN[2]	I	A/D CONVERTER analog input port	—	—	
129	C6	AIN[3]	I	A/D CONVERTER analog input port	—	—	
130	D6	NC	—	NC (ML674001Series)	—	—	
		VREFN	GND	A/D CONVERTER Reference GND (ML675001Series)			
131	B6	AGND	GND	GND for A/D CONVERTER	—	—	
132	B5	GND	GND	GND	—	—	
133	A6	PIOA[2]	I/O	General port (with interrupt function)	CTS	I	UART Clear To Send
134	D5	VDD_IO	VDD	IO power supply	—	—	
135	B4	PIOA[3]	I/O	General port (with interrupt function)	DSR	I	UART Set Ready
136	A5	PIOA[4]	I/O	General port (with interrupt function)	DCD	I	UART Carrier Detect
137	C5	VDD_CORE	VDD	CORE power supply	—	—	
138	C4	PIOA[5]	I/O	General port (with interrupt function)	DTR	O	UART Data Terminal Ready
139	A4	PIOA[6]	I/O	General port (with interrupt function)	RTS	O	UART Request To Send
140	B3	PIOA[7]	I/O	General port (with interrupt function)	RI	I	UART Ring Indicator
141	D4	GND	GND	GND	—	—	
142	A3	PIOE[3]	I/O	General port (with interrupt function)	SDA	I/O	I2C Data In/Out
143	B2	PIOE[4]	I/O	General port (with interrupt function)	SCL	O	I2C Clock out
144	A2	PIOB[6]	I/O	General port (with interrupt function)	STXD	O	SIO send data output



1.3.3 Pin Descriptions

Pin Name	I/O	Description	Primary/ Secondary	Logic
<b>System</b>				
RESET_N	I	Reset input	—	Negative
BSEL[1:0]	I	Boot device select signal BSEL[1] BSEL[0] Boot device 0 0 Internal Flash 0 1 External ROM 1 * Boot mode The selected device is mapped to BANK0 (0x0000_0000 - 0x07 FF_FFFF) after reset.	—	Positive
CLKMD[1:0]	I	Clock mode input.	—	Positive
OSC0	I	Crystal connection or external clock input. Connect a crystal , if used, to OSC0 and OSC1_N. (ML674001Series:16 MHz to 33 MHz) (ML675001Series:5 MHz to 14 MHz) It is also possible to input a direct clock. (ML674001Series:1 MHz to 33 MHz) (ML675001Series:5 MHz to 14 MHz, 20MHz to 56MHz)	—	—
OSC1_N	O	Oscillation output pin When not using a crystal, leave this pin unconnected.	—	—
CKO	O	Clock out	—	—
CKOE_N	I	Clock out enable	—	Negative
<b>Debugging support.</b>				
TCK	I	Debugging pin. Normally connect to ground level.	—	—
TMS	I	Debugging pin. Normally drive at High level.	—	Positive
nTRST	I	Debugging pin. Normally connect to ground level.	—	Negative
TDI	I	Debugging pin. Normally drive at High level.	—	Positive
TDO	O	Debugging pin. Normally leave open.	—	Positive
<b>General-purpose I/O ports</b>				
PIOA[7:0]	I/O	General-purpose port. Not available for use as port pins when secondary functions are in use.	Primary	Positive
PIOB[7:0]	I/O	General-purpose port. Not available for use as port pins when secondary functions are in use.	Primary	Positive
PIOC[7:0]	I/O	General-purpose port. Not available for use as port pins when secondary functions are in use.	Primary	Positive
PIOD[7:0]	I/O	General-purpose port. Not available for use as port pins when secondary functions are in use. Note that enabling DRAM controller with DRAME_N inputs permanently configures PIOD[7:0] for their secondary functions, making them unavailable for use as port pins.	Primary	Positive
PIOE[9:0]	I/O	General-purpose port. Not available for use as port pins when secondary functions are in use.	Primary	Positive

Pin Name	I/O	Description	Primary / Secondary	Logic
<b>External Bus</b>				
XA[23:19]	O	Address bus to external RAM, external ROM, external I/O banks, and external DRAM. After a reset, these pins are configured for their primary function (PIOC[6:2]).	Secondary	Positive
XA[18:0]	O	Address bus to external RAM, external ROM, external I/O banks, and external DRAM.	—	Positive
XD[15:0]	I/O	Data bus to external RAM, external ROM, external I/O banks, and external DRAM.	—	Positive
<b>External bus control signals (ROM/SRAM/IO)</b>				
XROMCS_N	O	ROM bank chip select	—	Negative
XRAMCS_N	O	SRAM bank chip select	—	Negative
XIOCS_N[0]	O	IO chip select 0	—	Negative
XIOCS_N[1]	O	IO chip select 1	—	Negative
XIOCS_N[2]	O	IO chip select 2	—	Negative
XIOCS_N[3]	O	IO chip select 3	—	Negative
XOE_N	O	Output enable/ Read enable	—	Negative
XWE_N	O	Write enable	—	Negative
XBS_N[1:0]	O	Byte select: XBS_N[1] is for MSB, XBS_N[0] is for LSB	—	Negative
XBWE_N[0]	O	LSB Write enable	—	Negative
XBWE_N[1]	O	MSB Write enable	—	Negative
XWR	O	Data transfer direction for external bus, used when connecting to Motorola I/O devices. This represent the secondary function of pin PIOC[7]. For ML675001 series, this pin is available for IO Bank0/1. L: read, H: write	Secondary	—
XWAIT	I	External I/O bank 0/1/2/3 WAIT signal. This input permits access to devices slower than register settings.	Secondary	Positive
<b>External bus control signals (DRAM)</b>				
XRAS_N	O	Row address strobe. Used for both EDO DRAM and SDRAM	Secondary	Negative
XCAS_N	O	Column address strobe signal (SDRAM)	Secondary	Negative
XSDCLK	O	SDRAM clock (same frequency as internal HCLK)	Secondary	—
XSDCKE	O	Clock enable (SDRAM)	Secondary	—
XSDCS_N	O	Chip select (SDRAM)	Secondary	Negative
XDQM[1]/XCAS_N[1]	O	Connected to SDRAM: DQM (MSB) Connected to EDO DRAM: column address strobe signal (MSB)	Secondary	Positive/ Negative
XDQM[0]/XCAS_N[0]	O	Connected to SDRAM: DQM (LSB) Connected to EDO DRAM: column address strobe signal (LSB)	Secondary	Positive/ Negative

Pin Name	I/O	Description	Primary / Secondary	Logic
<b>DMA control signals</b>				
DREQ[0]	I	Ch 0 DMA request signal, used when DMA controller configured for DREQ type	Secondary	Positive
DREQCLR[0]	O	Ch 0 DREQ signal clear request. The DMA device responds to this output by negating DREQ.	Secondary	Positive
TCOUT[0]	O	Indicates to Ch 0 DMA device that last transfer has started.	Secondary	Positive
DREQ[1]	I	Ch 1 DMA request signal, used when DMA controller configured for DREQ type	Secondary	Positive
DREQCLR[1]	O	Ch 1 DREQ signal clear request. The DMA device responds to this output by negating DREQ.	Secondary	Positive
TCOUT[1]	O	Indicates to Ch 1 DMA device that last transfer has started	Secondary	Positive
<b>UART</b>				
SIN	I	SIO receive signal	Secondary	Positive
SOUT	O	SIO transmit signal	Secondary	Positive
CTS	I	Clear To Send. Indicates that modem or data set is ready to transfer data. Bit 4 in modem status register reflects this input.	Secondary	Negative
DSR	I	Data Set Ready. Indicates that modem or data set is ready to establish a communications link with UART. Bit 5 in modem status register reflects this input.	Secondary	Negative
DCD	I	Data Carrier Detect. Indicates that modem or data set has detected data carrier signal. Bit 7 in modem status register reflects this input. Data Carrier Detect	Secondary	Negative
DTR	O	Data Terminal Ready. Indicates that UART is ready to establish a communications link with modem or data set. Bit 0 in modem control register controls this output.	Secondary	Negative
RTS	O	Request To Send. Indicates that UART is ready to transfer data to modem or data set. Bit 1 in modem control register controls this output.	Secondary	Negative
RI	I	Ring Indicator. Indicates that modem or data set has received telephone ring indicator. Bit 6 in modem status register reflects this input.	Secondary	Negative

Pin Name	I/O	Description	Primary / Secondary	Logic
<b>SIO</b>				
STXD	O	SIO transmit signal	Secondary	Positive
SRXD	I	SIO receive signal	Secondary	Positive
<b>I2C</b>				
SDA	I/O	I2C Data. This pin operates as NMOS open drain connect pull-up resistor.	Secondary	Positive
SCL	O	I2C Clock. This pin operates as NMOS open drain connect pull-up resistor.	Secondary	Positive
<b>Synchronous SIO</b>				
SCLK	I/O	Serial clock	Secondary	—
SDI	I	Serial receive data	Secondary	Positive
SDO	O	Serial transmit data	Secondary	Positive
<b>PWM signals</b>				
PWMOUT[0]	O	PWM output of CH0	Secondary	Positive
PWMOUT[1]	O	PWM output of CH1	Secondary	Positive
<b>Analog-to-digital converter</b>				
AIN[0]	I	Ch0 analog input	—	—
AIN[1]	I	Ch1 analog input	—	—
AIN[2]	I	Ch2 analog input	—	—
AIN[3]	I	Ch3 analog input	—	—
VREF	I	Analog-to-digital converter convert reference voltage (VDD) (ML674001Series only)	—	—
VREFP	I	Analog-to-digital converter convert reference voltage (VDD) (ML675001Series only)	—	—
VREFN	I	Analog-to-digital converter convert reference voltage (GND)(ML675001Series only)	—	—
AVDD		Analog-to-digital converter power supply	—	—
AGND		Analog-to-digital converter ground	—	—
<b>Interrupt signals</b>				
EXINT[3:0]	I	External interrupt input signals	Secondary	Positive / Negative
EFIQ_N	I	External fast interrupt input signal. Interrupt controller connects this to CPU FIQ input.	Secondary	Negative
<b>MODE configuration</b>				
DRAME_N	I	DRAM enable mode	—	Negative
TEST	I	Test mode	—	Positive
TEST1	I	Test mode (ML675001Series only)	—	Positive
FWR	I	Test mode	—	Positive
JSEL	I	JTAG select signal L: onboard debug H: boundary scan	—	—
<b>Power supplies</b>				
VDD_CORE		Core power supply	—	—
VDD_IO		I/O power supply	—	—
GND		GND for core and I/O	—	—
PLLVD		PLL power supply (ML675001Series Only)	—	—
PLLGND		PLL power supply (ML675001Series Only)	—	—

## 1.3.4 Pin States

Table 1.1 summarizes the output states for output and I/O pins during a reset; Table 1.2, for STANDBY mode.

**Table 1.1 Output States During Reset**

Pin Name	Reset with DRAM controller disabled	Reset with DRAM controller enabled
PIOA[7:0]	High impedance	High impedance
PIOB[7:0]	High impedance	High impedance
PIOC[7:0]	High impedance	High impedance
PIOD[7](XDQM[0]/XCAS_N[0])	High impedance	High level
PIOD[6](XDQM[1]/XCAS_N[1])	High impedance	High level
PIOD[5](XSDCKE)	High impedance	High level
PIOD[4](XSDCS_N)	High impedance	High level
PIOD[3](XSDCLK)	High impedance	Low level
PIOD[2](XRAS_N)	High impedance	High level
PIOD[1](XCAS_N)	High impedance	High level
PIOD[0](XWAIT)	High impedance	High impedance
PIOE[9:0]	High impedance	High impedance
XD[15:0]	High impedance	High impedance
XA[18:0]	Low level	Low level
XOE_N	High level	High level
XWE_N	High level	High level
XBWE_N[0]	High level	High level
XBWE_N[1]	High level	High level
XROMCS_N	High level	High level
XRAMCS_N	High level	High level
XIOCS_N[3:0]	High level	High level
XBS_N[1:0]	High level	High level
TDO	High impedance	High impedance

DRAME\_N pin input during a reset switches the DRAM controller on and off.

- DRAM controller disabled: DRAME\_N = High
- DRAM controller enabled: DRAME\_N = Low

TDO pin is always high-impedance outside JTAG mode.

**Table 1.2 Output States in STANDBY Mode**

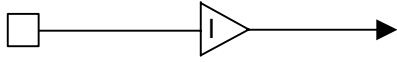
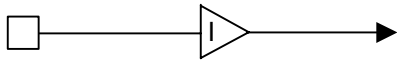
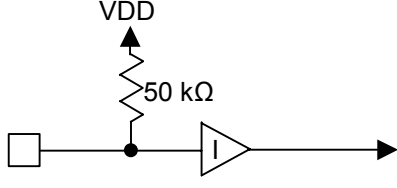
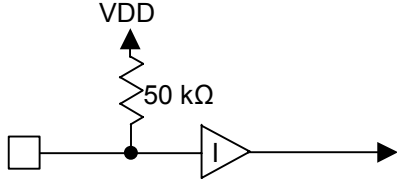
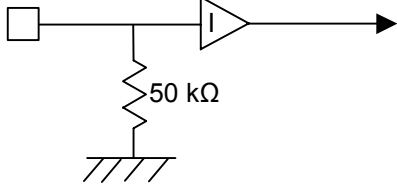
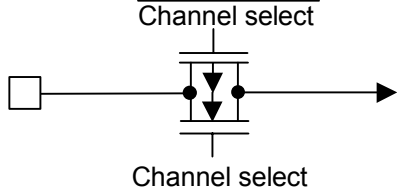
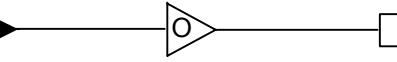
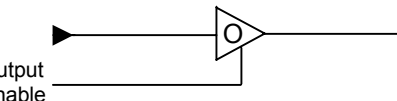
Pin Name	STANDBY Mode
PIOA[7:0] (primary function)	No change
PIOA[7:0] (secondary function)	No change
PIOB[7:0] (primary function)	No change
PIOC[7:0] (primary function)	No change
PIOC[6:2] (secondary function: XA[23:19])	Low level
PIOC[7] (secondary function: XWR)	High level
PIOD[7:0] (primary function)	No change
PIOD[7] (secondary function: XDQM[0]/XCAS_N[0])	High level
PIOD[6] (secondary function: XDQM[1]/XCAS_N[1])	High level
PIOD[5] (secondary function: XSDCKE)	High level
PIOD[4] (secondary function: XSDCS_N)	High level
PIOD[3] (secondary function: XSDCLK)	Low level
PIOD[2] (secondary function: XRAS_N)	High level
PIOD[1] (secondary function: XCAS_N)	High level
PIOE[9:0] (primary function)	No change
XD[15:0]	High impedance
XA[18:0]	Low level
XOE_N	High level
XWE_N	High level
XBWE_N[0]	High level
XBWE_N[1]	High level
XROMCS_N	High level
XRAMCS_N	High level
XIOCS_N[3:0]	High level
XBS_N[1:0]	High level
TDO	High impedance

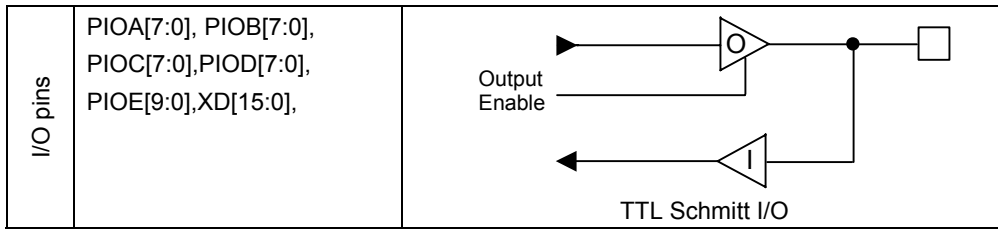
TDO pin is always high-impedance outside JTAG mode.

1.3.5 Pin Structure and Treatment

Table 1.3 summarizes the structures.

**Table 1.3 ML675001 Series Pin Structures**

Input pins	EFIQ_N, EXINT3,2,1,0, BSEL[1:0], TCK, CLKMD[1:0](ML675001Seir es Only)	 TTL input
	DRAME_N, CKOE_N, TEST, nTRST, TEST1(ML675001Seires Only)	 TTL Schmitt input
	TDI, TMS	 TTL input with 50 kΩ pull-up resistance
	RESET_N	 TTL Schmitt input with 50 kΩ pull-up resistance
	JSEL, FWR	 TTL input with 50 kΩ pull-down resistance
	AIN3 to AIN0	 Analog-to-digital converter input
Output pins	XA[18:0], XWE_N, XOE_N, XBWE_N[1:0], XROMCS_N, XRAMCS_N, XIOCS_N[3:0], XBS_N[1:0], CKO	
	TDO	





### 1.3.6 Treatment of Unused Pins

Table 1.4 specifies the treatment of unused pins.

**Table 1.4 Treatment of Unused Pins**

Pin Name	Treatment
TDI, TMS	High level
nTRST, TCK	Low level
TDO	Open
PIOA[7:0], PIOB[7:0] PIOC[7:0],PIOD[7:0],PIOE[9:0]	Configured for input: High or Low level Configured for output: open
AVDD,	VDD_IO
VREF	VDD_IO (ML674001 Series Only)
VREFP	VDD_IO (ML675001 Series Only)
VREFN	GND (ML675001 Series Only)
AIN3 to AIN0	VREF or AGND
AGND	GND
EXINT3,2,1,0	High level (Note)
EFIQ_N	High level
XA[18:0]	Open
XWE_N, XBWE_N[1:0], XRAMCS_N, XIOCS_N[3:0], XBS_N[1:0]	Open
DRAME_N	DRAM function enabled: Low level DRAM function disabled: High level
CKOE_N	CKO output enabled: Low level CKO output disabled: High level
TEST	Low level
TEST1	Low level (ML675001 Series Only)
JSEL	Low level
FWR	Low level
CKO	Open

The EXINT3,2,1,0 pins default to Low level interrupts, so drive them at High level immediately after a reset to prevent spurious interrupt requests. If the user application system subsequently switches to High level interrupts, drive them at Low level.

## *Chapter 2*

**CPU**

---

## Chapter 2 CPU

### 2.1 Overview

This LSI features the ARM7TDMI core, a RISC CPU developed by Advanced RISC Machines Limited (ARM). The ARM7TDMI core allows user application programs to freely switch, as necessary, between the ARM state running a 32-bit instruction set and the THUMB state running an alternate 16-bit set.

### 2.2 CPU Operation States

The CPU operates in either the ARM or THUMB state. Switching states does not affect the CPU operating mode or register contents.

- ARM state  
The CPU executes ARM instructions aligned at 32-bit word boundaries.
- THUMB state  
The CPU executes THUMB instructions aligned at 16-bit halfword boundaries.

#### 2.2.1 State Transitions

The CPU changes processor states in two ways.

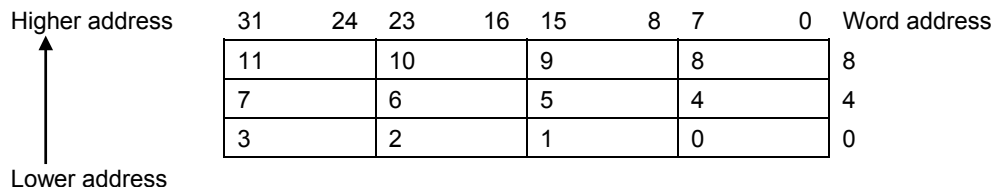
- BX instruction  
The program changes between ARM and THUMB states with a Branch and Exchange (BX) instruction.
- Exception processing  
The CPU always processes exceptions in the ARM state. If it was in the THUMB state when it accepted the exception, it automatically returns to THUMB state when it returns from the exception handler.

### 2.3 Address Space

This LSI uses 32-bit addresses to access a flat 4-gigabyte address space. This single address space contains both instructions and data.

## 2.4 Memory Format

Memory consists of bytes numbered sequentially from zero. Bytes 0 to 3 contain the first word of data; bytes 4 to 7, the second. The bytes making up a word are stored in little-endian format.



- The lowest byte is stored at the lowest address.
- The address of a word is the address of lowest byte.

**Figure 2.1 Little-Endian Byte Addresses Within Words**

### Note

The CPU architecture supports both big- and little-endian byte orders, but this LSI uses only the little-endian order.

## 2.5 Instruction Length

The instruction length is either 32 bits (ARM state) or 16 (THUMB state).

## 2.6 Data Types

This LSI supports the data types byte (8 bits), halfword (16 bits), and word (32 bits). Words must be aligned at 4-byte boundaries; halfwords, at 2-byte ones.

## 2.7 Processor Modes

This LSI supports seven processor modes.

User (usr):	Normal program execution state
FIQ (fiq):	High-speed interrupt handling
IRQ (irq):	General-purpose interrupt handling
Supervisor (svc):	Protected mode for the operating system
Abort (abt):	Prefetch abort (data or instruction) processing
System (sys):	Privileged user mode for the operating system
Undefined (und):	Undefined instruction exception processing

Mode changes are either under software control or in response to an interrupt or exception.

Most application programs run primarily in user mode, switching to non-user, privileged modes only in interrupt and exception handlers or to access protected resources.

## 2.8 Registers

The CPU has a total of 31 general-purpose 32-bit registers and six status registers. Not all registers are available simultaneously, however. The registers which a program can access depends on the CPU operation state and operating mode.

### 2.8.1 ARM State Registers

The ARM state provides access to 16 general-purpose registers and the current program status (CPSR) register at all times.

Non-user, privileged modes access their own private register banks and usually a private save program status register (SPSR).

Figure 2.2 indicates these private registers with shaded triangles.

The ARM state provides direct access to the sixteen registers R0 to R15. All but R15 are general-purpose registers.

There is also a seventeenth register, CPSR, which contains control and status information.

- **Register 14**  
This register functions as the subroutine link register because the branch with link (BL) instruction automatically copies the contents of R15 here. When not used for this purpose, this register is available for use as a general-purpose register.  
The corresponding bank registers R14\_svc, R14\_irq, R14\_fiq, R14\_abt, and R14\_und hold the return address for an interrupt or exception handler or a BL instruction executed within such a handler.
- **Register 15**  
This register is reserved for use as the program counter (PC). In the ARM state, the program counter is in bits 31 to 2, and bits 0 and 1 are always zero. In the THUMB state, the program counter is in bits 31 to 1, and bit 0 is always zero.
- **Register 16**  
This is the current program status register (CPSR). It contains the four condition code flags and bits specifying the current operating mode.

The FIQ mode has seven bank registers (R8\_fiq to R14\_fiq) mapped to R8 to R14. Having these registers available means that many FIQ handlers do not have to save registers to the stack.

The User, IRQ, Supervisor, Abort, and Undefined modes have two bank registers mapped to R13 and R14. These modes use these registers as a private stack pointer and a link register.

ARM State General Registers and Program Counter

System & User	FIQ	Supervisor	Abort	IRQ	Undefined
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8_fiq	R8	R8	R8	R8
R9	R9_fiq	R9	R9	R9	R9
R10	R10_fiq	R10	R10	R10	R10
R11	R11_fiq	R11	R11	R11	R11
R12	R12_fiq	R12	R12	R12	R12
R13	R13_fiq	R13_svc	R13_abt	R13_irq	R13_und
R14	R14_fiq	R14_svc	R14_abt	R14_irq	R14_und
R15(PC)	R15(PC)	R15(PC)	R15(PC)	R15(PC)	R15(PC)

ARM State Program Status Registers

CPSR	CPSR SPSR_fiq	CPSR SPSR_svc	CPSR SPSR_abt	CPSR SPSR_irq	CPSR SPSR_und
------	------------------	------------------	------------------	------------------	------------------

▲ = Banked register

Figure 2.2 ARM State Registers

## 2.8.2 THUMB State Registers

The THUMB state uses a subset of the ARM state register set. The programmer has direct access to eight general-purpose registers (R0 to R7), the program counter (PC), stack pointer (SP), link register (LR), and CPSR. The privileged user modes access three private registers: stack pointer, link register, and save program status register (SPSR).

Figure 2.3 summarizes these registers.

THUMB State General Registers and Program Counter

System & User	FIQ	Supervisor	Abort	IRQ	Undefined
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
SP	SP_fiq	SP_svc	SP_abt	SP_irq	SP_und
LR	LR_fiq	LR_svc	LR_abt	LR_irq	LR_und
PC	PC	PC	PC	PC	PC

THUMB State Program Status Registers

CPSR	CPSR SPSR_fiq	CPSR SPSR_svc	CPSR SPSR_abt	CPSR SPSR_irq	CPSR SPSR_und
------	------------------	------------------	------------------	------------------	------------------

▲ = Banked register

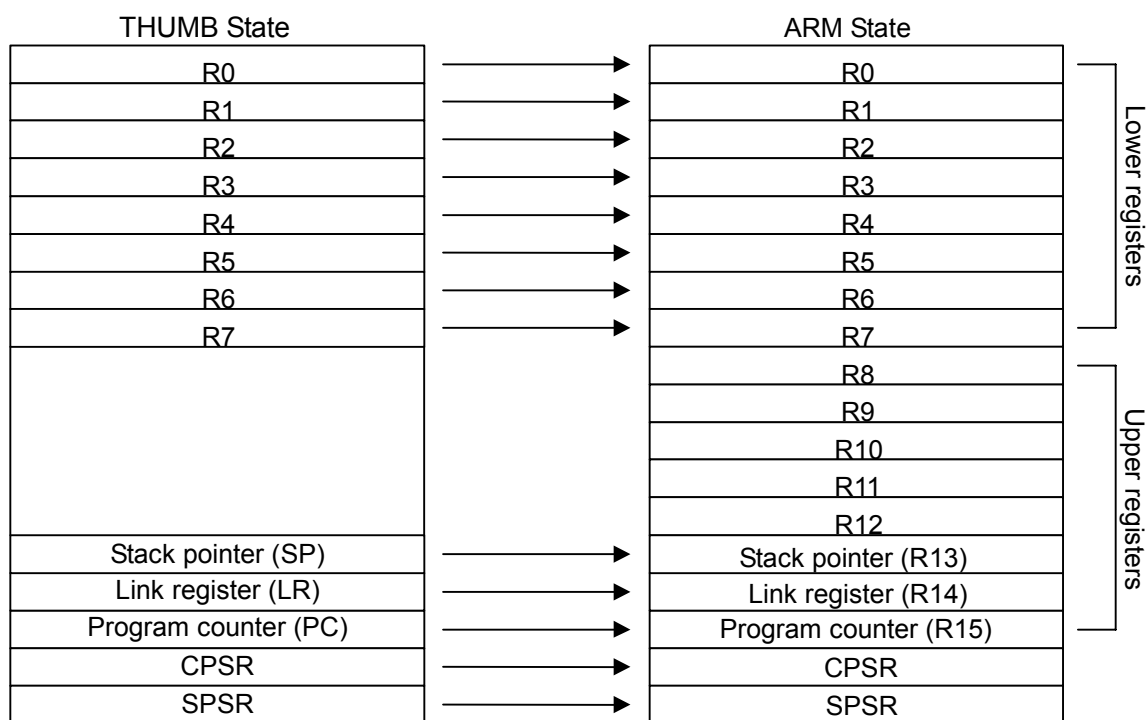
Figure 2.3 THUMB State Registers

### 2.8.3 Relationships Between ARM and THUMB State Registers

The THUMB state registers have the following relationships with the ARM state ones.

- The THUMB state general-purpose registers R0 to R7 are the same registers as their ARM state counterparts.
- The THUMB state status registers CPSR and SPSR are the same registers as their ARM state counterparts.
- The THUMB state stack pointer (SP) corresponds to the ARM state R13 register. The THUMB state link register (LR) corresponds to the ARM state R14 register.
- The THUMB state program counter (PC) corresponds to the ARM state R15 register.

Figure 2.4 summarizes these relationships.



**Figure 2.4 Mapping THUMB State Registers to ARM State Ones**

### 2.8.4 Accessing Upper Registers from THUMB State

The THUMB state does not include the upper registers (R8 to R15) as part of the standard register set. Assembly language programs can, however, obtain limited access to them for use as high-speed temporary storage. There are special MOV instructions for transferring data from a lower register to an upper one and in the reverse direction. The CMP and ADD instructions also accept an upper register for comparison with or addition to a lower register.



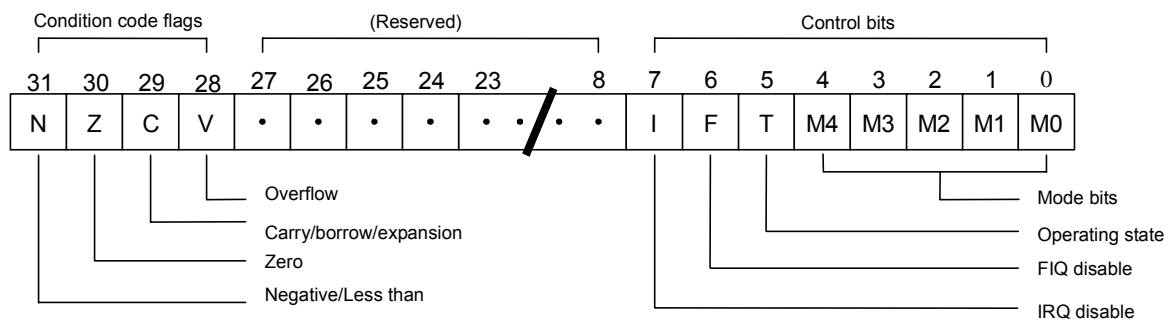
## 2.9 Program Status Registers

The CPU has six status registers: the current program status register (CPSR) and five save program status registers (SPSRs).

These registers have the following functions.

- Status flags indicate the results of the last arithmetic instruction executed by the ALU.
- Control bits enable and disable interrupts.
- Mode bits specify the CPU operating mode.

Figure 2.5 gives the bit positions in a program status register.



**Figure 2.5 Program Status Register Format**

### 2.9.1 Condition Code Flags

The N, Z, C, and V bits are condition code flags indicating the results of arithmetic and bitwise operations. The CPU tests these to determine whether to execute instructions. All ARM state instructions feature conditional execution. The only THUMB state instructions with it are conditional branches.

### 2.9.2 Control Bits

A program status register's lowest eight bits (I, F, T, and M[4:0]) are collectively called control bits. These change in response to exceptions. Software can manipulate them in privileged user modes.

- **T bit**  
This bit specifies the operation state: “1” for THUMB; “0” for ARM.  
Note that modifying this bit in software risks sending the CPU into an unpredictable state.
- **Interrupt disable bits**  
Setting the I or F bit to “1” disables IRQ or FIQ interrupts, respectively.
- **Mode bits**  
These five bits specify the CPU operating mode, as shown in Table 2.1.  
Note that not all combinations produce valid operating modes. Specifying one not listed risks sending the CPU into an unpredictable state.

**Table 2.1 PSR Mode Bits**

<b>M[4:0]</b>	<b>Mode</b>	<b>THUMB State Registers</b>	<b>ARM State Registers</b>
10000	User	R7..R0, LR, SP, PC, CPSR	R14..R0, PC, CPSR
10001	FIQ	R7..R0, LR_fiq, SP_fiq, PC, CPSR, SPSR_fiq	R7..R0, R14_fiq..R8_fiq, PC, CPSR, SPSR_fiq
10010	IRQ	R7..R0, LR_irq, SP_irq, PC, CPSR, SPSR_irq	R12..R0, R14_irq..R13_irq, PC, CPSR, SPSR_irq
10011	Supervisor	R7..R0, LR_svc, SP_svc, PC, CPSR, SPSR_svc	R12..R0, R14_svc..R13_svc, PC, CPSR, SPSR_svc
10011	Abort	R7..R0, LR_abt, SP_abt, PC, CPSR, SPSR_abt	R12..R0, R14_abt..R13_abt, PC, CPSR, SPSR_abt
11011	Undefined	R7..R0, LR_und, SP_und, PC, CPSR, SPSR_und	R12..R0, R14_und..R13_und, PC, CPSR
11111	System	R7..R0, LR, SP, PC, CPSR	R14..R0, PC, CPSR

### 2.9.3 Reserved Bits

Remaining program status register bits are reserved. Be careful not to modify their contents when changing the condition code flags or control bits.

## 2.10 Instruction Set Features

There are two instruction sets: 32-bit ARM instructions and 16-bit THUMB ones.

### 2.10.1 ARM Instruction Set

The ARM instruction set contains the following six main instruction types.

- Branch instructions
- Data processing instructions
- Status register transfer instructions
- Load/store instructions
- Coprocessor instructions
- Exception generation instructions

Most data processing instructions and one class of coprocessor instructions sometimes update the four condition code flags (sign, zero, carry, and overflow) in the current program status (CPSR).

Almost all ARM instructions include a 4-bit condition field. One such code (“Always”) specifies unconditional execution. The others specify execution only if the corresponding condition is valid at the start of instruction execution. There is no execution if the condition is not met.

There are 14 such codes for testing the following.

- Equality and inequality
- Inequality (<, <=, >, or >=) after signed or unsigned arithmetic
- Individual condition code flags

### 2.10.2 THUMB Instruction Set

Each 16-bit THUMB instruction has a corresponding 32-bit ARM instruction with the same effect on the processor model. The twin design goals here are to boost performance in user applications using a 16 bits wide (or smaller) memory data bus and to increase code density.

THUMB instructions retain the same 32-bit architecture as ARM instructions--in particular, 32-bit arithmetic and 32-bit addresses for data access instructions and instruction fetch. They are, however, subject to a few access limitations. THUMB instructions share the first eight general-purpose registers, R0 to R7, with ARM instructions. The upper eight, R8 to R15, are generally off limits, but certain THUMB instructions have access to the program counter (ARM register 15), link register (ARM register 14), and stack pointer (ARM register 13).

ARM register 15 holds the program counter in bits 31 to 1. Bit 0 returns “0” for reads. Writes to bit 0 are ignored.

There are no THUMB analogs for the ARM instructions (MSR and MRS) for direct transfers to and from current program status register (CPSR) and saved program status registers (SPSR).

## 2.11 Addressing Modes

This CPU provides a rich selection of addressing modes for accessing memory.

### 2.11.1 Load/Store Instructions

Load/store instructions offer three basic addressing modes, each specifying a base register and an offset.

- **Offset addressing mode**  
The memory address consists of the offset added to or subtracted from the base register contents.
- **Preindex addressing mode**  
The memory address is the same as for the offset addressing mode. This address then overwrites the base register contents.
- **Postindex addressing mode**  
The memory address is the base register contents. The hardware then updates the base register by adding or subtracting the offset.

The offset is either an immediate value or the contents of an index register. There are options for shifting the contents of an index register before the addition or subtraction.

### 2.11.2 Multiple Load/Store Instructions

Multiple load instructions load two or more general-purpose registers from memory. Multiple store instructions store two or more general-purpose registers in memory.

These instructions offer four addressing modes.

- **Increment after (IA)**  
The base register contents are incremented after the transfer.
- **Increment before (IB)**  
The base register contents are incremented before the transfer.
- **Decrement after (DA)**  
The base register contents are decremented after the transfer.
- **Decrement before (DB)**  
The base register contents are decremented before the transfer.

## 2.12 Exceptions

An exception indicates the need to temporarily suspend normal program flow--to process an interrupt request from a peripheral. The CPU must therefore save the current CPU state before switching to the exception handler and subsequently restore it when the handler returns.

If there are simultaneous exceptions, a fixed priority system determines the order in which they are accepted. For further details, see Section 2.12.10 "Exception Priority Order."

### 2.12.1 Switching to Exception Handler

Switching to an exception handler involves the following operations.

1. The CPU saves the return address in the corresponding link register (R14\_XXX).  
This return address is the current program counter (PC) contents plus an offset (2, 4, or 8) depending on the exception type and the CPU state at the time of the exception. For further details, see Table 2.2 "Returning from Exception Handlers."  
In the ARM state, the return address is simply the address of the next instruction. In the THUMB state, however, the offset sometimes changes to allow the exception handler to return with the same instruction--`MOVS PC, R14_svc` for a software interrupt, for example--regardless of the original state.
2. The CPU copies the current program status register (CPSR) contents to the corresponding save program status register (SPSR\_XXX).
3. The CPU sets the CPSR mode bits to the exception handler's operating mode.  
To prevent multiple exceptions from interfering with proper system control, the hardware also sets both interrupt disable bits to "1."
4. The CPU loads the exception handler's entry point from the corresponding exception vector into PC to transfer control.

If the CPU is in the THUMB state, this step simultaneously switches it to the ARM state.

### 2.12.2 Returning from Exception Handler

Returning from an exception handler involves the following operations.

1. The CPU subtracts the specified offset (0, 4, or 8) from the link register contents and loads the result into the program counter (PC). This offset depends on the exception type.
2. The CPU copies the save program status register (SPSR\_x) contents back into the current program status register (CPSR).

#### Note

This copy restores the interrupt disable bits to the states that they had prior to the exception. It also restores the T bit, so it is not necessary for the software to switch back to the THUMB state.

### 2.12.3 Summary of Exception Switching

Table 2.2 summarizes the instructions for returning from exception handlers and the return addresses saved in the corresponding R14 register.

**Table 2.2 Returning from Exception Handlers**

	Return Instruction	Return Address		Notes
		ARM	THUMB	
		R14_x	R14_x	
BL	MOV PC , R14	PC+4	PC+2	1
SWI	MOVS PC , R14_svc	PC+4	PC+2	1
UDEF	MOVS PC , R14_und	PC+4	PC+2	1
FIQ	SUBS PC , R14_fiq , #4	PC+4	PC+4	2
IRQ	SUBS PC , R14_irq , #4	PC+4	PC+4	2
PABT	SUBS PC , R14_abt , #4	PC+4	PC+4	1
DABT	SUBS PC , R14_abt , #8	PC+8	PC+8	3
RESET	NA	-	-	4

- Notes:**
1. The program counter contains the address of the instruction triggering the branch, software interrupt, or undefined instruction exception.
  2. The program counter contains the address of the instruction superseded by acceptance of the interrupt request.
  3. The program counter contains the address of the load/store instruction triggering the data abort.
  4. After a reset, R14\_svc contains an indeterminate value.

### 2.12.4 FIQ

This exception, for supporting a data transfer or channel process, provides an abundance of private registers to reduce the need to save registers and thus the context switching overhead.

The FIQ exception handler must, regardless of the original state (ARM or THUMB), terminate by executing the following instruction.

```
SUBS PC , R14_fiq , #4
```

Setting the CPSR F bit to “1” disables FIQs. Note, however, that this bit is not accessible in User mode.

### 2.12.5 IRQ

This exception is for normal interrupt requests. Such interrupt requests have a lower priority than FIQ, so are masked during the FIQ sequence.

Setting the CPSR I bit to “1” disables IRQs. Note, however, that this bit is only accessible in non-user, privileged modes.

The IRQ exception handler must, regardless of the original state (ARM or THUMB), terminate by executing the following instruction.

```
SUBS PC , R14_irq , #4
```

### 2.12.6 Aborts

An abort indicates failure to complete the current memory access. The CPU detects abort exceptions during the memory access cycle.

There are two types of aborts.

- Prefetch abort: during instruction prefetch
- Data abort: during data access

A prefetch abort marks the instruction as invalid, but the exception is not processed until the instruction reaches the head of the pipeline. If an intervening branch or other cause prevents execution, there is no abort exception.

The handling of a data abort depends on the instruction type.

- A single data transfer instruction (LDR or STR) proceeds as far as the base register update, if specified. The abort handler must watch out for this.
- A swap instruction (SWP) aborts without doing anything at all.
- A block data transfer instruction (LDM or STM) runs to completion, updating the base register, if specified.

If the instruction overwrites the base register contents with data--that is, includes the base register in the transfer list--the overwrite is aborted. Detection of an abort blocks all such register overwrites. One important consequence is that an aborted LDM instruction always preserves the contents of R15, the last register to be transferred.

The exception handler, after removing the cause of the abort, must, regardless of the state (ARM or THUMB), execute the appropriate instructions to restore the program counter (PC) and current program status (CPSR) register and re-execute the aborted instruction.

- Prefetch abort: SUBS PC, R14\_abt, #4
- Data abort: SUBS PC, R14\_abt, #8

### 2.12.7 Software Interrupts

A software interrupt (SWI) instruction switches the CPU to Supervisor mode, normally to request a special, supervisory function. The SWI handler must, regardless of the original state (ARM or THUMB), terminate by executing the following instruction.

```
MOV PC, R14_svc
```

### 2.12.8 Undefined Instructions

Attempting to execute an instruction not supported by the CPU triggers an undefined instruction trap. This mechanism allows the programmer to expand the THUMB or ARM instruction set using software emulation. The trap handler, after emulating the failed instruction, must, regardless of the original state (ARM or THUMB), terminate by executing the following instruction.

```
MOVS PC, R14_und
```

This instruction restores the current program status register (CPSR) and returns to the instruction following the undefined one.

### 2.12.9 Exception Vectors

Table 2.3 lists exception vector addresses.

**Table 2.3 Exception Vectors**

Address	Exception	Starting Mode
0x00000000	Reset	Supervisor
0x00000004	Undefined instruction	Undefined
0x00000008	Software interrupt	Supervisor
0x0000000C	Prefetch abort	Abort
0x00000010	Data abort	Abort
0x00000014	Reserved	Reserved
0x00000018	IRQ	IRQ
0x0000001C	FIQ	FIQ

### 2.12.10 Exception Priority Order

A fixed priority system determines the order in which simultaneous exceptions are accepted.

Highest priority:

1. Reset
2. Data abort
3. FIQ
4. IRQ
5. Prefetch abort

Lowest priority:

6. Undefined instruction, software interrupt

Not all exceptions can occur at once. Undefined instruction and software interrupt share a level because they are mutually exclusive, featuring nonoverlapping opcodes.

If a data abort occurs at the same time as a FIQ, and FIQs are enabled--that is, the CPSR's F flag is "0"--the CPU proceeds to the data abort handler and then immediately to the FIQ vector. Returning normally from the FIQ handler thus causes the data abort handler to resume execution.

Data abort has a higher priority than FIQ to ensure detection of transfer errors. The time for this exception entry should be added to worst-case FIQ latency calculations.



## 2.13 Resets

The following operations follow a system reset.

1. The CPU copies the current contents of the program counter (PC) and current program status register (CPSR) to R14\_svc and SPSR\_svc, overwriting the latter with indeterminate values.
2. The CPU loads CPSR with 10011 (Supervisor mode) in M[4:0], "1" in the I and F bits, and "0" in the T bit.
3. The CPU loads PC from address 0x0000 to fetch the next instruction.
4. The CPU resumes execution in the ARM state.

## *Chapter 3*

# **Address Mapping**

---

## Chapter 3 Address Mapping

### 3.1 Overview

This LSI has a 4-gigabyte address space, divided into 32 equal banks.

The flexible features of this MCU, enable it to remap any of the available memory banks to bank 0 (addresses 0x00000000 to 0x07FFFFFF). The remapping of bank 0 can be achieved either by hardware configuration of some of the external pins or through software register settings after boot-up.

In hardware, the BSEL[1:0] external pins can be configured to select external ROM or internal FLASH memory, as the memory region to be mapped to bank 0 for use as the boot device, or boot loading mode after a reset. After booting the MCU, the remap control register together with ROM select register can be used to remap an internal or external memory to bank 0.

#### 3.1.1 Pin List

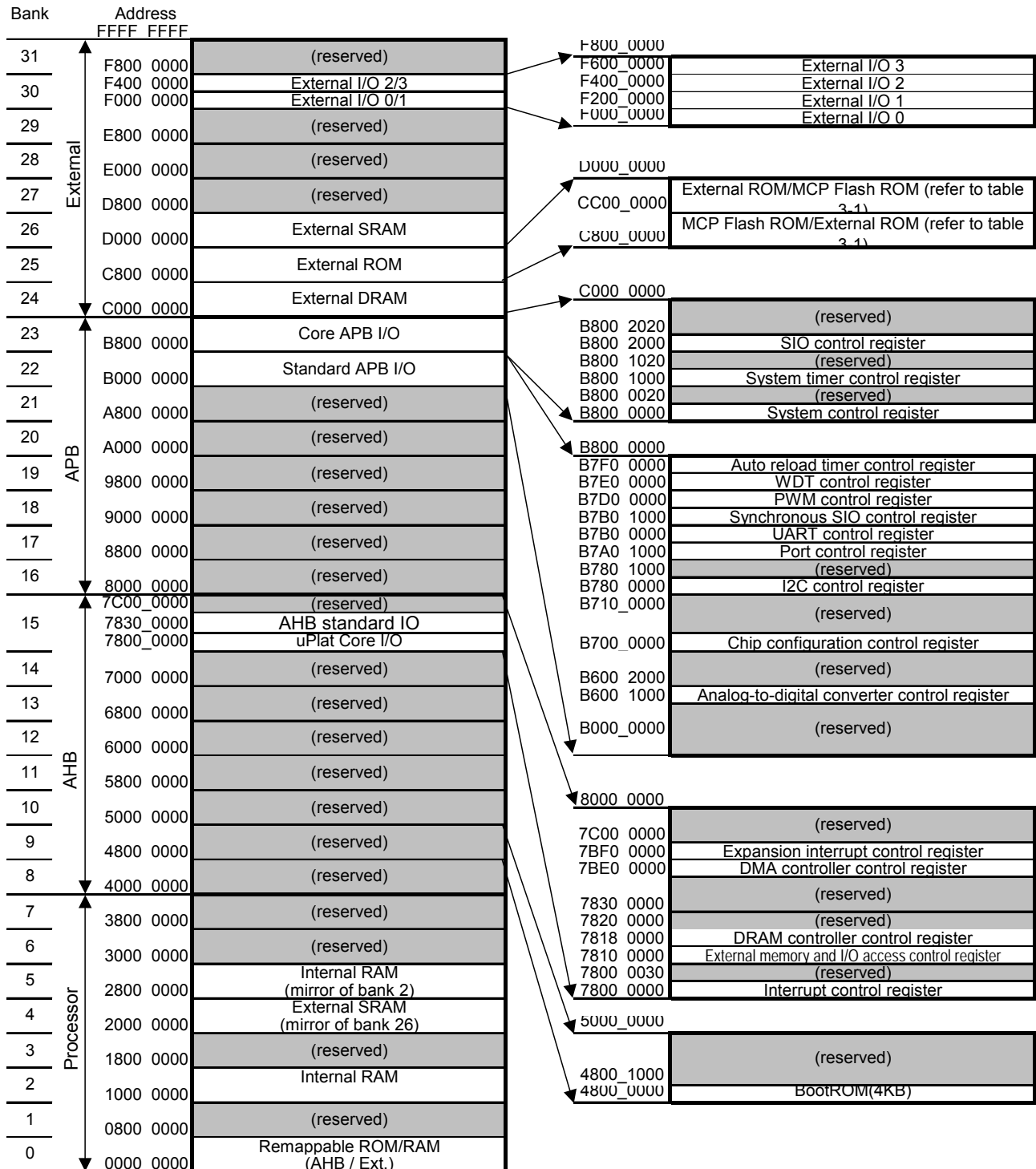
Pin Name	I/O	Description
BSEL[1]	I	Select a boot device.
BSEL[0]	I	Select a boot device.

#### 3.1.2 Register List

Address	Name	Abbreviation	R/W	Size	Initial Value
0xB800010	Remap control register	RMPCON	R/W	32	0x00000000
0xB70000C	ROM select register	ROMSEL	R/W	16	0x0000

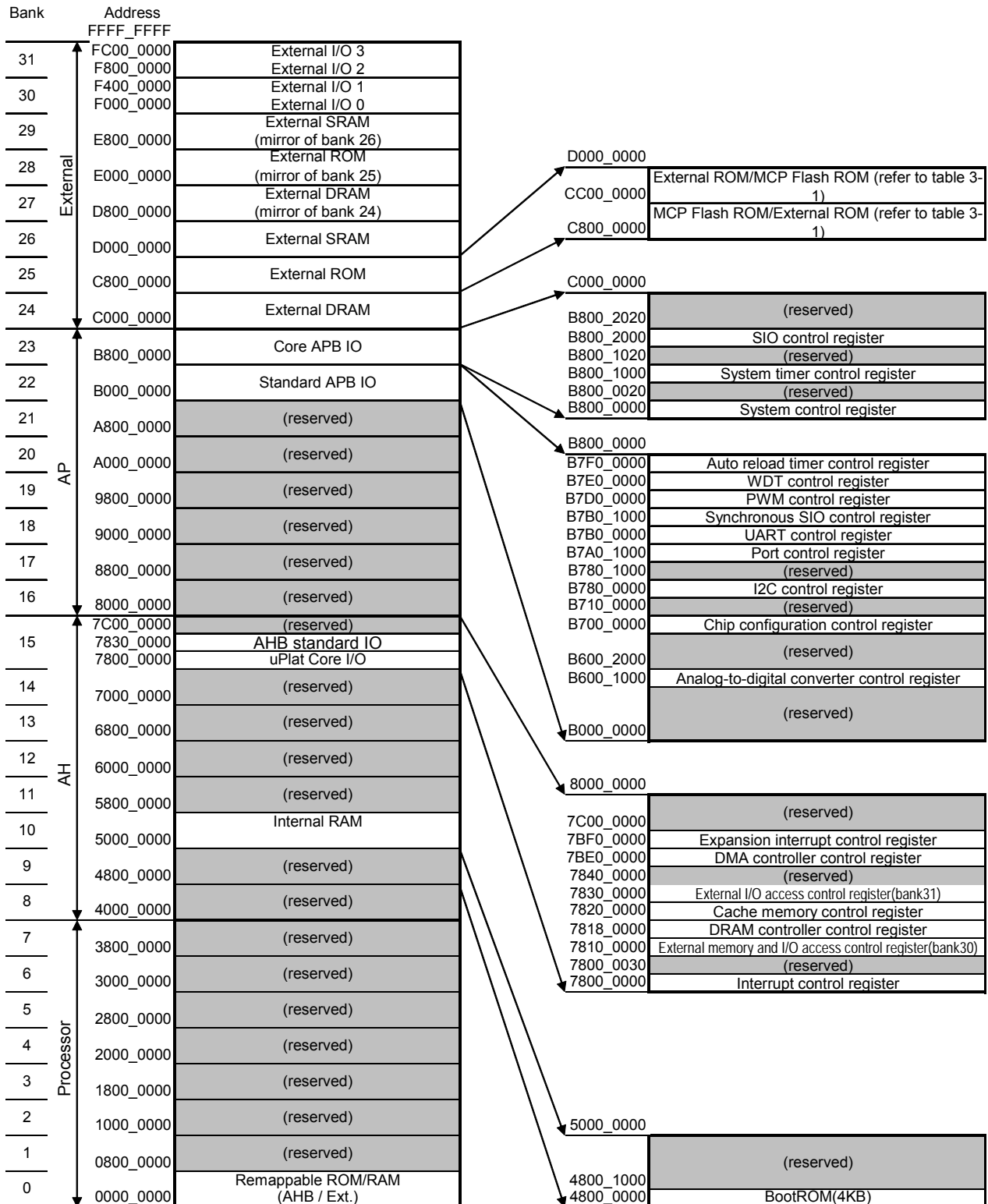
## 3.2 Address Map

### 3.2.1 ML674001 series address map



**Note:** Banks 4 and 5 respectively mirror banks 26 (external SRAM) and (internal RAM) so that software can treat the two as a single, contiguous memory region. Do not carry out access to the reserved regions in the address map. Operation is not guaranteed when accessing.

3.2.2 ML675001 series address map



**Note:** Do not carry out access to the reserved regions in the address map. Operation is not guaranteed when accessing.

### 3.3 Register Descriptions

#### 3.3.1 Remap Control Register (RMPCON)

The RMPCON register controls mapping of bank 0 to built-in SRAM, built-in FLASH ROM or external memory (ROM/DRAM/SRAM), with the ROMSEL register described in the next section, after booting is complete.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RMPCON	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	RMPM[3:0]				
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB8000010

Access: R/W

Access size: 32 bits

#### Note

– \*: These bits are for future expansion. They return “0” for reads. Writes to them are ignored.

This register features write protection to prevent errant software from accidentally overwriting register contents. The program must first write 0x3C to it immediately before updating its contents.

#### Bit Descriptions

- RMPM[3:0]** (bits 0 to 3):  
 This bit remaps bank 0 to a different device after a boot. A mapped device is selected by RMPM[3:0] bits, RSEL bit, and BSEL[1:0] pin. For a complete description of RMPM[3:0] settings and resulting effects refer to the table in Section 3.5.

### 3.3.2 ROM Select Register (ROMSEL)

The ROMSEL register controls mapping of bank 0 to built-in RAM, built-in FLASH ROM or external memory (ROM/DRAM/SRAM), with the RMPCON register described in the previous section, after booting is complete.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROMSEL	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	BSELM	RSEL
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**	0

Address: 0xB700000C  
 Access: R/W  
 Access size: 16 bits

**Note**

- \*: These bits are for future expansion. They return "0" for reads. Writes to them are ignored.  
 This register features write protection to prevent errant software from accidentally overwriting register contents. The program must first write 0x003C to it immediately before updating its contents.
- \*\* : The value depends on the input level of BSEL[0] pin.

Bit Descriptions

- RSEL** (bit 0):  
 This bit remaps bank 0 to a different device after a boot. A mapped device is selected by RMPM[3:0] bits, RSEL bit, and BSEL[1:0] pin. For a complete description of RSEL settings and resulting effects refer to the table in Section 3.5.
- BSELM** (bit 1):  
 This bit shows the input level of the BSEL[0] pin. Writing this bit is ignored. The software can use this bit to refer to the address arrangement of the built-in FlashROM of bank 25 and external ROM.

BSELM	Description
0	Input level of BSEL[0] pin is low
1	Input level of BSEL[0] pin is high

### 3.4 Remap Description

#### 3.4.1 Remap Setting

The following table shows a mapped device selected by RMPM[3:0] bits of the RMPCON register, RSEL bit of the ROMSEL register, and BSEL[1:0] pins.

MCU	RMPM[3:0]				RSEL	BSEL [1:0]		Device in bank 0	
	3	2	1	0		1	0	0x0000 0000~0x03FF FFFF	0x0400 0000~0x07FF FFFF
ML674001 ML675001	0	x	x	x	X	0	X	Bank 25 (External ROM)	
	0	x	x	x	X	1	X	Bank 9	
	1	0	0	0	X	X	X	Bank 26 (External SRAM)	
	1	0	0	1	X	X	X	Bank 24 (External DRAM)	
	1	0	1	0	X	X	X	Bank 10 (reserved) (ML674001only) Built-in RAM (ML675001only)	
	1	0	1	1	X	X	X	- (reserved)	
	1	1	x	x	X	X	X	Bank 2 Built-in RAM (ML674001only) (reserved) (ML675001only)	
ML67Q4002 ML67Q4003 ML67Q5002 ML67Q5003	0	x	x	x	0	0	0	Bank 25 Built-in FLASH memory (256K/512Kbyte)	Bank 25 External ROM (Max. 8Mbyte)
	0	x	x	x	1	0	0	Bank 25 External ROM (Max. 8Mbyte)	Bank 25 Built-in FLASH memory (256K/512Kbyte)
	0	x	x	x	0	0	1	Bank 25 External ROM (Max. 8Mbyte)	Bank 25 Built-in FLASH memory (256K/512Kbyte)
	0	x	x	x	1	0	1	Bank 25 Built-in FLASH memory (256K/512Kbyte)	Bank 25 External ROM (Max. 8Mbyte)
	0	x	x	x	X	1	X	Bank 9	
	1	0	0	0	X	X	X	Bank 26 (External SRAM)	
	1	0	0	1	X	X	X	Bank 24 (External DRAM)	
	1	0	1	0	X	X	X	Bank 10 (reserved) (ML67Q4002/ML67Q4003) Built-in RAM (ML67Q5002/ML67Q5003)	
1	0	1	1	X	X	X	- (reserved)		
1	1	x	x	X	X	X	Bank 2 Built-in RAM (ML67Q4002/ML67Q4003) (reserved) (ML67Q5002/ML67Q5003)		

**Note**

Operation is not guaranteed for a setting labeled “reserved.”

Any program that is to modify RMPCON or ROMSEL registers, must be running out of a memory region outside of bank0.

After setting an expected value as this register before Bank0 of this LSI changes, it takes worst 4 clock time. When accessing Bank0 after a setup, after performing dummy read-out to this register once after a register setup, it recommends accessing Bank0.

For remapping, several clocks are required after writing the RMPCON register. In order to guarantee the required number of clocks, it is recommended to access BANK0 after performing the dummy read to the same register (RMPCON).



### 3.4.2 Boot Control

The boot control features of this MCU enable it to map either the external ROM or built-in Flash memory to bank 0 (0x00000000 to 0x07FFFFFF) to be used as the boot device, or to set the MCU to the boot loading mode. The external BSEL[1:0] pins of the MCU must be configured so as to determine which memory bank is to be used as the boot memory after a reset. The BSEL[1:0] pin input levels must not be changed after a reset.

MCU	BSEL[1:0]		Boot device
	1	0	
ML674001/ML675001	0	X	External ROM
	1	X	Boot loading mode
ML67Q4002/ML67Q4003 ML67Q5002/ML67Q5003	0	0	Built-in FLASH ROM (256KB/512KB)
	0	1	External ROM
	1	X	Boot loading mode

### 3.4.3 Notes of Address map

The arrangement of the built-in Flash ROM and the external ROM in bank 25 varies according to the RSEL bit of the ROMSEL register and the input level of the BSEL[0] pin as the following table.

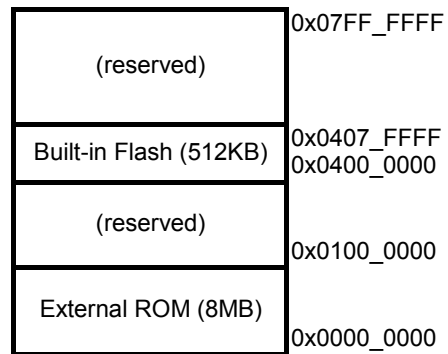
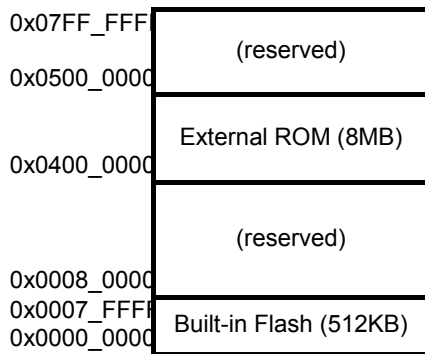
**Table 3-1 ROM address arrangement in bank 25**

MCU	RSEL	BSEL[0]	Bank 25	
			0xC800 0000~0xCBFF FFFF	0xCC00 0000~0xCFFF FFFF
ML674001 ML675001	X	X	External ROM (Max. 16Mbyte)	
ML67Q4002 ML67Q4003 ML67Q5002 ML67Q5003	0	0	Built-in FLASH ROM (256K/512Kbyte)	External ROM (Max. 8Mbyte)
	1	0	External ROM (Max. 8Mbyte)	Built-in FLASH ROM (256K/512Kbyte)
	0	1	External ROM (Max. 8Mbyte)	Built-in FLASH ROM (256K/512Kbyte)
	1	1	Built-in FLASH ROM (256K/512Kbyte)	External ROM (Max. 8Mbyte)

### 3.4.4 Remap Setting Example

ML67Q4003/ ML67Q5003  
 RMPM[3:0]=0xxx  
 BSEL[1:0]=00  
 RSEL=0  
 or  
 RMPM[3:0]=0xxx  
 BSEL[1:0]=01  
 RSEL=1

ML67Q4003/ ML67Q5003  
 RMPM[3:0]=0xxx  
 BSEL[1:0]=00  
 RSEL=1  
 or  
 RMPM[3:0]=0xxx  
 BSEL[1:0]=01  
 RSEL=0



## *Chapter 4*

# **Chip Configuration**

---

## Chapter 4 Chip Configuration

### 4.1 Overview

This LSI provides the following selection pins controlling the use of the clock output and the use of the DRAM controller blocks.

#### 4.1.1 Pin List

Pin Name	I/O	Description
DRAME_N	I	DRAM controller enable
CKOE_N	I	Clock output enable

### 4.2 Pin Description

A high level DRAME\_N input disables the DRAM controller, reducing power consumption by cutting off the clock signal to the DRAM controller block.

DRAME_N	Description
H	Disable the DRAM controller function
L	Enable the DRAM controller function

A low level CKOE\_N input enables output of HCLK to the CKO external pin.

CKOE_N	Description
H	Does not output HCLK to the CKO external pin
L	Output HCLK to the CKO external pin

*Chapter 5*

**Clock Generator**

---

## Chapter 5 Clock Generator

### 5.1 Overview

This block supplies two clock signals: HCLK to the CPU and CPU interfaces and CCLK to the timer and other counters.

The clock gear provides software control of the frequency divisors for each signal. The choices are 1, 2, 4, 8, 16 and 32\*.

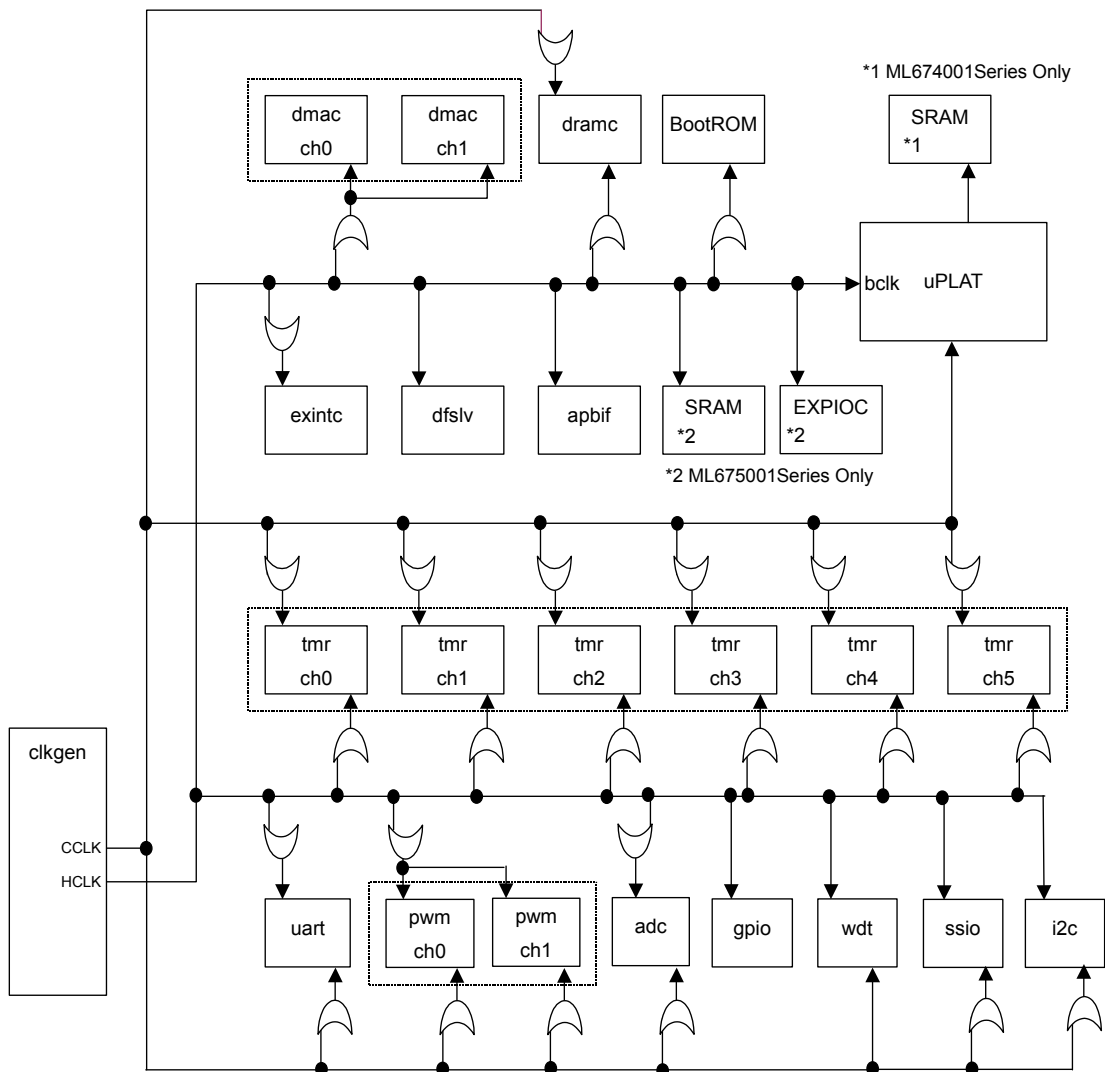
**Note:**

\*:32 can be configured ML675001 Series only.

For further details on the clock gear, see Chapter 7 "Power Management".

#### 5.1.1 Components

Figure 5.1 gives a block diagram for the clock generator.



**Figure 5.1 Clock Generator Block Diagram**

The block clock control (BCKCTL) register controls the clock signals to the functional blocks using the OR gates shown in the clock lines.

5.1.2 Pin List

Pin Name	I/O	Description
OSC0	I	Crystal connection or external clock input
OSC1_N	O	Crystal connection. Leave this pin unconnected if using external clock input.
CLKMD0	I	Clock mode 0 input (ML675001series only)
CLKMD1	I	Clock mode 1 input (ML675001series only)

5.1.3 PLL Clock frequency setup (ML675001 Series Only. \*)

PLL can be built in and the clock frequency to an inside can be set up by doubler setup of the input frequency from crystal connection terminals, and PLL.

PLL operation mode	CLKMD [1:0] setup	crystal connection terminals input clock frequency	Internal clock frequency	Note
8 time mode	11	5 to 7.5MHz	40 to 60MHz	
4 time mode	10	5 to 14MHz	20 to 56MHz	
1 time mode	01	20 to 56MHz	20 to 56MHz	A crystal cannot be used. Please use an external clock.
(Reserved)	00	—	—	—

- \* ML674001 Series does not have PLL. Internal clock frequency is same as OSC input clock frequency.
- \* In ML675001 series, PLL operation mode must not be changed during operation.

5.2 Sample Crystal Connections

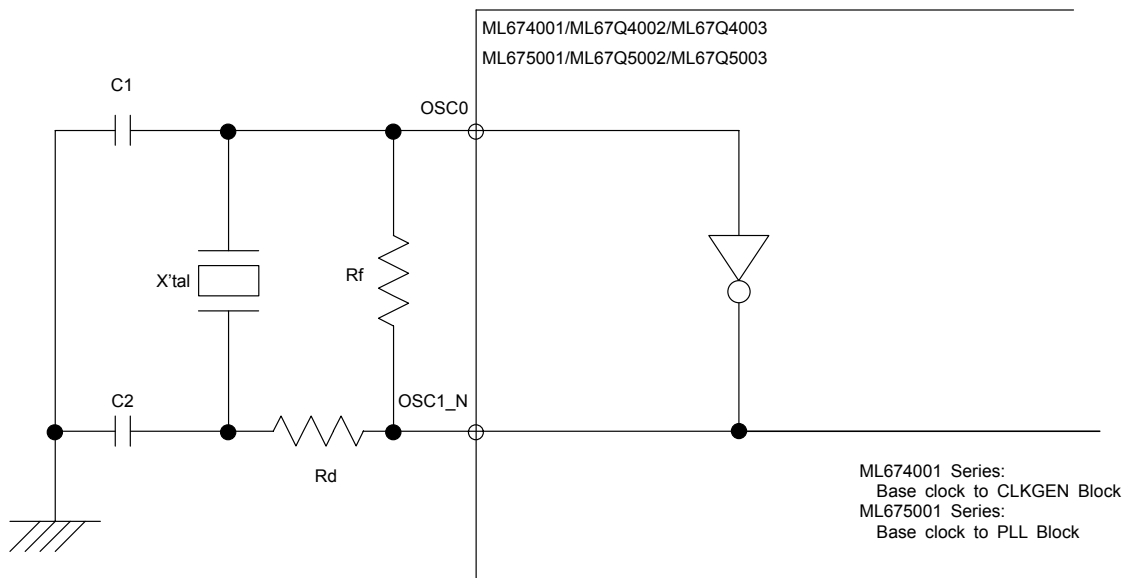


Figure 5.2 Sample Crystal Connections

## *Chapter 6*

# **Reset Control**

---



## Chapter 6 Reset Control

### 6.1 Overview

The RSTSTATUS bit in the watchdog status (WDSTAT) register indicates to the software the reason for the reset: external reset (RESET\_N) pin input or watchdog timer overflow.  
The minimum RESET\_N pulse width for triggering a system reset/initialization is 20 clock cycles.

#### 6.1.1 Pin List

Pin Name	I/O	Description
RESET_N	I	External reset input

### 6.2 Reset Types

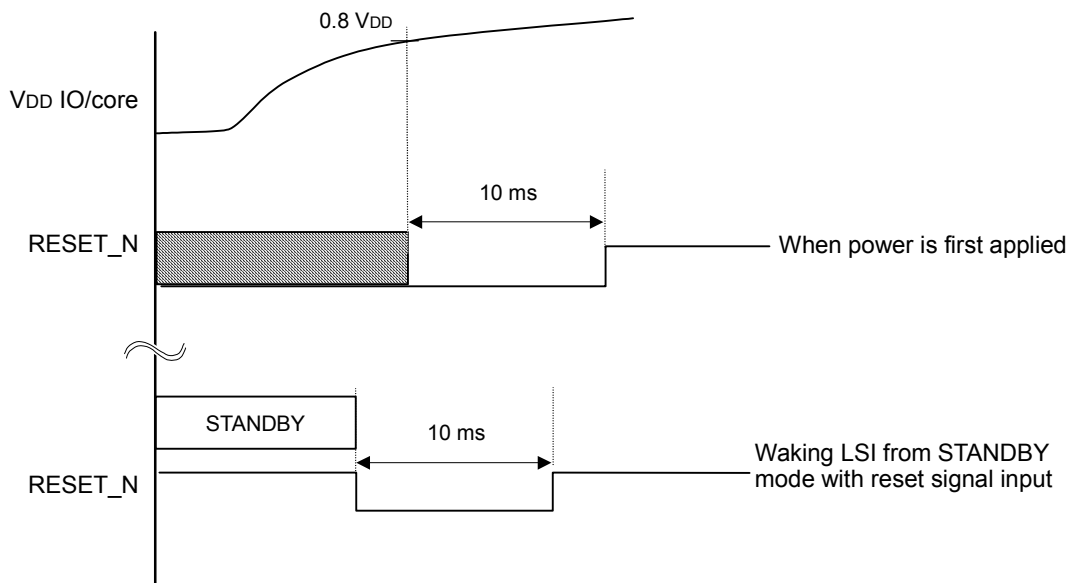
#### 6.2.1 External Reset Input

Driving the RESET\_N pin at Low level for a pulse width at least 20 clock cycles triggers a reset. When first applying the power or waking the LSI from STANDBY mode, however, increase the pulse width to at least 10 ms to allow the crystal oscillator sufficient time to stabilize.  
This type of reset leaves 0x0000 in the WDSTAT register.

#### 6.2.2 Watchdog Timer Overflow

Setting the ITM and OFINTMODE bits in the watchdog time base counter control (WDTBCON) register to "0" and "1," respectively, causes watchdog timer overflow to trigger a system reset.  
This type of reset leaves 0x0001 in the WDSTAT register.  
This type also leaves the following LSI settings unchanged: I/O port direction (input/output), I/O port function (primary/secondary), I/O port output levels, and the oscillation stabilization interval specified in the clock wait (CKWT) register.

### 6.3 Operational Description



**Figure 6.1 Reset Signal Timing**

## *Chapter 7*

# **Power Management**

---

## Chapter 7 Power Management

### 7.1 Overview

This LSI was designed with advanced power saving features to enable flexibility in optimizing power consumption. As such, a great level of configurability has been built into the power management block. This is achieved by varying the frequency of clock signal to different blocks or by stopping the clock signal entirely to certain designated blocks.

To save power, a user application system that does not need the DRAM controller, can save energy by disabling this unit through configuring the Mode Selection (DRAME\_N) pins as explained in chapter 4 of this manual. This has the effect of disable the clock signal input to this block and thus shutting them down. Note, however, that this modification cannot be undone in software. Furthermore, manipulating the input signals in hardware, after the MCU has been powered up, not only does not work, but also risks unreliable operation.

**Note:**

For further details on the DRAME\_N signals, see Chapter 4 "Chip Configuration."

Two CPU modes allow software to selectively reduce power consumption: STANDBY stops the system clock oscillation entirely; HALT mode stops the clock signals to the following functional blocks: CPU, system bus, bus control circuitry, and memory controller interfaces to built-in RAM and external memory.

The software can save energy by stopping clock signals to individual function blocks.

The software can also save energy by dynamically changing the HCLK or CCLK frequency to 1/1, 1/2, 1/4, 1/8, 1/16, or 1/32\* times the base frequency.

(\* 1/32: ML675001 Series only)

**Note:**

For further details, see "Clock Gear" below.

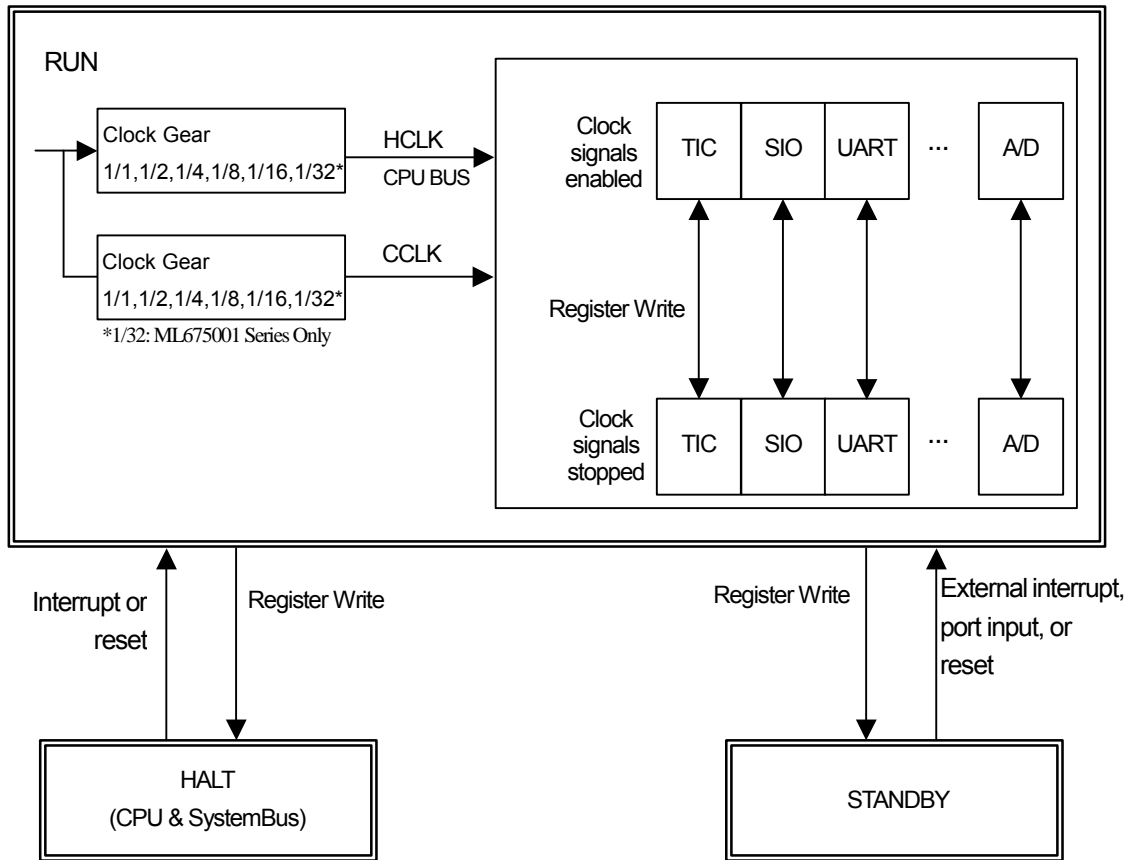
### 7.2 Power Management Functions

The following Table summarizes the power management functions available to software; Figure 7.1 gives a state transition diagram.

Operating Mode		Stopping or Changing Clock	Restarting clock signals
RUN	All functional blocks operative		
RUN	Stop clock signals to individual functional blocks	Software control	Software
RUN	Clock gear	Software control	Software
HALT	Stop clock signals to CPU, system bus, etc.	Software control	Interrupt or reset
STANDBY	Stop clock oscillation	Software control	External interrupt, port input, or reset

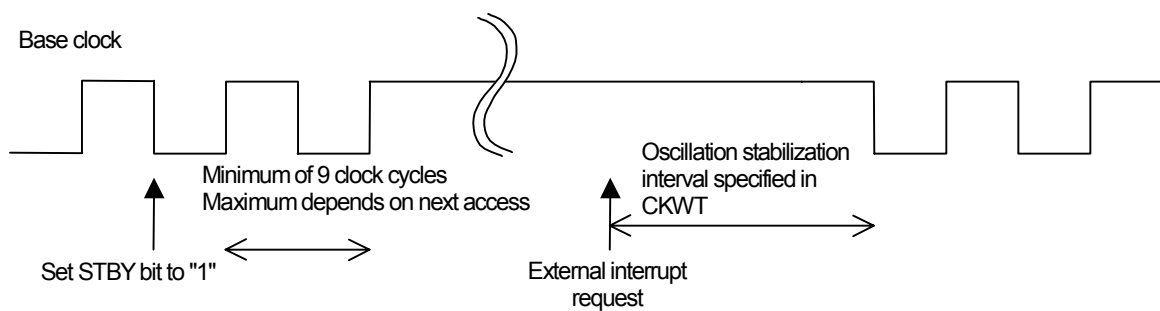
**Note:**

HALT mode stops the clock signals to the following functional blocks: CPU, system bus, bus control circuitry, and memory controller interfaces to built-in RAM and external memory.



**Figure 7.1 State Transition Diagram**

Figure 7.2 illustrates the timing for shifting to and from the HALT and STANDBY modes.



**Figure 7.2 Timing Chart**

**Notes**

1. The delay between setting the STBY bit in the CLKSTP register to "1" and stopping the clock depends on the contents of the ARM pipeline. Completing the next instruction's accesses takes a minimum of 9 clock cycles.
2. Waking from HALT mode takes only 3 to 6 clock cycles because there is no need to wait for the oscillation to stabilize before restoring clock signals to the functional blocks. The exact number depends on the wake-up signal.

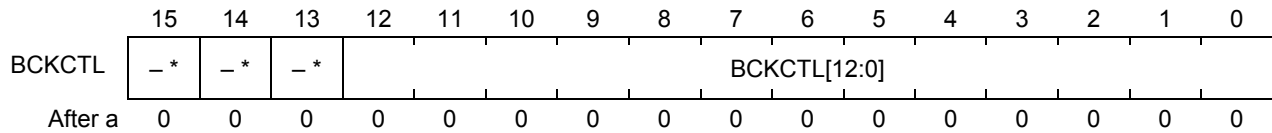
7.2.1 Register List

<b>Address</b>	<b>Name</b>	<b>Abbreviation</b>	<b>R/W</b>	<b>Size</b>	<b>Initial Value</b>
0xB7000004	Block clock control register	BCKCTL	R/W	16	0x0000
0xB8000004	Clock stop register	CLKSTP	R/W	32	0x00000000
0xB8000008	Clock select register	CGBCNT0	R/W	32	0x00000000
0xB800000C	Clock wait register	CKWT	R/W	32	0x000000FF

## 7.3 Register Descriptions

### 7.3.1 Block Clock Control Register (BCKCTL)

The block clock control (BCKCTL) register controls the clock signals to the functional blocks.



Address: 0xB7000004

Access: R/W

Access size: 16 bits

#### Note

- \*: These bits are for future expansion. They return "0" for reads. Writes to them are ignored.

#### Bit Descriptions

- **BCKCTL[0]** (bit 0):  
This bit controls the analog-to-digital converter clock signal.

BCKCTL[0]	Description
0	Supply
1	Stop

- **BCKCTL[1]** (bit 1):  
This bit controls the PWM block clock signal.

BCKCTL[1]	Description
0	Supply
1	Stop

- **BCKCTL[2]** (bit 2):  
This bit controls the timer 0 clock signal.

BCKCTL[2]	Description
0	Supply
1	Stop

- **BCKCTL[3]** (bit 3):  
This bit controls the timer 1 clock signal.

BCKCTL[3]	Description
0	Supply
1	Stop

- **BCKCTL[4]** (bit 4):  
This bit controls the timer 2 clock signal.

BCKCTL[4]	Description
0	Supply
1	Stop

- **BCKCTL[5]** (bit 5):  
This bit controls the timer 3 clock signal.

BCKCTL[5]	Description
0	Supply
1	Stop

- **BCKCTL[6]** (bit 6):  
This bit controls the timer 4 clock signal.

BCKCTL[6]	Description
0	Supply
1	Stop

- **BCKCTL[7]** (bit 7):  
This bit controls the timer 5 clock signal.

BCKCTL[7]	Description
0	Supply
1	Stop

- **BCKCTL[8]** (bit 8):  
This bit controls the DRAM controller clock signal.

BCKCTL[8]	Description
0	Supply
1	Stop

- **BCKCTL[9]** (bit 9):  
This bit controls the DMA controller clock signal.

BCKCTL[9]	Description
0	Supply
1	Stop



- **BCKCTL[10]** (bit 10):  
This bit controls the UART clock signal.

<b>BCKCTL[10]</b>	<b>Description</b>
0	Supply
1	Stop

- **BCKCTL[11]** (bit 11):  
This bit controls the SSIO clock signal.

<b>BCKCTL[11]</b>	<b>Description</b>
0	Supply
1	Stop

- **BCKCTL[12]** (bit 12):  
This bit controls the I2C clock signal.

<b>BCKCTL[12]</b>	<b>Description</b>
0	Supply
1	Stop

### 7.3.2 Clock Stop Register (CLKSTP)

This register controls transitions to the HALT and STANDBY modes and the clock signals to two more functional blocks.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLKSTP	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	STBY	-*	-*	-*	-*	HALT	TIC	SIO
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB8000004

Access: R/W

Access size: 32 bits

#### Notes

- \*: These bits are for future expansion. They return “0” for reads. Writes to them are ignored.

This register features write protection to prevent errant software from accidentally overwriting register contents. The program must first write 0x0000003C to it immediately before updating its contents.

#### Bit Descriptions

- **SIO** (bit 0):  
This bit controls the SIO block clock signal.

SIO	Description
0	Supply
1	Stop

- **TIC** (bit 1):  
This bit controls the TIC block clock signal. TIC is the AMBA Test Interface Controller which is not used in a microcontroller environment and thus can be safely disabled by setting this bit to “1”.

TIC	Description
0	Supply
1	Stop

- **HALT** (bit 2):  
Setting this bit to “1” shifts to HALT mode, stopping the clock signals to the following functional blocks: CPU, system bus, bus control circuitry, and memory controller interfaces to built-in RAM and external memory.

HALT	Description
0	Normal operation
1	Shift to HALT mode

- **STBY** (bit 7):  
Setting this bit to “1” shifts to STANDBY mode, stopping the system clock oscillation entirely.

<b>STANDBY</b>	<b>Description</b>
0	Normal operation
1	Shift to STANDBY mode

### 7.3.3 Clock Gear Control Register (CGBCNT0)

This register specifies the divisors for deriving the HCLK and CCLK clock signals from base clock.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CGBCNT0	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	CCLKSEL			-*	HCLKSEL		
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB8000008

Access: R/W

Access size: 32 bits

#### Notes

- \*: These bits are for future expansion. They return "0" for reads. Writes to them are ignored.

This register features write protection to prevent errant software from accidentally overwriting register contents. The program must first write 0x0000003C to it immediately before updating its contents.

#### Bit Descriptions

- **HCLKSEL** (bits 0 to 2) and **CCLKSEL** (bits 4 to 6):  
These bit fields respectively specify the HCLK and CCLK frequency divisors.

HCLKSEL			Frequency divisor
2	1	0	
0	0	0	1/1
0	0	1	1/2
0	1	0	1/4
0	1	1	1/8
1	0	0	1/16
1	0	1	1/32* (*:ML675001 Series Only)
1	1	0	(reserved)
1	1	1	(reserved)

CCLKSEL			Frequency divisor
6	5	4	
0	0	0	1/1
0	0	1	1/2
0	1	0	1/4
0	1	1	1/8
1	0	0	1/16
1	0	1	1/32* (*:ML675001 Series Only)
1	1	0	(reserved)
1	1	1	(reserved)

### 7.3.4 Clock Wait Register (CKWT)

This register specifies the interval to wait, after waking the LSI from the STANDBY mode, for the oscillation to stabilize before restoring clock signals to the functional blocks.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CKWT	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	CKWT			
	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Address: 0xB800000C  
 Access: R/W  
 Access size: 32 bits

#### Notes

- \*: These bits are for future expansion.

This register features write protection to prevent errant software from accidentally overwriting register contents. The program must first write 0x0000003C to it immediately before updating its contents.

Bit[31:8] return "0" for reads. Writes to them are ignored. Bit[7:4] return "1" for reads, and should be written "1" to them

#### Bit Descriptions

- CKWT (bits 0 to 3):**  
 These bits specify the amount of time to wait, after waking the LSI from the STANDBY mode, for the oscillation to stabilize before restoring clock signals to the functional blocks.  
 The following Table gives rough values for operation. The "0000" setting is for External clock input with 1clock wait of input clock to provide the chip internal clock (ML674001 Series only) . The setting "1111" is for using Crystal Oscillator with a wait between 10 ms and 24 ms regardless of the clock frequency.  
 After a reset, CKWT register has no effect. When the crystal is connected, assert the reset signal more than 10 ms to stabilize an internal clock.

CKWT				Wait Interval	
3	2	1	0	ML674001 Series (at 33MHz)	ML675001 Series (at 60MHz)
0	0	0	0	30.3 ns (for External clock input to OSC1)	(reserved)
0	0	0	1	(reserved)	(reserved)
1	1	1	0		
1	1	1	1	10 ms to 24 ms (waiting for Crystal Oscillator)	

### 7.3.5 Stopping Clock Signals to Functional Blocks

This LSI provides software control over the clock signals to individual functional blocks. The following Table lists the functional blocks with clock signal control.

Block	In software	With DRAME_N pins	Notes
SIO	O		
TIC	O		
UART	O		
SSIO	O		
I2C	O		
DMAC	O		Do not stop the clock signal while a DMA transfer is in progress.
DRAMC	O	O	Always switch DRAM to self refresh operation before stopping the clock signal in software.
TIMER	O (each timer individually)		Stopping the clock signal suspends the counter. If the timer is operational, restoring the clock signal causes counting to resume with that counter value.
PWM	O		Stopping the clock signal suspends the counter. If the PWM block is operational, restoring the clock signal causes counting to resume with that counter value.
A/D	O		Stop conversion before stopping the clock signal.

Do not access a functional block while its clock signal is stopped. Accessing the DMA controller or DRAM controller triggers an abort exception. Accessing other functional blocks risks unreliable operation. If the clock signal to a functional block is stopped, shifting to HALT or STANDBY mode and back again restarts the clock signal only if the wake-up signal is a reset.

### 7.3.6 Clock Gear

Modifying the CCLK or HCLK divisor in the clock gear control (CGBCNT0) register dynamically changes the corresponding clock frequency to 1/1, 1/2, 1/4, 1/8, 1/16, or 1/32\* times the base frequency. Changing the CCLK frequency with the clock gear affects the system timer, serial I/O (SIO) block, watchdog timer (WDT), timers, PWM block, UART, and DRAM refresh clock; changing the HCLK frequency does not.

If the clock gear is producing lower clock frequencies, shifting to HALT or STANDBY mode and back again restores the 1/1 settings only if the wake-up signal is a reset.

Always switch DRAM to self refresh operation before reducing the clock signal frequency below the minimum specified for reliable operation.

[Notes] \*1: 1/32 M675001 Series only

Notes for each function in using Clock Gear

SIO: When changing the CCLK, the communication through SIO should be stopped.

UART: When changing the CCLK, the communication through UART should be stopped.

WDT: The WDT interval will be changed when CCLK Clock Gear is changed.

DRAM: Refresh cycle setting should be set based on final CCLK frequency, before changing clock.

If refresh cycle is changed dynamically, must be set CCLK = 1/1 of clock gear temporarily.

PWM: The PWM frequency is changed when the CCLK clock frequency is changed.

AD converter: The AD conversion should be stopped when changing CCLK clock frequency.

### 7.3.7 HALT Mode

HALT mode stops the clock signals to the following functional blocks: CPU, system bus, bus control circuitry, and memory controller interfaces to built-in RAM and external memory.

External interrupt requests wake the LSI from HALT mode and restart the clock signals. Recovery requires a maximum of 10 clock cycles.

Functional blocks whose clock signals do not stop remain operational, so most can also provide such interrupt requests. The DMA controller is an exception, however, because it cannot transfer data while the system bus is disabled. It is not possible, for example, to wait in HALT mode for a DMA transfer to end and wake the LSI with an interrupt request.

The following Table summarizes the events producing shifts to and from this mode.

Shift to HALT mode	Wake from HALT mode
Write "1" to HALT bit	Unmasked interrupt request <ul style="list-style-type: none"> <li>• SIO</li> <li>• System Timer</li> <li>• UART</li> <li>• Timer</li> <li>• PWM</li> <li>• A/D</li> <li>• GPIO</li> <li>• External interrupt request</li> </ul> Reset

### 7.3.8 STANDBY Mode

STANDBY mode stops the system clock oscillation entirely, stopping all internal clock signals and greatly reducing power consumption.

When the LSI wakes from STANDBY mode, it does not restart these internal clock signals until the oscillation stabilization interval specified in the clock wait (CKWT) register has elapsed. The maximum such interval is approximately 24 ms.

The following Table summarizes the events producing shifts to and from this mode.

Shift to STANDBY mode	Wake from STANDBY mode
Write "1" to STBY bit	Unmasked interrupt request <ul style="list-style-type: none"> <li>• GPIO</li> <li>• External interrupt request (configured for level detection)</li> </ul> Reset*

#### Notes

- \* In order to use the External Interrupt Request (EXINT) for exiting STANDBY mode, the EXINT should be configured for LEVEL Detection. If using GPIOs, the GPIO interrupt signal should be held active, until exiting STANDBY.

## 7.4 Using Power Management with DRAM

Writing "1" to BCKCTL[8] to stop the clock signal to the DRAM controller does not automatically activate DRAM self refresh operation. The software must explicitly request activation and deactivation by writing 110 and 111, respectively, to the DRCMD bits in the DRAM command (DCMD) register.

For SDRAM, self refresh operation stops XSDCLK output.

Accessing the DRAM address space during self refresh operation triggers an abort exception.

### 7.4.1 Activating Self Refresh Operation

The following is the procedure for activating self refresh operation before stopping the clock signal to the DRAM controller or switching to STANDBY mode.

1. Write 110 in the DCMD register DRCMD bits to activate self refresh operation.  
The DRAM controller then suspends distributed CAS before RAS (CBR) refresh operation and stops the SDRAM clock signal (XSDCLK) until the next deactivate request.
2. Stop the clock signal to the DRAM controller in software or shift to STANDBY mode.

### 7.4.2 Deactivating Self Refresh Operation

When the software or return from STANDBY mode restarts the clock signal to the DRAM controller, write 111 in the DCMD register DRCMD bits to deactivate self refresh operation.

The DRAM controller then restarts both the SDRAM clock signal XSDCLK and distributed CAS before RAS (CBR) refresh operation.



## *Chapter 8*

# **Interrupt Controller**

---

## Chapter 8 Interrupt Controller

### 8.1 Overview

This LSI has an 8-level priority, individually maskable, highly configurable interrupt controller. The interrupt controller features are designed to provide flexibility to software programmer for designing an efficient interrupt handling routine.

The interrupt controller is connected to the nFIQ (fast interrupt request) and nIRQ (interrupt request) inputs of the ARM7TDMI processor. The processor nFIQ is only asserted in response to an external FIQ request through the pin EFIQ\_N. The processor nIRQ is asserted in response to interrupt requests from internal peripherals or external pins EXINT0-EXINT3.

In total, the interrupt controller supports 23 interrupt (IRQ) sources and 1 fast interrupt (FIQ) source. Nineteen of the interrupt sources are coming from internal peripherals such as DMA, UART, SIO, etc. The other four interrupt sources are from external inputs; EXINT0, EXINT1, EXINT2, and EXINT3. In addition it supports one external fast interrupt (FIQ) source through the pin EFIQ\_N.

The 8-level priority control feature allows the customer to define the interrupt priority level for the different interrupt sources.

External interrupt sources can be configured to be edge triggered or level triggered. Also, for edge triggered devices, the customer can configure the interrupt controller to trigger either on the negative or positive edge of the input. The FIQ is falling edge triggered.

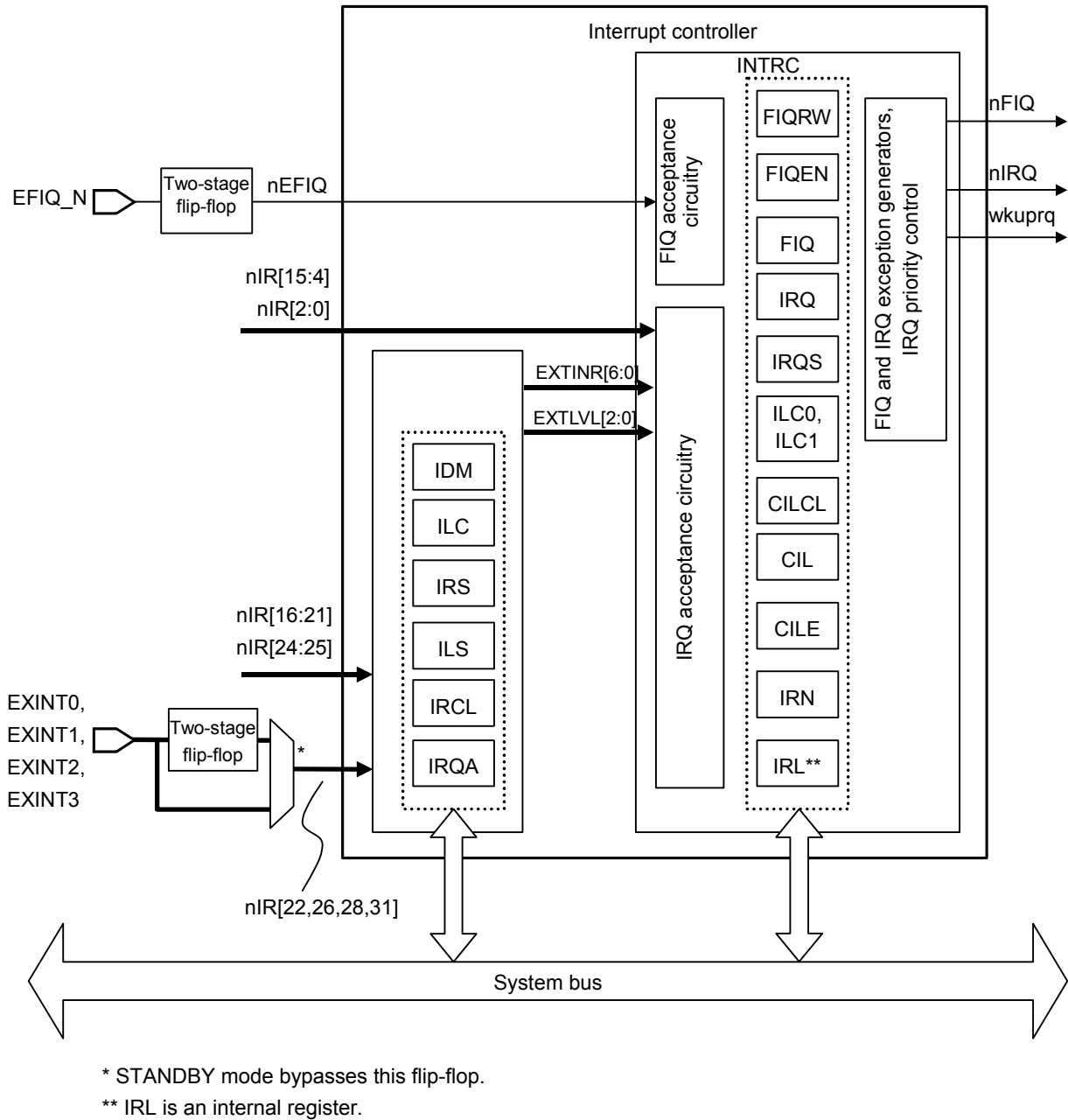
The following is an overview of main features of the interrupt controller:

#### Features

- One fast interrupt (FIQ) source (external)
- 27 interrupt (IRQ) sources (external and internal)
- Independent masking for each FIQ and IRQ source
- Independent interrupt priority level settings for each IRQ source
- Priority control blocking IRQ requests with priority levels at or below those for interrupt requests currently being processed
- Choice of level or edge sensing for external IRQ sources EXINT0 to EXINT3 (nIR22, nIR26, nIR28, and nIR31).
- Conversion of external interrupt requests to wake-up requests for restarting the clock and thus waking the LSI from STANDBY mode

8.1.1 Components

Figure 8.1 shows the interrupt controller components.



**Figure 8.1 Interrupt Controller Components**

### 8.1.2 Pin List

Pin Name	I/O	Description
EXINT0	I	Interrupt input 0
EXINT1	I	Interrupt input 1
EXINT2	I	Interrupt input 2
EXINT3	I	Interrupt input 3
EFIQ_N	I	FIQ input (negative logic)

### 8.1.3 Register List

Address	Name	Abbreviation	R/W	Size	Initial Value
0x78000000	IRQ register	IRQ	R	32	0x00000000
0x78000004	Software interrupt register	IRQS	R/W	32	0x00000000
0x78000008	FIQ register	FIQ	R	32	0x00000000
0x7800000C	FIQRAW register	FIQRAW	R	32	Reflects EFIQ_N interrupt input
0x78000010	FIQ enable register	FIQEN	R/W	32	0x00000000
0x78000014	IRQ number register	IRN	R	32	0x00000000
0x78000018	Current interrupt level register	CIL	R/W	32	0x00000000
0x7800001C	(Reserved)				
0x78000020	Interrupt level control register 0	ILC0	R/W	32	0x00000000
0x78000024	Interrupt level control register 1	ILC1	R/W	32	0x00000000
0x78000028	Current interrupt level clear register	CILCL	W	32	—
0x7800002C	Current interrupt level encode register	CILE	R	32	0x00000000
0x7BF00000	(Reserved)				
0x7BF00004	IRQ clear register	IRCL	W	32	—
0x7BF00010	IRQA register	IRQA	R/W	32	0x00000000
0x7BF00014	IRQ detection mode setting register	IDM	R/W	32	0x00000000
0x7BF00018	Interrupt level control register	ILC	R/W	32	0x00000000

## 8.2 Interrupt Sources

### 8.2.1 External Fast Interrupt (EFIQ\_N)

The external fast interrupt request (EFIQ\_N) pin is a high-priority interrupt request normally assigned to a single, time critical source. If FIQEN, bit 0 in the FIQEN register, does not mask the interrupt request, the interrupt controller asserts the fast interrupt request (nFIQ) signal to the CPU in response to detecting an incoming interrupt at the EFIQ\_N input pin of this LSI.

### 8.2.2 External Interrupts (EXINT[n])

There are four external interrupt request inputs available: EXINT[0] to EXINT[3] (nIR[22,26,28,31]). Each has its own register settings for selecting edge or level detection and specifying polarity.

### 8.2.3 Internal Interrupts (IRQn)

There are internal interrupts from the following functional blocks. (See Table in Section 8.2.4 "Interrupt Source List.")

- System timer
- Watchdog timer
- Watchdog timer interval timer operation
- General-purpose I/O ports (GPIOA / GPIOB/GPIOC/GPIOD/GPIOE)
- Software interrupt requests
- UART
- Serial I/O (SIO)
- Synchronous Serial I/O (SSIO)
- Inter Integrated Circuit (I2C)
- Analog-to-digital converter
- PWM outputs 0 and 1
- Timers 0 to 5
- DMA channels 0 and 1

8.2.4 Interrupt Source List

Interrupt Source Number	Interrupt Source	Notes
nFIQ	nFIQ	External input (EFIQ_N)
nIR0	System timer	
nIR1	Watchdog timer	
nIR2	Watchdog timer interval timer operation	
nIR3	(unused)	
nIR4	GPIOA	
nIR5	GPIOB	
nIR6	GPIOC	
nIR7	GPIOD/GPIOE	
nIR8	Software interrupt requests	
nIR9	UART	
nIR10	SIO	
nIR11	AD	
nIR12	PWM output 0	
nIR13	PWM output 1	
nIR14	SSIO	
nIR15	I2C	
nIR16	Timer 0	
nIR17	Timer 1	
nIR18	Timer 2	
nIR19	Timer 3	
nIR20	Timer 4	
nIR21	Timer 5	
nIR22	External interrupt 0	External input, choice of edge or level sensing
nIR23	(unused)	
nIR24	DMA channel 0	
nIR25	DMA channel 1	
nIR26	External interrupt 1	External input, choice of edge or level sensing
nIR27	(unused)	
nIR28	External interrupt 2	External input, choice of edge or level sensing
nIR29	(unused)	
nIR30	(unused)	
nIR31	External interrupt 3	External input, choice of edge or level sensing

### 8.3 Interrupt Levels

Each IRQ source has its own interrupt level setting. The higher the numerical value, the higher the priority. A setting of zero, on the other hand, masks interrupts from that source.

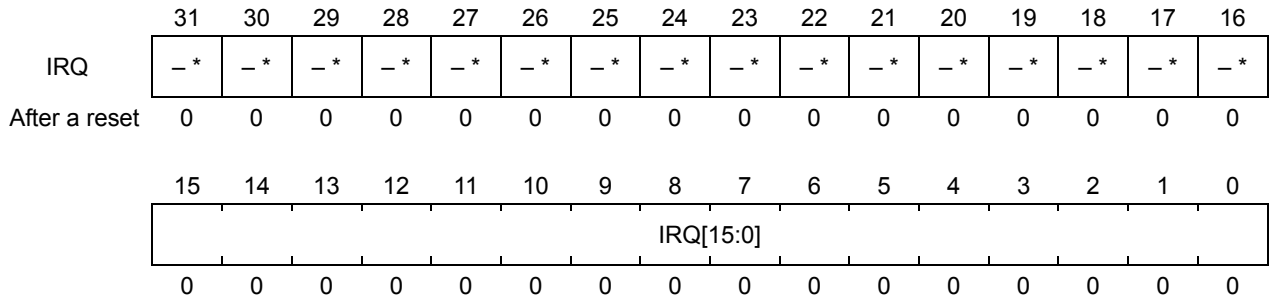
**Setting and Priority**

<b>Interrupt level setting</b>	<b>Priority</b>
7	High
↑	↑
↓	↓
1	Low
0	Interrupts masked

## 8.4 Register Descriptions

### 8.4.1 IRQ Register (IRQ)

A "1" in bit n indicates a pending (unmasked) interrupt request from the corresponding IRQ source (nIRn). The CPU has only read access to this register.



Address: 0x78000000  
Access: R  
Access size: 32 bits

#### Notes

Bits labeled "-\*" return "0" for reads, but we recommend that the program not assume "0" and mask them or adopt other "don't care" measures.

#### Bit Descriptions

- **IRQ[15:0]** (bits 0 to 15):  
A "1" in bit n indicates a pending (unmasked) interrupt request from the corresponding IRQ source (nIRn).

IRQ[15:0]	Description
0	No interrupt request pending
1	Interrupt request pending

The following Table summarizes how, for interrupt source number n, the bit IRQ[n] contents depend on the nIRn source and the interrupt level setting from the interrupt level control registers (ILC0 and ILC1).

nIRn Source	Interrupt Level Setting (from ILCx)	Bit IRQ[n]
X	0	0
1	Nonzero (1 to 7)	0
0	Nonzero (1 to 7)	1



8.4.2 Software Interrupt Register (IRQS)

Writing “1” to bit 1 in this register generates a software interrupt request, which the interrupt controller maps to nIR8. This interrupt request has the interrupt source number 8 interrupt level setting from interrupt level control register 1.

Writing “0” to this bit cancels the interrupt request.

The CPU has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IRQS	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	IRQS	-*
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0x78000004

Access: R/W

Access size: 32 bits

**Notes**

Bits labeled “- \*” return “0” for reads, but we recommend that the program not assume “0” and mask them or adopt other “don’t care” measures. Always write “0” to these bits.

**Bit Descriptions**

- **IRQS (bit 1):**  
This bit controls the software interrupt request signal.

IRQS	Description
0	Negate software interrupt request
1	Assert software interrupt request

### 8.4.3 FIQ Register (FIQ)

A "1" in bit 0 indicates a pending fast interrupt request (FIQ) from the external fast interrupt request (EFIQ\_N) pin.

The CPU has only read access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIQ	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	FIQ
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0x78000008

Access: R

Access size: 32 bits

#### Notes

Bits labeled "-\*" return "0" for reads, but we recommend that the program not assume "0" and mask them or adopt other "don't care" measures.

#### Bit Descriptions

- **FIQ (bit 0):**  
This bit indicates a pending fast interrupt request (FIQ) from the external fast interrupt request (EFIQ\_N) pin.

FIQ	Description
0	No FIQ request pending
1	FIQ request pending

The following Table summarizes how the bit FIQ[0] contents depend on the FIQRAW[0] bit in the FIQRAW register and FIQEN, bit 0 in the FIQEN register.

FIQRAW[0]	FIQEN[0]	FIQ[0]
X	0	0
0	X	0
1	1	1

## 8.4.4 FIQRAW Register (FIQRAW)

A "1" in bit 0 indicates a raw fast interrupt request (FIQ) from the external fast interrupt request (EFIQ\_N) pin. Here raw means not masked by FIQEN, bit 0 in the FIQEN register.

The CPU has only read access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIQRAW	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	FIQRAW
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(Note 2)

Address: 0x7800000C

Access: R

Access size: 32 bits

**Notes**

1. Bits labeled "-" return "0" for reads, but we recommend that the program not assume "0" and mask them or adopt other "don't care" measures.
2. The initial value reflects the EFIQ\_N interrupt input.

**Bit Descriptions**

- **FIQRAW** (bit 0):  
This bit reflects the EFIQ\_N interrupt input.

FIQRAW	Description
0	No FIQ request pending
1	FIQ request pending

### 8.4.5 FIQ Enable Register (FIQEN)

Bit 0 in this register controls masking of the external fast interrupt request (EFIQ\_N) pin input. The CPU has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIQEN	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	FIQEN
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0x78000010

Access: R/W

Access size: 32 bits

#### Notes

Bits labeled “- \*” return “0” for reads, but we recommend that the program not assume “0” and mask them or adopt other “don't care” measures. Always write “0” to these bits.

#### Bit Descriptions

- **FIQEN** (bit 0):  
 This bit controls masking of the external fast interrupt request (EFIQ\_N) pin input.

FIQEN	Description
0	Disable FIQ requests
1	Enable FIQ requests

## 8.4.6 IRQ Number Register (IRN)

This register gives the interrupt source number for the IRQ request with the highest priority.

Reading this register clears it to zero and sets the current interrupt level (CIL) register bit corresponding to the interrupt level to "1", masking pending interrupt requests at or below that level.

When an interrupt request with a higher interrupt level subsequently arrives, the interrupt controller writes its interrupt source number to this register and asserts the interrupt request (nIRQ) signal to the CPU.

The CPU has only read access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IRN	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	IRN[6:0]						
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0x78000014

Access: R

Access size: 32 bits

**Notes**

Bits labeled "-\*" return "0" for reads, but we recommend that the program not assume "0" and mask them or adopt other "don't care" measures.

**Bit Descriptions**

- **IRN[6:0]** (bits 0 to 6):  
These bits give the interrupt source number for the IRQ request with the highest priority. For the mapping of interrupt sources to source numbers, see Section 8.2.4 "Interrupt Source List."

### 8.4.7 Current Interrupt Level Register (CIL)

This register indicates the interrupt levels for interrupts currently being processed--that is, whose interrupt source numbers have been read from the IRN register. The CPU has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CIL	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	CIL[7:1]							-*
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0x78000018  
Access: R/W  
Access size: 32 bits

#### Notes

Bits labeled “- \*” return “0” for reads, but we recommend that the program not assume “0” and mask them or adopt other “don't care” measures. Always write “0” to these bits.

#### Bit Descriptions

- **CIL[7:1]** (bits 1 to 7):  
A “1” in bit n indicates that a level n interrupt request is currently being processed.

CIL[7]	CIL[6]	CIL[5]	CIL[4]	CIL[3]	CIL[2]	CIL[1]
Interrupt level 7	Interrupt level 6	Interrupt level 5	Interrupt level 4	Interrupt level 3	Interrupt level 2	Interrupt level 1

The interrupt controller masks pending interrupt requests at or below the level corresponding to the highest “1” bit in this set.

CIL[n] goes to “1” when the program reads the interrupt source number for a level n interrupt request from the IRN register.

CIL[n] returns to “0” when the program writes “1” to it. Alternatively, writing to the current interrupt level clear (CILCL) register clears the highest “1” bit in this set.

An interrupt handler must use one of these two writes to reset the highest “1” bit to “0” before returning.

8.4.8 Interrupt Level Control Register 0 (ILC0)

This register specifies the 3-bit interrupt levels for IRQ request sources nIR0 to nIR7 in four groups. The CPU has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ILC0	-*	-*	-*	-*	-*		ILR6		-*	-*	-*	-*	-*		ILR4	
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*		ILR1		-*		ILR0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0x78000020

Access: R/W

Access size: 32 bits

Notes

Bits labeled “- \*” return “0” for reads, but we recommend that the program not assume “0” and mask them or adopt other “don't care” measures. Always write “0” to these bits.

Bit Descriptions

- **ILR0** (bits 0 to 2), **ILR1** (bits 4 to 6), **ILR4** (bits 16 to 18), **ILR6** (bits 24 to 26):  
ILRn specifies the 3-bit interrupt level for IRQ request source nIRn and any others in its group.

The higher the numerical value, the higher the priority. A setting of zero, on the other hand, masks interrupts from that source.

ILR**			Interrupt Level (Priority)
2	1	0	
1	1	1	111B = 7 (high priority)
1	1	0	
1	0	1	
1	0	0	
0	1	1	
0	1	0	
0	0	1	001B = 1 (low priority)
0	0	0	Interrupts masked

The following Table summarizes the mapping of nIR0 to nIR7 to the four groups and the four fields in this register.

nIR[0]	ILR0	ILC0[2:0]
nIR[1]	ILR1	ILC0[6:4]
nIR[2]		
nIR[3]		
nIR[4]	ILR4	ILC0[18:16]
nIR[5]	ILR6	ILC0[26:24]
nIR[6]		
nIR[7]		



### 8.4.9 Interrupt Level Control Register 1 (ILC1)

This specifies the 3-bit interrupt levels for IRQ request sources nIR8 to nIR15. The CPU has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
ILC1	-*		ILR15			-*		ILR14			-*		ILR13			-*		ILR12		
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
	-*		ILR11			-*		ILR10			-*		ILR9			-*		ILR8		
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Address: 0x78000024  
 Access: R/W  
 Access size: 32 bits

#### Notes

Bits labeled “- \*” return “0” for reads, but we recommend that the program not assume “0” and mask them or adopt other “don’t care” measures. Always write “0” to these bits.

#### Bit Descriptions

- **ILR8** (bits 0 to 2), **ILR9** (bits 4 to 6), **ILR10** (bits 8 to 10), **ILR11** (bits 12 to 14), **ILR12** (bits 16 to 18), **ILR13** (bits 20 to 22), **ILR14** (bits 24 to 26), **ILR15** (bits 28 to 30): ILRn specifies the 3-bit interrupt level for IRQ request source nIRn.

The higher the numerical value, the higher the priority. A setting of zero, on the other hand, masks interrupts from that source.

ILR8 to 15			Interrupt Level (Priority)
2	1	0	
1	1	1	111B = 7 (high priority)
1	1	0	
1	0	1	
1	0	0	
0	1	1	
0	1	0	
0	0	1	001B = 1 (low priority)
0	0	0	Interrupts masked

The following Table summarizes the mappings of nIR0 to nIR7 to the fields in this register.

nIR[8]	ILR8	ILC1[2:0]
nIR[9]	ILR9	ILC1[6:4]
nIR[10]	ILR10	ILC1[10:8]
nIR[11]	ILR11	ILC1[14:12]
nIR[12]	ILR12	ILC1[18:16]
nIR[13]	ILR13	ILC1[22:20]
nIR[14]	ILR14	ILC1[26:24]
nIR[15]	ILR15	ILC1[30:28]

## 8.4.10 Current Interrupt Level Clear Register (CILCL)

Writing to this register clears the highest "1" bit in the current interrupt level (CIL) register, indicating to the interrupt controller's priority judgment circuitry that processing of the current interrupt is complete.

The data written does not matter.

The CPU has only write access to this register.



Address: 0x78000028

Access: W

Access size: 32 bits

**Note**

An interrupt handler, before returning, must reset the highest "1" bit in CIL to "0" either by writing to this register or by writing a "1" to the corresponding CIL bit.

### 8.4.11 Current Interrupt Level Encode Register (CILE)

This register gives, as a binary value, the bit position for the highest “1” bit in the current interrupt level (CIL) register and thus the interrupt level for the current interrupt request. The CPU has only read access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CIL	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	CILE[2:0]		
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0x7800002C

Access: R

Access size: 32 bits

#### Notes

Bits labeled “—\*” return “0” for reads, but we recommend that the program not assume “0” and mask them or adopt other “don’t care” measures.

#### Bit Descriptions

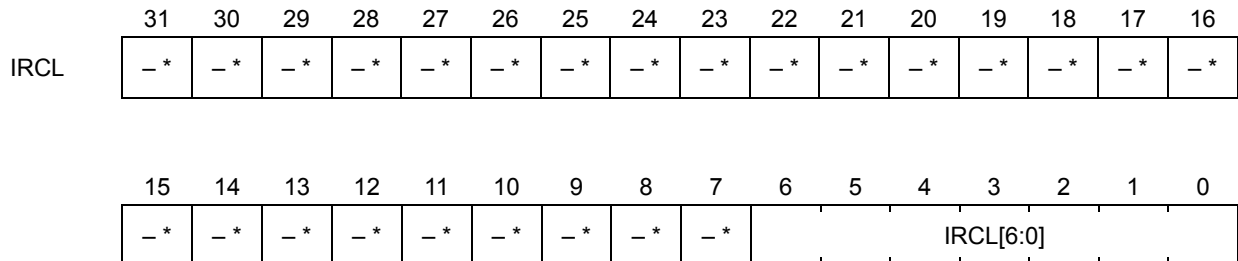
- **CILE[2:0]** (bits 0 to 2):  
These bits give the bit position and thus the interrupt level.

CIL[7:1]	CILE[2:0]	Interrupt Level (from CILE[2:0])
0000000	000	No interrupts pending
0000001	001	1
000001X	010	2
00001XX	011	3
0001XXX	100	4
001XXXX	101	5
01XXXXX	110	6
1XXXXXX	111	7

## 8.4.12 IRQ Clear Register (IRCL)

Writing an interrupt source number (nIR22, nIR26, nIR28, nIR31) to this register resets the corresponding interrupt request bit to “0”, but only if the specified external interrupt (nIR22, nIR26, nIR28, nIR31) uses edge detection as the trigger. Such writes are ignored for level detection.

The CPU has only write access to this register.



Address: 0x7BF00004

Access: W

Access size: 32 bits

**Notes**

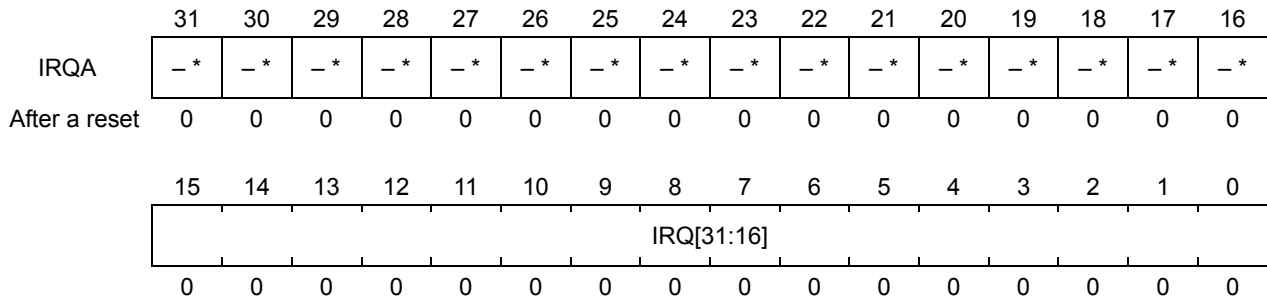
—\*: Always write “0” to these bits.

**Bit Descriptions**

- **IRCL[6:0]** (bits 0 to 6):  
These bits correspond to the interrupt source number for external interrupts (nIR22, nIR26, nIR28, nIR31). For edge triggered interrupts, writing an interrupt source number to these bits will de-assert the corresponding interrupt.

### 8.4.13 IRQA Register (IRQA)

Reading this register returns “1” bits for pending interrupt requests. Writing “1” to a bit resets it to “0,” clearing the corresponding interrupt request, but only if that external interrupt (nIR22,nIR26,nIR28 and nIR31) uses edge detection as the trigger. Such writes are ignored for level detection. The CPU has read/write access to this register, subject to write restrictions noted below.



Address: 0x7BF00010  
 Access: R/W  
 Access size: 32 bits

**Notes**

Bits labeled “—\*” return “0” for reads, but we recommend that the program not assume “0” and mask them or adopt other “don’t care” measures. Always write “0” to these bits.

**Bit Descriptions**

- **IRQ[31:16]** (bits 0 to 15):  
 When reading, these bits return the status of pending interrupts. Reading a “1” in bit n indicates a pending interrupt request (unmasked) from the corresponding IRQ source. The below table summarizes this.

IRQ[31:16]	Description
0	No interrupt request pending
1	Interrupt request pending

Writing a “1” to either of bits 6, 10, 12 and 15, resets it to “0”, clearing the corresponding interrupt request, but only if that external interrupt (nIR22, nIR26, nIR28 and nIR31) is configured as edge-triggered. Such writes are ignored for level detection as are writes to the IRQ clear (IRCL) register. Also, writes to bits other than bit 6, 10, 12 and 15 are ignored.

The following table summarizes the effects of a write to bits 6, 10, 12, 15 of this register.

IRQ[31:16]	Description
0	No effect
1	Clear interrupt request bit

The following table shows the condition that each IRQ[n] (n = 16 to 31) bit is set, if the interrupt uses level detection as the trigger. Interrupt Request State in the table is interrupt request signal from each internal device or external interrupt pin. Interrupt Level Setting is a value of interrupt level in the ILC1 register.

Interrupt Request State	Interrupt Level Setting	IRQ[n]
"H"	—	"0"
"L"	0	"0"
"L"	Nonzero (1 to 7)	"1"

The following table shows the condition that each IRQ[n] (n = 22,26,28,31) bit is set or cleared, if the interrupt uses edge detection as the trigger. The IRQA register provides two ways to clear the corresponding bit for an external interrupt (IRQ[22], IRQ[26], IRQ[28], IRQ[31],) that is using edge detection as the trigger.

Event	Polarity	Interrupt Level	IRQ[n]
Writing "1" to IRQ[n]	—	—	Clear to "0"
Writing corresponding interrupt number in IRCL register	—	—	Clear to "0"
Rising edge in interrupt request signal	Rising edge	0	Not set ("0")
		1 to 7	Set to "1"
	Falling edge	—	Not set ("0")
Falling edge in interrupt request signal*	Rising edge	—	Not set ("0")
		0	Not set ("0")
	Falling edge	1 to 7	Set to "1"

\*IRQ[n] goes to "1" only if the corresponding interrupt level is nonzero.

#### 8.4.14 IRQ Detection Mode Setting Register (IDM)

This register specifies the interrupt detection modes (level or edge detection) and polarities (negative or positive logic) for interrupt request pairs from nIR22, nIR26, nIR28 and nIR31. The CPU has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDM	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IDMP30	IDM30	IDMP28	IDM28	IDMP26	IDM26	-*	-*	IDMP22	IDM22	-*	-*	-*	-*	-*	-*
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0x7BF00014  
Access: R/W  
Access size: 32 bits

#### Notes

1. When switching triggers to edge detection, write to either the IRQ clear (IRCL) register or the IRQ register A (IRQA) to initialize the edge detection circuitry before disabling masking.
2. Bits labeled “- \*” return “0” for reads, but we recommend that the program not assume “0” and mask them or adopt other “don't care” measures. Always write “0” to these bits.

#### Bit Descriptions

- **IDM22** (bit 6), **IDMP22** (bit 7), **IDM26** (bit 10), **IDMP26** (bit 11), **IDM28** (bit 12), **IDMP28** (bit 13), **IDM30** (bit 14), **IDMP30** (bit 15):  
IDMn (n =22,26,28 or 30) specifies the detection mode for nIRm(m=22,26,28 or 31).  
IDMPn (n =22,26,28 or 30) specifies the polarity for nIRm(m=22,26,28 or 31).

IDMn	IDMPn	Detection Mode	Polarity
0	0	Level detection	Low level input
	1		High level input
1	0	Edge detection	Falling edge
	1		Rising edge

#### Notes

If the External Interrupt (EXINT) is going to be used for exiting STANDBY mode, it must be configured for LEVEL detection.



### 8.4.15 Interrupt Level Control Register (ILC)

This register specifies the 3-bit interrupt priority levels for interrupt request pairs from nIR16 to nIR31. The CPU has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
ILC	— *		ILC30			— *		ILC28			— *		ILC26			— *		ILC24		
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
	— *		ILC22			— *		ILC20			— *		ILC18			— *		ILC16		
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Address: 0x7BF00018

Access: R/W

Access size: 32 bits


#### Notes

Bits labeled “— \*” return “0” for reads, but we recommend that the program not assume “0” and mask them or adopt other “don't care” measures. Always write “0” to these bits.

#### Bit Descriptions

- **ILC16** (bits 0 to 2), **ILC18** (bits 4 to 6), **ILC20** (bits 8 to 10), **ILC22** (bits 12 to 14), **ILC24** (bits 16 to 18), **ILC26** (bits 20 to 22), **ILC28** (bits 24 to 26), **ILC30** (bits 28 to 30): ILCn specifies the 3-bit interrupt priority level for IRQn and IRQn+1.

The higher the numerical value, the higher the priority. A setting of zero, on the other hand, masks interrupts from that source.

ILC**			Interrupt Level (Priority)
2	1	0	
1	1	1	111B = 7 (high priority)  001B = 1 (low priority)
1	1	0	
1	0	1	
1	0	0	
0	1	1	
0	1	0	
0	0	1	
0	0	0	

The above settings apply to two interrupt requests each, IRQn and IRQn+1.

Interrupt Sources	Interrupt Priority Level
IRQ16	ILC16
IRQ17	
IRQ18	ILC18
IRQ19	
IRQ20	ILC20
IRQ21	
IRQ22	ILC22
IRQ23	

Interrupt Sources	Interrupt Priority Level
IRQ24	ILC24
IRQ25	
IRQ26	ILC26
IRQ27	
IRQ28	ILC28
IRQ29	
IRQ30	ILC30
IRQ31	

8.4.16 Register Settings for Interrupt Sources

The following Table summarizes the register settings for configuring interrupt requests.

Interrupt Source Number	Interrupt Source	Interrupt Level		Detection Mode			
		Register	Bit	Register	Bit		
nFIQ	nFIQ	—	—	—	—		
nIR0	System timer	ILC0	ILR0	—	—		
nIR1	Watchdog timer		ILR1				
nIR2	Watchdog timer interval timer operation						
nIR3	(unused)						
nIR4	GPIOA	ILC0	ILR4	—	—		
nIR5	GPIOB		ILR6				
nIR6	GPIOC						
nIR7	GPIOD/GPIOE	ILC1	ILR8	—	—		
nIR8	Software interrupt requests		ILR9				
nIR9	UART		ILR10				
nIR10	SIO		ILR11				
nIR11	AD		ILR12				
nIR12	PWM output 0		ILR13				
nIR13	PWM output 1		ILR14				
nIR14	SSIO		ILR15				
nIR16	Timer 0	ILC	ILC16	—	—		
nIR17	Timer 1		ILC18				
nIR18	Timer 2		ILC20				
nIR19	Timer 3		ILC22			IDM	IDM22, IDMP22
nIR20	Timer 4		ILC24			—	—
nIR21	Timer 5	ILC26	ILC26	IDM	IDM26, IDMP26		
nIR22	External interrupt 0		ILC28		ILC28	IDM28, IDMP28	
nIR23	(unused)				ILC30	ILC30	IDM30, IDMP30
nIR24	DMA channel 0		External interrupt 1			—	—
nIR25	DMA channel 1	(unused)	—	—			
nIR26	External interrupt 2	(unused)	—	—			
nIR27	(unused)	External interrupt 3	—	—			
nIR28	(unused)						
nIR29	(unused)						
nIR30	(unused)						
nIR31	External interrupt 3						

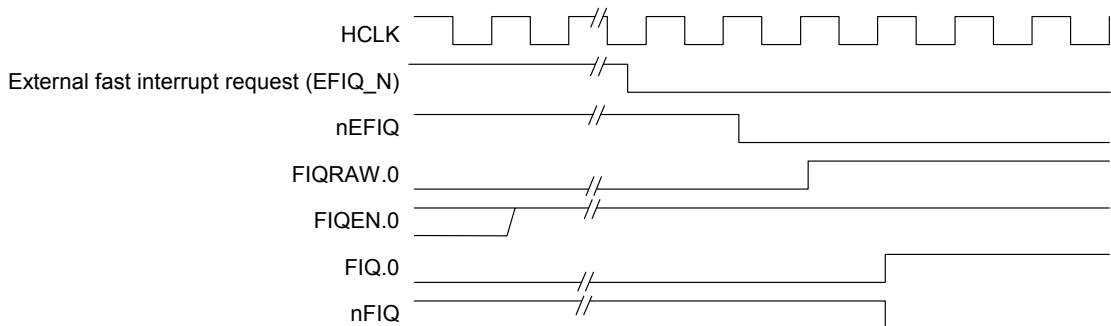
## 8.5 Description of Operation

### 8.5.1 External Fast Interrupt (EFIQ\_N)

The external fast interrupt request (EFIQ\_N) pin is a high-priority interrupt request normally assigned to a single, time critical source. If FIQEN, bit 0 in the FIQEN register, does not mask the interrupt request, the interrupt controller asserts the fast interrupt request (nFIQ) signal to the CPU in response to a detected interrupt trigger.

#### External Fast Interrupt (EFIQ\_N) Sequence

0. Remove FIQ interrupt mask  
Setting F, bit 6 in the CPU's current program status (CPSR) register, to "0" and FIQEN, bit 0 in the FIQEN register, to "1" disables masking.
1. Wait for interrupt  
External fast interrupt request input from the EFIQ\_N pin sets bit 0 in the FIQRAW register to "1."
2. Relay exception request to CPU  
The interrupt controller sets bit 0 in the FIQ register to "1" and asserts the fast interrupt request (nFIQ) signal to the CPU.



3. Accept request  
If F, bit 6 in the CPU's current program status (CPSR) register, enables FIQ exceptions, the CPU saves the address of the next instruction in r14\_fiq, saves the CPSR contents in SPSR\_fiq, and sets the CPSR F(bit6) and I(bit7) to "1" to block acceptance of both FIQ and IRQ exceptions by the CPU.
4. Process interrupt  
The FIQ exception handler must negate the FIQ request from the source before returning.
5. Return from interrupt  
The FIQ exception handler terminates by executing a return from interrupt instruction, which restores the instruction address and CPSR contents from r14\_fiq and SPSR\_fiq.

#### Note

For further details on CPU processing of FIQ exceptions, refer to the ARM7TDMI data sheet.

### 8.5.2 External and Internal Interrupts (IRQn)

These are regular interrupt requests with lower priority than the fast interrupt request (FIQ). The interrupt controller assigns priority of execution to simultaneous interrupt requests by comparing their interrupt levels. If a new interrupt request has a higher interrupt level than the one currently being processed, the interrupt controller asserts the interrupt request (nIRQ) signal to the CPU.

The IRQ exception handler reads the interrupt source number and level from interrupt controller registers.

#### Assigning Priority to Interrupt Requests

1. The higher the numerical value, the higher the priority.
2. If two interrupt requests have the same interrupt level, priority goes to the one with the higher interrupt source number.
3. Reading the IRN register in response to an IRQ exception masks interrupt requests at or below the new interrupt level.

#### External/Internal Interrupt (IRQn) Sequence

0. Specify interrupt level (software)  
Setting I, bit 7 in the CPU's current program status (CPSR) register, to "0" and specifying a nonzero interrupt level for the interrupt disables masking. An interrupt level of zero masks interrupts.

#### Note

Sources nIR22, nIR26, nIR28, nIR31 require an additional preliminary step: specifying the detection mode and polarity in the IRQ detection mode setting register (IDM). When switching triggers to edge detection, write to either the IRQ clear (IRCL) register or the IRQ register A (IRQA) to initialize the edge detection circuitry before disabling masking.

1. Wait for interrupt (hardware)  
If the interrupt request is an external interrupt (nIR22, nIR26, nIR28, nIR31) using edge detection as the trigger, the interrupt controller sets the corresponding bit in the IRQ register A (IRQA) to "1".
  2. Relay exception request to CPU (hardware)  
If the interrupt request has an interrupt level higher than the contents of the current interrupt level encode (CILE) register, which gives the bit position for the highest "1" bit in the current interrupt level (CIL) register, the interrupt controller writes the highest interrupt number for interrupt requests at that higher interrupt level to the IRQ number (IRN) register and asserts the interrupt (nIRQ) signal to the CPU.  
If the interrupt level is less than or equal to that in CIL (and CILE), however, there is no IRQ exception, and IRN goes to zero.
  3. Accept request (hardware)  
If I, bit 7 in the CPU's current program status (CPSR) register, enables IRQ exceptions, the CPU saves the address of the next instruction in the link (R14\_irq) register, saves the CPSR contents in the program status (SPSR\_irq) register, and sets I, bit 6 in the CPU's current program status (CPSR) register, to "1" to block acceptance of IRQ exceptions by the CPU. Control then passes to the IRQ exception handler.
  4. Process interrupt (software & hardware)  
The IRQ exception handler (software) reads the interrupt source number from the IRQ number (IRN) register and branches to the corresponding interrupt handler.  
The interrupt controller (hardware) clears the IRN register to zero, sets the current interrupt level (CIL) register bit corresponding to the interrupt level to "1," masking pending interrupt requests at or below that level, negates the interrupt request (nIRQ) signal to the CPU, and writes that interrupt level as a binary value to the current interrupt level (CIL) register.
-

The interrupt handler (software), before returning, resets the corresponding bit in the IRQ register A (IRQA) to "0" if the external interrupt (nIR22, nIR26, nIR28, nIR31) uses edge detection as the trigger.

5. Return from interrupt (software)

Writing to the current interrupt level clear (CILCL) register resets the highest "1" bit in CIL, the one indicating the interrupt level currently being processed, to "0." The data written does not matter.

The interrupt handler terminates by executing a return from interrupt instruction, which restores the instruction address and CPSR contents from the link (R14\_irq) and program status (SPSR\_irq) registers.

**Note**

For further details on CPU processing of IRQ exceptions, refer to the ARM7TDMI data sheet.

### 8.5.3 Nested Interrupts and Re-Entrant Interrupt Service Routines

Setting I, bit 7 in the CPU's current program status (CPSR) register, to "0" disables masking, allowing an interrupt request with an interrupt level higher than that for the one currently being processed to take control, and thus facilitating interrupt nesting.

Before an interrupt handler enables interrupt nesting, however, it must first save to the stack the contents of the link (R14\_irq) and program status (SPSR\_irq) registers because the CPU hardware overwrites them when it accepts such a nested IRQ exception.

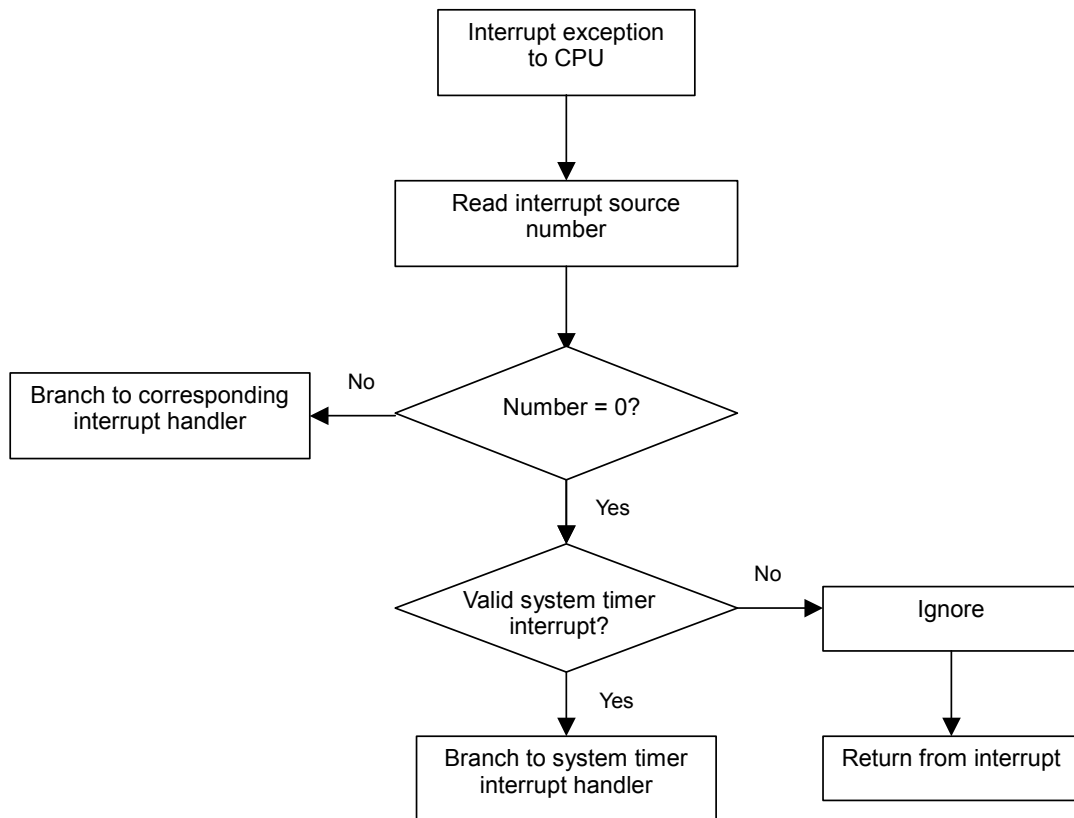
If the interrupt handler calls subroutines, it must also protect the subroutine's return address in R14\_irq from the overwrite inherent in accepting such a nested IRQ exception by shifting to system mode. System mode uses the same register bank as user mode, but allows the same access as IRQ mode to status registers and the like. Note that an interrupt handler running in this (or user) mode must save to the stack the contents of the link (R14\_irq) register and any work registers that it uses.

The following outlines the procedure.

1. Save contents of R14\_irq, SPSR\_irq, and any necessary work registers.
2. Read IRN.
3. Switch to system mode and enable IRQ exceptions.
4. Save contents of system (user) mode link register and any necessary work registers.
5. Execute body of interrupt handler.
6. Restore contents of system (user) mode registers.
7. Disable IRQ exceptions and switch back to IRQ mode.
8. Clear corresponding bit in CIL register.
9. Restore contents of work registers, SPSR\_irq, and R14\_irq.
10. Return from interrupt.

#### 8.5.4 Important Notes on Interrupts

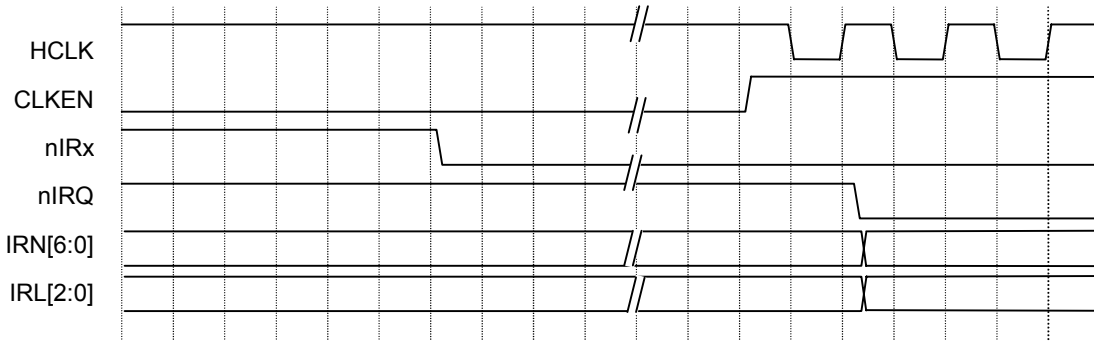
- Reading IRN  
Reading the IRN register clears it to zero.
- Invalid Interrupts  
Loss of the interrupt trigger after the interrupt controller asserts the interrupt request (nIRQ) signal to the CPU sometimes causes the read from the IRQ number (IRN) register to return 0, the interrupt source number assigned to the system timer interrupt. The IRQ exception handler must therefore cross-check the system timer interrupt status and branch to the corresponding interrupt handler only if the interrupt request is valid. Otherwise, it must ignore the invalid interrupt and simply return.



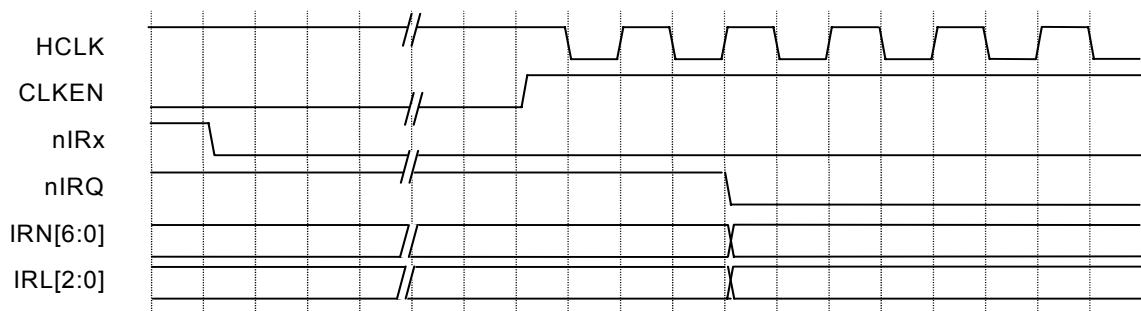


8.5.5 Waking from HALT and STANDBY Modes

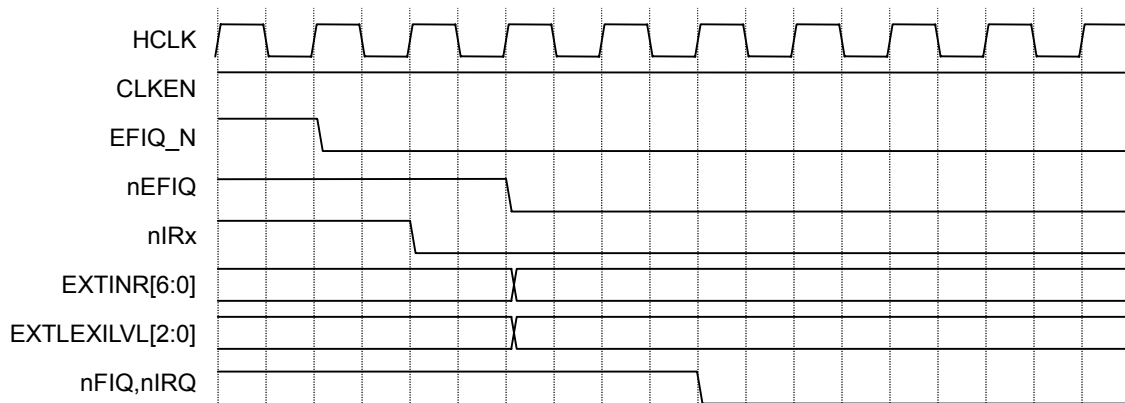
The following Figures represent timing charts for waking the CPU from HALT or STANDBY mode with interrupt requests.



(a) HALT/STANDBY/STOP → RUN for interrupt requests nIR0 to nIR15 (asynchronous clock)



(b) HALT/STANDBY/STOP → RUN for interrupt requests nIR16 to nIR31 (asynchronous clock)



(c) RUN (synchronous clock)

### 8.5.6 Error Response

Accessing addresses 0x78000030 to 0x780FFFFFF in the region assigned to the interrupt controller (INTRC) produces a data abort exception.

### 8.5.7 Interrupt Response Times

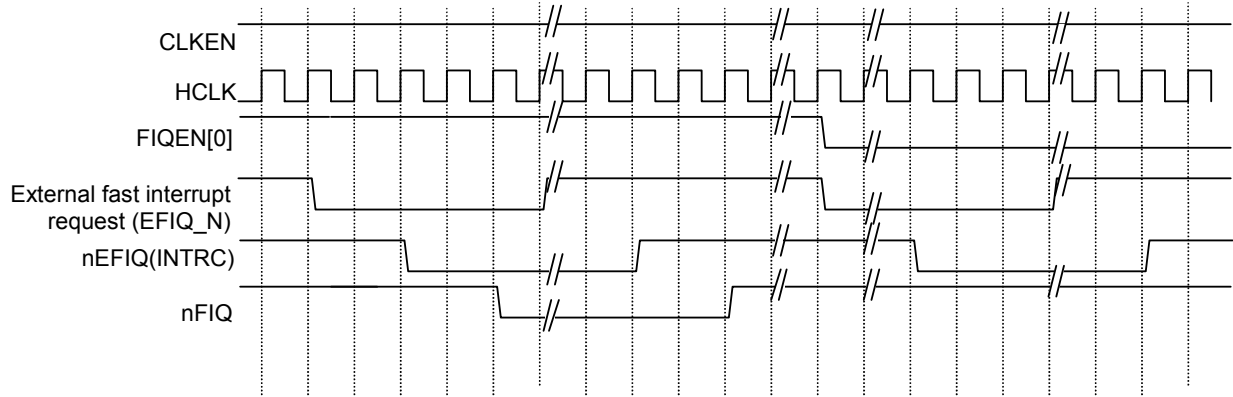
Fast interrupt (nFIQ)	4 clock cycles from EFIQ_N external interrupt input to CPU FIQ input
Interrupt request sources (nIR0 to nIR15)	2 clock cycles from IRQ request to CPU IRQ input
Interrupt request sources (nIR16 to nIR21, nIR24, nIR25)	3 clock cycles from IRQ request to CPU IRQ input
Interrupt request sources (nIR22, nIR26, nIR28, nIR31)	5 clock cycles from EXINTx external interrupt input to CPU IRQ input

**Note**

For the delays between the IRQ or FIQ exception request to the CPU and the actual start of the corresponding exception handler, refer to the ARM7TDMI data sheet.

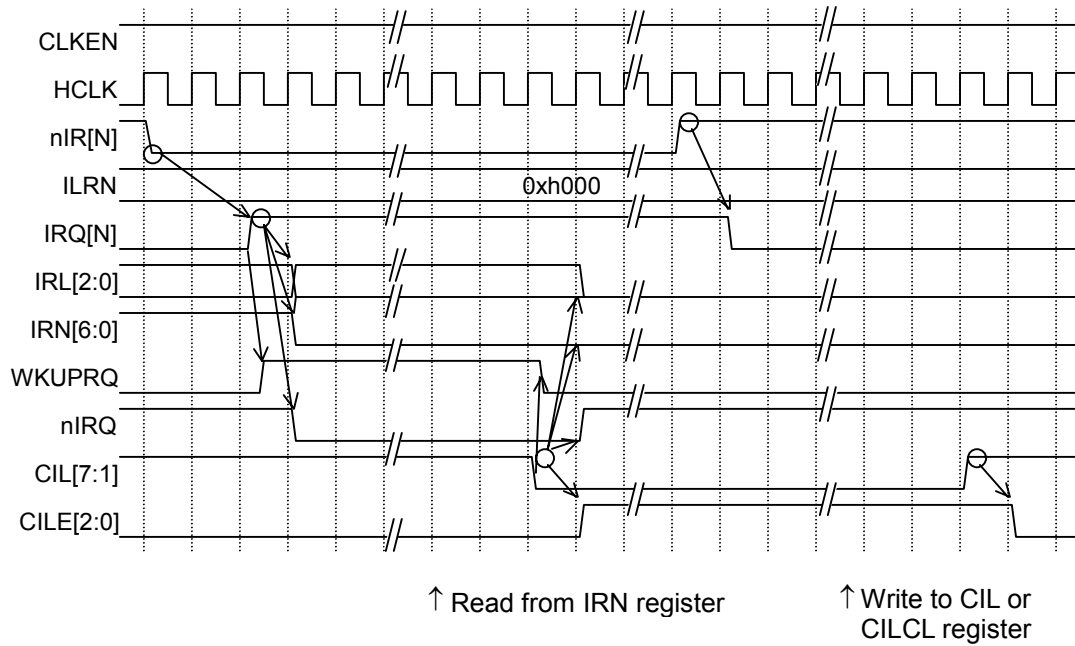
## 8.6 Interrupt Acceptance Timing Charts

### 8.6.1 FIQ Interrupt Timing Chart

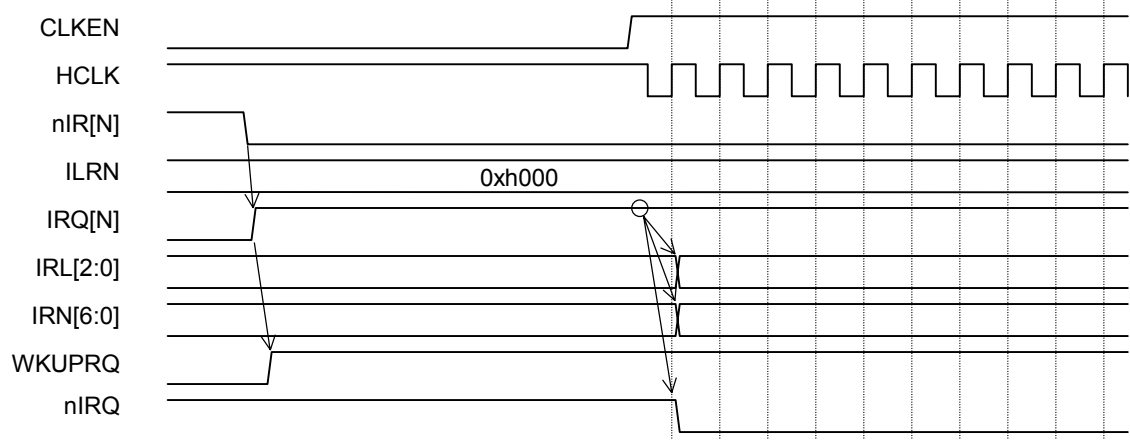


**Clock operational**

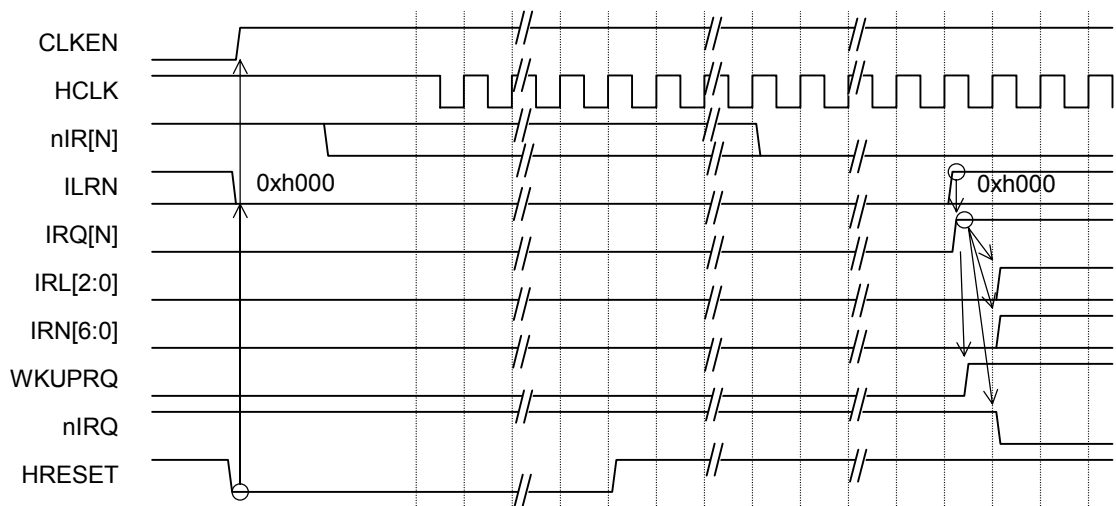
### 8.6.2 IRQ Interrupt Timing Chart (nIR0 to nIR15)



**Clock operational**



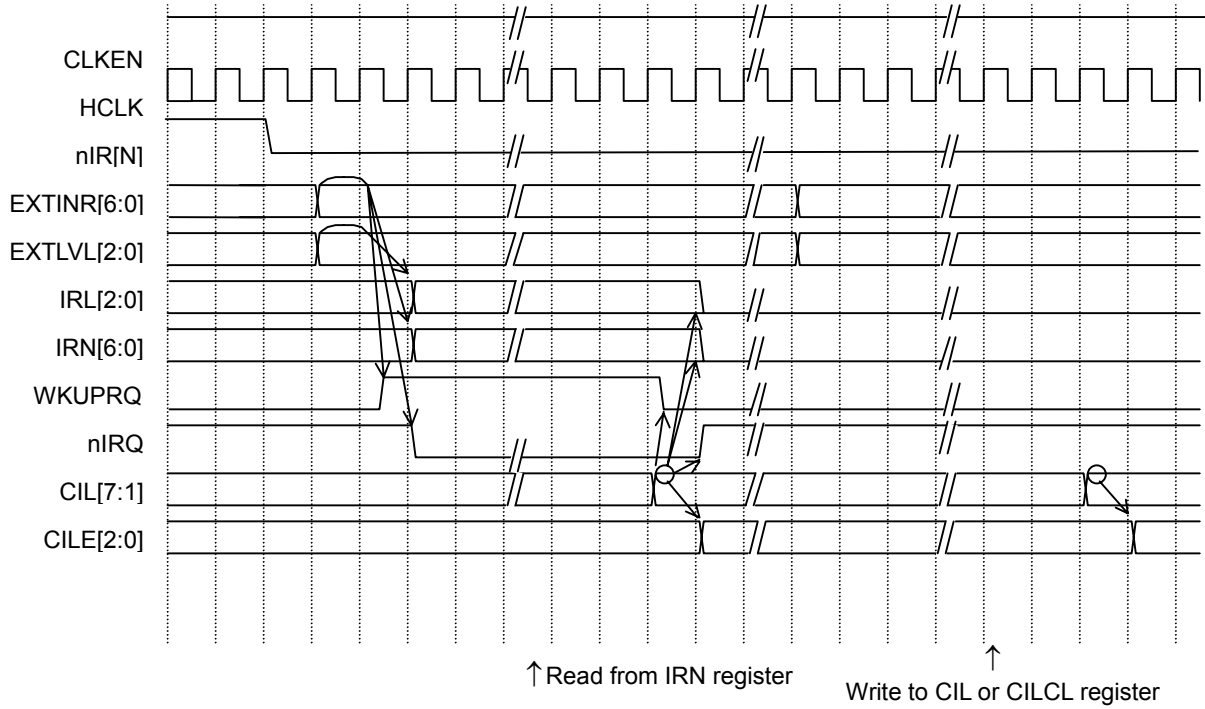
**STOP → RUN**



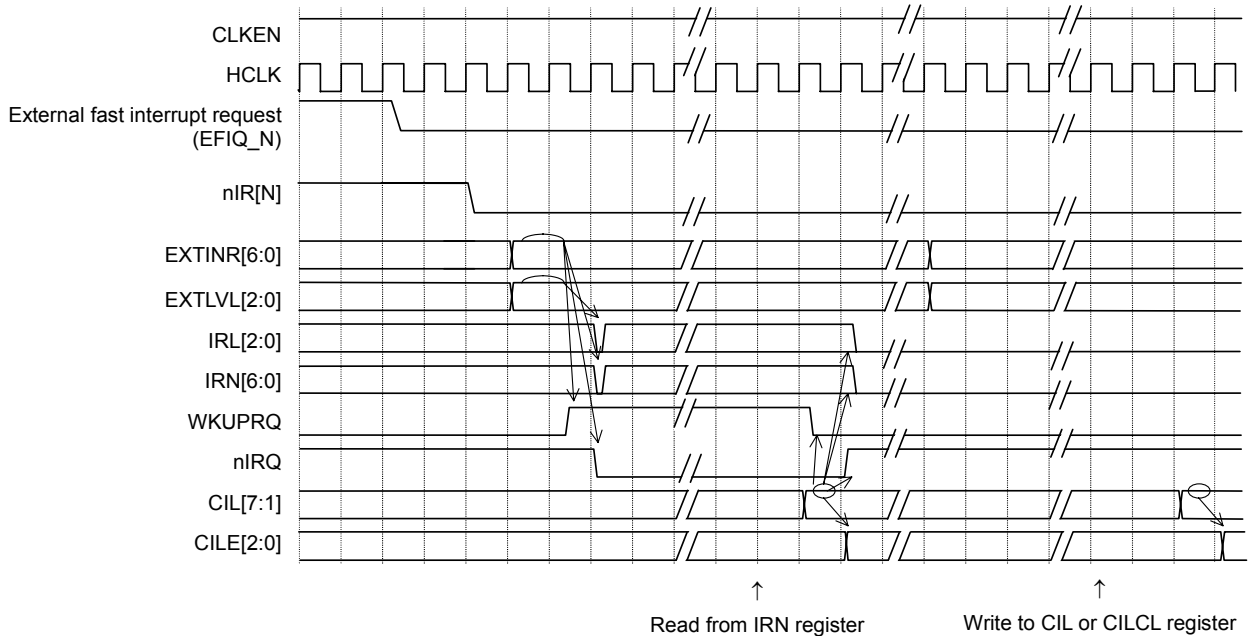
↑ Write to ILC0 or ILC1 register

**Sleep → RUN**

8.6.3 IRQ Interrupt Timing Chart (nIR16 to nIR31)



**Clock operational (nIR16 to nIR21, nIR24, nIR25)**



**Clock operational (nIR22, nIR26, nIR28, nIR31)**

## *Chapter 9*

# **Cache Memory**

---

## Chapter 9 Cache Memory

This chapter is applied only to ML675001 series. ML674001 series do not have the built-in cache memory.

### 9.1 Overview

ML675001 series has an 8k byte cache memory (unified cache) that contains both instructions and data.

Features:

- Cash memory with a capacity of 8k bytes that contains both instructions and data.
- Write-back method
- 4-way set associative
- 16-byte block size
- Cacheable/non-cacheable setting can be made in units of 128M byte banks.
- Lock setting of the cache memory is possible in units of a way. (It is possible to hold in the cache memory the high speed processing programs such as interrupt processing, etc.)
- Flush control by software

#### 9.1.1 Configuration

Figure 9-1 shows the cache memory configuration.

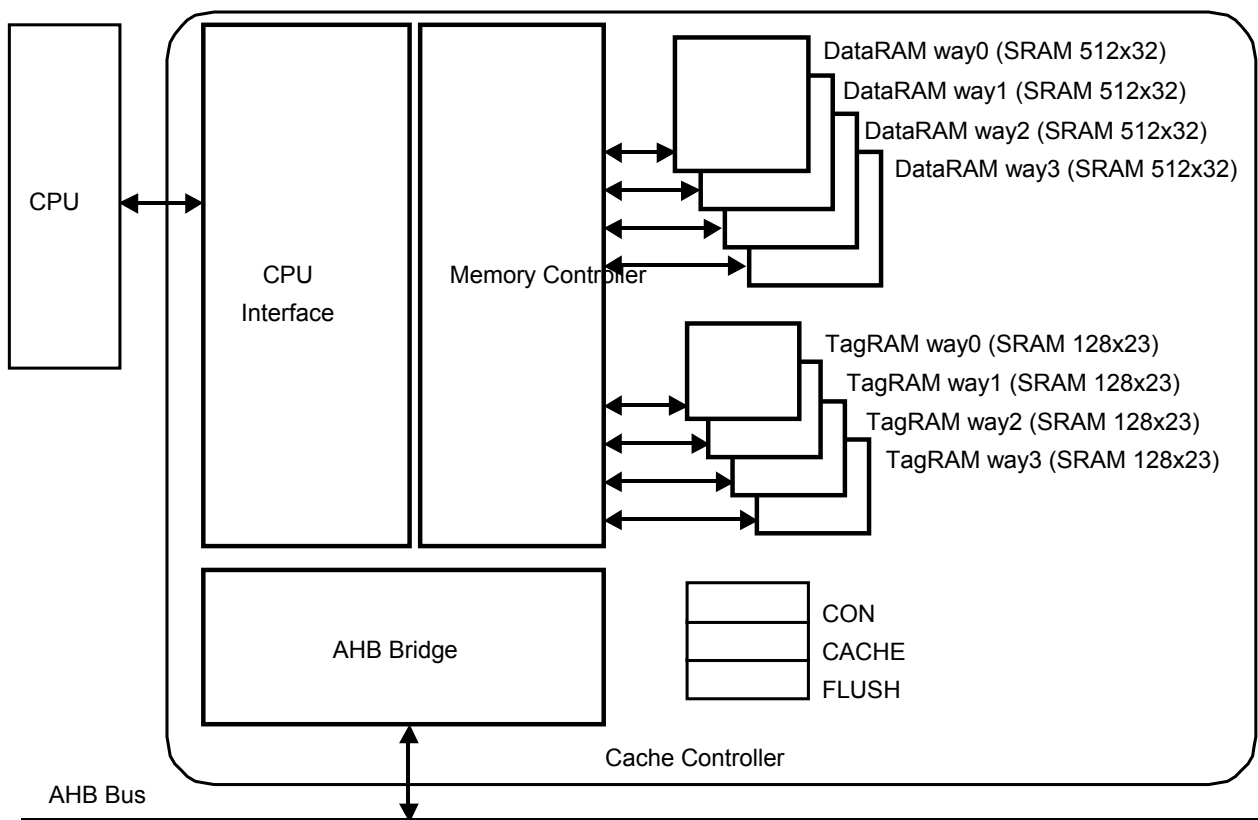


Figure 9.1 Cache Memory Configuration Diagram

9.1.2 List of Control Registers

Address	Abbreviation	R/W	Name, brief description	Initial value
0x7820_0004	CON	R/W	Cache lock control register	0x0000_0000
0x7820_0008	CACHE	R/W	Cacheable register	0x0000_0000
0x7820_001C	FLUSH	W	FLUSH register	Undefined

9.2 Description of Control Registers

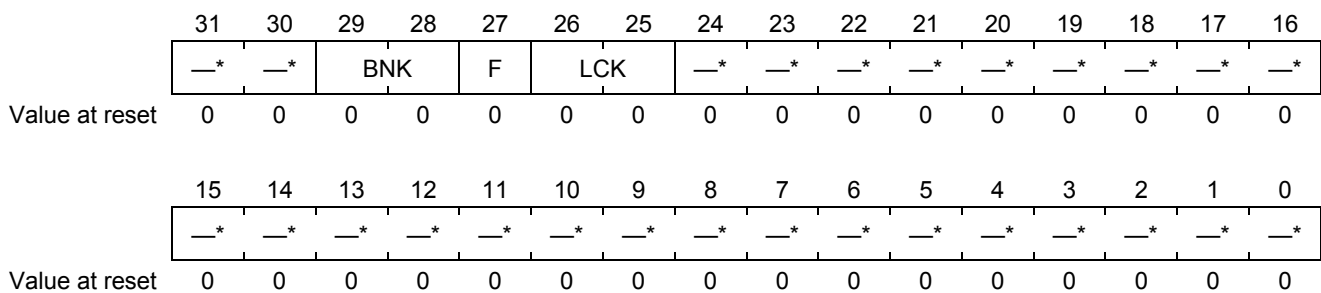
In this description of registers, the offset value expression method is used for giving the addresses. The absolute address is calculated by adding the offset address to the base address of each module.

- Base address value of cache memory controller: CACH Base = 0x7820\_0000

9.2.1 Cache Lock Control Register (CON)

The CON register is used for setting the locking operation of the cache memory. While the cache memory is split into four ways (storage locations), it is possible to lock the cache memory contents in units of one way by setting LCK[1:0]. Further, by using the BNK[1:0] and F settings, it is possible to load instructions or data to the way to be locked.

Set the value 0x0000\_0000 in this register when not using the locking function.



Address: CACH Base + 0x04  
 Access: R/W  
 Access size: 32 bits

Note: “—\*” indicates a reserved bit. Always write “0” to the bit. If a “1” is written to this bit, the operations cannot be guaranteed.

Description of bits:

- LCK  
 Selects the way of the cache memory to be locked.

LCK[1:0]	Way3	Way2	Way1	Way0	Operation
“00”	Cache	Cache	Cache	Cache	No way locked
“01”	Cache	Cache	Cache	Locked	One way locked
“10”	Cache	Cache	Locked	Locked	Two ways locked
“11”	Cache	Locked	Locked	Locked	Three ways locked

Note: A locked way will not be flushed even when a flushing operation is made using the FLUSH register.



- **F**  
Sets the cache memory in the Load mode.  
When F=0 is set, normal cache operations will be made. For details, please refer to Section 9.3.  
When F=1 is set, all store operations to the cache memory will be made forcibly to the way selected by BNK. Also, in this case, all instruction fetch accesses will be non-cacheable accesses. On the other hand, all data accesses will be made in the normal manner according to the setting of the cacheable register setting (cacheable or non-cacheable).

F	Description
0	Sets to normal cache operation (default setting).
1	Sets cache to the Load mode operation.

- **BNK**  
Selects the way of the cache memory in which loading is to be made.  
This setting is valid if the F bit has been set to "1".

BNK	Description
00	Way 0 selected
01	Way 1 selected
10	Way 2 selected
11	Way 3 selected

### 9.2.2 Cacheable Register (CACHE)

The CACHE register is used for making the cacheable/non-cacheable setting for each memory bank.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	C0
Value at reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	C31	C30	C29	C28	C27	C26	C25	C24	C15	C14	C13	C12	C11	C10	C9	C8
Value at reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: CACH Base + 0x08

Access: R/W

Access size: 32 bits

Note: “—\*” indicates a reserved bit. Always write “0” to the bit. If a “1” is written to this bit, the operations cannot be guaranteed.

Bit descriptions:

- C8, C9, C10, C11, C12, C13, C24, C25, C26, C28, C29, C0  
 The cacheable register allows cacheable/non-cacheable setting for each bank. Each of these bits sets the enable or disable condition for the following banks, respectively.  
 C8: Bank8, C9: Bank9, C10: Bank10, C11: Bank11, C12: Bank12, C13: Bank13, C24: Bank24, C25: Bank25, C26: Bank26, C27: Bank27, C28: Bank28, C29: Bank29, C0: Bank0

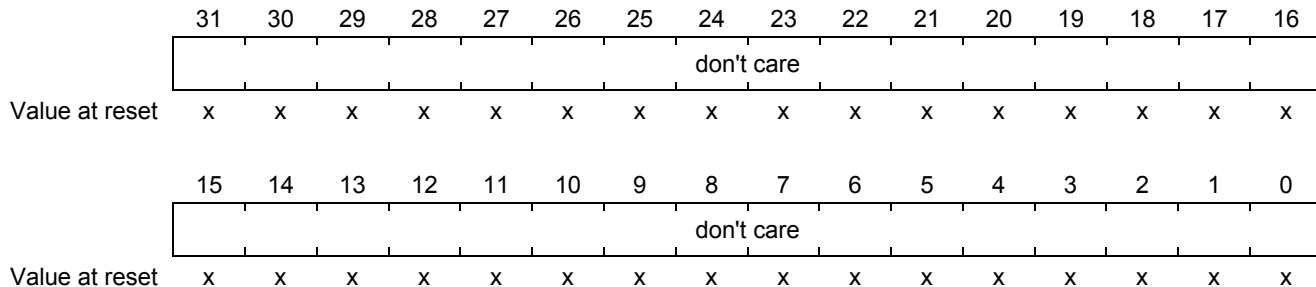
Cn	Description
0	Sets the corresponding bank non-cacheable
1	Sets the corresponding bank cacheable

(n: 8, 9, 10, 11, 12, 13, 24, 25, 26, 27, 28, 29, 0)

- C14, C15, C30, C31  
 These bits are all fixed at “0”. Always write “0s” to these bits when writing to the register. The operations cannot be guaranteed if a “1” is written to any of these bits.

### 9.2.3 FLUSH Register (FLUSH)

This FLUSH register is used for carrying out a flushing operation of the cache memory. The cache memory is initialized (flushed) by writing (any value) in the FLUSH register.



Address: CACH Base 0x1C  
 Access: W  
 Access size: 32 bits

**Bit descriptions:**

The cache memory is initialized (flushed) when any value is written into the FLUSH register.

**Notes:**

- The cache memory is not initialized for the ways that have been locked.
- Since no write back operation is made during initialization of the cache memory, if it is necessary to return the data in the cache memory to the actual memory, it will be necessary to carry out the writing back using a separate software procedure. (See Section 9.5.2.)
- Since the CPU will be made to wait until the cache memory initialization has been completed. The instruction of writing to the FLUSH register will take about 128 instruction cycles.

## 9.3 Description of Operations

### 9.3.1 Initialization of Cache Memory

It is necessary to initialize the cache memory before using this cache controller. If the cacheable setting is made without initializing the cache, correct values may not be written to or read from the memory.

### 9.3.2 Cacheable/Non-cacheable Setting

In this LSI, the entire memory space (4GB) of the CPU is split into 128MB segments called banks. There are 32 banks from Bank 0 to Bank 31. (See Chapter 3, "Address Map".) The cacheable/non-cacheable setting by the CACHE register can be made for each bank separately. The banks for which the cacheable setting can be made are the banks (Bank 0, 8 to 13, 24 to 29) of the memory space that is the target of caching (the cacheable memory area).

### 9.3.3 Description of operations

- Cache hit operation

Read hit: Data is returned from the cache memory to the CPU when there is a cache hit of the read access from the CPU.

Write hit: Data is written into the cache memory when there is a write access from the CPU.

- Cache miss operation

Read miss: When there is a cache miss of the read access from the CPU, the cache controller carries out an AHB bus access to read out one block (16 bytes) of data from the main memory, and stores that data in the cache memory and also returns that data to the CPU.

Write miss: When there is a cache miss of the write operation from the CPU, the cache controller carries out an AHB bus access to read out one block (16 bytes) of data from the main memory, and stores it in the cache memory and also stores the write data from the CPU.

- Replacement operation

When there is no free space for storing within the cache memory in the case of a cache miss, one block (16 bytes) of the data storage area that has not been used recently will be over-written by the newly required data.

If the data that is over-written has been updated, the cache memory controller carries out first a write back operation to free the storage area, and then carries out the above cache miss operation.

This type of operation is called a replacement operation.

- Non-cacheable access operation

When an access is made to an address in a bank for which the cacheable setting has not been made, the cache controller does not carry out the hit/miss judgment for the access made by the CPU, but always carries out an access to the actual memory. This type of access operation is called a non-cacheable access operation.

### 9.3.4 Lock Function

Locking is the function of retaining the contents of the cache memory within the cache memory itself as follows.

- When there is a cache hit, the normal cache hit operation is made.

- When there is a cache miss, the contents of the locked cache memory is retained by carrying out a cache miss operation to a way that has not been locked.

In general, in the cache memory, while there is the action of shortening the average access time of instructions and data, it also has the nature of lengthening the access time in the worst case when considered in the instantaneous behaviour. This is because accesses occur that are not intended by the software, such as the occurrence of a read access of one block when there is a cache read miss, or the occurrence of a write access due to a write back operation when there is a cache write miss.

In the case of programs in which the timing is very critical, such as in some types of interrupt servicing, this instantaneous increase in the access time can create problems. The lock function is used to reduce the effect of such problems.

The lock function puts a part of the cache memory in the locked state so that the instructions and data in that part of the cache memory is retained.

Since such locked instructions and data will always result in a cache hit operation, it is possible to carry out high speed access at all times.

While the cache memory is divided into four ways (storage locations), it is possible to lock the contents of the cache memory in units of a way by setting the LCK bits. Further, using the BNK[1:0] and F bits, it is possible to load instructions or data to a locked way.

Caution:

- The load operation is given priority when the target way of the load operation has been locked.  
(Example: When BNK = "00", F = "1", and LCK = "01", the target way (Way 0) of the load operation will be the same as the target way (Way 0) of the lock operation.)
- Note that Way 3 cannot be locked.

### 9.3.5 Load Mode

The load mode is that of storing data in the specified way of the cache memory by accessing the data in the main memory.

All instruction fetch accesses will be non-cacheable accesses.

On the other hand, the operations for all data accesses will be made in the normal manner according to the setting of the cacheable register (cacheable or non-cacheable).

All storing to the cache memory is made forcibly to the way selected by BNK[1:0] by the setting of the F bit.

By setting the BNK[1:0] and F bits of the cache lock control register (CON), it is possible to load data into a way that is to be locked. The data so loaded can be handled as instructions or data.

### 9.3.6 Flushing Function

The flushing function initializes the control information within the cache memory. The cache memory is flushed by writing any data in the FLUSH register. Since the cache memory is flushed by this flushing operation, after this operation, there will be no instructions or data in the cache memory. Since no write back operation is made during flushing, the cache memory will be initialized without writing back the latest instructions and data present in the cache memory into the actual memory.

Further, since the cache memory is not initialized (flushed) at the time this LSI is reset, be sure to flush the cache memory by software after resetting the LSI. (See "Examples of setting" Section.)

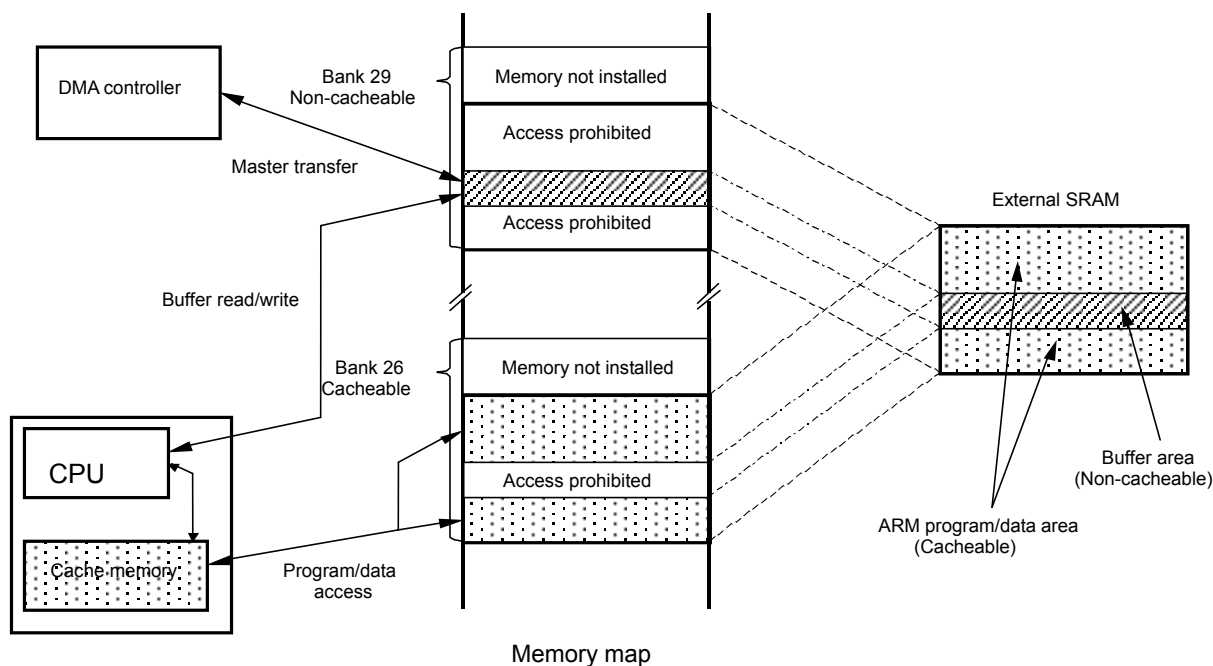
## 9.4 Precautions in Use

### 9.4.1 Precautions when Using DMA Transfer

A bank for which the cacheable setting has been made should not be set as the buffer area for carrying out DMA transfer. The reason for this is that, while DMA transfer is made with the actual memory, since the accesses from the CPU are made to the cache memory, setting a cache memory bank as a buffer for DMA transfer can cause mismatches in the contents of the memory. Therefore, use only a non-cacheable bank as the buffer area for DMA transfers.

By using the memory mirroring function, it will be possible to set the buffer area for DMA transfer within a bank for which the non-cacheable setting is made. The external memory controller of this LSI mirrors Bank 26 (external SRAM area) to Bank 29, Bank 25 (external ROM area) to Bank 28, and Bank 24 (external DRAM area) to Bank 27. For example, it is possible to access a physically single external SRAM using different addresses of Bank 26 and Bank 29. By setting one of these banks (say, Bank 26) cacheable and the other bank (that is, Bank 29) non-cacheable, it is possible to provide a cacheable area (the address area of Bank 26) and a non-cacheable area (Bank 29) in a single external SRAM.

In this example, the program and data of the CPU that need to be accessed at high speed using the cache function are placed in the addresses of Bank 26, and the buffer for DMA transfer is placed in the addresses in Bank 29. By doing this, since the program and data of the CPU are in Bank 26, they will be cacheable, while on the other hand, since the buffer accessed by DMA and ARM is in Bank 29, it will be non-cacheable.



### 9.4.2 Precautions in Remapping

After the cacheable setting is made for a remapped bank after remapping, do not access the bank before remapping. For example, when the external SRAM of Bank 26 for which the non-cacheable setting has been made is remapped to Bank 0 and Bank 0 is set cacheable, do not access the Bank 26 before remapping thereafter. This is because, since the cache is used at the time of accessing Bank 0, the contents of the memory will always be the continually updated data, but that latest data will not be updated in Bank 26 but only the old data will be remaining in it.

When setting cacheable so that the same memory is visible in several banks by remapping, always be sure to set only one of them cacheable and the other bank non-cacheable. Also, always be sure to access the bank for which the cacheable has been made. Otherwise, there will be mismatch in the contents of the memory as described above.

## 9.5 Examples of Setting

### 9.5.1 Example of Cache Memory Initialization

Before using this cache controller, it is necessary to initialize the cache memory using the following procedure. The values of memory read and write will not be correct when cacheable setting is made without initializing it.

- 1) Write 0x0000\_0000 in the cacheable register and set non-cacheable for all banks.
- 2) Write 0x0000\_0000 in the cache lock control register and disable the lock condition of all ways.
- 3) Write any value in the FLUSH register and initialize the cache memory.

(Note: No write back operation is made during the cache initialization. The instruction of writing to the FLUSH register takes about 128 instruction cycles.)

### 9.5.2 Example of Cache Memory Flushing Requiring Data Write Back

When it is necessary to write back the data in the cache memory to the main memory before flushing the cache memory, it is necessary to carry out the write back operation separately by software.

Sample procedure:

- 1) Acquire a dummy data of 8kB in continuous addresses of a cacheable space.  
(Example: Acquire 8kB of dummy data in Bank 26 so that the low-order 13 bits of the address are 0x0000-1FFC.)
- 2) Make the cacheable setting for the bank storing the dummy data by setting the cacheable register.
- 3) After putting Way 0 in the Load mode by setting the cache lock control register, carry out the read of 2kB of dummy data.  
(Example: By reading 2kB of dummy data which the low-order 13 bits of the address being 0x0000-0x07FC, write back to the main memory the data in Way 0 that needs to be written back.)
- 4) Similarly, put Way 1, Way 2, and Way 3 in the Load mode by setting the cache lock control register, and also read 2kB of dummy data.  
(All data will be written back by the data read operation of the dummy data.)

### 9.5.3 Example of Lock Setting Procedure

An example of the procedure (example of locking in Way 0) for locking instruction codes (up to 2048 contiguous bytes) in the cache is explained below.

- (1) Make cacheable setting for the bit in the cacheable register corresponding to the bank in which the instruction codes to be locked are stored.
- (2) Set F = "1" and BNK = "00" (Way 0) in the cache lock control register.
- (3) Carry out a data read operation for the addresses in which the instruction codes are stored. This causes the instruction codes to be locked to be loaded in Way 0.  
(Since the reading is done in units of 16 bytes, make sure that the total number of bytes is 2kB or less including the data before and after the starting address and the ending address of the data read operation.  
Example: When the starting address of data read is 0x4000\_000C, the address for loading will be 0x4000\_0000.)
- (4) After the data reading in (3) is completed, the target instruction codes are locked in Way 0 by setting F = "0" and LCK = "01" in the cache lock control register.

An example is shown below of the procedure (example of locking in Way 0 and Way 1) of locking in the cache the instruction codes (contiguous, 2049 bytes or more but less than or equal to 4096 bytes).

- (1) Make cacheable setting for the bit in the cacheable register corresponding to the bank in which the instruction codes to be locked are stored.
- (2) Set F = "1" and BNK = "00" (Way 0) in the cache lock control register.
- (3) Carry out a data read operation for the addresses corresponding to the first 2k bytes among the addresses storing the instruction codes. This causes the instruction codes to be locked to be loaded in Way 0.  
(Since the reading is done in units of 16 bytes, make sure that the total number of bytes is 2k bytes or less including the data before and after the starting address and the ending address of the data read operation.  
Example: When the starting address of data read is 0x4000\_000C, the address for loading will be 0x4000\_0000.)
- (4) After the data reading in (3) is completed, set F = "1" and BNK = "01" (Way 1) in the cache lock control register.
- (5) Carry out a data read operation (the addresses corresponding to the remaining instruction codes) for the addresses in which the instruction codes are stored. This causes the instruction codes to be locked to be loaded in Way 1.  
(Since the reading is done in units of 16 bytes, make sure that the total number of bytes is 2k bytes or less including the data before and after the starting address and the ending address of the data read operation.  
Example: When the starting address of data read is 0x4000\_080C, the address for loading will be 0x4000\_0800.)
- (6) After the data reading in (5) is completed, the target instruction codes are locked in Way 0 and Way 1 by setting F = "0" and LCK = "10" in the cache lock control register.

An example is shown below of the procedure (example of locking in Way 0, Way 1, and Way 2) of locking in the cache the instruction codes (contiguous, 4097 bytes or more but less than or equal to 6144 bytes).

- (1) Make cacheable setting for the bit in the cacheable register corresponding to the bank in which the instruction codes to be locked are stored.
- (2) Set F = "1" and BNK = "00" (Way 0) in the cache lock control register.
- (3) Carry out a data read operation for the addresses corresponding to the first 2k bytes among the addresses storing the instruction codes. This causes the instruction codes to be locked to be loaded in Way 0.  
(Since the reading is done in units of 16 bytes, make sure that the total number of bytes is 2k bytes or less including the data before and after the starting address and the ending address of the data read operation.  
Example: When the starting address of data read is 0x4000\_000C, the address for loading will be 0x4000\_0000.)
- (4) After the data reading in (3) is completed, set F = "1" and BNK = "01" (Way 1) in the cache lock control register.
- (5) Carry out a data read operation for the addresses corresponding to the next 2K bytes among the addresses in which the instructions codes are stored. This causes the instruction codes to be locked to be loaded in Way 1.



(Since the reading is done in units of 16 bytes, make sure that the total number of bytes is 2k bytes or less including the data before and after the starting address and the ending address of the data read operation.

Example: When the starting address of data read is 0x4000\_080C, the address for loading will be 0x4000\_0800.)

- (6) After the data reading in (5) is completed, set F = "1" and BNK = "10" (Way 2) in the cache lock control register.
- (7) Carry out a data read operation (the addresses corresponding to the remaining instruction codes) for the addresses in which the instruction codes are stored. This causes the instruction codes to be locked to be loaded in Way 2.

(Since the reading is done in units of 16 bytes, make sure that the total number of bytes is 2k bytes or less including the data before and after the starting address and the ending address of the data read operation.

Example: When the starting address of data read is 0x4000\_100C, the address for loading will be 0x4000\_1000.)

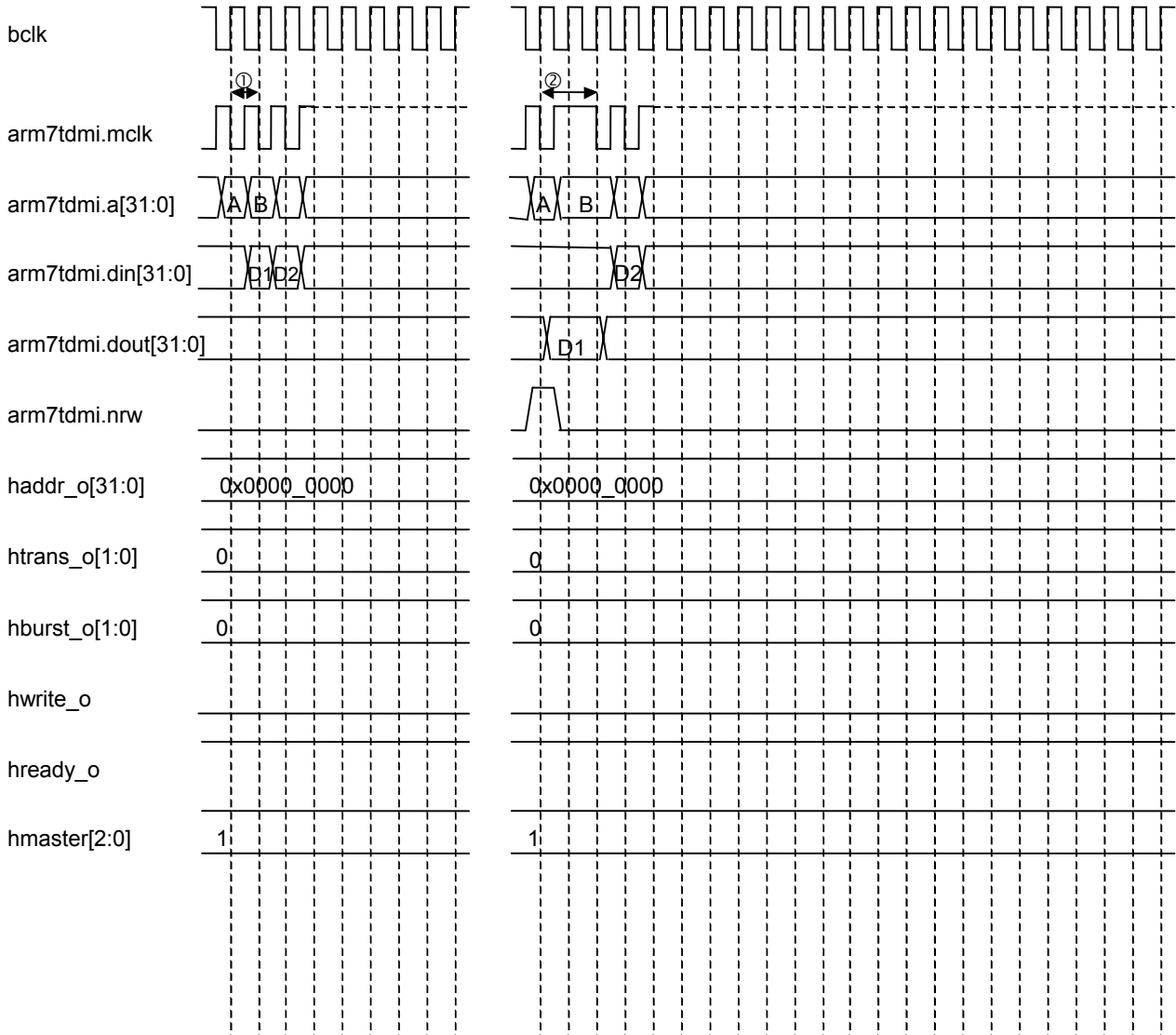
- (8) After the data reading in (7) is completed, the target instruction codes are locked in Way 0, Way 1, and Way 2 by setting F = "0" and LCK = "11" in the cache lock control register.

**Caution:** All data reads carried out during the load mode will be targets for the Load operation. Therefore, when executing the above locking procedure, take the following precautions in order to prevent loading of the data that are not to be locked.

Either set to "0" the cacheable bit of the bank storing the instruction codes for executing the above locking procedure, or prepare the program so that it consists of only instruction fetches (with no data accesses) for the instruction codes for executing the above locking procedure.

## 9.6 Typical Operation Timings

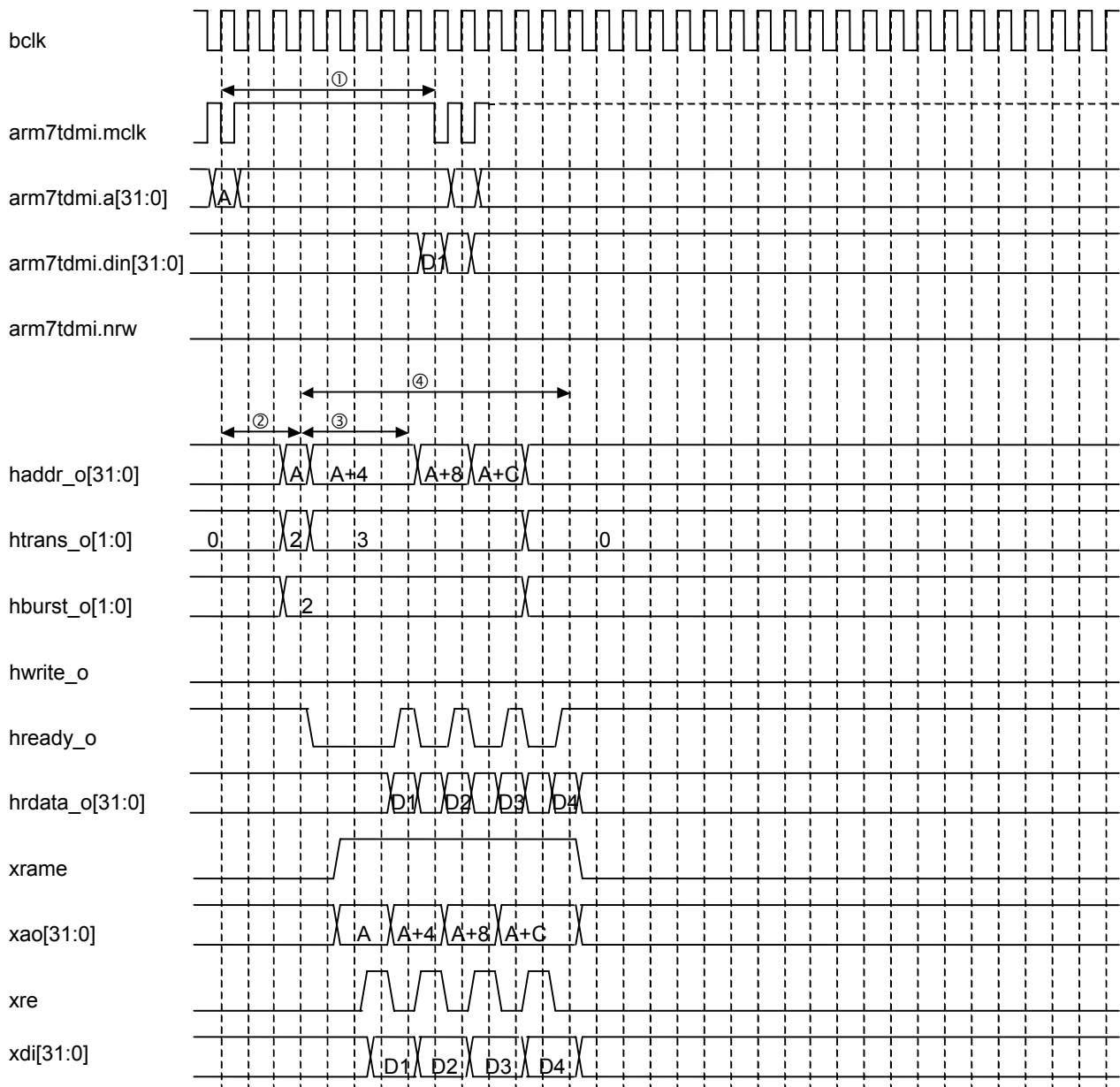
- Example of cache hit operation



- During a read hit, the data is returned to the CPU in one mclk cycle. (① in the figure.)
- During a write hit, the write operation is completed within a maximum of 2 cycles. (② in the figure.)
- During a cache hit operation, the cache controller does not require the memory access and bus access to the AHB.

(The above example is one in which the succeeding access is a read hit.)

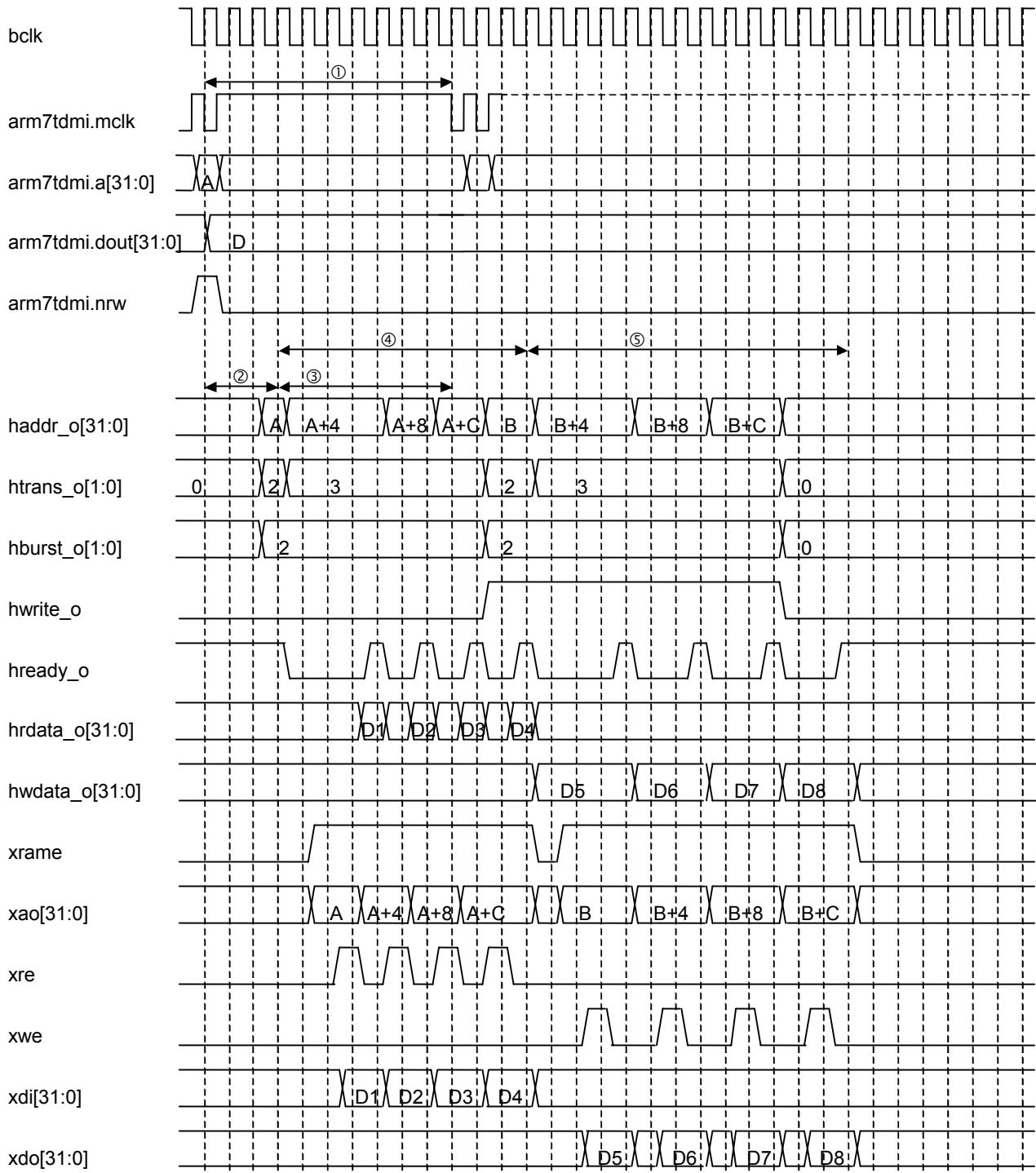
- Example of cache miss operation



- During a cache miss, the cache controller carries out a 4-word (wrap 4) AHB read access.
- The CPU will be kept waiting until the access is completed. (① in the figure.)  
 The waiting duration (① in the figure) of the CPU varies depending on the previous access state (② in the figure) and on the memory access duration (③ in the figure).
- The AHB read access is made not only in the case of a read miss operation but also in the case of a write miss operation. (④ in the figure.)

(The above example is one in which there is no write back, and is a read miss operation in the case of a succeeding read hit.)

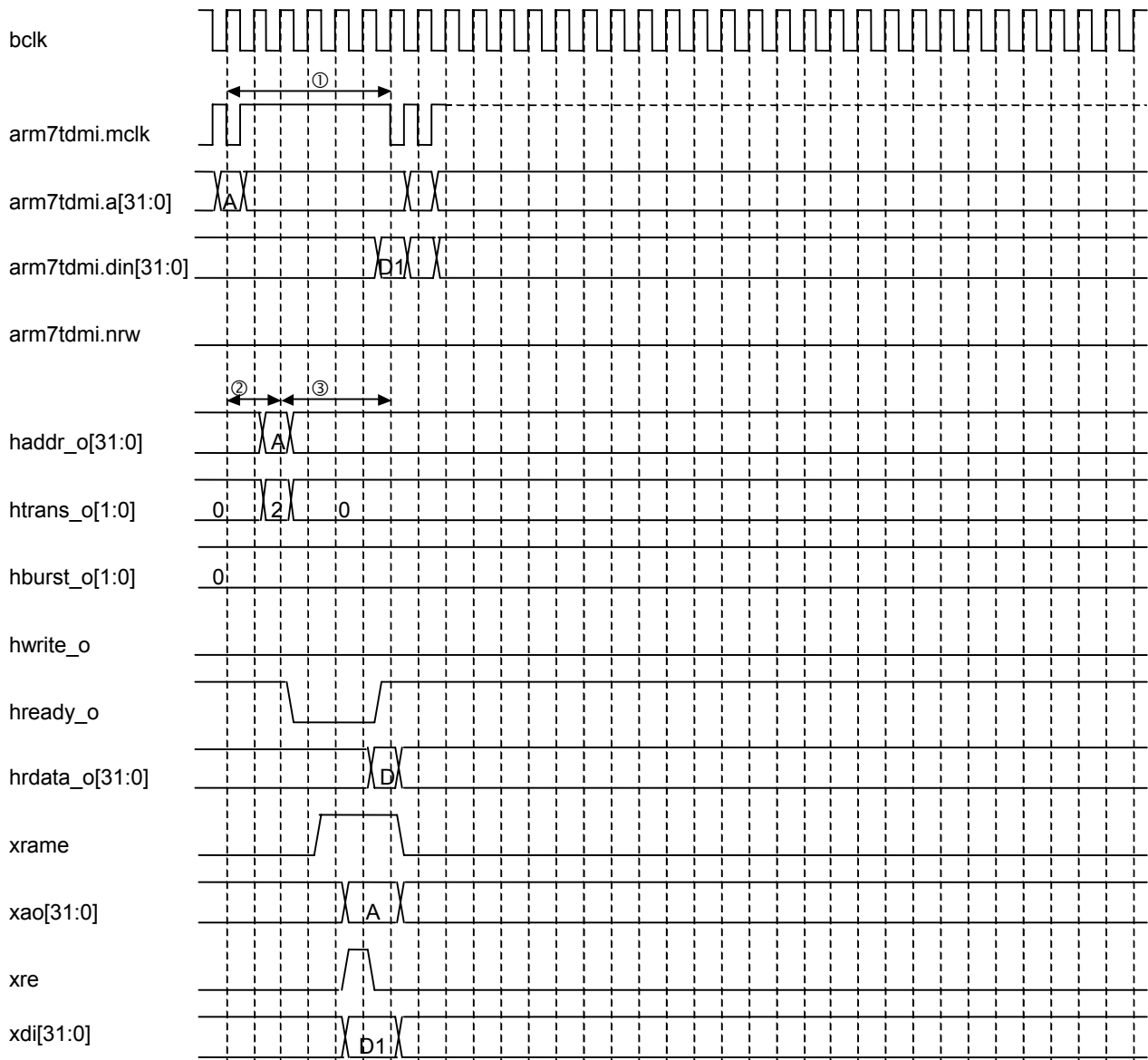
- Example of replacement operation (Write back operation is present)



- During a cache miss operation, the cache controller carries out a 4-word (wrap 4) AHB Read access. (④ in the figure.)
  - The CPU will be kept waiting until the access is completed. (① in the figure.)
- The waiting duration . (① in the figure) of the CPU varies depending on the previous access state (② in the figure), the presence or absence of write back operation, and on the memory access duration (③ in the figure).

(The above example is of an operation when the succeeding access is a read hit. The write back operation is ⑤ in the figure.)

- Example of non-cacheable operation



- During a non-cacheable access, the cache controller carries out an AHD access. (③ in the figure.)
- The CPU is kept waiting until the access is completed. (① in the figure.)  
 The waiting duration of the CPU varies depending on the previous access state (② in the figure) and on the memory access duration (③ in the figure).

(The above example is of an operation when the succeeding access is a read hit.)

## *Chapter 10*

# **Built-In Memory**

---

## Chapter 10 Built-In Memory

### 10.1 Overview

This LSI contains 32kilobytes (8K × 32 bits) of built-in SRAM and ROM Less or 256kilobytes (128K × 16 bits) or 512kilobytes (256K × 16 bits) of built-in FLASH ROM.

### 10.2 Built-In SRAM

The built-in RAM has the following features.

- 32kilobytes (8K × 32 bits)
- Bus width: 32 bits, with 8-, 16-, and 32-bit read/write access

	Address map	Access clock cycles		Bank	Mirror Bank
		read	write		
ML674001/ML67Q4002/ ML67Q4003	0x10000000 to 0x10007FFF	1	1(word access) 2(half word access) 2(byte access)	Bank 2	Bank 5
ML675001/ML67Q5002/ ML67Q5003	0x50000000 to 0x50007FFF	3(non cacheable)		Bank 10	-

Writing 0x000C to the remap control (RMPCON) register remaps bank 0 to built-in RAM for ML674001 Series.  
Writing 0x000A to the remap control (RMPCON) register remaps bank 0 to built-in RAM for ML675001 Series.

### 10.3 Built-In FLASH ROM

The built-in FLASH ROM has the following features.

- Bus width: 16 bits, with 8-, 16, and 32-bit read access
- Sample setting of memory access clock cycles:

HCLK frequency[MHz]	Access clock cycles(OE/WE pulse width)	
	[ML674001Series]	[ML675001Series]
1-13.3	1	1
13.3-26.7	2	2
26.7-33.3	3	3
33.3-40	-	3
40-53.3	-	5
53.3-60	-	5

In order to set up the number of clock cycles for access of FlashROM, please set ROMTYPE [2:0] of a ROMAC register as "OE/WE pulse width". Refer to section 11.2.3.

#### List of Flash ROM capacity

	Flash ROM capacity	Address map (Bank 25)
ML674001/ ML675001	ROM Less	-
ML67Q4002/ ML67Q5002	256kilobytes (128K 16 bits)	0xC8000000 to 0xC803FFFF
ML67Q4003/ ML67Q5003	512kilobytes (256K 16 bits)	0xC8000000 to 0xC807FFFF

If the BSEL[1:0] external pin input levels are “LL”, bank 0 (0x00000000 to 0x07FFFFFFF) is mapped to the FLASH ROM for use as the boot device after a reset.

Settings to the remap control (RMPCON) register and the ROM select register (ROMSEL) can remap bank 0 to built-in FLASH ROM.



# **External Memory Controller**

---

## Chapter 11 External Memory Controller

### 11.1 Overview

The external memory controller built into this LSI is for connecting DRAM, ROM, SRAM, I/O devices, and other external devices to this LSI. Supported devices include the following.

- Asynchronous ROM
- Asynchronous SRAM
- Flash memory
- I/O devices
- SDRAM
- EDO DRAM

#### 11.1.1 Pin List

Table 11.1 lists the pins for connecting devices to this controller.

**Table 11.1**

Pin Name	I/O	Function	Secondary Function
XA[23:19]	O	External address bus	PIOC[6:2]
XA[18:0]	O	External address bus	–
XD[15:0]	I/O	External data bus	–
XROMCS_N	O	External ROM chip select	–
XRAMCS_N	O	External RAM chip select	–
XIOCS_N[0]	O	I/O 0 chip select	–
XIOCS_N[1]	O	I/O 1 chip select	–
XIOCS_N[2]	O	I/O 2 chip select	–
XIOCS_N[3]	O	I/O 3 chip select	–
XOE_N	O	Output enable	–
XWE_N	O	Write enable	–
XBS_N[1:0]	O	External bus (XBUS) byte select	–
XBWE_N[0]	O	Write enable (LSB)	–
XBWE_N[1]	O	Write enable (MSB)	–
XSDCS_N	O	SDRAM chip select	PIOD[4]
XSDCLK	O	SDRAM clock	PIOD[3]
XSDCKE	O	SDRAM clock enable	PIOD[5]
XCAS_N	O	SDRAM column address strobe	PIOD[1]
XRAS_N	O	Row address strobe	PIOD[2]
XDQM[1]/XCAS_N[1]	O	Data I/O mask for SDRAM or MSB column address strobe for EDO DRAM	PIOD[6]
XDQM[0]/XCAS_N[0]	O	Data I/O mask for SDRAM or LSB column address strobe for EDO DRAM	PIOD[7]
XWAIT	I	Memory wait indicator	PIOD[0]
XWR	O	External bus (XBUS) transfer direction	PIOC[7]

11.1.2 Register List

Address	Name	Abbreviation	R/W	Size	Initial Setting
0x78100000	Bus width control register	BWC	R/W	32	0x00000008
0x78300000*1	External I/O bank 2/3 Bus width control register	IO23BWC	R/W	32	0x00000000
0x78100004	External ROM access control register	ROMAC	R/W	32	0x00000007
0x78100008	External SRAM access control register	RAMAC	R/W	32	0x00000007
0x7810000C	External I/O bank 0/1 access control register	IO01AC	R/W	32	0x00000007
0x78100010*2	External I/O bank 2/3 access control register X	IO23ACX	R/W	32	0x00000007
0x7830000C*1	External I/O bank 2/3 access control register Y	IO23ACY	R/W	32	0x00000007
0x78180000	DRAM bus width control register	DBWC	R/W	32	0x00000003
0x78180004	DRAM control register	DRMC	R/W	32	0x00000000
0x78180008	DRAM characteristics control register	DRPC	R/W	32	0x00000000
0x7818000C	SDRAM mode register	SDMD	R/W	32	0x00000001
0x78180010	DRAM command register	DCMD	R/W	32	0x00000000
0x78180014	DRAM refresh cycle control register 0	RFSH0	R/W	32	0x00000000
0x78180018	DRAM power down control register	RDWC	W	32	0x00000003
0x7818001C	DRAM refresh cycle control register 1	RFSH1	R/W	32	0x00000000

\*1:ML675001 Series only

\*2:ML674001 Series only

## 11.2 Register Descriptions

### 11.2.1 Bus Width Control Register (BWC)

This register specifies the external data bus widths for four banks/regions. The program has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BWC	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	IO23BW[1:0] *1	IO01BW[1:0]	RAMBW[1:0]	ROMBW[1:0]	-*	-*				
	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Address: 0x78100000

Access: R/W

Access size: 32 bits

#### Note

- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

\*1: This IO23BW[1:0] register exists only in ML674001 series. In ML675001 series, it becomes the treatment of reserved "-\*".

#### Bit Descriptions

- **ROMBW[1:0]** (bits 2 and 3):  
These bits specify the bus width for the ROM region.

ROMBW[1:0]		Description
3	2	
0	0	Not physically present (Access to bank/region disabled. Access from CPU produces abort response.)
0	1	8 bits (only for ML675001 series, reserved for ML674001 series)
1	0	16 bits
1	1	(reserved)

**Note:** The only bus width supported is 16 bits. Operation is not guaranteed for a setting labeled "reserved."

- **RAMBW[1:0]** (bits 4 and 5):  
 These bits specify the bus width for the SRAM region.

RAMBW[1:0]		Description
5	4	
0	0	Not physically present (Access to bank/region disabled. Access from CPU produces abort response. Access from DMA controller produces error response.)
0	1	8 bits (only for ML675001 series, reserved for ML674001 series)
1	0	16 bits
1	1	(reserved)

**Note:** The only bus width supported is 16 bits. Operation is not guaranteed for a setting labeled "reserved."

- **IO01BW[1:0]** (bits 6 and 7):  
 These bits specify the bus width for I/O bank 0/1.

IO01BW[1:0]		Description
7	6	
0	0	Not physically present (Access to bank/region disabled. Access from DMA controller produces error response.)
0	1	8 bits
1	0	16 bits
1	1	(reserved)

**Note:** Operation is not guaranteed for a setting labeled "reserved."

- **IO23BW[1:0]** (bits 8 and 9):  
 These bits specify the bus width for I/O bank 2/3.

IO23BW[1:0]		Description
9	8	
0	0	Not physically present (Access to bank/region disabled. Access from DMA controller produces error response.)
0	1	8 bits
1	0	16 bits
1	1	(reserved)

**Note:** Operation is not guaranteed for a setting labeled "reserved."

\*1: This IO1BW[1:0] register exists only in ML674001 series.

\*1: In ML675001 series, it becomes the treatment of reserved "-\*".

### 11.2.2 External I/O bank 2/3 Bus Width Control Register (IO23BWC) (\*1: ML675001 Series only)

This register specifies the external data bus widths for I/O bank 2/3.  
The program has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IO23BWC	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—*	—*	—*	—*	—*	—*	—*	—*	IO23BW[1:0]	—*	—*	—*	—*	—*	—*	—*
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0x78300000  
Access: R/W  
Access size: 32 bits

**Note**

—\*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

**Bit Descriptions**

- **IO23BW[1:0]** (bits 7 and 6):  
These bits specify the bus width for I/O bank 2/3.

IO23BW[1:0]		Description
9	8	
0	0	Not physically present (Access to bank/region disabled. Access from DMA controller produces error response.)
0	1	8 bits
1	0	16 bits
1	1	(reserved)

**Note:** Operation is not guaranteed for a setting labeled "reserved."  
This IO23BW[1:0] register exists only in ML675001 series.

### 11.2.3 External ROM Access Control Register (ROMAC)

This register controls ROM access timing.  
 The program has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ROMAC	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	ROM BRST *1	-*	ROMTYPE[2:0]		
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Address: 0x78100004  
 Access: R/W  
 Access size: 32 bits

#### Notes

-\*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

When switching operating frequencies, adjust the contents of this register at the lower frequency--that is, AFTER a change from high speed to low speed and BEFORE one from low speed to high speed.

#### Bit Descriptions

- **ROMTYPE[2:0]** (bits 0 to 2):  
 These bits specify the ROM access timing.

This table is applied only to ML674001 series.

ROMTYPE[2:0]			Pulse width in clock cycles		Notes
2	1	0	OE/WE pulse width	Read off timing <sup>1</sup>	
0	0	0	1	0	
0	0	1	2	0	
0	1	0	3	2	
0	1	1	4	2	
1	0	0	(reserved)		Operation is not guaranteed for a setting labeled "reserved."
1	0	1	(reserved)		Operation is not guaranteed for a setting labeled "reserved."
1	1	0	(reserved)		Operation is not guaranteed for a setting labeled "reserved."
1	1	1	8	4	

1: Read off time is the interval measured from, the time when Output Enable (OE) becomes inactive in one memory cycle to the time it becomes active in the next memory cycle.

This table is applied only to ML675001 series.

ROMTYPE[2:0]			Pulse width in clock cycles				Notes
2	1	0	Address setup time	OE/WE pulse width	Read off time <sup>1</sup>	Burst time	
0	0	0	1	1	1	1	
0	0	1	1	2	2	2	
0	1	0	(reserved)				Operation is not guaranteed for a setting labeled "reserved."
0	1	1	1	5	3	3	
1	0	0	2	8	4	5	
1	0	1	2	10	5	6	
1	1	0	(reserved)				Operation is not guaranteed for a setting labeled "reserved."
1	1	1	2	16	7	9	

- **ROMBRST** (bit 4):  
Page Mode is set up.

ROMBRST	Description
0	Page Mode off
1	Page Mode on

\*1 This ROMBRST register exists only in ML675001 series.  
 In ML674001 series, it becomes the treatment of reserved "-\*".



### 11.2.4 External SRAM Access Control Register (RAMAC)

This register controls SRAM access timing.  
 The program has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RAMAC	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	RAM BRST *1	-*	RAMTYPE[2:0]		
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Address: 0x78100008  
 Access: R/W  
 Access size: 32 bits

#### Notes

-\*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

When switching operating frequencies, adjust the contents of this register at the lower frequency--that is, AFTER a change from high speed to low speed and BEFORE one from low speed to high speed.

#### Bit Descriptions

- **RAMTYPE[2:0]** (bits 0 to 2):  
 These bits specify the SRAM access timing.

This table is applied only to ML674001 series.

RAMTYPE[2:0]			Pulse width in clock cycles		Notes
2	1	0	OE/WE pulse width	Read off timing <sup>1</sup>	
0	0	0	1	0	
0	0	1	2	0	
0	1	0	3	2	
0	1	1	4	2	
1	0	0	(reserved)		Operation is not guaranteed for a setting labeled "reserved."
1	0	1	(reserved)		Operation is not guaranteed for a setting labeled "reserved."
1	1	0	(reserved)		Operation is not guaranteed for a setting labeled "reserved."
1	1	1	8	4 (3) <sup>2</sup>	

- 1: Read off time is the interval measured from, the time when Output Enable (OE) becomes inactive in one memory cycle to the time it becomes active in the next memory cycle.
- 2: Read off timing is 3 in case of Accessing by DMA

This table is applied only to ML675001 series.

RAMTYPE[2:0]			Pulse width in clock cycles				Notes
2	1	0	Address setup time	OE/WE pulse width	Read off time <sup>1</sup>	Burst time	
0	0	0	1	1	1	1	
0	0	1	1	2	2	2	
0	1	0	(reserved)				Operation is not guaranteed for a setting labeled "reserved."
0	1	1	1	5	3	3	
1	0	0	2	8	4	5	
1	0	1	2	10	5	6	
1	1	0	(reserved)				Operation is not guaranteed for a setting labeled "reserved."
1	1	1	2	16	7	9	

- **RAMBRST** (bit 4):  
Page Mode is set up.

RAMBRST	Description
0	Page Mode off
1	Page Mode on

\*1 AMBRST exists only in ML675001 series.

### 11.2.5 External I/O Bank 0/1 Access Control Register (IO01AC)

This register controls the access timing for I/O bank 0/1.  
 The program has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IO01AC	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	IO01TYPE[2:0]		
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Address: 0x7810000C  
 Access: R/W  
 Access size: 32 bits

**Notes**

—\*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

When switching operating frequencies, adjust the contents of this register at the lower frequency—that is, AFTER a change from high speed to low speed and BEFORE one from low speed to high speed.

**Bit Descriptions**

- **IO01TYPE[2:0]** (bits 0 to 2):  
 These bits specify the access timing for I/O bank 0/1.

This table is applied only to ML674001 series.

IO01TYPE[2:0]			Pulse width in clock cycles			Notes
2	1	0	Address setup time	OE/WE pulse width	Read off time <sup>1</sup>	
0	0	0	1	1	1	
0	0	1	1	4	2	
0	1	0	(reserved)			Operation is not guaranteed for a setting labeled "reserved."
0	1	1	2	8	4	
1	0	0	2	12	6	
1	0	1	2	16	7	
1	1	0	(reserved)			Operation is not guaranteed for a setting labeled "reserved."
1	1	1	4	24	10	

This table is applied only to ML675001 series.

IO01TYPE[2:0]			Pulse width in clock cycles			Notes
2	1	0	Address setup time	OE/WE pulse width	Read off time <sup>1</sup>	
0	0	0	1	1	1	
0	0	1	1	4	3	
0	1	0	(reserved)			Operation is not guaranteed for a setting labeled "reserved."
0	1	1	2	8	5	
1	0	0	2	12	7	
1	0	1	2	16	8	
1	1	0	(reserved)			Operation is not guaranteed for a setting labeled "reserved."
1	1	1	4	24	11	

### 11.2.6 External I/O Bank 2/3 Access Control Register (IO23ACX \*1, IO23ACY \*2)

This register controls the access timing for I/O bank 2/3.  
 The program has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IO23AC	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	IO23TYPE[2:0]		
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Address: 0x78100010(IO23ACX \*1), 0x7830000C(IO23ACY \*2)

Access: R/W

Access size: 32 bits

#### Notes

—\*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

When switching operating frequencies, adjust the contents of this register at the lower frequency—that is, AFTER a change from high speed to low speed and BEFORE one from low speed to high speed.

\*1: ML674001 Series only

\*2: ML675001 Series only

#### Bit Descriptions

- **IO23TYPE[2:0]** (bits 0 to 2):  
 These bits specify the access timing for I/O bank 2/3.

This table is applied only to ML674001 series.

IO23TYPE[2:0]			Pulse width in clock cycles			Notes
2	1	0	Address setup time	OE/WE pulse width	Read off time <sup>1</sup>	
0	0	0	1	1	1	
0	0	1	1	4	2	
0	1	0	(reserved)			Operation is not guaranteed for a setting labeled "reserved."
0	1	1	2	8	4	
1	0	0	2	12	6	
1	0	1	2	16	7	
1	1	0	(reserved)			Operation is not guaranteed for a setting labeled "reserved."
1	1	1	4	24	10	

This table is applied only to ML675001 series.

IO23TYPE[2:0]			Pulse width in clock cycles			Notes
2	1	0	Address setup time	OE/WE pulse width	Read off time <sup>1</sup>	
0	0	0	1	1	1	
0	0	1	1	4	3	
0	1	0	(reserved)			Operation is not guaranteed for a setting labeled "reserved."
0	1	1	2	8	5	
1	0	0	2	12	7	
1	0	1	2	16	8	
1	1	0	(reserved)			Operation is not guaranteed for a setting labeled "reserved."
1	1	1	4	24	11	

### 11.2.7 DRAM Bus Width Control Register (DBWC)

This register specifies the bus width for the DRAM region.  
 The program has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBWC	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	BWDRAM [1:0]	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Address: 0x78180000

Access: R/W

Access size: 32 bits

#### Notes

—\*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

The contents of this register do not affect the SDRAM clock output XSDCLK.

#### Bit Descriptions

- BWDRAM (bits 0 and 1): These bits specify the bus width for the DRAM region.

BWDRAM[1:0]		Description
1	0	
0	0	Not physically present (Access produces error response.)
0	1	8 bits (only possible for EDO DRAM)
1	0	16 bits
1	1	(reserved)

### 11.2.8 DRAM Control Register (DRMC)

This register specifies the DRAM type, access timing, and other settings.  
The program has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DRMC	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	RF RSH	P DWN	-*	PRE LAT	-*	ARCH	AMUX[1:0]	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0x78180004

Access: R/W

Access size: 32 bits

#### Note

-\*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

#### Bit Descriptions

- **AMUX[1:0]** (bits 0 and 1):  
These bits specify the column length, which is the dividing, line for address multiplexing.

AMUX[1:0]		Description		
1	0	Column length	RAS address	CAS address
0	0	8 bits	A[23:8]	A[7:0]
0	1	9 bits	A[23:9]	A[8:0]
1	0	10 bits	A[23:10]	A[9:0]
1	1	(reserved)		

- **ARCH** (bit 2):  
This bit specifies the DRAM type.

ARCH	Description
0	SDRAM
1	EDO-DRAM

- **PRELAT** (bit 4):  
This bit specifies the SDRAM pre-charge latency.

PRELAT	Description
0	Fixed at 2 clock cycles
1	Use CAS latency setting



- **PDWN** (bit 6):  
This bit controls automatic shifting to SDRAM power down mode.

<b>PDWN</b>	<b>Description</b>
0	Disable automatic shift
1	Enable automatic shift

- **RFRSH** (bit 7): This bit controls distributed CAS before RAS (CBR) refresh operation.

<b>RFRSH</b>	<b>Description</b>
0	Stop distributed CBR refresh
1	Enable distributed CBR refresh

### 11.2.9 DRAM Characteristics Control Register (DRPC)

This register specifies the DRAM access timing parameters in clock cycles.  
The program has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DRPC	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	DRAMSPEC[3:0]			
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Access: 0x78180008

Address: R/W

Access size: 32 bits

#### Note

- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

#### Bit Descriptions

- **DRAMSPEC[3:0]** (bits 0 to 3): These bits specify the DRAM access timing parameters in clock cycles.

DRAMSPEC[3:0]				SDRAM operation				EDO DRAM operation				
3	2	1	0	tRCD	tRAS	tRP	tDPL	tRAH tCAS	tRCD	tCAC tOEZ	tRP	
0	0	0	0	1	2	1	1	1	2	1	1	<div style="display: flex; flex-direction: column; align-items: center;"> <div style="display: flex; justify-content: space-between; width: 100%;"> <span>High speed</span> <span>(Low frequency)</span> </div> <div style="margin: 10px 0;"> <span style="font-size: 2em;">↑</span> </div> <div style="margin: 10px 0;"> <span style="font-size: 2em;">↓</span> </div> <div style="display: flex; justify-content: space-between; width: 100%;"> <span>Low speed</span> <span>(High frequency)</span> </div> </div>
0	0	0	1	1	3	1	1	1	2	1	2	
0	0	1	0	2	3	2	1	1	3	1	2	
0	0	1	1	2	4	2	1	1	3	1	3	
0	1	0	0	2	4	2	2	1	3	2	3	
0	1	0	1	2	5	2	1	1	4	2	4	
0	1	1	0	2	5	2	2	1	5	2	5	
0	1	1	1	2	5	3	1	2	4	2	4	
1	0	0	0	3	5	3	2	2	5	2	5	
1	0	0	1	3	6	3	2	2	6	2	6	
1	0	1	0	(reserved)				3	8	3	7	
1	0	1	1	(reserved)				(reserved)				
1	1	0	0	(reserved)				(reserved)				
1	1	0	1	(reserved)				(reserved)				
1	1	1	0	(reserved)				(reserved)				
1	1	1	1	(reserved)				(reserved)				

#### Notes

1. Operation is not guaranteed for a setting labeled "reserved."
2. Choose a setting with a combination satisfying the access timing parameters (tRCD, tRAS, tRP, etc.) appearing in the DRAM data sheet.

### 11.2.10 SDRAM Mode Register (SDMD)

This register specifies the SDRAM CAS latency.  
 The program has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SDMD	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—*	—*	—*	—*	—*	—*	—*	—*	MODE WR	—*	—*	—*	—*	—*	—*	LT MODE
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Address: 0x781800C  
 Access: R/W  
 Access size: 32 bits

#### Notes

- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.
- Writing "1" to MODEWR in this register produces an SDRAM setting cycle. MODEWR always returns "0" for reads.

#### Bit Descriptions

- LTMODE** (bit 0):  
 This bit specifies the SDRAM CAS latency in clock cycles.

LTMODE	Description
0	2
1	3

- MODEWR** (bit 7):  
 Writing "1" to this bit produces an SDRAM setting cycle. It always returns "0" for reads.

MODEWR	Description
0	Ignore new settings
1	Use new settings

This setting cycle has the following format.

A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
OPECODE					LTMODE			WT	BL		

\* Axx are the SDRAM input address signals.

Field	Function	Setting	Meaning
OPCODE	Mode option	00000	Burst Read & Burst Write
LTMODE	CAS latency	010	2 clock cycles or
		011	3 clock cycles
WT	Wrap type	0	Sequential (Wrap Around)
BL	Burst length	011	Burst length 8

**Note:** LTMODE is the only field that varies--and then only by a single bit.

### 11.2.11 DRAM Command Register (DCMD)

Writing to this register executes the specified command (SDRAM) or produces the corresponding access (EDO DRAM).

The program has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DCMD	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	DRCMD[2:0]		
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0x78180010

Access: R/W

Access size: 32 bits

**Note**

- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

**Bit Descriptions**

- **DRCMD[2:0]** (bits 0 to 2):  
 These bits specify the DRAM command to execute.

DRCMD[2:0]			SDRAM Operation	EDO DRAM Operation
2	1	0		
0	0	0	No Operation	No Operation
0	0	1	(reserved)	(reserved)
0	1	0	(reserved)	(reserved)
0	1	1	(reserved)	(reserved)
1	0	0	PALL (pre-charge all SDRAM banks)	EDO DRAM pre-charge
1	0	1	REF (SDRAM CBR refresh)	EDO DRAM CBR refresh
1	1	0	SELF (start SDRAM self refresh)	Start EDO DRAM self refresh
1	1	1	SREX (end SDRAM self refresh)	End EDO DRAM self refresh

**Notes**

1. Operation is not guaranteed for a setting labeled "reserved."
2. During self refresh operation, operation is not guaranteed for commands other than the end self refresh command.
3. The SDRAM clock output SDCLK
  - \* Stops (Low level) for EDO DRAM operation (ARCH bit in DRMC register = 1)
  - \* Stops (Low level) after SELF command
  - \* Resumes after SREX command
4. The hardware blocks access to this register until the specified command finishes execution.
5. According to JEDEC standards, a DRAM chip requires 8 cycles of CBR to get properly initialized. Thus, when initializing a JEDEC standard DRAM, the application software must write '0x05', eight times to this register to initialize CBR for DRAM.

### 11.2.12 DRAM Refresh Cycle Control Register 0 (RFSH0)

This register specifies the DRAM refresh period relative to that specified in the RFSH1 register: double or same.

The program has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RFSH0	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	RCCON
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0x78180014

Access: R/W

Access size: 32 bits

#### Notes

-\*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

DRAM controller operation is not guaranteed after writes to this register when there is no DRAM physically present (DBWC = 0).

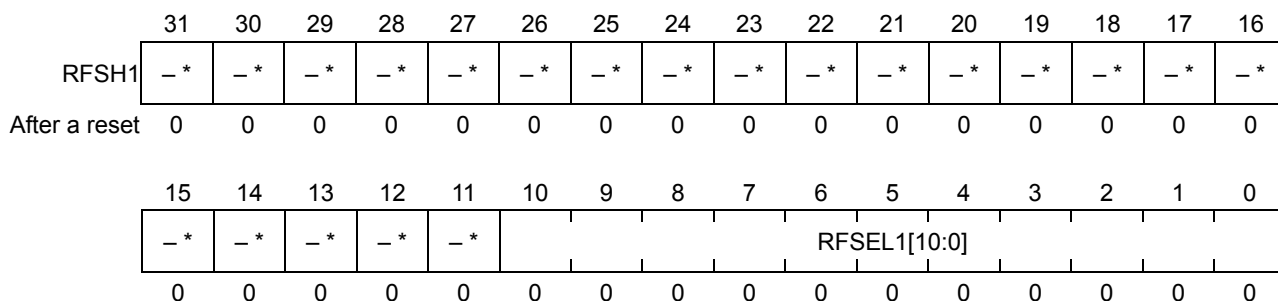
#### Bit Descriptions

- **RCCON** (bit 0):  
This bit specifies the DRAM refresh period relative to that specified in the RFSH1 register.

RCCON	Description
0	Refresh using twice clock period specified in RFSH1 register
1	Refresh using the clock period specified in RFSH1 register

### 11.2.13 DRAM Refresh Cycle Control Register 1 (RFSH1)

This register specifies the divider for deriving the DRAM refresh period from the CCLK period.  
 The program has read/write access to this register.



Address: 0x7818001C  
 Access: R/W  
 Access size: 32 bits

**Note**

– \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

**Bit Descriptions**

- **RFSEL1[10:0]** (bits 0 to 10):  
 These bits specify the divider for deriving the DRAM refresh period from the CCLK frequency using the following formula.

$$\text{refresh clock period} = \text{CCLK} / \text{RFSEL1}[10:0]$$

**Note**

The RFSH1 register should be set before modifying CCLK divider settings.

The following table lists the results for typical combinations of the RFSEL1 setting and CCLK frequency.

The RFSEL1 setting must be within the range shown in this Table: 0x00000008 to 0x00000406. Choose it to produce a refresh frequency of at least 32 kHz or 64 kHz.

CCLK(MHz)		0.56	2	4	8	10	20	25	33	60
RFSEL1 setting	Divisor	kHz	kHz	kHz	kHz	kHz	kHz	kHz	kHz	kHz
0008	8	70.00	250.00	500.00	1000.00	1250.00	2500.00	3125.00	4125.00	7500.00
0010	16	35.00	125.00	250.00	500.00	625.00	1250.00	1562.50	2062.50	3750.00
001F	31	18.06	64.52	129.03	258.06	322.58	645.16	806.45	1064.52	1935.48
003E	62	9.03	32.26	64.52	129.03	161.29	322.58	403.23	532.26	967.74
007C	124	4.52	16.13	32.26	64.52	80.65	161.29	201.61	266.13	483.87
009C	156	3.59	12.82	25.64	51.28	64.10	128.21	160.26	211.54	384.62
00F9	249	2.25	8.03	16.06	32.13	40.16	80.32	100.40	132.53	240.96
0138	312	1.79	6.41	12.82	25.64	32.05	64.10	80.13	105.77	192.31
0186	390	1.44	5.13	10.26	20.51	25.64	51.28	64.10	84.62	153.85
0203	515	1.09	3.88	7.77	15.53	19.42	38.83	48.54	64.08	116.50
0270	624	0.90	3.21	6.41	12.82	16.03	32.05	40.06	52.88	96.15
030C	780	0.72	2.56	5.13	10.26	12.82	25.64	32.05	42.31	76.92
03A9	937	0.60	2.13	4.27	8.54	10.67	21.34	26.68	35.22	64.03
0406	1030	0.54	1.94	3.88	7.77	9.71	19.42	24.27	32.04	58.25
0752	1874	0.30	1.07	2.13	4.27	5.34	10.67	13.34	17.61	32.02

 The shading indicates recommended settings.

**Note:** The user application system must modify the contents of this register each time that it changes the CCLK frequency with the clock gear.



### 11.2.14 DRAM Power Down Control Register (RDWC)

This register specifies the number of idle cycles to wait before shifting DRAM to power down mode.

The program has only write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDWC	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	PDCNT[3:0]			
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Address: 0x78180018

Access: W

Access size: 32 bits

**Note**

—\*: These bits are reserved for future expansion. Writes to them are ignored.

**Bit Descriptions**

- **PDCNT[3:0]** (bits 0 to 3):  
 This number plus one specifies the number of idle cycles to wait before shifting DRAM to power down mode.

PDCNT				Description
3	2	1	0	
0	0	0	0	Shift DRAM to power down mode after 1 or more cycles
0	0	0	1	Shift DRAM to power down mode after 2 or more cycles
0	0	1	0	Shift DRAM to power down mode after 3 or more cycles
0	0	1	1	Shift DRAM to power down mode after 4 or more cycles
0	1	0	0	Shift DRAM to power down mode after 5 or more cycles
0	1	0	1	Shift DRAM to power down mode after 6 or more cycles
0	1	1	0	Shift DRAM to power down mode after 7 or more cycles
0	1	1	1	Shift DRAM to power down mode after 8 or more cycles
1	0	0	0	Shift DRAM to power down mode after 9 or more cycles
1	0	0	1	Shift DRAM to power down mode after 10 or more cycles
1	0	1	0	Shift DRAM to power down mode after 11 or more cycles
1	0	1	1	Shift DRAM to power down mode after 12 or more cycles
1	1	0	0	Shift DRAM to power down mode after 13 or more cycles
1	1	0	1	Shift DRAM to power down mode after 14 or more cycles
1	1	1	0	Shift DRAM to power down mode after 15 or more cycles
1	1	1	1	Shift DRAM to power down mode after 16 or more cycles

## 11.3 Operational Description

### 11.3.1 Bus Width

The bus width control (BWC) register offers two bus width settings (0 and 16) for the external ROM and SRAM banks/regions and three (0, 8, and 16) for the external I/O banks.

**Note:**

A bus width setting of 0 means "not present."

After a reset, the contents (0x0008) specify that only ROM is present, accessible over a 16-bit bus. All banks/regions support 16-bit access.

The table shows which addressable regions support 8-bit access as well. Operation is not guaranteed for a bank/region with an X in the 8-bit column.

Bank/Region	Bus Width			
	8 bits		16 bits	
	ML674001Series	ML675001Series	ML674001Series	ML675001Series
ROM	X	O	O	O
RAM	X	O	O	O
IO0, IO1, IO2, IO3	O	O	O	O
SDRAM	X	X	O	O
EDO-DRAM	O	O	O	O

If the data size differs from the bus width, access uses the bus width. A word (32-bit) read over a 16-bit bus, for example, becomes two reads: first the lower 16 bits and then the upper ones. As a result, this 32-bit access produces two XOE\_N or XWE\_N output signals. For further details, see the timing charts in Figure 11.1, Figure 11.2, etc.

### 11.3.2 ROM/SRAM Control

1. To access SRAM, set the SRAM bus width to 16 bits in the BWC register.
2. Specify the XOE\_N/XWE\_N pulse width in the ROMAC or RAMAC register.

After a reset, the ROM bus width is 16 bits.

### 11.3.3 I/O Banks Control

1. To access I/O Banks, set the I/O Banks bus width to 8 or 16 bits in the BWC register.
  2. Specify the XOE\_N/XWE\_N pulse width in the IO01AC or IO23ACX\*/IO23ACY\* register.
- \*: IO23ACX is for ML674001 series only. IO23ACY is for ML675001 series only.

- XWR

Indicates the direction of data transfer on XD (Data Bus) for I/O Banks.

Low : read (from the external I/O device to the chip)

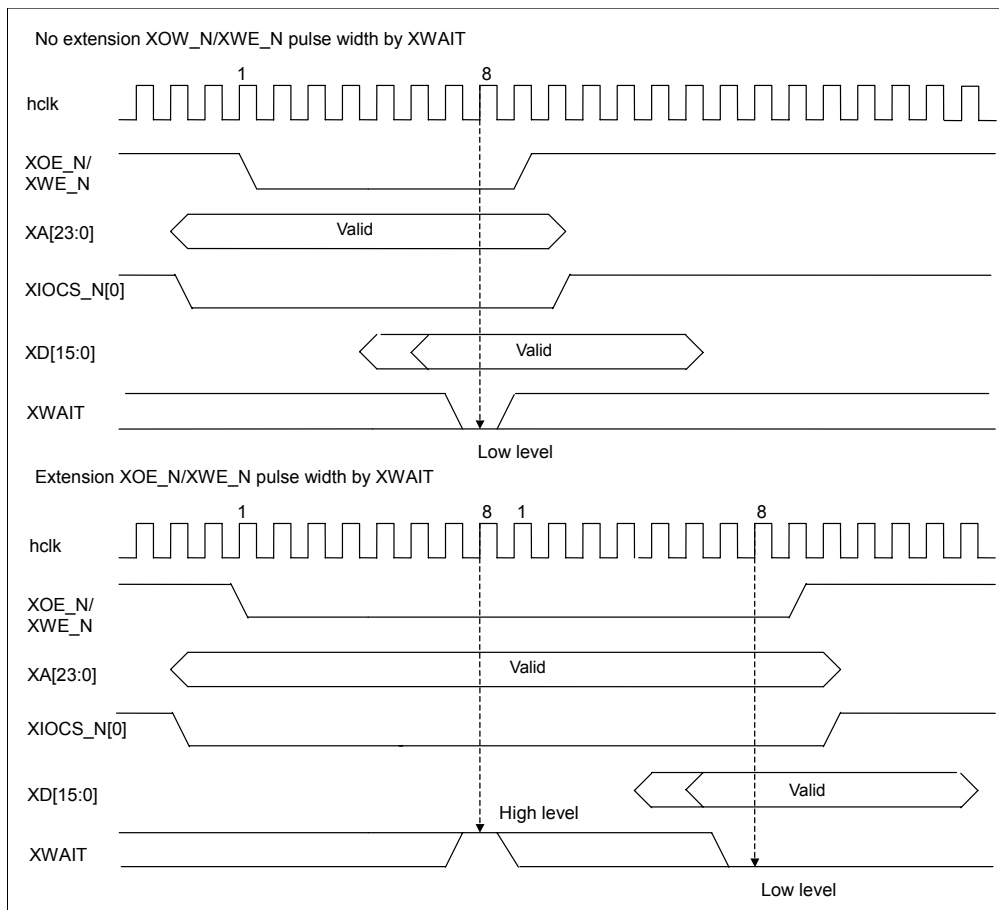
High : write (from the chip to the external I/O device)

\*: In the ML675001 series, XWR is available for only I/O 0/1.

- XWAIT (For I/O Bank 0/1/2/3)

XWAIT input is for extending the I/O device access time. XWAIT is sampled one system clock before XOE\_N/XWE\_N is de-asserted. If XWAIT is at high level at sampling time, the XOE\_N/XWE\_N pulse width will be extended for additional time based on parameters specified in the IO01AC/IO23ACX/IO23ACY register. When XWAIT is at low level at sampling time, XOE\_N/XWE\_N pulse width is not extended.

An example is shown below, when the OE/WE pulse width is set as 8 clocks in the IO01AC register.



#### 11.3.4 DRAM Control

##### DRAM Controller disable by DRAME\_N pin

For the DRAM controller to be operational, the DRAME\_N Pin of the MCU should be configured so as to enable the DRAM controller. The DRAME\_N Pin configuration is described in Chapter 4 --“Chip Configuration”--, of this manual.

When the DRAME\_N Pin of the MCU is configured so as to disable the DRAM controller, the DRAM controller registers will become inaccessible as well.

##### DRAM initialization by software

DRAM must be initialized when the power is first applied with a sequence similar to the following.

1. Wait at least 200  $\mu$ s after the end of the reset sequence before enabling DRAM access by writing a nonzero bus width to the DBWC register.

This interval is actually only necessary when the power is first applied, but taking this approach simplifies the program.

2. Specify the DRAM access timing parameters in clock cycles in the DRPC register.
3. Write 0x00000004 to the DCMD register to pre-charge the DRAM (all banks for SDRAM).
4. Write 0x00000005 to the DCMD register eight times to produce CAS before RAS (CBR) refresh operations.
5. Specify the DRAM type (SDRAM or EDO DRAM) and the column length for address multiplexing in the DRMC register. For SDRAM, also specify the SDRAM pre-charge latency and whether to automatically shift SDRAM to power down mode.
6. For SDRAM, specify the SDRAM CAS latency in the SDMD register.
7. Specify the DRAM refresh periods in the RFSH0 and RFSH1 registers.
8. Specify the number of idle cycles to wait before shifting DRAM to power down mode in the RDWC register.
9. Write DRAM commands to the DCMD register: pre-charge, CBR refresh, start/end self refresh, etc.

### Address Outputs and Bus Width

The following Table summarizes the relationships between address outputs (XA[\*]) and the DRAM address pins (A[\*]).

	AMUX[1:0]=00 Column length = 8 *2		AMUX[1:0]=01 Column length = 9 *2		AMUX[1:0]=10 Column length = 10 *2		DRAM address shifting with bus width *1	
	Row	Column	Row	Column	Row	Column	16 bits	8 bits
XA[23]	0	0	0	0	0	0	A[22]	A[23]
XA[22]	0	0	0	0	0	0	A[21]	A[22]
XA[21]	0	0	0	0	0	0	A[20]	A[21]
XA[20]	0	0	0	0	0	0	A[19]	A[20]
XA[19]	0	0	0	0	0	0	A[18]	A[19]
XA[18]	Ha[26]	Ha[26]*4	0	0	0	0	A[17]	A[18]
XA[17]	Ha[25]	Ha[25]*4	Ha[26]	Ha[26]*4	0	0	A[16]	A[17]
XA[16]	Ha[24]	Ha[24]*4	Ha[25]	Ha[25]*4	Ha[26]	Ha[26]*4	A[15]	A[16]
XA[15]	Ha[23]	Ha[23]*4	Ha[24]	Ha[24]*4	Ha[25]	Ha[25]*4	A[14]	A[15]
XA[14]	Ha[22]	Ha[22]*4	Ha[23]	Ha[23]*4	Ha[24]	Ha[24]*4	A[13]	A[14]
XA[13]	Ha[21]	Ha[21]*4	Ha[22]	Ha[22]*4	Ha[23]	Ha[23]*4	A[12]	A[13]
XA[12]	Ha[20]	Ha[20]	Ha[21]	Ha[21]	Ha[22]	Ha[22]	A[11]	A[12]
XA[11]	Ha[19]	Ha[11]*3	Ha[20]	ha[11]*3	Ha[21]	Ha[11]*3	A[10]*3	A[11]
XA[10]	Ha[18]	Ha[10]	Ha[19]	Ha[10]	Ha[20]	Ha[10]	A[9]	A[10]
XA[9]	Ha[17]	Ha[9]	Ha[18]	Ha[9]	Ha[19]	Ha[9]	A[8]	A[9]
XA[8]	Ha[16]	Ha[8]	Ha[17]	Ha[8]	Ha[18]	Ha[8]	A[7]	A[8]
XA[7]	Ha[15]	Ha[7]	Ha[16]	Ha[7]	Ha[17]	Ha[7]	A[6]	A[7]
XA[6]	Ha[14]	Ha[6]	Ha[15]	Ha[6]	Ha[16]	Ha[6]	A[5]	A[6]
XA[5]	Ha[13]	Ha[5]	Ha[14]	Ha[5]	Ha[15]	Ha[5]	A[4]	A[5]
XA[4]	Ha[12]	Ha[4]	Ha[13]	Ha[4]	Ha[14]	Ha[4]	A[3]	A[4]
XA[3]	Ha[11]	Ha[3]	Ha[12]	Ha[3]	Ha[13]	Ha[3]	A[2]	A[3]
XA[2]	Ha[10]	Ha[2]	Ha[11]	Ha[2]	Ha[12]	Ha[2]	A[1]	A[2]
XA[1]	Ha[9]	Ha[1]	Ha[10]	Ha[1]	Ha[11]	Ha[1]	A[0]	A[1]
XA[0]	Ha[8]	Ha[0]	Ha[9]	Ha[0]	Ha[10]	Ha[0]	N.C.	A[0]

#### Notes

Ha stands for host address, the address used by the program.

1. The BWD RAM bit in the DBWC register specifies the data bus width and thus the mapping of the XA[\*] signals to the DRAM address A[\*] pins regardless of the column length. SDRAM cannot use an 8-bit bus.
2. The AMUX bits in the DRMC register specify the column length, the dividing line (8 to 10 bits) for address multiplexing of the row and column address outputs to the XA pins. DRAM cannot use column lengths of 11 and higher.
3. This bit represents a column address bit during EDO DRAM column output. During SDRAM operation, however, it specifies auto pre-charge, going to "0" during column output to disable auto pre-charge during READ and WRITE commands.
4. For SDRAM ACT, READ, and WRITE commands, the XA[18:12] outputs have the same value, the bank number, during both row and column output.

### Distributed CAS before RAS (CBR) Refresh

The DRAM controller provides a facility for automatically performing distributed CBR refresh operations at regular intervals.

- Specify the distributed refresh periods in the RFSH0 and RFSH1 registers.
- Start and stop operation by changing the RFRSH bit in the DRMC register.

Note that manual refresh operation is available at any time by writing the CBR refresh command to the DCMD register.

### Self Refresh

SDRAM self refresh operation is used, for example, as part of power management.

Start and stop operation by writing the corresponding commands to the DRCMD bits in the DCMD register: start SDRAM self refresh and end SDRAM self refresh. The start command stops the SDRAM clock, fixing the SDCLK output at High level; the stop command restarts output.

Operation is not guaranteed if the program writes any other command to the DCMD register between the start and end commands.

During Self Refresh, changes to SDMD register or other SDRAM settings will have unpredictable results.

### SDRAM Commands

The DRAM controller provides the following SDRAM command outputs.

Command	Abbreviation	XSDCKE		XSDCS_N	XRAS_N	XCAS_N	XWE_N	XDQM	Address		
		n-1	n						A11	A10	A9-0
Mode register set	MRS	H	H	L	L	L	L	H	OPECODE		
CBR (auto) refresh	REFH	H	H	L	L	L	H	H	X	X	X
Start self refresh	SELF	H	L	L	L	L	H	H	X	X	X
End self refresh	SREX	L	H	L	H	H	H	H	X	X	X
		L	H	H	H	H	H	H	X	X	X
Pre-charge all banks	PALL	H	H	L	L	H	L	H	X	H	X
Bank active	ACT	H	H	L	L	H	H	H	BA	RA	RA
Write *2	WRIT	H	H	L	H	L	L	L	BA	L	CA
Read *2	READ	H	H	L	H	L	H	L	BA	L	CA
No operation	NOP	H	H	L	H	H	H	H	X	X	X
Deselect device	DESL	H	H	H	H	H	X	H	X	X	X
Power down *1	–	X	L	H	H	H	X	H	X	X	X
Continue self refresh *1	–	L	L	L	L	L	X	H	X	X	X

H: High level, L: Low level, X: Don't care level High or Low

BA: Bank number, RA: Row address, CA: Column address

#### Notes

1. These represent pseudo-commands defined for purposes of explication.
2. Unnecessary byte lanes are all at High level (masked state).

### SDRAM Power Down Mode

The DRAM controller drives the clock enable signal (CKE) output at Low level to shift SDRAM to power down mode, conserving power, when the number of idle cycles specified in the RDWC register have elapsed and there are no DRAM access requests--that is, no DRAM space access requests (or error responses either)<sup>\*1</sup> no DRAM register space access requests, and no distributed CBR refresh requests.

The DRAM controllers switches back if any of the above conditions are not met.

Switching back carries with it a penalty: an additional memory access overhead of three clock cycles.

#### Signal outputs in power down mode

- SDRAM clock signal (SDCLK) output: Stopped (fixed at Low level)
- Clock enable signal (CKE) output: Disabled (fixed at Low level)

#### Notes

1. After a power on reset, do not shift SDRAM to power down mode until the SDRAM initialization sequence is complete.
2. EDO DRAM does not provide an equivalent to SDRAM power down mode.

### 11.3.5 Access Timing Parameters for DRAM

The following Table lists DRAM device speeds compatible with various operating frequencies. Figuring out the necessary DRAMSPEC bit settings in the DRPC register is left as an exercise for the reader.

Operating Frequency (MHz)	SDRAM					EDO-DRAM (Trac [ns])				
	PC133	PC125	PC100	PC33	PC66	50	60	70	80	100
60.0	○	○	○	○	○	○	○	○	○	○
33.3	○	○	○	○	○	○	○	○	○	○
30.0	○	○	○	○	○	○	○	○	○	○
16.7	○	○	○	○	○	○	○	○	○	○
15.0	○	○	○	○	○	○	○	○	○	○
8.4	○	○	○	○	○	○	○	○	○	○
7.5	○	○	○	○	○	○	○	○	○	○

The DRAMSPEC bits in the DRPC register specify a combination of access timing parameters (tRCD, tRAS, tRP, etc.) as shown in the table under the register description. There is no way to specify these parameters individually.

#### Minimum Operating Frequencies

	Minimum Operating Frequencies (HCLK)	
	ML674001 Series	ML675001 Series
<b>SDRAM:</b>	2.56 MHz	2.56 MHz
<b>EDO-DRAM:</b>	6.4 MHz	6.4 MHz

#### SDRAM Access Timing Parameters and Operating Frequency

SDRAM Type (PCxx)	Access Parameters (ns)				Access Parameters (clock cycles) at 16.7 MHz				Access Parameters (clock cycles) at 33.3 MHz			
	tRCD	tRAS	tRP	tDPL	tRCD	tRAS	tRP	tDPL	tRCD	tRAS	tRP	tDPL
133	15	45	15	10	1	2	1	1	1	2	1	1
133	20	45	20	10	1	2	1	1	1	2	1	1
133	20	45	20	15	1	2	1	1	1	2	1	1
125	20	48	20	8	1	2	1	1	1	2	1	1
125	20	48	20	10	1	2	1	1	1	2	1	1
125	20	50	30	8	1	2	1	1	1	2	1	1
125	24	48	24	10	1	2	1	1	1	2	1	1
100	20	50	20	10	1	2	1	1	1	2	1	1
100	20	50	20	15	1	2	1	1	1	2	1	1
100	20	50	20	20	1	2	1	1	1	2	1	1
100	30	50	30	15	1	2	1	1	1	2	1	1
100	30	60	30	10	1	2	1	1	1	2	1	1
100	30	60	30	15	1	2	1	1	1	2	1	1
83	35	70	45	24	1	2	1	1	2	3	2	1
66	30	60	30	15	1	2	1	1	1	2	1	1
66	30	70	30	15	1	2	1	1	1	3	1	1

Note: The DRAM controller assumes  $tRC = tRAS + tRP$ , so also check  $tRC$  if the SDRAM specifies  $tRC > tRAS + tRP$ .



EDO DRAM Access Timing Parameters and Operating Frequency

tRAC (ns)	Access Parameters (ns)				Access Parameters (clock cycles) at 16.7 MHz				Access Parameters (clock cycles) at 33.3 MHz			
	tRAH tCAS	tRCD	tCAC tOEZ	tRP	tRAH tCAS	tRCD	tCAC tOEZ	tRP	tRAH tCAS	tRCD	tCAC tOEZ	tRP
50	7	37	13	30	1	2	1	1	1	2	1	1
50	8	35	15	30	1	2	1	1	1	2	1	1
50	8	37	13	30	1	2	1	1	1	2	1	1
50	10	36	20	40	1	2	1	1	1	2	1	2
50	15	45	15	40	1	2	1	1	1	2	1	2
60	10	40	20	40	1	2	1	1	1	2	1	2
60	10	45	15	40	1	2	1	1	1	2	1	2
60	20	50	20	50	1	2	1	1	1	2	1	2
70	10	50	20	40	1	2	1	1	1	2	1	2
70	10	50	20	50	1	2	1	1	1	2	1	2
70	13	50	20	50	1	2	1	1	1	2	1	2
70	20	60	20	60	1	2	1	1	1	2	1	2
80	10	60	15	60	1	2	1	1	1	2	1	2
100	25	75	25	70	1	2	1	2	1	3	1	3

11.3.6 Read off time Control

The External Memory Controller automatically inserts, as necessary, read off time to avoid collisions between XD data from successive accesses to external ROM, SRAM, I/O or DRAM. Read off time is the minimum number of clock cycles between chip select signal going inactive until the next chip select signal goes active. The actual number of cycles depends on the type of preceding and successive accesses.

The following Table summarizes the insertion of read off time for all combinations of access to external ROM, SRAM, I/O or DRAM for ML674001 series and ML675001 series. 'O' in the table means that read off time is inserted according to the setting of the ROMAC, RAMAC, IO01AC, IO23ACX or IO23ACY register. Space in the table means that read off time is not inserted for the combination of the accesses.

Current Cycle			Following Cycle											
			ARM				DMA		ARM & DMA					
			ROM		RAM		RAM		IO		SDRAM		EDO-DRAM	
			R	W	R	W	R	W	R	W	R	W	R	W
ARM	ROM	Read		O	O	O	O	O	O	O	O	O	O	O
		Write												
	RAM	Read	O	O		O	O	O	O	O	O	O	O	O
		Write												
	IO	Read	O	O	O	O	O	O		O	O	O	O	O
		Write												
DMA	RAM	Read	O	O	O	O		O	O	O	O	O	O	O
		Write												
	IO	Read	O	O	O	O	O	O		O	O	O	O	O
		Write												
ARM & DMA	SDRAM	Read											-	-
		Write												-
	EDO-DRAM	Read	O	O	O	O	O	O	O	O	-	-	O	O
		Write									-	-		

**Note:** The read off time for ROM, RAM, IO is specified by ROMAC RAMAC IO01AC IO23ACX IO23ACY registers. And that time for EDO-DRAM is specified by tOEZ of DRAMSPEC register. There is no specified Read off time for SDRAM.

## 11.4 Access Timing

### 11.4.1 Accessing External Devices

#### 11.4.1.1 External ROM/RAM Access

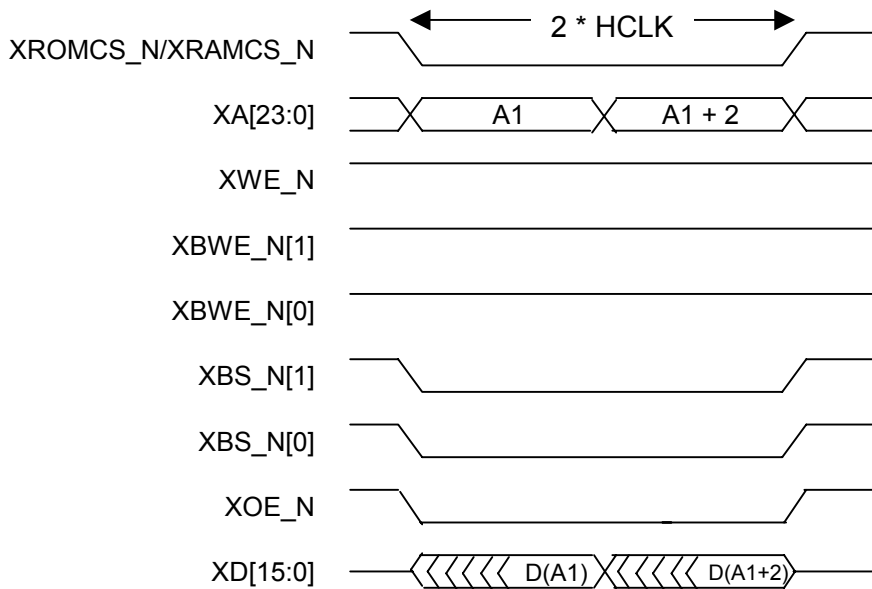


Figure 11.1 External ROM/RAM, Word Read, OE/WE Pulse Width 1 (Minimum Setting)

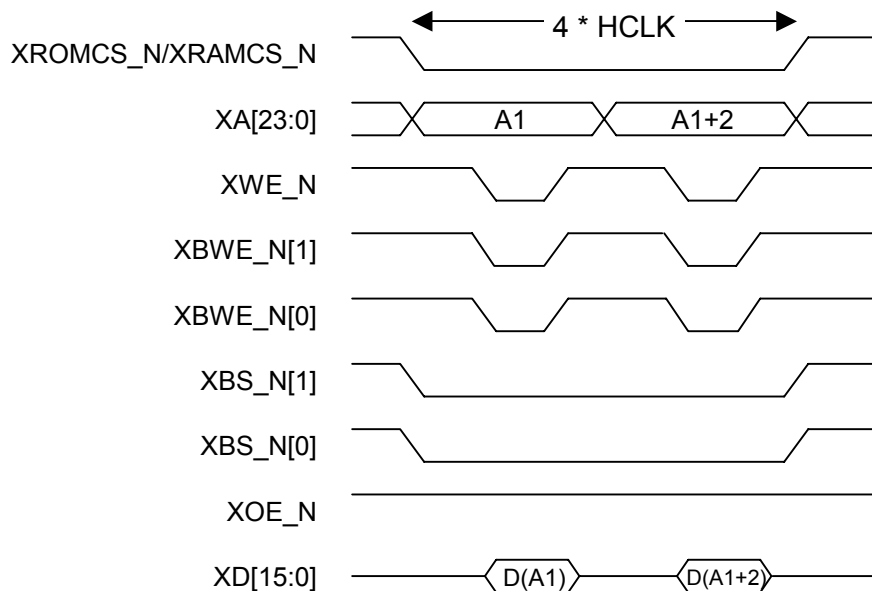
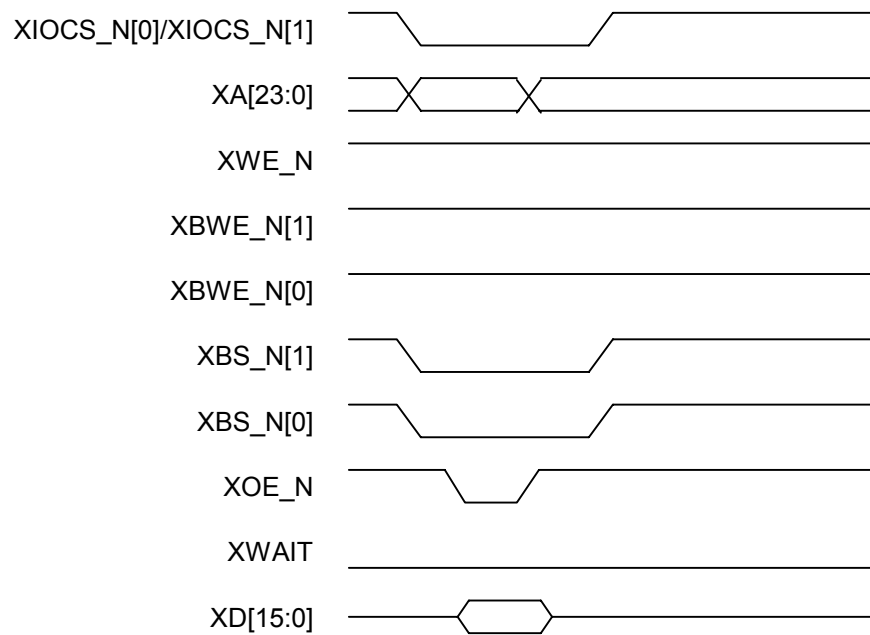
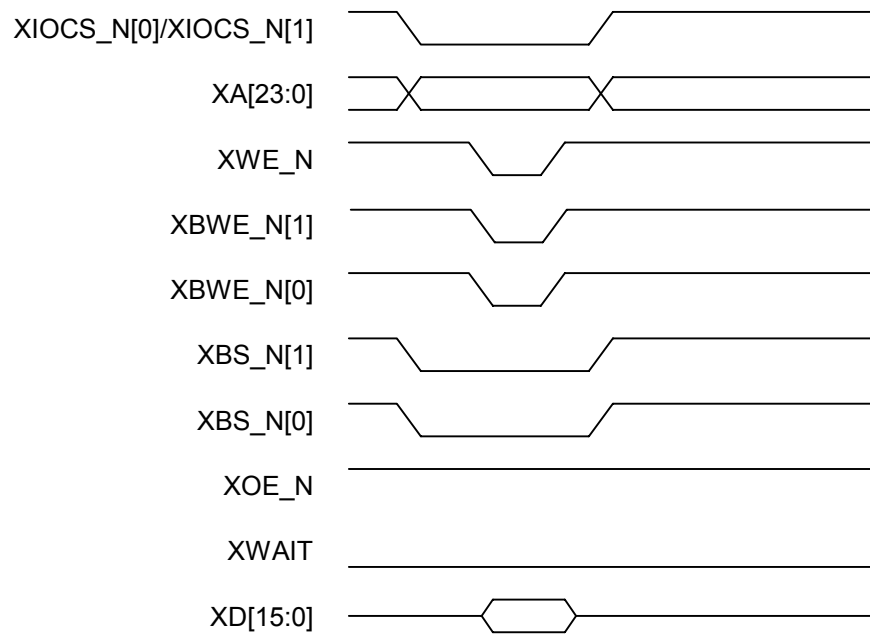


Figure 11.2 External ROM/RAM, Word Write, OE/WE Pulse Width 1 (Minimum Setting)

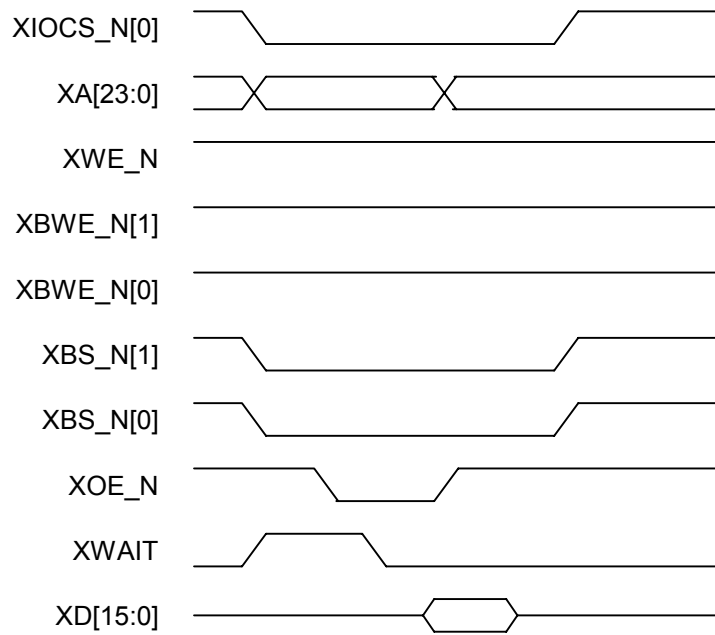
11.4.1.2 External I/O Bank Access



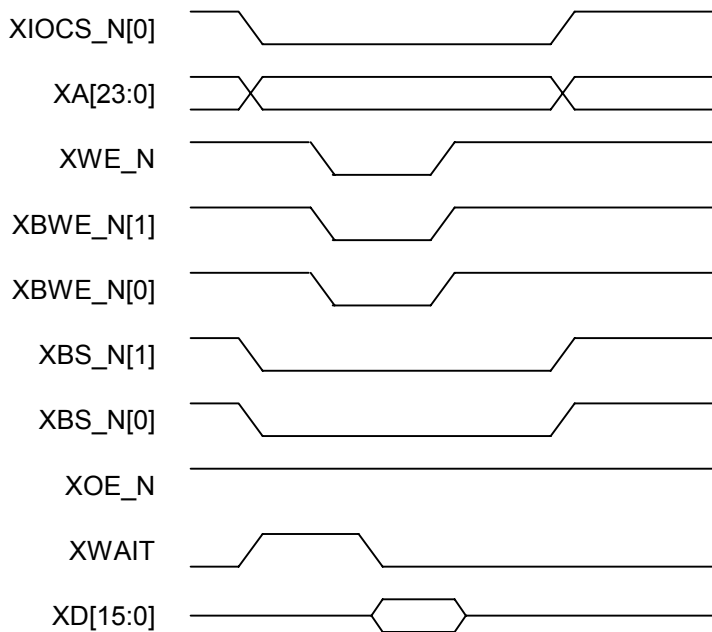
**Figure 11.3 External I/O Bank, 16-Bit External Bus, Half-word Read, OE/WE Pulse Width 1 (Minimum Setting) without XWAIT Memory Wait Cycles**



**Figure 11.4 External I/O Bank, 16-Bit External Bus, Half-word Write, OE/WE Pulse Width 1 (Minimum Setting) without XWAIT Memory Wait Cycles**

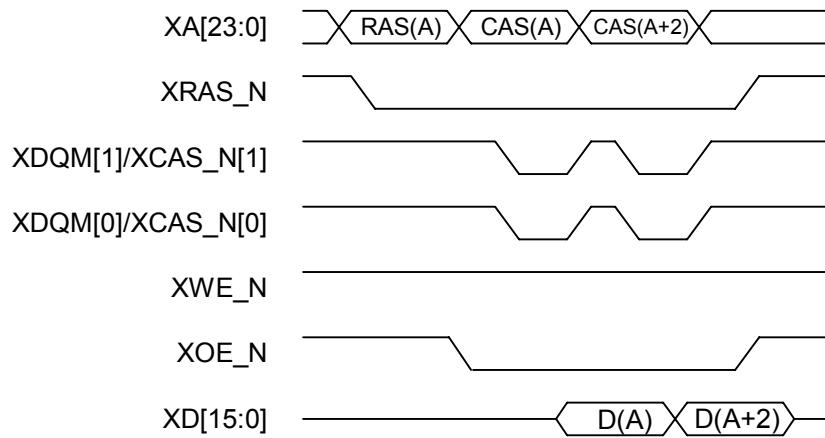


**Figure 11.5 External I/O Bank, 16-Bit External Bus, Half-word Read, OE/WE Pulse Width 1 (Minimum Setting) with XWAIT Memory Wait Cycles**

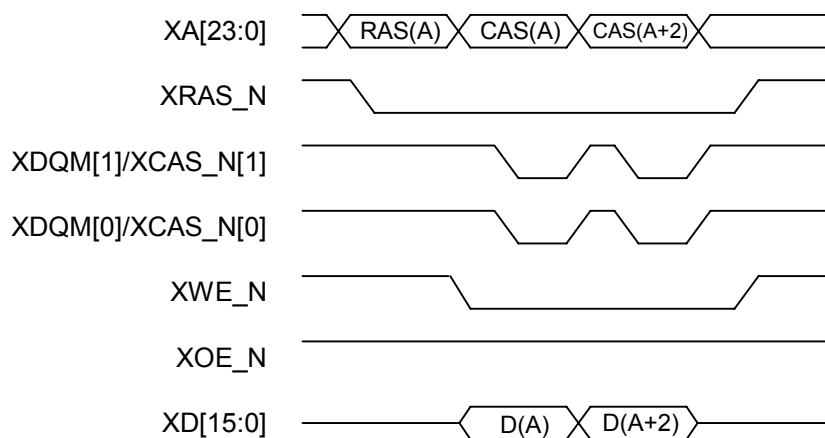


**Figure 11.6 External I/O Bank, 16-Bit External Bus, Half-word Write, OE/WE Pulse Width 1 (Minimum Setting) with XWAIT Memory Wait Cycles**

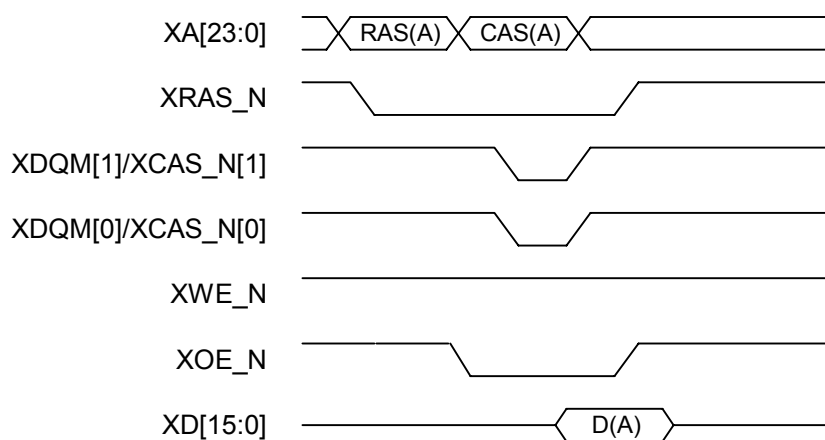
11.4.1.3 EDO DRAM Access



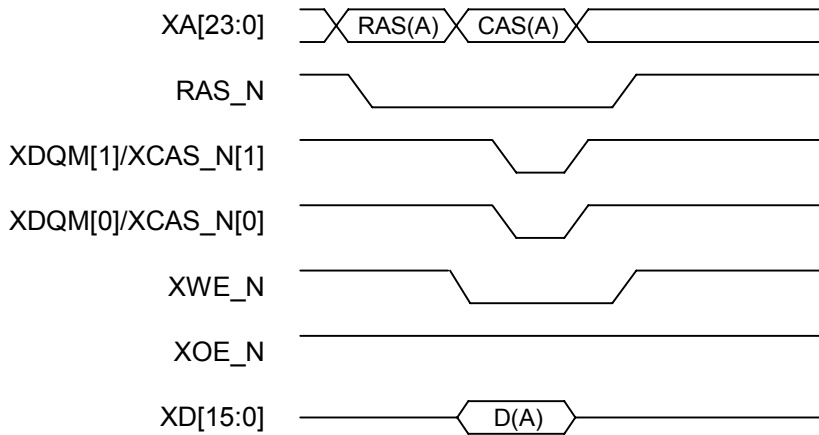
**Figure 11.7 EDO DRAM, Word Read**



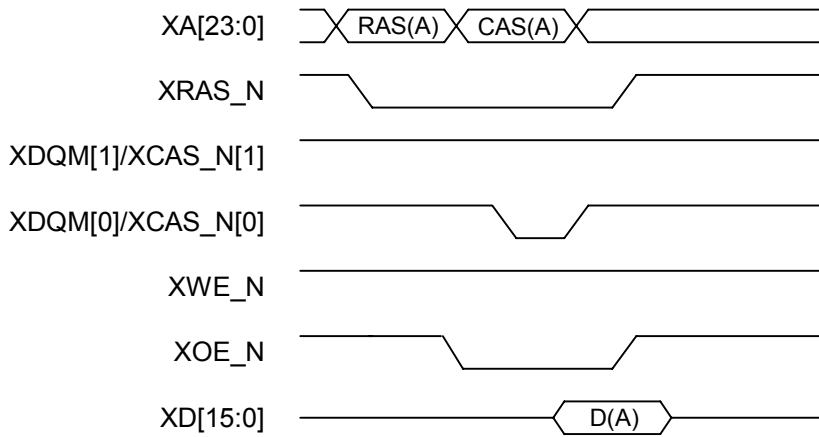
**Figure 11.8 EDO DRAM, Word Write**



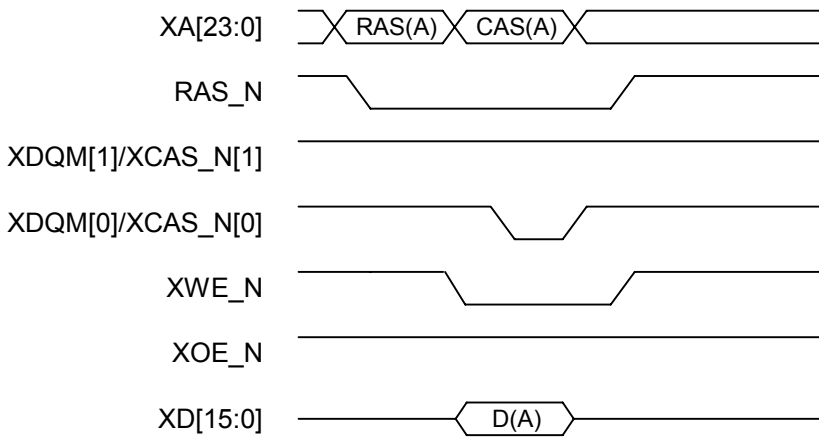
**Figure 11.9 EDO DRAM, Half-word Read**



**Figure 11.10 EDO DRAM, Half-word Write**

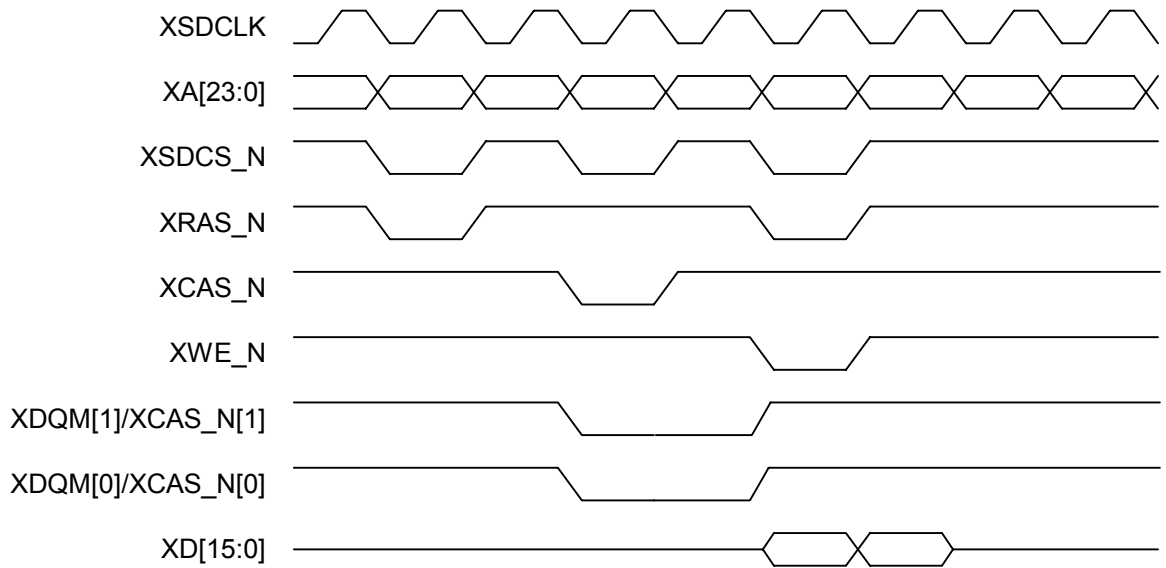


**Figure 11.11 EDO DRAM, Byte (LSB) Read**

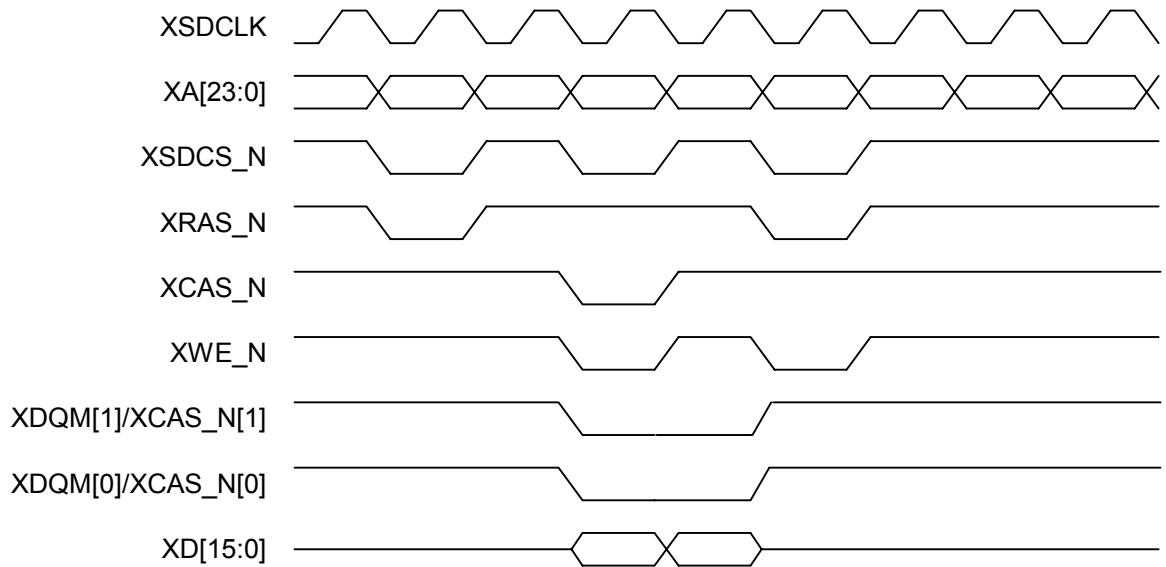


**Figure 11.12 EDO DRAM, Byte (LSB) Write**

11.4.1.4 SDRAM Access

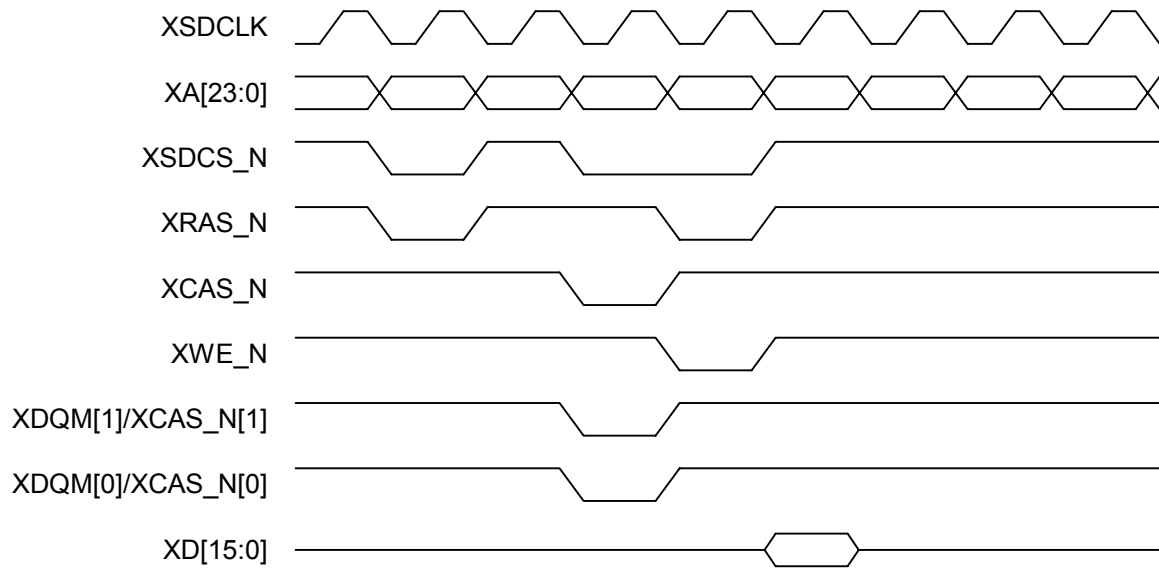


**Figure 11.13 SDRAM, Word Read**

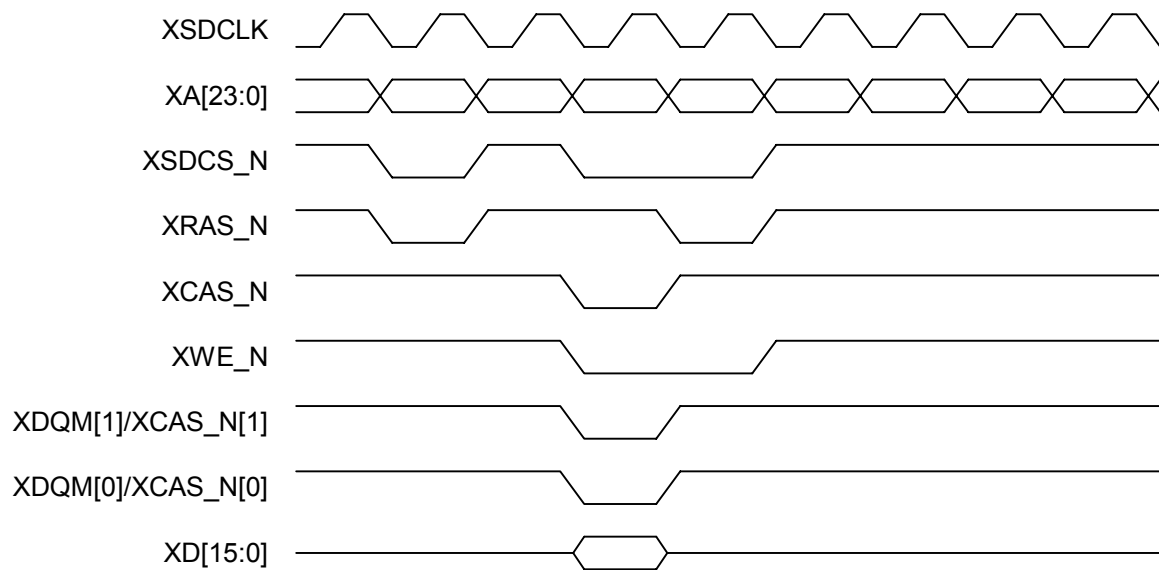


**Figure 11.14 SDRAM, Word Write**





**Figure 11.15 SDRAM, Half-word Read**



**Figure 11.16 SDRAM, Half-word Write**

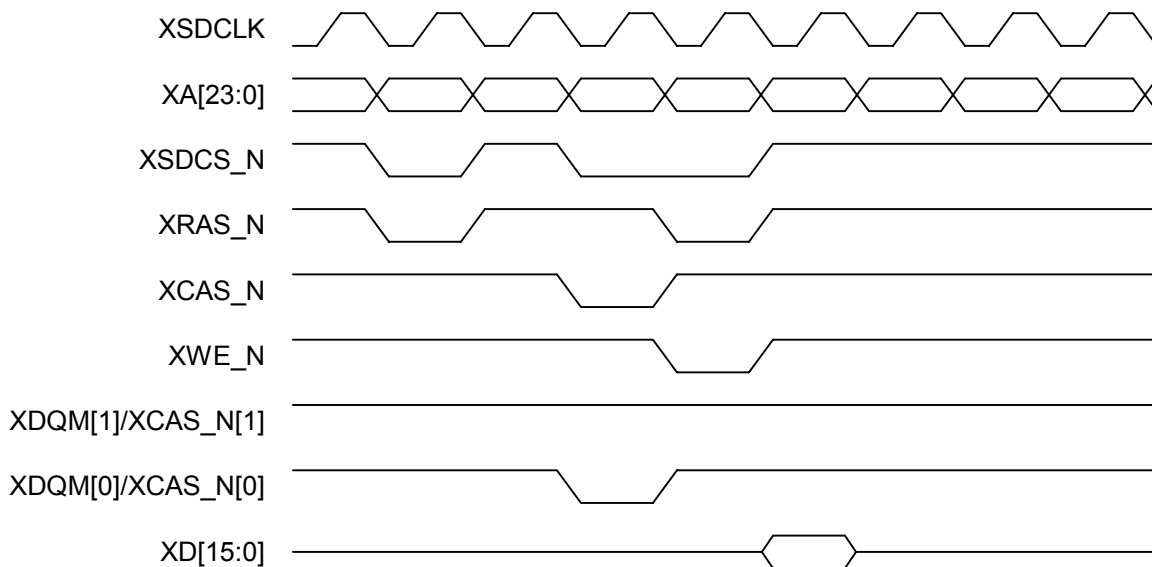


Figure 11.17 SDRAM, Byte (LSB) Read

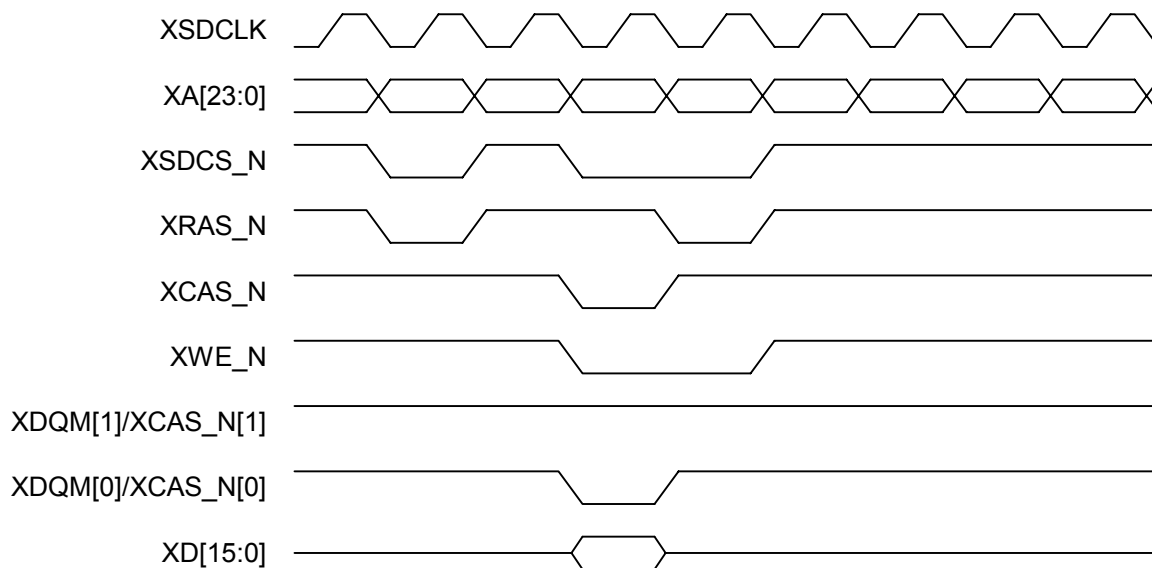


Figure 11.18 SDRAM, Byte (LSB) Write

## 11.5 DRAM Power Management

Using STANDBY or HALT mode requires program manipulation of the DRAM controller. For further details, see Chapter 7 “Power Management.”

## 11.6 Sample External Memory Connections

The following pages give connection examples for each device type.

Devices differ as to signal names and other points, so refer also to the data sheet for the actual intended device.

As the number of devices increases, insert buffer elements and other components to ensure adequate drive power, load capacity, etc.

We recommend pull-up resistors on the XD pins.

Pin Name	I/O	function	ROM	SRAM	IO				SDRAM	EDO-DRAM
					0	1	2	3		
XA[23:19]	O	External address bus	O *3	O *3	O *3	O *3				
XA[18:0]	O	External address bus	O	O	O	O	O	O	O	O
XD[15:0]	I/O	External data bus	O	O	O	O	O	O	O	O
XROMCS_N	O	External ROM chip select	O							
XRAMCS_N	O	External RAM chip select		O						
XIOCS_N[0]	O	I/O 0 chip select			O					
XIOCS_N[1]	O	I/O 1 chip select				O				
XIOCS_N[2]	O	I/O 2 chip select					O			
XIOCS_N[3]	O	I/O 3 chip select						O		
XOE_N	O	Output enable	O	O	O	O				O
XWE_N	O	Write enable	O *1	O *1	O *1	O *1	O *1	O	O	O
XBS_N[1:0]	O	External bus byte select	O *1	O *1	O *1	O *1				
XBWE_N[0]	O	Write enable (LSB)	O *2	O *2	O *2	O *2				
XBWE_N[1]	O	Write enable (MSB)	O *2	O *2	O *2	O *2				
XSDCS_N	O	SDRAM chip select						O		
XSDCLK	O	SDRAM clock						O		
XSDCKE	O	SDRAM clock enable						O		
XCAS_N	O	SDRAM column address strobe						O		
XRAS_N	O	Row address strobe						O	O	
XDQM[1]/ XCAS_N[1]	O	Data I/O mask for SDRAM or MSB column address strobe for EDO DRAM						O	O	
XDQM[0]/ XCAS_N[0]	O	Data I/O mask for SDRAM or LSB column address strobe for EDO DRAM						O	O	
XWAIT	I	Memory wait indicator			O	O				
XWR	O	External bus data transfer direction			O *4	O *4 *5				

### Notes

\*1. For devices using byte select signals for byte access

\*2. For devices using byte write enable signals for byte access

\*3. There is no XA[23:19] output during the boot sequence, so user application systems with devices needing these outputs must provide an external mechanism for fixing these pins at Low level until the secondary (XA) function takes effect.

\*4. Use as necessary.

\*5: Only ML674001 series.

11.6.1 Connecting ROM

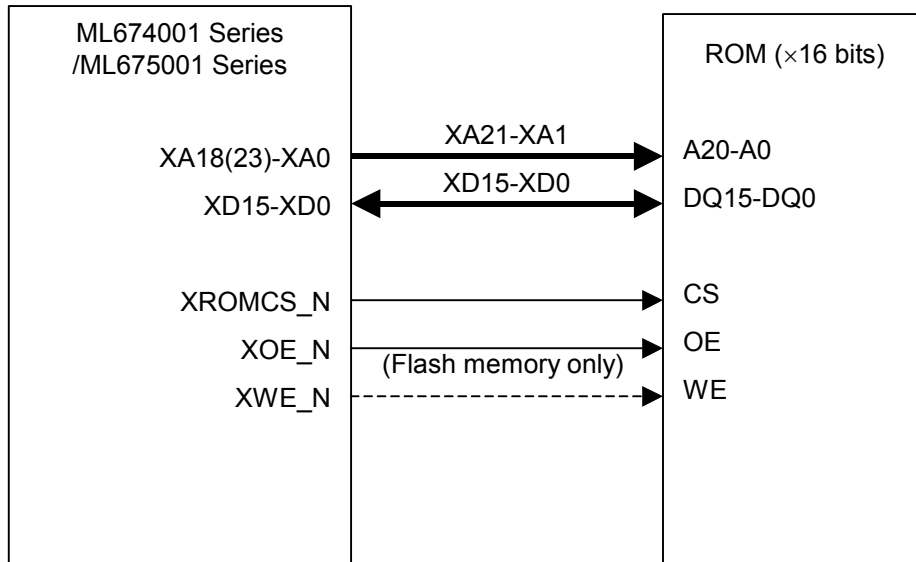


Figure 11.19 Connecting 2M x 16-Bit ROM (4 MB)

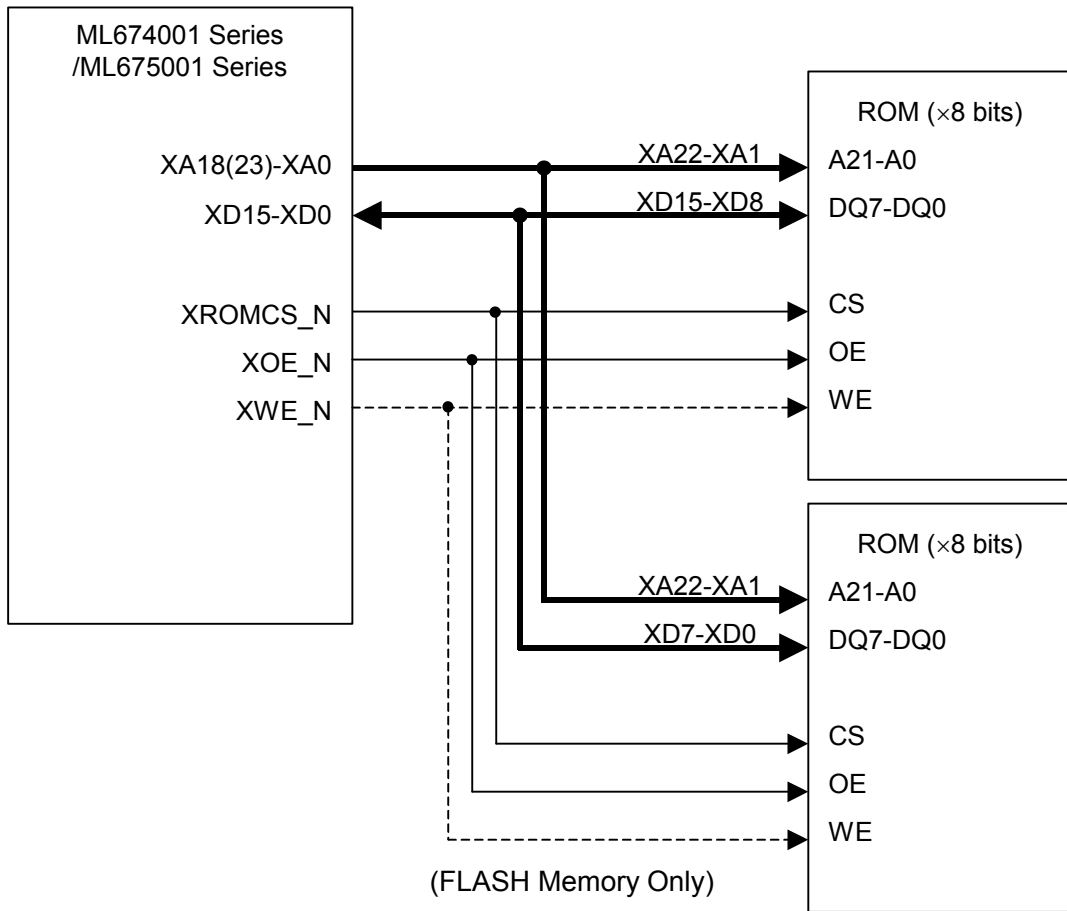
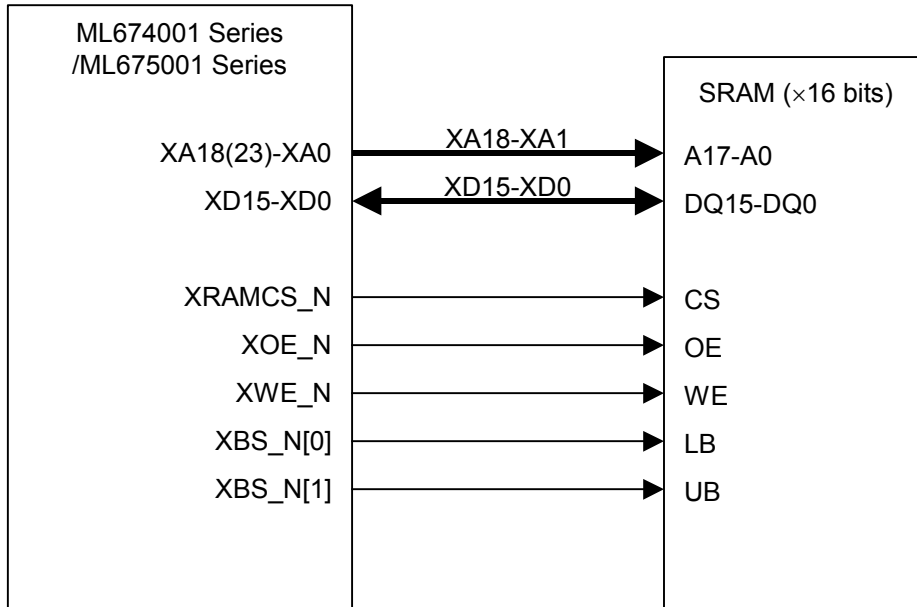


Figure 11.20 Connecting Two 4M x 8-Bit ROMs (4 MB)

### 11.6.2 Connecting SRAM



**Figure 11.21 Connecting 256K x 16-Bit SRAM (512 KB) Using Byte Select Signals**

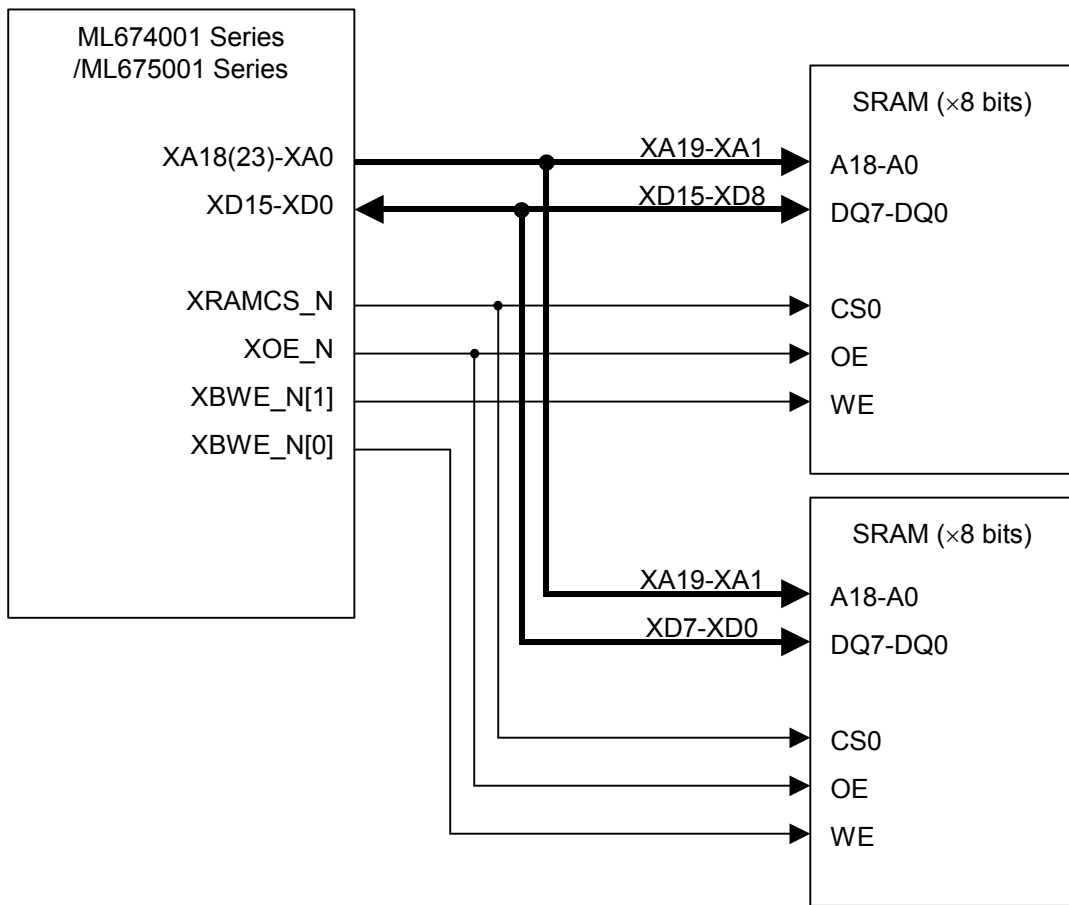
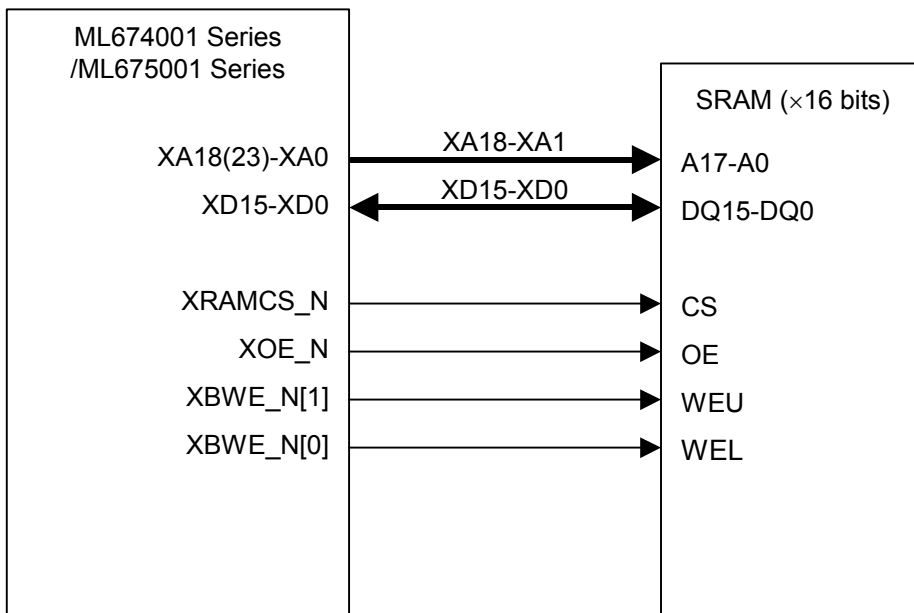


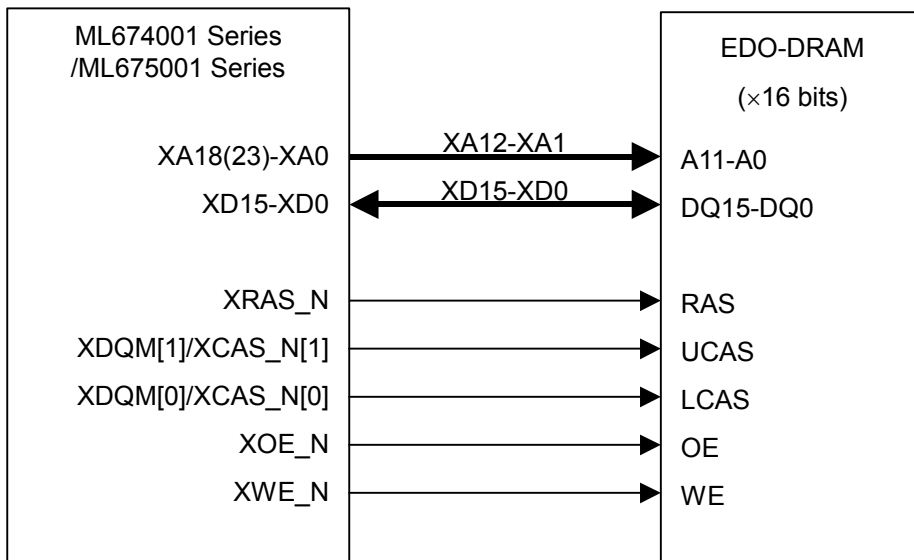
Figure 11.22 Connecting Two 512K x 8-Bit SRAMs (512 KB)





**Figure 11.23 Connecting 256K × 16-Bit SRAM (512 KB) Using Byte Write Enable Signals**

### 11.6.3 Connecting EDO DRAM



**Figure 11.24 Connecting 1M × 16-Bit EDO DRAM (2 MB)**

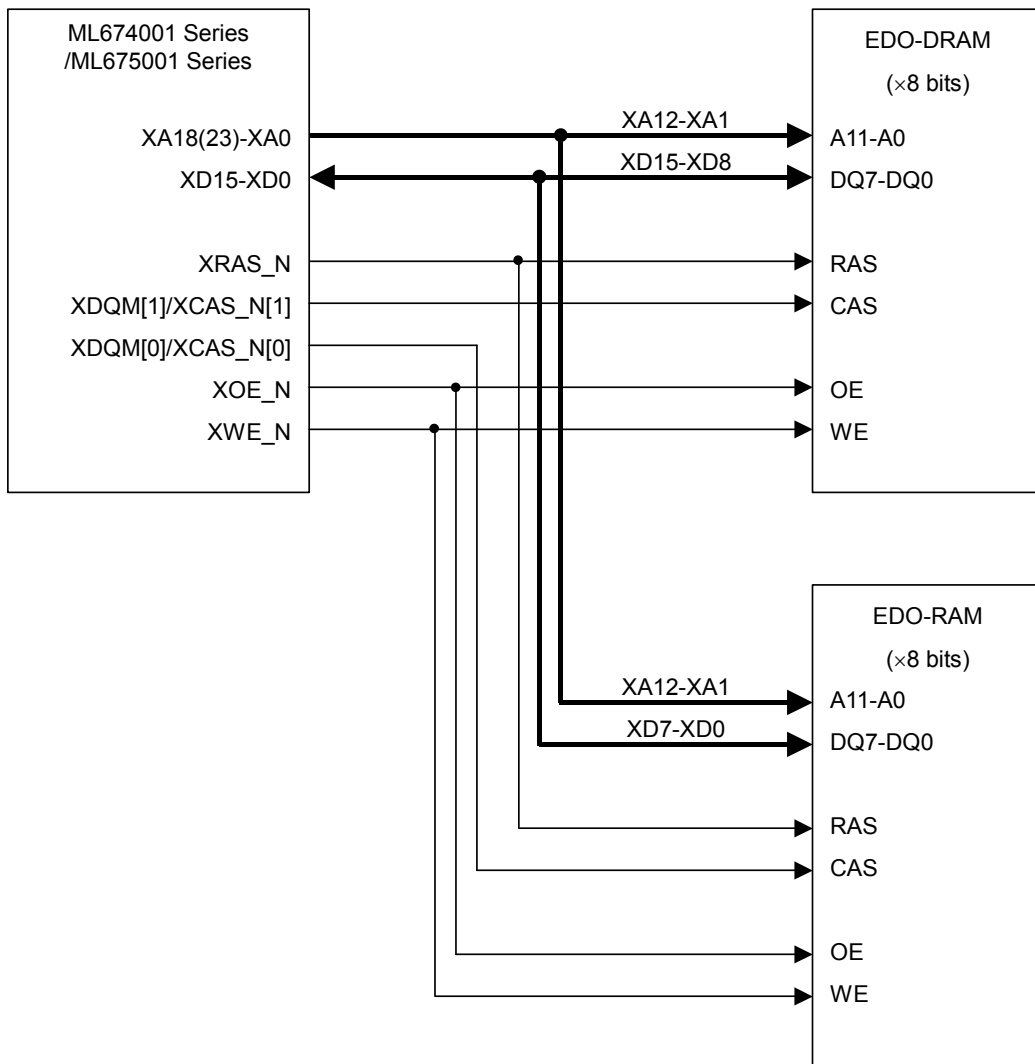
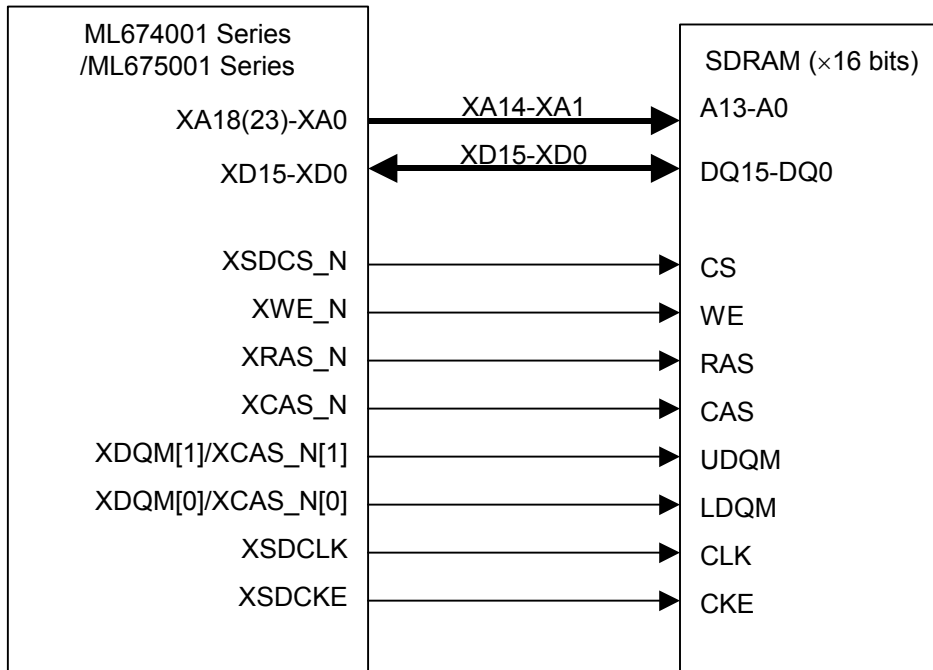


Figure 11.25 Connecting Two 2M x 8-Bit EDO DRAMs (2 MB)

### 11.6.4 Connecting SDRAM



**Figure 11.26 Connecting 1M × 16-Bit SDRAM (8 MB)**

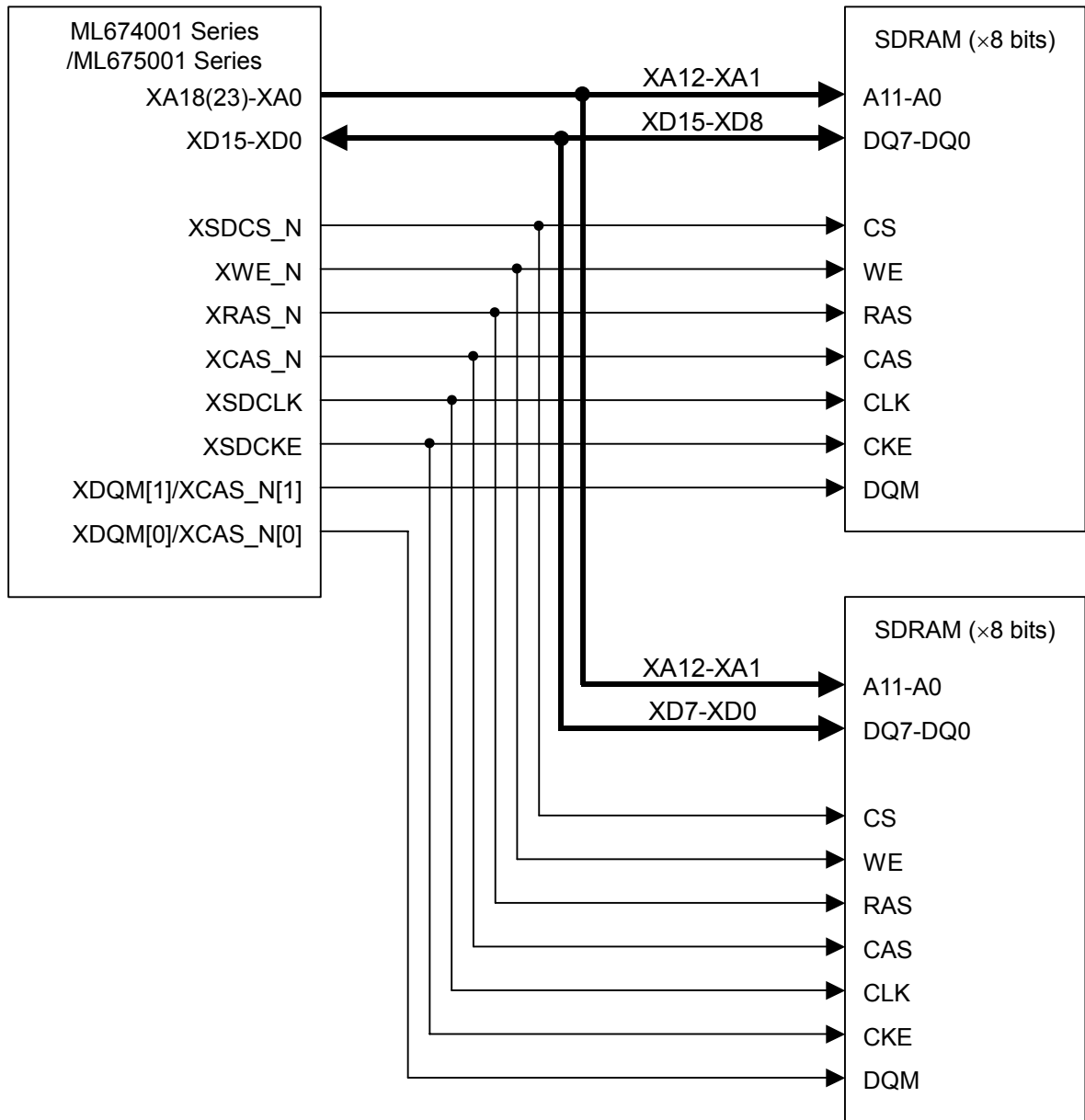


Figure 11.27 Connecting Two x 1M x 8-Bit SDRAMs (4 MB)

**Direct Memory Access Controller  
(DMAC)**

---

## Chapter 12 Direct Memory Access Controller (DMAC)

### 12.1 Overview

This 2-channel direct memory access controller (DMAC) transfers data directly between memory and memory, between an I/O device and memory, and between I/O devices, reducing the CPU load and thus boosting overall LSI operational efficiency.

#### Features

- Number of channels: 2
- Channel Priority
  - Fixed mode: Channel priority never changes (channel 0 > channel 1).
  - Round robin mode: Last channel used has lower priority.
- Maximum number of transfers: 65,536
- Transfer sizes: byte, halfword, and word (8, 16, and 32 bits)
- Dual address access: A read from the transfer source is independent from a write to the transfer destination.
- Bus Access
  - Cycle stealing mode: The DMA controller surrenders bus access after each individual DMA transfer.
  - Burst mode: The DMA controller does not surrender bus access until the specified number of transfers are complete.
- DMA Transfer Requests
  - Software request mode: The DMA controller generates transfer requests until the specified number of transfers are complete.
  - External request mode: The DMA controller generates transfer requests at rising edges in the external request signals (DREQ0 or DREQ1) for the channel.
- Interrupt requests: The DMA controller sends interrupt requests to the CPU when the specified number of transfers are complete or there is an error (nonexistent address specified for transfer destination or transfer source). There are separate interrupt request signals for each channel, and these can be masked by channel.

12.1.1 Components

Figure 12.1 shows DMA controller components; Figure 12.2, DMAC peripheral components.

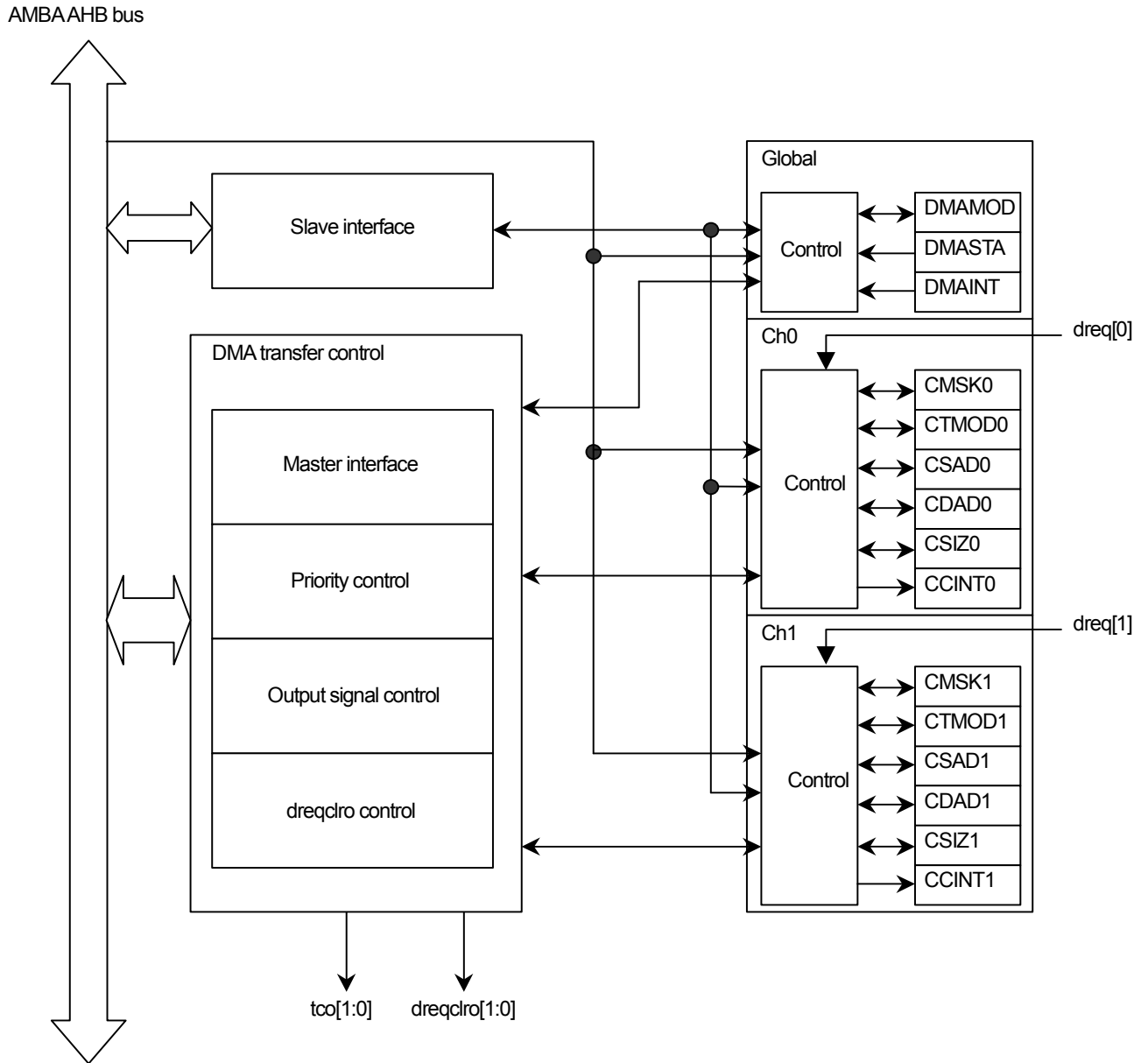
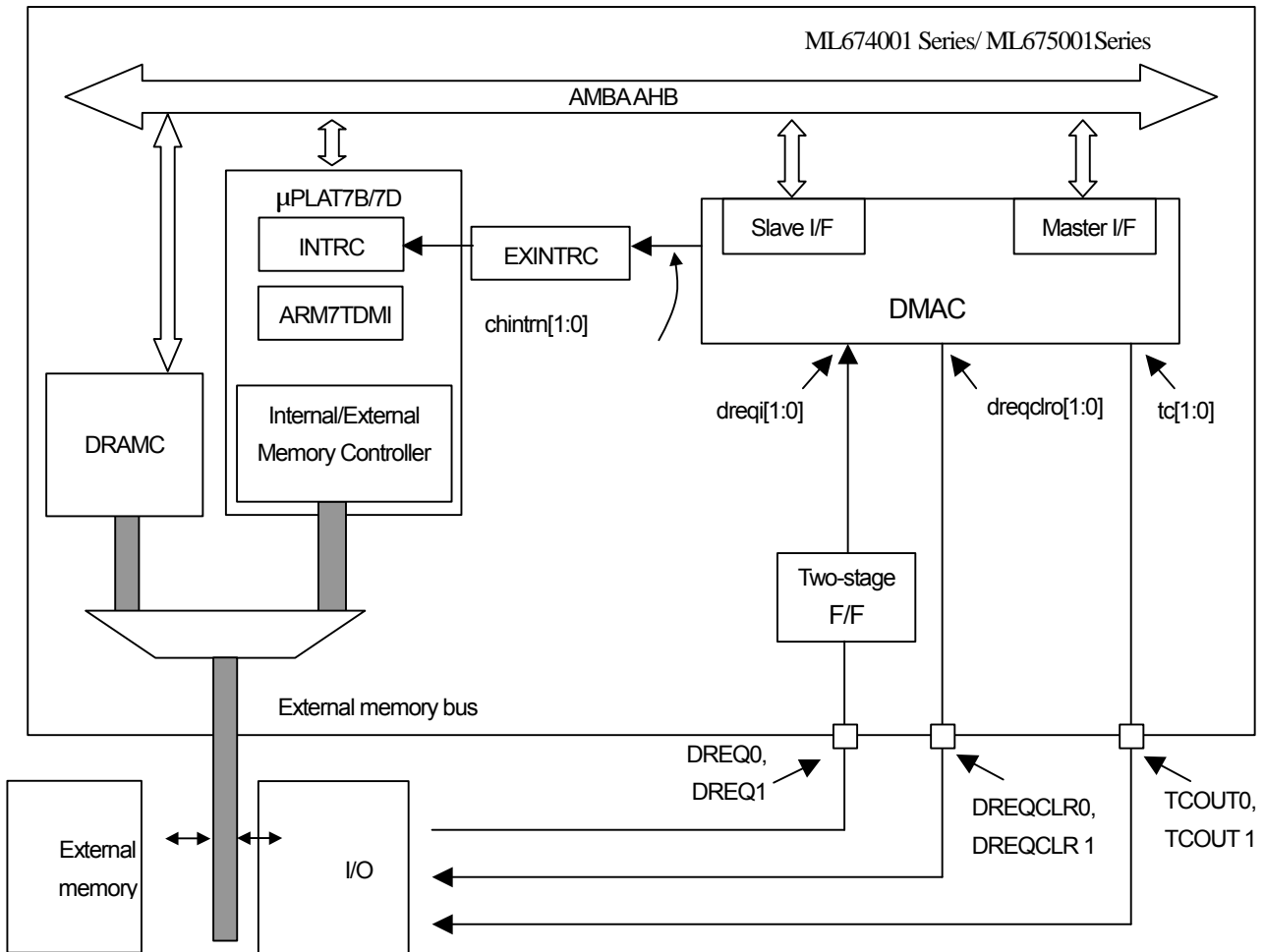


Figure 12.1 DMA Controller Block Diagram



**Figure 12.2 DMA Controller Peripheral Block Diagram**

#### AHB Interface

The DMA controller features a 32-bit connection to the AMBA AHB bus.

The DMA controller provides both slave and master interfaces.

- Slave interface: The program uses this to access DMA controller registers (both global and channel).
- Master interface: The DMA controller uses this for DMA transfers.



### 12.1.2 Pin List

Pin Name	I/O	Description
DREQ0	I	DMA request signal (channel 0)
DREQCLR0	O	DREQ clear request signal (channel 0)
DREQ1	I	DMA request signal (channel 1)
DREQCLR1	O	DREQ clear request signal (channel 1)
TCOUT0	O	DMA transfer complete signal (channel 0)
TCOUT1	O	DMA transfer complete signal (channel 1)

### 12.1.3 Register List

CH	Address [H]	Name	Abbreviation	R/W	Size	Initial Value [H]
Global	0x7BE00000	DMA mode register	DMAMOD	R/W	32	0x00000000
	0x7BE00004	DMA status register	DMASTA	R	32	0x00000000
	0x7BE00008	DMA transfer complete status register	DMAINT	R	32	0x00000000
Channel 0	0x7BE00100	DMA channel mask register	DMACMSK0	R/W	32	0x00000001
	0x7BE00104	DMA transfer mode register	DMACTMOD0	R/W	32	0x00000040
	0x7BE00108	DMA transfer source address register	DMACSDAD0	R/W	32	0x00000000
	0x7BE0010C	DMA transfer destination address register	DMACDAD0	R/W	32	0x00000000
	0x7BE00110	DMA transfer count register	DMACSIZE0	R/W	32	0x00000000
	0x7BE00114	DMA transfer complete status clear register	DMACCINT0	W	32	—
Channel 1	0x7BE00200	DMA channel mask register	DMACMSK1	R/W	32	0x00000001
	0x7BE00204	DMA transfer mode register	DMACTMOD1	R/W	32	0x00000040
	0x7BE00208	DMA transfer source address register	DMACSDAD1	R/W	32	0x00000000
	0x7BE0020C	DMA transfer destination address register	DMACDAD1	R/W	32	0x00000000
	0x7BE00210	DMA transfer count register	DMACSIZE1	R/W	32	0x00000000
	0x7BE00214	DMA transfer complete status clear register	DMACCINT1	W	32	—

**Note:** These registers require word (32-bit) access. Operation is not guaranteed for byte (8-bit) or halfword (16-bit) access.

## 12.2 Register Descriptions

### 12.2.1 DMA Mode Register (DMAMOD)

This register is for specifying DMA channel priority.  
The program has read/write access to this register.  
The contents after a reset are 0x00000000.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAMOD	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	PRI
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0x7BE00000  
Access: R/W  
Access size: 32 bits

**Note**

– \*: These bits are reserved for future expansion. They return “0” for reads. Writes to them are ignored.

**Bit Descriptions**

- **PRI** (bit 0):  
This bit specifies DMA channel priority.

PRI	Description
0	Fixed (channel 0 > channel 1)
1	Round robin (Last channel used has lower priority)

### 12.2.2 DMA Status Register (DMASTA)

This register gives the transfer status for each DMA channel. A “1” in a bit indicates that the transfer count register (DMACSI<sub>Z</sub>) for the channel contains a nonzero value. The bit returns to “0” when the specified number of transfers are complete (normal termination) or an error condition forces an abnormal termination.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMASTA	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	STA1	STA0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0x7BE00004  
 Access: R  
 Access size: 32 bits

**Note**

—\*: These bits are reserved for future expansion. They return “0” for reads.

**Bit Descriptions**

- **STA0** (bit 0):  
 This bit gives the transfer status for DMA channel 0.

STA0	Description
0	Idle (no data to transfer)
1	Busy (transferring data)

- **STA1** (bit 1):  
 This bit gives the transfer status for DMA channel 1.

STA1	Description
0	Idle (no data to transfer)
1	Busy (transferring data)

### 12.2.3 DMA Transfer Complete Status Register (DMAINT)

This register gives the source for an interrupt request indicating the end of a DMA transfer: the channel number, the termination reason (normal or error), and the cycle (read from transfer source or write to transfer destination). Writing anything to an interrupt clear register (DMACCINT0 or DMACCINT1) resets the IREQ, ISTA, and ISTEP bits for the corresponding DMA channel to "0."

The program has only read access to this register.

The contents after a reset are 0x00000000.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAINT	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	ISTP1	ISTP0
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	ISTA1	ISTA0	-*	-*	-*	-*	-*	-*	IREQ1	IREQ0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0x7BE00008

Access: R

Access size: 32 bits

#### Notes

- \*: These bits are reserved for future expansion. They return "0" for reads. These status bits remain "1" until the program explicitly resets them to "0."

#### Bit Descriptions

- IREQ0** (bit 0):  
This bit gives the transfer complete status for DMA channel 0.

IREQ0	Description
0	No request. DMA transfer is not complete or has not started
1	Request pending. DMA transfer is complete

- IREQ1** (bit 1):  
This bit gives the transfer complete status for DMA channel 1.

IREQ1	Description
0	No request. DMA transfer is not complete or has not started
1	Request pending. DMA transfer is complete

- ISTA0** (bit 8):  
This bit gives the termination reason for DMA channel 0.

ISTA0	Description
0	Normal termination
1	Error

- **ISTA1** (bit 9):  
This bit gives the termination reason for DMA channel 1.

<b>ISTA1</b>	<b>Description</b>
0	Normal termination
1	Error

- **ISTP0** (bit 16):  
This bit gives the cycle in which the DMA channel 0 error occurred. It is only valid when ISTA0 = 1.

<b>ISTP0</b>	<b>Description</b>
0	Read from transfer source
1	Write to transfer destination

- **ISTP1** (bit 17):  
This bit gives the cycle in which the DMA channel 1 error occurred. It is only valid when ISTA1 = 1.

<b>ISTP1</b>	<b>Description</b>
0	Read from transfer source
1	Write to transfer destination

### 12.2.4 DMA Channel Mask Registers (DMACMSK0 and DMACMSK1)

These registers control masking, which blocks transfers on the corresponding channel.

The program has read/write access to these registers.

The contents after a reset are 0x00000001, masking all channels.

		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMACMSK0 to 1	-	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
After a reset		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	MSK
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Address: 0x7BE00100 (CH0), 0x7BE00200 (CH1)

Access: R/W

Access size: 32 bits

**Note**

– \*: These bits are reserved for future expansion. They return “0” for reads.

**Bit Descriptions**

- **MSK** (bit 0):  
This bit specifies the channel mask.

MSK	Description
0	Remove mask to enable or restart DMA channel operation
1	Mask (stop) DMA channel operation

### 12.2.5 DMA Transfer Mode Registers (DMACTMOD0 and DMACTMOD1)

These registers specify the DMA transfer mode for the corresponding DMA channel: request source, transfer size, device types, and whether there is an interrupt when the specified number of transfers are complete.

The program has read/write access to this register.

The contents after a reset are 0x00000040.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMACTMOD0 to 1	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	IMK	BRQ	DDP	SDP	TSIZ	ARQ	
	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Address: 0x7BE00104 (CH0), 0x7BE00204 (CH1)

Access: R/W

Access size: 32 bits

#### Notes

- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored. Certain restrictions apply. For further details, see Section 8.3.6 "Important Usage Notes."

#### Bit Descriptions

- ARQ** (bit 0):  
This bit specifies the transfer request source.

ARQ	Description
0	External input (DREQ)
1	Software request mode

- TSIZ** (bits 2 and 1):  
This field specifies the transfer size.

TSIZ		Description
2	1	
0	0	Byte (8 bits)
0	1	Halfword (16 bits)
1	0	Word (32 bits)
1	1	Setting not allowed

- **SDP** (bit 3):  
This bit specifies the device type for the transfer source.

<b>SDP</b>	<b>Description</b>
0	Fixed address device. The DMA controller always reads from the same address.
1	Incremental address device. The DMA controller increments the address by the transfer size in bytes after each successful read from the transfer source.

- **DDP** (bit 4):  
This bit specifies the device type for the transfer destination.

<b>DDP</b>	<b>Description</b>
0	Fixed address device. The DMA controller always writes to the same address.
1	Incremental address device. The DMA controller increments the address by the transfer size in bytes after each successful write to the transfer destination.

- **BRQ** (bit 5):  
This bit specifies the bus request mode.

<b>BRQ</b>	<b>Description</b>
0	Burst mode. The DMA controller does not surrender bus access until the specified number of transfers are complete.
1	Cycle stealing mode. The DMA controller surrenders bus access after each individual DMA transfer.

- **IMK** (bit 6):  
This bit specifies the interrupt mask.

<b>IMK</b>	<b>Description</b>
0	Remove mask to enable interrupt requests
1	Mask (stop) interrupt requests



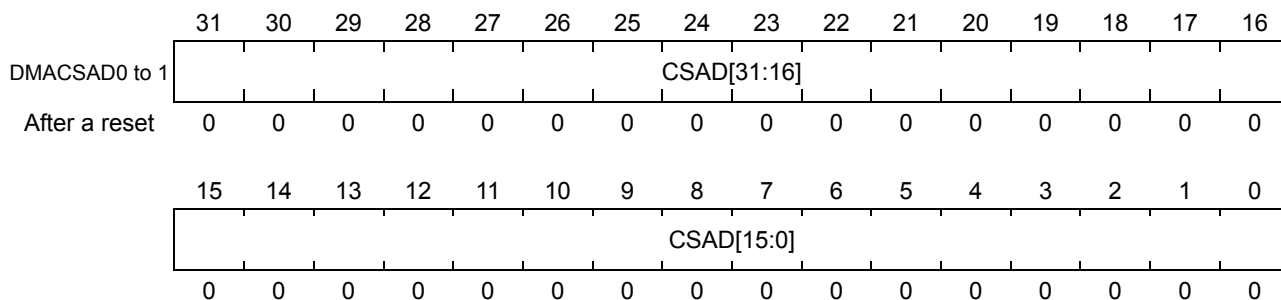
### 12.2.6 DMA Transfer Source Address Registers (DMACCSAD0 and DMACCSAD1)

These registers specify the transfer source address for the corresponding DMA channel. The DMA controller starts reading data for the DMA transfer from the address in this register.

The DMA controller increments the address by the transfer size in bytes after each successful read if the transfer source is an incremental address device. For a fixed address device, however, the address does not change.

The program has read/write access to these registers.

The contents after a reset are 0x00000000.



Address: 0x7BE00108 (CH0), 0x7BE00208 (CH1)

Access: R/W

Access size: 32 bits

**Note**

– \*: The DMA controller increments the source address (DMACCSAD0 or DMACCSAD1) by the transfer size in bytes if the corresponding device type so specifies. The hardware enforces alignment by internally ignoring the lowest address bits as appropriate for the transfer size, but reading an address register returns those bits exactly as written.

**Example:** Writing address of 0x10000011

Word (32-bit) transfer: The DMA controller internally forces the lowest 2 bits to “00” for an effective address of 0x10000000. Reading the register, however, returns 0x10000011.

Halfword (16-bit) transfer: The DMA controller internally forces the lowest bit to “0” for an effective address of 0x10000010. Reading the register, however, returns 0x10000011.

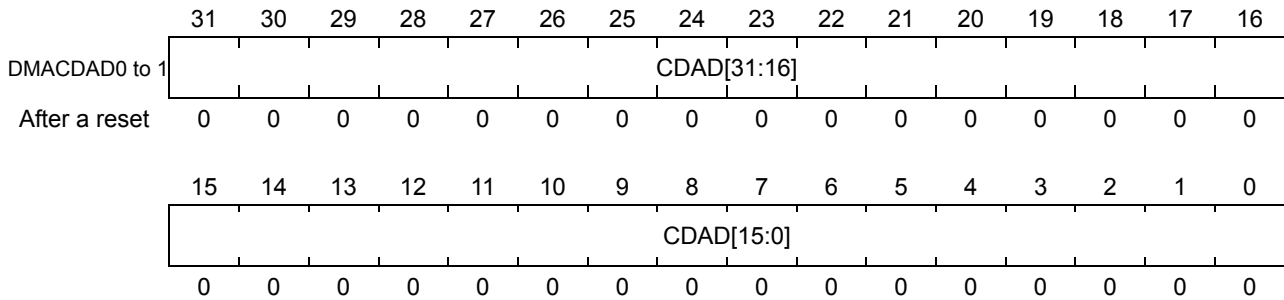
### 12.2.7 DMA Transfer Destination Address Registers (DMACDAD0 and DMACDAD1)

These registers specify the transfer destination address for the corresponding DMA channel. The DMA controller starts writing data for the DMA transfer to the address in this register.

The DMA controller increments the address by the transfer size in bytes after each successful write if the transfer destination is an incremental address device. For a fixed address device, however, the address does not change.

The program has read/write access to these registers.

The contents after a reset are 0x00000000.



Address: 0x7BE0010C (CH0), 0x7BE0020C (CH1)

Access: R/W

Access size: 32 bits

#### Notes

- \*: The DMA controller increments the destination address (DMACDAD0 or DMACDAD1) by the transfer size in bytes if the corresponding device type so specifies. The hardware enforces alignment by internally ignoring the lowest address bits as appropriate for the transfer size, but reading an address register returns those bits exactly as written.

**Example:** Writing address of 0x10000011

Word (32-bit) transfer: The DMA controller internally forces the lowest 2 bits to "00" for an effective address of 0x10000000. Reading the register, however, returns 0x10000011.

Halfword (16-bit) transfer: The DMA controller internally forces the lowest bit to "0" for an effective address of 0x10000010. Reading the register, however, returns 0x10000011.

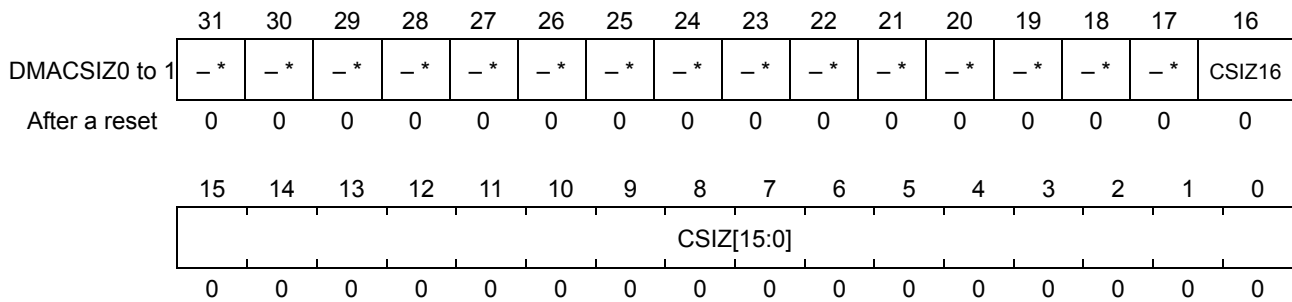
### 12.2.8 DMA Transfer Count Registers (DMACSIZE0 and DMACSIZE1)

These registers specify the transfer count for the corresponding DMA channel. The DMA controller decrements this count after each successful read from the transfer source.

Note that this count represents the number of transfers, not the total size of the transfer. The maximum possible setting for this register is "0x00010000" (65,536).

The program has read/write access to this register.

The contents after a reset are 0x00000000.



Address: 0x7BE00110 (CH0), 0x7BE00210 (CH1)

Access: R/W

Access size: 32 bits

**Notes**

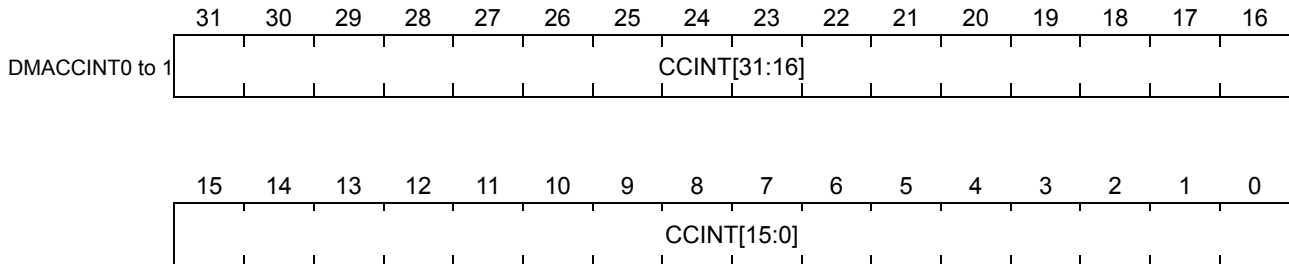
- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

The maximum possible setting for this register is "0x00010000" (65,536). Operation is not guaranteed for higher values.

### 12.2.9 DMA Transfer Complete Status Clear Registers (DMACCINT0 and DMACCINT1)

These registers are for clearing the DMA transfer complete status for the corresponding DMA channel. Writing a 32-bit value to one resets the following bits in the corresponding DMAINT0 or DMAINT1 register: DMA transfer complete (IREQ0 or IREQ1), termination reason (ISTA0 or ISTA1), and abnormal termination cycle (ISTP0 or ISTP1).

The program has only write access to these registers.



Address: 0x7BE00114 (CH0), 0x7BE00214 (CH1)

Access: W

Access size: 32 bits

## 12.3 Operational Description

A transfer request starts DMA transfers. The DMA controller automatically stops when the specified number of transfers are complete. The DMA controller utilizes Dual Address Mode transfer. This type of transfer involves two memory or I/O cycles. The data being transferred is first read from a source address and subsequently written to a destination address in the next cycle. The device will initially signal a request to the DMA controller, e.g. by asserting the DREQ signal. In response, the DMA controller will grant the device or memory access to the bus by causing the OUTPUT\_ENABLE or WRITE\_ENABLE and CS strobes of the device or memory to be asserted thus giving the device or memory access to the bus.

### 12.3.1 DMA Transfer Modes

BRQ, bit 5 in the DMA transfer mode register (DMACTMOD0 or DMACTMOD1), offers a choice of two DMA transfer modes for the corresponding DMA channel.

1. Cycle stealing mode: The DMA controller surrenders bus access after each individual DMA transfer and waits for another transfer request before acquiring bus access for the next transfer. This start and stop process repeats until the specified number of transfers are complete or there is an error.
2. Burst mode: The DMA controller does not surrender bus access until the specified number of transfers are complete or there is an error.

### 12.3.2 DMA Request Sources

ARQ, bit 0 in the DMA transfer mode register (DMACTMOD0 or DMACTMOD1), offers a choice of two sources for DMA transfer requests: external and internal.

1. External input (DREQ):  
The trigger here is a rising edge in the input signal DREQ. The external source must then negate (falling edge) DREQ for each individual transfer in cycle stealing mode. Burst mode ignores this input until the specified number of transfers are complete.

The output signal DREQCLR indicates when the DMA controller is ready for such DREQ edges. The external source must assert DREQ when DREQCLR is at Low level and negate DREQ when DREQCLR is at High level.

Note that the DREQCLR timing differs between cycle stealing and burst modes. For cycle stealing mode, DREQCLR goes to High level after each individual transfer (byte, halfword, or word), so the external source must wait for TCOU, the final transfer start signal, to also go to High level before starting any cleanup operations. For burst mode, DREQCLR and TCOU go to High level simultaneously after the specified number of transfers are complete.

Negating DREQ does not cancel a DMA transfer already in progress. The DMA controller retains bus access as described above. DREQCLR and TCOU go to High level regardless of whether DREQ is already negated. Ensure that the external source wait for DREQCLR to go to High level before negating DREQ.

2. Software request mode:  
For memory-to-memory transfers and transfers between memory and I/O modules that cannot generate DREQ transfer requests, the program sets a bit to have the DMA controller generate internal requests until the specified number of transfers are complete. As long as this bit remains set, the DMA controller automatically performs DMA transfers each time that it is started.

### 12.3.3 Starting a DMA Transfer

The following is the procedure for starting a DMA transfer.

1. Set the IMK bit in the DMA transfer mode register (DMACTMOD0 or DMACTMOD1) for the channel to "1" to mask channel operation. (This step is not necessary if the channel is currently masked.)
2. Write to the DMA interrupt clear register (DMACCINT0 or DMACCINT1) for the channel to clear the status bits ready for a new transfer. (This step is not necessary if there is no interrupt request pending from the last DMA transfer.)
3. Configure the DMA transfer mode register (DMACTMOD0 or DMACTMOD1) for the channel.
4. Load the DMA transfer source address register (DMACSAD0 or DMACSAD1) for the channel.
5. Load the DMA transfer destination address register (DMACDAD0 or DMACDAD1) for the channel.
6. Load the DMA transfer count register (DMACSIZ0 or DMACSIZ1) for the channel.
7. Set the IMK bit in the DMA transfer mode register (DMACTMOD0 or DMACTMOD1) for the channel to "0" to release the mask.

**Note:** DMA transfers do not start, however, if the DMA transfer complete status is not clear (step 2).

### 12.3.4 Ending a DMA Transfer

DMA transfers end in one of three ways.

1. Normal termination:  
The DMA transfer automatically ends when the number of transfers specified in the DMA transfer count register (DMACSIZ0 or DMACSIZ1) for the DMA channel are complete. The DMA controller sets IREQ0 or IREQ1 in the DMA transfer complete interrupt status register (DMAINT) to "1," but does not change the ISTA0 and ISTA1 bits. If the IMK bit in the DMA transfer mode register (DMACTMOD0 or DMACTMOD1) is "0," the DMA controller generates an interrupt request.
2. Abnormal termination:  
An error response during the cycle reading from the transfer source or the one writing to the transfer destination immediately terminates the DMA transfer. The DMA controller sets IREQ0 or IREQ1 in the DMA transfer complete interrupt status register (DMAINT) to "1," sets ISTA0 or ISTA1 to "1," and indicates which cycle triggered the error in ISTP0 or ISTP1. If the IMK bit in the DMA transfer mode register (DMACTMOD0 or DMACTMOD1) is "0," the DMA controller generates an interrupt request.
3. Forced termination:  
Setting the IMK bit in the DMA transfer mode register (DMACTMOD0 or DMACTMOD1) to "1" during a DMA transfer suspends operation after the write cycle. There is no interrupt. The program can restart operation by releasing the mask.

The program uses an interrupt handler or polling to determine termination and branches according to the DMA transfer complete status.

#### Normal termination

1. The program sets the IMK bit in the DMA transfer mode register (DMACTMOD0 or DMACTMOD1) to "1" to mask channel operation.
2. The program writes to the DMA interrupt clear register (DMACCINT0 or DMACCINT1) for the channel to clear the status bits.

#### Abnormal termination

1. The program sets the IMK bit in the DMA transfer mode register (DMACTMOD0 or DMACTMOD1) to "1" to mask channel operation.
2. The program reads the status bits (IREQ, ISTA, and ISTP) from the DMA transfer complete status register (DMAINT).
3. The program writes to the DMA interrupt clear register (DMACCINT0 or DMACCINT1) for the channel to clear the status bits.
4. The program processes the error in accordance with the needs of the user application system.

Table 12.1 summarizes register contents during and after a transfer.

**Table 12.1 Register Contents during and after Transfer**

Register		1. During operation	2. After normal termination	3. After abnormal termination	4. After forced termination
Status register (DMASTA)		1 = Busy (transferring data)	0 = Idle (no data to transfer)	0 = Idle (no data to transfer)	1 = Busy (transferring data)
Interrupt status register (DMAINT)	IREQ	0 = no effect	1 (*1)	1 (*1)	0 = no effect
	ISTA	0 = no effect	0 = normal termination	1 = abnormal termination	0 = no effect
	ISTP	0 = no effect	0 (initial state)	0 = error from transfer source 1 = error from transfer destination	0 = no effect
Transfer address registers (DMACSAD0 or DMACSAD1, DMACDAD0 or DMACDAD1)		Transferring Address	Address following that for final transfer	Address for error	Address following that at which transfer was interrupted
Transfer count register (DMACSIZE0 or DMACSIZE1)		Number of transfers remaining	0	(*2)	Number of transfers remaining

**Notes**

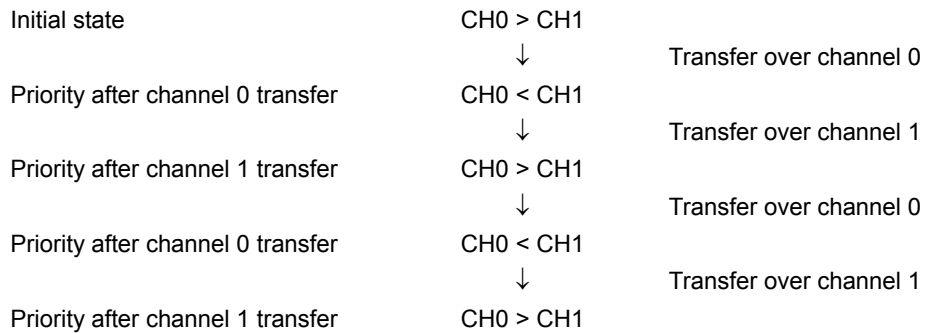
- \*1. The IREQ bit goes to "1" regardless of the interrupt mask (IMK) setting in the corresponding transfer mode register.
- \*2. The contents depend on which cycle triggered the error because the DMA controller decrements the transfer count register (DMACSIZE) only after a successful read from the transfer source.



### 12.3.5 DMA Channel Priority

The PRI bit in the DMA mode register (DMAMOD) specifies the strategy for assigning priority when there are simultaneous transfer requests for both channels 0 and 1: fixed priority or round robin.

- Fixed mode: Channel priority never changes (channel 0 > channel 1)
- Round robin mode: Last channel used has lower priority



**Figure 12.3 Round Robin Operation**

The DMA controller determines the channel priority when it acquires bus access from the CPU. A burst mode transfer over the lower priority channel therefore continues through to completion even if there is a transfer request on the other channel.

### 12.3.6 Important Usage Notes

1. When setting up a DMA transfer, always take into consideration bus width, address alignment, and any other access restrictions in effect for the source and destination devices.
2. There can be no DMA transfers in HALT mode because the CPU is not available to grant bus access.
3. The DMA controller increments the source (DMACSAD0 or DMACSAD1) and destination (DMACDAD0 or DMACDAD1) addresses by the transfer size in bytes if the corresponding device type so specifies. The hardware enforces alignment by internally ignoring the lowest address bits as appropriate for the transfer size, but reading an address register returns those bits exactly as written.

**Example:** Writing address of 0x10000011

Word (32-bit) transfer: The DMA controller internally forces the lowest 2 bits to "00" for an effective address of 0x10000000. Reading the register, however, returns 0x10000011.

Halfword (16-bit) transfer: The DMA controller internally forces the lowest bit to "0" for an effective address of 0x10000010. Reading the register, however, returns 0x10000011.

4. DMA transfers are not possible between certain combinations of source and destination devices. Built-in peripheral registers, for example SIO, UART with FIFO and I2C, can not be set as source or destination of a DMA transfer.

Table 8.2 lists the choices available for the transfer size (TSIZ) field in the DMA transfer mode register (DMAMOD0 or DMAMOD1) for the channel.

		Transfer destination	Internal device (internal SRAM)	External device			
			Incremental address device	Incremental address device		Fixed address device	
Transfer source		Bus width	32bit	8bit	16bit	8bit	16bit
Internal device (internal SRAM)	Incremental address device	32 bit	W/H/B	W/H/B	W/H/B	B	H
		8 bit	W/H/B	W/H/B	W/H/B	B	H
External device	Incremental address device	16 bit	W/H/B	W/H/B	W/H/B	B	H
		8 bit	B	B	B	B	X
	Fixed address device	16 bit	H	H	H	X	H
		8 bit	B	B	B	B	X

- W: Word (32-bit) transfer  
H: Halfword (16-bit) transfer  
B: Byte (8-bit) transfer  
X: Invalid combination

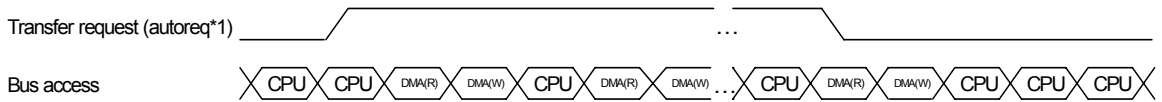
5. Restriction of DMA to External ROM area  
DMA controller can not access external ROM area/BANK25. This restriction applies to external ROM area when it is remapped to BANK0 as well. Thus, the DMA registers DMACSAD0/1 and DMACDAD0/1, should never be set to addresses for BANK25(0xC800\_0000~0xCFFF\_FFFF) or BANK0(0x0000\_0000~0x07FF\_FFFF). These restrictions were implemented in order to optimize efficiency during processor access to external ROM area.
6. Restrictions of DMA to the internal SRAM from external IO module (only ML674001 series)  
When DMA controller transmits to internal RAM from the external IO0/1 or the external IO2/3, in the setting of IO01TYPE= [111] of IO01AC or IO23TYPE= [111] of IO23AC, the DMA transfer in cycle steal mode cannot carry out. Because access to an external bus from CPU cannot be performed until all transmission is completed, even if DMA transfer is set as cycle steal mode. Since it may be in a deadlock state, do not set the above setting.

## 12.4 DMA Transfer Timing

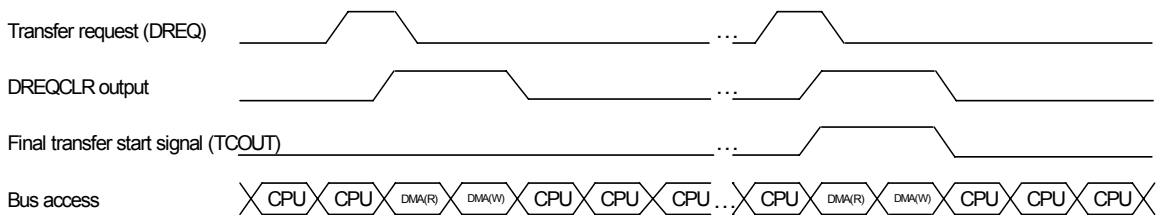
### 12.4.1 Starting a Transfer

At least five clock cycles elapse between a transfer request and the actual start of the DMA transfer.

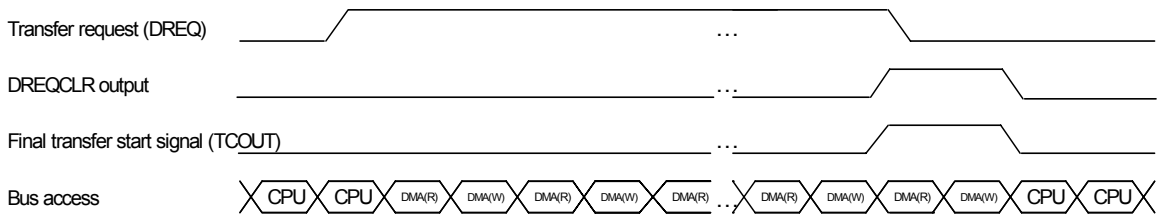
#### (1) Bus access for cycle stealing mode with auto request



#### (2) Bus access for cycle stealing mode with external requests



#### (3) Bus access for burst mode with external requests



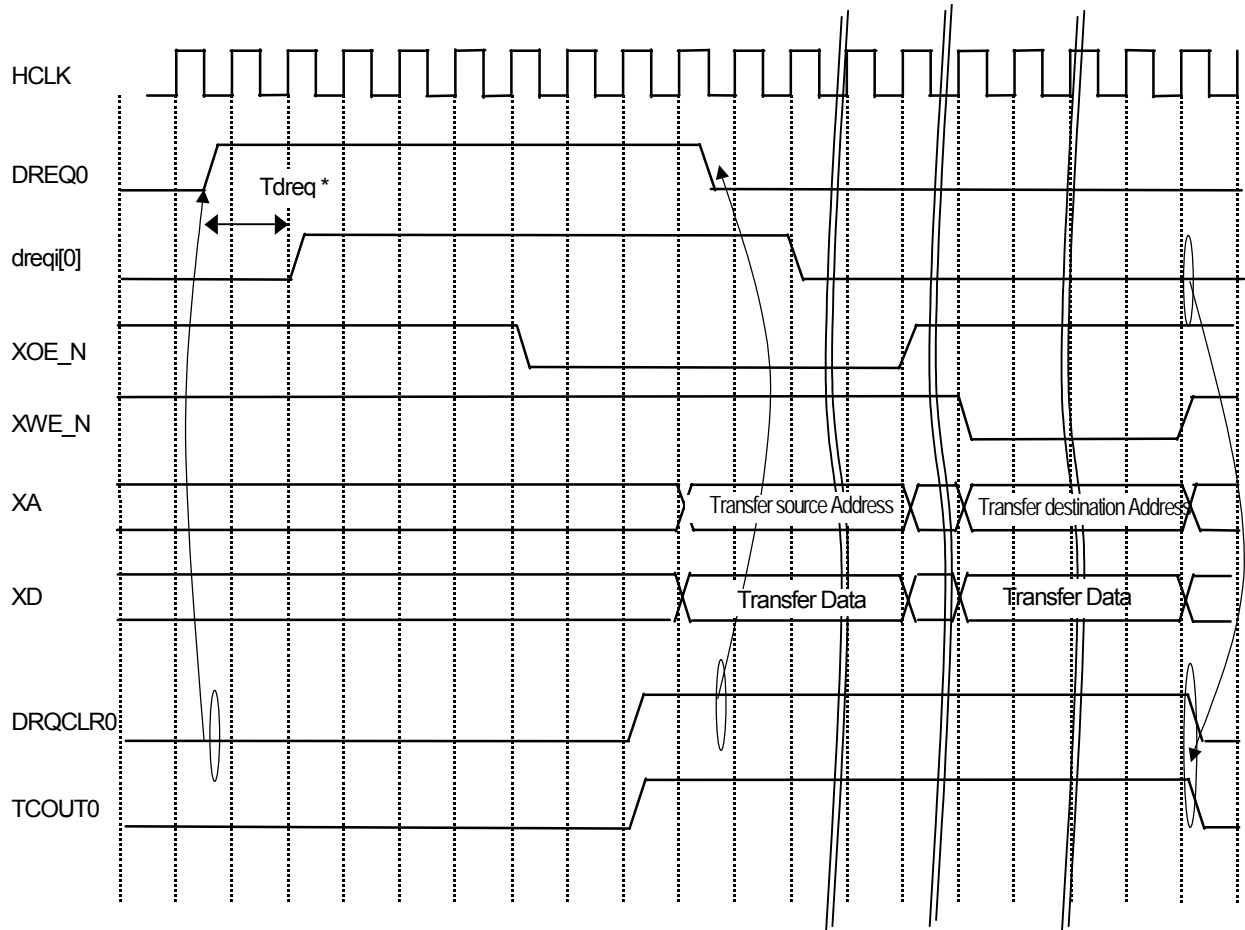
\*1. Setting a bit in the DMA transfer mode register (DMACTMOD0 or DMACTMOD1) for the channel automatically generates this internal transfer request signal.

\*2. The DMA controller uses dual address mode.

**Figure 12.1 Transfer Start Timing**

### 12.4.2 Transfer Timing

Figure 12.2 shows the timing for transferring data from an I/O device on the external bus to memory using cycle stealing mode and DREQ triggers.  
 (External memory bus width is set as 16 bits , the times of the shortest access)



\* Tdreq, the delay between the external request (DREQ) and acceptance of the internal one (dreqi), is a maximum of two clock cycles.

**Figure 12.2 Timing for a Transferring Half Word from External I/O Device to Memory Using Cycle Stealing Mode and DREQ Triggers**

## *Chapter 13*

# **GPIO**

---

## Chapter 13 GPIO

### 13.1 Overview

The four 8-bit GPIO (PIOA/PIOB/PIOC/PIOD) and one 10-bit GPIO (PIOE) ports have the following features.

- Ports: five
- The I/O direction is specified at the individual pin level. After reset, all GPIO pins are configured as inputs by default.
- Edge detection triggers for interrupts, interrupt masks, and interrupt modes settings are specified at the port level.

13.1.1 Components

Figure 13.1 illustrates the internal architecture of the GPIO ports (both the 10-bit and the 8-bit ports). Since all of the pins of the GPIO ports have exactly the same internal structure, only pin 0 is shown in detail below.

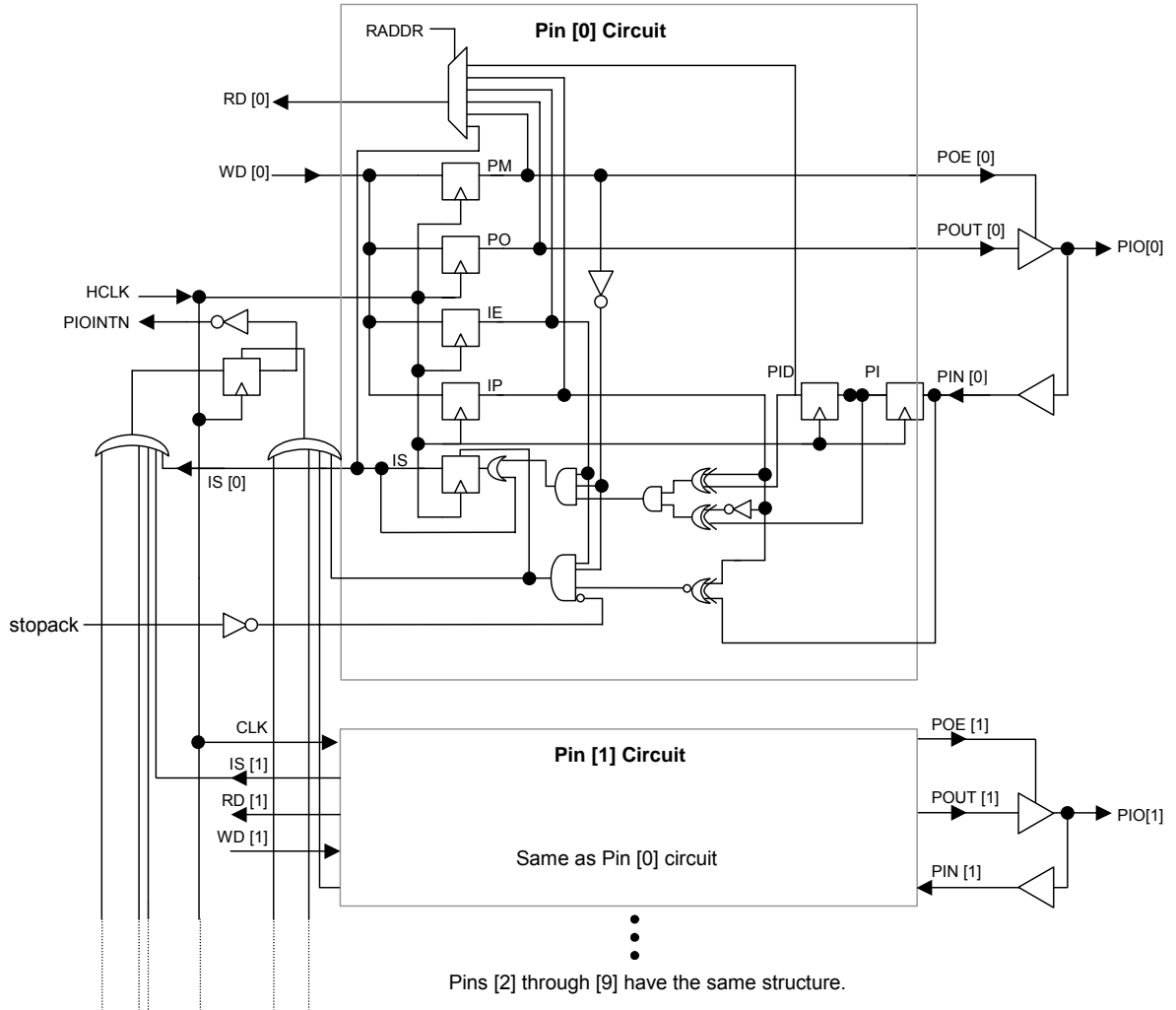


Figure 13.1 PIOx Block Diagram

13.1.2 Pin List

Pin Name	I/O	Function	
		Primary Function	Secondary Function (Pin Name)
PIOA[0]	I/O	General-purpose port A bit 0	SIN
PIOA[1]	I/O	General-purpose port A bit 1	SOUT
PIOA[2]	I/O	General-purpose port A bit 2	CTS
PIOA[3]	I/O	General-purpose port A bit 3	DSR
PIOA[4]	I/O	General-purpose port A bit 4	DCD
PIOA[5]	I/O	General-purpose port A bit 5	DTR
PIOA[6]	I/O	General-purpose port A bit 6	RTS
PIOA[7]	I/O	General-purpose port A bit 7	RI
PIOB[0]	I/O	General-purpose port B bit 0	DREQ0
PIOB[1]	I/O	General-purpose port B bit 1	DREQCLR0
PIOB[2]	I/O	General-purpose port B bit 2	DREQ1
PIOB[3]	I/O	General-purpose port B bit 3	DREQCLR1
PIOB[4]	I/O	General-purpose port B bit 4	TCOUT0
PIOB[5]	I/O	General-purpose port B bit 5	TCOUT1
PIOB[6]	I/O	General-purpose port B bit 6	STXD
PIOB[7]	I/O	General-purpose port B bit 7	SRXD
PIOC[0]	I/O	General-purpose port C bit 0	PWMOUT0
PIOC[1]	I/O	General-purpose port C bit 1	PWMOUT1
PIOC[2]	I/O	General-purpose port C bit 2	XA19
PIOC[3]	I/O	General-purpose port C bit 3	XA20
PIOC[4]	I/O	General-purpose port C bit 4	XA21
PIOC[5]	I/O	General-purpose port C bit 5	XA22
PIOC[6]	I/O	General-purpose port C bit 6	XA23
PIOC[7]	I/O	General-purpose port C bit 7	XWR
PIOD[0]	I/O	General-purpose port D bit 0	XWAIT
PIOD[1]	I/O	General-purpose port D bit 1	XCAS_N
PIOD[2]	I/O	General-purpose port D bit 2	XRAS_N
PIOD[3]	I/O	General-purpose port D bit 3	XSDCLK
PIOD[4]	I/O	General-purpose port D bit 4	XSDCS_N
PIOD[5]	I/O	General-purpose port D bit 5	XSDCKE
PIOD[6]	I/O	General-purpose port D bit 6	XDQM[1]/XCAS_N[1]
PIOD[7]	I/O	General-purpose port D bit 7	XDQM[0]/XCAS_N[0]
PIOE[0]	I/O	General-purpose port E bit 0	SCLK
PIOE[1]	I/O	General-purpose port E bit 1	SDI
PIOE[2]	I/O	General-purpose port E bit 2	SDO
PIOE[3]	I/O	General-purpose port E bit 3	SDA
PIOE[4]	I/O	General-purpose port E bit 4	SCL
PIOE[5]	I/O	General-purpose port E bit 5	EXINT[0]
PIOE[6]	I/O	General-purpose port E bit 6	EXINT[1]
PIOE[7]	I/O	General-purpose port E bit 7	EXINT[2]
PIOE[8]	I/O	General-purpose port E bit 8	EXINT[3]
PIOE[9]	I/O	General-purpose port E bit 9	EFIQ_N



13.1.3 Register List

Address	Name	Abbreviation	R/W	Size	Initial Value
0xB7A01000	Port A output register	GPPOA	R/W	16	Indeterminate
0xB7A01004	Port A input register	GPPIA	R	16	Reflects pin states
0xB7A01008	Port A mode register	GPPMA	R/W	16	0x0000
0xB7A0100C	Port A interrupt enable register	GPIEA	R/W	16	0x0000
0xB7A01010	Port A interrupt polarity register	GPIPA	R/W	16	0x0000
0xB7A01014	Port A interrupt status register	GPISA	R/W	16	0x0000
0xB7A01020	Port B output register	GPPOB	R/W	16	Indeterminate
0xB7A01024	Port B input register	GPPIB	R	16	Reflects pin states
0xB7A01028	Port B mode register	GPPMB	R/W	16	0x0000
0xB7A0102C	Port B interrupt enable register	GPIEB	R/W	16	0x0000
0xB7A01030	Port B interrupt polarity register	GPIPB	R/W	16	0x0000
0xB7A01034	Port B interrupt status register	GPISB	R/W	16	0x0000
0xB7A01040	Port C output register	GPPOC	R/W	16	Indeterminate
0xB7A01044	Port C input register	GPPIC	R	16	Reflects pin states
0xB7A01048	Port C mode register	GPPMC	R/W	16	0x0000
0xB7A0104C	Port C interrupt enable register	GPIEC	R/W	16	0x0000
0xB7A01050	Port C interrupt polarity register	GPIPC	R/W	16	0x0000
0xB7A01054	Port C interrupt status register	GPISC	R/W	16	0x0000
0xB7A01060	Port D output register	GPPOD	R/W	16	Indeterminate
0xB7A01064	Port D input register	GPPID	R	16	Reflects pin states
0xB7A01068	Port D mode register	GPPMD	R/W	16	0x0000
0xB7A0106C	Port D interrupt enable register	GPIED	R/W	16	0x0000
0xB7A01070	Port D interrupt polarity register	GPIPD	R/W	16	0x0000
0xB7A01074	Port D interrupt status register	GPISD	R/W	16	0x0000
0xB7A01080	Port E output register	GPPOE	R/W	16	Indeterminate
0xB7A01084	Port E input register	GPPIE	R	16	Reflects pin states
0xB7A01088	Port E mode register	GPPME	R/W	16	0x0000
0xB7A0108C	Port E interrupt enable register	GPIEE	R/W	16	0x0000
0xB7A01090	Port E interrupt polarity register	GPIPE	R/W	16	0x0000
0xB7A01094	Port E interrupt status register	GPISE	R/W	16	0x0000
0xB7000000	Port function select register	GPCTL	R/W	16	0x0000

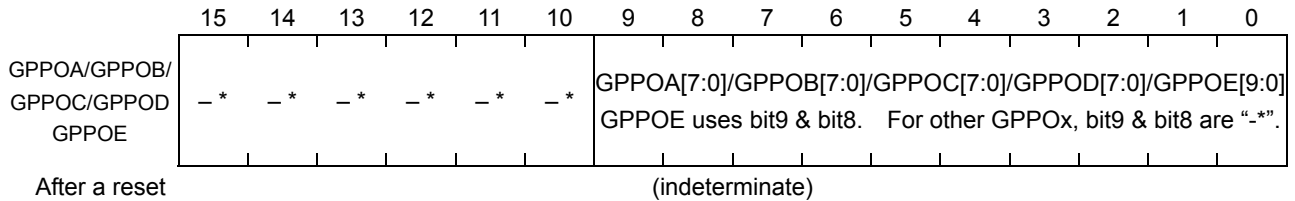
## 13.2 Register Descriptions

### 13.2.1 Port Output Registers (GPPOA, GPPOB, GPPOC, GPPOD and GPPOE)

These registers specify the output levels for the corresponding port pins (PIOA[7:0],PIOB[7:0],PIOC[7:0],PIOD[7:0] and PIOE[9:0]). A pin only has the specified output level, however, when both the GPCTL register configures it for its primary function and the corresponding port mode (GPPMx) register configures it for output.

The CPU has read/write access to these registers.

The register contents after a reset are indeterminate.



Address: 0xB7A01000 (GPPOA), 0xB7A01020 (GPPOB) , 0xB7A01040 (GPPOC) , 0xB7A01060 (GPPOD) , 0xB7A01080 (GPPOE)

Access: R/W

Access size: 16 bits

#### Note

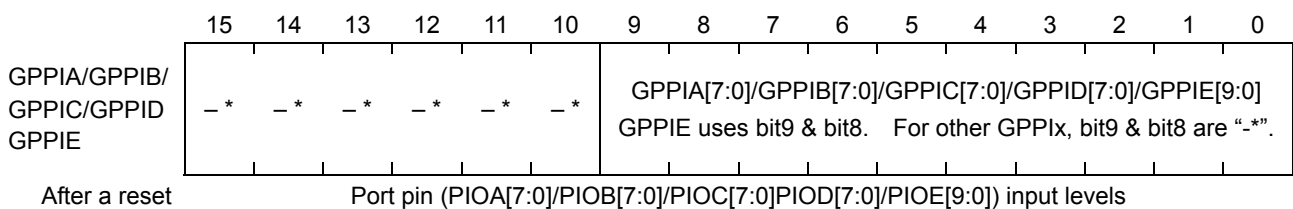
- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

### 13.2.2 Port Input Registers (GPPIA,GPPIB, GPPIC,GPPID, and GPPIE)

These registers reflect the input levels for the corresponding port pins (PIOA[7:0],PIOB[7:0],PIOC[7:0],PIOD[7:0] and PIOE[9:0]). If a pin is configured for output, however, the corresponding bit reflects its output level.

The CPU has only read access to these registers.

The register contents after a reset reflect the port pin (PIOA[7:0],PIOB[7:0],PIOC[7:0],PIOD[7:0] and PIOE[9:0]) input levels.



Address: 0xB7A01004 (GPPIA), 0xB7A01024 (GPPIB) , 0xB7A01044 (GPPIC) , 0xB7A01064 (GPPID) , 0xB7A01084 (GPPIE)

Access: R

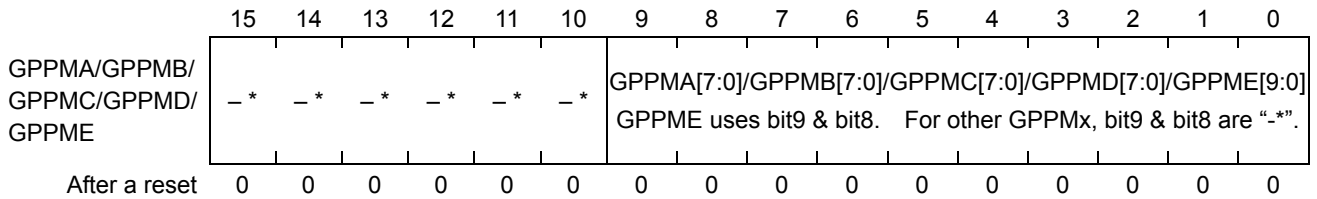
Access size: 16 bits

13.2.3 Port Mode Registers (GPPMA,GPPMB, GPPMC, GPPMD and GPPME)

These registers specify the I/O directions for the corresponding port pins (PIOA[7:0], PIOB[7:0], PIOC[7:0], PIOD[7:0] and PIOE[9:0]) at the individual pin level.

The CPU has read/write access to these registers.

The register contents after a reset are 0x0000.



Address: 0xB7A01008 (GPPMA), 0xB7A01028 (GPPMB) , 0xB7A01048 (GPPMC) , 0xB7A01068 (GPPMD) , 0xB7A01088 (GPPME)

Access: R/W

Access size: 16 bits

**Note**

- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

**Bit Descriptions**

- **GPPMA[7:0]/GPPMB[7:0]/ GPPMC[7:0]/GPPMD[7:0]** (bits 0 to 7)/**GPPME[9:0]** (bits 0 to 9): These bits specify the I/O directions for the corresponding port pins.

GPPMA[7:0] / GPPMB[7:0]/ GPPMC[7:0]/ GPPMD[7:0]/ GPPME[9:0]	Description
0	Input
1	Output

### 13.2.4 Port Interrupt Enable Registers (GPIEA,GPIEB, GPIEC,GPIED and GPIEE)

These registers enable or disable the interrupt detection for the corresponding GPIO pins. The interrupt function only works if the particular PIO pin is configured as input (GPPMx register). This interrupt detection requires that the desired trigger-edge-polarity be specified in the GPIPx register.

There are no such interrupt requests for pins configured for output. If configured as output, the interface ignores the corresponding bits in these registers.

The CPU has read/write access to these registers. The register contents after a reset are 0x0000.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIEA/GPIEB/ GPIEC/GPIED /GPIEE	-*	-*	-*	-*	-*	-*	GPIEA[7:0] / GPIEB[7:0]/ GPIEC[7:0]/ GPIED[7:0]/ GPIEE[9:0] GPPEE uses bit9 & bit8. For other GPPEX, bit9 & bit8 are "-*".									
After a reset	0	0	0	0	0	0	0									

Address: 0xB7A0100C (GPIEA), 0xB7A0102C (GPIEB), 0xB7A0104C (GPIEC), 0xB7A0106C (GPIED), 0xB7A0108C (GPIEE)

Access: R/W

Access size: 16 bits

#### Note

-\*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

PIOx pin input restarts the clock, releasing the LSI from STANDBY mode, if the pin is configured for input and interrupts are enabled. The outside source must, however, maintain the input signal at the specified level until clock supply resumes. For further details on starting and stopping the clock signal, see Chapter 7 "Power Management."

#### Bit Descriptions

- GPIEA[7:0]/GPIEB[7:0]/ GPIEC[7:0]/GPIED[7:0]** (bits 0 to 7)/**GPIEE[9:0]** (bits 0 to 9):  
 These bits control the masking of interrupt requests from the corresponding port pins.  
 (PIOA[7:0], PIOB[7:0], PIOC[7:0], PIOD[7:0] and PIOE[9:0])

<b>GPIEA[7:0] / GPIEB[7:0]/ GPIEC[7:0]/ GPIED[7:0]/ GPIEE[9:0]</b>	<b>Description</b>
0	Disable interrupts
1	Enable interrupts

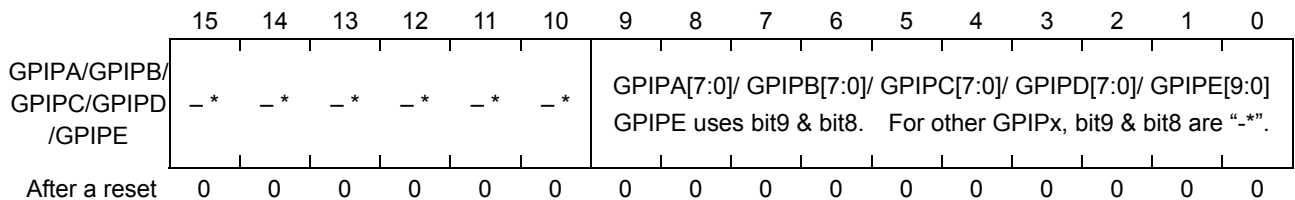
13.2.5 Port Interrupt Polarity Register (GPIPA,GPIPB,GPIPC,GPIPD and GPIPE)

These registers specify the triggering edge for interrupt requests from the corresponding port pins (PIOA[7:0],PIOB[7:0],PIOC[7:0],PIOD[7:0] and PIOE[9:0]).

Detection of an edge with the specified polarity in the input from a PIOx pin configured (GPPMx register) for input with interrupts enabled (GPIEx register) triggers an interrupt request.

The CPU has read/write access to these registers.

The register contents after a reset are 0x0000.



Address: 0xB7A01010 (GPIPA), 0xB7A01030 (GPIPB) , 0xB7A01050 (GPIPC) , 0xB7A01070 (GPIPD) , 0xB7A01090 (GPIPE)

Access: R/W

Access size: 16 bits

**Note**

- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

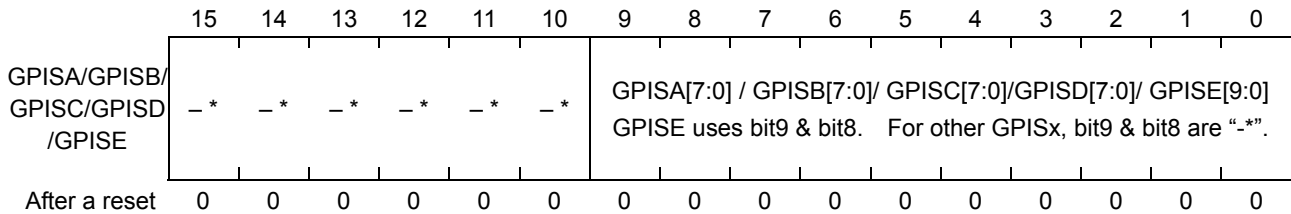
**Bit Descriptions**

- **GPIPA[7:0]/GPIPB[7:0]/GPIPC[7:0]/GPIPD[7:0](bits 0 to 7)/ GPIPE[9:0] (bits 0 to 9):**  
These bits specify the triggering edge for the corresponding GPIO pin at the individual pin level.

GPIPA[7:0] / GPIPB[7:0]/ GPIPC[7:0]/ GPIPD[7:0]/ GPIPE[9:0]	Description
0	Falling edge of input signal triggers interrupt request
1	Rising edge of input signal triggers interrupt request

### 13.2.6 Port Interrupt Status Registers (GPISA,GPISB, GPISC,GPISD and GPISE)

These registers contain flags indicating the sources for pending interrupt requests. Writing "1" to a bit resets it to "0." Writes of "0" are ignored. The register contents after a reset are 0x0000.



Address: 0xB7A01014 (GPISA), 0xB7A01034 (GPISB), 0xB7A01054 (GPISC), 0xB7A01074 (GPISD), 0xB7A01094 (GPISE)  
Access: R/W  
Access size: 16 bits

**Note**

- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

**Bit Descriptions**

- **GPISA[7:0]/GPISB[7:0]/ GPISC[7:0]/ GPISD[7:0] (bits 0 to 7)/ GPISE[9:0] (bits 0 to 9):**  
A "1" in a bit indicates an interrupt request pending from the corresponding pin.

GPISA[7:0] / GPISB[7:0]/ GPISC[7:0]/ GPISD[7:0]/ GPISE[9:0]	Description
0	No interrupt pending
1	Interrupt request pending

**Example:**

GPISA bit 0 goes to "1"

- at a rising edge of an input signal going to PIOA[0] pin, when the following register combination is set: GPIPA[0] = "1" plus GPPMA[0] = "0" and GPIEA[0] = "1".
- at a falling edge of an input signal going to PIOA[0] pin, when the following register combination is set: GPIPA[0] = "0" plus GPPMA[0] = "0" and GPIEA[0] = "1"

### 13.2.7 Port Function Select Register (GPCTL)

This register configures groups of GPIO pins within each GPIO port as their primary or secondary functions.

The CPU has read/write access to this register.

The register contents after a reset are 0x0000.



Address: 0xB7000000

Access: R/W

Access size: 16 bits

#### Note

– \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

#### Bit Descriptions

- **GPCTL0** (bit 0):  
This bit controls the function of pins PIOA[7:0]. Their secondary function is as a UART interface.

GPCTL0 = "0" (primary function)		GPCTL0 = "1" (secondary function)	
Function	In/Out	Function	In/Out
PIOA[0]	In/Out	SIN	Input
PIOA[1]	In/Out	SOUT	Output
PIOA[2]	In/Out	CTS	Input
PIOA[3]	In/Out	DSR	Input
PIOA[4]	In/Out	DCD	Input
PIOA[5]	In/Out	DTR	Output
PIOA[6]	In/Out	RTS	Output
PIOA[7]	In/Out	RI	Input

- **GPCTL1** (bit 1):  
This bit controls the function of pins PIOB[7:6]. Their secondary function is as serial interface.

GPCTL1 = "0" (primary function)		GPCTL1 = "1" (secondary function)	
Function	In/Out	Function	In/Out
PIOB[6]	In/Out	STXD	Output
PIOB[7]	In/Out	SRXD	Input

- **GPCTL2** (bit 2):  
This bit controls the function of pins PIOC[6:2]. Their secondary function is as an external bus.

<b>GPCTL2 = "0" (primary function)</b>		<b>GPCTL2 = "1" (secondary function)</b>	
<b>Function</b>	<b>In/Out</b>	<b>Function</b>	<b>In/Out</b>
PIOC[2]	In/Out	XA[19]	Output
PIOC[3]	In/Out	XA[20]	Output
PIOC[4]	In/Out	XA[21]	Output
PIOC[5]	In/Out	XA[22]	Output
PIOC[6]	In/Out	XA[23]	Output

- **GPCTL3** (bit 3): This bit controls the function of pins PIOB[4] and PIOB[1:0]. Their secondary function is as DMA channel 0.

<b>GPCTL3 = "0" (primary function)</b>		<b>GPCTL3 = "1" (secondary function)</b>	
<b>Function</b>	<b>In/Out</b>	<b>Function</b>	<b>In/Out</b>
PIOB[0]	In/Out	DREQ0	Input
PIOB[1]	In/Out	DREQCLR0	Output
PIOB[4]	In/Out	TCOUT0	Output

- **GPCTL4** (bit 4):  
This bit controls the function of pins PIOB[5] and PIOB[3:2]. Their secondary function is as DMA channel 1.

<b>GPCTL4 = "0" (primary function)</b>		<b>GPCTL4 = "1" (secondary function)</b>	
<b>Function</b>	<b>In/Out</b>	<b>Function</b>	<b>In/Out</b>
PIOB[2]	In/Out	DREQ1	Input
PIOB[3]	In/Out	DREQCLR1	Output
PIOB[5]	In/Out	TCOUT1	Output

- **GPCTL5** (bit 5):  
This bit controls the function of pins PIOC[1:0]. Their secondary function is as PWM outputs.

<b>GPCTL5 = "0" (primary function)</b>		<b>GPCTL5 = "1" (secondary function)</b>	
<b>Function</b>	<b>In/Out</b>	<b>Function</b>	<b>In/Out</b>
PIOC[0]	In/Out	PWMOUT0	Output
PIOC[1]	In/Out	PWMOUT1	Output

- **GPCTL6** (bit 6):  
This bit controls the function of pin PIOD[0]. Its secondary function is as external bus WAIT input.

<b>GPCTL6 = "0" (primary function)</b>		<b>GPCTL6 = "1" (secondary function)</b>	
<b>Function</b>	<b>In/Out</b>	<b>Function</b>	<b>In/Out</b>
PIOD[0]	In/Out	XWAIT	Input



- **GPCTL7** (bit 7):  
 This bit controls the function of pin PIOC[7]. Its secondary function is as external bus data direction.

GPCTL7 = "0" (primary function)		GPCTL7 = "1" (secondary function)	
Function	In/Out	Function	In/Out
PIOC[7]	In/Out	XWR	Output

- **GPCTL8** (bit 8):  
 This bit controls the function of pin PIOE[2:0]. Its secondary function is as SSIO.

GPCTL8 = "0" (primary function)		GPCTL8 = "1" (secondary function)	
Function	In/Out	Function	In/Out
PIOE[0]	In/Out	SCLK	In/Out
PIOE[1]	In/Out	SDI	Input
PIOE[2]	In/Out	SDO	Output

- **GPCTL9** (bit 9):  
 This bit controls the function of pin PIOE[4:3]. Its secondary function is as SSIO.

GPCTL9 = "0" (primary function)		GPCTL9 = "1" (secondary function)	
Function	In/Out	Function	In/Out
PIOE[3]	In/Out	SDA	In/Out
PIOE[4]	In/Out	SCL	Output

- **GPCTL10** (bit 10):  
 This bit controls the function of pin PIOE[5]. Its secondary function is as EXINT[0].

GPCTL10 = "0" (primary function)		GPCTL10 = "1" (secondary function)	
Function	In/Out	Function	In/Out
PIOE[5]	In/Out	EXINT[0]	Input

- **GPCTL11** (bit 11):  
 This bit controls the function of pin PIOE[6]. Its secondary function is as EXINT[1].

GPCTL11 = "0" (primary function)		GPCTL11 = "1" (secondary function)	
Function	In/Out	Function	In/Out
PIOE[6]	In/Out	EXINT[1]	Input

- **GPCTL12** (bit 12):  
 This bit controls the function of pin PIOE[7]. Its secondary function is as EXINT[2].

GPCTL12 = "0" (primary function)		GPCTL12 = "1" (secondary function)	
Function	In/Out	Function	In/Out
PIOE[7]	In/Out	EXINT[2]	Input

- **GPCTL13** (bit 13):

This bit controls the function of pin PIOE[8]. Its secondary function is as EXINT[3].

<b>GPCTL13 = "0" (primary function)</b>		<b>GPCTL13 = "1" (secondary function)</b>	
<b>Function</b>	<b>In/Out</b>	<b>Function</b>	<b>In/Out</b>
PIOE[8]	In/Out	EXINT[3]	Input

- **GPCTL14** (bit 14):

This bit controls the function of pin PIOE[9]. Its secondary function is as EFIQ\_N.

<b>GPCTL14 = "0" (primary function)</b>		<b>GPCTL14 = "1" (secondary function)</b>	
<b>Function</b>	<b>In/Out</b>	<b>Function</b>	<b>In/Out</b>
PIOE[9]	In/Out	EFIQ_N	Input

### 13.3 Description of Operation

This MCU has 42 multiplexed input/output pins designed to be easily software configurable to meet various application requirements. The set of registers defined in this section can configure the I/O pins to work in either primary or secondary mode. In PIO mode, these pins can be configured as input or outputs. When in input mode, the ports can be set as interrupt inputs with configurable edge triggering.

When configured in their primary function as PIO pins, three sets of registers provide control at the individual pin level.

- The port mode (GPPMx) registers specifying the pin I/O directions
- The port interrupt enable (GPIEx) registers controlling interrupt requests
- The port interrupt polarity (GPIPx) registers specifying the edge trigger polarity for interrupts

#### 13.3.1 Interrupt Requests

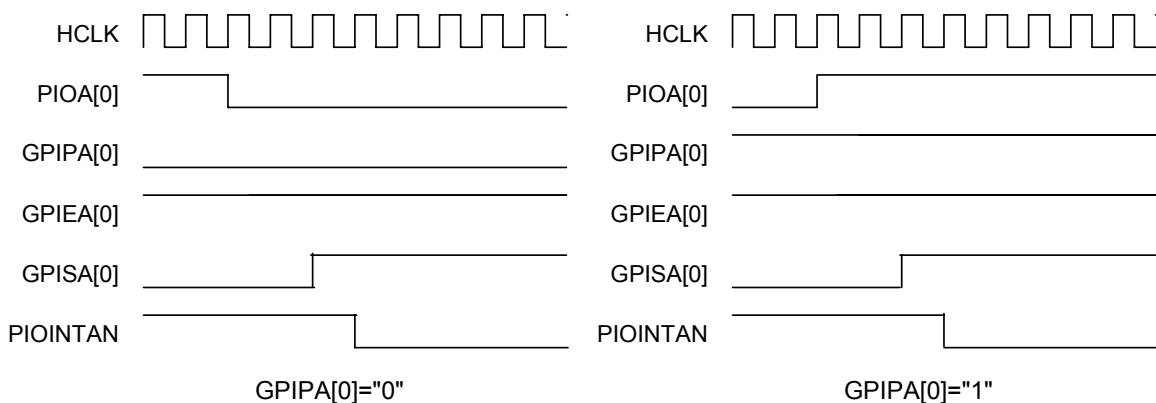
Figure 13.2 illustrates interrupt operation for PIOA pin 0. All pins operate exactly the same way.

As an example, for a given configuration as follows:

1. PIOA[0] pin is configured for primary function (GPCTL[0]="0")
2. PIOA[0] is configured as an input pin (GPPMA[0]="0")
3. PIOA[0] interrupt is enabled (GPIEA[0]=1)
4. PIOA[0] triggering edge has been set to falling edge (GPIPA[0]=0)

If the falling edge of an input signal is detected at the PIOA[0] pin of the MCU, the interrupt status bit will be asserted (GPISA[0]="1") thus asserting the interrupt signal PIOINTAN signal to the interrupt controller.

The above scenario is illustrated in the left timing diagram shown in Figure 13.2.



**Figure 13.2 Interrupt Operation**

### 13.3.2 Primary/Secondary function configuration

PIOA[7:0], PIOB[7:0], PIOC[7:0], PIOD[7:0] and PIOE[9:0] are multiplexed pins that have secondary functions as well. These pins can be configured for their primary or secondary function by setting the port function control register [GPCTL] as described in this chapter.

The port function control register (GPCTL) defines whether each pin should be used in its primary mode as an I/O pin or in the secondary multiplexed mode in its secondary role. For example, PIOA[7:0] pins function as port A GPIO pins in primary mode. But when port function control register configures these pins for their secondary role (GPCTL[0]="1"), these pin take on the role of the UART signals.

PIOD[7:1] are configured by DRAME\_N pin for DRAM access function as explained in Chapter 4 "Chip Configuration" of this manual. By configuring the DRAME\_N pins so as to disables the DRAM controller, the processor will be configured to use the pins PIOD[7:1] in their primary function, thus as GPIOs. By setting the DRAME\_N pins in a configuration to enable the DRAM controller, the pins PIOD[7:1] will be configured in their secondary function which is as control signals for the DRAM bank.

# **Watchdog Timer (WDT)**

---

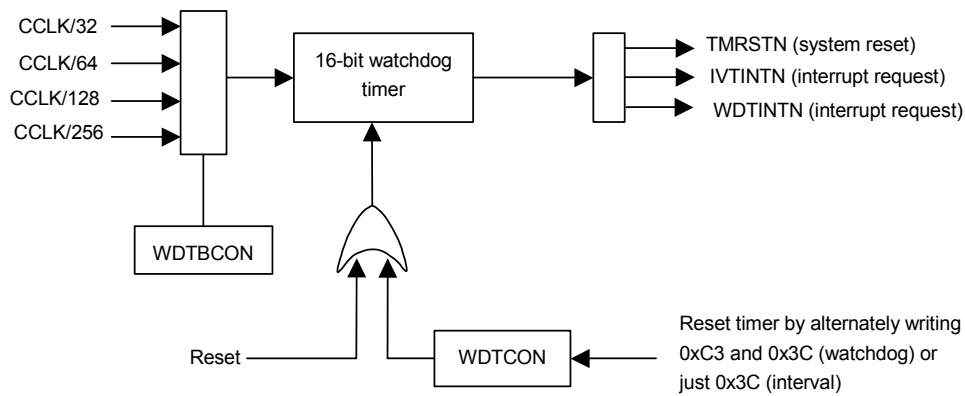
## Chapter 14 Watchdog Timer (WDT)

### 14.1 Overview

The 16-bit watchdog timer monitors whether the program has run out of control. Alternatively, it can be used as an interval timer.

Counter overflow produces an interrupt request or, for the watchdog timer configuration, a forced system reset.

#### 14.1.1 Components



**Figure 14.1 Watchdog Timer Components**

#### 14.1.2 Register List

Address	Name	Abbreviation	R/W	Size	Initial Value
0xB7E00000	Watchdog timer control register	WDTCON	W	8	—
0xB7E00004	Time base counter control register	WDTBCON	R/W	8	0x00
0xB7E00014	Status register	WDSTAT	R/W	8	0x00

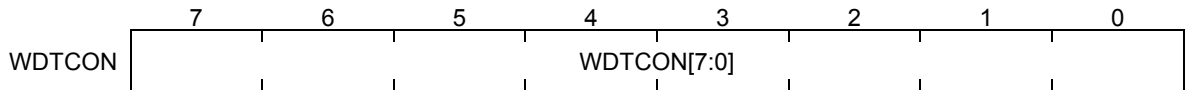
## 14.2 Register Descriptions

### 14.2.1 Watchdog Timer Control Register (WDTCON)

This register is for resetting the watchdog timer to zero.

The CPU has only write access to this register

To start the timer after a system reset, write 0x3C to this register. From that point onward, the program must reset the counter to zero at regular intervals by writing to this register: 0xC3 and 0x3C alternately for watchdog timer operation or just 0x3C for interval timer operation.



Address: 0xB7E0000

Access: W

Access size: 8 bits

### 14.2.2 Time Base Counter Control Register (WDTBCON)

This register controls watchdog timer operation, specifying such things as the operating clock frequency, the operation mode (interval timer or watchdog timer), and whether counter overflow produces an interrupt request or a system reset (watchdog timer operation only).

The CPU has read/write access to this register.

A write protection function shields the contents from random writes. The program must first write 0x5A and then the new value.

The register contents after a reset are 0x00.

	7	6	5	4	3	2	1	0
WDTBCON	WDHLT	OFINT MODE	- *	ITEN	ITM	- *	WDCLK[1:0]	
After a	0	0	0	0	0	0	0	0

Address: 0xB7E00004

Access: R/W

Access size: 8 bits

#### Note

- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

#### Bit Descriptions

- **WDCLK[1:0]** (bits 0 and 1):  
These bits specify the frequency divisor for deriving the operating clock from CCLK.

WDCLK[1:0]		Description
1	0	
0	0	CCLK/32
0	1	CCLK/64
1	0	CCLK/128
1	1	CCLK/256

- **ITM** (bit 3):  
This bit specifies the operation mode: watchdog timer or interval timer.

ITM	Description
0	Watchdog timer
1	Interval timer

- **ITEN** (bit 4):  
This bit controls interval timer (ITM = 1) operation.

ITEN	Description
0	Stop
1	Start/Enable



- **OFINTMODE** (bit 6):  
This bit specifies the action after counter overflow (watchdog operation only).

<b>OFINTMODE</b>	<b>Description</b>
0	Interrupt request
1	System reset

- **WDHLT** (bit 7):  
This bit controls watchdog timer/interval timer operation.

<b>WDHLT</b>	<b>Description</b>
0	Start/Enable
1	Stop

### 14.2.3 Status Register (WDSTAT)

This register gives the status of the watchdog timer interrupt and system reset requests. The CPU has read/write access to this register. The register contents after a reset are 0x00.

	7	6	5	4	3	2	1	0
WDSTAT	— *	— *	IVTIST	WDTIST	— *	— *	— *	RST STATUS
After a	0	0	0	0	0	0	0	0

Address: 0xB7E00014  
Access: R/W  
Access size: 8 bits

#### Note

— \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

#### Bit Descriptions

- **RSTSTATUS** (bit 0):  
This bit gives the reset source. Once this bit goes to "1," it does not return to "0" until there is RESET\_N input.

RSTSTATUS	Description
0	Power on reset
1	Watchdog timer reset

- **WDTIST** (bit 4):  
A "1" in this bit indicates an interrupt request from the watchdog timer. Writing "1" to this bit resets it to "0."

**Note:** Failure to clear this bit after a interrupt/reset request produces a series of such requests. Clearing this bit requires up to 20 HCLK cycles(\*1), so always wait at least 20 HCLK cycles (\*1) before clearing the interrupt controller. (\*1: 40HCLK cycles for ML675001 series.)

WDTST	Description
0	No watchdog timer interrupt request pending
1	Watchdog timer interrupt request pending

- **IVTIST** (bit 5):  
A "1" in this bit indicates an interrupt request from the interval timer. Writing "1" to this bit resets it to "0."

**Note:** Failure to clear this bit after a interrupt/reset request produces a series of such requests. Clearing this bit requires up to 20 HCLK cycles(\*1), so always wait at least 20 HCLK cycles (\*1) before clearing the interrupt controller. (\*1: 40HCLK cycles for ML675001 series.)

IVTST	Description
0	No interval timer interrupt request pending
1	Interval timer interrupt request pending

## 14.3 Description of Operation

The watchdog timer allows the programmer to detect when the program has run out of control by having counter overflow generate an interrupt or system reset request.

If the design does not call for a watchdog timer, this timer is available for use as an interval timer.

### 14.3.1 Operation Modes

The ITM bit in the WDTBCON register specifies the operation mode: watchdog timer or interval timer.

The WDCLK[1:0] bits in the WDTBCON register specify the frequency divisor for deriving the operating clock from CCLK.

### 14.3.2 Interval Timer Operation

Setting both the ITM and ITEN bits in the WDTBCON register to "1" produces interval timer operation. Writing 0x3C to the WDTCON register then starts counting up from zero. Additional writes of the same value reset the counter to zero. Failure to reset the counter in a timely fashion produces overflow and an interrupt request.

This configuration does not generate system reset signals.

### 14.3.3 Watchdog Timer Operation

Setting the ITM bit in the WDTBCON register to "0" produces watchdog timer operation. Writing 0x3C to the WDTCON register then starts counting up from zero. After that, alternately writing 0xC3 and 0x3C resets the timer to zero. Failure to reset the counter in a timely fashion produces overflow and an interrupt or system reset request.

In a typical application, timer overflow should be avoided entirely except as an indication that the program has run out of control.

### 14.3.4 Starting Timer

Specifying the operation mode and other settings in the count operation (WDTBCON) register and then writing 0x3C to the watchdog timer control (WDTCON) register starts counter operation. From that point onward, the program must reset the counter to zero at regular intervals by writing to WDTCON: 0xC3 and 0x3C alternately for watchdog timer operation or just 0x3C for interval timer operation.

Failure to reset the counter in a timely fashion produces overflow and an interrupt or system reset request as specified by the OFINTMODE bit in the WDTBCON register. Note, however, that interval timer operation only produces interrupt requests.

Watchdog timer operation continues even after the CPU shifts to HALT mode. If this behavior is not desirable, set the ITM and ITEN bits in the WDTBCON register to "1" and "0," respectively, to switch to interval timer operation and stop the counter. When the CPU leaves HALT mode, write "0" to the ITM bit to resume watchdog timer operation and restart the counter.

STANDBY mode suspends watchdog timer operation.

## *Chapter 15*

# **Timers**

---

## Chapter 15 Timers

### 15.1 Overview

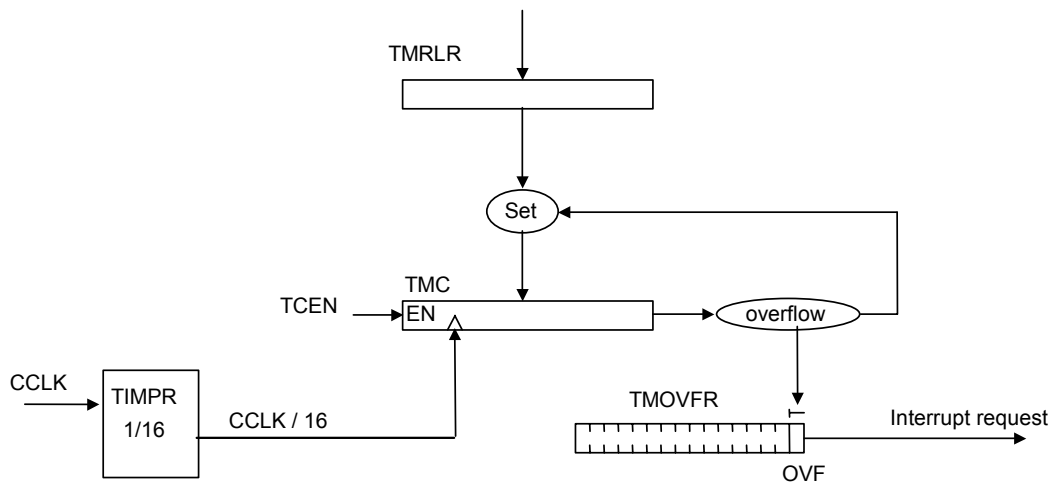
This LSI features one 16-bit System timer and a Timer Counter block which includes six identical 16-bit timer counter channels. The System Timer is internal to the core block of the MCU and is ideal for handling system tasks. The Auto Reload Timer channels are each programmed independently to perform a wide variety of tasks. In addition, each Timer drives an internal interrupt signal, which can be programmed to generate processor interrupts.

The following list summarizes the main features of the System Timer and the Auto Reload Timers:

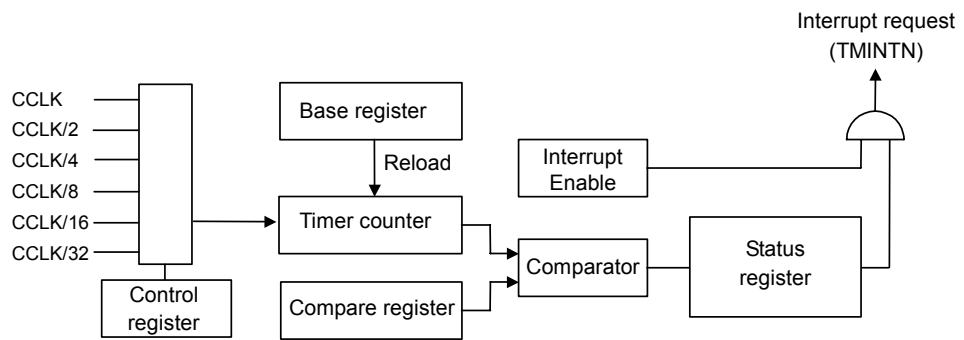
- System Timer
  - 16-bit counter
  - Interrupt request on overflow
  - Count clock is CCLK/16
  - Interval timer operation
- Auto Reload Timers
  - 16-bit counters for each timer
  - Interrupt request on compare match
  - Independent clock settings for each timer
  - Choice of one-shot or interval timer operation for each timer

#### 15.1.1 Components

Figure 15.1 is a block diagram giving components for the system timer; Figure 15.2, one for an auto reload timer.



**Figure 15.1 System Timer Components**



**Figure 15.2 Auto Reload Timer Components**

15.1.2 Register List

Address	Name	Abbreviation	R/W	Size	Initial Value
0xB8001004	System timer enable register	TMEN	R/W	32	0x00000000
0xB8001008	System timer reload register	TMRLR	R/W	32	0x00000000
0xB8001010	System counter overflow register	TMOVFR	R/W	32	0x00000000
0xB7F00000	Timer 0 control register	TIMECNTL0	R/W	16	0x0000
0xB7F00004	Timer 0 base register	TIMEBASE0	R/W	16	0x0000
0xB7F00008	Timer 0 counter register	TIMECNT0	R	16	0x0000
0xB7F0000C	Timer 0 compare register	TIMECMP0	R/W	16	0xFFFF
0xB7F00010	Timer 0 status register	TIMESTAT0	R/W	16	0x0000
0xB7F00020	Timer 1 control register	TIMECNTL1	R/W	16	0x0000
0xB7F00024	Timer 1 base register	TIMEBASE1	R/W	16	0x0000
0xB7F00028	Timer 1 counter register	TIMECNT1	R	16	0x0000
0xB7F0002C	Timer 1 compare register	TIMECMP1	R/W	16	0xFFFF
0xB7F00030	Timer 1 status register	TIMESTAT1	R/W	16	0x0000
0xB7F00040	Timer 2 control register	TIMECNTL2	R/W	16	0x0000
0xB7F00044	Timer 2 base register	TIMEBASE2	R/W	16	0x0000
0xB7F00048	Timer 2 counter register	TIMECNT2	R	16	0x0000
0xB7F0004C	Timer 2 compare register	TIMECMP2	R/W	16	0xFFFF
0xB7F00050	Timer 2 status register	TIMESTAT2	R/W	16	0x0000
0xB7F00060	Timer 3 control register	TIMECNTL3	R/W	16	0x0000
0xB7F00064	Timer 3 base register	TIMEBASE3	R/W	16	0x0000
0xB7F00068	Timer 3 counter register	TIMECNT3	R	16	0x0000
0xB7F0006C	Timer 3 compare register	TIMECMP3	R/W	16	0xFFFF
0xB7F00070	Timer 3 status register	TIMESTAT3	R/W	16	0x0000
0xB7F00080	Timer 4 control register	TIMECNTL4	R/W	16	0x0000
0xB7F00084	Timer 4 base register	TIMEBASE4	R/W	16	0x0000
0xB7F00088	Timer 4 counter register	TIMECNT4	R	16	0x0000
0xB7F0008C	Timer 4 compare register	TIMECMP4	R/W	16	0xFFFF
0xB7F00090	Timer 4 status register	TIMESTAT4	R/W	16	0x0000
0xB7F000A0	Timer 5 control register	TIMECNTL5	R/W	16	0x0000
0xB7F000A4	Timer 5 base register	TIMEBASE5	R/W	16	0x0000
0xB7F000A8	Timer 5 counter register	TIMECNT5	R	16	0x0000
0xB7F000AC	Timer 5 compare register	TIMECMP5	R/W	16	0xFFFF
0xB7F000B0	Timer 5 status register	TIMESTAT5	R/W	16	0x0000

## 15.2 Register Descriptions

### 15.2.1 System Timer Enable Register (TMEN)

This register starts and stops counting by the timer counter (TMC) register. The CPU has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TMEN	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	TCEN
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB8001004

Access: R/W

Access size: 32 bits

#### Notes

—\*: These bits are reserved for future expansion. They return “0” for reads. Writes to them are ignored.

If the timer's operating clock, CCLK with no frequency division, is slower than the bus clock (HCLK), leave the following interval between successive writes to this register. Otherwise, because such writes are synchronized with CCLK not HCLK, the data for the second write might arrive before the hardware has properly latched that for the first write.

$$n \times \text{HCLK} + 32 \times \text{CCLK}$$

where  $n = 16 \times \text{HCLK frequency} / \text{CCLK frequency}$

#### Bit Descriptions

- **TCEN** (bit 0):  
This bit controls system timer operation.

TCEN	Description
0	Stop system timer
1	Start system timer

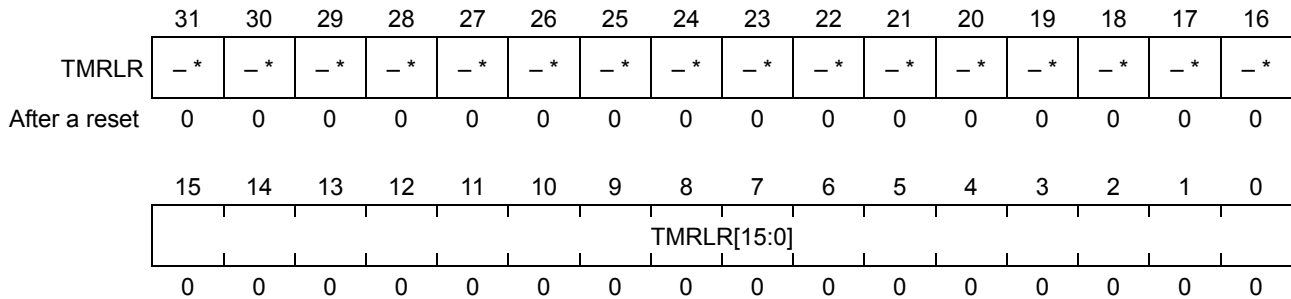


### 15.2.2 System Timer Reload Register (TMRLR)

This register specifies the timer counter (TMC) reload value.

Writing to this register simultaneously writes the same value to the timer counter (TMC) register.

The CPU has read/write access to this register.



Address: 0xB8001008

Access: R/W

Access size: 32 bits

**Note:**

- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

Leave the following interval between successive writes to this register.

$$n \times \text{HCLK} + 79 \times \text{CCLK}$$

### 15.2.3 System Timer Overflow Register (TMOVFR)

The OVF bit in this register goes to “1” when timer counter (TMC) overflow generates a system counter overflow interrupt request.

The CPU has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TMOVFR	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	OVF
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB8001010

Access: R/W

Access size: 32 bits

**Note**

—\*: These bits are reserved for future expansion. They return “0” for reads. Writes to them are ignored.

**Bit Descriptions**

- **OVF (bit 0):**  
 A “1” in this bit indicates counter overflow. Writing “1” to this bit resets it to “0.” Writes of “0” are ignored.

OVF	Description
0	No overflow detected
1	Overflow detected

**Note:** Failure to clear this bit after a interrupt/reset request produces a series of such requests.

### 15.2.4 Timer Control Registers (TIMECNTL0 to TIMECNTL5)

These registers control timer operation, specifying the operating clock, enabling and disabling interrupts, starting and stopping the timer, and specifying the timer operation mode. The CPU has read/write access to these registers.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMECNTL0 to 5	—*	—*	—*	—*	—*	—*	—*	—*	CLKSEL[2:0]		IE	START	—*	—*	MODE	
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB7F00000 (CH0), 0xB7F00020 (CH1), 0xB7F00040 (CH2), 0xB7F00060 (CH3),  
0xB7F00080 (CH4), 0xB7F000A0 (CH5)

Access: R/W

Access size: 16 bits

#### Note

CLKSEL should be set before START bit setting, if application need correct timer interval. Because timer start count with maximum 6 count by the clock that selected by CLKSEL before this register write.

—\*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

#### Bit Descriptions

- **MODE** (bit 0):  
This bit specifies the timer operation mode.

MODE	Description
0	Interval timer
1	One-shot timer

- **START** (bit 3):  
This bit controls timer operation.

START	Description
0	Stop timer
1	Start timer

- **IE** (bit 4):  
This bit controls interrupt requests.

IE	Description
0	Disable interrupts
1	Enable interrupts

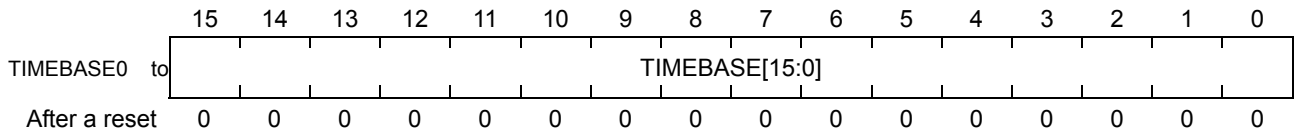
- **CLKSEL** (bit 5 to 7):  
These bits specify, as a power of two, the frequency divisor for deriving the operating clock from CCLK.

CLKSEL			Description
7	6	5	
0	0	0	CCLK
0	0	1	CCLK/2
0	1	0	CCLK/4
0	1	1	CCLK/8
1	0	0	CCLK/16
1	0	1	CCLK/32
1	1	0	(reserved)
1	1	1	(reserved)

### 15.2.5 Timer Base Registers (TIMEBASE0 to TIMEBASE5)

These registers specify the counter contents at the start of operation for the corresponding timer. Writing to one simultaneously writes the same value to the corresponding timer counter register (TIMECNT0 to TIMECNT5).

The CPU has read/write access to these registers.



Address: 0xB7F00004 (CH0), 0xB7F00024 (CH1), 0xB7F00044 (CH2), 0xB7F00064 (CH3),  
0xB7F00084 (CH4), 0xB7F000A4 (CH5)

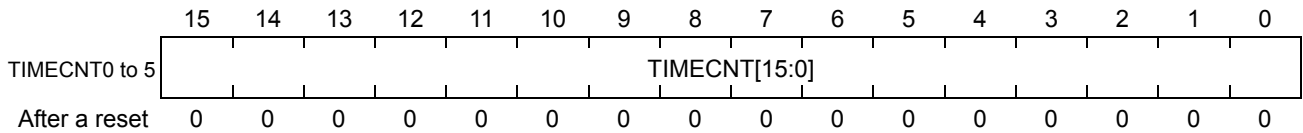
Access: R/W

Access size: 16 bits

### 15.2.6 Timer Counter Register (TIMECNT0 to TIMECNT5)

These registers represent the 16-bit up counters for the timers. Reading one returns the current counter contents.

The CPU has only read access to these registers.



Address: 0xB7F00008 (CH0), 0xB7F00028 (CH1), 0xB7F00048 (CH2), 0xB7F00068 (CH3),  
0xB7F00088 (CH4), 0xB7F000A8 (CH5)

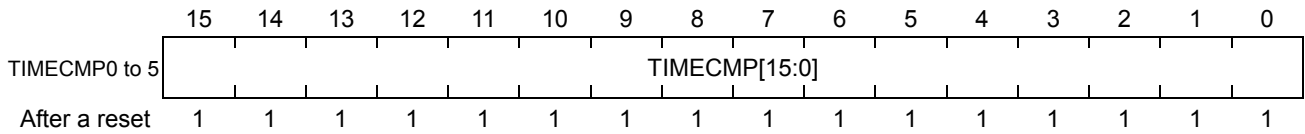
Access: R

Access size: 16 bits

### 15.2.7 Timer Compare Registers (TIMECMP0 to TIMECMP5)

A match between the contents of a timer counter register (TIMECNTx) and those of the corresponding TIMECMPx register triggers an interrupt request and reloads TIMECNTx from the corresponding timer base register (TIMEBASEx).

The CPU has read/write access to these registers.



Address: 0xB7F0000C (CH0), 0xB7F0002C (CH1), 0xB7F0004C (CH2), 0xB7F0006C (CH3),  
0xB7F0008C (CH4), 0xB7F000AC (CH5)

Access: R/W

Access size: 16 bits

## 15.2.8 Timer Status Registers (TIMESTAT0 to TIMESTAT5)

The STATUS bit in this register goes to “1” to indicate a match between the counter and compare registers. The CPU has read/write access to these registers.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMESTAT0 to 5	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	STATUS
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB7F00010 (CH0), 0xB7F00030 (CH1), 0xB7F00050 (CH2), 0xB7F00070 (CH3),  
0xB7F00090 (CH4), 0xB7F000B0 (CH5)

Access: R/W

Access size: 16 bits

**Note**

—\*: These bits are reserved for future expansion. They return “0” for reads. Writes to them are ignored.

## Bit Descriptions

- **STATUS** (bit 0):  
This bit goes to “1” to indicate a match between the counter and compare registers. Writing “1” to this bit resets it to “0.” Writes of “0” are ignored.

STATUS	Description
0	No match
1	Match



## 15.3 Description of Operation

### 15.3.1 System Timer

#### 1. Reload timer operation

The System Timer operates independently from the six Auto Reload Timers. The three registers that control this timer are outlined in the Register List table on page 15-3.

The System Timer is organized around a 16-bit counter. The input time-base to the System Timer block is the CCLK. The CCLK input is divided by a factor of 16 and this forms the time-base for the counter. Once enabled by writing a "1" to the Timer Enable Register (TMEN), the counter counts up each cycle of the 'CCLK/16' input-signal starting from the initial value programmed in to the Timer Counter Register (TMC). Note that when the user program sets the Timer Reload Register (TMRLR), the hardware simultaneously copies the contents of TMRLR to the Timer Counter (TMC) register. The user program does not have direct access to the TMC register.

The value of the counter is incremented at each cycle of the clock. When the counter has reached the value of 0xFFFF, an overflow occurs and bit OVF of the System Timer Overflow Register (TMOVFR) is set to '1', thus asserting an interrupt request signal. In the next processor clock cycle, the timer counter is reloaded with the program specified value from TMRLR register, and continues counting up.

The interval of the System Timer Overflow is defined as follows:

$$\text{Interval}[\mu\text{SEC}] = 16 \times (65536 - \text{TMRLR}[15:0]) / \text{CCLK}[\text{MHz}]$$

The time-base for System Timer is CCLK and as noted in Chapter 7, CCLK can be divided by setting the Clock Gear Control Register (CGBCNT0) as necessary. The user should note that changing CGBCNT0 register will affect the operation of other peripherals in this LSI.

To avoid timing conflicts in the System Timer block, the program must leave the following interval between successive writes to TMRLR register.

$$n \times \text{HCLK} + 79 \times \text{CCLK}$$

The System Timer operation can be stopped at anytime by writing a "0" to the System Timer Enable Register (TMEN).

#### 2. Timer overflow interrupt

Timer counter overflow triggers an interrupt request and sets the OVF bit in the counter overflow register (TMOVFR) to "1."

Writing "1" to TMOVFR clears the interrupt request.

### 15.3.2 Auto Reload Timers

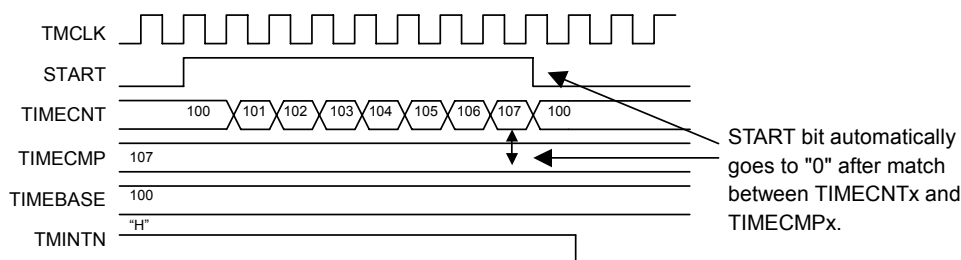
There are six auto reload timers, each with its own counter and settings for such parameters as operating clock frequency and operation mode.

#### 1. One-shot operation

A match between the counter (TIMECNTx) and compare (TIMECMPx) registers reloads TIMECNTx from the corresponding timer base register (TIMEBASEx) and resets the START bit to "0" to stop counting.

The timer then generates an interrupt request if interrupts are enabled.

Note, however, that the hardware does not automatically cancel this interrupt request. The interrupt handler must do so by writing "1" to the STATUS bit.



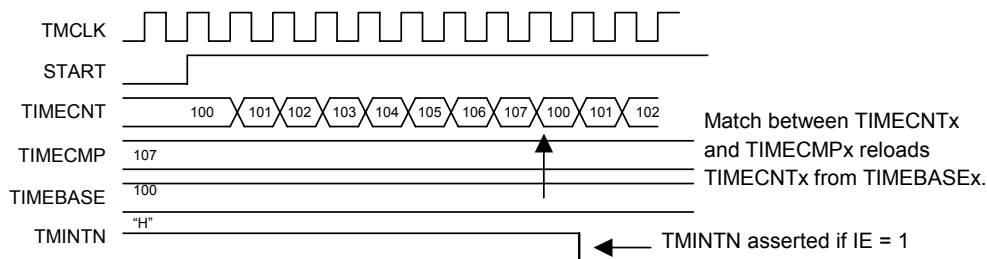
**Figure 15.3 One-Shot Operation**

#### 2. Interval timer operation

A match between TIMECNTx and TIMECMPx reloads TIMECNTx from TIMEBASEx, but does not reset the START bit to "0," so counting continues.

The timer then generates an interrupt request if interrupts are enabled.

Note, however, that the hardware does not automatically cancel this interrupt request. The interrupt handler must do so by writing "1" to the STATUS bit.



**Figure 15.4 Interval Timer Operation**

### 15.3.3 Specifying Clock and Starting Auto Reload Timers

The CLKSEL[2:0] bits in a timer control register (TIMECNTLx) specify, for the corresponding timer, the frequency divisor for deriving the operating clock from CCLK. The choices available are 1, 2, 4, 8, 16, and 32.

Writing "1" to the START bit in the same register starts the timer.

For one-shot operation, a match between the counter and compare registers automatically resets the START bit to "0" to stop counting. Interval timer operation does not clear START, so counting continues.

**Notes:**

1. Operation is not guaranteed for clock (CLKSEL[2:0]) settings "110" and "111."
2. Do not change the clock (CLKSEL[2:0]) setting while the timer is in operation.

## *Chapter 16*

# **PWM Generator**

---

## Chapter 16. PWM Generator

### 16.1 Overview

This block provides two pulse width modulation (PWM) outputs, PWMOUT0 and PWMOUT1, that support changing the duty under program control with each cycle. These outputs have a resolution of 16 bits.

#### 16.1.1 Components

The block diagram in Figure 16.1 illustrates the different components within the PWM Generator.

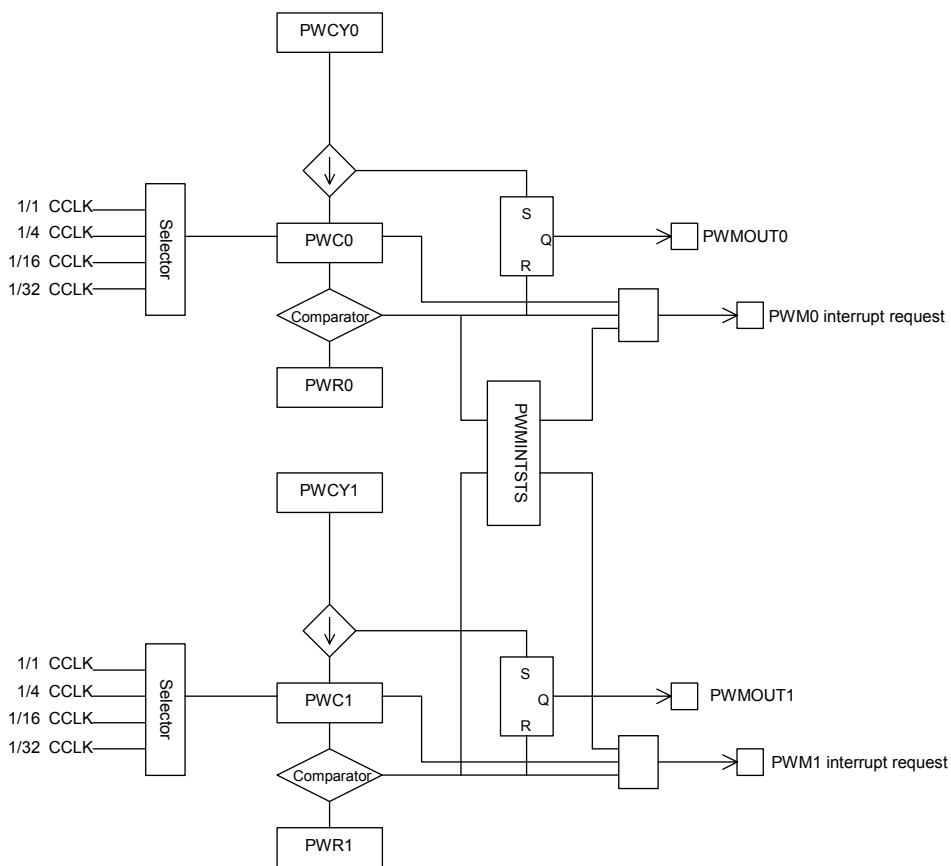


Figure 16.1 PWM Generator Components

## 16.1.2 Pin List

Pin Name	I/O	Description
PWMOUT0	O	PWM output 0. Secondary function for general-purpose port PIOC[0].
PWMOUT1	O	PWM output 1. Secondary function for general-purpose port PIOC[1].

## 16.1.3 Register List

Address	Name	Abbreviation	R/W	Size	Initial Value
0xB7D00000	PWM register 0	PWR0	R/W	16	0x0000
0xB7D00004	PWM period register 0	PWCY0	R/W	16	0x0000
0xB7D00008	PWM counter 0	PWC0	R/W	16	0x0000
0xB7D0000C	PWM control register 0	PWCON0	R/W	16	0x0000
0xB7D00020	PWM register 1	PWR1	R/W	16	0x0000
0xB7D00024	PWM period register 1	PWCY1	R/W	16	0x0000
0xB7D00028	PWM counter 1	PWC1	R/W	16	0x0000
0xB7D0002C	PWM control register 1	PWCON1	R/W	16	0x0000
0xB7D0003C	Interrupt status register	PWINTSTS	R/W	16	0x0000

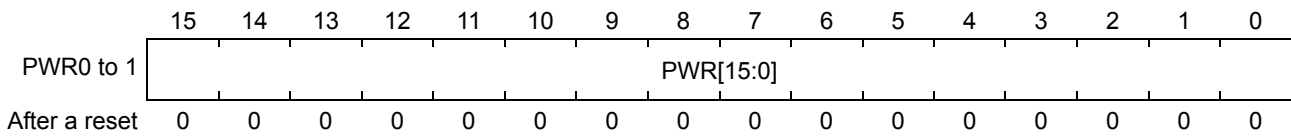
## 16.2 Register Descriptions

### 16.2.1 PWM Registers (PWR0 and PWR1)

These registers specify the duty, the High level pulse width, for the corresponding PWM outputs.  
 Note that the setting must be less than that in the corresponding PWM period register (PWCY<sub>x</sub>), which specifies the PWM output frequency and thus the period. (See register description below.)

$$\text{Duty} = ((\text{PWM} - \text{PWCY} + 1) / (65536 - \text{PWCY})) \times 100 (\%)$$

The CPU has read/write access to these registers.



Address: 0xB7D00000 (CH0), 0xB7D00020 (CH1)

Access: R/W

Access size: 16 bits

### 16.2.2 PWM Period Registers (PWCY0 and PWCY1)

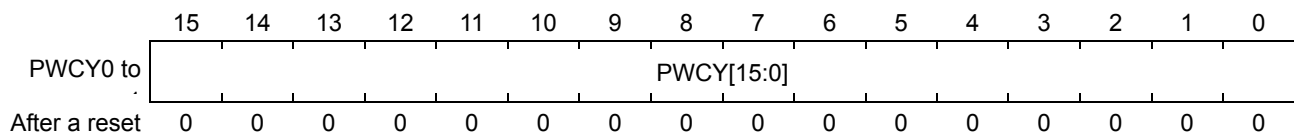
These registers specify the periods for the corresponding PWM outputs according to the following formula.

$$T_{\text{pwmcy}} = (65536 - \text{PWCY}_x) / ((\text{frequency divisor from PWCK}_x) \times \text{CCLK}) \quad [\text{sec}]$$

where the PWCK<sub>x</sub> bits in the corresponding PWM control (PWCON<sub>x</sub>) register specify the frequency divider for CCLK.

For example, when CCLK is 16.67 MHz and PWCK<sub>x</sub> is set to "01" (divisor is 1/4), and PWCY<sub>x</sub> is set to 0xffff, the period for the corresponding PWM output can be calculated as below:

$$\begin{aligned} T_{\text{pwmcy}} &= (65536 - 65535) / (1/4 \times (16.67 \times 10^6)) \quad [\text{sec}] \\ &= 239.952 \quad [\text{nsec}] \end{aligned}$$



Address: 0xB7D00004 (CH0), 0xB7D00024 (CH1)

Access: R/W

Access size: 16 bits

#### Note

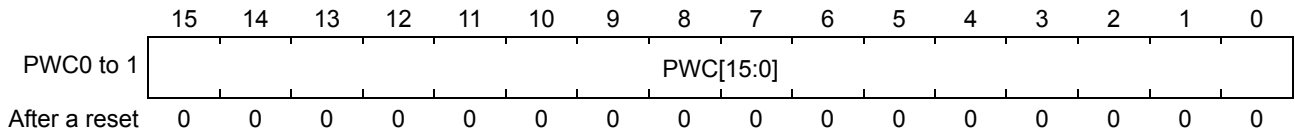
Note that the setting must be greater than that in the corresponding PWM register (PWR<sub>x</sub>), which specifies the duty.



### 16.2.3 PWM Counter (PWC0 and PWC1)

These up counters automatically reload from the corresponding PWM period register (PWCYx) when they overflow.

The CPU has read/write access to these registers.



Address: 0xB7D00008 (CH0), 0xB7D00028 (CH1)

Access: R/W

Access size: 16 bits

**Note**

Writing to PWCx simultaneously writes the same value to PWCYx.

### 16.2.4 PWM Control Registers (PWCON0 and PWCON1)

These registers start and stop the PWM counter, specify the operating clock, and specify the interrupt source.

The CPU has read/write access to these registers.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PWCON0 to 1	-*	-*	-*	-*	-*	-*	-*	-*	PWC0OV/ PWC1OV	INTIE0/ INTIE1	-*	-*	-*	PWCK0/ PWCK1	PW0R/ PW1R	
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB7D0000C (CH0), 0xB7D0002C (CH1)

Access: R/W

Access size: 16 bits

**Note**

- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

**Bit Descriptions**

- **PW0R/PW1R** (bit 0):  
This bit controls PWM counter operation.

PW0R/PW1R	Description
0	Stop PWCx counter
1	Start/enable PWCx counter

- **PWCK0/PWCK1** (bits 1 and 2):  
These bits specify the frequency divisor for deriving the PWCx operating clock from CCLK.

PWCK0/PWCK1		Description
2	1	
0	0	1/1CCLK
0	1	1/4CCLK
1	0	1/16CCLK
1	1	1/32CCLK

- **INTIE0/INTIE1** (bit 6):  
This bit controls interrupt requests.

INTIE0/INTIE1	Description
0	Disable interrupts
1	Enable interrupts

- **PWC0OV/PWC1OV** (bit 7):  
This bit specifies the interrupt request trigger and output timing.

PWC0OV/PWC1OV	Description
0	When PWCx = PWRx (at PWM output falling edge)
1	PWCx overflow (at PWM output rising edge)

### 16.2.5 PWM Interrupt Status Register (PWINTSTS)

This register gives the status of PWM output interrupt requests and provides bits for clearing them. The CPU has read/write access to this register.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PWINTSTS	—*	—*	—*	—*	—*	—*	INT1S	INT0S	—*	—*	—*	—*	—*	—*	INT1CLR	INT0CLR
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB7D0003C

Access: R/W

Access size: 16 bits

#### Notes

- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored. Writes to bits 2 to 15 in this register are ignored.

#### Bit Descriptions

- INT0CLR** (bit 0):  
Writing "1" to this bit clears the PWM output 0 interrupt request (INT0S). Writes of "0" are ignored. Reads always return "0."
- INT1CLR** (bit 1):  
Writing "1" to this bit clears the PWM output 1 interrupt request (INT1S). Writes of "0" are ignored. Reads always return "0."
- INT0S** (bit 8):  
This flag gives the interrupt request status for PWM output 0.

INT0S	Description
0	No interrupt pending
1	Interrupt pending

- INT1S** (bit 9):  
This flag gives the interrupt request status for PWM output 1.

INT1S	Description
0	No interrupt pending
1	Interrupt pending

## 16.3 Description of Operation

### 16.3.1 PWM Operation

This block provides two output pins: PWMOUT0 and PWMOUT1.

To start PWM output, write "1" to PWxR which will in turn start the counter (PWCx) and set the output flip-flop automatically to "1" to drive the PWMOUTx pin output at High level.

When the PWCx contents match those of PWRx, the output flip-flop switches to "0" to drive the PWMOUTx pin output at Low level. If INTIEx is "1" and PWCxOV is "0," an interrupt request will signal this transition.

The hardware repeats the above two steps, producing PWMOUTx pin output whose duty is under program control, until the program resets PWxR to "0."

#### Notes

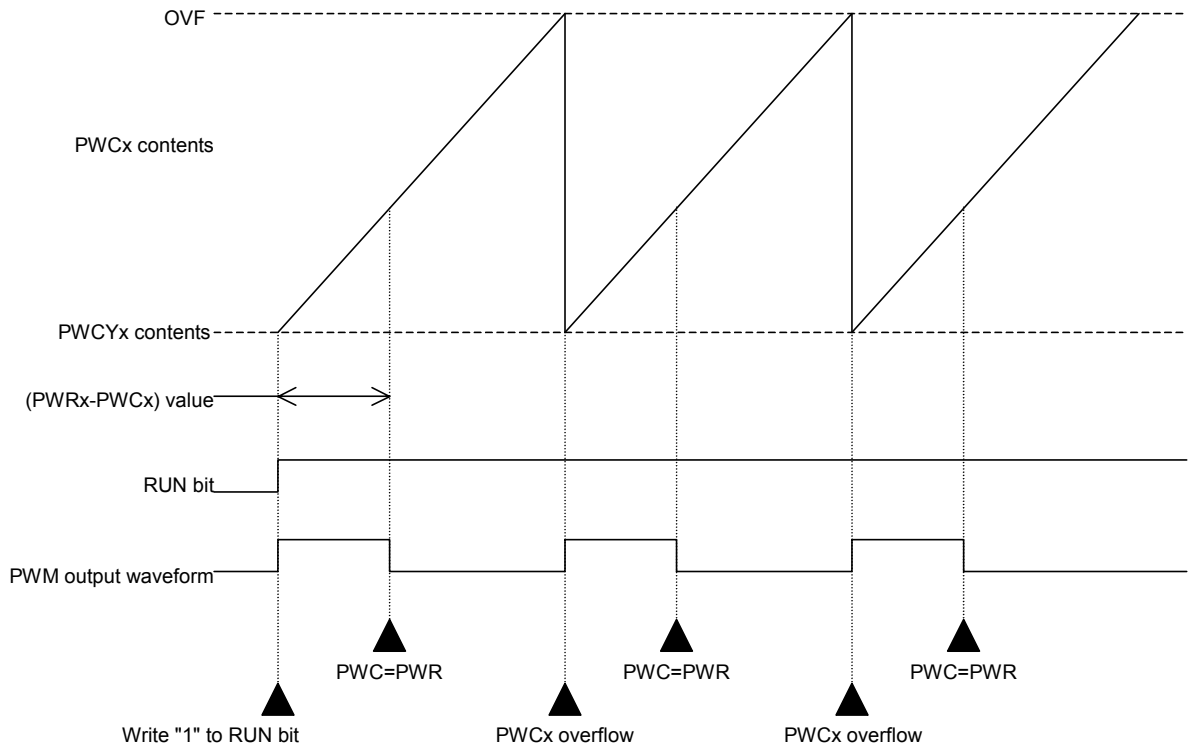
- Depending on the operating clock selected, the PWM output duty is sometimes less than specified for the first cycle immediately after starting output.
- Setting both PWCx and PWRx to 0x0000 produces an output duty (High level pulse width) of 1/65536. Increasing PWRx increases the output duty. The maximum value, 0xFFFF, produces an output duty of 65536/65536 or 100%.  
The PWM generator cannot produce 0/65536 (0%) duty.

### 16.3.2 Timing Examples

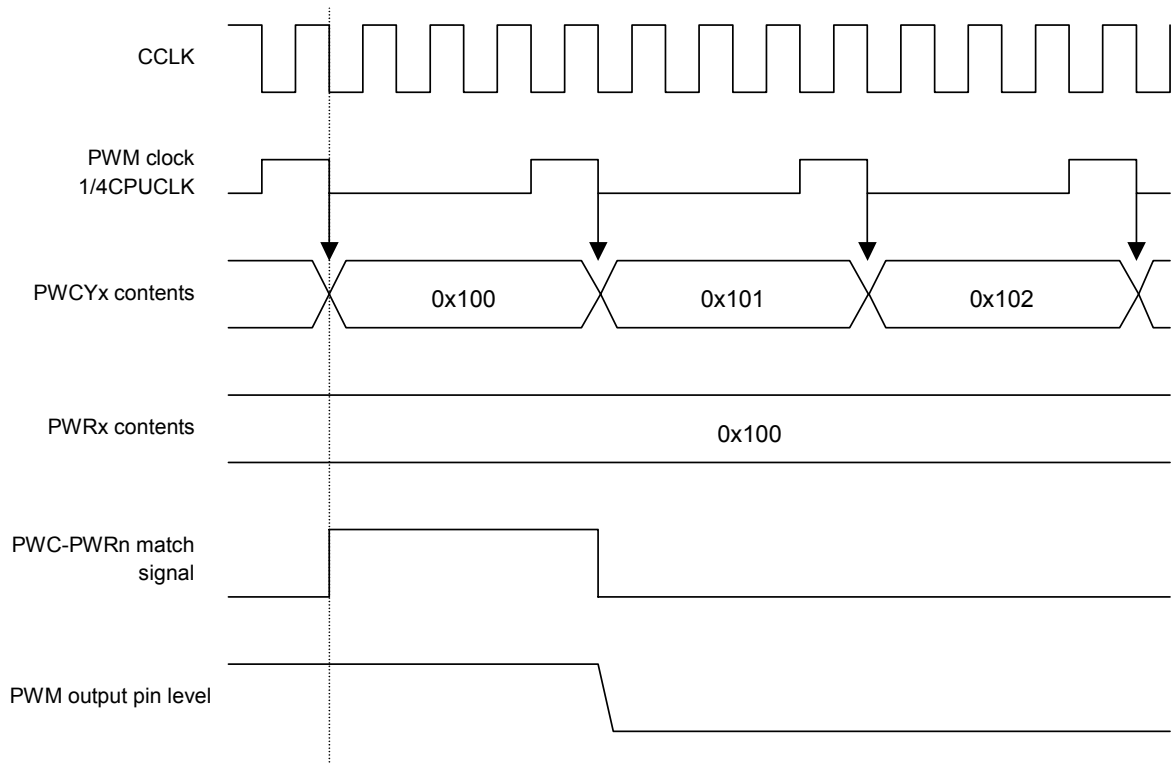
Figure 16.2 summarizes basic output operation. Figure 16.3 shows the results of changing PWM output timing.

#### Notes

- Write to PWC/PWCY/PWR under PWM stop state.
- Next PWM interrupt will be merged if the interrupt occurs before clearing the status bit of previous interrupt.



**Figure 16.2 Basic Output Operation**



**Figure 16.3 Changing PWM Output Timing**

*Chapter 17*

**SIO**

---

## Chapter 17 SIO

### 17.1 Overview

This serial port transfers data, synchronizing individual characters.

A dedicated baud rate generator provides program control over a baud rate clock that determines the transfer speed independently of the bus clock.

Frame parameters available include the character length, number of stop bits, and parity.

### Features

- Full duplex asynchronous operation
- Sampling rate = baud rate  $\times$  16
- Data bits: 7 or 8
- Stop bits: 1 or 2
- Parity: even, odd, or none
- Error detection: parity, framing, and overrun errors
- Loopback function: ON/OFF control plus forced addition of parity, framing, and overrun errors
- Baud rate: Dedicated baud rate generator, with 8-bit counter, providing a baud rate clock that is independent of the bus clock.

#### 17.1.1 Components

Figure 17.1 is a block diagram showing asynchronous serial interface components.

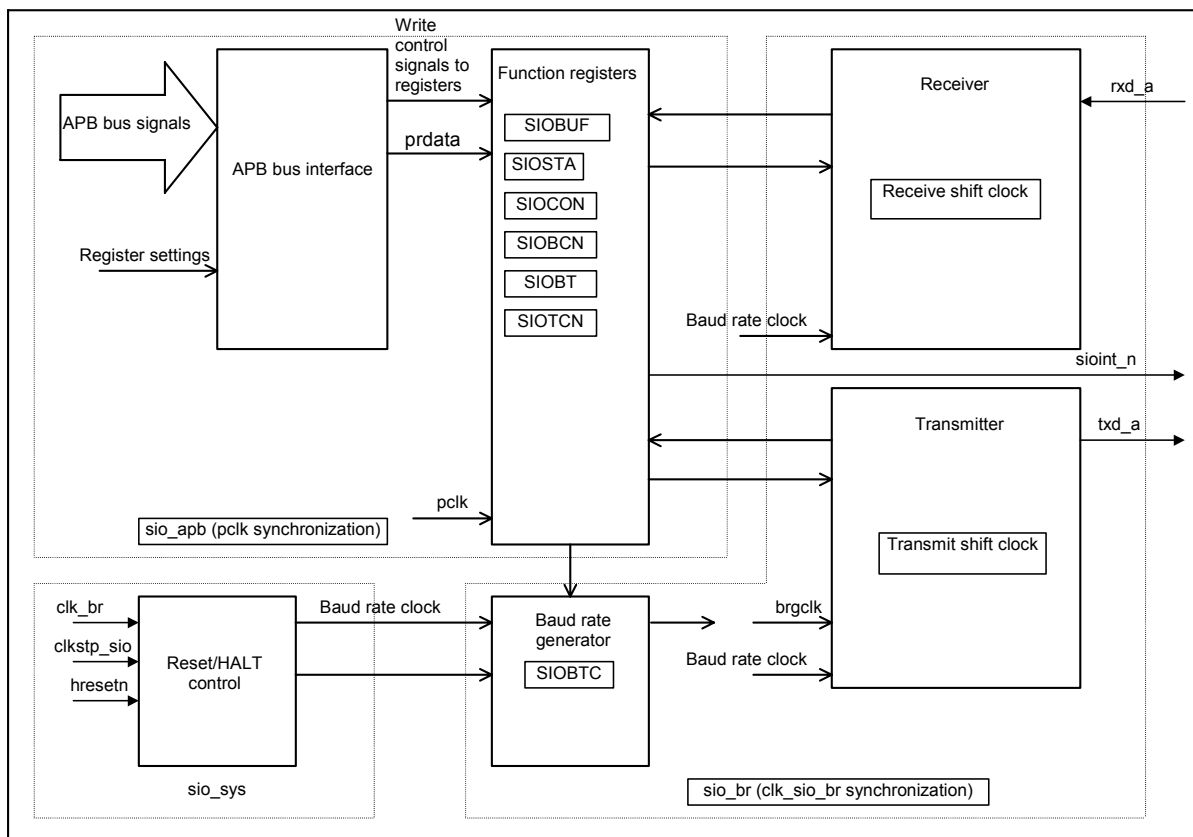


Figure 17.1 Block Components



## 17.1.2 Pin List

Pin Name	I/O	Description
STXD	O	SIO transmitter output. Secondary function for PIOB[6].
SRXD	I	SIO receiver input. Secondary function for PIOB[7].

## 17.1.3 Control Register List

Address	Name	Abbreviation	R/W	Size	Initial Value
0xB8002000	SIO transfer buffer register	SIobuf	R/W	32	0x00000000
0xB8002004	SIO status register	SIOSTA	R/W	32	0x00000000
0xB8002008	SIO control register	SIOCON	R/W	32	0x00000000
0xB800200C	Baud rate control register	SIOBCN	R/W	32	0x00000000
0xB8002010	(reserved)		—	32	0x00000000
0xB8002014	Baud rate timer register	SIOBT	R/W	32	0x00000000
0xB8002018	SIO test control register	SIOTCN	R/W	32	0x00000000

## 17.2 Control Register Descriptions

### 17.2.1 Transfer Buffer Register (SIOBUF)

This register holds transfer data.

The CPU has read/write access to this register, but the same register has two separate functions. It functions as a receive buffer for reads and as a transmit buffer for writes.

When a receive operation ends, the interface transfers the contents of the receive shift register to the receive buffer and generates a receive interrupt request. The receive buffer contents remain valid until the end of the next receive operation.

		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SIOBUF		-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		-*	-*	-*	-*	-*	-*	-*	-*	SIOBUF[7:0]							
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB8002000

Access: R/W

Access size: 32 bits

**Note**

- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

### 17.2.2 SIO Status Register (SIOSTA)

This register contains flags indicating asynchronous serial (SIO) interface transmitter and receiver ready status and ones indicating errors detected during receive operations.

The CPU has read/write access to this register.

The interface updates the lowest three bits at the end of each receive operation to indicate any errors detected. Once an error bit goes to "1" to indicate detection of a receive error, it remains "1" until the program writes "0" to it. It does not automatically return to "0" if the next receive operation is free of the corresponding error.

The program is also responsible for resetting the transmitter and receiver ready flags to "0" by writing "1" to them. Writes of "0" are ignored.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SIOSTA	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	TRIRQ	RVIRQ	-*	PERR	OERR	FERR
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB8002004  
 Access: R/W  
 Access size: 32 bits

**Note**

- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

**Bit Descriptions**

- **FERR** (bit 0):  
 A "1" in this bit indicates detection of a framing error. This error flag goes to "1" when a stop bit of "0" instead of "1" is detected which indicates a loss of frame synchronization.

FERR	Description
0	No framing error detected
1	Framing error detected

- **OERR** (bit 1):  
 A "1" in this bit indicates detection of an overrun error. This bit goes to "1" if the CPU has not read the previously received byte from the receive buffer (SIOBUF) register when the interface transfers the next one there from the receive shift register, invalidating the previous data.

OERR	Description
0	No overrun error detected
1	Overrun error detected

- **PERR** (bit 2):  
 A "1" in this bit indicates detection of a parity error. This bit goes to "1" if the parity bit received does not match the parity calculated for the data bits received.

PERR	Description
0	No parity error detected
1	Parity error detected

- **RVIRQ** (bit 4):

This bit indicates the receiver ready status. This bit goes to "1" when the interface updates the contents of the receive buffer (SIOBUF) from the receive shift register.

Branching to an interrupt handler or reading the received byte from SIOBUF does not automatically reset this bit to "0." The program must reset it to "0" by writing a "1" to it.

Note that the interface always updates SIOBUF and sets this bit to "1" whenever a receive operation is complete regardless of any framing, overrun, or parity errors detected.

RVIRQ	Description
0	Receiver not ready
1	Receiver ready

- **TRIRQ** (bit 5):

This bit indicates the transmitter ready status. This bit goes to "1" when the interface transfers data from the transmit buffer (SIOBUF) to the transmit shift register.

Branching to an interrupt handler or writing transmit data to SIOBUF does not automatically reset this bit to "0." The program must reset it to "0" by writing "1" to it.

TRIRQ	Description
0	Transmitter not ready
1	Transmitter ready

### 17.2.3 SIO Control Register (SIOCON)

This register specifies the frame format for transfers over the asynchronous serial (SIO) interface. The CPU has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SIOCON	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	TSTB	EVN	PEN	LN
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB8002008  
 Access: R/W  
 Access size: 32 bits

#### Notes

- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored. Operation is not guaranteed when the program changes the contents of this register during a transmit or receive operation.

#### Bit Descriptions

- LN (bit 0): This bit specifies the number of data bits for SIO transfers.

LN	Description
0	8 data bits
1	7 data bits

- PEN (bit 1): This bit controls the use of parity bits during SIO transfers. Setting this bit to "1" adds a parity bit to outgoing frames and checks parity for incoming ones.

PEN	Description
0	Disable parity
1	Enable parity

- EVN (bit 2): This bit specifies the parity logic (odd or even) for SIO transfers. It is only valid, however, when PEN is "1."

EVN	Description
0	Odd parity
1	Even parity

- TSTB (bit 3): This bit specifies the number of stop bits for SIO transfers.

<b>TSTB</b>	<b>Description</b>
0	2 stop bits
1	1 stop bit

### 17.2.4 Baud Rate Control Register (SIOBCN)

This register starts and stops the baud rate timer counter (SIOBTC).  
 The CPU has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SIOBCN	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	BGRUN	-*	-*	-*	-*
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB800200C

Access: R/W

Access size: 32 bits

**Note**

– \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

**Bit Descriptions**

- **BGRUN** (bit 4):  
 This bit controls SIOBTC operation.

BGRUN	Description
0	Stop counter
1	Start counter

### 17.2.5 Baud Rate Timer Register (SIOBT)

This register holds the starting value that SIOBTC overflow automatically reloads into the baud rate timer counter (SIOBTC) register.

The CPU has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SIOBT	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*									
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB8002014

Access: R/W

Access size: 32 bits

**Note**

- \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.



### 17.2.6 SIO test control Register (SIOTCN)

This register is for testing the SIO function.  
 The CPU has read/write access to this register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SIOTCN	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*	-*
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-*	-*	-*	-*	-*	-*	-*	-*	LBTST	-*	-*	-*	-*	-*	WPERR	MFERR
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB8002018

Access: R/W

Access size: 32 bits

**Note**

– \*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

**Bit Descriptions**

- **MFERR** (bit 0):  
 Setting this bit to "1" during loopback testing (LBTST = "1") forces framing errors.

MFERR	Description
0	Skip framing errors
1	Add framing errors

- **MPERR** (bit 1):  
 Setting this bit to "1" during loopback testing (LBTST = "1") forces parity errors.

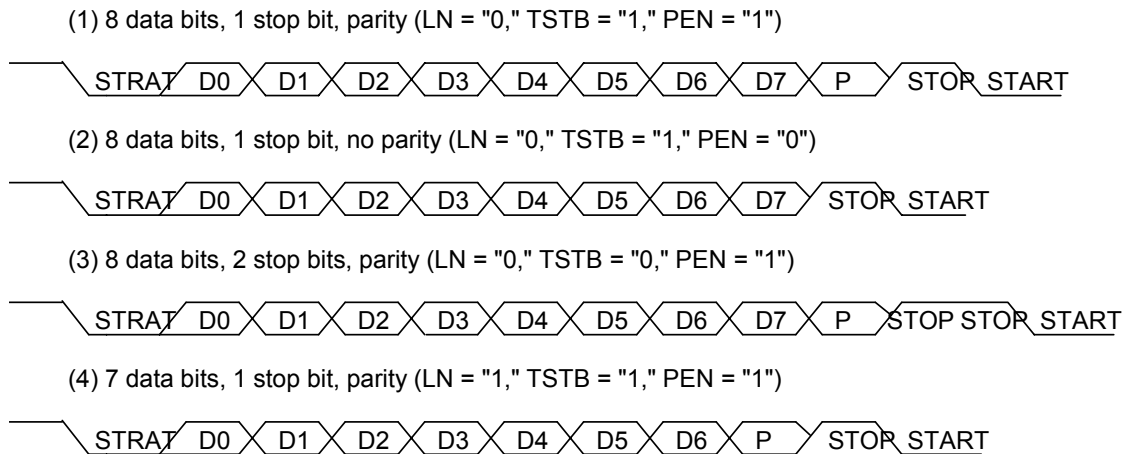
MPERR	Description
0	Skip parity errors
1	Add parity errors

- **LBTST** (bit 7):  
 Setting this bit to "1" internally connects the transmitter outputs to the receiver inputs for testing.

LBTST	Description
0	Disable loopback
1	Enable loopback

### 17.3 Description of Operation

Settings in the SIOCON register specify the frame format: character length, number of stop bits, and parity. Figure 17.2 gives the register settings for sample formats.



**Figure 17.2 Sample Frame Formats**

#### 17.3.1 Transmitting Data

Writing 8-bit data to the transmit buffer (SIOBUF) register starts a transmit operation. Note that the interface does not transmit the top bit if the character length is 7 bits.

When the interface transfers the SIOBUF contents to the transmit shift register, it generates a transmitter ready interrupt request, and sets the TRIRQ bit in the SIOSTA register to "1" to indicate that SIOBUF is now empty, ready for the next write of transmit data to SIOBUF.

The interface then transmits the frame one bit at a time as specified by the SIOCON register settings: start bit, seven or eight data bits, optional parity bit, and one or two stop bits.

#### 17.3.2 Receiving Data

When the interface detects the start bit for a new frame, it begins latching the data bits into the receive shift register as specified by the frame format in the SIOCON register.

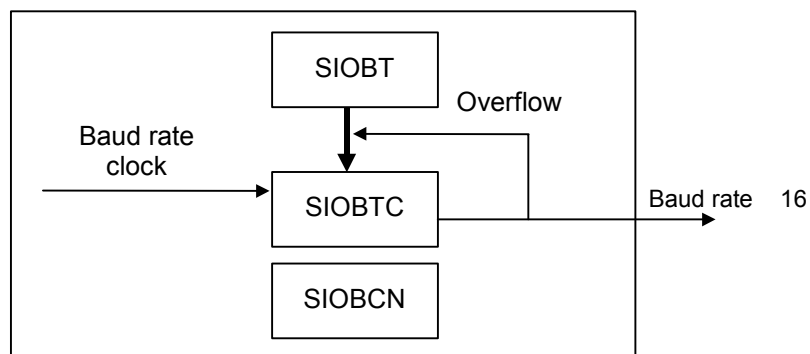
When the interface detects a stop bit, it transfers the received data from the shift register to the receive buffer (SIOBUF) register, signals any errors detected by writing "1" to the corresponding bits (PERR, OERR, and FERR) in the SIOSTA register, sets the RVIRQ bit in the SIOSTA register to "1," and generates a receive ready interrupt request to indicate that the receive shift register is empty, ready to receive the next frame.

### 17.3.3 Generating Baud Rate Clock

Figure 17.3 gives a block diagram showing the following registers for the baud rate generator block.

- Baud rate timer counter (SIOBTC), with 8-bit counter
- Baud rate timer (SIOBT) register
- Baud rate control (SIOBCN) register

Baud rate generator generates the baud rate for serial communication by dividing baud rate clock specified by SIOBT. Writing to SIOBT simultaneously writes the same value to SIOBTC. The baud rate control register (SIOBCN) controls start/stop of a baud rate.



**Figure 17.3 Baud Rate Generator Block Diagram**

The following is the procedure for specifying the baud rate.

1. Calculate the starting value (D) from the desired baud rate frequency (B) and the baud rate clock frequency ( $f_{(B)}$ ) using the following formula. Note, however, that D must be between 0 and 255.

$$D = 256 - \frac{f_{(B)}}{16 \times B} \quad \left( B = f_{(B)} \times \frac{1}{256 - D} \times \frac{1}{16} \right)$$

B: Baud rate

$f_{(B)}$ : Input counter clock frequency, CCLK (Hz)

D: Reload value (0 to 255)

2. Write the starting value (D) to SIOBT.
3. Set the BGRUN bit in the SIOBCN register to "1" to start counting by the SIOBTC register and thus baud rate clock output.  
The interface is ready to transfer data once five baud rate clock cycles have elapsed.

### 17.3.4 Receive Interrupts

A receive ready interrupt request indicates that the interface has transferred data from the receive shift register to the receive buffer (SIOBUF) register, signaled any errors detected by writing "1" to the corresponding bits (PERR, OERR, and FERR) in the SIOSTA register, and set the RVIRQ bit in the SIOSTA register to "1."

This signal remains asserted until the program writes "0" to RVIRQ to negate the interrupt request. Assertion takes precedence over negation if there is any conflict.

### 17.3.5 Transmit Interrupts

A transmit ready interrupt request indicates that the interface has transferred data from the transmit buffer (SIOBUF) register to the transmit shift register and set the TRIRQ bit in the SIOSTA register to "1."

This signal remains asserted until the program writes "0" to TRIRQ to negate the interrupt request. Assertion takes precedence over negation if there is any conflict.

## 17.4 Important Usage Notes

1. Initialize the baud rate generator by writing to the baud rate timer (SIOBT) and baud rate control (SIOBCN) registers in that order. After the write to SIOBCN, the interface allows five baud rate clock cycles for the clock period to stabilize before starting clock output.
2. Operation is not guaranteed if the program changes the SIOCON register settings after transfers start.
3. The interrupt handler must respond and complete its processing within the interval for transferring a single frame.
4. The interrupt handler must clear the interrupt source flag before reading from or writing to the SIOBUF register.
5. Leave at least five baud rate clock cycles between successive writes to the SIOBT register.
6. Register access from the bus must use word access.

## *Chapter 18*

# **UART with FIFO(16byte)**

---

## Chapter 18 UART with FIFO(16byte)

### 18.1 Overview

The UART built into the LSI is an asynchronous communication element (ACE) functionally equivalent to the industry standard 16550A with separate 16-byte queues for both transmitting and receiving.

This UART functions as an I/O interface, converting serial data from a modem or other peripheral equipment into parallel data and converting parallel data from the CPU into serial data.

After a reset, the UART registers function as an industry standard 16450.

During buffered (16550A) operation, each queue holds up to 16 bytes of data.

The receive queue provides three error bits for each byte of data.

The CPU can read the ACE status at any time. The status information available includes the type and status of transfer operations in progress, parity, overrun, framing, and other errors, and the status of break and other interrupts.

### Features

- Full duplex operation
- Reporting functions for all states
- 16-byte queues for both transmitting and receiving
- Transmit, receive, and line status data set interrupts plus independent control over each queue
- Modem control signals CTS, DCD, DSR, DTR, RI, and RTS
- Programmable serial interface
  - Choice of 5, 6, 7, or 8 bits per character
  - Choice of odd, even, or no parity
  - 1, 1.5, or 2 stop bits

18.1.1 Components

Figure 18.1 shows the interface components.

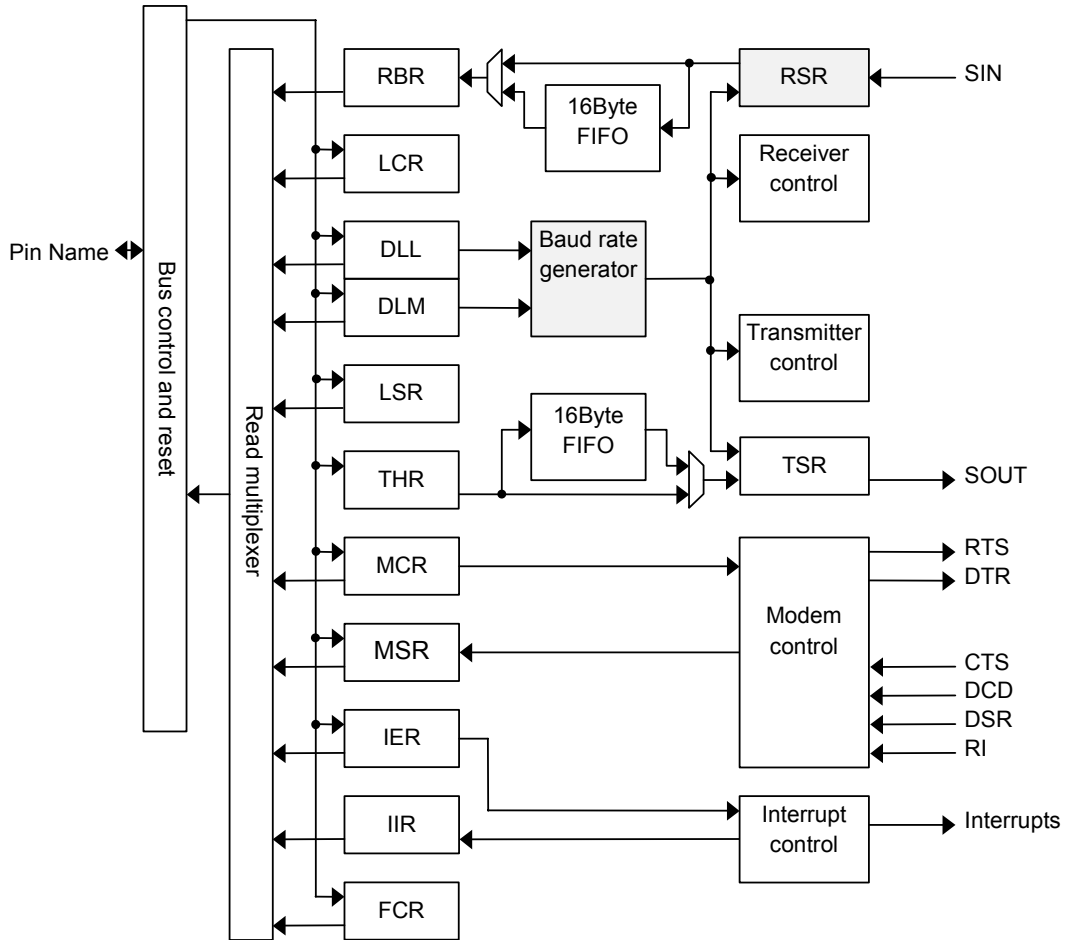


Figure 18.1 Asynchronous Serial Interface Components

18.1.2 Pins

Pin Name	I/O Direction	Description
SIN	I	UART serial data in. Secondary function for PIOA[0].
SOUT	O	UART serial data out. Secondary function for PIOA[1].
CTS	I	UART clear to send. Secondary function for PIOA[2].
DSR	I	UART data set ready. Secondary function for PIOA[3].
DCD	I	UART data carrier detect. Secondary function for PIOA[4].
DTR	O	UART data terminal ready. Secondary function for PIOA[5].
RTS	O	UART request to send. Secondary function for PIOA[6].
RI	I	UART ring indicator. Secondary function for PIOA[7].

Note: The interface does not support OUT1 and OUT2.



### 18.1.3 Register List

Address	Name	Symbol	R/W	Size	Initial Value
0xB7B00000	Receiver Buffer Register	UARTBR	R	8	Indeterminate
0xB7B00000	Transmitter Holding Register	UARTTHR	W	8	Indeterminate
0xB7B00004	Interrupt Enable Register	UARTIER	R/W	8	0x00
0xB7B00008	Interrupt Identification Register	UARTIIR	R	8	0x01
0xB7B00008	FIFO Control Register	UARTFCR	W	8	0x00
0xB7B0000C	Line Control Register	UARTLCR	R/W	8	0x00
0xB7B00010	Modem Control Register	UARTMCR	R/W	8	0x00
0xB7B00014	Line Status Register	UARTLSR	R/W	8	0x60
0xB7B00018	Modem Status Register	UARTMSR	R/W	8	Contents depend on pin states
0xB7B0001C	Scratch Register	UARTSCR	R/W	8	Indeterminate
0xB7B00000	Divisor Latch (LSB)	UARTDLL	R/W	8	Indeterminate
0xB7B00004	Divisor Latch (MSB)	UARTDLM	R/W	8	Indeterminate

## 18.2 Register Descriptions

### 18.2.1 Receiver Buffer Register (UARTRBR)

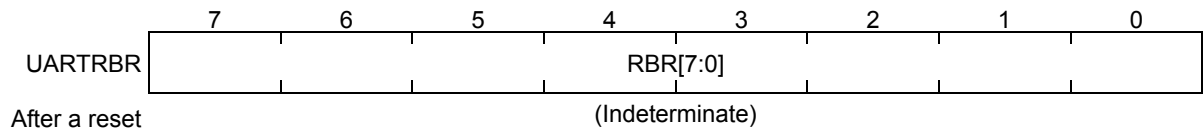
This data register holds 5 to 8 bits of data depending on the character length.

Data transfers always start from the lowest bit (bit 0) in the serial data.

Double-buffering allows the software to read ACE data registers even while the UART is converting data between parallel and serial formats.

The CPU has only read access to this register.

The register contents after a reset are indeterminate.



Address: 0xB7B00000

Access: R

Access size: 8 bits

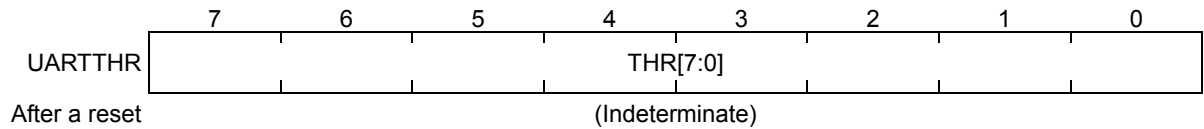
### 18.2.2 Transmitter Holding Register (UARTTHR)

This data register holds 5 to 8 bits of data depending on the character length.

If the character length is less than 8 bits, the bits must be at the low end because data transfers always start from the lowest bit (bit 0).

Double-buffering allows the software to read ACE data registers even while the UART is converting data between parallel and serial formats.

The CPU has only write access to this register.



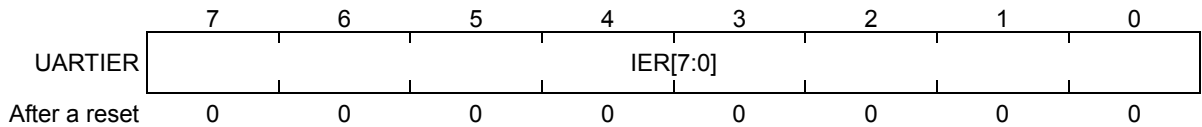
Address: 0xB7B00000

Access: W

Access size: 8 bits

### 18.2.3 Interrupt Enable Register (UARTIER)

This register is for independently enabling four serial interface interrupts.  
 The CPU has read/write access to this register.  
 The register contents after a reset are 0x00.



Address: 0xB7B00004  
 Access: R/W  
 Access size: 8 bits

#### Bit Descriptions

- **IER[0]** (bit 0)  
 This bit enables/disables received data available interrupts (plus character timeout interrupts during buffered operation).

IER[0]	Description
0	Disable received data available interrupts (plus character timeout interrupts during buffered operation)
1	Enable received data available interrupts (plus character timeout interrupts during buffered operation)

- **IER[1]** (bit 1)  
 This bit enables/disables transmitter holding register empty interrupts.

IER[1]	Description
0	Disable transmitter holding register empty interrupts
1	Enable transmitter holding register empty interrupts

- **IER[2]** (bit 2)  
 This bit enables/disables receiver line status interrupts.

IER[2]	Description
0	Disable receiver line status interrupts
1	Enable receiver line status interrupts

- **IER[3]** (bit 3)  
 This bit enables/disables modem status interrupts.

IER[3]	Description
0	Disable modem status interrupts
1	Enable modem status interrupts

- **IER[7:4]** (bits 7 to 4)  
 Unused. These return "0" for reads.

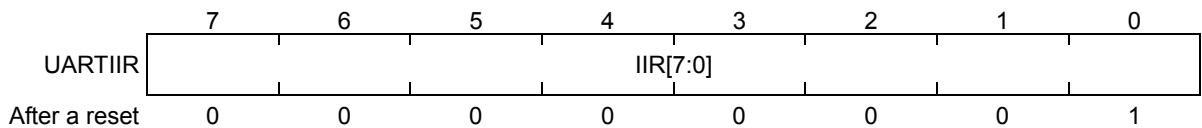
### 18.2.4 Interrupt Identification Register (UARTIIR)

This register indicates whether there is a prioritized interrupt pending and, if so, the source for that interrupt.

The IIR field gives the source for the interrupt with the highest priority. The hardware does not recognize other interrupts until the CPU processes the current interrupt.

The CPU has only read access to this register.

The register contents after a reset are 0x01.



Address: 0xB7B00008

Access: R

Access size: 8 bits

#### Bit Descriptions

- **IIR[0]** (bit 0)  
 This bit indicates whether there is an interrupt pending.

IIR[0]	Description
0	There is an interrupt pending.
1	There are no interrupts pending.

- **IIR[3:1]** (bits 3 to 1)  
 These bits indicate the interrupt source.

Interrupt		Setting and Resetting Interrupt			Notes
IIR [3:1]	Priority level	Interrupt Flag	Interrupt Source	Resetting Interrupt	
011	1	Receiver Line Status	OverrunError/ ParityError/ FlamingError/ BreakInterrupt	Read LSR	
010	2	Received Data Available	Unbuffered (16450) operation: Receive data available Buffered operation: Trigger level reached	Read RBR (until queue below trigger level)	
110	2	Character Timeout Indication	There is at least one character in the receive queue, and there has been no data movement into or off the receive queue for the equivalent of four characters.	Read RBR	Buffered operation only
001	3	Transmitter Holding Register Empty (THRE)	Unbuffered (16450) operation: It is safe to write to THR. Buffered operation: Transmit queue is empty.	Read IIR or write to THR.	
000	4	Modem Status	CTS, DSR, RI, or DCD	Read MSR	

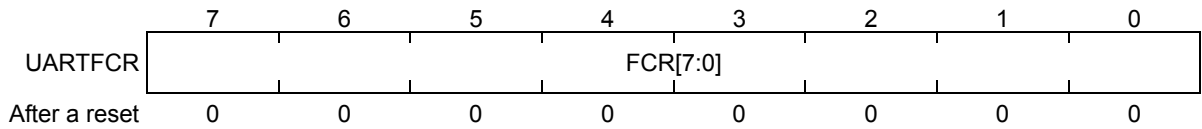
- **IIR[5:4]** (bits 5 to 4)  
Unused. These return "0" for reads.
- **IIR[7:6]** (bits 7 to 6)  
These bits indicate buffered operation.

<b>IIR[7:6]</b>		<b>Description</b>
<b>7</b>	<b>6</b>	
0	0	Unbuffered (16450) operation
0	1	Unused
1	0	Unused
1	1	Buffered operation

### 18.2.5 FIFO Control Register (UARTFCR)

This register enables buffered operation, clears the queues, and specifies the trigger level for the receive queue.

The CPU has only write access to this register.



Address: 0xB7B00008

Access: W

Access size: 8 bits

#### Bit Descriptions

- **FCR[0]** (bit 0)  
This bit switches buffered operation on (“1”) and off (“0”).

FCR[0]	Description
0	Unbuffered (16450) operation
1	Buffered operation

**Note:** Changing this bit automatically clears both queues.

- **FCR[1]** (bit 1)  
RCVR queue reset. Setting this bit to “1” clears the receive queue.

FCR[1]	Description
0	Normal operation
1	Clear receive queue

**Note:** This operation does not clear the receive shift register.

- **FCR[2]** (bit 2)  
XMIT queue reset. Setting this bit to “1” clears the transmit queue.

FCR[2]	Description
0	Normal operation
1	Clear transmit queue

**Note:** This operation does not clear the transmit shift register.

- **FCR[3]** (bit 3)  
DMA mode select

**Note:** This LSI does not support DMA transfers to or from the UART.

- **FCR[5:4]** (bits 5 to 4)  
Reserved

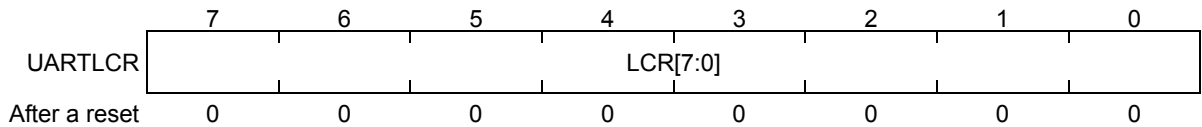
- **FCR[7:6]** (bits 7 to 6)  
RCVR queue interrupt trigger level. These bits specify the trigger level for receive queue interrupts.

<b>FCR[7:6]</b>		<b>Description</b>
<b>7</b>	<b>6</b>	
0	0	1 byte
0	1	4 byte
1	0	8 byte
1	1	14 byte



### 18.2.6 Line Control Register (UARTLCR)

This register specifies the character format.  
 The CPU has read/write access to this register.  
 Allowing reads eliminates the need to save line characters separately in system memory.  
 The register contents after a reset are 0x00.



Address: 0xB7B0000C  
 Access: R/W  
 Access size: 8 bits

#### Bit Descriptions

- **LCR[1:0]** (bits 1 to 0)  
 These bits specify the character length (5 to 8 bits).

LCR[1:0]		Description
LCR[1]	LCR[0]	
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

- **LCR[2]** (bit 2)  
 This bit specifies the number of stop bits for transmitting characters.

LCR[2]	Description
0	1 stop bit
1	2 stop bits (1.5 for character length = 5)

- **LCR[3]** (bit 3)  
 This bit controls the use of parity.

LCR[3]	Description
0	No parity
1	Parity used

- **LCR[4]** (bit 4)  
 This bit specifies even or odd parity. This setting is only valid, however, when parity is enabled (LCR[3] = "1").

LCR[4]	Description
0	Odd parity
1	Even parity

- **LCR[5]** (bit 5)

LCR[5] (bit 5) Stick parity. A “1” in this bit sets all parity bits to a fixed value, the inverse of the bit in LCR[4]. This setting is only valid, however, when parity is enabled (LCR[3] = “1”). The receiver can therefore check the parity using a known state.

LCR[5:3]			Description
LCR[5]	LCR[4]	LCR[3]	
0	0	1	Odd parity
0	1	1	Even parity
1	0	1	Fixed parity (“1”)
1	1	1	Fixed parity (“0”)

- **LCR[6]** (bit 6)

Break control. Setting this bit to “1” sends a break signal by driving the serial output (SOUT) at Low level, or spacing state. Setting this bit to “0” disables the break.

This break control function affects only SOUT, masking SOUT output. The transmit operation continues internally.

The CPU can use this break control function to send a warning to the computer communications system terminal.

LCR[6]	Description
0	Normal operation
1	Transmit break signal

- **LCR[7]** (bit 7)

Divisor latch access bit (DLAB). This bit switches access between UARTDLL and UARTDLM and UARTRBR, UARTTHR, and UARTIER.

LCR[7]	Description
0	Normal operation (accessing UARTRBR, UARTTHR, UARTIIR)
1	Divisor latch (UARTDLL and UARTDLM) access

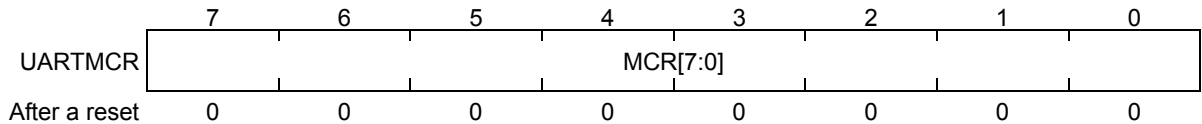
### 18.2.7 Modem Control Register (UARTMCR)

This register controls the modem and data set interface.

The CPU has read/write access to this register.

This register provides direct control over RTS and DTR output. Writing “1” to the corresponding bit drives the pin output at Low level, its active state.

The register contents after a reset are 0x00.



Address: 0xB7B00010

Access: R/W

Access size: 8 bits

#### Bit Descriptions

- **MCR[0]** (bit 0)  
Data terminal ready (DTR) output control.

MCR[0]	Description
0	Drive DTR pin output at High level
1	Drive DTR pin output at Low level

- **MCR[1]** (bit 1)  
Request to send (RTS) output control.

MCR[1]	Description
0	Drive RTS pin output at High level
1	Drive RTS pin output at Low level

- **MCR[2]** (bit 2)  
Reserved

**Note:** This LSI does not support OUT1.

- **MCR[3]** (bit 3)  
Reserved

**Note:** This LSI does not support OUT2.

- **MCR[4]** (bit 4)  
LOOPBACK control. The loop back configuration drives SOUT at High level and connects the transmitter shift register output to the receiver shift register input.

<b>MCR[4]</b>	<b>Description</b>
0	Normal operation
1	Loop back operation

- **MCR[7:5]** (bits 7 to 5)  
Unused. These return "0" for reads.

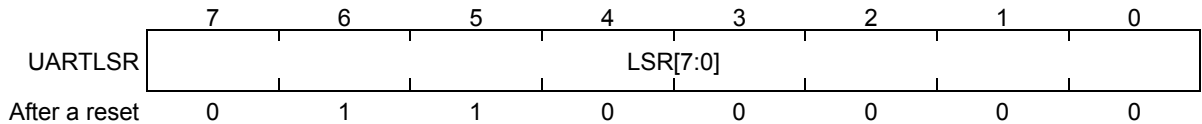
### 18.2.8 Line Status Register (UARTLSR)

This register displays the line status.

It is normally the first register that the CPU reads to determine the interrupt source or to poll the serial interface status.

The CPU has read/write access to this register.

The register contents after a reset are 0x60.



Address: 0xB7B00014

Access: R/W

Access size: 8 bits

#### Bit Descriptions

- **LSR[0]** (bit 0)  
 Data ready bit. This bit goes to “1” when an input character is ready for reading from the UARTRBR register. Reading UARTRBR resets this bit to “0.”

LSR[0]	Description
0	UARTRBR does not contain valid data
1	UARTRBR contains valid data

- **LSR[1]** (bit 1)  
 A “1” in this bit indicates an overrun error.  
 For unbuffered (16450) operation, this indicates that the hardware has overwritten the contents of the UARTRBR register with a new character before the CPU read the former character.  
 For buffered operation, this indicates that the queue is full when the next character is completely received. Reading the UARTLSR register clears the error. Note that the contents of the shift register are not stored in the queue, but overwritten by the next character.  
 Reading the UARTLSR register resets this bit to “0.”

LSR[1]	Description
0	No overrun error pending
1	Overrun error pending

- **LSR[2]** (bit 2)  
 A “1” in this bit indicates a parity error. This setting is only valid, however, when parity is enabled (LCR[3] = “1”).  
 Reading the UARTLSR register resets this bit to “0.”  
 For buffered operation, this indicates a parity error in the data at the head of the queue. Note that parity errors for other characters in the queue do not affect LSR[2] contents.

LSR[2]	Description
0	No parity error pending
1	Parity error pending

- **LSR[3]** (bit 3)

A “1” in this bit indicates a framing error.

A framing error indicates that the corresponding character was not followed by a valid stop bit.

This bit goes to “1” when the bit following the last data bit (or parity bit) is “0” (spacing level), not “1” (stop bit). Reading the UARTLSR register resets this bit to “0.”

For buffered operation, this bit goes to “1” when the character with the framing error reaches the head of the queue.

LSR[3]	Description
0	No framing error pending
1	Framing error pending

- **LSR[4]** (bit 4)

LSR[4] (bit 4) A “1” in this bit indicates a break interrupt, “0” (spacing level) input for one frame interval (start bit, data bits, parity bit, and stop bit).

This bit goes to “1” immediately for unbuffered operation.

For buffered operation, the interface first adds a zero byte to the queue.

Later, when that character reaches the head of the queue, the interface sets this bit to “1” and sets the parity, framing, and overrun error bits (LSR[3:1]) to “0” if the CPU has not already done so by reading the UARTLSR register.

Reading the UARTLSR register resets this bit to “0.”

LSR[4]	Description
0	No break interrupt pending
1	Break interrupt pending

**LSR[1]** to **LSR[4]** transitions to “1” represent sources for receiver line status interrupts, */!/priority/!* 1 interrupts in the interrupt identification register (IIR). Setting IER[2] in the UARTIER register to “1” enables this interrupt.

- **LSR[5]** (bit 5)

Transmitter holding register empty (THRE). A “1” in this bit indicates that the ACE is ready to read a new character to transmit.

This bit goes to “1” when the current character moves from the UARTTHR register to the transmitter shift register. Writing to the UARTTHR register resets this bit to “0.” Reading the UARTLSR register does not.

For buffered operation, this bit goes to “1” when the transmit queue is empty. Writing a byte to the transmit queue resets this bit to “0.”

If IER[1] is “1,” enabling THRE interrupts, the transition to “1” produces a THRE interrupt of priority 3 in the UARTIIR register. If THRE is the source for the interrupt indicated by the UARTIIR register, reading the UARTIIR register resets this bit to “0.”

LSR[5]	Description
0	UARTTHR contains transmit data
1	UARTTHR ready to accept data

- **LSR[6]** (bit 6)

Transmitter empty. This bit goes to “1” when the UARTTHR and transmitter shift (TSR) registers are both empty. Writing a character to the UARTTHR register clears the UARTLSR register to “0,” driving SOUT at Low level until the hardware transmits that character. Reading the UARTLSR register does not reset this bit to “0.”

For buffered operation, this bit goes to “1” when the transmit queue and the shift register are both empty.

<b>LSR[6]</b>	<b>Description</b>
0	There is data in either UARTTHR or TSR
1	UARTTHR and TSR are both empty

- **LSR[7]** (bit 7)

This bit is always “0” during unbuffered (16450) operation.

For buffered operation, this bit goes to “1” when there is a data error (parity error, framing error, or break interrupt) anywhere in the queued data. Reading the data with the error from RBR or discarding it by clearing the entire queue and then reading LSR resets this bit to “0.”

<b>LSR[7]</b>	<b>Description</b>
0	No data errors (buffered operation)
1	Parity error, framing error, or break interrupt (buffered operation)

### 18.2.9 Modem Status Register (UARTMSR)

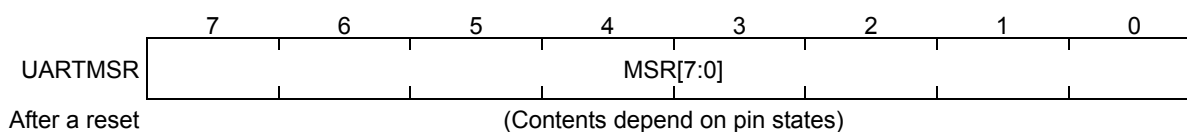
This register allows the CPU to monitor the status of four control signal inputs from the modem or peripheral equipment: CTS, DSR, RI, and DCD. The CPU uses the UARTMSR register to access the ACE data bus interface and read these inputs.

In addition to the four bits giving the current status, this register also provides four delta bits indicating whether these inputs have changed since the CPU last read this register. A delta bit goes to "1" if the corresponding control signal has changed since the last read and returns to "0" when the CPU reads this register.

Bits MSR[4] to MSR[7] monitor CTS, DSR, RI, and DCD, respectively. A "1" indicates Low level input; "0," High. If IER[3] in the interrupt enable register is "1," enabling modem status interrupts, a "0" to "1" transition in the corresponding delta bits MSR[0] to MSR[3] produces a modem status interrupt of priority 4.

The CPU has read/write access to this register.

The register contents after a reset depend on the input pin states.



Address: 0xB7B00018

Access: R/W

Access size: 8 bits

#### Bit Descriptions

- **MSR[0]** (bit 0)

Delta clear to send (DCTS). A "1" in this bit indicates a change in the CTS input state since the last time that the CPU read that state.

MSR[0]	Description
0	No change in CTS input
1	Change in CTS input

- **MSR[1]** (bit 1)

Delta data set ready (DDSR). A "1" in this bit indicates a change in the DSR input state since the last time that the CPU read that state.

MSR[1]	Description
0	No change in DSR input
1	Change in DSR input

- **MSR[2]** (bit 2)

Trailing edge of ring indicator (TERI). A "1" in this bit indicates a change from Low level to High in the RI input state since the last time that the CPU read that state. A transition from High level to Low does not affect this bit.

MSR[2]	Description
0	No change in RI input
1	Change in RI input



- **MSR[3]** (bit 3)  
Delta data carrier detect (DDCD). A “1” in this bit indicates a change in the DCD input state since the last time that the CPU read that state.

MSR[3]	Description
0	No change in DCD input
1	Change in DCD input

- **MSR[4]** (bit 4)  
Clear to send (CTS). This bit, the inverse of the CTS input from the modem, indicates whether the modem is ready to receive data from the serial interface's transmit output (SOUT) pin.  
In loop back mode (MCR[4] = “1”), this bit has the same value as MCR[1].

MSR[4]	Description
0	Modem is not ready to receive data from SOUT (CTS = 1)
1	Modem is ready to receive data from SOUT (CTS = 0)

- **MSR[5]** (bit 5)  
Data set ready (DSR). This bit, the inverse of the DSR input from the modem, indicates whether the modem is ready to transmit data to the serial interface's receive circuit.  
In loop back mode (MCR[4] = “1”), this bit has the same value as MCR[0].

MSR[5]	Description
0	Modem is not ready to transmit data (DSR = 1)
1	Modem is ready to transmit data (DSR = 0)

- **MSR[6]** (bit 6)  
Ring indicator (RI). This bit is the inverse of the RI input from the modem.  
In loop back mode (MCR[4] = “1”), this bit has the same value as MCR[2].

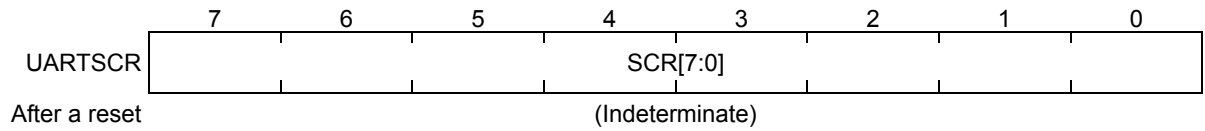
MSR[6]	Description
0	RI = 1
1	RI = 0

- **MSR[7]** (bit 7)  
Data carrier detect (DCD). This bit is the inverse of the DCD input from the modem.  
In loop back mode (MCR[4] = “1”), this bit has the same value as MCR[3].

MSR[7]	Description
0	DCD = 1
1	DCD = 0

### 18.2.10 Scratch Register (UARTSCR)

This register is for use as temporary data storage. It plays no role in ACE transfer operations.  
The CPU has read/write access to this register.  
The register contents after a reset are indeterminate.



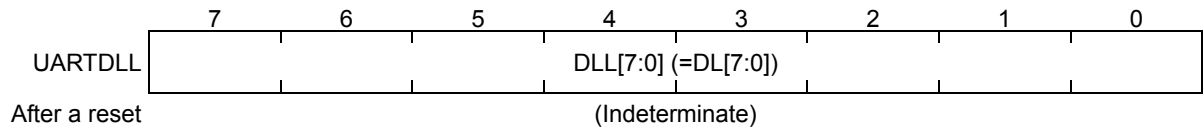
Address: 0xB7B0001C  
Access: R/W  
Access size: 8 bits

### 18.2.11 Divisor Latch (LSB) (UARTDLL)

This register contains the lower half of the 16-bit divisor latch for the baud rate generator. For further details, see the section on baud rate clock generation.

The CPU has read/write access to this register when LCR[7] = 1.

The register contents after a reset are indeterminate.



Address: 0xB7B00000

Access: R/W

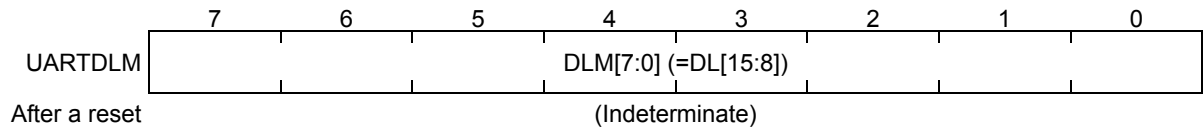
Access size: 8 bits

### 18.2.12 Divisor Latch (MSB) (UARTDLM)

This register contains the upper half of the 16-bit divisor latch for the baud rate generator. For further details, see the section on baud rate clock generation.

The CPU has read/write access to this register when LCR[7] = 1.

The register contents after a reset are indeterminate.



Address: 0xB7B00004

Access: R/W

Access size: 8 bits

### 18.3 Description of Operation

The registers LCR, IER, DLL, DLM, and MCR control serial interface operation. These control registers specify the character length, number of stop bits, parity, baud rate, modem interface, and other parameters. They can be written to in any order except that IER must come last because it enables interrupts. Once the serial interface has been configured for operation, these registers can be updated anytime that the interface is not transferring data.

#### 18.3.1 Transmitting Data

Figure 18.2 gives the timing for transmitting data.

Writing data to the UARTTHR register sends that data to the transmit shift register via the transmit queue. Within 16 baud rate clock cycles after the THRE bit goes to "1," the hardware transmits the start bit and the data bits one bit at a time from the lowest bit. If the character length is 7 bits, the hardware does not transmit the top bit.

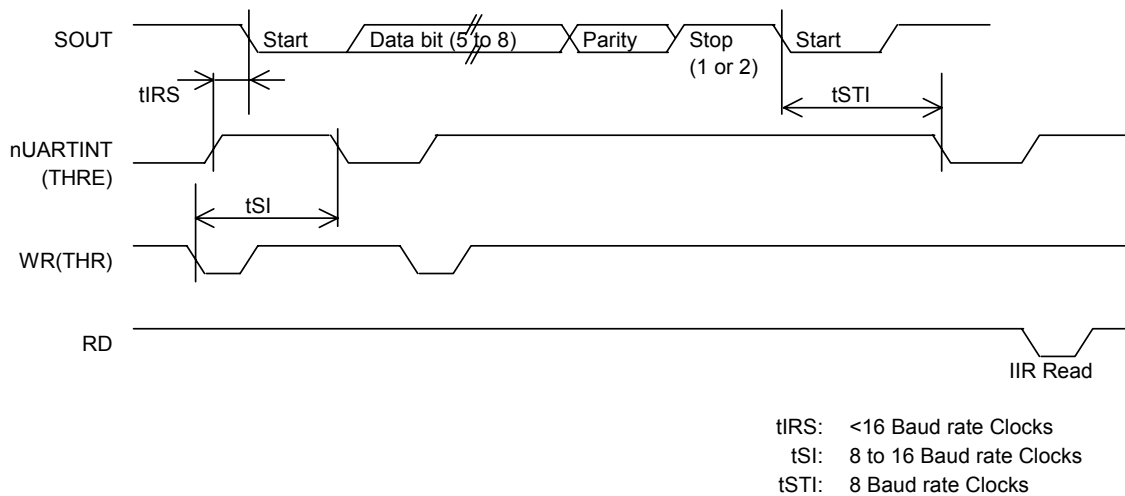
If the LCR[3] bit in the UARTLCR register enables parity, the hardware then transmits the parity bit.

Finally, the hardware transmits a stop bit to complete the frame.

When the hardware finishes transmitting the data, the LSR[5] bit in the UARTLSR register goes to "1" to indicate that the hardware is ready to transmit more data. Writing a byte to the transmit queue resets this bit to "0."

If the IER[1] bit enables THRE interrupts, this transition produces an interrupt of priority 3 in the UARТИIR register.

If THRE is the source for the interrupt indicated by the UARТИIR register, reading the UARТИIR register resets this bit to "0."



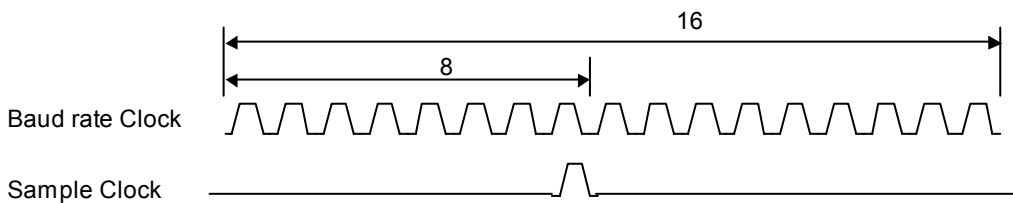
**Figure 18.2 Transmit Timing**

### 18.3.2 Receiving Data

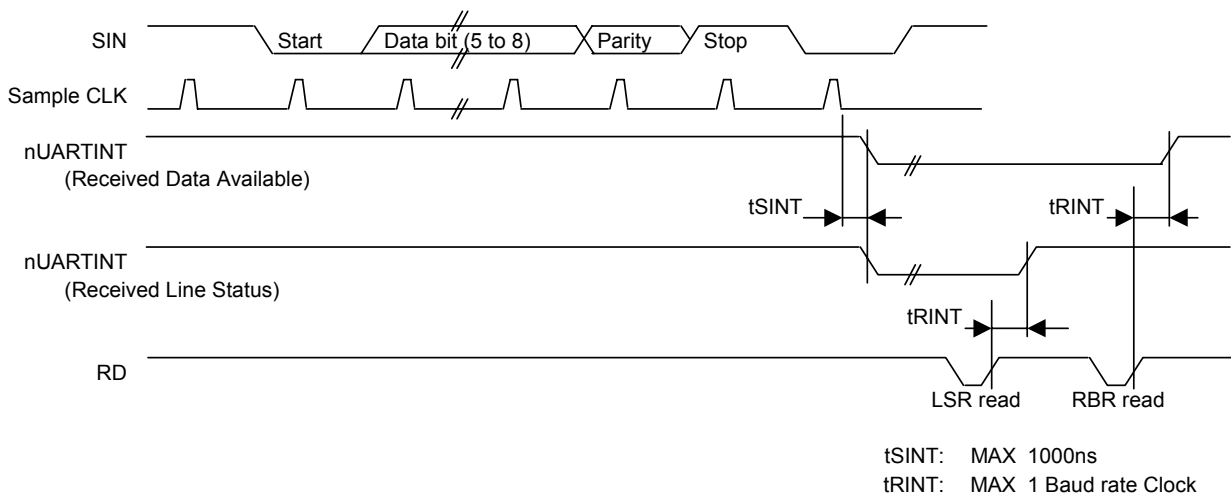
Figure 18.4 gives the timing for receiving data; Figure 18.5, that for reading the first byte from the receive queue; Figure 18.6, that for reading the last byte.

The baud rate is generated by dividing baud rate clock by 16. A sampling is performed by 8th clock of a baud rate clock.

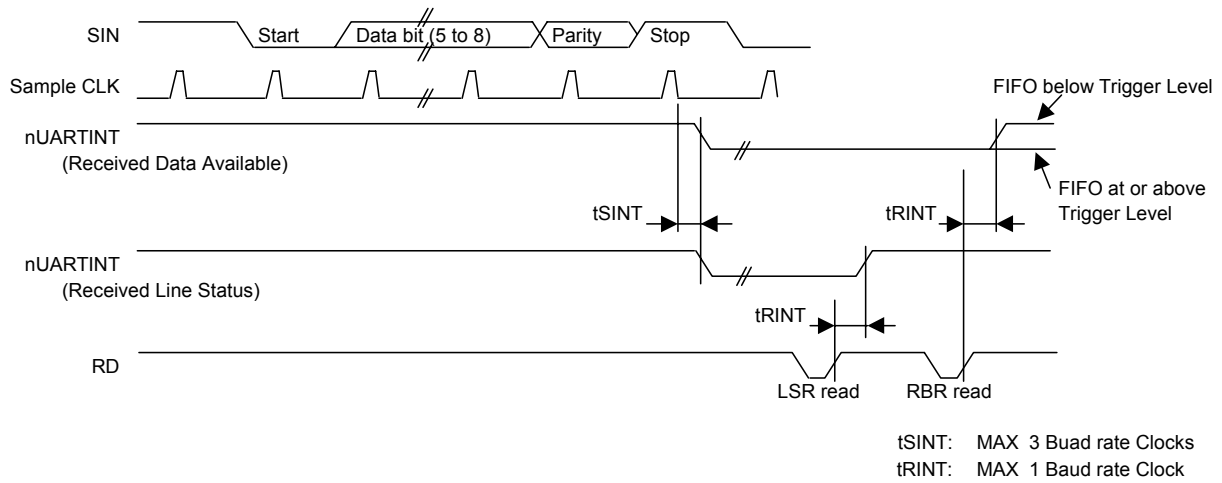
When the hardware detects the start bit from the SIN pin, it first latches the data bits into the receive shift register. It then transfers the received character from the shift register to the UARTRBR register via the receive queue. When the character reaches the UARTRBR register, the LSR[0] bit in the UARTLSR register goes to "1" to indicate that there is valid data in the UARTRBR register. Reading the UARTRBR register resets this bit to "0."



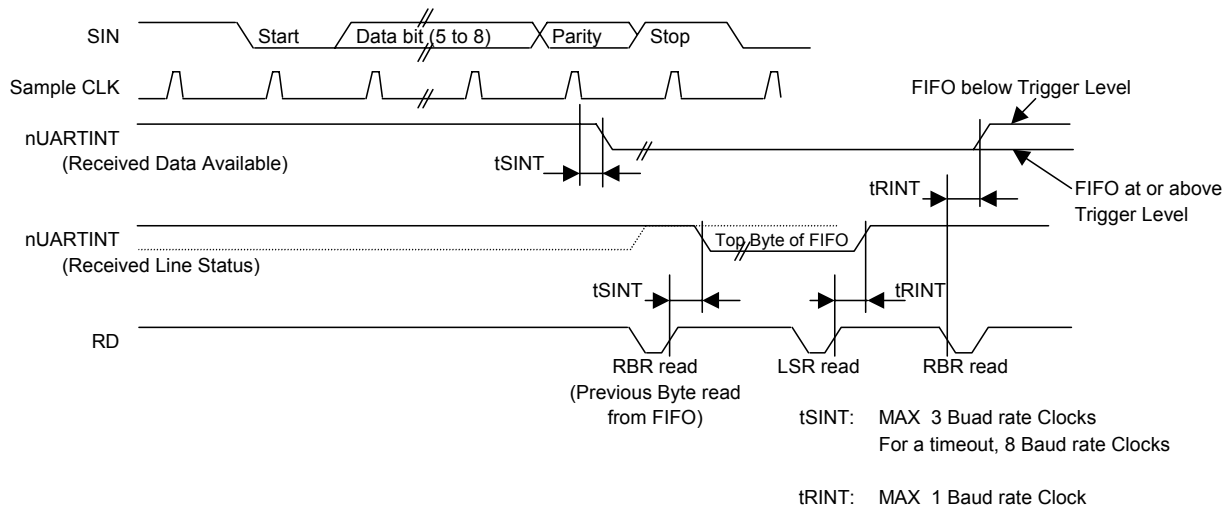
**Figure 18.3 Baud Rate and Sampling Clocks**



**Figure 18.4 Receive Timing**



**Figure 18.5 Reading First Byte from Receive Queue (Setting RDR)**



**Figure 18.6 Reading Last Byte from Receive Queue**

### 18.3.3 Generating Baud Rate Clock

The following is the formula for calculating the baud rate frequency.

$$\text{Baud rate frequency} = \text{CCLK}/(\text{DL}[15:0] \times 16)$$

Maximum baud rate clock depends on the software, the application, the system load and etc. But there is the capability of the transmit/receive baud rate with DL="2" setting in an ideal condition (ex. UART test) as long as in a small value of the baud rate deviation.

Note: A divisor (DL[15:0]) of 1 is not allowed. The setting must be either 0, to stop the clock, or a value of 2 or more.

The frequency of CCLK should be set up less than or equal to the frequency of HCLK.

The following Table lists DL settings for CPU clock and baud rate combinations.

Baud Rate	CCLK=60MHz *		CCLK = 33MHz		CCLK = 20MHz		CCLK = 8MHz	
	DL [H]	Deviation (%)	DL [H]	Deviation (%)	DL [H]	Deviation (%)	DL [H]	Deviation (%)
50	—	—	A122	0.00000	61A8	0.00000	2710	0.00000
75	C350	0.00000	6B6C	0.00000	411B	-0.00200	1A0B	-0.00500
110	852B	-0.00027	493E	0.00000	2C64	-0.00320	11C1	0.01000
134.5	6CE9	0.00015	3BE7	-0.00279	244E	-0.00344	0E85	0.01270
150	61A8	0.00000	35B6	0.00000	208D	0.00400	0D05	0.01000
300	30D4	0.00000	1ADB	0.00000	1047	-0.00800	0683	-0.02000
600	186A	0.00000	0D6E	-0.01454	0823	0.01600	0341	0.04002
1200	0C35	0.00000	06B7	-0.01454	0412	-0.03199	01A1	-0.07994
1800	0823	0.01600	047A	-0.01454	02B6	0.06404	0116	-0.07994
2000	0753	0.00000	0407	0.02425	0271	0.00000	00FA	0.00000
2400	061A	0.03200	035B	-0.04366	0209	-0.03199	00D0	0.16026
3600	0412	-0.03200	023D	-0.01454	015B	0.06404	008B	-0.07994
4800	030D	0.03200	01AE	-0.07267	0104	0.16026	0068	0.16026
7200	0209	-0.03200	011E	0.16026	00AE	-0.22340	0045	0.64412
9600	0187	-0.09600	00D7	-0.07267	0082	0.16026	0034	0.16026
19200	00C3	0.16000	006B	0.39428	0041	0.16026	001A	0.16026
38400	0062	-0.35200	0036	-0.53530	0021	-1.35732	000D	0.16026
56000	0043	-0.05333	0025	-0.45849	0016	1.46104	0009	-0.79365
115200	0021	-1.37600	0012	-0.53530	000B	-1.35732	—	—

Note:

\*As for the notation of 'CCLK=60MHz', only in the case of ML675001 series.



#### 18.3.4 Buffered Operation

##### Received Data Available Interrupts

Enabling both the receive queue and receive interrupts produces a received data available interrupt when the number of characters in the queue exceeds the specified trigger level. The hardware immediately clears this interrupt when the number of characters in the queue falls back to this trigger level.

The received data available bit in IIR is similar in operation. It goes to "1" when the number of characters in the queue exceeds the specified trigger level and returns to "0" when the number of characters in the queue falls back to this trigger level. It goes to "1" immediately after the hardware transfers the triggering data from the receive shift register to the queue and returns to "0" when the queue becomes empty.

Receiver line status interrupts have higher priority than these interrupts.

##### Character Timeout Interrupts

Enabling both the receive queue and receive interrupts produces a character timeout interrupt when the following conditions are met.

- There is at least one character in the receive queue.
- The time equivalent of at least four characters has elapsed since the last character was received (If the frame format specifies two stop bits, the timer starts after the first stop bit.) or since the last character was read off the queue.

If the frame format specifies one start bit, eight character bits, one parity bit, and two stop bits, for example, the timeout interval for a transfer speed of 300 baud is approximately 160 ms.

The clock used to calculate the character time is CCLK.

Reading a character from the queue clears the character timeout interrupt and resets the timeout detection timer.

If there is no character timeout interrupt pending, the hardware resets the timeout detection timer each time that the CPU reads a character from the queue or the interface receives a new character.

##### Transmitter Holding Register Empty Interrupts

Enabling both the transmit queue and receive queue interrupts produces a transmitter holding register empty interrupt when the transmit queue is empty. Writing a character to the transmit queue or reading IIR clears this interrupt.

This interrupt is delayed by the time equivalent of the frame less the final stop bit when the following conditions are met.

- There is only a single character in the queue after the transmitter holding register empty (THRE) bit goes to "1."
- The THRE bit goes to "1."

### 18.3.5 Queue Polled Mode

Enabling buffering, but disabling interrupts by setting the IER[3:0] bits all to “0” produces queue polled mode operation. The receive and transmit blocks have independent controls, so can use separate queue polled modes. Because there are no interrupts, the CPU must check the status of these two blocks by reading LSR.

- A “1” in LSR[0] indicates that there is at least one character in the receive queue.
- LSR[4:1] indicate any errors. Note that setting IER[2] to “0” means that such errors neither produce interrupts nor update the contents of IIR.
- A “1” in LSR[5] indicates that the transmit queue is empty.
- A “1” in LSR[6] indicates that the transmit queue and transmit shift register are both empty.
- A “1” in LSR[7] indicates an error receiving at least one character in the receive queue.

This mode uses the queue, but does not detect the trigger level or timeouts because those functions use interrupts, which are all disabled.

### 18.3.6 Error Status

#### Overrun error:

The LSR[1] bit in the UARTLSR register goes to "1" to indicate that the hardware has overwritten the contents of the UARTRBR register with a new character before the CPU read the former character.

#### (1) Parity error:

The LSR[2] bit in the UARTLSR register goes to "1" to indicate that the parity calculated from the received data does not match that received with the data. This only applies, however, when parity is enabled (LCR[3] = "1").

For buffered operation, LSR[2] indicates an error in the data at the head of the queue. Parity errors for other characters in the queue do not affect its contents.

#### (2) Framing error:

The LSR[3] bit in the UARTLSR register goes to "1" to indicate that the bit following the last data bit (or parity bit) is "0" (spacing level), not "1" (stop bit).

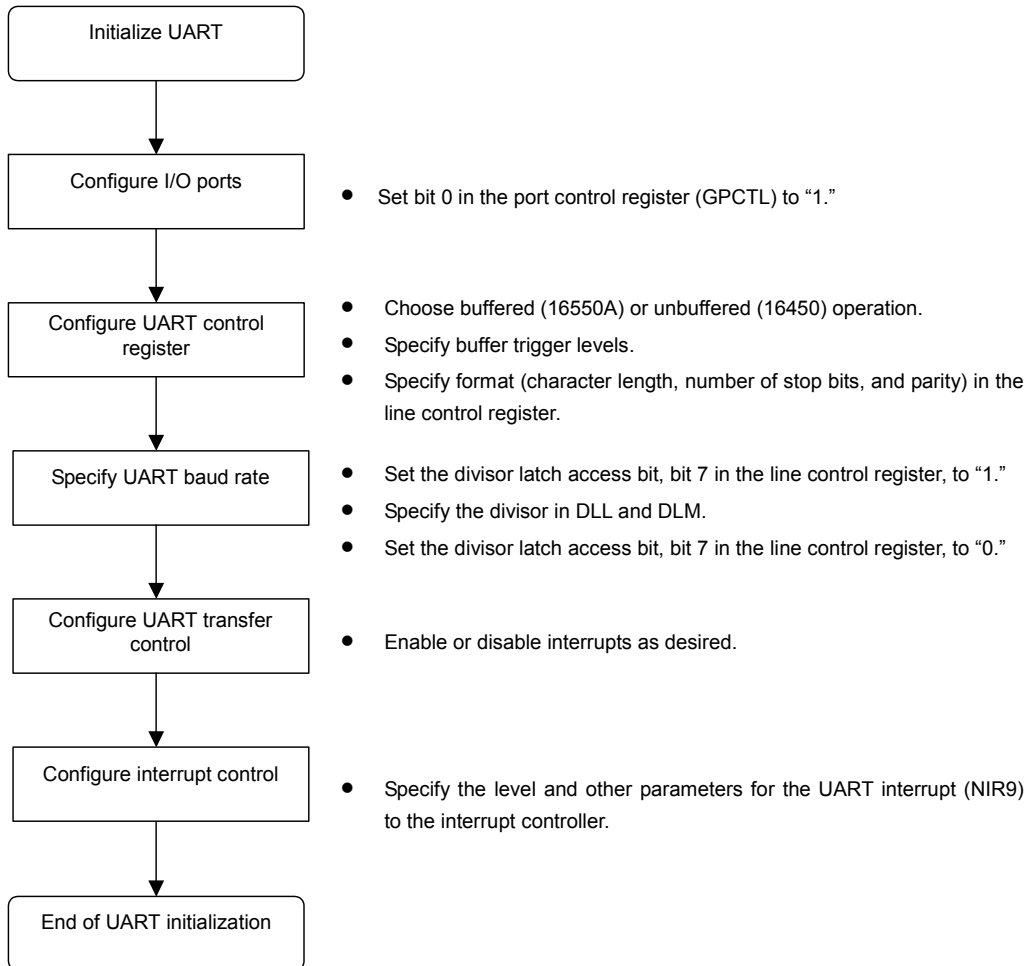
For buffered operation, LSR[3] goes to "1" when the character with the framing error reaches the head of the queue.

#### (3) Break interrupt:

The LSR[4] bit in the UARTLSR register goes to "1" to indicate that the input is "0" (spacing level) for one frame interval (start bit, data bits, parity bit, and stop bit).

For buffered operation, LSR[4] goes to "1" when the character with the break interrupt reaches the head of the queue.

### 18.3.7 Setup Procedure



## *Chapter 19*

# **Synchronous SIO**

---

## Chapter 19 Synchronous SIO

### 19.1 Overview

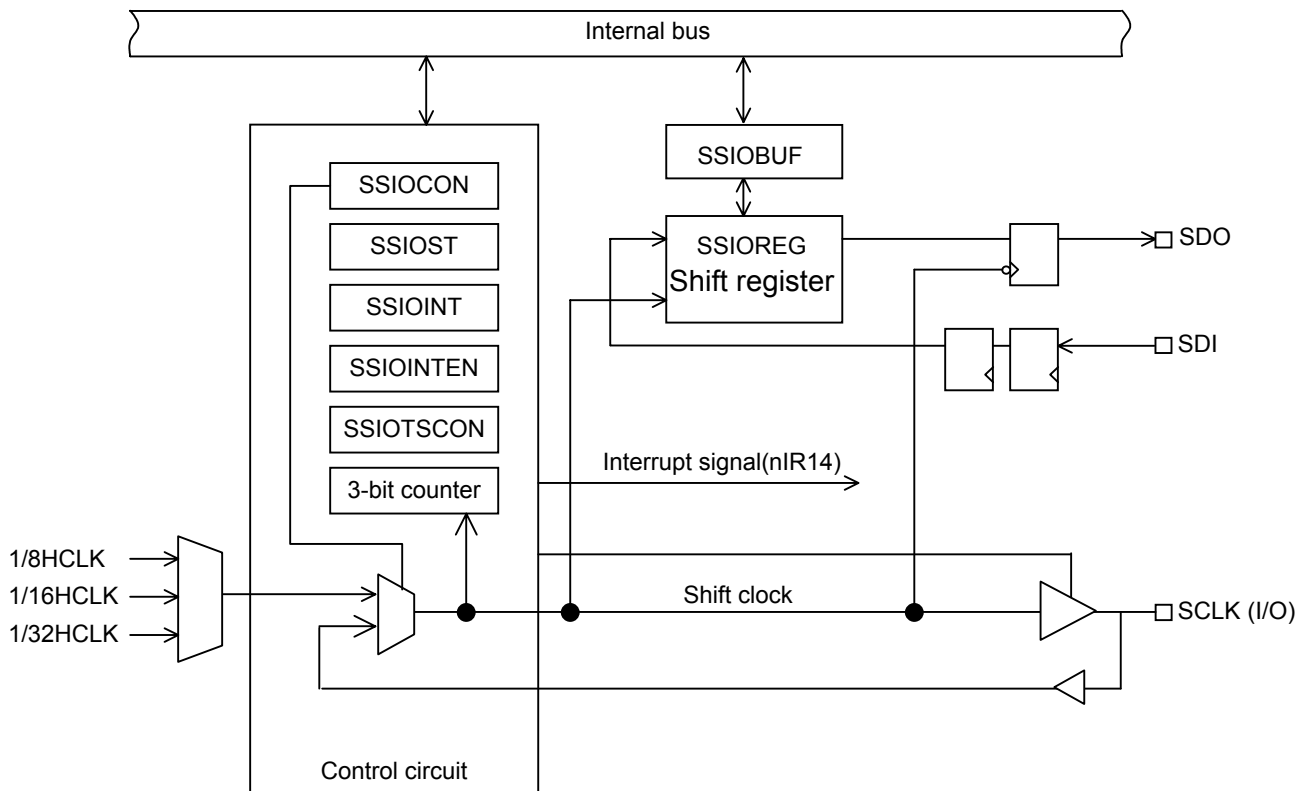
This LSI has one built-in 8-bit clock synchronous SIO channel.

#### Features

- Operating clock selectable at 1/8, 1/16, or 1/32 the frequency of the HCLK.
- Configurable as LSB first or MSB first.
- Configurable as Master or Slave.
- Generates transmit-receive complete interrupts and transmit-receive buffer empty interrupts.
- Built-in test function that loops-back the transmit output of the Synchronous SIO back to the receive input of it.

#### 19.1.1 Configuration

The configuration of the synchronous SIO is shown in Figure 19.1.



**Figure 19.1 Synchronous SIO Block Diagram**

SSIOCON	: Synchronous SIO control register
SSIOST	: Synchronous SIO status register
SSIOINT	: Synchronous SIO interrupt request register
SSIOINTEN	: Synchronous SIO interrupt enable register
SSIOBUF	: Synchronous SIO transmit-receive buffer register
SSIOREG	: Synchronous SIO transmit-receive shift register
SSIOCON	: Synchronous SIO test control register

## 19.1.2 List of Pins

Pin name	I/O	Function
SDI	I	Receive data input
SDO	O	Transmit data output
SCLK	I/O	Transmit-receive data synchronous clock input and output

## 19.1.3 List of Registers

Address	Name	Symbol	R/W	Size	Initial value
0xB7B0_1000	Synchronous SIO transmit-receive buffer register	SSIOBUF	R/W	8	0x00
0xB7B0_1004	Synchronous SIO status register	SSIOST	R/W	8	0x00
0xB7B0_1008	Synchronous SIO interrupt request register	SSIOINT	R/W	8	0x00
0xB7B0_100C	Synchronous SIO interrupt enable register	SSIOINTEN	R/W	8	0x00
0xB7B0_1010	Synchronous SIO transmit-receive control register	SSIOCON	R/W	8	0x00
0xB7B0_1014	Synchronous SIO test control register	SSIOTSCON	R/W	8	0x00

## 19.2 Registers

### 19.2.1 Synchronous SIO Transmit/Receive Buffer Register (SSIOBUF)

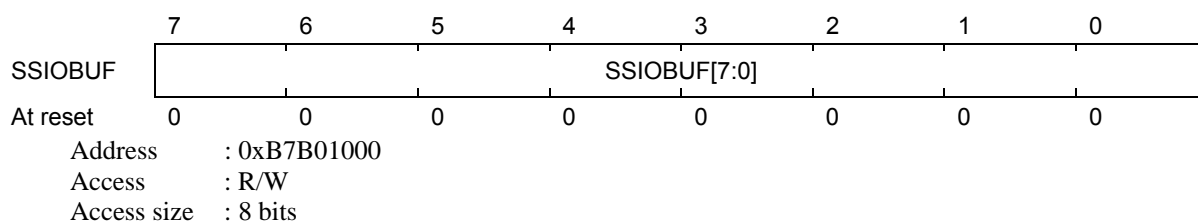
The SSIOBUF register holds transmit-receive data in transmit-receive operations.

The SSIOBUF register can be read/written using a program.

When writing, the register acts as a transmit buffer, and when reading the register acts as a receive buffer.

When receiving, the contents of SSIOBUF are held until the next receive operation is completed.

The SSIOREG register in the configuration diagram in Figure 19.1 is a shift register that converts parallel transmit data to serial data and converts serial receive data to parallel data. When transmit data is written to SSIOBUF in a transmit operation, the SSIOBUF data is automatically transferred to SSIOREG. In a receive operation, the SSIOREG data is transferred to SSIOBUF after the final bit has been received. The SSIOREG register cannot be read/written using a program.

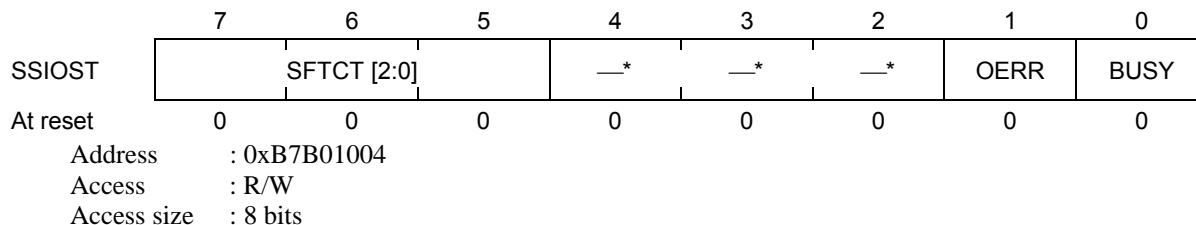




## 19.2.2 Synchronous SIO Status Register (SSIOST)

The SSIOST register indicates the operating state of the synchronous SIO.

OERR and BUSY can be read/written using a program. The SFTCT[2:0] bits of this register are read-only. Writes to these bits are invalid. When writing to SFTCT[2:0] bits, write "0".



## [NOTE]

"—\*" denotes a reserved bit. Always write "0" to the bit.

If "1" is written, normal operation is not guaranteed.

## [Explanation of Bits]

## • BUSY (bit 0)

This bit indicates that data is being transmitted or received. This bit is automatically set to "1" when data transmission or reception begins and is cleared to "0" when the transmission or reception is completed. When this bit is "1", it means that data is being transferred.

By writing "0" to the BUSY bit during data transfer, it is possible to abort the transmit or receive operation and initialize the synchronous SIO.

If "1" is written during transmit and receive idle, the write operation is invalid.

BUSY	Description
0	Transmit-receive idle
1	Transmit-receive in progress

## • OERR (bit 1)

This bit indicates the presence or absence of an overrun error. If the previously received data has not been read by the CPU, the bit is set to "1".

Once set to "1", the bit is not cleared to "0" even if there have been no overrun errors at the end of the next receive operation. Therefore it is necessary to clear the bit to "0" with a program. Writing "0" to this bit will clear to "0", but writing "1" will set to "1".

OERR	Description
0	No overrun error
1	Overrun error

## • SFTCT[2:0] (bit 7 to bit 5)

These bits indicate the count value of the 3-bit shift counter when transmitting or receiving data. While data is not being transmitted or received, the counter indicates '000'(binary). Each time the transmit-receive data is shifted by one bit, the counter increments by 1. The counter returns to '000' (binary) when the transmit or receive operation is completed or when "0" is written to the BUSY bit during reception.

### 19.2.3 Synchronous SIO Interrupt Request Register (SSIOINT)

The SSIOINT register indicates a synchronous SIO interrupt request.  
 The SSIOINT register can be read or written using a program. Writing "1" to a bit clears that bit.

	7	6	5	4	3	2	1	0
SSIOINT	—	—	—	—	—	TREMP	RXCMP	TXCMP
At reset	0	0	0	0	0	0	0	0
Address	: 0xB7B01008							
Access	: R/W							
Access size	: 8 bits							

**[NOTE]**

"—" denotes a reserved bit. Always write "0" to the bit.  
 If "1" is written, normal operation is not guaranteed.

**[Explanation of Bits]**

- TXCMP (bit 0)  
 This bit indicates that data transmission is complete

TXCMP	Description
0	Transmission not complete
1	Transmission complete

- RXCMP (bit 1)  
 This bit indicates that data reception is complete

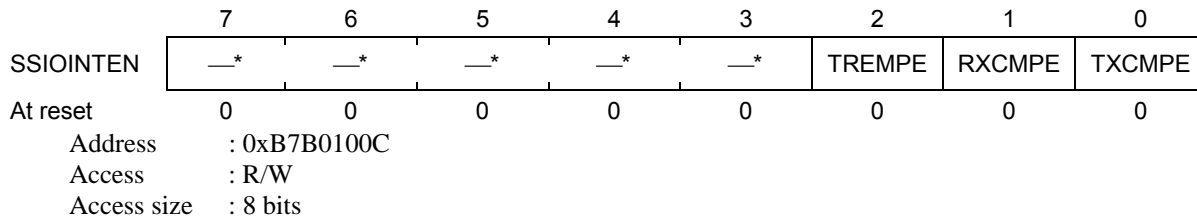
RXCMP	Description
0	Reception not complete
1	Reception complete

- TREMP (bit 2)  
 This bit indicates that transmit data has been transferred from the transmit-receive buffer register to the shift register and the transmit-receive buffer register is now empty.

TREMP	Description
0	Transmit data not transferred
1	Transmit data transferred

### 19.2.4 Synchronous SIO Interrupt Enable Register (SSIOINTEN)

SSIOINTEN is the register that enables the synchronous SIO interrupt cause.  
 The SSIOINTEN register can be read or written using a program.



**[NOTE]**

"—\*" denotes a reserved bit. Always write "0" to the bit.  
 If "1" is written, normal operation is not guaranteed.

**[Explanation of Bits]**

- TXCMPEN (bit 0)  
 This bit enables transmit complete interrupt requests

TXCMPEN	Description
0	Transmit complete interrupt request masked
1	Transmit complete interrupt request enabled

- RXCMPEN (bit 1)  
 This bit enables receive complete interrupt requests.

RXCMPEN	Description
0	Receive complete interrupt request masked
1	Receive complete interrupt request enabled

- TREMPEN (bit 2)  
 This bit enables transmit-receive buffer empty interrupt requests.

TREMPEN	Description
0	Transmit-receive buffer empty interrupt request masked
1	Transmit-receive buffer empty interrupt request enabled

### 19.2.5 Synchronous SIO Control Register (SSIOCON)

SSIOCON is the register that controls transmit-receive operations.

The SSIOCON register can be read or written using a program. If the SSIOCON register is to be modified, make those changes after transmission or reception is completed. If the SSIOCON register is modified during transmission or reception, the current transmission or reception will not be performed correctly.

	7	6	5	4	3	2	1	0
SSIOCON	—*	—*	SLMSB	SFTSLV	—*	—*	SFTCLK [1:0]	
At reset	0	0	0	0	0	0	0	0
Address	: 0xB7B01010							
Access	: R/W							
Access size	: 8 bits							

[NOTE]

"—\*" denotes a reserved bit. Always write "0" to the bit.  
If "1" is written, normal operation is not guaranteed.

[Explanation of Bits]

- SFTCLK [1:0] (bit 1 to bit 0)  
These bits select the synchronous clock in the master mode and are disabled in the slave mode.

SFTCLK [1:0]		Description
0	0	1/8 HCLK
0	1	1/16 HCLK
1	X	1/32 HCLK

\*See Chapter 7, Power Consumption Control, for HCLK.

- SFTSLV (bit 4)  
This bit selects the master or the slave mode.

SFTSLV	Description
0	Master mode
1	Slave mode

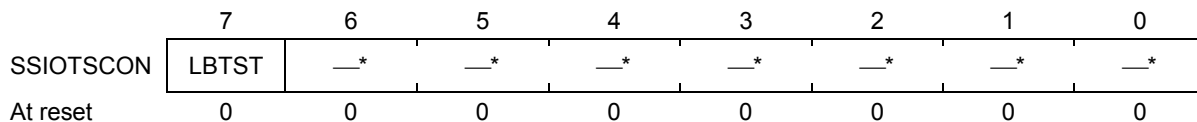
- SLMSB (bit 5)  
This bit selects LSB first or MSB first for the transmit-receive data.

SLMSB	Description
0	LSB first
1	MSB first

### 19.2.6 Synchronous SIO Test Control Register (SSIOTSCON)

SSIOTSCON is a register designed to simplify the synchronous SIO internal tests.

The SSIOTSCON register can be read or written using a program. In normal operation, set SSIOTSCON to 0x00. Do not rewrite the content of this register while transmitting or receiving data.



Address : 0xB7B01014

Access : R/W

Access size : 8 bits

**[NOTE]**

"—\*" denotes a reserved bit. Always write "0" to the bit.

If "1" is written, normal operation is not guaranteed.

**[Explanation of Bits]**

- LBTST (bit 7)

This bit enables the loop back test function. When the loop back test function is enabled, transmit signals are returned as receive signals.

LBTST	Description
0	Loop back test function disabled (normal mode)
1	Loop back test function enabled (test mode)

## 19.3 Operations

### 19.3.1 Master Mode/Slave Mode

There are two modes for transmission and reception, the master mode and the slave mode. The mode to be used for transmission and reception is selected by the SFTSLV bit of the synchronous SIO control register (SSIOCON).

In the master mode, a synchronous clock with 1/8, 1/16, or 1/32 the frequency of HCLK is output from the SCLK pin and data is transmitted and received in synchronization with that clock. The synchronous clock is selected by the SFTCLK[1:0] bits of the synchronous SIO control register (SSIOCON).

In the slave mode, an external synchronous clock is input from the SCLK pin and data is transmitted and received in synchronization with that clock. When operating in slave mode, the synchronous input clock, SCLK, must have a frequency equal to or less than 1/8 the frequency of HCLK.\*<sup>1</sup>

The communication functionality when operating in slave mode or master mode is nearly the same. The only difference is that during master mode operation, the SSIO outputs its clock from the SCLK pin. During slave mode operation, the SCLK pin works as an input for the clock signal coming from an external synchronous I/O.

#### [NOTE]

- \*1 The transmit data output from the SDO pin in the slave mode is output for synchronization after two cycles of HCLK from the falling edge of the synchronous clock input from the SCLK pin. The setup of data receiving on the master side is time-critical by two cycles of HCLK.

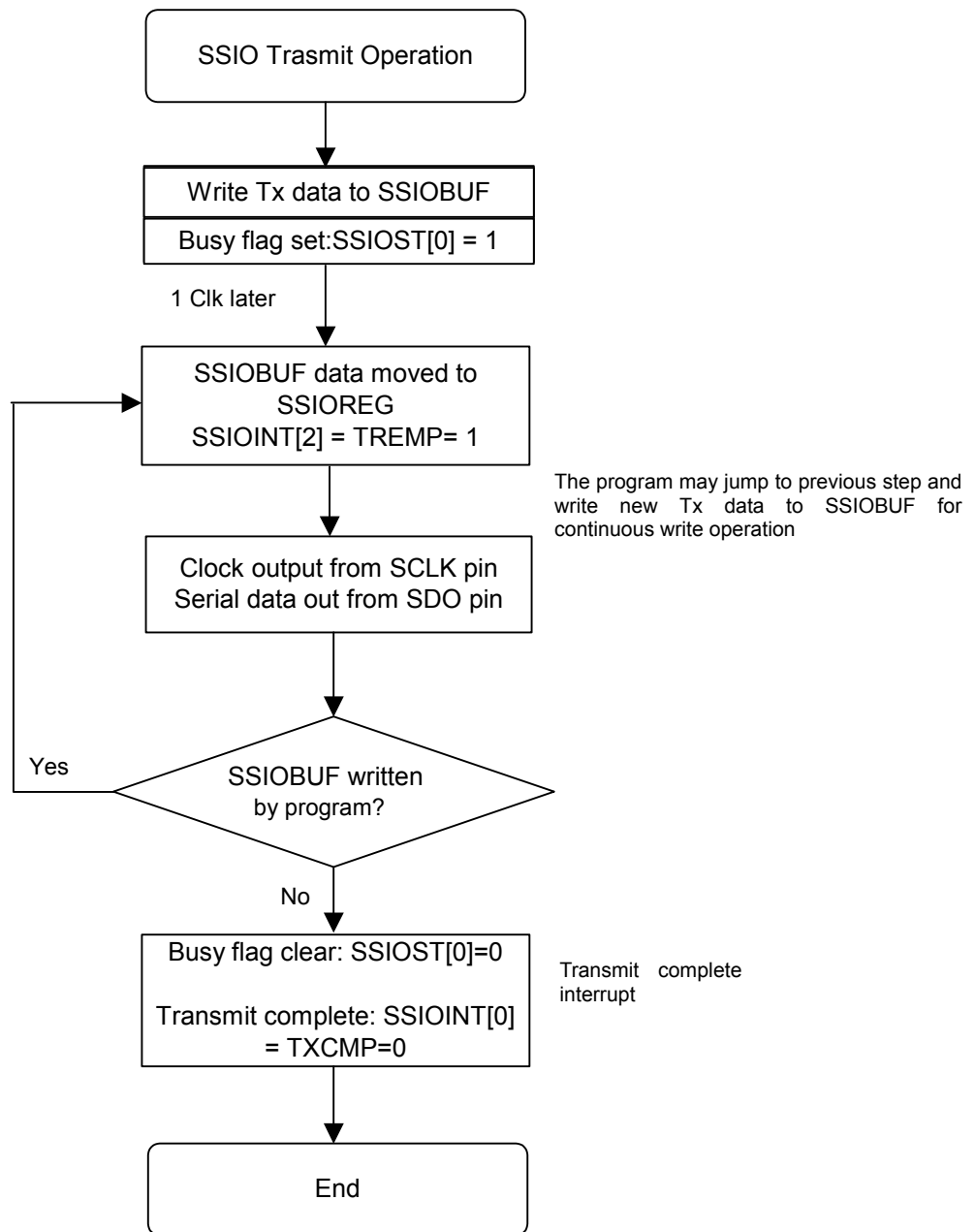
### 19.3.2 Transmit Operation

- ① Writing of transmit data to the synchronous SIO transmit-receive buffer register (SSIOBUF) triggers data transmission and the synchronous SIO status register (SSIOST) BUSY flag is set to "1".
- ② One clock (HCLK) after the transmit data is written, the transmit data is transferred from SSIOBUF to the synchronous SIO transmit-receive shift register (SSIOREG). At the same time, the TREMP bit of the synchronous SIO interrupt request register (SSIOINT) is set to "1", allowing the next transmit data to be written.
- ③ The synchronous clock is then output from the SCLK pin (only if Synchronous SIO is in master mode), and the transmit data is output from the SDO pin, either as LSB first or MSB first, in synchronization with the falling edge of the synchronous clock.
- ④ After that, transmit data is output in synchronization with the synchronous clock according to the specification of the synchronous clock SIO control register (SSIOCON), and one frame of data is transmitted. If there is no new transmit data written to the transmit buffer register, the BUSY flag is cleared to "0" and at the same time the TXCMP bit of the SSIOINT register is set to "1" to complete the transmit operation.

#### Continuous transmit operation

Steps ① to ③ of Section 19.3.2 are performed. If the next transmit data is written to SSIOBUF from the time TREMP = 1 is set in ② to when the transmit operation is completed, when the current transmit operation is completed, the next transmit data is automatically transferred to SSIOREG and data is transmitted continuously. Transmission is completed with step ④ in Section 19.3.2. If the next data has been written to the transmit buffer when one frame of data has been transmitted, no transmit complete interrupt is generated.

Transmit operation flow



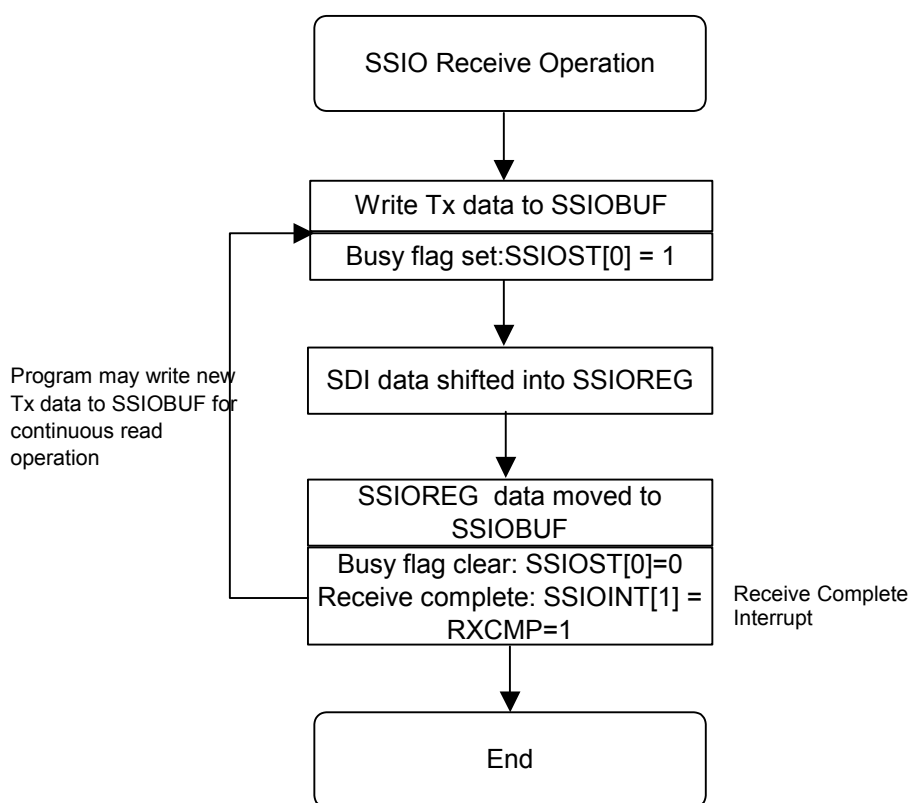
Notes:

In the above flow diagram, all tasks are implemented automatically in hardware except for writes to the SSI0BUF. Write operations to the SSI0BUF are done by the application program. If the Synchronous SIO is working in slave mode, the synchronous clock signal will be an input to the SCLK pin and will be monitored by Synchronous SIO for synchronous operation.

### 19.3.3 Receive Operation

- ① Writing of dummy data to the synchronous SIO transmit-receive buffer register (SSIOBUF) initiates the receive operation. The BUSY flag of the synchronous SIO status register (SSIOST) is set to "1" when data is written to SSIOBUF.
- ② Data input from the SDI pin is shifted into the transmit-receive shift register on the rising edge of the synchronous clock.
- ③ After that, data reception continues, and when one frame of data has been received, the content of SSIOREG are transferred to SSIOBUF. At the same time the BUSY flag is cleared to "0" and the receive complete flag (RXCMP) of the synchronous SIO interrupt request register (SSIOINT) is set to "1".
- ④ If data is received before the previously received data is read out of the transmit-receive buffer register (SSIOBUF), an overrun error will occur (the previously received data will be overwritten)

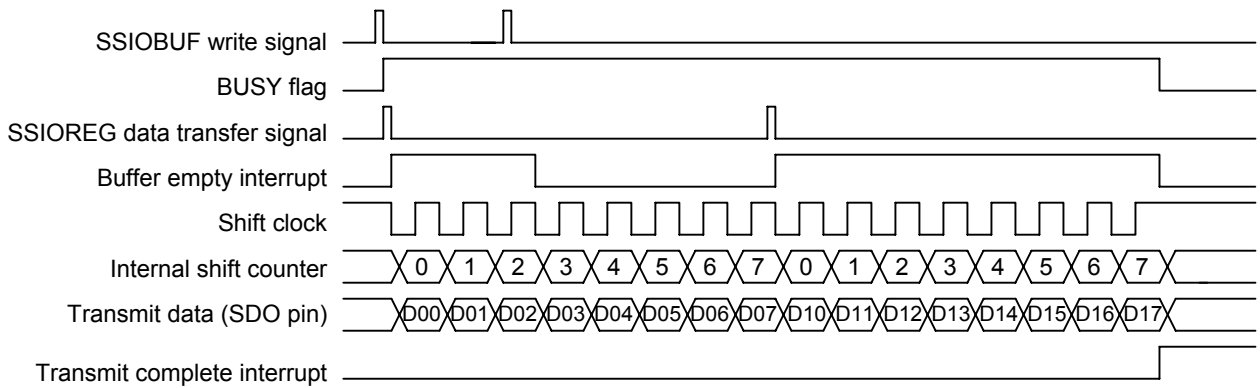
Receive operation flow



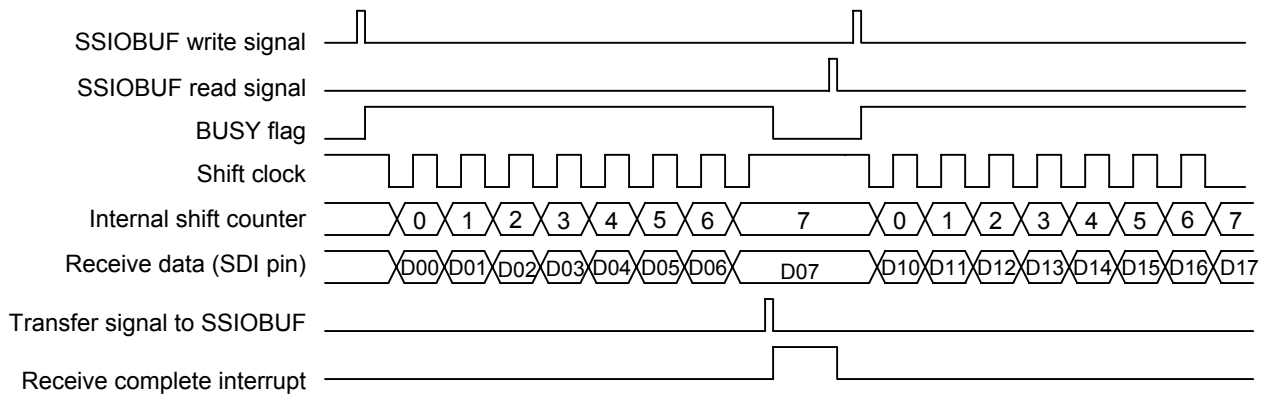
Note:

In the above flow diagram, all tasks are implemented automatically in hardware. Once a frame of data has been received and written to SSIOBUF, the application program should read this data from SSIOBUF.





**Figure 19.2 Transmit Operation Timing Diagram (LSB First)**



**Figure 19.3 Receive Operation Timing Diagram (LSB First)**

19.3.4 Interrupt Signal

Figure 19.4 shows the logic of the interrupt signal. The interrupt causes (bit 2 to bit 0 of the SSIOINT register) are masked or enabled by bit 2 to bit 0 of the SSIOINTEN register, respectively and the outputs (bit 2 to bit 0) are ORed. The logically ORed output is the synchronous SIO interrupt signal.

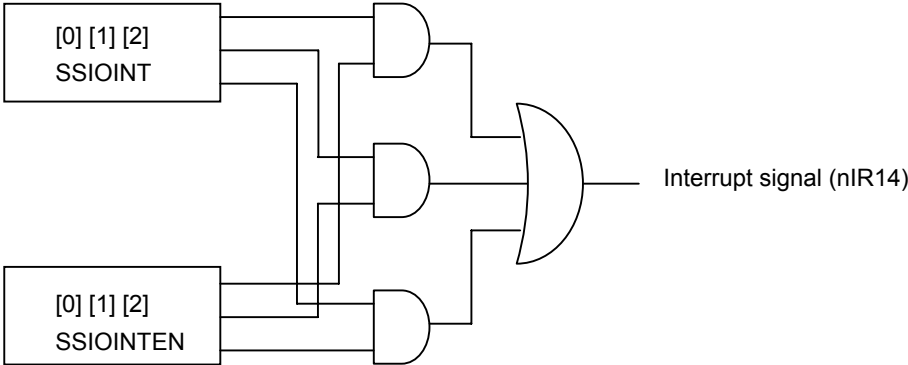


Figure 19.4 Logic of Interrupt Signals

## *Chapter 20*

**I2C**

---

## Chapter 20 I2C

### 20.1 Overview

This LSI has one channel of built-in I2C bus interface that conforms to the standard I2C bus specifications. The I2C block is designed to work as a single master.

#### Features

Communication mode: Master transmitter/master receiver (no slave function)  
Communication speed: 100 Kbps (Standard mode)/400 Kbps (Fast mode)  
Addressing format: 7 bits/10bits

#### 20.1.1 Configuration

Figure 20.1 shows the configuration of the I2C bus interface.

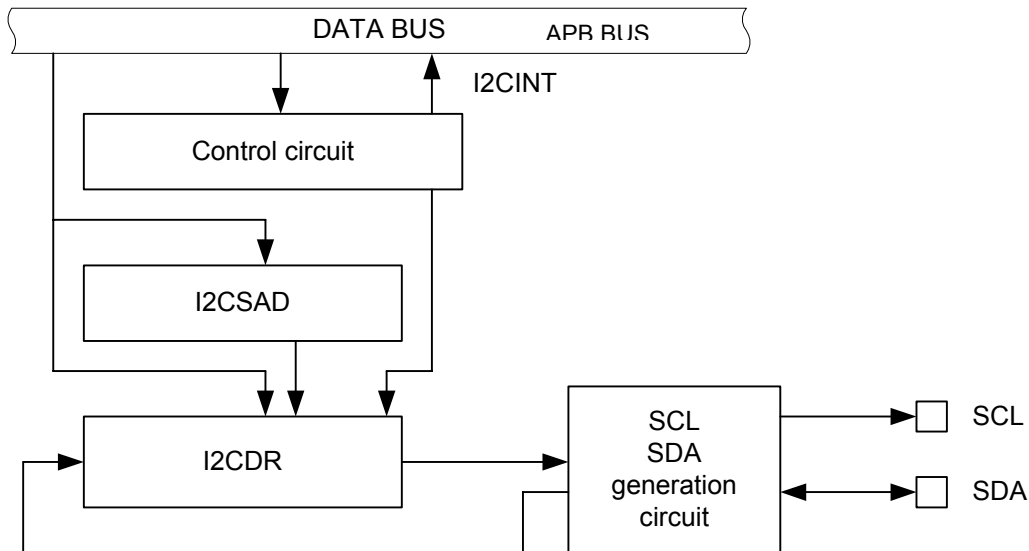


Figure 20.1 Configuration of I2C Bus Interface

### 20.1.2 List of Pins

Pin name	I/O	Function
SDA	I/O	Serial data Input-output pin. This is a secondary function of PIOE[3].
SCL	O	Serial data Transfer clock. This is a secondary function of PIOE[4].

This SDA and SCL terminal carry out logic operation of NMOS open drain output at the time of secondary functional operation. For the reason, please attach external Pull-Up resistor for an I2C function at the time of use.

### 20.1.3 List of Registers

Address	Name	Symbol	R/W	Size	Initial value
0xB780_0000	I2C bus control register	I2CCON	R/W	8	0x00
0xB780_0004	I2C bus slave address mode register	I2CSAD	R/W	8	0x00
0xB780_0008	I2C bus transfer speed register	I2CCLR	R/W	8	0x00
0xB780_000C	I2C bus status register	I2CSR	R	8	0x00
0xB780_0010	I2C bus interrupt request register	I2CIR	R/W	8	0x00
0xB780_0014	I2C bus interrupt mask register	I2CIMR	R/W	8	0x01
0xB780_0018	I2C bus transmit/receive data register	I2CDR	R/W	8	0x00
0xB780_001C	I2C bus transfer speed counter	I2CBC	R/W	8	0x00

## 20.2 Registers

### 20.2.1 I2C Bus Control Register (I2CCON)

The I2CCON register controls the transmission and reception of the I2C bus. The data in this register can be read and written by programs.

The value at the time of reset is 0x00.

	7	6	5	4	3	2	1	0
I2CCON	-*	-*	-*	START	RESTR	STCM	I2COC	I2CEN
At reset	0	0	0	0	0	0	0	0

Address : 0xB7800000

Access : R/W

Access size : 8 bits

[Note]

-\*: This is a reserved bit for future expansion. In this LSI, "0" is read during reading, and this bit is ignored during writing.

[Explanation of Bits]

- I2CEN (bit 0)  
This bit designates the transmission of the restart sequence. The I2CEN bit is not cleared automatically. Writing "0" to this bit resets it to "0".

I2CEN	Description
0	No restart sequence transmission.
1	Restart sequence transmission.

- I2COC (bit 1)  
This bit designates the transmission of the stop sequence after data is transmitted or received. The I2COC bit is not cleared automatically.

I2COC	Description
0	Stop sequence transmission.
1	No stop sequence transmission.

- STCM (bit 2)  
This bit designates the transmission/reception of 1-byte data. The transmission/reception of 1-byte data includes the transmission of the slave address, the transmission/reception of data, and the transmission/reception of an acknowledge or negative acknowledge. This bit is automatically cleared to "0" after data is transmitted or received.

STCM	Description
0	No data transmission/reception.
1	Data transmission/reception.

- RESTR (bit 3)  
This bit controls the transmission of an acknowledge signal for the data to be received next.

RESTR	Description
0	Acknowledge signal transmission.
1	No acknowledge signal transmission.

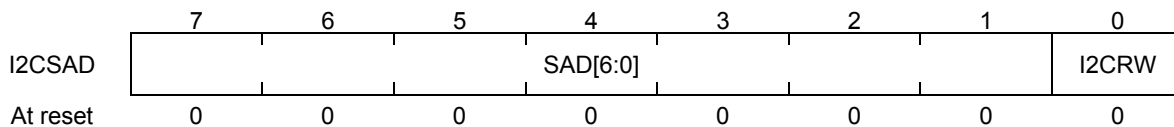
- START (bit 4)

This bit indicates the presence or absence of a start byte in the I2CSAD register. To set the START bit to "1," set the I2CSAD slave address mode setting register to "0000\_0001b"(the start byte pattern defined by the I2C specification) in advance.

START	Description
0	No START byte.
1	START byte.

### 20.2.2 I2C Bus Slave Address Mode Register (I2CSAD)

The I2CSAD register sets the address of the slave device as well as the data transfer direction (transmission/reception). The data in this register can be read and written by programs. The value at the time of reset is 0x00. Set the I2CSAD register before setting the I2CCON register. Specify the address of the following a general call address to the I2CDR register.



Address : 0xB7800004  
Access : R/W  
Access size : 8 bits

[Explanation of Bits]

- I2 CRW (bit 0)  
The I2CRW bit designates the communication direction (transmission/reception) to slave device.

I2CRW	Description
0	Transmission
1	Reception

- SAD[6:0] (bits 7 to 1)  
These bits specify the slave address of the communication destination.



### 20.2.3 I2C Bus Transfer Speed Register (I2CCLR)

The I2CCLR register sets the communication speed (mode) of the I2C bus. The data in this register can be read and written by programs. The value at the time of reset is 0x00. Always set the I2CCLR register before setting the I2CCON register.

	7	6	5	4	3	2	1	0
I2CCLR	-*	-*	-*	-*	-*	-*	-*	I2CMD
At reset	0	0	0	0	0	0	0	0
Address	: 0xB7800008							
Access	: R/W							
Access size	: 8 bits							

[Note]

-\*: This is a reserved bit for future expansion. In this LSI, "0" is read during reading, and this bit is ignored during writing.

[Explanation of Bits]

- I2CMD (bits 0)  
 This bit selects Standard-mode or Fast-mode.

I2CMD	Description
0	Selects Standard -mode (100 kHz).
1	Selects Fast -mode (400 kHz).

### 20.2.4 I2C Bus Status Register (I2CSR)

The I2CSR register indicates the status of the I2C bus. The data in this register can only be read by a program. The value at the time of reset is 0x00.

	7	6	5	4	3	2	1	0
I2CSR	-*	-*	-*	-*	-*	-*	I2CAAK	I2CDAK
At reset	0	0	0	0	0	0	0	0
Address	: 0xB780000C							
Access	: R							
Access size	: 8 bits							

[Note]

-\*: This is a reserved bit for future expansion. In this LSI, "0" is read during reading, and this bit is ignored during writing.

[Explanation of Bits]

- I2CDAK (bit 0)

This bit indicates the presence or absence of an acknowledge from the slave device for the data transmitted. Once this bit is set to "1", it is not automatically cleared to "0" even if an acknowledge is received normally upon completion of the next transfer. Therefore, it is necessary to clear it to "0" by a program. This bit is automatically cleared to "0" by writing "1" to the I2CIR bit of the I2CIR register.

I2CDAK	Description
0	Normal reception of acknowledge for transmit data.
1	Acknowledge error for transmit data.

- I2CAAK (bit 1)

This bit indicates the presence or absence of an acknowledge from the slave device in response to the broadcasting of its address over the bus. Once this bit is set to "1," it is not automatically cleared to "0" even after an acknowledge is received normally upon completion of the next transfer. Therefore, it is necessary to clear it to "0" by a program. This bit is automatically cleared to "0" by writing "1" to the I2CIR bit of the I2CIR register.

I2CAAK	Description
0	Normal reception of acknowledge for slave address.
1	Receive error of acknowledge for slave address.

## 20.2.5 I2C Bus Interrupt Request Register (I2CIR)

The I2CIR register indicates the status of interrupts of the I2C block. The data in this register can be read and written by programs. The value at the time of reset is 0x00.

	7	6	5	4	3	2	1	0
I2CIR	-*	-*	-*	-*	-*	-*	-*	I2CIR
At reset	0	0	0	0	0	0	0	0
Address	: 0xB7800010							
Access	: R/W							
Access size	: 8 bits							

## [Note]

-\*: This is a reserved bit for future expansion. In this LSI, "0" is read during reading, and this bit is ignored during writing.

## [Explanation of Bits]

- I2CIR (bit 0)

This bit indicates the presence of an interrupt request. When "1" is written to this I2CIR bit, the I2CAAK and I2CDAK bits of the I2CSR register are cleared. When the I2CDAK bit is set to "1," this bit is immediately set to "1." However, when the I2CAAK bit is set to "1," this bit is set to "1" after the transmission/reception of 1-byte of data following the transmission of slave address.

I2CIR	Description
0	No interrupt request.
1	Interrupt request pending.

### 20.2.6 I2C Bus Interrupt Mask Register (I2CIMR)

The I2CIMR register masks the interrupt cause of the I2C bus. The data in this register can be read and written by programs. The value at the time of reset is 0x01. Set the I2CIMR register before setting the I2CCON register.

	7	6	5	4	3	2	1	0
I2CIMR	-*	-*	-*	-*	-*	-*	-*	I2CMF
At reset	0	0	0	0	0	0	0	1
Address	: 0xB7800014							
Access	: R/W							
Access size	: 8 bits							

[Note]

-\*: This is a reserved bit for future expansion. In this LSI, "0" is read during reading, and this bit is ignored during writing.

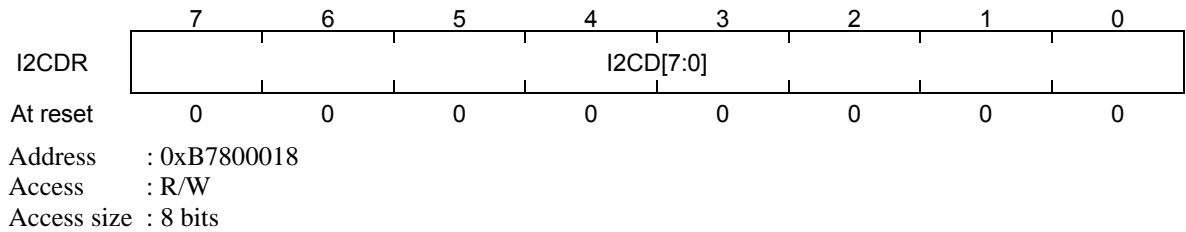
[Explanation of Bits]

- I2CMF (bit 0)  
 This bit masks the I2C interrupt request.

I2CMF	Description
0	Interrupt enabled.
1	Interrupt masked.

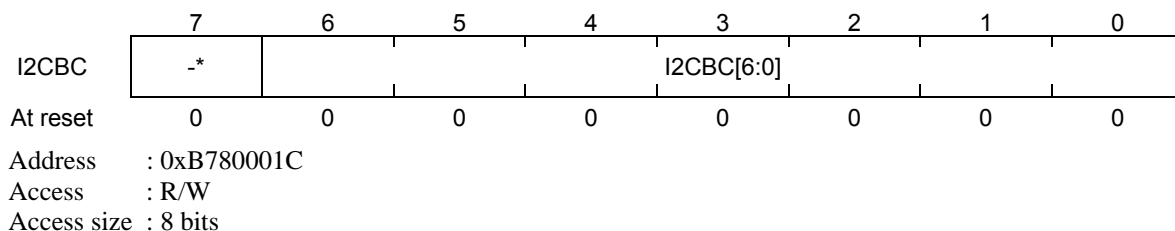
### 20.2.7 I2C Bus Transmit/Receive Data Register (I2CDR)

The I2CDR register sets transmit data or stores received data. The data in this register can be read and written by programs. The value at the time of reset is 0x00. When reading received data, read it before transmitting or receiving the next data. Set or read the I2CDR register before setting the I2CCON register. Read the I2CDR register following completion of byte data transfer after setting the I2CCON register.



### 20.2.8 I2C Bus Transfer Speed Counter (I2CBC)

The I2CBC register sets counter values in the counter that generates the transfer timing from HCLK to the I2C bus. The data in this register can be read and written by programs. The value at the time of reset is 0x00.



[Note]

-\*: This is a reserved bit for future expansion. In this LSI, "0" is read during reading, and this bit is ignored during writing.

The bit 6 of an I2CBC register does not exist in ML674001 series. In ML674001 series, I2CBC (bits 6) becomes the treatment of "-\*".

- I2CBC (bits 6 to 0)

The relationships between the setting values of the I2CBC register and the transfer speed of the I2C bus are as follows.

$$\text{I2C bus transfer speed (Hz)} = \text{HCLK frequency} / (\text{I2CBC setting value} \times 8)$$

Thus

$$\text{I2CBC} = \text{HCLK frequency} / (\text{I2C bus transfer speed (bps)} \times 8)$$

The following table shows some examples for setting I2CBC register values.

HCLKfrequency	I2CBC <sup>1</sup>	I2CBC <sup>2</sup>
60 MHz *1	75 (0x4B)	19 (0x13)
33 MHz	42 (0x2A)	11 (0x0B)
25 MHz	32 (0x20)	08 (0x08)
20 MHz	25 (0x19)	07 (0x07)

1 For I2C bus operating at 100 Kbps. Thus I2CCLR = 0x00

2 For I2C bus operating at 400 Kbps. Thus I2CCLR = 0x01

Note: When the I2CBC register is set to "0", the timing generating counter stops.

[Note]

\*: Set the above I2CBC register before setting both the I2CCLR and I2CCON registers.

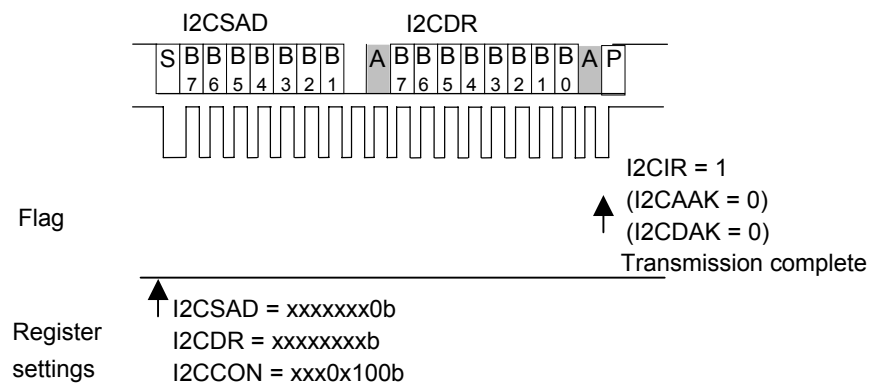
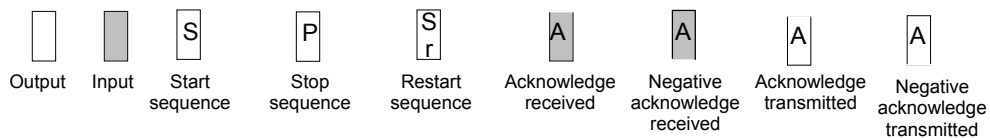
\*1: This setting condition is applied only to ML675001 series.

## 20.3 Operations

The following is a function description of the I2C block operation. Through out this section, I2C bus transfer illustrations are used as visual aides. Use the following legend for all of the illustrations in this section.

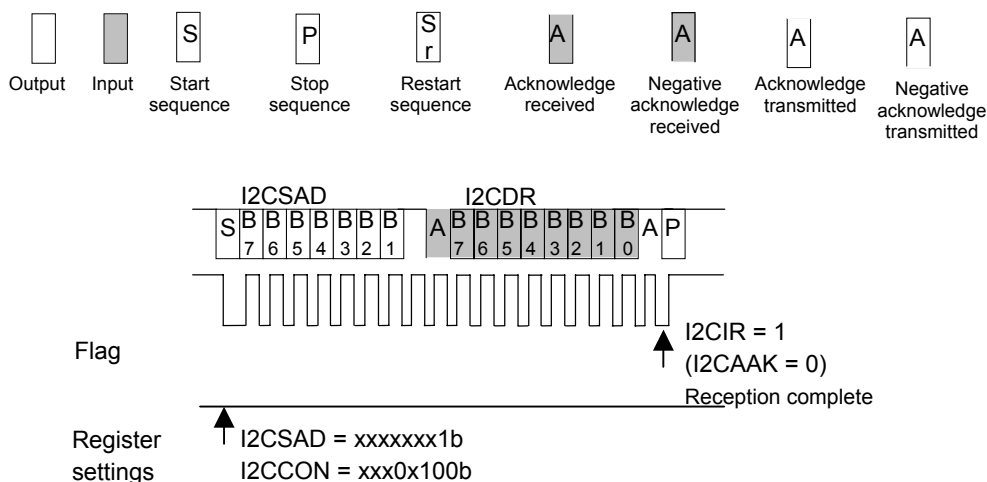
### 20.3.1 Transmit operation (transfer of 1 byte from master to slave, in 7-bit address mode)

- ① To identify the communication destination, the address of the communication destination (slave) is written to SAD[6:0] of the I2CSAD register, and to set the transfer direction as transmission, the I2CRW bit is set to "0" (transmitted by the master device).
- ② Write transmit data to the I2CDR register.
- ③ The I2CCON register is set to XXX0X100. The following series of operations is automatically performed when the STCM bit (bit 2 of the I2CCON register) is set to "1": Transmission of the start sequence (to attain bus access), transmission of the slave address and the transfer direction specified to the I2CSAD register, confirmation of an acknowledge from the slave device for the address transmitted, transmission of the transmit data written to the I2CDR register, and confirmation of an acknowledge from the slave device for the data transmitted. Also, when the I2COC bit (bit 1 of the I2CCON register) is set to "0", the stop sequence is successively transmitted (the bus is released) and communication is then finished. At this point, the I2CIR bit is set to "1," indicating that the transmission of 1-byte of data has been finished. If acknowledges for the address and data transmitted have not been returned normally, both the I2CAAK and I2CDAK bits are set to "1" upon completion of transmission.



20.3.2 Receive operation (transfer of 1 byte from slave to master, in 7-bit address mode)

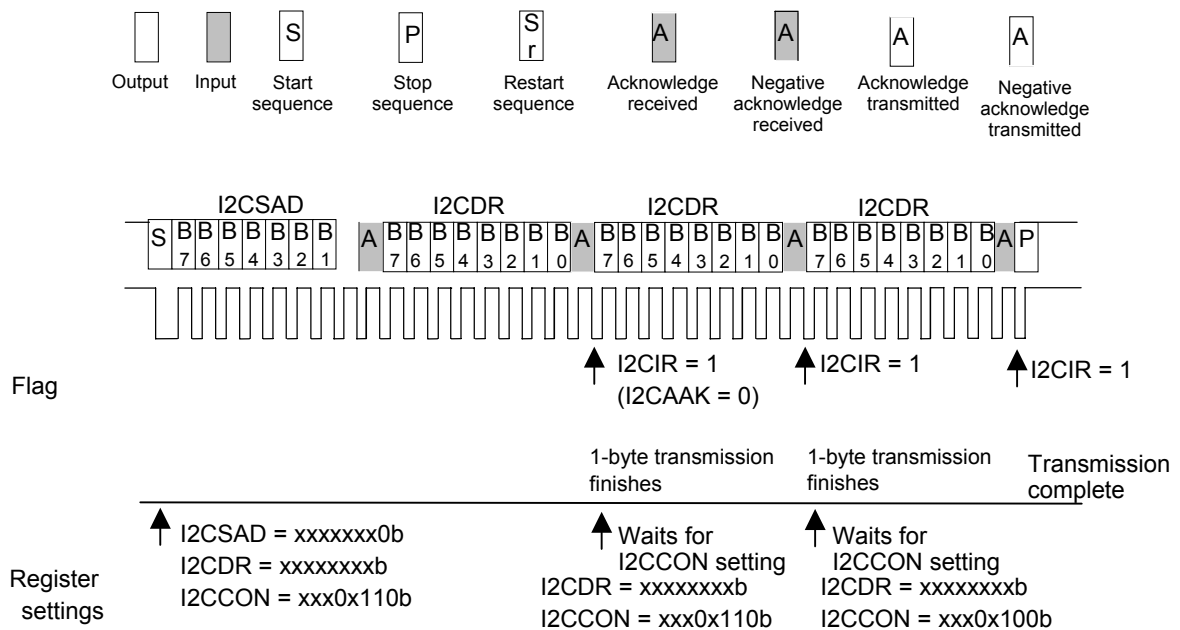
- ① To specify the communication destination, the address of the communication destination (slave) is written to SAD[6:0] of the I2CSAD register, and to set the transfer direction as reception, the I2CRW bit is set to "1" (received by the master device).
- ② The I2CCON register is set to XXX0X100. The following series of operations is automatically performed when the STCM bit (bit 2 of the I2CCON register) is set to "1": Transmission of the start sequence (to attain bus access), transmission of the slave address and the transfer direction specified to the I2CSAD register, confirmation of an acknowledge from the slave device for the address transmitted, and storage of 8-bit data transmitted from the slave device into the I2CDR register. Also, when the I2COC bit (bit 1 of the I2CCON register) is set to "0," a negative acknowledge for received data as well as the stop sequence are successively transmitted (the bus is released), and communication is then finished. At this point, the I2CIR bit is set to "1," indicating that the reception of 1-byte of data has been finished. The data received can be retrieved by reading the I2CDR register. If an acknowledge for the address transmitted has not been returned normally, the I2CAAK bit is set to "1" upon completion of reception.





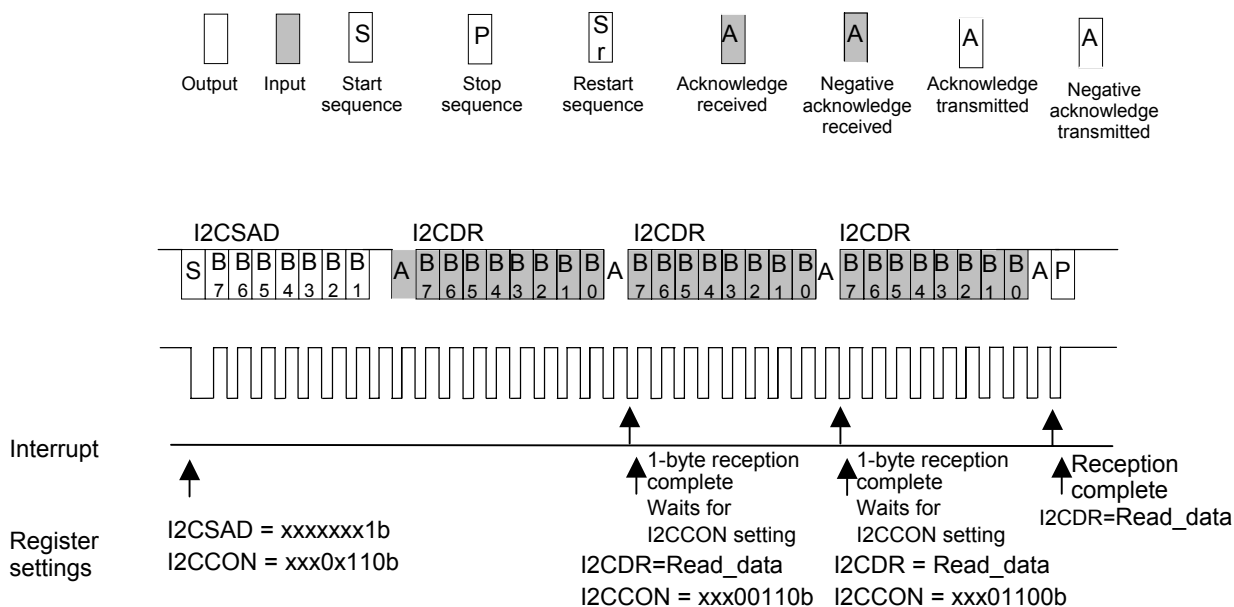
20.3.3 Transmit operation (transfer of 2 or more bytes from master to slave, in 7-bit address mode)

- ① Repeat step ① from section 20.3.1
- ② Repeat step ② from section 20.3.1
- ③ The I2CCON register is set to XXX0X110. The following series of operations is automatically performed when the STCM bit (bit 2 of the I2CCON register) is set to "1": Transmission of the start sequence, transmission of the slave address and the transfer direction specified to the I2CSAD register, confirmation of an acknowledge from the slave device for the address transmitted, transmission of the transmit data specified to the I2CDR register, and confirmation of an acknowledge from the slave device for the data transmitted. Here, because the I2COC bit (bit 1 of the I2CCON register) is set to "1," the stop sequence will not be transmitted (the bus remains busy) and the command wait state is activated. At this point, the I2CIR bit is set to "1," indicating that the transmission of 1-byte of data has been finished. If acknowledges for the address and data transmitted have not been returned normally, both the I2CAAK and I2CDAK bits are set to "1" upon completion of transmission.
- ④ At this stage, the application program must monitor I2CAAK and I2CDAK bits for proper error handling and recovery from I2C bus errors. Also, the application program should reset the I2CIR bit as necessary.
- ⑤ Multiple bytes are transmitted by repeating from step ③ to ④ the required number of times.
- ⑥ To transmit the last byte after multiple bytes have been transmitted, step ③ at the time of 1-byte transmit operation is performed first. Next, the stop sequence is transmitted (the bus is released) after the last byte is transmitted, and transmission is then finished.



20.3.4 Receive operation (transfer of 2 byte or more from slave to master, in 7-bit address mode)

- ① Repeat step ① from section 20.3.2
- ② The I2CCON register is set to XXX0X110. The following series of operations is automatically performed when the STCM bit (bit 2 of the I2CCON register) is set to "1": Transmission of the start sequence, transmission of the slave address and the transfer direction specified to the I2CSAD register, confirmation of an acknowledge from the slave device for the address transmitted, storage of 8-bit data transmitted from the slave device into the I2CDR register, and transmission of an acknowledge for the data received. Here, because the I2COC bit (bit 1 of the I2CCON register) is set to "1," the stop sequence will not be transmitted (the bus remains busy) and the command wait state is activated. At this point, the I2CIR bit is set to "1," indicating that the reception of 1-byte of data has been finished. The data received can be retrieved by reading the I2CDR register. If an acknowledge for the address transmitted has not been returned normally, the I2CAAK bit is set to "1" upon completion of reception.
- ③ Multiple bytes are received by repeating step ② the required number of times.
- ④ To receive the last byte after multiple bytes have been received, step ②. Next, an acknowledge is transmitted for the data received, the stop sequence is transmitted (the bus is released) after the last byte is received, and reception is then finished.



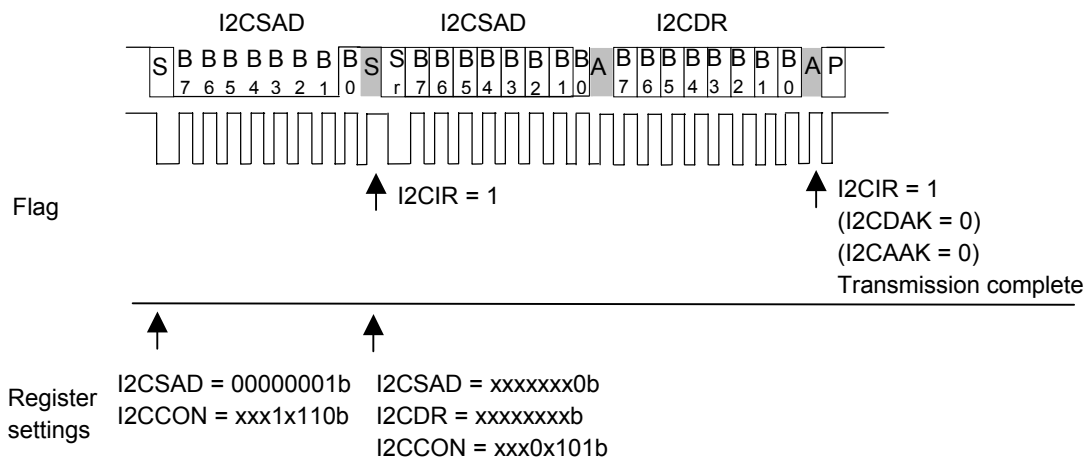


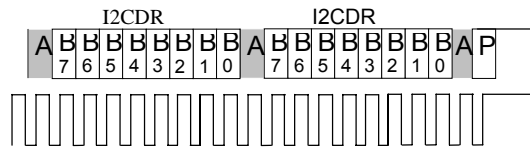
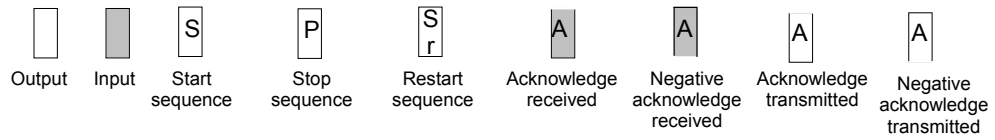
### 20.3.7 Start byte transmit operation

There are some I2C devices that require a unique start sequence in order to work properly. This sequence can be supported in this I2C block by using the special start byte generation technique. In this LSI, the transmission of the start byte is accomplished by setting the START bit (bit 4 of the I2CCON register) to "1."

To transfer 1-byte of data to a slave device that requires the start byte:

- ① The I2CSAD register is set to 00000001 (start byte pattern).
- ② The I2CCON register is set to XXX1X110. The following series of operation is automatically performed when both the STCM bit (bit 2 of the I2CCON register) and the START bit (bit 4 of the I2CCON register) are set to "1": Transmission of the start sequence (to attain bus access), and transmission of the start byte pattern specified to the I2CSAD register.
- ③ Step ① at the time of 1-byte transmit operation is performed.
- ④ Step ② at the time of 1-byte transmit operation is performed.
- ⑤ The I2CCON register is set to XXX0X101. The following series of operations is automatically performed when the I2CEN bit (bit 0 of the I2CCON register) and the STCM bit (bit 2 of the I2CCON register) are set to "1": Transmission of the restart sequence, transmission of the slave address and the transfer direction specified to the I2CSAD register, confirmation of an acknowledge from the slave device for the data transmitted, transmission of the transmit data specified to the I2CDR register, and confirmation of an acknowledge from the slave device for the data transmitted. Also, when the I2COC bit (bit 1 of the I2CCON register) is set to "0," the stop sequence is successively transmitted (the bus is released) and communication is then finished. At this point, the I2CIR bit is set to "1," indicating that the transmission of 1-byte of data has been finished. If acknowledges for the address and data transmitted have not been returned normally, both the I2CAAK and I2CDAK bits are set to "1" upon completion of transmission.





I2CIR = 1 (I2CAAK = 1) (I2CDR = 1) 1-byte transmission finishes  
 ↑ Waits for I2CCON setting  
 I2CDR = xxxxxxxxb  
 I2CCON = xxx0x110b

I2CIR = 1 (I2CDR = 1) 1-byte transmission finishes  
 ↑ Waits for I2CCON setting  
 I2CDR = xxxxxxxxb  
 I2CCON = xxx0x100b

I2CIR = 1 (I2CDR = 1) Transmission complete

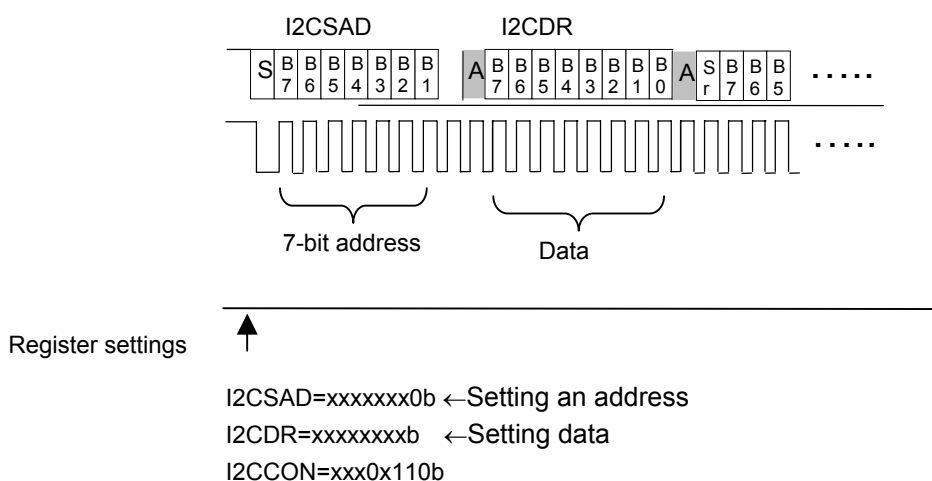
### 20.3.8 7-bit Address Mode and 10-bit Address Mode

There are two types of slave address specification methods: 7-bit address mode and 10-bit address mode.

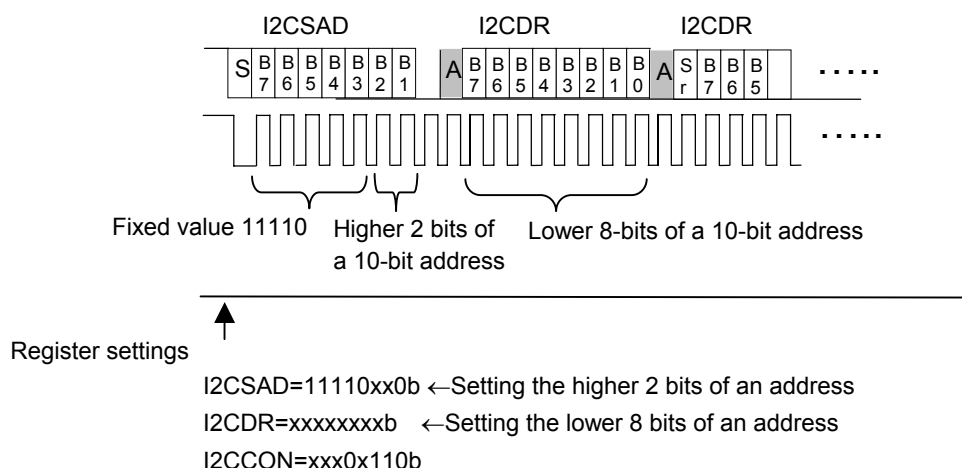
In 7-bit address mode, set an address in the higher 7 bits of the I2CSAD register, namely SAD [7:1]. Then, set the data transfer direction (transmission or reception) in the least significant bit, SAD0.

In 10-bit address mode, set an address in the higher 2 bits of the I2CSAD register, namely SAD [2:1], as well as in the lower 8 bits of the I2CDR register, namely XXXX[7:0]. Then, set a fixed value of 11110, which indicates that succeeding addresses are in 10-bit address mode, in the higher 5 bits of the I2CSAD register, namely SAD [7:3]. Furthermore, set the data transfer direction (transmission or reception) in the least significant bit, SAD0, of the I2CSAD register.

#### 7-bit Address Mode



#### 10-bit Address Mode



*Chapter 21*

**Analog-to-Digital Converter**

---

## Chapter 21 Analog-to-Digital Converter

### 21.1 Overview

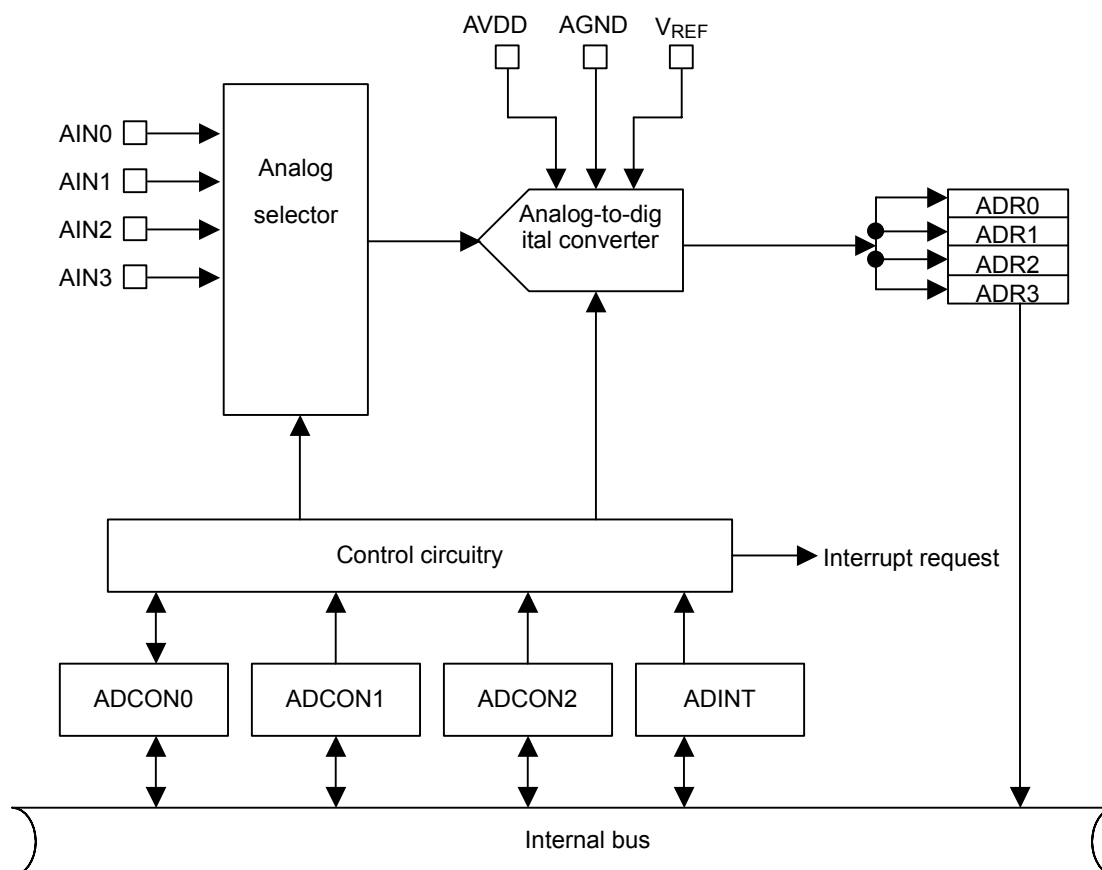
The built-in 4-channel, 10-bit resolution analog-to-digital converter supports two modes of operation: Scan mode sequentially converts input from the selected range of channels; select mode converts input from a single channel.

The conversion from an analog quantity to a digital one uses consecutive comparison with a sample and hold function.

At the user's option, the analog-to-digital converter issues an interrupt request after one cycle through the specified channels in scan mode and after each conversion in the select mode.

#### 21.1.1 Components

Figure 21.1 shows the analog-to-digital converter components.



- AIN0 to AIN3: Analog input pins
- ADR0 to ADR3: Converter 10-bit result registers
- ADINT: Analog-to-digital converter interrupt control register
- ADCON0: Converter control register 0
- ADCON1: Converter control register 1
- ADCON2: Converter control register 2
- AVDD: Analog VDD pin
- AGND: Analog ground pin
- V<sub>REF</sub>: Analog reference voltage

**Figure 21.1 Analog-to-Digital Converter Components**



### 21.1.2 Pin List

Pin Name	I/O	Description
AVDD	VDD	Power supply for the analog-to-digital converter
VREF	I	High Reference voltage for the analog-to-digital converter *1
VREFP	I	High Reference voltage for the analog-to-digital converter *2
VREFN	I	Low Reference voltage for the analog-to-digital converter *2
AIN[0]	I	Analog-to-digital converter analog input port 0
AIN[1]	I	Analog-to-digital converter analog input port 1
AIN[2]	I	Analog-to-digital converter analog input port 2
AIN[3]	I	Analog-to-digital converter analog input port 3
AGND	GND	Ground for the analog-to-digital converter

\*1: ML674001 Series only

\*2: ML675001 Series only

### 21.1.3 Control Register List

Address	Name	Abbreviation	R/W	Size	Initial Value
0xB6001000	Analog-to-digital converter control register 0	ADCON0	R/W	16	0x0000
0xB6001004	Analog-to-digital converter control register 1	ADCON1	R/W	16	0x0000
0xB6001008	Analog-to-digital converter control register 2	ADCON2	R/W	16	0x0003
0xB600100C	Analog-to-digital converter interrupt control register	ADINT	R/W	16	0x0000
0xB6001010	Analog-to-digital converter forced interrupt register	ADFINT	R/W	16	0x0000
0xB6001014	Analog-to-digital converter result register 0	ADR0	R/W	16	0x0000
0xB6001018	Analog-to-digital converter result register 1	ADR1	R/W	16	0x0000
0xB600101C	Analog-to-digital converter result register 2	ADR2	R/W	16	0x0000
0xB6001020	Analog-to-digital converter result register 3	ADR3	R/W	16	0x0000

**Note:** Writing to a result register (ADR0 to ADR7) while the analog-to-digital converter is in operation invalidates the contents of the entire set.

## 21.2 Control Register Descriptions

### 21.2.1 Analog-to-Digital Converter Control Register 0 (ADCON0)

This register controls scan mode operation.  
The program has read/write access to this register.  
The contents after a reset are 0x0000.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCON0	—*	—*	—*	—*	—*	—*	—*	—*	—*	SCNC	—*	ADRUN	—*	—*	ADSNM[1:0]	
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB6001000 [H]

Access: R/W

Access size: 16 bits

#### Note

\*: These bits are reserved for future expansion. They return “0” for reads. Writes to them are ignored.  
It needs 2HCLK + 2CCLK for ADCON0 register write recovery time.

#### Bit Descriptions

- **ADSNM[1:0]** (bits 1 to 0):  
This field specifies the first channel to scan. The last is always #7.

ADSNM		Channels	Scan order
1	0		
0	0	ch0 to ch3	ch0→ch1→ch2→ch3
0	1	ch1 to ch3	ch1→ch2→ch3
1	0	ch2 to ch3	ch2→ch3
1	1	ch3 to ch3	ch3

Do not change this setting while the analog-to-digital converter is in operation. Writes only take effect when ADRUN (bit 4) is “0.”

- **ADRUN** (bit 4):  
This bit turns the analog-to-digital converter operation on and off in scan mode.

ADRUN	Description
0	Stop analog-to-digital converter
1	Start analog-to-digital converter

Note that the ADRUN bit is a control bit. It is not a flag bit indicating the current analog-to-digital converter status (converting/idle).

- **SCNC** (bit 6):  
This bit specifies the action to take after one cycle through the specified channels.

<b>SCNC</b>	<b>Description</b>
0	Cycle back to first channel after one cycle through the specified channels, analog-to-digital converter
1	Stop after one cycle through the specified channels, analog-to-digital converter

If SCNC is "1," setting INTSN, the bit in the ADINT register indicating that the cycle is complete, to "0" is sufficient to start the next cycle because the ADRUN bit remains at "1."

**Note:** If SCNC is "1," stopping conversion in the middle requires simultaneously setting both SCNC and ADRUN to "0" with the same write. Simultaneously setting both back to "1" then restarts conversion. Note that stopping conversion this way invalidates the contents of the result register for the current channel.

### 21.2.2 Analog-to-Digital Converter Control Register 1 (ADCON1)

This register controls select mode operation.  
The program has read/write access to this register.  
The contents after a reset are 0x0000.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCON1	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	STS	—*	—*	ADSTM[1:0]	
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB6001004 [H]

Access: R/W

Access size: 16 bits

#### Note

\*: These bits are reserved for future expansion. They return “0” for reads. Writes to them are ignored.  
It needs 2HCLK + 2CCLK for ADCON1 register write recovery time.

#### Bit Descriptions

- **ADSTM[1:0]** (bits 1 to 0):  
This field specifies the channel.

ADSTM		Channel
1	0	
0	0	ch0
0	1	ch1
1	0	ch2
1	1	ch3

Do not change this setting while the analog-to-digital converter is in operation. Writes only take effect when STS (bit 4) is “0.”

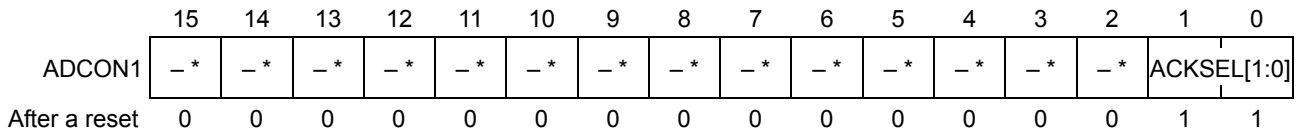
- **STS** (bit 4):  
This bit turns the analog-to-digital converter operation on and off in select mode.

STS	Description
0	Stop analog-to-digital converter
1	Start analog-to-digital converter

The hardware automatically resets this bit to “0” when conversion is complete.

### 21.2.3 Analog-to-Digital Converter Control Register 2 (ADCON2)

This register specifies the operating clock frequency for the analog-to-digital converter register. The program has read/write access to this register. The contents after a reset are 0x0003.



Address: 0xB6001008 [H]  
 Access: R/W  
 Access size: 16 bits

**Note**

\*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

**Bit Descriptions**

- **ACKSEL[1:0]** (bits 1 and 0):  
 This field specifies the divisor for deriving the operating clock from CCLK.

ACKSEL		Channel
1	0	
0	0	Reserved
0	1	CCLK/2
1	0	CCLK/4
1	1	CCLK/8

**Notes**

In ML674001 series, choose the divisor so that the period to produce is 200 - 1000 ns.  
 In ML675001 series, choose the divisor so that the period to produce is 80 - 1250 ns.  
 The conversion time, 25 clock cycles, depends on the operating frequency and ACKSEL. The combination of 33 MHz for CCLK and 11B for ACKSEL, for example, yields one of 240 ns  $\times$  25 = 6  $\mu$ s.

**Table 18.1 Sample CCLK-ACKSEL Combinations**

CCLK (MHz)	ACKSEL		
	01	10	11
60	-	-	133 ns *1
33	-	120 ns *1	240 ns
20	100 ns *1	200 ns	400 ns
16	120 ns *1	240 ns	480 ns
8	250 ns	500 ns	1000 ns
2	1000 ns	-	-

**Note:** The shading indicates combinations that produce periods out of range.

\*1: ML675001 series can be set up.

### 21.2.4 Analog-to-Digital Converter Interrupt Control Register (ADINT)

This register contains analog-to-digital converter interrupt settings.  
 The program has read/write access to this register.  
 The contents after a reset are 0x0000.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADINT	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	ADSTIE	ADSNIE	INTST	INTSN
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB600100C [H]  
 Access: R/W  
 Access size: 16 bits

**Note**

\*: These bits are reserved for future expansion. They return “0” for reads. Writes to them are ignored.

**Bit Descriptions**

- **INTSN (bit 0):**  
 A “1” in this bit indicates completion of one cycle through the specified channels--that is, that channel 3 conversion in scan mode is complete.

INTSN	Description
0	Scan not complete (channel 3 conversion in scan mode not complete)
1	Scan complete (channel 3 conversion in scan mode complete)

The program must explicitly reset this bit to “0” by writing “1” to it.

- **INTST (bit 1):**  
 A “1” in this bit indicates completion of select mode conversion.

INTST	Description
0	Select mode conversion not complete
1	Select mode conversion complete

The program must explicitly reset this bit to “0” by writing “1” to it.

- **ADSNIE (bit 2):**  
 Setting this bit to “1” produces an interrupt request when one cycle through the specified channels is complete--that is, when channel 3 conversion in scan mode is complete.

ADSNIE	Description
0	Disable interrupt after scan
1	Enable interrupt after scan

- **ADSTIE** (bit 3):  
Setting this bit to “1” produces an interrupt request when select mode conversion is complete.

<b>ADSTIE</b>	<b>Description</b>
0	Disable interrupt after conversion
1	Enable interrupt after conversion

### 21.2.5 Analog-to-Digital Converter Forced Interrupt Register (ADFINT)

This register is for forcing an analog-to-digital converter interrupt.  
 The program has read/write access to this register.  
 The contents after a reset are 0x0000.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADFINT	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	—*	ADFAS
After a reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: 0xB6001010 [H]

Access: R/W

Access size: 16 bits

**Note**

\*: These bits are reserved for future expansion. They return “0” for reads. Writes to them are ignored.

**Bit Descriptions**

- **ADFAS** (bit 0):  
 Setting this bit to “1” forces assertion of the interrupt signal. This facility is for test purposes. Writing “0” to this bit resets it.

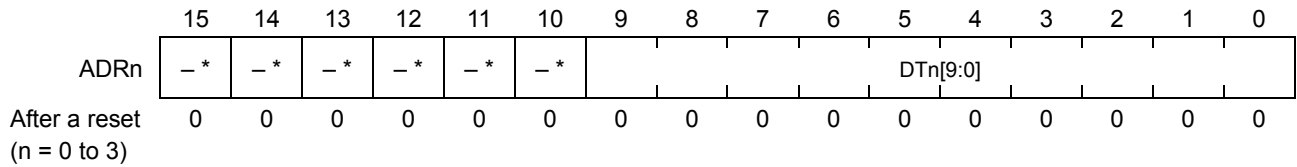


### 21.2.6 Analog-to-Digital Converter Result Registers (ADR0 to ADR3)

These registers hold the result of converting the corresponding analog input.

The program has read/write access to these registers.

The contents after a reset are 0x0000.



Address: 0xB6001014(ADR0), 0xB6001018(ADR1), 0xB600101C (ADR2), 0xB6001020(ADR3)

Access: R/W

Access size: 16 bits

#### Notes

\*: These bits are reserved for future expansion. They return "0" for reads. Writes to them are ignored.

Writing to a result register (ADR0 to ADR3) while the analog-to-digital converter is in operation invalidates the contents of the entire set.

#### Bit Descriptions

- **DTn** (bits 9 to 0):  
This is the result of converting the corresponding analog input.

## 21.3 Operational Description

The analog-to-digital converter supports two modes of operation: Scan mode sequentially converts input from the selected range of channels; select mode converts input from a single channel.

These two modes cannot be used simultaneously.

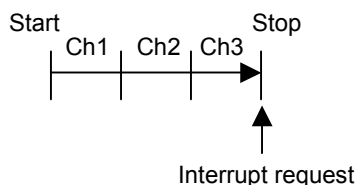
Do not change modes while the analog-to-digital converter is in operation.

Changing modes invalidates the contents of the result registers (ADR0 to ADR3).

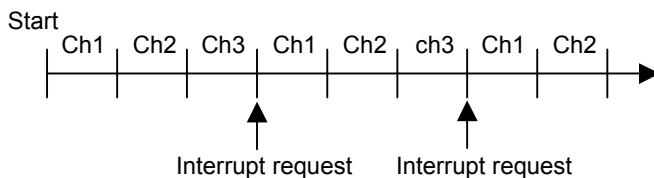
### 21.3.1 Scan Mode

Scan mode sequentially converts input from the selected channel (0 to 3) through channel 3. The program has a choice of stopping conversion or cycling back to the first channel when channel 3 conversion is complete.

Figure 21.2 and Figure 21.3 illustrate scan mode operation.



**Figure 21.2 Sample Scan Mode Operation over Channels 1 to 3 Stopping after Cycle (SCNC = 1)**

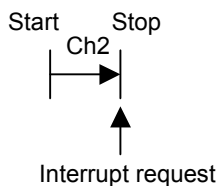


**Figure 21.3 Sample Scan Mode Operation over Channels 1 to 3 Recycling (SCNC = 0)**

### 21.3.2 Select Mode

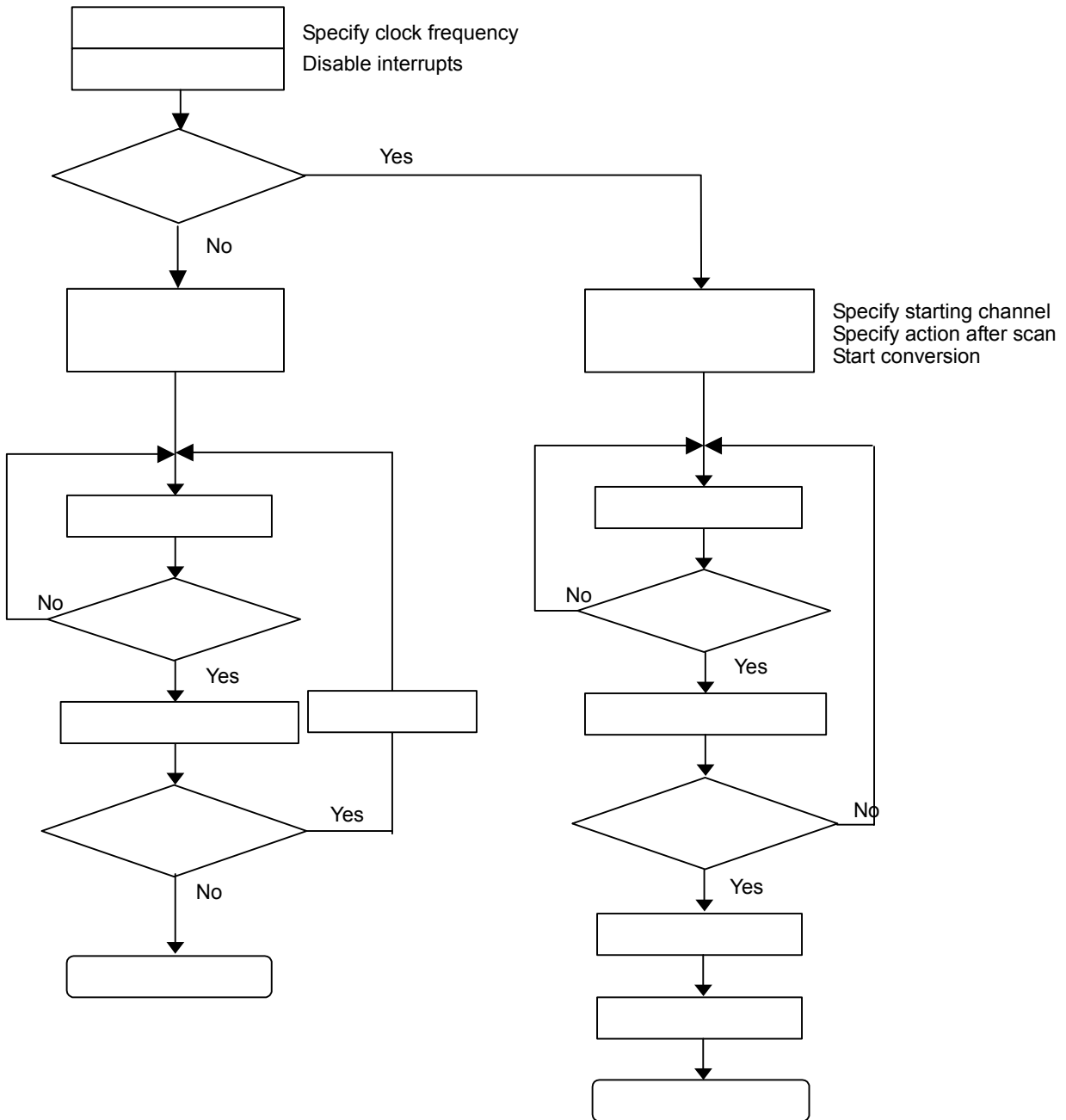
Select mode converts input from a single channel.

Figure 21.4 illustrates select mode operation.



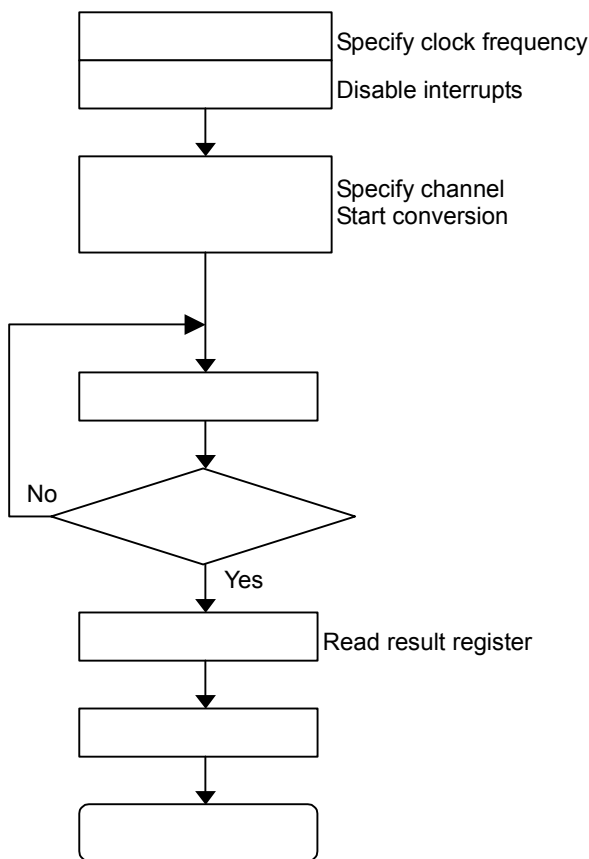
**Figure 21.4 Sample Select Mode Operation Using Channel 2**

Flowcharts



**Figure 21.5 Scan Mode Example**

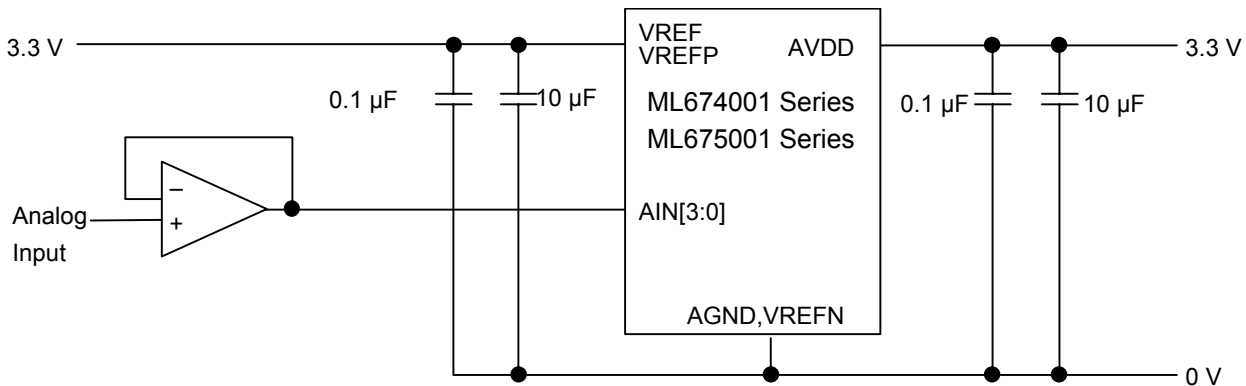
- Operating conditions
- Operating clock: 1/8 CCLK
  - Interrupt requests: No
  - Scan channels: 1 to 3



**Figure 21.6 Select Mode Example**

- |                             |
|-----------------------------|
| Operating conditions        |
| • Operating clock: 1/8 CCLK |
| • Interrupt requests: No    |
| • Channel: 2                |

**Sample Pin Circuit**



**Figure 21.7 Sample Analog-to-Digital Converter Connections**

## *Chapter 22*

# **Built-In Flash Memory**

---

## Chapter 22 Built-In Flash Memory

### 22.1 Overview

This LSI\*<sup>1</sup> has built-in flash memory. The flash memory is electrically programmable, non-volatile memory; thus, this LSI provides simple and powerful, multiple programming methods. The features of built-in flash memory are as follows:

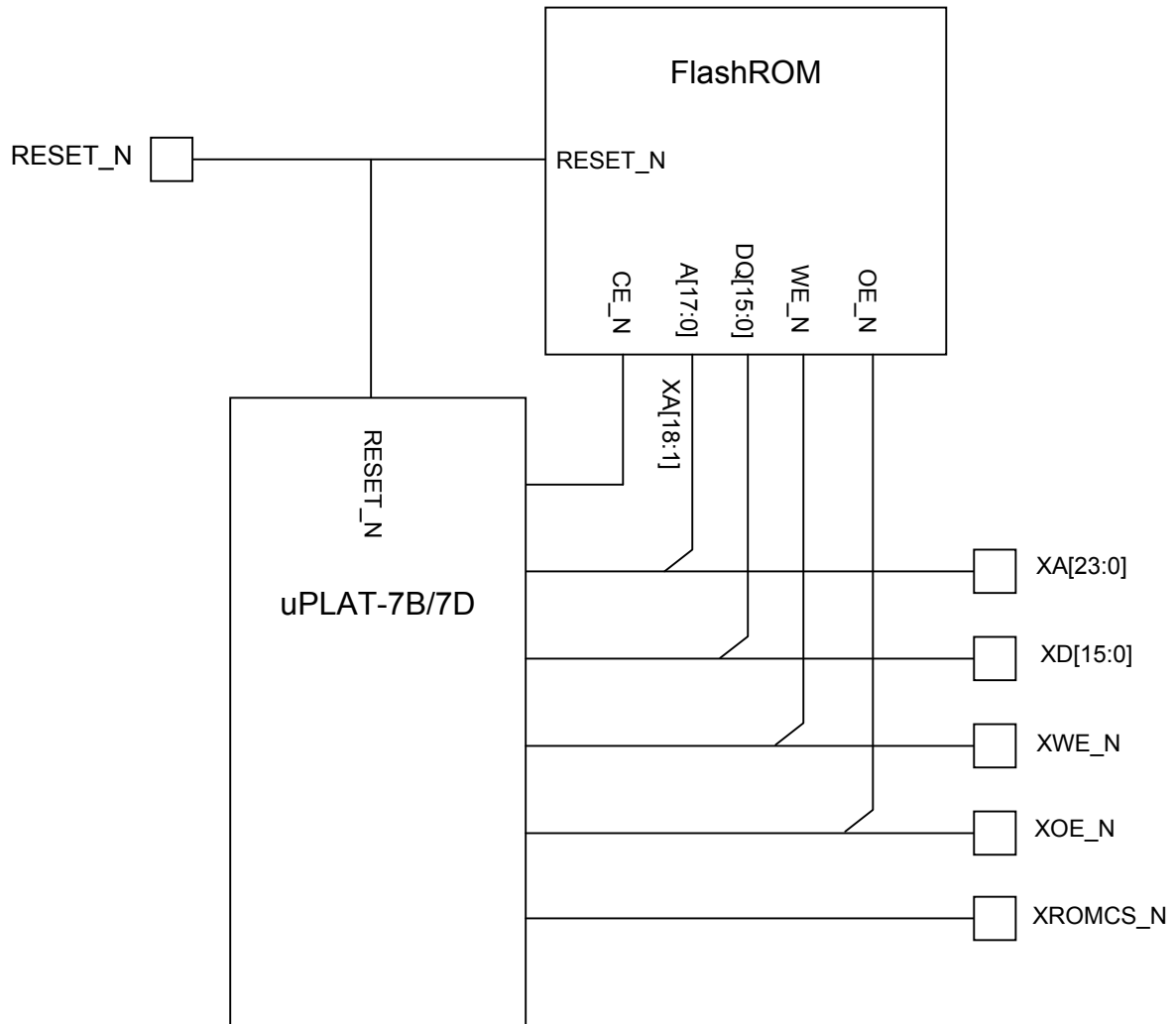
\*1: The ML674001 and ML675001 have no built-in flash memory.

#### Features

- Size of built-in flash memory
  - ML67Q4002: 256 Kbyte (128K x 16 bits)
  - ML67Q4003: 512 Kbyte (256K x 16 bits)
  - ML67Q5002: 256 Kbyte (128K x 16 bits)
  - ML67Q5003: 512 Kbyte (256K x 16 bits)
- Flash memory can be read and programmed with a single power supply.
- Built-in flash memory programming methods
  - Programming of flash memory using the JTAG (Joint Test Action Group) debug function
  - Programming of flash memory using the built-in boot program
- Programming unit: 2 bytes
- Erasing units
  - Sector erase: 2 Kbyte/sector
  - Block erase: 64 Kbyte/block
  - Chip erase: Entire area in a batch
- High-speed programming: Programming time 2 bytes/20 μsec
- High-speed erasing: Erase time
  - Sector erase/block erase: 25 msec
  - Chip erase: 100 msec
- Write protect function
  - Block protect: 16 Kbyte on the top address side
  - Chip protect: Entire flash memory area
- Write protect/unprotect time
  - Block protect /chip protect: 20 μsec
  - Block unprotect /chip unprotect: 25 msec
- Control by JEDEC-conformed SDP (Software Data Protect) command sequence
- Highly reliable reading/writing
  - Sector writing: 1000 times
  - Data holding period: 10 years

### 22.1.1 Block Diagram

Figure 22.1 gives a block diagram for the Built In Flash Memory.



**Figure 22.1** Block Diagram for the Built In Flash Memory

## 22.2 Flash Memory Programming

### 22.2.1 General Description of Flash Memory Programming

The following two methods are available to program built-in flash memory of this LSI.

- **Programming flash memory using the JTAG debug function**  
This method is used to program flash memory using the JTAG debug function. This method allows the on-board programming of flash memory.  
Using the built-in debug function, this method is used to load the flash memory program routine to built-in RAM or external SRAM on the board via the JTAG interface, execute the routine, and program flash memory.
- **Programming flash memory using the built-in boot program**  
This method is used to program flash memory using a serial interface. This method allows on-board programming of flash memory.  
In this method, the built-in boot program loads the flash memory program routine via the universal asynchronous receiver-transmitter (UART) to built-in RAM, executes the routine, and programs flash memory.

### 22.2.2 Flash Memory Programming Method Using the JTAG Debug Function

This method programs flash memory using the JTAG debug function. This method is used to load the flash memory rewrite routine to built-in RAM or external SRAM on the board via the JTAG interface, execute the routine, and program flash memory.

Flash memory is programmed when the flash memory program routine issues a command conforming to the JEDEC standard SDP command sequence. The downloading and starting of the flash memory program routine and the transfer of application data to be written to flash memory are controlled by using the built-in debug function via the JTAG interface.



(1) Pin Settings

The following table lists the pin settings necessary to use the JTAG debug function (see Chapter 23).

Pin name	Setting
JSEL	"L"
BSEL[1]	"L"
BSEL[0]	"L" or "H"

(2) Connection Method

Connect the JTAG pin with the host, such as a PC, or JTAG-compatible Flash Programmer via a JTAG debug interface tool.

For more information about the connections of the JTAG connectors on the user board, see Chapter 23.

(3) Description of the Flash Memory Programming Procedure

**Step 1: Initial Status**

Provide the flash memory program routine and application codes to be written to flash memory on the host.

**Step 2: Downloading the flash memory program routine**

Download the flash memory program routine to built-in RAM using the built-in debug function via the JTAG interface.

**Note:** In the case of on-board programming, if external SRAM has been mounted on the board, the routine can be downloaded to external SRAM instead of to built-in RAM.

**Step 3: Erasing Flash Memory**

Start up the flash memory program routine, which has been downloaded to built-in RAM, using the built-in debug function via the JTAG interface. The flash memory program routine erases the contents of flash memory.

**Step 4: Writing Application Codes**

The flash memory program routine loads the application codes to be written to flash memory via the JTAG interface, and then writes them to built-in flash memory. After writing is finished, apply protect to flash memory as necessary.

**Note:** By changing the flash memory program routine, it is possible to program flash memory mounted on the board in addition to built-in flash memory.

### 22.2.3 Flash Memory Programming Method Using the Built-in Boot Program

This method is used to program flash memory using a serial interface. This method is used to start the built-in boot program, load the flash memory program routine via the UART to built-in RAM, execute the routine, and program flash memory.

Flash memory is programmed when the flash memory program routine issues a command conforming to the JEDEC standard SDP command sequence.

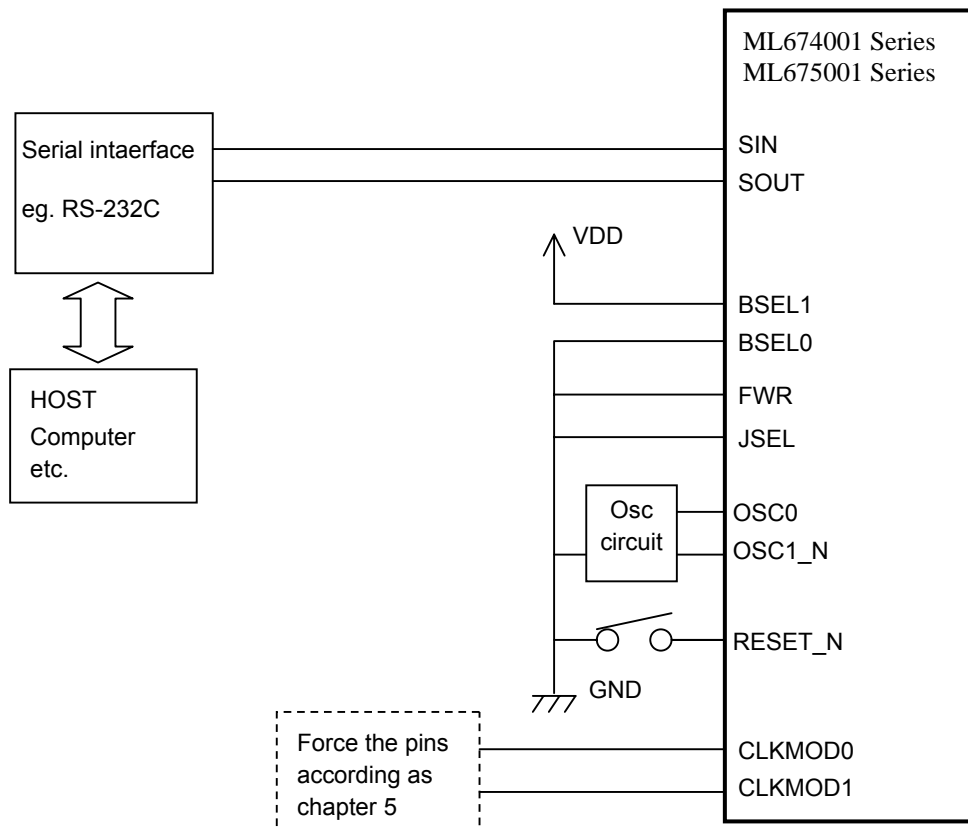
#### (1) Pin Settings

The following table lists the pin settings necessary to start the built-in boot program at the time of reset. As for the JTAG pin, set it to the setting when it is not used.

Pin name	Setting
JSEL	"L"
BSEL[1]	"H"
BSEL[0]	"L" or "H"

#### (2) Example of a Connection Circuit

Connect the UART with the host such as a PC.



\* Set the other unused pins, according as the section 1.3.6.

Figure 22-2 Example of a Connection for programming the flash memory (use of boot loading function)

(3) Description of the Flash Memory Programming Procedure

Step 1: Initial Status

Provide the flash memory program routine and application codes to be written to flash memory on the host.

Step 2: Downloading the flash memory program routine

Set the pins in such a way that the built-in boot program is executed at the time of reset. The built-in boot program downloads the flash memory program routine via the UART to built-in RAM.

Step 3: Erasing Flash Memory

The built-in boot program starts the flash memory program routine that has been downloaded to built-in RAM. The flash memory program routine erases the contents of flash memory.

Step 4: Writing Application Codes

The flash memory program routine loads the application codes to be written to flash memory via the UART, and then writes them to built-in flash memory. After writing is finished, apply protect to flash memory as necessary.

Note: Please obtain the flash memory program routine in advance.

By changing the flash memory program routine, it is possible to program flash memory mounted on the board in addition to built-in flash memory.

## 22.3 Built-in Boot Program

### 22.3.1 Functional Description

The built-in boot program is the software that downloads the programs you created to built-in RAM via the UART and executes them. With the built-in boot program, writing to built-in flash memory can be performed by executing the flash writing program you created. Note, however, that programs that will be downloaded to this built-in RAM must be in the Intel-HEX format.

### 22.3.2 Operating Procedure

The built-in boot program runs the programs you created on this LSI according to the following procedure:

1. Start this LSI from boot ROM.
2. Transmit binary 0 data from terminal software to this LSI via the UART.
3. Download the program you created from terminal software to built-in RAM via the UART.
4. Following the instructions of the built-in boot program, start the program you created.

### 22.3.3 Operating Environment

1. Devices Equipped with This LSI
  - (1) Operating Frequency: ML674001 Series  
8 to 33 MHz  
ML675001 Series  
8 to 60 MHz
  - (2) UART port: 1 channel  
Connect PIOA[0]/SIN and PIOA[1]/SOUT to the serial port of the host.
2. Host (PC, etc.)
  - (1) Software: Software that supports serial communication with the settings list in the table below
  - (2) Serial port: 1 channel

**List of Settings for Serial Communication Software**

Item	Contents	
	ML674001 Series	ML675001 Series
Baud rate	9600 (8 MHz to 33 MHz)	9600 (8 MHz to 60 MHz)
	19200 (16 MHz to 33 MHz)	19200 (16 MHz to 60 MHz)
	38400 (33 MHz)	38400 (33 MHz to 60 MHz)
Parity bit	None	
Character length	8 bits	
Stop bit	1	
Flow control	Xon/Xoff	

## 22.4 SDP Command Sequence Control for Built-In Flash Memory

### 22.4.1 Operational Description

Built-in flash memory is equipped with a command register to facilitate interface and control. Read, erase, program, and other functions required for built-in flash memory are executed via the command register. Commands are written into the command register by the SDP (Software Data Protect) command sequence employing standard JEDEC commands. A command list and the functional descriptions of the commands are shown below:

**Command Sequence List**

Command	Req. cycle	1st cycle		2nd cycle		3rd cycle		4th cycle		5th cycle		6th cycle	
		Address Note 1	Data Note 2	Address Note 1	Data Note 2	Address Note 1	Data Note 2	Address Note 1	Data Note 2	Address Note 1	Data Note 2	Address Note 1	Data Note 2
Read/Reset	1	XXX	F0										
	3	555	AA	2AA	55	555	F0	RA Note 3	RD Note 3				
Software ID Entry/Verify Protect	3	555	AA	2AA	55	555	90	RA Note 4	RD Note 4				
Word Program	4	555	AA	2AA	55	555	A0	PA Note 3	PD Note 3				
Sector Erase	6	555	AA	2AA	55	555	80	555	AA	2AA	55	SA Note 3	30
Block Erase	6	555	AA	2AA	55	555	80	555	AA	2AA	55	BA Note 3	50
Chip Erase	6	555	AA	2AA	55	555	80	555	AA	2AA	55	555	10
Block Protect	4	555	AA	2AA	55	555	E0	XXX	00				
Chip Protect	4	555	AA	2AA	55	555	D0	XXX	00				
Protect Cancel	4	555	AA	2AA	55	555	E0	XXX	01				

Note 1: • Address format: XA11 to XA1 (hexadecimal notation)

- XA17 to XA12 are don't cares.

Note 2: • Data format: DQ7 to DQ0 (hexadecimal notation)

- DQ15 to DQ8 are don't cares.

Note 3: • RA: Read address, PA: Program address, SA: Sector address (XA17 to XA11)

- RD: Read data, PD: Program data, SD: Sector data

Note 4: • RA: Read address, RD: Read data, Refer to section 22.4.8 or 22.4.9.

[Caution]

Be sure to use half-word (16-bit) access for the command sequence.

#### 22.4.2 Command Entries

For command entries, use the JEDEC-conformed SDP command sequence. After the SDP command sequence is finished, the selected operation is initiated automatically. If an incorrect address or data is entered in the SDP command sequence, the SDP command sequence is suspended and the mode returns to read mode.

#### 22.4.3 Read/Reset (Software Reset)

The Read/Reset command is used to end Software ID Entry/Verify Protect, or stop an erase or program operation.

A Read/Reset operation is performed by entering an SDP command of either one cycle or three cycles to the command register, and the mode returns to the read mode.

The flash memory is automatically set in read mode when the power is turned on.

#### 22.4.4 Erase

An erase operation is executed by entering an SDP command of six cycles - the sector erase command, block erase command, or chip erase command - to the command register, and is ended automatically by the control of the internal timer within the flash memory.

During an erase operation, DATA\_N poling that performs the detection of internal operations, the toggle bit, hardware reset, and software reset become valid.

The Sector Erase command places the selected 2KB memory array in the state of "1."

The Block Erase command places the selected 64KB memory array in the state of "1."

The Chip Erase command places the entire memory array area in the state of "1."

#### 22.4.5 Program

A program operation is performed by entering an SDP command of four cycles - the Program command - to the command register, and is ended automatically by the control of internal timing.

During a program operation, DATA\_N poling that performs the detection of internal operations, the toggle bit, hardware reset, and software reset become valid.

Please note, however, that the memory arrays of addresses to be programmed must be placed in the erase state prior to programming.

Programming is performed in units of 16 bits (2 bytes).

#### **Caution:**

**It is prohibited to re-program an address that has already been programmed once without erasing the existing program first.**

#### 22.4.6 Protect

The protect operation disables erase and program operations in the specified areas.

There are two types of protect: block protect and chip protect. Each type of protect is activated by entering an SDP command of four cycles - the Block Protect or Chip Protect command - to the command register.

Block protect function protects an address space of 16KB (8 sectors) from the top address side, and chip protect function protects the address space of the entire chip area.

Neither protect is canceled even if the power is turned off.

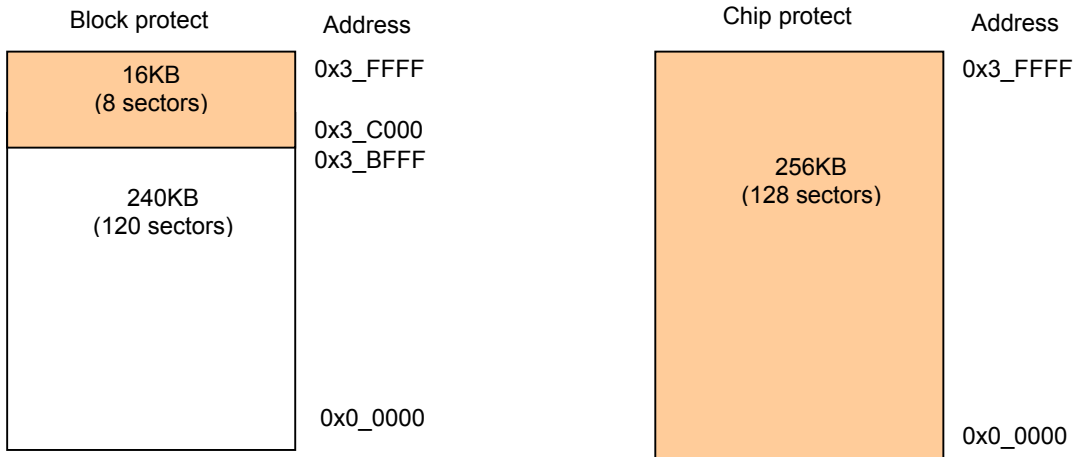
#### 22.4.7 Protect Cancel

Protect is cancelled by entering an SDP command of four cycles - the Cancel Protect command - to the command register.

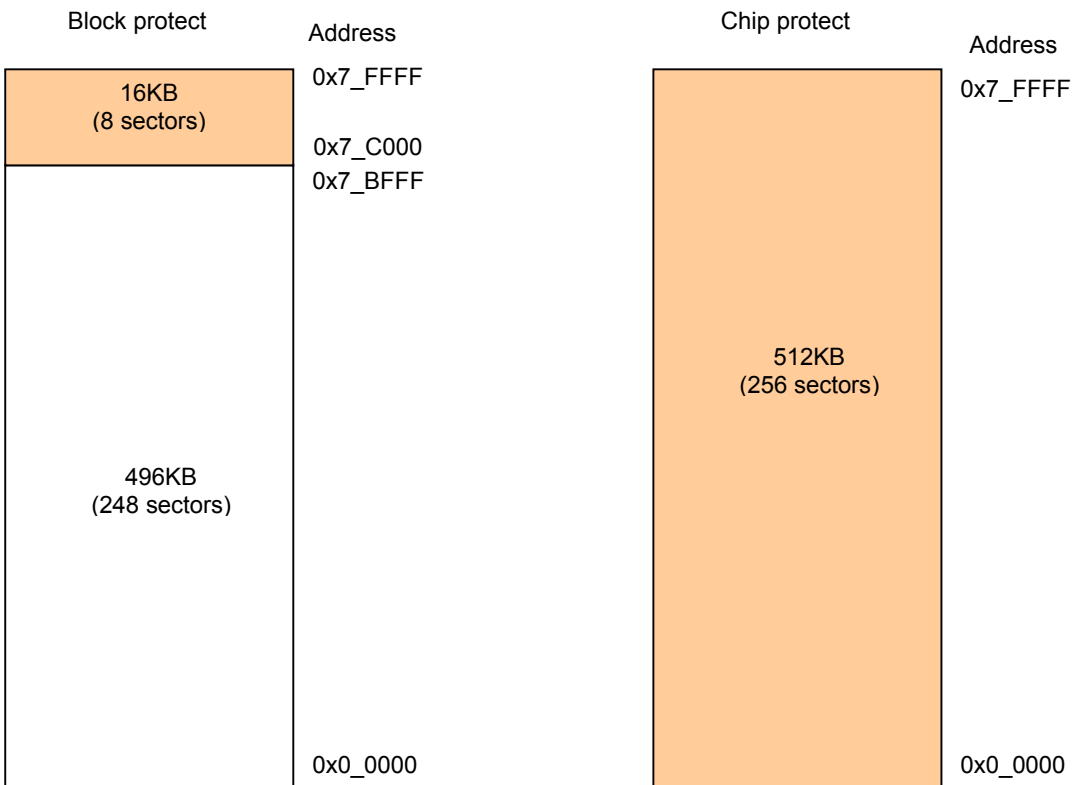
The execution of the Cancel Protect command cancels both block protect and chip protect at the same time.

Protected Areas

ML67Q4002/ML67Q5002



ML67Q4003/ML67Q5003



#### 22.4.8 Product Identification/Software ID

The product identification/software ID operation outputs the manufacturer code and device code, which are then ready by the programming device, making it possible to select erase and programming algorithms suitable for the device.

The manufacturer code and device code are output by an SDP command of four cycles - the Software ID command - to the command register. The following contents are output by the last read operation:

	Address	Output	Description
ML67Q4002/ML67Q4003/ ML67Q5002/ML67Q5003	0x0000	0x0062	Manufacturer code
	0x0001	0x0046	Device code (ML67Q4002/ML67Q5002)
		0x0002	Device code (ML67Q4003/ML67Q5003)

The operation ends when the Read/Reset command is entered.

#### 22.4.9 Verify Protect

Whether or not flash memory is protected is verified by entering an SDP command of four cycles - the Verify Protect command - to the command register. The following contents are output by the last read operation:

	Address	Output	Description
ML67Q4002/ML67Q4003/ ML67Q5002/ML67Q5003	0x0002	0x0001	Block Protected.
		0x0000	Not Block protected.
	0x0003	0x0001	Chip Protected.
		0x0000	Not Chip protected.

The operation ends when the Read/Reset command is entered.

#### 22.4.10 Hardware Reset

The hardware reset operation resets an erase operation, stops a program operation, or cancels the mode entered by an SDP command.



### 22.4.11 Detecting the End of an Erase or Program Cycle

The end of an erase or program cycle can be detected by DATA\_N polling or the toggle bit.

#### (1) DATA\_N Polling

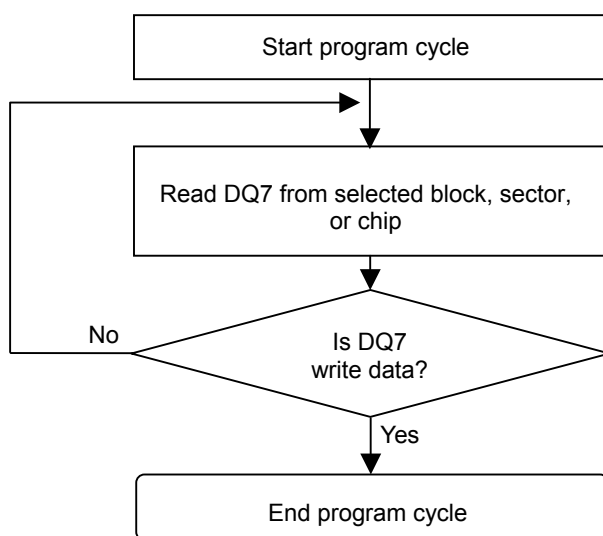
DATA\_N polling can be used to detect the end of an erase or program cycle.

During an erase cycle, "0" is read into DQ7\*, and upon completion of the erase cycle, "1" is read into DQ7. During a program cycle, an inverted value of the data loaded last is read into DQ7, and upon completion of the program cycle, the value of data loaded last is read into DQ7.

DATA\_N polling can be monitored any time during an erase or program cycle. It is recommended to verify the detection by DATA\_N polling twice continuously in order to avoid erroneous detections.

Note that, in order to make DATA-N polling function correctly, it is necessary to erase the existing program first before programming.

The following shows the flowchart for detecting the end of a program cycle by using DATA\_N polling:



\*Note: For DQ7, see Figure 22-1.

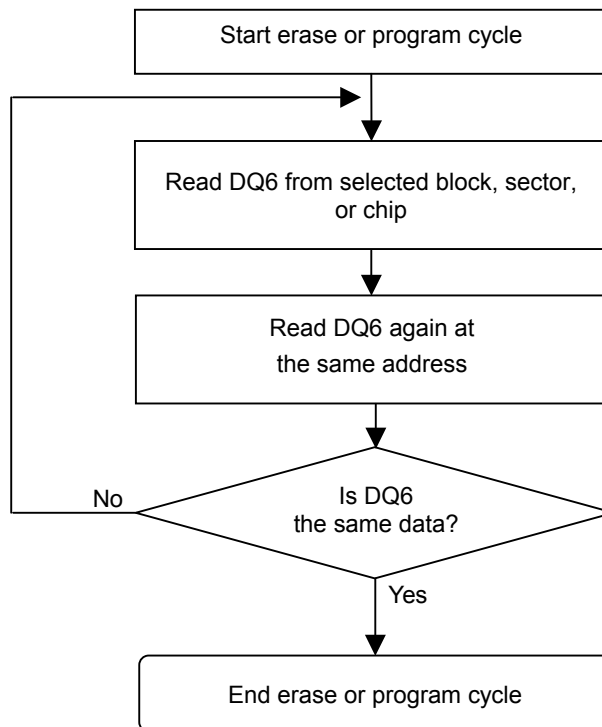
(2) Toggle Bit

The toggle bit can be used to detect the end of an erase or program cycle.

During an erase or program cycle, "0" and "1" are continuously read out from DQ6\* alternately. When an erase or program cycle finishes, toggling stops and the cycle returns to the normal read cycle.

The toggle bit can be monitored any time during an erase or program cycle. It is recommended to verify the detection by the toggle bit twice continuously in order to avoid erroneous detections.

The following shows the flowchart for detecting the end of a cycle by using the toggle bit



\*Note: For DQ6, see Figure 22-1.

## *Chapter 23*

# **JTAG**

---

## Chapter 23 JTAG

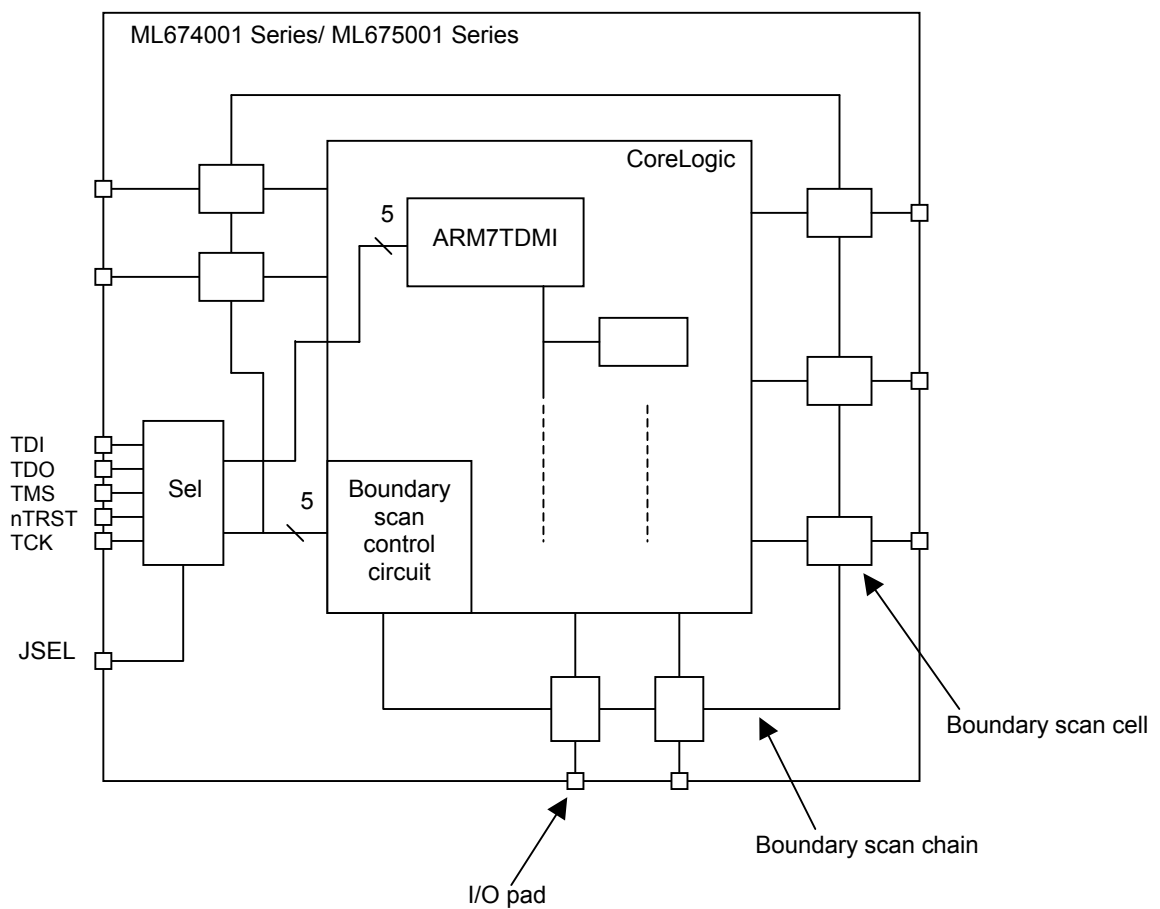
### 23.1 Overview

This LSI provides the following functions via the JTAG interface:

- On-board debug function
- Boundary scan function

#### 23.1.1 Configuration

Figure 23-1 shows the configuration of the JTAG interface circuit.



**Figure 23.1 JTAG Interface Circuit**

23.1.2 Pin List

Pin name	I/O	Function
TDI	I	Test data input
TDO	O	Test data output
nTRST	I	TAP controller reset signal
TMS	I	Test mode select
TCK	I	Test clock
JSEL	I	Mode setting signal 0: When performing on-board debugging 1: When performing boundary scanning

## 23.2 On-Board Debug Function

### 23.2.1 Necessary Conditions

This LSI has built-in debugging support, with JTAG pins for connecting debuggers and the like. The JTAG connector on the board provides access to these pins for onboard debugging. Onboard debugging requires the following.

- A standard JTAG to JTAG-ICE hardware such as ARM Multi-ICE.
- ARM's software development tools or other debugger software compatible with the JTAG-ICE hardware being utilized.
- Development host with the above debugger installed
- Appropriate connector cables

### 23.2.2 Connections

The following Figure gives the circuits for connecting this LSI to the JTAG-ICE hardware. (Adaptive = off)

The following are only the most important usage notes. For full details, refer to the JTAG-ICE manual and other references.

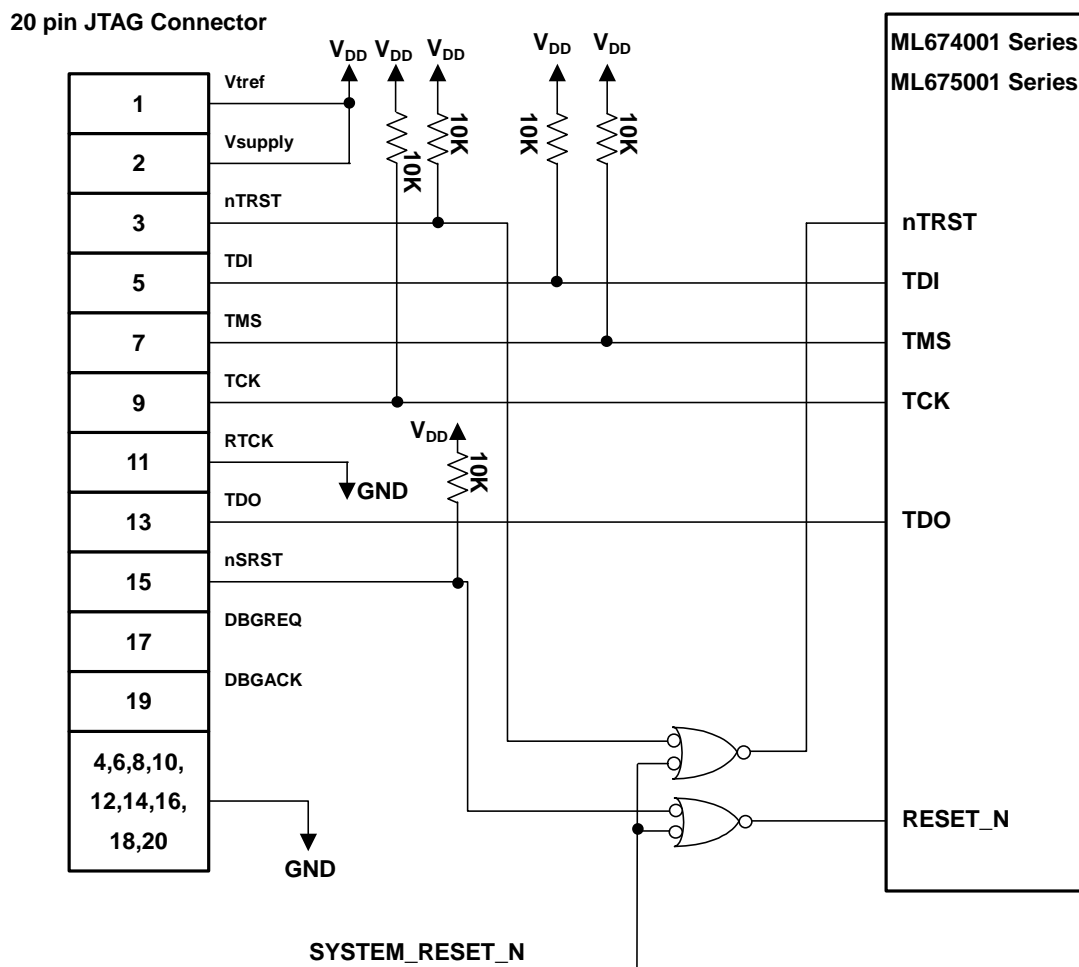


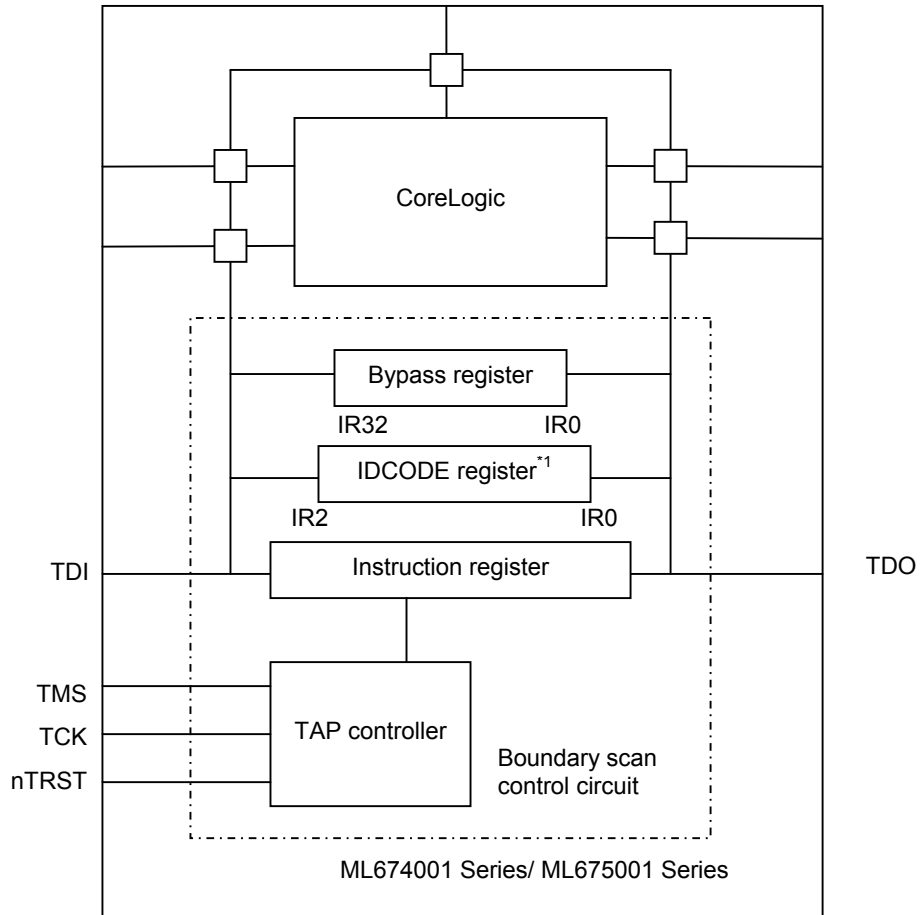
Figure 23.2 The circuits for connecting this LSI to the JTAG-ICE hardware

### 23.3 Boundary Scan Function

The boundary scan function built in this LSI conforms with the JTAG Boundary Scan Standard (IEEE1149-199). By using the boundary scan function, testing and failure diagnostics at chip level and board level are facilitated. The boundary scan circuit consists of five JTAG interface pins, a boundary scan control circuit, and boundary scan cells. Each of the boundary scan cells is placed between an I/O pad and the core logic circuit. Each of the boundary scan cells has the shift register function (boundary scan register); they are connected so as to form a scan chain surrounding the core logic circuit. In a normal operation, input/output signals are input to and output from the core logic circuit via the boundary scan cells. During a boundary scan test, only test signals are input to and output from the core logic circuit via the boundary scan chain and JTAG interface pins.

### 23.3.1 Boundary Scan Control Circuit

The boundary scan control circuit consists of a TAP controller, an instruction register, a bypass register, and an IDCODE register.



\*1 The IDCODE register is equipped in the ML674001 series only.

**Figure 23.3 Boundary Scan Control Circuit**



### 23.3.2 Registers

The following three registers are built in the boundary control circuit.

Register	Bit length	Function
Instruction register	3	Reads and decodes instructions for the TAP controller.
IDCODE register	32	Retains codes for identifying devices.
Bypass register	1	Provides the shortest route from TDI to TDO.

### 23.3.3 TAP Controller

The TAP controller generates signals that control the operations of the instruction register and data register (boundary scan register) by a state machine of 16 states. The TAP controller's transition states are mainly divided into two paths: for the instruction register and for the data register (boundary scan register). Figure 23-4 shows the TAP controller's state transition diagram. In this diagram, each of the labels enclosed by a frame represents a state name, and the number (0 or 1) surrounding each frame indicates the value of the TMS signal when the state makes a transition. Each state transition is determined by the value of the TMS signal at the rising edge of the TCK signal. In addition, the state name ending with -DR controls the data register, and the state name ending with -IR controls the instruction register.

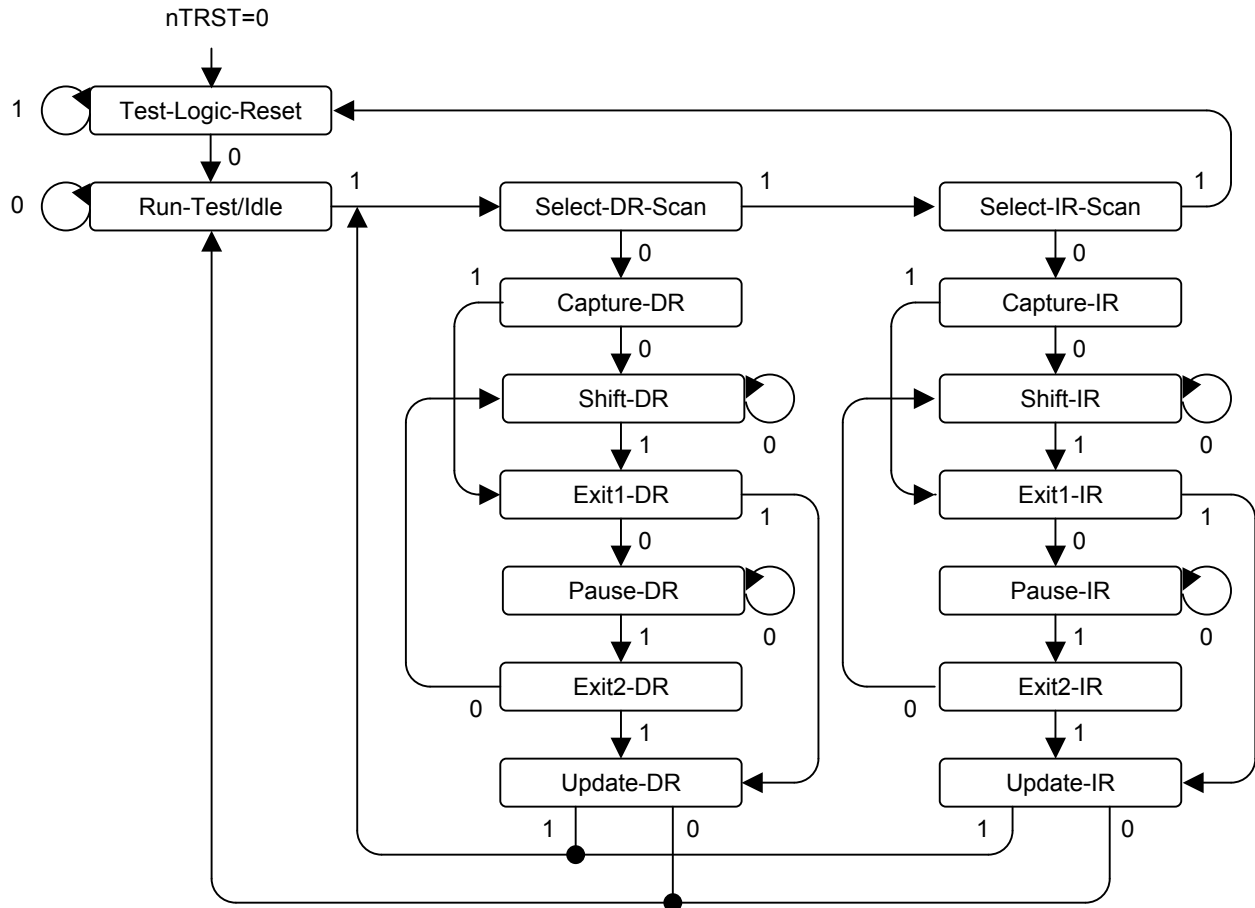


Figure 23.4 TAP Controller State Transition Diagram

### 23.3.4 Instructions

The instructions supported by the boundary scan function of this LSI are as follows. The INTEST and RUNBIST instructions, which are optional instructions in the JTAG standard, are not supported.

Instruction	Register to be selected	Instruction code		
		IR2	IR1	IR0
EXTEST	Boundary scan register	0	0	0
SAMPLE	Boundary scan register	0	1	0
IDCODE	ID register	0	0	1
BYPASS	Bypass register	1	1	1

#### EXTEST

The EXTEST instruction is used for testing the connection status at board level. It inputs the output signals of the boundary scan cells of the device in the previous step into the input boundary scan cells in the Capture-DR state.

It outputs the data in the output boundary cells to the output pins in the Update-DR state. Because the core logic circuit and the boundary scan cells are disconnected, the data that has been input is not transmitted to the core logic circuit, but is input into the input boundary scan cells. The data, which has been input into the input boundary scan cells, can be output from the TDO pin by repeating a shift operation.

#### SAMPLE

The SAMPLE instruction is used to sample the status of input/output pins during a normal operation. It inputs input/output signals into the input and output boundary scan cells in the Capture-DR state. The data that has been input is latched inside the boundary scan cells in the Update-DR state. Unlike the EXTEST instruction, the core logic circuit and the boundary scan cells are connected in the case of the SAMPLE instruction. However, the status of input/output data can be sampled during a normal operation via the boundary scan register, without affecting system operations adversely. The data, which has been input into the boundary scan cells, can be output from the TDO pin by repeating a shift operation.

#### IDCODE

The IDCODE instruction is used to select an ID register in which device information including serial numbers and parts numbers is stored. Using this instruction, whether or not correct parts are mounted on the board can be verified by testing.

The IDCODE of ML674001 series is "0000 0000 0010 0000 0000 0000 0101 1101".

From MSB, version(000), parts No.(0000 0010 0000 0000), Manufacturer ID(0000 0101 110), 1(fixed value)

The IDCODE of ML675001 series is undefined.

#### BYPASS

The BYPASS instruction is used to shorten the serial boundary scan chain if it is not necessary to test this LSI when testing the connection status at board level. Only the 1-bit BYPASS register is connected between the TDI and TDO pins. By setting BYPASS mode, the test data, which has been input to the TDI pin, is output to the TDO pin without routing through the boundary scan chain.

# **Electrical Characteristics**

---

## Chapter 24 Electrical Characteristics

### 24.1 Absolute Maximum Ratings

Item	Symbol	Conditions	Rating	Unit
Digital power supply voltage (core)	$V_{DD\_CORE}$	GND = AGND = 0 V PLL GND *1 = 0 V Ta = 25°C	-0.3 to +3.6	V
Digital power supply voltage (I/O)	$V_{DD\_IO}$		-0.3 to +4.6	
PLL power supply voltage*1	$V_{DD\_PLL}$		-0.3 to +3.6	
Input voltage	$V_I$		-0.3 to $V_{DD\_IO}+0.3$	
Output voltage	$V_O$		-0.3 to $V_{DD\_IO}+0.3$	
Analog power supply voltage	$AV_{DD}$		-0.3 to $V_{DD\_IO}+0.3$	
Analog reference voltage	$V_{REF}$		-0.3 to $V_{DD\_IO}+0.3$ and -0.3 to $AV_{DD} +0.3$	
Analog input voltage	$V_{AI}$		-0.3 to $V_{REF}$	
Input current	$I_I$	Ta = 85°C per package	-10 to +10	mA
Output current *2	$I_O$		-20 to +20	
Output current *3			-30 to +30	
Power losses	$P_D$		680(LFBGA) 1000(LQFP)	mW
Storage temperature	$T_{STG}$	—	-50 to +150	°C

**Note**

1. ML675001 Series only
2. All output pins except XA[15:0]
3. XA[15:0]

## 24.2 Operating Conditions

(GND = 0 V)

Item	Symbol	Conditions	Minimum	Typical	Maximum	Unit
Digital power supply voltage (core)	$V_{DD\_CORE}$	$V_{DD\_IO} \geq V_{DD\_CORE}$	2.25	2.5	2.75	V
Digital power supply voltage (I/O)	$V_{DD\_IO}$		3.0	3.3	3.6	
PLL power supply voltage *1	$V_{DD\_PLL}$	$V_{DD\_PLL} = V_{DD\_CORE}$	2.25	2.5	2.75	
Analog power supply voltage	$A_{VDD}$	$A_{VDD} = V_{DD\_IO}$	3.0	3.3	3.6	
Analog reference voltage	$V_{REF}$	$V_{REF} = A_{VDD} = V_{DD\_IO}$	3.0	3.3	3.6	
Operating frequency for ML674001 Series *2	$f_{OSC}$	$V_{DD\_CORE} = 2.25$ to $2.75$ $V_{DD\_IO} = 3.0$ to $3.6$	1 *4	—	33.333	MHz
Operating frequency for ML675001 Series *3	$f_{OSC}$	$V_{DD\_CORE} = 2.25$ to $2.75$ $V_{DD\_IO} = 3.0$ to $3.6$	1 *4	—	60	MHz
Ambient temperature	$T_a$	—	-40	25	85	°C

**Note**

1. Applied for ML675001 Series.
2. Crystal frequencies between 16 MHz and 33 MHz for ML674001 Series.
3. Crystal frequencies between 5 MHz and 14 MHz for ML675001 Series.
4. Minimum of 2.56 MHz for external SDRAM. Minimum of 6.4 MHz for external EDO DRAM. Minimum of 2 MHz for analog-to-digital converter.

## 24.3 Electrical Characteristics

### 24.3.1 DC Characteristics for ML674001 Series

( $V_{DD\_CORE} = 2.25$  to  $2.75V$ ,  $V_{DD\_IO} = 3.0$  to  $3.6V$ ,  $T_a = -40$  to  $+85^{\circ}C$ )

Item	Symbol	Conditions	Minimum	Typical	Maximum	Unit
High level input voltage	$V_{IH}$	—	$V_{DD\_IO} \times 0.8$	—	$V_{DD\_IO} + 0.3$	V
Low level input voltage	$V_{IL}$		-0.3	—	$V_{DD\_IO} \times 0.2$	
Schmitt Input Buffer Threshold Voltage	$V_{T+}$		—	1.6	2.1	
	$V_{T-}$		0.7	1.1	—	
	$V_{HYS}$		0.4	0.5	—	
High level output voltage	$V_{OH}$	$I_{OH} = -100 \mu A$	$V_{DD} - 0.2$	—	—	
		$I_{OH} = -4 mA$	2.35	—	—	
Low level output voltage	$V_{OL}$	$I_{OL} = 100 \mu A$	—	—	0.2	
Low level output voltage *1		$I_{OL} = 4 mA$	—	—	0.45	
Low level output voltage *2		$I_{OL} = 6 mA$	—	—	0.45	
Input leak current *3	$I_{IH} / I_{IL}$	$V_I = 0V / V_{DD\_IO}$	-50	—	50	μA
Input leak current *4	$I_{IL}$	$V_I = 0V$ Pull-up resistance of 50 kΩ	-200	-66	-10	
Input leak current *5	$I_I$	$V_I = AV_{DD} / 0V$	-5	—	5	
Output leak current	$I_{LO}$	$V_O = 0V / V_{DD\_IO}$	-50	—	50	
Input pin capacitance	$C_I$	—	—	6	—	pF
Output pin capacitance	$C_O$	—	—	9	—	
I/O pin capacitance	$C_{IO}$	—	—	10	—	
Analog reference power supply current	$I_{REF}$	Analog-to-digital converter operative*6	—	320	650	μA
		Analog-to-digital converter stopped	—	1	2	
Current consumption (STANDBY)	$I_{DD\_CORE}$	$T_a = 25^{\circ}C$ *7	—	20	100	
	$I_{DD\_IO}$			5	20	
Current consumption (HALT) *8	$I_{DDH\_CORE}$	$f_{OSC} = 33 MHz$ $CL = 30pF$	—	20	40	mA
	$I_{DDH\_IO}$			5	10	
Current consumption (RUN) *9	$I_{DD\_CORE}$			—	40	
	$I_{DD\_IO}$	—	18	30		

#### Notes

- All output pins except XA[15:0]
- XA[15:0]
- All input pins except RESET\_N
- RESET\_N pin, with 50 kΩ pull-up resistance
- Analog input pins (AIN0 to AIN3)
- Analog-to-Digital Converter operation ratio is 20%
- $V_{DD\_IO}$  or 0 V for input ports; no load for other pins
- DRAM Controller blocks stopped by DRAME\_N pin setting
- External ROM used

24.3.2 DC Characteristics for ML675001 Series

( $V_{DD\_CORE} = 2.25$  to  $2.75V$ ,  $V_{DD\_IO} = 3.0$  to  $3.6V$ ,  $T_a = -40$  to  $+85^{\circ}C$ )

Item	Symbol	Conditions	Minimum	Typical	Maximum	Unit	
High level input voltage	$V_{IH}$	—	$V_{DD\_IO} \times 0.8$	—	$V_{DD\_IO} + 0.3$	V	
Low level input voltage	$V_{IL}$		-0.3	—	$V_{DD\_IO} \times 0.2$		
Schmitt Input Buffer Threshold Voltage	$V_{T+}$		—	—	1.6		2.1
	$V_{T-}$		0.7	—	1.1		—
	$V_{HYS}$		0.4	—	0.5		—
High level output voltage	$V_{OH}$		$I_{OH} = -100 \mu A$	$V_{DD} - 0.2$	—		—
		$I_{OH} = -4 mA$	2.35	—	—		
Low level output voltage	$V_{OL}$	$I_{OL} = 100 \mu A$	—	—	0.2		
Low level output voltage *1		$I_{OL} = 4 mA$	—	—	0.45		
Low level output voltage *2		$I_{OL} = 6 mA$	—	—	0.45		
Input leak current *3	$I_{IH} / I_{IL}$	$V_I = 0V / V_{DD\_IO}$	-50	—	50	$\mu A$	
Input leak current *4	$I_{IL}$	$V_I = 0V$ Pull-up resistance of 50 k $\Omega$	-200	-73	-10		
Input leak current *5	$I_I$	$V_I = AV_{DD} / 0V$	-5	—	5		
Output leak current	$I_{LO}$	$V_O = 0V / V_{DD\_IO}$	-50	—	50		
I/O pin capacitance	$C_{IO}$	—	—	10	—		
Analog reference power supply current	$I_{REF}$	Analog-to-digital converter operative*6	—	320	650	$\mu A$	
		Analog-to-digital converter stopped	—	1	2		
Current consumption (STANDBY)	$I_{DDS\_CORE}$	$T_a = 25^{\circ}C$ *7	—	20	150	$\mu A$	
	$I_{DDS\_IO}$			10	40		
Current consumption (HALT) *8	$I_{DDH\_CORE}$	$f_{OSC} = 60 MHz$ $CL = 30pF$	—	37	55	mA	
	$I_{DDH\_IO}$			6	10		
Current consumption (RUN) *9	$I_{DD\_CORE}$			—	75		120
	$I_{DD\_IO}$			—	17		25

**Notes**

1. All output pins except XA[15:0]
2. XA[15:0]
3. All input pins except RESET\_N
4. RESET\_N pin, with 50 k $\Omega$  pull-up resistance
5. Analog input pins (AIN0 to AIN3)
6. Analog-to-Digital Converter operation ratio is 20%
7.  $V_{DD\_IO}$  or 0 V for input ports; no load for other pins
8. DRAM Controller blocks stopped by DRAME\_N pins settings
9. Cacheable setting and External ROM used



## 24.4 AC Characteristics

### 24.4.1 AC Characteristics for ML674001 Series

#### ■ Power Supply On/OFF Timing

( $V_{DD\_CORE} = 2.25$  to  $2.75V$ ,  $V_{DD\_IO} = 3.0$  to  $3.6V$ ,  $T_a = -40$  to  $+85^{\circ}C$ )

Item	Symbol	Condi- tions	Minimum	Typical	Maximum	Unit	Notes
AVDD Supply on Delay	$t_{AVDD\_ON}$		0	—	—	ns	
VDDcore Supply on Delay	$t_{VDDCORE\_ON}$		0	—	—	ns	
AVDD Supply off Delay	$t_{AVDD\_OFF}$		0	—	—	ns	
VDDio Supply off Delay	$t_{VDDIO\_OFF}$		0	—	—	ns	

■ Clock Timing

( $V_{DD\_CORE} = 2.25$  to  $2.75V$ ,  $V_{DD\_IO} = 3.0$  to  $3.6V$ ,  $T_a = -40$  to  $+85^{\circ}C$ )

Item	Symbol	Condi- tions	Minimum	Typical	Maximum	Unit	Notes
Input Clock frequency	$f_C$	—	1	—	33.333	MHz	For external clock input
Input Clock cycle time	$t_C$		30	—	1000	ns	
Input Clock High level pulse width	$t_{CH}$		14	—	—		
Input Clock Low level pulse width	$t_{CL}$		14	—	—		
Input Clock Rise time	$t_{CR}$		—	—	4		
Input Clock Fall time	$t_{CF}$		—	—	4		
XSDCLK frequency	$f_{SDC}$	—	1	—	33.333	MHz	The same frequency as HCLK
XSDCLK cycle time	$t_{SDC}$		30	—	1000	ns	
XSDCLK High level pulse width	$t_{SDCH}$	CL=30 pF	12	—	—		
XSDCLK Low level pulse width	$t_{SDCL}$		12	—	—		
XSDCLK Rise time	$t_{SDCR}$		—	—	2		
XSDCLK Fall time	$t_{SDCF}$		—	—	2		
HCLK frequency	$f_{HC}$		—	0.125 <sup>*1</sup>	—	33.333	MHz
HCLK cycle time	$t_{HC}$	30		—	8000 <sup>*1</sup>	ns	
CCLK frequency	$f_{CC}$	—	0.125 <sup>*2</sup>	—	33.333	MHz	
CCLK cycle time	$t_{CC}$		30	—	8000 <sup>*2</sup>	ns	
CKO frequency	$f_{CK}$	—	0.125 <sup>*1</sup>	—	33.333	MHz	The same frequency as HCLK
CKO cycle time	$t_{CK}$		30	—	8000 <sup>*1</sup>	ns	
CKO High level pulse width	$t_{CKH}$	CL=30 pF	12	—	—		
CKO Low level pulse width	$t_{CKL}$		12	—	—		
CKO Rise time	$t_{CKR}$		—	—	2		
CKO Fall time	$t_{CKF}$		—	—	2		

**Notes**

1. Minimum of 2.56 MHz / Maximum of 390.6ns for external SDRAM.  
 Minimum of 6.4 MHz / Maximum of 156ns for external EDO DRAM.
2. Minimum of 2 MHz / Maximum of 500ns for analog-to-digital converter.

■ Reset/ Interrupt/ DMA Timing

( $V_{DD\_CORE} = 2.25$  to  $2.75V$ ,  $V_{DD\_IO} = 3.0$  to  $3.6V$ ,  $T_a = -40$  to  $+85^{\circ}C$ )

Item	Symbol	Condi- tions	Minimum	Typical	Maximum	Unit	Notes
RESET_N pulse width 1	$t_{RSTW1}$	—	$20 t_{HC}$	—	—	ns	Except for when power is first applied or returning from STANDBY mode.
RESET_N pulse width 2	$t_{RSTW2}$		Oscillation stabilization interval	—	—	—	When power is first applied or returning from STANDBY mode.
EFIQ_N pulse width	$t_{EFIQW}$		$2 t_{HC}$	—	—	ns	Except for STANDBY mode Release from STANDBY mode
EXINT pulse width 1	$t_{EXINTW1}$		$2 t_{HC}$	—	—		
EXINT pulse width 2	$t_{EXINTW2}$		$t_{HC}$	—	—		
DREQCLR0/DREQCLR1 delay 1	$t_{DCLR1}$	CL = 30 pF	$8 t_{HC} + 10.5$	—	—		
TCOUT0/TCOUT1 delay 1	$t_{TCOUT1}$		$8 t_{HC} + 10.5$	—	—		
DREQCLR0/DREQCLR1 delay 2	$t_{DCLR2}$		$2 t_{HC} + 11$	—	—		
TCOUT0/TCOUT1 delay 2	$t_{TCOUT2}$		$2 t_{HC} + 11$	—	—		
DREQ0/DREQ1 hold time	$t_{DREQH}$	—	$t_{HC}$	—	—		

■ SRAM/ROM Control signal timing

( $V_{DD\_CORE} = 2.25$  to  $2.75V$ ,  $V_{DD\_IO} = 3.0$  to  $3.6V$ ,  $T_a = -40$  to  $+85^{\circ}C$ )

SRAM/ROM							
Item	Symbol	Condi-tions	Minimum	Typical	Maximum	Unit	Notes
XROMCS_N/XRAMCS_N access time1 (SRAM/ROM) READ ACCESS	$t_{CSR1}$	CL = 30 pF	$(n_{R1} + n_{R3}) t_{HC} - 2$	—	$(n_{R1} + n_{R3}) t_{HC} + 2$	ns	The ROMAC and RAMAC registers specify the OE/WE pulse width and read off time for ROM and SRAM access, respectively. For further details, refer to sections 11.2.3 and 11.2.4 of chapter 11.  $n_{R1} = 0$ (CPU Access) $n_{R1} = 1$ (DMA Access)) $n_{R2} = 0.5$ (CPU Access) $n_{R2} = 1.5$ (DMA Access) $n_{R3} =$ (OE/WE pulse width) $n_{R4} =$ (OE/WE pulse width)+ 0.5 $n_{R5} = n_{R2} +$ (OE/WE pulse width) + 0.5
XROMCS_N/XRAMCS_N access time2 (SRAM/ROM) READ ACCESS	$t_{CSR2}$		$2*(n_{R1} + n_{R3}) t_{HC} - 2$	—	$2*(n_{R1} + n_{R3}) t_{HC} + 2$		
XA[23:0] access time (SRAM/ROM)	$t_{ACC}$		$(n_{R1} + n_{R3}) t_{HC} - 2.5$	—	$(n_{R1} + n_{R3}) t_{HC} + 0.5$		
XBS_N[1:0] access time (SRAM/ROM)	$t_{BS}$		$(n_{R1} + n_{R3}) t_{HC} - 2.5$	—	$(n_{R1} + n_{R3}) t_{HC} + 3$		
XOE_N delay (SRAM/ROM)	$t_{OED}$		$n_{R1} t_{HC} - 1$	—	$n_{R1} t_{HC} + 3$		
XWE_N delay (SRAM/ROM)	$t_{WED}$		$n_{R2} t_{HC} - 3$	—	$n_{R2} t_{HC} + 2$		
XROMCS_N/XRAMCS_N access time1 (SRAM/ROM) WRITE ACCESS	$t_{CSW1}$		$(n_{R2} + n_{R3}) t_{HC} - 5$	—	$(n_{R2} + n_{R3}) t_{HC} + 2.5$		
XROMCS_N/XRAMCS_N access time2 (SRAM/ROM) WRITE ACCESS	$t_{CSW2}$		$(n_{R5} + n_{R2} + n_{R3}) t_{HC} - 3.5$	—	$(n_{R5} + n_{R2} + n_{R3}) t_{HC}$		
XBWE_N[1:0] delay (SRAM/ROM)	$t_{WELHD}$		$n_{R2} t_{HC} - 4$	—	$n_{R2} t_{HC} + 2$		
XBWE_N[1:0] hold time (SRAM/ROM)	$t_{WELHH}$		-0.5	—	0.5		
XOE_N, XWE_N pulse width (SRAM/ROM)	$t_{OE/WEW}$		$n_{R3} t_{HC} - 4$	—	$n_{R3} t_{HC} + 2$		
XOE_N pulse width (SRAM/ROM) ARM ACCESS	$t_{OEW}$		$2* n_{R3} t_{HC} - 4$	—	$2* n_{R3} t_{HC} + 1$		
XBS_N[1:0] delay (SRAM/ROM)	$t_{BSD}$		-3	—	2		
XBS_N[1:0] output hold time 1 (SRAM/ROM)	$t_{BSH1}$		$n_{R1} t_{HC} - 2$	—	—		
XBS_N[1:0] output hold time 2 (SRAM/ROM)	$t_{BSH2}$		$0.5 t_{HC} - 2$	—	—		
XA[23:0] delay (SRAM/ROM)	$t_{XAD}$		-0.5	—	6		
XA[23:0] output hold time 1 (SRAM/ROM)	$t_{XAH1}$		$n_{R1} t_{HC}$	—	—		
XA[23:0] output hold time 2 (SRAM/ROM)	$t_{XAH2}$		$0.5 t_{HC}$	—	—		

SRAM/ROM (continued)							
Item	Symbol	Condi-tions	Minimum	Typical	Maximum	Unit	Notes
XD[15:0] input setup time (SRAM/ROM) ARM ACCESS	$t_{XDIS}$	—	19	—	—	ns	
XD[15:0] input setup time (SRAM/ROM) DMAC ACCESS	$t_{XDIS}$		17	—	—		
XD[15:0] input hold time (SRAM/ROM)	$t_{XDIH}$		0	—	—		
XD[15:0] output delay (SRAM/ROM) ARM ACCESS	$t_{XDOD}$	CL = 30 pF	$n_{R4} t_{HC} - 14.5$	—	—		
XD[15:0] output delay (SRAM/ROM) DMAC ACCESS	$t_{XDOD}$		$n_{R4} t_{HC} - 7.5$	—	—		
XD[15:0] output Enable time (SRAM/ROM)	$t_{XDOE}$		$n_{R1} t_{HC} + 1.5$	—	—		
XD[15:0] output Disable time (SRAM/ROM)	$t_{XDODE}$		-6	—	—		
XD[15:0] output hold time (SRAM/ROM) ARM ACCESS	$t_{XDOH}$		$0.5 t_{HC} + 3$	—	—		
XD[15:0] output hold time (SRAM/ROM) DMAC ACCESS	$t_{XDOH}$		$0.5 t_{HC} + 5.5$	—	—		
XROMCS_N, XRAMCS_N output hold time 1 (SRAM/ROM)	$t_{CSH1}$		$n_{R1} t_{HC} - 1.5$	—	—		
XROMCS_N, XRAMCS_N Output hold time 2 (SRAM/ROM)	$t_{CSH2}$		$0.5 t_{HC} + 0.5$	—	—		

■ I/O Control signal timing

( $V_{DD\_CORE} = 2.25$  to  $2.75V$ ,  $V_{DD\_IO} = 3.0$  to  $3.6V$ ,  $T_a = -40$  to  $+85^{\circ}C$ )

IO 0/ IO 1/ IO 2/ IO 3									
Item	Symbol	Condi- tions	Minimum	Typical	Maximum	Unit	Notes		
XIOCS_N[0]/XIOCS_N[1] access time 1 (external I/O 0,1,2, and 3) READ ACCESS	$t_{XIOCSR1}$	—	$n_{IO1} t_{HC} - 5$	—	$n_{IO1} t_{HC} + 1$	ns	The IO01AC,IO23ACX,and IO23ACY registers specify the OE/WE pulse width and read off time for accessing external I/O 0, 1, 2, and 3, respectively. For further details, refer to sections 11.2.5 and 11.2.6 of chapter 11.		
XIOCS_N[0]/XIOCS_N[1] access time 2 (external I/O 0,1,2, and 3) READ ACCESS	$t_{XIOCSR2}$		$2*n_{IO1} t_{HC} - 5$	—	$2*n_{IO1} t_{HC} + 1$				
XA[23:0] access time (external I/O 0,1,2, and 3)	$t_{XIOACC}$		$n_{IO1} t_{HC} - 5$	—	$n_{IO1} t_{HC} + 0.5$				
XBS_N[1:0] access time (external I/O 0,1,2, and 3)	$t_{XIOBS}$		$n_{IO1} t_{HC} - 3$	—	$n_{IO1} t_{HC} + 1.5$				
XWR delay (external I/O 0,1,2, and 3)	$t_{XIOWRD}$	CL = 30 pF	0	—	3.5			ns	$n_{IO1} = (\text{Address Setup}) +$ $(\text{OE/WE Pulse width})$ $n_{IO2} = (\text{Address Setup}) - 1$ $+ (\text{OE/WE Pulse}$ $\text{width})$ $n_{IO3} = (\text{Address Setup})$ $n_{IO4} = (\text{Address Setup}) + 1$ $n_{IO5} = (\text{OE/WE pulse width})$ $n_{IO6} = (\text{Address Setup}) + 1$ $+ (\text{OE/WE Pulse width})$ $n_{IO7} = (\text{Address Setup}) + 2$ $+ (\text{OE/WE Pulse width})$
XWR hold time (external I/O 0,1,2, and 3)	$t_{XIOWRH}$		$t_{HC} - 1$	—	$t_{HC} + 3$				
XWAIT sampling timing delay 1 (external I/O 0,1,2, and 3)	$t_{XIOWAITD1}$	—	$n_{IO2} t_{HC}$	—	$n_{IO2} t_{HC}$				
XWAIT sampling timing delay 2 (external I/O 0,1,2, and 3)	$t_{XIOWAITD2}$		$n_{IO1} t_{HC}$	—	$n_{IO1} t_{HC}$				
XWAIT setup time (external I/O 0,1,2, and 3)	$t_{XIOWAITS}$		20	—	—				
XWAIT hold time (external I/O 0,1,2, and 3)	$t_{XIOWAITH}$		0	—	—				
XOE_N delay (external I/O 0,1,2, and 3)	$t_{XIOOED}$		$n_{IO3} t_{HC} - 2$	—	$n_{IO3} t_{HC} + 1$				
XWE_N delay 1 (external I/O 0,1,2, and 3)	$t_{XIOWED}$	CL = 30 pF	$n_{IO4} t_{HC} - 1.5$	—	$n_{IO4} t_{HC} + 1.5$				
XIOCS_N[0]/XIOCS_N[1] access time 1(external I/O 0,1,2, and 3) WRITE ACCESS	$t_{XIOCSW1}$		$n_{IO6} t_{HC} - 3$	—	$n_{IO6} t_{HC} + 3$				
XIOCS_N[0]/XIOCS_N[1] access time 2(external I/O 0,1,2, and 3) WRITE ACCESS	$t_{XIOCSW2}$		$(n_{IO6}+n_{IO7})t_{HC}$ $- 3$	—	$(n_{IO6}+n_{IO7})t_{HC}$ $+ 3$				

IO 0/ IO 1/ IO 2/ IO 3 (continued)							
Item	Symbol	Condi- tions	Minimum	Typical	Maximum	Unit	Notes
XBWE_N[1:0] delay (external I/O 0,1,2, and 3)	$t_{XIOVELHD}$	CL = 30 pF	$n_{IO4} t_{HC} - 2$	—	$n_{IO4} t_{HC} + 1$	ns	
XOE_N, XWE_N pulse width (external I/O 0,1,2, and 3)	$t_{XIOOE/WEW}$		$n_{IO5} t_{HC} - 1.5$	—	$n_{IO5} t_{HC} + 1.5$		
XBWE_N[1:0] hold time (external I/O 0,1,2, and 3)	$t_{XIOVELHH}$		-1		0.5		
XBS_N[1:0] delay (external I/O 0,1,2, and 3)	$t_{XIOBSD}$		-4	—	2		
XBS_N[1:0] output hold time (external I/O 0,1,2, and 3)	$t_{XIOBSH}$		$t_{HC} - 2$	—	—		
XA[23:0] delay (external I/O 0,1,2, and 3)	$t_{XIOXAD}$	—	-0.5	—	2		
XA[23:0] output hold time 1 (external I/O 0,1,2, and 3)	$t_{XIOXAH1}$		0	—	—		
XA[23:0] output hold time 2 (external I/O 0,1,2, and 3)	$t_{XIOXAH2}$		$t_{HC} - 1$	—	—		
XD[15:0] input setup time (external I/O 0,1,2, and 3)	$t_{XIODIS}$		20	—	—		
XD[15:0] input hold time (external I/O 0,1,2, and 3)	$t_{XIODIH}$		0	—	—		
XD[15:0] output delay (external I/O 0,1,2, and 3)	$t_{XIODOD}$	CL = 30 pF	$n_{IO1} t_{HC} - 5$	—	$n_{IO1} t_{HC} - 2$		
XD[15:0] output Enable time (external I/O 0,1,2, and 3)	$t_{XIODOE}$		$t_{HC} + 2$	—	—		
XD[15:0] output Disable time (external I/O 0,1,2, and 3)	$t_{XIODODE}$		-6	—	—		
XD[15:0] output hold time (external I/O 0,1,2, and 3)	$t_{XIODOH}$		$t_{HC} + 3$	—	—		
XIOCS_N[1:0] output hold time (external I/O 0,1,2, and 3)	$t_{XIOCSH}$		$t_{HC} - 2$	—	—		

■ DRAM Access Timing

( $V_{DD\_CORE} = 2.25$  to  $2.75V$ ,  $V_{DD\_IO} = 3.0$  to  $3.6V$ ,  $T_a = -40$  to  $+85^{\circ}C$ )

SDRAM								
Item	Symbol	Condi-tions	Minimum	Typical	Maximum	Unit	Notes	
XSDCS_N delay (SDRAM)	$t_{SDCSD}$	CL = 30 pF	$0.5t_{SDC}$	—	$0.5t_{SDC} + 5$	ns	The DRPC register specifies the SDRAM access parameters $t_{RAS}$ , $t_{RCD}$ , $t_{RP}$ , $t_{DPL}$ . For further details, refer to section 11.2.9 of chapter 11.	
XSDCKE delay (SDRAM)	$t_{SDCKED}$		$0.5t_{SDC}$	—	$0.5t_{SDC} + 5$			
XDQM[0]/XDQM[1] delay (SDRAM)	$t_{SDDQMD}$		$0.5t_{SDC}$	—	$0.5t_{SDC} + 5.5$			
XRAS_N delay (SDRAM)	$t_{SDRASD}$		$0.5t_{SDC}$	—	$0.5t_{SDC} + 5.5$			$n_{SD1} = t_{RCD}$
XCAS_N delay (SDRAM)	$t_{SDCASD}$		$0.5t_{SDC}$	—	$0.5t_{SDC} + 4.5$			$n_{SD2} = t_{RAS}$
RASCAS minimum delay (SDRAM)	$t_{SDRCD}$	—	$n_{SD1}t_{SDC}$	—	—		$n_{SD3} = t_{RP}$	
RAS active time (SDRAM)	$t_{SDRAS}$		$n_{SD2}t_{SDC}$	—	—			
RAS precharge time (SDRAM)	$t_{SDRP}$		$n_{SD3}t_{SDC}$	—	—			
XWE_N delay (SDRAM)	$t_{SDWED}$	CL = 30 pF	$0.5t_{SDC}$	—	$0.5t_{SDC} + 4$			
XD[15:0] input setup time (SDRAM)	$t_{SDXDIS}$	—	14	—	—			
XD[15:0] input hold time (SDRAM)	$t_{SDXDIH}$		0	—	—			
XA[15:0] output delay (SDRAM)	$t_{SDXAD}$	CL = 30 pF	$0.5t_{SDC}$	—	$0.5t_{SDC} + 5$			
XD[15:0] output delay (SDRAM)	$t_{SDXDOD}$		$0.5t_{SDC}$	—	$0.5t_{SDC} + 6$			
XD[15:0] output hold time (SDRAM)	$t_{SDXDOH}$		$0.5t_{SDC}$	—	—			
XD[15:0] output enable time (SDRAM)	$t_{SDXDOE}$		$0.5t_{SDC}$	—	—			
XD[15:0] output disable time (SDRAM)	$t_{SDXDODE}$		—	—	$0.5t_{SDC} + 8$			



( $V_{DD\_CORE} = 2.25$  to  $2.75V$ ,  $V_{DD\_IO} = 3.0$  to  $3.6V$ ,  $T_a = -40$  to  $+85^{\circ}C$ )

EDODRAM								
Item	Symbol	Condi-tions	Minimum	Typical	Maximum	Unit	Notes	
RASCAS delay (EDO DRAM)	$t_{EDRCD}$	—	$n_{ED1} t_{HC} - 2$	—	$n_{ED1} t_{HC} + 2$	ns	The DRPC register specifies the EDO DRAM access parameters $t_{RAH}$ , $t_{CAC}$ , $t_{CAS}$ , $t_{RCD}$ , $t_{RP}$ . For further details, refer to section 11.2.9 of chapter 11	
CAS pulse width (EDO DRAM)	$t_{EDCAS}$	CL = 30 pF	$n_{ED2} t_{HC} - 4$	—	$n_{ED2} t_{HC} + 2$			
RAS pulse width (EDO DRAM)	$t_{EDRAS}$		$n_{ED7} t_{HC} - 4$	—	$n_{ED7} t_{HC} + 1$			
RAS precharge time (EDO DRAM)	$t_{EDRP}$	—	$n_{ED3} t_{HC} - 4$	—	—			$n_{ED1} = t_{RCD}$
CAS precharge time (EDO DRAM)	$t_{EDCP}$		$n_{ED4} t_{HC} - 4$	—	—			$n_{ED2} = t_{CAS}$
XRAS_N delay (EDO DRAM)	$t_{EDRASD}$	CL = 30 pF	$t_{HC} - 2$	—	$t_{HC} + 2$			$n_{ED3} = t_{RP}$
XOE_N delay 1 (EDO DRAM)	$t_{EDOED1}$		$n_{ED5} t_{HC} - 4$	—	$n_{ED5} t_{HC}$			$n_{ED4} = t_{CAC} + 1 - t_{CAS}$
XOE_N delay 2 (EDO DRAM)	$t_{EDOED2}$		$n_{ED4} t_{HC} - 2$	—	$n_{ED4} t_{HC}$			$n_{ED5} = t_{RAH}$
XWE_N delay 1 (EDO DRAM)	$t_{EDWED1}$		$n_{ED5} t_{HC} - 4$	—	$n_{ED5} t_{HC} + 2$			$n_{ED6} = t_{CAC} + 1$
XWE_N delay 2 (EDO DRAM)	$t_{EDWED2}$		$n_{ED4} t_{HC} - 2$	—	$n_{ED4} t_{HC}$			$n_{ED7} = t_{RCD} + N(t_{CAC} + 1)$
Row address hold time (EDO DRAM)	$t_{EDRAH}$		$n_{ED5} t_{HC} - 2$	—	$n_{ED5} t_{HC} + 4$			N = Data Size / Bus Width
Column address delay (EDO DRAM)	$t_{EDCAD}$		$t_{HC} - 2$	—	$t_{HC} + 4$			
Column address hold time (EDO DRAM)	$t_{EDCAH}$		$n_{ED2} t_{HC} - 4$	—	$n_{ED2} t_{HC} + 2$			
XD[15:0] sampling timing delay (EDO DRAM)	$t_{EDXDSMPLD}$		$n_{ED6} t_{HC}$	—	$n_{ED6} t_{HC}$			
XD[15:0] input setup time (EDO DRAM)	$t_{EDXDIS}$		—	16	—			—
XD[15:0] input hold time (EDO DRAM)	$t_{EDXDIH}$	—	0	—	—			
XD[15:0] output delay 1 (EDO DRAM)	$t_{EDXDOD1}$	CL = 30 pF	0	—	6			
XD[15:0] output delay 2 (EDO DRAM)	$t_{EDXDOD2}$		0.5	—	6			
XD[15:0] output hold time (EDO DRAM)	$t_{EDXDOH}$		$n_{ED2} t_{HC} - 2$	—	—			
XD[15:0] output enable time (EDO DRAM)	$t_{EDXDOE}$		$-(n_{ED5} + 1) t_{HC} + 1$	—	—			
XD[15:0] output disable time (EDO DRAM)	$t_{EDXDODE}$		—	—	$(n_{ED2} + n_{ED4}) t_{HC} + 0.5$			

24.4.2 AC Characteristics for ML675001 Series

■ Power Supply On/OFF Timing

( $V_{DD\_CORE} = 2.25$  to  $2.75V$ ,  $V_{DD\_IO} = 3.0$  to  $3.6V$ ,  $T_a = -40$  to  $+85^{\circ}C$ )

Item	Symbol	Condi- tions	Minimum	Typical	Maximum	Unit	Notes
AVDD Supply on Delay	$t_{AVDD\_ON}$		0	—	—	ns	
VDD_pll Supply on Delay	$t_{VDDPLL\_ON}$		0	—	—	ns	
VDDcore Supply on Delay	$t_{VDDCORE\_ON}$		0	—	—	ns	
AVDD Supply off Delay	$t_{AVDD\_OFF}$		0	—	—	ns	
VDD_pll Supply off Delay	$t_{VDDPLL\_OFF}$		0	—	—	ns	
VDDio Supply off Delay	$t_{VDDIO\_OFF}$		0	—	—	ns	

■ Clock Timing

( $V_{DD\_CORE} = 2.25$  to  $2.75V$ ,  $V_{DD\_IO} = 3.0$  to  $3.6V$ ,  $T_a = -40$  to  $+85^{\circ}C$ )

Item	Symbol	Condi- tions	Minimum	Typical	Maximum	Unit	Notes
Input Clock frequency	$f_C$	—	5	—	56	MHz	For external clock input
Input Clock cycle time	$t_C$		17.9	—	200	ns	
Input Clock High level pulse width	$t_{CH}$		7	—	—		
Input Clock Low level pulse width	$t_{CL}$		7	—	—		
Input Clock Rise time	$t_{CR}$		—	—	4		
Input Clock Fall time	$t_{CF}$		—	—	4		
XSDCLK frequency	$f_{SDC}$	—	0.625	—	60	MHz	The same frequency as HCLK
XSDCLK cycle time	$t_{SDC}$	CL=30 pF	16.7	—	1600	ns	
XSDCLK High level pulse width	$t_{SDCH}$		7	—	—		
XSDCLK Low level pulse width	$t_{SDCL}$		7	—	—		
XSDCLK Rise time	$t_{SDCR}$		—	—	2		
XSDCLK Fall time	$t_{SDCF}$		—	—	2		
HCLK frequency	$f_{HC}$		—	0.625 <sup>*1</sup>	—		
HCLK cycle time	$t_{HC}$	—	16.7	—	1600 <sup>*1</sup>	ns	
CCLK frequency	$f_{CC}$	—	0.625 <sup>*2</sup>	—	60	MHz	
CCLK cycle time	$t_{CC}$	—	16.7	—	1600 <sup>*2</sup>	ns	
CKO frequency	$f_{CK}$	—	0.625	—	60	MHz	The same frequency as HCLK
CKO cycle time	$t_{CK}$	CL=30 pF	16.7	—	1600	ns	
CKO High level pulse width	$t_{CKH}$		6	—	—		
CKO Low level pulse width	$t_{CKL}$		6	—	—		
CKO Rise time	$t_{CKR}$		—	—	2		
CKO Fall time	$t_{CKF}$		—	—	2		

**Notes**

1. Minimum of 2.56 MHz / Maximum of 390.6ns for external SDRAM.  
Minimum of 6.4 MHz / Maximum of 156ns for external EDO DRAM.
2. Minimum of 2 MHz / Maximum of 500ns for analog-to-digital converter.
3. Refer to Chapter 5 about the relation of  $t_C$  and  $t_{HC}$ .

■ Reset/ Interrupt/ DMA Timing

( $V_{DD\_CORE} = 2.25$  to  $2.75V$ ,  $V_{DD\_IO} = 3.0$  to  $3.6V$ ,  $T_a = -40$  to  $+85^{\circ}C$ )

Item	Symbol	Condi- tions	Minimum	Typical	Maximum	Unit	Notes
RESET_N pulse width 1	$t_{RSTW1}$	—	$20 t_{HC}$	—	—	ns	Except for when power is first applied or returning from STANDBY mode.
RESET_N pulse width 2	$t_{RSTW2}$		Oscillation stabilization interval	—	—	—	When power is first applied or returning from STANDBY mode.
EFIQ_N pulse width	$t_{EFIQW}$		$2 t_{HC}$	—	—	ns	Except for STANDBY mode Release from STANDBY mode
EXINT pulse width 1	$t_{EXINTW1}$		$2 t_{HC}$	—	—		
EXINT pulse width 2	$t_{EXINTW2}$		$t_{HC}$	—	—		
DREQCLR0/DREQCLR1 delay 1	$t_{DCLR1}$	CL = 30 pF	$8 t_{HC} + 10.5$	—	—		
TCOUT0/TCOUT1 delay 1	$t_{TCOUT1}$		$8 t_{HC} + 10.5$	—	—		
DREQCLR0/DREQCLR1 delay 2	$t_{DCLR2}$		$2 t_{HC} + 11$	—	—		
TCOUT0/TCOUT1 delay 2	$t_{TCOUT2}$		$2 t_{HC} + 11$	—	—		
DREQ0/DREQ1 hold time	$t_{DREQH}$	—	$t_{HC}$	—	—		

■ SRAM/ROM Control signal timing

( $V_{DD\_CORE} = 2.25$  to  $2.75V$ ,  $V_{DD\_IO} = 3.0$  to  $3.6V$ ,  $T_a = -40$  to  $+85^{\circ}C$ )

SRAM/ROM							
Item	Symbol	Condi-tions	Minimum	Typical	Maximum	Unit	Notes
XROMCS_N/XRAMCS_N access time1 (SRAM/ROM) READ ACCESS	$t_{CSR1}$	CL = 30 pF	$(n_{R1} + n_{R2}) t_{HC} - 2$	—	$(n_{R1} + n_{R2}) t_{HC} + 2$	ns	The ROMAC and RAMAC registers specify the OE/WE pulse width and read off time for ROM and SRAM access, respectively. For further details, refer to sections 11.2.3 and 11.2.4 of chapter 11.  $n_{R1}$ = (address setup) $n_{R2}$ = (OE/WE pulse width)
XROMCS_N/XRAMCS_N access time2 (SRAM/ROM) READ ACCESS	$t_{CSR2}$		$2*(n_{R1} + n_{R2}) t_{HC} - 2$	—	$2*(n_{R1} + n_{R2}) t_{HC} + 2$		
XA[23:0] access time (SRAM/ROM)	$t_{ACC}$		$(n_{R1} + n_{R2}) t_{HC} - 2.5$	—	$(n_{R1} + n_{R2}) t_{HC} + 0.5$		
XBS_N[1:0] access time (SRAM/ROM)	$t_{BS}$		$(n_{R1} + n_{R2}) t_{HC} - 2.5$	—	$(n_{R1} + n_{R2}) t_{HC} + 3$		
XOE_N delay (SRAM/ROM)	$t_{OED}$		$n_{R1} t_{HC} - 1$	—	$n_{R1} t_{HC} + 3$		
XWE_N delay (SRAM/ROM)	$t_{WED}$		$n_{R1} t_{HC} - 3$	—	$n_{R1} t_{HC} + 2$		
XROMCS_N/XRAMCS_N access time1 (SRAM/ROM) WRITE ACCESS	$t_{CSW1}$		$(n_{R1} + n_{R2}) t_{HC} - 5$	—	$(n_{R1} + n_{R2}) t_{HC} + 2.5$		
XROMCS_N/XRAMCS_N access time2 (SRAM/ROM) WRITE ACCESS	$t_{CSW2}$		$2*(n_{R1} + n_{R2} + 0.5) t_{HC} - 3.5$	—	$2*(n_{R1} + n_{R2} + 0.5) t_{HC}$		
XBWE_N[1:0] delay (SRAM/ROM)	$t_{WELHD}$		$n_{R1} t_{HC} - 4$	—	$n_{R1} t_{HC} + 2$		
XBWE_N[1:0] hold time (SRAM/ROM)	$t_{WELHH}$		-0.5	—	0.5		
XOE_N, XWE_N pulse width (SRAM/ROM)	$t_{OE/WEW}$		$n_{R2} t_{HC} - 4$	—	$n_{R2} t_{HC} + 2$		
XOE_N pulse width (SRAM/ROM) ARM ACCESS	$t_{OEW}$		$2* n_{R2} t_{HC} - 4$	—	$2* n_{R2} t_{HC} + 1$		
XBS_N[1:0] delay (SRAM/ROM)	$t_{BSD}$		-3	—	2		
XBS_N[1:0] output hold time 1 (SRAM/ROM)	$t_{BSH1}$		0	—	—		
XBS_N[1:0] output hold time 2 (SRAM/ROM)	$t_{BSH2}$		$t_{HC} - 2$	—	—		
XA[23:0] delay (SRAM/ROM)	$t_{xAD}$		-0.5	—	6		
XA[23:0] output hold time 1 (SRAM/ROM)	$t_{XAH1}$	0	—	—			

SRAM/ROM (continued)							
Item	Symbol	Condi-tions	Minimum	Typical	Maximum	Unit	Notes
XA[23:0] output hold time 2 (SRAM/ROM)	$t_{XAH2}$		$t_{HC}$	—	—	ns	
XD[15:0] input setup time (SRAM/ROM) ARM ACCESS	$t_{XDIS}$	—	19	—	—		
XD[15:0] input setup time (SRAM/ROM) DMAC ACCESS	$t_{XDIS}$		17	—	—		
XD[15:0] input hold time (SRAM/ROM)	$t_{XDIH}$		0	—	—		
XD[15:0] output delay (SRAM/ROM) ARM ACCESS	$t_{XDOD}$	CL = 30 pF	$n_{R2} t_{HC} - 14.5$	—	—		
XD[15:0] output delay (SRAM/ROM) DMAC ACCESS	$t_{XDOD}$		$n_{R2} t_{HC} - 7.5$	—	—		
XD[15:0] output Enable time (SRAM/ROM)	$t_{XDOE}$		$n_{R1} t_{HC} + 1.5$	—	—		
XD[15:0] output Disable time (SRAM/ROM)	$t_{XDODE}$		-6	—	—		
XD[15:0] output hold time (SRAM/ROM) ARM ACCESS	$t_{XDOH}$		$t_{HC} + 3$	—	—		
XD[15:0] output hold time (SRAM/ROM) DMAC ACCESS	$t_{XDOH}$		$t_{HC} + 5.5$	—	—		
XROMCS_N, XRAMCS_N output hold time 1 (SRAM/ROM)	$t_{CSH1}$		$t_{HC} - 1.5$	—	—		
XROMCS_N, XRAMCS_N Output hold time 2 (SRAM/ROM)	$t_{CSH2}$		$t_{HC} + 0.5$	—	—		

■ I/O Control signal timing

( $V_{DD\_CORE} = 2.25$  to  $2.75V$ ,  $V_{DD\_IO} = 3.0$  to  $3.6V$ ,  $T_a = -40$  to  $+85^{\circ}C$ )

IO 0/ IO 1/ IO 2/ IO 3								
Item	Symbol	Condi- tions	Minimum	Typical	Maximum	Unit	Notes	
XIOCS_N[0]/XIOCS_N[1] access time 1 (external I/O 0,1,2, and 3) READ ACCESS	$t_{XIOCSR1}$	—	$n_{IO1} t_{HC} - 5$	—	$n_{IO1} t_{HC} + 1$	ns	The IO01AC,IO23ACX,and IO23ACY registers specify the OE/WE pulse width and read off time for accessing external I/O 0, 1, 2, and 3, respectively. For further details, refer to section 11.2.5 and 11.2.6 of chapter 11.	
XIOCS_N[0]/XIOCS_N[1] access time 2 (external I/O 0,1,2, and 3) READ ACCESS	$t_{XIOCSR2}$		$2 * n_{IO1} t_{HC} - 5$	—	$2 * n_{IO1} t_{HC} + 1$			
XA[23:0] access time (external I/O 0,1,2, and 3)	$t_{XIOACC}$		$n_{IO1} t_{HC} - 5$	—	$n_{IO1} t_{HC} + 0.5$			
XBS_N[1:0] access time (external I/O 0,1,2, and 3)	$t_{XIOBS}$		$n_{IO1} t_{HC} - 3$	—	$n_{IO1} t_{HC} + 1.5$			
XWR delay (external I/O 0,1,2, and 3)	$t_{XIOWRD}$	CL = 30 pF	0	—	3.5			
XWR hold time (external I/O 0,1,2, and 3)	$t_{XIOWRH}$		$t_{HC} - 1$	—	$t_{HC} + 3$			
XWAIT sampling timing delay 1 (external I/O 0,1,2, and 3)	$t_{XIOWAITD1}$	—	$n_{IO2} t_{HC}$	—	$n_{IO2} t_{HC}$			$n_{IO1} = (\text{Address Setup}) + (\text{OE/WE Pulse width})$
XWAIT sampling timing delay 2 (external I/O 0,1,2, and 3)	$t_{XIOWAITD2}$		$n_{IO1} t_{HC}$	—	$n_{IO1} t_{HC}$			$n_{IO2} = (\text{Address Setup}) - 1 + (\text{OE/WE Pulse width})$
XWAIT setup time (external I/O 0,1,2, and 3)	$t_{XIOWAITS}$		20	—	—			$n_{IO3} = (\text{Address Setup})$
XWAIT hold time (external I/O 0,1,2, and 3)	$t_{XIOWAITH}$		0	—	—			$n_{IO4} = (\text{Address Setup}) + 1$
XOE_N delay (external I/O 0,1,2, and 3)	$t_{XIOOED}$	CL = 30 pF	$n_{IO3} t_{HC} - 2$	—	$n_{IO3} t_{HC} + 1$	$n_{IO5} = (\text{OE/WE pulse width})$		
XWE_N delay 1 (external I/O 0,1,2, and 3)	$t_{XIOWED}$		$n_{IO4} t_{HC} - 1.5$	—	$n_{IO4} t_{HC} + 1.5$	$n_{IO6} = (\text{Address Setup}) + 1 + (\text{OE/WE Pulse width})$		
XIOCS_N[0]/XIOCS_N[1] access time 1 (external I/O 0,1,2, and 3) WRITE ACCESS	$t_{XIOCSW1}$		$n_{IO6} t_{HC} - 3$	—	$n_{IO6} t_{HC} + 3$	$n_{IO7} = (\text{Address Setup}) + 2 + (\text{OE/WE Pulse width})$		
XIOCS_N[0]/XIOCS_N[1] access time 2 (external I/O 0,1,2, and 3) WRITE ACCESS	$t_{XIOCSW2}$		$(n_{IO6} + n_{IO7}) t_{HC} - 3$	—	$(n_{IO6} + n_{IO7}) t_{HC} + 3$			

IO 0/ IO 1/ IO 2/ IO 3 (continued)							
Item	Symbol	Condi- tions	Minimum	Typical	Maximum	Unit	Notes
XBWE_N[1:0] delay (external I/O 0,1,2, and 3)	$t_{XIOVELHD}$	CL = 30 pF	$n_{IO4} t_{HC} - 2$	—	$n_{IO4} t_{HC} + 1$	ns	
XOE_N, XWE_N pulse width (external I/O 0,1,2, and 3)	$t_{XIOOE/WEW}$		$n_{IO5} t_{HC} - 1.5$	—	$n_{IO5} t_{HC} + 1.5$		
XBWE_N[1:0] hold time (external I/O 0,1,2, and 3)	$t_{XIOVELHH}$		-1		0.5		
XBS_N[1:0] delay (external I/O 0,1,2, and 3)	$t_{XIOBSD}$		-4	—	2		
XBS_N[1:0] output hold time (external I/O 0,1,2, and 3)	$t_{XIOBSH}$		$t_{HC} - 2$	—	—		
XA[23:0] delay (external I/O 0,1,2, and 3)	$t_{XIOXAD}$	—	-0.5	—	2		
XA[23:0] output hold time 1 (external I/O 0,1,2, and 3)	$t_{XIOXAH1}$		0	—	—		
XA[23:0] output hold time 2 (external I/O 0,1,2, and 3)	$t_{XIOXAH2}$		$t_{HC} - 1$	—	—		
XD[15:0] input setup time (external I/O 0,1,2, and 3)	$t_{XIODIS}$		20	—	—		
XD[15:0] input hold time (external I/O 0,1,2, and 3)	$t_{XIODIH}$		0	—	—		
XD[15:0] output delay (external I/O 0,1,2, and 3)	$t_{XIODOD}$	CL = 30 pF	$n_{IO1} t_{HC} - 5$	—	$n_{IO1} t_{HC} - 2$		
XD[15:0] output Enable time (external I/O 0,1,2, and 3)	$t_{XIODOE}$		$t_{HC} + 2$	—	—		
XD[15:0] output Disable time (external I/O 0,1,2, and 3)	$t_{XIODODE}$		-6	—	—		
XD[15:0] output hold time (external I/O 0,1,2, and 3)	$t_{XIODOH}$		$t_{HC} + 3$	—	—		
XIOCS_N[1:0] output hold time (external I/O 0,1,2, and 3)	$t_{XIOCSH}$		$t_{HC} - 2$	—	—		



■ DRAM Access Timing

( $V_{DD\_CORE} = 2.25$  to  $2.75V$ ,  $V_{DD\_IO} = 3.0$  to  $3.6V$ ,  $T_a = -40$  to  $+85^{\circ}C$ )

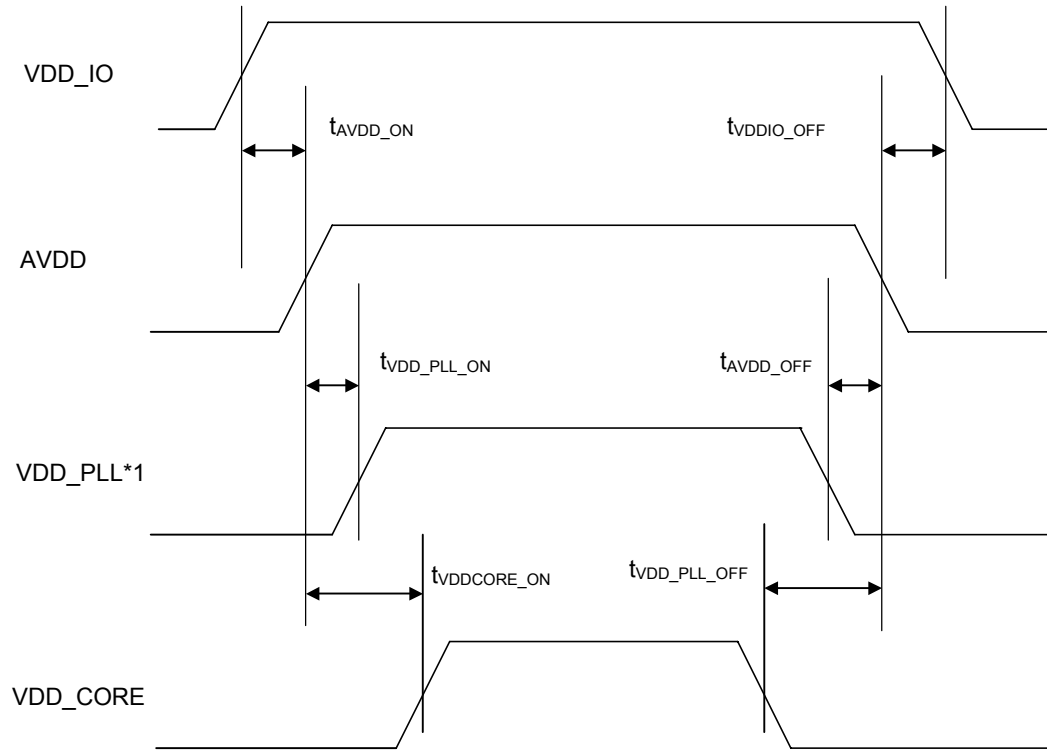
SDRAM								
Item	Symbol	Condi- tions	Minimum	Typical	Maximum	Unit	Notes	
XSDCS_N delay (SDRAM)	$t_{SDCSN}$	CL = 30 pF	$0.5t_{SDC}$	—	$0.5t_{SDC} + 5$	ns	The DRPC register specifies the SDRAM access parameters $t_{RAS}$ , $t_{RCD}$ , $t_{RP}$ , $t_{DPL}$ . For further details, refer to section 11.2.9 of chapter 11.	
XSDCKE delay (SDRAM)	$t_{SDCKED}$		$0.5t_{SDC}$	—	$0.5t_{SDC} + 5$			
XDQM[0]/XDQM[1] delay (SDRAM)	$t_{SDQMD}$		$0.5t_{SDC}$	—	$0.5t_{SDC} + 5.5$			$n_{SD1} = t_{RCD}$
XRAS_N delay (SDRAM)	$t_{SDRASD}$		$0.5t_{SDC}$	—	$0.5t_{SDC} + 5.5$			$n_{SD2} = t_{RAS}$
XCAS_N delay (SDRAM)	$t_{SDCASD}$		$0.5t_{SDC}$	—	$0.5t_{SDC} + 4.5$			$n_{SD3} = t_{RP}$
RASCAS minimum delay (SDRAM)	$t_{SDRCD}$	—	$n_{SD1}t_{SDC}$	—	—			
RAS active time (SDRAM)	$t_{SDRAS}$		$n_{SD2}t_{SDC}$	—	—			
RAS precharge time (SDRAM)	$t_{SDRP}$		$n_{SD3}t_{SDC}$	—	—			
XWE_N delay (SDRAM)	$t_{SDWED}$	CL = 30 pF	$0.5t_{SDC}$	—	$0.5t_{SDC} + 4$			
XD[15:0] input setup time (SDRAM)	$t_{SDXDIS}$	—	14	—	—			
XD[15:0] input hold time (SDRAM)	$t_{SDXDIH}$		0	—	—			
XA[15:0] output delay (SDRAM)	$t_{SDXAD}$	CL = 30 pF	$0.5t_{SDC}$	—	$0.5t_{SDC} + 5$			
XD[15:0] output delay (SDRAM)	$t_{SDXDOD}$		$0.5t_{SDC}$	—	$0.5t_{SDC} + 6$			
XD[15:0] output hold time (SDRAM)	$t_{SDXDOH}$		$0.5t_{SDC}$	—	—			
XD[15:0] output enable time (SDRAM)	$t_{SDXDOE}$		$0.5t_{SDC}$	—	—			
XD[15:0] output disable time (SDRAM)	$t_{SDXDODE}$		—	—	$0.5t_{SDC} + 8$			

( $V_{DD\_CORE} = 2.25$  to  $2.75V$ ,  $V_{DD\_IO} = 3.0$  to  $3.6V$ ,  $T_a = -40$  to  $+85^{\circ}C$ )

EDODRAM								
Item	Symbol	Condi-tions	Minimum	Typical	Maximum	Unit	Notes	
RASCAS delay (EDO DRAM)	$t_{EDRCD}$	—	$n_{ED1} t_{HC} - 2$	—	$n_{ED1} t_{HC} + 2$	ns	The DRPC register specifies the EDO DRAM access parameters $t_{RAH}$ , $t_{CAC}$ , $t_{CAS}$ , $t_{RCD}$ , $t_{RP}$ . For further details, refer to the table on page 19-10.  $n_{ED1} = t_{RCD}$ $n_{ED2} = t_{CAS}$ $n_{ED3} = t_{RP}$ $n_{ED4} = t_{CAC} + 1 - t_{CAS}$ $n_{ED5} = t_{RAH}$ $n_{ED6} = t_{CAC} + 1$ $n_{ED7} = t_{RCD} + N(t_{CAC} + 1)$ N = Data Size / Bus Width	
CAS pulse width (EDO DRAM)	$t_{EDCAS}$	CL = 30 pF	$n_{ED2} t_{HC} - 4$	—	$n_{ED2} t_{HC} + 2$			
RAS pulse width (EDO DRAM)	$t_{EDRAS}$		$n_{ED7} t_{HC} - 4$	—	$n_{ED7} t_{HC} + 1$			
RAS precharge time (EDO DRAM)	$t_{EDRP}$	—	$n_{ED3} t_{HC} - 4$	—	—			
CAS precharge time (EDO DRAM)	$t_{EDCP}$		$n_{ED4} t_{HC} - 4$	—	—			
XRAS_N delay (EDO DRAM)	$t_{EDRASD}$	CL = 30 pF	$t_{HC} - 2$	—	$t_{HC} + 2$			
XOE_N delay 1 (EDO DRAM)	$t_{EDOED1}$		$n_{ED5} t_{HC} - 4$	—	$n_{ED5} t_{HC}$			
XOE_N delay 2 (EDO DRAM)	$t_{EDOED2}$		$n_{ED4} t_{HC} - 2$	—	$n_{ED4} t_{HC}$			
XWE_N delay 1 (EDO DRAM)	$t_{EDWED1}$		$n_{ED5} t_{HC} - 4$	—	$n_{ED5} t_{HC} + 2$			
XWE_N delay 2 (EDO DRAM)	$t_{EDWED2}$		$n_{ED4} t_{HC} - 2$	—	$n_{ED4} t_{HC}$			
Row address hold time (EDO DRAM)	$t_{EDRAH}$		$n_{ED5} t_{HC} - 2$	—	$n_{ED5} t_{HC} + 4$			
Column address delay (EDO DRAM)	$t_{EDCAD}$		$t_{HC} - 2$	—	$t_{HC} + 4$			
Column address hold time (EDO DRAM)	$t_{EDCAH}$		$n_{ED2} t_{HC} - 4$	—	$n_{ED2} t_{HC} + 2$			
XD[15:0] sampling timing delay (EDO DRAM)	$t_{EDXDSMPLD}$		—	$n_{ED6} t_{HC}$	—			$n_{ED6} t_{HC}$
XD[15:0] input setup time (EDO DRAM)	$t_{EDXDIS}$			16	—			—
XD[15:0] input hold time (EDO DRAM)	$t_{EDXDIH}$	0		—	—			
XD[15:0] output delay 1 (EDO DRAM)	$t_{EDXDOD1}$	CL = 30 pF	0	—	6			
XD[15:0] output delay 2 (EDO DRAM)	$t_{EDXDOD2}$		0.5	—	6			
XD[15:0] output hold time (EDO DRAM)	$t_{EDXDOH}$		$n_{ED2} t_{HC} - 2$	—	—			
XD[15:0] output enable time (EDO DRAM)	$t_{EDXDOE}$		$-(n_{ED5} + 1) t_{HC} + 1$	—	—			
XD[15:0] output disable time (EDO DRAM)	$t_{EDXDODE}$		—	—	$(n_{ED2} + n_{ED4}) t_{HC} + 0.5$			

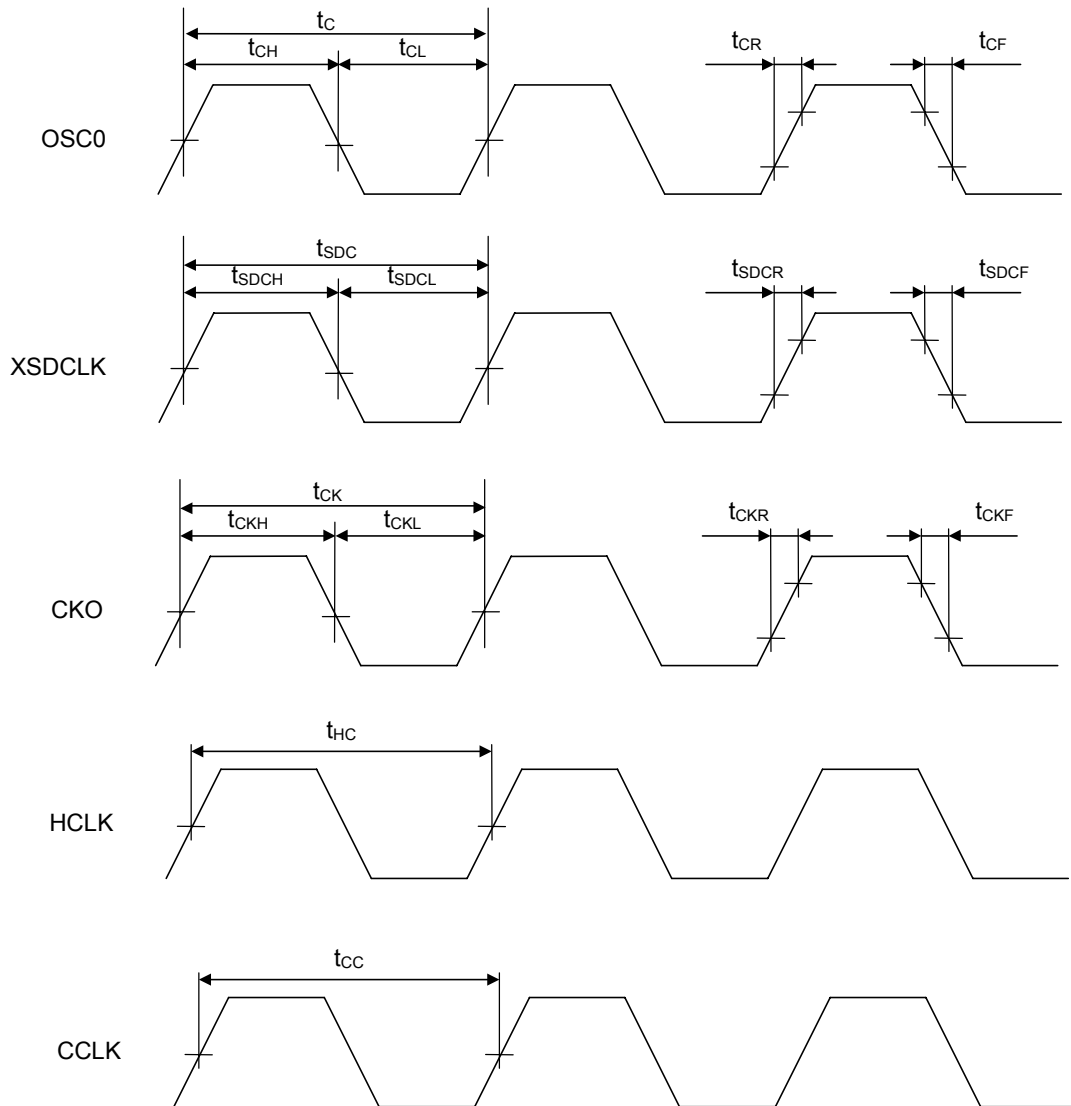
24.4.3 Timing Charts

■ Power Supply On/OFF Timing

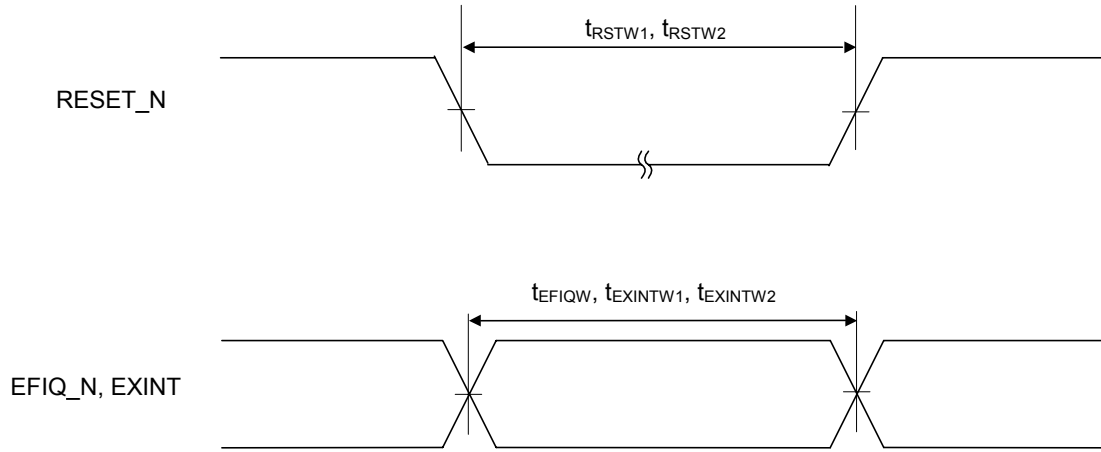


\*1:VDD\_PLL for ML675001 Series only

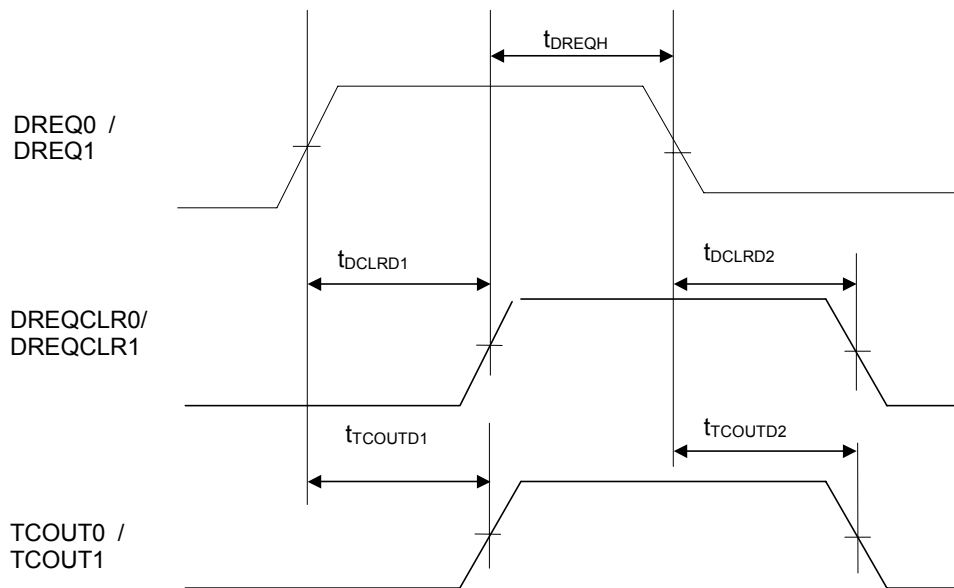
■ Clock Timing



### ■ Control Signal Timing

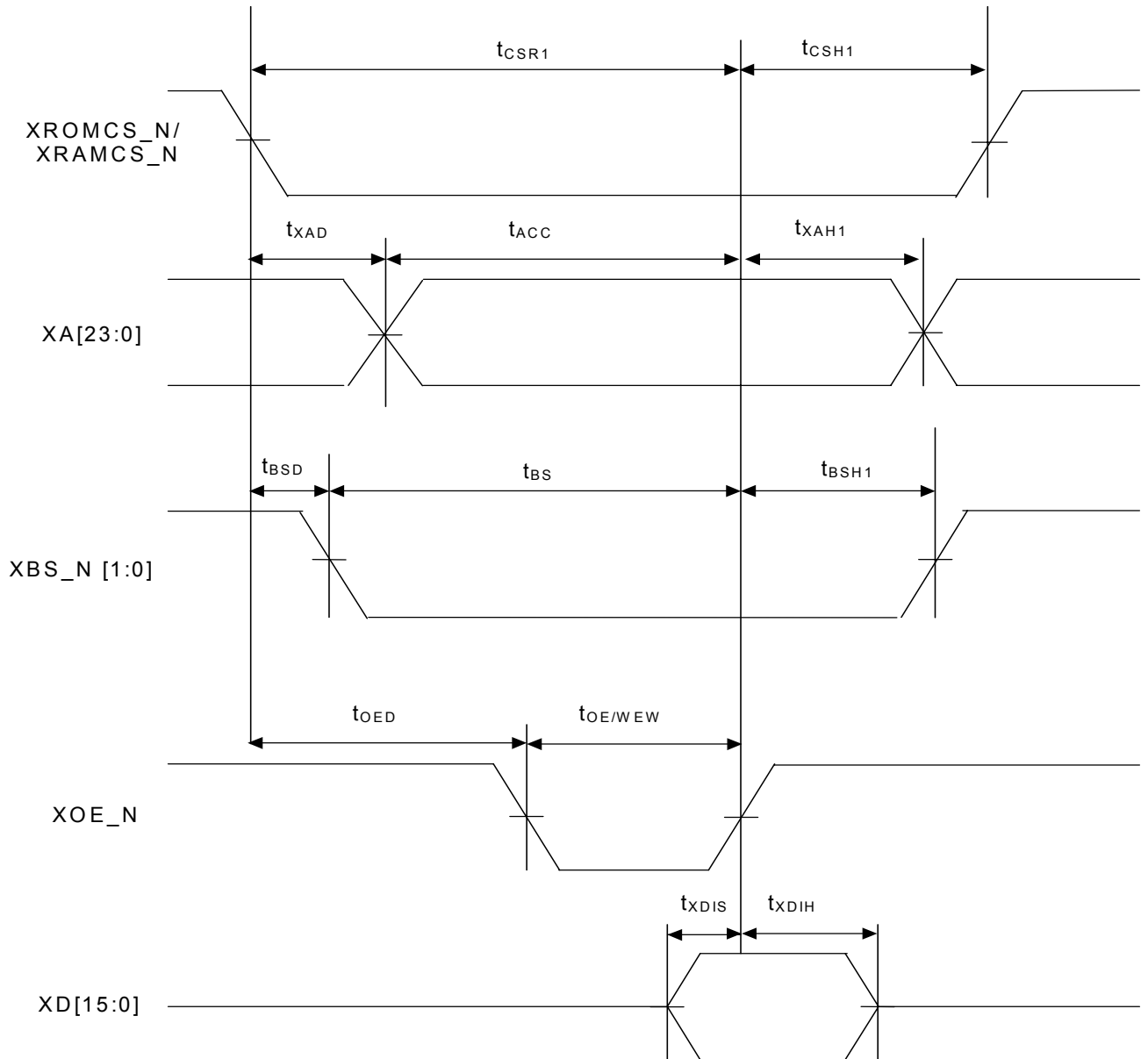


### ■ DMA Timing

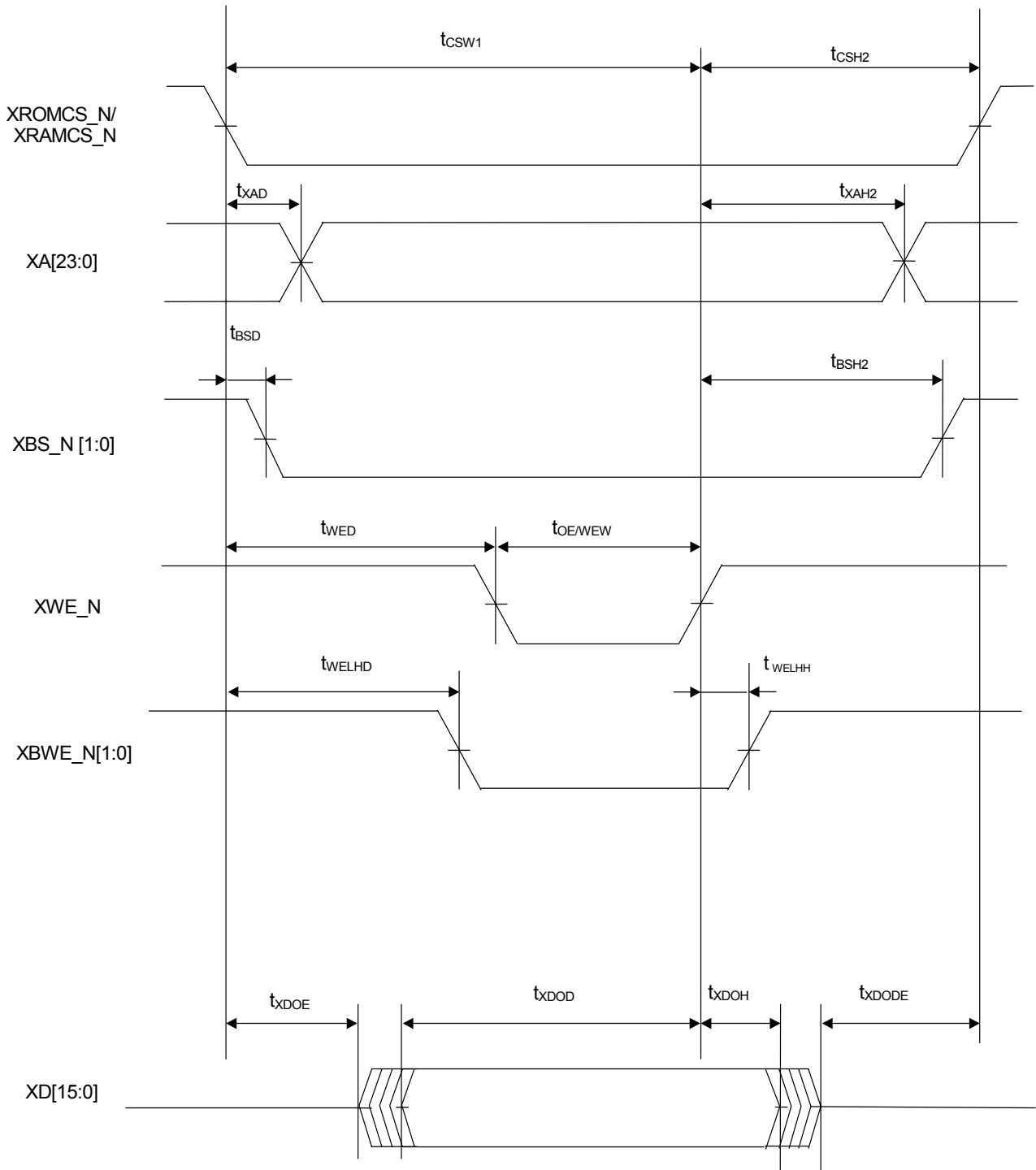


■ External Bus Timing

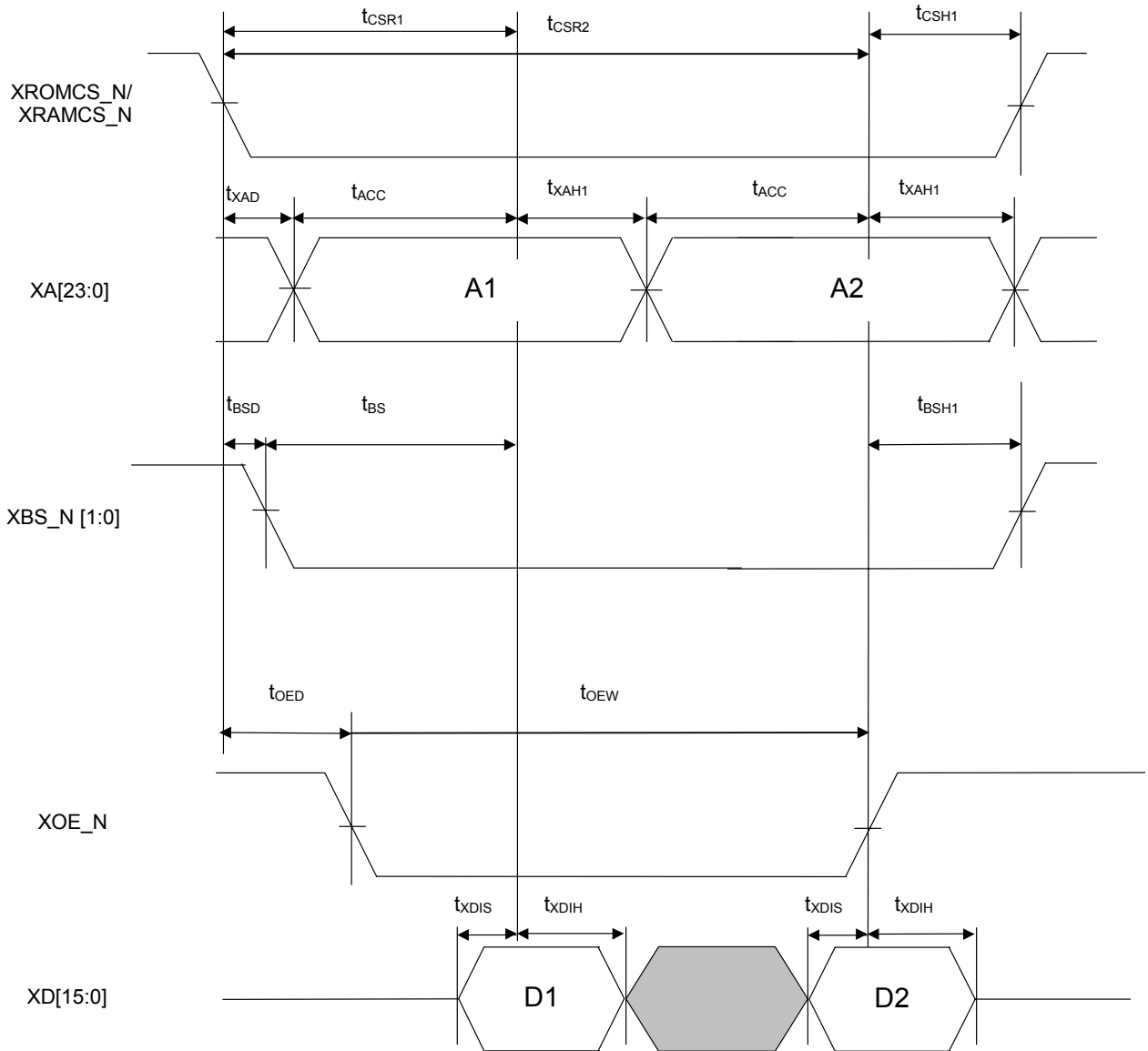
- External ROM/RAM Read Cycle  
 (Bus Width 16 bit External ROM/RAM Byte/Half Word Access)



- External ROM/RAM Write Cycle  
 (Bus Width 16 bit External ROM/RAM Byte/Half Word Access)

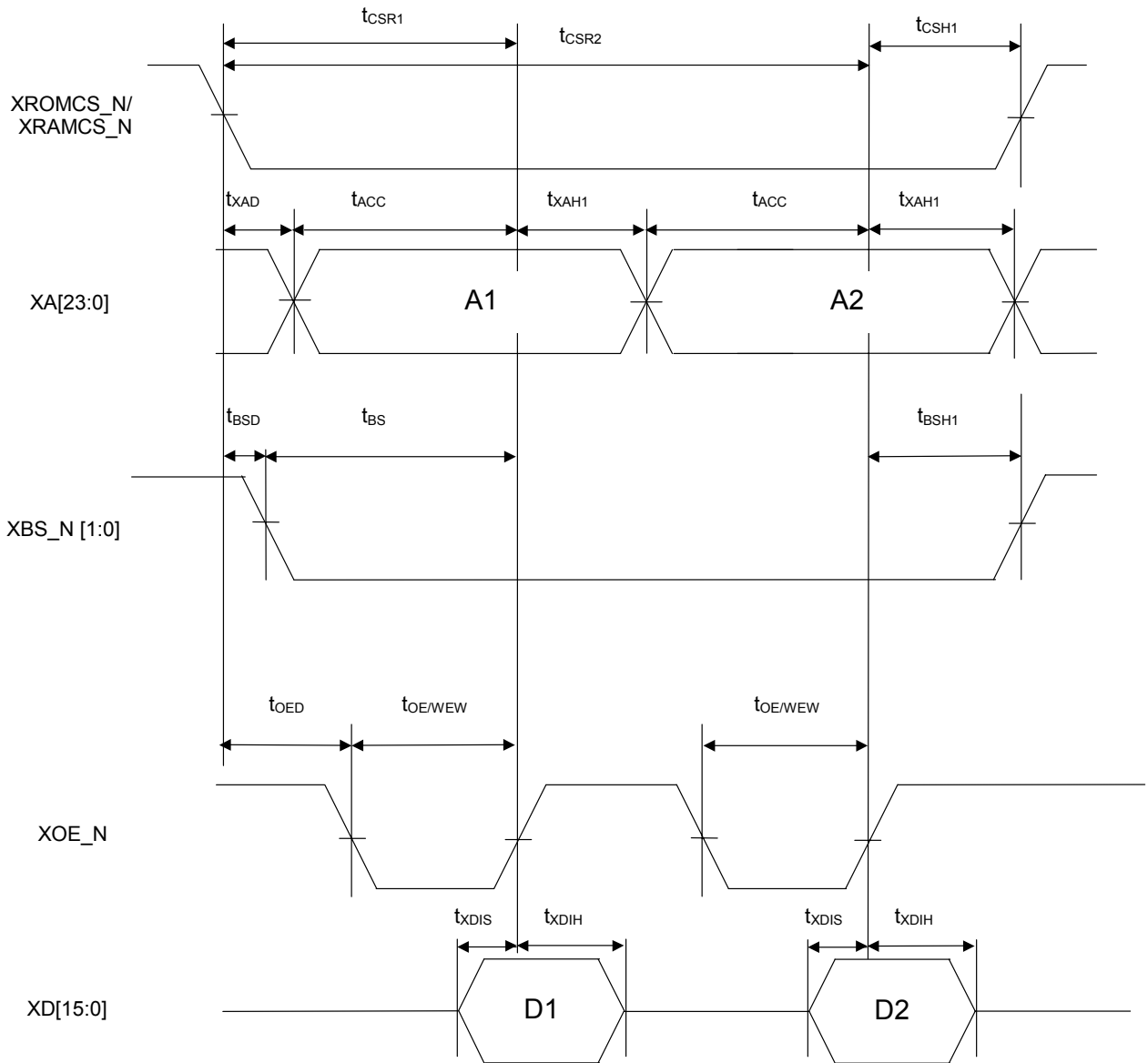


- External ROM/RAM Read Cycle  
 (Bus Width 16 bit External ROM/RAM Word Access from CPU)

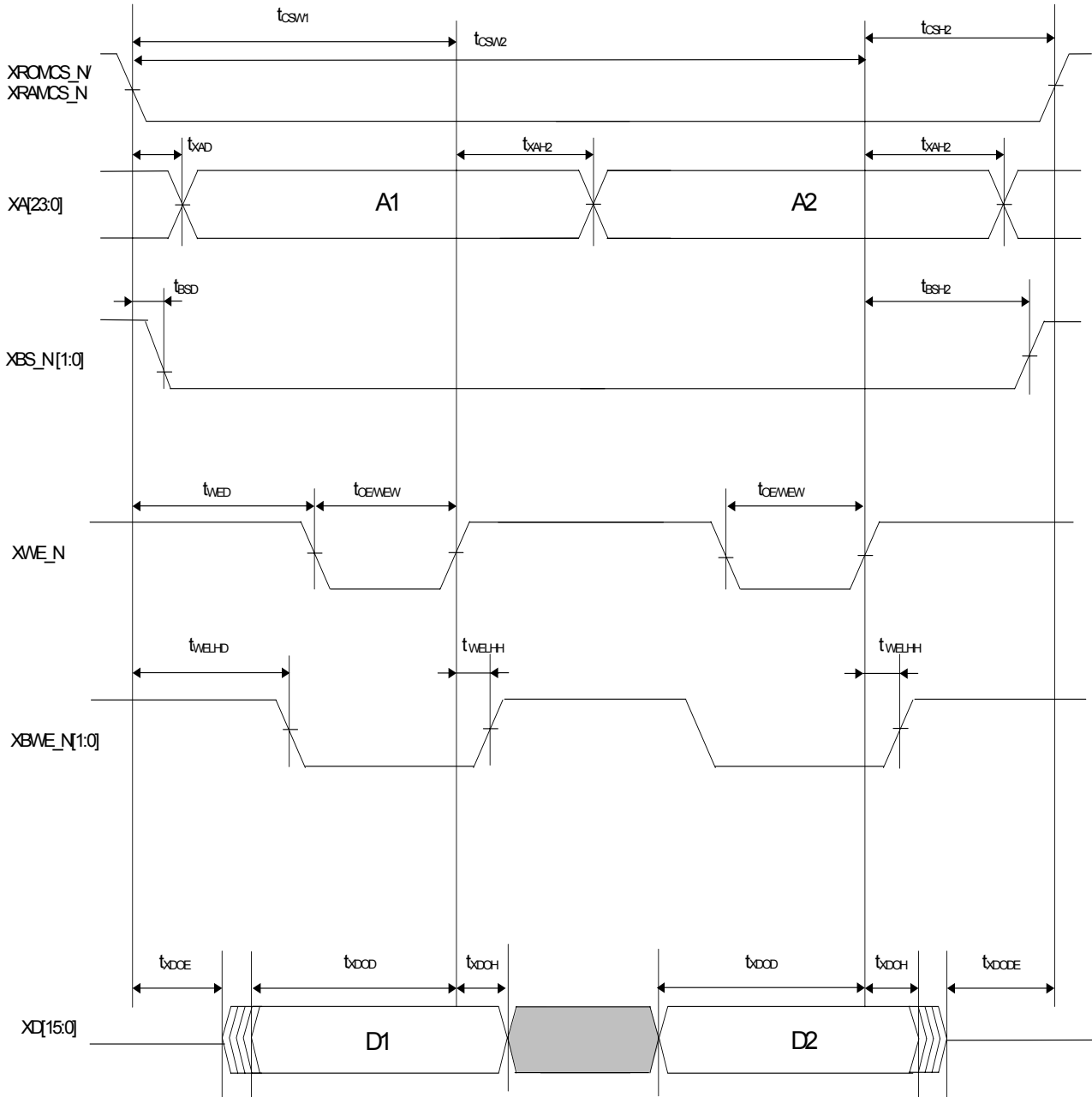




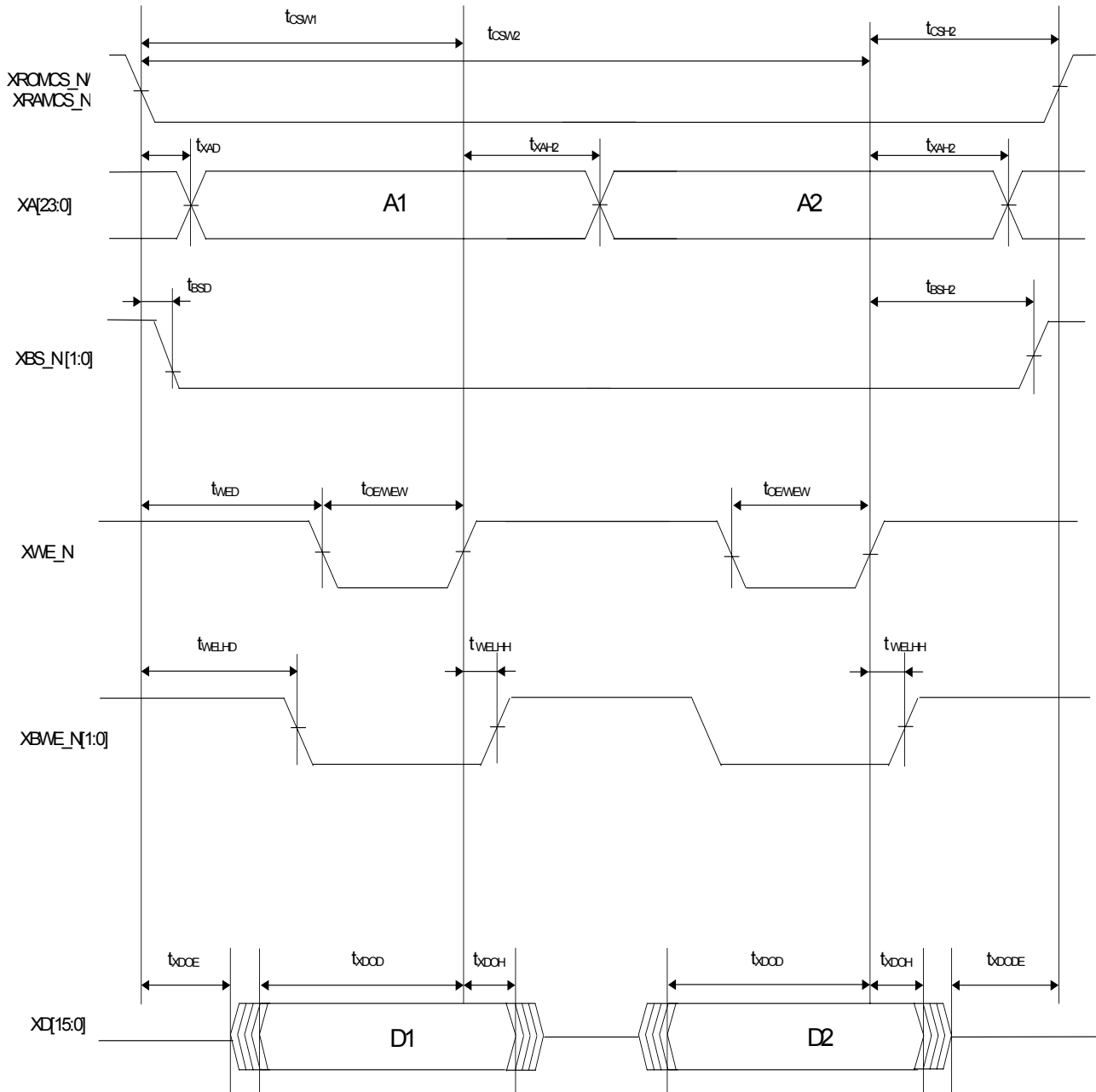
- External ROM/RAM Read Cycle  
 (Bus Width 16 bit External ROM/RAM Word Access from DMAC)



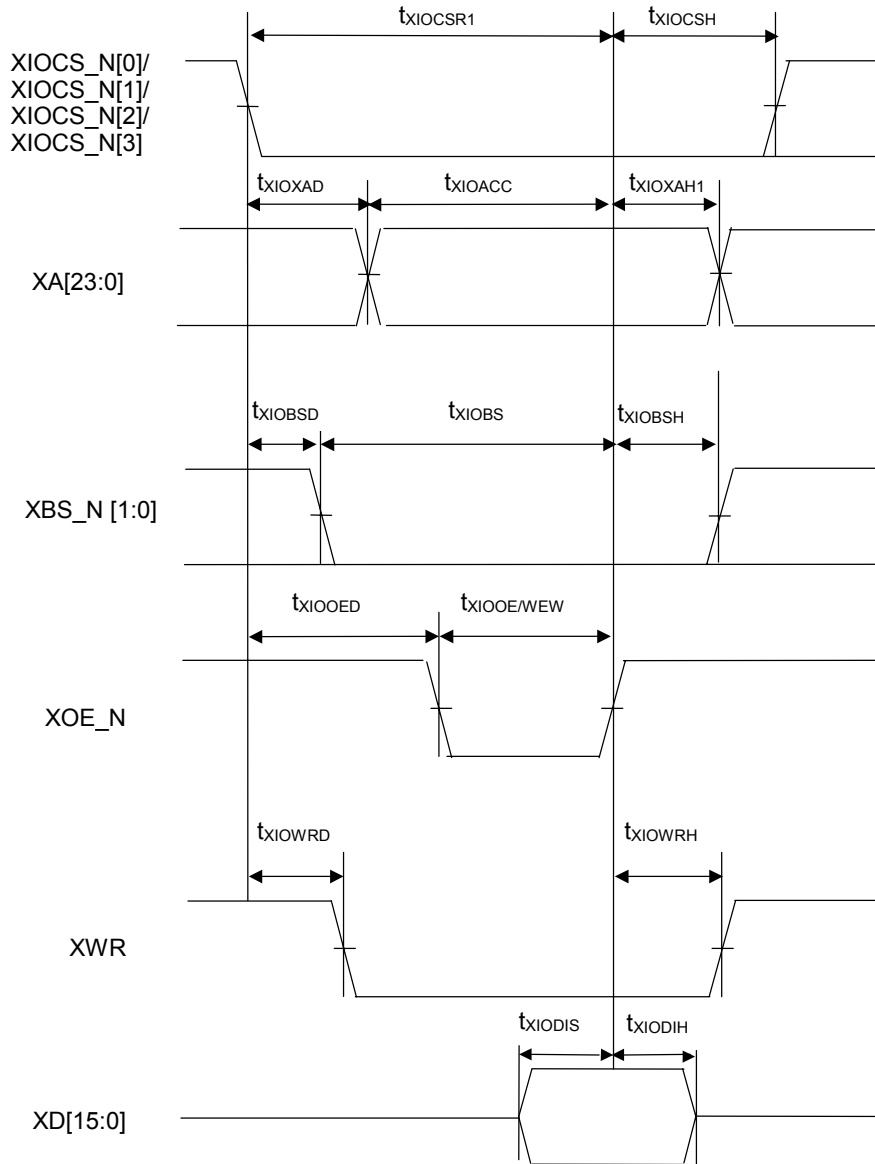
- External ROM/RAM Write Cycle  
 (Bus Width 16 bit External ROM/RAM Word Access from CPU)



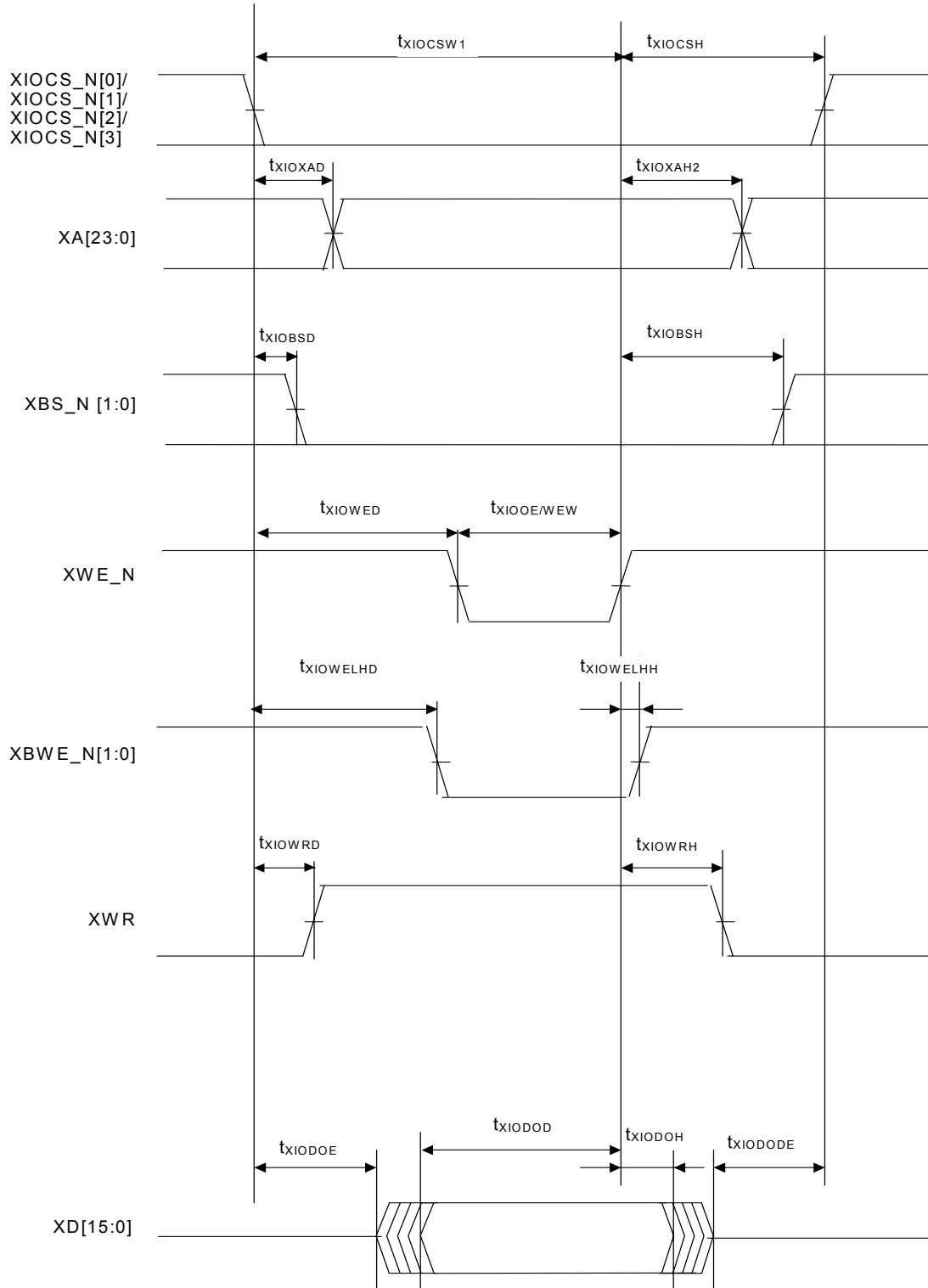
- External ROM/RAM Write Cycle  
 (Bus Width 16 bit External ROM/RAM Word Access from DMAC)



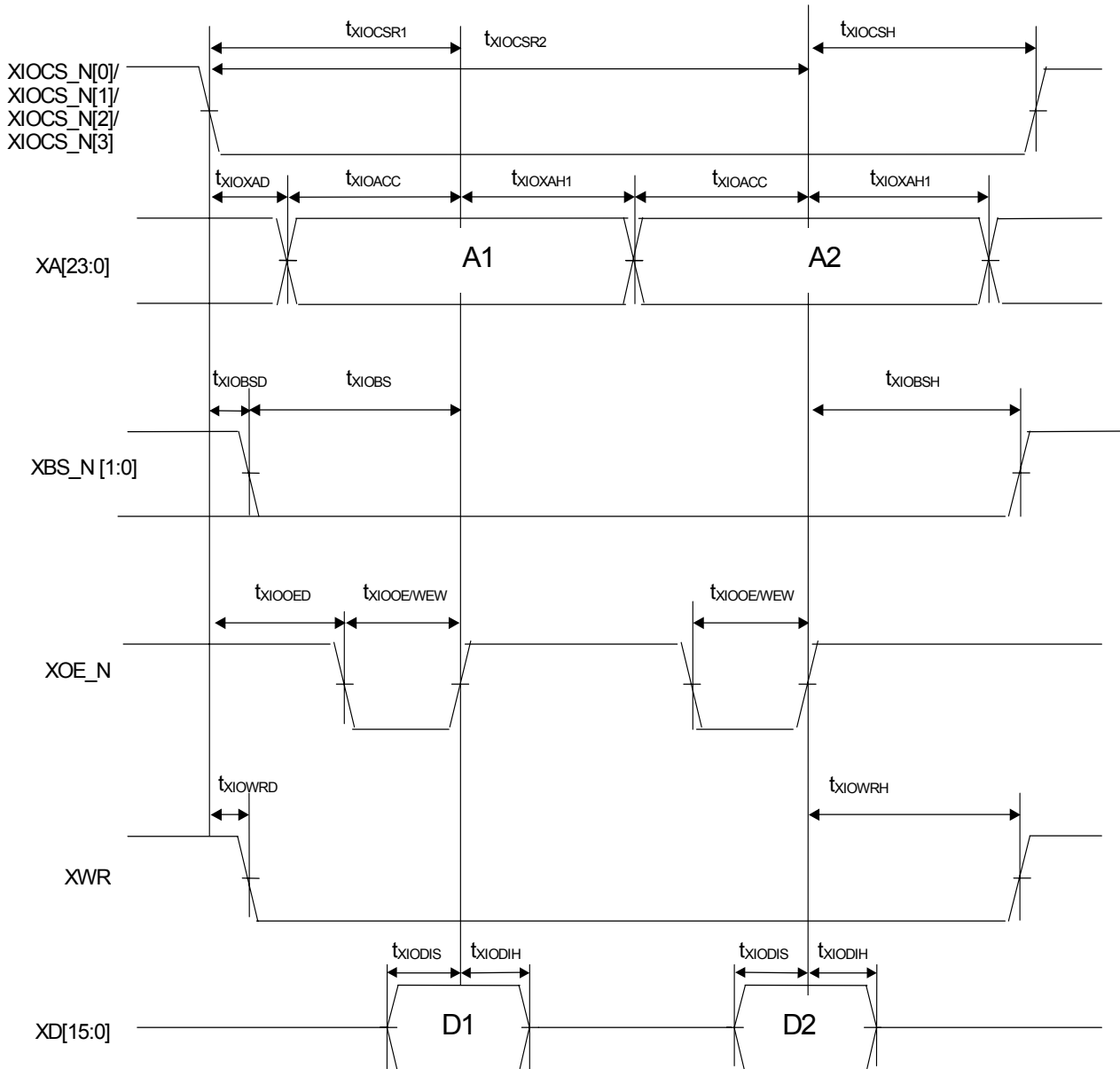
- External I/O 0, 1, 2, 3 Read Cycle  
 (Bus Width 16 bit External I/O 0, 1, 2, 3 Byte/Half Word Access)



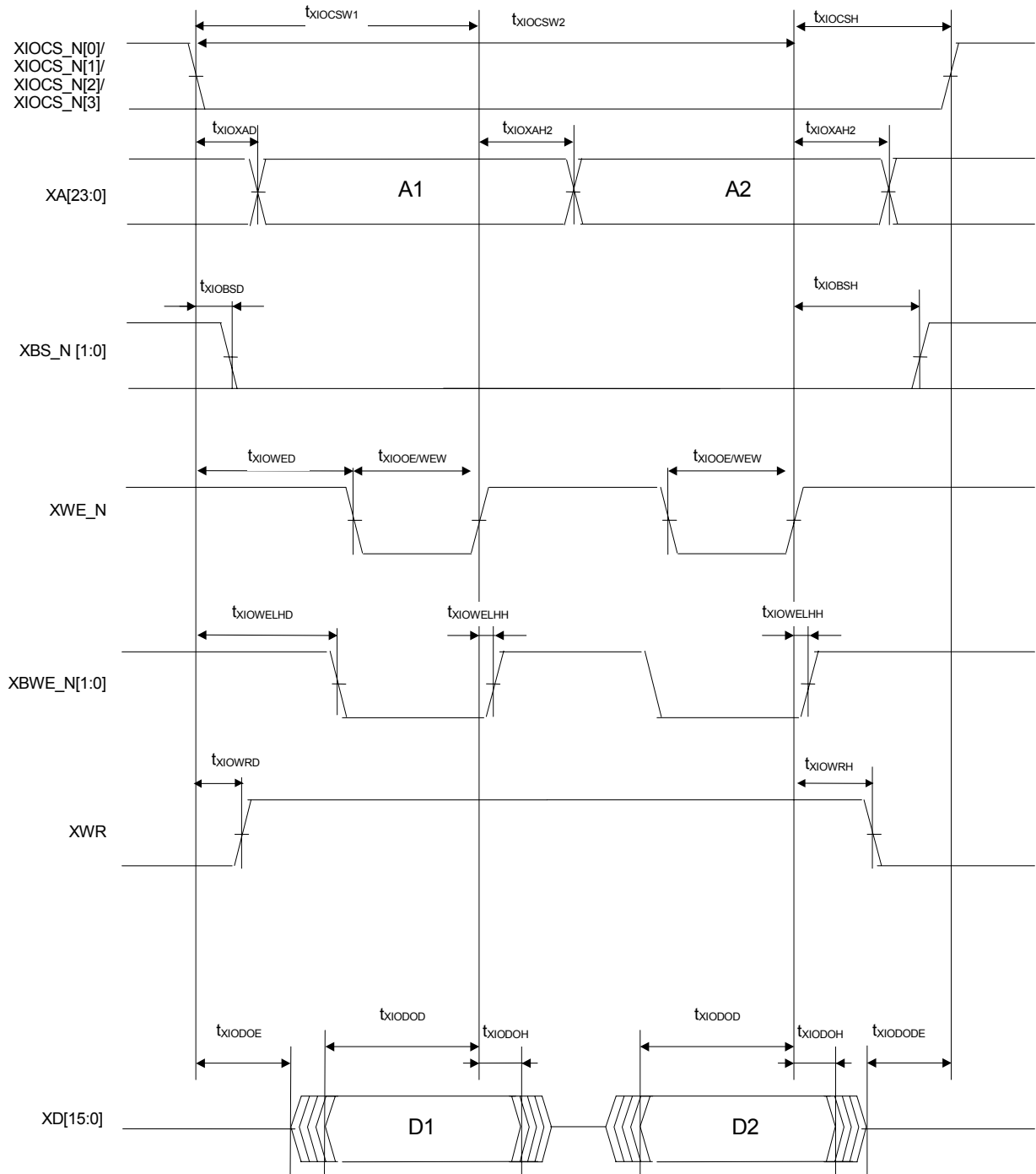
- External I/O 0, 1, 2, 3 Write Cycle  
 (Bus Width 16 bit External I/O 0, 1, 2, 3 Byte/Half Word Access)



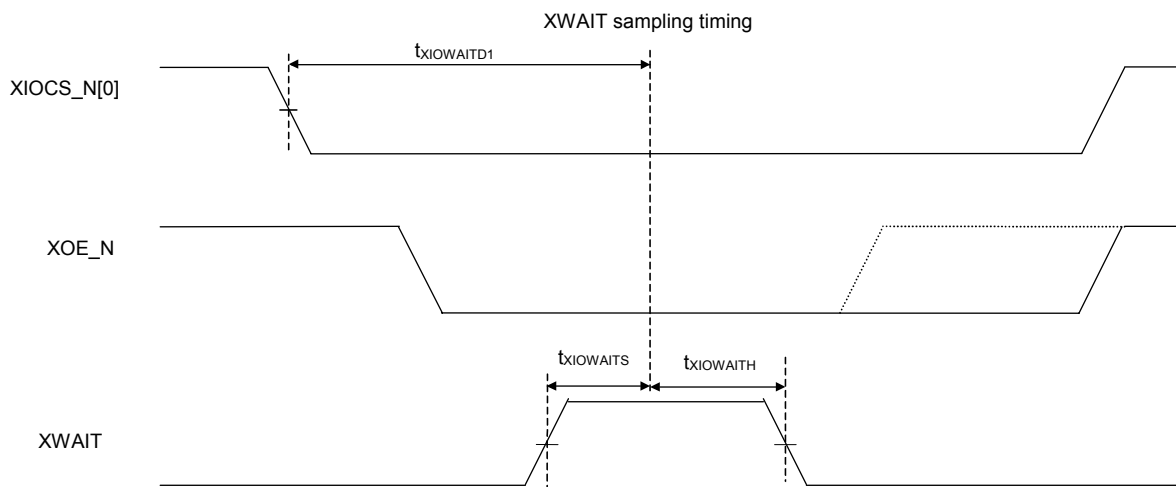
- External I/O 0, 1, 2, 3 Read Cycle  
 (Bus Width 16 bit External I/O 0, 1, 2, 3 Word Access/  
 Bus Width 8 bit External I/O 0, 1, 2, 3 Half Word Access)



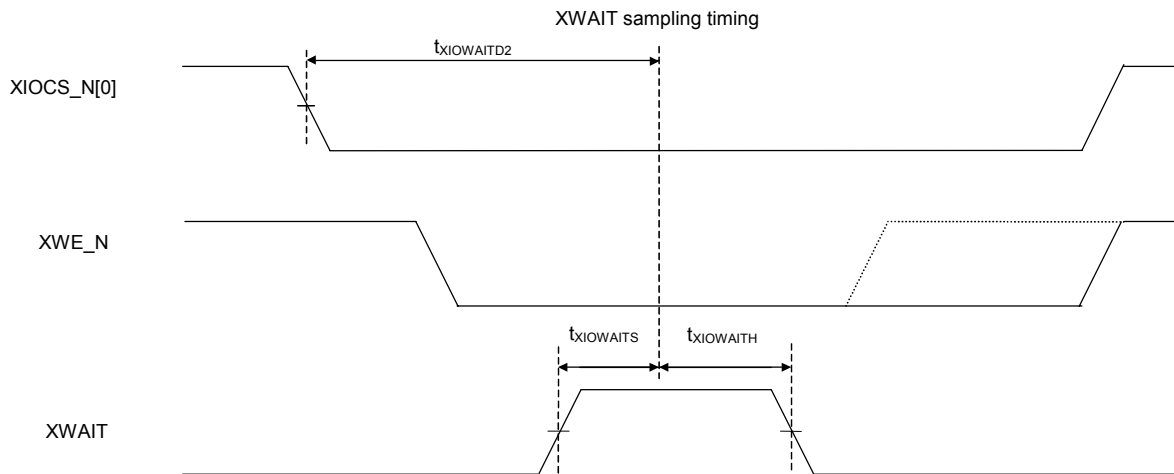
- External I/O 0, 1, 2, 3 Write Cycle  
 (Bus Width 16 bit External I/O 0, 1, 2, 3 Word Access/  
 Bus Width 8 bit External I/O 0, 1, 2, 3 Half Word Access)



- XWAIT Signal Input Timing



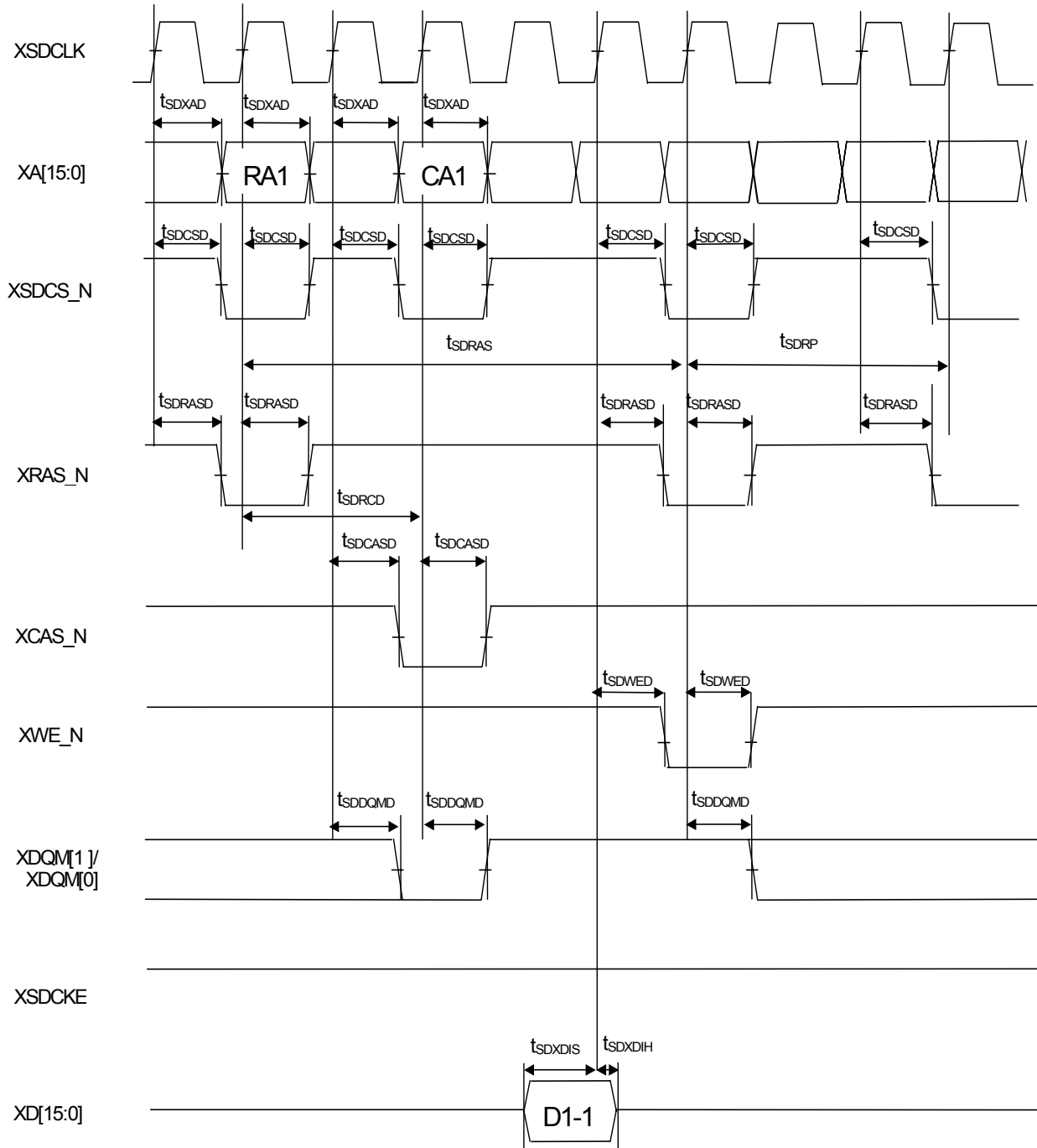
- XWAIT Signal Input Timing



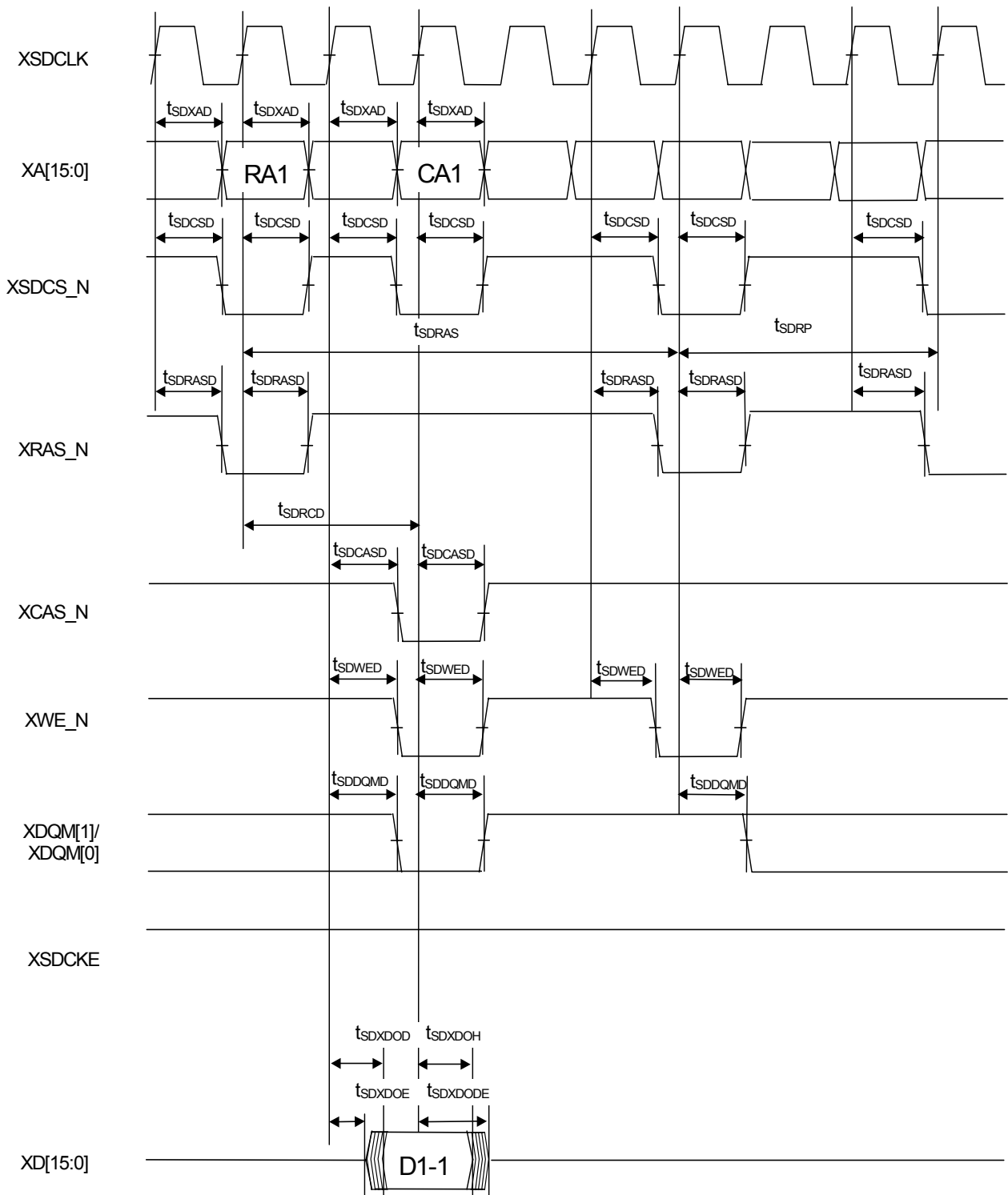


■ SDRAM Control Signal Timing

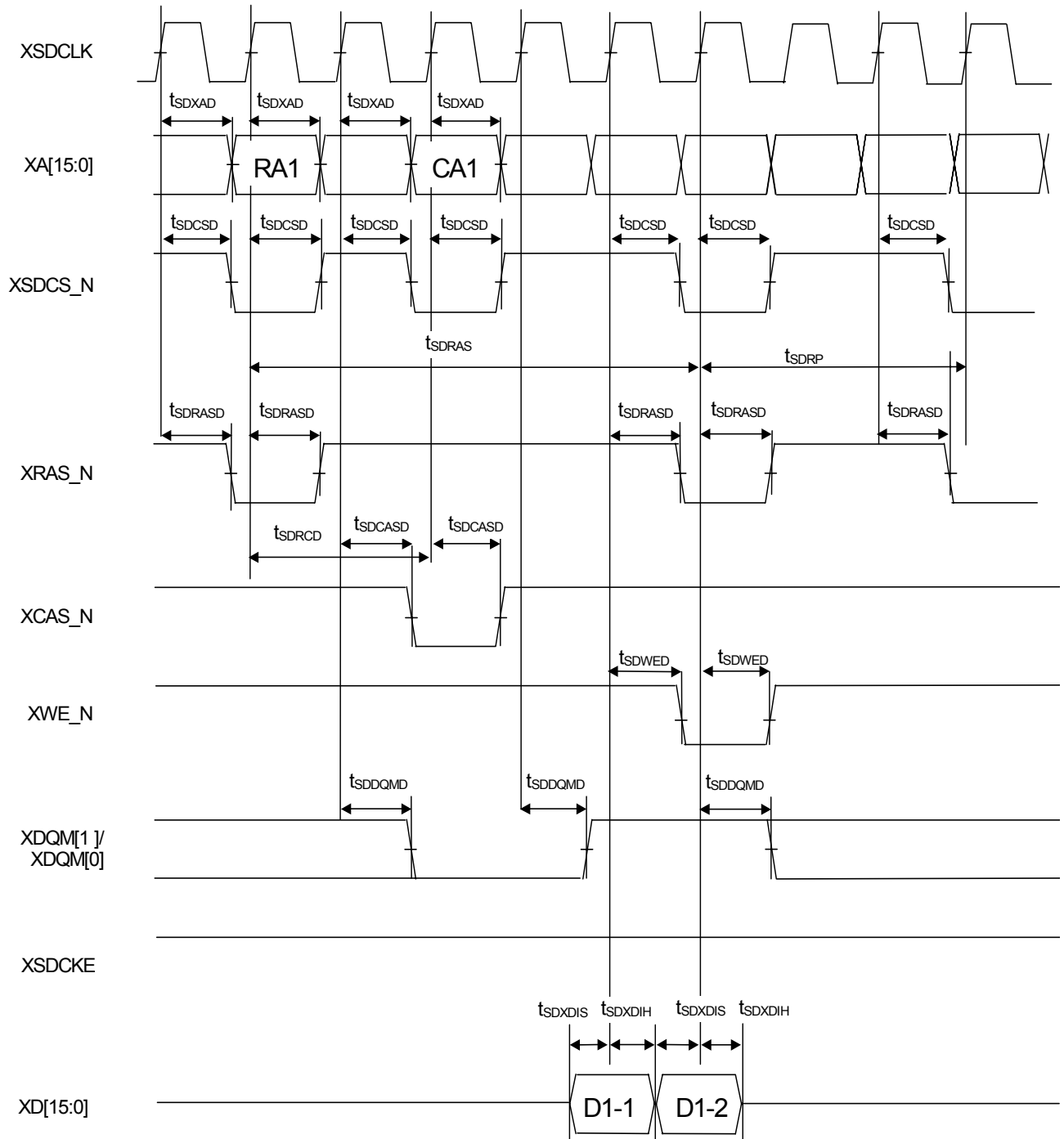
- SDRAM Read Cycle  
 (Bus Width 16 bit SDRAM Byte/Half Word Access)



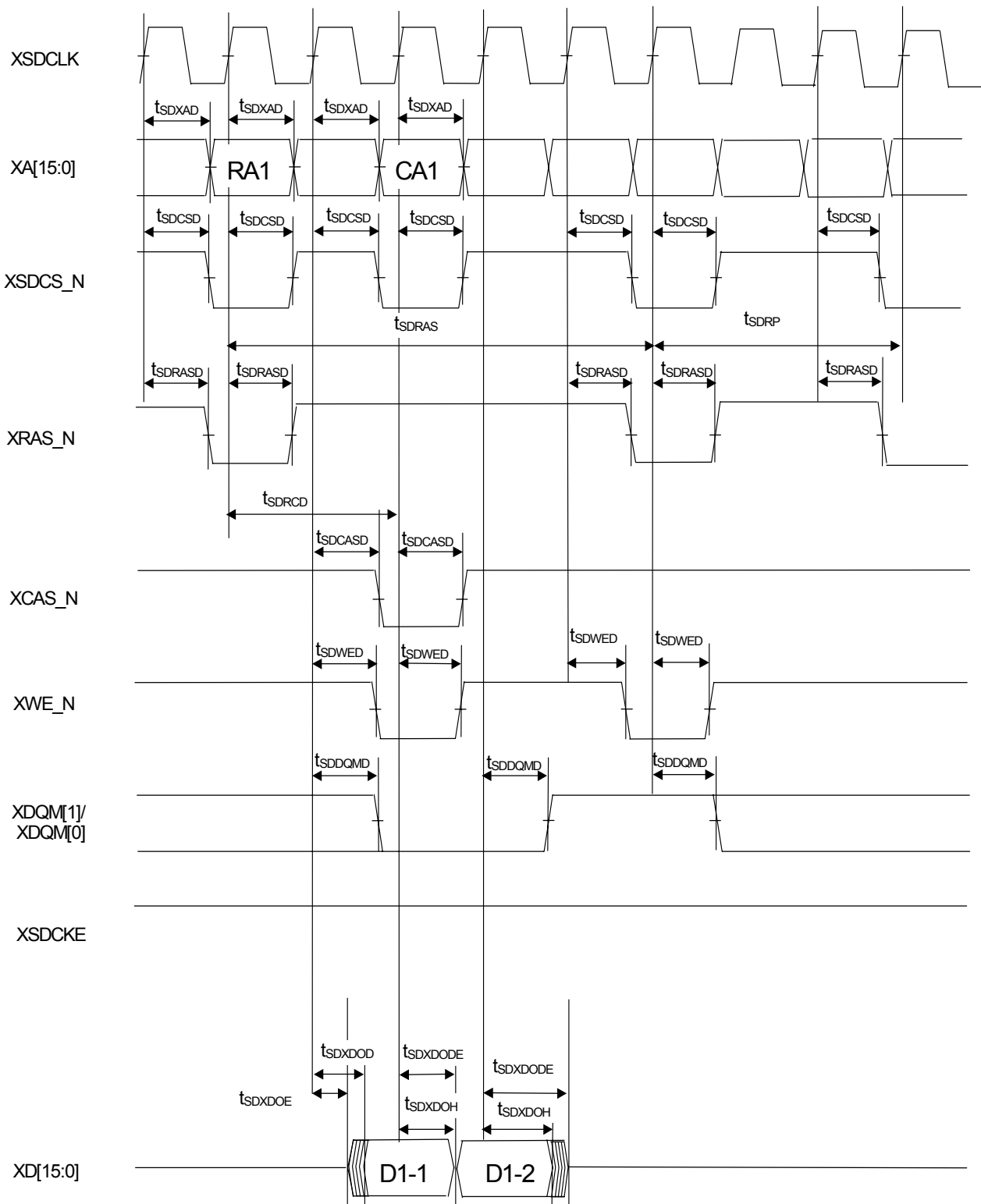
• SDRAM Write Cycle  
 (Bus Width 16 bit SDRAM Byte/Half Word Access)



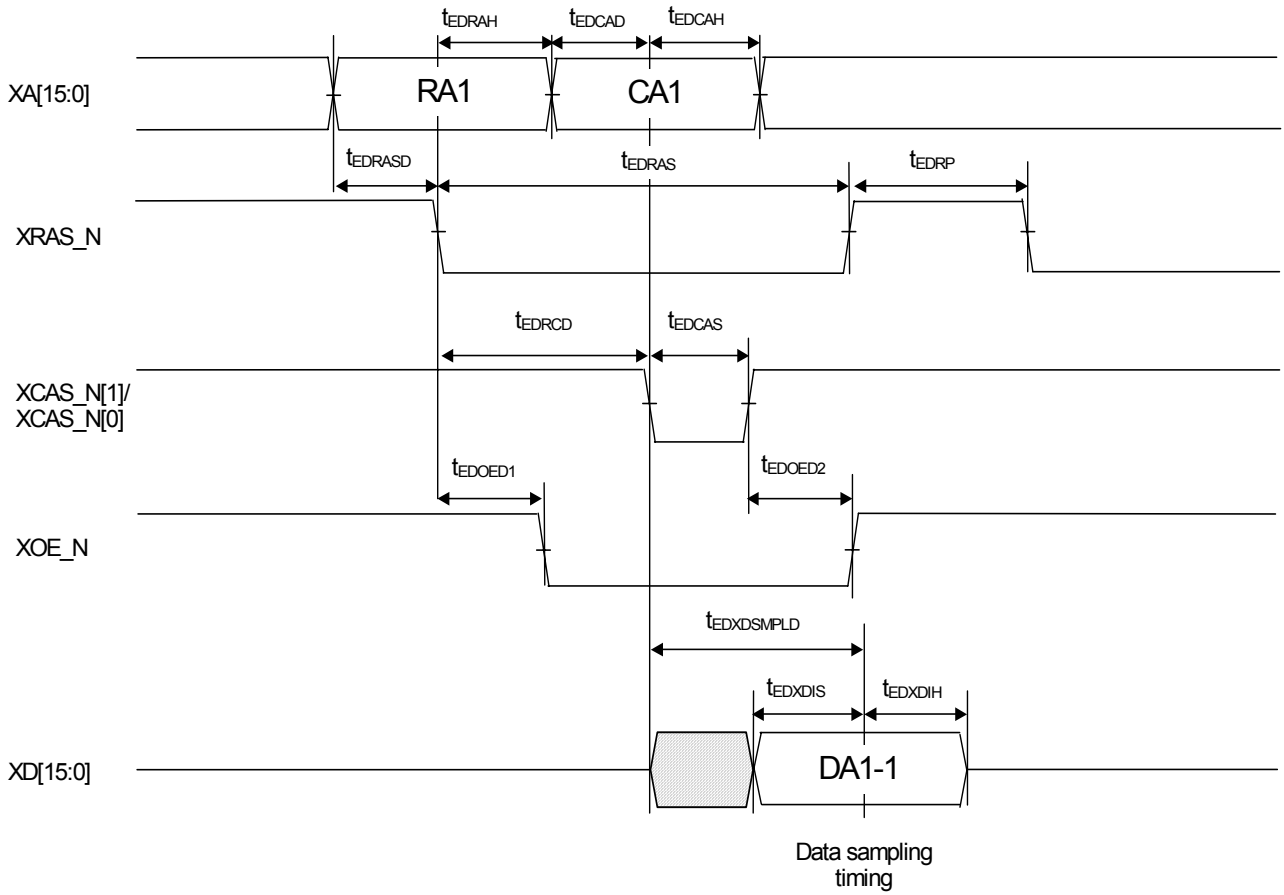
- SDRAM Read Cycle  
 (Bus Width 16 bit SDRAM Word Access)



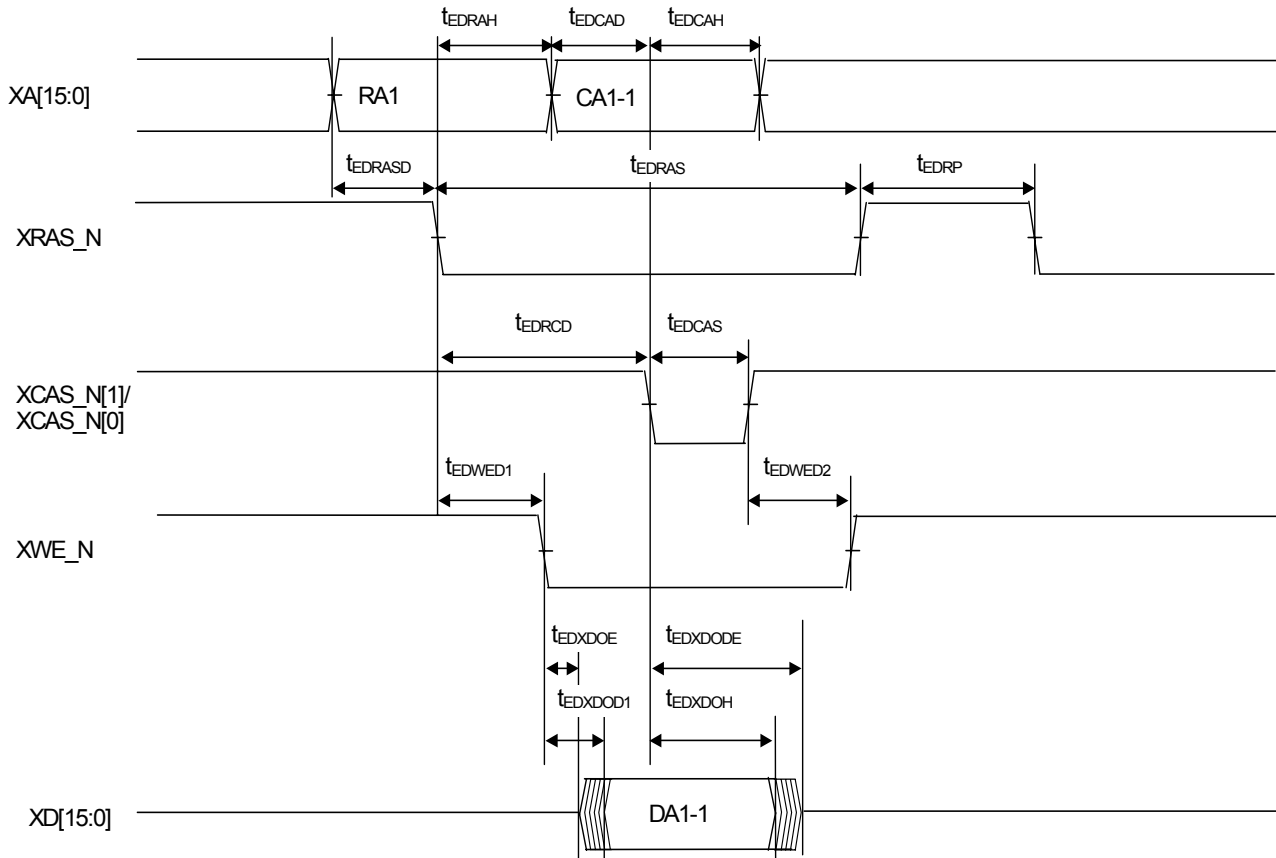
- SDRAM Write Cycle  
 (Bus Width 16 bit SDRAM Word Access)



- EDO DRAM Read Cycle  
 (Bus Width 16 bit EDO DRAM Byte/Half Word Access)

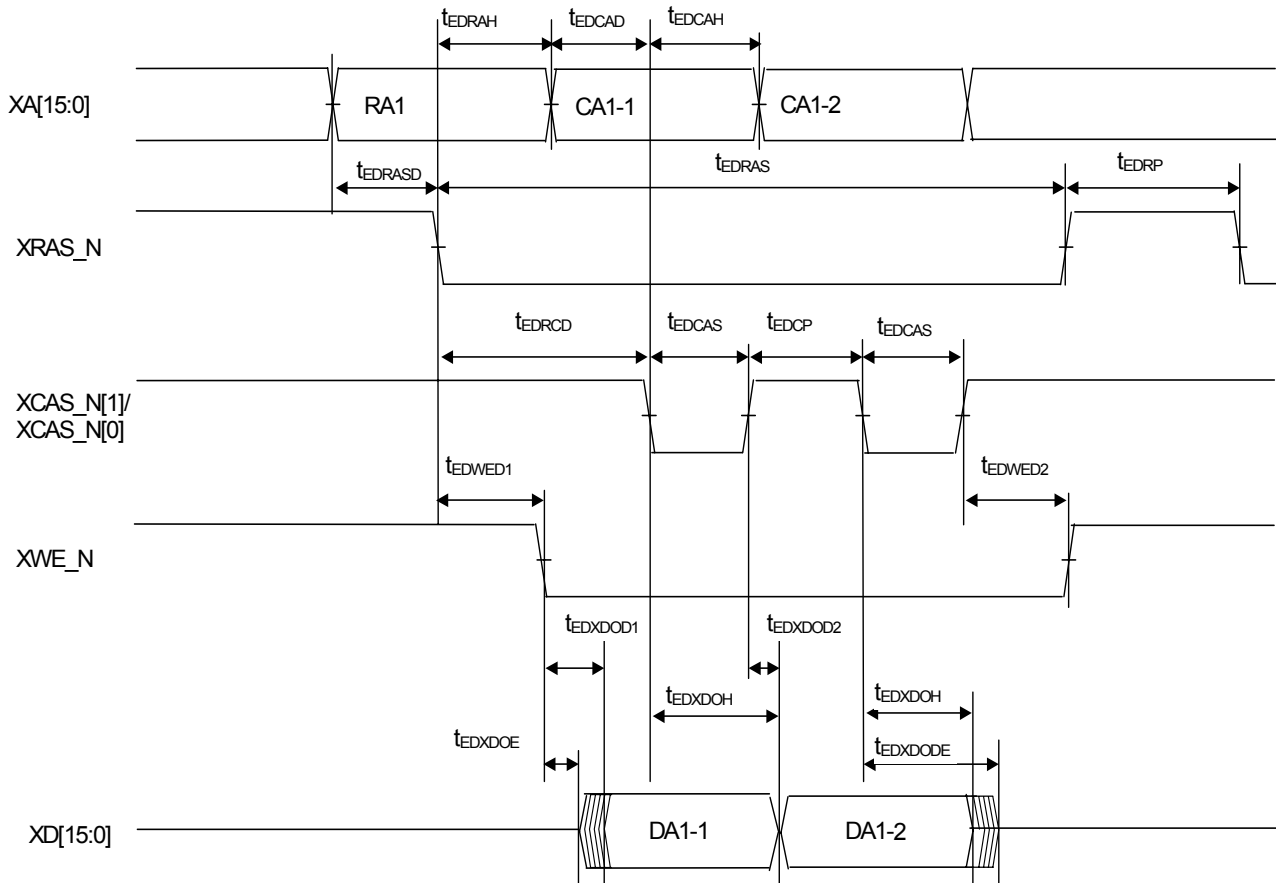


- EDO DRAM Write Cycle  
 (Bus Width 16 bit EDO DRAM Byte/Half Word Access)



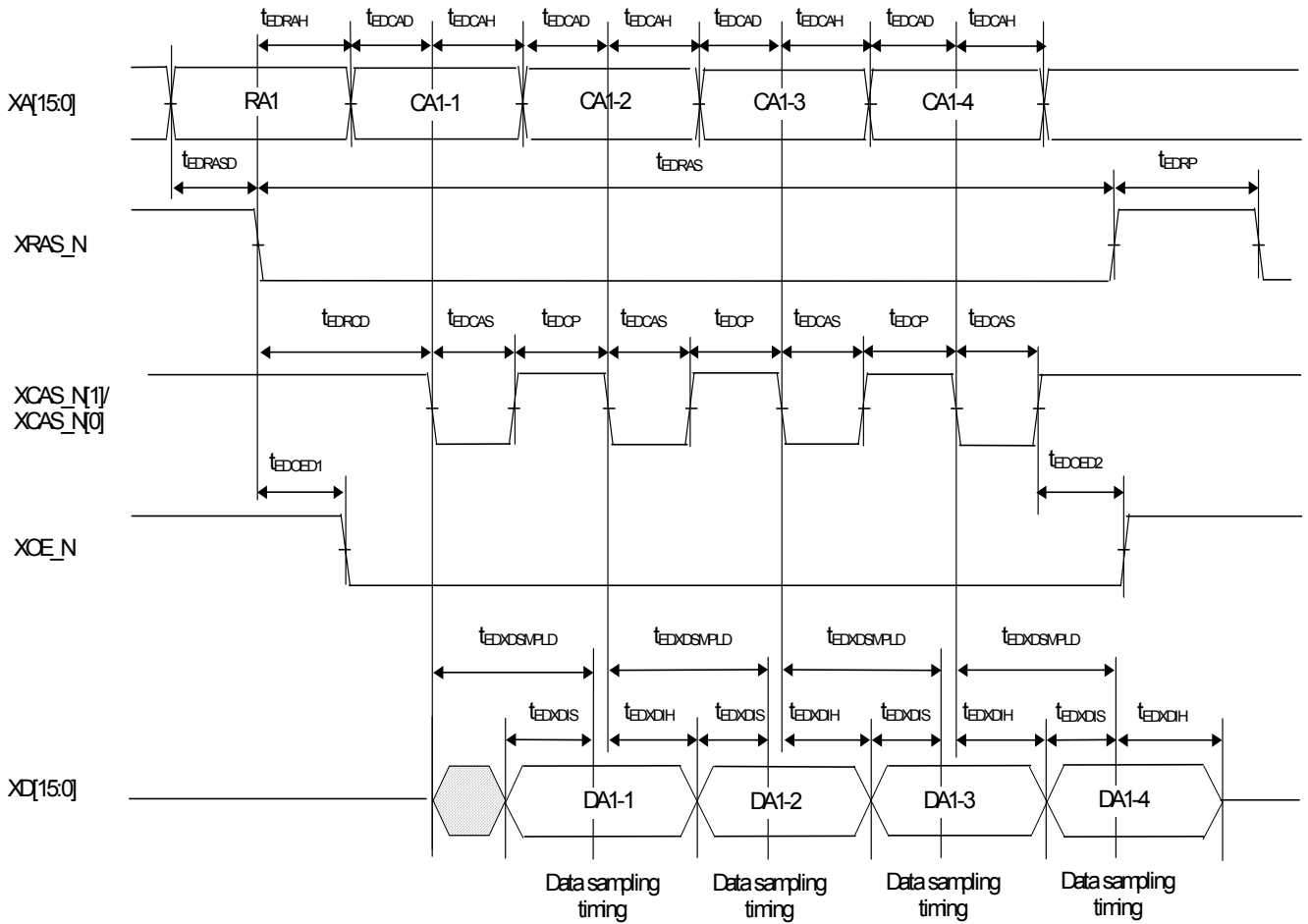


- EDO DRAM Write Cycle  
 (Bus Width 8 bit EDO DRAM Half Word Access/  
 Bus Width 16 bit EDO DRAM Word Access)

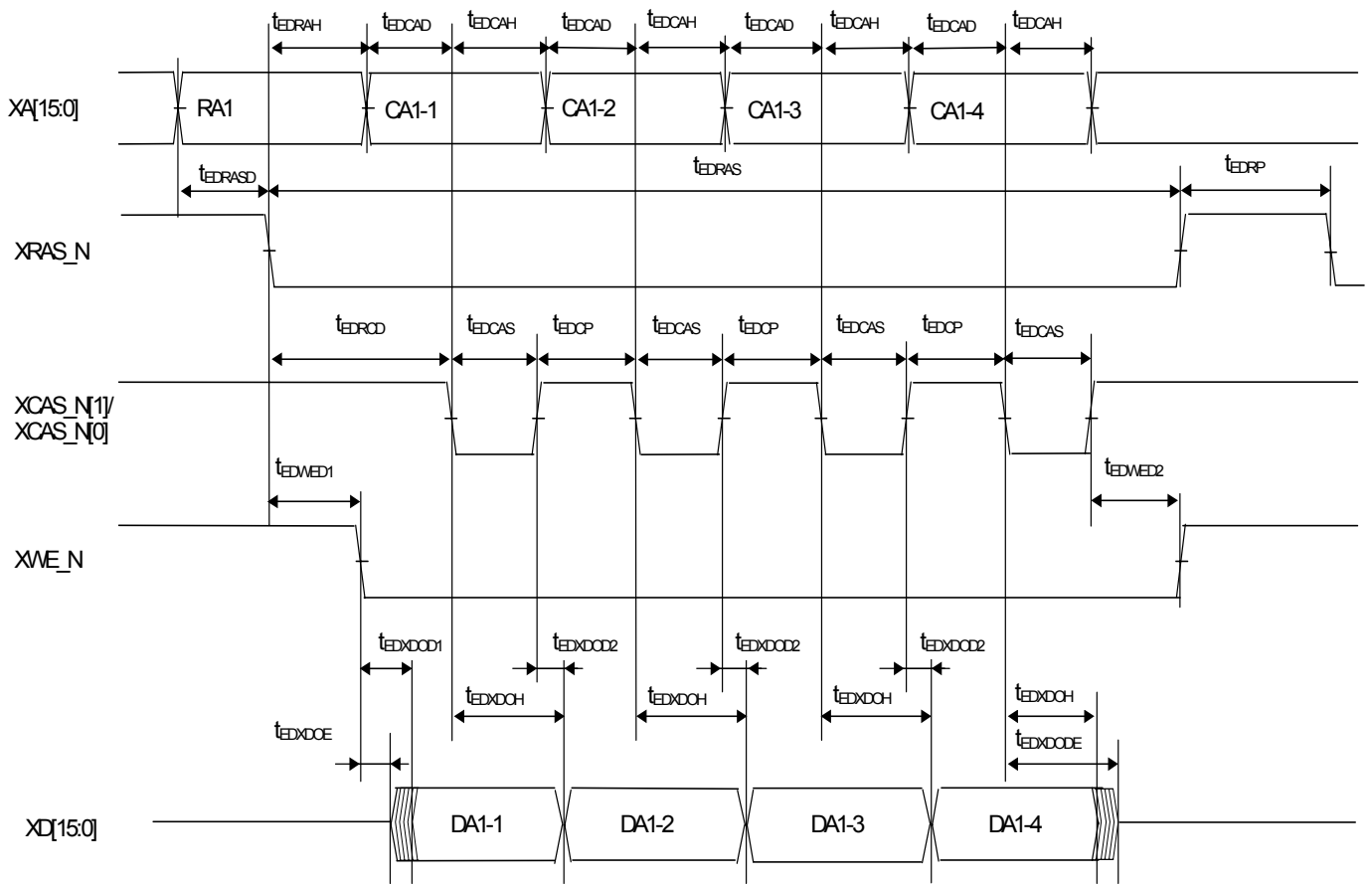




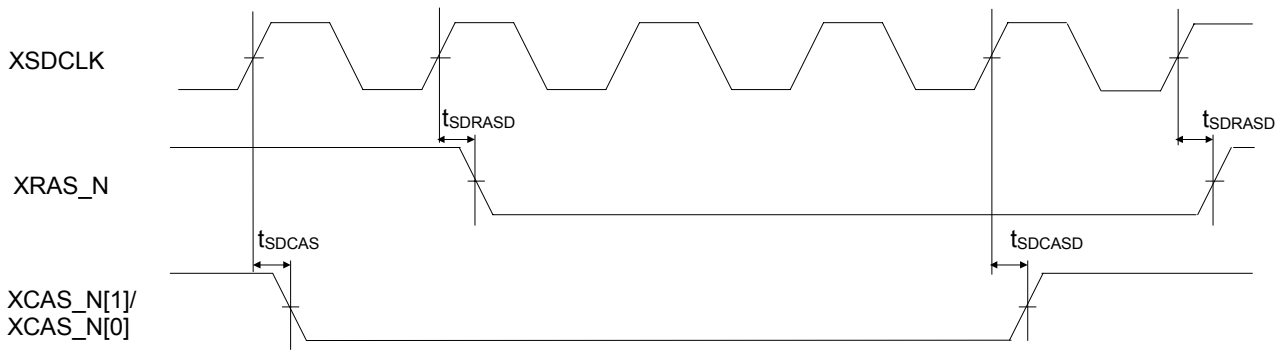
- EDO DRAM Read Cycle  
 (Bus Width 8 bit EDO DRAM Word Access)



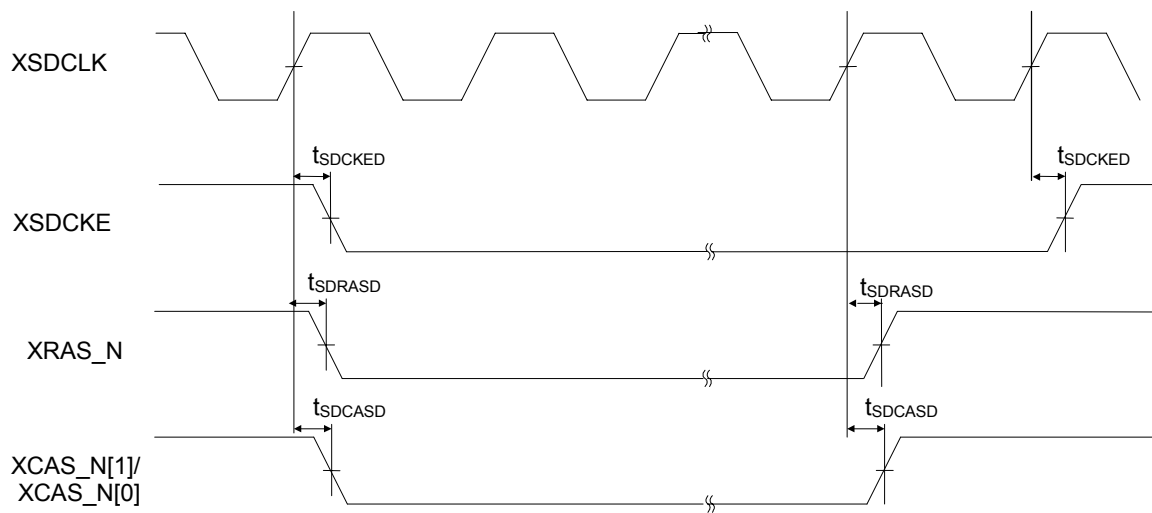
- EDO DRAM Write Cycle  
 (Bus Width 8 bit EDO DRAM Word Access)



- CAS before RAS (CBR) Refresh Operation



- Self Refresh Operation



## 24.5 Analog-to-Digital Converter Characteristics

### ■ Analog-to-Digital Converter Characteristics

( $V_{DD\_CORE} = 2.50V, V_{DD\_IO} = 3.3V, T_a = 25C$ )

Item	Symbol	Conditions	Minimum	Typical	Maximum	Unit
Resolution	n	—	—	—	10	bit
Linearity error	$E_L$	Analog input source impedance $R_i \leq 1k\Omega$	—	$\pm 3$	—	LSB
Differential linearity error	$E_D$		—	$\pm 3$	—	
Zero scale error	$E_{ZS}$		—	$\pm 3$	—	
Full scale error	$E_{FS}$		—	$\pm 3$	—	
Conversion time	$t_{CONV}$	—	5	—	—	$\mu s$
Throughput		—	10	—	200	kHz

Notes:  $V_{DD\_io}$  and  $AV_{DD}$  should be supplied separately

### ■ Definition of Terms

- (1) Resolution: Minimum input analog value recognized. For 10-bit resolution, this is  $(V_{REF} - A_{ground}) \div 1024$ .
- (2) Linearity error: Difference between the theoretical and actual conversion characteristics. (Note that it does not include quantization error.) The theoretical conversion characteristic divides the voltage range between  $V_{REF}$  and  $AGND$  into 1024 equal steps.
- (3) Differential linearity error: Difference between the theoretical and actual input voltage change producing a 1-bit change in the digital output anywhere within the conversion range. This is an indicator of conversion characteristic smoothness. The theoretical value is  $(V_{REF} - A_{ground}) \div 1024$ .
- (4) Zero scale error: Difference between the theoretical and actual conversion characteristics at the point where the digital output switches from “0x000” to “0x001.”
- (5) Full scale error: Difference between the theoretical and actual conversion characteristics at the point where the digital output switches from “0x3FE” to “0x3FF.”

# **Appendixes**

---

## Appendixes

### Appendix A. Register List

Address	Name	Abbreviation	R/W	Size	Initial Value	Ref. Pages
0x78000000	IRQ register	IRQ	R	32	0x00000000	8-7
0x78000004	Software interrupt register	IRQS	R/W	32	0x00000000	8-8
0x78000008	FIQ register	FIQ	R	32	0x00000000	8-9
0x7800000C	FIQRAW register	FIQRAW	R	32	Reflects EFIQ_N interrupt input	8-10
0x78000010	FIQ enable register	FIQEN	R/W	32	0x00000000	8-11
0x78000014	IRQ number register	IRN	R	32	0x00000000	8-12
0x78000018	Current interrupt level register	CIL	R/W	32	0x00000000	8-13
0x78000020	Interrupt level control register 0	ILC0	R/W	32	0x00000000	8-14
0x78000024	Interrupt level control register 1	ILC1	R/W	32	0x00000000	8-16
0x78000028	Current interrupt level clear register	CILCL	W	32	—	8-18
0x7800002C	Current interrupt level encode register	CILE	R	32	0x00000000	8-19
0x78100000	Bus width control register	BWC	R/W	32	0x00000008	11-3
0x78300000*	External I/O bank 2/3 Bus width control register *: ML675001 Series only	IO23BWC	R/W	32	0x00000000	11-5
0x78100004	External ROM access control register	ROMAC	R/W	32	0x00000007	11-6
0x78100008	External RAM access control register	RAMAC	R/W	32	0x00000007	11-8
0x7810000C	External I/O bank 0/1 access control register	IO01AC	R/W	32	0x00000007	11-10
0x78100010*	External I/O bank 2/3 access control register X *: ML674001 Series only	IO23ACX	R/W	32	0x00000007	11-12
0x7830000C*	External I/O bank 2/3 access control register Y *: ML675001 Series only	IO23ACY	R/W	32	0x00000007	11-12
0x78180000	DRAM bus width control register	DBWC	R/W	32	0x00000003	11-14
0x78180004	DRAM control register	DRMC	R/W	32	0x00000000	11-15
0x78180008	DRAM characteristics control register	DRPC	R/W	32	0x00000000	11-17
0x7818000C	SDRAM mode register	SDMD	R/W	32	0x00000001	11-18
0x78180010	DRAM command register	DCMD	R/W	32	0x00000000	11-20
0x78180014	DRAM refresh cycle control register 0	RFSH0	R/W	32	0x00000000	11-21
0x78180018	DRAM power down control register	RDWC	W	32	0x00000003	11-24
0x7818001C	DRAM refresh cycle control register 1	RFSH1	R/W	32	0x00000000	11-22
0x7BE00000	DMA mode register	DMAMOD	R/W	32	0x00000000	12-5
0x7BE00004	DMA status register	DMASTA	R	32	0x00000000	12-6
0x7BE00008	DMA transfer complete status register	DMAMINT	R	32	0x00000000	12-7
0x7BE00100	DMA channel mask register	DMACMSK0	R/W	32	0x00000001	12-9
0x7BE00104	DMA transfer mode register	DMACTMOD0	R/W	32	0x00000040	12-10
0x7BE00108	DMA transfer source address register	DMACSDAD0	R/W	32	0x00000000	12-12
0x7BE0010C	DMA transfer destination address register	DMACDAD0	R/W	32	0x00000000	12-13
0x7BE00110	DMA transfer count register	DMACSIZE0	R/W	32	0x00000000	12-14

Address	Name	Abbreviation	R/W	Size	Initial Value	Ref. Pages
0x7BE00114	DMA transfer complete status clear register	DMACCINT0	W	32	—	12-15
0x7BE00200	DMA channel mask register	DMACMSK1	R/W	32	0x00000001	12-9
0x7BE00204	DMA transfer mode register	DMACTMOD1	R/W	32	0x00000040	12-10
0x7BE00208	DMA transfer source address register	DMACSAD1	R/W	32	0x00000000	12-12
0x7BE0020C	DMA transfer destination address register	DMACDAD1	R/W	32	0x00000000	12-13
0x7BE00210	DMA transfer count register	DMACSIZ1	R/W	32	0x00000000	12-14
0x7BE00214	DMA transfer complete status clear register	DMACCINT1	W	32	—	12-15
0x7BF00004	IRQ clear register	IRCL	W	32	—	8-20
0x7BF00010	IRQA register	IRQA	R/W	32	0x00000000	8-21
0x7BF00014	IRQ detection mode setting register	IDM	R/W	32	0x00000000	8-23
0x7BF00018	Interrupt level control register	ILC	R/W	32	0x00000000	8-24
0xB6001000	Analog-to-digital converter control register 0	ADCON0	R/W	16	0x0000	21-3
0xB6001004	Analog-to-digital converter control register 1	ADCON1	R/W	16	0x0000	21-5
0xB6001008	Analog-to-digital converter control register 2	ADCON2	R/W	16	0x0003	21-6
0xB600100C	Analog-to-digital converter interrupt control register	ADINT	R/W	16	0x0000	21-7
0xB6001010	Analog-to-digital converter forced interrupt register	ADFINT	R/W	16	0x0000	21-9
0xB6001014	Analog-to-digital converter result register 0	ADR0	R/W	16	0x0000	21-10
0xB6001018	Analog-to-digital converter result register 1	ADR1	R/W	16	0x0000	21-10
0xB600101C	Analog-to-digital converter result register 2	ADR2	R/W	16	0x0000	21-10
0xB6001020	Analog-to-digital converter result register 3	ADR3	R/W	16	0x0000	21-10
0xB7000000	Port function select register	GPCTL	R/W	16	0x0000	13-10
0xB7000004	Block clock control register	BCKCTL	R/W	16	0x0000	7-4
0xB700000C	ROM select register	ROMSEL	R/W	16	0x0000	3-5
0xB7800000	I2C bus control register	I2CCON	R/W	8	0x00	20-3
0xB7800004	I2C bus slave address mode register	I2CSAD	R/W	8	0x00	20-4
0xB7800008	I2C bus transfer speed register	I2CCLR	R/W	8	0x00	20-5
0xB780000C	I2C bus status register	I2CSR	R	8	0x00	20-6
0xB7800010	I2C bus interrupt request register	I2CIR	R/W	8	0x00	20-7
0xB7800014	I2C bus interrupt mask register	I2CIMR	R/W	8	0x01	20-7
0xB7800018	I2C bus transmit/receive data register	I2CDR	R/W	8	0x00	20-8
0xB780001C	I2C bus transfer speed counter	I2CBC	R/W	8	0x00	20-8
0xB7A01000	Port A output register	GPPOA	R/W	16	Indeterminate	13-5
0xB7A01004	Port A input register	GPPIA	R	16	Reflects pin states	13-5

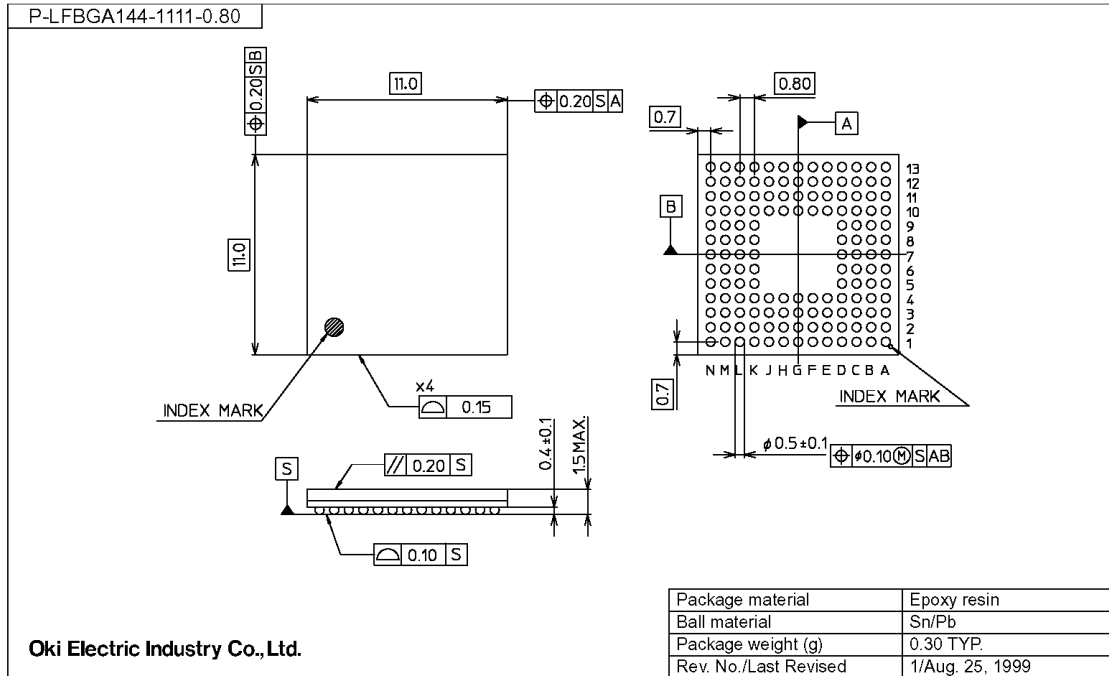
Address	Name	Abbreviation	R/W	Size	Initial Value	Ref. Pages
0xB7A01008	Port A mode register	GPPMA	R/W	16	0x0000	13-6
0xB7A0100C	Port A interrupt enable register	GPIEA	R/W	16	0x0000	13-7
0xB7A01010	Port A interrupt polarity register	GPIPA	R/W	16	0x0000	13-8
0xB7A01014	Port A interrupt status register	GPISA	R/W	16	0x0000	13-9
0xB7A01020	Port B output register	GPPOB	R/W	16	Indeterminate	13-5
0xB7A01024	Port B input register	GPPIB	R	16	Reflects pin states	13-5
0xB7A01028	Port B mode register	GPPMB	R/W	16	0x0000	13-6
0xB7A0102C	Port B interrupt enable register	GPIEB	R/W	16	0x0000	13-7
0xB7A01030	Port B interrupt polarity register	GPIPB	R/W	16	0x0000	13-9
0xB7A01034	Port B interrupt status register	GPISB	R/W	16	0x0000	13-10
0xB7A01040	Port C output register	GPPOC	R/W	16	Indeterminate	13-5
0xB7A01044	Port C input register	GPPIB	R	16	Reflects pin states	13-6
0xB7A01048	Port C mode register	GPPMC	R/W	16	0x0000	13-7
0xB7A0104C	Port C interrupt enable register	GPIEC	R/W	16	0x0000	13-8
0xB7A01050	Port C interrupt polarity register	GPIPC	R/W	16	0x0000	13-9
0xB7A01054	Port C interrupt status register	GPISC	R/W	16	0x0000	13-10
0xB7A01060	Port D output register	GPPOD	R/W	16	Indeterminate	13-5
0xB7A01064	Port D input register	GPPID	R	16	Reflects pin states	13-6
0xB7A01068	Port D mode register	GPPMD	R/W	16	0x0000	13-7
0xB7A0106C	Port D interrupt enable register	GPIED	R/W	16	0x0000	13-8
0xB7A01070	Port D interrupt polarity register	GPIPD	R/W	16	0x0000	13-9
0xB7A01074	Port D interrupt status register	GPISD	R/W	16	0x0000	13-10
0xB7A01080	Port E output register	GPPOE	R/W	16	Indeterminate	13-5
0xB7A01084	Port E input register	GPPIE	R	16	Reflects pin states	13-6
0xB7A01088	Port E mode register	GPPME	R/W	16	0x0000	13-7
0xB7A0108C	Port E interrupt enable register	GPIEE	R/W	16	0x0000	13-8
0xB7A01090	Port E interrupt polarity register	GPIPE	R/W	16	0x0000	13-9
0xB7A01094	Port E interrupt status register	GPISE	R/W	16	0x0000	13-10
0xB7B00000	Divisor Latch (LSB)	UARTDLL	R/W	8	Indeterminate	18-21
0xB7B00000	Receiver Buffer Register	UARTRBR	R	8	Indeterminate	18-4
0xB7B00000	Transmitter Holding Register	UARTTHR	W	8	Indeterminate	18-5
0xB7B00004	Divisor Latch (MSB)	UARTDLM	R/W	8	Indeterminate	18-22
0xB7B00004	Interrupt Enable Register	UARTIER	R/W	8	0x00	18-6
0xB7B00008	FIFO Control Register	UARTFCR	W	8	0x00	18-9
0xB7B00008	Interrupt Identification Register	UARTIIR	R	8	0x01	18-7
0xB7B0000C	Line Control Register	UARTLCR	R/W	8	0x00	18-11
0xB7B00010	Modem Control Register	UARTMCR	R/W	8	0x00	18-13
0xB7B00014	Line Status Register	UARTLSR	R/W	8	0x60	18-15
0xB7B00018	Modem Status Register	UARTMSR	R/W	8	Contents depend on pin states	18-18
0xB7B0001C	Scratch Register	UARTSCR	R/W	8	Indeterminate	18-20



Address	Name	Abbreviation	R/W	Size	Initial Value	Ref. Pages
0xB7B01000	Synchronous SIO transmit-receive buffer register	SSIOBUF	R/W	8	0x00	19-3
0xB7B01004	Synchronous SIO status register	SSIOST	R/W	8	0x00	19-3
0xB7B01008	Synchronous SIO interrupt request register	SSIOINT	R/W	8	0x00	19-4
0xB7B0100C	Synchronous SIO interrupt enable register	SSIOINTEN	R/W	8	0x00	19-5
0xB7B01010	Synchronous SIO transmit-receive control register	SSIOCON	R/W	8	0x00	19-6
0xB7B01014	Synchronous SIO test control register	SSIOTSCON	R/W	8	0x00	19-7
0xB7D00000	PWM register 0	PWR0	R/W	16	0x0000	16-3
0xB7D00004	PWM period register 0	PWCY0	R/W	16	0x0000	16-4
0xB7D00008	PWM counter 0	PWC0	R/W	16	0x0000	16-5
0xB7D0000C	PWM control register 0	PWCON0	R/W	16	0x0000	16-6
0xB7D00020	PWM register 1	PWR1	R/W	16	0x0000	16-3
0xB7D00024	PWM period register 1	PWCY1	R/W	16	0x0000	16-4
0xB7D00028	PWM counter 1	PWC1	R/W	16	0x0000	16-5
0xB7D0002C	PWM control register 1	PWCON1	R/W	16	0x0000	16-6
0xB7D0003C	PWM Interrupt status register	PWINTSTS	R/W	16	0x0000	16-7
0xB7E00000	Watchdog timer control register	WDTCN	W	8	—	14-2
0xB7E00004	Time base counter control register	WDTBCN	R/W	8	0x00	14-3
0xB7E00014	Status register	WDSTAT	R/W	8	0x00	14-5
0xB7F00000	Timer 0 control register	TIMECNTL0	R/W	16	0x0000	15-7
0xB7F00004	Timer 0 base register	TIMEBASE0	R/W	16	0x0000	15-9
0xB7F00008	Timer 0 counter register	TIMECNT0	R	16	0x0000	15-10
0xB7F0000C	Timer 0 compare register	TIMECMP0	R/W	16	0xFFFF	15-11
0xB7F00010	Timer 0 status register	TIMESTAT0	R/W	16	0x0000	15-12
0xB7F00020	Timer 1 control register	TIMECNTL1	R/W	16	0x0000	15-7
0xB7F00024	Timer 1 base register	TIMEBASE1	R/W	16	0x0000	15-9
0xB7F00028	Timer 1 counter register	TIMECNT1	R	16	0x0000	15-10
0xB7F0002C	Timer 1 compare register	TIMECMP1	R/W	16	0xFFFF	15-11
0xB7F00030	Timer 1 status register	TIMESTAT1	R/W	16	0x0000	15-12
0xB7F00040	Timer 2 control register	TIMECNTL2	R/W	16	0x0000	15-7
0xB7F00044	Timer 2 base register	TIMEBASE2	R/W	16	0x0000	15-9
0xB7F00048	Timer 2 counter register	TIMECNT2	R	16	0x0000	15-10
0xB7F0004C	Timer 2 compare register	TIMECMP2	R/W	16	0xFFFF	15-11
0xB7F00050	Timer 2 status register	TIMESTAT2	R/W	16	0x0000	15-12
0xB7F00060	Timer 3 control register	TIMECNTL3	R/W	16	0x0000	15-7
0xB7F00064	Timer 3 base register	TIMEBASE3	R/W	16	0x0000	15-9
0xB7F00068	Timer 3 counter register	TIMECNT3	R	16	0x0000	15-10
0xB7F0006C	Timer 3 compare register	TIMECMP3	R/W	16	0xFFFF	15-11
0xB7F00070	Timer 3 status register	TIMESTAT3	R/W	16	0x0000	15-12
0xB7F00080	Timer 4 control register	TIMECNTL4	R/W	16	0x0000	15-7
0xB7F00084	Timer 4 base register	TIMEBASE4	R/W	16	0x0000	15-9
0xB7F00088	Timer 4 counter register	TIMECNT4	R	16	0x0000	15-10

Address	Name	Abbreviation	R/W	Size	Initial Value	Ref. Pages
0xB7F0008C	Timer 4 compare register	TIMECMP4	R/W	16	0xFFFF	15-11
0xB7F00090	Timer 4 status register	TIMESTAT4	R/W	16	0x0000	15-12
0xB7F000A0	Timer 5 control register	TIMECNTL5	R/W	16	0x0000	15-7
0xB7F000A4	Timer 5 base register	TIMEBASE5	R/W	16	0x0000	15-9
0xB7F000A8	Timer 5 counter register	TIMECNT5	R	16	0x0000	15-10
0xB7F000AC	Timer 5 compare register	TIMECMP5	R/W	16	0xFFFF	15-11
0xB7F000B0	Timer 5 status register	TIMESTAT5	R/W	16	0x0000	15-12
0xB8000004	Clock stop register	CLKSTP	R/W	32	0x00000000	7-7
0xB8000008	Clock gear control register	CGBCNT0	R/W	32	0x00000000	7-9
0xB800000C	Clock wait register	CKWT	R/W	32	0x000000FF	7-10
0xB8000010	Remap control register	RMPCON	R/W	32	0x00000000	3-4
0xB8001004	System timer enable register	TMEN	R/W	32	0x00000000	15-4
0xB8001008	System timer reload register	TMRLR	R/W	32	0x00000000	15-5
0xB8001010	System timer overflow register	TMOVFR	R/W	32	0x00000000	15-6
0xB8002000	SIO transfer buffer register	SIobuf	R/W	32	0x00000000	17-3
0xB8002004	SIO status register	SIOSTA	R/W	32	0x00000000	17-4
0xB8002008	SIO control register	SIOCON	R/W	32	0x00000000	17-6
0xB800200C	Baud rate control register	SIOBCN	R/W	32	0x00000000	17-8
0xB8002014	Baud rate timer register	SIOBT	R/W	32	0x00000000	17-9
0xB8002018	SIO test control register	SIOTCN	R/W	32	0x00000000	17-10





#### Notes for Mounting the Surface Mount Type Package

The surface mount type packages are very susceptible to heat in reflow mounting and humidity absorbed in storage. Therefore, before you perform reflow mounting, contact Oki's responsible sales person for the product name, package name, pin number, package code and desired mounting conditions (reflow method, temperature and times).

## **Revision History**

---

## Revision History

Document No.	Date	Page		Description
		Previous Edition	Current Edition	
PEUL675001-01	Apr.16, 2003	–	–	Preliminary edition 1
FEUL675001-01	Jan.31, 2004	–	–	Final edition 1
		3-1, 3-6	3-1, 3-6	Correct a misdescription about the size of the ROMSEL register to 16-bit.
		3-4 to 3-8	3-4 to 3-8	Change the structure of sections. Add section 3.4.3 "Notes of Address map".
		3-6	3-5	Add the BSELM bit to the ROMSEL register.
		3-7	3-6	Correct misdescriptions in the table.
		5-2	5-2	Add the CLKMD[1] and CLKMD[0] pins for ML675001 series in the Pin list. Add a note about CLKMD[1:0] pins.
		7-10	7-10	Add a description about the CKWT register after a reset.
		8-22	8-22	Modify the table of the condition that each IRQ[n] bit is set or cleared.
		9-4	9-4	Correct a misdescription in the table about cacheable control.
		10-1	10-1	Modify access clocks of built-in SRAM and built-in FLASH ROM.
		11-2 to 11-3	11-2 to 11-3	Add 8 bit bus width setting to ROMBW and RAMBW bit field of the BWC register for ML675001 series.
		11-26	11-26	Add a note about XWR for ML675001 series. Correct a misdescription about XWAIT.
		12-21	12-21	Add notes about DMA.
		14-5	14-5	Add notes to WDTIST and IVTIST bit field of the WDSTAT register.
		15-13	15-13	Rewrite the description of the section 15.3.1.
		16-2	16-2	Correct a misdescription in the pin list.
		18-24	18-24	Correct a misdescription about baud rate clock.
		18-26	18-26	Add a note about the frequency of CCLK and HCLK.
		19-1 19-7 19-9	19-1 19-7 19-9	Modify the operating clock for SSIO from CCLK to HCLK
		20-3	20-3	Correct a misdescription about the I2CEN bit of the I2CCON register.
22-1	22-1	Add write protect/unprotect time.		
22-5	22-5	Add a sample connection for programming the FLASH memory.		
23-8	23-8	Add IDC CODE.		
24-1 to 24-42	24-1 to 24-48	Rewrite the electrical characteristics.		

## NOTICE

1. The information contained herein can change without notice owing to product and/or technical improvements. Before using the product, please make sure that the information being referred to is up-to-date.
2. The outline of action and examples for application circuits described herein have been chosen as an explanation for the standard action and performance of the product. When planning to use the product, please ensure that the external conditions are reflected in the actual circuit, assembly, and program designs.
3. When designing your product, please use our product below the specified maximum ratings and within the specified operating ranges including, but not limited to, operating voltage, power dissipation, and operating temperature.
4. Oki assumes no responsibility or liability whatsoever for any failure or unusual or unexpected operation resulting from misuse, neglect, improper installation, repair, alteration or accident, improper handling, or unusual physical or electrical stress including, but not limited to, exposure to parameters beyond the specified maximum ratings or operation outside the specified operating range.
5. Neither indemnity against nor license of a third party's industrial and intellectual property right, etc. is granted by us in connection with the use of the product and/or the information and drawings contained herein. No responsibility is assumed by us for any infringement of a third party's right which may result from the use thereof.
6. The products listed in this document are intended for use in general electronics equipment for commercial applications (e.g., office automation, communication equipment, measurement equipment, consumer electronics, etc.). These products are not, unless specifically authorized by Oki, authorized for use in any system or application that requires special or enhanced quality and reliability characteristics nor in any system or application where the failure of such system or application may result in the loss or damage of property, or death or injury to humans.  
Such applications include, but are not limited to, traffic and automotive equipment, safety devices, aerospace equipment, nuclear power control, medical equipment, and life-support systems.
7. Certain products in this document may need government approval before they can be exported to particular countries. The purchaser assumes the responsibility of determining the legality of export of these products and will take appropriate and necessary steps at their own expense for these.
8. No part of the contents contained herein may be reprinted or reproduced without our prior permission.



## **Oki REGIONAL SALES OFFICES**

---

### **Northwest Area**

785 N. Mary Avenue  
Sunnyvale, CA 94085  
Tel: 408/720-1900  
Fax: 408/720-8965

### **Northeast Area**

Shattuck Office Center  
138 River Road  
Andover, MA 01810  
Tel: 978/688-8687  
Fax: 978/688-8896

### **North Central Area**

1450 East American Lane  
Suite 1400  
Schaumburg, IL 60173  
Tel: 847/330-4494  
Fax: 847/330-4491

### **Southwest and South Central Area**

1902 Wright Place, Suite 200  
Carlsbad, CA 92008  
Tel: 760/918-5830  
Fax: 760/918-5505

### **Southeast Area**

4800 Whitesburg Drive # 30  
PMB 263  
Huntsville, AL 35802  
Tel: 256/468-7037

### **Oki Web Site:**

*<http://www.okisemi.com/us>*

**February 2004**

# **Oki Semiconductor**

### **Corporate Headquarters**

785 N. Mary Avenue  
Sunnyvale, CA 94085-2909  
Tel: 408/720-1900  
Fax: 408/720-1918