21-S3-C2410A-032004

USER'S MANUAL

S3C2410A – 200MHz & 266MHz

32-Bit RISC Microprocessor

Revision 1.0

SAMSUNG ELECTRONICS

S3C2410A

200MHz & 266MHz

32-BIT RISC

MICROPROCESSOR

USER'S MANUAL

Revision 1.0

SAMSUNG ELECTRONICS

Important Notice

The information in this publication has been carefully checked and is believed to be entirely accurate at the time of publication. Samsung assumes no responsibility, however, for possible errors or omissions, or for any consequences resulting from the use of the information contained herein.

Samsung reserves the right to make changes in its products or product specifications with the intent to improve function or design at any time and without notice and is not required to update this documentation to reflect such changes.

This publication does not convey to a purchaser of semiconductor devices described herein any license under the patent rights of Samsung or others.

Samsung makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Samsung assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

"Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by the customer's technical experts.

Samsung products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, for other applications intended to support or sustain life, or for any other application in which the failure of the Samsung product could create a situation where personal injury or death may occur.

Should the Buyer purchase or use a Samsung product for any such unintended or unauthorized application, the Buyer shall indemnify and hold Samsung and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorized use, even if such claim alleges that Samsung was negligent regarding the design or manufacture of said product.

S3C2410A – 200MHz & 266MHz 32-Bit RISC Microprocessor
User's Manual, Revision 1.0 (March 2004)
Publication Number: 21-S3-C2410A-032004

Samsung Electronics' microcontroller business has been awarded full ISO-9001 certification (BSI Certificate No. FM24653). All semiconductor products are designed and manufactured in accordance with the highest quality standards and objectives.

Samsung Electronics Co., Ltd.
San #24 Nongseo-Ri, Giheung- Eup
Yongin-City, Gyeonggi-Do, Korea
C.P.O. Box #37, Suwon 449-900

TEL: (82)-(031)-209-1934
FAX: (82)-(031)-209-1899

Home Page: http://www.samsung.com

Printed in the Republic of Korea

Table of Contents

Chapter 1 Product Overview

Chapter 2 Programmer's model

Table of Contents (Continued)

Chapter 3 ARM Instruction set

Table of Contents (Continued)

Chapter 3 ARM Instruction set (Continued)

Table of Contents (Continued)

Chapter 3 ARM Instruction set (Continued)

Chapter 4 Thumb Instruction Set

Table of Contents (Continued)

Chapter 4 Thumb Instruction Set (Continued)

Table of Contents (Continued)

Chapter 4 Thumb Instruction Set (Continued)

Chapter 5 Memory Controller

Table of Contents (Continued)

Chapter 6 NAND Flash Controller

Chapter 7 Clock & Power Management

Table of Contents (Continued)

Chapter 8 DMA

Chapter 9 I/O Ports

Table of Contents (Continued)

Chapter 10 PWM Timer

Chapter 11 UART

Table of Contents (Continued)

Chapter 12 USB Host Controller

Chapter 13 USB Device Controller

Table of Contents (Continued)

Chapter 14 Interrupt Controller

Chapter 15 LCD Controller

Table of Contents (Continued)

Chapter 16 ADC & Touch Screen Interface

Chapter 17 Real Time Clock (RTC)

Table of Contents (Continued)

Chapter 18 WatchDog Timer

Chapter 19 MMC/SD/SDIO Host controller

Chapter 20 IIC-BUS Interface

Table of Contents (Continued)

Chapter 21 IIS-BUS Interface

Chapter 22 SPI Interface

Table of Contents (Continued)

Chapter 23 BUS Priorities

Chapter 24 Electrical Data

Chapter 25 Mechanical Data

Table of Contents (Continued)

Appendix 1- ARM920T Introduction

Appendix 2- Programmer's Model

Appendix 3- MMU

Table of Contents (Concluded)

Appendix 3- MMU (Continued)

Appendix 4- Caches, Write Buffer

Appendix 5- Clock Modes

List of Figures

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Concluded)

List of Tables

List of Tables (Continued)

List of Table (Concluded)

1 PRODUCT OVERVIEW

INTRODUCTION

This manual describes SAMSUNG's S3C2410A 16/32-bit RISC microprocessor. This product is designed to provide hand-held devices and general applications with cost-effective, low-power, and high-performance micro-controller solution in small die size. To reduce total system cost, the S3C2410A includes the following components separate 16KB Instruction and 16KB Data Cache, MMU to handle virtual memory management, LCD Controller (STN & TFT), NAND Flash Boot Loader, System Manager (chip select logic and SDRAM Controller), 3-ch UART, 4-ch DMA, 4-ch Timers with PWM, I/O Ports, RTC, 8-ch 10-bit ADC and Touch Screen Interface, IIC-BUS Interface, IIS-BUS Interface, USB Host, USB Device, SD Host & Multi-Media Card Interface, 2-ch SPI and PLL for clock generation.

The S3C2410A was developed using an ARM920T core, 0.18um CMOS standard cells and a memory complier. Its low-power, simple, elegant and fully static design is particularly suitable for cost- and power-sensitive applications. It adopts a new bus architecture called Advanced Microcontroller Bus Architecture (AMBA).

The S3C2410A offers outstanding features with its CPU core, a 16/32-bit ARM920T RISC processor designed by Advanced RISC Machines, Ltd. The ARM920T implements MMU, AMBA BUS, and Harvard cache architecture with separate 16KB instruction and 16KB data caches, each with an 8-word line length.

By providing a complete set of common system peripherals, the S3C2410A minimizes overall system costs and eliminates the need to configure additional components. The integrated on-chip functions that are described in this document include:

- 1.8V/2.0V int., 3.3V memory, 3.3V external I/O microprocessor with 16KB I-Cache/16KB D-Cache/MMU
- External memory controller (SDRAM Control and Chip Select logic)
- LCD controller (up to 4K color STN and 256K color TFT) with 1-ch LCD-dedicated DMA
- 4-ch DMAs with external request pins
- 3-ch UART (IrDA1.0, 16-Byte Tx FIFO, and 16-Byte Rx FIFO) / 2-ch SPI
- 1-ch multi-master IIC-BUS/1-ch IIS-BUS controller
- SD Host interface version 1.0 & Multi-Media Card Protocol version 2.11 compatible
- 2-port USB Host /1- port USB Device (ver 1.1)
- 4-ch PWM timers & 1-ch internal timer
- Watch Dog Timer
- 117-bit general purpose I/O ports / 24-ch external interrupt source
- Power control: Normal, Slow, Idle and Power-off mode
- 8-ch 10-bit ADC and Touch screen interface
- RTC with calendar function
- On-chip clock generator with PLL

FEATURES

Architecture

- Integrated system for hand-held devices and general embedded applications
- 16/32-Bit RISC architecture and powerful instruction set with ARM920T CPU core
- Enhanced ARM architecture MMU to support WinCE, EPOC 32 and Linux
- Instruction cache, data cache, write buffer and Physical address TAG RAM to reduce the effect of main memory bandwidth and latency on performance
- ARM920T CPU core supports the ARM debug architecture.
- Internal Advanced Microcontroller Bus Architecture (AMBA) (AMBA2.0, AHB/APB)

System Manager

- Little/Big Endian support
- Address space: 128M bytes for each bank (total 1G bytes)
- Supports programmable 8/16/32-bit data bus width for each bank
- Fixed bank start address from bank 0 to bank 6
- Programmable bank start address and bank size for bank 7
- Eight memory banks:
  - – Six memory banks for ROM, SRAM, and others.
  - – Two memory banks for ROM/SRAM/ Synchronous DRAM
- Fully Programmable access cycles for all memory banks
- Supports external wait signals to expend the bus cycle
- Supports self-refresh mode in SDRAM for power-down
- Supports various types of ROM for booting (NOR/NAND Flash, EEPROM, and others)

NAND Flash Boot Loader

- Supports booting from NAND flash memory
- 4KB internal buffer for booting
- Supports storage memory for NAND flash memory after booting

Cache Memory

- 64-way set-associative cache with I-Cache (16KB) and D-Cache (16KB)
- 8words length per line with one valid bit and two dirty bits per line
- Pseudo random or round robin replacement algorithm
- Write-through or write-back cache operation to update the main memory
- The write buffer can hold 16 words of data and four addresses.

Clock & Power Manager

- On-chip MPLL and UPLL:
  UPLL generates the clock to operate USB Host/Device.
  MPLL generates the clock to operate MCU at maximum 266MHz @ 2.0V.
- Clock can be fed selectively to each function block by software.
- Power mode: Normal, Slow, Idle, and Power-off mode
  Normal mode: Normal operating mode
  Slow mode: Low frequency clock without PLL
  Idle mode: The clock for only CPU is stopped.
  Power-off mode: The Core power including all peripherals is shut down.
- Woken up by EINT[15:0] or RTC alarm interrupt from Power-Off mode

FEATURES (Continued)

Interrupt Controller

- 55 Interrupt sources
  (One Watch dog timer, 5 timers, 9 UARTs, 24 external interrupts, 4 DMA, 2 RTC, 2 ADC, 1 IIC, 2 SPI, 1 SDI, 2 USB, 1 LCD, and 1 Battery Fault)
- Level/Edge mode on external interrupt source
- Programmable polarity of edge and level
- Supports Fast Interrupt request (FIQ) for very urgent interrupt request

Timer with Pulse Width Modulation (PWM)

- 4-ch 16-bit Timer with PWM / 1-ch 16-bit internal timer with DMA-based or interrupt-based operation
- Programmable duty cycle, frequency, and polarity
- Dead-zone generation
- Supports external clock sources

RTC (Real Time Clock)

- Full clock feature: second, minute, hour, date, day, month, and year
- 32.768 KHz operation
- Alarm interrupt
- Time tick interrupt

General Purpose Input/Output Ports

- 24 external interrupt ports
- multiplexed input/output ports

UART

- 3-channel UART with DMA-based or interrupt-based operation
- Supports 5-bit, 6-bit, 7-bit, or 8-bit serial data transmit/receive (Tx/Rx)
- Supports external clocks for the UART operation (UEXTCLK)
- Programmable baud rate
- Supports IrDA 1.0
- Loopback mode for testing
- Each channel has internal 16-byte Tx FIFO and 16-byte Rx FIFO.

DMA Controller

- 4-ch DMA controller
- Supports memory to memory, IO to memory, memory to IO, and IO to IO transfers
- Burst transfer mode to enhance the transfer rate

A/D Converter & Touch Screen Interface

- 8-ch multiplexed ADC
- Max. 500KSPS and 10-bit Resolution

LCD Controller STN LCD Displays Feature

- Supports 3 types of STN LCD panels: 4-bit dual scan, 4-bit single scan, 8-bit single scan display type
- Supports monochrome mode, 4 gray levels, 16 gray levels, 256 colors and 4096 colors for STN LCD
- Supports multiple screen size
- Typical actual screen size: 640x480, 320x240, 160x160, and others
- Maximum virtual screen size is 4 Mbytes.
- Maximum virtual screen size in 256 color mode: 4096x1024, 2048x2048, 1024x4096, and others

TFT(Thin Film Transistor) Color Displays Feature

- Supports 1, 2, 4 or 8 bpp (bit-per-pixel) palette color displays for color TFT
- Supports 16 bpp non-palette true-color displays for color TFT
- Supports maximum 16M color TFT at 24 bpp mode
- Supports multiple screen size
- Typical actual screen size: 640x480, 320x240, 160x160, and others
- Maximum virtual screen size is 4Mbytes.
- Maximum virtual screen size in 64K color mode: 2048x1024, and others

FEATURES (Continued)

Watchdog Timer

- 16-bit Watchdog Timer
- Interrupt request or system reset at time-out

IIC-Bus Interface

- 1-ch Multi-Master IIC-Bus
- Serial, 8-bit oriented and bi-directional data transfers can be made at up to 100 Kbit/s in Standard mode or up to 400 Kbit/s in Fast mode.

IIS-Bus Interface

- 1-ch IIS-bus for audio interface with DMA-based operation
- Serial, 8-/16-bit per channel data transfers
- 128 Bytes (64-Byte + 64-Byte) FIFO for Tx/Rx
- Supports IIS format and MSB-justified data format

USB Host

- 2-port USB Host
- Complies with OHCI Rev. 1.0
- Compatible with USB Specification version 1.1

USB Device

- 1-port USB Device
- 5 Endpoints for USB Device
- Compatible with USB Specification version 1.1

SD Host Interface

- Compatible with SD Memory Card Protocol version 1.0
- Compatible with SDIO Card Protocol version 1.0
- Bytes FIFO for Tx/Rx
- DMA based or Interrupt based operation
- Compatible with Multimedia Card Protocol version 2.11

SPI Interface

- Compatible with 2-ch Serial Peripheral Interface Protocol version 2.11
- 2x8 bits Shift register for Tx/Rx
- DMA-based or interrupt-based operation

Operating Voltage Range

- Core: 1.8V for 200MHz (S3C2410A-20)
  2.0V for 266MHz (S3C2410A-26)
- Memory & IO: 3.3V

Operating Frequency

- Up to 266MHz

Package

- 272-FBGA

BLOCK DIAGRAM

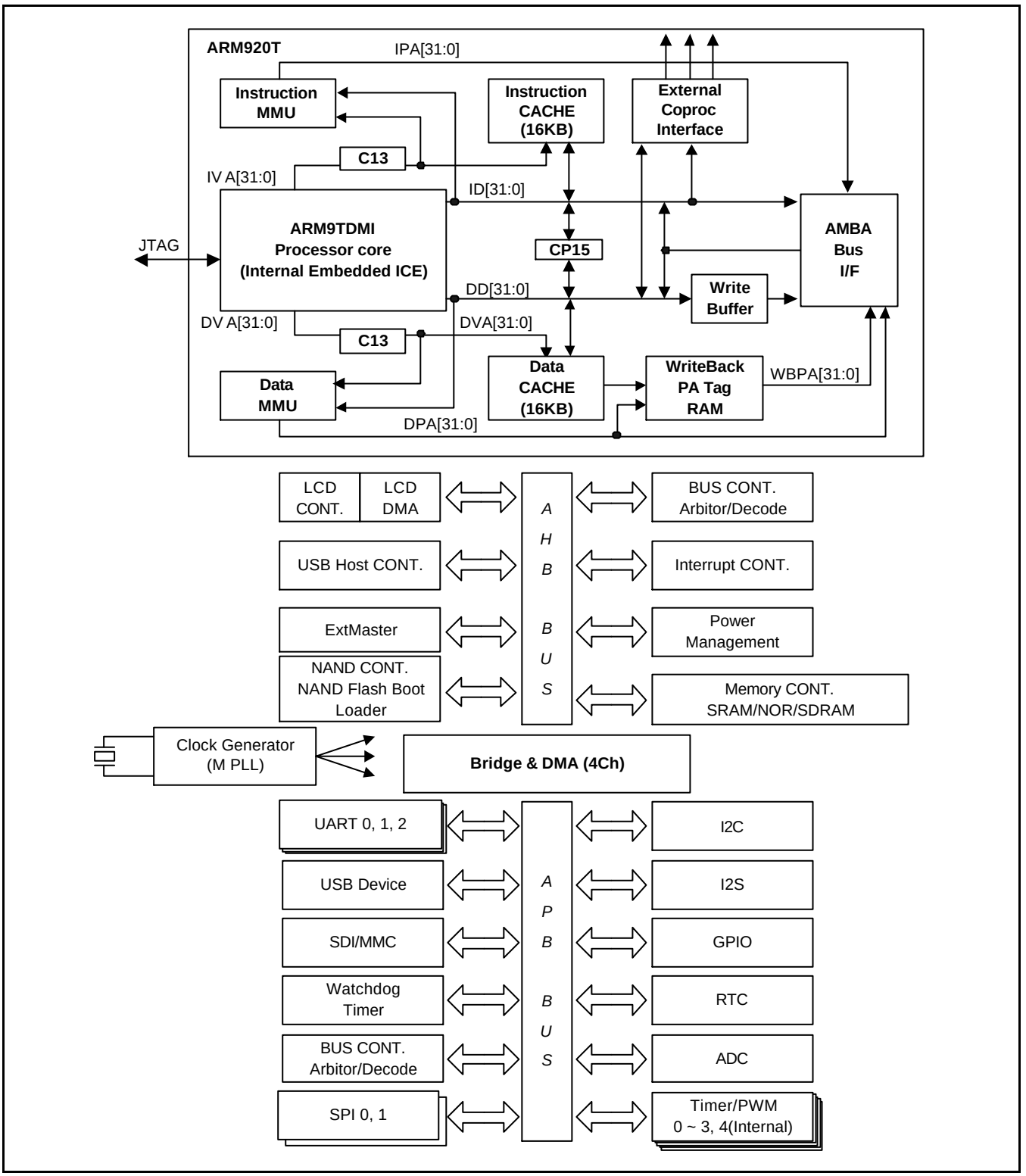


Figure 1-1. S3C2410A Block Diagram

PIN ASSIGNMENTS

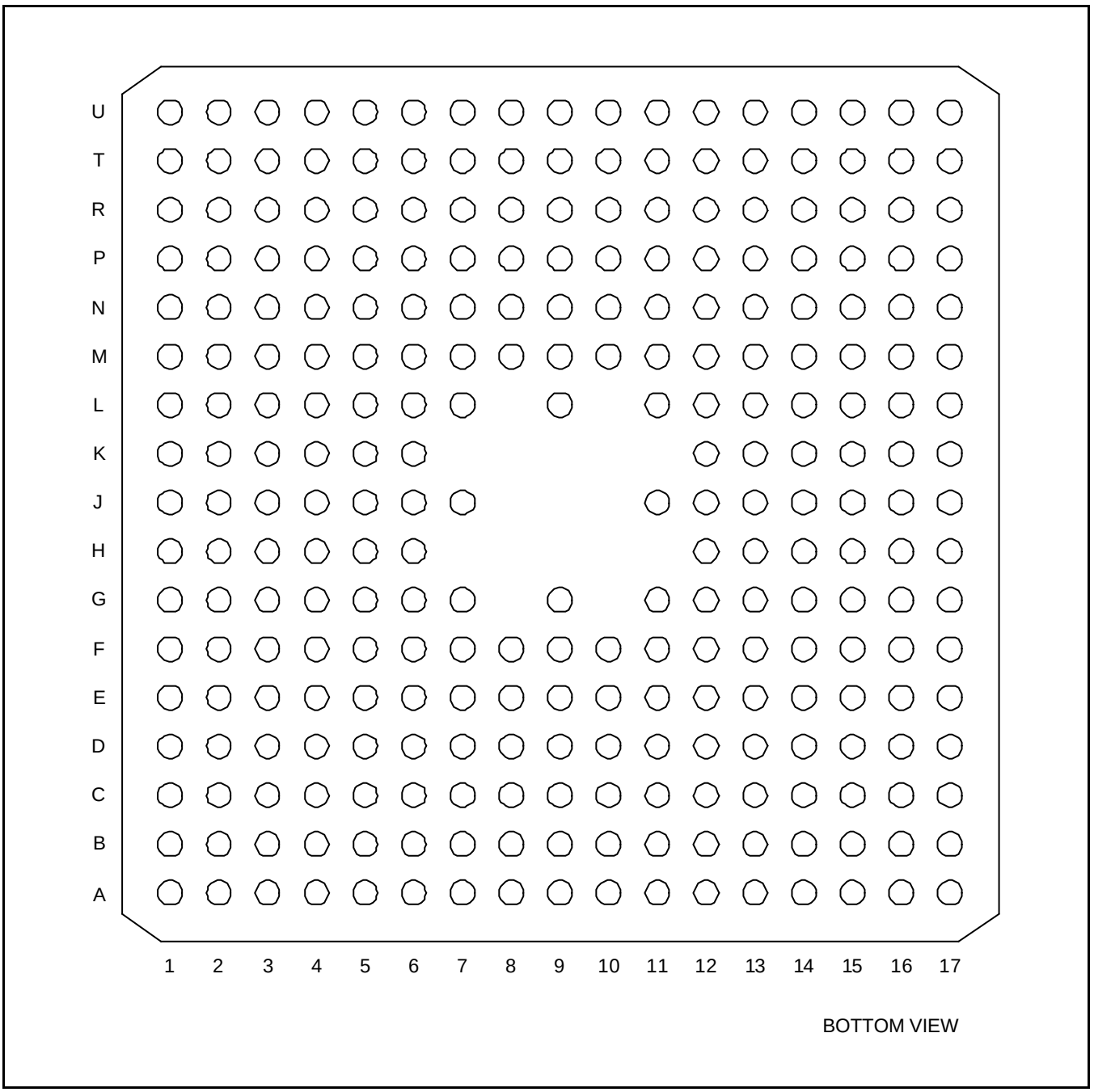


Figure 1-2. S3C2410A Pin Assignments (272-FBGA)

Table 1-1. 272-Pin FBGA Pin Assignments – Pin Number Order

Pin Number	Pin Name	Pin Number	Pin Name	Pin Number	Pin Name
A1	DATA19	B14	ADDR0/GPA0	D10	ADDR19/GPA4
A2	DATA18	B15	nSRAS	D11	VDDi
A3	DATA16	B16	nBE1:nWBE1:DQM1	D12	ADDR10
A4	DATA15	B17	VSSi	D13	ADDR5
A5	DATA11	C1	DATA24	D14	ADDR1
A6	VDDMOP	C2	DATA23	D15	VSSMOP
A7	DATA6	C3	DATA21	D16	SCKE
A8	DATA1	C4	VDDi	D17	nGCS0
A9	ADDR21/GPA6	C5	DATA12	E1	DATA31
A10	ADDR16/GPA1	C6	DATA7	E2	DATA29
A11	ADDR13	C7	DATA4	E3	DATA28
A12	VSSMOP	C8	VDDi	E4	DATA30
A13	ADDR6	C9	ADDR25/GPA10	E5	VDDMOP
A14	ADDR2	C10	VSSMOP	E6	VSSMOP
A15	VDDMOP	C11	ADDR14	E7	DATA3
A16	nBE3:nWBE3:DQM3	C12	ADDR7	E8	ADDR26/GPA11
A17	nBE0:nWBE0:DQM0	C13	ADDR3	E9	ADDR23/GPA8
B1	DATA22	C14	nSCAS	E10	ADDR18/GPA3
B2	DATA20	C15	nBE2:nWBE2:DQM2	E11	VDDMOP
B3	DATA17	C16	nOE	E12	ADDR11
B4	VDDMOP	C17	VDDi	E13	nWE
B5	DATA13	D1	DATA27	E14	nGCS3/GPA14
B6	DATA9	D2	DATA25	E15	nGCS1/GPA12
B7	DATA5	D3	VSSMOP	E16	nGCS2/GPA13
B8	DATA0	D4	DATA26	E17	nGCS4/GPA15
B9	ADDR24/GPA9	D5	DATA14	F1	TOUT1/GPB1
B10	ADDR17/GPA2	D6	DATA10	F2	TOUT0/GPB0
B11	ADDR12	D7	DATA2	F3	VSSMOP
B12	ADDR8	D8	VDDMOP	F4	TOUT2/GPB2
B13	ADDR4	D9	ADDR22/GPA7	F5	VSSOP

Table 1-1. 272-Pin FBGA Pin Assignments – Pin Number Order (Continued)

Pin Number	Pin Name	Pin Number	Pin Name	Pin Number	Pin Name
F6	VSSi	H4	nXDREQ1/GPB8	K13	TXD2/nRTS1/GPH6
F7	DATA8	H5	nTRST	K14	RXD1/GPH5
F8	VSSMOP	H6	TCK	K15	TXD0/GPH2
F9	VSSi	H12	CLE/GPA17	K16	TXD1/GPH4
F10	ADDR20/GPA5	H13	VSSOP	K17	RXD0/GPH3
F11	VSSi	H14	VDDMOP	L1	VD0/GPC8
F12	VSSMOP	H15	VSSi	L2	VD1/GPC9
F13	SCLK0	H16	XTOpll	L3	LCDVF2/GPC7
F14	SCLK1	H17	XTIpll	L4	VD2/GPC10
F15	nGCS5/GPA16	J1	TDI	L5	VDDiarm
F16	nGCS6:nSCS0	J2	VCLK:LCD_HCLK/GPC1	L6	LCDVF1/GPC6
F17	nGCS7:nSCS1	J3	TMS	L7	IICSCL/GPE14
G1	nXBACK/GPB5	J4	LEND:STH/GPC0	L9	EINT11/nSS1/GPG3
G2	nXDACK1/GPB7	J5	TDO	L11	VDDi_UPLL
G3	TOUT3/GPB3	J6	VLINE:HSYNC:CPV/GPC2	L12	nRTS0/GPH1
G4	TCLK0/GPB4	J7	VSSiarm	L13	UPLLCAP
G5	nXBREQ/GPB6	J11	EXTCLK	L14	nCTS0/GPH0
G6	VDDalive	J12	nRESET	L15	EINT6/GPF6
G7	VDDiarm	J13	VDDi	L16	UEXTCLK/GPH8
G9	VSSMOP	J14	VDDalive	L17	EINT7/GPF7
G11	ADDR15	J15	PWREN	M1	VSSiarm
G12	ADDR9	J16	nRSTOUT/GPA21	M2	VD5/GPC13
G13	nWAIT	J17	nBATT_FLT	M3	VD3/GPC11
G14	ALE/GPA18	K1	VDDOP	M4	VD4/GPC12
G15	nFWE/GPA19	K2	VM:VDEN:TP/GPC4	M5	VSSiarm
G16	nFRE/GPA20	K3	VDDiarm	M6	VDDOP
G17	nFCE/GPA22	K4	VFRAME:VSYNC:STV/GPC3	M7	VDDiarm
H1	VSSiarm	K5	VSSOP	M8	IICSDA/GPE15
H2	nXDACK0/GPB9	K6	LCDVF0/GPC5	M9	VSSiarm
H3	nXDREQ0/GPB10	K12	RXD2/nCTS1/GPH7	M10	DP1/PDP0

Table 1-1. 272-Pin FBGA Pin Assignments – Pin Number Order (Continued)

Pin Number	Pin Name	Pin Number	Pin Name	Pin Number	Pin Name
M11	EINT23/nYPON/GPG15	P8	SPICLK0/GPE13	T5	I2SLRCK/GPE0
M12	RTCVDD	P9	EINT12/LCD_PWREN/GPG4	T6	SDCLK/GPE5
M13	VSSi_MPLL	P10	EINT18/GPG10	T7	SPIMISO0/GPE11
M14	EINT5/GPF5	P11	EINT20/XMON/GPG12	T8	EINT10/nSS0/GPG2
M15	EINT4/GPF4	P12	VSSOP	T9	VSSOP
M16	EINT2/GPF2	P13	DP0	T10	EINT17/GPG9
M17	EINT3/GPF3	P14	VDDi_MPLL	T11	EINT22/YMON/GPG14
N1	VD6/GPC14	P15	VDDA_ADC	T12	DN0
N2	VD8/GPD0	P16	XTIrtc	T13	OM3
N3	VD7/GPC15	P17	MPLLCAP	T14	VSSA_ADC
N4	VD9/GPD1	R1	VDDiarm	T15	AIN1
N5	VDDiarm	R2	VD14/GPD6	T16	AIN3
N6	CDCLK/GPE2	R3	VD17/GPD9	T17	AIN5
N7	SDDAT1/GPE8	R4	VD18/GPD10	U1	VD15/GPD7
N8	VSSiarm	R5	VSSOP	U2	VD19/GPD11
N9	VDDOP	R6	SDDAT0/GPE7	U3	VD21/GPD13
N10	VDDiarm	R7	SDDAT3/GPE10	U4	VSSiarm
N11	DN1/PDN0	R8	EINT8/GPG0	U5	I2SSDI/nSS0/GPE3
N12	Vref	R9	EINT14/SPIMOSI1/GPG6	U6	I2SSDO/I2SSDI/GPE4
N13	AIN7	R10	EINT15/SPICLK1/GPG7	U7	SPIMOSI0/GPE12
N14	EINT0/GPF0	R11	EINT19/TCLK1/GPG11	U8	EINT9/GPG1
N15	VSSi_UPLL	R12	CLKOUT0/GPH9	U9	EINT13/SPIMISO1/GPG5
N16	VDDOP	R13	R/nB	U10	EINT16/GPG8
N17	EINT1/GPF1	R14	OM0	U11	EINT21/nXPON/GPG13
P1	VD10/GPD2	R15	AIN4	U12	CLKOUT1/GPH10
P2	VD12/GPD4	R16	AIN6	U13	NCON
P3	VD11/GPD3	R17	XTOrtc	U14	OM2
P4	VD23/nSS0/GPD15	T1	VD13/GPD5	U15	OM1
P5	I2SSCLK/GPE1	T2	VD16/GPD8	U16	AIN0
P6	SDCMD/GPE6	T3	VD20/GPD12	U17	AIN2
P7	SDDAT2/GPE9	T4	VD22/nSS1/GPD14	–	–

Table 1-2. 272-Pin FBGA Pin Assignments

Pin Number	Pin Name	Default Function	I/O State @BUS REQ	I/O State @PWR-off	I/O State @nRESET	I/O Type
C3	DATA21	DATA21	Hi-z	Hi-z	I	t12
B1	DATA22	DATA22	Hi-z	Hi-z	I	t12
C2	DATA23	DATA23	Hi-z	Hi-z	I	t12
D3	VSSMOP	VSSMOP	P	P	P	s3o
E5	VDDMOP	VDDMOP	P	P	P	d3o
C1	DATA24	DATA24	Hi-z	Hi-z	I	t12
D2	DATA25	DATA25	Hi-z	Hi-z	I	t12
D4	DATA26	DATA26	Hi-z	Hi-z	I	t12
D1	DATA27	DATA27	Hi-z	Hi-z	I	t12
E3	DATA28	DATA28	Hi-z	Hi-z	I	t12
E2	DATA29	DATA29	Hi-z	Hi-z	I	t12
E4	DATA30	DATA30	Hi-z	Hi-z	I	t12
E1	DATA31	DATA31	Hi-z	Hi-z	I	t12
F3	VSSMOP	VSSMOP	P	P	P	s3o
F5	VSSOP	VSSOP	P	P	P	s3o
F2	TOUT0/GPB0	GPB0	–/–	O(L)/–	I	t8
F1	TOUT1/GPB1	GPB1	–/–	O(L)/–	I	t8
F4	TOUT2/GPB2	GPB2	–/–	O(L)/–	I	t8
G3	TOUT3/GPB3	GPB3	–/–	O(L)/–	I	t8
G4	TCLK0/GPB4	GPB4	–/–	–/–	I	t8
G1	nXBACK/GPB5	GPB5	–/–	–/–	I	t8
G5	nXBREQ/GPB6	GPB6	–/–	–/–	I	t8
G2	nXDACK1/GPB7	GPB7	–/–	–/–	I	t8
G6	VDDalive	VDDalive	P	P	P	d1i
G7	VDDiarm	VDDiarm	P	P	P	d1c
H1	VSSiarm	VSSiarm	P	P	P	s3i
H4	nXDREQ1/GPB8	GPB8	–/–	–/–	I	t8
H2	nXDACK0/GPB9	GPB9	–/–	–/–	I	t8
H3	nXDREQ0/GPB10	GPB10	–/–	–/–	I	t8
H5	nTRST	nTRST	I	I	I	is
H6	TCK	TCK	I	I	I	is
J1	TDI	TDI	I	I	I	is
J3	TMS	TMS	I	I	I	is

Table 1-2. 272-Pin FBGA Pin Assignments (Continued)

Pin Number	Pin Name	Default Function	I/O State @BUS REQ	I/O State @PWR-off	I/O State @nRESET	I/O Type
J5	TDO	TDO	O	O	O	ot
J4	LEND:STH/GPC0	GPC0	–/–	O(L)/–	I	t8
J2	VCLK:LCD_HCLK/GPC1	GPC1	–/–	O(L)/–	I	t8
J6	VLINE:HSYNC:CPV/GPC2	GPC2	–/–	O(L)/–	I	t8
K3	VDDiarm	VDDiarm	P	P	P	d1c
J7	VSSiarm	VSSiarm	P	P	P	s3i
K2	VM:VDEN:TP/GPC4	GPC4	–/–	O(L)/–	I	t8
K4	VFRAME:VSYNC:STV/GPC3	GPC3	–/–	O(L)/–	I	t8
K1	VDDOP	VDDOP	P	P	P	d3o
K5	VSSOP	VSSOP	P	P	P	s3o
K6	LCDVF0/GPC5	GPC5	–/–	O(L)/–	I	t8
L6	LCDVF1/GPC6	GPC6	–/–	O(L)/–	I	t8
L3	LCDVF2/GPC7	GPC7	–/–	O(L)/–	I	t8
L1	VD0/GPC8	GPC8	–/–	O(L)/–	I	t8
L2	VD1/GPC9	GPC9	–/–	O(L)/–	I	t8
L4	VD2/GPC10	GPC10	–/–	O(L)/–	I	t8
M3	VD3/GPC11	GPC11	–/–	O(L)/–	I	t8
L5	VDDiarm	VDDiarm	P	P	P	d1c
M1	VSSiarm	VSSiarm	P	P	P	s3i
M4	VD4/GPC12	GPC12	–/–	O(L)/–	I	t8
M2	VD5/GPC13	GPC13	–/–	O(L)/–	I	t8
N1	VD6/GPC14	GPC14	–/–	O(L)/–	I	t8
N3	VD7/GPC15	GPC15	–/–	O(L)/–	I	t8
N2	VD8/GPD0	GPD0	–/–	O(L)/–	I	t8
N4	VD9/GPD1	GPD1	–/–	O(L)/–	I	t8
P1	VD10/GPD2	GPD2	–/–	O(L)/–	I	t8
P3	VD11/GPD3	GPD3	–/–	O(L)/–	I	t8
P2	VD12/GPD4	GPD4	–/–	O(L)/–	I	t8
R1	VDDiarm	VDDiarm	P	P	P	d1c
M5	VSSiarm	VSSiarm	P	P	P	s3i
T1	VD13/GPD5	GPD5	–/–	O(L)/–	I	t8
R2	VD14/GPD6	GPD6	–/–	O(L)/–	I	t8
U1	VD15/GPD7	GPD7	–/–	O(L)/–	I	t8

Table 1-2. 272-Pin FBGA Pin Assignments (Continued)

Pin Number	Pin Name	Default Function	I/O State @BUS REQ	I/O State @PWR-off	I/O State @nRESET	I/O Type
T2	VD16/GPD8	GPD8	–/–	O(L)/–	I	t8
R3	VD17/GPD9	GPD9	–/–	O(L)/–	I	t8
R4	VD18/GPD10	GPD10	–/–	O(L)/–	I	t8
U2	VD19/GPD11	GPD11	–/–	O(L)/–	I	t8
T3	VD20/GPD12	GPD12	–/–	O(L)/–	I	t8
U3	VD21/GPD13	GPD13	–/–	O(L)/–	I	t8
T4	VD22/nSS1/GPD14	GPD14	–/–	O(L)/–	I	t8
P4	VD23/nSS0/GPD15	GPD15	–/–	O(L)/–	I	t8
N5	VDDiarm	VDDiarm	P	P	P	d1c
U4	VSSiarm	VSSiarm	P	P	P	s3i
M6	VDDOP	VDDOP	P	P	P	d3o
R5	VSSOP	VSSOP	P	P	P	s3o
T5	I2SLRCK/GPE0	GPE0	–/–	O(L)/–	I	t8
P5	I2SSCLK/GPE1	GPE1	–/–	O(L)/–	I	t8
N6	CDCLK/GPE2	GPE2	–/–	O(L)/–	I	t8
U5	I2SSDI/nSS0/GPE3	GPE3	–/–/–	–/–/–	I	t8
U6	I2SSDO/I2SSDI/GPE4	GPE4	–/–/–	O(L)/–/–	I	t8
T6	SDCLK/GPE5	GPE5	–/–	O(L)/–	I	t8
P6	SDCMD/GPE6	GPE6	–/–	Hi-z/–	I	t8
R6	SDDAT0/GPE7	GPE7	–/–	Hi-z/–	I	t8
N7	SDDAT1/GPE8	GPE8	–/–	Hi-z/–	I	t8
P7	SDDAT2/GPE9	GPE9	–/–	Hi-z/–	I	t8
R7	SDDAT3/GPE10	GPE10	–/–	Hi-z/–	I	t8
T7	SPIMISO0/GPE11	GPE11	–/–	Hi-z/–	I	t8
U7	SPIMOSI0/GPE12	GPE12	–/–	Hi-z/–	I	t8
P8	SPICLK0/GPE13	GPE13	–/–	Hi-z/–	I	t8
M7	VDDiarm	VDDiarm	P	P	P	d1c
N8	VSSiarm	VSSiarm	P	P	P	s3i
L7	IICSCL/GPE14	GPE14	–/–	Hi-z/–	I	d8
M8	IICSDA/GPE15	GPE15	–/–	Hi-z/–	I	d8
R8	EINT8/GPG0	GPG0	–/–	–/–	I	t8
U8	EINT9/GPG1	GPG1	–/–	–/–	I	t8
T8	EINT10/nSS0/GPG2	GPG2	–/–/–	–/–/–	I	t8

Table 1-2. 272-Pin FBGA Pin Assignments (Continued)

Pin Number	Pin Name	Default Function	I/O State @BUS REQ	I/O State @PWR-off	I/O State @nRESET	I/O Type
L9	EINT11/nSS1/GPG3	GPG3	–/–/–	–/–/–	I	t8
P9	EINT12/LCD_PWREN/GPG4	GPG4	–/–/–	–/O(L)/–	I	t8
U9	EINT13/SPIMISO1/GPG5	GPG5	–/–/–	–/Hi-z/–	I	t8
R9	EINT14/SPIMOSI1/GPG6	GPG6	–/–/–	–/Hi-z/–	I	t8
T9	VSSOP	VSSOP	P	P	P	s3o
N9	VDDOP	VDDOP	P	P	P	d3o
N10	VDDiarm	VDDiarm	P	P	P	d1c
M9	VSSiarm	VSSiarm	P	P	P	s3i
R10	EINT15/SPICLK1/GPG7	GPG7	–/–/–	–/Hi-z/–	I	t8
U10	EINT16/GPG8	GPG8	–/–	–/–	I	t6
T10	EINT17/GPG9	GPG9	–/–	–/–	I	t6
P10	EINT18/GPG10	GPG10	–/–	–/–	I	t6
R11	EINT19/TCLK1/GPG11	GPG11	–/–/–	–/–/–	I	t12
P11	EINT20/XMON/GPG12	GPG12	–/–/–	–/O(L)/–	I	t12
U11	EINT21/nXPON/GPG13	GPG13	–/–/–	–/O(L)/–	I	t12
T11	EINT22/YMON/GPG14	GPG14	–/–/–	–/O(L)/–	I	t12
M11	EINT23/nYPON/GPG15	GPG15	–/–/–	–/O(L)/–	I	t12
R12	CLKOUT0/GPH9	GPH9	–/–	O(L)/–	I	t12
U12	CLKOUT1/GPH10	GPH10	–/–	O(L)/–	I	t12
M10	DP1/PDP0	DP1	–	–	AI	us
N11	DN1/PDN0	DN1	–	–	AI	us
P13	DP0	DP0	–	–	AI	us
T12	DN0	DN0	–	–	AI	us
U13	NCON	NCON	–	–	I	is
R13	R/nB	R/nB	–	–	I	is
T13	OM3	OM3	–	–	I	is
U14	OM2	OM2	–	–	I	is
U15	OM1	OM1	–	–	I	is
R14	OM0	OM0	–	–	I	is
P12	VSSOP	VSSOP	P	P	P	s3o
T14	VSSA_ADC	VSSA_ADC	P	P	P	s3t
N12	Vref	Vref	–	–	AI	ia
U16	AIN0	AIN0	–	–	AI	r10

Table 1-2. 272-Pin FBGA Pin Assignments (Continued)

Pin Number	Pin Name	Default Function	I/O State @BUS REQ	I/O State @PWR-off	I/O State @nRESET	I/O Type
T15	AIN1	AIN1	–	–	AI	r10
U17	AIN2	AIN2	–	–	AI	r10
T16	AIN3	AIN3	–	–	AI	r10
R15	AIN4	AIN4	–	–	AI	r10
T17	AIN5	AIN5	–	–	AI	r10
R16	AIN6	AIN6	–	–	AI	r10
N13	AIN7	AIN7	–	–	AI	r10
P15	VDDA_ADC	VDDA_ADC	P	P	P	d3t
R17	XTOrtc	XTOrtc	–	–	AO	gp
P16	XTIrtc	XTIrtc	–	–	AI	gp
M12	RTCVDD	RTCVDD	P	P	P	d1i
P14	VDDi_MPLL	VDDi_MPLL	P	P	P	d1c
M13	VSSi_MPLL	VSSi_MPLL	P	P	P	s3i
P17	MPLLCAP	MPLLCAP	–	–	AI	gp
L11	VDDi_UPLL	VDDi_UPLL	P	P	P	d1c
N15	VSSi_UPLL	VSSi_UPLL	P	P	P	s3i
L13	UPLLCAP	UPLLCAP	–	–	AI	gp
N16	VDDOP	VDDOP	P	P	P	d3o
N14	EINT0/GPF0	GPF0	–/–	–/–	I	t8
N17	EINT1/GPF1	GPF1	–/–	–/–	I	t8
M16	EINT2/GPF2	GPF2	–/–	–/–	I	t8
M17	EINT3/GPF3	GPF3	–/–	–/–	I	t8
M15	EINT4/GPF4	GPF4	–/–	–/–	I	t8
M14	EINT5/GPF5	GPF5	–/–	–/–	I	t8
L15	EINT6/GPF6	GPF6	–/–	–/–	I	t8
L17	EINT7/GPF7	GPF7	–/–	–/–	I	t8
L16	UEXTCLK/GPH8	GPH8	–/–	–/–	I	t8
L14	nCTS0/GPH0	GPH0	–/–	–/–	I	t8
L12	nRTS0/GPH1	GPH1	–/–	O(H)/–	I	t8
K15	TXD0/GPH2	GPH2	–/–	O(H)/–	I	t8
K17	RXD0/GPH3	GPH3	–/–	–/–	I	t8
K16	TXD1/GPH4	GPH4	–/–	O(H)/–	I	t8
K14	RXD1/GPH5	GPH5	–/–	–/–	I	t8

Table 1-2. 272-Pin FBGA Pin Assignments (Continued)

Pin Number	Pin Name	Default Function	I/O State @BUS REQ	I/O State @PWR-off	I/O State @nRESET	I/O Type
K13	TXD2/nRTS1/GPH6	GPH6	–/–	O(H)/–	I	t8
K12	RXD2/nCTS1/GPH7	GPH7	–/–	–/–	I	t8
J17	nBATT_FLT	nBATT_FLT	–	–	I	is
J16	nRSTOUT/GPA21	nRSTOUT	–/–	O(L)/–	O(L)	b8
J15	PWREN	PWREN	O(H)	O(L)	O(H)	b8
J12	nRESET	nRESET	–	–	I	is
J14	VDDalive	VDDalive	P	P	P	d1i
J11	EXTCLK	EXTCLK	–	–	AI	is
J13	VDDi	VDDi	P	P	P	d1c
H17	XTIpll	XTIpll	–	–	AI	m26
H16	XTOpll	XTOpll	–	–	AO	m26
H15	VSSi	VSSi	P	P	P	s3i
H13	VSSOP	VSSOP	P	P	P	s3o
H14	VDDMOP	VDDMOP	P	P	P	d3o
G17	nFCE/GPA22	nFCE	O(H)/–	O(H)/–	O(H)	b8
G16	nFRE/GPA20	nFRE	O(H)/–	O(H)/–	O(H)	b8
G15	nFWE/GPA19	nFWE	O(H)/–	O(H)/–	O(H)	b8
G14	ALE/GPA18	ALE	O(L)/–	O(L)/–	O(L)	b8
H12	CLE/GPA17	CLE	O(L)/–	O(L)/–	O(L)	b8
G13	nWAIT	nWAIT	–	–	I	is
F17	nGCS7:nSCS1	nGCS7	Hi-z	O(H)	O(H)	ot
F16	nGCS6:nSCS0	nGCS6	Hi-z	O(H)	O(H)	ot
F15	nGCS5/GPA16	nGCS5	Hi-z	O(H)/–	O(H)	ot
E17	nGCS4/GPA15	nGCS4	Hi-z	O(H)/–	O(H)	ot
E14	nGCS3/GPA14	nGCS3	Hi-z	O(H)/–	O(H)	ot
E16	nGCS2/GPA13	nGCS2	Hi-z	O(H)/–	O(H)	ot
E15	nGCS1/GPA12	nGCS1	Hi-z	O(H)/–	O(H)	ot
D17	nGCS0	nGCS0	Hi-z	O(H)	O(H)	ot
D16	SCKE	SCKE	Hi-z	O(L)	O(H)	ot
D15	VSSMOP	VSSMOP	P	P	P	s3o
F14	SCLK1	SCLK1	Hi-z	O(L)	O(SCLK)	t16
C17	VDDi	VDDi	P	P	P	d1c
F13	SCLK0	SCLK0	Hi-z	O(L)	O(SCLK)	t16

Table 1-2. 272-Pin FBGA Pin Assignments (Continued)

Pin Number	Pin Name	Default Function	I/O State @BUS REQ	I/O State @PWR-off	I/O State @nRESET	I/O Type
B17	VSSi	VSSi	P	P	P	s3i
E13	nWE	nWE	Hi-z	O(H)	O(H)	ot
C16	nOE	nOE	Hi-z	O(H)	O(H)	ot
A17	nBE0:nWBE0:DQM0	DQM0	Hi-z	O(H)	O(H)	ot
B16	nBE1:nWBE1:DQM1	DQM1	Hi-z	O(H)	O(H)	ot
C15	nBE2:nWBE2:DQM2	DQM2	Hi-z	O(H)	O(H)	ot
A16	nBE3:nWBE3:DQM3	DQM3	Hi-z	O(H)	O(H)	ot
B15	nSRAS	nSRAS	Hi-z	O(H)	O(H)	ot
C14	nSCAS	nSCAS	Hi-z	O(H)	O(H)	ot
A15	VDDMOP	VDDMOP	P	P	P	d3o
F12	VSSMOP	VSSMOP	P	P	P	s3o
B14	ADDR0/GPA0	ADDR0	Hi-z/–	O(L)/–	O(L)	ot
D14	ADDR1	ADDR1	Hi-z	O(L)	O(L)	ot
A14	ADDR2	ADDR2	Hi-z	O(L)	O(L)	ot
C13	ADDR3	ADDR3	Hi-z	O(L)	O(L)	ot
B13	ADDR4	ADDR4	Hi-z	O(L)	O(L)	ot
D13	ADDR5	ADDR5	Hi-z	O(L)	O(L)	ot
A13	ADDR6	ADDR6	Hi-z	O(L)	O(L)	ot
C12	ADDR7	ADDR7	Hi-z	O(L)	O(L)	ot
B12	ADDR8	ADDR8	Hi-z	O(L)	O(L)	ot
G12	ADDR9	ADDR9	Hi-z	O(L)	O(L)	ot
A12	VSSMOP	VSSMOP	P	P	P	s3o
E11	VDDMOP	VDDMOP	P	P	P	d3o
D12	ADDR10	ADDR10	Hi-z	O(L)	O(L)	ot
E12	ADDR11	ADDR11	Hi-z	O(L)	O(L)	ot
D11	VDDi	VDDi	P	P	P	d1c
F11	VSSi	VSSi	P	P	P	s3i
B11	ADDR12	ADDR12	Hi-z	O(L)	O(L)	ot
A11	ADDR13	ADDR13	Hi-z	O(L)	O(L)	ot
C11	ADDR14	ADDR14	Hi-z	O(L)	O(L)	ot
G11	ADDR15	ADDR15	Hi-z	O(L)	O(L)	ot
A10	ADDR16/GPA1	ADDR16	Hi-z	O(L)/–	O(L)	ot
B10	ADDR17/GPA2	ADDR17	Hi-z	O(L)/–	O(L)	ot

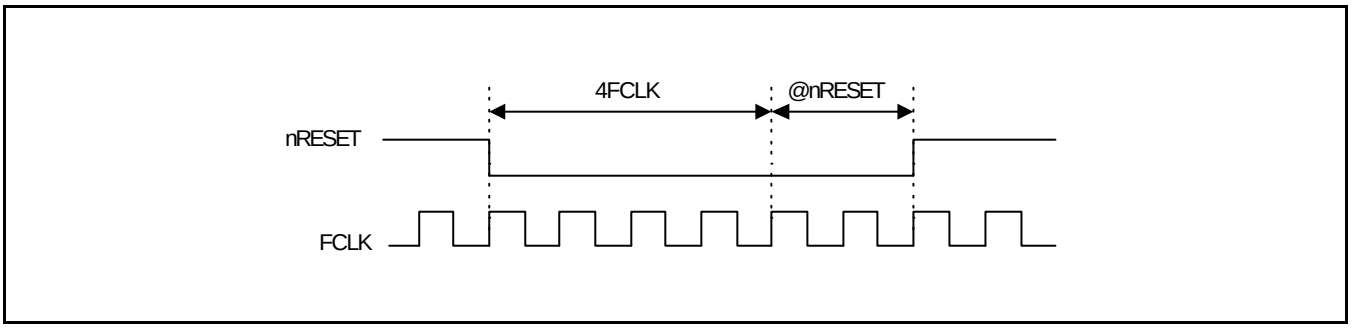
Table 1-2. 272-Pin FBGA Pin Assignments (Continued)

Pin Number	Pin Name	Default Function	I/O State @BUS REQ	I/O State @PWR-off	I/O State @nRESET	I/O Type
C10	VSSMOP	VSSMOP	P	P	P	s3o
E10	ADDR18/GPA3	ADDR18	Hi-z/–	O(L)	O(L)	ot
D10	ADDR19/GPA4	ADDR19	Hi-z/–	O(L)	O(L)	ot
F10	ADDR20/GPA5	ADDR20	Hi-z/–	O(L)	O(L)	ot
A9	ADDR21/GPA6	ADDR21	Hi-z/–	O(L)	O(L)	ot
D9	ADDR22/GPA7	ADDR22	Hi-z/–	O(L)	O(L)	ot
E9	ADDR23/GPA8	ADDR23	Hi-z/–	O(L)	O(L)	ot
B9	ADDR24/GPA9	ADDR24	Hi-z/–	O(L)	O(L)	ot
C9	ADDR25/GPA10	ADDR25	Hi-z/–	O(L)	O(L)	ot
E8	ADDR26/GPA11	ADDR26	Hi-z/–	O(L)	O(L)	ot
C8	VDDi	VDDi	P	P	P	d1c
F9	VSSi	VSSi	P	P	P	s3i
D8	VDDMOP	VDDMOP	P	P	P	d3o
G9	VSSMOP	VSSMOP	P	P	P	s3o
B8	DATA0	DATA0	Hi-z	Hi-z	Hi-z	t12
A8	DATA1	DATA1	Hi-z	Hi-z	Hi-z	t12
D7	DATA2	DATA2	Hi-z	Hi-z	Hi-z	t12
E7	DATA3	DATA3	Hi-z	Hi-z	Hi-z	t12
C7	DATA4	DATA4	Hi-z	Hi-z	Hi-z	t12
B7	DATA5	DATA5	Hi-z	Hi-z	Hi-z	t12
A7	DATA6	DATA6	Hi-z	Hi-z	Hi-z	t12
C6	DATA7	DATA7	Hi-z	Hi-z	Hi-z	t12
A6	VDDMOP	VDDMOP	P	P	P	d3o
F8	VSSMOP	VSSMOP	P	P	P	s3o
F7	DATA8	DATA8	Hi-z	Hi-z	Hi-z	t12
B6	DATA9	DATA9	Hi-z	Hi-z	Hi-z	t12
D6	DATA10	DATA10	Hi-z	Hi-z	Hi-z	t12
A5	DATA11	DATA11	Hi-z	Hi-z	Hi-z	t12
C5	DATA12	DATA12	Hi-z	Hi-z	Hi-z	t12
B5	DATA13	DATA13	Hi-z	Hi-z	Hi-z	t12
D5	DATA14	DATA14	Hi-z	Hi-z	Hi-z	t12
A4	DATA15	DATA15	Hi-z	Hi-z	Hi-z	t12
B4	VDDMOP	VDDMOP	P	P	P	d3o

Table 1-2. 272-Pin FBGA Pin Assignments (Continued)

Pin Number	Pin Name	Default Function	I/O State @BUS REQ	I/O State @PWR-off	I/O State @nRESET	I/O Type
E6	VSSMOP	VSSMOP	P	P	P	s3o
C4	VDDi	VDDi	P	P	P	d1c
F6	VSSi	VSSi	P	P	P	s3i
A3	DATA16	DATA16	Hi-z	Hi-z	Hi-z	t12
B3	DATA17	DATA17	Hi-z	Hi-z	Hi-z	t12
A2	DATA18	DATA18	Hi-z	Hi-z	Hi-z	t12
A1	DATA19	DATA19	Hi-z	Hi-z	Hi-z	t12
B2	DATA20	DATA20	Hi-z	Hi-z	Hi-z	t12

NOTES:
1. The @BUS REQ. shows the pin states at the external bus, which is used by the other bus master.
2. ' – ' mark indicates the unchanged pin state at Bus Request mode.
3. Hi-z or Pre means Hi-z or Previous state and it is determined by the setting of MISCCR register.
4. AI/AO means analog input/analog output.
5. P, I, and O mean power, input and output respectively.
6. The I/O state @nRESET shows the pin status in the @nRESET duration below.



7. The table below shows I/O types and the descriptions.

I/O Type	Descriptions
d1i(vdd1ih), s3i(vss3i)	1.8V / 2.0V V_{DD}/V_{SS} for internal logic
d1c(vdd1ih_core), s3i(vss3i)	1.8V / 2.0V V_{DD}/V_{SS} for internal logic without input driver
d3o(vdd3op), s3o(vss3op)	3.3V V_{DD}/V_{SS} for external logic
d3t(vdd3t_abb), s3t(vss3t_abb)	3.3V V_{DD}/V_{SS} for analog circuitry
is(phis)	Input pad, LVCMOS schmitt-trigger level
us(pbusb)	USB pad
ot(phot8)	Output pad, tri-state, Io = 8mA
b8(phob8)	Output pad, Io = 8mA
t16(phot16sm)	Output pad, tri-state, medium slew rate, Io = 16mA
r10(phiar10_abb)	Analog input pad with 10Ω resistor
ia(phia_abb)	Analog input pad
gp(phgpad_option)	Pad for analog pin
m26(phsoscm26)	Oscillator cell with enable and feedback resistor
t6(phtbsu100ct6sm)	Bi-directional pad, 5V tolerant LVCMOS schmitt-trigger, 100Kohm pull-up resistor with control, tri-state, Io = 6mA
t8(phbsu100ct8sm)	Bi-directional pad, LVCMOS schmitt-trigger, 100Kohm pull-up resistor with control, tri-state, Io = 8mA
t12(phbsu100ct12sm)	Bi-directional pad, LVCMOS schmitt-trigger, 100Kohm pull-up resistor with control, tri-state, Io = 12mA
d8(phbsu100cd8sm)	Bi-directional pad, LVCMOS schmitt-trigger, 100Kohm pull-up resistor with control, open-drain, Io = 8mA

SIGNAL DESCRIPTIONS

Table 1-3. S3C2410A Signal Descriptions

Signal	I/O	Descriptions
Bus Controller		
OM [1:0]	I	OM [1:0] sets S3C2410A in the TEST mode, which is used only at fabrication. Also, it determines the bus width of nGCS0. The pull-up/down resistor determines the logic level during the RESET cycle. 00:Nand-boot 01:16-bit 10:32-bit 11:Test mode
ADDR [26:0]	O	ADDR [26:0] (Address Bus) outputs the memory address of the corresponding bank.
DATA [31:0]	IO	DATA [31:0] (Data Bus) inputs data during memory read and outputs data during memory write. The bus width is programmable among 8/16/32-bit.
nGCS [7:0]	O	nGCS [7:0] (General Chip Select) are activated when the address of a memory is within the address region of each bank. The number of access cycles and the bank size can be programmed.
nWE	O	nWE (Write Enable) indicates that the current bus cycle is a write cycle.
nOE	O	nOE (Output Enable) indicates that the current bus cycle is a read cycle.
nXBREQ	I	nXBREQ (Bus Hold Request) allows another bus master to request control of the local bus. BACK active indicates that bus control has been granted.
nXBACK	O	nXBACK (Bus Hold Acknowledge) indicates that the S3C2410A has surrendered control of the local bus to another bus master.
nWAIT	I	nWAIT requests to prolong a current bus cycle. As long as nWAIT is L, the current bus cycle cannot be completed. If nWAIT signal isn't used in your system, nWAIT signal must be tied on pull-up resistor.
SDRAM/SRAM		
nSRAS	O	SDRAM Row Address Strobe
nSCAS	O	SDRAM Column Address Strobe
nSCS [1:0]	O	SDRAM Chip Select
DQM [3:0]	O	SDRAM Data Mask
SCLK [1:0]	O	SDRAM Clock
SCKE	O	SDRAM Clock Enable
nBE [3:0]	O	Upper Byte/Lower Byte Enable (In case of 16-bit SRAM)
nWBE [3:0]	O	Write Byte Enable

Table 1-3. S3C2410A Signal Descriptions (Continued)

Signal	I/O	Descriptions
NAND Flash		
CLE	O	Command Latch Enable
ALE	O	Address Latch Enable
nFCE	O	NAND Flash Chip Enable
nFRE	O	NAND Flash Read Enable
nFWE	O	NAND Flash Write Enable
NCON	I	NAND Flash Configuration. If NAND Flash Controller isn't used, it has to be tied on pull-up resistor.
R/nB	I	NAND Flash Ready/Busy. If NAND Flash Controller isn't used, it has to be tied on pull-up resistor.
LCD Control Unit		
VD [23:0]	O	**STN/TFT/SEC TFT:** LCD Data Bus
LCD_PWREN	O	**STN/TFT/SEC TFT:** LCD panel power enable control signal
VCLK	O	**STN/TFT:** LCD clock signal
VFRAME	O	**STN:** LCD Frame signal
VLINE	O	**STN:** LCD line signal
VM	O	**STN:** VM alternates the polarity of the row and column voltage
VSYNC	O	**TFT:** Vertical synchronous signal
HSYNC	O	**TFT:** Horizontal synchronous signal
VDEN	O	**TFT:** Data enable signal
LEND	O	**TFT:** Line End signal
STV	O	**SEC TFT:** SEC (Samsung Electronics Company) TFT LCD panel control signal
CPV	O	**SEC TFT:** SEC (Samsung Electronics Company) TFT LCD panel control signal
LCD_HCLK	O	**SEC TFT:** SEC (Samsung Electronics Company) TFT LCD panel control signal
TP	O	**SEC TFT:** SEC (Samsung Electronics Company) TFT LCD panel control signal
STH	O	**SEC TFT:** SEC (Samsung Electronics Company) TFT LCD panel control signal
LCDVF [2:0]	O	**SEC TFT:** Timing control signal for specific TFT LCD (OE/REV/REVB)
Interrupt Control Unit		
EINT [23:0]	I	External Interrupt request
DMA		
nXDREQ [1:0]	I	External DMA request
nXDACK [1:0]	O	External DMA acknowledge

Table 1-3. S3C2410A Signal Descriptions (Continued)

Signal	I/O	Descriptions
UART		
RxD [2:0]	I	UART receives data input
TxD [2:0]	O	UART transmits data output
nCTS [1:0]	I	UART clear to send input signal
nRTS [1:0]	O	UART request to send output signal
UEXTCLK	I	UART clock signal
ADC		
AIN [7:0]	AI	ADC input [7:0]. If it isn't used pin, it has to be in Ground.
Vref	AI	ADC Vref
IIC-Bus		
IICSDA	IO	IIC-bus data
IICSCL	IO	IIC-bus clock
IIS-Bus		
I2SLRCK	IO	IIS-bus channel select clock
I2SSDO	O	IIS-bus serial data output
I2SSDI	I	IIS-bus serial data input
I2SSCLK	IO	IIS-bus serial clock
CDCLK	O	CODEC system clock
Touch Screen		
nXPON	O	Plus X-axis on-off control signal
XMON	O	Minus X-axis on-off control signal
nYPON	O	Plus Y-axis on-off control signal
YMON	O	Minus Y-axis on-off control signal
USB Host		
DN [1:0]	IO	DATA (–) from USB host. (15Kohm pull-down)
DP [1:0]	IO	DATA (+) from USB host. (15Kohm pull-down)
USB Device		
PDN0	IO	DATA (–) for USB peripheral. (470Kohm pull-down)
PDP0	IO	DATA (+) for USB peripheral. (1.5Kohm pull-up)
SPI		
SPIMISO [1:0]	IO	SPIMISO is the master data input line, when SPI is configured as a master. When SPI is configured as a slave, these pins reverse its role.
SPIMOSI [1:0]	IO	SPIMOSI is the master data output line, when SPI is configured as a master. When SPI is configured as a slave, these pins reverse its role.
SPICLK [1:0]	IO	SPI clock
nSS [1:0]	I	SPI chip select (only for slave mode)

Table 1-3. S3C2410A Signal Descriptions (Continued)

Signal	I/O	Description
SD		
SDDAT [3:0]	IO	SD receive/transmit data
SDCMD	IO	SD receive response/ transmit command
SDCLK	O	SD clock
General Port		
GPn [116:0]	IO	General input/output ports (some ports are output only)
TIMMER/PWM		
TOUT [3:0]	O	Timer Output [3:0]
TCLK [1:0]	I	External timer clock input
JTAG TEST LOGIC		
nTRST	I	nTRST (TAP Controller Reset) resets the TAP controller at start. If debugger is used, A 10K pull-up resistor has to be connected. If debugger (black ICE) is not used, nTRST pin must be issued by a low active pulse (Typically connected to nRESET).
TMS	I	TMS (TAP Controller Mode Select) controls the sequence of the TAP controller's states. A 10K pull-up resistor has to be connected to TMS pin.
TCK	I	TCK (TAP Controller Clock) provides the clock input for the JTAG logic. A 10K pull-up resistor must be connected to TCK pin.
TDI	I	TDI (TAP Controller Data Input) is the serial input for test instructions and data. A 10K pull-up resistor must be connected to TDI pin.
TDO	O	TDO (TAP Controller Data Output) is the serial output for test instructions and data.
Reset, Clock & Power		
nRESET	ST	nRESET suspends any operation in progress and places S3C2410A into a known reset state. For a reset, nRESET must be held to L level for at least 4 FCLK after the processor power has been stabilized.
nRSTOUT	O	For external device reset control (nRSTOUT = nRESET & nWDTRST & SW_RESET)
PWREN	O	2.0V core power on-off control signal
nBATT_FLT	I	Probe for battery state (Does not wake up at power-off mode in case of low battery state). If it isn't used, it has to be High (3.3V).
OM [3:2]	I	OM [3:2] determines how the clock is made. OM [3:2] = 00b, Crystal is used for MPLL CLK source and UPLL CLK source. OM [3:2] = 01b, Crystal is used for MPLL CLK source and EXTCLK is used for UPLL CLK source. OM [3:2] = 10b, EXTCLK is used for MPLL CLK source and Crystal is used for UPLL CLK source. OM [3:2] = 11b, EXTCLK is used for MPLL CLK source and UPLL CLK source.

Table 1-3. S3C2410A Signal Descriptions (Continued)

Signal	I/O	Description
Reset, Clock & Power (Continued)		
EXTCLK	I	External clock source. When OM [3:2] = 11b, EXTCLK is used for MPLL CLK source and UPLL CLK source. When OM [3:2] = 10b, EXTCLK is used for MPLL CLK source only. When OM [3:2] = 01b, EXTCLK is used for UPLL CLK source only. If it isn't used, it has to be High (3.3V).
XTIpll	AI	Crystal Input for internal osc circuit. When OM [3:2] = 00b, XTIpll is used for MPLL CLK source and UPLL CLK source. When OM [3:2] = 01b, XTIpll is used for MPLL CLK source only. When OM [3:2] = 10b, XTIpll is used for UPLL CLK source only. If it isn't used, XTIpll has to be High (3.3V).
XTOpll	AO	Crystal Output for internal osc circuit. When OM [3:2] = 00b, XTIpll is used for MPLL CLK source and UPLL CLK source. When OM [3:2] = 01b, XTIpll is used for MPLL CLK source only. When OM [3:2] = 10b, XTIpll is used for UPLL CLK source only. If it isn't used, it has to be a floating pin.
MPLLCAP	AI	Loop filter capacitor for main clock.
UPLLCAP	AI	Loop filter capacitor for USB clock.
XTIrtc	AI	32.768 kHz crystal input for RTC. If it isn't used, it has to be in High (RTCVDD = 1.8V).
XTOrtc	AO	32.768 kHz crystal output for RTC. If it isn't used, it has to be Float.
CLKOUT [1:0]	O	Clock output signal. The CLKSEL of MISCCR register configures the clock output mode among the MPLL CLK, UPLL CLK, FCLK, HCLK and PCLK.

Table 1-3. S3C2410A Signal Descriptions (Continued)

Signal	I/O	Description
Power		
VDDalive	P	S3C2410A reset block and port status register V_{DD} (1.8V / 2.0V). It should be always supplied whether in normal mode or in power-off mode.
VDDi/VDDiarm	P	S3C2410A core logic V_{DD} (1.8V / 2.0V) for CPU.
VSSi/VSSiarm	P	S3C2410A core logic V_{SS}
VDDi_MPLL	P	S3C2410A MPLL analog and digital V_{DD} (1.8V / 2.0V).
VSSi_MPLL	P	S3C2410A MPLL analog and digital V_{SS}.
VDDOP	P	S3C2410A I/O port V_{DD} (3.3V)
VDDMOP	P	S3C2410A Memory I/O V_{DD} 3.3V: SCLK up to 133MHz
VSSMOP	P	S3C2410A Memory I/O V_{SS}
VSSOP	P	S3C2410A I/O port V_{SS}
RTCVDD	P	RTC V_{DD} (1.8 V, Not support 2.0 and 3.3V) (This pin must be connected to power properly if RTC isn't used)
VDDi_UPLL	P	S3C2410A UPLL analog and digital V_{DD} (1.8V / 2.0V)
VSSi_UPLL	P	S3C2410A UPLL analog and digital V_{SS}
VDDA_ADC	P	S3C2410A ADC V_{DD} (3.3V)
VSSA_ADC	P	S3C2410A ADC V_{SS}

NOTES:
1. I/O means input/output.
2. AI/AO means analog input/analog output.
3. ST means schmitt-trigger.
4. P means power.

S3C2410A SPECIAL REGISTERS

Table 1-4. S3C2410A Special Registers

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
Memory Controller					
BWSCON	0x48000000	←	W	R/W	Bus Width & Wait Status Control
BANKCON0	0x48000004				Boot ROM Control
BANKCON1	0x48000008				BANK1 Control
BANKCON2	0x4800000C				BANK2 Control
BANKCON3	0x48000010				BANK3 Control
BANKCON4	0x48000014				BANK4 Control
BANKCON5	0x48000018				BANK5 Control
BANKCON6	0x4800001C				BANK6 Control
BANKCON7	0x48000020				BANK7 Control
REFRESH	0x48000024				DRAM/SDRAM Refresh Control
BANKSIZE	0x48000028				Flexible Bank Size
MRSRB6	0x4800002C				Mode register set for SDRAM
MRSRB7	0x48000030				Mode register set for SDRAM

Table 1-4. S3C2410A Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
USB Host Controller					
HcRevision	0x49000000	←	W		Control and Status Group
HcControl	0x49000004				
HcCommonStatus	0x49000008				
HcInterruptStatus	0x4900000C				
HcInterruptEnable	0x49000010				
HcInterruptDisable	0x49000014				
HcHCCA	0x49000018				Memory Pointer Group
HcPeriodCuttentED	0x4900001C				
HcControlHeadED	0x49000020				
HcControlCurrentED	0x49000024				
HcBulkHeadED	0x49000028				
HcBulkCurrentED	0x4900002C				
HcDoneHead	0x49000030				
HcRmInterval	0x49000034				Frame Counter Group
HcFmRemaining	0x49000038				
HcFmNumber	0x4900003C				
HcPeriodicStart	0x49000040				
HcLSThreshold	0x49000044				
HcRhDescriptorA	0x49000048				Root Hub Group
HcRhDescriptorB	0x4900004C				
HcRhStatus	0x49000050				
HcRhPortStatus1	0x49000054				
HcRhPortStatus2	0x49000058				
Interrupt Controller					
SRCPND	0X4A000000	←	W	R/W	Interrupt Request Status
INTMOD	0X4A000004			W	Interrupt Mode Control
INTMSK	0X4A000008			R/W	Interrupt Mask Control
PRIORITY	0X4A00000C			W	IRQ Priority Control
INTPND	0X4A000010			R/W	Interrupt Request Status
INTOFFSET	0X4A000014			R	Interrupt request source offset
SUBSRCPND	0X4A000018			R/W	Sub source pending
INTSUBMSK	0X4A00001C			R/W	Interrupt sub mask

Table 1-4. S3C2410A Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
DMA					
DISRC0	0x4B000000	←	W	R/W	DMA 0 Initial Source
DISRCC0	0x4B000004				DMA 0 Initial Source Control
DIDST0	0x4B000008				DMA 0 Initial Destination
DIDSTC0	0x4B00000C				DMA 0 Initial Destination Control
DCON0	0x4B000010				DMA 0 Control
DSTAT0	0x4B000014			R	DMA 0 Count
DCSRC0	0x4B000018				DMA 0 Current Source
DCDST0	0x4B00001C				DMA 0 Current Destination
DMASKTRIG0	0x4B000020	←	W	R/W	DMA 0 Mask Trigger
DISRC1	0x4B000040				DMA 1 Initial Source
DISRCC1	0x4B000044				DMA 1 Initial Source Control
DIDST1	0x4B000048				DMA 1 Initial Destination
DIDSTC1	0x4B00004C				DMA 1 Initial Destination Control
DCON1	0x4B000050				DMA 1 Control
DSTAT1	0x4B000054			R	DMA 1 Count
DCSRC1	0x4B000058				DMA 1 Current Source
DCDST1	0x4B00005C	←	W		DMA 1 Current Destination
DMASKTRIG1	0x4B000060			R/W	DMA 1 Mask Trigger
DISRC2	0x4B000080				DMA 2 Initial Source
DISRCC2	0x4B000084				DMA 2 Initial Source Control
DIDST2	0x4B000088				DMA 2 Initial Destination
DIDSTC2	0x4B00008C				DMA 2 Initial Destination Control
DCON2	0x4B000090				DMA 2 Control
DSTAT2	0x4B000094			R	DMA 2 Count
DCSRC2	0x4B000098	←	W		DMA 2 Current Source
DCDST2	0x4B00009C				DMA 2 Current Destination
DMASKTRIG2	0x4B0000A0			R/W	DMA 2 Mask Trigger
DISRC3	0x4B0000C0	←	W	R/W	DMA 3 Initial Source
DISRCC3	0x4B0000C4				DMA 3 Initial Source Control
DIDST3	0x4B0000C8				DMA 3 Initial Destination
DIDSTC3	0x4B0000CC				DMA 3 Initial Destination Control
DCON3	0x4B0000D0				DMA 3 Control
DSTAT3	0x4B0000D4			R	DMA 3 Count
DCSRC3	0x4B0000D8				DMA 3 Current Source
DCDST3	0x4B0000DC				DMA 3 Current Destination
DMASKTRIG3	0x4B0000E0			R/W	DMA 3 Mask Trigger

Table 1-4. S3C2410A Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
Clock & Power Management					
LOCKTIME	0x4C000000	←	W	R/W	PLL Lock Time Counter
MPLLCON	0x4C000004				MPLL Control
UPLLCON	0x4C000008				UPLL Control
CLKCON	0x4C00000C				Clock Generator Control
CLKSLOW	0x4C000010				Slow Clock Control
CLKDIVN	0x4C000014				Clock divider Control
LCD Controller					
LCDCON1	0X4D000000	←	W	R/W	LCD Control 1
LCDCON2	0X4D000004				LCD Control 2
LCDCON3	0X4D000008				LCD Control 3
LCDCON4	0X4D00000C				LCD Control 4
LCDCON5	0X4D000010				LCD Control 5
LCDSADDR1	0X4D000014				STN/TFT: Frame Buffer Start Address1
LCDSADDR2	0X4D000018				STN/TFT: Frame Buffer Start Address2
LCDSADDR3	0X4D00001C				STN/TFT: Virtual Screen Address Set
REDLUT	0X4D000020				STN: Red Lookup Table
GREENLUT	0X4D000024				STN: Green Lookup Table
BLUELUT	0X4D000028				STN: Blue Lookup Table
DITHMODE	0X4D00004C				STN: Dithering Mode
TPAL	0X4D000050				TFT: Temporary Palette
LCDINTPND	0X4D000054				LCD Interrupt Pending
LCDSRCPND	0X4D000058				LCD Interrupt Source
LCDINTMSK	0X4D00005C				LCD Interrupt Mask
LPCSEL	0X4D000060				LPC3600 Control
NAND Flash					
NFCONF	0x4E000000	←	W	R/W	NAND Flash Configuration
NFCMD	0x4E000004				NAND Flash Command
NFADDR	0x4E000008				NAND Flash Address
NFDATA	0x4E00000C				NAND Flash Data
NFSTAT	0x4E000010			R	NAND Flash Operation Status
NFECC	0x4E000014			R/W	NAND Flash ECC

Table 1-4. S3C2410A Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
UART					
ULCON0	0x50000000	←	W	R/W	UART 0 Line Control
UCON0	0x50000004				UART 0 Control
UFCON0	0x50000008				UART 0 FIFO Control
UMCON0	0x5000000C				UART 0 Modem Control
UTRSTAT0	0x50000010			R	UART 0 Tx/Rx Status
UERSTAT0	0x50000014				UART 0 Rx Error Status
UFSTAT0	0x50000018				UART 0 FIFO Status
UMSTAT0	0x5000001C				UART 0 Modem Status
UTXH0	0x50000023	0x50000020	B	W	UART 0 Transmission Hold
URXH0	0x50000027	0x50000024		R	UART 0 Receive Buffer
UBRDIV0	0x50000028	←	W	R/W	UART 0 Baud Rate Divisor
ULCON1	0x50004000	←	W	R/W	UART 1 Line Control
UCON1	0x50004004				UART 1 Control
UFCON1	0x50004008				UART 1 FIFO Control
UMCON1	0x5000400C				UART 1 Modem Control
UTRSTAT1	0x50004010			R	UART 1 Tx/Rx Status
UERSTAT1	0x50004014				UART 1 Rx Error Status
UFSTAT1	0x50004018				UART 1 FIFO Status
UMSTAT1	0x5000401C				UART 1 Modem Status
UTXH1	0x50004023	0x50004020	B	W	UART 1 Transmission Hold
URXH1	0x50004027	0x50004024		R	UART 1 Receive Buffer
UBRDIV1	0x50004028	←	W	R/W	UART 1 Baud Rate Divisor
ULCON2	0x50008000	←	W	R/W	UART 2 Line Control
UCON2	0x50008004				UART 2 Control
UFCON2	0x50008008				UART 2 FIFO Control
UTRSTAT2	0x50008010			R	UART 2 Tx/Rx Status
UERSTAT2	0x50008014				UART 2 Rx Error Status
UFSTAT2	0x50008018				UART 2 FIFO Status
UTXH2	0x50008023	0x50008020	B	W	UART 2 Transmission Hold
URXH2	0x50008027	0x50008024		R	UART 2 Receive Buffer
UBRDIV2	0x50008028	←	W	R/W	UART 2 Baud Rate Divisor

Table 1-4. S3C2410A Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
PWM Timer					
TCFG0	0x51000000	←	W	R/W	Timer Configuration
TCFG1	0x51000004				Timer Configuration
TCON	0x51000008				Timer Control
TCNTB0	0x5100000C				Timer Count Buffer 0
TCMPB0	0x51000010				Timer Compare Buffer 0
TCNTO0	0x51000014			R	Timer Count Observation 0
TCNTB1	0x51000018			R/W	Timer Count Buffer 1
TCMPB1	0x5100001C				Timer Compare Buffer 1
TCNTO1	0x51000020			R	Timer Count Observation 1
TCNTB2	0x51000024			R/W	Timer Count Buffer 2
TCMPB2	0x51000028				Timer Compare Buffer 2
TCNTO2	0x5100002C			R	Timer Count Observation 2
TCNTB3	0x51000030			R/W	Timer Count Buffer 3
TCMPB3	0x51000034				Timer Compare Buffer 3
TCNTO3	0x51000038			R	Timer Count Observation 3
TCNTB4	0x5100003C			R/W	Timer Count Buffer 4
TCNTO4	0x51000040			R	Timer Count Observation 4

Table 1-4. S3C2410A Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
USB Device					
FUNC_ADDR_REG	0x52000143	0x52000140	B	R/W	Function Address
PWR_REG	0x52000147	0x52000144			Power Management
EP_INT_REG	0x5200014B	0x52000148			EP Interrupt Pending and Clear
USB_INT_REG	0x5200015B	0x52000158			USB Interrupt Pending and Clear
EP_INT_EN_REG	0x5200015F	0x5200015C			Interrupt Enable
USB_INT_EN_REG	0x5200016F	0x5200016C			Interrupt Enable
FRAME_NUM1_REG	0x52000173	0x52000170		R	Frame Number Lower Byte
INDEX_REG	0x5200017B	0x52000178		R/W	Register Index
EP0_CSR	0x52000187	0x52000184			Endpoint 0 Status
IN_CSR1_REG	0x52000187	0x52000184			In Endpoint Control Status
IN_CSR2_REG	0x5200018B	0x52000188			In Endpoint Control Status
MAXP_REG	0x52000183	0x52000180			Endpoint Max Packet
OUT_CSR1_REG	0x52000193	0x52000190			Out Endpoint Control Status
OUT_CSR2_REG	0x52000197	0x52000194			Out Endpoint Control Status
OUT_FIFO_CNT1_REG	0x5200019B	0x52000198		R	Endpoint Out Write Count
OUT_FIFO_CNT2_REG	0x5200019F	0x5200019C			Endpoint Out Write Count
EP0_FIFO	0x520001C3	0x520001C0		R/W	Endpoint 0 FIFO
EP1_FIFO	0x520001C7	0x520001C4			Endpoint 1 FIFO
EP2_FIFO	0x520001CB	0x520001C8			Endpoint 2 FIFO
EP3_FIFO	0x520001CF	0x520001CC			Endpoint 3 FIFO
EP4_FIFO	0x520001D3	0x520001D0			Endpoint 4 FIFO
EP1_DMA_CON	0x52000203	0x52000200			EP1 DMA Interface Control
EP1_DMA_UNIT	0x52000207	0x52000204			EP1 DMA Tx Unit Counter
EP1_DMA_FIFO	0x5200020B	0x52000208			EP1 DMA Tx FIFO Counter
EP1_DMA_TTC_L	0x5200020F	0x5200020C			EP1 DMA Total Tx Counter
EP1_DMA_TTC_M	0x52000213	0x52000210			EP1 DMA Total Tx Counter
EP1_DMA_TTC_H	0x52000217	0x52000214			EP1 DMA Total Tx Counter

Table 1-4. S3C2410A Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/Write	Function
USB Device (Continued)					
EP2_DMA_CON	0x5200021B	0x52000218	B	R/W	EP2 DMA Interface Control
EP2_DMA_UNIT	0x5200021F	0x5200021C			EP2 DMA Tx Unit Counter
EP2_DMA_FIFO	0x52000223	0x52000220			EP2 DMA Tx FIFO Counter
EP2_DMA_TTC_L	0x52000227	0x52000224			EP2 DMA Total Tx Counter
EP2_DMA_TTC_M	0x5200022B	0x52000228			EP2 DMA Total Tx Counter
EP2_DMA_TTC_H	0x5200022F	0x5200022C			EP2 DMA Total Tx Counter
EP3_DMA_CON	0x52000243	0x52000240			EP3 DMA Interface Control
EP3_DMA_UNIT	0x52000247	0x52000244			EP3 DMA Tx Unit Counter
EP3_DMA_FIFO	0x5200024B	0x52000248			EP3 DMA Tx FIFO Counter
EP3_DMA_TTC_L	0x5200024F	0x5200024C			EP3 DMA Total Tx Counter
EP3_DMA_TTC_M	0x52000253	0x52000250			EP3 DMA Total Tx Counter
EP3_DMA_TTC_H	0x52000257	0x52000254			EP3 DMA Total Tx Counter
EP4_DMA_CON	0x5200025B	0x52000258			EP4 DMA Interface Control
EP4_DMA_UNIT	0x5200025F	0x5200025C			EP4 DMA Tx Unit Counter
EP4_DMA_FIFO	0x52000263	0x52000260			EP4 DMA Tx FIFO Counter
EP4_DMA_TTC_L	0x52000267	0x52000264			EP4 DMA Total Tx Counter
EP4_DMA_TTC_M	0x5200026B	0x52000268			EP4 DMA Total Tx Counter
EP4_DMA_TTC_H	0x5200026F	0x5200026C			EP4 DMA Total Tx Counter
Watchdog Timer					
WTCON	0x53000000	←	W	R/W	Watchdog Timer Mode
WTDAT	0x53000004				Watchdog Timer Data
WTCNT	0x53000008				Watchdog Timer Count
IIC					
IICCON	0x54000000	←	W	R/W	IIC Control
IICSTAT	0x54000004				IIC Status
IICADD	0x54000008				IIC Address
IICDS	0x5400000C				IIC Data Shift
IIS					
IISCON	0x55000000,02	0x55000000	HW,W	R/W	IIS Control
IISMOD	0x55000004,06	0x55000004	HW,W		IIS Mode
IISPSR	0x55000008,0A	0x55000008	HW,W		IIS Prescaler
IISFCON	0x5500000C,0E	0x5500000C	HW,W		IIS FIFO Control
IISFIFO	0x55000012	0x55000010	HW		IIS FIFO Entry

Table 1-4. S3C2410A Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
I/O port					
GPACON	0x56000000	←	W	R/W	Port A Control
GPADAT	0x56000004				Port A Data
GPBCON	0x56000010				Port B Control
GPBDAT	0x56000014				Port B Data
GPBUP	0x56000018				Pull-up Control B
GPCCON	0x56000020				Port C Control
GPCDAT	0x56000024				Port C Data
GPCUP	0x56000028				Pull-up Control C
GPDCON	0x56000030				Port D Control
GPDDA1T	0x56000034				Port D Data
GPDUP	0x56000038				Pull-up Control D
GPECON	0x56000040				Port E Control
GPEDAT	0x56000044				Port E Data
GPEUP	0x56000048				Pull-up Control E
GPFCON	0x56000050				Port F Control
GPFDAT	0x56000054				Port F Data
GPFUP	0x56000058				Pull-up Control F
GPGCON	0x56000060				Port G Control
GPGDAT	0x56000064				Port G Data
GPGUP	0x56000068				Pull-up Control G
GPHCON	0x56000070				Port H Control
GPHDAT	0x56000074				Port H Data
GPHUP	0x56000078				Pull-up Control H
MISCCR	0x56000080	←	W	R/W	Miscellaneous Control
DCLKCON	0x56000084				DCLK0/1 Control
EXTINT0	0x56000088				External Interrupt Control Register 0
EXTINT1	0x5600008C				External Interrupt Control Register 1
EXTINT2	0x56000090				External Interrupt Control Register 2
EINTFLT0	0x56000094				Reserved
EINTFLT1	0x56000098				Reserved
EINTFLT2	0x5600009C				External Interrupt Filter Control Register 2
EINTFLT3	0x560000A0				External Interrupt Filter Control Register 3
EINTMASK	0x560000A4				External Interrupt Mask
EINTPEND	0x560000A8				External Interrupt Pending
GSTATUS0	0x560000AC				External Pin Status
GSTATUS1	0x560000B0				External Pin Status

Table 1-4. S3C2410A Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
RTC					
RTCCON	0x57000043	0x57000040	B	R/W	RTC Control
TICNT	0x57000047	0x57000044			Tick time count
RTCALM	0x57000053	0x57000050			RTC Alarm Control
ALMSEC	0x57000057	0x57000054			Alarm Second
ALMMIN	0x5700005B	0x57000058			Alarm Minute
ALMHOUR	0x5700005F	0x5700005C			Alarm Hour
ALMDATE	0x57000063	0x57000060			Alarm Day
ALMMON	0x57000067	0x57000064			Alarm Month
ALMYEAR	0x5700006B	0x57000068			Alarm Year
RTCRST	0x5700006F	0x5700006C			RTC Round Reset
BCDSEC	0x57000073	0x57000070			BCD Second
BCDMIN	0x57000077	0x57000074			BCD Minute
BCDHOUR	0x5700007B	0x57000078			BCD Hour
BCDDATE	0x5700007F	0x5700007C			BCD Day
BCDDAY	0x57000083	0x57000080			BCD Date
BCDMON	0x57000087	0x57000084			BCD Month
BCDYEAR	0x5700008B	0x57000088			BCD Year
A/D converter					
ADCCON	0x58000000	←	W	R/W	ADC Control
ADCTSC	0x58000004				ADC Touch Screen Control
ADCDLY	0x58000008				ADC Start or Interval Delay
ADCDAT0	0x5800000C			R	ADC Conversion Data
ADCDAT1	0x58000010				ADC Conversion Data
SPI					
SPCON0,1	0x59000000,20	←	W	R/W	SPI Control
SPSTA0,1	0x59000004,24			R	SPI Status
SPPIN0,1	0x59000008,28			R/W	SPI Pin Control
SPPRE0,1	0x5900000C,2C				SPI Baud Rate Prescaler
SPTDAT0,1	0x59000010,30				SPI Tx Data
SPRDAT0,1	0x59000014,34			R	SPI Rx Data

Table 1-4. S3C2410A Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
SD interface					
SDICON	0x5A000000	←	W	R/W	SDI Control
SDIPRE	0x5A000004				SDI Baud Rate Prescaler
SDICmdArg	0x5A000008				SDI Command Argument
SDICmdCon	0x5A00000C				SDI Command Control
SDICmdSta	0x5A000010			R/(C)	SDI Command Status
SDIRSP0	0x5A000014			R	SDI Response
SDIRSP1	0x5A000018				SDI Response
SDIRSP2	0x5A00001C				SDI Response
SDIRSP3	0x5A000020				SDI Response
SDIDTimer	0x5A000024			R/W	SDI Data / Busy Timer
SDIBSize	0x5A000028				SDI Block Size
SDIDatCon	0x5A00002C				SDI Data control
SDIDatCnt	0x5A000030			R	SDI Data Remain Counter
SDIDatSta	0x5A000034			R/(C)	SDI Data Status
SDIFSTA	0x5A000038			R	SDI FIFO Status
SDIDAT	0x5A00003F	0x5A00003C	B	R/W	SDI Data
SDIIntMsk	0x5A000040	←	W		SDI Interrupt Mask

Cautions on S3C2410A Special Registers

1. In the little endian mode, L. endian address must be used. In the big endian mode, B. endian address must be used.
2. The special registers have to be accessed for each recommended access unit.
3. All registers except ADC registers, RTC registers and UART registers must be read/written in word unit (32-bit) at little/big endian.
4. Make sure that the ADC registers, RTC registers and UART registers be read/written by the specified access unit and the specified address. Moreover, one must carefully consider which endian mode is used.
5. W : 32-bit register, which must be accessed by LDR/STR or int type pointer(int *).
   HW : 16-bit register, which must be accessed by LDRH/STRH or short int type pointer(short int *).
   B : 8-bit register, which must be accessed by LDRB/STRB or char type pointer(char int *).

2 PROGRAMMER'S MODEL

OVERVIEW

S3C2410A has been developed using the advanced ARM920T core, which has been designed by Advanced RISC Machines, Ltd.

PROCESSOR OPERATING STATES

From the programmer's point of view, the ARM920T can be in one of two states:

- ARM state which executes 32-bit, word-aligned ARM instructions.
- THUMB state which can execute 16-bit, halfword-aligned THUMB instructions. In this state, the PC uses bit 1 to select between alternate halfwords.

NOTE

Transition between these two states does not affect the processor mode or the contents of the registers.

SWITCHING STATE

Entering THUMB State

Entry into THUMB state can be achieved by executing a BX instruction with the state bit (bit 0) set in the operand register.

Transition to THUMB state will also occur automatically on return from an exception (IRQ, FIQ, UNDEF, ABORT, SWI etc.), if the exception was entered with the processor in THUMB state.

Entering ARM State

Entry into ARM state happens:

- On execution of the BX instruction with the state bit clear in the operand register.
- On the processor taking an exception (IRQ, FIQ, RESET, UNDEF, ABORT, SWI etc.). In this case, the PC is placed in the exception mode's link register, and execution commences at the exception's vector address.

MEMORY FORMATS

ARM920T views memory as a linear collection of bytes numbered upwards from zero. Bytes 0 to 3 hold the first stored word, bytes 4 to 7 the second and so on. ARM920T can treat words in memory as being stored either in Big-Endian or Little-Endian format.

BIG-ENDIAN FORMAT

In Big-Endian format, the most significant byte of a word is stored at the lowest numbered byte and the least significant byte at the highest numbered byte. Byte 0 of the memory system is therefore connected to data lines 31 through 24.

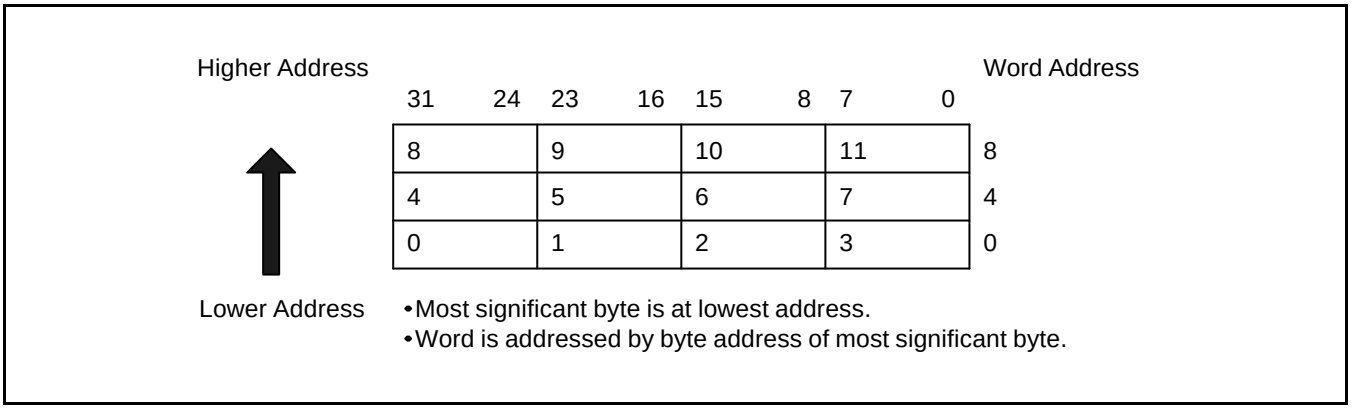


Figure 2-1. Big-Endian Addresses of Bytes within Words

LITTLE-ENDIAN FORMAT

In Little-Endian format, the lowest numbered byte in a word is considered the word's least significant byte, and the highest numbered byte the most significant. Byte 0 of the memory system is therefore connected to data lines 7 through 0.

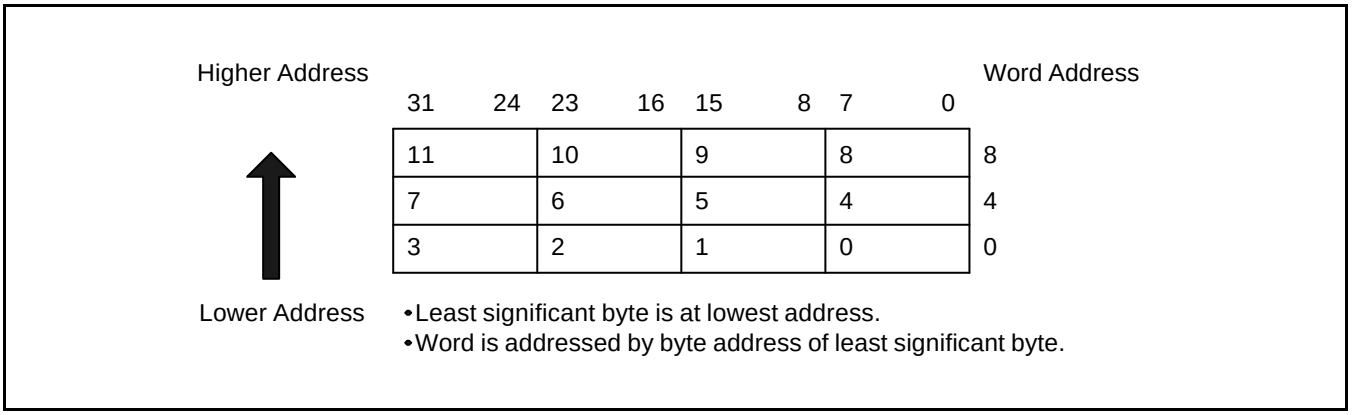


Figure 2-2. Little-Endian Addresses of Bytes within Words

INSTRUCTION LENGTH

Instructions are either 32 bits long (in ARM state) or 16 bits long (in THUMB state).

Data Types

ARM920T supports byte (8-bit), halfword (16-bit) and word (32-bit) data types. Words must be aligned to four-byte boundaries and half words to two-byte boundaries.

OPERATING MODES

ARM920T supports seven modes of operation:

- User (usr): The normal ARM program execution state
- FIQ (fiq): Designed to support a data transfer or channel process
- IRQ (irq): Used for general-purpose interrupt handling
- Supervisor (svc): Protected mode for the operating system
- Abort mode (abt): Entered after a data or instruction prefetch abort
- System (sys): A privileged user mode for the operating system
- Undefined (und): Entered when an undefined instruction is executed

Mode changes may be made under software control, or may be brought about by external interrupts or exception processing. Most application programs will execute in User mode. The non-user modes' known as privileged modes- are entered in order to service interrupts or exceptions, or to access protected resources.

REGISTERS

ARM920T has a total of 37 registers - 31 general-purpose 32-bit registers and six status registers - but these cannot all be seen at once. The processor state and operating mode dictate which registers are available to the programmer.

The ARM State Register Set

In ARM state, 16 general registers and one or two status registers are visible at any one time. In privileged (non-User) modes, mode-specific banked registers are switched in. Figure 2-3 shows which registers are available in each mode: the banked registers are marked with a shaded triangle.

The ARM state register set contains 16 directly accessible registers: R0 to R15. All of these except R15 are general-purpose, and may be used to hold either data or address values. In addition to these, there is a seventeenth register used to store status information.

Register 14	is used as the subroutine link register. This receives a copy of R15 when a Branch and Link (BL) instruction is executed. At all other times it may be treated as a general-purpose register. The corresponding banked registers R14_svc, R14_irq, R14_fiq, R14_abt and R14_und are similarly used to hold the return values of R15 when interrupts and exceptions arise, or when Branch and Link instructions are executed within interrupt or exception routines.
Register 15	holds the Program Counter (PC). In ARM state, bits [1:0] of R15 are zero and bits [31:2] contain the PC. In THUMB state, bit [0] is zero and bits [31:1] contain the PC.
Register 16	is the CPSR (Current Program Status Register). This contains condition code flags and the current mode bits.

FIQ mode has seven banked registers mapped to R8-14 (R8_fiq-R14_fiq). In ARM state, many FIQ handlers do not need to save any registers. User, IRQ, Supervisor, Abort and Undefined each have two banked registers mapped to R13 and R14, allowing each of these modes to have a private stack pointer and link registers.

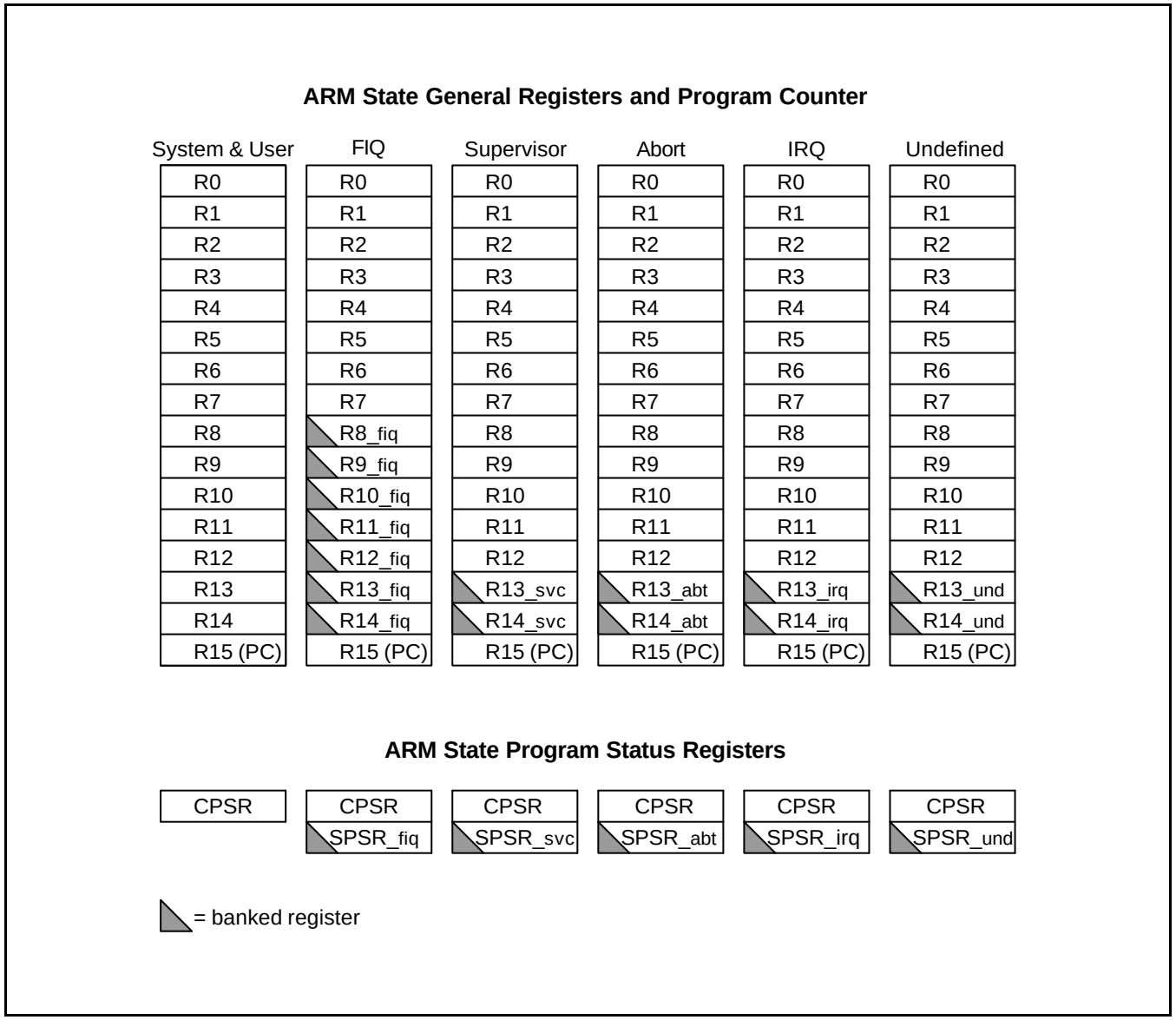


Figure 2-3. Register Organization in ARM State

The THUMB State Register Set

The THUMB state register set is a subset of the ARM state set. The programmer has direct access to eight general registers, R0-R7, as well as the Program Counter (PC), a stack pointer register (SP), a link register (LR), and the CPSR. There are banked Stack Pointers, Link Registers and Saved Process Status Registers (SPSRs) for each privileged mode. This is shown in Figure 2-4.

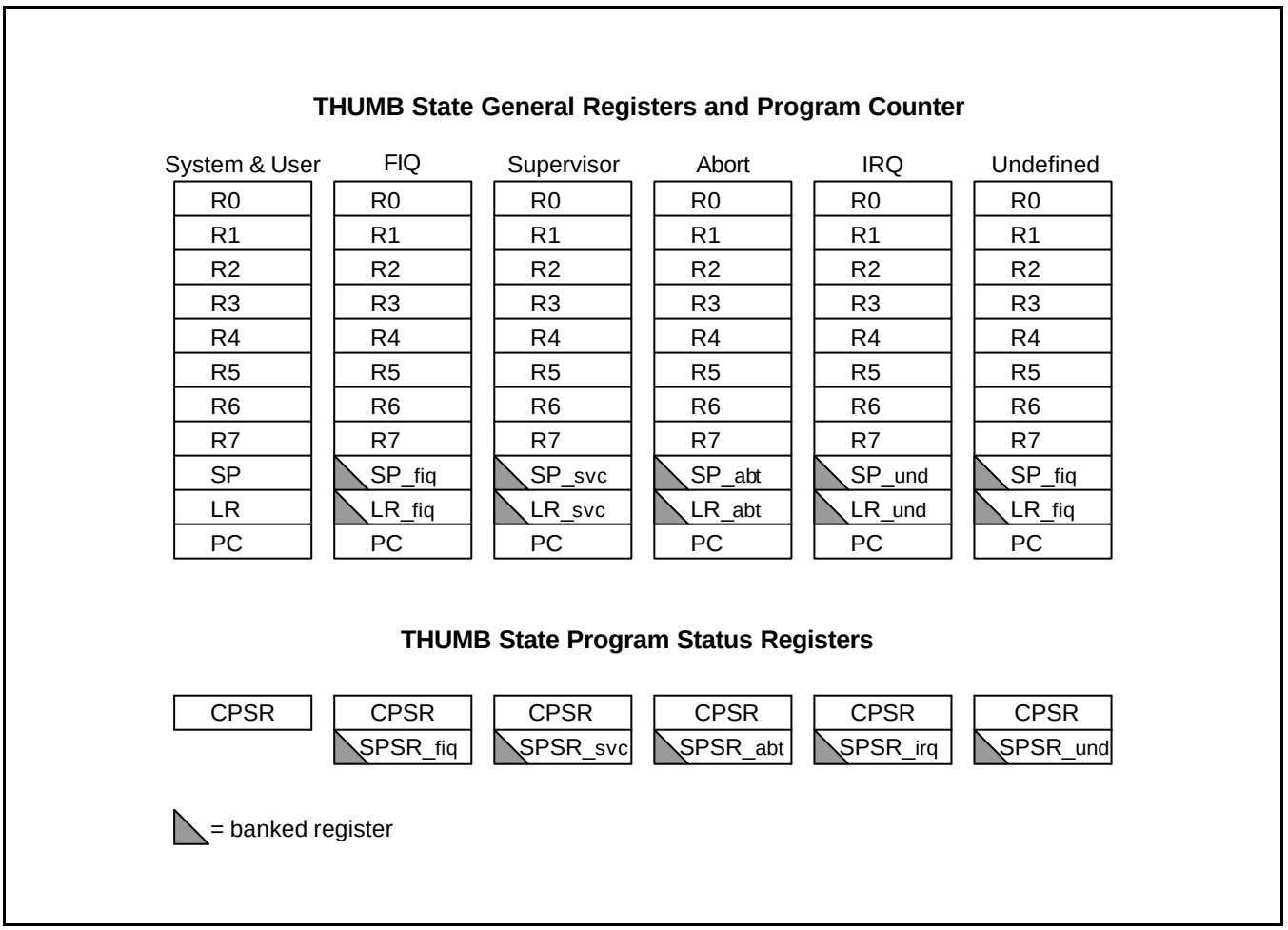


Figure 2-4. Register Organization in THUMB state

The relationship between ARM and THUMB state registers

The THUMB state registers relate to the ARM state registers in the following way:

- THUMB state R0-R7 and ARM state R0-R7 are identical
- THUMB state CPSR and SPSRs and ARM state CPSR and SPSRs are identical
- THUMB state SP maps onto ARM state R13
- THUMB state LR maps onto ARM state R14
- The THUMB state Program Counter maps onto the ARM state Program Counter (R15)

This relationship is shown in Figure 2-5.

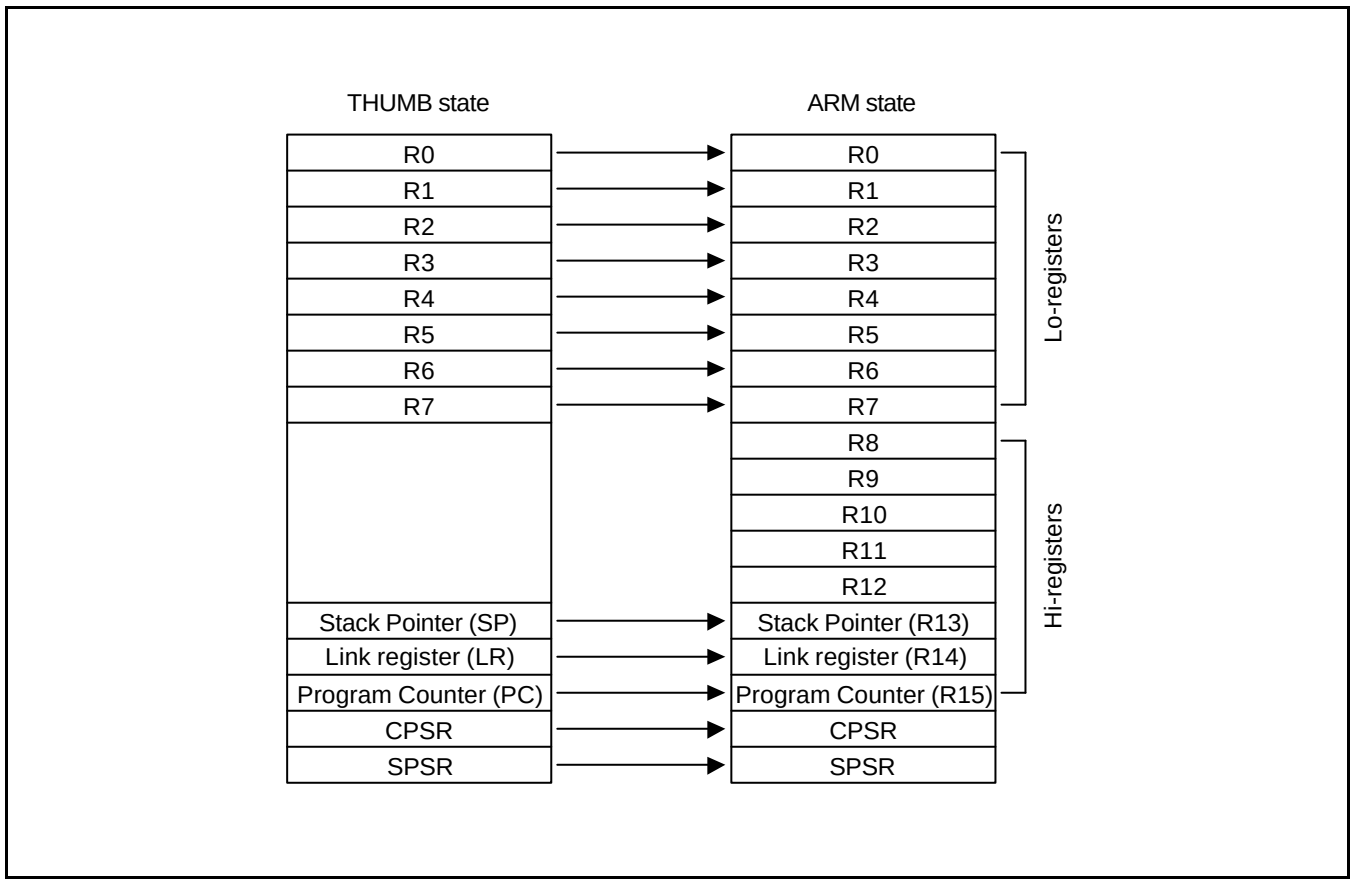


Figure 2-5. Mapping of THUMB State Registers onto ARM State Registers

Accessing Hi-Registers in THUMB State

In THUMB state, registers R8-R15 (the Hi registers) are not part of the standard register set. However, the assembly language programmer has limited access to them, and can use them for fast temporary storage.

A value may be transferred from a register in the range R0-R7 (a Lo register) to a Hi register, and from a Hi register to a Lo register, using special variants of the MOV instruction. Hi register values can also be compared against or added to Lo register values with the CMP and ADD instructions. For more information, refer to Figure 3-34.

THE PROGRAM STATUS REGISTERS

The ARM920T contains a Current Program Status Register (CPSR), plus five Saved Program Status Registers (SPSRs) for use by exception handlers. These register's functions are:

- Hold information about the most recently performed ALU operation
- Control the enabling and disabling of interrupts
- Set the processor operating mode

The arrangement of bits is shown in Figure 2-6.

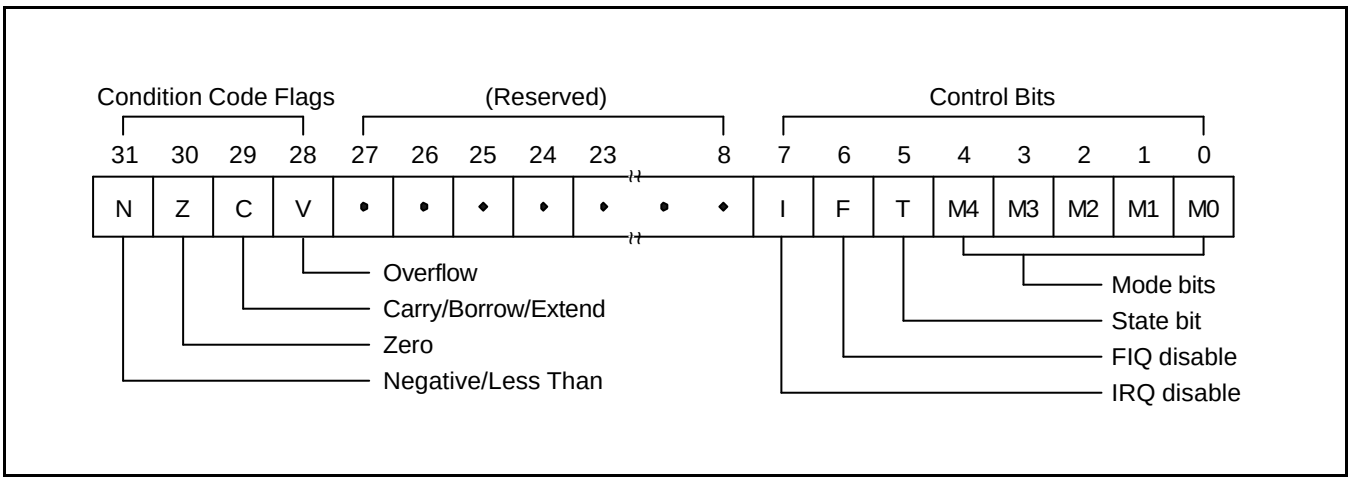


Figure 2-6. Program Status Register Formats

The Condition Code Flags

The N, Z, C and V bits are the condition code flags. These may be changed as a result of arithmetic and logical operations, and may be tested to determine whether an instruction should be executed.

In ARM state, all instructions may be executed conditionally: see Table 3-2 for details.

In THUMB state, only the Branch instruction is capable of conditional execution: see Figure 3-46 for details.

The Control Bits

The bottom 8 bits of a PSR (incorporating I, F, T and M[4:0]) are known collectively as the control bits. These will be changed when an exception arises. If the processor is operating in a privileged mode, they can also be manipulated by software.

The T bit	This reflects the operating state. When this bit is set, the processor is executing in THUMB state, otherwise it is executing in ARM state. This is reflected on the **TBIT** external signal. Note that the software must never change the state of the **TBIT** in the CPSR. If this happens, the processor will enter an unpredictable state.
Interrupt disable bits	The I and F bits are the interrupt disable bits. When set, these disable the IRQ and FIQ interrupts respectively.
The mode bits	The M4, M3, M2, M1 and M0 bits (M[4:0]) are the mode bits. These determine the processor's operating mode, as shown in Table 2-1. Not all combinations of the mode bits define a valid processor mode. Only those explicitly described shall be used. The user should be aware that if any illegal value is programmed into the mode bits, M[4:0], then the processor will enter an unrecoverable state. If this occurs, reset should be applied.
Reserved bits	The remaining bits in the PSRs are reserved. When changing a PSR's flag or control bits, you must ensure that these unused bits are not altered. Also, your program should not rely on them containing specific values, since in future processors they may read as one or zero.

Table 2-1. PSR Mode Bit Values

M[4:0]	Mode	Visible THUMB state registers	Visible ARM state registers
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
10111	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

Reserved bits — The remaining bits in the PSR's are reserved. When changing a PSR's flag or control bits, you must ensure that these unused bits are not altered. Also, your program should not rely on them containing specific values, since in future processors they may read as one or zero.

EXCEPTIONS

Exceptions arise whenever the normal flow of a program has to be halted temporarily, for example to service an interrupt from a peripheral. Before an exception can be handled, the current processor state must be preserved so that the original program can resume when the handler routine has finished.

It is possible for several exceptions to arise at the same time. If this happens, they are dealt with in a fixed order. See Exception Priorities on page 2-14.

Action on Entering an Exception

When handling an exception, the ARM920T:

1. Preserves the address of the next instruction in the appropriate Link Register. If the exception has been entered from ARM state, then the address of the next instruction is copied into the Link Register (that is, current PC + 4 or PC + 8 depending on the exception. See Table 2-2 on for details). If the exception has been entered from THUMB state, then the value written into the Link Register is the current PC offset by a value such that the program resumes from the correct place on return from the exception. This means that the exception handler need not determine which state the exception was entered from. For example, in the case of SWI, MOVS PC, R14_svc will always return to the next instruction regardless of whether the SWI was executed in ARM or THUMB state.
2. Copies the CPSR into the appropriate SPSR
3. Forces the CPSR mode bits to a value which depends on the exception
4. Forces the PC to fetch the next instruction from the relevant exception vector

It may also set the interrupt disable flags to prevent otherwise unmanageable nestings of exceptions.

If the processor is in THUMB state when an exception occurs, it will automatically switch into ARM state when the PC is loaded with the exception vector address.

Action on Leaving an Exception

On completion, the exception handler:

1. Moves the Link Register, minus an offset where appropriate, to the PC. (The offset will vary depending on the type of exception.)
2. Copies the SPSR back to the CPSR
3. Clears the interrupt disable flags, if they were set on entry

NOTE

An explicit switch back to THUMB state is never needed, since restoring the CPSR from the SPSR automatically sets the T bit to the value it held immediately prior to the exception.

Exception Entry/Exit Summary

Table 2-2 summarizes the PC value preserved in the relevant R14 on exception entry, and the recommended instruction for exiting the exception handler.

Table 2-2. Exception Entry/Exit

	Return Instruction	Previous State		Notes
		ARM R14_x	THUMB 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. Where PC is the address of the BL/SWI/Undefined Instruction fetch which had the prefetch abort.
2. Where PC is the address of the instruction which did not get executed since the FIQ or IRQ took priority.
3. Where PC is the address of the Load or Store instruction which generated the data abort.
4. The value saved in R14_svc upon reset is unpredictable.

FIQ

The FIQ (Fast Interrupt Request) exception is designed to support a data transfer or channel process, and in ARM state has sufficient private registers to remove the need for register saving (thus minimizing the overhead of context switching).

FIQ is externally generated by taking the nFIQ input LOW. This input can except either synchronous or asynchronous transitions, depending on the state of the ISYNC input signal. When ISYNC is LOW, nFIQ and nIRQ are considered asynchronous, and a cycle delay for synchronization is incurred before the interrupt can affect the processor flow.

Irrespective of whether the exception was entered from ARM or Thumb state, a FIQ handler should leave the interrupt by executing

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

FIQ may be disabled by setting the CPSR's F flag (but note that this is not possible from User mode). If the F flag is clear, ARM920T checks for a LOW level on the output of the FIQ synchronizer at the end of each instruction.

IRQ

The IRQ (Interrupt Request) exception is a normal interrupt caused by a LOW level on the nIRQ input. IRQ has a lower priority than FIQ and is masked out when a FIQ sequence is entered. It may be disabled at any time by setting the I bit in the CPSR, though this can only be done from a privileged (non-User) mode.

Irrespective of whether the exception was entered from ARM or Thumb state, an IRQ handler should return from the interrupt by executing

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

Abort

An abort indicates that the current memory access cannot be completed. It can be signaled by the external ABORT input. ARM920T checks for the abort exception during memory access cycles.

There are two types of abort:

- *Prefetch abort:* occurs during an instruction prefetch.
- *Data abort:* occurs during a data access.

If a prefetch abort occurs, the prefetched instruction is marked as invalid, but the exception will not be taken until the instruction reaches the head of the pipeline. If the instruction is not executed - for example because a branch occurs while it is in the pipeline - the abort does not take place.

If a data abort occurs, the action taken depends on the instruction type:

- Single data transfer instructions (LDR, STR) write back modified base registers: the Abort handler must be aware of this.
- The swap instruction (SWP) is aborted as though it had not been executed.
- Block data transfer instructions (LDM, STM) complete. If write-back is set, the base is updated. If the instruction would have overwritten the base with data (ie it has the base in the transfer list), the overwriting is prevented. All register overwriting is prevented after an abort is indicated, which means in particular that R15 (always the last register to be transferred) is preserved in an aborted LDM instruction.

The abort mechanism allows the implementation of a demand paged virtual memory system. In such a system the processor is allowed to generate arbitrary addresses. When the data at an address is unavailable, the Memory Management Unit (MMU) signals an abort. The abort handler must then work out the cause of the abort, make the requested data available, and retry the aborted instruction. The application program needs no knowledge of the amount of memory available to it, nor is its state in any way affected by the abort.

After fixing the reason for the abort, the handler should execute the following irrespective of the state (ARM or Thumb):

```
SUBS    PC,R14_abt,#4    ;  for a prefetch abort, or
SUBS    PC,R14_abt,#8    ;  for a data abort
```

This restores both the PC and the CPSR, and retries the aborted instruction.

Software Interrupt

The software interrupt instruction (SWI) is used for entering Supervisor mode, usually to request a particular supervisor function. A SWI handler should return by executing the following irrespective of the state (ARM or Thumb):

```
MOV     PC,R14_svc
```

This restores the PC and CPSR, and returns to the instruction following the SWI.

NOTE

nFIQ, nIRQ, ISYNC, LOCK, BIGEND, and ABORT pins exist only in the ARM920T CPU core.

Undefined Instruction

When ARM920T comes across an instruction which it cannot handle, it takes the undefined instruction trap. This mechanism may be used to extend either the THUMB or ARM instruction set by software emulation.

After emulating the failed instruction, the trap handler should execute the following irrespective of the state (ARM or Thumb):

```
MOVS    PC,R14_und
```

This restores the CPSR and returns to the instruction following the undefined instruction.

Exception Vectors

The following table shows the exception vector addresses.

Table 2-3. Exception Vectors

Address	Exception	Mode in Entry
0x00000000	Reset	Supervisor
0x00000004	Undefined instruction	Undefined
0x00000008	Software Interrupt	Supervisor
0x0000000C	Abort (prefetch)	Abort
0x00000010	Abort (data)	Abort
0x00000014	Reserved	Reserved
0x00000018	IRQ	IRQ
0x0000001C	FIQ	FIQ

Exception Priorities

When multiple exceptions arise at the same time, a fixed priority system determines the order in which they are handled:

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 are mutually exclusive, since they each correspond to particular (non-overlapping) decodings of the current instruction.

If a data abort occurs at the same time as a FIQ, and FIQs are enabled (ie the CPSR's F flag is clear), ARM920T enters the data abort handler and then immediately proceeds to the FIQ vector. A normal return from FIQ will cause the data abort handler to resume execution. Placing data abort at a higher priority than FIQ is necessary to ensure that the transfer error does not escape detection. The time for this exception entry should be added to worst-case FIQ latency calculations.

INTERRUPT LATENCIES

The worst case latency for FIQ, assuming that it is enabled, consists of the longest time the request can take to pass through the synchronizer (Tsyncmax if asynchronous), plus the time for the longest instruction to complete (Tldm, the longest instruction is an LDM which loads all the registers including the PC), plus the time for the data abort entry (Texc), plus the time for FIQ entry (Tfiq). At the end of this time ARM920T will be executing the instruction at 0x1C.

Tsyncmax is 3 processor cycles, Tldm is 20 cycles, Texc is 3 cycles, and Tfiq is 2 cycles. The total time is therefore 28 processor cycles. This is just over 1.4 microseconds in a system which uses a continuous 20 MHz processor clock. The maximum IRQ latency calculation is similar, but must allow for the fact that FIQ has higher priority and could delay entry into the IRQ handling routine for an arbitrary length of time. The minimum latency for FIQ or IRQ consists of the shortest time the request can take through the synchronizer (*Tsyncmin*) plus *Tfiq*. This is 4 processor cycles.

RESET

When the nRESET signal goes LOW, ARM920T abandons the executing instruction and then continues to fetch instructions from incrementing word addresses.

When nRESET goes HIGH again, ARM920T:

1. Overwrites R14_svc and SPSR_svc by copying the current values of the PC and CPSR into them. The value of the saved PC and SPSR is not defined.
2. Forces M[4:0] to 10011 (Supervisor mode), sets the I and F bits in the CPSR, and clears the CPSR's T bit.
3. Forces the PC to fetch the next instruction from address 0x00.
4. Execution resumes in ARM state.

NOTES

3 ARM INSTRUCTION SET

INSTRUCTION SET SUMMAY

This chapter describes the ARM instruction set in the ARM920T core.

FORMAT SUMMARY

The ARM instruction set formats are shown below.

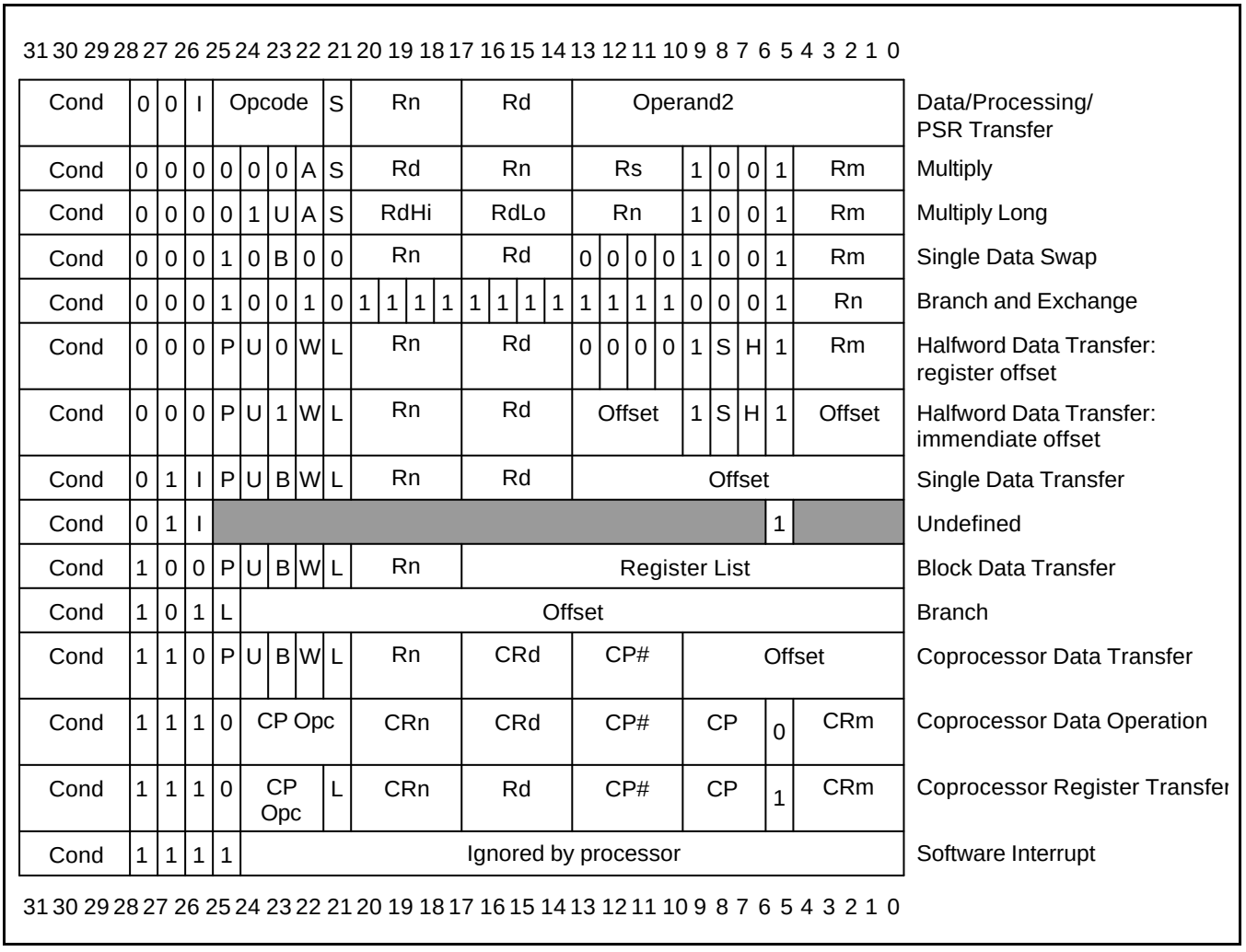


Figure 3-1. ARM Instruction Set Format

NOTE

Some instruction codes are not defined but do not cause the Undefined instruction trap to be taken, for instance a Multiply instruction with bit 6 changed to a 1. These instructions should not be used, as their action may change in future ARM implementations.

INSTRUCTION SUMMARY

Table 3-1. The ARM Instruction Set

Mnemonic	Instruction	Action
ADC	Add with carry	Rd: = Rn + Op2 + Carry
ADD	Add	Rd: = Rn + Op2
AND	AND	Rd: = Rn AND Op2
B	Branch	R15: = address
BIC	Bit Clear	Rd: = Rn AND NOT Op2
BL	Branch with Link	R14: = R15, R15: = address
BX	Branch and Exchange	R15: = Rn, T bit: = Rn[0]
CDP	Coprocessor Data Processing	(Coprocessor-specific)
CMN	Compare Negative	CPSR flags: = Rn + Op2
CMP	Compare	CPSR flags: = Rn - Op2
EOR	Exclusive OR	Rd: = (Rn AND NOT Op2) OR (Op2 AND NOT Rn)
LDC	Load coprocessor from memory	Coprocessor load
LDM	Load multiple registers	Stack manipulation (Pop)
LDR	Load register from memory	Rd: = (address)
MCR	Move CPU register to coprocessor register	cRn: = rRn {<op>cRm}
MLA	Multiply Accumulate	Rd: = (Rm × Rs) + Rn
MOV	Move register or constant	Rd: = Op2

Table 3-1. The ARM Instruction Set (Continued)

Mnemonic	Instruction	Action
MRC	Move from coprocessor register to CPU register	Rn: = cRn {<op>cRm}
MRS	Move PSR status/flags to register	Rn: = PSR
MSR	Move register to PSR status/flags	PSR: = Rm
MUL	Multiply	Rd: = Rm × Rs
MVN	Move negative register	Rd: = 0 × FFFFFFFF EOR Op2
ORR	OR	Rd: = Rn OR Op2
RSB	Reverse Subtract	Rd: = Op2 - Rn
RSC	Reverse Subtract with Carry	Rd: = Op2 - Rn - 1 + Carry
SBC	Subtract with Carry	Rd: = Rn - Op2 - 1 + Carry
STC	Store coprocessor register to memory	address: = CRn
STM	Store Multiple	Stack manipulation (Push)
STR	Store register to memory	<address>: = Rd
SUB	Subtract	Rd: = Rn - Op2
SWI	Software Interrupt	OS call
SWP	Swap register with memory	Rd: = [Rn], [Rn] := Rm
TEQ	Test bitwise equality	CPSR flags: = Rn EOR Op2
TST	Test bits	CPSR flags: = Rn AND Op2

THE CONDITION FIELD

In ARM state, all instructions are conditionally executed according to the state of the CPSR condition codes and the instruction's condition field. This field (bits 31:28) determines the circumstances under which an instruction is to be executed. If the state of the C, N, Z and V flags fulfils the conditions encoded by the field, the instruction is executed, otherwise it is ignored.

There are sixteen possible conditions, each represented by a two-character suffix that can be appended to the instruction's mnemonic. For example, a Branch (B in assembly language) becomes BEQ for "Branch if Equal", which means the Branch will only be taken if the Z flag is set.

In practice, fifteen different conditions may be used: these are listed in Table 3-2. The sixteenth (1111) is reserved, and must not be used.

In the absence of a suffix, the condition field of most instructions is set to "Always" (suffix AL). This means the instruction will always be executed regardless of the CPSR condition codes.

Table 3-2. Condition Code Summary

Code	Suffix	Flags	Meaning
0000	EQ	Z set	equal
0001	NE	Z clear	not equal
0010	CS	C set	unsigned higher or same
0011	CC	C clear	unsigned lower
0100	MI	N set	negative
0101	PL	N clear	positive or zero
0110	VS	V set	overflow
0111	VC	V clear	no overflow
1000	HI	C set and Z clear	unsigned higher
1001	LS	C clear or Z set	unsigned lower or same
1010	GE	N equals V	greater or equal
1011	LT	N not equal to V	less than
1100	GT	Z clear AND (N equals V)	greater than
1101	LE	Z set OR (N not equal to V)	less than or equal
1110	AL	(ignored)	always

BRANCH AND EXCHANGE (BX)

This instruction is only executed if the condition is true. The various conditions are defined in Table 3-2.

This instruction performs a branch by copying the contents of a general register, Rn, into the program counter, PC. The branch causes a pipeline flush and refill from the address specified by Rn. This instruction also permits the instruction set to be exchanged. When the instruction is executed, the value of Rn[0] determines whether the instruction stream will be decoded as ARM or THUMB instructions.

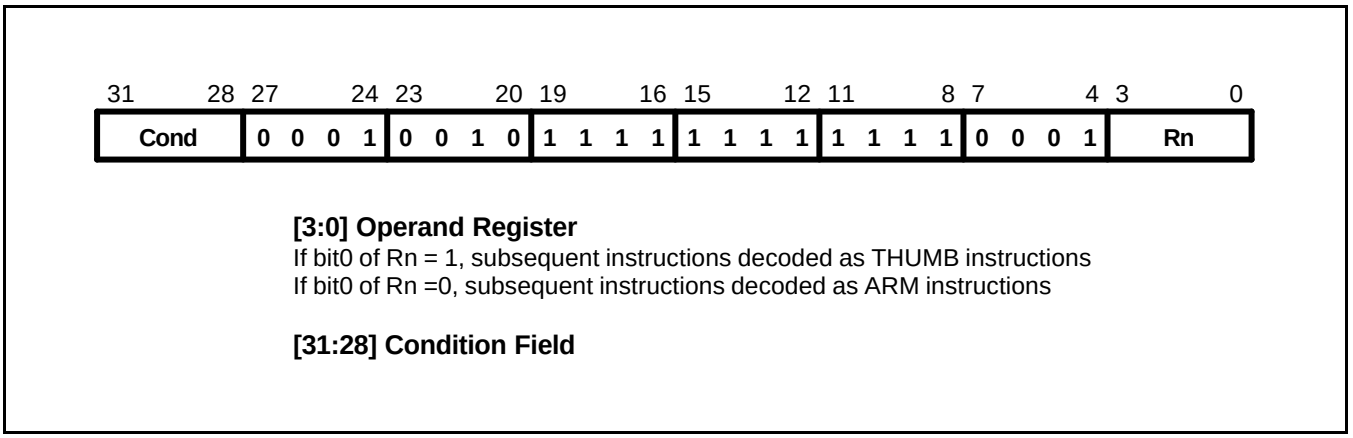


Figure 3-2. Branch and Exchange Instructions

INSTRUCTION CYCLE TIMES

The BX instruction takes 2S + 1N cycles to execute, where S and N are defined as sequential (S-cycle) and non-sequential (N-cycle), respectively.

ASSEMBLER SYNTAX

BX - branch and exchange.

BX {cond} Rn

{cond}	Two character condition mnemonic. See Table 3-2.
Rn	is an expression evaluating to a valid register number.

USING R15 AS AN OPERAND

If R15 is used as an operand, the behavior is undefined.

Examples

```
ADR       R0, Into_THUMB + 1    ; Generate branch target address
                                ; and set bit 0 high - hence
                                ; arrive in THUMB state.
BX        R0                    ; Branch and change to THUMB
                                ; state.
CODE16                          ; Assemble subsequent code as
Into_THUMB                      ; THUMB instructions
•
•
•
ADR R5, Back_to_ARM             ; Generate branch target to word aligned address
                                ; - hence bit 0 is low and so change back to ARM state.
BX R5                           ; Branch and change back to ARM state.
•
•
•
ALIGN                           ; Word align
CODE32                          ; Assemble subsequent code as ARM instructions
Back_to_ARM
```

BRANCH AND BRANCH WITH LINK (B, BL)

The instruction is only executed if the condition is true. The various conditions are defined Table 3-2. The instruction encoding is shown in Figure 3-3, below.

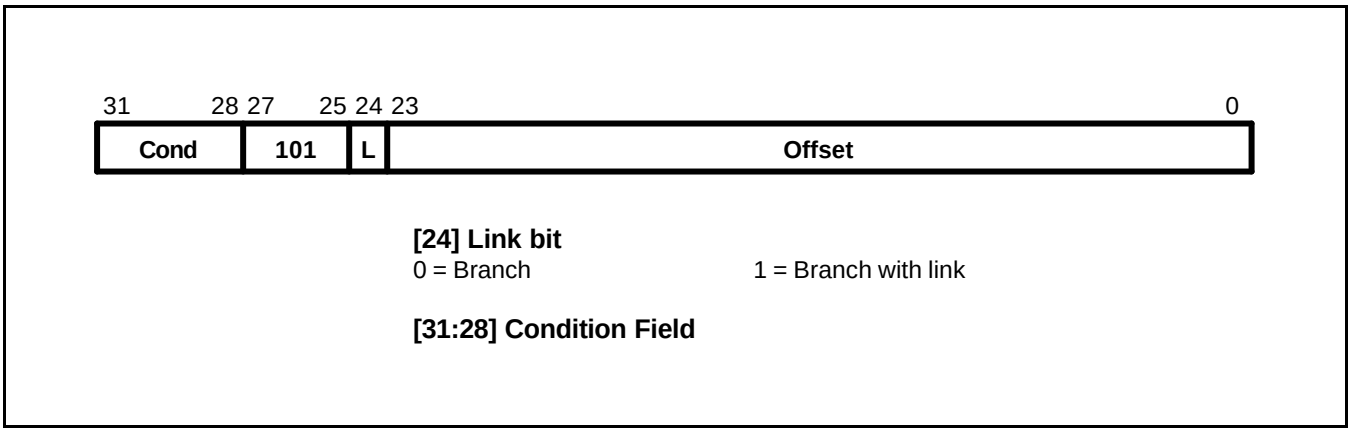


Figure 3-3. Branch Instructions

Branch instructions contain a signed 2's complement 24-bit offset. This is shifted left two bits, sign extended to 32 bits, and added to the PC. The instruction can therefore specify a branch of +/- 32Mbytes. The branch offset must take account of the prefetch operation, which causes the PC to be 2 words (8 bytes) ahead of the current instruction.

Branches beyond +/- 32Mbytes must use an offset or absolute destination which has been previously loaded into a register. In this case the PC should be manually saved in R14 if a Branch with Link type operation is required.

THE LINK BIT

Branch with Link (BL) writes the old PC into the link register (R14) of the current bank. The PC value written into R14 is adjusted to allow for the prefetch, and contains the address of the instruction following the branch and link instruction. Note that the CPSR is not saved with the PC and R14[1:0] are always cleared.

To return from a routine called by Branch with Link use MOV PC,R14 if the link register is still valid or LDM Rn!,{..PC} if the link register has been saved onto a stack pointed to by Rn.

INSTRUCTION CYCLE TIMES

Branch and Branch with Link instructions take 2S + 1N incremental cycles, where S and N are defined as sequential (S-cycle) and internal (I-cycle).

ASSEMBLER SYNTAX

Items in {} are optional. Items in <> must be present.

B{L}{cond} <expression>

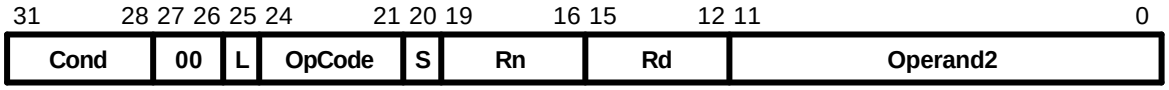
{L}	Used to request the Branch with Link form of the instruction. If absent, R14 will not be affected by the instruction.
{cond}	A two-character mnemonic as shown in Table 3-2. If absent then AL (ALways) will be used.
<expression>	The destination. The assembler calculates the offset.

Examples

```
here    BAL     here        ;  Assembles to 0xEAFFFFFE (note effect of PC offset).
        B       there       ;  Always condition used as default.
        CMP     R1,#0       ;  Compare R1 with zero and branch to fred
                            ;  if R1 was zero, otherwise continue.
        BEQ     fred        ;  Continue to next instruction.
        BL      sub+ROM     ;  Call subroutine at computed address.
        ADDS    R1,#1       ;  Add 1 to register 1, setting CPSR flags
                            ;  on the result then call subroutine if
        BLCC    sub         ;  the C flag is clear, which will be the
                            ;  case unless R1 held 0xFFFFFFFF.
```

DATA PROCESSING

The data processing instruction is only executed if the condition is true. The conditions are defined in Table 3-2. The instruction encoding is shown in Figure 3-4.



[15:12] Destination register

0 = Branch 1 = Branch with link

[19:16] 1st operand register

0 = Branch 1 = Branch with link

[20] Set condition codes

0 = Do not after condition codes 1 = Set condition codes

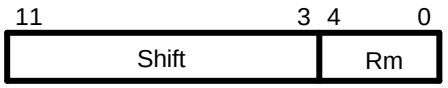
[24:21] Operation codes

0000 = AND-Rd: = Op1 AND Op2
0001 = EOR-Rd: = Op1 EOR Op2
0010 = SUB-Rd: = Op1-Op2
0011 = RSB-Rd: = Op2-Op1
0100 = ADD-Rd: = Op1+Op2
0101 = ADC-Rd: = Op1+Op2+C
0110 = SBC-Rd: = OP1-Op2+C-1
0111 = RSC-Rd: = Op2-Op1+C-1
1000 = TST-set condition codes on Op1 AND Op2
1001 = TEO-set condition codes on OP1 EOR Op2
1010 = CMP-set condition codes on Op1-Op2
1011 = SMN-set condition codes on Op1+Op2
1100 = ORR-Rd: = Op1 OR Op2
1101 = MOV-Rd: =Op2
1110 = BIC-Rd: = Op1 AND NOT Op2
1111 = MVN-Rd: = NOT Op2

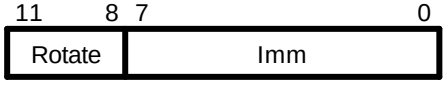
[25] Immediate operand

0 = Operand 2 is a register 1 = Operand 2 is an immediate value

[11:0] Operand 2 type selection



[3:0] 2nd operand register [11:4] Shift applied to Rm



[7:0] Unsigned 8 bit immediate value [11:8] Shift applied to Imm

[31:28] Condition field

Figure 3-4. Data Processing Instructions

The instruction produces a result by performing a specified arithmetic or logical operation on one or two operands. The first operand is always a register (Rn).

The second operand may be a shifted register (Rm) or a rotated 8 bit immediate value (Imm) according to the value of the I bit in the instruction. The condition codes in the CPSR may be preserved or updated as a result of this instruction, according to the value of the S bit in the instruction.

Certain operations (TST, TEQ, CMP, CMN) do not write the result to Rd. They are used only to perform tests and to set the condition codes on the result and always have the S bit set. The instructions and their effects are listed in Table 3-3.

CPSR FLAGS

The data processing operations may be classified as logical or arithmetic. The logical operations (AND, EOR, TST, TEQ, ORR, MOV, BIC, MVN) perform the logical action on all corresponding bits of the operand or operands to produce the result. If the S bit is set (and Rd is not R15, see below) the V flag in the CPSR will be unaffected, the C flag will be set to the carry out from the barrel shifter (or preserved when the shift operation is LSL #0), the Z flag will be set if and only if the result is all zeros, and the N flag will be set to the logical value of bit 31 of the result.

Table 3-3. ARM Data Processing Instructions

Assembler Mnemonic	OP Code	Action
AND	0000	Operand1 AND operand2
EOR	0001	Operand1 EOR operand2
WUB	0010	Operand1 - operand2
RSB	0011	Operand2 operand1
ADD	0100	Operand1 + operand2
ADC	0101	Operand1 + operand2 + carry
SBC	0110	Operand1 - operand2 + carry - 1
RSC	0111	Operand2 - operand1 + carry - 1
TST	1000	As AND, but result is not written
TEQ	1001	As EOR, but result is not written
CMP	1010	As SUB, but result is not written
CMN	1011	As ADD, but result is not written
ORR	1100	Operand1 OR operand2
MOV	1101	Operand2 (operand1 is ignored)
BIC	1110	Operand1 AND NOT operand2 (Bit clear)
MVN	1111	NOT operand2 (operand1 is ignored)

The arithmetic operations (SUB, RSB, ADD, ADC, SBC, RSC, CMP, CMN) treat each operand as a 32 bit integer (either unsigned or 2's complement signed, the two are equivalent). If the S bit is set (and Rd is not R15) the V flag in the CPSR will be set if an overflow occurs into bit 31 of the result; this may be ignored if the operands were considered unsigned, but warns of a possible error if the operands were 2's complement signed. The C flag will be set to the carry out of bit 31 of the ALU, the Z flag will be set if and only if the result was zero, and the N flag will be set to the value of bit 31 of the result (indicating a negative result if the operands are considered to be 2's complement signed).

SHIFTS

When the second operand is specified to be a shifted register, the operation of the barrel shifter is controlled by the Shift field in the instruction. This field indicates the type of shift to be performed (logical left or right, arithmetic right or rotate right). The amount by which the register should be shifted may be contained in an immediate field in the instruction, or in the bottom byte of another register (other than R15). The encoding for the different shift types is shown in Figure 3-5.

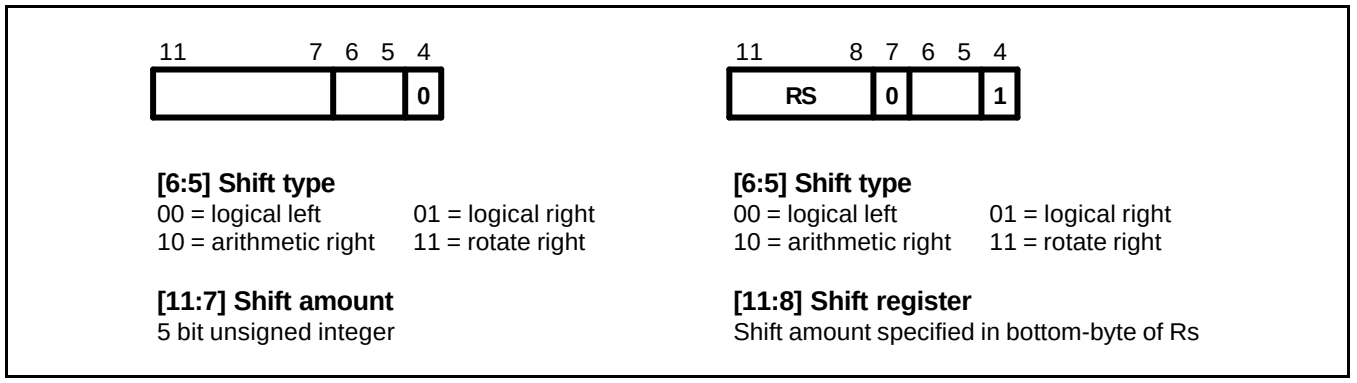


Figure 3-5. ARM Shift Operations

Instruction specified shift amount

When the shift amount is specified in the instruction, it is contained in a 5-bit field which may take any value from 0 to 31. A logical shift left (LSL) takes the contents of Rm and moves each bit by the specified amount to a more significant position. The least significant bits of the result are filled with zeros, and the high bits of Rm which do not map into the result are discarded, except that the least significant discarded bit becomes the shifter carry output which may be latched into the C bit of the CPSR when the ALU operation is in the logical class (see above). For example, the effect of LSL #5 is shown in Figure 3-6.

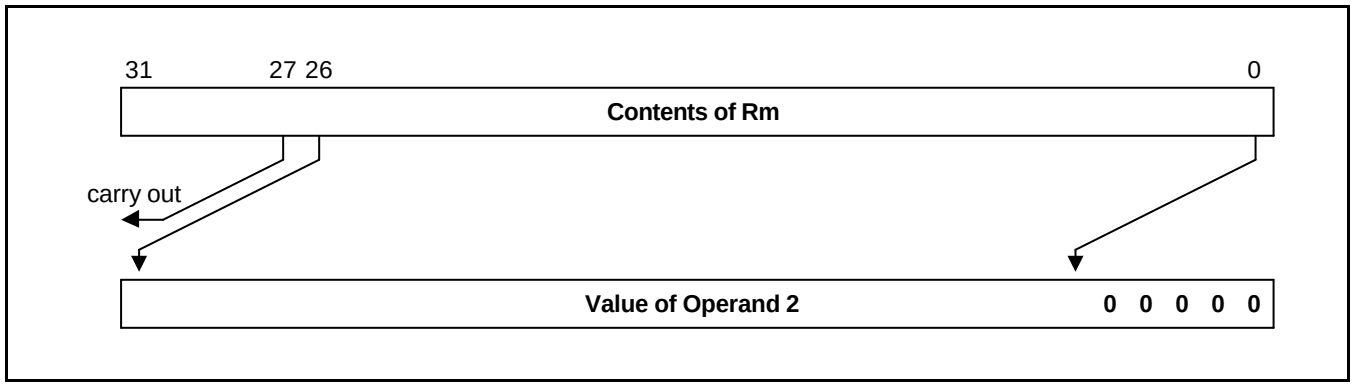


Figure 3-6. Logical Shift Left

NOTE

LSL #0 is a special case, where the shifter carry out is the old value of the CPSR C flag. The contents of Rm are used directly as the second operand. A logical shift right (LSR) is similar, but the contents of Rm are moved to less significant positions in the result. LSR #5 has the effect shown in Figure 3-7.

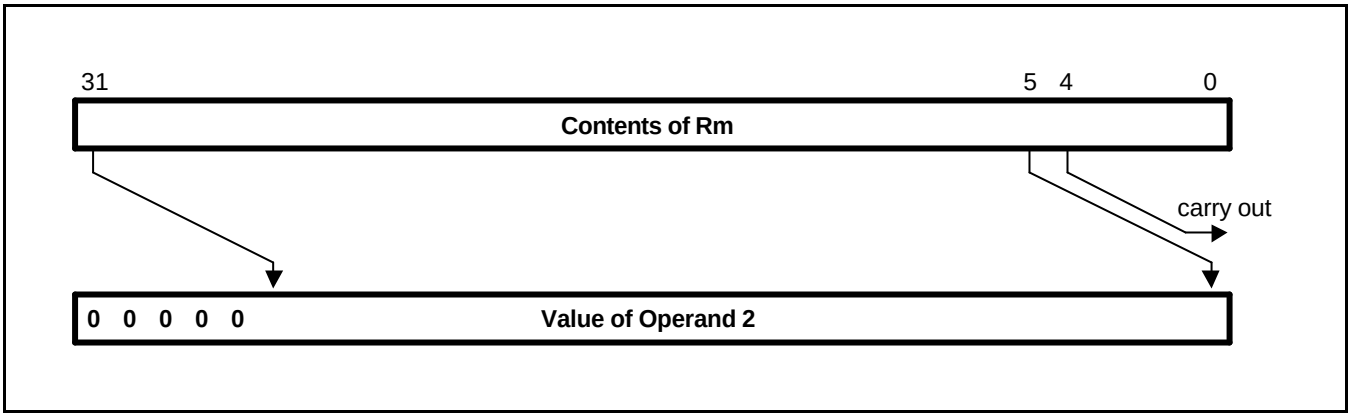


Figure 3-7. Logical Shift Right

The form of the shift field which might be expected to correspond to LSR #0 is used to encode LSR #32, which has a zero result with bit 31 of Rm as the carry output. Logical shift right zero is redundant as it is the same as logical shift left zero, so the assembler will convert LSR #0 (and ASR #0 and ROR #0) into LSL #0, and allow LSR #32 to be specified.

An arithmetic shift right (ASR) is similar to logical shift right, except that the high bits are filled with bit 31 of Rm instead of zeros. This preserves the sign in 2's complement notation. For example, ASR #5 is shown in Figure 3-8.

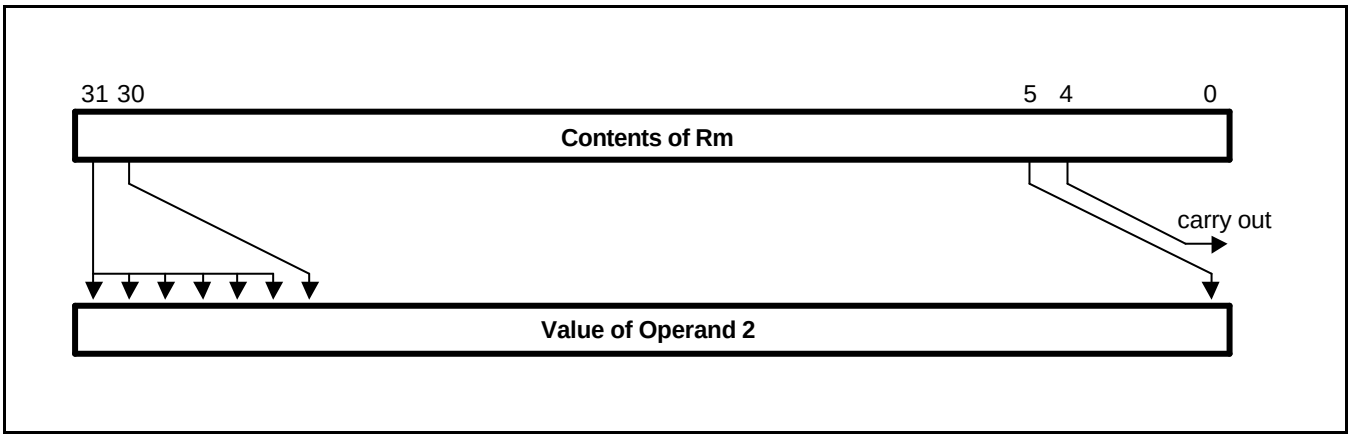


Figure 3-8. Arithmetic Shift Right

The form of the shift field which might be expected to give ASR #0 is used to encode ASR #32. Bit 31 of Rm is again used as the carry output, and each bit of operand 2 is also equal to bit 31 of Rm. The result is therefore all ones or all zeros, according to the value of bit 31 of Rm.

Rotate right (ROR) operations reuse the bits which "overshoot" in a logical shift right operation by reintroducing them at the high end of the result, in place of the zeros used to fill the high end in logical right operations. For example, ROR #5 is shown in Figure 3-9.

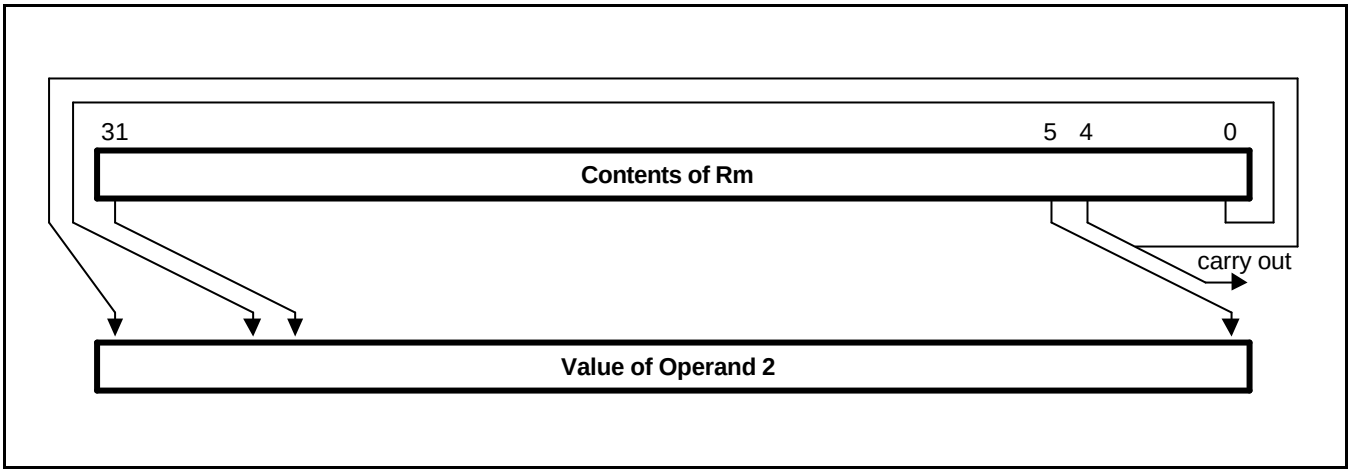


Figure 3-9. Rotate Right

The form of the shift field which might be expected to give ROR #0 is used to encode a special function of the barrel shifter, rotate right extended (RRX). This is a rotate right by one bit position of the 33 bit quantity formed by appending the CPSR C flag to the most significant end of the contents of Rm as shown in Figure 3-10.

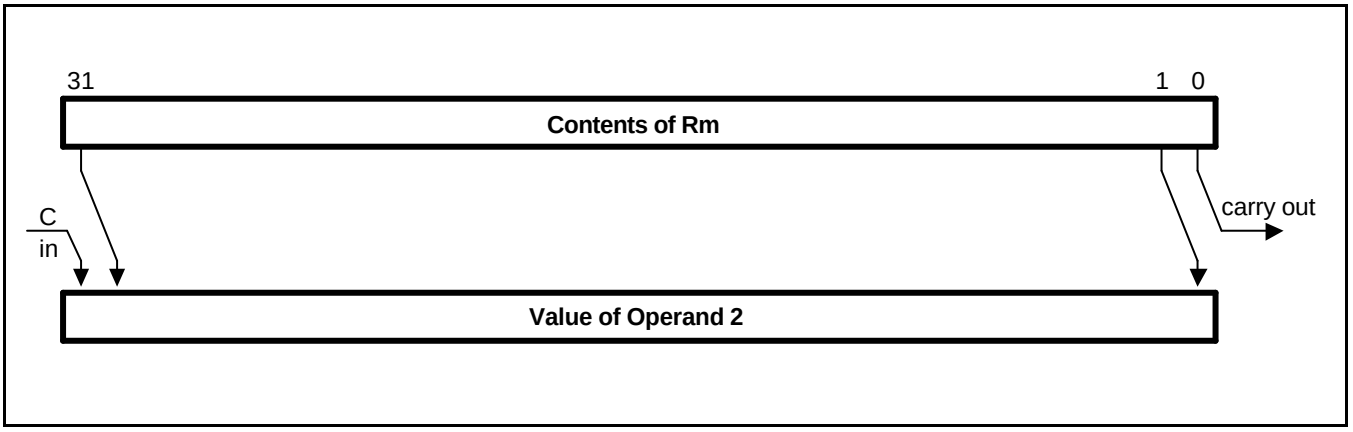


Figure 3-10. Rotate Right Extended

Register Specified Shift Amount

Only the least significant byte of the contents of Rs is used to determine the shift amount. Rs can be any general register other than R15.

If this byte is zero, the unchanged contents of Rm will be used as the second operand, and the old value of the CPSR C flag will be passed on as the shifter carry output.

If the byte has a value between 1 and 31, the shifted result will exactly match that of an instruction specified shift with the same value and shift operation.

If the value in the byte is 32 or more, the result will be a logical extension of the shift described above:

1. LSL by 32 has result zero, carry out equal to bit 0 of Rm.
2. LSL by more than 32 has result zero, carry out zero.
3. LSR by 32 has result zero, carry out equal to bit 31 of Rm.
4. LSR by more than 32 has result zero, carry out zero.
5. ASR by 32 or more has result filled with and carry out equal to bit 31 of Rm.
6. ROR by 32 has result equal to Rm, carry out equal to bit 31 of Rm.
7. ROR by n where n is greater than 32 will give the same result and carry out as ROR by n-32; therefore repeatedly subtract 32 from n until the amount is in the range 1 to 32 and see above.

NOTE

The zero in bit 7 of an instruction with a register controlled shift is compulsory; a one in this bit will cause the instruction to be a multiply or undefined instruction.

IMMEDIATE OPERAND ROTATES

The immediate operand rotate field is a 4-bit unsigned integer which specifies a shift operation on the 8-bit immediate value. This value is zero extended to 32 bits, and then subject to a rotate right by twice the value in the rotate field. This enables many common constants to be generated, for example all powers of 2.

WRITING TO R15

When Rd is a register other than R15, the condition code flags in the CPSR may be updated from the ALU flags as described above.

When Rd is R15 and the S flag in the instruction is not set the result of the operation is placed in R15 and the CPSR is unaffected.

When Rd is R15 and the S flag is set the result of the operation is placed in R15 and the SPSR corresponding to the current mode is moved to the CPSR. This allows state changes which atomically restore both PC and CPSR. This form of instruction should not be used in User mode.

USING R15 AS AN OPERANDY

If R15 (the PC) is used as an operand in a data processing instruction the register is used directly.

The PC value will be the address of the instruction, plus 8 or 12 bytes due to instruction prefetching. If the shift amount is specified in the instruction, the PC will be 8 bytes ahead. If a register is used to specify the shift amount the PC will be 12 bytes ahead.

TEQ, TST, CMP AND CMN OPCODES

NOTE

TEQ, TST, CMP and CMN do not write the result of their operation but do set flags in the CPSR. An assembler should always set the S flag for these instructions even if this is not specified in the mnemonic.

The TEQP form of the TEQ instruction used in earlier ARM processors must not be used: the PSR transfer operations should be used instead.

The action of TEQP in the ARM920T is to move SPSR_<mode> to the CPSR if the processor is in a privileged mode and to do nothing if in User mode.

INSTRUCTION CYCLE TIMES

Data Processing instructions vary in the number of incremental cycles taken as follows:

Table 3-4. Incremental Cycle Times

Processing Type	Cycles
Normal data processing	1S
Data processing with register specified shift	1S + 1I
Data processing with PC written	2S + 1N
Data processing with register specified shift and PC written	2S + 1N +1I

NOTE: S, N and I are as defined sequential (S-cycle), non-sequential (N-cycle), and internal (I-cycle) respectively.

ASSEMBLER SYNTAX

- MOV,MVN (single operand instructions).
  <opcode>{cond}{S} Rd,<Op2>
- CMP,CMN,TEQ,TST (instructions which do not produce a result).
  <opcode>{cond} Rn,<Op2>
- AND,EOR,SUB,RSB,ADD,ADC,SBC,RSC,ORR,BIC
  <opcode>{cond}{S} Rd,Rn,<Op2>

where:

<Op2>	Rm{,<shift>} or,<#expression>
{cond}	A two-character condition mnemonic. See Table 3-2.
{S}	Set condition codes if S present (implied for CMP, CMN, TEQ, TST).
Rd, Rn and Rm	Expressions evaluating to a register number.
<#expression>	If this is used, the assembler will attempt to generate a shifted immediate 8-bit field to match the expression. If this is impossible, it will give an error.
<shift>	<Shiftname> <register> or <shiftname> #expression, or RRX (rotate right one bit with extend).
<shiftname>s	ASL, LSL, LSR, ASR, ROR. (ASL is a synonym for LSL, they assemble to the same code.)

EXAMPLES

```
ADDEQ   R2,R4,R5        ; If the Z flag is set make R2:=R4+R5
TEQS    R4,#3           ; Test R4 for equality with 3.
                        ; (The S is in fact redundant as the
                        ; assembler inserts it automatically.)
SUB     R4,R5,R7,LSR R2 ; Logical right shift R7 by the number in
                        ; the bottom byte of R2, subtract result
                        ; from R5, and put the answer into R4.
MOV     PC,R14          ; Return from subroutine.
MOVS    PC,R14          ; Return from exception and restore CPSR
                        ; from SPSR_mode.
```

PSR TRANSFER (MRS, MSR)

The instruction is only executed if the condition is true. The various conditions are defined in Table 3-2.

The MRS and MSR instructions are formed from a subset of the Data Processing operations and are implemented using the TEQ, TST, CMN and CMP instructions without the S flag set. The encoding is shown in Figure 3-11.

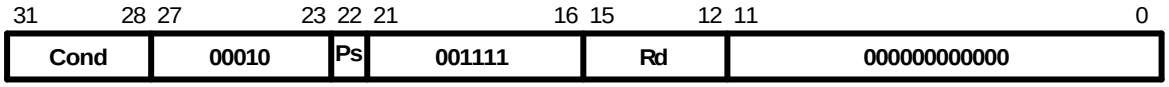
These instructions allow access to the CPSR and SPSR registers. The MRS instruction allows the contents of the CPSR or SPSR_<mode> to be moved to a general register. The MSR instruction allows the contents of a general register to be moved to the CPSR or SPSR_<mode> register.

The MSR instruction also allows an immediate value or register contents to be transferred to the condition code flags (N,Z,C and V) of CPSR or SPSR_<mode> without affecting the control bits. In this case, the top four bits of the specified register contents or 32 bit immediate value are written to the top four bits of the relevant PSR.

OPERAND RESTRICTIONS

- In user mode, the control bits of the CPSR are protected from change, so only the condition code flags of the CPSR can be changed. In other (privileged) modes the entire CPSR can be changed.
- Note that the software must never change the state of the T bit in the CPSR. If this happens, the processor will enter an unpredictable state.
- The SPSR register which is accessed depends on the mode at the time of execution. For example, only SPSR_fiq is accessible when the processor is in FIQ mode.
- You must not specify R15 as the source or destination register.
- Also, do not attempt to access an SPSR in User mode, since no such register exists.

MRS (transfer PSR contents to a register)

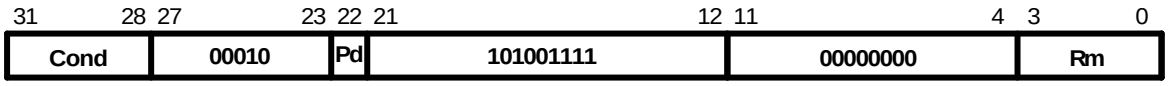


[15:12] Destination Register

[22] Source PSR
0 = CPSR 1 = SPSR_<current mode>

[31:28] Condition Field

MSR (transfer register contents to PSR)

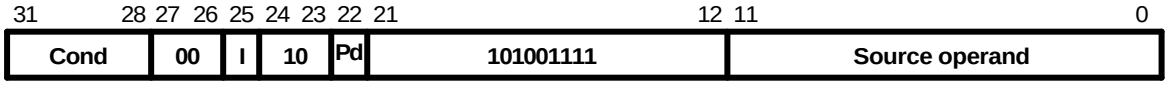


[3:0] Source Register

[22] Destination PSR
0 = CPSR 1 = SPSR_<current mode>

[31:28] Condition Field

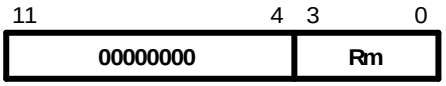
MSR (transfer register contents or immediate value to PSR flag bits only)



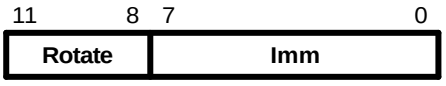
[22] Destination PSR
0 = CPSR 1 = SPSR_<current mode>

[25] Immediate Operand
0 = Source operand is a register
1 = SPSR_<current mode>

[11:0] Source Operand



[3:0] Source Register
[11:4] Source operand is an immediate value



[7:0] Unsigned 8 bit immediate value
[11:8] Shift applied to Imm

[31:28] Condition Field

Figure 3-11. PSR Transfer

RESERVED BITS

Only twelve bits of the PSR are defined in ARM920T (N,Z,C,V,I,F, T & M[4:0]); the remaining bits are reserved for use in future versions of the processor. Refer to Figure 2-6 for a full description of the PSR bits.

To ensure the maximum compatibility between ARM920T programs and future processors, the following rules should be observed:

- The reserved bits should be preserved when changing the value in a PSR.
- Programs should not rely on specific values from the reserved bits when checking the PSR status, since they may read as one or zero in future processors.

A read-modify-write strategy should therefore be used when altering the control bits of any PSR register; this involves transferring the appropriate PSR register to a general register using the MRS instruction, changing only the relevant bits and then transferring the modified value back to the PSR register using the MSR instruction.

EXAMPLES

The following sequence performs a mode change:

```
MRS     R0,CPSR          ;  Take a copy of the CPSR.
BIC     R0,R0,#0x1F      ;  Clear the mode bits.
ORR     R0,R0,#new_mode  ;  Select new mode
MSR     CPSR,R0          ;  Write back the modified CPSR.
```

When the aim is simply to change the condition code flags in a PSR, a value can be written directly to the flag bits without disturbing the control bits. The following instruction sets the N,Z,C and V flags:

```
MSR     CPSR_flg,#0xF0000000     ;  Set all the flags regardless of their previous state
                                 ;  (does not affect any control bits).
```

No attempt should be made to write an 8-bit immediate value into the whole PSR since such an operation cannot preserve the reserved bits.

INSTRUCTION CYCLE TIMES

PSR transfers take 1S incremental cycles, where S is defined as Sequential (S-cycle).

ASSEMBLY SYNTAX

- MRS - transfer PSR contents to a register
  MRS{cond} Rd,<psr>
- MSR - transfer register contents to PSR
  MSR{cond} <psr>,Rm
- MSR - transfer register contents to PSR flag bits only
  MSR{cond} <psrf>,Rm

The most significant four bits of the register contents are written to the N,Z,C & V flags respectively.

- MSR - transfer immediate value to PSR flag bits only
  MSR{cond} <psrf>,<#expression>

The expression should symbolise a 32-bit value of which the most significant four bits are written to the N,Z,C and V flags respectively.

Key:

{cond}	Two-character condition mnemonic. See Table 3-2..
Rd and Rm	Expressions evaluating to a register number other than R15
<psr>	CPSR, CPSR_all, SPSR or SPSR_all. (CPSR and CPSR_all are synonyms as are SPSR and SPSR_all)
<psrf>	CPSR_flg or SPSR_flg
<#expression>	Where this is used, the assembler will attempt to generate a shifted immediate 8-bit field to match the expression. If this is impossible, it will give an error.

EXAMPLES

In User mode the instructions behave as follows:

```
MSR     CPSR_all,Rm             ;  CPSR[31:28] <- Rm[31:28]
MSR     CPSR_flg,Rm             ;  CPSR[31:28] <- Rm[31:28]
MSR     CPSR_flg,#0xA0000000    ;  CPSR[31:28] <- 0xA (set N,C; clear Z,V)
MRS     Rd,CPSR                 ;  Rd[31:0] <- CPSR[31:0]
```

In privileged modes the instructions behave as follows:

```
MSR     CPSR_all,Rm             ;  CPSR[31:0]  <- Rm[31:0]
MSR     CPSR_flg,Rm             ;  CPSR[31:28] <- Rm[31:28]
MSR     CPSR_flg,#0x50000000    ;  CPSR[31:28] <- 0x5 (set Z,V; clear N,C)
MSR     SPSR_all,Rm             ;  SPSR_<mode>[31:0]<- Rm[31:0]
MSR     SPSR_flg,Rm             ;  SPSR_<mode>[31:28] <- Rm[31:28]
MSR     SPSR_flg,#0xC0000000    ;  SPSR_<mode>[31:28] <- 0xC (set N,Z; clear C,V)
MRS     Rd,SPSR                 ;  Rd[31:0] <- SPSR_<mode>[31:0]
```

MULTIPLY AND MULTIPLY-ACCUMULATE (MUL, MLA)

The instruction is only executed if the condition is true. The various conditions are defined in Table 3-2. The instruction encoding is shown in Figure 3-12.

The multiply and multiply-accumulate instructions use an 8-bit Booth's algorithm to perform integer multiplication.

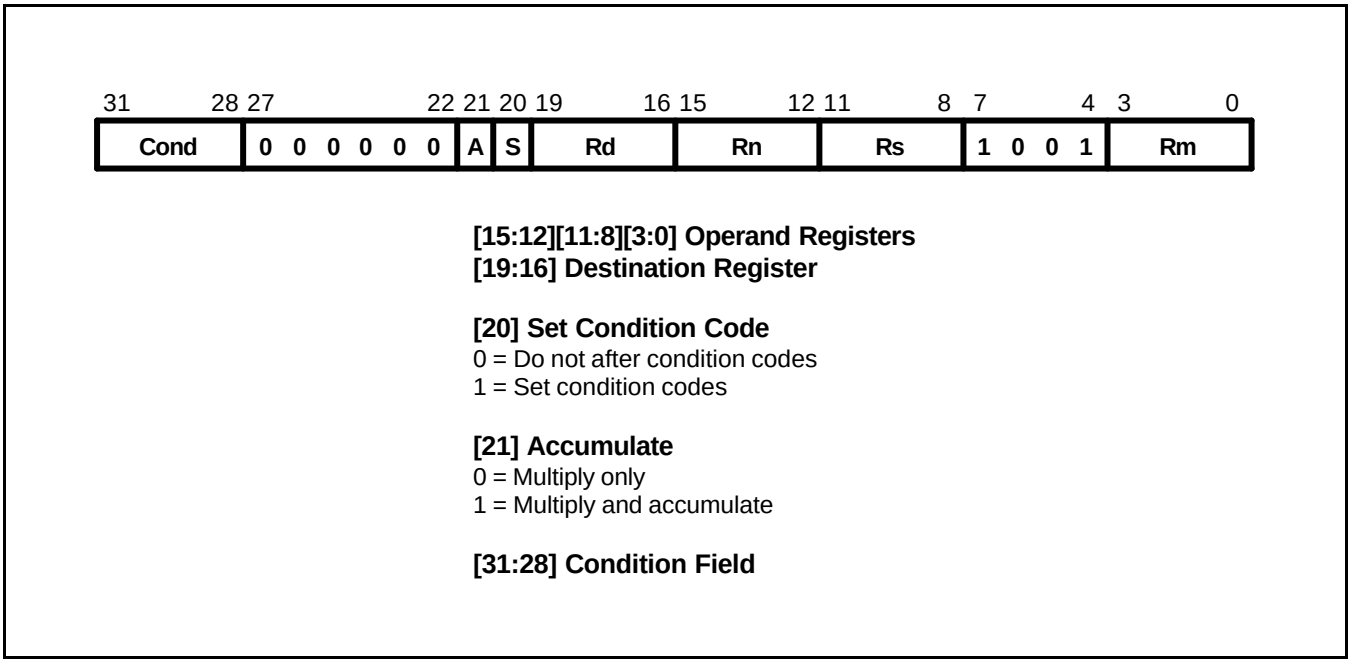


Figure 3-12. Multiply Instructions

The multiply form of the instruction gives Rd:=Rm*Rs. Rn is ignored, and should be set to zero for compatibility with possible future upgrades to the instruction set. The multiply-accumulate form gives Rd:=Rm*Rs+Rn, which can save an explicit ADD instruction in some circumstances. Both forms of the instruction work on operands which may be considered as signed (2's complement) or unsigned integers.

The results of a signed multiply and of an unsigned multiply of 32-bit operands differ only in the upper 32 bits - the low 32 bits of the signed and unsigned results are identical. As these instructions only produce the low 32 bits of a multiply, they can be used for both signed and unsigned multiplies.

For example consider the multiplication of the operands:

Operand A	Operand B	Result
0xFFFFFFF6	0x0000001	0xFFFFFF38

If the Operands Are Interpreted as Signed

Operand A has the value -10, operand B has the value 20, and the result is -200 which is correctly represented as 0xFFFFFF38.

If the Operands Are Interpreted as Unsigned

Operand A has the value 4294967286, operand B has the value 20 and the result is 85899345720, which is represented as 0x13FFFFFF38, so the least significant 32 bits are 0xFFFFFF38.

Operand Restrictions

The destination register Rd must not be the same as the operand register Rm. R15 must not be used as an operand or as the destination register.

All other register combinations will give correct results, and Rd, Rn and Rs may use the same register when required.

CPSR FLAGS

Setting the CPSR flags is optional, and is controlled by the S bit in the instruction. The N (Negative) and Z (Zero) flags are set correctly on the result (N is made equal to bit 31 of the result, and Z is set if and only if the result is zero). The C (Carry) flag is set to a meaningless value and the V (oVerflow) flag is unaffected.

INSTRUCTION CYCLE TIMES

MUL takes 1S + mI and MLA 1S + (m+1)I cycles to execute, where S and I are defined as sequential (S-cycle) and internal (I-cycle), respectively.

m	The number of 8-bit multiplier array cycles is required to complete the multiply, which is controlled by the value of the multiplier operand specified by Rs. Its possible values are as follows
1	If bits [32:8] of the multiplier operand are all zero or all one.
2	If bits [32:16] of the multiplier operand are all zero or all one.
3	If bits [32:24] of the multiplier operand are all zero or all one.
4	In all other cases.

ASSEMBLER SYNTAX

MUL{cond}{S} Rd,Rm,Rs
MLA{cond}{S} Rd,Rm,Rs,Rn

{cond}	Two-character condition mnemonic. See Table 3-2..
{S}	Set condition codes if S present
Rd, Rm, Rs and Rn	Expressions evaluating to a register number other than R15.

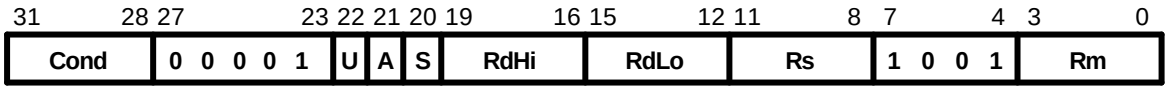
EXAMPLES

```
MUL     R1,R2,R3        ; R1:=R2*R3
MLAEQS  R1,R2,R3,R4     ; Conditionally R1:=R2*R3+R4, Setting condition codes.
```

MULTIPLY LONG AND MULTIPLY-ACCUMULATE LONG (MULL, MLAL)

The instruction is only executed if the condition is true. The various conditions are defined in Table 3-2. The instruction encoding is shown in Figure 3-13.

The multiply long instructions perform integer multiplication on two 32-bit operands and produce 64-bit results. Signed and unsigned multiplication each with optional accumulate give rise to four variations.



[11:8][3:0] Operand Registers
[19:16][15:12] Source Destination Registers

[20] Set Condition Code
0 = Do not alter condition codes
1 = Set condition codes

[21] Accumulate
0 = Multiply only
1 = Multiply and accumulate

[22] Unsigned
0 = Unsigned
1 = Signed

[31:28] Condition Field

Figure 3-13. Multiply Long Instructions

The multiply forms (UMULL and SMULL) take two 32-bit numbers and multiply them to produce a 64-bit result of the form RdHi,RdLo := Rm * Rs. The lower 32 bits of the 64-bit result are written to RdLo, the upper 32 bits of the result are written to RdHi.

The multiply-accumulate forms (UMLAL and SMLAL) take two 32-bit numbers, multiply them and add a 64 bit number to produce a 64-bit result of the form RdHi,RdLo := Rm * Rs + RdHi,RdLo. The lower 32 bits of the 64-bit number to add is read from RdLo. The upper 32 bits of the 64 bit number to add is read from RdHi. The lower 32 bits of the 64-bit result are written to RdLo. The upper 32 bits of the 64 bit result are written to RdHi.

The UMULL and UMLAL instructions treat all of their operands as unsigned binary numbers and write an unsigned 64 bit result. The SMULL and SMLAL instructions treat all of their operands as two's-complement signed numbers and write a two's-complement signed 64-bit result.

OPERAND RESTRICTIONS

- R15 must not be used as an operand or as a destination register.
- RdHi, RdLo, and Rm must all specify different registers.

CPSR FLAGS

Setting the CPSR flags is optional, and is controlled by the S bit in the instruction. The N and Z flags are set correctly on the result (N is equal to bit 63 of the result, Z is set if and only if all 64 bits of the result are zero). Both the C and V flags are set to meaningless values.

INSTRUCTION CYCLE TIMES

MULL takes 1S + (m+1)I and MLAL 1S + (m+2)I cycles to execute, where m is the number of 8 bit multiplier array cycles required to complete the multiply, which is controlled by the value of the multiplier operand specified by Rs.

Its possible values are as follows:

For Signed INSTRUCTIONS SMULL, SMLAL:

- If bits [31:8] of the multiplier operand are all zero or all one.
- If bits [31:16] of the multiplier operand are all zero or all one.
- If bits [31:24] of the multiplier operand are all zero or all one.
- In all other cases.

For Unsigned Instructions UMULL, UMLAL:

- If bits [31:8] of the multiplier operand are all zero.
- If bits [31:16] of the multiplier operand are all zero.
- If bits [31:24] of the multiplier operand are all zero.
- In all other cases.

S and I are defined as sequential (S-cycle) and internal (I-cycle), respectively.

ASSEMBLER SYNTAX

Table 3-5. Assembler Syntax Descriptions

Mnemonic	Description	Purpose
UMULL{cond}{S} RdLo,RdHi,Rm,Rs	Unsigned Multiply Long	32 x 32 = 64
UMLAL{cond}{S} RdLo,RdHi,Rm,Rs	Unsigned Multiply & Accumulate Long	32 x 32 + 64 = 64
SMULL{cond}{S} RdLo,RdHi,Rm,Rs	Signed Multiply Long	32 x 32 = 64
SMLAL{cond}{S} RdLo,RdHi,Rm,Rs	Signed Multiply & Accumulate Long	32 x 32 + 64 = 64

where:

{cond}	Two-character condition mnemonic. See Table 3-2.
{S}	Set condition codes if S present
RdLo, RdHi, Rm, Rs	Expressions evaluating to a register number other than R15.

EXAMPLES

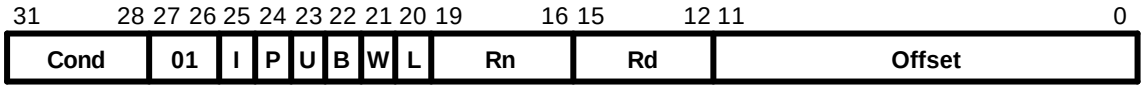
```
UMULL    R1,R4,R2,R3      ; R4,R1:=R2*R3
UMLALS   R1,R5,R2,R3      ; R5,R1:=R2*R3+R5,R1 also setting condition codes
```

SINGLE DATA TRANSFER (LDR, STR)

The instruction is only executed if the condition is true. The various conditions are defined in Table 3-2. The instruction encoding is shown in Figure 3-14.

The single data transfer instructions are used to load or store single bytes or words of data. The memory address used in the transfer is calculated by adding an offset to or subtracting an offset from a base register.

The result of this calculation may be written back into the base register if auto-indexing is required.



[15:12] Source/Destination Registers

[19:16] Base Register

[20] Load/Store Bit
0 = Store to memory
1 = Load from memory

[21] Write-back Bit
0 = No write-back
1 = Write address into base

[22] Byte/Word Bit
0 = Transfer word quantity
1 = Transfer byte quantity

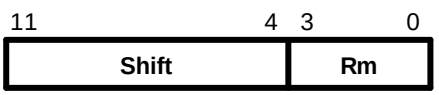
[23] Up/Down Bit
0 = Down: subtract offset from base
1 = Up: add offset to base

[24] Pre/Post Indexing Bit
0 = Post: add offset after transfer
1 = Pre: add offset before transfer

[25] Immediate Offset
0 = Offset is an immediate value

[11:0] Offset

[11:0] Unsigned 12-bit immediate offset



[3:0] Offset register [11:4] Shift applied to Rm

[31:28] Condition Field

Figure 3-14. Single Data Transfer Instructions

OFFSETS AND AUTO-INDEXING

The offset from the base may be either a 12-bit unsigned binary immediate value in the instruction, or a second register (possibly shifted in some way). The offset may be added to (U = 1) or subtracted from (U = 0) the base register Rn. The offset modification may be performed either before (pre-indexed, P = 1) or after (post-indexed, P = 0) the base is used as the transfer address.

The W bit gives optional auto increment and decrement addressing modes. The modified base value may be written back into the base (W = 1), or the old base value may be kept (W = 0). In the case of post-indexed addressing, the write back bit is redundant and is always set to zero, since the old base value can be retained by setting the offset to zero. Therefore post-indexed data transfers always write back the modified base. The only use of the W bit in a post-indexed data transfer is in privileged mode code, where setting the W bit forces non-privileged mode for the transfer, allowing the operating system to generate a user address in a system where the memory management hardware makes suitable use of this hardware.

SHIFTED REGISTER OFFSET

The 8 shift control bits are described in the data processing instructions section. However, the register specified shift amounts are not available in this instruction class. See Figure 3-5.

BYTES AND WORDS

This instruction class may be used to transfer a byte (B = 1) or a word (B = 0) between an ARM920T register and memory.

The action of LDR(B) and STR(B) instructions is influenced by the **BIGEND** control signal of ARM920T core. The two possible configurations are described below.

Little-Endian Configuration

A byte load (LDRB) expects the data on data bus inputs 7 through 0 if the supplied address is on a word boundary, on data bus inputs 15 through 8 if it is a word address plus one byte, and so on. The selected byte is placed in the bottom 8 bits of the destination register, and the remaining bits of the register are filled with zeros. Please see Figure 2-2.

A byte store (STRB) repeats the bottom 8 bits of the source register four times across data bus outputs 31 through 0. The external memory system should activate the appropriate byte subsystem to store the data.

A word load (LDR) will normally use a word aligned address. However, an address offset from a word boundary will cause the data to be rotated into the register so that the addressed byte occupies bits 0 to 7. This means that half-words accessed at offsets 0 and 2 from the word boundary will be correctly loaded into bits 0 through 15 of the register. Two shift operations are then required to clear or to sign extend the upper 16 bits.

A word store (STR) should generate a word aligned address. The word presented to the data bus is not affected if the address is not word aligned. That is, bit 31 of the register being stored always appears on data bus output 31.

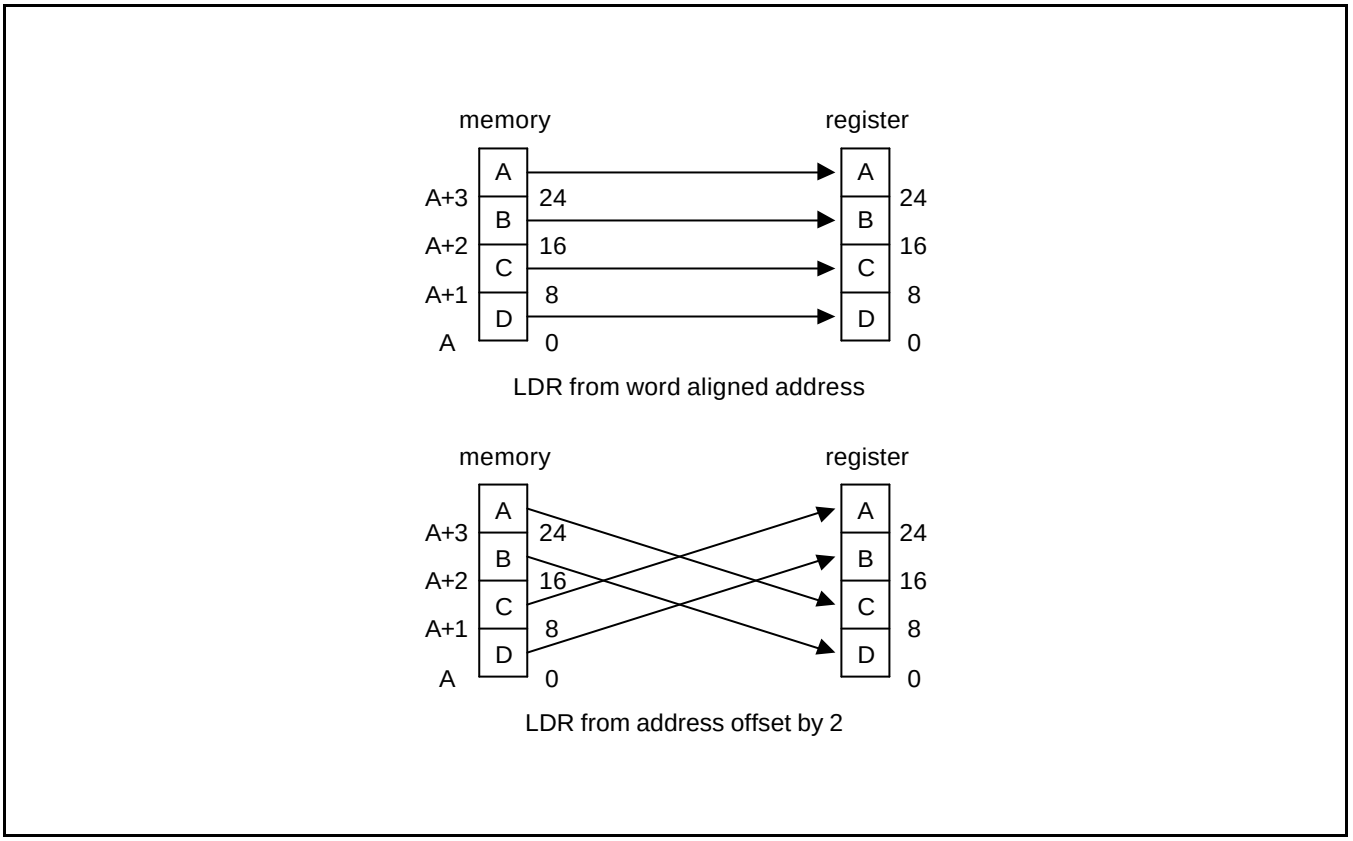


Figure 3-15. Little-Endian Offset Addressing

Big-Endian Configuration

A byte load (LDRB) expects the data on data bus inputs 31 through 24 if the supplied address is on a word boundary, on data bus inputs 23 through 16 if it is a word address plus one byte, and so on. The selected byte is placed in the bottom 8 bits of the destination register and the remaining bits of the register are filled with zeros. Please see Figure 2-1.

A byte store (STRB) repeats the bottom 8 bits of the source register four times across data bus outputs 31 through 0. The external memory system should activate the appropriate byte subsystem to store the data.

A word load (LDR) should generate a word aligned address. An address offset of 0 or 2 from a word boundary will cause the data to be rotated into the register so that the addressed byte occupies bits 31 through 24. This means that half-words accessed at these offsets will be correctly loaded into bits 16 through 31 of the register. A shift operation is then required to move (and optionally sign extend) the data into the bottom 16 bits. An address offset of 1 or 3 from a word boundary will cause the data to be rotated into the register so that the addressed byte occupies bits 15 through 8.

A word store (STR) should generate a word aligned address. The word presented to the data bus is not affected if the address is not word aligned. That is, bit 31 of the register being stored always appears on data bus output 31.

USE OF R15

Write-back must not be specified if R15 is specified as the base register (Rn). When using R15 as the base register you must remember it contains an address 8 bytes on from the address of the current instruction.

R15 must not be specified as the register offset (Rm).

When R15 is the source register (Rd) of a register store (STR) instruction, the stored value will be address of the instruction plus 12.

Restriction on the use of base register

When configured for late aborts, the following example code is difficult to unwind as the base register, Rn, gets updated before the abort handler starts. Sometimes it may be impossible to calculate the initial value.

After an abort, the following example code is difficult to unwind as the base register, Rn, gets updated before the abort handler starts. Sometimes it may be impossible to calculate the initial value.

EXAMPLE:

```
LDR     R0,[R1],R1
```

Therefore a post-indexed LDR or STR where Rm is the same register as Rn should not be used.

DATA ABORTS

A transfer to or from a legal address may cause problems for a memory management system. For instance, in a system which uses virtual memory the required data may be absent from main memory. The memory manager can signal a problem by taking the processor ABORT input HIGH whereupon the Data Abort trap will be taken. It is up to the system software to resolve the cause of the problem, then the instruction can be restarted and the original program continued.

INSTRUCTION CYCLE TIMES

Normal LDR instructions take 1S + 1N + 1I and LDR PC take 2S + 2N +1I incremental cycles, where S,N and I are defined as sequential (S-cycle), non-sequential (N-cycle), and internal (I-cycle), respectively. STR instructions take 2N incremental cycles to execute.

ASSEMBLER SYNTAX

<LDR|STR>{cond}{B}{T} Rd,<Address>

where:

LDR	Load from memory into a register
STR	Store from a register into memory
{cond}	Two-character condition mnemonic. See Table 3-2.
{B}	If B is present then byte transfer, otherwise word transfer
{T}	If T is present the W bit will be set in a post-indexed instruction, forcing non-privileged mode for the transfer cycle. T is not allowed when a pre-indexed addressing mode is specified or implied.
Rd	An expression evaluating to a valid register number.
Rn and Rm	Expressions evaluating to a register number. If Rn is R15 then the assembler will subtract 8 from the offset value to allow for ARM920T pipelining. In this case base write-back should not be specified.

<Address>can be:

1	An expression which generates an address: The assembler will attempt to generate an instruction using the PC as a base and a corrected immediate offset to address the location given by evaluating the expression. This will be a PC relative, pre-indexed address. If the address is out of range, an error will be generated.	
2	A pre-indexed addressing specification: [Rn] offset of zero [Rn,<#expression>]{!} offset of <expression> bytes [Rn,{+/-}Rm{,<shift>}]{!} offset of +/- contents of index register, shifted by <shift>	
3	A post-indexed addressing specification: [Rn],<#expression> offset of <expression> bytes [Rn],{+/-}Rm{,<shift>} offset of +/- contents of index register, shifted as by <shift>.	
<shift>	General shift operation (see data processing instructions) but you cannot specify the shift amount by a register.	
{!}	Writes back the base register (set the W bit) if! is present.	

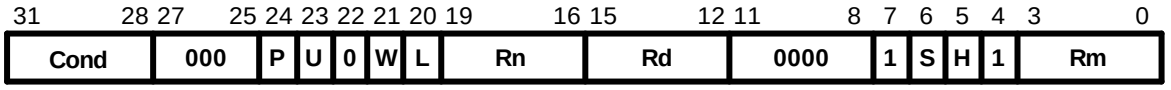
EXAMPLES

```
STR     R1,[R2,R4]!     ; Store R1 at R2+R4 (both of which are registers)
                        ; and write back address to R2.
STR     R1,[R2],R4      ; Store R1 at R2 and write back R2+R4 to R2.
LDR     R1,[R2,#16]     ; Load R1 from contents of R2+16, but don't write back.
LDR     R1,[R2,R3,LSL#2] ; Load R1 from contents of R2+R3*4.
LDREQB  R1,[R6,#5]      ; Conditionally load byte at R6+5 into
                        ; R1 bits 0 to 7, filling bits 8 to 31 with zeros.
STR     R1,PLACE        ; Generate PC relative offset to address PLACE.
PLACE
```

HALFWORD AND SIGNED DATA TRANSFER (LDRH/STRH/LDRSB/LDRSH)

The instruction is only executed if the condition is true. The various conditions are defined in Table 3-2. The instruction encoding is shown in Figure 3-16.

These instructions are used to load or store half-words of data and also load sign-extended bytes or half-words of data. The memory address used in the transfer is calculated by adding an offset to or subtracting an offset from a base register. The result of this calculation may be written back into the base register if auto-indexing is required.



[3:0] Offset Register

[6][5] S H
0 0 = SWP instruction
0 1 = Unsigned halfword
1 1 = Signed byte
1 1 = Signed halfword

[15:12] Source/Destination Register

[19:16] Base Register

[20] Load/Store
0 = Store to memory
1 = Load from memory

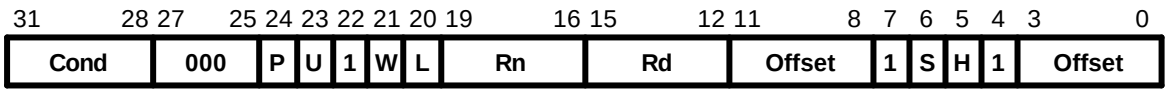
[21] Write-back
0 = No write-back
1 = Write address into base

[23] Up/Down
0 = Down: subtract offset from base
1 = Up: add offset to base

[24] Pre/Post Indexing
0 = Post: add/subtract offset after transfer
1 = Pre: add/subtract offset bofore transfer

[31:28] Condition Field

Figure 3-16. Halfword and Signed Data Transfer with Register Offset



[3:0] Immediate Offset (Low Nibble)

[6][5] S H
0 0 = SWP instruction
0 1 = Unsigned halfword
1 1 = Signed byte
1 1 = Signed halfword

[11:8] Immediate Offset (High Nibble)

[15:12] Source/Destination Register

[19:16] Base Register

[20] Load/Store
0 = Store to memory
1 = Load from memory

[21] Write-back
0 = No write-back
1 = Write address into base

[23] Up/Down
0 = Down: subtract offset from base
1 = Up: add offset to base

[24] Pre/Post Indexing
0 = Post: add/subtract offset after transfer
1 = Pre: add/subtract offset bofore transfer

[31:28] Condition Field

Figure 3-17. Halfword and Signed Data Transfer with Immediate Offset and Auto-Indexing

OFFSETS AND AUTO-INDEXING

The offset from the base may be either a 8-bit unsigned binary immediate value in the instruction, or a second register. The 8-bit offset is formed by concatenating bits 11 to 8 and bits 3 to 0 of the instruction word, such that bit 11 becomes the MSB and bit 0 becomes the LSB. The offset may be added to (U = 1) or subtracted from (U = 0) the base register Rn. The offset modification may be performed either before (pre-indexed, P = 1) or after (post-indexed, P = 0) the base register is used as the transfer address.

The W bit gives optional auto-increment and decrement addressing modes. The modified base value may be written back into the base (W = 1), or the old base may be kept (W = 0). In the case of post-indexed addressing, the write back bit is redundant and is always set to zero, since the old base value can be retained if necessary by setting the offset to zero. Therefore post-indexed data transfers always write back the modified base.

The Write-back bit should not be set high (W = 1) when post-indexed addressing is selected.

HALFWORD LOAD AND STORES

Setting S=0 and H=1 may be used to transfer unsigned Half-words between an ARM920T register and memory.

The action of LDRH and STRH instructions is influenced by the BIGEND control signal. The two possible configurations are described in the section below.

Signed byte and halfword loads

The S bit controls the loading of sign-extended data. When S = 1 the H bit selects between Bytes (H=0) and Half-words (H = 1). The L bit should not be set low (Store) when Signed (S = 1) operations have been selected.

The LDRSB instruction loads the selected Byte into bits 7 to 0 of the destination register and bits 31 to 8 of the destination register are set to the value of bit 7, the sign bit.

The LDRSH instruction loads the selected Half-word into bits 15 to 0 of the destination register and bits 31 to 16 of the destination register are set to the value of bit 15, the sign bit.

The action of the LDRSB and LDRSH instructions is influenced by the BIGEND control signal. The two possible configurations are described in the following section.

Endianness and byte/halfword selection

Little-Endian Configuration

A signed byte load (LDRSB) expects data on data bus inputs 7 through to 0 if the supplied address is on a word boundary, on data bus inputs 15 through to 8 if it is a word address plus one byte, and so on. The selected byte is placed in the bottom 8 bit of the destination register, and the remaining bits of the register are filled with the sign bit, bit 7 of the byte. Please see Figure 2-2.

A halfword load (LDRSH or LDRH) expects data on data bus inputs 15 through to 0 if the supplied address is on a word boundary and on data bus inputs 31 through to 16 if it is a halfword boundary, (A[1] = 1).The supplied address should always be on a halfword boundary. If bit 0 of the supplied address is HIGH then the ARM920T will load an unpredictable value. The selected halfword is placed in the bottom 16 bits of the destination register. For unsigned half-words (LDRH), the top 16 bits of the register are filled with zeros and for signed half-words (LDRSH) the top 16 bits are filled with the sign bit, bit 15 of the halfword.

A halfword store (STRH) repeats the bottom 16 bits of the source register twice across the data bus outputs 31 through to 0. The external memory system should activate the appropriate halfword subsystem to store the data. Note that the address must be halfword aligned, if bit 0 of the address is HIGH this will cause unpredictable behaviour.

Big-Endian Configuration

A signed byte load (LDRSB) expects data on data bus inputs 31 through to 24 if the supplied address is on a word boundary, on data bus inputs 23 through to 16 if it is a word address plus one byte, and so on. The selected byte is placed in the bottom 8 bit of the destination register, and the remaining bits of the register are filled with the sign bit, bit 7 of the byte. Please see Figure 2-1.

A halfword load (LDRSH or LDRH) expects data on data bus inputs 31 through to 16 if the supplied address is on a word boundary and on data bus inputs 15 through to 0 if it is a halfword boundary, (A[1]=1). The supplied address should always be on a halfword boundary. If bit 0 of the supplied address is HIGH then the ARM920T will load an unpredictable value. The selected halfword is placed in the bottom 16 bits of the destination register. For unsigned half-words (LDRH), the top 16 bits of the register are filled with zeros and for signed half-words (LDRSH) the top 16 bits are filled with the sign bit, bit 15 of the halfword.

A halfword store (STRH) repeats the bottom 16 bits of the source register twice across the data bus outputs 31 through to 0. The external memory system should activate the appropriate halfword subsystem to store the data. Note that the address must be halfword aligned, if bit 0 of the address is HIGH this will cause unpredictable behaviour.

USE OF R15

Write-back should not be specified if R15 is specified as the base register (Rn). When using R15 as the base register you must remember it contains an address 8 bytes on from the address of the current instruction.

R15 should not be specified as the register offset (Rm).

When R15 is the source register (Rd) of a Half-word store (STRH) instruction, the stored address will be address of the instruction plus 12.

DATA ABORTS

A transfer to or from a legal address may cause problems for a memory management system. For instance, in a system which uses virtual memory the required data may be absent from the main memory. The memory manager can signal a problem by taking the processor ABORT input HIGH whereupon the Data Abort trap will be taken. It is up to the system software to resolve the cause of the problem, then the instruction can be restarted and the original program continued.

INSTRUCTION CYCLE TIMES

Normal LDR(H,SH,SB) instructions take 1S + 1N + 1I. LDR(H,SH,SB) PC take 2S + 2N + 1I incremental cycles. S,N and I are defined as sequential (S-cycle), non-sequential (N-cycle), and internal (I-cycle), respectively. STRH instructions take 2N incremental cycles to execute.

ASSEMBLER SYNTAX

<LDR|STR>{cond}<H|SH|SB> Rd,<address>

LDR	Load from memory into a register
STR	Store from a register into memory
{cond}	Two-character condition mnemonic. See Table 3-2..
H	Transfer halfword quantity
SB	Load sign extended byte (Only valid for LDR)
SH	Load sign extended halfword (Only valid for LDR)
Rd	An expression evaluating to a valid register number.

<address> can be:

1	An expression which generates an address: The assembler will attempt to generate an instruction using the PC as a base and a corrected immediate offset to address the location given by evaluating the expression. This will be a PC relative, pre-indexed address. If the address is out of range, an error will be generated.	
2	A pre-indexed addressing specification:	
	[Rn]	offset of zero
	[Rn,<#expression>]{!}	offset of <expression> bytes
	[Rn,{+/-}Rm]{!}	offset of +/- contents of index register
3	A post-indexed addressing specification:	
	[Rn],<#expression>	offset of <expression> bytes
	[Rn],{+/-}Rm	offset of +/- contents of index register.
4	Rn and Rm are expressions evaluating to a register number. If Rn is R15 then the assembler will subtract 8 from the offset value to allow for ARM920T pipelining. In this case base write-back should not be specified.	
{!}	Writes back the base register (set the W bit) if ! is present.	

EXAMPLES

```
LDRH    R1,[R2,-R3]!        ;  Load R1 from the contents of the halfword address
                            ;  contained in R2-R3 (both of which are registers)
                            ;  and write back address to R2
STRH    R3,[R4,#14]         ;  Store the halfword in R3 at R14+14 but don't write back.
LDRSB   R8,[R2],#-223       ;  Load R8 with the sign extended contents of the byte
                            ;  address contained in R2 and write back R2-223 to R2.
LDRNESH R11,[R0]            ;  Conditionally load R11 with the sign extended contents
                            ;  of the halfword address contained in R0.
HERE                        ;  Generate PC relative offset to address FRED.
STRH    R5,[PC,#(FRED-HERE-8)] ;  Store the halfword in R5 at address FRED
FRED
```

BLOCK DATA TRANSFER (LDM, STM)

The instruction is only executed if the condition is true. The various conditions are defined in Table 3-2. The instruction encoding is shown in Figure 3-18.

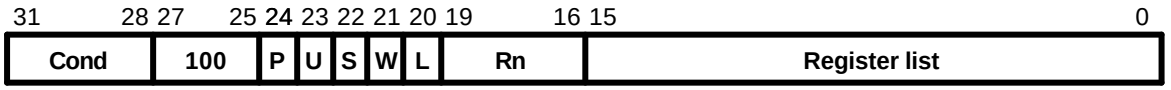
Block data transfer instructions are used to load (LDM) or store (STM) any subset of the currently visible registers. They support all possible stacking modes, maintaining full or empty stacks which can grow up or down memory, and are very efficient instructions for saving or restoring context, or for moving large blocks of data around main memory.

THE REGISTER LIST

The instruction can cause the transfer of any registers in the current bank (and non-user mode programs can also transfer to and from the user bank, see below). The register list is a 16-bit field in the instruction, with each bit corresponding to a register. A 1 in bit 0 of the register field will cause R0 to be transferred, a 0 will cause it not to be transferred; similarly bit 1 controls the transfer of R1, and so on.

Any subset of the registers, or all the registers, may be specified. The only restriction is that the register list should not be empty.

Whenever R15 is stored to memory the stored value is the address of the STM instruction plus 12.



[19:16] Base Register

[20] Load/Store Bit
0 = Store to memory
1 = Load from memory

[21] Write-back Bit
0 = No write-back
1 = Write address into base

[22] PSR & Force User Bit
0 = Do not load PSR or user mode
1 = Load PSR or force user mode

[23] Up/Down Bit
0 = Down: subtract offset from base
1 = Up: add offset to base

[24] Pre/Post Indexing Bit
0 = Post: add offset after transfer
1 = Pre: add offset bofore transfer

[31:28] Condition Field

Figure 3-18. Block Data Transfer Instructions

ADDRESSING MODES

The transfer addresses are determined by the contents of the base register (Rn), the pre/post bit (P) and the up/down bit (U). The registers are transferred in the order lowest to highest, so R15 (if in the list) will always be transferred last. The lowest register also gets transferred to/from the lowest memory address. By way of illustration, consider the transfer of R1, R5 and R7 in the case where Rn = 0x1000 and write back of the modified base is required (W = 1). Figure 3.19-22 show the sequence of register transfers, the addresses used, and the value of Rn after the instruction has completed.

In all cases, had write back of the modified base not been required (W = 0), Rn would have retained its initial value of 0x1000 unless it was also in the transfer list of a load multiple register instruction, when it would have been overwritten with the loaded value.

ADDRESS ALIGNMENT

The address should normally be a word aligned quantity and non-word aligned addresses do not affect the instruction. However, the bottom 2 bits of the address will appear on A[1:0] and might be interpreted by the memory system.

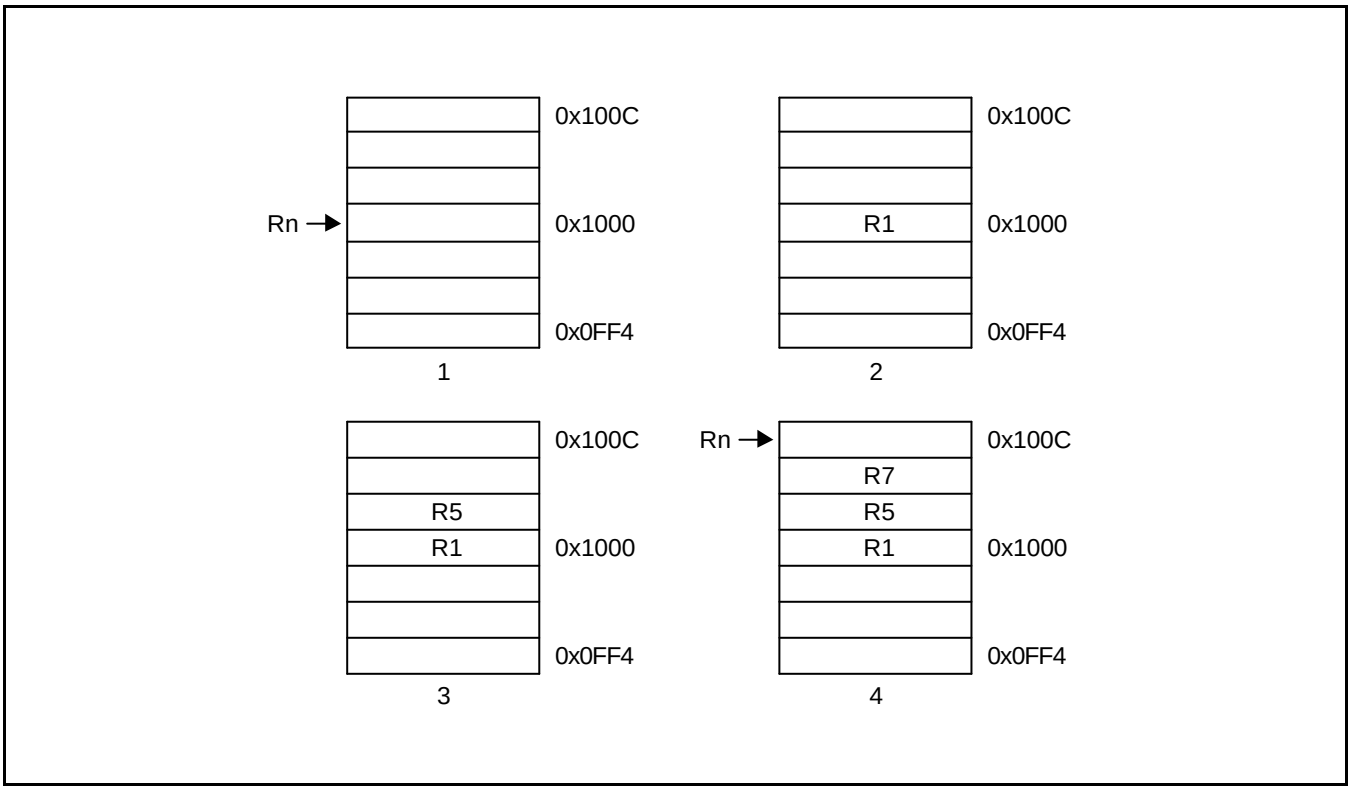


Figure 3-19. Post-Increment Addressing

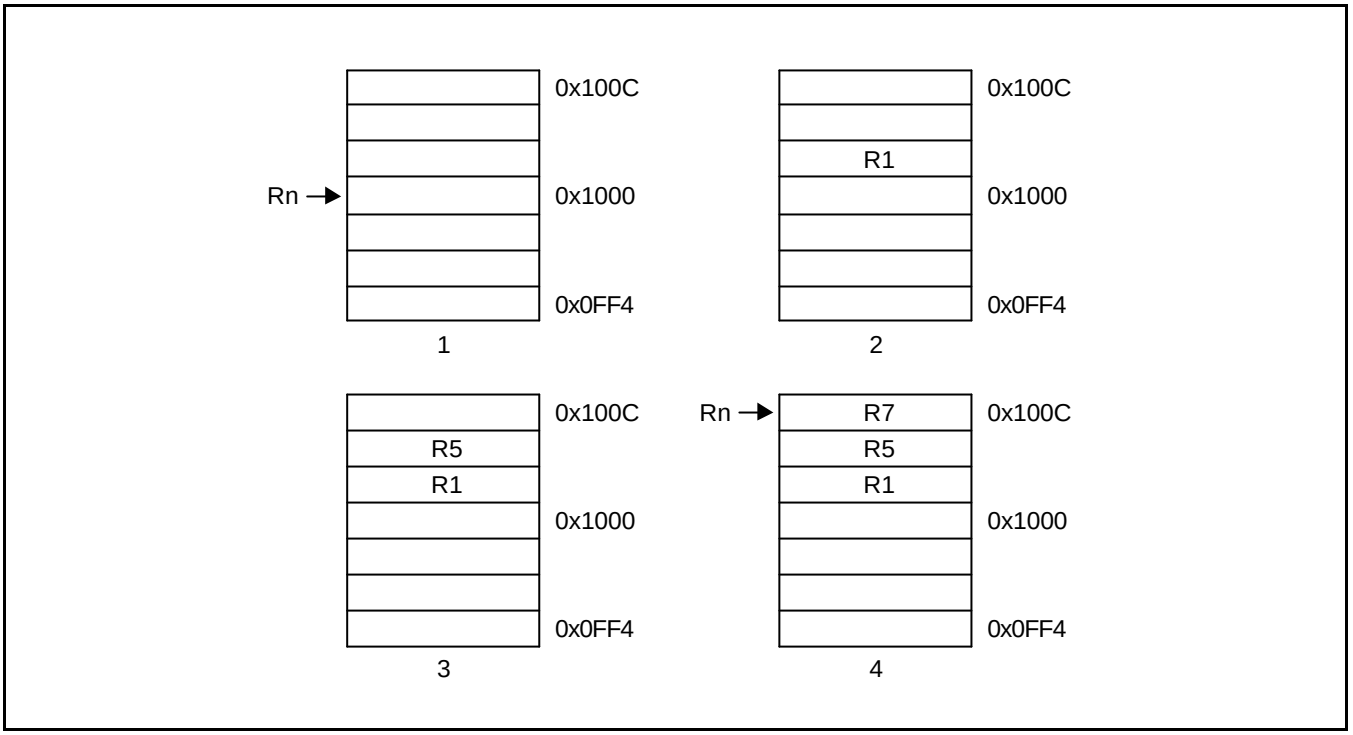


Figure 3-20. Pre-Increment Addressing

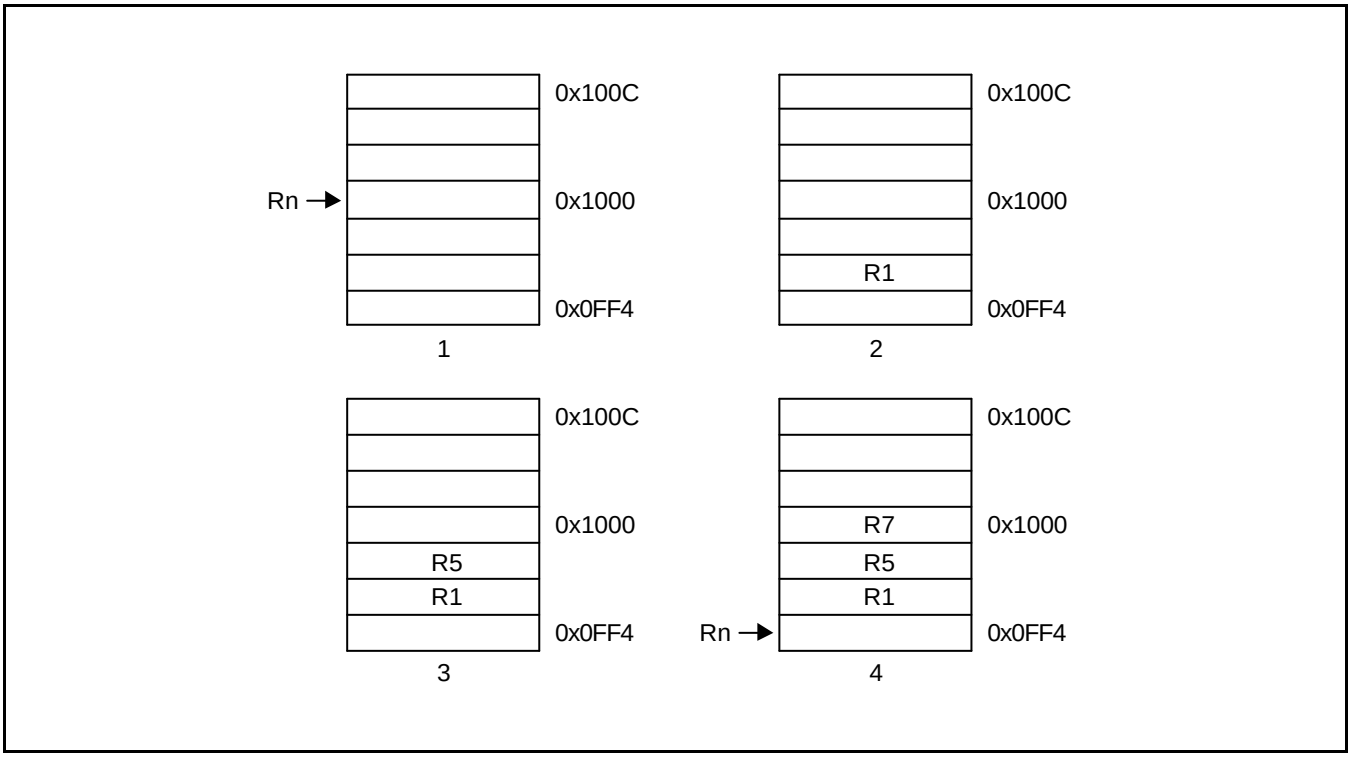


Figure 3-21. Post-Decrement Addressing

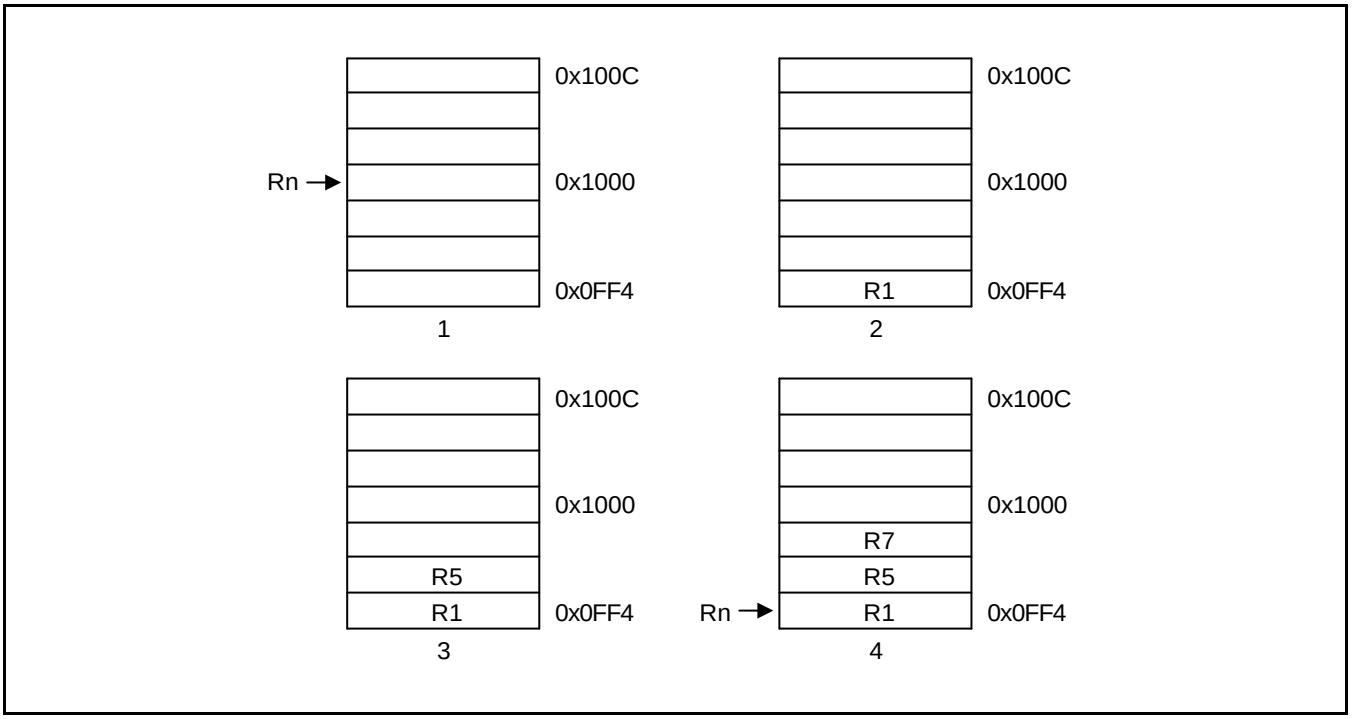


Figure 3-22. Pre-Decrement Addressing

USE OF THE S BIT

When the S bit is set in a LDM/STM instruction its meaning depends on whether or not R15 is in the transfer list and on the type of instruction. The S bit should only be set if the instruction is to execute in a privileged mode.

LDM with R15 in Transfer List and S Bit Set (Mode Changes)

If the instruction is a LDM then SPSR_<mode> is transferred to CPSR at the same time as R15 is loaded.

STM with R15 in Transfer List and S Bit Set (User Bank Transfer)

The registers transferred are taken from the User bank rather than the bank corresponding to the current mode. This is useful for saving the user state on process switches. Base write-back should not be used when this mechanism is employed.

R15 not in List and S Bit Set (User Bank Transfer)

For both LDM and STM instructions, the User bank registers are transferred rather than the register bank corresponding to the current mode. This is useful for saving the user state on process switches. Base write-back should not be used when this mechanism is employed.

When the instruction is LDM, care must be taken not to read from a banked register during the following cycle (inserting a dummy instruction such as MOV R0, R0 after the LDM will ensure safety).

USE OF R15 AS THE BASE

R15 should not be used as the base register in any LDM or STM instruction.

INCLUSION OF THE BASE IN THE REGISTER LIST

When write-back is specified, the base is written back at the end of the second cycle of the instruction. During a STM, the first register is written out at the start of the second cycle. A STM which includes storing the base, with the base as the first register to be stored, will therefore store the unchanged value, whereas with the base second or later in the transfer order, will store the modified value. A LDM will always overwrite the updated base if the base is in the list.

DATA ABORTS

Some legal addresses may be unacceptable to a memory management system, and the memory manager can indicate a problem with an address by taking the ABORT signal HIGH. This can happen on any transfer during a multiple register load or store, and must be recoverable if ARM920T is to be used in a virtual memory system.

Abort during STM Instructions

If the abort occurs during a store multiple instruction, ARM920T takes little action until the instruction completes, whereupon it enters the data abort trap. The memory manager is responsible for preventing erroneous writes to the memory. The only change to the internal state of the processor will be the modification of the base register if write-back was specified, and this must be reversed by software (and the cause of the abort resolved) before the instruction may be retried.

Aborts during LDM Instructions

When ARM920T detects a data abort during a load multiple instruction, it modifies the operation of the instruction to ensure that recovery is possible.

- Overwriting of registers stops when the abort happens. The aborting load will not take place but earlier ones may have overwritten registers. The PC is always the last register to be written and so will always be preserved.
- The base register is restored, to its modified value if write-back was requested. This ensures recoverability in the case where the base register is also in the transfer list, and may have been overwritten before the abort occurred.

The data abort trap is taken when the load multiple has completed, and the system software must undo any base modification (and resolve the cause of the abort) before restarting the instruction.

INSTRUCTION CYCLE TIMES

Normal LDM instructions take nS + 1N + 1I and LDM PC takes (n+1)S + 2N + 1I incremental cycles, where S,N and I are defined as sequential (S-cycle), non-sequential (N-cycle), and internal (I-cycle), respectively. STM instructions take (n-1)S + 2N incremental cycles to execute, where *n* is the number of words transferred.

ASSEMBLER SYNTAX

<LDM|STM>{cond}<FD|ED|FA|EA|IA|IB|DA|DB> Rn{!},<Rlist>{^}

where:

{cond}	Two character condition mnemonic. See Table 3-2.
Rn	An expression evaluating to a valid register number
<Rlist>	A list of registers and register ranges enclosed in {} (e.g. {R0,R2-R7,R10}).
{!}	If present requests write-back (W = 1), otherwise W = 0.
{^}	If present set S bit to load the CPSR along with the PC, or force transfer of user bank when in privileged mode.

Addressing Mode Names

There are different assembler mnemonics for each of the addressing modes, depending on whether the instruction is being used to support stacks or for other purposes. The equivalence between the names and the values of the bits in the instruction are shown in the following table 3-6.

Table 3-6. Addressing Mode Names

Name	Stack	Other	L bit	P bit	U bit
Pre-Increment Load	LDMED	LDMIB	1	1	1
Post-Increment Load	LDMFD	LDMIA	1	0	1
Pre-Decrement Load	LDMEA	LDMDB	1	1	0
Post-Decrement Load	LDMFA	LDMDA	1	0	0
Pre-Increment Store	STMFA	STMIB	0	1	1
Post-Increment Store	STMEA	STMIA	0	0	1
Pre-Decrement Store	STMFD	STMDB	0	1	0
Post-Decrement Store	STMED	STMDA	0	0	0

FD, ED, FA, EA define pre/post indexing and the up/down bit by reference to the form of stack required. The F and E refer to a "full" or "empty" stack, i.e. whether a pre-index has to be done (full) before storing to the stack. The A and D refer to whether the stack is ascending or descending. If ascending, a STM will go up and LDM down, if descending, vice-versa.

IA, IB, DA, DB allow control when LDM/STM are not being used for stacks and simply mean Increment After, Increment Before, Decrement After, Decrement Before.

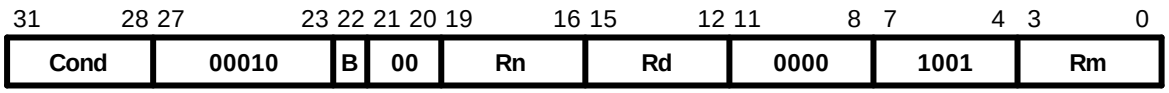
EXAMPLES

```
LDMFD   SP!,{R0,R1,R2}    ;  Unstack 3 registers.
STMIA   R0,{R0-R15}       ;  Save all registers.
LDMFD   SP!,{R15}         ;  R15 ← (SP), CPSR unchanged.
LDMFD   SP!,{R15}^        ;  R15 ← (SP), CPSR <- SPSR_mode
                          ;  (allowed only in privileged modes).
STMFD   R13,{R0-R14}^     ;  Save user mode regs on stack
                          ;  (allowed only in privileged modes).
```

These instructions may be used to save state on subroutine entry, and restore it efficiently on return to the calling routine:

```
STMED   SP!,{R0-R3,R14}   ;  Save R0 to R3 to use as workspace
                          ;  and R14 for returning.
BL      somewhere         ;  This nested call will overwrite R14
LDMED   SP!,{R0-R3,R15}   ;  Restore workspace and return.
```

SINGLE DATA SWAP (SWP)



[3:0] Source Register

[15:12] Destination Register

[19:16] Base Register

[22] Byte/Word Bit
0 = Swap word quantity
1 = Swap word quantity

[31:28] Condition Field

Figure 3-23. Swap Instruction

The instruction is only executed if the condition is true. The various conditions are defined in Table 3-2. The instruction encoding is shown in Figure 3-23.

The data swap instruction is used to swap a byte or word quantity between a register and external memory. This instruction is implemented as a memory read followed by a memory write which are "locked" together (the processor cannot be interrupted until both operations have completed, and the memory manager is warned to treat them as inseparable). This class of instruction is particularly useful for implementing software semaphores.

The swap address is determined by the contents of the base register (Rn). The processor first reads the contents of the swap address. Then it writes the contents of the source register (Rm) to the swap address, and stores the old memory contents in the destination register (Rd). The same register may be specified as both the source and destination.

The **LOCK** output goes HIGH for the duration of the read and write operations to signal to the external memory manager that they are locked together, and should be allowed to complete without interruption. This is important in multi-processor systems where the swap instruction is the only indivisible instruction which may be used to implement semaphores; control of the memory must not be removed from a processor while it is performing a locked operation.

BYTES AND WORDS

This instruction class may be used to swap a byte (B = 1) or a word (B = 0) between an ARM920T register and memory. The SWP instruction is implemented as a LDR followed by a STR and the action of these is as described in the section on single data transfers. In particular, the description of Big and Little Endian configuration applies to the SWP instruction.

USE OF R15

Do not use R15 as an operand (Rd, Rn or Rs) in a SWP instruction.

DATA ABORTS

If the address used for the swap is unacceptable to a memory management system, the memory manager can flag the problem by driving ABORT HIGH. This can happen on either the read or the write cycle (or both), and in either case, the Data Abort trap will be taken. It is up to the system software to resolve the cause of the problem, then the instruction can be restarted and the original program continued.

INSTRUCTION CYCLE TIMES

Swap instructions take 1S + 2N +1I incremental cycles to execute, where S,N and I are defined as sequential (S-cycle), non-sequential, and internal (I-cycle), respectively.

ASSEMBLER SYNTAX

<SWP>{cond}{B} Rd,Rm,[Rn]

{cond}	Two-character condition mnemonic. See Table 3-2.
{B}	If B is present then byte transfer, otherwise word transfer
Rd,Rm,Rn	Expressions evaluating to valid register numbers

Examples

```
SWP     R0,R1,[R2]    ; Load R0 with the word addressed by R2, and
                      ; store R1 at R2.
SWPB    R2,R3,[R4]    ; Load R2 with the byte addressed by R4, and
                      ; store bits 0 to 7 of R3 at R4.
SWPEQ   R0,R0,[R1]    ; Conditionally swap the contents of the
                      ; word addressed by R1 with R0.
```

SOFTWARE INTERRUPT (SWI)

The instruction is only executed if the condition is true. The various conditions are defined in Table 3-2. The instruction encoding is shown in Figure 3-24, below.

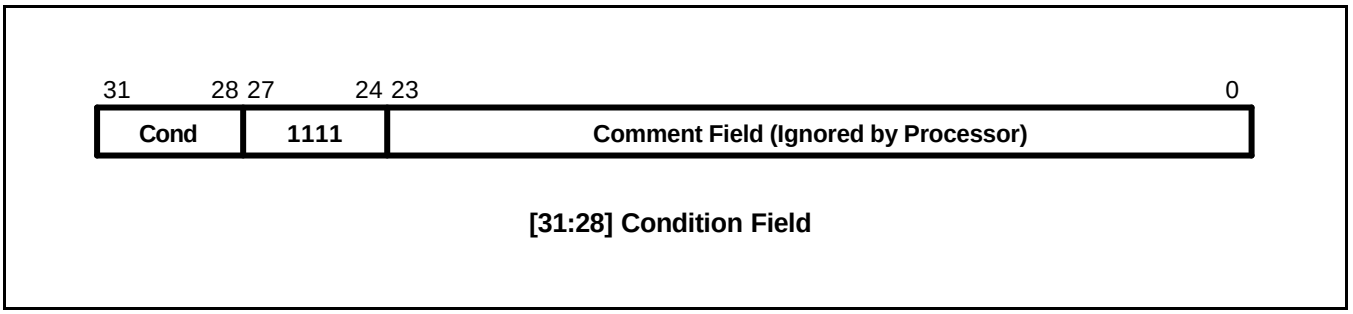


Figure 3-24. Software Interrupt Instruction

The software interrupt instruction is used to enter Supervisor mode in a controlled manner. The instruction causes the software interrupt trap to be taken, which effects the mode change. The PC is then forced to a fixed value (0x08) and the CPSR is saved in SPSR_svc. If the SWI vector address is suitably protected (by external memory management hardware) from modification by the user, a fully protected operating system may be constructed.

RETURN FROM THE SUPERVISOR

The PC is saved in R14_svc upon entering the software interrupt trap, with the PC adjusted to point to the word after the SWI instruction. MOVS PC,R14_svc will return to the calling program and restore the CPSR.

Note that the link mechanism is not re-entrant, so if the supervisor code wishes to use software interrupts within itself it must first save a copy of the return address and SPSR.

COMMENT FIELD

The bottom 24 bits of the instruction are ignored by the processor, and may be used to communicate information to the supervisor code. For instance, the supervisor may look at this field and use it to index into an array of entry points for routines which perform the various supervisor functions.

INSTRUCTION CYCLE TIMES

Software interrupt instructions take 2S + 1N incremental cycles to execute, where S and N are defined as sequential (S-cycle) and non-sequential (N-cycle).

ASSEMBLER SYNTAX

SWI{cond} <expression>

{cond}	Two character condition mnemonic, Table 3-2.
<expression>	Evaluated and placed in the comment field (which is ignored by ARM920T).

Examples

```
        SWI     ReadC           ;  Get next character from read stream.
        SWI     WriteI+"k”      ;  Output a "k" to the write stream.
        SWINE   0               ;  Conditionally call supervisor with 0 in comment field.
```

Supervisor code

The previous examples assume that suitable supervisor code exists, for instance:

```
        0x08 B Supervisor               ;  SWI entry point
        EntryTable                      ;  Addresses of supervisor routines
        DCD ZeroRtn
        DCD ReadCRtn
        DCD WriteIRtn
        • • •
        Zero        EQU  0
ReadC   EQU  256
WriteI  EQU  512

        Supervisor                      ;  SWI has routine required in bits 8-23 and data (if any) in
                                        ;  bits 0-7. Assumes R13_svc points to a suitable stack
        STMFD       R13,{R0-R2,R14}     ;  Save work registers and return address.
        LDR         R0,[R14,#-4]        ;  Get SWI instruction.
        BIC         R0,R0,#0xFF000000   ;  Clear top 8 bits.
        MOV         R1,R0,LSR#8         ;  Get routine offset.
        ADR         R2,EntryTable       ;  Get start address of entry table.
        LDR         R15,[R2,R1,LSL#2]   ;  Branch to appropriate routine.
        WriteIRtn                       ;  Enter with character in R0 bits 0-7.
        • • •
        LDMFD       R13,{R0-R2,R15}^    ;  Restore workspace and return,
                                        ;  restoring processor mode and flags.
```

COPROCESSOR DATA OPERATIONS (CDP)

The instruction is only executed if the condition is true. The various conditions are defined in Table 3-2. The instruction encoding is shown in Figure 3-25.

This class of instruction is used to tell a coprocessor to perform some internal operation. No result is communicated back to ARM920T, and it will not wait for the operation to complete. The coprocessor could contain a queue of such instructions awaiting execution, and their execution can overlap other activity, allowing the coprocessor and ARM920T to perform independent tasks in parallel.

COPROCESSOR INSTRUCTIONS

The S3C2410A, unlike some other ARM-based processors, does not have an external coprocessor interface. It does not have a on-chip coprocessor also.

So then all coprocessor instructions will cause the undefined instruction trap to be taken on the S3C2410A. These coprocessor instructions can be emulated by the undefined trap handler. Even though external coprocessor can not be connected to the S3C2410A, the coprocessor instructions are still described here in full for completeness. (Remember that any external coprocessor described in this section is a software emulation.)

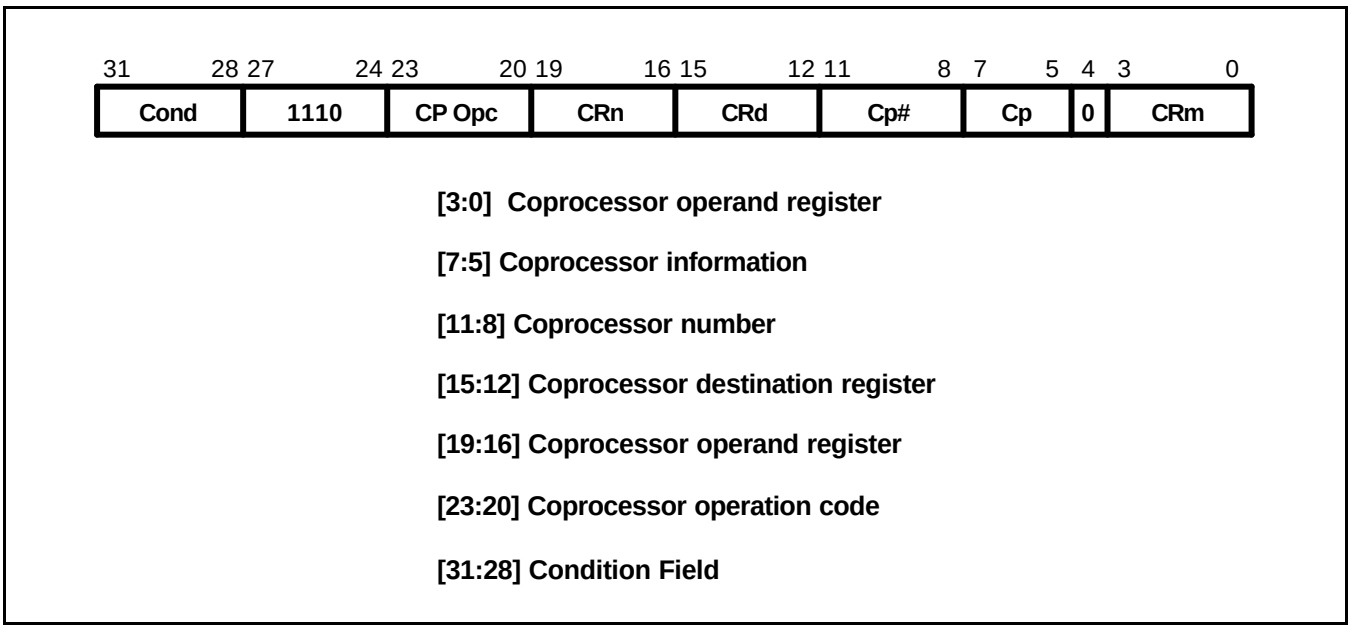


Figure 3-25. Coprocessor Data Operation Instruction

Only bit 4 and bits 24 to 31 The coprocessor fields are significant to ARM920T. The remaining bits are used by coprocessors. The above field names are used by convention, and particular coprocessors may redefine the use of all fields except CP# as appropriate. The CP# field is used to contain an identifying number (in the range 0 to 15) for each coprocessor, and a coprocessor will ignore any instruction which does not contain its number in the CP# field.

The conventional interpretation of the instruction is that the coprocessor should perform an operation specified in the CP Opc field (and possibly in the CP field) on the contents of CRn and CRm, and place the result in CRd.

INSTRUCTION CYCLE TIMES

Coprocessor data operations take 1S + bI incremental cycles to execute, where *b* is the number of cycles spent in the coprocessor busy-wait loop.

S and I are defined as sequential (S-cycle) and internal (I-cycle).

Assembler syntax

CDP{cond} p#,<expression1>,cd,cn,cm{,<expression2>}

{cond}	Two character condition mnemonic. See Table 3-2.
p#	The unique number of the required coprocessor
<expression1>	Evaluated to a constant and placed in the CP Opc field
cd, cn and cm	Evaluate to the valid coprocessor register numbers CRd, CRn and CRm respectively
<expression2>	Where present is evaluated to a constant and placed in the CP field

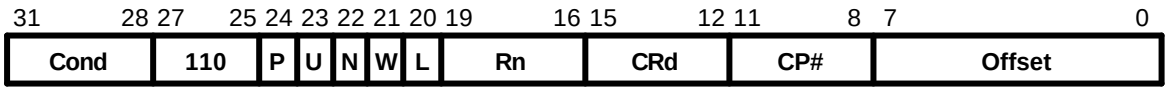
EXAMPLES

```
CDP     p1,10,c1,c2,c3      ;  Request coproc 1 to do operation 10
                            ;  on CR2 and CR3, and put the result in CR1.
CDPEQ   p2,5,c1,c2,c3,2     ;  If Z flag is set request coproc 2 to do operation 5 (type 2)
                            ;  on CR2 and CR3, and put the result in CR1.
```

COPROCESSOR DATA TRANSFERS (LDC, STC)

The instruction is only executed if the condition is true. The various conditions are defined in Table 3-2. The instruction encoding is shown in Figure 3-26.

This class of instruction is used to load (LDC) or store (STC) a subset of a coprocessors's registers directly to memory. ARM920T is responsible for supplying the memory address, and the coprocessor supplies or accepts the data and controls the number of words transferred.



[7:0] Unsigned 8 Bit Immediate Offset

[11:8] Coprocessor Number

[15:12] Coprocessor Source/Destination Register

[19:16] Base Register

[20] Load/Store Bit
0 = Store to memory
1 = Load from memory

[21] Write-back Bit
0 = No write-back
1 = Write address into base

[22] Transfer Length

[23] Up/Down Bit
0 = Down: subtract offset from base
1 = Up: add offset to base

[24] Pre/Post Indexing Bit
0 = Post: add offset after transfer
1 = Pre: add offset before transfer

[31:28] Condition Field

Figure 3-26. Coprocessor Data Transfer Instructions

THE COPROCESSOR FIELDS

The CP# field is used to identify the coprocessor which is required to supply or accept the data, and a coprocessor will only respond if its number matches the contents of this field.

The CRd field and the N bit contain information for the coprocessor which may be interpreted in different ways by different coprocessors, but by convention CRd is the register to be transferred (or the first register where more than one is to be transferred), and the N bit is used to choose one of two transfer length options. For instance N=0 could select the transfer of a single register, and N = 1 could select the transfer of all the registers for context switching.

ADDRESSING MODES

ARM920T is responsible for providing the address used by the memory system for the transfer, and the addressing modes available are a subset of those used in single data transfer instructions. Note, however, that the immediate offsets are 8 bits wide and specify word offsets for coprocessor data transfers, whereas they are 12 bits wide and specify byte offsets for single data transfers.

The 8-bit unsigned immediate offset is shifted left 2 bits and either added to (U = 1) or subtracted from (U = 0) the base register (Rn); this calculation may be performed either before (P = 1) or after (P = 0) the base is used as the transfer address. The modified base value may be overwritten back into the base register (if W = 1), or the old value of the base may be preserved (W = 0). Note that post-indexed addressing modes require explicit setting of the W bit, unlike LDR and STR which always write-back when post-indexed.

The value of the base register, modified by the offset in a pre-indexed instruction, is used as the address for the transfer of the first word. The second word (if more than one is transferred) will go to or come from an address one word (4 bytes) higher than the first transfer, and the address will be incremented by one word for each subsequent transfer.

ADDRESS ALIGNMENT

The base address should normally be a word aligned quantity. The bottom 2 bits of the address will appear on **A[1:0]** and might be interpreted by the memory system.

Use of R15

If Rn is R15, the value used will be the address of the instruction plus 8 bytes. Base write-back to R15 must not be specified.

DATA ABORTS

If the address is legal but the memory manager generates an abort, the data trap will be taken. The write-back of the modified base will take place, but all other processor state will be preserved. The coprocessor is partly responsible for ensuring that the data transfer can be restarted after the cause of the abort has been resolved, and must ensure that any subsequent actions it undertakes can be repeated when the instruction is retried.

Instruction cycle times

Coprocessor data transfer instructions take (n-1)S + 2N + bI incremental cycles to execute, where:

n	The number of words transferred.
b	The number of cycles spent in the coprocessor busy-wait loop.

S, N and I are defined as sequential (S-cycle), non-sequential (N-cycle), and internal (I-cycle), respectively.

ASSEMBLER SYNTAX

<LDC|STC>{cond}{L} p#,cd,<Address>

LDC	Load from memory to coprocessor
STC	Store from coprocessor to memory
{L}	When present perform long transfer (N = 1), otherwise perform short transfer (N = 0)
{cond}	Two character condition mnemonic. See Table 3-2..
p#	The unique number of the required coprocessor
cd	An expression evaluating to a valid coprocessor register number that is placed in the CRd field
<Address>	can be:
1	An expression which generates an address: The assembler will attempt to generate an instruction using the PC as a base and a corrected immediate offset to address the location given by evaluating the expression. This will be a PC relative, pre-indexed address. If the address is out of range, an error will be generated
2	A pre-indexed addressing specification: [Rn] offset of zero [Rn,<#expression>]{!} offset of <expression> bytes
3	A post-indexed addressing specification: [Rn],<#expression offset of <expression> bytes {!} write back the base register (set the W bit) if! is present Rn is an expression evaluating to a valid ARM920T register number.

NOTE

If Rn is R15, the assembler will subtract 8 from the offset value to allow for ARM920T pipelining.

EXAMPLES

```
        LDC     p1,c2,table     ;  Load c2 of coproc 1 from address
                                ;  table, using a PC relative address.
        STCEQL  p2,c3,[R5,#24]! ;  Conditionally store c3 of coproc 2
                                ;  into an address 24 bytes up from R5,
                                ;  write this address back to R5, and use
                                ;  long transfer option (probably to store multiple words).
```

NOTE

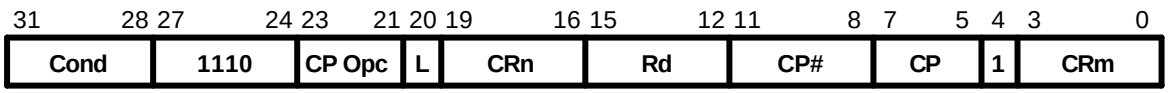
Although the address offset is expressed in bytes, the instruction offset field is in words. The assembler will adjust the offset appropriately.

COPROCESSOR REGISTER TRANSFERS (MRC, MCR)

The instruction is only executed if the condition is true. The various conditions are defined in Table 3-2.. The instruction encoding is shown in Figure 3-27.

This class of instruction is used to communicate information directly between ARM920T and a coprocessor. An example of a coprocessor to ARM920T register transfer (MRC) instruction would be a FIX of a floating point value held in a coprocessor, where the floating point number is converted into a 32-bit integer within the coprocessor, and the result is then transferred to ARM920T register. A FLOAT of a 32 bit value in ARM920T register into a floating point value within the coprocessor illustrates the use of ARM920T register to coprocessor transfer (MCR).

An important use of this instruction is to communicate control information directly from the coprocessor into the ARM920T CPSR flags. As an example, the result of a comparison of two floating point values within a coprocessor can be moved to the CPSR to control the subsequent flow of execution.



[3:0] Coprocessor Operand Register

[7:5] Coprocessor Information

[11:8] Coprocessor Number

[15:12] ARM Source/Destination Register

[19:16] Coprocessor Source/Destination Register

[20] Load/Store Bit
0 = Store to coprocessor
1 = Load from coprocessor

[21] Coprocessor Operation Mode

[31:28] Condition Field

Figure 3-27. Coprocessor Register Transfer Instructions

THE COPROCESSOR FIELDS

The CP# field is used, as for all coprocessor instructions, to specify which coprocessor is being called upon.

The CP Opc, CRn, CP and CRm fields are used only by the coprocessor, and the interpretation presented here is derived from convention only. Other interpretations are allowed where the coprocessor functionality is incompatible with this one. The conventional interpretation is that the CP Opc and CP fields specify the operation the coprocessor is required to perform, CRn is the coprocessor register which is the source or destination of the transferred information, and CRm is a second coprocessor register which may be involved in some way which depends on the particular operation specified.

TRANSFERS TO R15

When a coprocessor register transfer to ARM920T has R15 as the destination, bits 31, 30, 29 and 28 of the transferred word are copied into the N, Z, C and V flags respectively. The other bits of the transferred word are ignored, and the PC and other CPSR bits are unaffected by the transfer.

TRANSFERS FROM R15

A coprocessor register transfer from ARM920T with R15 as the source register will store the PC+12.

INSTRUCTION CYCLE TIMES

MRC instructions take 1S + (b+1)I +1C incremental cycles to execute, where S, I and C are defined as sequential (S-cycle), internal (I-cycle), and coprocessor register transfer (C-cycle), respectively. MCR instructions take 1S + bI +1C incremental cycles to execute, where *b* is the number of cycles spent in the coprocessor busy-wait loop.

ASSEMBLER SYNTAX

<MCR|MRC>{cond} p#,<expression1>,Rd,cn,cm{,<expression2>}

MRC	Move from coprocessor to ARM920T register (L=1)
MCR	Move from ARM920T register to coprocessor (L=0)
{cond}	Two character condition mnemonic. See Table 3-2
p#	The unique number of the required coprocessor
<expression1>	Evaluated to a constant and placed in the CP Opc field
Rd	An expression evaluating to a valid ARM920T register number
cn and cm	Expressions evaluating to the valid coprocessor register numbers CRn and CRm respectively
<expression2>	Where present is evaluated to a constant and placed in the CP field

EXAMPLES

```
        MRC     p2,5,R3,c5,c6   ; Request coproc 2 to perform operation 5
                                ; on c5 and c6, and transfer the (single
                                ; 32-bit word) result back to R3.
        MCR     p6,0,R4,c5,c6   ; Request coproc 6 to perform operation 0
                                ; on R4 and place the result in c6.
        MRCEQ   p3,9,R3,c5,c6,2 ; Conditionally request coproc 3 to
                                ; perform operation 9 (type 2) on c5 and
                                ; c6, and transfer the result back to R3.
```

UNDEFINED INSTRUCTION

The instruction is only executed if the condition is true. The various conditions are defined in Table 3-2. The instruction format is shown in Figure 3-28.

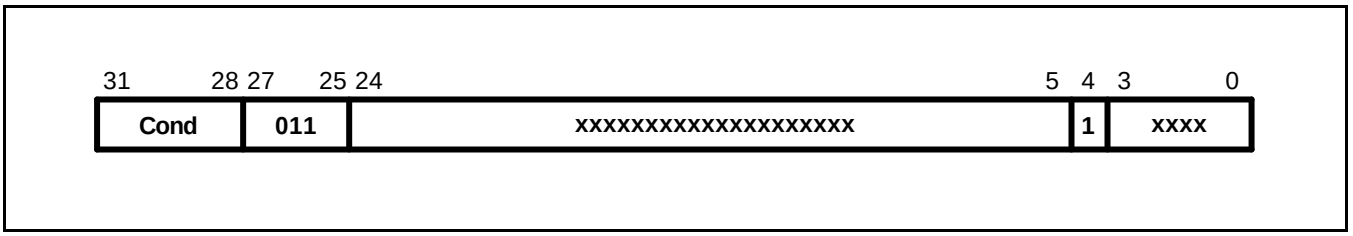


Figure 3-28. Undefined Instruction

If the condition is true, the undefined instruction trap will be taken.

Note that the undefined instruction mechanism involves offering this instruction to any coprocessors which may be present, and all coprocessors must refuse to accept it by driving **CPA** and **CPB** HIGH.

INSTRUCTION CYCLE TIMES

This instruction takes 2S + 1I + 1N cycles, where S, N and I are defined as sequential (S-cycle), non-sequential (N-cycle), and internal (I-cycle).

ASSEMBLER SYNTAX

The assembler has no mnemonics for generating this instruction. If it is adopted in the future for some specified use, suitable mnemonics will be added to the assembler. Until such time, this instruction must not be used.

INSTRUCTION SET EXAMPLES

The following examples show ways in which the basic ARM920T instructions can combine to give efficient code. None of these methods saves a great deal of execution time (although they may save some), mostly they just save code.

USING THE CONDITIONAL INSTRUCTIONS

Using Conditionals for Logical OR

```
CMP     Rn,#p           ;  If Rn=p OR Rm=q THEN GOTO Label.
BEQ     Label
CMP     Rm,#q
BEQ     Label
```

This can be replaced by

```
CMP     Rn,#p
CMPNE   Rm,#q           ;  If condition not satisfied try other test.
BEQ     Label
```

Absolute Value

```
TEQ     Rn,#0           ;  Test sign
RSBMI   Rn,Rn,#0        ;  and 2's complement if necessary.
```

Multiplication by 4, 5 or 6 (Run Time)

```
MOV     Rc,Ra,LSL#2     ;  Multiply by 4,
CMP     Rb,#5           ;  Test value,
ADDCS   Rc,Rc,Ra        ;  Complete multiply by 5,
ADDHI   Rc,Rc,Ra        ;  Complete multiply by 6.
```

Combining Discrete and Range Tests

```
TEQ     Rc,#127         ;  Discrete test,
CMPNE   Rc,# " "-1      ;  Range test
MOVLS   Rc,# ""         ;  IF Rc<= "" OR Rc=ASCII(127)
                        ;  THEN Rc:= "."
```

Division and Remainder

A number of divide routines for specific applications are provided in source form as part of the ANSI C library provided with the ARM Cross Development Toolkit, available from your supplier. A short general purpose divide routine follows.

```
                                        ;  Enter with numbers in Ra and Rb.
        MOV     Rcnt,#1                 ;  Bit to control the division.
Div1    CMP     Rb,#0x80000000          ;  Move Rb until greater than Ra.
        CMPCC   Rb,Ra
        MOVCC   Rb,Rb,ASL#1
        MOVCC   Rcnt,Rcnt,ASL#1
        BCC     Div1
        MOV     Rc,#0
Div2    CMP     Ra,Rb                   ;  Test for possible subtraction.
        SUBCS   Ra,Ra,Rb                ;  Subtract if ok,
        ADDCS   Rc,Rc,Rcnt              ;  Put relevant bit into result
        MOVS    Rcnt,Rcnt,LSR#1         ;  Shift control bit
        MOVNE   Rb,Rb,LSR#1             ;  Halve unless finished.
        BNE     Div2                    ;  Divide result in Rc, remainder in Ra.
```

Overflow Detection in the ARM920T

1. Overflow in unsigned multiply with a 32-bit result

```
        UMULL   Rd,Rt,Rm,Rn             ;  3 to 6 cycles
        TEQ     Rt,#0                   ;  +1 cycle and a register
        BNE     overflow
```

2. Overflow in signed multiply with a 32-bit result

```
        SMULL   Rd,Rt,Rm,Rn             ;  3 to 6 cycles
        TEQ     Rt,Rd ASR#31            ;  +1 cycle and a register
        BNE     overflow
```

3. Overflow in unsigned multiply accumulate with a 32-bit result

```
        UMLAL   Rd,Rt,Rm,Rn             ;  4 to 7 cycles
        TEQ     Rt,#0                   ;  +1 cycle and a register
        BNE     overflow
```

4. Overflow in signed multiply accumulate with a 32-bit result

```
        SMLAL   Rd,Rt,Rm,Rn             ;  4 to 7 cycles
        TEQ     Rt,Rd, ASR#31           ;  +1 cycle and a register
        BNE     overflow
```

5. Overflow in unsigned multiply accumulate with a 64-bit result

```
UMULL   Rl,Rh,Rm,Rn     ;  3 to 6 cycles
ADDS    Rl,Rl,Ra1       ;  Lower accumulate
ADC     Rh,Rh,Ra2       ;  Upper accumulate
BCS     overflow        ;  1 cycle and 2 registers
```

6. Overflow in signed multiply accumulate with a 64-bit result

```
SMULL   Rl,Rh,Rm,Rn     ;  3 to 6 cycles
ADDS    Rl,Rl,Ra1       ;  Lower accumulate
ADC     Rh,Rh,Ra2       ;  Upper accumulate
BVS     overflow        ;  1 cycle and 2 registers
```

NOTE

Overflow checking is not applicable to unsigned and signed multiplies with a 64-bit result, since overflow does not occur in such calculations.

PSEUDO-RANDOM BINARY SEQUENCE GENERATOR

It is often necessary to generate (pseudo-) random numbers and the most efficient algorithms are based on shift generators with exclusive-OR feedback rather like a cyclic redundancy check generator. Unfortunately the sequence of a 32-bit generator needs more than one feedback tap to be maximal length (i.e. 2^32-1 cycles before repetition), so this example uses a 33-bit register with taps at bits 33 and 20. The basic algorithm is newbit:=bit 33 eor bit 20, shift left the 33-bit number and put in newbit at the bottom; this operation is performed for all the newbits needed (i.e. 32 bits). The entire operation can be done in 5 S cycles:

```
                        ;  Enter with seed in Ra (32 bits),
                        ;  Rb (1 bit in Rb lsb), uses Rc.
TST     Rb,Rb,LSR#1     ;  Top bit into carry
MOVS    Rc,Ra,RRX       ;  33 bit rotate right
ADC     Rb,Rb,Rb        ;  Carry into lsb of Rb
EOR     Rc,Rc,Ra,LSL#12 ;  (involved!)
EOR     Ra,Rc,Rc,LSR#20 ;  (similarly involved!) new seed in Ra, Rb as before
```

MULTIPLICATION BY CONSTANT USING THE BARREL SHIFTER

Multiplication by 2^n (1,2,4,8,16,32..)

```
MOV     Ra, Rb, LSL #n
```

Multiplication by 2^n+1 (3,5,9,17..)

```
ADD     Ra,Ra,Ra,LSL #n
```

Multiplication by 2^n-1 (3,7,15..)

```
RSB     Ra,Ra,Ra,LSL #n
```

Multiplication by 6

```
ADD     Ra,Ra,Ra,LSL #1     ;  Multiply by 3
MOV     Ra,Ra,LSL#1         ;  and then by 2
```

Multiply by 10 and add in extra number

```
ADD     Ra,Ra,Ra,LSL#2      ;  Multiply by 5
ADD     Ra,Rc,Ra,LSL#1      ;  Multiply by 2 and add in next digit
```

General recursive method for Rb := Ra*C, C a constant:

1. If C even, say C = 2^n*D, D odd:

```
D=1:    MOV   Rb,Ra,LSL #n
D<>1:   {Rb := Ra*D}
MOV     Rb,Rb,LSL #n
```

2. If C MOD 4 = 1, say C = 2^n*D+1, D odd, n>1:

```
D=1:    ADD   Rb,Ra,Ra,LSL #n
D<>1:   {Rb := Ra*D}
ADD     Rb,Ra,Rb,LSL #n
```

3. If C MOD 4 = 3, say C = 2^n*D-1, D odd, n>1:

```
D=1:    RSB   Rb,Ra,Ra,LSL #n
D<>1:   {Rb := Ra*D}
RSB     Rb,Ra,Rb,LSL #n
```

This is not quite optimal, but close. An example of its non-optimality is multiply by 45 which is done by:

```
RSB     Rb,Ra,Ra,LSL#2      ;  Multiply by 3
RSB     Rb,Ra,Rb,LSL#2      ;  Multiply by 4*3-1 = 11
ADD     Rb,Ra,Rb,LSL# 2     ;  Multiply by 4*11+1 = 45
```

rather than by:

```
ADD     Rb,Ra,Ra,LSL#3      ;  Multiply by 9
ADD     Rb,Rb,Rb,LSL#2      ;  Multiply by 5*9 = 45
```

LOADING A WORD FROM AN UNKNOWN ALIGNMENT

```
                                ;  Enter with address in Ra (32 bits) uses
                                ;  Rb, Rc result in Rd. Note d must be less than c e.g. 0,1
BIC       Rb,Ra,#3              ;  Get word aligned address
LDMIA     Rb,{Rd,Rc}            ;  Get 64 bits containing answer
AND       Rb,Ra,#3              ;  Correction factor in bytes
MOVS      Rb,Rb,LSL#3           ;   ...now in bits and test if aligned
MOVNE     Rd,Rd,LSR Rb          ;  Produce bottom of result word (if not aligned)
RSBNE     Rb,Rb,#32             ;  Get other shift amount
ORRNE     Rd,Rd,Rc,LSL Rb       ;  Combine two halves to get result
```

NOTES

4 THUMB INSTRUCTION SET

THUMB INSTRUCTION SET FORMAT

The thumb instruction sets are 16-bit versions of ARM instruction sets (32-bit format). The ARM instructions are reduced to 16-bit versions, Thumb instructions, at the cost of versatile functions of the ARM instruction sets. The thumb instructions are decompressed to the ARM instructions by the Thumb decompressor inside the ARM920T core.

As the Thumb instructions are compressed ARM instructions, the Thumb instructions have the 16-bit format instructions and have some restrictions. The restrictions by 16-bit format is fully notified for using the Thumb instructions.

FORMAT SUMMARY

The THUMB instruction set formats are shown in the following figure.

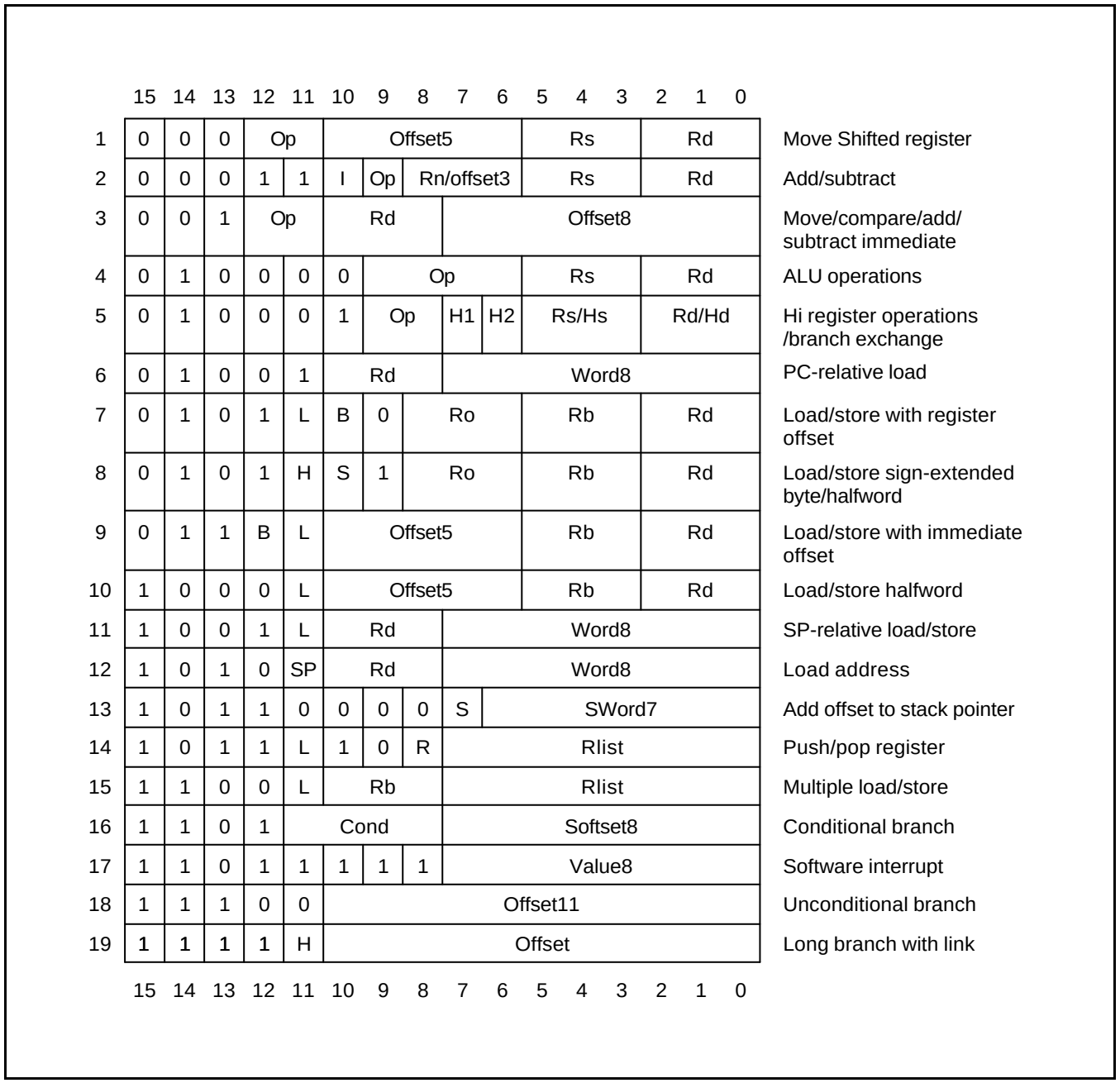


Figure 4-1. THUMB Instruction Set Formats

OPCODE SUMMARY

The following table summarizes the THUMB instruction set. For further information about a particular instruction please refer to the sections listed in the right-most column.

Table 4-1. THUMB Instruction Set Opcodes

Mnemonic	Instruction	Lo-Register Operand	Hi-Register Operand	Condition Codes Set
ADC	Add with Carry	Y	–	Y
ADD	Add	Y	–	Y (1)
AND	AND	Y	–	Y
ASR	Arithmetic Shift Right	Y	–	Y
B	Unconditional branch	Y	–	–
Bxx	Conditional branch	Y	–	–
BIC	Bit Clear	Y	–	Y
BL	Branch and Link	–	–	–
BX	Branch and Exchange	Y	Y	–
CMN	Compare Negative	Y	–	Y
CMP	Compare	Y	Y	Y
EOR	EOR	Y	–	Y
LDMIA	Load multiple	Y	–	–
LDR	Load word	Y	–	–
LDRB	Load byte	Y	–	–
LDRH	Load halfword	Y	–	–
LSL	Logical Shift Left	Y	–	Y
LDSB	Load sign-extended byte	Y	–	–
LDSH	Load sign-extended halfword	Y	–	–
LSR	Logical Shift Right	Y	–	Y
MOV	Move register	Y	Y	Y (2)
MUL	Multiply	Y	–	Y
MVN	Move Negative register	Y	–	Y

Table 4-1. THUMB Instruction Set Opcodes (Continued)

Mnemonic	Instruction	Lo-Register Operand	Hi-Register Operand	Condition Codes Set
NEG	Negate	Y	–	Y
ORR	OR	Y	–	Y
POP	Pop register	Y	–	–
PUSH	Push register	Y	–	–
ROR	Rotate Right	Y	–	Y
SBC	Subtract with Carry	Y	–	Y
STMIA	Store Multiple	Y	–	–
STR	Store word	Y	–	–
STRB	Store byte	Y	–	–
STRH	Store halfword	Y	–	–
SWI	Software Interrupt	–	–	–
SUB	Subtract	Y	–	Y
TST	Test bits	Y	–	Y

NOTES:

1. The condition codes are unaffected by the format 5, 12 and 13 versions of this instruction.
2. The condition codes are unaffected by the format 5 version of this instruction.

FORMAT 1: MOVE SHIFTED REGISTER

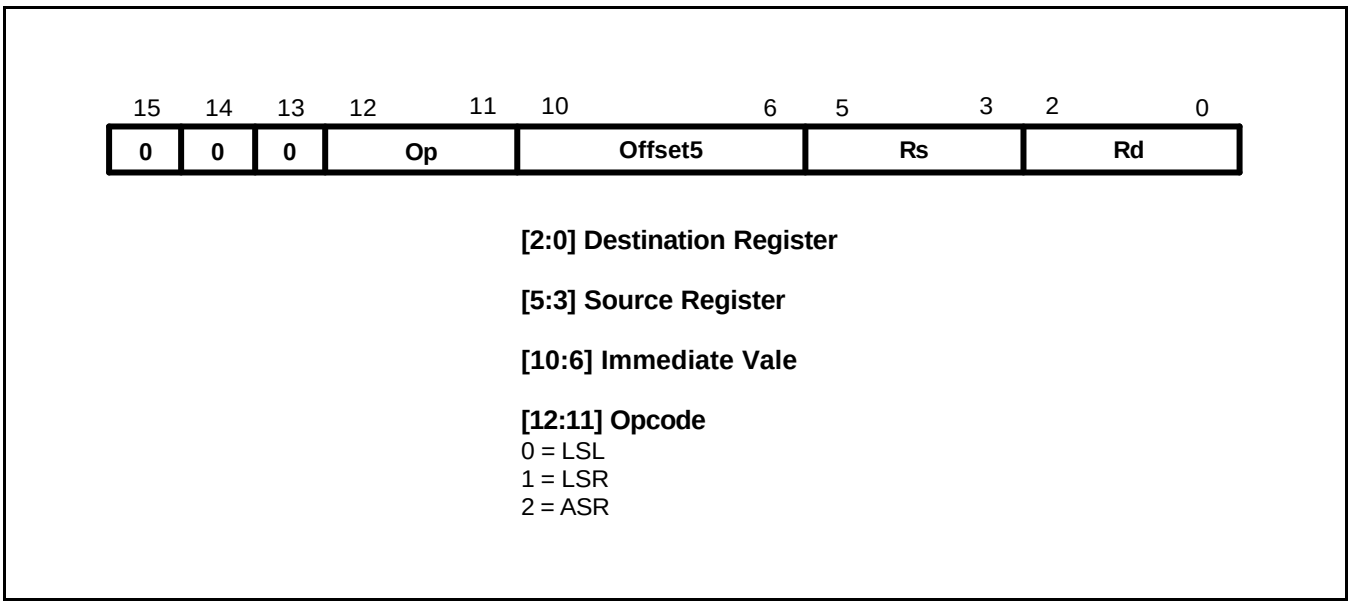


Figure 4-2. Format 1

OPERATION

These instructions move a shifted value between Lo registers. The THUMB assembler syntax is shown in Table 4-2.

NOTE

All instructions in this group set the CPSR condition codes.

Table 4-2. Summary of Format 1 Instructions

OP	THUMB Assembler	ARM Equipment	Action
00	LSL Rd, Rs, #Offset5	MOVS Rd, Rs, LSL #Offset5	Shift Rs left by a 5-bit immediate value and store the result in Rd.
01	LSR Rd, Rs, #Offset5	MOVS Rd, Rs, LSR #Offset5	Perform logical shift right on Rs by a 5-bit immediate value and store the result in Rd.
10	ASR Rd, Rs, #Offset5	MOVS Rd, Rs, ASR #Offset5	Perform arithmetic shift right on Rs by a 5-bit immediate value and store the result in Rd.

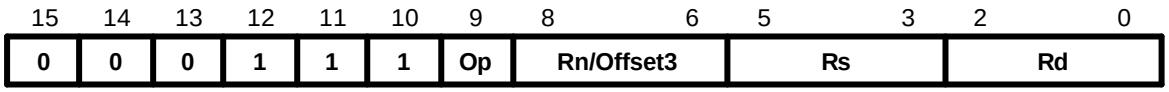
INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 4-2. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

```
LSR     R2, R5, #27     ; Logical shift right the contents
                        ; of R5 by 27 and store the result in R2.
                        ; Set condition codes on the result.
```

FORMAT 2: ADD/SUBTRACT



[2:0] Destination Register

[5:3] Source Register

[8:6] Register/Immediate Vale

[9] Opcode
0 = ADD
1 = SUB

[10] Immediate Flag
0 = Register operand
1 = Immediate oerand

Figure 4-3. Format 2

OPERATION

These instructions allow the contents of a Lo register or a 3-bit immediate value to be added to or subtracted from a Lo register. The THUMB assembler syntax is shown in Table 4-3.

NOTE

All instructions in this group set the CPSR condition codes.

Table 4-3. Summary of Format 2 Instructions

OP	I	THUMB Assembler	ARM Equipment	Action
0	0	ADD Rd, Rs, Rn	ADDS Rd, Rs, Rn	Add contents of Rn to contents of Rs. Place result in Rd.
0	1	ADD Rd, Rs, #Offset3	ADDS Rd, Rs, #Offset3	Add 3-bit immediate value to contents of Rs. Place result in Rd.
1	0	SUB Rd, Rs, Rn	SUBS Rd, Rs, Rn	Subtract contents of Rn from contents of Rs. Place result in Rd.
1	1	SUB Rd, Rs, #Offset3	SUBS Rd, Rs, #Offset3	Subtract 3-bit immediate value from contents of Rs. Place result in Rd.

INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 4-3. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

```
ADD     R0, R3, R4      ;  R0 := R3 + R4 and set condition codes on the result.
SUB     R6, R2, #6      ;  R6 := R2 - 6 and set condition codes.
```

FORMAT 3: MOVE/COMPARE/ADD/SUBTRACT IMMEDIATE

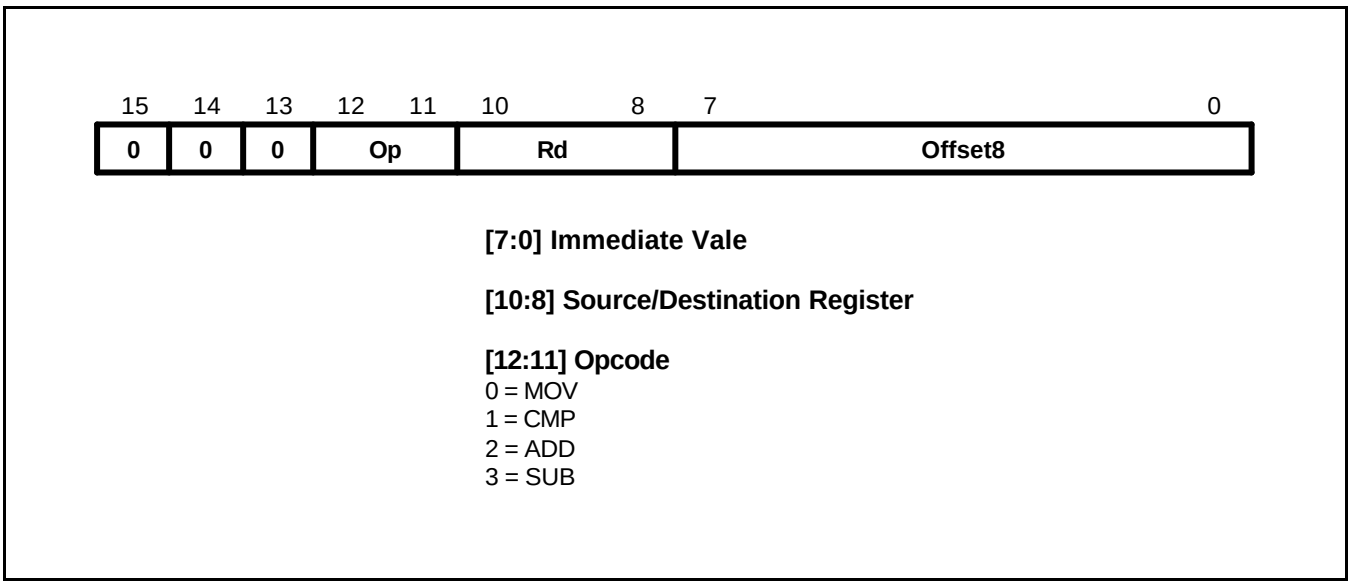


Figure 4-4. Format 3

OPERATIONS

The instructions in this group perform operations between a Lo register and an 8-bit immediate value. The THUMB assembler syntax is shown in Table 4-4.

NOTE

All instructions in this group set the CPSR condition codes.

Table 4-4. Summary of Format 3 Instructions

OP	THUMB Assembler	ARM Equipment	Action
00	MOV Rd, #Offset8	MOVS Rd, #Offset8	Move 8-bit immediate value into Rd.
01	CMP Rd, #Offset8	CMP Rd, #Offset8	Compare contents of Rd with 8-bit immediate value.
10	ADD Rd, #Offset8	ADDS Rd, Rd, #Offset8	Add 8-bit immediate value to contents of Rd and place the result in Rd.
11	SUB Rd, #Offset8	SUBS Rd, Rd, #Offset8	Subtract 8-bit immediate value from contents of Rd and place the result in Rd.

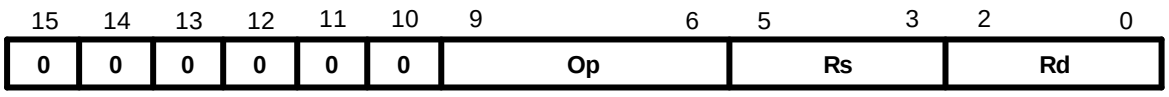
INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 4-4. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

```
MOV     R0, #128        ;   R0 := 128 and set condition codes
CMP     R2, #62         ;   Set condition codes on R2 - 62
ADD     R1, #255        ;   R1 := R1 + 255 and set condition codes
SUB     R6, #145        ;   R6 := R6 - 145 and set condition codes
```

FORMAT 4: ALU OPERATIONS



[2:0] Source/Destination Register

[5:3] Source Register 2

[9:6] Opcode

Figure 4-5. Format 4

OPERATION

The following instructions perform ALU operations on a Lo register pair.

NOTE

All instructions in this group set the CPSR condition codes.

Table 4-5. Summary of Format 4 Instructions

OP	THUMB Assembler	ARM Equipment	Action
0000	AND Rd, Rs	ANDS Rd, Rd, Rs	Rd:= Rd AND Rs
0001	EOR Rd, Rs	EORS Rd, Rd, Rs	Rd:= Rd EOR Rs
0010	LSL Rd, Rs	MOVS Rd, Rd, LSL Rs	Rd := Rd << Rs
0011	LSR Rd, Rs	MOVS Rd, Rd, LSR Rs	Rd := Rd >> Rs
0100	ASR Rd, Rs	MOVS Rd, Rd, ASR Rs	Rd := Rd ASR Rs
0101	ADC Rd, Rs	ADCS Rd, Rd, Rs	Rd := Rd + Rs + C-bit
0110	SBC Rd, Rs	SBCS Rd, Rd, Rs	Rd := Rd - Rs - NOT C-bit
0111	ROR Rd, Rs	MOVS Rd, Rd, ROR Rs	Rd := Rd ROR Rs
1000	TST Rd, Rs	TST Rd, Rs	Set condition codes on Rd AND Rs
1001	NEG Rd, Rs	RSBS Rd, Rs, #0	Rd = - Rs
1010	CMP Rd, Rs	CMP Rd, Rs	Set condition codes on Rd - Rs
1011	CMN Rd, Rs	CMN Rd, Rs	Set condition codes on Rd + Rs
1100	ORR Rd, Rs	ORRS Rd, Rd, Rs	Rd := Rd OR Rs
1101	MUL Rd, Rs	MULS Rd, Rs, Rd	Rd := Rs * Rd
1110	BIC Rd, Rs	BICS Rd, Rd, Rs	Rd := Rd AND NOT Rs
1111	MVN Rd, Rs	MVNS Rd, Rs	Rd := NOT Rs

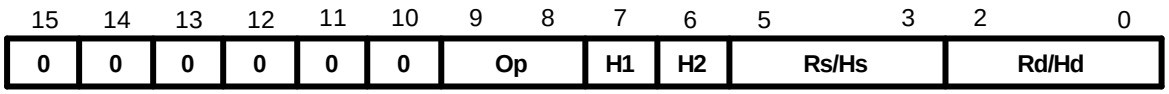
INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 4-5. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

```
EOR     R3, R4      ;  R3 := R3 EOR R4 and set condition codes
ROR     R1, R0      ;  Rotate Right R1 by the value in R0, store
                    ;  the result in R1 and set condition codes
NEG     R5, R3      ;  Subtract the contents of R3 from zero,
                    ;  Store the result in R5. Set condition codes ie R5 = - R3
CMP     R2, R6      ;  Set the condition codes on the result of R2 - R6
MUL     R0, R7      ;  R0 := R7 * R0 and set condition codes
```

FORMAT 5: HI-REGISTER OPERATIONS/BRANCH EXCHANGE



[2:0] Destination Register

[5:3] Source Register

[6] Hi Operand Flag 2

[7] Hi Operand Flag 1

[9:8] Opcode

Figure 4-6. Format 5

OPERATION

There are four sets of instructions in this group. The first three allow ADD, CMP and MOV operations to be performed between Lo and Hi registers, or a pair of Hi registers. The fourth, BX, allows a Branch to be performed which may also be used to switch processor state. The THUMB assembler syntax is shown in Table 4-6.

NOTE

In this group only CMP (Op = 01) sets the CPSR condition codes.

The action of H1= 0, H2 = 0 for Op = 00 (ADD), Op =01 (CMP) and Op = 10 (MOV) is undefined, and should not be used.

Table 4-6. Summary of Format 5 Instructions

Op	H1	H2	THUMB assembler	ARM equivalent	Action
00	0	1	ADD Rd, Hs	ADD Rd, Rd, Hs	Add a register in the range 8-15 to a register in the range 0-7.
00	1	0	ADD Hd, Rs	ADD Hd, Hd, Rs	Add a register in the range 0-7 to a register in the range 8-15.
00	1	1	ADD Hd, Hs	ADD Hd, Hd, Hs	Add two registers in the range 8-15
01	0	1	CMP Rd, Hs	CMP Rd, Hs	Compare a register in the range 0-7 with a register in the range 8-15. Set the condition code flags on the result.
01	1	0	CMP Hd, Rs	CMP Hd, Rs	Compare a register in the range 8-15 with a register in the range 0-7. Set the condition code flags on the result.

Table 4-6. Summary of Format 5 Instructions (Continued)

Op	H1	H2	THUMB assembler	ARM equivalent	Action
01	1	1	CMP Hd, Hs	CMP Hd, Hs	Compare two registers in the range 8-15. Set the condition code flags on the result.
10	0	1	MOV Rd, Hs	MOV Rd, Hs	Move a value from a register in the range 8-15 to a register in the range 0-7.
10	1	0	MOV Hd, Rs	MOV Hd, Rs	Move a value from a register in the range 0-7 to a register in the range 8-15.
10	1	1	MOV Hd, Hs	MOV Hd, Hs	Move a value between two registers in the range 8-15.
11	0	0	BX Rs	BX Rs	Perform branch (plus optional state change) to address in a register in the range 0-7.
11	0	1	BX Hs	BX Hs	Perform branch (plus optional state change) to address in a register in the range 8-15.

INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 4-6. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

THE BX INSTRUCTION

BX performs a Branch to a routine whose start address is specified in a Lo or Hi register.

Bit 0 of the address determines the processor state on entry to the routine:

Bit 0 = 0 Causes the processor to enter ARM state.
Bit 0 = 1 Causes the processor to enter THUMB state.

NOTE

The action of H1 = 1 for this instruction is undefined, and should not be used.

EXAMPLES

Hi-Register Operations

```
        ADD     PC, R5          ;  PC := PC + R5 but don't set the condition codes.
        CMP     R4, R12         ;  Set the condition codes on the result of R4 - R12.
        MOV     R15, R14        ;  Move R14 (LR) into R15 (PC)
                                ;  but don't set the condition codes,
                                ;  eg. return from subroutine.
```

Branch and Exchange

```
                                ;  Switch from THUMB to ARM state.
        ADR     R1,outofTHUMB   ;  Load address of outofTHUMB into R1.
        MOV     R11,R1
        BX      R11             ;  Transfer the contents of R11 into the PC.
                                ;  Bit 0 of R11 determines whether
                                ;  ARM or THUMB state is entered, ie. ARM state here.
        •
        •
        ALIGN
        CODE32
outofTHUMB                      ;  Now processing ARM instructions...
```

USING R15 AS AN OPERAND

If R15 is used as an operand, the value will be the address of the instruction + 4 with bit 0 cleared. Executing a BX PC in THUMB state from a non-word aligned address will result in unpredictable execution.

FORMAT 6: PC-RELATIVE LOAD

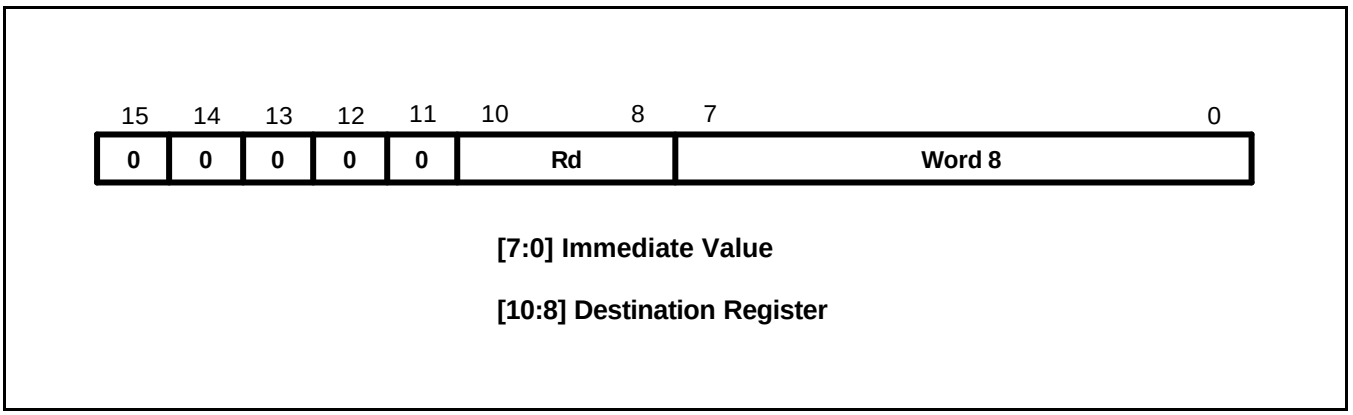


Figure 4-7. Format 6

OPERATION

This instruction loads a word from an address specified as a 10-bit immediate offset from the PC. The THUMB assembler syntax is shown below.

Table 4-7. Summary of PC-Relative Load Instruction

THUMB assembler	ARM equivalent	Action
LDR Rd, [PC, #Imm]	LDR Rd, [R15, #Imm]	Add unsigned offset (255 words, 1020 bytes) in Imm to the current value of the PC. Load the word from the resulting address into Rd.

NOTE: The value specified by #Imm is a full 10-bit address, but must always be word-aligned (ie with bits 1:0 set to 0), since the assembler places #Imm >> 2 in field Word 8. The value of the PC will be 4 bytes greater than the address of this instruction, but bit 1 of the PC is forced to 0 to ensure it is word aligned.

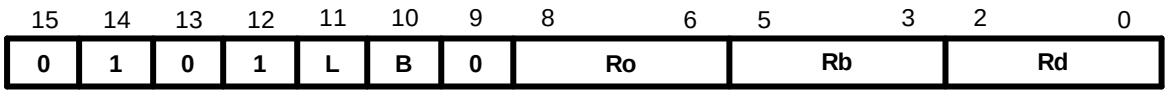
INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

```
LDR R3,[PC,#844]    ; Load into R3 the word found at the
                    ; address formed by adding 844 to PC.
                    ; bit[1] of PC is forced to zero.
                    ; Note that the THUMB opcode will contain
                    ; 211 as the Word8 value.
```

FORMAT 7: LOAD/STORE WITH REGISTER OFFSET



[2:0] Source/Destination Register

[5:3] Base Register

[8:6] Offset Register

[10] Byte/Word Flag
0 = Transfer word quantity
1 = Transfer byte quantity

[11] Load/Store Flag
0 = Store to memory
1 = Load from memory

Figure 4-8. Format 7

OPERATION

These instructions transfer byte or word values between registers and memory. Memory addresses are pre-indexed using an offset register in the range 0-7. The THUMB assembler syntax is shown in Table 4-8.

Table 4-8. Summary of Format 7 Instructions

L	B	THUMB assembler	ARM equivalent	Action
0	0	STR Rd, [Rb, Ro]	STR Rd, [Rb, Ro]	Pre-indexed word store: Calculate the target address by adding together the value in Rb and the value in Ro. Store the contents of Rd at the address.
0	1	STRB Rd, [Rb, Ro]	STRB Rd, [Rb, Ro]	Pre-indexed byte store: Calculate the target address by adding together the value in Rb and the value in Ro. Store the byte value in Rd at the resulting address.
1	0	LDR Rd, [Rb, Ro]	LDR Rd, [Rb, Ro]	Pre-indexed word load: Calculate the source address by adding together the value in Rb and the value in Ro. Load the contents of the address into Rd.
1	1	LDRB Rd, [Rb, Ro]	LDRB Rd, [Rb, Ro]	Pre-indexed byte load: Calculate the source address by adding together the value in Rb and the value in Ro. Load the byte value at the resulting address.

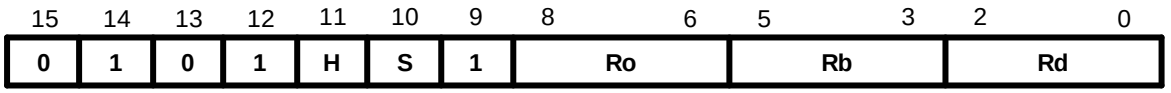
INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 4-8. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

```
STR     R3, [R2,R6]     ; Store word in R3 at the address
                        ; formed by adding R6 to R2.
LDRB    R2, [R0,R7]     ; Load into R2 the byte found at
                        ; the address formed by adding R7 to R0.
```

FORMAT 8: LOAD/STORE SIGN-EXTENDED BYTE/HALFWORD



[2:0] Destination Register

[5:3] Base Register

[8:6] Offset Register

[10] Sign-Extended Flag
0 = Operand not sing-extended
1 = Operand sing-extended

[11] H Flag

Figure 4-9. Format 8

OPERATION

These instructions load optionally sign-extended bytes or halfwords, and store halfwords. The THUMB assembler syntax is shown below.

Table 4-9. Summary of format 8 instructions

L	B	THUMB assembler	ARM equivalent	Action
0	0	STRH Rd, [Rb, Ro]	STRH Rd, [Rb, Ro]	Store halfword: Add Ro to base address in Rb. Store bits 0-15 of Rd at the resulting address.
0	1	LDRH Rd, [Rb, Ro]	LDRH Rd, [Rb, Ro]	Load halfword: Add Ro to base address in Rb. Load bits 0-15 of Rd from the resulting address, and set bits 16-31 of Rd to 0.
1	0	LDSB Rd, [Rb, Ro]	LDRSB Rd, [Rb, Ro]	Load sign-extended byte: Add Ro to base address in Rb. Load bits 0-7 of Rd from the resulting address, and set bits 8-31 of Rd to bit 7.
1	1	LDSH Rd, [Rb, Ro]	LDRSH Rd, [Rb, Ro]	Load sign-extended halfword: Add Ro to base address in Rb. Load bits 0-15 of Rd from the resulting address, and set bits 16-31 of Rd to bit 15.

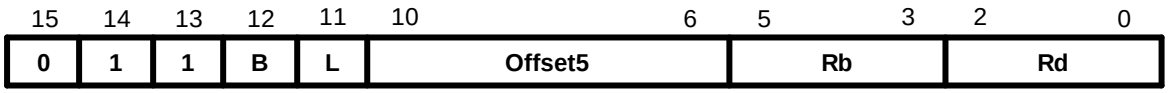
INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 4-9. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

```
STRH    R4, [R3, R0]    ;  Store the lower 16 bits of R4 at the
                        ;  address formed by adding R0 to R3.
LDSB    R2, [R7, R1]    ;  Load into R2 the sign extended byte
                        ;  found at the address formed by adding R1 to R7.
LDSH    R3, [R4, R2]    ;  Load into R3 the sign extended halfword
                        ;  found at the address formed by adding R2 to R4.
```

FORMAT 9: LOAD/STORE WITH IMMEDIATE OFFSET



[2:0] Source/Destination Register

[5:3] Base Register

[10:6] Offset Register

[11] Load/Store Flag
0 = Store to memory
1 = Load from memory

[12] Byte/Word Flad
0 = Transfer word quantity
1 = Transfer byte quantity

Figure 4-10. Format 9

OPERATION

These instructions transfer byte or word values between registers and memory using an immediate 5 or 7-bit offset. The THUMB assembler syntax is shown in Table 4-10.

Table 4-10. Summary of Format 9 Instructions

L	B	THUMB assembler	ARM equivalent	Action
0	0	STR Rd, [Rb, #Imm]	STR Rd, [Rb, #Imm]	Calculate the target address by adding together the value in Rb and Imm. Store the contents of Rd at the address.
1	0	LDR Rd, [Rb, #Imm]	LDR Rd, [Rb, #Imm]	Calculate the source address by adding together the value in Rb and Imm. Load Rd from the address.
0	1	STRB Rd, [Rb, #Imm]	STRB Rd, [Rb, #Imm]	Calculate the target address by adding together the value in Rb and Imm. Store the byte value in Rd at the address.
1	1	LDRB Rd, [Rb, #Imm]	LDRB Rd, [Rb, #Imm]	Calculate source address by adding together the value in Rb and Imm. Load the byte value at the address into Rd.

NOTE: For word accesses (B = 0), the value specified by #Imm is a full 7-bit address, but must be word-aligned (ie with bits 1:0 set to 0), since the assembler places #Imm >> 2 in the Offset5 field.

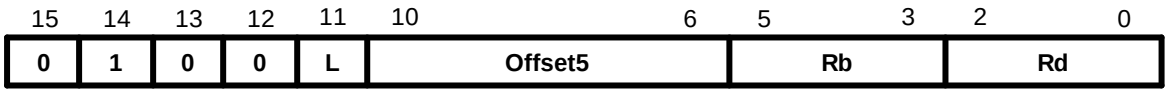
INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 4-10. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

```
LDR     R2, [R5,#116]   ; Load into R2 the word found at the
                        ; address formed by adding 116 to R5.
                        ; Note that the THUMB opcode will
                        ; contain 29 as the Offset5 value.
STRB    R1, [R0,#13]    ; Store the lower 8 bits of R1 at the
                        ; address formed by adding 13 to R0.
                        ; Note that the THUMB opcode will
                        ; contain 13 as the Offset5 value.
```

FORMAT 10: LOAD/STORE HALFWORD



[2:0] Source/Destination Register

[5:3] Base Register

[10:6] Immediate Value

[11] Load/Store Flag
0 = Store to memory
1 = Load from memory

Figure 4-11. Format 10

OPERATION

These instructions transfer halfword values between a Lo register and memory. Addresses are pre-indexed, using a 6-bit immediate value. The THUMB assembler syntax is shown in Table 4-11.

Table 4-11. Halfword Data Transfer Instructions

L	THUMB assembler	ARM equivalent	Action
0	STRH Rd, [Rb, #Imm]	STRH Rd, [Rb, #Imm]	Add #Imm to base address in Rb and store bits 0 - 15 of Rd at the resulting address.
1	LDRH Rd, [Rb, #Imm]	LDRH Rd, [Rb, #Imm]	Add #Imm to base address in Rb. Load bits 0-15 from the resulting address into Rd and set bits 16-31 to zero.

NOTE: #Imm is a full 6-bit address but must be halfword-aligned (ie with bit 0 set to 0) since the assembler places #Imm >> 1 in the Offset5 field.

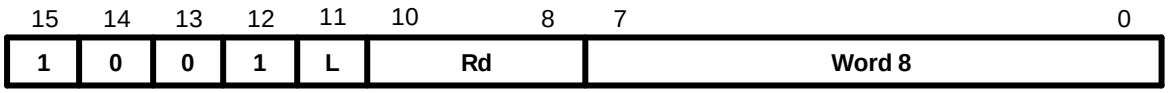
INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 4-11. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

```
STRH    R6, [R1, #56]   ; Store the lower 16 bits of R4 at the address formed by
                        ; adding 56 R1. Note that the THUMB opcode will contain
                        ; 28 as the Offset5 value.
LDRH    R4, [R7, #4]    ; Load into R4 the halfword found at the address formed by
                        ; adding 4 to R7. Note that the THUMB opcode will contain
                        ; 2 as the Offset5 value.
```

FORMAT 11: SP-RELATIVE LOAD/STORE



[7:0] Immediate Value

[10:8] Destination Register

[11] Load/Store Bit
0 = Store to memory
1 = Load from memory

Figure 4-12. Format 11

OPERATION

The instructions in this group perform an SP-relative load or store. The THUMB assembler syntax is shown in the following table.

Table 4-12. SP-Relative Load/Store Instructions

L	THUMB assembler	ARM equivalent	Action
0	STR Rd, [SP, #Imm]	STR Rd, [R13 #Imm]	Add unsigned offset (255 words, 1020 bytes) in Imm to the current value of the SP (R7). Store the contents of Rd at the resulting address.
1	LDR Rd, [SP, #Imm]	LDR Rd, [R13 #Imm]	Add unsigned offset (255 words, 1020 bytes) in Imm to the current value of the SP (R7). Load the word from the resulting address into Rd.

NOTE: The offset supplied in #Imm is a full 10-bit address, but must always be word-aligned (ie bits 1:0 set to 0), since the assembler places #Imm >> 2 in the Word8 field.

INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 4-12. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

```
STR     R4, [SP,#492]   ; Store the contents of R4 at the address
                        ; formed by adding 492 to SP (R13).
                        ; Note that the THUMB opcode will contain
                        ; 123 as the Word8 value.
```

FORMAT 12: LOAD ADDRESS

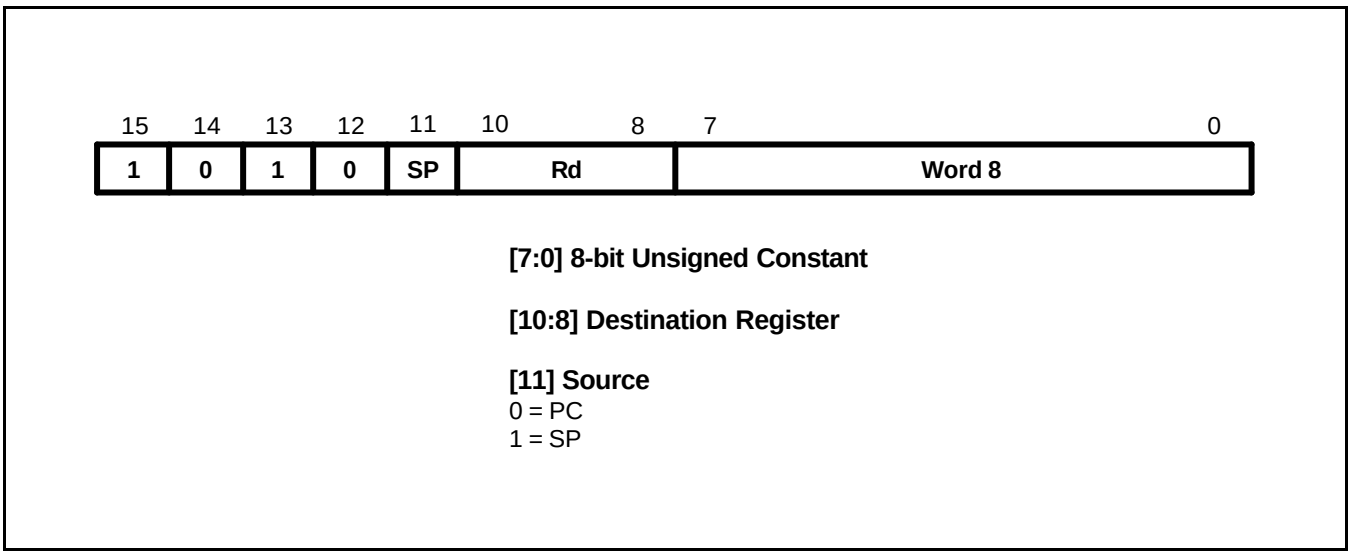


Figure 4-13. Format 12

OPERATION

These instructions calculate an address by adding an 10-bit constant to either the PC or the SP, and load the resulting address into a register. The THUMB assembler syntax is shown in the following table.

Table 4-13. Load Address

L	THUMB assembler	ARM equivalent	Action
0	ADD Rd, PC, #Imm	ADD Rd, R15, #Imm	Add #Imm to the current value of the program counter (PC) and load the result into Rd.
1	ADD Rd, SP, #Imm	ADD Rd, R13, #Imm	Add #Imm to the current value of the stack pointer (SP) and load the result into Rd.

NOTE: The value specified by #Imm is a full 10-bit value, but this must be word-aligned (ie with bits 1:0 set to 0) since the assembler places #Imm >> 2 in field Word 8.

Where the PC is used as the source register (SP = 0), bit 1 of the PC is always read as 0. The value of the PC will be 4 bytes greater than the address of the instruction before bit 1 is forced to 0.

The CPSR condition codes are unaffected by these instructions.

INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 4-13. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

```
ADD     R2, PC, #572    ; R2 := PC + 572, but don't set the
                        ; condition codes. bit[1] of PC is forced to zero.
                        ; Note that the THUMB opcode will
                        ; contain 143 as the Word8 value.
ADD     R6, SP, #212    ; R6 := SP (R13) + 212, but don't
                        ; set the condition codes.
                        ; Note that the THUMB opcode will
                        ; contain 53 as the Word 8 value.
```

FORMAT 13: ADD OFFSET TO STACK POINTER

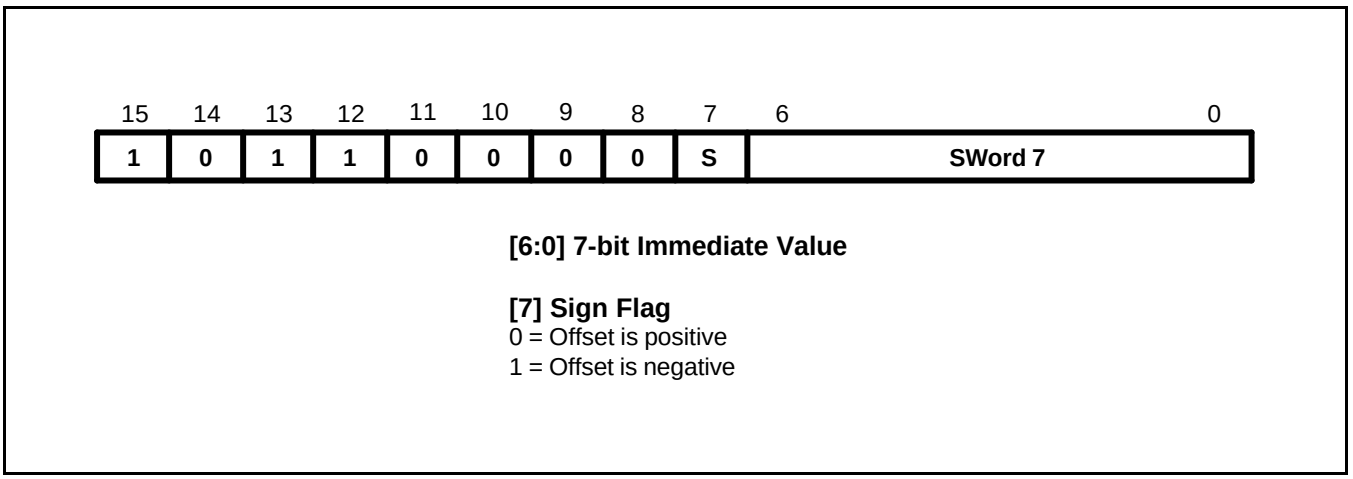


Figure 4-14. Format 13

OPERATION

This instruction adds a 9-bit signed constant to the stack pointer. The following table shows the THUMB assembler syntax.

Table 4-14. The ADD SP Instruction

L	THUMB assembler	ARM equivalent	Action
0	ADD SP, #Imm	ADD R13, R13, #Imm	Add #Imm to the stack pointer (SP).
1	ADD SP, # -Imm	SUB R13, R13, #Imm	Add #-Imm to the stack pointer (SP).

NOTE: The offset specified by #Imm can be up to -/+ 508, but must be word-aligned (ie with bits 1:0 set to 0) since the assembler converts #Imm to an 8-bit sign + magnitude number before placing it in field SWord7. The condition codes are not set by this instruction.

INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 4-14. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

```
ADD     SP, #268        ; SP (R13) := SP + 268, but don't set the condition codes.
                        ; Note that the THUMB opcode will
                        ; contain 67 as the Word7 value and S=0.
ADD     SP, #-104       ; SP (R13) := SP - 104, but don't set the condition codes.
                        ; Note that the THUMB opcode will contain
                        ; 26 as the Word7 value and S=1.
```

FORMAT 14: PUSH/POP REGISTERS

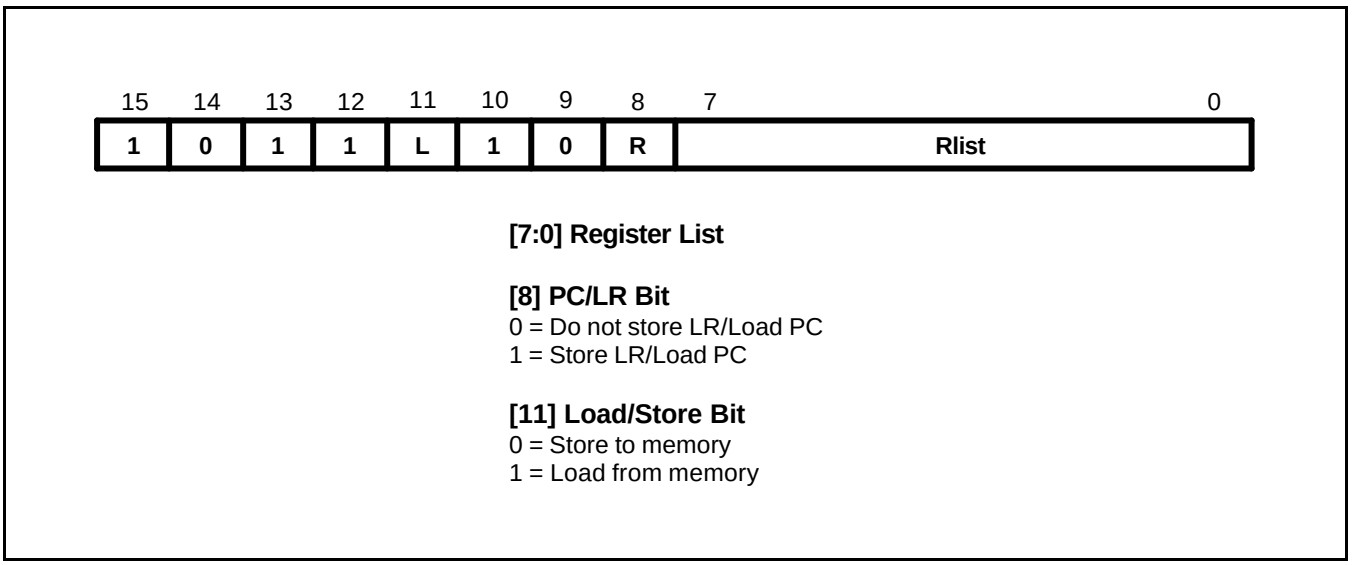


Figure 4-15. Format 14

OPERATION

The instructions in this group allow registers 0-7 and optionally LR to be pushed onto the stack, and registers 0-7 and optionally PC to be popped off the stack. The THUMB assembler syntax is shown in Table 4-15.

NOTE

The stack is always assumed to be Full Descending.

Table 4-15. PUSH and POP Instructions

L	B	THUMB assembler	ARM equivalent	Action
0	0	PUSH { Rlist }	STMDB R13!, { Rlist }	Push the registers specified by Rlist onto the stack. Update the stack pointer.
0	1	PUSH { Rlist, LR }	STMDB R13!, { Rlist, R14 }	Push the Link Register and the registers specified by Rlist (if any) onto the stack. Update the stack pointer.
1	0	POP { Rlist }	LDMIA R13!, { Rlist }	Pop values off the stack into the registers specified by Rlist. Update the stack pointer.
1	1	POP { Rlist, PC }	LDMIA R13!, {Rlist, R15}	Pop values off the stack and load into the registers specified by Rlist. Pop the PC off the stack. Update the stack pointer.

INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 4-15. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

```
PUSH    {R0-R4,LR}      ; Store R0,R1,R2,R3,R4 and R14 (LR) at
                        ; the stack pointed to by R13 (SP) and update R13.
                        ; Useful at start of a sub-routine to
                        ; save workspace and return address.
POP     {R2,R6,PC}      ; Load R2,R6 and R15 (PC) from the stack
                        ; pointed to by R13 (SP) and update R13.
                        ; Useful to restore workspace and return from sub-routine.
```

FORMAT 15: MULTIPLE LOAD/STORE

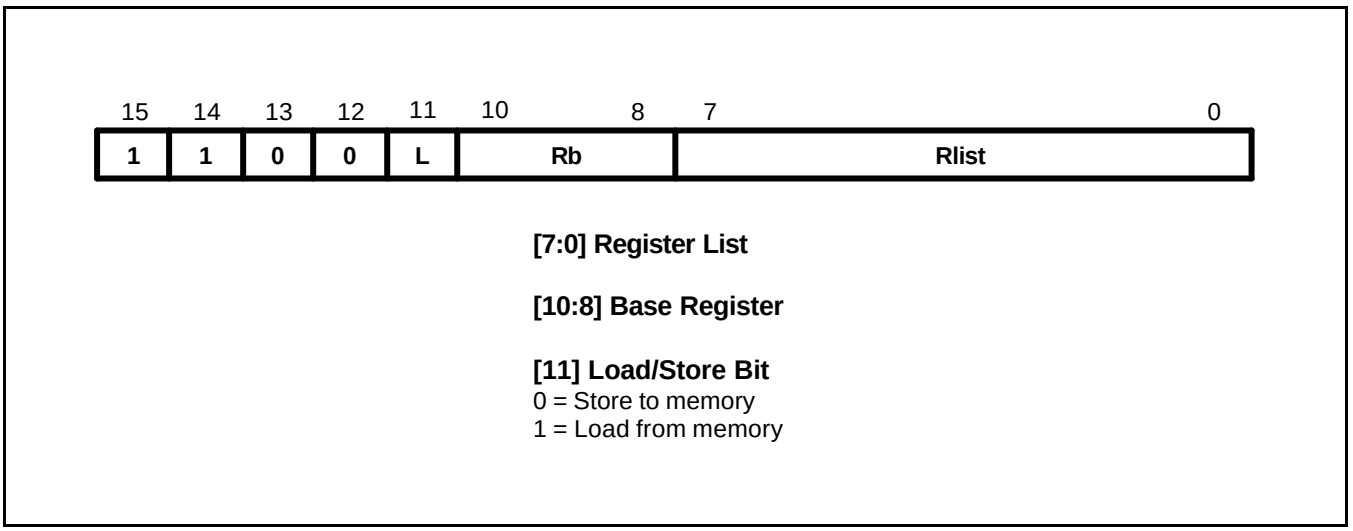


Figure 4-16. Format 15

OPERATION

These instructions allow multiple loading and storing of Lo registers. The THUMB assembler syntax is shown in the following table.

Table 4-16. The Multiple Load/Store Instructions

L	THUMB assembler	ARM equivalent	Action
0	STMIA Rb!, { Rlist }	STMIA Rb!, { Rlist }	Store the registers specified by Rlist, starting at the base address in Rb. Write back the new base address.
1	LDMIA Rb!, { Rlist }	LDMIA Rb!, { Rlist }	Load the registers specified by Rlist, starting at the base address in Rb. Write back the new base address.

INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 4-16. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

```
STMIA   R0!, {R3-R7}    ; Store the contents of registers R3-R7
                        ; starting at the address specified in
                        ; R0, incrementing the addresses for each word.
                        ; Write back the updated value of R0.
```

FORMAT 16: CONDITIONAL BRANCH

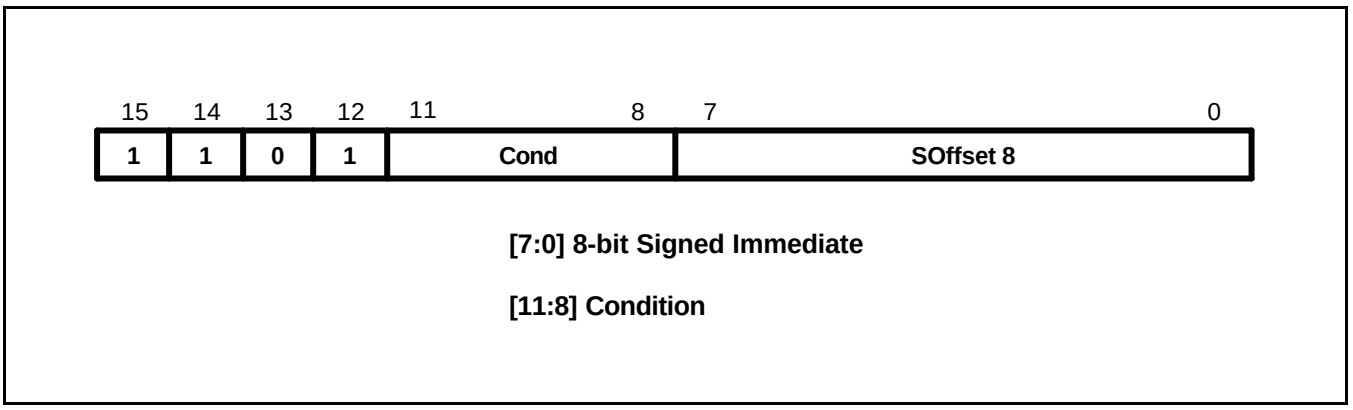


Figure 4-17. Format 16

OPERATION

The instructions in this group all perform a conditional Branch depending on the state of the CPSR condition codes. The branch offset must take account of the prefetch operation, which causes the PC to be 1 word (4 bytes) ahead of the current instruction.

The THUMB assembler syntax is shown in the following table.

Table 4-17. The Conditional Branch Instructions

L	THUMB assembler	ARM equivalent	Action
0000	BEQ label	BEQ label	Branch if Z set (equal)
0001	BNE label	BNE label	Branch if Z clear (not equal)
0010	BCS label	BCS label	Branch if C set (unsigned higher or same)
0011	BCC label	BCC label	Branch if C clear (unsigned lower)
0100	BMI label	BMI label	Branch if N set (negative)
0101	BPL label	BPL label	Branch if N clear (positive or zero)
0110	BVS label	BVS label	Branch if V set (overflow)
0111	BVC label	BVC label	Branch if V clear (no overflow)
1000	BHI label	BHI label	Branch if C set and Z clear (unsigned higher)

Table 4-17. The Conditional Branch Instructions (Continued)

L	THUMB assembler	ARM equivalent	Action
1001	BLS label	BLS label	Branch if C clear or Z set (unsigned lower or same)
1010	BGE label	BGE label	Branch if N set and V set, or N clear and V clear (greater or equal)
1011	BLT label	BLT label	Branch if N set and V clear, or N clear and V set (less than)
1100	BGT label	BGT label	Branch if Z clear, and either N set and V set or N clear and V clear (greater than)
1101	BLE label	BLE label	Branch if Z set, or N set and V clear, or N clear and V set (less than or equal)

NOTES:

1. While label specifies a full 9-bit two's complement address, this must always be halfword-aligned (ie with bit 0 set to 0) since the assembler actually places label >> 1 in field SOffset8.
2. Cond = 1110 is undefined, and should not be used.
   Cond = 1111 creates the SWI instruction: see .

INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 3-1. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

```
        CMP R0, #45 ;       Branch to over-if R0 > 45.
        BGT over            ;  Note that the THUMB opcode will contain
        •                   ;  the number of halfwords to offset.
        •
over    •                   ;  Must be halfword aligned.
```

FORMAT 17: SOFTWARE INTERRUPT

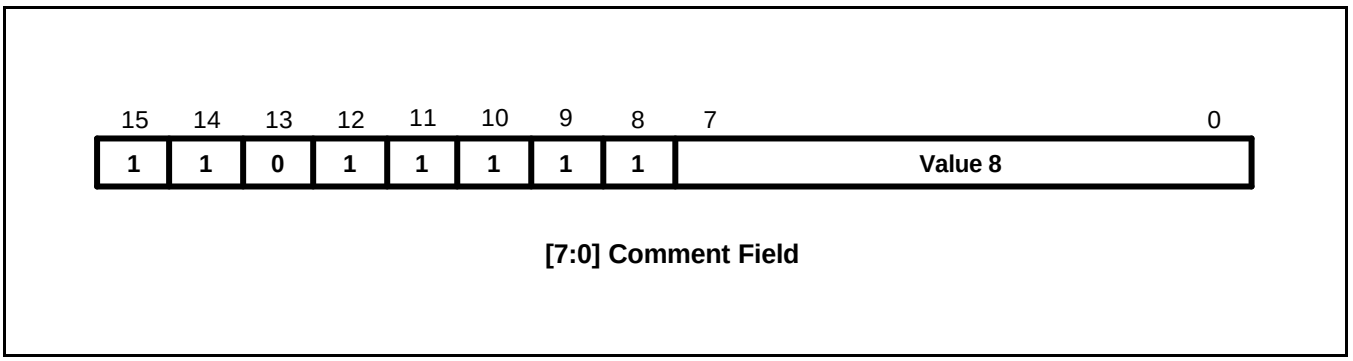


Figure 4-18. Format 17

OPERATION

The SWI instruction performs a software interrupt. On taking the SWI, the processor switches into ARM state and enters Supervisor (SVC) mode.

The THUMB assembler syntax for this instruction is shown below.

Table 4-18. The SWI Instruction

THUMB assembler	ARM equivalent	Action
SWI Value 8	SWI Value 8	Perform Software Interrupt: Move the address of the next instruction into LR, move CPSR to SPSR, load the SWI vector address (0x8) into the PC. Switch to ARM state and enter SVC mode.

NOTE: Value8 is used solely by the SWI handler; it is ignored by the processor.

INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 4-18. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

```
        SWI 18          ; Take the software interrupt exception.
                        ; Enter Supervisor mode with 18 as the
                        ; requested SWI number.
```

FORMAT 18: UNCONDITIONAL BRANCH

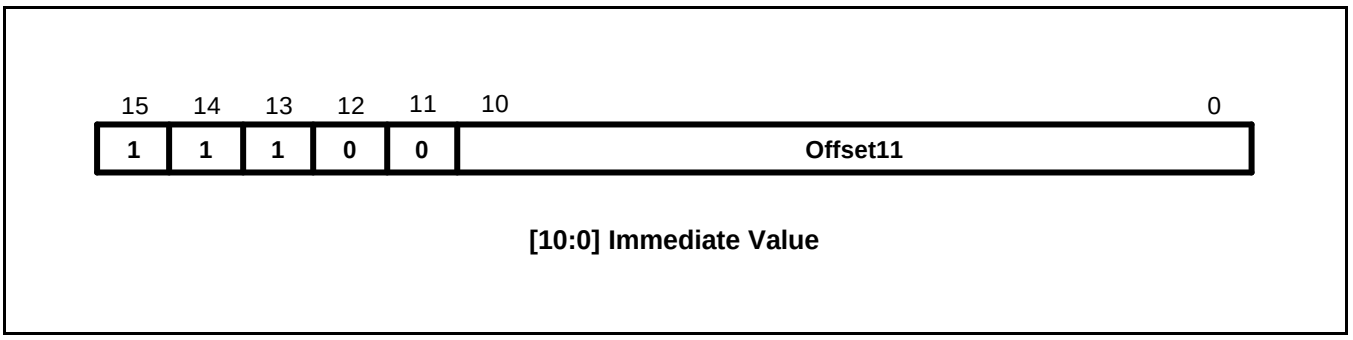


Figure 4-19. Format 18

OPERATION

This instruction performs a PC-relative Branch. The THUMB assembler syntax is shown below. The branch offset must take account of the prefetch operation, which causes the PC to be 1 word (4 bytes) ahead of the current instruction.

Table 4-19. Summary of Branch Instruction

THUMB assembler	ARM equivalent	Action
B label	BAL label (halfword offset)	Branch PC relative +/- Offset11 << 1, where label is PC +/- 2048 bytes.

NOTE: The address specified by label is a full 12-bit two's complement address, but must always be halfword aligned (ie bit 0 set to 0), since the assembler places label >> 1 in the Offset11 field.

EXAMPLES

```
here    B here          ;  Branch onto itself. Assembles to 0xE7FE.
                        ;  (Note effect of PC offset).
        B jimmy         ;  Branch to 'jimmy'.
        •               ;  Note that the THUMB opcode will contain the number of
        •
        •               ;  halfwords to offset.
jimmy   •               ;  Must be halfword aligned.
```

FORMAT 19: LONG BRANCH WITH LINK

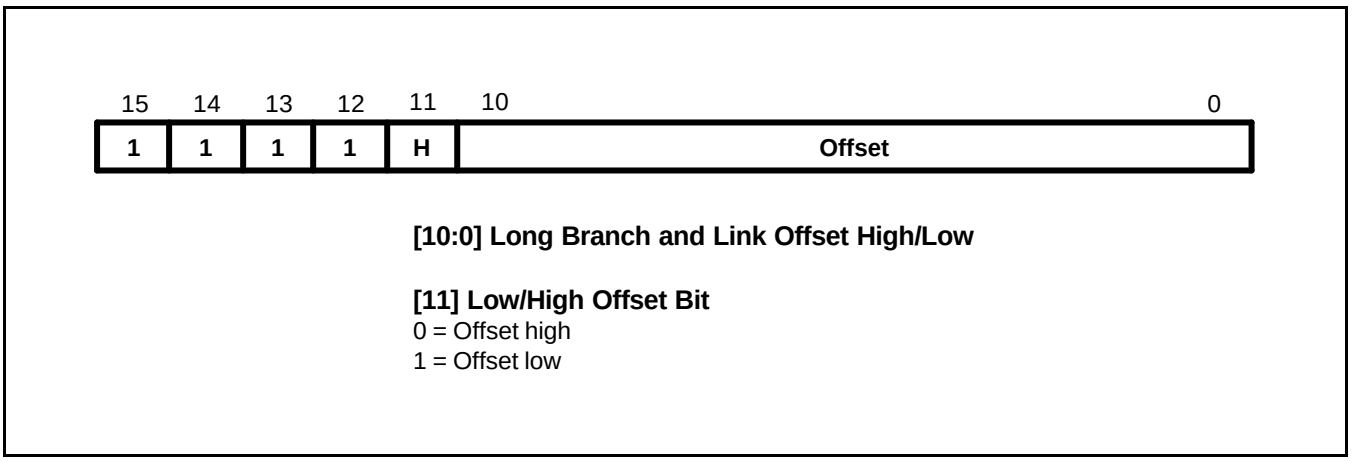


Figure 4-20. Format 19

OPERATION

This format specifies a long branch with link.

The assembler splits the 23-bit two's complement half-word offset specified by the label into two 11-bit halves, ignoring bit 0 (which must be 0), and creates two THUMB instructions.

Instruction 1 (H = 0)

In the first instruction the Offset field contains the upper 11 bits of the target address. This is shifted left by 12 bits and added to the current PC address. The resulting address is placed in LR.

Instruction 2 (H =1)

In the second instruction the Offset field contains an 11-bit representation lower half of the target address. This is shifted left by 1 bit and added to LR. LR, which now contains the full 23-bit address, is placed in PC, the address of the instruction following the BL is placed in LR and bit 0 of LR is set.

The branch offset must take account of the prefetch operation, which causes the PC to be 1 word (4 bytes) ahead of the current instruction

INSTRUCTION CYCLE TIMES

This instruction format does not have an equivalent ARM instruction.

Table 4-20. The BL Instruction

L	THUMB assembler	ARM equivalent	Action
0	BL label	none	LR := PC + OffsetHigh << 12
1			temp := next instruction address PC := LR + OffsetLow << 1 LR := temp | 1

EXAMPLES

```
        BL faraway      ; Unconditionally Branch to 'faraway'
next    •               ; and place following instruction
        •               ; address, ie "next", in R14,the Link
                        ; register and set bit 0 of LR high.
                        ; Note that the THUMB opcodes will
                        ; contain the number of halfwords to offset.
faraway •               ; Must be Half-word aligned.
        •
```

INSTRUCTION SET EXAMPLES

The following examples show ways in which the THUMB instructions may be used to generate small and efficient code. Each example also shows the ARM equivalent so these may be compared.

MULTIPLICATION BY A CONSTANT USING SHIFTS AND ADDS

The following shows code to multiply by various constants using 1, 2 or 3 Thumb instructions alongside the ARM equivalents. For other constants it is generally better to use the built-in MUL instruction rather than using a sequence of 4 or more instructions.

```
            Thumb       ARM
```

1. Multiplication by 2^n (1,2,4,8,...)

```
            LSL         Ra, Rb, LSL #n      ;  MOV Ra, Rb, LSL #n
```

2. Multiplication by 2^n+1 (3,5,9,17,...)

```
            LSL         Rt, Rb, #n          ;  ADD Ra, Rb, Rb, LSL #n
            ADD         Ra, Rt, Rb
```

3. Multiplication by 2^n-1 (3,7,15,...)

```
            LSL         Rt, Rb, #n          ;  RSB Ra, Rb, Rb, LSL #n
            SUB         Ra, Rt, Rb
```

4. Multiplication by -2^n (-2, -4, -8, ...)

```
            LSL         Ra, Rb, #n          ;  MOV Ra, Rb, LSL #n
            MVN         Ra, Ra              ;  RSB Ra, Ra, #0
```

5. Multiplication by -2^n-1 (-3, -7, -15, ...)

```
            LSL         Rt, Rb, #n          ;  SUB Ra, Rb, Rb, LSL #n
            SUB         Ra, Rb, Rt
```

Multiplication by any C = {2^n+1, 2^n-1, -2^n or -2^n-1} * 2^n
Effectively this is any of the multiplications in 2 to 5 followed by a final shift. This allows the following additional constants to be multiplied. 6, 10, 12, 14, 18, 20, 24, 28, 30, 34, 36, 40, 48, 56, 60, 62 .....

```
            (2..5)                          ;  (2..5)
            LSL         Ra, Ra, #n          ;  MOV Ra, Ra, LSL #n
```

GENERAL PURPOSE SIGNED DIVIDE

This example shows a general purpose signed divide and remainder routine in both Thumb and ARM code.

Thumb code

```
;signed_divide                          ;  Signed divide of R1 by R0: returns quotient in R0,
                                        ;  remainder in R1

;Get abs value of R0 into R3
            ASR     R2, R0, #31         ;  Get 0 or -1 in R2 depending on sign of R0
            EOR     R0, R2              ;  EOR with -1 (0×FFFFFFFF) if negative
            SUB     R3, R0, R2          ;  and ADD 1 (SUB -1) to get abs value

;SUB always sets flag so go & report division by 0 if necessary
            BEQ     divide_by_zero

;Get abs value of R1 by xoring with 0xFFFFFFFF and adding 1 if negative
            ASR     R0, R1, #31         ;  Get 0 or -1 in R3 depending on sign of R1
            EOR     R1, R0              ;  EOR with -1 (0×FFFFFFFF) if negative
            SUB     R1, R0              ;  and ADD 1 (SUB -1) to get abs value

;Save signs (0 or -1 in R0 & R2) for later use in determining ; sign of quotient & remainder.
            PUSH    {R0, R2}

;Justification, shift 1 bit at a time until divisor (R0 value) ; is just <= than dividend (R1 value). To do this shift dividend
; right by 1 and stop as soon as shifted value becomes >.
            LSR     R0, R1, #1
            MOV     R2, R3
            B       %FT0
just_l      LSL     R2, #1
0           CMP     R2, R0
            BLS     just_l
            MOV     R0, #0              ;  Set accumulator to 0
            B       %FT0                ;  Branch into division loop

div_l       LSR     R2, #1
0           CMP     R1, R2              ;  Test subtract
            BCC     %FT0
            SUB     R1, R2              ;  If successful do a real subtract
0           ADC     R0, R0              ;  Shift result and add 1 if subtract succeeded

            CMP     R2, R3              ;  Terminate when R2 == R3 (ie we have just
            BNE     div_l               ;  tested subtracting the 'ones' value).
```

```
Now fix up the signs of the quotient (R0) and remainder (R1)
                POP     {R2, R3}                ;   Get dividend/divisor signs back
                EOR     R3, R2                  ;   Result sign
                EOR     R0, R3                  ;   Negate if result sign = - 1
                SUB     R0, R3
                EOR     R1, R2                  ;   Negate remainder if dividend sign = - 1
                SUB     R1, R2
                MOV     pc, lr
```

ARM Code

```
signed_divide           ;                       Effectively zero a4 as top bit will be shifted out later
                ANDS    a4, a1, #&80000000
                RSBMI   a1, a1, #0
                EORS    ip, a4, a2, ASR #32
;ip bit 31 = sign of result
;ip bit 30 = sign of a2
                RSBCS    a2, a2, #0

;Central part is identical code to udiv (without MOV a4, #0 which comes for free as part of signed entry sequence)
                MOVS    a3, a1
                BEQ     divide_by_zero

just_l                                          ;   Justification stage shifts 1 bit at a time
                CMP     a3, a2, LSR #1
                MOVLS   a3, a3, LSL #1          ;   NB: LSL #1 is always OK if LS succeeds
                BLO     s_loop

div_l
                CMP     a2, a3
                ADC     a4, a4, a4
                SUBCS   a2, a2, a3
                TEQ     a3, a1
                MOVNE   a3, a3, LSR #1
                BNE     s_loop2
                MOV     a1, a4
                MOVS    ip, ip, ASL #1
                RSBCS   a1, a1, #0
                RSBMI   a2, a2, #0
                MOV     pc, lr
```

DIVISION BY A CONSTANT

Division by a constant can often be performed by a short fixed sequence of shifts, adds and subtracts.

Here is an example of a divide by 10 routine based on the algorithm in the ARM Cookbook in both Thumb and ARM code.

Thumb Code

```
udiv10                                  ;  Take argument in a1 returns quotient in a1,
                                        ;  remainder in a2
        MOV     a2, a1
        LSR     a3, a1, #2
        SUB     a1, a3
        LSR     a3, a1, #4
        ADD     a1, a3
        LSR     a3, a1, #8
        ADD     a1, a3
        LSR     a3, a1, #16
        ADD     a1, a3
        LSR     a1, #3
        ASL     a3, a1, #2
        ADD     a3, a1
        ASL     a3, #1
        SUB     a2, a3
        CMP     a2, #10
        BLT     %FT0
        ADD      a1, #1
        SUB     a2, #10
0
        MOV      pc, lr
```

ARM Code

```
udiv10                                  ;  Take argument in a1 returns quotient in a1,
                                        ;  remainder in a2
        SUB     a2, a1, #10
        SUB     a1, a1, a1, lsr #2
        ADD     a1, a1, a1, lsr #4
        ADD     a1, a1, a1, lsr #8
        ADD     a1, a1, a1, lsr #16
        MOV     a1, a1, lsr #3
        ADD     a3, a1, a1, asl #2
        SUBS    a2, a2, a3, asl #1
        ADDPL   a1, a1, #1
        ADDMI   a2, a2, #10
        MOV     pc, lr
```

NOTES

5 MEMORY CONTROLLER

OVERVIEW

The S3C2410A's memory controller provides memory control signals required for external memory access.

The S3C2410A has the following features:

— Little/Big endian (selectable by a software)

— Address space: 128Mbytes per bank (total 1GB/8 banks)

— Programmable access size (8/16/32-bit) for all banks except bank0 (16/32-bit)

— Total 8 memory banks
Six memory banks for ROM, SRAM, etc.
Remaining two memory banks for ROM, SRAM, SDRAM, etc .

— Seven fixed memory bank start address

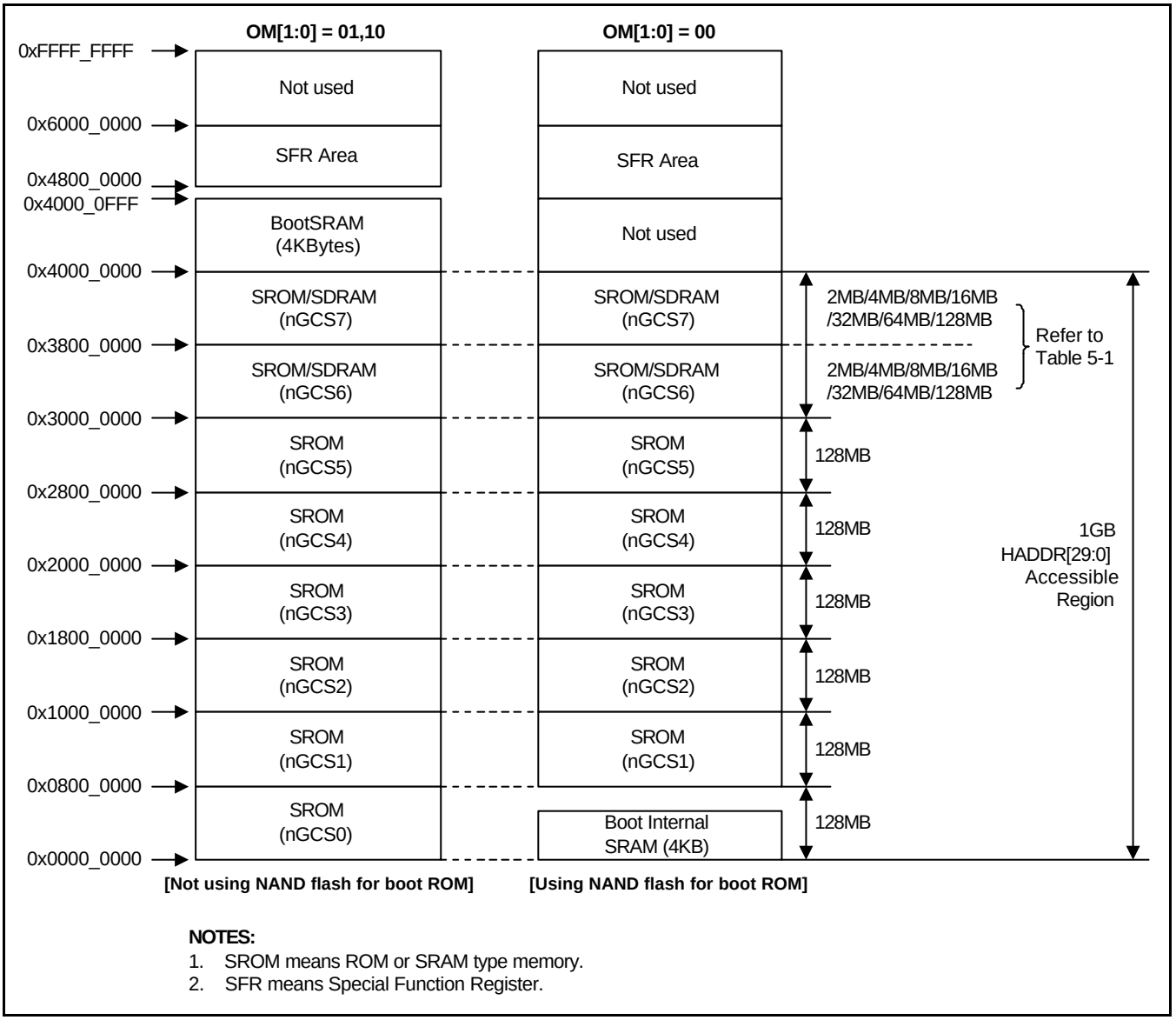
— Adjustable start address for the last bank.

— Programmable bank size for the last two banks.

— Programmable access cycles for all memory banks

— External wait to extend the bus cycles

— Supporting self-refresh and power down mode for SDRAM



NOTES:
1. SROM means ROM or SRAM type memory.
2. SFR means Special Function Register.

Figure 5-1. S3C2410A Memory Map after Reset

Table 5-1. Bank 6/7 Addresses

Address	2MB	4MB	8MB	16MB	32MB	64MB	128MB
Bank 6							
Start address	0x3000_0000	0x3000_0000	0x3000_0000	0x3000_0000	0x3000_0000	0x3000_0000	0x3000_0000
End address	0x301f_ffff	0x303f_ffff	0x307f_ffff	0x30ff_ffff	0x31ff_ffff	0x33ff_ffff	0x37ff_ffff
Bank 7							
Start address	0x3020_0000	0x3040_0000	0x3080_0000	0x3100_0000	0x3200_0000	0x3400_0000	0x3800_0000
End address	0x303f_ffff	0x307f_ffff	0x30ff_ffff	0x31ff_ffff	0x33ff_ffff	0x37ff_ffff	0x3fff_ffff

NOTE: Bank 6 and 7 must have the same memory size.

FUNCTION DESCRIPTION

BANK0 BUS WIDTH

The data bus of BANK0 (nGCS0) should be configured to either 16-bit or 32-bit accordingly. Because the BANK0 works as the booting ROM bank (map to 0x0000_0000), the bus width of BANK0 should be determined before the first ROM access, which will depend on the logic level of OM[1:0] at Reset.

OM1 (Operating Mode 1)	OM0 (Operating Mode 0)	Booting ROM Data width
0	0	Nand Flash Mode
0	1	16-bit
1	0	32-bit
1	1	Test Mode

MEMORY (SROM/SDRAM) ADDRESS PIN CONNECTIONS

MEMORY ADDR. PIN	S3C2410A ADDR. @ 8-bit DATA BUS	S3C2410A ADDR. @ 16-bit DATA BUS	S3C2410A ADDR. @ 32-bit DATA BUS
A0	A0	A1	A2
A1	A1	A2	A3
•••	•••	•••	•••

SDRAM BANK ADDRESS PIN CONNECTION

Table 5-2. SDRAM Bank Address Configuration

Bank Size	Bus Width	Base Component	Memory Configuration	Bank Address
2MB	x8	16Mb	(1M x 8 x 2banks) x 1 ea	A20
	x16		(512K x 16 x 2banks) x 1 ea	
4MB	x8	16Mb	(2M x 4 x 2banks) x 2 ea	A21
	x16		(1M x 8 x 2banks) x 2 ea	
	x32		(512K x 16 x 2banks) x 2 ea	
8MB	x16	16Mb	(2M x 4 x 2banks) x 4 ea	A22
	x32		(1M x 8 x 2banks) x 4 ea	
	x8	64Mb	(4M x 8 x 2banks) x 1 ea	
	x8		(2M x 8 x 4banks) x 1 ea	A[22:21]
	x16		(2M x 16 x 2banks) x 1 ea	A22
	x16		(1M x 16 x 4banks) x 1 ea	A[22:21]
	x32		(512K x 32 x 4banks) x 1 ea	
16MB	x32	16Mb	(2M x 4 x 2banks) x 8 ea	A23
	x8	64Mb	(8M x 4 x 2banks) x 2 ea	
	x8		(4M x 4 x 4banks) x 2 ea	A[23:22]
	x16		(4M x 8 x 2banks) x 2 ea	A23
	x16		(2M x 8 x 4banks) x 2 ea	A[23:22]
	x32		(2M x 16 x 2banks) x 2 ea	A23
	x32		(1M x 16 x 4banks) x 2 ea	A[23:22]
	x8	128Mb	(4M x 8 x 4banks) x 1 ea	
	x16		(2M x 16 x 4banks) x 1 ea	
32MB	x16	64Mb	(8M x 4 x 2banks) x 4 ea	A24
	x16		(4M x 4 x 4banks) x 4 ea	A[24:23]
	x32		(4M x 8 x 2banks) x 4 ea	A24
	x32		(2M x 8 x 4banks) x 4 ea	A[24:23]
	x16	128Mb	(4M x 8 x 4banks) x 2 ea	
	x32		(2M x 16 x 4banks) x 2 ea	
	x8	256Mb	(8M x 8 x 4banks) x 1 ea	
	x16		(4M x 16 x 4banks) x 1 ea	
64MB	x32	128Mb	(4M x 8 x 4banks) x 4 ea	A[25:24]
	x16	256Mb	(8M x 8 x 4banks) x 2 ea	
	x32		(4M x 16 x 4banks) x 2 ea	
	x8	512Mb	(16M x 8 x 4banks) x 1 ea	
128MB	x32	256Mb	(8M x 8 x 4banks) x 4 ea	A[26:25]
	x8	512Mb	(32M x 4 x 4banks) x 2 ea	
	x16		(16M x 8 x 4banks) x 2 ea	
	x32		(8M x 16 x 4banks) x 2 ea	

nWAIT PIN OPERATION

If the WAIT corresponding to each memory bank is enabled, the nOE duration should be prolonged by the external nWAIT pin while the memory bank is active. nWAIT is checked from tacc-1. nOE will be deasserted at the next clock after sampling nWAIT is high. The nWE signal have the same relation with nOE.

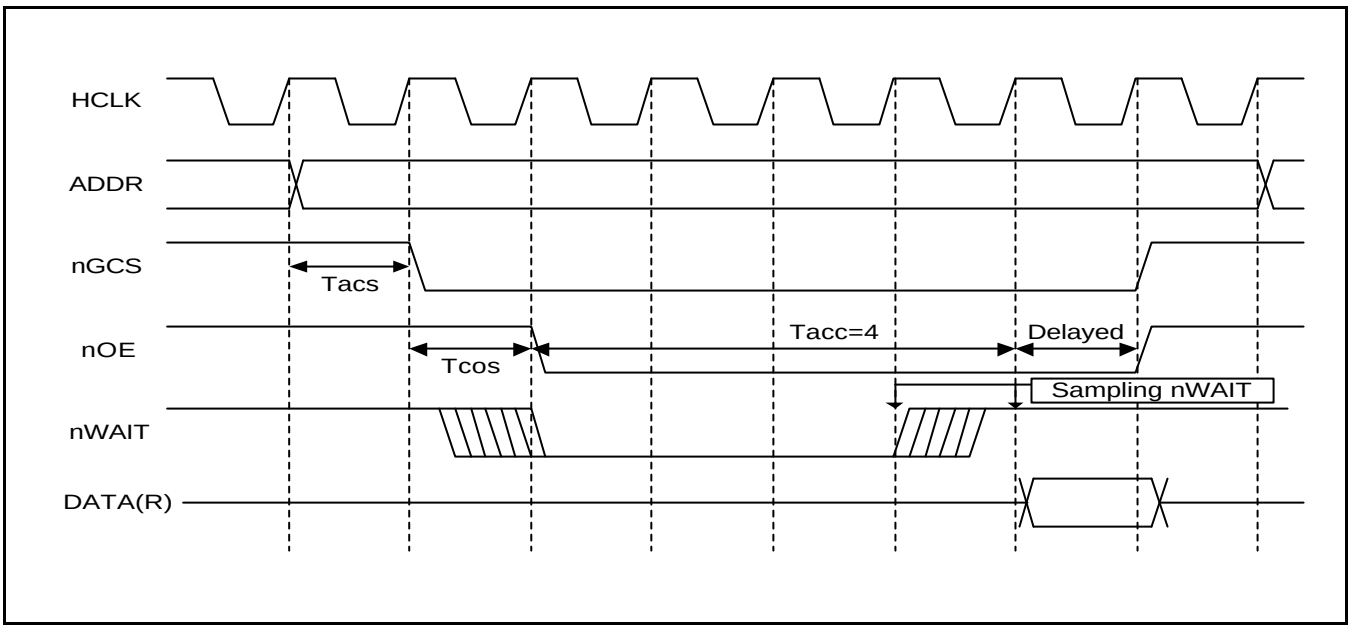


Figure 5-2. S3C2410A External nWAIT Timing Diagram (Tacc = 4)

nXBREQ/nXBACK Pin Operation

If nXBREQ is asserted, the S3C2410A will respond by lowering nXBACK. If nXBACK = L, the address/data bus and memory control signals are in Hi-z state as shown in Table 1-1. When nXBREQ is de-asserted, the nXBACK will also be de-asserted.

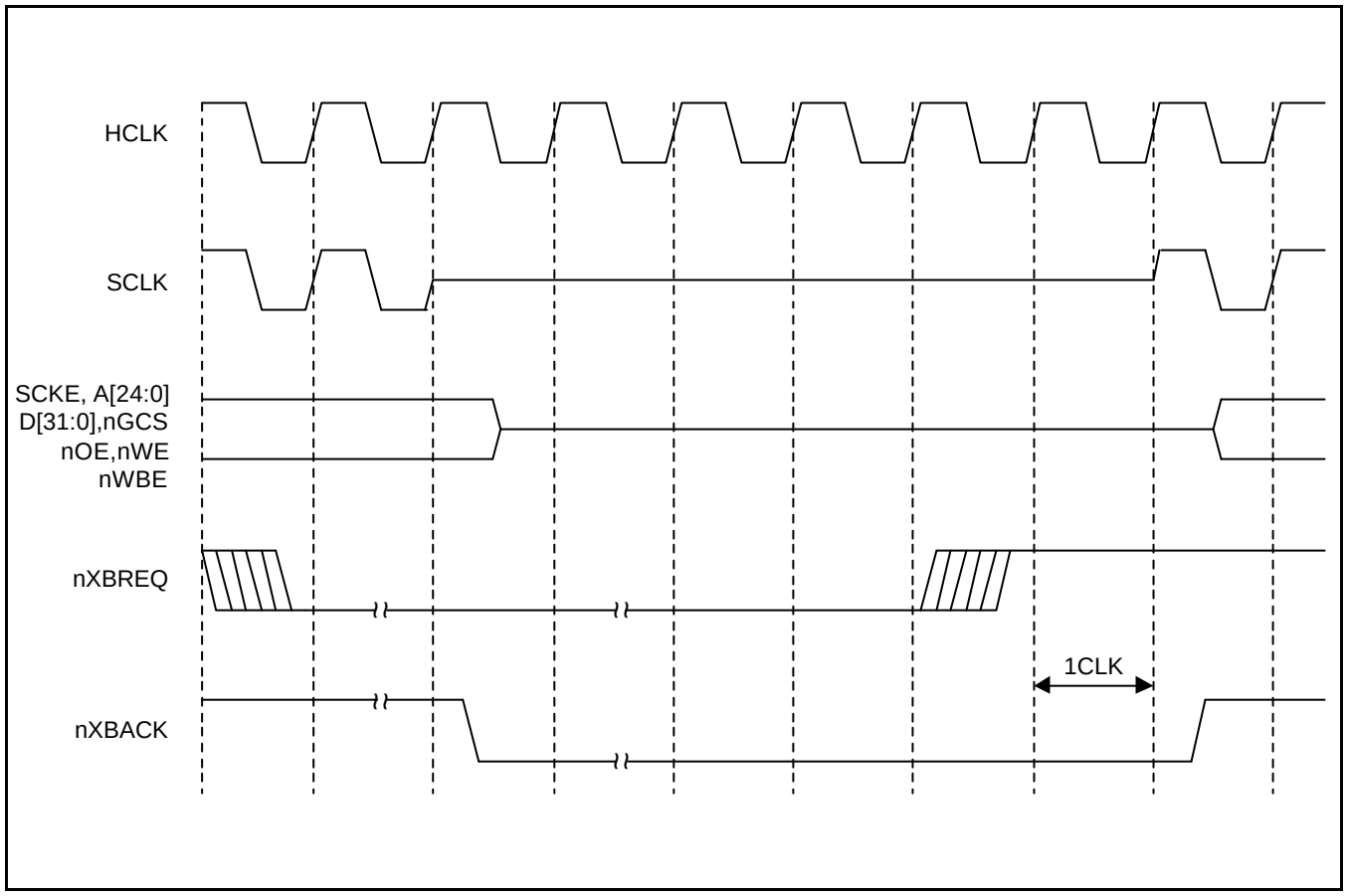


Figure 5-3. S3C2410A nXBREQ/nXBACK Timing Diagram

ROM Memory Interface Examples

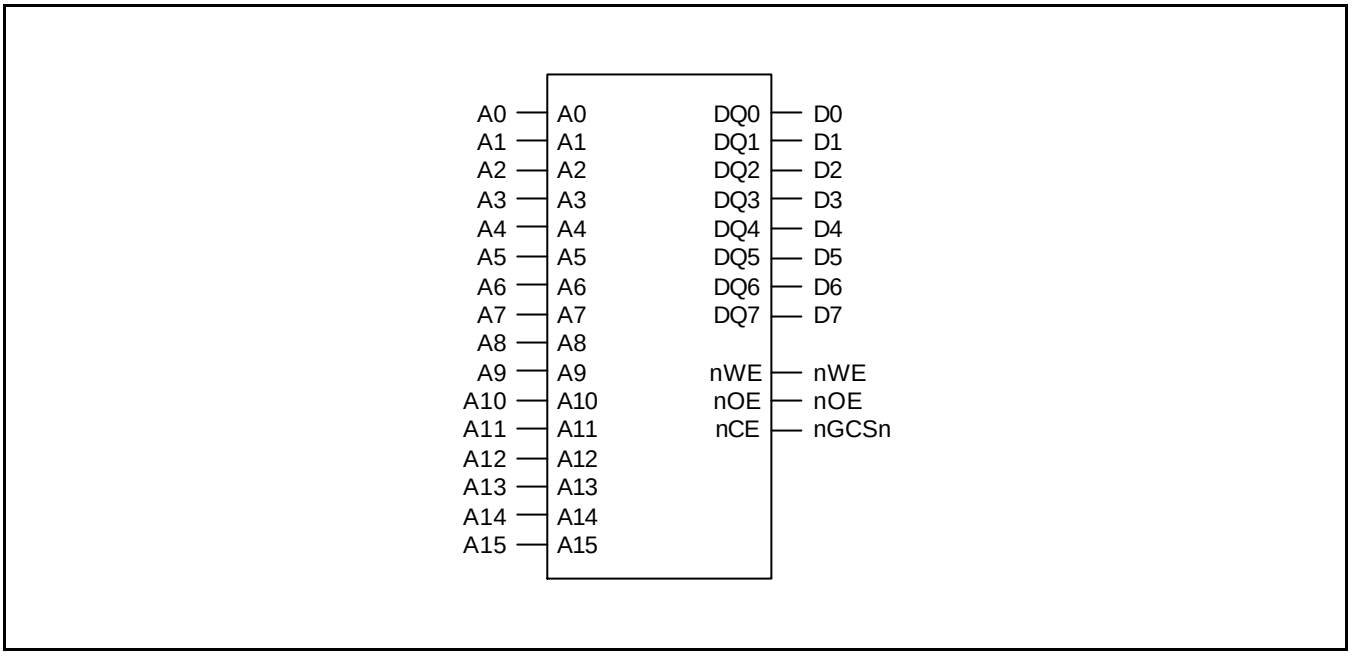


Figure 5-4. Memory Interface with 8-bit ROM

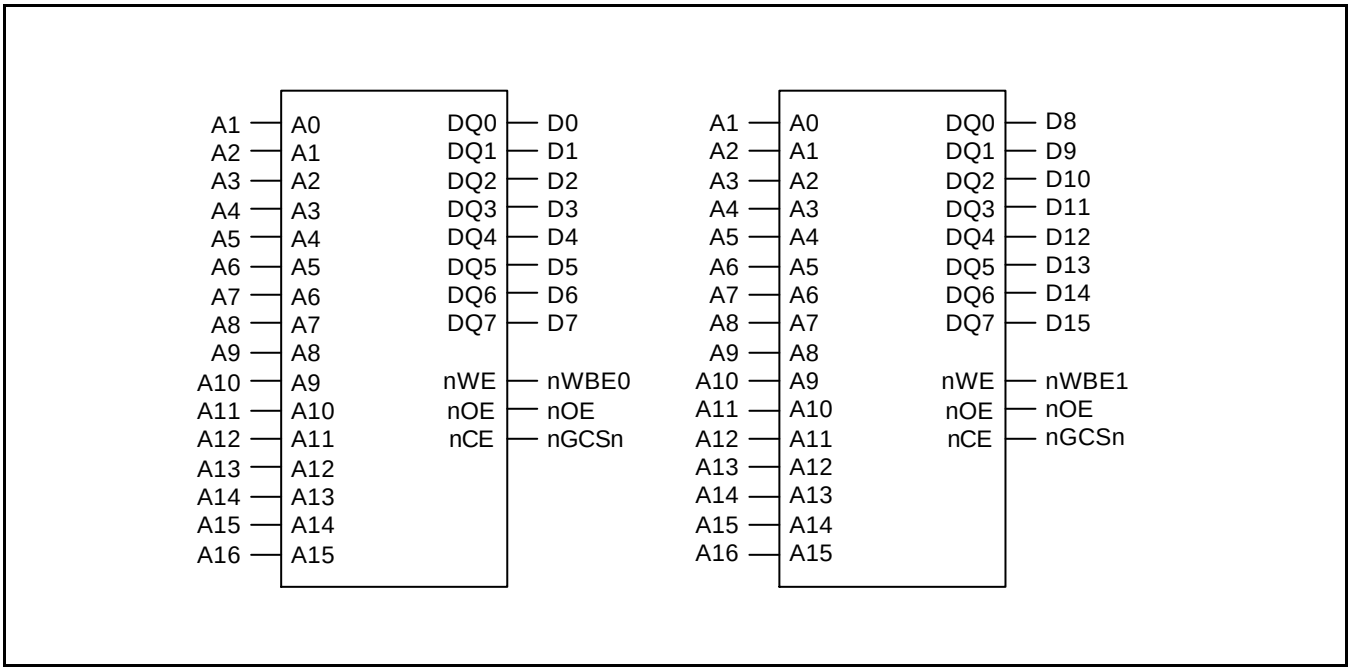


Figure 5-5. Memory Interface with 8-bit ROM × 2

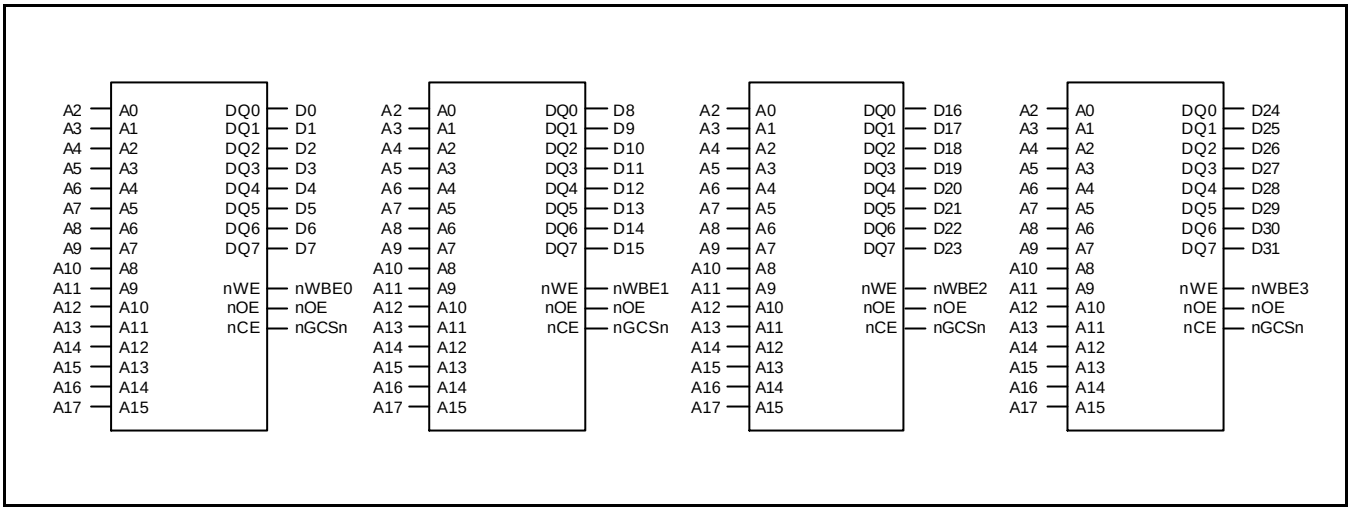


Figure 5-6. Memory Interface with 8-bit ROM ´ 4

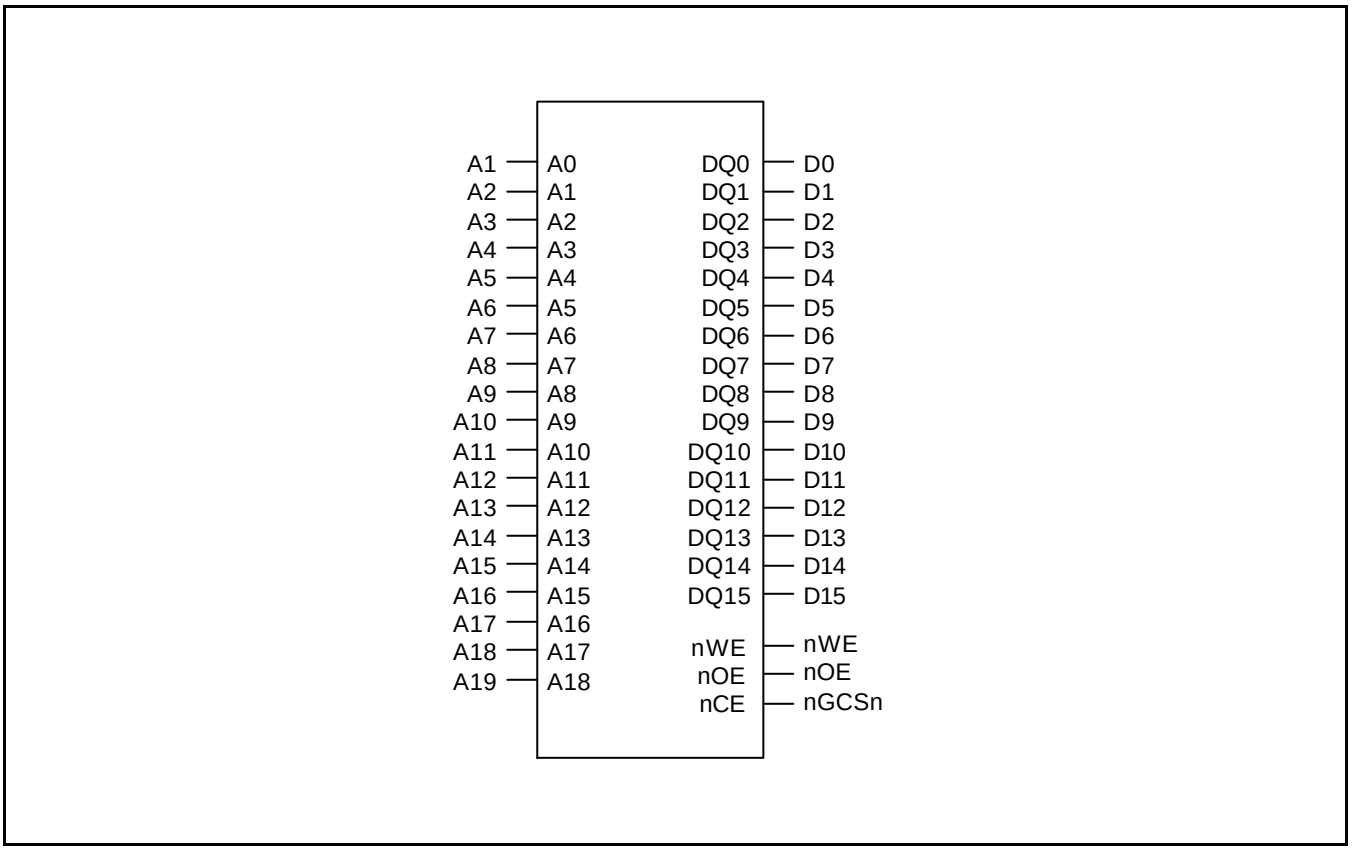


Figure 5-7. Memory Interface with 16-bit ROM

SRAM Memory Interface Examples

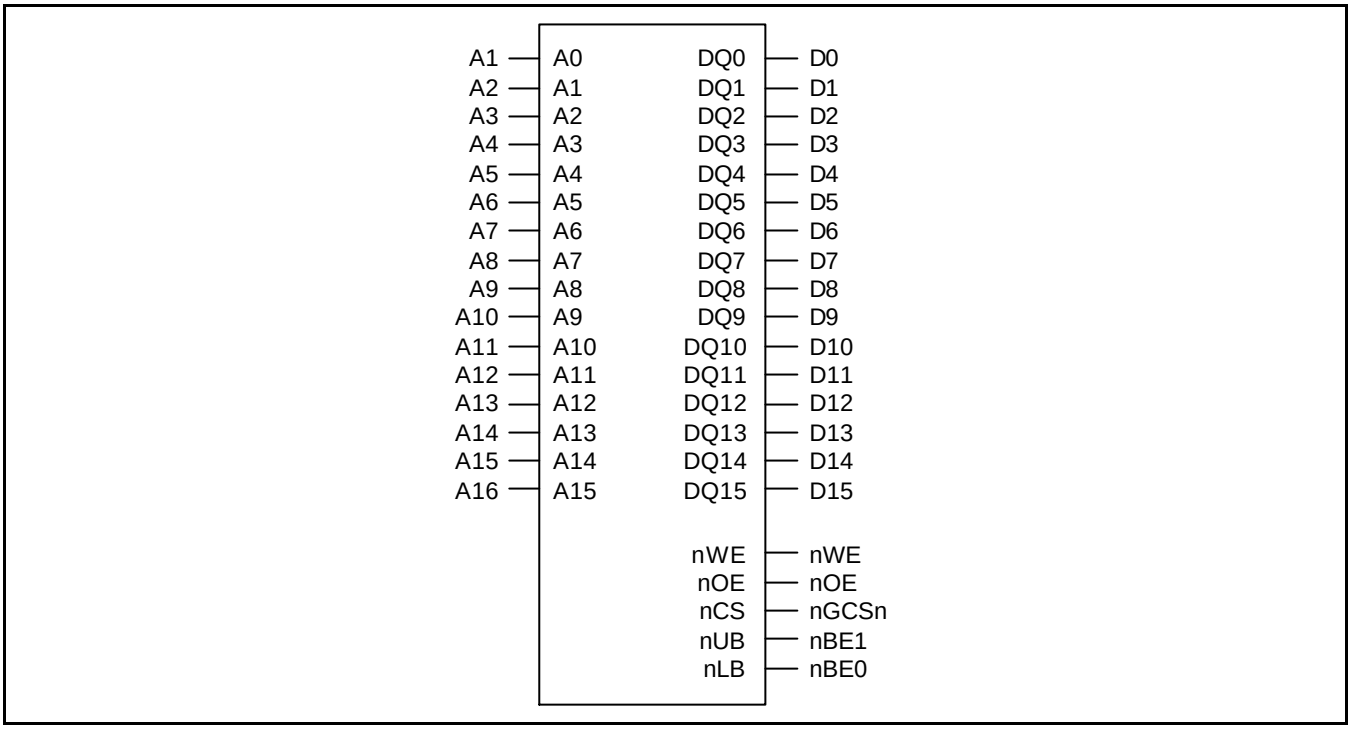


Figure 5-8. Memory Interface with 16-bit SRAM

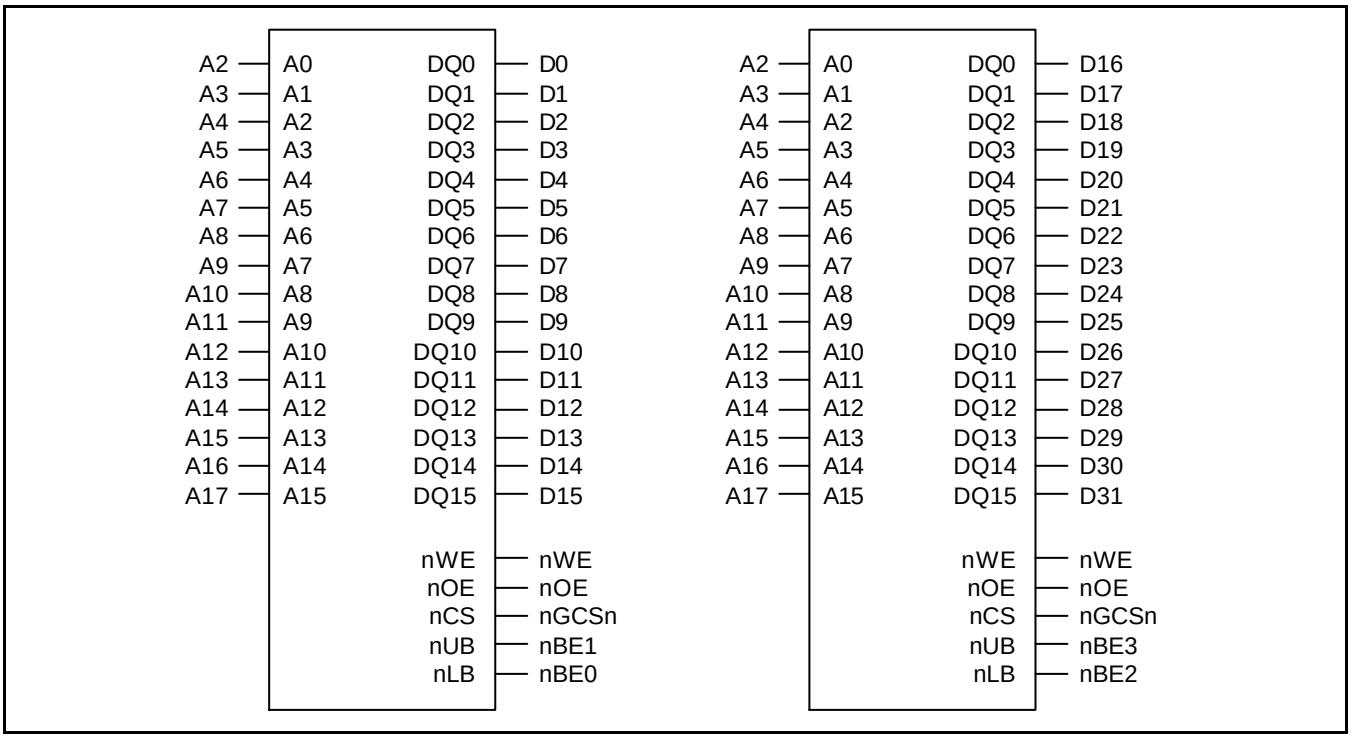


Figure 5-9. Memory Interface with 16-bit SRAM × 2

SDRAM Memory Interface Examples

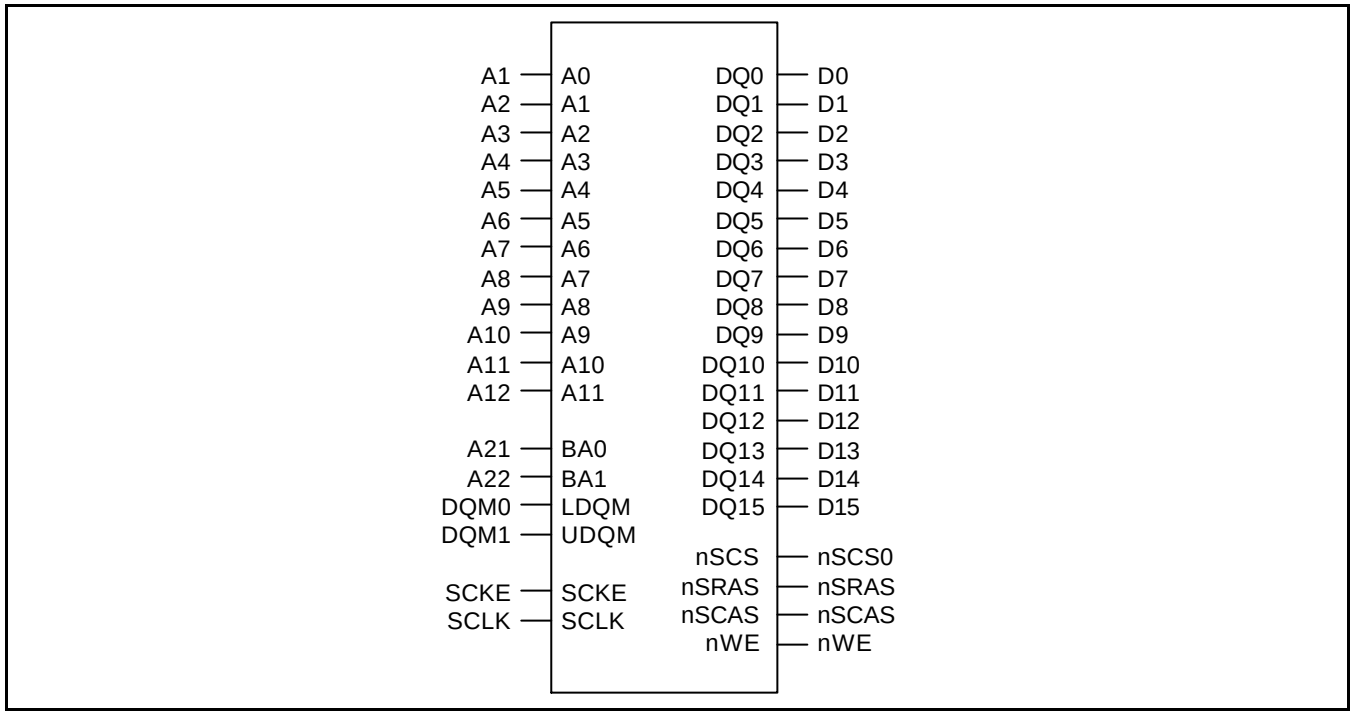


Figure 5-10. Memory Interface with 16-bit SDRAM(8MB: 1Mb ´ 16 ´ 4banks)

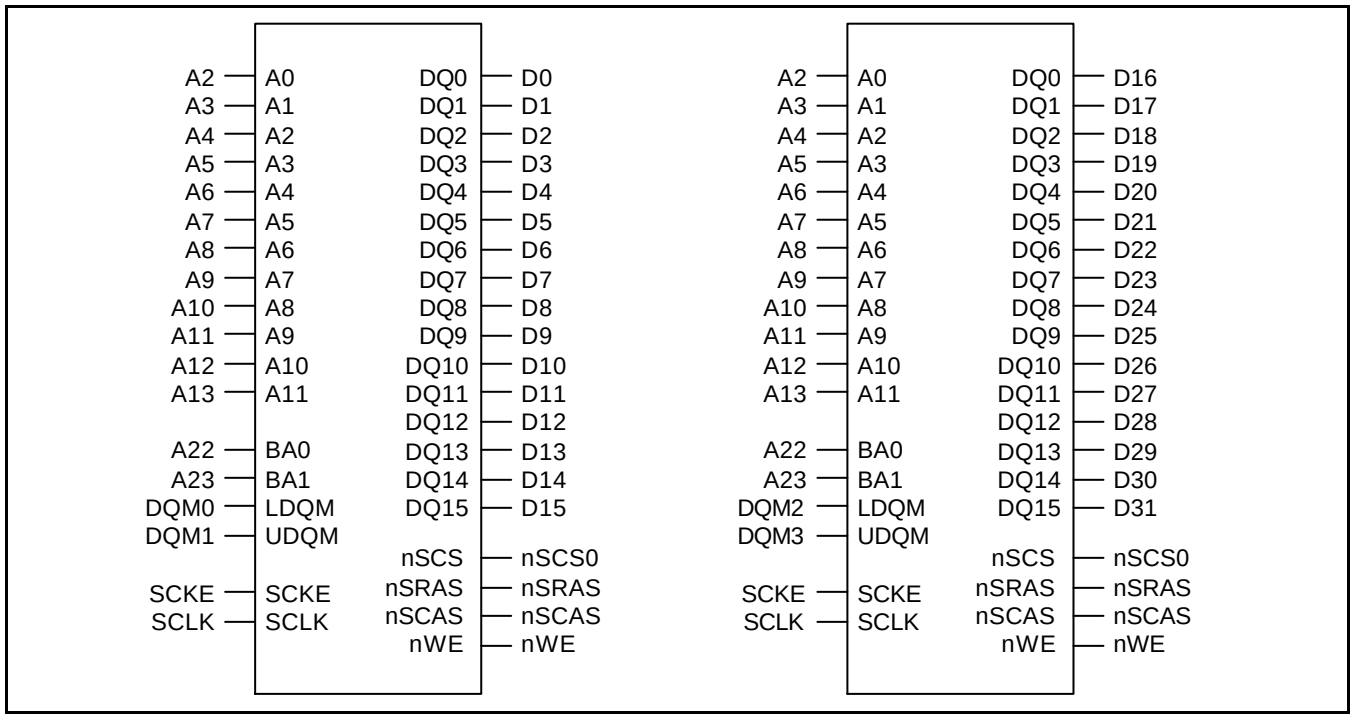


Figure 5-11. Memory Interface with 16-bit SDRAM (16MB: 1Mb ´ 16 ´ 4banks ´ 2ea)

NOTE: Refer to Table 5-2 for the Bank Address configurations of SDRAM.

PROGRAMMABLE ACCESS CYCLE

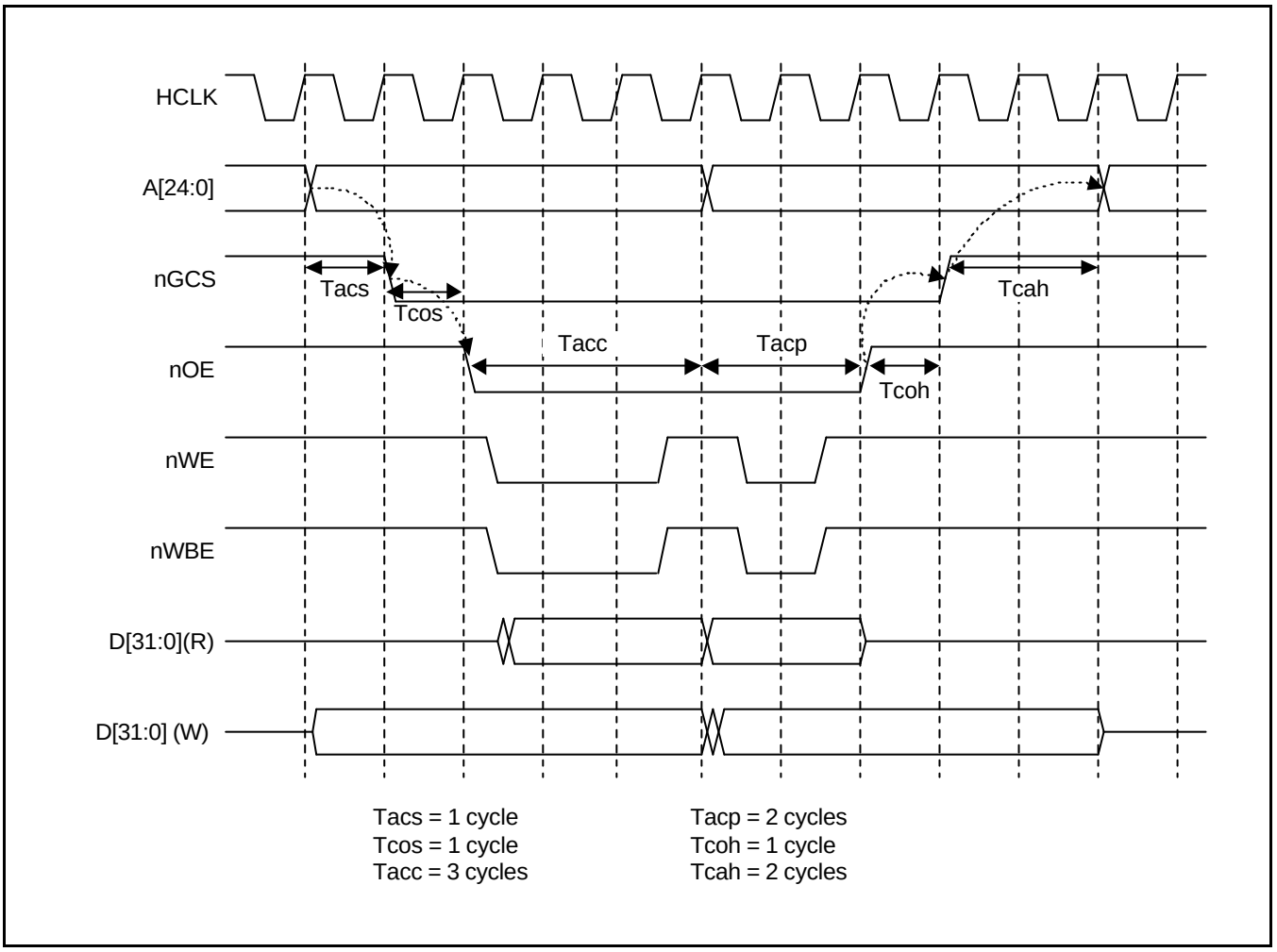


Figure 5-12. S3C2410A nGCS Timing Diagram

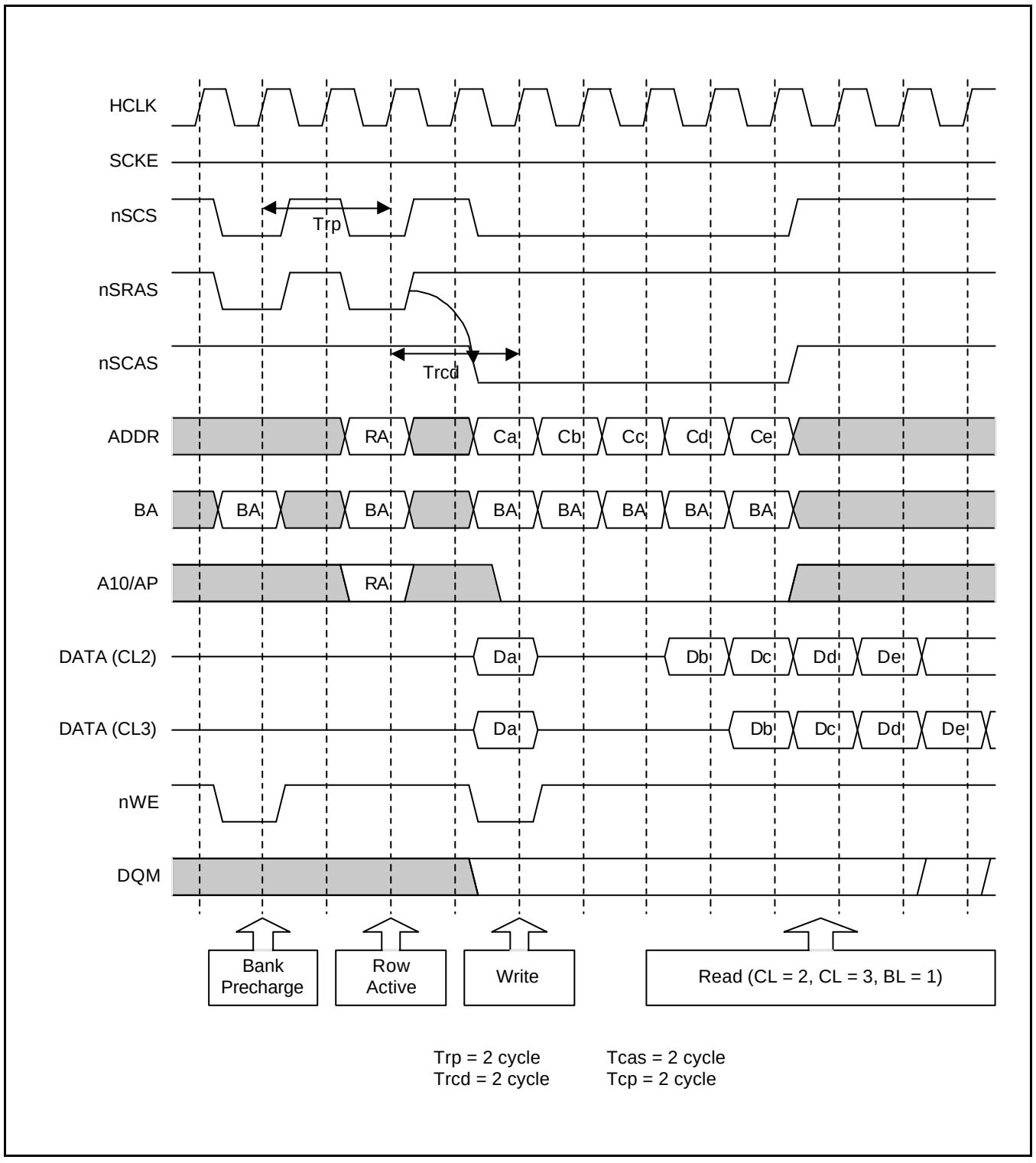


Figure 5-13. S3C2410A SDRAM Timing Diagram

BUS WIDTH & WAIT CONTROL REGISTER (BWSCON)

Register	Address	R/W	Description	Reset Value
BWSCON	0x48000000	R/W	Bus width & wait status control register	0x000000

BWSCON	Bit	Description	Initial state
ST7	[31]	Determine SRAM for using UB/LB for bank 7. 0 = Not using UB/LB (The pins are dedicated nWBE[3:0]) 1 = Using UB/LB (The pins are dedicated nBE[3:0])	0
WS7	[30]	Determine WAIT status for bank 7. 0 = WAIT disable 1 = WAIT enable	0
DW7	[29:28]	Determine data bus width for bank 7. 00 = 8-bit 01 = 16-bit, 10 = 32-bit 11 = reserved	0
ST6	[27]	Determine SRAM for using UB/LB for bank 6. 0 = Not using UB/LB (The pins are dedicated nWBE[3:0) 1 = Using UB/LB (The pins are dedicated nBE[3:0])	0
WS6	[26]	Determine WAIT status for bank 6. 0 = WAIT disable, 1 = WAIT enable	0
DW6	[25:24]	Determine data bus width for bank 6. 00 = 8-bit 01 = 16-bit, 10 = 32-bit 11 = reserved	0
ST5	[23]	Determine SRAM for using UB/LB for bank 5. 0 = Not using UB/LB (The pins are dedicated nWBE[3:0]) 1 = Using UB/LB (The pins are dedicated nBE[3:0])	0
WS5	[22]	Determine WAIT status for bank 5. 0 = WAIT disable, 1 = WAIT enable	0
DW5	[21:20]	Determine data bus width for bank 5. 00 = 8-bit 01 = 16-bit, 10 = 32-bit 11 = reserved	0
ST4	[19]	Determine SRAM for using UB/LB for bank 4. 0 = Not using UB/LB (The pins are dedicated nWBE[3:0]) 1 = Using UB/LB (The pins are dedicated nBE[3:0])	0
WS4	[18]	Determine WAIT status for bank 4. 0 = WAIT disable 1 = WAIT enable	0
DW4	[17:16]	Determine data bus width for bank 4. 00 = 8-bit 01 = 16-bit, 10 = 32-bit 11 = reserved	0
ST3	[15]	Determine SRAM for using UB/LB for bank 3. 0 = Not using UB/LB (The pins are dedicated nWBE[3:0]) 1 = Using UB/LB (The pins are dedicated nBE[3:0])	0
WS3	[14]	Determine WAIT status for bank 3. 0 = WAIT disable 1 = WAIT enable	0
DW3	[13:12]	Determine data bus width for bank 3. 00 = 8-bit 01 = 16-bit, 10 = 32-bit 11 = reserved	0
ST2	[11]	Determine SRAM for using UB/LB for bank 2. 0 = Not using UB/LB (The pins are dedicated nWBE[3:0]) 1 = Using UB/LB (The pins are dedicated nBE[3:0].)	0

BUS WIDTH & WAIT CONTROL REGISTER (BWSCON) (Continued)

BWSCON	Bit	Description	Initial state
WS2	[10]	Determine WAIT status for bank 2. 0 = WAIT disable 1 = WAIT enable	0
DW2	[9:8]	Determine data bus width for bank 2. 00 = 8-bit 01 = 16-bit, 10 = 32-bit 11 = reserved	0
ST1	[7]	Determine SRAM for using UB/LB for bank 1. 0 = Not using UB/LB (The pins are dedicated nWBE[3:0]) 1 = Using UB/LB (The pins are dedicated nBE[3:0])	0
WS1	[6]	Determine WAIT status for bank 1. 0 = WAIT disable, 1 = WAIT enable	0
DW1	[5:4]	Determine data bus width for bank 1. 00 = 8-bit 01 = 16-bit, 10 = 32-bit 11 = reserved	0
DW0	[2:1]	Indicate data bus width for bank 0 (read only). 01 = 16-bit, 10 = 32-bit The states are selected by OM[1:0] pins	-
Reserved	[0]		-

NOTES:

1. All types of master clock in this memory controller correspond to the bus clock. For example, HCLK in SRAM is the same as the bus clock, and SCLK in SDRAM is also the same as the bus clock. In this chapter (Memory Controller), one clock means one bus clock.
2. nBE[3:0] is the 'AND' signal nWBE[3:0] and nOE.

BANK CONTROL REGISTER (BANKCONN: nGCS0-nGCS5)

Register	Address	R/W	Description	Reset Value
BANKCON0	0x48000004	R/W	Bank 0 control register	0x0700
BANKCON1	0x48000008	R/W	Bank 1 control register	0x0700
BANKCON2	0x4800000C	R/W	Bank 2 control register	0x0700
BANKCON3	0x48000010	R/W	Bank 3 control register	0x0700
BANKCON4	0x48000014	R/W	Bank 4 control register	0x0700
BANKCON5	0x48000018	R/W	Bank 5 control register	0x0700

BANKCONn	Bit	Description	Initial State
Tacs	[14:13]	Address set-up time before nGCSn 00 = 0 clock 01 = 1 clock 10 = 2 clocks 11 = 4 clocks	00
Tcos	[12:11]	Chip selection set-up time before nOE 00 = 0 clock 01 = 1 clock 10 = 2 clocks 11 = 4 clocks	00
Tacc	[10:8]	Access cycle 000 = 1 clock 001 = 2 clocks 010 = 3 clocks 011 = 4 clocks 100 = 6 clocks 101 = 8 clocks 110 = 10 clocks 111 = 14 clocks **NOTE**: When nWAIT signal is used, Tacc ≥ 4 clocks.	111
Tcoh	[7:6]	Chip selection hold time after nOE 00 = 0 clock 01 = 1 clock 10 = 2 clocks 11 = 4 clocks	000
Tcah	[5:4]	Address hold time after nGCSn 00 = 0 clock 01 = 1 clock 10 = 2 clocks 11 = 4 clocks	00
Tacp	[3:2]	Page mode access cycle @ Page mode 00 = 2 clocks 01 = 3 clocks 10 = 4 clocks 11 = 6 clocks	00
PMC	[1:0]	Page mode configuration 00 = normal (1 data) 01 = 4 data 10 = 8 data 11 = 16 data	00

BANK CONTROL REGISTER (BANKCONn: nGCS6-nGCS7)

Register	Address	R/W	Description	Reset Value
BANKCON6	0x4800001C	R/W	Bank 6 control register	0x18008
BANKCON7	0x48000020	R/W	Bank 7 control register	0x18008

BANKCONn	Bit	Description	Initial State
MT	[16:15]	Determine the memory type for bank6 and bank7. 00 = ROM or SRAM 01 = Reserved (Do not use) 10 = Reserved (Do not use) 11 = Sync. DRAM	11
Memory Type = ROM or SRAM [MT=00] (15-bit)			
Tacs	[14:13]	Address set-up time before nGCS 00 = 0 clock 01 = 1 clock 10 = 2 clocks 11 = 4 clocks	00
Tcos	[12:11]	Chip selection set-up time before nOE 00 = 0 clock 01 = 1 clock 10 = 2 clocks 11 = 4 clocks	00
Tacc	[10:8]	Access cycle 000 = 1 clock 001 = 2 clocks 010 = 3 clocks 011 = 4 clocks 100 = 6 clocks 101 = 8 clocks 110 = 10 clocks 111 = 14 clocks	111
Tcoh	[7:6]	Chip selection hold time after nOE 00 = 0 clock 01 = 1 clock 10 = 2 clocks 11 = 4 clocks	00
Tcah	[5:4]	Address hold time after nGCSn 00 = 0 clock 01 = 1clock 10 = 2 clocks 11 = 4 clocks	00
Tacp	[3:2]	Page mode access cycle @ Page mode 00 = 2 clocks 01 = 3 clocks 10 = 4 clocks 11 = 6 clocks	00
PMC	[1:0]	Page mode configuration 00 = normal (1 data) 01 = 4 consecutive accesses 10 = 8 consecutive accesses 11 = 16 consecutive accesses	00
Memory Type = SDRAM [MT=11] (4-bit)			
Trcd	[3:2]	RAS to CAS delay 00 = 2 clocks 01 = 3 clocks 10 = 4 clocks	10
SCAN	[1:0]	Column address number 00 = 8-bit 01 = 9-bit 10= 10-bit	00

REFRESH CONTROL REGISTER

Register	Address	R/W	Description	Reset Value
REFRESH	0x48000024	R/W	SDRAM refresh control register	0xac0000

REFRESH	Bit	Description	Initial State
REFEN	[23]	SDRAM Refresh Enable 0 = Disable 1 = Enable (self/auto refresh)	1
TREFMD	[22]	SDRAM Refresh Mode 0 = Auto Refresh 1 = Self Refresh In self-refresh time, the SDRAM control signals are driven to the appropriate level.	0
Trp	[21:20]	SDRAM RAS pre-charge Time 00 = 2 clocks 01 = 3 clocks 10 = 4 clocks 11 = Not support	10
Tsrc	[19:18]	SDRAM Semi Row Cycle Time 00 = 4 clocks 01 = 5 clocks 10 = 6 clocks 11 = 7 clocks SDRAM's Row-Cycle time(Trc) = Tsrc + Trp If) Trp=3 clocks & Tsrc=7 clocks, Trc = 3 + 7 = 10 clocks	11
Reserved	[17:16]	Not used	00
Reserved	[15:11]	Not used	0000
Refresh Counter	[10:0]	SDRAM refresh count value. Refresh period = (2^{11}-refresh_count+1)/HCLK Ex) If refresh period is 15.6 us and HCLK is 60 MHz, the refresh count is as follows: Refresh count = 2^{11} + 1 - 60x15.6 = 1113	0

BANKSIZE REGISTER

Register	Address	R/W	Description	Reset Value
BANKSIZE	0x48000028	R/W	Flexible bank size register	0x0

BANKSIZE	Bit	Description	Initial State
BURST_EN	[7]	ARM core burst operation enable. 0 = Disable burst operation. 1 = Enable burst operation.	0
Reserved	[6]	Not used	0
SCKE_EN	[5]	SCKE enable control 0 = SDRAM SCKE disable 1 = SDRAM SCKE enable	0
SCLK_EN	[4]	SCLK is enabled only during SDRAM access cycle for reducing power consumption. When SDRAM is not accessed, SCLK becomes 'L' level. 0 = SCLK is always active. 1 = SCLK is active only during the access (recommended).	0
Reserved	[3]	Not used	0
BK76MAP	[2:0]	BANK6/7 memory map 010 = 128MB/128MB 001 = 64MB/64MB 000 = 32M/32M 111 = 16M/16M 110 = 8M/8M 101 = 4M/4M 100 = 2M/2M	010

SDRAM MODE REGISTER SET REGISTER (MRSR)

Register	Address	R/W	Description	Reset Value
MRSRB6	0x4800002C	R/W	Mode register set register bank6	xxx
MRSRB7	0x48000030	R/W	Mode register set register bank7	xxx

MRSR	Bit	Description	Initial State
Reserved	[11:10]	Not used	–
WBL	[9]	Write burst length 0: Burst (Fixed) 1: Reserved	x
TM	[8:7]	Test mode 00: Mode register set (Fixed) 01, 10 and 11: Reserved	xx
CL	[6:4]	CAS latency 000 = 1 clock, 010 = 2 clocks, 011=3 clocks Others: reserved	xxx
BT	[3]	Burst type 0: Sequential (Fixed) 1: Reserved	x
BL	[2:0]	Burst length 000: 1 (Fixed) Others: Reserved	xxx

NOTE: MRSR register must not be reconfigured while the code is running on SDRAM.

IMPORTANT NOTES

In Power_OFF mode, SDRAM has to enter SDRAM self-refresh mode.

NOTES

6 NAND FLASH CONTROLLER

OVERVIEW

Recently, a NOR flash memory gets high in price while an SDRAM and a NAND flash memory get moderate, motivating some users to execute the boot code on a NAND flash and execute the main code on an SDRAM.

S3C2410A boot code can be executed on an external NAND flash memory. In order to support NAND flash boot loader, the S3C2410A is equipped with an internal SRAM buffer called "Steppingstone". When booting, the first 4 KBytes of the NAND flash memory will be loaded into Steppingstone and the boot code loaded into Steppingstone will be executed.

Generally, the boot code will copy NAND flash content to SDRAM. Using hardware ECC generating, the NAND flash data validity will be checked. Upon the completion of the copy, the main program will be executed on the SDRAM.

FEATURES

- NAND Flash mode: Support read/erase/program NAND flash memory
- Auto boot mode : The boot code is transferred into Steppingstone after reset. After the transfer, the boot code will be executed on the Steppingstone.
- Hardware ECC generating block (for hardware generating and software correcting)
- The Steppingstone 4-KB internal SRAM buffer can be used for another purpose after NAND flash booting.

BLOCK DIAGRAM

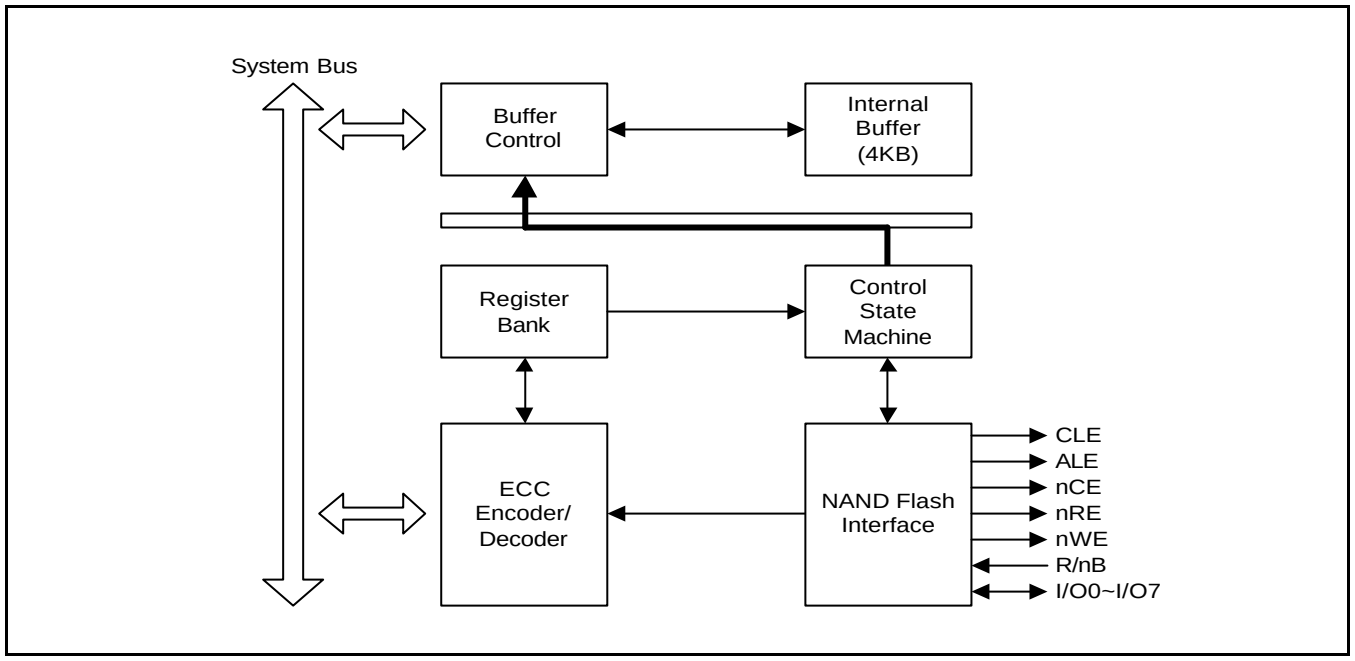


Figure 6-1. NAND Flash Controller Block Diagram

OPERATION SCHEME

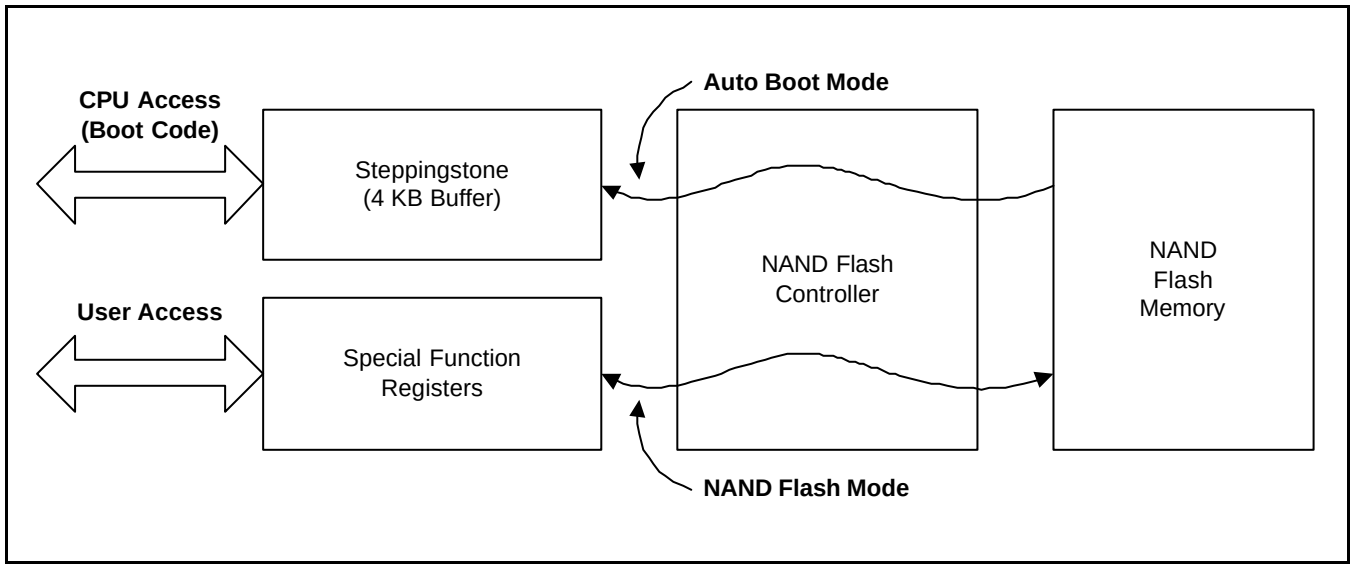


Figure 6-2. NAND Flash Operation Scheme

AUTO BOOT MODE SEQUENCE

1. Reset is completed.
2. When the auto boot mode is enabled, the first 4 KBytes of NAND flash memory is copied onto Steppingstone 4-KB internal buffer.
3. The Steppingstone is mapped to nGCS0.
4. CPU starts to execute the boot code on the Steppingstone 4-KB internal buffer.

NOTE

In the auto boot mode, ECC is not checked. So, The first 4 KBytes of NAND flash should have no bit error.

NAND FLASH MODE CONFIGURATION

1. Set NAND flash configuration by NFCONF register.
2. Write NAND flash command onto NFCMD register.
3. Write NAND flash address onto NFADDR register.
4. Read/Write data while checking NAND flash status by NFSTAT register. R/nB signal should be checked before read operation or after program operation.

NAND FLASH MEMORY TIMING

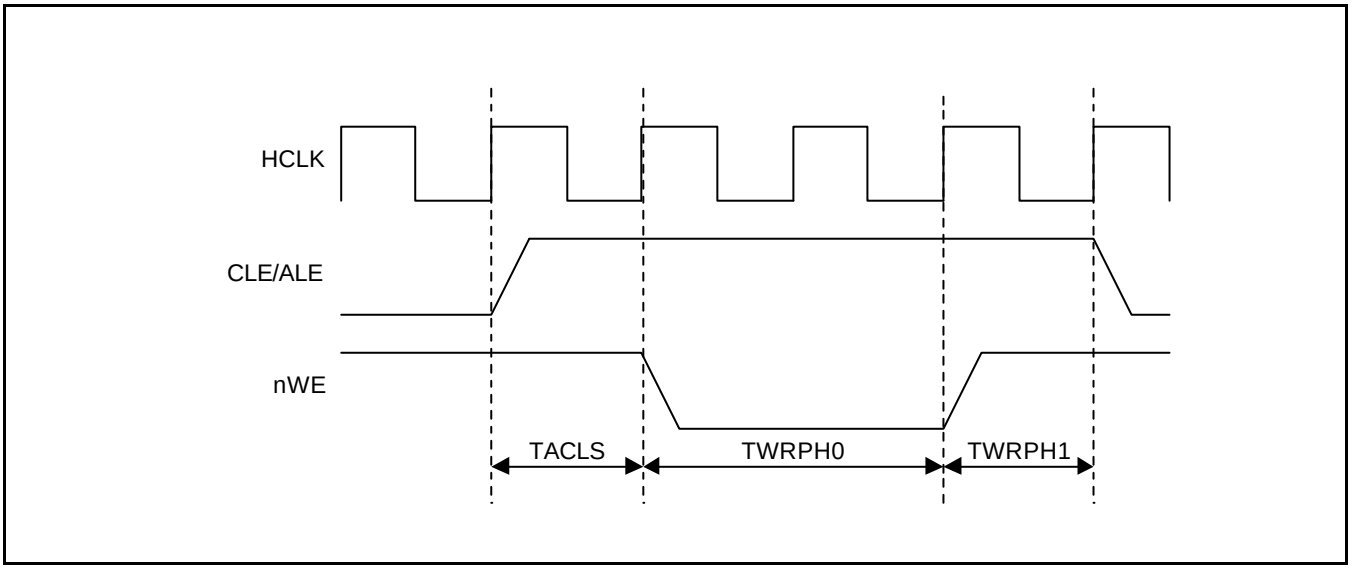


Figure 6-3. TACLS = 0, TWRPH0 = 1, TWRPH1 = 0

PIN CONFIGURATION

D[7:0] : Data/Command/Address In/Out Port (shared with the data bus)

CLE : Command Latch Enable (Output)

ALE : Address Latch Enable (Output)

nFCE : NAND Flash Chip Enable (Output)

nFRE : NAND Flash Read Enable (Output)

nFWE : NAND Flash Write Enable (Output)

R/nB : NAND Flash Ready/nBusy (Input)

BOOT AND NAND FLASH CONFIGURATIONS

1. OM[1:0] = 00b : Enable NAND flash controller auto boot mode
2. NAND flash memory page size should be 512Bytes.
3. NCON : NAND flash memory address step selection

   0 : 3 Step addressing
   1 : 4 Step addressing

512-BYTE ECC PARITY CODE ASSIGNMENT TABLE

	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0
ECC0	P64	P64’	P32	P32’	P16	P16’	P8	P8’
ECC1	P1024	P1024’	P512	P512’	P256	P256’	P128	P128’
ECC2	P4	P4’	P2	P2’	P1	P1’	P2048	P2048’

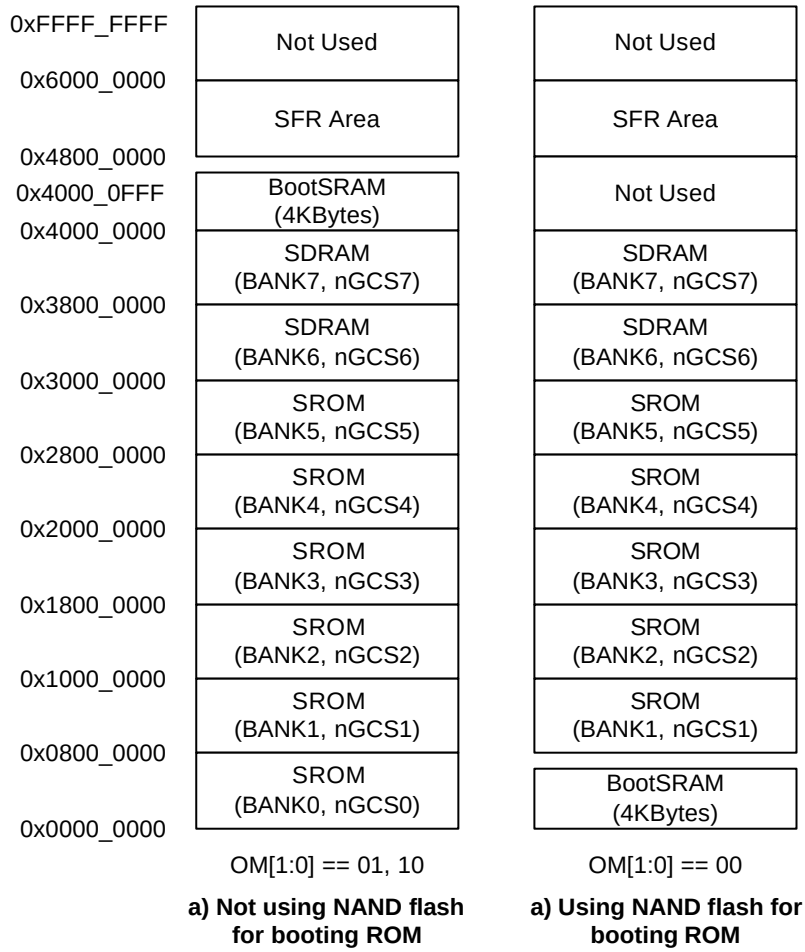
S3C2410A generates 512-Byte ECC Parity Code during Write/Read operation. ECC Parity Code consists of 3 Bytes per 512-Byte data.

24-bit ECC Parity Code = 18-bit Line parity + 6-bit Column Parity

ECC generator block executes the followings:

1. When MCU writes data to NAND, the ECC generator block generates ECC code.
2. When MCU reads data from NAND, the ECC generator block generates ECC code and users compare it with pre-written ECC code.

NAND FLASH MEMORY MAPPING



NOTE: SFR means Special Function Register.

Figure 6-4. NAND Flash Memory Mapping

SPECIAL FUNCTION REGISTERS

NAND FLASH CONFIGURATION (NFCONF) REGISTER

Register	Address	R/W	Description	Reset Value
NFCONF	0x4E000000	R/W	NAND flash configuration	–

NFCONF	Bit	Description	Initial State
Enable/Disable	[15]	NAND flash controller enable/disable 0 = Disable NAND Flash Controller 1 = Enable NAND Flash Controller After auto-boot, this bit is cleared to 0 automatically. For the access to the NAND flash memory, this bit must be set.	0
Reserved	[14:13]	Reserved	–
Initialize ECC	[12]	Initialize ECC decoder/encoder 0 : Not initialize ECC 1 : Initialize ECC (S3C2410A supports only 512-Byte ECC checking, so it is required to set ECC initialization per 512 Bytes.)	0
NAND Flash Memory chip enable	[11]	NAND flash memory nFCE control 0 : NAND flash nFCE = L (active) 1 : NAND flash nFCE = H (inactive) (After auto-boot, nFCE will be inactive.)	–
TACLS	[10:8]	CLE & ALE duration setting value (0~7) Duration = HCLK * (TACLS + 1)	0
Reserved	[7]	Reserved	–
TWRPH0	[6:4]	TWRPH0 duration setting value (0~7) Duration = HCLK * (TWRPH0 + 1)	0
Reserved	[3]	Reserved	–
TWRPH1	[2:0]	TWRPH1 duration setting value (0~7) Duration = HCLK * (TWRPH1 + 1)	0

NAND FLASH COMMAND SET (NFCMD) REGISTER

Register	Address	R/W	Description	Reset Value
NFCMD	0x4E000004	R/W	NAND flash command set register	–

NFCMD	Bit	Description	Initial State
Reserved	[15:8]	Reserved	–
Command	[7:0]	NAND flash memory command value	0x00

NAND FLASH ADDRESS SET (NFADDR) REGISTER

Register	Address	R/W	Description	Reset Value
NFADDR	0x4E000008	R/W	NAND flash address set register	–

NFADDR	Bit	Description	Initial State
Reserved	[15:8]	Reserved	–
Address	[7:0]	NAND flash memory address value	0x00

NAND FLASH DATA (NFDATA) REGISTER

Register	Address	R/W	Description	Reset Value
NFDATA	0x4E00000C	R/W	NAND flash data register	–

NFDATA	Bit	Description	Initial State
Reserved	[15:8]	Reserved	–
Data	[7:0]	NAND flash read/program data value In case of write: Programming data In case of read: Read data.	–

NAND FLASH OPERATION STATUS (NFSTAT) REGISTER

Register	Address	R/W	Description	Reset Value
NFSTAT	0x4E000010	R	NAND flash operation status	–

NFSTAT	Bit	Description	Initial State
Reserved	[16:1]	Reserved	–
RnB	[0]	NAND flash memory ready/busy status. (This signal is checked through R/nB pin.) 0 = NAND flash memory busy 1 = NAND flash memory ready to operate	–

NAND FLASH ECC (NFECC) REGISTER

Register	Address	R/W	Description	Reset Value
NFECC	0x4E000014	R	NAND flash ECC (Error Correction Code) register	–

NFECC	Bit	Description	Initial State
ECC2	[23:16]	Error Correction Code #2	–
ECC1	[15:8]	Error Correction Code #1	–
ECC0	[7:0]	Error Correction Code #0	–

Known Problems

- Problem : NAND flash controller can't be accessed by DMA.
- Solution : Instead of DMA, use LDM/STM Instructions like our boot loader example code.

7 CLOCK & POWER MANAGEMENT

OVERVIEW

The clock & power management block consists of three parts: clock control, USB control, and power control.

The Clock control logic in S3C2410A can generate the required clock signals including FCLK for CPU, HCLK for the AHB bus peripherals, and PCLK for the APB bus peripherals. The S3C2410A has two Phase Locked Loops (PLLs): one for FCLK, HCLK, and PCLK, and the other dedicated for USB block (48MHz). The clock control logic can make slow clocks without PLL and connect/disconnect the clock to each peripheral block by software, which will reduce the power consumption.

For the power control logic, the S3C2410A has various power management schemes to keep optimal power consumption for a given task. The power management block in the S3C2410A can activate four modes: NORMAL mode, SLOW mode, IDLE mode, and Power_OFF mode.

NORMAL mode: the block supplies clocks to CPU as well as all peripherals in the S3C2410A. In this mode, the power consumption will be maximized when all peripherals are turned on. It allows the user to control the operation of peripherals by software. For example, if a timer is not needed, the user can disconnect the clock to the timer to reduce power consumption.

SLOW mode: Non-PLL mode. Unlike the Normal mode, the Slow mode uses an external clock (XTIpll or EXTCLK) directly as FCLK in the S3C2410A without PLL. In this mode, the power consumption depends on the frequency of the external clock only. The power consumption due to PLL is excluded.

IDLE mode: the block disconnects clocks (FCLK) only to the CPU core while it supplies clocks to all other peripherals. The IDLE mode results in reduced power consumption due to CPU core. Any interrupt request to CPU can be woken up from the Idle mode.

Power_OFF mode: the block disconnects the internal power. So, there occurs no power consumption due to CPU and the internal logic except the wake-up logic in this mode. Activating the Power_OFF mode requires two independent power sources. One of the two power sources supplies the power for the wake-up logic. The other one supplies other internal logics including CPU, and should be controlled for power on/off. In the Power_OFF mode, the second power supply source for the CPU and internal logics will be turned off. The wakeup from Power_OFF mode can be issued by the EINT[15:0] or by RTC alarm interrupt.

FUNCTIONAL DESCRIPTION

CLOCK ARCHITECTURE

Figure 7-1 shows a block diagram of the clock architecture. The main clock source comes from an external crystal (XTIpll) or an external clock (EXTCLK). The clock generator includes an oscillator (Oscillation Amplifier), which is connected to an external crystal, and also has two PLLs (Phase-Locked-Loop), which generate the high frequency clock required in the S3C2410A.

CLOCK SOURCE SELECTION

Table 7-1 shows the relationship between the combination of mode control pins (OM3 and OM2) and the selection of source clock for the S3C2410A. The OM[3:2] status is latched internally by referring the OM3 and OM2 pins at the rising edge of nRESET.

Table 7-1. Clock Source Selection at Boot-Up

Mode OM[3:2]	MPLL State	UPLL State	Main Clock source	USB Clock Source
00	On	On	Crystal	Crystal
01	On	On	Crystal	EXTCLK
10	On	On	EXTCLK	Crystal
11	On	On	EXTCLK	EXTCLK

NOTES:

1. Although the MPLL starts just after a reset, the MPLL output (Mpll) is not used as the system clock until the software writes valid settings to the MPLLCON register. Before this valid setting, the clock from external crystal or EXTCLK source will be used as the system clock directly. Even if the user does not want to change the default value of MPLLCON register, the user should write the same value into MPLLCON register.
2. OM[3:2] is used to determine a test mode when OM[1:0] is 11.

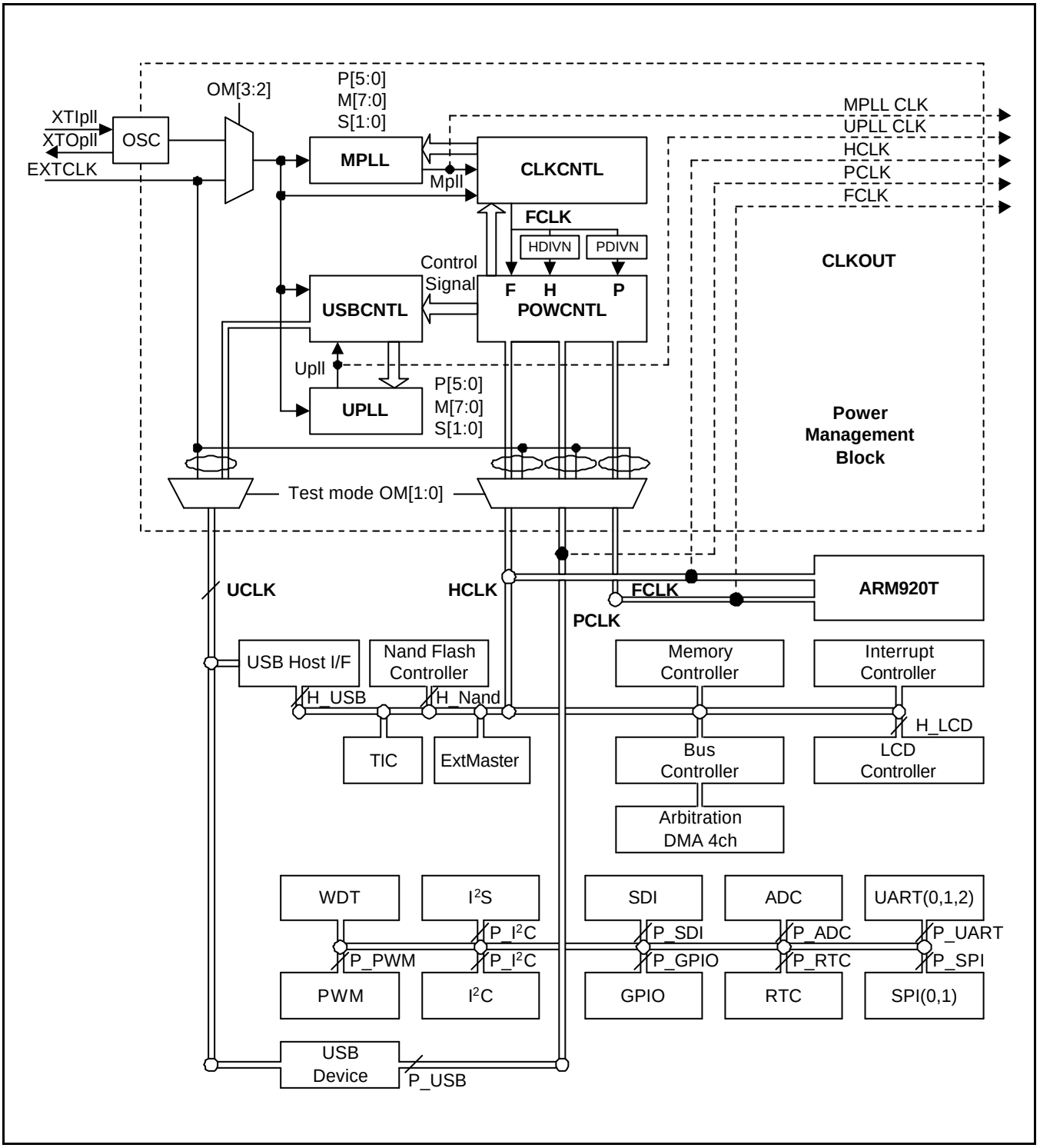


Figure 7-1. Clock Generator Block Diagram

PHASE LOCKED LOOP (PLL)

The MPLL within the clock generator, as a circuit, synchronizes an output signal with a reference input signal in frequency and phase. In this application, it includes the following basic blocks as shown in Figure 7-2: the Voltage Controlled Oscillator (VCO) to generate the output frequency proportional to input DC voltage, the divider P to divide the input frequency (Fin) by p, the divider M to divide the VCO output frequency by m which is input to Phase Frequency Detector (PFD), the divider S to divide the VCO output frequency by s which is Mpll (the output frequency from MPLL block), the phase difference detector, the charge pump, and the loop filter. The output clock frequency Mpll is related to the reference input clock frequency Fin by the following equation:

*Mpll = (m * Fin) / (p * 2^s)*
m = M (the value for divider M)+ 8, p = P (the value for divider P) + 2

The UPLL within the clock generator is the same as the MPLL in every aspect.

The following sections describe the operation of the PLL, including the phase difference detector, the charge pump, the Voltage controlled oscillator (VCO), and the loop filter.

Phase Frequency Detector (PFD)

The PFD monitors the phase difference between Fref and Fvco, and generates a control signal (tracking signal) when it detects a difference. The Fref means the reference frequency as shown in the Figure 7-2.

Charge Pump (PUMP)

The charge pump converts PFD control signals into a proportional charge in voltage across the external filter that drives the VCO.

Loop Filter

The control signal, which the PFD generates for the charge pump, may generate large excursions (ripples) each time the Fvco is compared to the Fref. To avoid overloading the VCO, a low pass filter samples and filters the high-frequency components out of the control signal. The filter is typically a single-pole RC filter with a resistor and a capacitor.

Voltage Controlled Oscillator (VCO)

The output voltage from the loop filter drives the VCO, causing its oscillation frequency to increase or decrease linearly as a function of variations in average voltage. When the Fvco matches Fref in terms of frequency as well as phase, the PFD stops sending control signals to the charge pump, which in turn stabilizes the input voltage to the loop filter. The VCO frequency then remains constant, and the PLL remains fixed onto the system clock.

Usual Conditions for PLL & Clock Generator

PLL & Clock Generator generally uses the following conditions.

Loop filter capacitance	5 pF
External X-tal frequency	10 – 20 MHz (note)
External capacitance used for X-tal	15 – 22 pF

NOTES:
1. The value could be changed.
2. FCLK must be more than three times X-tal or EXTCLK (FCLK ≥ 3X-tal or 3EXTCLK)

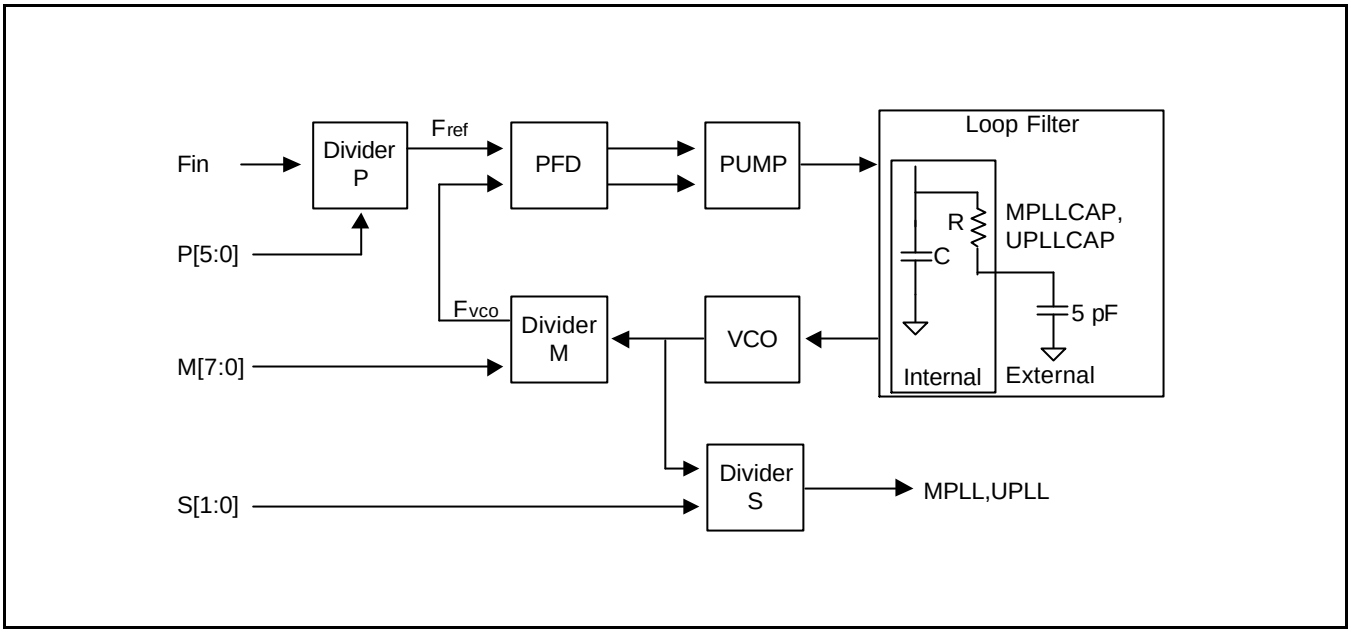
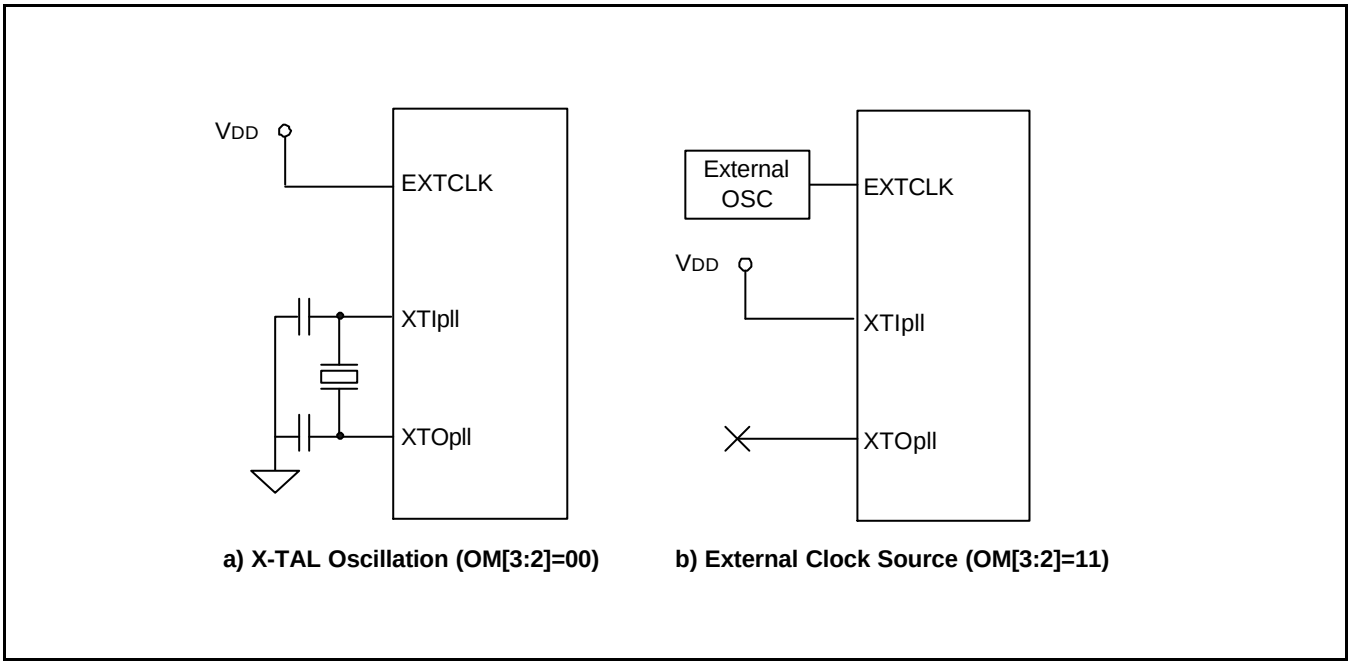


Figure 7-2. PLL (Phase-Locked Loop) Block Diagram



a) X-TAL Oscillation (OM[3:2]=00) **b) External Clock Source (OM[3:2]=11)**

Figure 7-3. Main Oscillator Circuit Examples

CLOCK CONTROL LOGIC

The clock control logic determines the clock source to be used, i.e., the PLL clock (Mpll) or the direct external clock (XTIpll or EXTCLK). When PLL is configured to a new frequency value, the clock control logic disables the FCLK until the PLL output is stabilized using the PLL locking time. The clock control logic is also activated at power-on reset and wakeup from power-down mode.

Power-On Reset (XTIpll)

Figure 7-4 shows the clock behavior during the power-on reset sequence. The crystal oscillator begins oscillation within several milliseconds. When nRESET is released after the stabilization of OSC (XTIpll) clock, the PLL starts to operate according to the default PLL configuration. However, PLL is commonly known to be unstable after power-on reset, so Fin is fed directly to FCLK instead of the Mpll (PLL output) before the software newly configures the PLLCON. Even if the user does not want to change the default value of PLLCON register after reset, the user should write the same value into PLLCON register by software.

The PLL restarts the lockup sequence toward the new frequency only after the software configures the PLL with a new frequency. FCLK can be configured as PLL output (Mpll) immediately after lock time.

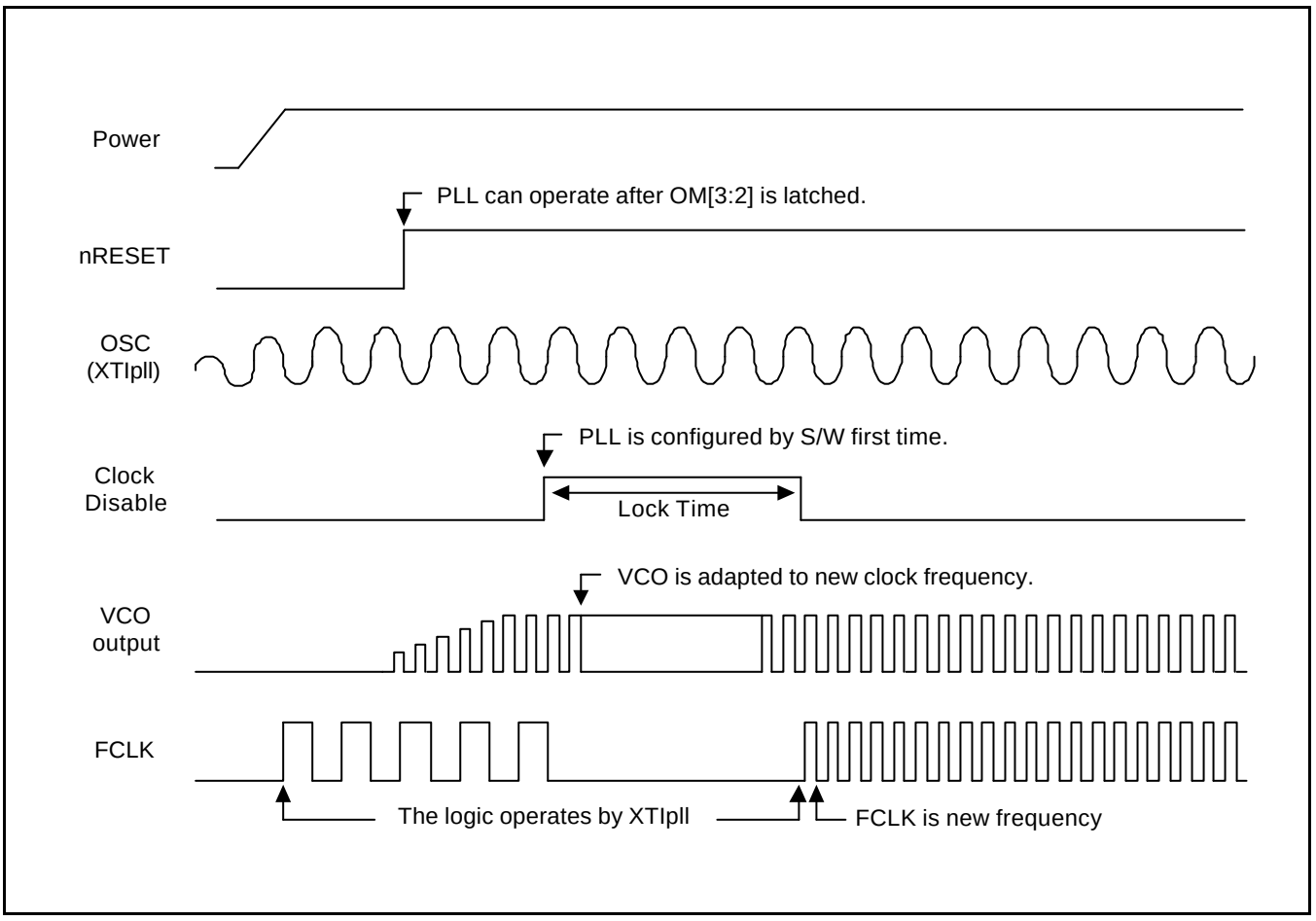


Figure 7-4. Power-On Reset Sequence (when the external clock source is a crystal oscillator)

Change PLL Settings In Normal Operation Mode

During the operation of the S3C2410A in NORMAL mode, the user can change the frequency by writing the PMS value and the PLL lock time will be automatically inserted. During the lock time, the clock is not supplied to the internal blocks in the S3C2410A. Figure 7-5 shows the timing diagram.

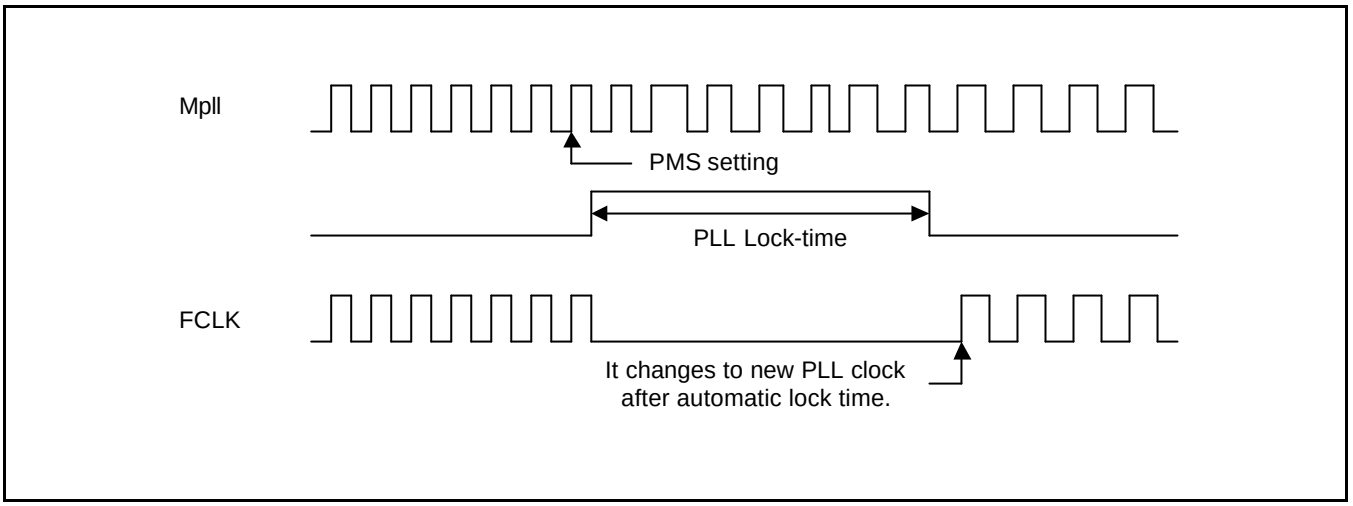


Figure 7-5. Changing Slow Clock by Setting PMS Value

USB Clock Control

USB host interface and USB device interface needs 48Mhz clock. In the S3C2410A, the USB dedicated PLL (UPLL) generates 48MHz for USB. UCLK does not fed until the PLL (UPLL) is configured.

Condition	UCLK State	UPLL State
After reset	XTlpll or EXTCLK	On
After configuring UPLL	L : during PLL lock time 48MHz: after PLL lock time	On
UPLL is turned off by CLKSLOW register	XTlpll or EXTCLK	Off
UPLL is turned on by CLKSLOW register	48MHz	On

FCLK, HCLK, and PCLK

FCLK is used by ARM920T.
HCLK is used for AHB bus, which is used by the ARM920T, the memory controller, the interrupt controller, the LCD controller, the DMA and the USB host block.
PCLK is used for APB bus, which is used by the peripherals such as WDT, IIS, I2C, PWM timer, MMC interface, ADC, UART, GPIO, RTC and SPI.

The S3C2410A supports selection of Dividing Ratio between FCLK, HLCK and PCLK. This ratio is determined by HDIVN and PDIVN of CLKDIVN control register.

HDIVN1	HDIVN	PDIVN	FCLK	HCLK	PCLK	Divide Ratio
0	0	0	FCLK	FCLK	FCLK	1 : 1 : 1 (Default)
0	0	1	FCLK	FCLK	FCLK / 2	1 : 1 : 2
0	1	0	FCLK	FCLK / 2	FCLK / 2	1 : 2 : 2
0	1	1	FCLK	FCLK / 2	FCLK / 4	1 : 2 : 4 (Recommended)
1	0	0	FCLK	FCLK / 4	FCLK / 4	1 : 4 : 4

After setting PMS value, it is required to set CLKDIVN register. The setting value of CLKDIVN will be valid after PLL lock time. The value is also available for reset and changing Power Management Mode.
The setting value can also be valid after 1.5 HCLK. Only, 1HCLK can validate the value of CLKDIVN register changed from Default (1:1:1) to other Divide Ratio (1:1:2, 1:2:2, 1:2:4 and 1:4:4)

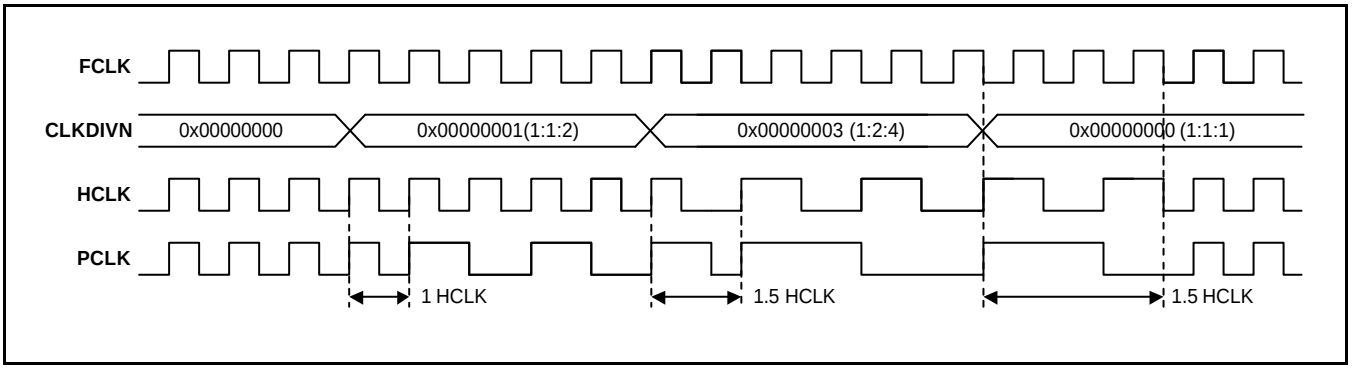


Figure 7-6. Changing CLKDIVN Register Value

NOTES

1. CLKDIVN should be set carefully not to exceed the limit of HCLK and PCLK.
2. If HDIVN = 1, the CPU bus mode has to be changed from the fast bus mode to the asynchronous bus mode using following instructions.

```
MMU_SetAsyncBusMode
    mrc  p15,0,r0,c1,c0,0
    orr  r0,r0,#R1_nF:OR:R1_iA
    mcr  p15,0,r0,c1,c0,0
```

If HDIVN=1 and the CPU bus mode is the fast bus mode, the CPU will operate by the HCLK. This feature can be used to change the CPU frequency as a half without affecting the HCLK and PCLK.

POWER MANAGEMENT

The power management block controls the system clocks by software for the reduction of power consumption in the S3C2410A. These schemes are related to PLL, clock control logics (FCLK, HCLK, and PCLK) and wakeup signals. Figure 7-7 shows the clock distribution of the S3C2410A.

The S3C2410A has four power modes. The following section describes each power management mode. The transition between the modes is not allowed freely. For available transitions among the modes, see Figure 7-8.

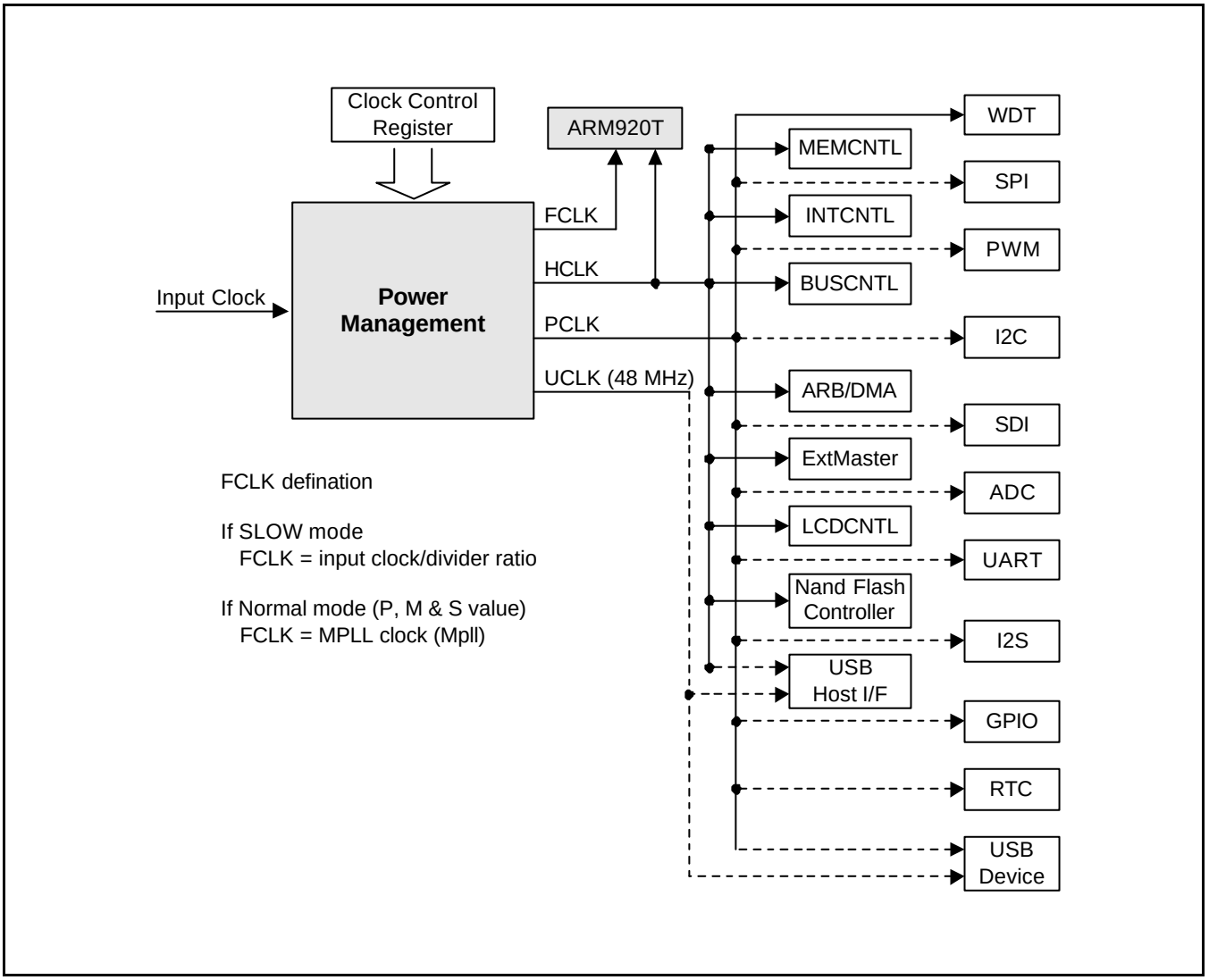


Figure 7-7. The Clock Distribution Block Diagram

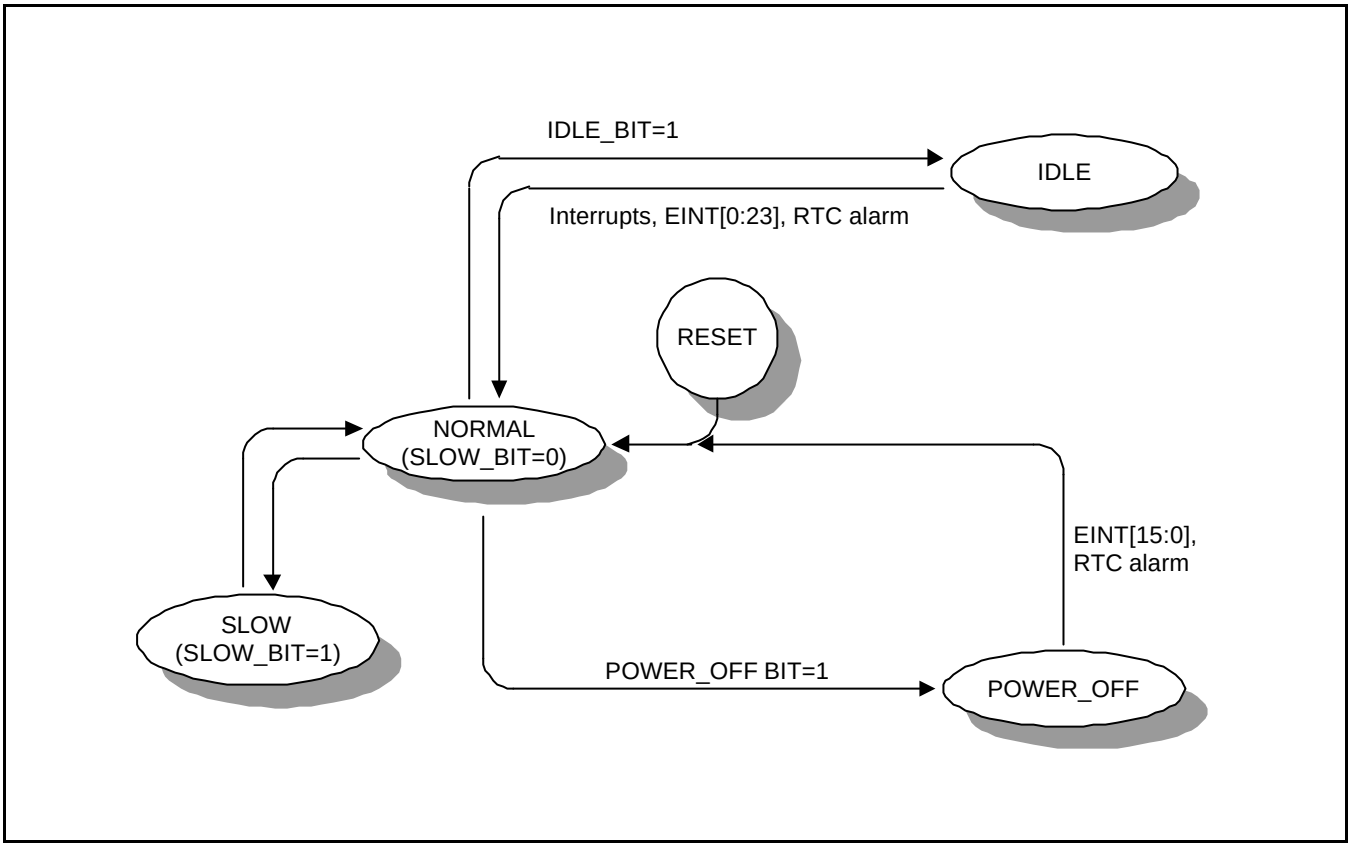


Figure 7-8. Power Management State Diagram

Table 7-2. Clock and Power State in Each Power Mode

Mode	ARM920T	AHB Modules (1) /WDT	Power Management	GPIO	32.768kHz RTC clock	APB Modules (2) & USBH/LCD/NAND
NORMAL	O	O	O	SEL	O	SEL
IDLE	X	O	O	SEL	O	SEL
SLOW	O	O	O	SEL	O	SEL
POWER_OFF	OFF	OFF	Wait for wake-up event	Previous state	O	OFF

NOTES:
1. USB host,LCD, and NAND are excluded.
2. WDT is excluded. RTC interface for CPU access is included.
3. SEL : selectable(O,X), O : enable , X : disable OFF: power is turned off

NORMAL Mode

In normal mode, all peripherals and the basic blocks including power management block, the CPU core, the bus controller, the memory controller, the interrupt controller, DMA, and the external master may operate fully. But, the clock to each peripheral, except the basic blocks, can be stopped selectively by software to reduce the power consumption.

IDLE Mode

In IDLE mode, the clock to the CPU core is stopped except the bus controller, the memory controller, the interrupt controller, and the power management block. To exit the IDLE mode, EINT[23:0], or RTC alarm interrupt, or the other interrupts should be activated. (EINT is not available until GPIO block is turned on).

SLOW Mode (Non-PLL Mode)

Power consumption can be reduced in the SLOW mode by applying a slow clock and excluding the power consumption from the PLL. The FCLK is the frequency of divide_by_n of the input clock (XTIpll or EXTCLK) without PLL. The divider ratio is determined by SLOW_VAL in the CLKSLOW control register and CLKDIVN control register.

Table 7-3. CLKSLOW and CLKDIVN Register Settings for SLOW Clock

SLOW_VAL	FCLK	HCLK		PCLK		UCLK
		1/1 Option (HDIVN = 0)	1/2 Option (HDIVN = 1)	1/1 Option (PDIVN = 0)	1/2 Option (PDIVN = 1)	
0 0 0	EXTCLK or XTIpll / 1	EXTCLK or XTIpll / 1	EXTCLK or XTIpll / 2	HCLK	HCLK / 2	48 MHz
0 0 1	EXTCLK or XTIpll / 2	EXTCLK or XTIpll / 2	EXTCLK or XTIpll / 4	HCLK	HCLK / 2	48 MHz
0 1 0	EXTCLK or XTIpll / 4	EXTCLK or XTIpll / 4	EXTCLK or XTIpll / 8	HCLK	HCLK / 2	48 MHz
0 1 1	EXTCLK or XTIpll / 6	EXTCLK or XTIpll / 6	EXTCLK or XTIpll / 12	HCLK	HCLK / 2	48 MHz
1 0 0	EXTCLK or XTIpll / 8	EXTCLK or XTIpll / 8	EXTCLK or XTIpll / 16	HCLK	HCLK / 2	48 MHz
1 0 1	EXTCLK or XTIpll / 10	EXTCLK or XTIpll / 10	EXTCLK or XTIpll / 20	HCLK	HCLK / 2	48 MHz
1 1 0	EXTCLK or XTIpll / 12	EXTCLK or XTIpll / 12	EXTCLK or XTIpll / 24	HCLK	HCLK / 2	48 MHz
1 1 1	EXTCLK or XTIpll / 14	EXTCLK or XTIpll / 14	EXTCLK or XTIpll / 28	HCLK	HCLK / 2	48 MHz

In SLOW mode, PLL will be turned off to reduce the PLL power consumption. When the PLL is turned off in the SLOW mode and the user changes power mode from SLOW mode to NORMAL mode, the PLL needs clock stabilization time (PLL lock time). This PLL stabilization time is automatically inserted by the internal logic with lock time count register. The PLL stability time will take 150us after the PLL is turned on. During PLL lock time, the FCLK becomes SLOW clock.

Users can change the frequency by enabling SLOW mode bit in CLKSLOW register in PLL on state. The SLOW clock is generated during the SLOW mode. Figure 7-9 shows the timing diagram.

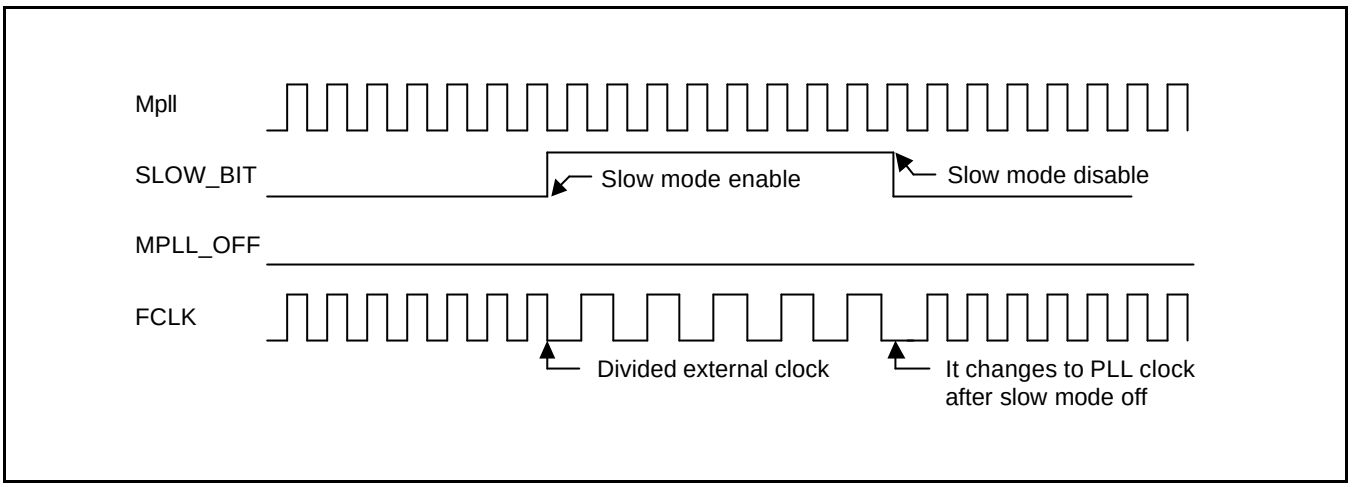


Figure 7-9. Issuing Exit_from_Slow_mode Command in PLL on State

If the user switches from SLOW mode to Normal mode by disabling the SLOW_BIT in the CLKSLOW register after PLL lock time, the frequency is changed just after SLOW mode is disabled. Figure 7-10 shows the timing diagram.

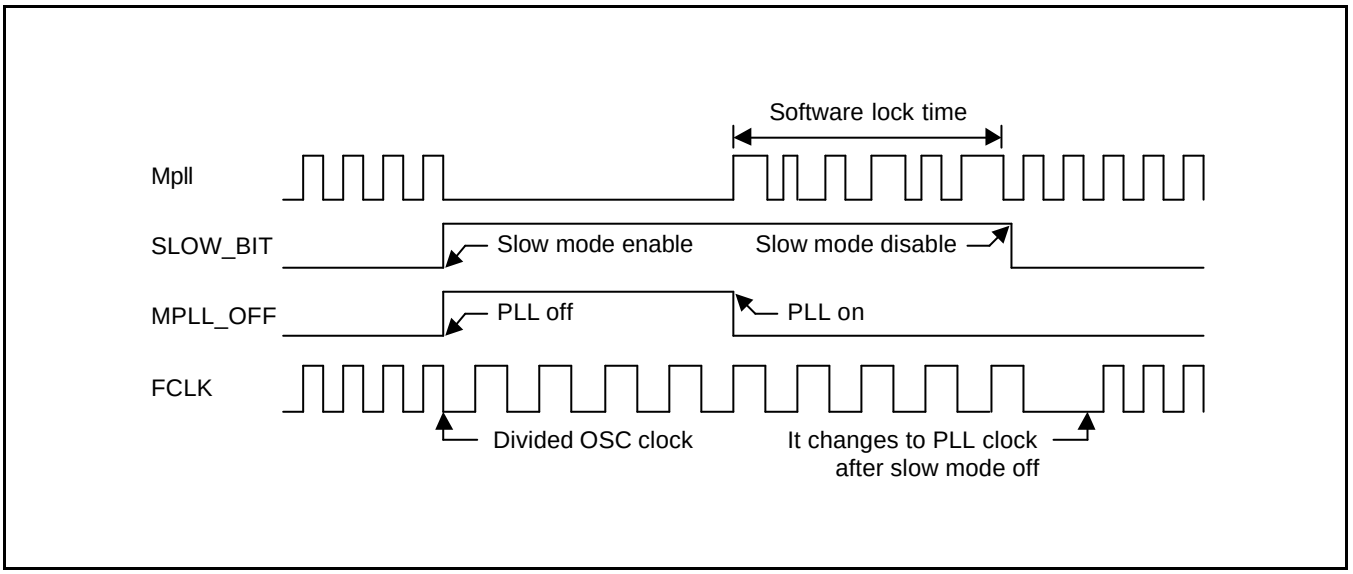


Figure 7-10. Issuing Exit_from_Slow_mode Command After Lock Time

If the user switches from SLOW mode to Normal mode by disabling SLOW_BIT and MPLL_OFF bit simultaneously in the CLKSLOW register, the frequency is changed just after the PLL lock time. Figure 7-11 shows the timing diagram.

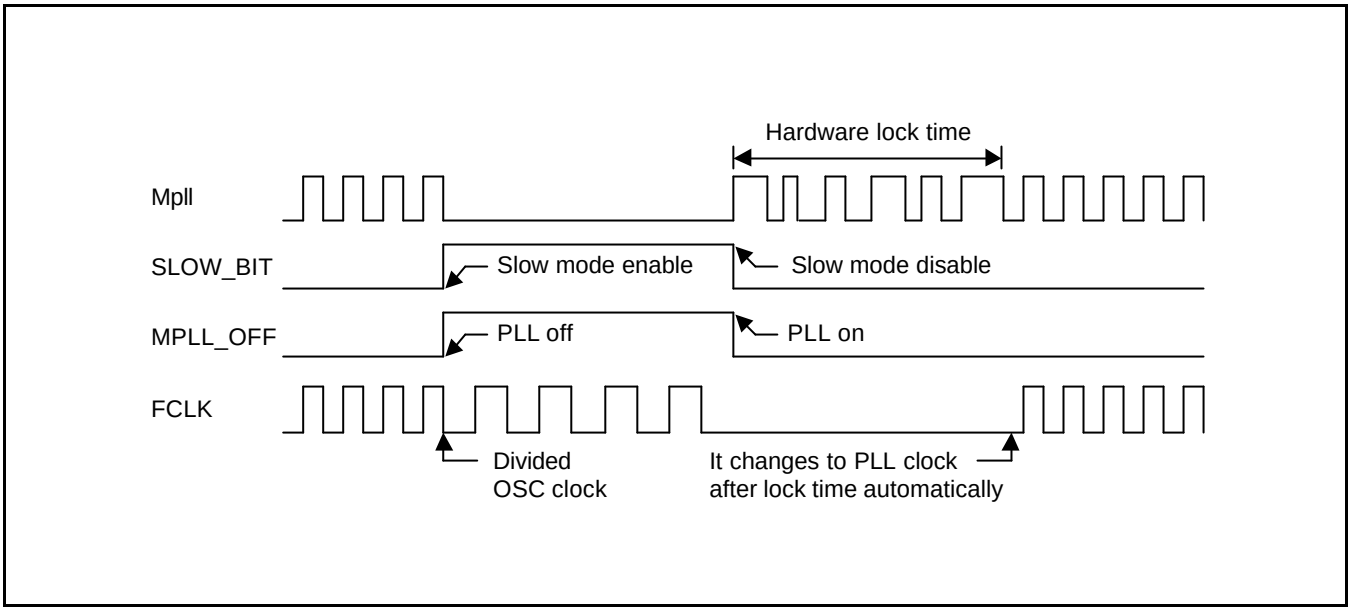


Figure 7-11. Issuing Exit_from_Slow_mode Command and the Instant PLL_on Command Simultaneously

Power_OFF Mode

The block disconnects the internal power. So, there occurs no power consumption due to CPU and the internal logic except the wake-up logic in this mode. Activating the Power_OFF mode requires two independent power sources. One of the two power sources supplies the power for the wake-up logic. The other one supplies other internal logics including CPU, and should be controlled for power on/off. In the Power_OFF mode, the second power supply source for the CPU and internal logics will be turned off. The wakeup from Power_OFF mode can be issued by the EINT[15:0] or by RTC alarm interrupt.

Procedure to Enter Power_OFF mode

1. Set the GPIO configuration adequate for Power_OFF mode.
2. Mask all interrupts in the INTMSK register.
   Configure the wake-up sources properly including RTC alarm.
   The bit of EINTMASK corresponding to the wake-up source has not to be masked in order to let the corresponding bit of SRCPND or EINTPEND set. Although a wake-up source is issued and the corresponding bit of EINTMASK is masked, the wake-up will occur and the corresponding bit of SRCPND or EINTPEND will not be set.
4. Set USB pads as suspend mode. (MISCCR[13:12]=11b)
5. Save some meaning values into GSTATUS3,4 register. These register are preserved during Power_OFF mode.
6. Configure MISCCR[1:0] for the pull-up resisters on the data bus,D[31:0]. If there is an external BUS holder, such as 74LVCH162245, turn off the pull-up resistors. If not, turn on the pull-up resistors
7. Stop LCD by clearing LCDCON1.ENVID bit.
8. Read rREFRESH and rCLKCON registers in order to fill the TLB.
9. Let SDRAM enter the self-refresh mode by setting the REFRESH[22]=1b.
10. Wait until SDRAM self-refresh is effective.
11. Set MISCCR[19:17]=111b to make SDRAM signals (SCLK0,SCLK1 and SCKE) protected during Power_OFF mode
12. Set the Power_OFF mode bit in the CLKCON register.

Procedure to Wake-up from Power_OFF mode

1. The internal reset signal will be asserted if one of the wake-up sources is issued. This reset duration is determined by the internal 16-bit counter logic and the reset assertion time is calculated as tRST = (65535 / XTAL_frequency).
2. Check GSTATUS2[2] in order to know whether or not the power-up is caused by the wake-up from Power_OFF mode.
3. Release the SDRAM signal protection by setting MISCCR[19:17]=000b.
4. Configure the SDRAM memory controller.
5. Wait until the SDRAM self-refresh is released. Mostly SDRAM needs the refresh cycle of all SDRAM row.
6. The information in GSTATUS3,4 can be used for user's own purpose because the value in GSTATUS3,4 has been preserved during Power_OFF mode.
7. – For EINT[3:0], check the SRCPND register.
   – For EINT[15:4], check the EINTPEND instead of SRCPND (SRCPND will not be set although some bits of EINTPEND are set.).
   – For alarm wake-up, check the RTC time because the RTC bit of SRCPND isn't set at the alarm wake-up.
   – If there was the nBATT_FLT assertion during POWER_OFF mode, the corresponding bit of SRCPND has been set.

Pin States in Power_OFF Mode

The pin state of the Power_OFF mode is as follows;

Pin Type	Pin Example	Pin States in Power_OFF Mode
GPIO output pin	GPB0: output	Output (GPIO data register value is used.)
GPIO input pin	GPB0: input	Input
GPIO bi-directional pin	GPG6:SPIMOSI	Input
Function output pin	nGCS0	Output (the last output level is held.)
Function input pin	nWAIT	Input

Power Control of VDDi and VDDiarm

In Power_OFF mode, only VDDi and VDDiarm will be turned off, which is controlled by PWREN pin.

If PWREN signal is active(H), VDDi and VDDiarm are supplied by an external voltage regulator. If PWREN pin is inactive (L), the VDDi and VDDiarm are turned off.

NOTE

Although VDDi, VDDiarm, VDDi_MPLL and VDDi_UPLL may be turned off, the other power pins have to be supplied.

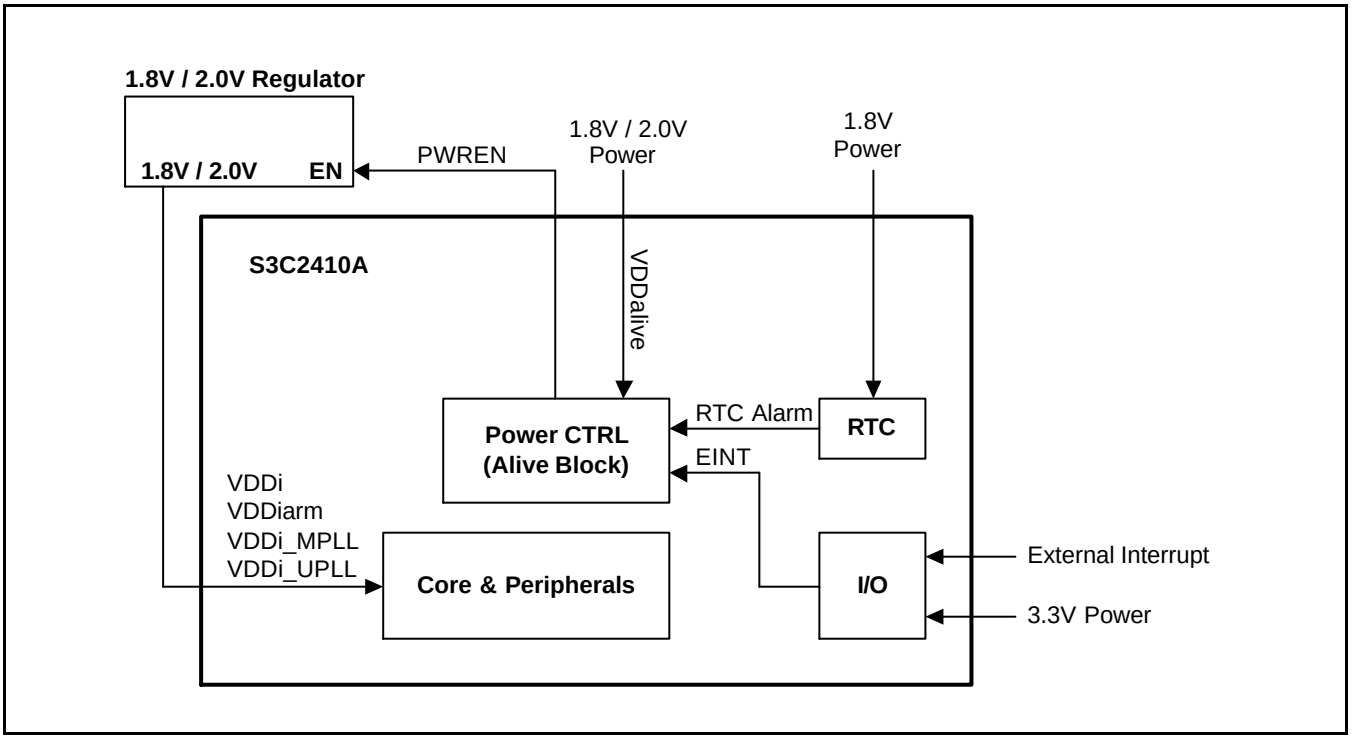


Figure 7-12. Power_OFF Mode

Signaling EINT[15:0] for Wakeup

The S3C2410A can be woken up from Power_OFF mode only if the following conditions are met.

a) Level signals (H or L) or edge signals (rising or falling or both) are asserted on EINTn input pin.

b) The EINTn pin has to be configured as EINT in the GPIO control register.

c) nBATT_FLT pin has to be H level. It is important to configure the EINTn in the GPIO control register as an external interrupt pins, considering the condition a) above.

Just after the wake-up, the corresponding EINTn pin will not be used for wakeup. This means that the pin can be used as an external interrupt request pin again.

Entering IDLE Mode

If CLKCON[2] is set to 1 to enter the IDLE mode, the S3C2410A will enter IDLE mode after some delay (until the power control logic receives ACK signal from the CPU wrapper).

PLL On/Off

The PLL can only be turned off for low power consumption in slow mode. If the PLL is turned off in any other mode, MCU operation is not guaranteed.

When the processor is in SLOW mode and tries to change its state into other state with the PLL turned on, then SLOW_BIT should be clear to move to another state after PLL stabilization

Pull-up Resistors on the Data Bus and Power_OFF Mode

In Power_OFF mode, the data bus (D[31:0] or D[15:0]) is in Hi-z state.

But, because of the characteristics of I/O pad, the data bus pull-up resistors have to be turned on for low power consumption in Power_OFF mode. D[31:0] pin pull-up resistors can be controlled by the GPIO control register (MISCCR). However, if there is an external bus holder, such as 74LVCH162245, on the data bus, turning off the data bus pull-up resistors will be reduce power consumption.

Output Port State and Power_OFF Mode

The output port should have a proper logic level in power off mode, which makes the current consumption minimized. If there is no load on an output port pin, H level is preferred. If output is L, the current will be consumed through the internal parasitic resistance; if the output is H, the current will not be consumed. For an output port, the current consumption can be reduced if the output state is H.

Battery Fault Signal(nBATT_FLT)

There are two functions in nBATT_FLT pin as follows;

— When CPU is not in Power_OFF mode, nBATT_FLT pin will cause the interrupt request. The interrupt attribute of the nBATT_FLT is L-level triggered.

— While CPU is in Power_OFF mode, assertion of the nBATT_FLT will prohibit the wake up from the power-down mode. So, Any wake-up source will be masked if nBATT_FLT is asserted, which is protecting the system malfunction of the low battery capacity

ADC Power Down

The ADC has an additional power-down bit in ADCCON. If the S3C2410A enters the Power_OFF mode, the ADC should enter its own power-down mode.

S/W Work-Around

After 'wake-up' from the Power_OFF mode by RTC_ALARM, the RTC source pending bit of the SRCPND register is not set. So, the ALARM date has to be checked after the wake-up from Power_OFF mode.

CLOCK GENERATOR & POWER MANAGEMENT SPECIAL REGISTER

LOCK TIME COUNT REGISTER (LOCKTIME)

Register	Address	R/W	Description	Reset Value
LOCKTIME	0x4C000000	R/W	PLL lock time count register	0x00FFFFFF

LOCKTIME	Bit	Description	Initial State
U_LTIME	[23:12]	UPLL lock time count value for UCLK. (U_LTIME > 150uS)	0xFFF
M_LTIME	[11:0]	MPLL lock time count value for FCLK, HCLK, and PCLK (M_LTIME > 150uS)	0xFFF

PLL Control Register (MPLLCON and UPLLCON)

Mpll = (m * Fin) / (p * 2^s)
m = (MDIV + 8), p = (PDIV + 2), s = SDIV

NOTE: Although there is the rule for choosing PLL value, we recommend only the values in the PLL value recommendation table. If you have to use another value, please contact us.

Register	Address	R/W	Description	Reset Value
MPLLCON	0x4C000004	R/W	MPLL configuration register	0x0005C080
UPLLCON	0x4C000008	R/W	UPLL configuration register	0x00028080

PLLCON	Bit	Description	Initial State
MDIV	[19:12]	Main divider control	0x5C / 0x28
PDIV	[9:4]	Pre-divider control	0x08 / 0x08
SDIV	[1:0]	Post divider control	0x0 / 0x0

NOTE: When you set MPLL&UPLL values simultaneously, set UPLL value first and then MPLL value. (Needs intervals approximately 7 NOP)

PLL VALUE SELECTION TABLE

It is not easy to find a proper PLL value. So, We recommend referring to the following PLL value recommendation table.

Input Frequency	Output Frequency	MDIV	PDIV	SDIV
12.00MHz	11.289MHz	N/A	N/A	N/A
12.00MHz	16.934MHz	N/A	N/A	N/A
12.00MHz	22.50MHz	N/A	N/A	N/A
12.00MHz	33.75MHz	82 (0x52)	2	3
12.00MHz	45.00MHz	82 (0x52)	1	3
12.00MHz	50.70MHz	161 (0xa1)	3	3
12.00Mhz	48.00Mhz (note)	120 (0x78)	2	3
12.00MHz	56.25MHz	142 (0x8e)	2	3
12.00MHz	67.50MHz	82 (0x52)	2	2
12.00MHz	79.00MHz	71 (0x47)	1	2
12.00MHz	84.75MHz	105 (0x69)	2	2
12.00MHz	90.00MHz	112 (0x70)	2	2
12.00MHz	101.25MHz	127 (0x7f)	2	2
12.00MHz	113.00MHz	105 (0x69)	1	2
12.00MHz	118.50MHz	150 (0x96)	2	2
12.00MHz	124.00MHz	116 (0x74)	1	2
12.00MHz	135.00MHz	82 (0x52)	2	1
12.00MHz	147.00MHz	90 (0x5a)	2	1
12.00MHz	152.00MHz	68 (0x44)	1	1
12.00MHz	158.00MHz	71 (0x47)	1	1
12.00MHz	170.00MHz	77 (0x4d)	1	1
12.00MHz	180.00MHz	82 (0x52)	1	1
12.00MHz	186.00MHz	85 (0x55)	1	1
12.00MHz	192.00MHz	88 (0x58)	1	1
12.00MHz	202.80MHz	161 (0xa1)	3	1
12.00MHz	266.00MHz	125 (0x7d)	1	1
12.00MHz	268.00MHz	126 (0x7e)	1	1
12.00MHz	270.00MHz	127 (0x7f)	1	1

NOTE: The 48.00MHz output is used for UPLLCON register.

CLOCK CONTROL REGISTER (CLKCON)

Register	Address	R/W	Description	Reset Value
CLKCON	0x4C00000C	R/W	Clock generator control register	0x7FFF0

CLKCON	Bit	Description	Initial State
SPI	[18]	Control PCLK into SPI block. 0 = Disable, 1 = Enable	1
IIS	[17]	Control PCLK into IIS block. 0 = Disable, 1 = Enable	1
IIC	[16]	Control PCLK into IIC block. 0 = Disable, 1 = Enable	1
ADC (&Touch Screen)	[15]	Control PCLK into ADC block. 0 = Disable, 1 = Enable	1
RTC	[14]	Control PCLK into RTC control block. Even if this bit is cleared to 0, RTC timer is alive. 0 = Disable, 1 = Enable	1
GPIO	[13]	Control PCLK into GPIO block. 0 = Disable, 1 = Enable	1
UART2	[12]	Control PCLK into UART2 block. 0 = Disable, 1 = Enable	1
UART1	[11]	Control PCLK into UART1 block. 0 = Disable, 1 = Enable	1
UART0	[10]	Control PCLK into UART0 block. 0 = Disable, 1 = Enable	1
SDI	[9]	Control PCLK into SDI interface block. 0 = Disable, 1 = Enable	1
PWMTIMER	[8]	Control PCLK into PWMTIMER block. 0 = Disable, 1 = Enable	1
USB device	[7]	Control PCLK into USB device block. 0 = Disable, 1 = Enable	1
USB host	[6]	Control HCLK into USB host block. 0 = Disable, 1 = Enable	1
LCDC	[5]	Control HCLK into LCDC block. 0 = Disable, 1 = Enable	1
NAND Flash Controller	[4]	Control HCLK into NAND Flash Controller block. 0 = Disable, 1 = Enable	1
POWER_OFF	[3]	Control Power Off mode of S3C2410. 0 = Disable, 1 = Transition to Power_OFF mode	0
IDLE BIT	[2]	Enter IDLE mode. This bit is not cleared automatically. 0 = Disable, 1 = Transition to IDLE mode	0
Reserved	[1]	Reserved	0
SM_BIT	[0]	SPECIAL mode. '0' is recommended normally. This bit can be used to enter SPECIAL mode in only the special condition, OM3=1 & wake-up by nRESET. Please contact us to use this bit.	0

CLOCK SLOW CONTROL (CLKSLOW) REGISTER

Register	Address	R/W	Description	Reset Value
CLKSLOW	0x4C000010	R/W	Slow clock control register	0x00000004

CLKSLOW	Bit	Description	Initial State
UCLK_ON	[7]	0: UCLK ON (UPLL is also turned on and the UPLL lock time is inserted automatically.) 1: UCLK OFF (UPLL is also turned off.)	0
Reserved	[6]	Reserved	–
MPLL_OFF	[5]	0 : PLL is turned on. After PLL stabilization time (minimum 150us), SLOW_BIT can be cleared to 0. 1 : PLL is turned off. PLL is turned off only when SLOW_BIT is 1.	0
SLOW_BIT	[4]	0 : FCLK = Mpll (MPLL output) 1: SLOW mode FCLK = input clock / (2 x SLOW_VAL) (SLOW_VAL > 0) FCLK = input clock (SLOW_VAL = 0) input clock = XTIpll or EXTCLK	0
Reserved	[3]	–	–
SLOW_VAL	[2:0]	The divider value for the slow clock when SLOW_BIT is on.	0x4

CLOCK DIVIDER CONTROL (CLKDIVN) REGISTER

Register	Address	R/W	Description	Reset Value
CLKDIVN	0x4C000014	R/W	Clock divider control register	0x00000000

CLKDIVN	Bit	Description	Initial State
HDIVN1	[2]	Special bus clock ratio available. (1:4:4) 0: Reserved 1: HCLK has the clock same as the FCLK/4. PCLK has the clock same as the FCLK/4. **Note:** If this bit is "0b1", HDIVN and PDIVN must be set "0b0".	0
HDIVN	[1]	0: HCLK has the clock same as the FCLK. 1: HCLK has the clock same as the FCLK/2.	0
PDIVN	[0]	0: PCLK has the clock same as the HCLK. 1: PCLK has the clock same as the HCLK/2.	0

8 DMA

OVERVIEW

The S3C2410A supports four-channel DMA controller that is located between the system bus and the peripheral bus. Each channel of DMA controller can perform data movements between devices in the system bus and/or peripheral bus with no restrictions. In other words, each channel can handle the following four cases: 1) both source and destination are in the system bus, 2) the source is in the system bus while the destination is in the peripheral bus, 3) the source is in the peripheral bus while the destination is in the system bus, and 4) both source and destination are in the peripheral bus.

The main advantage of the DMA is that it can transfer the data without CPU intervention. The operation of DMA can be initiated by software, or requests from internal peripherals or external request pins.

DMA REQUEST SOURCES

Each channel of the DMA controller can select one of DMA request source among four DMA sources if H/W DMA request mode is selected by DCON register. (Note that if S/W request mode is selected, this DMA request sources have no meaning at all.) Table 8-1 shows four DMA sources for each channel.

Table 8-1. DMA Request Sources for Each Channel

	Source0	Source1	Source2	Source3	Source4
Ch-0	nXDREQ0	UART0	SDI	Timer	USB device EP1
Ch-1	nXDREQ1	UART1	I2SSDI	SPI0	USB device EP2
Ch-2	I2SSDO	I2SSDI	SDI	Timer	USB device EP3
Ch-3	UART2	SDI	SPI1	Timer	USB device EP4

Here, nXDREQ0 and nXDREQ1 represent two external sources(External Devices), and I2SSDO and I2SSDI represent IIS transmitting and receiving, respectively.

DMA OPERATION

DMA uses three-state FSM (finite state machine) for its operation, which is described in the three following steps:

State-1. As an initial state, the DMA waits for a DMA request. If it comes, it goes to state-2. At this state, DMA ACK and INT REQ are 0.

State-2. In this state, DMA ACK becomes 1 and the counter (CURR_TC) is loaded from DCON[19:0] register. Note that the DMA ACK remains 1 until it is cleared later.

State-3. In this state, sub-FSM handling the atomic operation of DMA is initiated. The sub-FSM reads the data from the source address and then writes it to destination address. In this operation, data size and transfer size (single or burst) are considered. This operation is repeated until the counter (CURR_TC) becomes 0 in Whole service mode, while performed only once in Single service mode. The main FSM (this FSM) counts down the CURR_TC when the sub-FSM finishes each of atomic operation. In addition, this main FSM asserts the INT REQ signal when CURR_TC becomes 0 and the interrupt setting of DCON[29] register is set to 1. In addition, it clears DMA ACK if one of the following conditions is met.
1) CURR_TC becomes 0 in the Whole service mode
2) Atomic operation finishes in the Single service mode.

Note that in the Single service mode, these three states of main FSM are performed and then stops, and waits for another DMA REQ. And if DMA REQ comes in, all three states are repeated. Therefore, DMA ACK is asserted and then deasserted for each atomic transfer. In contrast, in the Whole service mode, main FSM waits at state-3 until CURR_TC becomes 0. Therefore, DMA ACK is asserted during all the transfers and then deasserted when TC reaches 0.

However, INT REQ is asserted only if CURR_TC becomes 0 regardless of the service mode (Single service mode or Whole service mode).

EXTERNAL DMA DREQ/DACK PROTOCOL

There are three types of external DMA request/acknowledge protocols (Single service Demand, Single service Handshake and Whole service Handshake mode). Each type defines how the signals like DMA request and acknowledge are related to these protocols.

Basic DMA Timing

The DMA service means performing paired Reads and Writes cycles during DMA operation, which can make one DMA operation. Figure 8-1 shows the basic Timing in the DMA operation of the S3C2410A.

— The setup time and the delay time of XnXDREQ and XnXDACK are the same in all the modes.

— If the completion of XnXDREQ meets its setup time, it is synchronized twice and then XnXDACK is asserted.

— After assertion of XnXDACK, DMA requests the bus and if it gets the bus it performs its operations. XnXDACK is deasserted when DMA operation is completed.

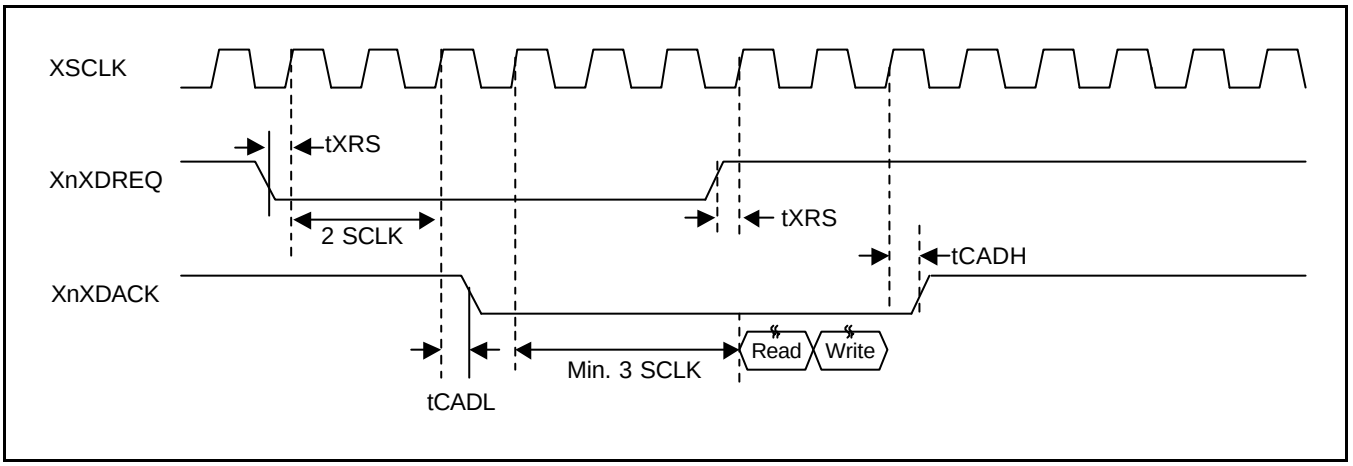


Figure 8-1. Basic DMA Timing Diagram

Table 8-2. DMA Controller Module Signal Timing Constants

(V_{DD} = 1.8 V ± 0.15 V/2.0V ± 0.1V, T_A = -40 to 85 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ.	Max	Unit
eXternal Request Setup	t_{XRS}	2	–	6 / 5	ns
aCcess to Ack Delay when Low transition	t_{CADL}	9	–	11 / 10	ns
aCcess to Ack Delay when High transition	t_{CADH}	9	–	11 / 10	ns

Demand/Handshake Mode Comparison

Demand and Handshake modes are related to the protocol between XnXDREQ and XnXDACK. Figure 8-2 shows the differences between the two modes.

At the end of one transfer (Single/Burst transfer), DMA checks the state of double-synched XnXDREQ.

Demand Mode

— If XnXDREQ remains asserted, the next transfer starts immediately. Otherwise it waits for XnXDREQ to be asserted.

Handshake Mode

— If XnXDREQ is deasserted, DMA deasserts XnXDACK in 2cycles. Otherwise it waits until XnXDREQ is deasserted.

CAUTION

XnXDREQ has to be asserted (low) only after the deassertion (high) of XnXDACK.

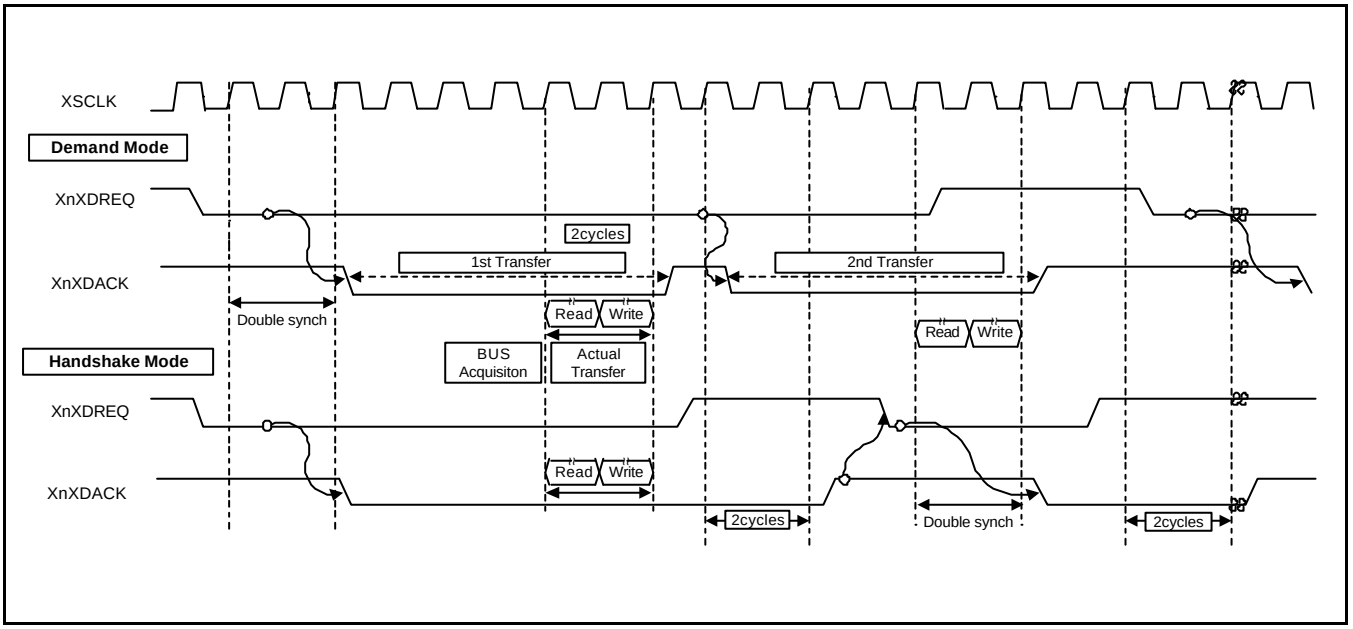


Figure 8-2. Demand/Handshake Mode Comparison

Transfer Size

— There are two different transfer sizes; unit and Burst 4.

— DMA holds the bus firmly during the transfer of the chunk of data. Thus, other bus masters cannot get the bus.

Burst 4 Transfer Size

Four sequential Reads and Writes are performed respectively in the Burst 4 Transfer.

NOTE

Unit Transfer size: One read and one write are performed.

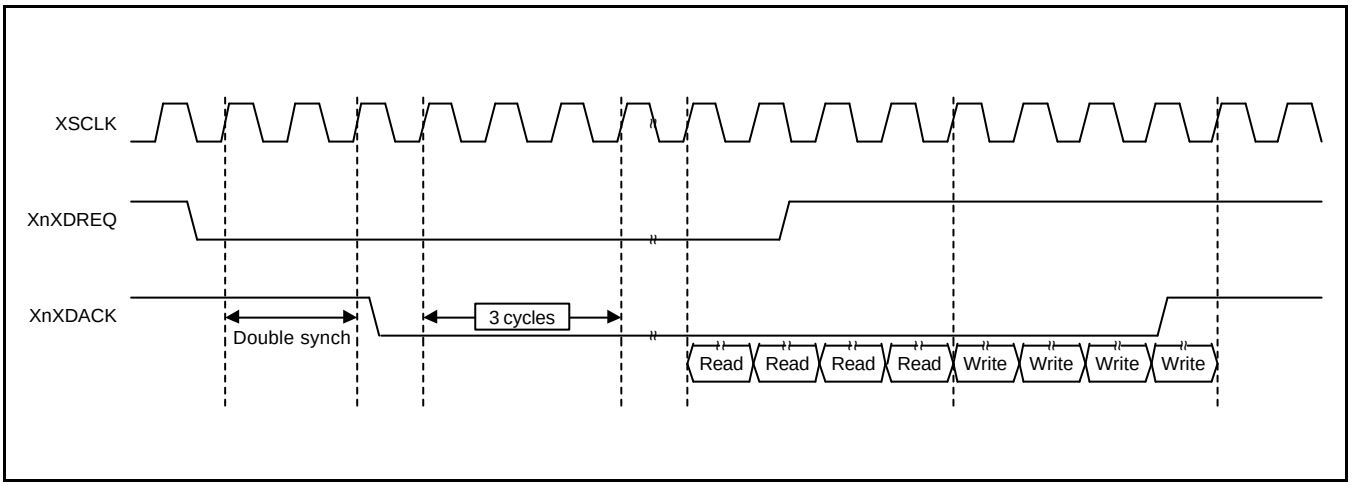


Figure 8-3. Burst 4 Transfer Size

EXAMPLES

Single service in Demand Mode with Unit Transfer Size

The assertion of XnXDREQ is need for every unit transfer (Single service mode). The operation continues while the XnXDREQ is asserted (Demand mode), and one pair of Read and Write (Single transfer size) is performed.

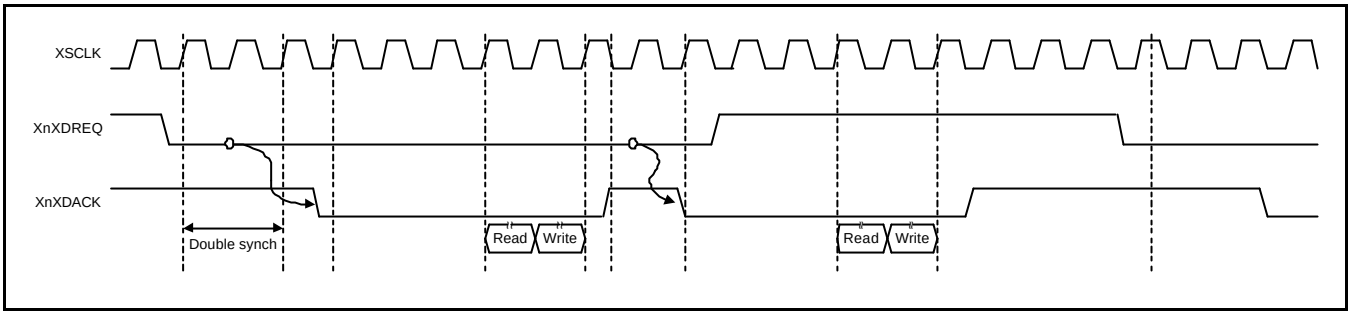


Figure 8-4. Single service in Demand Mode with Unit Transfer Size

Single service in Handshake Mode with Unit Transfer Size

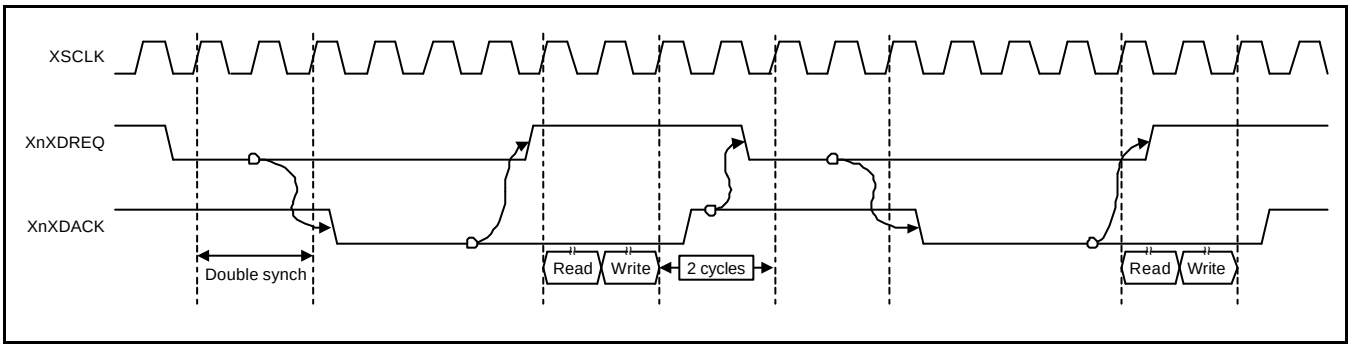


Figure 8-5. Single service in Handshake Mode with Unit Transfer Size

Whole service in Handshake Mode with Unit Transfer Size

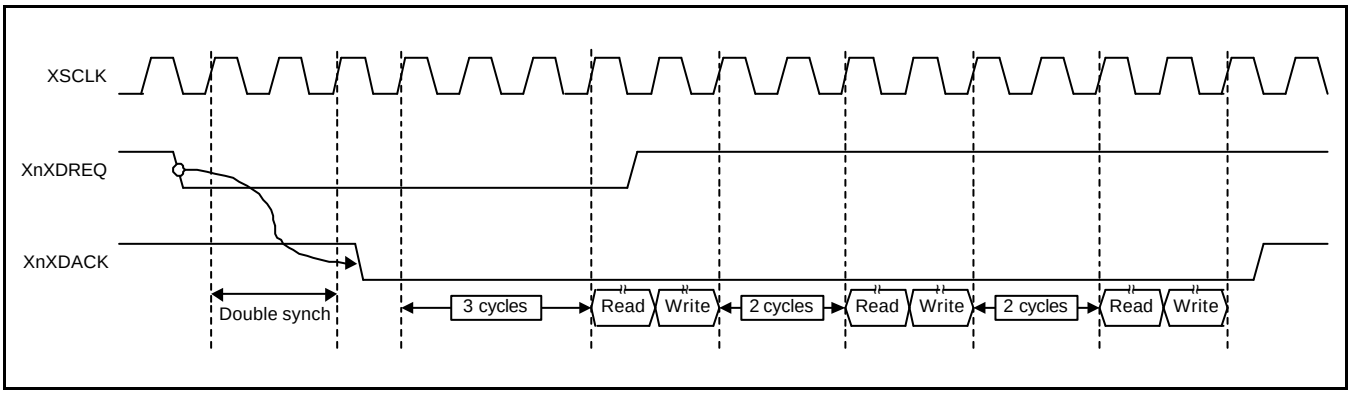


Figure 8-6. Whole service in Handshake Mode with Unit Transfer Size

DMA SPECIAL REGISTERS

Each DMA channel has nine control registers (36 in total since there are four channels for DMA controller). Six of the control registers control the DMA transfer, and other three ones monitor the status of DMA controller. The details of those registers are as follows.

DMA INITIAL SOURCE (DISRC) REGISTER

Register	Address	R/W	Description	Reset Value
DISRC0	0x4B000000	R/W	DMA 0 initial source register	0x00000000
DISRC1	0x4B000040	R/W	DMA 1 initial source register	0x00000000
DISRC2	0x4B000080	R/W	DMA 2 initial source register	0x00000000
DISRC3	0x4B0000C0	R/W	DMA 3 initial source register	0x00000000

DISRCn	Bit	Description	Initial State
S_ADDR	[30:0]	Base address (start address) of source data to transfer. This bit value will be loaded into CURR_SRC only if the CURR_SRC is 0 and the DMA ACK is 1.	0x00000000

DMA INITIAL SOURCE CONTROL (DISRCC) REGISTER

Register	Address	R/W	Description	Reset Value
DISRCC0	0x4B000004	R/W	DMA 0 initial source control register	0x00000000
DISRCC1	0x4B000044	R/W	DMA 1 initial source control register	0x00000000
DISRCC2	0x4B000084	R/W	DMA 2 initial source control register	0x00000000
DISRCC3	0x4B0000C4	R/W	DMA 3 initial source control register	0x00000000

DISRCCn	Bit	Description	Initial State
LOC	[1]	Bit 1 is used to select the location of source. 0: the source is in the system bus (AHB). 1: the source is in the peripheral bus (APB).	0
INC	[0]	Bit 0 is used to select the address increment. 0 = Increment 1= Fixed If it is 0, the address is increased by its data size after each transfer in burst and single transfer mode. If it is 1, the address is not changed after the transfer. (In the burst mode, address is increased during the burst transfer, but the address is recovered to its first value after the transfer.)	0

DMA INITIAL DESTINATION (DIDST) REGISTER

Register	Address	R/W	Description	Reset Value
DIDST0	0x4B000008	R/W	DMA 0 initial destination register	0x00000000
DIDST1	0x4B000048	R/W	DMA 1 initial destination register	0x00000000
DIDST2	0x4B000088	R/W	DMA 2 initial destination register	0x00000000
DIDST3	0x4B0000B8	R/W	DMA 3 initial destination register	0x00000000

DIDSTn	Bit	Description	Initial State
D_ADDR	[30:0]	Base address (start address) of destination for the transfer. This bit value will be loaded into CURR_SRC only if the CURR_DST is 0 and the DMA ACK is 1.	0x00000000

DMA INITIAL DESTINATION CONTROL (DIDSTC) REGISTER

Register	Address	R/W	Description	Reset Value
DIDSTC0	0x4B00000C	R/W	DMA 0 initial destination control register	0x00000000
DIDSTC1	0x4B00004C	R/W	DMA 1 initial destination control register	0x00000000
DIDSTC2	0x4B00008C	R/W	DMA 2 initial destination control register	0x00000000
DIDSTC3	0x4B0000CC	R/W	DMA 3 initial destination control register	0x00000000

DIDSTCn	Bit	Description	Initial State
LOC	[1]	Bit 1 is used to select the location of destination. 0: the destination is in the system bus (AHB). 1: the destination is in the peripheral bus (APB).	0
INC	[0]	Bit 0 is used to select the address increment. 0 = Increment 1= Fixed If it is 0, the address is increased by its data size after each transfer in burst and single transfer mode. If it is 1, the address is not changed after the transfer. (In the burst mode, address is increased during the burst transfer, but the address is recovered to its first value after the transfer.)	0

DMA CONTROL (DCON) REGISTER

Register	Address	R/W	Description	Reset Value
DCON0	0x4B000010	R/W	DMA 0 control register	0x00000000
DCON1	0x4B000050	R/W	DMA 1 control register	0x00000000
DCON2	0x4B000090	R/W	DMA 2 control register	0x00000000
DCON3	0x4B0000D0	R/W	DMA 3 control register	0x00000000

DCONn	Bit	Description	Initial State
DMD_HS	[31]	Select one between Demand mode and Handshake mode. 0: Demand mode is selected. 1: Handshake mode is selected. In both modes, DMA controller starts its transfer and asserts DACK for a given asserted DREQ. The difference between the two modes is whether it waits for the deasserted DACK or not. In the Handshake mode, DMA controller waits for the deasserted DREQ before starting a new transfer. If it finds the deasserted DREQ, it deasserts DACK and waits for another asserted DREQ. In contrast, in the Demand mode, DMA controller does not wait until the DREQ is deasserted. It just deasserts DACK and then starts another transfer if DREQ is asserted. We recommend using Handshake mode for external DMA request sources to prevent unintended starts of new transfers.	0
SYNC	[30]	Select DREQ/DACK synchronization. 0: DREQ and DACK are synchronized to PCLK (APB clock). 1: DREQ and DACK are synchronized to HCLK (AHB clock). Therefore, for devices attached to AHB system bus, this bit has to be set to 1, while for those attached to APB system, it should be set to 0. For the devices attached to external systems, the user should select this bit depending on which the external system is synchronized with between AHB system and APB system.	0
INT	[29]	Enable/Disable the interrupt setting for CURR_TC (terminal count) 0: CURR_TC interrupt is disabled. The user has to view the transfer count in the status register (i.e. polling). 1: interrupt request is generated when all the transfer is done (i.e. CURR_TC becomes 0).	0
TSZ	[28]	Select the transfer size of an atomic transfer (i.e. transfer performed each time DMA owns the bus before releasing the bus). 0: a unit transfer is performed. 1: a burst transfer of length four is performed.	0

DMA CONTROL (DCON) REGISTER (Continued)

DCONn	Bit	Description	Initial State
SERVMODE	[27]	Select the service mode between Single service mode and Whole service mode. 0: Single service mode is selected in which after each atomic transfer (single or burst of length four) DMA stops and waits for another DMA request. 1: Whole service mode is selected in which one request gets atomic transfers to be repeated until the transfer count reaches to 0. In this mode, additional request are not required. Note that even in the Whole service mode, DMA releases the bus after each atomic transfer and then tries to re-get the bus to prevent starving of other bus masters.	0
HWSRCSEL	[26:24]	Select DMA request source for each DMA. DCON0: 000:nXDREQ0 001:UART0 010:SDI 011:Timer 100:USB device EP1 DCON1: 000:nXDREQ1 001:UART1 010:I2SSDI 011:SPI 100:USB device EP2 DCON2: 000:I2SSDO 001:I2SSDI 010:SDI 011:Timer 100:USB device EP3 DCON3: 000:UART2 001:SDI 010:SPI 011:Timer 100:USB device EP4 These bits control the 4-1 MUX to select the DMA request source of each DMA. These bits have meanings only if H/W request mode is selected by DCONn[23].	00
SWHW_SEL	[23]	Select the DMA source between software (S/W request mode) and hardware (H/W request mode). 0: S/W request mode is selected and DMA is triggered by setting SW_TRIG bit of DMASKTRIG control register. 1: DMA source selected by bit[26:24] triggers the DMA operation.	0
RELOAD	[22]	Set the reload on/off option. 0: auto reload is performed when a current value of transfer count becomes 0 (i.e. all the required transfers are performed). 1: DMA channel (DMA REQ) is turned off when a current value of transfer count becomes 0. The channel on/off bit (DMASKTRIGn[1]) is set to 0 (DREQ off) to prevent unintended further start of new DMA operation.	0
DSZ	[21:20]	Data size to be transferred. 00 = Byte 01 = Half word 10 = Word 11 = reserved	00
TC	[19:0]	Initial transfer count (or transfer beat). Note that the actual number of bytes that are transferred is computed by the following equation: DSZ x TSZ x TC. Where, DSZ, TSZ (1 or 4), and TC represent data size (DCONn[21:20]), transfer size (DCONn[28]), and initial transfer count, respectively. This value will be loaded into CURR_SRC only if the CURR_SRC is 0 and the DMA ACK is 1.	00000

DMA STATUS (DSTAT) REGISTER

Register	Address	R/W	Description	Reset Value
DSTAT0	0x4B000014	R	DMA 0 count register	000000h
DSTAT1	0x4B000054	R	DMA 1 count register	000000h
DSTAT2	0x4B000094	R	DMA 2 count register	000000h
DSTAT3	0x4B0000D4	R	DMA 3 count register	000000h

DSTATn	Bit	Description	Initial State
STAT	[21:20]	Status of this DMA controller. 00: Indicates that DMA controller is ready for another DMA request. 01: Indicates that DMA controller is busy for transfers.	00b
CURR_TC	[19:0]	Current value of transfer count. Note that transfer count is initially set to the value of DCONn[19:0] register and decreased by one at the end of every atomic transfer.	00000h

DMA CURRENT SOURCE (DCSRC) REGISTER

Register	Address	R/W	Description	Reset Value
DCSRC0	0x4B000018	R	DMA 0 current Source Register	0x00000000
DCSRC1	0x4B000058	R	DMA 1 current Source Register	0x00000000
DCSRC2	0x4B000098	R	DMA 2 current Source Register	0x00000000
DCSRC3	0x4B0000D8	R	DMA 3 current Source Register	0x00000000

DCSRCn	Bit	Description	Initial State
CURR_SRC	[30:0]	Current source address for DMAn	0x00000000

CURRENT DESTINATION (DCDST) REGISTER

Register	Address	R/W	Description	Reset Value
DCDST0	0x4B00001C	R	DMA 0 current destination register	0x00000000
DCDST1	0x4B00005C	R	DMA 1 current destination register	0x00000000
DCDST2	0x4B00009C	R	DMA 2 current destination register	0x00000000
DCDST3	0x4B0000DC	R	DMA 3 current destination register	0x00000000

DCDSTn	Bit	Description	Initial State
CURR_DST	[30:0]	Current destination address for DMAn	0x00000000

DMA MASK TRIGGER (DMASKTRIG) REGISTER

Register	Address	R/W	Description	Reset Value
DMASKTRIG0	0x4B000020	R/W	DMA 0 mask trigger register	000
DMASKTRIG1	0x4B000060	R/W	DMA 1 mask trigger register	000
DMASKTRIG2	0x4B0000A0	R/W	DMA 2 mask trigger register	000
DMASKTRIG3	0x4B0000E0	R/W	DMA 3 mask trigger register	000

DMASKTRIGn	Bit	Description	Initial State
STOP	[2]	Stop the DMA operation. 1: DMA stops as soon as the current atomic transfer ends. If there is no current running atomic transfer, DMA stops immediately. The CURR_TC will be 0. **NOTE:** Due to possible current atomic transfer, “stop” operation may take several cycles. The finish of the operation (i.e. actual stop time) can be detected as soon as the channel on/off bit (DMASKTRIGn[1]) is set to off. This stop is “actual stop”.	0
ON_OFF	[1]	DMA channel on/off bit. 0: DMA channel is turned off. (DMA request to this channel is ignored.) 1: DMA channel is turned on and the DMA request is handled. This bit is automatically set to off if we set the DCONn[22] bit to “no auto reload” and/or STOP bit of DMASKTRIGn to “stop”. Note that when DCON[22] bit is "no auto reload", this bit becomes 0 when CURR_TC reaches 0. If the STOP bit is 1, this bit becomes 0 as soon as the current atomic transfer is completed. **NOTE:** This bit should not be changed manually during DMA operations (i.e. this has to be changed only by using DCON[22] or STOP bit).	0
SW_TRIG	[0]	Trigger the DMA channel in S/W request mode. 1: it requests a DMA operation to this controller. Note that this trigger gets effective after S/W request mode has to be selected (DCONn[23]) and channel ON_OFF bit has to be set to 1 (channel on). When DMA operation starts, this bit is cleared automatically.	0

NOTE: You can freely change the values of DISRC register, DIDST registers, and TC field of DCON register. Those changes take effect only after the finish of current transfer (i.e. when CURR_TC becomes 0). On the other hand, any change made to other registers and/or fields takes immediate effect. Therefore, be careful in changing those registers and fields.

S/W Work-Around

The DMA auto-reload is occurred only when the DMA request is issued after the DMA counter reaches 0. So, the following code should be used in the DMA done interrupt handler before setting the DMA source address, destination address and counter register for the next auto-reload. This code will wait until the first DMA request is issued and the previous auto-reload value is loaded.

```
while((rDSTATn&0xfffff)==0);
```

9 I/O PORTS

OVERVIEW

The S3C2410A has 117 multi-functional input/output port pins. The ports are:

— Port A (GPA): 23-output port
— Port B (GPB): 11-input/output port
— Port C (GPC): 16-input/output port
— Port D (GPD): 16-input/output port
— Port E (GPE): 16-input/output port
— Port F (GPF): 8-input/output port
— Port G (GPG): 16-input/output port
— Port H (GPH): 11-input/output port

Each port can be easily configured by software to meet various system configurations and design requirements. You have to define which function of each pin is used before starting the main program. If a pin is not used for multiplexed functions, the pin can be configured as I/O ports.

Initial pin states are configured seamlessly to avoid problems.

Table 9-1. S3C2410A Port Configuration

Port A	Selectable Pin Functions			
GPA22	Output only	nFCE	–	–
GPA21	Output only	nRSTOUT	–	–
GPA20	Output only	nFRE	–	–
GPA19	Output only	nFWE	–	–
GPA18	Output only	ALE	–	–
GPA17	Output only	CLE	–	–
GPA16	Output only	nGCS5	–	–
GPA15	Output only	nGCS4	–	–
GPA14	Output only	nGCS3	–	–
GPA13	Output only	nGCS2	–	–
GPA12	Output only	nGCS1	–	–
GPA11	Output only	ADDR26	–	–
GPA10	Output only	ADDR25	–	–
GPA9	Output only	ADDR24	–	–
GPA8	Output only	ADDR23	–	–
GPA7	Output only	ADDR22	–	–
GPA6	Output only	ADDR21	–	–
GPA5	Output only	ADDR20	–	–
GPA4	Output only	ADDR19	–	–
GPA3	Output only	ADDR18	–	–
GPA2	Output only	ADDR17	–	–
GPA1	Output only	ADDR16	–	–
GPA0	Output only	ADDR0	–	–

Table 9-1. S3C2410A Port Configuration (Continued)

Port B	Selectable Pin Functions			
GPB10	Input/output	nXDREQ0	–	–
GPB9	Input/output	nXDACK0	–	–
GPB8	Input/output	nXDREQ1	–	–
GPB7	Input/output	nXDACK1	–	–
GPB6	Input/output	nXBREQ	–	–
GPB5	Input/output	nXBACK	–	–
GPB4	Input/output	TCLK0	–	–
GPB3	Input/output	TOUT3	–	–
GPB2	Input/output	TOUT2	–	–
GPB1	Input/output	TOUT1	–	–
GPB0	Input/output	TOUT0	–	–

Port C	Selectable Pin Functions			
GPC15	Input/output	VD7	–	–
GPC14	Input/output	VD6	–	–
GPC13	Input/output	VD5	–	–
GPC12	Input/output	VD4	–	–
GPC11	Input/output	VD3	–	–
GPC10	Input/output	VD2	–	–
GPC9	Input/output	VD1	–	–
GPC8	Input/output	VD0	–	–
GPC7	Input/output	LCDVF2	–	–
GPC6	Input/output	LCDVF1	–	–
GPC5	Input/output	LCDVF0	–	–
GPC4	Input/output	VM	–	–
GPC3	Input/output	VFRAME	–	–
GPC2	Input/output	VLINE	–	–
GPC1	Input/output	VCLK	–	–
GPC0	Input/output	LEND	–	–

Table 9-1. S3C2410A Port Configuration (Continued)

Port D	Selectable Pin Functions			
GPD15	Input/output	VD23	nSS0	–
GPD14	Input/output	VD22	nSS1	–
GPD13	Input/output	VD21	–	–
GPD12	Input/output	VD20	–	–
GPD11	Input/output	VD19	–	–
GPD10	Input/output	VD18	–	–
GPD9	Input/output	VD17	–	–
GPD8	Input/output	VD16	–	–
GPD7	Input/output	VD15	–	–
GPD6	Input/output	VD14	–	–
GPD5	Input/output	VD13	–	–
GPD4	Input/output	VD12	–	–
GPD3	Input/output	VD11	–	–
GPD2	Input/output	VD10	–	–
GPD1	Input/output	VD9	–	–
GPD0	Input/output	VD8	–	–

Port E	Selectable Pin Functions			
GPE15	Input/output	IICSDA	–	–
GPE14	Input/output	IICSCL	–	–
GPE13	Input/output	SPICLK0	–	–
GPE12	Input/output	SPIMOSI0	–	–
GPE11	Input/output	SPIMISO0	–	–
GPE10	Input/output	SDDAT3	–	–
GPE9	Input/output	SDDAT2	–	–
GPE8	Input/output	SDDAT1	–	–
GPE7	Input/output	SDDAT0	–	–
GPE6	Input/output	SDCMD	–	–
GPE5	Input/output	SDCLK	–	–
GPE4	Input/output	I2SSDO	I2SSDI	–
GPE3	Input/output	I2SSDI	nSS0	–
GPE2	Input/output	CDCLK	–	–
GPE1	Input/output	I2SSCLK	–	–
GPE0	Input/output	I2SLRCK	–	–

Table 9-1. S3C2410A Port Configuration (Continued)

Port F	Selectable Pin Functions			
GPF7	Input/output	EINT7	–	–
GPF6	Input/output	EINT6	–	–
GPF5	Input/output	EINT5	–	–
GPF4	Input/output	EINT4	–	–
GPF3	Input/output	EINT3	–	–
GPF2	Input/output	EINT2		
GPF1	Input/output	EINT1		
GPF0	Input/output	EINT0		

Port G	Selectable Pin Functions			
GPG15	Input/output	EINT23	nYPON	–
GPG14	Input/output	EINT22	YMON	–
GPG13	Input/output	EINT21	nXPON	–
GPG12	Input/output	EINT20	XMON	–
GPG11	Input/output	EINT19	TCLK1	–
GPG10	Input/output	EINT18	–	–
GPG9	Input/output	EINT17	–	–
GPG8	Input/output	EINT16	–	–
GPG7	Input/output	EINT15	SPICLK1	–
GPG6	Input/output	EINT14	SPIMOSI1	–
GPG5	Input/output	EINT13	SPIMISO1	–
GPG4	Input/output	EINT12	LCD_PWREN	–
GPG3	Input/output	EINT11	nSS1	–
GPG2	Input/output	EINT10	nSS0	–
GPG1	Input/output	EINT9	–	–
GPG0	Input/output	EINT8	–	–

Table 9-1. S3C2410A Port Configuration (Continued)

Port H	Selectable Pin Functions			
GPH10	Input/output	CLKOUT1	–	–
GPH9	Input/output	CLKOUT0	–	–
GPH8	Input/output	UEXTCLK	–	–
GPH7	Input/output	RXD2	nCTS1	–
GPH6	Input/output	TXD2	nRTS1	–
GPH5	Input/output	RXD1	–	–
GPH4	Input/output	TXD1	–	–
GPH3	Input/output	RXD0	–	–
GPH2	Input/output	TXD0	–	–
GPH1	Input/output	nRTS0	–	–
GPH0	Input/output	nCTS0	–	–

PORT CONTROL DESCRIPTIONS

PORT CONFIGURATION REGISTER (GPACON-GPHCON)

In the S3C2410A, most pins are multiplexed. So, It is require to determine which function is selected for each pin. port control register (PnCON) determines the function of each pin.

If GPF0 – GPF7 and GPG0 – GPG7 are used for wakeup signals in Power-OFF mode, these ports must be configured in Interrupt mode.

PORT DATA REGISTER (GPADAT-GPHDAT)

If ports are configured as output ports, data can be written to the corresponding bit of the PnDAT. If ports are configured as input ports, the data can be read from the corresponding bit of the PnDAT.

PORT PULL-UP REGISTER (GPBUP-GPHUP)

The port pull-up register controls the pull-up resister enable/disable of each port group. When the corresponding bit is 0, the pull-up resister of the pin is enabled. When 1, the pull-up resister is disabled.

If the port pull-up register is enabled, the pull-up resisters work without pin's functional setting (input, output, DATAn, EINTn, etc).

MISCELLANEOUS CONTROL REGISTER

This register controls DATA port pull-up resister, hi-z state, USB pad, and CLKOUT selection.

EXTERNAL INTERRUPT CONTROL REGISTER (EXTINTN)

The 24 external interrupts are requested by various signaling methods. The EXTINTn register configures the signaling method among the low level trigger, high level trigger, falling edge trigger, rising edge trigger, and both edge trigger for the external interrupt request.

The 8 external interrupt pin has a digital filter (refer to EINTFLTn on page 9-25).

Only 16 EINT pins (EINT [15:0]) are used for wakeup sources.

POWER_OFF MODE AND I/O PORTS

All GPIO register values are preserved in Power_OFF mode. Refer to the Power_OFF mode in the chapter, Clock & Power Management.

The EINTMASK can't prohibit the wake-up from Power_OFF mode, But, If ENTMASK is masking one of EINT[15:4], the wake-up can be done but the EINT4_7 bit and EINT8_23 bit of the SRCPND will not set to 1 just after the wake-up.

I/O PORT CONTROL REGISTER

PORT A CONTROL REGISTERS (GPACON/GPADAT)

Register	Address	R/W	Description	Reset Value
GPACON	0x56000000	R/W	Configure the pins of port A	0x7FFFFF
GPADAT	0x56000004	R/W	The data register for port A	Undefined
Reserved	0x56000008	–	Reserved	Undefined
Reserved	0x5600000C	–	Reserved	Undefined

GPACON	Bit	Description
GPA22	[22]	0 = Output 1 = nFCE
GPA21	[21]	0 = Output 1 = nRSTOUT (nRSTOUT = nRESET & nWDTRST & SW_RESET(MISCCR[16]))
GPA20	[20]	0 = Output 1 = nFRE
GPA19	[19]	0 = Output 1 = nFWE
GPA18	[18]	0 = Output 1 = ALE
GPA17	[17]	0 = Output 1 = CLE
GPA16	[16]	0 = Output 1 = nGCS5
GPA15	[15]	0 = Output 1 = nGCS4
GPA14	[14]	0 = Output 1 = nGCS3
GPA13	[13]	0 = Output 1 = nGCS2
GPA12	[12]	0 = Output 1 = nGCS1
GPA11	[11]	0 = Output 1 = ADDR26
GPA10	[10]	0 = Output 1 = ADDR25
GPA9	[9]	0 = Output 1 = ADDR24
GPA8	[8]	0 = Output 1 = ADDR23
GPA7	[7]	0 = Output 1 = ADDR22
GPA6	[6]	0 = Output 1 = ADDR21
GPA5	[5]	0 = Output 1 = ADDR20
GPA4	[4]	0 = Output 1 = ADDR19
GPA3	[3]	0 = Output 1 = ADDR18
GPA2	[2]	0 = Output 1 = ADDR17
GPA1	[1]	0 = Output 1 = ADDR16
GPA0	[0]	0 = Output 1 = ADDR0

GPADAT	Bit	Description
GPA[22:0]	[22:0]	When the port is configured as output port, the pin state is the same as the that of the corresponding bit. When the port is configured as functional pin, undefined value will be read.

PORT B CONTROL REGISTERS (GPBCON, GPBDAT, and GPBUP)

Register	Address	R/W	Description	Reset Value
GPBCON	0x56000010	R/W	Configure the pins of port B	0x0
GPBDAT	0x56000014	R/W	The data register for port B	Undefined
GPBUP	0x56000018	R/W	Pull-up disable register for port B	0x0
Reserved	0x5600001C	–	Reserved	Undefined

GPBCON	Bit	Description
GPB10	[21:20]	00 = Input 01 = Output 10 = nXDREQ0 11 = reserved
GPB9	[19:18]	00 = Input 01 = Output 10 = nXDACK0 11 = reserved
GPB8	[17:16]	00 = Input 01 = Output 10 = nXDREQ1 11 = Reserved
GPB7	[15:14]	00 = Input 01 = Output 10 = nXDACK1 11 = Reserved
GPB6	[13:12]	00 = Input 01 = Output 10 = nXBREQ 11 = reserved
GPB5	[11:10]	00 = Input 01 = Output 10 = nXBACK 11 = reserved
GPB4	[9:8]	00 = Input 01 = Output 10 = TCLK0 11 = reserved
GPB3	[7:6]	00 = Input 01 = Output 10 = TOUT3 11 = reserved
GPB2	[5:4]	00 = Input 01 = Output 10 = TOUT2 11 = reserved]
GPB1	[3:2]	00 = Input 01 = Output 10 = TOUT1 11 = reserved
GPB0	[1:0]	00 = Input 01 = Output 10 = TOUT0 11 = reserved

GPBDAT	Bit	Description
GPB[10:0]	[10:0]	When the port is configured as input port, data from external sources can be read to the corresponding pin. When the port is configured as output port, data written in this register can be sent to the corresponding pin. When the port is configured as functional pin, undefined value will be read.

GPBUP	Bit	Description
GPB[10:0]	[10:0]	0: The pull-up function attached to to the corresponding port pin is enabled. 1: The pull-up function is disabled.

PORT C CONTROL REGISTERS (GPCCON, GPCDAT, and GPCUP)

Register	Address	R/W	Description	Reset Value
GPCCON	0x56000020	R/W	Configure the pins of port C	0x0
GPCDAT	0x56000024	R/W	The data register for port C	Undefined
GPCUP	0x56000028	R/W	Pull-up disable register for port C	0x0
Reserved	0x5600002C	–	Reserved	Undefined

GPCCON	Bit	Description
GPC15	[31:30]	00 = Input 01 = Output 10 = VD[7] 11 = Reserved
GPC14	[29:28]	00 = Input 01 = Output 10 = VD[6] 11 = Reserved
GPC13	[27:26]	00 = Input 01 = Output 10 = VD[5] 11 = Reserved
GPC12	[25:24]	00 = Input 01 = Output 10 = VD[4] 11 = Reserved
GPC11	[23:22]	00 = Input 01 = Output 10 = VD[3] 11 = Reserved
GPC10	[21:20]	00 = Input 01 = Output 10 = VD[2] 11 = Reserved
GPC9	[19:18]	00 = Input 01 = Output 10 = VD[1] 11 = Reserved
GPC8	[17:16]	00 = Input 01 = Output 10 = VD[0] 11 = Reserved
GPC7	[15:14]	00 = Input 01 = Output 10 = LCDVF2 11 = Reserved
GPC6	[13:12]	00 = Input 01 = Output 10 = LCDVF1 11 = Reserved
GPC5	[11:10]	00 = Input 01 = Output 10 = LCDVF0 11 = Reserved
GPC4	[9:8]	00 = Input 01 = Output 10 = VM 11 = Reserved
GPC3	[7:6]	00 = Input 01 = Output 10 = VFRAME 11 = Reserved
GPC2	[5:4]	00 = Input 01 = Output 10 = VLINE 11 = Reserved
GPC1	[3:2]	00 = Input 01 = Output 10 = VCLK 11 = Reserved
GPC0	[1:0]	00 = Input 01 = Output 10 = LEND 11 = Reserved

GPCDAT	Bit	Description
GPC[15:0]	[15:0]	When the port is configured as input port, data from external sources can be read to the corresponding pin. When the port is configured as output port, data written in this register can be sent to the corresponding pin. When the port is configured as functional pin, undefined value will be read.

GPCUP	Bit	Description
GPC[15:0]	[15:0]	0: The pull-up function attached to to the corresponding port pin is enabled. 1: The pull-up function is disabled.

PORT D CONTROL REGISTERS (GPDCON, GPDDAT, and GPDUP)

Register	Address	R/W	Description	Reset Value
GPDCON	0x56000030	R/W	Configure the pins of port D	0x0
GPDDAT	0x56000034	R/W	The data register for port D	Undefined
GPDUP	0x56000038	R/W	Pull-up disable register for port D	0xF000
Reserved	0x5600003C	–	Reserved	Undefined

GPDCON	Bit	Description
GPD15	[31:30]	00 = Input 01 = Output 10 = VD23 11 = nSS0
GPD14	[29:28]	00 = Input 01 = Output 10 = VD22 11 = nSS1
GPD13	[27:26]	00 = Input 01 = Output 10 = VD21 11 = Reserved
GPD12	[25:24]	00 = Input 01 = Output 10 = VD20 11 = Reserved
GPD11	[23:22]	00 = Input 01 = Output 10 = VD19 11 = Reserved
GPD10	[21:20]	00 = Input 01 = Output 10 = VD18 11 = Reserved
GPD9	[19:18]	00 = Input 01 = Output 10 = VD17 11 = Reserved
GPD8	[17:16]	00 = Input 01 = Output 10 = VD16 11 = Reserved
GPD7	[15:14]	00 = Input 01 = Output 10 = VD15 11 = Reserved
GPD6	[13:12]	00 = Input 01 = Output 10 = VD14 11 = Reserved
GPD5	[11:10]	00 = Input 01 = Output 10 = VD13 11 = Reserved
GPD4	[9:8]	00 = Input 01 = Output 10 = VD12 11 = Reserved
GPD3	[7:6]	00 = Input 01 = Output 10 = VD11 11 = Reserved
GPD2	[5:4]	00 = Input 01 = Output 10 = VD10 11 = Reserved
GPD1	[3:2]	00 = Input 01 = Output 10 = VD9 11 = Reserved
GPD0	[1:0]	00 = Input 01 = Output 10 = VD8 11 = Reserved

GPDDAT	Bit	Description
GPD[15:0]	[15:0]	When the port is configured as input port, data from external sources can be read to the corresponding pin. When the port is configured as output port, data written in this register can be sent to the corresponding pin. When the port is configured as functional pin, undefined value will be read.

GPDUP	Bit	Description
GPD[15:0]	[15:0]	0: The pull-up function attached to to the corresponding port pin is enabled. 1: The pull-up function is disabled. (GPD[15:12] are "pull-up disabled" state at the initial condition.)

PORT E CONTROL REGISTERS (GPECON, GPEDAT, and GPEUP)

Register	Address	R/W	Description	Reset Value
GPECON	0x56000040	R/W	Configure the pins of port E	0x0
GPEDAT	0x56000044	R/W	The data register for port E	Undefined
GPEUP	0x56000048	R/W	pull-up disable register for port E	0x0
Reserved	0x5600004C	–	Reserved	Undefined

GPECON	Bit	Description
GPE15	[31:30]	00 = Input 01 = Output (open drain output) 10 = IICSDA 11 = Reserved
GPE14	[29:28]	00 = Input 01 = Output (open drain output) 10 = IICSCL 11 = Reserved
GPE13	[27:26]	00 = Input 01 = Output 10 = SPICLK0 11 = Reserved
GPE12	[25:24]	00 = Input 01 = Output 10 = SPIMOSI0 11 = Reserved
GPE11	[23:22]	00 = Input 01 = Output 10 = SPIMISO0 11 = Reserved
GPE10	[21:20]	00 = Input 01 = Output 10 = SDDAT3 11 = Reserved
GPE9	[19:18]	00 = Input 01 = Output 10 = SDDAT2 11 = Reserved
GPE8	[17:16]	00 = Input 01 = Output 10 = SDDAT1 11 = Reserved
GPE7	[15:14]	00 = Input 01 = Output 10 = SDDAT0 11 = Reserved
GPE6	[13:12]	00 = Input 01 = Output 10 = SDCMD 11 = Reserved
GPE5	[11:10]	00 = Input 01 = Output 10 = SDCLK 11 = Reserved
GPE4	[9:8]	00 = Input 01 = Output 10 = I2SSDO 11 = I2SSDI
GPE3	[7:6]	00 = Input 01 = Output 10 = I2SSDI 11 = nSS0
GPE2	[5:4]	00 = Input 01 = Output 10 = CDCLK 11 = Reserved
GPE1	[3:2]	00 = Input 01 = Output 10 = I2SSCLK 11 = Reserved
GPE0	[1:0]	00 = Input 01 = Output 10 = I2SLRCK 11 = Reserved

GPEDAT	Bit	Description
GPE[15:0]	[15:0]	When the port is configured as input port, data from external sources can be read to the corresponding pin. When the port is configured as output port, data written in this register can be sent to the corresponding pin. When the port is configured as a functional pin, undefined value will be read.

GPEUP	Bit	Description
GPE[15:0]	[15:0]	0: The pull-up function attached to to the corresponding port pin is enabled. 1: The pull-up function is disabled.

PORT F CONTROL REGISTERS (GPFCON, GPFDAT, and GPFPU)

If GPF0 - GPF7 are used for wakeup signals in Power_OFF mode, the ports must be configured as external interrupt (set in Interrupt mode).

Register	Address	R/W	Description	Reset Value
GPFCON	0x56000050	R/W	Configure the pins of port F	0x0
GPFDAT	0x56000054	R/W	The data register for port F	Undefined
GPFUP	0x56000058	R/W	Pull-up disable register for port F	0x0
Reserved	0x5600005C	–	Reserved	Undefined

GPFCON	Bit	Description
GPF7	[15:14]	00 = Input 01 = Output 10 = EINT7 11 = Reserved
GPF6	[13:12]	00 = Input 01 = Output 10 = EINT6 11 = Reserved
GPF5	[11:10]	00 = Input 01 = Output 10 = EINT5 11 = Reserved
GPF4	[9:8]	00 = Input 01 = Output 10 = EINT4 11 = Reserved
GPF3	[7:6]	00 = Input 01 = Output 10 = EINT3 11 = Reserved
GPF2	[5:4]	00 = Input 01 = Output 10 = EINT2 11 = Reserved
GPF1	[3:2]	00 = Input 01 = Output 10 = EINT1 11 = Reserved
GPF0	[1:0]	00 = Input 01 = Output 10 = EINT0 11 = Reserved

NOTES:

1.

GPFDAT	Bit	Description
GPF[7:0]	[7:0]	When the port is configured as input port, data from external sources can be read to the corresponding pin. When the port is configured as output port, data written in this register can be sent to the corresponding pin. When the port is configured as functional pin, undefined value will be read.

2.

GPFUP	Bit	Description
GPF[7:0]	[7:0]	0: The pull-up function attached to to the corresponding port pin is enabled. 1: The pull-up function is disabled.

PORT G CONTROL REGISTERS (GPGCON, GPGDAT, AND GPGUP)

If GPG [7:0] are used for wakeup signals in Power_OFF mode, the ports must be configured as external interrupt (set in Interrupt mode).

Register	Address	R/W	Description	Reset Value
GPGCON	0x56000060	R/W	Configure the pins of port G	0x0
GPGDAT	0x56000064	R/W	The data register for port G	Undefined
GPGUP	0x56000068	R/W	Pull-up disable register for port G	0xF800
Reserved	0x5600006C	–	Reserved	Undefined

GPGCON	Bit	Description
GPG15	[31:30]	00 = Input 01 = Output 10 = EINT23 11 = nYPON
GPG14	[29:28]	00 = Input 01 = Output 10 = EINT22 11 = YMON
GPG13	[27:26]	00 = Input 01 = Output 10 = EINT21 11 = nXPON
GPG12	[25:24]	00 = Input 01 = Output 10 = EINT20 11 = XMON
GPG11	[23:22]	00 = Input 01 = Output 10 = EINT19 11 = TCLK1
GPG10 (5V Tolerant Input)	[21:20]	00 = Input 01 = Output 10 = EINT18 11 = Reserved
GPG9 (5V Tolerant Input)	[19:18]	00 = Input 01 = Output 10 = EINT17 11 = Reserved
GPG8 (5V Tolerant Input)	[17:16]	00 = Input 01 = Output 10 = EINT16 11 = Reserved
GPG7	[15:14]	00 = Input 01 = Output 10 = EINT15 11 = SPICLK1
GPG6	[13:12]	00 = Input 01 = Output 10 = EINT14 11 = SPIMOSI1
GPG5	[11:10]	00 = Input 01 = Output 10 = EINT13 11 = SPIMISO1
GPG4	[9:8]	00 = Input 01 = Output 10 = EINT12 11 = LCD_PWREN
GPG3	[7:6]	00 = Input 01 = Output 10 = EINT11 11 = nSS1
GPG2	[5:4]	00 = Input 01 = Output 10 = EINT10 11 = nSS0
GPG1	[3:2]	00 = Input 01 = Output 10 = EINT9 11 = Reserved
GPG0	[1:0]	00 = Input 01 = Output 10 = EINT8 11 = Reserved

GPGDAT	Bit	Description
GPG[15:0]	[15:0]	When the port is configured as input port, data from external sources can be read to the corresponding pin. When the port is configured as output port, data written in this register can be sent to the corresponding pin. When the port is configured as functional pin, undefined value will be read.

GPGUP	Bit	Description
GPG[15:0]	[15:0]	0: The pull-up function attached to to the corresponding port pin is enabled. 1: The pull-up function is disabled. (GPG[15:11] are "pull-up disabled" state at the initial condition.)

PORT H CONTROL REGISTERS (GPHCON, GPHDAT, AND GPHUP)

Register	Address	R/W	Description	Reset Value
GPHCON	0x56000070	R/W	Configure the pins of port H	0x0
GPHDAT	0x56000074	R/W	The data register for port H	Undefined
GPHUP	0x56000078	R/W	Pull-up disable register for port H	0x0
Reserved	0x5600007C	–	Reserved	Undefined

GPHCON	Bit	Description
GPH10	[21:20]	00 = Input 01 = Output 10 = CLKOUT1 11 = Reserved
GPH9	[19:18]	00 = Input 01 = Output 10 = CLKOUT0 11 = Reserved
GPH8	[17:16]	00 = Input 01 = Output 10 = UEXTCLK 11 = Reserved
GPH7	[15:14]	00 = Input 01 = Output 10 = RXD2 11 = nCTS1
GPH6	[13:12]	00 = Input 01 = Output 10 = TXD2 11 = nRTS1
GPH5	[11:10]	00 = Input 01 = Output 10 = RXD1 11 = Reserved
GPH4	[9:8]	00 = Input 01 = Output 10 = TXD1 11 = Reserved
GPH3	[7:6]	00 = Input 01 = Output 10 = RXD0 11 = reserved
GPH2	[5:4]	00 = Input 01 = Output 10 = TXD0 11 = Reserved
GPH1	[3:2]	00 = Input 01 = Output 10 = nRTS0 11 = Reserved
GPH0	[1:0]	00 = Input 01 = Output 10 = nCTS0 11 = Reserved

GPHDAT	Bit	Description
GPH[10:0]	[10:0]	When the port is configured as input port, data from external sources can be read to the corresponding pin. When the port is configured as output port, data written in this register can be sent to the corresponding pin. When the port is configured as functional pin, undefined value will be read.

GPHUP	Bit	Description
GPH[10:0]	[10:0]	0: The pull-up function attached to to the corresponding port pin is enabled. 1: The pull-up function is disabled.

MISCELLANEOUS CONTROL REGISTER (MISCCR)

Pads related USB are controlled by this register for USB host, or for USB device.

Register	Address	R/W	Description	Reset Value
MISCCR	0x56000080	R/W	Miscellaneous control register	0x10330

MISCCR	Bit	Description
Reserved	[21:20]	Reserved to 00b
nEN_SCKE	[19]	0: SCKE = Normal 1: SCKE = L level Used to protect SDRAM during the Power_OFF moe.
nEN_SCLK1	[18]	0: SCLK1 = SCLK 1: SCLK1 = L level Used to protect SDRAM during the Power_OFF moe.
nEN_SCLK0	[17]	0: SCLK0 = SCLK 1: SCLK0 = L level Used to protect SDRAM during the Power_OFF moe.
nRSTCON	[16]	nRSTOUT software control (SW_RESET) 0: nRSTOUT = 0, 1: nRSTOUT = 1.
Reserved	[15:14]	Reserved to 00b
USBSUSPND1	[13]	[13] USB Port 1 mode 0 = Normal 1= Suspend
USBSUSPND0	[12]	[12] USB Port 0 mode 0 = Normal 1= Suspend
Reserved	[11]	Reserved to 0b
CLKSEL1	[10:8]	CLKOUT1 output singnal source 000 = MPLL CLK 001 = UPLL CLK 010 = FCLK 011 = HCLK 100 = PCLK 101 = DCLK1 11x = Reserved
Reserved	[7]	0
CLKSEL0	[6:4]	CLKOUT0 output singnal source 000 = MPLL CLK 001 = UPLL CLK 010 = FCLK 011 = HCLK 100 = PCLK 101 = DCLK0 11x = Reserved
USBPAD	[3]	0 = Use pads related USB for USB device 1 = Use pads related USB for USB host
Reserved	[2]	Reserved to 0b.
SPUCR_L	[1]	DATA[15:0] port pull-up resister 0 = Enabled 1 = Disabled
SPUCR_H	[0]	DATA[31:16] port pull-up resister 0 = Enabled 1 = Disabled

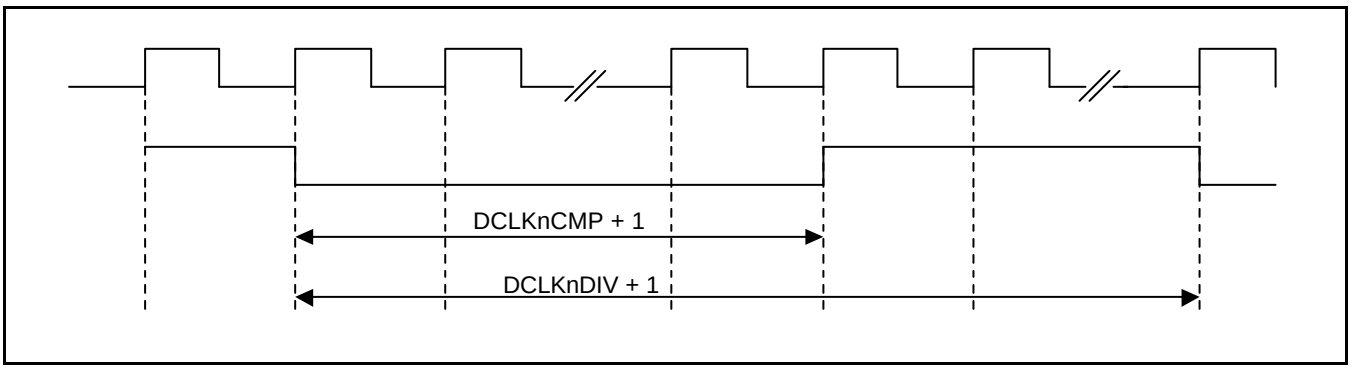
NOTE: CLKOUT is prepared only for monitoring an internal clock situation (On/Off status or frequency).

DCLK CONTROL REGISTERS (DCLKCON)

This register defines DCLKn signals, which work as clocks for external sources. See the following figure for how to make the DCLKn signals. The DCLKCON can actually operate only when CLKOUT[1:0] is set to send the DCLKn signals.

Register	Address	R/W	Description	Reset Value
DCLKCON	0x56000084	R/W	DCLK0/1 control register	0x0

DCLKCON	Bit	Description
DCLK1CMP	[27:24]	DCLK1 Compare value clock toggle value. (< DCLK1DIV) If the DCLK1CMP is n, Low level duration is (n + 1). High level duration is ((DCLK1DIV + 1) – (n +1)).
DCLK1DIV	[23:20]	DCLK1 Divide value DCLK1 frequency = source clock / (DCLK1DIV + 1)
Reserved	[19:18]	00b
DCLK1SelCK	[17]	Select DCLK1 source clock 0 = PCLK 1 = UCLK (USB)
DCLK1EN	[16]	DCLK1 Enable 0 = Disable 1 = Enable
Reserved	[15:12]	0000b
DCLK0CMP	[11:8]	DCLK0 Compare value clock toggle value. (< DCLK0DIV) If the DCLK0CMP is n, Low level duration is (n + 1). High level duration is ((DCLK0DIV + 1) – (n +1)).
DCLK0DIV	[7:4]	DCLK0 Divide value. DCLK0 frequency = source clock / (DCLK0DIV + 1)
Reserved	[3:2]	00b
DCLK0SelCK	[1]	Select DCLK0 source clock 0 = PCLK 1 = UCLK (USB)
DCLK0EN	[0]	DCLK0 Enable 0 = Disable 1 = Enable



EXTERNAL INTERRUPT CONTROL REGISTER (EXTINTn)

The 24 external interrupts can be requested by various signaling methods. The EXTINTn configures the signaling method between the level trigger and edge trigger for the external interrupt request, and also configures the signal polarity.

To recognize the level interrupt, the valid logic level on EXTINTn pin must be retained at least for 40ns because of the noise filter (EINT[15:0]).

Register	Address	R/W	Description	Reset Value
EXTINT0	0x56000088	R/W	External interrupt control register 0	0x0
EXTINT1	0x5600008C	R/W	External interrupt control register 1	0x0
EXTINT2	0x56000090	R/W	External interrupt control register 2	0x0

EXTINT0	Bit	Description
EINT7	[30:28]	Set the signaling method of the EINT7. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
EINT6	[26:24]	Set the signaling method of the EINT6. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
EINT5	[22:20]	Set the signaling method of the EINT5. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
EINT4	[18:16]	Set the signaling method of the EINT4. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
EINT3	[14:12]	Set the signaling method of the EINT3. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
EINT2	[10:8]	Set the signaling method of the EINT2. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
EINT1	[6:4]	Set the signaling method of the EINT1. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
EINT0	[2:0]	Set the signaling method of the EINT0. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered

EXTINT1	Bit	Description
Reserved	[31]	Reserved
EINT15	[30:28]	Set the signaling method of the EINT15. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
Reserved	[27]	Reserved
EINT14	[26:24]	Set the signaling method of the EINT14. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
Reserved	[23]	Reserved
EINT13	[22:20]	Set the signaling method of the EINT13. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
Reserved	[19]	Reserved
EINT12	[18:16]	Set the signaling method of the EINT12. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
Reserved	[15]	Reserved
EINT11	[14:12]	Set the signaling method of the EINT11. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
Reserved	[11]	Reserved
EINT10	[10:8]	Set the signaling method of the EINT10. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
Reserved	[7]	Reserved
EINT9	[6:4]	Set the signaling method of the EINT9. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
Reserved	[3]	Reserved
EINT8	[2:0]	Set the signaling method of the EINT8. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered

EXTINT2	Bit	Description
FLTEN23	[31]	Filter Enable for EINT23 0 = Disable 1= Enable
EINT23	[30:28]	Set the signaling method of the EINT23. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
FLTEN22	[27]	Filter Enable for EINT22 0 = Disable 1= Enable
EINT22	[26:24]	Set the signaling method of the EINT22. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
FLTEN21	[23]	Filter Enable for EINT21 0 = Disable 1= Enable
EINT21	[22:20]	Set the signaling method of the EINT21. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
FLTEN20	[19]	Filter Enable for EINT20 0 = Disable 1= Enable
EINT20	[18:16]	Set the signaling method of the EINT20. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
FLTEN19	[15]	Filter Enable for EINT19 0 = Disable 1= Enable
EINT19	[14:12]	Set the signaling method of the EINT19. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
FLTEN18	[11]	Filter Enable for EINT18 0 = Disable 1= Enable
EINT18	[10:8]	Set the signaling method of the EINT18. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
FLTEN17	[7]	Filter Enable for EINT17 0 = Disable 1= Enable
EINT17	[6:4]	Set the signaling method of the EINT17. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
FLTEN16	[3]	Filter Enable for EINT16 0 = Disable 1= Enable
EINT16	[2:0]	Set the signaling method of the EINT16. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered

EXTERNAL INTERRUPT FILTER REGISTER (EINTFLTn)

The EINTFLTn controls the length of filter for 8 external interrupts (EINT[23:16]).

Register	Address	R/W	Description	Reset Value
EINTFLT0	0x56000094	R/W	Reserved	
EINTFLT1	0x56000098	R/W	Reserved	
EINTFLT2	0x5600009C	R/W	External interrupt control register 2	0x0
EINTFLT3	0x4C6000A0	R/W	External interrupt control register 3	0x0

EINTFLT2	Bit	Description
FLTCLK19	[31]	Filter clock of EINT19 0 = PCLK 1= EXTCLK/OSC_CLK (Selected by OM pin)
EINTFLT19	[30:24]	Filter width of EINT19
FLTCLK18	[23]	Filter clock of EINT18 0 = PCLK 1= EXTCLK/OSC_CLK (Selected by OM pin)
EINTFLT18	[22:16]	Filter width of EINT18
FLTCLK17	[15]	Filter clock of EINT17 0 = PCLK 1= EXTCLK/OSC_CLK (Selected by OM pin)
EINTFLT17	[14:8]	Filter width of EINT17
FLTCLK16	[7]	Filter clock of EINT16 0 = PCLK 1= EXTCLK/OSC_CLK (Selected by OM pin)
EINTFLT16	[6:0]	Filter width of EINT16

EINTFLT3	Bit	Description
FLTCLK23	[31]	Filter clock of EINT23 0 = PCLK 1= EXTCLK/OSC_CLK (Selected by OM pin)
EINTFLT23	[30:24]	Filter width of EINT23
FLTCLK22	[23]	Filter clock of EINT22 0 = PCLK 1= EXTCLK/OSC_CLK (Selected by OM pin)
EINTFLT22	[22:16]	Filter width of EINT22
FLTCLK21	[15]	Filter clock of EINT21 0 = PCLK 1= EXTCLK/OSC_CLK (Selected by OM pin)
EINTFLT21	[14:8]	Filter width of EINT21
FLTCLK20	[7]	Filter clock of EINT20 0 = PCLK 1= EXTCLK/OSC_CLK (Selected by OM pin)
EINTFLT20	[6:0]	Filter width of EINT20

EXTERNAL INTERRUPT MASK REGISTER (EINTMASK)

Interrupt mask register for 20 external interrupts (EINT[23:4]).

Register	Address	R/W	Description	Reset Value
EINTMASK	0x560000A4	R/W	External interupt mask register	0x00FFFFF0

EINTMASK	Bit	Description
EINT23	[23]	0 = Enable Interrupt 1= Masked
EINT22	[22]	0 = Enable Interrupt 1= Masked
EINT21	[21]	0 = Enable Interrupt 1= Masked
EINT20	[20]	0 = Enable Interrupt 1= Masked
EINT19	[19]	0 = Enable Interrupt 1= Masked
EINT18	[18]	0 = Enable Interrupt 1= Masked
EINT17	[17]	0 = Enable Interrupt 1= Masked
EINT16	[16]	0 = Enable Interrupt 1= Masked
EINT15	[15]	0 = Enable Interrupt 1= Masked
EINT14	[14]	0 = Enable Interrupt 1= Masked
EINT13	[13]	0 = Enable Interrupt 1= Masked
EINT12	[12]	0 = Enable Interrupt 1= Masked
EINT11	[11]	0 = Enable Interrupt 1= Masked
EINT10	[10]	0 = Enable Interrupt 1= Masked
EINT9	[9]	0 = Enable Interrupt 1= Masked
EINT8	[8]	0 = Enable Interrupt 1= Masked
EINT7	[7]	0 = Enable Interrupt 1= Masked
EINT6	[6]	0 = Enable Interrupt 1= Masked
EINT5	[5]	0 = Enable Interrupt 1= Masked
EINT4	[4]	0 = Enable Interrupt 1= Masked
Reserved	[3:0]	0

EXTERNAL INTERRUPT PENDING REGISTER (EINTPENDn)

Interrupt pending register for 20 external interrupts (EINT[23:4]). You can clear a specific bit of the ENITPEND register by writing "1" on the corresponding bit of this register.

Register	Address	R/W	Description	Reset Value
EINTPEND	0x560000A8	R/W	External interupt pending register	0x0

EINTPEND	Bit	Description
EINT23	[23]	0 = Not requested 1= Requested
EINT22	[22]	0 = Not requested 1= Requested
EINT21	[21]	0 = Not requested 1= Requested
EINT20	[20]	0 = Not requested 1= Requested
EINT19	[19]	0 = Not requested 1= Requested
EINT18	[18]	0 = Not requested 1= Requested
EINT17	[17]	0 = Not requested 1= Requested
EINT16	[16]	0 = Not requested 1= Requested
EINT15	[15]	0 = Not requested 1= Requested
EINT14	[14]	0 = Not requested 1= Requested
EINT13	[13]	0 = Not requested 1= Requested
EINT12	[12]	0 = Not requested 1= Requested
EINT11	[11]	0 = Not requested 1= Requested
EINT10	[10]	0 = Not requested 1= Requested
EINT9	[9]	0 = Not requested 1= Requested
EINT8	[8]	0 = Not requested 1= Requested
EINT7	[7]	0 = Not requested 1= Requested
EINT6	[6]	0 = Not requested 1= Requested
EINT5	[5]	0 = Not requested 1= Requested
EINT4	[4]	0 = Not requested 1= Requested
Reserved	[3:0]	0

GENERAL STATUS REGISTER (GSTATUSn)

Register	Address	R/W	Description	Reset Value
GSTATUS0	0x560000AC	R	External pin status	Undefined
GSTATUS1	0x560000B0	R	Chip ID	0x32410000
GSTATUS2	0x560000B4	R/W	Reset status	0x1
GSTATUS3	0x560000B8	R/W	Infrom register	0x0
GSTATUS4	0x560000BC	R/W	Infrom register	0x0

GSTATUS0	Bit	Description
nWAIT	[3]	Status of nWAIT pin
NCON	[2]	Status of NCON pin
RnB	[1]	Status of R/nB pin
nBATT_FLT	[0]	Status of nBATT_FLT pin

GSTATUS1	Bit	Description
CHIP ID	[31:0]	ID register = 0x32410002

GSTATUS2	Bit	Description
PWRST	[0]	Power on reset, if this bit is set to "1". The setting is cleared by writing "1" to this bit.
OFFRST	[1]	Power_OFF reset. The reset after the wakeup from Power_OFF mode. The setting is cleared by writing "1" to this bit.
WDTRST	[2]	Watchdog reset. The reset derived from Watchdog timer. The setting is cleared by writing "1" to this bit.

GSTATUS3	Bit	Description
INFORM	[31:0]	Inform register. This register is cleared by nRESET or watchdog timer. Otherwise, preserve data value.

GSTATUS4	Bit	Description
INFORM	[31:0]	Inform register. This register is cleared by nRESET or watchdog timer. Otherwise, preserve data value.

10 PWM TIMER

OVERVIEW

The S3C2410A has five 16-bit timers. Timer 0, 1, 2, and 3 have Pulse Width Modulation (PWM) function. Timer 4 has an internal timer only with no output pins. The timer 0 has a dead-zone generator, which is used with a large current device.

The timer 0 and 1 share an 8-bit prescaler, while the timer 2, 3 and 4 share other 8-bit prescaler. Each timer has a clock divider which 5 different divided signals (1/2, 1/4, 1/8, 1/16, and TCLK). Each timer block receives its own clock signals from the clock divider, which receives the clock from the corresponding 8-bit prescaler. The 8-bit prescaler is programmable and divides the PCLK according to the loading value, which is stored in TCFG0 and TCFG1 registers.

The timer count buffer register (TCNTBn) has an initial value which is loaded into the down-counter when the timer is enabled. The timer compare buffer register (TCMPBn) has an initial value which is loaded into the compare register to be compared with the down-counter value. This double buffering feature of TCNTBn and TCMPBn makes the timer generate a stable output when the frequency and duty ratio are changed.

Each timer has its own 16-bit down counter, which is driven by the timer clock. When the down counter reaches zero, the timer interrupt request is generated to inform the CPU that the timer operation has been completed. When the timer counter reaches zero, the value of corresponding TCNTBn is automatically loaded into the down counter to continue the next operation. However, if the timer stops, for example, by clearing the timer enable bit of TCONn during the timer running mode, the value of TCNTBn will not be reloaded into the counter.

The value of TCMPBn is used for pulse width modulation (PWM). The timer control logic changes the output level when the down-counter value matches the value of the compare register in the timer control logic. Therefore, the compare register determines the turn-on time (or turn-off time) of an PWM output.

FEATURE

— Five 16-bit timers

— Two 8-bit prescalers & Two 4-bit divider

— Programmable duty control of output waveform (PWM)

— Auto reload mode or one-shot pulse mode

— Dead-zone generator

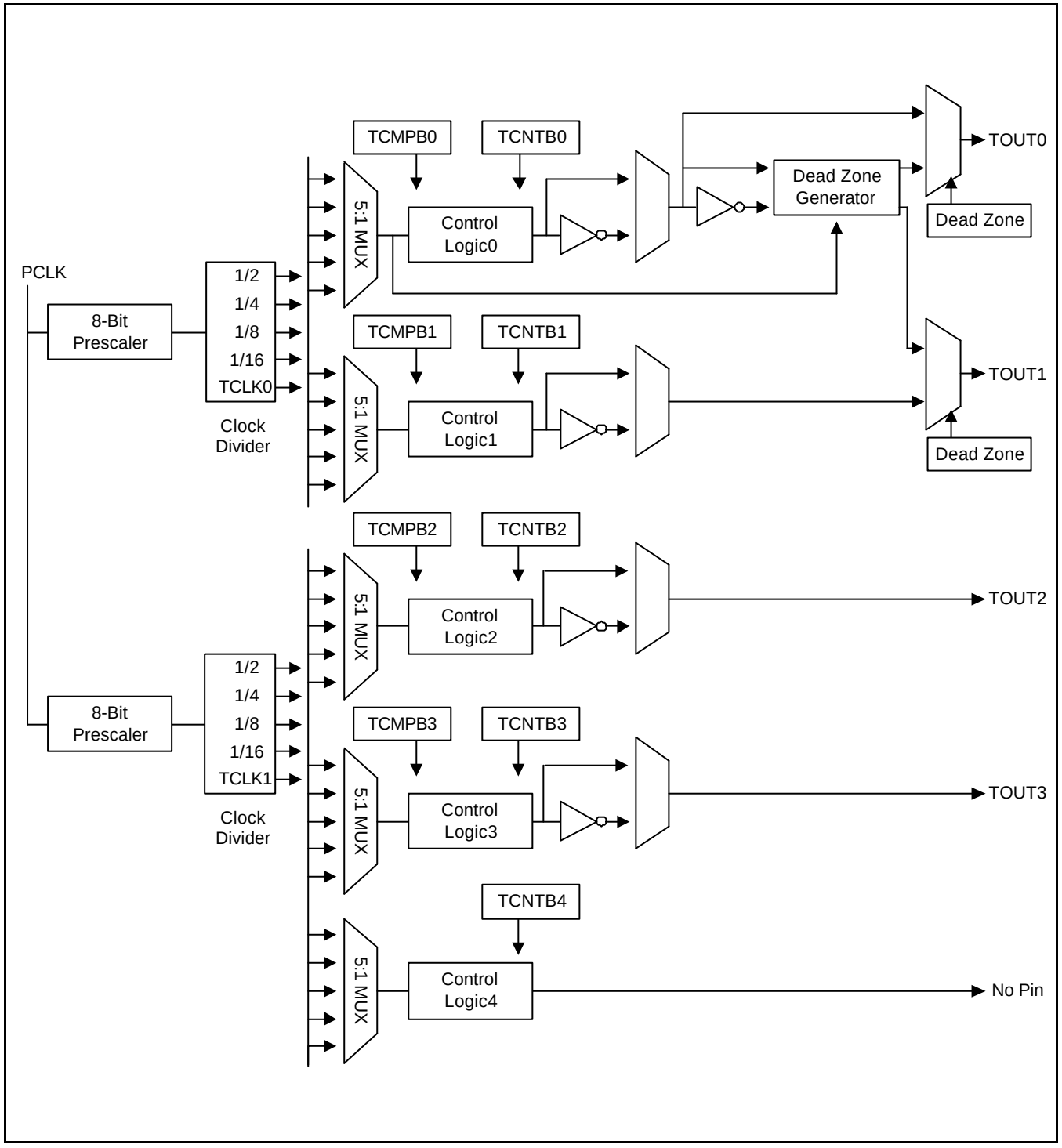


Figure 10-1. 16-bit PWM Timer Block Diagram

PWM TIMER OPERATION

PRESCALER & DIVIDER

An 8-bit prescaler and a 4-bit divider make the following output frequencies:

4-bit divider settings	Minimum resolution (prescaler = 0)	Maximum resolution (prescaler = 255)	Maximum interval (TCNTBn = 65535)
1/2 (PCLK = 66.5 MHz)	0.0300 us (33.2500 MHz)	7.6992 us (129.8828 KHz)	0.5045 sec
1/4 (PCLK = 66.5 MHz)	0.0601 us (16.6250 MHz)	15.3984 us (64.9414 KHz)	1.0091 sec
1/8 (PCLK = 66.5 MHz)	0.1203 us (8.3125 MHz)	30.7968 us (32.4707 KHz)	2.0182 sec
1/16 (PCLK = 66.5 MHz)	0.2406 us (4.1562 MHz)	61.5936 us (16.2353 KHz)	4.0365 sec

BASIC TIMER OPERATION

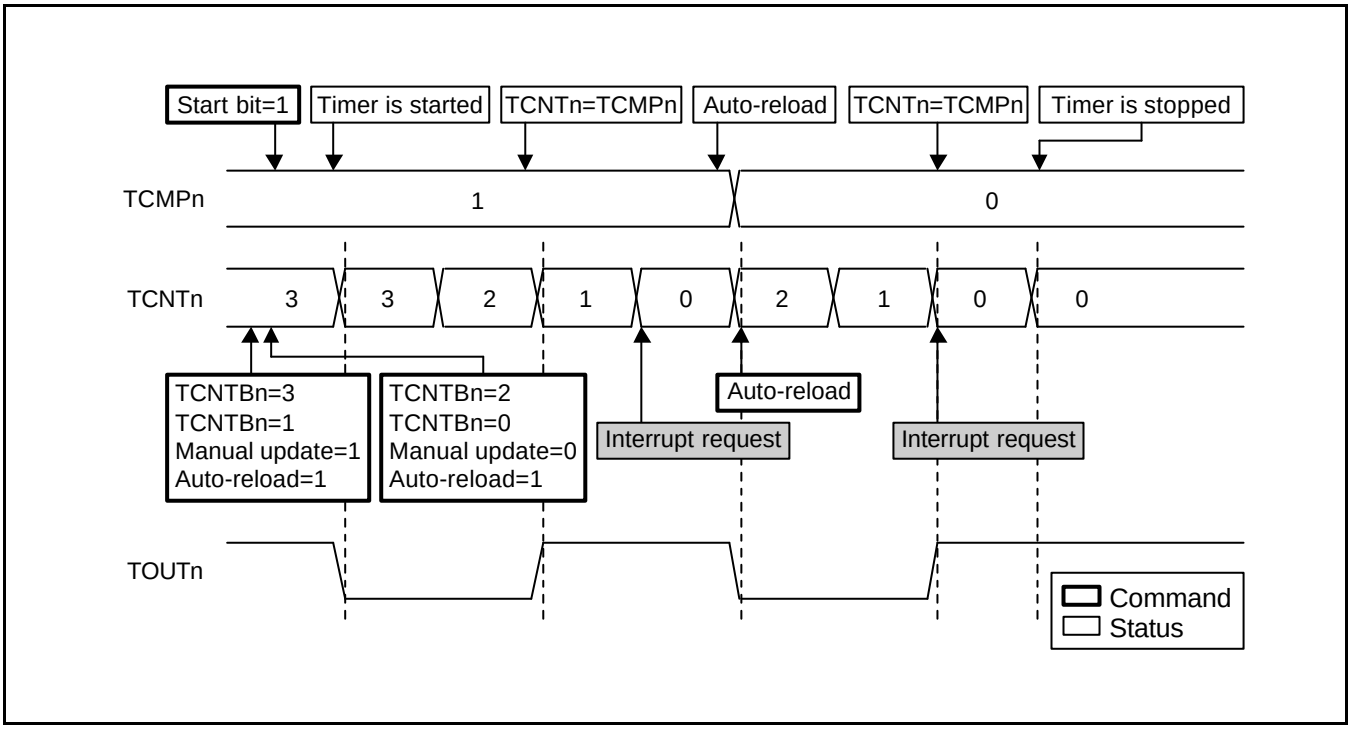


Figure 10-2. Timer Operations

A timer (except the timer ch-5) has TCNTBn, TCNTn, TCMPBn and TCMPn. (TCNTn and TCMPn are the names of the internal registers. The TCNTn register can be read from the TCNTOn register) The TCNTBn and the TCMPBn are loaded into the TCNTn and the TCMPn when the timer reaches 0. When the TCNTn reaches 0, an interrupt request will occur if the interrupt is enabled.

AUTO RELOAD & DOUBLE BUFFERING

S3C2410A PWM Timers have a double buffering function, enabling the reload value changed for the next timer operation without stopping the current timer operation. So, although the new timer value is set, a current timer operation is completed successfully.

The timer value can be written into Timer Count Buffer register (TCNTBn) and the current counter value of the timer can be read from Timer Count Observation register (TCNTOn). If the TCNTBn is read, the read value does not indicate the current state of the counter but the reload value for the next timer duration.

The auto reload operation copies the TCNTBn into TCNTn when the TCNTn reaches 0. The value, written into the TCNTBn, is loaded to the TCNTn only when the TCNTn reaches 0 and auto reload is enabled. If the TCNTn becomes 0 and the auto reload bit is 0, the TCNTn does not operate any further.

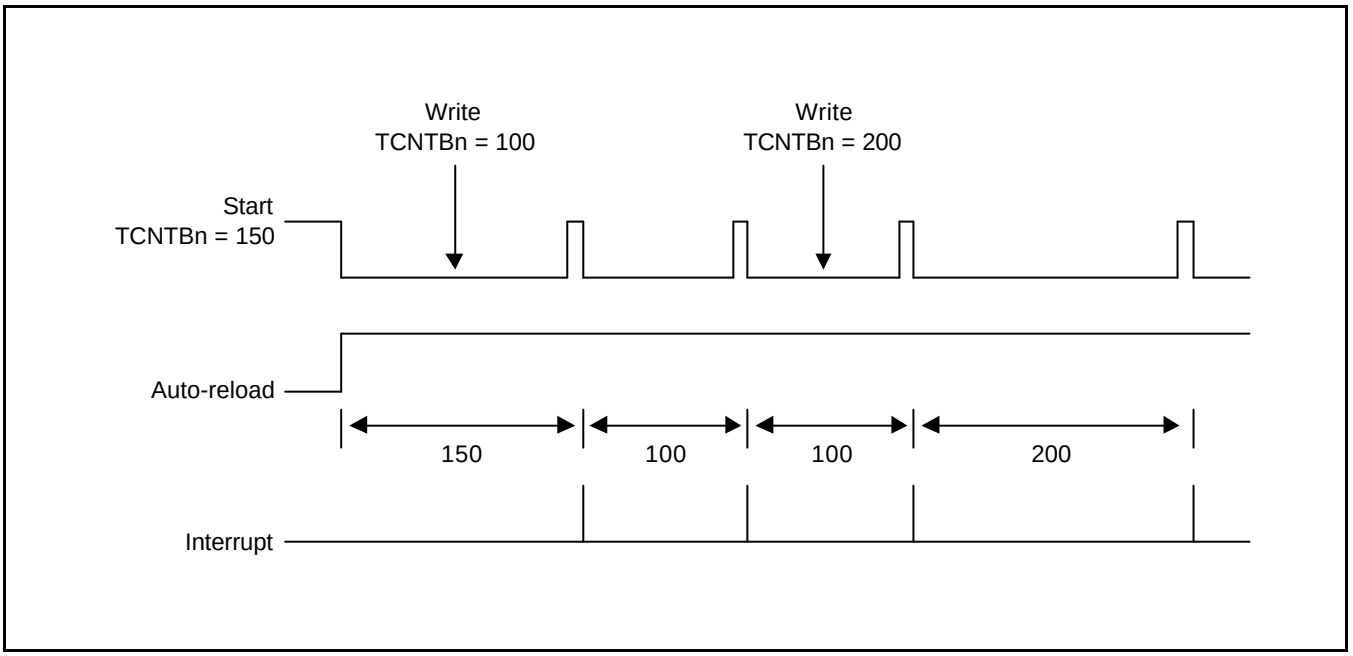


Figure 10-3. Example of Double Buffering Function

TIMER INITIALIZATION USING MANUAL UPDATE BIT AND INVERTER BIT

An auto reload operation of the timer occurs when the down counter reaches 0. So, a starting value of the TCNTn has to be defined by the user in advance. In this case, the starting value has to be loaded by the manual update bit. The following steps describe how to start a timer:

1) Write the initial value into TCNTBn and TCMPBn.
2) Set the manual update bit of the corresponding timer. It is recommended that you configure the inverter on/off bit. (whether use inverter or not).
3) Set start bit of the corresponding timer to start the timer (and clear the manual update bit).

If the timer is stopped by force, the TCNTn retains the counter value and is not reloaded from TCNTBn. If a new value has to be set, perform manual update.

NOTE

Whenever TOUT inverter on/off bit is changed, the TOUTn logic value will also be changed whether the timer runs. Therefore, it is desirable that the inverter on/off bit is configured with the manual update bit.

TIMER OPERATION

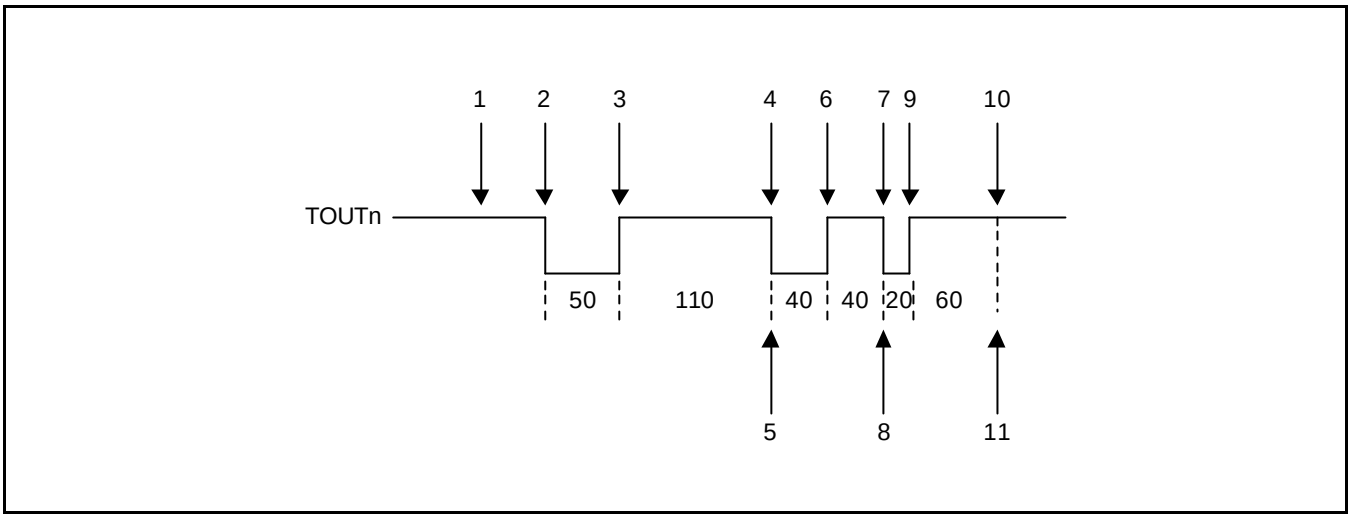


Figure 10-4. Example of a Timer Operation

Figure 10-4 shows the result of the following procedure:

1. Enable the auto reload function. Set the TCNTBn to 160 (50+110) and the TCMPBn to 110. Set the manual update bit and configure the inverter bit (on/off). The manual update bit sets TCNTn and TCMPn to the values of TCNTBn and TCMPBn, respectively.

   And then, set the TCNTBn and the TCMPBn to 80 (40+40) and 40, respectively, to determine the next reload value.
2. Set the start bit, provided that manual_update is 0 and the inverter is off and auto reload is on. The timer starts counting down after latency time within the timer resolution.
3. When the TCNTn has the same value as that of the TCMPn, the logic level of the TOUTn is changed from low to high.
4. When the TCNTn reaches 0, the interrupt request is generated and TCNTBn value is loaded into a temporary register. At the next timer tick, the TCNTn is reloaded with the temporary register value (TCNTBn).
5. In Interrupt Service Routine (ISR), the TCNTBn and the TCMPBn are set to 80 (20+60) and 60, respectively, for the next duration.
6. When the TCNTn has the same value as the TCMPn, the logic level of TOUTn is changed from low to high.
7. When the TCNTn reaches 0, the TCNTn is reloaded automatically with the TCNTBn, triggering an interrupt request.
8. In Interrupt Service Routine (ISR), auto reload and interrupt request are disabled to stop the timer.
9. When the value of the TCNTn is same as the TCMPn, the logic level of the TOUTn is changed from low to high.
10. Even when the TCNTn reaches 0, the TCNTn is not any more reloaded and the timer is stopped because auto reload has been disabled.
11. No more interrupt requests are generated.

PULSE WIDTH MODULATION (PWM)

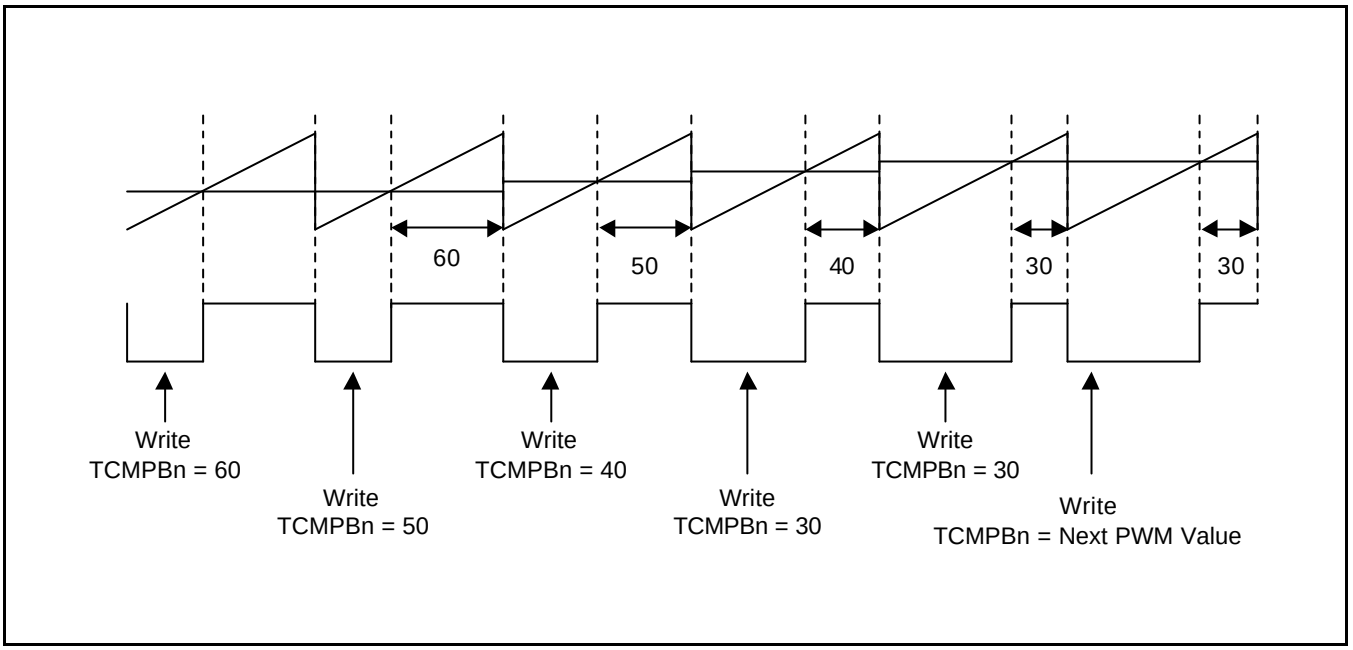


Figure 10-5. Example of PWM

PWM function can be implemented by using the TCMPBn. PWM frequency is determined by TCNTBn. Figure 10-5 shows a PWM value determined by TCMPBn.

For a higher PWM value, decrease the TCMPBn value. For a lower PWM value, increase the TCMPBn value. If an output inverter is enabled, the increment/decrement may be reversed.

The double buffering function allows the TCMPBn, for the next PWM cycle, written at any point in the current PWM cycle by ISR or other routine.

OUTPUT LEVEL CONTROL

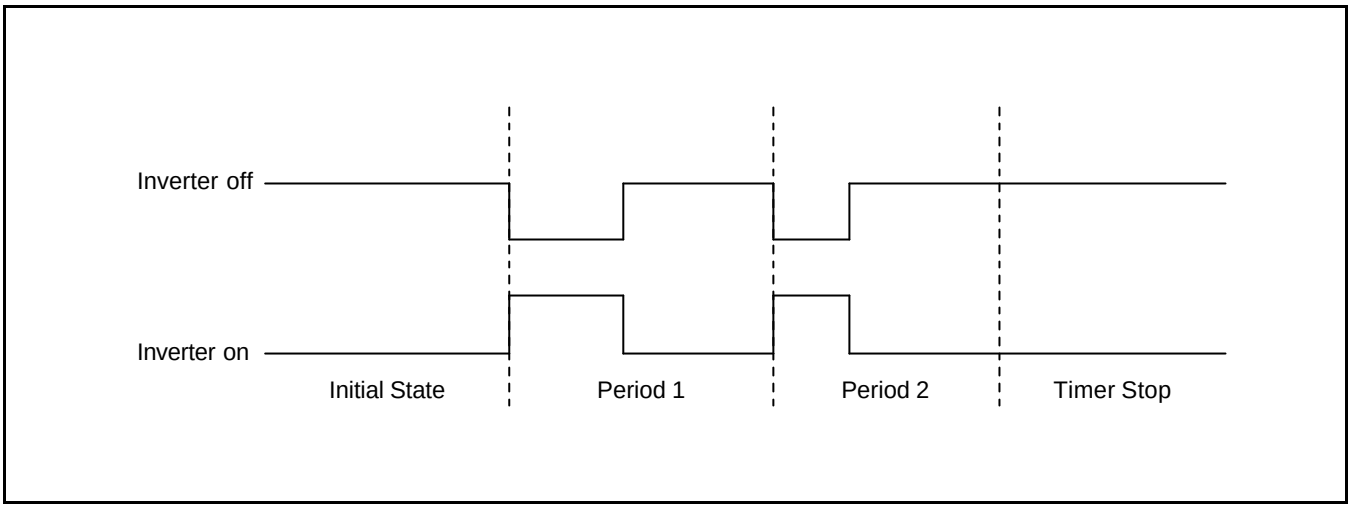


Figure 10-6. Inverter On/Off

The following procedure describes how to maintain TOUT as high or low (assume the inverter is off):

1. Turn off the auto reload bit. And then, TOUTn goes to high level and the timer is stopped after the TCNTn reaches 0 (recommended).
2. Stop the timer by clearing the timer start/stop bit to 0. If TCNTn ≤ TCMPn, the output level is high. If TCNTn >TCMPn, the output level is **low**.
3. The TOUTn can be inverted by the inverter on/off bit in TCON. The inverter removes the additional circuit to adjust the output level.

DEAD ZONE GENERATOR

The dead zone is for the PWM control in a power device. This function enables the insertion of the time gap between a turn-off of a switching device and a turn on of another switching device. This time gap prohibits the two switching devices from being turned on simultaneously, even for a very short time.

TOUT0 is the PWM output. nTOUT0 is the inversion of the TOUT0. If the dead zone is enabled, the output wave form of TOUT0 and nTOUT0 will be TOUT0_DZ and nTOUT0_DZ, respectively. nTOUT0_DZ is routed to the TOUT1 pin.

In the dead zone interval, TOUT0_DZ and nTOUT0_DZ can never be turned on simultaneously.

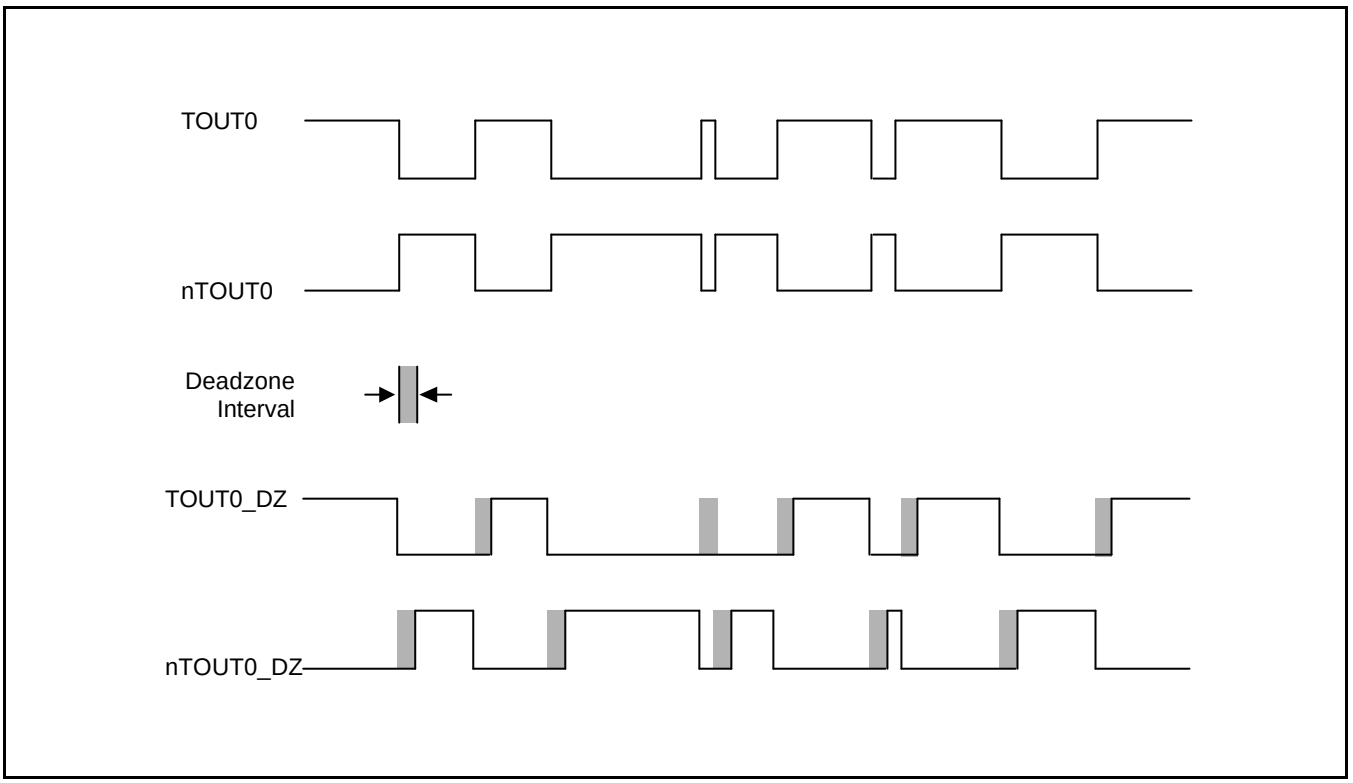


Figure 10-7. The Wave Form When a Dead Zone Feature is Enabled

DMA REQUEST MODE

The PWM timer can generate a DMA request at every specific time. The timer keeps DMA request signals (nDMA_REQ) low until the timer receives an ACK signal. When the timer receives the ACK signal, it makes the request signal inactive. The timer, which generates the DMA request, is determined by setting DMA mode bits (in TCFG1 register). If one of timers is configured as DMA request mode, that timer does not generate an interrupt request. The others can generate interrupt normally.

DMA mode configuration and DMA / interrupt operation

DMA Mode	DMA Request	Timer0 INT	Timer1 INT	Timer2 INT	Timer3 INT	Timer4 INT
0000	No select	ON	ON	ON	ON	ON
0001	Timer0	OFF	ON	ON	ON	ON
0010	Timer1	ON	OFF	ON	ON	ON
0011	Timer2	ON	ON	OFF	ON	ON
0100	Timer3	ON	ON	ON	OFF	ON
0101	Timer4	ON	ON	ON	ON	OFF
0110	No select	ON	ON	ON	ON	ON

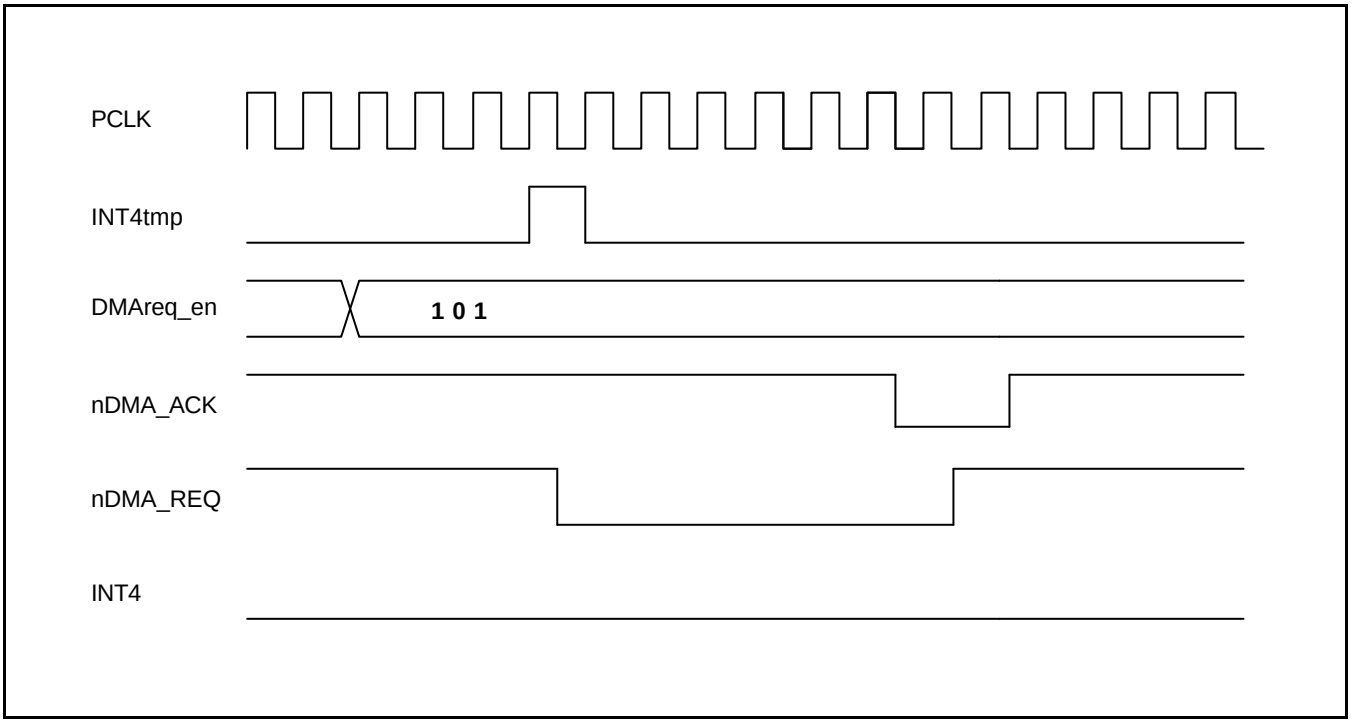


Figure 10-8. Timer4 DMA Mode Operation

PWM TIMER CONTROL REGISTERS

TIMER CONFIGURATION REGISTER0 (TCFG0)

Timer input clock Frequency = PCLK / {prescaler value+1} / {divider value}
{prescaler value} = 0~255
{divider value} = 2, 4, 8, 16

Register	Address	R/W	Description	Reset Value
TCFG0	0x51000000	R/W	Configures the two 8-bit prescalers	0x00000000

TCFG0	Bit	Description	Initial State
Reserved	[31:24]		0x00
Dead zone length	[23:16]	These 8 bits determine the dead zone length. The 1 unit time of the dead zone length is equal to that of timer 0.	0x00
Prescaler 1	[15:8]	These 8 bits determine prescaler value for Timer 2, 3 and 4.	0x00
Prescaler 0	[7:0]	These 8 bits determine prescaler value for Timer 0 and 1.	0x00

TIMER CONFIGURATION REGISTER1 (TCFG1)

Register	Address	R/W	Description	Reset Value
TCFG1	0x51000004	R/W	5-MUX & DMA mode selecton register	0x00000000

TCFG1	Bit	Description	Initial State
Reserved	[31:24]		00000000
DMA mode	[23:20]	Select DMA request channel 0000 = No select (all interrupt) 0001 = Timer0 0010 = Timer1 0011 = Timer2 0100 = Timer3 0101 = Timer4 0110 = Reserved	0000
MUX 4	[19:16]	Select MUX input for PWM Timer4. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = External TCLK1	0000
MUX 3	[15:12]	Select MUX input for PWM Timer3. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = External TCLK1	0000
MUX 2	[11:8]	Select MUX input for PWM Timer2. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = External TCLK1	0000
MUX 1	[7:4]	Select MUX input for PWM Timer1. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = External TCLK0	0000
MUX 0	[3:0]	Select MUX input for PWM Timer0. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = External TCLK0	0000

TIMER CONTROL (TCON) REGISTER

Register	Address	R/W	Description	Reset Value
TCON	0x51000008	R/W	Timer control register	0x00000000

TCON	Bit	Description	Initial state
Timer 4 auto reload on/off	[22]	Determine auto reload on/off for Timer 4. 0 = One-shot 1 = Interval mode (auto reload)	0
Timer 4 manual update (note)	[21]	Determine the manual update for Timer 4. 0 = No operation 1 = Update TCNTB4	0
Timer 4 start/stop	[20]	Determine start/stop for Timer 4. 0 = Stop 1 = Start for Timer 4	0
Timer 3 auto reload on/off	[19]	Determine auto reload on/off for Timer 3. 0 = One-shot 1 = Interval mode (auto reload)	0
Timer 3 output inverter on/off	[18]	Determine output inverter on/off for Timer 3. 0 = Inverter off 1 = Inverter on for TOUT3	0
Timer 3 manual update (note)	[17]	Determine manual update for Timer 3. 0 = No operation 1 = Update TCNTB3 & TCMPB3	0
Timer 3 start/stop	[16]	Determine start/stop for Timer 3. 0 = Stop 1 = Start for Timer 3	0
Timer 2 auto reload on/off	[15]	Determine auto reload on/off for Timer 2. 0 = One-shot 1 = Interval mode (auto reload)	0
Timer 2 output inverter on/off	[14]	Determine output inverter on/off for Timer 2. 0 = Inverter off 1 = Inverter on for TOUT2	0
Timer 2 manual update (note)	[13]	Determine the manual update for Timer 2. 0 = No operation 1 = Update TCNTB2 & TCMPB2	0
Timer 2 start/stop	[12]	Determine start/stop for Timer 2. 0 = Stop 1 = Start for Timer 2	0
Timer 1 auto reload on/off	[11]	Determine the auto reload on/off for Timer1. 0 = One-shot 1 = Interval mode (auto reload)	0
Timer 1 output inverter on/off	[10]	Determine the output inverter on/off for Timer1. 0 = Inverter off 1 = Inverter on for TOUT1	0
Timer 1 manual update (note)	[9]	Determine the manual update for Timer 1. 0 = No operation 1 = Update TCNTB1 & TCMPB1	0
Timer 1 start/stop	[8]	Determine start/stop for Timer 1. 0 = Stop 1 = Start for Timer 1	0

NOTE: The bits have to be cleared at next writing.

TIMER CONTROL (TCON) REGISTER (Continued)

TCON	Bit	Description	Initial state
Reserved	[7:5]	Reserved	
Dead zone enable	[4]	Determine the dead zone operation. 0 = Disable 1 = Enable	0
Timer 0 auto reload on/off	[3]	Determine auto reload on/off for Timer 0. 0 = One-shot 1 = Interval mode(auto reload)	0
Timer 0 output inverter on/off	[2]	Determine the output inverter on/off for Timer 0. 0 = Inverter off 1 = Inverter on for TOUT0	0
Timer 0 manual update (note)	[1]	Determine the manual update for Timer 0. 0 = No operation 1 = Update TCNTB0 & TCMPB0	0
Timer 0 start/stop	[0]	Determine start/stop for Timer 0. 0 = Stop 1 = Start for Timer 0	0

NOTE: The bit have to be cleared at next writing.

TIMER 0 COUNT BUFFER REGISTER & COMPARE BUFFER REGISTER (TCNTB0/TCMPB0)

Register	Address	R/W	Description	Reset Value
TCNTB0	0x5100000C	R/W	Timer 0 count buffer register	0x00000000
TCMPB0	0x51000010	R/W	Timer 0 compare buffer register	0x00000000

TCMPB0	Bit	Description	Initial State
Timer 0 compare buffer register	[15:0]	Set compare buffer value for Timer 0	0x00000000

TCNTB0	Bit	Description	Initial State
Timer 0 count buffer register	[15:0]	Set count buffer value for Timer 0	0x00000000

TIMER 0 COUNT OBSERVATION REGISTER (TCNTO0)

Register	Address	R/W	Description	Reset Value
TCNTO0	0x51000014	R	Timer 0 count observation register	0x00000000

TCNTO0	Bit	Description	Initial State
Timer 0 observation register	[15:0]	Set count observation value for Timer 0	0x00000000

TIMER 1 COUNT BUFFER REGISTER & COMPARE BUFFER REGISTER (TCNTB1/TCMPB1)

Register	Address	R/W	Description	Reset Value
TCNTB1	0x51000018	R/W	Timer 1 count buffer register	0x00000000
TCMPB1	0x5100001C	R/W	Timer 1 compare buffer register	0x00000000

TCMPB1	Bit	Description	Initial State
Timer 1 compare buffer register	[15:0]	Set compare buffer value for Timer 1	0x00000000

TCNTB1	Bit	Description	Initial State
Timer 1 count buffer register	[15:0]	Set count buffer value for Timer 1	0x00000000

TIMER 1 COUNT OBSERVATION REGISTER (TCNTO1)

Register	Address	R/W	Description	Reset Value
TCNTO1	0x51000020	R	Timer 1 count observation register	0x00000000

TCNTO1	Bit	Description	initial state
Timer 1 observation register	[15:0]	Set count observation value for Timer 1	0x00000000

TIMER 2 COUNT BUFFER REGISTER & COMPARE BUFFER REGISTER (TCNTB2/TCMPB2)

Register	Address	R/W	Description	Reset Value
TCNTB2	0x51000024	R/W	Timer 2 count buffer register	0x00000000
TCMPB2	0x51000028	R/W	Timer 2 compare buffer register	0x00000000

TCMPB2	Bit	Description	Initial State
Timer 2 compare buffer register	[15:0]	Set compare buffer value for Timer 2	0x00000000

TCNTB2	Bit	Description	Initial State
Timer 2 count buffer register	[15:0]	Set count buffer value for Timer 2	0x00000000

TIMER 2 COUNT OBSERVATION REGISTER (TCNTO2)

Register	Address	R/W	Description	Reset Value
TCNTO2	0x5100002C	R	Timer 2 count observation register	0x00000000

TCNTO2	Bit	Description	Initial State
Timer 2 observation register	[15:0]	Set count observation value for Timer 2	0x00000000

TIMER 3 COUNT BUFFER REGISTER & COMPARE BUFFER REGISTER (TCNTB3/TCMPB3)

Register	Address	R/W	Description	Reset Value
TCNTB3	0x51000030	R/W	Timer 3 count buffer register	0x00000000
TCMPB3	0x51000034	R/W	Timer 3 compare buffer register	0x00000000

TCMPB3	Bit	Description	Initial State
Timer 3 compare buffer register	[15:0]	Set compare buffer value for Timer 3	0x00000000

TCNTB3	Bit	Description	Initial State
Timer 3 count buffer register	[15:0]	Set count buffer value for Timer 3	0x00000000

TIMER 3 COUNT OBSERVATION REGISTER (TCNTO3)

Register	Address	R/W	Description	Reset Value
TCNTO3	0x51000038	R	Timer 3 count observation register	0x00000000

TCNTO3	Bit	Description	Initial State
Timer 3 observation register	[15:0]	Set count observation value for Timer 3	0x00000000

TIMER 4 COUNT BUFFER REGISTER (TCNTB4)

Register	Address	R/W	Description	Reset Value
TCNTB4	0x5100003C	R/W	Timer 4 count buffer register	0x00000000

TCNTB4	Bit	Description	Initial State
Timer 4 count buffer register	[15:0]	Set count buffer value for Timer 4	0x00000000

TIMER 4 COUNT OBSERVATION REGISTER (TCNTO4)

Register	Address	R/W	Description	Reset Value
TCNTO4	0x51000040	R	Timer 4 count observation register	0x00000000

TCNTO4	Bit	Description	Initial State
Timer 4 observation register	[15:0]	Set count observation value for Timer 4	0x00000000

NOTES

11 UART

OVERVIEW

The S3C2410A UART (Universal Asynchronous Receiver and Transmitter) provides three independent asynchronous serial I/O (SIO) ports, each of which can operate in Interrupt-based or DMA-based mode. In other words, the UART can generate an interrupt or a DMA request to transfer data between CPU and the UART. The UART can support bit rates of up to 230.4K bps using system clock. If an external device provides the UART with UEXTCLK, then the UART can operate at higher speed. Each UART channel contains two 16-byte FIFOs for receive and transmit.

The S3C2410A UART includes programmable baud rates, infra-red (IR) transmit/receive, one or two stop bit insertion, 5-bit, 6-bit, 7-bit or 8-bit data width and parity checking.

Each UART contains a baud-rate generator, a transmitter, a receiver and a control unit, as shown in Figure11-1. The baud-rate generator can be clocked by PCLK or UEXTCLK. The transmitter and the receiver contain 16-byte FIFOs and data shifters. Data is written to FIFO and then copied to the transmit shifter before being transmitted. The data is then shifted out by the transmit data pin (TxDn). Meanwhile, received data is shifted from the receive data pin (RxDn), and then copied to FIFO from the shifter.

FEATURES

— RxD0, TxD0, RxD1, TxD1, RxD2, and TxD2 with DMA-based or interrupt-based operation

— UART Ch 0, 1, and 2 with IrDA 1.0 & 16-byte FIFO

— UART Ch 0 and 1 with nRTS0, nCTS0, nRTS1, and nCTS1

— Supports handshake transmit/receive

BLOCK DIAGRAM

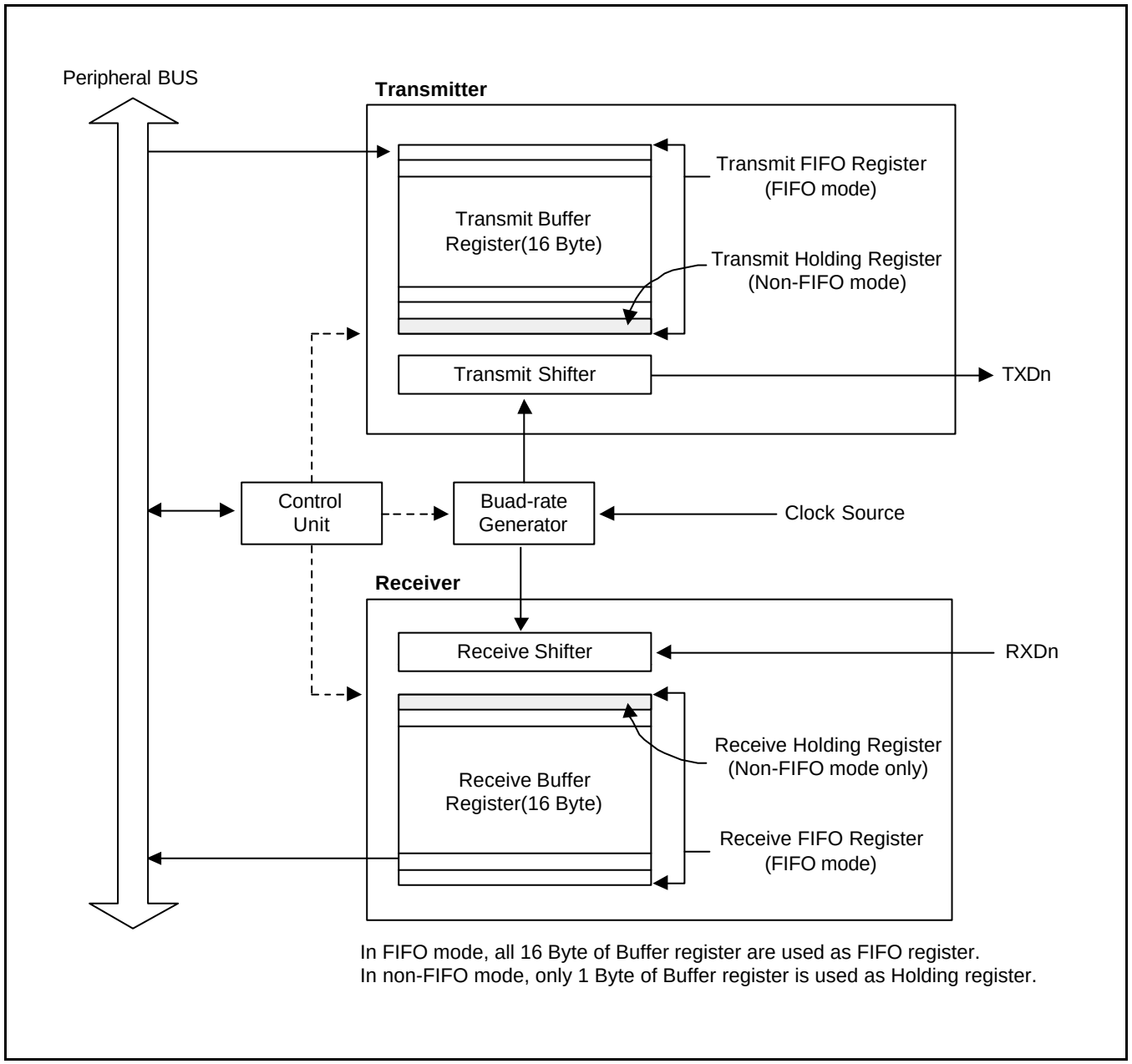


Figure 11-1. UART Block Diagram (with FIFO)

UART OPERATION

The following sections describe the UART operations that include data transmission, data reception, interrupt generation, baud-rate generation, Loopback mode, Infra-red mode, and auto flow control.

Data Transmission

The data frame for transmission is programmable. It consists of a start bit, 5 to 8 data bits, an optional parity bit and 1 to 2 stop bits, which can be specified by the line control register (ULCONn). The transmitter can also produce the break condition, which forces the serial output to logic 0 state for one frame transmission time. This block transmits break signals after the present transmission word is transmitted completely. After the break signal transmission, it continuously transmits data into the Tx FIFO (Tx holding register in the case of Non-FIFO mode).

Data Reception

Like the transmission, the data frame for reception is also programmable. It consists of a start bit, 5 to 8 data bits, an optional parity bit and 1 to 2 stop bits in the line control register (ULCONn). The receiver can detect overrun error and frame error.

— The overrun error indicates that new data has overwritten the old data before the old data has been read.

— The frame error indicates that the received data does not have a valid stop bit.

Receive time-out condition occurs when it does not receive any data during the 3 word time (this interval follows the setting of Word Length bit) and the Rx FIFO is not empty in the FIFO mode.

Auto Flow Control (AFC)

The S3C2410A's UART 0 and UART 1 support auto flow control with nRTS and nCTS signals. In case, it can be connected to external UARTs. If users want to connect a UART to a Modem, disable auto flow control bit in UMCONn register and control the signal of nRTS by software.

In AFC, nRTS depends on the condition of the receiver and nCTS signals control the operation of the transmitter. The UART's transmitter transfers the data in FIFO only when nCTS signals are activated (in AFC, nCTS means that other UART's FIFO is ready to receive data). Before the UART receives data, nRTS has to be activated when its receive FIFO has a spare more than 2-byte and has to be inactivated when its receive FIFO has a spare under 1-byte (in AFC, nRTS means that its own receive FIFO is ready to receive data).

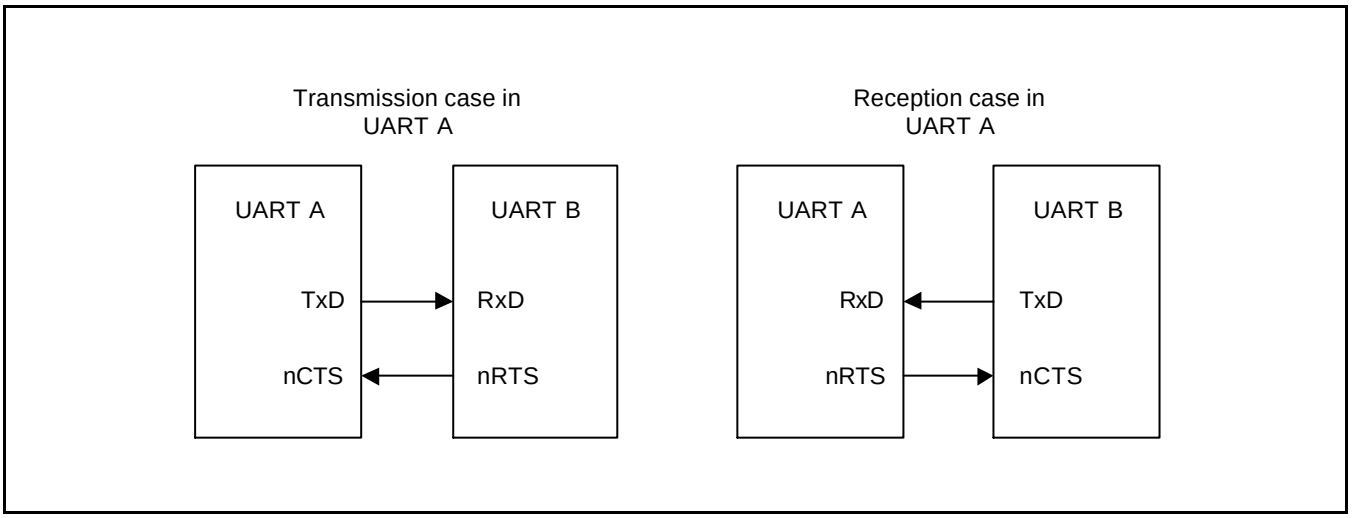


Figure 11-2. UART AFC Interface

NOTE

UART 2 does not support AFC function, because the S3C2410A has no nRTS2 and nCTS2.

Example of Non Auto-Flow control (controlling nRTS and nCTS by software)

Rx operation with FIFO

1. Select receive mode (Interrupt or DMA mode).
2. Check the value of Rx FIFO count in UFSTATn register. If the value is less than 15, users have to set the value of UMCONn[0] to '1' (activating nRTS), and if it is equal or larger than 15 users have to set the value to '0' (inactivating nRTS).
3. Repeat the Step 2.

Tx operation with FIFO

1. Select transmit mode (Interrupt or DMA mode).
2. Check the value of UMSTATn[0]. If the value is '1' (activating nCTS), users write the data to Tx FIFO register.

RS-232C interface

If users want to connect the UART to modem interface (instead of null modem), nRTS, nCTS, nDSR, nDTR, DCD and nRI signals are needed. In this case, the users can control these signals with general I/O ports by software because the AFC does not support the RS-232C interface.

Interrupt/DMA Request Generation

Each UART of the S3C2410A has five status (Tx/Rx/Error) signals: Overrun error, Frame error, Receive buffer data ready, Transmit buffer empty, and Transmit shifter empty, all of which are indicated by the corresponding UART status register (UTRSTATn/UERSTATn).

The overrun error and frame error are referred to as the receive error status, each of which can cause the receive error status interrupt request, if the receive-error-status-interrupt-enable bit is set to one in the control register, UCONn. When a receive-error-status-interrupt-request is detected, the signal causing the request can be identified by reading the value of UERSTSTn.

When the receiver transfers the data of the receive shifter to the receive FIFO register in FIFO mode and the number of received data reaches Rx FIFO Trigger Level, Rx interrupt is generated, if Receive mode in control register (UCONn) is selected as 1 (Interrupt request or polling mode).
In the Non-FIFO mode, transferring the data of the receive shifter to the receive holding register will cause Rx interrupt under the Interrupt request and polling mode.

When the transmitter transfers data from its transmit FIFO register to its transmit shifter and the number of data left in transmit FIFO reaches Tx FIFO Trigger Level, Tx interrupt is generated, if Transmit mode in control register is selected as Interrupt request or polling mode.
In the Non-FIFO mode, transferring data from the transmit holding register to the transmit shifter will cause Tx interrupt under the Interrupt request and polling mode.

If the Receive mode and Transmit mode in control register are selected as the DMAn request mode then DMAn request occurs instead of Rx or Tx interrupt in the situation mentioned above.

Table 11-1. Interrupts in Connection with FIFO

Type	FIFO Mode	Non-FIFO Mode
Rx interrupt	Generated whenever receive data reaches the trigger level of receive FIFO. Generated when the number of data in FIFO does not reaches Rx FIFO trigger Level and does not receive any data during 3 word time (receive time out). This interval follows the setting of Word Length bit.	Generated by the receive holding register whenever receive buffer becomes full.
Tx interrupt	Generated whenever transmit data reaches the trigger level of transmit FIFO (Tx FIFO trigger Level).	Generated by the transmit holding register whenever transmit buffer becomes empty.
Error interrupt	Generated when frame error has detected. Generated when it gets to the top of the receive FIFO without reading out data in it (overrun error).	Generated by all errors. However if another error occurs at the same time, only one interrupt is generated.

UART Error Status FIFO

UART has the error status FIFO besides the Rx FIFO register. The error status FIFO indicates which data, among FIFO registers, is received with an error. The error interrupt will be issued only when the data, which has an error, is ready to read out. To clear the error status FIFO, the URXHn with an error and UERSTATn must be read out.

For example,

It is assumed that the UART Rx FIFO receives A, B, C, and D characters sequentially and the frame error occurs while receiving 'B'.

The actual UART receive error will not generate any error interrupt because the character, which was received with an error, has not been read yet. The error interrupt will occur when the character is read out.

Figure 11-3 shows the UART receiving the four characters including the one error.

Time	Sequence Flow	Error Interrupt	Note
#0	When no character is read out	–	
#1	A, B, C, and D is received	–	
#2	After A is read out	The frame error (in B) interrupt occurs.	The 'B' has to be read out.
#3	After B is read out	–	
#4	After C is read out	–	
#5	After D is read out	–	

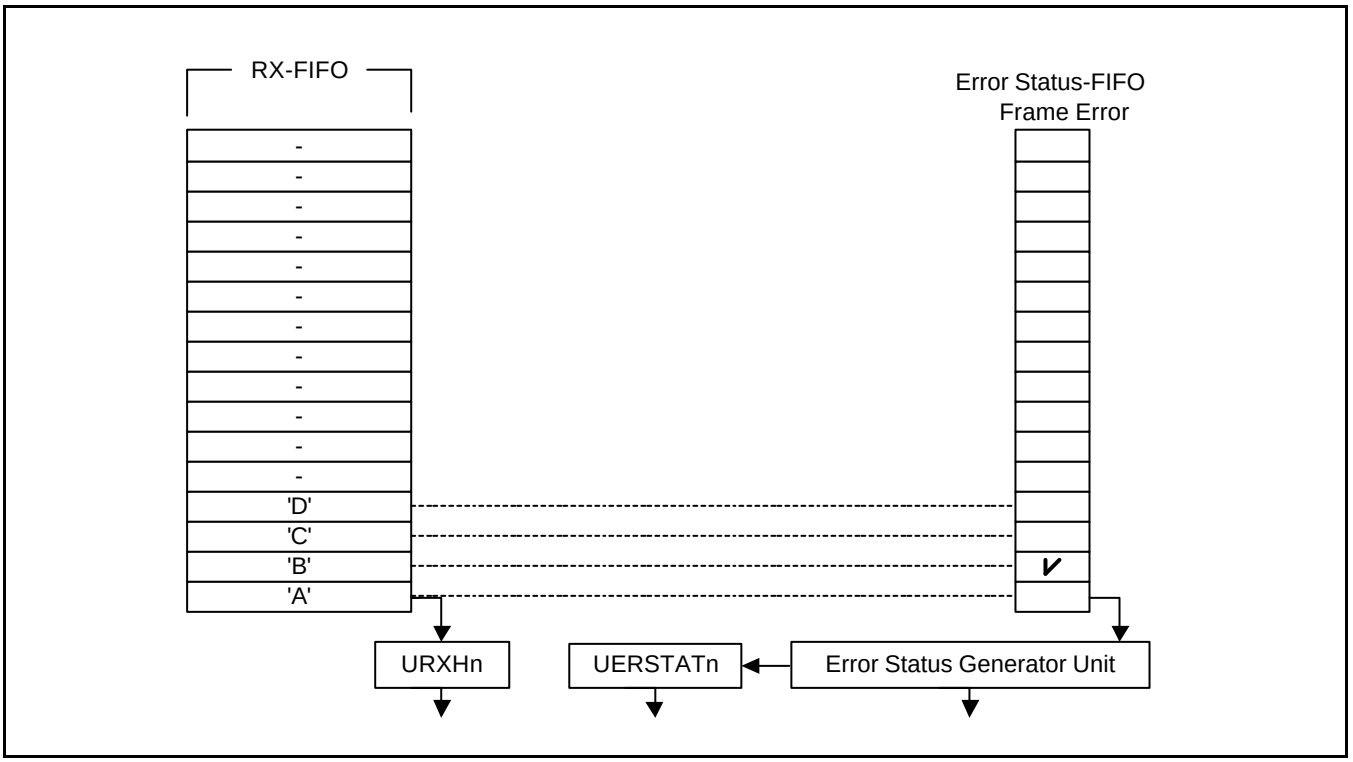


Figure 11-3. UART Receiving 4 Characters with 1 Error

Baud-Rate Generation

Each UART's baud-rate generator provides the serial clock for the transmitter and the receiver. The source clock for the baud-rate generator can be selected with the S3C2410A's internal system clock or UEXTCLK. In other words, dividend is selectable by setting Clock Selection of UCONn. The baud-rate clock is generated by dividing the source clock (PCLK or UEXTCLK) by 16 and a 16-bit divisor specified in the UART baud-rate divisor register (UBRDIVn). The UBRDIVn can be determined by the following expression:

UBRDIVn = (int)(PCLK/(bps x 16)) -1

Where, the divisor should be from 1 to (2^{16}-1).

For accurate UART operation, the S3C2410A also supports UEXTCLK as a dividend.
If the S3C2410A uses UEXTCLK, which is supplied by an external UART device or system, then the serial clock of UART is exactly synchronized with UEXTCLK. So, the user can get the more precise UART operation. The UBRDIVn can be determined:

UBRDIVn = (int)(UEXTCLK / (bps x 16)) –1

Where, the divisor should be from 1 to (2^{16}-1) and UEXTCLK should be smaller than PCLK.

For example, if the baud-rate is 115200 bps and PCLK or UEXTCLK is 40 MHz, UBRDIVn is determined:

UBRDIVn = (int)(40000000/(115200 x 16)) -1
= (int)(21.7) -1
= 21 -1 = 20

Baud-Rate Error Tolerance

UART Frame error should be less than 1.87%(3/160).

tUPCLK = (UBRDIVn + 1) x 16 x 1Frame / PCLK tUPCLK : Real UART Clock

tUEXACT = 1Frame / baud-rate tUEXACT : Ideal UART Clock

UART error = (tUPCLK – tUEXACT) / tUEXACT x 100%

NOTES

1. 1Frame = start bit + data bit + parity bit + stop bit.
2. In specific condition, we can support baud rate up to 921.6K bps. For example, when PCLK is 60MHz, you can use baud rate of 921.6K bps under UART error of 1.69%.

Loopback Mode

The S3C2410A UART provides a test mode referred to as the Loopback mode, to aid in isolating faults in the communication link. This mode structurally enables the connection of RXD and TXD in the UART. In this mode, therefore, transmitted data is received to the receiver, via RXD. This feature allows the processor to verify the internal transmit and to receive the data path of each SIO channel. This mode can be selected by setting the loopback bit in the UART control register (UCONn).

Infra-Red (IR) Mode

The S3C2410A UART block supports infra-red (IR) transmission and reception, which can be selected by setting the Infra-red-mode bit in the UART line control register (ULCONn). Figure 11-4 illustrates how to implement the IR mode.

In IR transmit mode, the transmit pulse comes out at a rate of 3/16, the normal serial transmit rate (when the transmit data bit is zero); In IR receive mode, the receiver must detect the 3/16 pulsed period to recognize a zero value (see the frame timing diagrams shown in Figure 11-6 and 11-7).

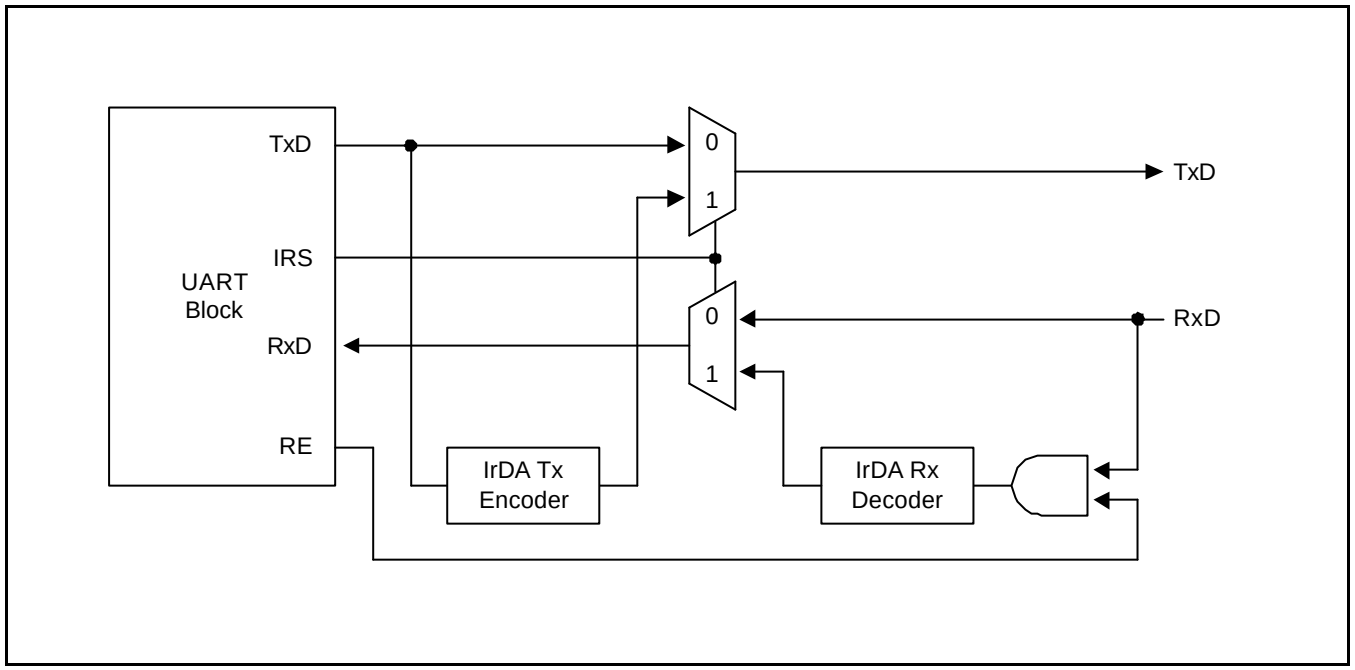


Figure 11-4. IrDA Function Block Diagram

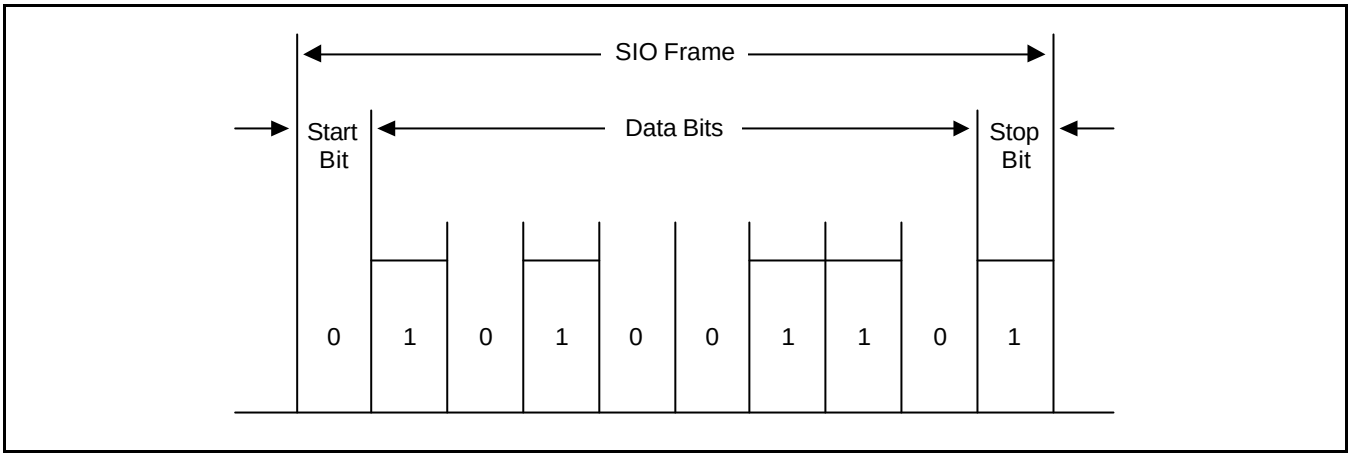


Figure 11-5. Serial I/O Frame Timing Diagram (Normal UART)

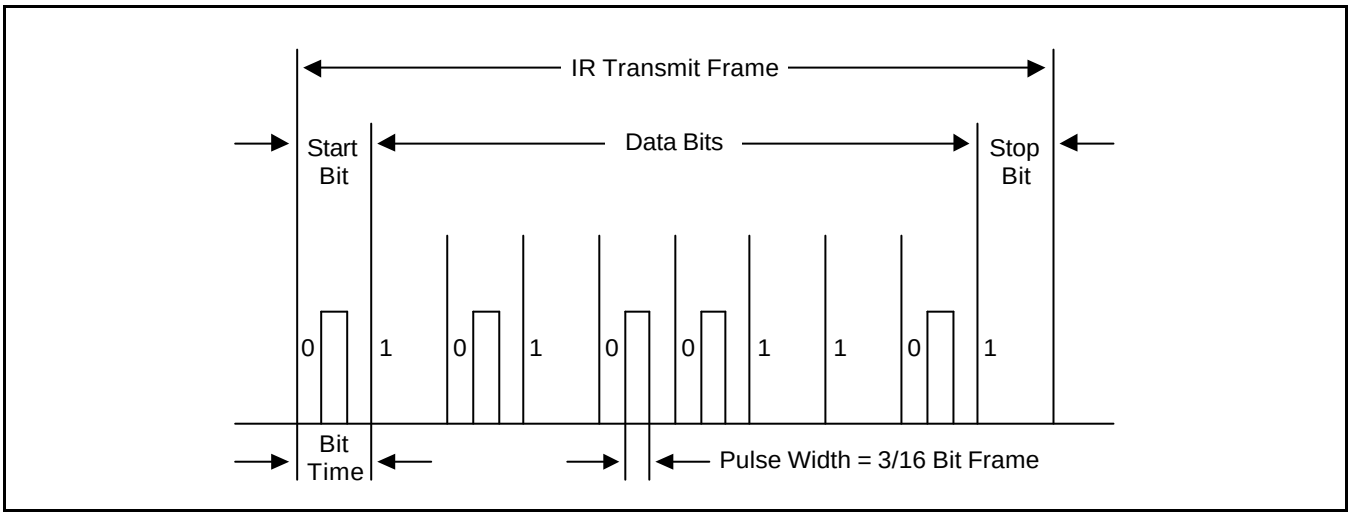


Figure 11-6. Infra-Red Transmit Mode Frame Timing Diagram

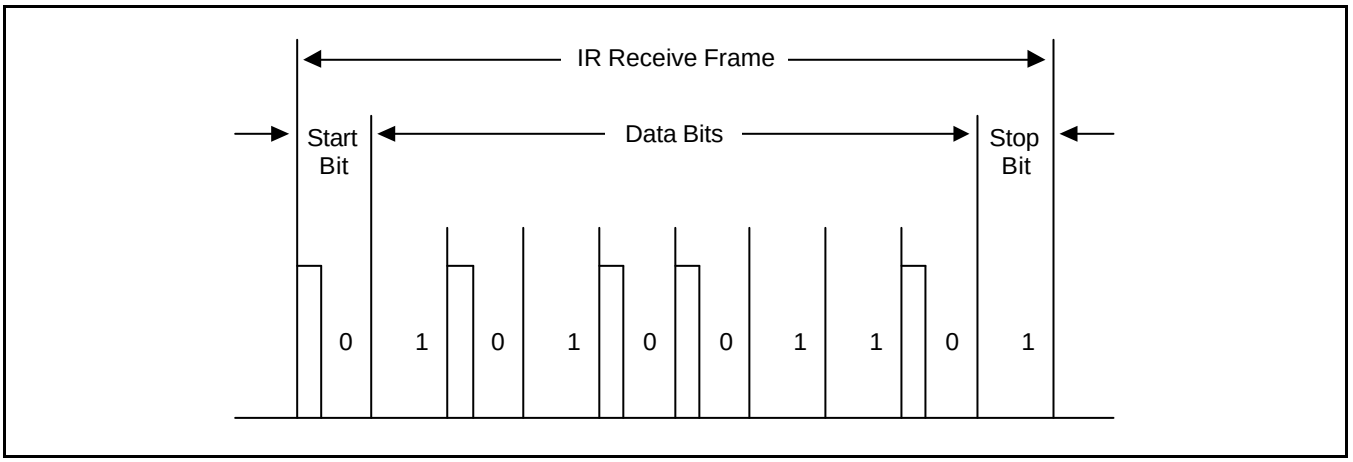


Figure 11-7. Infra-Red Receive Mode Frame Timing Diagram

UART SPECIAL REGISTERS

UART LINE CONTROL REGISTER

There are three UART line control registers including ULCON0, ULCON1, and ULCON2 in the UART block.

Register	Address	R/W	Description	Reset Value
ULCON0	0x50000000	R/W	UART channel 0 line control register	0x00
ULCON1	0x50004000	R/W	UART channel 1 line control register	0x00
ULCON2	0x50008000	R/W	UART channel 2 line control register	0x00

ULCONn	Bit	Description	Initial State
Reserved	[7]		0
Infra-Red Mode	[6]	Determine whether or not to use the Infra-Red mode. 0 = Normal mode operation 1 = Infra-Red Tx/Rx mode	0
Parity Mode	[5:3]	Specify the type of parity generation and checking during UART transmit and receive operation. 0xx = No parity 100 = Odd parity 101 = Even parity 110 = Parity forced/checked as 1 111 = Parity forced/checked as 0	000
Number of Stop Bit	[2]	Specify how many stop bits are to be used for end-of-frame signal. 0 = One stop bit per frame 1 = Two stop bit per frame	0
Word Length	[1:0]	Indicate the number of data bits to be transmitted or received per frame. 00 = 5-bit 01 = 6-bit 10 = 7-bit 11 = 8-bit	00

UART CONTROL REGISTER

There are three UART control registers including UCON0, UCON1 and UCON2 in the UART block.

Register	Address	R/W	Description	Reset Value
UCON0	0x50000004	R/W	UART channel 0 control register	0x00
UCON1	0x50004004	R/W	UART channel 1 control register	0x00
UCON2	0x50008004	R/W	UART channel 2 control register	0x00

UCONn	Bit	Description	Initial State
Clock Selection	[10]	Select PCLK or UEXTCLK for the UART baud rate. 0=PCLK : UBRDIVn = (int)(PCLK / (bps x 16)) -1 1=UEXTCLK(@GPH8) : UBRDIVn = (int)(UEXTCLK / (bps x 16)) -1	0
Tx Interrupt Type	[9]	Interrupt request type. 0 = Pulse (Interrupt is requested as soon as the Tx buffer becomes empty in Non-FIFO mode or reaches Tx FIFO Trigger Level in FIFO mode.) 1 = Level (Interrupt is requested while Tx buffer is empty in Non-FIFO mode or reaches Tx FIFO Trigger Level in FIFO mode.)	0
Rx Interrupt Type	[8]	Interrupt request type. 0 = Pulse (Interrupt is requested the instant Rx buffer receives the data in Non-FIFO mode or reaches Rx FIFO Trigger Level in FIFO mode.) 1 = Level (Interrupt is requested while Rx buffer is receiving data in Non-FIFO mode or reaches Rx FIFO Trigger Level in FIFO mode.)	0
Rx Time Out Enable	[7]	Enable/Disable Rx time out interrupt when UART FIFO is enabled. The interrupt is a receive interrupt. 0 = Disable 1 = Enable	0
Rx Error Status Interrupt Enable	[6]	Enable the UART to generate an interrupt upon an exception, such as a frame error, or overrun error during a receive operation. 0 = Do not generate receive error status interrupt. 1 = Generate receive error status interrupt.	0
Loopback Mode	[5]	Setting loopback bit to 1 causes the UART to enter the loopback mode. This mode is provided for test purposes only. 0 = Normal operation 1 = Loopback mode	0
Reserved	[4]	Reserved	0

UART CONTROL REGISTER (Continued)

UCONn	Bit	Description	Initial State
Transmit Mode	[3:2]	Determine which function is currently able to write Tx data to the UART transmit buffer register. (UART Tx Enable/Disable) 00 = Disable 01 = Interrupt request or polling mode 10 = DMA0 request (Only for UART0), DMA3 request (Only for UART2) 11 = DMA1 request (Only for UART1)	00
Receive Mode	[1:0]	Determine which function is currently able to read data from UART receive buffer register. (UART Rx Enable/Disable) 00 = Disable 01 = Interrupt request or polling mode 10 = DMA0 request (Only for UART0), DMA3 request (Only for UART2) 11 = DMA1 request (Only for UART1)	00

NOTE: When the UART does not reach the FIFO trigger level and does not receive data during 3 word time in DMA receive mode with FIFO, the Rx interrupt will be generated (receive time out), and the users should check the FIFO status and read out the rest.

UART FIFO CONTROL REGISTER

There are three UART FIFO control registers including UFCON0, UFCON1 and UFCON2 in the UART block.

Register	Address	R/W	Description	Reset Value
UFCON0	0x50000008	R/W	UART channel 0 FIFO control register	0x0
UFCON1	0x50004008	R/W	UART channel 1 FIFO control register	0x0
UFCON2	0x50008008	R/W	UART channel 2 FIFO control register	0x0

UFCONn	Bit	Description	Initial State
Tx FIFO Trigger Level	[7:6]	Determine the trigger level of transmit FIFO. 00 = Empty 01 = 4-byte 10 = 8-byte 11 = 12-byte	00
Rx FIFO Trigger Level	[5:4]	Determine the trigger level of receive FIFO. 00 = 4-byte 01 = 8-byte 10 = 12-byte 11 = 16-byte	00
Reserved	[3]		0
Tx FIFO Reset	[2]	Auto-cleared after resetting FIFO 0 = Normal 1= Tx FIFO reset	0
Rx FIFO Reset	[1]	Auto-cleared after resetting FIFO 0 = Normal 1= Rx FIFO reset	0
FIFO Enable	[0]	0 = Disable 1 = Enable	0

NOTE: When the UART does not reach the FIFO trigger level and does not receive data during 3 word time in DMA receive mode with FIFO, the Rx interrupt will be generated (receive time out), and the users should check the FIFO status and read out the rest.

UART MODEM CONTROL REGISTER

There are two UART MODEM control registers including UMCON0 and UMCON1 in the UART block.

Register	Address	R/W	Description	Reset Value
UMCON0	0x5000000C	R/W	UART channel 0 Modem control register	0x0
UMCON1	0x5000400C	R/W	UART channel 1 Modem control register	0x0
Reserved	0x5000800C	-	Reserved	Undef

UMCONn	Bit	Description	Initial State
Reserved	[7:5]	These bits must be 0's	00
Auto Flow Control (AFC)	[4]	0 = Disable 1 = Enable	0
Reserved	[3:1]	These bits must be 0's	00
Request to Send	[0]	If AFC bit is enabled, this value will be ignored. In this case the S3C2410A will control nRTS automatically. If AFC bit is disabled, nRTS must be controlled by software. 0 = 'H' level (Inactivate nRTS) 1 = 'L' level (Activate nRTS)	0

NOTE: UART 2 does not support AFC function, because the S3C2410A has no nRTS2 and nCTS2.

UART TX/RX STATUS REGISTER

There are three UART Tx/Rx status registers including UTRSTAT0, UTRSTAT1 and UTRSTAT2 in the UART block.

Register	Address	R/W	Description	Reset Value
UTRSTAT0	0x50000010	R	UART channel 0 Tx/Rx status register	0x6
UTRSTAT1	0x50004010	R	UART channel 1 Tx/Rx status register	0x6
UTRSTAT2	0x50008010	R	UART channel 2 Tx/Rx status register	0x6

UTRSTATn	Bit	Description	Initial State
Transmitter empty	[2]	Set to 1 automatically when the transmit buffer register has no valid data to transmit and the transmit shift register is empty. 0 = Not empty 1 = Transmitter (transmit buffer & shifter register) empty	1
Transmit buffer empty	[1]	Set to 1 automatically when transmit buffer register is empty. 0 =The buffer register is not empty 1 = Empty (In Non-FIFO mode, Interrupt or DMA is requested. In FIFO mode, Interrupt or DMA is requested, when Tx FIFO Trigger Level is set to 00 (Empty)) If the UART uses the FIFO, users should check Tx FIFO Count bits and Tx FIFO Full bit in the UFSTAT register instead of this bit.	1
Receive buffer data ready	[0]	Set to 1 automatically whenever receive buffer register contains valid data, received over the RXDn port. 0 = Empty 1 = The buffer register has a received data (In Non-FIFO mode, Interrupt or DMA is requested) If the UART uses the FIFO, users should check Rx FIFO Count bits and Rx FIFO Full bit in the UFSTAT register instead of this bit.	0

UART ERROR STATUS REGISTER

There are three UART Rx error status registers including UERSTAT0, UERSTAT1 and UERSTAT2 in the UART block.

Register	Address	R/W	Description	Reset Value
UERSTAT0	0x50000014	R	UART channel 0 Rx error status register	0x0
UERSTAT1	0x50004014	R	UART channel 1 Rx error status register	0x0
UERSTAT2	0x50008014	R	UART channel 2 Rx error status register	0x0

UERSTATn	Bit	Description	Initial State
Reserved	[3]	0 = No frame error during receive 1 = Frame error (Interrupt is requested.)	0
Frame Error	[2]	Set to 1 automatically whenever a frame error occurs during receive operation. 0 = No frame error during receive 1 = Frame error (Interrupt is requested.)	0
Reserved	[1]	0 = No frame error during receive 1 = Frame error (Interrupt is requested.)	0
Overrun Error	[0]	Set to 1 automatically whenever an overrun error occurs during receive operation. 0 = No overrun error during receive 1 = Overrun error (Interrupt is requested.)	0

NOTE: These bits (UERSATn[3:0]) are automatically cleared to 0 when the UART error status register is read.

UART FIFO STATUS REGISTER

There are three UART FIFO status registers including UFSTAT0, UFSTAT1 and UFSTAT2 in the UART block.

Register	Address	R/W	Description	Reset Value
UFSTAT0	0x50000018	R	UART channel 0 FIFO status register	0x00
UFSTAT1	0x50004018	R	UART channel 1 FIFO status register	0x00
UFSTAT2	0x50008018	R	UART channel 2 FIFO status register	0x00

UFSTATn	Bit	Description	Initial State
Reserved	[15:10]		0
Tx FIFO Full	[9]	Set to 1 automatically whenever transmit FIFO is full during transmit operation 0 = 0-byte ≤ Tx FIFO data ≤ 15-byte 1 = Full	0
Rx FIFO Full	[8]	Set to 1 automatically whenever receive FIFO is full during receive operation 0 = 0-byte ≤ Rx FIFO data ≤ 15-byte 1 = Full	0
Tx FIFO Count	[7:4]	Number of data in Tx FIFO	0
Rx FIFO Count	[3:0]	Number of data in Rx FIFO	0

UART MODEM STATUS REGISTER

There are two UART modem status registers including UMSTAT0 and UMSTAT1 in the UART block.

Register	Address	R/W	Description	Reset Value
UMSTAT0	0x5000001C	R	UART channel 0 Modem status register	0x0
UMSTAT1	0x5000401C	R	UART channel 1 Modem status register	0x0
Reserved	0x5000801C	–	Reserved	Undef

UMSTAT0	Bit	Description	Initial State
Delta CTS	[4]	Indicate that the nCTS input to the S3C2410A has changed state since the last time it was read by CPU. (Refer to Figure 11-8.) 0 = Has not changed 1 = Has changed	0
Reserved	[3:1]		0
Clear to Send	[0]	0 = CTS signal is not activated (nCTS pin is high.) 1 = CTS signal is activated (nCTS pin is low.)	0

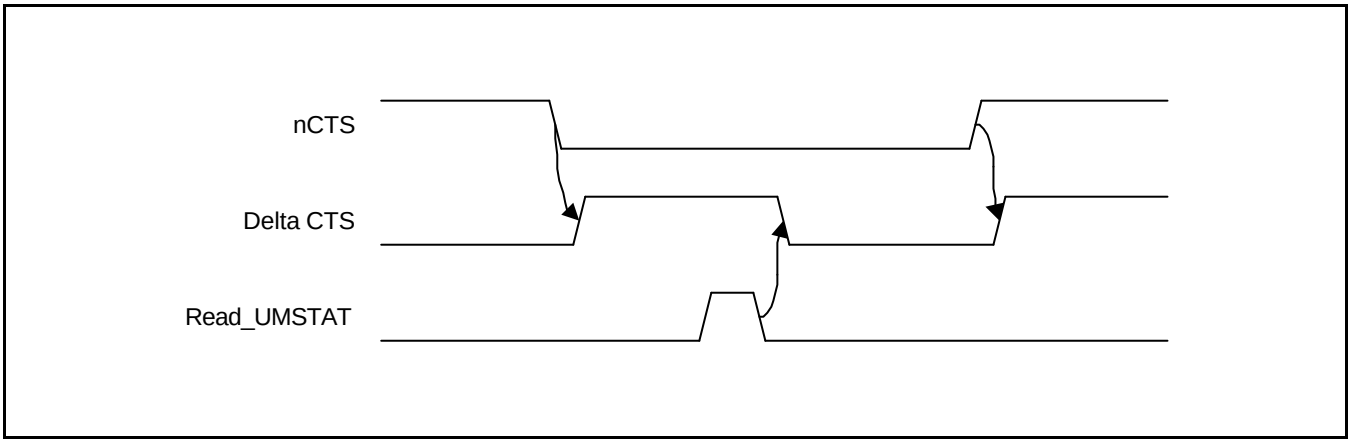


Figure 11-8. nCTS and Delta CTS Timing Diagram

UART TRANSMIT BUFFER REGISTER (HOLDING REGISTER & FIFO REGISTER)

There are three UART transmit buffer registers including UTXH0, UTXH1 and UTXH2 in the UART block. UTXHn has an 8-bit data for transmission data.

Register	Address	R/W	Description	Reset Value
UTXH0	0x50000020(L) 0x50000023(B)	W (by byte)	UART channel 0 transmit buffer register	–
UTXH1	0x50004020(L) 0x50004023(B)	W (by byte)	UART channel 1 transmit buffer register	–
UTXH2	0x50008020(L) 0x50008023(B)	W (by byte)	UART channel 2 transmit buffer register	–

UTXHn	Bit	Description	Initial State
TXDATAn	[7:0]	Transmit data for UARTn	–

NOTE: (L): The endian mode is Little endian.
(B): The endian mode is Big endian.

UART RECEIVE BUFFER REGISTER (HOLDING REGISTER & FIFO REGISTER)

There are three UART receive buffer registers including URXH0, URXH1 and URXH2 in the UART block. URXHn has an 8-bit data for received data.

Register	Address	R/W	Description	Reset Value
URXH0	0x50000024(L) 0x50000027(B)	R (by byte)	UART channel 0 receive buffer register	–
URXH1	0x50004024(L) 0x50004027(B)	R (by byte)	UART channel 1 receive buffer register	–
URXH2	0x50008024(L) 0x50008027(B)	R (by byte)	UART channel 2 receive buffer register	–

URXHn	Bit	Description	Initial State
RXDATAn	[7:0]	Receive data for UARTn	–

NOTE: When an overrun error occurs, the URXHn must be read. If not, the next received data will also make an overrun error, even though the overrun bit of UERSTATn had been cleared.

UART BAUD RATE DIVISOR REGISTER

There are three UART baud rate divisor registers including UBRDIV0, UBRDIV1 and UBRDIV2 in the UART block. The value stored in the baud rate divisor register (UBRDIVn), is used to determine the serial Tx/Rx clock rate (baud rate) as follows:

UBRDIVn = (int)(PCLK / (bps x 16)) –1

or

UBRDIVn = (int)(UEXTCLK / (bps x 16)) –1

Where, the divisor should be from 1 to (2^{16}-1) and UEXTCLK should be smaller than PCLK.

For example, if the baud-rate is 115200 bps and PCLK or UEXTCLK is 40 MHz, UBRDIVn is:

UBRDIVn = (int)(40000000 / (115200 x 16)) -1
= (int)(21.7) -1
= 21 -1 = 20

Register	Address	R/W	Description	Reset Value
UBRDIV0	0x50000028	R/W	Baud rate divisior register 0	–
UBRDIV1	0x50004028	R/W	Baud rate divisior register 1	–
UBRDIV2	0x50008028	R/W	Baud rate divisior register 2	–

UBRDIVn	Bit	Description	Initial State
UBRDIV	[15:0]	Baud rate division value UBRDIVn > 0	–

12 USB HOST CONTROLLER

OVERVIEW

S3C2410A supports 2-port USB host interface as follows:

- OHCI Rev 1.0 compatible
- USB Rev1.1 compatible
- Two down stream ports
- Support for both LowSpeed and FullSpeed USB devices

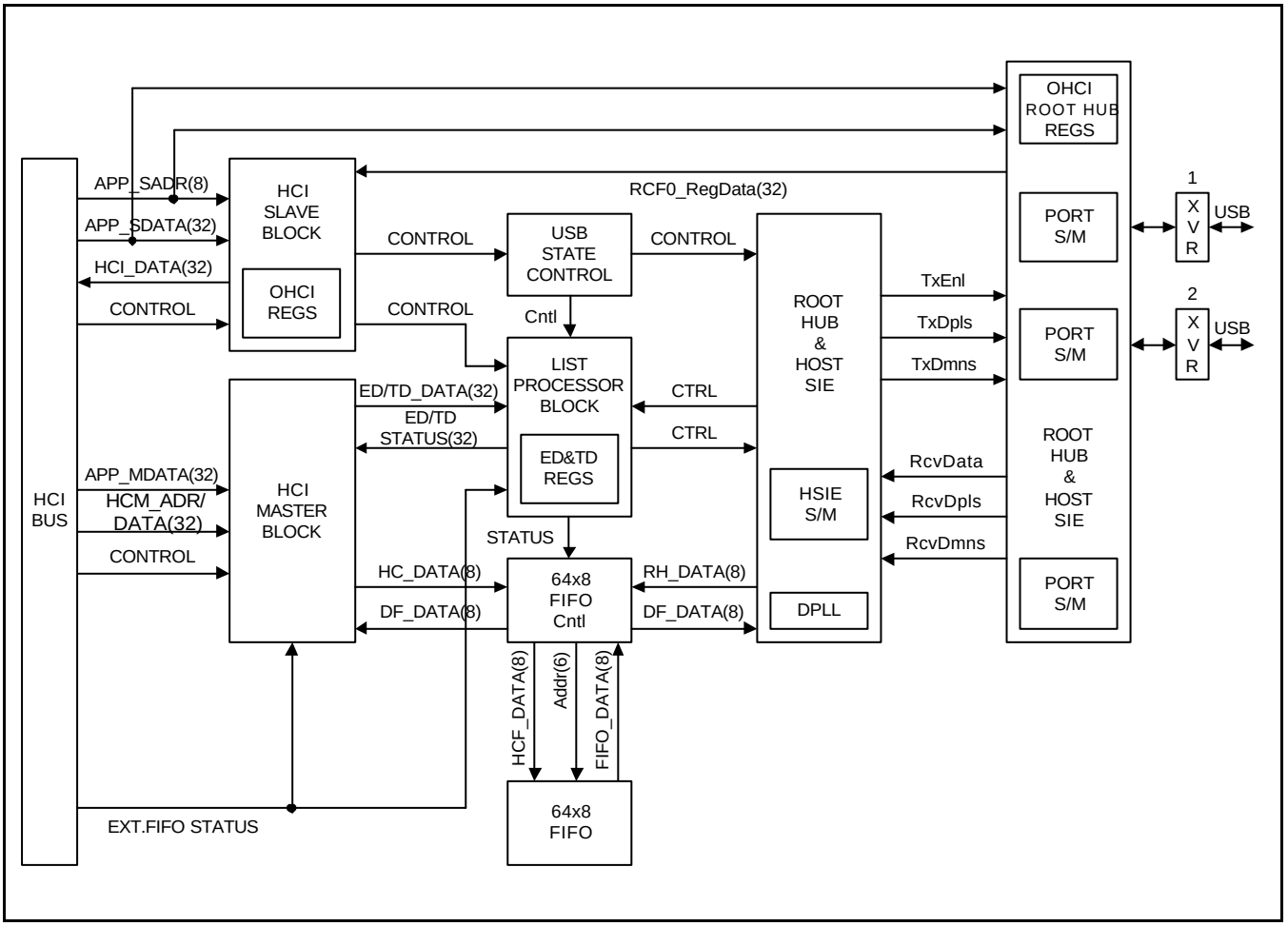


Figure 12-1. USB Host Controller Block Diagram

USB HOST CONTROLLER SPECIAL REGISTERS

The S3C2410A USB host controller complies with OHCI Rev 1.0. Refer to Open Host Controller Interface Rev 1.0 specification for detail information.

Table 12-1. OHCI Registers for USB Host Controller

Register	Base Address	R/W	Description	Reset Value
HcRevision	0x49000000	–	Control and status group	–
HcControl	0x49000004	–		–
HcCommonStatus	0x49000008	–		–
HcInterruptStatus	0x4900000C	–		–
HcInterruptEnable	0x49000010	–		–
HcInterruptDisable	0x49000014	–		–
HcHCCA	0x49000018	–	Memory pointer group	–
HcPeriodCuttentED	0x4900001C	–		–
HcControlHeadED	0x49000020	–		–
HcControlCurrentED	0x49000024	–		–
HcBulkHeadED	0x49000028	–		–
HcBulkCurrentED	0x4900002C	–		–
HcDoneHead	0x49000030	–		–
HcRmInterval	0x49000034	–	Frame counter group	–
HcFmRemaining	0x49000038	–		–
HcFmNumber	0x4900003C	–		–
HcPeriodicStart	0x49000040	–		–
HcLSThreshold	0x49000044	–		–
HcRhDescriptorA	0x49000048	–	Root hub group	–
HcRhDescriptorB	0x4900004C	–		–
HcRhStatus	0x49000050	–		–
HcRhPortStatus1	0x49000054	–		–
HcRhPortStatus2	0x49000058	–		–

13 USB DEVICE CONTROLLER

OVERVIEW

Universal Serial Bus (USB) device controller is designed to provide a high performance full speed function controller solution with DMA interface. USB device controller allows bulk transfer with DMA, interrupt transfer and control transfer.

USB device controller supports:

— Full speed USB device controller compatible with the USB specification version 1.1

— DMA interface for bulk transfer

— Five endpoints with FIFO
EP0: 16byte (Register)
EP1: 64byte IN/OUT FIFO (dual port asynchronous RAM): interrupt or DMA
EP2: 64byte IN/OUT FIFO (dual port asynchronous RAM): interrupt or DMA
EP3: 64byte IN/OUT FIFO (dual port asynchronous RAM): interrupt or DMA
EP4: 64byte IN/OUT FIFO (dual port asynchronous RAM): interrupt or DMA

— Integrated USB Transceiver

FEATURE

— Fully compliant with USB Specification Version 1.1

— Full speed (12Mbps) device

— Integrated USB Transceiver

— Supports control, interrupt and bulk transfer

— Five endpoints with FIFO:
One bi-directional control endpoint with 16-byte FIFO (EP0)
Four bi-directional bulk endpoints with 64-byte FIFO (EP1, EP2, EP3, and EP4)

— Supports DMA interface for receive and transmit bulk endpoints. (EP1, EP2, EP3, and EP4)

— Independent 64byte receive and transmit FIFO to maximize throughput

— Supports suspend and remote wakeup function

NOTE

PCLK should be more than 20MHz to use USB Device Controller stably.

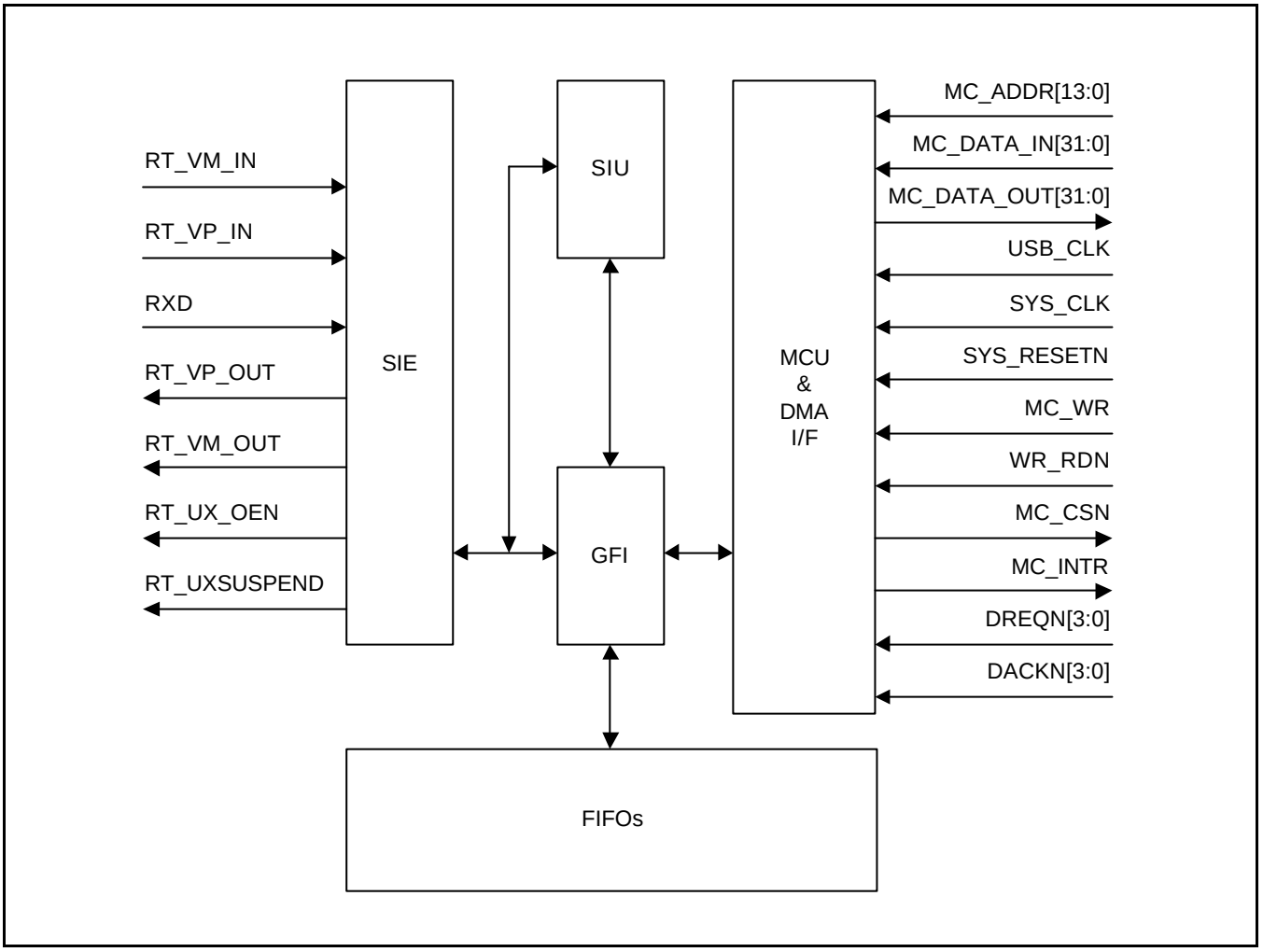


Figure 13-1. USB Device Controller Block Diagram

USB DEVICE CONTROLLER SPECIAL REGISTERS

This section describes detailed functionalities about register sets of USB device controller.
All special function register is byte-accessible or word-accessible. If you access byte mode offset-address is different in little endian and big endian. All reserved bit is zero.
Common indexed registers depend on INDEX register (INDEX_REG) (offset address: 0x178) value. For example if you want to write EP0 CSR register, you must write "0x00" on the INDEX_REG before writing IN_CSR1 register.

NOTE

All register must be resettled after performing Host Reset Signaling.

Register Name	Description	Offset Address
NON INDEXED REGISTERS		
FUNC_ADDR_REG	Function address register	0x140(L) / 0x143(B)
PWR_REG	Power management register	0x144(L) / 0x147(B)
EP_INT_REG (EP0–EP4)	Endpoint interrupt register	0x148(L) / 0x14B(B)
USB_INT_REG	USB interrupt register	0x158(L) / 0x15B(B)
EP_INT_EN_REG (EP0–EP4)	Endpoint interrupt enable register	0x15C(L) / 0x15F(B)
USB_INT_EN_REG	USB Interrupt enable register	0x16C(L) / 0x16F(B)
FRAME_NUM1_REG	Frame number 1 register	0x170(L) / 0x173(B)
FRAME_NUM2_REG	Frame number 2 register	0x174(L) / 0x177(B)
INDEX_REG	Index register	0x178(L) / 0x17B(B)
EP0_FIFO_REG	Endpoint0 FIFO register	0x1C0(L) / 0x1C3(B)
EP1_FIFO_REG	Endpoint1 FIFO register	0x1C4(L) / 0x1C7(B)
EP2_FIFO_REG	Endpoint2 FIFO register	0x1C8(L) / 0x1CB(B)
EP3_FIFO_REG	Endpoint3 FIFO register	0x1CC(L) / 0x1CF(B)
EP4_FIFO_REG	Endpoint4 FIFO register	0x1D0(L) / 0x1D3(B)
EP1_DMA_CON	Endpoint1 DMA control register	0x200(L) / 0x203(B)
EP1_DMA_UNIT	Endpoint1 DMA unit counter register	0x204(L) / 0x207(B)
EP1_DMA_FIFO	Endpoint1 DMA FIFO counter register	0x208(L) / 0x20B(B)
EP1_DMA_TTC_L	Endpoint1 DMA transfer counter low-byte register	0x20C(L) / 0x20F(B)
EP1_DMA_TTC_M	Endpoint1 DMA transfer counter middle-byte register	0x210(L) / 0x213(B)
EP1_DMA_TTC_H	Endpoint1 DMA transfer counter high-byte register	0x214(L) / 0x217(B)
EP2_DMA_CON	Endpoint2 DMA control register	0x218(L) / 0x21B(B)
EP2_DMA_UNIT	Endpoint2 DMA unit counter register	0x21C(L) / 0x21F(B)
EP2_DMA_FIFO	Endpoint2 DMA FIFO counter register	0x220(L) / 0x223(B)
EP2_DMA_TTC_L	Endpoint2 DMA transfer counter low-byte register	0x224(L) / 0x227(B)

USB Device Controller Special Registers (Continued)

Register Name	Description	Offset Address
EP2_DMA_TTC_M	Endpoint2 DMA transfer counter middle-byte register	0x228(L) / 0x22B(B)
EP2_DMA_TTC_H	Endpoint2 DMA transfer counter high-byte register	0x22C(L) / 0x22F(B)
EP3_DMA_CON	Endpoint3 DMA control register	0x240(L) / 0x243(B)
EP3_DMA_UNIT	Endpoint3 DMA unit counter register	0x244(L) / 0x247(B)
EP3_DMA_FIFO	Endpoint3 DMA FIFO counter register	0x248(L) / 0x24B(B)
EP3_DMA_TTC_L	Endpoint3 DMA transfer counter low-byte register	0x24C(L) / 0x24F(B)
EP3_DMA_TTC_M	Endpoint3 DMA transfer counter middle-byte register	0x250(L) / 0x253(B)
EP3_DMA_TTC_H	Endpoint3 DMA transfer counter high-byte register	0x254(L) / 0x247(B)
EP4_DMA_CON	Endpoint4 DMA control register	0x258(L) / 0x25B(B)
EP4_DMA_UNIT	Endpoint4 DMA unit counter register	0x25C(L) / 0x25F(B)
EP4_DMA_FIFO	Endpoint4 DMA FIFO counter register	0x260(L) / 0x263(B)
EP4_DMA_TTC_L	Endpoint4 DMA transfer counter low-byte register	0x264(L) / 0x267(B)
EP4_DMA_TTC_M	Endpoint4 DMA transfer counter middle-byte register	0x268(L) / 0x26B(B)
EP4_DMA_TTC_H	Endpoint4 DMA transfer counter high-byte register	0x26C(L) / 0x26F(B)
COMMON INDEXED REGISTERS		
MAXP_REG	Endpoint MAX packet register	0x180(L) / 0x183(B)
IN INDEXED REGISTERS		
IN_CSR1_REG/EP0_CSR	EP In control status register 1/EP0 control status register	0x184(L) / 0x187(B)
IN_CSR2_REG	EP In control status register 2	0x188(L) / 0x18B(B)
OUT INDEXED REGISTERS		
OUT_CSR1_REG	EP out control status register 1	0x190(L) / 0x193(B)
OUT_CSR2_REG	EP out control status register 2	0x194(L) / 0x197(B)
OUT_FIFO_CNT1_REG	EP out write count register 1	0x198(L) / 0x19B(B)
OUT_FIFO_CNT2_REG	EP out write count register 2	0x19C(L) / 0x19F(B)

FUNCTION ADDRESS REGISTER (FUNC_ADDR_REG)

This register maintains the USB device controller address assigned by the host. The Micro Controller Unit (MCU) writes the value received through a SET_ADDRESS descriptor to this register. This address is used for the next token.

Register	Address	R/W	Description	Reset Value
FUNC_ADDR_REG	0x52000140(L) 0x52000143(B)	R/W (byte)	Function address register	0x00

FUNC_ADDR_REG	Bit	MCU	USB	Description	Initial State
ADDR_UPDATE	[7]	R /SET	R /CLEAR	Set by the MCU whenever it updates the FUNCTION_ADDR field in this register. This bit will be cleared by USB when DATA_END bit in EP0_CSR register.	0
FUNCTION_ADDR	[6:0]	R/W	R	The MCU write the unique address, assigned by host, to this field.	00

POWER MANAGEMENT REGISTER (PWR_REG)

This register acts as a power control register in the USB block.

Register	Address	R/W	Description	Reset Value
PWR_REG	0x52000144(L) 0x52000147(B)	R/W (byte)	Power management register	0x00

PWR_ADDR	Bit	MCU	USB	Description	Initial State
Reserved	[7:4]	–	–	–	–
USB_RESET	[3]	R	SET	Set by the USB if reset signaling is received from the host. This bit remains set as long as reset signaling persists on the bus	0
MCU_RESUME	[2]	R/W	R /CLEAR	Set by the MCU for MCU Resume. The USB generates the resume signaling during 10ms, if this bit is set in suspend mode.	
SUSPEND_MODE	[1]	R	SET /CLEAR	Set by USB automatically when the device enter into suspend mode. It is cleared under the following conditions: 1) The MCU clears the MCU_RESUME bit by writing "0", in order to end remote resume signaling. 2) The resume signal form host is received.	0
SUSPEND_EN	[0]	R/W	R	Suspend mode enable control bit 0 = Disable (default). The device will not enter suspend mode. 1 = Enable suspend mode.	0

INTERRUPT REGISTER (EP_INT_REG/USB_INT_REG)

The USB core has two interrupt registers.
These registers act as status registers for the MCU when it is interrupted. The bits are cleared by writing a "1" (not "0") to each bit that was set.
Once the MCU is interrupted, MCU should read the contents of interrupt-related registers and write back to clear the contents if it is necessary.

Register	Address	R/W	Description	Reset Value
EP_INT_REG	0x52000148(L) 0x5200014B(B)	R/W (byte)	EP interrupt pending/clear register	0x00

EP_INT_REG	Bit	MCU	USB	Description	Initial State
EP1~EP4 Interrupt	[4:1]	R /CLEAR	SET	**For BULK/INTERRUPT IN endpoints:** Set by the USB under the following conditions: 1. IN_PKT_RDY bit is cleared. 2. FIFO is flushed 3. SENT_STALL set. **For BULK/INTERRUPT OUT endpoints:** Set by the USB under the following conditions: 1. Sets OUT_PKT_RDY bit 2. Sets SENT_STALL bit **NOTE:** Conditions 1 and 2 are mutually exclusive.	0
EP0 Interrupt	[0]	R /CLEAR	SET	Correspond to endpoint 0 interrupt. Set by the USB under the following conditions: 1. OUT_PKT_RDY bit is set. 2. IN_PKT_RDY bit is cleared. 3. SENT_STALL bit is set 4. SETUP_END bit is set 5. DATA_END bit is cleared (it indicates the end of control transfer).	0

INTERRUPT REGISTER (EP_INT_REG/USB_INT_REG) (Continued)

Register	Address	R/W	Description	Reset Value
USB_INT_REG	0x52000158(L) 0x5200015B(B)	R/W (byte)	USB interrupt pending/clear register	0x00

USB_INT_REG	Bit	MCU	USB	Description	Initial State
RESET Interrupt	[2]	R /CLEAR	SET	Set by the USB when it receives reset signaling.	0
RESUME Interrupt	[1]	R /CLEAR	SET	Set by the USB when it receives resume signaling, *while in Suspend mode*. If the resume occurs due to a USB reset, then the MCU is first interrupted with a RESUME interrupt. Once the clocks resume and the SE0 condition persists for 3ms, USB RESET interrupt will be asserted.	0
SUSPEND Interrupt	[0]	R /CLEAR	SET	Set by the USB when it receives suspend signalizing. This bit is set whenever there is no activity for 3ms on the bus. Thus, if the MCU does not stop the clock after the first suspend interrupt, it will continue to be interrupted every 3ms as long as there is no activity on the USB bus. By default, this interrupt is disabled.	0

NOTE: If the RESET interrupt is occurred, all USB device registers should be re-configured.

INTERRUPT ENABLE REGISTER (EP_INT_EN_REG/USB_INT_EN_REG)

Corresponding to each interrupt register, The USB device controller also has two interrupt enable registers (except resume interrupt enable). By default, usb reset interrupt is enabled.

If bit = 0, the interrupt is disabled.
If bit = 1, the interrupt is enabled.

Register	Address	R/W	Description	Reset Value
EP_INT_EN_REG	0x5200015C(L) 0x5200015F(B)	R/W (byte)	Determine which interrupt is enabled	0xFF

EP_INT_EN_REG	Bit	MCU	USB	Description	Initial State
EP4_INT_EN	[4]	R/W	R	EP4 Interrupt Enable bit 0 = Interrupt disable 1 = Enable	1
EP3_INT_EN	[3]	R/W	R	EP3 Interrupt Enable bit 0 = Interrupt disable 1 = Enable	1
EP2_INT_EN	[2]	R/W	R	EP2 Interrupt Enable bit 0 = Interrupt disable 1 = Enable	1
EP1_INT_EN	[1]	R/W	R	EP1 Interrupt Enable bit 0 = Interrupt disable 1 = Enable	1
EP0_INT_EN	[0]	R/W	R	EP0 Interrupt Enable bit 0 = Interrupt disable 1 = Enable	1

Register	Address	R/W	Description	Reset Value
USB_INT_EN_REG	0x5200016C(L) 0x5200016F(B)	R/W (byte)	Determine which interrupt is enabled	0x04

INT_MASK_REG	Bit	MCU	USB	Description	Initial State
RESET_INT_EN	[2]	R/W	R	Reset interrupt enable bit 0 = Interrupt disable 1 = Enable	1
Reserved	[1]	–	–	–	0
SUSPEND_INT_EN	[0]	R/W	R	Suspend interrupt enable bit 0 = Interrupt disable 1 = Enable	0

FRAME NUMBER REGISTER (FPAME_NUM1_REG/FRAME_NUM2_REG)

When the host transfers USB packets, each Start Of Frame (SOF) packet includes a frame number. The USB device controller catches this frame number and loads it into this register automatically.

Register	Address	R/W	Description	Reset Value
FRAME_NUM1_REG	0x52000170(L) 0x52000173(B)	R (byte)	Frame number lower byte register	0x00

FRAME_NUM_REG	Bit	MCU	USB	Description	Initial State
FRAME_NUM1	[7:0]	R	W	Frame number lower byte value	00

Register	Address	R/W	Description	Reset Value
FRAME_NUM2_REG	0x52000174(L) 0x52000177(B)	R (byte)	Frame number higher byte register	0x00

FRAME_NUM_REG	Bit	MCU	USB	Description	Initial State
FRAME_NUM2	[7:0]	R	W	Frame number higher byte value	00

INDEX REGISTER (INDEX_REG)

The INDEX register is used to indicate certain endpoint registers effectively. The MCU can access the endpoint registers (MAXP_REG, IN_CSR1_REG, IN_CSR2_REG, OUT_CSR1_REG, OUT_CSR2_REG, OUT_FIFO_CNT1_REG, and OUT_FIFO_CNT2_REG) for an endpoint inside the core using the INDEX register.

Register	Address	R/W	Description	Reset Value
INDEX_REG	0x52000178(L) 0x5200017B(B)	R/W (byte)	Register index register	0x00

INDEX_REG	Bit	MCU	USB	Description	Initial State
INDEX	[7:0]	R/W	R	Indicate a certain endpoint	00

END POINT0 CONTROL STATUS REGISTER (EP0_CSR)

This register has the control and status bits for Endpoint 0. Since a control transaction is involved with both IN and OUT tokens, there is only one CSR register, mapped to the IN CSR1 register. (share IN1_CSR and can access by writing index register "0" and read/write IN1_CSR)

Register	Address	R/W	Description	Reset Value
EP0_CSR	0x52000184(L) 0x52000187(B)	R/W (byte)	Endpoint 0 status register	0x00

EP0_CSR	Bit	MCU	USB	Description	Initial State
SERVICED_SETUP_END	[7]	W	CLEAR	The MCU should write a "1" to this bit to clear SETUP_END.	0
SERVICED_OUT_PKT_RDY	[6]	W	CLEAR	The MCU should write a "1" to this bit to clear OUT_PKT_RDY.	0
SEND_STALL	[5]	R/W	CLEAR	MCU should write a "1" to this bit at the same time it clears OUT_PKT_RDY, if it decodes an invalid token. 0 = Finish the STALL condition 1 = The USB issues a STALL and shake to the current control transfer.	0
SETUP_END	[4]	R	SET	Set by the USB when a control transfer ends before DATA_END is set. When the USB sets this bit, an interrupt is generated to the MCU. When such a condition occurs, the USB flushes the FIFO and invalidates MCU access to the FIFO.	0
DATA_END	[3]	SET	CLEAR	Set by the MCU on the conditions below: 1. After loading the last packet of data into the FIFO, at the same time IN_PKT_RDY is set. 2. While it clears OUT_PKT_RDY after unloading the last packet of data. 3. For a zero length data phase.	0
SENT_STALL	[2]	CLEAR	SET	Set by the USB if a control transaction is stopped due to a protocol violation. An interrupt is generated when this bit is set. The MCU should write "0" to clear this bit.	0

END POINT0 CONTROL STATUS REGISTER (EP0_CSR) (Continued)

EP0_CSR	Bit	MCU	USB	Description	Initial State
IN_PKT_RDY	[1]	SET	CLEAR	Set by the MCU after writing a packet of data into EP0 FIFO. The USB clears this bit once the packet has been successfully sent to the host. An interrupt is generated when the USB clears this bit, so as the MCU to load the next packet. For a zero length data phase, the MCU sets DATA_END at the same time.	0
OUT_PKT_RDY	[0]	R	SET	Set by the USB once a valid token is written to the FIFO. An interrupt is generated when the USB sets this bit. The MCU clears this bit by writing a "1" to the SERVICED_OUT_PKT_RDY bit.	0

END POINT IN CONTROL STATUS REGISTER (IN_CSR1_REG/IN_CSR2_REG)

Register	Address	R/W	Description	Reset Value
IN_CSR1_REG	0x52000184(L) 0x52000187(B)	R/W (byte)	IN END POINT control status register1	0x00

IN_CSR1_REG	Bit	MCU	USB	Description	Initial State
Reserved	[7]	–	–	–	0
CLR_DATA_ TOGGLE	[6]	R/W	R/ CLEAR	Used in Set-up procedure. 0: There are alternation of DATA0 and DATA1 1: The data toggle bit is cleared and PID in packet will maintain DATA0	0
SENT_STALL	[5]	R/ CLEAR	SET	Set by the USB when an IN token issues a STALL handshake, after the MCU sets SEND_STALL bit to start STALL handshaking. When the USB issues a STALL handshake, IN_PKT_RDY is cleared	0
SEND_STALL	[4]	W/R	R	0: The MCU clears this bit to finish the STALL condition. 1: The MCU issues a STALL handshake to the USB.	0
FIFO_FLUSH	[3]	R/W	CLEAR	Set by the MCU if it intends to flush the packet in Input-related FIFO. This bit is cleared by the USB when the FIFO is flushed. The MCU is interrupted when this happens. If a token is in process, the USB waits until the transmission is complete before FIFO flushing. If two packets are loaded into the FIFO, only first packet (The packet is intended to be sent to the host) is flushed, and the corresponding IN_PKT_RDY bit is cleared	0
Reserved	[2:1]	–	–	–	0

END POINT IN CONTROL STATUS REGISTER (IN_CSR1_REG/IN_CSR2_REG) (Continued)

IN_CSR1_REG	Bit	MCU	USB	Description	Initial State
IN_PKT_RDY	[0]	R/SET	CLEAR	Set by the MCU after writing a packet of data into the FIFO. The USB clears this bit once the packet has been successfully sent to the host. An interrupt is generated when the USB clears this bit, so the MCU can load the next packet. While this bit is set, the MCU will not be able to write to the FIFO. If the MCU sets SEND STALL bit, this bit cannot be set.	0

Register	Address	R/W	Description	Reset Value
IN_CSR2_REG	0x52000188(L) 0x5200018B(B)	R/W (byte)	IN END POINT control status register2	0x20

IN_CSR2_REG	Bit	MCU	USB	Description	Initial State
AUTO_SET	[7]	R/W	R	If set, whenever the MCU writes MAXP data, IN_PKT_RDY will automatically be set by the core without any intervention from MCU. If the MCU writes less than MAXP data, IN_PKT_RDY bit has to be set by the MCU.	0
ISO	[6]	R/W	R	*Used only for endpoints whose transfer type is programmable.* 1: Reserved 0: Configures endpoint to Bulk mode	0
MODE_IN	[5]	R/W	R	*Used only for endpoints whose direction is programmable.* 1: Configures Endpoint Direction as IN 0: Configures Endpoint Direction as OUT	1
IN_DMA_INT_EN	[4]	R/W	R	Determine whether the interrupt should be issued or not, when the IN_PKT_RDY condition happens. This is only useful for DMA mode. 0 = Interrupt enable, 1 = Interrupt Disable	0
Reserved	[3:0]	–	–	–	–

END POINT OUT CONTROL STATUS REGISTER (OUT_CSR1_REG/OUT_CSR2_REG)

Register	Address	R/W	Description	Reset Value
OUT_CSR1_REG	0x52000190(L) 0x52000193(B)	R/W (byte)	End Point out control status register1	0x00

OUT_CSR1_REG	Bit	MCU	USB	Description	Initial State
CLR_DATA_ TOGGLE	[7]	R/W	CLEAR	When the MCU writes a 1 to this bit, the data toggle sequence bit is reset to DATA0.	0
SENT_STALL	[6]	R/ CLEAR	SET	Set by the USB when an OUT token is ended with a STALL handshake. The USB issues a stall handshake to the host if it sends more than MAXP data for the OUT TOKEN.	0
SEND_STALL	[5]	R/W	R	0: The MCU clears this bit to end the STALL condition handshake, IN PKT RDY is cleared. 1: The MCU issues a STALL handshake to the USB. The MCU clears this bit to end the STALL condition handshake, IN PKT RDY is cleared.	0
FIFO_FLUSH	[4]	R/W	CLEAR	The MCU writes a 1 to flush the FIFO. This bit can be set only when OUT_PKT_RDY (D0) is set. The packet due to be unloaded by the MCU will be flushed.	0
Reserved	[3:1]	–	–	–	0
OUT_PKT_RDY	[0]	R/ CLEAR	SET	Set by the USB after it has loaded a packet of data into the FIFO. Once the MCU reads the packet from FIFO, this bit should be cleared by MCU (write a "0").	0

END POINT OUT CONTROL STATUS REGISTER (OUT_CSR1_REG/OUT_CSR2_REG) (Continued)

Register	Address	R/W	Description	Reset Value
OUT_CSR2_REG	0x52000194(L) 0x52000197(B)	R/W (byte)	End Point out control status register2	0x00

OUT_CSR2_REG	Bit	MCU	USB	Description	Initial State
AUTO_CLR	[7]	R/W	R	If the MCU is set, whenever the MCU reads data from the OUT FIFO, OUT_PKT_RDY will automatically be cleared by the logic without any intervention from the MCU.	0
ISO	[6]	R/W	R	Determine endpoint transfer type. 0: Configures endpoint to Bulk mode. 1: Reserved	0
OUT_DMA_INT_ MASK	[5]	R/W	R	Determine whether the interrupt should be issued or not. OUT_PKT_RDY condition happens. This is only useful for DMA mode 0 = Interrupt Enable 1 = Interrupt Disable	0

END POINT FIFO REGISTER (EPN_FIFO_REG)

The EPn_FIFO_REG enables the MCU to access to the EPn FIFO.

Register	Address	R/W	Description	Reset Value
EP0_FIFO	0x520001C0(L) 0x520001C3 (B)	R/W (byte)	End Point0 FIFO register	0xXX
EP1_FIFO	0x520001C4(L) 0x520001C7(B)	R/W (byte)	End Point1 FIFO register	0xXX
EP2_FIFO	0x520001C8(L) 0x520001CB(B)	R/W (byte)	End Point2 FIFO register	0xXX
EP3_FIFO	0x520001CC(L) 0x520001CF(B)	R/W (byte)	End Point3 FIFO register	0xXX
EP4_FIFO	0x520001D0(L) 0x520001D3(B)	R/W (byte)	End Point4 FIFO register	0xXX

EPn_FIFO	Bit	MCU	USB	Description	Initial State
FIFO_DATA	[7:0]	R/W	R/W	FIFO data value	0xXX

MAX PACKET REGISTER (MAXP_REG)

Register	Address	R/W	Description	Reset Value
MAXP_REG	0x52000180(L) 0x52000183(B)	R/W (byte)	End Point MAX packet register	0x01

MAXP_REG	Bit	MCU	USB	Description	Initial State
MAXP	[3:0]	R/W	R	0000: Reserved 0001: MAXP = 8 Byte0010: MAXP = 16 Byte 0100: MAXP = 32 Byte1000: MAXP = 64 Byte For EP0, MAXP=8 is recommended. For EP1~4, MAXP=32 or MAXP=64 is recommended. And, if MAXP=32, the dual packet mode will be enabled automatically.	0001

END POINT OUT WRITE COUNT REGISTER (OUT_FIFO_CNT1_REG/OUT_FIFO_CNT2_REG)

These registers maintain the number of bytes in the packet as the number is unloaded by the MCU.

Register	Address	R/W	Description	Reset Value
OUT_FIFO_CNT1_ REG	0x52000198(L) 0x5200019B(B)	R (byte)	End Point out write count register1	0x00

OUT_FIFO_CNT1_REG	Bit	MCU	USB	Description	Initial State
OUT_CNT_LOW	[7:0]	R	W	Lower byte of write count	0x00

Register	Address	R/W	Description	Reset Value
OUT_FIFO_CNT2_REG	0x5200019C(L) 0x5200019F(B)	R (byte)	End Point out write count register2	0x00

OUT_FIFO_CNT2_REG	Bit	MCU	USB	Description	Initial State
OUT_CNT_HIGH	[7:0]	R	W	Higher byte of write count. The OUT_CNT_HIGH may be always 0 normally.	0x00

DMA INTERFACE CONTROL REGISTER (EPN_DMA_CON)

Register	Address	R/W	Description	Reset Value
EP1_DMA_CON	0x52000200(L) 0x52000203(B)	R/W (byte)	EP1 DMA interface control register	0x00
EP2_DMA_CON	0x52000218(L) 0x5200021B(B)	R/W (byte)	EP2 DMA interface control register	0x00
EP3_DMA_CON	0x52000240(L) 0x52000243(B)	R/W (byte)	EP3 DMA interface control register	0x00
EP4_DMA_CON	0x52000258(L) 0x5200025B(B)	R/W (byte)	EP4 DMA interface control register	0x00

EPn_DMA_CON	Bit	MCU	USB	Description	Initial State
IN_RUN_OB	[7]	R/W	W	Read) IN_DMA_Run Observation 0: DMA is stopped 1:DMA is running Write) Ignore EPn_DMA_TTC_n register 0: DMA requests will be stopped if EPn_DMA_TTC_n reaches 0. 1: DMA requests will be continued although EPn_DMA_TTC_n reaches 0.	0
STATE	[6:4]	R	W	DMA State Monitoring	0
DEMAND_MODE	[3]	R/W	R	DMA Demand mode enable bit 0: Demand mode disable 1: Demand mode enable	0
OUT_RUN_OB/ OUT_DMA_RUN	[2]	R/W	R/W	Functionally separated into write and read operation. Write operation: "0" = Stop "1" = Run Read operation: OUT DMA Run Observation	0
IN_DMA_RUN	[1]	R/W	R	Start DMA operation. 0 = Stop 1 = Run	0
DMA_MODE_EN	[0]	R/W	R/ CLEAR	Set DMA mode.If the IN_RUN_OB has been wrtten as 0 and EPn_DMA_TTC_n reaches 0, DMA_MODE_EN bit will be cleared by USB. 0 = Interrupt mode 1 = DMA mode	0

DMA UNIT COUNTER REGISTER (EPN_DMA_UNIT)

This register is valid in Demand mode. In other modes, this register value must be set to "0x01"

Register	Address	R/W	Description	Reset Value
EP1_DMA_UNIT	0x52000204(L) 0x52000207(B)	R/W (byte)	EP1 DMA transfer unit counter base register	0x00
EP2_DMA_UNIT	0x5200021C(L) 0x5200021F(B)	R/W (byte)	EP2 DMA transfer unit counter base register	0x00
EP3_DMA_UNIT	0x52000244(L) 0x52000247(B)	R/W (byte)	EP3 DMA transfer unit counter base register	0x00
EP4_DMA_UNIT	0x5200025C(L) 0x5200025F(B)	R/W (byte)	EP4 DMA transfer unit counter base register	0x00

DMA_UNIT	Bit	MCU	USB	Description	Initial State
EPn_UNIT_CNT	[7:0]	R/W	R	EP DMA transfer unit counter value	0x00

DMA FIFO COUNTER REGISTER (EPN_DMA_FIFO)

This register has values in byte size in FIFO to be transferred by DMA. In case of OUT_DMA_RUN enabled, the value in OUT FIFO Write Count Register1 will be loaded in this register automatically. In case of IN DMA mode, the MCU should set proper value by software.

Register	Address	R/W	Description	Reset Value
EP1_DMA_FIFO	0x52000208(L) 0x5200020B(B)	R/W (byte)	EP1 DMA transfer FIFO counter base register	0x00
EP2_DMA_FIFO	0x52000220(L) 0x52000223(B)	R/W (byte)	EP2 DMA transfer FIFO counter base register	0x00
EP3_DMA_FIFO	0x52000248(L) 0x5200024B(B)	R/W (byte)	EP3 DMA transfer FIFO counter base register	0x00
EP4_DMA_FIFO	0x52000260(L) 0x52000263(B)	R/W (byte)	EP4 DMA transfer FIFO counter base register	0x00

DMA_FIFO	Bit	MCU	USB	Description	Initial State
EPn_FIFO_CNT	[7:0]	R/W	R	EP DMA transfer FIFO counter value	0x00

DMA TOTAL TRANSFER COUNTER REGISTER (EPN_DMA_TTC_L, M, H)

This register should have total number of bytes to be transferred using DMA (total 20-bit counter).

Register	Address	R/W	Description	Reset Value
EP1_DMA_TTC_L	0x5200020C(L) 0x5200020F(B)	R/W (byte)	EP1 DMA total transfer counter(lower byte)	0x00
EP1_DMA_TTC_M	0x52000210(L) 0x52000213(B)	R/W (byte)	EP1 DMA total transfer counter(middle byte)	0x00
EP1_DMA_TTC_H	0x52000214(L) 0x52000217(B)	R/W (byte)	EP1 DMA total transfer counter(higher byte)	0x00
EP2_DMA_TTC_L	0x52000224(L) 0x52000227(B)	R/W (byte)	EP2 DMA total transfer counter(lower byte)	0x00
EP2_DMA_TTC_M	0x52000228(L) 0x5200022B(B)	R/W (byte)	EP2 DMA total transfer counter(middle byte)	0x00
EP2_DMA_TTC_H	0x5200022C(L) 0x5200022F(B)	R/W (byte)	EP2 DMA total transfer counter(higher byte)	0x00
EP3_DMA_TTC_L	0x5200024C(L) 0x5200024F(B)	R/W (byte)	EP3 DMA total transfer counter(lower byte)	0x00
EP3_DMA_TTC_M	0x52000250(L) 0x52000253(B)	R/W (byte)	EP3 DMA total transfer counter(middle byte)	0x00
EP3_DMA_TTC_H	0x52000254(L) 0x52000257(B)	R/W (byte)	EP3 DMA total transfer counter(higher byte)	0x00
EP4_DMA_TTC_L	0x52000264(L) 0x52000267(B)	R/W (byte)	EP4 DMA total transfer counter(lower byte)	0x00
EP4_DMA_TTC_M	0x52000268(L) 0x5200026B(B)	R/W (byte)	EP4 DMA total transfer counter(middle byte)	0x00
EP4_DMA_TTC_H	0x5200026C(L) 0x5200026F(B)	R/W (byte)	EP4 DMA total transfer counter(higher byte)	0x00

DMA_TX	Bit	MCU	USB	Description	Initial State
EPn_TTC_L	[7:0]	R/W	R	DMA total transfer count value (lower byte)	0x00
EPn_TTC_M	[7:0]	R/W	R	DMA total transfer count value (middle byte)	0x00
EPn_TTC_H	[3:0]	R/W	R	DMA total transfer count value (higher byte)	0x00

14 INTERRUPT CONTROLLER

OVERVIEW

The interrupt controller in the S3C2410A receives the request from 56 interrupt sources. These interrupt sources are provided by internal peripherals such as the DMA controller, the UART, IIC, and others. In these interrupt sources, the UARTn and EINTn interrupts are 'OR'ed to the interrupt controller.

When receiving multiple interrupt requests from internal peripherals and external interrupt request pins, the interrupt controller requests FIQ or IRQ interrupt of the ARM920T core after the arbitration procedure.

The arbitration procedure depends on the hardware priority logic and the result is written to the interrupt pending register, which helps users notify which interrupt is generated out of various interrupt sources.

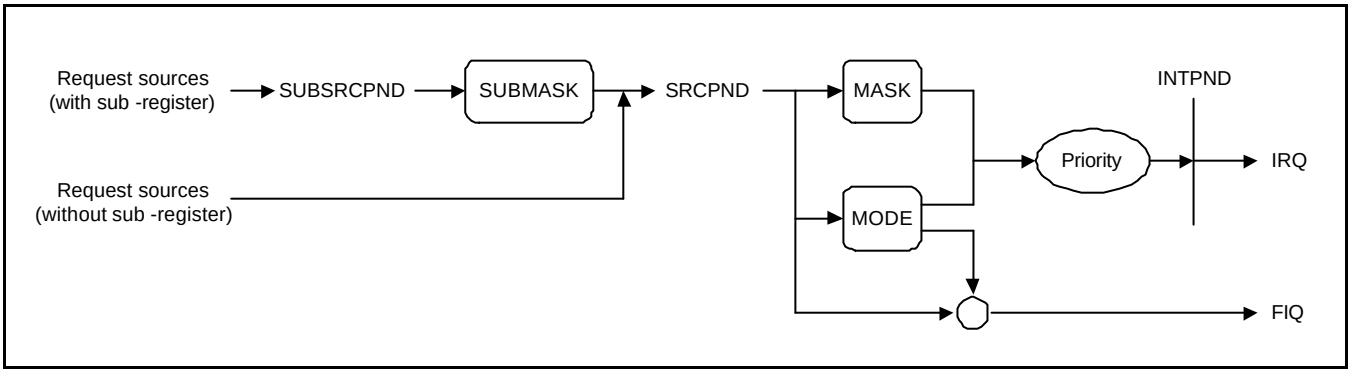


Figure 14-1. Interrupt Process Diagram

INTERRUPT CONTROLLER OPERATION

F-bit and I-bit of Program Status Register (PSR)

If the F-bit of PSR in ARM920T CPU is set to 1, the CPU does not accept the Fast Interrupt Request (FIQ) from the interrupt controller. Likewise, If I-bit of the PSR is set to 1, the CPU does not accept the Interrupt Request (IRQ) from the interrupt controller. So, the interrupt controller can receive interrupts by clearing F-bit or I-bit of the PSR to 0 and setting the corresponding bit of INTMSK to 0.

Interrupt Mode

The ARM920T has two types of Interrupt mode: FIQ or IRQ. All the interrupt sources determine which mode is used at interrupt request.

Interrupt Pending Register

The S3C2410A has two interrupt pending resisters: source pending register (SRCPND) and interrupt pending register (INTPND). These pending registers indicate whether or not an interrupt request is pending. When the interrupt sources request interrupt service, the corresponding bits of SRCPND register are set to 1, and at the same time, only one bit of the INTPND register is set to 1 automatically after arbitration procedure. If interrupts are masked, the corresponding bits of the SRCPND register are set to 1. This does not cause the bit of INTPND register changed. When a pending bit of the INTPND register is set, the interrupt service routine starts whenever the I-flag or F-flag is cleared to 0. The SRCPND and INTPND registers can be read and written, so the service routine must clear the pending condition by writing a 1 to the corresponding bit in the SRCPND register first and then clear the pending condition in the INTPND registers by using the same method.

Interrupt Mask Register

This register indicates that an interrupt has been disabled if the corresponding mask bit is set to 1. If an interrupt mask bit of INTMSK is 0, the interrupt will be serviced normally. If the corresponding mask bit is 1 and the interrupt is generated, the source pending bit will be set.

INTERRUPT SOURCES

The interrupt controller supports 56 interrupt sources as shown in the table below.

Sources	Descriptions	Arbiter Group
INT_ADC	ADC EOC and Touch interrupt (INT_ADC/INT_TC)	ARB5
INT_RTC	RTC alarm interrupt	ARB5
INT_SPI1	SPI1 interrupt	ARB5
INT_UART0	UART0 Interrupt (ERR, RXD, and TXD)	ARB5
INT_IIC	IIC interrupt	ARB4
INT_USBH	USB Host interrupt	ARB4
INT_USBD	USB Device interrupt	ARB4
Reserved	Reserved	ARB4
INT_UART1	UART1 Interrupt (ERR, RXD, and TXD)	ARB4
INT_SPI0	SPI0 interrupt	ARB4
INT_SDI	SDI interrupt	ARB 3
INT_DMA3	DMA channel 3 interrupt	ARB3
INT_DMA2	DMA channel 2 interrupt	ARB3
INT_DMA1	DMA channel 1 interrupt	ARB3
INT_DMA0	DMA channel 0 interrupt	ARB3
INT_LCD	LCD interrupt (INT_FrSyn and INT_FiCnt)	ARB3
INT_UART2	UART2 Interrupt (ERR, RXD, and TXD)	ARB2
INT_TIMER4	Timer4 interrupt	ARB2
INT_TIMER3	Timer3 interrupt	ARB2
INT_TIMER2	Timer2 interrupt	ARB2
INT_TIMER1	Timer1 interrupt	ARB 2
INT_TIMER0	Timer0 interrupt	ARB2
INT_WDT	Watch-Dog timer interrupt	ARB1
INT_TICK	RTC Time tick interrupt	ARB1
nBATT_FLT	Battery Fault interrupt	ARB1
Reserved	Reserved	ARB1
EINT8_23	External interrupt 8 – 23	ARB1
EINT4_7	External interrupt 4 – 7	ARB1
EINT3	External interrupt 3	ARB0
EINT2	External interrupt 2	ARB0
EINT1	External interrupt 1	ARB0
EINT0	External interrupt 0	ARB0

INTERRUPT PRIORITY GENERATING BLOCK

The priority logic for 32 interrupt requests is composed of seven rotation based arbiters: six first-level arbiters and one second-level arbiter as shown in Figure 14-2 below.

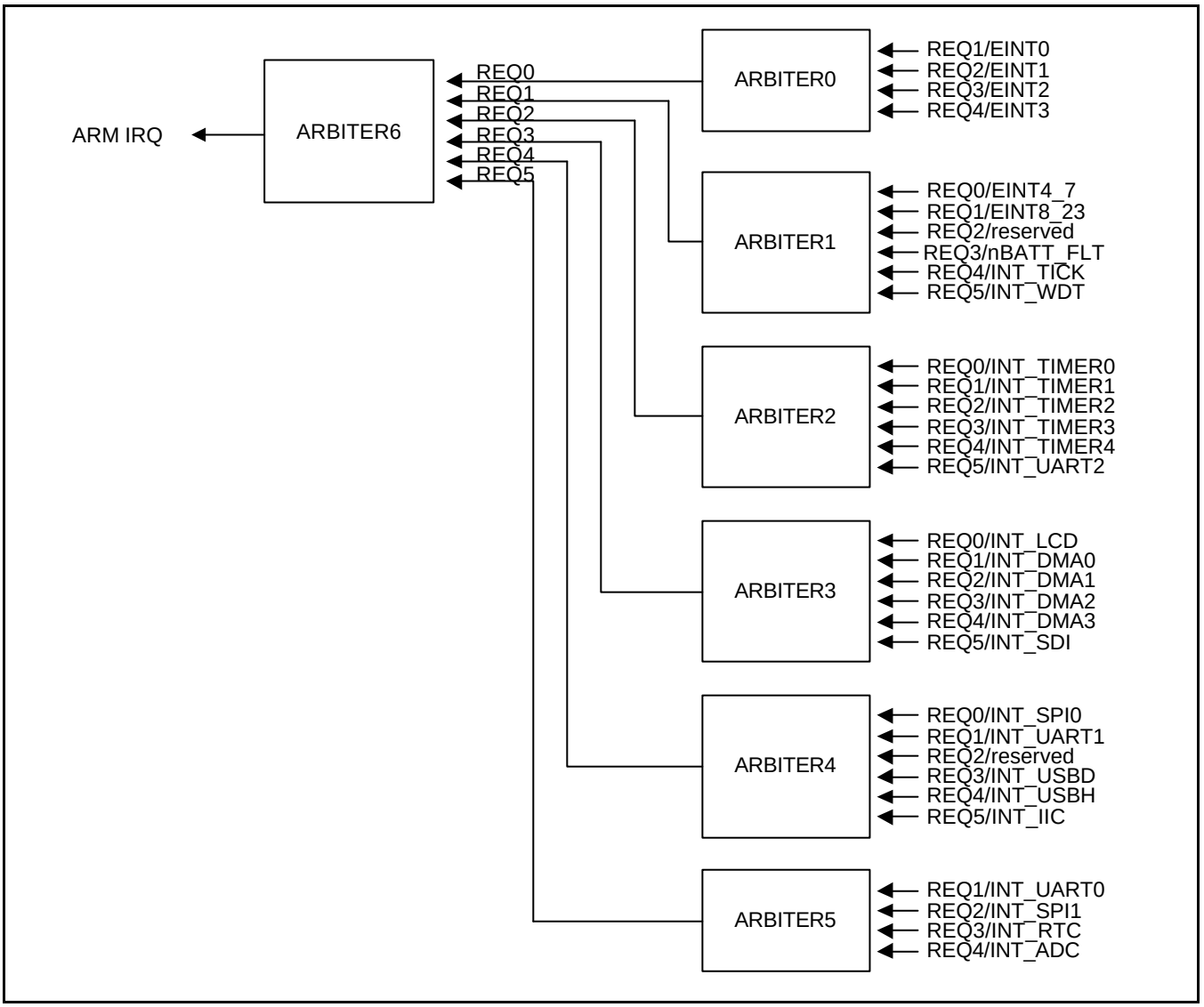


Figure 14-2. Priority Generating Block

INTERRUPT PRIORITY

Each arbiter can handle six interrupt requests based on the one bit arbiter mode control (ARB_MODE) and two bits of selection control signals (ARB_SEL) as follows:

— If ARB_SEL bits are 00b, the priority order is REQ0, REQ1, REQ2, REQ3, REQ4, and REQ5.

— If ARB_SEL bits are 01b, the priority order is REQ0, REQ2, REQ3, REQ4, REQ1, and REQ5.

— If ARB_SEL bits are 10b, the priority order is REQ0, REQ3, REQ4, REQ1, REQ2, and REQ5.

— If ARB_SEL bits are 11b, the priority order is REQ0, REQ4, REQ1, REQ2, REQ3, and REQ5.

Note that REQ0 of an arbiter always has the highest priority, and REQ5 has the lowest one. In addition, by changing the ARB_SEL bits, we can rotate the priority of REQ1 to REQ4.

Here, if ARB_MODE bit is set to 0, ARB_SEL bits are not automatically changed, making the arbiter to operate in the fixed priority mode (note that even in this mode, we can reconfigure the priority by manually changing the ARB_SEL bits). On the other hand, if ARB_MODE bit is 1, ARB_SEL bits are changed in rotation fashion, e.g., if REQ1 is serviced, ARB_SEL bits are changed to 01b automatically so as to put REQ1 into the lowest priority. The detailed rules of ARB_SEL change are as follows:

— If REQ0 or REQ5 is serviced, ARB_SEL bits are not changed at all.

— If REQ1 is serviced, ARB_SEL bits are changed to 01b.

— If REQ2 is serviced, ARB_SEL bits are changed to 10b.

— If REQ3 is serviced, ARB_SEL bits are changed to 11b.

— If REQ4 is serviced, ARB_SEL bits are changed to 00b.

INTERRUPT CONTROLLER SPECIAL REGISTERS

There are five control registers in the interrupt controller: source pending register, interrupt mode register, mask register, priority register, and interrupt pending register.

All the interrupt requests from the interrupt sources are first registered in the source pending register. They are divided into two groups including Fast Interrupt Request (FIQ) and Interrupt Request (IRQ), based on the interrupt mode register. The arbitration procedure for multiple IRQs is based on the priority register.

SOURCE PENDING (SRCPND) REGISTER

The SRCPND register is composed of 32 bits each of which is related to an interrupt source. Each bit is set to 1 if the corresponding interrupt source generates the interrupt request and waits for the interrupt to be serviced. Accordingly, this register indicates which interrupt source is waiting for the request to be serviced. Note that each bit of the SRCPND register is automatically set by the interrupt sources regardless of the masking bits in the INTMASK register. In addition, the SRCPND register is not affected by the priority logic of interrupt controller.

In the interrupt service routine for a specific interrupt source, the corresponding bit of the SRCPND register has to be cleared to get the interrupt request from the same source correctly. If you return from the ISR without clearing the bit, the interrupt controller operates as if another interrupt request came in from the same source. In other words, if a specific bit of the SRCPND register is set to 1, it is always considered as a valid interrupt request waiting to be serviced.

The time to clear the corresponding bit depends on the user's requirement. If you want to receive another valid request from the same source, you should clear the corresponding bit first, and then enable the interrupt.

You can clear a specific bit of the SRCPND register by writing a data to this register. It clears only the bit positions of the SRCPND corresponding to those set to one in the data. The bit positions corresponding to those that are set to 0 in the data remains as they are.

Register	Address	R/W	Description	Reset Value
SRCPND	0X4A000000	R/W	Indicate the interrupt request status. 0 = The interrupt has not been requested. 1 = The interrupt source has asserted the interrupt request.	0x00000000

SOURCE PENDING (SRCPND) REGISTER (Continued)

SRCPND	Bit	Description	Initial State
INT_ADC	[31]	0 = Not requested, 1 = Requested	0
INT_RTC	[30]	0 = Not requested, 1 = Requested	0
INT_SPI1	[29]	0 = Not requested, 1 = Requested	0
INT_UART0	[28]	0 = Not requested, 1 = Requested	0
INT_IIC	[27]	0 = Not requested, 1 = Requested	0
INT_USBH	[26]	0 = Not requested, 1 = Requested	0
INT_USBD	[25]	0 = Not requested, 1 = Requested	0
Reserved	[24]	Not used	0
INT_UART1	[23]	0 = Not requested, 1 = Requested	0
INT_SPI0	[22]	0 = Not requested, 1 = Requested	0
INT_SDI	[21]	0 = Not requested, 1 = Requested	0
INT_DMA3	[20]	0 = Not requested, 1 = Requested	0
INT_DMA2	[19]	0 = Not requested, 1 = Requested	0
INT_DMA1	[18]	0 = Not requested, 1 = Requested	0
INT_DMA0	[17]	0 = Not requested, 1 = Requested	0
INT_LCD	[16]	0 = Not requested, 1 = Requested	0
INT_UART2	[15]	0 = Not requested, 1 = Requested	0
INT_TIMER4	[14]	0 = Not requested, 1 = Requested	0
INT_TIMER3	[13]	0 = Not requested, 1 = Requested	0
INT_TIMER2	[12]	0 = Not requested, 1 = Requested	0
INT_TIMER1	[11]	0 = Not requested, 1 = Requested	0
INT_TIMER0	[10]	0 = Not requested, 1 = Requested	0
INT_WDT	[9]	0 = Not requested, 1 = Requested	0
INT_TICK	[8]	0 = Not requested, 1 = Requested	0
nBATT_FLT	[7]	0 = Not requested, 1 = Requested	0
Reserved	[6]	Not used	0
EINT8_23	[5]	0 = Not requested, 1 = Requested	0
EINT4_7	[4]	0 = Not requested, 1 = Requested	0
EINT3	[3]	0 = Not requested, 1 = Requested	0
EINT2	[2]	0 = Not requested, 1 = Requested	0
EINT1	[1]	0 = Not requested, 1 = Requested	0
EINT0	[0]	0 = Not requested, 1 = Requested	0

INTERRUPT MODE (INTMOD) REGISTER

This register is composed of 32 bits each of which is related to an interrupt source. If a specific bit is set to 1, the corresponding interrupt is processed in the FIQ (fast interrupt) mode. Otherwise, it is processed in the IRQ mode (normal interrupt).

Note that only one interrupt source can be serviced in the FIQ mode in the interrupt controller (you should use the FIQ mode only for the urgent interrupt). Thus, only one bit of INTMOD can be set to 1.

Register	Address	R/W	Description	Reset Value
INTMOD	0X4A000004	R/W	Interrupt mode regiseter. 0 = IRQ mode 1 = FIQ mode	0x00000000

NOTE: If an interrupt mode is set to FIQ mode in the INTMOD register, FIQ interrupt will not affect both INTPND and INTOFFSET registers. In this case, the two registers are valid only for IRQ mode interrupt source.

INTMOD	Bit	Description	Initial State
INT_ADC	[31]	0 = IRQ, 1 = FIQ	0
INT_RTC	[30]	0 = IRQ, 1 = FIQ	0
INT_SPI1	[29]	0 = IRQ, 1 = FIQ	0
INT_UART0	[28]	0 = IRQ, 1 = FIQ	0
INT_IIC	[27]	0 = IRQ, 1 = FIQ	0
INT_USBH	[26]	0 = IRQ, 1 = FIQ	0
INT_USBD	[25]	0 = IRQ, 1 = FIQ	0
Reserved	[24]	Not used	0
INT_URRT1	[23]	0 = IRQ, 1 = FIQ	0
INT_SPI0	[22]	0 = IRQ, 1 = FIQ	0
INT_SDI	[21]	0 = IRQ, 1 = FIQ	0
INT_DMA3	[20]	0 = IRQ, 1 = FIQ	0
INT_DMA2	[19]	0 = IRQ, 1 = FIQ	0
INT_DMA1	[18]	0 = IRQ, 1 = FIQ	0
INT_DMA0	[17]	0 = IRQ, 1 = FIQ	0
INT_LCD	[16]	0 = IRQ, 1 = FIQ	0
INT_UART2	[15]	0 = IRQ, 1 = FIQ	0
INT_TIMER4	[14]	0 = IRQ, 1 = FIQ	0
INT_TIMER3	[13]	0 = IRQ, 1 = FIQ	0
INT_TIMER2	[12]	0 = IRQ, 1 = FIQ	0
INT_TIMER1	[11]	0 = IRQ, 1 = FIQ	0
INT_TIMER0	[10]	0 = IRQ, 1 = FIQ	0
INT_WDT	[9]	0 = IRQ, 1 = FIQ	0
INT_TICK	[8]	0 = IRQ, 1 = FIQ	0
nBATT_FLT	[7]	0 = IRQ, 1 = FIQ	0
Reserved	[6]	Not used	0
EINT8_23	[5]	0 = IRQ, 1 = FIQ	0
EINT4_7	[4]	0 = IRQ, 1 = FIQ	0
EINT3	[3]	0 = IRQ, 1 = FIQ	0
EINT2	[2]	0 = IRQ, 1 = FIQ	0
EINT1	[1]	0 = IRQ, 1 = FIQ	0
EINT0	[0]	0 = IRQ, 1 = FIQ	0

INTERRUPT MASK (INTMSK) REGISTER

This register also has 32 bits each of which is related to an interrupt source. If a specific bit is set to 1, the CPU does not service the interrupt request from the corresponding interrupt source (note that even in such a case, the corresponding bit of SRCPND register is set to 1). If the mask bit is 0, the interrupt request can be serviced.

Register	Address	R/W	Description	Reset Value
INTMSK	0X4A000008	R/W	Determine which interrupt source is masked. The masked interrupt source will not be serviced. 0 = Interrupt service is available. 1 = Interrupt service is masked.	0xFFFFFFFF

INTMSK	Bit	Description	Initial State
INT_ADC	[31]	0 = Service available, 1 = Masked	1
INT_RTC	[30]	0 = Service available, 1 = Masked	1
INT_SPI1	[29]	0 = Service available, 1 = Masked	1
INT_UART0	[28]	0 = Service available, 1 = Masked	1
INT_IIC	[27]	0 = Service available, 1 = Masked	1
INT_USBH	[26]	0 = Service available, 1 = Masked	1
INT_USBD	[25]	0 = Service available, 1 = Masked	1
Reserved	[24]	Not used	1
INT_UART1	[23]	0 = Service available, 1 = Masked	1
INT_SPI0	[22]	0 = Service available, 1 = Masked	1
INT_SDI	[21]	0 = Service available, 1 = Masked	1
INT_DMA3	[20]	0 = Service available, 1 = Masked	1
INT_DMA2	[19]	0 = Service available, 1 = Masked	1
INT_DMA1	[18]	0 = Service available, 1 = Masked	1
INT_DMA0	[17]	0 = Service available, 1 = Masked	1
INT_LCD	[16]	0 = Service available, 1 = Masked	1
INT_UART2	[15]	0 = Service available, 1 = Masked	1
INT_TIMER4	[14]	0 = Service available, 1 = Masked	1
INT_TIMER3	[13]	0 = Service available, 1 = Masked	1
INT_TIMER2	[12]	0 = Service available, 1 = Masked	1
INT_TIMER1	[11]	0 = Service available, 1 = Masked	1
INT_TIMER0	[10]	0 = Service available, 1 = Masked	1
INT_WDT	[9]	0 = Service available, 1 = Masked	1
INT_TICK	[8]	0 = Service available, 1 = Masked	1
nBATT_FLT	[7]	0 = Service available, 1 = Masked	1
Reserved	[6]	Not used	1
EINT8_23	[5]	0 = Service available, 1 = Masked	1
EINT4_7	[4]	0 = Service available, 1 = Masked	1
EINT3	[3]	0 = Service available, 1 = Masked	1
EINT2	[2]	0 = Service available, 1 = Masked	1
EINT1	[1]	0 = Service available, 1 = Masked	1
EINT0	[0]	0 = Service available, 1 = Masked	1

PRIORITY REGISTER (PRIORITY)

Register	Address	R/W	Description	Reset Value
PRIORITY	0x4A00000C	R/W	IRQ priority control register	0x7F

PRIORITY	Bit	Description	Initial State
ARB_SEL6	[20:19]	Arbiter 6 group priority order set 00 = REQ 0-1-2-3-4-5 01 = REQ 0-2-3-4-1-5 10 = REQ 0-3-4-1-2-5 11 = REQ 0-4-1-2-3-5	0
ARB_SEL5	[18:17]	Arbiter 5 group priority order set 00 = REQ 1-2-3-4 01 = REQ 2-3-4-1 10 = REQ 3-4-1-2 11 = REQ 4-1-2-3	0
ARB_SEL4	[16:15]	Arbiter 4 group priority order set 00 = REQ 0-1-2-3-4-5 01 = REQ 0-2-3-4-1-5 10 = REQ 0-3-4-1-2-5 11 = REQ 0-4-1-2-3-5	0
ARB_SEL3	[14:13]	Arbiter 3 group priority order set 00 = REQ 0-1-2-3-4-5 01 = REQ 0-2-3-4-1-5 10 = REQ 0-3-4-1-2-5 11 = REQ 0-4-1-2-3-5	0
ARB_SEL2	[12:11]	Arbiter 2 group priority order set 00 = REQ 0-1-2-3-4-5 01 = REQ 0-2-3-4-1-5 10 = REQ 0-3-4-1-2-5 11 = REQ 0-4-1-2-3-5	0
ARB_SEL1	[10:9]	Arbiter 1 group priority order set 00 = REQ 0-1-2-3-4-5 01 = REQ 0-2-3-4-1-5 10 = REQ 0-3-4-1-2-5 11 = REQ 0-4-1-2-3-5	0
ARB_SEL0	[8:7]	Arbiter 0 group priority order set 00 = REQ 1-2-3-4 01 = REQ 2-3-4-1 10 = REQ 3-4-1-2 11 = REQ 4-1-2-3	0
ARB_MODE6	[6]	Arbiter 6 group priority rotate enable 0 = Priority does not rotate 1 = Priority rotate enable	1
ARB_MODE5	[5]	Arbiter 5 group priority rotate enable 0 = Priority does not rotate 1 = Priority rotate enable	1
ARB_MODE4	[4]	Arbiter 4 group priority rotate enable 0 = Priority does not rotate 1 = Priority rotate enable	1
ARB_MODE3	[3]	Arbiter 3 group priority rotate enable 0 = Priority does not rotate 1 = Priority rotate enable	1
ARB_MODE2	[2]	Arbiter 2 group priority rotate enable 0 = Priority does not rotate 1 = Priority rotate enable	1

PRIORITY REGISTER (PRIORITY) (Continued)

PRIORITY	Bit	Description	Initial State
ARB_MODE1	[1]	Arbiter 1 group priority rotate enable 0 = Priority does not rotate 1 = Priority rotate enable	1
ARB_MODE0	[0]	Arbiter 0 group priority rotate enable 0 = Priority does not rotate 1 = Priority rotate enable	1

INTERRUPT PENDING (INTPND) REGISTER

Each of the 32 bits in the interrupt pending register shows whether the corresponding interrupt request, which is unmasked and waits for the interrupt to be serviced, has the highest priority . Since the INTPND register is located after the priority logic, only one bit can be set to 1, and that interrupt request generates IRQ to CPU. In interrupt service routine for IRQ, you can read this register to determine which interrupt source is serviced among the 32 sources.

Like the SRCPND register, this register has to be cleared in the interrupt service routine after clearing the SRCPND register. We can clear a specific bit of the INTPND register by writing a data to this register. It clears only the bit positions of the INTPND register corresponding to those set to one in the data. The bit positions corresponding to those that are set to 0 in the data remains as they are.

Register	Address	R/W	Description	Reset Value
INTPND	0X4A000010	R/W	Indicate the interrupt request status. 0 = The interrupt has not been requested. 1 = The interrupt source has asserted the interrupt request.	0x00000000

NOTES:

1. If the FIQ mode interrupt occurs, the corresponding bit of INTPND will not be turned on as the INTPND register is available only for IRQ mode interrupt.
2. Cautions in clearing the INTPND register.
   The INTPND register is cleared to "0" by writing "1". If the INTPND bit, which has "1", is cleared by "0", the INTPND register & INTOFFSET register may have unexpected value in some case.
   So, you never write "0" on the INTPND bit having "1". The convenient method to clear the INTPND register is writing the INTPND register value on the INTPND register. (In even our example code, this guide hasn't been applied yet.)

INTPND	Bit	Description	Initial State
INT_ADC	[31]	0 = Not requested, 1 = Requested	0
INT_RTC	[30]	0 = Not requested, 1 = Requested	0
INT_SPI1	[29]	0 = Not requested, 1 = Requested	0
INT_UART0	[28]	0 = Not requested, 1 = Requested	0
INT_IIC	[27]	0 = Not requested, 1 = Requested	0
INT_USBH	[26]	0 = Not requested, 1 = Requested	0
INT_USBD	[25]	0 = Not requested, 1 = Requested	0
Reserved	[24]	Not used	0
INT_UART1	[23]	0 = Not requested, 1 = Requested	0
INT_SPI0	[22]	0 = Not requested, 1 = Requested	0
INT_SDI	[21]	0 = Not requested, 1 = Requested	0
INT_DMA3	[20]	0 = Not requested, 1 = Requested	0
INT_DMA2	[19]	0 = Not requested, 1 = Requested	0
INT_DMA1	[18]	0 = Not requested, 1 = Requested	0
INT_DMA0	[17]	0 = Not requested, 1 = Requested	0
INT_LCD	[16]	0 = Not requested, 1 = Requested	0
INT_UART2	[15]	0 = Not requested, 1 = Requested	0
INT_TIMER4	[14]	0 = Not requested, 1 = Requested	0
INT_TIMER3	[13]	0 = Not requested, 1 = Requested	0
INT_TIMER2	[12]	0 = Not requested, 1 = Requested	0
INT_TIMER1	[11]	0 = Not requested, 1 = Requested	0
INT_TIMER0	[10]	0 = Not requested, 1 = Requested	0
INT_WDT	[9]	0 = Not requested, 1 = Requested	0
INT_TICK	[8]	0 = Not requested, 1 = Requested	0
nBATT_FLT	[7]	0 = Not requested, 1 = Requested	0
Reserved	[6]	Not used	0
EINT8_23	[5]	0 = Not requested, 1 = Requested	0
EINT4_7	[4]	0 = Not requested, 1 = Requested	0
EINT3	[3]	0 = Not requested, 1 = Requested	0
EINT2	[2]	0 = Not requested, 1 = Requested	0
EINT1	[1]	0 = Not requested, 1 = Requested	0
EINT0	[0]	0 = Not requested, 1 = Requested	0

INTERRUPT OFFSET (INTOFFSET) REGISTER

The value in the interrupt offset register shows which interrupt request of IRQ mode is in the INTPND register. This bit can be cleared automatically by clearing SRCPND and INTPND.

Register	Address	R/W	Description	Reset Value
INTOFFSET	0X4A000014	R	Indicate the IRQ interrupt request source	0x00000000

INT Source	The OFFSET Value	INT Source	The OFFSET Value
INT_ADC	31	INT_UART2	15
INT_RTC	30	INT_TIMER4	14
INT_SPI1	29	INT_TIMER3	13
INT_UART0	28	INT_TIMER2	12
INT_IIC	27	INT_TIMER1	11
INT_USBH	26	INT_TIMER0	10
INT_USBD	25	INT_WDT	9
Reserved	24	INT_TICK	8
INT_UART1	23	nBATT_FLT	7
INT_SPI0	22	Reserved	6
INT_SDI	21	EINT8_23	5
INT_DMA3	20	EINT4_7	4
INT_DMA2	19	EINT3	3
INT_DMA1	18	EINT2	2
INT_DMA0	17	EINT1	1
INT_LCD	16	EINT0	0

NOTE: FIQ mode interrupt does not affect the INTOFFSET register as the register is available only for IRQ mode interrupt.

SUB SOURCE PENDING (SUBSRCPND) REGISTER

You can clear a specific bit of the SUBSRCPND register by writing a data to this register. It clears only the bit positions of the SUBSRCPND register corresponding to those set to one in the data. The bit positions corresponding to those that are set to 0 in the data remains as they are.

Register	Address	R/W	Description	Reset Value
SUBSRCPND	0X4A000018	R/W	Indicate the interrupt request status. 0 = The interrupt has not been requested. 1 = The interrupt source has asserted the interrupt request.	0x00000000

SUBSRCPND	Bit	Description	Initial State
Reserved	[31:11]	Not used	0
INT_ADC	[10]	0 = Not requested, 1 = Requested	0
INT_TC	[9]	0 = Not requested, 1 = Requested	0
INT_ERR2	[8]	0 = Not requested, 1 = Requested	0
INT_TXD2	[7]	0 = Not requested, 1 = Requested	0
INT_RXD2	[6]	0 = Not requested, 1 = Requested	0
INT_ERR1	[5]	0 = Not requested, 1 = Requested	0
INT_TXD1	[4]	0 = Not requested, 1 = Requested	0
INT_RXD1	[3]	0 = Not requested, 1 = Requested	0
INT_ERR0	[2]	0 = Not requested, 1 = Requested	0
INT_TXD0	[1]	0 = Not requested, 1 = Requested	0
INT_RXD0	[0]	0 = Not requested, 1 = Requested	0

INTERRUPT SUB MASK (INTSUBMSK) REGISTER

This register has 11 bits each of which is related to an interrupt source. If a specific bit is set to 1, the interrupt request from the corresponding interrupt source is not serviced by the CPU (note that even in such a case, the corresponding bit of the SUBSRCPND register is set to 1). If the mask bit is 0, the interrupt request can be serviced.

Register	Address	R/W	Description	Reset Value
INTSUBMSK	0X4A00001C	R/W	Determine which interrupt source is masked. The masked interrupt source will not be serviced. 0 = Interrupt service is available. 1 = Interrupt service is masked.	0x7FF

INTSUBMSK	Bit	Description	Initial State
Reserved	[31:11]	Not used	0
INT_ADC	[10]	0 = Service available, 1 = Masked	1
INT_TC	[9]	0 = Service available, 1 = Masked	1
INT_ERR2	[8]	0 = Service available, 1 = Masked	1
INT_TXD2	[7]	0 = Service available, 1 = Masked	1
INT_RXD2	[6]	0 = Service available, 1 = Masked	1
INT_ERR1	[5]	0 = Service available, 1 = Masked	1
INT_TXD1	[4]	0 = Service available, 1 = Masked	1
INT_RXD1	[3]	0 = Service available, 1 = Masked	1
INT_ERR0	[2]	0 = Service available, 1 = Masked	1
INT_TXD0	[1]	0 = Service available, 1 = Masked	1
INT_RXD0	[0]	0 = Service available, 1 = Masked	1

15 LCD CONTROLLER

OVERVIEW

The LCD controller in the S3C2410A consists of the logic for transferring LCD image data from a video buffer located in system memory to an external LCD driver.

The LCD controller supports monochrome, 2-bit per pixel (4-level gray scale) or 4-bit per pixel (16-level gray scale) mode on a monochrome LCD, using a time-based dithering algorithm and Frame Rate Control (FRC) method and it can be interfaced with a color LCD panel at 8-bit per pixel (256-level color) and 12-bit per pixel (4096-level color) for interfacing with STN LCD.
It can support 1-bit per pixel, 2-bit per pixel, 4-bit per pixel, and 8-bit per pixel for interfacing with the palettized TFT color LCD panel, and 16-bit per pixel and 24-bit per pixel for non-palettized true-color display.

The LCD controller can be programmed to support different requirements on the screen related to the number of horizontal and vertical pixels, data line width for the data interface, interface timing, and refresh rate.

FEATURES

STN LCD displays:

— Supports 3 types of LCD panels: 4-bit dual scan, 4-bit single scan, and 8-bit single scan display type

— Supports the monochrome, 4 gray levels, and 16 gray levels

— Supports 256 colors and 4096 colors for color STN LCD panel

— Supports multiple screen size
Typical actual screen size: 640×480, 320×240, 160×160, and others
Maximum virtual screen size is 4Mbytes.
Maximum virtual screen size in 256 color mode: 4096×1024, 2048×2048, 1024×4096, and others

TFT LCD displays:

— Supports 1, 2, 4 or 8-bpp (bit per pixel) palettized color displays for TFT

— Supports 16-bpp non-palettized true-color displays for color TFT

— Supports 24-bpp non-palettized true-color displays for color TFT

— Supports maximum 16M color TFT at 24-bit per pixel mode

— Supports multiple screen size
Typical actual screen size: 640×480, 320×240, 160×160, and others
Maximum virtual screen size is 4Mbytes.
Maximum virtual screen size in 64K color mode: 2048×1024 and others

COMMON FEATURES

The LCD controller has a dedicated DMA that supports to fetch the image data from video buffer located in system memory. Its features also include:

— Dedicated interrupt functions (INT_FrSyn and INT_FiCnt)

— The system memory is used as the display memory.

— Supports Multiple Virtual Display Screen (Supports Hardware Horizontal/Vertical Scrolling)

— Programmable timing control for different display panels

— Supports little and big-endian byte ordering, as well as WinCE data formats

— Supports SEC TFT LCD panel (SAMSUNG 3.5″ Portrait/256K Color/Reflective a-Si TFT LCD)
LTS350Q1-PD1: TFT LCD panel with touch panel and front light unit
LTS350Q1-PD2: TFT LCD panel only

NOTE

WinCE doesn't support the 12-bit packed data format.
Please check if WinCE can support the 12-bit color-mode.

EXTERNAL INTERFACE SIGNAL

VFRAME/VSYNC/STV	: Frame synchronous signal (STN)/vertical synchronous signal (TFT)/SEC TFT signal
VLINE/HSYNC/CPV	: Line synchronous pulse signal (STN)/horizontal sync signal (TFT)/SEC TFT signal
VCLK/LCD_HCLK	: Pixel clock signal (STN/TFT)/SEC TFT signal
VD[23:0]	: LCD pixel data output ports (STN/TFT/SEC TFT)
VM/VDEN/TP	: AC bias signal for the LCD driver (STN)/data enable signal (TFT)/SEC TFT signal
LEND/STH	: Line end signal (TFT)/SEC TFT signal
LCD_PWREN	: LCD panel power enable control signal
LCDVF0	: SEC TFT Signal OE
LCDVF1	: SEC TFT Signal REV
LCDVF2	: SEC TFT Signal REVB

The 33 output ports in total includes 24 data bits and 9 control bits

BLOCK DIAGRAM

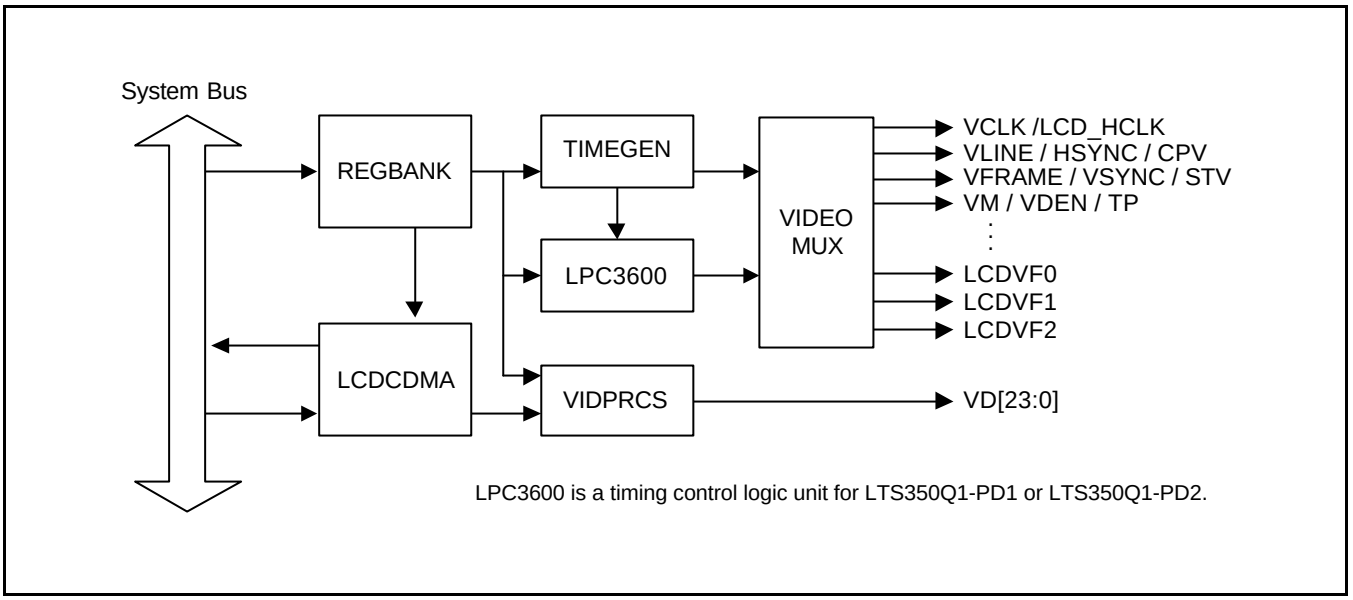


Figure 15-1. LCD Controller Block Diagram

The S3C2410A LCD controller is used to transfer the video data and to generate the necessary control signals, such as VFRAME, VLINE, VCLK, VM, and so on. In addition to the control signals, the S3C2410A has the data ports for video data, which are VD[23:0] as shown in Figure 15-1. The LCD controller consists of a REGBANK, LCDCDMA, VIDPRCS, TIMEGEN, and LPC3600 (See the Figure 15-1 LCD Controller Block Diagram). The REGBANK has 17 programmable register sets and 256x16 palette memory which are used to configure the LCD controller. The LCDCDMA is a dedicated DMA, which can transfer the video data in frame memory to LCD driver automatically. By using this special DMA, the video data can be displayed on the screen without CPU intervention. The VIDPRCS receives the video data from the LCDCDMA and sends the video data through the VD[23:0] data ports to the LCD driver after changing them into a suitable data format, for example 4/8-bit single scan or 4-bit dual scan display mode. The TIMEGEN consists of programmable logic to support the variable requirements of interface timing and rates commonly found in different LCD drivers. The TIMEGEN block generates VFRAME, VLINE, VCLK, VM, and so on.

The description of data flow is as follows:

FIFO memory is present in the LCDCDMA. When FIFO is empty or partially empty, the LCDCDMA requests data fetching from the frame memory based on the burst memory transfer mode (consecutive memory fetching of 4 words (16 bytes) per one burst request without allowing the bus mastership to another bus master during the bus transfer). When the transfer request is accepted by bus arbitrator in the memory controller, there will be four successive word data transfers from system memory to internal FIFO. The total size of FIFO is 28 words, which consists of 12 words FIFOL and 16 words FIFOH, respectively. The S3C2410A has two FIFOs to support the dual scan display mode. In case of single scan mode, one of the FIFOs (FIFOH) can only be used.

STN LCD CONTROLLER OPERATION

TIMING GENERATOR (TIMEGEN)

The TIMEGEN generates the control signals for the LCD driver, such as VFRAME, VLINE, VCLK, and VM. These control signals are closely related to the configuration on the LCDCON1/2/3/4/5 registers in the REGBANK. Based on these programmable configurations on the LCD control registers in the REGBANK, the TIMEGEN can generate the programmable control signals suitable to support many different types of LCD drivers.

The VFRAME pulse is asserted for the duration of the entire first line at a frequency of once per frame. The VFRAME signal is asserted to bring the LCD's line pointer to the top of the display to start over.

The VM signal helps the LCD driver alternate the polarity of the row and column voltages, which are used to turn the pixel on and off. The toggling rate of VM signals depends on the MMODE bit of the LCDCON1 register and MVAL field of the LCDCON4 register. If the MMODE bit is 0, the VM signal is configured to toggle on every frame. If the MMODE bit is 1, the VM signal is configured to toggle on the every event of the elapse of the specified number of VLINE by the MVAL[7:0] value. Figure 15-4 shows an example for MMODE = 0 and for MMODE = 1 with the value of MVAL[7:0] = 0x2. When MMODE = 1, the VM rate is related to MVAL[7:0], as shown below:

VM Rate = VLINE Rate / (2 * MVAL)

The VFRAME and VLINE pulse generation relies on the configurations of the HOZVAL field and the LINEVAL field in the LCDCON2/3 register. Each field is related to the LCD size and display mode. In other words, the HOZVAL and LINEVAL can be determined by the size of the LCD panel and the display mode according to the following equation:

HOZVAL = (Horizontal display size / Number of the valid VD data line)-1
In color mode: Horizontal display size = 3 * Number of Horizontal Pixel

In the 4-bit single scan display mode, the Number of valid VD data line should be 4. In case of 4-bit dual scan display, the Number of valid VD data line should also be 4 while in case of 8-bit single scan display mode, the Number of valid VD data line should be 8.

LINEVAL = (Vertical display size) -1: In case of single scan display type
LINEVAL = (Vertical display size / 2) -1: In case of dual scan display type

The rate of VCLK signal depends on the configuration of the CLKVAL field in the LCDCON1 register. Table 15-1 defines the relationship of VCLK and CLKVAL. The minimum value of CLKVAL is 2.

VCLK(Hz)=HCLK/(CLKVAL x 2)

The frame rate is the VFRAM signal frequency. The frame rate is closely related to the field of WLH[1:0](VLINE pulse width) WDLY[1:0] (the delay width of VCLK after VLINE pulse), HOZVAL, LINEBLANK, and LINEVAL in the LCDCON1/2/3/4 registers as well as VCLK and HCLK. Most LCD drivers need their own adequate frame rate. The frame rate is calculated as follows:

$$\text{frame\_rate(Hz)} = 1 / [\ \{\ (1/\text{VCLK}) \times (\text{HOZVAL}+1)+(1/\text{HCLK}) \times (\mathbf{A}+\mathbf{B}+(\text{LINEBLANK} \times 8)\ )\ \}\ \times\ (\ \text{LINEVAL}+1)\ ]$$

$$\boldsymbol{A} = 2^{(4+WLH)},\quad \boldsymbol{B} = 2^{(4+WDLY)}$$

Table 15-1. Relation Between VCLK and CLKVAL (STN, HCLK = 60 MHz)

CLKVAL	60 MHz/X	VCLK
2	60 MHz/4	15.0 MHz
3	60 MHz/6	10.0 MHz
:	:	:
1023	60 MHz/2046	29.3 kHz

VIDEO OPERATION

The S3C2410A LCD controller supports 8-bit color mode (256 color mode), 12-bit color mode (4096 color mode), 4 level gray scale mode, 16 level gray scale mode as well as the monochrome mode. For the gray or color mode, it is required to implement the shades of gray level or color according to time-based dithering algorithm and Frame Rate Control (FRC) method. The selection can be made following a programmable lockup table, which will be explained later. The monochrome mode bypasses these modules (FRC and lookup table) and basically serializes the data in FIFOH (and FIFOL if a dual scan display type is used) into 4-bit (or 8-bit if a 4-bit dual scan or 8-bit single scan display type is used) streams by shifting the video data to the LCD driver.

The following sections describe the operation on the gray and color mode in terms of the lookup table and FRC.

Lookup Table

The S3C2410A can support the lookup table for various selection of color or gray level mapping, ensuring flexible operation for users. The lookup table is the palette which allows the selection on the level of color or gray (Selection on 4-gray levels among 16 gray levels in case of 4 gray mode, selection on 8 red levels among 16 levels, 8 green levels among 16 levels and 4 blue levels among 16 levels in case of 256 color mode). In other words, users can select 4 gray levels among 16 gray levels by using the lookup table in the 4 gray level mode. The gray levels cannot be selected in the 16 gray level mode; all 16 gray levels must be chosen among the possible 16 gray levels. In case of 256 color mode, 3 bits are allocated for red, 3 bits for green and 2 bits for blue. The 256 colors mean that the colors are formed from the combination of 8 red, 8 green and 4 blue levels (8x8x4 = 256). In the color mode, the lookup table can be used for suitable selections. Eight red levels can be selected among 16 possible red levels, 8 green levels among 16 green levels, and 4 blue levels among 16 blue levels. In case of 4096 color mode, there is no selection as in the 256 color mode.

Gray Mode Operation

The S3C2410A LCD controller supports two gray modes: 2-bit per pixel gray (4 level gray scale) and 4-bit per pixel gray (16 level gray scale). The 2-bit per pixel gray mode uses a lookup table (BLUELUT), which allows selection on 4 gray levels among 16 possible gray levels. The 2-bit per pixel gray lookup table uses the BULEVAL[15:0] in Blue Lookup Table (BLUELUT) register as same as blue lookup table in color mode. The gray level 0 will be denoted by BLUEVAL[3:0] value. If BLUEVAL[3:0] is 9, level 0 will be represented by gray level 9 among 16 gray levels. If BLUEVAL[3:0] is 15, level 0 will be represented by gray level 15 among 16 gray levels, and so on. Following the same method as above, level 1 will also be denoted by BLUEVAL[7:4], the level 2 by BLUEVAL[11:8], and the level 3 by BLUEVAL[15:12]. These four groups among BLUEVAL[15:0] will represent level 0, level 1, level 2, and level 3. In 16 gray levels, there is no selection as in the 16 gray levels.

256 Level Color Mode Operation

The S3C2410A LCD controller can support an 8-bit per pixel 256 color display mode. The color display mode can generate 256 levels of color using the dithering algorithm and FRC. The 8-bit per pixel are encoded into 3-bits for red, 3-bits for green, and 2-bits for blue. The color display mode uses separate lookup tables for red, green, and blue. Each lookup table uses the REDVAL[31:0] of REDLUT register, GREENVAL[31:0] of GREENLUT register, and BLUEVAL[15:0] of BLUELUT register as the programmable lookup table entries.

Similar to the gray level display, 8 group or field of 4 bits in the REDLUR register, i.e., REDVAL[31:28], REDLUT[27:24], REDLUT[23:20], REDLUT[19:16], REDLUT[15:12], REDLUT[11:8], REDLUT[7:4], and REDLUT[3:0], are assigned to each red level. The possible combination of 4 bits (each field) is 16, and each red level should be assigned to one level among possible 16 cases. In other words, the user can select the suitable red level by using this type of lookup table. For green color, the GREENVAL[31:0] of the GREENLUT register is assigned as the lookup table, as was done in the case of red color. Similarly, the BLUEVAL[15:0] of the BLUELUT register is also assigned as a lookup table. For blue color, 2 bits are allocated for 4 blue levels, different from the 8 red or green levels.

4096 Level Color Mode Operation

The S3C2410A LCD controller can support a 12-bit per pixel 4096 color display mode. The color display mode can generate 4096 levels of color using the dithering algorithm and FRC. The 12-bit per pixel are encoded into 4-bit for red, 4-bit for green, and 4-bit for blue. The 4096 color display mode does not use lookup tables.

DITHERING AND FRAME RATE CONTROL

For STN LCD displays (except monochrome), video data must be processed by a dithering algorithm. The DITHFRC block has two functions, such as Time-based Dithering Algorithm for reducing flicker and Frame Rate Control (FRC) for displaying gray and color level on the STN panel. The main principle of gray and color level display on the STN panel based on FRC is described. For example, to display the third gray (3/16) level from a total of 16 levels, the 3 times pixel should be on and 13 times pixel off. In other words, 3 frames should be selected among the 16 frames, of which 3 frames should have a pixel-on on a specific pixel while the remaining 13 frames should have a pixel-off on a specific pixel. These 16 frames should be displayed periodically. This is basic principle on how to display the gray level on the screen, so-called gray level display by FRC. The actual example is shown in Table 15-2. To represent the 14^{th} gray level in the table, we should have a 6/7 duty cycle, which mean that there are 6 times pixel-on and one time pixel-off. The other cases for all gray levels are also shown in Table 15-2.

In the STN LCD display, we should be reminded of one item, i.e., Flicker Noise due to the simultaneous pixel-on and -off on adjacent frames. For example, if all pixels on first frame are turned on and all pixels on next frame are turned off, the Flicker Noise will be maximized. To reduce the Flicker Noise on the screen, the average probability of pixel-on and -off between frames should be the same. In order to realize this, the Time-based Dithering Algorithm, which varies the pattern of adjacent pixels on every frame, should be used. This is explained in detail. For the 16 gray level, FRC should have the following relationship between gray level and FRC. The 15^{th} gray level should always have pixel-on, and the 14^{th} gray level should have 6 times pixel-on and one times pixel-off, and the 13^{th} gray level should have 4 times pixel-on and one times pixel-off, ,,,,,,,, , and the 0^{th} gray level should always have pixel-off as shown in Table 15-2.

Table 15-2. Dither Duty Cycle Examples

Pre-Dithered Data (gray level number)	Duty Cycle	Pre-Dithered Data (gray level number)	Duty Cycle
15	1	7	1/2
14	6/7	6	3/7
13	4/5	5	2/5
12	3/4	4	1/3
11	5/7	3	1/4
10	2/3	2	1/5
9	3/5	1	1/7
8	4/7	0	0

Display Types

The LCD controller supports 3 types of LCD drivers: 4-bit dual scan, 4-bit single scan, and 8-bit single scan display mode. Figure 15-2 shows these 3 different display types for monochrome displays, and Figure 15-3 show these 3 different display types for color displays.

4-bit Dual Scan Display Type

A 4-bit dual scan display uses 8 parallel data lines to shift data to both the upper and lower halves of the display at the same time. The 4 bits of data in the 8 parallel data lines are shifted to the upper half and 4 bits of data is shifted to the lower half, as shown in Figure 15-2. The end of frame is reached when each half of the display has been shifted and transferred. The 8 pins (VD[7:0]) for the LCD output from the LCD controller can be directly connected to the LCD driver.

4-bit Single Scan Display Type

A 4-bit single scan display uses 4 parallel data lines to shift data to successive single horizontal lines of the display at a time, until the entire frame has been shifted and transferred. The 4 pins (VD[3:0]) for the LCD output from the LCD controller can be directly connected to the LCD driver, and the 4 pins (VD[7:4]) for the LCD output are not used.

8-bit Single Scan Display Type

An 8-bit single scan display uses 8 parallel data lines to shift data to successive single horizontal lines of the display at a time, until the entire frame has been shifted and transferred. The 8 pins (VD[7:0]) for the LCD output from the LCD controller can be directly connected to the LCD driver.

256 Color Displays

Color displays require 3 bits (Red, Green, and Blue) of image data per pixel, and so the number of horizontal shift registers for each horizontal line corresponds to three times the number of pixels of one horizontal line. resulting in a horizontal shift register of length 3 times the number of pixels per horizontal line This RGB is shifted to the LCD driver as consecutive bits via the parallel data lines. Figure 15-3 shows the RGB and order of the pixels in the parallel data lines for the 3 types of color displays.

4096 Color Displays

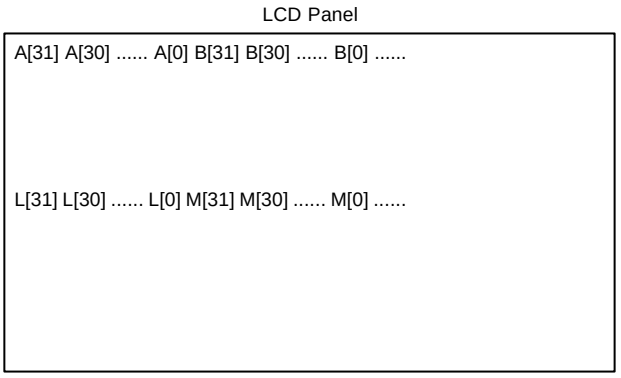
Color displays require 3 bits (Red, Green, and Blue) of image data per pixel, and so the number of horizontal shift registers for each horizontal line corresponds to three times the number of pixels of one horizontal line. This RGB is shifted to the LCD driver as consecutive bits via the parallel data lines. This RGB order is determined by the sequence of video data in video buffers.

MEMORY DATA FORMAT (STN, BSWP=0)

Mono 4-bit Dual Scan Display:

Video Buffer Memory:

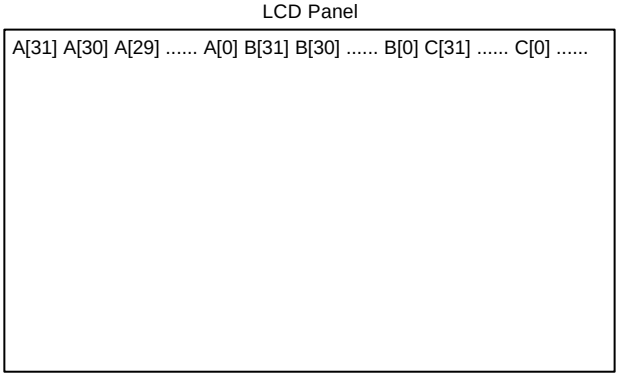
Address	Data
0000H	A[31:0]
0004H	B[31:0]
	•
	•
	•
1000H	L[31:0]
1004H	M[31:0]
	•
	•
	•



Mono 4-bit Single Scan Display & 8-bit Single Scan Display:

Video Buffer Memory:

Address	Data
0000H	A[31:0]
0004H	B[31:0]
0008H	C[31:0]
	•
	•
	•



MEMORY DATA FORMAT (STN, BSWP = 0) (Continued)

In 4-level gray mode, 2 bits of video data correspond to 1 pixel.

In 16-level gray mode, 4 bits of video data correspond to 1 pixel.

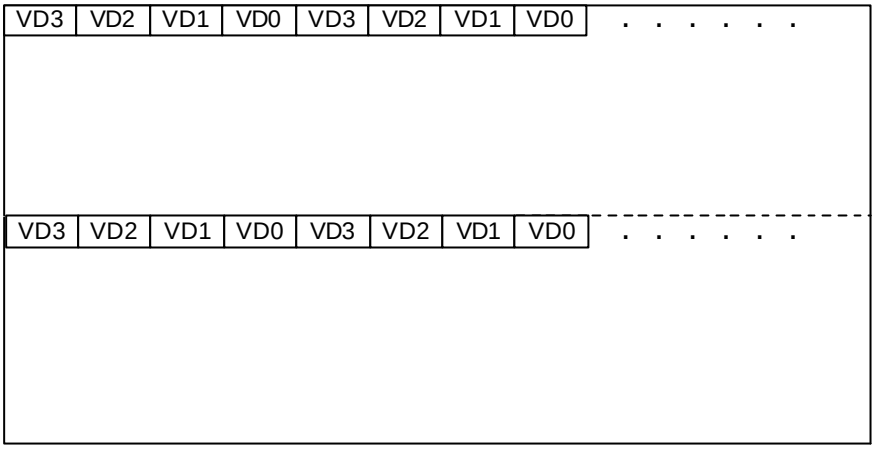
In 256 level color mode, 8 bits (3 bits of red, 3 bits of green, and 2 bits of blue) of video data correspond to 1 pixel. The color data format in a byte is as follows:

Bit [7:5]	Bit [4:2]	Bit[1:0]
Red	Green	Blue

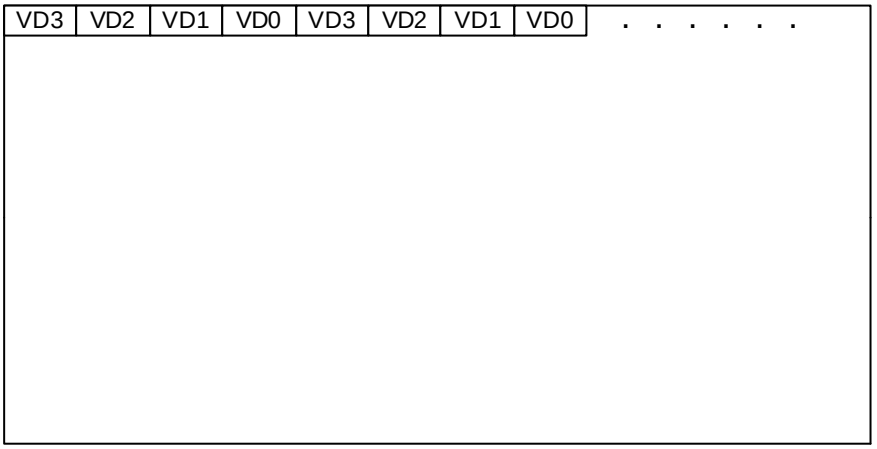
In 4096 level color mode, 12 bits (4 bits of red, 4 bits of green, 4 bits of blue) of video data correspond to 1 pixel. The following table shows color data format in words: (Video data must reside at 3 word boundaries (8 pixel), as follows)

RGB Order

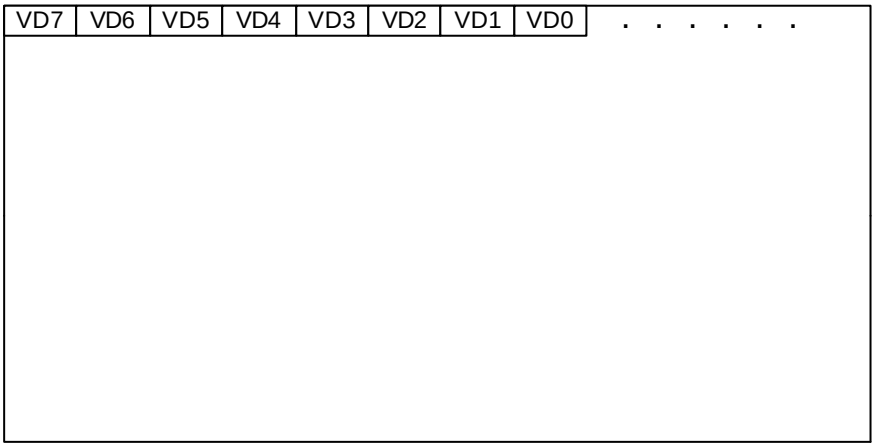
DATA	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
Word #1	Red(1)	Green(1)	Blue(1)	Red(2)	Green(2)	Blue(2)	Red(3)	Green(3)
Word #2	Blue(3)	Red(4)	Green(4)	Blue(4)	Red(5)	Green(5)	Blue(5)	Red(6)
Word #3	Green(6)	Blue(6)	Red(7)	Green(7)	Blue(7)	Red(8)	Green(8)	Blue(8)



4-bit Dual Scan Display



4-bit Single Scan Display



8-bit Single Scan Display

Figure 15-2. Monochrome Display Types (STN)

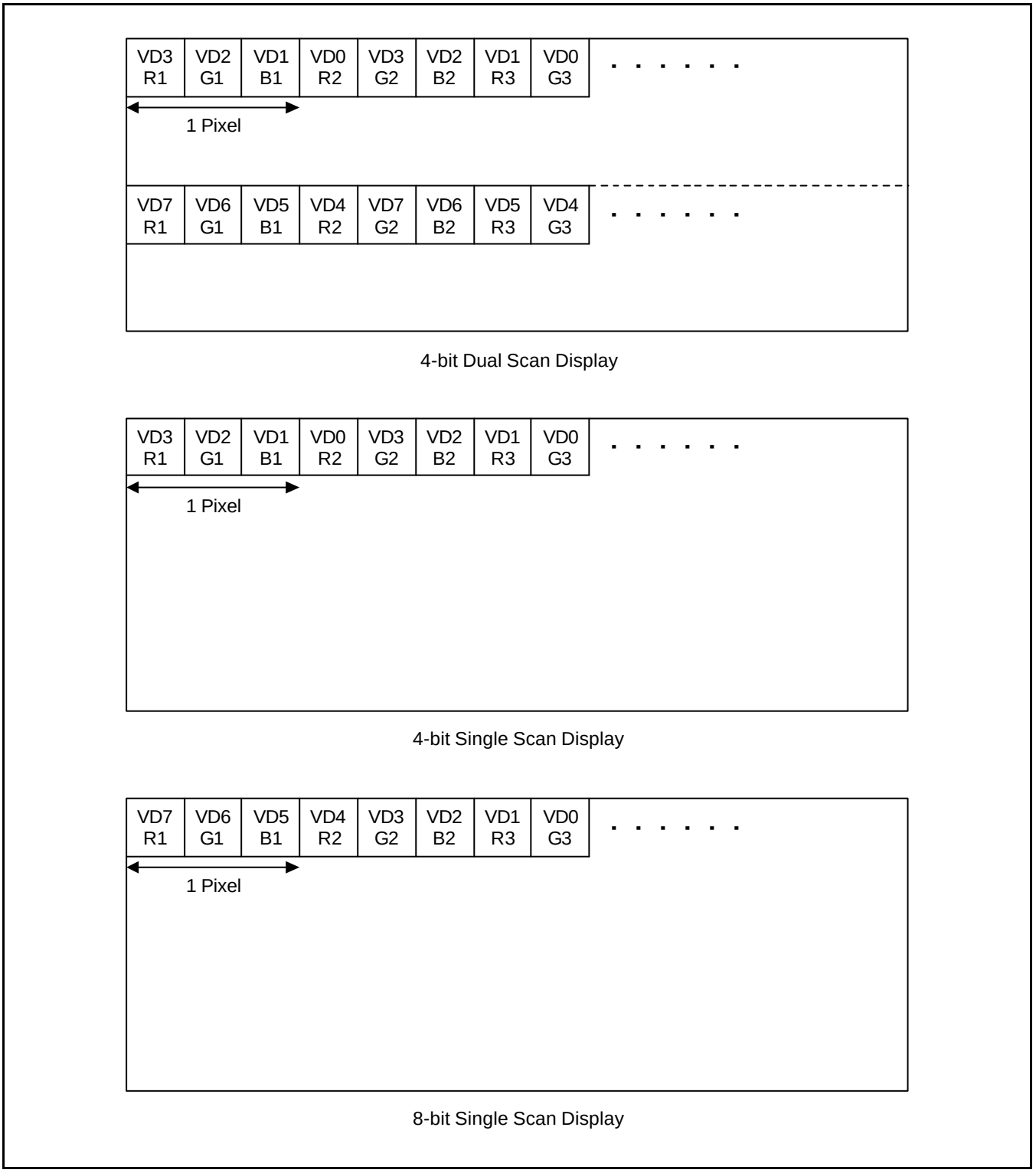


Figure 15-3. Color Display Types (STN)

Timing Requirements

Image data should be transferred from the memory to the LCD driver using the VD[7:0] signal. VCLK signal is used to clock the data into the LCD driver's shift register. After each horizontal line of data has been shifted into the LCD driver's shift register, the VLINE signal is asserted to display the line on the panel.

The VM signal provides an AC signal for the display. The LCD uses the signal to alternate the polarity of the row and column voltages, which are used to turn the pixels on and off, because the LCD plasma tends to deteriorate whenever subjected to a DC voltage. It can be configured to toggle on every frame or to toggle every programmable number of VLINE signals.

Figure 15-4 shows the timing requirements for the LCD driver interface.

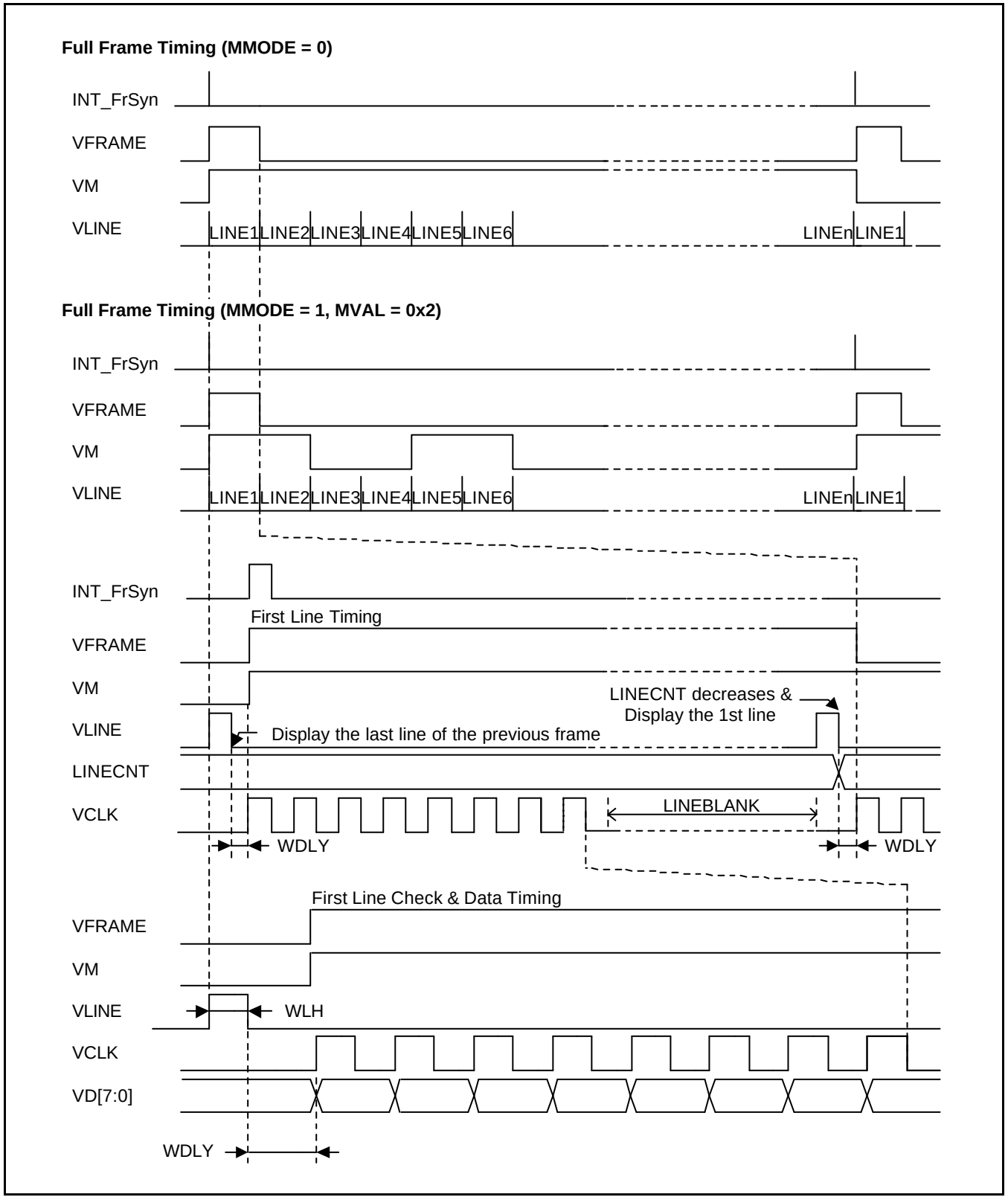


Figure 15-4. 8-bit Single Scan Display Type STN LCD Timing

TFT LCD CONTROLLER OPERATION

The TIMEGEN generates the control signals for LCD driver, such as VSYNC, HSYNC, VCLK, VDEN, and LEND signal. These control signals are highly related with the configurations on the LCDCON1/2/3/4/5 registers in the REGBANK. Base on these programmable configurations on the LCD control registers in the REGBANK, the TIMEGEN can generate the programmable control signals suitable for the support of many different types of LCD drivers.

The VSYNC signal is asserted to cause the LCD's line pointer to start over at the top of the display.

The VSYNC and HSYNC pulse generation depends on the configurations of both the HOZVAL field and the LINEVAL field in the LCDCON2/3 registers. The HOZVAL and LINEVAL can be determined by the size of the LCD panel according to the following equations:

HOZVAL = (Horizontal display size) -1
LINEVAL = (Vertical display size) -1

The rate of VCLK signal depends on the CLKVAL field in the LCDCON1 register. Table 15-3 defines the relationship of VCLK and CLKVAL. The minimum value of CLKVAL is 0.

VCLK(Hz)=HCLK/[(CLKVAL+1)x2]

The frame rate is VSYNC signal frequency. The frame rate is related with the field of VSYNC, VBPD, VFPD, LINEVAL, HSYNC, HBPD, HFPD, HOZVAL, and CLKVAL in LCDCON1 and LCDCON2/3/4 registers. Most LCD drivers need their own adequate frame rate. The frame rate is calculated as follows:

$$\text{Frame Rate} = 1/ [\{ (VSPW+1) + (VBPD+1) + (LIINEVAL + 1) + (VFPD+1) \} \times \{(HSPW+1) + (HBPD +1) + (HFPD+1) + (HOZVAL + 1) \} \times \{ 2 \times (CLKVAL+1) / (HCLK) \}]$$

Table 15-3. Relation Between VCLK and CLKVAL (TFT, HCLK = 60 MHz)

CLKVAL	60 MHz/X	VCLK
1	60 MHz/4	15.0 MHz
2	60 MHz/6	10.0 MHz
:	:	:
1023	60 MHz/2048	30.0 kHz

VIDEO OPERATION

The TFT LCD controller within the S3C2410A supports 1, 2, 4 or 8 bpp (bit per pixel) palettized color displays and 16 or 24 bpp non-palettized true-color displays.

256 Color Palette

The S3C2410A can support the 256 color palette for various selection of color mapping, providing flexible operation for users.

MEMORY DATA FORMAT (TFT)

This section includes some examples of each display mode.

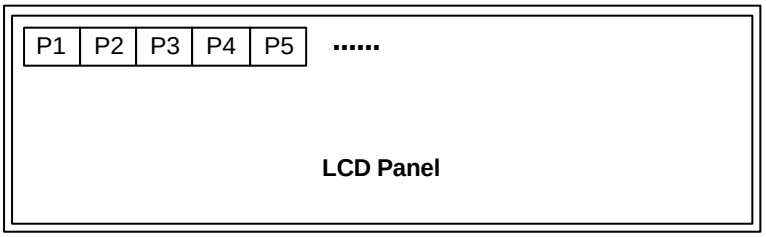
24BPP Display

(BSWP = 0, HWSWP = 0, BPP24BL = 0)

	D[31:24]	D[23:0]
000H	Dummy Bit	P1
004H	Dummy Bit	P2
008H	Dummy Bit	P3
…		

(BSWP = 0, HWSWP = 0, BPP24BL = 1)

	D[31:8]	D[7:0]
000H	P1	Dummy Bit
004H	P2	Dummy Bit
008H	P3	Dummy Bit
…		



VD Pin Descriptions at 24BPP

VD	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RED	7	6	5	4	3	2	1	0																
GREEN									7	6	5	4	3	2	1	0								
BLUE																	7	6	5	4	3	2	1	0

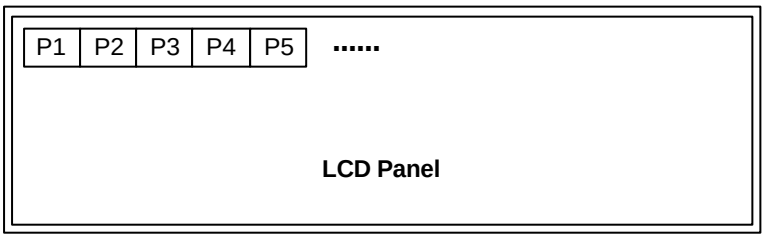
16BPP Display

(BSWP = 0, HWSWP = 0)

	D[31:16]	D[15:0]
000H	P1	P2
004H	P3	P4
008H	P5	P6
…		

(BSWP = 0, HWSWP = 1)

	D[31:16]	D[15:0]
000H	P2	P1
004H	P4	P3
008H	P6	P5
…		



VD Pin Connections at 16BPP

(5:6:5)

VD	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RED	4	3	2	1	0	NC									NC							NC		
GREEN									5	4	3	2	1	0										
BLUE																	4	3	2	1	0			

(5:5:5:I)

VD	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RED	4	3	2	1	0	I	NC								NC								NC	
GREEN									4	3	2	1	0	I										
BLUE																	4	3	2	1	0	I		

NOTE: The unused VD pins can be used as GPIO.

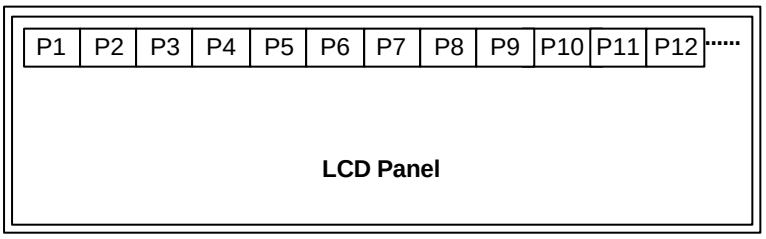
8BPP Display

(BSWP = 0, HWSWP = 0)

	D[31:24]	D[23:16]	D[15:8]	D[7:0]
000H	P1	P2	P3	P4
004H	P5	P6	P7	P8
008H	P9	P10	P11	P12
...				

(BSWP = 1, HWSWP = 0)

	D[31:24]	D[23:16]	D[15:8]	D[7:0]
000H	P4	P3	P2	P1
004H	P8	P7	P6	P5
008H	P12	P11	P10	P9
...				



4BPP Display

(BSWP = 0, HWSWP = 0)

	D[31:28]	D[27:24]	D[23:20]	D[19:16]	D[15:12]	D[11:8]	D[7:4]	D[3:0]
000H	P1	P2	P3	P4	P5	P6	P7	P8
004H	P9	P10	P11	P12	P13	P14	P15	P16
008H	P17	P18	P19	P20	P21	P22	P23	P24
…								

(BSWP = 1, HWSWP = 0)

	D[31:28]	D[27:24]	D[23:20]	D[19:16]	D[15:12]	D[11:8]	D[7:4]	D[3:0]
000H	P7	P8	P5	P6	P3	P4	P1	P2
004H	P15	P16	P13	P14	P11	P12	P9	P10
008H	P23	P24	P21	P22	P19	P20	P17	P18
…								

2BPP Display

(BSWP = 0, HWSWP = 0)

D	[31:30]	[29:28]	[27:26]	[25:24]	[23:22]	[21:20]	[19:18]	[17:16]
000H	P1	P2	P3	P4	P5	P6	P7	P8
004H	P17	P18	P19	P20	P21	P22	P23	P24
008H	P33	P34	P35	P36	P37	P38	P39	P40
…								

D	[15:14]	[13:12]	[11:10]	[9:8]	[7:6]	[5:4]	[3:2]	[1:0]
000H	P9	P10	P11	P12	P13	P14	P15	P16
004H	P25	P26	P27	P28	P29	P30	P31	P32
008H	P41	P42	P43	P44	P45	P46	P47	P48
…								

256 PALETTE USAGE (TFT)

Palette Configuration and Format Control

The S3C2410A provides 256 color palette for TFT LCD Control.
The user can select 256 colors from the 64K colors in these two formats.
The 256 color palette consists of the 256 (depth) × 16-bit SPSRAM. The palette supports 5:6:5 (R:G:B) format and 5:5:5:1(R:G:B:I) format.
When the user uses 5:5:5:1 format, the intensity data(I) is used as a common LSB bit of each RGB data. So, 5:5:5:1 format is the same as R(5+I):G(5+I):B(5+I) format.
In 5:5:5:1 format, for example, the user can write the palette as in Table 15-5 and then connect VD pin to TFT LCD panel(R(5+I)=VD[23:19]+VD[18], VD[10] or VD[2], G(5+I)=VD[15:11]+ VD[18], VD[10] or VD[2], B(5+I)=VD[7:3]+ VD[18], VD[10] or VD[2].), and set FRM565 of LCDCON5 register to 0.

Table 15-4. 5:6:5 Format

INDEX\Bit Pos.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address
00H	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	0X4D000400 (note1)
01H	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	0X4D000404
…….																	…….
FFH	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	0X4D0007FC
Number of VD	23	22	21	20	19	15	14	13	12	11	10	7	6	5	4	3	

Table 15-5. 5:5:5:1 Format

INDEX\Bit Pos.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address
00H	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	I	0X4D000400
01H	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	I	0X4D000404
…….																	…….
FFH	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	I	0X4D0007FC
Number of VD	23	22	21	20	19	15	14	13	12	11	7	6	5	4	3	2)	

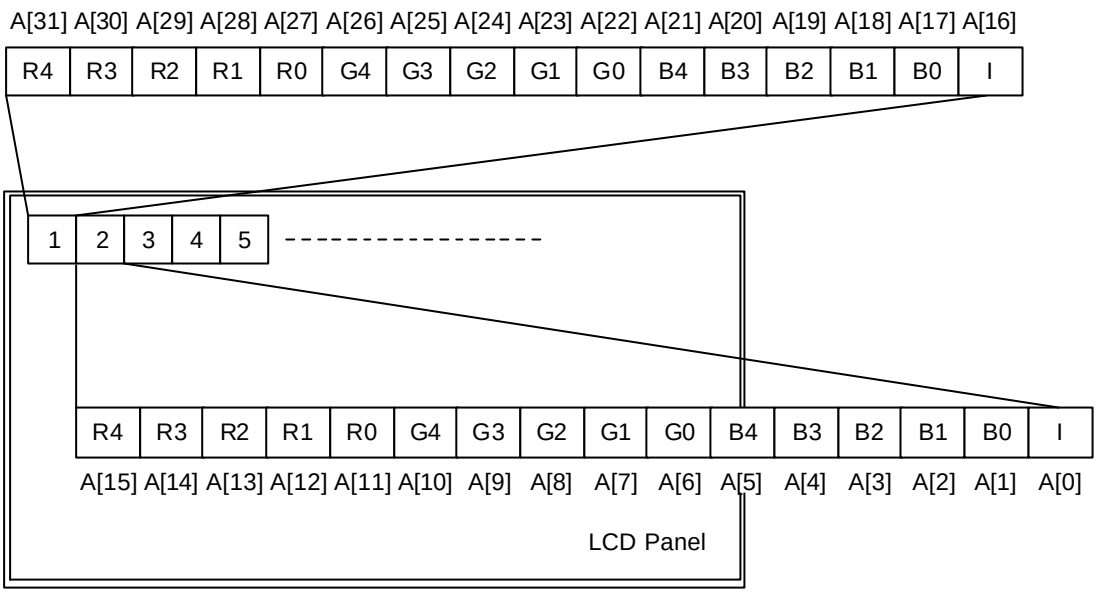
NOTES:
1. 0x4D000400 is Palette start address.
2. VD18, VD10 and VD2 have the same output value, I.
3. DATA[31:16] is invalid.

Palette Read/Write

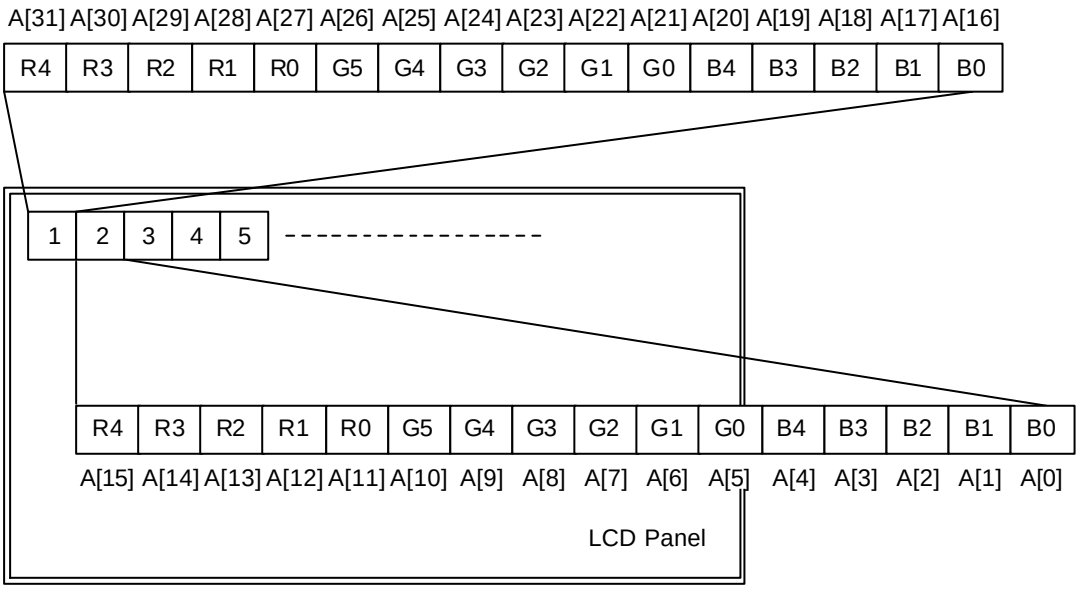
When the user performs Read/Write operation on the palette, HSTATUS and VSTATUS of LCDCON5 register must be checked, for Read/Write operation is prohibited during the ACTIVE status of HSTATUS and VSTATUS.

Temporary Palette Configuration

The S3C2410A allows the user to fill a frame with one color without complex modification to fill the one color to the frame buffer or palette. The one colored frame can be displayed by the writing a value of the color which is displayed on LCD panel to TPALVAL of TPAL register and enable TPALEN.



16BPP 5:5:5+1 Format (Non-Palette)



16BPP 5:6:5 Format (Non-Palette)

Figure 15-5. 16BPP Display Types (TFT)

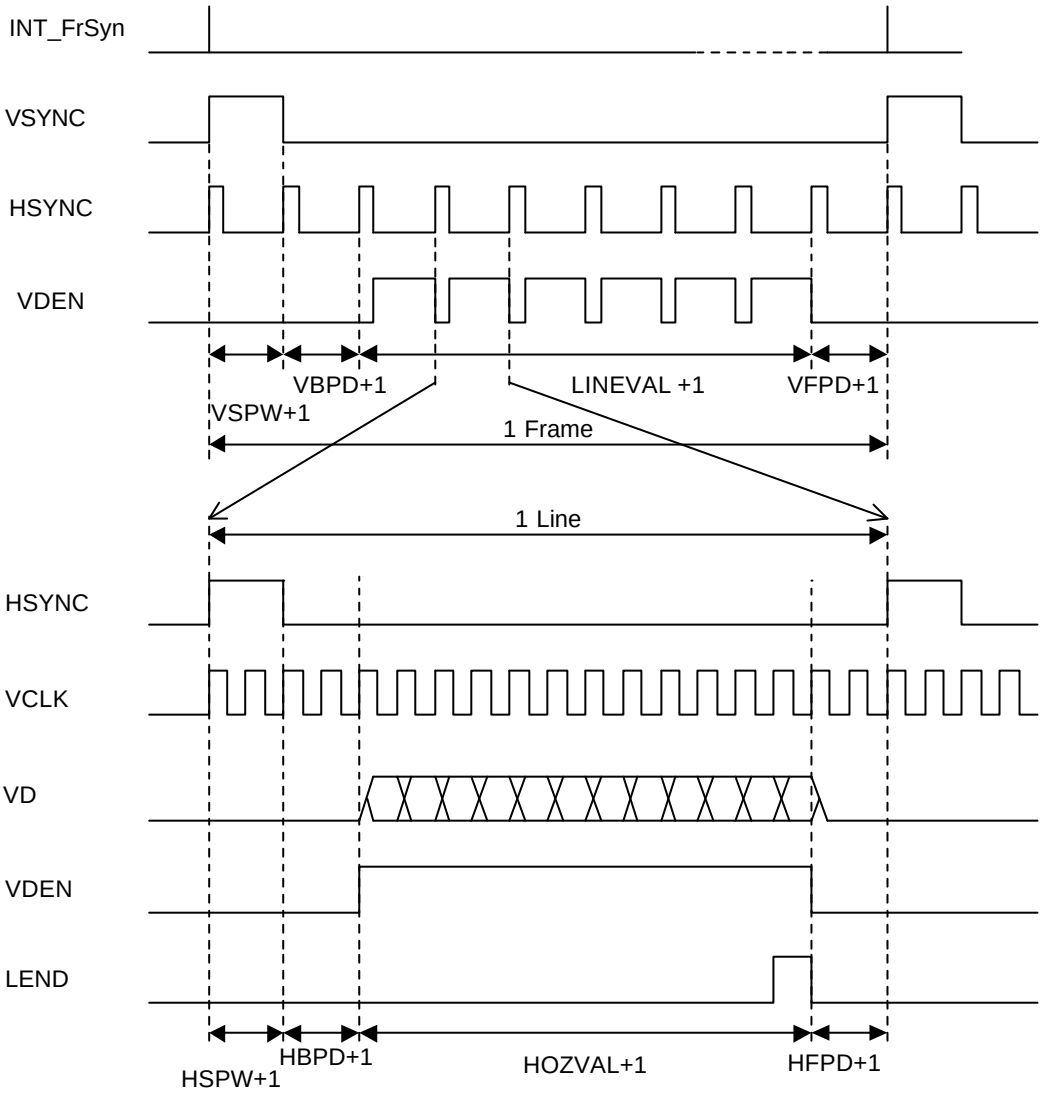


Figure 15-6. TFT LCD Timing Example

SAMSUNG TFT LCD PANEL (3.5² PORTRAIT / 256K COLOR /REFLECTIVE A-SI TFT LCD)

The S3C2410A supports SEC TFT LCD panel (SAMSUNG 3.5″ Portrait / 256K Color /Reflective a-Si TFT LCD).

LTS350Q1-PD1: TFT LCD panel with touch panel and front light unit
LTS350Q1-PD2: TFT LCD panel only

The S3C2410A provides timing signals as follows to use LTS350Q1-PD1 and PD2:

STH:	Horizontal Start Pulse
TP:	Source Driver Data Load Pulse
INV:	Digital Data Inversion
LCD_HCLK:	Horizontal Sampling Clock
CPV:	Vertical Shift Clock
STV:	Vertical Start Pulse
OE:	Gate On Enable
REV:	Inversion Signal
REVB:	Inversion Signal

So, LTS350Q1-PD1 and PD2 can be connected with the S3C2410A without using the additional timing control logic. But the user should additionally apply Vcom generator circuit, various voltages, INV signal and Gray scale voltage generator circuit, which is recommended by PRODUCT INFORMATION (SPEC) of LTS350Q1-PD1 and PD2. Detailed timing diagram is also described in PRODUCT INFORMATION (SPEC) of LTS350Q1-PD1 and PD2.

Refer to the documentation (PRODUCT INFORMATION of LTS350Q1-PD1 and PD2), which is prepared by AMLCD Technical Customer Center of Samsung Electronics Co., LTD.

Caution:

The S3C2410A has HCLK, working as the clock of AHB bus.
Accidentally, SEC TFT LCD panel (LTS350Q1-PD1 and PD2) has Horizontal Sampling Clock (HCLK).
These two HCLKs may cause a confusion. So, note that HCLK of the S3C2410A is HCLK and other HCLK of the LTS350 is LCD_HCLK.

Check that the HCLK of SEC TFT LCD panel (LTS350Q1-PD1 and PD2) is changed to LCD_HCLK.

VIRTUAL DISPLAY (TFT/STN)

The S3C2410A supports hardware horizontal or vertical scrolling. If the screen is scrolled, the fields of LCDBASEU and LCDBASEL in LCDSADDR1/2 registers need to be changed (see Figure 15-7), except the values of PAGEWIDTH and OFFSIZE.
The video buffer in which the image is stored should be larger than the LCD panel screen in size.

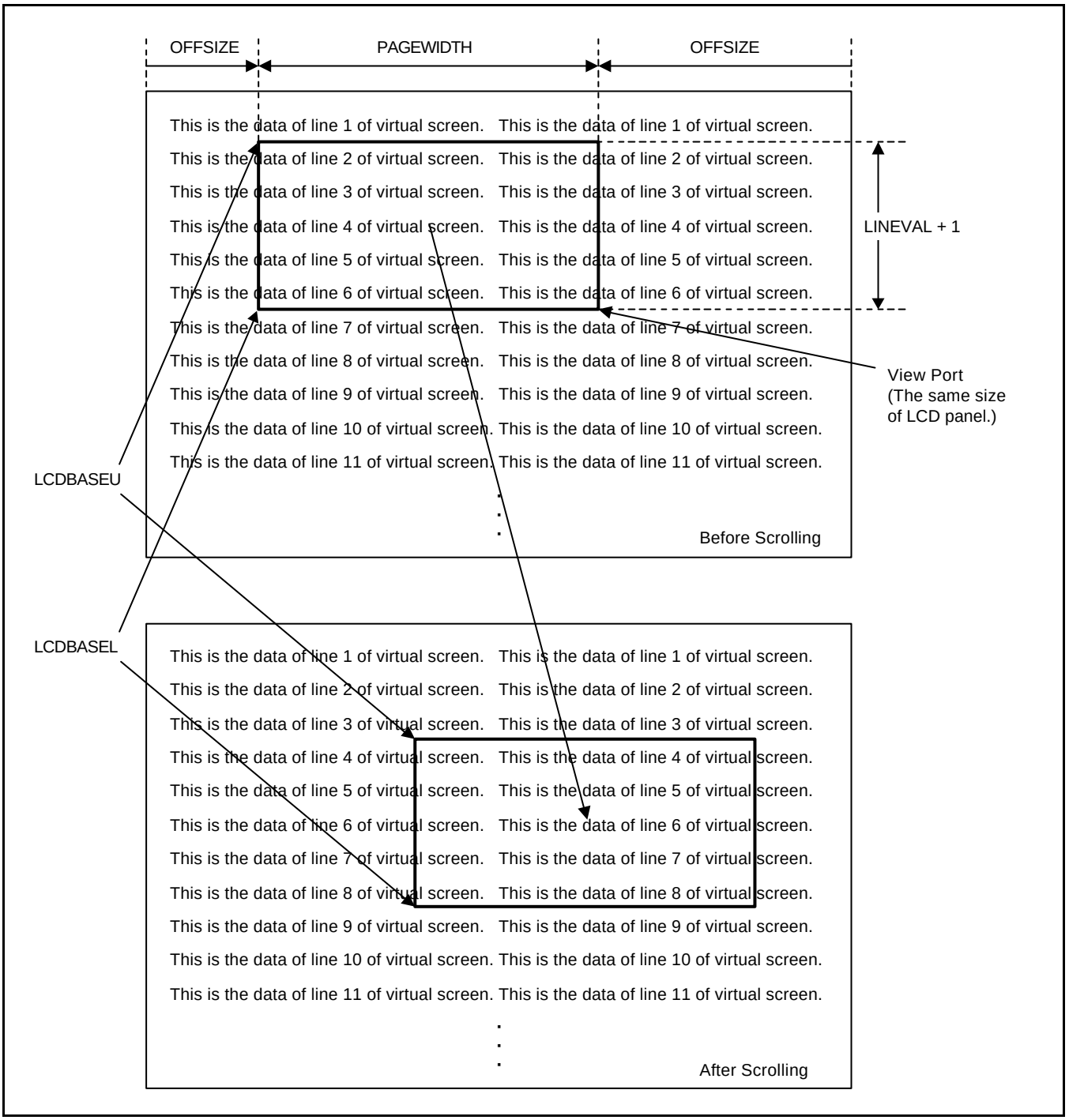


Figure 15-7. Example of Scrolling in Virtual Display (Single Scan)

LCD POWER ENABLE (STN/TFT)

The S3C2410A provides Power enable (PWREN) function. When PWREN is set to make PWREN signal enabled, the output value of LCD_PWREN pin is controlled by ENVID. In other words, If LCD_PWREN pin is connected to the power on/off control pin of the LCD panel, the power of LCD panel is controlled by the setting of ENVID automatically. The S3C2410A also supports INVPWREN bit to invert polarity of the PWREN signal.

This function is available only when LCD panel has its own power on/off control port and when port is connected to LCD_PWREN pin.

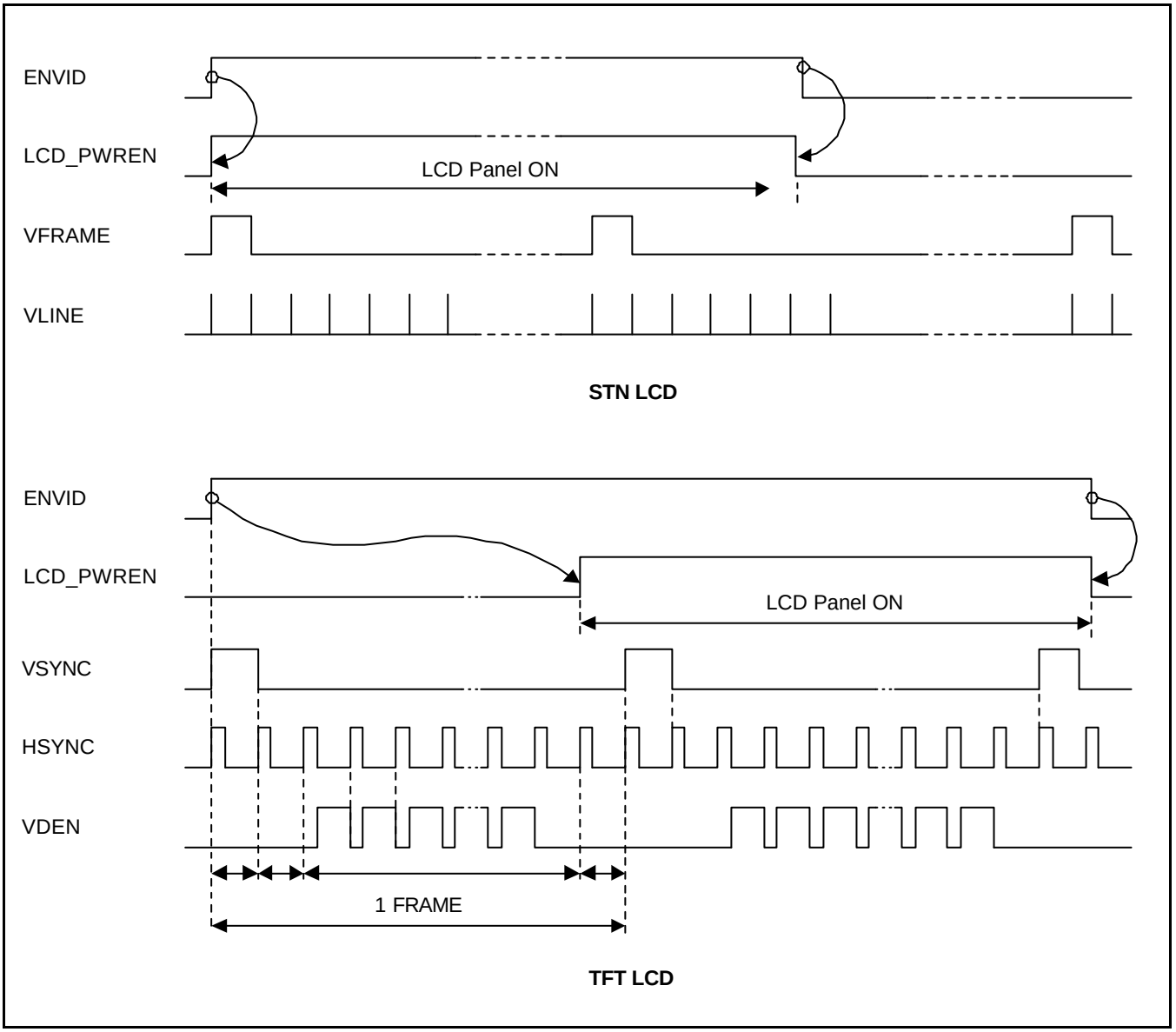


Figure 15-8. Example of PWREN Function (PWREN = 1, INVPWREN = 0)

LCD CONTROLLER SPECIAL REGISTERS

LCD Control 1 Register

Register	Address	R/W	Description	Reset Value
LCDCON1	0X4D000000	R/W	LCD control 1 register	0x00000000

LCDCON1	Bit	Description	Initial State
LINECNT (read only)	[27:18]	Provide the status of the line counter. Down count from LINEVAL to 0	0000000000
CLKVAL	[17:8]	Determine the rates of VCLK and CLKVAL[9:0]. **STN**: VCLK = HCLK / (CLKVAL × 2) (CLKVAL ≥ 2) **TFT**: VCLK = HCLK / [(CLKVAL+1) × 2] (CLKVAL ≥ 0)	0000000000
MMODE	[7]	Determine the toggle rate of the VM. 0 = Each Frame, 1 = The rate defined by the MVAL	0
PNRMODE	[6:5]	Select the display mode. 00 = 4-bit dual scan display mode (STN) 01 = 4-bit single scan display mode (STN) 10 = 8-bit single scan display mode (STN) 11 = TFT LCD panel	00
BPPMODE	[4:1]	Select the BPP (Bits Per Pixel) mode. 0000 = 1 bpp for STN, Monochrome mode 0001 = 2 bpp for STN, 4-level gray mode 0010 = 4 bpp for STN, 16-level gray mode 0011 = 8 bpp for STN, color mode 0100 = 12 bpp for STN, color mode 1000 = 1 bpp for TFT 1001 = 2 bpp for TFT 1010 = 4 bpp for TFT 1011 = 8 bpp for TFT 1100 = 16 bpp for TFT 1101 = 24 bpp for TFT	0000
ENVID	[0]	LCD video output and the logic enable/disable. 0 = Disable the video output and the LCD control signal. 1 = Enable the video output and the LCD control signal.	0

LCD Control 2 Register

Register	Address	R/W	Description	Reset Value
LCDCON2	0X4D000004	R/W	LCD control 2 register	0x00000000

LCDCON2	Bit	Description	Initial State
VBPD	[31:24]	**TFT**: Vertical back porch is the number of inactive lines at the start of a frame, after vertical synchronization period. **STN**: These bits should be set to zero on STN LCD.	0x00
LINEVAL	[23:14]	**TFT/STN**: These bits determine the vertical size of LCD panel.	0000000000
VFPD	[13:6]	**TFT**: Vertical front porch is the number of inactive lines at the end of a frame, before vertical synchronization period. **STN**: These bits should be set to zero on STN LCD.	00000000
VSPW	[5:0]	**TFT**: Vertical sync pulse width determines the VSYNC pulse's high level width by counting the number of inactive lines. **STN**: These bits should be set to zero on STN LCD.	000000

LCD Control 3 Register

Register	Address	R/W	Description	Reset Value
LCDCON3	0X4D000008	R/W	LCD control 3 register	0x00000000

LCDCON3	Bit	Description	Initial state
HBPD (TFT)	[25:19]	**TFT**: Horizontal back porch is the number of VCLK periods between the falling edge of HSYNC and the start of active data.	0000000
WDLY (STN)		**STN**: WDLY[1:0] bits determine the delay between VLINE and VCLK by counting the number of the HCLK. WDLY[7:2] are reserved. 00 = 16 HCLK, 01 = 32 HCLK, 10 = 48 HCLK, 11 = 64 HCLK	
HOZVAL	[18:8]	**TFT/STN**: These bits determine the horizontal size of LCD panel. HOZVAL has to be determined to meet the condition that total bytes of 1 line are 4n bytes. If the x size of LCD is 120 dot in mono mode, x=120 cannot be supported because 1 line consists of 15 bytes. Instead, x=128 in mono mode can be supported because 1 line is composed of 16 bytes (4n). LCD panel driver will discard the additional 8 dot.	00000000000
HFPD (TFT)	[7:0]	**TFT**: Horizontal front porch is the number of VCLK periods between the end of active data and the rising edge of HSYNC.	0X00
LINEBLANK (STN)		**STN**: These bits indicate the blank time in one horizontal line duration time. These bits adjust the rate of the VLINE finely. The unit of LINEBLANK is HCLK X 8. Ex) If the value of LINEBLANK is 10, the blank time is inserted to VCLK during 80 HCLK.	

Programming NOTE

: In case of STN LCD, (LINEBLANK + WLH + WDLY) value should be bigger than (14+12xTmax).

$$(\text{LINEBLANK} + \text{WLH} + \text{WDLY}) = (14 + 8 \times T_{max1} + 4 \times T_{max2}) = (14 + 12 \times T_{max})$$

LEGEND:

(1) 14: SDRAM Auto refresh bus acquisition cycles

(2) 8x Tmax1: Cache fill cycle X the Slowest Memory access time (Ex, ROM)

(3) 4x Tmax2: 0xC~0xE address Frame memory Access time

(4) Tmax: Large one of the Tmax1 and Tmax2.

LCD Control 4 Register

Register	Address	R/W	Description	Reset Value
LCDCON4	0X4D00000C	R/W	LCD control 4 register	0x00000000

LCDCON4	Bit	Description	Initial state
MVAL	[15:8]	**STN**: These bit define the rate at which the VM signal will toggle if the MMODE bit is set to logic '1'.	0X00
HSPW(TFT)	[7:0]	**TFT**: Horizontal sync pulse width determines the HSYNC pulse's high level width by counting the number of the VCLK.	0X00
WLH(STN)		**STN**: WLH[1:0] bits determine the VLINE pulse's high level width by counting the number of the HCLK. WLH[7:2] are reserved. 00 = 16 HCLK, 01 = 32 HCLK, 10 = 48 HCLK, 11 = 64 HCLK	

LCD Control 5 Register

Register	Address	R/W	Description	Reset Value
LCDCON5	0X4D000010	R/W	LCD control 5 register	0x00000000

LCDCON5	Bit	Description	Initial state
Reserved	[31:17]	This bit is reserved and the value should be '0'.	0
VSTATUS	[16:15]	**TFT**: Vertical Status (read only). 00 = VSYNC 01 = BACK Porch 10 = ACTIVE 11 = FRONT Porch	00
HSTATUS	[14:13]	**TFT**: Horizontal Status (read only). 00 = HSYNC 01 = BACK Porch 10 = ACTIVE 11 = FRONT Porch	00
BPP24BL	[12]	**TFT**: This bit determines the order of 24 bpp video memory. 0 = LSB valid 1 = MSB Valid	0
FRM565	[11]	**TFT**: This bit selects the format of 16 bpp output video data. 0 = 5:5:5:1 Format 1 = 5:6:5 Format	0
INVVCLK	[10]	**STN/TFT**: This bit controls the polarity of the VCLK active edge. 0 = The video data is fetched at VCLK falling edge 1 = The video data is fetched at VCLK rising edge	0
INVVLINE	[9]	**STN/TFT**: This bit indicates the VLINE/HSYNC pulse polarity. 0 = Normal 1 = Inverted	0
INVVFRAME	[8]	**STN/TFT**: This bit indicates the VFRAME/VSYNC pulse polarity. 0 = Normal 1 = Inverted	0
INVVD	[7]	**STN/TFT**: This bit indicates the VD (video data) pulse polarity. 0 = Normal 1 = VD is inverted.	0

LCD Control 5 Register (Continued)

LCDCON5	Bit	Description	Initial state
INVVDEN	[6]	**TFT**: This bit indicates the VDEN signal polarity. 0 = Normal 1 = Inverted	0
INVPWREN	[5]	**STN/TFT**: This bit indicates the PWREN signal polarity. 0 = Normal 1 = Inverted	0
INVLEND	[4]	**TFT**: This bit indicates the LEND signal polarity. 0 = Normal 1 = Inverted	0
PWREN	[3]	**STN/TFT**: LCD_PWREN output signal enable/disable. 0 = Disable PWREN signal 1 = Enable PWREN signal	0
ENLEND	[2]	**TFT**: LEND output signal enable/disable. 0 = Disable LEND signal 1 = Enable LEND signal	0
BSWP	[1]	**STN/TFT**: Byte swap control bit. 0 = Swap Disable 1 = Swap Enable	0
HWSWP	[0]	**STN/TFT**: Half-Word swap control bit. 0 = Swap Disable 1 = Swap Enable	0

FRAME BUFFER START ADDRESS 1 REGISTER

Register	Address	R/W	Description	Reset Value
LCDSADDR1	0X4D000014	R/W	**STN/TFT**: Frame buffer start address 1 register	0x00000000

LCDSADDR1	Bit	Description	Initial State
LCDBANK	[29:21]	These bits indicate A[30:22] of the bank location for the video buffer in the system memory. LCDBANK value cannot be changed even when moving the view port. LCD frame buffer should be within aligned 4MB region, which ensures that LCDBANK value will not be changed when moving the view port. So, care should be taken to use the malloc() function.	0x00
LCDBASEU	[20:0]	For dual-scan LCD: These bits indicate A[21:1] of the start address of the upper address counter, which is for the upper frame memory of dual scan LCD or the frame memory of single scan LCD. For single-scan LCD: These bits indicate A[21:1] of the start address of the LCD frame buffer.	0x000000

FRAME Buffer Start Address 2 Register

Register	Address	R/W	Description	Reset Value
LCDSADDR2	0X4D000018	R/W	**STN/TFT**: Frame buffer start address 2 register	0x00000000

LCDSADDR2	Bit	Description	Initial State
LCDBASEL	[20:0]	For dual-scan LCD: These bits indicate A[21:1] of the start address of the lower address counter, which is used for the lower frame memory of dual scan LCD. For single scan LCD: These bits indicate A[21:1] of the end address of the LCD frame buffer. LCDBASEL = ((the fame end address) >>1) + 1 = LCDBASEU + (PAGEWIDTH+OFFSIZE)x(LINEVAL+1)	0x0000

NOTE: Users can change the LCDBASEU and LCDBASEL values for scrolling while the LCD controller is turned on. But, users must not change the value of the LCDBASEU and LCDBASEL registers at the end of FRAME by referring to the LINECNT field in LCDCON1 register, for the LCD FIFO fetches the next frame data prior to the change in the frame.

So, if you change the frame, the pre-fetched FIFO data will be obsolete and LCD controller will display an incorrect screen. To check the LINECNT, interrupts should be masked. If any interrupt is executed just after reading LINECNT,

the read LINECNT value may be obsolete because of the execution time of Interrupt Service Routine (ISR).

FRAME Buffer Start Address 3 Register

Register	Address	R/W	Description	Reset Value
LCDSADDR3	0X4D00001C	R/W	**STN/TFT**: Virtual screen address set	0x00000000

LCDSADDR3	Bit	Description	Initial State
OFFSIZE	[21:11]	Virtual screen offset size (the number of half words). This value defines the difference between the address of the last half word displayed on the previous LCD line and the address of the first half word to be displayed in the new LCD line.	00000000000
PAGEWIDTH	[10:0]	Virtual screen page width (the number of half words). This value defines the width of the view port in the frame.	000000000

NOTE: The values of PAGEWIDTH and OFFSIZE must be changed when ENVID bit is 0.

Example 1. LCD panel = 320*240, 16gray, single scan
Frame start address = 0x0c500000
Offset dot number = 2048 dots (512 half words)

LINEVAL = 240-1 = 0xef
PAGEWIDTH = 320*4/16 = 0x50
OFFSIZE = 512 = 0x200
LCDBANK = 0x0c500000 >> 22 = 0x31
LCDBASEU = 0x100000 >> 1 = 0x80000
LCDBASEL = 0x80000 + (0x50 + 0x200) * (0xef + 1) = 0xa2b00

Example 2. LCD panel = 320*240, 16gray, dual scan
Frame start address = 0x0c500000
Offset dot number = 2048 dots (512 half words)

LINEVAL = 120-1 = 0x77
PAGEWIDTH = 320*4/16 = 0x50
OFFSIZE = 512 = 0x200
LCDBANK = 0x0c500000 >> 22 = 0x31
LCDBASEU = 0x100000 >> 1 = 0x80000
LCDBASEL = 0x80000 + (0x50 + 0x200) * (0x77 + 1) = 0x91580

Example 3. LCD panel = 320*240, color, single scan
Frame start address = 0x0c500000
Offset dot number = 1024 dots (512 half words)

LINEVAL = 240-1 = 0xef
PAGEWIDTH = 320*8/16 = 0xa0
OFFSIZE = 512 = 0x200
LCDBANK = 0x0c500000 >> 22 = 0x31
LCDBASEU = 0x100000 >> 1 = 0x80000
LCDBASEL = 0x80000 + (0xa0 + 0x200) * (0xef + 1) = 0xa7600

RED Lookup Table Register

Register	Address	R/W	Description	Reset Value
REDLUT	0X4D000020	R/W	**STN**: Red lookup table register	0x00000000

REDLUT	Bit	Description	Initial State
REDVAL	[31:0]	These bits define which of the 16 shades will be chosen by each of the 8 possible red combinations. 000 = REDVAL[3:0], 001 = REDVAL[7:4] 010 = REDVAL[11:8], 011 = REDVAL[15:12] 100 = REDVAL[19:16], 101 = REDVAL[23:20] 110 = REDVAL[27:24], 111 = REDVAL[31:28]	0x00000000

GREEN Lookup Table Register

Register	Address	R/W	Description	Reset Value
GREENLUT	0X4D000024	R/W	**STN**: Green lookup table register	0x00000000

GREENLUT	Bit	Description	Initial State
GREENVAL	[31:0]	These bits define which of the 16 shades will be chosen by each of the 8 possible green combinations. 000 = GREENVAL[3:0], 001 = GREENVAL[7:4] 010 = GREENVAL[11:8], 011 = GREENVAL[15:12] 100 = GREENVAL[19:16], 101 = GREENVAL[23:20] 110 = GREENVAL[27:24], 111 = GREENVAL[31:28]	0x00000000

BLUE Lookup Table Register

Register	Address	R/W	Description	Reset Value
BLUELUT	0X4D000028	R/W	**STN**: Blue lookup table register	0x0000

BULELUT	Bit	Description	Initial State
BLUEVAL	[15:0]	These bits define which of the 16 shades will be chosen by each of the 4 possible blue combinations. 00 = BLUEVAL[3:0], 01 = BLUEVAL[7:4] 10 = BLUEVAL[11:8], 11 = BLUEVAL[15:12]	0x0000

NOTE: Address from **0x4D00002C** to **0x4D000048** should not be used. This area is reserved for Test mode.

Dithering Mode Register

Register	Address	R/W	Description	Reset Value
DITHMODE	0X4D00004C	R/W	**STN**: Dithering mode register. This register reset value is 0x00000 But, user can change this value to 0x12210. (Refer to a sample program source for the latest value of this register.)	0x00000

DITHMODE	Bit	Description	Initial state
DITHMODE	[18:0]	Use one of following value for your LCD: 0x00000 or 0x12210	0x00000

Temp Palette Register

Register	Address	R/W	Description	Reset Value
TPAL	0X4D000050	R/W	**TFT**: Temporary palette register. This register value will be video data at next frame.	0x00000000

TPAL	Bit	Description	Initial state
TPALEN	[24]	Temporary palette register enable bit. 0 = Disable 1 = Enable	0
TPALVAL	[23:0]	Temporary palette value register. TPALVAL[23:16] : RED TPALVAL[15:8] : GREEN TPALVAL[7:0] : BLUE	0x000000

LCD Interrupt Pending Register

Register	Address	R/W	Description	Reset Value
LCDINTPND	0X4D000054	R/W	Indicate the LCD interrupt pending register	0x0

LCDINTPND	Bit	Description	Initial state
INT_FrSyn	[1]	LCD frame synchronized interrupt pending bit. 0 = The interrupt has not been requested. 1 = The frame has asserted the interrupt request.	0
INT_FiCnt	[0]	LCD FIFO interrupt pending bit. 0 = The interrupt has not been requested. 1 = LCD FIFO interrupt is requested when LCD FIFO reaches trigger level.	0

LCD Source Pending Register

Register	Address	R/W	Description	Reset Value
LCDSRCPND	0X4D000058	R/W	Indicate the LCD interrupt source pending register	0x0

LCDSRCPND	Bit	Description	Initial state
INT_FrSyn	[1]	LCD frame synchronized interrupt source pending bit. 0 = The interrupt has not been requested. 1 = The frame has asserted the interrupt request.	0
INT_FiCnt	[0]	LCD FIFO interrupt source pending bit. 0 = The interrupt has not been requested. 1 = LCD FIFO interrupt is requested when LCD FIFO reaches trigger level.	0

LCD Interrupt Mask Register

Register	Address	R/W	Description	Reset Value
LCDINTMSK	0X4D00005C	R/W	Determine which interrupt source is masked. The masked interrupt source will not be serviced.	0x3

LCDINTMSK	Bit	Description	Initial state
FIWSEL	[2]	Determine the trigger level of LCD FIFO. 0 = 4 words 1 = 8 words	
INT_FrSyn	[1]	Mask LCD frame synchronized interrupt. 0 = The interrupt service is available. 1 = The interrupt service is masked.	1
INT_FiCnt	[0]	Mask LCD FIFO interrupt. 0 = The interrupt service is available. 1 = The interrupt service is masked.	1

LPC3600 Control Register

Register	Address	R/W	Description	Reset Value
LPCSEL	0X4D000060	R/W	This register controls the LPC3600 modes.	0x4

LPCSEL	Bit	Description	Initial state
Reserved	[2]	Reserved	1
RES_SEL	[1]	1 = 240×320	0
LPC_EN	[0]	Determine LPC3600 Enable/Disable. 0 = LPC3600 Disable 1 = LPC3600 Enable	0

Register Setting Guide (STN)

The LCD controller supports multiple screen sizes by special register setting.

The CLKVAL value determines the frequency of VCLK. This value has to be determined such that the VCLK value is greater than data transmission rate. The data transmission rate for the VD port of the LCD controller is used to determine the value of CLKVAL register.

The data transmission rate is given by the following equation:

Data transmission rate = HS × VS × FR × MV

HS: Horizontal LCD size

VS: Vertical LCD size

FR: Frame rate

MV: Mode dependent value

Table 15-6. MV Value for Each Display Mode

Mode	MV Value
Mono, 4-bit single scan display	1/4
Mono, 8-bit single scan display or 4-bit dual scan display	1/8
4 level gray, 4-bit single scan display	1/4
4 level gray, 8-bit single scan display or 4-bit dual scan display	1/8
16 level gray, 4-bit single scan display	1/4
16 level gray, 8-bit single scan display or 4-bit dual scan display	1/8
Color, 4-bit single scan display	3/4
Color, 8-bit single scan display or 4-bit dual scan display	3/8

The LCDBASEU register value is the first address value of the frame buffer. The lowest 4 bits must be eliminated for burst 4 word access. The LCDBASEL register value depends on LCD size and LCDBASEU. The LCDBASEL value is given by the following equation:

LCDBASEL = LCDBASEU + LCDBASEL offset

Example 1:

160 x 160, 4-level gray, 80 frame/sec, 4-bit single scan display, HCLK frequency is 60 MHz WLH = 1, WDLY = 1.

Data transmission rate = 160 × 160 × 80 × 1/4 = 512 kHz
CLKVAL = 58, VCLK = 517 kHz
HOZVAL = 39, LINEVAL = 159

LINEBLANK =10

LCDBASEL = LCDBASEU + 3200

NOTE: The higher the system load is, the lower the cpu performance is.

Example 2 (Virtual Screen Register):

4 -level gray, Virtual screen size = 1024 × 1024, LCD size = 320 × 240, LCDBASEU = 0×64, 4-bit dual scan.

1 halfword = 8 pixels (4-level gray),
Virtual screen 1 line = 128 halfword = 1024 pixels,
LCD 1 line = 320 pixels = 40 halfword,
OFFSIZE = 128 - 40 = 88 = 0×58,
PAGEWIDTH = 40 = 0×28

LCDBASEL = LCDBASEU + (PAGEWIDTH + OFFSIZE) × (LINEVAL +1) = 100 + (40 + 88) × 120 = 0 × 3C64

Known Problem of STN LCD

: If Enable and Disable Video output, [ENVID] of LCDCON1 Register, it is repeated over 100times,
there could be 1Word data loss in case of STN. But the probability of this symptom is very low.

=> **Workaround** : You should insert the below function in your code right before Enable Video output.

```
void Lcd_Workaround (void)
{

    int loop=7;

    rLCDCON5=rLCDCON5&~(1<<3); // Disable LCD_PWREN, If you don't use LCD_PWREN,  you can
remove this line.

    for(i=0; I<loop; I++)

  {

        rLCDCON1|=1; // ENVID=ON
        rLCDCON1 =rLCDCON1 & 0x3fffe; // ENVID Off
   }

   rLCDCON5|=(1<<3); //Enable LCD_PWREN, If you don't use LCD_PWREN, then you can remove this line.
 }
```

Gray Level Selection Guide

The S3C2410A LCD controller can generate 16 gray level using Frame Rate Control (FRC). The FRC characteristics may cause unexpected patterns in gray level. These unwanted erroneous patterns may be shown in fast response LCD or at lower frame rates.

Because the quality of LCD gray levels depends on LCD's own characteristics, the user has to select an appropriate gray level after viewing all gray levels on user's own LCD.

Select the gray level quality through the following procedures:

1. Get the latest dithering pattern register value from SAMSUNG.
2. Display 16 gray bar in LCD.
3. Change the frame rate into an optimal value.
4. Change the VM alternating period to get the best quality.
5. As viewing 16 gray bars, select a good gray level, which is displayed well on your LCD.
6. Use only the good gray levels for quality.

LCD Refresh Bus Bandwidth Calculation Guide

The S3C2410A LCD controller can support various LCD display sizes. To select a suitable size (for the flicker free LCD system application), the user have to consider the LCD refresh bus bandwidth determined by the LCD display size, bit per pixel (bpp), frame rate, memory bus width, memory type, and so on.

LCD Data Rate (Byte/s) = bpp x (Horizontal display size) x (Vertical display size) x (Frame rate) /8

LCD DMA Burst Count (Times/s) = LCD Data Rate(Byte/s) /16(Byte) ; LCD DMA using 4words(16Byte) burst

Pdma means LCD DMA access period. In other words, the value of Pdma indicates the period of four-beat burst (4-words burst) for video data fetch. So, Pdma depends on memory type and memory setting.

Eventually, LCD System Load is determined by LCD DMA Burst Count and Pdma.

LCD System Load = LCD DMA Burst Count x Pdma

Example 3:

640 × 480, 8bpp, 60 frame/sec, 16-bit data bus width, SDRAM (Trp=2HCLK / Trcd=2HCLK / CL=2HCLK) and HCLK frequency is 60 MHz

LCD Data Rate = 8 × 640 × 480 × 60 / 8 = 18.432Mbyte/s
LCD DMA Burst Count = 18.432 / 16 = 1.152M/s
Pdma = (Trp+Trcd+CL+(2 × 4)+1) × (1/60 MHz) = 0.250ms
LCD System Load = 1.152 × 250 = 0.288

System Bus Occupation Rate = (0.288/1) × 100 = 28.8%

Register Setting Guide (TFT LCD)

The CLKVAL register value determines the frequency of VCLK and frame rate.

Frame Rate = 1/ [{ (VSPW+1) + (VBPD+1) + (LIINEVAL + 1) + (VFPD+1) } × {(HSPW+1) + (HBPD +1) + (HFPD+1) + (HOZVAL + 1) } × { 2 × (CLKVAL+1) / (HCLK) }]

For applications, the system timing must be considered to avoid under-run condition of the fifo of the lcd controller caused by memory bandwidth contention.

Example 4:

TFT Resolution: 240 × 240,

VSPW = 2, VBPD = 14, LINEVAL = 239, VFPD = 4

HSPW = 25, HBPD = 15, HOZVAL = 239, HFPD = 1

CLKVAL = 5

HCLK = 60M (Hz)

The parameters below must be referenced by LCD size and driver specifications:

VSPW, VBPD, LINEVAL, VFPD, HSPW, HBPD, HOZVAL, and HFPD

If target frame rate is 60–70Hz, then CLKVAL should be 5.

So, Frame Rate = 67Hz

Known Problems

Problem : In a MDS, such as Multi-ICE, some of the LCD controller registers may be displayed incorrectly in the memory view window of the ARM debugger.

Solution : The LCD controller register will be displayed correctly unless the memory view window is used. Instead, use 'pr' command in the debugger console window.

16 ADC & TOUCH SCREEN INTERFACE

OVERVIEW

The 10-bit CMOS analog to digital converter (ADC) of the S3C2410A is a recycling typed device with 8-channel analog inputs. It converts the analog input signal into 10-bit binary digital codes at a maximum conversion rate of 500KSPS with 2.5 MHz A/D converter clock. The A/D converter operates with on-chip, sample-and-hold function and power down mode is supported.

The S3C2410A supports Touch Screen Interface, which consists of a touch screen panel, four external transistors, an external voltage source, AIN[7] and AIN[5] (see Figure 16-2).
Touch Screen Interface controls and selects control signals (nYPON, YMON, nXPON and XMON) and analog pads (AIN[7], AIN[5]) which are connected with pads of touch screen panel and the external transistor for X-position conversion and Y-position conversion.
Touch Screen Interface contains an external transistor control logic and an ADC interface logic with an interrupt generation logic.

FEATURES

— Resolution: 10-bit
— Differential Linearity Error: ± 1.0 LSB
— Integral Linearity Error: ± 2.0 LSB
— Maximum Conversion Rate: 500 KSPS
— Low Power Consumption
— Power Supply Voltage: 3.3V
— Analog Input Range: 0 ~ 3.3V
— On-chip Sample-and-hold Function
— Normal Conversion Mode
— Separate X/Y position conversion Mode
— Auto (Sequential) X/Y Position Conversion Mode
— Waiting for Interrupt Mode

ADC & TOUCH SCREEN INTERFACE OPERATION

BLOCK DIAGRAM

Figure 16-1 shows the functional block diagram of the S3C2410A A/D converter and Touch Screen Interface. Note that the A/D converter is a recycling type.
A pull-up resister is attached to AIN[7] on VDDA_ADC. So, XP pad of the touch screen panel should be connected with AIN[7] of the S3C2410A and YP pad of the touch screen panel should be connected with AIN[5].

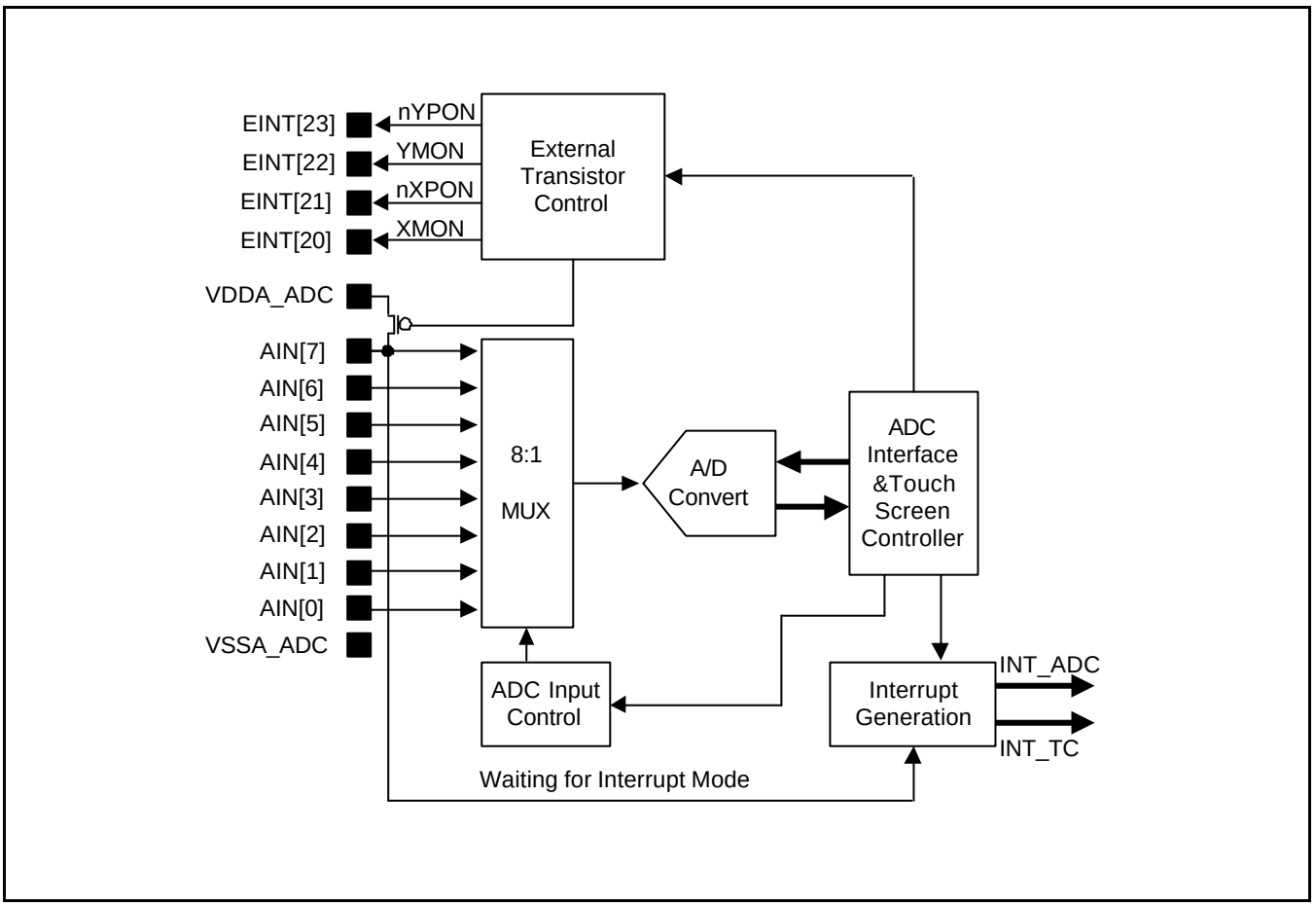


Figure 16-1. ADC and Touch Screen Interface Block Diagram

EXAMPLE FOR TOUCH SCREEN

In this example, AIN[7] is connected with XP and AIN[5]is connected with YP pad of the touch screen panel. To control pads of the touch screen panel (XP, XM, YP and YM), four external transistors are applied and control signals including nYPON, YMON, nXPON and XMON are connected with four external transistors.

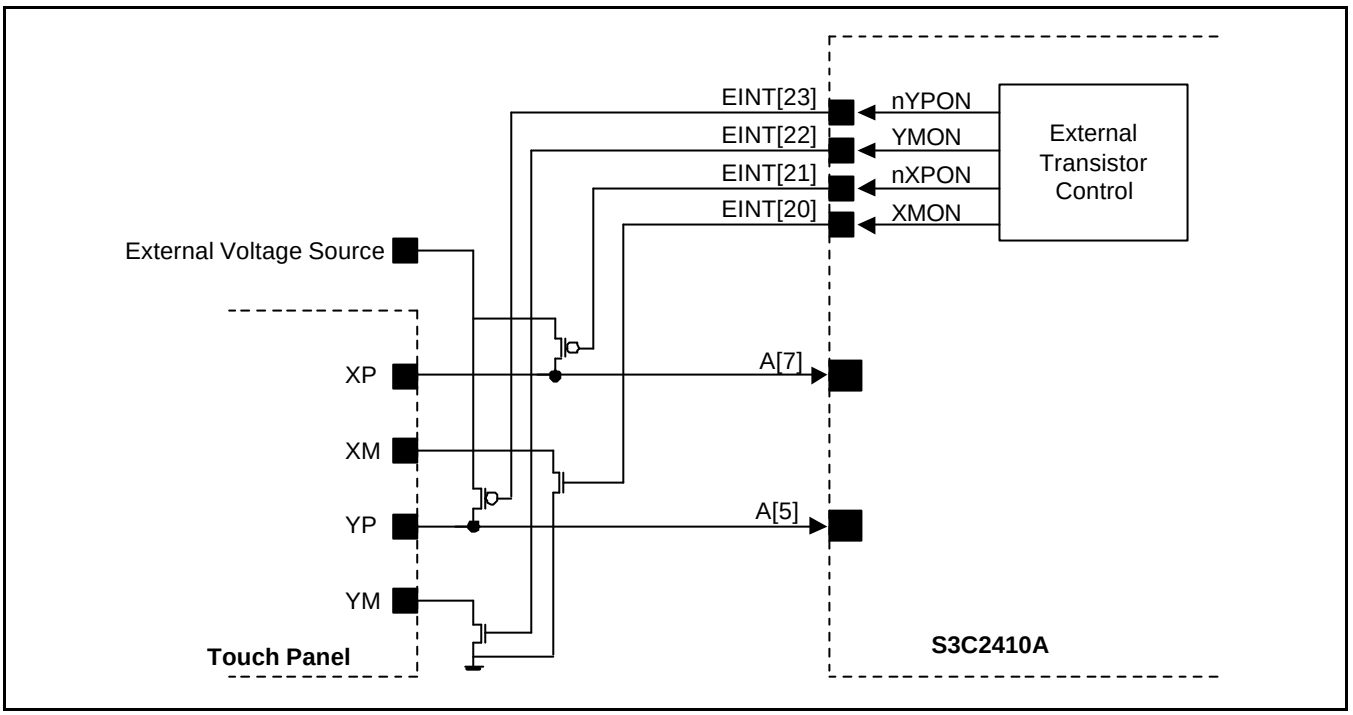


Figure 16-2. Example of ADC and Touch Screen Interface

The following procedure is suggested:

1. Connect pads of the touch screen panel to the S3C2410A using external transistor (see Figure 16-2).
2. Select Separate X/Y Position Conversion Mode or Auto (Sequential) X/Y Position Conversion Mode to get X/Y position.
3. Set Touch Screen Interface to Waiting Interrupt Mode,
4. If interrupt occurs, then appropriate conversion (Separate X/Y Position Conversion Mode or Auto (Sequential) X/Y Position Conversion Mode) is activated.
5. After get the proper value of X/Y position, return to Waiting for Interrupt Mode.

NOTES

1. External voltage source should be 3.3 V.
2. Internal resistance of the external transistor should be under 5 ohm.

FUNCTION DESCRIPTIONS

A/D Conversion Time

When the PCLK frequency is 50 MHz and the prescaler value is 49, total 10-bit conversion time is given:

A/D converter freq. = 50 MHz/(49+1) = 1 MHz

Conversion time = 1/(1 MHz / 5cycles) = 1/200 kHz = 5 us

NOTE

This A/D converter is designed to operate at maximum 2.5 MHz clock, so the conversion rate can go up to 500 KSPS.

Touch Screen Interface Mode

1. Normal Conversion Mode

   Normal Conversion Mode (AUTO_PST = 0, XY_PST = 0) is generally used for General Purpose ADC Conversion. This mode can be initialized by setting the ADCCON and ADCTSC and completed with a read the XPDATA (Normal ADC) value of ADCDAT0 (ADC Data Register 0).

2. Separate X/Y Position Conversion Mode

   Separate X/Y Position Conversion Mode is consist of two Conversion Modes: X-Position Mode and Y-Position Mode.
   The first mode is operated in the following way:
   X-Position Mode (AUTO_PST = 0 and XY_PST = 1) writes X-position conversion data to XPDATA of ADCDAT0 register. After conversion, The Touch Screen Interface generates the Interrupt source (INT_ADC) to Interrupt Controller.
   Y-Position Mode (AUTO_PST = 0 and XY_PST = 2) writes Y-position conversion data to YPDATA of ADCDAT1. After the conversion, the Touch Screen Interface also generates the Interrupt source (INT_ADC) to Interrupt Controller.

Table 16-1. Condition of Touch Screen Panel Pads in Separate X/Y Position Conversion Mode

	XP	XM	YP	YM
X Position Conversion	External Voltage	GND	AIN[5]	Hi-Z
Y Position Conversion	AIN[7]	Hi-Z	External Voltage	GND

3. Auto (Sequential) X/Y Position Conversion Mode.

   Auto (Sequential) X/Y Position Conversion Mode (AUTO_PST = 1 and XY_PST = 0) is operated in the following way:
   The Touch Screen Controller automatically converts X-position and Y-position. The Touch Screen Controller writes X-measurement data to XPDATA of ADCDAT0, and then writes Y-measurement data to YPDATA of ADCDAT1. After Auto (Sequential) Position Conversion, The Touch Screen Controller generates Interrupt source (INT_ADC) to Interrupt Controller.

Table 16-2. Condition of Touch Screen Panel Pads in Auto (Sequential) X/Y Position Conversion Mode.

	XP	XM	YP	YM
X Position Conversion	External Voltage	GND	AIN[5]	Hi-Z
Y Position Conversion	AIN[7]	Hi-Z	External Voltage	GND

4. Waiting for Interrupt Mode.

   When Touch Screen Controller is in Waiting for Interrupt Mode, it waits for Stylus down. The controller generates Interrupt (INT_TC) signals when the Stylus is down on Touch Screen Panel.
   After an interrupt occurs, X and Y position can be read by the proper conversion mode (Separate X/Y position conversion Mode or Auto X/Y Position Conversion Mode).

Table 16-3. Condition of Touch Screen Panel Pads in Waiting for Interrupt Mode.

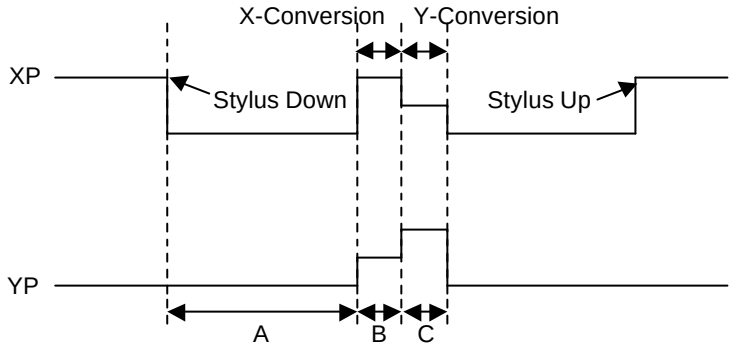
	XP	XM	YP	YM
Waiting for Interrupt Mode	Pull-up	Hi-Z	AIN[5]	GND

Standby Mode

Standby mode is activated when STDBM of ADCCON register is set to '1'. In this mode, A/D conversion operation is halted and XPDATA (Normal ADC) of ADCDAT0 and YPDATA of ADCDAT1 contain the previous converted data.

Programming Notes

1. The A/D converted data can be accessed by means of interrupt or polling method. With interrupt method, the overall conversion time - from A/D converter start to converted data read - may be delayed because of the return time of interrupt service routine and data access time. With polling method, by checking the ADCCON[15] - end of conversion flag-bit, the read time from ADCDAT register can be determined.
2. A/D conversion can be activated in different way: After ADCCON[1] - A/D conversion start-by-read mode-is set to 1, A/D conversion starts simultaneously whenever converted data is read.



A = **D** x (1/X-Tal Clock) or **A** = **D** x (1/External Clock)
B = **D** x (1/PCLK)
C = **D** x (1/PCLK)
D = DELAY value of ADCDLY Register

Figure 16-3 Timing Diagram in Auto (Sequential) X/Y Position Conversion Mode

ADC AND TOUCH SCREEN INTERFACE SPECIAL REGISTERS

ADC CONTROL (ADCCON) REGISTER

Register	Address	R/W	Description	Reset Value
ADCCON	0x58000000	R/W	ADC control register	0x3FC4

ADCCON	Bit	Description	Initial State
ECFLG	[15]	End of conversion flag (read only). 0 = A/D conversion in process 1 = End of A/D conversion	0
PRSCEN	[14]	A/D converter prescaler enable. 0 = Disable 1 = Enable	0
PRSCVL	[13:6]	A/D converter prescaler value. Data value: 1 ~ 255 Note that division factor is (N+1) when the prescaler value is N. **NOTE:** ADC frequency should be set less than PCLK by 5 times. (Ex. PCLK = 10MHz, ADC Frequency < 2MHz)	0xFF
SEL_MUX	[5:3]	Analog input channel select. 000 = AIN 0 001 = AIN 1 010 = AIN 2 011 = AIN 3 100 = AIN 4 101 = AIN 5 110 = AIN 6 111 = AIN 7 (XP)	0
STDBM	[2]	Standby mode select. 0 = Normal operation mode 1 = Standby mode	1
READ_ START	[1]	A/D conversion start by read. 0 = Disable start by read operation 1 = Enable start by read operation	0
ENABLE_ START	[0]	A/D conversion starts by setting this bit. If READ_START is enabled, this value is not valid. 0 = No operation 1 = A/D conversion starts and this bit is cleared after the start-up.	0

ADC TOUCH SCREEN CONTROL (ADCTSC) REGISTER

Register	Address	R/W	Description	Reset Value
ADCTSC	0x58000004	R/W	ADC touch screen control register	0x058

ADCTSC	Bit	Description	Initial State
Reserved	[8]	This bit should be zero.	0
YM_SEN	[7]	Select output value of YMON. 0 = YMON output is 0 (YM = Hi-Z). 1 = YMON output is 1 (YM = GND).	0
YP_SEN	[6]	Select output value of nYPON. 0 = nYPON output is 0 (YP = External voltage). 1 = nYPON output is 1 (YP is connected with AIN[5]).	1
XM_SEN	[5]	Select output value of XMON. 0 = XMON output is 0 (XM = Hi-Z). 1 = XMON output is 1 (XM = GND).	0
XP_SEN	[4]	Select output value of nXPON. 0 = nXPON output is 0 (XP = External voltage). 1 = nXPON output is 1 (XP is connected with AIN[7]).	1
PULL_UP	[3]	Pull-up switch enable. 0 = XP pull-up enable 1 = XP pull-up disable	1
AUTO_PST	[2]	Automatically sequencing conversion of X-position and Y-position 0 = Normal ADC conversion 1 = Auto (Sequential) X/Y Position Conversion Mode	0
XY_PST	[1:0]	Manual measurement of X-position or Y-position. 00 = No operation mode 01 = X-position measurement 10 = Y-position measurement 11 = Waiting for Interrupt Mode	0

NOTE: In Auto mode, ADCTSC register should be reconfigured before starting read operation.

ADC START DELAY (ADCDLY) REGISTER

Register	Address	R/W	Description	Reset Value
ADCDLY	0x58000008	R/W	ADC start or interval delay register	0x00ff

ADCDLY	Bit	Description	Initial State
DELAY	[15:0]	1) Normal Conversion Mode, Separate X/Y Position Conversion Mode, and Auto (Sequential) X/Y Position Conversion Mode. → X/Y Position Conversion Delay Value. 2) Waiting for Interrupt Mode. When Stylus down occurs in Waiting for Interrupt Mode, this register generates Interrupt signal (INT_TC) at intervals of several ms for Auto X/Y Position conversion. **NOTE:** Do not use Zero value (0x0000)	00ff

NOTES:

1. Before ADC conversion, Touch screen uses X-tal clock or EXTCLK (Waiting for Interrupt Mode).
2. During ADC conversion, PCLK is used.

ADC CONVERSION DATA (ADCDAT0) REGISTER

Register	Address	R/W	Description	Reset Value
ADCDAT0	0x5800000C	R	ADC conversion data register	-

ADCDAT0	Bit	Description	Initial State
UPDOWN	[15]	Up or down state of Stylus at Waiting for Interrupt Mode. 0 = Stylus down state 1 = Stylus up state	-
AUTO_PST	[14]	Automatic sequencing conversion of X-position and Y-position. 0 = Normal ADC conversion 1 = Sequencing measurement of X-position, Y-position	-
XY_PST	[13:12]	Manual measurement of X-position or Y-position. 00 = No operation mode 01 = X-position measurement 10 = Y-position measurement 11 = Waiting for Interrupt Mode	-
Reserved	[11:10]	Reserved	
XPDATA (Normal ADC)	[9:0]	X-position conversion data value. (include Normal ADC conversion data value) Data value: 0 ~ 3FF	-

ADC CONVERSION DATA (ADCDAT1) REGISTER

Register	Address	R/W	Description	Reset Value
ADCDAT1	0x58000010	R	ADC conversion data register	-

ADCDAT1	Bit	Description	Initial State
UPDOWN	[15]	Up or down state of Stylus at Waiting for Interrupt Mode. 0 = Stylus down state 1 = Stylus up state	-
AUTO_PST	[14]	Automatically sequencing conversion of X-position and Y-position. 0 = Normal ADC conversion 1 = Sequencing measurement of X-position, Y-position	-
XY_PST	[13:12]	Manual measurement of X-position or Y-position. 00 = No operation mode 01 = X-position measurement 10 = Y-position measurement 11 = Waiting for Interrupt Mode	-
Reserved	[11:10]	Reserved	
YPDATA	[9:0]	Y-position conversion data value Data value: 0 ~ 3FF	-

NOTES

17 REAL TIME CLOCK (RTC)

OVERVIEW

The Real Time Clock (RTC) unit can be operated by the backup battery while the system power is off. The RTC can transmit 8-bit data to CPU as Binary Coded Decimal (BCD) values using the STRB/LDRB ARM operation. The data include the time by second, minute, hour, date, day, month, and year. The RTC unit works with an external 32.768 kHz crystal and also can perform the alarm function.

FEATURES

— BCD number: second, minute, hour, date, day, month, and year

— Leap year generator

— Alarm function: alarms interrupt or wake-up from power-off mode

— Year 2000 problem is removed.

— Independent power pin (RTCVDD)

— Supports millisecond tick time interrupt for RTOS kernel time tick.

— Round reset function

REAL TIME CLOCK OPERATION

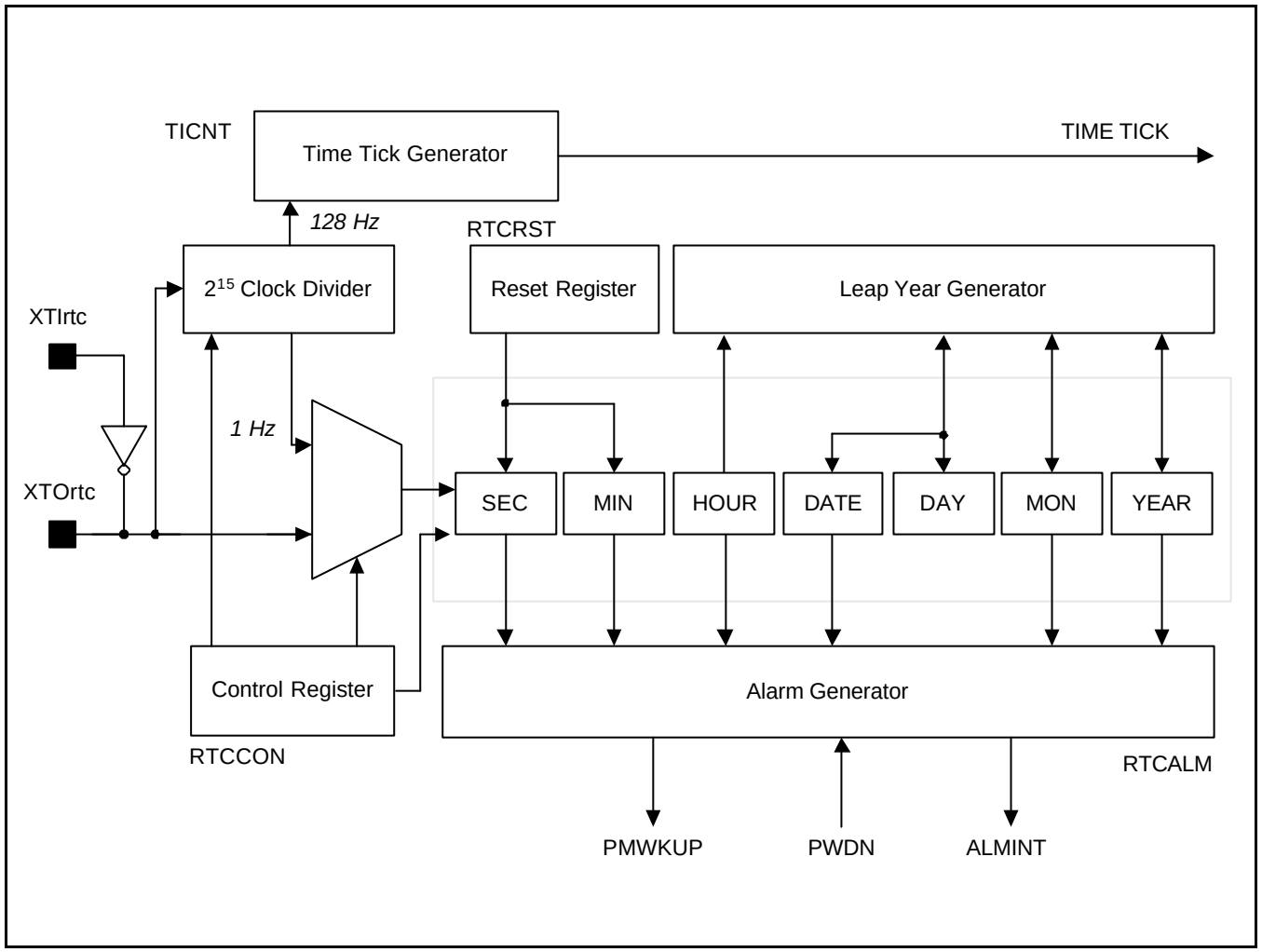


Figure 17-1. Real Time Clock Block Diagram

LEAP YEAR GENERATOR

The leap year generator can determine the last date of each month out of 28, 29, 30, or 31, based on data from BCDDATE, BCDMON, and BCDYEAR. This block considers leap year in deciding on the last date. An 8-bit counter can only represent 2 BCD digits, so it cannot decide whether "00" year (the year with its last two digits zeros) is a leap year or not. For example, it cannot discriminate between 1900 and 2000. To solve this problem, the RTC block in S3C2410A has hard-wired logic to support the leap year in 2000. Note 1900 is not leap year while 2000 is leap year. Therefore, two digits of 00 in S3C2410A denote 2000, not 1900.

READ/WRITE REGISTERS

Bit 0 of the RTCCON register must be set high in order to write the BCD register in RTC block. To display the second, minute, hour, date, month, and year, the CPU should read the data in BCDSEC, BCDMIN, BCDHOUR, BCDDAY, BCDDATE, BCDMON, and BCDYEAR registers, respectively, in the RTC block. However, a one second deviation may exist because multiple registers are read. For example, when the user reads the registers from BCDYEAR to BCDMIN, the result is assumed to be 2059 (Year), 12 (Month), 31 (Date), 23 (Hour) and 59 (Minute). When the user read the BCDSEC register and the value ranges from 1 to 59 (Second), there is no problem, but, if the value is 0 sec., the year, month, date, hour, and minute may be changed to 2060 (Year), 1 (Month), 1 (Date), 0 (Hour) and 0 (Minute) because of the one second deviation that was mentioned. In this case, the user should re-read from BCDYEAR to BCDSEC if BCDSEC is zero.

BACKUP BATTERY OPERATION

The RTC logic can be driven by the backup battery, which supplies the power through the RTCVDD pin into the RTC block, even if the system power is off. When the system is off, the interfaces of the CPU and RTC logic should be blocked, and the backup battery only drives the oscillation circuit and the BCD counters to minimize power dissipation.

ALARM FUNCTION

The RTC generates an alarm signal at a specified time in the power-off mode or normal operation mode. In normal operation mode, the alarm interrupt (ALMINT) is activated. In the power-off mode, the power management wakeup (PMWKUP) signal is activated as well as the ALMINT. The RTC alarm register (RTCALM) determines the alarm enable/disable status and the condition of the alarm time setting.

TICK TIME INTERRUPT

The RTC tick time is used for interrupt request. The TICNT register has an interrupt enable bit and the count value for the interrupt. The count value reaches '0' when the tick time interrupt occurs. Then the period of interrupt is as follows:

Period = (n+1) / 128 second
n: Tick time count value (1~127)

This RTC time tick may be used for real time operating system (RTOS) kernel time tick. If time tick is generated by the RTC time tick, the time related function of RTOS will always synchronized in real time.

ROUND RESET FUNCTION

The round reset function can be performed by the RTC round reset register (RTCRST). The round boundary (30, 40, or 50 sec.) of the second carry generation can be selected, and the second value is rounded to zero in the round reset. For example, when the current time is 23:37:47 and the round boundary is selected to 40 sec, the round reset changes the current time to 23:38:00.

NOTE

All RTC registers have to be accessed for each byte unit using the STRB and LDRB instructions or char type pointer.

32.768KHZ X-TAL CONNECTION EXAMPLE

The Figure 17-2 shows a circuit of the RTC unit oscillation at 32.768 kHz.

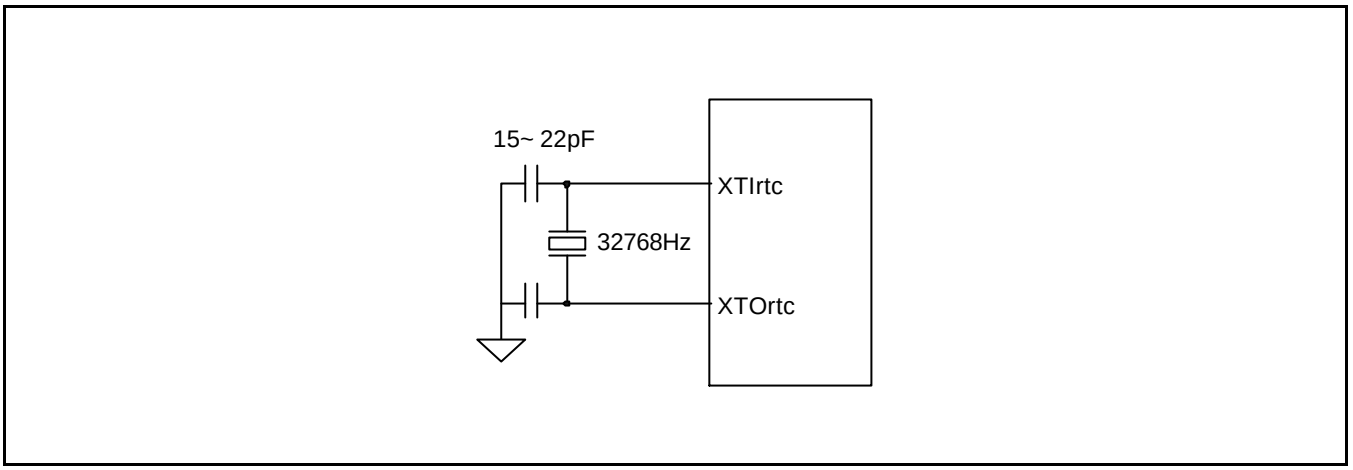


Figure 17-2. Main Oscillator Circuit Example

REAL TIME CLOCK SPECIAL REGISTERS

REAL TIME CLOCK CONTROL (RTCCON) REGISTER

The RTCCON register consists of 4 bits such as the RTCEN, which controls the read/write enable of the BCD registers, CLKSEL, CNTSEL, and CLKRST for testing.

RTCEN bit can control all interfaces between the CPU and the RTC, so it should be set to 1 in an RTC control routine to enable data read/write after a system reset. Also before power off, the RTCEN bit should be cleared to 0 to prevent inadvertent writing into RTC registers.

Register	Address	R/W	Description	Reset Value
RTCCON	0x57000040(L) 0x57000043(B)	R/W (by byte)	RTC control register	0x0

RTCCON	Bit	Description	Initial State
CLKRST	[3]	RTC clock count reset. 0 = No reset, 1 = Reset	0
CNTSEL	[2]	BCD count select. 0 = Merge BCD counters 1 = Reserved (Separate BCD counters)	0
CLKSEL	[1]	BCD clock select. 0 = XTAL $1/2^{15}$ divided clock 1 = Reserved (XTAL clock only for test)	0
RTCEN	[0]	RTC control enable. 0 = Disable 1 = Enable **NOTE:** Only BCD time count and read operation can be performed.	0

NOTES:

1. All RTC registers have to be accessed for each byte unit using STRB and LDRB instructions or char type pointer.
2. (L): Little endian.
   (B): Big endian.

TICK TIME COUNT (TICNT) REGISTER

Register	Address	R/W	Description	Reset Value
TICNT	0x57000044(L) 0x57000047(B)	R/W (by byte)	Tick time count register	0x0

TICNT	Bit	Description	Initial State
TICK INT ENABLE	[7]	Tick time interrupt enable. 0 = Disable 1 = Enable	0
TICK TIME COUNT	[6:0]	Tick time count value (1~127). This counter value decreases internally, and users cannot read this counter value in working.	000000

RTC ALARM CONTROL (RTCALM) REGISTER

The RTCALM register determines the alarm enable and the alarm time. Note that the RTCALM register generates the alarm signal through both ALMINT and PMWKUP in power down mode, but only through ALMINT in the normal operation mode.

Register	Address	R/W	Description	Reset Value
RTCALM	0x57000050(L) 0x57000053(B)	R/W (by byte)	RTC alarm control register	0x0

RTCALM	Bit	Description	Initial State
Reserved	[7]		0
ALMEN	[6]	Alarm global enable. 0 = Disable, 1 = Enable	0
YEAREN	[5]	Year alarm enable. 0 = Disable, 1 = Enable	0
MONREN	[4]	Month alarm enable. 0 = Disable, 1 = Enable	0
DATEEN	[3]	Date alarm enable. 0 = Disable, 1 = Enable	0
HOUREN	[2]	Hour alarm enable. 0 = Disable, 1 = Enable	0
MINEN	[1]	Minute alarm enable. 0 = Disable, 1 = Enable	0
SECEN	[0]	Second alarm enable. 0 = Disable, 1 = Enable	0

ALARM SECOND DATA (ALMSEC) REGISTER

Register	Address	R/W	Description	Reset Value
ALMSEC	0x57000054(L) 0x57000057(B)	R/W (by byte)	Alarm second data register	0x0

ALMSEC	Bit	Description	Initial State
Reserved	[7]		0
SECDATA	[6:4]	BCD value for alarm second. 0 ~ 5	000
	[3:0]	0 ~ 9	0000

ALARM MIN DATA (ALMMIN) REGISTER

Register	Address	R/W	Description	Reset Value
ALMMIN	0x57000058(L) 0x5700005B(B)	R/W (by byte)	Alarm minute data register	0x00

ALMMIN	Bit	Description	Initial State
Reserved	[7]		0
MINDATA	[6:4]	BCD value for alarm minute. 0 ~ 5	000
	[3:0]	0 ~ 9	0000

ALARM HOUR DATA (ALMHOUR) REGISTER

Register	Address	R/W	Description	Reset Value
ALMHOUR	0x5700005C(L) 0x5700005F(B)	R/W (by byte)	Alarm hour data register	0x0

ALMHOUR	Bit	Description	Initial State
Reserved	[7:6]		00
HOURDATA	[5:4]	BCD value for alarm hour. 0 ~ 2	00
	[3:0]	0 ~ 9	0000

ALARM DATE DATA (ALMDATE) REGISTER

Register	Address	R/W	Description	Reset Value
ALMDATE	0x57000060(L) 0x57000063(B)	R/W (by byte)	Alarm date data register	0x01

ALMDAY	Bit	Description	Initial State
Reserved	[7:6]		00
DATEDATA	[5:4]	BCD value for alarm date, from 0 to 28, 29, 30, 31. 0 ~ 3	00
	[3:0]	0 ~ 9	0001

ALARM MON DATA (ALMMON) REGISTER

Register	Address	R/W	Description	Reset Value
ALMMON	0x57000064(L) 0x57000067(B)	R/W (by byte)	Alarm month data register	0x01

ALMMON	Bit	Description	Initial State
Reserved	[7:5]		00
MONDATA	[4]	BCD value for alarm month. 0 ~ 1	0
	[3:0]	0 ~ 9	0001

ALARM YEAR DATA (ALMYEAR) REGISTER

Register	Address	R/W	Description	Reset Value
ALMYEAR	0x57000068(L) 0x5700006B(B)	R/W (by byte)	Alarm year data register	0x0

ALMYEAR	Bit	Description	Initial State
YEARDATA	[7:0]	BCD value for year. 00 ~ 99	0x0

RTC ROUND RESET (RTCRST) REGISTER

Register	Address	R/W	Description	Reset Value
RTCRST	0x5700006C(L) 0x5700006F(B)	R/W (by byte)	RTC round reset register	0x0

RTCRST	Bit	Description	Initial State
SRSTEN	[3]	Round second reset enable. 0 = Disable, 1 = Enable	0
SECCR	[2:0]	Round boundary for second carry generation. 011 = over than 30 sec 100 = over than 40 sec 101 = over than 50 sec **NOTE:** If other values (0, 1, 2, 6, or 7) are set, no second carry is generated. But second value can be reset.	000

BCD SECOND (BCDSEC) REGISTER

Register	Address	R/W	Description	Reset Value
BCDSEC	0x57000070(L) 0x57000073(B)	R/W (by byte)	BCD second register	Undefined

BCDSEC	Bit	Description	Initial State
SECDATA	[6:4]	BCD value for second. 0 ~ 5	-
	[3:0]	0 ~ 9	-

BCD MINUTE (BCDMIN) REGISTER

Register	Address	R/W	Description	Reset Value
BCDMIN	0x57000074(L) 0x57000077(B)	R/W (by byte)	BCD minute register	Undefined

BCDMIN	Bit	Description	Initial State
MINDATA	[6:4]	BCD value for minute. 0 ~ 5	-
	[3:0]	0 ~ 9	-

BCD HOUR (BCDHOUR) REGISTER

Register	Address	R/W	Description	Reset Value
BCDHOUR	0x57000078(L) 0x5700007B(B)	R/W (by byte)	BCD hour register	Undefined

BCDHOUR	Bit	Description	Initial State
Reserved	[7:6]		-
HOURDATA	[5:4]	BCD value for hour. 0 ~ 2	-
	[3:0]	0 ~ 9	-

BCD DATE (BCDDATE) REGISTER

Register	Address	R/W	Description	Reset Value
BCDDATE	0x5700007C(L) 0x5700007F(B)	R/W (by byte)	BCD date register	Undefined

BCDDATE	Bit	Description	Initial State
Reserved	[7:6]		-
DATEDATA	[5:4]	BCD value for date. 0 ~ 3	-
	[3:0]	0 ~ 9	-

BCD DAY (BCDDAY) REGISTER

Register	Address	R/W	Description	Reset Value
BCDDAY	0x57000080(L) 0x57000083(B)	R/W (by byte)	BCD a day of the week register	Undefined

BCDDAY	Bit	Description	Initial State
Reserved	[7:3]		-
DAYDATA	[2:0]	BCD value for a day of the week. 1 ~ 7	-

BCD MONTH (BCDMON) REGISTER

Register	Address	R/W	Description	Reset Value
BCDMON	0x57000084(L) 0x57000087(B)	R/W (by byte)	BCD month register	Undefined

BCDMON	Bit	Description	Initial State
Reserved	[7:5]		-
MONDATA	[4]	BCD value for month. 0 ~ 1	-
	[3:0]	0 ~ 9	-

BCD YEAR (BCDYEAR) REGISTER

Register	Address	R/W	Description	Reset Value
BCDYEAR	0x57000088(L) 0x5700008B(B)	R/W (by byte)	BCD year register	Undefined

BCDYEAR	Bit	Description	Initial State
YEARDATA	[7:0]	BCD value for year. 00 ~ 99	-

NOTES

18 WATCHDOG TIMER

OVERVIEW

The S3C2410A watchdog timer is used to resume the controller operation whenever it is disturbed by malfunctions such as noise and system errors. It can be used as a normal 16-bit interval timer to request interrupt service. The watchdog timer generates the reset signal for 128 PCLK cycles.

FEATURES

— Normal interval timer mode with interrupt request

— Internal reset signal is activated for 128 PCLK cycles when the timer count value reaches 0 (time-out).

WATCHDOG TIMER OPERATION

Figure 18-1 shows the functional block diagram of the watchdog timer. The watchdog timer uses only PCLK as its source clock. The PCLK frequency is prescaled to generate the corresponding watchdog timer clock, and the resulting frequency is divided again.

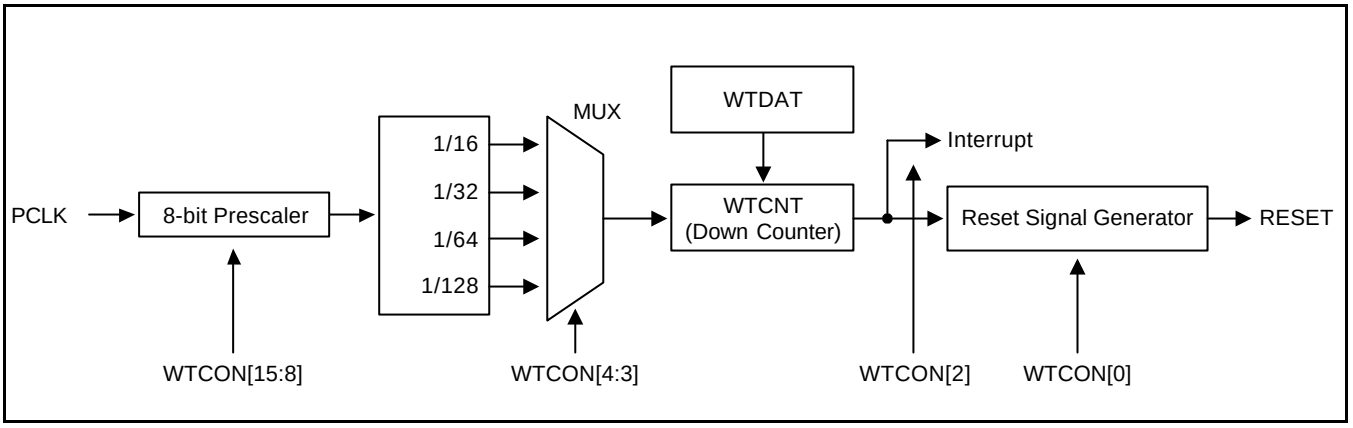


Figure 18-1. Watchdog Timer Block Diagram

The prescaler value and the frequency division factor are specified in the watchdog timer control (WTCON) register. Valid prescaler values range from 0 to 2^8-1. The frequency division factor can be selected as 16, 32, 64, or 128.

Use the following equation to calculate the watchdog timer clock frequency and the duration of each timer clock cycle:

t_watchdog = 1/(PCLK / (Prescaler value + 1) / Division_factor)

WTDAT & WTCNT

Once the watchdog timer is enabled, the value of watchdog timer data (WTDAT) register cannot be automatically reloaded into the timer counter (WTCNT). In this reason, an initial value must be written to the watchdog timer count (WTCNT) register, before the watchdog timer starts.

CONSIDERATION OF DEBUGGING ENVIRONMENT

When the S3C2410A is in debug mode using Embedded ICE, the watchdog timer is disabled, will be disabled automatically.

The watchdog timer can determine whether or not it is currently in the debug mode from the CPU core signal (DBGACK signal). Once the DBGACK signal is asserted, the reset output of the watchdog timer is not activated as the watchdog timer is expired.

WATCHDOG TIMER SPECIAL REGISTERS

WATCHDOG TIMER CONTROL (WTCON) REGISTER

The WTCON register allows the user to enable/disable the watchdog timer, select the clock signal from 4 different sources, enable/disable interrupts, and enable/disable the watchdog timer output.
The Watchdog timer is used to resume the S3C2410A restart on mal-function after its power on; if controller restart is not desired, the Watchdog timer should be disabled.

If the user wants to use the normal timer provided by the Watchdog timer, enable the interrupt and disable the Watchdog timer.

Register	Address	R/W	Description	Reset Value
WTCON	0x53000000	R/W	Watchdog timer control register	0x8021

WTCON	Bit	Description	Initial State
Prescaler Value	[15:8]	Prescaler value. The valid range is from 0 to (2^8-1).	0x80
Reserved	[7:6]	Reserved. These two bits must be 00 in normal operation.	00
Watchdog Timer	[5]	Enable or disable bit of Watchdog timer. 0 = Disable 1 = Enable	1
Clock Select	[4:3]	Determine the clock division factor. 00: 16 01: 32 10: 64 11: 128	00
Interrupt Generation	[2]	Enable or disable bit of the interrupt. 0 = Disable 1 = Enable	0
Reserved	[1]	Reserved. This bit must be 0 in normal operation.	0
Reset Enable/Disable	[0]	Enable or disable bit of Watchdog timer output for reset signal. 1: Assert reset signal of the S3C2410A at watchdog time-out 0: Disable the reset function of the watchdog timer.	1

WATCHDOG TIMER DATA (WTDAT) REGISTER

The WTDAT register is used to specify the time-out duration. The content of WTDAT cannot be automatically loaded into the timer counter at initial watchdog timer operation. However, using 0x8000 (initial value) will drive the first time-out. In this case, the value of WTDAT will be automatically reloaded into WTCNT.

Register	Address	R/W	Description	Reset Value
WTDAT	0x53000004	R/W	Watchdog timer data register	0x8000

WTDAT	Bit	Description	Initial State
Count Reload Value	[15:0]	Watchdog timer count value for reload.	0x8000

WATCHDOG TIMER COUNT (WTCNT) REGISTER

The WTCNT register contains the current count values for the watchdog timer during normal operation. Note that the content of the WTDAT register cannot be automatically loaded into the timer count register when the watchdog timer is enabled initially, so the WTCNT register must be set to an initial value before enabling it.

Register	Address	R/W	Description	Reset Value
WTCNT	0x53000008	R/W	Watchdog timer count register	0x8000

WTCNT	Bit	Description	Initial State
Count Value	[15:0]	The current count value of the watchdog timer	0x8000

19 MMC/SD/SDIO HOST CONTROLLER

OVERVIEW

The S3C2410A SD Host controller can support MMC/SD card and SDIO devices.

FEATURES

— SD Memory Card Spec. (ver. 1.0) / MMC Spec. (2.11) compatible
— SDIO Card Spec (ver. 1.0) compatible
— 16 words (64 bytes) FIFO (depth 16) for data Tx/Rx
— 40-bit Command Register (SDICARG[31:0]+SDICCON[7:0])
— 136-bit Response Register (SDIRSPn[127:0]+ SDICSTA[7:0])
— 8-bit Prescaler logic (Freq. = System Clock / (2(P + 1)))
— CRC7 & CRC16 Generator
— Polling, Interrupt and DMA Data Transfer Mode (Byte or Word transfer)
— 1-bit / 4-bit (wide bus) Mode & Block / Stream Mode Switch support
— Supports up to 25 MHz in data transfer mode for SD/SDIO
— Supports up to 20 MHz in data transfer mode for MMC

BLOCK DIAGRAM

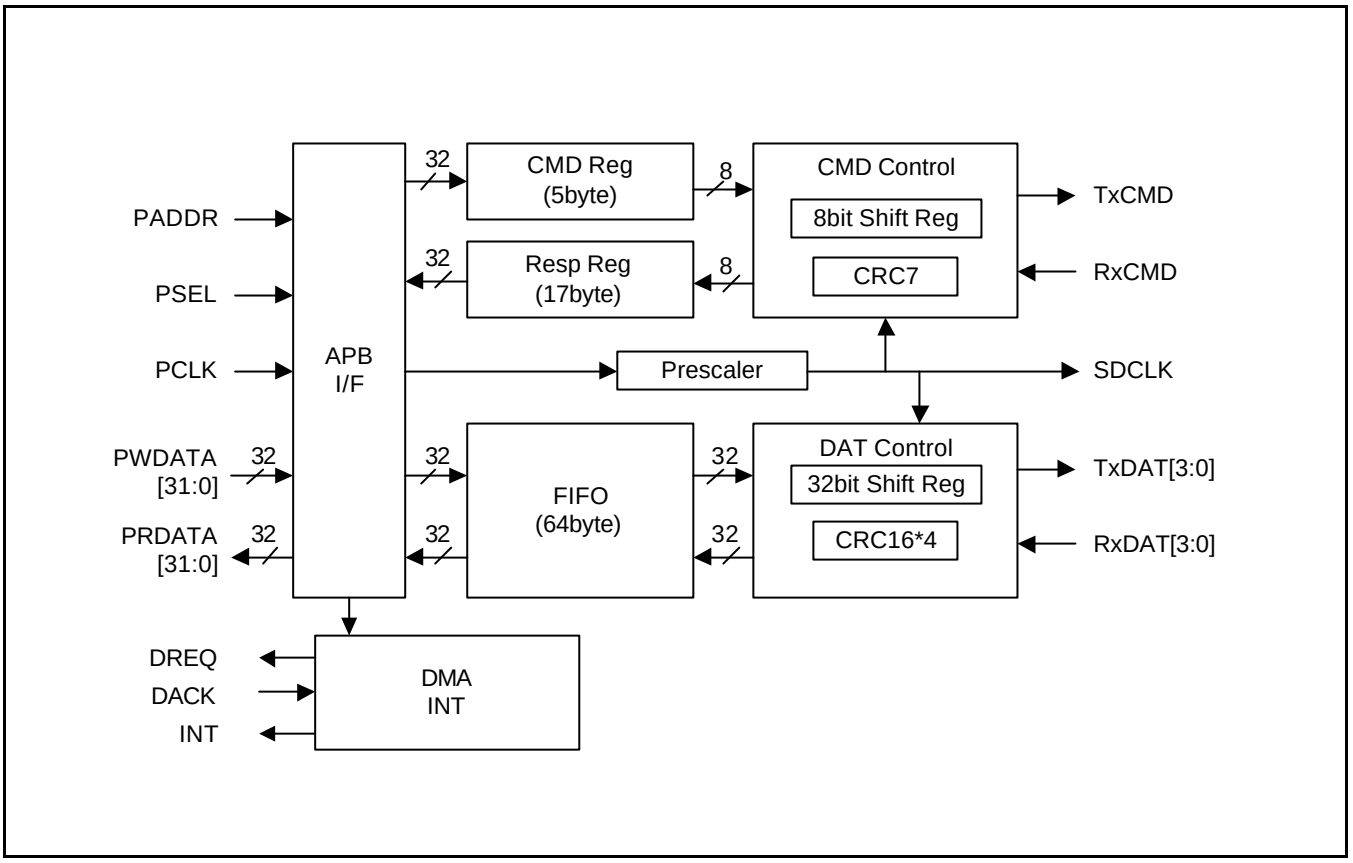


Figure 19-1. Block Diagram

SDI OPERATION

A serial clock line is synchronized with the five data lines for shifting and sampling of the information. Making the appropriate bit settings to the SDIPRE register depends on the transmission frequency. You can modify its frequency to adjust the baud rate data register value.

Programming Procedure (common)

SDI modules can be programmed, following these basic steps:

1. Set SDICON to configure properly with clock and interrupt.
2. Set SDIPRE to configure with a proper value.
3. Wait 74 SDCLK clock cycle in order to initialize the card.

CMD Path Programming

1. Write command argument (32-bit) to SDICARG register.
2. Determine command types and start command by setting SDICCON[8].
3. Confirm the end of SDI command operation when the specific flag of SDICSTA is set.
   — If the type of command is no-response, the flag is SDICSTA[11].
   — If the type of command is with-response, the flag is SDICSTA[9].
4. Clear the corresponding flag of the SDICSTA register by writing one to the flag bit.

DAT Path Programming

1. Write timeout period to SDIDTIMER register.
2. Write block size (block length) to SDIBSIZE register (normally 0x200 byte).
3. Determine the mode of block, wide bus, DMA, etc. and start data transfer with setting SDIDCON register.
4. Write Tx-data to SDIDAT register while Tx FIFO is available by checking SDIFSTA (available, half or empty) register.
5. Read Rx-data from SDIDAT register while Rx FIFO is available by checking SDIFSTA (available, half or be last data) register.
6. Confirm the end of SDI data operation when the flag of data transfer finish (SDIDSTA[4]) is set.
7. Clear the corresponding flag of SDIDSTA register by writing one to the flag bit.

NOTE

In case of long response command, CRC error can be wrong by H/W but a user can ignore this. It should be detected by software if it is need to check.

SDIO OPERATION

There are two functions of the SDIO operation: SDIO Interrupt receiving and Read Wait Request generation. These two functions can operate when RcvIOInt bit and RwaitEn bit of SDICON register is activated respectively. Detailed steps and conditions for the two functions are described below.

SDIO Interrupt

In SD 1-bit mode, the interrupt is received through all ranges from SDDAT1 pin.

In SD 4-bit mode, SDDAT1 pin is shared between to receive data and interrupts. When interrupt detection ranges (Interrupt Period) are:

1. Single Block: the time between A and B
   - A: 2clocks after the completion of a data packet
   - B: The completion of sending the end bit of the next with-data command

2. Multi Block, SDIDCON[21] = 0: the time between A and B, restart interrupt detection range at C
   - A: 2clocks after the completion of a data packet
   - B: 2clocks after A
   - C: 2clocks after the end bit of the abort command response

3. Multi Block, SDIDCON[21] = 1: the time between A and B, restart at A
   - A: 2clocks after the completion of a data packet
   - B: 2clocks after A

In case of last block, interrupt period begins at last A, but it does not end at B (CMD53 case).

Read Wait Request

Regardless of 1-bit or 4-bit mode, Read Wait Request signal transmits to SDDAT2 pin in the condition below.

- In read multiple operation, request signal transmission begins in 2clocks after the end of data block.
- Transmission ends when the user writes one to SDIDSTA[10].

SDI SPECIAL REGISTERS

SDI Control (SDICON) Register

Register	Address	R/W	Description	Reset Value
SDICON	0x5A000000	R/W	SDI control register	0x0

SDICON	Bit	Description	Initial Value
Byte Order Type (ByteOrder)	[4]	Determine byte order type when you read (write) data from (to) SD host FIFO with word boundary. 0 = Type A, 1 = Type B	0
Receive SDIO Interrupt from card (RcvIOInt)	[3]	Determine whether SD host receives SDIO Interrupt from the card or not (for SDIO). 0 = ignore, 1 = receive SDIO Interrupt	0
Read Wait Enable (RWaitEn)	[2]	Determine read wait request signal generate when SD host waits the next block in multiple block read mode. This bit needs to delay the next block to be transmitted from the card (for SDIO). 0 = disable (no generate), 1 = Read wait enable (use SDIO)	0
FIFO Reset (FRST)	[1]	Reset FIFO value. This bit is automatically cleared. 0 = normal mode, 1 = FIFO reset	0
Clock Type (CTYP)	[0]	Determines which clock type is used as SDCLK. 0 = MMC Type, 1 = SD Type	0

NOTE: Byte Order Type
- Type A: D[7:0] → D[15:8] → D[23:16] → D[31:24]
- Type B: D[31:24] → D[23:16] → D[15:8] → D[7:0]

SDI Baud Rate Prescaler (SDIPRE) Register

Register	Address	R/W	Description	Reset Value
SDIPRE	0x5A000004	R/W	SDI baud rate prescaler register	0x0

SDIPRE	Bit	Description	Initial Value
Prescaler Value	[7:0]	Determine SDI clock (SDCLK) rate as above equation. Baud rate = PCLK / 2 / (Prescaler value + 1)	0x00

SDI Command Argument Register (SDICARG)

Register	Address	R/W	Description	Reset Value
SDICARG	0x5A000008	R/W	SDI command argument register	0x0

SDICARG	Bit	Description	Initial Value
CmdArg	[31:0]	Command Argument	0x00000000

SDI Command Control (SDICCON) Register

Register	Address	R/W	Description	Reset Value
SDICCON	0x5A00000C	R/W	SDI command control register	0x0

SDICCON	Bit	Description	Initial Value
Abort Command (AbortCmd)	[12]	Determine whether command type is for abort (for SDIO). 0 = normal command, 1 = abort command (CMD12, CMD52)	0
Command with Data (WithData)	[11]	Determine whether command type is with data (for SDIO). 0 = without data, 1 = with data	0
LongRsp	[10]	Determine whether host receives a 136-bit long response or not. 0 = short response, 1 = long response	0
WaitRsp	[9]	Determine whether host waits for a response or not. 0 = no response, 1 = wait response	0
Command Start(CMST)	[8]	Determine whether command operation starts or not. 0 = command ready, 1 = command start	0
CmdIndex	[7:0]	Command index with start 2-bit (8-bit)	0x00

SDI Command Status (SDICSTA) Register

Register	Address	R/W	Description	Reset Value
SDICSTA	0x5A000010	R/(W)	SDI command status register	0x0

SDICSTA	Bit	Description	Initial Value
Response CRC Fail(RspCrc)	[12] R/W	CRC check failed when command response received. This flag is cleared by setting one to this bit. 0 = not detect, 1 = crc fail	0
Command Sent (CmdSent)	[11] R/W	Command sent (not concerned with response). This flag is cleared by setting one to this bit. 0 = not detect, 1 = command end	0
Command Time Out (CmdTout)	[10] R/W	Command response timeout (64clk). This flag is cleared by setting one to this bit. 0 = not detect, 1 = timeout	0
Response Receive End (RspFin)	[9] R/W	Command response received. This flag is cleared by setting one to this bit. 0 = not detect, 1 = response end	0
CMD line progress On (CmdOn)	[8] R	Command transfer in progress. 0 = not detect, 1 = in progress	0
RspIndex	[7:0] R	Response index 6bit including start 2-bit (8-bit)	0x00

SDI Response Register 0 (SDIRSP0)

Register	Address	R/W	Description	Reset Value
SDIRSP0	0x5A000014	R	SDI response register 0	0x0

SDIRSP0	Bit	Description	Initial Value
Response0	[31:0]	Card status[31:0](short), card status[127:96](long)	0x00000000

SDI Response Register 1 (SDIRSP1)

Register	Address	R/W	Description	Reset Value
SDIRSP1	0x5A000018	R	SDI response register 1	0x0

SDIRSP1	Bit	Description	Initial Value
RCRC7	[31:24]	CRC7 (with end bit, short), card status[95:88](long)	0x00
Response1	[23:0]	Unused (short), card status[87:64](long)	0x000000

SDI Response Register 2 (SDIRSP2)

Register	Address	R/W	Description	Reset Value
SDIRSP2	0x5A00001C	R	SDI response register 2	0xy0

SDIRSP2	Bit	Description	Initial Value
Response2	[31:0]	Unused (short), card status[63:32](long)	0x00000000

SDI Response Register 3 (SDIRSP3)

Register	Address	R/W	Description	Reset Value
SDIRSP3	0x5A000020	R	SDI response register 3	0x0y

SDIRSP3	Bit	Description	Initial Value
Response3	[31:0]	Unused (short), card status[31:0](long)	0x00000000

SDI Data / Busy Timer (SDIDTIMER) Register

Register	Address	R/W	Description	Reset Value
SDIDTIMER	0x5A000024	R/W	SDI data / busy timer register	0x2000

SDIDTIMER	Bit	Description	Initial Value
DataTimer	[15:0]	Data / busy timeout period (0~65535 cycle)	0x2000

SDI Block Size (SDIBSIZE) Register

Register	Address	R/W	Description	Reset Value
SDIBSIZE	0x5A000028	R/W	SDI block size register	0x0

SDIBSIZE	Bit	Description	Initial Value
BlkSize	[11:0]	Block size value (0~4095 byte). Do not care when stream mode	0x000

NOTE: In Case of multi block, BlkSize should be divided by word(4byte).(BlkSize[1:0] = 00)

SDI Data Control (SDIDCON) Register

Register	Address	R/W	Description	Reset Value
SDIDCON	0x5A00002C	R/W	SDI data control register	0x0

SDIDCON	Bit	Description	Initial Value
SDIO Interrupt Period Type (PrdType)	[21]	Determine whether SDIO Interrupt period is 2 cycle or extend more cycle when last data block is transferred (for SDIO). 0 = exactly 2 cycle, 1 = more cycles(like single block)	0
Transmit After Response (TARSP)	[20]	Determine when data transmit start after response receive or not. 0 = directly after DatMode set, 1 = after response receive(assume DatMode sets to 2'b11)	0
Receive After Command (RACMD)	[19]	Determine when data receive start after command sent or not. 0 = directly after DatMode set, 1 = after command sent (assume DatMode sets to 2'b10)	0
Busy After Command (BACMD)	[18]	Determine when busy receive start after command sent or not. 0 = directly after DatMode set, 1 = after command sent (assume DatMode sets to 2'b01)	0
Block mode (BlkMode)	[17]	Data transfer mode. 0 = stream data transfer, 1 = block data transfer	0
Wide bus enable (WideBus)	[16]	Determine enable wide bus mode. 0 = standard bus mode(only SDIDAT[0] used), 1 = wide bus mode(SDIDAT[3:0] used)	0
DMA Enable (EnDMA)	[15]	Enable DMA. 0 = disable(polling), 1 = dma enable When DMA operation is completed, this bit should be disabled.	0
Stop by force (STOP)	[14]	Determine whether data transfer stop by force or not. 0 = normal, 1 = stop by force	0
Data Transfer Mode (DatMode)	[13:12]	Determine the direction of data transfer. 00 = ready, 01 = only busy check start 10 = data receive start, 11 = data transmit start	00
BlkNum	[11:0]	Block Number (0~4095). Do not care when stream mode.	0x000

NOTES:

1. If you want one of TARSP, RACMD and BACMD bits (SDIDCON[20:18]) to "1", you need to write on SDIDCON register head of on SDICCON register (always need for SDIO).
2. When DMA operation is completed, DMA Enable[15] bit of SDIDCON register should be disabled.

SDI Data Remain Counter (SDIDCNT) Register

Register	Address	R/W	Description	Reset Value
SDIDCNT	0x5A000030	R	SDI data remain counter register	0x0

SDIDCNT	Bit	Description	Initial Value
BlkNumCnt	[23:12]	Remaining block number	0x000
BlkCnt	[11:0]	Remaining data byte of 1 block	0x000

SDI Data Status (SDIDSTA) Register

Register	Address	R/W	Description	Reset Value
SDIDSTA	0x5A000034	R/(W)	SDI data status register	0x0

SDIDSTA	Bit	Description	Initial Value
Read Wait Request Occur (RWaitReq)	[10] R/W	Read wait request signal transmits to SD card. The request signal is stopped and this flag is cleared by setting one to this bit.(for SDIO) 0 = not occur, 1 = Read wait request occur	0
SDIO Interrupt Detect (IOIntDet)	[9] R/W	SDIO interrupt detects. This flag is cleared by setting one to this bit.(for SDIO) 0 = not detect, 1 = SDIO interrupt detect	0
FIFO Fail error (FFfail)	[8] R/W	FIFO fail error when FIFO occurs overrun / underrun / misaligned data saving. This flag is cleared by setting one to this bit. 0 = not detect, 1 = FIFO fail	0
CRC Status Fail (CrcSta)	[7] R/W	CRC Status error when data block sent (CRC check failed - returned from card). This flag is cleared by setting one to this bit. 0 = not detect, 1 = crc status fail	0
Data Receive CRC Fail (DatCrc)	[6] R/W	Data block received error (CRC check failed - calculated by host). This flag is cleared by setting one to this bit. 0 = not detect, 1 = receive crc fail	0
Data Time Out (DatTout)	[5] R/W	Data / Busy receive timeout. This flag is cleared by setting one to this bit. 0 = not detect, 1 = timeout	0
Data Transfer Finish (DatFin)	[4] R/W	Data transfer completes (data counter is zero). This flag is cleared by setting one to this bit. 0 = not detect, 1 = data finish detect	0
Busy Finish (BusyFin)	[3] R/W	Only busy check finish. This flag is cleared by setting one to this bit. 0 = not detect, 1 = busy finish detect	0
Reserved	[2]		0
Tx Data progress On (TxDatOn)	[1] R	Data transmit in progress. 0 = not active, 1 = data Tx in progress	0
Rx Data Progress On (RxDatOn)	[0] R	Data receive in progress. 0 = not active, 1 = data Rx in progress	0

SDI FIFO Status (SDIFSTA) Register

Register	Address	R/W	Description	Reset Value
SDIFSTA	0x5A000038	R	SDI FIFO status register	0x0

SDIFSTA	Bit	Description	Initial State
FIFO available Detect for Tx (TFDET)	[13]	Indicate that FIFO data is available for transmission when DatMode (SDIDCON[12]) is data transmit mode. If DMA mode is enable, SD host requests DMA operation. 0 = not detect (FIFO full), 1 = detect(0 ≤ FIFO ≤ 63)	0
FIFO available Detect for Rx (RFDET)	[12]	Indicate that FIFO data is available for reception when DatMode (SDIDCON[12]) is data receive mode. If DMA mode is enable, SD host requests DMA operation. 0 = not detect(FIFO empty), 1 = detect(1 ≤ FIFO ≤ 64)	0
Tx FIFO Half Full (TFHalf)	[11]	Set to 1 whenever Tx FIFO is less than 33byte. 0 = 33 ≤ Tx FIFO ≤ 64, 1 = 0 ≤ Tx FIFO ≤ 32	0
Tx FIFO Empty (TFEmpty)	[10]	Set to 1 whenever Tx FIFO is empty. 0 = 1 ≤ Tx FIFO ≤ 64, 1 = Empty(0byte)	0
Rx FIFO Last Data Ready (RFLast)	[9]	Set to 1 whenever Rx FIFO has last data of all block. 0 = not received yet, 1 = Last data ready	0
Rx FIFO Full (RFFull)	[8]	Set to 1 whenever Rx FIFO is full. 0 = 0 ≤ Rx FIFO ≤ 63, 1 = Full(64byte)	0
Rx FIFO Half Full (RFHalf)	[7]	Set to 1 whenever Rx FIFO is more than 31byte. 0 = 0 ≤ Rx FIFO ≤ 31, 1 = 32 ≤ Rx FIFO ≤ 64	0
FIFO Count (FFCNT)	[6:0]	Number of data (byte) in FIFO	0000000

SDI Data (SDIDAT) Register

Register	Address	R/W	Description	Reset Value
SDIDAT	0x5A00003C(Li/W, Li/B, Bi/W) 0x5A00003F(Bi/B)	R/W	SDI data register	0x0

SDIDAT	Bit	Description	Initial State
Data Register	[31:0]	This field contains the data to be transmitted or received over the SDI channel.	0x00000000

NOTES:

1. (Li/W, Li/B): Access by Word/Byte unit when endian mode is Little
2. (Bi/W): Access by Word unit when endian mode is Big
3. (Bi/B) : Access by Byte unit when endian mode is Big

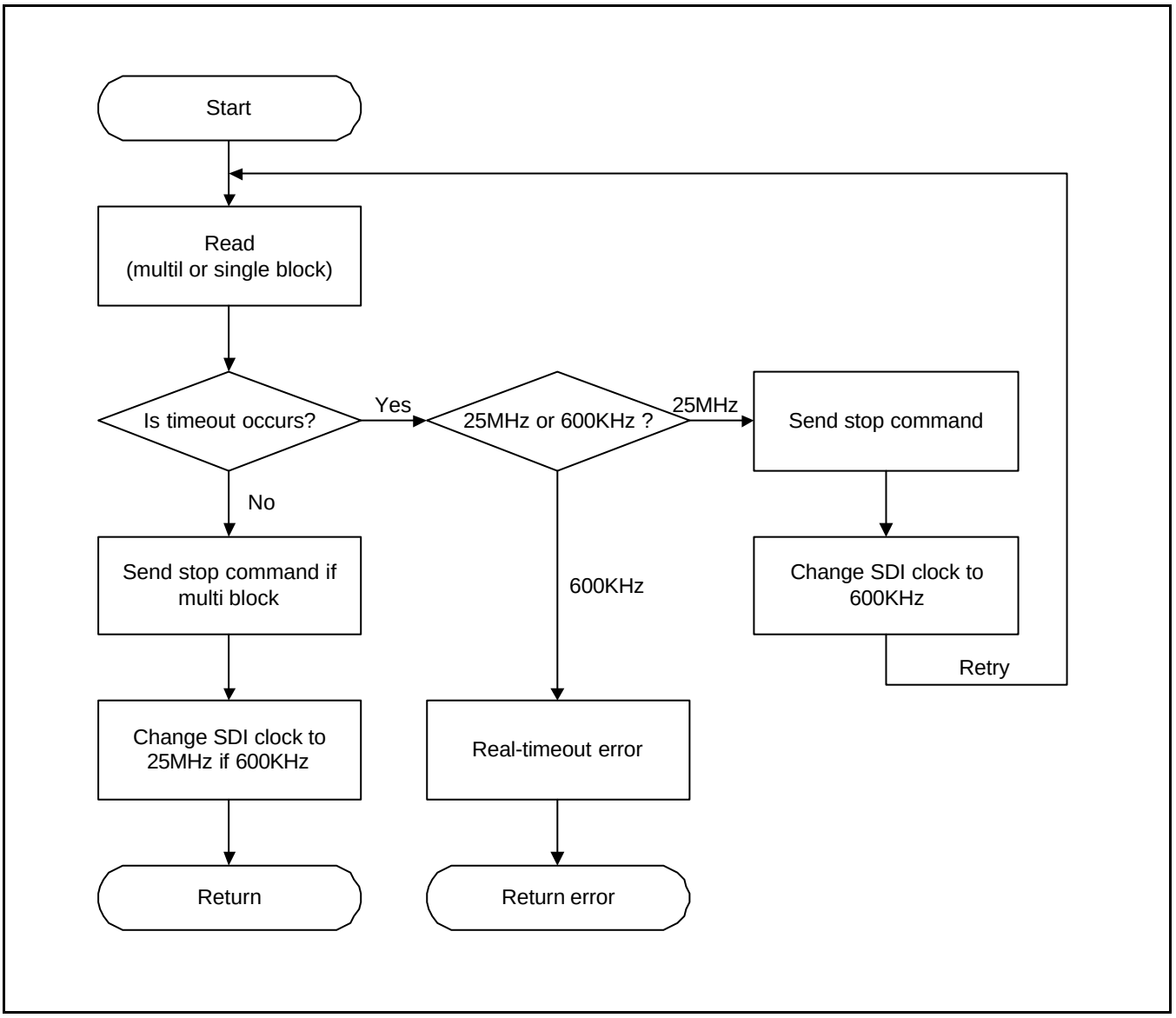
SDI Interrupt Mask (SDIIMSK) Register

Register	Address	R/W	Description	Reset Value
SDIIMSK	0x5A000040	R/W	SDI interrupt mask register	0x0

SDIIMSK	Bit	Description	Initial Value
RspCrc Interrupt Enable	[17]	Response CRC error interrupt. 0 = disable, 1 = interrupt enable	0
CmdSent Interrupt Enable	[16]	Command sent(without response) interrupt. 0 = disable, 1 = interrupt enable	0
CmdTout Interrupt Enable	[15]	Command response timeout interrupt. 0 = disable, 1 = interrupt enable	0
RspEnd Interrupt Enable	[14]	Command response received interrupt. 0 = disable, 1 = interrupt enable	0
RWaitReq Interrupt Enable	[13]	Read wait request interrupt. 0 = disable, 1 = interrupt enable	0
IOIntDet Interrupt Enable	[12]	SD host receives SDIO Interrupt from the card (for SDIO). 0 = disable, 1 = interrupt enable	0
FFfail Interrupt Enable	[11]	FIFO fail error interrupt. 0 = disable, 1 = interrupt enable	0
CrcSta Interrupt Enable	[10]	CRC status errors interrupt. 0 = disable, 1 = interrupt enable	0
DatCrc Interrupt Enable	[9]	Data CRC fail interrupt. 0 = disable, 1 = interrupt enable	0
DatTout Interrupt Enable	[8]	Data timeout interrupt. 0 = disable, 1 = interrupt enable	0
DatFin Interrupt Enable	[7]	Data counter zero interrupt. 0 = disable, 1 = interrupt enable	0
BusyFin Interrupt Enable	[6]	Busy checks complete interrupt. 0 = disable, 1 = interrupt enable	0
Reserved	[5]		0
TFHalf Interrupt Enable	[4]	Tx FIFO half interrupt. 0 = disable, 1 = interrupt enable	0
TFEmpty Interrupt Enable	[3]	Tx FIFO empty interrupt. 0 = disable, 1 = interrupt enable	0
RFLast Interrupt Enable	[2]	Rx FIFO has last data interrupt. 0 = disable, 1 = interrupt enable	0
RFFull Interrupt Enable	[1]	Rx FIFO full interrupt. 0 = disable, 1 = interrupt enable	0
RFHalf Interrupt Enable	[0]	Rx FIFO half interrupt. 0 = disable, 1 = interrupt enable	0

SDI Data/Busy Timer Register

SDI data/ busy timer register has 16-bit counter. In case of 25MHz operation, the countable maximum time is 2.6ms (40ns * 0x10000). But, some cards have very long access time (TAAC), their TAAC are up to 100ms. In this case the SDI generates data timeout error state. To solve this problem follow the below flow chart.



20 IIC-BUS INTERFACE

OVERVIEW

The S3C2410A RISC microprocessor can support a multi-master IIC-bus serial interface. A dedicated serial data line (SDA) and a serial clock line (SCL) carry information between bus masters and peripheral devices which are connected to the IIC-bus. The SDA and SCL lines are bi-directional.

In multi-master IIC-bus mode, multiple S3C2410A RISC microprocessors can receive or transmit serial data to or from slave devices. The master S3C2410A can initiate and terminate a data transfer over the IIC-bus. The IIC-bus in the S3C2410A uses Standard bus arbitration procedure.

To control multi-master IIC-bus operations, values must be written to the following registers:

— Multi-master IIC-bus control register, IICCON

— Multi-master IIC-bus control/status register, IICSTAT

— Multi-master IIC-bus Tx/Rx data shift register, IICDS

— Multi-master IIC-bus address register, IICADD

When the IIC-bus is free, the SDA and SCL lines should be both at High level. A High-to-Low transition of SDA can initiate a Start condition. A Low-to-High transition of SDA can initiate a Stop condition while SCL remains steady at High Level.

The Start and Stop conditions can always be generated by the master devices. A 7-bit address value in the first data byte, which is put onto the bus after the Start condition has been initiated, can determine the slave device which the bus master device has selected. The 8th bit determines the direction of the transfer (read or write).

Every data byte put onto the SDA line should be eight bits in total. The bytes can be unlimitedly sent or received during the bus transfer operation. Data is always sent from most-significant bit (MSB) first, and every byte should be immediately followed by an acknowledge (ACK) bit.

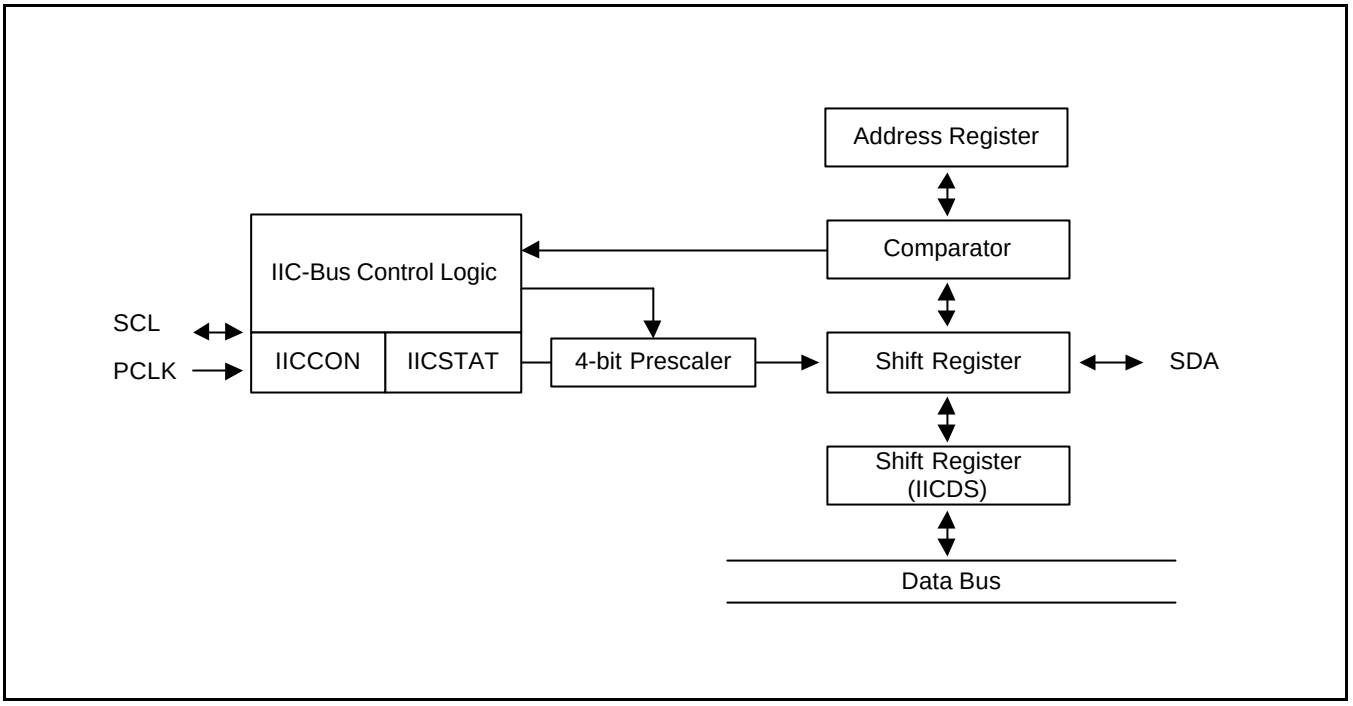


Figure 20-1. IIC-Bus Block Diagram

NOTE: IIC DATA HOLD TIME

The IIC data hold time(tSDAH) is minimum 0ns.
(IIC data hold time is minimum 0ns for standard/fast bus mode in IIC specification v2.1.)
Please check the data hold time of your IIC device if it's 0 nS or not.
The IIC controller supports only IIC bus device(standard/fast bus mode), not C bus device.

IIC-BUS INTERFACE

The S3C2410A IIC-bus interface has four operation modes:

— Master transmitter mode

— Master receive mode

— Slave transmitter mode

— Slave receive mode

Functional relationships among these operating modes are described below.

START AND STOP CONDITIONS

When the IIC-bus interface is inactive, it is usually in Slave mode. In other words, the interface should be in Slave mode before detecting a Start condition on the SDA line (a Start condition can be initiated with a High-to-Low transition of the SDA line while the clock signal of SCL is High). When the interface state is changed to Master mode, a data transfer on the SDA line can be initiated and SCL signal generated.

A Start condition can transfer a one-byte serial data over the SDA line, and a Stop condition can terminate the data transfer. A Stop condition is a Low-to-High transition of the SDA line while SCL is High. Start and Stop conditions are always generated by the master. The IIC-bus gets busy when a Start condition is generated. A Stop condition will make the IIC-bus free.

When a master initiates a Start condition, it should send a slave address to notify the slave device. One byte of address field consists of a 7-bit address and a 1-bit transfer direction indicator (showing write or read).
If bit 8 is 0, it indicates a write operation (transmit operation); if bit 8 is 1, it indicates a request for data read (receive operation).

The master will finish the transfer operation by transmitting a Stop condition. If the master wants to continue the data transmission to the bus, it should generate another Start condition as well as a slave address. In this way, the read-write operation can be performed in various formats.

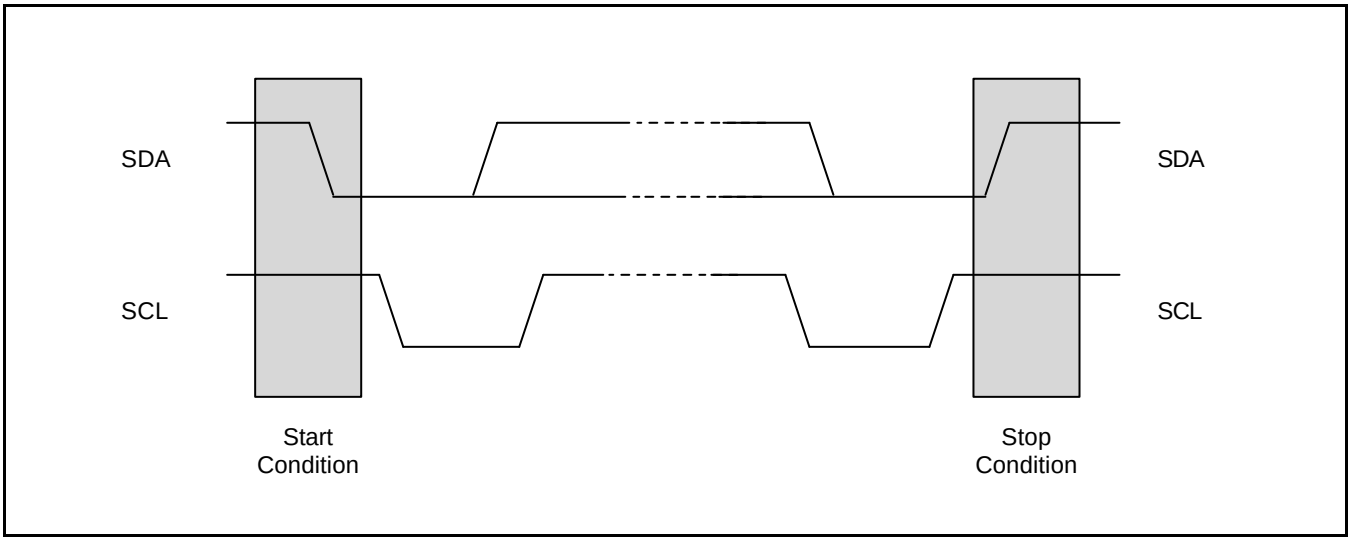


Figure 20-2. Start and Stop Condition

DATA TRANSFER FORMAT

Every byte placed on the SDA line should be eight bits in length. The bytes can be unlimitedly transmitted per transfer. The first byte following a Start condition should have the address field. The address field can be transmitted by the master when the IIC-bus is operating in Master mode. Each byte should be followed by an acknowledgement (ACK) bit. The MSB bit of the serial data and addresses are always sent first.

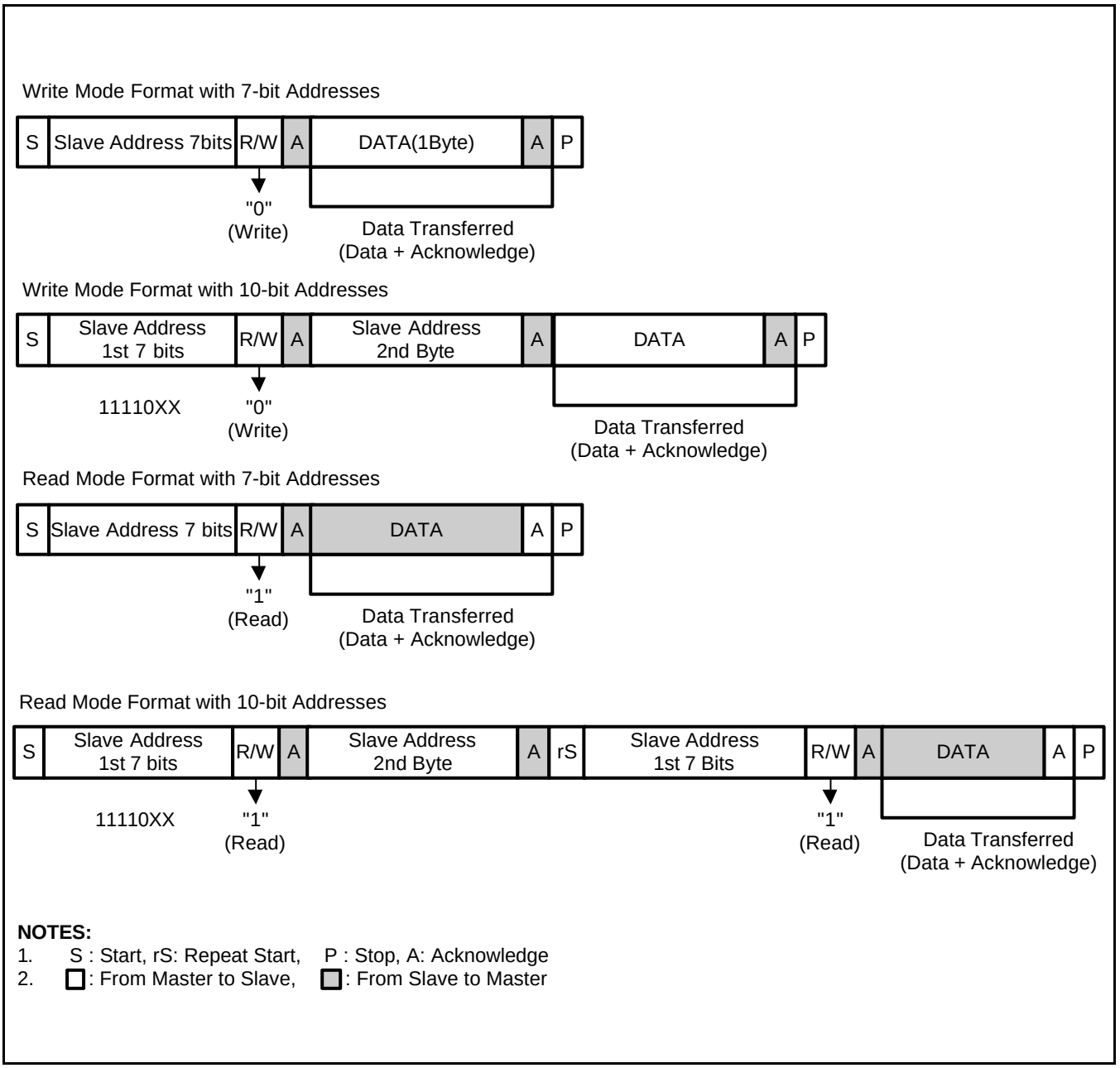


Figure 20-3. IIC-Bus Interface Data Format

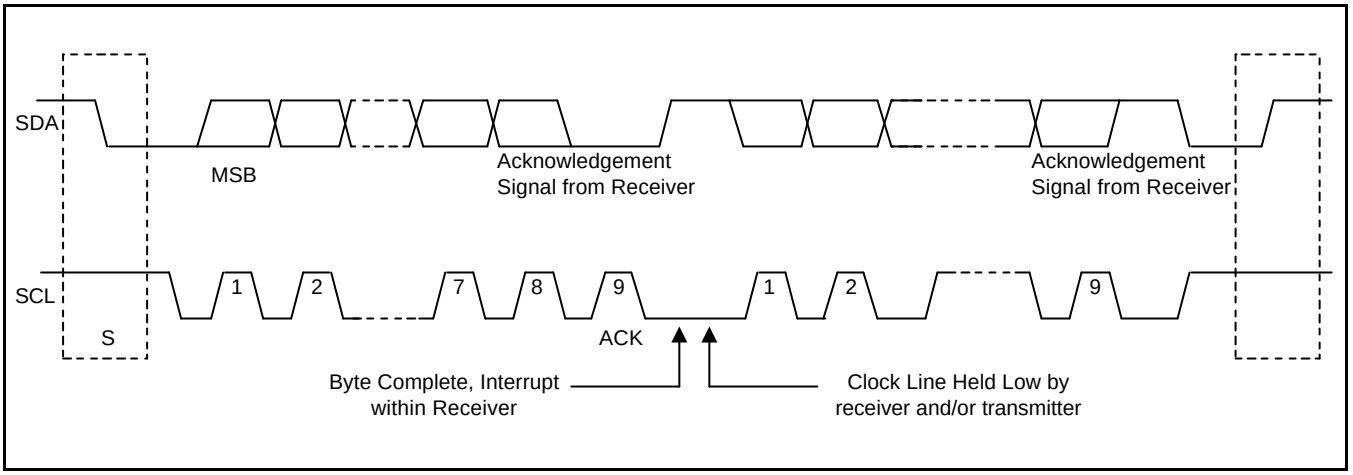


Figure 20-4. Data Transfer on the IIC-Bus

ACK SIGNAL TRANSMISSION

To complete a one-byte transfer operation, the receiver should send an ACK bit to the transmitter. The ACK pulse should occur at the ninth clock of the SCL line. Eight clocks are required for the one-byte data transfer. The master should generate the clock pulse required to transmit the ACK bit.

The transmitter should release the SDA line by making the SDA line High when the ACK clock pulse is received. The receiver should also drive the SDA line Low during the ACK clock pulse so that the SDA keeps Low during the High period of the ninth SCL pulse.

The ACK bit transmit function can be enabled or disabled by software (IICSTAT). However, the ACK pulse on the ninth clock of SCL is required to complete the one-byte data transfer operation.

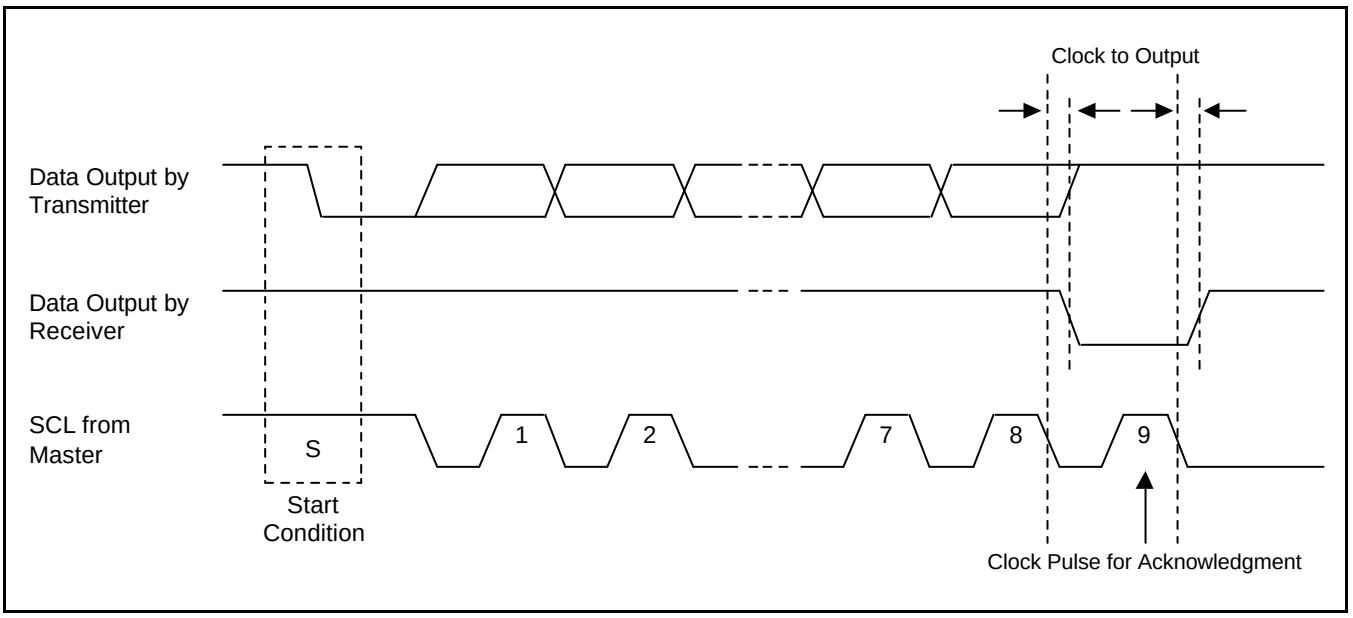


Figure 20-5. Acknowledge on the IIC-Bus

READ-WRITE OPERATION

In Transmitter mode, when the data is transferred, the IIC-bus interface will wait until IIC-bus Data Shift (IICDS) register receives a new data. Before the new data is written into the register, the SCL line will be held low, and then released after it is written. The S3C2410A should hold the interrupt to identify the completion of current data transfer. After the CPU receives the interrupt request, it should write a new data into the IICDS register, again.

In Receive mode, when a data is received, the IIC-bus interface will wait until IICDS register is read. Before the new data is read out, the SCL line will be held low and then released after it is read. The S3C2410A should hold the interrupt to identify the completion of the new data reception. After the CPU receives the interrupt request, it should read the data from the IICDS register.

BUS ARBITRATION PROCEDURES

Arbitration takes place on the SDA line to prevent the contention on the bus between two masters. If a master with a SDA High level detects the other master with a SDA active Low level, it will not initiate a data transfer because the current level on the bus does not correspond to its own. The arbitration procedure will be extended until the SDA line turns High.

However, when the masters simultaneously lower the SDA line, each master should evaluate whether or not the mastership is allocated to itself. For the purpose of evaluation, each master should detect the address bits. While each master generates the slaver address, it should also detect the address bit on the SDA line because the SDA line is likely to get Low rather than to keep High. Assume that one master generates a Low as first address bit, while the other master is maintaining High. In this case, both masters will detect Low on the bus because the Low status is superior to the High status in power. When this happens, Low (as the first bit of address) generating master will get the mastership while High (as the first bit of address) generating master should withdraw the mastership. If both masters generate Low as the first bit of address, there should be an arbitration for the second address bit, again. This arbitration will continue to the end of last address bit.

ABORT CONDITIONS

If a slave receiver cannot acknowledge the confirmation of the slave address, it should hold the level of the SDA line High. In this case, the master should generate a Stop condition and to abort the transfer.

If a master receiver is involved in the aborted transfer, it should signal the end of the slave transmit operation by canceling the generation of an ACK after the last data byte received from the slave. The slave transmitter should then release the SDA to allow a master to generate a Stop condition.

CONFIGURING IIC-BUS

To control the frequency of the serial clock (SCL), the 4-bit prescaler value can be programmed in the IICCON register. The IIC-bus interface address is stored in the IIC-bus address (IICADD) register. (By default, the IIC-bus interface address has an unknown value.)

FLOWCHARTS OF OPERATIONS IN EACH MODE

The following steps must be executed before any IIC Tx/Rx operations.

1) Write own slave address on IICADD register, if needed.
2) Set IICCON register.
   a) Enable interrupt
   b) Define SCL period
3) Set IICSTAT to enable Serial Output

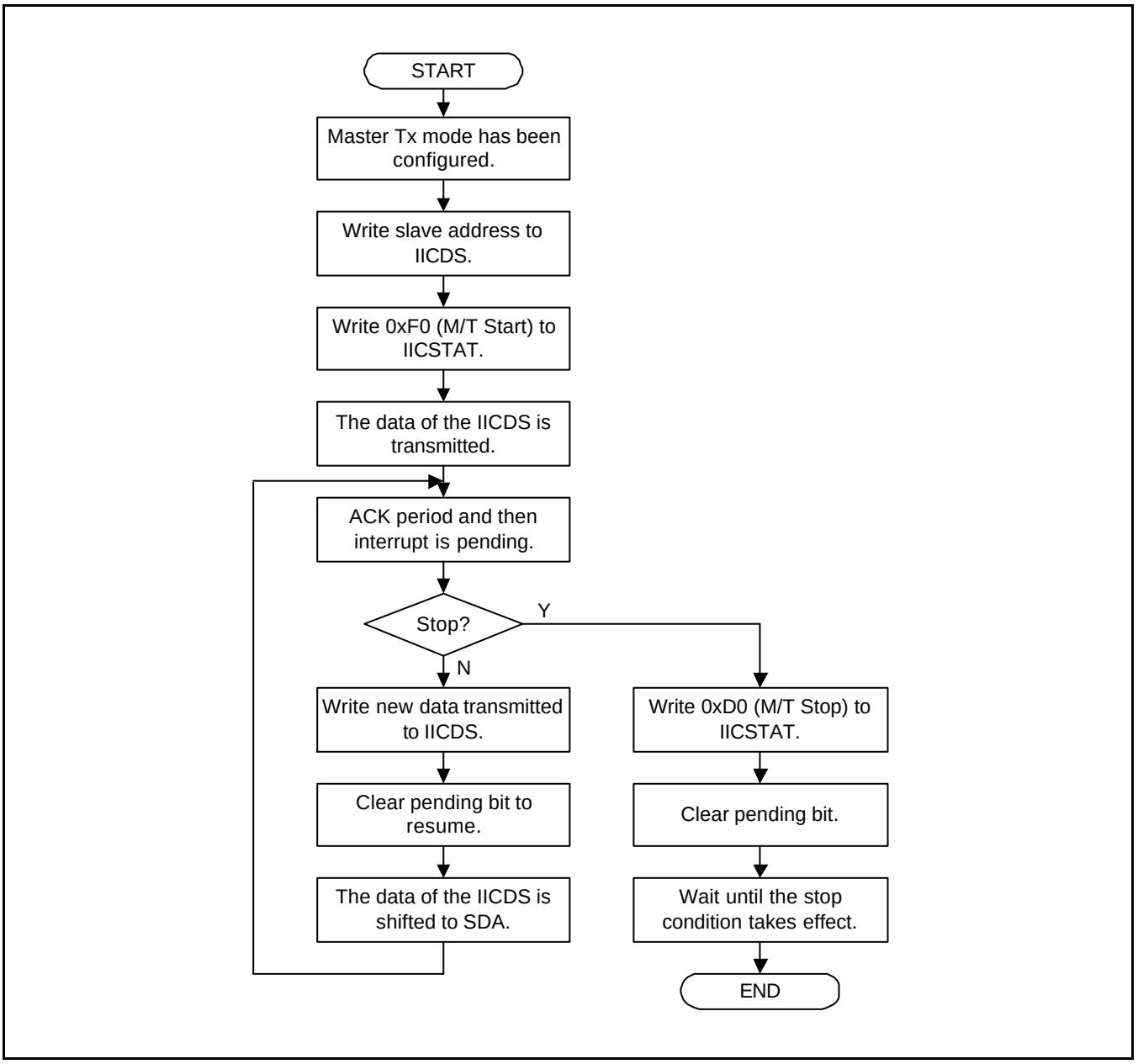


Figure 20-6. Operations for Master/Transmitter Mode

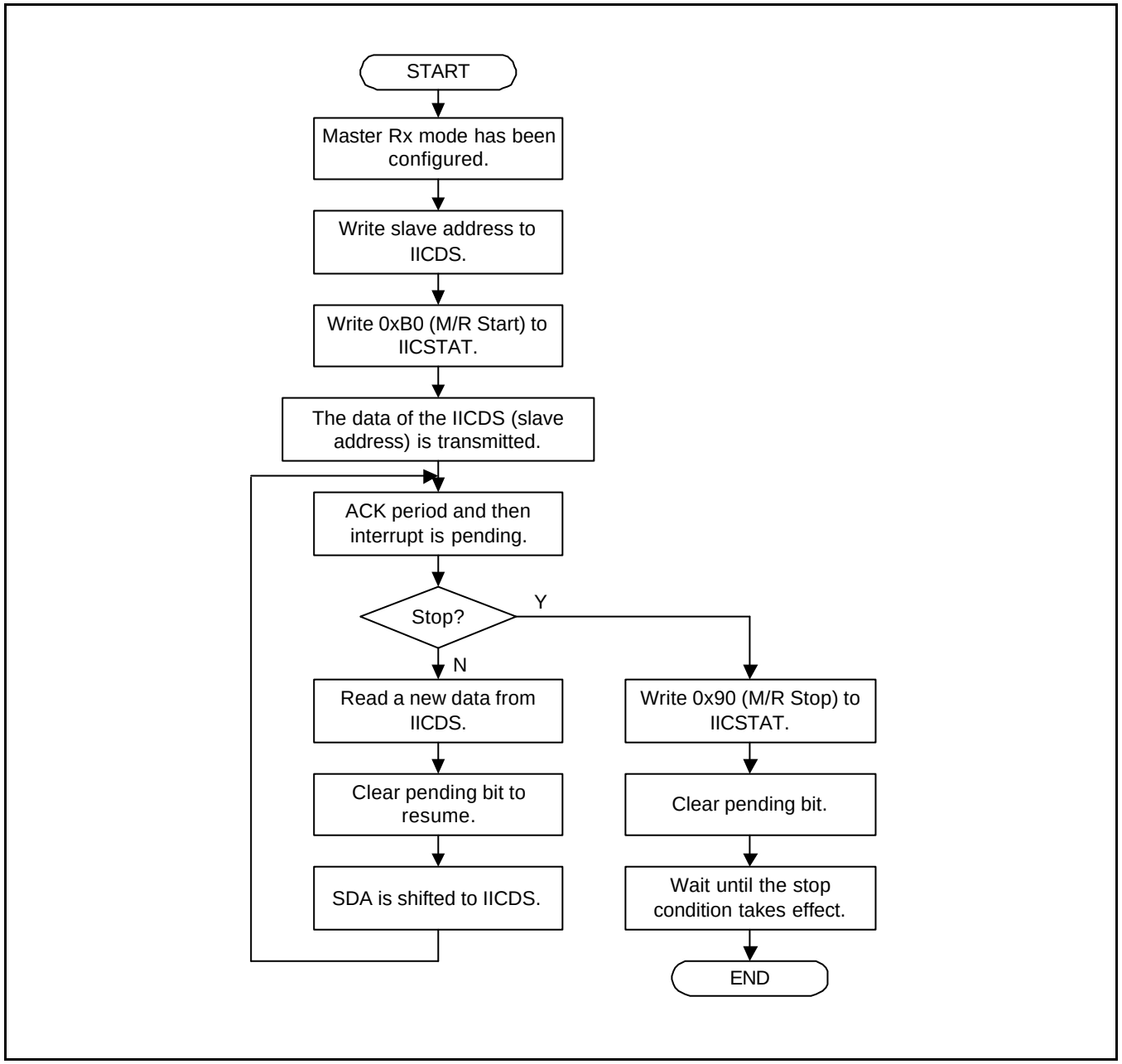


Figure 20-7. Operations for Master/Receiver Mode

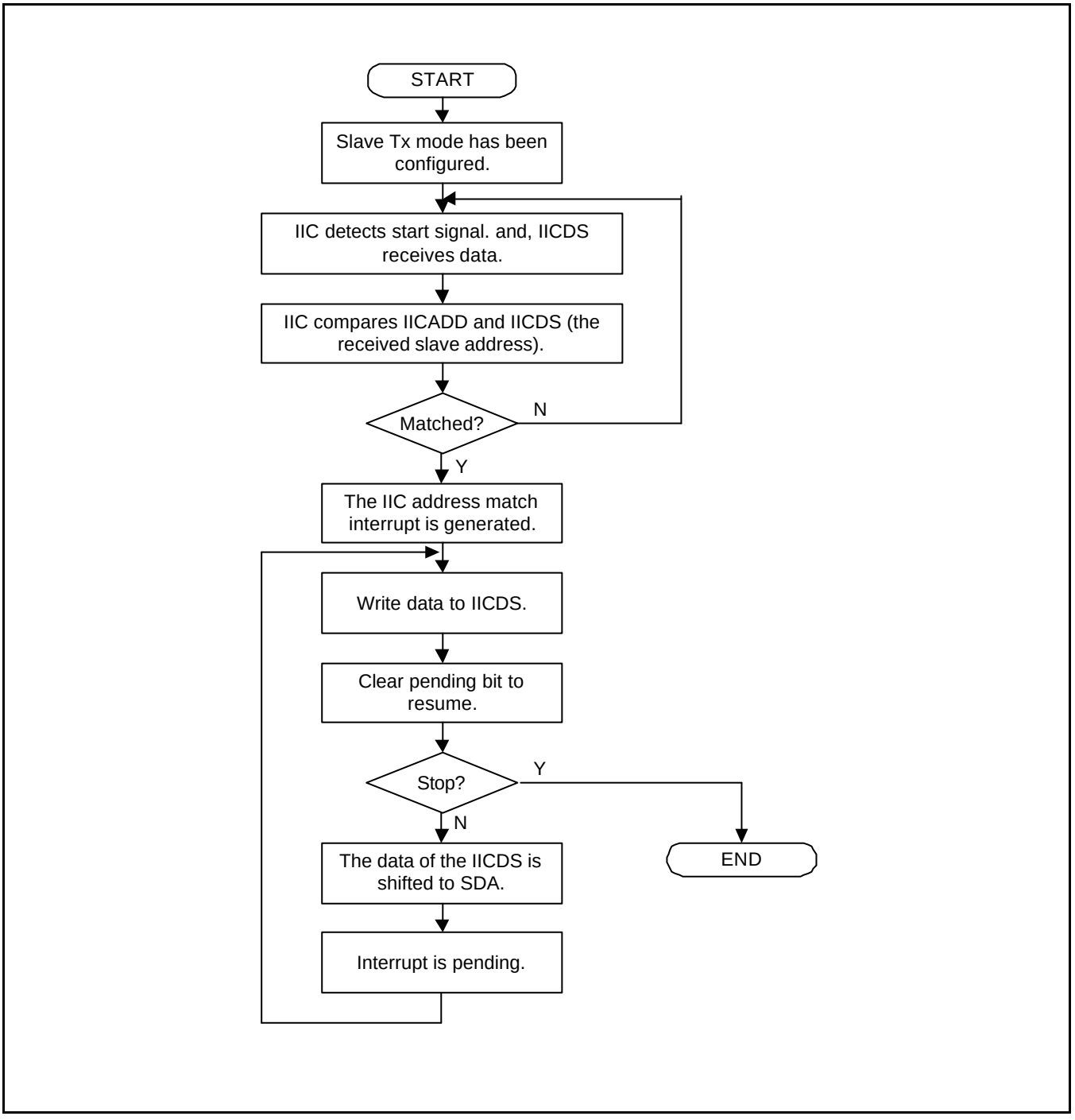


Figure 20-8. Operations for Slave/Transmitter Mode

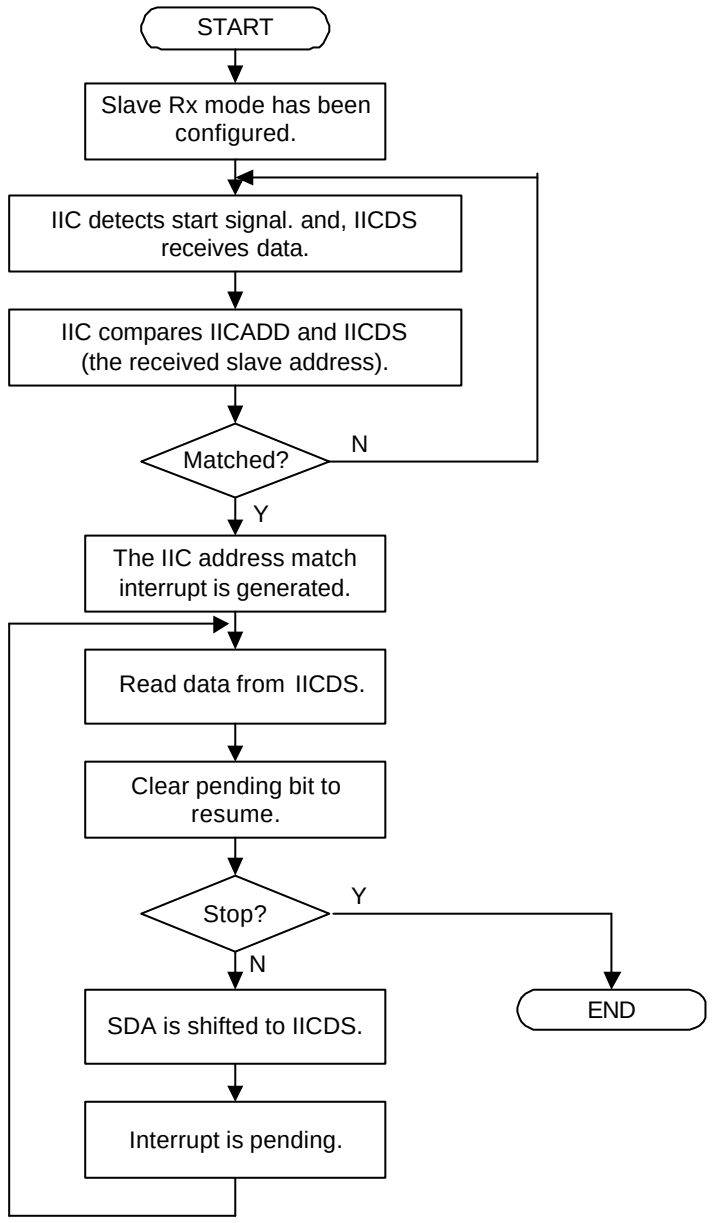


Figure 20-9. Operations for Slave/Receiver Mode

IIC-BUS INTERFACE SPECIAL REGISTERS

MULTI-MASTER IIC-BUS CONTROL (IICCON) REGISTER

Register	Address	R/W	Description	Reset Value
IICCON	0x54000000	R/W	IIC-Bus control register	0x0X

IICCON	Bit	Description	Initial State
Acknowledge generation (note 1)	[7]	IIC-bus acknowledge enable bit. 0 = Disable, 1 = Enable In Tx mode, the IICSDA is free in the ack time. In Rx mode, the IICSDA is L in the ack time.	0
Tx clock source selection	[6]	Source clock of IIC-bus transmit clock prescaler selection bit. 0 = IICCLK = f_{PCLK}/16 1 = IICCLK = f_{PCLK}/512	0
Tx/Rx Interrupt (note 5)	[5]	IIC-Bus Tx/Rx interrupt enable/disable bit. 0 = Disable, 1 = Enable	0
Interrupt pending flag (note 2), (note 3)	[4]	IIC-bus Tx/Rx interrupt pending flag. This bit cannot be written to 1. When this bit is read as 1, the IICSCL is tied to L and the IIC is stopped. To resume the operation, clear this bit as 0. 0 = 1) No interrupt pending (when read). 2) Clear pending condition & Resume the operation (when write). 1 = 1) Interrupt is pending (when read) 2) N/A (when write)	0
Transmit clock value (note 4)	[3:0]	IIC-Bus transmit clock prescaler. IIC-Bus transmit clock frequency is determined by this 4-bit prescaler value, according to the following formula: Tx clock = IICCLK/(IICCON[3:0]+1).	Undefined

NOTES:

1. Interfacing with EEPROM, the ack generation may be disabled before reading the last data in order to generate the STOP condition in Rx mode.
2. An IIC-bus interrupt occurs 1) when a 1-byte transmit or receive operation is completed, 2) when a general call or a slave address match occurs, or 3) if bus arbitration fails.
3. To adjust the setup time of IICSDA before IISSCL rising edge, IICDS has to be written before clearing the IIC interrupt pending bit.
4. IICCLK is determined by IICCON[6].
   Tx clock can vary by SCL transition time.
   When IICCON[6]=0, IICCON[3:0]=0x0 or 0x1 is not available.
5. If the IICON[5]=0, IICON[4] does not operate correctly.
   So, It is recommended that you should set IICCON[5]=1, although you does not use the IIC interrupt.

MULTI-MASTER IIC-BUS CONTROL/STATUS (IICSTAT) REGISTER

Register	Address	R/W	Description	Reset Value
IICSTAT	0x54000004	R/W	IIC-Bus control/status register	0x0

IICSTAT	Bit	Description	Initial State
Mode selection	[7:6]	IIC-bus master/slave Tx/Rx mode select bits. 00: Slave receive mode 01: Slave transmit mode 10: Master receive mode 11: Master transmit mode	00
Busy signal status / START STOP condition	[5]	IIC-Bus busy signal status bit. 0 = read) Not busy (when read) write) STOP signal generation 1 = read) Busy (when read) write) START signal generation. The data in IICDS will be transferred automatically just after the start signal.	0
Serial output	[4]	IIC-bus data output enable/disable bit. 0 = Disable Rx/Tx, 1 = Enable Rx/Tx	0
Arbitration status flag	[3]	IIC-bus arbitration procedure status flag bit. 0 = Bus arbitration successful 1 = Bus arbitration failed during serial I/O	0
Address-as-slave status flag	[2]	IIC-bus address-as-slave status flag bit. 0 = Cleared when START/STOP condition was detected 1 = Received slave address matches the address value in the IICADD	0
Address zero status flag	[1]	IIC-bus address zero status flag bit. 0 = Cleared when START/STOP condition was detected. 1 = Received slave address is 00000000b.	0
Last-received bit status flag	[0]	IIC-bus last-received bit status flag bit. 0 = Last-received bit is 0 (ACK was received). 1 = Last-received bit is 1 (ACK was not received).	0

MULTI-MASTER IIC-BUS ADDRESS (IICADD) REGISTER

Register	Address	R/W	Description	Reset Value
IICADD	0x54000008	R/W	IIC-Bus address register	0xXX

IICADD	Bit	Description	Initial State
Slave address	[7:0]	7-bit slave address, latched from the IIC-bus. When serial output enable = 0 in the IICSTAT, IICADD is write-enabled. The IICADD value can be read any time, regardless of the current serial output enable bit (IICSTAT) setting. Slave address = [7:1] Not mapped = [0]	XXXXXXXX

MULTI-MASTER IIC-BUS TRANSMIT/RECEIVE DATA SHIFT (IICDS) REGISTER

Register	Address	R/W	Description	Reset Value
IICDS	0x5400000C	R/W	IIC-Bus transmit/receive data shift register	0xXX

IICDS	Bit	Description	Initial State
Data shift	[7:0]	8-bit data shift register for IIC-bus Tx/Rx operation. When serial output enable = 1 in the IICSTAT, IICDS is write-enabled. The IICDS value can be read any time, regardless of the current serial output enable bit (IICSTAT) setting.	XXXXXXXX

NOTES

21 IIS-BUS INTERFACE

OVERVIEW

Currently, many digital audio systems are attracting the consumers on the market, in the form of compact discs, digital audio tapes, digital sound processors, and digital TV sound. The S3C2410A Inter-IC Sound (IIS) bus interface can be used to implement a CODEC interface to an external 8/16-bit stereo audio CODEC IC for mini-disc and portable applications. The IIS bus interface supports both IIS bus data format and MSB-justified data format. The interface provides DMA transfer mode for FIFO access instead of an interrupt. It can transmit and receive data simultaneously as well as transmit or receive data alternatively at a time.

BLOCK DIAGRAM

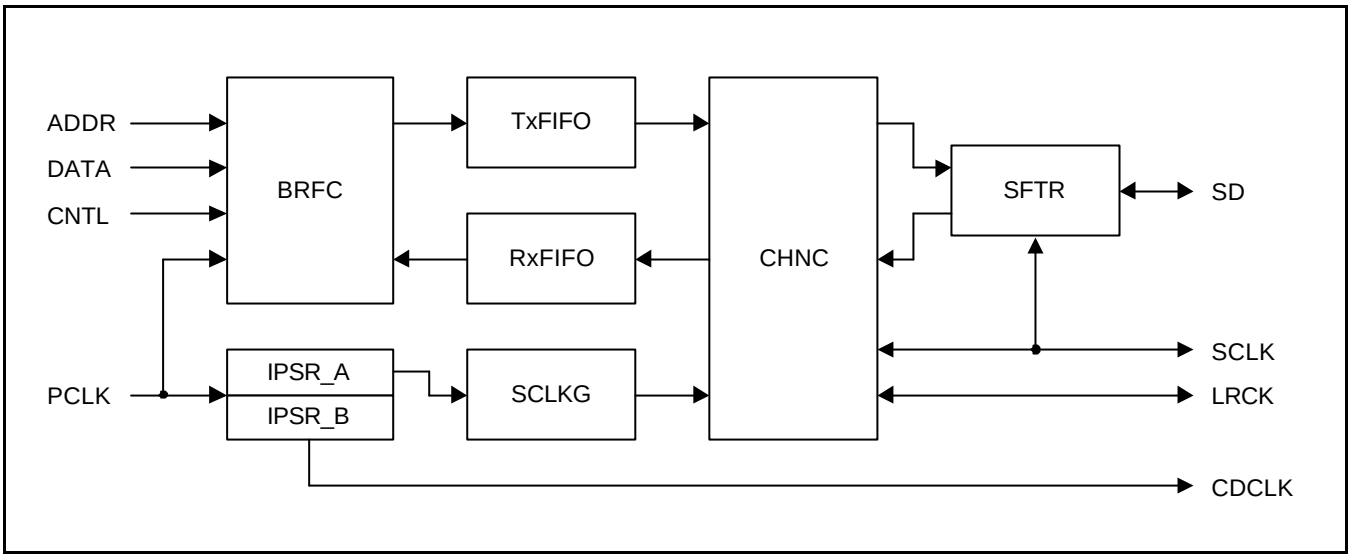


Figure 21-1. IIS-Bus Block Diagram

FUNCTIONAL DESCRIPTIONS

Bus interface, register bank, and state machine (BRFC): Bus interface logic and FIFO access are controlled by the state machine.

5-bit dual prescaler (IPSR): One prescaler is used as the master clock generator of the IIS bus interface and the other is used as the external CODEC clock generator.

64-byte FIFOs (TxFIFO and RxFIFO): In transmit data transfer, data are written to TxFIFO, and, in the receive data transfer, data are read from RxFIFO.

Master IISCLK generator (SCLKG): In master mode, serial bit clock is generated from the master clock.

Channel generator and state machine (CHNC): IISCLK and IISLRCK are generated and controlled by the channel state machine.

16-bit shift register (SFTR): Parallel data is shifted to serial data output in the transmit mode, and serial data input is shifted to parallel data in the receive mode.

TRANSMIT OR RECEIVE ONLY MODE

Normal Transfer

IIS control register has FIFO ready flag bits for transmit and receive FIFOs. When FIFO is ready to transmit data, the FIFO ready flag is set to '1' if transmit FIFO is not empty.
If transmit FIFO is empty, FIFO ready flag is set to '0'. When receive FIFO is not full, the FIFO ready flag for receive FIFO is set to '1' ; it indicates that FIFO is ready to receive data. If receive FIFO is full, FIFO ready flag is set to '0'. These flags can determine the time that CPU is to write or read FIFOs. Serial data can be transmitted or received while the CPU is accessing transmit and receive FIFOs in this way.

DMA Transfer

In this mode, transmit or receive FIFO is accessible by the DMA controller. DMA service request in transmit or receive mode is made by the FIFO ready flag automatically.

Transmit and Receive Mode

In this mode, IIS bus interface can transmit and receive data simultaneously.

AUDIO SERIAL INTERFACE FORMAT

IIS-BUS FORMAT

The IIS bus has four lines including serial data input (IISDI), serial data output (IISDO), left/right channel select (IISLRCK), and serial bit clock (IISCLK); the device generating IISLRCK and IISCLK is the master.

Serial data is transmitted in 2's complement with the MSB first. The MSB is transmitted first because the transmitter and receiver may have different word lengths. The transmitter does not have to know how many bits the receiver can handle, nor does the receiver need to know how many bits are being transmitted.

When the system word length is greater than the transmitter word length, the word is truncated (least significant data bits are set to '0') for data transmission. If the receiver gets more bits than its word length, the bits after the LSB are ignored. On the other hand, if the receiver gets fewer bits than its word length, the missing bits are set to zero internally. And therefore, the MSB has a fixed position, whereas the position of the LSB depends on the word length. The transmitter sends the MSB of the next word at one clock period whenever the IISLRCK is changed.

Serial data sent by the transmitter may be synchronized with either the trailing (HIGH to LOW) or the leading (LOW to HIGH) edge of the clock signal. However, the serial data must be latched into the receiver on the leading edge of the serial clock signal, and so there are some restrictions when transmitting data that is synchronized with the leading edge.

The LR channel select line indicates the channel being transmitted. IISLRCK may be changed either on a trailing or leading edge of the serial clock, but it does not need to be symmetrical. In the slave, this signal is latched on the leading edge of the clock signal. The IISLRCK line changes one clock period before the MSB is transmitted. This allows the slave transmitter to derive synchronous timing of the serial data that will be set up for transmission. Furthermore, it enables the receiver to store the previous word and clear the input for the next word.

MSB (LEFT) JUSTIFIED

MSB / left justified bus format is the same as IIS bus format architecturally. Only, different from the IIS bus format, the MSB justified format realizes that the transmitter always sends the MSB of the next word whenever the IISLRCK is changed.

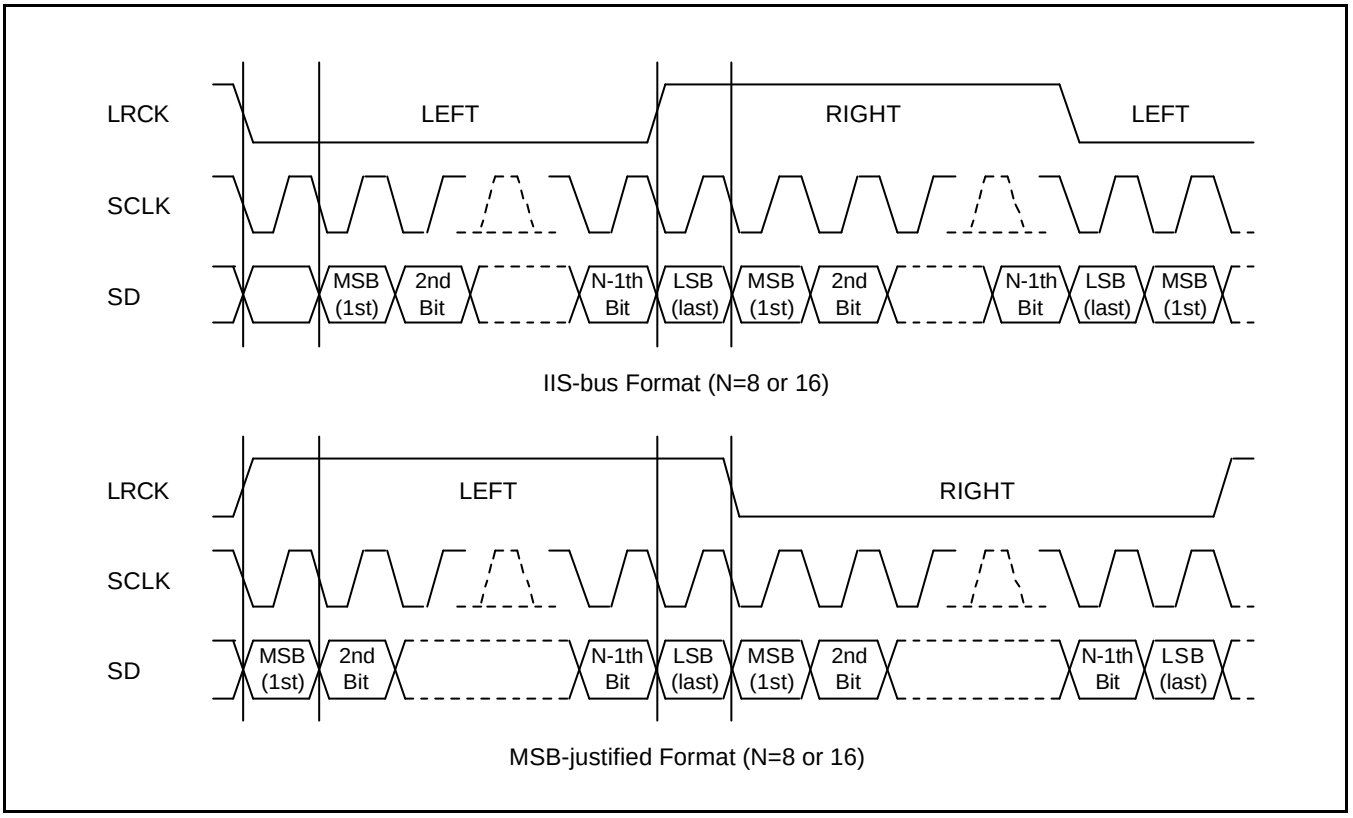


Figure 21-2. IIS-Bus and MSB (Left)-justified Data Interface Formats

SAMPLING FREQUENCY AND MASTER CLOCK

Master clock frequency (PCLK) can be selected by sampling frequency as shown in Table 21-1. Because PCLK is made by IIS prescaler, the prescaler value and PCLK type (256 or 384fs) should be determined properly. Serial bit clock frequency type (16/32/48fs) can be selected by the serial bit per channel and PCLK as shown in Table 21-2.

Table 21-1. CODEC clock (CODECLK = 256 or 384fs)

IISLRCK (fs)	8.000 kHz	11.025 kHz	16.000 kHz	22.050 kHz	32.000 kHz	44.100 kHz	48.000 kHz	64.000 kHz	88.200 kHz	96.000 kHz
CODECLK (MHz)	256fs									
	2.0480	2.8224	4.0960	5.6448	8.1920	11.2896	12.2880	16.3840	22.5792	24.5760
	384fs									
	3.0720	4.2336	6.1440	8.4672	12.2880	16.9344	18.4320	24.5760	33.8688	36.8640

Table 21-2 Usable Serial Bit Clock Frequency (IISCLK = 16 or 32 or 48fs)

Serial bit per channel	8-bit	16-bit
Serial clock frequency (IISCLK)		
@CODECLK = 256fs	16fs, 32fs	32fs
@CODECLK = 384fs	16fs, 32fs, 48fs	32fs, 48fs

IIS-BUS INTERFACE SPECIAL REGISTERS

IIS CONTROL (IISCON) REGISTER

Register	Address	R/W	Description	Reset Value
IISCON	0x55000000 (Li/HW, Li/W, Bi/W) 0x55000002 (Bi/HW)	R/W	IIS control register	0x100

IISCON	Bit	Description	Initial State
Left/Right channel index (Read only)	[8]	0 = Left 1 = Right	1
Transmit FIFO ready flag (Read only)	[7]	0 = empty 1 = not empty	0
Receive FIFO ready flag (Read only)	[6]	0 = full 1 = not full	0
Transmit DMA service request	[5]	0 = Disable 1 = Enable	0
Receive DMA service request	[4]	0 = Disable 1 = Enable	0
Transmit channel idle command	[3]	In Idle state the IISLRCK is inactive (Pause Tx). 0 = Not idle 1 = Idle	0
Receive channel idle command	[2]	In Idle state the IISLRCK is inactive (Pause Rx). 0 = Not idle 1 = Idle	0
IIS prescaler	[1]	0 = Disable 1 = Enable	0
IIS interface	[0]	0 = Disable (stop) 1 = Enable (start)	0

NOTES:

1. The IISCON register is accessible for each byte, halfword and word unit using STRB/STRH/STR and LDRB/LDRH/LDR instructions or char/short int/int type pointer in Little/Big endian mode.
2. (Li/HW/W): Little/HalfWord/Word
   (Bi/HW/W): Big/HalfWord/Word

IIS MODE REGISTER (IISMOD) REGISTER

Register	Address	R/W	Description	Reset Value
IISMOD	0x55000004 (Li/W, Li/HW, Bi/W) 0x55000006 (Bi/HW)	R/W	IIS mode register	0x0

IISMOD	Bit	Description	Initial State
Master/slave mode select	[8]	0 = Master mode (IISLRCK and IISCLK are output mode). 1 = Slave mode (IISLRCK and IISCLK are input mode).	0
Transmit/receive mode select	[7:6]	00 = No transfer 01 = Receive mode 10 = Transmit mode 11 = Transmit and receive mode	00
Active level of left/right channel	[5]	0 = Low for left channel (High for right channel) 1 = High for left channel (Low for right channel)	0
Serial interface format	[4]	0 = IIS compatible format 1 = MSB (Left)-justified format	0
Serial data bit per channel	[3]	0 = 8-bit 1 = 16-bit	0
Master clock frequency select	[2]	0 = 256fs 1 = 384fs (fs: sampling frequency)	0
Serial bit clock frequency select	[1:0]	00 = 16fs 01 = 32fs 10 = 48fs 11 = N/A	00

NOTES:

1. The IISMOD register is accessible for each halfword and wordunit using STRH/STR and LDRH/LDR instructions or short int/int type pointer in Little/Big endian mode.
2. (Li/HW/W): Little/HalfWord/Word.
   (Bi/HW/W): Big/HalfWord/Word.

IIS PRESCALER (IISPSR) REGISTER

Register	Address	R/W	Description	Reset Value
IISPSR	0x55000008 (Li/HW, Li/W, Bi/W) 0x5500000A (Bi/HW)	R/W	IIS prescaler register	0x0

IISPSR	Bit	Description	Initial State
Prescaler control A	[9:5]	Data value: 0 ~ 31 Note: Prescaler A makes the master clock that is used the internal block and division factor is N+1.	00000
Prescaler control B	[4:0]	Data value: 0 ~ 31 Note: Prescaler B makes the master clock that is used the external block and division factor is N+1.	00000

NOTES:

1. The IISPSR register is accessible for each byte, halfword and word unit using STRB/STRH/STR and LDRB/LDRH/LDR instructions or char/short int/int type pointer in Little/Big endian mode.
2. (Li/HW/W) : Little/HalfWord/Word.
   (Bi/HW/W) : Big/HalfWord/Word.

IIS FIFO CONTROL (IISFCON) REGISTER

Register	Address	R/W	Description	Reset Value
IISFCON	0x5500000C (Li/HW, Li/W, Bi/W) 0x5500000E (Bi/HW)	R/W	IIS FIFO interface register	0x0

IISFCON	Bit	Description	Initial State
Transmit FIFO access mode select	[15]	0 = Normal 1 = DMA	0
Receive FIFO access mode select	[14]	0 = Normal 1 = DMA	0
Transmit FIFO	[13]	0 = Disable 1 = Enable	0
Receive FIFO	[12]	0 = Disable 1 = Enable	0
Transmit FIFO data count (Read only)	[11:6]	Data count value = 0 ~ 32	000000
Receive FIFO data count (Read only)	[5:0]	Data count value = 0 ~ 32	000000

NOTES:
1. The IISFCON register is accessible for each halfword and word unit using STRH/STR and LDRH/LDR instructions or short int/int type pointer in Little/Big endian mode.
2. (Li/HW/W): Little/HalfWord/Word.
   (Bi/HW/W): Big/HalfWord/Word.

IIS FIFO (IISFIFO) REGISTER

IIS bus interface contains two 64-byte FIFO for the transmit and receive mode. Each FIFO has 16-width and 32-depth form, which allows the FIFO to handles data for each halfword unit regardless of valid data size. Transmit and receive FIFO access is performed through FIFO entry; the address of FENTRY is 0x55000010.

Register	Address	R/W	Description	Reset Value
IISFIFO	0x55000010(Li/HW) 0x55000012(Bi/HW)	R/W	IIS FIFO register	0x0

IISFIFO	Bit	Description	Initial State
FENTRY	[15:0]	Transmit/Receive data for IIS	0x0

NOTES:
1. The IISFIFO register is accessible for each halfword and word unit using STRH and LDRH instructions or short int type pointer in Little/Big endian mode.
2. (Li/HW): Little/HalfWord.
   (Bi/HW): Big/HalfWord.

22 SPI INTERFACE

OVERVIEW

The S3C2410A Serial Peripheral Interface (SPI) can interface the serial data transfer. The S3C2410A includes two SPI, each of which has two 8-bit shift registers for transmission and receiving, respectively. During an SPI transfer, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). 8-bit serial data at a frequency is determined by its corresponding control register settings. If you only want to transmit, received data can be dummy. Otherwise, if you only want to receive, you should transmit dummy '1' data.

There are 4 I/O pin signals associated with SPI transfers: the SCK (SPICLK0,1), the MISO (SPIMISO0,1) data line, the MOSI (SPIMOSI0,1) data line, and the active low /SS (nSS0,1) pin (input).

FEATURES

— SPI Protocol (ver. 2.11) compatible

— 8-bit Shift Register for transmit

— 8-bit Shift Register for receive

— 8-bit Prescaler logic

— Polling, Interrupt, and DMA transfer mode

BLOCK DIAGRAM

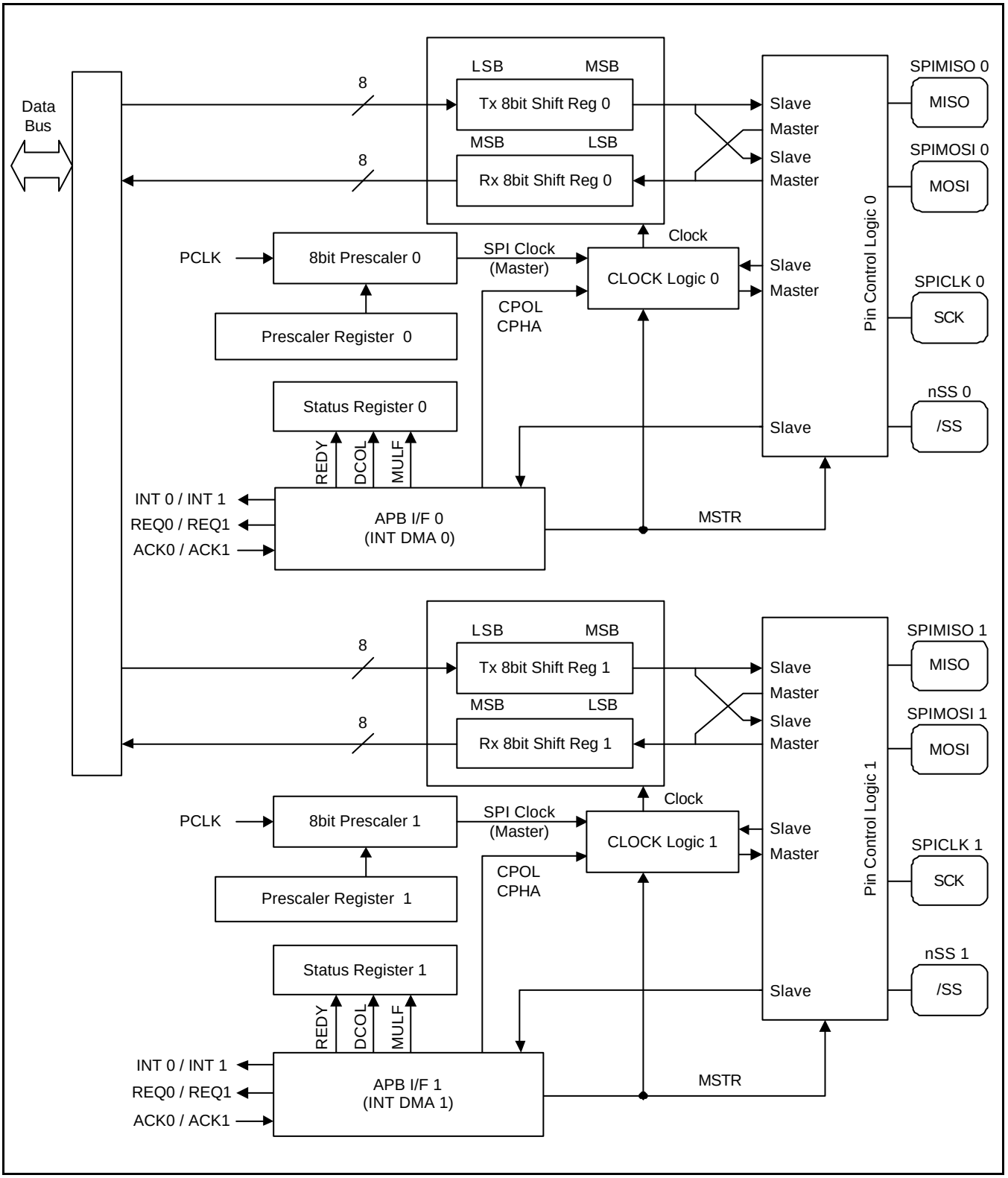


Figure 22-1. SPI Block Diagram

SPI OPERATION

Using the SPI interface, the S3C2410A can send/receive 8 –bit data simultaneously with an external device. A serial clock line is synchronized with the two data lines for shifting and sampling of the information. When the SPI is the master, transmission frequency can be controlled by setting the appropriate bit to SPPREn register. You can modify its frequency to adjust the baud rate data register value. When the SPI is a slave, other master supplies the clock. When the programmer writes byte data to SPTDATn register, SPI transmit/receive operation will start simultaneously. In some cases, nSS should be activated before writing byte data to SPTDATn.

Programming Procedure

When a byte data is written into the SPTDATn register, SPI starts to transmit if ENSCK and MSTR of SPCONn register are set. You can use a typical programming procedure to operate an SPI card.

To program the SPI modules, follow these basic steps:

1. Set Baud Rate Prescaler Register (SPPREn).
2. Set SPCONn to configure properly the SPI module.
3. Write data 0xFF to SPTDATn 10 times in order to initialize MMC or SD card.
4. Set a GPIO pin, which acts as nSS, to low to activate the MMC or SD card.
5. Tx data → Check the status of Transfer Ready flag (REDY = 1), and then write data to SPTDATn.
6. Rx data(1): SPCONn's TAGD bit disable = normal mode
   → write 0xFF to SPTDATn, then confirm REDY to set, and then read data from Read Buffer.
7. Rx data(2): SPCONn's TAGD bit enable = Tx Auto Garbage Data mode
   → confirm REDY to set, and then read data from Read Buffer (then automatically start to transfer).
8. Set a GPIO pin, which acts as nSS, to high, to deactivate MMC or SD card.

SPI Transfer Format

The S3C2410A supports 4 different format to transfer the data. Figure 22-2 shows four waveforms for SPICLK..

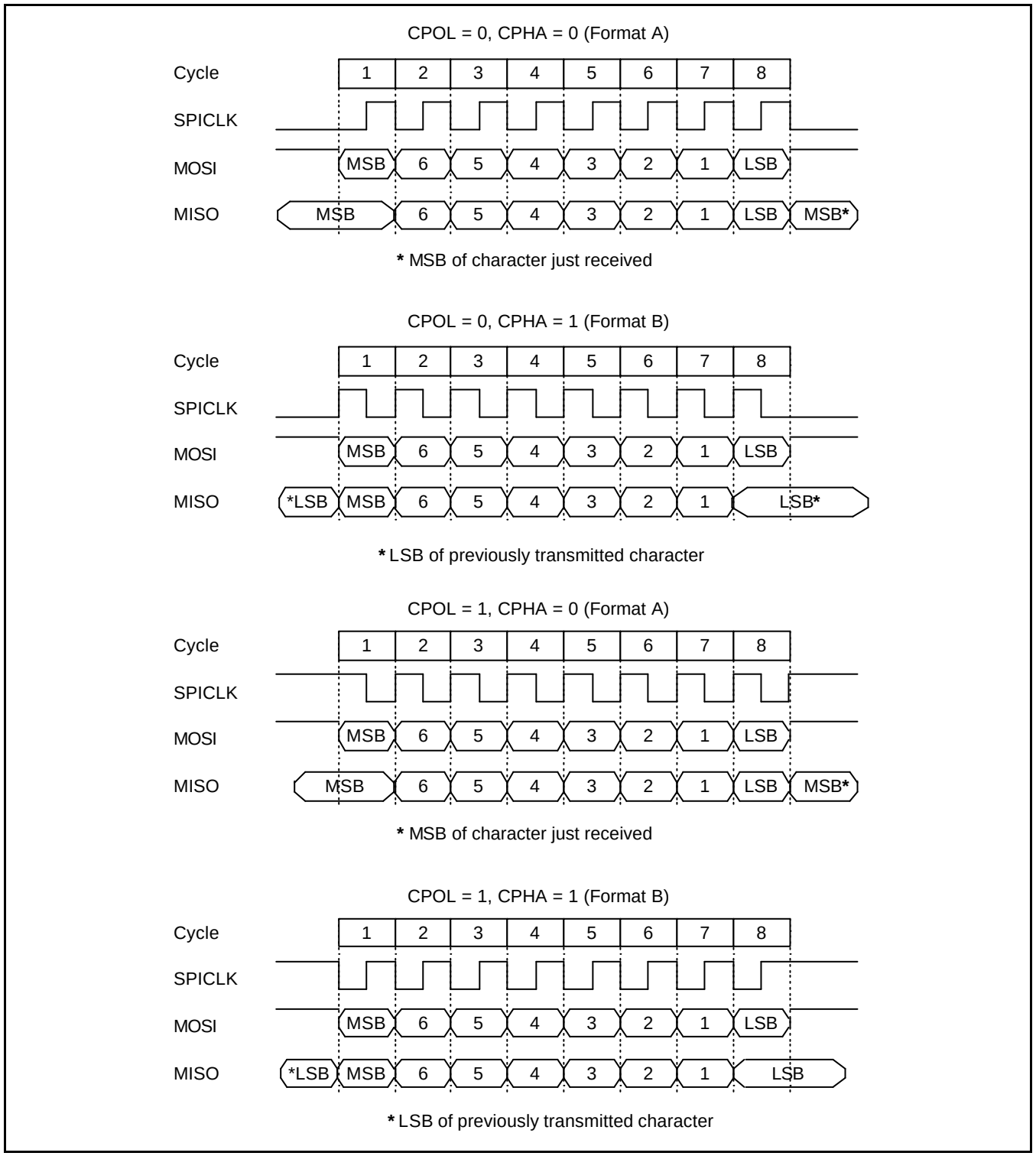


Figure 22-2. SPI Transfer Format

Transmitting Procedure by DMA

1. The SPI is configured as DMA mode.
2. DMA is configured properly.
3. The SPI requests DMA service.
4. DMA transmits 1byte data to the SPI.
5. The SPI transmits the data to card.
6. Return to Step 3 until DMA count becomes 0.
7. The SPI is configured as interrupt or polling mode with SMOD bits.

Receiving Procedure by DMA

1. The SPI is configured as DMA start with SMOD bits and setting TAGD bit.
2. DMA is configured properly.
3. The SPI receives 1byte data from card.
4. The SPI requests DMA service.
5. DMA receives the data from the SPI.
6. Write data 0xFF automatically to SPTDATn.
7. Return to Step 4 until DMA count becomes 0.
8. The SPI is configured as polling mode with SMOD bits and clearing TAGD bit.
9. If SPSTAn's REDY flag is set, then read the last byte data.

NOTE

Total received data = DMA TC values + the last data in polling mode (Step 9).
The first DMA received data is dummy and so the user can neglect it.

SPI Slave Rx Mode with Format B

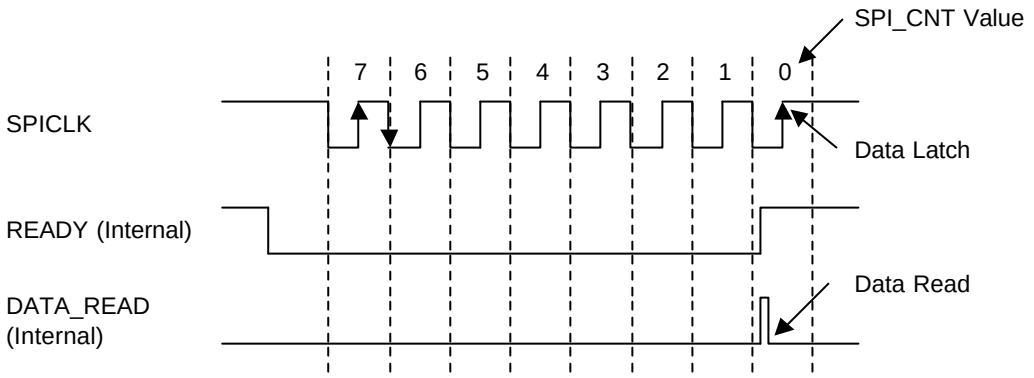
If the SPI slave Rx mode is activated and SPI format is set to format B, then SPI operation will be failed:

The READY signal, one of internal signals, becomes high before the SPI_CNT reaches 0. Therefore, in DMA mode, DATA_READ signal is generated before the last data is latched.

Guide

1) DMA mode: This mode cannot be used at SPI slave Rx mode with format B.
2) Polling mode:
   DATA_READ signal should be delayed by 1phase of SPICLK at SPI slave Rx mode with format B.
3) Interrupt mode:
   DATA_READ signal should be delayed 1phase of SPICLK at SPI slave Rx mode with format B.

SPI Data Receive Function Error (Slave/Receive/Format B Mode)



SPI SPECIAL REGISTERS

SPI CONTROL REGISTER

Register	Address	R/W	Description	Reset Value
SPCON0	0x59000000	R/W	SPI channel 0 control register	0x00
SPCON1	0x59000020	R/W	SPI channel 1 control register	0x00

SPCONn	Bit	Description	Initial State
SPI Mode Select (SMOD)	[6:5]	Determine how and by what SPTDAT is read/written. 00 = polling mode, 01 = interrupt mode 10 = DMA mode, 11 = reserved	00
SCK Enable (ENSCK)	[4]	Determine whether you want SCK enable or not (for only master). 0 = disable, 1 = enable	0
Master/Slave Select (MSTR)	[3]	Determine the desired mode (master or slave). 0 = slave, 1 = master **NOTE:** In slave mode, there should be set up time for master to initiate Tx/Rx.	0
Clock Polarity Select (CPOL)	[2]	Determine an active high or active low clock. 0 = active high, 1 = active low	0
Clock Phase Select (CPHA)	[1]	Select one of two fundamentally different transfer formats. 0 = format A, 1 = format B	0
Tx Auto Garbage Data mode enable (TAGD)	[0]	Decide whether the receiving data only needs or not. 0 = normal mode, 1 = Tx auto garbage data mode **NOTE:** In normal mode, if you only want to receive data, you should transmit dummy 0xFF data.	0

SPI STATUS REGISTER

Register	Address	R/W	Description	Reset Value
SPSTA0	0x59000004	R	SPI channel 0 status register	0x01
SPSTA1	0x59000024	R	SPI channel 1 status register	0x01

SPSTAn	Bit	Description	Initial State
Reserved	[7:3]		
Data Collision Error Flag (DCOL)	[2]	This flag is set if the SPTDATn is written or the SPRDATn is read while a transfer is in progress and cleared by reading the SPSTAn. 0 = not detect, 1 = collision error detect	0
Multi Master Error Flag (MULF)	[1]	This flag is set if the nSS signal goes to active low while the SPI is configured as a master, and SPPINn's ENMUL bit is multi master errors detect mode. MULF is cleared by reading SPSTAn. 0 = not detect, 1 = multi master error detect	0
Transfer Ready Flag (REDY)	[0]	This bit indicates that SPTDATn or SPRDATn is ready to transmit or receive. This flag is automatically cleared by writing data to SPTDATn. 0 = not ready, 1 = data Tx/Rx ready	1

SPI PIN CONTROL REGISTER

When the SPI system is enabled, the direction of pins, except nSS pin, is controlled by MSTR bit of SPCONn register. The direction of nSS pin is always input.
When the SPI is a master, nSS pin is used to check multi-master error, provided the SPPIN's ENMUL bit is active, and another GPIO should be used to select a slave.
If the SPI is configured as a slave, the nSS pin is used to select SPI as a slave by one master.

Register	Address	R/W	Description	Reset Value
SPPIN0	0x59000008	R/W	SPI channel 0 pin control register	0x02
SPPIN1	0x59000028	R/W	SPI channel 1 pin control register	0x02

SPPINn	Bit	Description	Initial State
Reserved	[7:3]		
Multi Master error detect Enable (ENMUL)	[2]	The /SS pin is used as an input to detect multi master error when the SPI system is a master. 0 = disable (general purpose) 1 = multi master error detect enable	0
Reserved	[1]	This bit should be "1".	1
Master Out Keep (KEEP)	[0]	Determine MOSI drive or release when 1byte transmit is completed (only master). 0 = release, 1 = drive the previous level	0

The SPIMISO (MISO) and SPIMOSI (MOSI) data pins are used for transmitting and receiving serial data. When the SPI is configured as a master, SPIMISO (MISO) is the master data input line, SPIMOSI (MOSI) is the master data output line, and SPICLK (SCK) is the clock output line. When the SPI becomes a slave, these pins perform reversed roles. In a multiple-master system, SPICLK (SCK) pins, SPIMOSI (MOSI) pins, and SPIMISO (MISO) pins are tied to configure a group respectively.

A master SPI can experience a multi master error, when other SPI device working as a master selects the S3C2410 SPI as a slave. When this error is detected, the following actions are taken immediately. But you must previously set SPPINn's ENMUL bit if you want to detect this error.

1. The SPCONn's MSTR bit is forced to 0 to operate slave mode.
2. The SPSTAn's MULF flag is set, and an SPI interrupt is generated.

SPI Baud Rate Prescaler Register

Register	Address	R/W	Description	Reset Value
SPPRE0	0x5900000C	R/W	SPI cannel 0 baud rate prescaler register	0x00
SPPRE1	0x5900002C	R/W	SPI cannel 1 baud rate prescaler register	0x00

SPPREn	Bit	Description	Initial State
Prescaler Value	[7:0]	Determine SPI clock rate as above equation. Baud rate = PCLK / 2 / (Prescaler value + 1)	0x00

NOTE: Baud rate should be less than 25 MHz.

SPI Tx Data Register

Register	Address	R/W	Description	Reset Value
SPTDAT0	0x59000010	R/W	SPI channel 0 Tx data register	0x00
SPTDAT1	0x59000030	R/W	SPI channel 1 Tx data register	0x00

SPTDATn	Bit	Description	Initial State
Tx Data Register	[7:0]	This field contains the data to be transmitted over the SPI channel.	0x00

SPI Rx Data Register

Register	Address	R/W	Description	Reset Value
SPRDAT0	0x59000014	R	SPI channel 0 Rx data register	0x00
SPRDAT1	0x59000034	R	SPI channel 1 Rx data register	0x00

SPRDATn	Bit	Description	Initial State
Rx Data Register	[7:0]	This field contains the data to be received over the SPI channel.	0x00

23 BUS PRIORITIES

OVERVIEW

The bus arbitration logic determines the priorities of bus masters. It supports a combination of rotation priority mode and fixed priority mode.

BUS PRIORITY MAP

The S3C2410A holds eleven bus masters including SDRAM refresh controller, LCD_DMA, DMA0, DMA1, DMA2, DMA3, USB_HOST_DMA, EXT_BUS_MASTER, Test interface controller (TIC), and ARM920T. The following list shows the priorities among these bus masters after a reset:

1. SDRAM refresh controller
2. LCD_DMA
3. DMA0
4. DMA1
5. DMA2
6. DMA3
7. USB host DMA
8. External bus master
9. TIC
10. ARM920T
11. Reserved

Among those bus masters, four DMAs operate under the rotation priority, while others run under the fixed priority.

NOTES

24 ELECTRICAL DATA

ABSOLUTE MAXIMUM RATINGS

Table 24-1. Absolute Maximum Rating

Parameter	Symbol	Rating (200MHz / 266MHz)		Unit
DC Supply Voltage	V_{DDi}	1.8V / 2.0V V_{DD}	2.7	V
	V_{DDRTC}	1.8V V_{DD}	2.7	
	V_{DDIO}	3.3V V_{DD}	3.8	
DC Input Voltage	V_{IN}	3.3V Input buffer	3.8	
		3.3V Interface / 5V Tolerant input buffer	6.5	
DC Output Voltage	V_{OUT}	3.3V Output buffer	3.8	
DC Input (Latch-up) Current	I_{IN}	± 200		mA
Storage Temperature	T_{STG}	– 65 to 150		°C

RECOMMENDED OPERATING CONDITIONS

Table 24-2. Recommended Operating Conditions

Parameter	Symbol	Rating (200MHz / 266MHz)		Unit
DC Supply Voltage for Internal	V_{DDi}	1.8V / 2.0V V_{DD}	1.8 ± 0.15 / 2.0 ± 0.1	V
DC Supply Voltage for RTC	V_{DDRTC}	1.8V V_{DD}	1.8 ± 0.15	
DC Supply Voltage for I/O Block	V_{DDIO}	3.3V V_{DD}	3.3 ± 0.3	
DC Supply Voltage for Analog Core	V_{DD}	3.3V V_{DD}	3.3 ± 5%	
DC Input Voltage	V_{IN}	3.3V Input buffer	3.3 ± 0.3	
		3.3V Interface / 5V Tolerant input buffer	3.0 – 5.25	
DC Output Voltage	V_{OUT}	3.3V Output buffer	3.3 ± 0.3	
Operating Temperature	T_{OPR}	Commercial	0 to 70	°C
		Industrial	-40 to 85	°C

D.C. ELECTRICAL CHARACTERISTICS

Table 24-3 and 24-4 define the DC electrical characteristics for the standard LVCMOS I/O buffers.

Table 24-3. Normal I/O PAD DC Electrical Characteristics

(V_{DD} = 3.3V ± 0.3V, T_A = -40 to 85 °C)

Symbol	Parameters	Condition	Min	Type	Max	Unit
V_{IH}	High level input voltage					
	LVCMOS interface		2.0			V
V_{IL}	Low level input voltage					
	LVCMOS interface				0.8	V
VT	Switching threshold			1.4		V
VT+	Schmitt trigger, positive-going threshold	CMOS			2.0	V
VT-	Schmitt trigger, negative-going threshold	CMOS	0.8			V
I_{IH}	High level input current					
	Input buffer	$V_{IN} = V_{DD}$	-10		10	µA
I_{IL}	Low level input current					
	Input buffer	$V_{IN} = V_{SS}$	-10		10	µA
	Input buffer with pull-up		-60	-33	-10	
V_{OH}	High level output voltage					
	Type B6	I_{OH}= - 6 mA	2.4			V
	Type B8	I_{OH}= - 8 mA				
	Type B12	I_{OH} = -12 mA				
V_{OL}	Low level output voltage					
	Type B6	I_{OL} = 6 mA			0.4	V
	Type B8	I_{OL} = 8 mA				
	Type B12	I_{OL} = 12 mA				

NOTES:
1. Type B6 means 6mA output driver cell.
2. Type B8 means 8mA output driver cell.
3. Type B12 means 12mA output driver cells.

Table 24-4. USB DC Electrical Characteristics

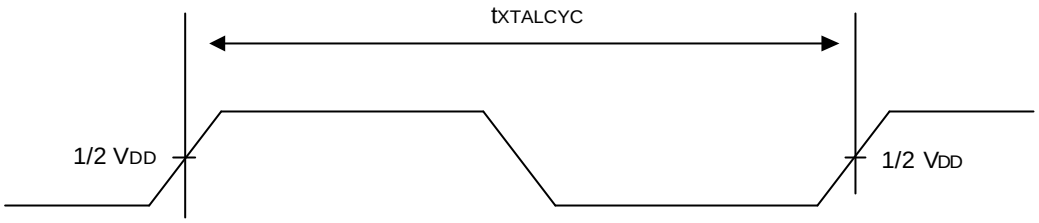
Symbol	Parameter	Condition	Min	Max	Unit
V_{IH}	High level input voltage		2.5		V
V_{IL}	Low level input voltage			0.8	V
I_{IH}	High level input current	Vin = 3.3V	-10	10	µA
I_{IL}	Low level input current	Vin = 0.0V	-10	10	µA
V_{OH}	Static Output High	15KΩ to GND	2.8	3.6	V
V_{OL}	Static Output Low	1.5KΩ to 3.6V		0.3	V

Table 24-5. S3C2410A Power Supply Voltage and Current

Parameter	200MHz	266MHz	Unit	Condition
Typical V_{DDi} / V_{DDIO}	1.8 / 3.3	2.0 / 3.3	V	
Max. Operating frequency (FCLK)	200	266	MHz	
Max. Operating frequency (HCLK)	100	133	MHz	
Max. Operating frequency (PCLK)	50	66.5	MHz	
Typical normal mode power NOTE(3) (Total V_{DDi} + V_{DDIO})	259	335	mW	NOTE(1) Bus Rate = 1:2:4
Typical idle mode power NOTE(3) (Total V_{DDi} + V_{DDIO})	124	177	mW	NOTE(2) Bus Rate = 1:2:4
Typical slow mode power NOTE(3) (Total V_{DDi} + V_{DDIO})	33	33	mW	FCLK = 12MHz Bus Rate = 1:1:1
Maximum Power_OFF mode power NOTE(3) Typical Power_OFF mode power NOTE(3)	80 8	100 50	uA uA	Just running 32.768KHz oscillator (for RTC), all other I/O static.
Maximum RTC power NOTE(3) Typical RTC power NOTE(3)	5 3	5 3	uA uA	X-tal = 32.768KHz for RTC VDD_{RTC}=1.8V

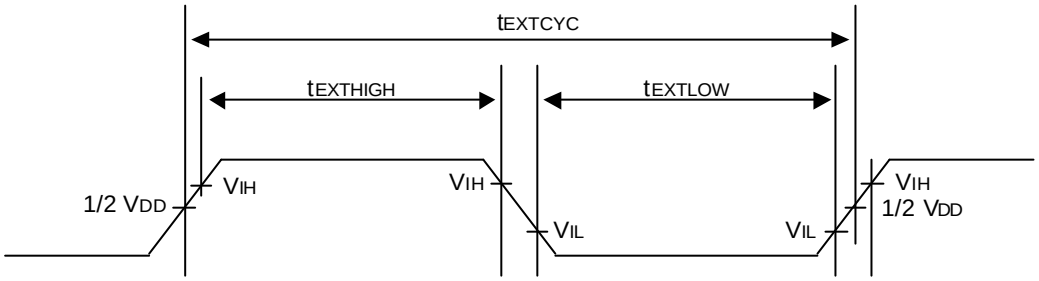
NOTES:
1. Playing matrix2.wmv on PPC2003.
2. No threads ready to run on PPC2003.
3. Room temperature specification

A.C. ELECTRICAL CHARACTERISTICS



NOTE: The clock input from the XTIpll pin.

Figure 24-1. XTIpll Clock Timing



NOTE: The clock input from the EXTCLK pin.

Figure 24-2. EXTCLK Clock Input Timing

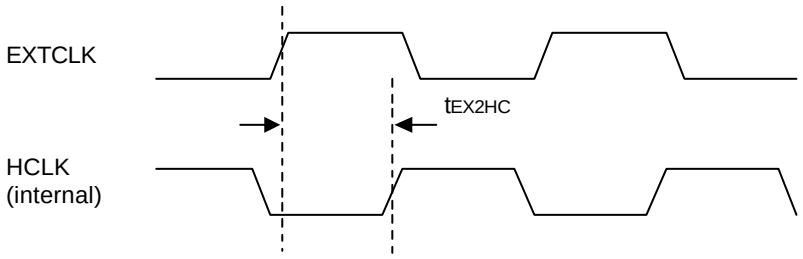


Figure 24-3. EXTCLK/HCLK in case that EXTCLK is used without the PLL

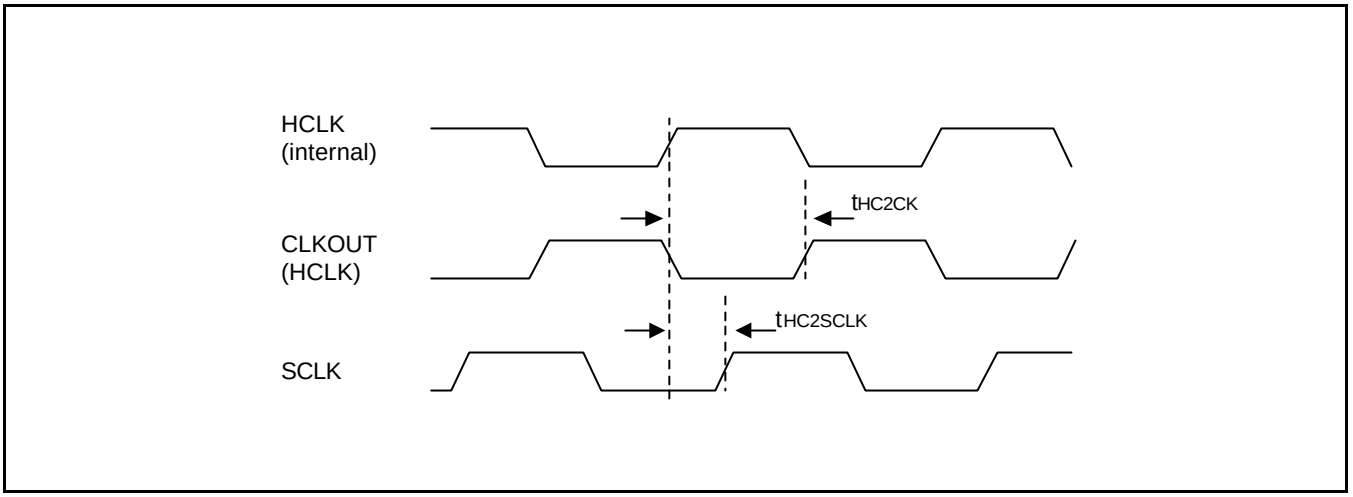


Figure 24-4. HCLK/CLKOUT/SCLK in case that EXTCLK is used

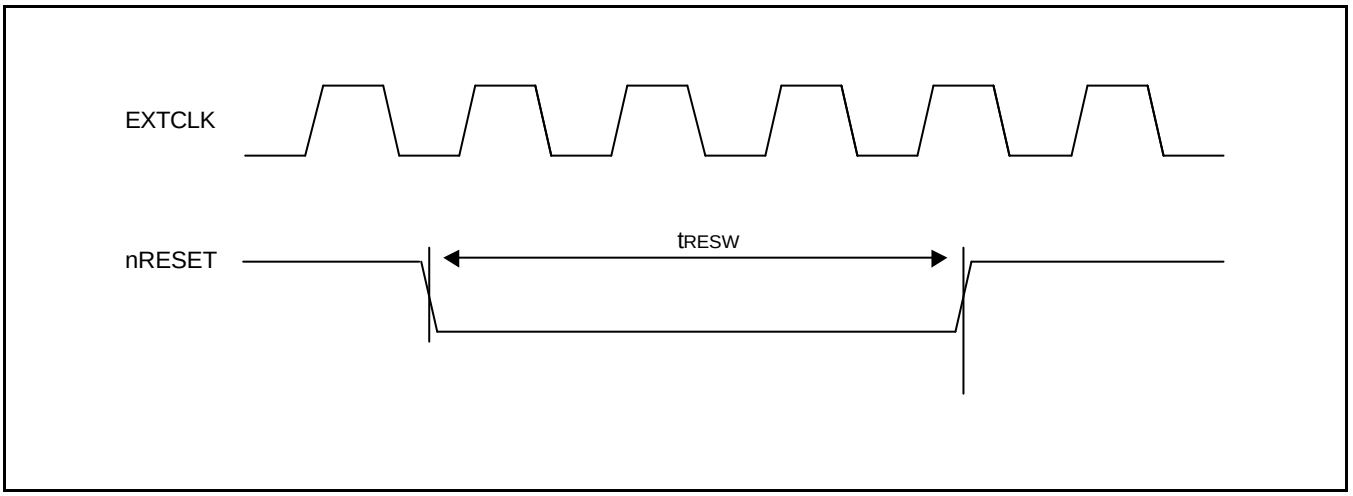


Figure 24-5. Manual Reset Input Timing

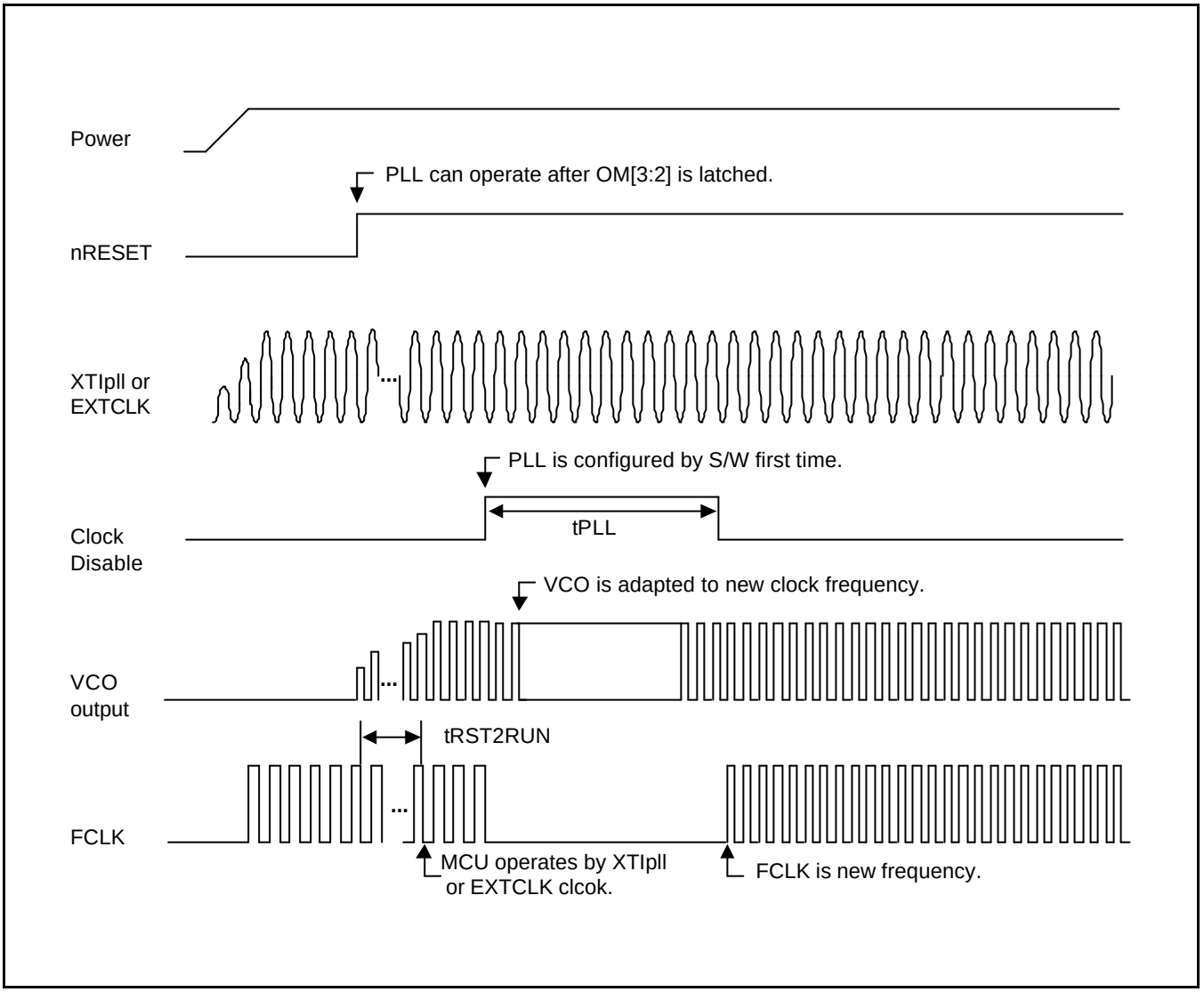


Figure 24-6. Power-On Oscillation Setting Timing

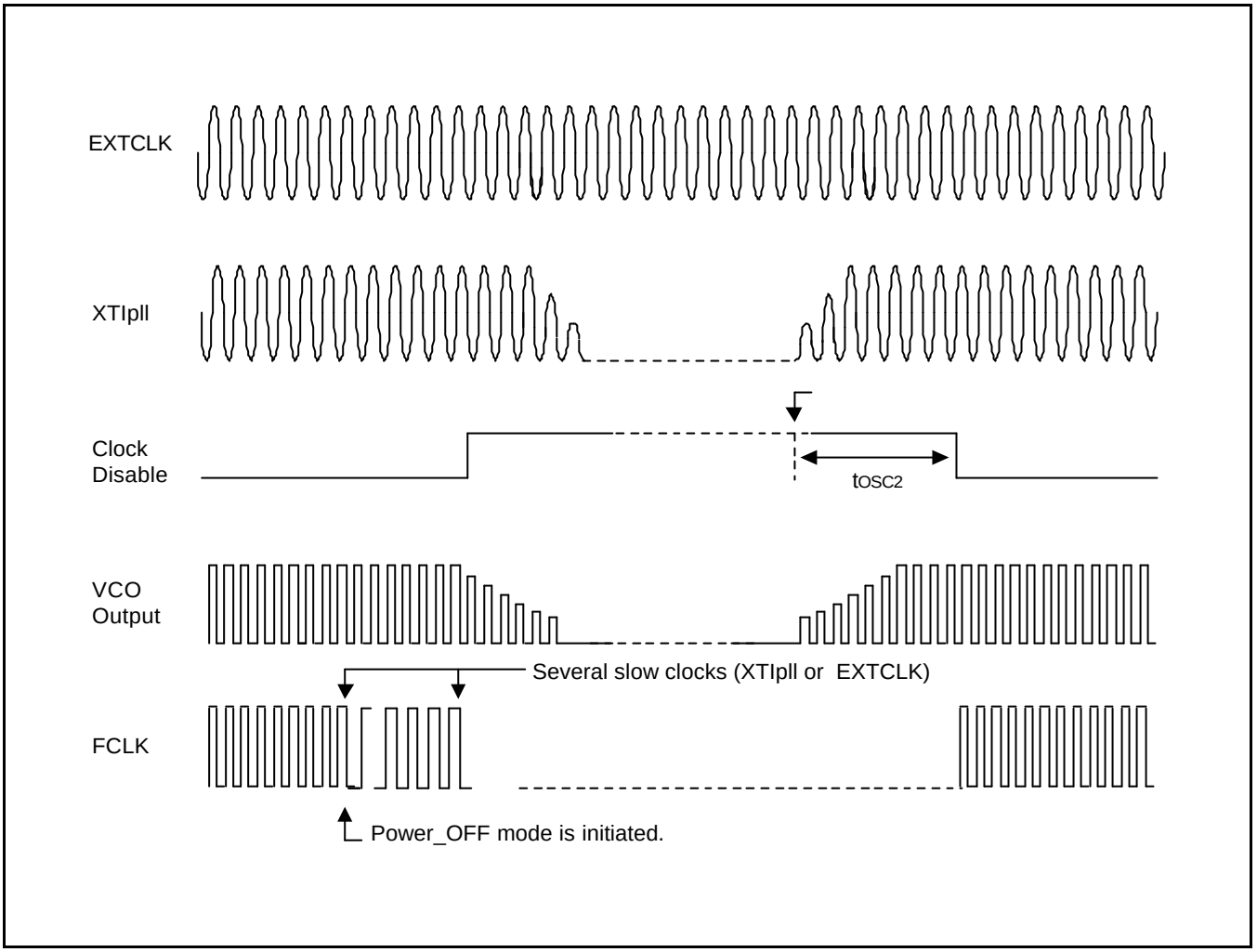


Figure 24-7. Power_OFF Mode Return Oscillation Setting Timing

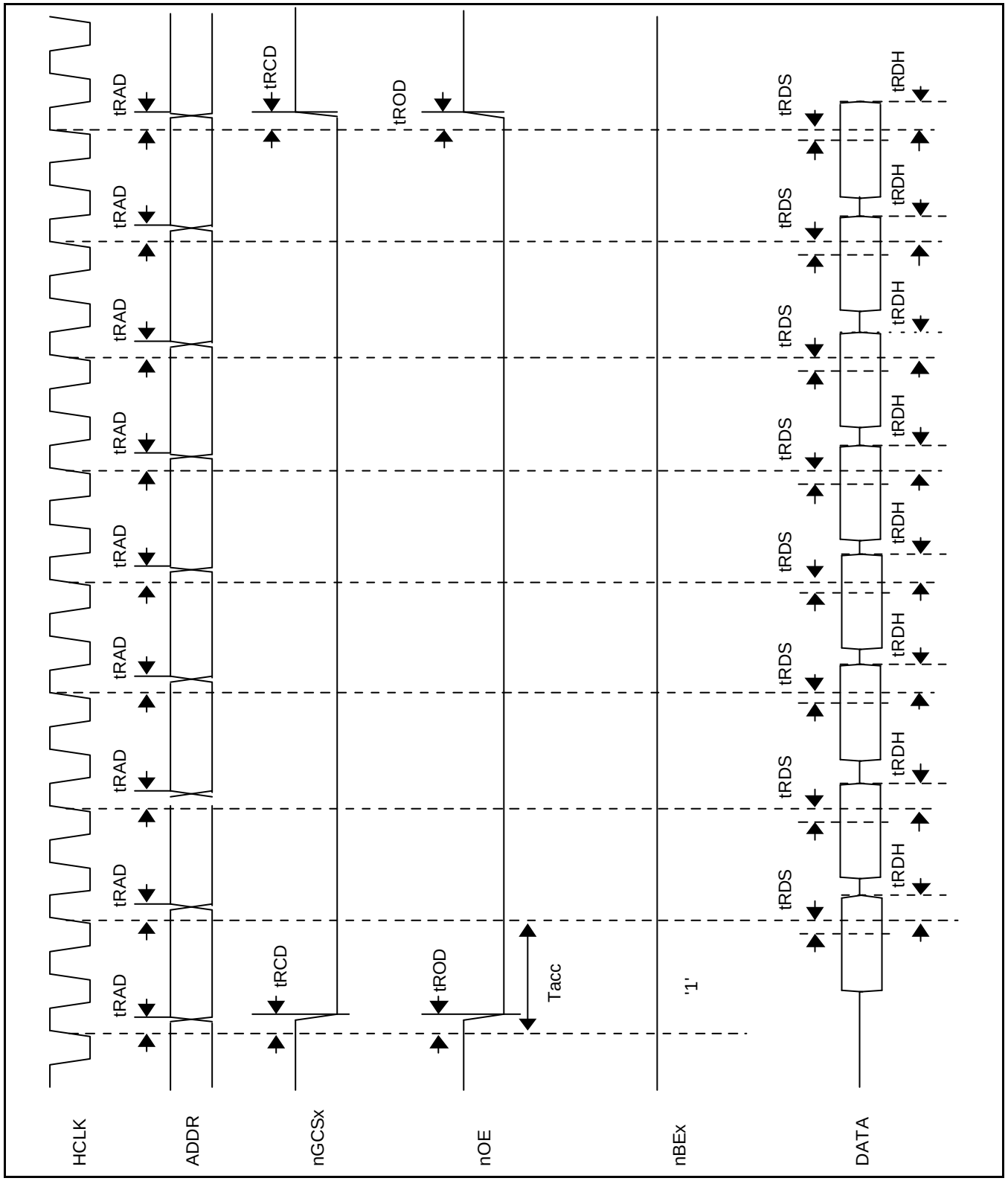


Figure 24-8. ROM/SRAM Burst READ Timing(I)
(Tacs = 0, Tcos = 0, Tacc = 2, Tcoh = 0, Tcah = 0, PMC = 0, ST = 0, DW = 16-bit)

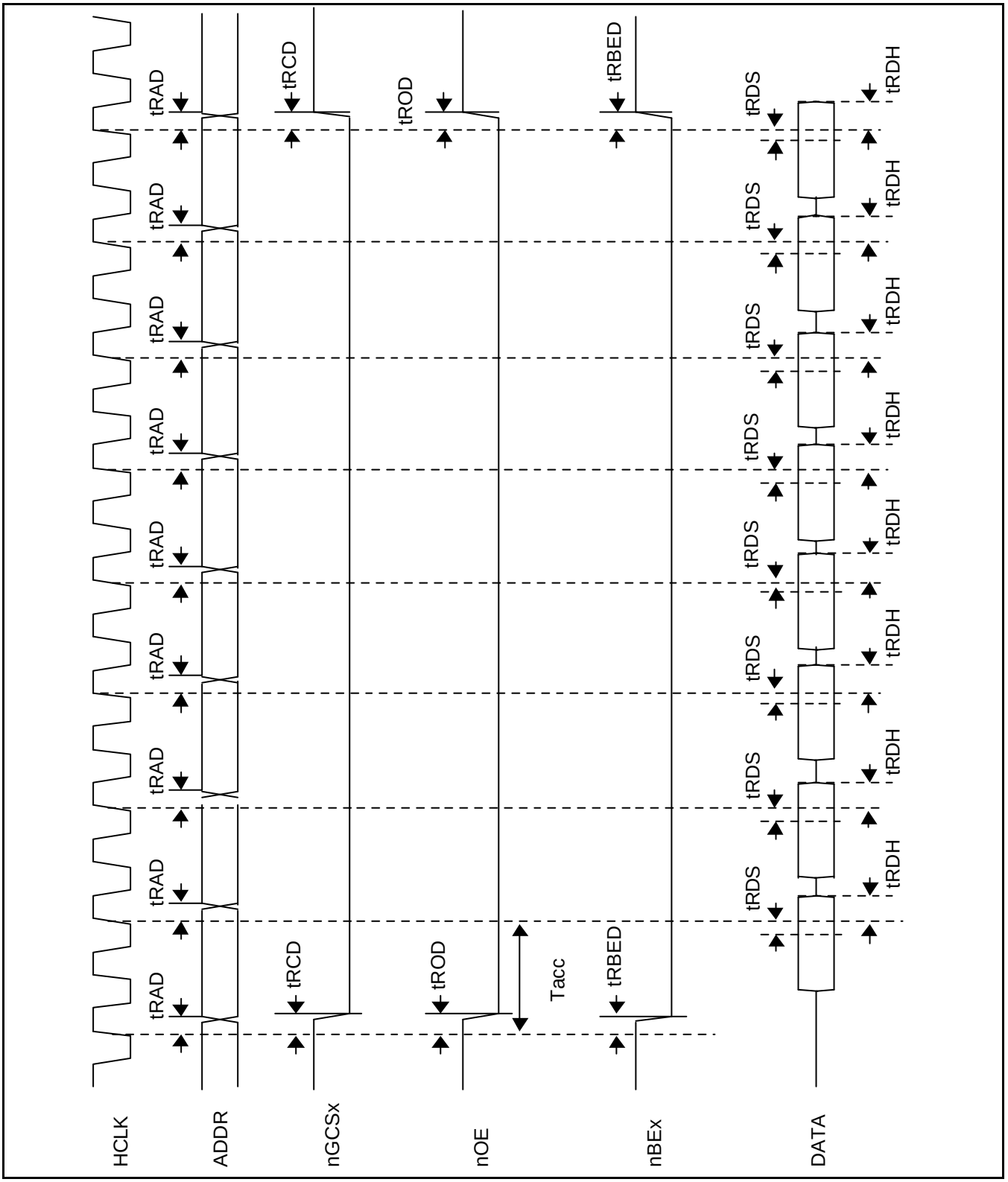


Figure 24-9. ROM/SRAM Burst READ Timing(II)
(Tacs = 0, Tcos = 0, Tacc = 2, Tcoh = 0, Tcah = 0, PMC = 0, ST = 1, DW = 16-bit)

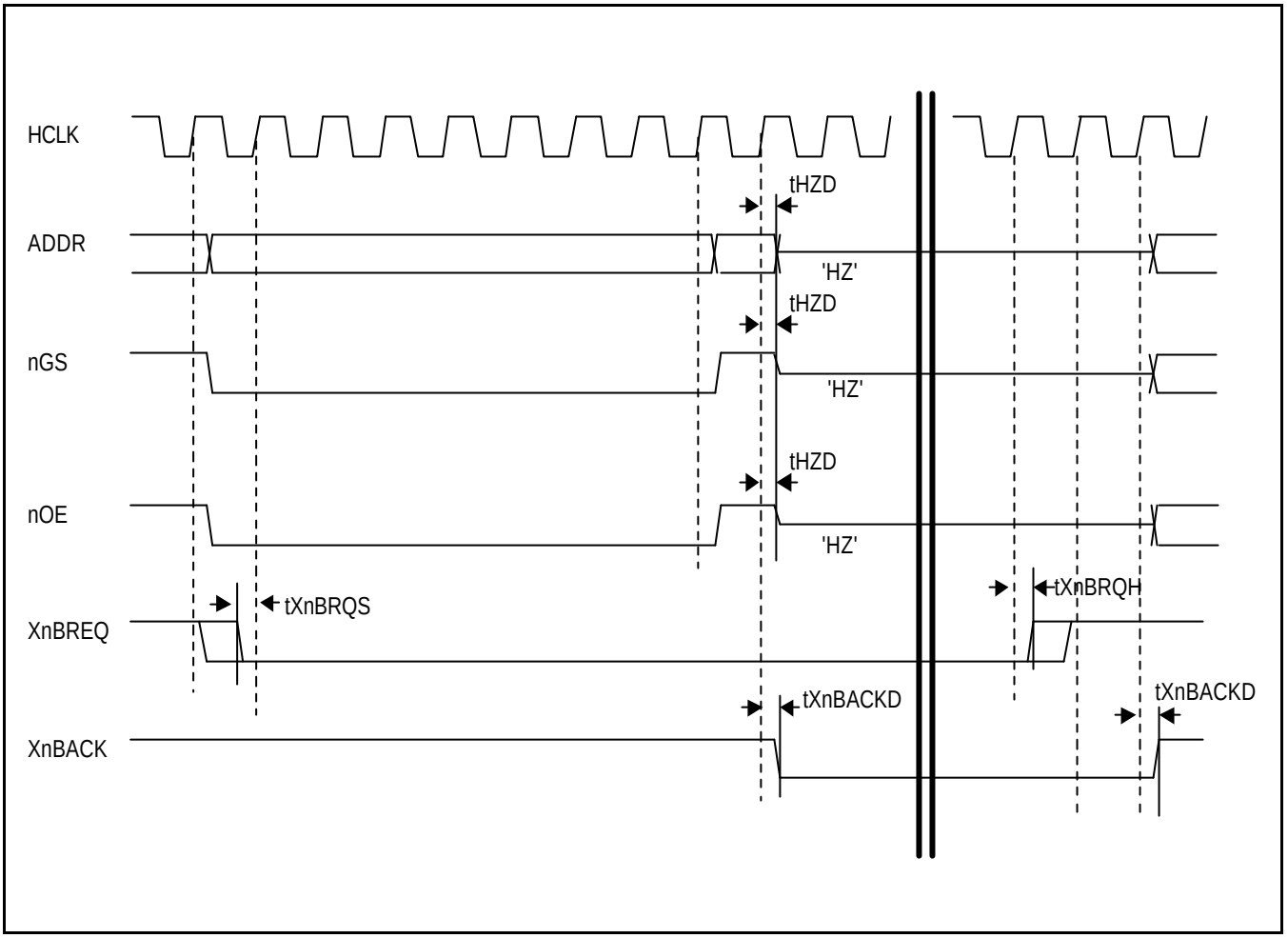


Figure 24-10. External Bus Request in ROM/SRAM Cycle
(Tacs = 0, Tcos = 0, Tacc = 8, Tcoh = 0, Tcah = 0, PMC = 0, ST = 0)

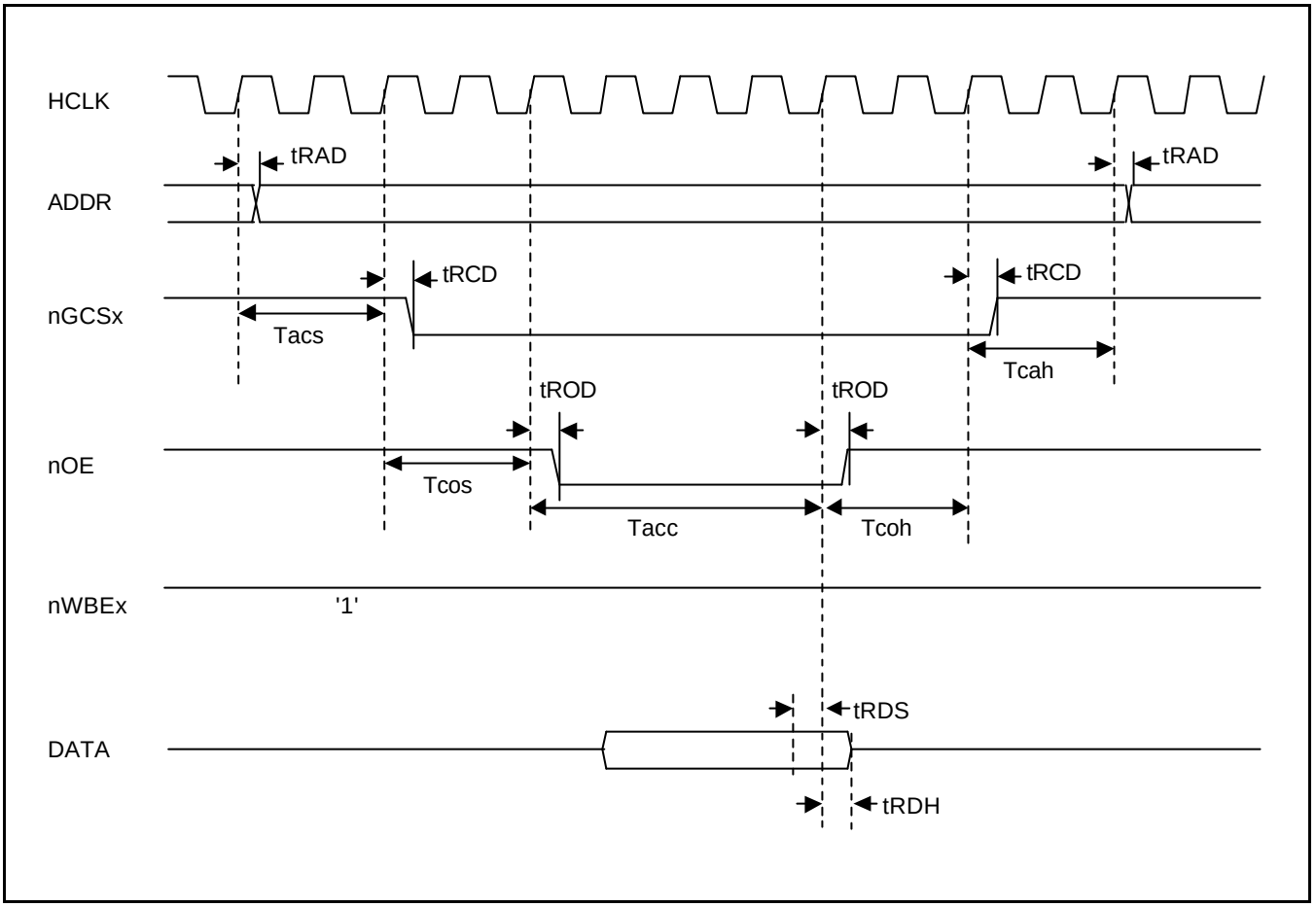


Figure 24-11. ROM/SRAM READ Timing (I)
(Tacs = 2,Tcos = 2, Tacc = 4, Tcoh = 2, Tcah = 2, PMC = 0, ST = 0)

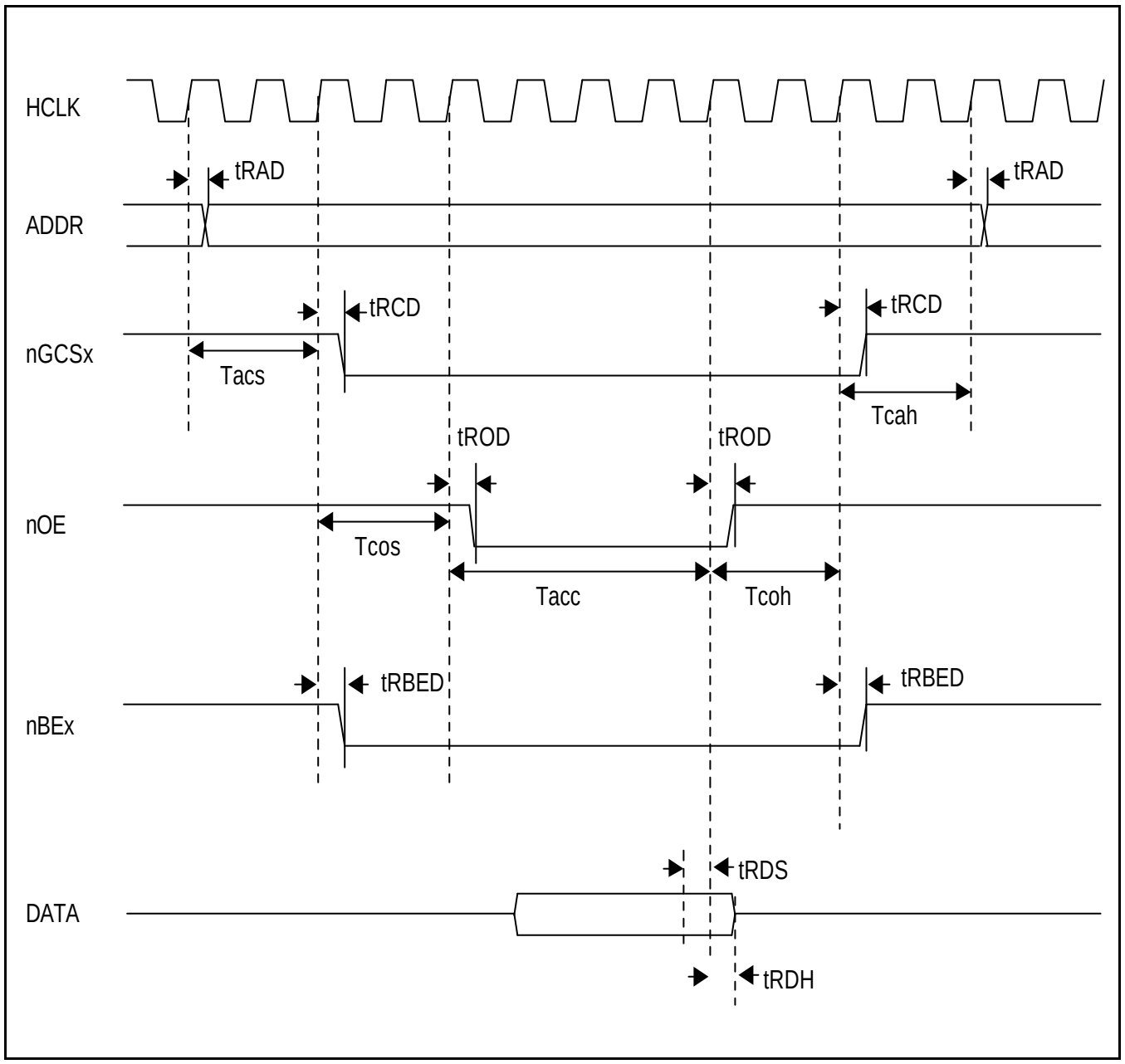


Figure 24-12. ROM/SRAM READ Timing (II)
(Tacs = 2, Tcos = 2, Tacc = 4, Tcoh = 2, Tcah = 2, PMC = 0, ST = 1)

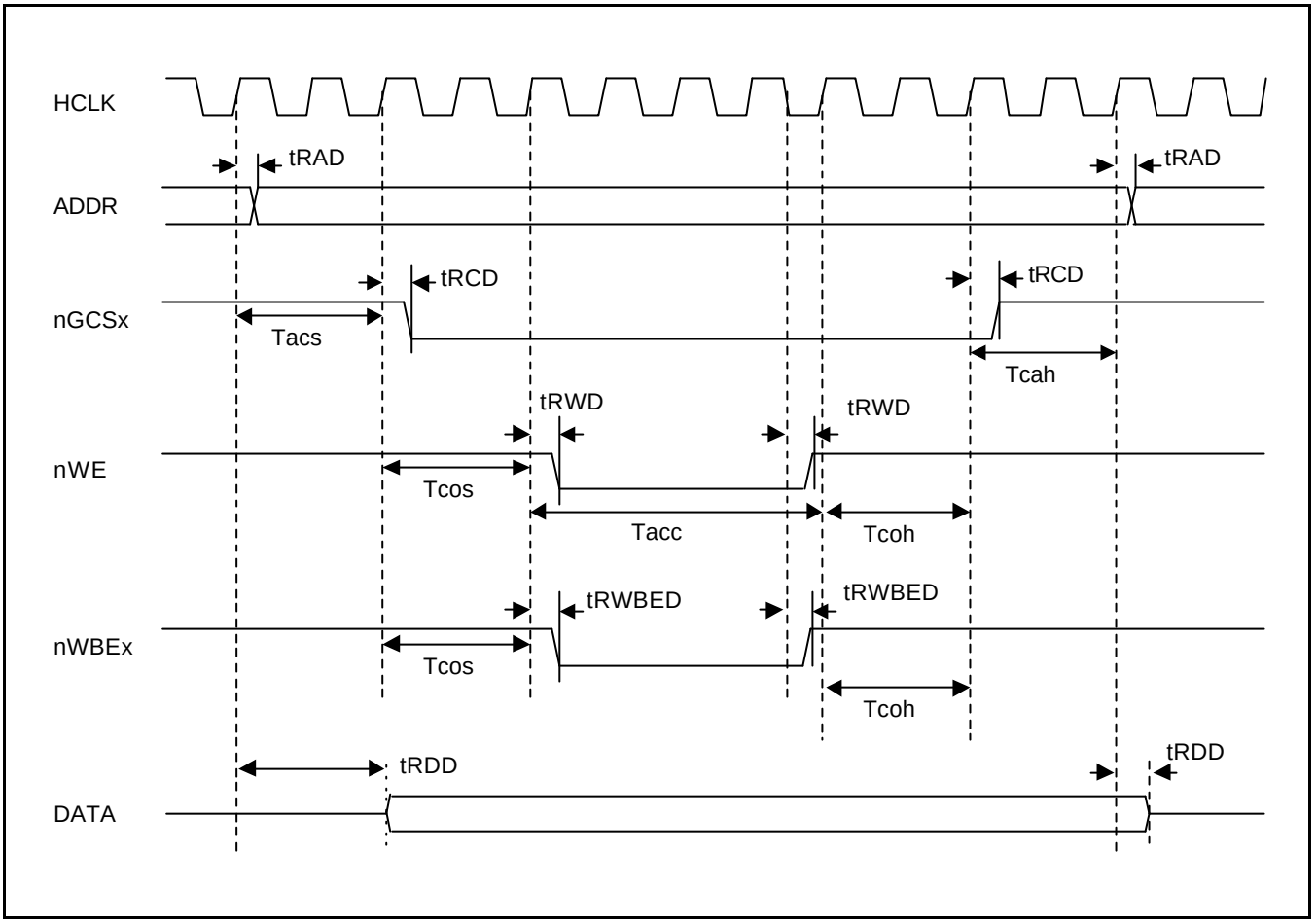


Figure 24-13. ROM/SRAM WRITE Timing (I)
(Tacs = 2,Tcos = 2,Tacc = 4,Tcoh = 2, Tcah = 2, PMC = 0, ST = 0

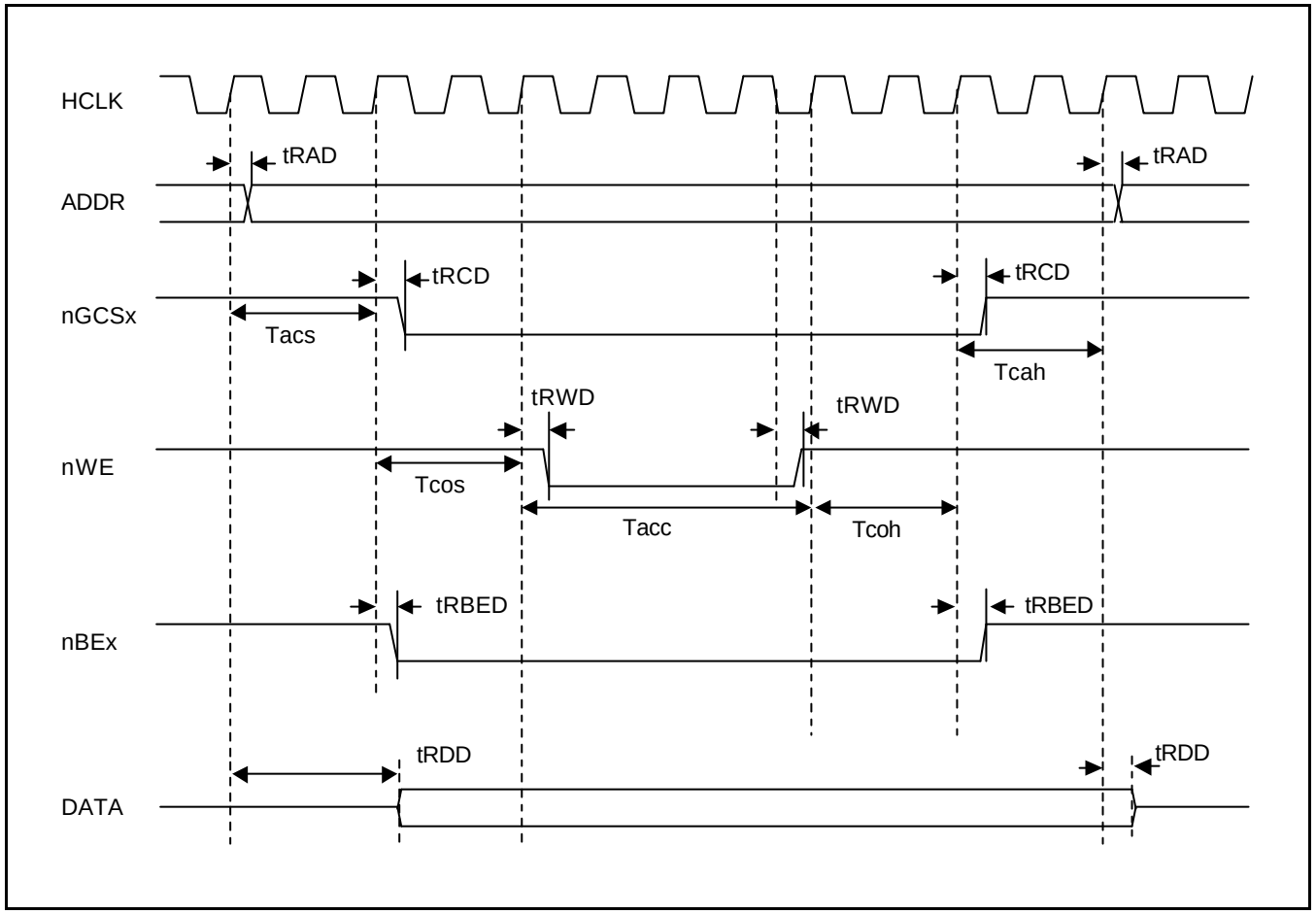
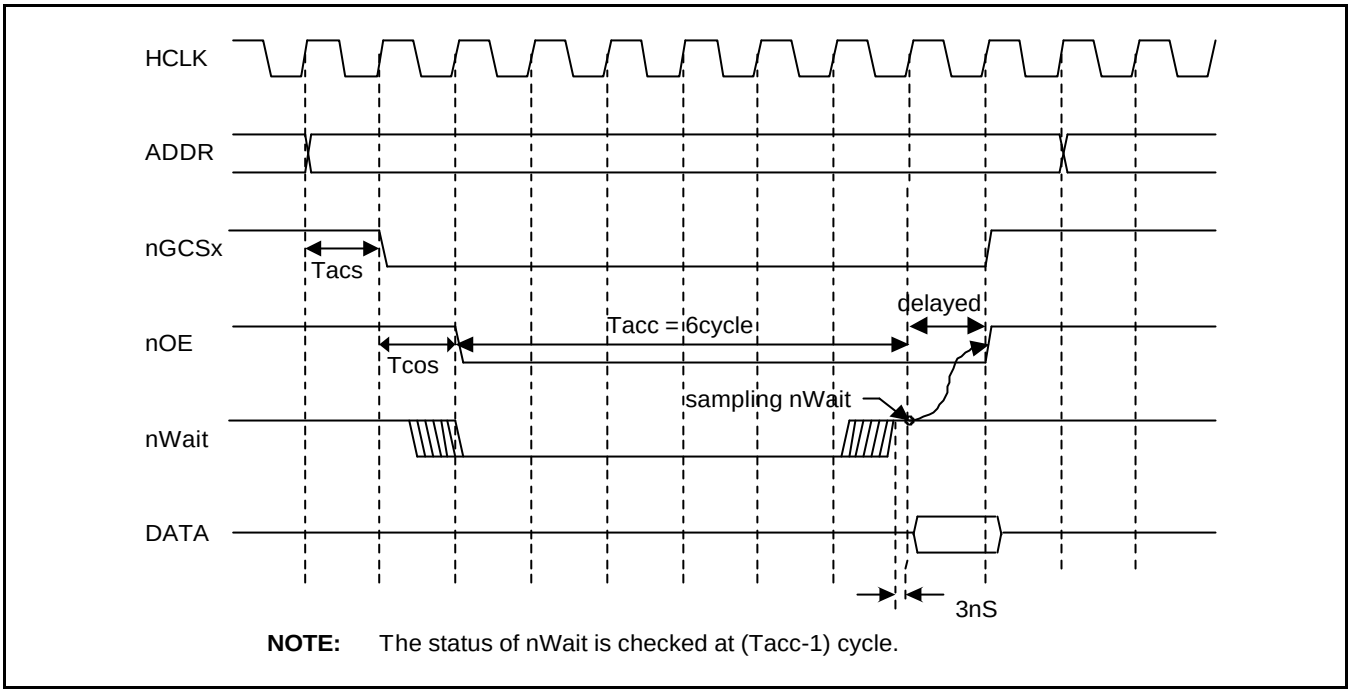


Figure 24-14. ROM/SRAM WRITE Timing (II)
(Tacs = 2, Tcos = 2, Tacc = 4, Tcoh = 2, Tcah = 2, PMC = 0, ST = 1)



NOTE: The status of nWait is checked at (Tacc-1) cycle.

Figure 24-15. External nWAIT READ Timing
(Tacs = 1, Tcos = 1, Tacc = 4, Tcoh = 0, Tcah = 1, PMC = 0, ST = 0)

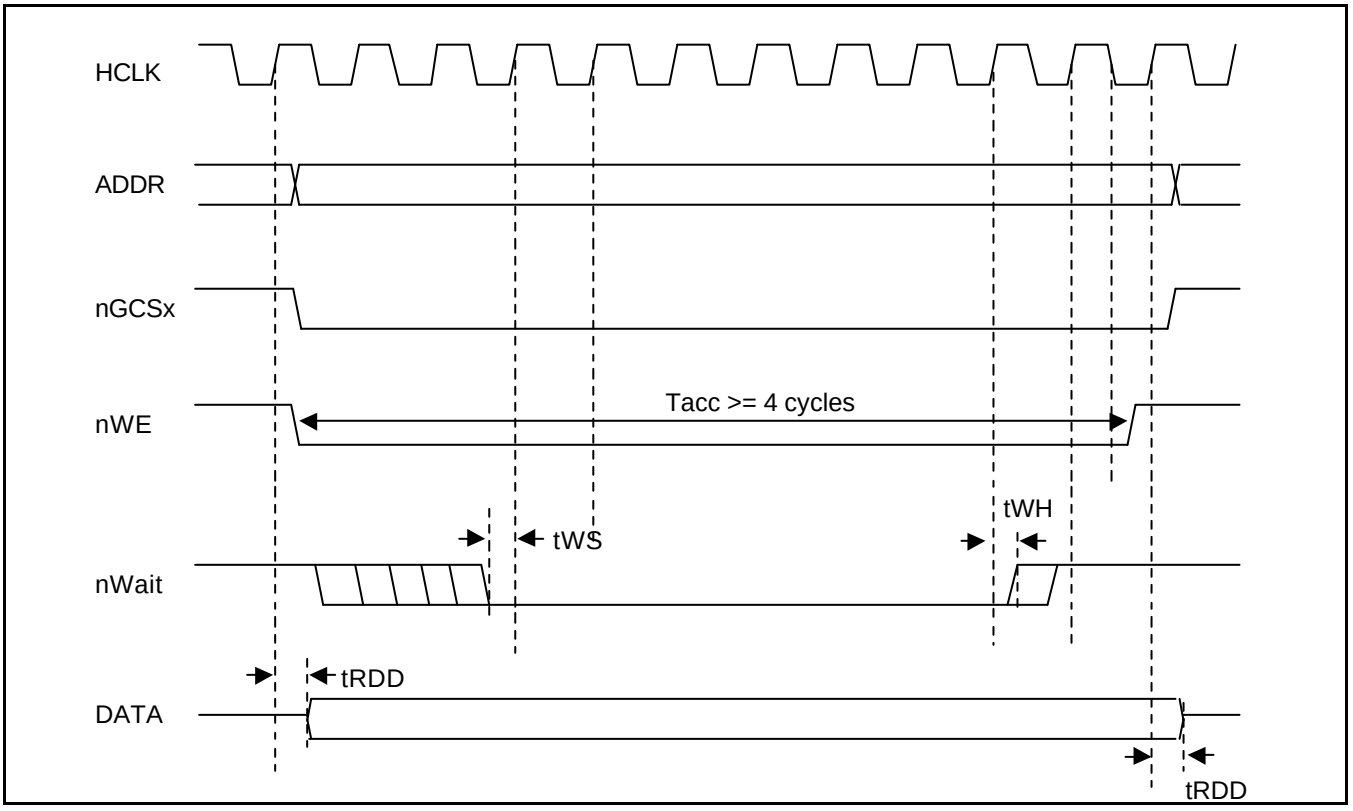


Figure 24-16. External nWAIT WRITE Timing
(Tacs = 0, Tcos = 0, Tacc = 4, Tcoh = 0, Tcah = 0, PMC = 0, ST = 0)

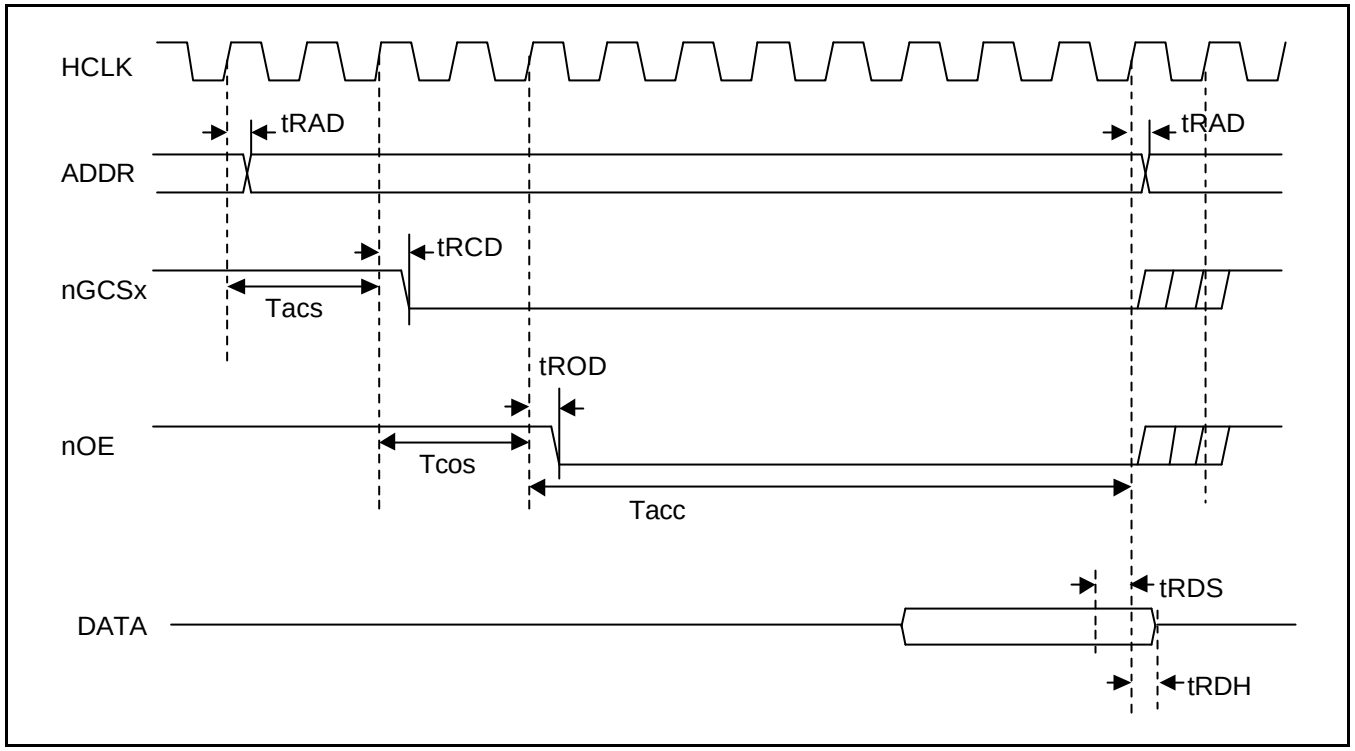


Figure 24-17. Masked-ROM Single READ Timing (Tacs = 2, Tcos = 2, Tacc = 8, PMC = 01/10/11)

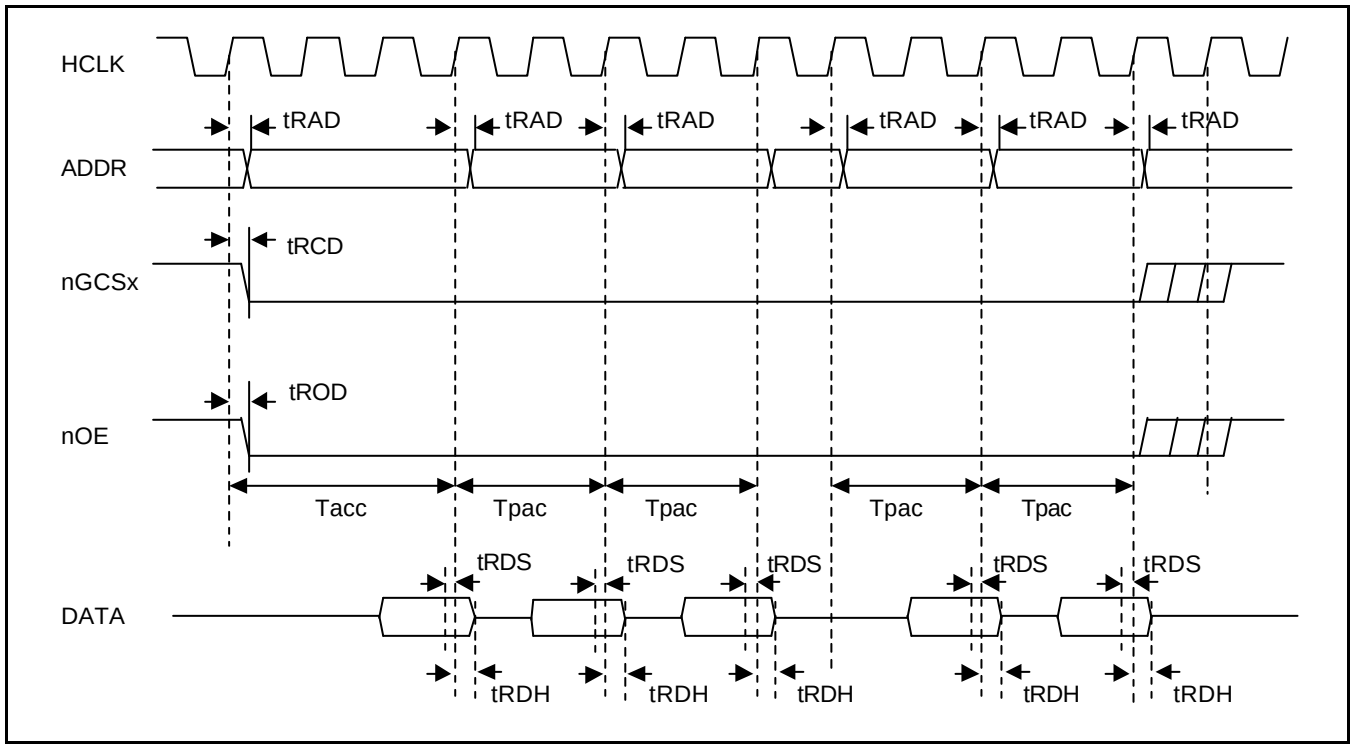


Figure 24-18. Masked-ROM Consecutive READ Timing (Tacs = 0, Tcos = 0, Tacc = 3, Tpac = 2, PMC = 01/10/11)

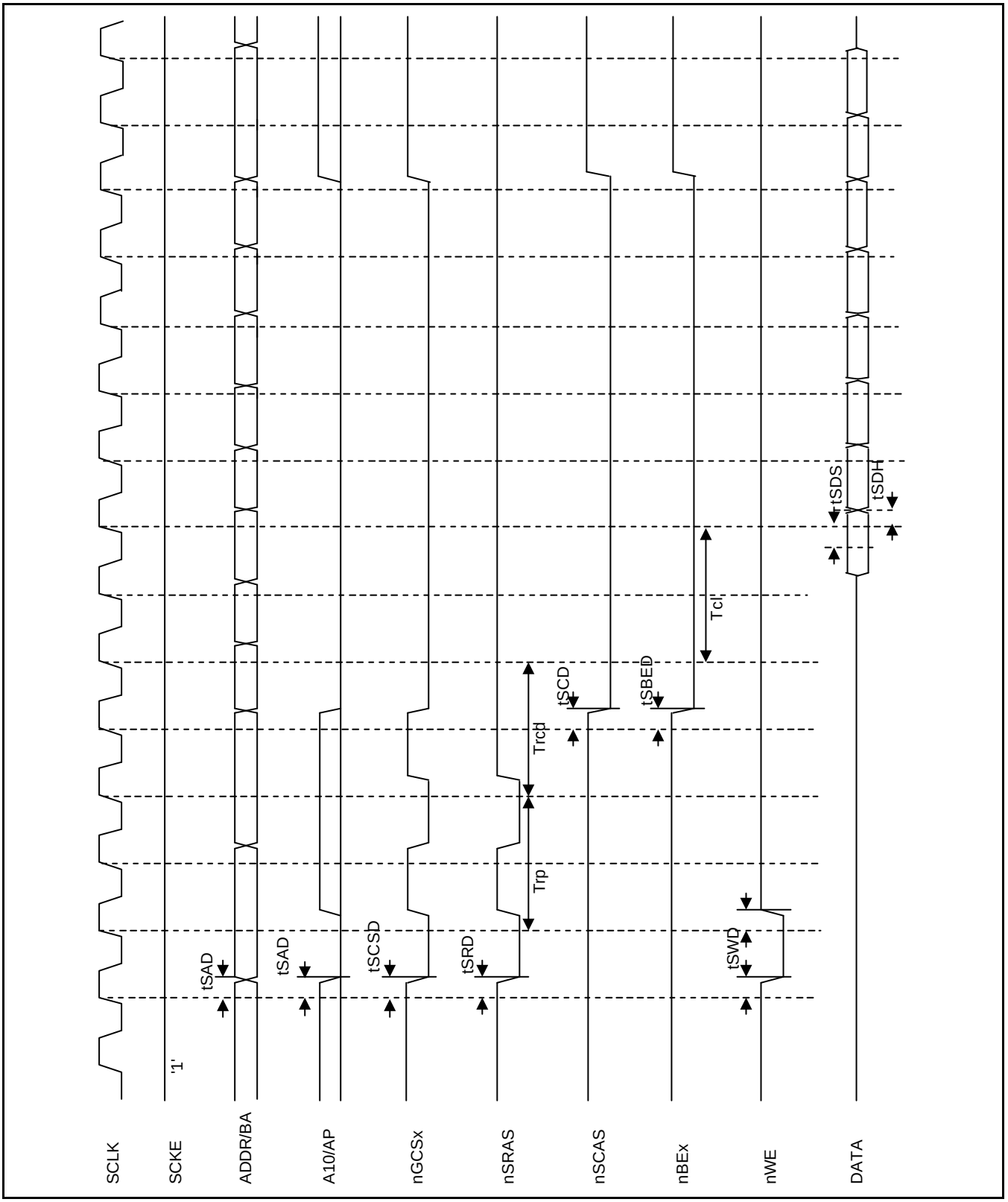


Figure 24-19. SDRAM Single Burst READ Timing (Trp = 2, Trcd = 2, Tcl = 2, DW = 16-bit)

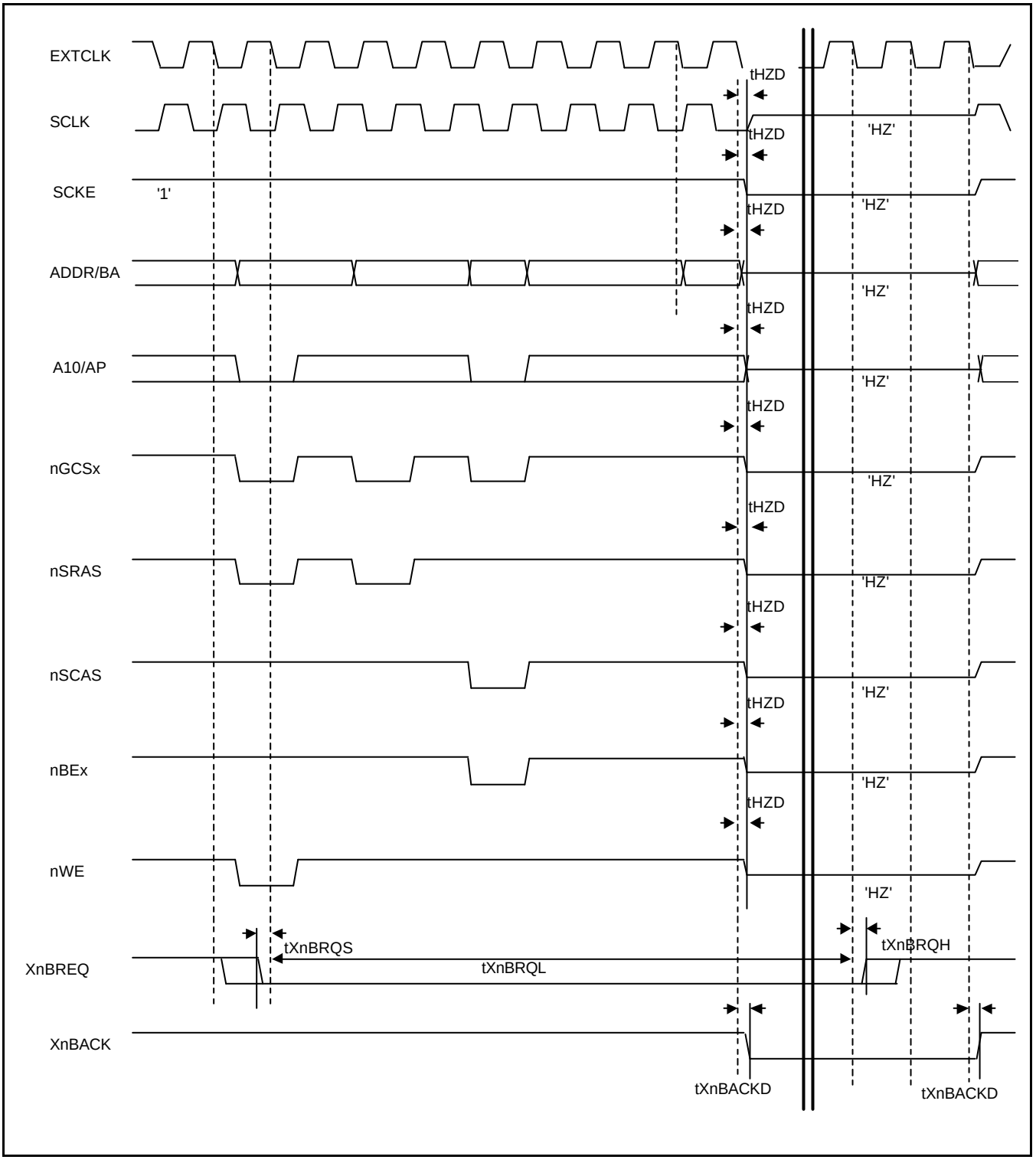


Figure 24-20. External Bus Request in SDRAM Timing (Trp = 2, Trcd = 2, Tcl = 2)

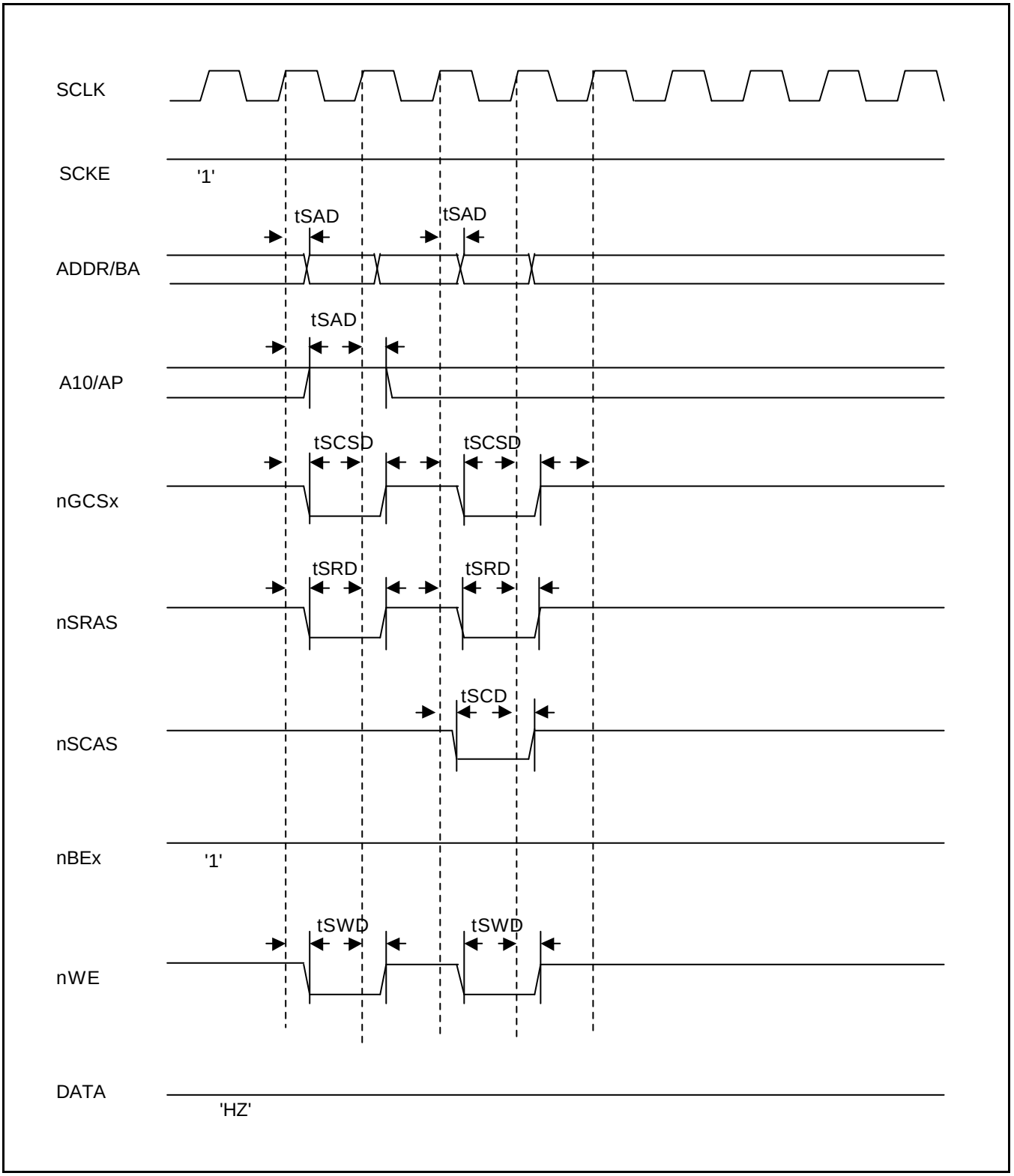


Figure 24-21. SDRAM MRS Timing

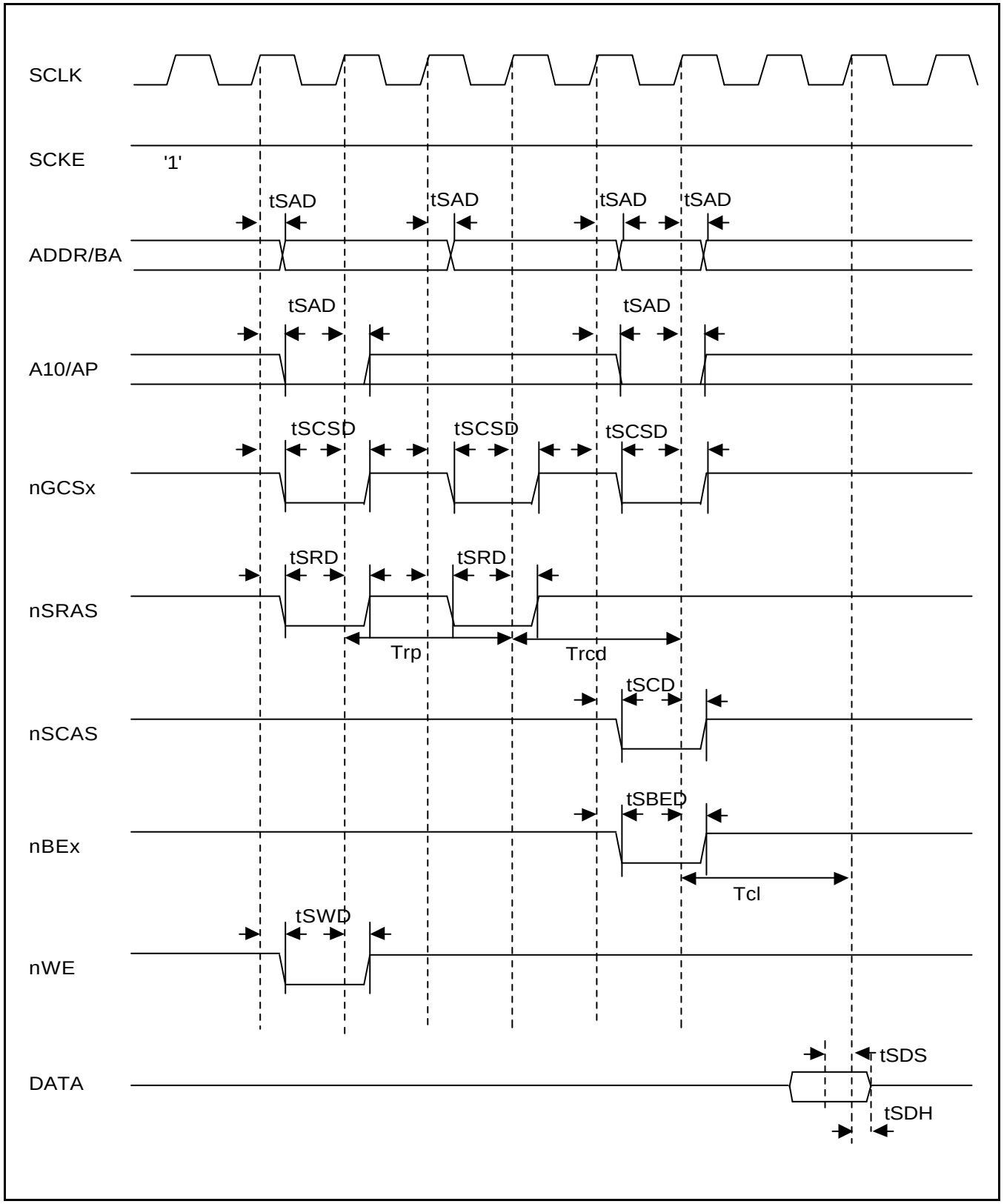


Figure 24-22. SDRAM Single READ Timing(I) (Trp = 2, Trcd = 2, Tcl = 2)

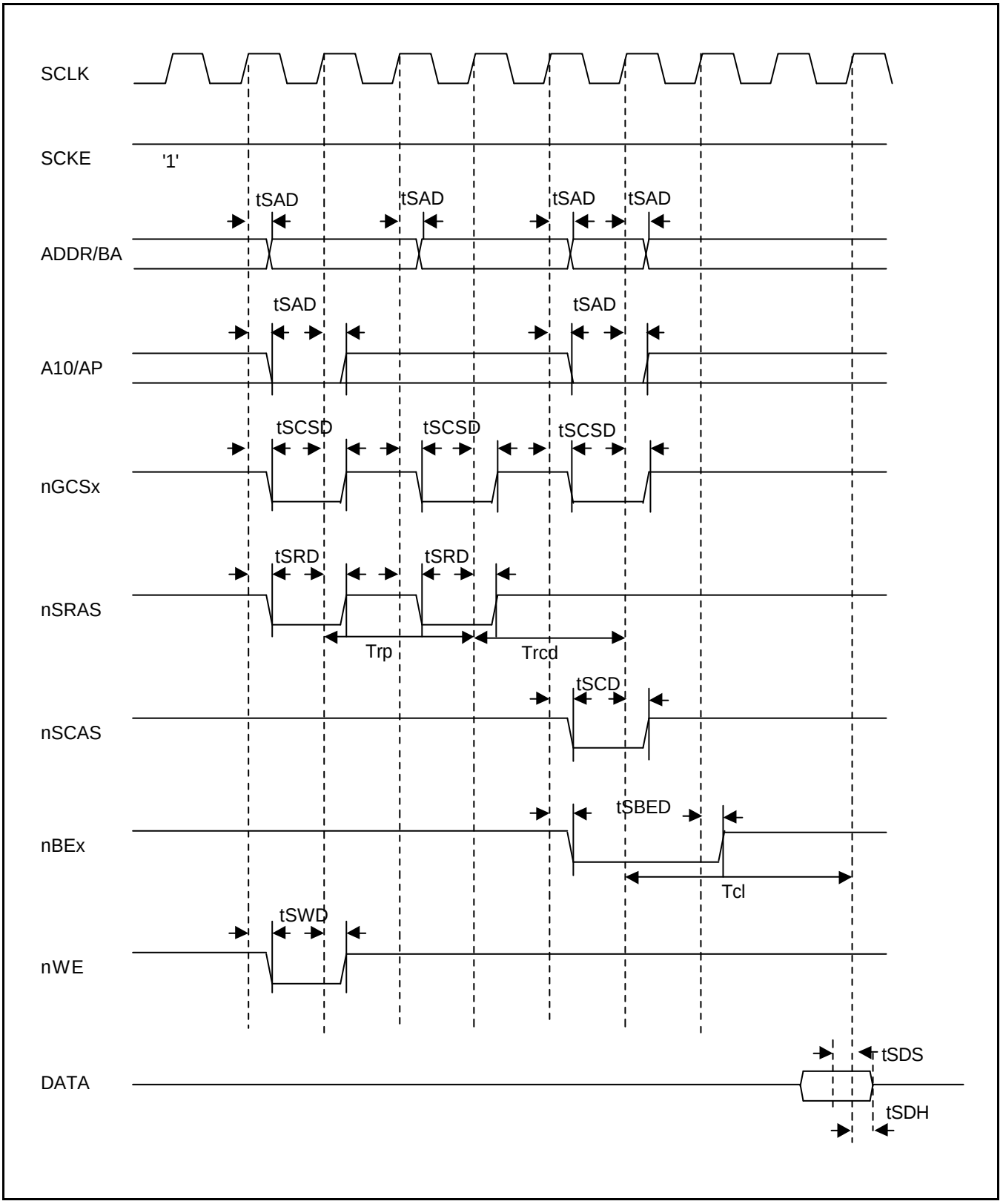
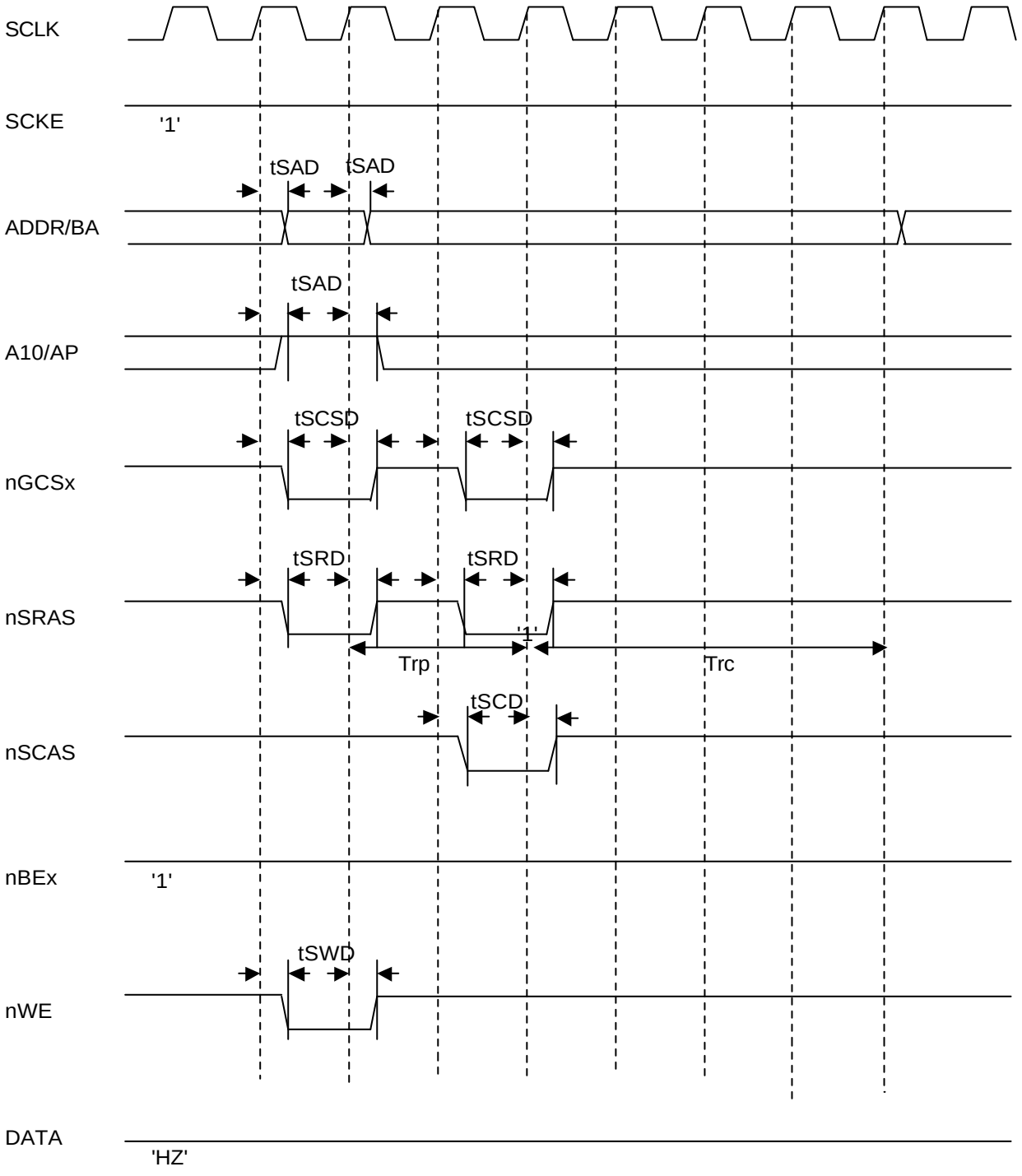


Figure 24-23 SDRAM Single READ Timing(II) (Trp = 2, Trcd = 2, Tcl = 3)



NOTE: Before executing auto/self refresh command, all banks must be in idle state.

Figure 24-24. SDRAM Auto Refresh Timing (Trp = 2, Trc = 4)

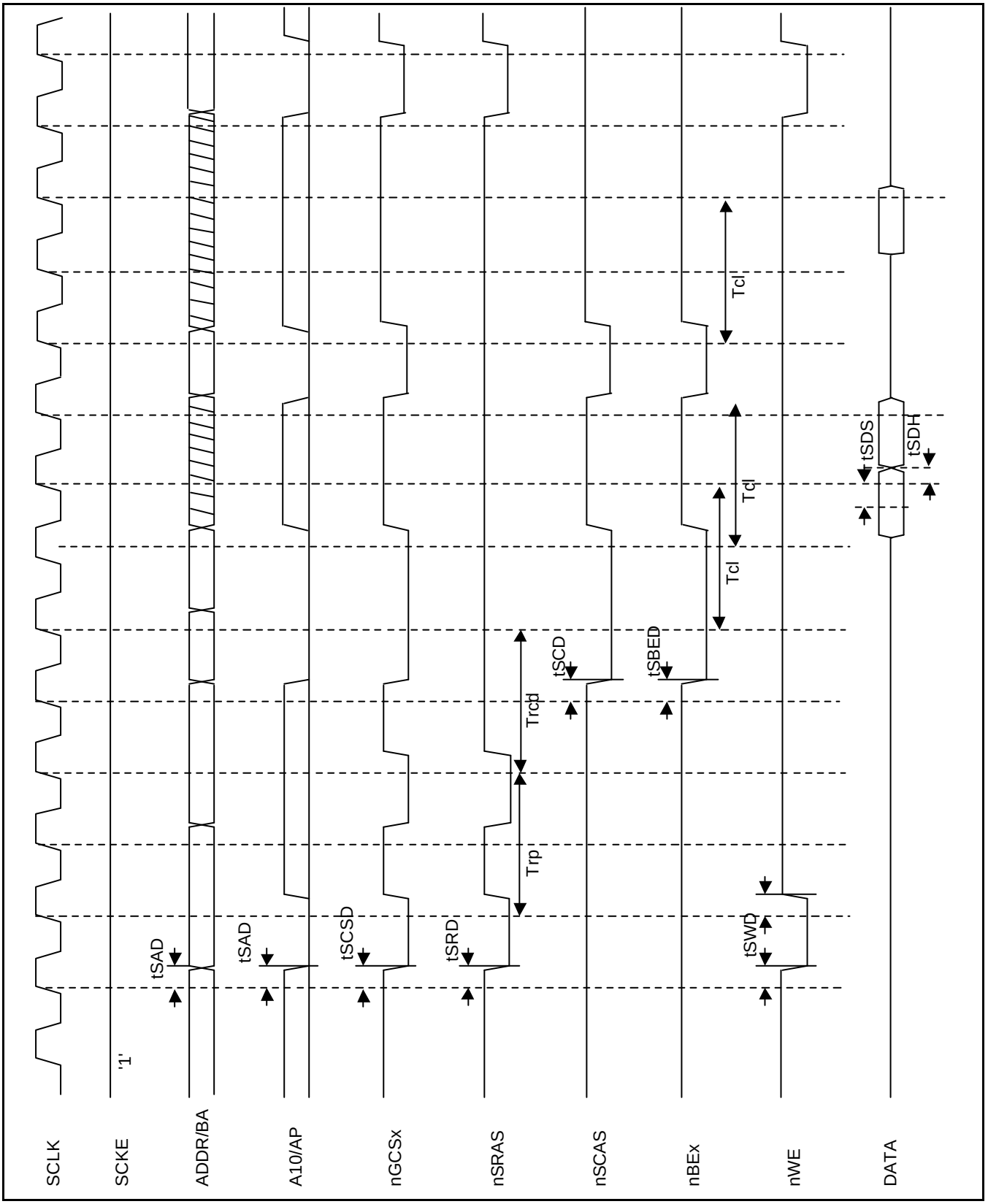
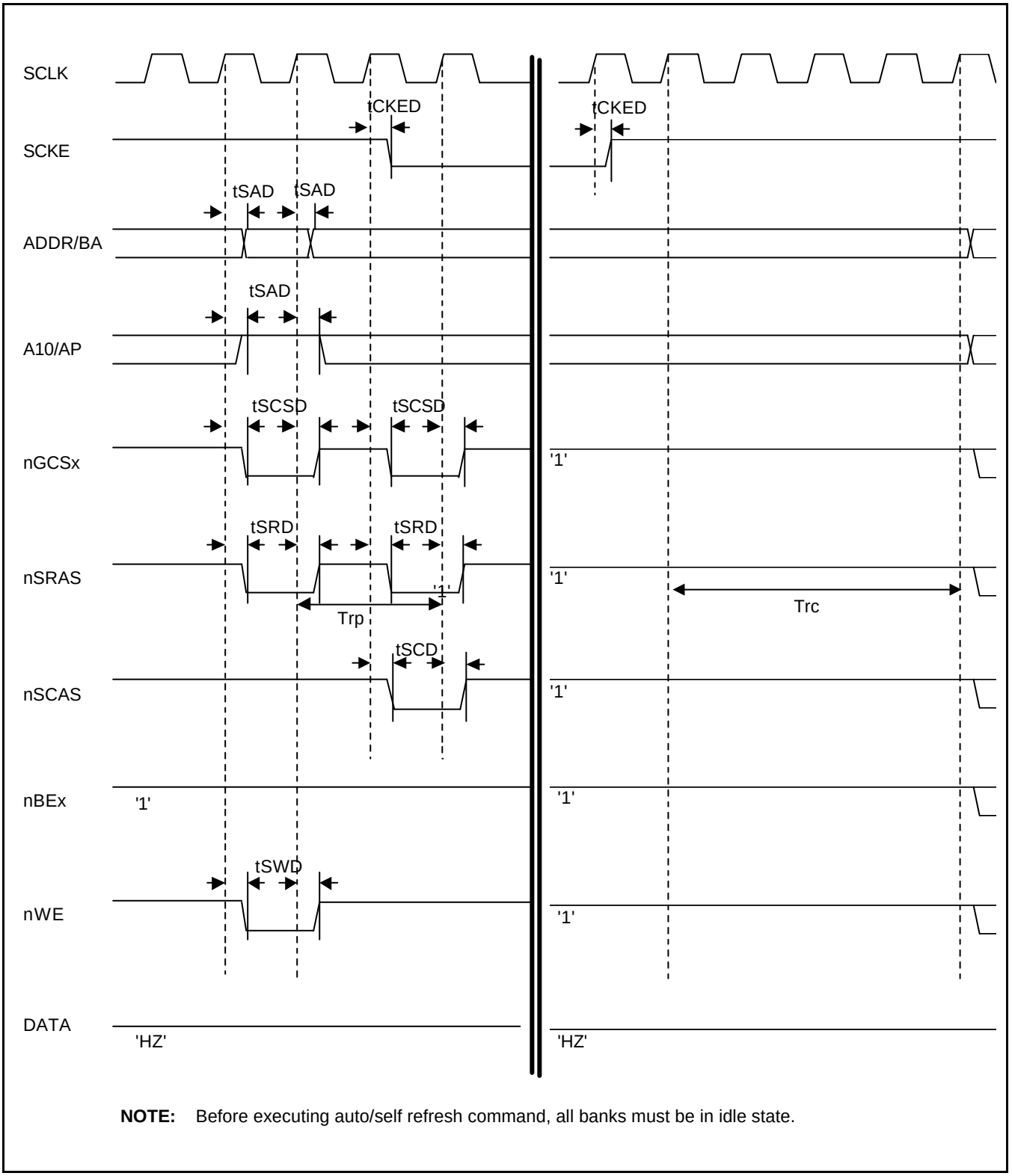


Figure 24-25. SDRAM Page Hit-Miss READ Timing (Trp = 2, Trcd = 2, Tcl = 2)



NOTE: Before executing auto/self refresh command, all banks must be in idle state.

Figure 24-26. SDRAM Self Refresh Timing (Trp = 2, Trc = 4)

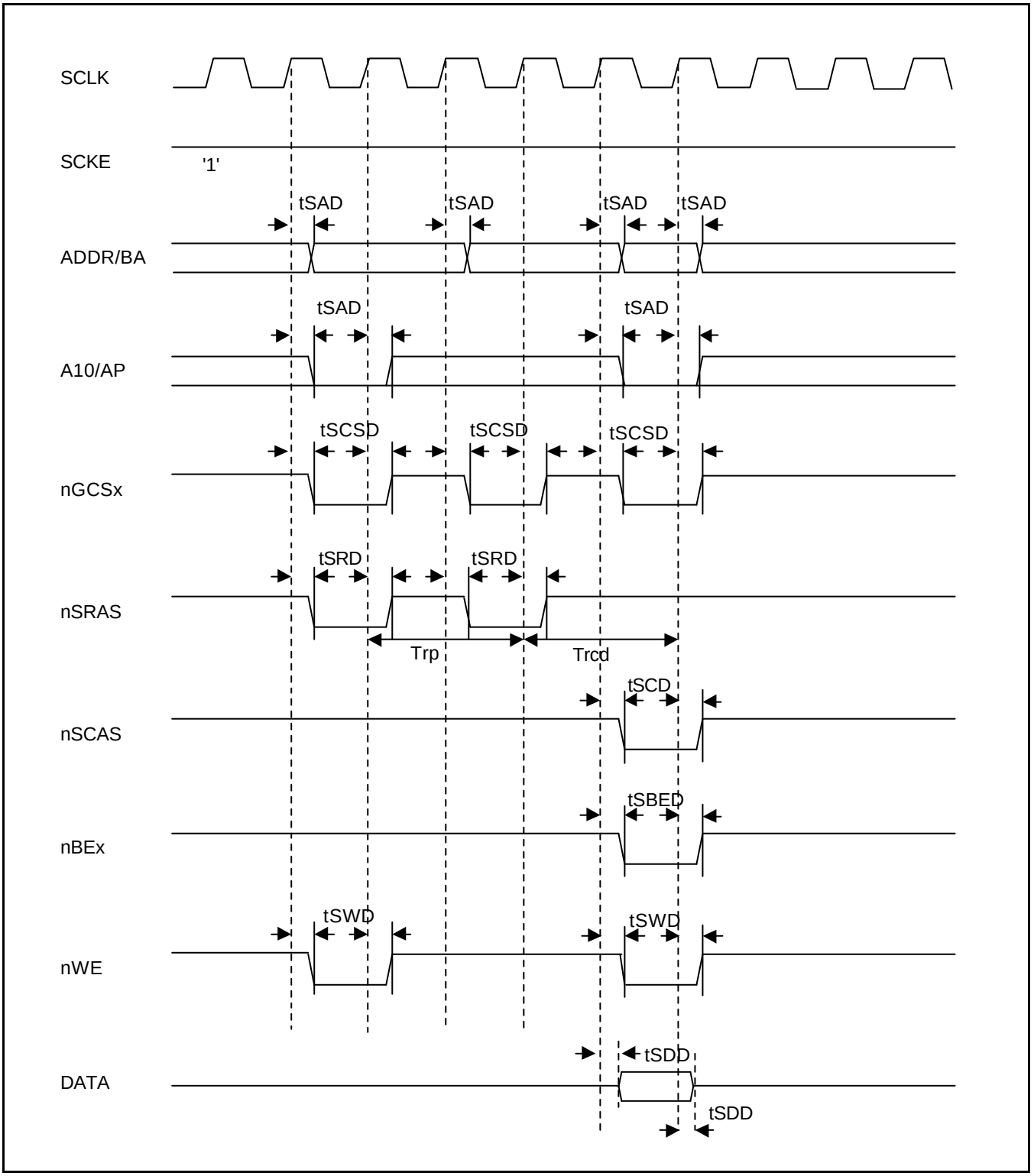


Figure 24-27. SDRAM Single Write Timing (Trp = 2, Trcd = 2)

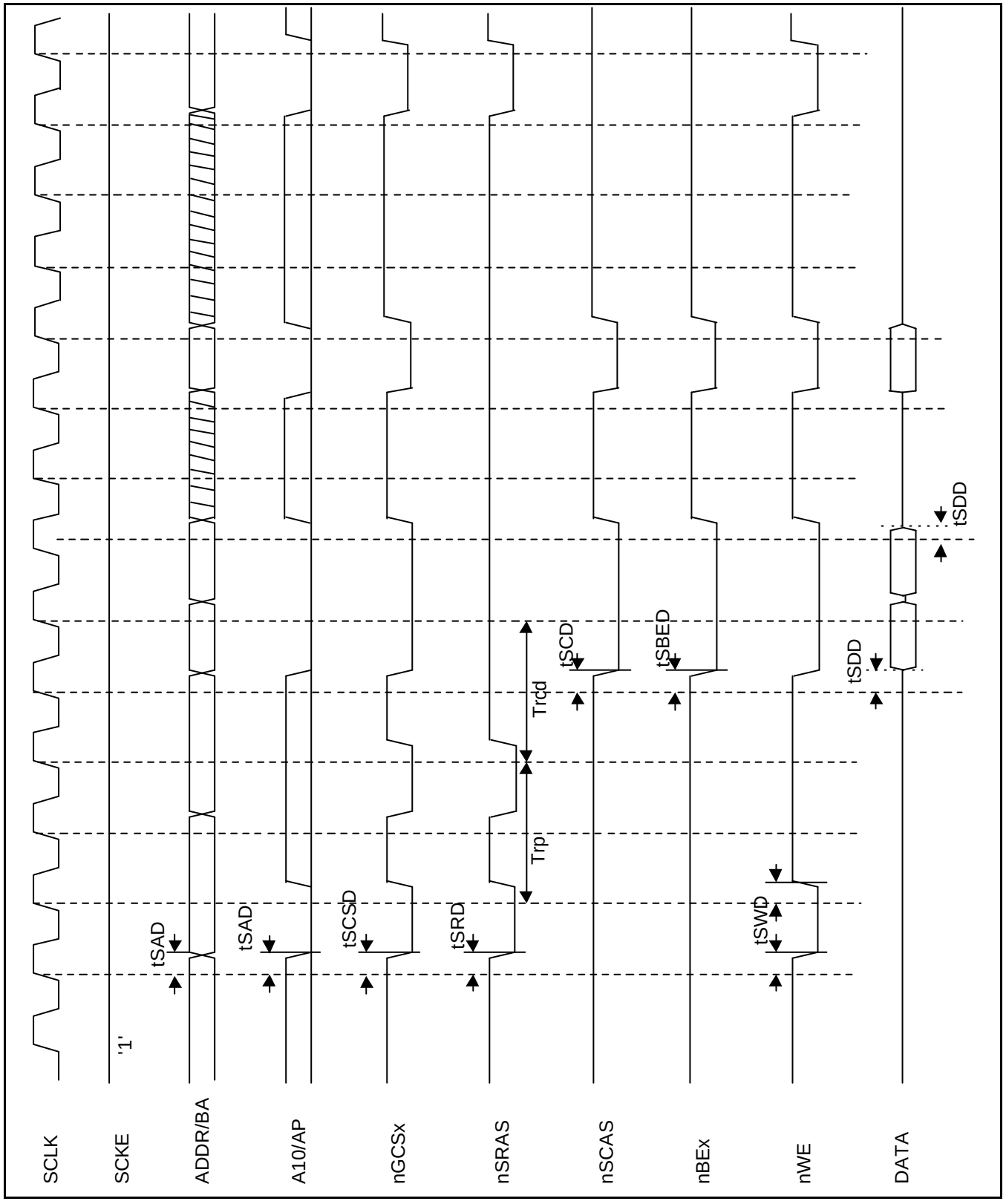


Figure 24-28. SDRAM Page Hit-Miss Write Timing (Trp = 2, Trcd = 2, Tcl = 2)

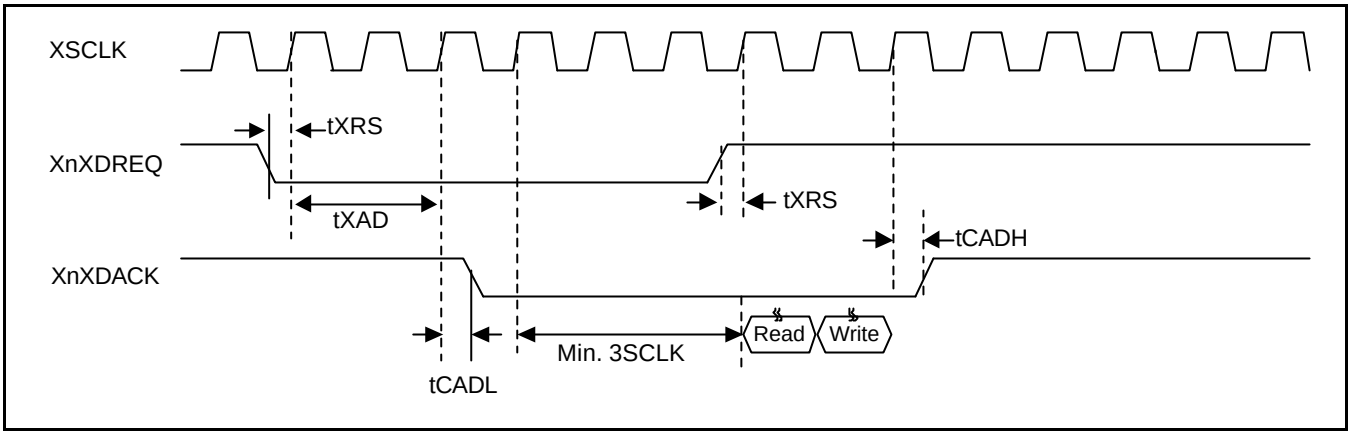


Figure 24-29. External DMA Timing (Handshake, Single transfer)

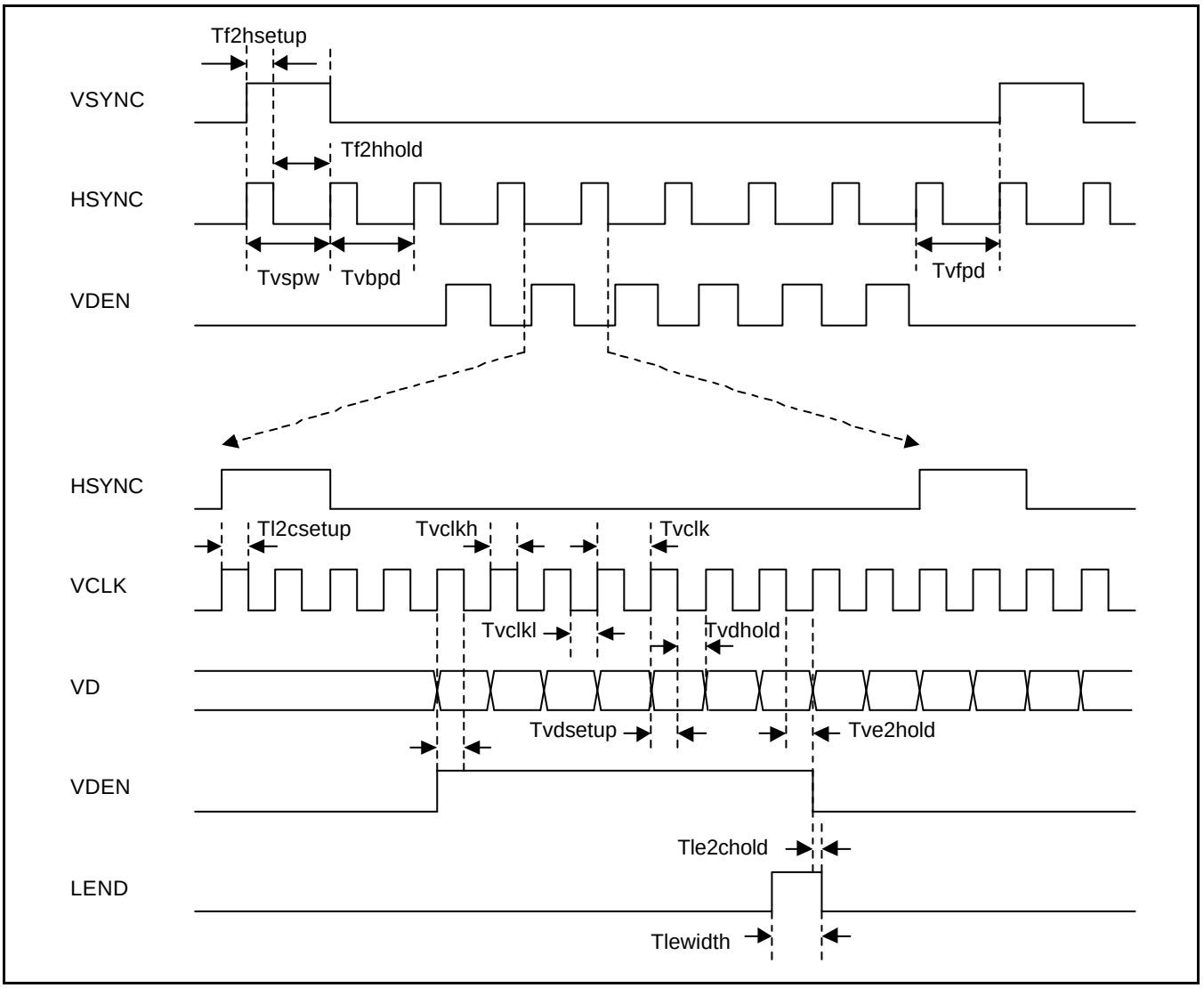


Figure 24-30. TFT LCD Controller Timing

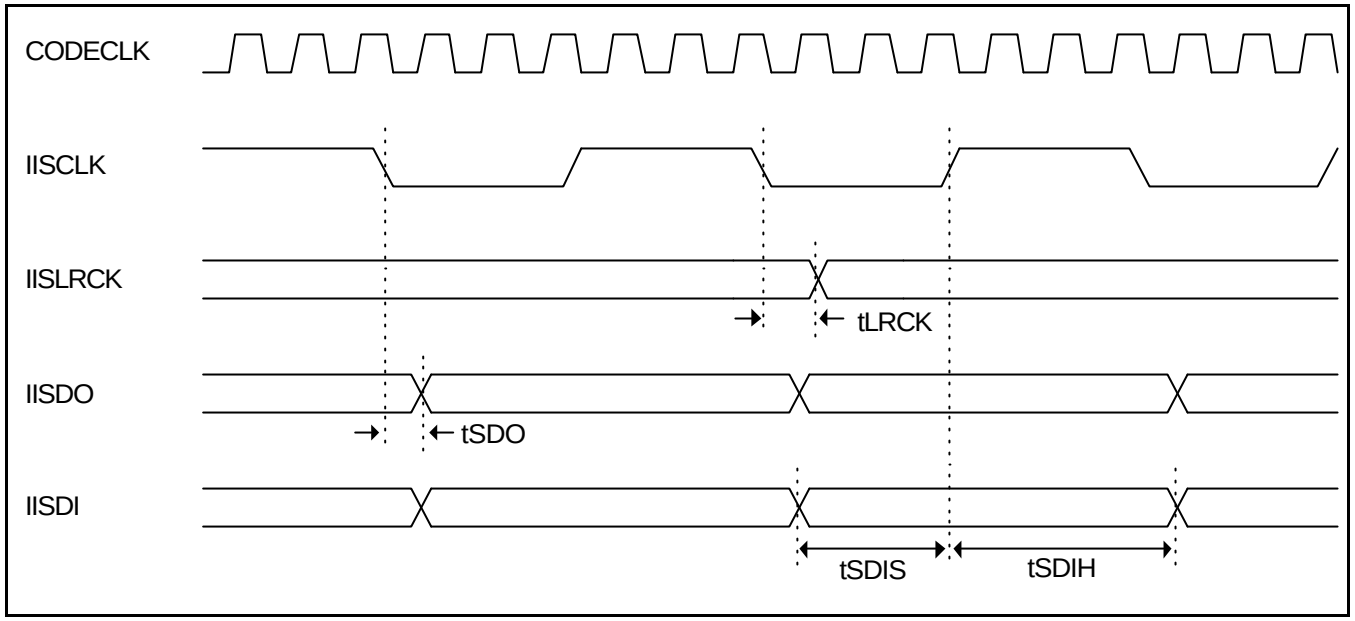


Figure 24-31. IIS Interface Timing

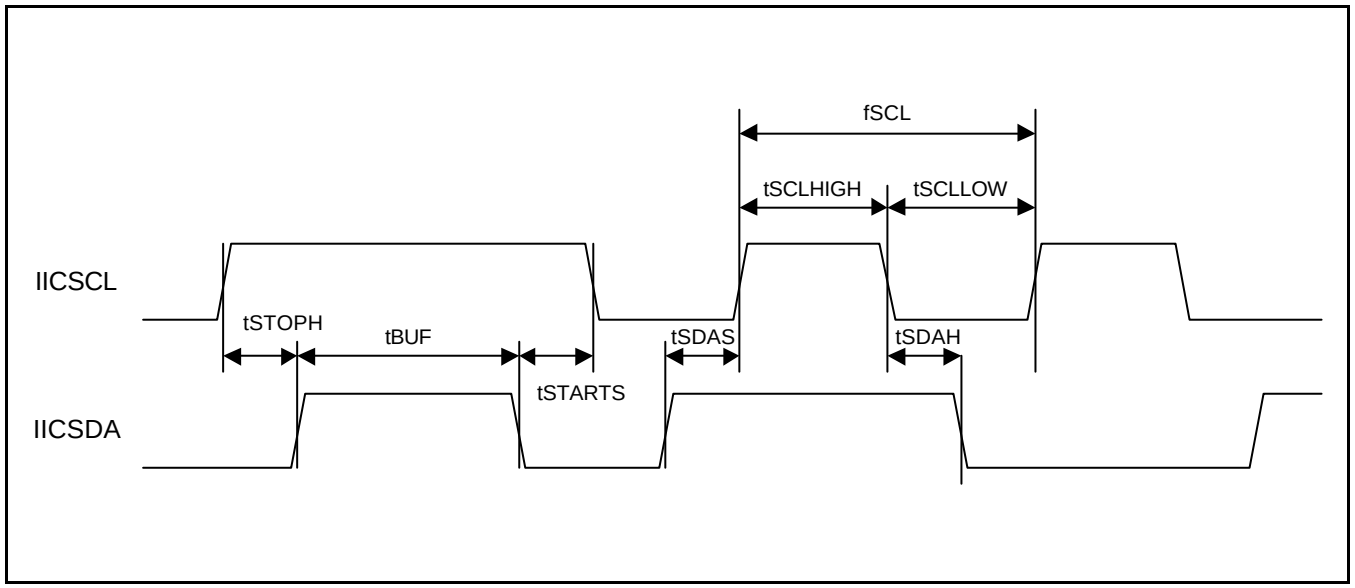


Figure 24-32. IIC Interface Timing

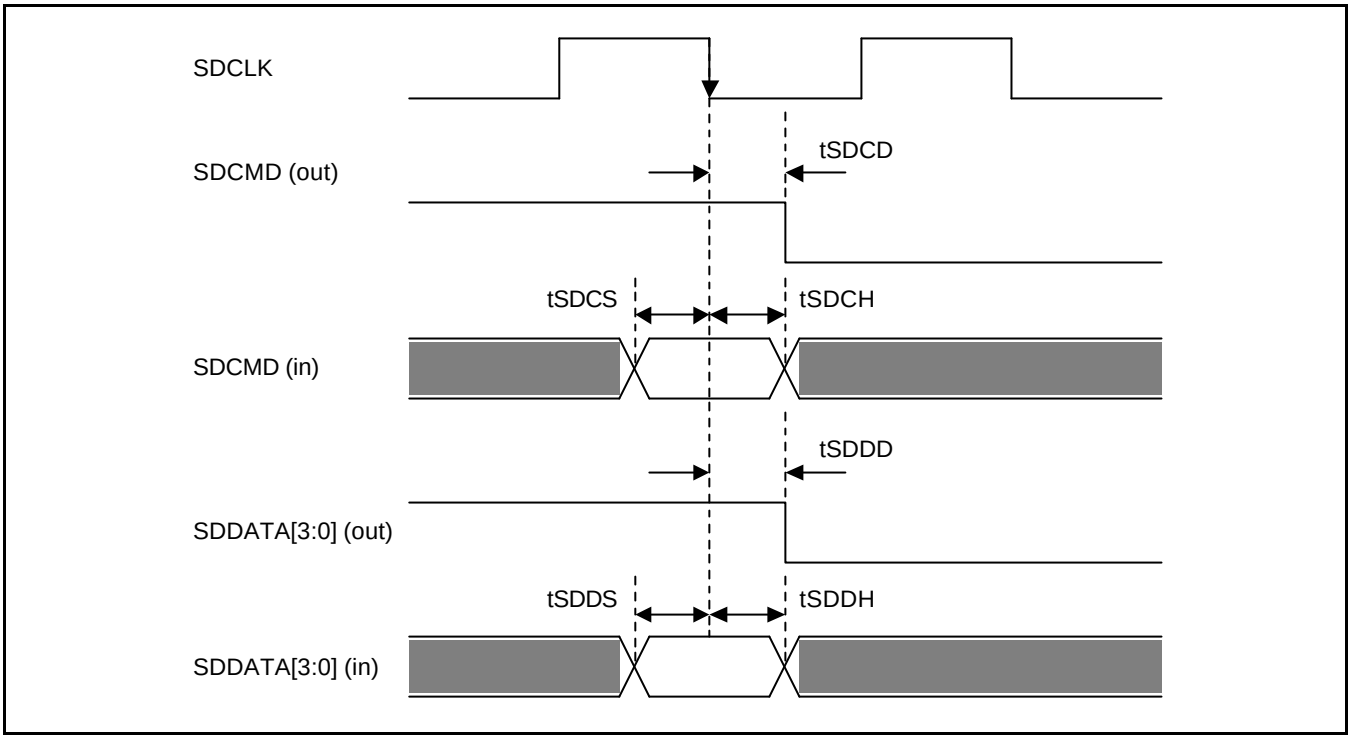


Figure 24-33. SD/MMC Interface Timing

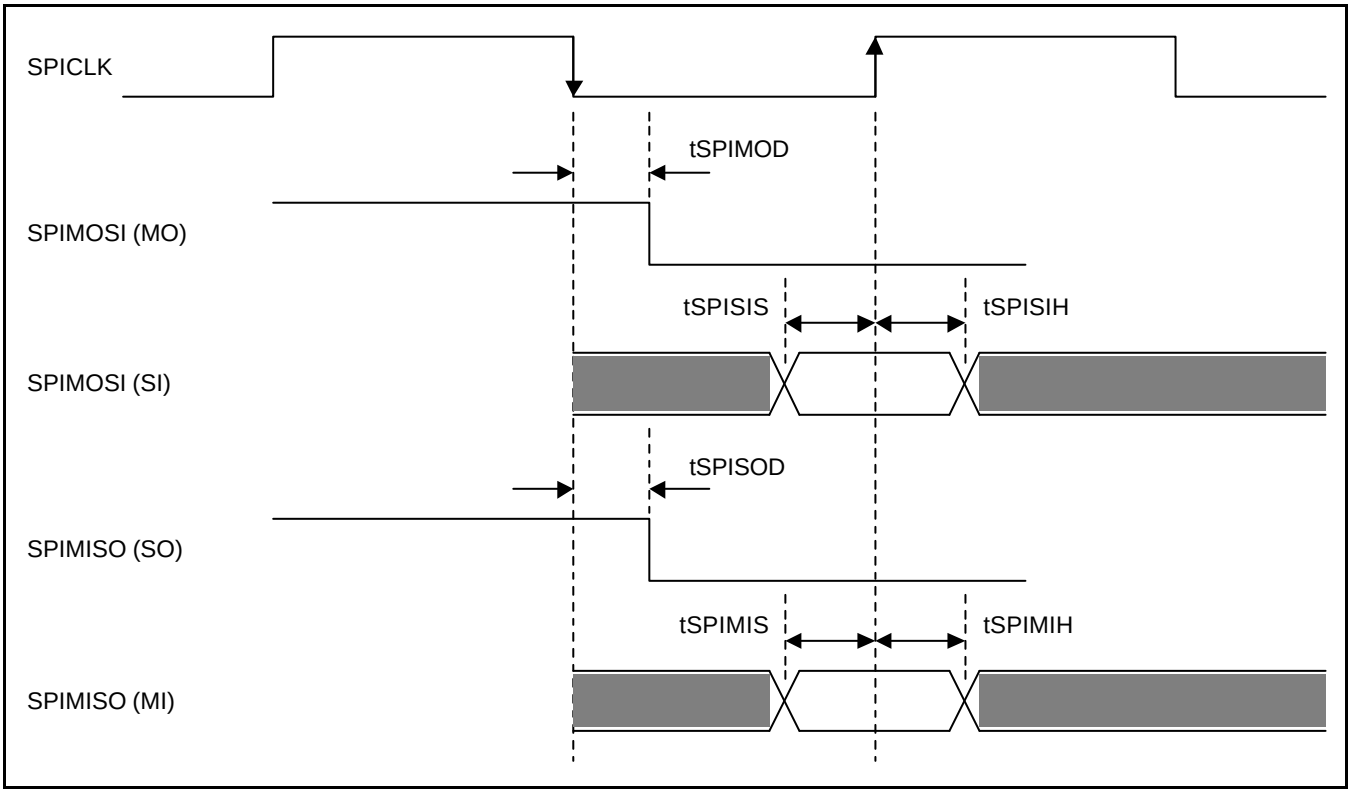


Figure 24-34. SPI Interface Timing (CPHA = 1, CPOL = 1)

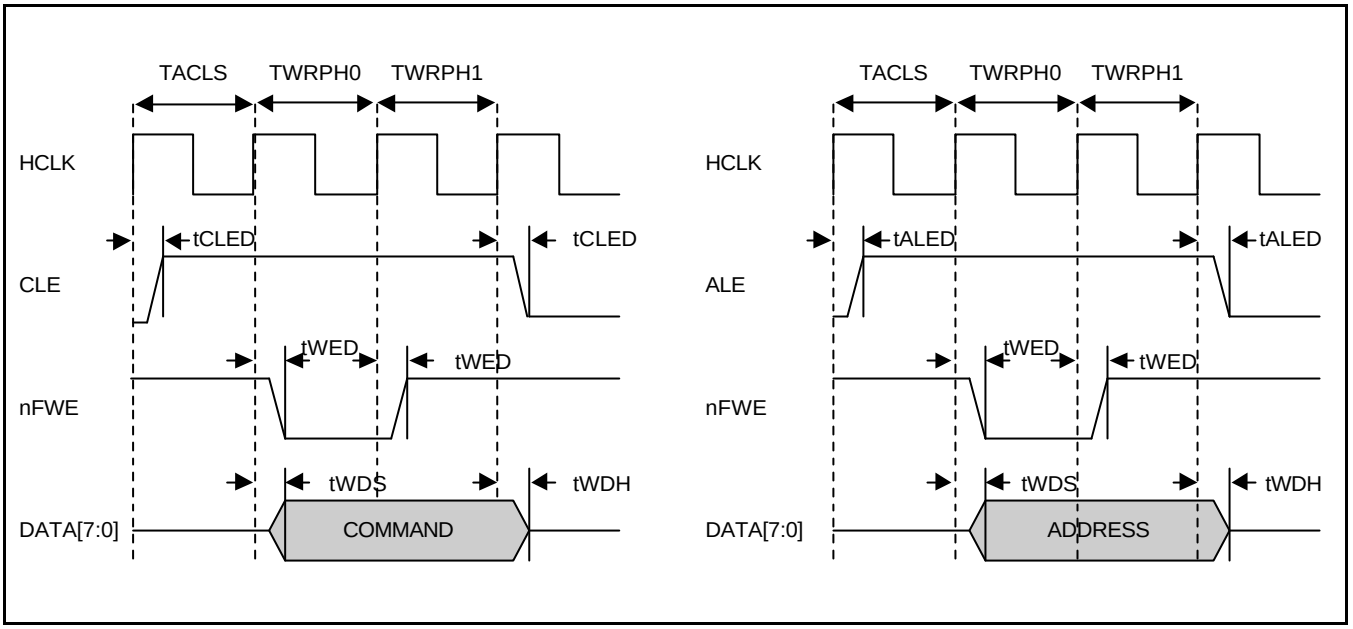


Figure 24-35. NAND Flash Address/Command Timing

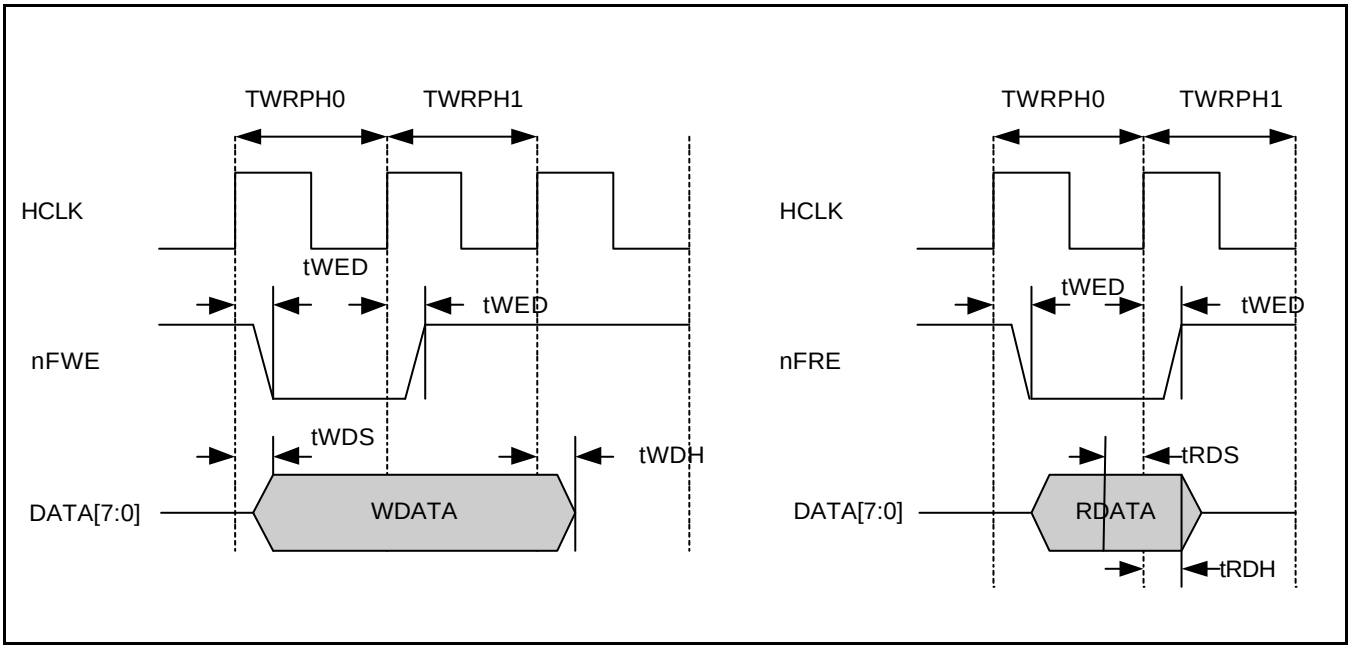


Figure 24-36. NAND Flash Timing

Table 24-6. Clock Timing Constants

(V_{DDi} =$V_{DDalive}$ =V_{DDiarm} = 1.8V ± 0.15 / 2.0 V ± 0.1 V, T_A = -40 to 85 °C, V_{DDMOP} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ	Max	Unit
Crystal clock input frequency	f_{XTAL}	10	–	20	MHz
Crystal clock input cycle time	$t_{XTALCYC}$	50	–	100	ns
External clock input frequency	f_{EXT}	–	–	66	MHz
External clock input cycle time	t_{EXTCYC}	15	–	–	ns
External clock input low level pulse width	t_{EXTLOW}	7	–	–	ns
External clock to HCLK (without PLL)	t_{EX2HC}	3	–	9	ns
HCLK (internal) to CLKOUT	t_{HC2CK}	3	–	11	ns
HCLK (internal) to SCLK	$t_{HC2SCLK}$	0	–	3	ns
External clock input high level pulse width	$t_{EXTHIGH}$	4	–	–	ns
Reset assert time after clock stabilization	t_{RESW}	4	–	–	XTIpll or EXTCLK
PLL Lock Time	t_{PLL}	200	–		uS
Power_OFF mode return oscillation setting time	t_{OSC2}	–	–	65536	XTIpll or EXTCLK
The interval before CPU runs after nRESET is released.	$t_{RST2RUN}$	–	7	–	XTIpll or EXTCLK

Table 24-7. ROM/SRAM Bus Timing Constants

(V_{DDi} =$V_{DDalive}$ =V_{DDiarm} = 1.8V ± 0.15 / 2.0 V ± 0.1 V, T_A = -40 to 85 °C, V_{DDMOP} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ	Max	Unit
ROM/SRAM Address Delay	t_{RAD}	3	–	11 / 10.5	ns
ROM/SRAM Chip select Delay	t_{RCD}	2	–	9 / 8.5	ns
ROM/SRAM Output enable Delay	t_{ROD}	2	–	8 / 7.5	ns
ROM/SRAM read Data Setup time.	t_{RDS}	4	–	–	ns
ROM/SRAM read Data Hold time.	t_{RDH}	0	–	–	ns
ROM/SRAM Byte Enable Delay	t_{RBED}	2	–	8 / 7.5	ns
ROM/SRAM Write Byte Enable Delay	t_{RWBED}	2	–	10 / 9.5	ns
ROM/SRAM output Data Delay	t_{RDD}	3	–	12 / 11.5	ns
ROM/SRAM external Wait Setup time	t_{WS}	5	–	–	ns
ROM/SRAM external Wait Hold time	t_{WH}	0	–	–	ns
ROM/SRAM Write enable Delay	t_{RWD}	2	–	9 / 8.5	ns

Table 24-8. Memory Interface Timing Constants (3.3V)

(V_{DDi} =$V_{DDalive}$ =V_{DDiarm} = 1.8V ± 0.15 / 2.0 V ± 0.1 V, T_A = -40 to 85 °C, V_{DDMOP} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ	Max	Unit
SDRAM Address Delay	t_{SAD}	2	–	7 / 6.5	ns
SDRAM Chip Select Delay	t_{SCSD}	2	–	6 / 5.5	ns
SDRAM Row active Delay	t_{SRD}	1	–	5 / 4.5	ns
SDRAM Column active Delay	t_{SCD}	1	–	5 / 4.5	ns
SDRAM Byte Enable Delay	t_{SBED}	2	–	6 / 5.5	ns
SDRAM Write enable Delay	t_{SWD}	2	–	6 / 5.5	ns
SDRAM read Data Setup time	t_{SDS}	4	–	–	ns
SDRAM read Data Hold time	t_{SDH}	0	–	–	ns
SDRAM output Data Delay	t_{SDD}	2	–	7 / 6.5	ns
SDRAM Clock Enable Delay	t_{CKED}	2	–	5 / 4.5	ns

Table 24-9. External Bus Request Timing Constants

(V_{DD} = 1.8V ± 0.15 / 2.0 V ± 0.1 V, T_A = -40 to 85 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ.	Max	Unit
eXternal Bus Request Setup time	t_{XnBRQS}	2	–	5 / 4	ns
eXternal Bus Request Hold time	t_{XnBRQH}	–	–	1 / 0.5	ns
eXternal Bus Ack Delay	$t_{XnBACKD}$	9	–	11 / 10	ns
HZ Delay	t_{HZD}	4	–	12 / 11	ns

Table 24-10. DMA Controller Module Signal Timing Constants

(V_{DD} = 1.8V ± 0.15 / 2.0 V ± 0.1 V, T_A = -40 to 85 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ.	Max	Unit
eXternal Request Setup	t_{XRS}	2	–	6 / 5	ns
aCcess to Ack Delay when Low transition	t_{CADL}	9	–	11 / 10	ns
aCcess to Ack Delay when High transition	t_{CADH}	9	–	11 / 10	ns
eXternal Request Delay	t_{XAD}	2	–	–	SCLK

Table 24-11. TFT LCD Controller Module Signal Timing Constants

(V_{DD} = 1.8V ± 0.15 / 2.0 V ± 0.1 V, T_A = -40 to 85 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ	Max	Units
Vertical sync pulse width	Tvspw	VSPW + 1	–	–	Phclk (note1)
Vertical back porch delay	Tvbpd	VBPD+1	–	–	Phclk
Vertical front porch dealy	Tvfpd	VFPD+1	–	–	Phclk
VCLK pulse width	Tvclk	1	–	–	Pvclk (note2)
VCLK pulse width high	Tvclkh	0.5	–	–	Pvclk
VCLK pulse width low	Tvclkl	0.5	–	–	Pvclk
Hsync setup to VCLK falling edge	Tl2csetup	0.5	–	–	Pvclk
VDEN set up to VCLK falling edge	Tde2csetup	0.5	–	–	Pvclk
VDEN hold from VCLK falling edge	Tde2chold	0.5	–	–	Pvclk
VD setup to VCLK falling edge	Tvd2csetup	0.5	–	–	Pvclk
VD hold from VCLK falling edge	Tvd2chold	0.5	–	–	Pvclk
LEND width	Tlewidth	–	1	–	Pvclk
LEND hold from VCLK rising edge	Tle2chold	3	–	–	ns
VSYNC setup to HSYNC falling edge	Tf2hsetup	HSPW + 1	–	–	Pvclk
VSYNC hold from HSYNC falling edge	Tf2hhold	HBPD + HFPD + HOZVAL + 3	–	–	Pvclk

NOTES:
1. HSYNC period
2. VCLK period

Table 24-12. IIS Controller Module Signal Timing Constants

(V_{DD} = 1.8V ± 0.15 / 2.0 V ± 0.1 V, T_A = -40 to 85 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ.	Max	Unit
IISLRCK delay time	t_{LRCK}	0.7	–	1.4	ns
IISDO delay time	t_{SDO}	0.8	–	1.7	ns
IISDI Input Setup time	t_{SDIS}	6.2	–	16.3 / 15.3	ns
IISDI Input Hold time	t_{SDIH}	0.1	–	0.1	ns
CODEC clock frequency	f_{CODEC}	1/16	–	1	f_{IIS_BLOCK}

Table 24-13. IIC BUS Controller Module Signal Timing

(V_{DD} = 1.8V ± 0.15 / 2.0 V ± 0.1 V, T_A = -40 to 85 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ.	Max	Unit
SCL clock frequency	f_{SCL}	–	–	std. 100 fast 400	kHz
SCL high level pulse width	$t_{SCLHIGH}$	std. 4.0 fast 0.6	–	–	µs
SCL low level pulse width	t_{SCLLOW}	std. 4.7 fast 1.3	–	–	µs
Bus free time between STOP and START	t_{BUF}	std. 4.7 fast 1.3	–	–	µs
START hold time	t_{STARTS}	std. 4.0 fast 0.6	–	–	µs
SDA hold time	t_{SDAH}	std. 0 fast 0	–	std. - fast 0.9	µs
SDA setup time	t_{SDAS}	std. 250 fast 100	–	–	ns
STOP setup time	t_{STOPH}	std. 4.0 fast 0.6	–	–	µs

NOTES: Std. means Standard Mode and fast means Fast Mode.

1. The IIC data hold time(tSDAH) is minimum 0ns.
   (IIC data hold time is minimum 0ns for standard/fast bus mode in IIC specification v2.1.)
   Please check the data hold time of your IIC device if it's 0 nS or not.
2. The IIC controller supports only IIC bus device(standard/fast bus mode), not C bus device.

Table 24-14. SD/MMC Interface Transmit/Receive Timing Constants

(V_{DD} = 1.8V ± 0.15 / 2.0 V ± 0.1 V, T_A = -40 to 85 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ.	Max	Unit
SD Command output Delay time	t_{SDCD}	0.5	–	1.3	ns
SD Command input Setup time	t_{SDCS}	5.8	–	15.2 / 14.2	ns
SD Command input Hold time	t_{SDCH}	-	–	0.1	ns
SD Data output Delay time	t_{SDDD}	0.3	–	0.6	ns
SD Data input Setup time	t_{SDDS}	6.3	–	15.3 / 14.3	ns
SD Data input Hold time	t_{SDDH}	-	–	0.1	ns

Table 24-15. SPI Interface Transmit/Receive Timing Constants

(V_{DD} = 1.8V ± 0.15 / 2.0 V ± 0.1 V, T_A = -40 to 85 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ.	Max	Unit
SPI MOSI Master Output Delay time	t_{SPIMOD}	1.0	–	4.2	ns
SPI MOSI Slave Input Setup time	t_{SPISIS}	0.1	–	0.1	ns
SPI MOSI Slave Input Hold time	t_{SPISIH}	0.8	–	1.8	ns
SPI MISO Slave output Delay time	t_{SPISOD}	8.2	–	21.4 / 20.4	ns
SPI MISO Master Input Setup time	t_{SPIMIS}	5.6	–	14.7 / 13.7	ns
SPI MISO Master Input Hold time	t_{SPIMIH}	0.1	–	0.1	ns

Table 24-16. USB Electrical Specifications

(V_{DD} = 1.8V ± 0.15 / 2.0 V ± 0.1 V, T_A = -40 to 85 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Condition	Min	Max	Unit
Supply Current					
Suspend Device	ICCS			10	µA
Leakage Current					
Hi-Z state Input Leakage	ILO	0V < VIN < 3.3V	-10	10	µA
Input Levels					
Differential Input Sensitivity	VDI	| (D+) – (D-) |	0.2		V
Differential Common Mode Range	VCM	Includes VDI range	0.8	2.5	
Single Ended Receiver Threshold	VSE		0.8	2.0	
Output Levels					
Static Output Low	VOL	RL of 1.5Kohm to 3.6V		0.3	V
Static Output High	VOH	RL of 15Kohm to GND	2.8	3.6	
Capacitance					
Transceiver Capacitance	CIN	Pin to GND		20	pF

Table 24-17. USB Full Speed Output Buffer Electrical Characteristics

(V_{DD} = 1.8V ± 0.15 / 2.0 V ± 0.1 V, T_A = -40 to 85 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Condition	Min	Max	Unit
Driver Characteristics					
Transition Time Rise Time Fall Time	 TR TF	 CL = 50pF CL = 50pF	 4.0 4.0	 2.0 2.0	 ns
Rise/Fall Time Matching	TRFM	(TR / TF)	90	110	%
Output Signal Crossover Voltage	VCRS		1.3	2.0	V
Drive Output Resistance	ZDRV	Steady state drive	28	43	ohm

Table 24-18. USB Low Speed Output Buffer Electrical Characteristics

(V_{DD} = 1.8V ± 0.15 / 2.0 V ± 0.1 V, T_A = -40 to 85 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Condition	Min	Max	Unit
Driver Characteristics					
Transition Time Rising Time Falling Time 	 TR TF 	 CL = 50pF CL = 350pF CL = 50pF CL = 350pF	 75 75 	 300 300	 ns
Rise/Fall Time Matching	TRFM	(TR / TF)	80	120	%
Output Signal Crossover Voltage	VCRS		1.3	2.0	V

Table 24-19. NAND Flash Interface Timing Constants

(V_{DDi} =$V_{DDalive}$ =V_{DDiarm} = 1.8V ± 0.15 / 2.0 V ± 0.1 V, T_A = -40 to 85 °C, V_{DDIO} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ	Max	Unit
NFCON Chip Enable delay	t_{CED}	–	–	6.2 / 5.2	ns
NFCON CLE delay	t_{CLED}	–	–	7.1 / 6.1	ns
NFCON ALE delay	t_{ALED}	–	–	7.5 / 6.5	ns
NFCON Write Enable delay	t_{WED}	–	–	7.2 / 6.2	ns
NFCON Read Enable delay	t_{RED}	–	–	7.1 / 6.1	ns
NFCON Write Data Setup time	t_{WDS}	–	–	6.5 / 5.5	ns
NFCON Write Data Hold time	t_{WDH}	1.7	–	–	ns
NFCON Read Data Setup requirement time	t_{RDS}	0.3	–	–	ns
NFCON Read Data Hold requirement time	t_{RDH}	0.3	–	–	ns

25 MECHANICAL DATA

PACKAGE DIMENSIONS

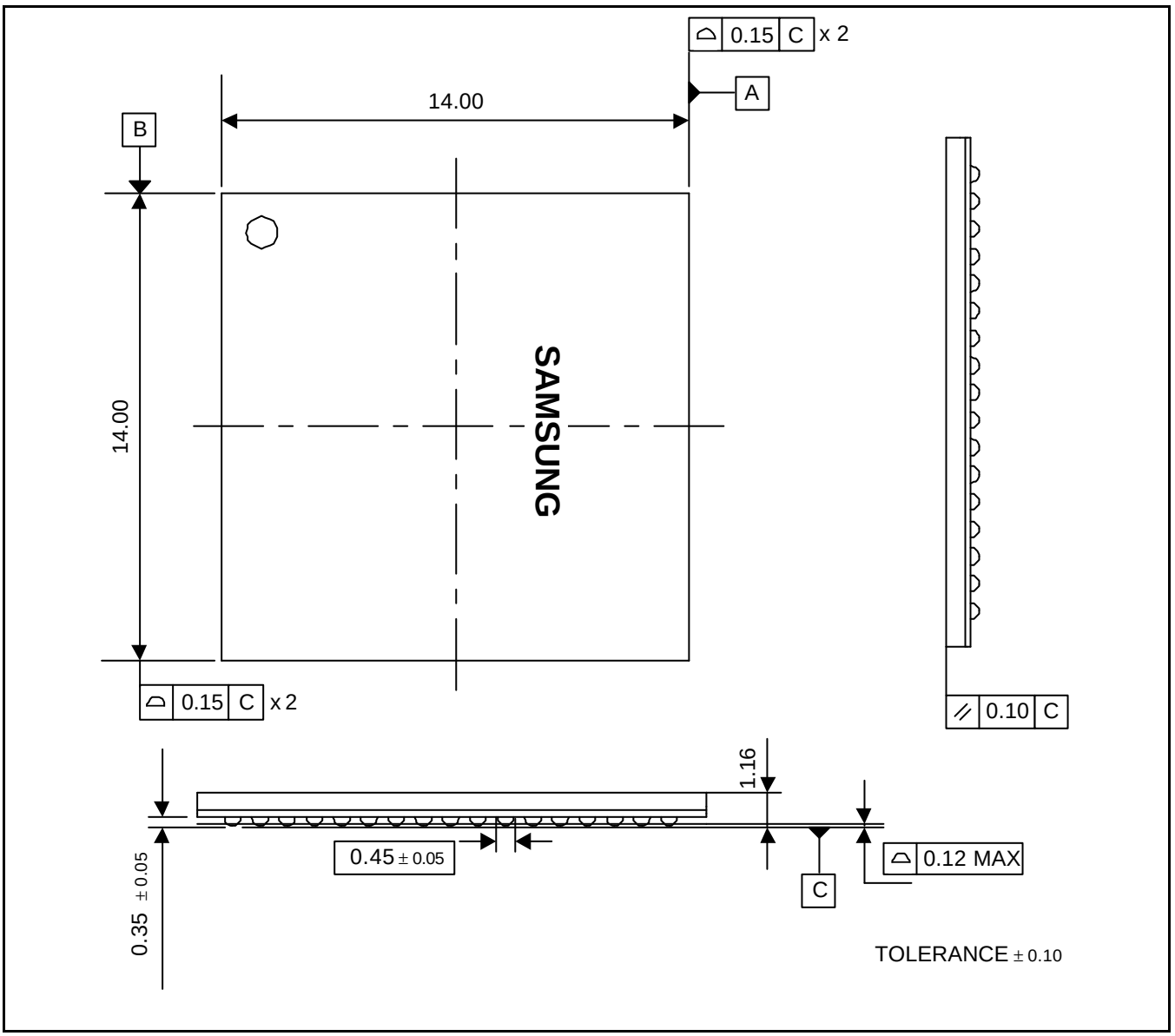


Figure 25-1. 272-FBGA-1414 Package Dimension 1 (Top View)

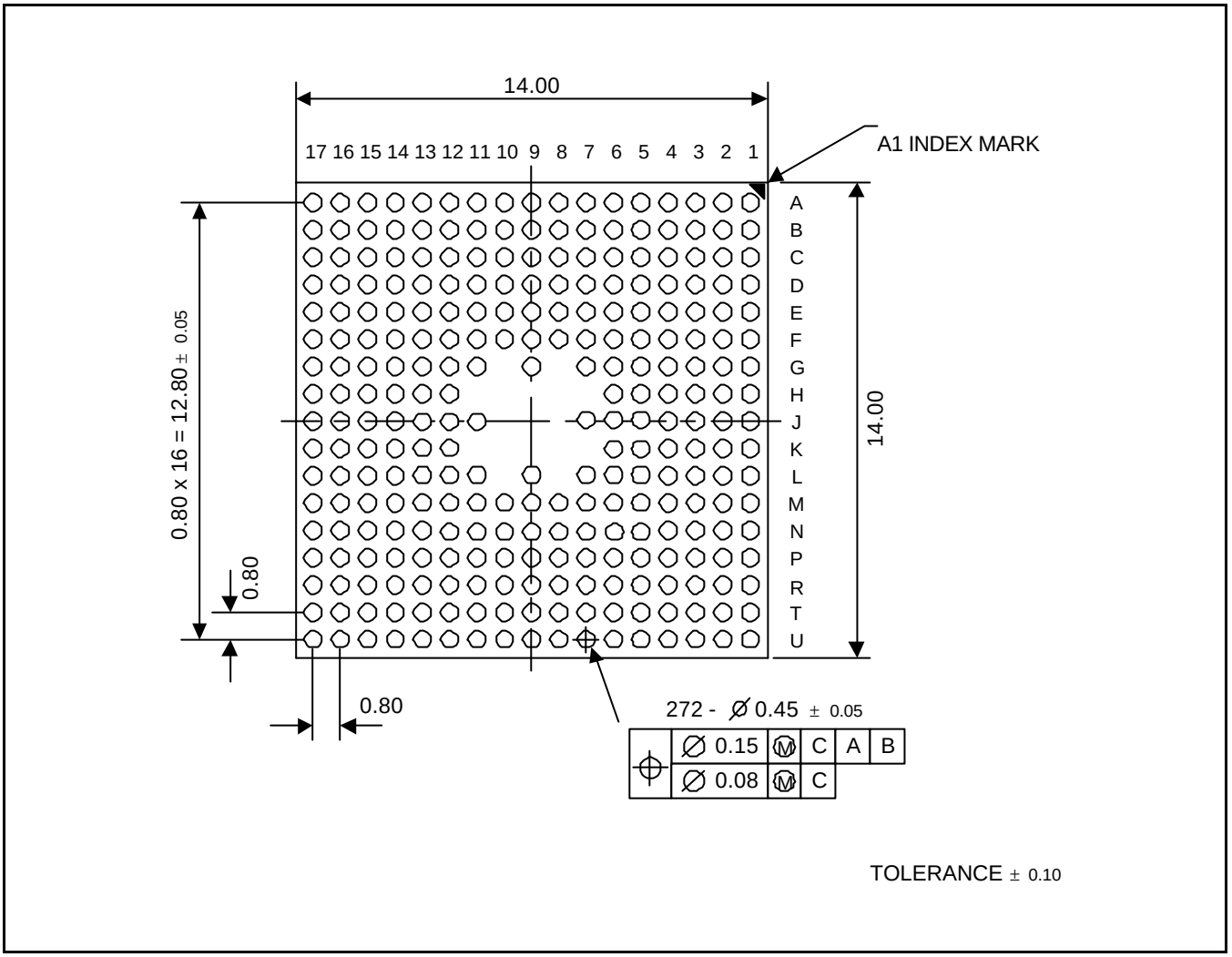


Figure 25-2. 272-FBGA-1414 Package Dimension 2 (Bottom View)

The recommended land open size is 390 – 410 μm (0.39 – 0.41mm) diameter.

Appendix 1

ARM920T INTRODUCTION

ABUOT THE INTRODUCTION

The ARM920T is a member of the ARM9TDMI family of general-purpose microprocessors, which includes:

— ARM9TDMI (ARM9TDMI core)

— ARM940T (ARM9TDMI core plus cache and protection unit)

— ARM920T (ARM9TDMI core plus cache and MMU).

The ARM9TDMI processor core is a Harvard architecture device implemented using a five-stage pipeline consisting of fetch, decode, execute, memory and write stages, and can be provided as a stand-alone core which can be embedded into more complex devices. The stand-alone core has a simple bus interface that allows users to design their own caches/memory systems around it.

The ARM9TDMI family of microprocessors supports both the 32-bit ARM and 16-bit Thumb instruction sets, allowing the user to trade off between high performance and high code density.

The ARM920T is a Harvard cache architecture processor which is targeted at multiprogrammer applications where full memory management, high performance, and low power are all-important. The separate instruction and data caches in this design are 16KB each in size, with an 8-word line length. The ARM920T implements an enhanced ARM Architecture V4 MMU to provide translation and access permission checks for instruction and data addresses.

The ARM920T supports the ARM debug architecture and includes logic to assist in both hardware and software debug. The ARM920T also includes support for coprocessors, exporting the instruction and data buses along with simple handshaking signals.

The ARM920T interface to the rest of the system is via unified address and data buses. This interface is compatible with the Advanced Microcontroller Bus Architecture (AMBA) bus scheme, either as a fully compliant AMBA bus master, or as a slave for production test. The ARM920T also has a TrackingICE mode which allows an approach similar to a conventional ICE mode of operation.

PROCESSOR FUNCTIONAL BLOCK DIAGRAM

Shows the functional block diagram of the ARM920T

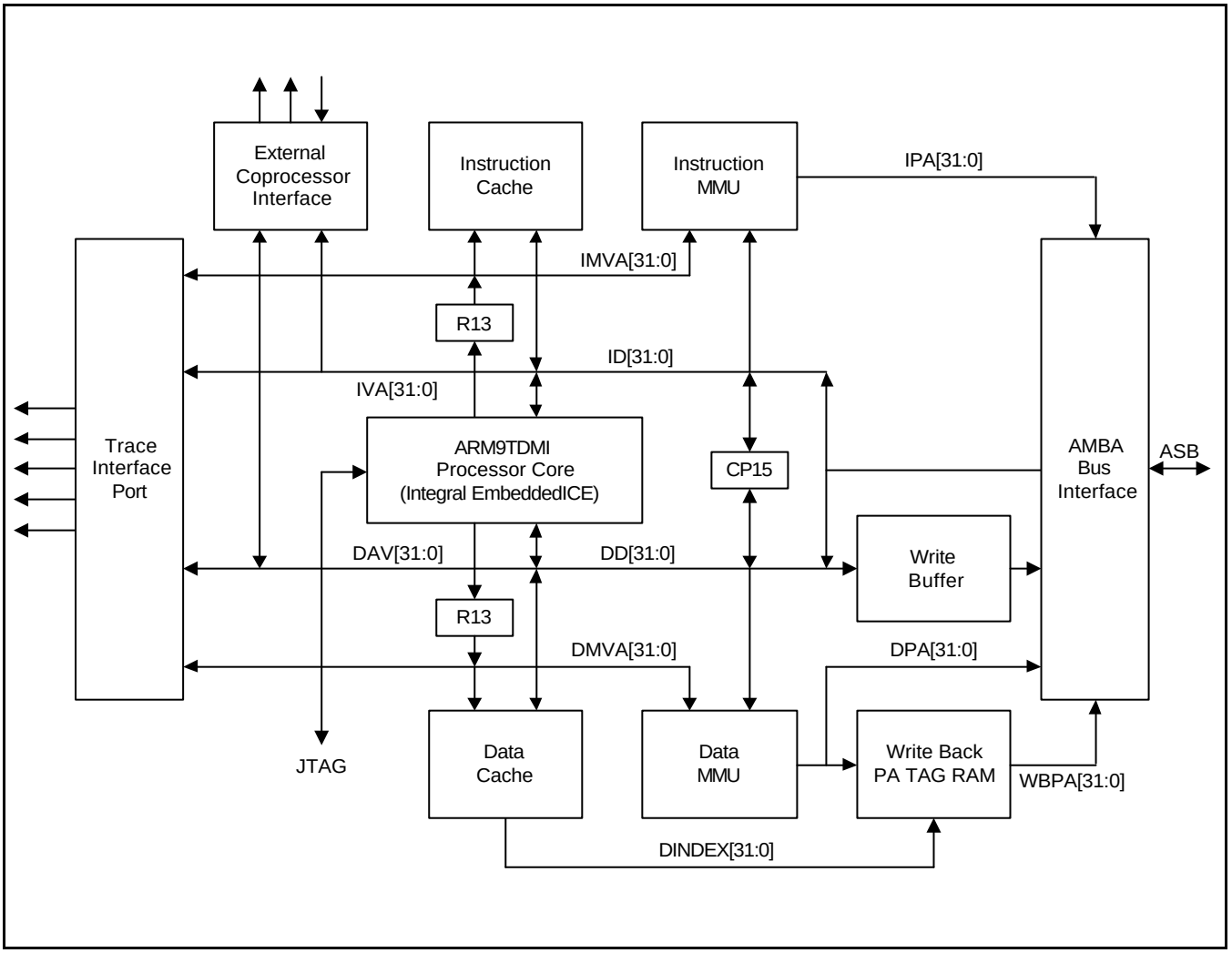


Figure 1-1. ARM920T Functional Block Diagram

Appendix 2

PROGRAMMER'S MODEL

ABOUT THE PROGRAMMER'S MODEL

ARM920T incorporates the ARM9TDMI integer core, which implements the ARMv4T architecture. It executes the ARM and Thumb instruction sets, and includes Embedded ICE JTAG software debug features.

The programmer's model of the ARM920T consists of the programmer's model of the ARM9TDMI with the following additions and modifications:

- The ARM920T incorporates two coprocessors:
  - — CP14, which allows software access to the debug communications channel. The registers defined in CP14 are accessible with MCR and MRC instructions.
  - — The system control coprocessor (CP15), which provides additional registers that are used to configure and control the caches, MMU, protection system, the clocking mode and other system options of the ARM920T, such as big or little-endian operation. The registers defined in CP15 are accessible with MCR and MRC instructions. These are described in CP15 register map summary on page 2-4.
- The ARM920T also features an external coprocessor interface which allows the attachment of a closely coupled coprocessor on the same chip, for example, a floating point unit. Registers and operations provided by any coprocessors attached to the external coprocessor interface will be accessible with appropriate coprocessor instructions.
- Memory accesses for instruction fetches and data loads and stores may be cached or buffered. Cache and write buffer configuration and operation is described in detail in following chapters.
- The MMU page tables which reside in main memory describe the virtual to physical address mapping, access permissions, and cache and write buffer configuration. These are created by the operating system software and accessed automatically by the ARM920T MMU hardware whenever an access causes a TLB miss.
- The ARM920T has a Trace Interface Port which allows the use of Trace hardware and tools for real-time tracing of instructions and data.

ABOUT THE ARM9TDMI PROGRAMMER'S MODEL

The ARM9TDMI processor core implements ARM v4T architecture, and so executes the ARM 32-bit instruction set and the compressed Thumb 16-bit instruction set. The programmer's model is fully described in the ARM Architecture Reference Manual. The ARM9TDMI Technical Reference Manual gives implementation details including instruction execution cycle times.

The ARM v4T architecture specifies a small number of implementation options. The options selected in the ARM9TDMI implementation are listed in Table 2-1. For comparison, the options selected for the ARM7TDMI implementation are also shown

Table 2-1. ARM9TDMI Implementation Option

Processor core	ARM architecture	Data abort model	Value stored by direct STR, STRT, STM of PC
ARM7TDMI	v4T	Base updated	Address of Inst + 12
ARM9TDMI	v4T	Base restored	Address of Inst + 12

The ARM9TDMI is code-compatible with the ARM7TDMI, with two exceptions:

- The ARM9TDMI implements the *base restored data abort model*, which significantly simplifies the software data abort handler.
- The ARM9TDMI fully implements the instruction set extension spaces added to the ARM (32-bit) instruction set in architecture v4 and v4T.

These differences are explained in more detail below.

DATA ABORT MODEL

The base restored data abort model differs from the base updated data abort model implemented by ARM7TDMI.

The difference in the data abort model affects only a very small section of operating system code, the data abort handler. It does not affect user code. With the base restored data abort model, when a data abort exception occurs during the execution of a memory access instruction, the base register is always restored by the processor hardware to the value the register contained before the instruction was executed. This removes the need for the data abort handler to *unwind* any base register update which may have been specified by the aborted instruction.

INSTRUCTION SET EXTENSION SPACES

All ARM processors implement the undefined instruction space as one of the entry mechanisms for the undefined instruction exception. That is, ARM instructions with opcode[27:25] = 0b011 and opcode[4] = 1 are undefined on all ARM processors including the ARM9TDMI and ARM7TDMI.

ARM architecture v4 and v4T also introduced a number of instruction set extension spaces to the ARM instruction set. These are:

- arithmetic instruction extension space
- control instruction extension space
- coprocessor instruction extension space
- load/store instruction extension space.

Instructions in these spaces are undefined (they cause an undefined instruction exception). The ARM9TDMI fully implements all the instruction set extension spaces defined in ARM architecture v4T as undefined instructions, allowing emulation of future instruction set additions.

The system control coprocessor (CP15) allows configuration and control of the caches, MMU, protection system and clocking mode of the ARM920T.

The ARM920T coprocessor 15 registers are described under the following sections:

- Accessing CP15 registers on page 2-5
- Register 0: ID code register on page 2-7
- Register 0: Cache type register on page 2-8
- Register 1: Control register on page 2-10
- Register 2: Translation table base (TTB) register on page 2-12
- Register 3: Domain access control register on page 2-13
- Register 4: Reserved on page 2-14
- Register 5: Fault status registers on page 2-14
- Register 6: Fault address register on page 2-15
- Register 7: Cache operations on page 2-15
- Register 8: TLB operations on page 2-18
- Register 9: Cache lock down register on page 2-19
- Register 10: TLB lock down register on page 2-21
- Registers 11 -12 & 14: Reserved on page 2-22
- Register 13: Process ID on page 2-22
- Addresses in ARM920T on page 2-6
- Register 15: Test configuration register on page 2-24.

CP15 REGISTER MAP SUMMARY

CP15 defines 16 registers. The register map for CP15 is shown in Table 2-2

Table 2-2. CP15 Register Map

Register	Read	Write
0	ID code (1)	Unpredictable
0	Cache type (1)	Unpredictable
1	Control	Control
2	Translation table base	Translation table base
3	Domain access control	Domain access control
4	Unpredictable	Unpredictable
5	Fault status (2)	Fault status (2)
6	Fault address	Fault address
7	Unpredictable	Cache operations
8	Unpredictable	TLB operations
9	Cache lockdown (2)	Cache lockdown (2)
10	TLB lock down (2)	TLB lock down (2)
11	Unpredictable	Unpredictable
12	Unpredictable	Unpredictable
13	Process ID	Process ID
14	Unpredictable	Unpredictable
15	Test configuration	Test configuration

NOTES:
1. Register location 0 provides access to more than one register. The register accessed depends upon the value of the opcode_2 field. See the register description for details.
2. Separate registers for instruction and data. See the register description for details.

ACCESSING CP15 REGISTERS

Throughout this section the following terms and abbreviations are used.

Table 2-3. CP15 Abbreviations

Term	Abbreviation	Description
Unpredictable	UNP	For reads, the data returned when reading from this location is unpredictable; it could have any value. For writes, writing to this location will cause unpredictable behavior, or an unpredictable change in device configuration.
Should be zero	SBZ	When writing to this location, all bits of this field should be 0.

In all cases, reading from, or writing any data values to any CP15 registers, including those fields specified as unpredictable or should be zero, will not cause any permanent damage.

All CP15 register bits that are defined and contain state, are set to zero by BnRES except V-Bit in register 1, which takes the value of macrocell input VINITHI when BnRES is asserted.

CP15 registers can only be accessed with MRC and MCR instructions in a privileged mode. The instruction bit pattern of the MCR and MRC instructions is shown in Figure 2-1. The assembler for these instructions is

MCR/MRC{cond} P15,opcode_1,Rd,CRn,CRm,opcode_2

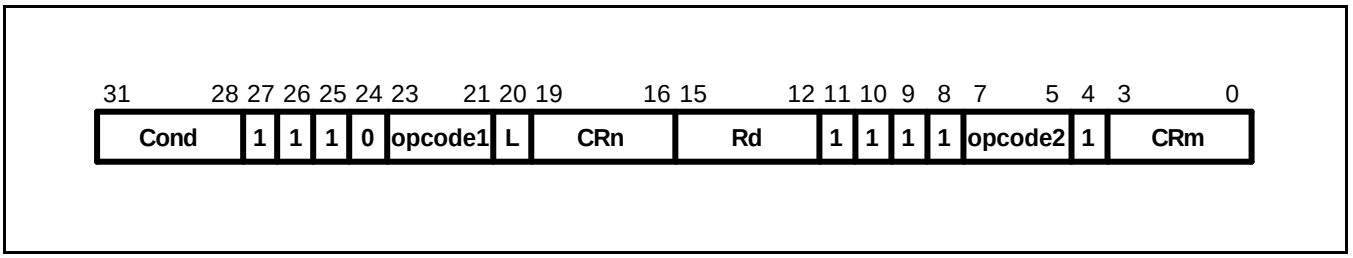


Figure 2-1. CP15 MRC and MCR Bit Pattern

Instructions CDP, LDC and STC, along with unprivileged MRC and MCR instructions to CP15 will cause the undefined instruction trap to be taken. The CRn field of MRC and MCR instructions specifies the coprocessor register to access. The CRm field and opcode_2 field are used to specify a particular action when addressing registers.

Attempting to read from a non-readable register, or writing to a non-writable register will cause unpredictable results.

The opcode_1, opcode_2 and CRm fields should be zero, except when the values specified are used to select the desired operations, in all instructions which access CP15. Using other values will result in unpredictable behavior.

Addresses in ARM920T

Three distinct types of address exist in an ARM920T system:

- virtual address (VA)
- modified virtual address (MVA)
- physical address (PA).

Below is an example of the address manipulation when the ARM9TDMI requests an instruction.

1) The VA of the instruction (IVA) is issued by the ARM9TDMI.
2) This is translated by the ProcID to the instruction MVA (IMVA). It is the IMVA that the instruction cache and MMU see.
3) If the protection check carried out by the IMMU on the IMVA does not abort and the IMVA tag is in the instruction cache the instruction data is returned to the ARM9TDMI.
4) If the instruction cache misses (the IMVA tag is not in the instruction cache) then the IMMU performs a translation to produce the instruction PA (IPA). This address is given to the AMBA bus interface to perform an external access

Table 2-4. Address Types in ARM920

Domain	Domain	Caches & TLBs	AMBA bus
Address	Virtual	Modified Virtual	Physical

REGISTER 0: ID CODE REGISTER

This is a read-only register which returns a 32-bit device ID code.

The ID code register is accessed by reading CP15 register 0 with the opcode_2 field set to any value other than 1 (the CRm field should be zero when reading). For example:

MRC p15,0,Rd,c0,c0,0; returns ID register

The contents of the ID code are shown in Table 2-5.

Table 2-5. Register 0: ID Code

Register bits	Function	Value
31:24	Implementor	0x41
23:20	Specification revision	0x1
19:16	Architecture version (4T)	0x2
15:4	Part number	0x920
3:0	Layout revision	0x0

REGISTER 0: CACHE TYPE REGISTER

This is a read-only register which contains information about the size and architecture of the caches, allowing operating systems to establish how to perform such operations as cache cleaning and lockdown. Future ARM cached processors will contain this register, allowing RTOS vendors to produce future-proof versions of their operating systems.

The cache type register is accessed by reading CP15 register 0 with the opcode_2 field set to 1. For example:

MRC p15,0,Rd,c0,c0,1; returns cache details

The format of the register is shown in Table 2-6.

Table 2-6. Cache Type Register Format

Register Bits	Function	Value
31:29	Reserved	000
28:25	Cache type	0110
24	Harvard/Unified	1 (defines Harvard cache)
23:21	Reserved	000
20:18	DCache size	101 (defines 16KB)
17:15	DCache associativity	110 (defines 64 way)
14	Reserved	0
13:12	DCache words per line	10 (defines 8 words per line)
11:9	Reserved	000
8:6	ICache size	101 (defines 16KB)
5:3	ICache Associativity	110 (defines 64 way)
2	Reserved	0
1:0	ICache words per line	10 (defines 8 words per line)

Bits [28:25] indicate which major cache class the implementation falls into. 0x6 means that the cache provides:

- Cache-clean-step operation
- Cache-flush-step operation
- Lock down facilities

Bits [20:18] give the data cache size. Bits [8:6] give the instruction cache size.
Table 2-7 on page 2-9 shows the meaning of values used for cache size encoding.

Table 2-7. Cache Size Encoding

Bits [20:18]/Bits [8:6]	Cache Size
000	512B
001	1KB
010	2KB
011	4KB
100	8KB
101	16KB
110	32KB
111	64KB

Bits [17:15] give the data cache associativity. Bits [5:3] give the instruction cache associativity. Table 2-8 on page 2-9 shows the meaning of values used for cache associativity encoding.

Table 2-8. Cache associativity encoding

Bits [17:15]/Bits [5:3]	Associativity
000	Direct mapped
001	2
010	4
011	8
100	16
101	32
110	64
111	128

Bits [13:12] give the data cache line length. Bits [1:0] give the instruction cache line length.

Table 2-9 shows the meaning of values used for line length encoding

Table 2-9. Line Length Encoding

Bits [13:12]/Bits [1:0]	Words Per Line
00	2
01	4
10	8
11	16

REGISTER 1: CONTROL REGISTER

This register contains the control bits of the ARM920T. All reserved bits should either be written with zero or one, as indicated, or written using read-modify-write. The reserved bits have an unpredictable value when read. To read and write this register:

MRC p15, 0, Rd, c1, c0, 0; read control register
MCR p15, 0, Rd, c1, c0, 0; write control register

All defined control bits are set to zero on reset except the V-Bit which is set to zero at reset if the VINITHI pin is LOW, or one if the VINITHI pin is HIGH. The functions of the control bits are shown in Table 2-10.

Table 2-10. Control Register 1-bit Functions

Register Bits	Name	Function	Value
31	iA bit	Asynchronous clock select	See Table 2-11 on page 2-11.
30	nF bit	notFastBus select	See Table 2-11 on page 2-11.
29:15	–	Reserved	Read = Unpredictable Write = Should be zero
14	RR bit	Round robin replacement	0 = Random replacement 1 = Round robin replacement
13	V bit	Base location of exception registers	0 = Low addresses = 0x0000 0000 1 = High addresses = 0xFFFF 0000
12	I bit	Instruction cache enable	0 = Instruction cache disabled 1 = Instruction cache enabled
11:10	–	Reserved	Read = 00 Write = 00
9	R bit	ROM protection	This bit modifies the MMU protection system. See Table 3-6 on page 3-20
8	S bit	System protection	This bit modifies the MMU protection system. See Table 3-6 on page 3-20
7	B bit	Big-endian/little-endian	0 = Little-endian operation 1 = Big-endian operation
6:3	–	Reserved	Read = 1111 Write = 1111
2	C bit	Data cache enable	0 = Data cache disabled 1 = Data cache enabled
1	A bit	Alignment fault enable	Data address alignment fault checing. 0 = Fault checking disabled 1 = Fault checking enabled
0	M bit	MMU enable	0 = MMU disabled 1 = MMU enabled

Register 1 bits 31:30 select the clocking mode of the ARM920T, as shown in Table 2-11.

Table 2-11. Clocking Modes

Clocking Mode	iA	nF
FastBus mode	0	0
Reserved	1	0
Synchronous	0	1
Asynchronous	1	1

Enabling the MMU

Care must be taken with the address mapping of the code sequence used to enable the MMU, see Enabling the MMU on page 3-25.

See Instruction cache enable/disable on page 4-3 and Data cache and write buffer enable/disable on page 4-6 for restrictions and effects of having caches enabled with the MMU disabled

REGISTER 2: TRANSLATION TABLE BASE (TTB) REGISTER

This is the translation table base register, for the currently active first level translation table. The contents of register 2 are shown in Table 2-12.

Table 2-12. Register 2: Translation Table Base

Register Bits	Function
31:14	Pointer to first level translation table base. Read/write
13:0	Reserved Read = Unpredictable Write = Should be zero

Reading from register 2 returns the pointer to the currently active first level translation table in bits[31:14]. Writing to register 2 updates the pointer to the first level translation table from the value in bits[31:14] of the written value.

Bits[13:0] should be zero when written, and are unpredictable when read.

The following instructions can be used to access the TTB:

MRC p15, 0, Rd, c2, c0, 0; read TTB register
MCR p15, 0, Rd, c2, c0, 0; write TTB register

REGISTER 3: DOMAIN ACCESS CONTROL REGISTER

Register 3 is the read/write domain access control register consisting of sixteen 2-bit fields. Each of these 2-bit fields defines the access permissions for the domains shown in Table 2-13.

Table 2-13. Register 3: Domain Access Control

Register Bits	Domain
31:30	D15
29:28	D14
27:26	D13
25:24	D12
23:22	D11
21:20	D10
19:18	D9
17:16	D8
15:14	D7
13:12	D6
11:10	D5
9:8	D4
7:6	D3
5:4	D2
3:2	D1
1:0	D0

The encoding of the two bit domain access permission field is given in Table 3-5 on page 3-19. The following instructions can be used to access the domain access control register:

MRC p15, 0, Rd, c3, c0, 0; read domain 15:0 access permissions
MCR p15, 0, Rd, c3, c0, 0; write domain 15:0 access permissions

REGISTER 4: RESERVED

Accessing (reading or writing) this register will cause unpredictable behavior.

REGISTER 5: FAULT STATUS REGISTERS

Register 5 is the fault status register (FSR). The FSR contains the source of the last data fault, indicating the domain and type of access being attempted when the data abort occurred.

Table 2-14. Fault Status Register

Register Bits	Description
31:9	UNP when read/SBZ for write.
8	0 when read/SBZ for write.
7:4	Domain being accessed when fault occurred (D15-D0).
3:0	Fault type.

The fault type encoding is shown in Fault address and fault status registers on page 3-18.

The data FSR is defined in ARM architecture v4T. Additionally, a pipelined prefetch FSR is available, for debug purposes only. The pipeline matches that of the ARM9TDMI.

The following instructions can be used to access the data and prefetch FSR:

MRC p15, 0, Rd, c5, c0, 0 ;read data FSR value

MCR p15, 0, Rd, c5, c0, 0 ;write data FSR value

MRC p15, 0, Rd, c5, c0, 1 ;read prefetch FSR value

MCR p15, 0, Rd, c5, c0, 1 ;write prefetch FSR value

The ability to write to the FSR is useful for a debugger to restore the value of the FSR. The register should be written using the read-modify-write method. Bits[31:8] should be zero.

REGISTER 6: FAULT ADDRESS REGISTER

Register 6 is the fault address register (FAR) which contains the modified virtual address of the access being attempted when the last fault occurred. The FAR is only updated for data faults, not for prefetch faults. (The address for a prefetch fault can be found in R14.)

The following instructions can be used to access the FAR:

MRC p15, 0, Rd, c6, c0, 0 ;read FAR data
MCR p15, 0, Rd, c6, c0, 0 ;write FAR data

The ability to write to the FAR is intended for a debugger to restore a previous state.

REGISTER 7: CACHE OPERATIONS

Register 7 is a write-only register used to manage the instruction and data caches, ICache and DCache.

The cache operations provided by register 7 are described in Table 2-15.

Table 2-15. Function Descriptions Register 7

Function	Description
Invalidate cache	Invalidates all cache data, including any dirty data (note). Use with caution.
Invalidate single entry using modified virtual address	Invalidates a single cache line, discarding any dirty data (note). Use with caution.
Clean D single entry using either index or modified virtual address	Writes the specified cache line to main memory if the line is marked valid and dirty and marks the line as not dirty (note). The valid bit is unchanged.
Clean and Invalidate D entry using either index or modified virtual address	Writes the specified cache line to main memory if the line is marked valid and dirty (note). The line is marked not valid.
Prefetch cache line	Performs an ICache lookup of the specified modified virtual address. If the cache misses, and the region is cacheable, a linefill will be performed.

NOTE: Dirty data is data that has been modified in the cache but not yet written to main memory.

The function of each cache operation is selected by the opcode_2 and CRm fields in the MCR instruction used to write CP15 register 7. Writing other opcode_2 or CRm values is unpredictable.

Reading from CP15 register 7 is unpredictable.

Table 2-16 on page 2-16 shows instructions that can be used to perform cache operations with register 7

Table 2-16. Cache Operations Register 7

Function	Data	Instruction
Invalidate ICache & DCache	SBZ	MCR p15,0,Rd,c7,c7,0
Invalidate ICache	SBZ	MCR p15,0,Rd,c7,c5,0
Invalidate ICache single entry (using MVA)	MVA format	MCR p15,0,Rd,c7,c5,1
Prefetch ICache line (using MVA)	MVA format	MCR p15,0,Rd,c7,c13,1
Invalidate DCache	SBZ	MCR p15,0,Rd,c7,c6,0
Invalidate DCache single entry (using MVA)	MVA format	MCR p15,0,Rd,c7,c6,1
Clean DCache single entry (using MVA)	MVA format	MCR p15,0,Rd,c7,c10,1
Clean and Invalidate DCache entry (using MVA)	MVA format	MCR p15,0,Rd,c7,c14,1
Clean DCache single entry (using index)	Index format	MCR p15,0,Rd,c7,c10,2
Clean and Invalidate DCache entry (using index)	Index format	MCR p15,0,Rd,c7,c14,2
Drain write buffer (1)	SBZ	MCR p15,0,Rd,c7,c10,4
Wait for interrupt (2)	SBZ	MCR p15,0,Rd,c7,c0,4

NOTES:
1. Will stop execution until the write buffer has drained.
2. Will stop execution in a LOW power state until an interrupt occurs.

The operations which can be carried out upon a single cache line identify the line using the data passed in the MCR instruction. The data is interpreted using one of the following formats:

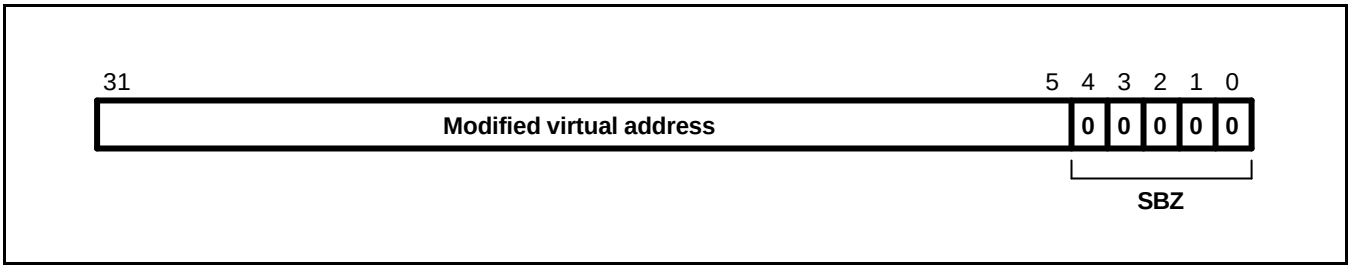


Figure 2-2. Register 7 MVA Format

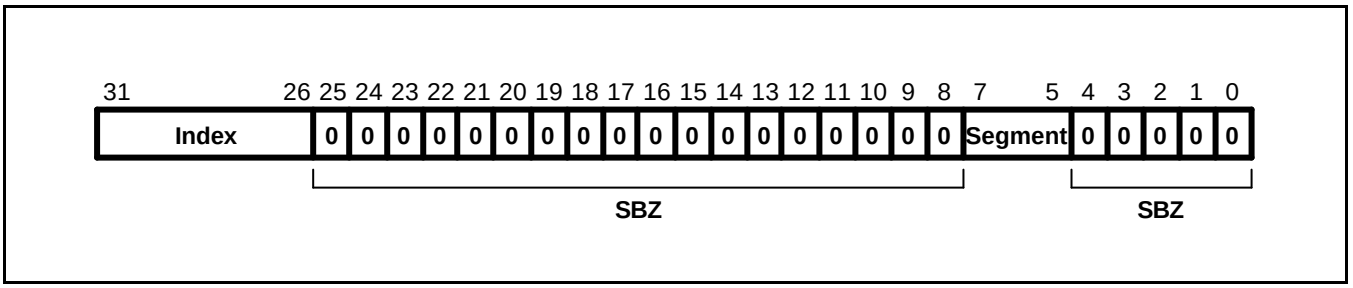


Figure 2-3. Register 7 Index Format

The use of register 7 is discussed in Chapter 4 Caches, Write Buffer and Physical Address TAG (PATAG) RAM.

REGISTER 8: TLB OPERATIONS

Register 8 is a write-only register used to manage the *translation lookaside buffers* (TLBs), the instruction TLB and the data TLB.

Five TLB operations are defined and the function to be performed is selected by the opcode_2 and CRm fields in the MCR instruction used to write CP15 register 8. Writing other opcode_2 or CRm values is unpredictable. Reading from CP15 register 8 is unpredictable.

Table 2-17 on page 2-18 shows instructions that can be used to perform TLB operations using register 8.

Table 2-17. TLB Operations Register 8

Function	Data	Instruction
Invalidate TLB(s)	SBZ	MCR p15,0,Rd,c8,c7,0
Invalidate I TLB	SBZ	MCR p15,0,Rd,c8,c5,0
Invalidate I TLB single entry (using MVA)	MVA format	MCR p15,0,Rd,c8,c5,1
Invalidate D TLB	SBZ	MCR p15,0,Rd,c8,c6,0
Invalidate D TLB single entry (using MVA)	MVA format	MCR p15,0,Rd,c8,c6,1

NOTE: These functions invalidate all the un-preserved entries in the TLB.
Invalidate TLB single entry functions invalidate any TLB entry corresponding to the modified virtual address given in Rd, regardless of its preserved state. See Register 10: TLB lock down register on page 2-21

Figure 2-4 shows the modified virtual address format used for operations on single entry TLB lines using register 8.

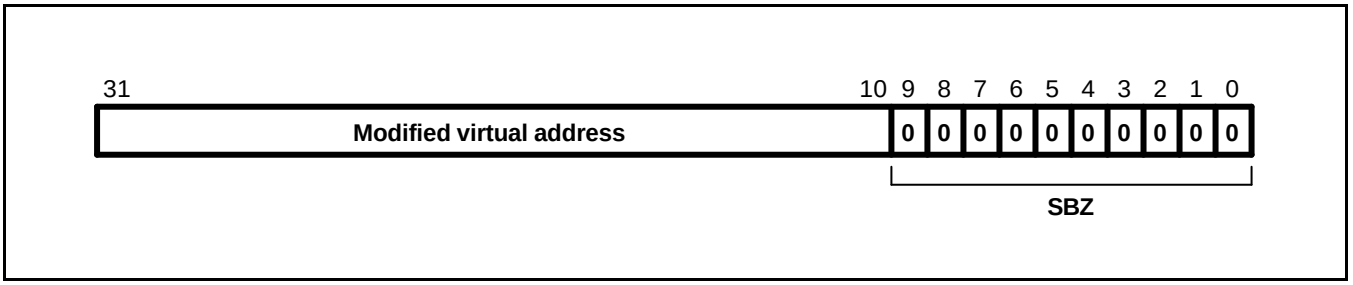


Figure 2-4. Register 8 MVA Format

REGISTER 9: CACHE LOCK DOWN REGISTER

Register 9 is the cache lock down register. The cache lock down register is 0x0 on reset. The cache lock down register allows software to control which cache line in the ICache or DCache respectively is loaded for a linefill and to prevent lines in the ICache or DCache from being evicted during a linefill, locking them into the cache.

There is a register for each of the ICache and DCache, the value of opcode_2 determines which cache register to access:

opcode_2 = 0x0 causes the DCache register to be accessed
opcode_2 = 0x1 causes the ICache register to be accessed.

The Opcode_1 and CRm fields should be zero.

Reading CP15 register 9 returns the value of the cache lock down register, which is the base pointer for all cache segments.

NOTE

Only bits [31:26] are returned. Bits [25:0] are unpredictable.

Writing CP15 register 9 updates the cache lock down register, both the base and the current victim pointer for all cache segments. Bits [25:0] should be zero.

The victim counter specifies the cache line to be used as the victim for the next linefill. This is incremented using either a random or round-robin replacement policy, determined by the state of the RR-bit in register 1. The victim counter generates values in the range (base to 63). This locks lines with index values in the range (0 to base-1). If base = 0, there are no locked lines.

Writing to CP15 register 9 updates the base pointer and the current victim pointer. The next linefill will use and then increment the victim pointer. The victim pointer will continue incrementing on linefills and will wrap around to the base pointer. For example, setting the base pointer to 0x3 prevents the victim pointer from selecting entries 0x0 to 0x2, locking them into the cache.

Load a cache line into ICache line 0 and lock it down:

MCR to CP15 register 9, opcode_2 = 0x1, Victim = Base = 0x0
MCR I prefetch. Assuming the ICache misses, a linefill will occur to line 0.
MCR to CP15 register 9, opcode_2 = 0x1, Victim = Base = 0x1

Further ICache linefills will now occur into lines 1 - 63.

Load a cache line into DCache line 0 and lock it down:

MCR to CP15 register 9, opcode_2 = 0x0, Victim = Base = 0x0
Data load (LDR/LDM). Assuming the DCache misses, a linefill will occur to line 0.
MCR to CP15 register 9, opcode_2 = 0x0, Victim = Base = 0x1
Further DCache linefills will now occur into lines 1 - 63.

NOTE

Writing CP15 register 9, with the CRm field set to 0b0001, updates the current victim pointer only for the specified segment only. Bits [31:26] specify the victim; bits [7:5] specify the segment (for a 16KB cache) and all other bits should be zero. This encoding is intended for debug use. It is not necessary and not advised, to use this encoding.

Figure 2-5 shows the format of bits in register 9

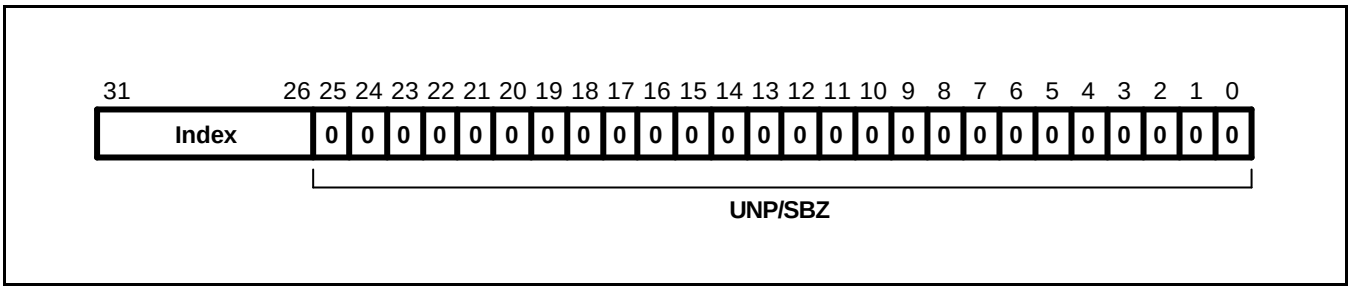


Figure 2-5. Register 9

Table 2-18 shows the instructions needed to access the cache lock down register:

Table 2-18. Accessing the Cache Lock Down Register 9

Function	Data	Instruction
Read DCache lock down base	Base	MRC p15,0,Rd,c9,c0,0
Write DCache victim and lockdown base	Victim=Base	MCR p15,0,Rd,c9,c0,0
Read ICache lock down base	Base	MRC p15,0,Rd,c9,c0,1
Write ICache victim and lockdown base	Victim = Base	MCR p15,0,Rd,c9,c0,1

REGISTER 10: TLB LOCK DOWN REGISTER

Register 10 is the TLB lock down register. The TLB lock down register is 0x0 on reset. There is a TLB lock down register for each of the TLBs, the value of opcode_2 determines which TLB register to access:

opcode_2 = 0x0 causes the D TLB register to be accessed

opcode_2 = 0x1 causes the I TLB register to be accessed.

Reading CP15 register 10 returns the value of the TLB lock down counter base register, the current victim number and the preserve bit (P bit). Note that bits [19:1] are unpredictable when read.

Writing CP15 register 10 updates the TLB lock down counter base register, the current victim pointer and the state of the preserve bit. Bits [19:1] should be zero when written.

Table 2-19 shows the instructions needed to access the TLB lock down register.

Table 2-19. Accessing the TLB Lock Down Register 10

Function	Data	Instruction
Read D TLB lock down	TLB lock-down	MRC p15,0,Rd,c10,c0,0
Write D TLB lock down	TLB lock-down	MCR p15,0,Rd,c10,c0,0
Read I TLB lock down	TLB lock-down	MRC p15,0,Rd,c10,c0,1
Write I TLB lock down	TLB lock-down	MCR p15,0,Rd,c10,c0,1

Figure 2-6 shows the format of bits in register 10.

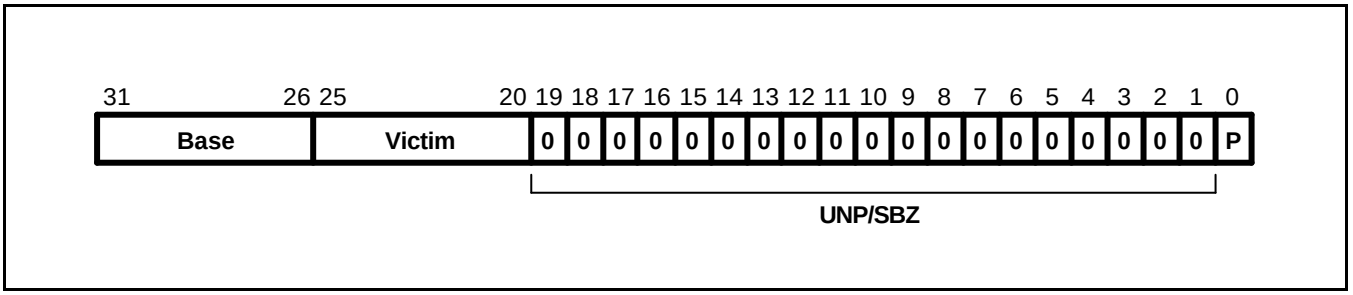


Figure 2-6. Register 10

The entries in the TLBs are replaced using a round robin replacement policy. This is implemented using a victim counter which counts from entry 0 up to 63 and then wraps back round to the base value and continues counting, wrapping around to the base value from 63 each time.

There are two mechanisms available for ensuring entries are not removed from the TLB:

Locking an entry down prevents it from being selected for overwriting during a table walk, this is achieved by programming the base value to which the victim counter reloads. For example, if the bottom 3 entries (0–2) are to be locked down, the base counter should be programmed to 3.

An entry can also be preserved during an Invalidate All instruction. This is done by ensuring the P bit is set when the entry is loaded into the TLB.

Load a single entry into I TLB location 0, make it immune to Invalidate All and lock it down:

MCR to CP15 register 10, opcode_2 = 0x1, Base Value = 0, Current Victim = 0, P = 1
MCR I prefetch. Assuming an I TLB miss occurs, then entry 0 will be loaded.
MCR to CP15 register 10, opcode_2 = 0x1, Base Value = 1, Current Victim = 1, P = 0

Load a single entry into D TLB location 0, make it immune to Invalidate All and lock it down:

MCR to CP15 register 10, opcode_2 = 0x0, Base Value = 0, Current Victim = 0, P = 1
Data load (LDR/LDM) or store (STR/STM). Assuming a D TLB miss occurs, then entry 0 will be loaded.
MCR to CP15 register 10, opcode_2 = 0x0, Base Value = 1, Current Victim = 1, P = 0

REGISTERS 11-12 & 14: RESERVED

Accessing (reading or writing) any of these registers will cause unpredictable behavior.

REGISTER 13: PROCESS ID

Register 13 is the process identifier register. The process identifier register is 0x0 on reset.

Reading from CP15 register 13 returns the value of the process identifier. Writing CP15 register 13 updates the process identifier to the value in bits [31:25]. Bits [24:0] should be zero.

Register 13 bit assignments are shown in Figure 2-7.

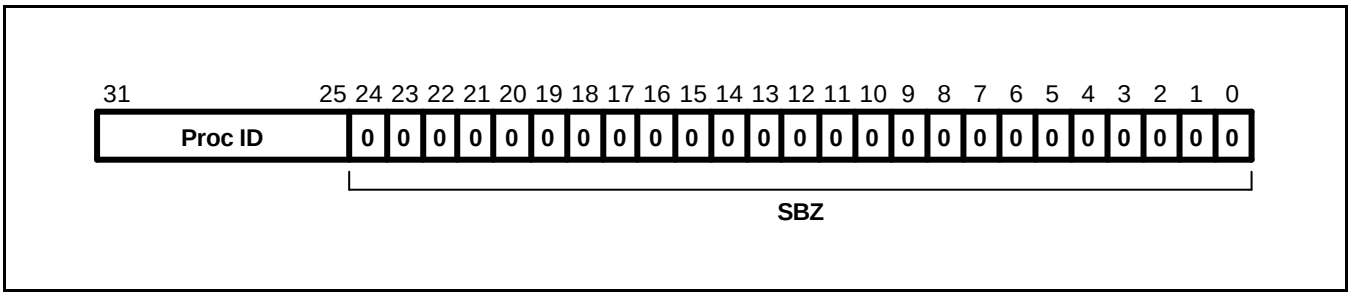


Figure 2-7. Register 13

Register 13 can be accessed using the following instructions:

MRC p15, 0, Rd, c13, c0, 0 ;read process identifier
MCR p15, 0, Rd, c13, c0, 0 ;write process identifier

Using the process Identifier (ProcID)

Addresses issued by the ARM9TDMI core in the range 0 to 32MB are translated by CP15 register 13, the ProcID register. Address A becomes A + (ProcID x 32MB). It is this translated address that is seen by both the Caches and MMU. Addresses above 32MB undergo no translation. This is shown in Figure 2-8 on page 2-23 .

The ProcID is a seven bit field, enabling 64 x 32MB processes to be mapped.

NOTE

If ProcID is zero, as it is on reset, then there is a flat mapping between the ARM9TDMI and the Caches and MMU.

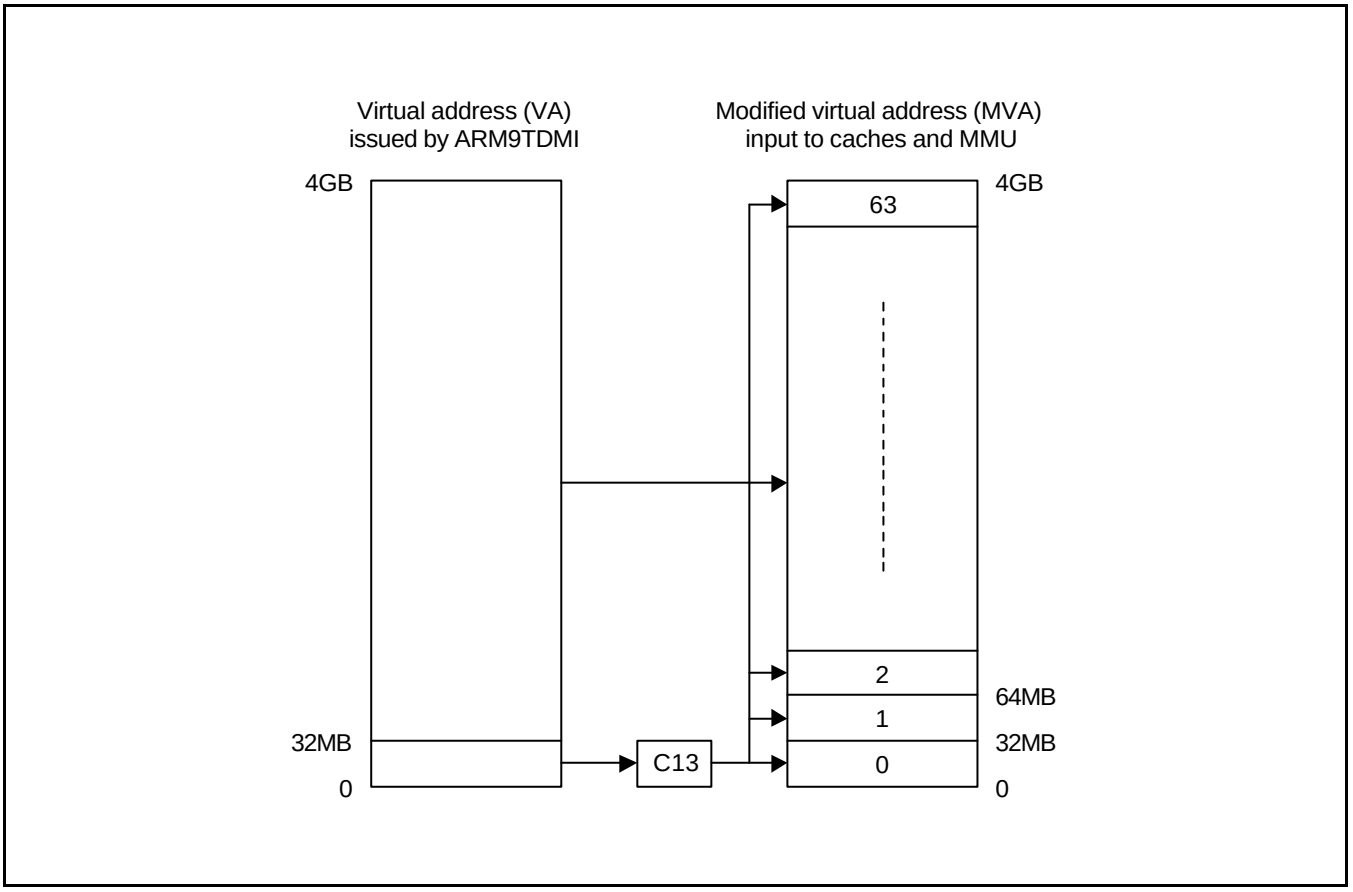


Figure 2-8. Address Mapping Using CP15 Register 13

Changing the ProcID - performing a fast context switch

A fast context switch is done by writing to CP15 register 13. The contents of the caches and TLBs do not have to be flushed after a fast context switch because they still hold valid address tags. It should be noted that the two instructions after the MCR to write the ProcID will have been fetched with the old ProcID:

```
{ProcID = 0}

MOV r0, #1:SHL:25          ; Fetched with ProcID = 0

MCR p15,0,r0,c13,c0,0      ; Fetched with ProcID = 0

A1                         ; Fetched with ProcID = 0

A2                         ; Fetched with ProcID = 0

A3                         ; Fetched with ProcID = 1
```

REGISTER 15: TEST CONFIGURATION REGISTER

Register 15 is used for test purposes. Accessing (reading or writing) this register will cause the ARM920T to have unpredictable behavior.

Appendix 3

MMU

ABOUT THE MMU

ARM920T implements an enhanced ARM Architecture V4 MMU to provide translation and access permission checks for the instruction and data address ports of the ARM9TDMI. The MMU is controlled from a single set of two-level page tables stored in main memory, and are enabled by M-Bit in CP15 register 1, providing a single address translation and protection scheme. The instruction and data TLBs in the MMU can be independently locked and flushed.

The MMU features are:

- standard ARM V4 MMU mapping sizes, domains, and access protection scheme
- mapping sizes are 1MB sections, 64KB large pages, 4KB small pages and new 1KB tiny pages
- access permissions for sections
- access permissions for large pages and small pages can be specified separately for each quarter of the page (these quarters are called sub-pages)
- 16 domains implemented in hardware
- 64 entry instruction TLB and 64 entry data TLB
- hardware page table walks
- round-robin replacement algorithm (also called cyclic)
- invalidate whole TLB via CP15 Register 8
- invalidate TLB entry, selected by modified virtual address, via CP15 register 8
- independent lockdown of instruction TLB and data TLB via CP15 register 10.

ACCESS PERMISSIONS AND DOMAINS

For large and small pages, access permissions are defined for each sub-page (1KB for small pages, 16KB for large pages). Sections and tiny pages have a single set of access permissions.

All regions of memory have an associated domain. A domain is the primary access control mechanism for a region of memory and defines the conditions in which an access can proceed. The domain determines whether:

the access permissions are used to qualify the access

the access is unconditionally allowed to proceed

the access is unconditionally aborted.

In the latter two cases, the access permission attributes are ignored.
There are 16 domains, which are configured using the domain access control register.

TRANSLATED ENTRIES

Each TLB caches 64 translated entries. During CPU memory accesses, the TLB provides the protection information to the access control logic.

If the TLB contains a translated entry for the modified virtual address, the access control logic determines whether access is permitted:

- If access is permitted and an off-chip access is required, the MMU outputs the appropriate physical address corresponding to the modified virtual address.
- If access is permitted and an off-chip access is not required, the cache services the access.
- If access is not permitted, the MMU signals the CPU core to abort.

If a TLB misses (it does not contain an entry for the virtual address) the translation table walk hardware is invoked to retrieve the translation information from a translation table in physical memory. Once retrieved, the translation information is written into the TLB, possibly overwriting an existing value.

The entry to be written is chosen by cycling sequentially through the TLB locations. To enable use of TLB locking features, the location to be written can be specified using CP15 register 10, TLB lockdown.

When the MMU is turned off (as happens on reset), no address mapping occurs and all regions are marked as non-cacheable and non-bufferable. See About the caches and write buffer on page 4-1.

MMU PROGRAM ACCESSIBLE REGISTERS

Table 3-1 shows system control coprocessor (CP15) registers which are used, in conjunction with page table descriptors stored in memory, to determine the operation of the MMU

Table 3-1. CP15 Register Functions

Register	Number	Bits	Register description
Control register	1	M, A, S, R	Contains bits to enable the MMU (M bit), enable data address alignment checks (A bit) and to control the access protection scheme (S bit and R bit).
Translation table base register	2	31:14	Holds the physical address of the base of the translation table maintained in main memory. This base address must be on a 16KB boundary and is common to both TLBs.
Domain access control register	3	31:0	Comprises sixteen 2-bit fields. Each field defines the access control attributes for one of 16 domains (D15–D0).
Fault status register	5 (I & D)	7:0	Indicates the cause of a data and prefetch abort and the domain number of the aborted access, when an abort occurs. Bits 7:4 specify which of the 16 domains (D15–D0) was being accessed when a fault occurred. Bits 3:0 indicate the type of access being attempted. The value of all other bits is unpredictable. The encoding of these bits is shown in Table 3-4 on page 3-18 .
Fault address register	6 (D)	31:0	Holds the virtual address associated with the access that caused the data abort. See Table 3-4 on page 3-18 for details of the address stored for each type of fault. ARM9TDMI Register 14 can be used to determine the virtual address associated with a prefetch abort.
TLB operations register	8	31:0	Writing to this register causes the MMU to perform TLB maintenance operations; either invalidating all the (unpreserved) entries in the TLB, or invalidating a specific entry.
TLB lock down register	10 (I & D)	31:20 & 0	Allows specific page table entries to be locked into the TLB and the TLB victim index to be read/written: opcode 2 = 0x0 accesses the D TLB lock down register opcode 2 = 0x1 accesses the I TLB lock down register. Locking entries in the TLB guarantees that accesses to the locked page or section can proceed without incurring the time penalty of a TLB miss. This allows the execution latency for time-critical pieces of code such as interrupt handlers to be minimized.

All the CP15 MMU registers, except register 8, contain state and can be read using MRC instructions and written using MCR instructions. Registers 5 and 6 are also written by the MMU during a data abort. Writing to Register 8 causes the MMU to perform a TLB operation, to manipulate TLB entries. This register cannot be read. The instruction TLB (I TLB) and data TLB (D TLB) both have a copy of register 10, the opcode_2 field in the CP15 instruction is used to determine which one is accessed.

The system control coprocessor (CP15) is described in Programmer's Model on page 2-1. Details of register format and the coprocessor instructions to access them are given there.

ADDRESS TRANSLATION

The MMU translates virtual addresses generated by the CPU core (and by CP15 register 13) into physical addresses to access external memory. It also derives and checks the access permission, using a translation lookaside buffer (TLB).

The MMU table walking hardware is used to add entries to the TLB. The translation information, which comprises both the address translation data and the access permission data, resides in a translation table located in physical memory. The MMU provides the logic needed to traverse this translation table and load entries into the TLB.

There are up to two stages to the hardware table walking (and hence permission checking) process. The number of stages depends on whether the address in question has been marked as a section-mapped access or a page-mapped access. There is one size of section and three sizes of page-mapped access (large pages, small pages and tiny pages). The translation process always starts out in the same way, with a level one fetch. A section-mapped access requires only a level one fetch, but a page-mapped access requires a subsequent level two fetch.

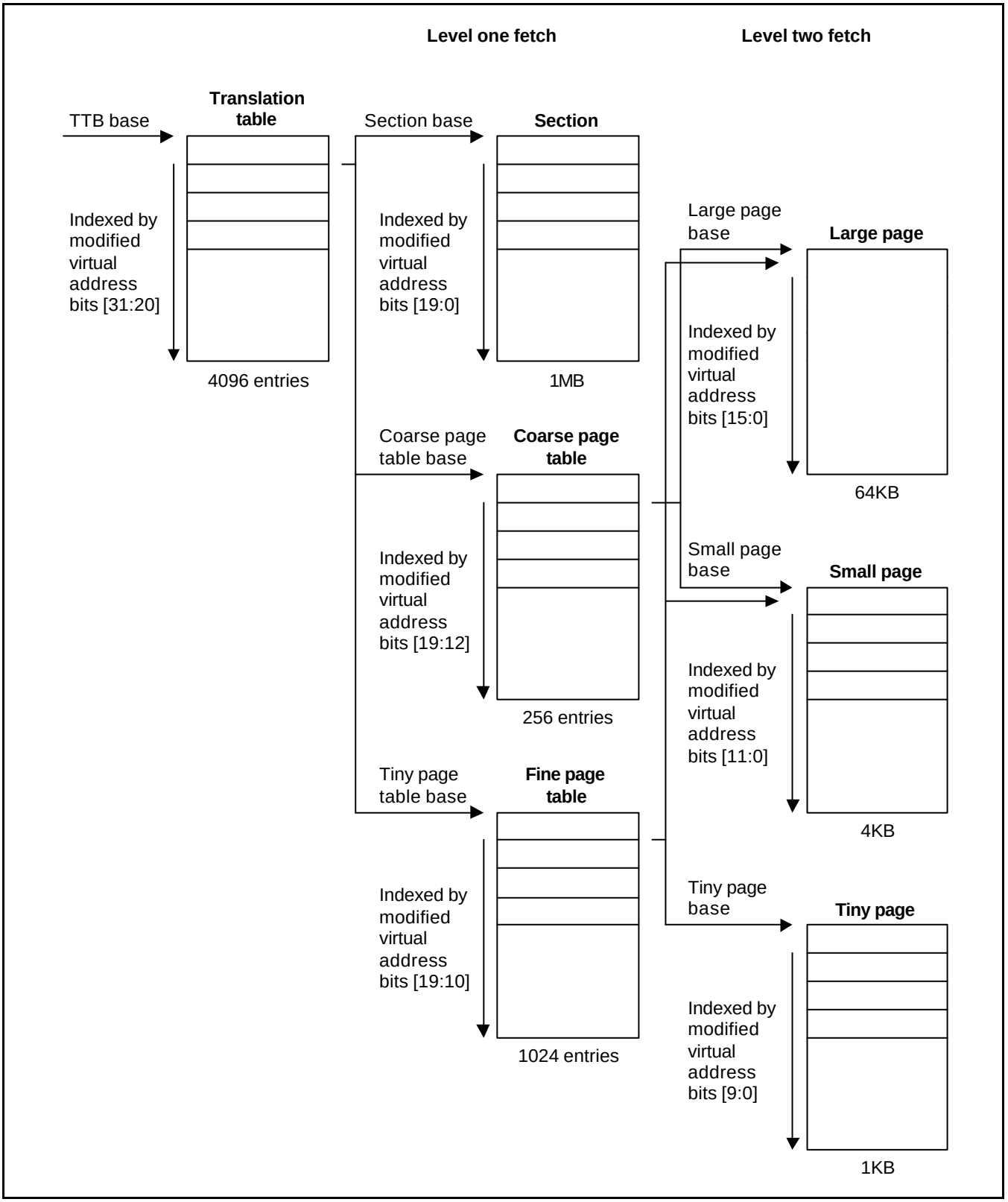


Figure 3-1. Translating Page Tables

HARDWARE TRANSLATION PROCESS

TRANSLATION TABLE BASE

The hardware translation process is initiated when the TLB does not contain a translation for the requested modified virtual address. The translation table base (TTB) register points to the base address of a table in physical memory which contains section and/or Page descriptors. The 14 low-order bits of the TTB register are set to zero on a read and the table must reside on a 16KB boundary.

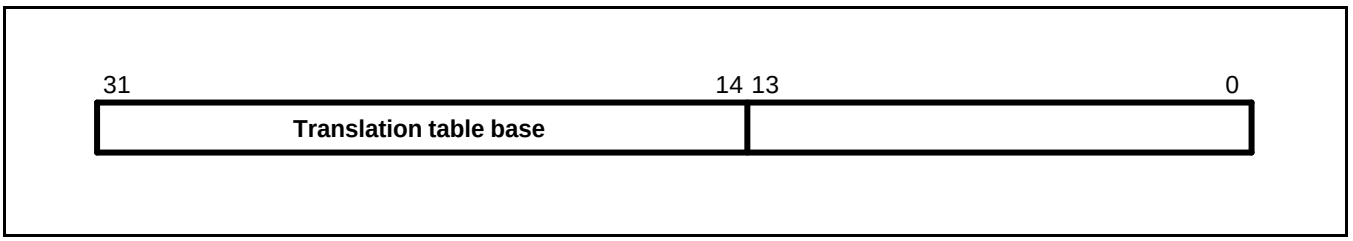


Figure 3-2. Translation Table Base Register

The translation table has up to 4096 x 32-bit entries, each describing 1MB of virtual memory. This allows up to 4GB of virtual memory to be addressed. Figure 3-1 on page 3-5 illustrates the table walk process.

LEVEL ONE FETCH

Bits 31:14 of the translation table base register are concatenated with bits 31:20 of the modified virtual address to produce a 30-bit address as illustrated in Figure 3-3 on page 3-7.

This address selects a 4-byte translation table entry which is a level one descriptor for either a section or a page table.

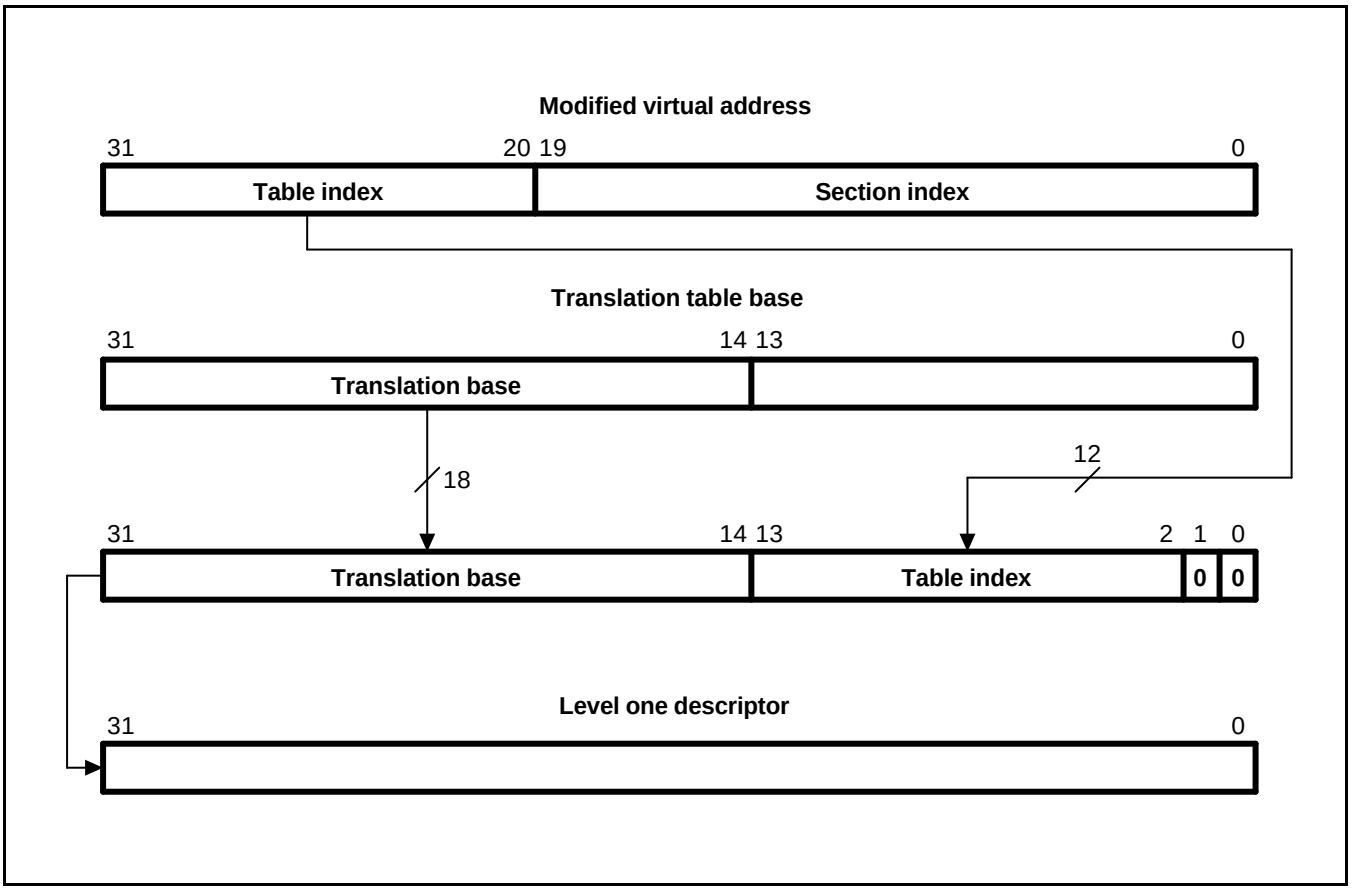


Figure 3-3. Accessing the Translation Table Level One Descriptors

LEVEL ONE DESCRIPTOR

The level one descriptor returned is either a section descriptor a coarse page table descriptor, or a fine page table descriptor. A section descriptor provides the base address of a 1MB block of memory. The page table descriptors provide the base address of a page table that contains level two descriptors.

There are two sizes of page table:

- coarse page tables have 256 entries, splitting the 1MB the table describes into 4KB blocks
- fine page tables have 1024 entries, splitting the 1MB the table describes into 1KB blocks

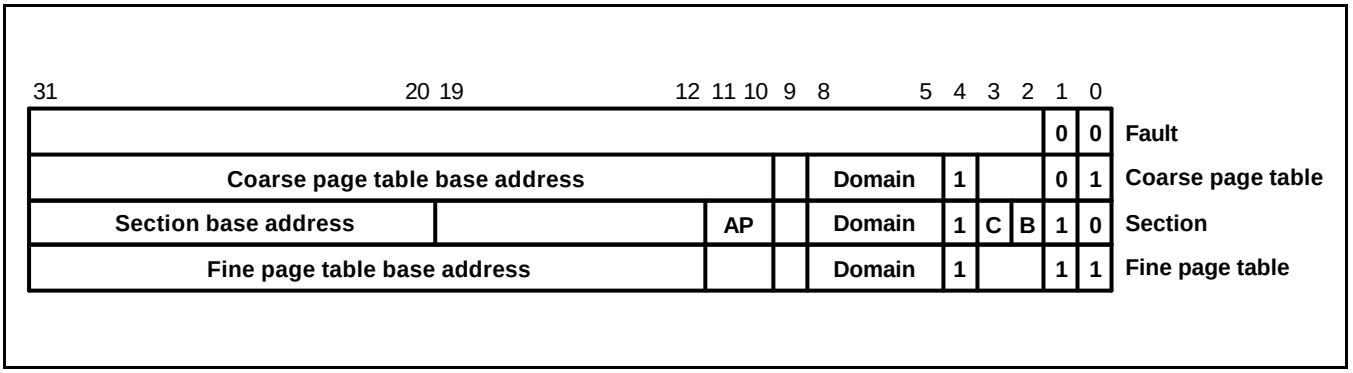


Figure 3-4. Level One Descriptors

The two least significant bits indicate the descriptor type

Table 3-2. Interpreting Level One Descriptor Bits [1:0]

Value	Meaning	Notes
00	Invalid	Generates a section translation fault.
01	Coarse page table	Indicates that this is a coarse page table descriptor.
10	Section	Indicates that this is a section descriptor.
11	Fine page table	Indicates that this is a fine page table descriptor.

SECTION DESCRIPTOR

Bits 3:2 (C & B) indicate whether the area of memory mapped by this section is treated as write-back cacheable, write-through cacheable, non-cached buffered or non-cached non-buffered.

Bit 4 should be written to 1 for backward compatibility.

Bits 8:5 specify one of the 16 possible domains (held in the domain access control registers) that contain the primary access controls.

Bit 9 is always written as 0.

Bits 11:10 (AP) specify the access permissions for this section.

Bits 19:12 are always written as 0.

Bits 31:20 form the corresponding bits of the physical address for a section.

COARSE PAGE TABLE DESCRIPTOR

Bits 3:2 are always written as 0.

Bit 4 is always written as 1.

Bits 8:5 specify one of the 16 possible domains (held in the Domain access control registers) that contain the primary access controls.

Bit 9 is always written as 0.

Bits 31:10 form the base for referencing the level two descriptor. (The coarse page table index for the entry is derived from the modified virtual address.)

If a coarse page table descriptor is returned from the level one fetch, a level two fetch is initiated.

FINE PAGE TABLE DESCRIPTOR

Bits 3:2 are always written as 0.

Bit 4 is always written as 1.

Bits 8:5 specify one of the 16 possible domains (held in the domain access control registers) that contain the primary access controls.

Bits 11:9 are always written as 0.

Bits 31:12 form the base for referencing the level two descriptor. (The fine page table index for the entry is derived from the modified virtual address.)

If a fine page table descriptor is returned from the level one fetch, a level two fetch is initiated.

TRANSLATING SECTION REFERENCES

Figure 3-5 illustrates the complete section translation sequence. Note that access permissions contained in the level one descriptor must be checked before the physical address is generated.

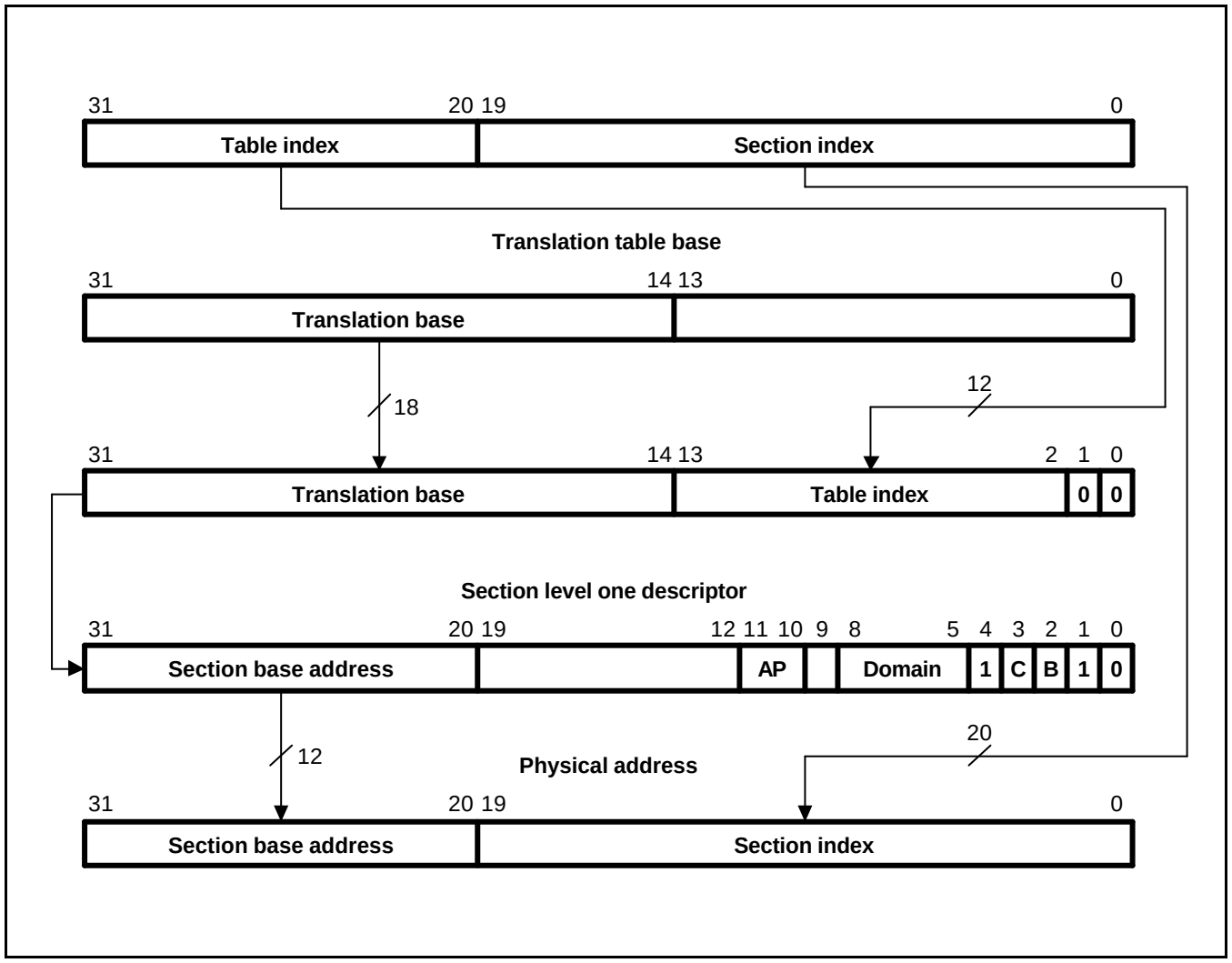


Figure 3-5. Section Translation

LEVEL TWO DESCRIPTOR

If the level one fetch returns either a coarse page table descriptor or a fine page table descriptor, this provides the base address of the page table to be used. The page table is then accessed and a level two descriptor is returned. This defines either a tiny, a small or a large page descriptor:

- a tiny page descriptor provides the base address of a 1KB block of memory
- a small page descriptor provides to the base address of a 4KB block of memory
- a large page descriptor provides the base address of a 64KB block of memory

Coarse page tables have 256 entries, each entry describing 4KB. These entries can provide base addresses for either small or large pages. Large page descriptors must be repeated in 16 consecutive entries.

Fine page tables have 1024 entries, each entry describing 1KB. These entries can provide base addresses for either tiny, small or large pages. Small page descriptors must be repeated in 4 consecutive entries and large page descriptors must be repeated in 64 consecutive entries.

The figure below shows the format of level one descriptors.

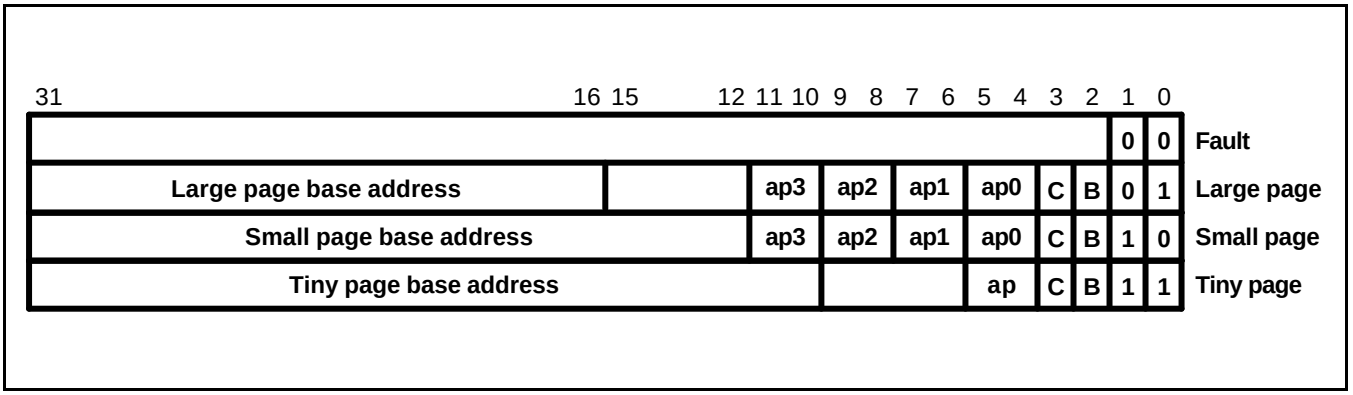


Figure 3-6. Page Table Entry (Level One Descriptor)

Bits[1:0] indicate the page size and validity and are interpreted as follows.

Table 3-3. Interpreting Page Table Entry Bits 1:0

Value	Meaning	Notes
00	Invalid	Generates a page translation fault.
01	Large page	Indicates that this is a 64KB page.
10	Small page	Indicates that this is a 4KB page.
11	Tiny page	Indicates that this is a 1KB page.

Bit 3:2 (C & B) indicate whether the area of memory mapped by this page is treated as write-back cacheable, write-through cacheable, non cached buffered or non-cached non-buffered.

Domain access control on page 3-19 and Fault checking sequence on page 3-21 show how to interpret the access permission (ap) bits.

NOTE

Tiny pages do not support sub page permissions and therefore only have one set of access permission bits.

Bits 31:10 (tiny pages), 31:12 (small pages) or bits 31:16 (large pages) are used to form the corresponding bits of the physical address.

TRANSLATING LARGE PAGE REFERENCES

Figure 3-7 on page 3-13 illustrates the complete translation sequence for a 64KB large page.

As the upper four bits of the page index and low-order four bits of the coarse page table index overlap, each coarse page table entry for a large page must be duplicated 16 times (in consecutive memory locations) in the coarse page table.

If a large page descriptor is included in a fine page table the upper six bits of the page index and low-order six bits of the fine page table index overlap, each fine page table entry for a large page must therefore be duplicated 64 times.

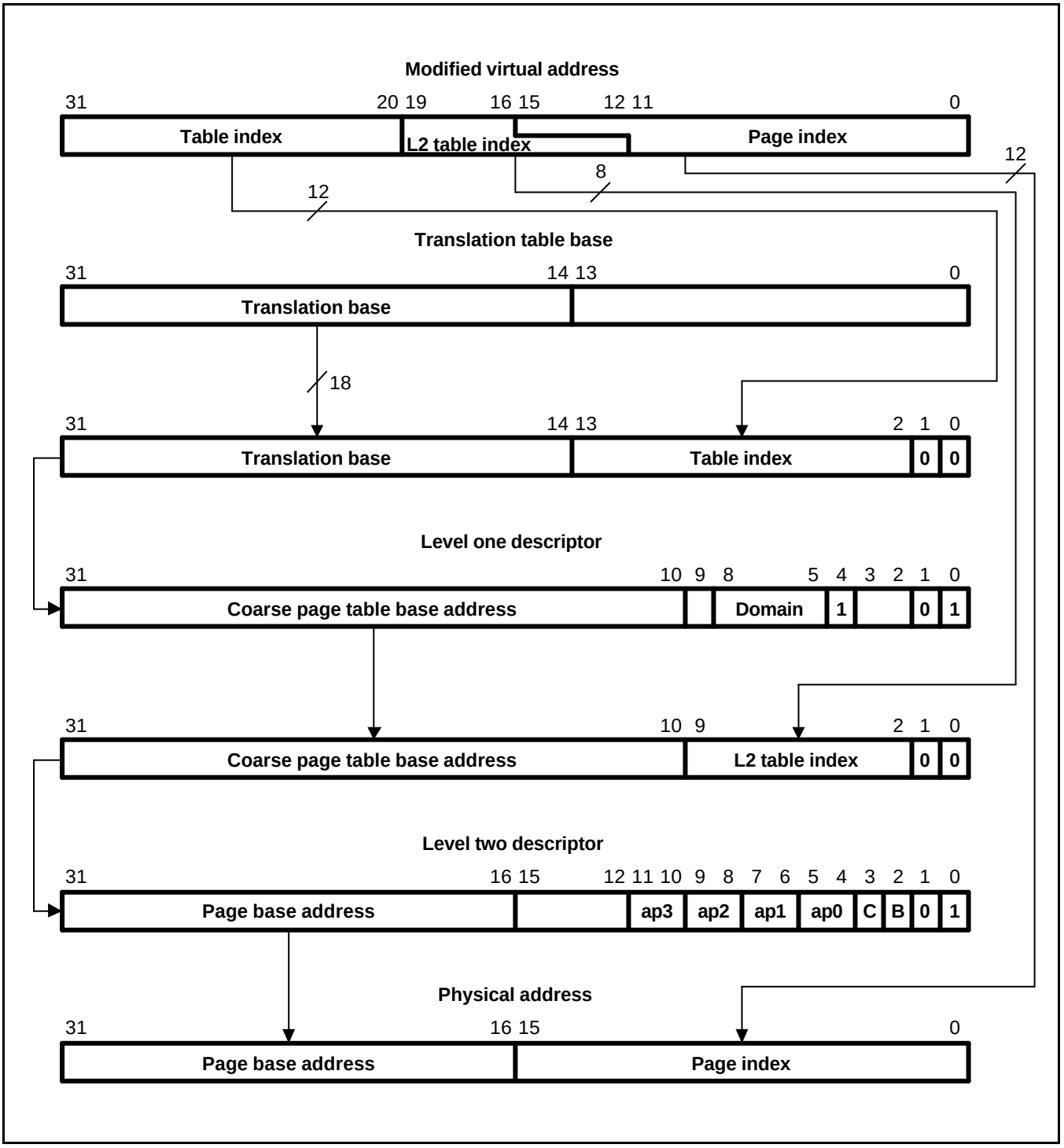


Figure 3-7. Large Page Translation from a Coarse Page Table

TRANSLATING SMALL PAGE REFERENCES

Figure 3-8 illustrates the complete translation sequence for a 4KB small page. If a small page descriptor is included in a fine page table, the upper two bits of the page index and low-order two bits of the fine page table index overlap. Each fine page table entry for a small page must therefore be duplicated four times.

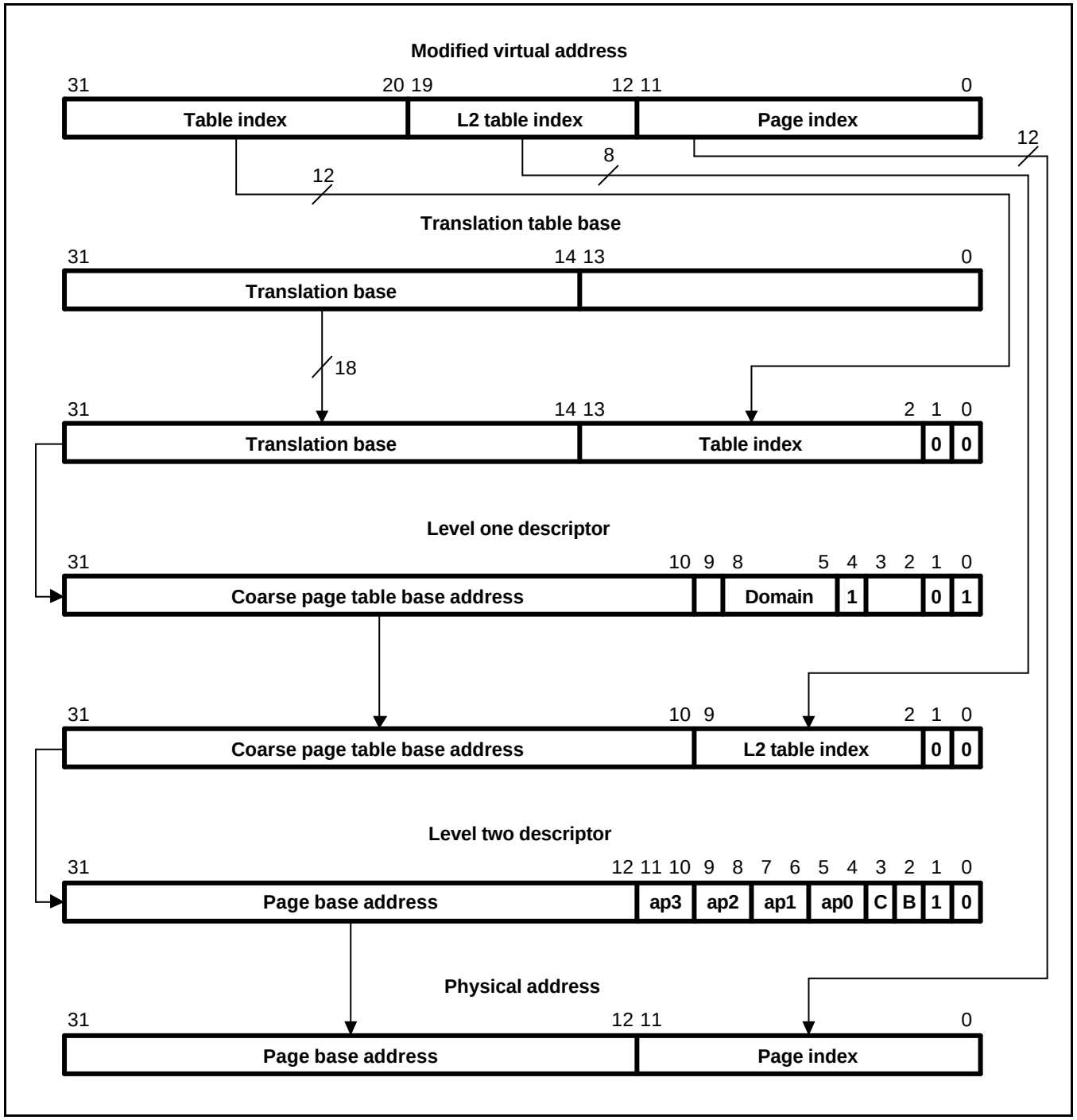


Figure 3-8. Small Page Translation from a Coarse Page Table

TRANSLATING TINY PAGE REFERENCES

Figure 3-9 on page 3-16 illustrates the complete translation sequence for a 1KB tiny page. Page translation involves one additional step beyond that of a section translation: the level one descriptor is the fine page table descriptor and this is used to point to the level one descriptor.

NOTE

The domain specified in the level one description and access permissions specified in the level one description together determine whether the access has permissions to proceed. See section Domain access control on page 3-19) for details.

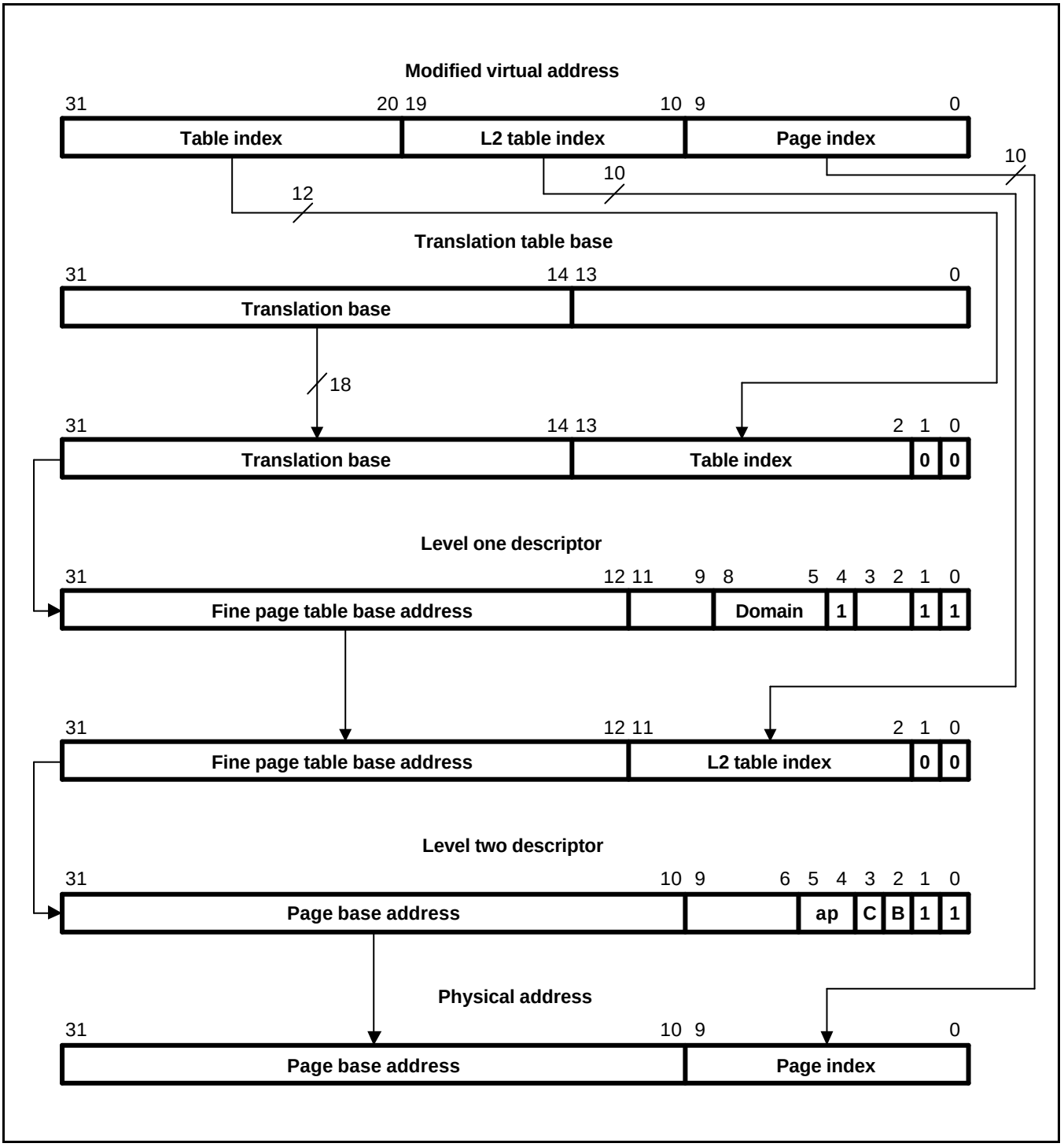


Figure 3-9. Tiny Page Translation from a Fine Page Table

SUB-PAGES

Access permissions can be defined for sub pages of small and large pages. If, during a page walk, a small or large page has a non-identical sub page permission, only the sub page being accessed is written into the TLB. For example, a 16KB (large page) sub page entry will be written into the TLB if the sub page permission differs and a 64KB entry will be put in the TLB if the sub page permissions are identical.

When sub page permissions are used and the page entry then needs invalidating, all four sub pages must be invalidated separately.

MMU FAULTS AND CPU ABORTS

The MMU generates an abort on the following types of faults:

- alignment faults (data accesses only)
- translation faults
- domain faults
- permission faults.

In addition, an external abort may be raised by the external system as a result of certain types of external data access.

Alignment fault checking is enabled by the A-bit in CP15 register 1. Alignment fault checking is not affected by whether or not the MMU is enabled. Translation, domain and permission faults are only generated when the MMU is enabled.

The access control mechanisms of the MMU detect the conditions that produce these faults. If a fault is detected as the result of a memory access, the MMU will abort the access and signal the fault condition to the CPU core. The MMU retains status and address information about faults generated by the data accesses in the fault status register and fault address register (see section Fault address and fault status registers on page 3-18). The MMU does not retain status about faults generated by instruction fetches.

An access violation for a given memory access inhibits any corresponding external access, with an abort returned to the CPU core.

FAULT ADDRESS AND FAULT STATUS REGISTERS

On a data abort, the MMU places an encoded 4 bit value, FS[3:0], along with the 4-bit encoded domain number, in the Data fault status register (FSR). Similarly, on a prefetch abort, in the Prefetch fault status register, intended for debug purposes only. In addition, the modified virtual address associated with the data abort is latched into the *fault address register* (FAR). If an access violation simultaneously generates more than one source of abort, they are encoded in the priority given in Table 3-4 . The fault address register is not updated by faults caused by instruction prefetches.

FAULT STATUS

The remainder of this chapter describes the various access permissions and controls supported by the data MMU and details how these are interpreted to generate faults.

Table 3-4. Priority Encoding of Fault Status

	Source		Status	Domain	FAR
Highest priority	Alignment		0b00x1	invalid	MVA of access causing abort
	Translation	Section Page	0b0101 0b0111	invalid valid	MVA of access causing abort
	Domain	Section Page	0b1001 0b1011	valid valid	MVA of access causing abort
	Permission	Section Page	0b1101 0b1111	valid valid	MVA of access causing abort
Lowest priority	External abort on NCNB access or NCB read.	Section Page	0b1000 0b1010	valid valid	MVA of access causing abort

NOTES:

1. Data FSR only.
   Alignment faults may write either 0b0001 or 0b0011 into FS[3:0].
   Invalid values in domain[3:0] occur because the fault is raised before a valid domain field has been read from a page table descriptor.
   Any abort masked by the priority encoding may be regenerated by fixing the primary abort and restarting the instruction.
   NCNB means Non-Cacheable and Non-Bufferable.
   NCB means Non-Cacheable but Bufferable.
2. Instruction FSR only.
   The same priority applies as for the Data fault status register, except that alignment faults cannot occur, and external aborts apply only to NC (Non-cacheable) reads.

DOMAIN ACCESS CONTROL

MMU accesses are primarily controlled via domains. There are 16 domains and each has a 2-bit field to define access to it. Two types of user are supported, clients and managers. See Table 3-5. The domains are defined in the domain access control register. Figure 3-10 illustrates how the 32 bits of the register are allocated to define the 16 2-bit domains.

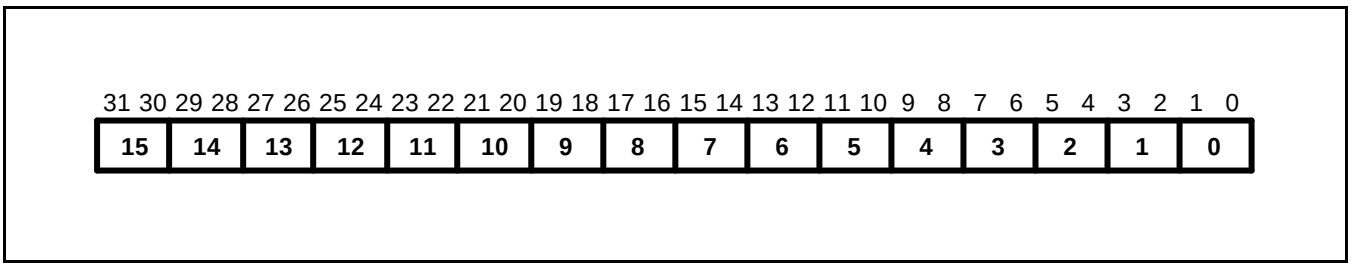


Figure 3-10. Domain Access Control Register Format

Table 3-5 defines how the bits within each domain are interpreted to specify the access permissions

Table 3-5. Interpreting Access Control Bits in Domain Access Control Register

Value	Meaning	Notes
00	No Access	Any access will generate a domain fault.
01	Client	Accesses are checked against the access permission bits in the section or page descriptor.
10	Reserved	Reserved. Currently behaves like the no access mode.
11	Manager	Accesses are *not* checked against the access permission bits so a permission fault cannot be generated.

Table 3-6 shows how to interpret the access permission (AP) bits and how their interpretation is dependent upon the S and R bits (control register bits 8 and 9).

Table 3-6. Interpreting Access Permission (AP) Bits

AP	S	R	Supervisor Permissions	User Permissions	Notes
00	0	0	No access	No access	Any access generates a permission fault
00	1	0	Read only	No access	Supervisor read only permitted
00	0	1	Read only	Read only	Any write generates a permission fault
00	1	1	Reserved		
01	x	x	Read/write	No access	Access allowed only in supervisor mode
10	x	x	Read/write	Read only	Writes in user mode cause permission fault
11	x	x	Read/write	Read/write	All access types permitted in both modes.
xx	1	1	Reserved		

FAULT CHECKING SEQUENCE

The sequence by which the MMU checks for access faults is different for sections and pages. The sequence for both types of access is shown below. The conditions that generate each of the faults are described on the following pages.

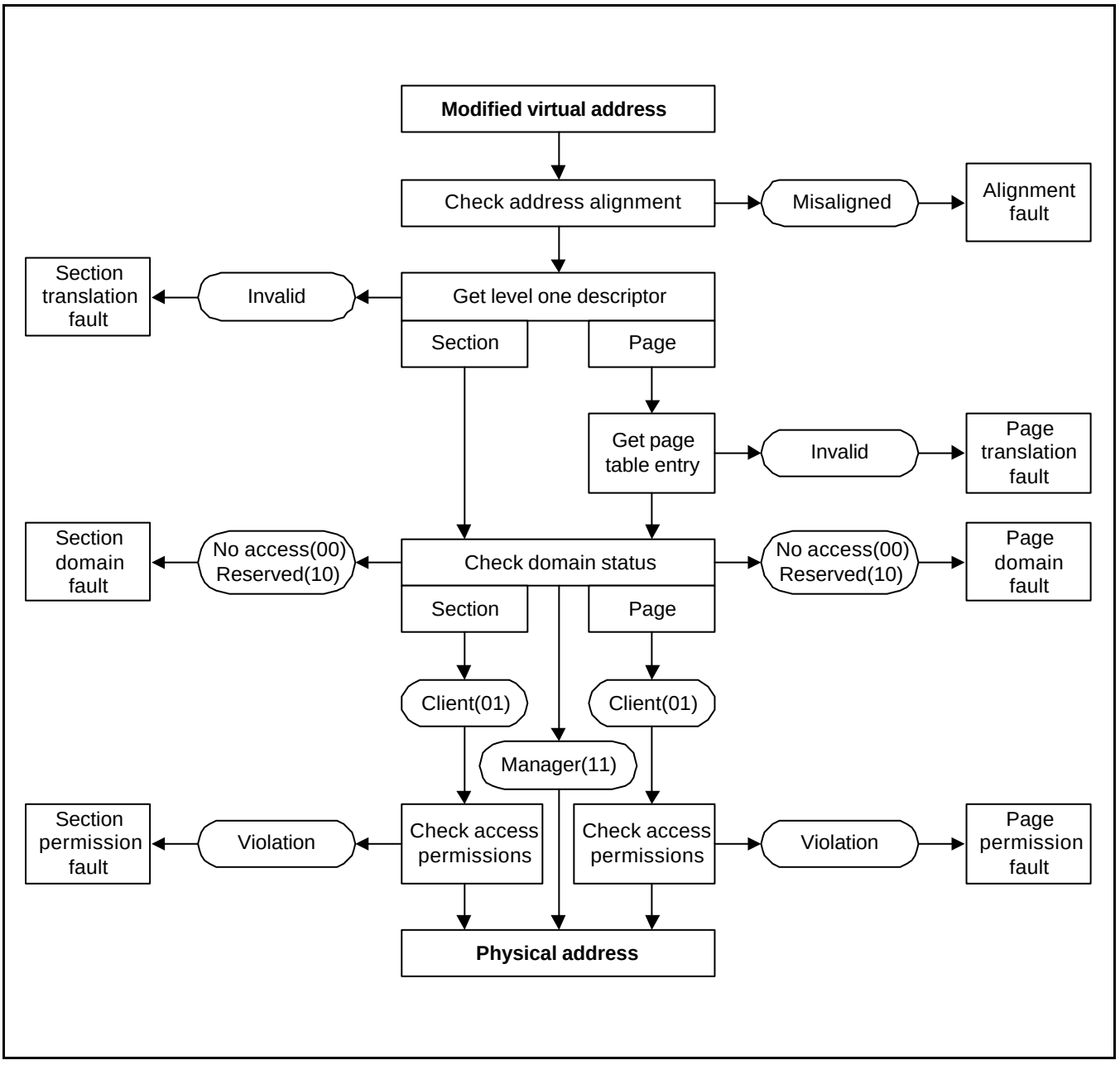


Figure 3-11. Sequence for Checking Faults

ALIGNMENT FAULT

If alignment fault is enabled (A-Bit in CP15 register 1 set), the MMU will generate an alignment fault on any data word access the address of which is not word aligned, or on any halfword access the address of which is not halfword aligned, irrespective of whether the MMU is enabled or not. An alignment fault will not be generated on any instruction fetch, nor on any byte access.

NOTE

If the access generates an alignment fault, the access sequence will abort without reference to further permission checks.

TRANSLATION FAULT

There are two types of translation fault, section and page:

Section	A section translation fault is generated if the level one descriptor is marked as invalid. This happens if bits[1:0] of the descriptor are both 0.
Page	A page translation fault is generated if the level one descriptor is marked as invalid. This happens if bits[1:0] of the descriptor are both 0.

DOMAIN FAULT

There are two types of domain fault, section and page. In both cases the level one descriptor holds the 4-bit domain field which selects one of the 16 2-bit domains in the domain access control register. The two bits of the specified domain are then checked for access permissions as detailed in Table 3-6 on page 3-20. In the case of a section, the domain is checked once the level one descriptor is returned and in the case of a page, the domain is checked once the level one descriptor is returned.

If the specified access is either no access (00) or reserved (10) then either a section domain fault or page domain fault occurs.

PERMISSION FAULT

If the 2-bit domain field returns 01 (client) then access permissions are checked as follows:

Section	If the level one descriptor defines a section-mapped access, the AP bits of the descriptor define whether or not the access is allowed according to Table 3-6 on page 3-20. Their interpretation is dependent upon the setting of the S and R bits (control register bits 8 and 9). If the access is not allowed, a section permission fault is generated.
Large page, small page	If the level one descriptor defines a page-mapped access and the level two descriptor is for a large or small page, four access permission fields (ap3-ap0) are specified, each corresponding to one quarter of the page. Hence, for small pages ap3 is selected by the top 1KB of the page and ap0 is selected by the bottom 1KB of the page. For large pages, ap3 is selected by the top 16KB of the page and ap0 is selected by the bottom 16KB of the page. The selected AP bits are then interpreted in exactly the same way as for a section (see Table 3-6 on page 3-20), the only difference being the fault generated is a page permission fault.
Tiny page	If the level one descriptor defines a page-mapped access and the level two descriptor is for a tiny page, the AP bits of the level one descriptor define whether or not the access is allowed in the same way as for a section. The fault generated is a page permission fault.

EXTERNAL ABORTS

In addition to the MMU-generated aborts the ARM920T can be externally aborted by the AMBA bus, which may be used to flag an error on an external memory access. However, not all accesses can be aborted in this way and the Bus Interface Unit (BIU) ignores external aborts that can not be handled.

The following accesses may be aborted:

non-cached reads

unbuffered writes

read-lock-write sequence, to non-cacheable memory.

In the case of a read-lock-write (SWP) sequence in which the read aborts, the write will always be attempted.

INTERACTION OF THE MMU AND CACHES

The MMU is enabled and disabled using bit 0 of the CP15 control register.

ENABLING THE MMU

To enable the MMU:

1) Program the translation table base and domain access control registers.
2) Program level 1 and level 2 page tables as required.
3) Enable the MMU by setting bit 0 in the control register.

Care must be taken if the translated address differs from the untranslated address as several instructions following the enabling of the MMU may have been prefetched with the MMU off (using physical = virtual address - flat translation) and enabling the MMU may be considered as a branch with delayed execution. A similar situation occurs when the MMU is disabled. Consider the following code sequence:

```
MRC     p15, 0, R1, c1, C0, 0: Read control rejection
ORR     R1, #0x1
MCR     p15,0,R1,C1, C0,0 ; Enable MMUS
Fetch Flat
Fetch Flat
Fetch Translated
```

The instruction and data caches can be enabled simultaneously with the MMU using a single MCR instruction.

DISABLING THE MMU

To disable the MMU, clear bit 0 in the control register. The data cache should be disabled prior to, or, at the same time as the MMU is disabled by clearing Bit 2 of the control register. See the paragraph in *Enabling the MMU* regarding prefetch effects.

NOTE

If the MMU is enabled, then disabled and subsequently re-enabled the contents of the TLBs will have been preserved. If these are now invalid, the TLBs should be invalidated before the MMU is re-enabled. See Register 8: TLB operations on page 2-18.

NOTES

Appendix 4

CACHES, WRITE BUFFER

ABOUT THE CACHES AND WRITE BUFFER

The ARM920T includes an instruction cache, a data cache, a write buffer and a Physical Address TAG RAM to reduce the effect of main memory bandwidth and latency on performance.

- The ARM920T implements separate 16KB instruction and 16KB data caches.
- The caches have the following features:
- Virtually-addressed 64-way associative cache.
- 8 words per line (32 bytes per line) with one valid bit and two dirty bits per line, allowing half-line write-backs.
- Write-through and write-back cache operation (write-back caches are also known as copy back caches), selected per memory region by the C and B bits in the MMU translation tables (for data cache only).
- Pseudo-random or round-robin replacement, selectable via RR bit in CP15 register 1.
- Low-power CAM-RAM implementation.
- Caches independently lockable with granularity of 1/64th of cache, which is 64 words (256 bytes).
- For compatibility with Microsoft WindowsCE, and to reduce interrupt latency, the physical address corresponding to each data cache entry is stored in the physical address TAG RAM for use during cache line write-backs, in addition to the virtual address TAG stored in the cache CAMs. This means that the MMU is not involved in cache write-back operations, removing the possibility of TLB misses related to the write-back address.
- Cache maintenance operations to provide efficient cleaning of the entire data cache, and to provide efficient cleaning and invalidation of small regions of virtual memory. The latter allows ICache coherency to be efficiently maintained when small code changes occur, for example self-modifying code and changes to exception vectors.

The write buffer can hold 16 words of data and four addresses.

INSTRUCTION CACHE

The ARM920T includes a 16KB instruction cache. The ICache has 512 lines of 32 bytes (8 words), arranged as a 64-way set-associative cache and uses modified virtual addresses, translated by CP15 register 13 (see Address translation on page 3-4), from the ARM9TDMI core.

The ICache implements allocate-on-read-miss. Random or round-robin replacement can be selected under software control via the RR bit (CP15 register 1, bit 14). Random replacement is selected at reset.

Instructions can also be locked in the ICache such that they cannot be overwritten by a linefill. This operates with a granularity of 1/64th of the cache, which is 64 words (256 bytes).

All instruction accesses are subject to MMU permission and translation checks. Instruction fetches which are aborted by the MMU will not cause linefills or instruction fetches to appear on the ASB.

For clarity, the I bit (bit 12 in CP15 register 1) is referred to as the Icr bit throughout the following text. The C bit from the MMU translation table descriptor corresponding to the address being accessed is referred to as Ctt.

INSTRUCTION CACHE ENABLE/DISABLE

On reset, the ICache entries are all invalidated and the ICache is disabled.

The ICache is enabled by writing 1 to the Icr bit, and disabled by writing 0 to the Icr bit.

The ICache is usually used with the MMU enabled, in which case the C bit in the relevant MMU translation table descriptor indicates whether an area of memory is cacheable. If the ICache is enabled with the MMU disabled, all instruction fetches are treated as cacheable.

When the ICache is disabled, the cache contents are ignored and all instruction fetches appear on the ASB as separate non-sequential accesses.

NOTE

ARM920T implements a non-sequential access on the ASB as an A-TRAN cycle followed by an S-TRAN cycle. It does not produce N-TRAN cycles.

If the cache is subsequently re-enabled its contents will be unchanged. If the contents are no longer coherent with main memory the ICache should be invalidated prior to being enabled (see Register 7: Cache operations on page 2-15).

The MMU and ICache can be enabled simultaneously by writing a 1 to bit 0 and to bit 12 in CP15 register 1 with a single MCR instruction.

INSTRUCTION CACHE OPERATION

If the ICache is disabled, each instruction fetch results in a separate non-sequential memory access on the ASB, giving very low performance to burst memory such as page mode DRAM or synchronous DRAM. Therefore, the ICache should be enabled as soon as possible after reset.

If the ICache is enabled, an ICache lookup is performed for each instruction fetch regardless of the setting of the Ctt bit in the relevant MMU translation table descriptor. If the required instruction is found in the cache, the lookup is called a cache hit. If the required instruction is not found in the cache, the lookup is called a cache miss.

If the instruction fetch is a cache hit and Ctt = 1 indicating a cacheable region of memory, then the instruction is returned from the cache to the ARM9TDMI CPU core. If it is a cache miss and Ctt = 1, then an 8-word linefill will be performed, possibly replacing another entry. The entry to be replaced, (called the victim), is chosen from the entries which are not locked using either a random or round-robin replacement policy.

If Ctt = 0, indicating a non-cacheable region of memory, then a single non-sequential memory access will appear on the ASB.

NOTE

If Ctt=0, indicating a non-cacheable region of memory, then the cache lookup should result in a cache miss. The only way that it can result in a cache hit is if software has changed the value of the Ctt bit in the MMU translation table descriptor without invalidating the cache contents. This is a programming error, as the behavior in this case is architecturally unpredictable and varies between implementations.

INSTRUCTION CACHE REPLACEMENT ALGORITHM

The ICache and DCache replacement algorithm is selected by the RR bit in the CP15 control register (CP15 register 1, bit 14). Random replacement is selected at reset. Setting the RR bit to 1 selects round-robin replacement.

INSTRUCTION CACHE LOCKDOWN

Instructions can be locked into the ICache, causing the ICache to guarantee a hit, and providing optimum and predictable execution time.

Instructions are locked into the ICache by first ensuring the code to be locked is not already in the cache. This is tested by flushing either the whole ICache or specific lines. A short software routine can then be used to load the instructions into the ICache. The software routine must either be non-cacheable, or already in the ICache, but not in an ICache line which is about to be overwritten. The instructions to be loaded must be from a memory region which is cacheable.

The software routine operates by writing to CP15 register 9 to force the replacement counter to a specific ICache line and by using the prefetch ICache line operation available via CP15 register 7 to force the ICache to perform a lookup. This will miss and a linefill will be performed loading the cache line into the entry specified by the replacement counter. Once all the instructions have been loaded, they are then locked by writing to CP15 register 9 to set the replacement counter base to be one higher than the number of locked cache lines.

See Data cache lockdown on page 4-9 for a more complete explanation of cache locking.

DATA CACHE AND WRITE BUFFER

The ARM920T includes a 16KB data cache and a write buffer to reduce the effect of main memory bandwidth and latency on data access performance. The DCache has 512 lines of 32 bytes (8-words), arranged as a 64-way set-associative cache and uses virtual addresses from the ARM9TDMI CPU core. The write buffer can hold up to 16 words of data and 4 separate addresses. The operation of the data cache and write buffer are intimately connected.

The DCache supports write-through (WT) and writeback (WB) memory regions, controlled by the C and B bits in each section and page descriptor within the MMU translation tables. For clarity, these bits are referred to as Ctt and Btt in the following text. For details see Data cache and write buffer operation on page 4-6.

Each DCache line has two dirty bits, one for the first 4-words of the line, the other for the last 4-words, and a single virtual TAG address and valid bit for the entire 8-word line. The physical address from which each line was loaded is stored in the PA TAG RAM and is used when writing modified lines back to memory.

A linefill always loads a complete 8-word line.

When a store hits in the DCache, if the memory region is WB, the associated dirty bit is set marking the appropriate half-line as being modified. If the cache line is replaced due to a linefill, or if the line is the target of a DCache clean operation, the dirty bits are used to decide whether the whole, half, or none of the line is written back to memory. The line is written back to the same physical address from which it was loaded, regardless of any changes to the MMU translation tables.

The DCache implements allocate-on-read-miss. Random or round-robin replacement can be selected under software control via the RR bit (CP15 register 1, bit 14). Random replacement is selected at reset.

Data can also be locked in the DCache such that it cannot be overwritten by a linefill. This operates with a granularity of 1/64th of the cache, which is 64 words (256 bytes).

All data accesses are subject to MMU permission and translation checks. Data accesses which are aborted by the MMU will not cause linefills or data accesses to appear on the ASB.

For clarity, the C bit (bit 2 in CP15 register 1) is referred to as the Ccr bit throughout the following text.

DATA CACHE AND WRITE BUFFER ENABLE/DISABLE

On reset, all DCache entries are invalidated, the DCache is disabled, and the write buffer contents are discarded.

There is no explicit write buffer enable bit implemented in ARM920T. Situations in which the write buffer is used are described below.

The DCache is enabled by writing 1 to the Ccr bit, and disabled by writing 0 to the Ccr bit.

The DCache must be enabled only when the MMU is enabled. This is because the MMU translation tables define the cache and write buffer configuration for each memory region.

When the DCache is disabled the cache contents are ignored and all data accesses appear on the Advanced System Bus as separate non-sequential accesses. If the cache is subsequently re-enabled its contents will be unchanged. Depending on the software system design, the cache may need to be cleaned after it is disabled, and invalidated before it is re-enabled. See Cache coherence on page 4-10.

The MMU and DCache can be enabled or disabled simultaneously with a single MCR which changes bit 0 and bit 2 in the control register (CP15 register 1).

DATA CACHE AND WRITE BUFFER OPERATION

The DCache and write buffer configuration of each memory region is controlled by the C and B bits in each section and page descriptor in the MMU translation tables. For clarity, these bits are referred to as Ctt and Btt in the following text. The configuration is modified by the DCache enable bit in the CP15 control register, which is referred to as Ccr.

If the DCache is enabled, a DCache lookup is performed for each data access initiated by the ARM9TDMI CPU core, regardless of the value of the Ctt bit in the relevant MMU translation table descriptor. If the accessed virtual address matches the virtual address of an entry in the cache, the lookup is called a cache hit. If the required address does not match any entry in the cache, the lookup is called a cache miss. In this context a data access means any type of load (read) or store (write) or swap instruction, including LDR, LDRB, LDRH, LDM, LDC, STR, STRB, STRH, STC, SWP and SWPB.

To ensure that accesses appear on the ASB in program order, ARM920T will wait for all writes in the write buffer to complete on the ASB before starting any other ASB access. The ARM9TDMI CPU core can continue executing at full speed reading instructions and data from the caches and writing to the DCache and write buffer while buffered writes are being written to memory via the ASB.

Table 4-1 describes the DCache and write buffer behavior for each type of memory configuration. Ctt AND Ccr means the bitwise Boolean AND of Ctt with Ccr.

Table 4-1. Data Cache and Write Buffer Configuration

Ctt and Ccr	Btt	Data cache, write buffer and memory access behavior
0 (1)	0	Non-cached, non-buffered (NCNB) Reads and writes are not cached and always perform accesses on the ASB and may be externally aborted. Writes are not buffered. The CPU halts until the write is completed on the ASB. Cache hits should never occur. (2)
0	1	Non-cached buffered (NCB) Reads and writes are not cached, and always perform accesses on the ASB. Cache hits should never occur. Writes are placed in the write buffer and will appear on the ASB. The CPU continues execution as soon as the write is placed in the write buffer. Reads may be externally aborted. Writes can not be externally aborted.
1	0	Cached, write-through mode (WT) Reads which hit in the cache will read the data from the cache and do not perform an access on the ASB. Reads which miss in the cache cause a linefill. All writes are placed in the write buffer and will appear on the ASB. The CPU continues execution as soon as the write is placed in the write buffer. Writes which hit in the cache update the cache. Writes cannot be externally aborted.
1	1	Cached, write-back mode (WB) Reads which hit in the cache will read the data from the cache and do not perform an ASB access. Reads which miss in the cache cause a linefill. Writes which miss in the cache are placed in the write buffer and will appear on the ASB. The CPU continues execution as soon as the write is placed in the write buffer. Writes which hit in the cache update the cache and mark the appropriate half of the cache line as dirty, and do not cause an ASB access. Cache write-backs are buffered. Writes (Cache write-misses and cache write-backs) cannot be externally aborted.

NOTES:

1. The control register C bit (Ccr) being zero disables all lookups in the cache, while the translation table descriptor C bit (Ctt) being zero only stops new data being loaded into the cache. With Ccr = 1 and Ctt = 0 the cache will still be searched
   on every access to check whether the cache contains an entry for the data.
2. It is an operating system software error if a cache hit occurs when reading from, or writing to, a region of memory marked as NCNB or NCB. The only way this can occur is if the operating system changes the value of the C and B bits in a page table descriptor, while the cache contains data from the area of virtual memory controlled by that descriptor. The cache and memory system behavior resulting from changing the page table descriptor in this way is unpredictable. If the operating system needs to change the C and B bits of a page table descriptor, it must ensure that the caches do not contain any data controlled by that descriptor. In some circumstances, the operating system may need to clean and flush the caches to ensure this.

A linefill performs an 8-word burst read from the ASB and places it as a new entry in the cache, possible replacing another line at the same location within the cache. The location which is replaced (called the victim) is chosen from the entries which are not locked using either a random or round-robin replacement policy. If the cache line being replaced is marked as dirty, indicating that it has been modified and that main memory has not been updated to reflect the change, a cache writeback occurs.

Depending on whether one or both halves of the cache line are dirty, the writeback will perform a 4 or 8-word sequential burst write access on the ASB. The writeback data is placed in the write buffer and then the linefill data is read from the ASB. The CPU can the continue while the writeback data is written to memory via the ASB.

Load multiple (LDM) instructions accessing NCNB or NCB regions perform sequential bursts on the ASB. Store multiple (STM) instructions accessing NCNB regions also perform sequential bursts on the ASB.

The sequential burst will be split into two bursts if it crosses a 1KB boundary. This is because the smallest MMU protection and mapping size is 1KB, so the memory regions on each size of the 1KB boundary may have different properties.

This means that no sequential access generated by ARM920T will cross a 1KB boundary, which can be exploited to simplify memory interface design. For example, a simple page mode DRAM controller could perform a page-mode access for each sequential access, provided the DRAM page size is 1KB or larger.

See also Cache coherence on page 4-10.

DATA CACHE REPLACEMENT ALGORITHM

The DCache and ICache replacement algorithm is selected by the RR bit in the CP15 Control register (CP15 register 1, bit 14). Random replacement is selected at reset. Setting the RR bit to 1 selects round-robin replacement.

SWAP INSTRUCTIONS

Swap instruction (SWP or SWPB) behavior is dependent on whether the memory region is cacheable or non-cacheable.

Swap instructions to cacheable regions of memory are useful for implementing semaphores or other synchronization primitives in multithreaded uniprocessor software systems.

Swap instructions to non-cacheable memory regions are useful for synchronization between two bus masters in a multi-master bus system. This could be two processors, or a processor and a DMA controller.

When a swap instruction accesses a cacheable region of memory (WT or WB), the DCache and write buffer behavior will be the same as having a load followed by a store according to the normal rules described. The BLOK pin will not be asserted during the execution of the instruction. It is guaranteed that no interrupt can occur between the load and store portions of the swap.

When a swap instruction accesses a non-cacheable (NCB or NCNB) region of memory, the write buffer is drained, and a single word or byte will be read from the ASB. The write portion of the swap will then be treated as non-bufferable, regardless of the value of Btt, and the processor stalled until the write is completed on the ASB. The BLOK pin will be asserted to indicate that the read and write should be treated as an atomic operation on the bus.

Like all other data accesses, a swap to a non-cacheable region which hits in the cache indicates a programming error.

DATA CACHE ORGANIZATION

The DCache is organized as 8 segments, each containing 64 lines, and each line containing 8-words. The line's position within its segment is a number from 0 to 63 which is called the index. A line in the cache can be uniquely identified by its segment and index. The index is independent of the line's virtual address. The segment is selected by bits [7:5] of the virtual address of the line.

Bits [4:2] of the virtual address specify which word within a cache line is accessed. For halfword operations, bit [1] of the virtual address specifies which halfword is accessed within the word. For byte operations, bits [1:0] specify which byte within the word is accessed.

Bits [31:8] of the virtual address of the each cache line is called the TAG. The virtual address TAG is stored in the cache along with the 8-words of data, when the line is loaded by a linefill.

Cache lookups compare bits [31:8] of the modified virtual address of the access with the stored TAG to determine whether the access is a hit or miss. The cache is therefore said to be virtually addressed.

DATA CACHE LOCKDOWN

Data can be locked into the DCache causing the DCache to guarantee a hit, and providing optimum and predictable execution time.

When no data is locked in the DCache, and a linefill occurs, the replacement algorithm chooses a victim cache line to be replaced by selecting an index in the range (0 to 63). The segment is specified by bits [7:5] of the virtual address of the data access which missed.

Data is locked into the DCache by restricting the range of victim numbers produced by the replacement algorithm, so that some cache lines are never selected as victims. The base pointer for the DCache victim generator can be set by writing to CP15 register 9. The replacement algorithm chooses a victim cache line in the range (base to 63), locking in the cache the lines with index in the range (0 to base - 1).

Data is loaded and locked into the DCache by first ensuring the data to be locked is not already in the cache. This can be ensured by cleaning and flushing either the whole DCache or specific lines. A short software routine can then be used to load the data into the DCache.

The software routine to load the data operates by writing to CP15 register 9 to force the replacement counter to a specific DCache line and then executing a load instruction to perform a cache lookup. This will miss and a linefill will be performed, bringing 8 words of data into the cache line specified by the replacement counter, in the segment specified by bits [7:5] of the modified virtual address accessed by the load.

To load further lines into the cache, the software routine can loop performing one load from each line to be loaded. As each line contains 8 words, each loop should add 32 (bytes) to the load address. The software routine needs to move the victim counter to the next index after it has loaded a line into the last available segment with the current index. As there are 8-segments, this will occur after 8-cache lines have been loaded.

Once all the data has been loaded, it is locked by writing to CP15 register 9 to move the replacement counter base to be one higher than the highest index of the locked cache lines.

The software routine that loads and locks the data in the DCache can be located in a cacheable region of memory providing it does not contain any loads or stores other than the loads which are used to bring the data to be locked into the DCache. The data to be loaded must be from a memory region which is cacheable.

CACHE COHERENCE

The ICache and DCache contain copies of information normally held in main memory. If these copies of memory information get out of step with each other because one is updated and the others are not updated, they are said to have become incoherent. If the DCache contains a line which has been modified by a store or swap instruction, and the main memory has not been updated, the cache line is said to be dirty. Clean operations force the cache to write dirty lines back to main memory.

On the ARM920T, software is responsible for maintaining coherence between main memory, the ICache and the DCache.

Register 7: Cache operations on page 2-15 describes facilities for invalidating the entire ICache or individual ICache lines, and for cleaning or invalidating DCache lines, or for invalidating the entire DCache.

To clean the entire DCache efficiently, software should loop though each cache entry using the clean D single entry (using index) operation or the clean and invalidate D entry (using index) operation. This should be performed by a two-level nested loop going though each index value for each segment. See Data cache organization on page 4-9.

DCache, ICache, and memory coherence is generally achieved by:

- cleaning the DCache to ensure memory is up to date with all changes
- invalidating the ICache to ensure that the ICache is forced to re-load instructions from memory.

Software can minimize the performance penalties of cleaning and invalidating caches by:

- Cleaning only small portions of the cache when only a small area of memory needs to be made coherent, for example, when updating an exception vector entry.
- Invalidating only small portions of the ICache when only a small number of instructions are modified, for example, when updating an exception vector entry.
- Not invalidating the ICache in situations where it is known that the modified area of memory cannot be in the cache, for example, when mapping a new page into the currently running process.

The ICache needs to be made coherent with a changed area of memory after any changes to the instructions which appear at a virtual address, and before the new instructions are executed.

Dirty data in the DCache can be pushed out to main memory by cleaning the cache.

Situations which necessitate cache cleaning and invalidating include:

- writing instructions to a cacheable area of memory using STR or STM instructions, for example:
  - self-modifying code
  - JIT compilation
  - copying code from another location
  - downloading code via the EmbeddedICE JTAG debug features
  - updating an exception vector entry.
- another bus master, such as a DMA controller, modifying a cacheable area main memory
- turning the MMU on or off
- changing the virtual-to-physical mappings in the MMU page tables
- turning the ICache or DCache on, if its contents are no longer coherent.

The DCache should be cleaned, and both caches invalidated, before the cache and write buffer configuration of an area of memory is changed by modifying Ctt or Btt in the MMU translation table descriptor. This is not necessary if it is known that the caches cannot contain any entries from the area of memory whose translation table descriptor is being modified.

Changing the process ID in CP15 register 13 does not change the contents of the cache or memory, and does not affect the mapping between cache entries and physical memory locations. It only changes the mapping between ARM9TDMI addresses and cache entries. This means that changing the process ID does not lead to any coherency issues. No cache cleaning or cache invalidation is required when the process ID is changed.

At reset the DCache and ICache entries are all invalidated and the DCache and ICache are disabled.

The software design also needs to consider that the pipelined design of the ARM9TDMI core means that it fetches three instructions ahead of the current execution point. So, for example, the three instructions following an MCR which invalidates the ICache, will have been read from the ICache before it is invalidated.

CACHE CLEANING WHEN LOCKDOWN IS IN USE

The clean D single entry (using index) and clean and invalidate D entry (using index) operations can leave the victim pointer set to the index value used by the operation. In some circumstances, if DCache locking is in use, this could leave the victim pointer in the locked region, leading to locked data being evicted from the cache. The victim pointer can be moved outside the locked region by implementing the cache loop enclosed by the reading and writing of the Base and Victim pointer:

```
MRC p15, 0, Rd, c9, c0, 0        ; Read D Cache Base into Rd

Index Clean or Index Clean and Invalidate loops

MCR p15, 0, Rd, c9, c0, 0        ; Write D Cache Base and Victim from Rd
```

Clean D single entry (using VA) and clean and invalidate D entry (using VA) operations do not move the victim pointer, so there is no need to reposition the victim pointer after using these operations.

IMPLEMENTATION NOTES

This section describes the behavior of the ARM920T implementation in areas which are architecturally unpredictable. For portability to other ARM implementations, software should not depend on this behavior.

A read from a non-cacheable (NCB or NCNB) region which unexpectedly hits in the cache will still read the required data from the ASB. The contents of the cache will be ignored, and the cache contents will not be modified. This includes the read portion of a swap (SWP or SWPB) instruction.

A write to a non-cacheable (NCB or NCNB) region which unexpectedly hits in the cache will update the cache and will still cause an access on the ASB.

PHYSICAL ADDRESS TAG RAM

The ARM920T implements a PA TAG RAM in order to perform write backs from the data cache.

A write back occurs when dirty data that is about to be overwritten by linefill data comes from a memory region that is marked as a write back region. This data is written back to main memory to maintain memory coherency.

Dirty data is data that has been modified in the cache, but not updated in main memory.

When a line is written into the data cache, the physical address TAG (DPA[31:5]) is written into the PA TAG RAM. If this line comes to be written back to main memory, the PA TAG RAM is indexed into by the data cache and the physical address (WBPA[31:0]) is returned to the AMBA Bus interface so that it can perform the write back.

The PA TAG RAM Array for a 16k data cache comprises 8 segments x 64 rows/segment x 26 bits/row. There are two test interfaces to the PA TAG RAM:

Debug interface, see Scan chain 4 - debug access to the PA TAG RAM

AMBA test interface, see PA TAG RAM test

Appendix 5

CLOCK MODES

OVERVIEW

The ARM920T has two functional clock inputs, BCLK and FCLK. Internally, the ARM920T is clocked by GCLK, which can be seen on the CPCLK output as shown in Figure 5-1. GCLK can be sourced from either BCLK or FCLK depending on the clocking mode, selected using nF bit and iA bit in CP15 register 1 (see Register 1: Control register on page 2-10). The three clocking modes are FastBus, synchronous and asynchronous.

The ARM920T is a static design and both clocks can be stopped indefinitely without loss of state. From Figure 5-1 it can be seen that some of the ARM920T macrocell signals will have timing specified with relation to GCLK, which can be either FCLK or BCLK depending on the clocking mode

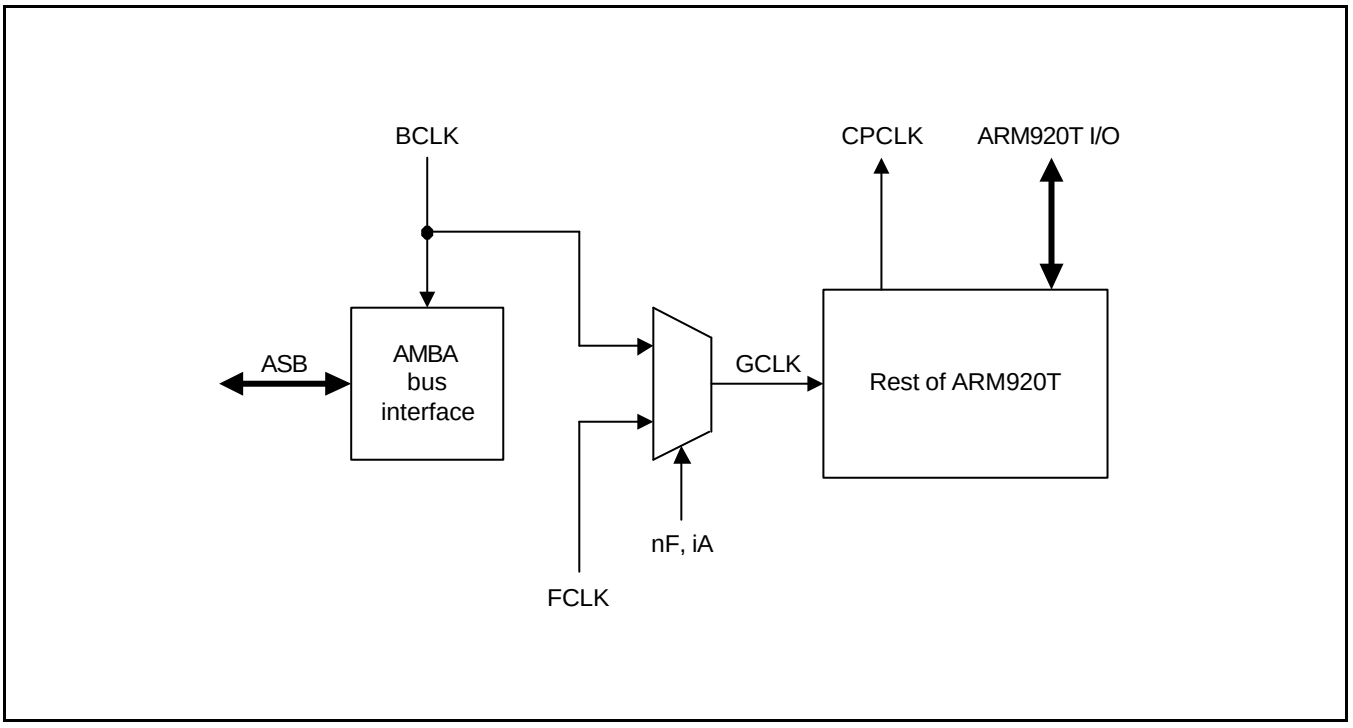


Figure 5-1. ARM920T Clocking

FASTBUS MODE

In this mode of operation the BCLK input is the source for GCLK. The FCLK input is ignored. This mode is typically used in systems with high speed memory.

SYNCHRONOUS MODE

This mode is typically used in systems with low speed memory. In this mode GCLK can be sourced from BCLK and FCLK. BCLK is used to control the AMBA memory interface. FCLK is used to control the internal ARM9TDMI processor core and any cache operations. FCLK must have a higher frequency and must also be an integer multiple of BCLK, with a BCLK transition only when FCLK is HIGH. An example is shown in Figure 5-2.

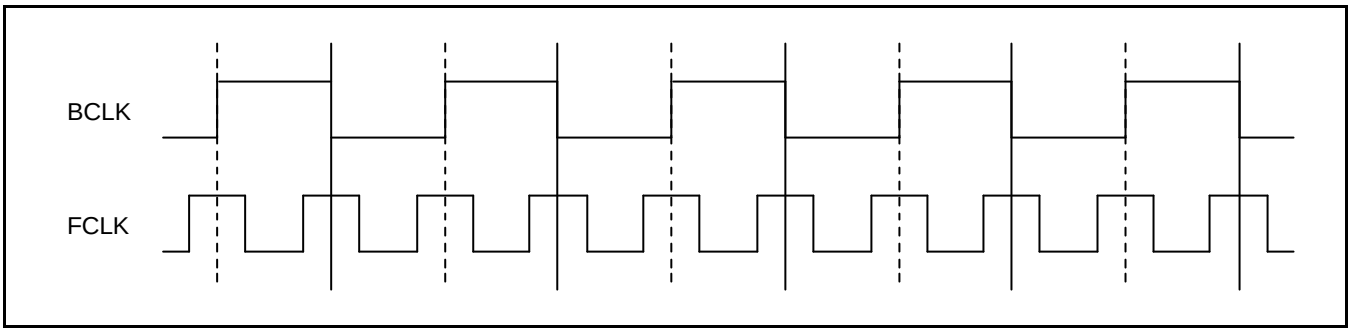


Figure 5-2. Synchronous Clocking Mode

If the ARM920T performs an external access, for example, a cache linefill, the ARM920T will switch to BCLK to perform the access. The delay when switching from FCLK to BCLK is a minimum of one FCLK phase and a maximum of one BCLK cycle. An example of the clock switching is shown in Figure 5-3 . The delay when switching from BCLK to FCLK is a maximum of one FCLK phase.

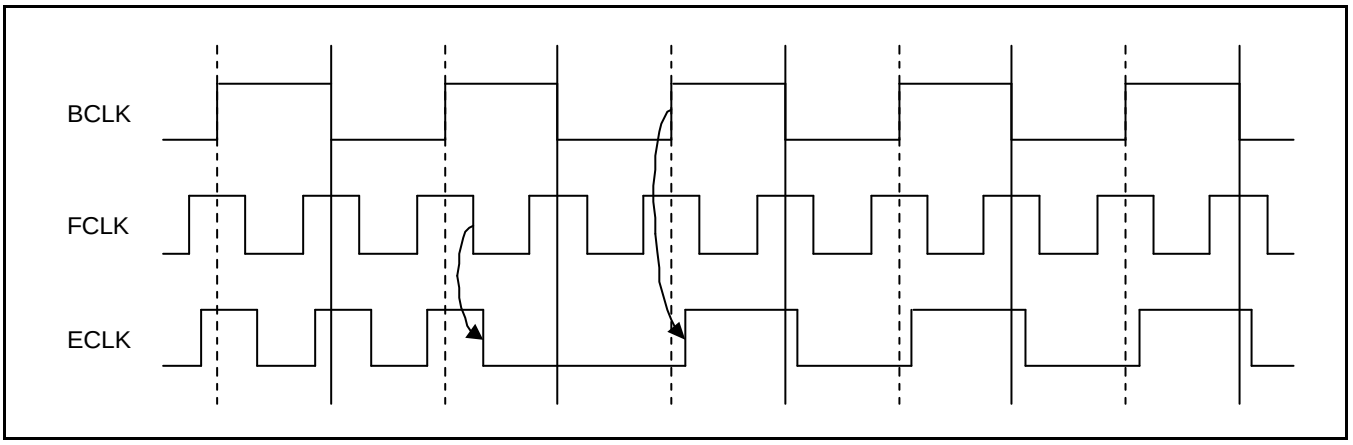


Figure 5-3. Switching from FCLK to BCLK in Synchronous Mode

Care must be taken if BCLK is stopped by the system so that when BCLK is restarted it does not violate any of the above restrictions.

ASYNCHRONOUS MODE

This mode is typically used in systems with low speed memory. In this mode of operation GCLK can be sourced from BCLK and FCLK. BCLK is used to control the AMBA memory interface. FCLK is used to control the internal ARM9TDMI processor core and any cache operations. The one restriction is that FCLK must have a higher frequency than BCLK. An example is shown in Figure 5-4.

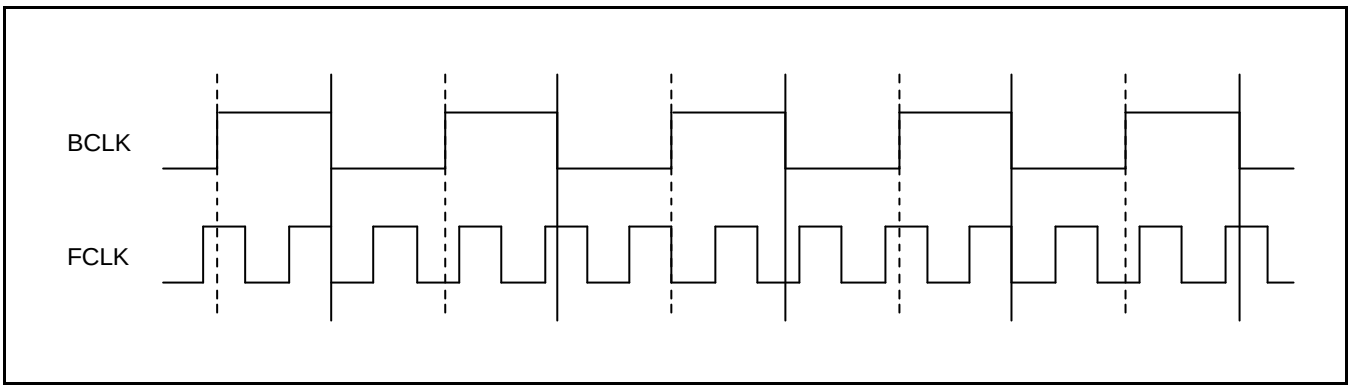


Figure 5-4. Asynchronous Clocking Mode

If the ARM920T performs an external access, for example, a cache miss or a cache line fill, ARM920T will switch to BCLK to perform the access. The delay when switching from FCLK and BCLK is a minimum of one BCLK cycle, and a maximum of one and a half BCLK cycles. An example of the clock switching is shown in Figure 5-4 . When switching from BCLK to FCLK the minimum delay is one FCLK cycle and the maximum delay is one and a half FCLK cycles. An example of the clock switching is shown in Figure 5-5.

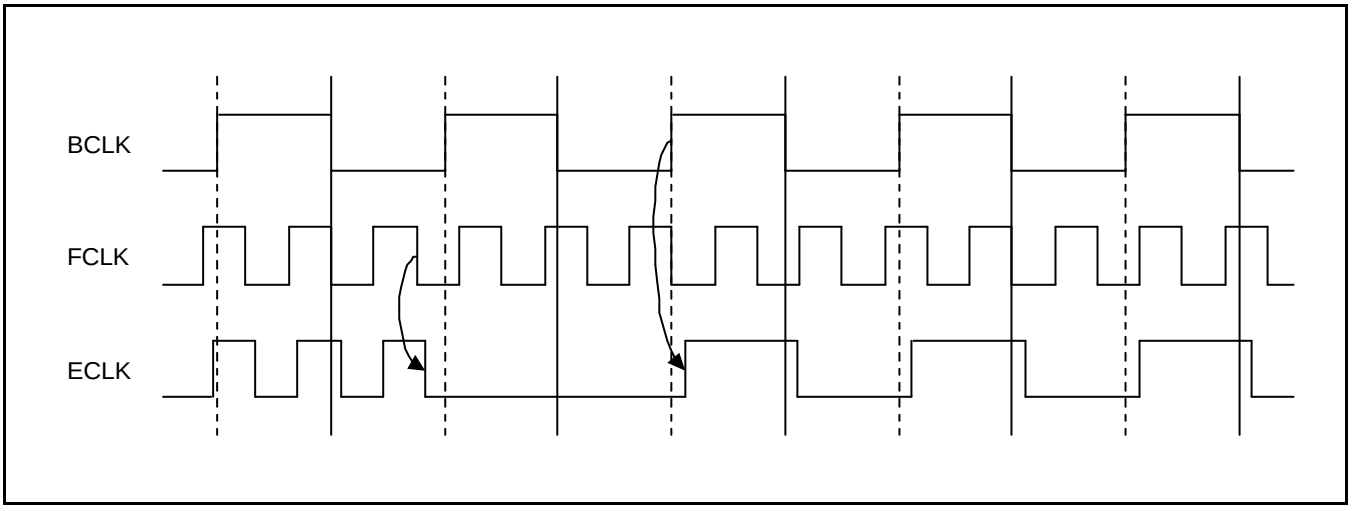


Figure 5-5. Switching from FCLK to BCLK in Asynchronous Mode

NOTES

S3C SERIES MASK ROM ORDER FORM

Product description:

Device Number: S3C_____- _____(write down the ROM code number)

Product Order Form: ☐ Package ☐ Pellet ☐ Wafer Package Type: _____

Package Marking (Check One):

☐ Standard ☐ Custom A (Max 10 chars) ☐ Custom B (Max 10 chars each line)

Standard	Custom A	Custom B
SEC @ YWW Device Name	@ YWW Device Name	@ YWW

@ : Assembly site code, Y : Last number of assembly year, WW : Week of assembly

Delivery Dates and Quantities:

Deliverable	Required Delivery Date	Quantity	Comments
ROM code	–	Not applicable	See ROM Selection Form
Customer sample			
Risk order			See Risk Order Sheet

Please answer the following questions:

☞ **For what kind of product will you be using this order?**

☐ New product ☐ Upgrade of an existing product

☐ Replacement of an existing product ☐ Other

If you are replacing an existing product, please indicate the former product name

( )

☞ **What are the main reasons you decided to use a Samsung microcontroller in your product?**

Please check all that apply.

☐ Price ☐ Product quality ☐ Features and functions

☐ Development system ☐ Technical support ☐ Delivery on time

☐ Used same micom before ☐ Quality of documentation ☐ Samsung reputation

Mask Charge (US$ / Won): _____

Customer Information:

Company Name: _____ Telephone number _____

Signatures: _____ _____

(Person placing the order) (Technical Manager)

(For duplicate copies of this form, and for additional ordering information, please contact your local Samsung sales representative. Samsung sales offices are listed on the back cover of this book.)

S3C SERIES
REQUEST FOR PRODUCTION AT CUSTOMER RISK

Customer Information:

Company Name: _____

Department: _____

Telephone Number: _____ Fax: _____

Date: _____

Risk Order Information:

Device Number: S3C_____- _____ (write down the ROM code number)

Package: Number of Pins: _____ Package Type: _____

Intended Application: _____

Product Model Number: _____

Customer Risk Order Agreement:

We hereby request SEC to produce the above named product in the quantity stated below. We believe our risk order product to be in full compliance with all SEC production specifications and, to this extent, agree to assume responsibility for any and all production risks involved.

Order Quantity and Delivery Schedule:

Risk Order Quantity: _____ PCS

Delivery Schedule:

Delivery Date (s)	Quantity	Comments

Signatures: _____ _____

(Person Placing the Risk Order) (SEC Sales Representative)

(For duplicate copies of this form, and for additional ordering information, please contact your local Samsung sales representative. Samsung sales offices are listed on the back cover of this book.)

S3C2410A MASK OPTION SELECTION FORM

Device Number: S3C2410A -_____(write down the ROM code number)

Attachment (Check one): ☐ Diskette ☐ PROM

Customer Checksum: _____

Company Name: _____

Signature (Engineer): _____

Please answer the following questions:

☞ **Application** (Product Model ID: _____)

☐ Audio ☐ Video ☐ Telecom

☐ LCD Databank ☐ Caller ID ☐ LCD Game

☐ Industrials ☐ Home Appliance ☐ Office Automation

☐ Remocon ☐ Other

Please describe in detail its application

_____

_

(For duplicate copies of this form, and for additional ordering information, please contact your local Samsung sales representative. Samsung sales offices are listed on the back cover of this book.)