21.0-S3-C2400X-022002

USER'S MANUAL

S3C2400X

32-Bit RISC Microprocessor

Revision 1.0

SAMSUNG ELECTRONICS

NOTIFICATION OF REVISIONS

ORIGINATOR: Samsung Electronics, SOC Development Group, Ki-Heung, South Korea

PRODUCT NAME: S3C2400X RISC Microcontroller

DOCUMENT NAME: S3C2400X User's Manual, Revision 1.0

DOCUMENT NUMBER: 21.0-S3-C2400X-022002

EFFECTIVE DATE: February, 2002

SUMMARY: As a result of additional product testing and evaluation, some specifications published in the S3C2400X User's Manual, Revision 0.1, have been changed. These changes for S3C2400X microcontroller are described in detail in the *Revision Descriptions* section below.

DIRECTIONS: Please note the changes in your copy (copies) of the S3C2400X User's Manual, Revision 1.0 Or, simply attach the *Revision Descriptions* of the next page to S3C2400X User's Manual, Revision 1.0

REVISION HISTORY

Revision	Date	Remark
1	August, 2000	There is no preliminary spec. This is the first edition.
2	September, 2001	
3	February, 2002	

REVISION DESCRIPTIONS

ERRATA from Preliminary S3C2400X01 User's Manual

This is a preliminary user's manual.
So, our company present its revision as the date on the page header.
After formal publishing, we will show its revision as a proper number.

[FORMAT]
🗁<CHAPTER> : <PAGE> : <CHANGED_ITEM> : <CHANGED_DATE>
<CHANGED DESCRIPTIONS>

🗁LCD CONTROLLER : 15-4 : 14th line from the bottom : 9/22/2001
A sentence is modified as follow.
In case of 4-bit dual scan display the number of valid VD data lines should be 4 and in case of 8-bit single scan display mode, the number of valid VD data lines should be 8.

🗁ELECTRICAL DATA : 23-1 : Table 23-1&Table 23-2 : !0/23/2001
Table 23-1 Absolute Maximum Rating & Table 23-2Recommended Operating Conditions were modified.

Table 23-1. Absolute Maximum Rating

Symbol	Parameter	Rating		Unit
V_{DD}	DC Supply Voltage	3.8		V
V_{IN}	DC Input Voltage	3.3 V Input buffer	3.8	
V_{OUT}	DC Input Voltage	3.3 V Output buffer	3.8	
I_{IN}	DC Input Current	± 200		mA
T_{STG}	Storage Temperature	– 65 to 150		^{o}C

Table 23-2. Recommended Operating Conditions

Symbol	Parameter	Rating		Unit
V_{DD}	DC Supply Voltage for I/O Block	3.3V V_{DD}	3.3±0.3	V
	DC Supply Voltage for Analog Core	3.3V V_{DD}	3.3±0.5%	
V_{IN}	DC Input Voltage	3.3V Input buffer	3.3±0.3	
V_{OUT}	DC Output Voltage	3.3V Output buffer	3.3±0.3	
T_{OPR}	Operating Temperature	Commercial	0 to 70	^{o}C

🗁**ELECTRICAL DATA : 23-2 : Table 23-3 : 10/23/2001**
Table 23-3 Normal I/O PAD DC Electrical Characteristics were modified.

Table 23-3. 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					V
	LVCMOS interface		2.0			
V_{IL}	Low level input voltage					V
	LVCMOS interface				0.8	
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			
I_{IH}	High level input current					
	Input buffer	$V_{IN} = V_{DD}$	-10		10	µA
	Input buffer with pull-up		10	33	60	
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 B8	I_{OH}= -8 mA	2.4			V
	Type B12	I_{OH} = -12 mA				
V_{OL}	Low level output voltage					
	Type B8	I_{OL} = 8 mA			0.4	V
	Type B12	I_{OL} = 12 mA				
I_{DS}	Stop current	$V_{IN} = V_{SS}$ or V_{DD}		TBD (note)		µA @25 °C
I_{DD}	Operating current				TBD (note)	mA/MHz

NOTE: TBD — To be Determined.

🗁**CLOCK & POWER MANAGEMENT : 6-22 : PLL VALUE SELECTION GUIDE : 11/12/2001**
PLL VALUE Selection Guide was modified as below.

PLL VALUE SELECTION GUIDE

1. Fout = m * Fin / p*s, where : m=M+8, p=P+2, s=2^s

2. Fin/(25*p) < 16.7e6/m < Fin/(10*p)

3. 0.7 < 6.48/sqrt(m) < 1.8

4. (Fin/p) * m < 600e6Mpll * 2^s has to be less than 300 MHz.

🗁**ELECTRICAL DATA : 23-8~23-24 : Figure 23-8~23-27 : 12/01/2001**
EXTCLK was changed to HCLK, and the Phase of Timing was reversed in all Figures from 23-8 to 23-27.

🗁**ELECTRICAL DATA : 23-5 : Figure 23-3~23-4 : 12/03/2001**
Timing diagrams of Figure 23-3 & 23-4 were modified as follows.

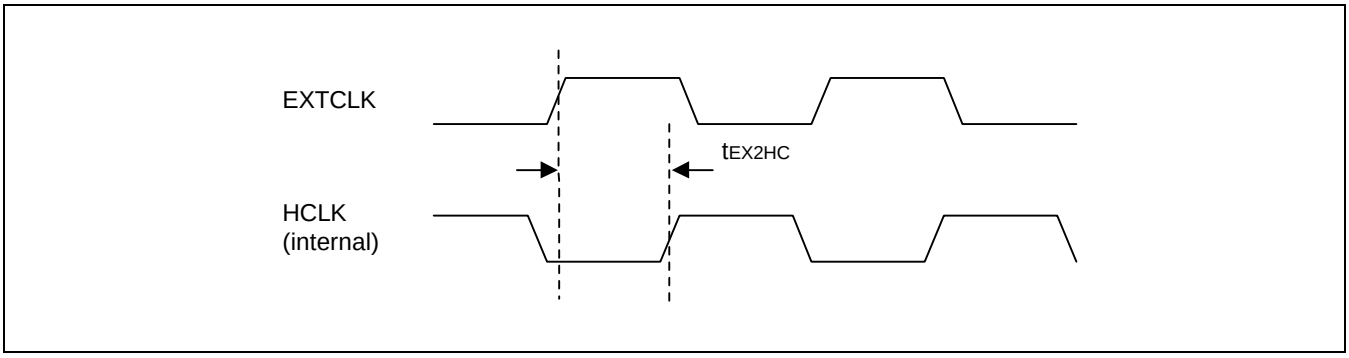


Figure 23-3. EXTCLK/HCLK in case that EXTCLK is used without the PLL

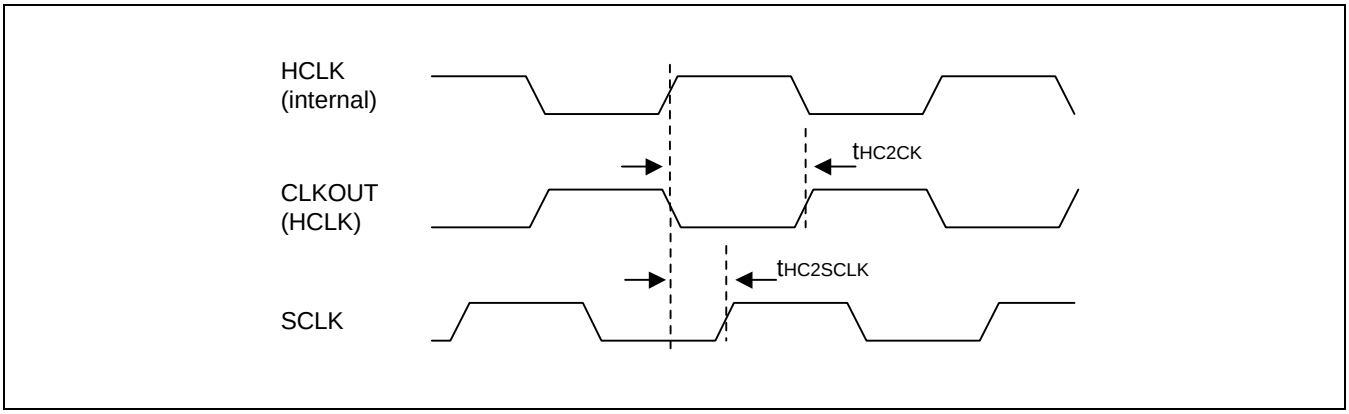


Figure 23-4. HCLK/CLKOUT/SCLK in case that EXTCLK is used

🗁**ELECTRICAL DATA : 23-39 : Table 23-6 : 12/03/2001**
Parameters of Table 23-6 were modified as follows.

Table 23-6. ROM/SRAM Bus Timing Constants

(V_{DD} = 1.8 V ± 0.15 V, T_A = 0 to 70 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ	Max	Unit
ROM/SRAM Address Delay	t_{RAD}	–	8	–	ns
ROM/SRAM Chip select Delay	t_{RCD}	–	7	–	ns
ROM/SRAM Output enable Delay	t_{ROD}	–	7	–	ns
ROM/SRAM read Data Setup time.	t_{RDS}	–	1	–	ns
ROM/SRAM read Data Hold time.	t_{RDH}	–	5	–	ns
ROM/SRAM Byte Enable Delay	t_{RBED}	–	7	–	ns
ROM/SRAM Write Byte Enable Delay	t_{RWBED}	–	7	–	ns
ROM/SRAM output Data Delay	t_{RDD}	–	7	–	ns
ROM/SRAM external Wait Setup time	t_{WS}	–	1	–	ns
ROM/SRAM external Wait Hold time	t_{WH}	–	5	–	ns
ROM/SRAM Write enable Delay	t_{RWD}	–	8	–	ns

🗁**ELECTRICAL DATA : 23-40 : Table 23-7 : 12/03/2001**
Parameters of Table 23-7 were modified as follows.

Table 23-7. Clock Timing Constants

(V_{DD} = 1.8 V ± 0.15 V, T_A = 0 to 70 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ	Max	Unit
DRAM Address Delay	t_{DAD}	–	8	–	ns
DRAM Row active Delay	t_{DRD}	–	7	–	ns
DRAM Read Column active Delay	t_{DRCD}	–	7	–	ns
DRAM Output enable Delay	t_{DOD}	–	7	–	ns
DRAM read Data Setup time	t_{DDS}	–	1	–	ns
DRAM read Data Hold time	t_{DDH}	–	5	–	ns
DRAM Write Cas active Delay	t_{DWCD}	–	7	–	ns
DRAM Cbr Cas active Delay	t_{DCCD}	–	7	–	ns
DRAM Write enable Delay	t_{DWD}	–	7	–	ns
DRAM output Data Delay	t_{DDD}	–	10	–	ns

🗁 **S3C2400X NOTICE IN PROGRAMMING: 25-1~25-3 : CHAPTER : 12/04/2001**
Programming notice is added for known problems

🗁 **PLL : 6-22 ~ 6-23 : DMA Control register parameter : 12/04/2001**
PLL value selection guide was modified as follows

PLL VALUE SELECTION GUIDE

p6-22

1. Fout = m * Fin / p*s, where : m=M+8, p=P+2, s=2^s

2. Fin/(25*p) < 16.7e6/m < Fin/(10*p)

3. 0.7 < 6.48/sqrt(m) < 1.8

4. (Fin/p) * m < 330e6

p6-23

It's difficult to find the proper PLL value. So, We recommended 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	37	1	3
12.00Mhz	33.75Mhz	37	0	3
12.00Mhz	44.25Mhz	51	2	2
12.00Mhz	48.00Mhz	40	1	2
12.00Mhz	51.00Mhz	43	1	2
12.00Mhz	56.00Mhz	48	1	2
12.00Mhz	67.50Mhz	37	0	2
12.00Mhz	79.00Mhz	71	1	2
12.00Mhz	84.00Mhz	34	1	1
12.00Mhz	90.00Mhz	37	1	1
12.00Mhz	102.00Mhz	43	1	1
12.00Mhz	112.50Mhz	67	2	1
12.00Mhz	118.50Mhz	71	2	1
12.00Mhz	124.00Mhz	54	1	1
12.00Mhz	133.50Mhz	81	2	1

🗁 **DMA : 8-3 : Figure 8-1 : 12/04/2001**

Timing diagrams of Figure 8-1 were modified as follows.

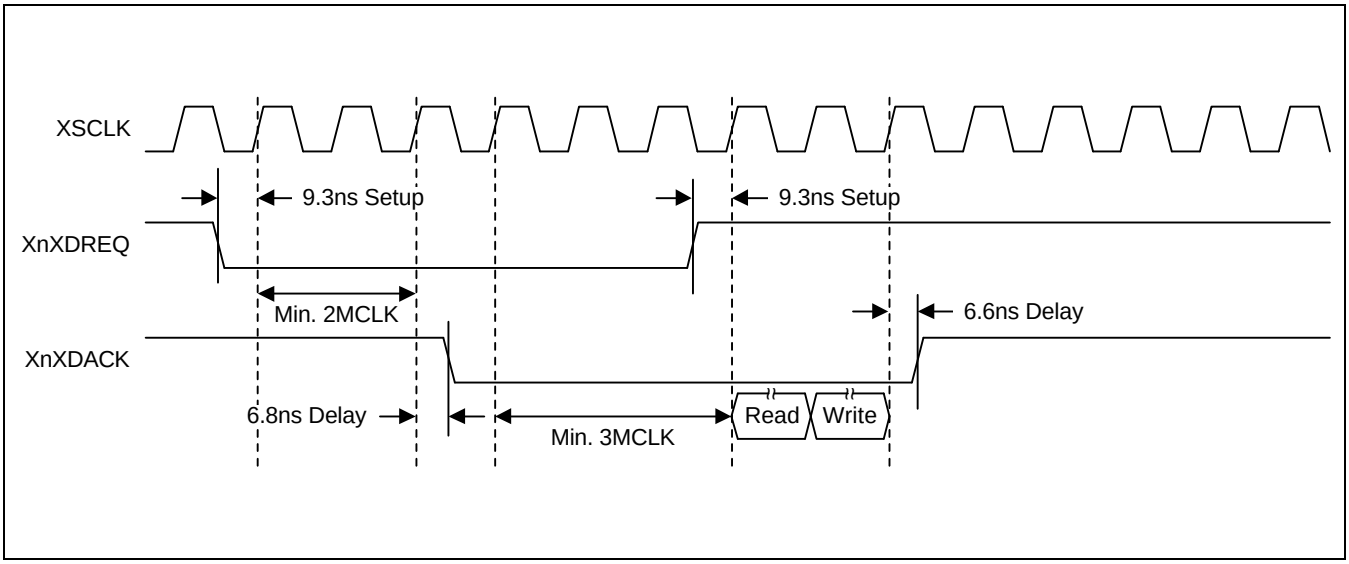


Figure 8-1. Basic DMA Timing Diagram

🗁 **DMA : 8-10 : DMA Control register parameter : 12/04/2001**

Parameter description was modified as follows

RELOAD	[22]	Set the auto-reload on/off option. 0 = auto reload is performed when the following 2 conditions - The current value of transfer counter becomes 0 - And then, the next first DMA request is received. 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

🗁 **MEMORY CONTROLLER: 5-5 : Figure 5-2 : 12/04/2001**
Timing diagrams of Figure 5-2 were modified as follows.

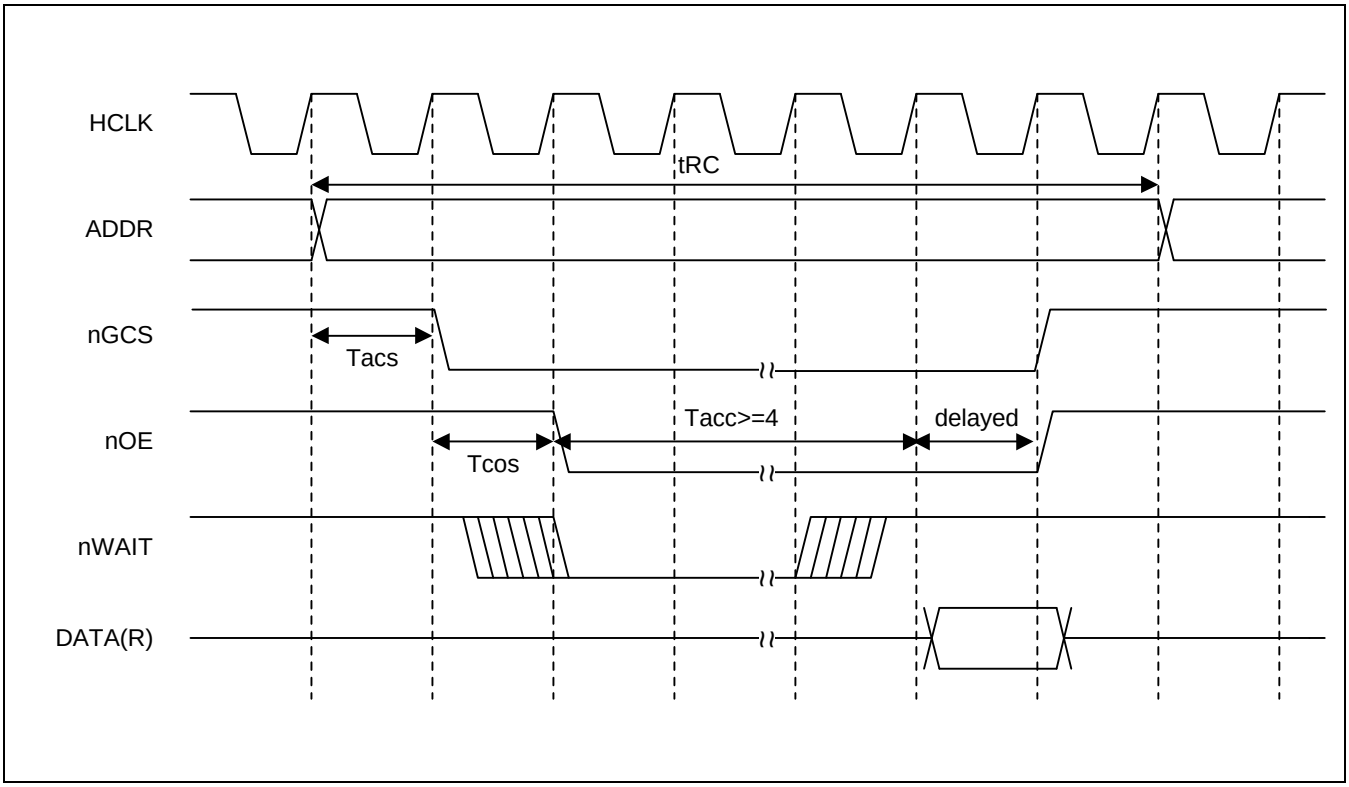


Figure 5-2. S3C2400 External nWAIT Timing Diagram (Tacc≥4)

🗁 **INTERRUPT CONTROLLER: 14-13 : NOTICE : 12/04/2001**
Notice is added as follows

Although the INTMASK is masking some interrupt source, the interrupt can be executed in the case that the corresponding interrupt has been set in the INTPND register

🗁 **ELELCTRICAL DATA: 23-13~23-14 : Figure 23-13, 14 : 12/04/2001**

Timing diagrams of Figure 23-13 and 14 were modified as follows.

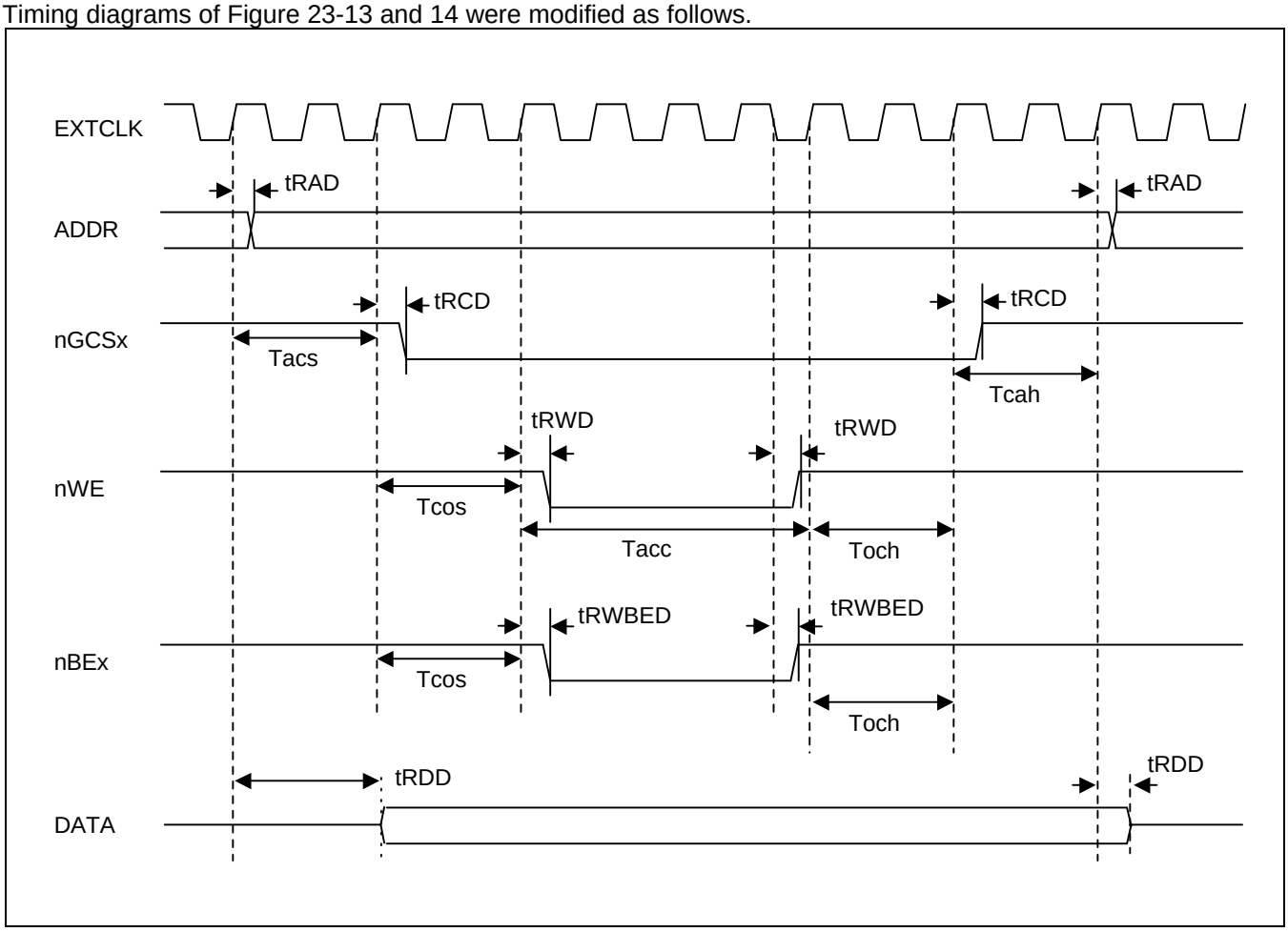


Figure 23-13. ROM/SRAM WRITE Timing (I)
(Tacs=2,Tcos=2,Tacc=4,Toch=2, Tcah=2, PMC=0, ST=0)

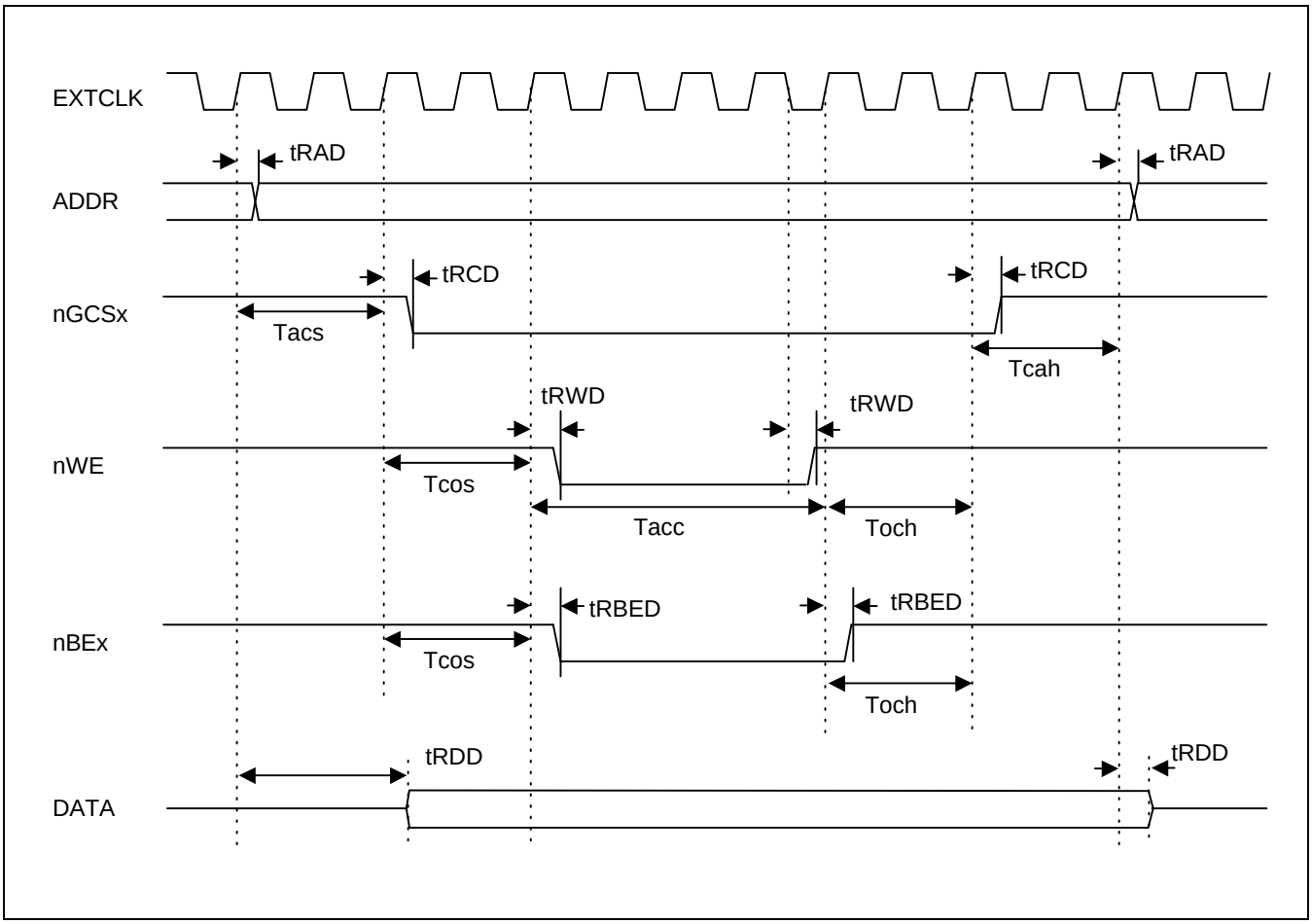


Figure 23-14. ROM/SRAM WRITE Timing (II)
(Tacs=2, Tcos=2, Tacc=4, Toch=2, Tcah=2, PMC=0, ST=1)

🗁 **LCD CONTROLLER : 15-4 : 1st line from the bottom : 12/05/2001**
A equation is modified as follow.

$$frame_rate(Hz) = 1 / [\{(1/VCLK) \times (HOZVAL+1)+(1/HCLK) \times (\mathbf{A}+\mathbf{B}+(LINEBLANK \times 8)) \} \times (LINEVAL+1)]$$

$$\mathbf{A} = 2^{(4+WLH)}, \mathbf{B} = 2^{(4+WDLY)}$$

🗁 **LCD CONTROLLER : 15-25 : WDLY (STN) bit description of LCDCON3: 12/05/2001**
A description is modified as follow.

00 = 16 HCLK, 01 = 32 HCLK, 10 = 64 HCLK, 11 = 128 HCLK

🗁 **LCD CONTROLLER : 15-26 : WLH (STN) bit description of LCDCON4: 12/05/2001**
A description is modified as follow.

00 = 16 HCLK, 01 = 32 HCLK, 10 = 64 HCLK, 11 = 128 HCLK

🗁 **LCD CONTROLLER : 15-27 : Bit 16 description of LCDCON5: 12/05/2001**
A description is modified as follow.

Reserved	[16]	This bit is reserved and the value should be ‘0’.	0

🗁 **ELECTRICAL DATA : TABLE 23-3 ~ 23-4: 01/08/2002**

Table 23-3 is modified as follows and Figure 23-4 is added as follows.

Table 23-3. 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					V
	LVCMOS interface		2.0			
V_{IL}	Low level input voltage					V
	LVCMOS interface				0.8	
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			
I_{IH}	High level input current					
	Input buffer	$V_{IN} = V_{DD}$	-10		10	µA
	Input buffer with pull-up		10	33	60	
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 B8	I_{OH}= -8 mA	2.4			V
	Type B12	I_{OH} = -12 mA				
V_{OL}	Low level output voltage					
	Type B8	I_{OL} = 8 mA			0.4	V
	Type B12	I_{OL} = 12 mA				
I_{DS}	Stop current	$V_{IN} = V_{SS}$ or V_{DD}		50		µA @25 °C
I_{DD}	Operating current				0.5	mA/MHz

NOTE: Type B8 means 8mA output driver cell, and type B12 means 12mA output driver cells.

Table 23-4. DC Electrical Characteristics

(TA = 0 to 70 °C)

Item	Symbol	Min	Typ	Max	Unit	Remarks
Normal operation	I_{DDCPU}	–	66.5*	80	mA	* : 133MHz
Idle mode			26.3*	32		Both oscillators running, CPU static, LCD refresh active
** Slow mode(@1MHz)	I_{TOTAL}		5	–		** : Total current consumption. (CPU+I/O)
** SL-Idle mode(@12MHz)			10			
** Stop mode			50		uA (@25 °C)	Just running 32KHz oscillator(for RTC), all other I/O static.
RTC consumption	I_{RTC}		3	5	uA	x-tal = 32.768KHz for RTC

🗁 **SPI INTERFACE : Figure 22-2: 01/08/2002**
Figure 22-2 is modified as follows.

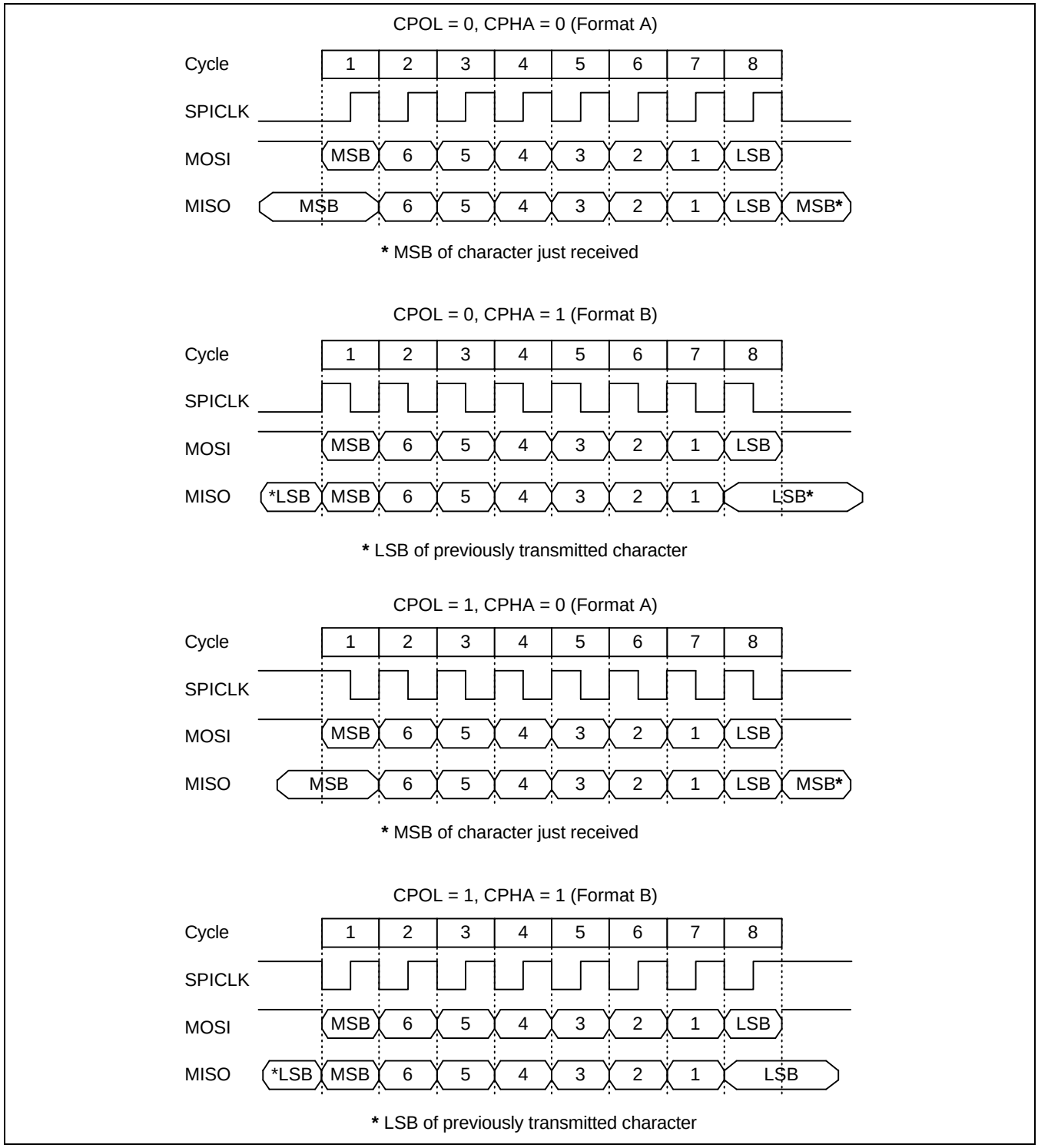


Figure 22-2. SPI Transfer Format

🗁 **PRODUCT OVERVIEW : TABLE 1-3 : 02/14/2002**

Pin# K17 of Table 1-3 is modified as follows.

Table 1-3. 208-Pin FBGA (15×15, 0.8mm Pitch) Pin Assignment (Continued)

Pin #	Pin Name	Pin #	Pin Name
E1	DATA24/GPB8	H1	CLKOUT/GPF6
E2	DATA22/GPB6/nRTS1	H2	VSSIO
E3	VSSIO	H3	VSSi
E4	DATA20/GPB4/RXD1	H4	DATA31/GPB15
E14	nGCS3/GPA15	H14	XTOpll
E15	nGCS1/GPA13	H15	VDDi
E16	VSSIO	H16	VDDIO
E17	nGCS0	H17	VSSi
F1	DATA28/GPB12	J1	TDI
F2	DATA25/GPB9/I2SSDI	J2	nTRST
F3	DATA26/GPB10/nSS	J3	TMS
F4	DATA23/GPB7	J4	VDDi
F14	nGCS7:nSCS1:nRAS1	J14	nRESET
F15	nGCS4/GPA16	J15	VSSIO
F16	nGCS5/GPA17	J16	EXTCLK
F17	nGCS2/GPA14	J17	XTIpll
G1	VDDIO	K1	TOUT2/GPD7
G2	DATA29/GPB13	K2	TDO
G3	DATA30/GPB14	K3	TOUT0/GPD5
G4	DATA27/GPB11	K4	TCK
G14	SCLK	K14	EINT5/GPE5/TCLK1
G15	nWAIT/GPD10	K15	N. C.
G16	SCKE/GPA10	K16	EINT7/GPE7
G17	nGCS6:nSCS0:nRAS0	K17	VDDi

🗁 **ELECTRICAL DATA : TABLE 23-1 ~ 3 : 02/20/2002**

Table 23-1 ~ 3 is modified as follows.

Table 23-1. Absolute Maximum Rating

Parameter	Symbol	Rating		Unit
DC Supply Voltage	V_{DDi}	1.8V V_{DD}	2.7	V
	V_{DDIO}	3.3V V_{DD}	3.8	
DC Input Voltage	V_{IN}	3.3V Input buffer	3.8	
DC Output Voltage	V_{OUT}	3.3V Output buffer	3.8	
DC Input Current	I_{IN}	± 200		mA
Storage Temperature	T_{STG}	– 65 to 150		°C

Table 23-2. Recommended Operating Conditions

Parameter	Symbol	Rating		Unit
DC Supply Voltage for Internal	V_{DDi}	1.8V V_{DD}	1.8±0.15	V
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±0.5%	
DC Input Voltage	V_{IN}	3.3V Input buffer	3.3±0.3	
DC Output Voltage	V_{OUT}	3.3V Output buffer	3.3±0.3	
Operating Temperature	T_{OPR}	Commercial	0 to 70	°C

Table 23-3. 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					V
	LVCMOS interface		2.0			
V_{IL}	Low level input voltage					V
	LVCMOS interface				0.8	
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			
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		-110	-66	-40	
V_{OH}	High level output voltage					
	Type B8	I_{OH}= -8 mA	2.4			V
	Type B12	I_{OH} = -12 mA				
V_{OL}	Low level output voltage					
	Type B8	I_{OL} = 8 mA			0.4	V
	Type B12	I_{OL} = 12 mA				

NOTE: Type B8 means 8mA output driver cell, and type B12 means 12mA output driver cells.

🗁 **ELECTRICAL DATA : TABLE 23-4 ~ 5 : 02/20/2002**

Table 23-4 ~ 5 is modified as follows.

Table 23-4. USB DC Electrical Characteristics

Parameter	Symbol	Condition	Min	Max	Unit
High level input voltage	V_{IH}		2.5		V
Low level input voltage	V_{IL}			0.8	V
High level input current	I_{IH}	Vin = 3.3V	-10	10	μA
Low level input current	I_{IL}	Vin = 0.0V	-10	10	μA
Static Output High	V_{OH}	15KΩ to GND	2.8	3.6	V
Static Output Low	V_{OL}	1.5KΩ to 3.6V		0.3	V

Table 23-5. S3C2400X Power Supply Voltage and Current

Parameter	Value	Unit	Condition
Typical V_{DDi} / V_{DDIO}	1.8 / 3.3	V	
Max. Operating frequency (FCLK)	133	MHz	
Max. Operating frequency (HCLK,PCLK)	66	MHz	
Typical normal mode power (4) (V_{DDi} + V_{DDIO})	100.4	mW	(1)
Typical normal mode power (4) (V_{DDi} + V_{DDIO})	111.7	mW	(2)
Typical normal mode power (4) (V_{DDi} + V_{DDIO})	197.6	mW	(3)
Typical idle mode power (4) (V_{DDi} + V_{DDIO})	66.7	mW	HCLK = 66MHz (F:H:P = 1:2:4)
Typical slow mode power (4) (V_{DDi} + V_{DDIO})	33.8	mW	FCLK = 12MHz (F:H:P = 1:2:4)
Maximum stop mode power Typical stop mode power (4)	100 20	uA uA	Just running 32 kHz oscillator (for RTC), all other I/O static.
Maximum RTC power Typical RTC power (4)	5 2	uA uA	x-tal = 32.768 kHz for RTC

NOTES:

1. I/D cache: ON, MMU: ON, Code on SRAM, FCLK:HCLK:PCLK = 133MHz:66MHz:33MHz
2. Note1 + LCD ON (320x240x8bppx140Hz, 256 color STN)
3. Note1 + Download from USB to SDRAM with DMA
4. Room temperature specification.

S3C2400X

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.

S3C2400X 32-Bit RISC Microprocessor
User's Manual, Revision 1.0
Publication Number: 21.0-S3-C2400X-022002

© 2002 Samsung Electronics

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electric or mechanical, by photocopying, recording, or otherwise, without the prior written consent of Samsung Electronics.

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.samsungsemi.com

Printed in the Republic of Korea

Table of Contents

Chapter 1 Product Overview

Introduction....1-1
Features....1-2
Block Diagram....1-5
Pin Assignments....1-6
- Signal Descriptions....1-25
- S3C2400X Special Registers....1-30
- Important Notes about S3C2400X Special Registers....1-41

Chapter 2 Programmer's Model

Overview....2-1
- Processor Operating States....2-1
- Switching State....2-1
- Memory Formats....2-1
- Big-Endian Format....2-2
- Little-Endian Format....2-2
- Instruction Length....2-2
- Operating Modes....2-3
- Registers....2-3
- The Program Status Registers....2-7
- Exceptions....2-10
- Interrupt Latencies....2-15
- Reset....2-15

Table of Contents (Continued)

Chapter 3 ARM Instruction Set

Instruction Set Summay 3-1
Format Summary 3-1
Instruction Summary 3-2
The Condition Field 3-4
Branch and Exchange (BX) 3-5
Instruction Cycle Times 3-5
Assembler Syntax 3-5
Using R15 As An Operand 3-5
Branch and Branch With Link (B, BL) 3-7
The Link Bit 3-7
Instruction Cycle Times 3-7
Assembler Syntax 3-8
Data Processing 3-9
Cpsr Flags 3-11
Shifts 3-12
Immediate Operand Rotates 3-16
Writing to R15 3-16
Using R15 As an Operandy 3-16
TEQ, TST, CMP and CMN Opcodes 3-16
Instruction Cycle Times 3-16
Assembler Syntax 3-17
Examples 3-17
PSR Transfer (MRS, MSR) 3-18
Operand Restrictions 3-18
Reserved Bits 3-20
Examples 3-20
Instruction Cycle Times 3-20
Assembly Syntax 3-21
Examples 3-21
Multiply and Multiply-Accumulate (MUL, MLA) 3-22
CPSR Flags 3-24
Instruction Cycle Times 3-24
Assembler Syntax 3-24
Examples 3-24
Multiply Long and Multiply-Accumulate Long (MULL, MLAL) 3-25
Operand Restrictions 3-26
Cpsr Flags 3-26
Instruction Cycle Times 3-26
Assembler Syntax 3-27
Examples 3-27

Table of Contents (Continued)

Chapter 3 ARM Instruction Set (Continued)

Single Data Transfer (LDR, STR) 3-28
Offsets and Auto-Indexing 3-29
Shifted Register Offset 3-29
Bytes and Words 3-29
Use Of R15 3-31
Example: 3-31
Data Aborts 3-31
Instruction Cycle Times 3-31
Assembler Syntax 3-32
Examples 3-33
Halfword and Signed Data Transfer (LDRH/STRH/LDRSB/LDRSH) 3-34
Offsets and Auto-Indexing 3-35
Halfword Load And Stores 3-36
Use of R15 3-37
Data Aborts 3-37
Instruction Cycle Times 3-37
Assembler Syntax 3-38
Examples 3-39
Block Data Transfer (LDM, STM) 3-40
The Register List 3-40
Addressing Modes 3-41
Address Alignment 3-41
Use of the S Bit 3-43
Use of R15 as the Base 3-43
Inclusion of the Base In the Register List 3-44
Data Aborts 3-44
Instruction Cycle Times 3-44
Assembler Syntax 3-45
Examples 3-46
Single Data Swap (SWP) 3-47
Bytes and Words 3-47
Use of R15 3-48
Data Aborts 3-48
Instruction Cycle Times 3-48
Assembler Syntax 3-48
Software Interrupt (SWI) 3-49
Return from the Supervisor 3-49
Comment Field 3-49
Instruction Cycle Times 3-49
Assembler Syntax 3-50
Coprocessor Data Operations (CDP) 3-51
Coprocessor Instructions 3-51
Instruction Cycle Times 3-52
Examples 3-52

Table of Contents (Continued)

Chapter 3 ARM Instruction Set (Continued)

Coprocessor Data Transfers (LDC, STC) 3-53
The Coprocessor Fields 3-54
Addressing Modes 3-54
Address Alignment 3-54
Data Aborts 3-54
Assembler Syntax 3-55
Examples 3-55
Coprocessor Register Transfers (MRC, MCR) 3-56
The Coprocessor Fields 3-56
Transfers to R15 3-57
Transfers from R15 3-57
Instruction Cycle Times 3-57
Assembler Syntax 3-57
Examples 3-57
Undefined Instruction 3-58
Instruction Cycle Times 3-58
Assembler Syntax 3-58
Instruction Set Examples 3-59
Using The Conditional Instructions 3-59
Pseudo-Random Binary Sequence Generator 3-61
Multiplication by Constant Using the Barrel Shifter 3-61
Loading a Word from an Unknown Alignment 3-63

Chapter 4 THUMB Instruction Set

THUMB Instruction Set Format 4-1
Format Summary 4-2
Opcode Summary 4-3
Format 1: Move Shifted Register 4-5
Operation 4-5
Instruction Cycle Times 4-6
Examples 4-6
Format 2: Add/Subtract 4-7
Operation 4-7
Instruction Cycle Times 4-8
Examples 4-8
Format 3: Move/Compare/Add/Subtract Immediate 4-9
Operations 4-9
Instruction Cycle Times 4-10
Examples 4-10
Format 4: ALU Operations 4-11
Operation 4-11
Instruction Cycle Times 4-12
Examples 4-12

Table of Contents (Continued)

Chapter 4 THUMB Instruction Set (Continued)

Format 5: Hi-Register Operations/Branch Exchange 4-13
Operation 4-13
Instruction Cycle Times 4-14
The Bx Instruction 4-14
Examples 4-15
Using R15 As An Operand 4-15
Format 6: Pc-Relative Load 4-16
Operation 4-16
Instruction Cycle Times 4-17
Examples 4-17
Format 7: Load/Store With Register Offset 4-18
Operation 4-19
Instruction Cycle Times 4-19
Examples 4-19
Format 8: Load/Store Sign-Extended Byte/Halfword 4-20
Operation 4-20
Instruction Cycle Times 4-21
Examples 4-21
Format 9: Load/Store With Immediate Offset 4-22
Operation 4-23
Instruction Cycle Times 4-23
Examples 4-23
Format 10: Load/Store Halfword 4-24
Operation 4-24
Examples 4-25
Format 11: Sp-Relative Load/Store 4-26
Operation 4-26
Instruction Cycle Times 4-27
Examples 4-27
Format 12: Load Address 4-28
Operation 4-28
Instruction Cycle Times 4-29
Examples 4-29
Format 13: Add Offset To Stack Pointer 4-30
Operation 4-30
Instruction Cycle Times 4-30
Examples 4-30
Format 14: Push/Pop Registers 4-31
Operation 4-31
Instruction Cycle Times 4-32
Examples 4-32
Format 15: Multiple Load/Store 4-33
Operation 4-33
Instruction Cycle Times 4-33
Examples 4-33

Table of Contents (Continued)

Chapter 4 THUMB Instruction Set (Continued)

Format 16: Conditional Branch 4-34
Operation 4-34
Instruction Cycle Times 4-35
Examples 4-35
Operation 4-36
Instruction Cycle Times 4-36
Examples 4-36
Format 18: Unconditional Branch 4-37
Operation 4-37
Examples 4-37
Operation 4-38
Instruction Cycle Times 4-39
Examples 4-39
Instruction Set Examples 4-40
Multiplication By a Constant Using Shifts and Adds 4-40
General Purpose Signed Divide 4-41
Division by a Constant 4-43

Chapter 5 Memory Controller

Overview 5-1
Function Description 5-3
Bank0 Bus Width 5-3
Memory (SROM/DRAM/SDRAM) Address Pin Connections 5-3
Sdram Bank Address Pin Connection 5-4
Nwait Pin Operation 5-5
Nxbreq/Nxback Pin Operation 5-6
Programmable Access Cycle 5-12
Bus Width & Wait Control Register (BWSCON) 5-15
Bank Control Register (Bankconn: NGCS6-NGCS7) 5-18
Supported Bank 6/7 Memory Configuration 5-19
Refresh Control Register 5-19
Banksize Register 5-20
Sdram Mode Register Set Register (MRSR) 5-21

Table of Contents (Continued)

Chapter 6 Clock & Power Management

Overview 6-1
Function Description 6-2
Clock Architecture 6-2
Clock Source Selection 6-2
PLL (Phase-Locked-Loop) 6-4
Clock Control Logic 6-6
Power Management 6-10
Clock Generator & Power Management Special Register 6-22
Lock Time Count Register (LOCKTIME) 6-22
PLL Control Register (MPLLCON, UPLLCON) 6-22
PLL Value Selection Guide 6-22
PLL Value Selection Guide 6-23
Clock Control Register (CLKCON) 6-24
Clock Slow Control Register (CLKSLOW) 6-25
Clock Divider Control Register (CLKDIVN) 6-25

Chapter 7 Bus Priorities

Overview 7-1
Bus Priority Map 7-1

Chapter 8 DMA

Overview 8-1
Dma Request Sources 8-2
Dma Operation 8-2
External DMA DREQ/DACK Protocol 8-3
Examples of Possible Cases 8-6
Dma Special Registers 8-7
Dma Initial Source Register (DISRC) 8-7
DMA Initial Destination Register (DIDST) 8-8
DMA Control Register (DCON) 8-9
DMA Status Register (DSTAT) 8-11
DMA Current Source Register (DCSRC) 8-12
Current Destination Register (DCDST) 8-12
DMA Mask Trigger Register (DMASKTRIG) 8-13

Table of Contents (Continued)

Chapter 9 I/O ports

Overview 9-1
Port Control Descriptions 9-6
- Port Configuration Register (PACON-PGCON) 9-6
- Port Data Register (PADAT-PGDAT) 9-6
- Port Pull-Up Register (PBUP-PGUP) 9-6
- Open Drain Control Register 9-6
- Miscellaneous Control Register 9-6
- External Interrupt Control Register 9-6

I/O Port Control Register 9-7
- Port A Control Registers (PACON, PADAT) 9-7
- Port B Control Registers (PBCON, PBDAT, PBUP) 9-8
- Port C Control Registers (PCCON, PCDAT, PCUP) 9-10
- Port D Control Registers (PDCON, PDDAT, PDUP) 9-12
- Port F Control Registers (PFCON, PFDAT, PFUP) 9-15
- Port G Control Registers (PGCON, PGDAT, PGUP) 9-16
- Open Drain Control Register (OPENCR) 9-17
- Extint (External Interrupt Control Register) 9-19

Chapter 10 PWM Timer

Overview 10-1
- Features 10-1

Pwm Timer Operation 10-3
- Prescaler & Divider 10-3
- Basic Timer Operation 10-3
- Auto-Reload & Double Buffering 10-4
- Timer Initialization Using Manual Update Bit and Inverter Bit 10-5
- Example of a Timer Operation 10-6
- Pwm (Pulse Width Modulation) 10-7
- Output Level Control 10-8
- Dead Zone Generator 10-9
- Dma Request Mode 10-10

Pwm Timer Control Registers 10-11
- Timer Configuration Register0 (TCFG0) 10-11
- Timer Configuration Register1 (TCFG1) 10-12
- Timer Control Register (TCON) 10-13
- Timer 0 Count Buffer Register & Compare Buffer Register (TCNTB0, TCMPB0) 10-15
- Timer 0 Count Observation Register (TCNTO0) 10-15
- Timer 1 Count Buffer Register & Compare Buffer Register (TCNTB1, TCMPB1) 10-16
- Timer 1 Count Observation Register (TCNTO1) 10-16
- Timer 2 Count Buffer Register & Compare Buffer Register (TCNTB2, TCMPB2) 10-17
- Timer 2 Count Observation Register (TCNTO2) 10-17
- Timer 3 Count Buffer Register & Compare Buffer Register (TCNTB3, TCMPB3) 10-18
- Timer 3 Count Observation Register (TCNTO3) 10-18
- Timer 4 Count Buffer Register (TCNTB4) 10-19
- Timer 4 Count Observation Register (TCNTO4) 10-19

Table of Contents (Continued)

Chapter 11 UART

Overview 11-1
Features 11-1
Block Diagram 11-2
UART Operation 11-3
UART Special Registers 11-10
UART Line Control Register 11-10
UART Control Register 11-11
UART FIFO Control Register 11-12
UART Modem Control Register 11-12
UART Tx/Rx Status Register 11-13
UART Error Status Register 11-14
UART FIFO Status Register 11-15
UART Modem Status Register 11-16
UART Transmit Buffer Register(Holding Register & FIFO Register) 11-17
UART Receive Buffer Register (Holding Register & FIFO Register) 11-17
UART Baud Rate Division Register 11-18

Chapter 12 USB Host Controller

Overview 12-1
USB Host Controller Special Registers 12-2
OHCI Registers for USB Host Controller 12-2

Table of Contents (Continued)

Chapter 13 USB Device

Overview 13-1
Features 13-1
Usb Device Special Registers 13-3
Function Addres Register (FUNC_ADDR_REG) 13-3
Power Management Register (PWR_REG) 13-4
Interrupt Register (INT_REG) 13-5
Interrupt Mask Register (INT_MASK_REG) 13-6
Frame Number Register (FPAME_NUM_REG) 13-7
Resume Signal Control Register (RESUME_CON_REG) 13-7
End Point0 Control Status Register (EP0_CSR) 13-8
End Point0 Max Packet Register (EP0_MAXP) 13-9
End Point0 Out Write Count Register (EP0_OUT_CNT) 13-9
End Point0 FIFO Read/Write Register (EP0_FIFO) 13-9
End Point In Control Status Register 13-10
End Point In Max Packet Register 13-12
End Point In FIFO Write Register 13-12
End Point Out Control Status Register 13-13
End Point Out Max Packet Register 13-14
End Point Out Write Count Register 13-14
End Point FIFO Read Register 13-15
DMA Interface Control Register (DMA_CON) 13-15
DMA Unit Counter Register (DMA_UNIT) 13-16
DMA FIFO Counter Register (DMA_FIFO) 13-16
Dma Total Transfer Counter Register (DMA_TX) 13-16
Test Mode Control Register (TEST_MODE) 13-17
In Packet Number Control Register (IN_CON_REG) 13-17

Chapter 14 Interrupt Controller

Overview 14-1
Interrupt Controller Operation 14-2
Interrupt Sources 14-3
Interrupt Priority Generating Block 14-4
Interrupt Priority 14-5
Interrupt Controller Special Registers 14-6
Source Pending Register (SRCPND) 14-6
Interrupt Mode Register (INTMOD) 14-8
Interrupt Mask Register (INTMSK) 14-10
Priority Register (PRIORITY) 14-12
Interrupt Pending Register (INTPND) 14-13
Interrupt Offset Register (INTOFFSET) 14-15

Table of Contents (Continued)

Chapter 15 LCD Controller

Overview 15-1
- Features 15-1
- External Interface Signal 15-2
- Block Diagram 15-3

Lcd Controller Operation (STN Case) 15-4
- Timing Generator 15-4
- Video Operation 15-5
- Dithering And Frc (Frame Rate Control) 15-7
- Memory Data Format (STN, BSWP=0) 15-9

Lcd Controller Operation(TFT Case) 15-15
- Video Operation 15-15
- Memory Data Format (TFT) 15-16
- 256 Palette Usage(TFT) 15-19
- Virtual Display (TFT/STN) 15-22
- LCD Controller Special Registers 15-23

Chapter 16 A/D Converter

Overview 16-1
- Features 16-1

A/D Converter Operation 16-2
- Block Diagram 16-2
- Function Descriptions 16-2

A/D Converter Special Registers 16-4
- A/D Converter Control Register (ADCCON) 16-4
- A/D Converter Data Register (ADCDAT) 16-5

Table of Contents (Continued)

Chapter 17 RTC (Real Time Clock)

Overview 17-1
- Feature 17-1
- Real Time Clock Operation 17-2
- Leap Year Generator 17-2
- Read/Write Registers 17-2
- Backup Battery Operation 17-2
- Alarm Function 17-3
- TICK Time Interrupt 17-3
- Round Reset Function 17-3
- 32.768 Khz X-Tal Connection Example 17-3

Real Time Clock Special Registers 17-4
- Real Time Clock Control Register (RTCCON) 17-4
- Tick Time Count Register (TICNT) 17-4
- RTC Alarm Control Register (RTCALM) 17-5
- Alarm Second Data Register (ALMSEC) 17-6
- Alarm Min Data Register (ALMMIN) 17-6
- Alarm Hour Data Register (ALMHOUR) 17-6
- Alarm Day Data Register (ALMDAY) 17-7
- Alarm Mon Data Register (ALMMON) 17-7
- Alarm Year Data Register (ALMYEAR) 17-7
- RTC Round Reset Register (RTCRST) 17-8
- Bcd Second Register (BCDSEC) 17-8
- BCD Minute Register (BCDMIN) 17-8
- BCD Hour Register (BCDHOUR) 17-9
- BCD Day Register (BCDDAY) 17-9
- BCD Month Register (BCDMON) 17-10
- BCD Year Register (BCDYEAR) 17-10

Chapter 18 Watchdog Timer

Overview 18-1
- Features 18-1
- Watchdog Timer Operation 18-2
- WTDAT & WTCNT 18-2
- Consideration Of Debugging Environment 18-2

Watchdog Timer Special Registers 18-3
- Watchdog Timer Control Register (WTCON) 18-3
- Watchdog Timer Data Register (WTDAT) 18-4
- Watchdog Timer Count Register (WTCNT) 18-4

Table of Contents (Continued)

Chapter 19 MMC Interface

Overview 19-1
Features 19-1
Block Diagram 19-2
MMC Operation 19-3
Programming Procedure 19-3
Steps for Transmit by DMA 19-3
Steps for Receive by DMA 19-3
MMC Special Registers 19-4
MMC Control Register (MMCON) 19-4
MMC Command Related Register (MMCRR) 19-5
MMCRR Register Setting According To Multi Media Card Cmd 19-6
MMC Status Register (MMSTA) 19-8
MMC FIFO Status Register (MMFSTA) 19-9
MMC Baud Rate Prescaler Register (MMPRE) 19-9
MMC Block Length Register (MMLEN) 19-9
Response CRC7 Register (MMCR7) 19-10
MMC Response Status Register 0 (MMRSP0) 19-10
MMC Response Status Register 1 (MMRSP1) 19-10
MMC Response Status Register 2 (MMRSP2) 19-11
MMC Response Status Register 3 (MMRSP3) 19-11
MMC Command Register 0 (MMCMD0) 19-11
MMC Command Register 1 (MMCMD1) 19-12
Data Receive CRC16 Buffer Register (MMCR16) 19-12
MMC Data Register (MMDAT) 19-12

Chapter 20 IIC-Bus Interface

Overview 20-1
The IIC-Bus Interface 20-3
Start and Stop Conditions 20-3
Data Transfer Format 20-4
ACK Signal Transmission 20-5
Read-Write Operation 20-6
Bus Arbitration Procedures 20-6
Abort Conditions 20-6
Configuring the IIC-Bus 20-6
Flowcharts of the Operations in Each Mode 20-7
IIC-Bus Interface Special Registers 20-11
Multi-Master IIC-Bus Control Register (IICCON) 20-11
Multi-Master IIC-Bus Address Register (IICADD) 20-13
Multi-Master IIC-Bus Transmit/Receive Data Shift Register (IICDS) 20-13

Table of Contents (Continued)

Chapter 21 IIS-Bus Interface

Overview 21-1
Features 21-1
Block Diagram 21-2
Functional Descriptions 21-2
- Transmit or Receive only Mode 21-3
- DMA Transfer 21-3
- Transmit and Receive Mode 21-3

Audio Serial Interface Format 21-4
- IIS-Bus Format 21-4
- MSB(Left) Justified 21-4
- Sampling Frequency and Master Clock 21-5

IIS-Bus Interface Special Registers 21-6
- IIS Control Register (IISCON) 21-6
- IIS Mode Register (IISMOD) 21-7
- IIS Prescaler Register (IISPSR) 21-8
- IIS FIFO Control Register (IISFCON) 21-8
- IIS FIFO Register (IISFIF) 21-9

Chapter 22 SPI Interface

Overview 22-1
- Features 22-1
- Block Diagram 22-2

SPI Operation 22-3
SPI Special Registers 22-6
- SPI Control Register(SPCON) 22-6
- SPI Status Register(SPSTA) 22-6
- SPI Pin Control Register(SPPIN) 22-7
- SPI Baud Rate Prescaler Register(SPPRE) 22-8
- SPI Tx Data Register(SPTDAT) 22-8
- SPI Rx Data Register(SPRDAT) 22-8

Chapter 23 Electrical Data

Absolute Maximum Ratings 23-1
Recommended Operating Conditions 23-1
D.C. Electrical Characteristics 23-2
A.C. Electrical Characteristics 23-4

Table of Contents (Concluded)

Chapter 24 Mechanical Data

Package Dimension 20-1

Chapter 25 Notice In Programming

Known Problems 25-1
Cautions 25-3
Work-Arounds 25-3

List of Figures

Figure Number	Title	Page Number
1-1	S3C2400X Block Diagram	1-5
1-2	S3C2400X Pin Assignments (208-LQFP)	1-6
1-3	S3C2400X Pin Assignments (208-FBGA, 12×12, 0.65mm Pitch)	1-7
1-4	S3C2400X Pin Assignments (208-FBGA, 15×15, 0.8mm Pitch)	1-8
2-1	Big-Endian Addresses of Bytes within Words	2-2
2-2	Little-Endian Addresses of Bytes within Words	2-2
2-3	Register Organization in ARM State	2-4
2-4	Register Organization in THUMB state	2-5
2-5	Mapping of THUMB State Registers onto ARM State Registers	2-6
2-6	Program Status Register Format	2-7
3-1	ARM Instruction Set Format	3-1
3-2	Branch and Exchange Instructions	3-5
3-3	Branch Instructions	3-7
3-4	Data Processing Instructions	3-9
3-5	ARM Shift Operations	3-12
3-6	Logical Shift Left	3-12
3-7	Logical Shift Right	3-13
3-8	Arithmetic Shift Right	3-13
3-9	Rotate Right	3-14
3-10	Rotate Right Extended	3-14
3-11	PSR Transfer	3-19
3-12	Multiply Instructions	3-22
3-13	Multiply Long Instructions	3-25
3-14	Single Data Transfer Instructions	3-28
3-15	Little-Endian Offset Addressing	3-30
3-16	Halfword and Signed Data Transfer with Register Offset	3-34
3-17	Halfword and Signed Data Transfer with Immediate Offset and Auto-Indexing	3-35
3-18	Block Data Transfer Instructions	3-40
3-19	Post-Increment Addressing	3-41
3-20	Pre-Increment Addressing	3-42
3-21	Post-Decrement Addressing	3-42
3-22	Pre-Decrement Addressing	3-43
3-23	Swap Instruction	3-47
3-24	Software Interrupt Instruction	3-49
3-25	Coprocessor Data Operation Instruction	3-51
3-26	Coprocessor Data Transfer Instructions	3-53
3-27	Coprocessor Register Transfer Instructions	3-56
3-28	Undefined Instruction	3-58

List of Figures (Continued)

Figure Number	Title	Page Number
4-1	THUMB Instruction Set Formats	4-2
4-2	Format 1	4-5
4-3	Format 2	4-7
4-4	Format 3	4-9
4-5	Format 4	4-11
4-6	Format 5	4-13
4-7	Format 6	4-16
4-8	Format 7	4-18
4-9	Format 8	4-20
4-10	Format 9	4-22
4-11	Format 10	4-24
4-12	Format 11	4-26
4-13	Format 12	4-28
4-14	Format 13	4-30
4-15	Format 14	4-31
4-16	Format 15	4-33
4-17	Format 16	4-34
4-18	Format 17	4-36
4-19	Format 18	4-37
4-20	Format 19	4-38
5-1	S3C2400X Memory Map after Reset	5-2
5-2	S3C2400X External nWAIT Timing Diagram (Tacc≥4)	5-5
5-3	S3C2400X nXBREQ/nXBACK Timing Diagram	5-6
5-4	Memory Interface with 8bit ROM	5-7
5-5	Memory Interface with 8bit ROM x 2	5-7
5-6	Memory Interface with 8bit ROM x 4	5-8
5-7	Memory Interface with 16bit ROM	5-8
5-8	Memory Interface with 16bit SRAM	5-9
5-9	Memory Interface with 16bit SRAM x 2	5-9
5-10	Memory Interface with 16bit DRAM	5-10
5-11	Memory Interface with 16bit DRAM x 2	5-10
5-12	Memory Interface with 16bit SDRAM (4Mx16, 4bank)	5-11
5-13	Memory Interface with 16bit SDRAM (4Mx16 * 2ea, 4bank)	5-11
5-14	S3C2400X nGCS Timing Diagram	5-12
5-15	S3C2400X DRAM Timing Diagram	5-13
5-16	S3C2400X DRAM Refresh Timing Diagram	5-13
5-17	S3C2400X SDRAM Timing Diagram	5-14

List of Figures (Continued)

Figure Number	Title	Page Number
6-1	Clock Generator Block Diagram	6-3
6-2	PLL (Phase-Locked Loop) Block Diagram	6-5
6-3	Main Oscillator Circuit Examples	6-5
6-4	Power-On Reset Sequence (When the external clock source is a crystal oscillator)	6-6
6-5	Timing Diagram of Changing FCLK	6-7
6-6	Timing Diagram of Changing PMS without the Interference of LCD Display	6-8
6-7	Example that changes CLKDIVN register value	6-9
6-8	The Clock Distribution Block Diagram	6-11
6-9	Power Management State Machine	6-12
6-10	Entering STOP Mode and Exiting STOP mode (Wake-up)	6-14
6-11	Entering SL_IDLE Mode and Exiting from SL_IDLE Mode (Wake-up)	6-16
6-12	The case that Exit_from_Slow_mode command is issued in PLL on state	6-18
6-13	The Case that Exit_from_Slow_Mode Command is Issued after Lock Time is End	6-19
6-14	The Case that the Exit_from_Slow_Mode Command and the Instant PLL_on Command is Issued Aimultaneously	6-19
8-1	Basic DMA Timing Diagram	8-3
8-2	Demand/Handshake Mode Comparison	8-4
8-3	Burst 4 Transfer Size	8-5
8-4	Single service, Demand Mode, Single Transfer Size	8-6
8-5	Single Service, Handshake Mode, Single Transfer Size	8-6
8-6	Whole Service, Handshake Mode, Single Transfer Size	8-6
10-1	16-bit PWM Timer Block Diagram	10-2
10-2	Timer Operations	10-3
10-3	Example of Double Buffering Feature	10-4
10-4	Example of a Timer Operation	10-6
10-5	Example of PWM	10-7
10-6	Inverter On/Off	10-8
10-7	The Wave Form When a Dead Zone Feature is Enabled	10-9
10-8	The Timer 4 DMA Mode Operation	10-10
11-1	UART Block Diagram (with FIFO)	11-2
11-2	UART AFC interface	11-3
11-3	A Case showing UART Receiving 5 Characters with 2 Errors	11-6
11-4	IrDA Function Block Diagram	11-8
11-5	Serial I/O Frame Timing Diagram (Normal UART)	11-9
11-6	Infra-Red Transmit Mode Frame Timing Diagram	11-9
11-7	Infra-Red Receive Mode Frame Timing Diagram	11-9
11-8	nCTS and Delta CTS Timing Diagram	11-16
12-1	USB Host Controller Block Diagram	12-1
13-1	USB Device Block Diagram	13-2

List of Figures (Continued)

Figure Number	Title	Page Number
14-1	Interrupt Process Diagram	14-1
14-2	Priority Generating Block	14-4
15-1	LCD Controller Block Diagram	15-3
15-2	Monochrome Display Types (STN)	15-12
15-3	Color Display Types (STN)	15-13
15-4	8-bit Single Scan Display Type STN LCD Timing	15-14
15-5	BPP16 5:6:5 Display Types (TFT)	15-20
15-6	TFT LCD Timing Example	15-21
15-7	Example of Scrolling in Virtual Display (Single Scan)	15-22
16-1	A/D Converter Functional Block Diagram	16-2
17-1	Real Time Clock Block Diagram	17-2
17-2	Main Oscillator Circuit Examples	17-3
18-1	Watch-Dog Timer Block Diagram	18-2
19-1	MMC Block Diagram	19-2
20-1	IIC-Bus Block Diagram	20-2
20-2	Start and Stop Condition	20-3
20-3	IIC-Bus Interface Data Format	20-4
20-4	Data Transfer on the IIC-Bus	20-5
20-5	Acknowledge on the IIC-Bus	20-5
20-6	Operations for Master/Transmitter Mode	20-7
20-7	Operations for Master/Receiver Mode	20-8
20-8	Operations for Slave/Transmitter Mode	20-9
20-9	Operations for Slave / Receiver Mode	20-10
21-1	IIS-Bus Block Diagram	21-2
21-2	IIS-Bus and MSB(Left)-justified Data Interface Formats	21-5
22-1	SPI Block Diagram	22-2

List of Figures (Continued)

Figure Number	Title	Page Number
23-1	XTIpll Clock Timing	23-4
23-2	EXTCLK Clock Input Timing	23-4
23-3	EXTCLK/HCLK in case that EXTCLK is used without the PLL	23-5
23-4	HCLK/CLKOUT/SCLK in case that EXTCLK is used	23-5
23-5	Manual Reset and OM[3:0] Input Timing	23-5
23-6	Power-On Oscillation Setting Timing	23-6
23-7	STOP Mode Return Oscillation Setting Timing	23-7
23-8	ROM/SRAM Burst READ Timing(I) (Tacs=0, Tcos=0, Tacc=2, Toch=0, Tcah=0, PMC=0, ST=0, DW=16bit)	23-8
23-9	ROM/SRAM Burst READ Timing(II) (Tacs=0, Tcos=0, Tacc=2, Toch=0, Tcah=0, PMC=0, ST=1, DW=16bit)	23-9
23-10	External Bus Request in ROM/SRAM Cycle (Tacs=0, Tcos=0, Tacc=8, Toch=0, Tcah=0, PMC=0, ST=0)	23-10
23-11	ROM/SRAM READ Timing (I) (Tacs=2,Tcos=2, Tacc=4, Toch=2, Tcah=2, PMC=0, ST=0)	23-11
23-12	ROM/SRAM READ Timing (II) (Tacs=2, Tcos=2, Tacc=4, Toch=2, Tcah=2cycle, PMC=0, ST=1)	23-12
23-13	ROM/SRAM WRITE Timing (I) (Tacs=2,Tcos=2,Tacc=4,Toch=2, Tcah=2, PMC=0, ST=0)	23-13
23-14	ROM/SRAM WRITE Timing (II) (Tacs=2, Tcos=2, Tacc=4, Toch=2, Tcah=2, PMC=0, ST=1)	23-14
23-15	External nWAIT READ Timing (Tacs=0, Tcos=0, Tacc=6, Toch=0, Tcah=0, PMC=0, ST=0)	23-15
23-16	External nWAIT WRITE Timing (Tacs=0, Tcos=0, Tacc=4, Toch=0, Tcah=0, PMC=0, ST=0)	23-16
23-17	DRAM (EDO) Burst READ Timing (Trcd=2, Tcas=1, Tcp=1, Trp=3.5, MT=10, DW = 16bit)	23-17
23-18	External Bus Request in DRAM Cycle (Trcd=3, Tcas=2, Tcp=1, Trp=4.5)	23-18
23-19	DRAM(FP) Single READ Timing (Trcd=3, Tcas=2, Tcp=1, Trp=4.5, MT=01)	23-19
23-20	DRAM(EDO) Single READ Timing (Trcd=3, Tcas=2, Tcp=1, Trp=4.5, MT=10)	23-20
23-21	DRAM CBR Refresh Timing (Tchr=4)	23-20
23-22	DRAM(EDO) Page Hit-Miss READ Timing (Trcd=2, Tcas=2, Tcp=1, Trp=3.5, MT=10)	23-21
23-23	DRAM Self Refresh Timing	23-22
23-24	DRAM(FP/EDO) Single Write Timing (Trcd=3, Tcas=2, Tcp=1, Trp=4.5, MT=01/10)	23-22
23-25	DRAM(FP/EDO) Page Hit-Miss Write Timing (Trcd=2, Tcas=2, Tcp=1, Trp=3.5, MT=01/10)	23-23
23-26	Masked-ROM Single READ Timing (Tacs=2, Tcos=2, Tacc=8, PMC=01/10/11)	23-24
23-27	Masked-ROM Consecutive READ Timing (Tacs=0, Tcos=0, Tacc=3, Tpac=2, PMC=01/10/11)	23-24

List of Figures (Continued)

Figure Number	Title	Page Number
23-28	SDRAM Single Burst READ Timing (Trp=2, Trcd=2, Tcl=2, DW=16bit)	23-25
23-29	External Bus Request in SDRAM Timing (Trp=2, Trcd=2, Tcl=2)	23-26
23-30	SDRAM MRS Timing	23-27
23-31	SDRAM Single READ Timing(I) (Trp=2, Trcd=2, Tcl=2)	23-28
23-32	SDRAM Single READ Timing(II) (Trp=2, Trcd=2, Tcl=3)	23-29
23-33	SDRAM Auto Refresh Timing (Trp=2, Trc=4)	23-30
23-34	SDRAM Page Hit-Miss READ Timing (Trp=2, Trcd=2, Tcl=2)	23-31
23-35	SDRAM Self Refresh Timing (Trp=2, Trc=4)	23-32
23-36	SDRAM Single Write Timing (Trp=2, Trcd=2)	23-33
23-37	SDRAM Page Hit-Miss Write Timing (Trp=2, Trcd=2, Tcl=2)	23-34
23-38	External DMA Timing (Handshake, Single transfer)	23-35
23-39	STN LCD Controller Timing	23-35
23-40	TFT LCD Controller Timing	23-36
23-41	MMC Interface Receive Timing	23-37
23-42	SPI Interface Receive Timing	23-37
24-1	208-LQFP- 2828 Package Dimension	24-1
24-2	208-FBGA-12.0X12.0 Package Dimension 1	24-2
24-3	208-FBGA-12.0X12.0 Package Dimension 2	24-3
24-4	208-FBGA-15.0X15.0 Package Dimension 1	24-4
24-5	208-FBGA-15.0X15.0 Package Dimension 2	24-5

List of Tables

Table Number	Title	Page Number
1-1	208-Pin LQFP Pin Assignment	1-9
1-2	208-Pin FBGA (12×12, 0.65mm Pitch) Pin Assignment	1-17
1-3	208-Pin FBGA (15×15, 0.8mm Pitch) Pin Assignment	1-21
1-4	S3C2400X Signal Descriptions	1-25
1-5	S3C2400X Special Registers	1-30
2-1	PSR Mode Bit Values	2-9
2-2	Exception Entry/Exit	2-11
2-3	Exception Vectors	2-13
3-1	The ARM Instruction Set	3-2
3-2	Condition Code Summary	3-4
3-3	ARM Data Processing Instructions	3-11
3-4	Incremental Cycle Times	3-16
3-5	Assembler Syntax Descriptions	3-27
3-6	Addressing Mode Names	3-45
4-1	THUMB Instruction Set Opcodes	4-3
4-2	Summary of Format 1 Instructions	4-5
4-3	Summary of Format 2 Instructions	4-7
4-4	Summary of Format 3 Instructions	4-9
4-5	Summary of Format 4 Instructions	4-11
4-6	Summary of Format 5 Instructions	4-13
4-7	Summary of PC-Relative Load Instruction	4-16
4-8	Summary of Format 7 Instructions	4-19
4-9	Summary of format 8 instructions	4-20
4-10	Summary of Format 9 Instructions	4-23
4-11	Halfword Data Transfer Instructions	4-24
4-12	SP-Relative Load/Store Instructions	4-26
4-13	Load Address	4-28
4-14	The ADD SP Instruction	4-30
4-15	PUSH and POP Instructions	4-31
4-16	The Multiple Load/Store Instructions	4-33
4-17	The Conditional Branch Instructions	4-34
4-18	The SWI Instruction	4-36
4-19	Summary of Branch Instruction	4-37
4-20	The BL Instruction	4-39
5-1	Bank 6/7 Address	5-2
5-2	SDRAM Bank Address configuration	5-4
6-1	Clock source selection at boot-up	6-2
6-2	Functional Block Clock State In Each Power Mode	6-12
6-3	CLKSLOW and CLKDIVN Register Settings for SLOW Clock	6-17
6-4	The Status of PLL and FCLK After Wake-Up	6-20

List of Tables (Continued)

Table Number	Title	Page Number
8-1	DMA request sources for each channel	8-2
9-1	S3C2400X Port Configuration Overview	9-2
15-1	Relation between VCLK and CLKVAL(STN, HCLK=60MHz)	15-5
15-2	Dither Duty Cycle Examples	15-7
15-3	Relation between VCLK and CLKVAL(TFT, HCLK=60MHz)	15-15
15-4	5:6:5 Format	15-19
15-5	5:5:5:1 Format	15-19
15-6	MV Value for Each Display Mode	15-34
21-1	CODEC clock (CODECLK = 256 or 384fs)	21-5
21-2	Usable serial bit clock frequency (IISCLK = 16 or 32 or 48fs)	21-5
23-1	Absolute Maximum Rating	23-1
23-2	Recommended Operating Conditions	23-1
23-3	Normal I/O PAD DC Electrical Characteristics	23-2
23-4	USB DC Electrical Characteristics	23-3
23-5	S3C2400X Power Supply Voltage and Current	23-3
23-6	Clock Timing Constants	23-38
23-7	ROM/SRAM Bus Timing Constants	23-39
23-8	Clock Timing Constants	23-40
23-9	Memory Interface Timing Constants	23-40
23-10	External Bus Request Timing Constants	23-41
23-11	DMA Controller Module Signal Timing Constants	23-41
23-12	STN LCD Controller Module Signal Timing Constants	23-42
23-13	TFT LCD Controller Module Signal Timing Constants	23-42
23-14	IIS Controller Module Signal Timing Constants	23-43
23-15	IIC BUS Controller Module Signal Timing	23-43
23-16	MMC Interface Transmit/Receive Timing Constants	23-44
23-17	SPI Interface Transmit/Receive Timing Constants	23-44
23-18	USB Electrical Specifications	23-45
23-19	USB Full Speed Output Buffer Electrical Characteristics	23-45
23-20	USB Low Speed Output Buffer Electrical Characteristics	23-46

1 PRODUCT OVERVIEW

INTRODUCTION

SAMSUNG's S3C2400X 16/32-bit RISC microprocessor is designed to provide a cost-effective, low power, small die size and high performance micro-controller solution for hand-held devices and general applications. To reduce total system cost, S3C2400X also provides the following: separate 16KB Instruction and 16KB Data Cache, MMU to handle virtual memory management, LCD controller (STN & TFT), 2-channel UART with handshake, 4-channel DMA, System Manager (chip select logic, EDO/SDRAM controller), 4-channel Timers with PWM, I/O Ports, RTC, 8-channel 10-bit ADC, IIC-BUS interface, IIS-BUS interface, USB Host, USB Device, Multi-Media Card Interface, SPI and PLL for clock generation.

The S3C2400X 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-sensitive and power sensitive applications. Also S3C2400X adopts a new bus architecture, AMBA (Advanced Microcontroller Bus Architecture)

An outstanding feature of the S3C2400X is 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 a 8-word line length.

By providing complete set of common system peripherals, the S3C2400X 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 internal, 3.3V external (I/O boundary) microprocessor with 16KB I-Cache, 16KB D-Cache, and MMU.
- External memory controller. (EDO/SDRAM Control, Chip Select logic)
- LCD controller (up to 4K color STN and 64K color TFT) with 1-ch LCD-dedicated DMA.
- 4-ch DMAs with external request pins
- 2-ch UART with handshake (IrDA1.0, 16-byte FIFO)/1-ch SPI
- 1-ch multi-master IIC-BUS/1-ch IIS-BUS controller
- MMC interface (ver 2.11)
- 2-port USB Host /1- port USB Device (ver 1.1)
- 4-ch PWM timers & 1-ch internal timer
- Watchdog Timer
- 90-bit general purpose I/O ports/8-ch external interrupt source
- Power control: Normal, Slow, Idle, Stop and SL_IDLE mode
- 8-ch 10-bit ADC.
- 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 AMBA(Advanced Microcontroller Bus Architecture) (AMBA2.0, AHB/APB)

System Manager

- Little/Big Endian support.
- Address space: 32M bytes for each bank (Total 256Mbyte)
- Supports programmable 8/16/32-bit data bus width for each bank.
- Fixed bank start address and programmable bank size for 7 banks.
- Programmable bank start address and bank size for one bank.
- 8 memory banks.
  — 6 memory banks for ROM, SRAM etc.
  — 2 memory banks for ROM/SRAM/DRAM(EDO or Synchronous DRAM)
- Fully Programmable access cycles for all memory banks.
- Supports external wait signal to expend the bus cycle.
- Supports self-refresh mode in DRAM/SDRAM for power-down.
- Supports asymmetric/symmetric address of DRAM.

Cache Memory

- 64 way set-associative cache with I-Cache(16KB) and D-Cache(16KB).
- 8-words 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 address.

Clock & Power Manager

- Low power
- The on-chip MPLL and UPLL
  UPLL makes the clock for operating USB Host/Device.
  MPLL makes the clock for operating MCU at maximum 133Mhz @ 1.8V.
- Clock can be fed selectively to each function block by software.
- Power mode: Normal, Slow, Idle, Stop mode and SL_IDLE mode.
  Normal mode: Normal operating mode.
  Slow mode: Low frequency clock without PLL.
  Idle mode: Stop the clock for only CPU.
  Stop mode: All clocks are stopped.
  SL_IDLE mode: All clocks except LCD are stopped.
- Wake up by EINT[7:0] or RTC alarm interrupt from Stop mode.

Interrupt Controller

- 32 Interrupt sources
  (Watch dog timer, 5Timer, 6UART, 8External interrupts, 4 DMA, 2 RTC, 1 ADC, 1 IIC, 1 SPI, 1 MMC, 2 USB)
- Level/Edge mode on external interrupt source.
- Programmable polarity of edge and level.
- Supports FIQ (Fast Interrupt request) for very urgent interrupt request.

Timer with PWM (Pulse Width Modulation)

- 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 source.

RTC (Real Time Clock)

- Full clock feature: msec, sec, min, hour, day, week, month, year.
- 32.768 KHz operation.
- Alarm interrupt.
- Time tick interrupt

General Purpose Input/Output Ports

- 8 external interrupt ports
- 90 multiplexed input/output ports

UART

- 2-channel UART with DMA-based or interrupt-based operation
- Supports 5-bit, 6-bit, 7-bit, or 8-bit serial data transmit/receive
- Supports H/W handshaking during transmit/receive
- Programmable baud rate
- Supports IrDA 1.0
- Loop back mode for testing
- Each channel has internal 16-byte Tx FIFO and 16-byte Rx FIFO.

DMA Controller

- 4-ch DMA controller.
- Support memory to memory, IO to memory, memory to IO, IO to IO
- Burst transfer mode to enhance the transfer rate.

A/D Converter

- 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 the monochrome, 4 gray levels, 16gray levels, 256 color and 4096 colors for STN LCD.
- Supports multiple screen size
  - — Typical actual screen size: 320x240, 160x160 (pixels)
  - — Maximum virtual screen size (color mode): 4096x1024, 2048x2048, 1024x4096 etc.
- Supports power saving mode(Enhanced SL_IDLE mode.)

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 32K (64K using intensity) color TFT at 16 bpp mode.
- Supports multiple screen size
  - — Typical actual screen size: 720x240, 320x240, 160x160 (pixels)
  - — Maximum virtual screen size (16bpp mode): 2048x1024 etc

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 the standard mode or up to 400 Kbit/s in the fast mode.

IIS-BUS Interface

- 1-ch IIS-bus for audio interface with DMA-based operation.
- Serial, 8/16bit per channel data transfers.
- Supports IIS format and MSB-justified data format.

USB Host

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

USB Device

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

MMC Interface

- Multi-Media Card Protocol version 2.11 compatible
- 2x16 Bytes FIFO for receive/transmit.
- DMA-based or interrupt-based operation.

SPI Interface

- Serial Peripheral Interface Protocol version 2.11 compatible
- 2x8 bits Shift register for receive/transmit.
- DMA-based or interrupt-based operation.

Operating Voltage Range

- Core: 1.8V
- I/O: 3.3V

Operating Frequency

- Up to 133 MHz

Package

- 208 LQFP/208 FBGA

BLOCK DIAGRAM

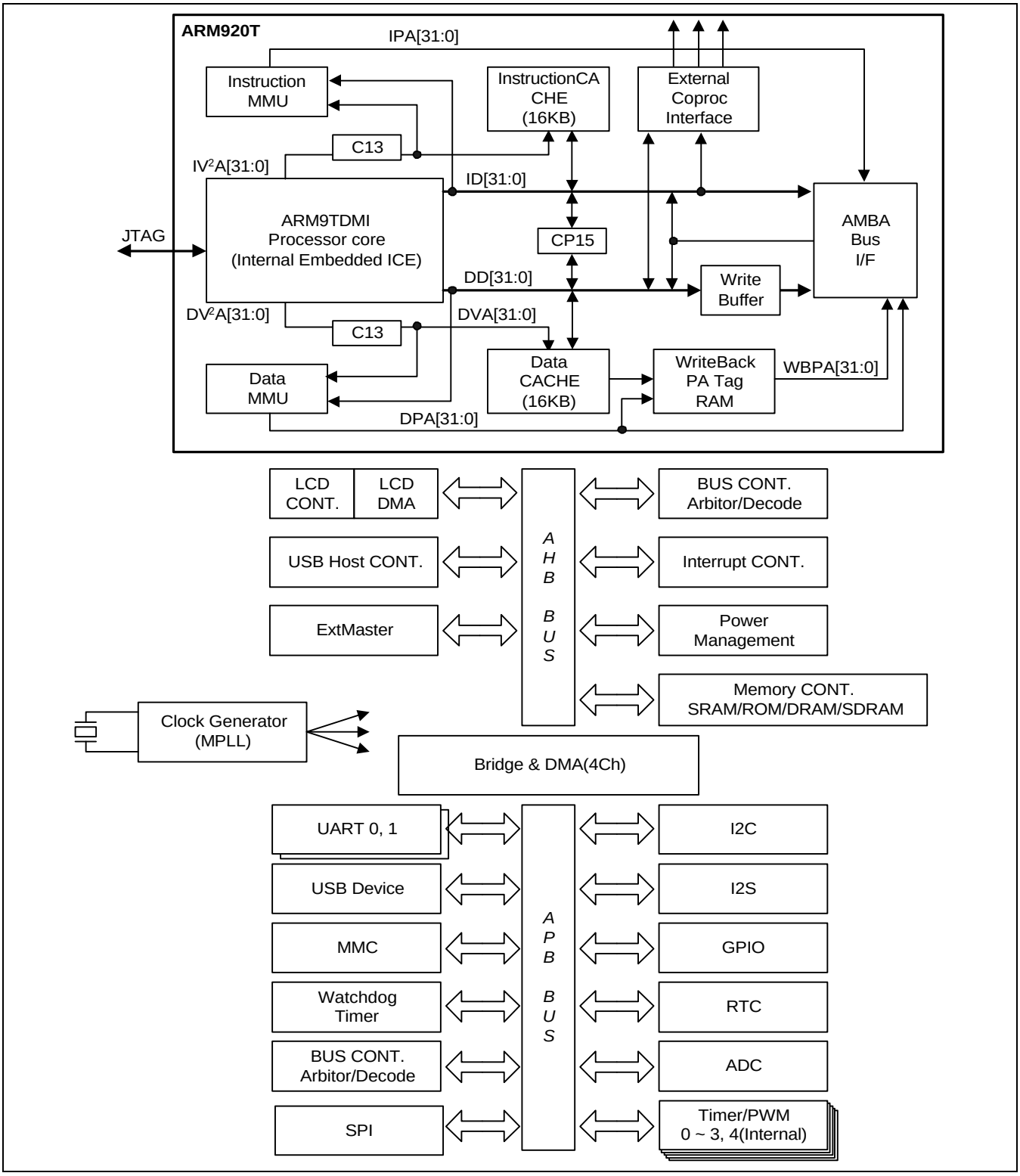


Figure 1-1. S3C2400X Block Diagram

PIN ASSIGNMENTS

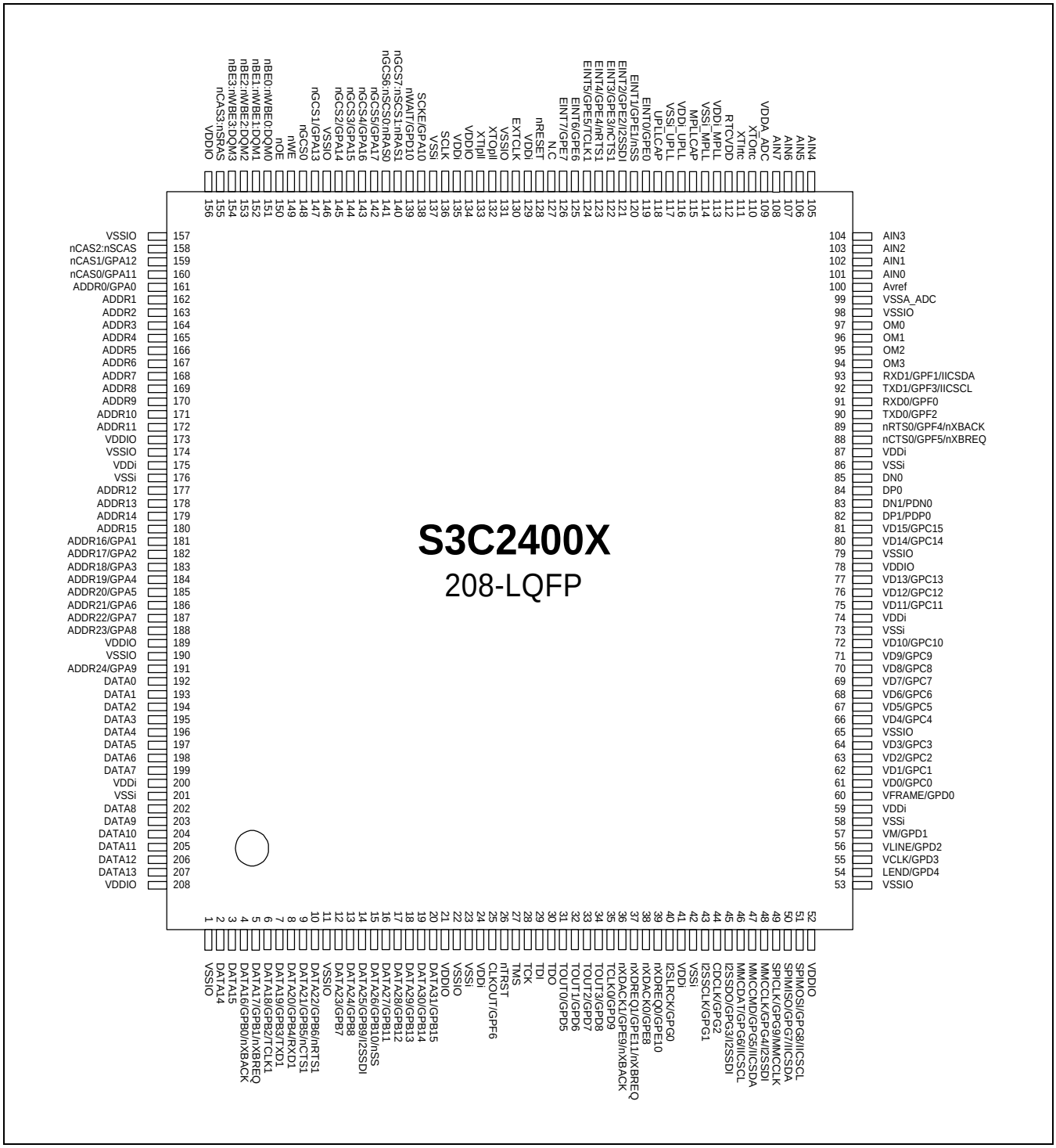


Figure 1-2. S3C2400X Pin Assignments (208-LQFP)

SAMSUNG ELECTRONICS

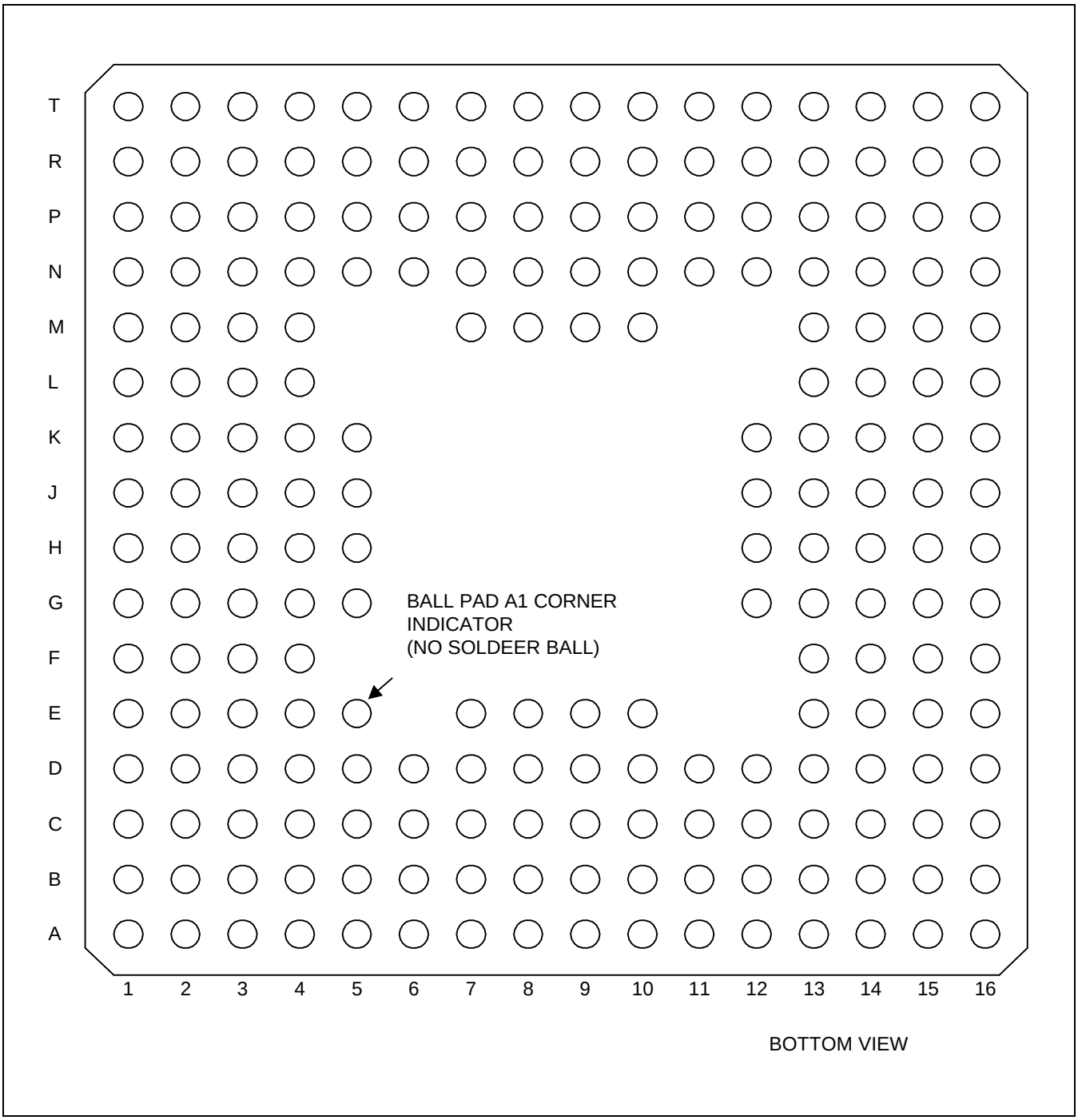


Figure 1-3. S3C2400X Pin Assignments (208-FBGA, 12×12, 0.65mm Pitch)

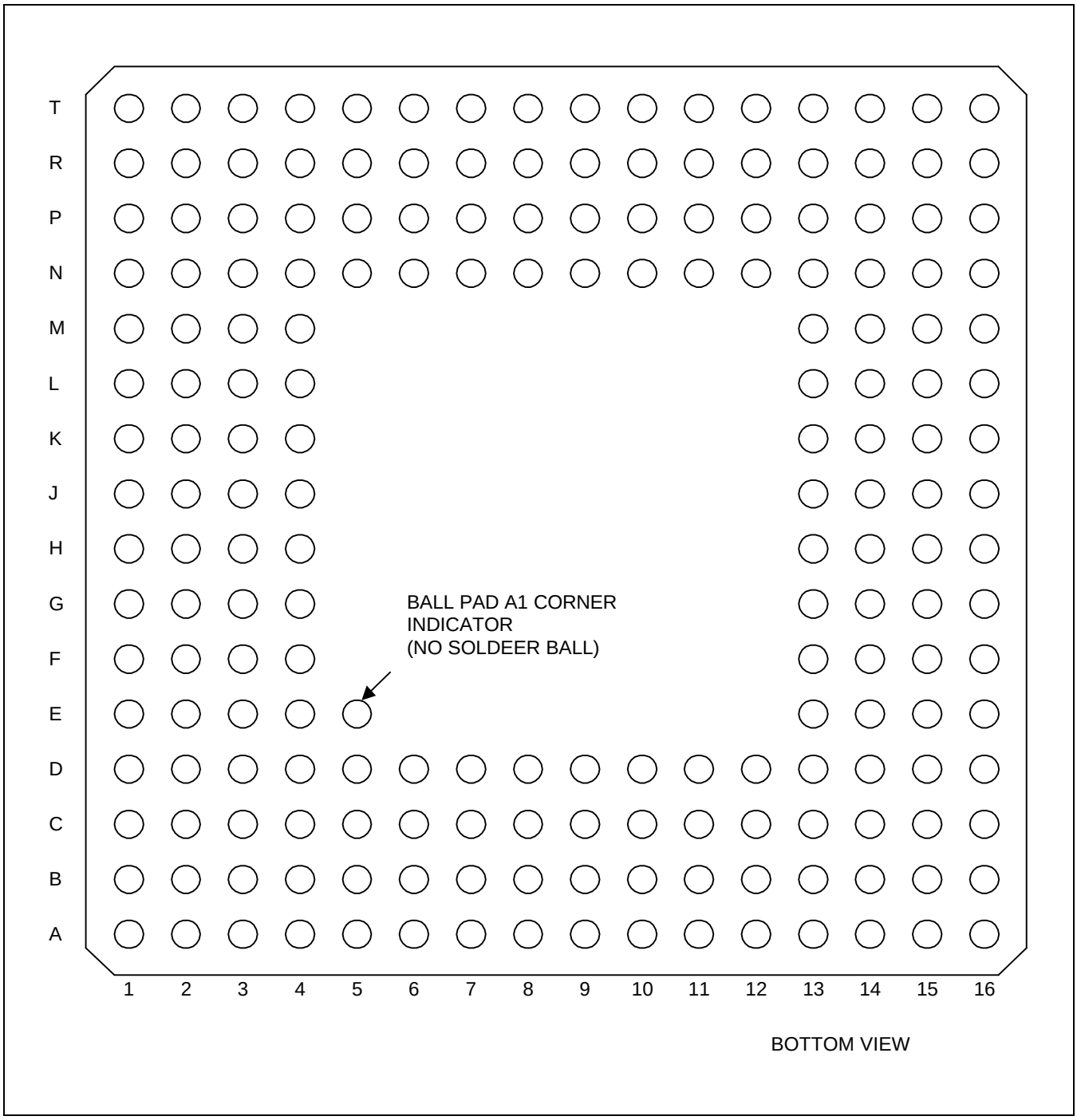


Figure 1-4. S3C2400X Pin Assignments (208-FBGA, 15×15, 0.8mm Pitch)

Table 1-1. 208-Pin LQFP Pin Assignment

Pin Number	Pin Name	Default Function	I/O State @BUS REQ.	I/O State @STOP	I/O State @nRESET	I/O Type
1	VSSIO	VSSIO	–	–	P	vss3op
2	DATA14	DATA14	Hi-z	Hi-z	I	phbsu50ct12sm
3	DATA15	DATA15	Hi-z	Hi-z	I	phbsu50ct12sm
4	DATA16/GPB0/nXBACK	DATA16	Hi-z/–/–	Hi-z/–/–	I	phbsu50ct12sm
5	DATA17/GPB1/nXBREQ	DATA17	Hi-z/–/–	Hi-z/–/–	I	phbsu50ct12sm
6	DATA18/GPB2/TCLK1	DATA18	Hi-z/–/–	Hi-z/–/–	I	phbsu50ct12sm
7	DATA19/GPB3/TXD1	DATA19	Hi-z/–/–	Hi-z/–/–	I	phbsu50ct12sm
8	DATA20/GPB4/RXD1	DATA20	Hi-z/–/–	Hi-z/–/–	I	phbsu50ct12sm
9	DATA21/GPB5/nCTS1	DATA21	Hi-z/–/–	Hi-z/–/–	I	phbsu50ct12sm
10	DATA22/GPB6/nRTS1	DATA22	Hi-z/–/–	Hi-z/–/–	I	phbsu50ct12sm
11	VSSIO	VSSIO	–	–	P	vss3op
12	DATA23/GPB7	DATA23	Hi-z/–/–	Hi-z/–/–	I	phbsu50ct12sm
13	DATA24/GPB8	DATA24	Hi-z/–/–	Hi-z/–/–	I	phbsu50ct12sm
14	DATA25/GPB9/I2SSDI	DATA25	Hi-z/–/–	Hi-z/–/–	I	phbsu50ct12sm
15	DATA26/GPB10/nSS	DATA26	Hi-z/–/–	Hi-z/–/–	I	phbsu50ct12sm
16	DATA27/GPB11	DATA27	Hi-z/–	Hi-z/–	I	phbsu50ct12sm
17	DATA28/GPB12	DATA28	Hi-z/–	Hi-z/–	I	phbsu50ct12sm
18	DATA29/GPB13	DATA29	Hi-z/–	Hi-z/–	I	phbsu50ct12sm
19	DATA30/GPB14	DATA30	Hi-z/–	Hi-z/–	I	phbsu50ct12sm
20	DATA31/GPB15	DATA31	Hi-z/–	Hi-z/–	I	phbsu50ct12sm
21	VDDIO	VDDIO	–	–	P	vdd3op
22	VSSIO	VSSIO	–	–	P	vss3op
23	VSSi	VSSi	–	–	P	Vss3i
24	VDDi	VDDi	–	–	P	vdd1ih_core
25	CLKOUT/GPF6	GPF6	–/–	-/-	I	phbsu50ct8sm
26	nTRST	nTRST	–	–	I	phic
27	TMS	TMS	–	–	I	phic
28	TCK	TCK	–	–	I	phic
29	TDI	TDI	–	–	I	phic
30	TDO	TDO	–	–	O	phot8
31	TOUT0/GPD5	GPD5	–/–	–/–	I	phbsu50ct8sm
32	TOUT1/GPD6	GPD6	–/–	–/–	I	phbsu50ct8sm
33	TOUT2/GPD7	GPD7	–/–/–	–/–/–	I	phbsu50ct8sm

Table 1-1. 208-Pin LQFP Pin Assignment (Continued)

Pin Number	Pin Name	Default Function	I/O State @BUS REQ.	I/O State @STOP	I/O State @nRESET	I/O Type
34	TOUT3/GPD8	GPD8	–/–/–	–/–/–	I	phbsu50ct8sm
35	TCLK0/GPD9	GPD9	–/–	–/–	I	phbsu50ct8sm
36	nXDACK1/GPE9/ nXBACK	GPE9	–/–/–	–/–/–	I	phbsu50ct8sm
37	nXDREQ1/GPE11/ nXBREQ	GPE11	–/–/–	–/–/–	I	phbsu50ct8sm
38	nXDACK0/GPE8	GPE8	–/–	–/–	I	phbsu50ct8sm
39	nXDREQ0/GPE10	GPE10	–/–	–/–	I	phbsu50ct8sm
40	I2SLRCK/GPG0	GPG0	–/–	–/–	I	phbsu50ct8sm
41	VDDi	VDDi	–	–	P	vdd1ih_core
42	VSSi	VSSi	–	–	P	vss3i
43	I2SSCLK/GPG1	GPG1	–/–	–/–	I	phbsu50ct8sm
44	CDCLK/GPG2	GPG2	–/–	–/–	I	phbsu50ct8sm
45	I2SSDO/GPG3/I2SSDI	GPG3	–/–/–	–/–/–	I	phbsu50ct8sm
46	MMCDAT/GPG6/IICSCL	GPG6	–/–/–	–/–/–	I	phbsu50cdct8sm
47	MMCCMD/GPG5/IICSD A	GPG5	–/–/–	–/–/–	I	phbsu50cdct8sm
48	MMCCLK/GPG4/I2SSDI	GPG4	–/–/–	–/–/–	I	phbsu50ct8sm
49	SPICLK/GPG9/MMCCL K	GPG9	–/–/–	–/–/–	I	phbsu50ct8sm
50	SPIMISO/GPG7/IICSDA	GPG7	–/–/–	–/–/–	I	phbsu50cdct8sm
51	SPIMOSI/GPG8/IICSCL	GPG8	–/–/–	–/–/–	I	phbsu50cdct8sm
52	VDDIO	VDDIO	–	–	P	vdd3op
53	VSSIO	VSSIO	–	–	P	vss3op
54	LEND/GPD4	GPD4	–/–	–/–	I	phbsu50ct8sm
55	VCLK/GPD3	GPD3	–/–	–/–	I	phbsu50ct8sm
56	VLINE:HSYNC/GPD2	GPD2	–:–/–	–:–/–	I	phbsu50ct8sm
57	VM:VDEN/GPD1	GPD1	–:–/–	–:–/–	I	phbsu50ct8sm
58	VSSi	VSSi	–	–	P	vss3i
59	VDDi	VDDi	–	–	P	vdd1ih_core
60	VFRAME:VSYNC/GPD0	GPD0	–:–/–	–:–/–	I	phbsu50ct8sm
61	VD0/GPC0	GPC0	–/–	–/–	I	phbsu50ct8sm
62	VD1/GPC1	GPC1	–/–	–/–	I	phbsu50ct8sm
63	VD2/GPC2	GPC2	–/–	–/–	I	phbsu50ct8sm
64	VD3/GPC3	GPC3	–/–	–/–	I	phbsu50ct8sm
65	VSSIO	VSSIO	–	–	P	vss3op

Table 1-1. 208-Pin LQFP Pin Assignment (Continued)

Pin Number	Pin Name	Default Function	I/O State @BUS REQ.	I/O State @STOP	I/O State @nRESET	I/O Type
66	VD4/GPC4	GPC4	–/–	–/–	I	phbsu50ct8sm
67	VD5/GPC5	GPC5	–/–	–/–	I	phbsu50ct8sm
68	VD6/GPC6	GPC6	–/–	–/–	I	phbsu50ct8sm
69	VD7/GPC7	GPC7	–/–	–/–	I	phbsu50ct8sm
70	VD8/GPC8	GPC8	–/–	–/–	I	phbsu50ct8sm
71	VD9/GPC9	GPC9	–/–	–/–	I	phbsu50ct8sm
72	VD10/GPC10	GPC10	–/–	–/–	I	phbsu50ct8sm
73	VSSi	VSSi	–	–	P	vss3i
74	VDDi	VDDi	–	–	P	vdd1ih_core
75	VD11/GPC11	GPC11	–/–	–/–	I	phbsu50ct8sm
76	VD12/GPC12	GPC12	–/–	–/–	I	phbsu50ct8sm
77	VD13/GPC13	GPC13	–/–	–/–	I	phbsu50ct8sm
78	VDDIO	VDDIO	–	–	P	vdd3op
79	VSSIO	VSSIO	–	–	P	vss3op
80	VD14/GPC14	GPC14	–/–	–/–	I	phbsu50ct8sm
81	VD15/GPC15	GPC15	–/–	–/–	I	phbsu50ct8sm
82	DP1/PDP0	PDP0	–/–	–/–	AI	pbusb
83	DN1/PDN0	PDN0	–/–	–/–	AI	pbusb
84	DP0	DP0	–	–	AI	pbusb
85	DN0	DN0	–	–	AI	pbusb
86	VSSi	VSSi	–	–	P	vss3i
87	VDDi	VDDi	–	–	P	vdd1ih_core
88	nCTS0/GPF5/nXBREQ	GPF5	–/–/–	–/–/–	I	phbsu50ct8sm
89	nRTS0/GPF4/nXBACK	GPF4	–/–/–	–/–/–	I	phbsu50ct8sm
90	TXD0/GPF2	GPF2	–/–	–/–	I	phbsu50ct8sm
91	RXD0/GPF0	GPF0	–/–	–/–	I	phbsu50ct8sm
92	TXD1/GPF3/IICSCL	GPF3	–/–/–	–/–/–	I	phbsu50cdct8sm
93	RXD1/GPF1/IICSDA	GPF1	–/–/–	–/–/–	I	phbsu50cdct8sm
94	OM3	OM3	–	–	I	phic
95	OM2	OM2	–	–	I	phic
96	OM1	OM1	–	–	I	phic
97	OM0	OM0	–	–	I	phic
98	VSSIO	VSSIO	–	–	P	vss3op
99	VSSA_ADC	VSSA_ADC	–	–	P	vss3t_abb

Table 1-1. 208-Pin LQFP Pin Assignment (Continued)

Pin Number	Pin Name	Default Function	I/O State @BUS REQ.	I/O State @STOP	I/O State @nRESET	I/O Type
100	Avref	Avref	–	–	AI	phia_abb
101	AIN0	AIN0	–	–	AI	phia_abb
102	AIN1	AIN1	–	–	AI	phia_abb
103	AIN2	AIN2	–	–	AI	phia_abb
104	AIN3	AIN3	–	–	AI	phia_abb
105	AIN4	AIN4	–	–	AI	phia_abb
106	AIN5	AIN5	–	–	AI	phia_abb
107	AIN6	AIN6	–	–	AI	phia_abb
108	AIN7	AIN7	–	–	AI	phia_abb
109	VDDA_ADC	VDDA_ADC	–	–	P	vdd3t_abb
110	XTOrtc	XTOrtc	–	–	AO	phgpad_option
111	XTIrtc	XTIrtc	–	–	AI	phgpad_option
112	RTCVDD	RTCVDD	–	–	P	vdd1ih
113	VDDi_MPLL	VDDi_MPLL	–	–	P	vdd1ih_core
114	VSSi_MPLL	VSSi_MPLL	–	–	P	vss3i
115	MPLLCAP	MPLLCAP	–	–	AI	phgpad_option
116	VDDi_UPLL	VDDi_UPLL	–	–	P	vdd1ih_core
117	VSSi_UPLL	VSSi_UPLL	–	–	P	vss3i
118	UPLLCAP	UPLLCAP	–	–	AI	phgpad_option
119	EINT0/GPE0	GPE0	–/–	–/–	I	phbsu50ct8sm
120	EINT1/GPE1/nSS	GPE1	–/–/–	–/–/–	I	phbsu50ct8sm
121	EINT2/GPE2/I2SSDI	GPE2	–/–/–	–/–/–	I	phbsu50ct8sm
122	EINT3/GPE3/nCTS1	GPE3	–/–/–	–/–/–	I	phbsu50ct8sm
123	EINT4/GPE4/nRTS1	GPE4	–/–/–	–/–/–	I	phbsu50ct8sm
124	EINT5/GPE5/TCLK1	GPE5	–/–/–	–/–/–	I	phbsu50ct8sm
125	EINT6/GPE6	GPE6	–/–/–	–/–/–	I	phbsu50ct8sm
126	EINT7/GPE7	GPE7	–/–/–	–/–/–	I	phbsu50ct8sm
127	N.C		–	–	–	–
128	nRESET	nRESET	–	–	I	phis
129	VDDi	VDDi	–	–	P	vdd1ih
130	EXTCLK	EXTCLK	–	–	I	phic
131	VSSIO	VSSIO	–	–	P	vss3op
132	XTOpll	XTOpll	–	–	AO	phsoscm26

Table 1-1. 208-Pin LQFP Pin Assignment (Continued)

Pin Number	Pin Name	Default Function	I/O State @BUS REQ.	I/O State @STOP	I/O State @nRESET	I/O Type
133	XTIpll	XTIpll	–	–	AI	phsoscm26
134	VDDIO	VDDIO	–	–	P	vdd3op
135	VDDi	VDDi	–	–	P	vdd1ih_core
136	SCLK	SCLK	Hi-z	Low	O(SCLK)	phot12sm
137	VSSi	VSSi	–	–	P	vss3i
138	SCKE/GPA10	SCKE	Hi-z/O	Low/O	O(H)	phot8
139	nWAIT/GPD10	GPD10	–/–	–/–	I	phbsu50ct8sm
140	nGCS7:nSCS1:nRAS1	nGCS7	Hi-z:Hi-z:Hi-z	High:High:Low	O(H)	phot8
141	nGCS6:nSCS0:nRAS0	nGCS6	Hi-z:Hi-z:Hi-z	High:High:Low	O(H)	phot8
142	nGCS5/GPA17	nGCS5	Hi-z/O	Hi-z or Pre/O	O(H)	phot8
143	nGCS4/GPA16	nGCS4	Hi-z/O	Hi-z or Pre/O	O(H)	phot8
144	nGCS3/GPA15	nGCS3	Hi-z/O	Hi-z or Pre/O	O(H)	phot8
145	nGCS2/GPA14	nGCS2	Hi-z/O	Hi-z or Pre/O	O(H)	phot8
146	VSSIO	VSSIO	–	–	P	vss3op
147	nGCS1/GPA13	nGCS1	Hi-z/O	Hi-z or Pre/O	O(H)	phot8
148	nGCS0	nGCS0	Hi-z	Hi-z or Pre	O(H)	phot8
149	nWE	nWE	Hi-z	Hi-z or Pre	O(H)	phot8
150	nOE	nOE	Hi-z	Hi-z or Pre	O(H)	phot8
151	nBE0:nWBE0:DQM0	DQM0	Hi-z:Hi-z:Hi-z	Hi-z or Pre: Hi-z or Pre: Hi-z or Pre	O(H)	phot8
152	nBE1:nWBE1:DQM1	DQM1	Hi-z:Hi-z:Hi-z	Hi-z or Pre: Hi-z or Pre: Hi-z or Pre	O(H)	phot8
153	nBE2:nWBE2:DQM2	DQM2	Hi-z:Hi-z:Hi-z	Hi-z or Pre: Hi-z or Pre: Hi-z or Pre	O(H)	phot8
154	nBE3:nWBE3:DQM3	DQM3	Hi-z:Hi-z:Hi-z	Hi-z or Pre: Hi-z or Pre: Hi-z or Pre	O(H)	phot8
155	nCAS3:nSRAS	nSRAS	Hi-z:Hi-z	Low:High	O(H)	phot8
156	VDDIO	VDDIO	–	–	P	vdd3op
157	VSSIO	VSSIO	–	–	P	vss3op

Table 1-1. 208-Pin LQFP Pin Assignment (Continued)

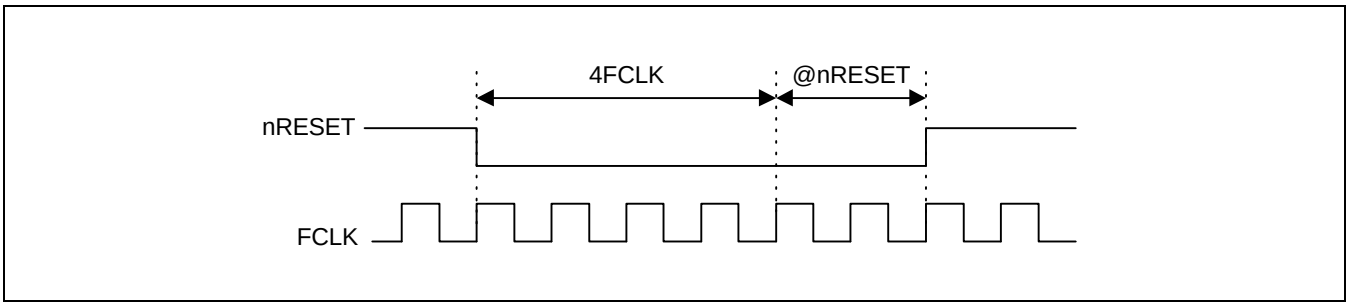
Pin Number	Pin Name	Default Function	I/O State @BUS REQ.	I/O State @STOP	I/O State @nRESET	I/O Type
158	nCAS2:nSCAS	nSCAS	Hi-z:Hi-z	Low:High	O(H)	phot8
159	nCAS1/GPA12	nCAS1	Hi-z/O	Low/O	O(H)	phot8
160	nCAS0/GPA11	nCAS0	Hi-z/O	Low/O	O(H)	phot8
161	ADDR0/GPA0	ADDR0	Hi-z/O	Hi-z or Pre/O	O(L)	phot8
162	ADDR1	ADDR1	Hi-z	Hi-z or Pre	O(L)	phot8
163	ADDR2	ADDR2	Hi-z	Hi-z or Pre	O(L)	phot8
164	ADDR3	ADDR3	Hi-z	Hi-z or Pre	O(L)	phot8
165	ADDR4	ADDR4	Hi-z	Hi-z or Pre	O(L)	phot8
166	ADDR5	ADDR5	Hi-z	Hi-z or Pre	O(L)	phot8
167	ADDR6	ADDR6	Hi-z	Hi-z or Pre	O(L)	phot8
168	ADDR7	ADDR7	Hi-z	Hi-z or Pre	O(L)	phot8
169	ADDR8	ADDR8	Hi-z	Hi-z or Pre	O(L)	phot8
170	ADDR9	ADDR9	Hi-z	Hi-z or Pre	O(L)	phot8
171	ADDR10	ADDR10	Hi-z	Hi-z or Pre	O(L)	phot8
172	ADDR11	ADDR11	Hi-z	Hi-z or Pre	O(L)	phot8
173	VDDIO	VDDIO	–	–	P	vdd3op
174	VSSIO	VSSIO	–	–	P	vss3op
175	VDDi	VDDi	–	–	P	vdd1ih_core
176	VSSi	VSSi	–	–	P	vss3i
177	ADDR12	ADDR12	Hi-z	Hi-z or Pre	O(L)	phot8
178	ADDR13	ADDR13	Hi-z	Hi-z or Pre	O(L)	phot8
179	ADDR14	ADDR14	Hi-z	Hi-z or Pre	O(L)	phot8
180	ADDR15	ADDR15	Hi-z	Hi-z or Pre	O(L)	phot8
181	ADDR16/GPA1	ADDR16	Hi-z/O	Hi-z or Pre/O	O(L)	phot8
182	ADDR17/GPA2	ADDR17	Hi-z/O	Hi-z or Pre/O	O(L)	phot8
183	ADDR18/GPA3	ADDR18	Hi-z/O	Hi-z or Pre/O	O(L)	phot8
184	ADDR19/GPA4	ADDR19	Hi-z/O	Hi-z or Pre/O	O(L)	phot8
185	ADDR20/GPA5	ADDR20	Hi-z/O	Hi-z or Pre/O	O(L)	phot8
186	ADDR21/GPA6	ADDR21	Hi-z/O	Hi-z or Pre/O	O(L)	phot8
187	ADDR22/GPA7	ADDR22	Hi-z/O	Hi-z or Pre/O	O(L)	phot8
188	ADDR23/GPA8	ADDR23	Hi-z/O	Hi-z or Pre/O	O(L)	phot8
189	VDDIO	VDDIO	–	–	P	vdd3op
190	VSSIO	VSSIO	–	–	P	vss3op

Table 1-1. 208-Pin LQFP Pin Assignment (Continued)

Pin Number	Pin Name	Default Function	I/O State @BUS REQ.	I/O State @STOP	I/O State @nRESET	I/O Type
191	ADDR24/GPA9	ADDR24	Hi-z/O	Hi-z or Pre/O	O(L)	phot8
192	DATA0	DATA0	Hi-z	Hi-z	I	phbsu50ct12sm
193	DATA1	DATA1	Hi-z	Hi-z	I	phbsu50ct12sm
194	DATA2	DATA2	Hi-z	Hi-z	I	phbsu50ct12sm
195	DATA3	DATA3	Hi-z	Hi-z	I	phbsu50ct12sm
196	DATA4	DATA4	Hi-z	Hi-z	I	phbsu50ct12sm
197	DATA5	DATA5	Hi-z	Hi-z	I	phbsu50ct12sm
198	DATA6	DATA6	Hi-z	Hi-z	I	phbsu50ct12sm
199	DATA7	DATA7	Hi-z	Hi-z	I	phbsu50ct12sm
200	VDDi	VDDi	–	–	P	vdd1ih_core
201	VSSi	VSSi	–	–	P	vss3i
202	DATA8	DATA8	Hi-z	Hi-z	I	phbsu50ct12sm
203	DATA9	DATA9	Hi-z	Hi-z	I	phbsu50ct12sm
204	DATA10	DATA10	Hi-z	Hi-z	I	phbsu50ct12sm
205	DATA11	DATA11	Hi-z	Hi-z	I	phbsu50ct12sm
206	DATA12	DATA12	Hi-z	Hi-z	I	phbsu50ct12sm
207	DATA13	DATA13	Hi-z	Hi-z	I	phbsu50ct12sm
208	VDDIO	VDDIO	–	–	P	vdd3op

NOTES:

1. The @BUS REQ. shows the pin states at the external bus, which is used by the other bus master. The @STOP shows the pin states when S3C2400X is in STOP mode.
2. ' – ' mark indicates the unchanged pin state at STOP mode or Bus Request mode.
3. Hi-z or Pre means Hi-z or Previous state and which is determined by the setting of MISCCR register.
4. AI/AO means analog input/output.
5. P, I, and O mean power, input and output respectively.
6. The I/O state @nRESET shows the pin status in the below @nRESET duration.



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

I/O Type	Descriptions
vdd1ih, vss3I	1.8V VDD/VSS for internal logic
vdd1ih_core, vss3I	1.8V VDD/VSS for internal logic without input driver
vdd3op, vss3op	3.3V VDD/VSS for external logic
vdd3t_abb, vss3t_abb	3.3V VDD/VSS for analog circuitry
phic	input pad, LVCMOS level
phis	input pad, LVCMOS schmitt-trigger level
pbusb	USB pad
phot8	output pad, tri-state, Io=8mA
phob8sm	output pad, medium slew rate, Io=8mA
phot12sm	output pad, tri-state, medium slew rate, Io=12mA
phia_abb	bi-directional analog pad
phgpad_option	Pad for analog pin
phsoscm26	Oscillator cell with enable and feedback resistor
phbsu50ct8sm	bi-directional pad, LVCMOS schmit-trigger, 50Kohm pull-up resistor with control, tri-state, Io=8mA
phbsu50ct12sm	bi-directional pad, LVCMOS schmit-trigger, 50Kohm pull-up resistor with control, tri-state, Io=12mA
phbsu50cdct8sm	bi-directional pad, LVCMOS schmit-trigger, 50Kohm pull-up resistor with control, tri-state, selectable output pad(open-drain or tri-state), Io=8mA

Table 1-2. 208-Pin FBGA (12×12, 0.65mm Pitch) Pin Assignment

Pin Number	Pin Name	Pin Number	Pin Name
A1	VSSIO	C1	DATA20/GPB4/RXD1
A2	DATA12	C2	DATA19/GPB3/TXD1
A3	DATA9	C3	DATA15
A4	DATA8	C4	DATA13
A5	DATA6	C5	DATA11
A6	DATA2	C6	DATA7
A7	VSSIO	C7	DATA1
A8	ADDR21/GPA6	C8	ADDR23/GPA8
A9	ADDR17/GPA2	C9	ADDR19/GPA4
A10	ADDR13	C10	ADDR15
A11	VDDi	C11	VSSIO
A12	ADDR10	C12	ADDR9
A13	ADDR6	C13	ADDR4
A14	ADDR3	C14	nCAS1/GPA12
A15	ADDR1	C15	VDDIO
A16	ADDR0/GPA0	C16	nCAS3:nSRAS
B1	DATA17/GPB1/nXBREQ	D1	DATA23/GPB7
B2	DATA14	D2	VSSIO
B3	VDDIO	D3	DATA21/GPB5/nCTS1
B4	DATA10	D4	DATA18/GPB2/TCLK1
B5	VDDi	D5	DATA16/GPB0/nXBACK
B6	DATA3	D6	VSSi
B7	DATA0	D7	DATA4
B8	ADDR22/GPA7	D8	VDDIO
B9	ADDR18/GPA3	D9	ADDR20/GPA5
B10	ADDR14	D10	ADDR12
B11	VSSi	D11	VDDIO
B12	ADDR11	D12	ADDR5
B13	ADDR7	D13	VSSIO
B14	ADDR2	D14	nBE3:nWBE3:DQM3
B15	nCAS0/GPA11	D15	nBE2:nWBE2:DQM2
B16	nCAS2:nSCAS	D16	nBE1:nWBE1:DQM1

Table 1-2. 208-Pin FBGA (12×12, 0.65mm Pitch) Pin Assignment (Continued)

Pin Number	Pin Name	Pin Number	Pin Name
E1	DATA25/GPB9/I2SSDI	H1	nTRST
E2	DATA26/GPB10/nSS	H2	TMS
E3	DATA24/GPB8	H3	TCK
E4	DATA22/GPB6/nRTS1	H4	TDI
E7	DATA5	H5	CLKOUT/GPF6
E8	ADDR24/GPA9	H12	SCKE/GPA10
E9	ADDR16/GPA1	H13	VDDIO
E10	ADDR8	H14	VDDi
E13	nBE0:nWBE0:DQM0	H15	SCLK
E14	nOE	H16	VSSi
E15	nWE	J1	TDO
E16	nGCS0	J2	TOUT0/GPD5
F1	DATA29/GPB13	J3	TOUT1/GPD6
F2	DATA30/GPB14	J4	TOUT2/GPD7
F3	DATA31/GPB15	J5	I2SLRCK/GPG0
F4	DATA27/GPB11	J12	EINT7/GPE7
F13	nGCS1/GPA13	J13	EXTCLK
F14	nGCS4/GPA16	J14	VSSIO
F15	nGCS3/GPA15	J15	XTOpll
F16	nGCS2/GPA14	J16	XTIpll
G1	VSSIO	K1	TOUT3/GPD8
G2	VSSi	K2	TCLK0/GPD9
G3	VDDi	K3	nXDACK1/GPE9/nXBACK
G4	VDDIO	K4	nXDREQ1/GPE11/nXBREQ
G5	DATA28/GPB12	K5	I2SSCLK/GPG1
G12	VSSIO	K12	EINT0/GPE0
G13	nGCS5/GPA17	K13	EINT3/GPE3/nCTS1
G14	nWAIT/GPD10	K14	N.C
G15	nGCS7:nSCS1:nRAS1	K15	nRESET
G16	nGCS6:nSCS0:nRAS0	K16	VDDi

Table 1-2. 208-Pin FBGA (12×12, 0.65mm Pitch) Pin Assignment (Continued)

Pin Number	Pin Name	Pin Number	Pin Name
L1	nXDACK0/GPE8	N1	MMCDAT/GPG6/IICSCL
L2	nXDREQ0/GPE10	N2	MMCCLK/GPG4/I2SSDI
L3	VDDi	N3	SPIMOSI/GPG8/IICSCL
L4	I2SSDO/GPG3/I2SSDI	N4	VM:VDEN/GPD1
L13	UPLLCAP	N5	VD0/GPC0
L14	EINT4/GPE4/nRTS1	N6	VD7/ GPC7
L15	EINT5/GPE5/TCLK1	N7	VSSi
L16	EINT6/GPE6	N8	VDDIO
M1	VSSi	N9	DN0
M2	CDCLK/GPG2	N10	RXD0/GPF0
M3	MMCCMD/GPG5/IICSDA	N11	VSSA_ADC
M4	SPIMISO/GPG7/IICSDA	N12	AIN2
M7	VD4/GPC4	N13	VDDA_ADC
M8	VD13/ GPC13	N14	VSSi_MPLL
M9	nRTS0/GPF4/nXBACK	N15	MPLLCAP
M10	OM2	N16	VSSi_UPLL
M13	RTCVDD	P1	SPICLK/GPG9/MMCCLK
M14	VDDi_UPLL	P2	VDDIO
M15	EINT1/GPE1/nSS	P3	LEND/GPD4
M16	EINT2/GPE2/I2SSDI	P4	VSSi
		P5	VD3/GPC3
		P6	VD8/ GPC8
		P7	VDDi
		P8	VD15/ GPC15
		P9	DP0
		P10	nCTS0/GPF5/nXBREQ
		P11	RXD1/GPF1/IICSDA
		P12	OM0
		P13	AIN1
		P14	AIN7
		P15	XTIrtc
		P16	VDDi_MPLL

Table 1-2. 208-Pin FBGA (12×12, 0.65mm Pitch) Pin Assignment (Continued)

Pin Number	Pin Name	Pin Number	Pin Name
R1	VSSIO	T1	VLINE:HSYNC/GPD2
R2	VCLK/GPD3	T2	VDDi
R3	VFRAME:VSYNC/GPD0	T3	VD1/GPC1
R4	VD2/GPC2	T4	VSSIO
R5	VD5/ GPC5	T5	VD6/ GPC6
R6	VD9/ GPC9	T6	VD10/ GPC10
R7	VD12/ GPC12	T7	VD11/ GPC11
R8	VD14/ GPC14	T8	VSSIO
R9	DN1/PDN0	T9	DP1/PDP0
R10	VDDi	T10	VSSi
R11	TXD1/GPF3/IICSCL	T11	TXD0/GPF2
R12	OM1	T12	OM3
R13	Avref	T13	VSSIO
R14	AIN4	T14	AIN0
R15	AIN6	T15	AIN3
R16	XTOrtc	T16	AIN5

Table 1-3. 208-Pin FBGA (15×15, 0.8mm Pitch) Pin Assignment

Pin #	Pin Name	Pin #	Pin Name
A1	VSSIO	C1	DATA17/GPB1/nXBREQ
A2	DATA11	C2	DATA14
A3	DATA10	C3	DATA13
A4	VSSi	C4	DATA9
A5	VDDi	C5	DATA7
A6	DATA5	C6	DATA3
A7	DATA1	C7	ADDR24/GPA9
A8	VDDI0	C8	ADDR22/GPA7
A9	ADDR20/GPA5	C9	ADDR18/GPA3
A10	ADDR16/GPA1	C10	ADDR14
A11	ADDR12	C11	VDDi
A12	VDDIO	C12	ADDR10
A13	ADDR8	C13	ADDR6
A14	ADDR4	C14	ADDR2
A15	ADDR0/GPA0	C15	nCAS3:nSRAS
A16	nCAS0/GPA11	C16	nBE3:nWBE3:DQM3
A17	VSSIO	C17	nBE1:nWBE1:DQM1
B1	DATA16/GPB0/nXBACK	D1	DATA21/GPB5/nCTS1
B2	VDDIO	D2	DATA18/GPB2/TCLK1
B3	DATA12	D3	DATA19/GPB3/TXD1
B4	DATA8	D4	DATA15
B5	DATA6	D5	DATA4
B6	DATA2	D6	DATA0
B7	VSSIO	D7	ADDR23/GPA8
B8	ADDR21/GPA6	D8	ADDR19/GPA4
B9	ADDR17/GPA2	D9	ADDR15
B10	ADDR13	D10	VSSi
B11	VSSIO	D11	ADDR11
B12	ADDR9	D12	ADDR7
B13	ADDR5	D13	ADDR3
B14	ADDR1	D14	nCAS1/GPA12
B15	nCAS2:nSCAS	D15	nBE0:nWBE0:DQM0
B16	VDDIO	D16	nOE
B17	nBE2:nWBE2:DQM2	D17	nWE

Table 1-3. 208-Pin FBGA (15×15, 0.8mm Pitch) Pin Assignment (Continued)

Pin #	Pin Name	Pin #	Pin Name
E1	DATA24/GPB8	H1	CLKOUT/GPF6
E2	DATA22/GPB6/nRTS1	H2	VSSIO
E3	VSSIO	H3	VSSi
E4	DATA20/GPB4/RXD1	H4	DATA31/GPB15
E14	nGCS3/GPA15	H14	XTOpll
E15	nGCS1/GPA13	H15	VDDi
E16	VSSIO	H16	VDDIO
E17	nGCS0	H17	VSSi
F1	DATA28/GPB12	J1	TDI
F2	DATA25/GPB9/I2SSDI	J2	nTRST
F3	DATA26/GPB10/nSS	J3	TMS
F4	DATA23/GPB7	J4	VDDi
F14	nGCS7:nSCS1:nRAS1	J14	nRESET
F15	nGCS4/GPA16	J15	VSSIO
F16	nGCS5/GPA17	J16	EXTCLK
F17	nGCS2/GPA14	J17	XTIpll
G1	VDDIO	K1	TOUT2/GPD7
G2	DATA29/GPB13	K2	TDO
G3	DATA30/GPB14	K3	TOUT0/GPD5
G4	DATA27/GPB11	K4	TCK
G14	SCLK	K14	EINT5/GPE5/TCLK1
G15	nWAIT/GPD10	K15	N. C.
G16	SCKE/GPA10	K16	EINT7/GPE7
G17	nGCS6:nSCS0:nRAS0	K17	VDDi

Table 1-3. 208-Pin FBGA (15×15, 0.8mm Pitch) Pin Assignment (Continued)

Pin #	Pin Name	Pin #	Pin Name
L1	nXDREQ1/GPE11/nXBREQ	N1	CDCLK/GPG2
L2	TOUT3/GPD8	N2	VSSi
L3	TCLK0/GPD9	N3	I2SSCLK/GPG1
L4	TOUT1/GPD6	N4	I2SLRCK/GPG0
L14	EINT1/GPE1/nSS	N14	RTCVDD
L15	EINT4/GPE4/nRTS1	N15	MPLLCAP
L16	EINT3/GPE3/nCTS1	N16	VSSi_MPLL
L17	EINT6/GPE6	N17	VSSi_UPLL
M1	VDDi	P1	I2SSDO/GPG3/I2SSDR
M2	nXDACK0/GPE8	P2	MMCDAT/GPG6/IICSCL
M3	nXDREQ0/GPE10	P3	MMCCMD/GPG5/IICSDA
M4	nXDACK1/GPE9/nXBACK	P4	VCLK/GPD3
M14	VDDi_UPLL	P5	VFRAME:VSYNC/GPD0
M15	EINT0/GPE0	P6	VD3/GPC3
M16	UPLLCAP	P7	VD6/GPC6
M17	EINT2/GPE2/I2SSDI	P8	VD10/GPC10
		P9	VD12/GPC12
		P10	VD14/GPC14
		P11	DP0
		P12	nCTS0/GPF5/nXBREQ
		P13	TXD1/GPF3/IICSCL
		P14	AIN6
		P15	XTIrtc
		P16	XTOrtc
		P17	VDDi_MPLL

Table 1-3. 208-Pin FBGA (15×15, 0.8mm Pitch) Pin Assignment (Continued)

Pin #	Pin Name	Pin #	Pin Name
R1	MMCCLK/GPG4/I2SSDI	U1	VSSIO
R2	SPIMISO/GPG7/IICSDA	U2	VLINE:HSYNC/GPD2
R3	SPIMOSI/GPG8/IICSCL	U3	VM:VDEN/GPD1
R4	VDDi	U4	VD0/GPC0
R5	VD2/ GPC2	U5	VSSIO
R6	VD5/ GPC5	U6	VD7/GPC7
R7	VD9/ GPC9	U7	VSSi
R8	VD11/ GPC11	U8	VD13/GPC13
R9	VSSIO	U9	VD15/GPC15
R10	DN1/PDN0	U10	DN0
R11	VDDi	U11	nRTS0/GPF4/nXBACK
R12	RXD0/GPF0	U12	RXD1/GPF1/IICSDA
R13	OM2	U13	OM1
R14	VSSA_ADC	U14	OM0
R15	AIN2	U15	Avref
R16	AIN5	U16	AIN0
R17	VDDA_ADC	U17	AIN4
T1	SPICLK/GPG9/MMCCLK		
T2	VDDIO		
T3	LEND/GPD4		
T4	VSSi		
T5	VD1/ GPC1		
T6	VD4/ GPC4		
T7	VD8/ GPC8		
T8	VDDi		
T9	VDDIO		
T10	DP1/PDP0		
T11	VSSi		
T12	TXD0/GPF2		
T13	OM3		
T14	VSSIO		
T15	AIN1		
T16	AIN3		
T17	AIN7		

SIGNAL DESCRIPTIONS

Table 1-4. S3C2400X Signal Descriptions

Signal	I/O	Description
BUS CONTROLLER		
OM[1:0]	I	OM[1:0] sets S3C2400X in the TEST mode, which is used only at fabrication. Also, it determines the bus width of nGCS0. The logic level is determined by the pull-up/down resistor during the RESET cycle. 00:8-bit 01:16-bit 10:32-bit 11:Test mode
ADDR[24:0]	O	ADDR[24:0] (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.
nWBE[3:0]	O	Write Byte Enable
nBE[3:0]	O	Upper Byte/Lower Byte Enable(In case of SRAM)
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 S3C2400X 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.
DRAM/SDRAM/SRAM		
nRAS[1:0]	O	Row Address Strobe
nCAS[3:0]	O	Column Address strobe
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	O	SDRAM Clock
SCKE	O	SDRAM Clock Enable
nBE[3:0]	O	16-bit SRAM Byte Enable

Table 1-4. S3C2400X Signal Descriptions (Continued)

Signal	I/O	Description
LCD CONTROL UNIT		
VD[15:0]	O	**STN/TFT:** LCD Data Bus
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
INTERRUPT CONTROL UNIT		
EINT[7:0]	I	External Interrupt request
DMA		
nXDREQ[1:0]	I	External DMA request
nXDACK[1:0]	O	External DMA acknowledge
UART		
RxD[1:0]	I	UART receives data input
TxD[1:0]	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
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

Table 1-4. S3C2400X Signal Descriptions (Continued)

Signal	I/O	Description
ADC		
AIN[7:0]	AI	ADC input[7:0]
Avref	AI	ADC Vref
USB HOST		
DN[1:0]	IO	DATA - from USB host
DP[1:0]	IO	DATA + from USB host
USB DEVICE		
PDN0	IO	DATA - for USB peripheral
PDP0	IO	DATA + for USB peripheral
SPI		
SPIMISO	IO	SPIMISO is the master data input line, when SPI is configured as a master. When SPI is configured as a slave, this pin reverse its role.
SPIMOSI	IO	SPIMOSI is the master data output line, when SPI is configured as a master. When SPI is configured as a slave, this pin reverse its role.
SPICLK	IO	SPI clock
nSS	IO	SPI chip select When SPI is configured as a master and ENMUL is set, nSS is a slave select. When SPI is configured as a slave, nSS is also a slave select.
MMC		
MMCDAT	IO	MMC receive/transmit data
MMCCMD	IO	MMC receive/transmit command
MMCCLK	O	MMC clock
GENERAL PORT		
GPn[89:0]	IO	General input/output ports (some ports are output mode only)
TIMMER/PWM		
TOUT[3:0]	O	Timer output[3:0]
TCLK[1:0]	I	External clock input

Table 1-4. S3C2400X Signal Descriptions (Continued)

Signal	I/O	Description
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 at L or low active pulse.
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 S3C2400X 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.
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.
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 H (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 H (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.

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

Table 1-4. S3C2400X Signal Descriptions (Continued)

Signal	I/O	Description
RESET & CLOCK & POWER (continued)		
MPLLCAP	AI	Loop filter capacitor for main clock.
UPLLCAP	AI	Loop filter capacitor for USB clock.
XTIrtc	AI	32 KHz crystal input for RTC.
XTOrtc	AO	32 KHz crystal output for RTC.
CLKOUT	O	Clock output signal. The CLKSEL of MISCCR register configures the clock output mode among the MPLL CLK, UPLL CLK, FCLK, HCLK, PCLK.
POWER		
VDDi	P	S3C2400X core logic VDD(1.8V) for CPU.
VSSi	P	S3C2400X core logic VSS
VDDi_MPLL	P	S3C2400X MPLL analog and digital VDD (1.8 V).
VSSi_MPLL	P	S3C2400X MPLL analog and digital VSS.
VDDIO	P	S3C2400X I/O port VDD(3.3V)
VSSIO	P	S3C2400X I/O port VSS
RTCVDD	P	RTC VDD (1.8 V, Not support 3.3V) (This pin must be connected to power properly if RTC isn't used)
VDDi_UPLL	P	S3C2400X UPLL analog and digital VDD (1.8V)
VSSi_UPLL	P	S3C2400X UPLL analog and digital VSS
VDDA_ADC	P	S3C2400X ADC VDD(3.3V)
VSSA_ADC	P	S3C2400X ADC VSS

S3C2400X SPECIAL REGISTERS

Table 1-5. S3C2400X Special Registers

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
MEMORY CONTROLLER					
BWSCON	0x14000000	←	W	R/W	Bus Width & Wait Status Control
BANKCON0	0x14000004				Boot ROM Control
BANKCON1	0x14000008				BANK1 Control
BANKCON2	0x1400000c				BANK2 Control
BANKCON3	0x14000010				BANK3 Control
BANKCON4	0x14000014				BANK4 Control
BANKCON5	0x14000018				BANK5 Control
BANKCON6	0x1400001c				BANK6 Control
BANKCON7	0x14000020				BANK7 Control
REFRESH	0x14000024				DRAM/SDRAM Refresh Control
BANKSIZE	0x14000028				Flexible Bank Size
MRSRB6	0x1400002c				Mode register set for SDRAM
MRSRB7	0x14000030				Mode register set for SDRAM

Table 1-5. S3C2400X Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
USB HOST CONTROLLER					
HcRevision	0x14200000	←	W		Control and Status Group
HcControl	0x14200004				
HcCommonStatus	0x14200008				
HcInterruptStatus	0x1420000c				
HcInterruptEnable	0x14200010				
HcInterruptDisable	0x14200014				
HcHCCA	0x14200018				Memory Pointer Group
HcPeriodCuttentED	0x1420001c				
HcControlHeadED	0x14200020				
HcControlCurrentED	0x14200024				
HcBulkHeadED	0x14200028				
HcBulkCurrentED	0x1420002c				
HcDoneHead	0x14200030				
HcRmInterval	0x14200034				Frame Counter Group
HcFmRemaining	0x14200038				
HcFmNumber	0x1420003c				
HcPeriodicStart	0x14200040				
HcLSThreshold	0x14200044				
HcRhDescriptorA	0x14200048				Root Hub Group
HcRhDescriptorB	0x1420004c				
HcRhStatus	0x14200050				
HcRhPortStatus1	0x14200054				
HcRhPortStatus2	0x14200058				
INTERRUPT CONTROLLER					
SRCPND	0x14400000	←	W	R/W	Interrupt Request Status
INTMOD	0x14400004			W	Interrupt Mode Control
INTMSK	0x14400008			R/W	Interrupt Mask Control
PRIORITY	0x1440000c			W	IRQ Priority Control
INTPND	0x14400010			R/W	Interrupt Request Status
INTOFFSET	0x14400014			R	Interrupt request source

Table 1-5. S3C2400X Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
DMA					
DISRC0	0x14600000	←	W	R/W	DMA 0 Initial Source
DIDST0	0x14600004				DMA 0 Initial Destination
DCON0	0x14600008				DMA 0 Control
DSTAT0	0x1460000c			R	DMA 0 Count
DCSRC0	0x14600010				DMA 0 Current Source Address
DCDST0	0x14600014				DMA 0 Current Destination Address
DMASKTRIG0	0x14600018			R/W	DMA 0 Mask Trigger
DISRC1	0x14600020	←	W	R/W	DMA 1 Initial Source
DIDST1	0x14600024				DMA 1 Initial Destination
DCON1	0x14600028				DMA 1 Control
DSTAT1	0x1460002c			R	DMA 1 Count
DCSRC1	0x14600030				DMA 1 Current Source Address
DCDST1	0x14600034				DMA 1 Current Destination Address
DMASKTRIG1	0x14600038			R/W	DMA 1 Mask Trigger
DISRC2	0x14600040	←	W	R/W	DMA 2 Initial Source
DIDST2	0x14600044				DMA 2 Initial Destination
DCON2	0x14600048				DMA 2 Control
DSTAT2	0x1460004c			R	DMA 2 Count
DCSRC2	0x14600050				DMA 2 Current Source Address
DCDST2	0x14600054				DMA 2 Current Destination Address
DMASKTRIG2	0x14600058			R/W	DMA 2 Mask Trigger
DISRC3	0x14600060	←	W	R/W	DMA 3 Initial Source
DIDST3	0x14600064				DMA 3 Initial Destination
DCON3	0x14600068				DMA 3 Control
DSTAT3	0x1460006c			R	DMA 3 Count
DCSRC3	0x14600060				DMA 3 Current Source Address
DCDST3	0x14600064				DMA 3 Current Destination Address
DMASKTRIG3	0x14600068			R/W	DMA 3 Mask Trigger

Table 1-5. S3C2400X Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
CLOCK & POWER MANAGEMENT					
LOCKTIME	0x14800000	←	W	R/W	PLL Lock Time Counter
MPLLCON	0x14800004				MPLL Control
UPLLCON	0x14800008				UPLL Control
CLKCON	0x1480000c				Clock Generator Control
CLKSLOW	0x14800010				Slow Clock Control
CLKDIVN	0x14800014				Clock divider Control
LCD CONTROLLER					
LCDCON1	0x14a00000	←	W	R/W	LCD Control 1
LCDCON2	0x14a00004				LCD Control 2
LCDCON3	0x14a00008				LCD Control 3
LCDCON4	0x14a0000c				LCD Control 4
LCDCON5	0x14a00010				LCD Control 5
LCDSADDR1	0x14a00014				STN/TFT: Frame Buffer Start Address1
LCDSADDR2	0x14a00018				STN/TFT: Frame Buffer Start Address2
LCDSADDR3	0x14a0001c				STN/TFT: Virtual Screen Address Set
REDLUT	0x14a00020				STN: Red Lookup Table
GREENLUT	0x14a00024				STN: Green Lookup Table
BLUELUT	0x14a00028				STN: Blue Lookup Table
DITHMODE	0x14a0004c				STN: Dithering Mode
TPAL	0x14a00050				TFT: Temporary Palette

Table 1-5. S3C2400X Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
UART					
ULCON0	0x15000000	←	W	R/W	UART 0 Line Control
UCON0	0x15000004				UART 0 Control
UFCON0	0x15000008				UART 0 FIFO Control
UMCON0	0x1500000c				UART 0 Modem Control
UTRSTAT0	0x15000010			R	UART 0 Tx/Rx Status
UERSTAT0	0x15000014				UART 0 Rx Error Status
UFSTAT0	0x15000018				UART 0 FIFO Status
UMSTAT0	0x1500001c				UART 0 Modem Status
UTXH0	0x15000023	0x15000020	B	W	UART 0 Transmission Hold
URXH0	0x15000027	0x15000024		R	UART 0 Receive Buffer
UBRDIV0	0x15000028	←	W	R/W	UART 0 Baud Rate Divisor
ULCON1	0x15004000	←	W	R/W	UART 1 Line Control
UCON1	0x15004004				UART 1 Control
UFCON1	0x15004008				UART 1 FIFO Control
UMCON1	0x1500400c				UART 1 Modem Control
UTRSTAT1	0x15004010			R	UART 1 Tx/Rx Status
UERSTAT1	0x15004014				UART 1 Rx Error Status
UFSTAT1	0x15004018				UART 1 FIFO Status
UMSTAT1	0x1500401c				UART 1 Modem Status
UTXH1	0x15004023	0x15004020	B	W	UART 1 Transmission Hold
URXH1	0x15004027	0x15004024		R	UART 1 Receive Buffer
UBRDIV1	0x15004028	←	W	R/W	UART 1 Baud Rate Divisor

Table 1-5. S3C2400X Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
PWM TIMER					
TCFG0	0x15100000	←	W	R/W	Timer Configuration
TCFG1	0x15100004				Timer Configuration
TCON	0x15100008				Timer Control
TCNTB0	0x1510000c				Timer Count Buffer 0
TCMPB0	0x15100010				Timer Compare Buffer 0
TCNTO0	0x15100014			R	Timer Count Observation 0
TCNTB1	0x15100018			R/W	Timer Count Buffer 1
TCMPB1	0x1510001c				Timer Compare Buffer 1
TCNTO1	0x15100020			R	Timer Count Observation 1
TCNTB2	0x15100024			R/W	Timer Count Buffer 2
TCMPB2	0x15100028				Timer Compare Buffer 2
TCNTO2	0x1510002c			R	Timer Count Observation 2
TCNTB3	0x15100030			R/W	Timer Count Buffer 3
TCMPB3	0x15100034				Timer Compare Buffer 3
TCNTO3	0x15100038			R	Timer Count Observation 3
TCNTB4	0x1510003c			R/W	Timer Count Buffer 4
TCNTO4	0x15100040			R	Timer Count Observation 4

Table 1-5. S3C2400X Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/Write	Function
USB DEVICE					
FUNC_ADDR_REG	0x15200140	←	W	R/W	Function Address
PWR_REG	0x15200144				Power Management
INT_REG	0x15200148			R	Interrupt Pending and Clear
INT_MASK_REG	0x1520014c			R/W	Interrupt Mask
FRAME_NUM_REG	0x15200150			R	Frame Number
RESUME_CON_REG	0x15200154			R/W	Resume Signal Control
EP0_CSR	0x15200160				Clock Generator Control
EP0_MAXP	0x15200164				End Point0 MAX Packet
EP0_OUT_CNT	0x15200168				End Point0 Out Write Count
EP0_FIFO	0x1520016c				End Point0 FIFO Read/Write
EP1_IN_CSR	0x15200180				End Point1 in Control Status
EP1_IN_MAXP	0x15200184				End Point1 in MAX Packet
EP1_FIFO	0x15200188			W	End Point2 FIFO Write
EP2_IN_CSR	0x15200190			R/W	End Point2 in Control Status
EP2_IN_MAXP	0x15200194				End Point2 in MAX Packet
EP2_FIFO	0x15200198			W	End Point2 FIFO Write
EP3_OUT_CSR	0x152001a0			R/W	End Point3 Out Control Status
EP3_OUT_MAXP	0x152001a4				End Point3 Out MAX Packet
EP3_OUT_CNT	0x152001a8			R	End Point3 Out Write Count
EP3_FIFO	0x152001ac				End Point3 FIFO Read
EP4_OUT_CSR	0x152001b0			R/W	End Point4 Out Control Status
EP4_OUT_MAXP	0x152001b4				End Point4 Out MAX Packet
EP4_OUT_CNT	0x152001b8			R	End Point4 Out Write Count
EP4_FIFO	0x152001bc				End Point4 FIFO Read
DMA_CON	0x152001c0			R/W	DMA Interface Control
DMA_UNIT	0x152001c4				DMA Transfer Unit Counter
DMA_FIFO	0x152001c8				DMA Transfer FIFO Counter
DMA_TX	0x152001cc				DMA Total Transfer Counter
TEST_MODE	0x152001f4			W	Test Mode Control
IN_CON_REG	0x152001f8				In Packet Number Control

Table 1-5. S3C2400X Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
WATCHDOG TIMER					
WTCON	0x15300000	←	W	R/W	Watch-Dog Timer Mode
WTDAT	0x15300004				Watch-Dog Timer Data
WTCNT	0x15300008				Watch-Dog Timer Count
IIC					
IICCON	0x15400000	←	W	R/W	IIC Control
IICSTAT	0x15400004				IIC Status
IICADD	0x15400008				IIC Address
IICDS	0x1540000c				IIC Data Shift
IIS					
IISCON	0x15508000,02	0x15508000	HW,W	R/W	IIS Control
IISMOD	0x15508004,06	0x15508004	HW,W		IIS Mode
IISPSR	0x15508008,0a	0x15508008	HW,W		IIS Prescaler
IISFIFCON	0x1550800c,0e	0x1550800c	HW,W		IIS FIFO Control
IISFIF	0x15508012	0x15508010	HW		IIS FIFO Entry

Table 1-5. S3C2400X Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
I/O PORT					
PACON	0x15600000	←	W	R/W	Port A Control
PADAT	0x15600004				Port A Data
PBCON	0x15600008				Port B Control
PBDAT	0x1560000c				Port B Data
PBUP	0x15600010				Pull-up Control B
PCCON	0x15600014				Port C Control
PCDAT	0x15600018				Port C Data
PCUP	0x1560001c				Pull-up Control C
PDCON	0x15600020				Port D Control
PDDAT	0x15600024				Port D Data
PDUP	0x15600028				Pull-up Control D
PECON	0x1560002c				Port E Control
PEDAT	0x15600030				Port E Data
PEUP	0x15600034				Pull-up Control E
PFCON	0x15600038				Port F Control
PFDAT	0x1560003c				Port F Data
PFUP	0x15600040				Pull-up Control F
PGCON	0x15600044				Port G Control
PGDAT	0x15600048				Port G Data
PGUP	0x1560004c				Pull-up Control G
OPENCR	0x15600050				Open Drain Enable
MISCCR	0x15600054				Miscellaneous Control
EXTINT	0x15600058				External Interrupt Control

Table 1-5. S3C2400X Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
RTC					
RTCCON	0x15700043	0x15700040	B	R/W	RTC Control
TICINT	0x15700047	0x15700044			Tick time count
RTCALM	0x15700053	0x15700050			RTC Alarm Control
ALMSEC	0x15700057	0x15700054			Alarm Second
ALMMIN	0x1570005b	0x15700058			Alarm Minute
ALMHOUR	0x1570005f	0x1570005c			Alarm Hour
ALMDAY	0x15700063	0x15700060			Alarm Day
ALMMON	0x15700067	0x15700064			Alarm Month
ALMYEAR	0x1570006b	0x15700068			Alarm Year
RTCRST	0x1570006f	0x1570006c			RTC Round Reset
BCDSEC	0x15700073	0x15700070			BCD Second
BCDMIN	0x15700077	0x15700074			BCD Minute
BCDHOUR	0x1570007b	0x15700078			BCD Hour
BCDDAY	0x1570007f	0x1570007c			BCD Day
BCDDATE	0x15700083	0x15700080			BCD Date
BCDMON	0x15700087	0x15700084			BCD Month
BCDYEAR	0x1570008b	0x15700088			BCD Year
A/D CONVERTER					
ADCCON	0x15800000	←	W	R/W	ADC Control
ADCDAT	0x15800004			R	ADC Data
SPI					
SPCON	0x15900000	←	W	R/W	SPI Control
SPSTA	0x15900004			R	SPI Status
SPPIN	0x15900008			R/W	SPI Pin Control
SPPRE	0x1590000c				SPI Baud Rate Prescaler
SPTDAT	0x15900010				SPI Tx Data
SPRDAT	0x15900014			R	SPI Rx Data

Table 1-5. S3C2400X Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
MMC INTERFACE					
MMCON	0x15a00000,02,03	0x15a00000	B, HW, W	R/W	MMC Control
MMCRR	0x15a00004,06,07	0x15a00004			MMC Command
MMFCON	0x15a00008,0a,0b	0x15a00008			MMC FIFO Control
MMSTA	0x15a0000c,0e,0f	0x15a0000c		R	MMC Status
MMFSTA	0x15a00010,12	0x15a00010	HW, W		MMC FIFO Status
MMPRE	0x15a00014,16,17	0x15a00014	B, HW, W	R/W	MMC Baud Rate Prescaler
MMLEN	0x15a00018,1a	0x15a00018	HW, W		MMC Block Length
MMCR7	0x15a0001c,0e,0f	0x15a0001c	B, HW, W	R	Response CRC7
MMRSP0	0x15a00020	←	W		MMC Response Status 0
MMRSP1	0x15a00024				MMC Response Status 1
MMRSP2	0x15a00028				MMC Response Status 2
MMRSP3	0x15a0002c				MMC Response Status 3
MMCMD0	0x15a00030,32,33	0x15a00030	B, HW, W	R/W	MMC Command 0
MMCMD1	0x15a00034	←	W		MMC Command 1
MMCR16	0x15a00038,3a	0x15a00038	HW, W	R	Data Read CRC16 Buffer
MMDAT	0x15a0003c,3e,3f	0x15a0003c	B, HW, W	R/W	MMC Data

IMPORTANT NOTES ABOUT S3C2400X 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 by the recommended access unit.
3. All registers except ADC registers, RTC registers and UART registers must be read/written in word unit (32bit) at little/big endian.
4. It is very important 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 *).

NOTES

2 PROGRAMMER'S MODEL

OVERVIEW

S3C2400X 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.

NOTES

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.

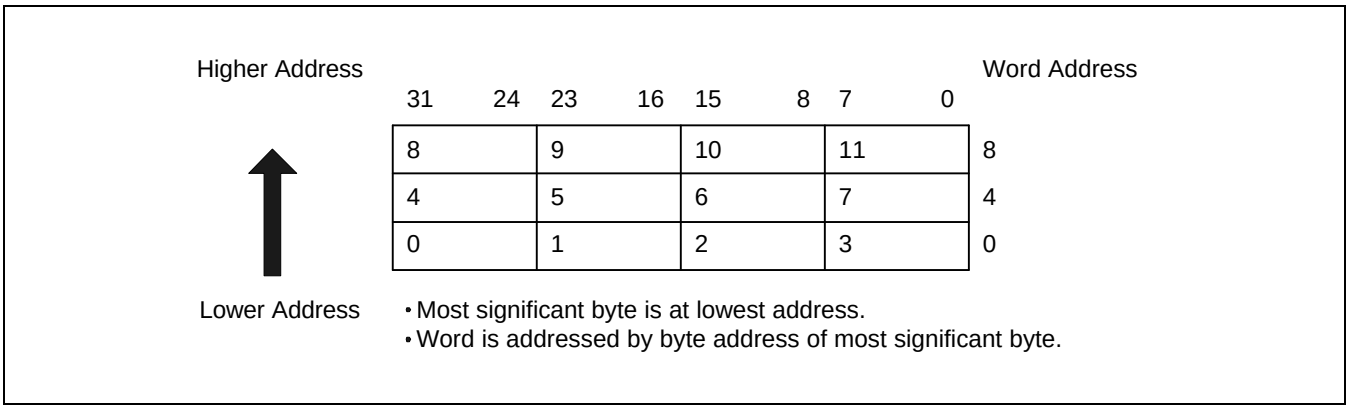


Figure 2-1. Big-Endian Addresses of Bytes within Words

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.

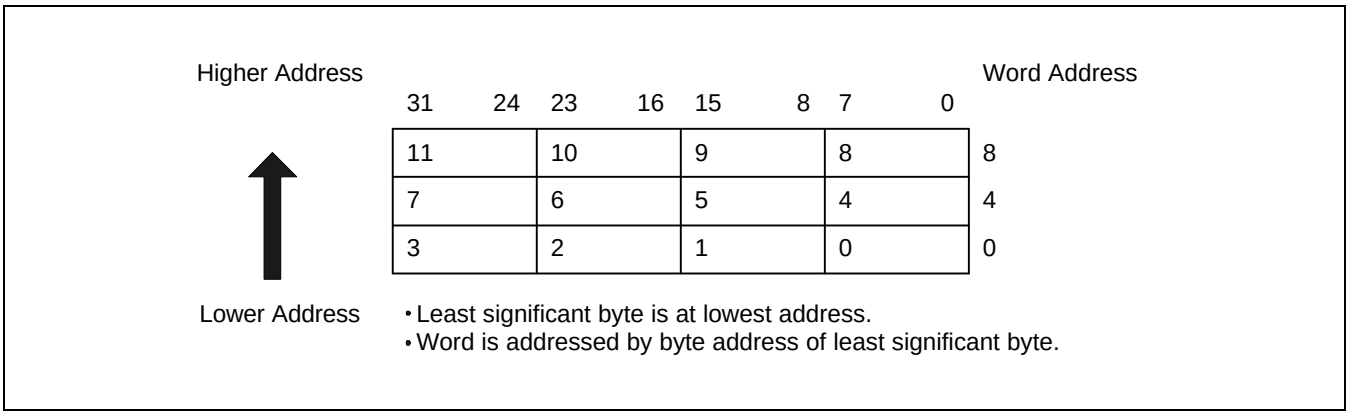


Figure 2-2. Little-Endian Addresses of Bytes within Words

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.

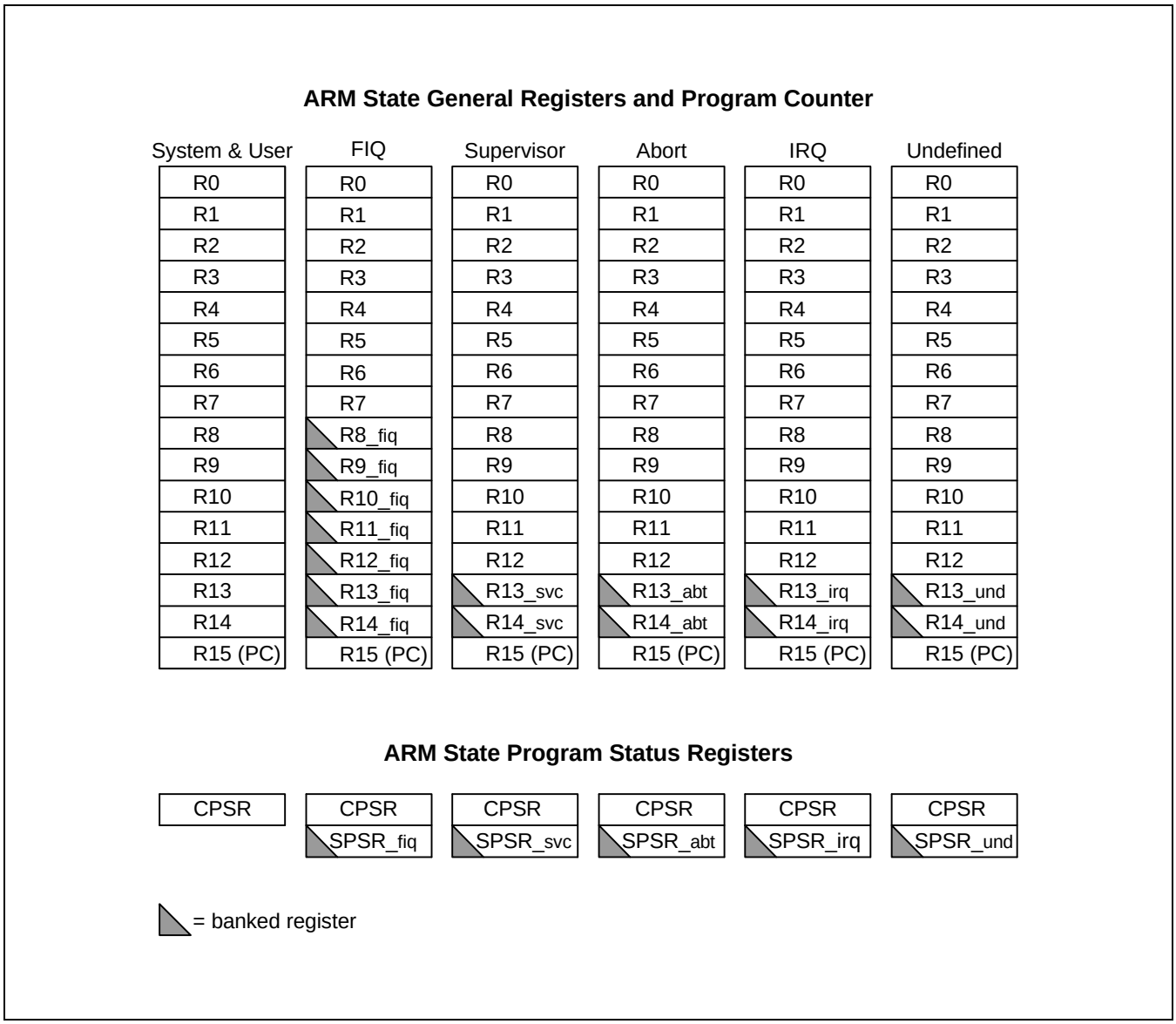


Figure 2-3. Register Organization in ARM State

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.

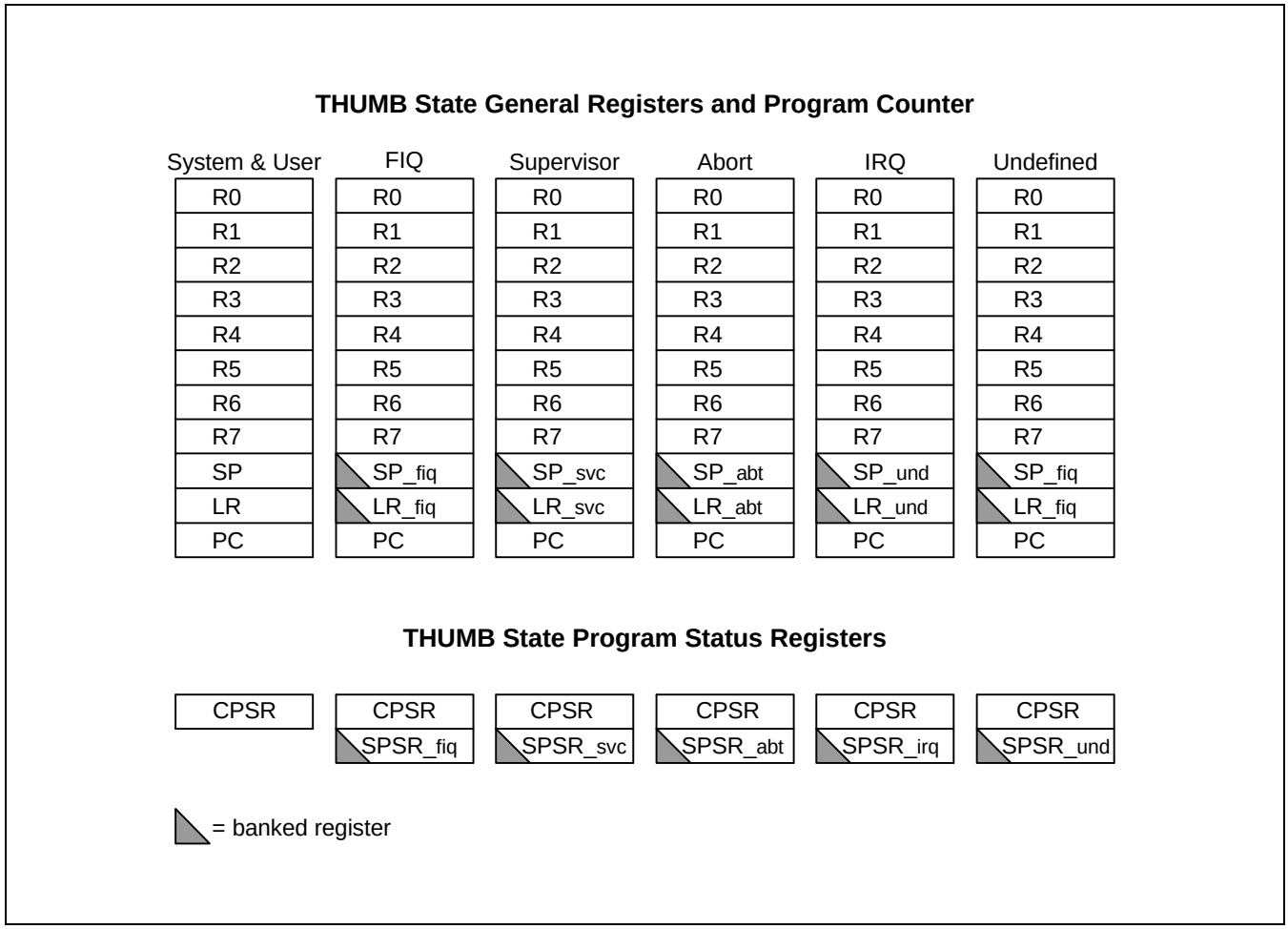


Figure 2-4. Register Organization in THUMB state

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.

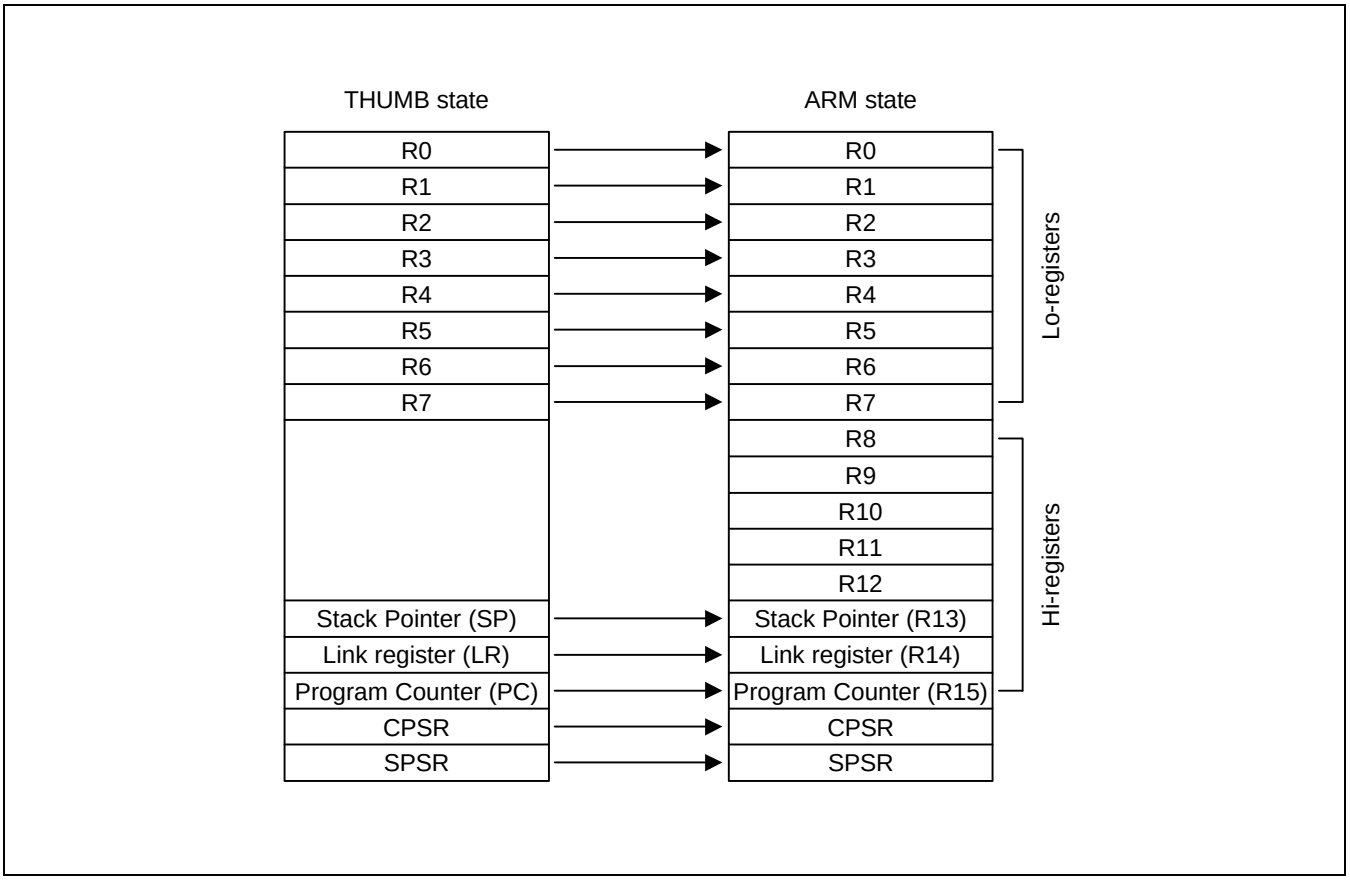


Figure 2-5. Mapping of THUMB State Registers onto ARM State Registers

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.

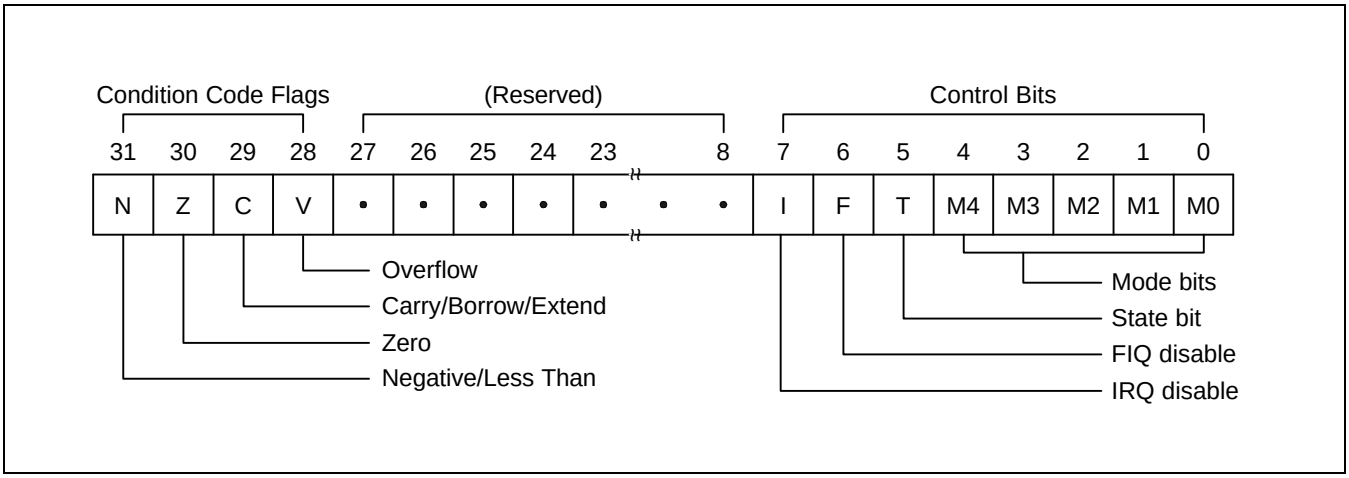


Figure 2-6. Program Status Register Format

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

NOTES

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.

NOTES

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.

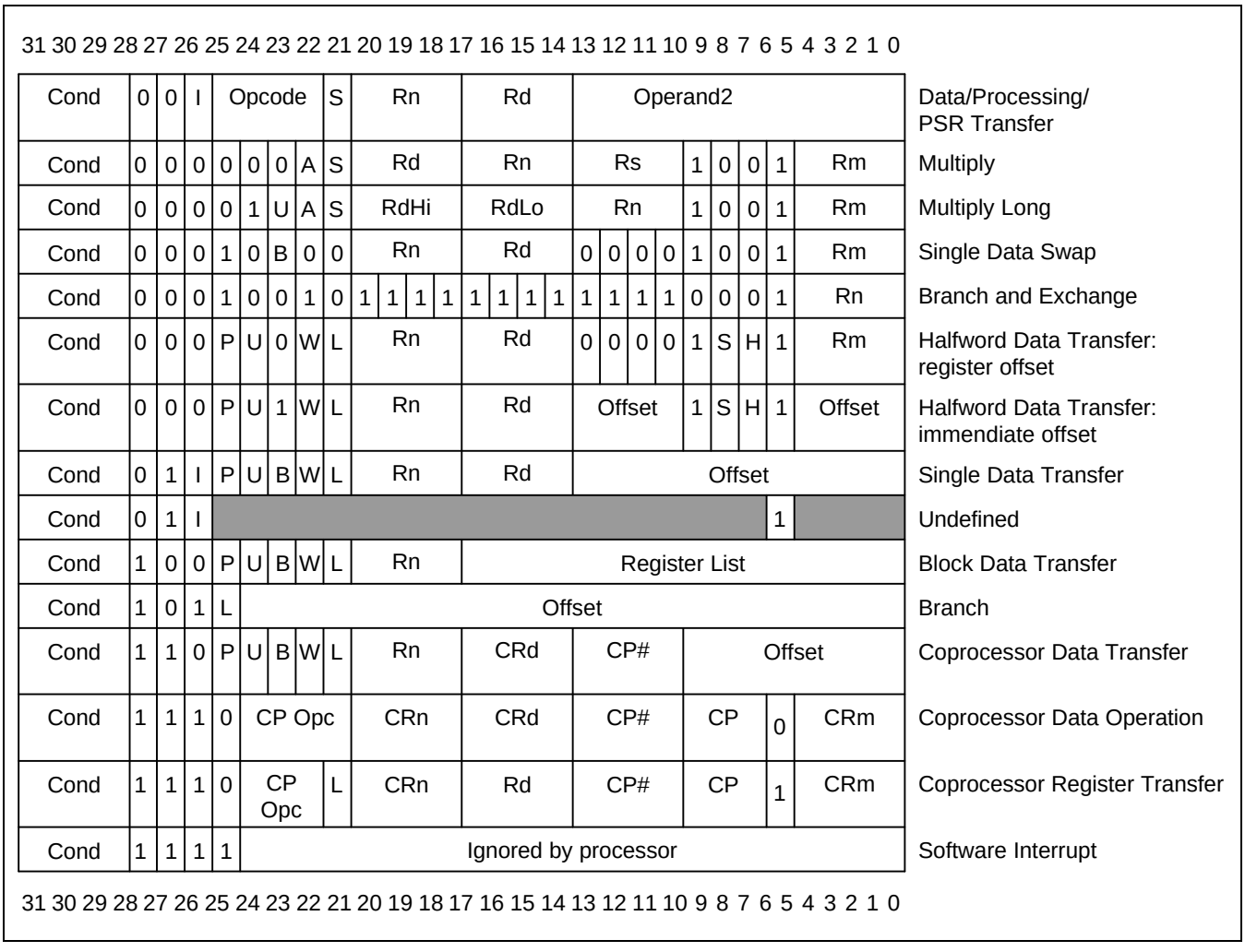


Figure 3-1. ARM Instruction Set Format

NOTES

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.

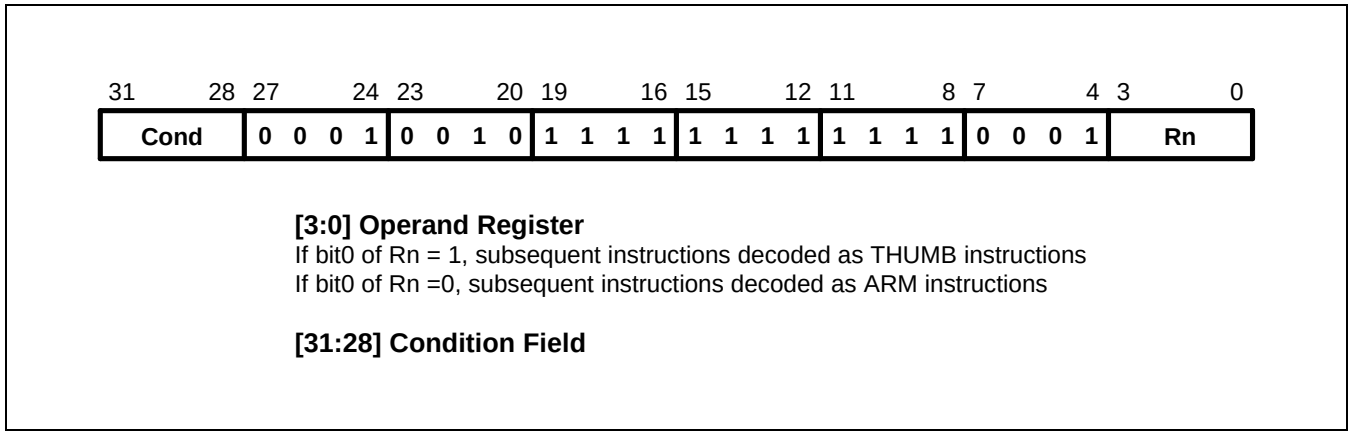


Figure 3-2. Branch and Exchange Instructions

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.

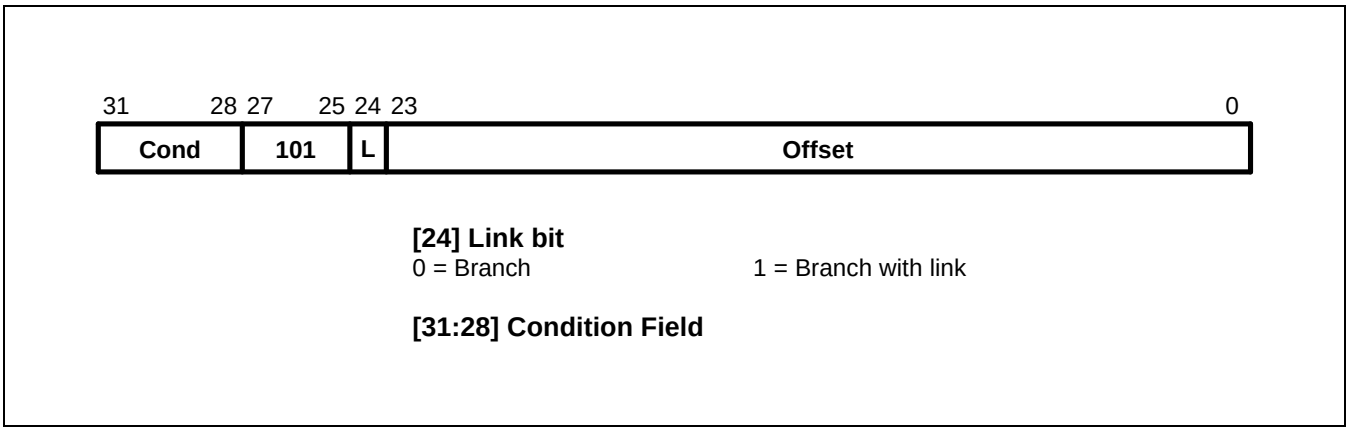


Figure 3-3. Branch Instructions

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>

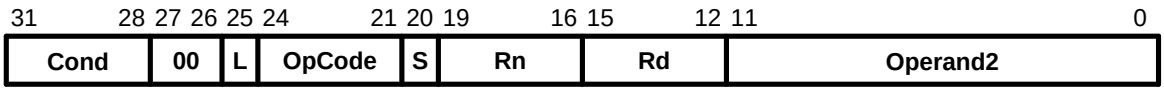
{L}	Used to request the Branch with Link form of the instruction. If absent, R14 will not be affected by the instruction.
{cond}	A two-character mnemonic as shown in Table 3-2. If absent then AL (ALways) will be used.
<expression>	The destination. The assembler calculates the offset.

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

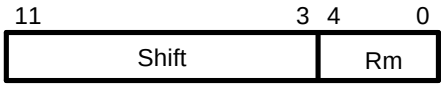
[24:21] Operation codes

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

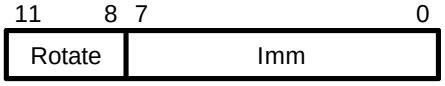
[25] Immediate operand

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.

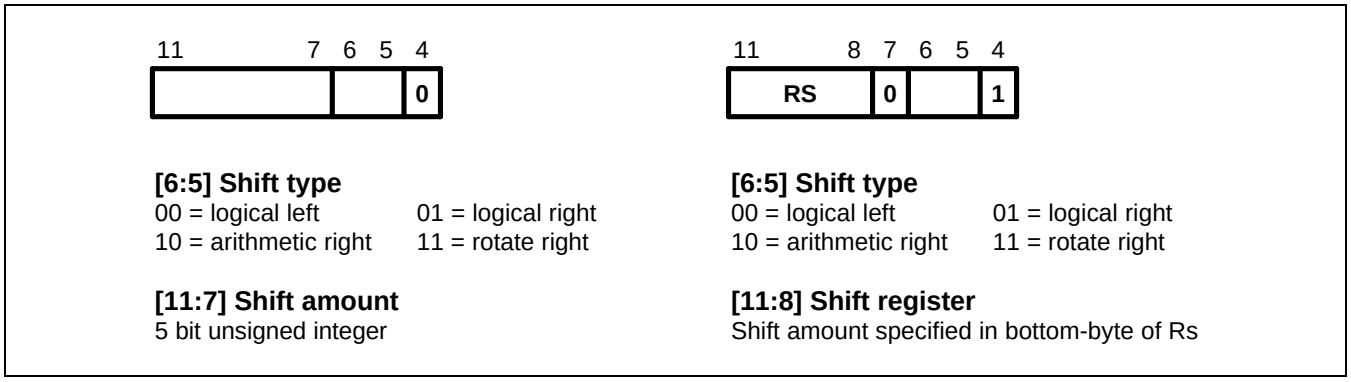


Figure 3-5. ARM Shift Operations

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.

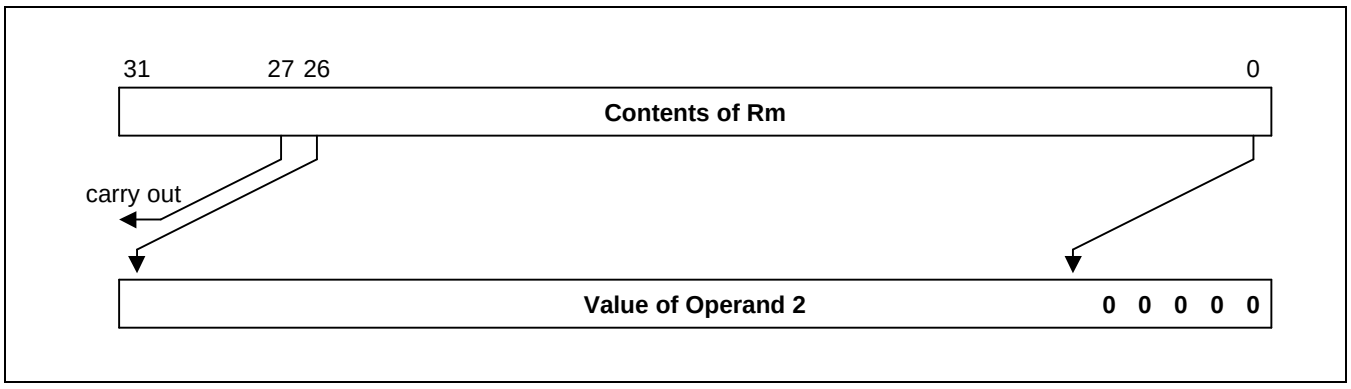


Figure 3-6. Logical Shift Left

NOTES

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.

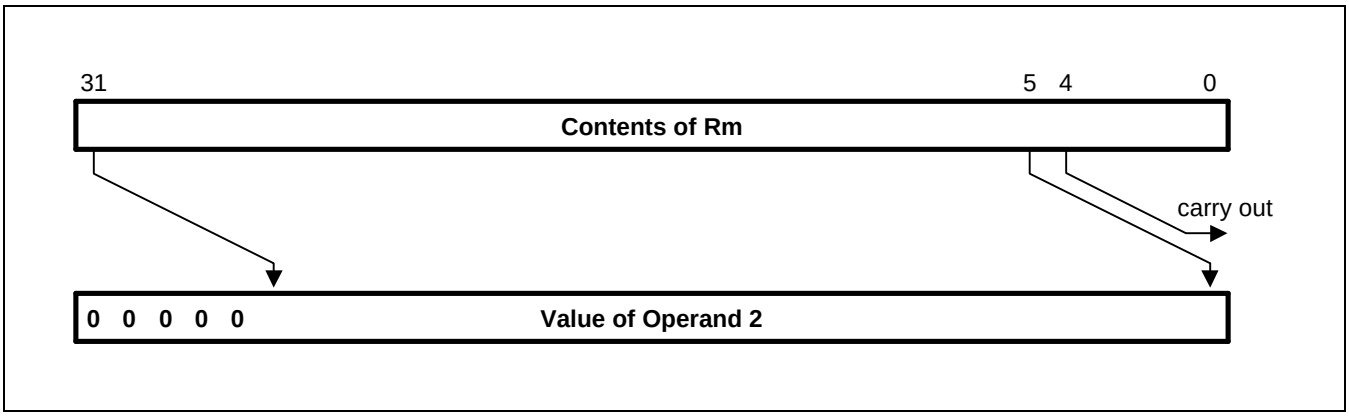


Figure 3-7. Logical Shift Right

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.

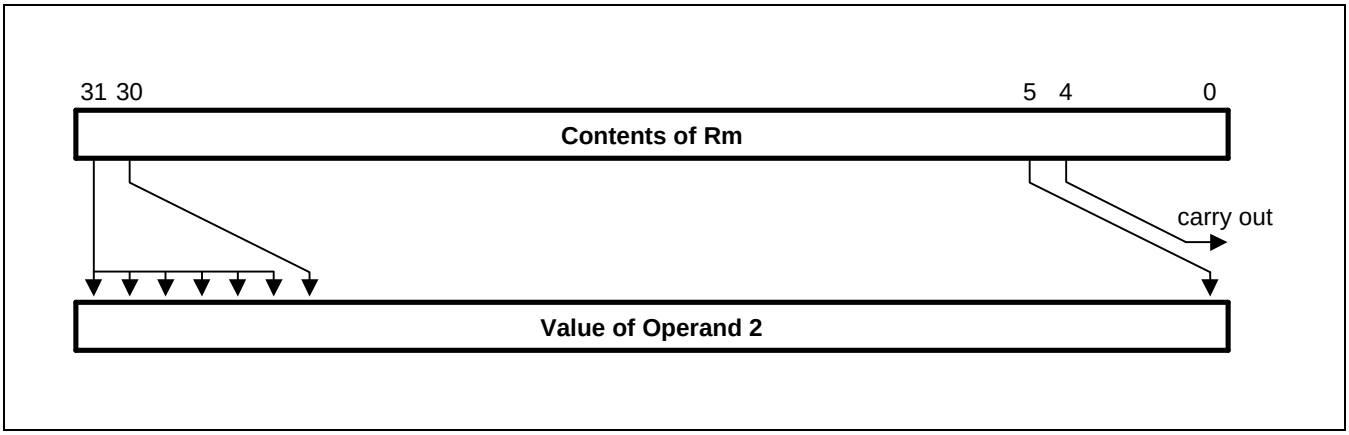


Figure 3-8. Arithmetic Shift Right

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.

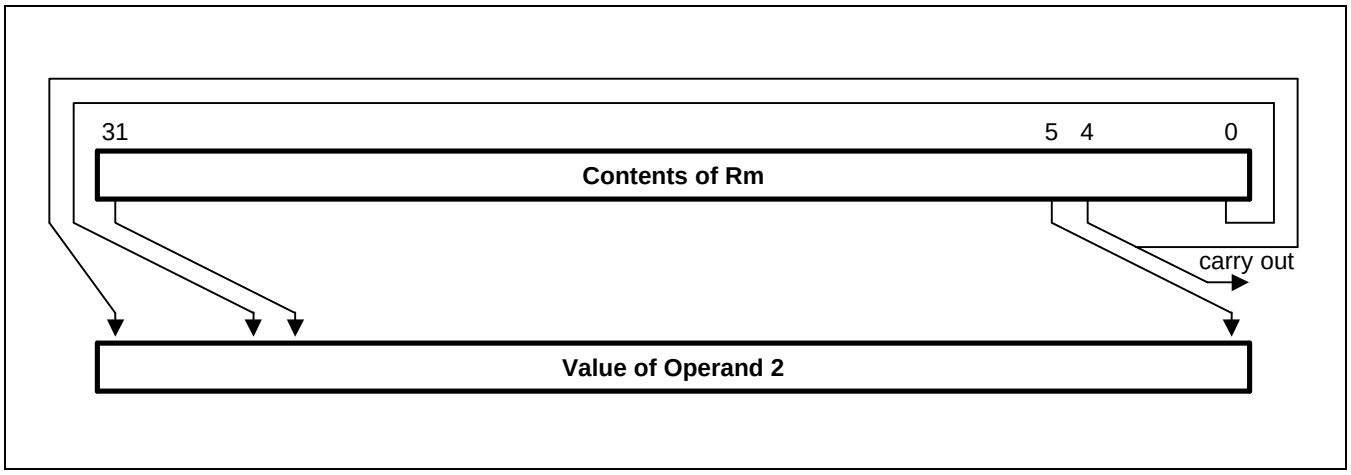


Figure 3-9. Rotate Right

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.

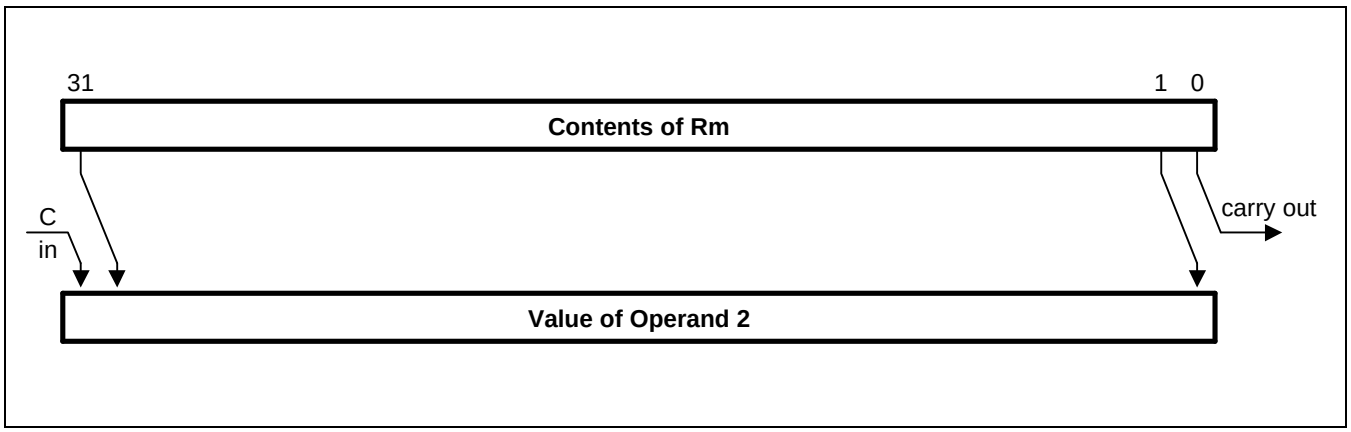


Figure 3-10. Rotate Right Extended

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.

NOTES

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

NOTES

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.

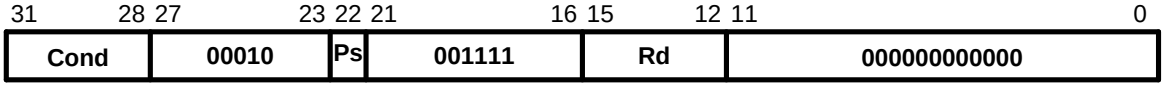
These instructions allow access to the CPSR and SPSR registers. The MRS instruction allows the contents of the CPSR or SPSR_<mode> to be moved to a general register. The MSR instruction allows the contents of a general register to be moved to the CPSR or SPSR_<mode> register.

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)

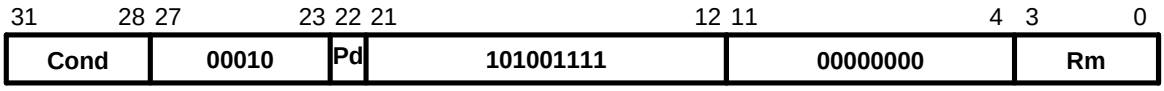


[15:21] Destination Register

[19:16] Source PSR
0 = CPSR 1 = SPSR_<current mode>

[31:28] Condition Field

MRS (transfer register contents to PSR)

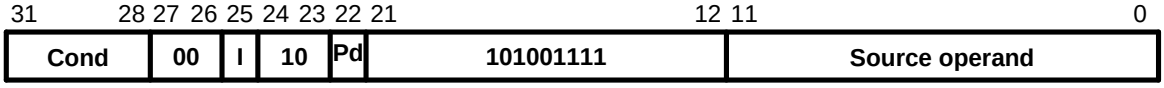


[3:0] Source Register

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

[31:28] Condition Field

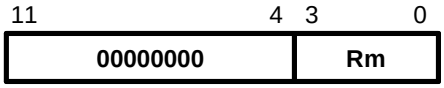
MRS (transfer register contents or immediate value to PSR flag bits only)



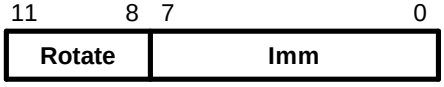
[22] Destination PSR
0 = CPSR 1 = SPSR_<current mode>

[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> are SPSR	CPSR, CPSR_all, SPSR or SPSR_all. (CPSR and CPSR_all are synonyms as and SPSR_all)
<psrf>	CPSR_flg or SPSR_flg
<#expression> 8-bit field	Where this is used, the assembler will attempt to generate a shifted immediate 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.

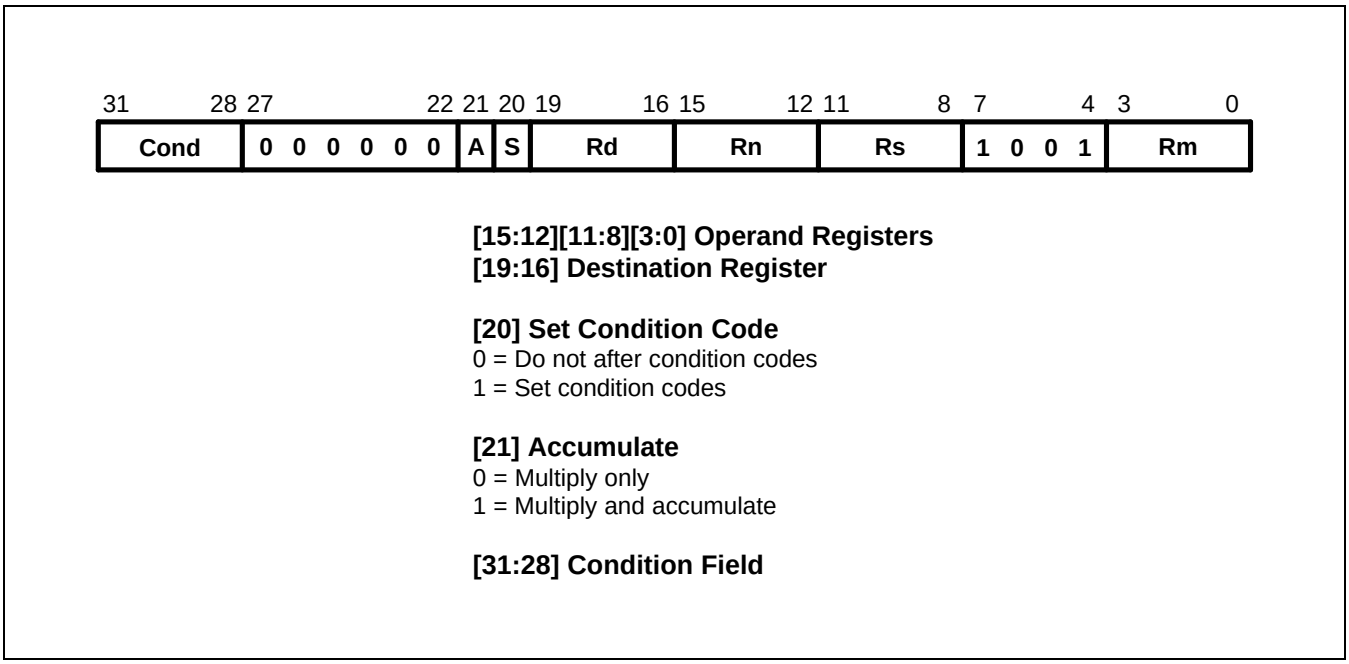


Figure 3-12. Multiply Instructions

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.

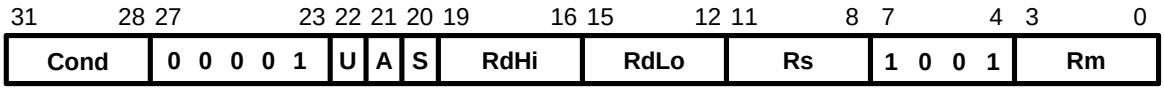
EXAMPLES

```
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

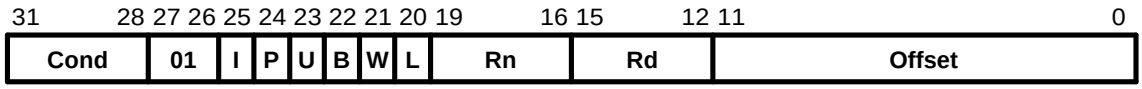
```
UMULL   R1,R4,R2,R3     ; R4,R1:=R2*R3
UMLALS  R1,R5,R2,R3     ; R5,R1:=R2*R3+R5,R1 also setting condition codes
```

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

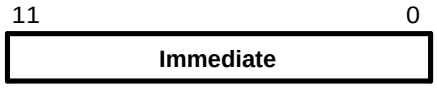
[22] Byte/Word Bit
0 = Transfer word quantity
1 = Transfer byte quantity

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

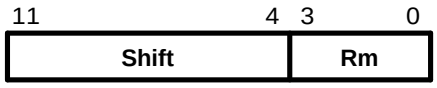
[24] Pre/Post Indexing Bit
0 = Post: add offset after transfer
1 = Pre: add offset before transfer

[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.

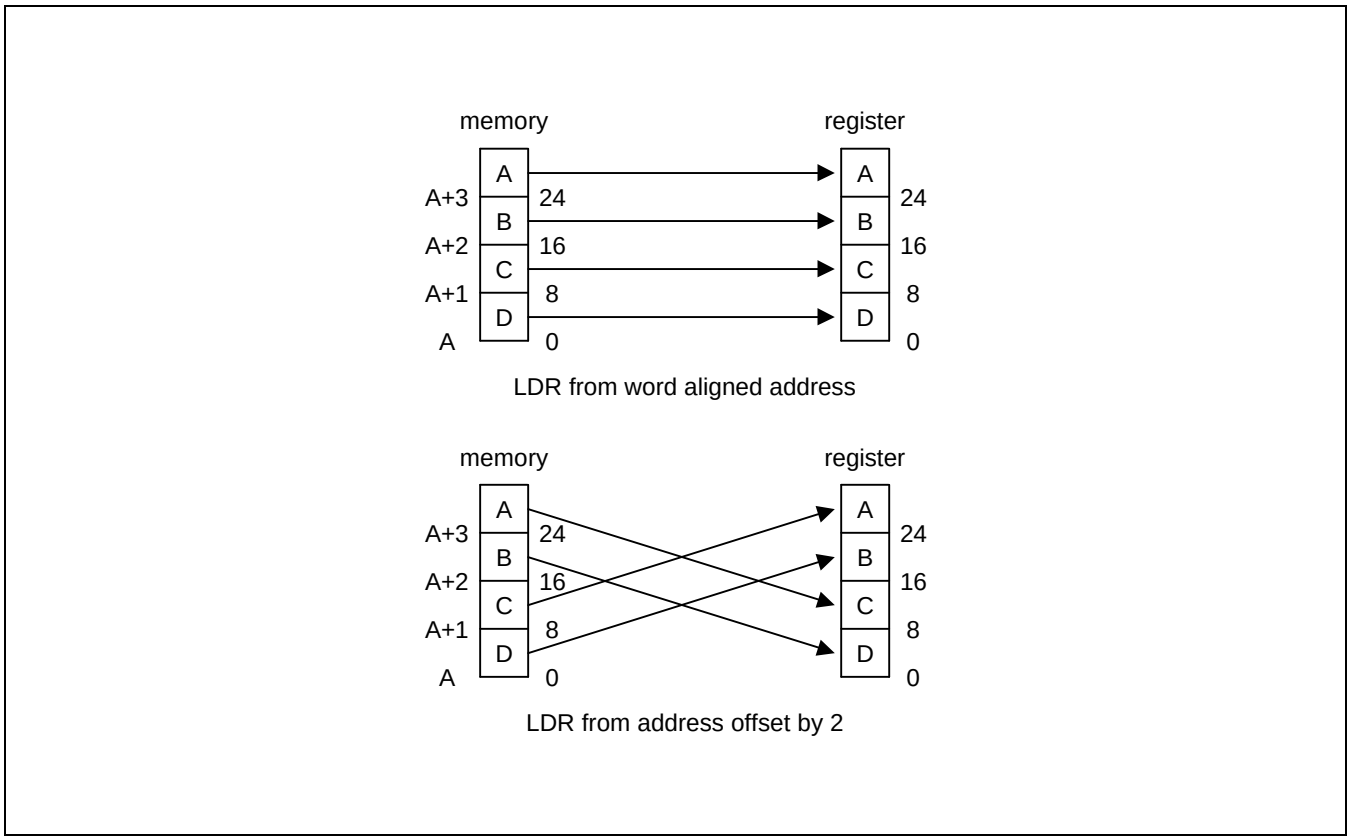


Figure 3-15. Little-Endian Offset Addressing

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.

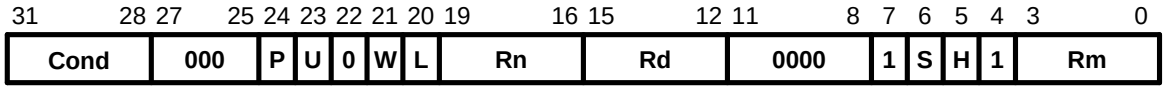
EXAMPLES

```
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

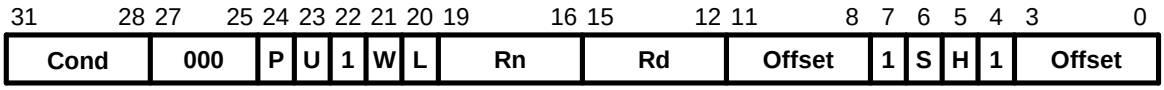
[21] Write-back
0 = No write-back
1 = Write address into base

[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.

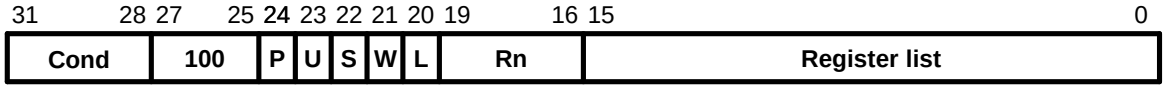
Block data transfer instructions are used to load (LDM) or store (STM) any subset of the currently visible registers. They support all possible stacking modes, maintaining full or empty stacks which can grow up or down memory, and are very efficient instructions for saving or restoring context, or for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

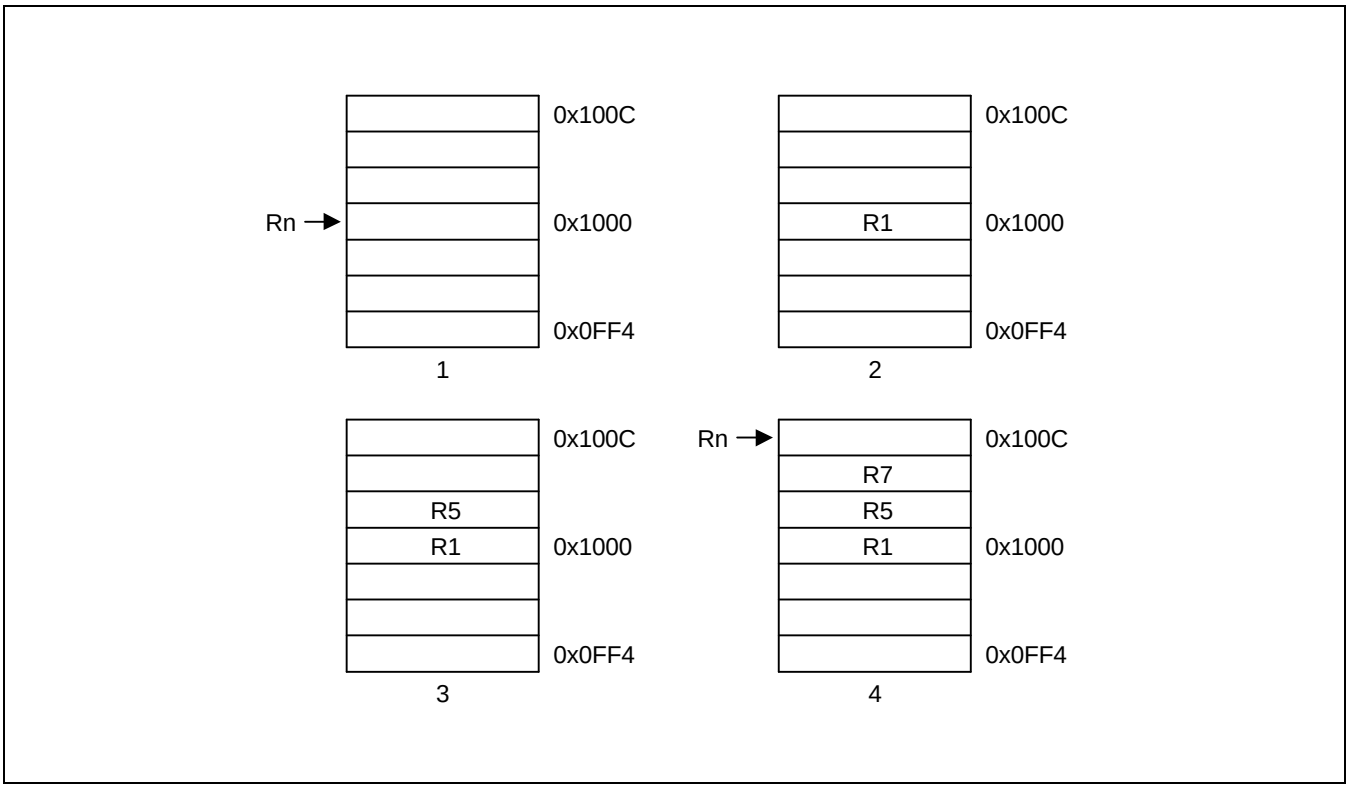


Figure 3-19. Post-Increment Addressing

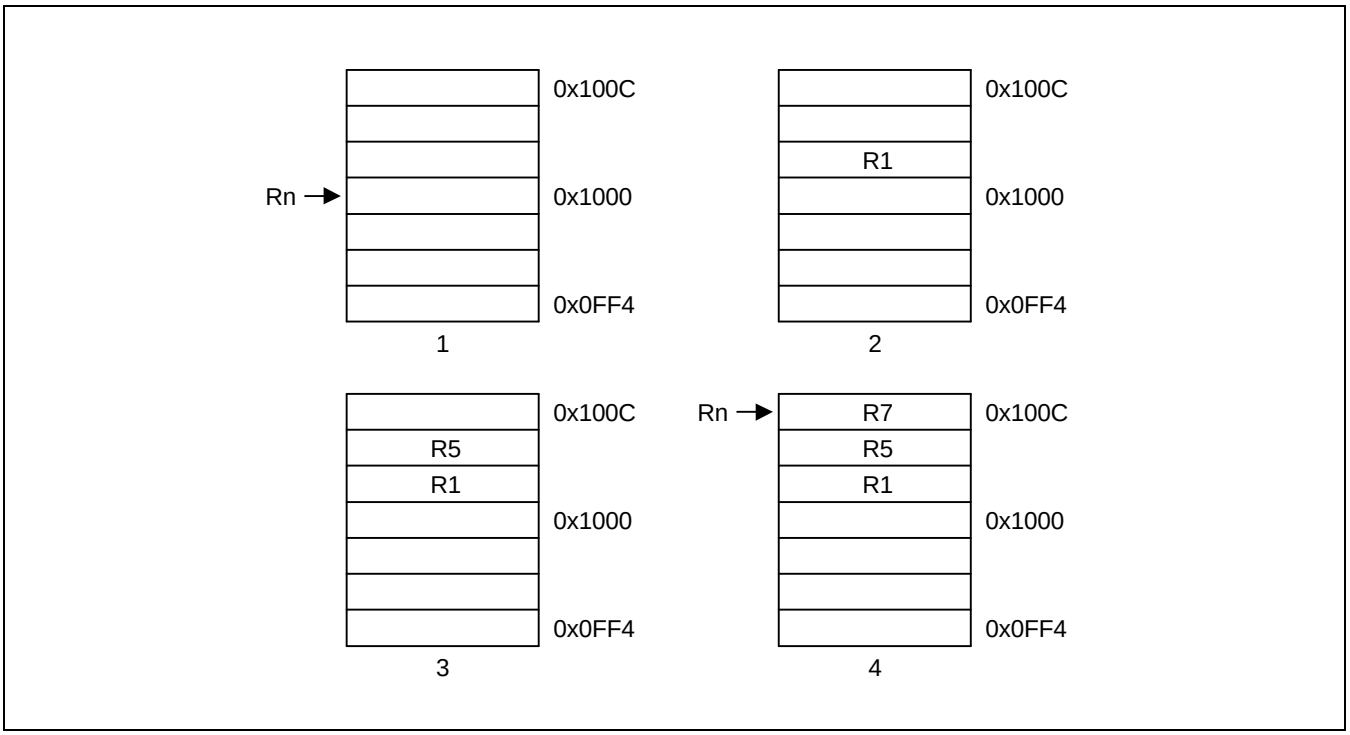


Figure 3-20. Pre-Increment Addressing

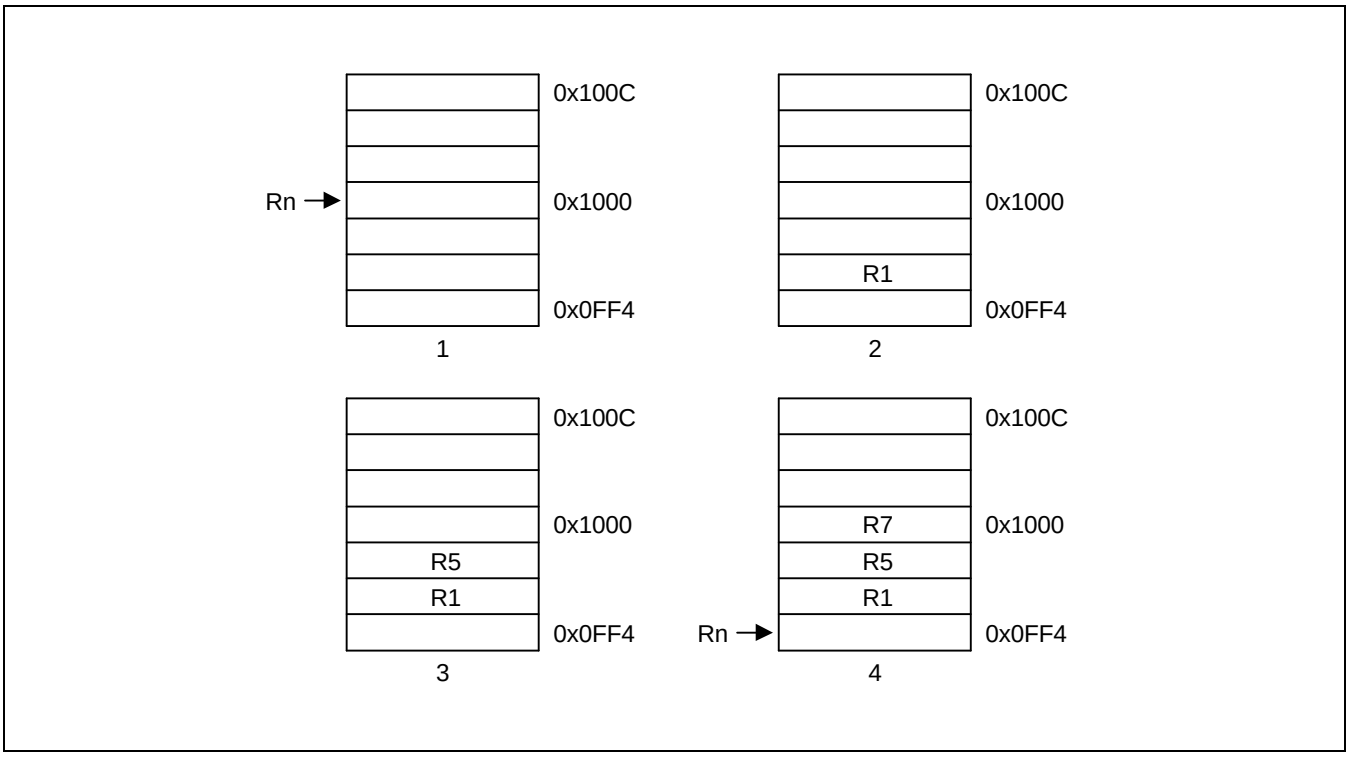


Figure 3-21. Post-Decrement Addressing

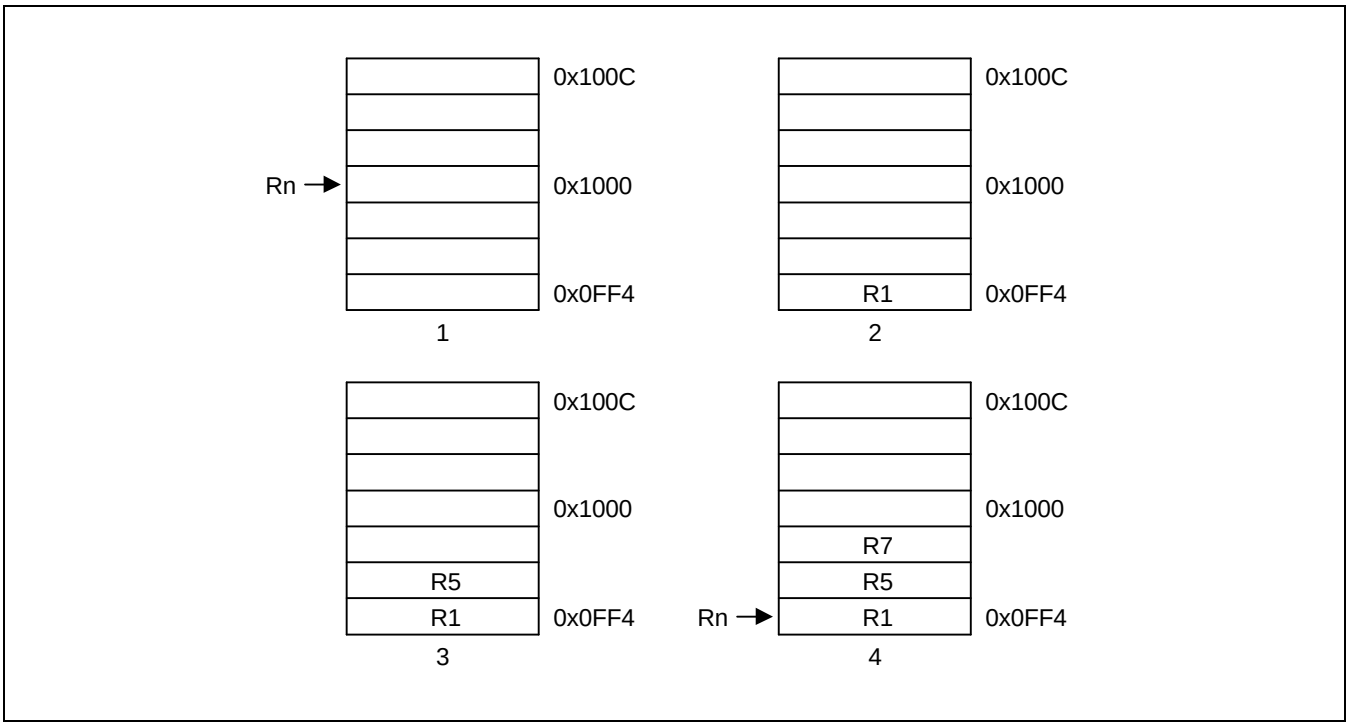


Figure 3-22. Pre-Decrement Addressing

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)

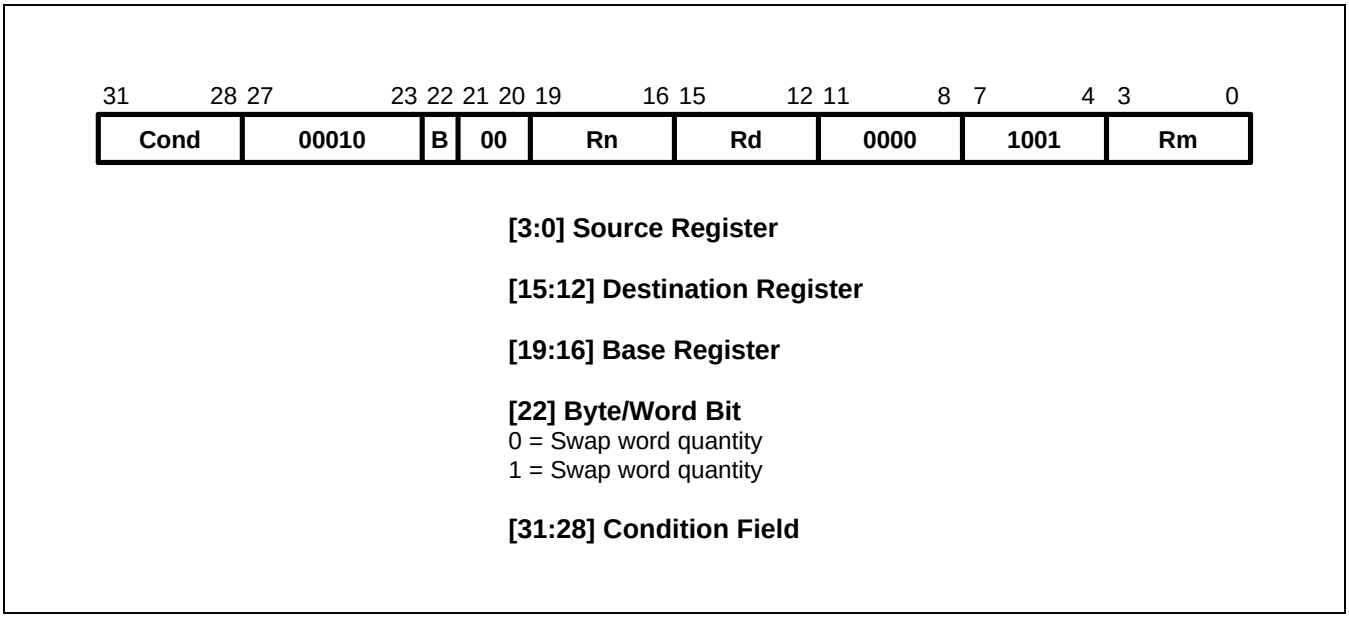


Figure 3-23. Swap Instruction

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.

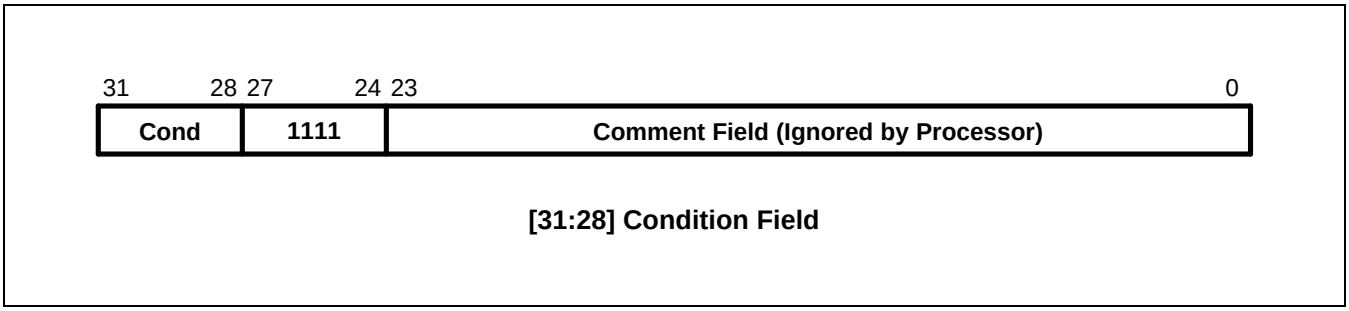


Figure 3-24. Software Interrupt Instruction

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 S3C44B0X, 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 S3C44B0X. These coprocessor instructions can be emulated by the undefined trap handler. Even though external coprocessor can not be connected to the S3C44B0X, the coprocessor instructions are still described here in full for completeness. (Remember that any external coprocessor described in this section is a software emulation.)

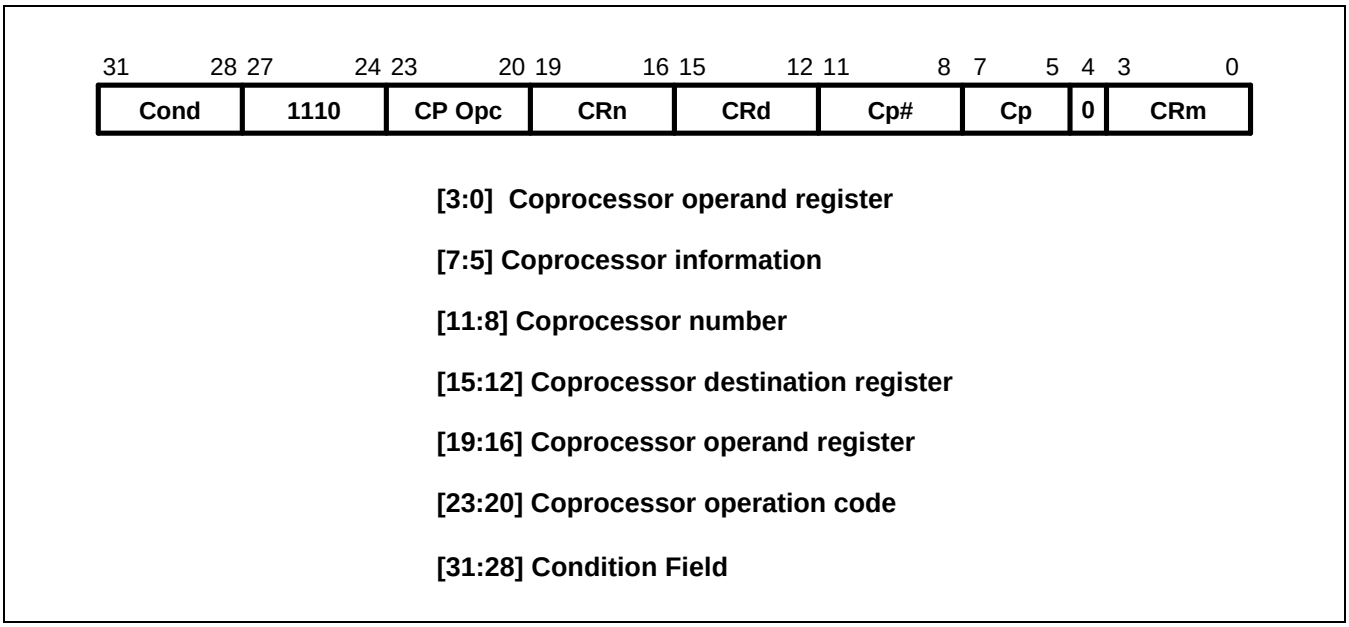


Figure 3-25. Coprocessor Data Operation Instruction

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

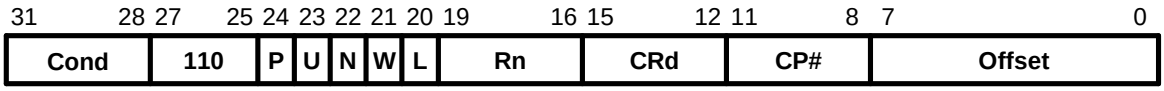
EXAMPLES

```
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.

NOTES

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).
```

NOTES

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.

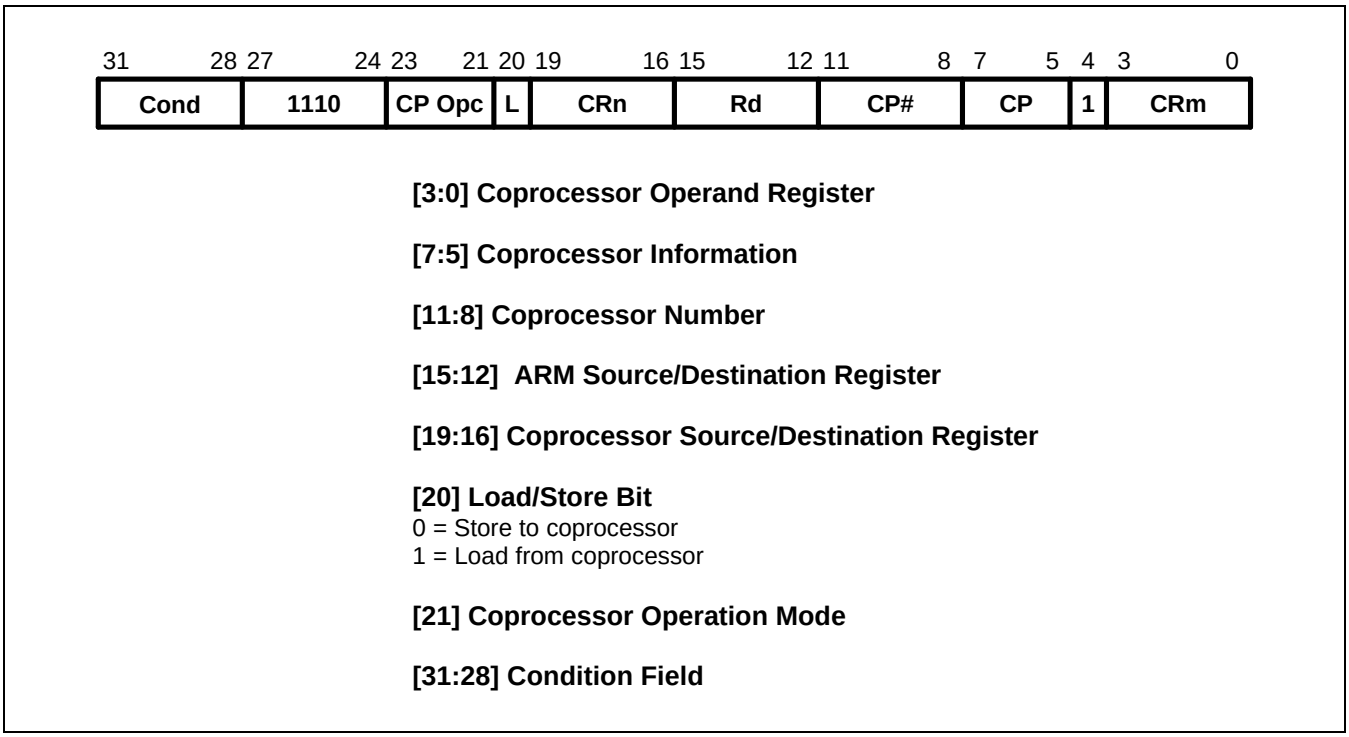


Figure 3-27. Coprocessor Register Transfer Instructions

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.

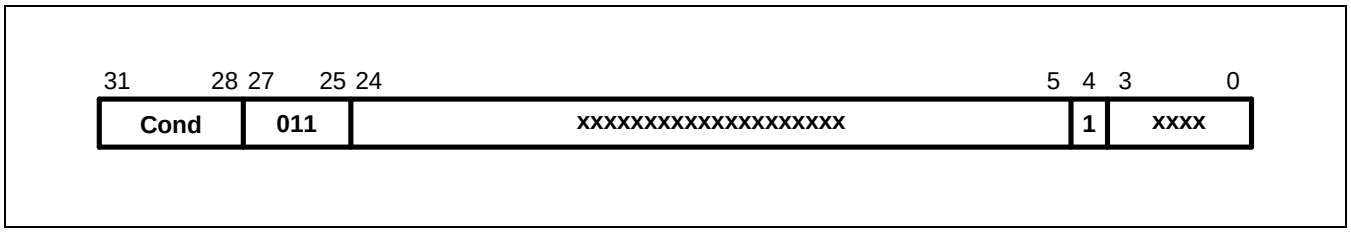


Figure 3-28. Undefined Instruction

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
```

NOTES

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.

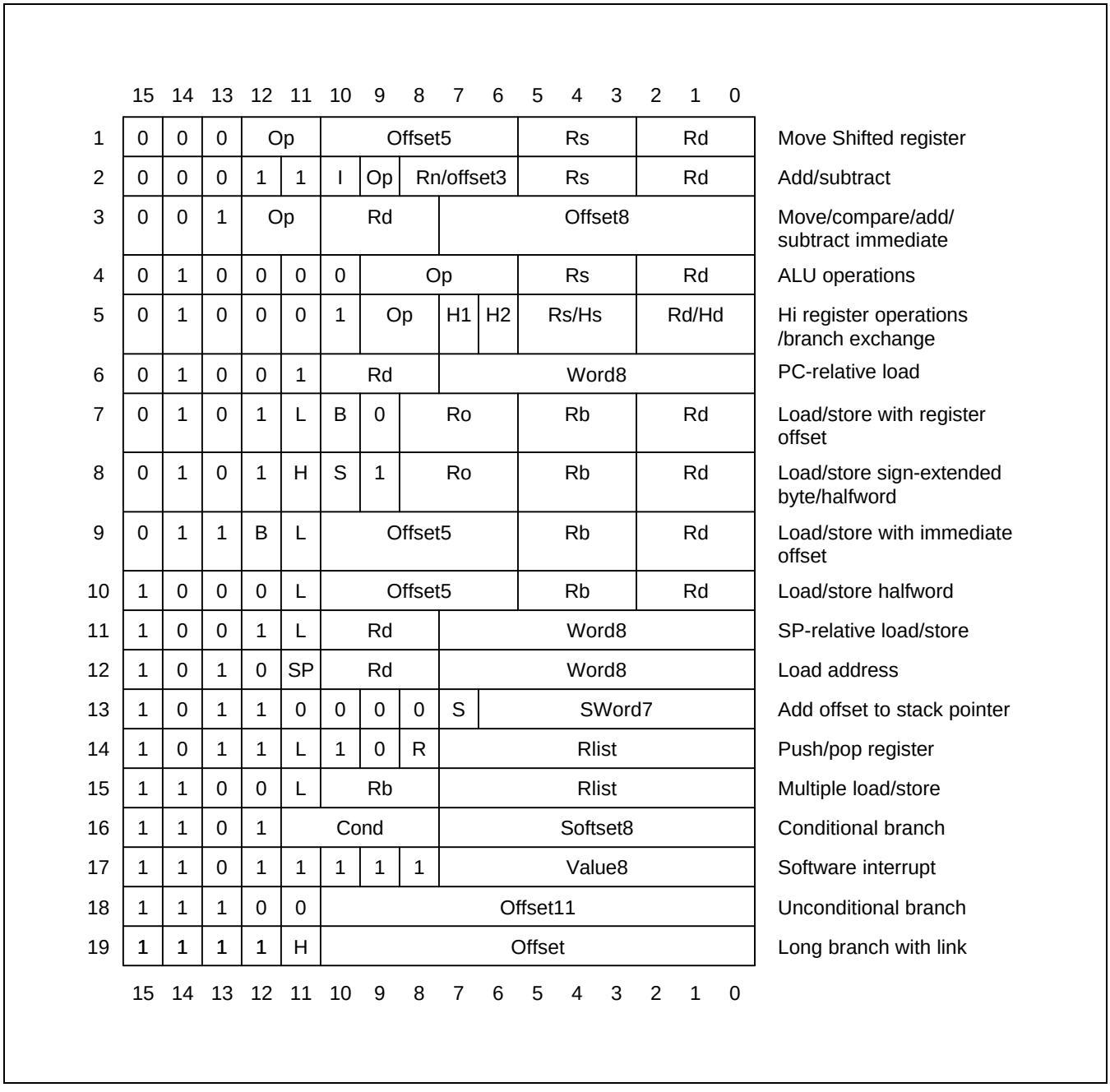


Figure 4-1. THUMB Instruction Set Formats

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

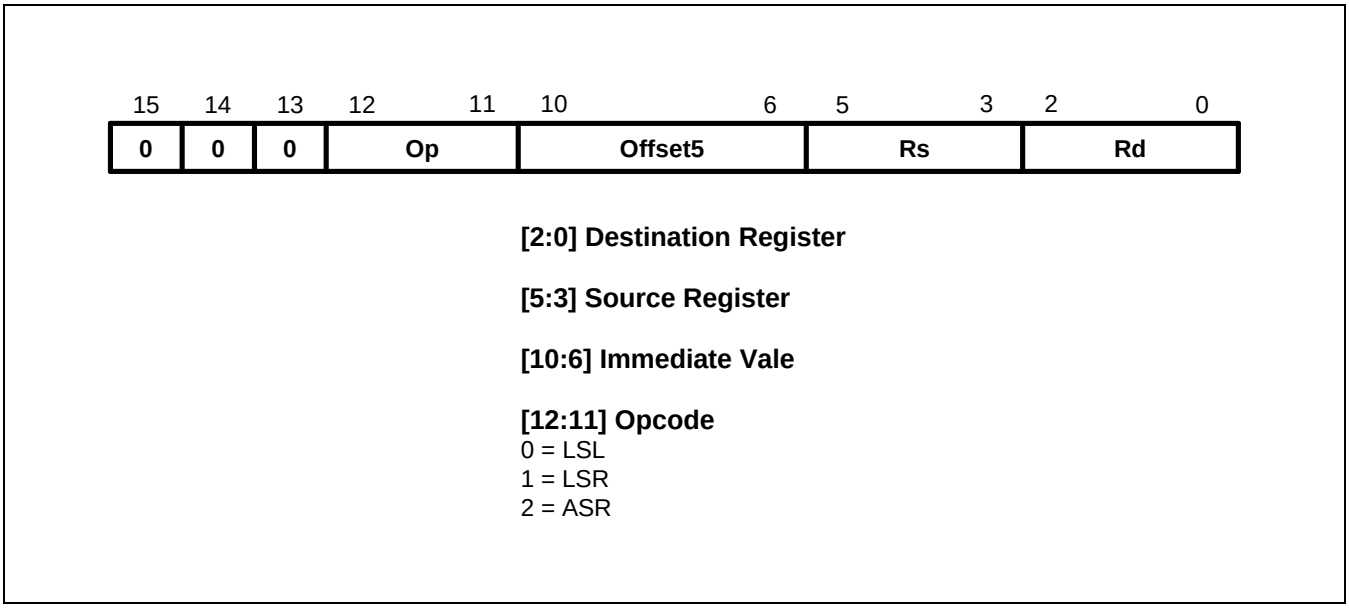


Figure 4-2. Format 1

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

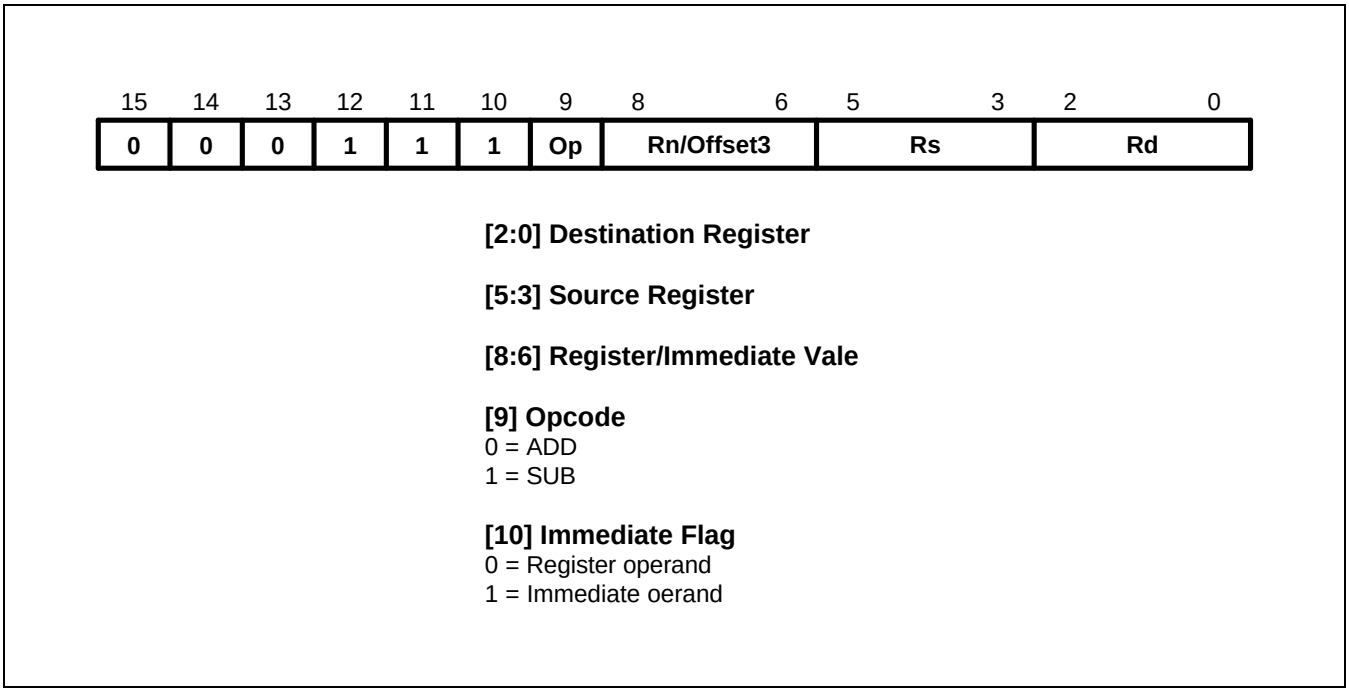


Figure 4-3. Format 2

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

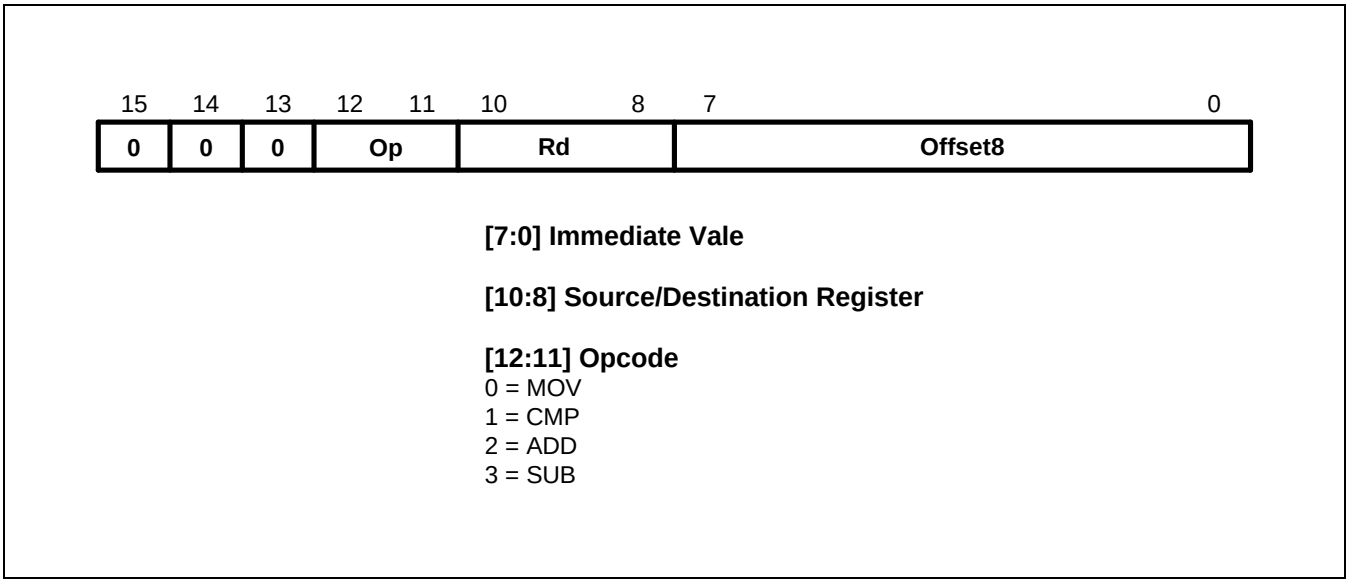


Figure 4-4. Format 3

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.

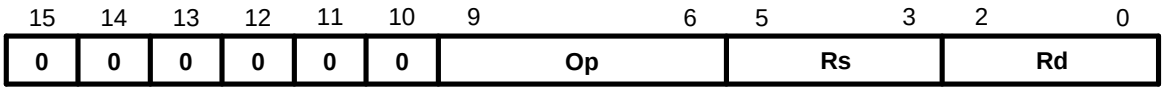
INSTRUCTION CYCLE TIMES

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

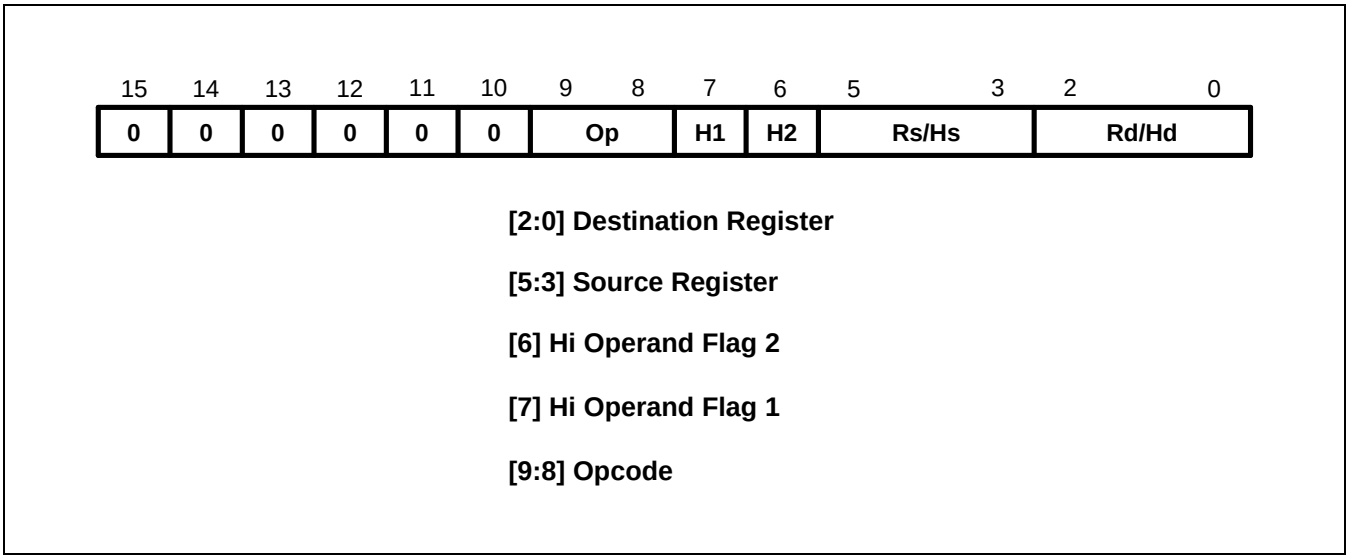


Figure 4-6. Format 5

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.

NOTES

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

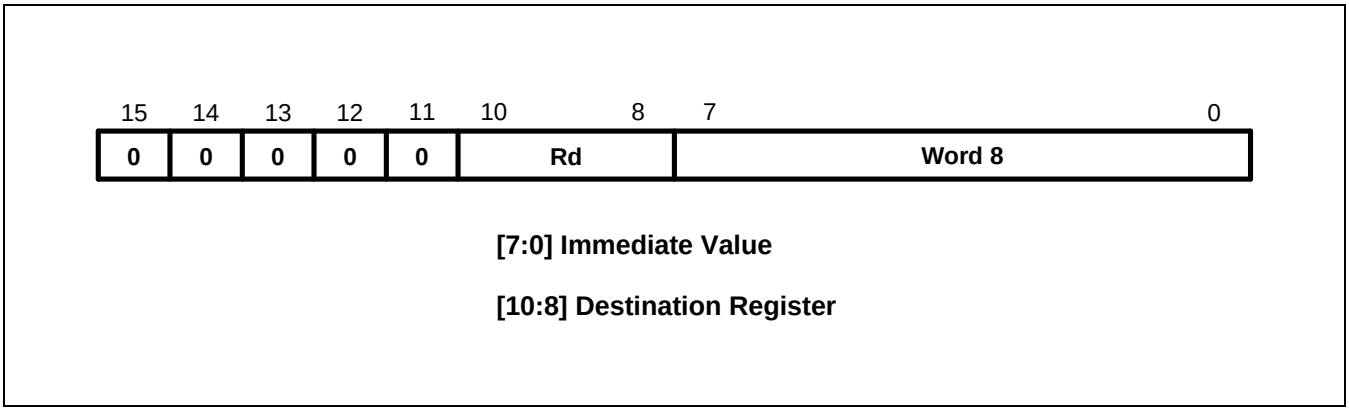


Figure 4-7. Format 6

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.

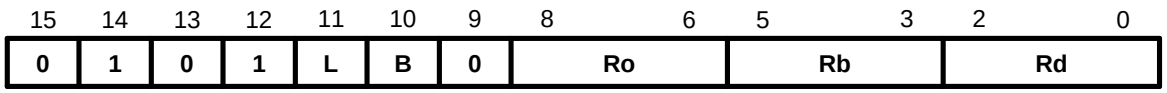
INSTRUCTION CYCLE TIMES

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

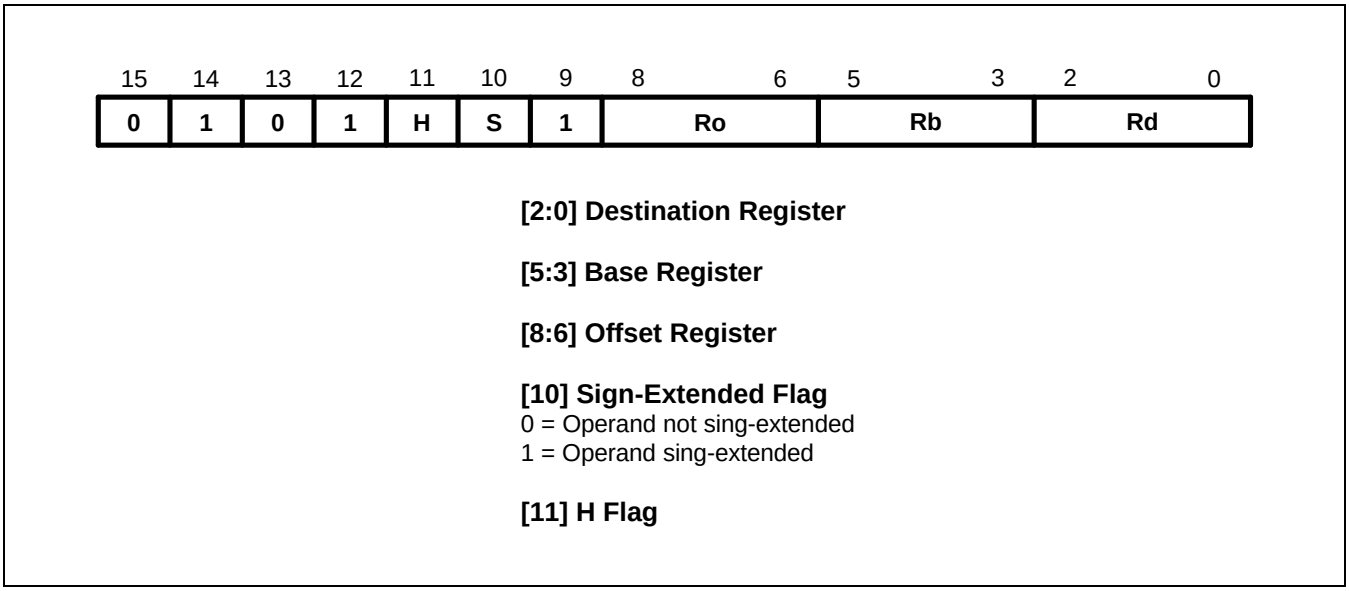


Figure 4-9. Format 8

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.

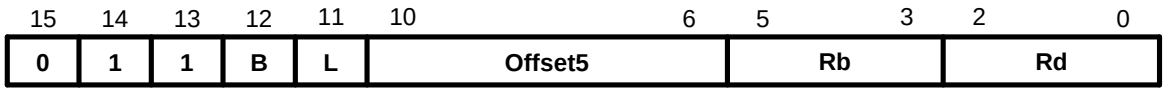
INSTRUCTION CYCLE TIMES

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

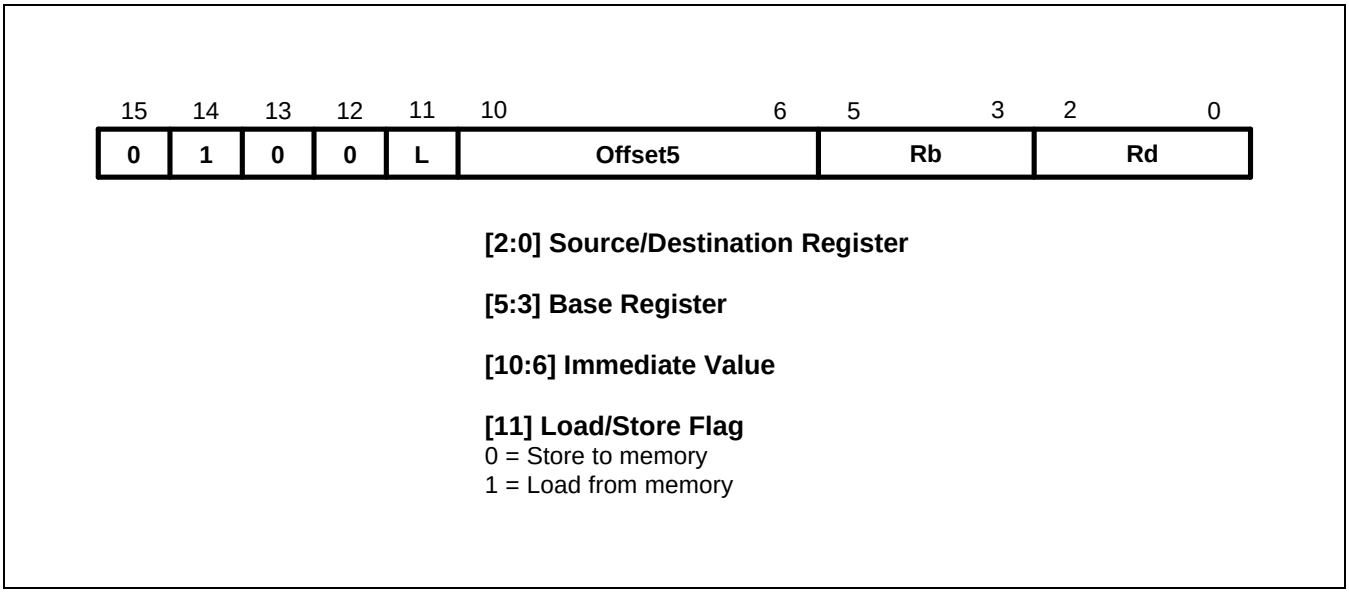


Figure 4-11. Format 10

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

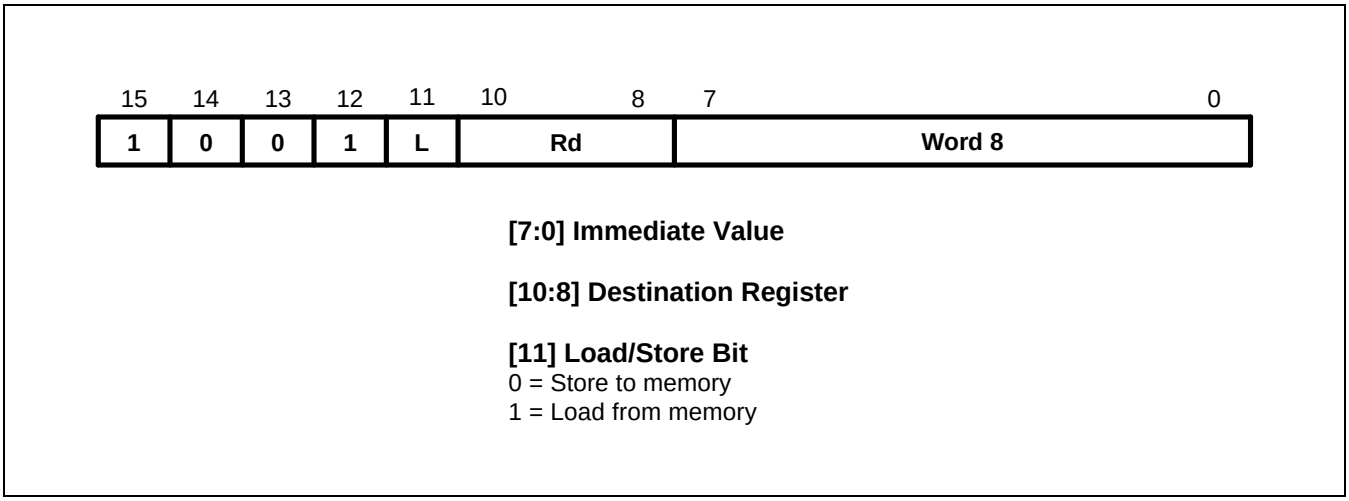


Figure 4-12. Format 11

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

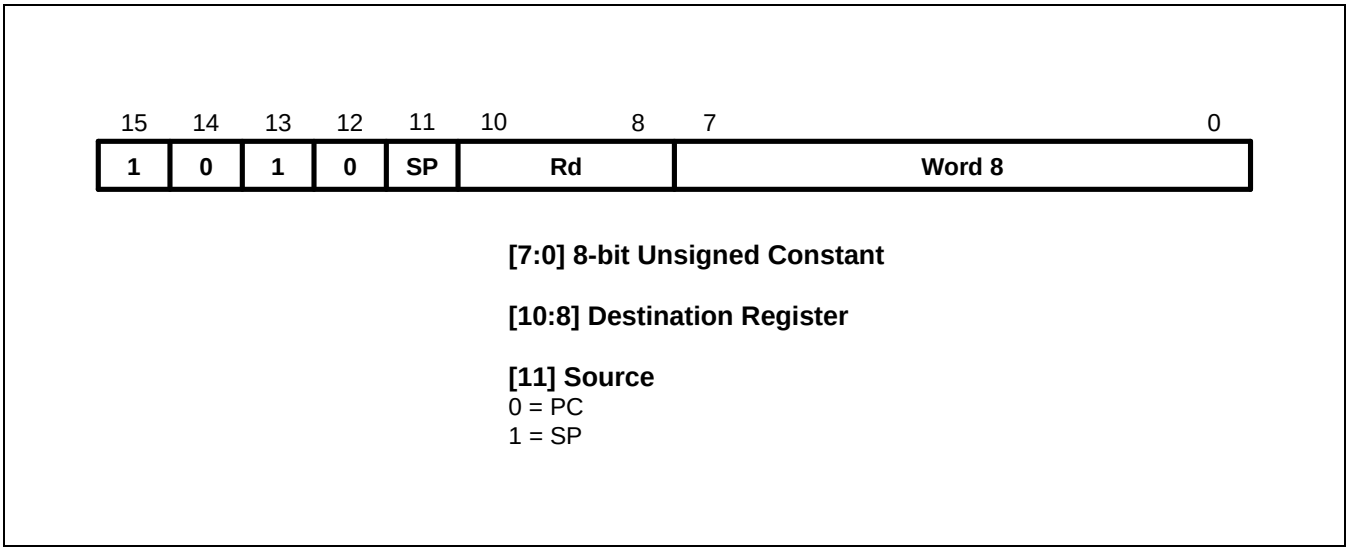


Figure 4-13. Format 12

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

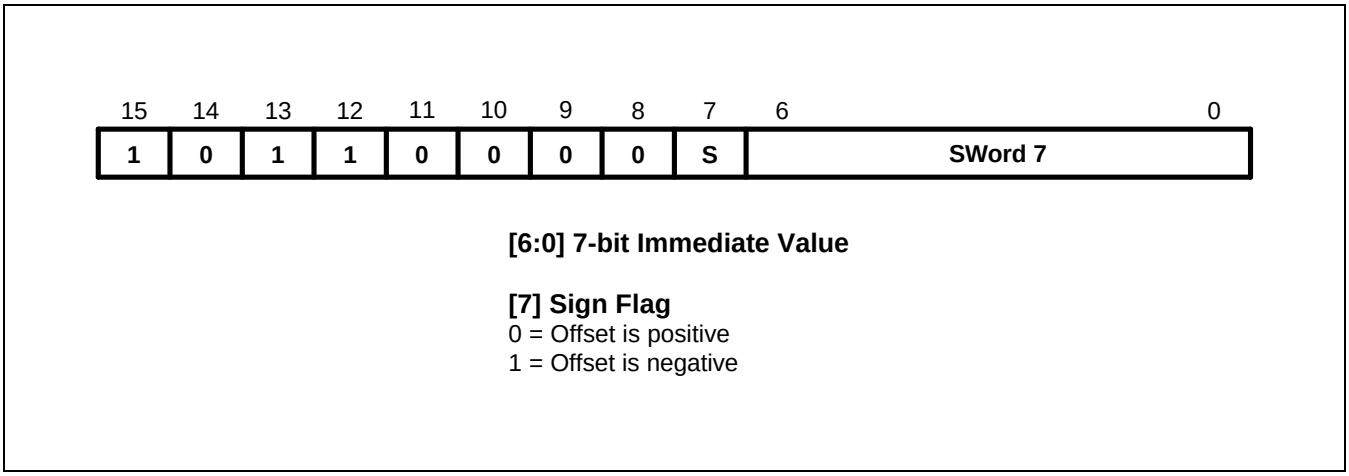


Figure 4-14. Format 13

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.

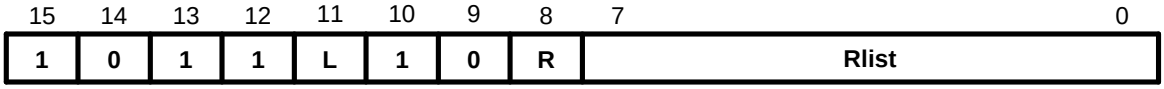
INSTRUCTION CYCLE TIMES

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



[7:0] Register List

[8] PC/LR Bit
0 = Do not store LR/Load PC
1 = Store LR/Load PC

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

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

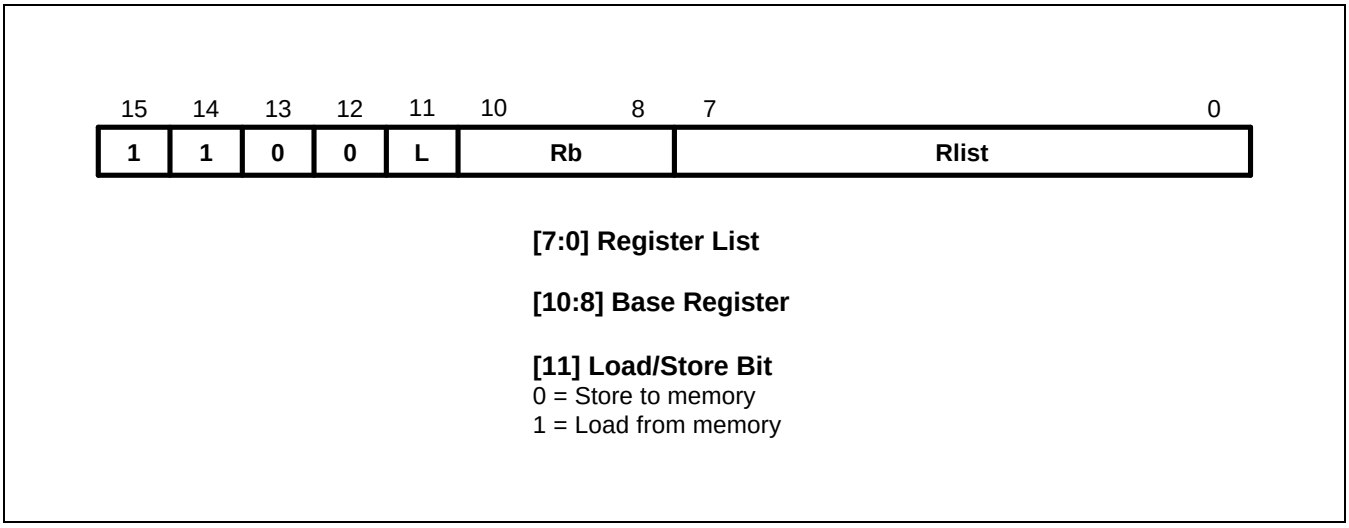


Figure 4-16. Format 15

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

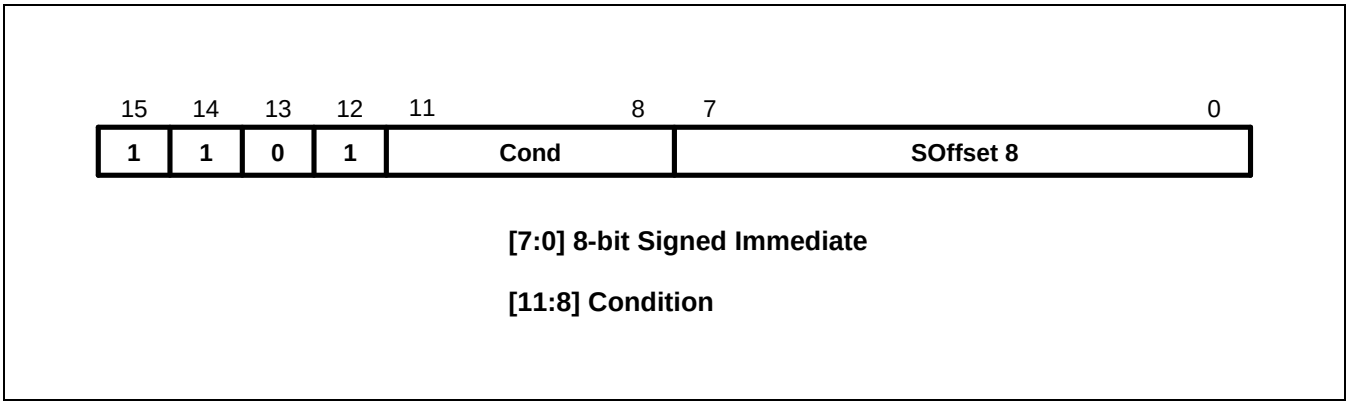


Figure 4-17. Format 16

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

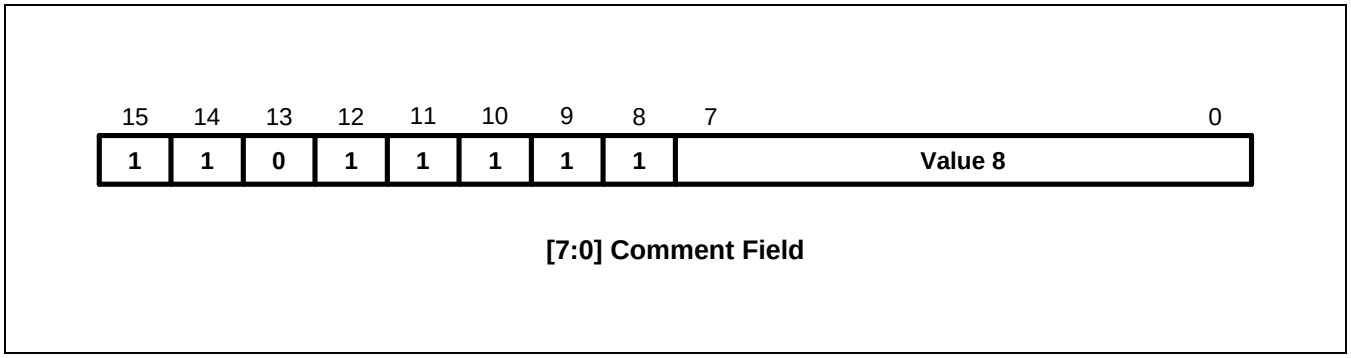


Figure 4-18. Format 17

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

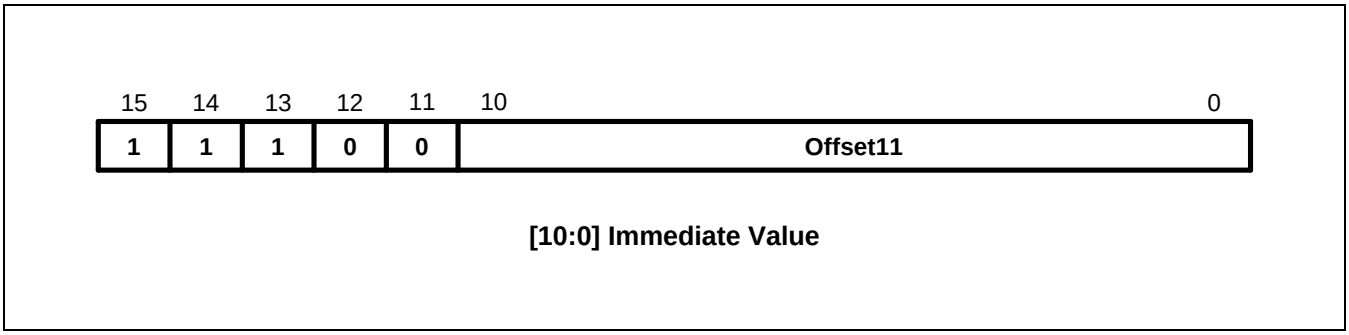


Figure 4-19. Format 18

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

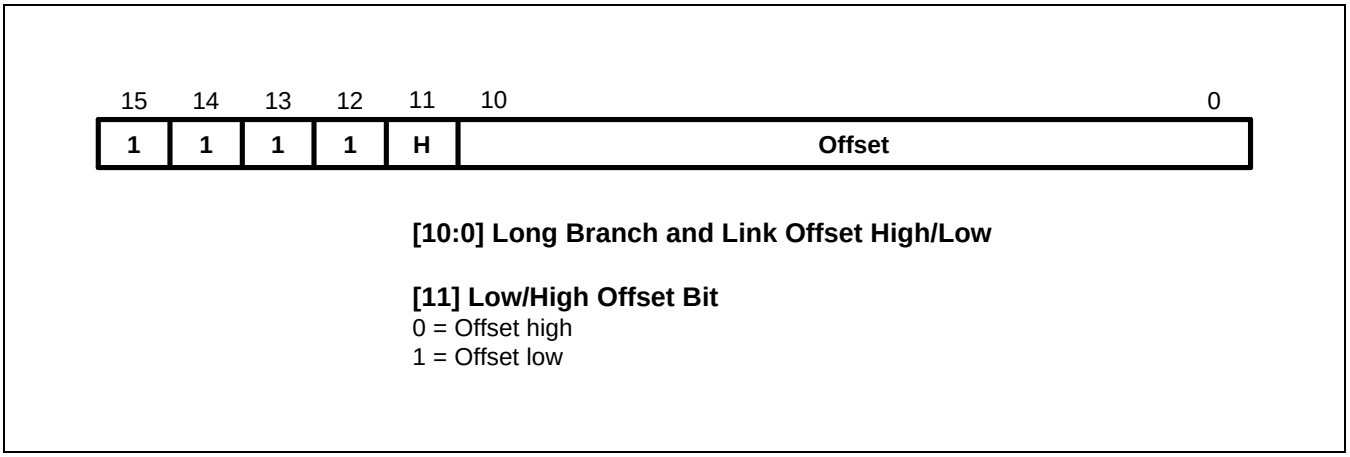


Figure 4-20. Format 19

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 S3C2400X memory controller provides the necessary memory control signals for external memory access. S3C2400X has the following features;

— Little/Big endian (selectable by a S/W)

— Address space: 32Mbytes per each bank (total 256MB: 8 banks)

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

— Total 8 memory banks
6 memory banks for ROM, SRAM etc.
2 memory banks for ROM, SRAM, EDO/SDRAM etc .

— 7 fixed memory bank start address and programmble bank size

— 1 flexible memory bank start address and programmable bank size

— Programmable access cycles for all memory banks

— External wait to extend the bus cycles

— Supports self-refresh mode in DRAM/SDRAM for power-down

— Supports asymmetrically or symmetrically addressable DRAM

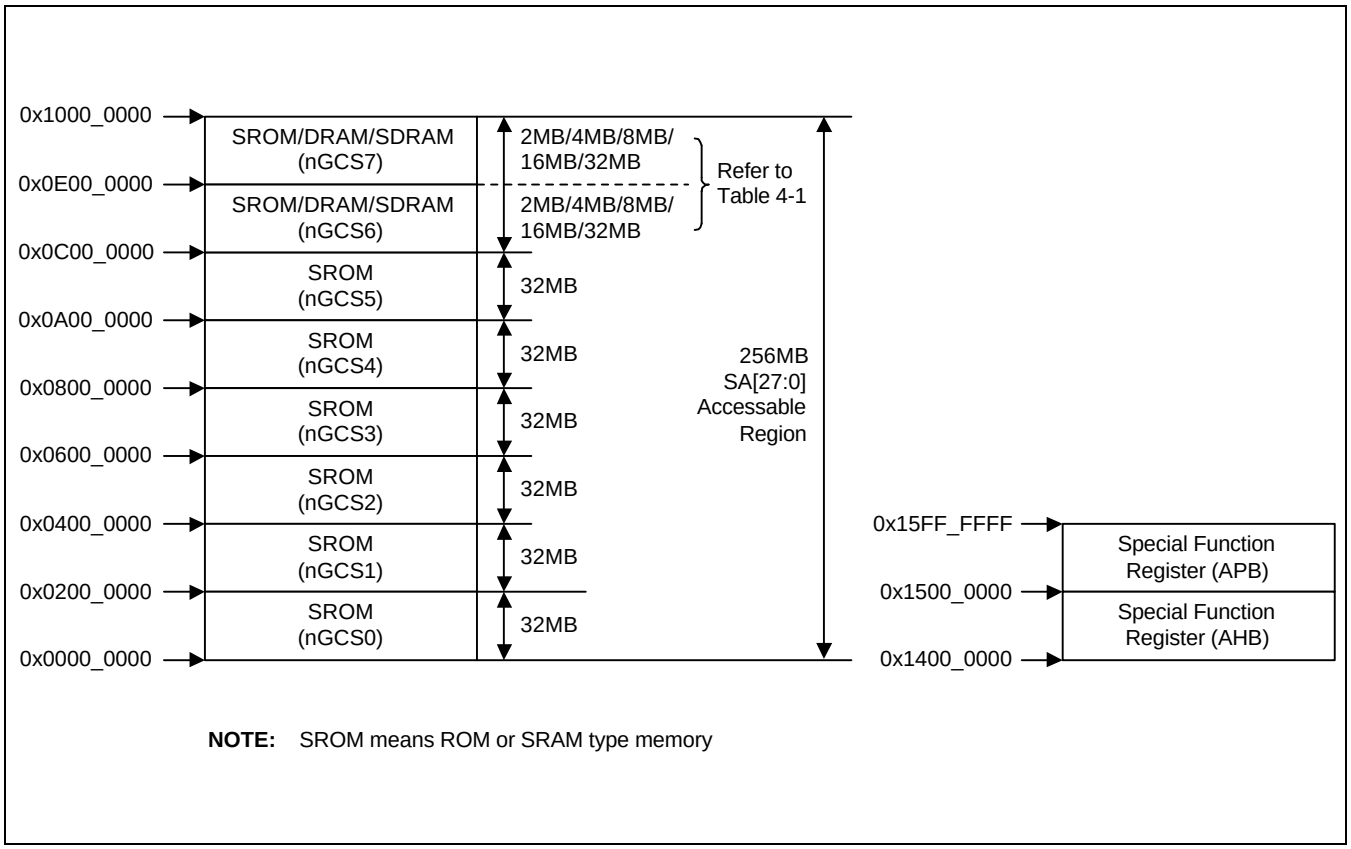


Figure 5-1. S3C2400X Memory Map after Reset

Table 5-1. Bank 6/7 Address

Address	2MB	4MB	8MB	16MB	32MB
Bank 6					
Start address	0xc00_0000	0xc00_0000	0xc00_0000	0xc00_0000	0xc00_0000
End address	0xc1f_ffff	0xc3f_ffff	0xc7f_ffff	0xcff_ffff	0xdff_ffff
Bank 7					
Start address	0xc20_0000	0xc40_0000	0xc80_0000	0xd00_0000	0xe00_0000
End address	0xc3f_ffff	0xc7f_ffff	0xcff_ffff	0xdff_ffff	0xfff_ffff

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

FUNCTION DESCRIPTION

BANK0 BUS WIDTH

The data bus width of BANK0 (nGCS0) should be configured as one of 8-bit,16-bit and 32-bit. Because the BANK0 is the booting ROM bank(map to 0x0000_0000), the bus width of BANK0 should be determined before the first ROM access, which will be determined by the logic level of OM[1:0] at Reset.

OM1 (Operating Mode 1)	OM0 (Operating Mode 0)	Booting ROM Data width
0	0	8-bit
0	1	16-bit
1	0	32-bit
1	1	Test Mode

Programming Memory Controller

All thirteen memory control registers have to be written using the STMIA instruction as shown in the following example;

```
                ;Set memory control registers
                ldr     r0,=SMRDATA
                ldr     r1,=BWSCON              ;   BWSCON Address
                add             r2, r0, #52     ;   End address of SMRDATA
0
                ldr     r3, [r0], #4
                str     r3, [r1], #4
                cmp      r2, r0
                bne     %B0
SMRDATA     DATA
                DCD             0x22221210      ;   BWSCON
                DCD             0x00000600      ;   GCS0
                DCD             0x00000700      ;   GCS1
                DCD             0x00000700      ;   GCS2
                DCD             0x00000700      ;   GCS3
                DCD             0x00000700      ;   GCS4
                DCD             0x00000700      ;   GCS5
                DCD             0x0001002a      ;   GCS6, EDO DRAM(Trcd=3, Tcas=2, Tcp=1, CAN=10bit)
                DCD             0x0001002a      ;   GCS7, EDO DRAM
                DCD             0x00960000 + 953    ;   Refresh(REFEN=1, TREFMD=0, Trp=3, Trc=5, Tchr=3)
                DCD             0x0             ;   Bank Size, 32MB/32MB
                DCD             0x20            ;   MRSR 6(CL=2)
                DCD             0x20            ;   MRSR 7(CL=2)
```

MEMORY (SROM/DRAM/SDRAM) ADDRESS PIN CONNECTIONS

Memory Address Pin	S3C2400X Address @ 8-bit Data Bus	S3C2400X Address @ 16-bit Data Bus	S3C2400X Address @ 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
2MByte	× 8	16Mbit	(1M × 8 × 2B) × 1	A20
	× 16		(512K × 16 × 2B) × 1	
4MB	× 8	16Mb	(2M × 4 × 2B) × 2	A21
	× 16		(1M × 8 × 2B) × 2	
	× 32		(512K × 16 × 2B) × 2	
8MB	× 16	16Mb	(2M × 4 × 2B) × 4	A22
	× 32		(1M × 8 × 2B) × 4	
	× 8	64Mb	(4M × 8 × 2B) × 1	
	× 8		(2M × 8 × 4B) × 1	A[22:21]
	× 16		(2M × 16 × 2B) × 1	A22
	× 16		(1M × 16 × 4B) × 1	A[22:21]
	× 32		(512K × 32 × 4B) × 1	
16MB	× 32	16Mb	(2M × 4 × 2B) × 8	A23
	× 8	64Mb	(8M × 4 × 2B) × 2	
	× 8		(4M × 4 × 4B) × 2	A[23:22]
	× 16		(4M × 8 × 2B) × 2	A23
	× 16		(2M × 8 × 4B) × 2	A[23:22]
	× 32		(2M × 16 × 2B) × 2	A23
	× 32		(1M × 16 × 4B) × 2	A[23:22]
	× 8	128Mb	(4M × 8 × 4B) × 1	
	× 16		(2M × 16 × 4B) × 1	
32MB	× 16	64Mb	(8M × 4 x 2B) × 4	A24
	× 16		(4M × 4 x 4B) × 4	A[24:23]
	× 32		(4M × 8 × 2B) × 4	A24
	× 32		(2M × 8 × 4B) × 4	A[24:23]
	× 16	128Mb	(4M × 8 × 4B) × 2	
	× 32		(2M × 16 × 4B) × 2	
	× 8	256Mb	(8M × 8 × 4B) × 1	
	× 16		(4M × 16 × 4B) × 1	

nWAIT PIN OPERATION

If the WAIT corresponding 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 that nWAIT is high. nWE signal have same relation with nOE as 5-14..

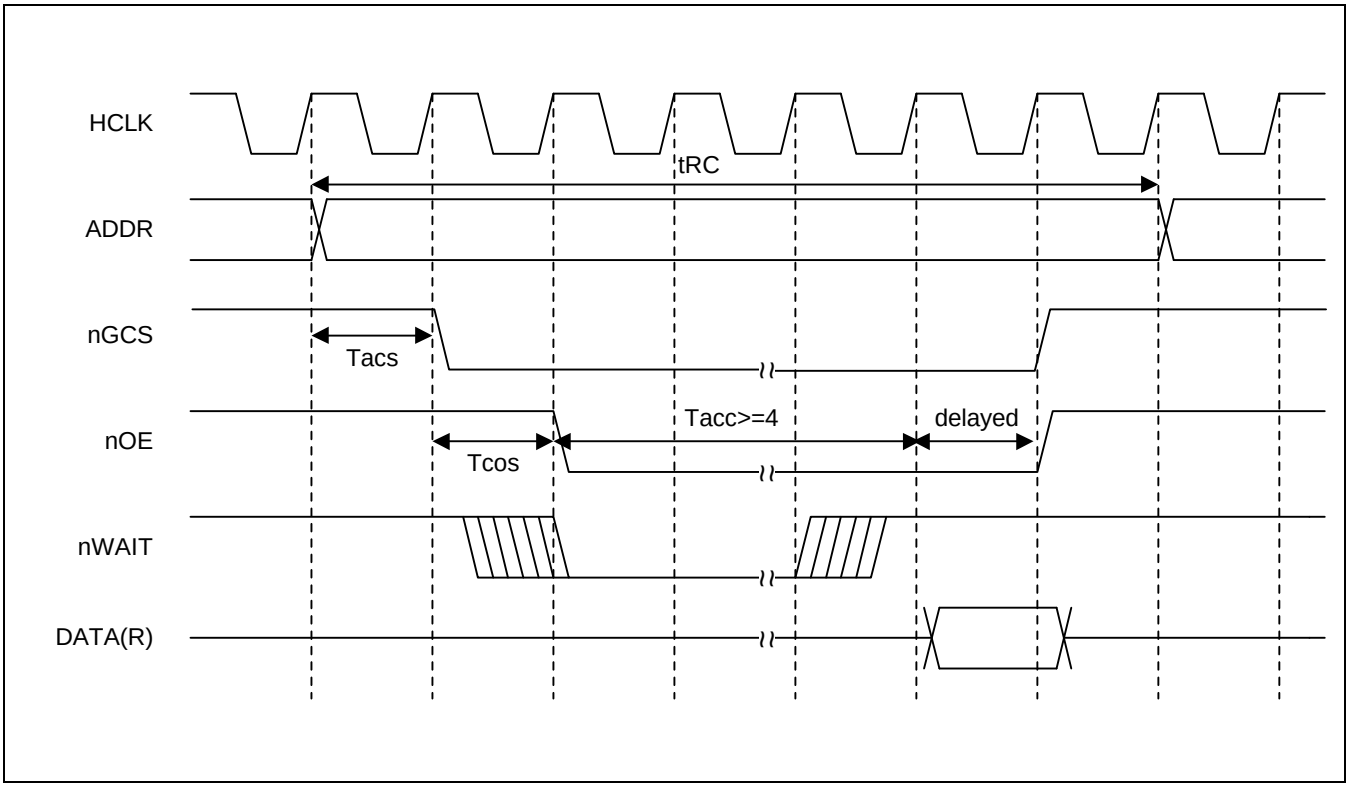


Figure 5-2. S3C2400X External nWAIT Timing Diagram (Tacc≥4)

nXBREQ/NXBACK PIN OPERATION

If nXBREQ is asserted, S3C2400X 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 be de-asserted.

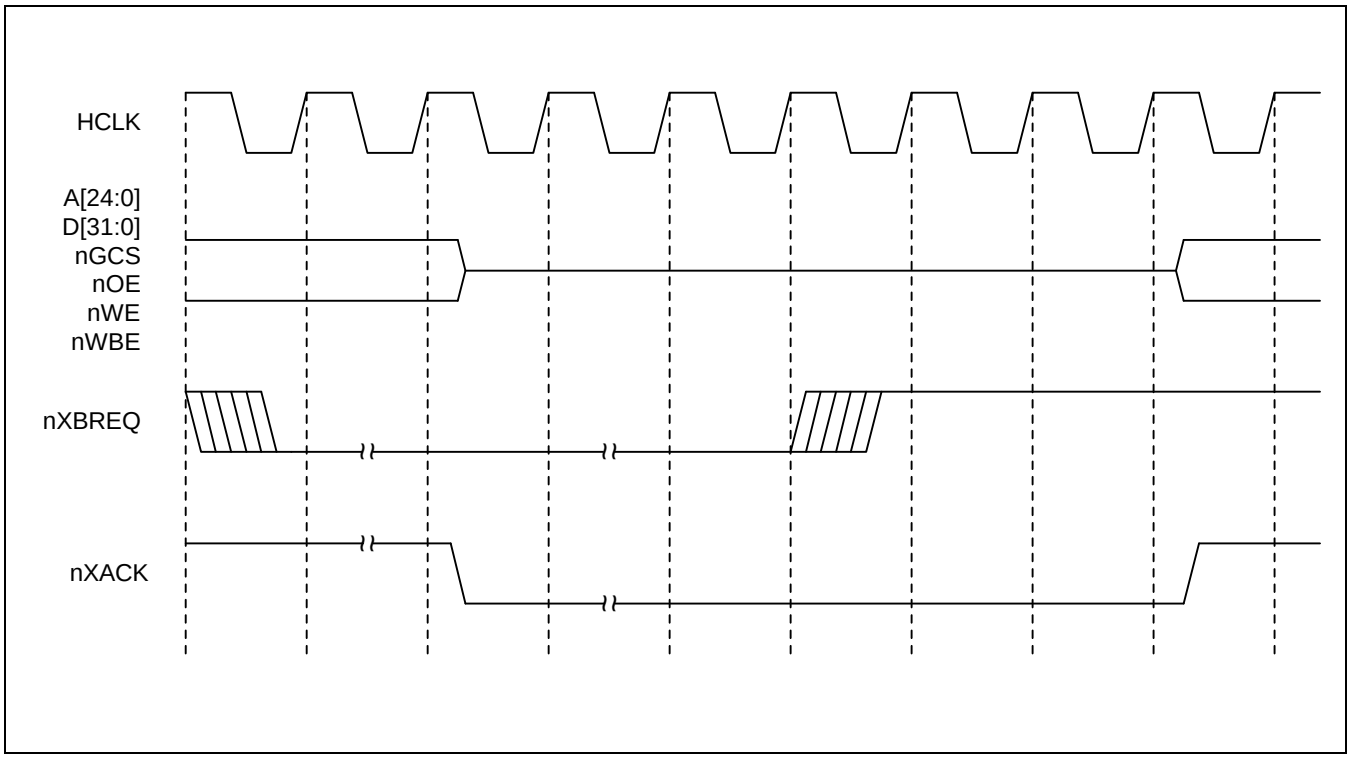


Figure 5-3. S3C2400X nXBREQ/nXBACK Timing Diagram

ROM Memory Interface Example

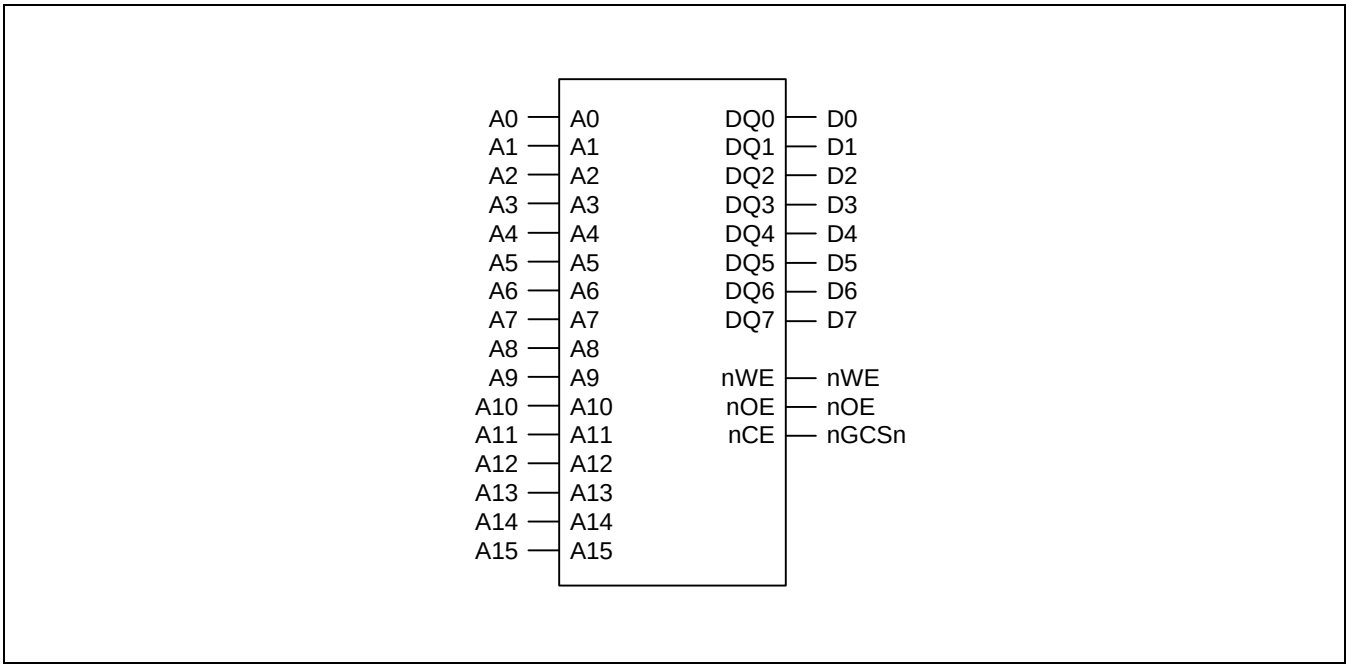


Figure 5-4. Memory Interface with 8bit ROM

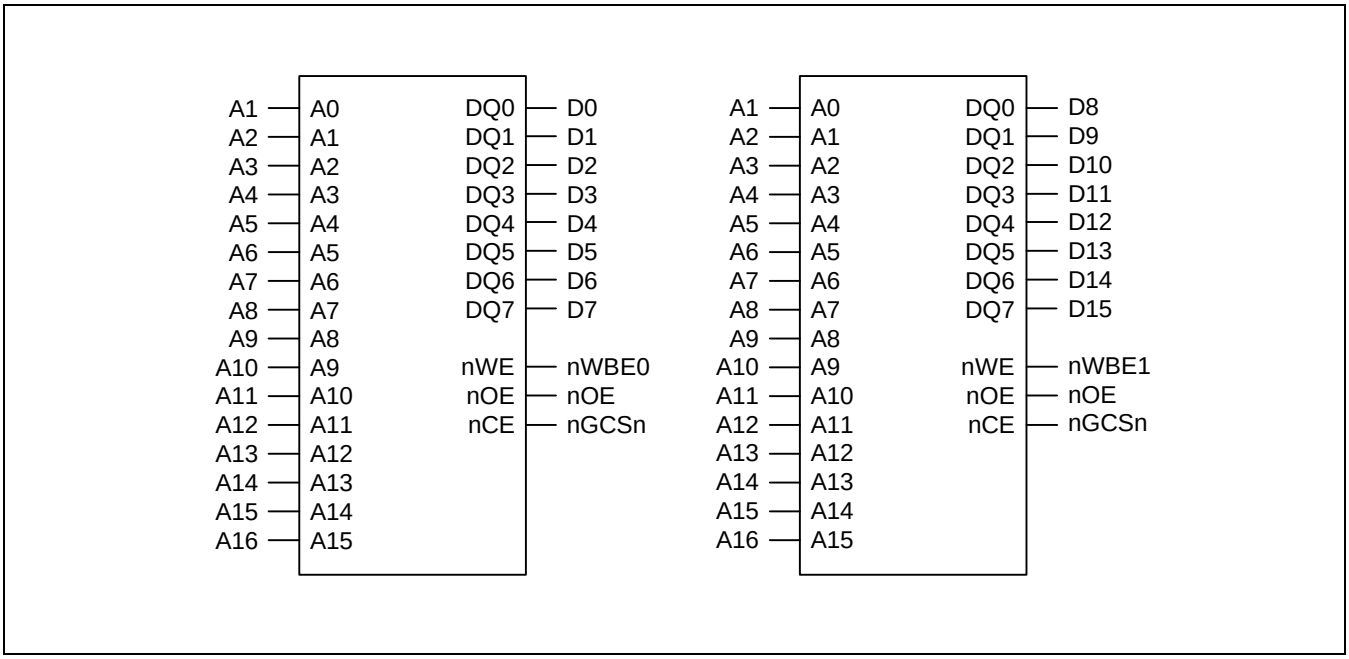


Figure 5-5. Memory Interface with 8bit ROM x 2

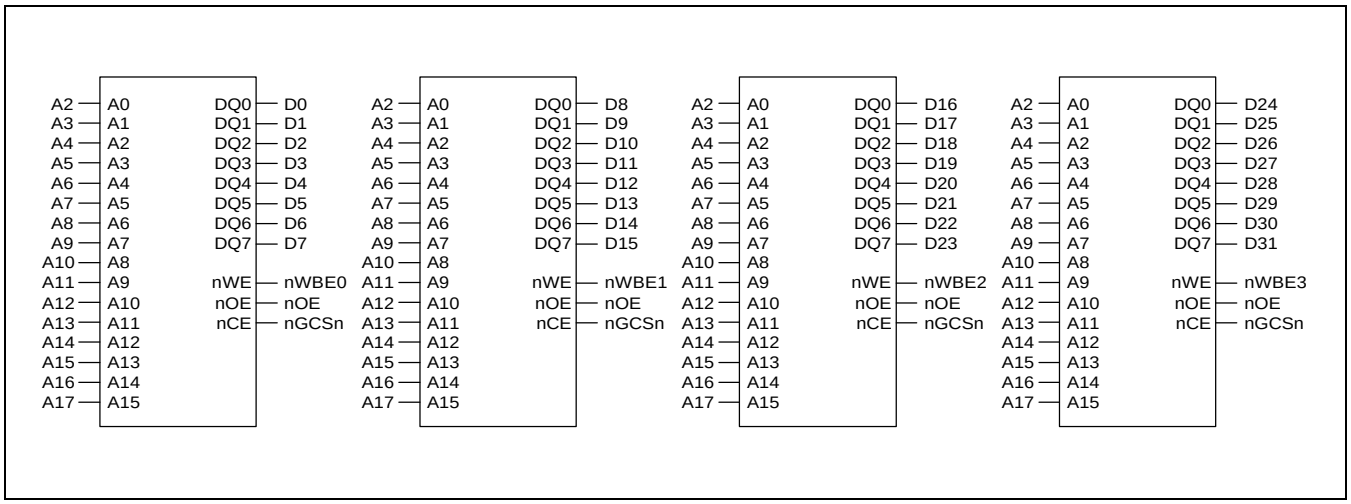


Figure 5-6. Memory Interface with 8bit ROM x 4

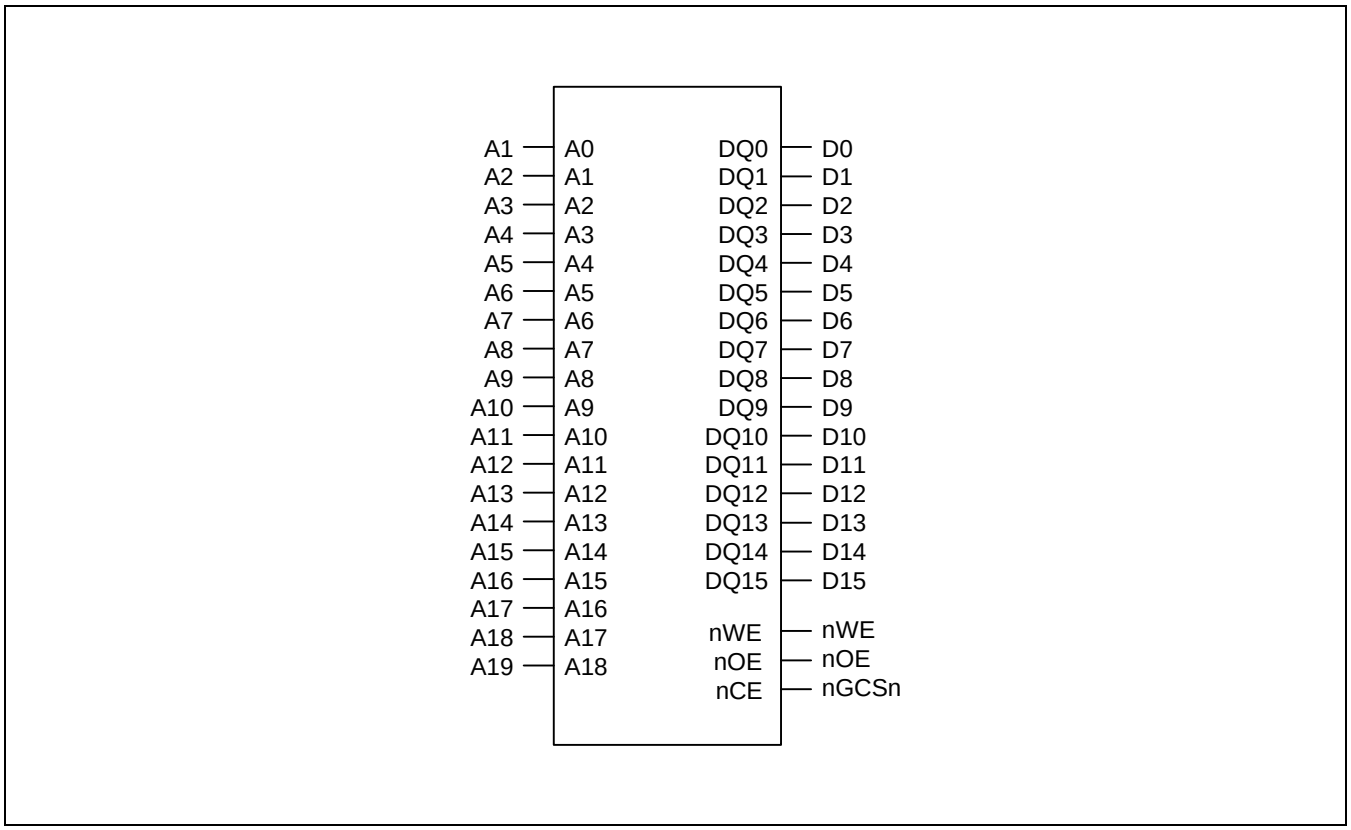


Figure 5-7. Memory Interface with 16bit ROM

SRAM Memory Interface Example

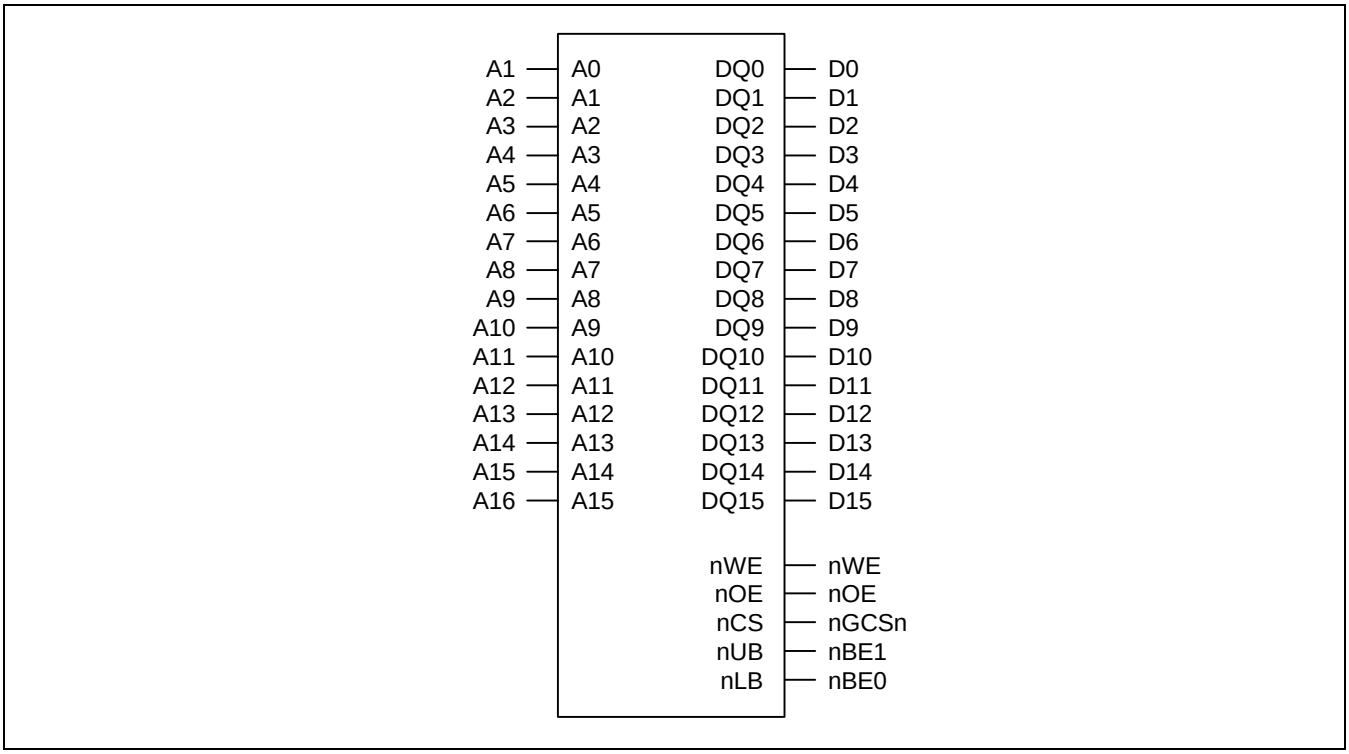


Figure 5-8. Memory Interface with 16bit SRAM

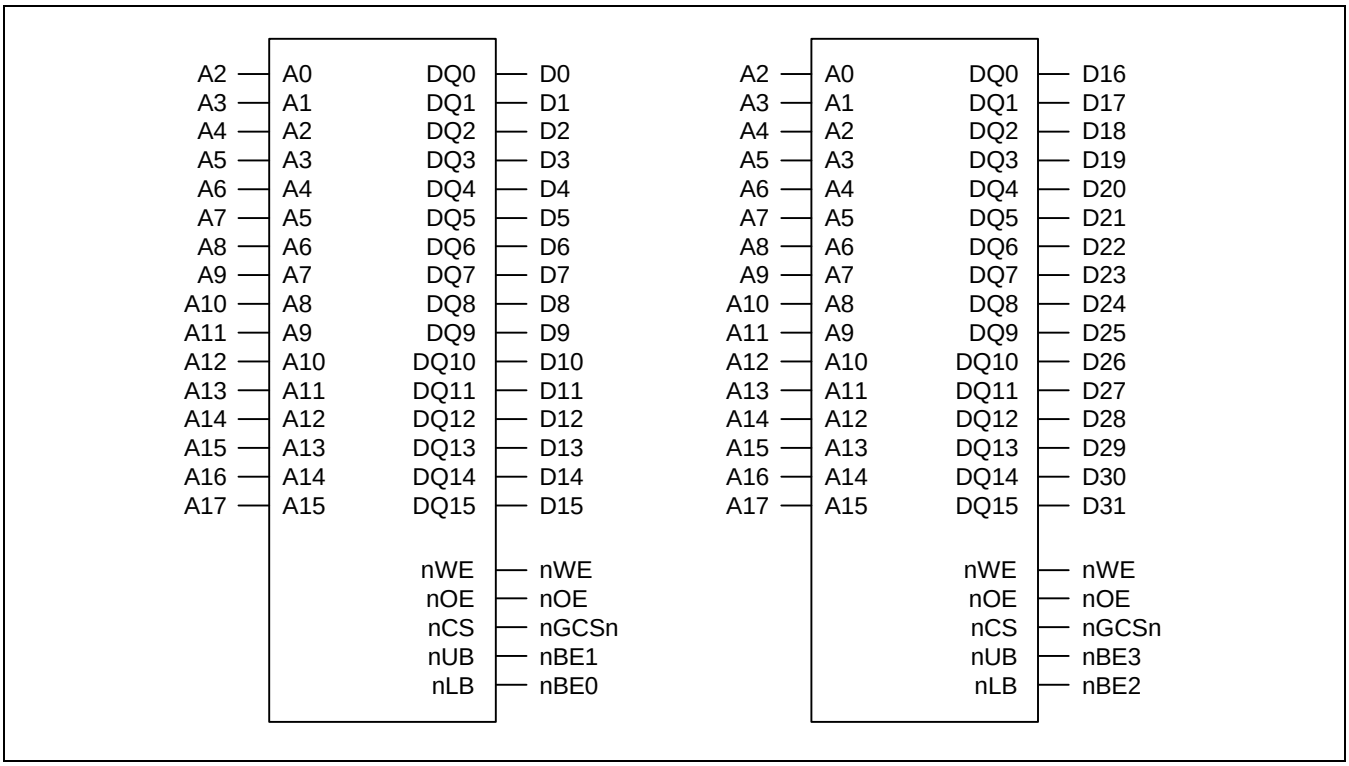


Figure 5-9. Memory Interface with 16bit SRAM x 2

DRAM Memory Interface Example

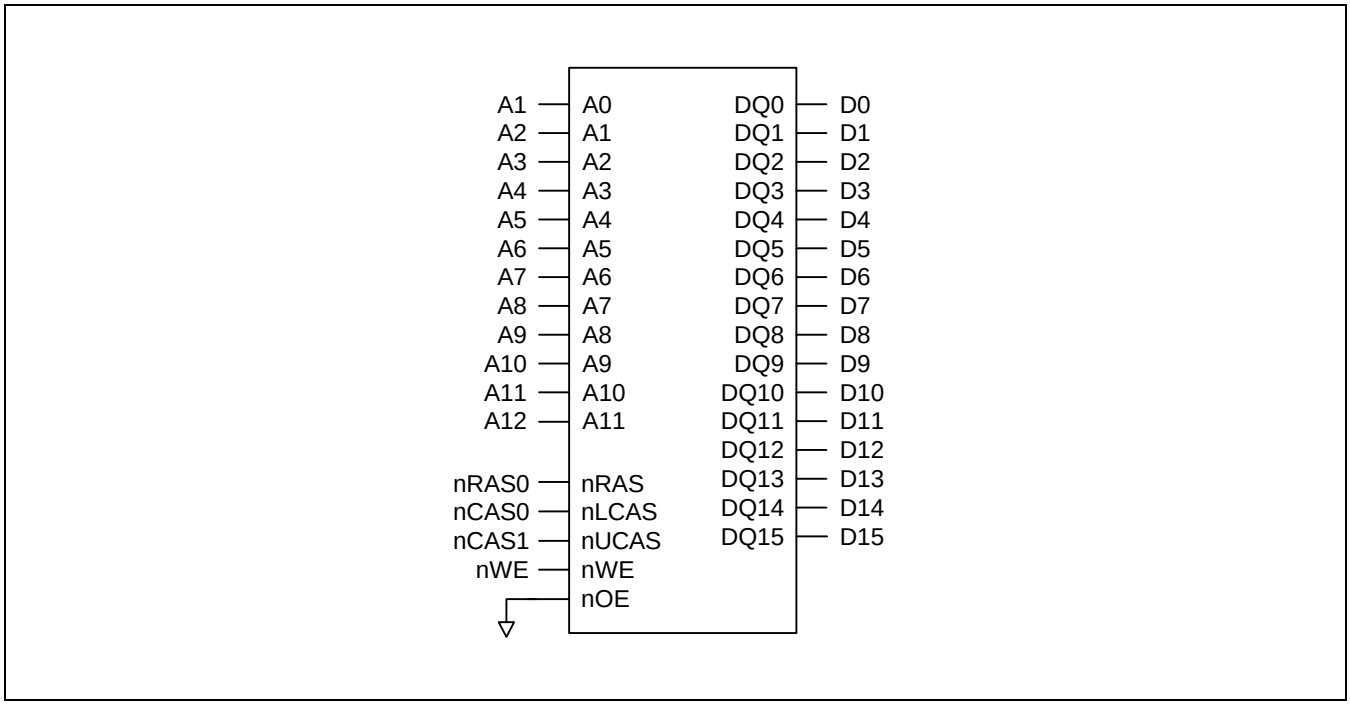


Figure 5-10. Memory Interface with 16bit DRAM

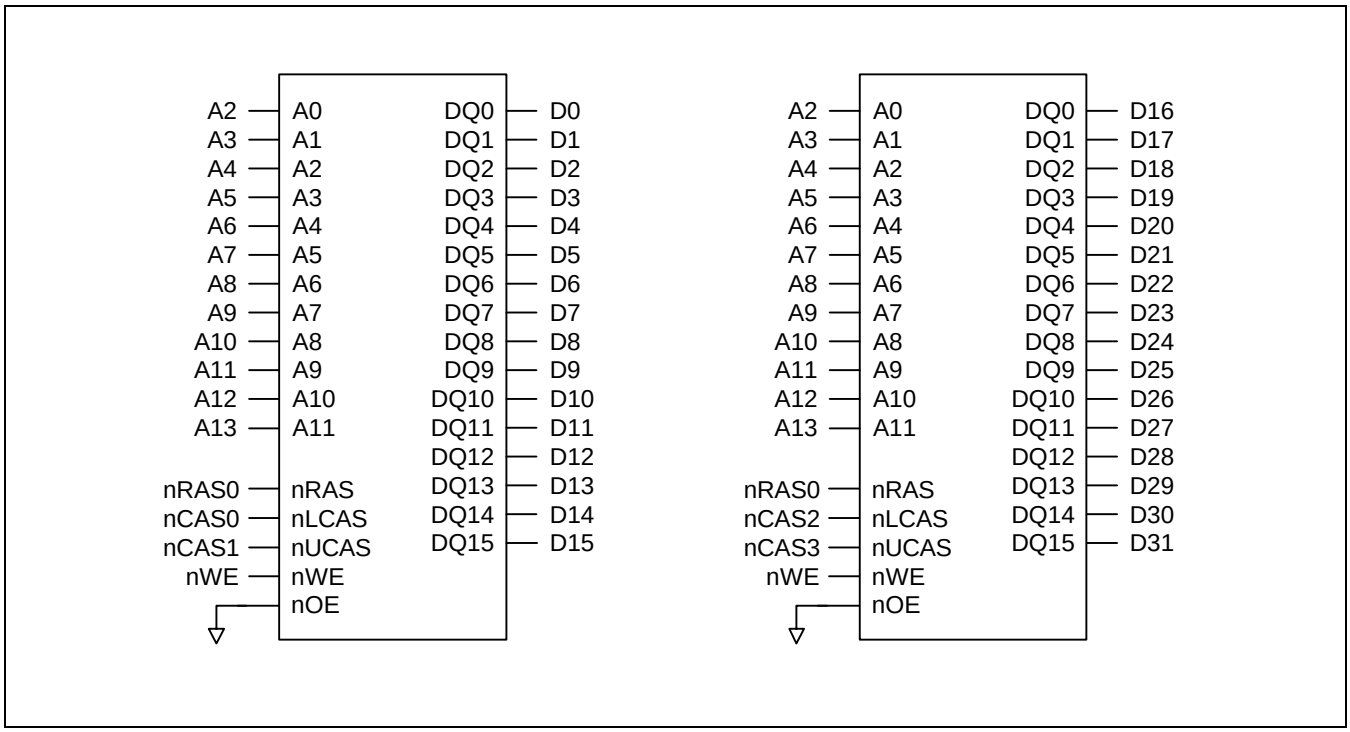


Figure 5-11. Memory Interface with 16bit DRAM x 2

SAMSUNG ELECTRONICS

SDRAM Memory Interface Example

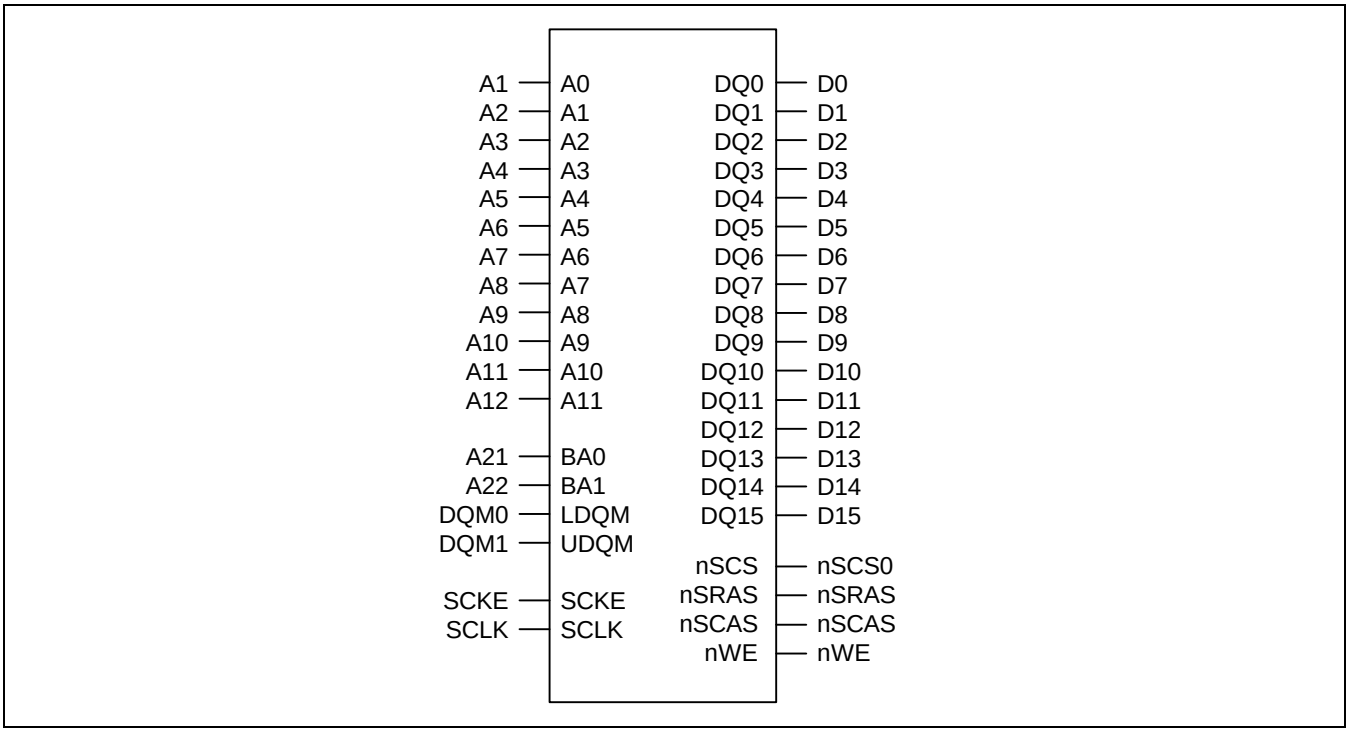


Figure 5-12. Memory Interface with 16bit SDRAM (4Mx16, 4bank)

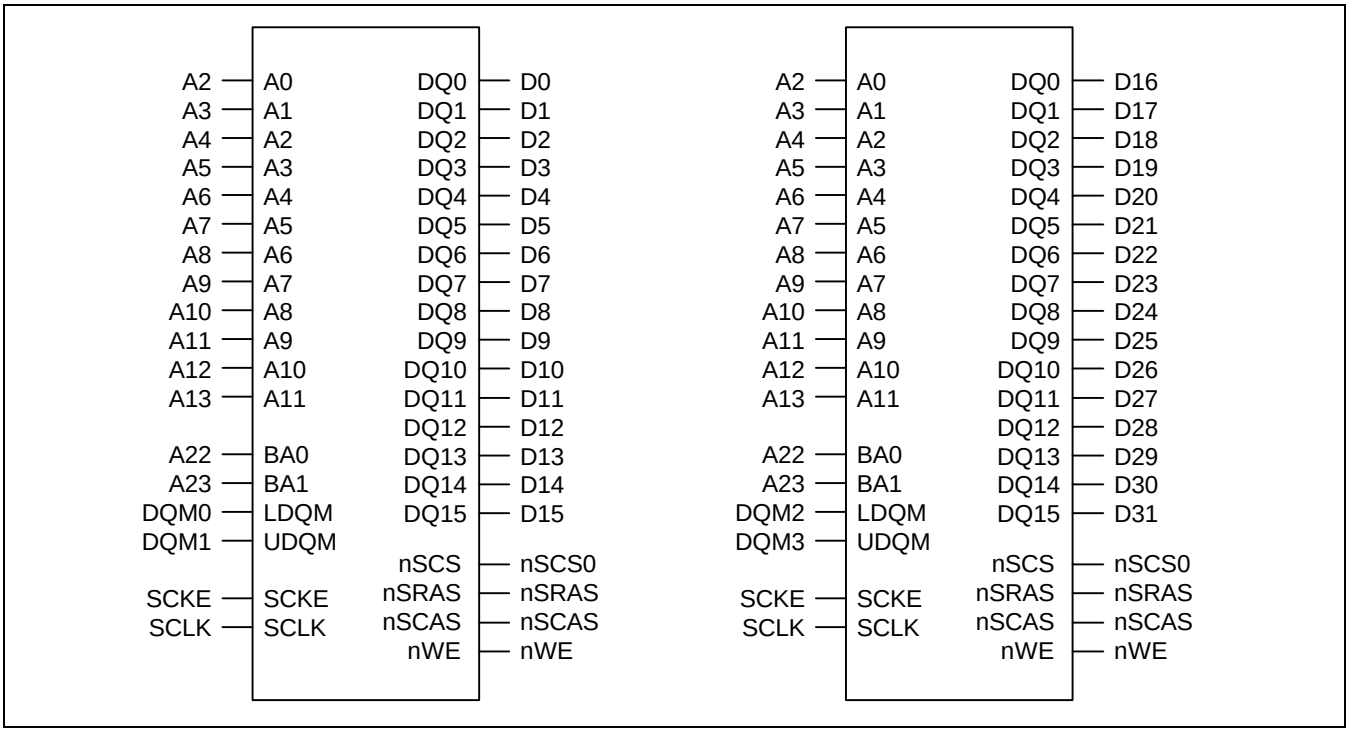


Figure 5-13. Memory Interface with 16bit SDRAM (4Mx16 * 2ea, 4bank)

NOTE: Please refer to Table 5-2 the Bank Address configurations of SDRAM.

PROGRAMMABLE ACCESS CYCLE

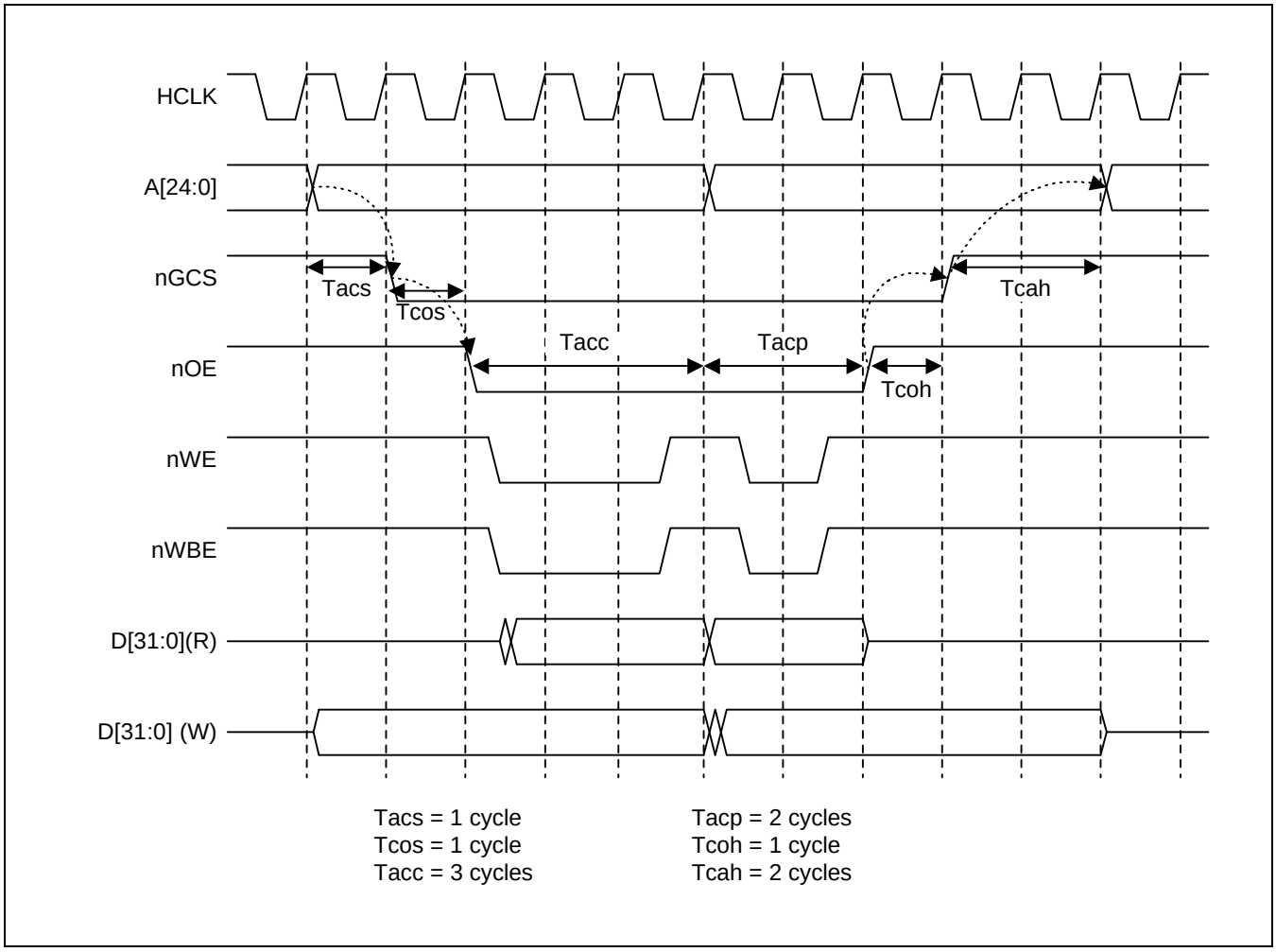


Figure 5-14. S3C2400X nGCS Timing Diagram

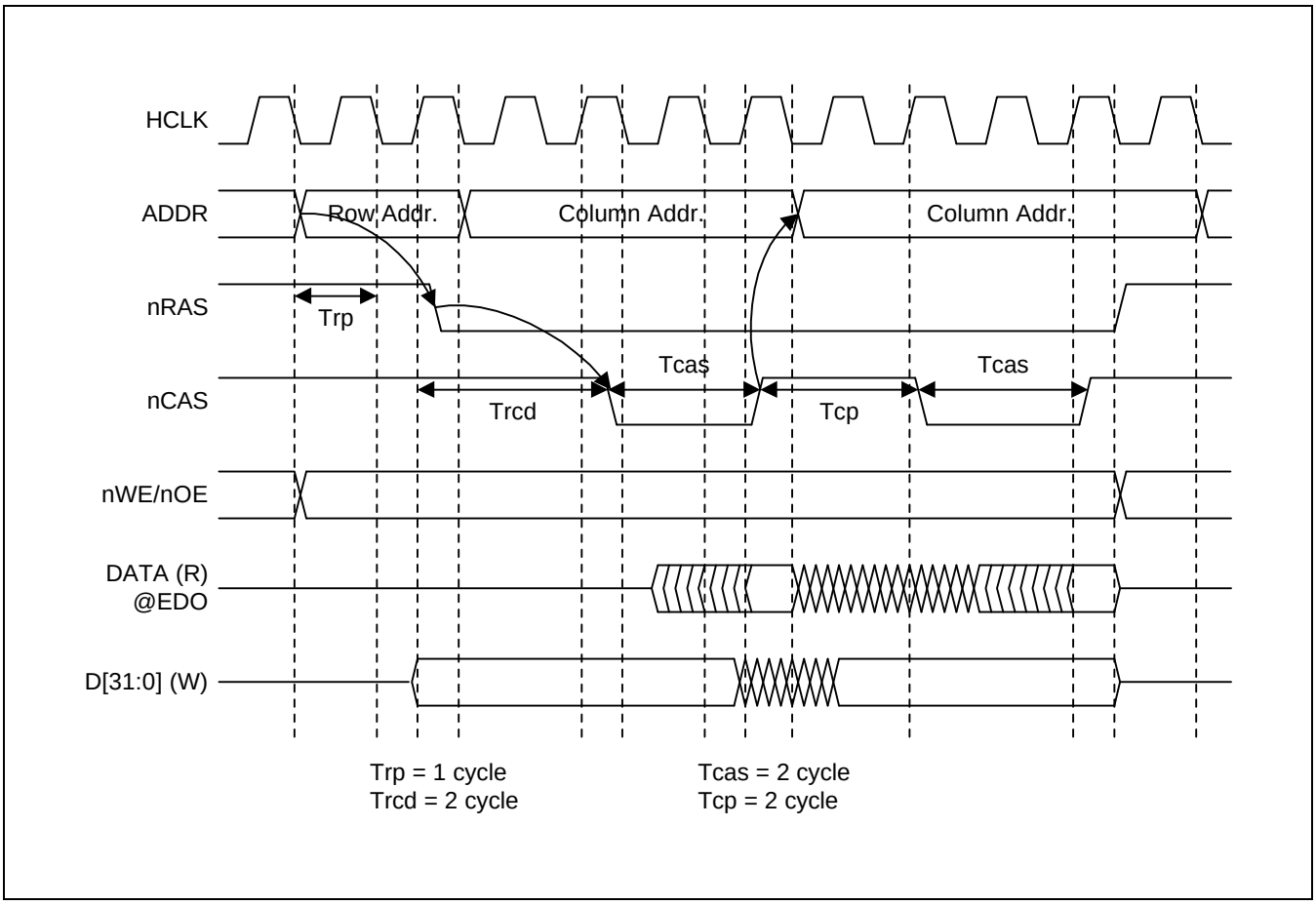


Figure 5-15. S3C2400X DRAM Timing Diagram

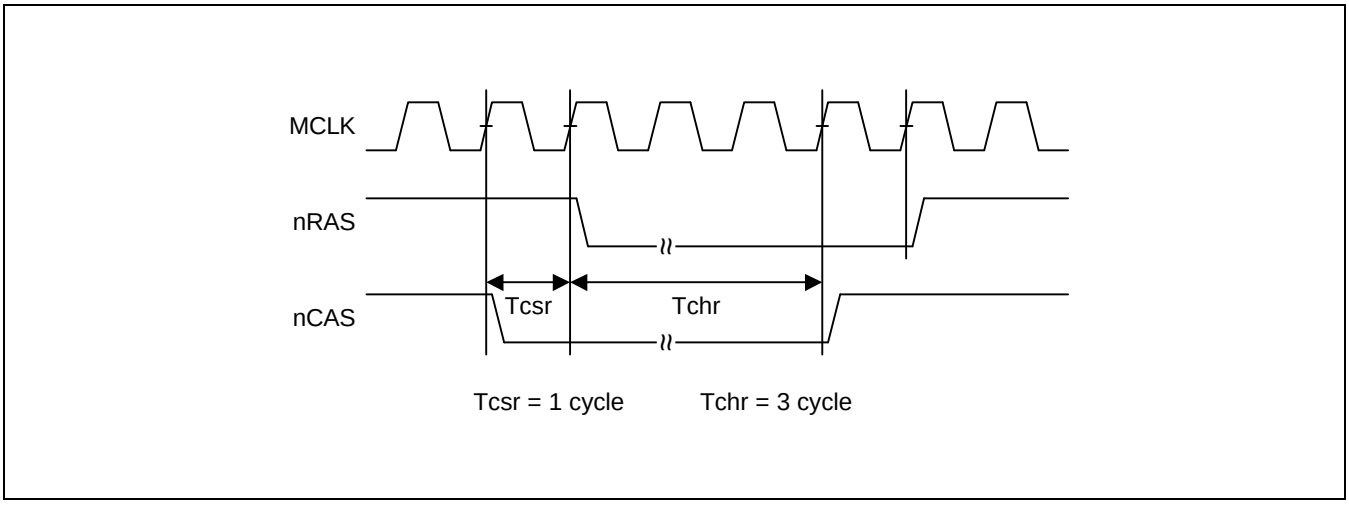


Figure 5-16. S3C2400X DRAM Refresh Timing Diagram

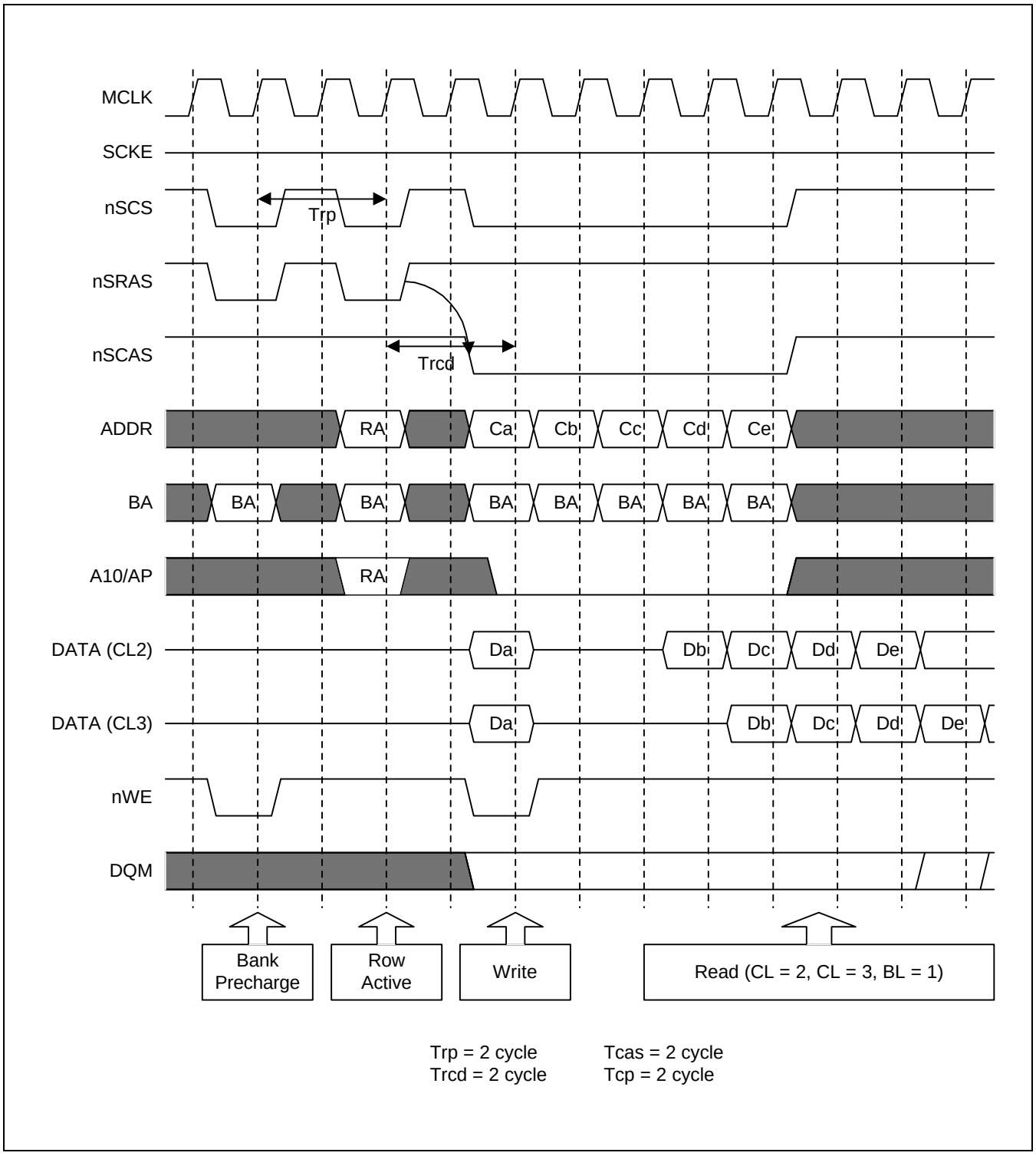


Figure 5-17. S3C2400X SDRAM Timing Diagram

BUS WIDTH & WAIT CONTROL REGISTER (BWSCON)

Register	Address	R/W	Description	Reset Value
BWSCON	0x14000000	R/W	Bus Width & Wait Status Control Register	0x000000

BWSCON	Bit	Description	Initial state
ST7	[31]	This bit determines SRAM for using UB/LB for bank 7 0 = Not using UB/LB (Pin[154:151] is dedicated nWBE[3:0]) 1 = Using UB/LB (Pin[154:151] is dedicated nBE[3:0])	0
WS7	[30]	This bit determines WAIT status for bank 7 0 = WAIT disable 1 = WAIT enable	0
DW7	[29:28]	These two bits determine data bus width for bank 7 00 = 8-bit 01 = 16-bit, 10 = 32-bit	0
ST6	[27]	This bit determines SRAM for using UB/LB for bank 6 0 = Not using UB/LB (Pin[154:151] is dedicated nWBE[3:0]) 1 = Using UB/LB (Pin[154:151] is dedicated nBE[3:0])	0
WS6	[26]	This bit determines WAIT status for bank 6 0 = WAIT disable, 1 = WAIT enable	0
DW6	[25:24]	These two bits determine data bus width for bank 6 00 = 8-bit 01 = 16-bit, 10 = 32-bit	0
ST5	[23]	This bit determines SRAM for using UB/LB for bank 5 0 = Not using UB/LB (Pin[154:151] is dedicated nWBE[3:0]) 1 = Using UB/LB (Pin[154:151] is dedicated nBE[3:0])	0
WS5	[22]	This bit determines WAIT status for bank 5 0 = WAIT disable, 1 = WAIT enable	0
DW5	[21:20]	These two bits determine data bus width for bank 5 00 = 8-bit 01 = 16-bit, 10 = 32-bit	0
ST4	[19]	This bit determines SRAM for using UB/LB for bank 4 0 = Not using UB/LB (Pin[154:151] is dedicated nWBE[3:0]) 1 = Using UB/LB (Pin[154:151] is dedicated nBE[3:0])	0
WS4	[18]	This bit determines WAIT status for bank 4 0 = WAIT disable 1 = WAIT enable	0
DW4	[17:16]	These two bits determine data bus width for bank 4 00 = 8-bit 01 = 16-bit, 10 = 32-bit	0
ST3	[15]	This bit determines SRAM for using UB/LB for bank 3 0 = Not using UB/LB (Pin[154:151] is dedicated nWBE[3:0]) 1 = Using UB/LB (Pin[154:151] is dedicated nBE[3:0])	0
WS3	[14]	This bit determines WAIT status for bank 3 0 = WAIT disable 1 = WAIT enable	0
DW3	[13:12]	These two bits determine data bus width for bank 3 00 = 8-bit 01 = 16-bit, 10 = 32-bit	0
ST2	[11]	This bit determines SRAM for using UB/LB for bank 2 0 = Not using UB/LB (Pin[154:151] is dedicated nWBE[3:0]) 1 = Using UB/LB (Pin[154:151] is dedicated nBE[3:0])	0

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

BWSCON	Bit	Description	Initial state
WS2	[10]	This bit determines WAIT status for bank 2 0 = WAIT disable 1 = WAIT enable	0
DW2	[9:8]	These two bits determine data bus width for bank 2 00 = 8-bit 01 = 16-bit, 10 = 32-bit	0
ST1	[7]	This bit determines SRAM for using UB/LB for bank 1 0 = Not using UB/LB (Pin[154:151] is dedicated nWBE[3:0]) 1 = Using UB/LB (Pin[154:151] is dedicated nBE[3:0])	0
WS1	[6]	This bit determines WAIT status for bank 1 0 = WAIT disable, 1 = WAIT enable	0
DW1	[5:4]	These two bits determine data bus width for bank 1 00 = 8-bit 01 = 16-bit, 10 = 32-bit	0
DW0	[2:1]	Indicates data bus width for bank 0 (read only) 00 = 8-bit 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 DRAM and SRAM is 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	0x14000004	R/W	Bank 0 control register	0x0700
BANKCON1	0x14000008	R/W	Bank 1 control register	0x0700
BANKCON2	0x1400000C	R/W	Bank 2 control register	0x0700
BANKCON3	0x14000010	R/W	Bank 3 control register	0x0700
BANKCON4	0x14000014	R/W	Bank 4 control register	0x0700
BANKCON5	0x14000018	R/W	Bank 5 control register	0x0700

BANKCONn	Bit	Description	Initial State
Tacs	[14:13]	Address set-up before nGCSn 00 = 0 clock 01 = 1 clock 10 = 2 clocks 11 = 4 clocks	00
Tcos	[12:11]	Chip selection set-up 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
Toch	[7:6]	Chip selection hold on nOE 00 = 0 clock 01 = 1 clock 10 = 2 clocks 11 = 4 clocks	000
Tcah	[5:4]	Address holding 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	0x1400001C	R/W	Bank 6 control register	0x18008
BANKCON7	0x14000020	R/W	Bank 7 control register	0x18008

BANKCONn	Bit	Description	Initial State
MT	[16:15]	These two bits determine the memory type for bank6 and bank7 00 = ROM or SRAM 01 = Reserved(Don't use) 10 = EDO DRAM 11 = Sync. DRAM	11
Memory Type = ROM or SRAM [MT=00] (15-bit)			
Tacs	[14:13]	Address set-up before nGCS 00 = 0 clock 01 = 1 clock 10 = 2 clocks 11 = 4 clocks	00
Tcos	[12:11]	Chip selection set-up 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
Toch	[7:6]	Chip selection hold on nOE 00 = 0 clock 01 = 1 clock 10 = 2 clocks 11 = 4 clocks	00
Tcah	[5:4]	Address hold time on 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 = EDO DRAM [MT=10] (6-bit)			
Trcd	[5:4]	RAS to CAS delay 00 = 1 clock 01 = 2 clocks 10 = 3 clocks 11 = 4 clocks	00
Tcas	[3]	CAS pulse width 0 = 1 clock 1 = 2 clocks	0
Tcp	[2]	CAS pre-charge 0 = 1 clock 1 = 2 clocks	0
CAN	[1:0]	Column address number 00 = 8-bit 01 = 9-bit 10 = 10-bit 11 = 11-bit	00

BANK CONTROL REGISTER (BANKCONn: nGCS6-nGCS7) (Continued)

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

SUPPORTED BANK 6/7 MEMORY CONFIGURATION

Bank	Support						Not support	
Bank7	SROM	SROM	DRAM	DRAM	SDRAM	SDRAM	SDRAM	DRAM
Bank6	DRAM	SDRAM	SROM	DRAM	SROM	SDRAM	DRAM	SDRAM

NOTE: SROM means ROM or SRAM type memory

REFRESH CONTROL REGISTER

Register	Address	R/W	Description	Reset Value
REFRESH	0x14000024	R/W	DRAM/SDRAM refresh control register	0xac0000

REFRESH	Bit	Description	Initial State
REFEN	[23]	DRAM/SDRAM Refresh Enable 0 = Disable 1 = Enable(self or CBR/auto refresh)	1
TREFMD	[22]	DRAM/SDRAM Refresh Mode 0 = CBR/Auto Refresh 1 = Self Refresh In self-refresh time, the DRAM/SDRAM control signals are driven to the appropriate level.	0
Trp	[21:20]	DRAM/SDRAM RAS pre-charge Time DRAM: 00 = 1.5 clocks 01 = 2.5 clocks 10 = 3.5 clocks 11 = 4.5 clocks SDRAM: 00 = 2 clocks 01 = 3 clocks 10 = 4 clocks 11 = Not support	10
Trc	[19:18]	SDRAM RC minimum Time 00 = 4 clocks 01 = 5 clocks 10 = 6 clocks 11 = 7 clocks	11
Tchr	[17:16]	CAS Hold Time(DRAM) 00 = 1 clock 01 = 2 clocks 10 = 3 clocks 11 = 4 clocks	00
Reserved	[15:11]	Not use	0000
Refresh Counter	[10:0]	DRAM/SDRAM refresh count value. Please, refer to chap. 6 DRAM refresh controller bus priority section. 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	0x14000028	R/W	Flexible bank size register	0x0

BANKSIZE	Bit	Description	Initial State
SCLKEN	[4]	SCLK is enabled only during SDRAM access cycle for reducing power consumption. When SDRAM isn't be accessed, SCLK is '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 000 = 32M/32M 100 = 2M/2M 101 = 4M/4M 110 = 8M/8M 111 = 16M/16M	000

SDRAM MODE REGISTER SET REGISTER (MRSR)

Register	Address	R/W	Description	Reset Value
MRSRB6	0x1400002C	R/W	Mode register set register bank6	xxx
MRSRB7	0x14000030	R/W	Mode register set register bank7	xxx

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

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

IMPORTANT NOTES

1. All 13 memory control registers have to be written using the STM instruction (not STMIA instruction).
2. In STOP mode/SL_IDLE mode, DRAM/SDRAM has to enter the DRAM/SDRAM self-refresh mode.

NOTES

6 CLOCK & POWER MANAGEMENT

OVERVIEW

The clock & power management unit consists of 3 parts, clock control, USB control, and power control.

The Clock control logic in S3C2400X can generate the required clock signals, FCLK for CPU, HCLK for the AHB bus peripherals, and PCLK for the APB bus peripherals. There are two PLL in S3C2400X. One is for FCLK,HCLK, and PCLK, the other is dedicated for USB block (48MHz). The clock control logic can make slow clock without PLL and connect/disconnect the clock to each peripheral block by S/W, which will reduce the power consumption.

In the power control logic, S3C2400X has various power management schemes to keep optimal power consumption for a given task. The power management in S3C2400X consists of five modes: NORMAL mode, SLOW mode, IDLE mode, STOP mode and SL_IDLE mode for the self-refresh mode LCD.

NORMAL mode is used to supply clocks to CPU as well as all peripherals in S3C2400X. In this case, the power consumption will be maximized when all peripherals are turned on. The user can control the operation of peripherals by S/W. For example, if a timer is not needed, the user can disconnect the clock to the timer to reduce power.

SLOW mode is non-PLL mode. Unlike the Normal mode, the Slow mode uses an external clock (XTIpll or EXTCLK) directly as FCLK in S3C2400X without PLL. In this case, the power consumption depends on the frequency of the external clock only. The power consumption due to PLL itself is excluded.

IDLE mode disconnects the clock (FCLK) only to CPU core while it supplies the clock to all peripherals. By using IDLE mode, power consumption due to CPU core can be reduced. Any interrupt request to CPU can wake-up from Idle mode.

STOP mode freezes all clocks to the CPU as well as peripherals by disabling PLLs. The power consumption is only due to the leakage current in S3C2400X, which is uA unit. The wake-up from STOP mode can be done by activating external interrupt pins or RTC alarm.

SL_IDLE mode causes only the LCD controller to work. In this case, the clock to CPU and all peripherals except LCD controller should be stopped, therefore, the power consumption in the SL_Idle mode is less than that in the Idle mode. But, SL_IDLE mode needs special LCD panel with SED1580D (mono STN LCD, Seiko Epson).

FUNCTION DESCRIPTION

CLOCK ARCHITECTURE

Figure 6-1 shows a block diagram of the clock architecture. The main clock source comes from an external crystal(XTIpll) or external clock(EXTCLK). The clock generator consists of an oscillator block(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 S3C2400X.

CLOCK SOURCE SELECTION

Table 6-1 shows the relationship between the combination of mode control pins (OM3 and OM2) and the selection of source clock for S3C2400X. The OM[3:2] status is latched internally by referring the OM3 and OM2 pins at the rising edge of nRESET.

Table 6-1. Clock source selection at boot-up

Mode OM[3:2]	MPLL state	UPLL state	Main Clock source	USB Clock source
00	On (1)	On (1)	Crystal	Crystal
01	On (1)	On (1)	Crystal	EXTCLK
10	On (1)	On (1)	EXTCLK	Crystal
11	On (1)	On (1)	EXTCLK	EXTCLK

NOTES:

1. Although the M(U)PLL starts just after a reset, the M(U)PLL output(Mpll,Upll) isn't used as the system clock until the S/W writes valid settings to the M(U)PLLCON 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 wants to maintain the default value of MPLLCON register, the user should write the same value into M(U)PLLCON register.
2. OM[3:2] is used to determine test mode when OM[1:0] is 11.

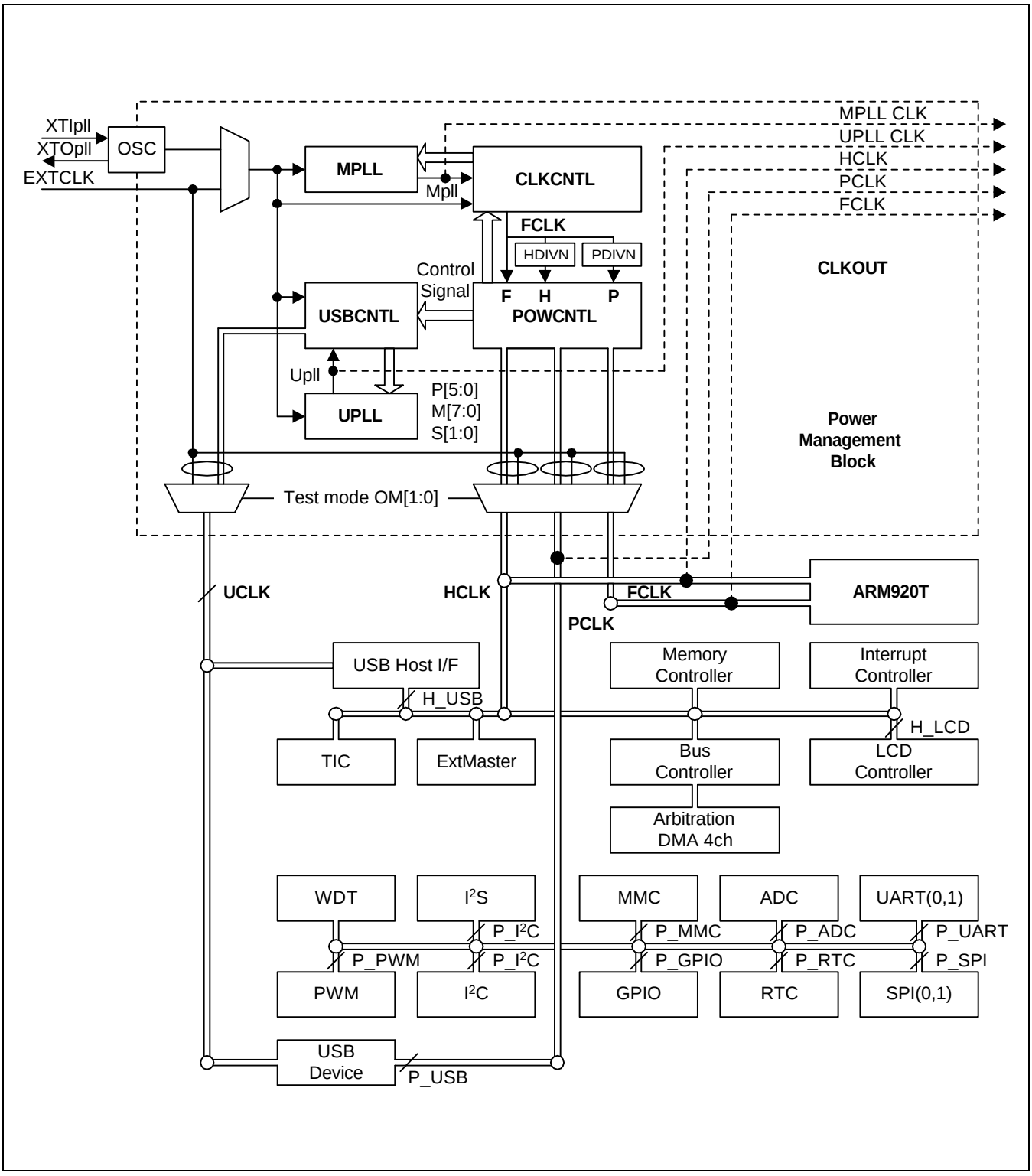


Figure 6-1. Clock Generator Block Diagram

PLL (PHASE-LOCKED-LOOP)

The MPLL within the clock generator is the circuit which synchronizes an output signal with a reference input signal in frequency and phase. In this application, it includes the following basic blocks (Figure 6-2 shows the clock generator block diagram); the VCO(Voltage Controlled Oscillator) 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 PFD(Phase Frequency Detector), the divider S to divide the VCO output frequency by s which is Mpll(the output frequency from MPLL block), the phase difference detector, charge pump, and 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 same as the MPLL in every aspect.

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

Phase Difference Detector(PFD)

The PFD monitors the phase difference between the Fref (the reference frequency as shown in Fig. 6-2) and Fvco, and generates a control signal(tracking signal) when it detects a difference.

Charge Pump(PUMP)

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

Loop Filter

The control signal that 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 consisting of a resistor and capacitor.

A recommended capacitance in the external loop filter(Capacitance as shown in Figure 6-2) is 5 pF or above.

Voltage Controlled Oscillator (VCO)

The output voltage from the loop filter drives the VCO, which causes 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 a control signal to the charge pump, which stabilizes the input voltage to the loop filter in turn. The VCO frequency then remains constant, and the PLL remains locked onto the system clock.

Usual Conditions for PLL & Clock Generator

The following conditions are generally used.

Loop filter capacitance for UPLL/MPLL	5 pF
External X-tal frequency	6 – 20 MHz (note)
External capacitance used for X-tal	15 – 22 pF

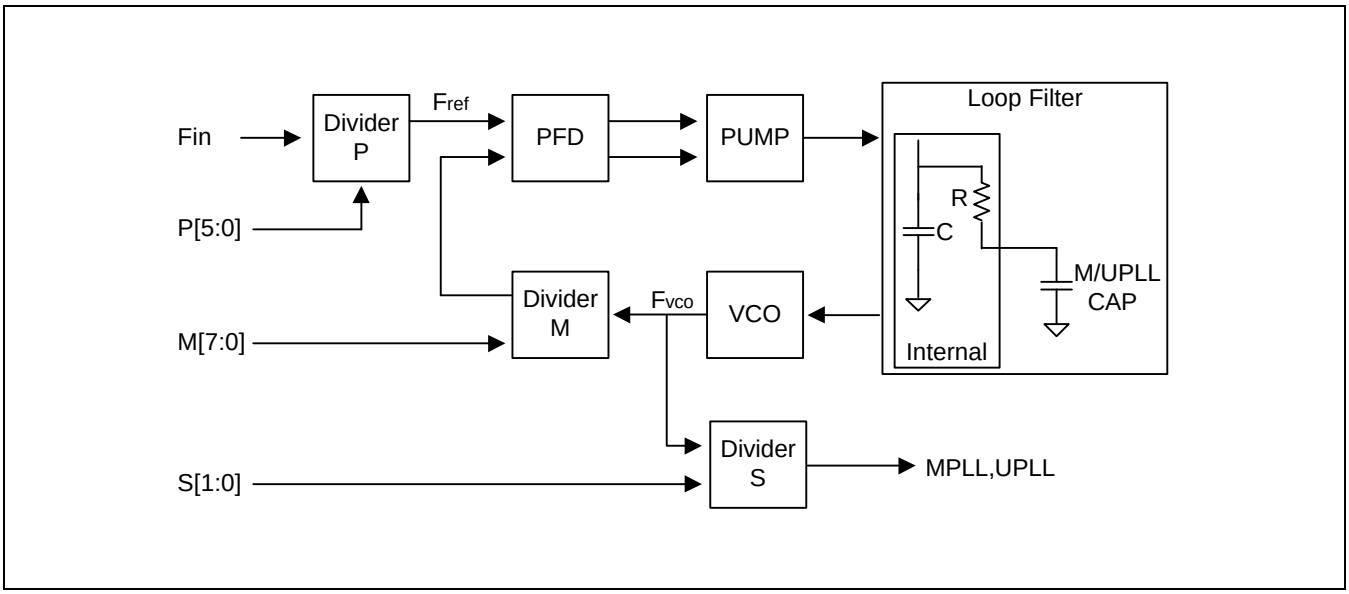
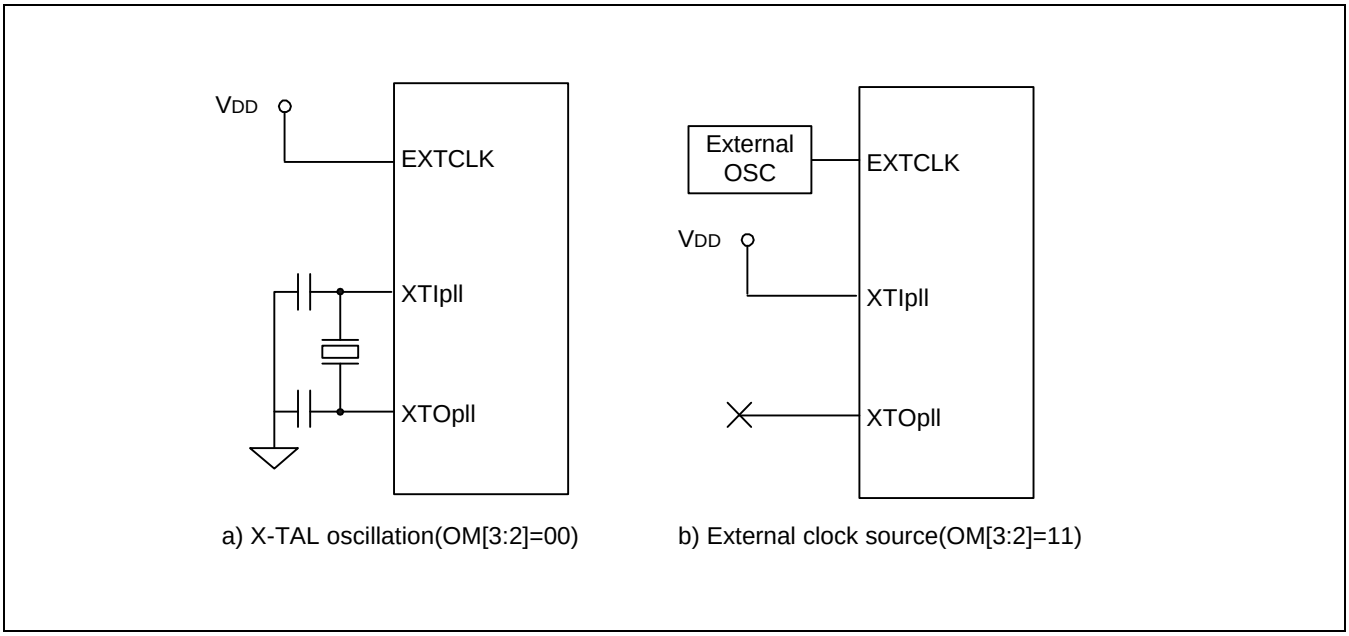


Figure 6-2. PLL (Phase-Locked Loop) Block Diagram



a) X-TAL oscillation(OM[3:2]=00) b) External clock source(OM[3:2]=11)

Figure 6-3. 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 wake-up from power-down mode.

PLL Lock Time

The lock time is the minimum time required for PLL output stabilization. The lock time should be a minimum of 150us. After reset and wake-up from STOP and SL_IDLE mode, respectively, the lock-time is inserted automatically by the internal logic with lock time count register. The automatically inserted lock time is calculated as follows;

t_lock(the PLL lock time by H/W logic) = (1/ Fin) x n, (n = M_LTIME,U_LTIME value)

Power-On Reset(XTIpll)

Figure 6-4 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 fed directly to FCLK instead of the Mpll(PLL output) before the S/W newly configures the PLLCON. Even if the user wants to use the default value of PLLCON register after reset, user should write the same value into PLLCON register by S/W.

The PLL begins the lockup sequence again toward the new frequency only after the S/W configures the PLL with a new frequency. FCLK can be configured to be PLL output (Mpll) immediately after lock time.

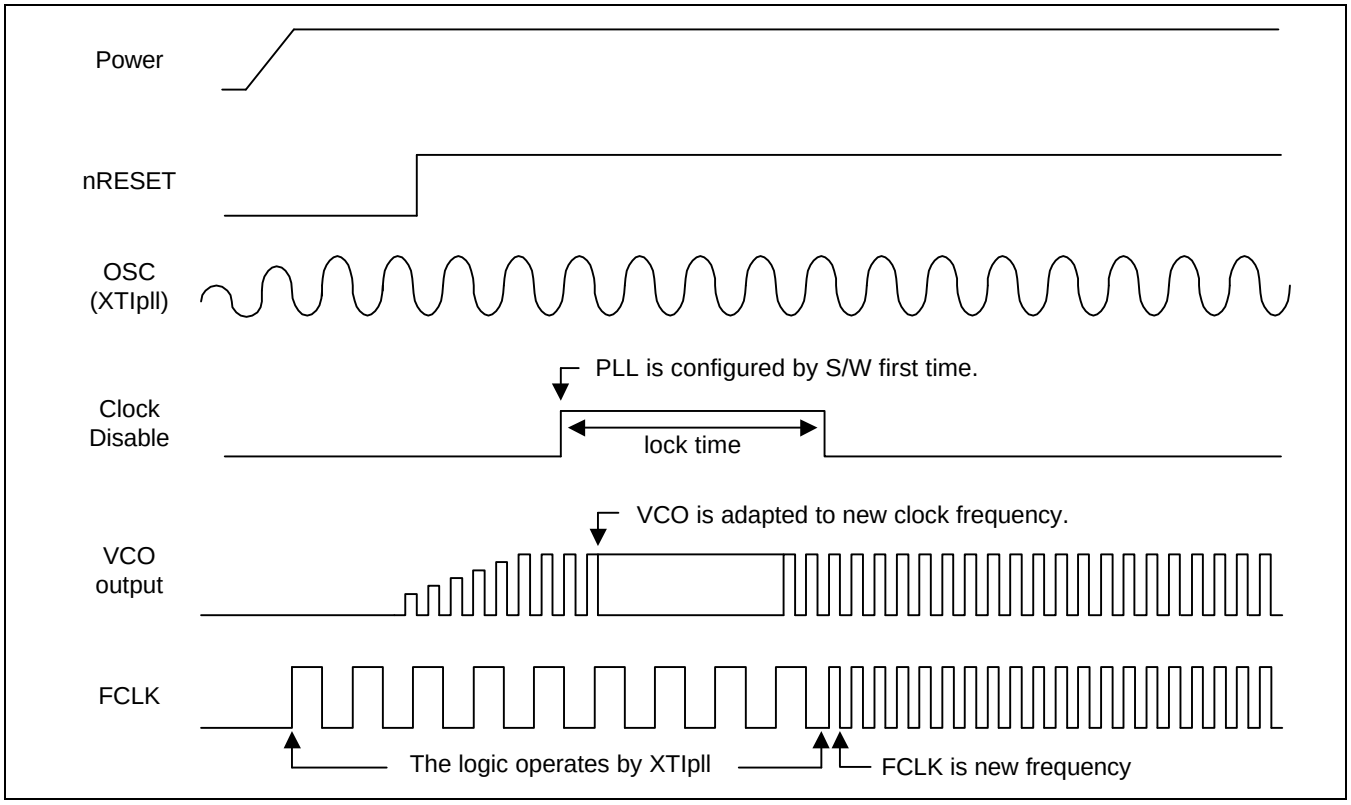


Figure 6-4. Power-On Reset Sequence (When the external clock source is a crystal oscillator.)

Change PLL Settings In Normal Operation Mode

During the operation of S3C2400X in NORMAL mode, if the user wants to change the frequency by writing the PMS value, the PLL lock time is automatically inserted. During the lock time, the clock is not supplied to the internal blocks in S3C2400X. The timing diagram is as follow.

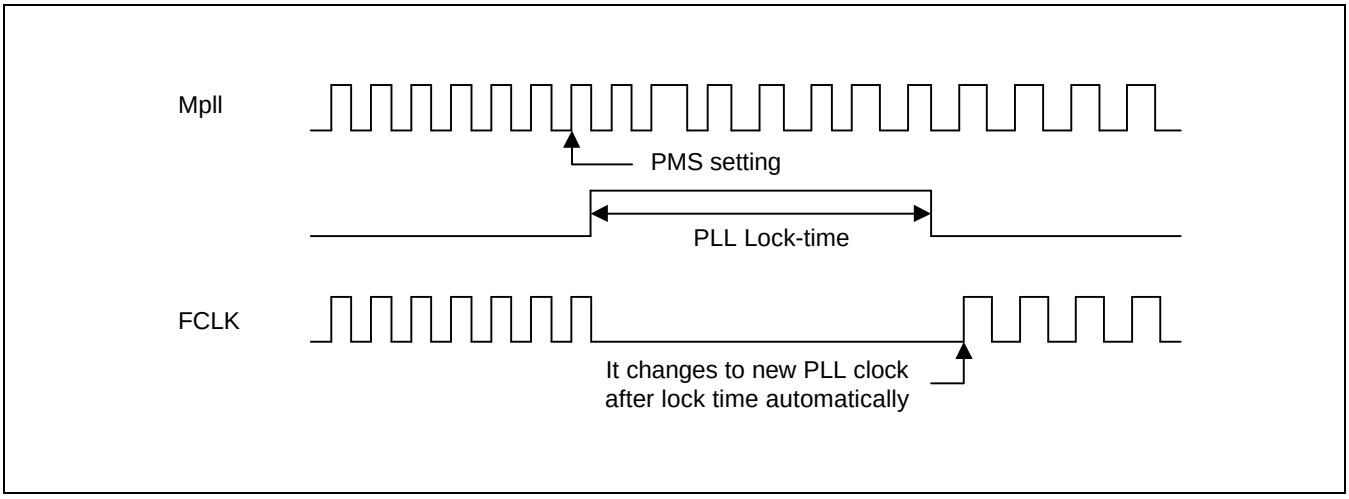


Figure 6-5. Timing Diagram of Changing FCLK

NOTE: Changing PMS value can cause problem in LCD display. Because changed PMS value means changed CLKVAL of LCD control register. In this case, the user has to use SLOW mode like the below.

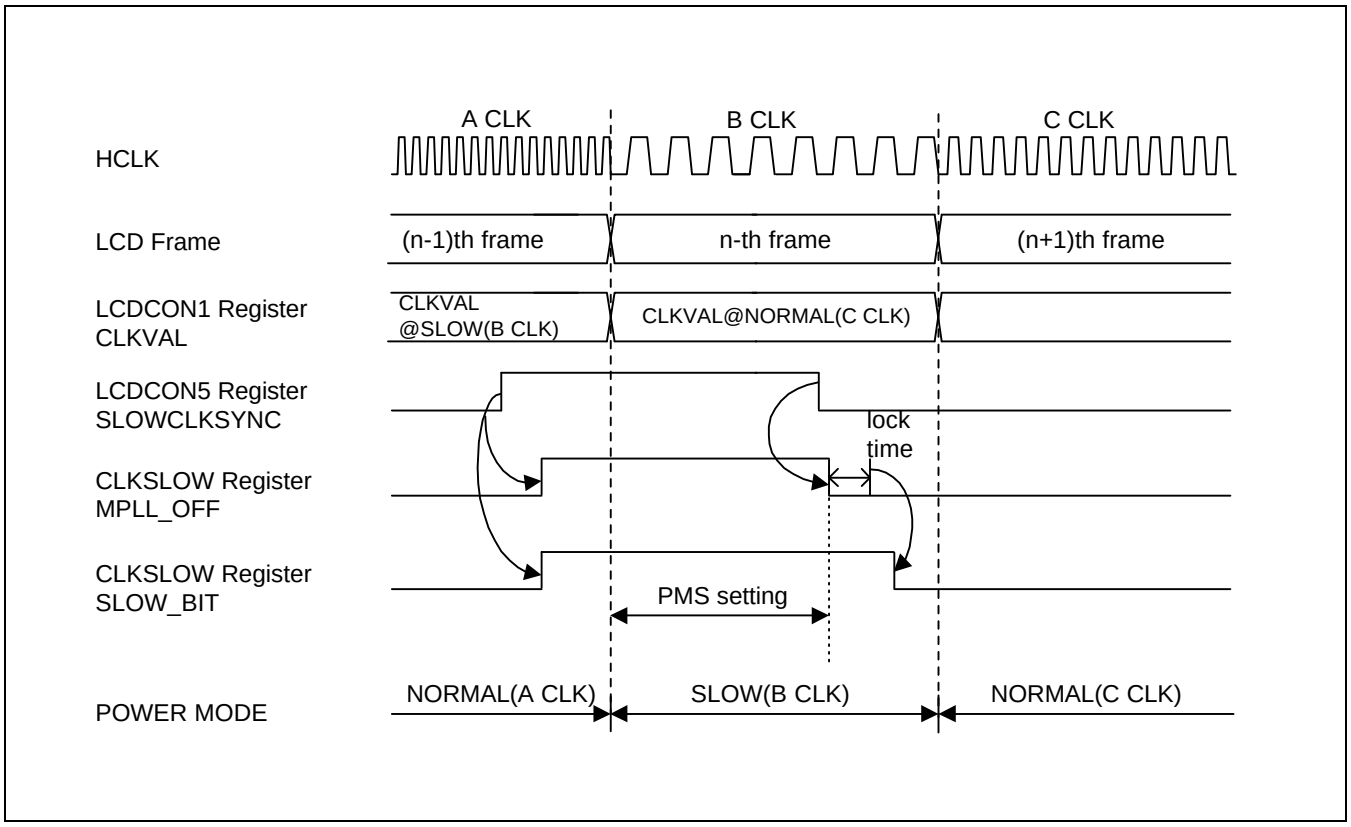


Figure 6-6. Timing Diagram of Changing PMS without the Interference of LCD Display

FCLK, HCLK, PCLK

FCLK is used by ARM920T.
HCLK is used for AHB bus which is used by ARM920T, the memory controller, the interrupt controller, 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.

S3C2400X supports selection of Dividing Ratio between FCLK, HLCK and PCLK. This ratio is determined by HDIVN and PDIVN of CLKDIVN control register.

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

When PMS value is set, CLKDIVN register should be set after PMS setting. These values of CLKDIVN are valid after PLL Lock-time. Which one of values is also available after reset and changing Power Management Mode. In other case, the setting value of CLKDIVN register is valid after 1.5 HCLK. But 1HCLK can be validated the value of CLKDIVN register changed from Default(1:1:1) to other Divide Ratio(1:1:2, 1:2:2 and 1:2:4)

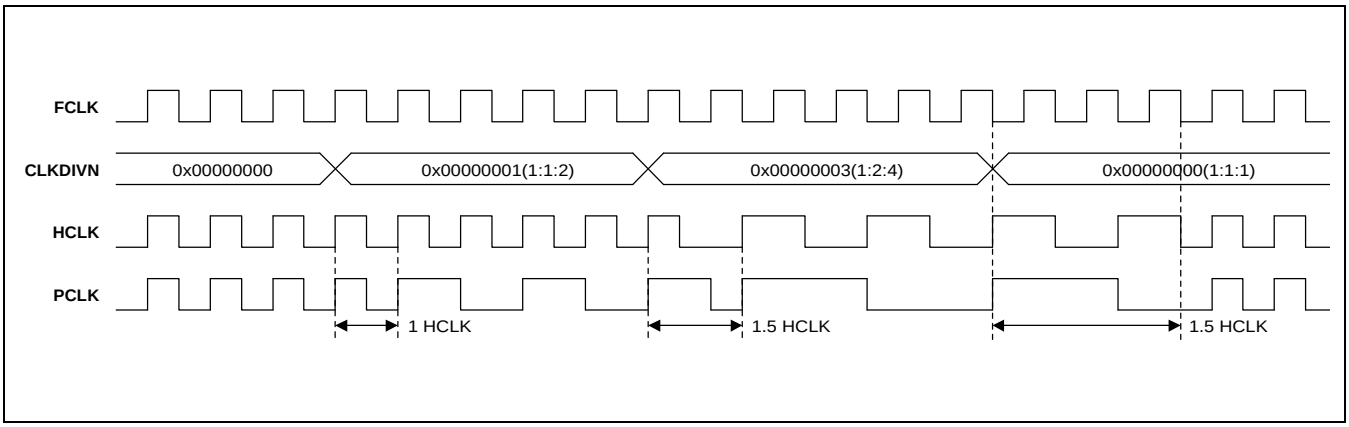


Figure 6-7. Example that changes CLKDIVN register value

NOTE: CLKDIVN should be set carefully not to exceed the limit of HCLK & PCLK (the limit of HCLK & PCLK are 66MHz).

USB Clock Control

USB host interface and USB device interface needs 48Mhz clock. In S3C2400X, The USB dedicated PLL(UPLL) generates 48Mhz for USB. UCLK doesn't fed until the PLL(UPLL) is configured. USB PLL(UPLL) will be turned off during SL_IDLE mode or STOP mode automatically. Also, USB PLL(UPLL) will be turned on after exiting SL_IDLE mode or STOP mode if UCLK_ON bit is enabled in CLKSLOW register.

Condition	UCLK state	UPLL state
After reset	XTI_{PLL} or EXTCLK	on
After configuring UPLL	L: during PLL lock time 48 MHz: after PLL lock time	on
UPLL is turned off by CLKSLOW register	XTI_{PLL} or EXTCLK	off
UPLL is turned on by CLKSLOW register	48 MHz	on

POWER MANAGEMENT

The power management block controls the system clocks by software for the reduction of power consumption in S3C2400X. These schemes are related to PLL, clock control logic(FCLK,HCLK,PCLK) and wake-up signal. The Figure 6-8 depicts the clock distribution of S3C2400X.

S3C2400X has five power-down modes. The following section describes each power managing mode. The transition between the modes is not allowed freely. For available transitions among the modes, please refer to Figure 6-9.

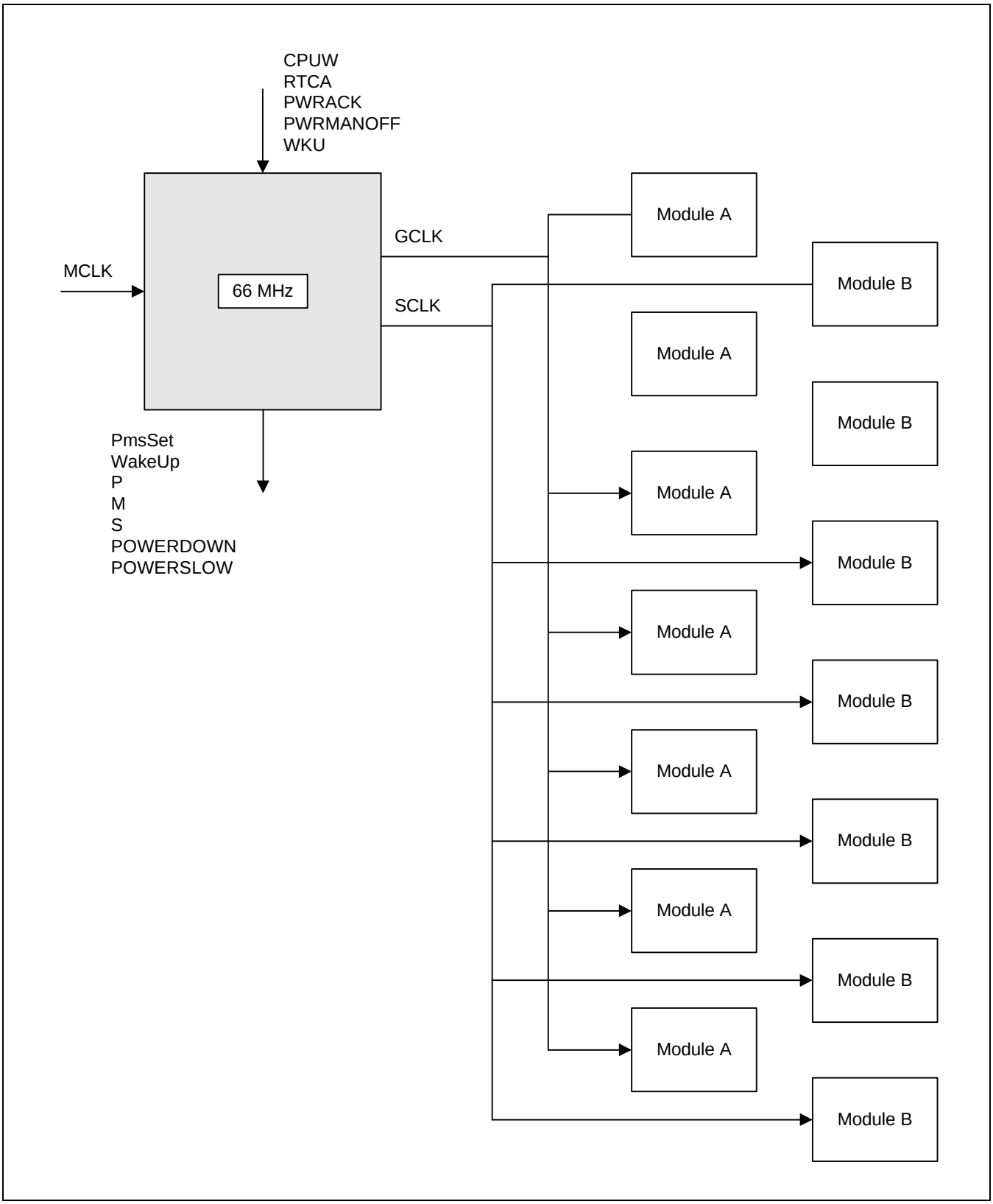


Figure 6-8. The Clock Distribution Block Diagram

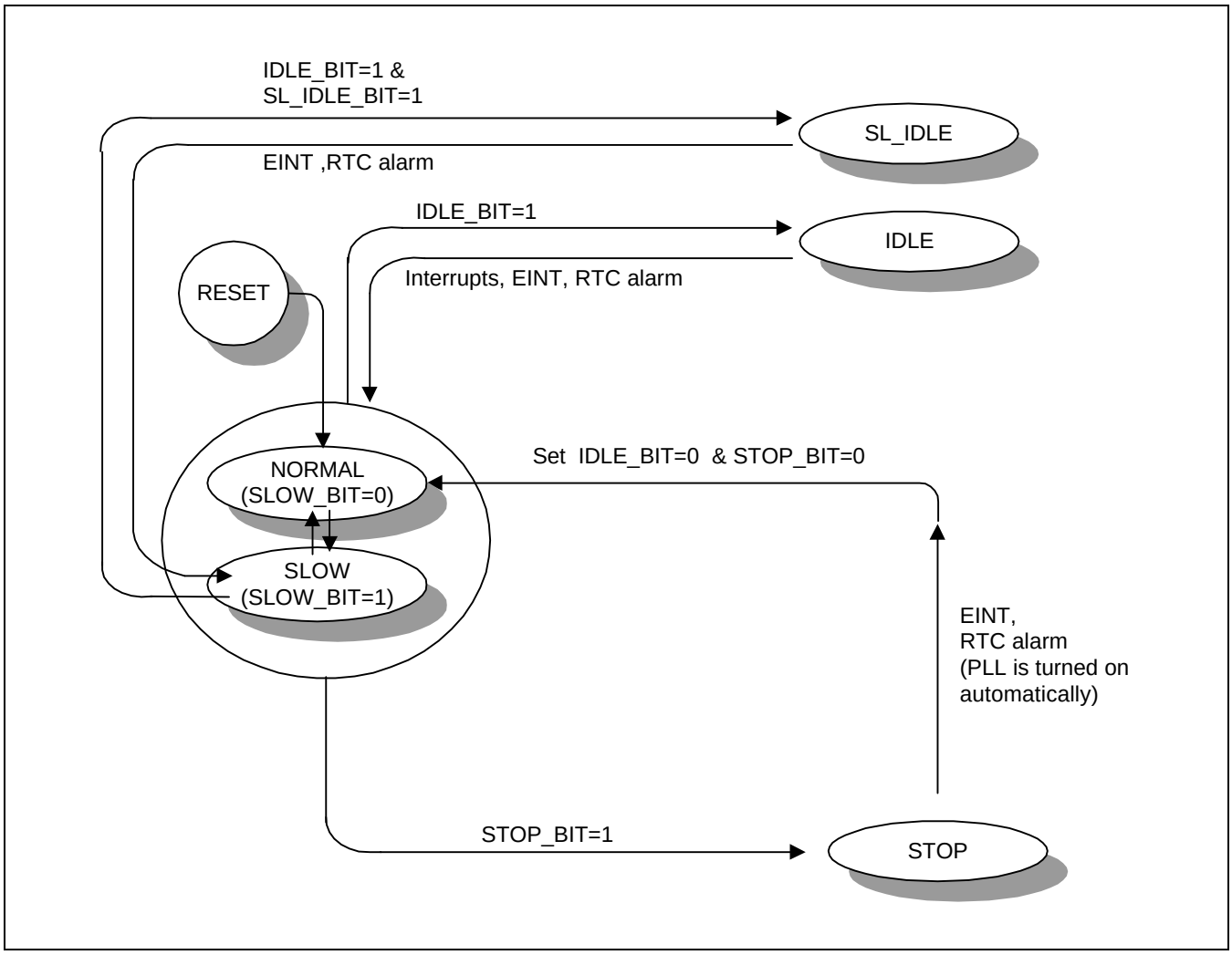


Figure 6-9. Power Management State Machine

Table 6-2. Functional Block Clock State In Each Power Mode

Mode	ARM920T	AHB Modules (1), WDT	LCD	APB Modules (2), USB host	UCLK
NORMAL	O	O	SEL	SEL	SEL
IDLE	X	O	SEL	SEL	SEL
SL_IDLE	X	X	O	X	SEL
STOP	X	X	X	X(3)	X
SLOW	O	O	SEL	SEL	SEL

NOTES:

1. USB host and RTC are excluded.
2. WDT is excluded.
3. RTC interface is turned off in STOP mode but RTC Timer is always turned on.
4. SEL: selectable, O: turned on, X: turned off

NORMAL Mode

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

IDLE Mode

In IDLE mode, the clock to CPU core is stopped except bus controller, memory controller, interrupt controller, and power management block. To exit IDLE mode, EINT, or RTC alarm interrupt, or the other interrupts should be activated. (If users are willing to use EINT, GPIO block has to be turned on before the activation).

STOP Mode

In STOP mode, all clocks are stopped for minimum power consumption. Therefore, the PLL and oscillator circuit are also stopped. Just after exiting the STOP mode, only NORMAL mode is available. In Figure 6-9, the user must return from STOP mode to NORMAL mode. To exit from STOP mode, EINT or RTC alarm has to be activated and CLKCON register is set properly.

— DRAM has to be in self-refresh mode during STOP mode to retain valid memory data.

— LCD must be stopped before STOP mode, because DRAM can't be accessed when it’s in self-refresh mode.

— All interrupts should be masked, because DRAM can't be accessed when it’s in self-refresh mode. (Even though all interrupts are masked, EINT can wake-up S3C2400X with the setting of EXINT register.)

— If MMU is turned on, TLB fill operation should not be allowed after entering STOP mode because SDRAM is entered the self-refresh mode. So, The TLB should be filled in advance. Please refer to our reference code.

The S3C2400X can exit from STOP mode by EINT(external interrupts) or RTC alarm. During the wake-up sequences, the crystal oscillator and PLL may begin to operate. The lock time is also needed to stabilize FCLK. The lock time is inserted automatically and guaranteed by power management logic. During this lock time the clock is not supplied. Just after wake-up sequences wake-up interrupt(RTC alarm or external interrupt) is requested.

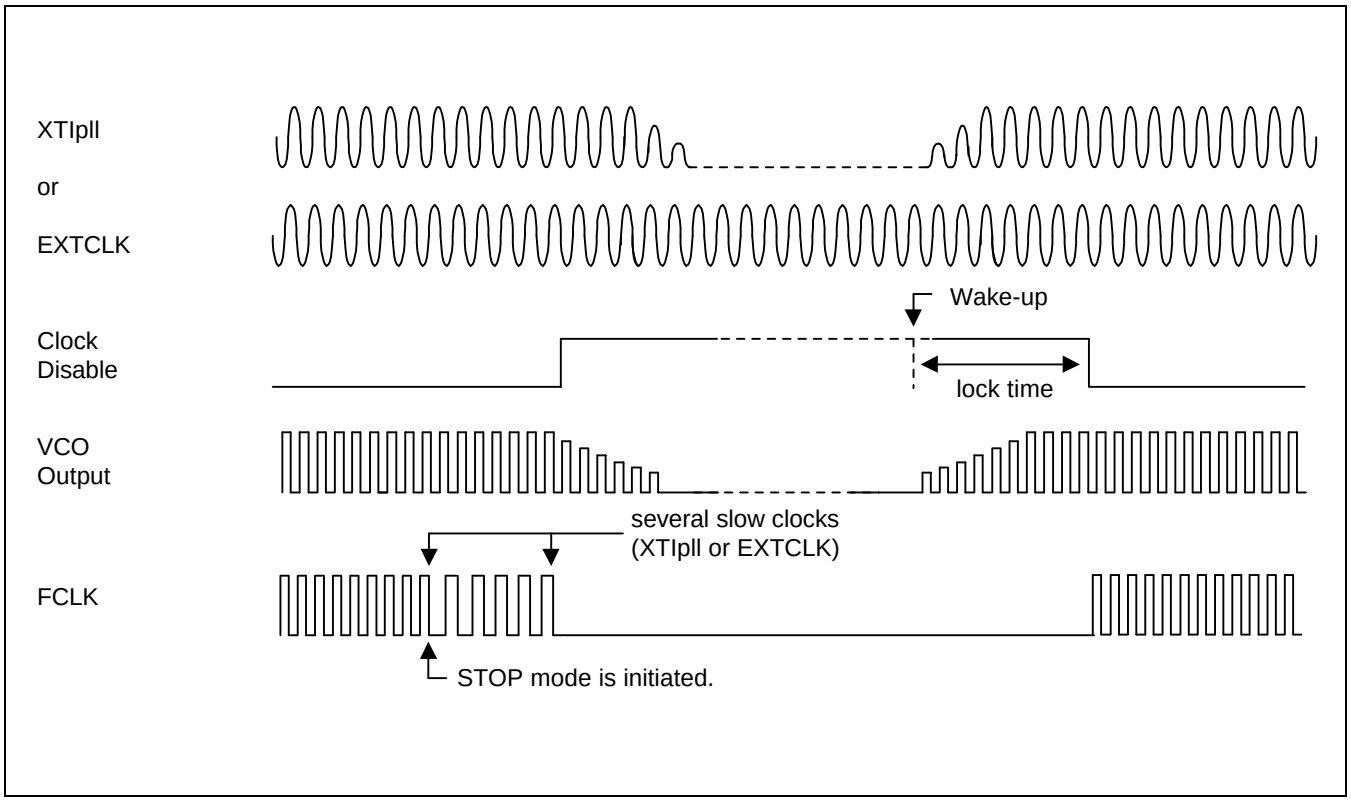


Figure 6-10. Entering STOP Mode and Exiting STOP mode (Wake-up)

SL_IDLE Mode (S_LCD Mode) for SED1580D

Only SED1580D(mono STN LCD controller, Seiko Epson) supports SL_IDLE mode. In SL_IDLE mode, the clock to the basic blocks is stopped. Only the LCD controller is working to maintain the LCD screen. Less power is consumed in the SL_IDLE mode than in the IDLE mode.

SDRAM has to be in self-refresh mode during SL_IDLE mode to retain the valid data in DRAM.

Procedure for Entering SL_IDLE Mode.

1. Mask all interrupts.
2. Check the LCD line counter(LINECNT) whether or not sufficient time is remained until entering SLOW mode. LCD frame should not be changed until STEP 4.
3. Set SLOWCLKSYNC bit in LCD control register
4. Wait until the LCD line counter reaches the self-refresh line. (The LCDCON5:SELFREF bit should be enabled at 4n+1th line to enable the LCD self-refresh at 4n+0th line.) At the moment that the LCD line reaches the LCD self-refresh line, turn on LCDCON5:SELFREF bit.
5. Reconfigure LCDCON1 for SLOW clock. These new parameter will be effective from next LCD frame. LCDCON2,3,4 can't be changed.
6. Wait until the LCD self-refresh command is effective.
7. Turn on SLOW mode. The SLOW mode will not be effective until the current LCD frame is completed.
8. As TLB fill operation can't be occurred after SDRAM self-refresh mode is effective. So, fill the TLB in advance like our example code.
9. Wait until the line counter reaches 0.
10. While the line counter is 0, SL_IDLE mode should be enabled by setting CLKCON resister.
11. Wait while the line counter is 0.
12. The SL_IDLE mode will be effective just after the current frame is completed.

Procedure for Exiting from SL_IDLE Mode

1. SL_IDLE mode is woken up by EINT or alarm interrupt. Just after wake-up, the operating mode is SLOW mode.
2. Change the CLKCON as normal mode(Actually, SLOW mode).
3. Check the LCD line counter whether sufficient time is remained or not until exiting SLOW mode. LCD frame should not be changed until STEP 4.
4. Turn on MPLL if the MPLL is turned off.
5. Configure LCD parameters for NORMAL clock. These new parameter will be effective from next LCD frame.
6. Wait until the line counter is 0.
7. Turn off SLOW mode. The NORMAL mode will not be effective until the current LCD frame is completed.
8. Wait until the current frame is completed.
9. Clear SLOWCLKSYNC bit in LCD control register.
10. Wait until NORMAL mode is effective by polling the LCD line counter.
11. Wait until the LCD line counter reaches the self-refresh line. (The LCDCON5:SELFREF bit should be disabled at 4n+1th line to disable the LCD self-refresh at 4n+0th line.) At the moment that the LCD line reaches the LCD self-refresh line, turn off LCDCON5:SELFREF bit.
12. Unmask interrupts. The interrupt(wake-up source) will be generated.

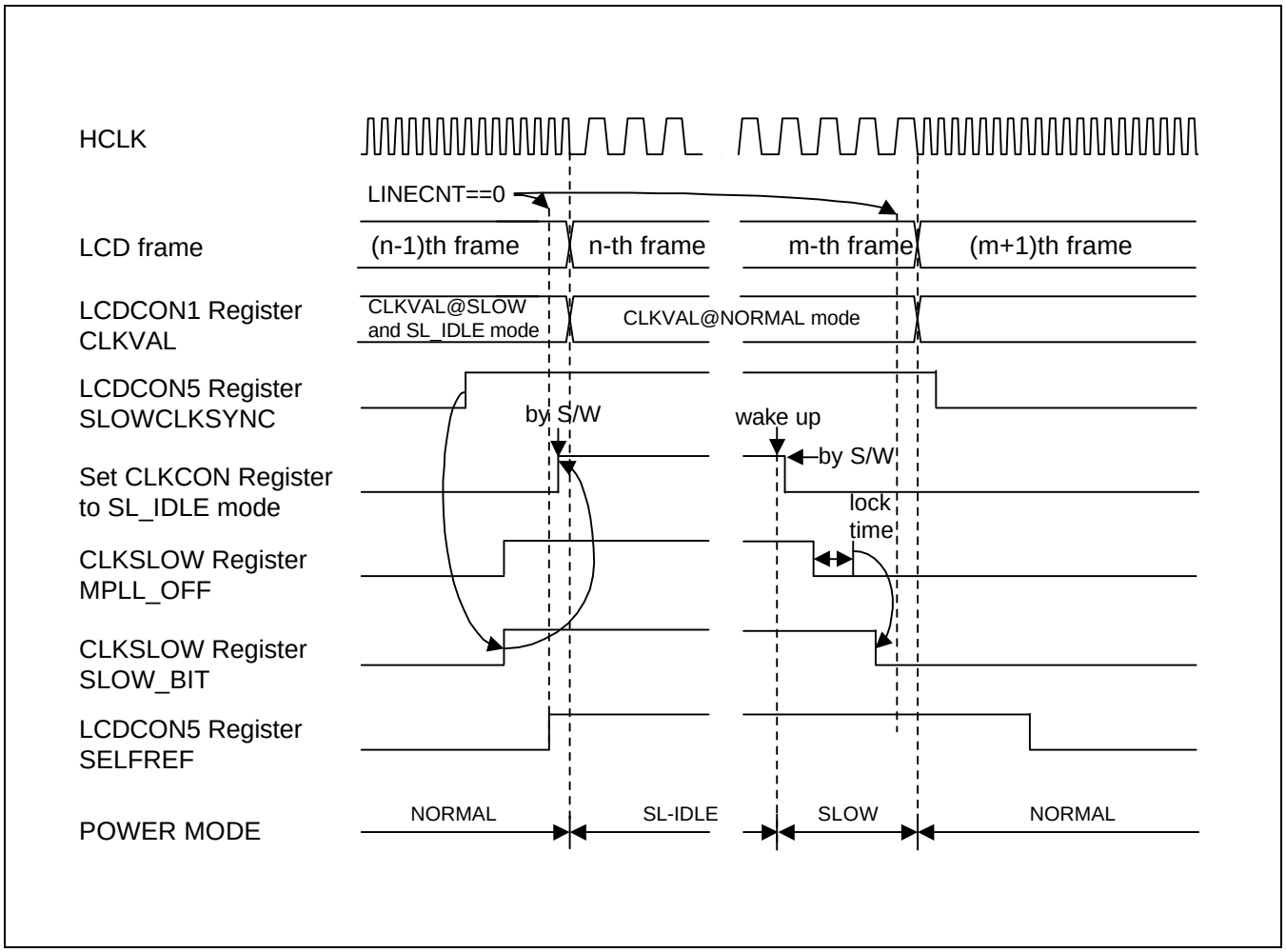


Figure 6-11. Entering SL_IDLE Mode and Exiting from SL_IDLE Mode (Wake-up)

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, itself. 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 6-3. 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	48MHz
0 0 1	EXTCLK or XTIpll / 2	EXTCLK or XTIpll / 2	EXTCLK or XTIpll / 4	HCLK	HCLK / 2	48MHz
0 1 0	EXTCLK or XTIpll / 4	EXTCLK or XTIpll / 4	EXTCLK or XTIpll / 8	HCLK	HCLK / 2	48MHz
0 1 1	EXTCLK or XTIpll / 6	EXTCLK or XTIpll / 6	EXTCLK or XTIpll / 12	HCLK	HCLK / 2	48MHz
1 0 0	EXTCLK or XTIpll / 8	EXTCLK or XTIpll / 8	EXTCLK or XTIpll / 16	HCLK	HCLK / 2	48MHz
1 0 1	EXTCLK or XTIpll / 10	EXTCLK or XTIpll / 10	EXTCLK or XTIpll / 20	HCLK	HCLK / 2	48MHz
1 1 0	EXTCLK or XTIpll / 12	EXTCLK or XTIpll / 12	EXTCLK or XTIpll / 24	HCLK	HCLK / 2	48MHz
1 1 1	EXTCLK or XTIpll / 14	EXTCLK or XTIpll / 14	EXTCLK or XTIpll / 28	HCLK	HCLK / 2	48MHz

In SLOW mode, the PLL will be turned off to reduce the PLL power consumption. When PLL is turned off in SLOW mode and users change 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 PLL is turn on. During PLL lock time, the FCLK is 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 SLOW mode. The timing diagram is in Figure 6-12.

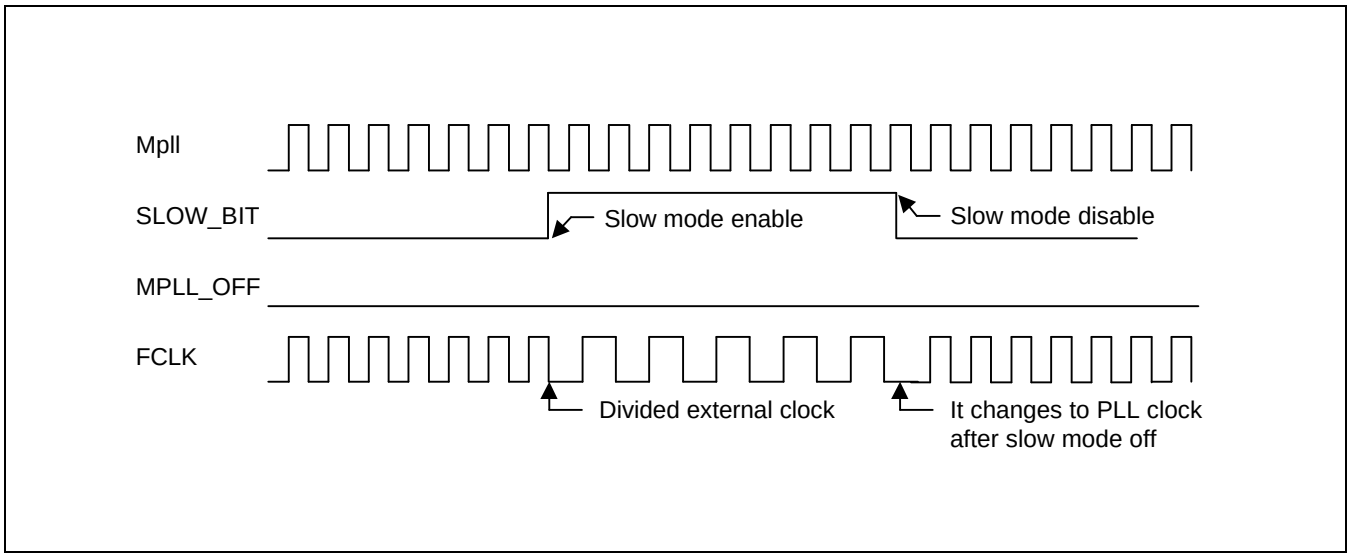


Figure 6-12. The case that Exit_from_Slow_mode command is issued in PLL on state

If users exit 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. The timing diagram is in Figure 6-13.

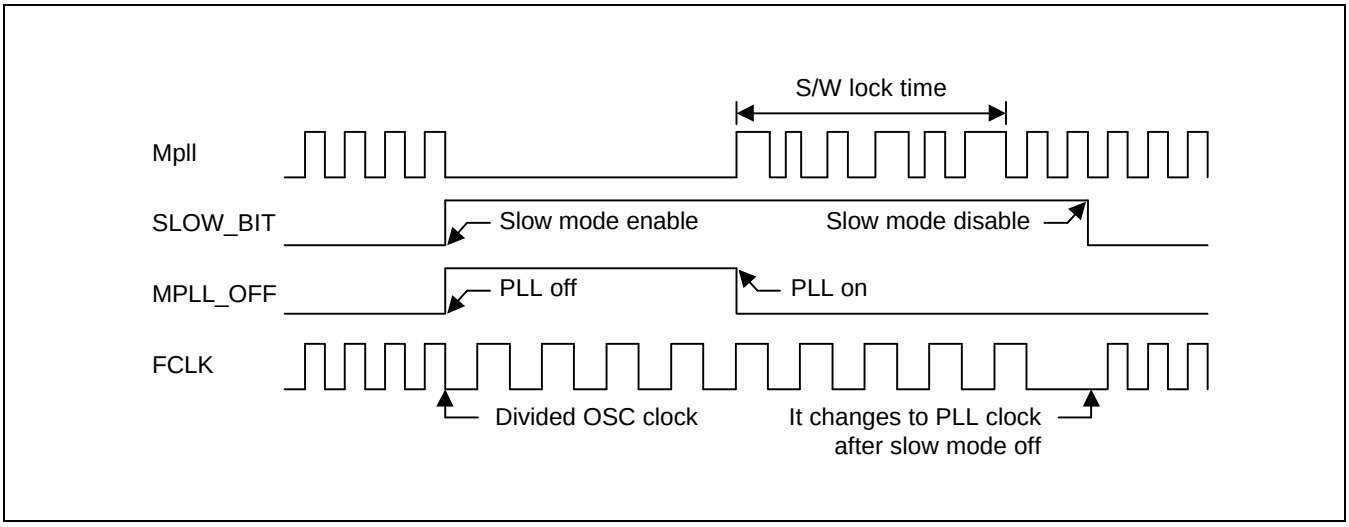


Figure 6-13. The Case that Exit_from_Slow_Mode Command is Issued after Lock Time is End

If users exit from SLOW mode to Normal mode by disabling SLOW_BIT and MPLL_OFF bit simultaneously in CLKSLOW register, the frequency is changed just after the PLL lock time. The timing diagram is as follow.

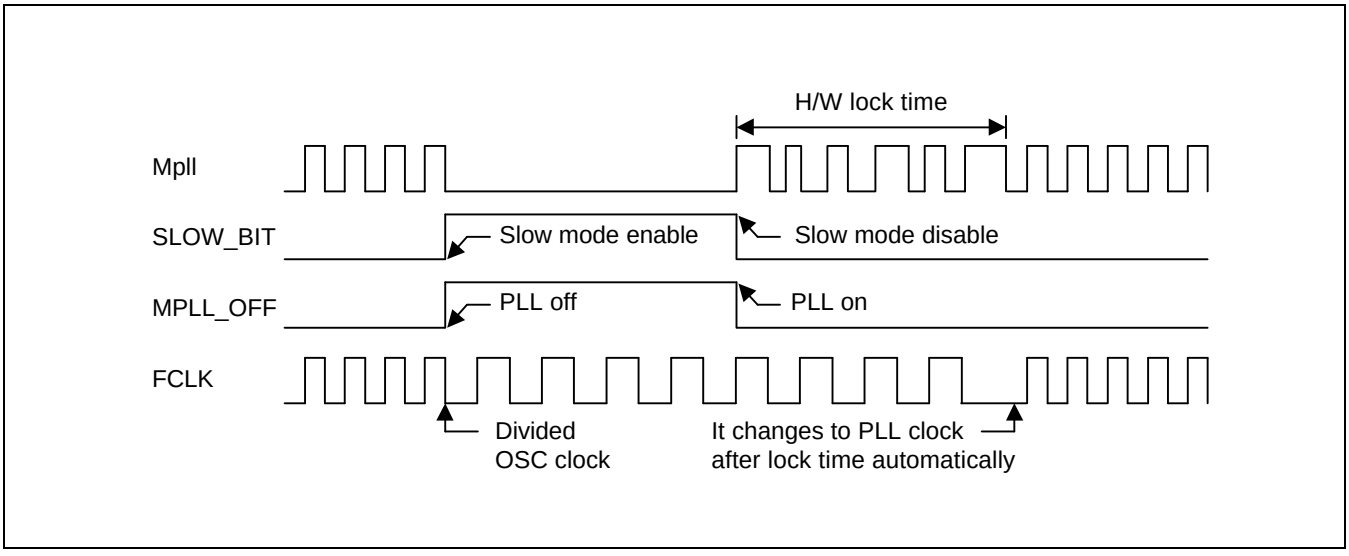


Figure 6-14. The Case that the Exit_from_Slow_Mode Command and the Instant PLL_on Command is Issued Aimultaneously

Wake-Up from STOP Mode

When the S3C2400X is woken up from power down mode(STOP mode) by an EINT or a RTC alarm interrupt, the PLL is turned on automatically. But S3C2400X is not in NORMAL mode yet. Because the configuration of the CLKCON is ignored. So, the user has to set CLKCON register.

After the wake-up from STOP mode, the processor is not in NORMAL mode as explained above. The new value, which reflects the new state, has to be re-written into the CLKCON register. Eventually, the processor state will be changed from STOP mode to Normal mode.

Table 6-4. The Status of PLL and FCLK After Wake-Up

Mode before Wake-up	PLL On/Off after Wake- up	FCLK after Wake-up and before the Lock Time	FCLK after the Lock Time by Internal Logic
STOP	off → on	no clock	normal mode clock
SL_IDLE	off	SLOW clock (the PLL lock time is issued by S/W)	unchanged
IDLE	unchanged	unchanged	unchanged

Signaling EINT For Wake-Up

The S3C2400X can be woken up from SL_IDLE mode or STOP mode only if the following conditions are met.

a) Level signal(H or L) or edge signal(rising or falling or both) is asserted on EINTn input pin.
b) EINTn pin has to be configured as EINT in the GPIO control register.

It is important to configure the EINTn in the GPIO control register as an external interrupt pins. For wake-up, we need H/L level or rising/falling edge or both edge signals on EINTn pin.

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

Entering IDLE Mode

If CLKCON[2] is set to 1 to enter the IDLE mode, S3C2400X will enter into IDLE mode after some delay(until when the power control logic receives ACK signal from the CPU wrapper).

PLL On/Off

The PLL can only be turned off for power saving in slow mode. If 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 requiring that PLL be turned on, then SLOW_BIT should be clear to move to another state after PLL stabilization.

PnUPs Register and STOP/SL_IDLE Mode

In STOP mode, the data bus(D[31:0] or D[15:0]) is Hi-z state.

But, because of the characteristics of I/O pad, the data bus pull-up resistors have to be turned on to reduce the power consumption in STOP/SL_IDLE mode. D[31:16] pin pull-up resistors can be controlled by GPIO control registers. D[15:0] pin pull-up resistors can be controlled by the GPIO control register.

OUTPUT PORT State and STOP/SL_IDLE mode

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. If a port is configured as an output port, the current consumption can be reduced if the output state is H.

The output ports are recommended to be in H state to reduce STOP mode current consumption.

ADC Power Down

The ADC has an additional power-down bit(STDBM) in ADCCON. If S3C2400X enters the STOP mode, the ADC should enter it's own power-down mode.

CLOCK GENERATOR & POWER MANAGEMENT SPECIAL REGISTER

LOCK TIME COUNT REGISTER (LOCKTIME)

Register	Address	R/W	Description	Reset Value
LOCKTIME	0x14800000	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,PCLK (M_LTIME>150uS)	0xfff

PLL CONTROL REGISTER (MPLLCON, UPLLCON)

Mpll = (m * Fin) / (p * 2^S)
m = (MDIV + 8), p = (PDIV + 2), s = SDIV

PLL VALUE SELECTION GUIDE

1. Fout = m * Fin / p*s, where : m=M+8, p=P+2, s=2^s

2. Fin/(25*p) < 16.7e6/m < Fin/(10*p)

3. 0.7 < 6.48/sqrt(m) < 1.8

4. (Fin/p) * m < 330e6

Register	Address	R/W	Description	Reset Value
MPLLCON	0x14800004	R/W	MPLL configuration register	0x0005c080
UPLLCON	0x14800008	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

Caution: When you set MPLL&UPLL values simultaneously, MPLL value first and then UPLL value should be set.

PLL VALUE SELECTION GUIDE

It's difficult to find the proper PLL value. So, We recommended 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	37	1	3
12.00Mhz	33.75Mhz	37	0	3
12.00Mhz	44.25Mhz	51	2	2
12.00Mhz	48.00Mhz	40	1	2
12.00Mhz	51.00Mhz	43	1	2
12.00Mhz	56.00Mhz	48	1	2
12.00Mhz	67.50Mhz	37	0	2
12.00Mhz	79.00Mhz	71	1	2
12.00Mhz	84.00Mhz	34	1	1
12.00Mhz	90.00Mhz	37	1	1
12.00Mhz	102.00Mhz	43	1	1
12.00Mhz	112.50Mhz	67	2	1
12.00Mhz	118.50Mhz	71	2	1
12.00Mhz	124.00Mhz	54	1	1
12.00Mhz	133.50Mhz	81	2	1

CLOCK CONTROL REGISTER (CLKCON)

Register	Address	R/W	Description	Reset Value
CLKCON	0x1480000C	R/W	Clock generator control Register	0xfff8

CLKCON	Bit	Description	Initial State
SPI	[15]	Controls PCLK into SPI block 0 = Disable, 1 = Enable	1
IIS	[14]	Controls PCLK into IIS block 0 = Disable, 1 = Enable	1
IIC	[13]	Controls PCLK into IIC block 0 = Disable, 1 = Enable	1
ADC	[12]	Controls PCLK into ADC block 0 = Disable, 1 = Enable	1
RTC	[11]	Controls PCLK into RTC control block. Even if this bit is cleared to 0, RTC timer is alive. 0 = Disable, 1 = Enable	1
GPIO	[10]	Controls PCLK into GPIO block 0 = Disable, 1 = Enable	1
UART1	[9]	Controls PCLK into UART1 block 0 = Disable, 1 = Enable	1
UART0	[8]	Controls PCLK into UART0 block 0 = Disable, 1 = Enable	1
MMC	[7]	Controls PCLK into MMC interface block 0 = Disable, 1 = Enable	1
PWMTIMER	[6]	Controls PCLK into PWMTIMER block 0 = Disable, 1 = Enable	1
USB device	[5]	Controls PCLK into USB device block 0 = Disable, 1 = Enable	1
USB host	[4]	Controls HCLK into USB host block 0 = Disable, 1 = Enable	1
LCDC	[3]	Controls HCLK into LCDC block 0 = Disable, 1 = Enable	1
IDLE BIT	[2]	Enters IDLE mode. This bit isn't be cleared automatically. 0 = Disable, 1 = Transition to IDLE(SL_IDLE) mode	0
SL_IDLE	[1]	SL_IDLE mode option. This bit isn't be cleared automatically. 0 = Disable, 1 = SL_IDLE mode. To enter SL_IDLE mode, CLKCON register has to be 0xe.	0
STOP BIT	[0]	Enters STOP mode. This bit isn't be cleared automatically. 0 = Disable 1 = Transition to STOP mode	0

CLOCK SLOW CONTROL REGISTER (CLKSLOW)

Register	Address	R/W	Description	Reset Value
CLKSLOW	0x14800010	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
SLOW_VAL	[2:0]	The divider value for the slow clock when SLOW_BIT is on.	0x4

CLOCK DIVIDER CONTROL REGISTER (CLKDIVN)

Register	Address	R/W	Description	Reset Value
CLKDIVN	0x14800014	R/W	Clock divider control register	0x00000000

CLKDIVN	Bit	Description	Initial State
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

NOTES

7 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

In S3C2400X, there are eleven bus masters, i.e., DRAM refresh controller, LCD_DMA, DMA0, DMA1, DMA2, DMA3, USB_HOST_DMA, EXT_BUS_MASTER, TIC (Test interface controller), and ARM920T. The priorities among these bus masters after a reset are as follows:

1. DRAM 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

8 DMA

OVERVIEW

S3C2400X 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) source is in the system bus while destination is in the peripheral bus, 3) source is in the peripheral bus while destination is in the system bus, 4) both source and destination are in the peripheral bus.

The main advantage of DMA is that it can transfer the data without CPU intervention. The operation of DMA can be initiated by S/W, the request from internal peripherals or the external request pins.

DMA REQUEST SOURCES

Each channel of 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.) The four DMA sources for each channel are as follows.

Table 8-1. DMA request sources for each channel

	Source0	Source1	Source2	Source3
Ch-0	nXDREQ0	UART0	MMC	Timer
Ch-1	nXDREQ1	UART1	I2SSDI	SPI
Ch-2	I2SSDO	I2SSDI	MMC	Timer
Ch-3	USB device	MMC	SPI	Timer

Here, nXDREQ0 and nXDREQ1 represent two external sources(External Devices), and I2SSDO and I2SSDI represent IIS transmitting and receiving, respectively.

DMA OPERATION

The details of DMA operation can be explained using three-state FSM(finite state machine) as follows:

State-1. As an initial state, it waits for the DMA request. If it comes, go 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 DMA ACK becomes 1 and 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 the whole service mode, while performed only once in a 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[28] register is set to 1. In addition, it clears DMA ACK if one of the following conditions are 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 de-asserted 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 de-asserted 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 four types of external DMA request/acknowledge protocols. Each type defines how the signals like DMA request and acknowledge are related to these protocols.

Basic DMA Timing

The DMA service means paired Reads and Writes cycles during DMA operation, which is one DMA operation. The Figure 8-1 shows the basic Timing in the DMA operation of the S3C2400X.

— The setup time and the delay time of XnXDREQ and XnXDACK are 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 finishes.

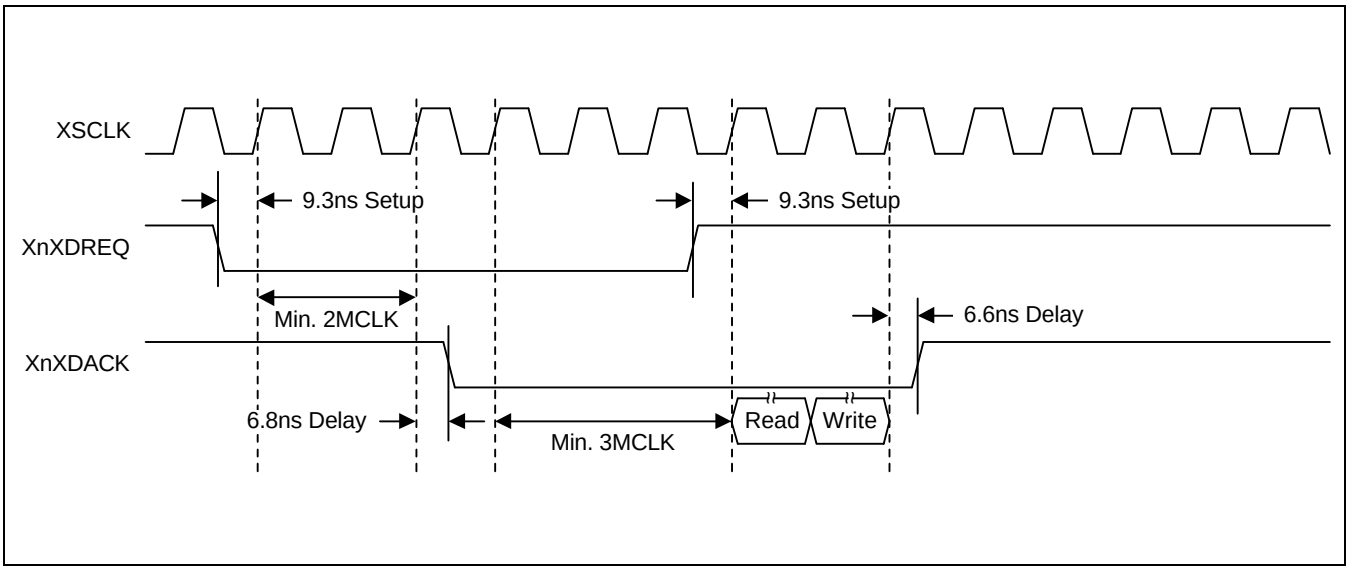


Figure 8-1. Basic DMA Timing Diagram

Demand/Handshake Mode Comparison – Related to the Protocol between XnXDREQ and XnXDACK

These are two different modes related to the protocol between XnXDREQ and XnXDACK. Fig. 8-2 shows the differences between these two modes i.e., Demand and Handshake 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.

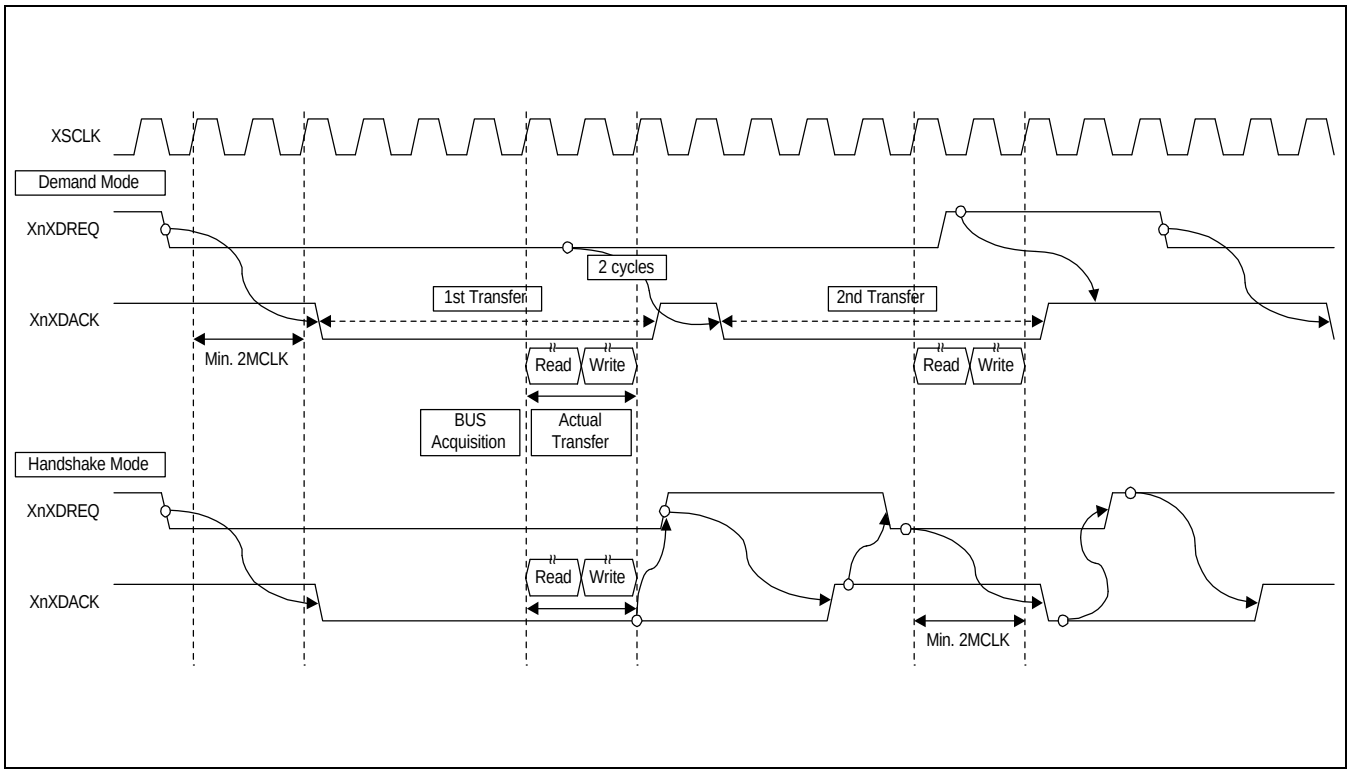


Figure 8-2. Demand/Handshake Mode Comparison

Transfer Size

— There are two different transfer sizes; single and Burst 4.

— DMA holds the bus firmly during the transfer of these chunk of data, thus other bus masters can not get the bus.

Burst 4 Transfer Size

4 sequential Reads and 4 sequential Writes are performed in the Burst 4 Transfer.

NOTE: Single Transfer size : One read and one write are performed.

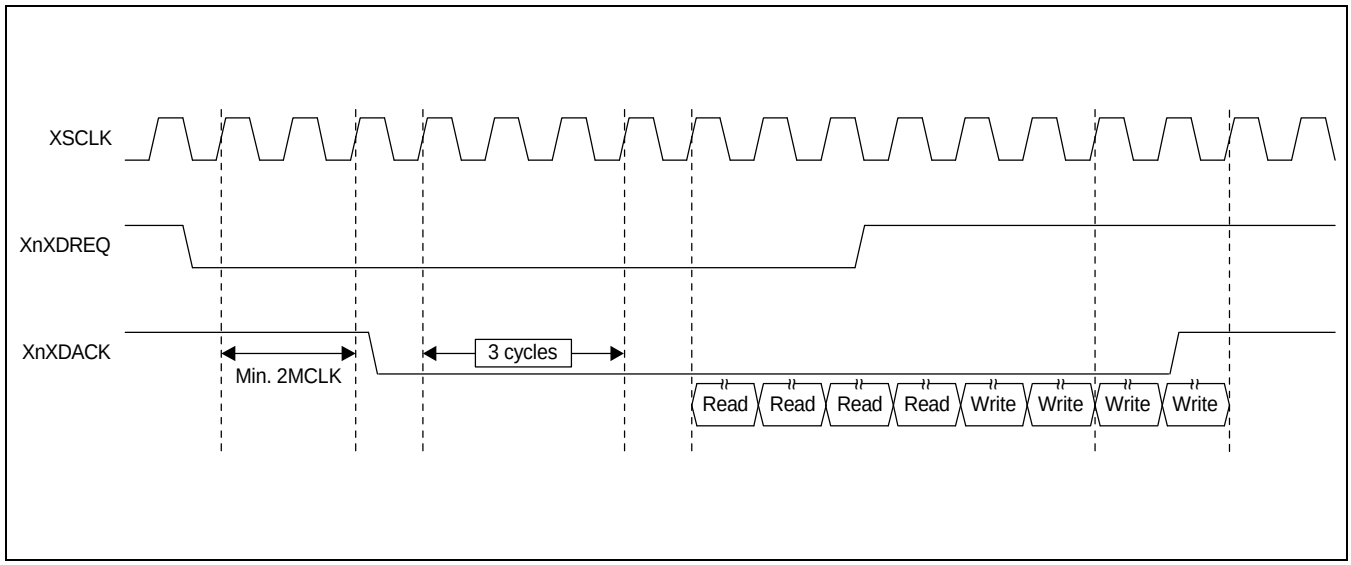


Figure 8-3. Burst 4 Transfer Size

EXAMPLES OF POSSIBLE CASES

Single service, Demand Mode, Single 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.

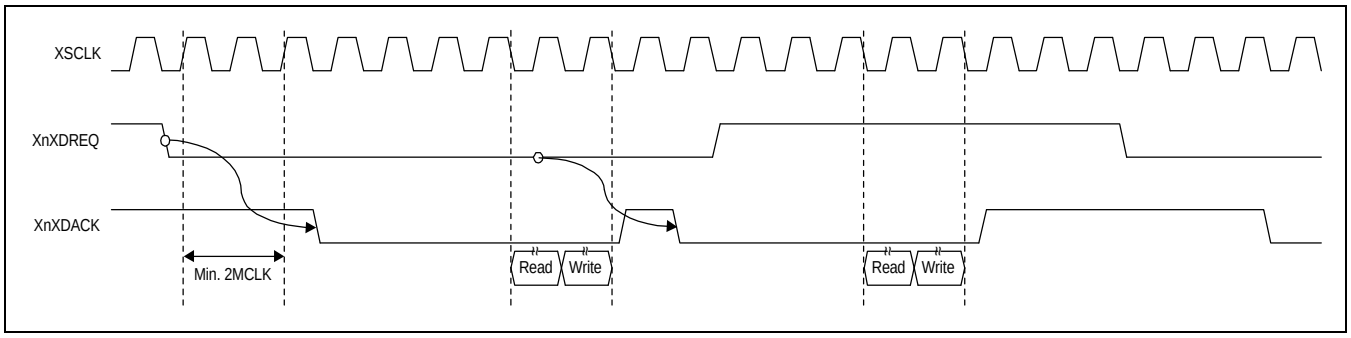


Figure 8-4. Single service, Demand Mode, Single Transfer Size

Single service/Handshake Mode, Single Transfer Size

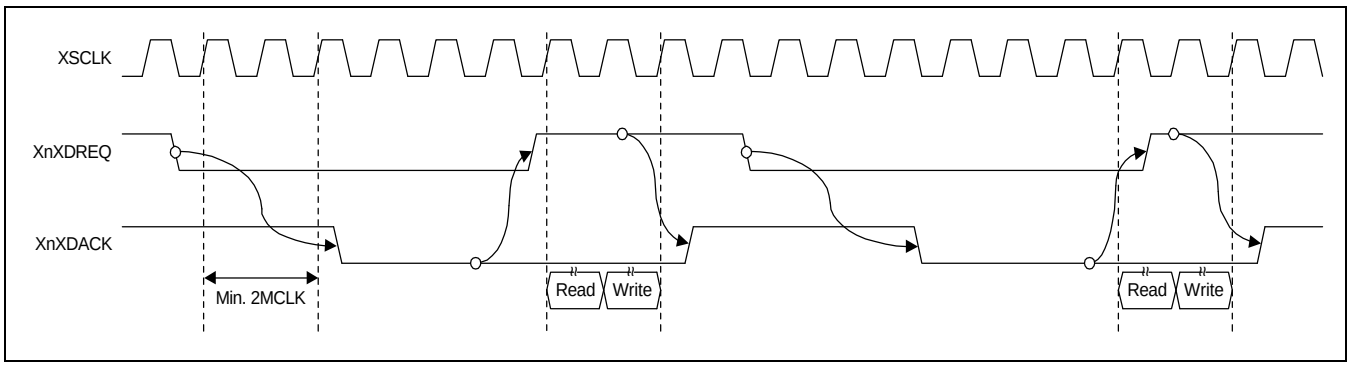


Figure 8-5. Single Service, Handshake Mode, Single Transfer Size

Whole service/Handshake Mode, Single Transfer Size

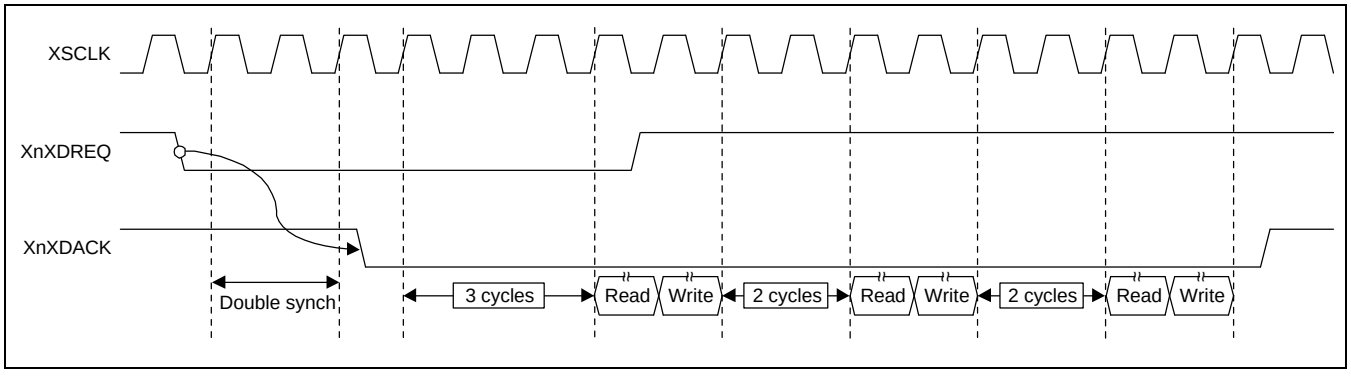


Figure 8-6. Whole Service, Handshake Mode, Single Transfer Size

DMA SPECIAL REGISTERS

There are seven control registers for each DMA channel. (Since there are four channels, the total number of control registers is 28.) Four of them are to control the DMA transfer, and other three are to see the status of DMA controller. The details of those registers are as follows.

DMA INITIAL SOURCE REGISTER (DISRC)

Register	Address	R/W	Description	Reset Value
DISRC0	0x14600000	R/W	DMA 0 Initial Source Register	0x00000000
DISRC1	0x14600020	R/W	DMA 1 Initial Source Register	0x00000000
DISRC2	0x14600040	R/W	DMA 2 Initial Source Register	0x00000000
DISRC3	0x14600060	R/W	DMA 3 Initial Source Register	0x00000000

DISRCn	Bit	Description	Initial State
LOC	[30]	Bit 30 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	[29]	Bit 29 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
S_ADDR	[28:0]	These bits are the base address (start address) of source data to transfer. This value will be loaded into CURR_SRC only if the CURR_SRC is 0 and the DMA ACK is 1.	0x00000000

DMA INITIAL DESTINATION REGISTER (DIDST)

Register	Address	R/W	Description	Reset Value
DIDST0	0x14600004	R/W	DMA 0 Initial Destination Register	0x00000000
DIDST1	0x14600024	R/W	DMA 1 Initial Destination Register	0x00000000
DIDST2	0x14600044	R/W	DMA 2 Initial Destination Register	0x00000000
DIDST3	0x14600064	R/W	DMA 3 Initial Destination Register	0x00000000

DIDSTn	Bit	Description	Initial State
LOC	[30]	Bit 30 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	[29]	Bit 29 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
D_ADDR	[28:0]	These bits are the base address (start address) of destination for the transfer. This value will be loaded into CURR_SRC only if the CURR_SRC is 0 and the DMA ACK is 1.	0x00000000

DMA CONTROL REGISTER (DCON)

Register	Address	R/W	Description	Reset Value
DCON0	0x14600008	R/W	DMA 0 Control Register	0x00000000
DCON1	0x14600028	R/W	DMA 1 Control Register	0x00000000
DCON2	0x14600048	R/W	DMA 2 Control Register	0x00000000
DCON3	0x14600068	R/W	DMA 3 Control Register	0x00000000

DCONn	Bit	Description	Initial State
DMD_HS	[30]	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 two modes is whether it waits for the de-asserted DACK or not. In handshake mode, DMA controller waits for the de-asserted DREQ before starting a new transfer. If it sees the de-asserted DREQ, it de-asserts DACK and waits for another asserted DREQ. In contrast, in the demand mode, DMA controller does not wait until the DREQ is de-asserted. It just de-asserts 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	[29]	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, devices attached to AHB system bus, this bit has to be set to 1, while those attached to APB system, it should be set to 0. For the devices attached to external system, user should select this bit depending on whether the external system is synchronized with AHB system or APB system.	0
INT	[28]	Enable/Disable the interrupt setting for CURR_TC(terminal count) 0 = CURR_TC interrupt is disabled. user has to look 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	[27]	Select the transfer size of an atomic transfer (i.e., transfer performed at 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

DCONn	Bit	Description	Initial State
SERVMODE	[26]	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 is not required. Here, 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	[25:24]	Select DMA request source for each DMA. DCON0: 00: nXDREQ0 01:UART0 10:MMC 11:Timer DCON1: 00: nXDREQ1 01:UART1 10:I2SSDI 11:SPI DCON2: 00:I2SSDO 01:I2SSDI 10:MMC 11:Timer DCON3: 00:USB device 01:MMC 10:SPI 11:Timer This bits control the 4-1 MUX to select the DMA request source of each DMA. These bits have meanings if and 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[25:24] is used to trigger the DMA operation.	0
RELOAD	[22]	Set the auto-reload on/off option. 0 = auto reload is performed when the following 2 conditions - The current value of transfer counter becomes 0 - And then, the next first DMA request is received. 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, and TC represent data size (DCONn[21:20]), transfer size (DCONn[27]), 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 REGISTER (DSTAT)

Register	Address	R/W	Description	Reset Value
DSTAT0	0x1460000c	R	DMA 0 Count Register	000000h
DSTAT1	0x1460002c	R	DMA 1 Count Register	000000h
DSTAT2	0x1460004c	R	DMA 2 Count Register	000000h
DSTAT3	0x1460006c	R	DMA 3 Count Register	000000h

DSTATn	Bit	Description	Initial State
STAT	[21:20]	Status of this DMA controller. 00 = It indicates that DMA controller is ready for another DMA request. 01 = It 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 REGISTER (DCSRC)

Register	Address	R/W	Description	Reset Value
DCSRC0	0x14600010	R	DMA 0 Current Source Register	0x00000000
DCSRC1	0x14600030	R	DMA 1 Current Source Register	0x00000000
DCSRC2	0x14600050	R	DMA 2 Current Source Register	0x00000000
DCSRC3	0x14600070	R	DMA 3 Current Source Register	0x00000000

DCSRCn	Bit	Description	Initial State
CURR_SRC	[28:0]	Current source address for DMAn.	0x00000000

CURRENT DESTINATION REGISTER (DCDST)

Register	Address	R/W	Description	Reset Value
DCDST0	0x14600014	R	DMA 0 Current Destination Register	0x00000000
DCDST1	0x14600034	R	DMA 1 Current Destination Register	0x00000000
DCDST2	0x14600054	R	DMA 2 Current Destination Register	0x00000000
DCDST3	0x14600074	R	DMA 3 Current Destination Register	0x00000000

DCDSTn	Bit	Description	Initial State
CURR_DST	[28:0]	Current destination address for DMAn.	0x00000000

DMA MASK TRIGGER REGISTER (DMASKTRIG)

Register	Address	R/W	Description	Reset Value
DMASKTRIG0	0x14600018	R/W	DMA 0 Mask Trigger Register	000
DMASKTRIG1	0x14600038	R/W	DMA 1 Mask Trigger Register	000
DMASKTRIG2	0x14600058	R/W	DMA 2 Mask Trigger Register	000
DMASKTRIG3	0x14600078	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, CURR_SRC, CURR_DST will be 0. **NOTE:** Due to possible current atomic transfer, “stop” may take several cycles. The finish of “stopping” operation (i.e., actual stop time) can be detected by waiting until 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 finishes. **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. However, note that for this trigger to have effects 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.

NOTES

9 I/O PORTS

OVERVIEW

S3C2400X has 90 multi-functional input/output port pins. There are seven ports:

— One 18-bit output ports. (Port A)
— Two 16-bit input/output ports. (Port B and C)
— One 11-bit input/output port. (Port D)
— One 12-bit input/output port. (Port E)
— One 7-bit input/output port. (Port F)
— One 10-bit input/output port. (Port G)

Each port can be easily configured by software to meet various system configuration and design requirements. You have to define which function of each pin is used before starting the main program. If the multiplexed functions on a pin are not used, the pin can be configured as I/O ports.

The initial pin states, before pin configurations, are configured elegantly to avoid some problems.

Table 9-1. S3C2400X Port Configuration Overview

Port A	Selectable Pin Functions			
PA17	output only	nGCS[5]	–	–
PA16	output only	nGCS[4]	–	–
PA15	output only	nGCS[3]	–	–
PA14	output only	nGCS[2]	–	–
PA13	output only	nGCS[1]	–	–
PA12	output only	nCAS[1]	–	–
PA11	output only	nCAS[0]	–	–
PA10	output only	SCKE	–	–
PA9	output only	A24	–	–
PA8	output only	A23	–	–
PA7	output only	A22	–	–
PA6	output only	A21	–	–
PA5	output only	A20	–	–
PA4	output only	A19	–	–
PA3	output only	A18	–	–
PA2	output only	A17	–	–
PA1	output only	A16	–	–
PA0	output only	A0	–	–

Table 9-1. S3C2400X Port Configuration Overview (Continued)

Port B	Selectable Pin Functions			
PB15	Input/output	DATA31	–	–
PB14	Input/output	DATA30	–	–
PB13	Input/output	DATA29	–	–
PB12	Input/output	DATA28	–	–
PB11	Input/output	DATA27	–	–
PB10	Input/output	DATA26	nSS	–
PB9	Input/output	DATA25	I2SSDI	–
PB8	Input/output	DATA24	–	–
PB7	Input/output	DATA23	–	–
PB6	Input/output	DATA22	nRTS[1]	–
PB5	Input/output	DATA21	nCTS[1]	–
PB4	Input/output	DATA20	RXD[1]	–
PB3	Input/output	DATA19	TXD[1]	–
PB2	Input/output	DATA18	TCLK[1]	–
PB1	Input/output	DATA17	nXBREQ	–
PB0	Input/output	DATA16	nXBACK	–

Table 9-1. S3C2400X Port Configuration Overview (Continued)

Port C	Selectable Pin Functions			
PC15	Input/output	VD[15]	–	–
PC14	Input/output	VD[14]	–	–
PC13	Input/output	VD[13]	–	–
PC12	Input/output	VD[12]	–	–
PC11	Input/output	VD[11]	–	–
PC10	Input/output	VD[10]	–	–
PC9	Input/output	VD[9]	–	–
PC8	Input/output	VD[8]	–	–
PC7	Input/output	VD[7]	–	–
PC6	Input/output	VD[6]	–	–
PC5	Input/output	VD[5]	–	–
PC4	Input/output	VD[4]	–	–
PC3	Input/output	VD[3]	–	–
PC2	Input/output	VD[2]	–	–
PC1	Input/output	VD[1]	–	–
PC0	Input/output	VD[0]	–	–

Port D	Selectable Pin Functions			
PD10	Input/output	nWAIT	–	–
PD9	Input/output	TCLK[0]	–	–
PD8	Input/output	TOUT[3]	–	–
PD7	Input/output	TOUT[2]	–	–
PD6	Input/output	TOUT[1]	–	–
PD5	Input/output	TOUT[0]	–	–
PD4	Input/output	LEND	I2SSDI	–
PD3	Input/output	VCLK	–	–
PD2	Input/output	VLINE	–	–
PD1	Input/output	VM	–	–
PD0	Input/output	VFRAME	–	–

Table 9-1. S3C2400X Port Configuration Overview (Continued)

Port E	Selectable Pin Functions			
PE11	Input/output	nXDREQ[1]	nXBREQ	–
PE10	Input/output	nXDREQ[0]	–	–
PE9	Input/output	nXDACK[1]	nXBACK	–
PE8	Input/output	nXDACK[0]	–	–
PE7	Input/output	EINT[7]	–	–
PE6	Input/output	EINT[6]	–	–
PE5	Input/output	EINT[5]	TCLK[1]	–
PE4	Input/output	EINT[4]	nRTS[1]	–
PE3	Input/output	EINT[3]	nCTS[1]	–
PE2	Input/output	EINT[2]	I2SSDI	–
PE1	Input/output	EINT[1]	nSS	–
PE0	Input/output	EINT[0]	–	–

Port F	Selectable Pin Functions			
PF6	input/output	CLKOUT	–	–
PF5	input/output	nCTS[0]	nXBREQ	–
PF4	input/output	nRTS[0]	nXBACK	–
PF3	input/output	TXD[1]	IICSCL	–
PF2	input/output	TXD[0]	–	–
PF1	input/output	RXD[1]	IICSDA	–
PF0	input/output	RXD[0]	–	–

Port G	Selectable Pin Functions			
PG9	input/output	SPICLK	MMCCLK	–
PG8	input/output	SPIMOSI	IICSCL	–
PG7	input/output	SPIMISO	IICSDA	–
PG6	input/output	MMCDAT	IICSCL	–
PG5	input/output	MMCCMD	IICSDA	–
PG4	input/output	MMCCLK	I2SSDI	–
PG3	input/output	I2SSDO	I2SSDI	–
PG2	input/output	CDCLK	–	–
PG1	input/output	I2SSCLK	–	–
PG0	input/output	I2SLRCK	–	–

NOTE: The underlined function name is selected just after a reset.

PORT CONTROL DESCRIPTIONS

PORT CONFIGURATION REGISTER (PACON-PGCON)

In S3C2400X, most pins are multiplexed pins. So, It is determined which function is selected for each pins. The PnCON (port control register) determines which function is used for each pin.

If PE0 – PE7 is used for the wakeup signal in power down mode, these ports must be configured in interrupt mode.

PORT DATA REGISTER (PADAT-PGDAT)

If Ports are configured as output ports, data can be written to the corresponding bit of PnDAT. If Ports are configured as input ports, the data can be read from the corresponding bit of PnDAT.

PORT PULL-UP REGISTER (PBUP-PGUP)

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 then the pull-up resisters work without pin's functional setting(input, output, DATAn, EINTn and etc)

OPEN DRAIN CONTROL REGISTER

The port open drain control register controls the open drain function enable/disable of 6 pad of 2 port group. When the corresponding bit is 1, the open drain function of the pin is enabled. When 0, the open drain function is disabled.

MISCELLANEOUS CONTROL REGISTER

This register controls lower two byte DATA port pull-up resister, hi-z state or previous state in stop mode, USB pad, and CLKOUT selection.

EXTERNAL INTERRUPT CONTROL REGISTER

The 8 external interrupts are requested by various signaling methods. The EXTINT 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

Because each external interrupt pin has a digital filter, the interrupt controller can recognize the request signal that is longer than 3 clocks.

I/O PORT CONTROL REGISTER

PORT A CONTROL REGISTERS (PACON, PADAT)

Register	Address	R/W	Description	Reset Value
PACON	0x15600000	R/W	Configures the pins of port A	0x3ffff
PADAT	0x15600004	R/W	The data register for port A	Undef.

PACON	Bit	Description	
PA0	[0]	0 = Output	1 = ADDR0
PA1	[1]	0 = Output	1 = ADDR16
PA2	[2]	0 = Output	1 = ADDR17
PA3	[3]	0 = Output	1 = ADDR18
PA4	[4]	0 = Output	1 = ADDR19
PA5	[5]	0 = Output	1 = ADDR20
PA6	[6]	0 = Output	1 = ADDR21
PA7	[7]	0 = Output	1 = ADDR22
PA8	[8]	0 = Output	1 = ADDR23
PA9	[9]	0 = Output	1 = ADDR24
PA10	[10]	0 = Output	1 = SCKE
PA11	[11]	0 = Output	1 = nCAS[0]
PA12	[12]	0 = Output	1 = nCAS[1]
PA13	[13]	0 = Output	1 = nGCS[1]
PA14	[14]	0 = Output	1 = nGCS[2]
PA15	[15]	0 = Output	1 = nGCS[3]
PA16	[16]	0 = Output	1 = nGCS[4]
PA17	[17]	0 = Output	1 = nGCS[5]

PADAT	Bit	Description
PA[17:0]	[17:0]	When the port is configured as output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

PORT B CONTROL REGISTERS (PBCON, PBDAT, PBUP)

Register	Address	R/W	Description	Reset Value
PBCON	0x15600008	R/W	Configures the pins of port B	0xaaaaaaaa
PBDAT	0x1560000C	R/W	The data register for port B	Undef.
PBUP	0x15600010	R/W	pull-up disable register for port B	0x0

PBCON	Bit	Description
PB0	[1:0]	00 = Input 01 = Output 10 = DATA16 11 = nXBACK
PB1	[3:2]	00 = Input 01 = Output 10 = DATA17 11 = nXBREQ
PB2	[5:4]	00 = Input 01 = Output 10 = DATA18 11 = TCLK[1]
PB3	[7:6]	00 = Input 01 = Output 10 = DATA19 11 = TXD[1]
PB4	[9:8]	00 = Input 01 = Output 10 = DATA20 11 = RXD[1]
PB5	[11:10]	00 = Input 01 = Output 10 = DATA21 11 = nCTS[1]
PB6	[13:12]	00 = Input 01 = Output 10 = DATA22 11 = nRTS[1]
PB7	[15:14]	00 = Input 01 = Output 10 = DATA23 11 = Reserved
PB8	[17:16]	00 = Input 01 = Output 10 = DATA24 11 = Reserved
PB9	[19:18]	00 = Input 01 = Output 10 = DATA25 11 = I2SSDI
PB10	[21:20]	00 = Input 01 = Output 10 = DATA26 11 = nSS
PB11	[23:22]	00 = Input 01 = Output 10 = DATA27 11 = Reserved
PB12	[25:24]	00 = Input 01 = Output 10 = DATA28 11 = Reserved
PB13	[27:26]	00 = Input 01 = Output 10 = DATA29 11 = Reserved
PB14	[29:28]	00 = Input 01 = Output 10 = DATA30 11 = Reserved
PB15	[31:30]	00 = Input 01 = Output 10 = DATA31 11 = Reserved

PBDAT	Bit	Description
PB[15:0]	[15:0]	When the port is configured as input port, the corresponding bit is the pin state. When the port is configured as output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

PBUP	Bit	Description
PB[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 C CONTROL REGISTERS (PCCON, PCDAT, PCUP)

Register	Address	R/W	Description	Reset Value
PCCON	0x15600014	R/W	Configures the pins of port C	0x0
PCDAT	0x15600018	R/W	The data register for port C	Undef.
PCUP	0x1560001C	R/W	pull-up disable register for port C	0x0

PCCON	Bit	Description
PC0	[1:0]	00 = Input 01 = Output 10 = VD[0] 11 = Reserved
PC1	[3:2]	00 = Input 01 = Output 10 = VD[1] 11 = Reserved
PC2	[5:4]	00 = Input 01 = Output 10 = VD[2] 11 = Reserved
PC3	[7:6]	00 = Input 01 = Output 10 = VD[3] 11 = Reserved
PC4	[9:8]	00 = Input 01 = Output 10 = VD[4] 11 = Reserved
PC5	[11:10]	00 = Input 01 = Output 10 = VD[5] 11 = Reserved
PC6	[13:12]	00 = Input 01 = Output 10 = VD[6] 11 = Reserved
PC7	[15:14]	00 = Input 01 = Output 10 = VD[7] 11 = Reserved
PC8	[17:16]	00 = Input 01 = Output 10 = VD[8] 11 = Reserved
PC9	[19:18]	00 = Input 01 = Output 10 = VD[9] 11 = Reserved
PC10	[21:20]	00 = Input 01 = Output 10 = VD[10] 11 = Reserved
PC11	[23:22]	00 = Input 01 = Output 10 = VD[11] 11 = Reserved
PC12	[25:24]	00 = Input 01 = Output 10 = VD[12] 11 = Reserved
PC13	[27:26]	00 = Input 01 = Output 10 = VD[13] 11 = Reserved
PC14	[29:28]	00 = Input 01 = Output 10 = VD[14] 11 = Reserved
PC15	[31:30]	00 = Input 01 = Output 10 = VD[15] 11 = Reserved

PCDAT	Bit	Description
PC[15:0]	[15:0]	When the port is configured as input port, the corresponding bit is the pin state. When the port is configured as output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

PCUP	Bit	Description
PC[15:0]	[15:0]	0 = the pull up function attached to the corresponding port pin is enabled. 1 = the pull up function is disabled.

PORT D CONTROL REGISTERS (PDCON, PDDAT, PDUP)

Register	Address	R/W	Description	Reset Value
PDCON	0x15600020	R/W	Configures the pins of port D	0x0
PDDAT	0x15600024	R/W	The data register for port D	Undef.
PDUP	0x15600028	R/W	pull-up disable register for port D	0x620

PDCON	Bit	Description
PD0	[1:0]	00 = Input 01 = Output 10 = VFRAME 11 = Reserved
PD1	[3:2]	00 = Input 01 = Output 10 = VM 11 = Reserved
PD2	[5:4]	00 = Input 01 = Output 10 = VLINE 11 = Reserved
PD3	[7:6]	00 = Input 01 = Output 10 = VCLK 11 = Reserved
PD4	[9:8]	00 = Input 01 = Output 10 = LEND 11 = I2SSDI
PD5 (note)	[11:10]	00 = Input 01 = Output 10 = TOUT[0] 11 = Reserved
PD6	[13:12]	00 = Input 01 = Output 10 = TOUT[1] 11 = Reserved
PD7	[15:14]	00 = Input 01 = Output 10 = TOUT[2] 11 = Reserved
PD8	[17:16]	00 = Input 01 = Output 10 = TOUT[3] 11 = Reserved
PD9 (note)	[19:18]	00 = Input 01 = Output 10 = TCLK[0] 11 = Reserved
PD10 (note)	[21:20]	00 = Input 01 = Output 10 = nWAIT 11 = Reserved

NOTE: pd5, pd9, pd10 are 'pull-up disable' state at the initial condition.

PDDAT	Bit	Description
PD[10:0]	[10:0]	When the port is configured as an input port, the corresponding bit is the pin state. When the port is configured as an output port, the pin state is the same as the corresponding bit. When the port is configured as a functional pin, the undefined value will be read.

PDUP	Bit	Description
PD[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 E CONTROL REGISTERS (PECON, PEDAT)

If PE0 - PE7 will be used for wake-up signals at power down mode, the ports will be set in interrupt mode.

Register	Address	R/W	Description	Reset Value
PECON	0x1560002C	R/W	Configures the pins of port E	0x0
PEDAT	0x15600030	R/W	The data register for port E	Undef.
PEUP	0x15600034	R/W	pull-up disable register for port E	0x003

PECON	Bit	Description
PE0 (NOTE)	[1:0]	00 = Input 01 = Output 10 = EINT[0] 11 = Reserved
PE1 (NOTE)	[3:2]	00 = Input 01 = Output 10 = EINT[1] 11 = nSS
PE2	[5:4]	00 = Input 01 = Output 10 = EINT2] 11 = I2SSDI
PE3	[7:6]	00 = Input 01 = Output 10 = EINT[3] 11 = nCTS[1]
PE4	[9:8]	00 = Input 01 = Output 10 = EINT[4] 11 = nRTS[1]
PE5	[11:10]	00 = Input 01 = Output 10 = EINT[5] 11 = TCLK[1]
PE6	[13:12]	00 = Input 01 = Output 10 = EINT[6] 11 = Reserved
PE7	[15:14]	00 = Input 01 = Output 10 = EINT[7] 11 = Reserved
PE8	[17:16]	00 = Input 01 = Output 10 = nXDACK[0] 11 = Reserved
PE9	[19:18]	00 = Input 01 = Output 10 = nXDACK[1] 11 = nXBACK
PE10	[21:20]	00 = Input 01 = Output 10 = nXDREQ[0] 11 = Reserved
PE11	[23:22]	00 = Input 01 = Output 10 = nXDREQ[1] 11 = nXBREQ

NOTE: PE0, PE1 are 'pull-up disable' state at the initial condition.

PEDAT	Bit	Description
PE[11:0]	[11:0]	When the port is configured as an input port, the corresponding bit is the pin state. When the port is configured as an output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

PEUP	Bit	Description
PE[11:0]	[11:0]	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 (PFCON, PFDAT, PFUP)

Register	Address	R/W	Description	Reset Value
PFCON	0x15600038	R/W	Configures the pins of port F	0x0
PFDAT	0x1560003C	R/W	The data register for port F	Undef.
PFUP	0x15600040	R/W	Pull-up disable register for port F	0x0

PFCON	Bit	Description
PF0	[1:0]	00 = Input 01 = Output 10 = RXD[0] 11 = Reserved
PF1	[3:2]	00 = Input 01 = Output 10 = RXD[1] 11 = IICSDA
PF2	[5:4]	00 = Input 01 = Output 10 = TXD[0] 11 = Reserved
PF3	[7:6]	00 = Input 01 = Output 10 = TXD[1] 11 = IICSCL
PF4	[9:8]	00 = Input 01 = Output 10 = nRTS[0] 11 = nXBACK
PF5	[11:10]	00 = Input 01 = Output 10 = nCTS[0] 11 = nXBREQ
PF6	[13:12]	00 = Input 01 = Output 10 = CLKOUT 11 = Reserved

PFDAT	Bit	Description
PF[6:0]	[6:0]	When the port is configured as an input port, the corresponding bit is the pin state. When the port is configured as an output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

PFUP	Bit	Description
PF[6:0]	[6:0]	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 (PGCON, PGDAT, PGUP)

Register	Address	R/W	Description	Reset Value
PGCON	0x15600044	R/W	Configures the pins of port G	0x0
PGDAT	0x15600048	R/W	The data register for port G	Undef.
PGUP	0x1560004C	R/W	Pull-up disable register for port G	0x0

PGCON	Bit	Description
PG0	[1:0]	00 = Input 01 = Output 10 = I2SLRCK 11 = Reserved
PG1	[3:2]	00 = Input 01 = Output 10 = I2SSCLK 11 = Reserved
PG2	[5:4]	00 = Input 01 = Output 10 = CDCLK 11 = Reserved
PG3	[7:6]	00 = Input 01 = Output 10 = I2SSDO 11 = I2SSDI
PG4	[9:8]	00 = Input 01 = Output 10 = MMCCLK 11 = I2SSDI
PG5	[11:10]	00 = Input 01 = Output 10 = MMCCMD 11 = IICSDA
PG6	[13:12]	00 = Input 01 = Output 10 = MMCDAT 11 = IICSCL
PG7	[15:14]	00 = Input 01 = Output 10 = SPIMISO 11 = IICSDA
PG8	[17:16]	00 = Input 01 = Output 10 = SPIMOSI 11 = IICSCL
PG9	[19:18]	00 = Input 01 = Output 10 = SPICLK 11 = MMCCLK

PGDAT	Bit	Description
PG[9:0]	[9:0]	When the port is configured as an input port, the corresponding bit is the pin state. When the port is configured as output port, the pin state is the same as the corresponding bit. When the port is configured as a functional pin, the undefined value will be read.

PGUP	Bit	Description
PG[9:0]	[9:0]	0 = the pull up function attached to to the corresponding port pin is enabled. 1 = the pull up function is disabled.

OPEN DRAIN CONTROL REGISTER (OPENCR)

Port F[1], F[3] and Port G[8:5] pin's open drain mode can be controlled by OPENCR register.

Register	Address	R/W	Description	Reset Value
OPENCR	0x15600050	R/W	Open-drain enable register	0x0

OPENCR	Bit	Description
OPC_RXD1	[0]	0 = PF[1] port open-drain mode is disabled 1 = PF[1] port open-drain mode is enabled
OPC_TXD1	[1]	0 = PF[3] port open-drain mode is disabled 1 = PF[3] port open-drain mode is enabled
OPC_CMD	[2]	0 = PG[5] port open-drain mode is disabled 1 = PG[5] port open-drain mode is enabled
OPC_DAT	[3]	0 = PG[6] port open-drain mode is disabled 1 = PG[6] port open-drain mode is enabled
OPC_MISO	[4]	0 = PG[7] port open-drain mode is disabled 1 = PG[7] port open-drain mode is enabled
OPC_MOSI	[5]	0 = PG[8] port open-drain mode is disabled 1 = PG[8] port open-drain mode is enabled

MISCELLANEOUS Control Register (MISCCR)

D[15:0] pin pull-up resister can be controlled by MISCCR register.

In STOP mode, the data bus(D[31:0] or D[15:0] is Hi-Z state. But, because of the characteristics of IO pad, the data bus pull-up resisters have to be turned on to reduce the power consumption in STOP mode. D[31:16] pin pull-up resisters can be controlled by PBUP register. D[15:0] pin pull-up resisters can be controlled by MISCCR register.

In STOP mode, memory control signals (A[24:0], nGCS[5:0], nWE, nOE, nBE:nWBE:DQM) can be selectable Hi-z state or previous state in order to protect memory mal-function by setting the **Error! Bookmark not defined.** field in MISCCR register.

Pads related USB are controlled by this register for USB host, or for USB device.

CLKOUT signal can be selectable FCLK, HCLK, PCLK, MPLL CLK, or UPLL CLK by setting the CLKSEL field. Because of the pad delay, CLKOUT signal doesn't have same phase with it's original clock source(Internal FCLK, HCLK, PCLK, MPLL CLK, and UPLL CLK)

Register	Address	R/W	Description	Reset Value
MISCCR	0x15600054	R/W	Miscellaneous control register	0x00

MISCCR	Bit	Description
SPUCR0	[0]	0 = DATA[7:0] port pull-up resister is enabled 1 = DATA[7:0] port pull-up resister is disabled
SPUCR1	[1]	0 = DATA[15:8] port pull-up resister is enabled 1 = DATA[15:8] port pull-up resister is disabled
HZ@STOP	[2]	0 = HZ @ stop 1 = Previous state of PAD.
USBPAD	[3]	0 = use pads related USB for USB device 1 = use pads related USB for USB host
CLKSEL	[6:4]	000 = Select MPLL CLK with CLKOUT pad 001 = Select UPLL CLK with CLKOUT pad 010 = Select FCLK with CLKOUT pad 011 = Select HCLK with CLKOUT pad 1xx = Select PCLK with CLKOUT pad

NOTE: CLKOUT is prepared to monitor an internal clock situation (On/Off status or frequency).

EXTINT (EXTERNAL INTERRUPT CONTROL REGISTER)

The 8 external interrupts can be requested by various signaling methods. The EXTINT register 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 for 40ns at least because of the noise filter.

Register	Address	R/W	Description	Reset Value
EXTINT	0x15600058	R/W	External Interrupt control Register	0x000000

EXTINT	Bit	Description
EINT0	[2:0]	Setting the signaling method of the EINT0. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
EINT1	[6:4]	Setting the signaling method of the EINT1. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
EINT2	[10:8]	Setting the signaling method of the EINT2. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
EINT3	[14:12]	Setting the signaling method of the EINT3. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
EINT4	[18:16]	Setting the signaling method of the EINT4. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
EINT5	[22:20]	Setting the signaling method of the EINT5. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
EINT6	[26:24]	Setting the signaling method of the EINT6. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
EINT7	[30:28]	Setting the signaling method of the EINT7. 000 = Low level 001 = High level 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered

NOTES

10 PWM TIMER

OVERVIEW

The S3C2400X has five 16-bit timers. The timer 0, 1, 2, 3 have PWM function (Pulse Width Modulation). Timer 4 has an internal timer only with no output pins. Timer 0 has a dead-zone generator, which is used with a large current device.

Timer 0 and timer 1 share an 8-bit prescaler, timers 2 & 3 share the other 8-bit prescaler. Each timer has a clock-divider which has 5 different divided signals (1/2, 1/4, 1/8, 1/16, 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 MUX output signal (PHCLK) 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 PWM (pulse width modulation). 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.

FEATURES

— Five 16-bit timers

— Two 8-bit prescalers & Five 4-bit divider

— Programmable duty control of output waveform (PWM)

— Auto-reload mode or one-shot pulse mode

— Dead-zone generator

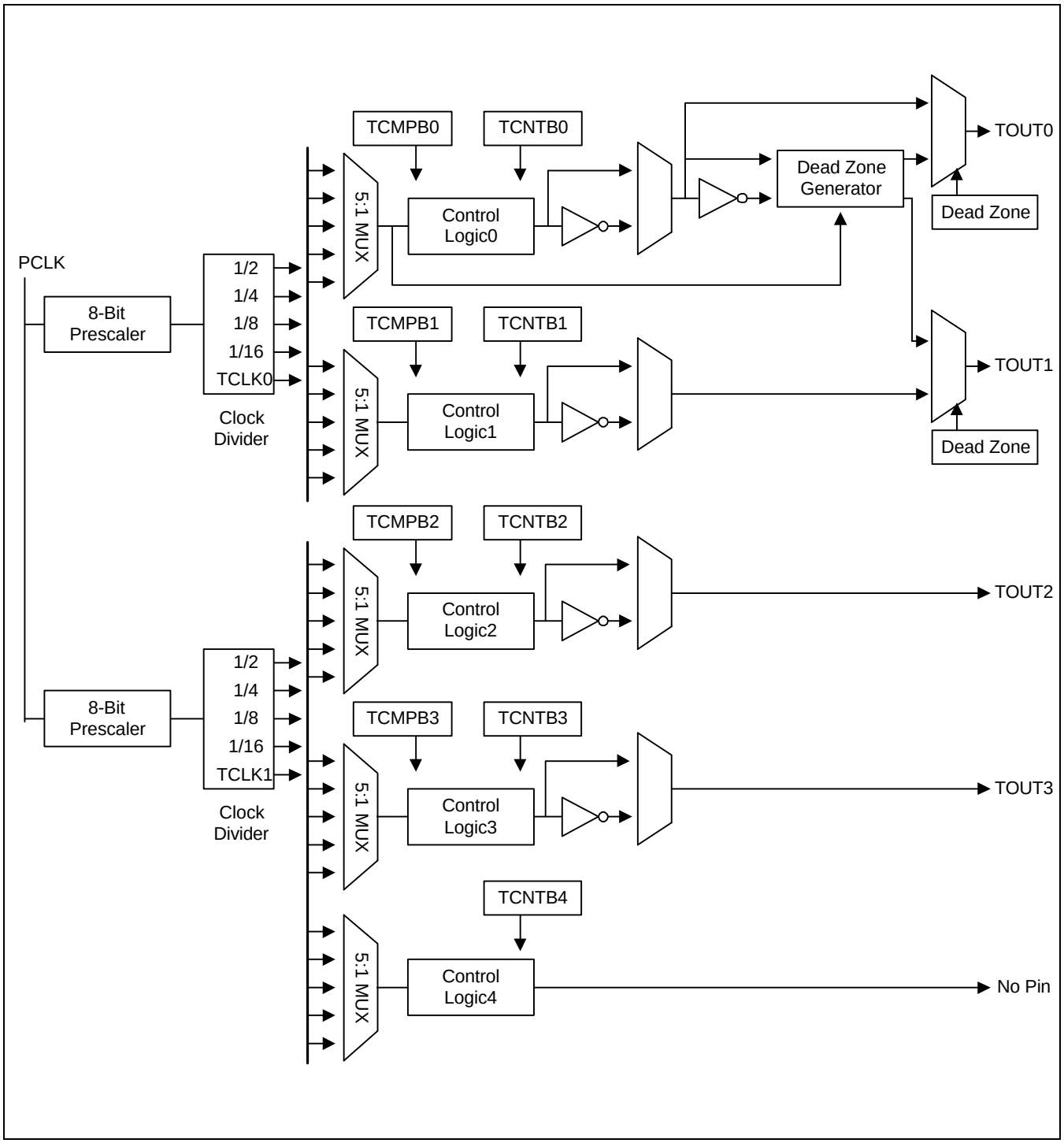


Figure 10-1. 16-bit PWM Timer Block Diagram

PWM TIMER OPERATION

PRESCALER & DIVIDER

An 8-bit prescaler and 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 MHz)	0.0303 us (33.0000 MHz)	7.7575 us (128.9063 KHz)	0.5084 sec
1/4 (PCLK = 66 MHz)	0.0606 us (16.5000 MHz)	15.5151 us (64.4531 KHz)	1.0168 sec
1/8 (PCLK = 66 MHz)	0.1212 us (8.2500 MHz)	31.0303 us (32.2266 KHz)	2.0336 sec
1/16 (PCLK = 66 MHz)	0.2424 us (4.1250 MHz)	62.0606 us (16.1133 KHz)	4.0671 sec

BASIC TIMER OPERATION

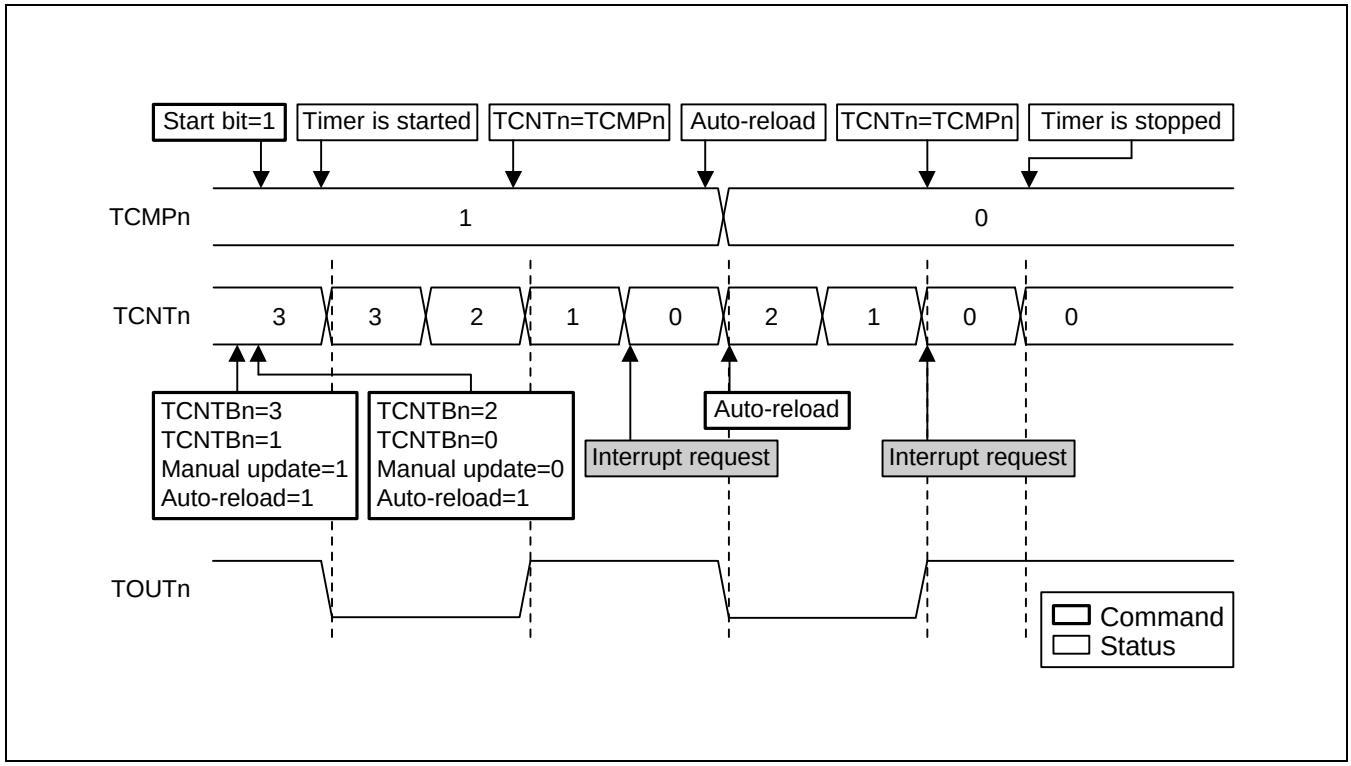


Figure 10-2. Timer Operations

A timer (except the timer ch-5) has TCNTBn, TCNTn, TCMPBn and TCMPn. TCNTBn and TCMPBn are loaded into TCNTn and TCMPn when the timer reaches 0. When TCNTn reaches 0, the interrupt request will occur if the interrupt is enabled. (TCNTn and TCMPn are the names of the internal registers. The TCNTn register can be read from the TCNTOn register)

AUTO-RELOAD & DOUBLE BUFFERING

S3C2400X PWM Timers have a double buffering feature, which can change the reload value 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 TCNTBn (Timer Count Buffer register) and the current counter value of the timer can be read from TCNTOn (Timer Count Observation register). If TCNTBn is read, the read value is not the current state of the counter but the reload value for the next timer duration.

The auto-reload is the operation, which copies the TCNTBn into TCNTn when TCNTn reaches 0. The value, written into TCNTBn, is loaded to TCNTn only when the TCNTn reaches to 0 and auto-reload is enabled. If the TCNTn is 0 and the auto-reload bit is 0, the TCNTn does not operate any further.

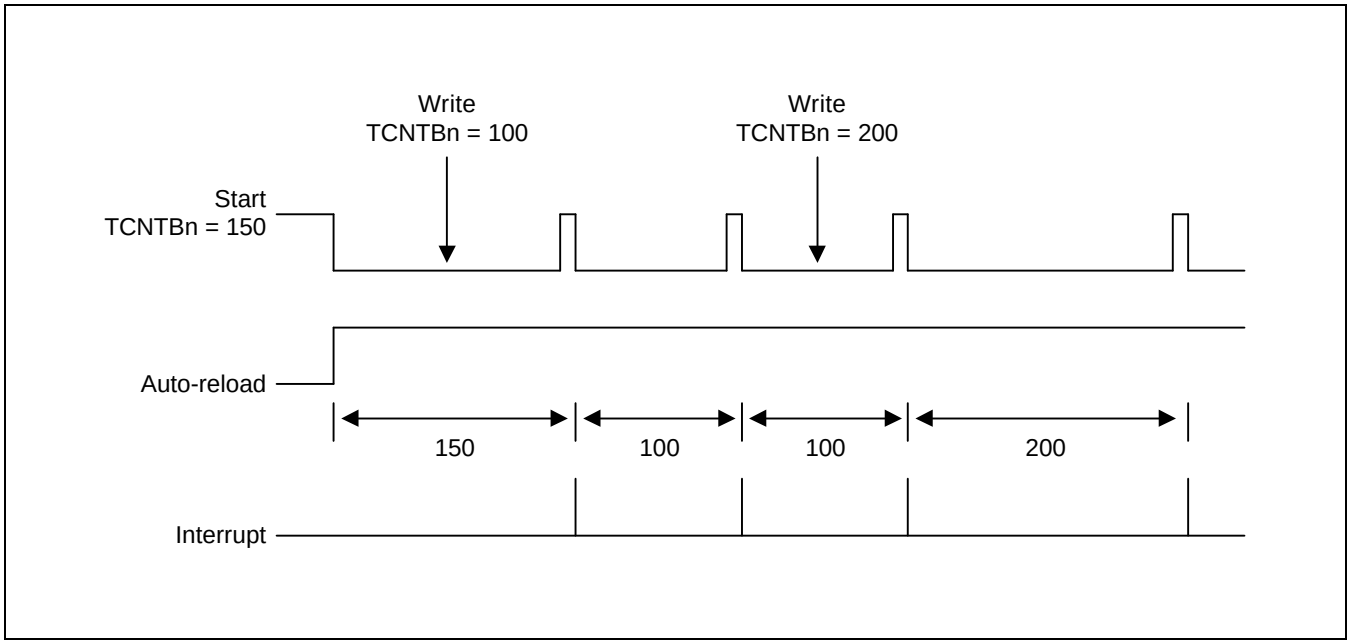


Figure 10-3. Example of Double Buffering Feature

TIMER INITIALIZATION USING MANUAL UPDATE BIT AND INVERTER BIT

Because an auto-reload operation of the timer occurs when the down counter reaches to 0, a starting value of the TCNTn has to be defined by the user at first. In this case, the starting value has to be loaded by the manual update bit. The sequence of starting a timer is as follows;

1) Write the initial value into TCNTBn and TCMPBn
2) Set the manual update bit of the corresponding timer. It is recommended to configure the inverter on/off bit. (whether use inverter or not)
3) Set start bit of corresponding timer to start the timer(At the same time, clear the manual update bit).

Also, if the timer is stopped by force, the TCNTn retains the counter value and is not reloaded from TCNTBn. If new value has to be set, manual update has to be done.

NOTE

Whenever TOUT inverter on/off bit is changed, the TOUTn logic value will be changed whether the timer runs or not. Therefore, it is desirable that the inverter on/off bit is configured with the manual update bit.

EXAMPLE OF A TIMER OPERATION

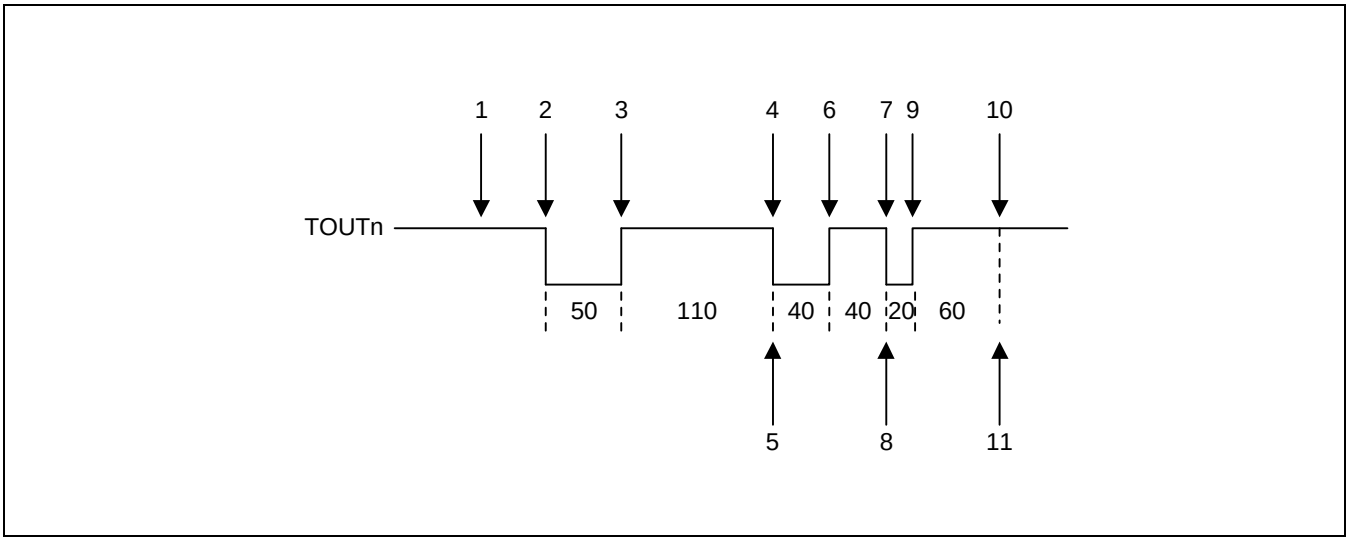


Figure 10-4. Example of a Timer Operation

The result of the following procedure is shown in Figure10-4;

1. Enable the auto-reload feature. Set the TCNTBn as 160 (50+110) and the TCMPBn as 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 TCNTBn and TCMPBn as 80 (40+40) and 40, respectively, to determine the next reload value.
2. Set the start bit, provided that manual_update is 0 and inverter is off and auto-reload is on. The timer starts counting down after latency time within the timer resolution.
3. When TCNTn has the same value with TCMPn, the logic level of TOUTn is changed from low to high.
4. When TCNTn reaches 0, the interrupt request is generated and TCNTBn value is loaded into a temporary register. At the next timer tick, TCNTn is reloaded with the temporary register value(TCNTBn).
5. In the ISR(Interrupt Service Routine), the TCNTBn and TCMPBn are set as 80 (20+60) and 60, respectively, which is used for the next duration.
6. When TCNTn has the same value as TCMPn, the logic level of TOUTn is changed from low to high.
7. When TCNTn reaches 0, TCNTn is reloaded automatically with TCNTBn. At the same time, the interrupt request is generated.
8. In the ISR (Interrupt Service Routine), auto-reload and interrupt request are disabled to stop the timer.
9. When the value of TCNTn is same as TCMPn, the logic level of TOUTn is changed from low to high.
10. Even when TCNTn reaches to 0, TCNTn is not any more reloaded and the timer is stopped because auto-reload has been disabled.
11. No interrupt request is generated.

PWM (PULSE WIDTH MODULATION)

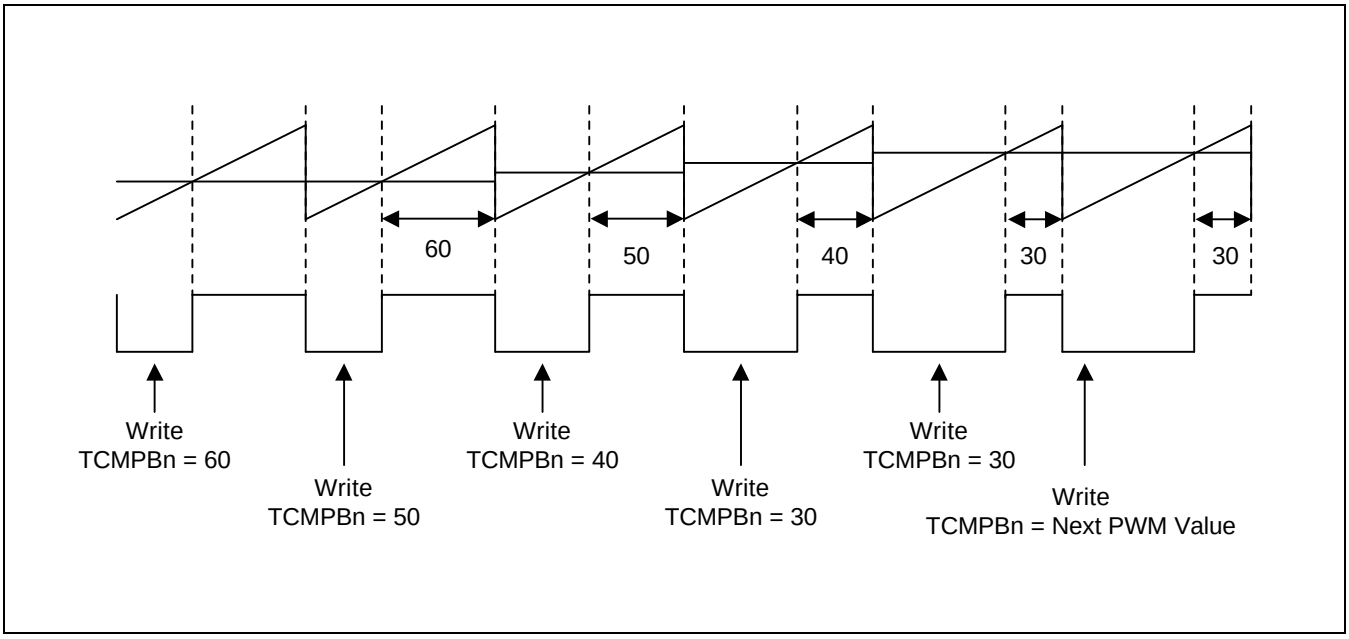


Figure 10-5. Example of PWM

PWM feature can be implemented by using the TCMPBn. PWM frequency is determined by TCNTBn. A PWM value is determined by TCMPBn in figure 10-5.

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.

Because of the double buffering feature, TCMPBn, for a next PWM cycle, can be written at any point in the current PWM cycle by ISR or something else

OUTPUT LEVEL CONTROL

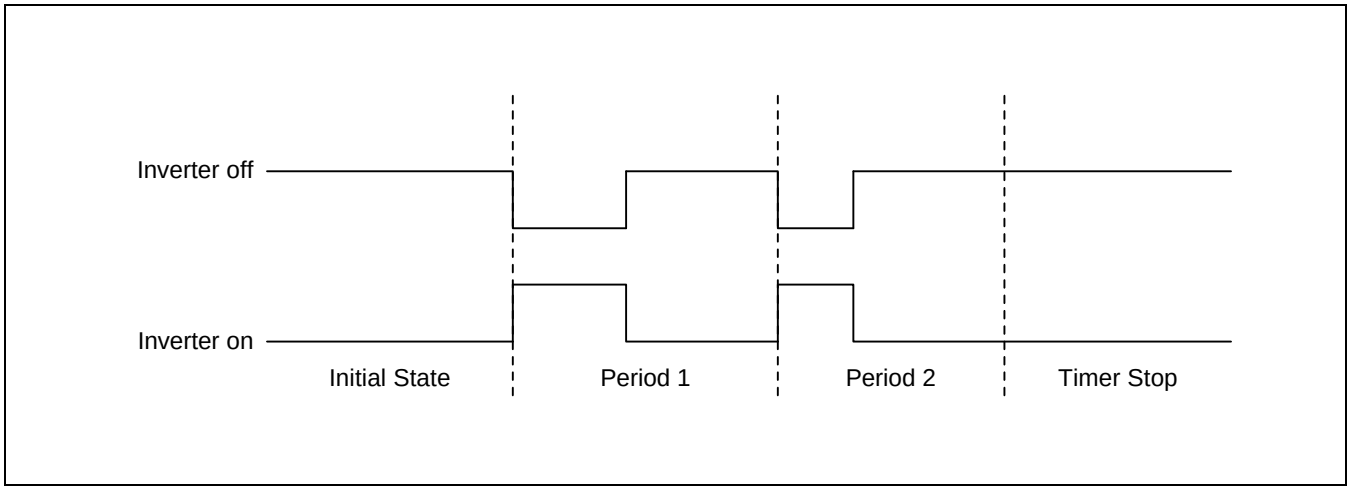


Figure 10-6. Inverter On/Off

The following methods can be used 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 TCNTn reaches to 0. This method is 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. 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 feature is used to insert 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 turning 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.

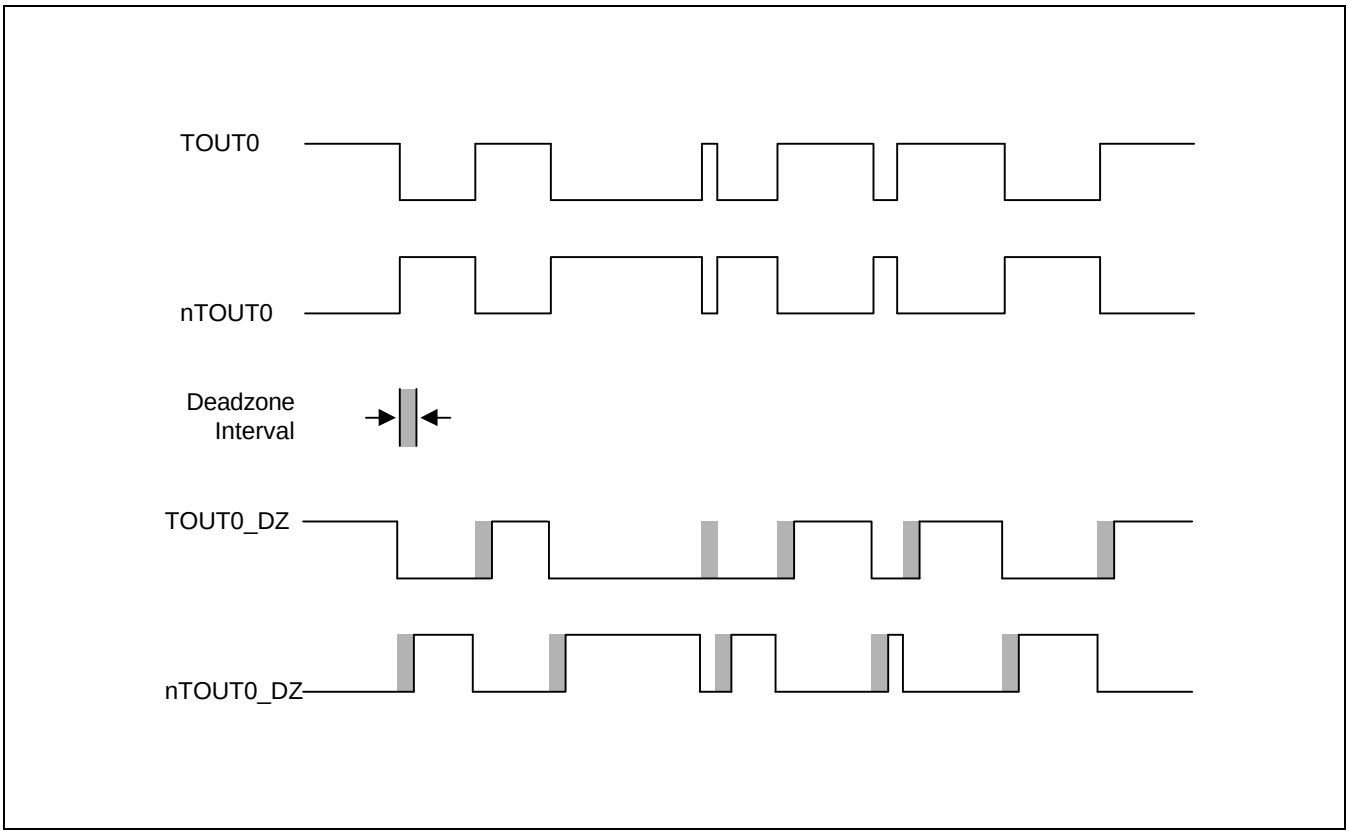


Figure 10-7. The Wave Form When a Dead Zone Feature is Enabled

DMA REQUEST MODE

The PWM timer can generate a DMA request at every specific times. The timer keeps DMA request signal (nDMA_REQ) low until the timer receives the 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

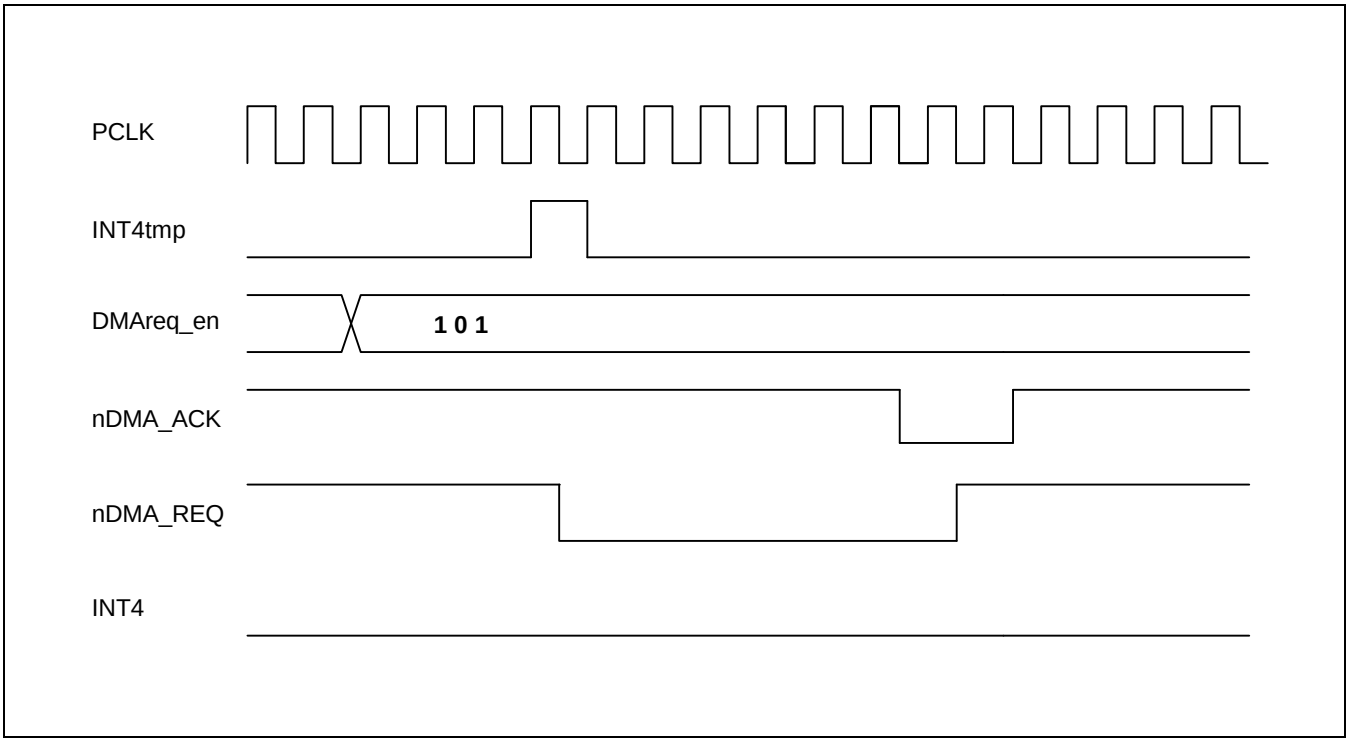


Figure 10-8. The Timer 4 DMA Mode Operation

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,TCLKn

Register	Address	R/W	Description	Reset Value
TCFG0	0x15100000	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 the 1 unit time 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	0x15100004	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 REGISTER (TCON)

Register	Address	R/W	Description	Reset Value
TCON	0x15100008	R/W	Timer control register	0x00000000

TCON	Bit	Description	initial state
Timer 4 auto reload on/off	[22]	This bit determines auto reload on/off for Timer 4. 0 = One-shot 1 = Interval mode (auto reload)	0
Timer 4 manual update (note)	[21]	This bit determines the manual update for Timer 4. 0 = No operation 1 = Update TCNTB4	0
Timer 4 start/stop	[20]	This bit determines start/stop for Timer 4. 0 = Stop 1 = Start for Timer 4	0
Timer 3 auto reload on/off	[19]	This bit determines auto reload on/off for Timer 3. 0 = One-shot 1 = Interval mode (auto reload)	0
Timer 3 output inverter on/off	[18]	This bit determines output inverter on/off for Timer 3. 0 = Inverter off 1 = Inverter on for TOUT3	0
Timer 3 manual update (note)	[17]	This bit determine manual update for Timer 3. 0 = No operation 1 = Update TCNTB3, TCMPB3	0
Timer 3 start/stop	[16]	This bit determines start/stop for Timer 3. 0 = Stop 1 = Start for Timer 3	0
Timer 2 auto reload on/off	[15]	This bit determines auto reload on/off for Timer 2. 0 = One-shot 1 = Interval mode (auto reload)	0
Timer 2 output inverter on/off	[14]	This bit determines output inverter on/off for Timer 2. 0 = Inverter off 1 = Inverter on for TOUT2	0
Timer 2 manual update (note)	[13]	This bit determines the manual update for Timer 2. 0 = No operation 1 = Update TCNTB2, TCMPB2	0
Timer 2 start/stop	[12]	This bit determines start/stop for Timer 2. 0 = Stop 1 = Start for Timer 2	0
Timer 1 auto reload on/off	[11]	This bit determines the auto reload on/off for Timer1. 0 = One-shot 1 = Interval mode (auto reload)	0
Timer 1 output inverter on/off	[10]	This bit determines the output inverter on/off for Timer1. 0 = Inverter off 1 = Inverter on for TOUT1	0
Timer 1 manual update (note)	[9]	This bit determines the manual update for Timer 1. 0 = No operation 1 = Update TCNTB1, TCMPB1	0
Timer 1 start/stop	[8]	This bit determines start/stop for Timer 1. 0 = Stop 1 = Start for Timer 1	0

NOTE: This bit has to be cleared at next writing.

TCON	Bit	Description	initial state
Dead zone enable	[4]	This bit determines the dead zone operation. 0 = Disable 1 = Enable	0
Timer 0 auto reload on/off	[3]	This bit determines auto reload on/off for Timer 0. 0 = One-shot 1 = Interval mode(auto reload)	0
Timer 0 output inverter on/off	[2]	This bit determines the output inverter on/off for Timer 0. 0 = Inverter off 1 = Inverter on for TOUT0	0
Timer 0 manual update (note)	[1]	This bit determines the manual update for Timer 0. 0 = No operation 1 = Update TCNTB0, TCMPB0	0
Timer 0 start/stop	[0]	This bit determines start/stop for Timer 0. 0 = Stop 1 = Start for Timer 0	0

NOTE: This bit has to be cleared at next writing.

TIMER 0 COUNT BUFFER REGISTER & COMPARE BUFFER REGISTER (TCNTB0, TCMPB0)

Register	Address	R/W	Description	Reset Value
TCNTB0	0x1510000C	R/W	Timer 0 count buffer register	0x00000000
TCMPB0	0x15100010	R/W	Timer 0 compare buffer register	0x00000000

TCMPB0	Bit	Description	Initial State
Timer 0 compare buffer register	[15:0]	Setting compare buffer value for Timer 0	0x00000000

TCNTB0	Bit	Description	Initial State
Timer 0 count buffer register	[15:0]	Setting count buffer value for Timer 0	0x00000000

TIMER 0 COUNT OBSERVATION REGISTER (TCNTO0)

Register	Address	R/W	Description	Reset Value
TCNTO0	0x15100014	R	Timer 0 count observation register	0x00000000

TCNTO0	Bit	Description	Initial State
Timer 0 observation register	[15:0]	Setting 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	0x15100018	R/W	Timer 1 count buffer register	0x00000000
TCMPB1	0x1510001C	R/W	Timer 1 campare buffer register	0x00000000

TCMPB1	Bit	Description	Initial State
Timer 1 compare buffer register	[15:0]	Setting compare buffer value for Timer 1	0x00000000

TCNTB1	Bit	Description	Initial State
Timer 1 count buffer register	[15:0]	Setting count buffer value for Timer 1	0x00000000

TIMER 1 COUNT OBSERVATION REGISTER (TCNTO1)

Register	Address	R/W	Description	Reset Value
TCNTO1	0x15100020	R	Timer 1 count observation register	0x00000000

TCNTO1	Bit	Description	initial state
Timer 1 observation register	[15:0]	Setting 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	0x15100024	R/W	Timer 2 count buffer register	0x00000000
TCMPB2	0x15100028	R/W	Timer 2 campare buffer register	0x00000000

TCMPB2	Bit	Description	Initial State
Timer 2 compare buffer register	[15:0]	Setting compare buffer value for Timer 2	0x00000000

TCNTB2	Bit	Description	Initial State
Timer 2 count buffer register	[15:0]	Setting count buffer value for Timer 2	0x00000000

TIMER 2 COUNT OBSERVATION REGISTER (TCNTO2)

Register	Address	R/W	Description	Reset Value
TCNTO2	0x1510002C	R	Timer 2 count observation register	0x00000000

TCNTO2	Bit	Description	Initial State
Timer 2 observation register	[15:0]	Setting 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	0x15100030	R/W	Timer 3 count buffer register	0x00000000
TCMPB3	0x15100034	R/W	Timer 3 campare buffer register	0x00000000

TCMPB3	Bit	Description	Initial State
Timer 3 compare buffer register	[15:0]	Setting compare buffer value for Timer 3	0x00000000

TCNTB3	Bit	Description	Initial State
Timer 3 count buffer register	[15:0]	Setting count buffer value for Timer 3	0x00000000

TIMER 3 COUNT OBSERVATION REGISTER (TCNTO3)

Register	Address	R/W	Description	Reset Value
TCNTO3	0x15100038	R	Timer 3 count observation register	0x00000000

TCNTO3	Bit	Description	Initial State
Timer 3 observation register	[15:0]	Setting count observation value for Timer 3	0x00000000

TIMER 4 COUNT BUFFER REGISTER (TCNTB4)

Register	Address	R/W	Description	Reset Value
TCNTB4	0x1510003C	R/W	Timer 4 count buffer register	0x00000000

TCNTB4	Bit	Description	Initial State
Timer 4 count buffer register	[15:0]	Setting count buffer value for Timer 4	0x00000000

TIMER 4 COUNT OBSERVATION REGISTER (TCNTO4)

Register	Address	R/W	Description	Reset Value
TCNTO4	0x15100040	R	Timer 4 count observation register	0x00000000

TCNTO4	Bit	Description	Initial State
Timer 4 observation register	[15:0]	Setting count observation value for Timer 4	0x00000000

NOTES

11 UART

OVERVIEW

The S3C2400X UART (Universal Asynchronous Receiver and Transmitter) unit provides two independent asynchronous serial I/O (SIO) ports, each of which can operate in interrupt-based or DMA-based mode. In other words, UART can generate an interrupt or DMA request to transfer data between CPU and UART. It can support bit rates of up to 115.2K bps. Each UART channel contains two 16-byte FIFOs for receive and transmit.

The S3C2400X 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, transmitter, receiver and control unit, as shown in Figure11-1. The baud-rate generator can be clocked by PCLK. The transmitter and the receiver contain 16-byte FIFOs and data shifters. Data, which is to be transmitted, is written to FIFO and then copied to the transmit shifter. It is then shifted out by the transmit data pin (TxDn). The received data is shifted from the receive data pin (RxDn), and then copied to FIFO from the shifter.

FEATURES

— RxD0,TxD0,RxD1,TxD1 with DMA-based or interrupt-based operation
— UART Ch 0 with IrDA 1.0 & 16-byte FIFO
— UART Ch 1 with IrDA 1.0 & 16-byte FIFO
— Supports handshake transmit / receive

BLOCK DIAGRAM

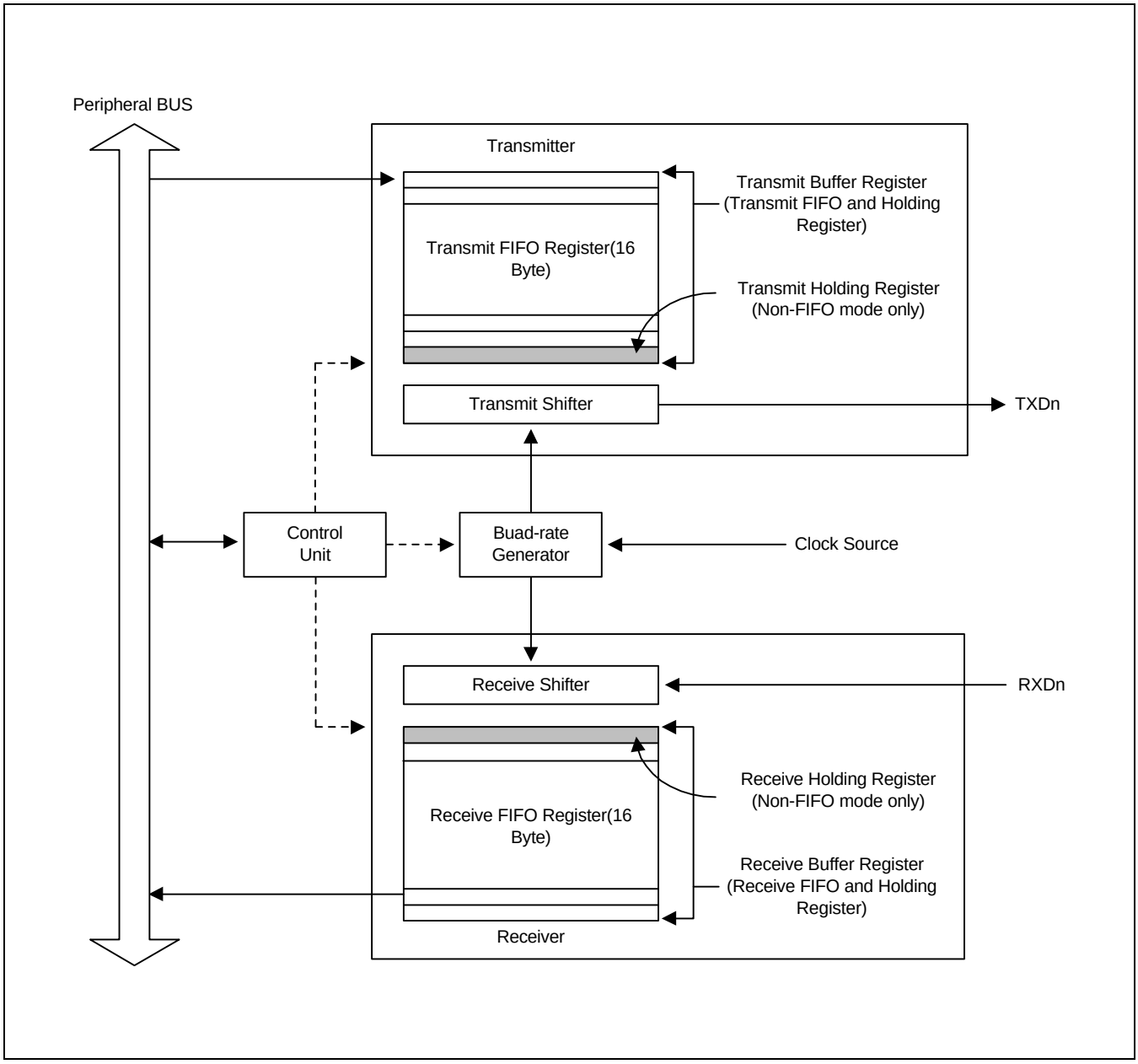


Figure 11-1. UART Block Diagram (with FIFO)

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. The break condition forces the serial output to logic 0 state for one frame transmission time. This block transmits break signal after the present transmission word transmits perfectly. After the break signal transmission, it continously 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, parity error, frame error and break condition, each of which can set an error flag.

— The overrun error indicates that new data has overwritten the old data before the old data has been read.

— The parity error indicates that the receiver has detected an unexpected parity condition.

— The frame error indicates that the received data does not have a valid stop bit.

— The break condition indicates that the RxDn input is held in the logic 0 state for a duration longer than one frame transmission time.

Receive time-out condition occurs when it does not receive 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)

S3C2400X's UART supports auto flow control with nRTS and nCTS signals, in case it would have to connect UART to UART. If users connect UART to a Modem, disable auto flow control bit in UMCONn register and control the signal of nRTS by software.

In AFC, nRTS is controlled by condition of the receiver and operation of transmitter is controlled by the nCTS signal. The UART's transmitter transfers the data in FIFO only when nCTS signal active(In AFC, nCTS means that the 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).

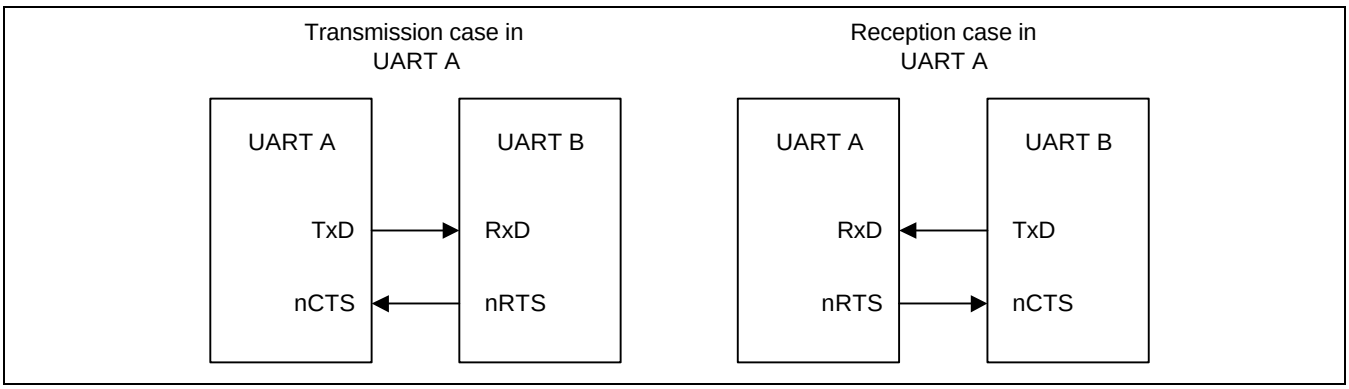


Figure 11-2. UART AFC interface

Non Auto-Flow control (Controlling nRTS and nCTS by S/W) Example

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'(activate nRTS), and if it is equal or larger than 15 users have to set the value to '0'(inactivate nRTS).
3. Repeat item 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'(nCTS is activated), users write the data to Tx FIFO register.

RS-232C interface

If users connect to modem interface (not equal null modem), nRTS, nCTS, nDSR, nDTR, DCD and nRI signals are need. In this case, users control these signals with general I/O ports by S/W because the AFC does not support the RS-232C interface.

Interrupt/DMA Request Generation

Each UART of S3C2400X has seven status(Tx/Rx/Error) signals: Overrun error, Parity error, Frame error, Break, 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, parity error, frame error and break condition 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 UERSTSTn.

When the receiver transfers the data of the receive shifter to the receive FIFO and the number of received data reaches Rx FIFO Trigger Level, Rx interrupt is generated, if Receive mode in control register is selected as Interrupt request or polling mode.
In the Non-FIFO mode, transfering the data of the receive shifter to the receive FIFO will cause Rx interrupt under the Interrupt request and polling mode.

When the transmitter transfers data from its transmit FIFO 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, transfering data from the transmit FIFO 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 is occurred 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	Each time receive data reaches the trigger level of receive FIFO, the Rx interrupt will be generated. When the number of data in FIFO does not reaches Rx FIFO trigger Level and does not receive data during 3 word time(This interval follows the setting of Word Length bit), the Rx interrupt will be generated(receive time out).	Each time receive data becomes full, the receive holding register, generates an interrupt.
Tx interrupt	Each time transmit data reaches the trigger level of transmit FIFO(Tx FIFO trigger Level), the Tx interrupt will be generated.	Each time transmit data become empty, the transmit holding register generates an interrupt.
Error interrupt	Frame error, parity error, and break signal are detected, and will generate an error interrupt. When it gets to the top of the receive FIFO without reading out data in it, the error interrupt will be generated(overrun error).	All errors generate an error interrupt immediately. However if another error occurs at the same time, only one interrupt is generated.

UART Error Status FIFO

UART has the status FIFO besides the Rx FIFO register. The 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 status FIFO, the URXHn with an error and UERSTATn must be read out.

For example,

It is assumed that the UART FIFO receives A, B, C, D, E characters sequentially and the frame error occurrs while receiving 'B', and the parity error occurs while receiving 'D'.

Although the actual UART error occurred, the error interrupt will not be generated 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.

Time	Sequence flow	Error interrupt	Note
#0	When no character is read out	–	
#1	After A is read out	The frame error(in B) interrupt occurs	The 'B' has to be read out
#2	After B is read out	–	
#3	After C is read out	The parity error(in D) interrupt occurs	The 'D' has to be read out
#4	After D is read out	–	
#5	After E is read out	–	

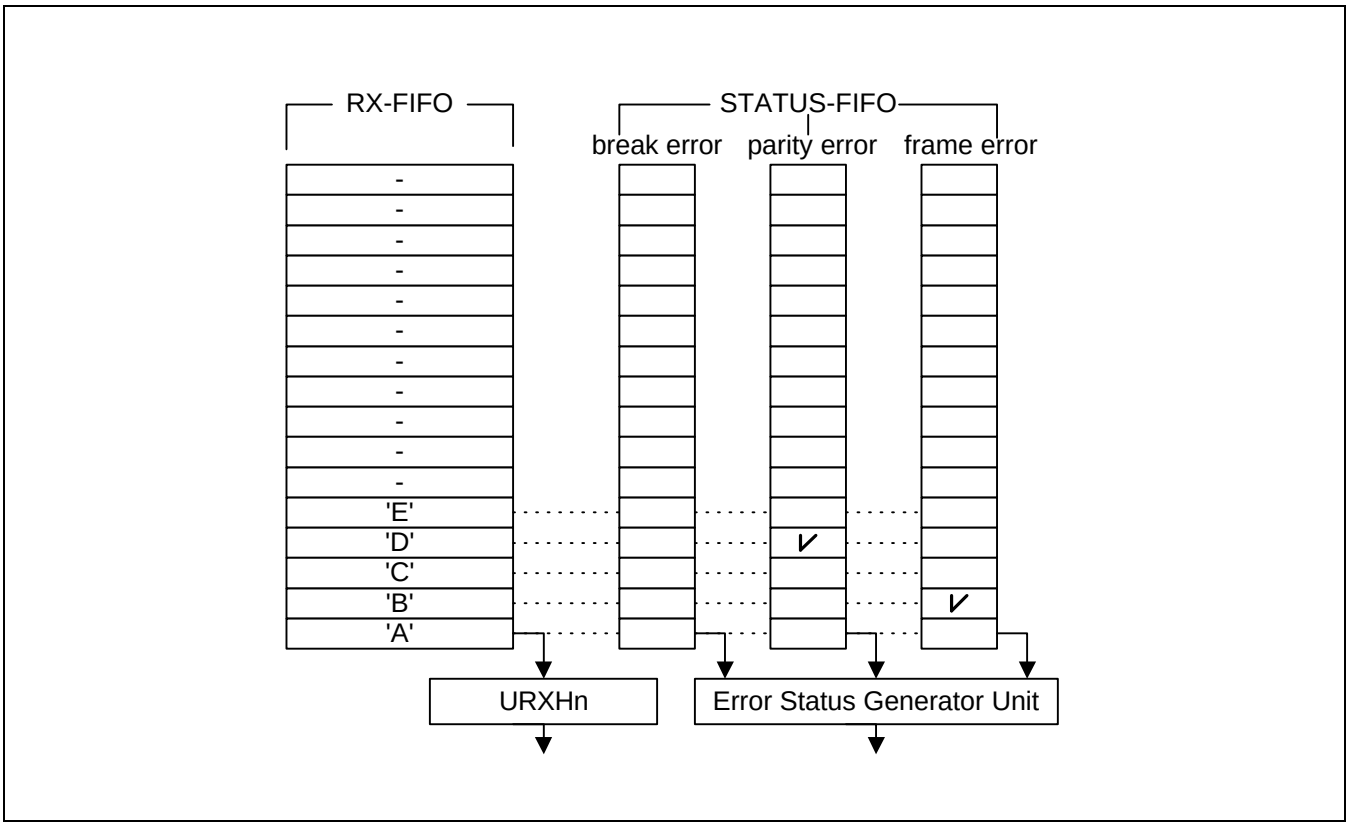


Figure 11-3. A Case showing UART Receiving 5 Characters with 2 Errors

Baud-Rate Generation

Each UART's baud-rate generator provides the serial clock for transmitter and receiver. The source clock for the baud-rate generator can be selected with the S3C2400X's internal system clock. The baud-rate clock is generated by dividing the source clock by 16 and a 16-bit divisor specified in the UART baud-rate divisor register (UBRDIVn). The UBRDIVn can be determined as follows:

UBRDIVn = (int)(PCLK/(bps x 16)) -1

where the divisor should be from 1 to (2^{16}-1). For example, if the baud-rate is 115200 bps and PCLK is 40 MHz, UBRDIVn is:

UBRDIVn = (int)(40000000/(115200 x 16)) -1
= (int)(21.7) -1
= 21 -1 = 20

Loop-back Mode

The S3C2400X UART provides a test mode referred to as the loopback mode, to aid in isolating faults in the communication link. In this mode, the transmitted data is immediately received. 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).

Break Condition

The break is defined as a continuous low level signal for one frame transmission time on the transmit data output.

IR (Infra-Red) Mode

The S3C2400X 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). The implementation of the mode is shown in Figure 11-3.

In IR transmit mode, the transmit period is pulsed 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 (refer to the frame timing diagrams shown in Figure 11-5 and 11-6).

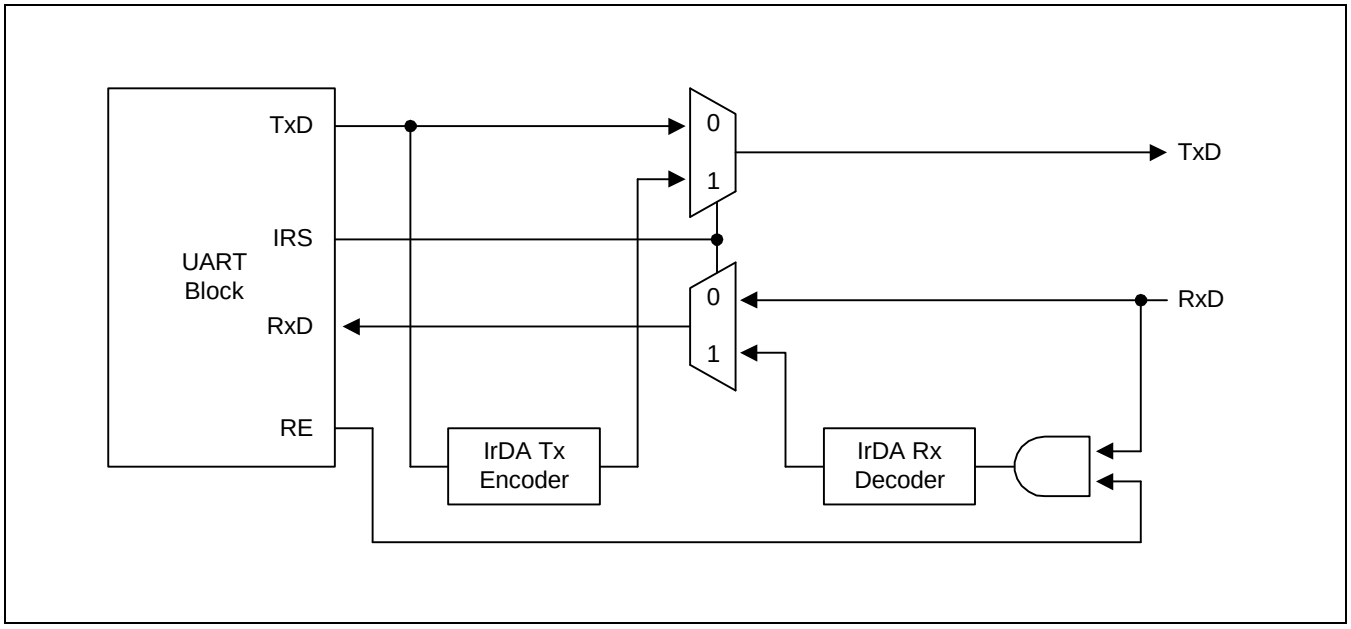


Figure 11-4. IrDA Function Block Diagram

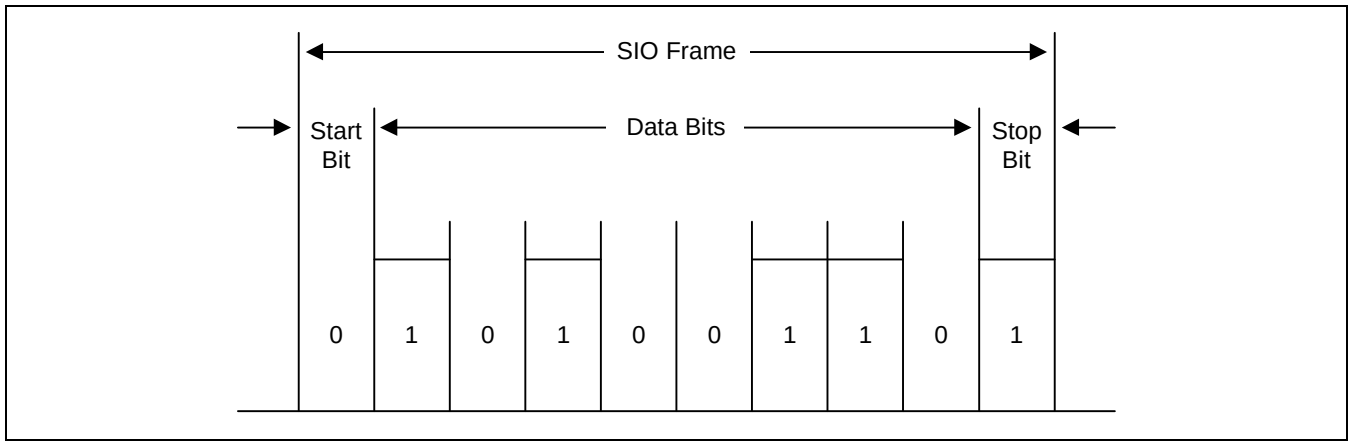


Figure 11-5. Serial I/O Frame Timing Diagram (Normal UART)

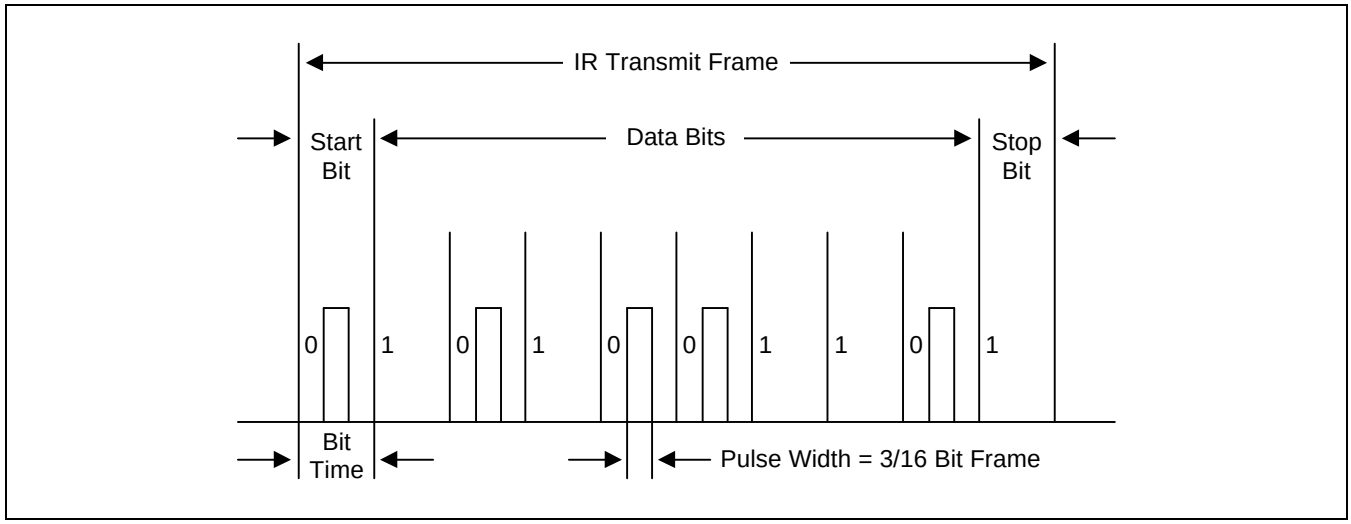


Figure 11-6. Infra-Red Transmit Mode Frame Timing Diagram

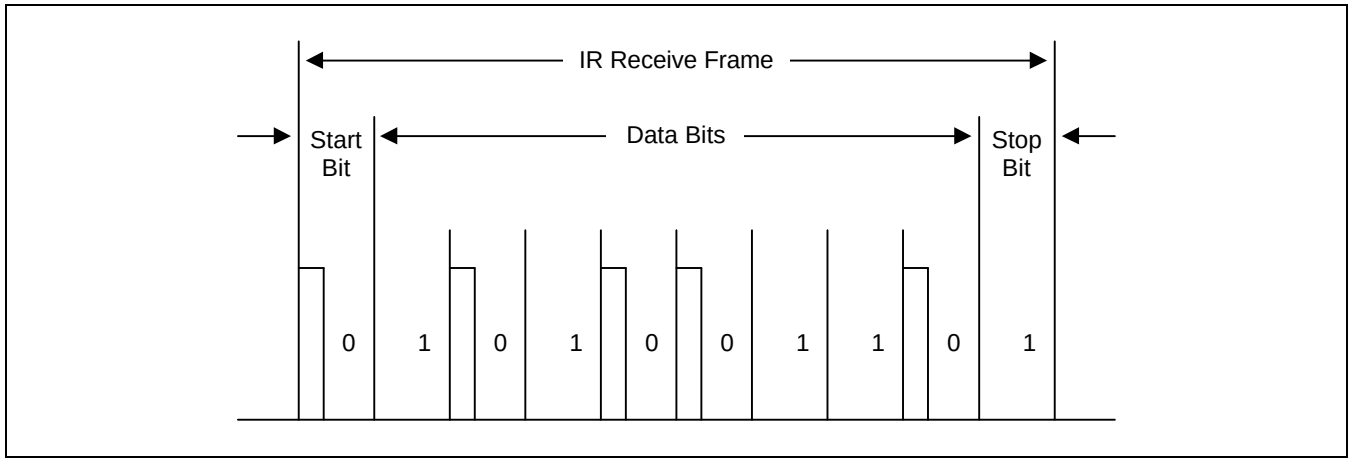


Figure 11-7. Infra-Red Receive Mode Frame Timing Diagram

UART SPECIAL REGISTERS

UART LINE CONTROL REGISTER

There are two UART line control registers, ULCON0 and ULCON1, in the UART block.

Register	Address	R/W	Description	Reset Value
ULCON0	0x15000000	R/W	UART channel 0 line control register	0x00
ULCON1	0x15004000	R/W	UART channel 1 line control register	0x00

ULCONn	Bit	Description	Initial State
Reserved	[7]		0
Infra-Red Mode	[6]	The Infra-Red mode determines whether or not to use the Infra-Red mode. 0 = Normal mode operation 1 = Infra-Red Tx/Rx mode	0
Parity Mode	[5:3]	The parity mode specifies how parity generation and checking are to be performed 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]	The number of stop bits specifies how many stop bits are to be used to signal end-of-frame. 0 = One stop bit per frame 1 = Two stop bit per frame	0
Word length	[1:0]	The word length indicates the number of data bits to be transmitted or received per frame. 00 = 5-bits 01 = 6-bits 10 = 7-bits 11 = 8-bits	00

UART CONTROL REGISTER

There are two UART control registers, UCON0 and UCON1, in the UART block.

Register	Address	R/W	Description	Reset Value
UCON0	0x15000004	R/W	UART channel 0 control register	0x00
UCON1	0x15004004	R/W	UART channel 1 control register	0x00

UCONn	Bit	Description	Initial State
Tx interrupt type	[9]	Interrupt request type 0 = Pulse (Interrupt is requested as soon as Tx buffer becomes empty in Non-FIFO mode or Tx FIFO reaches Trigger Level in FIFO mode) 1 = Level (Interrupt is requested while Tx buffer is empty in Non-FIFO mode or Tx FIFO has been being in 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]	This bit enables the UART to generate an interrupt if an exception, such as a break, frame error, parity error, or overrun error occurs during a receive operation. 0 = Do not generate receive error status interrupt 1 = Generate receive error status interrupt	0
Loop-back Mode	[5]	Setting loop-back bit to 1 causes the UART to enter the loop-back mode. This mode is provided for test purposes only. 0 = Normal operation 1 = Loop-back mode	0
Send Break Signal	[4]	Setting this bit causes the UART to send a break during 1 frame time. This bit is auto-cleared after sending the break signal. 0 = Normal transmit 1 = Send break signal	0
Transmit Mode	[3:2]	These two bits determine which function is currently able to write Tx data to the UART transmit buffer register. 00 = Disable 01 = Interrupt request or polling mode 10 = DMA0 request (Only for UART0) 11 = DMA1 request (Only for UART1)	00
Receive Mode	[1:0]	These two bits determine which function is currently able to read data from UART receive buffer register. 00 = Disable, 01 = Interrupt request or polling mode 10 = DMA0 request (Only for UART0) 11 = DMA1 request (Only for UART1)	00

UART FIFO CONTROL REGISTER

There are two UART FIFO control registers, UFCON0 and UFCON1, in the UART block.

Register	Address	R/W	Description	Reset Value
UFCON0	0x15000008	R/W	UART channel 0 FIFO control register	0x0
UFCON1	0x15004008	R/W	UART channel 1 FIFO control register	0x0

UFCONn	Bit	Description	Initial State
Tx FIFO Trigger Level	[7:6]	These two bits 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]	These two bits 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]	This bit is auto-cleared after resetting FIFO 0 = Normal 1= Tx FIFO reset	0
Rx FIFO Reset	[1]	This bit is auto-cleared after resetting FIFO 0 = Normal 1= Rx FIFO reset	0
FIFO Enable	[0]	0 = FIFO disable 1 = FIFO mode	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, UMCON0 and UMCON1 in the UART block.

Register	Address	R/W	Description	Reset Value
UMCON0	0x1500000C	R/W	UART channel 0 Modem control register	0x0
UMCON1	0x1500400C	R/W	UART channel 1 Modem control register	0x0

UMCONn	Bit	Description	Initial State
Reserved	[7:5]	These bits must be 0's	00
AFC (Auto Flow Control)	[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 S3C2400X will control nRTS automatically. If AFC bit is disabled, nRTS must be controlled by S/W. 0 = 'H' level(Inactivate nRTS) 1 = 'L' level(Activate nRTS)	0

UART TX/RX STATUS REGISTER

There are two UART Tx/Rx status registers, UTRSTAT0 and UTRSTAT1, in the UART block.

Register	Address	R/W	Description	Reset Value
UTRSTAT0	0x15000010	R	UART channel 0 Tx/Rx status register	0x6
UTRSTAT1	0x15004010	R	UART channel 1 Tx/Rx status register	0x6

UTRSTATn	Bit	Description	Initial State
Transmitter empty	[2]	This bit is automatically set to 1 when the transmit buffer register has no valid data to transmit and the transmit shift register is empty. 0 = Not empty 1 = Transmit buffer & shifter register empty	1
Transmit buffer empty	[1]	This bit is automatically set to 1 when the 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]	This bit is automatically set to 1 whenever the 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 two UART Rx error status registers, UERSTAT0 and UERSTAT1, in the UART block.

Register	Address	R/W	Description	Reset Value
UERSTAT0	0x15000014	R	UART channel 0 Rx error status register	0x0
UERSTAT1	0x15004014	R	UART channel 1 Rx error status register	0x0

UERSTATn	Bit	Description	Initial State
Break Detect	[3]	This bit is automatically set to 1 to indicate that a break signal has been received. 0 = No break receive 1 = Break receive(Interrupt is requested)	0
Frame Error	[2]	This bit is automatically set to 1 whenever a frame error occurs during receive operation. 0 = No frame error during receive 1 = Frame error(Interrupt is requested)	0
Parity Error	[1]	This bit is automatically set to 1 whenever a parity error occurs during receive operation. 0 = No parity error during receive 1 = Parity error(Interrupt is requested)	0
Overrun Error	[0]	This bit is automatically set to 1 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 two UART FIFO status registers, UFSTAT0 and UFSTAT1, in the UART block.

Register	Address	R/W	Description	Reset Value
UFSTAT0	0x15000018	R	UART channel 0 FIFO status register	0x00
UFSTAT1	0x15004018	R	UART channel 1 FIFO status register	0x00

UFSTATn	Bit	Description	Initial State
Reserved	[15:10]		0
Tx FIFO Full	[9]	This bit is automatically set to 1 whenever transmit FIFO is full during transmit operation 0 = 0-byte ≤ Tx FIFO data ≤ 15-byte 1 = Full	0
Rx FIFO Full	[8]	This bit is automatically set to 1 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, UMSTAT0 and UMSTAT1, in the UART block.

Register	Address	R/W	Description	Reset Value
UMSTAT0	0x1500001C	R	UART channel 0 Modem status register	0x0
UMSTAT1	0x1500401C	R	UART channel 1 Modem status register	0x0

UMSTAT0	Bit	Description	Initial State
Reserved	[3]		0
Delta CTS	[2]	This bit indicates that the nCTS input to S3C2400X has changed state since the last time it was read by CPU. (Refer to Figure 11-7) 0 = Has not changed 1 = Has changed	0
Reserved	[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

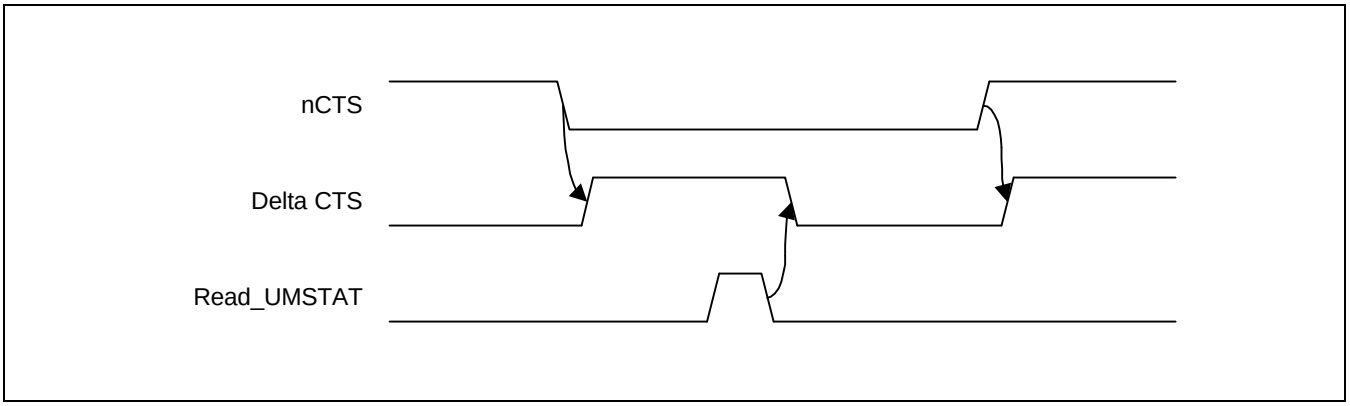


Figure 11-8. nCTS and Delta CTS Timing Diagram

UART TRANSMIT BUFFER REGISTER(HOLDING REGISTER & FIFO REGISTER)

UTXHn has an 8-bit data for transmission data

Register	Address	R/W	Description	Reset Value
UTXH0	0x15000020(L) 0x15000023(B)	W (by byte)	UART channel 0 transmit buffer register	–
UTXH1	0x15004020(L) 0x15004023(B)	W (by byte)	UART channel 1 transmit buffer register	–

UTXHn	Bit	Description	Initial State
TXDATAn	[7:0]	Transmit data for UARTn	–

NOTE: (L): When the endian mode is Little endian.
(B): When the endian mode is Big endian.

UART RECEIVE BUFFER REGISTER (HOLDING REGISTER & FIFO REGISTER)

URXHn has an 8-bit data for received data

Register	Address	R/W	Description	Reset Value
URXH0	0x15000024(L) 0x15000027(B)	R (by byte)	UART channel 0 receive buffer register	–
URXH1	0x15004024(L) 0x15004027(B)	R (by byte)	UART channel 1 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 USTATn had been cleared.

UART BAUD RATE DIVISION REGISTER

The value stored in the baud rate divisor register, UBRDIV, is used to determine the serial Tx/Rx clock rate (baud rate) as follows:

UBRDIVn = (int)(PCLK / (bps x 16)) -1

where the divisor should be from 1 to (2^{16}-1). For example, if the baud-rate is 115200 bps and PCLK 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	0x15000028	R/W	Baud rate divisior register 0	–
UBRDIV1	0x15004028	R/W	Baud rate divisior register 1	–

UBRDIV n	Bit	Description	Initial State
UBRDIV	[15:0]	Baud rate division value UBRDIVn > 0	–

12 USB HOST CONTROLLER

OVERVIEW

S3C2400X supports 2 port USB host interface as follows;

- Open HCI Rev 1.0 compatible.
- USB Rev1.1 compatible
- 2 down stream ports.
- Support for both LowSpeed and HighSpeed USB devices

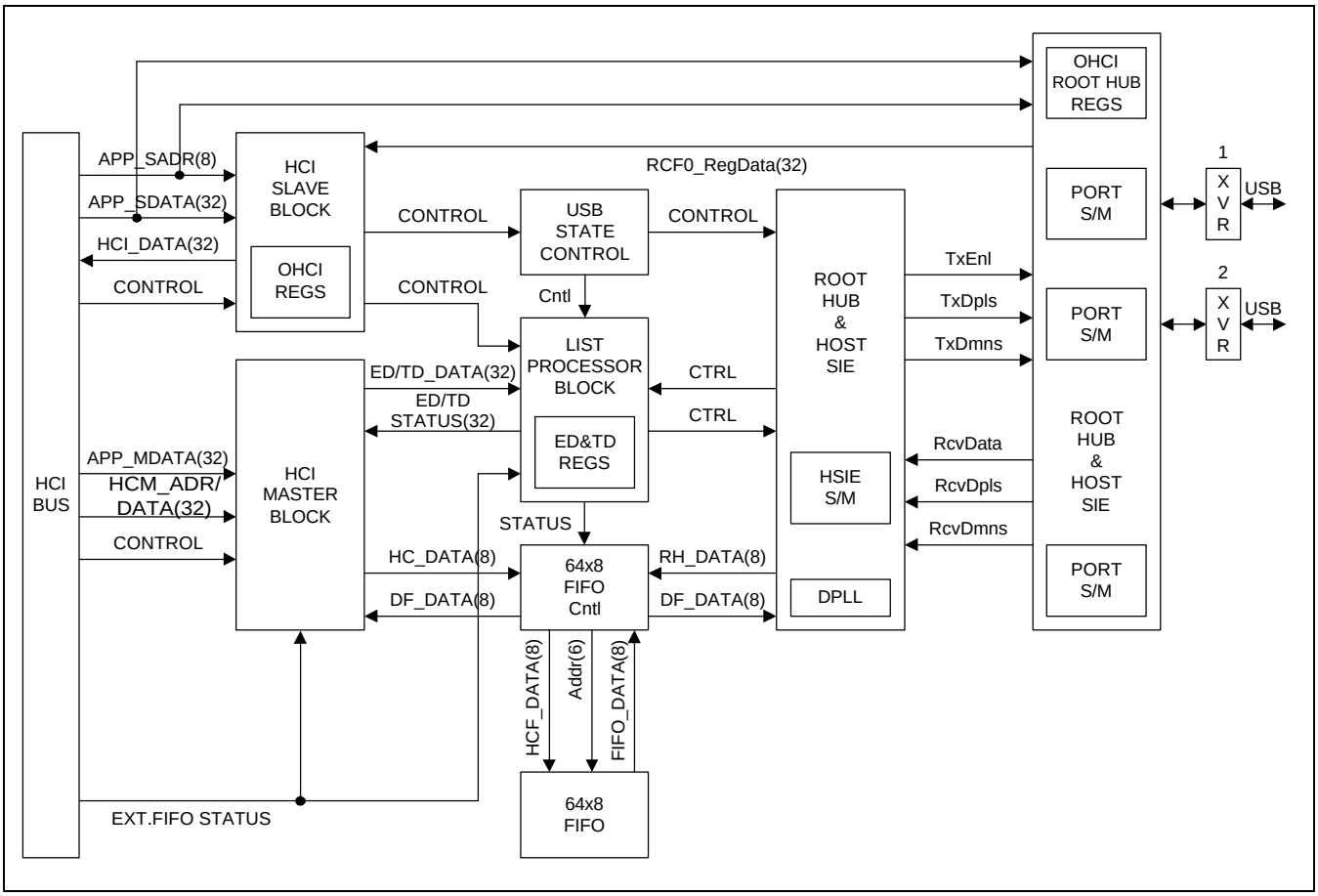


Figure 12-1. USB Host Controller Block Diagram

USB HOST CONTROLLER SPECIAL REGISTERS

The S3C2400X USB Host controller complies with OPEN HCI Rev 1.0. Please refer to Open Host Controller Interface Rev 1.0 specification for detail information.

OHCI REGISTERS FOR USB HOST CONTROLLER

Register	Base Address	R/W	Description	Reset Value
HcRevision	0x14200000	–	Control and status group	–
HcControl	0x14200004	–		–
HcCommonStatus	0x14200008	–		–
HcInterruptStatus	0x1420000c	–		–
HcInterruptEnable	0x14200010	–		–
HcInterruptDisable	0x14200014	–		–
HcHCCA	0x14200018	–	Memory pointer group	–
HcPeriodCuttentED	0x1420001c	–		–
HcControlHeadED	0x14200020	–		–
HcControlCurrentED	0x14200024	–		–
HcBulkHeadED	0x14200028	–		–
HcBulkCurrentED	0x1420002c	–		–
HcDoneHead	0x14200030	–		–
HcRmInterval	0x14200034	–	Frame counter group	–
HcFmRemaining	0x14200038	–		–
HcFmNumber	0x1420003c	–		–
HcPeriodicStart	0x14200040	–		–
HcLSThreshold	0x14200044	–		–
HcRhDescriptorA	0x14200048	–	Root hub group	–
HcRhDescriptorB	0x1420004c	–		–
HcRhStatus	0x14200050	–		–
HcRhPortStatus1	0x14200054	–		–
HcRhPortStatus2	0x14200058	–		–

13 USB DEVICE

OVERVIEW

USB function controller is designed to provide a high performance full speed function controller solution with DMA I/F.USB function controller allows bulk transfer with DMA, interrupt transfer and control transfer.

The integrated on-chip functions are as follows:

— Full Speed USB Function Controller compatible with the USB Specification Version 1.1

— DMA Interface for Bulk Transfer

— 5 Endpoint with FIFO
EP0: 16byte (dual port asynchronous ram)
EP1: 64byte IN FIFO (single port asynchronous ram): interrupt and DMA
EP2: 64byte IN FIFO (single port asynchronous ram): interrupt
EP3: 64byte OUT FIFO (single port asynchronous ram): interrupt and DMA
EP4: 64byte OUT FIFO (single port asynchronous ram): interrupt

— Integrated USB Transceiver (ASIC USB Pad)

FEATURES

— Fully compliant to USB Specification Version 1.1

— Full Speed (12Mbps) Device

— Integrated USB Transceiver (ASIC USB Pad)

— Supports Control, Interrupt and Bulk transfer

— 5 Endpoint with FIFO:
One Bi-directional Control Endpoint with 16 byte FIFO (EP0)
Two Receive Bulk Endpoint with 64 byte FIFO (EP1, EP2)
Two Transmit Bulk Endpoint with 64 byte FIFO (EP3, EP4)

— Supports DMA interface for receive and transmit bulk endpoints. (EP1, EP2, EP3, EP4)

— Independent 64byte receive and transmit FIFO to maximize throughput

— Support suspend and remote wake-up function

— Operating Frequency: 48MHz

— Not support ISO transfer

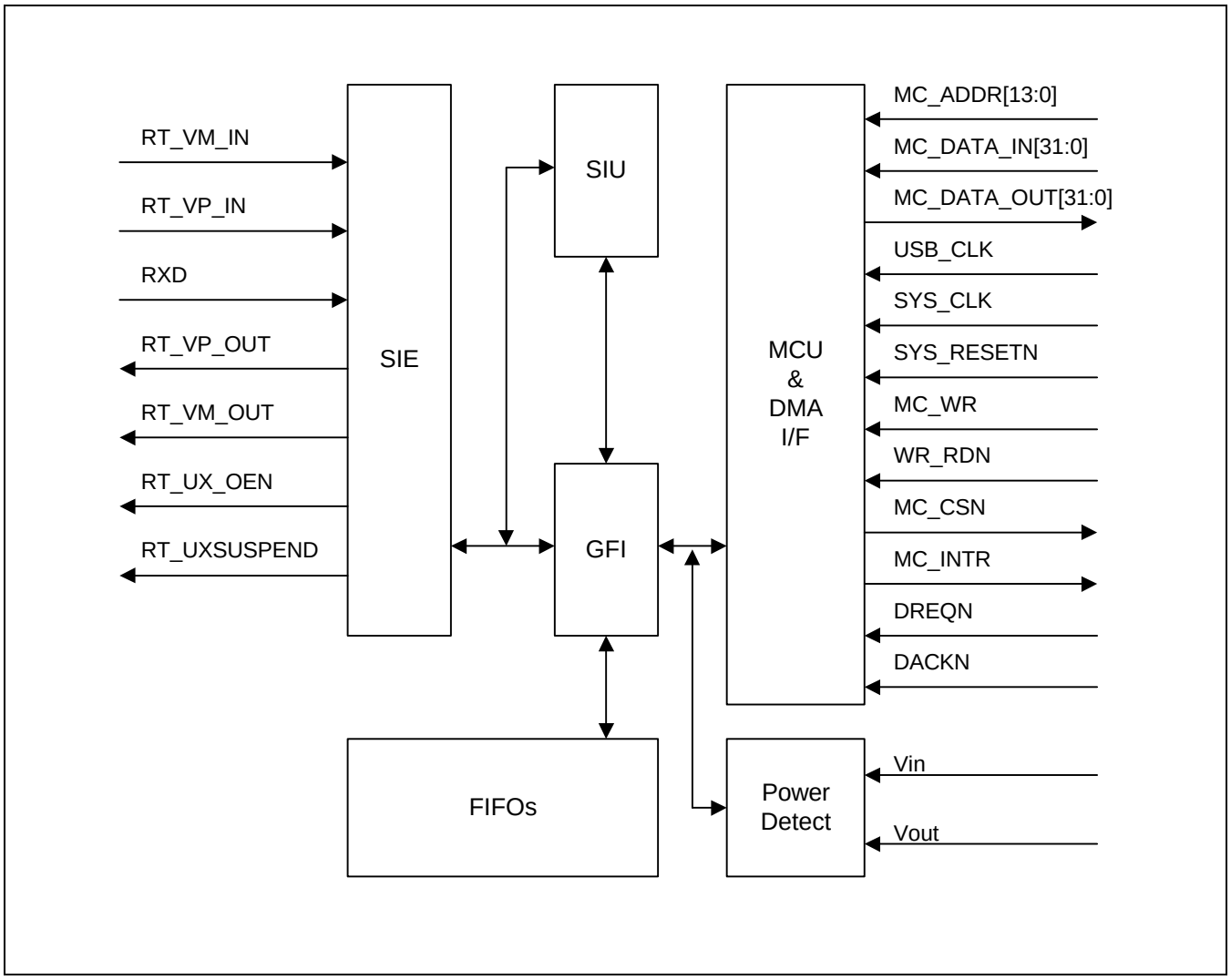


Figure 13-1. USB Device Block Diagram

USB DEVICE SPECIAL REGISTERS

This section describes the detail functionality about register set USB Device.
All Special Function Register is Word Access.
Reserved bit is zero.
All Register must be set after Host Reset Signaling.

FUNCTION ADDRES REGISTER (FUNC_ADDR_REG)

This register maintains the USB Device Address assigned by the host. The S3C2400X 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	0x15200140	R/W	Function address register	0x00000000

FUNC_ADDR_REG	Bit	MCU	USB	Description	Initial State
Reserved	[31:8]	R			0
ADDR_UPDATE	[7]	SET	R /CLEAR	The MCU sets this bit 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.	0000000

POWER MANAGEMENT REGISTER (PWR_REG)

This register is power control register in USB block.

Register	Address	R/W	Description	Reset Value
PWR_REG	0x15200144	R/W	Power management register	0x00000000

FUNC_ADDR	Bit	MCU	USB	Description	Initial State
Reserved	[31:9]				0
CLK_MASK	[8]	R/W	R	The MCU sets this bit to mask USB clock: 0 = Normal operation, 1 = USB clock mask Clear Method : The MCU writes ‘high’ on the MCU_RESET(bit [7]).	
MCU_RESET	[7]	W	R	The MCU sets this bit to reset USB: 0 = Normal operation, 1 = S/W Reset Clear Method: The MCU write 'low' on this bit.	0
EP0_FLUSH	[6]	R/W	R	The MCU sets this bit to flush EP0 FIFO	0
Reserved	[5]	–	–		0
Reserved	[4]	–	–		0
USB_RESET	[3]	R	SET	The USB sets this bit 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]	W	R /CLEAR	The MCU sets this bit for MCU resume. The USB generates the resume signaling depending RESUME CON Register, while this bit is set in suspend mode.	
SUSPEND_MODE	[1]	R	SET /CLEAR	This bit can be set by USB, automatically when the device enter into suspend mode. It can be cleared under the MCU or USB resume conditions.	0
SUSPEND_EN	[0]	R/W	R	Suspend mode enable control bit: 0 = Disable (default) 1 = Enable Disables masked suspend mode	0

INTERRUPT REGISTER (INT_REG)

These registers act as status registers for the MCU when there is an interrupt event. The bits in these registers are cleared by the MCU by writing a 1(Not "0") to each bit. 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
INT_REG	0x15200148	R	Interrupt pending/clear register	0x00000000

INT_REG	Bit	MCU	USB	Description	Initial State
ERROR Interrupt	[8]	R /CLEAR	SET	Error monitoring interrupt source. It can be set under the following conditions: 1. crc error 2. bit stuff error	0
RESET Interrupt	[7]	R /CLEAR	SET	Reset interrupt It can be set under the following conditions: 1. USB Reset 2. MCU Reset	0
RESUME Interrupt	[6]	R /CLEAR	SET	The USB sets this bit, when it receives the Resume signaling (Resume Command in USB bus), while in Suspend mode. If the Resume is due to a USB reset, then the MCU is first interrupted into a RESUME interrupt.	0
SUSPEND Interrupt	[5]	R /CLEAR	SET	The USB sets this bit when it receives the Suspend signaling. This bit is set whenever there is no activity during 3ms on the USB bus. Thus, if the MCU does not stop the clock after the first Suspend interrupt, that it will be continually to be interrupted every 3ms as long as there is no activity on the USB bus. By default, this interrupt is masked.	0
EP4 Interrupt	[4]	R /CLEAR	SET	Endpoint4 interrupt: 1. Sets OUT_PKT_RDY bit.(EP4_OUT_CSR1[0]) 2. Sets SENT_STALL bit.(EP4_OUT_CSR1[6])	0
EP3 Interrupt	[3]	R /CLEAR	SET	Endpoint3 interrupt: 1. Sets OUT_PKT_RDY bit.(EP3_OUT_CSR1[0]) 2. Sets SENT_STALL bit.(EP3_OUT_CSR1[6])	0
EP2 Interrupt	[2]	R /CLEAR	SET	Endpoint2 interrupt: 1. IN_PKT_RDY is cleared.(EP2_IN_CSR1[0]) 2. FIFO is flushed.(EP2_IN_CSR[2])	0
EP1 Interrupt	[1]	R /CLEAR	SET	Endpoint1 interrupt: 1. IN_PKT_RDY is cleared.(EP1_IN_CSR1[0]) 2. FIFO is flushed.(EP1_IN_CSR[2])	0
EP0 Interrupt	[0]	R /CLEAR	SET	Endpoint0 interrupt: 1. OUT_PKT_RDY bit (D0) is set.(EP0_CSR[0]) 2. IN_PKT_RDY bit (D1) is cleared.(EP0_CSR[1]) 3. SENT_STALL bit (D2) is set.(EP0_CSR[2]) 4. SETUP_END bit (D4) is set.(EP0_CSR[4]) 5. DATA_END bit is cleared.(EP0_CSR[3]) (Indicates End of control transfer)	0

INTERRUPT MASK REGISTER (INT_MASK_REG)

This register can mask interrupt(except RESUME Interrupt).

Register	Address	R/W	Description	Reset Value
INT_MASK_REG	0x1520014C	R/W	Determines which interrupt is masked. The masked interrupt will not be serviced. 0 = Interrupt is available 1 = Interrupt is masked	0x033F

INT_MASK_REG	Bit	MCU	USB	Description	Initial State
Reserved	[31:10]				0
CRC_MASK	[9]	R/W	R	CRC Error Interrupt Mask bit: 0 = Interrupt enable 1 = Masked	1
BIT_STUFF_MASK	[8]	R/W	R	BIT_STUFF Error interrupt Mask bit: 0 = Interrupt enable 1 = Masked	1
RESET_MASK	[7]	R/W	R	RESET Interrupt Mask bit: 0 = Interrupt enable 1 = Masked	0
Reserved	[6]				0
SUSPEND_MASK	[5]	R/W	R	SUSPEND Interrupt Mask bit: 0 = Interrupt enable 1 = Masked	1
EP4_MASK	[4]	R/W	R	EP4 Interrupt Mask bit: 0 = Interrupt enable 1 = Masked	1
EP3_MASK	[3]	R/W	R	EP3 Interrupt Mask bit: 0 = Interrupt enable 1 = Masked	1
EP2_MASK	[2]	R/W	R	EP2 Interrupt Mask bit: 0 = Interrupt enable 1 = Masked	1
EP1_MASK	[1]	R/W	R	EP1 Interrupt Mask bit: 0 = Interrupt enable 1 = Masked	1
EP0_MASK	[0]	R/W	R	EP0 Interrupt Mask bit: 0 = Interrupt enable 1 = Masked	1

FRAME NUMBER REGISTER (FPAME_NUM_REG)

When host transfer USB packet, there is frame number in SOF(Start Of Frame). The USB catch this frame number and load it into this register, automatically.

Register	Address	R/W	Description	Reset Value
FRAME_NUM_REG	0x15200150	R	Frame number register	0x00000000

FRAME_NUM_REG	Bit	MCU	USB	Description	Initial State
Reserved	[31:11]				0
FRAME_NUM	[10:0]	R	W	Frame number value	00000000000

RESUME SIGNAL CONTROL REGISTER (RESUME_CON_REG)

This register can control resume signal length. 5-bit down counts.

Register	Address	R/W	Description	Reset Value
RESUME_CON_REG	0x15200154	R/W	Resume signal control register	0x000A

RESUME_CON_REG	Bit	MCU	USB	Description	Initial State
Reserved	[31:5]				0
RESUME_CON	[4:0]	R/W	R	The value of these 5-bits means the length of resume signal. The unit is mili-second. 00000 = Can not be used 00001 = 1 ms 00010 = 2 ms 00011 = 3 ms 00100 = 4 ms 00101 = 5 ms 00110 = 6 ms 00111 = 7 ms 01000 = 8 ms 01001 = 9 ms 01010 = 10 ms 01011 = 11 ms 01100 = 12 ms 01101 = 13 ms 01110 = 14 ms 01111 = 15 ms 10000 = 16 ms 10001 = 17 ms 10010 = 18 ms 10011 = 19 ms 10100 = 20 ms 10101 = 21 ms 10110 = 22 ms 10111 = 23 ms 11000 = 24 ms 11001 = 25 ms 11010 = 26 ms 11011 = 27 ms 11100 = 28 ms 11101 = 29 ms 11110 = 30 ms 11111 = 31 ms	01010

END POINT0 CONTROL STATUS REGISTER (EP0_CSR)

Register	Address	R/W	Description	Reset Value
EP0_CSR	0x15200160	R/W	Clock generator control Register	0x00000000

EP0_CSR	Bit	MCU	USB	Description	Initial State
Reserved	[31:8]				0
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	The MCU should writes 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	The USB sets this bit 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	The MCU sets this bit below conditions: 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	The USB sets this bit 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
IN_PKT_RDY	[1]	SET	CLEAR	The MCU sets this bit 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	The USB sets this bit 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 POINT0 MAX PACKET REGISTER (EP0_MAXP)

Register	Address	R/W	Description	Reset Value
EP0_MAXP	0x15200164	R/W	End Point0 MAX packet register	0x00000001

EP0_MAXP	Bit	MCU	USB	Description	Initial State
Reserved	[31:2]				0
EP0_MAXP	[1:0]	R/W	R	00: MAXP = 0 01: MAXP = 8 10: MAXP = 16	01

END POINT0 OUT WRITE COUNT REGISTER (EP0_OUT_CNT)

This register has the number of bytes consist of one packet. Initially, OUT Write Count Register should keep the number of bytes consisting one packet and MCU should fetches number of bytes as many bytes as this register value indicates.

Register	Address	R/W	Description	Reset Value
EP0_OUT_CNT	0x15200168	R	End Point0 out write count register	0x00000000

EP0_OUT_CNT	Bit	MCU	USB	Description	Initial State
Reserved	[31:5]				0
OUT_CNT	[4:0]	R	R	EP0 out write count value	00000

END POINT0 FIFO READ/WRITE REGISTER (EP0_FIFO)

End Point0 is for input or output to host. To access EP0 FIFO, the MCU should access FIFO through EP0_FIFO register. Lower 8-bit is valid.

Register	Address	R/W	Description	Reset Value
EP0_FIFO	0x1520016C	R/W	End Point0 FIFO read/write register	0x000000xx

EP0_FIFO	Bit	MCU	USB	Description	Initial State
Reserved	[31:8]				0
FIFO_DATA	[7:0]	R/W	R/W	EP0 FIFO data values	0Xxx

END POINT IN CONTROL STATUS REGISTER

Register	Address	R/W	Description	Reset Value
EP1_IN_CSR	0x15200180	R/W	END POINT1 in control status register	0x00000000
EP2_IN_CSR	0x15200190	R/W	END POINT2 in control status register	0x00000000

EPn_IN_CSR	Bit	MCU	USB	Description	Initial State
Reserved	[31:8]				0
AUTO_SET	[7]	R/W	R	In case of single packet mode, if the packet size is same as FIFO size, IN_PKT_RDY bit can be set automatically (AUTO_SET=1) when MCU write a packet data into FIFO. This is a special feature in case that the single packet size is same as FIFO size. If it's not this case, MCU should set IN_PKT_RDY as explained in EPn_IN_CSR register.	0
DMA_IPR_IN_MASK	[6]	R/W	R	This bit determines whether the interrupt should be issued, or not, when the EP1 IN_PKT_RDY condition happens. This is only useful for DMA mode. 0 = Interrupt enable, 1 = Interrupt Masking	0
CLR_DATA_TOGGLE	[5]	R/(W)	R/ CLEAR	This bit can be 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	[4]	R/ CLEAR	SET	The USB sets this bit 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	[3]	W/R	R	0 = The MCU clears this bit to finish the STALL condition. 1 = The MCU issues a STALL and shake to the USB.	0

EPn_IN_CSR	Bit	MCU	USB	Description	Initial State
FIFO_FLUSH	[2]	R/W	CLEAR	The MCU sets this bit 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	[1]				0
IN_PKT_RDY	[0]	SET/ R	CLEAR	The MCU sets this bit after writing a packet 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 as the MCU to load the next packet. While this bit is set by MCU after writing a packet data into the FIFO, the MCU will not be able to write a new packet data to the FIFO without clearing by USB. If the SEND_STALL bit is set by the MCU, this bit also cannot be set.	0

END POINT IN MAX PACKET REGISTER

Register	Address	R/W	Description	Reset Value
EP1_IN_MAXP	0x15200184	R/W	End Point1 in MAX packet register	0x00000001
EP2_IN_MAXP	0x15200194	R/W	End Point2 in MAX packet register	0x00000001

EPn_IN_MAXP	Bit	MCU	USB	Description	Initial State
Reserved	[31:4]				0
IN_MAXP	[3:0]	R/W	R	0000 : MAXP = 0 0001 : MAXP = 8 0010 : MAXP = 16 0011 : MAXP = 24 0100 : MAXP = 32 0101 : MAXP = 40 0110 : MAXP = 48 0111 : MAXP = 56 1000 : MAXP = 64	0001

END POINT IN FIFO WRITE REGISTER

End Point1 or End Point2 is for output to host. To access EP1 FIFO or EP2 FIFO, the MCU should access FIFO through EP1_FIFO register or EP2_FIFO register. Lower 8-bit is valid.

Register	Address	R/W	Description	Reset Value
EP1_FIFO	0x15200188	W	End Point1 FIFO write register	0x000000xx
EP2_FIFO	0x15200198	W	End Point2 FIFO write register	0x000000xx

EPn_FIFO	Bit	MCU	USB	Description	Initial State
Reserved	[31:8]				0
IN_FIFO_DATA	[7:0]	W	R	IN FIFO data value	0Xxx

END POINT OUT CONTROL STATUS REGISTER

Register	Address	R/W	Description	Reset Value
EP3_OUT_CSR	0x152001A0	R/W	End Point3 out control status register	0x00000000
EP4_OUT_CSR	0x152001B0	R/W	End Point4 out control status register	0x00000000

EPn_OUT_CSR	Bit	MCU	USB	Description	Initial State
Reserved	[31:8]				0
AUTO_CLR	[7]	R/W	R	If MCU set, whenever the MCU reads data from the OUT FIFO, OUT_PKT_RDY will automatically be cleared by the logic, without any intervention from MCU.	0
DMA_OPR_INT _MASK	[6]	R/W	R	This bit determines whether the interrupt should be issued, or not, when the EP3 OUT_PKT_RDY condition happens. This is only useful for DMA mode 0 = Interrupt Enable 1 = Interrupt Masking	0
CLR_DATA _TOGGLE	[5]	R/W	CLEAR	When the MCU writes a 1 to this bit, the data toggle sequence bit is reset to DATA0.	0
SENT_STALL	[4]	CLEAR /R	SET	The USB sets this bit 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	[3]	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	[2]	R/W	CLEAR	The MCU write 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	[1]				0
OUT_PKT_RDY	[0]	R/ CLEAR	SET	The USB sets this bit 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 MAX PACKET REGISTER

Register	Address	R/W	Description	Reset Value
EP3_OUT_MAXP	0x152001A4	R/W	End Point3 out MAX packet register	0x00000001
EP4_OUT_MAXP	0x152001B4	R/W	End Point4 out MAX packet register	0x00000001

EPn_IN_MAXP	Bit	MCU	USB	Description	Initial State
Reserved	[31:4]				0
OUT_MAXP	[3:0]	R/W	R	0000 : MAXP = 0 0001 : MAXP = 8 0010 : MAXP = 16 0011 : MAXP = 24 0100 : MAXP = 32 0101 : MAXP = 40 0110 : MAXP = 48 0111 : MAXP = 56 1000 : MAXP = 64	0001

END POINT OUT WRITE COUNT REGISTER

This register has the number of bytes consist of one packet. Initially, OUT Write Count Register should keep the number of bytes consisting one packet and MCU should fetches number of bytes as many bytes as this register value indicates.

Register	Address	R/W	Description	Reset Value
EP3_OUT_CNT	0x152001A8	R	End Point3 out write count register	0x00000000
EP4_OUT_CNT	0x152001B8	R	End Point4 out write count register	0x00000000

EPn_OUT_CNT	Bit	MCU	USB	Description	Initial State
Reserved	[31:7]				0
OUT_CNT	[6:0]	R	W	Out write count value	0000000

END POINT FIFO READ REGISTER

End Point3 or End Point4 is for output to MCU. To access EP3 FIFO or EP4 FIFO, the MCU should access FIFO through EP3_FIFO Register or EP4_FIFO Register. Lower 8-bit is valid.

Register	Address	R/W	Description	Reset Value
EP3_FIFO	0x152001AC	R	End Point3 FIFO read register	0x000000xx
EP4_FIFO	0x152001BC	R	End Point4 FIFO read register	0x000000xx

EPn_FIFO	Bit	MCU	USB	Description	Initial State
Reserved	[31:8]				0
OUT_FIFO_DATA	[7:0]	R	W	Out FIFO data value	0Xxx

DMA INTERFACE CONTROL REGISTER (DMA_CON)

Register	Address	R/W	Description	Reset Value
DMA_CON	0x152001C0	R/W	DMA interface control register	0x00000000

CLKCON	Bit	MCU	USB	Description	Initial State
Reserved	[31:8]				0
IN_RUN_OB	[7]	R	W	In DMA Run Observation	0
STATE	[6:4]	R	W	DMA State Monitoring	0
DEMAND_MODE	[3]	R/W	R	Demand Mode Enable 0 = Disable 1 = Enable	0
OUT_DMA_RUN	[2]	R/W	CLEAR	This bit is used to start DMA operation for End Point3 0 = Stop 1 = Run	0
IN_DMA_RUN	[1]	R/W	CLEAR	This bit is used to start DMA operation for End Point1 0 = Stop 1 = Run	0
DMA_MODE_EN	[0]	R/W	R /CLEAR	This bit is used to set DMA mode 0 = Interrupt Mode 1 = DMA Mode for Bulk Endpoint	0

DMA UNIT COUNTER REGISTER (DMA_UNIT)

8-bit counter to setting DMA transfer unit. This counter value will be used to determine the number of data during one DMA request in the DMA demand mode.

Register	Address	R/W	Description	Reset Value
DMA_UNIT	0x152001C4	R/W	DMA transfer unit counter base register	0x00000000

DMA_UNIT	Bit	MCU	USB	Description	Initial State
UNIT_CNT	[7:0]	R/W	R	DMA transfer unit counter value	0x00

DMA FIFO COUNTER REGISTER (DMA_FIFO)

This register has byte size in FIFO to be transferred by DMA. In case of EP3 as soon as OUT_DMA_RUN enable, the value in OUT FIFO Write Count Register1 will be loaded in this register automatically. In case of EP1, the MCU should set proper value by S/W.

Register	Address	R/W	Description	Reset Value
DMA_FIFO	0x152001C8	R/W	DMA transfer FIFO counter base register	0x00000000

DMA_FIFO	Bit	MCU	USB	Description	Initial State
Reserved	[31:8]				0
FIFO_CNT	[7:0]	R/W	R	DMA transfer FIFO counter value	0x00

DMA TOTAL TRANSFER COUNTER REGISTER (DMA_TX)

This register (20-bit) should have total number of bytes to be transferred using DMA. If this counter reaches to 0, DMA_CON[2:0] will be cleared to 0. When DMA is active, writing this register may fail because USB also may update this register. So, If you want to write some value on this register, you have to check the write operation by verifying this register value.

Register	Address	R/W	Description	Reset Value
DMA_TX	0x152001CC	R/W	DMA total transfer counter base register	0x00000000

DMA_TX	Bit	MCU	USB	Description	Initial State
Reserved	[31:20]				0
TX_CNT	[19:0]	R/W	R/W	DMA total transfer count value	0x000000

TEST MODE CONTROL REGISTER (TEST_MODE)

Don't write this register in normal operation.

Register	Address	R/W	Description	Reset Value
TEST_MODE	0x152001F4	W	Test mode control register	0x00000000

TEST_MODE	Bit	MCU	USB	Description	Initial State
Reserved	[31:2]				0
EP1234_MEM _TEST	[1]	W	R	The MCU sets this bit to test EP1, EP2, EP3, and EP4 FIFO memory. This is set, and then the MCU can access EP1, EP2, EP3, and EP4 FIFO memory directly. 0 = Test disable (Normal operation mode) 1 = Test enable	0
EP0_MEM_TEST	[0]	W	R	The MCU sets this bit to test EP0 FIFO memory. This is set, and then the MCU can access EP0 FIFO memory directly. 0 = Test disable (Normal operation mode) 1 = Test enable	0

IN PACKET NUMBER CONTROL REGISTER (IN_CON_REG)

This register can control number of in packet terms of SOF(Start Of Frame)

Register	Address	R/W	Description	Reset Value
IN_CON_REG	0x152001F8	W	In packet number control register	0x00FF

IN_CON_REG	Bit	MCU	USB	Description	Initial State
Reserved	[31:8]				0
IN_CON_MASK	[7]	W	R	IN CON function masking 0 = Enable 1 = Masked (Normal operation mode)	1
IN_CON	[6:0]	W	R	This data is number of in packet terms of SOF. If this data were 3, USB Device responses NAK to 4th in packet.	1111111

NOTES

14 INTERRUPT CONTROLLER

OVERVIEW

The interrupt controller in S3C2400X receives the request from 32interrupt sources. These interrupt sources are provided by internal peripheral such as the DMA controller, UART and IIC, etc. In these interrupt sources, the UART0 and UART1 error interrupts are 'OR'ed to the interrupt controller.

The role of the interrupt controller is to ask for the FIQ or IRQ interrupt requests to the ARM920T core after the arbitration process when there are multiple interrupt requests from internal peripherals and external interrupt request pins.

The arbitration process is performed by the hardware priority logic and the result is written to the interrupt pending register and users notice that register to know which interrupt has been requested.

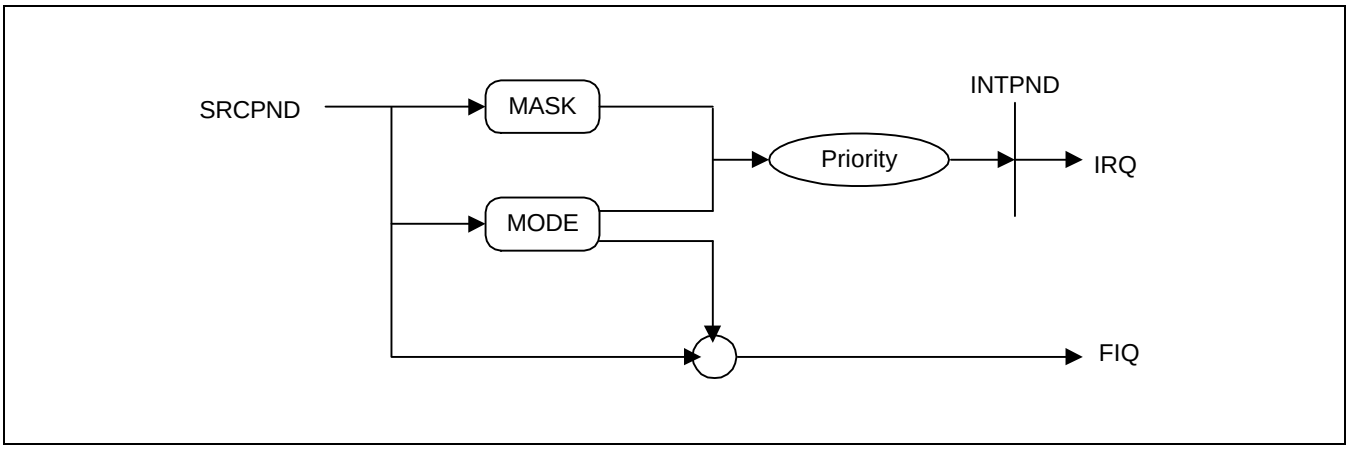


Figure 14-1. Interrupt Process Diagram

INTERRUPT CONTROLLER OPERATION

F-bit and I-bit of PSR (program status register)

If the F-bit of PSR (program status register in ARM920T CPU) is set to 1, the CPU does not accept the FIQ (fast interrupt request) from the interrupt controller. If I-bit of PSR (program status register in ARM920T CPU) is set to 1, the CPU does not accept the IRQ (interrupt request) from the interrupt controller. So, to enable the interrupt reception, the F-bit or I-bit of PSR has to be cleared to 0 and also the corresponding bit of INTMSK has to be set to 0.

Interrupt Mode

ARM920T has 2 types of interrupt mode, FIQ or IRQ. All the interrupt sources determine the mode of interrupt to be used at interrupt request.

Interrupt Pending Register

S3C2400X has two interrupt pending resisters. The one is source pending register(SRCPND), the other is 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, at the same time the only one bit of INTPND register is set to 1 automatically after arbitration process. If interrupts are masked, the corresponding bits of SRCPND register are set to 1, but the bit of INTPND register is not changed. When a pending bit of 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 SRCPND register first and then clear the pending condition in INTPND registers same method.

Interrupt Mask Register

Indicates that an interrupt has been disabled if the corresponding mask bit is 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

Interrupt controller supports 32 interrupt sources as shown in below table. Two UART error interrupt requests are Ored to provide a single interrupt source to the interrupt controller.

Sources	Descriptions	Arbiter Group
INT_ADC	ADC EOC interrupt	ARB5
INT_RTC	RTC alarm interrupt	ARB5
INT_UTXD1	UART1 transmit interrupt	ARB5
INT_UTXD0	UART0 transmit interrupt	ARB5
INT_IIC	IIC interrupt	ARB4
INT_USBH	USB Host interrupt	ARB4
INT_USBD	USB Device interrupt	ARB4
INT_URXD1	UART1 receive interrupt	ARB4
INT_URXD0	UART0 receive interrupt	ARB4
INT_SPI	SPI interrupt	ARB4
INT_MMC	MMC 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
Reserved	Reserved for future use	ARB3
INT_UERR0/1	UART0/1 error Interrupt	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
EINT7	External interrupt 7	ARB1
EINT6	External interrupt 6	ARB1
EINT5	External interrupt 5	ARB1
EINT4	External interrupt 4	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 the following figure.

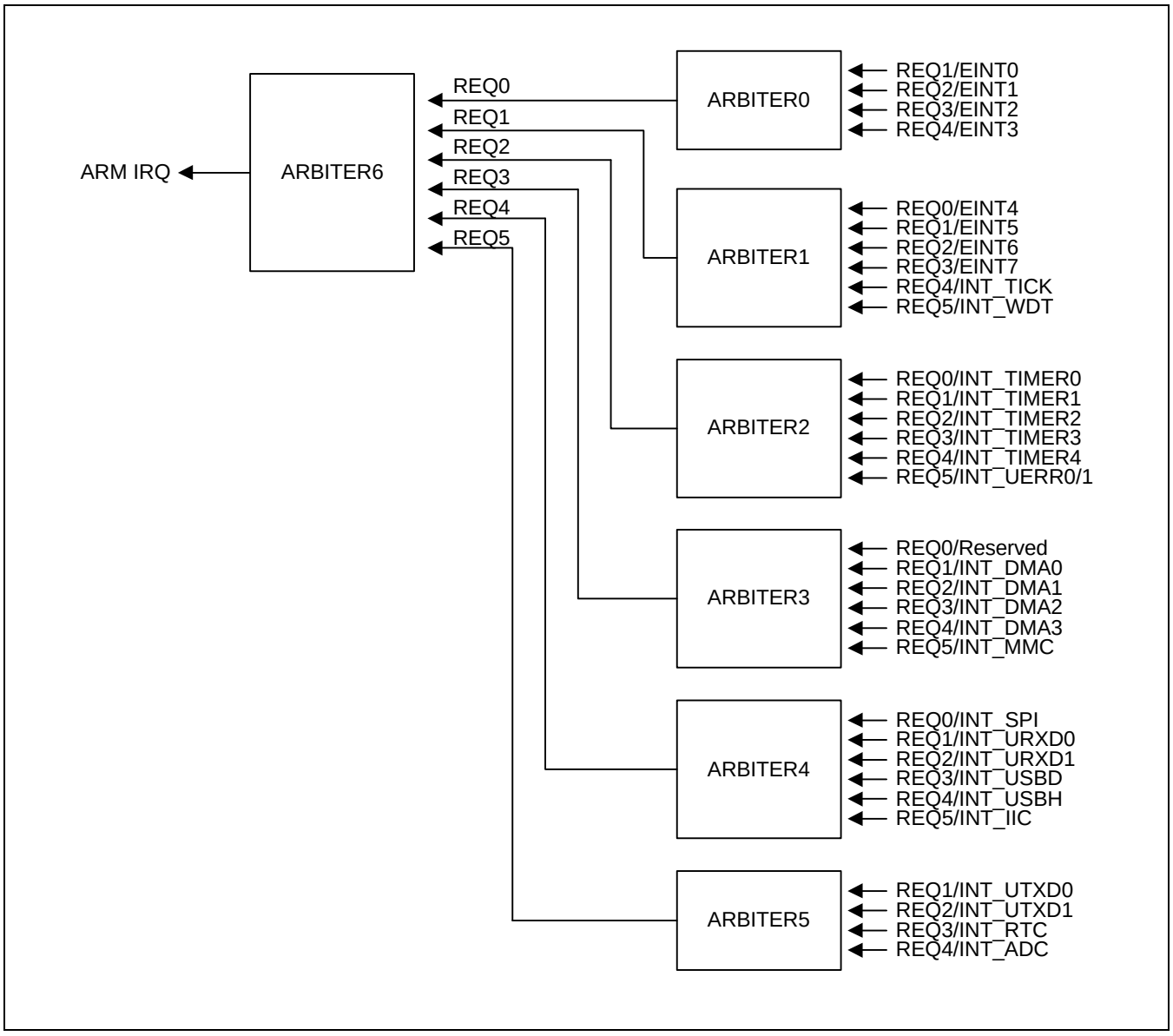


Figure 14-2. Priority Generating Block

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 is always the highest priority, and REQ5 is the lowest one. In addition, by changing the ARB_SEL bits, we can rotate the priority of REQ1 - REQ4.

Here, if ARB_MODE bit is set to 0, ARB_SEL bits are not automatically changed, thus the arbiter operates in the fixed priority mode. (Note that even in this mode, we can change 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 make REQ1 the lowest priority one. The detailed rule of ARB_SEL change is 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 based on the interrupt mode register, i.e., one FIQ request and the remaining IRQ requests. Arbitration process is performed for the multiple IRQ requests based on the priority register.

SOURCE PENDING REGISTER (SRCPND)

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. By reading this register, we can see the interrupt sources waiting for their requests to be serviced. Note that each bit of SRCPND register is automatically set by the interrupt sources regardless of the masking bits in the INTMASK register. In addition, it is not affected by the priority logic of interrupt controller.

In the interrupt service routine for a specific interrupt source, the corresponding bit of 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, interrupt controller operates as if another interrupt request comes in from the same source. In other words, if a specific bit of SRCPND register is set to 1, it is always considered as a valid interrupt request waiting to be serviced.

The specific time to clear the corresponding bit depends on the user's requirement. The bottom line is that 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 SRCPND register by writing a data to this register. It clears only the bit positions of 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 with no change

Register	Address	R/W	Description	Reset Value
SRCPND	0X14400000	R/W	Indicates the interrupt request status. 0 = The interrupt has not been requested. 1 = The interrupt source has asserted the interrupt request.	0x00000000

NOTE: When the user clear a interrupt pending, specific bit of SRCPND and INTPND, has to clear the bit of SRCPND.

SRCPND	Bit	Description	Initial State
INT_ADC	[31]	0 = Not requested, 1 = Requested	0
INT_RTC	[30]	0 = Not requested, 1 = Requested	0
INT_UTXD1	[29]	0 = Not requested, 1 = Requested	0
INT_UTXD0	[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
INT_URXD1	[24]	0 = Not requested, 1 = Requested	0
INT_URXD0	[23]	0 = Not requested, 1 = Requested	0
INT_SPI	[22]	0 = Not requested, 1 = Requested	0
INT_MMC	[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
Reserved	[16]	Not used	0
INT_UERR0/1	[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
EINT7	[7]	0 = Not requested, 1 = Requested	0
EINT6	[6]	0 = Not requested, 1 = Requested	0
EINT5	[5]	0 = Not requested, 1 = Requested	0
EINT4	[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 REGISTER (INTMOD)

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 at most 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 at most.

This register is write-only one, thus it cannot be read out.

Register	Address	R/W	Description	Reset Value
INTMOD	0X14400004	W	Interrupt mode register. 0 = IRQ mode 1 = FIQ mode	0x00000000

NOTE: If an interrupt mode is set to FIQ mode in INTMOD register, FIQ interrupt will not affect INTPND and INTOFFSET registers. The INTPND and INTOFFSET 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_UTXD1	[29]	0 = IRQ, 1 = FIQ	0
INT_UTXD0	[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
INT_URXD1	[24]	0 = IRQ, 1 = FIQ	0
INT_URXD0	[23]	0 = IRQ, 1 = FIQ	0
INT_SPI	[22]	0 = IRQ, 1 = FIQ	0
INT_MMC	[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
Reserved	[16]	Not used	0
INT_UERR0/1	[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
EINT7	[7]	0 = IRQ, 1 = FIQ	0
EINT6	[6]	0 = IRQ, 1 = FIQ	0
EINT5	[5]	0 = IRQ, 1 = FIQ	0
EINT4	[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 REGISTER (INTMSK)

Each of the 32 bits in the interrupt mask register is related to an interrupt source. If you set a specific bit 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 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	0X14400008	R/W	Determines 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_UTXD1	[29]	0 = Service available, 1 = Masked	1
INT_UTXD0	[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
INT_URXD1	[24]	0 = Service available, 1 = Masked	1
INT_URXD0	[23]	0 = Service available, 1 = Masked	1
INT_SPI	[22]	0 = Service available, 1 = Masked	1
INT_MMC	[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
Reserved	[16]	Not used	1
INT_UERR0/1	[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
EINT7	[7]	0 = Service available, 1 = Masked	1
EINT6	[6]	0 = Service available, 1 = Masked	1
EINT5	[5]	0 = Service available, 1 = Masked	1
EINT4	[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	0X1440000C	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
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 REGISTER (INTPND)

Each of the 32 bits in the interrupt pending register shows whether the corresponding interrupt request is the highest priority one that is unmasked and waits for the interrupt to be serviced. Since INTPND is located after the priority logic, only one bit can be set to 1 at most, and that is the very interrupt request generating IRQ to CPU. In interrupt service routine for IRQ, you can read this register to determine the interrupt source to be serviced among 32 sources.

Like the SRCPND, this register has to be cleared in the interrupt service routine after clearing SRCPND register. We can clear a specific bit of INTPND register by writing a data to this register. It clears only the bit positions of INTPND 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 with no change.

Register	Address	R/W	Description	Reset Value
INTPND	0X14400010	R/W	Indicates 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 is occurred, the corresponding bit of INTPND will not be turned on. Because the INTPND register is available only for IRQ mode interrupt.
2. Although the INTMASK is masking some interrupt source, the interrupt can be executed in the case that the corresponding interrupt has been set in the INTPND register

INTPND	Bit	Description	Initial State
INT_ADC	[31]	0 = Not requested, 1 = Requested	0
INT_RTC	[30]	0 = Not requested, 1 = Requested	0
INT_UTXD1	[29]	0 = Not requested, 1 = Requested	0
INT_UTXD0	[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
INT_URXD1	[24]	0 = Not requested, 1 = Requested	0
INT_URXD0	[23]	0 = Not requested, 1 = Requested	0
INT_SPI	[22]	0 = Not requested, 1 = Requested	0
INT_MMC	[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
Reserved	[16]	Not used	0
INT_UERR0/1	[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
EINT7	[7]	0 = Not requested, 1 = Requested	0
EINT6	[6]	0 = Not requested, 1 = Requested	0
EINT5	[5]	0 = Not requested, 1 = Requested	0
EINT4	[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 REGISTER (INTOFFSET)

The number 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	0X14400014	R	Indicates the IRQ interrupt request source	0x00000000

INT Source	The OFFSET value	INT Source	The OFFSET value
INT_ADC	31	INT_UERR0/1	15
INT_RTC	30	INT_TIMER4	14
INT_UTXD1	29	INT_TIMER3	13
INT_UTXD0	28	INT_TIMER2	12
INT_IIC	27	INT_TIMER1	11
INT_USBH	26	INT_TIMER0	10
INT_USBD	25	INT_WDT	9
INT_URXD1	24	INT_TICK	8
INT_URXD0	23	EINT7	7
INT_SPI	22	EINT6	6
INT_MMC	21	EINT5	5
INT_DMA3	20	EINT4	4
INT_DMA2	19	EINT3	3
INT_DMA1	18	EINT2	2
INT_DMA0	17	EINT1	1
Reserved	16	EINT0	0

NOTE: If the FIQ mode interrupt is occurred, the INTOFFSET will not be affected. Because the INTOFFSET register is available only for IRQ mode interrupt.

NOTES

15 LCD CONTROLLER

OVERVIEW

The LCD controller within S3C2400X consists of 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 FRC (Frame Rate Control) 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, 8-bit per pixel for interfacing with the palettized TFT color LCD panel and 16 non-palletized true-color display.

The LCD controller can be programmed to support the 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 Features

— Supports 3 types of LCD panels: 4-bit dual scan, 4-bit single scan, 8-bit single scan display type.

— Supports the monochrome, 4 gray levels, and 16 gray levels .

— Supports 256 level colors and 4096 level colors for color STN LCD panel.

— Supports multiple screen size.
Typical actual screen sizes: 320×240, 160×160 (pixels)
Maximum virtual screen sizes(color mode): 4096×1024, 2048×2048, 1024×4096, etc

TFT LCD Displays Features

— Supports 1, 2, 4 or 8 bpp (bit per pixel) palettized color displays for TFT.

— Supports 16 non-palettized true-color displays for color TFT.

— Supports maximum 64K color TFT at 16-bit per pixel mode.

— Supports multiple screen size.
Typical actual screen size: 720×240, 320×240, 160×160 (pixels)
Maximum virtual screen size (16 bpp mode): 2048×1024 etc

Common features

- Dedicated DMA supports to fetch the image data from video buffer located in system memory.
- Supports power saving mode.
- 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.

EXTERNAL INTERFACE SIGNAL

VFRAME / VSYNC:	Frame synchronous signal (STN) / Vertical synchronous signal (TFT).
VLINE / HSYNC:	Line synchronous pulse signal (STN) / Horizontal synchronous signal (TFT)
VCLK:	Pixel clock signal (STN/TFT)
VD[15:0]:	LCD pixel data output ports (STN/TFT)
VM / VDEN:	AC bias signal for the STN LCD driver (STN) / Data enable signal (TFT)
LEND:	Line end signal (TFT)

Total 21 output ports, data 16 bits, control 5 bits

BLOCK DIAGRAM

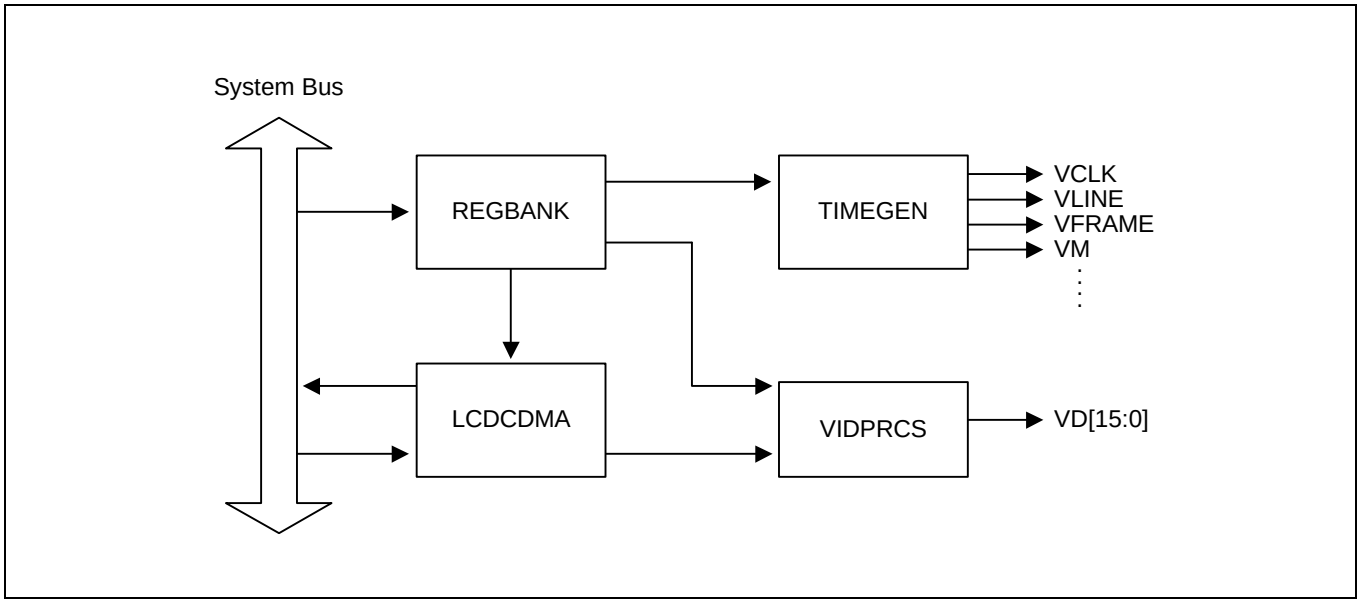


Figure 15-1. LCD Controller Block Diagram

The LCD controller within S3C2400X is used to transfer the video data and to generate the necessary control signals such as, VFRAME, VLINE, VCLK, VM and so on. As well as the control signals, S3C2400Xhas the data ports of video data, which are VD[15:0] as shown in Figure 15-1. The LCD controller consists of a REGBANK, LCDCDMA, VIDPRCS, and TIMEGEN (See Figure 15-1 LCD Controller Block Diagram). The REGBANK has 21 programmable register sets and 256x16 palette memory which are used to configure the LCD controller. The LCDCDMA is a dedicated DMA, which it 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 LCDCDMA and sends the video data through the VD[15:0] 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 requirement 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, 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 this kind of 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 24 words, which consists of FIFOL and FIFOH of 12 words, respectively. The S3C2400X has two FIFOs because it needs to support the dual scan display mode. In case of single scan mode, one of them can only be used.

LCD CONTROLLER OPERATION (STN CASE)

TIMING GENERATOR

The TIMEGEN generates the control signals for 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 register in the REGBANK. Based on these programmable configurations on the LCD control registers in REGBANK, the TIMEGEN can generate the programmable control signals suitable to support many different types of LCD drivers.

The VFRAME pulse is asserted for a 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 is used by the LCD driver to alternate the polarity of the row and column voltage used to turn the pixel on and off. The toggle rate of VM signal can be controlled by using the MMODE bit of LCDCON 1 register and MVAL field of 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 15-5 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 is controlled by 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, number of valid VD data lines should be 4. In case of 4-bit dual scan display the number of valid VD data lines should be 4 and in case of 8-bit single scan display mode, the number of valid VD data lines 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 can be controlled by the CLKVAL field in the LCDCON1 register. The 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 LCDCON1 and LCDCON2/3/4 registers as well as VCLK and HCLK. Most LCD drivers need their own adequate frame rate. The frame rate is calculated as follows;

frame_rate(Hz) = 1 / [{(1/VCLK) x (HOZVAL+1)+(1/HCLK) x (**A**+**B**+(LINEBLANK x 8)) } x (LINEVAL+1)]

$$\mathbf{A} = 2^{(4+WLH)},\ \mathbf{B} = 2^{(4+WDLY)}$$

Table 15-1. Relation between VCLK and CLKVAL(STN, HCLK=60MHz)

CLKVAL	60MHz/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 LCD controller within S3C2400X 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. When the gray or color mode is needed, the implementation of the shades of gray level or color should be followed by time-based dithering algorithm and FRC(Frame Rate Control) method can be used to implement the shades of gray or color from which selection can be made by using 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 gray mode and color mode in terms of the lookup table and FRC.

Lookup Table

The S3C2400X can support the lookup table for various selection of color or gray level mapping. This kind of selection gives users flexibility. 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, of course there is no selection as in the 256 color mode.

Gray Mode Operation

Two gray modes are supported by the LCD controller within the S3C2400X: 2-bit per pixel gray (4 level gray scale) or 4-bit per pixel gray (16 level gray scale). The 2-bit per pixel gray mode uses a lookup table, which allows selection on 4 gray levels among 16 possible gray levels. The 2-bit per pixel gray lookup table uses the BLUEVAL[15:0] in BLUELUT(Blue Lookup Table) 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. As same as in the case of level 0, 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, of course there is no selection as in the 4 gray levels.

256 Level Color Mode Operation

The LCD controller in S3C2400X 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.

Similarly with 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, we need 16bit for a lookup table because 2 bits are allocated for 4 blue levels, different from the 8 red or green levels.

4096 Level Color Mode Operation

The LCD controller in S3C2400Xcan support an 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-bits for red, 4-bits for green, and 4-bits for blue. The color display mode does not use lookup tables.

DITHERING AND FRC (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 a Time-based Dithering Algorithm for reducing flicker and FRC(Frame Rate Control) 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(Frame Rate Control). 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 as same as possible. 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 15-3 shows these 3 different display types for monochrome displays, and Figure 15-4 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 15-3. 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, Blue) of image data per pixel, 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 15-4 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, Blue) of image data per pixel, 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. 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]
•	
•	
•	

•

LCD Panel

A[31] A[30] ..... A[0] B[31] B[30] ..... B[0] .....

L[31] L[30] ..... L[0] M[31] M[30] ..... M[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]
•	
•	

LCD Panel

A[31] A[30] A[29] ..... A[0] B[31] B[30] ..... B[0] C[31] ..... C[0] .....

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, 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 color data format in words is as follows; (Video data must be 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)

LCD Self Refresh Mode

The LCD controller within S3C2400X can support the self refresh mode to reduce power consumption. The self refresh mode can only be applied to only the LCD which has the special LCD driver, for example, LCD panel of SED1580D from Seiko Epson Corporation. The SED1580D has the built-in monochrome display memory, which can display the previous stored image in the built-in monochrome display memory without image data fetch when the self refresh mode has been invoked. But SED1580D has the built-in display memory for monochrome only. So, the user has to set one more frame as a monochrome mode before the entering to self refresh mode. The kind of self refresh mode can be made by writing the control bit of SELFREF in the LCDCON5 register.

If the SELFREF bit is set to 1, the LCD controller enters into the self refresh mode from the next line. When the LCD controller enters into the self refresh mode, the signal of VCLK and VD should be fixed as Low and last data value, but the signal of VM, VFRAME, and VLINE will be generated continuously. To exit the self refresh mode, the user should disable SELFREF bit in LCDCON 5 register.

SL_IDLE Mode (LCD dedicated Idle Mode)

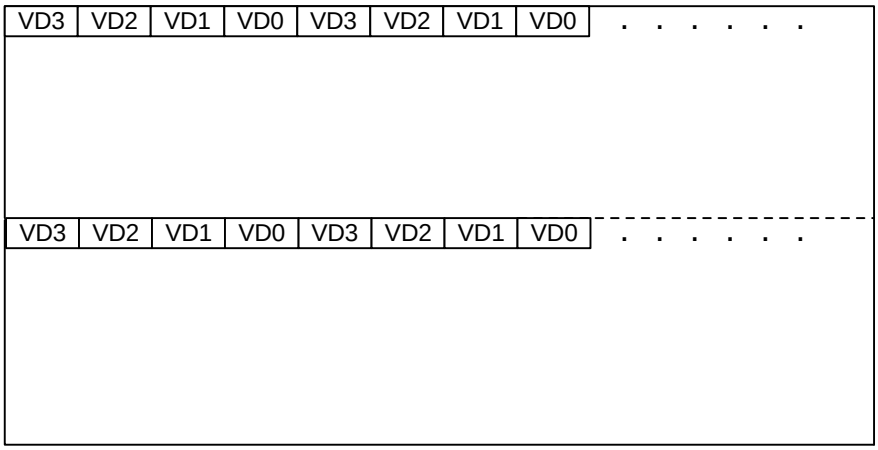
The SL_IDLE mode in the power management scheme should be used to enter into the LCD driver's self refresh mode. In SL_IDLE mode, all function blocks except the LCD controller within S3C2400X should be stopped to reduce the power consumption, because the power management block inserts divide_by_n input clock only to the LCD controller. For more information, please refer to the chapter, clock & power management and our example source code.

Timing Requirements

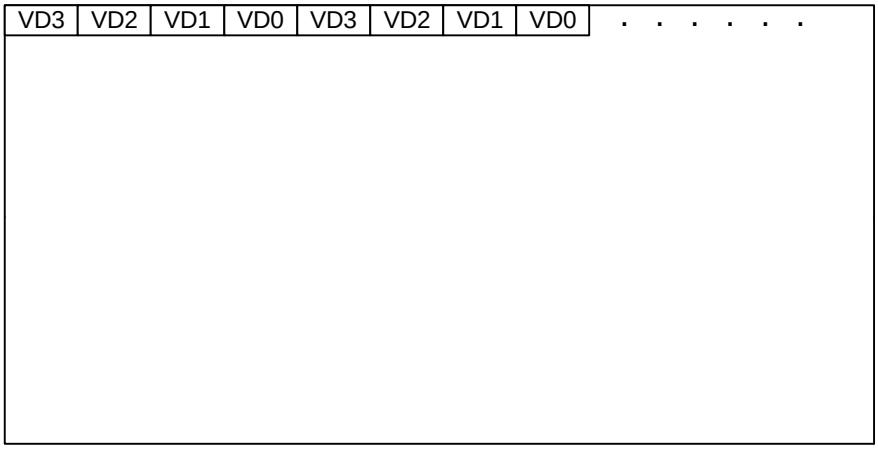
Image data should be transferred from the memory to the LCD driver using the VD[7:0] signal. VCLK signal is used to clock the data into the LCD driver's shift register. After each horizontal line of data has been shifted into the LCD driver's shift register, the VLINE signal is asserted to display the line on the panel.

The VM signal provides an AC signal for the display. It is used by the LCD to alternate the polarity of the row and column voltages, 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.

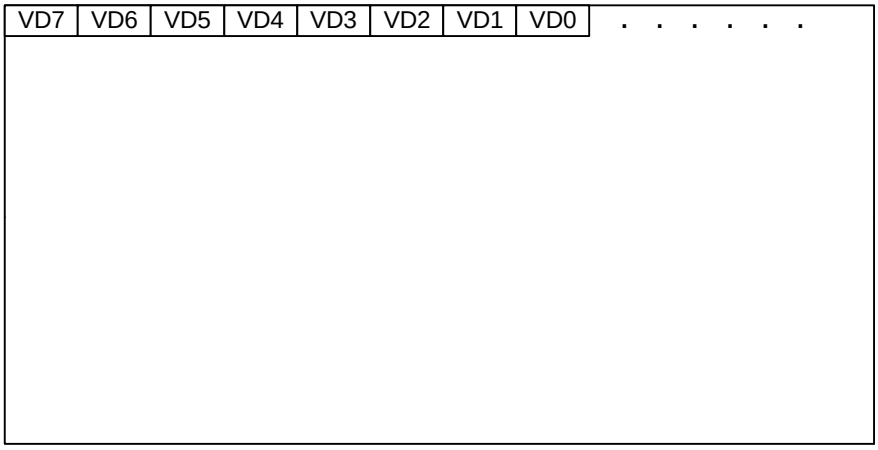
Figure 15-5 shows the timing requirements for the LCD driver interface.



4-bit Dual Scan Display

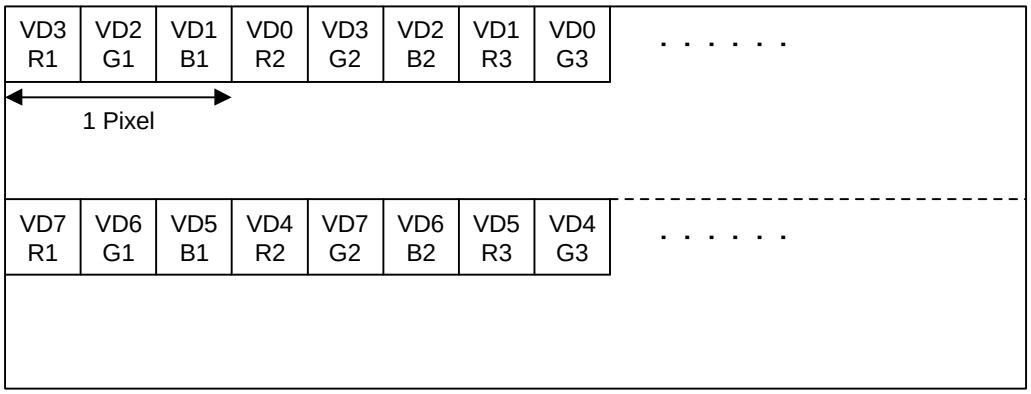


4-bit Single Scan Display

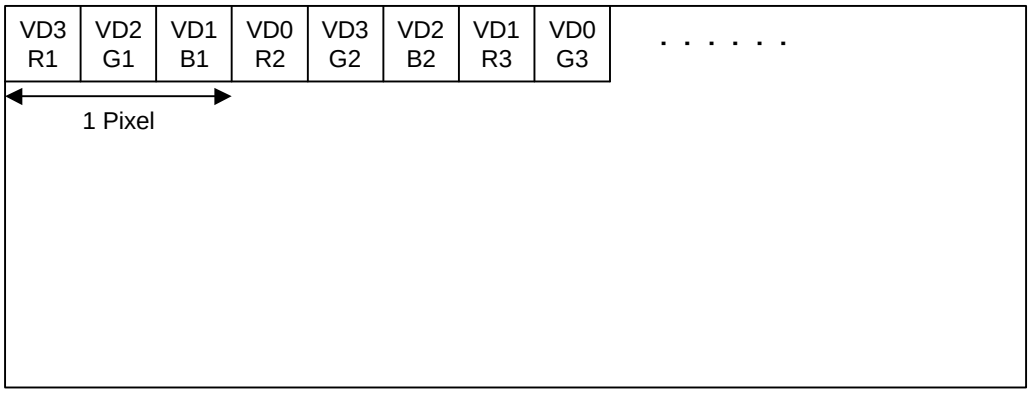


8-bit Single Scan Display

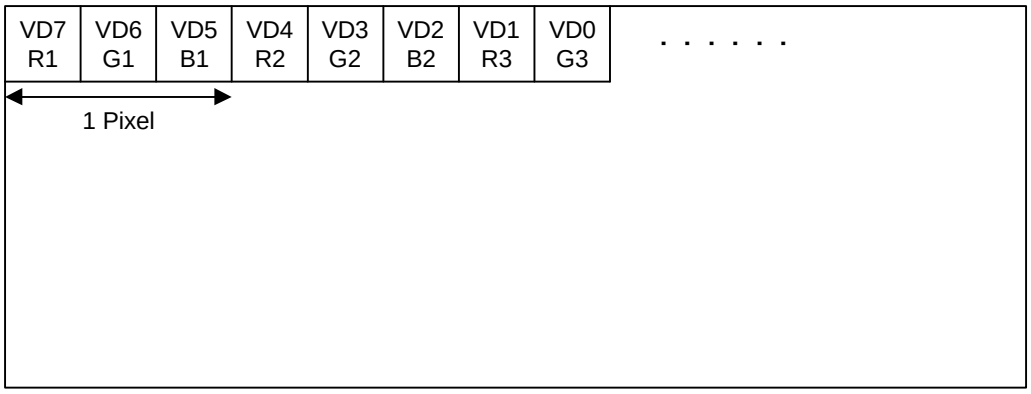
Figure 15-2. Monochrome Display Types (STN)



4-bit Dual Scan Display



4-bit Single Scan Display



8-bit Single Scan Display

Figure 15-3. Color Display Types (STN)

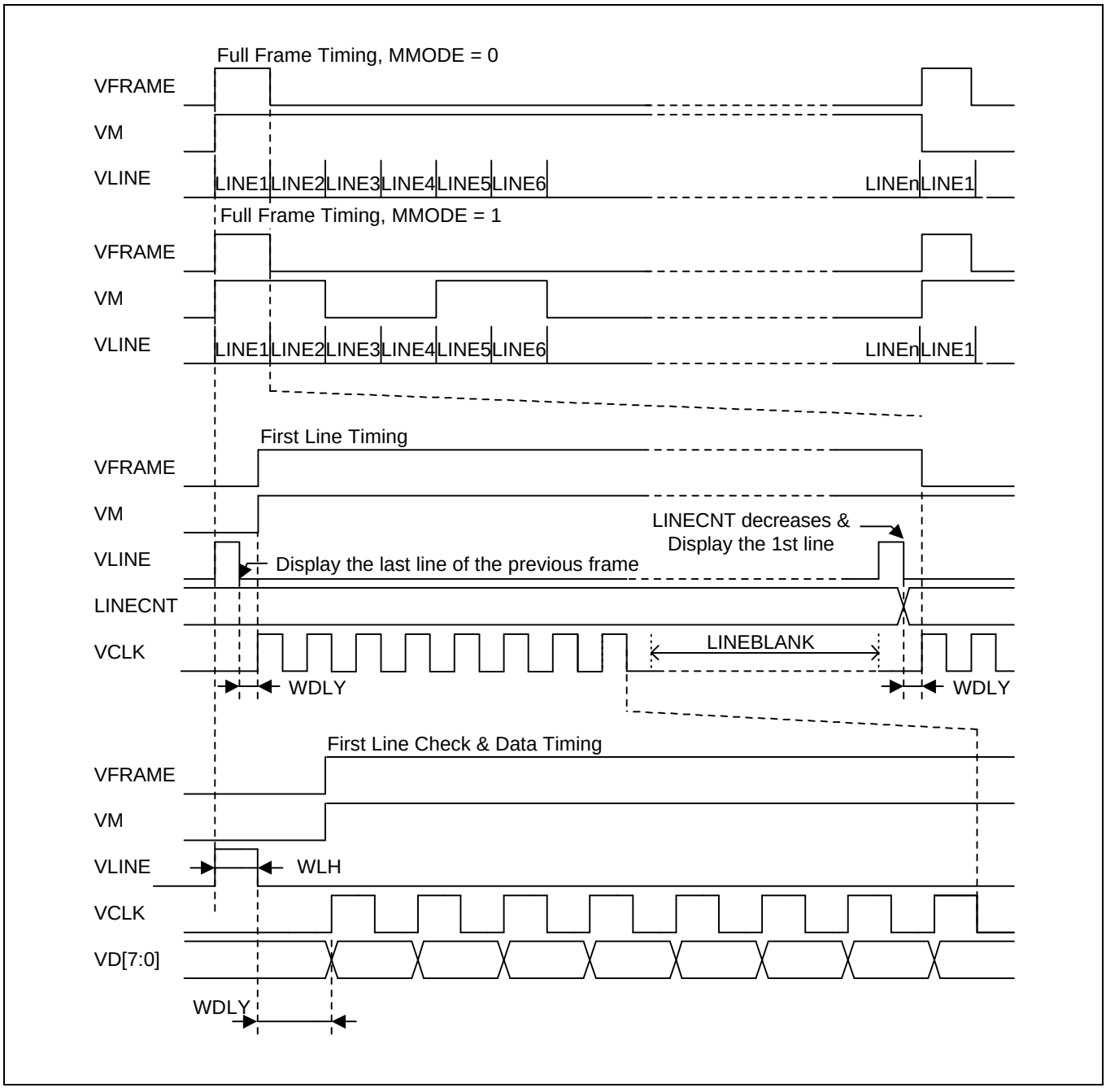


Figure 15-4. 8-bit Single Scan Display Type STN LCD Timing

LCD CONTROLLER OPERATION(TFT CASE)

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 configuration on the LCDCON1/2/3/4/5 registers in the REGBANK. Base on these programmable configuration on the LCD control registers in 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 is controlled by the configuration 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 can be controlled by the CLKVAL field in the LCDCON1 register. The table below defines the relationship of VCLK and CLKVAL. The minimum value of CLKVAL is 1.

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, CLKVAL in LCDCON1 and LCDCON2/3/4 registers. Most LCD driver need their own adequate frame rate. The frame rate is calculated as follows;

Frame Rate = 1/ [{ (VSPW+1) + (VBPD+1) + (LIINEVAL + 1) + (VFPD+1) } x {(HSPW+1) + (HBPD +1) + (HFPD+1) + (HOZVAL + 1) } x { 2 x (CLKVAL+1) / (HCLK) }]

Table 15-3. Relation between VCLK and CLKVAL(TFT, HCLK=60MHz)

CLKVAL	60MHz/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 S3C2400X supports 1, 2, 4 or 8 bpp(bit per pixel) palletized color displays and 16bpp non-palletized true-color displays.

256 Color Palette

The S3C2400X can support the 256 color palette for various selection of color mapping. This kind of selection can give the flexibility to users

MEMORY DATA FORMAT (TFT)

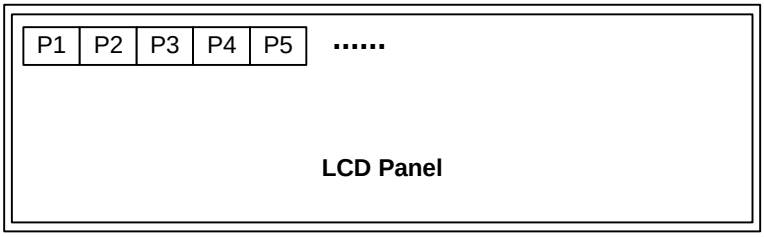
BPP16 Display

(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
...		



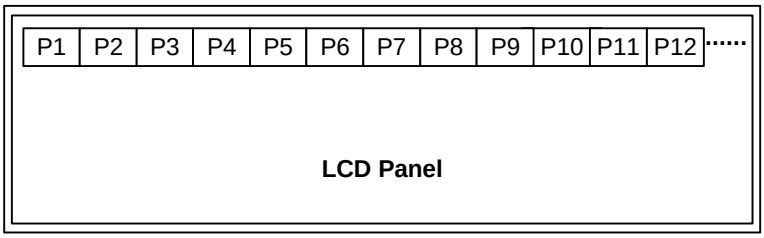
BPP8 Display

(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
...				



BPP4 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
...								

BPP2 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

S3C2400X provides 256 color palette for TFT LCD Control.
256 color palette consist of the 256(depth) × 16-bit SPSRAM.

Palette supports 5:6:5(R:G:B) format and 5:5:5:1(R:G:B:I) format. When the user use 5:5:5:1 format, the intensity data(I) is used as a common LSB bit of each RGB data. So, 5:5:5:I format is same as R(5+I):G(5+I):B(5+I) format.
For example, R(5+I)=VD[15:11]+VD[0], G(5+1)=VD[10:6]+VD[0], B(5+1)=VD[5:1]+VD[0].
The user can select 256 colors from the 64K colors through these two formats.

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	*0X14A00400
01H	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	0X14A00404
.....																	.....
FFH	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	0X14A007FC

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	0X14A00400
01H	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	I	0X14A00404
.....																	.....
FFH	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	I	0X14A007FC

NOTES:
1. * 0x14A00400 is Palette start address.
2. DATA[31:16] is invalid.

Palette Read/Write

When the user going to do Read/Write operation on the palette, VSTATUS of LCDCON5 register must be checked. Because Read/Write operation is prohibited during the ACTIVE status of VSTATUS.

Temp Configuration

S3C2400X supports that the user can fill a frame with one color without complex modification to fill the one color to the video 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.

Palette Offset Control

The index of Palette can be modified by the offset. This function provides special effect on LCD panel for the user.

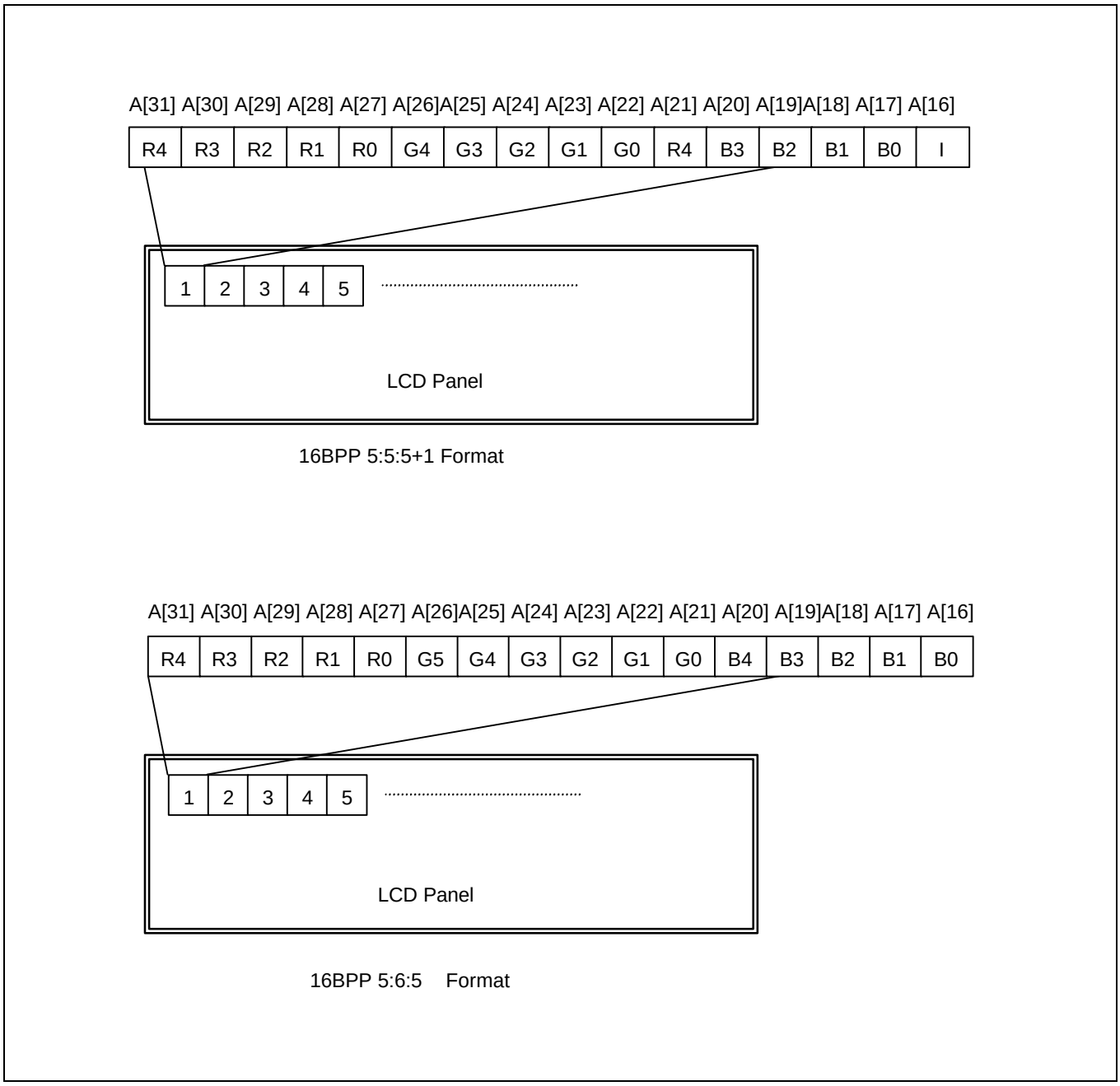


Figure 15-5. BPP16 5:6:5 Display Types (TFT)

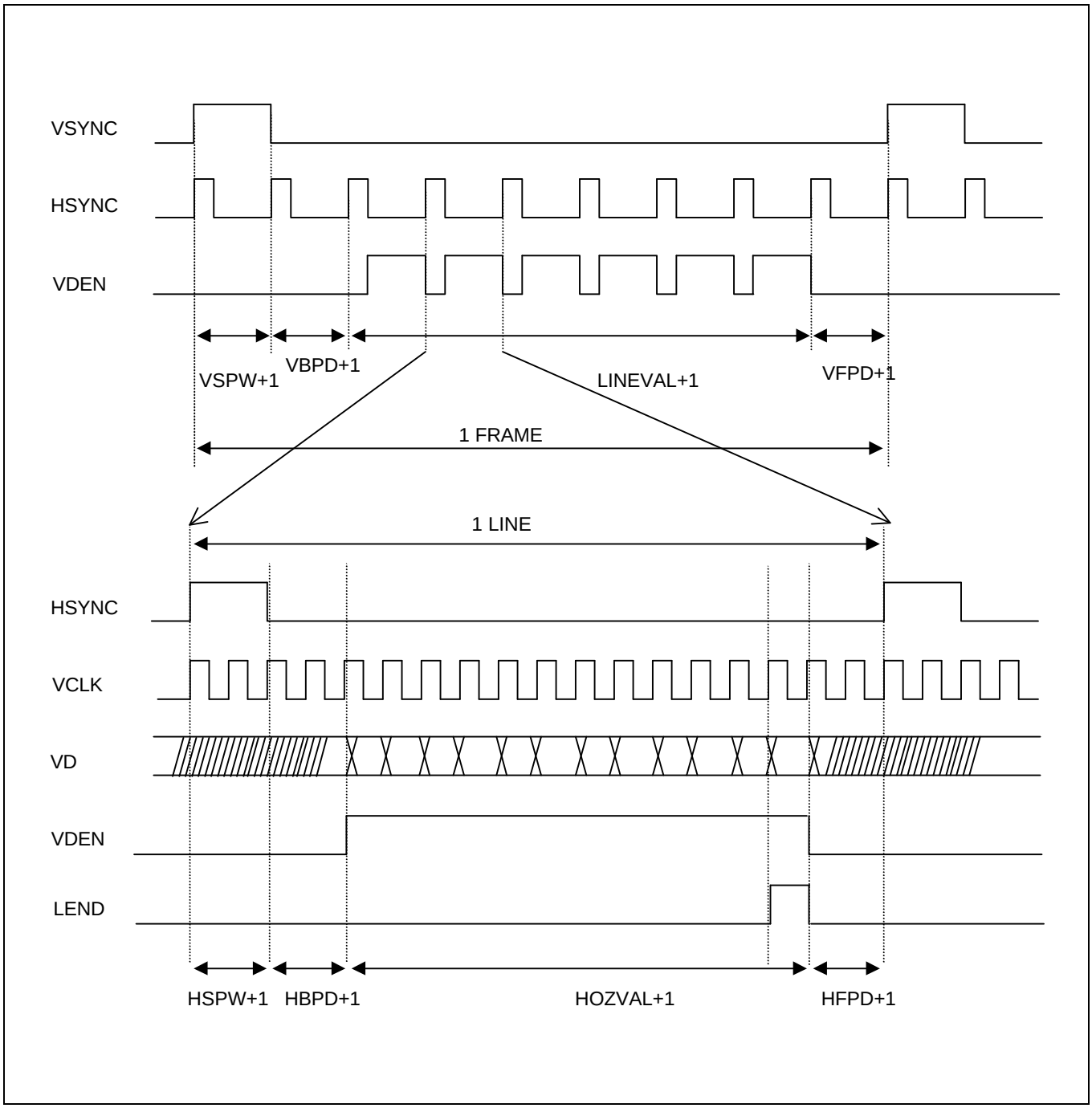


Figure 15-6. TFT LCD Timing Example

VIRTUAL DISPLAY (TFT/STN)

The S3C2400X 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(refer to Figure 15-8) but not the values of PAGEWIDTH and OFFSIZE.

The size of video buffer in which the image is stored should be larger than LCD panel screen size.

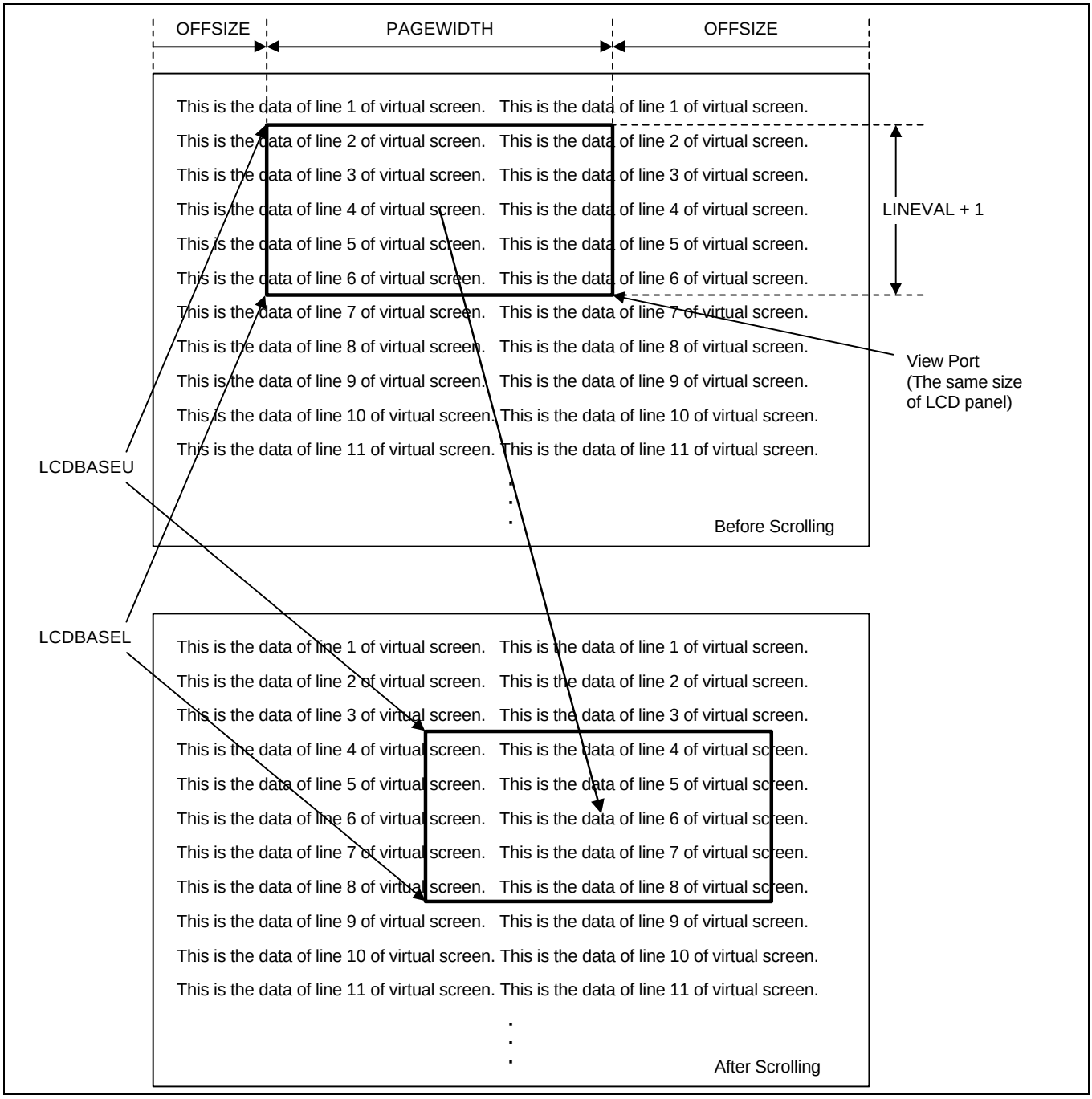


Figure 15-7. Example of Scrolling in Virtual Display (Single Scan)

LCD CONTROLLER SPECIAL REGISTERS

LCD Control 1 Register

Register	Address	R/W	Description	Reset Value
LCDCON1	0X14A00000	R/W	LCD control 1 register	0x00000000

LCDCON1	Bit	Description	Initial State
LINECNT (read only)	[27:18]	These bits provide the status of the line counter. Down count from LINEVAL to 0	0000000000
CLKVAL	[17:8]	These bits determine the rates of VCLK and CLKVAL[9:0]. **STN**: VCLK = HCLK / (CLKVAL x 2) (CLKVAL ≥ 2) **TFT**: VCLK = HCLK / [(CLKVAL+1) x 2] (CLKVAL ≥ 1)	0000000000
MMODE	[7]	This bit determines the toggle rate of the VM. 0 = Each Frame, 1 = The rate defined by the MVAL	0
PNRMODE	[6:5]	These bits 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]	These bits 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	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	0X14A00004	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	0X14A00008	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 = 64 HCLK, 11 = 128 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 2n bytes. If the x size of LCD is 120 dot in mono mode, x=120 can not be supported because 1 line is consist of 15 bytes. Instead, x=128 in mono mode can be supported because 1 line is consisted of 16 bytes (2n). 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 HCLKs.	

LCD Control 4 Register

Register	Address	R/W	Description	Reset Value
LCDCON4	0X14A0000C	R/W	LCD control 4 register	0x00000000

LCDCON4	Bit	Description	initial state
PALADDEN	[24]	**TFT**: Palette Index offset enable 0 = Disable 1 = Enable	0
ADDVAL	[23:16]	**TFT**: Palette Index offset value	0X00
MVAL	[15:8]	**STN**: These bit define the rate at which the VM signal will toggle if the MMODE bit is set 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 = 64 HCLK, 11 = 128 HCLK	

LCD Control 5 Register

Register	Address	R/W	Description	Reset Value
LCDCON5	0X14A00010	R/W	LCD control 5 register	0x00000000

LCDCON5	Bit	Description	initial state
VSTATUS	[20:19]	**TFT**: Vertical Status (Read only) 00 = VSYNC 01 = BACK Porch 10 = ACTIVE 11 = FRONT Porch	00
HSTATUS	[18:17]	**TFT**: Horizontal Status (Read only) 00 = HSYNC 01 = BACK Porch 10 = ACTIVE 11 = FRONT Porch	00
Reserved	[16]	This bit is reserved and the value should be '0'.	0
Reserved	[15]	This bit is reserved for Test mode and the value should be '0'.	0
SLOWCLKSYNC	[14]	**STN:** If SLOWCLKSYNC is 1, the SLOW mode will be entered from NORMAL mode synchronously when current LCD frame is completed. Also, if SLOWCLKSYNC is 1, the Normal mode will be entered from SLOW mode synchronously when current LCD frame is completed. This feature will be used for changing HCLK and SL_IDLE mode. 0 = Disable 1 = Enable	0
SELFREF	[13]	**STN**: LCD self refresh mode enable bit 0 = LCD self refresh mode disable 1 = LCD self refresh mode enable	0
Reserved	[12]		0
Reserved	[11]		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
Reserved	[5]		0
INVENDLINE	[4]	**TFT**: This bit indicates the LEND signal polarity. 0 = normal 1 = inverted	0
Reserved	[3]		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	0X14A00014	R/W	**STN/TFT**: Frame buffer start address 1 register	0x00000000

LCDSADDR1	Bit	Description	Initial State
LCDBANK	[27:21]	These bits indicate A[28:22] of the bank location for the video buffer in the system memory. LCDBANK value can not be changed even when moving the view port. LCD frame buffer should be inside aligned 4MB region, which ensures that LCDBANK value will not be changed when moving the view port. So, using the malloc() function the care should be taken.	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	0X14A00018	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 LCD controller is turned on. But, users must not change the LCDBASEU and LCDBASEL registers at the end of FRAME by referring to the LINECNT field in LCDCON1 register. Because of 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 the incorrect screen. To check the LINECNT, interrupt 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 ISR(interrupt service routine).

FRAME Buffer Start Address 3 Register

Register	Address	R/W	Description	Reset Value
LCDSADDR3	0X14A0001C	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 = 0xc500000
offset dot number = 2048 dots (512 half words)

LINEVAL = 240-1 = 0xef
PAGEWIDTH = 320*4/16 = 0x50
OFFSIZE = 512 = 0x200
LCDBANK = 0xc500000 >> 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 = 0xc500000
offset dot number = 2048 dots (512 half words)

LINEVAL = 120-1 = 0x77
PAGEWIDTH = 320*4/16 = 0x50
OFFSIZE = 512 = 0x200
LCDBANK = 0xc500000 >> 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 = 0xc500000
offset dot number = 1024 dots (512 half words)

LINEVAL = 240-1 = 0xef
PAGEWIDTH = 320*8/16 = 0xa0
OFFSIZE = 512 = 0x200
LCDBANK = 0xc500000 >> 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	0X14A00020	R/W	**STN**:Red lookup table register	0x00000000

REDLUT	Bit	Description	Initial State
REDVAL	[31:0]	These bits define which of the 16 shades each of the 8 possible red combinations will choose. 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	0X14A00024	R/W	**STN**:Green lookup table register	0x00000000

GREENLUT	Bit	Description	Initial State
GREENVAL	[31:0]	These bits define which of the 16 shades each of the 8 possible green combinations will choose. 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	0X14A00028	R/W	**STN**:Blue lookup table register	0x0000

BLUELUT	Bit	Description	Initial State
BLUEVAL	[15:0]	These bits define which of the 16 shades each of the 4 possible blue combinations will choose 00 = BLUEVAL[3:0], 01 = BLUEVAL[7:4] 10 = BLUEVAL[11:8], 11 = BLUEVAL[15:12]	0x0000

NOTE: Address from **0x14A0002C** to **0x14A00048** should not be used. This area is reserved for Test mode.

Dithering Mode Register

Register	Address	R/W	Description	Reset Value
DITHMODE	0X14A0004C	R/W	**STN**:Dithering Mode Register. This register reset value is 0x00000 But, user can change this value to 0x12210. (Please, 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	0X14A00050	R/W	**TFT**:Temporary Palette Register. This register value will be video data at next frame	0x00000000

TPAL	Bit	Description	initial state
TPALEN	[16]	Temporary Palette Register enable bit 0 = Disable 1 = Enable	0
TPALVAL	[15:0]	Temporary Palette Value Register. 5:6:5 format: TPALVAL[15:11] : RED TPALVAL[10:5] : GREEN TPALVAL[4:0] : BLUE 5:5:5:1 format: TPALVAL[15:11] : RED TPALVAL[10:6] : GREEN TPALVAL[5:1] : BLUE TPALVAL[1] : Intensity	0x000000

Register Setting Guide (STN)

The LCD controller supports multiple screen sizes by special register setting.

The CLKVAL value determines the frequency of VCLK. The data transmission rate for the VD port of the LCD controller should be calculated, in order to determine the value of CLKVAL register.

The data transmission rate is given by the following equation:

CLKVAL has to be determined, such that the VCLK value is greater than the data transmission rate.

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 is determined by 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 = 517KHz
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, Vertual screen size = 1024 x 1024, LCD size = 320 x 240, LCDBASEU = 0x64, 4-bit dual scan.

1 half-word = 8 pixels (4-level gray),
Virtual screen 1 line = 128 half-word = 1024 pixels,
LCD 1 line = 320 pixels = 40 half-word,
OFFSIZE = 128 - 40 = 88 = 0x58,
PAGEWIDTH = 40 = 0x28

LCDBASEL = LCDBASEU + (PAGEWIDTH + OFFSIZE) x (LINEVAL +1) = 100 + (40 +88) x 120 = 0x3C64

Gray Level Selection Guide

S3C2400XLCD controller can generate 16 gray level using FRC(frame rate control). The FRC characteristics may cause unexpected patterns in gray level. These unwanted erronous 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 the good gray levels after viewing all gray levels on user's own LCD.

Please select the gray level quality through the following procedures.

1. Get the latest dithering pattern register value from SAMSUNG.
2. Display 16gray 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 the good gray levels, which is displayed well on your LCD.
6. Use only the good gray levels for quality.

LCD Refresh Bus Bandwidth Calculation Guide

S3C2400XLCD controller can supports various LCD display size. To select suitable LCD display size(for the flicker free LCD system application), the user have to consider the LCD refresh bus bandwidth determined by 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 is the period of four-beat burst(4-words burst) for video data fetch. So, Pdma is determined by 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 x 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 = $8 \times 640 \times 480 \times 60 / 8 = 18.432$Mbyte/s
LCD DMA Burst Count = 18.432 / 16 = 1.152M/s
Pdma = (Trp+Trcd+CL+(2 × 4)+1) × (1/60MHz) = 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) } x {(HSPW+1) + (HBPD +1) + (HFPD+1) + (HOZVAL + 1) } x { 2 x (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 x 240,

VSPW =2 , VBPD =14, LINEVAL = 239, VFPD =4

HSPW =25, HBPD =15, HOZVAL = 239, HFPD =1

CLKVAL = 5

HCLK = 60 M (hz)

Below parameter must be referenced by LCD Size, and Driver specification:

VSPW, VBPD, LINEVAL, VFPD, HSPW, HBPD, HOZVAL, HFPD

If target frame rate is 60–70Hz then CLKVAL should be 5.

So, Frame Rate = 67Hz

NOTES

16 A/D CONVERTER

OVERVIEW

The 10-bit CMOS ADC (Analog to Digital Converter) of S3C2400X is a recycling type 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 2.5MHz A/D converter clock. A/D converter operates with on-chip sample-and-hold function and power down mode is supported.

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

A/D CONVERTER OPERATION

BLOCK DIAGRAM

Figure 16-1 shows the functional block diagram of S3C2400X A/D converter. Note that the A/D converter device is a recycling type.

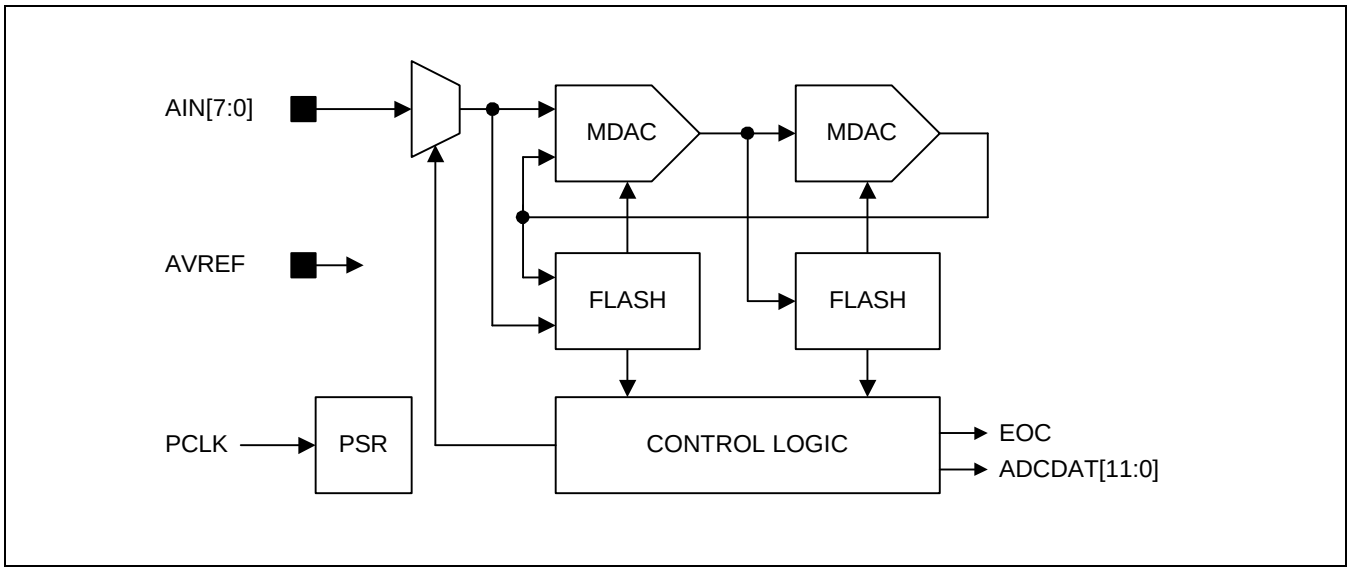


Figure 16-1. A/D Converter Functional Block Diagram

FUNCTION DESCRIPTIONS

A/D Conversion Time

When the PCLK frequency is 50MHz and the prescaler value is 49, total 10-bit conversion time is as follows.

A/D converter freq. = 50MHz/(49+1) = 1MHz

Conversion time = 1/(1MHz / 5cycles) = 1/200KHz = 5 us

NOTE: This A/D converter was designed to operate at maximum 2.5MHz clock, so the conversion rate can go up to 500 KSPS.

Standby Mode

Standby mode is activated when ADCCON[2] is set to '1'. In this mode, A/D conversion operation is halted and ADCDAT register contains 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. Another way for starting A/D conversion is provided. 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 CONVERTER SPECIAL REGISTERS

A/D CONVERTER CONTROL REGISTER (ADCCON)

Register	Address	R/W	Description	Reset Value
ADCCON	0x15800000	R/W	A/D Converter control Register	0x3FC4

ADCCON	Bit	Description	Initial State
ECFLG	[15]	Reserved	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 prescaler value is N.	0xFF
INPUT SELECT	[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	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 writing ‘1’. Write) 0 = No operation 1 = A/D conversion starts Read) 0 = A/D conversion is completed. 1 = A/D conversion is being processed.	0

A/D CONVERTER DATA REGISTER (ADCDAT)

Register	Address	R/W	Description	Reset Value
ADCDAT	0x15800004	R	A/D converter data register	–

ADCDAT	Bit	Description	Initial State
ADCDAT	[9:0]	A/D converter data value Data value: 0 – 3FF Note that the converted data is loaded automatically after ADCCON[15] goes to 1.	–

NOTES

17 RTC (REAL TIME CLOCK)

OVERVIEW

The RTC (Real Time Clock) unit can be operated by the backup battery while the system power is off. The RTC can transmit 8-bit data to CPU as BCD (Binary Coded Decimal) values using the STRB/LDRB ARM operation. The data include 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.

FEATURE

— BCD number: second, minute, hour, date, day, month, year

— Leap year generator

— Alarm function: alarm interrupt or wake-up from power down 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

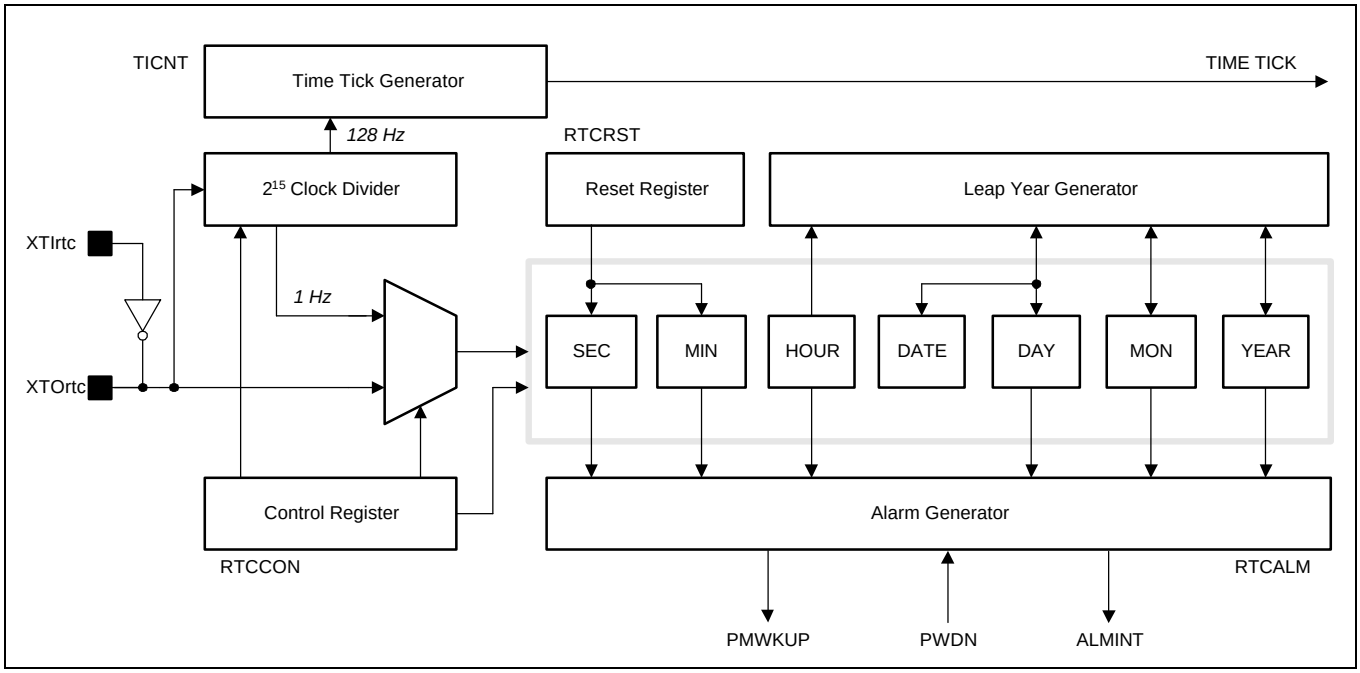


Figure 17-1. Real Time Clock Block Diagram

LEAP YEAR GENERATOR

This block can determine whether the last date of each month is 28, 29, 30, or 31, based on data from BCDDAY, BCDMON, and BCDYEAR. This block considers the 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 is a leap year or not. For example, it can not discriminate between 1900 and 2000. To solve this problem, the RTC block in S3C2400X has hard-wired logic to support the leap year in 2000. Please note 1900 is not leap year while 2000 is leap year. Therefore, two digits of 00 in S3C2400X denote 2000, not 1900.

READ/WRITE REGISTERS

Bit 0 of the RTCCON register must be set to high in order to write the BCD register in RTC block. To display the sec., min., 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 result is a value from 1 to 59(Second), there is no problem, but, if the result 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, 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 RTC block, even if the system power is off. When the system 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 down mode or normal operation mode. In normal operation mode, the alarm interrupt (ALMINT) is activated. In the power down mode the power management wakeup (PMWKUP) signal is activated as well as the ALMINT. The RTC alarm register, RTCALM, determines the alarm enable/disable 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 follow:

Period = (n+1) / 128 second
n: Tick time count value (1–127)

This RTC time tick may be used for RTOS(real time operating system) kernel time tick. If time tick is generated by RTC time tick, the time related function of RTOS will always synchronized with 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 by the byte unit using the STRB,LDRB instructions or char type pointer.

32.768 kHz X-TAL CONNECTION EXAMPLE

The Figure 17-2 is an example circuit of the RTC unit oscillation at 32.768 kHz.

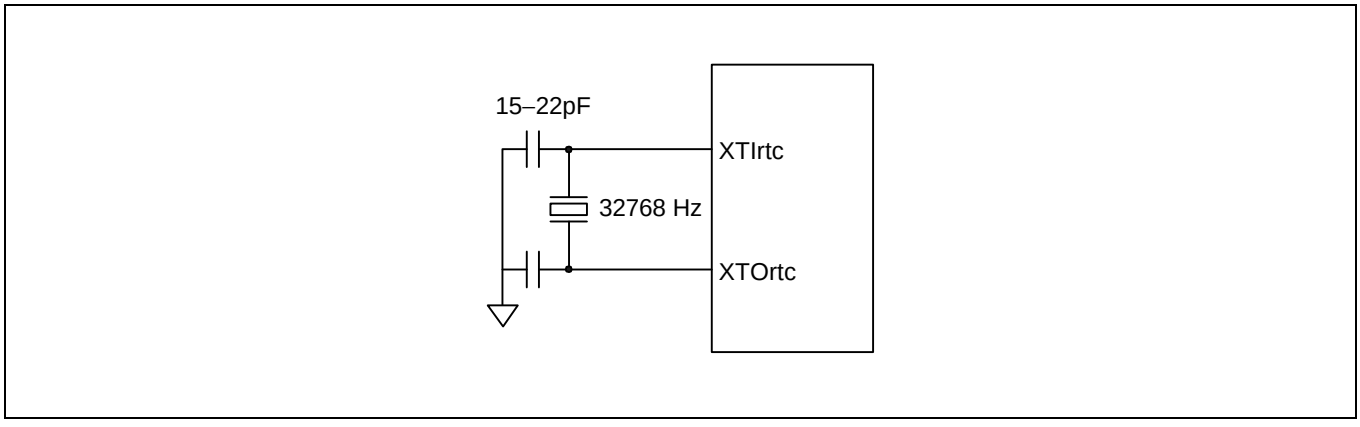


Figure 17-2. Main Oscillator Circuit Examples

REAL TIME CLOCK SPECIAL REGISTERS

REAL TIME CLOCK CONTROL REGISTER (RTCCON)

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	0x15700040(L) 0x15700043(B)	R/W (by byte)	RTC control Register	0x0

RTCCON	Bit	Description	Initial State
RTCEN	[0]	RTC control enable 0 = Disable 1 = Enable **NOTE:** Only BCD time count and read operation can be performed.	0
CLKSEL	[1]	BCD clock select 0 = XTAL $1/2^{15}$ divided clock 1 = Reserved (XTAL clock only for test)	0
CNTSEL	[2]	BCD count select 0 = Merge BCD counters 1 = Reserved (Separate BCD counters)	0
CLKRST	[3]	RTC clock count reset 0 = No reset, 1 = Reset	0

NOTES:

1. All RTC registers have to be accessed by byte unit using STRB and LDRB instructions or char type pointer.
2. (L): When the endian mode is little endian.
   (B): When the endian mode is Big endian.

TICK TIME COUNT REGISTER (TICNT)

Register	Address	R/W	Description	Reset Value
TICNT	0x15700044(L) 0x15700047(B)	R/W (by byte)	Tick time count Register	0x00000000

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 can not read this real counter value in working.	000000

RTC ALARM CONTROL REGISTER (RTCALM)

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	0x15700050(L) 0x15700053(B)	R/W (by byte)	RTC alarm control Register	0x00

RTCALM	Bit	Description	Initial State
Reserved	[7]	Reserved to 0.	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
DAYEN	[3]	Day 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 REGISTER (ALMSEC)

Register	Address	R/W	Description	Reset Value
ALMSEC	0x15700054(L) 0x15700057(B)	R/W (by byte)	Alarm second data Register	0x00

ALMSEC	Bit	Description	Initial State
Reserved	[7]		0
SECDATA	[6:4]	BCD value for alarm second from 0 to 5	000
	[3:0]	from 0 to 9	0000

ALARM MIN DATA REGISTER (ALMMIN)

Register	Address	R/W	Description	Reset Value
ALMMIN	0x15700058(L) 0x1570005B(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 from 0 to 5	000
	[3:0]	from 0 to 9	0000

ALARM HOUR DATA REGISTER (ALMHOUR)

Register	Address	R/W	Description	Reset Value
ALMHOUR	0x1570005C(L) 0x1570005F(B)	R/W (by byte)	Alarm hour data Register	0x00

ALMHOUR	Bit	Description	Initial State
Reserved	[7:6]		0
HOURDATA	[5:4]	BCD value for alarm hour from 0 to 2	00
	[3:0]	from 0 to 9	0000

ALARM DAY DATA REGISTER (ALMDAY)

Register	Address	R/W	Description	Reset Value
ALMDAY	0x15700060(L) 0x15700063(B)	R/W (by byte)	Alarm day data Register	0x01

ALMDAY	Bit	Description	Initial State
Reserved	[7:6]		0
DAYDATA	[5:4]	BCD value for alarm day, from 0 to 28, 29, 30, 31 from 0 to 3	00
	[3:0]	from 0 to 9	0001

ALARM MON DATA REGISTER (ALMMON)

Register	Address	R/W	Description	Reset Value
ALMMON	0x15700064(L) 0x15700067(B)	R/W (by byte)	Alarm month data Register	0x01

ALMMON	Bit	Description	Initial State
Reserved	[7:5]		0
MONDATA	[4]	BCD value for alarm month from 0 to 1	0
	[3:0]	from 0 to 9	0001

ALARM YEAR DATA REGISTER (ALMYEAR)

Register	Address	R/W	Description	Reset Value
ALMYEAR	0x15700068(L) 0x1570006B(B)	R/W (by byte)	Alarm year data Register	0x00

ALMYEAR	Bit	Description	Initial State
YEARDATA	[7:0]	BCD value for year from 00 to 99	0x00

RTC ROUND RESET REGISTER (RTCRST)

Register	Address	R/W	Description	Reset Value
RTCRST	0x1570006C(L) 0x1570006F(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(0,1,2,6,7) are set, no second carry is generated. But second value can be reset.	00

BCD SECOND REGISTER (BCDSEC)

Register	Address	R/W	Description	Reset Value
BCDSEC	0x15700070(L) 0x15700073(B)	R/W (by byte)	BCD second Register	Undef.

BCDSEC	Bit	Description	Initial State
SECDATA	[6:4]	BCD value for second from 0 to 5	–
	[3:0]	from 0 to 9	–

BCD MINUTE REGISTER (BCDMIN)

Register	Address	R/W	Description	Reset Value
BCDMIN	0x15700074(L) 0x15700077(B)	R/W (by byte)	BCD minute Register	Undef.

BCDMIN	Bit	Description	Initial State
MINDATA	[6:4]	BCD value for minute from 0 to 5	–
	[3:0]	from 0 to 9	–

BCD HOUR REGISTER (BCDHOUR)

Register	Address	R/W	Description	Reset Value
BCDHOUR	0x15700078(L) 0x1570007B(B)	R/W (by byte)	BCD hour Register	Undef.

BCDHOUR	Bit	Description	Initial State
Reserved	[7:6]		–
HOURDATA	[5:4]	BCD value for hour from 0 to 2	–
	[3:0]	from 0 to 9	–

BCD DAY REGISTER (BCDDAY)

Register	Address	R/W	Description	Reset Value
BCDDAY	0x1570007C(L) 0x1570007F(B)	R/W (by byte)	BCD day Register	Undef

BCDDAY	Bit	Description	Initial State
Reserved	[7:6]		–
DAYDATA	[5:4]	BCD value for day from 0 to 3	–
	[3:0]	from 0 to 9	–

BCD DATE REGISTER (BCDDATE)

Register	Address	R/W	Description	Reset Value
BCDDATE	0x15700080(L) 0x15700083(B)	R/W (by byte)	BCD date Register	Undef.

BCDDATE	Bit	Description	Initial State
Reserved	[7:3]		–
DATEDATA	[2:0]	BCD value for date from 1 to 7	–

BCD MONTH REGISTER (BCDMON)

Register	Address	R/W	Description	Reset Value
BCDMON	0x15700084(L) 0x15700087(B)	R/W (by byte)	BCD month Register	Undef.

BCDMON	Bit	Description	Initial State
Reserved	[7:5]		–
MONDATA	[4]	BCD value for month from 0 to 1	–
	[3:0]	from 0 to 9	–

BCD YEAR REGISTER (BCDYEAR)

Register	Address	R/W	Description	Reset Value
BCDYEAR	0x15700088(L) 0x1570008B(B)	R/W (by byte)	BCD year Register	Undef.

BCDYEAR	Bit	Description	Initial State
YEARDATA	[7:0]	BCD value for year from 00 to 99	–

18 WATCHDOG TIMER

OVERVIEW

The S3C2400X watchdog timer is used to resume the controller operation when it had been 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

The functional block diagram of the watchdog timer is shown in Figure 18-1. The watchdog timer uses PCLK as its only source clock. To generate the corresponding watchdog timer clock, the PCLK frequency is prescaled first, and the resulting frequency is divided again.

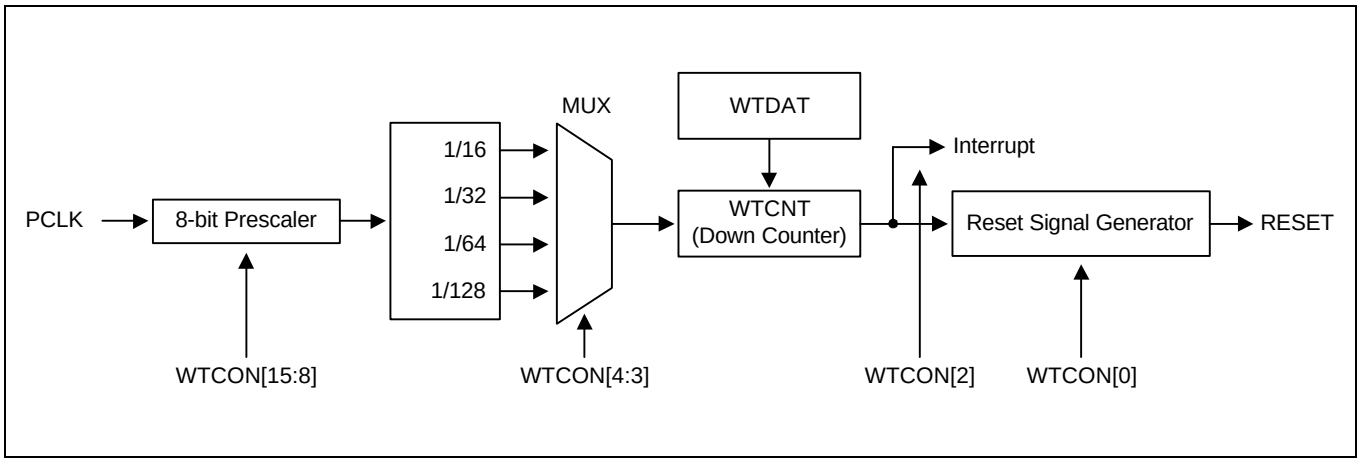


Figure 18-1. Watchdog Timer Block Diagram

The prescaler value and the frequency division factor are specified in the watchdog timer control register, WTCON. The 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

When the watchdog timer is enabled first, the value of WTDAT (watchdog timer data register) cannot be automatically reloaded into the WTCNT (timer counter). For this reason, an initial value must be written to the watchdog timer count register, WTCNT, before the watchdog timer starts.

CONSIDERATION OF DEBUGGING ENVIRONMENT

When S3C2400X is in debug mode using Embedded ICE, the watchdog timer must not operate.

The watchdog timer can determine whether or not the current mode is 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 when the watchdog timer is expired.

WATCHDOG TIMER SPECIAL REGISTERS

WATCHDOG TIMER CONTROL REGISTER (WTCON)

Using the Watchdog Timer Control register, WTCON, you can 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 S3C2400X restart on mal-function after 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, please enable the interrupt and disable the Watchdog timer.

Register	Address	R/W	Description	Reset Value
WTCON	0x15300000	R/W	Watchdog timer control Register	0x8021

WTCON	Bit	Description	Initial State
Prescaler value	[15:8]	the 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 enable/disable	[5]	Enable or disable bit of Watchdog timer. 0 = Disable Watchdog timer 1 = Enable Watchdog timer	1
Clock select	[4:3]	This two bits determines the clock division factor. 00: 16 01: 32 10: 64 11: 128	00
Interrupt enable/disable	[2]	Enable or disable bit of the interrupt. 0 = Disable interrupt generation 1 = Enable interrupt generation	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: Asserts reset signal of the S3C2400X at watchdog time-out 0: Disables the reset function of the watchdog timer.	1

WATCHDOG TIMER DATA REGISTER (WTDAT)

The watchdog timer data register, WTDAT is used to specify the time-out duration. The content of WTDAT can not be automatically loaded into the timer counter at initial watchdog timer operation. However, the first time-out occurs by using 0x8000(initial value), after then the value of WTDAT will be automatically reloaded into WTCNT.

Register	Address	R/W	Description	Reset Value
WTDAT	0x15300004	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 REGISTER (WTCNT)

The watchdog timer count register, WTCNT, contains the current count values for the watchdog timer during normal operation. Note that the content of the watchdog timer data register cannot be automatically loaded into the timer count register when the watchdog timer is enabled initially, so the watchdog timer count register must be set to an initial value before enabling it.

Register	Address	R/W	Description	Reset Value
WTCNT	0x15300008	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 INTERFACE

OVERVIEW

The S3C2400X MultiMediaCard (MMC) interface can interface with MMC type (referenced MMCA ver 2.11) serial data. There are four tokens in the MMC type serial data; transmit command, receive response, transmit data, receive data. MMC module needs to transfer these four tokens respectively. We will design MMC interface with four part division as Block Diagram. If MMC module transmits or receives data, each FIFO should always operate. However, if MMC module transmits command or receives response, neither FIFO nor DMA mode are not supplied.

FEATURES

— MultiMediaCard protocol (ver 2.11) compatible
— 16 bytes FIFO (depth 16) for the data transmission
— 16 bytes FIFO (depth 16) for the data receiving.
— 40-bit Command Register
— 128-bit Response Register
— 8-bit Prescaler logic (Freq. = System Clock / (2(P + 1)))
— CRC7 & CRC16 generator
— Data Fault Error detection logic
— Normal, Interrupt, and DMA transfer mode

BLOCK DIAGRAM

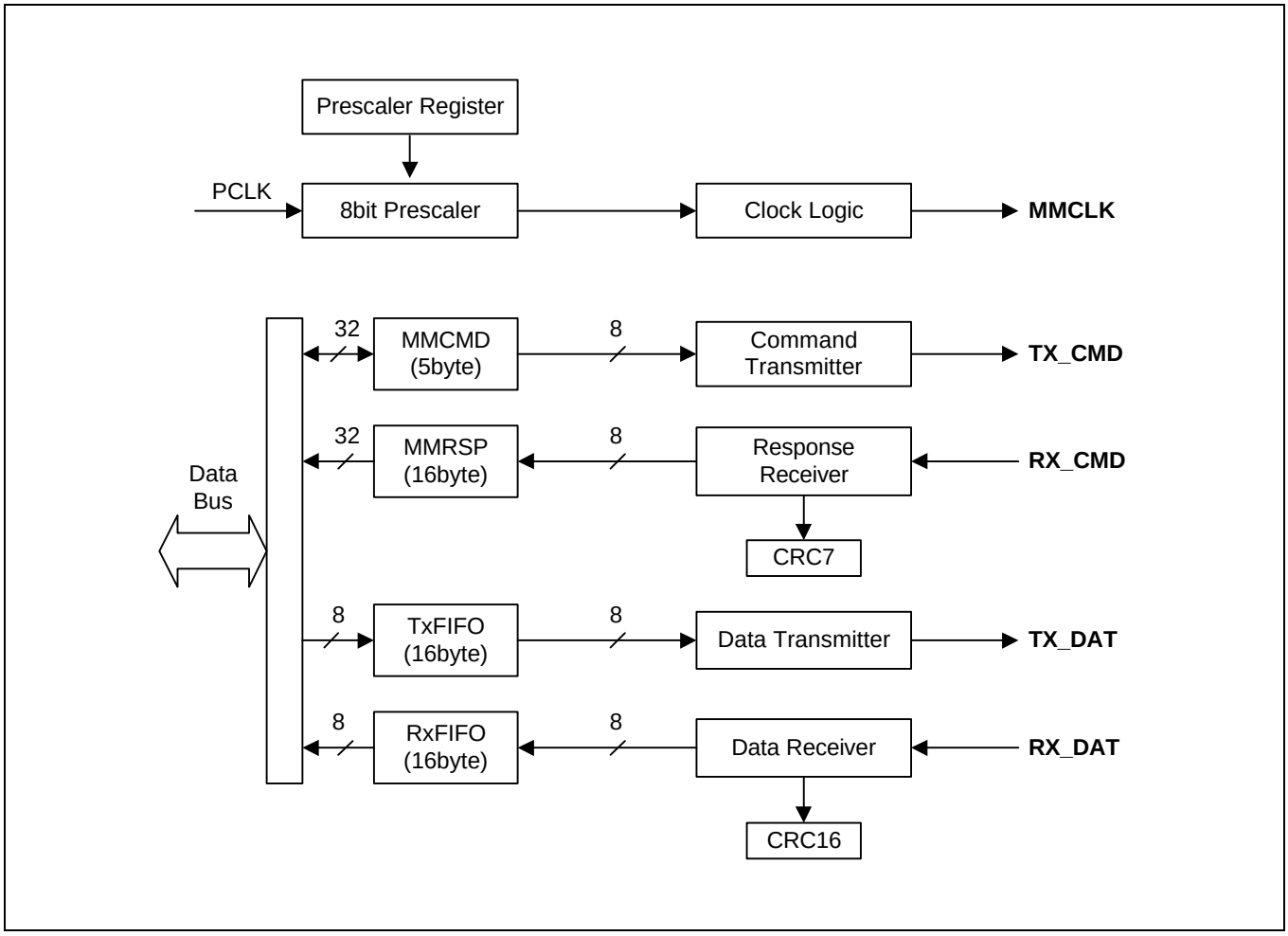


Figure 19-1. MMC Block Diagram

MMC OPERATION

A serial clock line synchronizes shifting and sampling of the information on the two serial data lines. The transmission frequency is controlled by making the appropriate bit settings to the MMPRE register. You can modify its frequency to adjust the baud rate data register value.

PROGRAMMING PROCEDURE

To program the MMC modules, follow these basic steps :

1. Set Baud Rate Prescaler Register(MMPRE).
2. Set MMCON, and MMFCON to configure properly the MMC module and FIFO, and Block Length Register(MMLEN).
3. Wait 74 MMCLK clock cycle in order to initialize the card.
4. Write command 5byte to MMCMD
5. Start command transmit and determine command types with setting MMCRR register.
6. Tx data → Write data to Data Register(MMDAT) while Tx FIFO is not beyond trigger level.
7. Rx data → Data receiver starts operation when MMC module detects start bit in DAT pin. User can read data from Rx FIFO when Rx FIFO reaches trigger level or Rx FIFO has the last data of a block.
8. Confirm the end of MMC command operation when NCFIN(Normal Command Finish) flag, or DCFIN(Data Command Finish) flag of MMSTA register is set
9. Clear the corresponding flag(NCFIN, or DCFIN) of MMSTA through writing one with this bit.

STEPS FOR TRANSMIT BY DMA

1. DMA is configured properly and the MMC is configured as DMA start with MMC Mode Select bits.
2. If Tx FIFO is beyond trigger level, go to step 4.
3. The MMC requests DMA service, and then DMA transmits 1byte data to MMC module until FIFO is full.
4. The MMC transmits the data to card.
5. Go to step 2 until DMA count is 0.

STEPS FOR RECEIVE BY DMA

1. DMA is configured properly and the MMC is configured as DMA start with MMC Mode Select bits.
2. The MMC receives the data from card.
3. If Rx FIFO neither reaches trigger level nor has the last data of a block, go to step 2.
4. The MMC requests DMA service, and then DMA receives 1byte data from MMC module until FIFO is empty.
5. Go to step 2 until DMA count is 0.

MMC SPECIAL REGISTERS

MMC CONTROL REGISTER (MMCON)

Register	Address	R/W	Description	Reset Value
MMCON	0x15A00000(Li/B, Li/HW, Li/W, Bi/W) 0x15A00002(Bi/HW) 0x15A00003(Bi/B)	R/W	MMC Control Register	0x00

MMCON	Bit	Description	Initial State
Reserved	[7:6]		
Exception Interrupt Enable (EINTE)	[5]	Determines MMC generate an interrupt if an exception(CRC error etc) or transfer completion occur. 0 = disable, 1 = interrupt enable	0
MMC Mode Select (MMOD)	[4:3]	Determines how the MMC interface is read/written. 00 = polling mode, 01 = interrupt mode 10 = DMA transmit mode, 11 = DMA receive mode	00
Prescaler Enable (ENPRE)	[2]	Determines what you want prescaler enable or not. 0 = counter reset, 1 = enable prescaler	0
Clock Polarity Select (CPOL)	[1]	Determines an active high or active low clock. 0 = active high, 1 = active low	0
Clock Pending mode(PEND)	[0]	Determines clock pends when APB is not ready to transfer data. 0 = disable, 1 = clock pending mode enable	0

NOTES:

1. When Clock Pending mode state is disable, if receive data collision error occurs and invalid data transmit error occurs, MMSTA[2], MMSTA[3] is set.
2. (Li/B/HW/W): Access by byte/half-word/word unit when the endian mode is Little.
   (Bi/B/HW/W): Access by byte/half-word/word unit when the endian mode is Big.

MMC COMMAND RELATED REGISTER (MMCRR)

Register	Address	R/W	Description	Reset Value
MMCRR	0x15A00004(Li/B, Li/HW, Li/W, Bi/W) 0x15A00006(Bi/HW) 0x15A00007(Bi/B)	R/W	MMC Command Related Register	0x00

MMCRR	Bit	Description	Initial State
Response Type (RTYP)	[7:6]	Determines what response type MMC command is. 00 = no response, 01 = short type (5 byte) 10 = short & busy type (5 byte), 11 = long type (16 byte)	00
Multi Block Mode (MULTI)	[5]	Determines whether MMC command type is MULTI or not. 0 = normal mode, 1 = multi mode	0
Sequential Mode (SEQ)	[4]	Determines whether MMC command type is SEQ or not. 0 = normal mode, 1 = sequential mode	0
Data Stop Command (STOP)	[3]	Determines whether MMC command function is STOP or not. 0 = not stop command, 1 = stop command	0
Data Read Command (RCMD)	[2]	Determines whether MMC command function is READ or not. 0 = not read command, 1 = read command	0
Data Write Command (WCMD)	[1]	Determines whether MMC command function is WRITE or not. 0 = not write command, 1 = write command	0
Command Start (CMST)	[0]	Determines whether MMC command operation starts or not. 0 = command ready, 1 = command start	0

NOTE: When you are going to operate STOP command, confirm transmit(or receive) all the data to send (or get). And then, you can execute STOP command. But, in the case of sequential(SEQ) write(WCMD) command, you should but also watch Tx FIFO's count number whether zero or not.

MMCRR REGISTER SETTING ACCORDING TO MULTI MEDIA CARD CMD

CMD No.	Operation	RTYP	MULTI	SEQ	STOP	RCMD	WCMD	Finish Flag
CMD0	Go Idle	0	0	0	0	0	0	Normal
CMD2	All Send CID	3	0	0	0	0	0	Normal
CMD4	Set DSR	0	0	0	0	0	0	Normal
CMD7	Card select	1	0	0	0	0	0	Normal
CMD7	Card deselect	0	0	0	0	0	0	Normal
CMD9	Send CSD	3	0	0	0	0	0	Normal
CMD10	Send CID	3	0	0	0	0	0	Normal
CMD11	Read Seq Data	1	0	set	0	set	0	Data
CMD12	Stop Transfer	1	0	0	set	0	0	Normal
CMD15	Go Inactive State	0	0	0	0	0	0	Normal
CMD17	Read Single Block	1	0	0	0	set	0	Data
CMD18	Read Multi Block	1	set	0	0	set	0	Data
CMD20	Write Seq Data	1	0	set	0	0	set	Data
CMD24	Write Single Block	1	0	0	0	0	set	Data
CMD25	Write Multi Block	1	set	0	0	0	set	Data
CMD26	Program CID	1	0	0	0	0	set	Data
CMD27	Program CSD	1	0	0	0	0	Set	Data
CMD28	Set Write Protect	2	0	0	0	0	0	Normal
CMD29	Clr Write Protect	2	0	0	0	0	0	Normal
CMD30	Send Write Protect	1	0	0	0	set	0	Data
CMD38	Erase	2	0	0	0	0	0	Normal
other	The Other CMD	1	0	0	0	0	0	Normal

NOTE: If you transmit CMD2 several times moer than the number of cards attatched, you must add to transmit dummy data through CMD bus. You can know the number of cards attatched when NCFIN flag wll not set although 189 MMCCLK cycles running(during the minimum length loop of 160 times) after starting CMD2. Typical value of dummy data are 0x00, 0x00, 0x00, 0x00, 0x00(5bytes). You put these value in the MMCMD0, MMCMD1 register, and then set MMCRR register to 0x01.

MMC FIFO CONTROL REGISTER (MMFCON)

Register	Address	R/W	Description	Reset Value
MMFCON	0x15A00008(Li/B, Li/HW, Li/W, Bi/W) 0x15A0000A(Bi/HW) 0x15A0000B(Bi/B)	R/W	MMC FIFO Control Register	0x00

MMFCON	Bit	Description	Initial State
Reserved	[7:6]		
Tx FIFO Trigger level (TFTRIG)	[5:4]	Determines trigger level of transmit FIFO 00 = empty, 01 = 4 byte 10 = 8 byte 11 = 12 byte	00
Rx FIFO Trigger level (RFTRIG)	[3:2]	Determines trigger level of receive FIFO. 00 = 4 byte, 01 = 8 byte 10 = 12 byte 11 = 16 byte	00
Tx FIFO Reset (TFRST)	[1]	Determines resetting for transmit FIFO. 0 = normal mode, 1 = Tx FIFO reset	0
Rx FIFO Reset (RFRST)	[0]	Determines resetting for receive FIFO. 0 = normal mode, 1 = Rx FIFO reset	0

NOTE: You must always reset Tx FIFO or Rx FIFO, after completion MULTI command operation and SEQ command operation. FIFO may have some of dummy data in these commands.

MMC STATUS REGISTER (MMSTA)

Register	Address	R/W	Description	Reset Value
MMSTA	0x15A0000C(Li/B, Li/HW, Li/W, Bi/W) 0x15A0000E(Bi/HW) 0x15A0000F(Bi/B)	R/(C)	MMC Status Register	0x00

MMSTA	Bit	Description	Initial State
Reserved	[7]		
Response CRC Mismatch Error (RCRC)	[6]	This flag is set when response token value is wrong. This flag is cleared by reading the MMSTA with RCRC set. 0 = not occur, 1 = CRC error occur	0
Data Read CRC Mismatch Error (DCRC)	[5]	This flag is set when data reading token value is wrong. This flag is cleared by reading the MMSTA with DCRC set. 0 = not occur, 1 = CRC error occur	0
CRC Status Error (CRCS)	[4]	This flag is set when data writing token is wrong. This flag is cleared by reading the MMSTA with CRCS set. 0 = not occur, 1 = CRC status error occur	0
Invalid Data Transmit Error (INVL)	[3]	This flag is set when invalid data should be transmitted because there is no data in the transmit FIFO. This flag is cleared by reading the MMSTA with INVL set. 0 = not detect, 1 = invalid data Tx error detect	0
Receive Data Collision Error (RCOL)	[2]	This flag is set when data collision occurs in the receive FIFO. This flag is cleared by reading the MMSTA with RCOL set. 0 = not detect, 1 = collision error detect	0
Data Command Finish (DCFIN)	[1] R/C	This bit indicates that data transfer related command operation completes. This flag is cleared by setting to one this bit. 0 = not finish, 1 = data command finish	
Normal Command Finish (NCFIN)	[0] R/C	This bit indicates that normal command operation completes. This flag is cleared by setting to one this bit. 0 = not finish, 1 = normal command finish	0

NOTES:

1. DCFIN bit is set after the data transfer
2. In case of the command with the response , NCFIN bit is set after the response.
   In case of the command without the response, NCFIN bit is set after the command transfer.

MMC FIFO STATUS REGISTER (MMFSTA)

Register	Address	R/W	Description	Reset Value
MMFSTA	0x15A00010(Li/HW, Li/W, Bi/W) 0x15A00012(Bi/HW)	R	MMC FIFO Status Register	0x0000

MMFSTA	Bit	Description	Initial State
Reserved	[15:12]		
Tx FIFO Count (TFCNT)	[11:8]	Number of data in transmit FIFO	0000
Rx FIFO Count (RFCNT)	[7:4]	Number of data in receive FIFO	0000
Tx FIFO Full (TFFULL)	[3]	This bit automatically set to 1 whenever transmit FIFO is full. 0 = 0 = Tx FIFO = 15, 1 = Full	0
Rx FIFO Full (RFFULL)	[2]	This bit automatically set to 1 whenever receive FIFO is full. 0 = 0 = Rx FIFO = 15, 1 = Full	0
Tx FIFO Trigger level Detect (TFDET)	[1]	This bit indicates that Tx FIFO reaches MMFCON's trigger level. 0 = not detect, 1 = detect	0
Rx FIFO Trigger level Detect (RFDET)	[0]	This bit indicates that Rx FIFO reaches MMFCON's trigger level or Rx FIFO has the last data of a block. 0 = not detect, 1 = detect	0

MMC BAUD RATE PRESCALER REGISTER (MMPRE)

Register	Address	R/W	Description	Reset Value
MMPRE	0x15A00014 (Li/B, Li/HW, Li/W, Bi/W) 0x15A00016 (Bi/HW) 0x15A00017 (Bi/B)	R/W	MMC Baud Rate Prescaler Register	0x00

MMPRE	Bit	Description	Initial State
Prescaler Value	[7:0]	Determines MMC clock rate as below equation. Baud rate = PCLK / 2 / (Prescaler value + 1)	0x00

MMC BLOCK LENGTH REGISTER (MMLEN)

Register	Address	R/W	Description	Reset Value
MMLEN	0x15A00018(Li/HW, Li/W, Bi/W) 0x15A0001A(Bi/HW)	R/W	MMC Block Length Register	0x0000

MMLEN	Bit	Description	Initial State
Block Length Value	[15:0]	Determines the block size to transfer data	0x0000

RESPONSE CRC7 REGISTER (MMCR7)

Register	Address	R/W	Description	Reset Value
MMCR7	0x15A0001C(Li/B, Li/HW, Li/W, Bi/W) 0x15A0001E(Bi/HW) 0x15A0001F(Bi/B)	R	Response CRC7 Register	0x00

RCRC7	Bit	Description	Initial State
Response CRC7 value	[7:1]	This field contains the Response CRC7 value.	0x00
Response Endbit	[0]	This field contains the Response Endbit.	0x0

MMC RESPONSE STATUS REGISTER 0 (MMRSP0)

Register	Address	R/W	Description	Reset Value
MMRSP0	0x15A00020	R	MMC Response Status Register 0	0x00000000

MMRSP	Bit	Little	Big	Description	Initial State
Response Register	[31:24]	4th byte	1st byte	This field contains the response value to be received over the MMC channel.	0x00
Response Register	[23:16]	3rd byte	2nd byte	This field contains the response value to be received over the MMC channel.	0x00
Response Register	[15:8]	2nd byte	3rd byte	This field contains the response value to be received over the MMC channel.	0x00
Response Register	[7:0]	1st byte	4th byte	This field contains the response value to be received over the MMC channel.	0x00

MMC RESPONSE STATUS REGISTER 1 (MMRSP1)

Register	Address	R/W	Description	Reset Value
MMRSP1	0x15A00024	R	MMC Response Status Register 1	0x00000000

MMRSP	Bit	Little	Big	Description	Initial State
Response Register	[31:24]	8th byte	5th byte	This field contains the response value to be received over the MMC channel.	0x00
Response Register	[23:16]	7th byte	6th byte	This field contains the response value to be received over the MMC channel.	0x00
Response Register	[15:8]	6th byte	7th byte	This field contains the response value to be received over the MMC channel.	0x00
Response Register	[7:0]	5th byte	8th byte	This field contains the response value to be received over the MMC channel.	0x00

MMC RESPONSE STATUS REGISTER 2 (MMRSP2)

Register	Address	R/W	Description	Reset Value
MMRSP2	0x15A00028	R	MMC Response Status Register 2	0x00000000

MMRSP	Bit	Little	Big	Description	Initial State
Response Register	[31:24]	12th byte	9th byte	This field contains the response value to be received over the MMC channel.	0x00
Response Register	[23:16]	11th byte	10th byte	This field contains the response value to be received over the MMC channel.	0x00
Response Register	[15:8]	10th byte	11th byte	This field contains the response value to be received over the MMC channel.	0x00
Response Register	[7:0]	9th byte	12th byte	This field contains the response value to be received over the MMC channel.	0x00

MMC RESPONSE STATUS REGISTER 3 (MMRSP3)

Register	Address	R/W	Description	Reset Value
MMRSP3	0x15A0002C	R	MMC Response Status Register 3	0x00000000

MMRSP	Bit	Little	Big	Description	Initial State
Response Register	[31:24]	16th byte	13th byte	This field contains the response value to be received over the MMC channel.	0x00
Response Register	[23:16]	15th byte	14th byte	This field contains the response value to be received over the MMC channel.	0x00
Response Register	[15:8]	14th byte	15th byte	This field contains the response value to be received over the MMC channel.	0x00
Response Register	[7:0]	13th byte	16th byte	This field contains the response value to be received over the MMC channel.	0x00

MMC COMMAND REGISTER 0 (MMCMD0)

Register	Address	R/W	Description	Reset Value
MMCMD0	0x15A00030(Li/B, Li/HW, Li/W, Bi/W) 0x15A00032(Bi/HW) 0x15A00033(Bi/B)	R/W	MMC Command Register 0	0x00

MMCMD	Bit	Description	Initial State
Command Register (1st byte)	[7:0]	This field contains the command value to be transmitted over the MMC channel.	0x00

MMC COMMAND REGISTER 1 (MMCMD1)

Register	Address	R/W	Description	Reset Value
MMCMD1	0x15A00034	R/W	MMC Command Register 1	0x00000000

MMCMD	Bit	Little	Big	Description	Initial State
Command Register (2nd byte)	[31:24]	4th byte	1st byte	This field contains the command value to be transmitted over the MMC channel.	0x00
Command Register (3rd byte)	[23:16]	3rd byte	2nd byte	This field contains the command value to be transmitted over the MMC channel.	0x00
Command Register (4th byte)	[15:8]	2nd byte	3rd byte	This field contains the command value to be transmitted over the MMC channel.	0x00
Command Register (5th byte)	[7:0]	1st byte	4th byte	This field contains the command value to be transmitted over the MMC channel.	0x00

DATA RECEIVE CRC16 BUFFER REGISTER (MMCR16)

Register	Address	R/W	Description	Reset Value
MMCR16	0x15A00038(Li/HW, Li/W, Bi/W) 0x15A0003A(Bi/HW)	R	Data Read CRC16 Buffer Register	0x0000

CRC16B	Bit	Description	Initial State
Data Read CRC16 value	[15:0]	This field contains the data receive CRC16 buffer value.	0x0000

MMC DATA REGISTER (MMDAT)

Register	Address	R/W	Description	Reset Value
MMDAT	0x15A0003C(Li/B, Li/HW, Li/W, Bi/W) 0x15A0003E(Bi/HW) 0x15A0003F(Bi/B)	R/W	MMC Data Register	0x00

MMDAT	Bit	Description	Initial State
Data Register	[7:0]	This field contains the data to be transmitted or received over the MMC channel	0x00

20 IIC-BUS INTERFACE

OVERVIEW

The S3C2400X 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 S3C2400X RISC microprocessors can receive or transmit serial data to or from slave devices. The master S3C2400X, which can initiate a data transfer over the IIC-bus, is responsible for terminating the transfer. Standard bus arbitration procedure is used in this IIC-bus in S3C2400X.

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 8^{th} bit determines the direction of the transfer (read or write).

Every data byte put onto the SDA line should total eight bits. The number of bytes which can be sent or received during the bus transfer operation is unlimited. Data is always sent from most-significant bit (MSB) first, and every byte should be immediately followed by an acknowledge (ACK) bit.

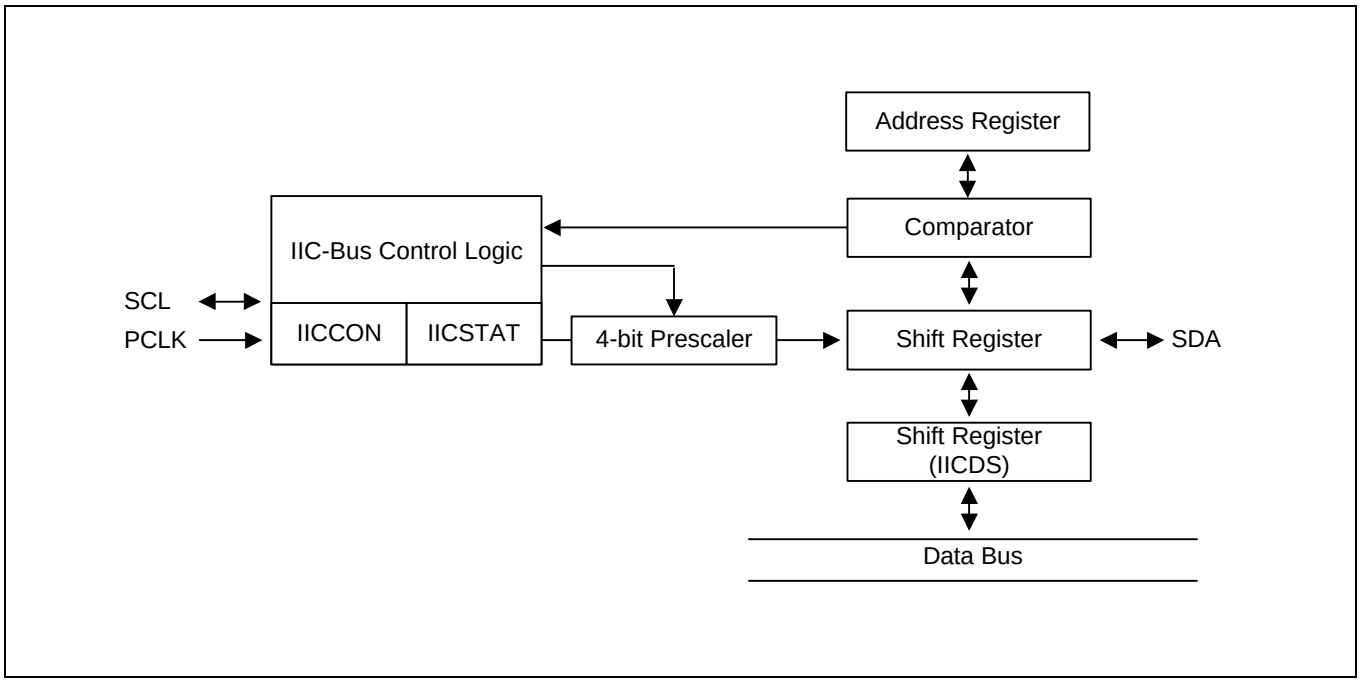


Figure 20-1. IIC-Bus Block Diagram

THE IIC-BUS INTERFACE

The S3C2400X 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 the 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 is busy when a Start condition is generated. A few clocks after a Stop condition, the IIC-bus will be free, again.

When a master initiates a Start condition, it should send a slave address to notify the slave device. The one byte of address field consist of a 7-bit address and a 1-bit transfer direction indicator (that is, 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.

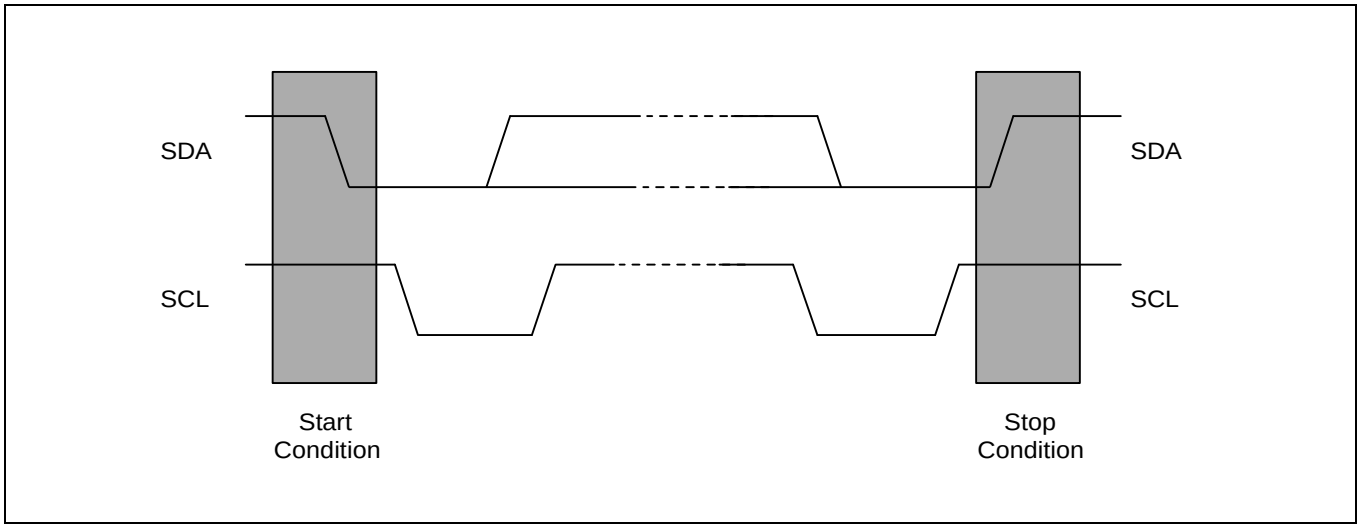
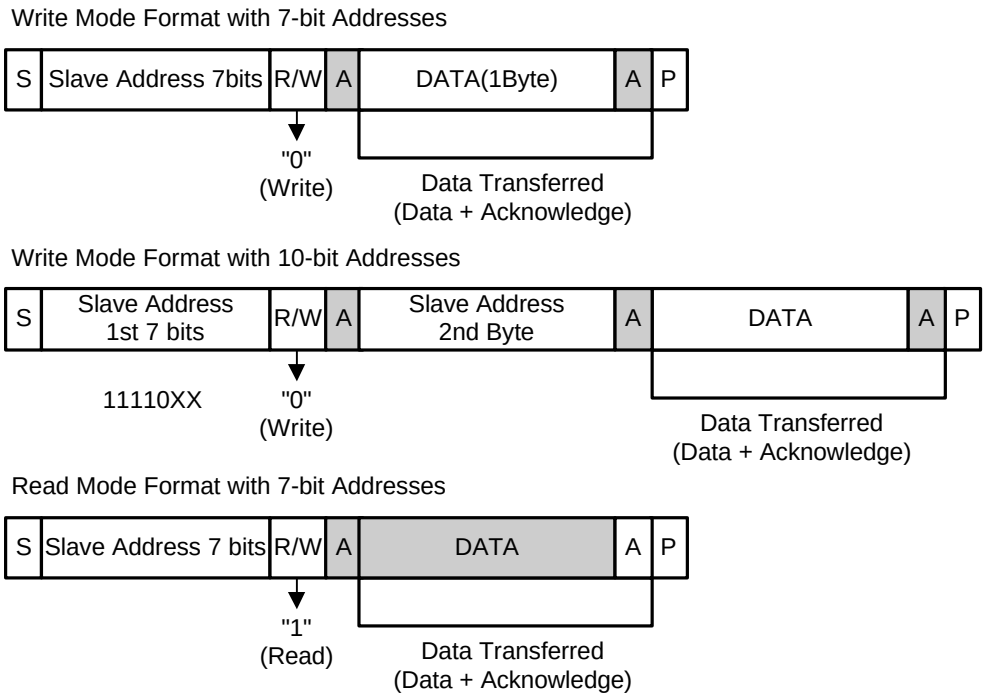


Figure 20-2. Start and Stop Condition

DATA TRANSFER FORMAT

Every byte placed on the SDA line should be eight bits in length. The number of bytes which can be transmitted per transfer is unlimited. 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.



NOTES:

1. S: Start, rS: Repeat Start, P: Stop, A: Acknowledge
2. ☐ : From Master to Slave, ☐ : from Slave to Master

Figure 20-3. IIC-Bus Interface Data Format

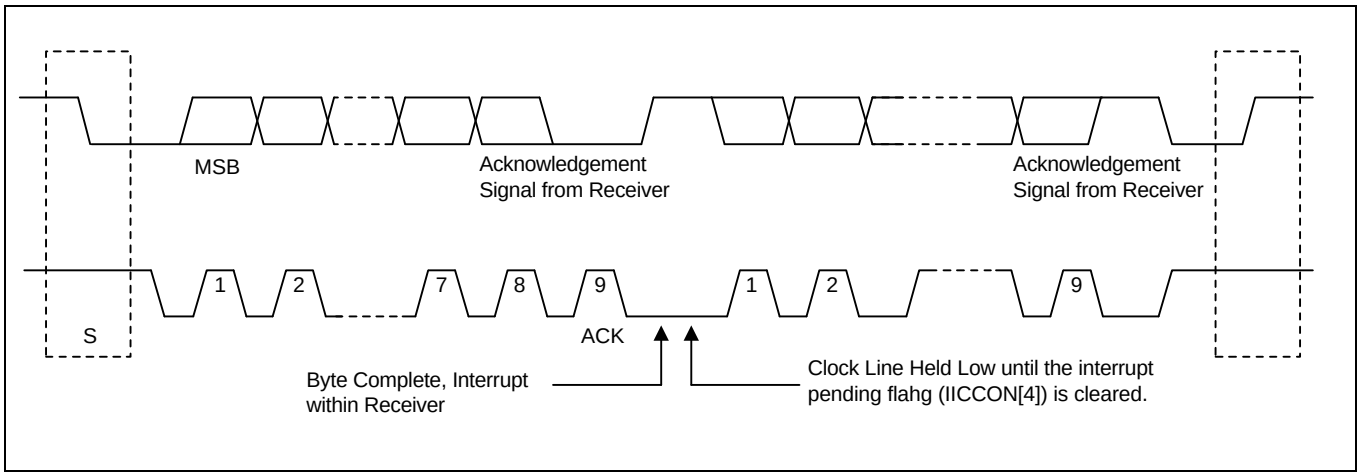


Figure 20-4. Data Transfer on the IIC-Bus

ACK SIGNAL TRANSMISSION

To finish a one-byte transfer operation completely, 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 is 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 a one-byte data transfer operation.

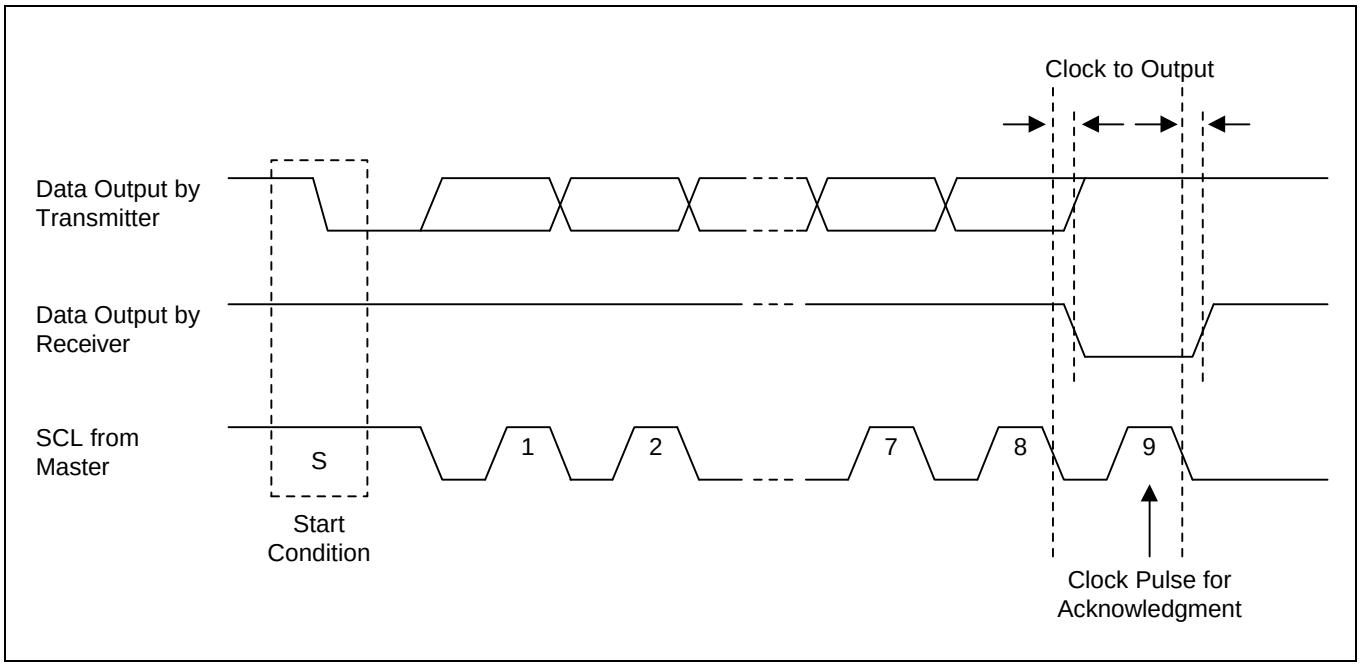


Figure 20-5. Acknowledge on the IIC-Bus

READ-WRITE OPERATION

In the transmitter mode, after the data is transferred, the IIC-bus interface will wait until IICDS(IIC-bus Data Shift Register) is written by a new data. Until the new data is written, the SCL line will be held low. After the new data is written to IICDS register, the SCL line will be released. The S3C2400X 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 IICDS, again.

In the receive mode, after a data is received, the IIC-bus interface will wait until IICDS register is read. Until the new data is read out, the SCL line will be held low. After the new data is read out from IICDS register, the SCL line will be released. The S3C2400X 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 IICDS.

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 another 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 lowering of SDA line is stronger than maintaining High on the line. For example, 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 Low is stronger than High even if first master is trying to maintain High on the line. When this happens, Low(as the first bit of address) -generating master will get the mastership and 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 second address bit, again. This arbitration will continue to the end of last address bit.

ABORT CONDITIONS

If a slave receiver can not 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 THE 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 register, IICADD. (By default, the IIC-bus interface address is an unknown value.)

FLOWCHARTS OF THE 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.

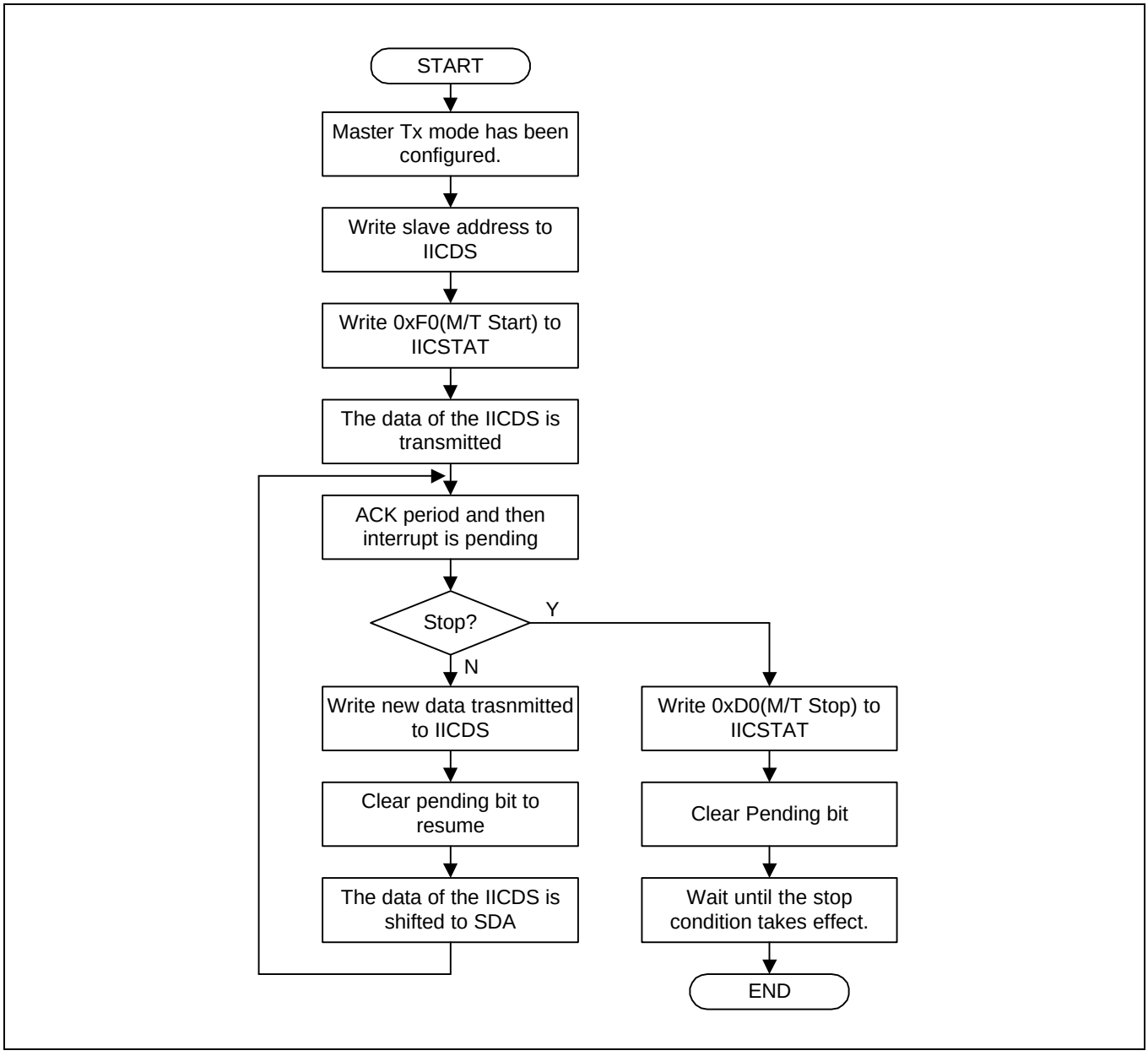


Figure 20-6. Operations for Master/Transmitter Mode

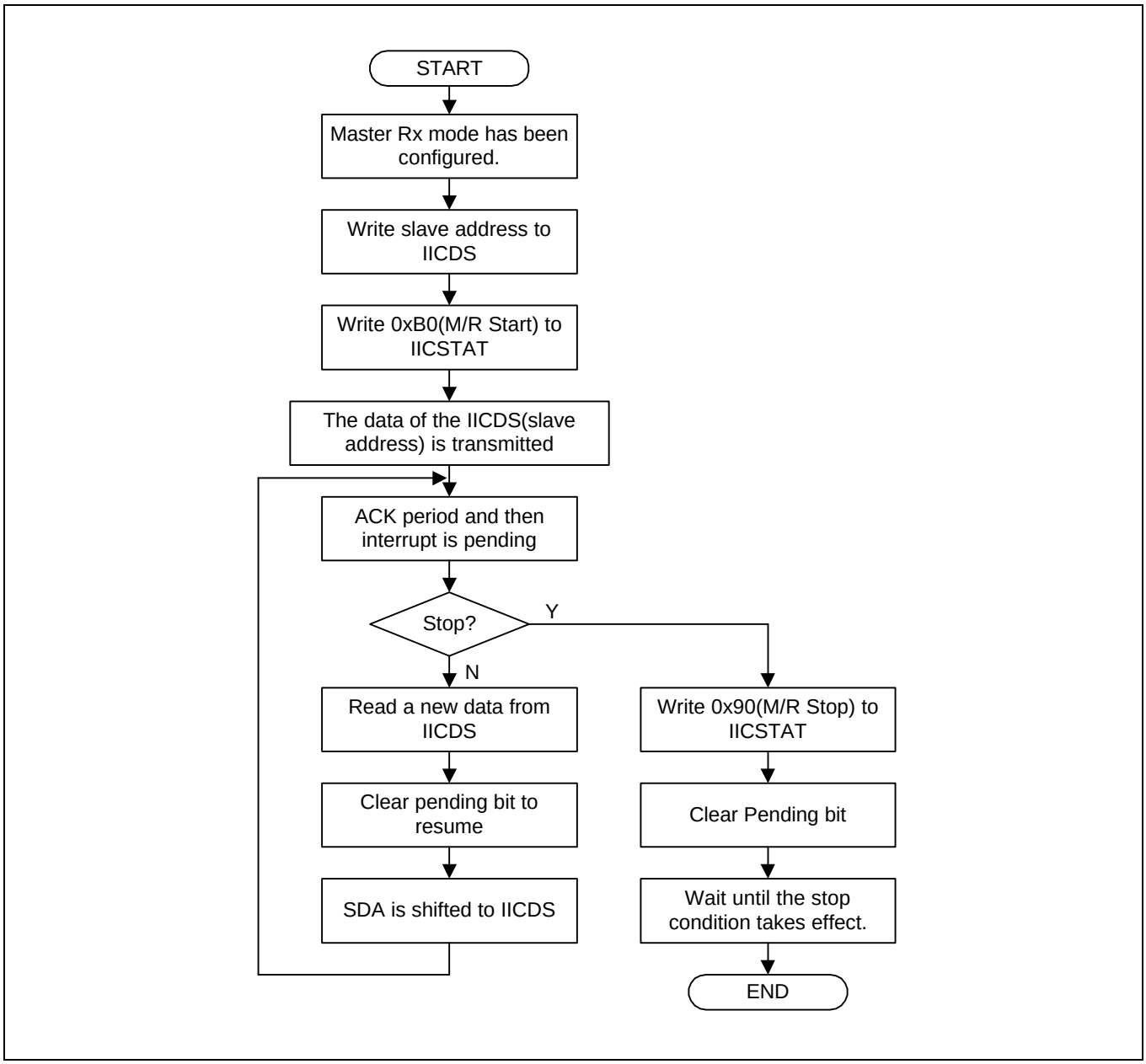


Figure 20-7. Operations for Master/Receiver Mode

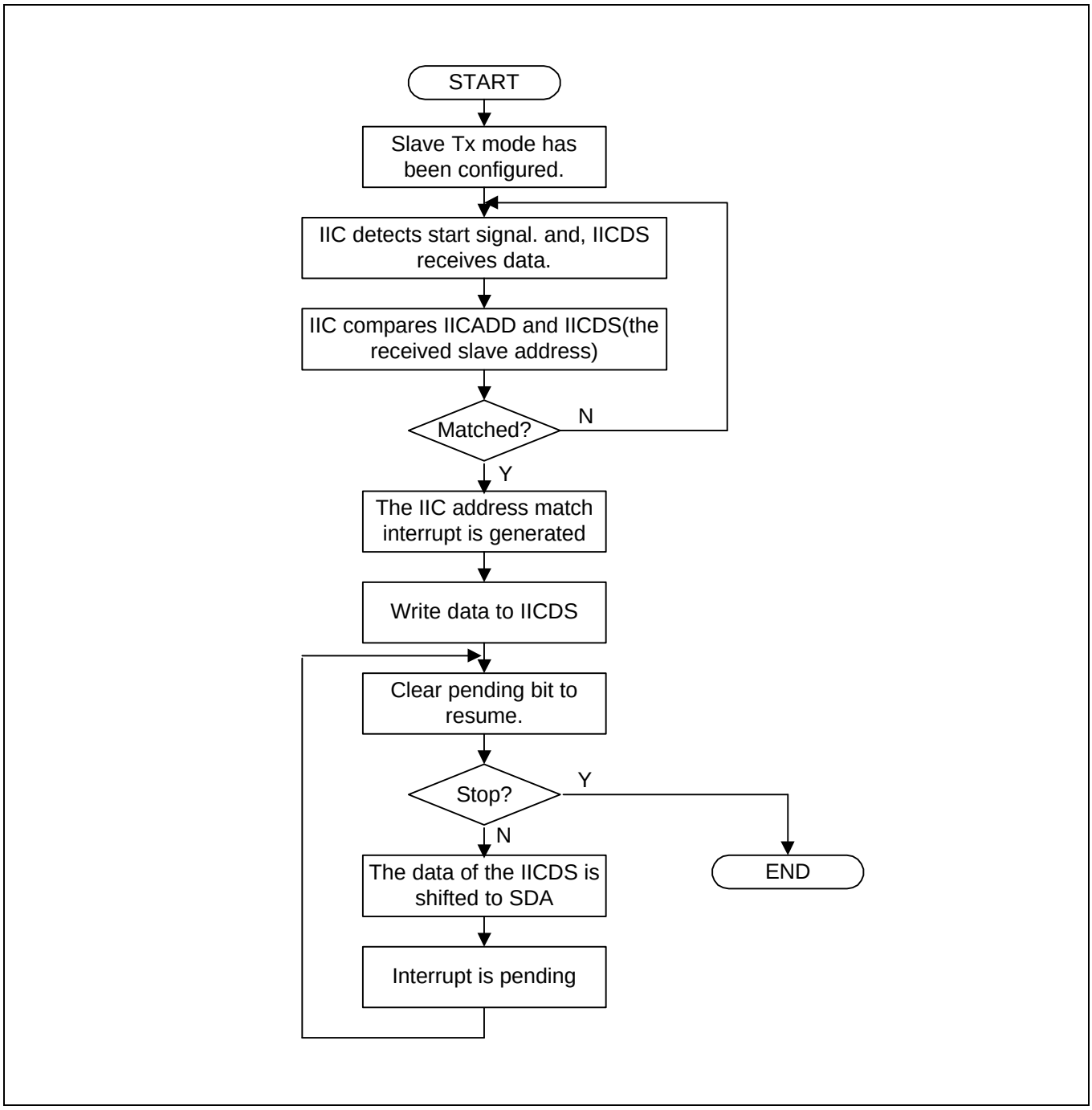


Figure 20-8. Operations for Slave/Transmitter Mode

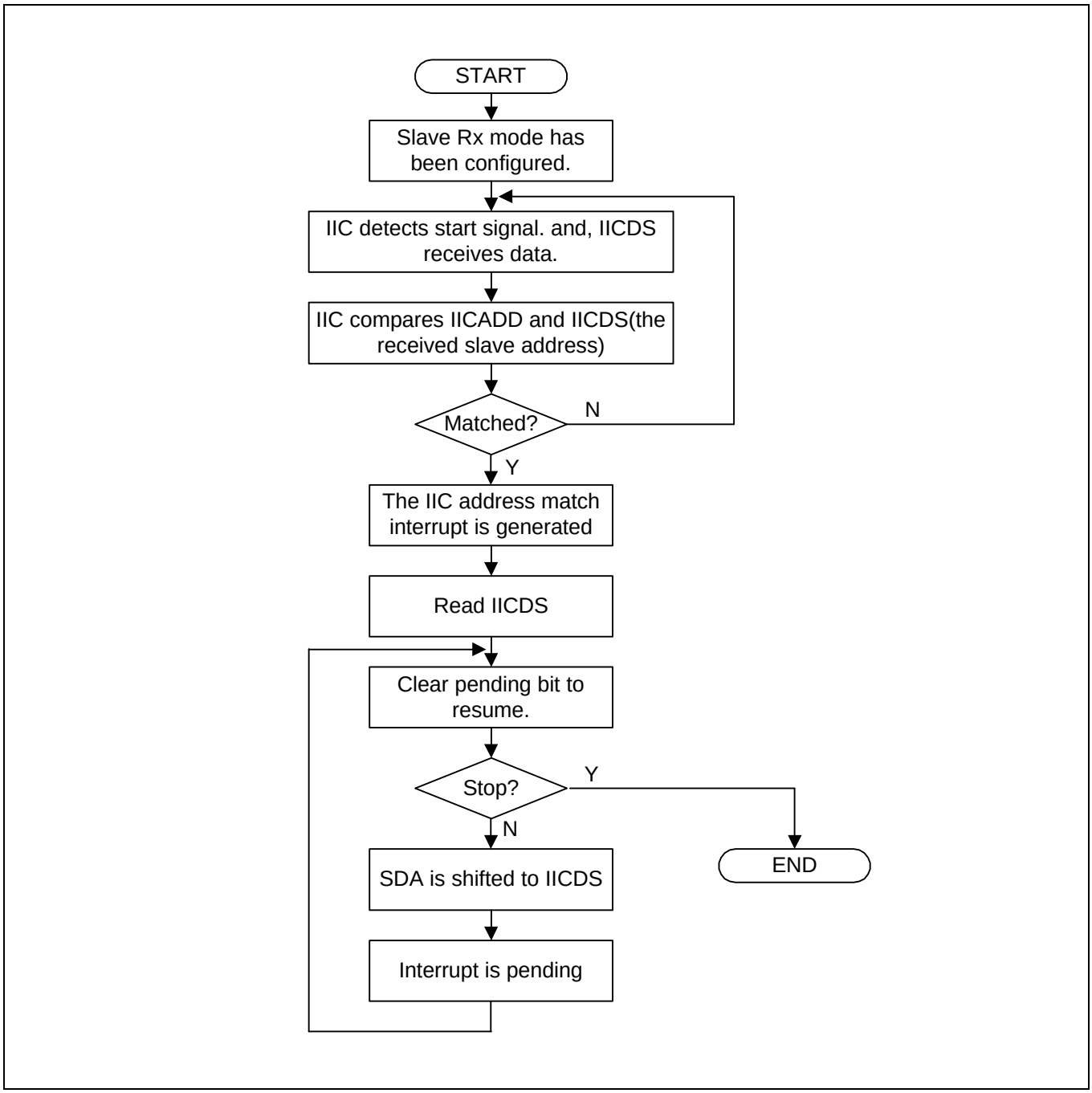


Figure 20-9. Operations for Slave / Receiver Mode

IIC-BUS INTERFACE SPECIAL REGISTERS

MULTI-MASTER IIC-BUS CONTROL REGISTER (IICCON)

Register	Address	R/W	Description	Reset Value
IICCON	0x15400000	R/W	IIC-Bus control register	0000_XXXX

IICCON	Bit	Description	Initial State
Acknowledge enable (1)	[7]	IIC-bus acknowledge enable bit. 0=Disable ACK generation 1=Enable ACK generation 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 enable (5)	[5]	IIC-Bus Tx/Rx interrupt enable/disable bit. 0=Disable interrupt, 1=Enable interrupt	0
Interrupt pending flag (2) (3)	[4]	IIC-bus Tx/Rx interrupt pending flag. Writing 1 is impossible. 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) Read) No interrupt pending 2) Write) Clear pending condition & Resume the operation. 1 = 1) Read) Interrupt is pending 2) Write) N/A	0
Transmit clock value (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. A 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 time 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 to set IICCON[4]=1, although you does not use the IIC interrupt.

MULTI-MASTER IIC-BUS CONTROL/STATUS REGISTER (IICSTAT)

Register	Address	R/W	Description	Reset Value
IICSTAT	0x15400004	R/W	IIC-Bus control/status register	0000_0000

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	0
Busy signal status / START STOP condition	[5]	IIC-Bus busy signal status bit: 0 = read) IIC-bus not busy. (IIC always senses BUS start/stop condition.) write) IIC-bus STOP signal generation 1 = read) IIC-bus busy write) IIC-bus START signal generation. The data in IICDS will be transferred automatically just after the start signal. Also, the delay to check the start condition is inserted automatically.	0
Serial output enable	[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 at the SDA/SCL line. 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-receive bit is 1 (ACK was not received).	0

MULTI-MASTER IIC-BUS ADDRESS REGISTER (IICADD)

Register	Address	R/W	Description	Reset Value
IICADD	0x15400008	R/W	IIC-Bus address register	XXXX_XXXX

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. IICADD is used only when the IIC mode is selected to slave receive/transmit mode. Slave address = [7:1] Not mapped = [0]	XXXX_XXXX

MULTI-MASTER IIC-BUS TRANSMIT/RECEIVE DATA SHIFT REGISTER (IICDS)

Register	Address	R/W	Description	Reset Value
IICDS	0x1540000C	R/W	IIC-Bus transmit/receive data shift register	XXXX_XXXX

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 NOTE: The bit[0] of the data, which is transferred just after start condition, is determined by the mode selection bit. If the mode selection bit is "receive", the bit will be 1(read). If the mode selection bit is "transmit", the bit will be 0(write).	XXXX_XXXX

NOTES

21 IIS-BUS INTERFACE

OVERVIEW

Many digital audio systems are introduced into the consumer audio market, including compact disc, digital audio tapes, digital sound processors, and digital TV sound. The S3C2400X IIS (Inter-IC Sound) 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. It supports the IIS bus data format and MSB-justified data format. IIS bus interface provides DMA transfer mode for FIFO access instead of an interrupt. It can transmit or receive data simultaneously as well as transmit or receive only.

FEATURES

— IIS, MSB-justified format compatible
— 8/16-bit data per channel
— 16, 32, 48fs (sampling frequency) serial bit clock per channel
— 256, 384fs master clock
— Programmable frequency divider for master clock and CODEC clock
— 32 bytes (2×16) FIFO for transmit and receive
— Normal and DMA transfer mode

BLOCK DIAGRAM

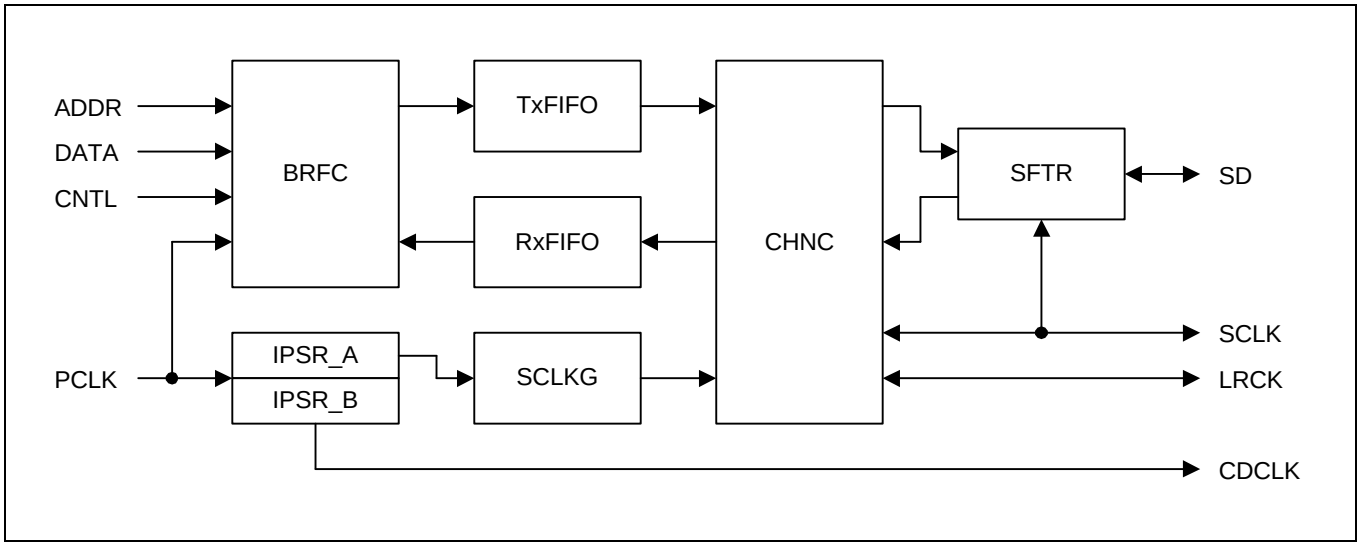


Figure 21-1. IIS-Bus Block Diagram

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.

16-byte FIFOs (TXFIFO, RXFIFO) - In transmit data transfer, data are written to TXFIFO, and, in the receive data transfer, data are read from RXFIFO.

Master IISCLK generaor (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 FIFO. 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 CPU is accessing transmit and receive FIFOs in this way.

DMA TRANSFER

In this mode, transmit or receive FIFO access is made 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, 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. It is not necessary for the transmitter 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 is sent more bits than its word length, the bits after the LSB are ignored. On the other hand, if the receiver is sent fewer bits than its word length, the missing bits are set to zero internally. And so, the MSB has a fixed position, whereas the position of the LSB depends on the word length. The transmitter always sends the MSB of the next word at one clock period after the IISLRCK change.

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 change 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 has the same lines as the IIS format. It is only different with the IIS bus that transmitter always sends the MSB of the next word when the IISLRCK change.

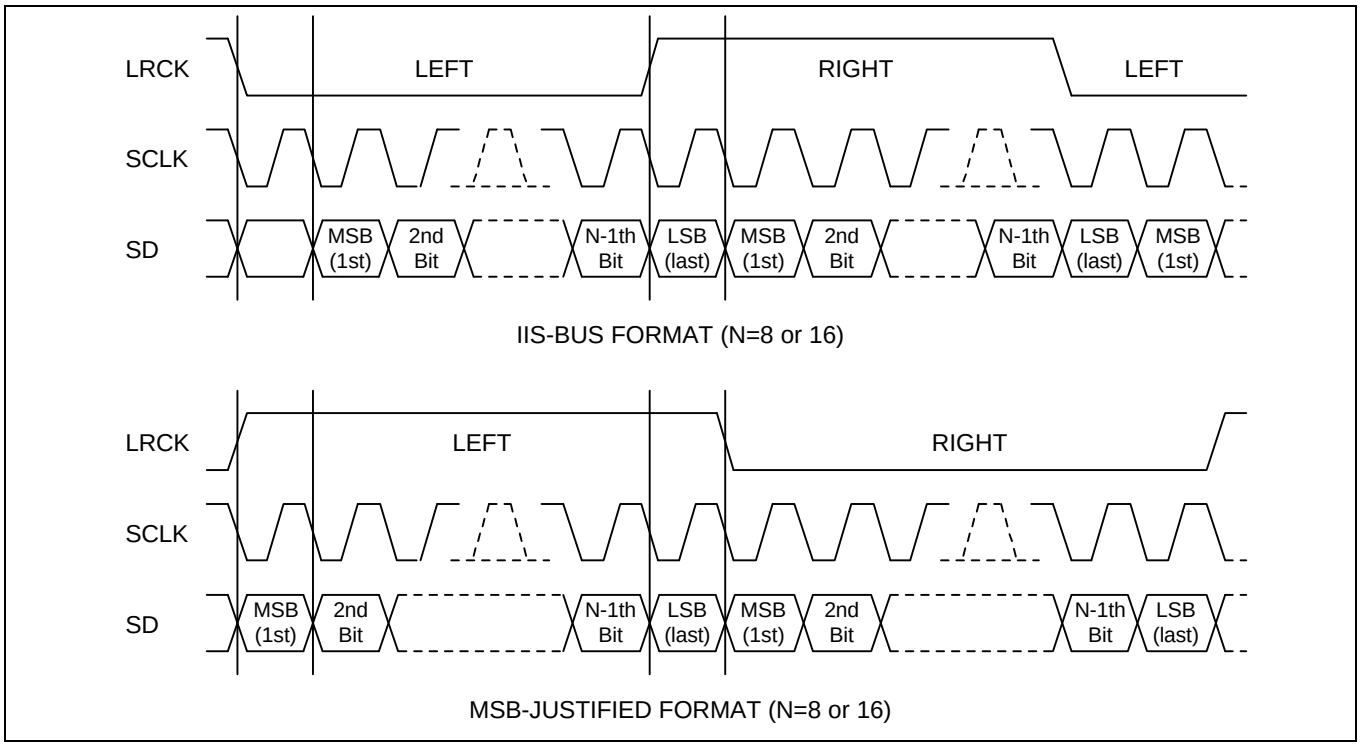


Figure 21-2. IIS-Bus and MSB(Left)-justified Data Interface Formats

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 REGISTER (IISCON)

Register	Address	R/W	Description	Reset Value
IISCON	0x15508000(Li/HW, Li/W, Bi/W) 0x15508002(Bi/HW)	R/W	IIS control register	0x100

IISCON	Bit	Description	Initial State
Left/Right channel index (read only)	[8]	0 = Left channel 1 = Right channel	1
Transmit FIFO ready flag (read only)	[7]	0 = FIFO is not ready (empty) 1 = FIFO is ready (not empty)	0
Receive FIFO ready flag (read only)	[6]	0 = FIFO is not ready (full) 1 = FIFO is ready (not full)	0
Transmit DMA service request enable	[5]	0 = Request disable 1 = Request enable	0
Receive DMA service request enable	[4]	0 = Request disable 1 = Request enable	0
Transmit channel idle command	[3]	In Idle state the IISLRCK is inactive(pause Tx) 0 = Channel not idle 1 = Channel idle	0
Receive channel idle command	[2]	In Idle state the IISLRCK is inactive(pause Rx) 0 = Channel not idle 1 = Channel idle	0
IIS prescaler enable	[1]	0 = Prescaler disable 1 = Prescaler enable	0
IIS interface enable (start)	[0]	0 = IIS disable (stop) 1 = IIS enable (start)	0

NOTES:

1. The IISCON register can be accessed by 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/B/HW/W): Access by byte/halfword/word unit when the endian mode is Little.
   (Bi/B/HW/W): Access by byte/halfword/word unit when the endian mode is Big.

IIS MODE REGISTER (IISMOD)

Register	Address	R/W	Description	Reset Value
IISMOD	0x15508004(Li/W, Li/HW, Bi/W) 0x15508006(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 can be accessed by 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): Access by halfword/word unit when the endian mode is Little.
   (Bi/HW/W): Access by halfword/word unit when the endian mode is Big.

IIS PRESCALER REGISTER (IISPSR)

Register	Address	R/W	Description	Reset Value
IISPSR	0x15508008(Li/HW, Li/W, Bi/W) 0x1550800A(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.	0
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.	000

NOTES:
1. The IISPSR register can be accessed by 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/B/HW/W): Access by byte/halfword/word unit when the endian mode is Little.
   (Bi/B/HW/W): Access by byte/halfword/word unit when the endian mode is Big.

IIS FIFO CONTROL REGISTER (IISFCON)

Register	Address	R/W	Description	Reset Value
IISFCON	0x1550800C(Li/HW, Li/W, Bi/W) 0x1550800E(Bi/HW)	R/W	IIS FIFO interface register	0x0

IISFCON	Bit	Description	Initial State
Transmit FIFO access mode select	[11]	0 = Normal access mode 1 = DMA access mode	0
Receive FIFO access mode select	[10]	0 = Normal access mode 1 = DMA access mode	0
Transmit FFO enable	[9]	0 = FIFO disable 1 = FIFO enable	0
Receive FIFO enable	[8]	0 = FIFO disable 1 = FIFO enable	0
Transmit FIFO data count (read only)	[7:4]	Data count value = 0 ~ 8	000
Receive FIFO data count (read only)	[3:0]	Data count value = 0 ~ 8	000

NOTES:
1. The IISFCON register can be accessed by 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): Access by halfword/word unit when the endian mode is Little.
   (Bi/HW/W): Access by halfword/word unit when the endian mode is Big.

IIS FIFO REGISTER (IISFIF)

IIS bus interface contains two 16-byte FIFO for the transmit and receive mode. Each FIFO has 16-width and 8-depth form, which allows the FIFO to handles data by halfword unit regardless of valid data size. Transmit and receive FIFO access is performed through FIFO entry; the address of FENTRY is 0x15508010.

Register	Address	R/W	Description	Reset Value
IISFIF	0x15508010(Li/HW) 0x15508012(Bi/HW)	R/W	IIS FIFO register	0x0

IISFIF	Bit	Description	Initial State
FENTRY	[15:0]	Transmit/Receive data for IIS	0

NOTES:

1. The IISFIF register can be accessed by halfword and word unit using STRH and LDRH instructions or short int type pointer in Little/Big endian mode.
2. (Li/HW): Access by halfword unit when the endian mode is Little.
   (Bi/HW): Access by halfword unit when the endian mode is Big.

NOTES

22 SPI INTERFACE

OVERVIEW

The S3C2400X Serial Peripheral Interface(SPI) can interface the serial data transfer. There are two 8bit shift register for transmission and receiving, respectively. During an SPI transfer, data is simultaneously transmitted (shifted out serially) and received (shifted in serially) 8bit serial data at a frequency determined by its corresponding control register settings. If you want only to transmit, you may treat the received data as dummy. Otherwise, if you want only to receive, you should transmit dummy '1' data.

There are 4 I/O pin signals associated with SPI transfers: the SCK, the MISO data line, the MOSI data line, and the active low /SS pin.

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

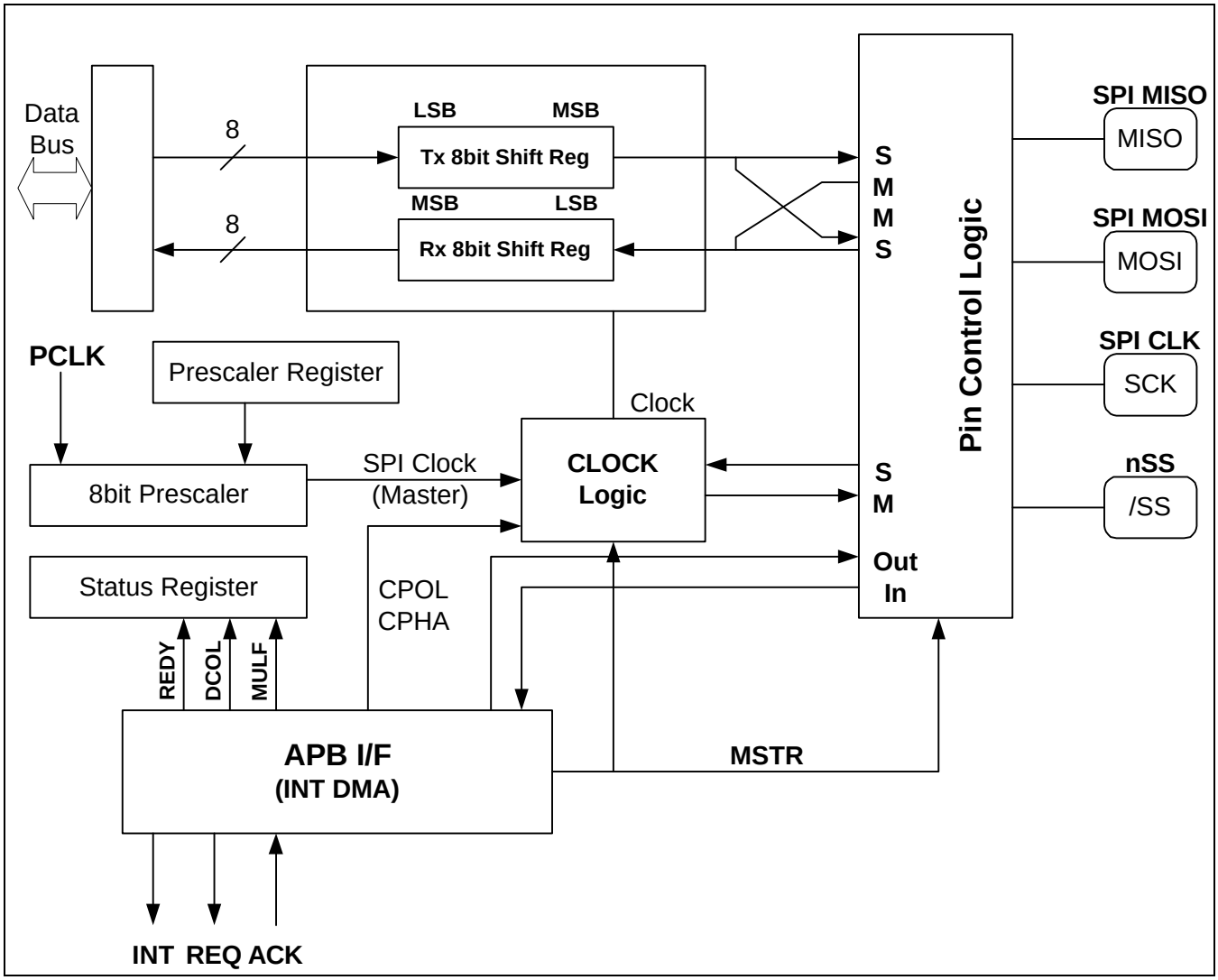


Figure 22-1. SPI Block Diagram

SPI OPERATION

Using the SPI interface, 8-bit data can be sending and receiving data simultaneously with an external device. A serial clock line synchronizes shifting and sampling of the information on the two serial data lines. The transmission frequency is controlled by making the appropriate bit settings to the SPPRE register when SPI is a master. You can modify its frequency to adjust the baud rate data register value. When SPI is a slave, other master supply the clock. When a programmer writes byte data to SPTDAT register, SPI transmit and receive operation will start simultaneously.

Programming Procedure

When a byte data is written into the SPTDAT register, SPI starts to transmit if the SPCON's SPE bit & MSTR bit are set. There is a typical programming procedure to operate an SPI card.

To program the SPI modules, follow these basic steps :

1. Set Baud Rate Prescaler Register(SPPRE)
2. Set SPCON & SPPIN to configure properly the SPI module
3. Write data 0xFF to SPTDAT 10 times in order to initialize the card
4. Set SPPIN's nCS bit(to '0') to activate the SPI card.
5. Tx data → confirm Transfer Ready flag(REDY) to set, and then write data to SPTDAT.
6. Rx data(1) : SPCON's TAGD bit disable = normal mode

→ write 0xFF to SPTDAT, then confirm REDY to set, and then read data from Read Buffer

7. Rx data(2) : SPCON'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. Reset SPPIN's nCS(to '1') to deactivate SPI card.

SPI Transfer Format

S3C2400X supports 4 different format to transfer the data. Four waveforms are shown for SPICLK.

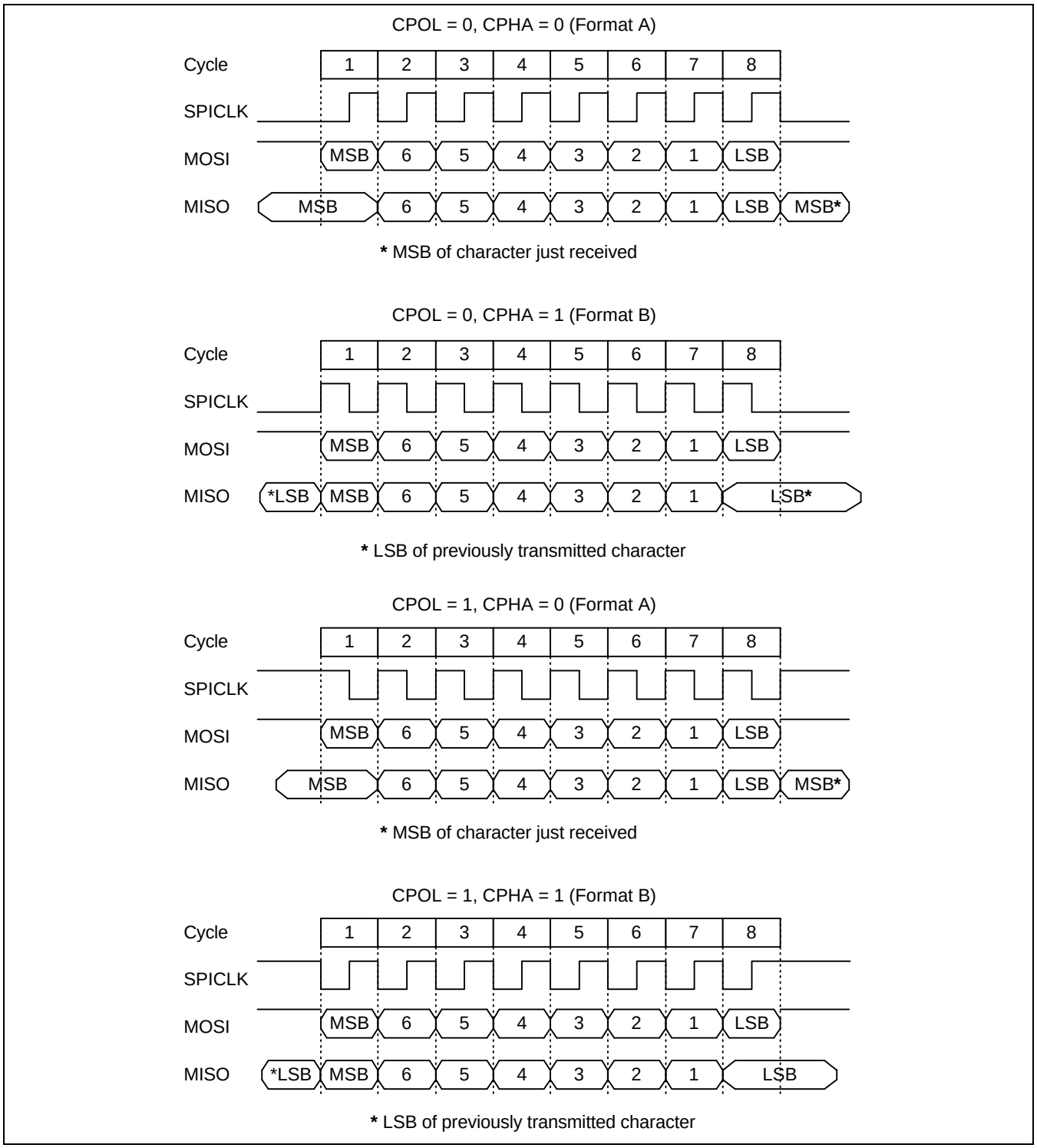


Figure 22-2. SPI Transfer Format

Steps for Transmit 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. Go to step 3 until DMA count is 0.
7. The SPI is configured as interrupt or polling mode with SMOD bits.

Steps for Receive 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 SPTDAT.
7. Go to step 4 until DMA count is 0.
8. The SPI is configured as polling mode with SMOD bits and clearing TAGD bit.
9. If SPSTA 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).
First DMA received data is dummy, so user can neglect that.

SPI SPECIAL REGISTERS

SPI CONTROL REGISTER(SPCON)

Register	Address	R/W	Description	Reset Value
SPCON	0x15900000	R/W	SPI Control Register	0x00

SPCON	Bit	Description	Initial State
SPI Mode Select (SMOD)	[6:5]	Determines how and by what SPTDAT is read/written 00 = polling mode, 01 = interrupt mode 10 = DMA mode, 11 = reserved	00
Prescaler Enable (ENPRE)	[4]	Determines what you want Prescaler enable or not(only master) 0 = disable, 1 = enable	0
Master/Slave Select(MSTR)	[3]	Determines what mode you want 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]	Determines an active high or active low clock. 0 = active high, 1 = active low	0
Clock Phase Select(CPHA)	[1]	This bit selects one of two fundamentally different transfer formats. 0 = format A, 1 = format B	0
Tx Auto Garbage Data mode enable (TAGD)	[0]	This bit decides whether the receiving data only needs or not. When this bit is normal mode and you only want to receive data, you should transmit dummy 0xFF data. 0 = normal mode, 1 = Tx auto garbage data mode	0

SPI STATUS REGISTER(SPSTA)

Register	Address	R/W	Description	Reset Value
SPSTA	0x15900004	R	SPI Status Register	0x01

SPSTA	Bit	Description	Initial State
Reserved	[7:3]		
Data Collision Error Flag(DCOL)	[2]	This flag is set if the SPTDAT is written or the SPRDAT is read while a transfer is in progress and cleared by reading the SPSTA with DCOL set 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 SPPIN's ENMUL bit is multi master errors detect mode. MULF is cleared by reading SPSTA with MULF set. 0 = not detect, 1 = multi master error detect	0
Transfer Ready Flag (REDY)	[0]	This bit indicates that SPT(R)DAT is ready to transmit or receive. This flag is automatically resetted by writing data to SPTDAT. Initial value is high. 0 = not ready, 1 = data Tx/Rx ready	1

SPI PIN CONTROL REGISTER(SPPIN)

When the SPI system is enabled, the direction of pin is controlled by SPCON's MSTR bit except /SS pin. The direction and usage of /SS pin is controlled by the SPPIN's ENMUL bit when the SPI system is a master. Otherwise, when the SPI system is a slave, /SS pin is always used input for slave select by one master.

Register	Address	R/W	Description	Reset Value
SPPIN	0x15900008	R/W	SPI Pin Control Register	0x02

SPPIN	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, regardless the SPPIN's nCS bit. 0 = disable(general purpose), 1 = multi master error detect enable	0
SPI Card Select(nCS)	[1]	The /SS pin is used as an output to select an SPI card when the SPI system is a master. If SPPIN's EMUL bit is enable, this bit ignores. 0 = card select, 1 = not select	1
Master Out Keep(KEEP)	[0]	Determines MOSI drive or release when 1byte transmit finish(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 as a slave, these pins reverse roles. In a multiple-master system, all SPICLK(SCK) pins are tied together, all SPIMOSI(MOSI) pins are tied together, and all SPIMISO(MISO) pins are tied together.

Only an SPI master can experience a multi master error, caused when a second SPI device becomes a master and selects this device as if it were a slave. When this type error is detected, the following action are taken immediately. But you must previously set SPPIN’s ENMUL bit if you want to detect this error.

1. The SPCON's MSTR bit is forced to 0 to operate slave mode.
2. The SPSTA's MULF flag is set, and an SPI interrupt is generated.

SPI BAUD RATE PRESCALER REGISTER(SPPRE)

Register	Address	R/W	Description	Reset Value
SPPRE	0x1590000C	R/W	SPI Baud Rate Prescaler Register	0x00

SPPRE	Bit	Description	Initial State
Prescaler Value	[7:0]	Determines SPI clock rate as above equation. Baud rate = PCLK / 2 / (Prescaler value + 1)	0x00

NOTE: Baud rate should be less than 25MHz.

SPI TX DATA REGISTER(SPTDAT)

Register	Address	R/W	Description	Reset Value
SPTDAT	0x15900010	R/W	SPI Tx Data Register	0x00

SPTDAT	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(SPRDAT)

Register	Address	R/W	Description	Reset Value
SPRDAT	0x15900014	R	SPI Rx Data Register	0x00

SPRDAT	Bit	Description	Initial State
Rx Data Register	[7:0]	This field contains the data to be received over the SPI channel.	0x00

23 ELECTRICAL DATA

ABSOLUTE MAXIMUM RATINGS

Table 23-1. Absolute Maximum Rating

Parameter	Symbol	Rating		Unit
DC Supply Voltage	V_{DDi}	1.8V V_{DD}	2.7	V
	V_{DDIO}	3.3V V_{DD}	3.8	
DC Input Voltage	V_{IN}	3.3V Input buffer	3.8	
DC Output Voltage	V_{OUT}	3.3V Output buffer	3.8	
DC Input Current	I_{IN}	± 200		mA
Storage Temperature	T_{STG}	– 65 to 150		°C

RECOMMENDED OPERATING CONDITIONS

Table 23-2. Recommended Operating Conditions

Parameter	Symbol	Rating		Unit
DC Supply Voltage for Internal	V_{DDi}	1.8V V_{DD}	1.8 ± 0.15	V
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 ± 0.5%	
DC Input Voltage	V_{IN}	3.3V Input buffer	3.3 ± 0.3	
DC Output Voltage	V_{OUT}	3.3V Output buffer	3.3 ± 0.3	
Operating Temperature	T_{OPR}	Commercial	0 to 70	°C

D.C. ELECTRICAL CHARACTERISTICS

The following tables define the DC electrical characteristics for the standard LVCMOS I/O buffers.

Table 23-3. 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					V
	LVCMOS interface		2.0			
V_{IL}	Low level input voltage					V
	LVCMOS interface				0.8	
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			
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		-110	-66	-40	
V_{OH}	High level output voltage					
	Type B8	I_{OH}= -8 mA	2.4			V
	Type B12	I_{OH} = -12 mA				
V_{OL}	Low level output voltage					
	Type B8	I_{OL} = 8 mA			0.4	V
	Type B12	I_{OL} = 12 mA				

NOTE: Type B8 means 8mA output driver cell, and type B12 means 12mA output driver cells.

Table 23-4. USB DC Electrical Characteristics

Parameter	Symbol	Condition	Min	Max	Unit
High level input voltage	V_{IH}		2.5		V
Low level input voltage	V_{IL}			0.8	V
High level input current	I_{IH}	Vin = 3.3V	-10	10	µA
Low level input current	I_{IL}	Vin = 0.0V	-10	10	µA
Static Output High	V_{OH}	15KΩ to GND	2.8	3.6	V
Static Output Low	V_{OL}	1.5KΩ to 3.6V		0.3	V

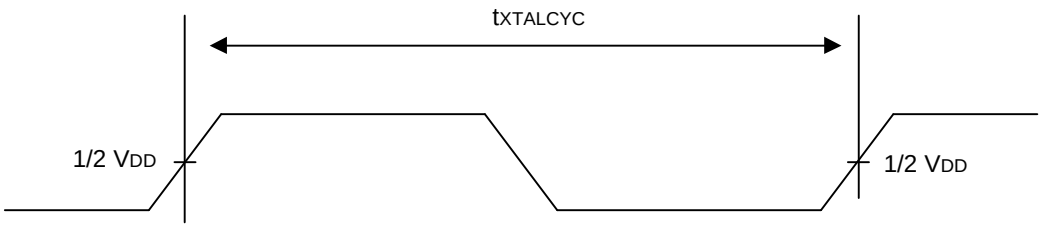
Table 23-5. S3C2400X Power Supply Voltage and Current

Parameter	Value	Unit	Condition
Typical V_{DDi} / V_{DDIO}	1.8 / 3.3	V	
Max. Operating frequency (FCLK)	133	MHz	
Max. Operating frequency (HCLK,PCLK)	66	MHz	
Typical normal mode power (4) (Total V_{DDi} + V_{DDIO})	100.4	mW	(1)
Typical normal mode power (4) (Total V_{DDi} + V_{DDIO})	111.7	mW	(2)
Typical normal mode power (4) (Total V_{DDi} + V_{DDIO})	197.6	mW	(3)
Typical idle mode power (4) (Total V_{DDi} + V_{DDIO})	66.7	mW	HCLK = 66MHz (F:H:P = 1:2:4)
Typical slow mode power (4) (Total V_{DDi} + V_{DDIO})	33.8	mW	FCLK = 12MHz (F:H:P = 1:2:4)
Maximum stop mode power Typical stop mode power (4)	100 20	uA uA	Just running 32 kHz oscillator (for RTC), all other I/O static.
Maximum RTC power Typical RTC power (4)	5 2	uA uA	x-tal = 32.768 kHz for RTC

NOTES:

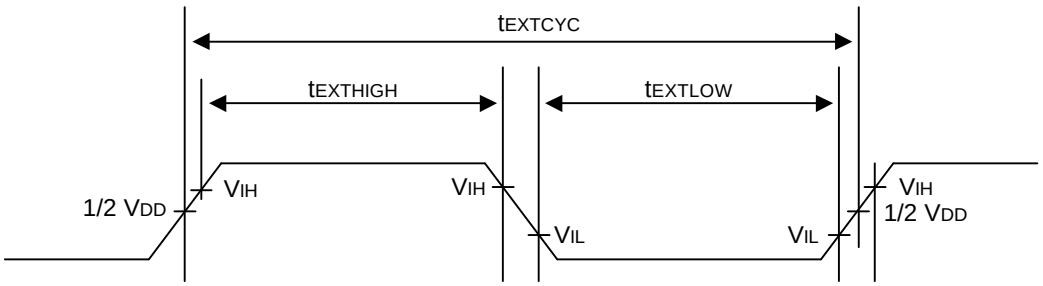
1. I/D cache: ON, MMU: ON, Code on SRAM, FCLK:HCLK:PCLK = 133MHz:66MHz:33MHz
2. Note1 + LCD ON (320x240x8bppx140Hz, 256 color STN)
3. Note1 + Download from USB to SDRAM with DMA
4. Room temperature specification.

A.C. ELECTRICAL CHARACTERISTICS



NOTE: The clock input from the XTIpll pin.

Figure 23-1. XTIpll Clock Timing



NOTE: The clock input from the EXTCLK pin.

Figure 23-2. EXTCLK Clock Input Timing

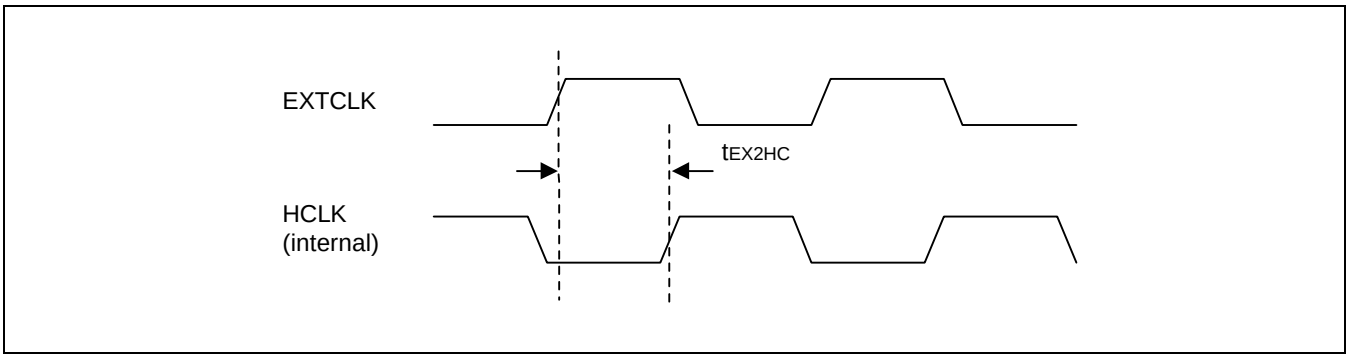


Figure 23-3. EXTCLK/HCLK in case that EXTCLK is used without the PLL

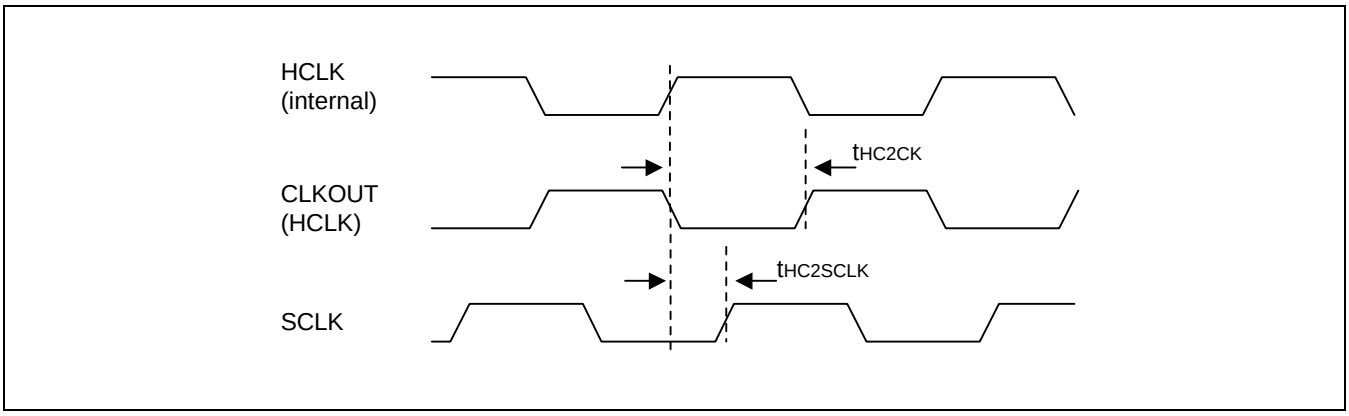


Figure 23-4. HCLK/CLKOUT/SCLK in case that EXTCLK is used

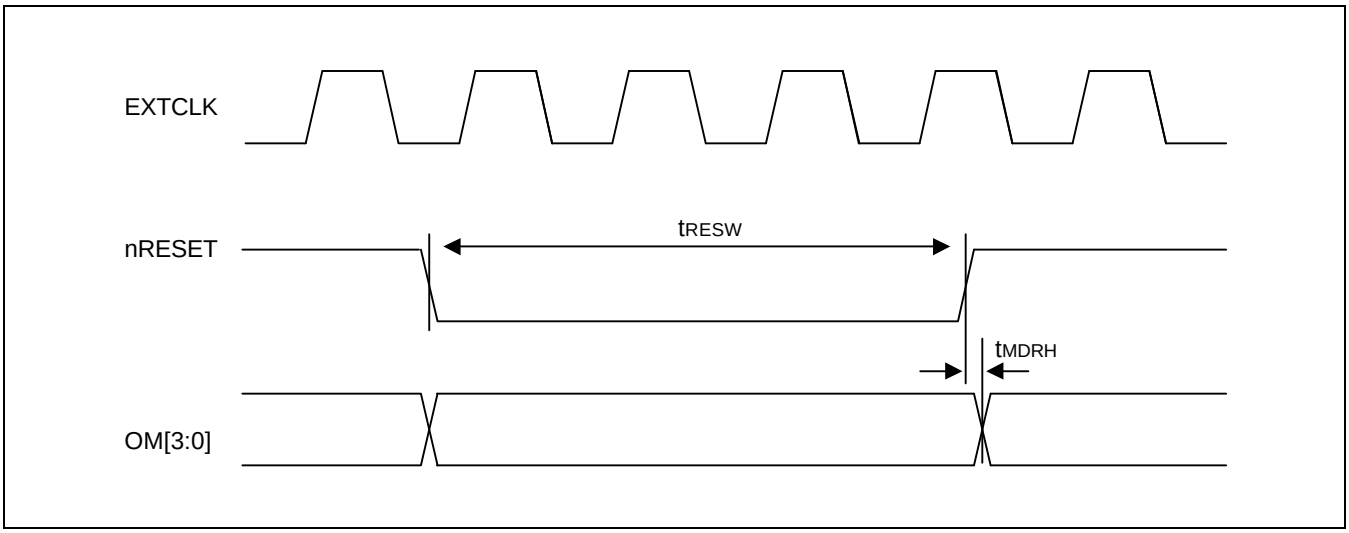


Figure 23-5. Manual Reset and OM[3:0] Input Timing

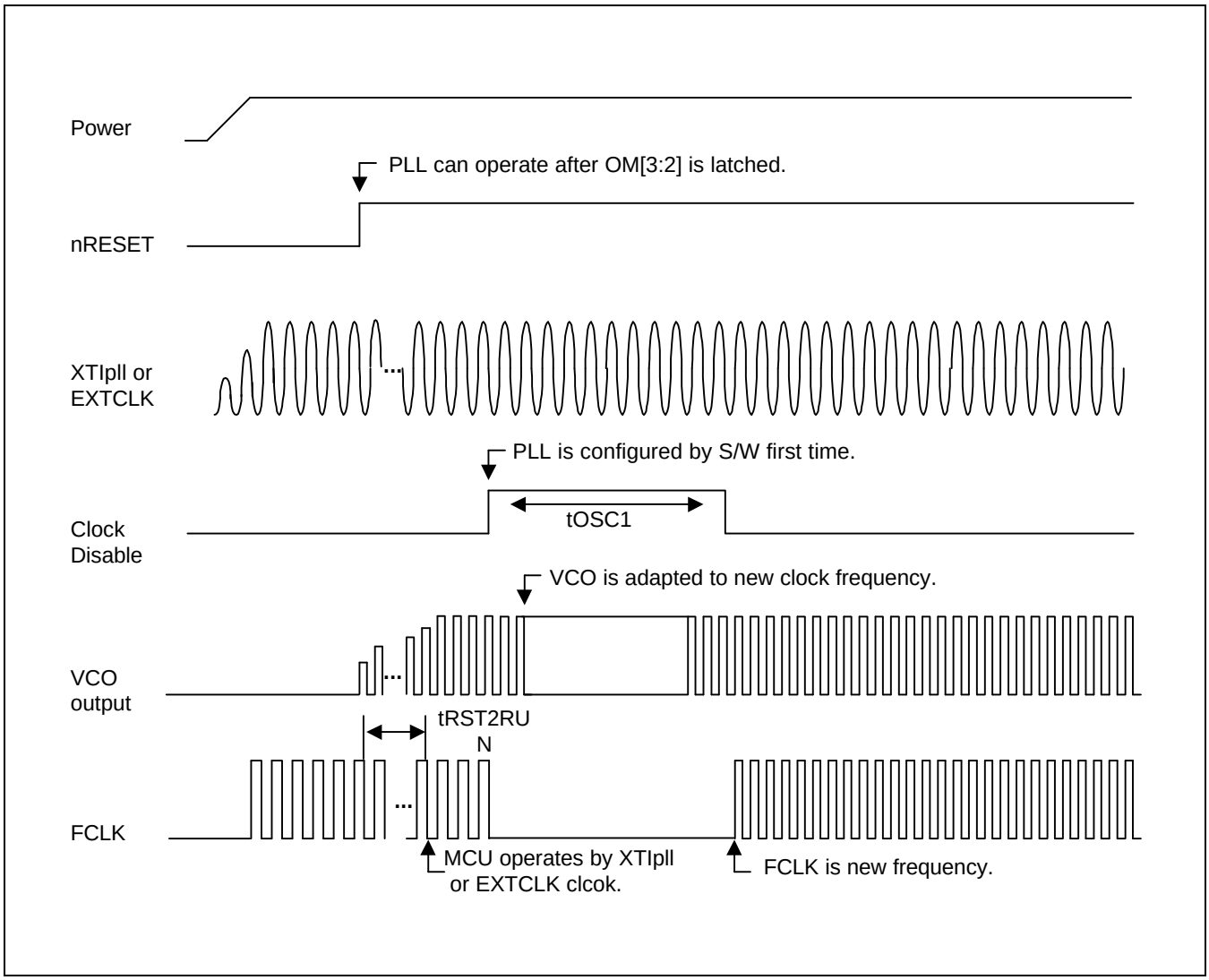


Figure 23-6. Power-On Oscillation Setting Timing

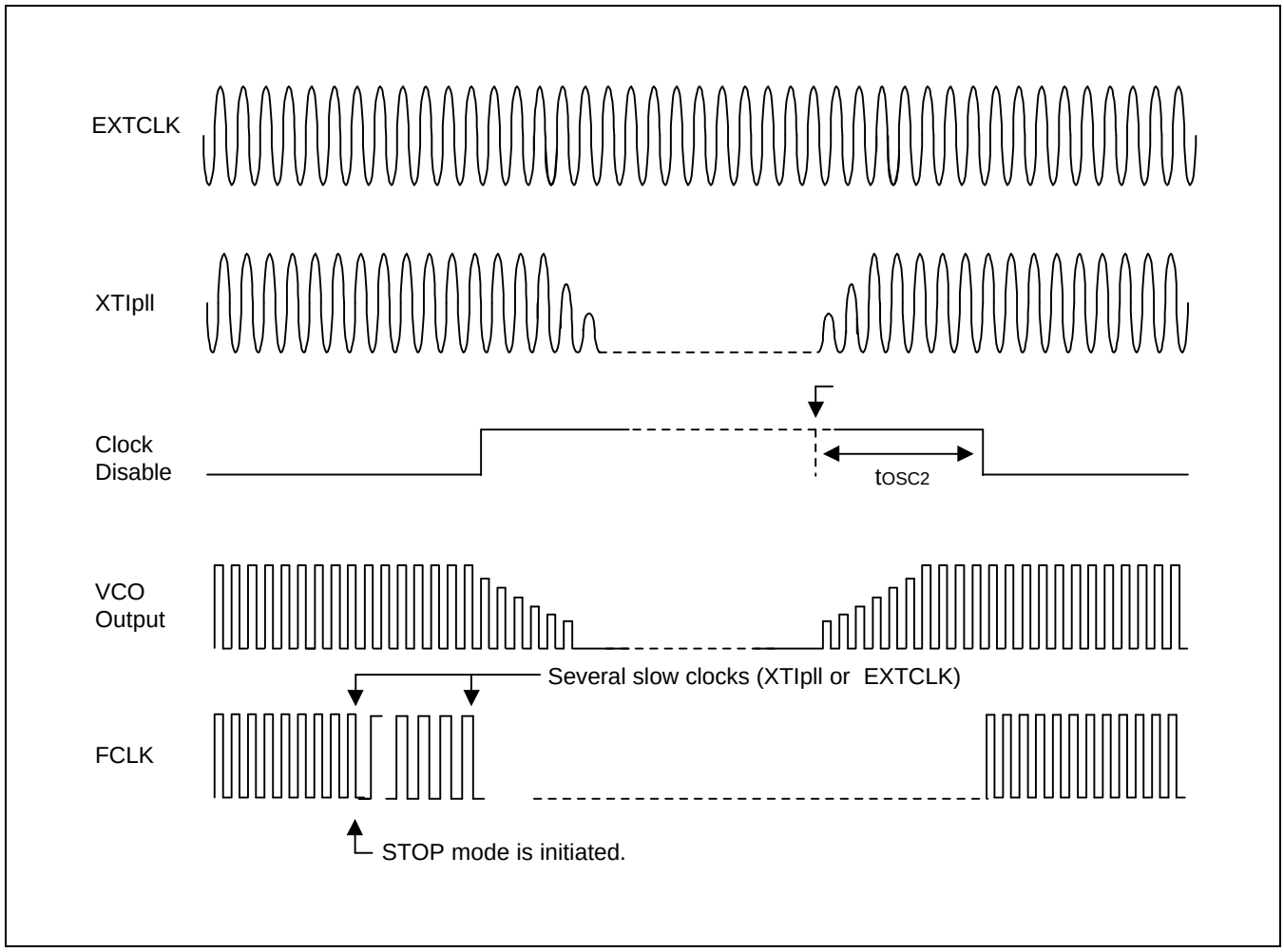


Figure 23-7. STOP Mode Return Oscillation Setting Timing

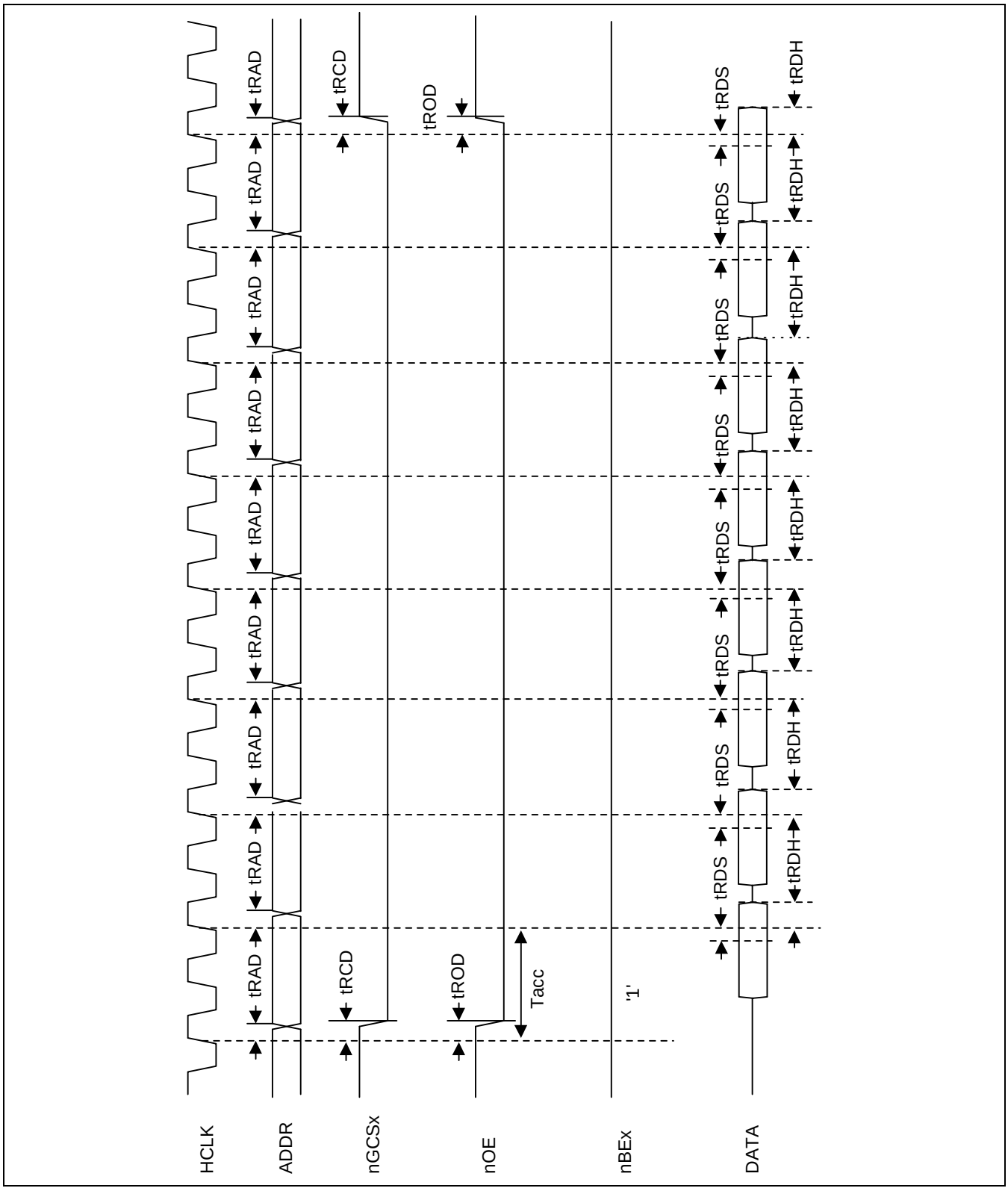


Figure 23-8. ROM/SRAM Burst READ Timing(I)
(Tacs=0, Tcos=0, Tacc=2, Toch=0, Tcah=0, PMC=0, ST=0, DW=16bit)

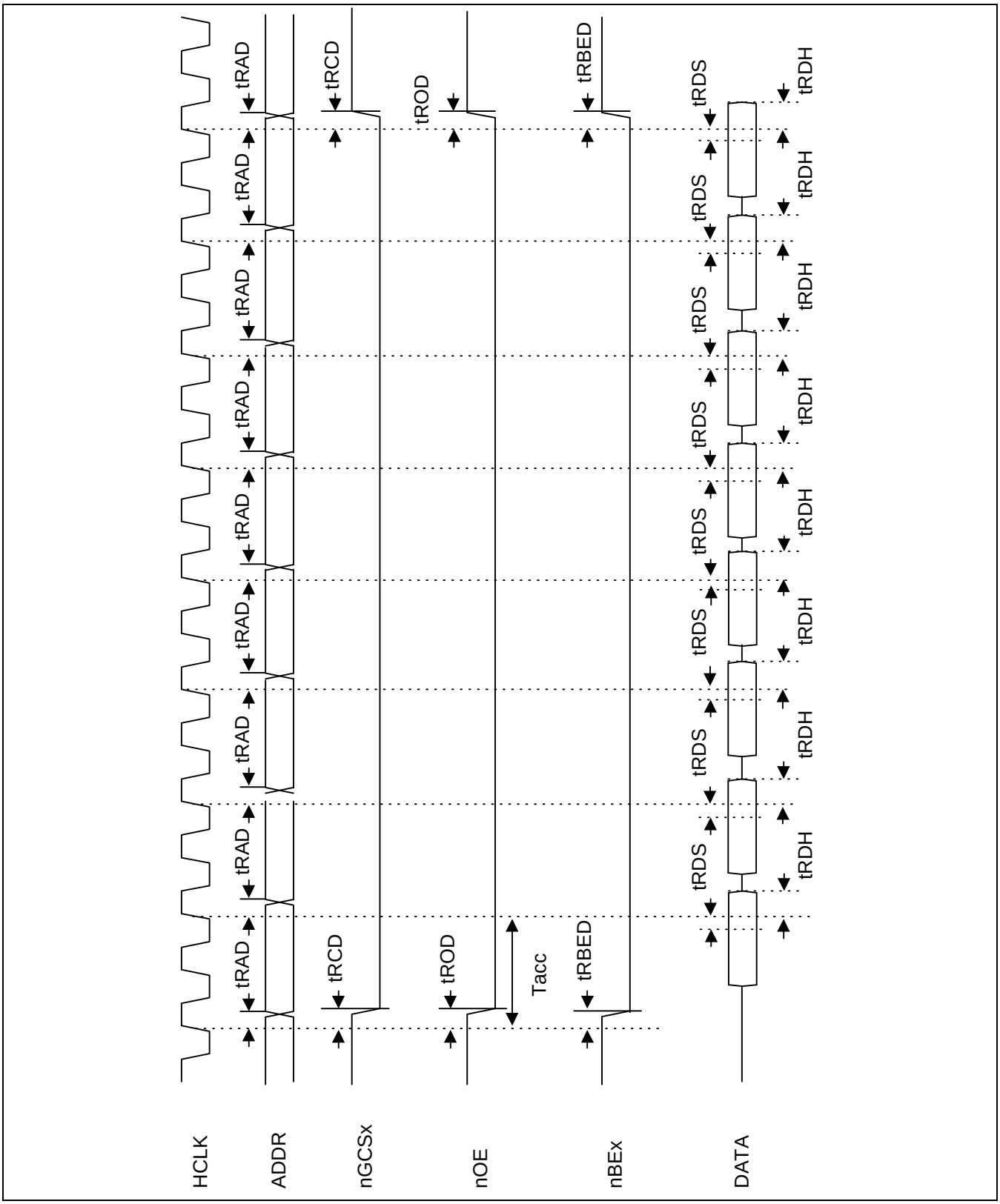


Figure 23-9. ROM/SRAM Burst READ Timing(II)
(Tacs=0, Tcos=0, Tacc=2, Toch=0, Tcah=0, PMC=0, ST=1, DW=16bit)

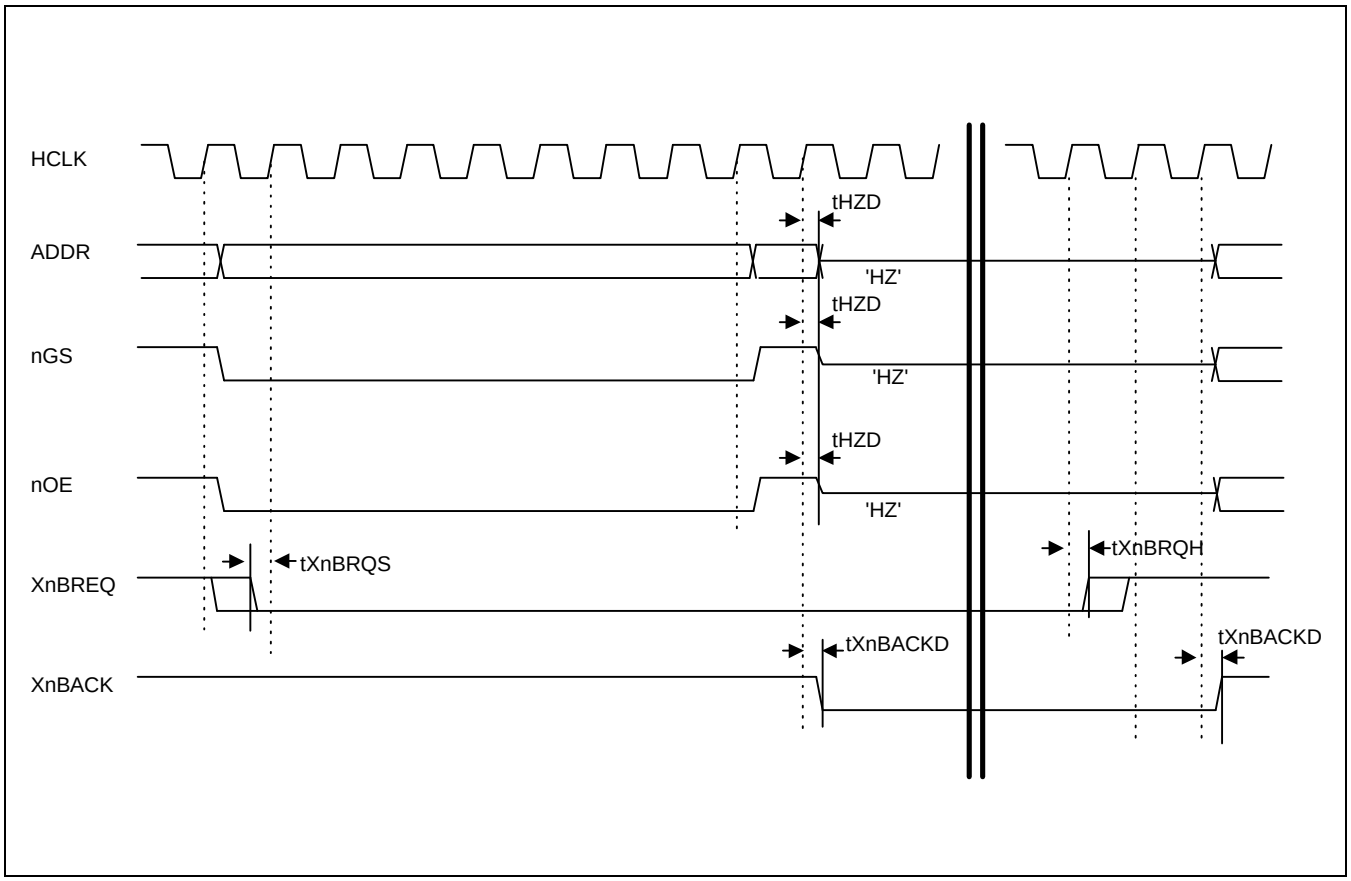


Figure 23-10. External Bus Request in ROM/SRAM Cycle (Tacs=0, Tcos=0, Tacc=8, Toch=0, Tcah=0, PMC=0, ST=0)

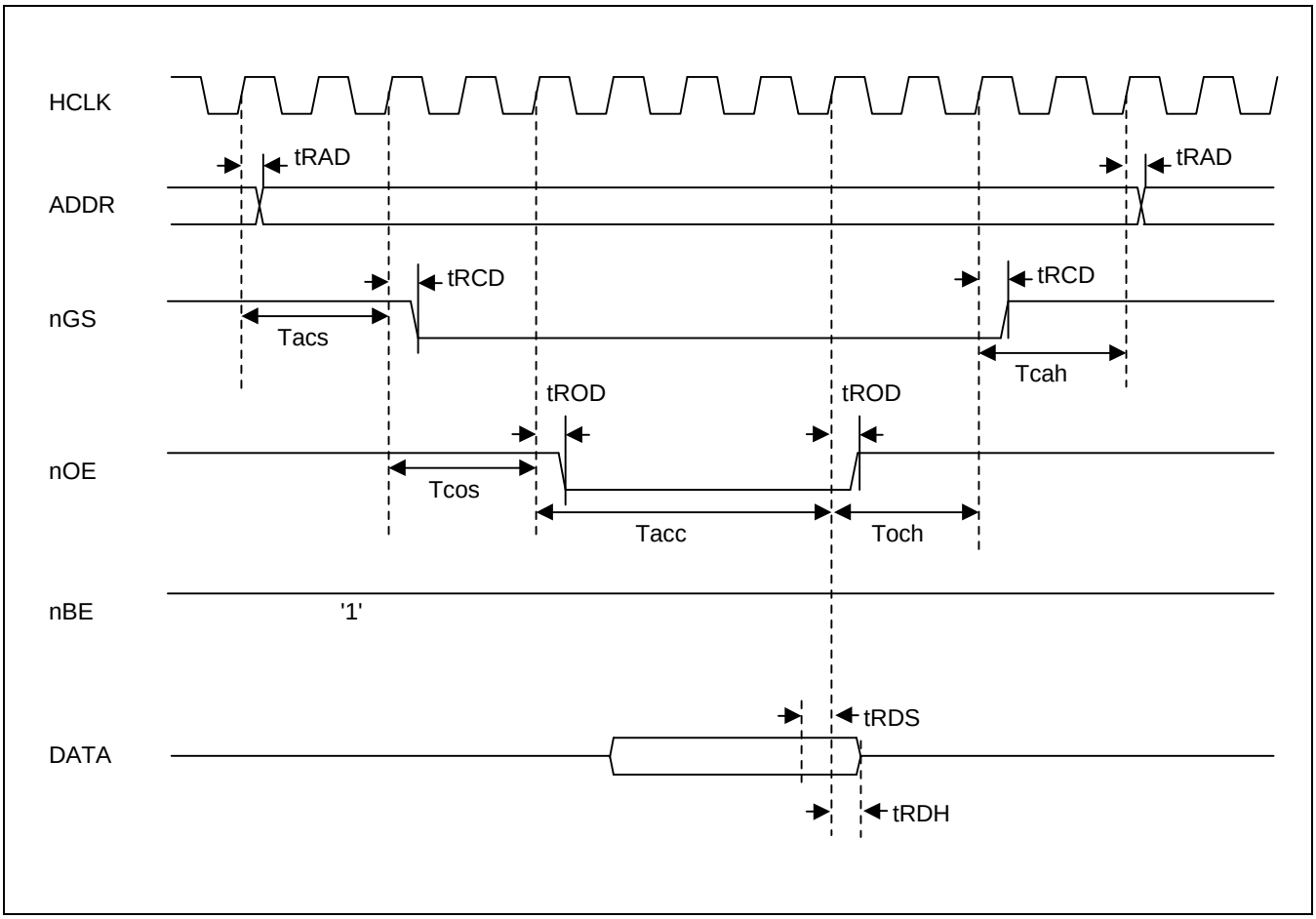


Figure 23-11. ROM/SRAM READ Timing (I)
(Tacs=2,Tcos=2, Tacc=4, Toch=2, Tcah=2, PMC=0, ST=0)

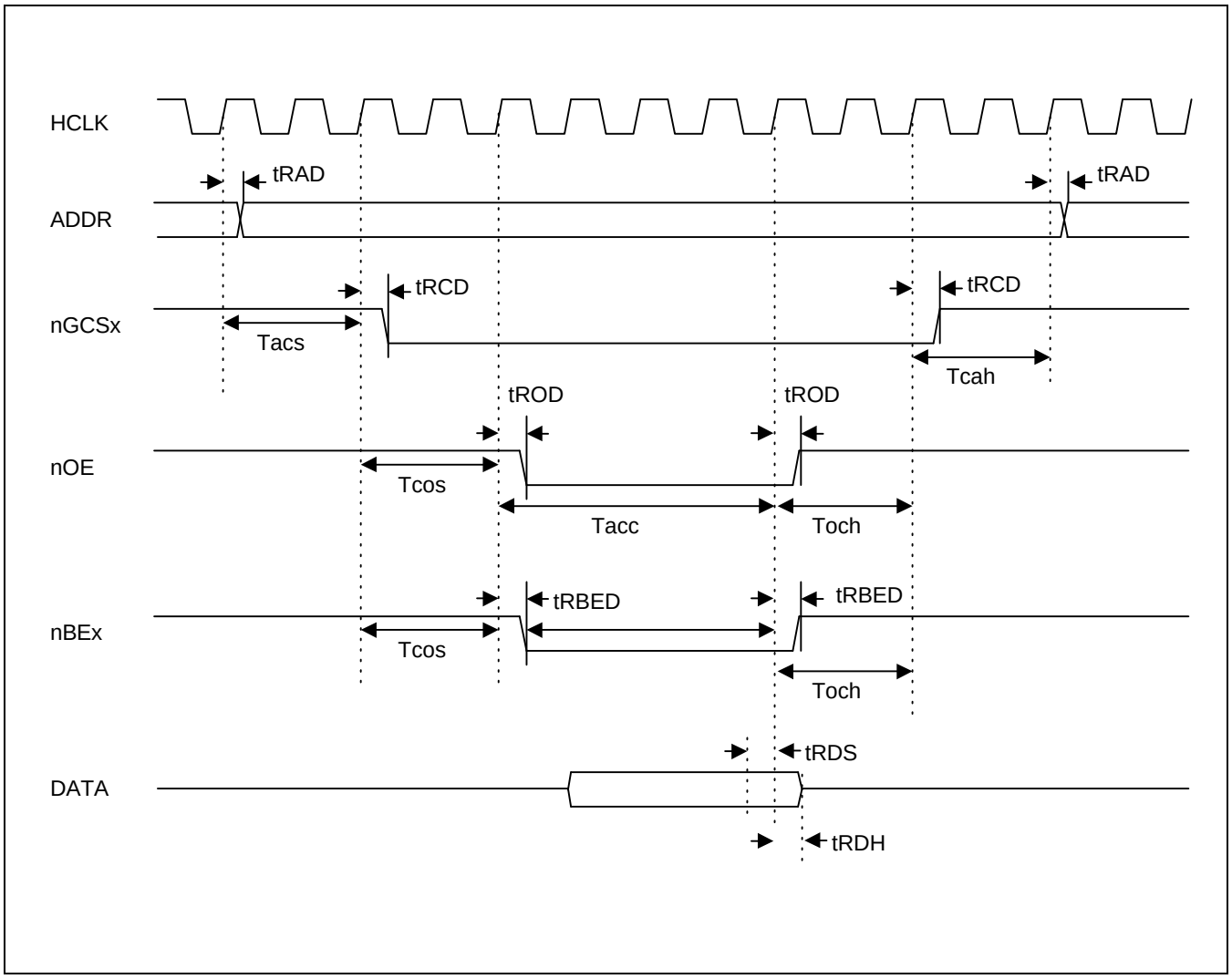


Figure 23-12. ROM/SRAM READ Timing (II)
(Tacs=2, Tcos=2, Tacc=4, Toch=2, Tcah=2cycle, PMC=0, ST=1)

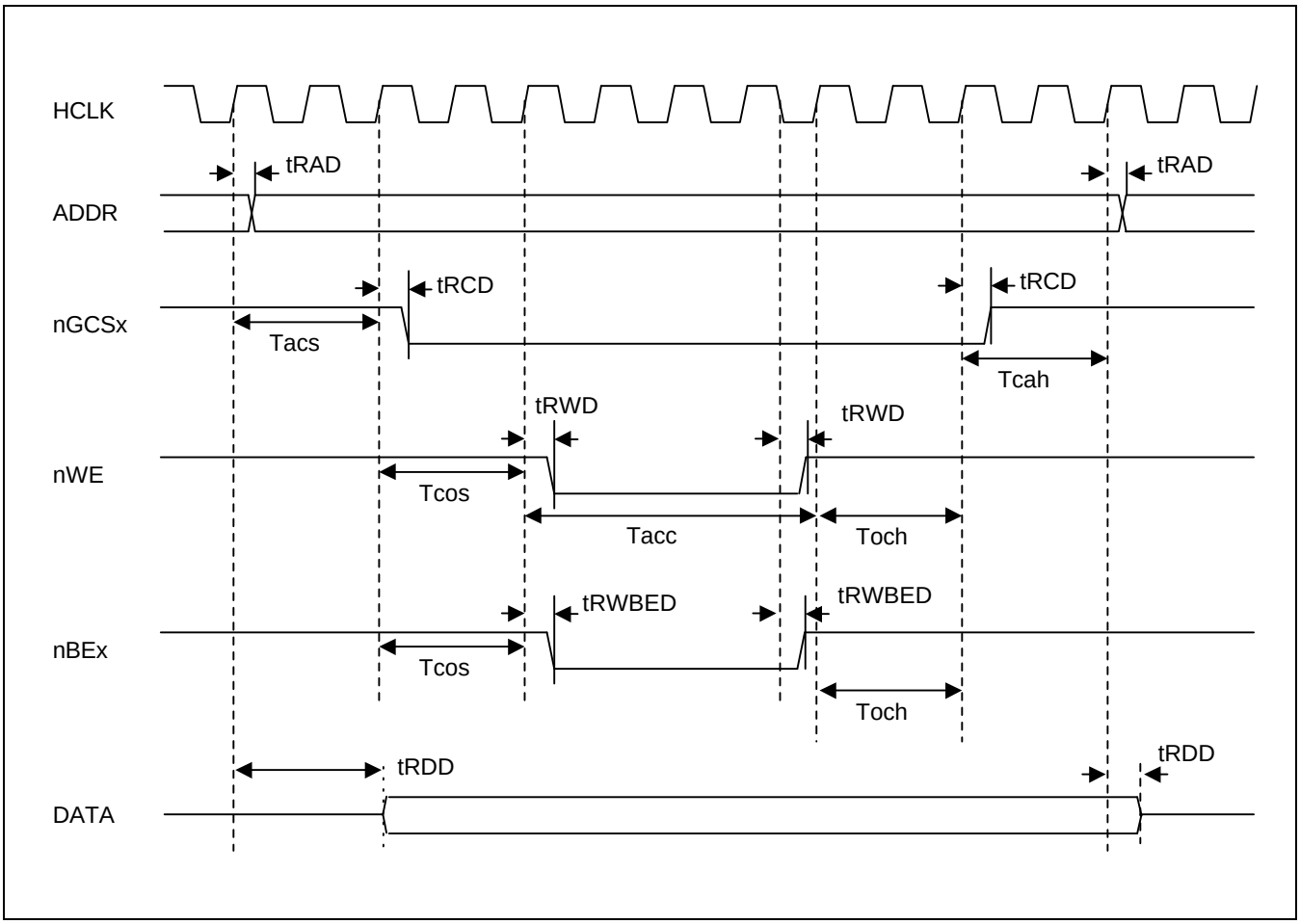


Figure 23-13. ROM/SRAM WRITE Timing (I)
(Tacs=2,Tcos=2,Tacc=4,Toch=2, Tcah=2, PMC=0, ST=0

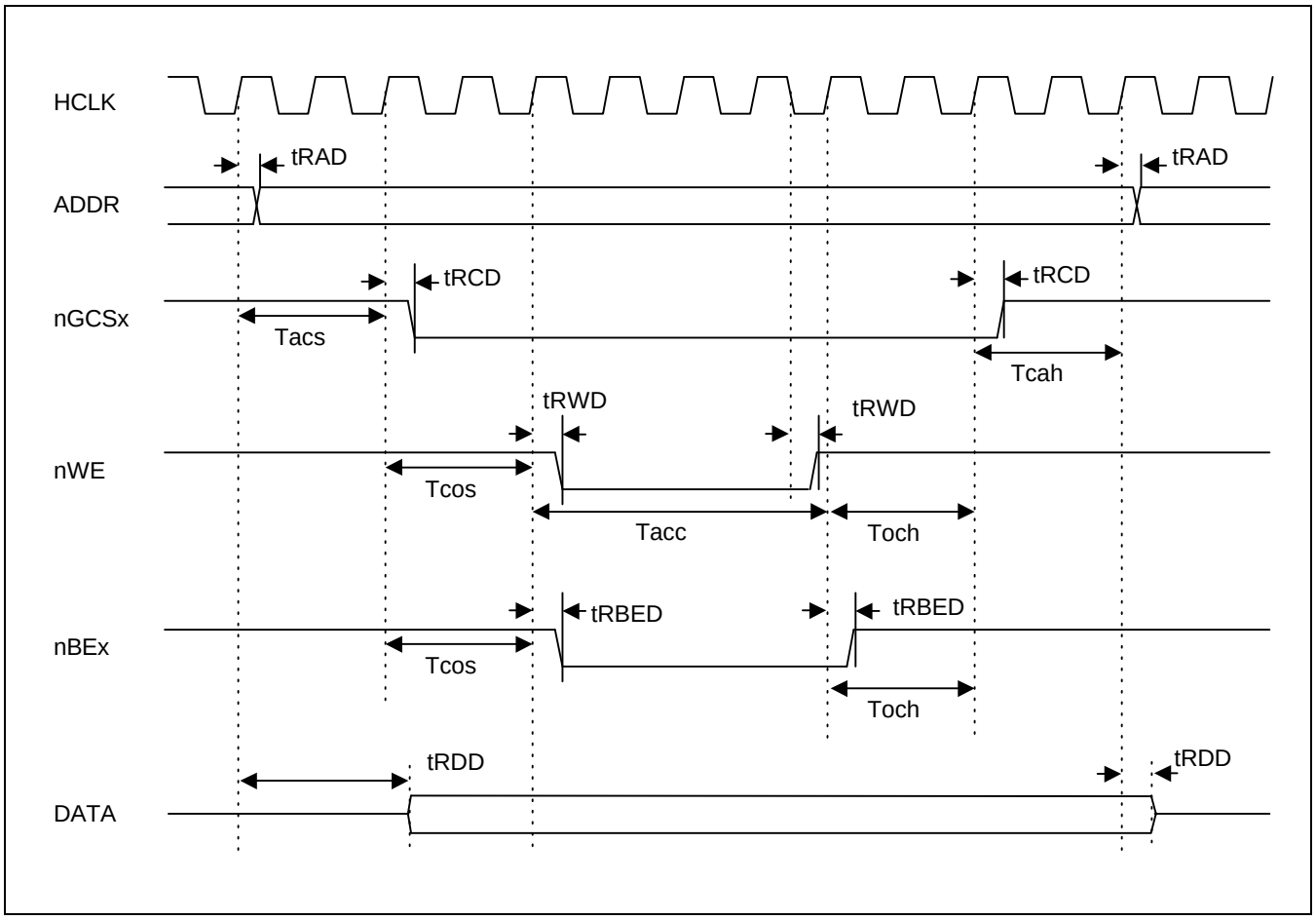
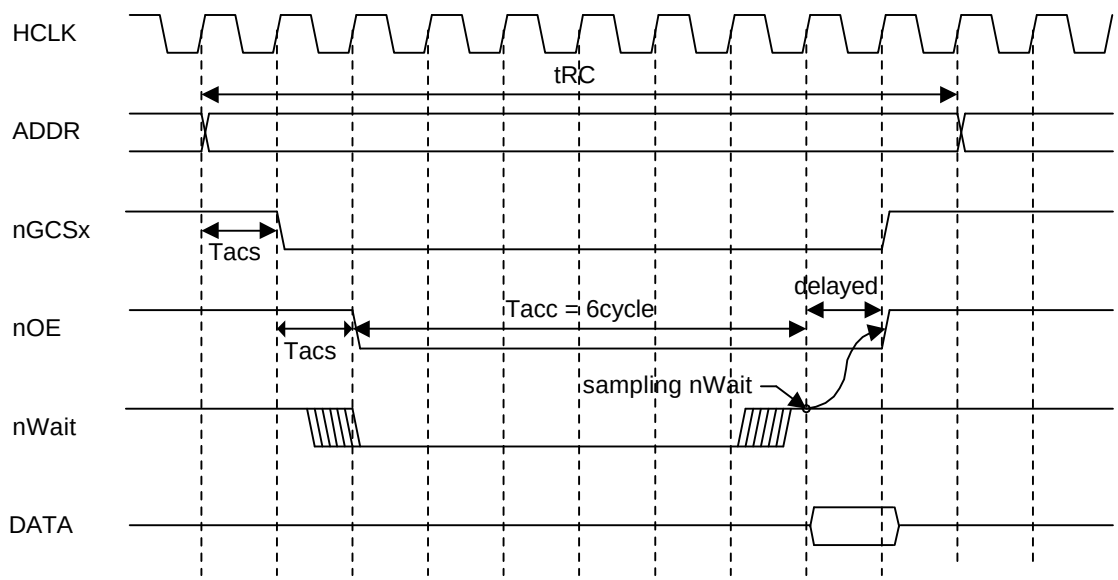


Figure 23-14. ROM/SRAM WRITE Timing (II)
(Tacs=2, Tcos=2, Tacc=4, Toch=2, Tcah=2, PMC=0, ST=1)



NOTE : The status of nWait is checked at (Tacc-1) cycle.

Figure 23-15. External nWAIT READ Timing (Tacs=0, Tcos=0, Tacc=6, Toch=0, Tcah=0, PMC=0, ST=0)

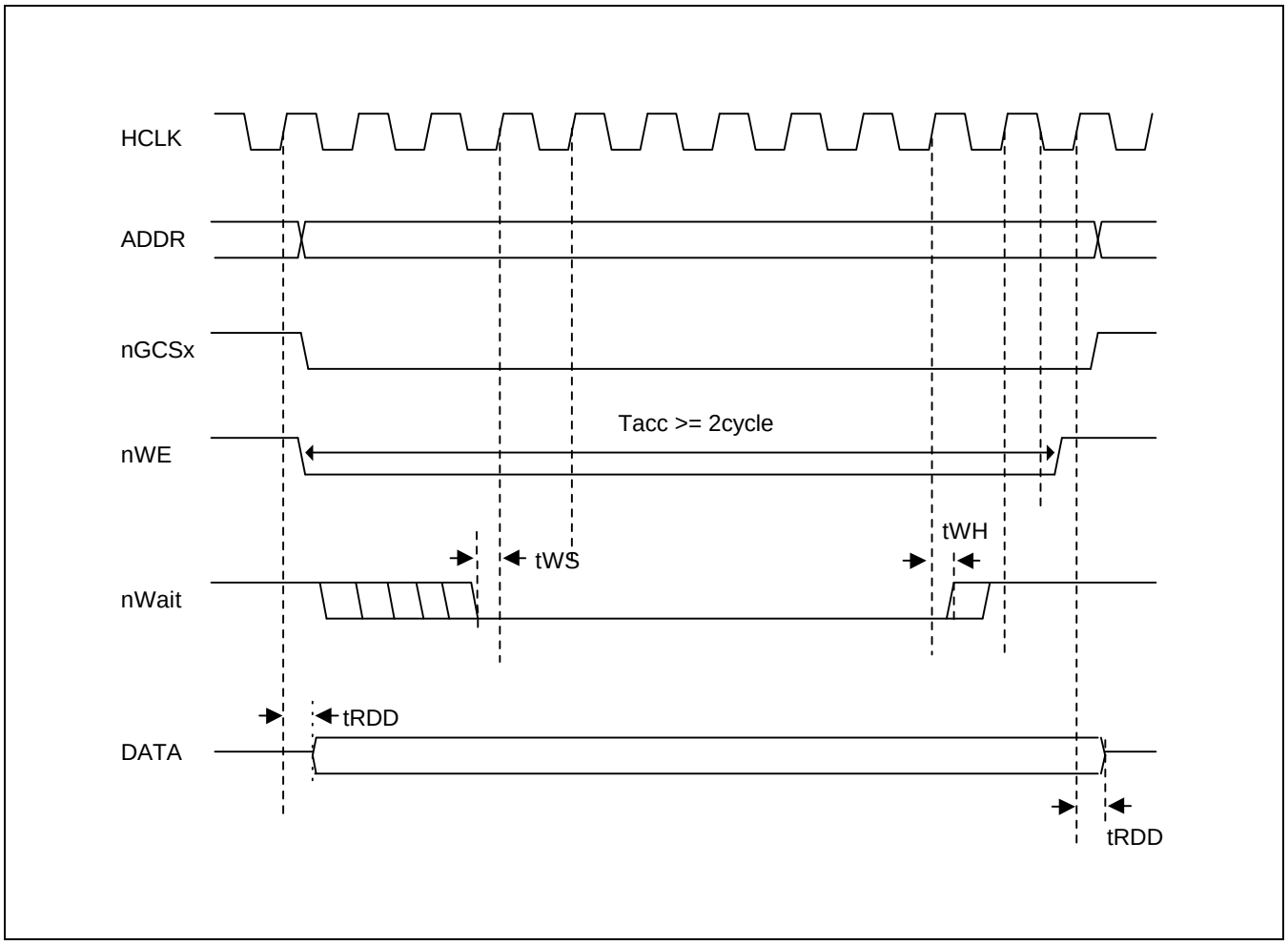


Figure 23-16. External nWAIT WRITE Timing (Tacs=0, Tcos=0, Tacc=4, Toch=0, Tcah=0, PMC=0, ST=0)

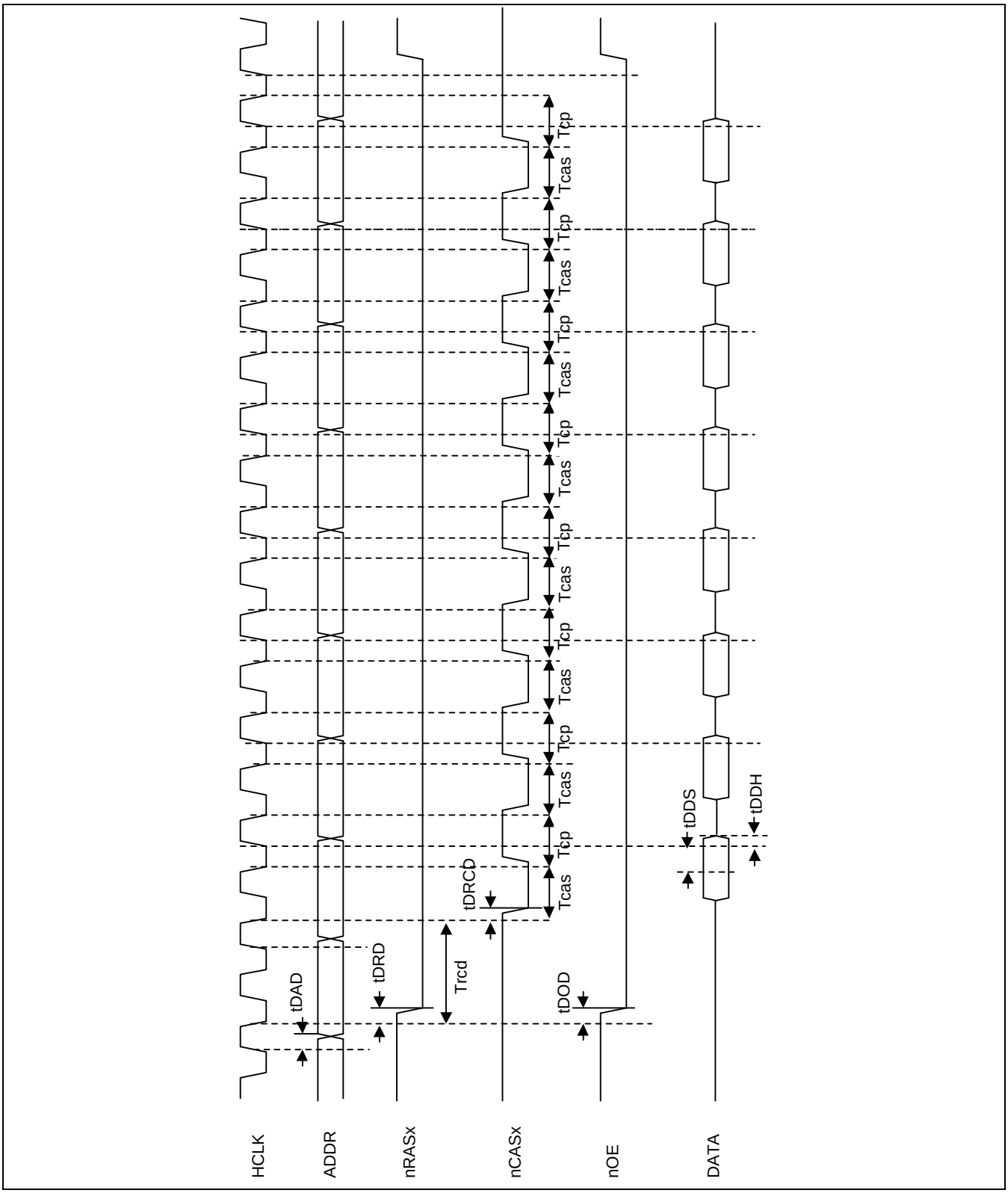


Figure 23-17. DRAM (EDO) Burst READ Timing
(Trcd=2, Tcas=1, Tcp=1, Trp=3.5, MT=10, DW = 16bit)

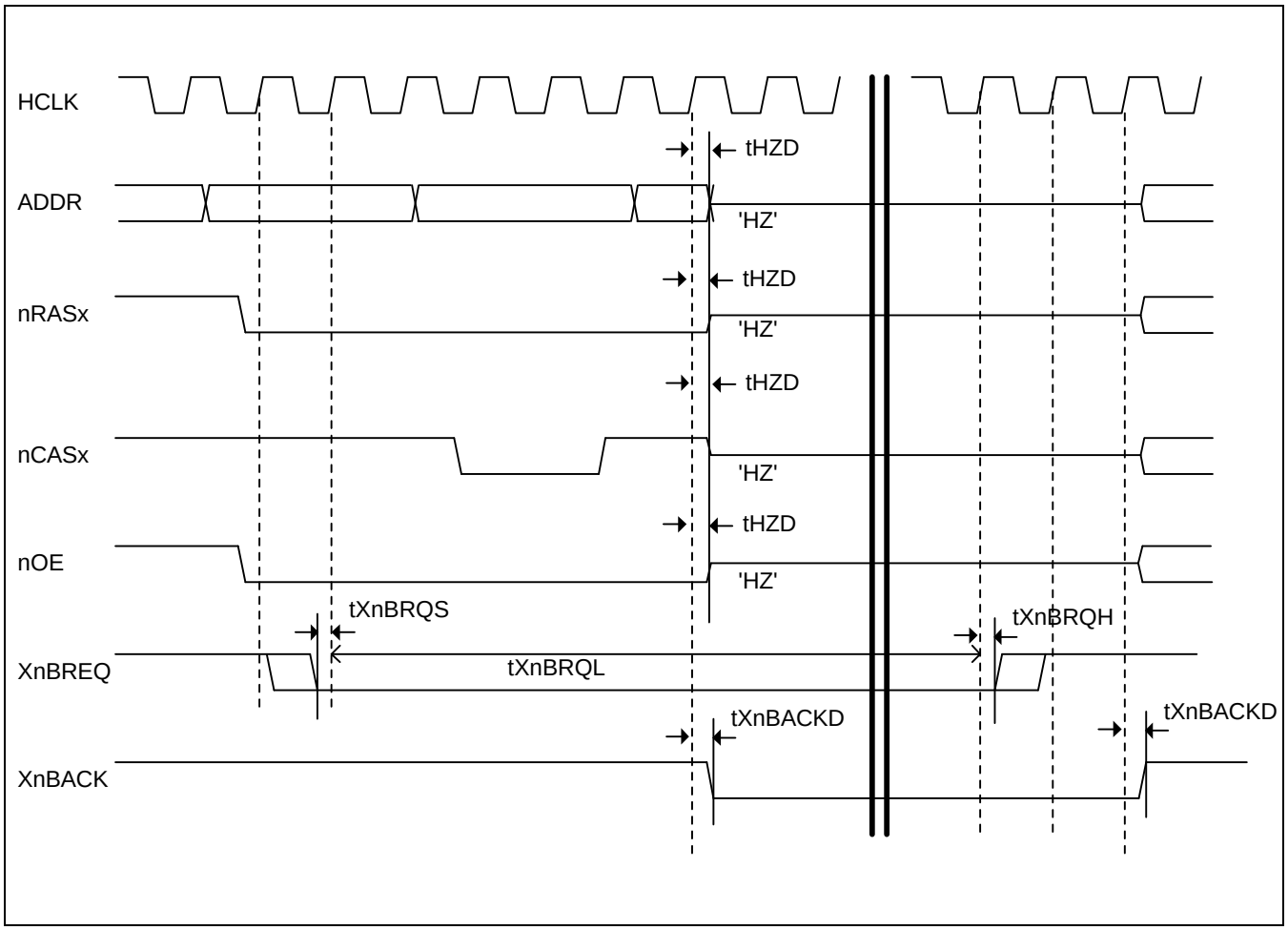


Figure 23-18. External Bus Request in DRAM Cycle (Trcd=3, Tcas=2, Tcp=1, Trp=4.5)

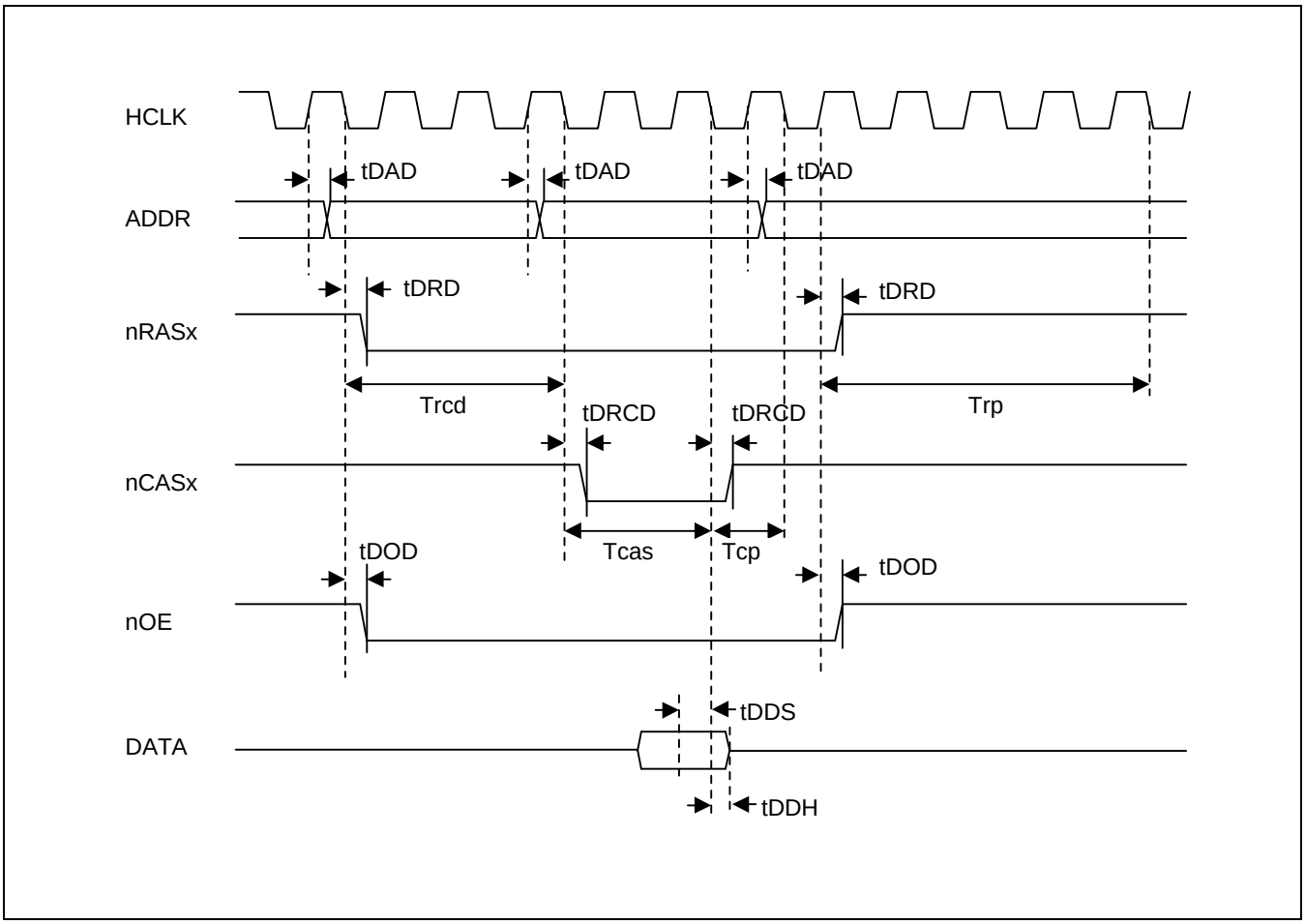


Figure 23-19. DRAM(FP) Single READ Timing (Trcd=3, Tcas=2, Tcp=1, Trp=4.5, MT=01)

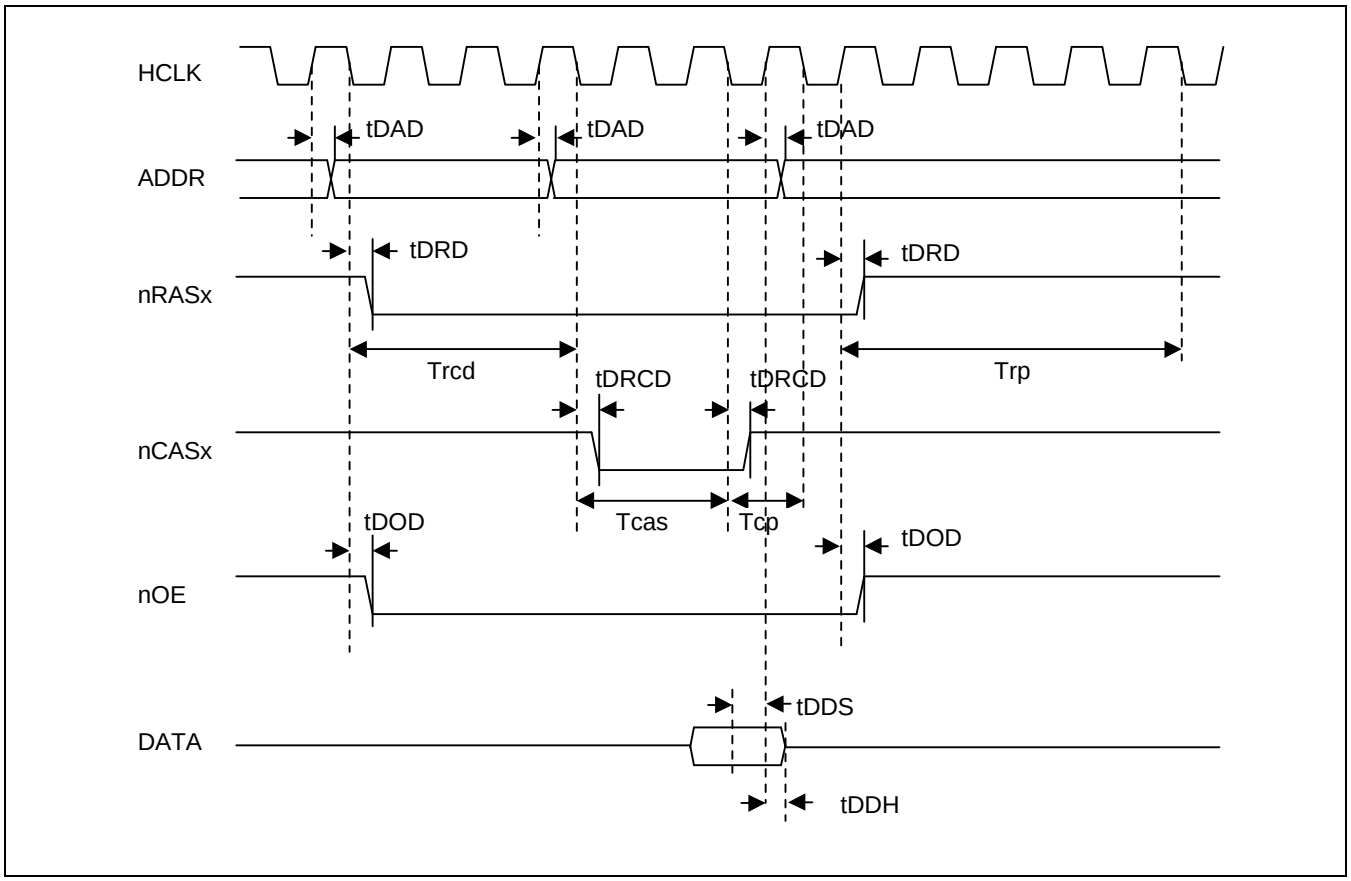


Figure 23-20. DRAM(EDO) Single READ Timing (Trcd=3, Tcas=2, Tcp=1, Trp=4.5, MT=10)

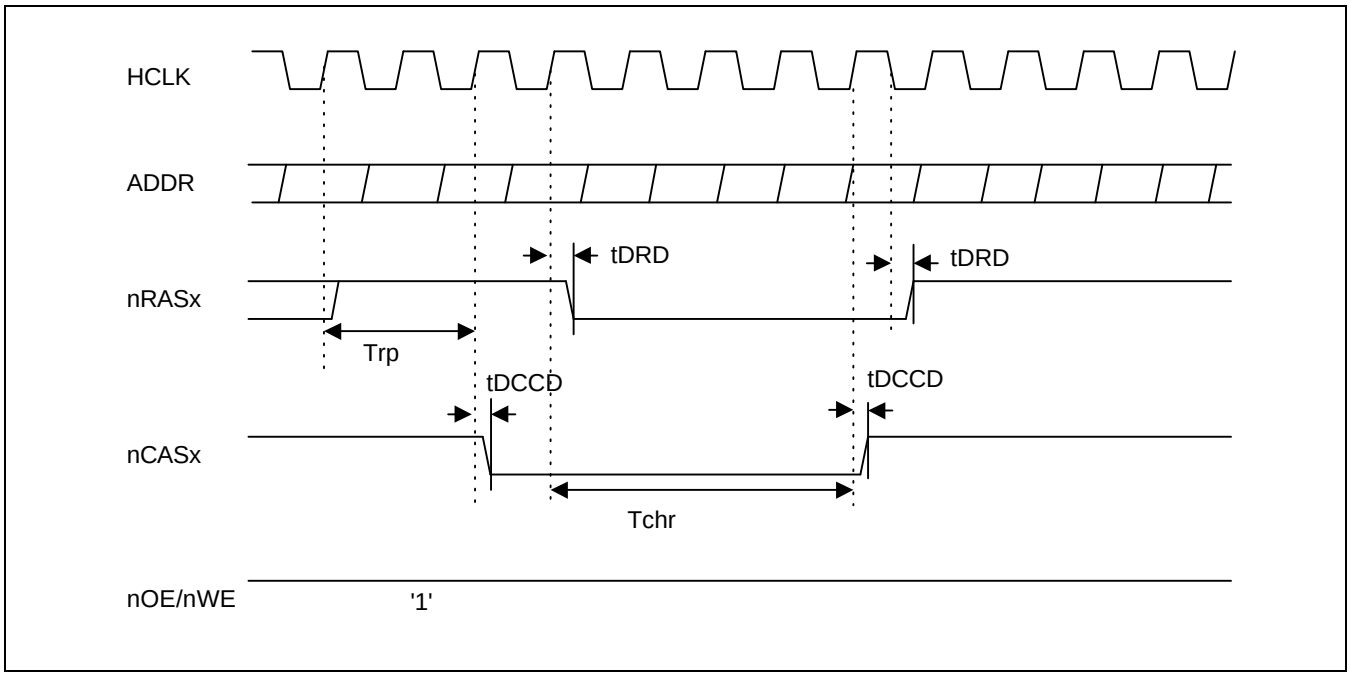


Figure 23-21. DRAM CBR Refresh Timing (Tchr=4)

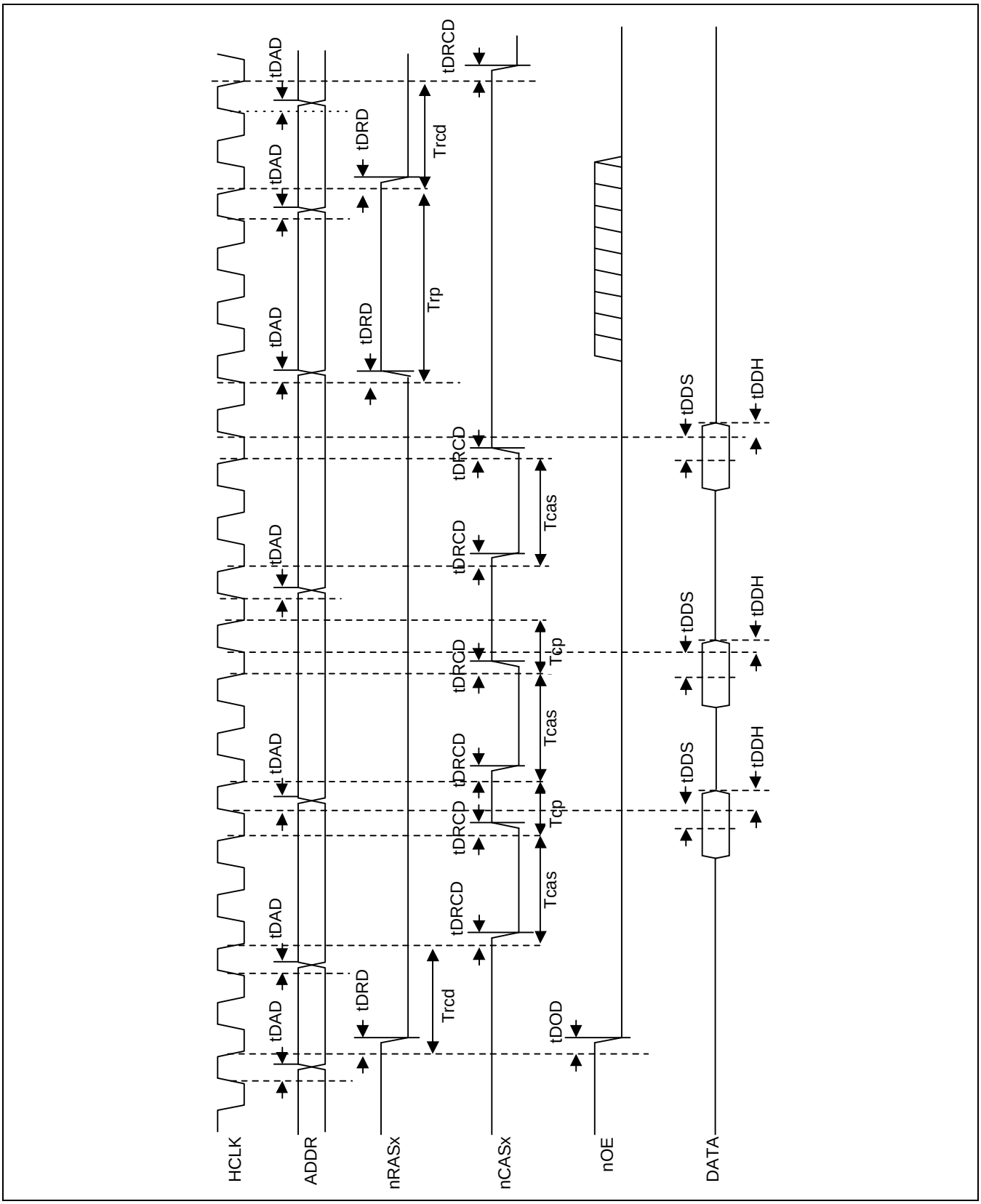


Figure 23-22. DRAM(EDO) Page Hit-Miss READ Timing (Trcd=2, Tcas=2, Tcp=1, Trp=3.5, MT=10)

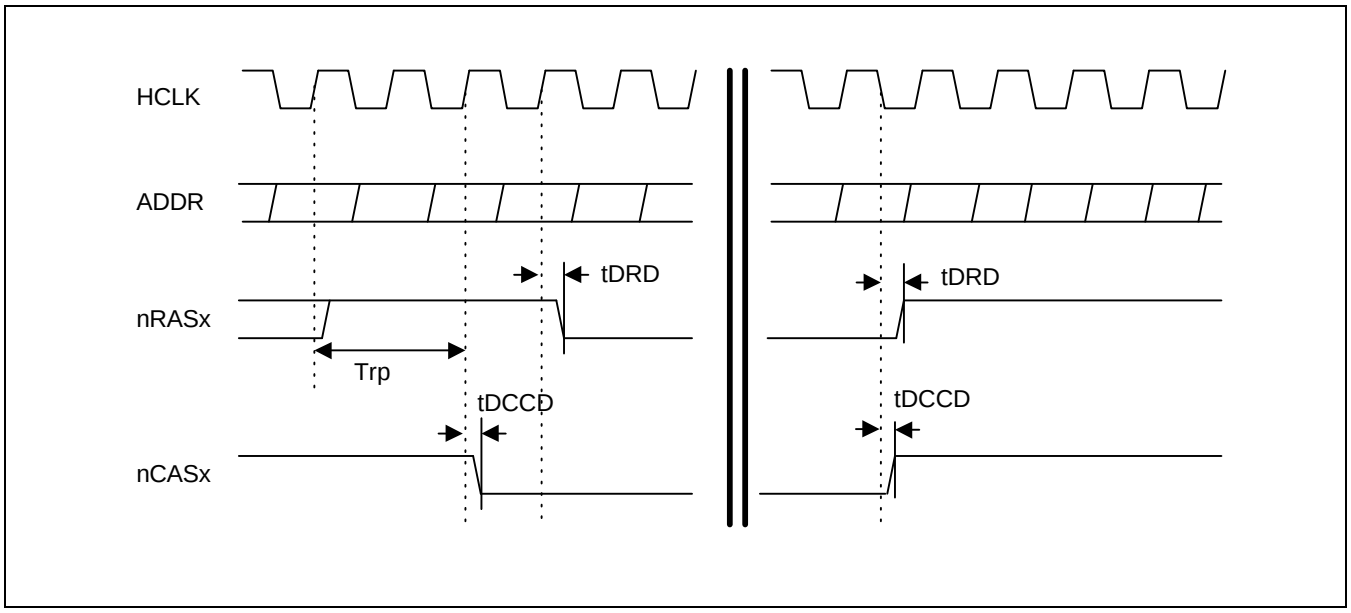


Figure 23-23. DRAM Self Refresh Timing

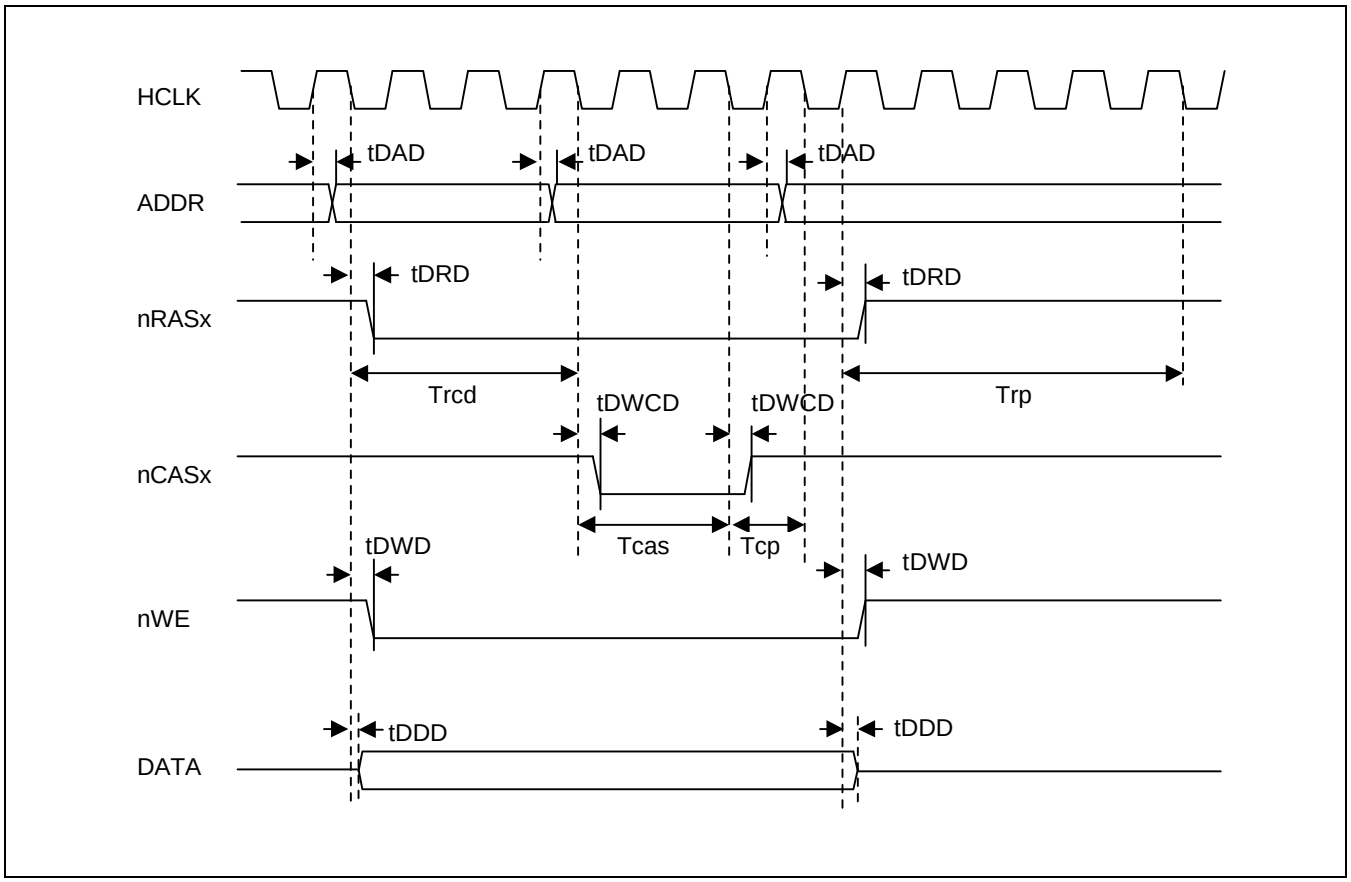


Figure 23-24. DRAM(FP/EDO) Single Write Timing (Trcd=3, Tcas=2, Tcp=1, Trp=4.5, MT=01/10)

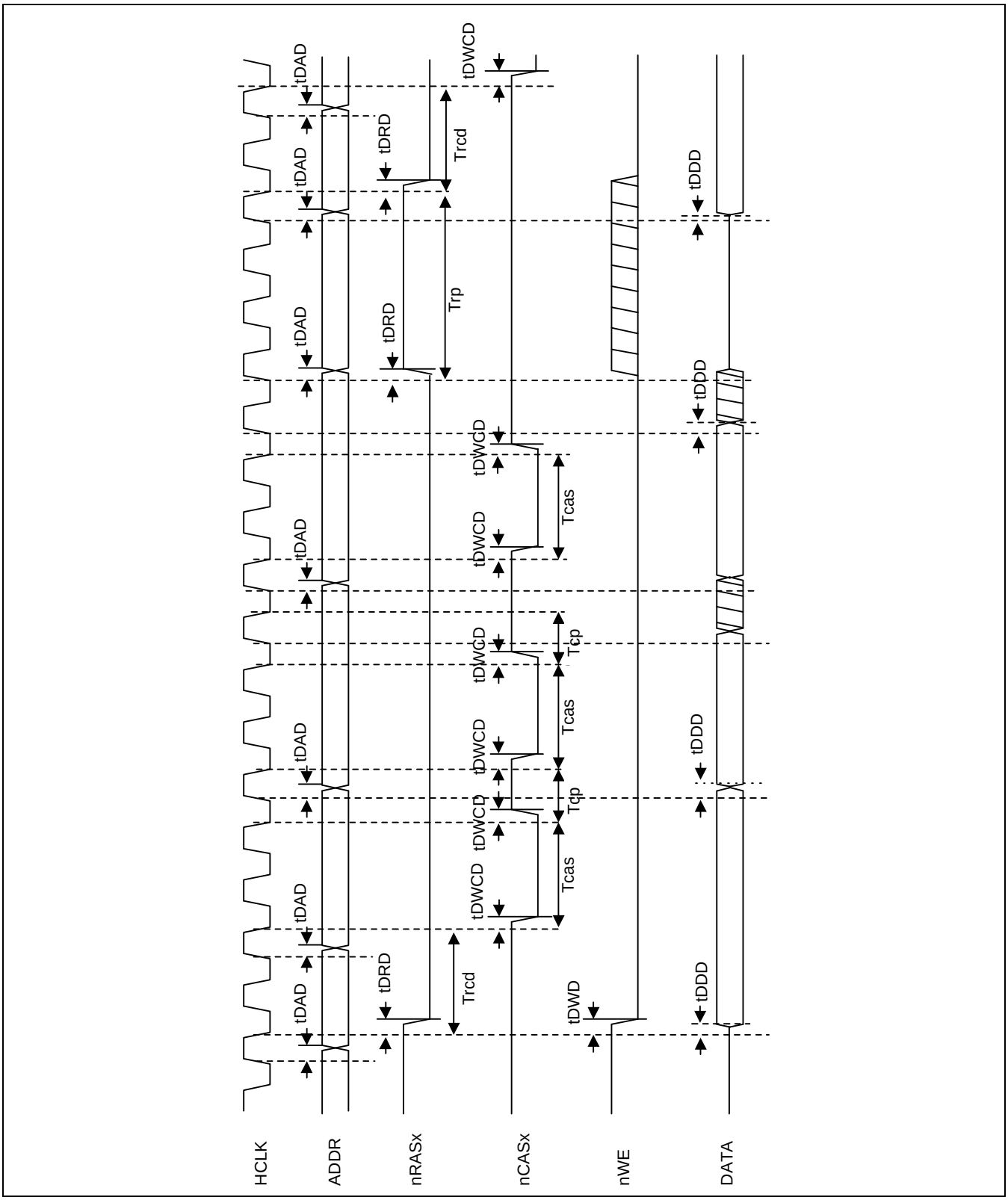


Figure 23-25. DRAM(FP/EDO) Page Hit-Miss Write Timing (Trcd=2, Tcas=2, Tcp=1, Trp=3.5, MT=01/10)

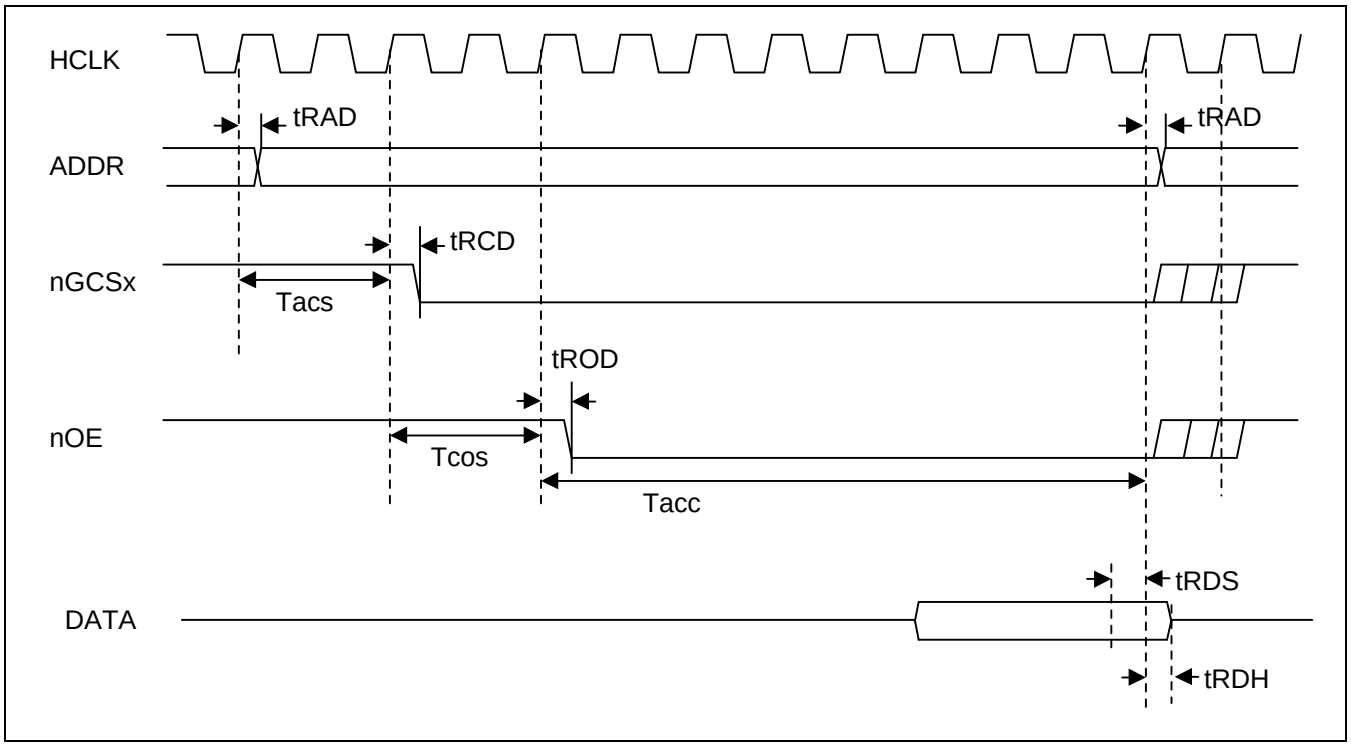


Figure 23-26. Masked-ROM Single READ Timing (Tacs=2, Tcos=2, Tacc=8, PMC=01/10/11)

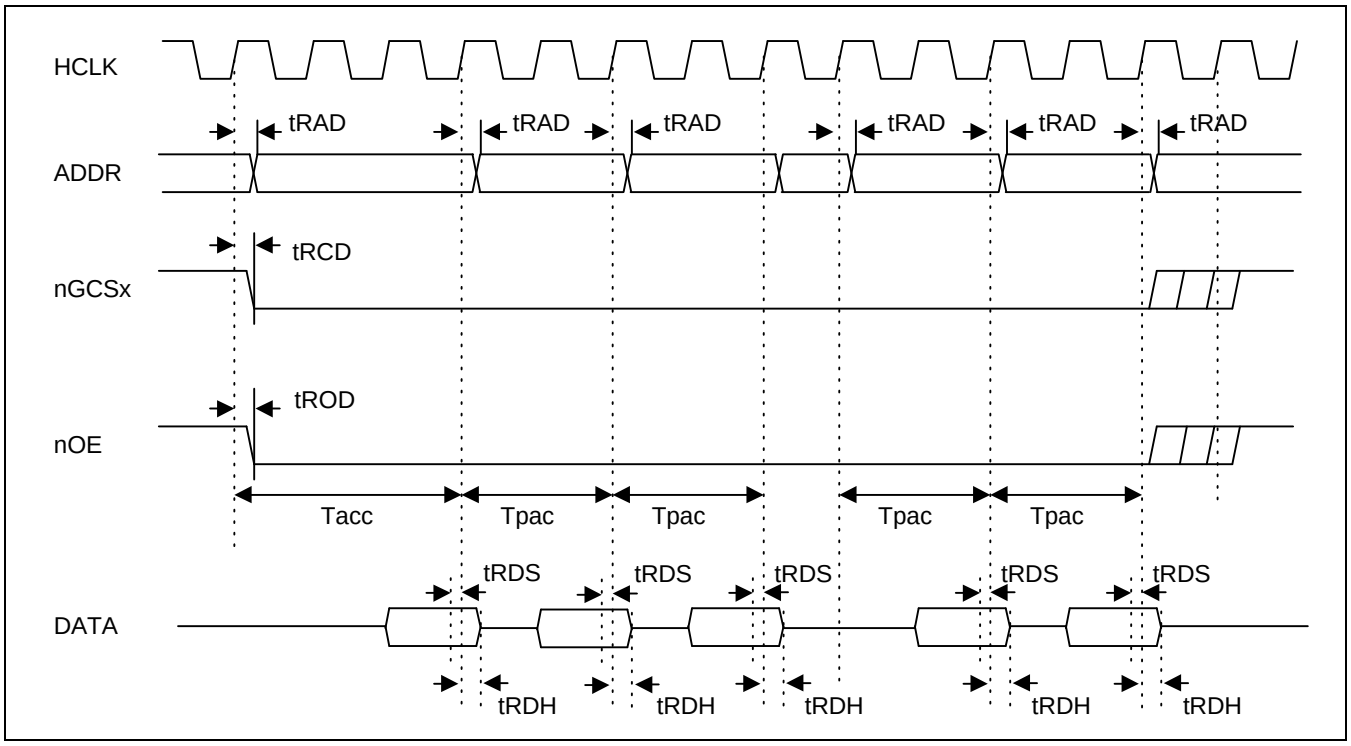


Figure 23-27. Masked-ROM Consecutive READ Timing (Tacs=0, Tcos=0, Tacc=3, Tpac=2, PMC=01/10/11)

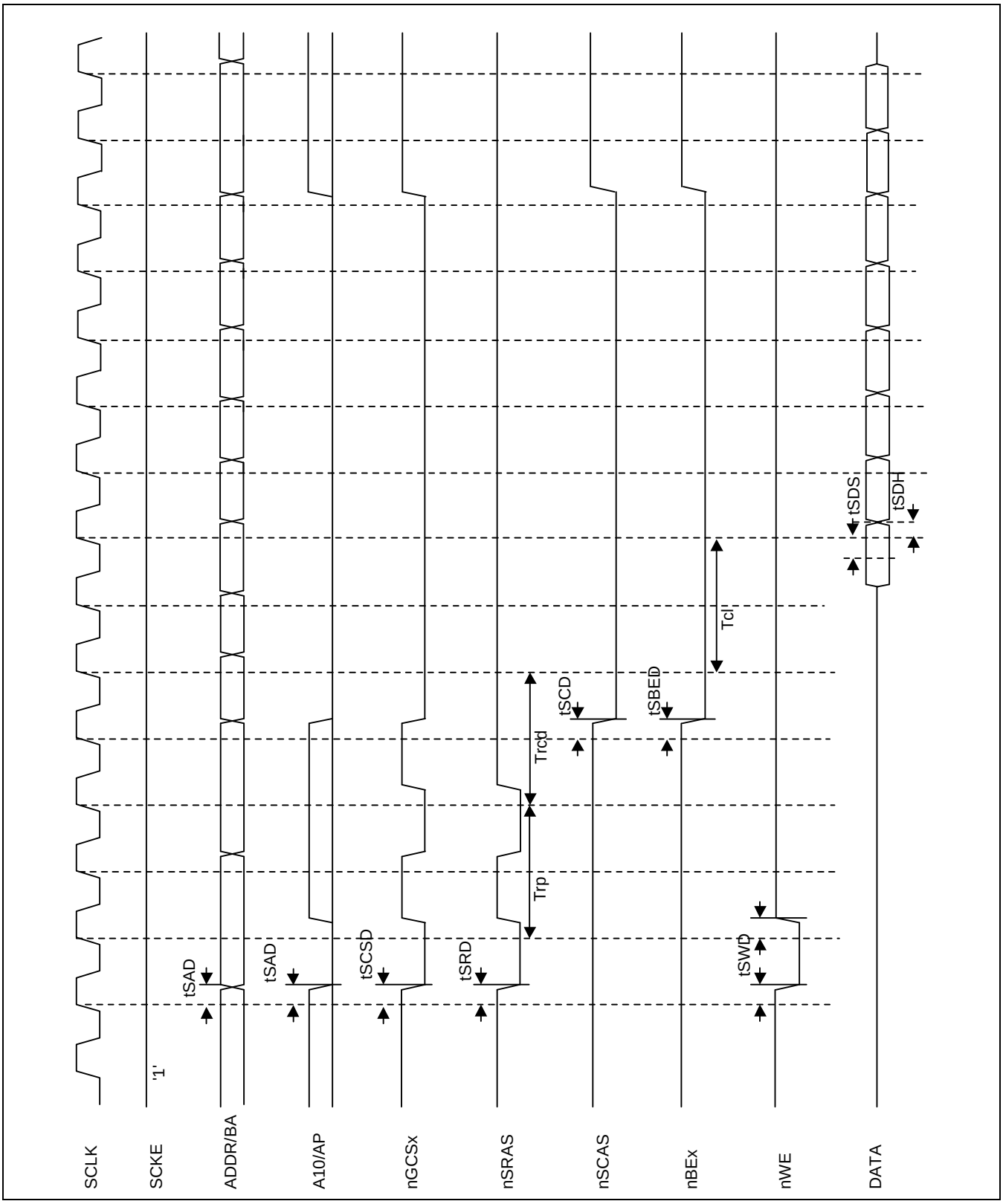


Figure 23-28. SDRAM Single Burst READ Timing (Trp=2, Trcd=2, Tcl=2, DW=16bit)

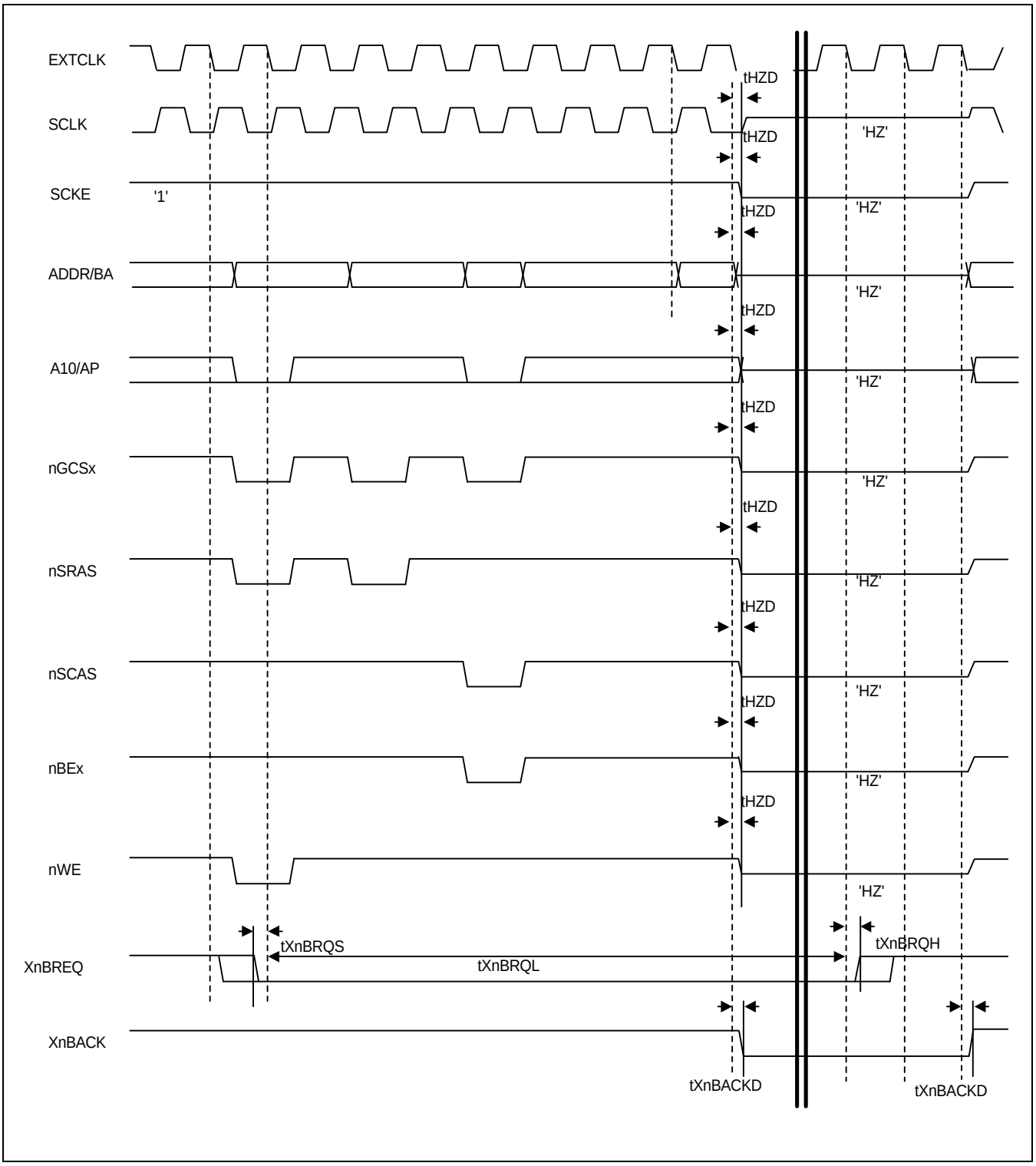


Figure 23-29. External Bus Request in SDRAM Timing (Trp=2, Trcd=2, Tcl=2)

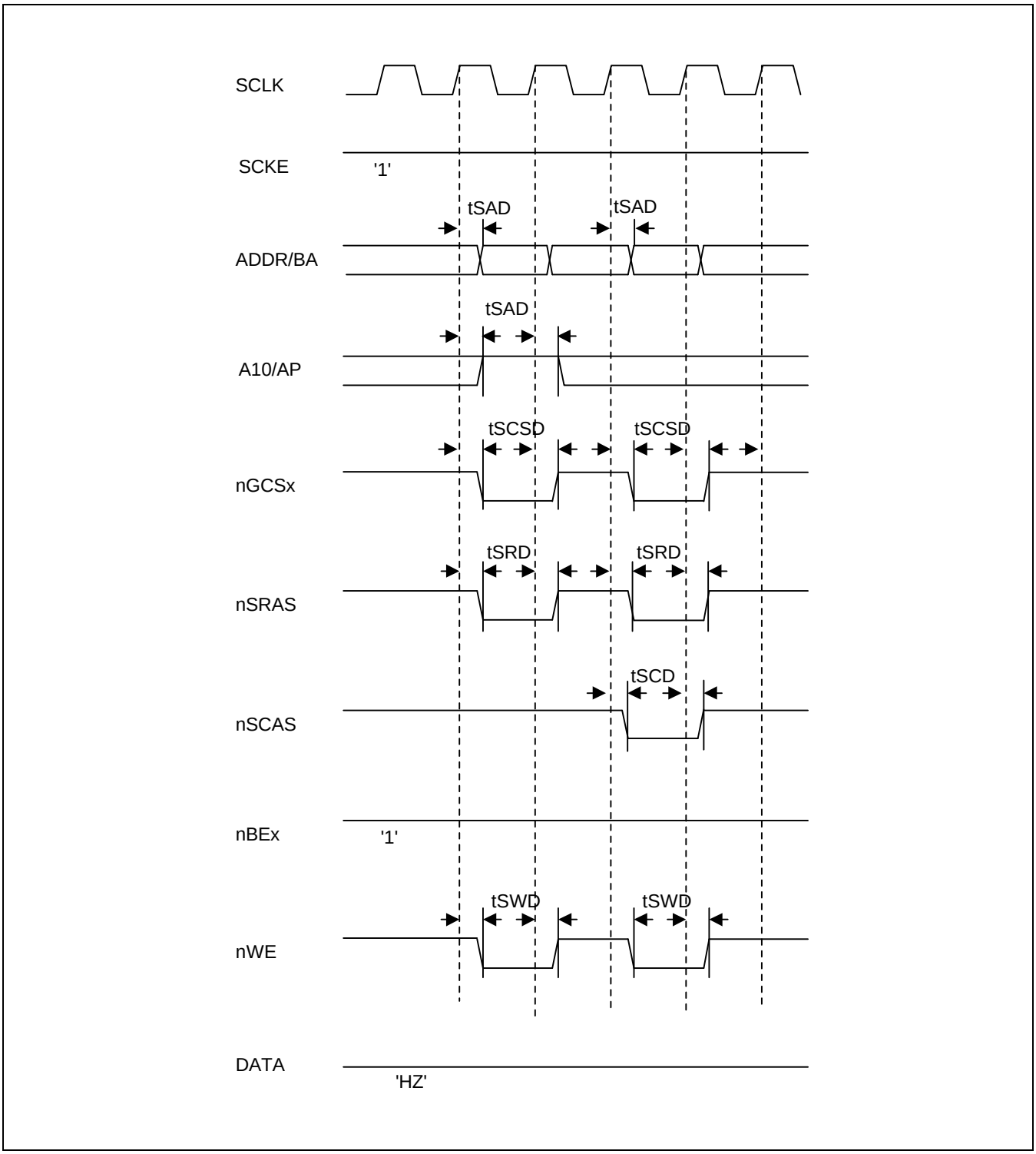


Figure 23-30. SDRAM MRS Timing

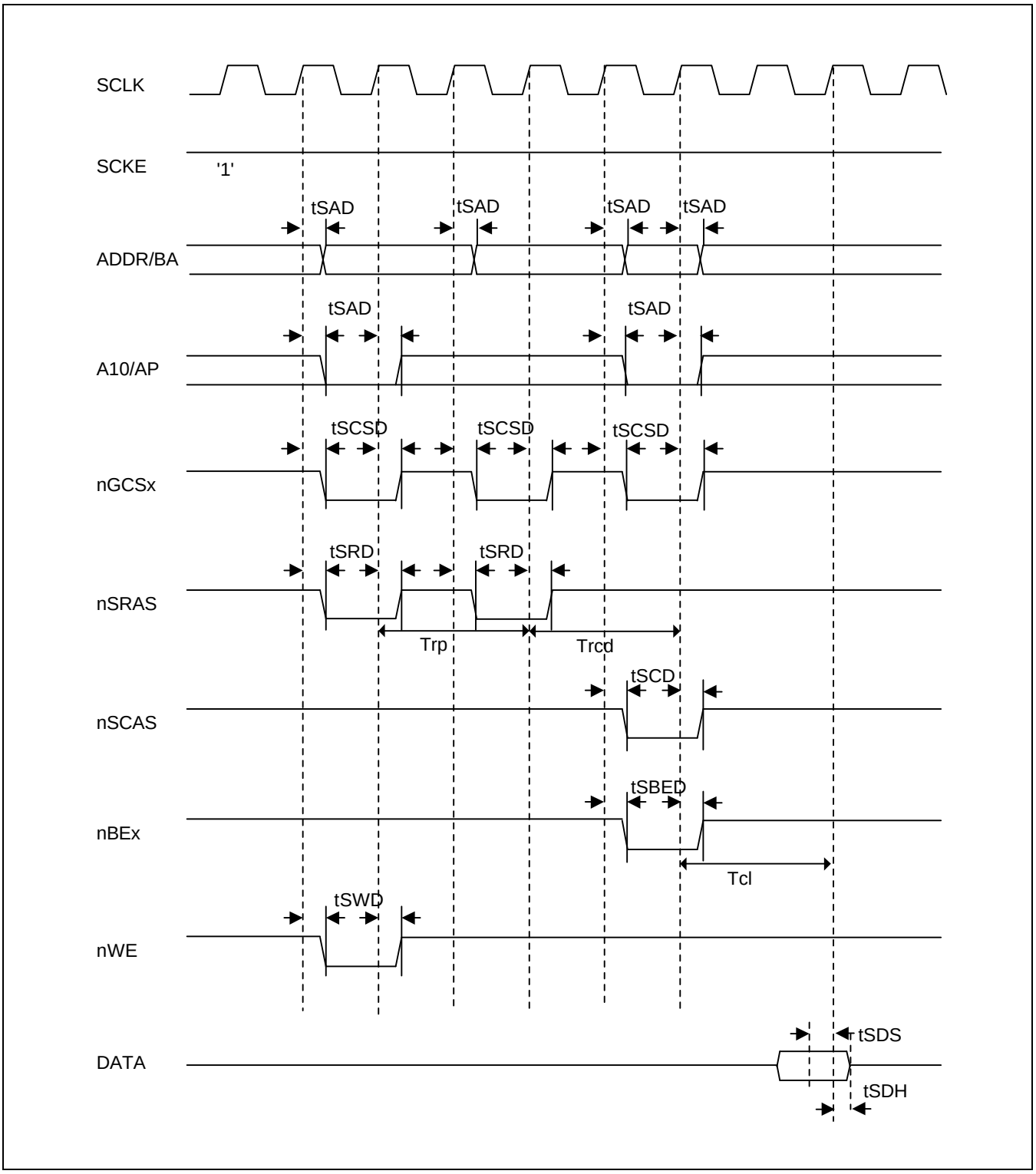


Figure 23-31. SDRAM Single READ Timing(I) (Trp=2, Trcd=2, Tcl=2)

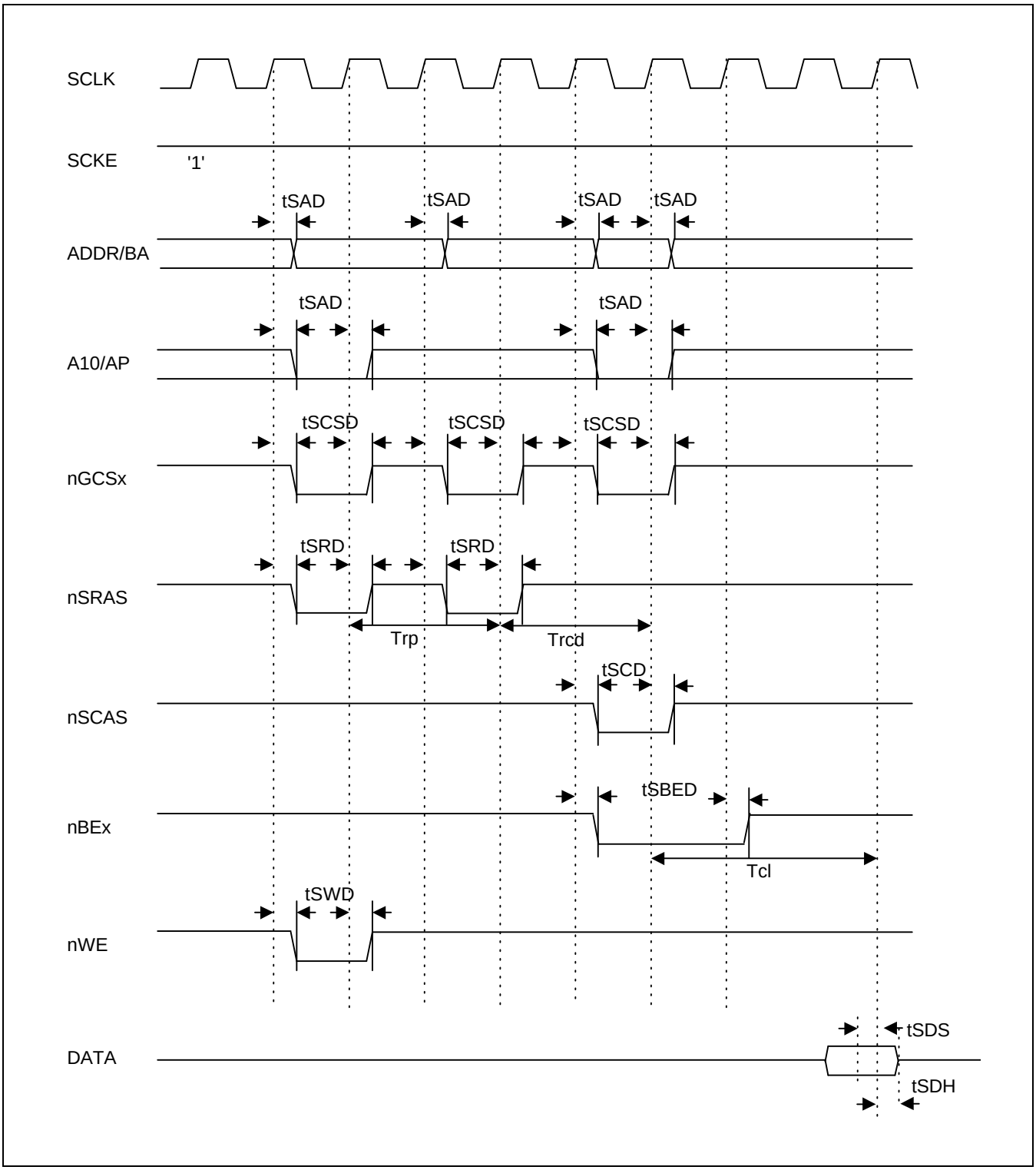
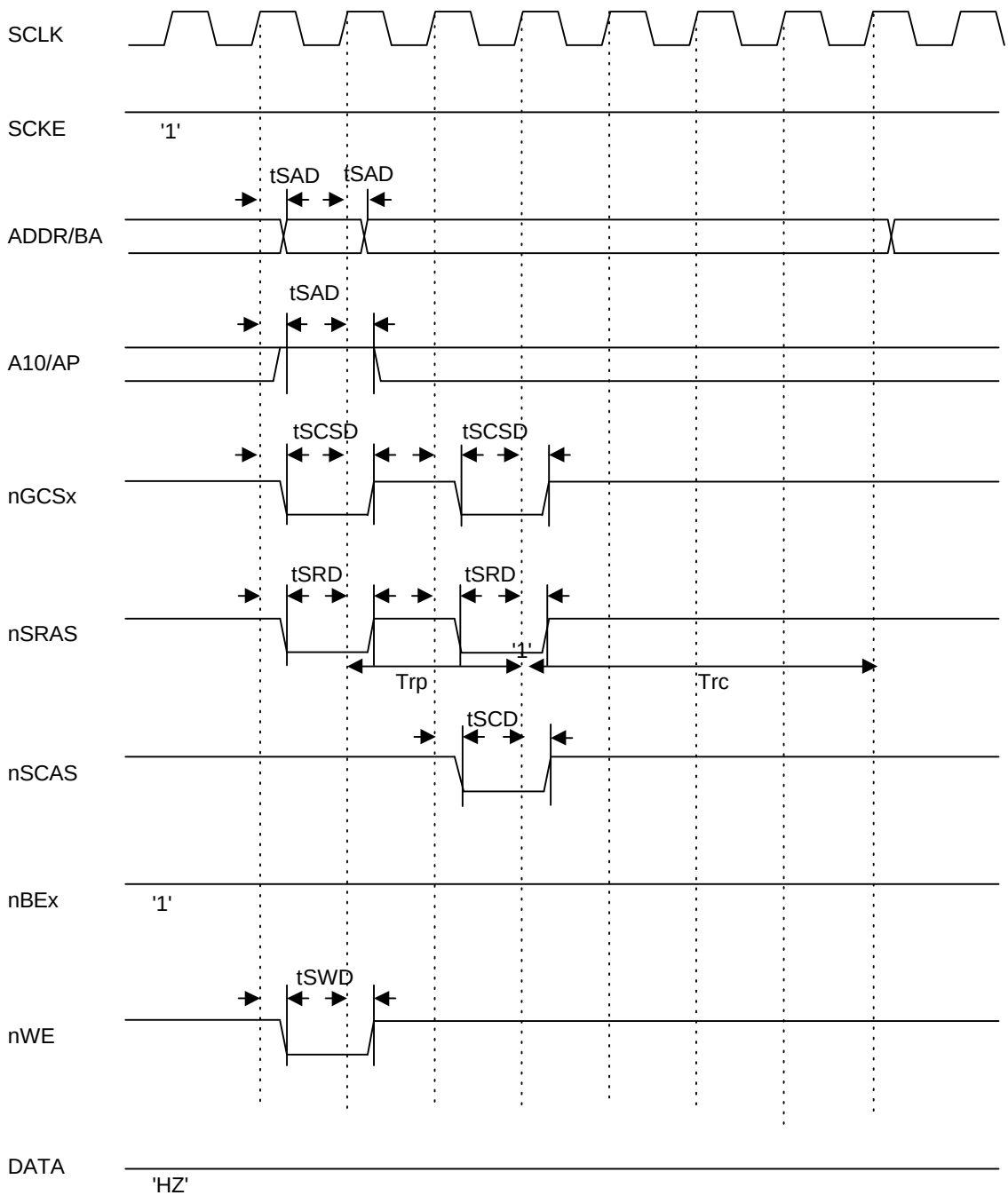


Figure 23-32. SDRAM Single READ Timing(II) (Trp=2, Trcd=2, Tcl=3)



NOTE: Before executing auto/self refresh command, all banks must be idle state.

Figure 23-33. SDRAM Auto Refresh Timing (Trp=2, Trc=4)

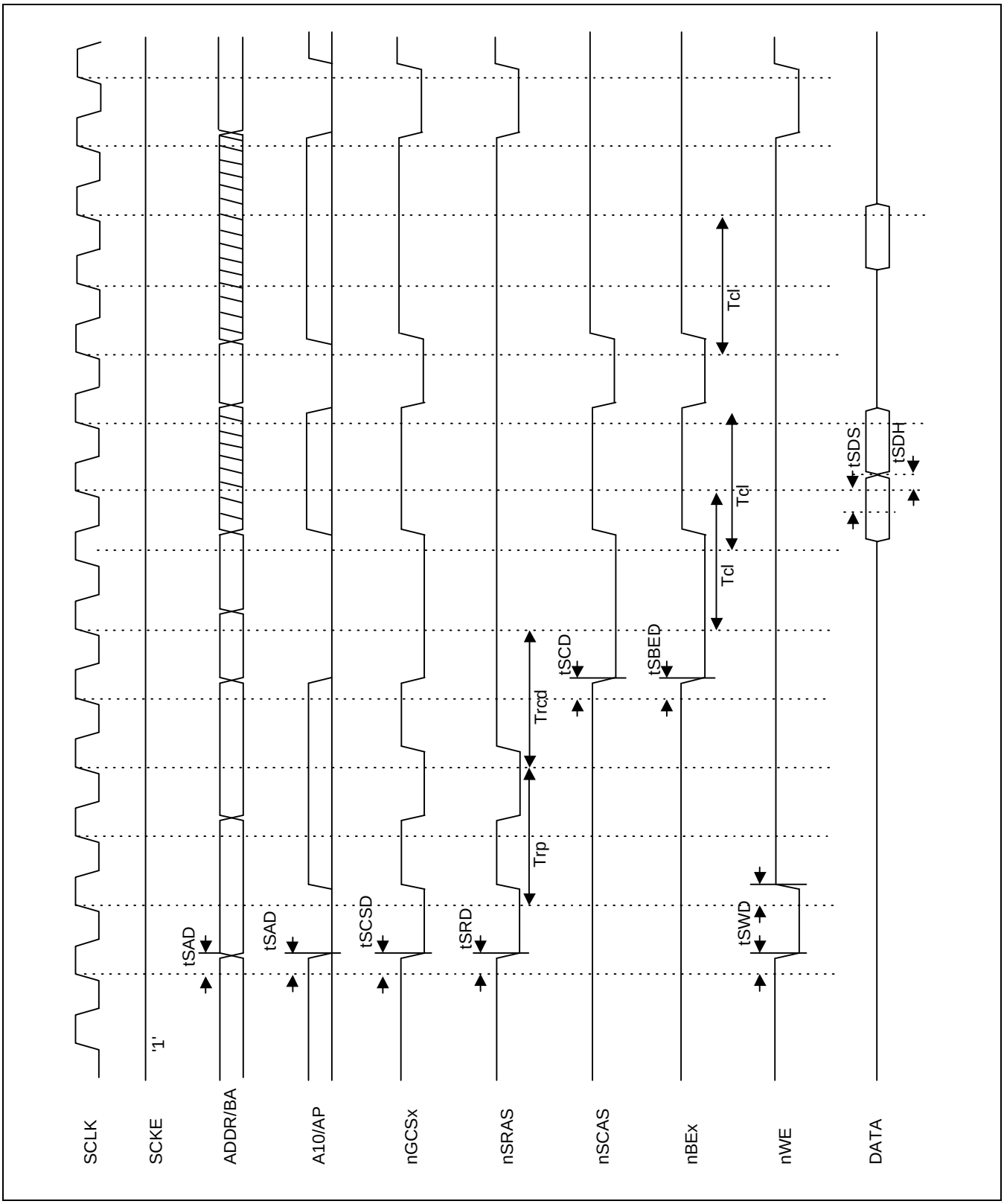
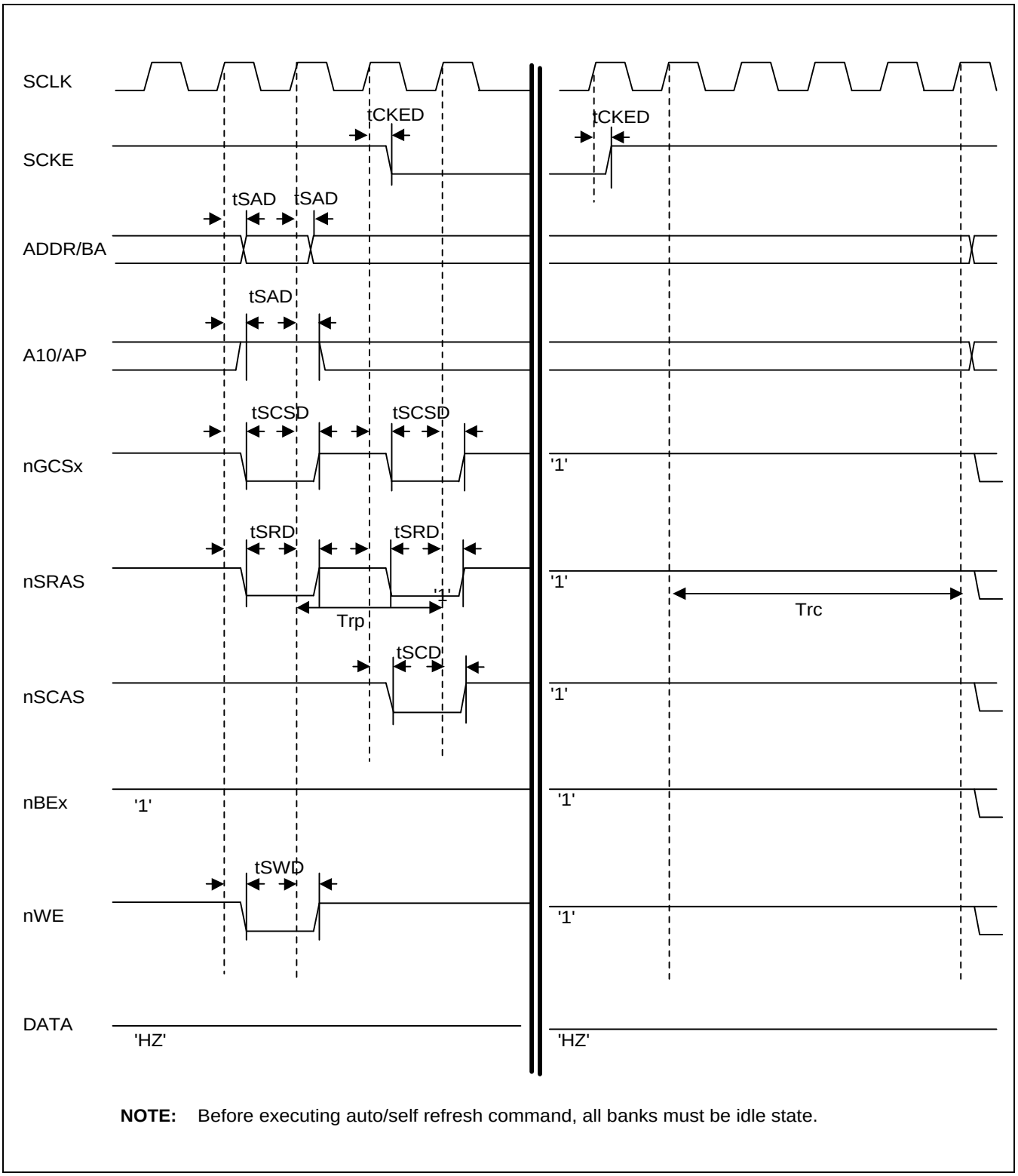


Figure 23-34. SDRAM Page Hit-Miss READ Timing (Trp=2, Trcd=2, Tcl=2)



NOTE: Before executing auto/self refresh command, all banks must be idle state.

Figure 23-35. SDRAM Self Refresh Timing (Trp=2, Trc=4)

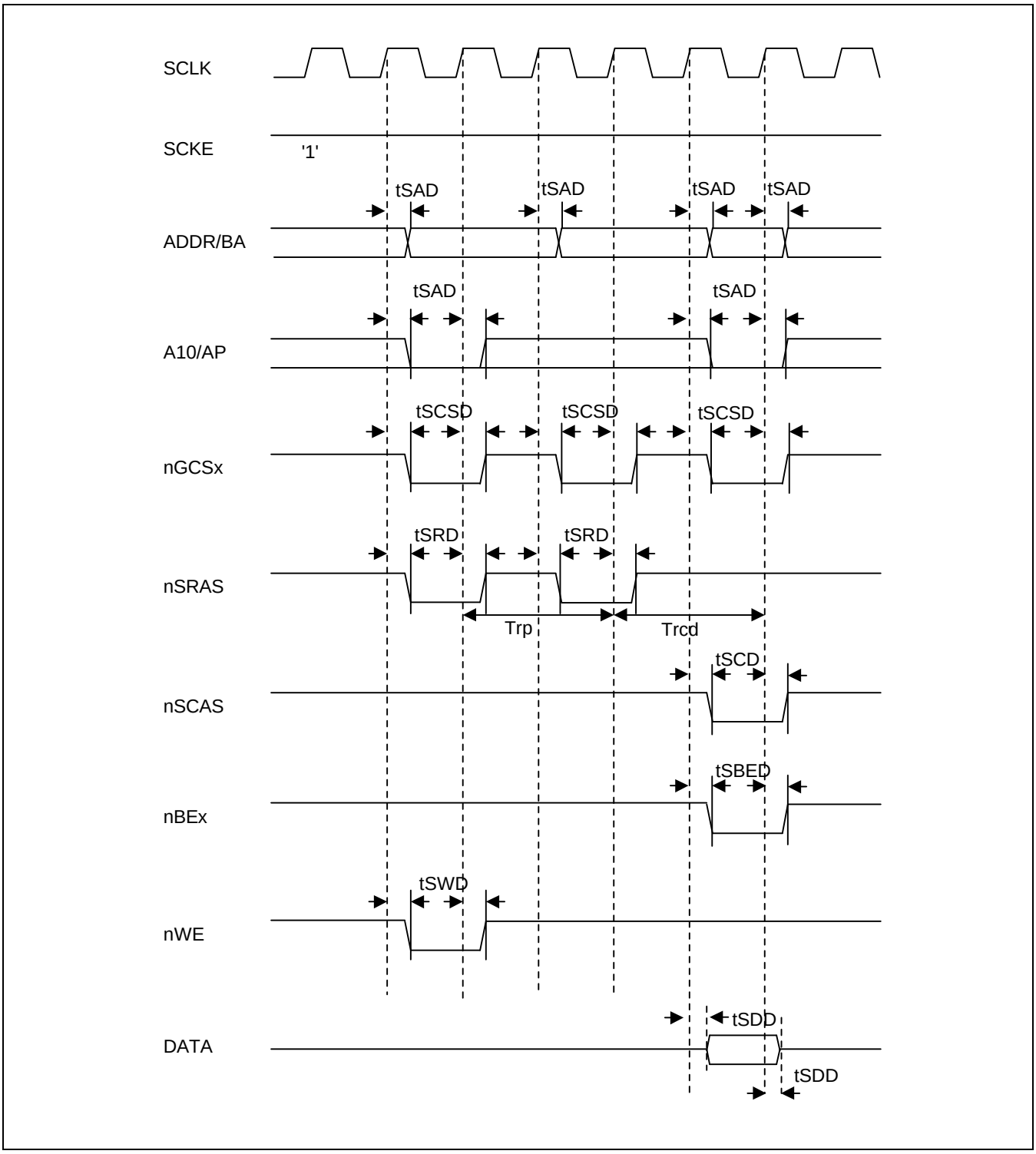


Figure 23-36. SDRAM Single Write Timing (Trp=2, Trcd=2)

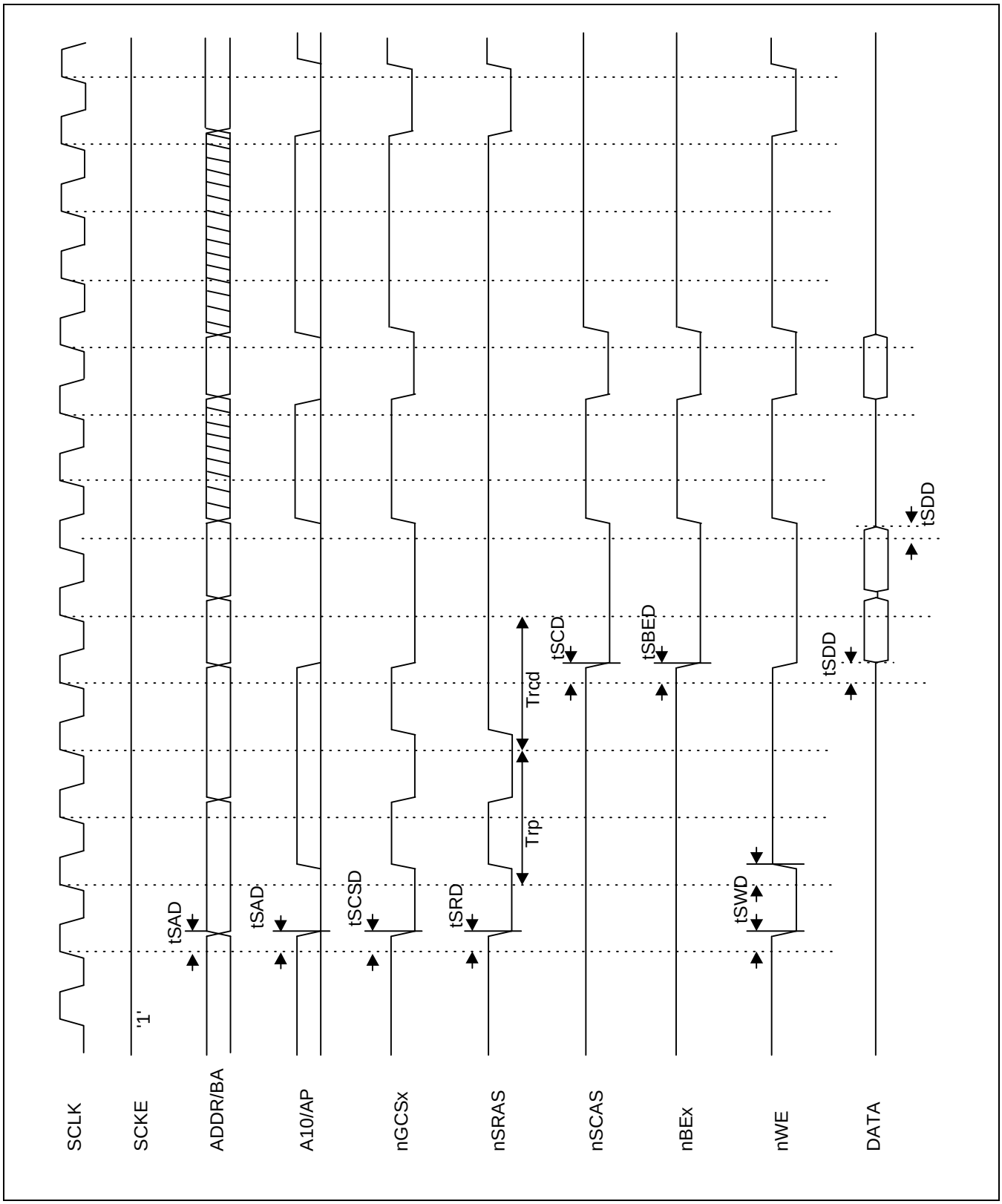


Figure 23-37. SDRAM Page Hit-Miss Write Timing (Trp=2, Trcd=2, Tcl=2)

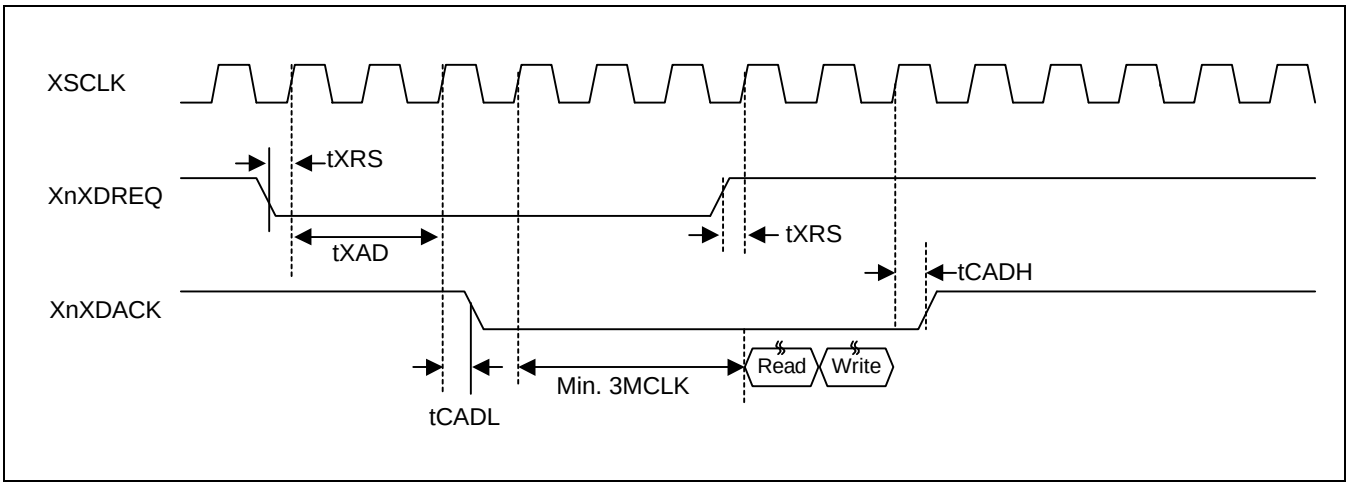


Figure 23-38. External DMA Timing (Handshake, Single transfer)

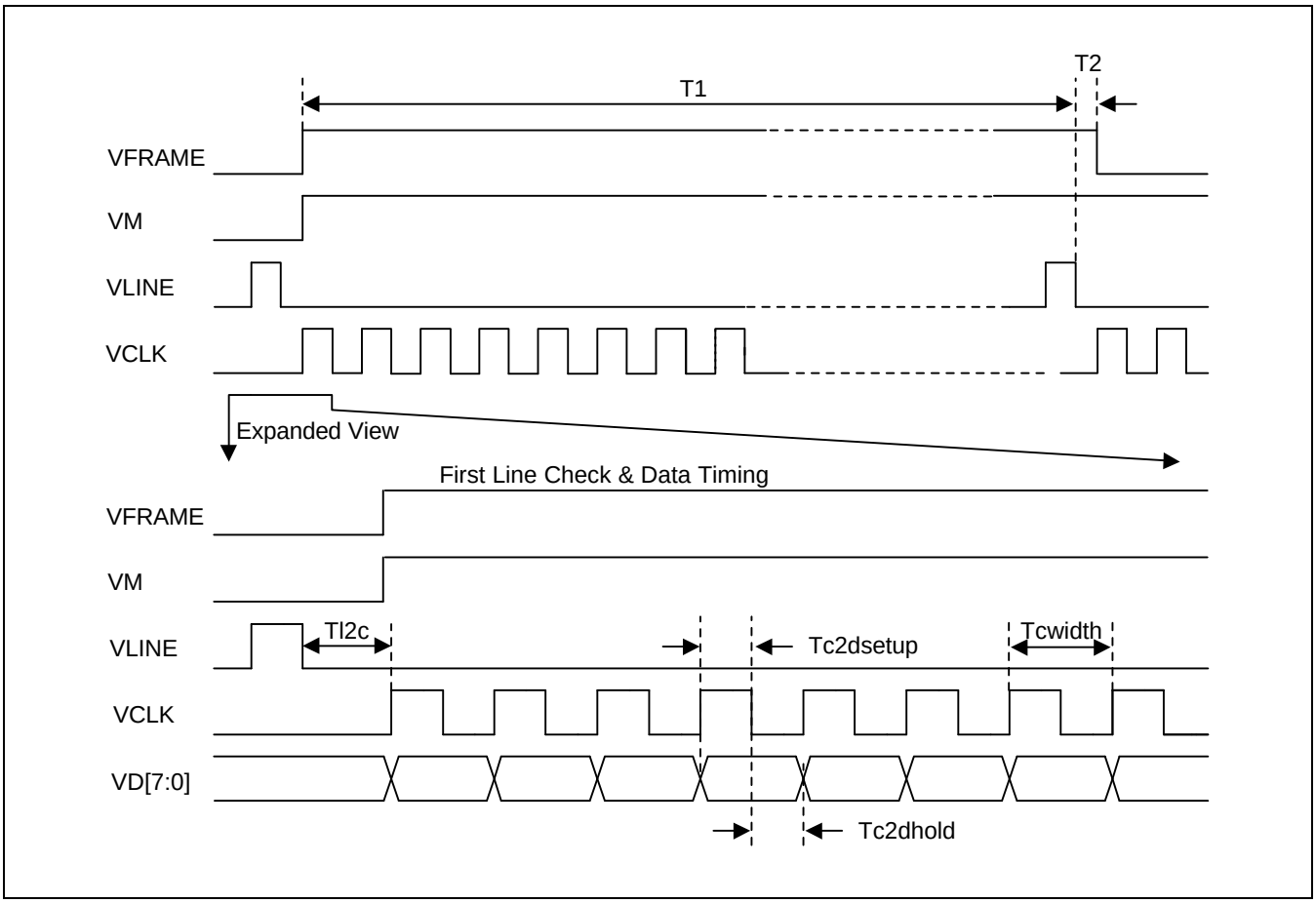


Figure 23-39. STN LCD Controller Timing

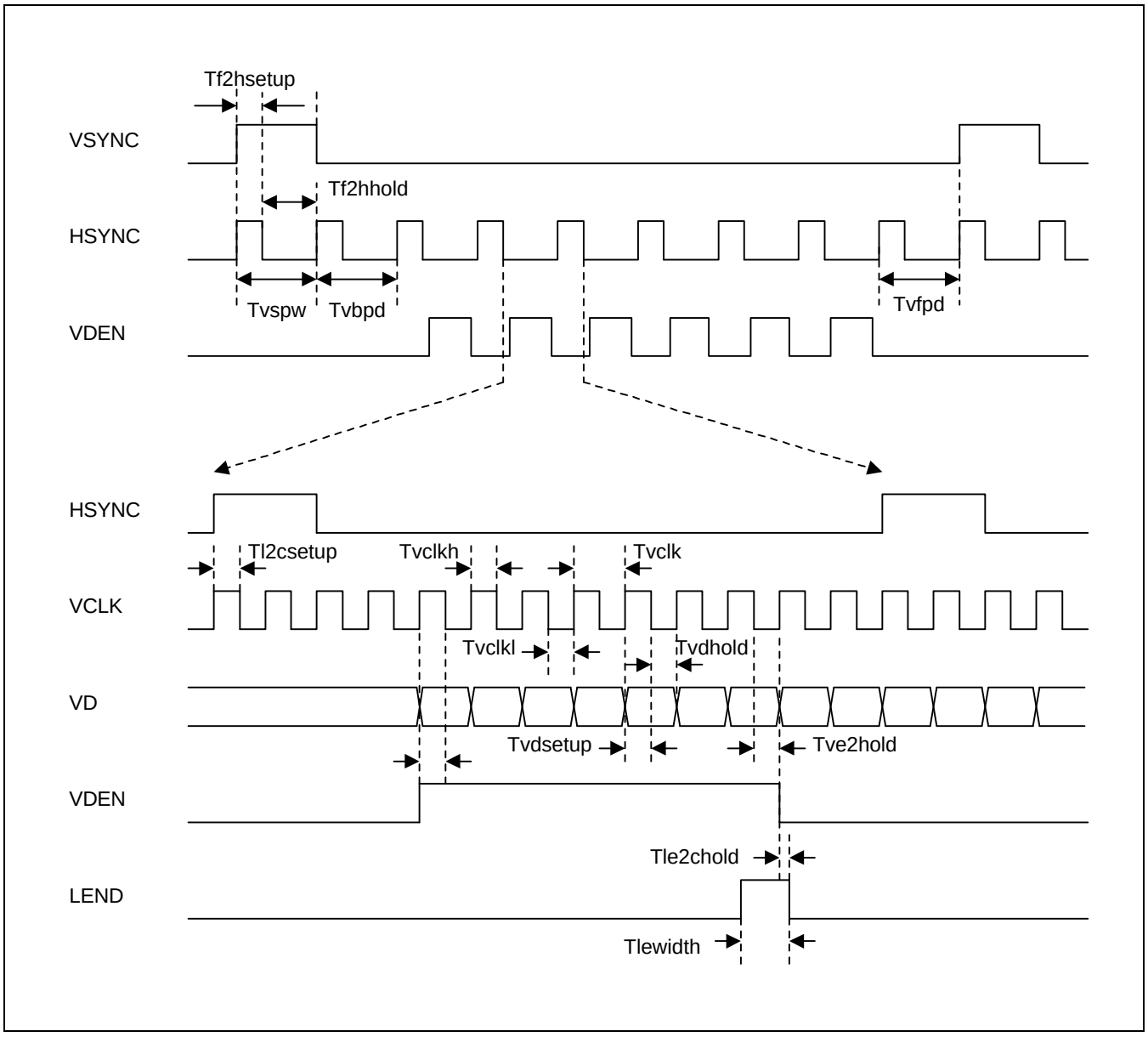


Figure 23-40. TFT LCD Controller Timing

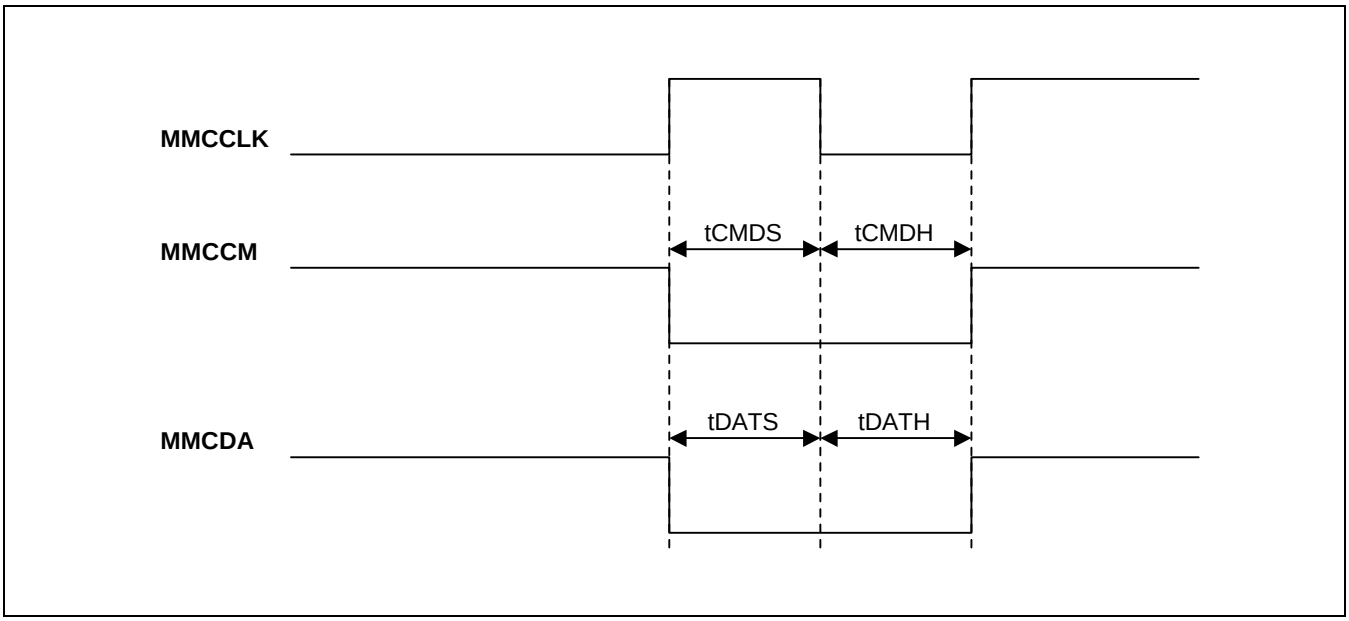


Figure 23-41. MMC Interface Receive Timing

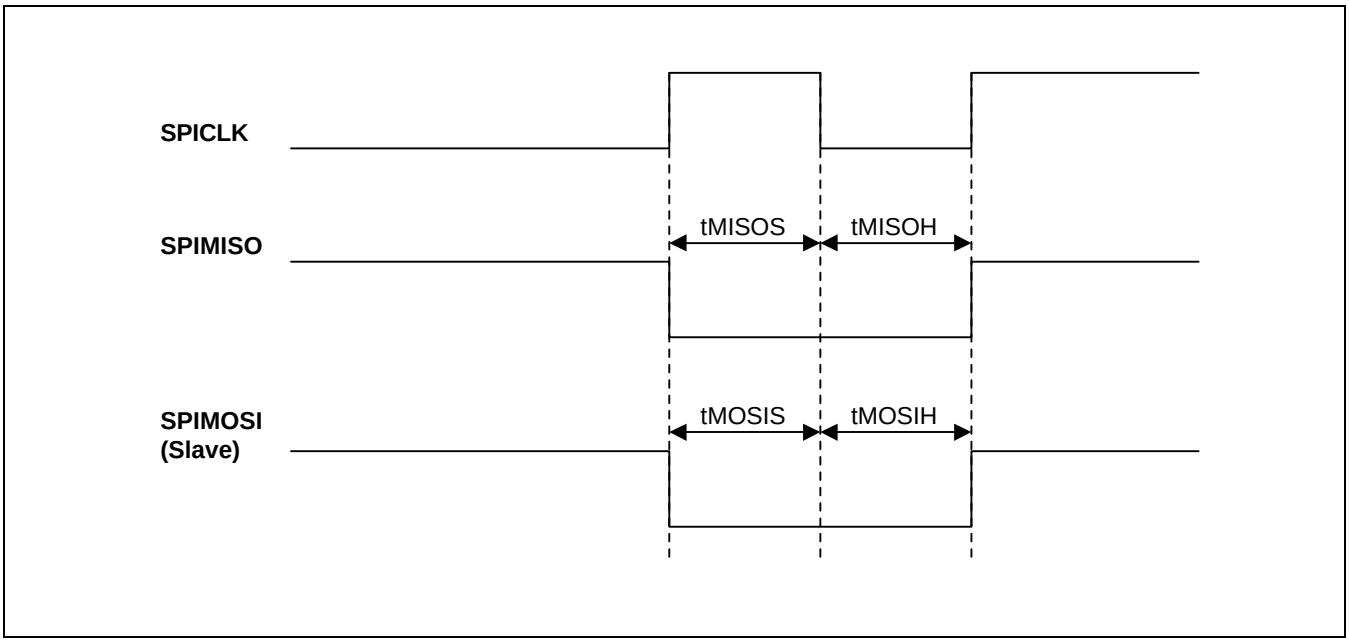


Figure 23-42. SPI Interface Receive Timing

Table 23-6. Clock Timing Constants

(V_{DDP} = 3.3V, V_{DDI} = 1.8V, T_A = 25 °C, max/min = typ ± 30%)

Parameter	Symbol	Min	Typ	Max	Unit
Crystal clock input frequency	f_{XTAL}	6	–	20	MHz
Crystal clock input cycle time	$t_{XTALCYC}$	50	–	166.7	ns
External clock input frequency	f_{EXT}	1	–	66	MHz
External clock input cycle time	t_{EXTCYC}	15.1	–	1000	ns
External clock input low level pulse width	t_{EXTLOW}	5	–	–	ns
External clock to HCLK (without PLL)	t_{EX2HC}	–	6.7	–	ns
HCLK(internal) to CLKOUT	t_{HC2CK}	–	8	–	ns
HCLK(internal) to SCLK	$t_{HC2SCLK}$	–	3.2	–	ns
External clock input high level pulse width	$t_{EXTHIGH}$	5	–	–	ns
Mode reset hold time	t_{MDRH}	3.0	–	–	ns
Reset assert time after clock stabilization	t_{RESW}	4	–	–	XTIpll or EXTCLK
Power-on oscillation setting time	t_{OSC1}	–	–	4096	XTIpll or EXTCLK
STOP mode return oscillation setting time	t_{OSC2}	–	–	4096	XTIpll or EXTCLK
the interval before CPU runs after nRESET is released.	$t_{RST2RUN}$	–	7	–	XTIpll or EXTCLK

Table 23-7. ROM/SRAM Bus Timing Constants

(V_{DD} = 1.8 V ± 0.15 V, T_A = 0 to 70 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ	Max	Unit
ROM/SRAM Address Delay	t_{RAD}	–	8	–	ns
ROM/SRAM Chip select Delay	t_{RCD}	–	7	–	ns
ROM/SRAM Output enable Delay	t_{ROD}	–	7	–	ns
ROM/SRAM read Data Setup time.	t_{RDS}	–	1	–	ns
ROM/SRAM read Data Hold time.	t_{RDH}	–	5	–	ns
ROM/SRAM Byte Enable Delay	t_{RBED}	–	7	–	ns
ROM/SRAM Write Byte Enable Delay	t_{RWBED}	–	7	–	ns
ROM/SRAM output Data Delay	t_{RDD}	–	7	–	ns
ROM/SRAM external Wait Setup time	t_{WS}	–	1	–	ns
ROM/SRAM external Wait Hold time	t_{WH}	–	5	–	ns
ROM/SRAM Write enable Delay	t_{RWD}	–	8	–	ns

Table 23-8. Clock Timing Constants

(V_{DD} = 1.8 V ± 0.15 V, T_A = 0 to 70 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ	Max	Unit
DRAM Address Delay	t_{DAD}	–	8	–	ns
DRAM Row active Delay	t_{DRD}	–	7	–	ns
DRAM Read Column active Delay	t_{DRCD}	–	7	–	ns
DRAM Output enable Delay	t_{DOD}	–	7	–	ns
DRAM read Data Setup time	t_{DDS}	–	1	–	ns
DRAM read Data Hold time	t_{DDH}	–	5	–	ns
DRAM Write Cas active Delay	t_{DWCD}	–	7	–	ns
DRAM Cbr Cas active Delay	t_{DCCD}	–	7	–	ns
DRAM Write enable Delay	t_{DWD}	–	7	–	ns
DRAM output Data Delay	t_{DDD}	–	10	–	ns

Table 23-9. Memory Interface Timing Constants

(V_{DD} = 1.8 V ± 0.15 V, T_A = 0 to 70 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ	Max	Unit
SDRAM Address Delay	t_{SAD}	–	5	–	ns
SDRAM Chip Select Delay	t_{SCSD}	–	4	–	ns
SDRAM Row active Delay	t_{SRD}	–	4	–	ns
SDRAM Column active Delay	t_{SCD}	–	4	–	ns
SDRAM Byte Enable Delay	t_{SBED}	–	4	–	ns
SDRAM Write enable Delay	t_{SWD}	–	4	–	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}	–	5	–	ns
SDRAM Clock Enable Delay	T_{cked}	–	5	–	ns

Table 23-10. External Bus Request Timing Constants

(V_{DD} = 1.8 V ± 0.15 V, T_A = 0 to 70 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ.	Max	Unit
eXternal Bus Request Setup time	t_{XnBRQS}	–	2	–	ns
eXternal Bus Request Hold time	t_{XnBRQH}	–	5	–	ns
eXternal Bus Ack Delay	$t_{XnBACKD}$	–	15	–	ns
HZ Delay	t_{HZD}	–	6	–	ns

Table 23-11. DMA Controller Module Signal Timing Constants

(V_{DD} = 1.8 V ± 0.15 V, T_A = 0 to 70 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ.	Max	Unit
eXternal Request Setup	t_{XRS}	–	9.3	–	ns
aCcess to Ack Delay when Low transition	t_{CADL}	–	6.8	–	ns
aCcess to Ack Delay when High transition	t_{CADH}	–	6.6	–	ns
eXternal Request Delay	t_{XAD}	2	–	–	MCLK

Table 23-12. STN LCD Controller Module Signal Timing Constants

(V_{DD} = 1.8 V ± 0.15 V, T_A = 0 to 70 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ	Max	Units
VFRAME setup to VLINE falling edge	T1	16	–	–	Phclk (note)
VFRAME hold from VLINE falling edge	T2	16	–	–	Phclk
VLINE falling edge to VCLK rising edge	Tl2c	16	–	–	Phclk
VD setup to VCLK falling edge	Tc2dsetup	2	–	–	Phclk
VD hold from VCLK falling edge	Tc2dhold	2	–	–	Phclk
VCLK Period	Tcwidth	4	–	–	Phclk

NOTE: HCLK period

Table 23-13. TFT LCD Controller Module Signal Timing Constants

(V_{DD} = 1.8 V ± 0.15 V, T_A = 0 to 70 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ	Max	Units
Vertical sync pulse width	Tvspw	VSPW + 1	–	–	Phclk (1)
Vertical back porch delay	Tvbpd	VBPD+1	–	–	Phclk
Vertical front porch dealy	Tvfpd	VFPD+1	–	–	Phclk
VCLK pulse width	Tvclk	1	–	–	Pvclk (2)
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 23-14. IIS Controller Module Signal Timing Constants

(V_{DD} = 1.8 V ± 0.15 V, T_A = 0 to 70 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ.	Max	Unit
IISLRCK delay time	t_{LRCK}	0.5	–	5.7	ns
IISDO delay time	t_{SDO}	0.4	–	2.5	ns
IISDI input setup time	t_{SDIS}	7.9	–	–	ns
IISDI input hold time	t_{SDIH}	0.3	–	–	ns
CODEC clock frequency	t_{CODEC}	1/16	–	1	f_{IIS_BLOCK}

Table 23-15. IIC BUS Controller Module Signal Timing

(V_{DD} = 1.8 V ± 0.15 V, T_A = 0 to 70 °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

NOTE: Std. means Standard Mode and fast means Fast Mode.

Table 23-16. MMC Interface Transmit/Receive Timing Constants

(V_{DD} = 1.8 V ± 0.15 V, T_A = 0 to 70 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ.	Max	Unit
CMD Receive data input setup time	t_{CMDS}	–	9.1	–	ns
CMD Receive data input hold time	t_{CMDH}	–	0.1	–	ns
DAT Receive data input setup time	t_{DATS}	–	8.8	–	ns
DAT Receive data input hold time	t_{DATH}	–	0.1	–	ns

Table 23-17. SPI Interface Transmit/Receive Timing Constants

(V_{DD} = 1.8 V ± 0.15 V, T_A = 0 to 70 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Min	Typ.	Max	Unit
MISO Receive data input setup time	t_{MISOS}	–	9.4	–	ns
MISO Receive data input hold time	t_{MISOH}	–	0.1	–	ns
MOSI(slave) Receive data input setup time	t_{MOSIS}	–	0.1	–	ns
MOSI(slave) Receive data input hold time	t_{MOSIH}	–	0.8	–	ns

Table 23-18. USB Electrical Specifications

(V_{DD} = 1.8 V ± 0.15 V, T_A = 0 to 70 °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 23-19. USB Full Speed Output Buffer Electrical Characteristics

(V_{DD} = 1.8 V ± 0.15 V, T_A = 0 to 70 °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 23-20. USB Low Speed Output Buffer Electrical Characteristics

(V_{DD} = 1.8 V ± 0.15 V, T_A = 0 to 70 °C, V_{EXT} = 3.3V ± 0.3V)

Parameter	Symbol	Condition	Min	Max	Unit
Driver Characteristics					
Transition Time Rising Time	TR	CL = 50pF	75		ns
		CL = 350pF		300	
Falling Time	TF	CL = 50pF	75		
		CL = 350pF		300	
Rise/Fall Time Matching	TRFM	(TR / TF)	80	120	%
Output Signal Crossover Voltage	VCRS		1.3	2.0	V

24 MECHANICAL DATA

PACKAGE DIMENSIONS

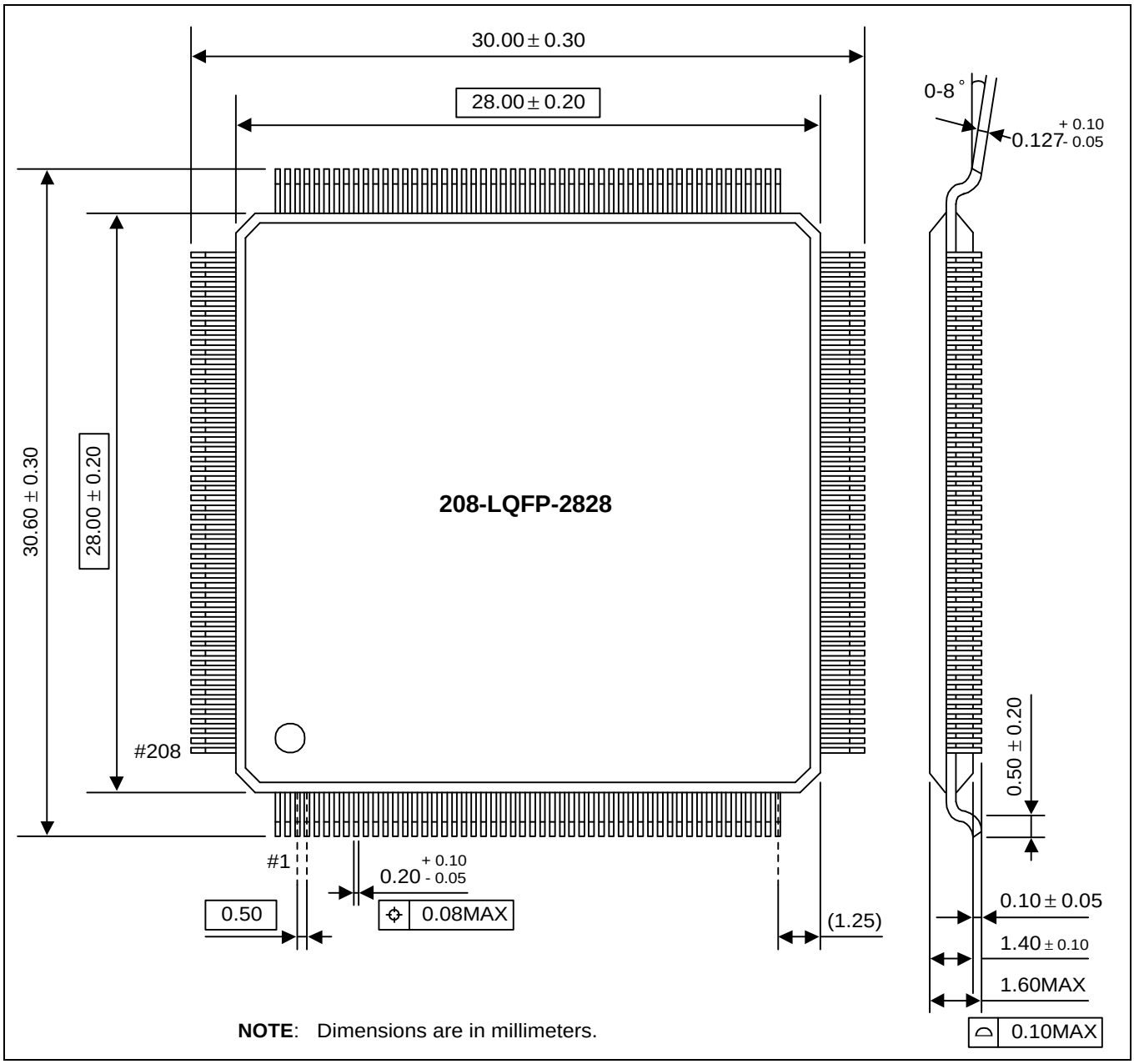


Figure 24-1. 208-LQFP- 2828 Package Dimension

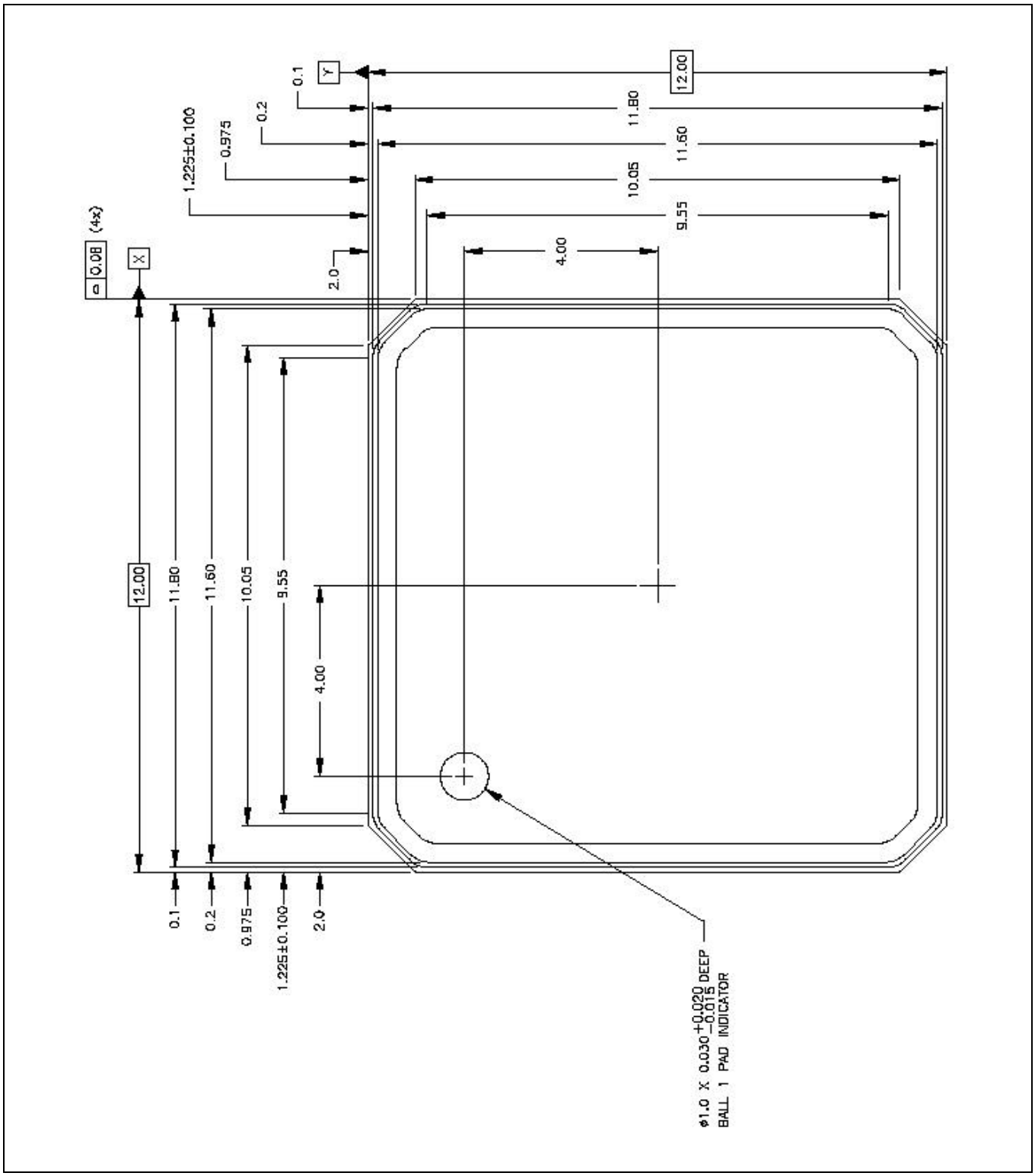


Figure 24-2. 208-FBGA-12.0X12.0 Package Dimension 1

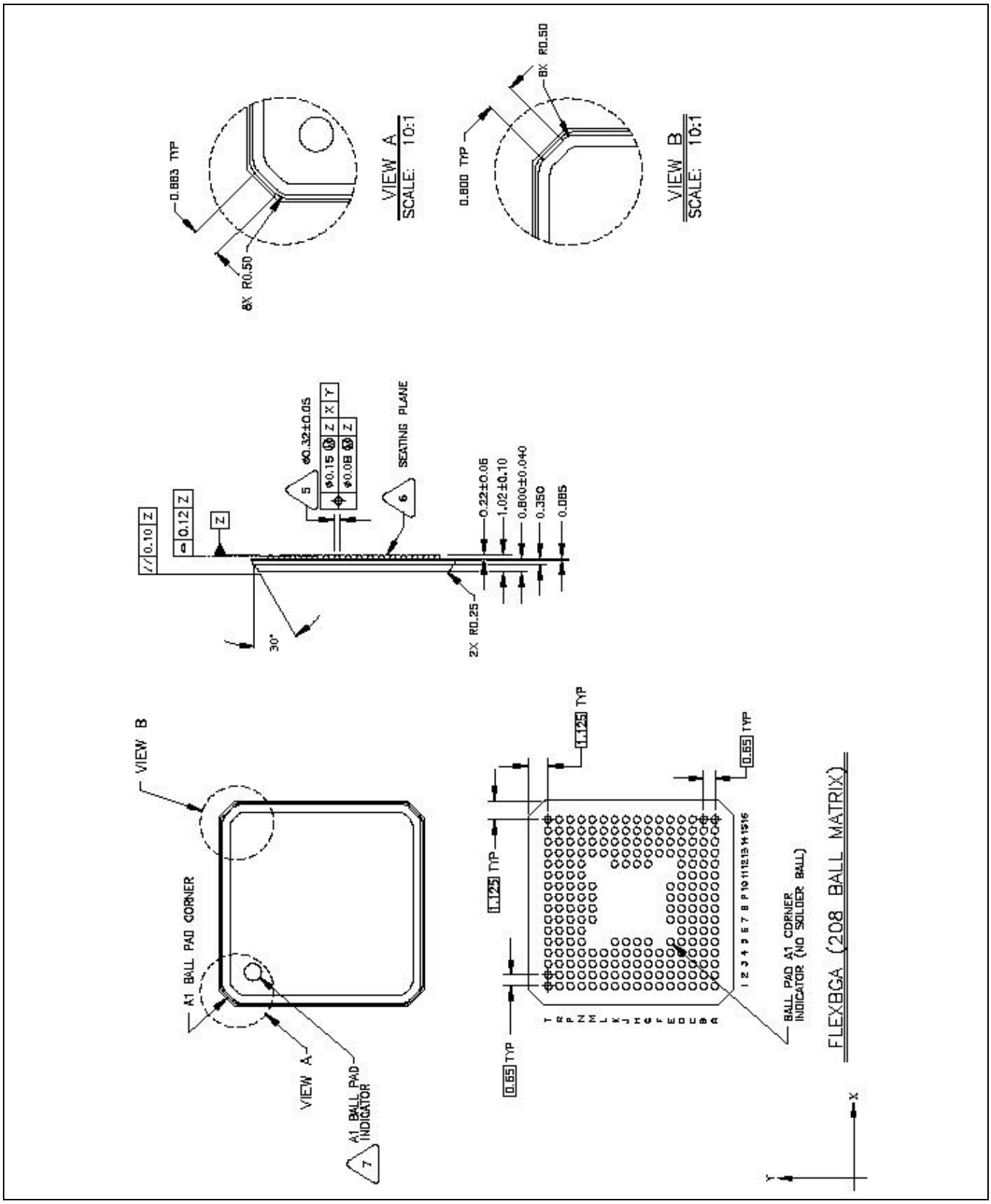


Figure 24-3. 208-FBGA-12.0X12.0 Package Dimension 2

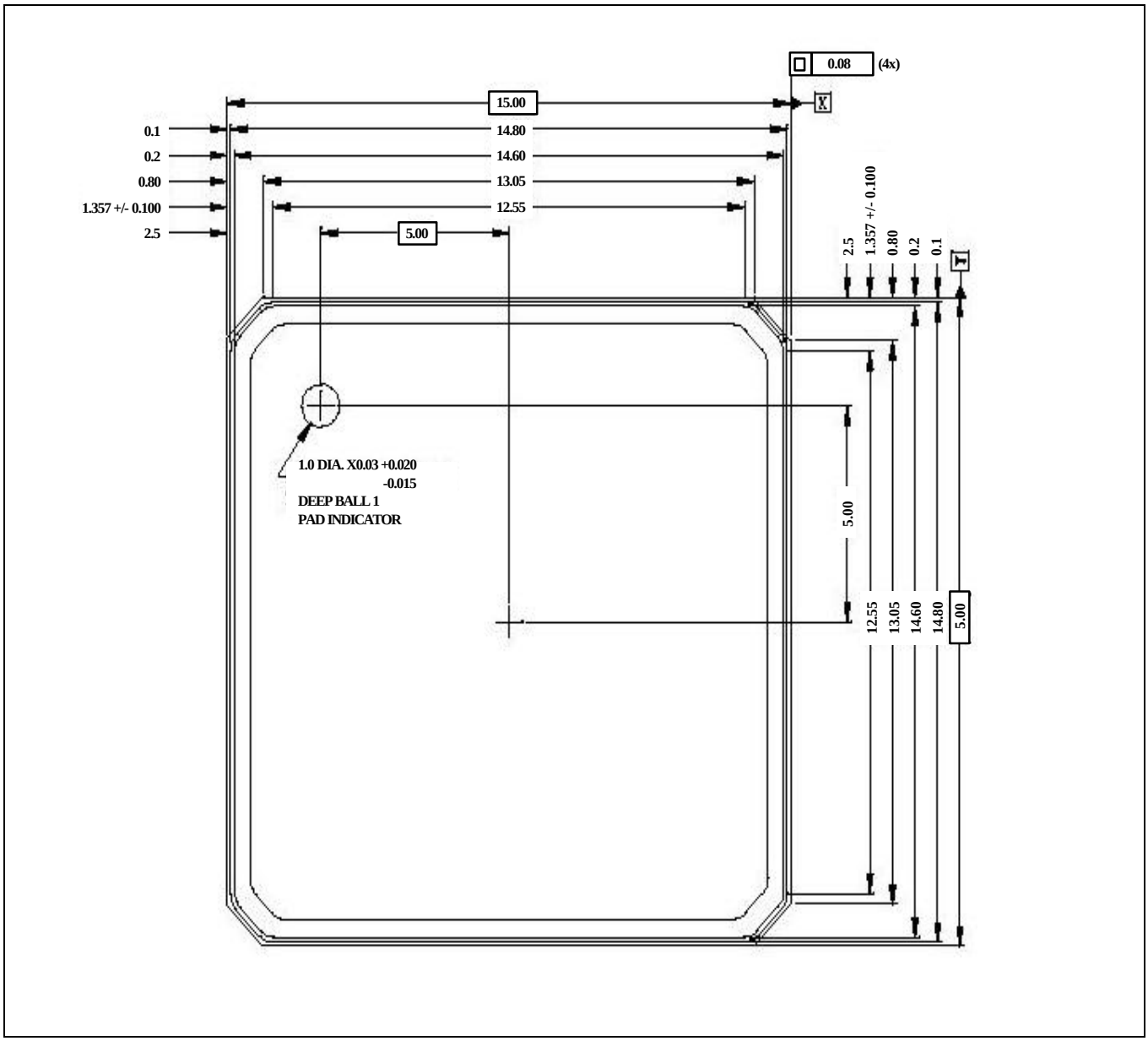


Figure 24-4. 208-FBGA-15.0X15.0 Package Dimension 1

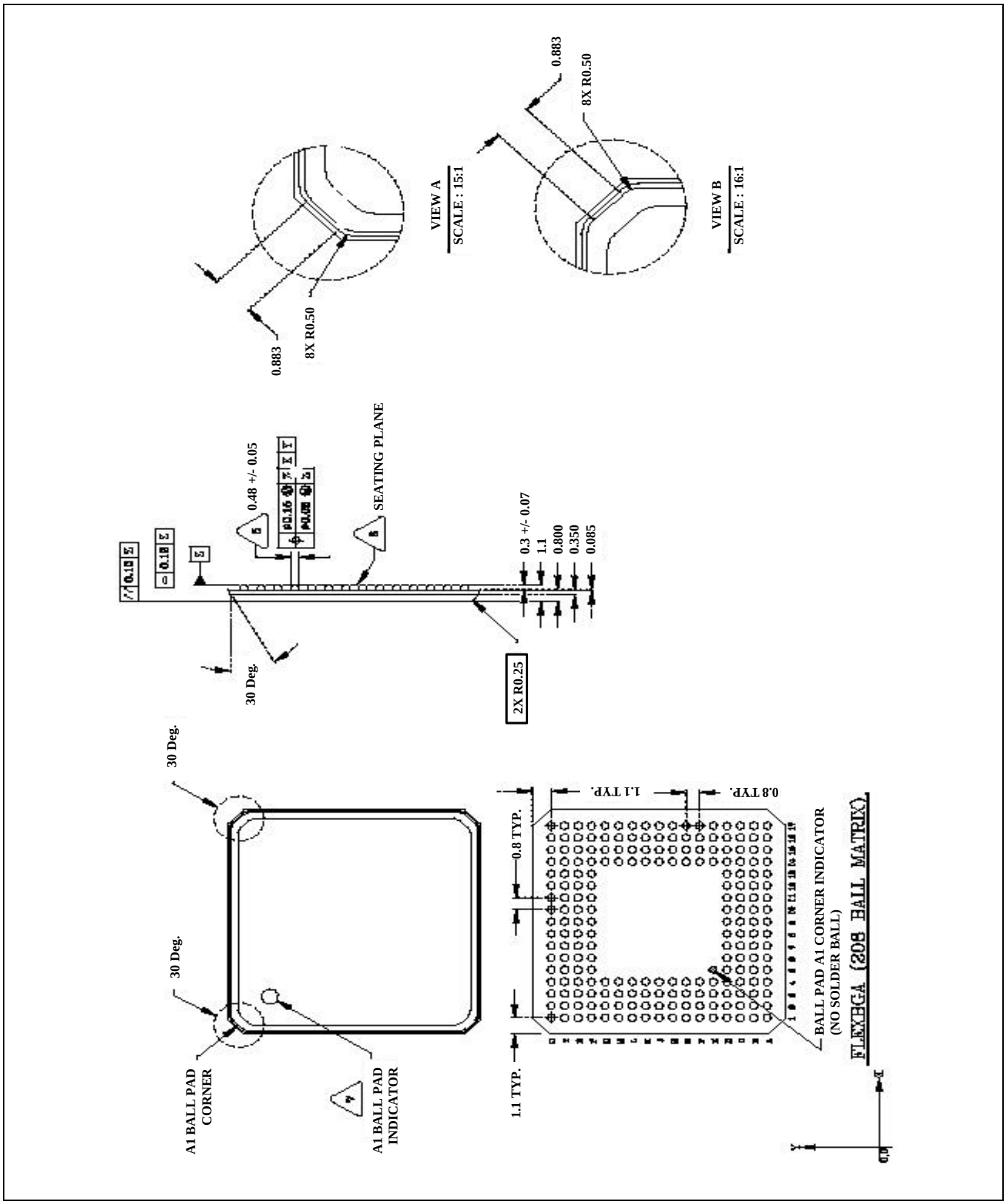


Figure 24-5. 208-FBGA-15.0X15.0 Package Dimension 2

NOTES

25 NOTICE IN PROGRAMMING

VER 1.41(DEC.3.2001)

The final S3C2400X is the chips after ES140(Some FIB sample has same functionality with ES140.). The final S3C2400X has the following problems and cautions.

KNOWN PROBLEMS

LDM Instruction of Precisely Two Items from a Non-cached Region

If the cache is turned on and there is an LDM instruction (load multiple) of precisely two items from a non-cached region, then the second item may not be loaded correctly.

So, for example, LDMIA r0,{r1,r2} from non-cacheable area is not allowed. But, LDMIA r0,{r1,r2,r3} from non-cacheable area is allowed because the number of items is not 2.

In assembly code, the software workaround was simply to ensure that there were no LDM instructions which load only 2 values. In compiled C code, LDM may be used only for stack save/restore operation, double type variable load/store and etc. But, we think there is no problem if the C stack area and C variable area is cacheable area.

This problem will be fixed in S3C2410X.

Guide

1. Locate C variables and stack in the cached area.
2. Do not use LDM instruction of precisely two items from a non-cached region (LCD frame buffer, IIS audio data buffer,...).

Excessive Continuous STM / STR Instructions and the Write Buffer.

If STM/STR instruction is used excessively onto the write-buffered area, the write-buffer may occupy the memory bus continuously. It's because the data rate from CPU to the write buffer is higher than the data rate from the write buffer to the memory.

The excessive continuous STM instruction on write-buffered memory area keeps the DMA(LCD,IIS,...) from fetching. So, LCD or IIS DMA will not fetch the data while the write-buffer occupies the bus. So, there will be noise in LCD screen or speaker sound.

So, don't use exc essive continuous STM/STR instructions onto write-buffered memory area as following examples

```
LOOP:   STMIA R0!,{R1-R8}
        CMP R0,R9
        BLS LOOP
```

The continuous STM/STR instruction on a non-cacheable area is okay because the write buffer can be turned off on the non-cacheable area.

Guide

1. Don't use continuous STM/STR instructions onto write -buffered memory. In this case, the area should be write buffer turned-off area by manipulating MMU table.
2. If the noise is generated by the write-buffer problems, just insert any dummy instruction or load instructions (LDR or LDM).

IIS FIFO Problem

Because of the write buffer problem (excessive continuous STR/STM), the IIS FIFO under-run may occur. If the IIS FIFO under-run is occurred, IIS_LRCK signal period may be 15-bit or longer than 16-bit. In this case, the incorrect data will be delivered to a CODEC. So, the noise may be generated sometimes during heavy BUS load.

By our experiment, excessive continuous STR/STM is found in MP3 decoding algorithm. But, the IIS under-run problem in decoding MP3 file has been solved using the following guide.

Guide

To prevent this noise, please refer to the following items to reduce the CPU bus usage.

1. We experienced the small noise at intervals of several seconds during MP3 decoding and a play at fs=44.1khz. But, we can't experienced any small noise at fs=22.05khz.
2. The write-back cache policy has generally less bus usage than write-through policy.
3. If the code is optimized fully by the C compiler option, the bus usage may increase.
4. The noise during the MP3 decoding has been removed by executing the decoding algorithm on a write-back cached region and compiling the MP3 play routine as not optimized. One of our customers has removed this type of noise completely by this method, the item 2 and 3.
5. During playing WAV file, we have not experienced this type of noise because the bus usage is lower than MP3 decoding.
6. The excessive continuous STM instruction onto the write-buffered region will also cause IIS FIFO under-run.

CAUTIONS

The following area should be non-cacheable area using MMU.

— LCD frame buffer

— DMA accessed area(read or read/write) for IIS and so on.

If these areas have write-back attribute, the writing contents will not be updated quickly. If the areas have write-through attribute, the DMA may not fetch the area if the ARM920T executes STM/STR instruction excessively continuously. So, the above areas should be non-cacheable area. Please refer to mmu4k.c for configuring the area as non-cacheable area.

WORK-AROUNDS

The following problems will be fixed by S/W work-around.

Entering STOP Mode

During STOP/IDLE mode, the memory control signals (nGCSn,...) is active. These will be fixed S/W work-around, which uses the I-Cache. Please refer to our test code, 24xinit.s. This work-around will not be needed in S3C2410X.

DMA Auto-reload

The DMA auto-reload is occurred only if the DMA request is issued after the DMA counter reaches 0. So, the following code should be used in DMA terminal counter interrupt handler before setting the DMA source address, destination address and counter register.

```
while((rDSTATn&0xfffff)==0);
```

This code will wait until the first DMA request is issued and the previous auto-reload value is loaded.

NOTES

Appendix 1

ARM920T 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

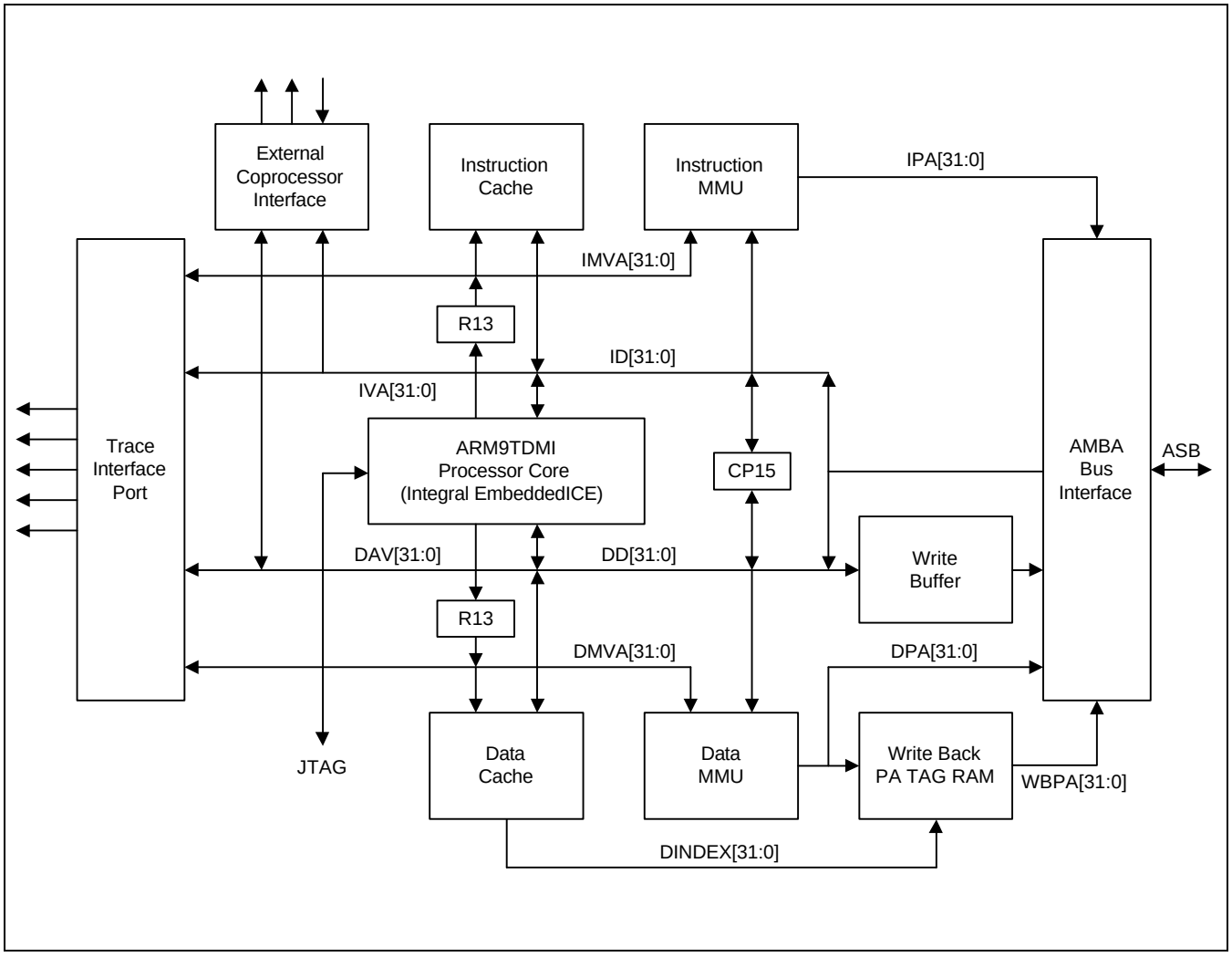


Figure 1-1. ARM920T Functional Block Diagram

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

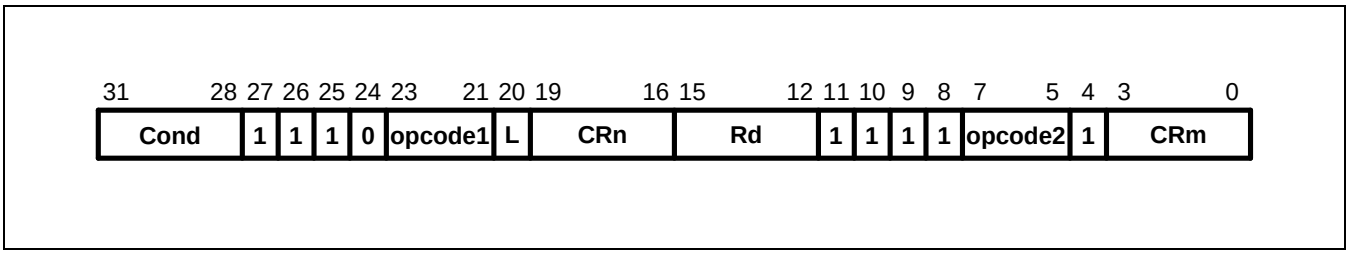


Figure 2-1. CP15 MRC and MCR Bit Pattern

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 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 [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:

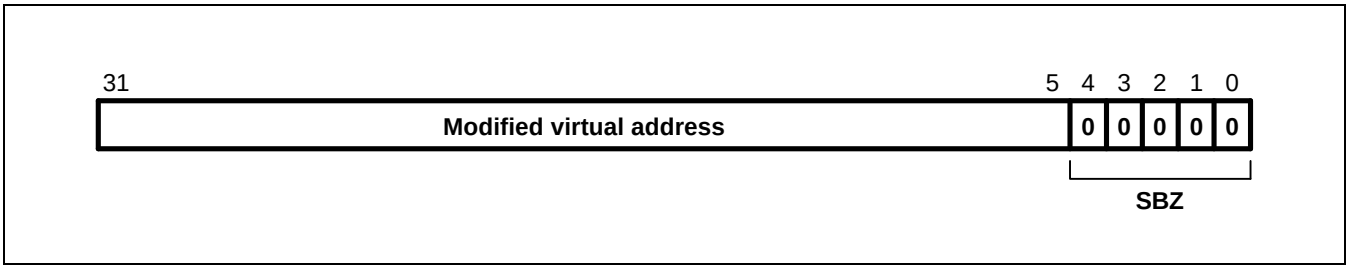


Figure 2-2. Register 7 MVA Format

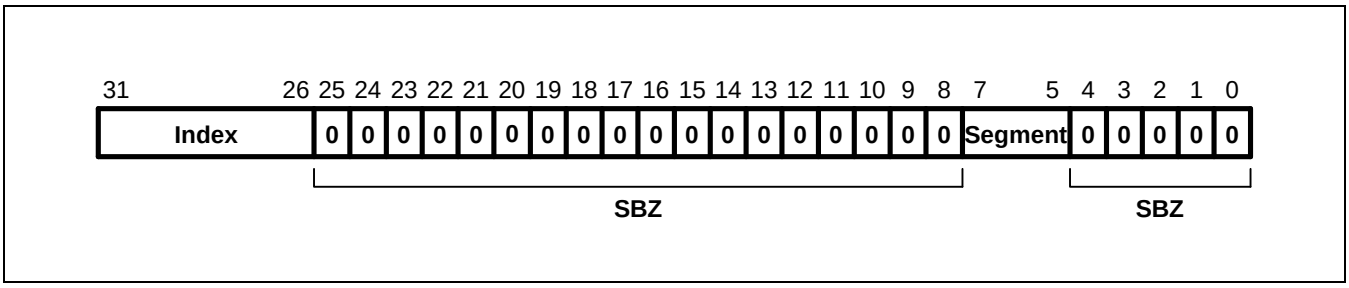


Figure 2-3. Register 7 Index Format

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.

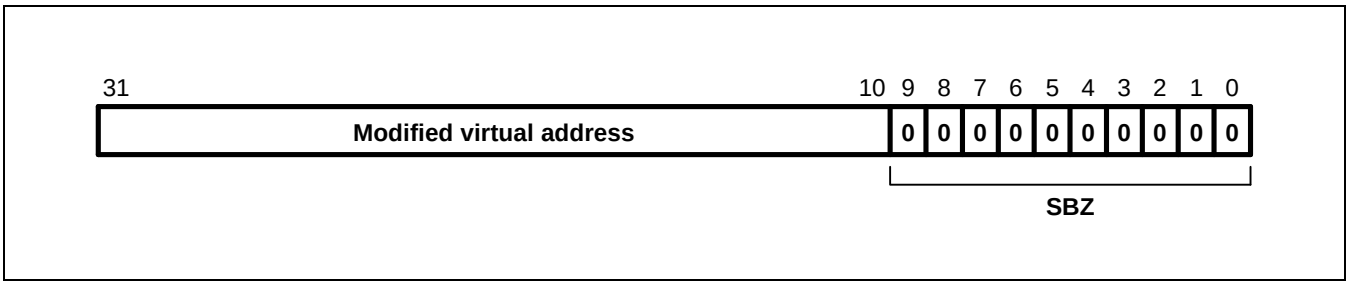


Figure 2-4. Register 8 MVA Format

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

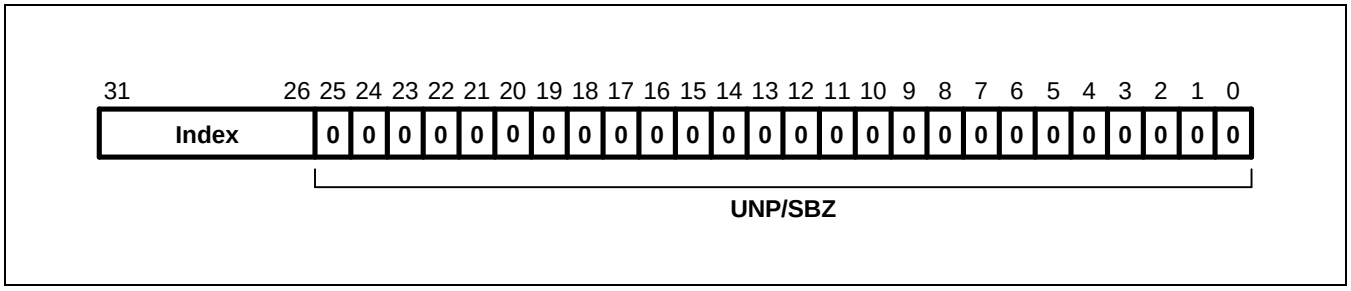


Figure 2-5. Register 9

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.

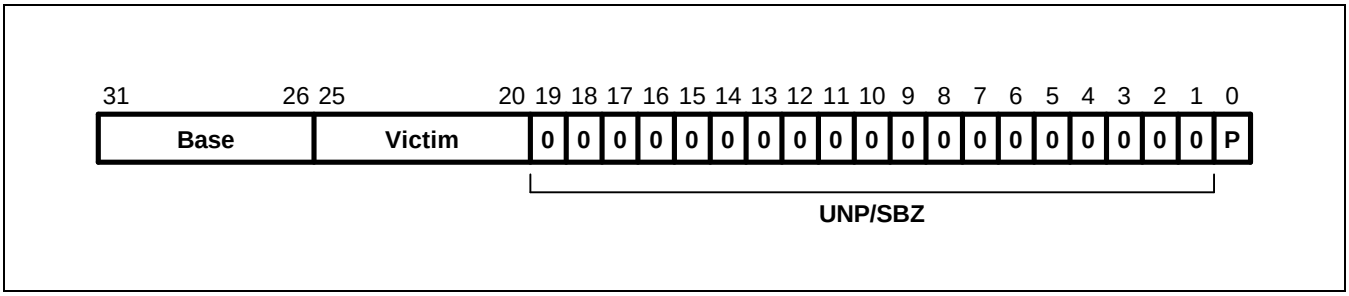


Figure 2-6. Register 10

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.

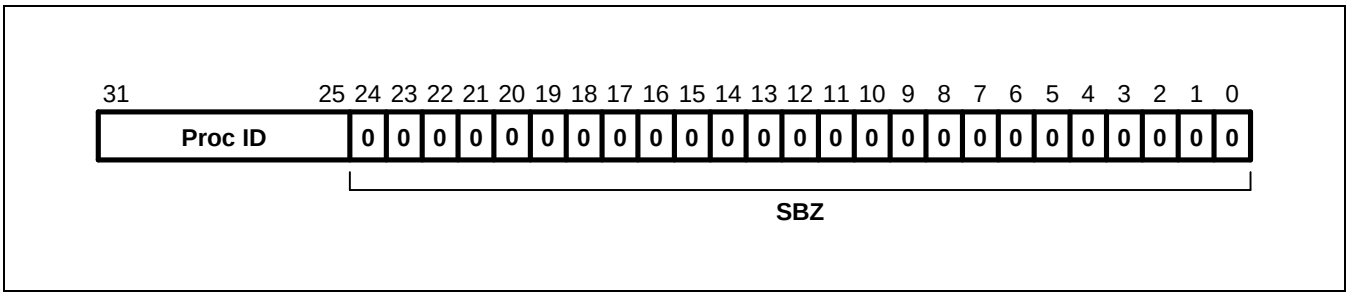


Figure 2-7. Register 13

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.

NOTES

If ProcID is zero, as it is on reset, then there is a flat mapping between the ARM9TDMI and the Caches and MMU.

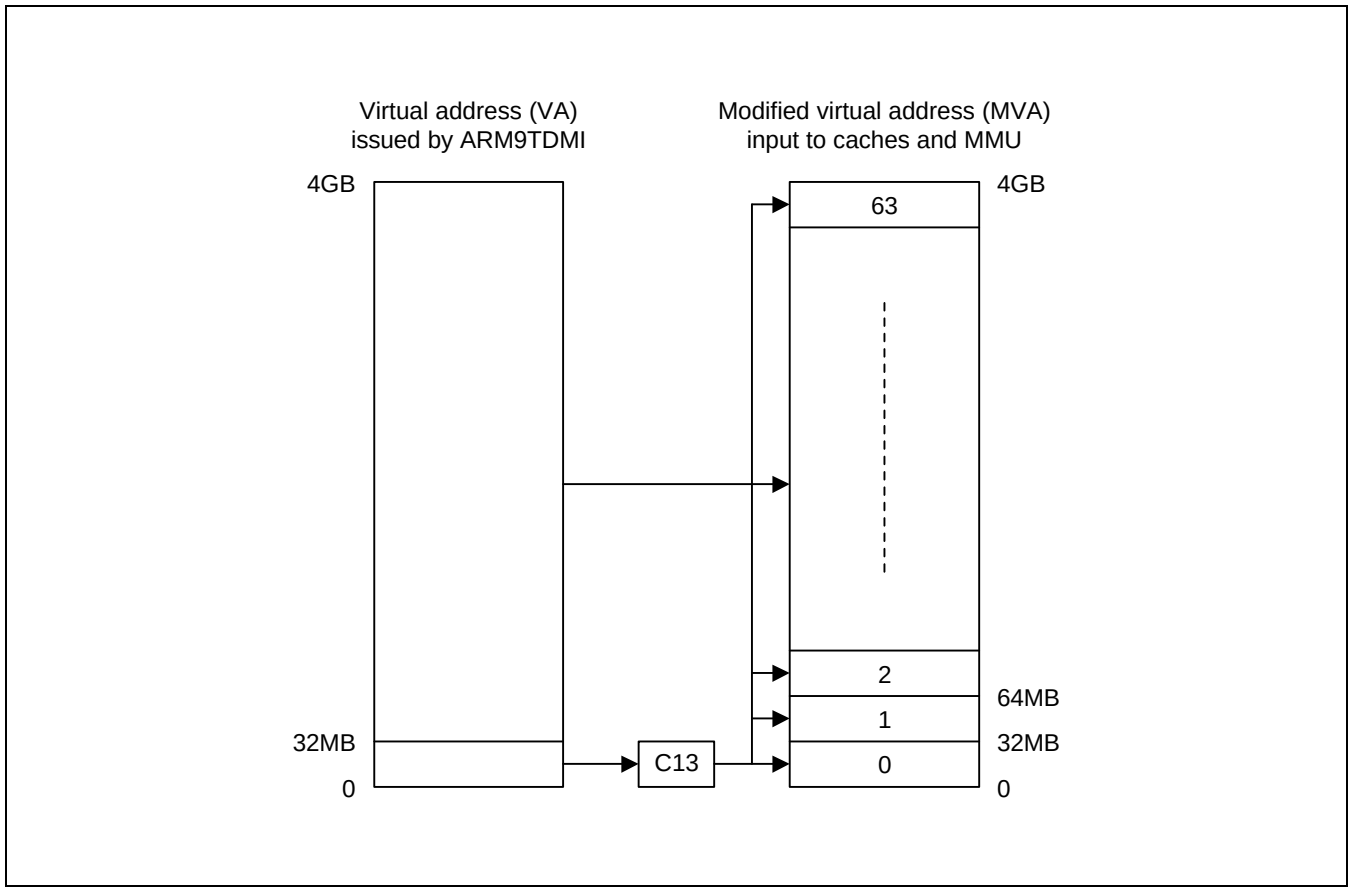


Figure 2-8. Address Mapping Using CP15 Register 13

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.

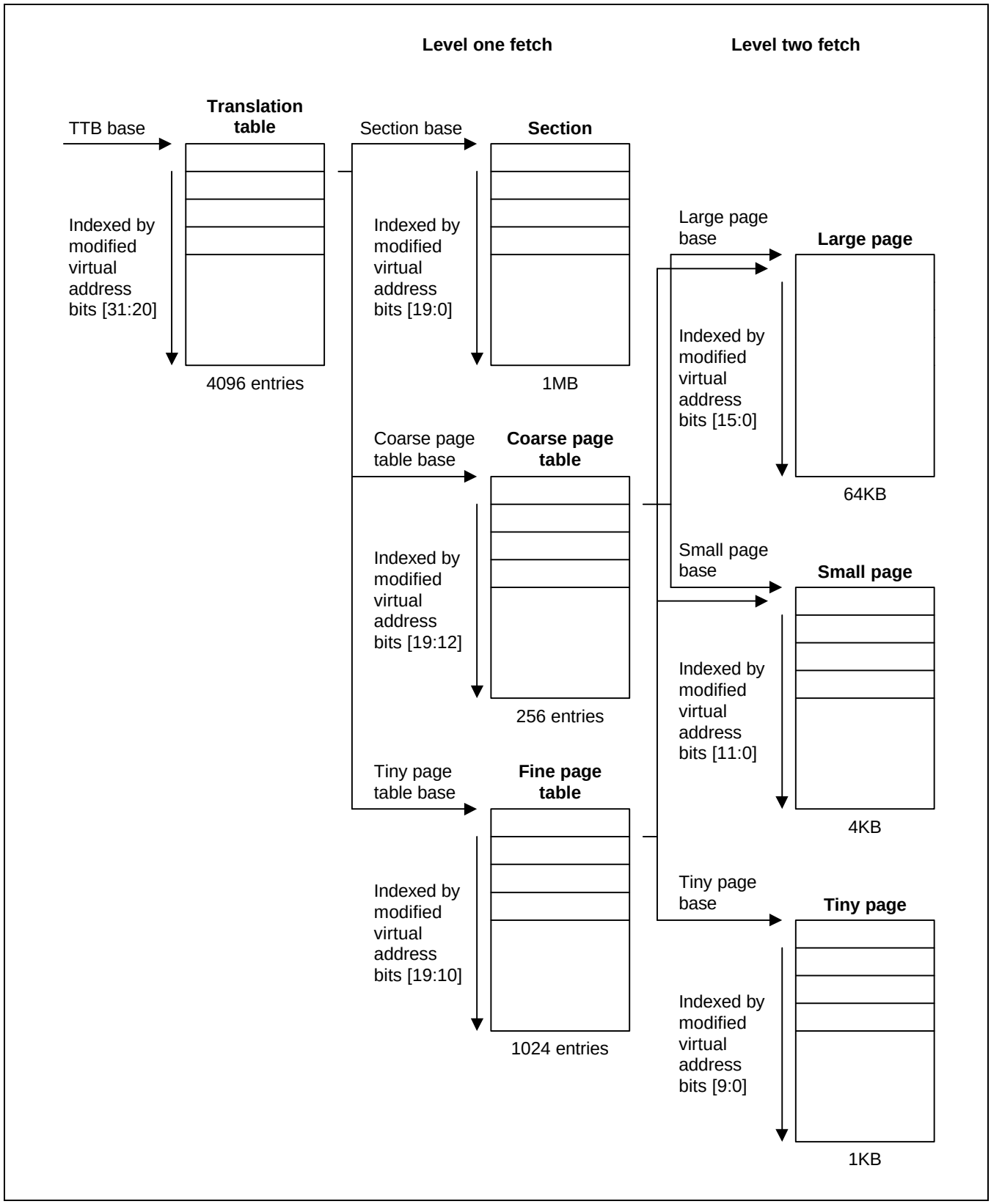


Figure 3-1. Translating Page Tables

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.

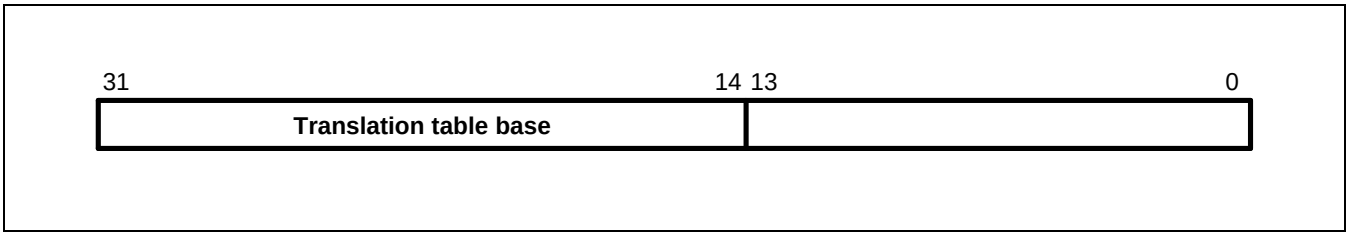


Figure 3-2. Translation Table Base Register

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.

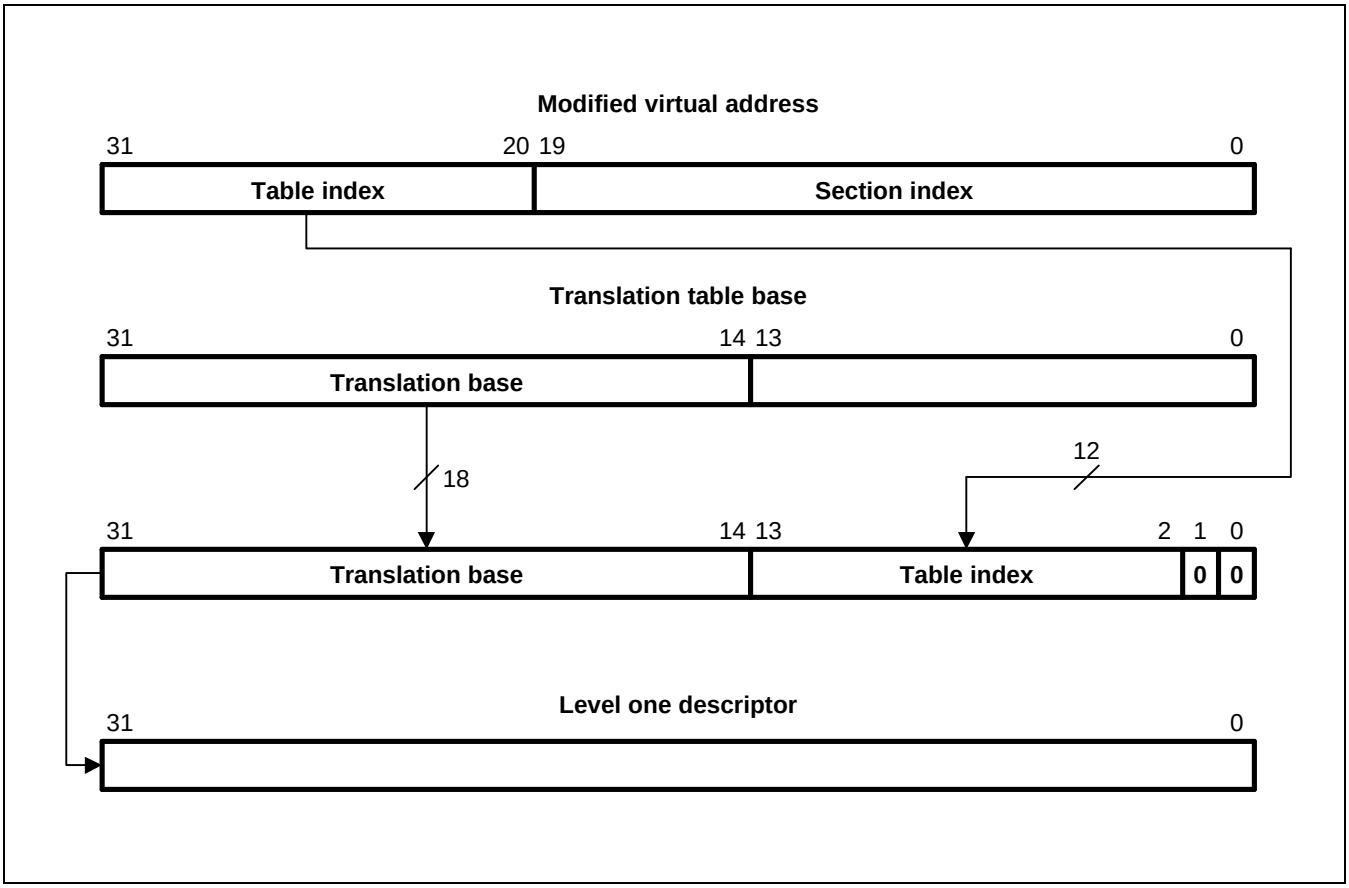


Figure 3-3. Accessing the Translation Table Level One Descriptors

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

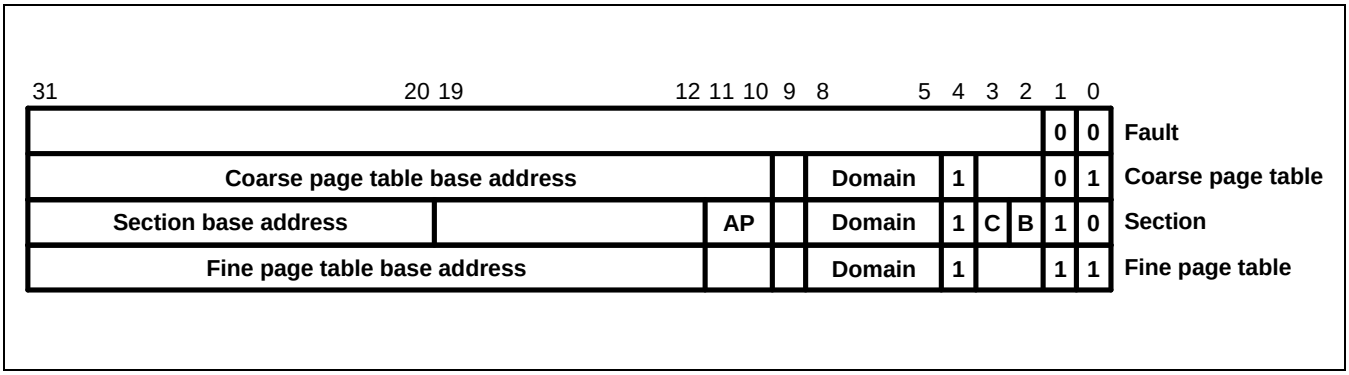


Figure 3-4. Level One Descriptors

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.

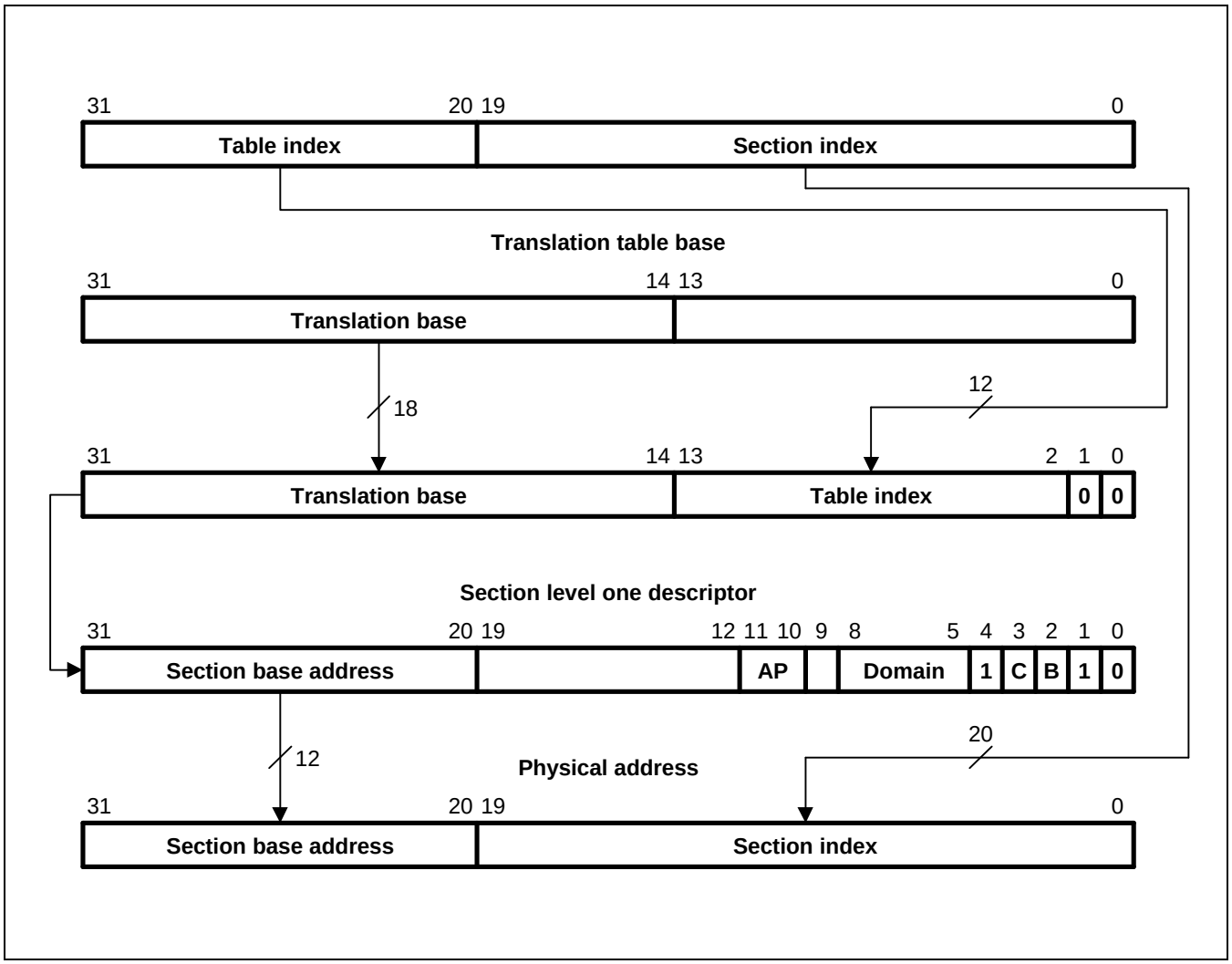


Figure 3-5. Section Translation

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.

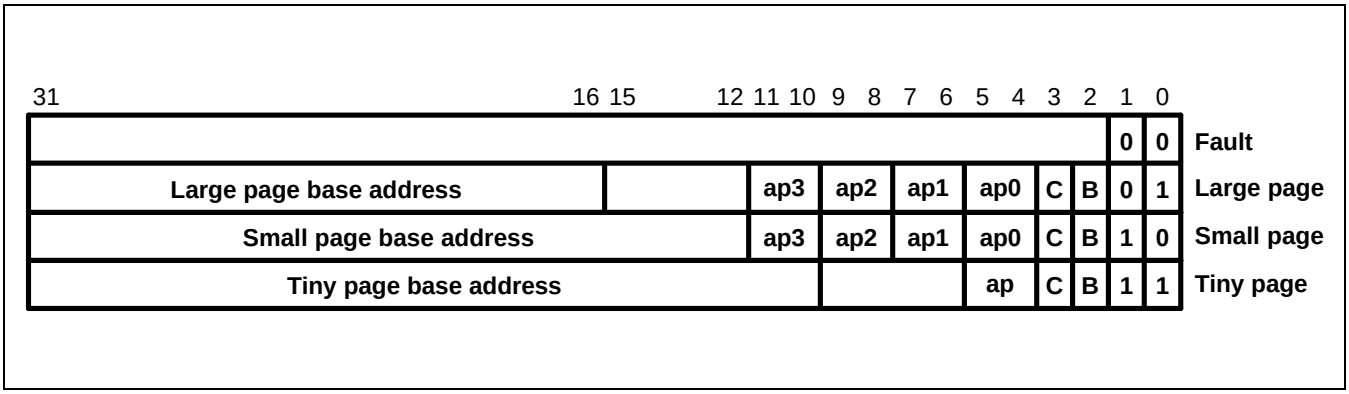


Figure 3-6. Page Table Entry (Level One Descriptor)

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.

NOTES

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.

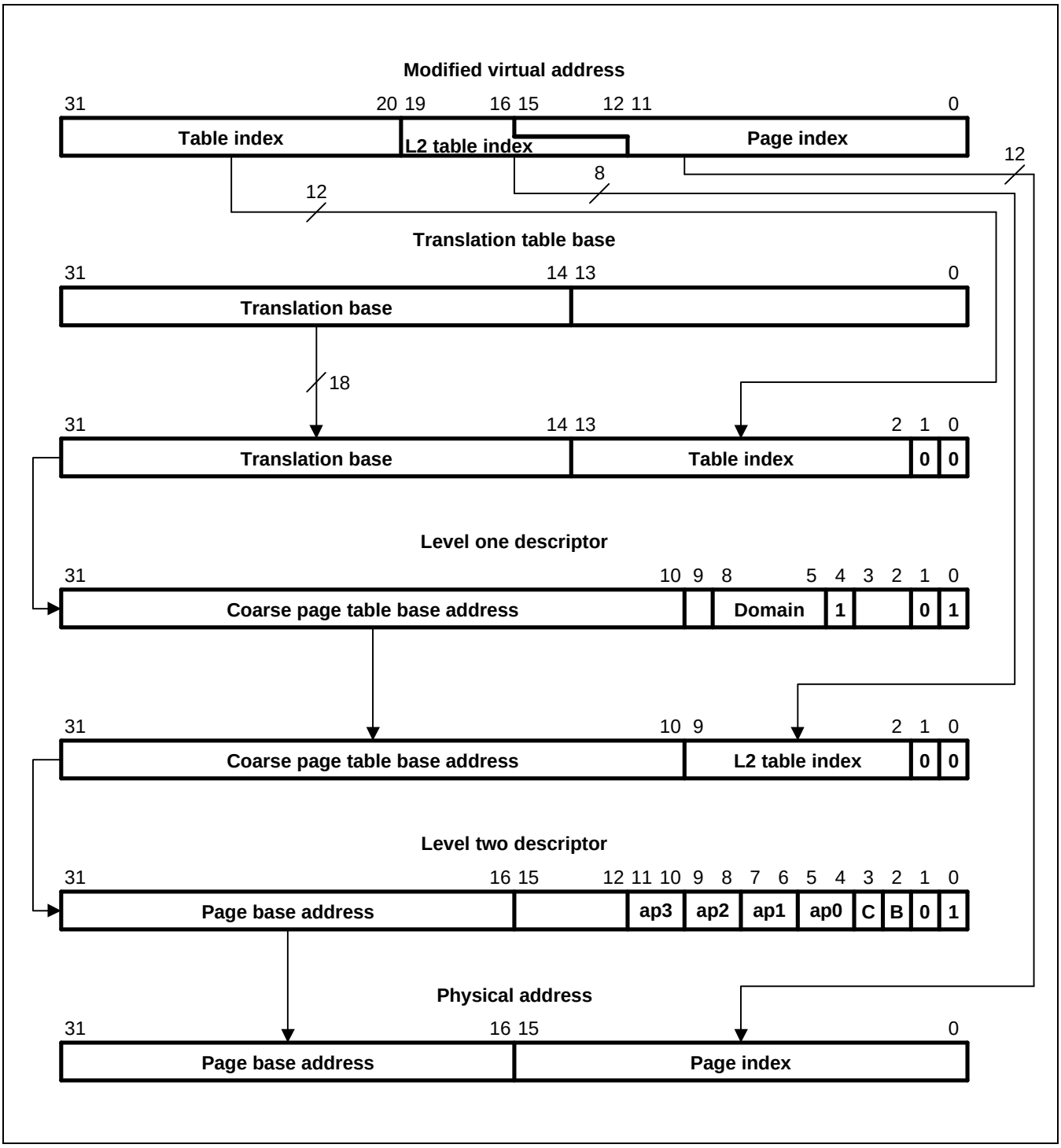


Figure 3-7. Large Page Translation from a Coarse Page Table

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.

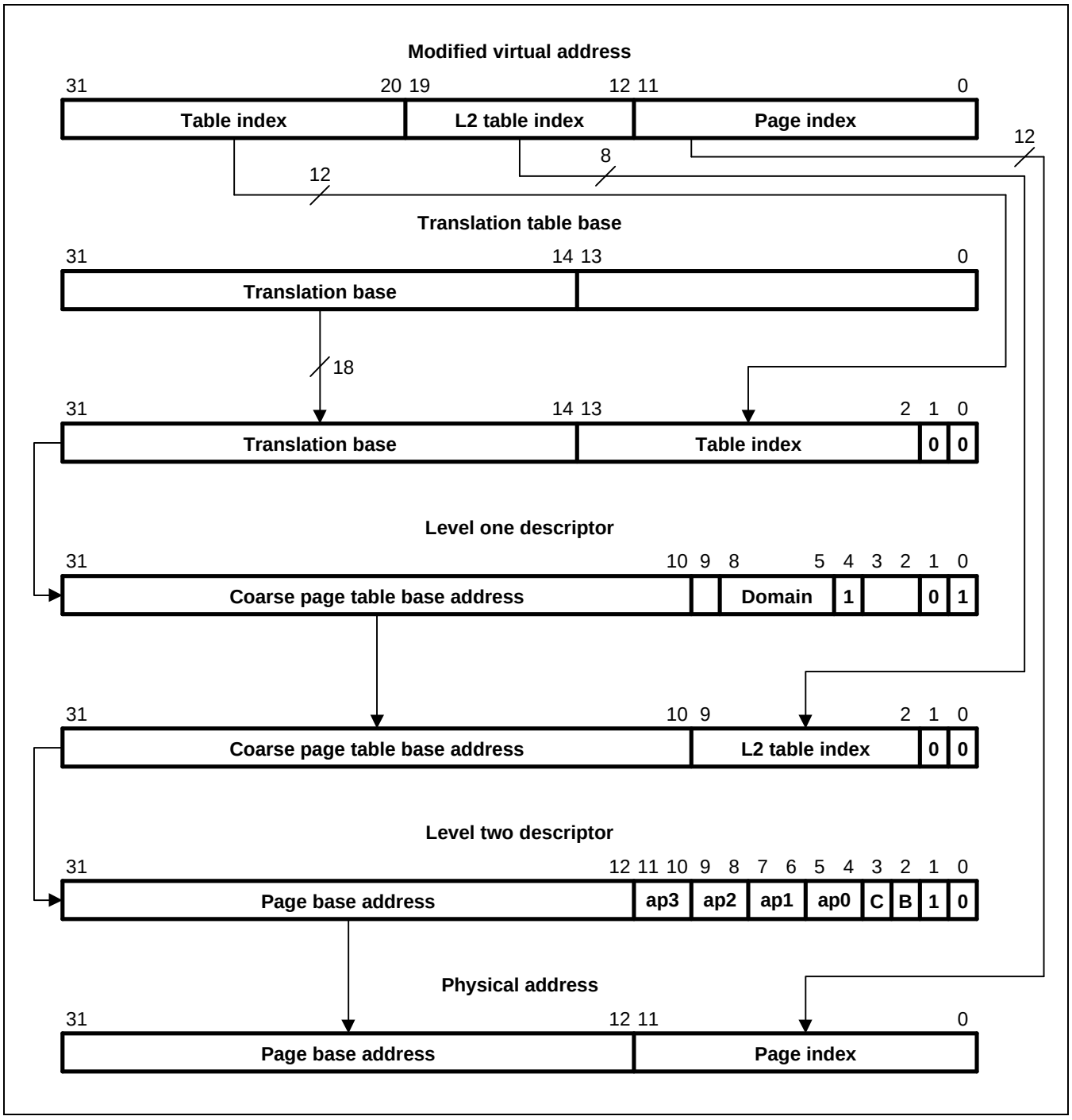


Figure 3-8. Small Page Translation from a Coarse Page Table

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.

NOTES

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.

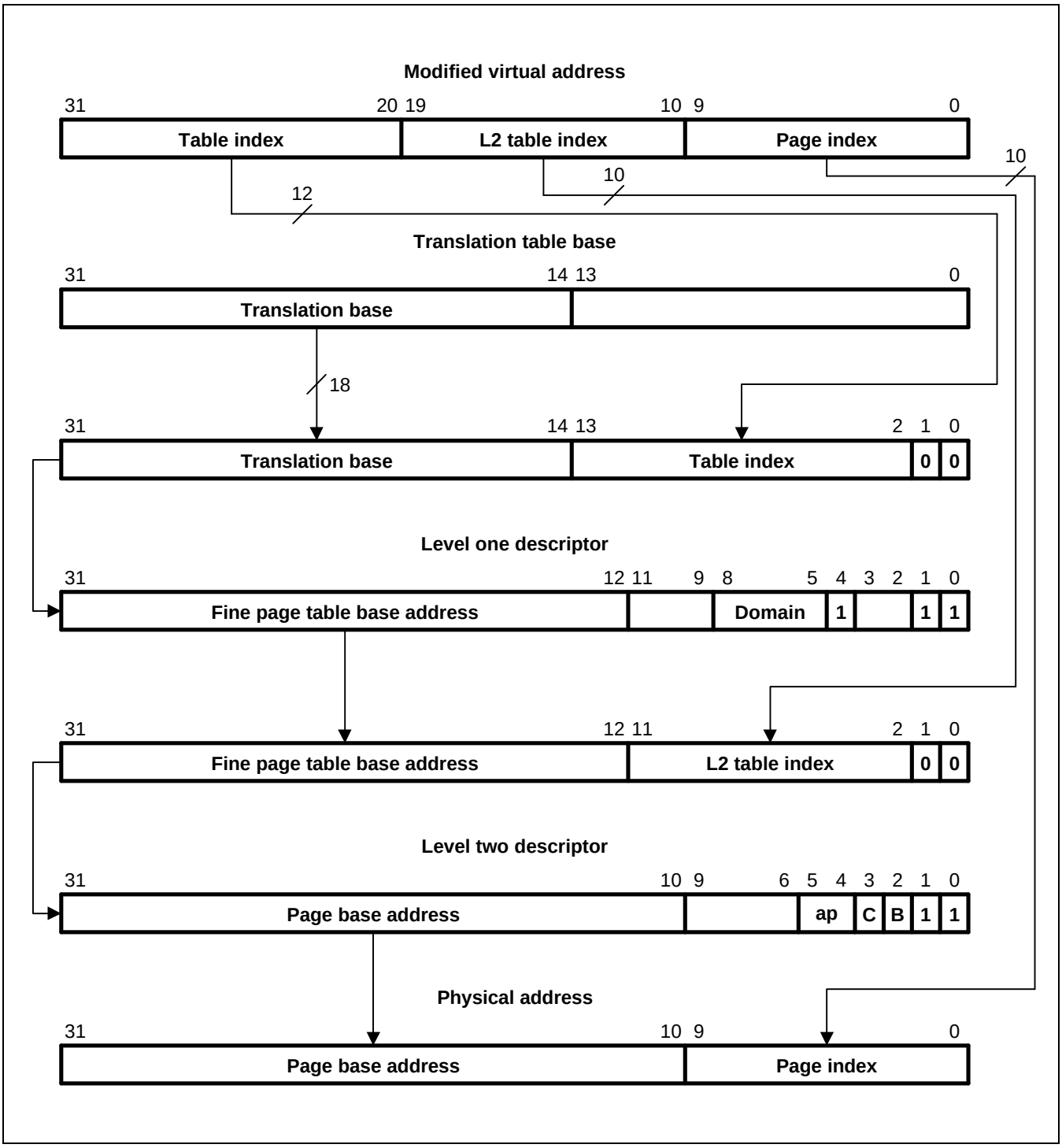


Figure 3-9. Tiny Page Translation from a Fine Page Table

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.

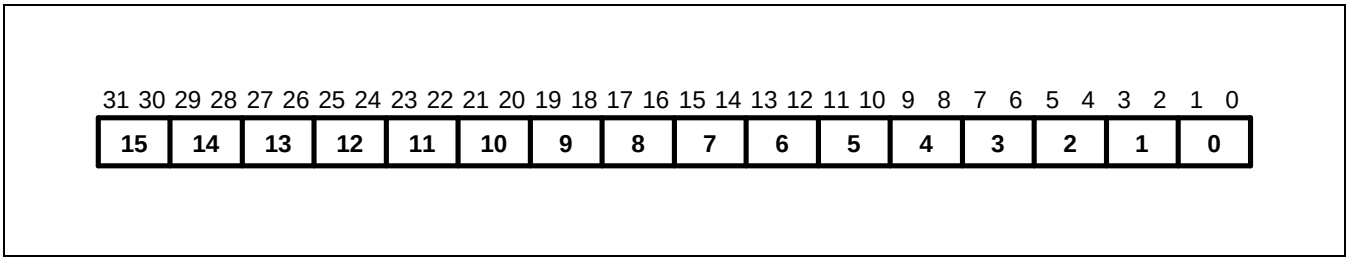


Figure 3-10. Domain Access Control Register Format

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.

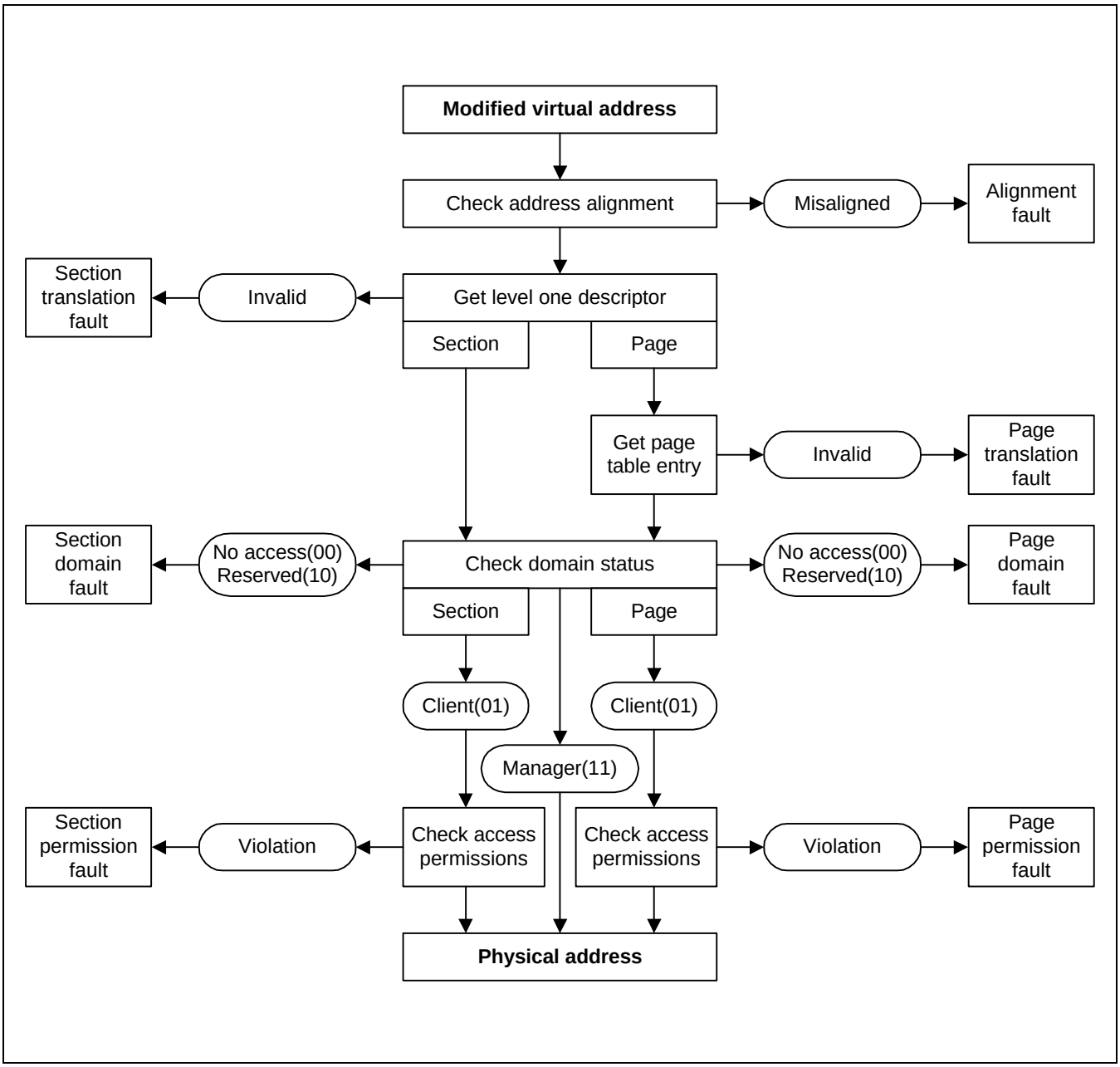


Figure 3-11. Sequence for Checking Faults

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.

NOTES

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.

NOTES

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.

NOTES

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.

NOTES

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

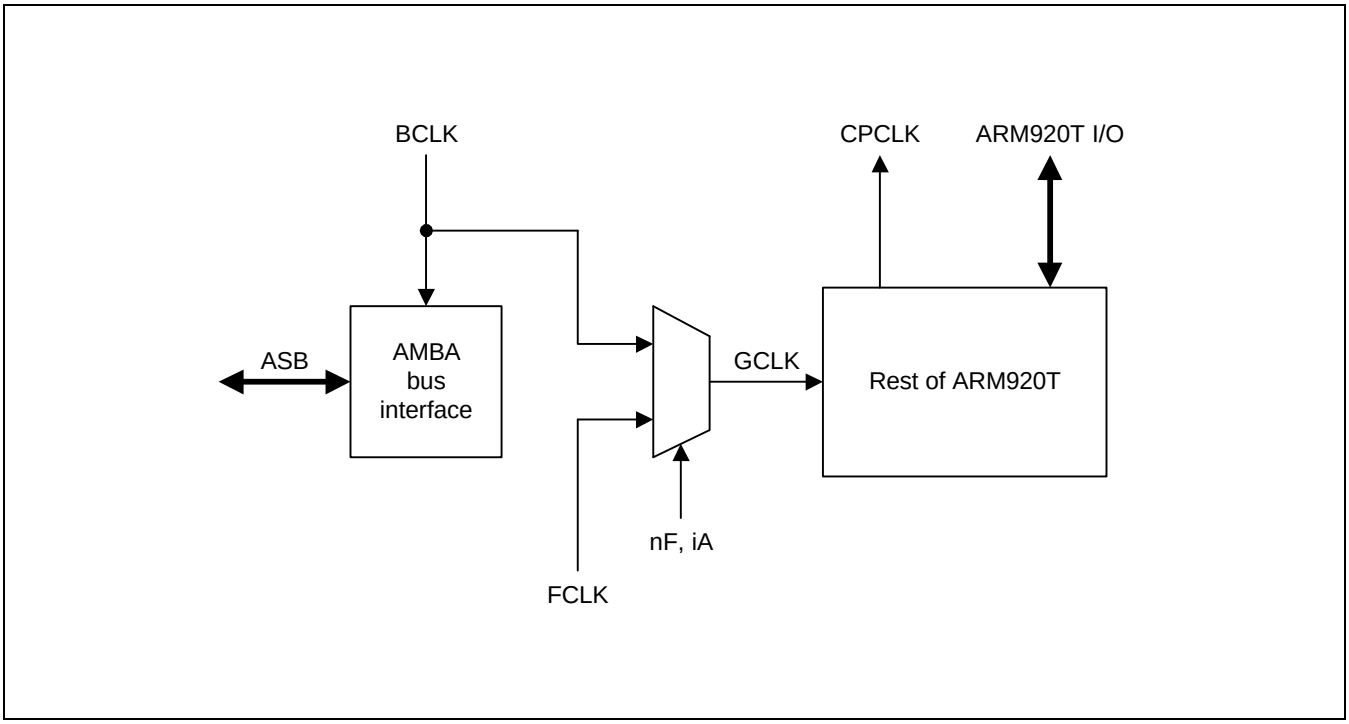


Figure 5-1. ARM920T Clocking

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.

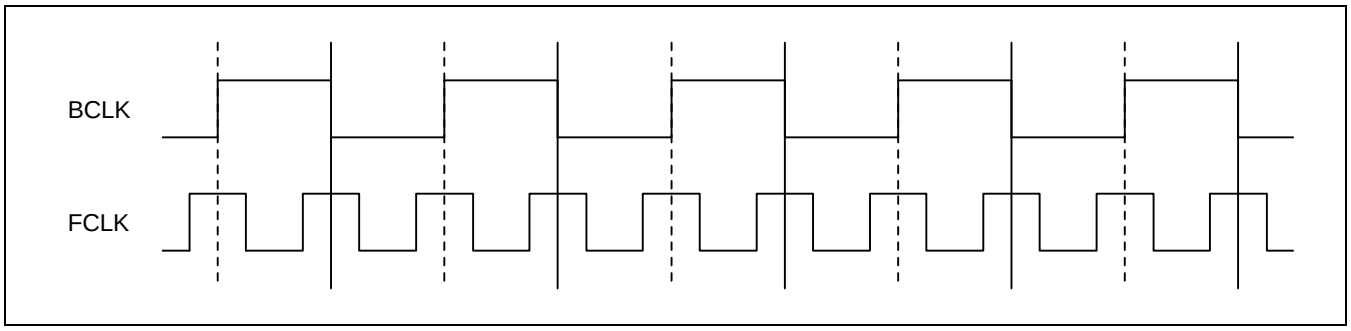


Figure 5-2. Synchronous Clocking Mode

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.

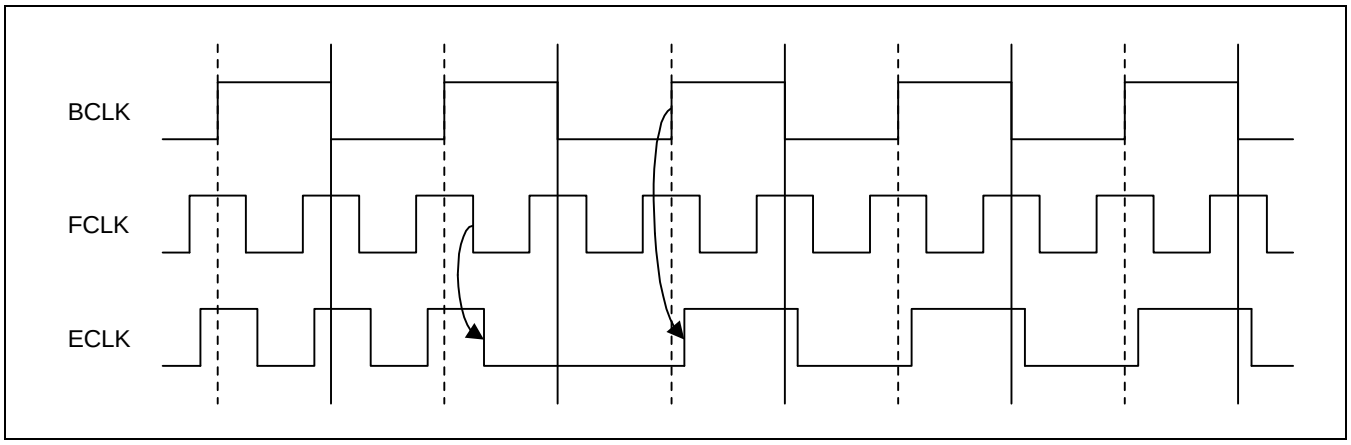


Figure 5-3. Switching from FCLK to BCLK in Synchronous Mode

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.

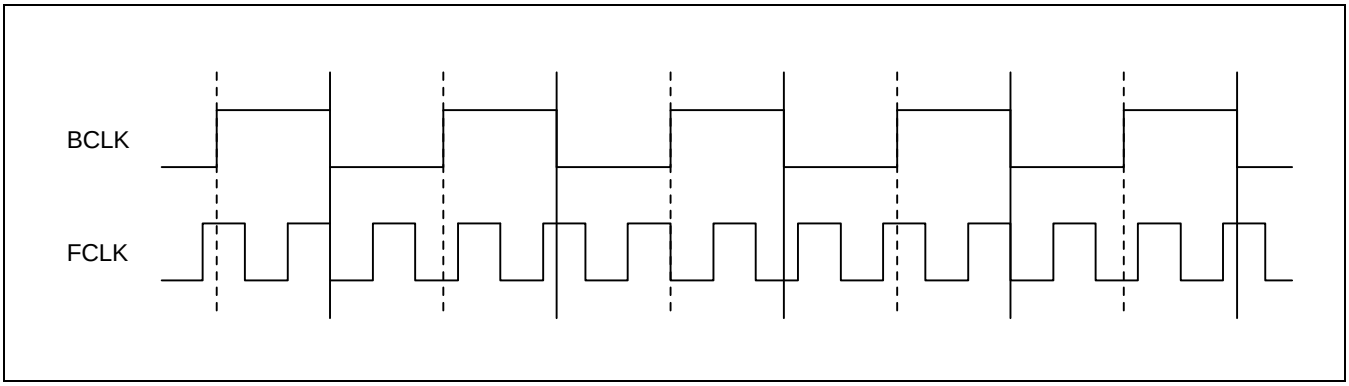


Figure 5-4. Asynchronous Clocking Mode

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.

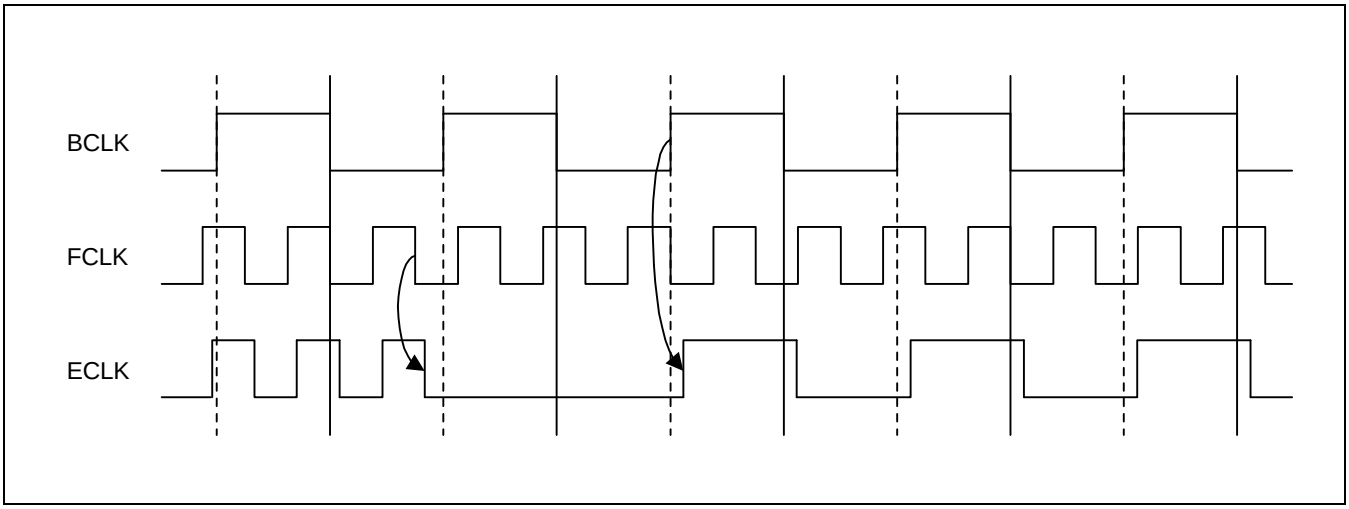


Figure 5-5. Switching from FCLK to BCLK in Asynchronous Mode

NOTES

S3C SERIES MASK ROM ORDER FORM

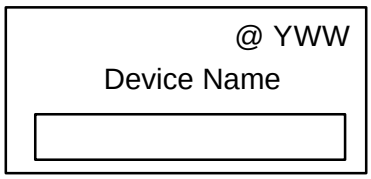
Product description:

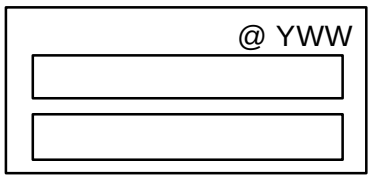
Device Number: S3C_____- _____(write down the ROM code number)

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)





@ : 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.)

S3C2400X MASK OPTION SELECTION FORM

Device Number: S3C2400X -_____(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.)