USER'S MANUAL

S3C44B0X
16/32-Bit RISC
Microprocessor

SAMSUNG
ELECTRONICS

1 PRODUCT OVERVIEW

INTRODUCTION

SAMSUNG's S3C44B0X 16/32-bit RISC microprocessor is designed to provide a cost-effective and high performance micro-controller solution for hand-held devices and general applications. To reduce total system cost, S3C44B0X also provides the following: 8KB cache, optional internal SRAM, LCD controller, 2-channel UART with handshake, 4-channel DMA, System manager (chip select logic, FP/ EDO/SDRAM controller), 5-channel timers with PWM, I/O ports, RTC, 8-channel 10-bit ADC, IIC-BUS interface, IIS-BUS interface, Sync. SIO interface and PLL for clock.

The S3C44B0X was developed using a ARM7TDMI core, 0.25 um CMOS standard cells, and a memory compiler. Its low-power, simple, elegant and fully static design is particularly suitable for cost-sensitive and power sensitive applications. Also S3C44B0X adopts a new bus architecture, SAMBA II (SAMSUNG ARM CPU embedded Microcontroller Bus Architecture).

An outstanding feature of the S3C44B0X is its CPU core, a 16/32-bit ARM7TDMI RISC processor (66MHz) designed by Advanced RISC Machines, Ltd. The architectural enhancements of ARM7TDMI include the Thumb de-compressor, an on-chip ICE breaker debug support, and a 32-bit hardware multiplier.

By providing a complete set of common system peripherals, the S3C44B0X minimizes overall system costs and eliminates the need to configure additional components. The integrated on-chip functions that are described in this document are as follows:

- 2.5V Static ARM7TDMI CPU core with 8KB cache . (SAMBA II bus architecture up to 66MHz)
- External memory controller. (FP/EDO/SDRAM Control, Chip Select logic)
- LCD controller (up to 256 color DSTN) with 1-ch LCD-dedicated DMA.
- 2-ch general DMAs / 2-ch peripheral DMAs with external request pins
- 2-ch UART with handshake(IrDA1.0, 16-byte FIFO) / 1-ch SIO
- 1-ch multi-master IIC-BUS controller
- 1-ch IIS-BUS controller
- 5-ch PWM timers & 1-ch internal timer
- Watch Dog Timer
- 71 general purpose I/O ports / 8-ch external interrupt source
- Power control: Normal, Slow, Idle, and Stop 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 ARM7TDMI CPU core.
- Thumb de-compressor maximizes code density while maintaining performance.
- On-chip ICEbreaker debug support with JTAG-based debugging solution.
- 32x8 bit hardware multiplier.
- New bus architecture to implement Low-Power SAMBA II(SAMSUNG's ARM CPU embedded Micro-controller Bus Architecture).

System Manager

- Little/Big endian support.
- Address space: 32Mbytes per 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(Fast Page, EDO, and 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 & internal SRAM

- 4-way set associative ID(Unified)-cache with 8Kbyte.
- The 0/4/8 Kbytes internal SRAM using unused cache memory.
- Pseudo LRU(Least Recently Used) Replace Algorithm.
- Write through policy to maintain the coherence between main memory and cache content.
- Write buffer with four depth.
- Request data first fill technique when cache miss occurs.

Clock & Power Manager

- Low power
- The on-chip PLL makes the clock for operating MCU at maximum 66MHz.
- Clock can be fed selectively to each function block by software.
- Power mode: Normal, Slow, Idle and Stop 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
- Wake up by EINT[7:0] or RTC alarm interrupt from Stop mode.

Interrupt Controller

- 30 Interrupt sources
  (Watch-dog timer, 6 Timer, 6 UART, 8 External interrupts, 4 DMA , 2 RTC, 1 ADC, 1 IIC, 1 SIO)
- Vectored IRQ interrupt mode to reduce interrupt latency.
- Level/edge mode on the external interrupt sources
- Programmable polarity of edge and level
- Supports FIQ (Fast Interrupt request) for very urgent interrupt request

FEATURES (Continued)

Timer with PWM (Pulse Width Modulation)

- 5-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 for CPU wake-up.
- Time tick interrupt

General purpose input/output ports

- 8 external interrupt ports
- 71 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 (115.2kbps)
- Loop back mode for testing
- Each channel have two internal 32-byte FIFO for Rx and Tx.

DMA Controller

- 2 channel general purpose Direct Memory Access controller without CPU intervention.
- 2 channel Bridge DMA (peripheral DMA) controller.
- Support IO to memory, memory to IO, IO to IO with the Bridge DMA which has 6 type's DMA requestor: Software, 4 internal function blocks (UART, SIO, Timer, IIS), and External pins.
- Programmable priority order between DMAs (fixed or round-robin mode)
- Burst transfer mode to enhance the transfer rate on the FPDRAM, EDODRAM and SDRAM.
- Supports fly-by mode on the memory to external device and external device to memory transfer mode

A/D Converter

- 8-ch multiplexed ADC.
- Max. 100KSPS/10-bit.

LCD Controller

- Supports color/monochrome/gray LCD panel
- Supports single scan and dual scan displays
- Supports virtual screen function
- System memory is used as display memory
- Dedicated DMA for fetching image data from system memory
- Programmable screen size
- Gray level: 16 gray levels
- 256 Color levels

FEATURES (Continued)

Watchdog Timer

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

IIC-BUS Interface

- 1-ch Multi-Master IIC-Bus with interrupt-based operation.
- Serial, 8-bit oriented, 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 MSB-justified data format

SIO (Synchronous Serial I/O)

- 1-ch SIO with DMA-based or interrupt -based operation.
- Programmable baud rates.
- Supports serial data transmit/receive operations 8-bit in SIO.

Operating Voltage Range

- Core : 2.5V I/O : 3.0 V to 3.6 V

Operating Frequency

- Up to 66 MHz

Package

- 160 LQFP / 160 FBGA

BLOCK DIAGRAM

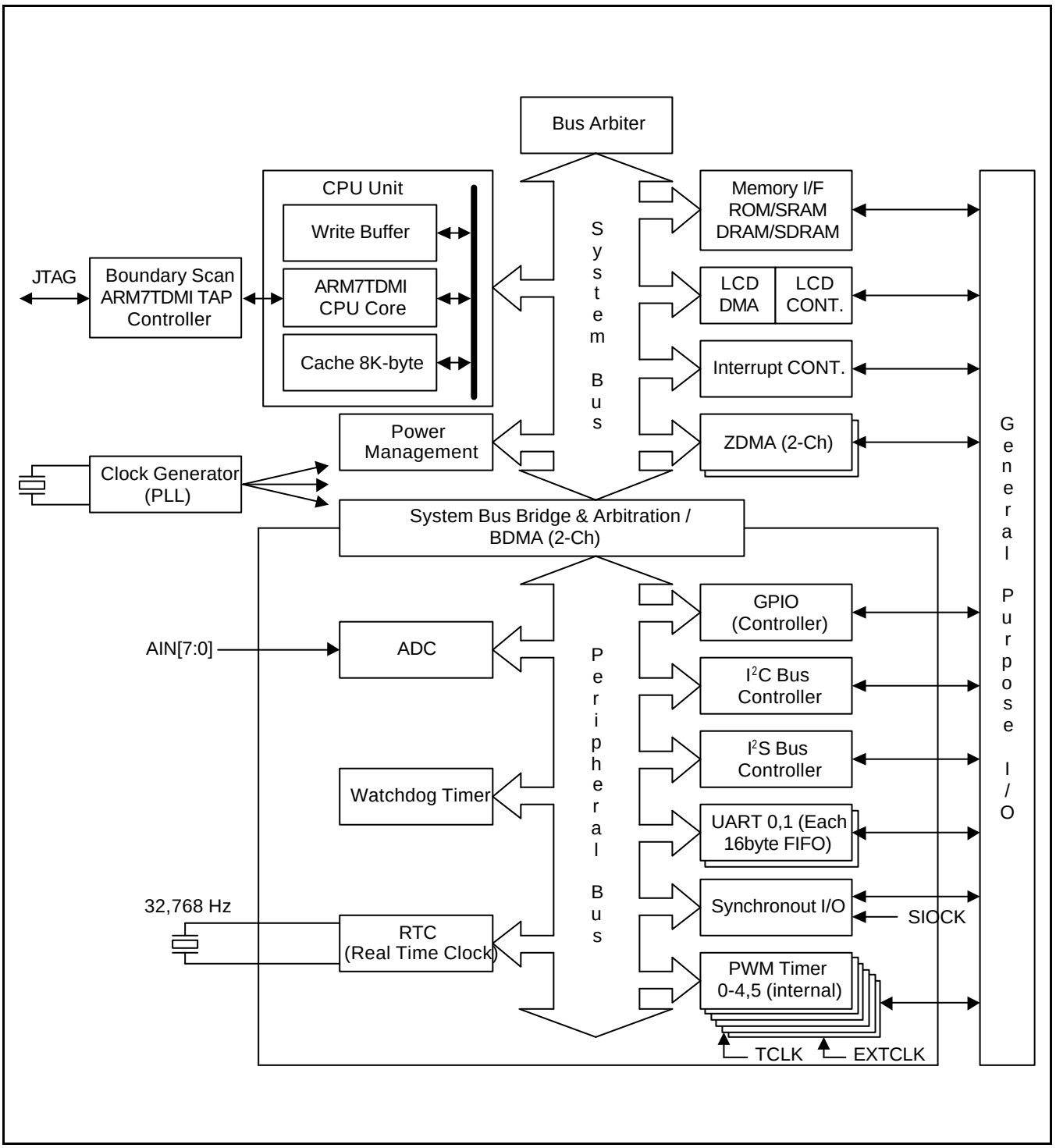


Figure 1-1. S3C44B0X Block Diagram

PIN ASSIGNMENTS

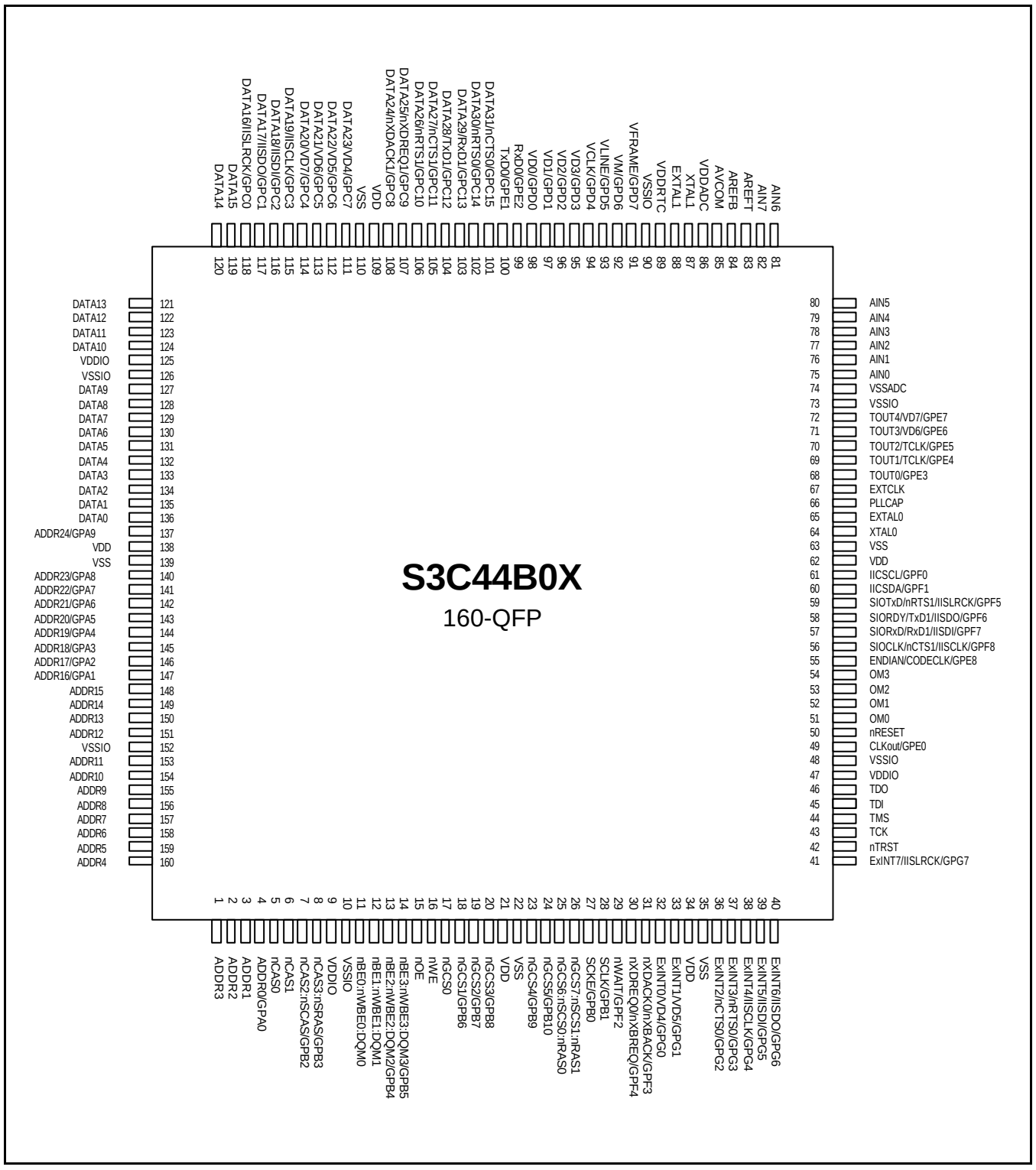


Figure 1-2. S3C44B0X Pin Assignments (160 LQFP)

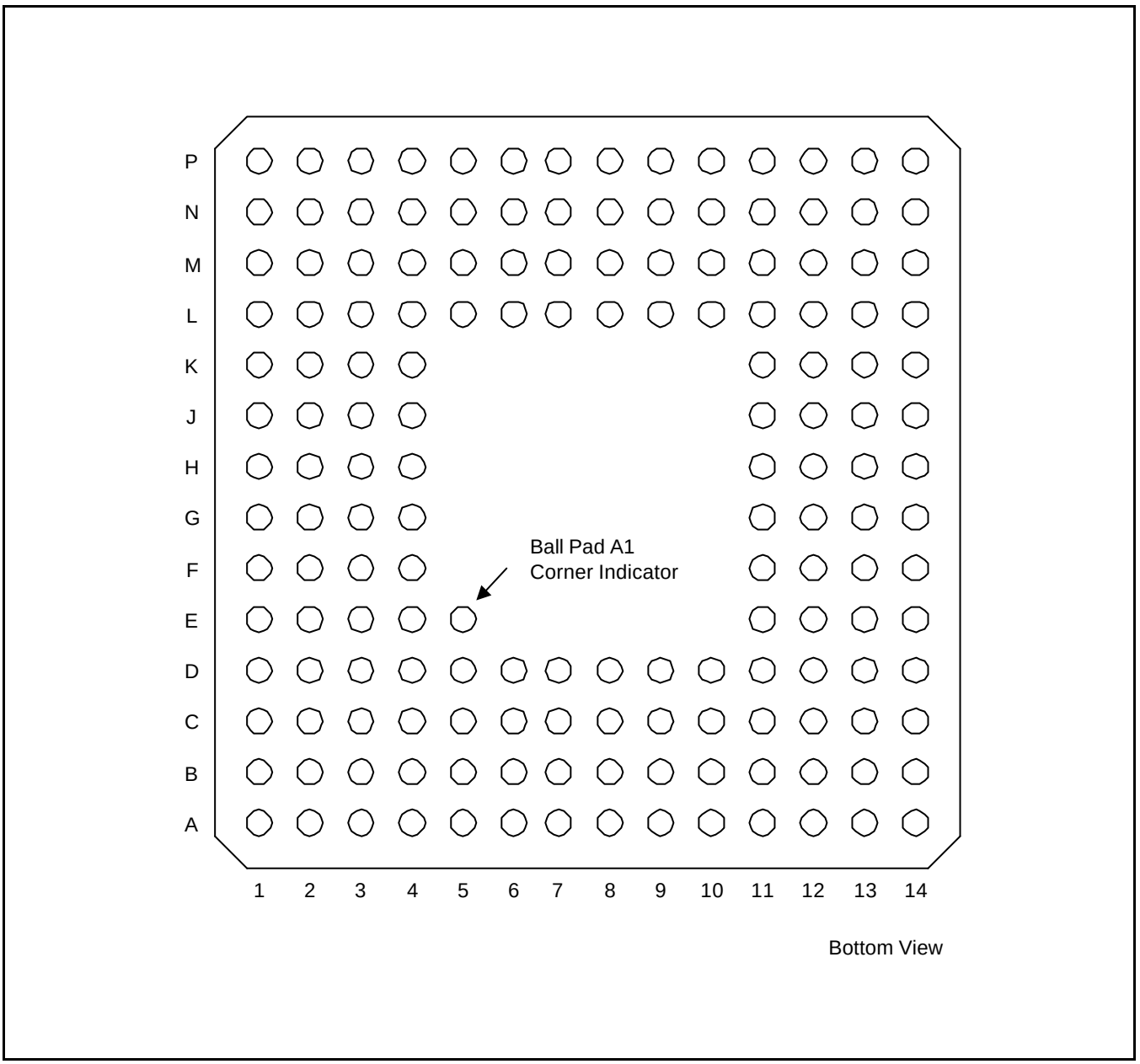


Figure 1-3. S3C44B0X Pin Assignments (160 FBGA)

Table 1-1. 160-Pin LQFP Pin Assignment

Pin No.	Pin Name	Default Function	I/O State(2) @BUS REQ.	I/O State(2) @STOP	I/O State @Initial	I/O TYPE(6)
1	ADDR3	ADDR3	Hi-z	Hi-z	O	phot8
2	ADDR2	ADDR2				
3	ADDR1	ADDR1				
4	ADDR0/GPA0	ADDR0	Hi-z/O	Hi-z/O		
5	nCAS0	nCAS0	Hi-z	Low		
6	nCAS1	nCAS1				
7	nCAS2:nSCAS/GPB2	nSCAS		High/Low/O		
8	nCAS3:nSRAS/GPB3	nSRAS				
9	VDDIO	VDDIO	_(3)	_(3)	P	vdd3op
10	VSSIO	VSSIO				vss3op
11	nBE0:nWBE0:DQM0	DQM0	Hi-z	Hi-z	O	phot6
12	nBE1:nWBE1:DQM1	DQM1				
13	nBE2:nWBE2:DQM2/GPB4	DQM2				
14	nBE3:nWBE3:DQM3/GPB5	DQM3				
15	nOE	nOE				phot8
16	nWE	nWE				phot6
17	nGCS0	nGCS0				phot8
18	nGCS1/GPB6	nGCS1	Hi-z/O	Hi-z/O		
19	nGCS2/GPB7	nGCS2				
20	nGCS3/GPB8	nGCS3				
21	VDD	VDD	–	–	P	vdd2l
22	VSS	VSS				vss2l
23	nGCS4/GPB9	nGCS4	Hi-z/O	Hi-z/O	O	phot8
24	nGCS5/GPB10	nGCS5				
25	nGCS6:nSCS0:nRAS0	nSCS0	Hi-z	High/High/Low		
26	nGCS7:nSCS1:nRAS1	nSCS1				
27	SCKE/GPB0	SCKE	Hi-z/O	Low/O		phot6
28	SCLK/GPB1	SCLK		High/O		phot10
29	nWAIT/GPF2	GPF2	–	–	IO	phbsu50ct8sm
30	nXDREQ0/nXBREQ/GPF4	GPF4				
31	nXDACK0/nXBACK/GPF3	GPF3				
32	ExINT0/VD4/GPG0	GPG0				
33	ExINT1/VD5/GPG1	GPG1				

Table 1-1. 160-Pin LQFP Pin Assignment (Continued)

Pin No.	Pin Name	Default Function	I/O State @BUS REQ.	I/O State @STOP	I/O State @Initial	I/O TYPE
34	VDD	VDD	–	–	P	vdd2i
35	VSS	VSS				vss2i
36	ExINT2/nCTS0/GPG2	GPG2			IO	phbsu50ct8sm
37	ExINT3/nRTS0/GPG3	GPG3				
38	ExINT4/IISCLK/GPG4	GPG4				
39	ExINT5/IISDI/GPG5	GPG5				
40	ExINT6/IISDO/GPG6	GPG6				
41	ExINT7/IISLRCK/GPG7	GPG7				
42	nTRST	nTRST			I	phis
43	TCK	TCK				
44	TMS	TMS				
45	TDI	TDI				
46	TDO	TDO			O	phot6
47	VDDIO	VDDIO			P	vdd3op
48	VSSIO	VSSIO				vss3op
49	CLKout/GPE0	GPE0			IO	phbsu50ct8sm
50	nRESET	nRESET			I	phis
51	OM0	OM0			I(1)	
52	OM1	OM1				
53	OM2	OM2				
54	OM3	OM3				
55	ENDIAN/CODECLK/GPE8	CODECLK			IO(1)	phbsu50ct8sm
56	SIOCLK/nCTS1/IISCLK/GPF8	GPF8				
57	SIORxD/RxD1/IISDI/GPF7	GPF7				
58	SIORDY/TxD1/IISDO/GPF6	GPF6				
59	SIOTxD/nRTS1/IISLRCK/GPF5	GPF5				
60	IICSDA/GPF1	GPF1				phbsu50cd4sm
61	IICSCL/GPF0	GPF0				
62	VDD	VDD			P	vdd2i
63	VSS	VSS				vss2i
64	XTAL0	XTAL0			AI(5)	phsoscm16
65	EXTAL0	EXTAL0			AO(5)	
66	PLLCAP	PLLCAP			AI(5)	phnc50_option

Table 1-1. 160-Pin LQFP Pin Assignment (Continued)

Pin No.	Pin Name	Default Function	I/O State @BUS REQ.	I/O State @STOP	I/O State @Initial	I/O TYPE
67	EXTCLK	EXTCLK	–	–	I	phis
68	TOUT0/GPE3	GPE3			IO	phbsu50ct8sm
69	TOUT1/TCLK/GPE4	GPE4				
70	TOUT2/TCLK/GPE5	GPE5				
71	TOUT3/VD6/GPE6	GPE6				
72	TOUT4/VD7/GPE7	GPE7				
73	VSSIO	VSSIO			P	vss3op
74	VSSADC	VSSADC				vss2t
75	AIN0	AIN0			AI(5)	phnc50
76	AIN1	AIN1				
77	AIN2	AIN2				
78	AIN3	AIN3				
79	AIN4	AIN4				
80	AIN5	AIN5				
81	AIN6	AIN6				
82	AIN7	AIN7				
83	AREFT	AREFT				phnc50_option
84	AREFB	AREFB				
85	AVCOM	AVCOM				
86	VDDADC	VDDADC			P	vdd2t
87	XTAL1	XTAL1			I	phnc50
88	EXTAL1	EXTAL1			O	
89	VDDRTC	VDDRTC			P	vdd2t
90	VSSIO	VSSIO				vss3op
91	VFRAME/GPD7	GPD7			IO	phbsu50ct8sm
92	VM/GPD6	GPD6				
93	VLINE/GPD5	GPD5				
94	VCLK/GPD4	GPD4				
95	VD3/GPD3	GPD3				
96	VD2/GPD2	GPD2				
97	VD1/GPD1	GPD1				
98	VD0/GPD0	GPD0				

Table 1-1. 160-Pin LQFP Pin Assignment (Continued)

Pin No.	Pin Name	Default Function	I/O State @BUS REQ.	I/O State @STOP	I/O State @Initial	I/O TYPE
99	RxD0/GPE2	GPE2	–	–	IO	phbsu50ct8sm
100	TxD0/GPE1	GPE1				
101	DATA31/nCTS0/GPC15	DATA31	Hi-z/IO	Hi-z/IO	I(Hi-z)	phbsu50ct12sm
102	DATA30/nRTS0/GPC14	DATA30				
103	DATA29/RxD1/GPC13	DATA29				
104	DATA28/TxD1/GPC12	DATA28				
105	DATA27/nCTS1/GPC11	DATA27				
106	DATA26/nRTS1/GPC10	DATA26				
107	DATA25/nXDREQ1/GPC9	DATA25				
108	DATA24/nXDACK1/GPC8	DATA24				
109	VDD	VDD	–	–	P	vdd2i
110	VSS	VSS				vss2i
111	DATA23/VD4/GPC7	DATA23	Hi-z/IO	Hi-z/IO	I(Hi-z)	phbsu50ct12sm
112	DATA22/VD5/GPC6	DATA22				
113	DATA21/VD6/GPC5	DATA21				
114	DATA20/VD7/GPC4	DATA20				
115	DATA19/IISCLK/GPC3	DATA19				
116	DATA18/IISDI/GPC2	DATA18				
117	DATA17/IISDO/GPC1	DATA17				
118	DATA16/IISLRCK/GPC0	DATA16				
119	DATA15	DATA15	Hi-z	Hi-z	I(Hi-z)	
120	DATA14	DATA14				
121	DATA13	DATA13				
122	DATA12	DATA12				
123	DATA11	DATA11				
124	DATA10	DATA10				
125	VDDIO	VDDIO	–	–	P	vdd3op
126	VSSIO	VSSIO				vss3op
127	DATA9	DATA9	Hi-z	Hi-z	I(Hi-z)	phbsu50ct12sm
128	DATA8	DATA8				
129	DATA7	DATA7				
130	DATA6	DATA6				

Table 1-1. 160-Pin LQFP Pin Assignment (Concluded)

Pin No.	Pin Name	Default Function	I/O State @BUS REQ.	I/O State @STOP	I/O State @Initial	I/O TYPE
131	DATA5	DATA5	Hi-z	Hi-z	I(Hi-z)	phbsu50ct12sm
132	DATA4	DATA4				
133	DATA3	DATA3				
134	DATA2	DATA2				
135	DATA1	DATA1				
136	DATA0	DATA0				
137	ADDR24/GPA9	ADDR24	Hi-z/O	Hi-z/O	O	phot8
138	VDD	VDD	–	–	P	vdd2i
139	VSS	VSS				vss2i
140	ADDR23/GPA8	ADDR23	Hi-z/O	Hi-z/O	O	phot8
141	ADDR22/GPA7	ADDR22				
142	ADDR21/GPA6	ADDR21				
143	ADDR20/GPA5	ADDR20				
144	ADDR19/GPA4	ADDR19				
145	ADDR18/GPA3	ADDR18				
146	ADDR17/GPA2	ADDR17				
147	ADDR16/GPA1	DATA16				
148	ADDR15	ADDR15	Hi-z	Hi-z		
149	ADDR14	ADDR14				
150	ADDR13	ADDR13				
151	ADDR12	ADDR12				
152	VSSIO	VSSIO	–	–	P	vss3op
153	ADDR11	ADDR11	Hi-z	Hi-z	O	phot8
154	ADDR10	ADDR10				
155	ADDR9	ADDR9				
156	ADDR8	ADDR8				
157	ADDR7	ADDR7				
158	ADDR6	ADDR6				
159	ADDR5	ADDR5				
160	ADDR4	ADDR4				

Table 1-2. 160-Pin FBGA Pin Assignment

Pin No.	Pin Name	Pin No.	Pin Name
A1	ADDR4	C1	ADDR1
A2	ADDR5	C2	ADDR0/GPA0
A3	ADDR6	C3	nCAS0
A4	ADDR10	C4	ADDR8
A5	ADDR13	C5	VSSIO
A6	ADDR17/GPA2	C6	ADDR15
A7	ADDR20/GPA5	C7	ADDR21/GPA6
A8	ADDR23/GPA8	C8	ADDR22/GPA7
A9	DATA0	C9	ADDR24/GPA9
A10	DATA4	C10	DATA3
A11	DATA8	C11	DATA7
A12	DATA11	C12	VDDIO
A13	DATA12	C13	DATA17/IISDO/GPC1
A14	DATA14	C14	DATA16/IISLRCK/GPC0
B1	ADDR2	D1	nCAS3:nSRAS/GPB3
B2	ADDR3	D2	nCAS2:nSCAS/GPB2
B3	ADDR7	D3	VDDIO
B4	ADDR9	D4	nCAS1
B5	ADDR12	D5	ADDR11
B6	ADDR16/GPA1	D6	ADDR14
B7	ADDR19/GPA4	D7	ADDR18/GPA3
B8	VSS	D8	VDD
B9	DATA1	D9	DATA2
B10	DATA5	D10	DATA6
B11	DATA9	D11	VSSIO
B12	DATA10	D12	DATA18/IISDI/GPC2
B13	DATA13	D13	DATA19/IISCLK/GPC3
B14	DATA15	D14	DATA20/VD7/GPC4

Table 1-2. 160-Pin FBGA Pin Assignment (Continued)

Pin No.	Pin Name	Pin No.	Pin Name
E1	nBE1:nWBE1:DQM1	H1	nGCS4/GPB9
E2	nBE0:nWBE0:DQM0	H2	nGCS5/GPB10
E3	nBE2:nWBE2:DQM2/GPB4	H3	VSS
E4	VSSIO	H4	nGCS6:nSCS0:nRAS0
E11	DATA21/VD6/GPC5	H11	VD0/GPD0
E12	DATA22/VD5/GPC6	H12	DATA31/nCTS0/GPC15
E13	DATA23/VD4/GPC7	H13	RxD0/GPE2
E14	VSS	H14	TxD0/GPE1
F1	nWE	J1	nGCS7:nSCS1:nRAS1
F2	nOE	J2	SCKE/GPB0
F3	nGCS0	J3	SCLK/GPB1
F4	nBE3:nWBE3:DQM3/GPB5	J4	nWAIT/GPF2
F11	VDD	J11	VCLK/GPD4
F12	DATA24/nXDACK1/GPC8	J12	VD1/GPD1
F13	DATA25/nXDREQ1/GPC9	J13	VD3/GPD3
F14	DATA26/nRTS1/GPC10	J14	VD2/GPD2
G1	nGCS3/GPB8	K1	nXDREQ0/nXBREQ0/GPF4
G2	nGCS2/GPB7	K2	nXDACK0/nXBACK0/GPF3
G3	VDD	K3	ExINT0/VD4/GPG0
G4	nGCS1/GPB6	K4	ExINT1/VD5/GPG1
G11	DATA27/nCTS1/GPC11	K11	VSSIO
G12	DATA30/nRTS0/GPC14	K12	VLINE/GPD5
G13	DATA28/TxD1/GPC12	K13	VFRAME/GPD7
G14	DATA29/RxD1/GPC13	K14	VM/GPD6

Table 1-2. 160-Pin FBGA Pin Assignment (Continued)

Pin No.	Pin Name	Pin No.	Pin Name
L1	VDD	N1	ExINT5/IISDI/GPG5
L2	VSS	N2	ExINT7/IISLRCK/GPG7
L3	ExINT2/nCTS0/GPG2	N3	TMS
L4	TDO	N4	VDDIO
L5	nRESET	N5	OM0
L6	OM3	N6	ENDIAN/CODECLK/GPE8
L7	SIORDY/TxD1/IISDO/GPF6	N7	SIOTxD/nRTS1/IISLRCK/GPF5
L8	EXTAL0	N8	XTAL0
L9	TOUT1/TCLK/GPE4	N9	EXTCLK
L10	VSSIO	N10	TOUT3/VD6/GPE6
L11	VDDADC	N11	AIN0
L12	VDDRTC	N12	AIN2
L13	XTAL1	N13	AIN6
L14	EXTAL1	N14	AIN7
M1	ExINT4/IISCLK/GPG4	P1	ExINT6/IISDO/GPG6
M2	ExINT3/nRTS0/GPG3	P2	nTRST
M3	TDI	P3	TCK
M4	CLKout/GPE0	P4	VSSIO
M5	OM2	P5	OM1
M6	SIORxD/RxD1/IISDI/GPF7	P6	SIOCLK/nCTS1/IISCLK/GPF8
M7	IICSCL/GPF0	P7	IICSDA/GPF1
M8	VDD	P8	VSS
M9	TOUT0/GPE3	P9	PLLCAP
M10	TOUT4/VD7/GPE7	P10	TOUT2/TCLK/GPE5
M11	AIN1	P11	VSSADC
M12	AVCOM	P12	AIN3
M13	AREFB	P13	AIN4
M14	AREFT	P14	AIN5

NOTES :

1. OM[3:0] and ENDIAN value are latched only at the rising edge of nRESET. Therefore, when nRESET is L, the pins of OM[3:0] and ENDIAN are in input state. After nRESET becomes H, the pin of ENDIAN will be in output state.
2. 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 S3C44B0X is in STOP mode.
3. ' - ' mark indicates the unchanged pin state at STOP mode or Bus released mode.
4. IICSDA,IICSCL pins are open-drain type.
5. AI/AO means analog input/output.

I/O Type	Descriptions
vdd2i, vss2i	2.5V Vdd/Vss for internal logic
vdd3op, vss3op	3.3V Vdd/Vss for external interface logic
vdd2t, vss2t	2.5V Vdd/Vss for analog circuitry
phsoscm16	Oscillator cell with enable and feedback resistor
phbsu50ct12sm	bi-directional pad, CMOS schmitt-trigger, 50KΩ pull-up resistor with control, tri-state, Io=12mA
phbsu50ct8sm	bi-directional pad, CMOS schmitt-trigger, 50KΩ pull-up resistor with control, tri-state, Io=8mA
phbsu50cd4sm	bi-directional pad, CMOS schmitt-trigger, 50KΩ pull-up resistor with control, tri-state, Io=4mA
phot6	output pad, tri-state, Io=6mA
phot8	output pad, tri-state, Io=8mA
phot10	output pad, tri-state, Io=10mA
phis	input pad, CMOS schmitt-trigger
phnc50, phnc50_option	pad for analog pin

SIGNAL DESCRIPTIONS

Table 1-3. S3C44B0X Signal Descriptions

Signal	I/O	Description
BUS CONTROLLER		
OM[1:0]	I	OM[1:0] sets S3C44B0X 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 S3C44B0X 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.
ENDIAN	I	It determines whether or not the data type is little endian or big endian. The logic level is determined by the pull-up/down resistor during the RESET cycle. 0:little endian 1:big endian
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

Table 1-3. S3C44B0X Signal Descriptions (Continued)

Signal	I/O	Description
LCD CONTROL UNIT		
VD[7:0]	O	LCD Data Bus
VFRAME	O	LCD Frame signal
VM	O	VM alternates the polarity of the row and column voltage
VLINE	O	LCD line signal
VCLK	O	LCD clock signal
TIMER/PWM		
TOUT[4:0]	O	Timer output[4:0]
TCLK	I	External clock input
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		
IISLRCK	IO	IIS-bus channel select clock
IISDO	O	IIS-bus serial data output
IISDI	I	IIS-bus serial data input
IISCLK	IO	IIS-bus serial clock
CODECLK	O	CODEC system clock
SIO		
SIORXD	I	SIO receives data input
SIOTXD	O	SIO transmits data output
SIOCK	IO	SIO clock
SIORDY	IO	SIO handshake signal when DMA completes the SIO operation

Table 1-3. S3C44B0X Signal Descriptions (Continued)

Signal	I/O	Description
ADC		
AIN[7:0]	AI	ADC input[7:0]
AREFT	AI	ADC Top Vref
AREFB	AI	ADC Bottom Vref
AVCOM	AI	ADC Common Vref
GENERAL PORT		
P[70:0]	IO	General input/output ports (some ports are output mode only)
RESET & CLOCK		
nRESET	ST	nRESET suspends any operation in progress and places S3C44B0X into a known reset state. For a reset, nRESET must be held to L level for at least 4 MCLK after the processor power has been stabilized.
OM[3:2]	I	OM[3:2] determines how the clock is made. 00 = Crystal(XTAL0,EXTAL0), PLL on 01 = EXTCLK, PLL on 10, 11 = Chip test mode.
EXTCLK	I	External clock source when OM[3:2] = 01b If it isn't used, it has to be H (3.3V).
XTAL0	AI	Crystal Input for internal osc circuit for system clock. If it isn't used, XTAL0 has to be H (3.3V).
EXTAL0	AO	Crystal Output for internal osc circuit for system clock. It is the inverted output of XTAL0. If it isn't used, it has to be a floating pin.
PLLCAP	AI	Loop filter capacitor for system clock PLL. (700pF)
XTAL1	AI	32 KHz crystal input for RTC.
EXTAL1	AO	32 KHz crystal output for RTC. It is the inverted output of XTAL1.
CLKout	O	Fout or Fpllo clock
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.

Table 1-3. S3C44B0X Signal Descriptions (Concluded)

Signal	I/O	Description
POWER		
VDD	P	S3C44B0X core logic VDD (2.5 V)
VSS	P	S3C44B0X core logic VSS
VDDIO	P	S3C44B0X I/O port VDD (3.3 V)
VSSIO	P	S3C44B0X I/O port VSS
RTCVDD	P	RTC VDD (2.5 V or 3.0 V, Not support 3.3V) (This pin must be connected to power properly if RTC isn't used)
VDDADC	P	ADC VDD(2.5 V)
VSSADC	P	ADC VSS

S3C44B0X SPECIAL REGISTERS

Table 1-4. S3C44B0X Special Registers

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
CPU WRAPPER					
SYSCFG	0x01c00000	←	W	R/W	System Configuration
NCACHBE0	0x01c00004				Non Cacheable Area 0
NCACHBE1	0x01c00008				Non Cacheable Area 1
SBUSCON	0x01c40000				System Bus Control
MEMORY CONTROLLER					
BWSCON	0x01c80000	←	W	R/W	Bus Width & Wait Status Control
BANKCON0	0x01c80004				Boot ROM Control
BANKCON1	0x01c80008				BANK1 Control
BANKCON2	0x01c8000c				BANK2 Control
BANKCON3	0x01c80010				BANK3 Control
BANKCON4	0x01c80014				BANK4 Control
BANKCON5	0x01c80018				BANK5 Control
BANKCON6	0x01c8001c				BANK6 Control
BANKCON7	0x01c80020				BANK7 Control
REFRESH	0x01c80024				DRAM/SDRAM Refresh Control
BANKSIZE	0x01c80028				Flexible Bank Size
MRSRB6	0x01c8002c				Mode register set for SDRAM
MRSRB7	0x01c80030				Mode register set for SDRAM

Table 1-4. S3C44B0X Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
UART					
ULCON0	0x01d00000	←	W	R/W	UART 0 Line Control
ULCON1	0x01d04000				UART 1 Line Control
UCON0	0x01d00004				UART 0 Control
UCON1	0x01d04004				UART 1 Control
UFCON0	0x01d00008				UART 0 FIFO Control
UFCON1	0x01d04008				UART 1 FIFO Control
UMCON0	0x01d0000c				UART 0 Modem Control
UMCON1	0x01d0400c				UART 1 Modem Control
UTRSTAT0	0x01d00010			R	UART 0 Tx/Rx Status
UTRSTAT1	0x01d04010				UART 1 Tx/Rx Status
UERSTAT0	0x01d00014				UART 0 Rx Error Status
UERSTAT1	0x01d04014				UART 1 Rx Error Status
UFSTAT0	0x01d00018				UART 0 FIFO Status
UFSTAT1	0x01d04018				UART 1 FIFO Status
UMSTAT0	0x01d0001c				UART 0 Modem Status
UMSTAT1	0x01d0401c				UART 1 Modem Status
UTXH0	0x01d00023	0x01d00020	B	W	UART 0 Transmission Hold
UTXH1	0x01d04023	0x01d04020			UART 1 Transmission Hold
URXH0	0x01d00027	0x01d00024		R	UART 0 Receive Buffer
URXH1	0x01d04027	0x01d04024			UART 1 Receive Buffer
UBRDIV0	0x01d00028	←	W	R/W	UART 0 Baud Rate Divisor
UBRDIV1	0x01d04028				UART 1 Baud Rate Divisor
SIO					
SIOCON	0x01d14000	←	W	R/W	SIO Control
SIODAT	0x01d14004				SIO Data
SBRDR	0x01d14008				SIO Baud Rate Prescaler
ITVCNT	0x01d1400c				SIO Interval Counter
DCNTZ	0x01d14010				SIO DMA Count Zero

Table 1-4. S3C44B0X Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
IIS					
IISCON	0x01d18000,02,03	0x01d18000	B,HW,W	R/W	IIS Control
IISMOD	0x01d18004,06	0x01d18004	HW,W		IIS Mode
IISPSR	0x01d18008,0a,0b	0x01d18008	B,HW,W		IIS Prescaler
IISFIFCON	0x01d1800c,0e	0x01d1800c	HW,W		IIS FIFO Control
IISFIF	0x01d18012	0x01d18010	HW		IIS FIFO Entry
I/O PORT					
PCONA	0x01d20000	←	W	R/W	Port A Control
PDATA	0x01d20004				Port A Data
PCONB	0x01d20008				Port B Control
PDATB	0x01d2000c				Port B Data
PCONC	0x01d20010				Port C Control
PDATC	0x01d20014				Port C Data
PUPC	0x01d20018				Pull-up Control C
PCOND	0x01d2001c				Port D Control
PDATD	0x01d20020				Port D Data
PUPD	0x01d20024				Pull-up Control D
PCONE	0x01d20028				Port E Control
PDATE	0x01d2002c				Port E Data
PUPE	0x01d20030				Pull-up Control E
PCONF	0x01d20034				Port F Control
PDATF	0x01d20038				Port F Data
PUPF	0x01d2003c				Pull-up Control F
PCONG	0x01d20040				Port G Control
PDATG	0x01d20044				Port G Data
PUPG	0x01d20048				Pull-up Control G
SPUCR	0x01d2004c				Special Pull-up
EXTINT	0x01d20050				External Interrupt Control
EXTINPND	0x01d20054				External Interrupt Pending
WATCHDOG TIMER					
WTCON	0x01d30000	←	W	R/W	Watchdog Timer Mode
WTDAT	0x01d30004				Watchdog Timer Data
WTCNT	0x01d30008				Watchdog Timer Count

Table 1-4. S3C44B0X Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
A/D CONVERTER					
ADCCON	0x01d40000,02,03	0x01d40000	B,HW,W	R/W	ADC Control
ADCPSR	0x01d40004,06,07	0x01d40004			ADC Prescaler
ADCDAT	0x01d40008,0a	0x01d40008	HW,W	R	Digitized 10 bit Data
PWM TIMER					
TCFG0	0x01d50000	←	W	R/W	Timer Configuration
TCFG1	0x01d50004				Timer Configuration
TCON	0x01d50008				Timer Control
TCNTB0	0x01d5000c				Timer Count Buffer 0
TCMPB0	0x01d50010				Timer Compare Buffer 0
TCNTO0	0x01d50014			R	Timer Count Observation 0
TCNTB1	0x01d50018			R/W	Timer Count Buffer 1
TCMPB1	0x01d5001c				Timer Compare Buffer 1
TCNTO1	0x01d50020			R	Timer Count Observation 1
TCNTB2	0x01d50024			R/W	Timer Count Buffer 2
TCMPB2	0x01d50028				Timer Compare Buffer 2
TCNTO2	0x01d5002c			R	Timer Count Observation 2
TCNTB3	0x01d50030			R/W	Timer Count Buffer 3
TCMPB3	0x01d50034				Timer Compare Buffer 3
TCNTO3	0x01d50038			R	Timer Count Observation 3
TCNTB4	0x01d5003c			R/W	Timer Count Buffer 4
TCMPB4	0x01d50040				Timer Compare Buffer 4
TCNTO4	0x01d50044			R	Timer Count Observation 4
TCNTB5	0x01d50048			R/W	Timer Count Buffer 5
TCNTO5	0x01d5004c			R	Timer Count Observation 5
IIC					
IICCON	0x01d60000	←	W	R/W	IIC Control
IICSTAT	0x01d60004				IIC Status
IICADD	0x01d60008				IIC Address
IICDS	0x01d6000c				IIC Data Shift

Table 1-4. S3C44B0X Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
RTC					
RTCCON	0x01d70043	0x01d70040	B	R/W	RTC Control
RTCALM	0x01d70053	0x01d70050			RTC Alarm
ALMSEC	0x01d70057	0x01d70054			Alarm Second
ALMMIN	0x01d7005b	0x01d70058			Alarm Minute
ALMHOUR	0x01d7005f	0x01d7005c			Alarm Hour
ALMDAY	0x01d70063	0x01d70060			Alarm Day
ALMMON	0x01d70067	0x01d70064			Alarm Month
ALMYEAR	0x01d7006b	0x01d70068			Alarm Year
RTCRST	0x01d7006f	0x01d7006c			RTC Round Reset
BCDSEC	0x01d70073	0x01d70070			BCD Second
BCDMIN	0x01d70077	0x01d70074			BCD Minute
BCDHOUR	0x01d7007b	0x01d70078			BCD Hour
BCDDAY	0x01d7007f	0x01d7007c			BCD Day
BCDDATE	0x01d70083	0x01d70080			BCD Date
BCDMON	0x01d70087	0x01d70084			BCD Month
BCDYEAR	0x01d7008b	0x01d70088			BCD Year
TICINT	0x01D7008E	0x01D7008C			Tick time count
CLOCK & POWER MANAGEMENT					
PLLCON	0x01d80000	←	W	R/W	PLL Control
CLKCON	0x01d80004				Clock Control
CLKSLOW	0x01d80008				Slow clock Control
LOCKTIME	0x01d8000c				PLL lock time Counter

Table 1-4. S3C44B0X Special Registers (Continued)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/ Write	Function
INTERRUPT CONTROLLER					
INTCON	0x01e00000	←	W	R/W	Interrupt Control
INTPND	0x01e00004			R	Interrupt Request Status
INTMOD	0x01e00008			R/W	Interrupt Mode Control
INTMSK	0x01e0000c				Interrupt Mask Control
I_PSLV	0x01e00010				IRQ Interrupt Previous Slave
I_PMST	0x01e00014				IRQ Interrupt Priority Master
I_CSLV	0x01e00018			R	IRQ Interrupt Current Slave
I_CMST	0x01e0001c				IRQ Interrupt Current Master
I_ISPR	0x01e00020				IRQ Interrupt Pending Status
I_ISPC	0x01e00024			W	IRQ Interrupt Pending Clear
F_ISPR	0x01e00038			R	FIQ Interrupt Pending
F_ISPC	0x01e0003c			W	FIQ Interrupt Pending Clear
LCD CONTROLLER					
LCDCON1	0x01f00000	←	W	R/W	LCD Control 1
LCDCON2	0x01f00004				LCD Control 2
LCDCON3	0x01f00040				LCD Control 3
LCDSADDR1	0x01f00008				Frame Upper Buffer Start Address 1
LCDSADDR2	0x01f0000c				Frame Lower Buffer Start Address 2
LCDSADDR3	0x01f00010				Virtual Screen Address
REDLUT	0x01f00014				RED Lookup Table
GREENLUT	0x01f00018				GREEN Lookup Table
BLUELUT	0x01f0001c				BLUE Lookup Table
DP1_2	0x01f00020				Dithering Pattern duty 1/2
DP4_7	0x01f00024				Dithering Pattern duty 4/7
DP3_5	0x01f00028				Dithering Pattern duty 3/5
DP2_3	0x01f0002c				Dithering Pattern duty 2/3
DP5_7	0x01f00030				Dithering Pattern duty 5/7
DP3_4	0x01f00034				Dithering Pattern duty 3/4
DP4_5	0x01f00038				Dithering Pattern duty 4/5
DP6_7	0x01f0003c				Dithering Pattern duty 6/7
DITHMODE	0x01f00044				Dithering Mode

Table 1-4. S3C44B0X Special Registers (Concluded)

Register Name	Address (B. Endian)	Address (L. Endian)	Acc. Unit	Read/Write	Function
DMA					
ZDCON0	0x01e80000	←	W	R/W	ZDMA0 Control
ZDISRC0	0x01e80004				ZDMA 0 Initial Source Address
ZDIDES0	0x01e80008				ZDMA 0 Initial Destination Address
ZDICNT0	0x01e8000c				ZDMA 0 Initial Transfer Count
ZDCSRC0	0x01e80010			R	ZDMA 0 Current Source Address
ZDCDES0	0x01e80014				ZDMA 0 Current Destination Address
ZDCCNT0	0x01e80018				ZDMA 0 Current Transfer Count
ZDCON1	0x01e80020			R/W	ZDMA 1 Control
ZDISRC1	0x01e80024				ZDMA 1 Initial Source Address
ZDIDES1	0x01e80028				ZDMA 1 Initial Destination Address
ZDICNT1	0x01e8002c				ZDMA 1 Initial Transfer Count
ZDCSRC1	0x01e80030			R	ZDMA 1 Current Source Address
ZDCDES1	0x01e80034				ZDMA 1 Current Destination Address
ZDCCNT1	0x01e80038				ZDMA 1 Current Transfer Count
BDCON0	0x01f80000			R/W	BDMA 0 Control
BDISRC0	0x01f80004				BDMA 0 Initial Source Address
BDIDES0	0x01f80008				BDMA 0 Initial Destination Address
BDICNT0	0x01f8000c				BDMA 0 Initial Transfer Count
BDCSRC0	0x01f80010			R	BDMA 0 Current Source Address
BDCDES0	0x01f80014				BDMA 0 Current Destination Address
BDCCNT0	0x01f80018				BDMA 0 Current Transfer Count
BDCON1	0x01f80020			R/W	BDMA 1 Control
BDISRC1	0x01f80024				BDMA 1 Initial Source Address
BDIDES1	0x01f80028				BDMA 1 Initial Destination Address
BDICNT1	0x01f8002c				BDMA 1 Initial Transfer Count
BDCSRC1	0x01f80030			R	BDMA 1 Current Source Address
BDCDES1	0x01f80034				BDMA 1 Current Destination Address
BDCCNT1	0x01f80038				BDMA 1 Current Transfer Count

IMPORTANT NOTES ABOUT S3C44B0X 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 *).

2 PROGRAMMER'S MODEL

OVERVIEW

S3C44B0X has been developed using the advanced ARM7TDMI core, which has been designed by Advanced RISC Machines, Ltd.

PROCESSOR OPERATING STATES

From the programmer's point of view, the ARM7TDMI can be in one of two states:

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

NOTE

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

SWITCHING STATE

Entering THUMB State

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

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

Entering ARM State

Entry into ARM state happens:

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

MEMORY FORMATS

ARM7TDMI 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. ARM7TDMI 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.

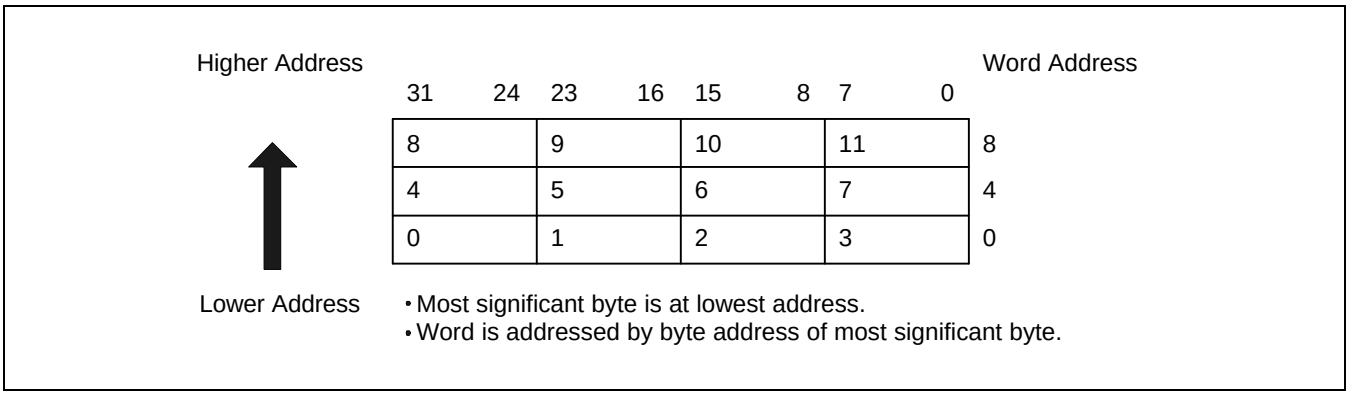


Figure 2-1. Big-Endian Addresses of Bytes within Words

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.

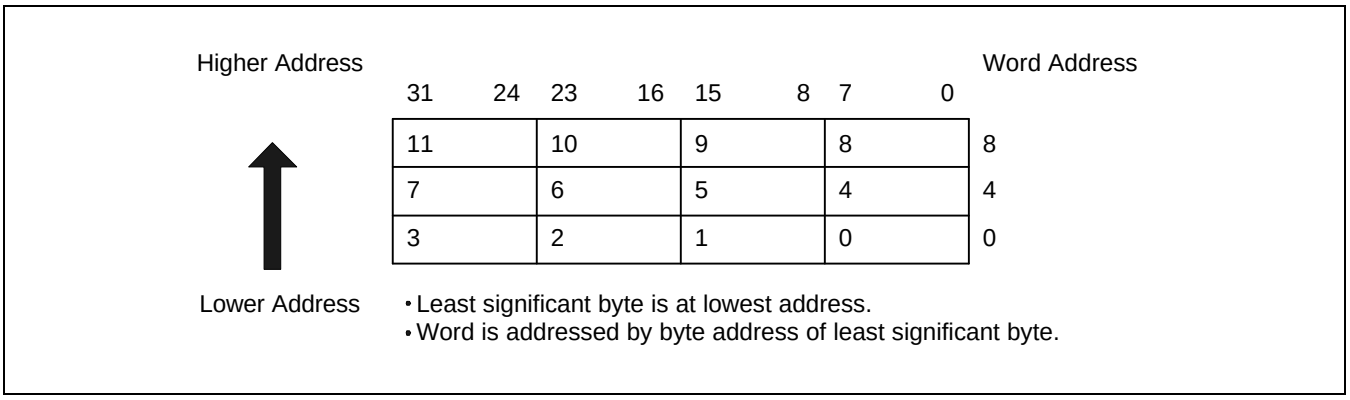


Figure 2-2. Little-Endian Addresses of Bytes whthin Words

INSTRUCTION LENGTH

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

Data Types

ARM7TDMI 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

ARM7TDMI 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

ARM7TDMI 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.

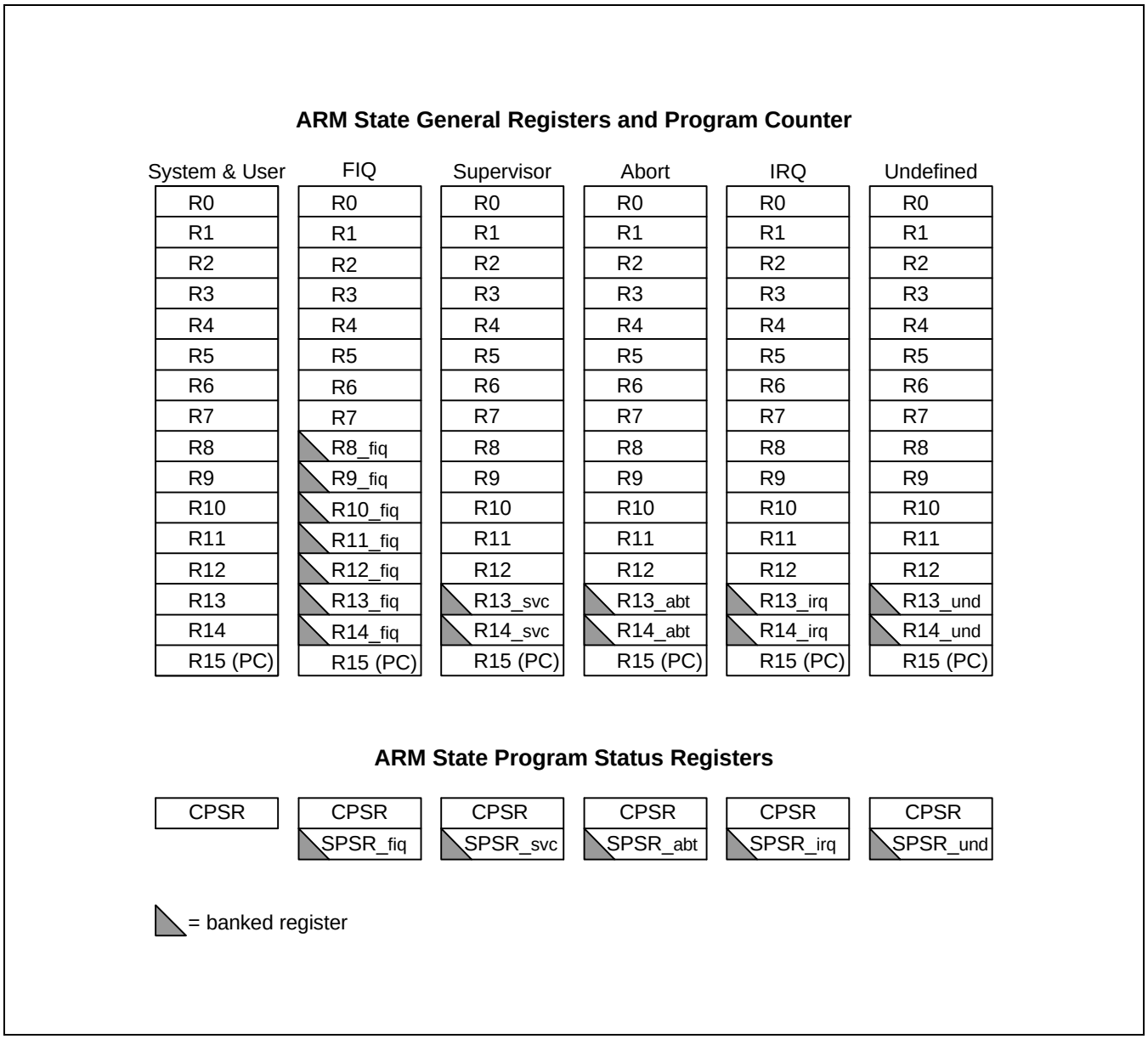


Figure 2-3. Register Organization in ARM State

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.

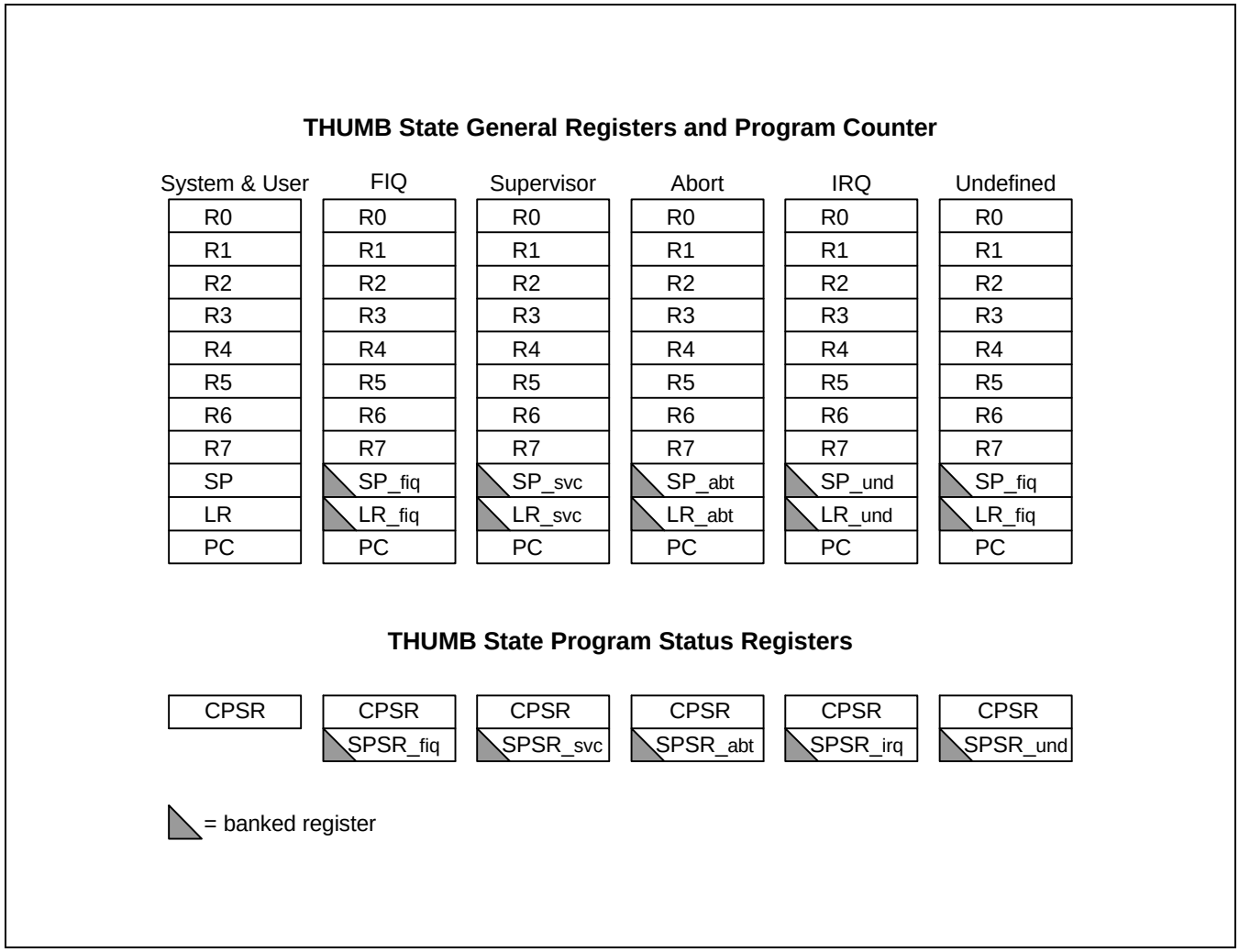


Figure 2-4. Register Organization in THUMB State

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.

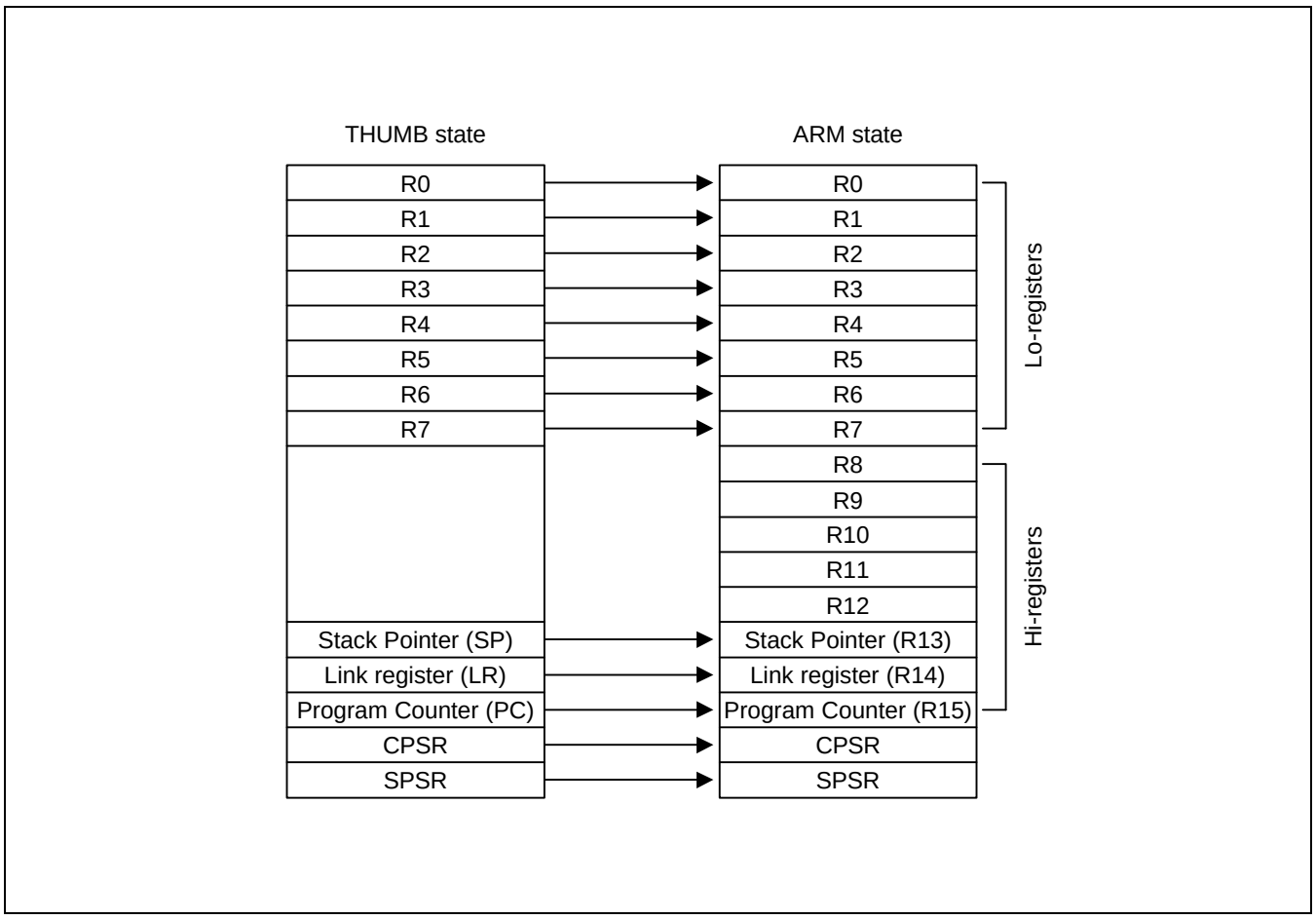


Figure 2-5. Mapping of THUMB State Registers onto ARM State Registers

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 ARM7TDMI 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.

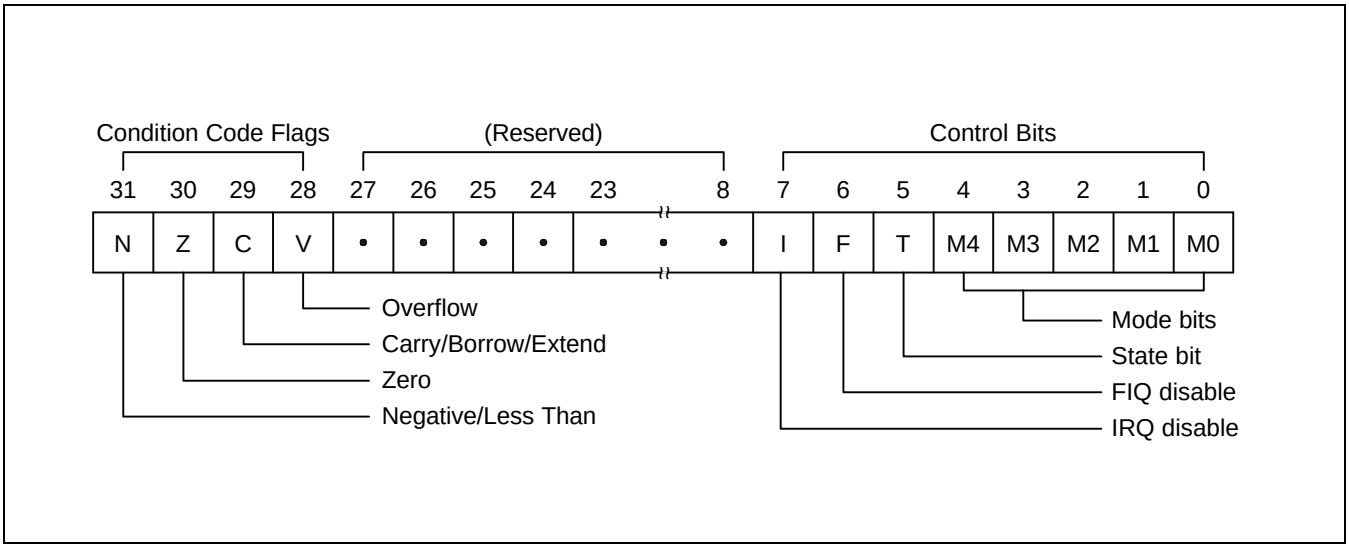


Figure 2-6. Program Status Register Format

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 ARM7TDMI:

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

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

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

Action on Leaving an Exception

On completion, the exception handler:

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

NOTE

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

Exception Entry/Exit Summary

Table 2-2 summarises 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 minimising 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, ARM7TDMI checks for a LOW level on the output of the FIQ synchroniser 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 signalled by the external **ABORT** input. ARM7TDMI checks for the abort exception during memory access cycles.

There are two types of abort:

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

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

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

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

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

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

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

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

Software Interrupt

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

```
MOV     PC,R14_svc
```

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

NOTE

nFIQ, nIRQ, ISYNC, LOCK, BIGEND, and ABORT pins exist only in the ARM7TDMI CPU core.

Undefined Instruction

When ARM7TDMI 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 Priorites

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), ARM7TDMI 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 synchroniser (*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 ARM7TDMI 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 synchroniser (*Tsyncmin*) plus *Tfiq*. This is 4 processor cycles.

RESET

When the **nRESET** signal goes LOW, ARM7TDMI abandons the executing instruction and then continues to fetch instructions from incrementing word addresses.

When **nRESET** goes HIGH again, ARM7TDMI:

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 INSTRUCTION SET

INSTRUCTION SET SUMMAY

This chapter describes the ARM instruction set and the THUMB instruction set in the ARM7TDMI core.

FORMAT SUMMARY

The ARM instruction set formats are shown below.

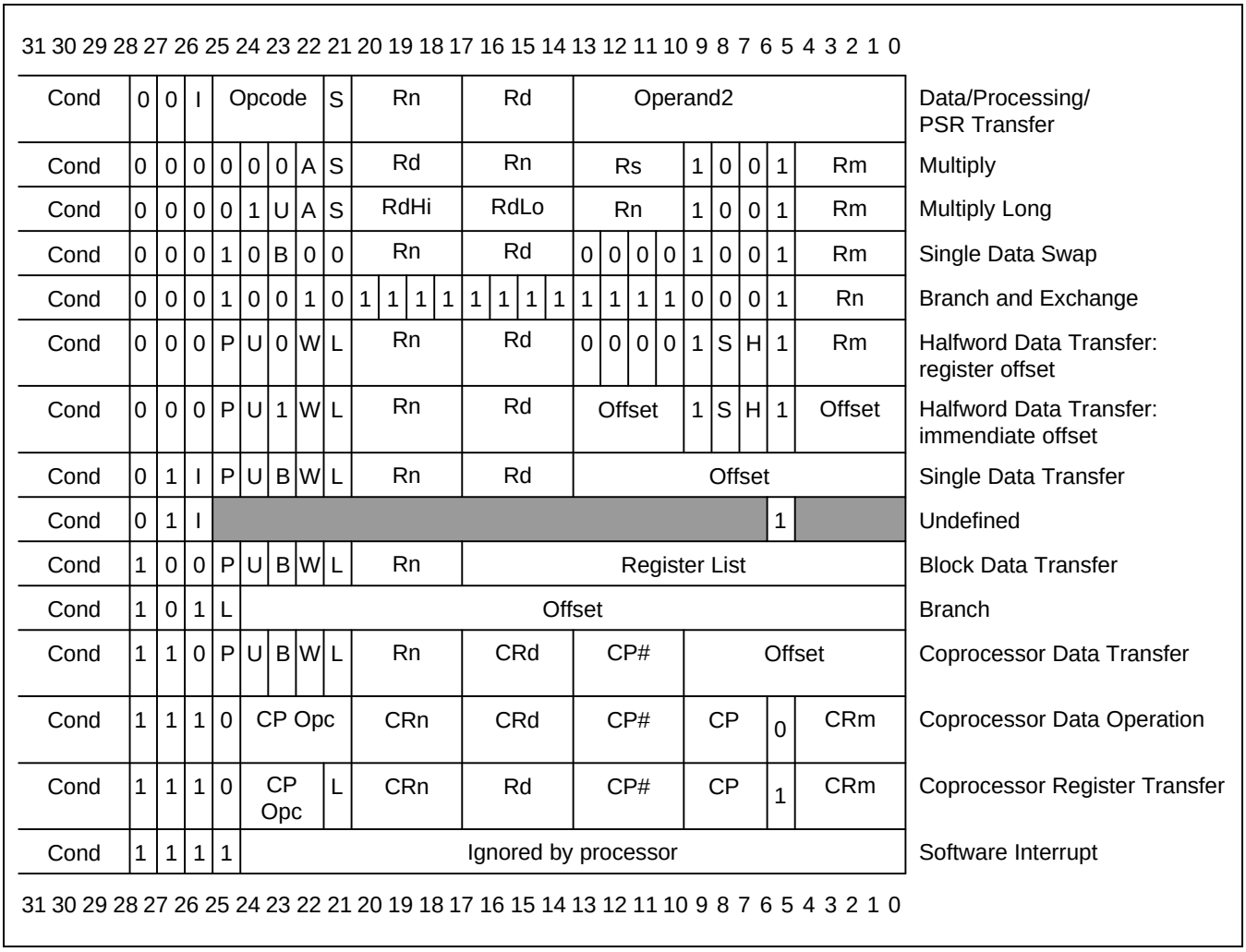


Figure 3-1. ARM Instruction Set Format

NOTE

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

INSTRUCTION SUMMARY

Table 3-1. The ARM Instruction Set

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

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

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

THE CONDITION FIELD

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

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

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

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

Table 3-2. Condition Code Summary

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

BRANCH AND EXCHANGE (BX)

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

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

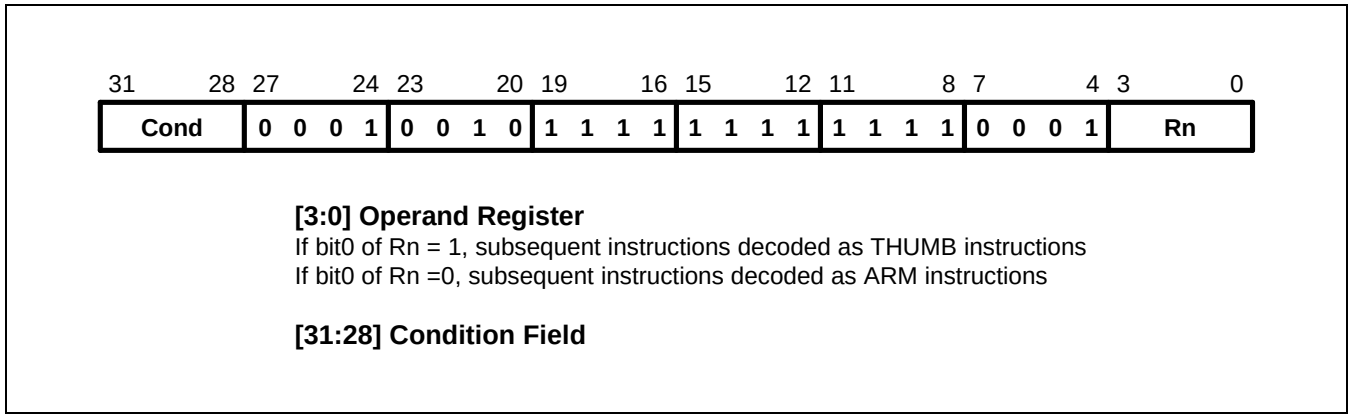


Figure 3-2. Branch and Exchange Instructions

INSTRUCTION CYCLE TIMES

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

ASSEMBLER SYNTAX

BX - branch and exchange.

BX {cond} Rn

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

USING R15 AS AN OPERAND

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

Examples

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

BRANCH AND BRANCH WITH LINK (B, BL)

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

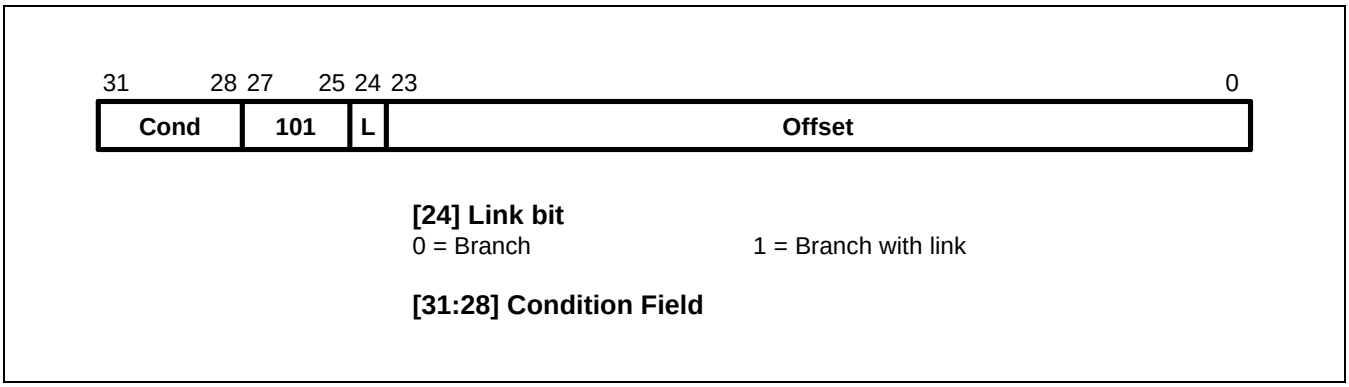


Figure 3-3. Branch Instructions

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

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

THE LINK BIT

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

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

INSTRUCTION CYCLE TIMES

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

ASSEMBLER SYNTAX

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

B{L}{cond} <expression>

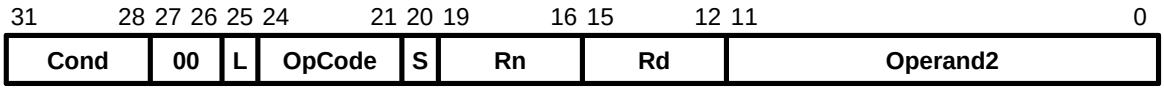
{L}	Used to request the Branch with Link form of the instruction. If absent, R14 will not be affected by the instruction.
{cond}	A two-character mnemonic as shown in Table 3-2. If absent then AL (ALways) will be used.
<expression>	The destination. The assembler calculates the offset.

EXAMPLES

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

DATA PROCESSING

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



[15:12] Destination register
0 = Branch 1 = Branch with link

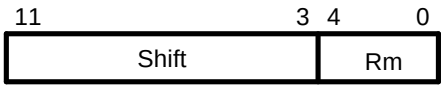
[19:16] 1st operand register
0 = Branch 1 = Branch with link

[20] Set condition codes
0 = Do not after condition codes 1 = Set condition codes

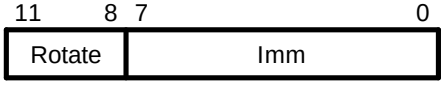
[24:21] Operation codes
0000 = AND-Rd: = Op1 AND Op2
0001 = EOR-Rd: = Op1 EOR Op2
0010 = SUB-Rd: = Op1-Op2
0011 = RSB-Rd: = Op2-Op1
0100 = ADD-Rd: = Op1+Op2
0101 = ADC-Rd: = Op1+Op2+C
0110 = SBC-Rd: = OP1-Op2+C-1
0111 = RSC-Rd: = Op2-Op1+C-1
1000 = TST-set condition codes on Op1 AND Op2
1001 = TEO-set condition codes on OP1 EOR Op2
1010 = CMP-set condition codes on Op1-Op2
1011 = SMN-set condition codes on Op1+Op2
1100 = ORR-Rd: = Op1 OR Op2
1101 = MOV-Rd: =Op2
1110 = BIC-Rd: = Op1 AND NOT Op2
1111 = MVN-Rd: = NOT Op2

[25] Immediate operand
0 = Operand 2 is a register 1 = Operand 2 is an immediate value

[11:0] Operand 2 type selection



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



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

[31:28] Condition field

Figure 3-4. Data Processing Instructions

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

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

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

CPSR FLAGS

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

Table 3-3. ARM Data Processing Instructions

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

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

SHIFTS

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

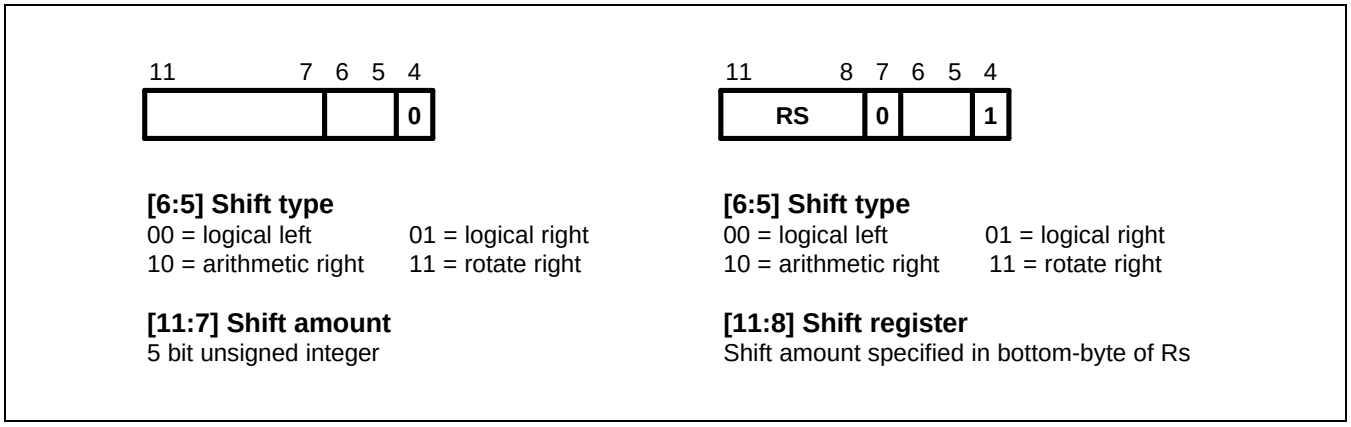


Figure 3-5. ARM Shift Operations

Instruction specified shift amount

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

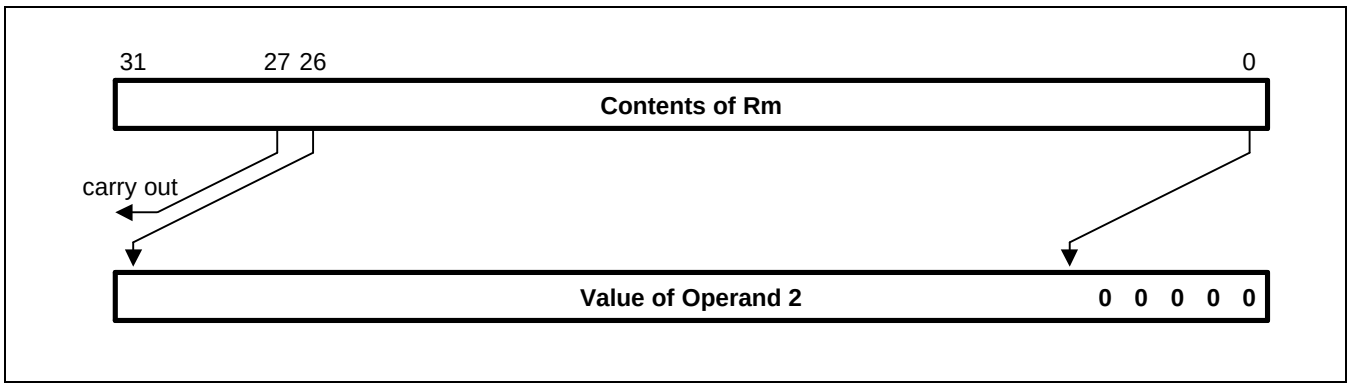


Figure 3-6. Logical Shift Left

NOTE

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

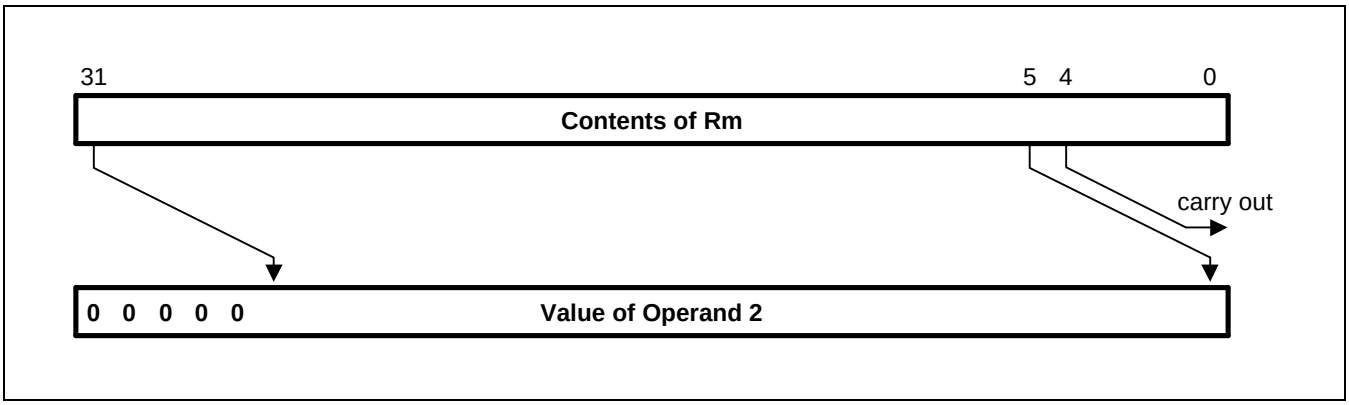


Figure 3-7. Logical Shift Right

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

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

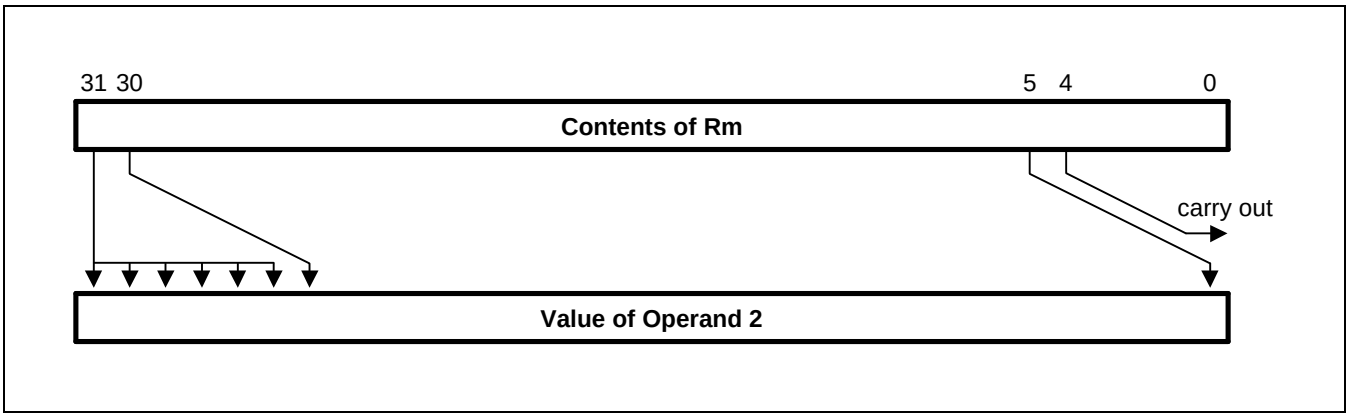


Figure 3-8. Arithmetic Shift Right

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

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

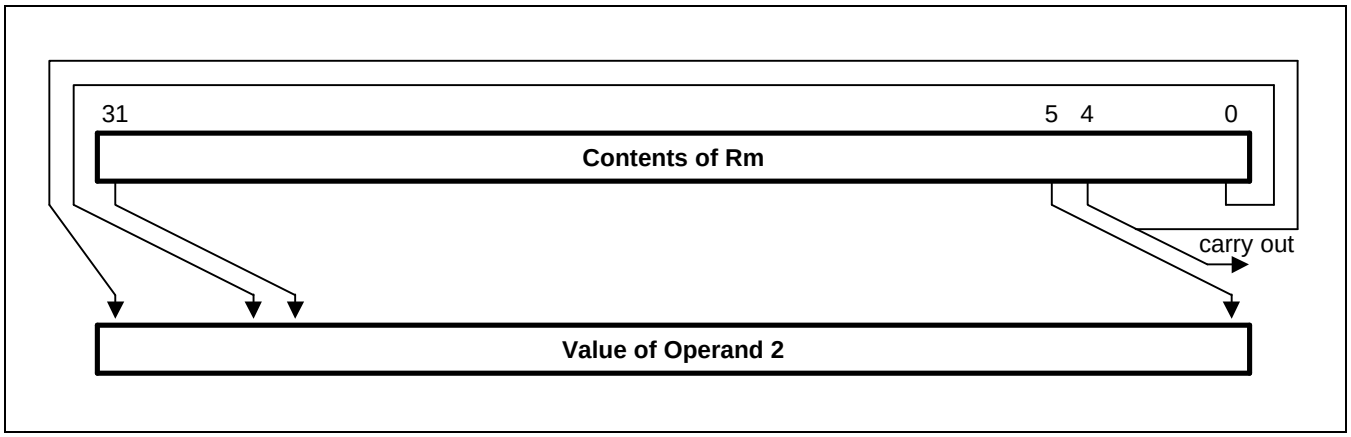


Figure 3-9. Rotate Right

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

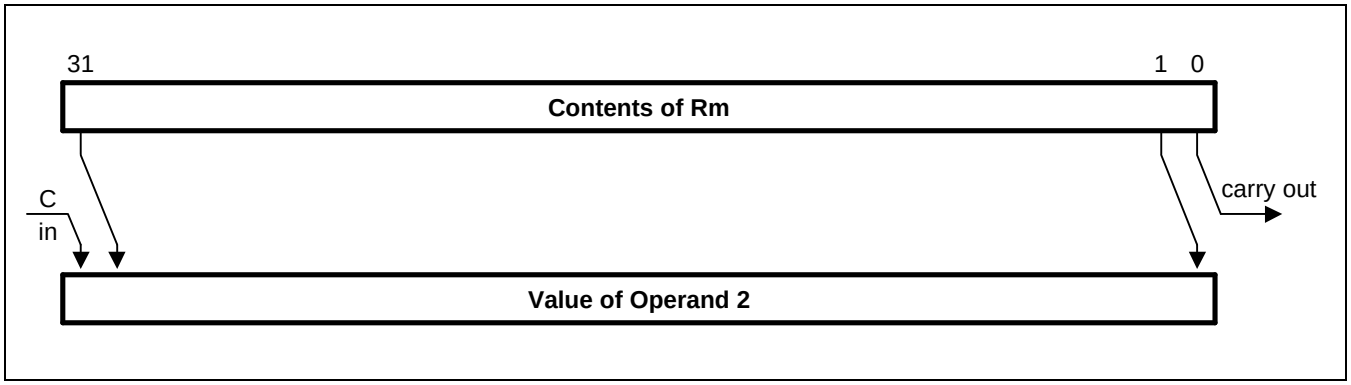


Figure 3-10. Rotate Right Extended

Register specified shift amount

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

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

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

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

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

NOTE

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

IMMEDIATE OPERAND ROTATES

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

WRITING TO R15

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

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

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

USING R15 AS AN OPERANDY

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

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

TEQ, TST, CMP AND CMN OPCODES

NOTE

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

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

The action of TEQP in the ARM7TDMI 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.

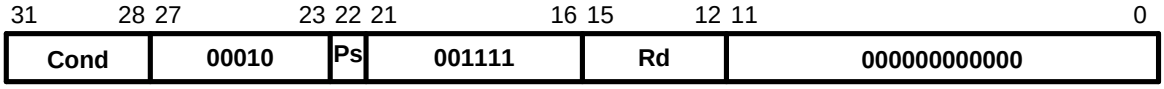
These instructions allow access to the CPSR and SPSR registers. The MRS instruction allows the contents of the CPSR or SPSR_<mode> to be moved to a general register. The MSR instruction allows the contents of a general register to be moved to the CPSR or SPSR_<mode> register.

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)

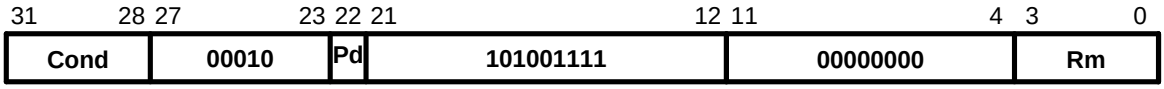


[15:21] Destination Register

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

[31:28] Condition Field

MRS (transfer register contents to PSR)

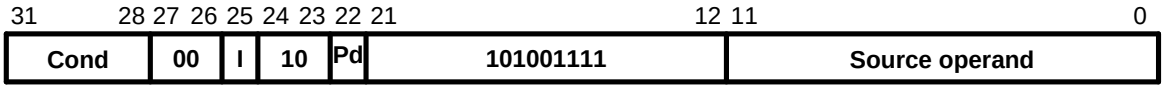


[3:0] Source Register

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

[31:28] Condition Field

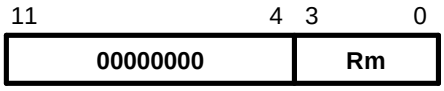
MRS (transfer register contents or immediate value to PSR flag bits only)



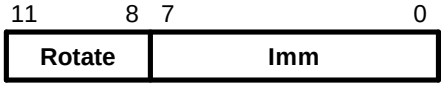
[22] Destination PSR
0 = CPSR 1 = SPSR_<current mode>

[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 ARM7TDMI (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 ARM7TDMI programs and future processors, the following rules should be observed:

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

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

EXAMPLES

The following sequence performs a mode change:

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

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

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

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

INSTRUCTION CYCLE TIMES

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

ASSEMBLY SYNTAX

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

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

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

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

Key:

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

EXAMPLES

In User mode the instructions behave as follows:

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

In privileged modes the instructions behave as follows:

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

MULTIPLY AND MULTIPLY-ACCUMULATE (MUL, MLA)

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

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

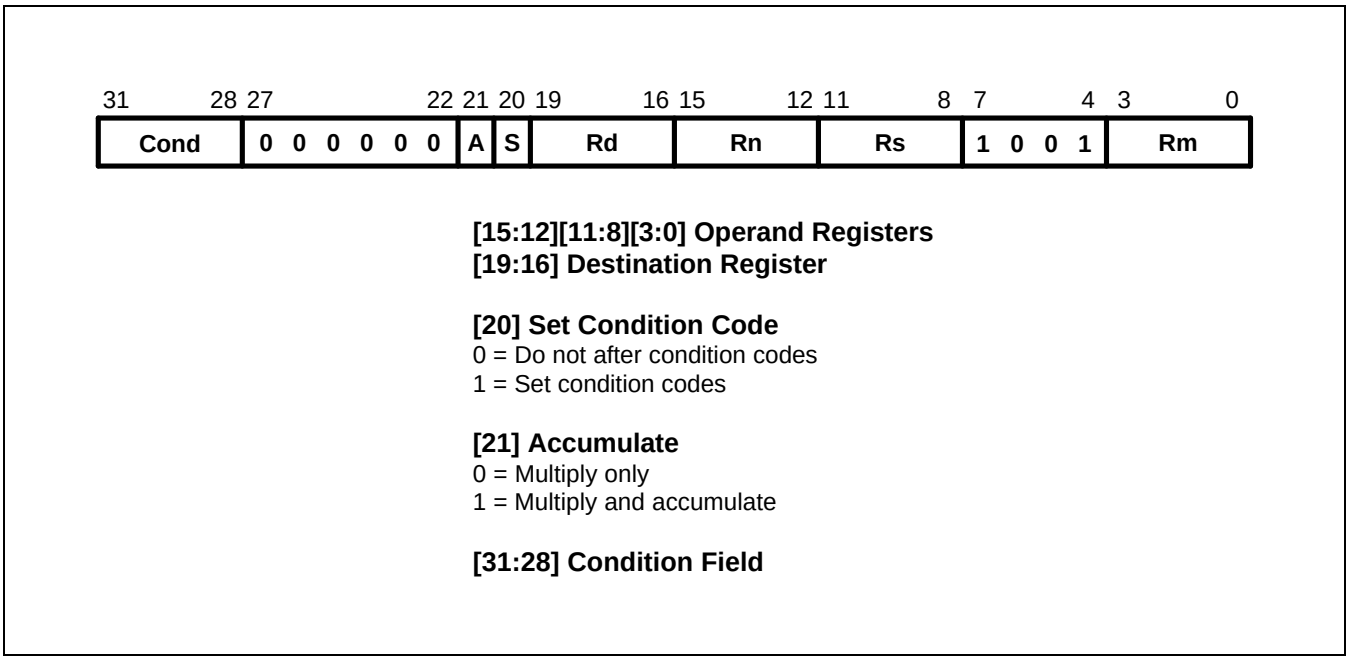


Figure 3-12. Multiply Instructions

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

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

For example consider the multiplication of the operands:

Operand A	Operand B	Result
0xFFFFFFF6	0x0000001	0xFFFFFF38

If the Operands Are Interpreted as Signed

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

If the Operands Are Interpreted as Unsigned

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

Operand Restrictions

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

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

CPSR FLAGS

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

INSTRUCTION CYCLE TIMES

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

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

ASSEMBLER SYNTAX

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

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

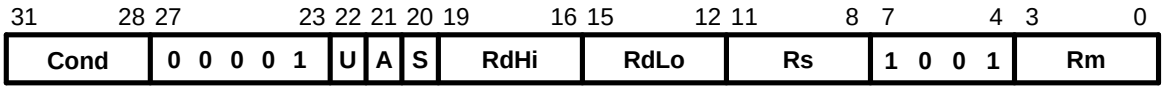
EXAMPLES

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

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

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

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



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

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

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

[22] Unsigned
0 = Unsigned
1 = Signed

[31:28] Condition Field

Figure 3-13. Multiply Long Instructions

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

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

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

OPERAND RESTRICTIONS

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

CPSR FLAGS

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

INSTRUCTION CYCLE TIMES

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

Its possible values are as follows:

For Signed INSTRUCTIONS SMULL, SMLAL:

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

For Unsigned Instructions UMULL, UMLAL:

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

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

ASSEMBLER SYNTAX

Table 3-5. Assembler Syntax Descriptions

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

where:

{cond} Two-character condition mnemonic. See Table 3-2.

{S} Set condition codes if S present

RdLo, RdHi, Rm, Rs Expressions evaluating to a register number other than R15.

EXAMPLES

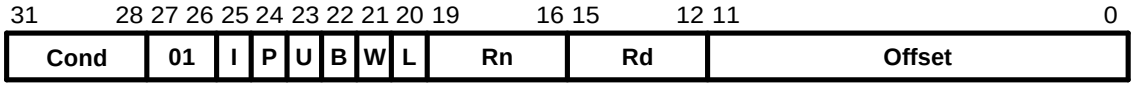
```
UMULL   R1,R4,R2,R3     ; R4,R1:=R2*R3
UMLALS  R1,R5,R2,R3     ; R5,R1:=R2*R3+R5,R1 also setting condition codes
```

SINGLE DATA TRANSFER (LDR, STR)

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

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

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



[15:12] Source/Destination Registers

[19:16] Base Register

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

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

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

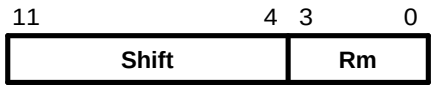
[23] Up/Down Bit
0 = Down: subtract offset from base
1 = Up: add offset to base

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

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

[11:0] Offset

[11:0] Unsigned 12-bit immediate offset



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

[31:28] Condition Field

Figure 3-14. Single Data Transfer Instructions

OFFSETS AND AUTO-INDEXING

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

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

SHIFTED REGISTER OFFSET

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

BYTES AND WORDS

This instruction class may be used to transfer a byte (B=1) or a word (B=0) between an ARM7TDMI register and memory.

The action of LDR(B) and STR(B) instructions is influenced by the **BIGEND** control signal of ARM7TDMI 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.

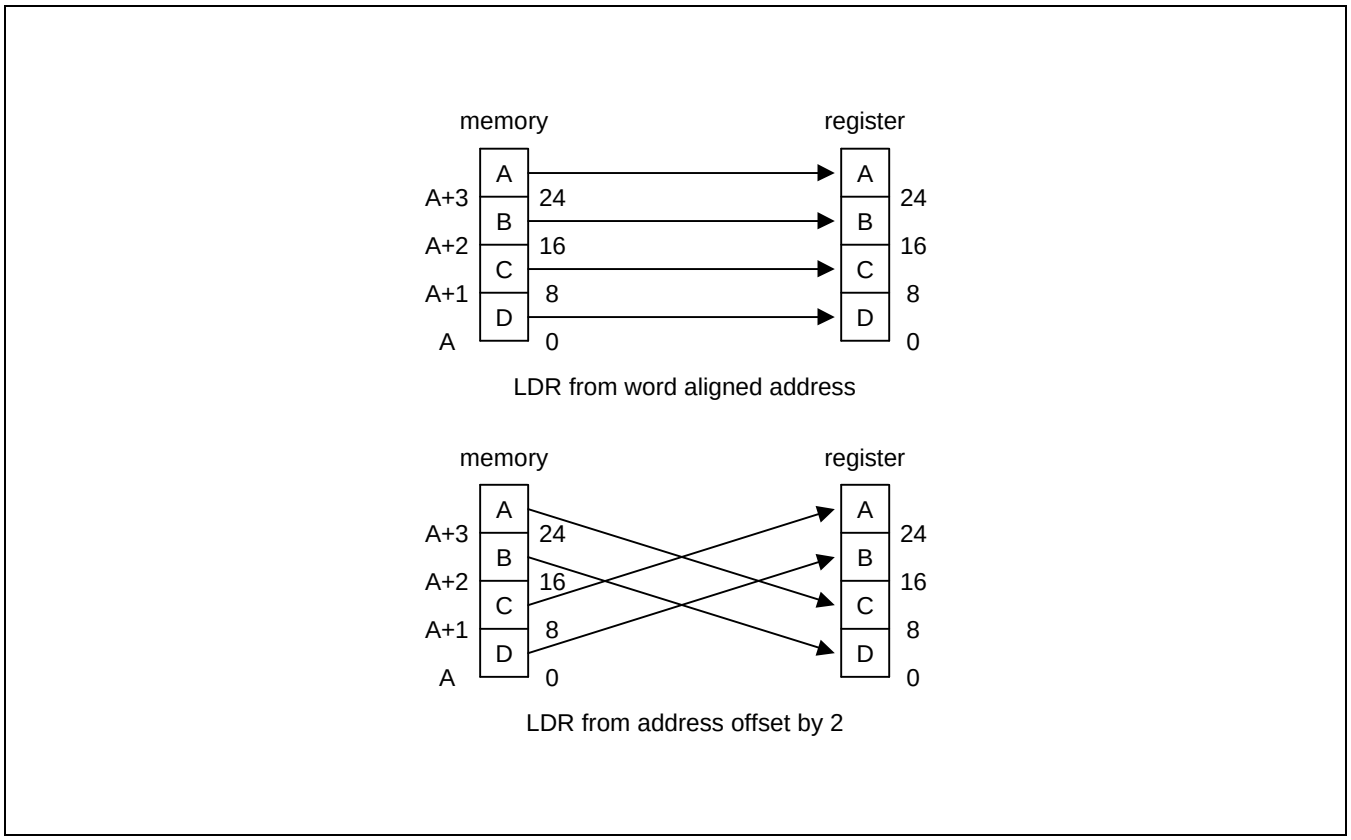


Figure 3-15. Little-Endian Offset Addressing

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 ARM7TDMI 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.	

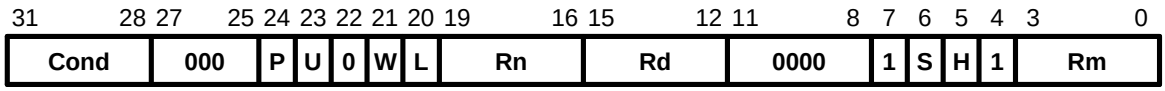
EXAMPLES

```
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

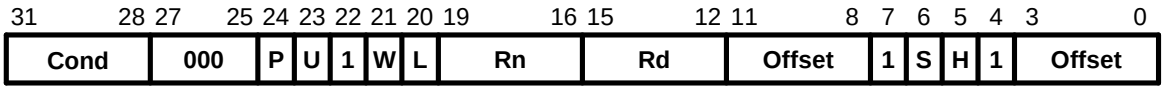
[21] Write-back
0 = No write-back
1 = Write address into base

[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 ARM7TDMI 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 ARM7TDMI 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 ARM7TDMI 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 ARM7TDMI 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.

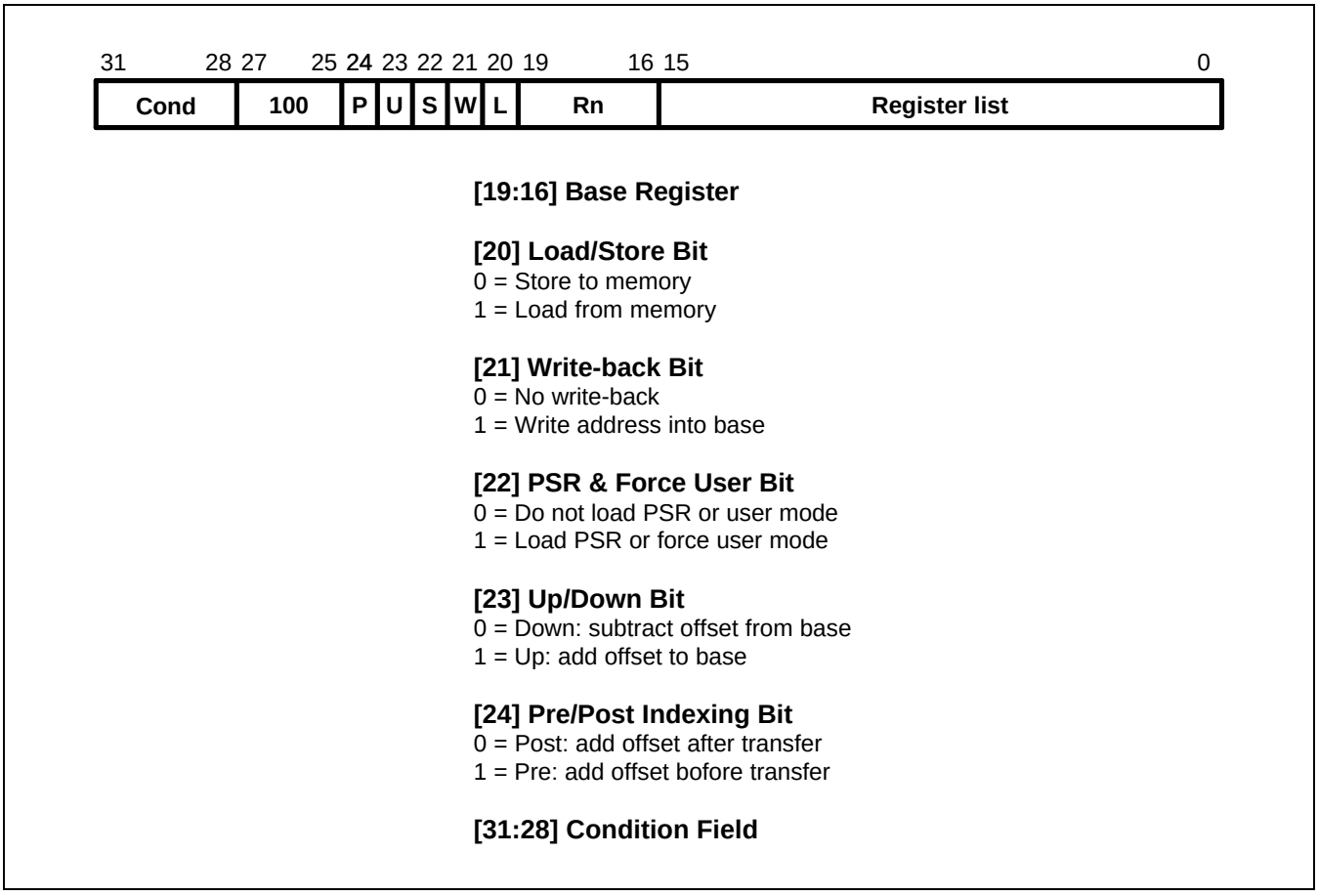


Figure 3-18. Block Data Transfer Instructions

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.

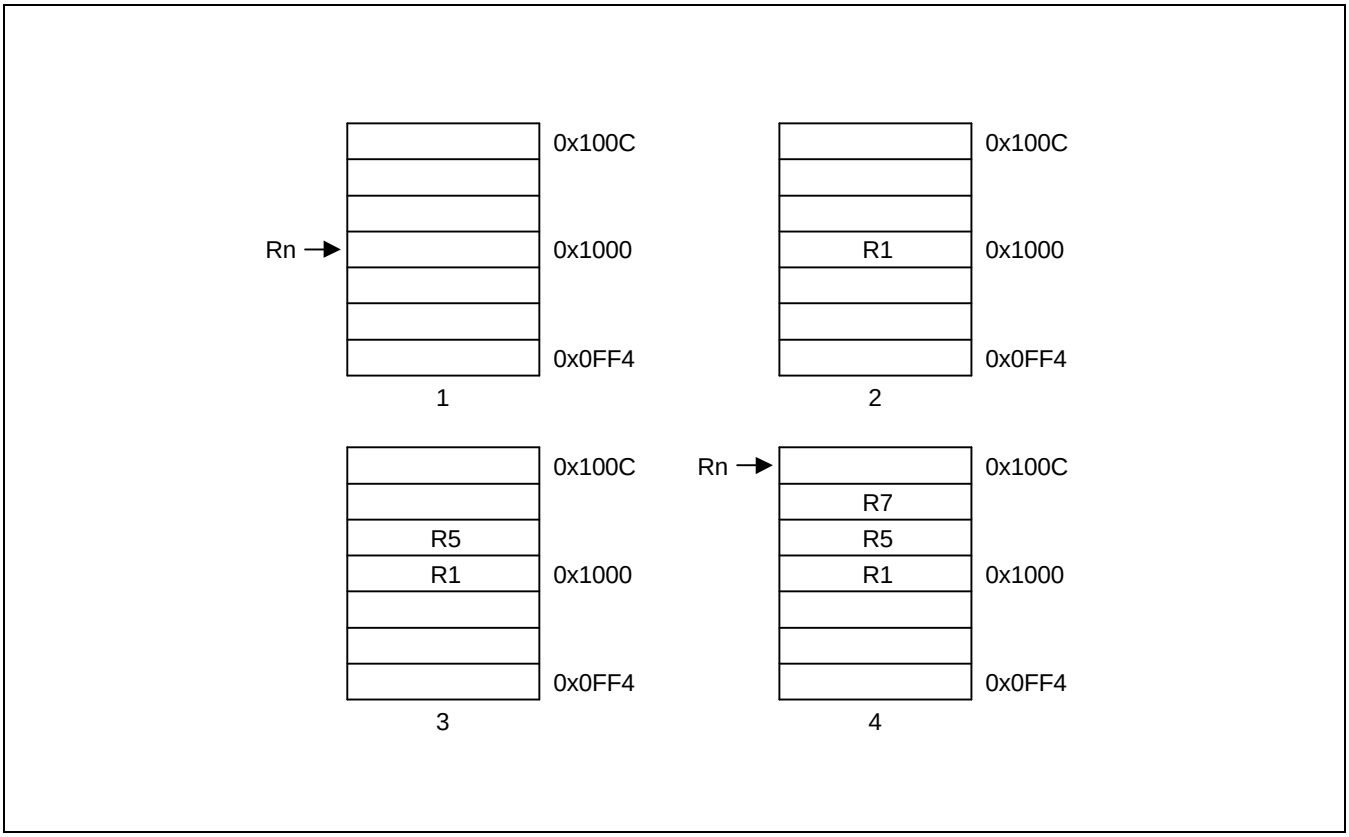


Figure 3-19. Post-Increment Addressing

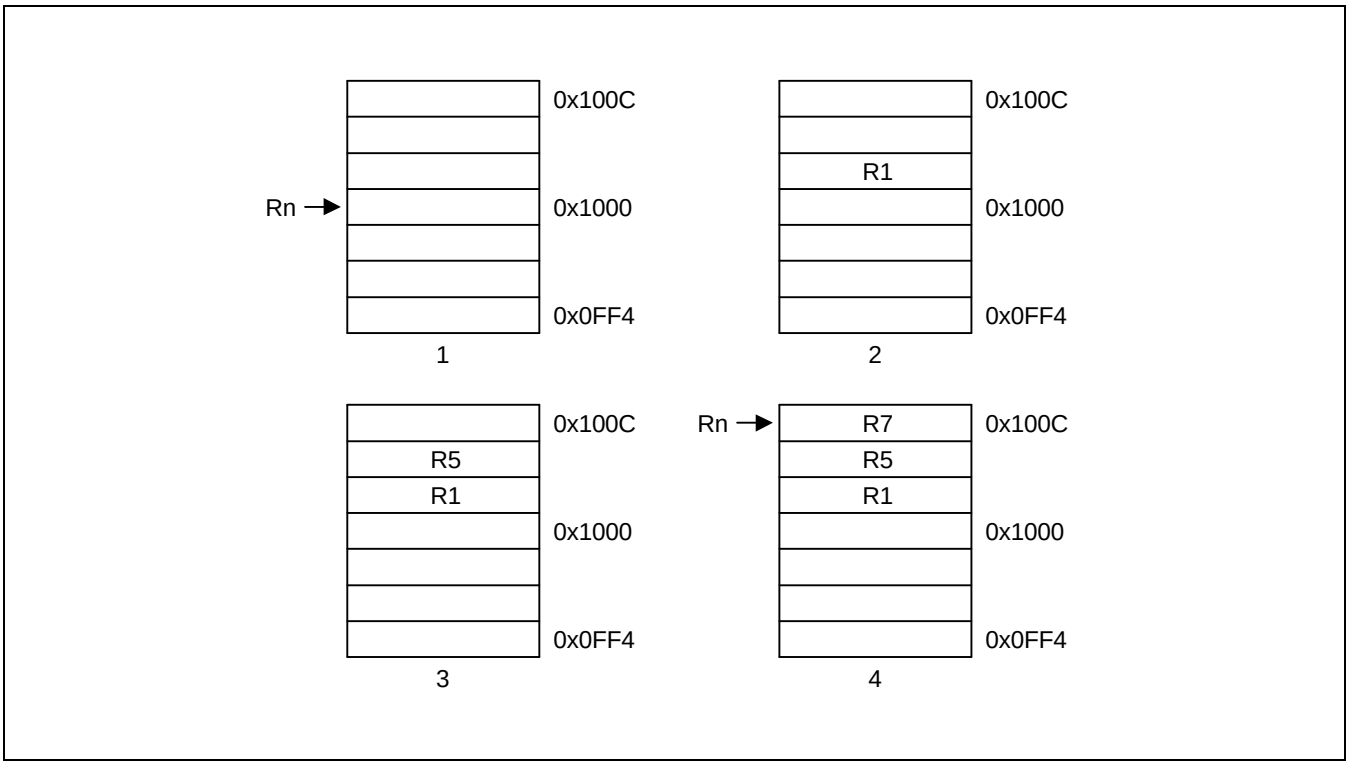


Figure 3-20. Pre-Increment Addressing

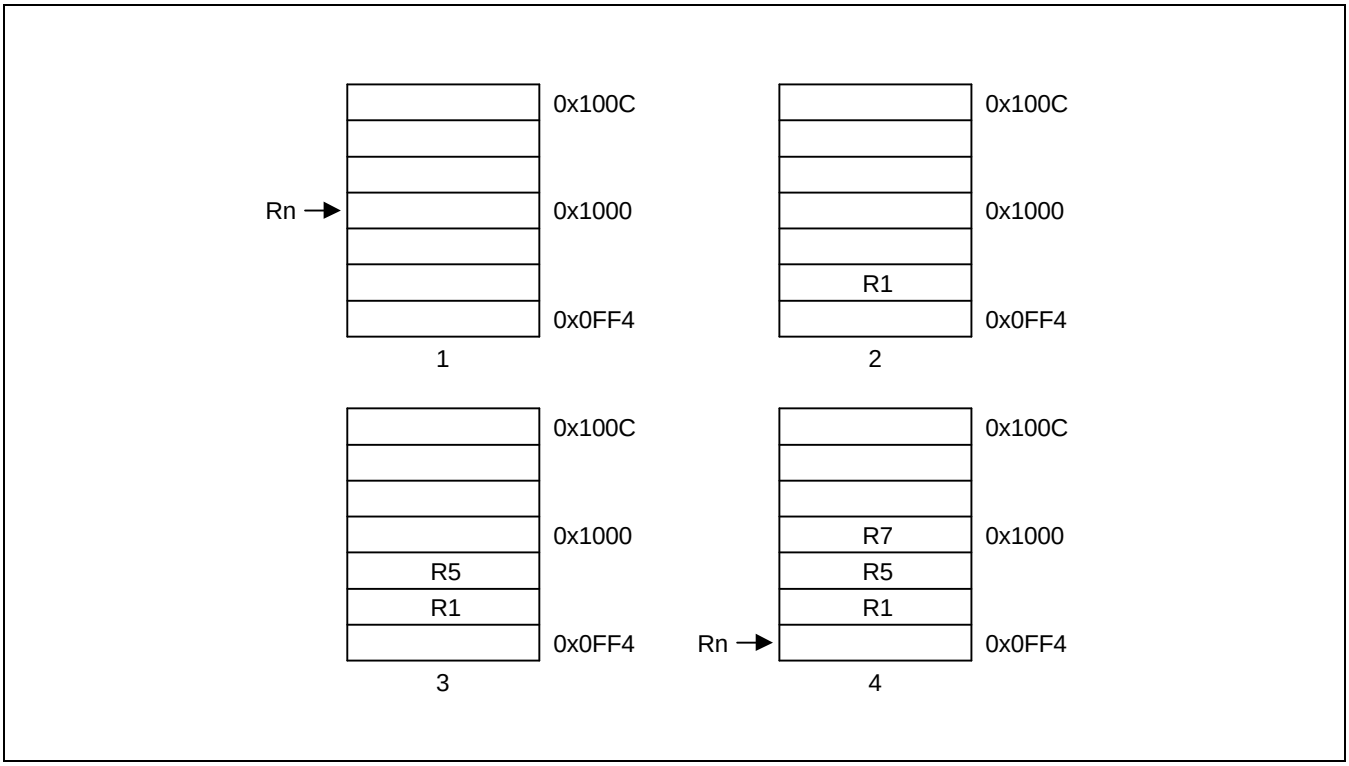


Figure 3-21. Post-Decrement Addressing

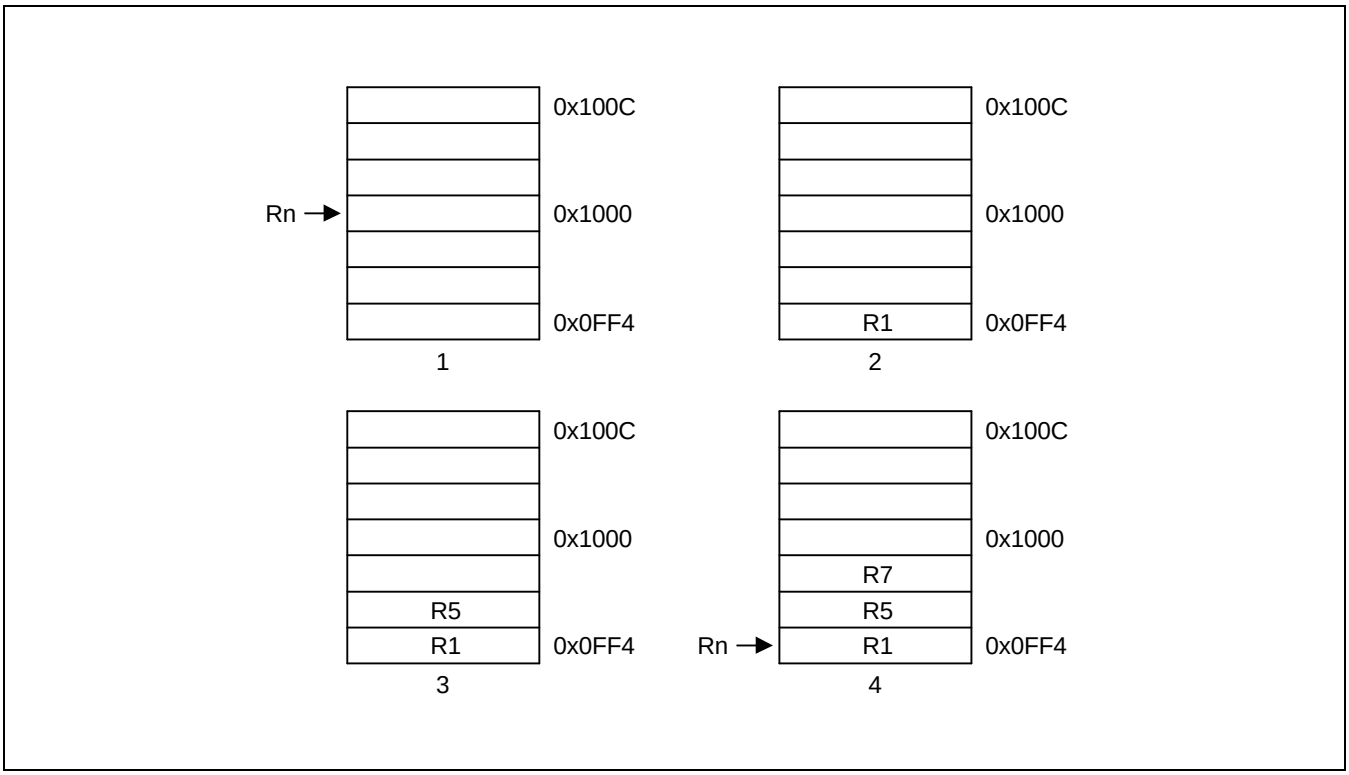


Figure 3-22. Pre-Decrement Addressing

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 ARM7TDMI is to be used in a virtual memory system.

Abort during STM Instructions

If the abort occurs during a store multiple instruction, ARM7TDMI 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 ARM7TDMI 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)

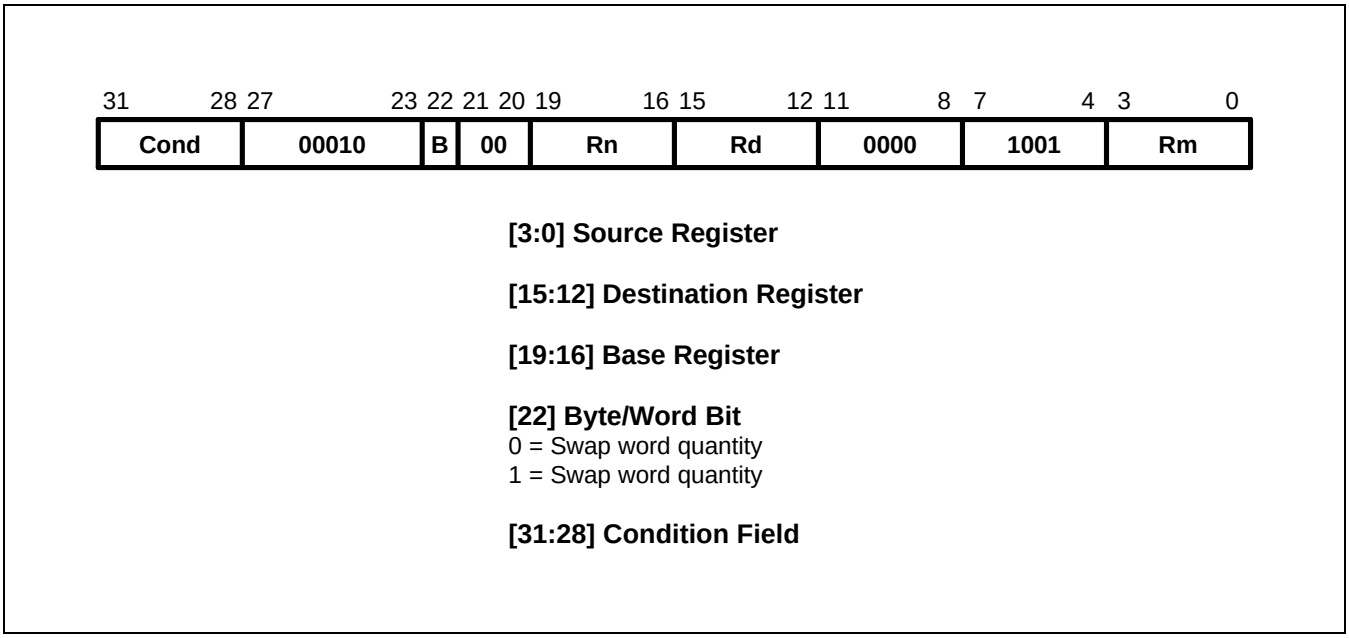


Figure 3-23. Swap Instruction

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 ARM7TDMI 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.

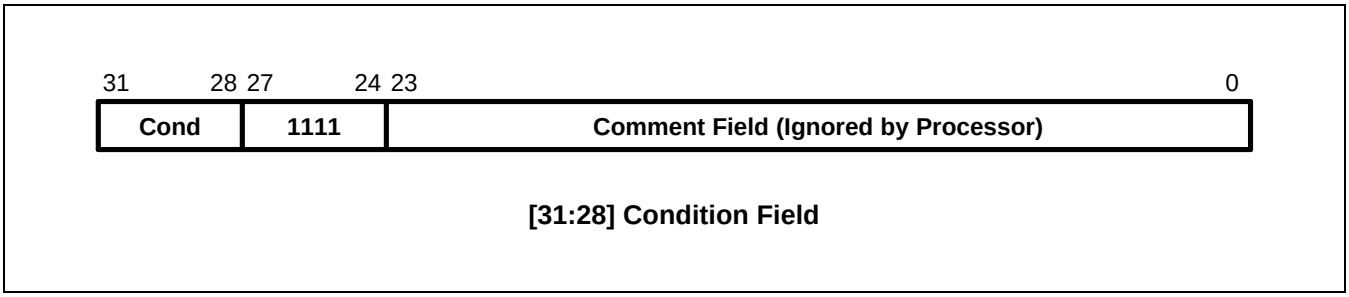


Figure 3-24. Software Interrupt Instruction

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 ARM7TDMI).

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 ARM7TDMI, 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 ARM7TDMI 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.)

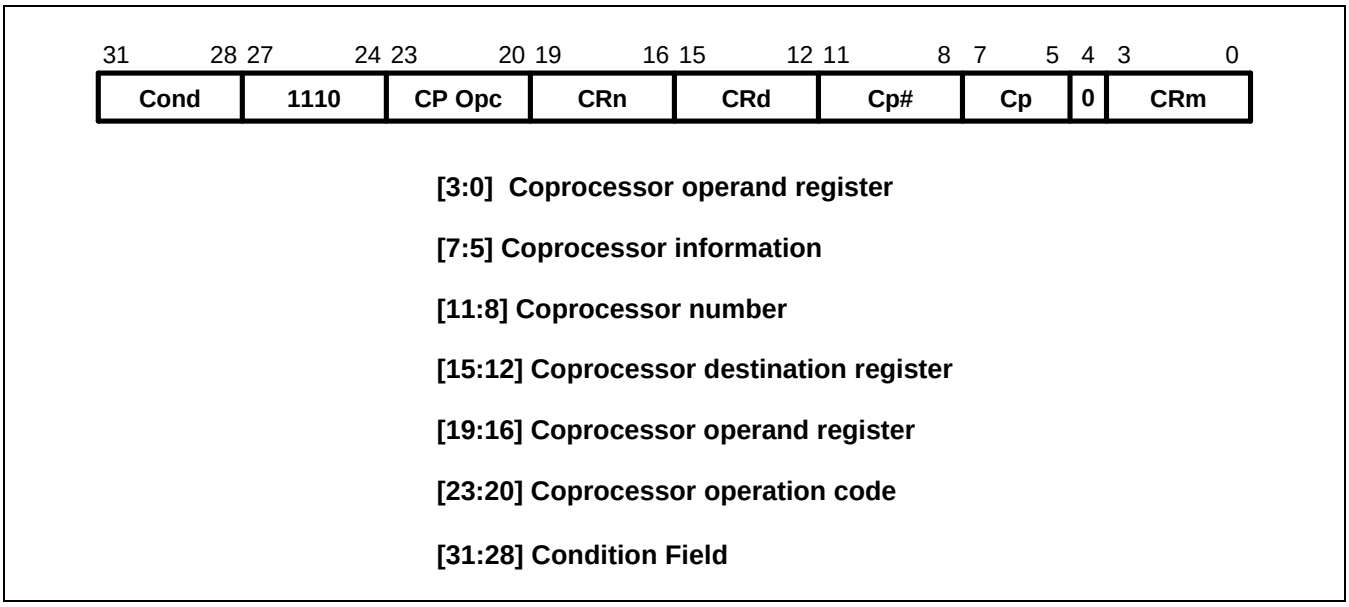


Figure 3-25. Coprocessor Data Operation Instruction

Only bit 4 and bits 24 to 31 The coprocessor fields are significant to ARM7TDMI. 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

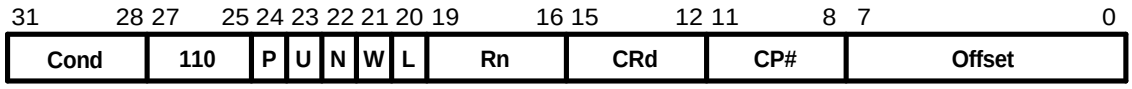
EXAMPLES

```
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. ARM7TDMI 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

ARM7TDMI 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 ARM7TDMI register number.

NOTE

If Rn is R15, the assembler will subtract 8 from the offset value to allow for ARM7TDMI pipelining.

EXAMPLES

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

NOTE

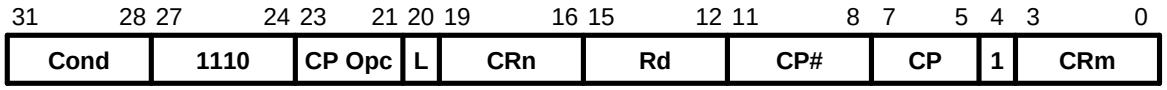
Although the address offset is expressed in bytes, the instruction offset field is in words. The assembler will adjust the offset appropriately.

COPROCESSOR REGISTER TRANSFERS (MRC, MCR)

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

This class of instruction is used to communicate information directly between ARM7TDMI and a coprocessor. An example of a coprocessor to ARM7TDMI 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 ARM7TDMI register. A FLOAT of a 32 bit value in ARM7TDMI register into a floating point value within the coprocessor illustrates the use of ARM7TDMI register to coprocessor transfer (MCR).

An important use of this instruction is to communicate control information directly from the coprocessor into the ARM7TDMI CPSR flags. As an example, the result of a comparison of two floating point values within a coprocessor can be moved to the CPSR to control the subsequent flow of execution.



[3:0] Coprocessor Operand Register

[7:5] Coprocessor Information

[11:8] Coprocessor Number

[15:12] ARM Source/Destination Register

[19:16] Coprocessor Source/Destination Register

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

[21] Coprocessor Operation Mode

[31:28] Condition Field

Figure 3-27. Coprocessor Register Transfer Instructions

THE COPROCESSOR FIELDS

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

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

TRANSFERS TO R15

When a coprocessor register transfer to ARM7TDMI 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 ARM7TDMI 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 ARM7TDMI register (L=1)
MCR	Move from ARM7TDMI 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 ARM7TDMI 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.

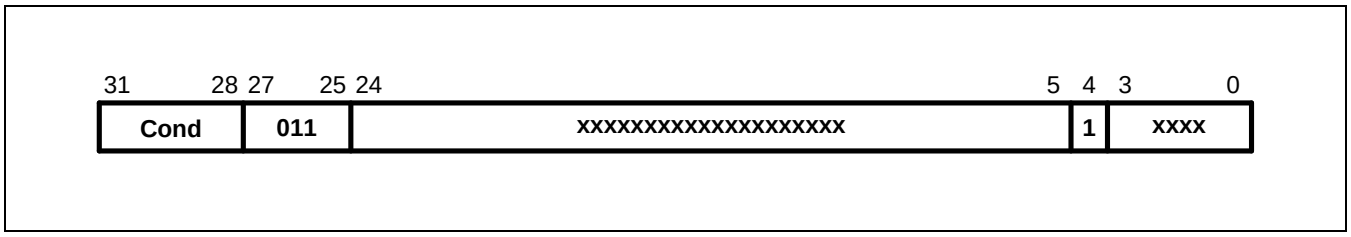


Figure 3-28. Undefined Instruction

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 ARM7TDMI 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 ARM7TDMI

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

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

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

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

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

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

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

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

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

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

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

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

NOTE

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

PSEUDO-RANDOM BINARY SEQUENCE GENERATOR

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

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

MULTIPLICATION BY CONSTANT USING THE BARREL SHIFTER

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

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

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

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

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

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

Multiplication by 6

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

Multiply by 10 and add in extra number

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

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

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

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

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

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

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

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

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

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

rather than by:

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

LOADING A WORD FROM AN UNKNOWN ALIGNMENT

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

NOTES

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 ARM7TDMI 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.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1	0	0	0	Op		Offset5					Rs			Rd			Move Shifted register
2	0	0	0	1	1	I	Op	Rn/offset3			Rs			Rd			Add/subtract
3	0	0	1	Op		Rd			Offset8								Move/compare/add/ subtract immediate
4	0	1	0	0	0	0	Op				Rs			Rd			ALU operations
5	0	1	0	0	0	1	Op		H1	H2	Rs/Hs			Rd/Hd			Hi register operations /branch exchange
6	0	1	0	0	1	Rd			Word8								PC-relative load
7	0	1	0	1	L	B	0	Ro			Rb			Rd			Load/store with register offset
8	0	1	0	1	H	S	1	Ro			Rb			Rd			Load/store sign-extended byte/halfword
9	0	1	1	B	L	Offset5					Rb			Rd			Load/store with immediate offset
10	1	0	0	0	L	Offset5					Rb			Rd			Load/store halfword
11	1	0	0	1	L	Rd			Word8								SP-relative load/store
12	1	0	1	0	SP	Rd			Word8								Load address
13	1	0	1	1	0	0	0	0	S	SWord7							Add offset to stack pointer
14	1	0	1	1	L	1	0	R	Rlist								Push/pop register
15	1	1	0	0	L	Rb			Rlist								Multiple load/store
16	1	1	0	1	Cond				Softset8								Conditional branch
17	1	1	0	1	1	1	1	1	Value8								Software interrupt
18	1	1	1	0	0	Offset11											Unconditional branch
19	1	1	1	1	H	Offset											Long branch with link
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Figure 3-29. 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 3-7. 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 3-7. 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

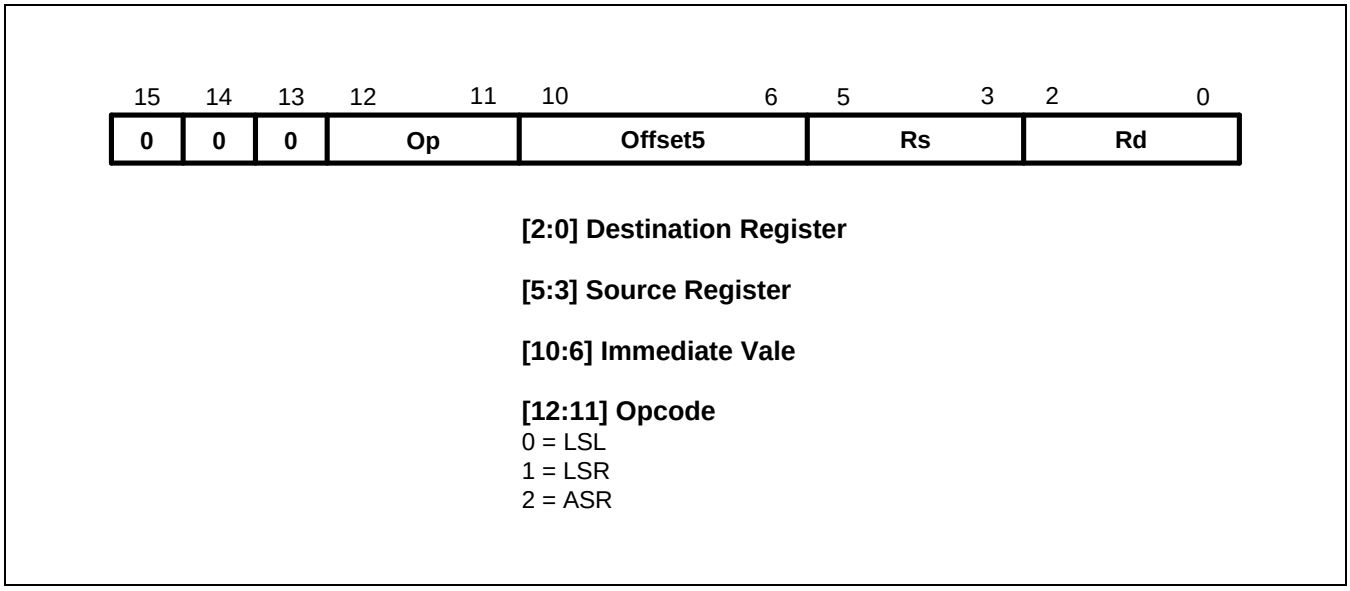


Figure 3-30. Format 1

OPERATION

These instructions move a shifted value between Lo registers. The THUMB assembler syntax is shown in Table 3-8.

NOTE

All instructions in this group set the CPSR condition codes.

Table 3-8. 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.

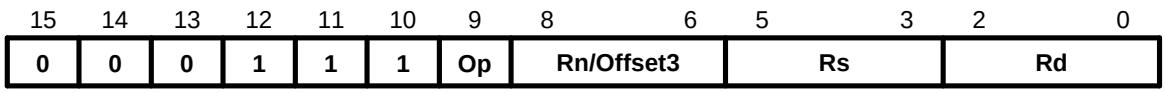
INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 3-8. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

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

FORMAT 2: ADD/SUBTRACT



[2:0] Destination Register

[5:3] Source Register

[8:6] Register/Immediate Vale

[9] Opcode
0 = ADD
1 = SUB

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

Figure 3-31. 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 3-9.

NOTE

All instructions in this group set the CPSR condition codes.

Table 3-9. 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 3-9. 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

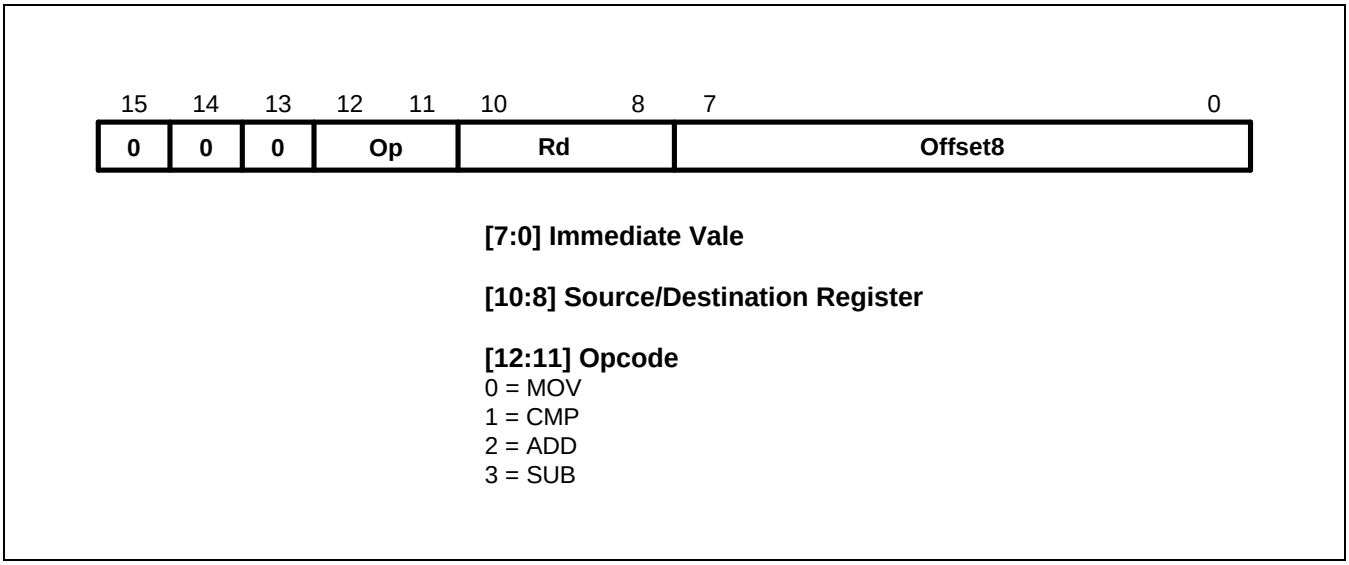


Figure 3-32. Format 3

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 3-10.

NOTE

All instructions in this group set the CPSR condition codes.

Table 3-10. 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 3-10. 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

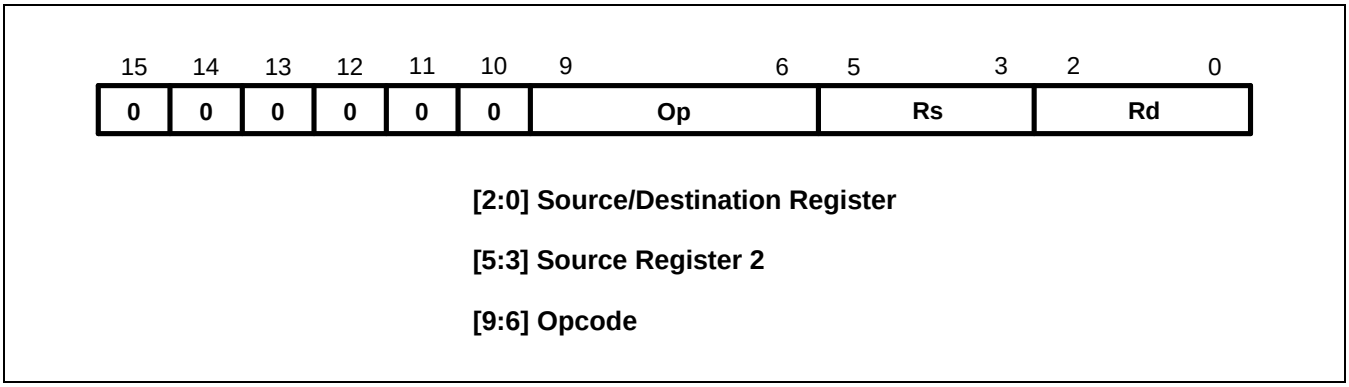


Figure 3-33. Format 4

OPERATION

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

NOTE

All instructions in this group set the CPSR condition codes.

Table 3-11. 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 3-11. 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

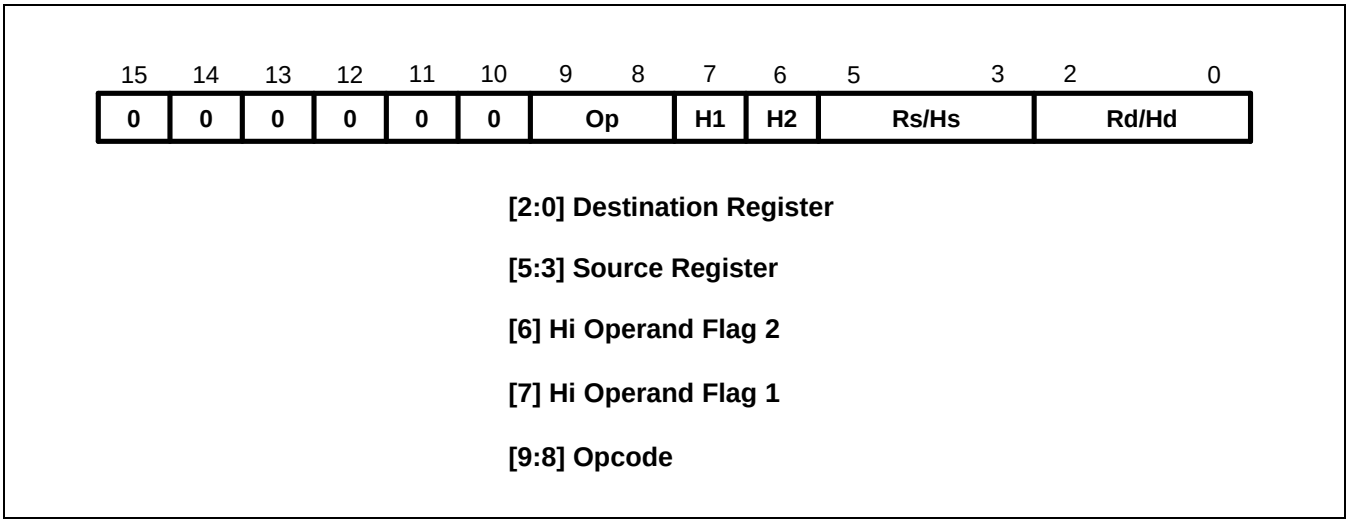


Figure 3-34. Format 5

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 3-12.

NOTE

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

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

Table 3-12. 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 3-12. 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 3-12. 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

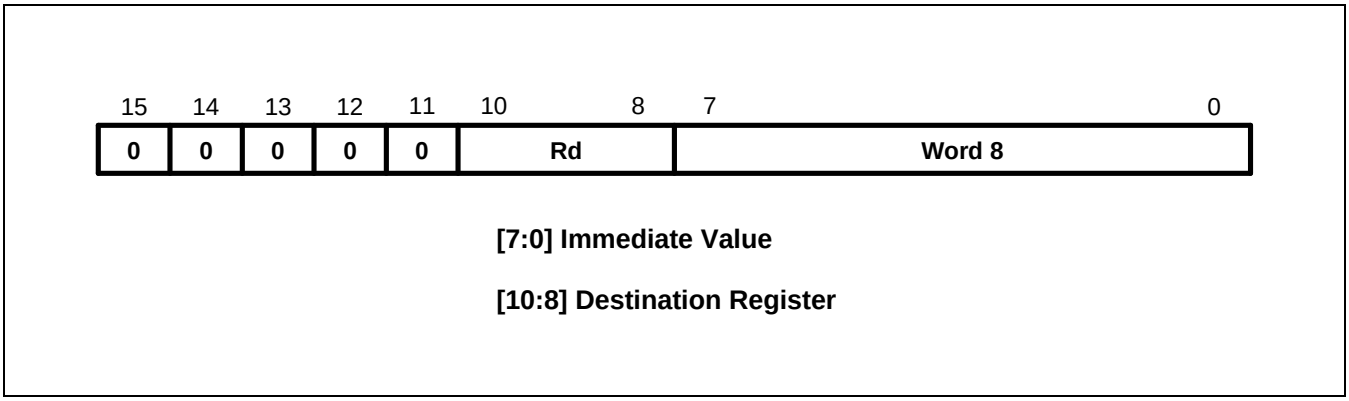


Figure 3-35. Format 6

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 3-13. 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.

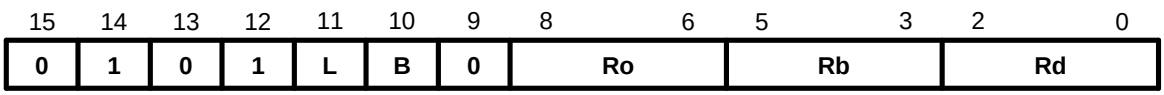
INSTRUCTION CYCLE TIMES

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 3-36. 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 3-14.

Table 3-14. 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 3-14. 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

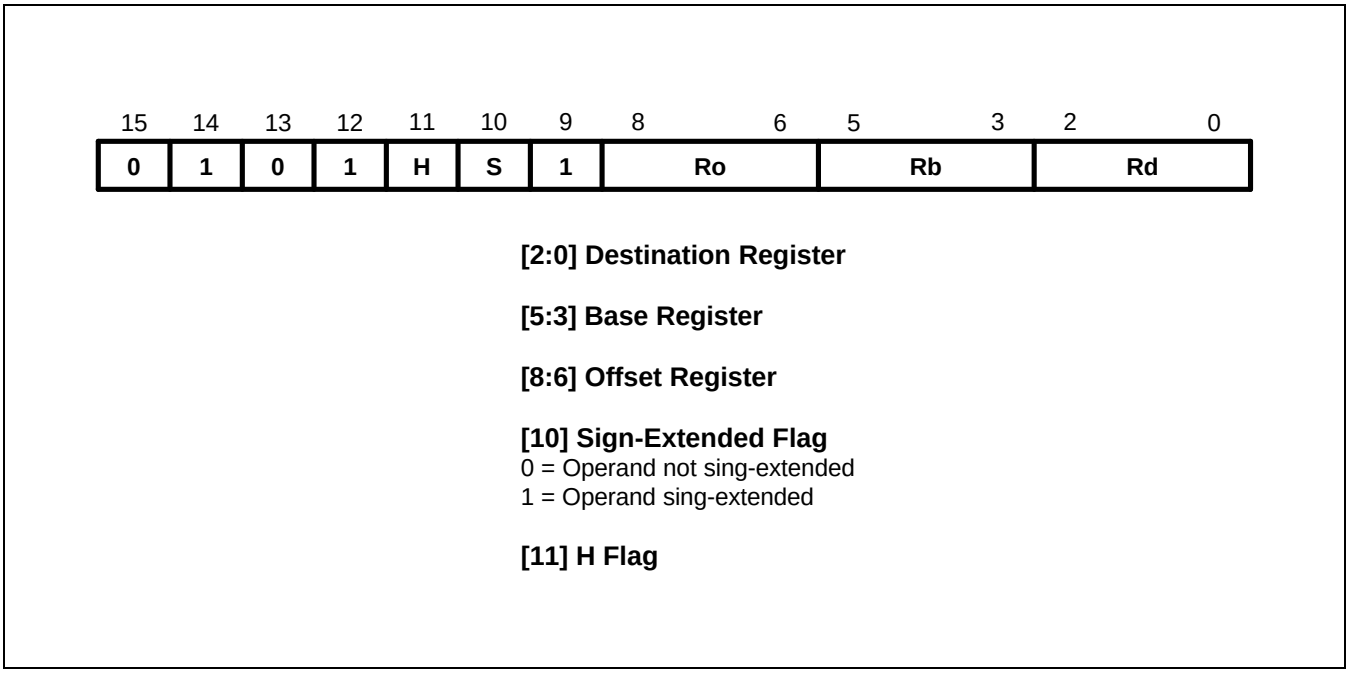


Figure 3-37. Format 8

OPERATION

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

Table 3-15. 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.

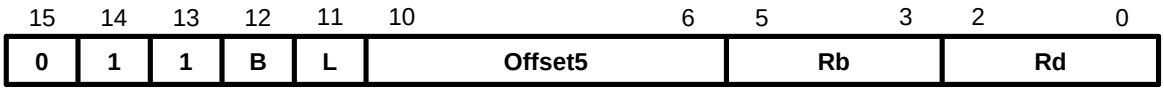
INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 3-15. 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 3-38. 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 3-16.

Table 3-16. 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.

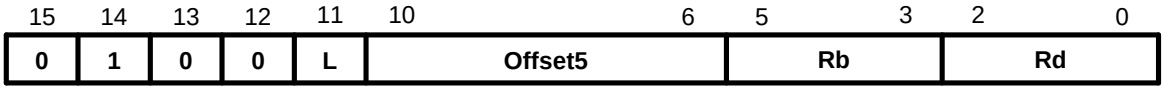
INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 3-16. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

EXAMPLES

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

FORMAT 10: LOAD/STORE HALFWORD



[2:0] Source/Destination Register

[5:3] Base Register

[10:6] Immediate Value

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

Figure 3-39. 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 3-17.

Table 3-17. 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 3-17. 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

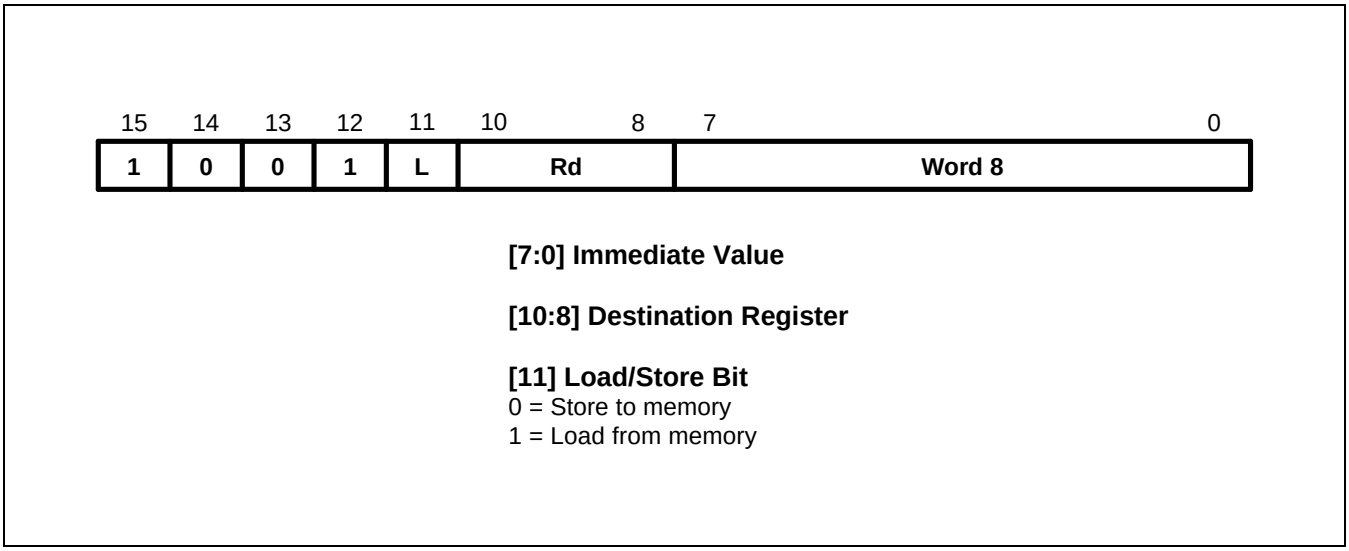


Figure 3-40. Format 11

OPERATION

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

Table 3-18. 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 3-18. 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

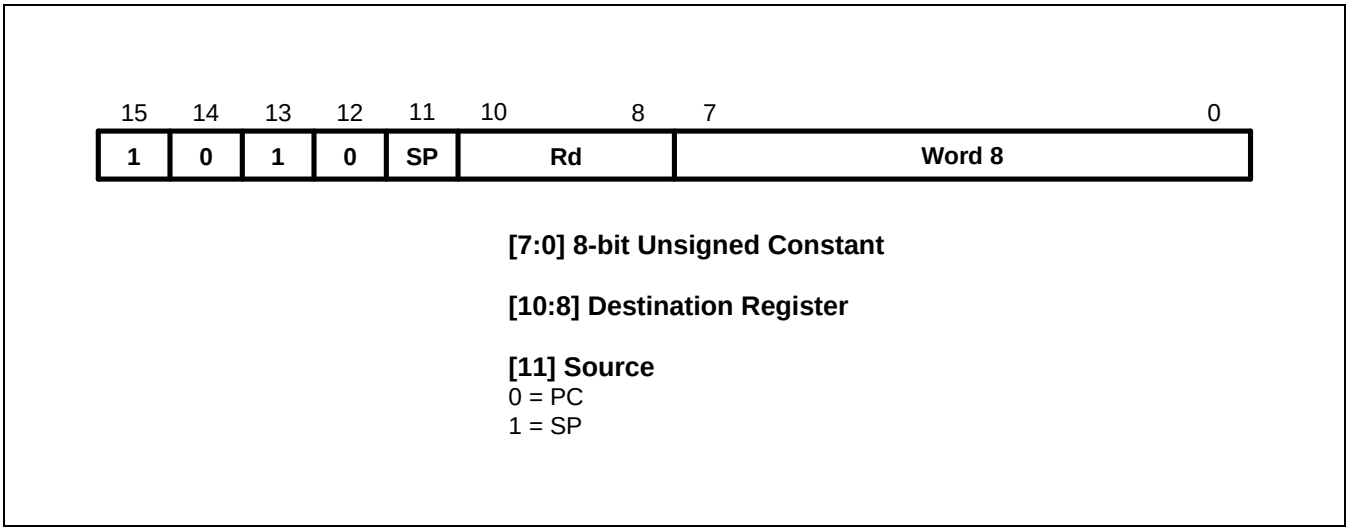


Figure 3-41. Format 12

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 3-19. 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 3-19. 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

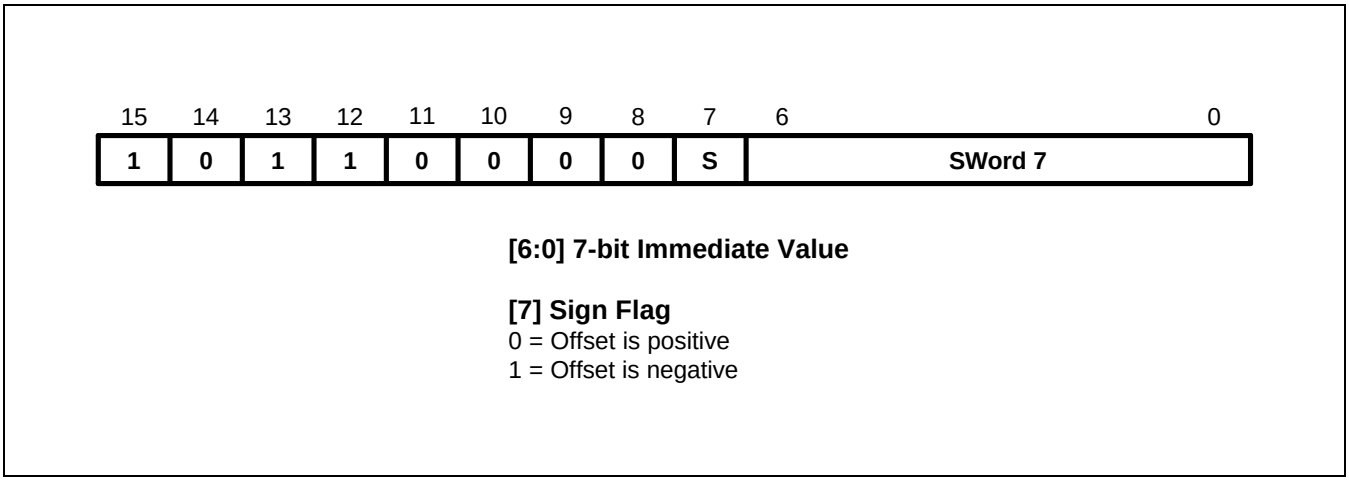


Figure 3-42. Format 13

OPERATION

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

Table 3-20. 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.

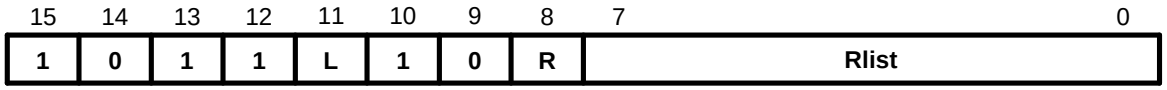
INSTRUCTION CYCLE TIMES

All instructions in this format have an equivalent ARM instruction as shown in Table 3-20. 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 3-43. 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 3-21.

NOTE

The stack is always assumed to be Full Descending.

Table 3-21. 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 3-21. 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

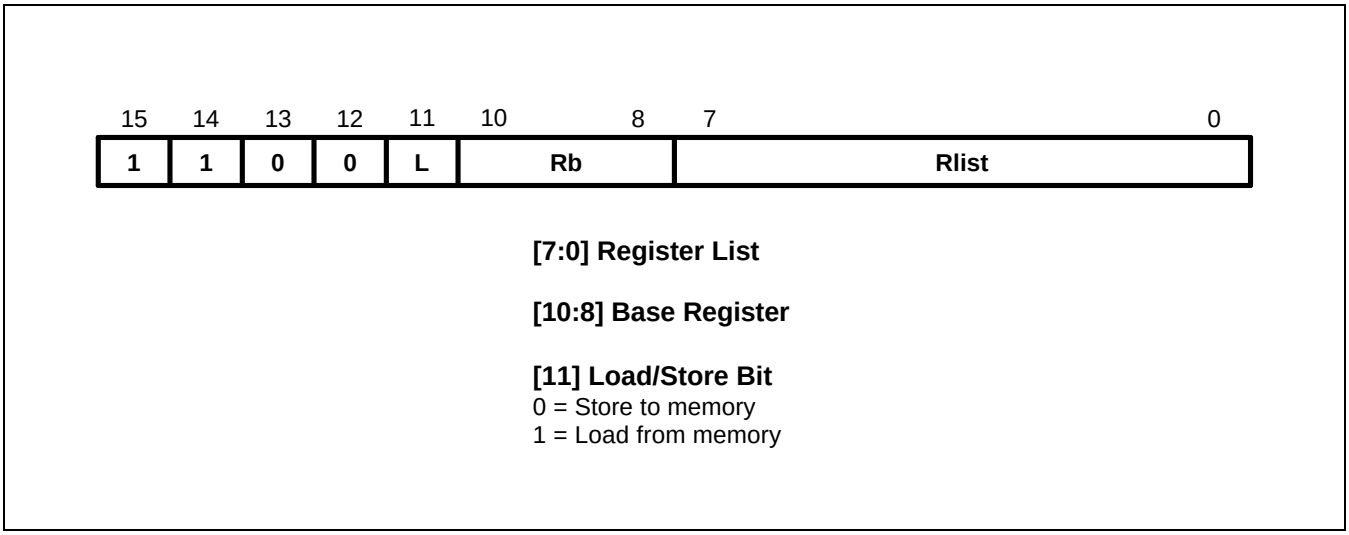


Figure 3-44. Format 15

OPERATION

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

Table 3-22. 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 3-22. 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

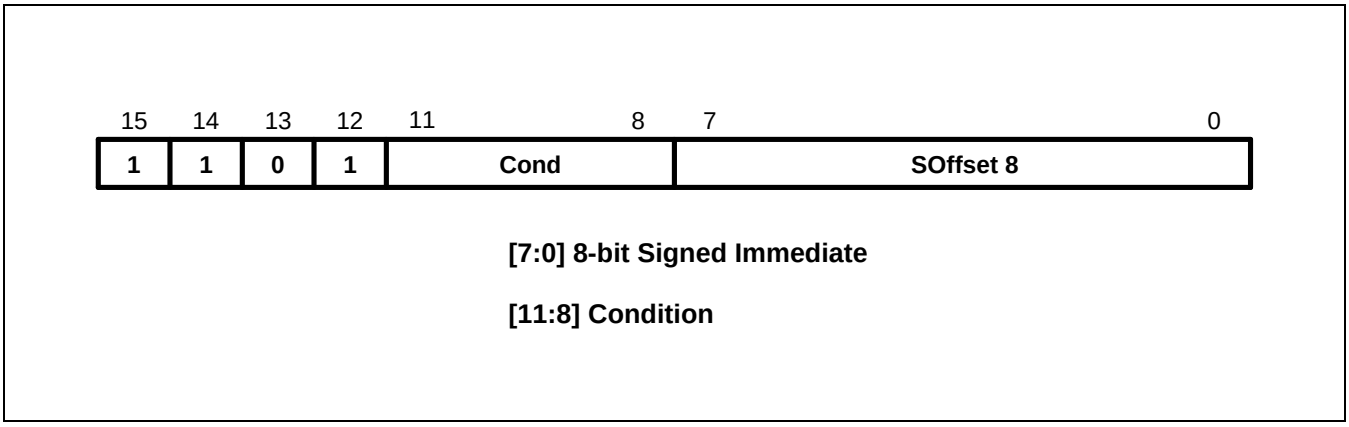


Figure 3-45. Format 16

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 2-23. 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 2-23. 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 3-23. 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

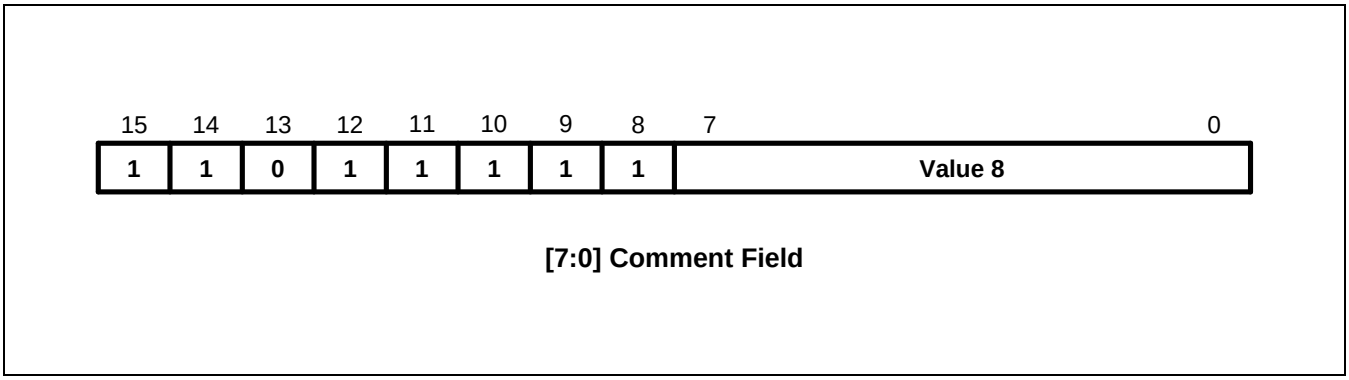


Figure 3-46. Format 17

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 3-24. 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 3-24. 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

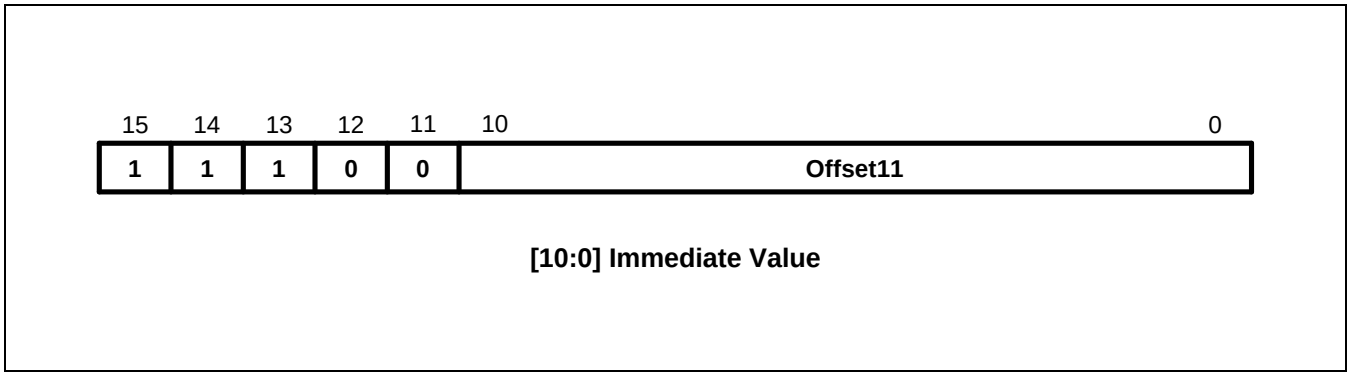


Figure 3-47. Format 18

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 3-25. 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

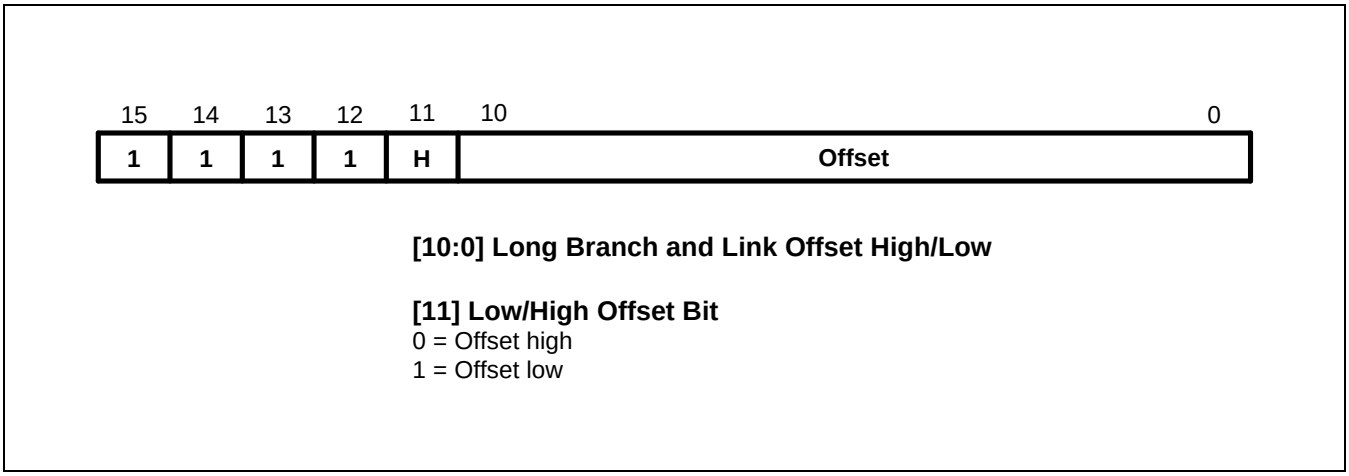


Figure 3-48. Format 19

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 3-26. 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
```

4 MEMORY CONTROLLER

OVERVIEW

The S3C44B0X memory controller provides the necessary memory control signals for external memory access. S3C44B0X has the following features;

— Little/Big endian(selectable by an external pin)

— 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, FP/EDO/SDRAM etc .

— 7 fixed memory bank start address and programmble bank size

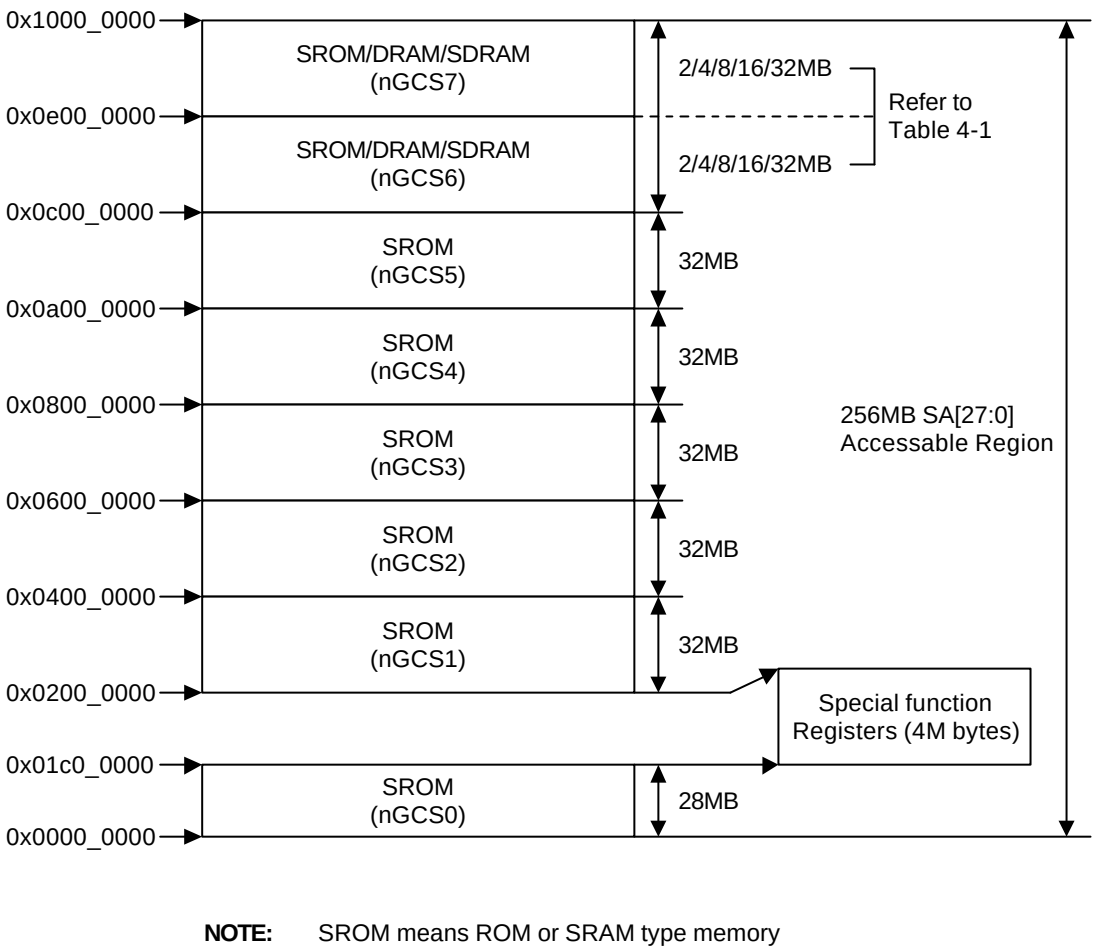
— 1 flexible memory bank start address and programmable bank size

— 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



NOTE: SROM means ROM or SRAM type memory

Figure 4-1. S3C44B0X Memory Map after Reset

Table 4-1. 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

LITTLE ENDIAN/BIG ENDIAN

While nRESET is L, the ENDIAN pin defines which endian mode should be selected. If the ENDIAN pin is connected to Vss with a pull-down resistor, the little endian mode is selected. If the pin is connected to Vdd with a pull-up resistor, the big endian mode is selected.

ENDIAN Input @Reset	ENDIAN Mode
0	Little endian
1	Big endian

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;

```
        ldr     r0, =SMRDATA
        ldmia   r0, {r1-r13}
        ldr     r0, =0x01c80000         ;  BWSCON Address
        stmia   r0, {r1-r13}
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 ADDR. PIN	S3C44B0X ADDR. @ 8-bit DATA BUS	S3C44B0X ADDR. @ 16-bit DATA BUS	S3C44B0X ADDR. @ 32-bit DATA BUS
A0	A0	A1	A2
A1	A1	A2	A3
A2	A2	A3	A4
A3	A3	A4	A5
. . .	. . .	. . .	. . .

SDRAM BANK ADDRESS PIN CONNECTION

Table 4-2. SDRAM Bank Address configuration

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

ROM Memory Interface Example

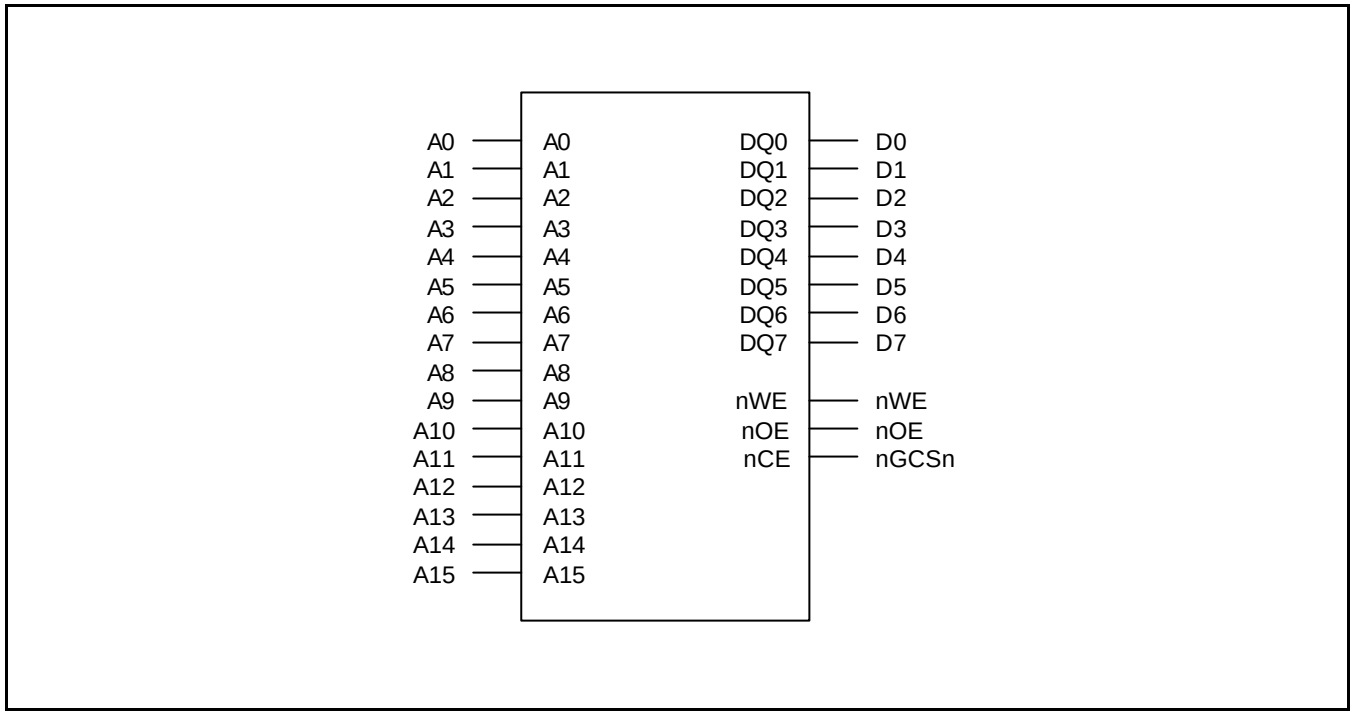


Figure 4-2. Memory Interface with 8bit ROM

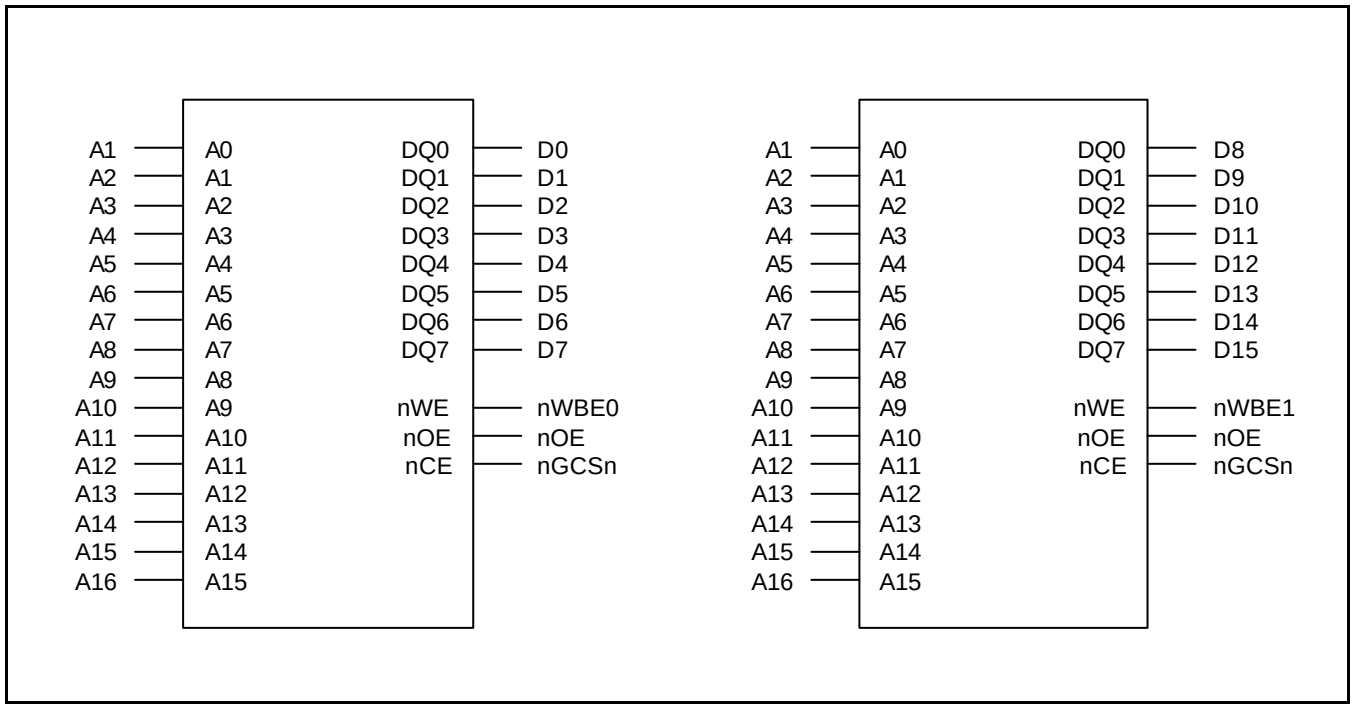


Figure 4-3. Memory Interface with 8bit ROM x 2

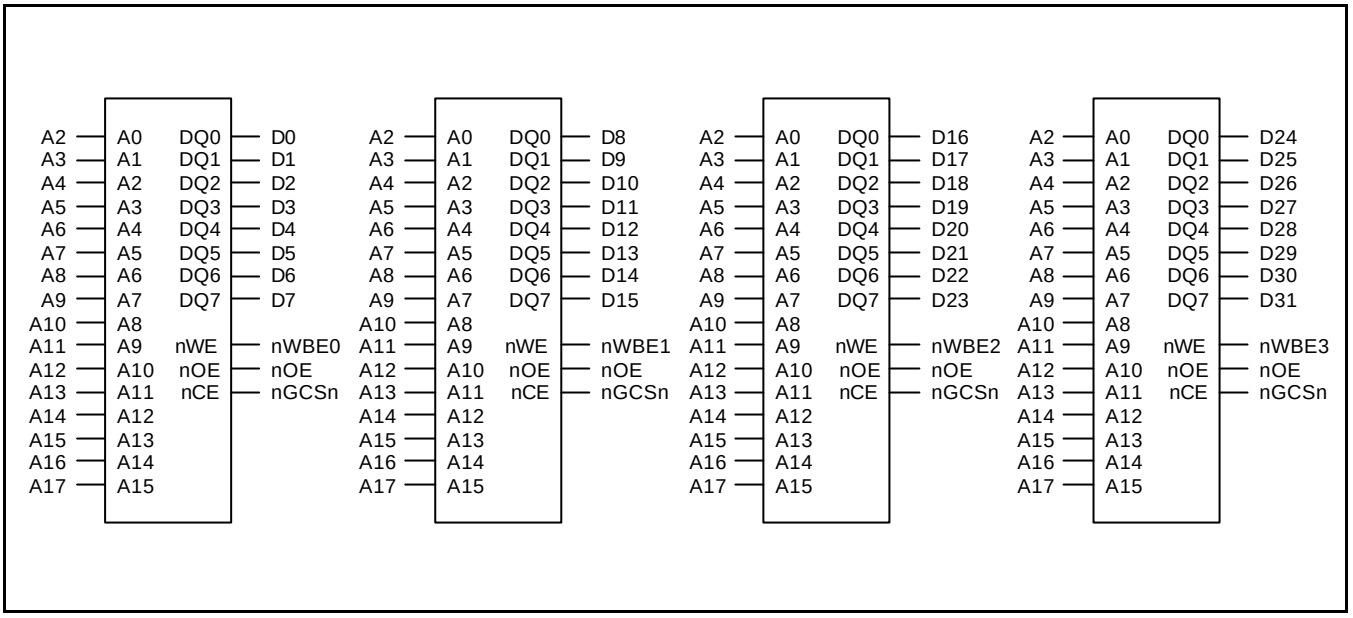


Figure 4-4. Memory Interface with 8bit ROM x 4

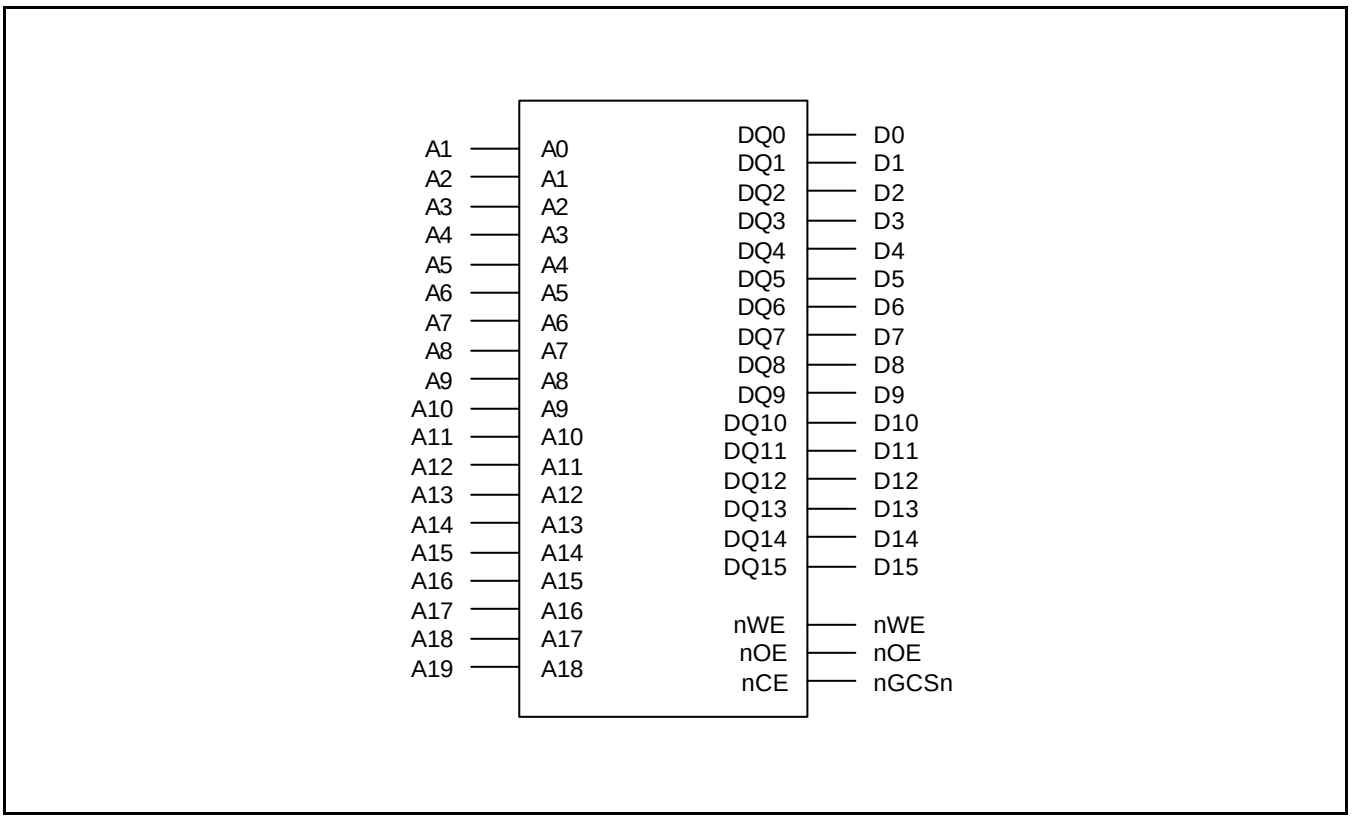


Figure 4-5. Memory Interface with 16bit ROM

SRAM Memory Interface Example

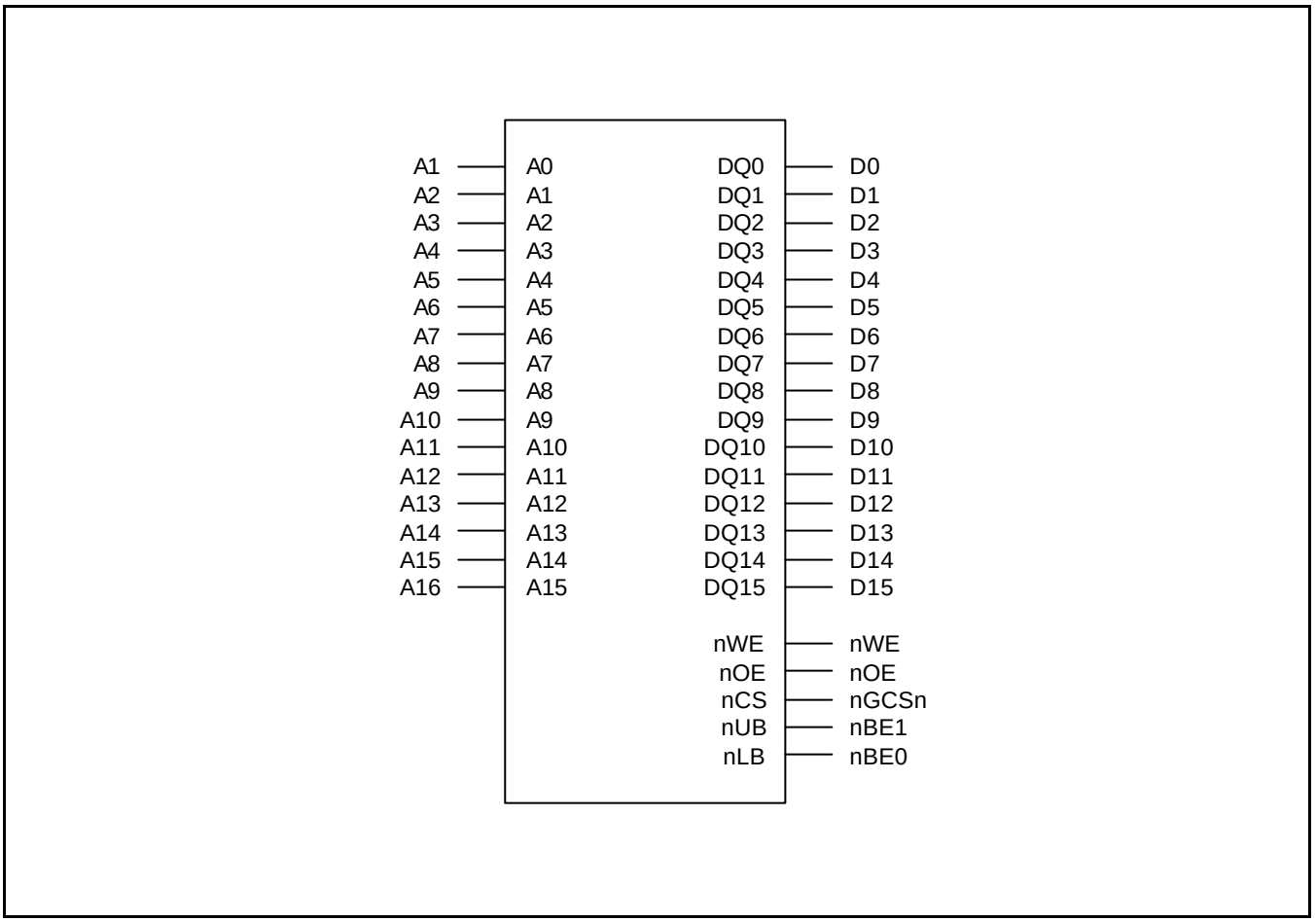


Figure 4-6. Memory Interface with 16bit SRAM

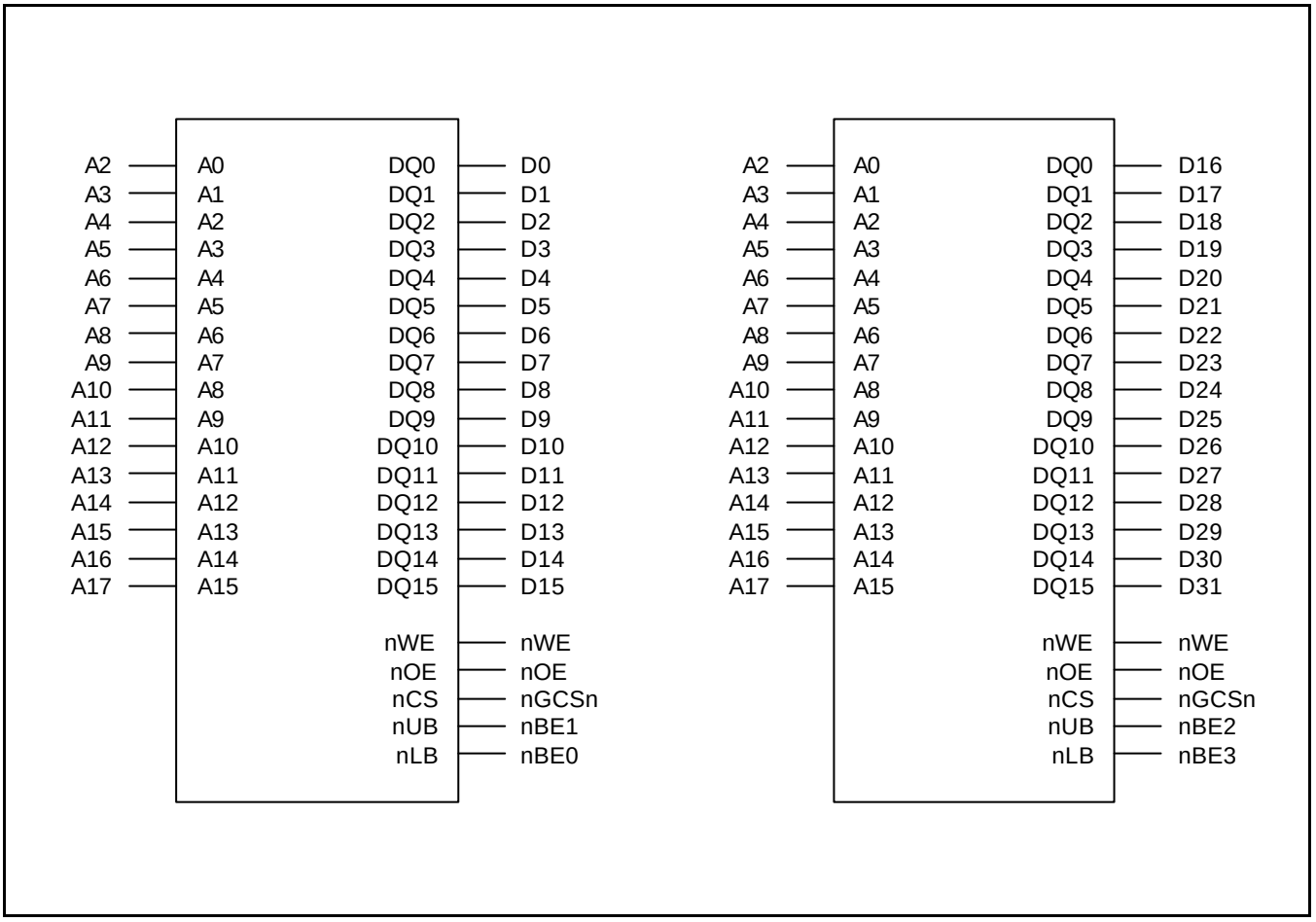


Figure 4-7. Memory Interface with 16bit SRAM x 2

DRAM Memory Interface Example

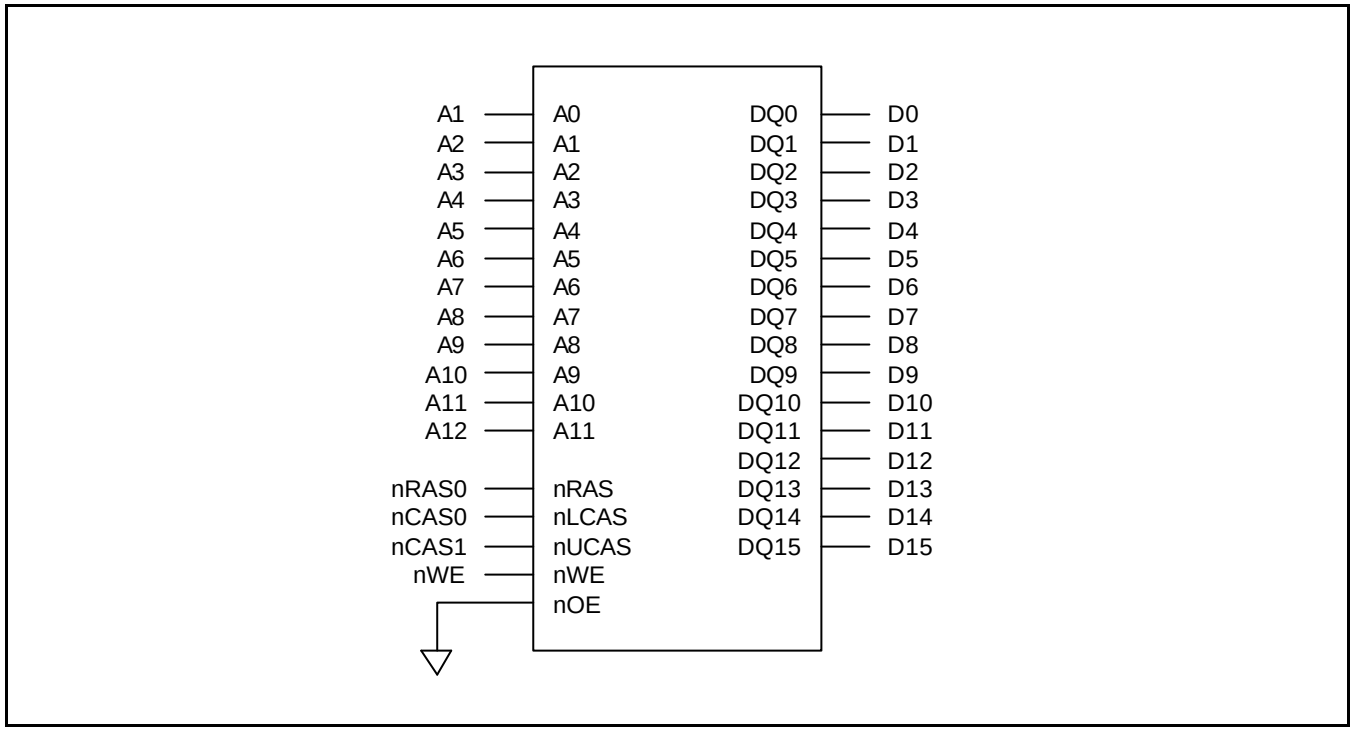


Figure 4-8. Memory Interface with 16bit DRAM

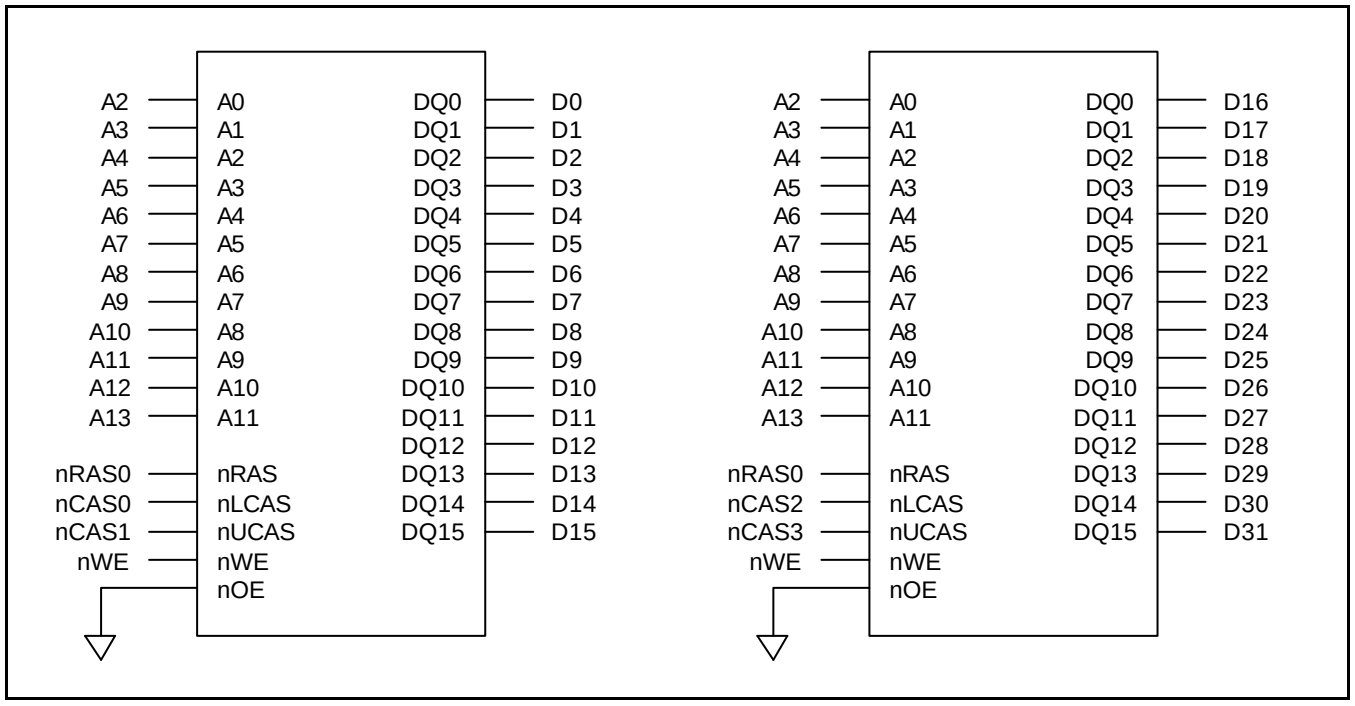


Figure 4-9. Memory Interface with 16bit DRAM x 2

SDRAM Memory Interface Example

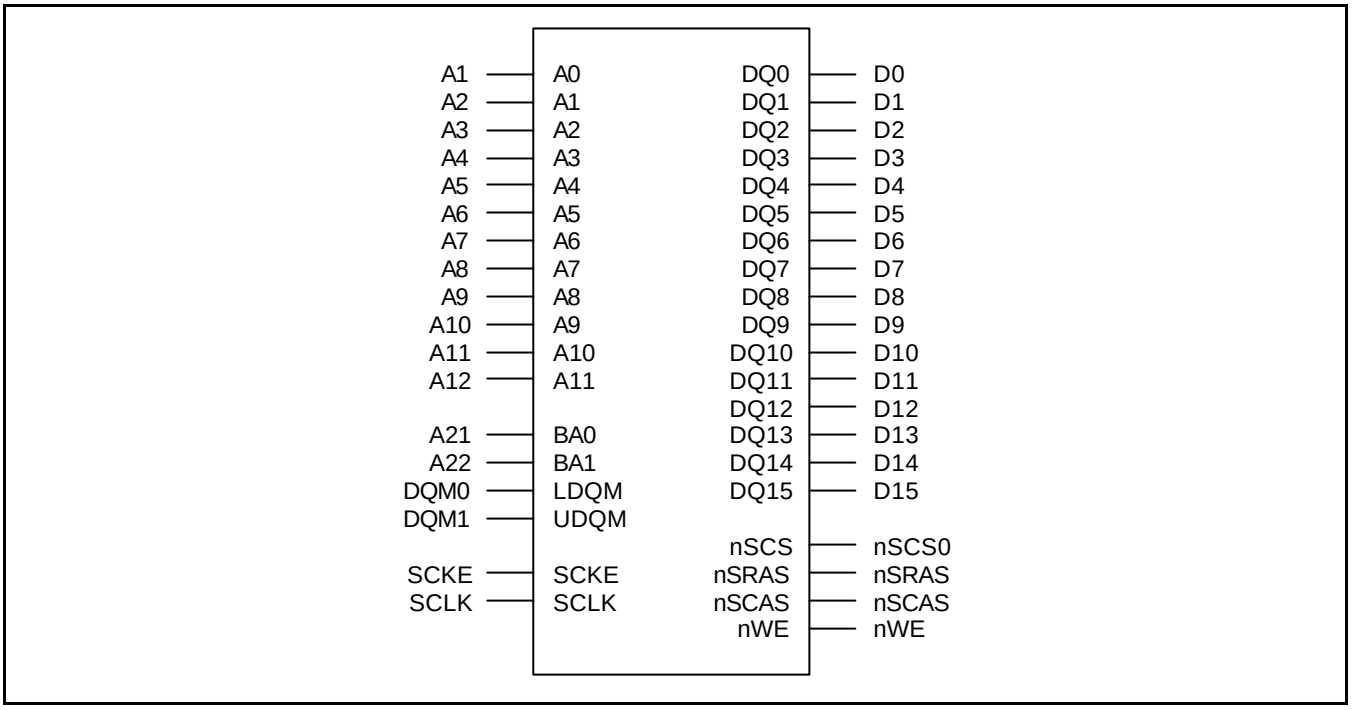


Figure 4-10. Memory Interface with 16bit SDRAM (4Mx16, 4bank)

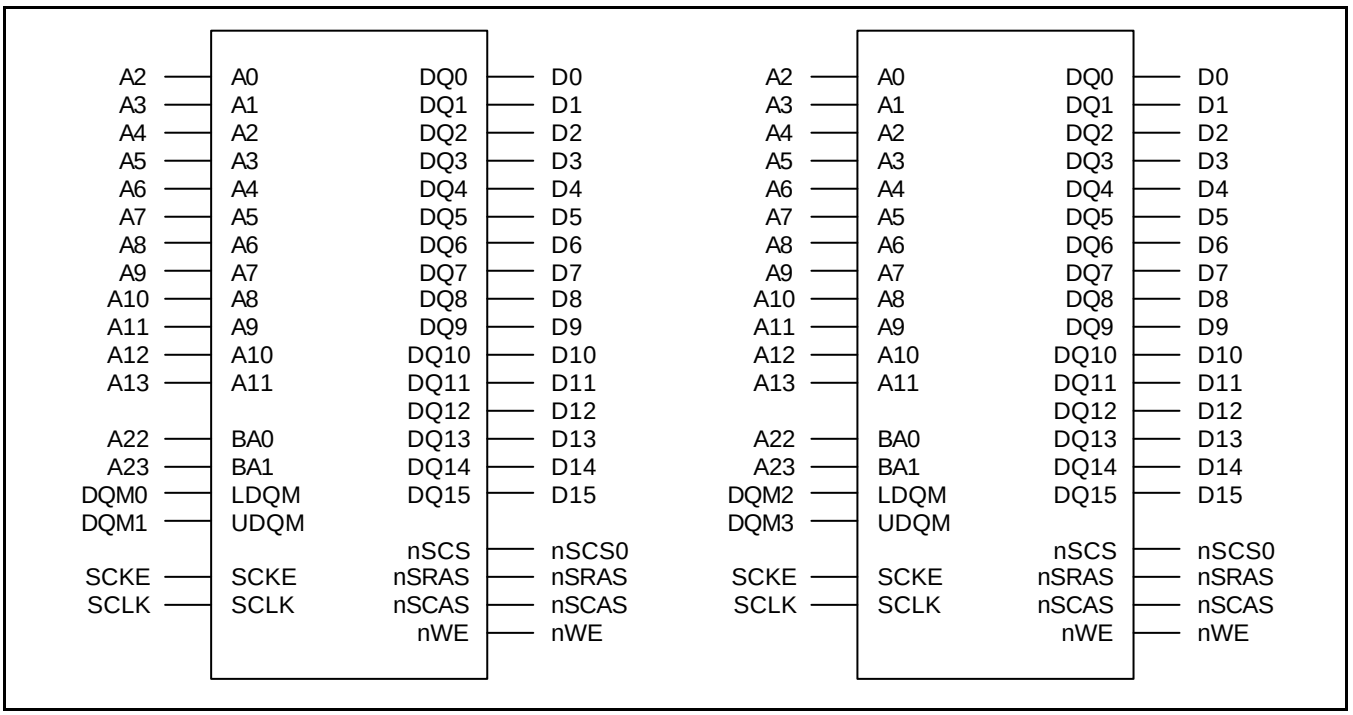


Figure 4-11. Memory Interface with 16bit SDRAM (4Mx16 * 2ea, 4bank)

NOTE: Please refer to Table 4-2 the Bank Address configurations of SDRAM.

MEMORY CONTROLLER SPECIAL REGISGERS

BUS WIDTH & WAIT CONTROL REGISTER (BWSCON)

Register	Address	R/W	Description	Reset Value
BWSCON	0x01C80000	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[14:11] is dedicated nWBE[3:0]) 1 = Using UB/LB (Pin[14:11] is dedicated nBE[3:0])	0
WS7	[30]	This bit determines WAIT status for bank 7 (If bank7 has DRAM or SDRAM, WAIT function is not supported) 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[14:11] is dedicated nWBE[3:0]) 1 = Using UB/LB (Pin[14:11] is dedicated nBE[3:0])	0
WS6	[26]	This bit determines WAIT status for bank 6 (If bank6 has DRAM or SDRAM, WAIT function is not supported) 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[14:11] is dedicated nWBE[3:0]) 1 = Using UB/LB (Pin[14:11] 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[14:11] is dedicated nWBE[3:0]) 1 = Using UB/LB (Pin[14:11] 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

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

BWSCON	Bit	Description	Initial state
ST3	[15]	This bit determines SRAM for using UB/LB for bank 3 0 = Not using UB/LB (Pin[14:11] is dedicated nWBE[3:0]) 1 = Using UB/LB (Pin[14:11] 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[14:11] is dedicated nWBE[3:0]) 1 = Using UB/LB (Pin[14:11] is dedicated nBE[3:0])	0
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[14:11] is dedicated nWBE[3:0]) 1 = Using UB/LB (Pin[14:11] 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	–
ENDIAN	[0]	Indicates endian mode (read only) 0 = Little endian 1 = Big endian The states are selected by ENDIAN pins	–

NOTES:

1. All types of master clock in this memory controller correspond to the bus clock.
   For example, MCLK 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	0x01C80004	R/W	Bank 0 control register	0x0700
BANKCON1	0x01C80008	R/W	Bank 1 control register	0x0700
BANKCON2	0x01C8000C	R/W	Bank 2 control register	0x0700
BANKCON3	0x01C80010	R/W	Bank 3 control register	0x0700
BANKCON4	0x01C80014	R/W	Bank 4 control register	0x0700
BANKCON5	0x01C80018	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	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
Tpac	[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	0x01C8001C	R/W	Bank 6 control register	0x18008
BANKCON7	0x01C80020	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 = FP DRAM 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
Tpac	[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 = FP DRAM [MT=01] or 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	DRAM	SDRAM	SROM	SDRAM	DRAM
Bank6	DRAM	SROM	SROM	SDRAM	DRAM	SDRAM

NOTE: SROM means ROM or SRAM type memory

REFRESH CONTROL REGISTER

Register	Address	R/W	Description	Reset Value
REFRESH	0x01C80024	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)/MCLK Ex) If refresh period is 15.6 us and MCLK 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	0x01C80028	R/W	Flexible bank size register	0x0

BANKSIZE	Bit	Description	Initial State
SCLKEN	[4]	SCLK will be generated only during SDRAM access cycle. This feature will reduce the power consumption. 1 is recommended. 0 = normal SCLK 1 = SCLK for reducing power consumption	0
Reserved	[3]	Not use	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	0x01C8002C	R/W	Mode register set register bank6	xxx
MRSRB7	0x01C80030	R/W	Mode register set register bank7	xxx

MRSR	Bit	Description	Initial State
Reserved	[11:10]	Not use	–
WBL	[9]	Write burst length 0 is the recommended value	x
TM	[8:7]	Test mode 00: mode register set, 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 (recommended) 1: N/A	x
BL	[2:0]	Burst length 000: 1 the others: N/A	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 STMIA instruction.
2. In STOP mode/SL_IDLE mode, DRAM/SDRAM has to enter the DRAM/SDRAM self-refresh mode.

NOTES

5 CLOCK & POWER MANAGEMENT

OVERVIEW

The Clock Generator in S3C44B0X can generate the required clock signals for the CPU as well as peripherals. The Clock Generator can be controlled to supply or disconnect the clock to each peripheral block by S/W, which will reduce the power. As well as this kind of S/W controllability, S3C44B0X has various power management schemes to keep optimal power consumption for a given task.

The power management in S3C44B0X consists of five modes : Normal mode, Slow mode, Idle mode, Stop mode and SL Idle mode for LCD. The Normal mode is used to supply clocks to CPU as well as all peripherals in S3C44B0X. 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 and DMA are not needed, the user can disconnect the clock to the timer and DMA to reduce power. The Slow mode is non-PLL mode. Unlike the Normal mode, the Slow mode uses an external clock directly as master clock in S3C44B0X 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. The Idle mode disconnects the clock only to CPU core while it supplies the clock to all peripherals. By using this Idle mode, power consumption due to CPU core can be reduced. Any interrupt request to CPU can wake-up from Idle mode. The Stop mode freezes all clocks to the CPU as well as peripherals by disabling PLL. The power consumption is only due to the leakage current in S3C44B0X, which is less than 10 uA. The wake-up from Stop mode can be done by external interrupt to CPU. The SL Idle mode causes 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.

FUNCTION DESCRIPTION

CLOCK GENERATION

Figure 5-1 shows a block diagram of the clock generator. The main clock source comes from an external crystal or external clock. The clock generator has an oscillator(Oscillation Amplifier) which should be connected to an external crystal, and also has a PLL (Phase-Locked-Loop) which takes the low frequency oscillator output as its input and generates the high frequency clock required by S3C44B0X. The clock generator block has the logic to generate a stable clock frequency after a reset or a stop mode.

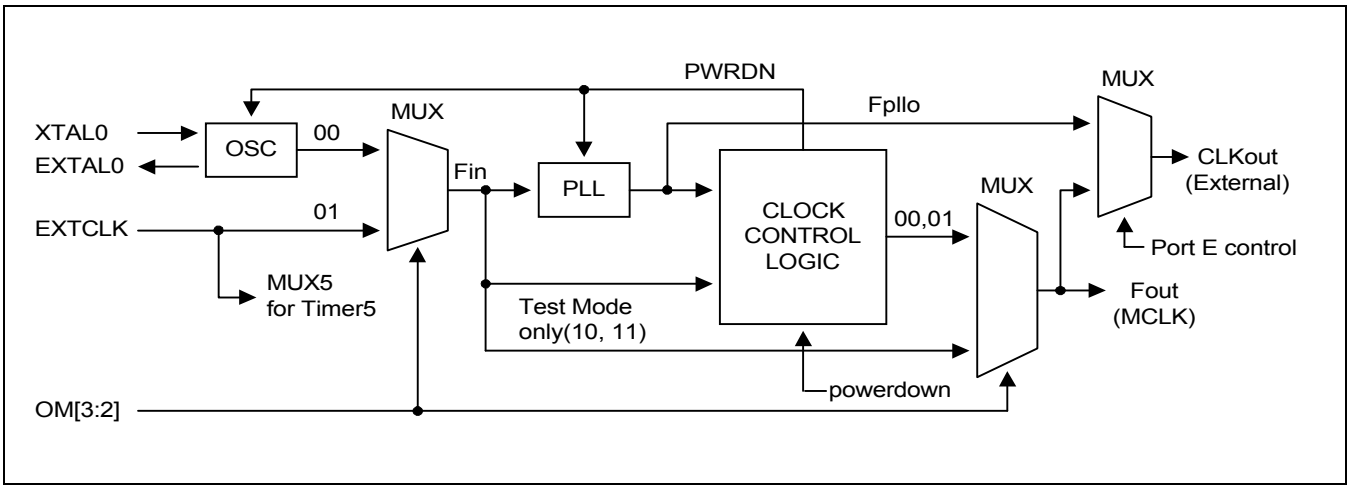


Figure 5-1. Clock Generator Block Diagram

CLOCK SOURCE SELECTION

Table 5-1 shows the relationship between the combination of mode control pins (OM3 and OM2) and the selection of source clock for S3C44B0X. The OM[3:2] status is latched internally by referring the OM3 and OM2 pins at the rising edge of nRESET.

Table 5-1. Clock source selection at boot-up

Mode OM[3:2]	Clock Source	Crystal Driver	PLL starting state	Fout
00	Crystal clock	enable	enable (1)	PLL Output (1)
01	Ext. Clock	disable	enable (1)	PLL Output (1)
Others(10, 11)	Test mode			

NOTE: Although the PLL starts just after a reset, the PLL output can not be used as Fout until the S/W writes valid settings to the PLLCON register. Before this valid setting, the clock from crystal oscillator or Ext. clock source will be used as Fout directly. Even if the user wants to maintain the default value of PLLCON register, the user should write the same value into PLLCON register.

If the S3C44B0X operates by PLL output from XTAL0 & EXTAL0, the EXTCLK can be dedicated as TCLK for Timer 5.

PLL (PHASE-LOCKED-LOOP)

The PLL 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 5-2 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 Fpllo(the output frequency from PLL block), the phase difference detector, charge pump, and loop filter. The output clock frequency Fpllo is related to the reference input clock frequency Fin by the following equation:

$$Fpllo = (m * Fin) / (p * 2^{s})$$

m = M (the value for divider M)+ 8, p = P(the value for divider P) + 2

The following sections describe the operation of the PLL, that includes the phase difference detector, charge pump, VCO (Voltage controlled oscillator), and loop filter. If the PLL is on, Fpllo is same as Fout as shown in Figure 5-1.

Phase Difference Detector (PFD)

The PFD monitors the phase difference between the Fref (the reference frequency as shown in Fig. 5-2) and Fvco (the output frequency from VCO and Divider M block) 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 output 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 5-2) is 700pF.

Voltage Controlled Oscillator (VCO)

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

Usual Condition for PLL & Clock Generator

The following conditions are generally used.

Loop filter capacitance	700-820 pF
External feedback resistance	1Mohm
External X-tal frequency	6-20 Mhz
External capacitance used for X-tal	15-22 pF

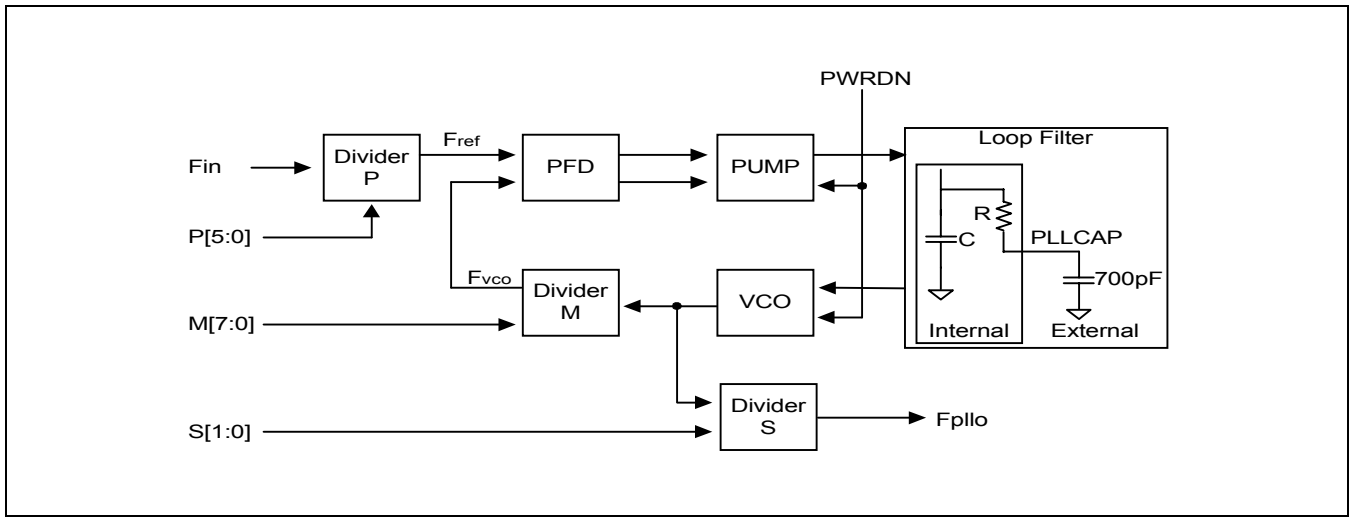


Figure 5-2. PLL (Phase-Locked Loop) Block Diagram

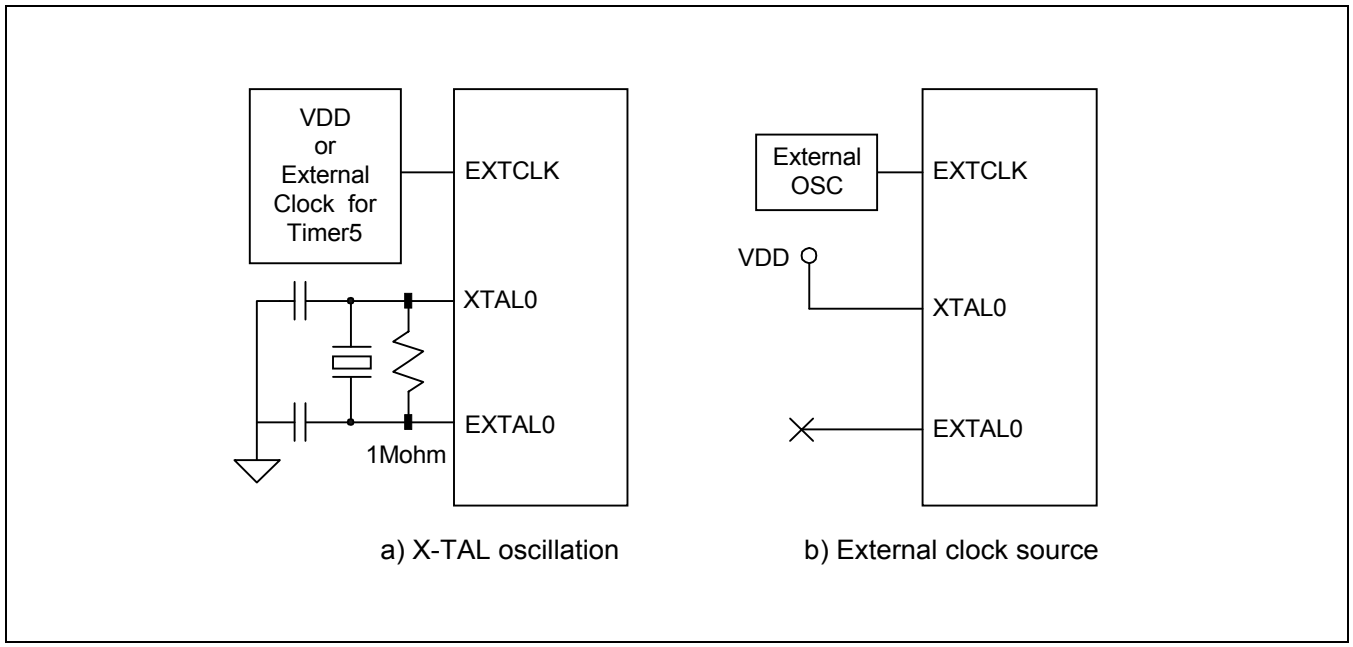


Figure 5-3. Main Oscillator Circuit Examples

CLOCK CONTROL LOGIC

The clock control logic determines the clock source to be used, i.e., the PLL clock or the direct OSC clock. When PLL is configured to a new frequency value, the clock control logic disables the FOUT 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 time required for PLL output stabilization. The lock time should be bigger than 208us. 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 = LTIMECNT value)

Power-On Reset

Figure 5-4 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 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 Fout instead of the Fpllo(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, the 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. Fout can be configured to be PLL output(Fpllo) immediately after lock time.

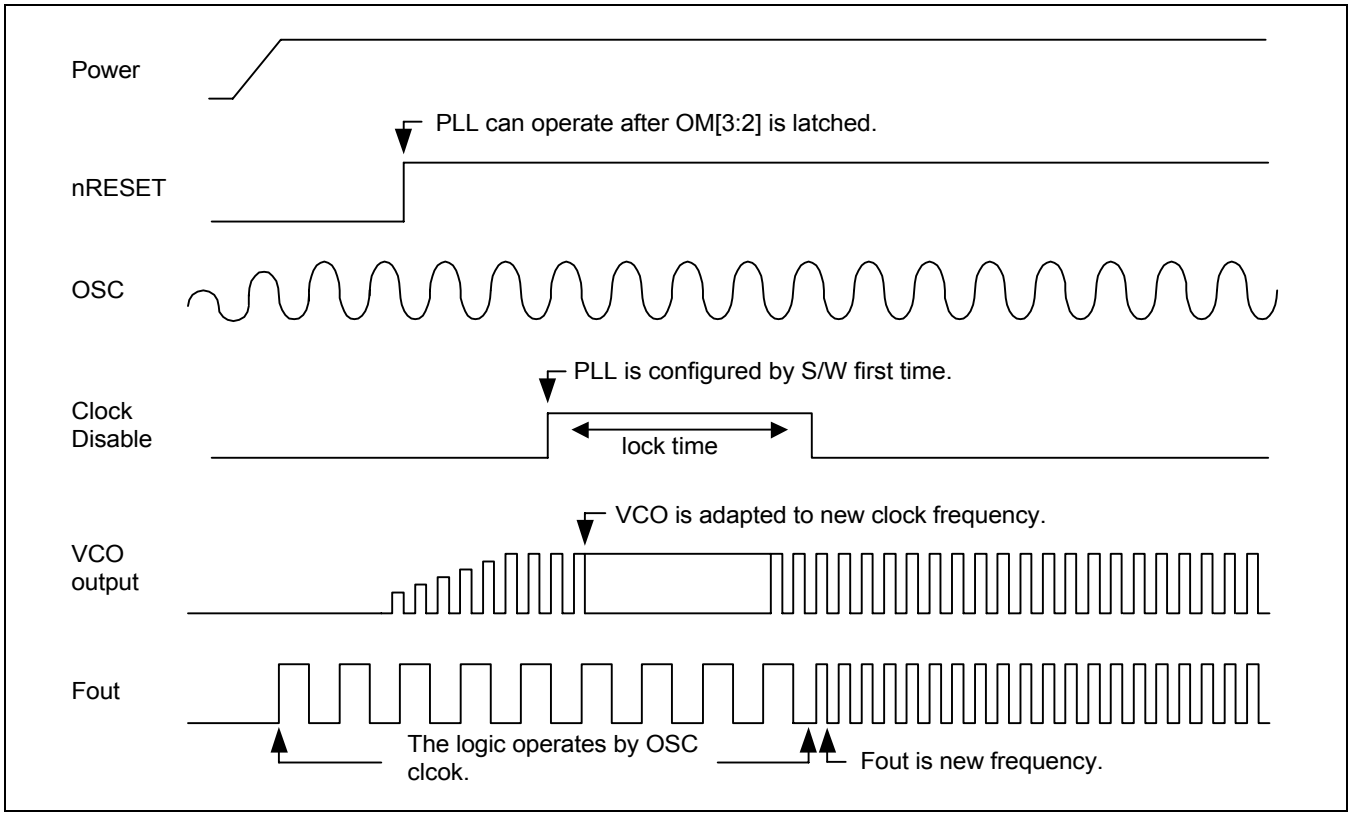


Figure 5-4. Power-On Reset Sequence

Change PLL Settings In Normal Operation Mode

During the operation of S3C44B0X 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 S3C440X. The timing diagram is as follow.

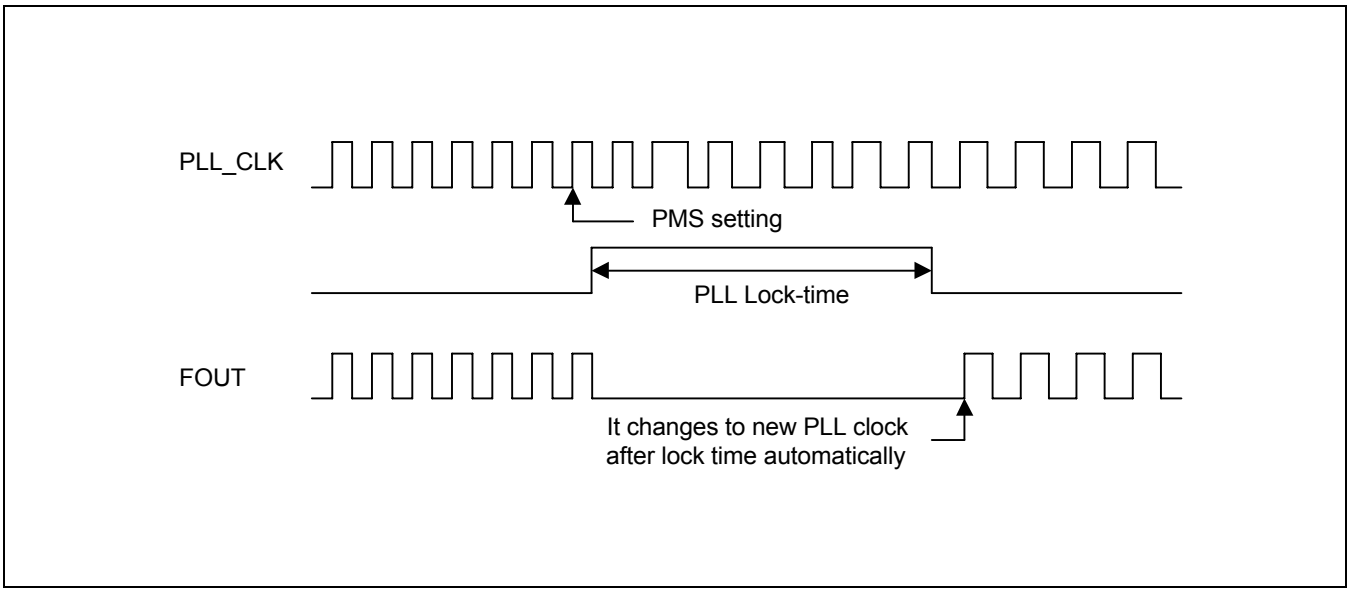


Figure 5-5. The Case that Changes Slow Clock by Setting PMS Value

POWER MANAGEMENT

The power management block controls the system clocks by software for reduction of power consumption in S3C44B0X. These schemes are related to PLL, clock control logic, peripheral clock control, and wake-up signal.

S3C44B0X 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 5-11.

Normal Mode

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

NOTE: The basic blocks consist of the CPU core, bus controller, memory controller, interrupt controller, and power management.

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[7:0], or RTC alarm interrupt, or the other interrupts should be activated. (If users are willing to use EINT[7:0], 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 THAW mode is available. In other words, the user cannot return to NORMAL mode from STOP mode as shown in Fig. 5-11, directly. To exit from STOP mode, EINT[7:0] or RTC alarm interrupt has to be activated.

Just after entering into the STOP mode, the Clock Control Logic output the Fin-clock, instead of the Fpllo-clock, from Fout during 16 Fin-clocks. After 16 Fin-clocks, the Fout is stopped and S3C44B0X enters into STOP mode completely. The latency time from command issue of the power down by STOP mode to actual entrance into power down mode is calculated as follows:

Power down latency time = Input clock period (crystal oscillator clock or external clock) * 16

If S3C44B0X is in the SLOW mode, the S3C44B0X enters into STOP mode immediately because the frequency of the clock in slow mode is lower than Fin.

The S3C44B0X can exit from STOP mode by external interrupts or RTC alarm interrupt. During the wake-up sequences, the crystal oscillator and PLL may begin to operate. The lock time is also needed to stabilize Fout. 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(alarm or external interrupt) is requested.

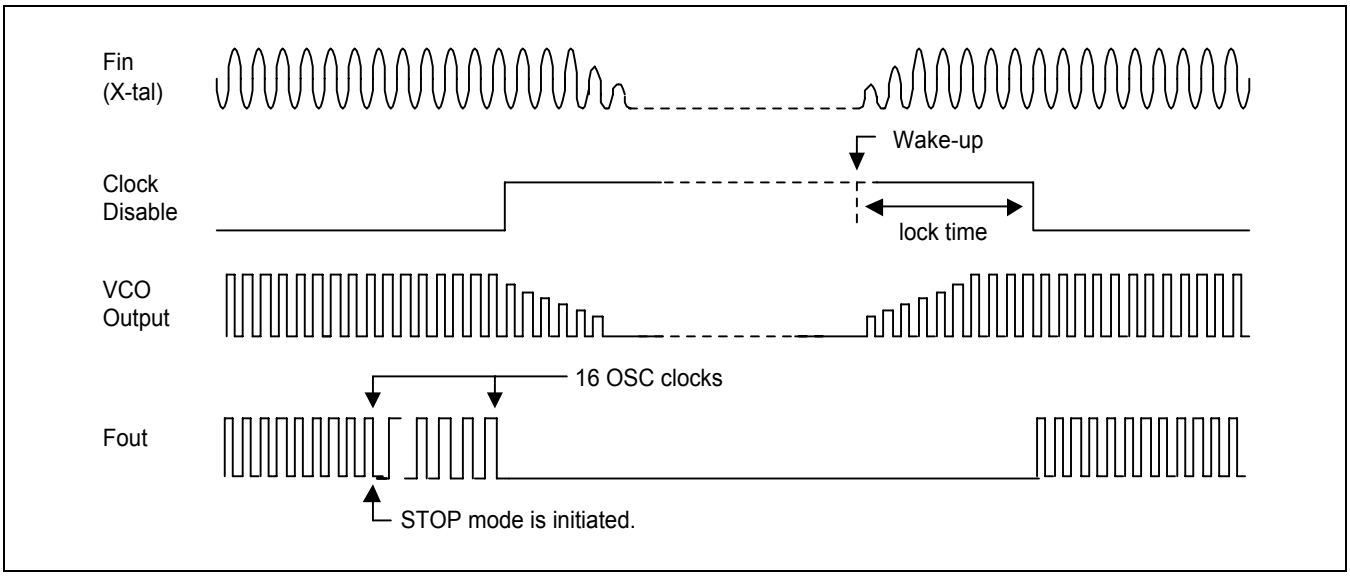


Figure 5-6. Entering STOP Mode and Exiting STOP Mode (Wake-up)

IMPORTANT NOTES

Before entering STOP mode, the following 6 items must be obeyed.

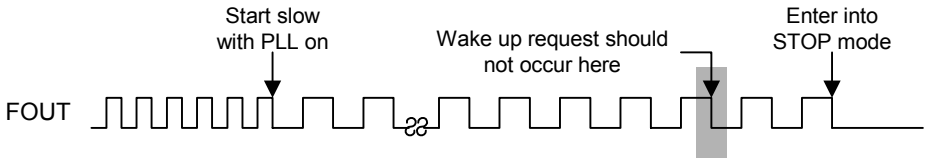
1) DRAM has to be in self-refresh mode during STOP mode to retain valid memory data.

2) LCD display must be stopped before entering STOP mode because DRAM is in self-refresh mode and LCD can't access DRAM during DRAM self-refresh mode. If LCD display is turned on, SYSTEM will be hanged up.

3) The ports of S3C44B0X must be configured properly according to your system to reduce power consumption.

4) Before entering STOP mode the CPU must be in SLOW mode with PLL on. The PLL will be turned off automatically in STOP mode.

5) For the period of entering into STOP mode, if there is any wake-up request at last 3rd clock edge before the CPU goes into STOP mode, S3C44B0X will never respond to that wake-up source. For example, if EIN0 is asserted at that point, the EINT0 cannot wake up the system anymore, however other sources can wake up the system and then the EINT0 can be used for wake-up source in the next time. So, it is strongly recommended that any wake-up signals should not be asserted until entering into STOP mode completely. If your application cannot prevent wake-up request at that clock, please refer to the workaround document, which is located on our web sight.



6) When S3C44B0X enters STOP mode, MCLK should be more than 2.5 times of Fin (X-tal frequency). After wake-up (in NORMAL mode), user can change MCLK to the frequency that user want. For example, if Fin is 20MHz and a user want MCLK=36MHz, the MCLK before entering into STOP mode should be more than 50MHz. After wake-up and S3C44B0X returns to NORMAL mode from STOP mode, MCLK can be changed from 50MHz to 36MHz by setting PLLCON register.

7) When S3C44B0X enters STOP mode in the level triggered EINT mode, the level EINT wake-up should not be active. If the level EINT wake-up is active, S3C44B0X should skip entering into STOP mode.

SL_IDLE Mode (S_LCD 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_ILDE mode than in the IDLE mode. Before entering into SL_IDLE mode, SLOW mode has to be entered and PLL has to be turned off. After SLOW mode entrance and PLL-off, 0x46(LCDC enable, IDLE enable, and SL_IDLE enable)should be written into the CLKCON register to enter into SL_IDLE mode.

To exit SL_IDLE mode, EINT[7:0] or RTC alarm interrupt has to be activated. In this case, the processor mode will change into Slow Mode automatically as shown in Fig. 5-11. To return to Normal mode, users have to wait until the end of lock time, then disable the SLOW mode and clear the SL_IDLE bit. In the PLL lock time, the SLOW clock is supplied. DRAM has to be in self-refresh mode during SL_IDLE mode to retain the valid data in DRAM.

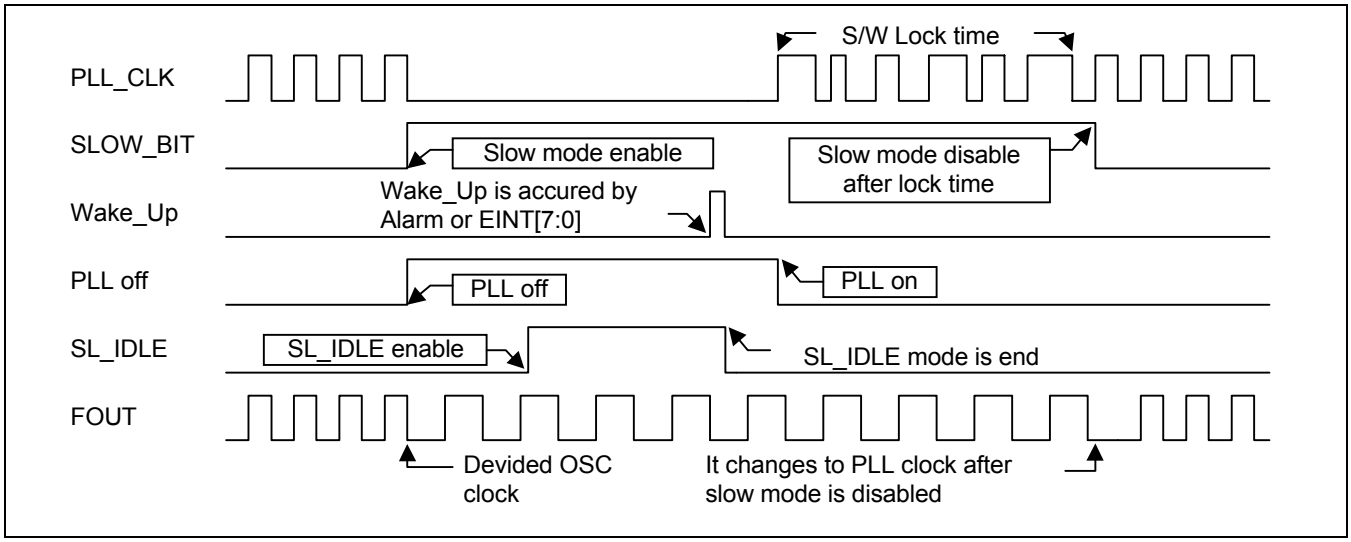


Figure 5-7. Entering SL_IDLE Mode and Exiting SL_IDLE Mode (Wake-up)

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 Fout is the frequency of divide_by_n of Fin without PLL. The divider ratio is determined by SLOW_VAL in the CLKSLOW control register.

Fout = Fin / (2 x SLOW_VAL), when SLOW_VAL is bigger than 0
Fout = Fin , when SLOW_VAL is 0

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 400us after PLL is turn on. During PLL lock time, the Fout 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 as follow.

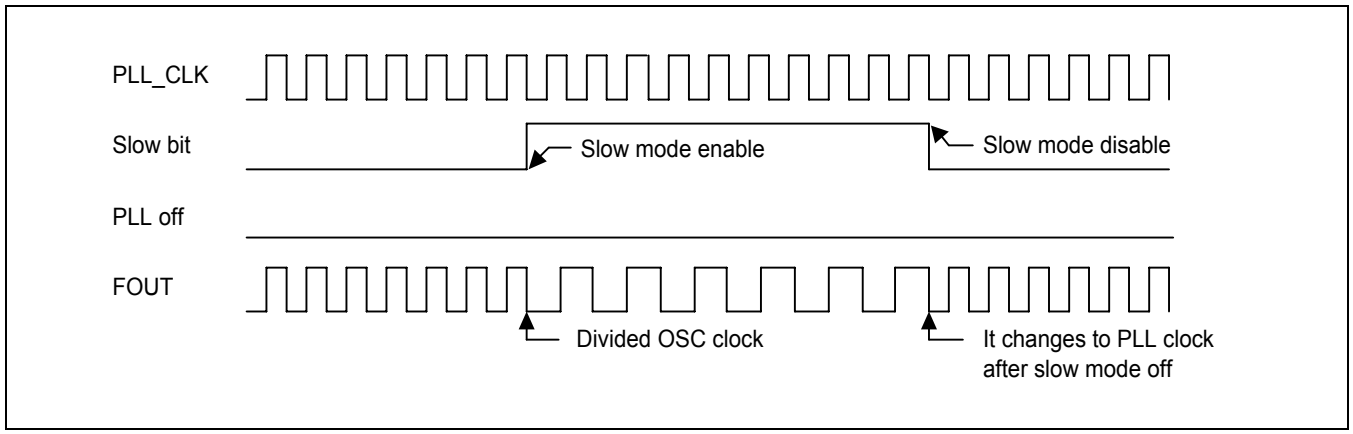


Figure 5-8. The Case that Exit_from_Slow_Mode Command is Issued in PLL on State

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

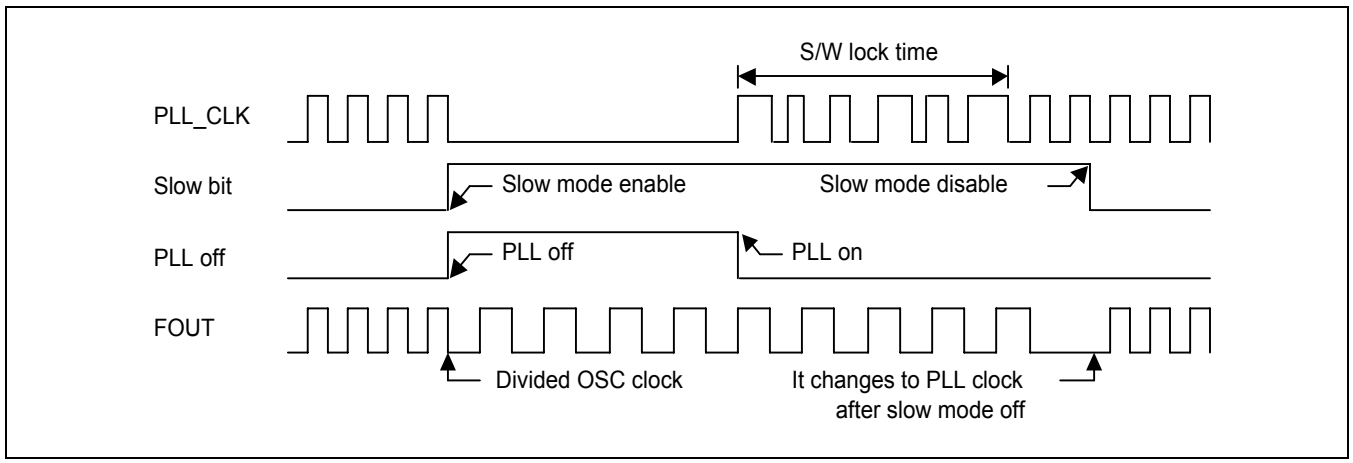


Figure 5-9. The Case that Exit_from_Slow_Mode Command is Issued after Lock Time is End

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

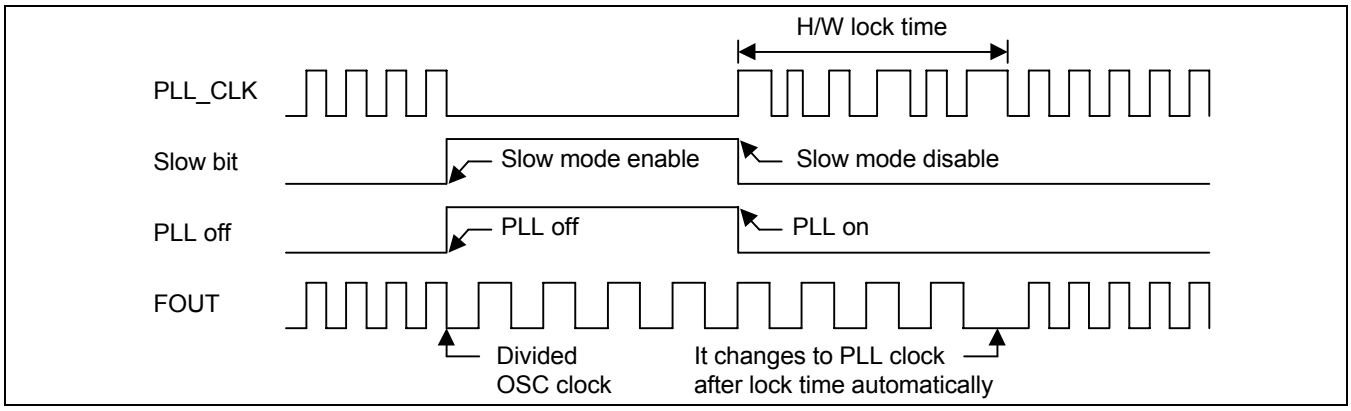


Figure 5-10. The Case that Exit_from_Slow_Mode Command is Issued the Instant PLL_on Command is Issued.

Wake-Up & THAW State

When the S3C44B0X is woken up from power down mode(STOP mode) by an EINT[7:0] or a RTC alarm interrupt, the processor state will be changed into THAW state as shown in Figure 5-11. In thaw state, the configuration of the CLKCON should be ignored because the CLKCON value, which had been set before the entrance to stop mode, can not reflect the actual processor state.

After the wake-up from STOP mode, the processor is in THAW 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 THAW state to Normal or SLOW or even STOP mode.

Just after writing the valid configuration value into the CLKCON, the mode returns to normal mode, slow mode, or even STOP mode.

Table 5-2. The Status of PLL and Fout after Wake-Up

Mode before wake-up	PLL on/off after wake up	Fout after wake up and before the lock time	Fout after the lock time by internal logic
STOP	off → on	no clock	normal mode clock
SL_IDLE	–	slow mode clock	–
IDLE	unchanged	unchanged	unchanged

Signaling EINT[7:0] For Wake-Up

The S3C44B0X 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 PCONG register.

It is important to configure the EINTn in the PCONG 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, S3C44B0X 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.

PUPS register and STOP/SL_IDLE mode

In STOP/SL_IDLE 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 PUPC and PUPD registers. D[15:0] pin pull-up resistors can be controlled by the PUPS 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 in ADCCON. If S3C44B0X enters the STOP mode, the ADC should enter it's own power-down mode.

POWER MANAGEMENT STATE MACHINE

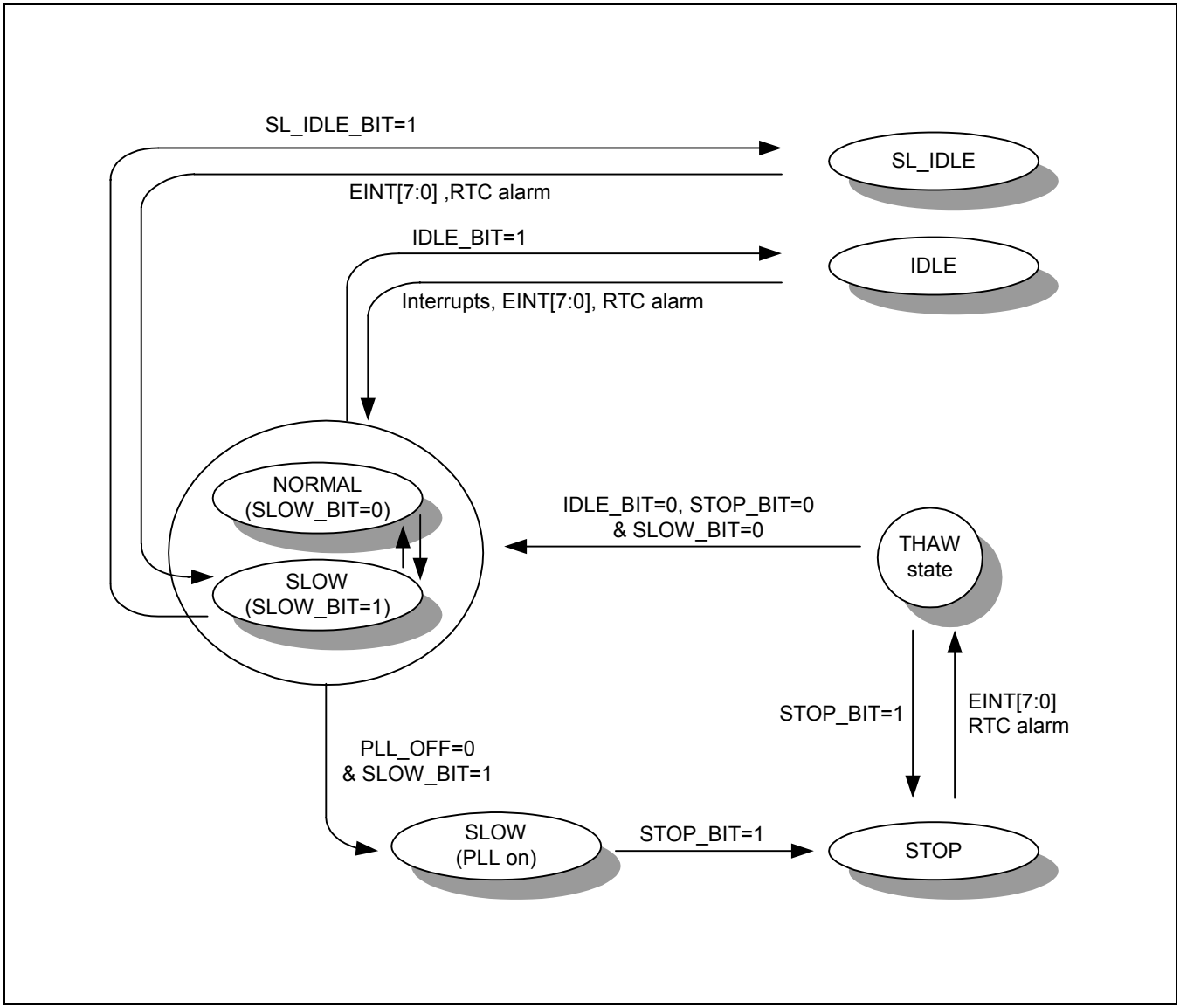


Figure 5-11. Power Management State Machine

CLOCK GENERATOR & POWER MANAGEMENT SPECIAL REGISTER

PLL CONTROL REGISTER (PLLCON)

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

NOTE: Fpllo must be greater than 20Mhz and less than 66Mhz.

Example

If Fin=14.318Mhz and Fout=60Mhz, the calculated value is as follows;
MDIV=59, PDIV=6 and SDIV=1 (This value may be calculated using PLLSET.EXE utility, provided by SAMSUNG.)

PLL VALUE SELECTION GUIDE

1. Fpllo * 2^s has to be less than 170 MHz.
2. S should be as great as possible.
3. (Fin / p) is recommended to be 1Mhz or above. But, (Fin / p) < 2Mhz.

Register	Address	R/W	Description	Reset Value
PLLCON	0x01D80000	R/W	PLL configuration Register	0x38080

PLLCON	Bit	Description	Initial State
MDIV	[19:12]	Main divider control	0x38
PDIV	[9:4]	Pre-divider control	0x08
SDIV	[1:0]	Post divider control	0x0

CLOCK CONTROL REGISTER (CLKCON)

Register	Address	R/W	Description	Reset Value
CLKCON	0x01D80004	R/W	Clock generator control Register	0x7ff8

CLKCON	Bit	Description	Initial State
IIS	[14]	Controls MCLK into IIS block 0 = Disable, 1 = Enable	1
IIC	[13]	Controls MCLK into IIC block 0 = Disable, 1 = Enable	1
ADC	[12]	Controls MCLK into ADC block 0 = Disable, 1 = Enable	1
RTC	[11]	Controls MCLK into RTC control block. Even if this bit is cleared to 0, RTC timer is alive. 0 = Disable, 1 = Enable	1
GPIO	[10]	Controls MCLK into GPIO block Set to 1 to use interrupt requests by EINT[4:7] 0 = Disable, 1 = Enable	1
UART1	[9]	Controls MCLK into UART1 block 0 = Disable, 1 = Enable	1
UART0	[8]	Controls MCLK into UART0 block 0 = Disable, 1 = Enable	1
BDMA0,1	[7]	Controls MCLK into BDMA block 0 = Disable, 1 = Enable (If BDMA is turned off, the peripherals in the peripheral bus may not be accessed)	1
LCDC	[6]	Controls MCLK into LCDC block 0 = Disable, 1 = Enable	1
SIO	[5]	Controls MCLK into SIO block 0 = Disable, 1 = Enable	1
ZDMA0,1	[4]	Controls MCLK into ZDMA block 0 = Disable, 1 = Enable	1
PWMTIMER	[3]	Controls MCLK into PWMTIMER block 0 = Disable, 1 = Enable	1
IDLE BIT	[2]	Enters IDLE mode. This bit can't be cleared automatically. 0 = Disable, 1 = Transition to IDLE(SL_IDLE) mode	0
SL_IDLE	[1]	SL_IDLE mode option. This bit can't be cleared automatically. 0 = Disable, 1 = SL_IDLE mode. To enter SL_IDLE mode, CLKCON register has to be 0x46.	0
STOP BIT	[0]	Enters STOP mode. This bit can'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	0x01D80008	R/W	Slow clock control register	0x9

CLKSLOW	Bit	Description	Initial State
PLL_OFF	[5]	0 : PLL is turned on. PLL is turned on only when SLOW_BIT is 1. After PLL stabilization time (minimum 150uS), SLOW_BIT may be cleared to 0. 1 : PLL is turned off. PLL is turned off only when SLOW_BIT is 1.	0x0
SLOW_BIT	[4]	0 : Fout = Fpllo (PLL output) 1: Fout = Fin / (2 x SLOW_VAL), (SLOW_VAL > 0) Fout = Fin, (SLOW_VAL = 0)	0x0
SLOW_VAL	[3:0]	The divider value for the slow clock when SLOW_BIT is on.	0x9

LOCK TIME COUNT REGISTER (LOCKTIME)

Register	Address	R/W	Description	Reset Value
LOCKTIME	0x01D8000C	R/W	PLL lock time count register	0xfff

LOCKTIME	Bit	Description	Initial State
LTIME CNT	[11:0]	PLL lock time count value	0xfff

NOTES

6 CPU WRAPPER & BUS PRIORITIES

OVERVIEW

The CPU wrapper consists of a cache, write buffer, and CPU core. The bus arbitration logic determines the priority of each bus master.

The CPU wrapper has an 8-Kbyte internal memory. The internal memory can be used in three ways. First the 8-Kbyte memory can be used as an 8KB unified (instruction/data) cache. Second, the internal memory can be used as a 4-Kbyte unified cache and a 4-Kbyte internal SRAM. Third, the internal memory can be used wholly as an 8-Kbyte internal SRAM.

The internal unified (instruction/data) cache adopts four-way set associative architecture with a four-word (16 bytes) line size. It has a write-through policy to keep data coherency. When a cache miss occurs, four words of memory are fetched sequentially from external memory. It has an LRU (Least Recently Used) algorithm to raise the hit ratio. The unified cache deals with instruction and data by distinguishing them.

The internal SRAM mainly will be used to reduce ISR(interrupt service routine) execution time. ISR execution time will be reduced because the internal SRAM has the fastest access time. Also, the ISR in SRAM is very efficient because most ISR codes may cause cache-miss.

The bus arbitration logic can determine the priorities of bus masters. The bus arbitration logic supports a round-robin priority mode and a fixed priority mode. Also, The priorities among LCD_DMA, BDMA, ZDMA, nBREQ (external bus masters) can be changed by S/W.

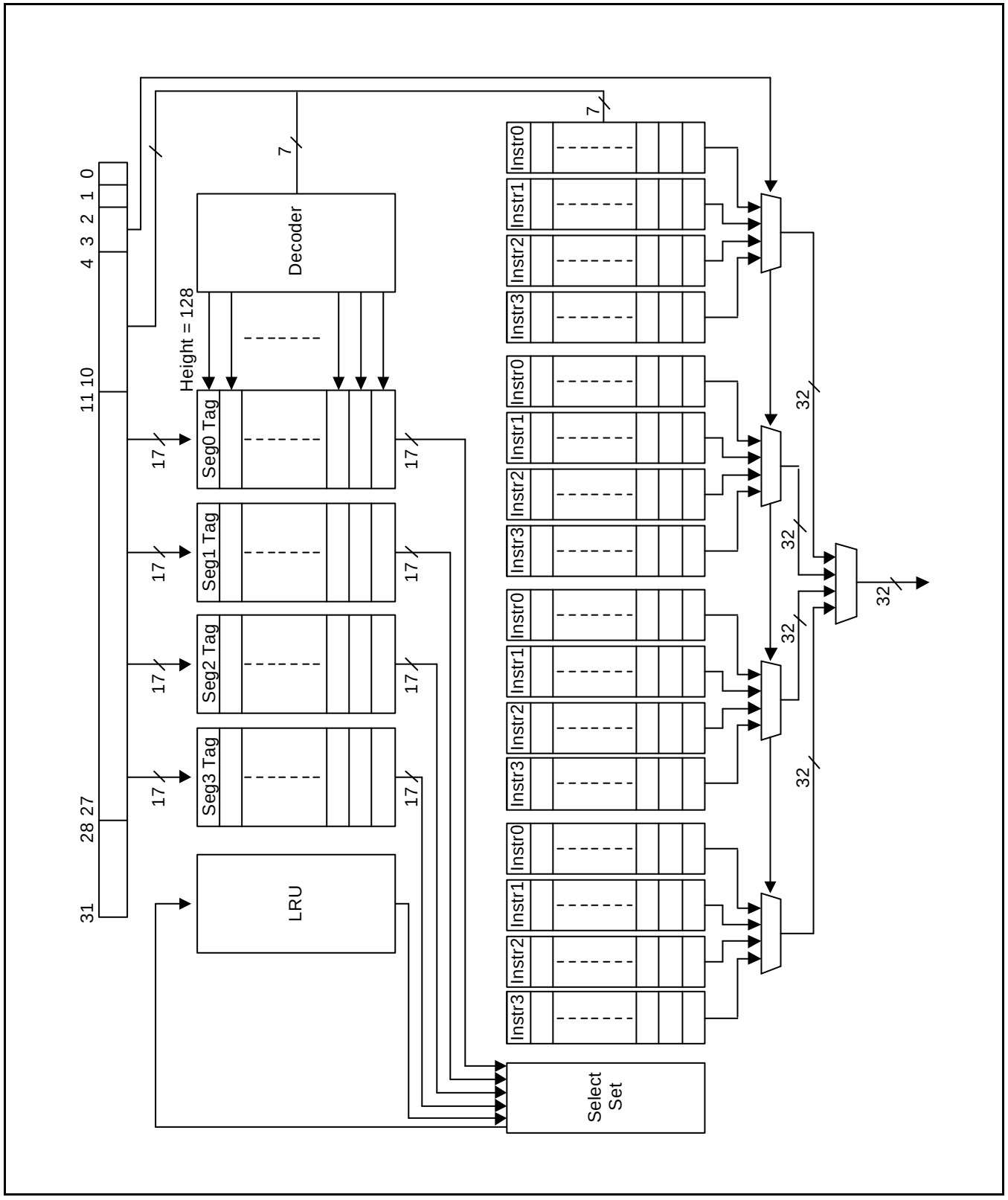


Figure 6-1. Cache Memory Configuration

CACHE OPERATION

Cache Organization

S3C44B0X cache has an 8KB (or 4KB) cache memory, four Tag RAMs and one LRU memory. The internal unified (instructions/data) cache adopts a four-way set associative architecture with 4-word (16 bytes) line size. It has a write-through policy to keep data coherency. It has an LRU (Least Recently Used) algorithm to raise the hit ratio.

Cache Replace Operation

After a system is initialized, the value of CS is set to "0000", signifying that the contents of set 0, set1, set2 and set 3 cache memories are invalid. When a cache fill occurs, the value of CS is changed to "0110" at the specified line, which signifies that only set 0 is valid. When the subsequent cache fill occurs, the value of CS will be "0011" at the specified line, which represents that contents of both set 0 and set 2 are valid. When the subsequent cache fill occurs, the value of CS will be "0101" at the specified line, which represents that contents of set 0, set 1, and set 2 are valid. And succesive cache fill make CS "1000" at the specified line, which represents that all caches are valid.

The value of CS "1xxx" represents that all of the sets are valid. Then the next cache miss occurs, the least significant 3 bits of CS select set are replaced. First bit selects a group of sets. "0" selects group 0 (which contains set0 and set1) , otherwise group1 (which contains set2 and set3). Second bit selects the set of group 0 . "0" selects set0, otherwise set1. Third bit selects the set of group 1. "0" selects set2, otherwise set 3. For example, if LS 3bit is 000, the victim is set0. If LS 3bit is "101", the victim is set3.

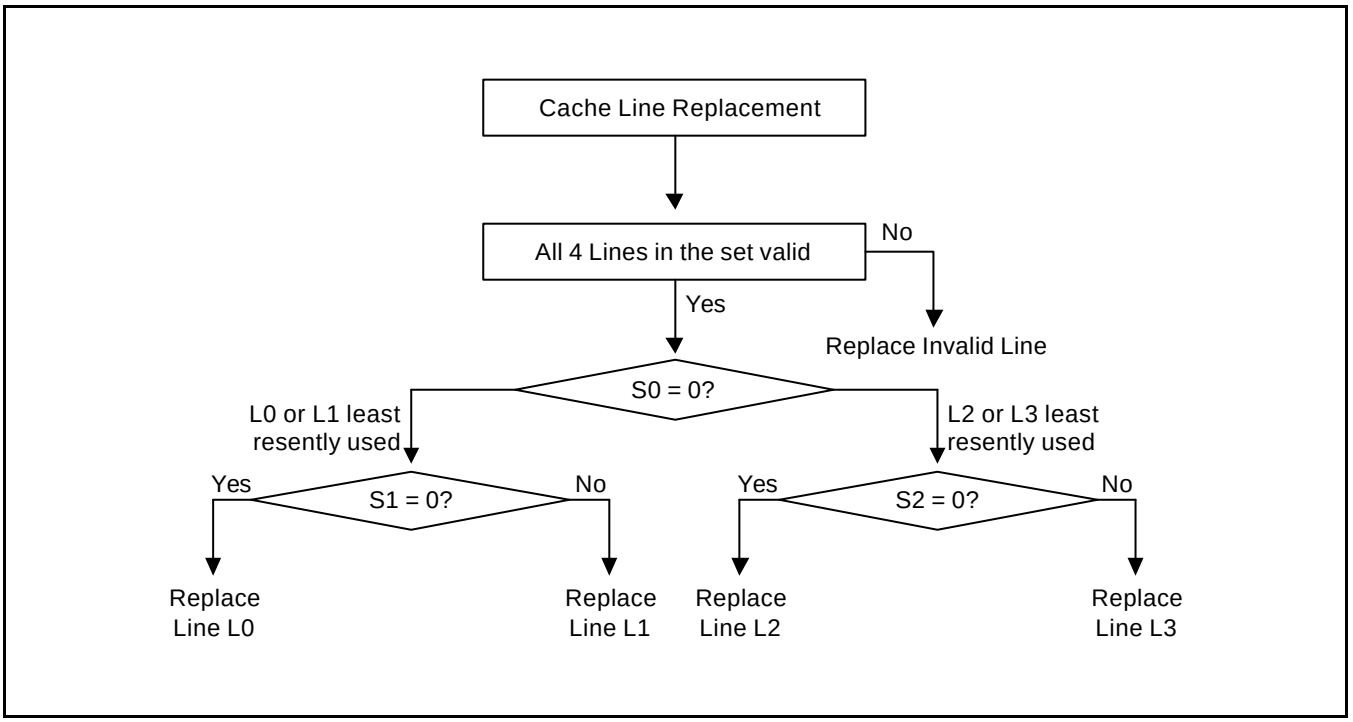


Figure 6-2. Cache Replace Configuration

Cache Disable Operation

The S3C44B0X cache provides the entire-cache-enable/disable mode. You can enable cache by setting the value of CM in SYSCFG to 01 or 11, and disable it by clearing SYSCFG[2:1] to 00. When the cache disable mode is specified, instructions and data are always fetched from external memory. The S3C44B0X can also provide non-cacheable areas in cache-enable-mode for some particular memory access operations, such as the DMA operation. The two non-cacheable areas are specified by four special registers to be introduced later.

Data coherency is important when the cache memory is re-enabled because the cache memory does not have auto flush mode. You also have to be cautious whether or not DMA changes memory data. The DMA accessible memory area should be non-cacheable to keep data coherency. To keep data coherency between cache and external memory, S3C44B0X uses the write-through method.

Cache Flushing

A cache flushing can re-enable the cache operation. When the cache is disabled, the LRU RAM can be manipulated exactly like normal memory. The cache can be flushed by writing 0 to the LRU RAM and making all cache data invalid. The memory location of the LRU memory is as follows:

NOTE

Cache flushing must be executed only in the cache disable mode.

Non-Cacheable Area

The S3C44B0X provides two non-cacheable areas. Each of them requires two cache control fields, which indicate the start and end address of each non-cacheable area. In a non-cacheable area, the cache is not updated when cache misses a read.

Usually a cache stores any data in the whole system memory area, but sometimes it needs a non-cacheable area because the cache cannot keep track of the external memory device whose contents are changed without read/write operation.

The size of a non-cacheable area can be increased/decreased by 4KB units. The end address has to point the next 4KB block. For example, if non-cacheable area is 0x10000~0x22fff, the start address value of NCACHBEn is 0x10 and the end address value of NCACHBEn is 0x23.

To Speed Up a Program Execution by Considering Cache Usage

1. Locates the ISRs, which is executed most frequently, on the internal SRAM.
2. Let ISR not be cached. Most ISR codes cause a cache miss, and the codes in the cache memory are not re-used because the code is erased by main codes, executed after exiting the ISR.
3. Locates the functions which are related to each other together and executes them concurrently. This function aggregation reduce cache misses.
4. Somtimes, if the data area is assigned as non cachable area, the program execution speed will be higher, because most variables are not re-used. Refreshing the 16 byte cache memory is wasteful for un-reused variables.

INTERNAL SRAM (INTERNAL MEMORY MAP)

S3C44B0X has a maximum 8 KB 4way set associative cache or internal SRAM. If the internal SRAM is 4 KB, the other 4KB internal memory can be used as a 2 way set associative cache. The memory access cycle of the internal SRAM is 1 MCLK cycle.

Cache Size	SRAM Size	Note
8KB (4way)	None	4 way set associative
4KB (2way)	4KB	2 way set associative SRAM uses the area allocated for sets 2 and 3 of 8KB cache.
None	8KB	4 way set associative

Addresses in a set memory is increased sequentially and addresses in TAG/LRU increases of 16byte.
Don't access the interval addresses between 0x10003004 and 0x1000300f .

Area(Set/Cache)	Memory Map Address	Size
cache set 0	0x10000000 - 0x100007ff	2KB
cache set 1	0x10000800 - 0x10000fff	2KB
cache set 2	0x10001000 - 0x100017ff	2KB
cache set 3	0x10001800 - 0x10001fff	2KB
cache tag 0	0x10002000 - 0x100027f0	512bytes (note)
cache tag 1	0x10002800 - 0x10002ff0	512bytes (note)
cache tag 2	0x10003000 - 0x100037f0	512bytes (note)
cache tag 3	0x10003800 - 0x10003ff0	512bytes (note)
LRU	0x10004000 - 0x100047f0	512bytes (note)

NOTE: The cache tag3:0 & LRU must be read/written by word access (32bit). The address bit[3:0] of .tag & LRU must be 0. For example, if you want to read the 2nd item among 128 cache tag 0 items, you should not read the address 0x10002004, but 0x10002010. Therefore, the tag0 addresses are 0x10002000, 0x10002010, 0x10002020,..., 0x100027f0.

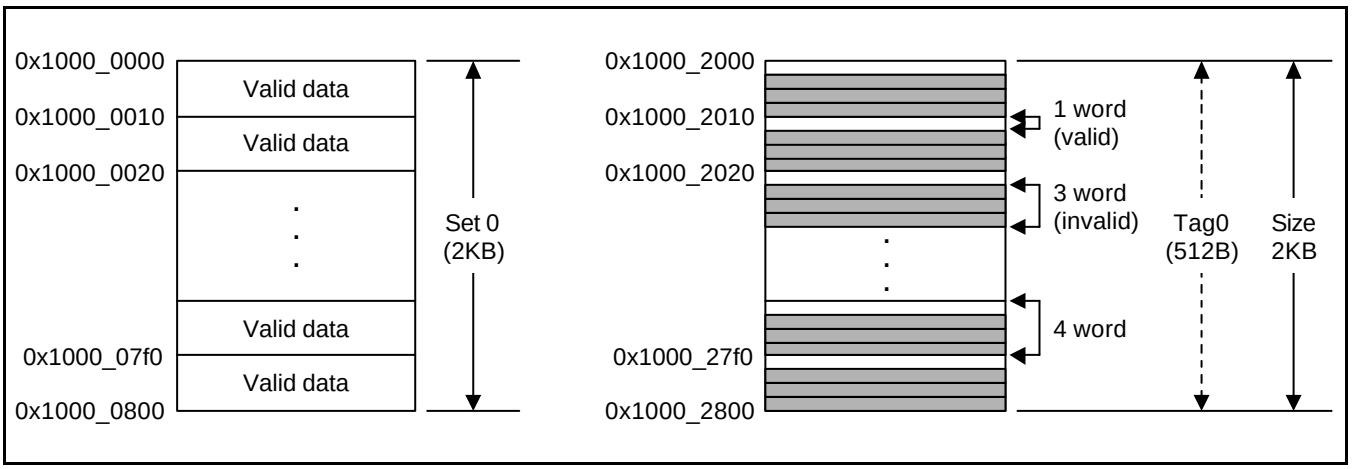


Figure 6-3. Cache Memory Mapping

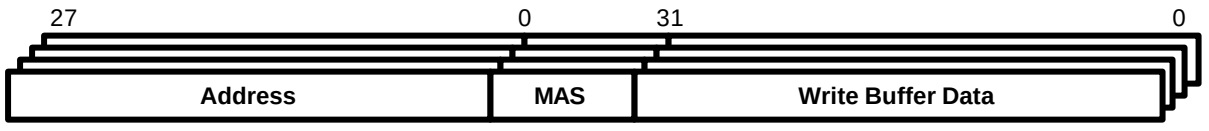
WRITE-BUFFER OPERATION

Write Buffer Operation

S3C44B0X has four write buffer registers to enhance memory writing performance. When the write buffer mode is enabled, the CPU writes data into the write buffer registers instead of an external memory even when the external bus is already occupied by another bus master like DMA.

The write buffer block will write the data when the system bus is not occupied by higher priority bus masters. Also, CPU performance will be enhanced because the CPU does not have to wait the completion of the write operation.

The write buffer has 4 registers. Each register includes a 32-bit data field, a 28-bit address field, and a 2-bit status field.



[31:0] Write Buffer Data
Data to be written into external memory

[1:0] MAS
00 = 8-bit data mode
01 = 16-bit data mode
10 = 32-bit data mode
11 = Not used

[27:0] Address
Indicates the address of write data

Figure 6-4. Write Buffer Configuration

BUS PRIORITY MAP

In S3C44B0X, there are seven bus masters, LCD_DMA, BDMA0, BDMA1, ZDMA0, ZDMA1, nBREQ (external bus masters), and CPU wrapper. The priorities among these bus masters after a reset are as follows;

1. DRAM refresh controller
2. LCD_DMA
3. ZDMA0,1
4. BDMA0,1
5. External bus master
6. Write buffer
7. Cache & CPU

The bus priorities among LCD_DMA, ZDMA, BDMA, and an external bus master can be programmed by the SBUSCON register. The CPU wrapper always has the lowest priority regardless of the SBUSCON register.

The round-robin priority mode or fixed priority mode can be selected. In the round-robin priority mode, the bus master which had once served will have the lowest priority. In this way, all the bus masters have equal priorities.

In the fixed priority mode, each bus master's priority is written onto SBUSCON. SBUSCON determines which is 1st - 4th priority bus master.

CPU WRAPPER SPECIAL REGISTERS

There are 3 control registers for the CPU wrapper block (cache, write buffer, and ARM7TDMI). SYSCFG register controls the general system operation. NCACHBE0 & NCACHBE1 registers provide non-cacheable areas.

SYSTEM CONFIGURATION REGISTER (SYSCFG)

Register	Address	R/W	Description	Reset Value
SYSCFG	0x01C00000	R/W	System Cofiguration Register	0x01

SYSCFG	Bit	Description	Initial State
Reserved	[7]	Reserved to 0	0
Reserved	[6]	Reserved to 0	0
DA(reserved)	[5]	DATA ABORT controls. This bit is recommended to be 0. 0: Enable data abort 1: Disable data abort	0
RSE(reserved)	[4]	Enable read stall option. This bit is recommended to be 0. 0: read stall disable 1: read stall enable (Read stall option: Insert one internal wait cycle when reading data for cache & CPU core.)	0
WE	[3]	This bit determines write buffer enable / disable. Some external devices, which require the minimum writing cycle time, do not operate normally because the period between consecutive writings is shortened the write buffer. 0 = Disable write buffer operation 1 = Enable write buffer operation	0
CM	[2:1]	These two bits determine cache mode 00 = Disable cache (8KB internal SRAM) 01 = Half cache enable (4KB cache, 4KB internal SRAM) 10 = Reserved 11 = Full Cache enable (8KB cache)	00
SE	[0]	Enable stall option. This bit is recommended to be 0. 0:stall disable 1:stall enable (Stall option: Insert one internal wait cycle when a non-sequential address is generated for caching)	1

NON-CACHEABLE AREA CONTROL REGISTER (NCACHBEn)

Register	Address	R/W	Description	Reset Value
NCACHBE0	0x01C00004	R/W	Start address & end address of non-cacheable area 0	0x00000000
NCACHBE1	0x01C00008	R/W	Start address & end address of non-cacheable area 1	0x00000000

NCACHBE0	Bit	Description	Initial State
SE0	[31:16]	End address of non-cacheable area 0. These 16 bits provide the end address of non-cacheable area 0. The minimum non-cacheable area is 4 Kbytes. SE0 = (End address + 1)/4K	0x0000
SA0	[15:0]	Start address of non-cacheable area 0. These 16 bits provide the start address of non-cacheable area 0. SA0 = Start address/4K	0x0000

NCACHBE1	Bit	Description	Initial State
SE1	[31:16]	End address of non-cacheable area 1 These 16 bits provide the end address of non-cacheable area 1. The minimum non-cacheable area is 4Kbytes. SE1 = (End address + 1)/4K	0x0000
SA1	[15:0]	Start address of non-cacheable area 1. These 16 bits provide the start address of non-cacheable area 1. The minimum non-cacheable area is 4Kbytes. SA1 = Start address/4K	0x0000

BUS PRIORITY SPECIAL REGISTER

SYSTEM BUS PRIORITY CONTROLLER (SBUSCON)

Register	Address	R/W	Description	Reset Value
SBUSCON	0x01C40000	R/W	Determines the bus priorities among the bus masters	0x80001B1B

SBUSCON	Bit	Description	Initial State
FIX	[31]	0: round-robin priorities 1: fixed priorities	0x1
S_LCD_DMA	[15:14]	Indicates the LCD_DMA bus priority (read only) 00: 1st 01: 2nd 10: 3rd 11: 4th	00
S_ZDMA	[13:12]	Indicates the ZDMA bus priority (read only) 00: 1st 01: 2nd 10: 3rd 11: 4th	01
S_BDMA	[11:10]	Indicates the BDMA bus priority (read only) 00: 1st 01: 2nd 10: 3rd 11: 4th	10
S_nBREQ	[9:8]	Indicates the nBREQ bus priority (read only) 00: 1st 01: 2nd 10: 3rd 11: 4th	11
LCD_DMA	[7:6]	Determines the LCD_DMA bus priority 00: 1st 01: 2nd 10: 3rd 11: 4th	00
ZDMA	[5:4]	Determines the ZDMA bus priority 00: 1st 01: 2nd 10: 3rd 11: 4th	01
BDMA	[3:2]	Determines the BDMA bus priority 00: 1st 01: 2nd 10: 3rd 11: 4th	10
nBREQ	[1:0]	Determines the nBREQ bus priority 00: 1st 01: 2nd 10: 3rd 11: 4th	11

NOTE: The priorities are only valid in the fixed priority mode.

7 DMA

OVERVIEW

The S3C44B0X has 4 channel DMA Controllers. The two DMAs(we call it ZDMA : General DMA) are attached to SSB (Samsung System Bus) and the other two DMAs(we call it as BDMA : Bridge DMA) are inside the bridge, which is an interface layer between SSB and SPB (Samsung Peripheral Bus).

The two ZDMA controllers attached to SSB are to transfer data from memory to memory, from memory to I/O memory(Fixed destination), and from I/O devices and I/O devices to memory. The other two BDMA controllers transfer data from memory to I/O devices and I/O devices to memory. In this case, I/O devices means the peripherals, attached to SPB like SIO, IIS and UART. The main advantage of DMA is that it can transfer the data without CPU intervention. The operation of ZDMA and BDMA can be initiated by S/W, the request from internal peripherals or the external request pins (nXDREQ0,1).

The most important feature in ZDMA is the on-the-fly mode, which reduces the number of cycles during DMA operation between external memory and a fixed external peripheral (Fixed source or destination addressed device). Usually, the DMA transfer consists of two separate cycles, one is 'Read' from the source memory or I/O device and the other is 'Write' to memory or destination I/O device. To perform these operations, the memory controller reads the data on data bus and writes this data to data bus, again. The on-the-fly-mode has inseparable Read/Write cycle. In other words, the memory controller generates the acknowledge signal for the source or destination device to read or write data on the data bus. At the same time, the memory controller also generates the Read or Write-related control signals for memory access. This kind of on-the-fly-mode can reduce the number of required DMA cycles, different from the general DMA cycles, which has separate Read and Write cycles. To operate the on-the-fly mode, the bus size of the source should be the same as that of the destination.

ZDMA/BDMA OPERATION

ZDMA (GENERAL DMA)

Figure 7-1 shows the internal diagram of a ZDMA block. The ZDMA is interfaced to SSB and can transfer data from external memory to external memory. Unlikely the BDMA(Bridge DMA), this DMA can be used to transfer data between memory-mapped device or memories. In other words, data transfer between fixed source and external memory, external memory and external memory, and external memory and fixed destination can be done by using this DMA. The DMA operation can be started by S/W or an external DMA request signal, which will be explained later.

In the ZDMA, there is a temporary buffer which allows multiple transfers to enhance bus utilization as well as transfer speed. In other words, the S3C44B0X has a 4-word FIFO-type buffer to support the 4-word burst transfer during DMA operation. For example, during the DMA operation between memories, a 4-word burst write happens after a 4-word burst read.

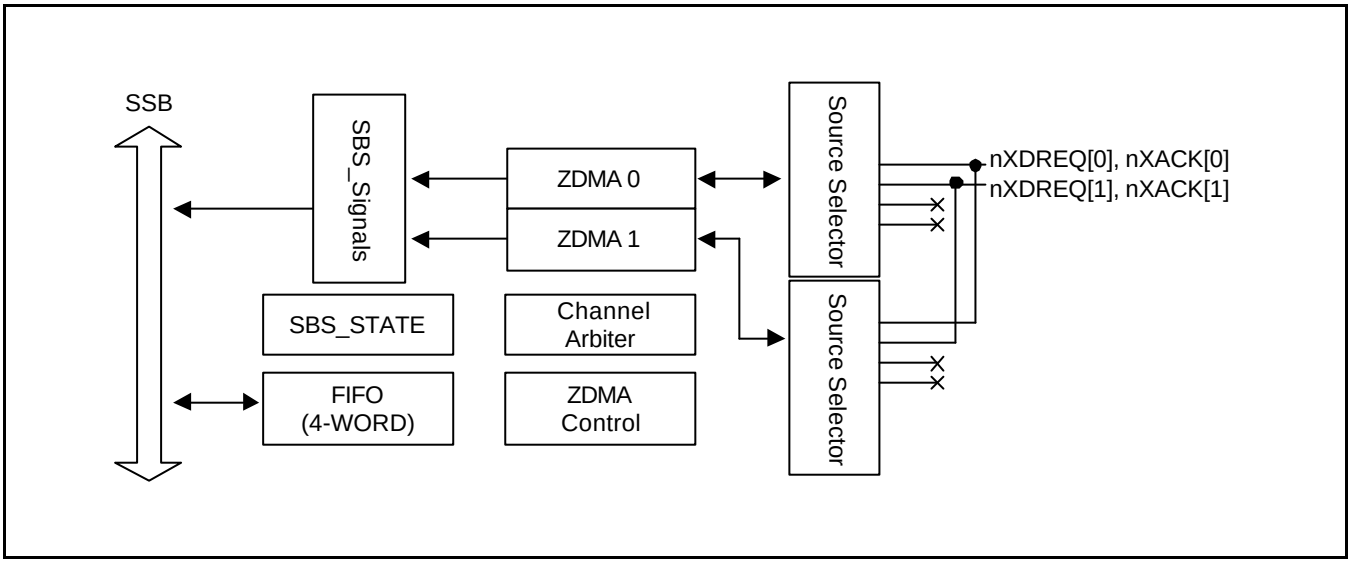


Figure 7-1. ZDMA Controller Block Diagram

BDMA (BRIDGE DMA)

Figure 7-2 shows the internal diagram of a BDMA block. The BDMA is in the Bridge, which is the interface layer between SSB and SPB. The main role of BDMA is to transfer the data between external memory and internal peripherals like UART, IIS and SIO, which are attached to SPB. The timer can also request a DMA operation anytime; it is useful for operating the ADC block automatically. Usually, the CPU or other master devices should access the external memory through memory controller, which is attached to SPB. Please be reminded that the BDMA is also a type of master device. To transfer the data from memory (peripheral devices) to peripheral devices (memory) attached to SPB (SSB), the memory controller attached to SSB should be used. Because the BDMA is in the Bridge, which is an interface layer between SSB and SPB, it can transfer the data between two devices, which are attached to SSB as well as SPB.

The BDMA cannot support a 4-word burst transfer (the block transfer mode) because BDMA does not have a temporary buffer and because the peripheral devices attached to SPB is slow. Specifically, the BDMA can support the data transfer from external memory to external memory, a slightly ineffective way of data transfer if you look at the block diagram. Even if BDMA can support the data transfer between external memories, ZDMA is recommended for use instead of to transfer data between external memories faster transfer and optional bus utilization are required. But, if more number of DMA channels for data transfer between external memories (Maximum 2 channels using ZDMA) is needed, the BDMA can be used.

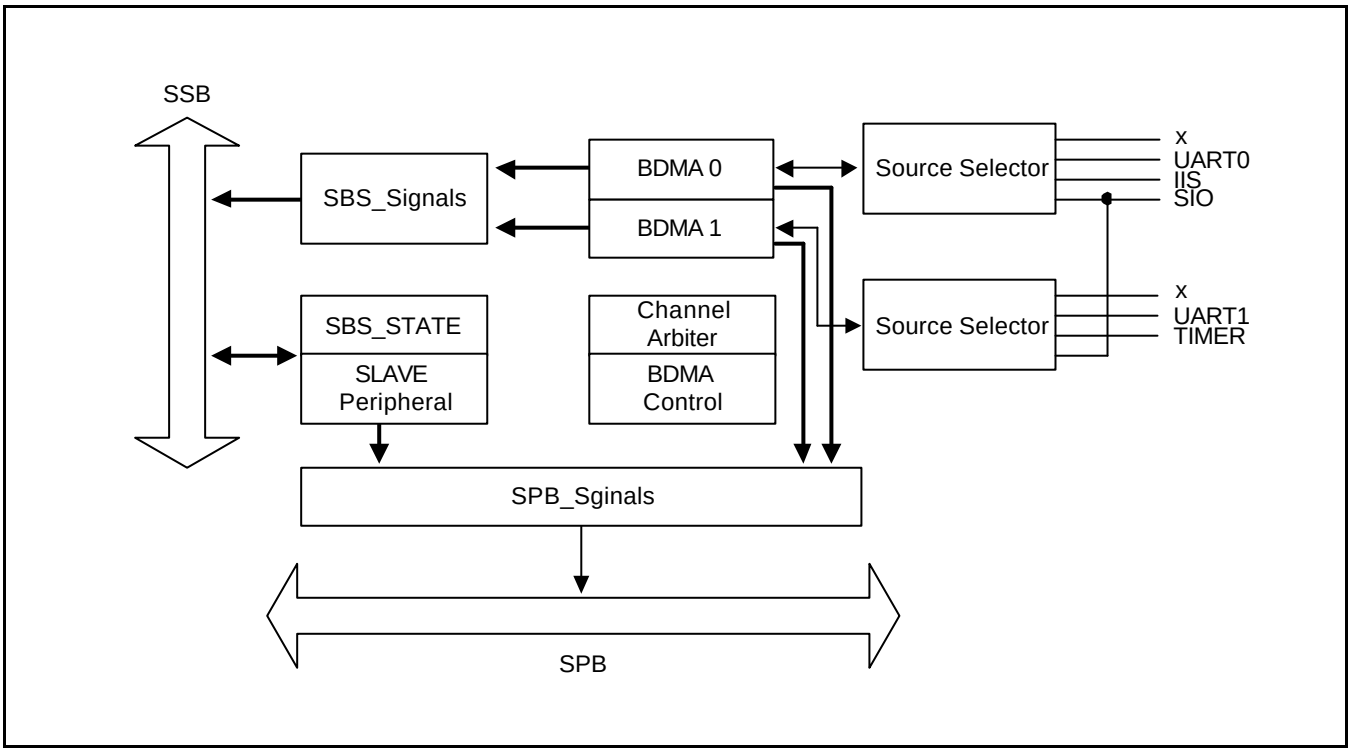


Figure 7-2. BDMA Controller Block Diagram

EXTERNAL DMA REQ/ACK 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. Because ZDMA and BDMA can support external triggering, these protocols correspond to ZMDA only, not BDMA.

Handshake Mode

In the handshake mode, the DMA can generate a single DMA acknowledge corresponding to the single DMA request. The Figure 7-3 shows the handshake mode of DMA operation. In this figure, the DMA service means a paired or an inseparable Read and Write cycle during DMA operation, which is one DMA operation. During one DMA operation (Pared or inseparable Read and Write cycle), the bus controller does not allocate bus usage right to other bus masters. If the user wants to allocate the bus usage properly for the higher priority master during one DMA operation, the user should use the single step mode, which is explained in the next page. The single step mode considers one DMA operation to consist of separable Read and Write cycle. It means that the bus controller can allocate the bus usage to other higher bus master between Read and Write cycle.

The DMA request by nXDREQ causes one byte, one half word, or one word to be transmitted. The handshake mode requires the DMA request for every data transfer. The nXDREQ can be released after active nXDACK and request again after inactive nXDACK as shown in Figure 7-3.

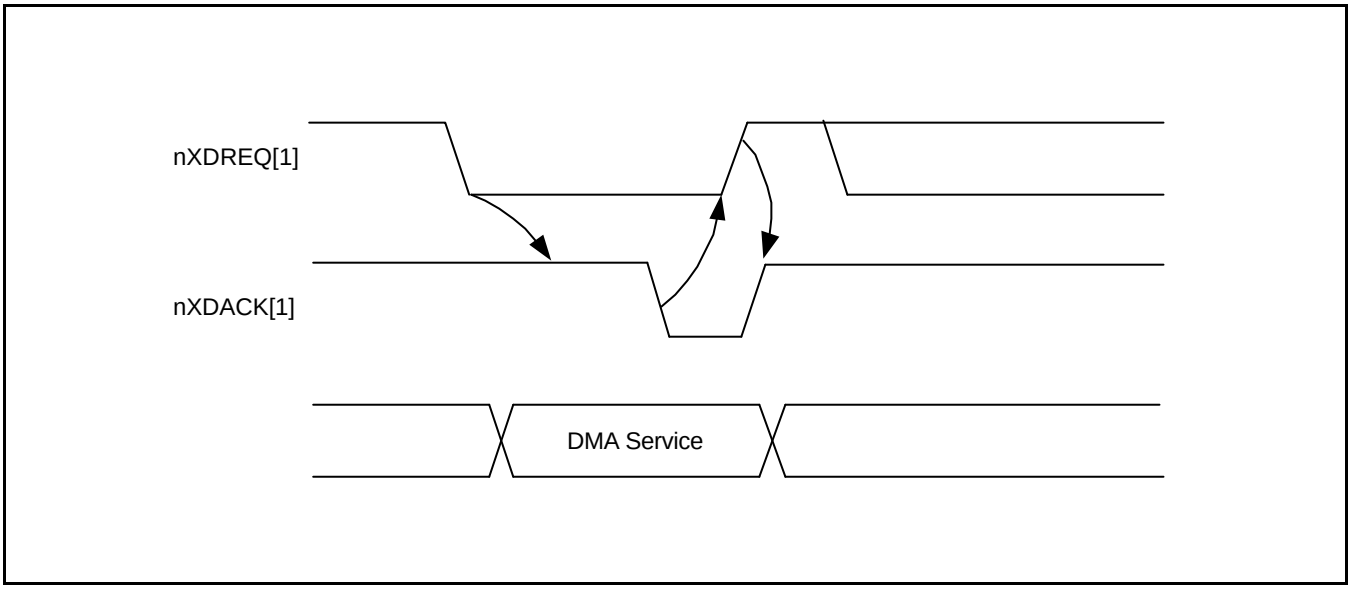


Figure 7-3. Handshake Mode Diagram

Single Step Mode

The single step mode means that there are two DMA acknowledge cycles indicating DMA read and write cycle. The single step mode is usually used for test or debugging because the bus mastership can be handed over to other bus master between Read and Write. During the inactive period of nXDACK, i.e., between Read and Write cycle, the bus controller re-evaluates the bus priority to determine the new bus mastership. Therefore, data transfer slower than that of the hand shake mode is expected.

When the DMA request signal goes low, the bus controller indicates the bus allocation for the DMA operation by lowering the DMA acknowledge signal if there is no higher priority bus request. During the first low level period of the DMA acknowledge signal, there will be a DMA read cycle. After the DMA read cycle, there will be a rising of the DMA acknowledge signal to indicate the end of the DMA read cycle. Simultaneously, the next DMA write cycle initiates if the DMA request signal is still low at the rising edge of DMA acknowledge. But, if the DMA request signal is already high at the rising edge of DMA acknowledge, the next DMA write cycle will be delayed until a new DMA request signal is activated. These two cases are shown in below Figure 7-4 and Figure 7-5.

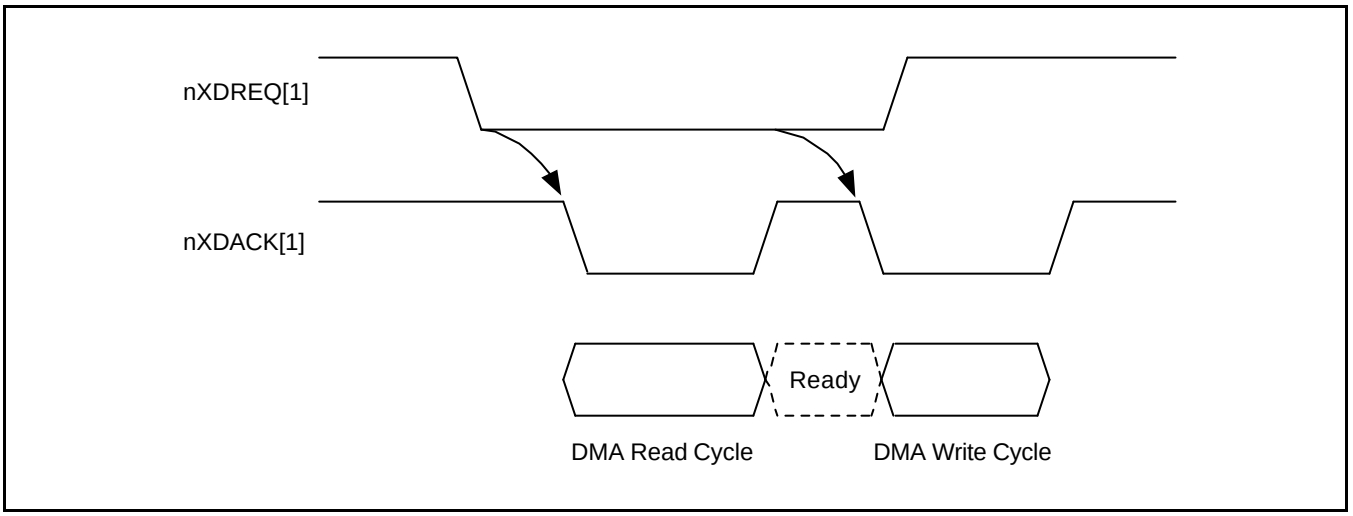


Figure 7-4. Single Step Mode (Case 1)

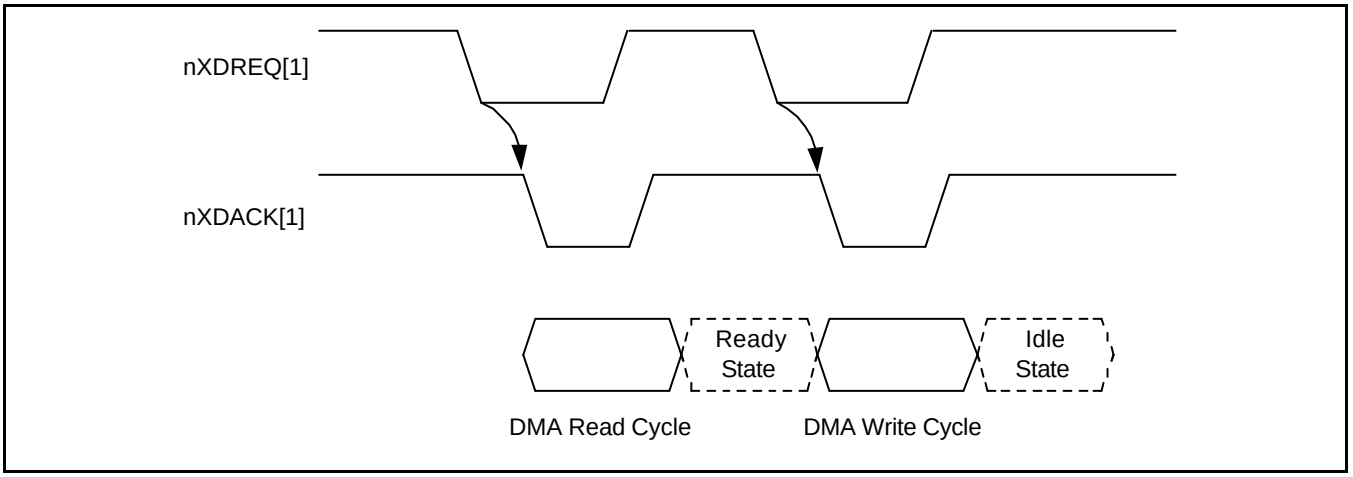


Figure 7-5. Single Step Mode (Case 2)

Whole Service Mode

The whole service mode means that the specified number of DMA operations, i.e., number of DMA operations based on transfer count, will be initiated by a single activation of DMA Request, and will be proceeded without further activations of DMA requests. The figure below shows how the whole service mode proceeds. The nXDACK signal will be active until the end of the whole DMA operations.

If the number of DMA transfer operation is too large, the long bus occupation during the whole service mode of DMA operation may cause problem because the other bus services will not be provided. To solve this kind of problem, the DMA releases the bus mastership in the whole service mode every time one unit (1byte, or 1 half-word, or 1 word) is transferred. When the DMA releases the bus mastership, the other bus masters, such as the CPU, the other DMA, and the external bus master, may have bus mastership. This feature in the whole service mode can provide the optimal bus sharing, preventing the monopoly of bus mastership by DMA. If the other master intercepts the bus mastership as shown in Figure 7-7, the remainder of DMA operation can be executed after servicing the impinged bus mastership, without the re-activation of nXDREQ.

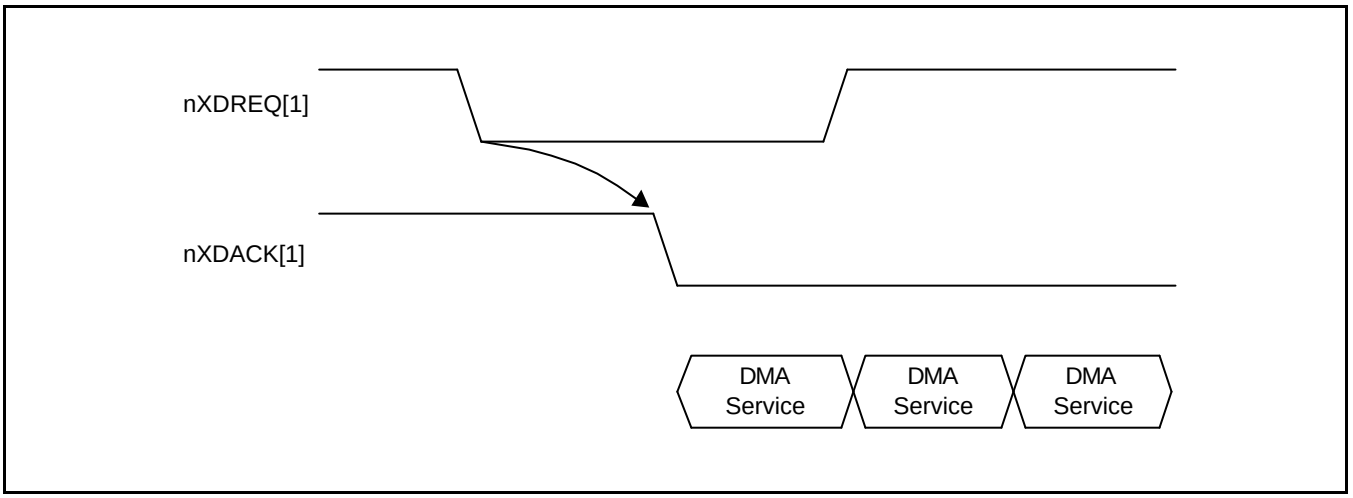


Figure 7-6. Whole Service Mode

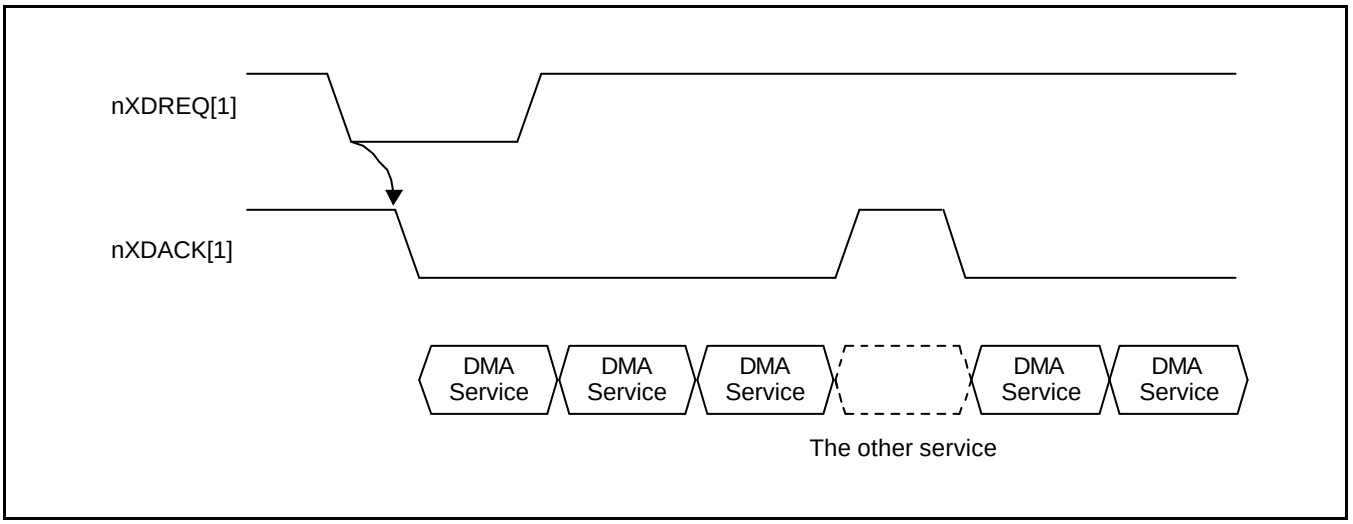


Figure 7-7. Whole Service Mode When Another Bus Master Acquires Bus Mastership

Demand Mode

Demand mode implies continuous DMA transfer cycles as long as DMA request signal is activated, as shown in figure 7-8.

Unlike the whole service mode, this mode does not permit the bus hand-over bus mastership to higher priority bus master, which make this request to bus controller during DMA operations. In other words, no other bus master can have bus mastership during the demand mode.

The sole monopoly of the bus mastership in demand mode prevents the demand mode from exceeding the specified maximum time, such as the DRAM refresh period.

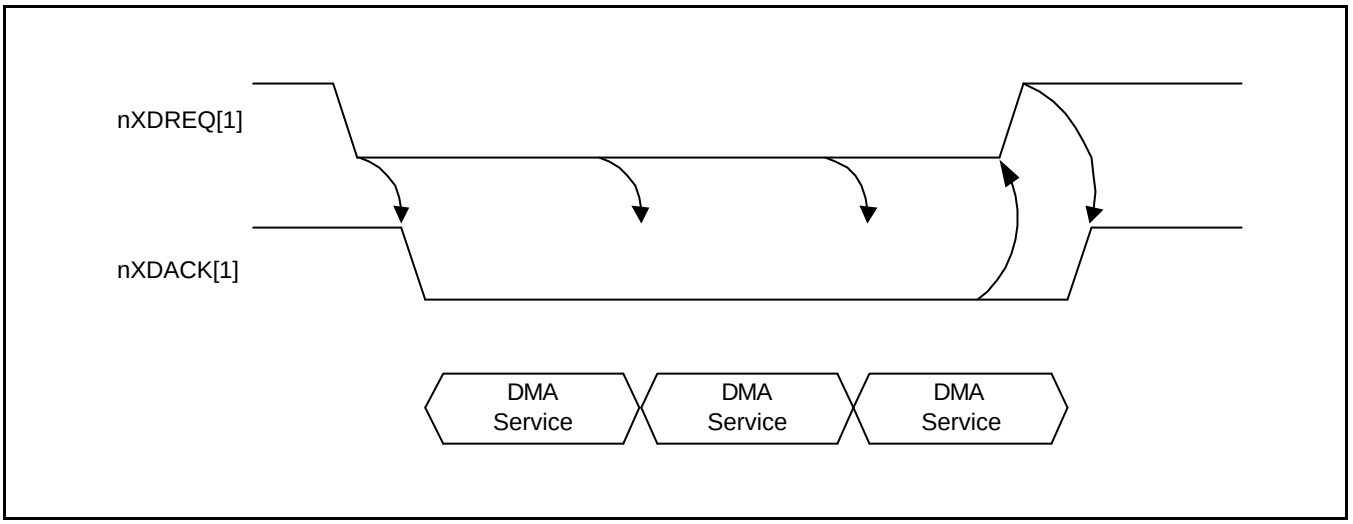


Figure 7-8. Demand Mode

NOTE

The bus controller does not permit the hand-over of bus mastership during the DMA operation using the demand mode. In other words, the DMA monopolizes bus usage right up to the completion of DMA operation. Care is warranted when using the DMA operation in the demand mode because this kind of monopoly may cause an un-expected malfunction on other masters by blocking optimal bus sharing.

DMA TRANSFER MODE

There are three types of DMA transfer modes (Unit transfer mode, Block transfer mode and On the fly transfer mode). Different from the external DMA request/acknowledge protocol, the DMA transfer mode defines the number of reads/writes per unit transfer as shown in the following table.

DMA Transfer Mode	Read/Write
Unit transfer	1 unit read, then 1 unit write
Block transfer	4 unit burst read, then 4 unit burst write
On-the-fly transfer	1 unit read or 1 unit write exclusively

Unit Transfer Mode

The unit transfer mode means that the paired DMA read/write cycle happens corresponding each DMA request as shown below in Figure. Figure 7-9 shows the example case of the unit transfer mode at the handshake mode.

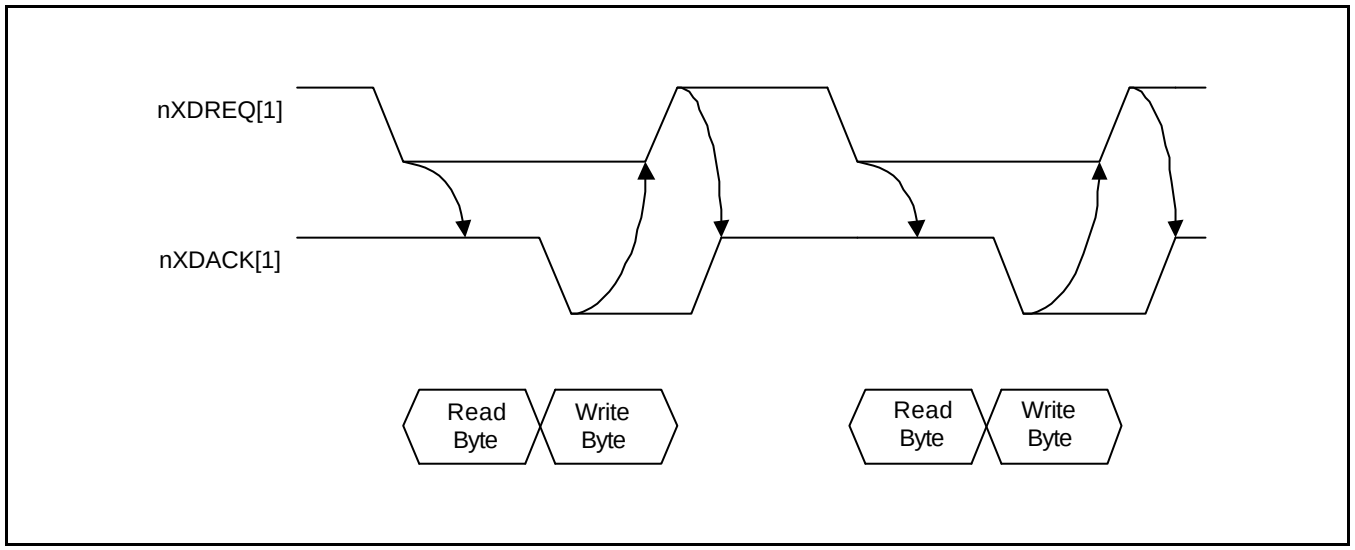


Figure 7-9. Unitary Transfer Mode with Handshake mode

Block (4-word) Transfer Mode

The block (4-word) transfer mode means that the successive 4-word DMA read cycle happens before the successive 4-word DMA write cycle, as shown in Figure 7-10. Figure 7-10 shows an example of the block transfer mode with single step mode.

If the block transfer mode is used, the total data size to be transferred should be a multiple of 16 bytes. In other words, the minimum transfer size is 16 bytes, i.e., 4 words. Because the DMA count is defined in byte unit, 16 should be the DMA transfer count in the case of 4 words transfer. If the transfer size or DMA count is not a multiple of 16, for example 16, 32, 48, 64, and so on, the DMA can not transfer the data completely. If assume 100 bytes-transfer (DMA count is 100), 6x16 = 96 bytes can be transferred. But, the remaining 4 bytes can not be transferred because DMA operation will be stopped after 96 bytes transfer. The users should be aware of this characteristics when they select the block transfer mode of DMA.

NOTE: The ADDR[3:0] should be '0' to meet 16-byte align condition in Block Transfer Mode.

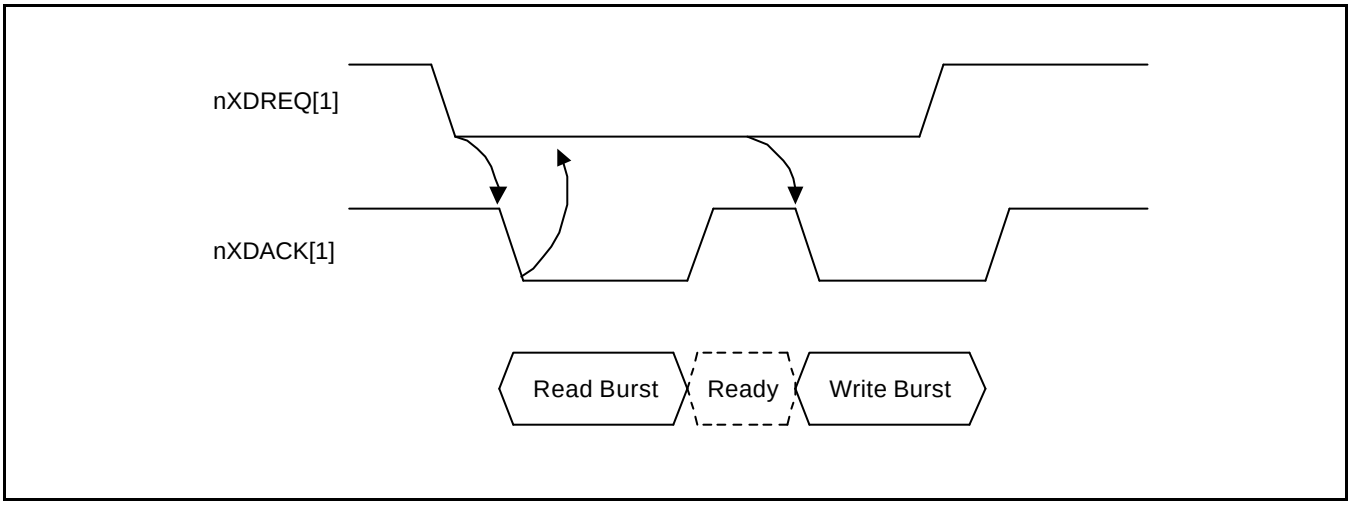


Figure 7-10. Block Transfer Mode With Single Step Mode

On-the-fly Transfer Mode

The on-the-fly transfer mode means that when DMA reads/writes data, a fixed addressed external device writes/reads the data by DMA acknowledge signals (nXDACK0/1). In the other modes, the DMA reads data before writing the data.

In on-the-fly transfer mode, the read and write operation occur simultaneously. The DMA acknowledge signal notifies the external device to read or write. Simultaneously, the memory controller should generate Read-related or Write-related control signals to the external memory. If the external device can support the on-the-fly mode (can read/write the data by DMA acknowledge), the data transfer rates will be doubled. During the on-the-fly transfer cycle, S3C44B0X data bus will be in Hi-z state. Figure 7-11 shows the example of the on-the-fly transfer mode with the whole service mode.

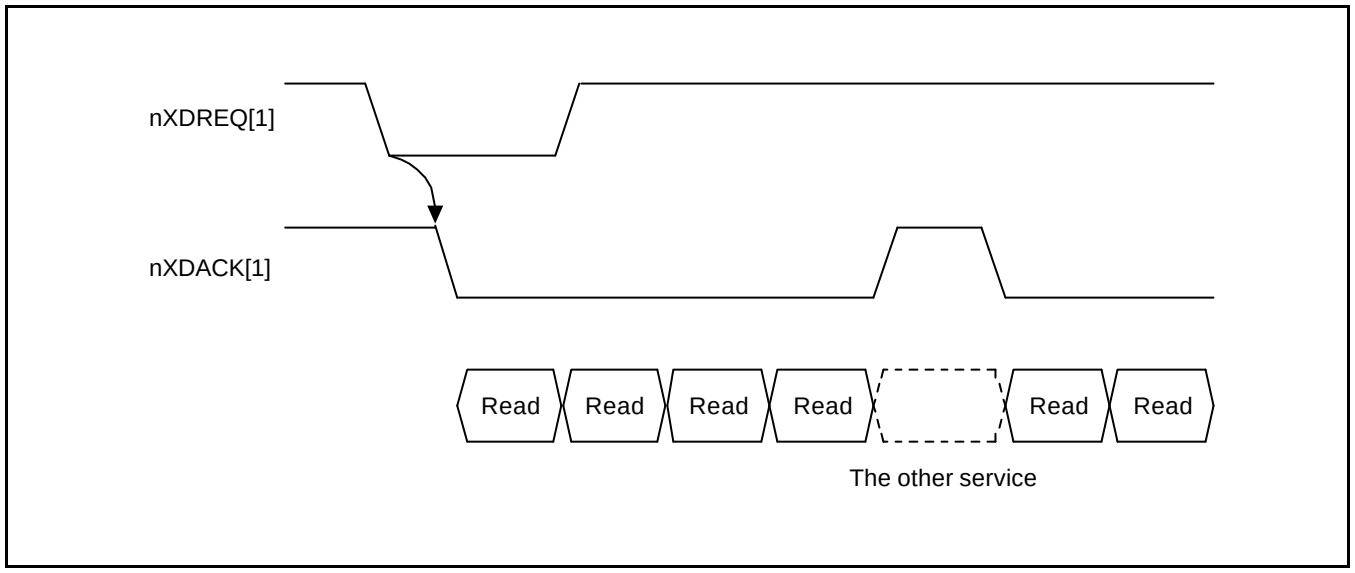


Figure 7-11. On-the-fly Transfer Mode with Whole Transfer Mode

DMA REQUEST SOURCE SELECTION

In ZDMA, S/W or H/W produces the nXDREQ (external DMA request signal), which is the DMA request source. The S/W trigger can be done by writing the CMD field as 01 in ZDCON0/1 register, i.e., the start of DMA. Before the start of DMA, the DMA-related parameters, such as source address, destination address, transfer count and so on, should be configured. Based-on these configuration, the DMA operation will start when the CMD field is written as 01. In S/W trigger, the DMA operations will continue as long as the burst mastership is allocated to the DMA master and as long as the DMA transfer count or TC(Terminal Count) reaches zero, i.e., the completion of DMA operation. If the higher bus master acquires the bus mastership, DMA operations will continue after the service of higher priority bus master. The DMA operations can also be initiated by nXDREQ(External DMA request signal) as well as S/W if the DMA is configured for the external trigger mode, i.e., enable External DMA request by writing QDS bit as 1 in the ZDCON0/1 register.
In BDMA, there are six hardware request sources, UART0, UART1, SIO , Timer and IIS. The BDMA can be initiated by software as the ZDMA. These sources can be selected by writing the QSC field in the BDICNT register.

AUTO RELOAD MODE

In the auto reload mode, the register content of Z(B)DCSRCn, Z(B)DCDSTn, and Z(B)DCCNTn are reloaded from the registers of Z(B)DISRCn, Z(B)DIDESn, and Z(B)DICNTn when the DMA count decreases to 0. The configuration parameters relating to DMA operation are contained in the registers of Z(B)DISRCn, Z(B)DIDESn, and Z(B)DICNTn, for example, soure/destination address and source/destination transfer count. This kind of Auto-reloading can preschedule DMA operation automatically. In other words, to change the configuration, the configuration in the registers of Z(B)DISRCn, Z(B)DIDESn, and Z(B)DICNTn should be changed before the end of DMA operation based-on current configuration. But, this kind of parameter auto-reloading can not guarantee the DMA re-run automatically after the current DMA operation. The DMA will re-run if Z(B)DCONn CMD field is written newly or external DMA request is issued.

To support the Auto-reload mdoe, the DMA should have two registers sets. The registers, Z(B)DISRCn, Z(B)DIDESn, and Z(B)DICNTn, have the initial configuration for DMA operation as above-mentioned and registers, Z(B)DCSRCn, Z(B)DCDESn and Z(B)DCCNTn, have the configuraion reflecting the current DMA operation. For example, these register should have dynamic values of source address, destination address, and the remained transfer count or TC(Terminal Count) during DMA operation.

The register contents of Z(B)DISRCn, Z(B)DIDESn, and Z(B)DICNTn can be reloaded into the registers Z(B)DCSRCn, Z(B)DCDESn and Z(B)DCCNTn under one of the four cases.

case 1) Auto Reload(AR) is equal to 1 and DMA Count reaches to 0, which are normal auto-reload mode of DMA operation.

case 2) Writes new configuration into the Z(B)DISRC0, Z(B)DIDES0, and Z(B)DICNT0. If DMA is in Auto-reload mode, these new contests of the register will be re-loaded automatically as same as above case. If DMA is not active, these new configuration will be written into registers, Z(B)DISRC0, Z(B)DIDES0, and Z(B)DICNT0, immediately

case 3) When DMA is enable, i.e., EN bit in Z(B)DICNT register changes from 0 to 1. The register contents of Z(B)DISRC0, Z(B)DIDES0, and Z(B)DICNT0 will be loaded into the registers of Z(B)DCSRC0, Z(B)DCDES0, and Z(B)DCCNT0 immediately to start of DMA operation, regardless of whether the DMA is in Auto-reload mode, or not.

case 4) S/W command is Cancel. When user writes the CMD field as 11 in the register of ZDCON0/1. In this case, the register content of Z(B)DISRC0, Z(B)DIDES0, and Z(B)DICNT0 will be loaded into the registers, Z(B)DCSRC0, Z(B)DCDES0, and Z(B)DCCNT0 immediately.

DMA SPECIAL REGISTERS

ZDMA CONTROL REGISTER (ZDCONn)

Register	Address	R/W	Description	Reset Value
ZDCON0	0x01E80000	R/W	ZDMA 0 Control Register	0x00
ZDCON1	0x01E80020	R/W	ZDMA 1 Control Register	0x00

ZDCONn	Bit	Description	Initial State
INT	[7:6]	Reserved	00
STE	[5:4]	Status of DMA channel (Read only) 00 = Ready 01 = Not TC yet 10 = Terminal Count 11 = N/A Before the DMA counter decreases from the initial counter value, STE is still in the ready state.	00
QDS	[3:2]	Disable/Enable External DMA request (nXDREQ) 00 = Enable other = Disable	00
CMD	[1:0]	Software commands 00: No command. After writing 01,10,11, CMD bit is cleared automatically. nXDREQ is available. 01: Starts DMA operation by S/W without nXDREQ. S/W start function can be used only in the whole mode. As DMA is in the whole mode, the DMA will operate until the counter is 0. If nXDREQ is used , this command must not be issued. 10: Pauses DMA operation. But nXDREQ is still available. 11: Cancels DMA operation.	00

NOTE: If users start the ZDMA operation by CMD=01b, the DREQ protocol must be whole service mode.

ZDMA0 INITIAL SOURCE/DESTINATION ADDRESS AND COUNT REGISTERS (ZDISRC0, ZDIDES0, ZDICNT0)

Register	Address	R/W	Description	Reset Value
ZDISRC0	0x01E80004	R/W	ZDMA 0 initial source address Register	0x00000000
ZDIDES0	0x01E80008	R/W	ZDMA 0 initial destination address Register	0x00000000
ZDICNT0	0x01E8000C	R/W	ZDMA 0 initial count register	0x00000000

ZDMA0 CURRENT SRC/DST ADDRESS AND COUNT REGISTERS (ZDCSRC0, ZDCDES0, ZDCCNT0)

Register	Address	R/W	Description	Reset Value
ZDCSRC0	0x01E80010	R	ZDMA 0 current source address Register	0x00000000
ZDCDES0	0x01E80014	R	ZDMA 0 current destination address Register	0x00000000
ZDCCNT0	0x01E80018	R	ZDMA 0 current count register	0x00000000

NOTE: These registers are read-only.

ZDMA1 INITIAL SOURCE/DESTINATION ADDRESS AND COUNT REGISTERS (ZDISRC1, ZDIDES1, ZDICNT1)

Register	Address	R/W	Description	Reset Value
ZDISRC1	0x01E80024	R/W	ZDMA 1 initial source address Register	0x00000000
ZDIDES1	0x01E80028	R/W	ZDMA 1 initial destination address Register	0x00000000
ZDICNT1	0x01E8002C	R/W	ZDMA 1 initial count register	0x00000000

ZDMA1 CURRENT SRC/DST ADDRESS AND COUNT REGISTERS (ZDCSRC1, ZDCDES1, ZDCCNT1)

Register	Address	R/W	Description	Reset Value
ZDCSRC1	0x01E80030	R	ZDMA 1 current source address Register	0x00000000
ZDCDES1	0x01E80034	R	ZDMA 1 current destination address Register	0x00000000
ZDCCNT1	0x01E80038	R	ZDMA 1 current count register	0x00000000

NOTE: These registers are read-only.

ZDMAn INITIAL/CURRENT SOURCE ADDRESS REGISTERS (ZDISRC, ZDCSRC)

ZDISRCn/ZDCSRCn	Bit	Description	Initial State
DST	[31:30]	Data size for transfer 00 = Byte, 01 = Half word 10 = Word, 11 = Not used If the block transfer mode is used, the DST must be 10.	00
DAL	[29:28]	Direction of address for load 00 = N/A, 01 = Increment 10 = Decrement, 11 = Fixed	00
ISADDR/CSADDR	[27:0]	Initial/current source address for ZDMAn	0x0000000

ZDMAn INITIAL/CURRENT DESTINATION ADDRESS REGISTERS (ZDIDES, ZDCDE)

ZDIDESn/ZDCDESn	Bit	Description	Initial State
OPT	[31:30]	DMA internal options. OPT = 10 is recommended. bit 31: Indicates how nXDREQ is sampled in the single step mode. 1 is recommended. bit 30: If the DST is half-word or word and if the DMA mode is not the block transfer mode, this bit takes a role. 1: DMA does word-swap or half-word swap Before transfer: B0,B1,B2,B3,B4,B5,B6,B7... word-swapped data: B3,B2,B1,B0,B7,B6,B5,B4,... half-word-swapped data: B1,B0,B3,B2,B5,B4,B7,B6,... 0: normal	00
DAS	[29:28]	Direction of address for store 00 = N/A 01 = Increment 10 = Decrement 11 = Fixed	00
IDADDR/CDADDR	[27:0]	Initial/current destination address for ZDMAn	0x0000000

ZDMAn INITIAL/CURRENT COUNT REGISTERS (ZDICNT, ZDCCNT)

ZDICNTn/ZDCCNTn	Bit	Description	Initial State
QSC	[31:30]	DREQ(DMA request) source selection 00 = nXDREQ[0] 01 = nXDREQ[1] 10 = N/A 11 = N/A	00
QTY	[29:28]	DREQ protocol 00 = Handshake 01 = Single step 10 = Whole Service 11 = Demand	00
TMD	[27:26]	Transfer mode 00 = Not used 01 = Unit transfer mode 10 = Block(4-word) transfer mode 11 = On the fly If block transfer mode is selected, the ADDR[3:0] should be '0' to meet 16-byte align condition.	00
OTF	[25:24]	On the fly mode 00 = N/A 01 = N/A 10 = Read time on the fly 11 = Write time on the fly	00
INTS	[23:22]	Interrupt mode set 00 = Polling mode 01 = N/A 10 = Int. whenever transferred 11 = Int. whenever terminated count	00
AR	[21]	Auto-reload and Auto-start after DMA count are 0. 0 = Disable 1 = Enable. Even after DMA count is 0, the DMA H/W enable bit (EN bit) is still 1. But, DMA will start to operate only if the start command or nXDREQ is activated.	0
EN	[20]	DMA H/W enable/disable 0 = Disable DMA 1 = Enable DMA. If the QDS bit is 00b, DMA request can be serviced. Also if the S/W command is started, the DMA operation will occur. If the EN bit is 0, DMA will not operate even though S/W command is started. If the S/W command is canceled, the DMA operation will be canceled and EN bit will be cleared to 0. At the terminal count, the EN bit will be cleared to 0. **NOTE:** Do not set the EN bit and the other bits of ZDICNT register at the same time. User have to set EN bit after setting the other bits of ZDICNT register as following steps, 1. Set ZDICNT register with disabled En bit. 2. Set EN bit enable.	0
ICNT/CCNT	[19:0]	Initial/current transfer count for ZDMAn. If 1 byte is transferred, the ICNT will be decreased by 1. If 1 half-word is transferred, the ICNT will be decreased by 2. If 1 word is transferred, the ICNT will be decreased by 4. For example, if the data size of a transfer is word and the count is 4n+3, the last 3 bytes will not be transferred.	0x00000

BDMAn CONTROL REGISTER (BDCON)

Register	Address	R/W	Description	Reset Value
BDCON0	0x01F80000	R/W	Bridge DMA 0 Control Register	0x00
BDCON1	0x01F80020	R/W	Bridge DMA 1 Control Register	0x00

BDCONn	Bit	Description	Initial State
INT	[7:6]	Reserved	00
STE	[5:4]	Status of DMA channel (Read only) 00 = Ready 01 = Not TC yet 10 = Terminal Count 11 = N/A Before the DMA counter decreases from a initial counter value, STE is still the ready state.	00
QDS	[3:2]	Disable/Enable External/Internal DMA request (UARTn, SIO, IIS, Timer) 00 = Enable Other = Disable	00
CMD	[1:0]	Software commands 00: No command. After writing 01, 10, 11, CMD bits are cleared automatically. 01: Reserved 10: Reserved 11: Cancels DMA operation.	00

BDMA0 INITIAL SRC/DST ADDRESS AND COUNT REGISTERS (BDISRC0, BDIDES0, BDICNT0)

Register	Address	R/W	Description	Reset Value
BDISRC0	0x01F80004	R/W	BDMA 0 initial source address Register	0x00000000
BDIDES0	0x01F80008	R/W	BDMA 0 initial destination address Register	0x00000000
BDICNT0	0x01F8000C	R/W	BDMA 0 initial count register	0x00000000

BDMA0 CURRENT SRC/DST ADDRESS AND COUNT REGISTERS (BDCSRC0, BDCDES0, BDCCNT0)

Register	Address	R/W	Description	Reset Value
BDCSRC0	0x01F80010	R	BDMA 0 current source address Register	0x00000000
BDCDES0	0x01F80014	R	BDMA 0 current destination address Register	0x00000000
BDCCNT0	0x01F80018	R	BDMA 0 current count register	0x00000000

NOTE: These registers are read-only.

BDMA1 INITIAL SRC/DST ADDRESS AND COUNT REGISTERS (BDISRC1, BDIDES1, BDICNT1)

Register	Address	R/W	Description	Reset Value
BDISRC1	0x01F80024	R/W	BDMA 1 initial source address Register	0x00000000
BDIDES1	0x01F80028	R/W	BDMA 1 initial destination address Register	0x00000000
BDICNT1	0x01F8002C	R/W	BDMA 1 initial count register	0x00000000

BDMA1 CURRENT SRC/DST ADDRESS AND COUNT REGISTERS (BDCSRC1, BDCDES1, BDCCNT1)

Register	Address	R/W	Description	Reset Value
BDCSRC1	0x01F80030	R	BDMA 1 current source address Register	0x00000000
BDCDES1	0x01F80034	R	BDMA 1 current destination address Register	0x00000000
BDCCNT1	0x01F80038	R	BDMA 1 current count register	0x00000000

NOTE: These registers are read-only.

BDMAn INITIAL/CURRENT SOURCE ADDRESS REGISTERS (BDISRC, BDCSRC)

BDISRCn/BDCSRCn	Bit	Description	Initial State
DST	[31:30]	Data size for transfer 00 = Byte 01 = Half word 10 = Word 11 = Not used	00
DAL	[29:28]	Direction of address for load 00 = N/A 01 = Increment 10 = Decrement 11 = Internal peripheral (fixed address)	00
ISADDR/CSADDR	[27:0]	Initial/current source address for BDMAn. If the destination is the internal peripherals, the SFR address has to be used. For example, if the source is the UART0 Rx buffer, the UART0 Rx buffer address will be used.	0x0000000

BDMAn INITIAL/CURRENT DESTINATION ADDRESS REGISTERS (BDIDES, BDCDES)

BDIDESn/BDCDESn	Bit	Description	Initial State
TDM	[31:30]	Transfer direction mode 00 = Reserved 01 = M2IO (from external memory to internal peripheral) 10 = IO2M (from internal peripheral to external memory) 11 = IO2IO (from internal peripheral to internal peripheral) **NOTE:** The initial value is '00' , but you must change TDM value as another though the BDMA channel is unused.	00
DAS	[29:28]	Direction of address for store 00 = N/A 01 = Increment 10 = Decrement 11 = Internal peripheral (fixed address)	00
IDADDR/CDADDR	[27:0]	Initial/current destination address for BDMAn If the destination is the internal peripherals, the SFR address has to be used. For example, if the destination is UART0 Tx buffer, the UART0 Tx buffer address will be used.	0x0000000

BDMA0 INITIAL/CURRENT COUNT REGISTERS (BDICNT0, BDCCNT0)

BDICNT0/BDCCNT0	Bit	Description	Initial State
QSC	[31:30]	DMA request source selection 00 = N/A 01 = IIS 10 = UART0 11 = SIO	00
Reserved	[29:28]	00: handshake mode	00
Reserved	[27:26]	01: unit transfer mode	01
Reserved	[25:24]	00: on-the-fly mode is not supported in BDMAn	00
INTS	[23:22]	Interrupt mode set 00 = Polling mode 01 = N/A 10 = Int. whenever transferred 11 = Int. whenever terminated count	00
AR	[21]	Auto-reload and Auto-start after DMA count are 0. 0= Disable 1= Enable. Even after DMA count is 0, the DMA H/W enable bit (EN bit) is still 1. But, DMA will start to operate only if the start command or DMA request is activated	0
EN	[20]	DMA H/W enable/disable 0 = Disable DMA 1 = Enable DMA. If the QDS bit is 00b, DMA request can be serviced. Also if the S/W command is started, the DMA operation will occur. If the EN bit is 0, DMA will not operate even though S/W command is started. If the S/W command is canceled, the DMA operation will be canceled and EN bit will be cleared to 0. At the terminal count, the EN bit will be cleared to 0. **NOTE:** Do not set the EN bit and the other bits of BDICNT register at the same time. User have to set EN bit after setting the other bits of BDICNT register as following steps, 1. Set BDICNT register with disabled En bit. 2. Set EN bit enable.	0
ICNT/CCNT	[19:0]	Transfer count for BDMA0. The transfer count must be right value. For example, if DST is word, ICNT must be 4n. If 1 byte is transferred, the ICNT will be decreased by 1. If 1 half-word is transferred, the ICNT will be decreased by 2. If 1 word is transferred, the ICNT will be decreased by 4.	0x00000

BDMA1 INITIAL/CURRENT COUNT REGISTERS (BDICNT1, BDCCNT1)

BDICNT1/BDCCNT1	Bit	Description	Initial State
QSC	[31:30]	DMA request source selection 00 = N/A 01 = Timer 10 = UART1 11 = SIO	00
Reserved	[29:28]	00: handshake mode	00
Reserved	[27:26]	01: unit transfer mode	01
Reserved	[25:24]	00: on-the-fly mode is not supported in BDMAn	00
INTS	[23:22]	Interrupt mode set 00 = Polling mode 01 = N/A 10 = Int. whenever transferred 11 = Int. whenever terminated count	00
AR	[21]	Auto-reload and Auto-start after DMA count are 0. 0= Disable 1= Enable. Even after DMA count is 0, the DMA H/W enable bit (EN bit) is still 1. But, DMA will start to operate only if the start command or DMA request is activated	0
EN	[20]	DMA H/W enable/disable 0 = Disable DMA 1 = Enable DMA. If the QDS bit is 00b, DMA request can be serviced. Also if the S/W command is started, the DMA operation will occur. If the EN bit is 0, DMA will not operate even though S/W command is started. If the S/W command is canceled, the DMA operation will be canceled and EN bit will be cleared to 0. At the terminal count, the EN bit will be cleared to 0. **NOTE:** Do not set the EN bit and the other bits of BDICNT register at the same time. User have to set EN bit after setting the other bits of BDICNT register as following steps, 1. Set BDICNT register with disabled En bit. 2. Set EN bit enable.	0
ICNT/CCNT	[19:0]	Transfer count for BDMA1. The transfer count must be right value. For example, if DST is word, ICNT must be 4n. If 1 byte is transferred, the ICNT will be decreased by 1. If 1 half-word is transferred, the ICNT will be decreased by 2. If 1 word is transferred, the ICNT will be decreased by 4.	0x00000

8 I/O PORTS

OVERVIEW

S3C44B0X has 71 multi-functional input/output port pins. There are seven ports:

— Two 9-bit input/output ports. (Port E and F)

— Two 8-bit input/output ports. (Port D and G)

— One 16-bit input/output port. (Port C)

— One 10-bit output port. (Port A)

— One 11-bit output port. (Port B)

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

Before pin configurations, the initial pin states are configured elegantly to avoid some problems.

Table 8-1. S3C44B0X Port Configuration Overview

Port A	Selectable Pin functions	
	Function 1	Function 2
PA9	output only	ADDR24
PA8	output only	ADDR23
PA7	output only	ADDR22
PA6	output only	ADDR21
PA5	output only	ADDR20
PA4	output only	ADDR19
PA3	output only	ADDR18
PA2	output only	ADDR17
PA1	output only	ADDR16
PA0	output only	ADDR0

Port B	Selectable Pin functions	
	Function 1	Function 2
PB10	output only	nGCS5
PB9	output only	nGCS4
PB8	output only	nGCS3
PB7	output only	nGCS2
PB6	output only	nGCS1
PB5	output only	nWBE3:nBE3:DQM3
PB4	output only	nWBE2:nBE2:DQM2
PB3	output only	nSRAS:nCAS3
PB2	output only	nSCAS:nCAS2
PB1	output only	SCLK
PB0	output only	SCKE

Table 8-1. S3C44B0X Port Configuration Overview (Continued)

Port C	Selectable Pin functions		
	Function 1	Function 2	Function 3
PC15	Input/output	DATA31	nCTS0
PC14	Input/output	DATA30	nRTS0
PC13	Input/output	DATA29	RxD1
PC12	Input/output	DATA28	TxD1
PC11	Input/output	DATA27	nCTS1
PC10	Input/output	DATA26	nRTS1
PC9	Input/output	DATA25	nXDREQ1
PC8	Input/output	DATA24	nXDACK1
PC7	Input/output	DATA23	VD4
PC6	Input/output	DATA22	VD5
PC5	Input/output	DATA21	VD6
PC4	Input/output	DATA20	VD7
PC3	Input/output	DATA19	IISCLK
PC2	Input/output	DATA18	IISDI
PC1	Input/output	DATA17	IISDO
PC0	Input/output	DATA16	IISLRCK

Port D	Selectable Pin functions	
	Function 1	Function 2
PD7	Input/output	VFRAME
PD6	Input/output	VM
PD5	Input/output	VLINE
PD4	Input/output	VCLK
PD3	Input/output	VD3
PD2	Input/output	VD2
PD1	Input/output	VD1
PD0	Input/output	VD0

Table 8-1. S3C44B0X Port Configuration Overview (Continued)

Port E	Selectable Pin functions		
	Function 1	Function 2	Function 3
PE8	ENDIAN	CODECLK	input/output
PE7	Input/output	TOUT4	VD7
PE6	Input/output	TOUT3	VD6
PE5	Input/output	TOUT2	TCLK
PE4	Input/output	TOUT1	TCLK
PE3	Input/output	TOUT0	-
PE2	Input/output	RxD0	-
PE1	Input/output	TxD0	-
PE0	Input/output	Fpllo	Fout

Port F	Selectable Pin functions			
	Function 1	Function 2	Function 3	Function 4
PF8	input/output	nCTS1	SIOCK	IISCLK
PF7	input/output	RxD1	SIORxD	IISDI
PF6	input/output	TxD1	SIORDY	IISDO
PF5	input/output	nRTS1	SIOTxD	IISLRCK
PF4	input/output	nXBREQ	nXDREQ0	–
PF3	input/output	nXBACK	nXDACK0	–
PF2	input/output	nWAIT	–	–
PF1	input/output	IICSDA	–	–
PF0	input/output	IICSCL	–	–

Port G	Selectable Pin Functions		
	Function 1	Function 2	Function 3
PG7	input/output	IISLRCK	EINT7
PG6	input/output	IISDO	EINT6
PG5	input/output	IISDI	EINT5
PG4	input/output	IISCLK	EINT4
PG3	input/output	nRTS0	EINT3
PG2	input/output	nCTS0	EINT2
PG1	input/output	VD5	EINT1
PG0	input/output	VD4	EINT0

NOTES:

1. The underlined function name is selected just after a reset.(ENDIAN(PE8) is used only when nRESET is L.
2. IICSDA and IICSCL pins are open-drain pin. So, this pin needs pull-up resistors when used as output port(PF[1:0]).

PORT CONTROL DESCRIPTIONS

PORT CONFIGURATION REGISTER (PCONA-G)

In S3C44B0X, most pins are multiplexed pins. Therefore, the functions for each pin should be selected. The PCONn (port control register) determines which function is used for each pin.

If PG0 - PG7 are used for the wakeup signal in power down mode, these ports must be configured in interrupt mode.

PORT DATA REGISTER (PDATA-G)

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

PORT PULL-UP REGISTER (PUPC-G)

The port pull-up resistor controls the pull-up resistor enable/disable of each port group. When the corresponding bit is 0, the pull-up resistor of the pin is enabled. When 1, the pull-up resistor is disabled.

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 triggers for the external interrupt request

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

I/O PORT CONTROL REGISTER

PORT A CONTROL REGISTERS (PCONA, PDATA, PUPA)

Port A control registers are shown in Table 8-2:

Register	Address	R/W	Description	Reset Value
PCONA	0x01D20000	R/W	Configures the pins of port A	0x3ff
PDATA	0x01D20004	R/W	The data register for port A	Undef.

Table 8-2. Port of Group A Control Registers (PCONA,PDATA)

PCONA	Bit	Description	
PA9	[9]	0 = Output	1 = ADDR24
PA8	[8]	0 = Output	1 = ADDR23
PA7	[7]	0 = Output	1 = ADDR22
PA6	[6]	0 = Output	1 = ADDR21
PA5	[5]	0 = Output	1 = ADDR20
PA4	[4]	0 = Output	1 = ADDR19
PA3	[3]	0 = Output	1 = ADDR18
PA2	[2]	0 = Output	1 = ADDR17
PA1	[1]	0 = Output	1 = ADDR16
PA0	[0]	0 = Output	1 = ADDR0

PDATA	Bit	Description
PA[9:0]	[9:0]	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, an undefined value will be read.

PORT B CONTROL REGISTERS (PCONB, PDATB)

Port B control registers are shown in Table 8-3:

Register	Address	R/W	Description	Reset Value
PCONB	0x01D20008	R/W	Configures the pins of port B	0x7ff
PDATB	0x01D2000C	R/W	The data register for port B	Undef.

Table 8-3. Port of Group B Control Registers (PCONB,PDATB)

PCONB	Bit	Description	
PB10	[10]	0 = Output	1 = nGCS5
PB9	[9]	0 = Output	1 = nGCS4
PB8	[8]	0 = Output	1 = nGCS3
PB7	[7]	0 = Output	1 = nGCS2
PB6	[6]	0 = Output	1 = nGCS1
PB5	[5]	0 = Output	1 = nWBE3/nBE3/DQM3
PB4	[4]	0 = Output	1 = nWBE2/nBE2/DQM2
PB3	[3]	0 = Output	1 = nSRAS/nCAS3
PB2	[2]	0 = Output	1 = nSCAS/nCAS2
PB1	[1]	0 = Output	1 = SCLK
PB0	[0]	0 = Output	1 = SCKE

PDATB	Bit	Description
PB[10:0]	[10:0]	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, an undefined value will be read.

PORT C CONTROL REGISTERS (PCONC, PDATC, PUPC)

Port C control registers are shown in Table 8-4:

Register	Address	R/W	Description	Reset Value
PCONC	0x01D20010	R/W	Configures the pins of port C	0xaaaaaaaa
PDATC	0x01D20014	R/W	The data register for port C	Undef.
PUPC	0x01D20018	R/W	pull-up disable register for port C	0x0

Table 8-4. Port of Group C Control Registers (PCONC,PDATC,PUPC)

PCONC	Bit	Description
PC15	[31:30]	00 = Input 01 = Output 10 = DATA31 11 = nCTS0
PC14	[29:28]	00 = Input 01 = Output 10 = DATA30 11 = nRTS0
PC13	[27:26]	00 = Input 01 = Output 10 = DATA29 11 = RxD1
PC12	[25:24]	00 = Input 01 = Output 10 = DATA28 11 = TxD1
PC11	[23:22]	00 = Input 01 = Output 10 = DATA27 11 = nCTS1
PC10	[21:20]	00 = Input 01 = Output 10 = DATA26 11 = nRTS1
PC9	[19:18]	00 = Input 01 = Output 10 = DATA25 11 = nXDREQ1
PC8	[17:16]	00 = Input 01 = Output 10 = DATA24 11 = nXDACK1
PC7	[15:14]	00 = Input 01 = Output 10 = DATA23 11 = VD4
PC6	[13:12]	00 = Input 01 = Output 10 = DATA22 11 = VD5
PC5	[11:10]	00 = Input 01 = Output 10 = DATA21 11 = VD6
PC4	[9:8]	00 = Input 01 = Output 10 = DATA20 11 = VD7
PC3	[7:6]	00 = Input 01 = Output 10 = DATA19 11 = IISCLK
PC2	[5:4]	00 = Input 01 = Output 10 = DATA18 11 = IISDI
PC1	[3:2]	00 = Input 01 = Output 10 = DATA17 11 = IISDO
PC0	[1:0]	00 = Input 01 = Output 10 = DATA16 11 = IISLRCK

PDATC	Bit	Description
PC[15:0]	[15:0]	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, an undefined value will be read.

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

PORT D CONTROL REGISTERS (PCOND, PDATD, PUPD)

Port D control registers are shown in Table 8-5.

Register	Address	R/W	Description	Reset Value
PCOND	0x01D2001C	R/W	Configures the pins of port D	0x0000
PDATD	0x01D20020	R/W	The data register for port D	Undef.
PUPD	0x01D20024	R/W	Pull-up disable register for port D	0x0

Table 8-5. Port of Group D Control Registers (PCOND, PDATD, PUPD)

PCOND	Bit	Description
PD7	[15:14]	00 = Input 01 = Output 10 = VFRAME 11 = Reserved
PD6	[13:12]	00 = Input 01 = Output 10 = VM 11 = Reserved
PD5	[11:10]	00 = Input 01 = Output 10 = VLINE 11 = Reserved
PD4	[9:8]	00 = Input 01 = Output 10 = VCLK 11 = Reserved
PD3	[7:6]	00 = Input 01 = Output 10 = VD3 11 = Reserved
PD2	[5:4]	00 = Input 01 = Output 10 = VD2 11 = Reserved
PD1	[3:2]	00 = Input 01 = Output 10 = VD1 11 = Reserved
PD0	[1:0]	00 = Input 01 = Output 10 = VD0 11 = Reserved

PDATD	Bit	Description
PD[7:0]	[7:0]	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, an undefined value will be read.

PUPD	Bit	Description
PD[7:0]	[7:0]	0: the pull up resistor attached to the corresponding port pin is enabled. 1: the pull up resistor is disabled.

PORT E CONTROL REGISTERS (PCONE, PDATE)

Port E control registers are shown in Table 8-6:

Register	Address	R/W	Description	Reset Value
PCONE	0x01D20028	R/W	Configures the pins of port E	0x00
PDATE	0x01D2002C	R/W	The data register for port E	Undef.
PUPE	0x01D20030	R/W	pull-up disable register for port E	0x00

Table 8-6. Port of Group E Control Registers (PCONE, PDATE)

PCONE	Bit	Description
PE8	[17:16]	00 = Reserved(ENDIAN) 01 = Output 10 = CODECLK 11 = Reserved PE8 can be used as ENDIAN only during the reset cycle.
PE7	[15:14]	00 = Input 01 = Output 10 = TOUT4 11 = VD7
PE6	[13:12]	00 = Input 01 = Output 10 = TOUT3 11 = VD6
PE5	[11:10]	00 = Input 01 = Output 10 = TOUT2 11 = TCLK in
PE4	[9:8]	00 = Input 01 = Output 10 = TOUT1 11 = TCLK in
PE3	[7:6]	00 = Input 01 = Output 10 = TOUT0 11 = Reserved
PE2	[5:4]	00 = Input 01 = Output 10 = RxD0 11 = Reserved
PE1	[3:2]	00 = Input 01 = Output 10 = TxD0 11 = Reserved
PE0	[1:0]	00 = Input 01 = Output 10 = Fpllo out 11 = Fout out

NOTE: Please refer to Fig. 5-1 when selecting Fpllo or Fout.

PDATE	Bit	Description
PE[8:0]	[8:0]	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.

PUPE	Bit	Description
PE[7:0]	[7:0]	0: the pull up resistor attached to the corresponding port pin is enabled. 1: the pull up resistor is disabled. PE8 do not have programmable pull-up resistor.

PORT F CONTROL REGISTERS (PCONF, PDATF, PUPF)

Port F control registers are shown in Table 8-7 below:

Register	Address	R/W	Description	Reset Value
PCONF	0x01D20034	R/W	Configures the pins of port F	0x0000
PDATF	0x01D20038	R/W	The data register for port F	Undef.
PUPF	0x01D2003C	R/W	pull-up disable register for port F	0x000

Table 8-7. Port of Group F Control Registers (PCONF, PDATF, PUPF)

PCONF	Bit	Description
PF8	[21:19]	000 = Input 001 = Output 010 = nCTS1 011 = SIOCLK 100 = IISCLK Others = Reserved
PF7	[18:16]	000 = Input 001 = Output 010 = RxD1 011 = SIORxD 100 = IISDI Others = Reserved
PF6	[15:13]	000 = Input 001 = Output 010 = TxD1 011 = SIORDY 100 = IISDO Others = Reserved
PF5	[12:10]	000 = Input 001 = Output 010 = nRTS1 011 = SIOTxD 100 = IISLRCK Others = Reserved
PF4	[9:8]	00 = Input 01 = Output 10 = nXBREQ 11 = nXDREQ0
PF3	[7:6]	00 = Input 01 = Output 10 = nXBACK 11 = nXDACK0
PF2	[5:4]	00 = Input 01 = Output 10 = nWAIT 11 = Reserved
PF1	[3:2]	00 = Input 01 = Output 10 = IICSDA 11 = Reserved
PF0	[1:0]	00 = Input 01 = Output 10 = IICSCL 11 = Reserved

PDATF	Bit	Description
PF[8:0]	[8:0]	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.

PUPF	Bit	Description
PF[8:0]	[8:0]	0: the pull up resistor attached to the corresponding port pin is enabled. 1: the pull up resistor is disabled.

PORT G CONTROL REGISTERS (PCONG, PDATG, PUPG)

Port G control registers are shown in Table 8-8:

If PG0 - PG7 are to be used for wake-up signals in power down mode, the ports will be set in the interrupt mode.

Register	Address	R/W	Description	Reset Value
PCONG	0x01D20040	R/W	Configures the pins of port G	0x0
PDATG	0x01D20044	R/W	The data register for port G	Undef.
PUPG	0x01D20048	R/W	Pull-up disable register for port G	0x0

Table 8-8. Port of Group G Control Registers (PCONG, PDATG, PUPG)

PCONG	Bit	Description
PG7	[15:14]	00 = Input 01 = Output 10 = IISLRCK 11 = EINT7
PG6	[13:12]	00 = Input 01 = Output 10 = IISDO 11 = EINT6
PG5	[11:10]	00 = Input 01 = Output 10 = IISDI 11 = EINT5
PG4	[9:8]	00 = Input 01 = Output 10 = IISCLK 11 = EINT4
PG3	[7:6]	00 = Input 01 = Output 10 = nRTS0 11 = EINT3
PG2	[5:4]	00 = Input 01 = Output 10 = nCTS0 11 = EINT2
PG1	[3:2]	00 = Input 01 = Output 10 = VD5 11 = EINT1
PG0	[1:0]	00 = Input 01 = Output 10 = VD4 11 = EINT0

PDATG	Bit	Description
PG[7:0]	[7:0]	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.

PUPG	Bit	Description
PG[7:0]	[7:0]	0: the pull up resistor attached to the corresponding port pin is enabled. 1: the pull up resistor is disabled.

SPECIAL PULL-UP RESISTOR CONTROL REGISTER (SPUCR)

D[15:0] pin pull-up resistor can be controlled by the SPUCR register.

In STOP/SL_IDLE mode, the data bus(D[31:0] or D[15:0] is in Hi-Z state. But, because of the characteristics of IO 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 PUPC register. D[15:0] pin pull-up resistors can be controlled by the SPUCR register.

In STOP mode, memory control signals can be selected as Hi-z state or previous state in order to protect against memory mal-functions by setting the HZ@STOP field in SPUCR register.

Register	Address	R/W	Description	Reset Value
SPUCR	0x01D2004C	R/W	Special Pull-up register[2:0]	0x4

Table 8-9. D[15:0] Pull-up Control Register (SPUCR)

PCONG	Bit	Description
HZ@STOP	[2]	0 = Previous state of PAD 1 = HZ @ stop
SPUCR1	[1]	0 = DATA[15:8] port pull-up resistor is enabled 1 = DATA[15:8] port pull-up resistor is disabled
SPUCR0	[0]	0 = DATA[7:0] port pull-up resistor is enabled 1 = DATA[7:0] port pull-up resistor is disabled

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.

Register	Address	R/W	Description	Reset Value
EXTINT	0x01D20050	R/W	External Interrupt control Register	0x000000

Table 8-10. External Interrupt Control Register (EXTINT)

EXTINT	Bit	Description
EINT7	[30:28]	Setting the signaling method of the EINT7. 000 = Low level interrupt 001 = High level interrupt 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 interrupt 001 = High level interrupt 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 interrupt 001 = High level interrupt 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 interrupt 001 = High level interrupt 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 interrupt 001 = High level interrupt 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 interrupt 001 = High level interrupt 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 interrupt 001 = High level interrupt 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered
EINT0	[2:0]	Setting the signaling method of the EINT0. 000 = Low level interrupt 001 = High level interrupt 01x = Falling edge triggered 10x = Rising edge triggered 11x = Both edge triggered

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

EXTINTPND (EXTERNAL INTERRUPT PENDING REGISTER)

The external interrupt requests(4, 5, 6, and 7) are 'OR'ed to provide a single interrupt source to interrupt controller. EINT4, EINT5, EINT6, and EINT7 share the same interrupt request line(EINT4/5/6/7) in interrupt controller. If each of the 4 bits in the external interrupt request is generated, EXTINTPNDn will be set as 1. The interrupt service routine must clear the interrupt pending condition(INTPND) after clearing the external pending condition(EXTINTPND). EXTINTPND is cleared by writing 1.

Register	Address	R/W	Description	Reset Value
EXTINTPND	0x01D20054	R/W	External interrupt pending Register	0x00

Table 8-11. D[15:0] Pull-Up Control Register (PUPS)

PUPS	Bit	Description
EXTINTPND3	[3]	If EINT7 is activated, EXINTPND3 bit is set to 1, and also INTPND[21] is set to 1.
EXTINTPND2	[2]	If EINT6 is activated, EXINTPND2 bit is set to 1, and also INTPND[21] is set to 1.
EXTINTPND1	[1]	If EINT5 is activated, EXINTPND1 bit is set to 1, and also INTPND[21] is set to 1.
EXTINTPND0	[0]	If EINT4 is activated, EXINTPND0 bit is set to 1, and also INTPND[21] is set to 1.

9 PWM TIMER

OVERVIEW

The S3C44B0X has six 16-bit timers, each timer can operate in interrupt-based or DMA-based mode. The timers 0, 1, 2, 3 and 4 have the PWM function (Pulse Width Modulation). Timer 5 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 another 8-bit prescaler; and timers 4 & 5 share the other 8-bit prescaler. Each timer, except timers 4 and 5, has a clock-divider which has 5 different divided signals (1/2, 1/4, 1/8, 1/16, 1/32). Timers 4/5 have 4 divided signals(1/2, 1/4, 1/8, 1/16) and one input TCLK/EXTCLK. 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 MCLK signal 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

— Six 16-bit timers with DMA-based or interrupt-based operation

— Three 8-bit prescalers & Two 5-bit dividers & One 4-bit divider

— Programmable duty control of output waveform (PWM)

— Auto-reload mode or one-shot pulse mode

— Dead-zone generator

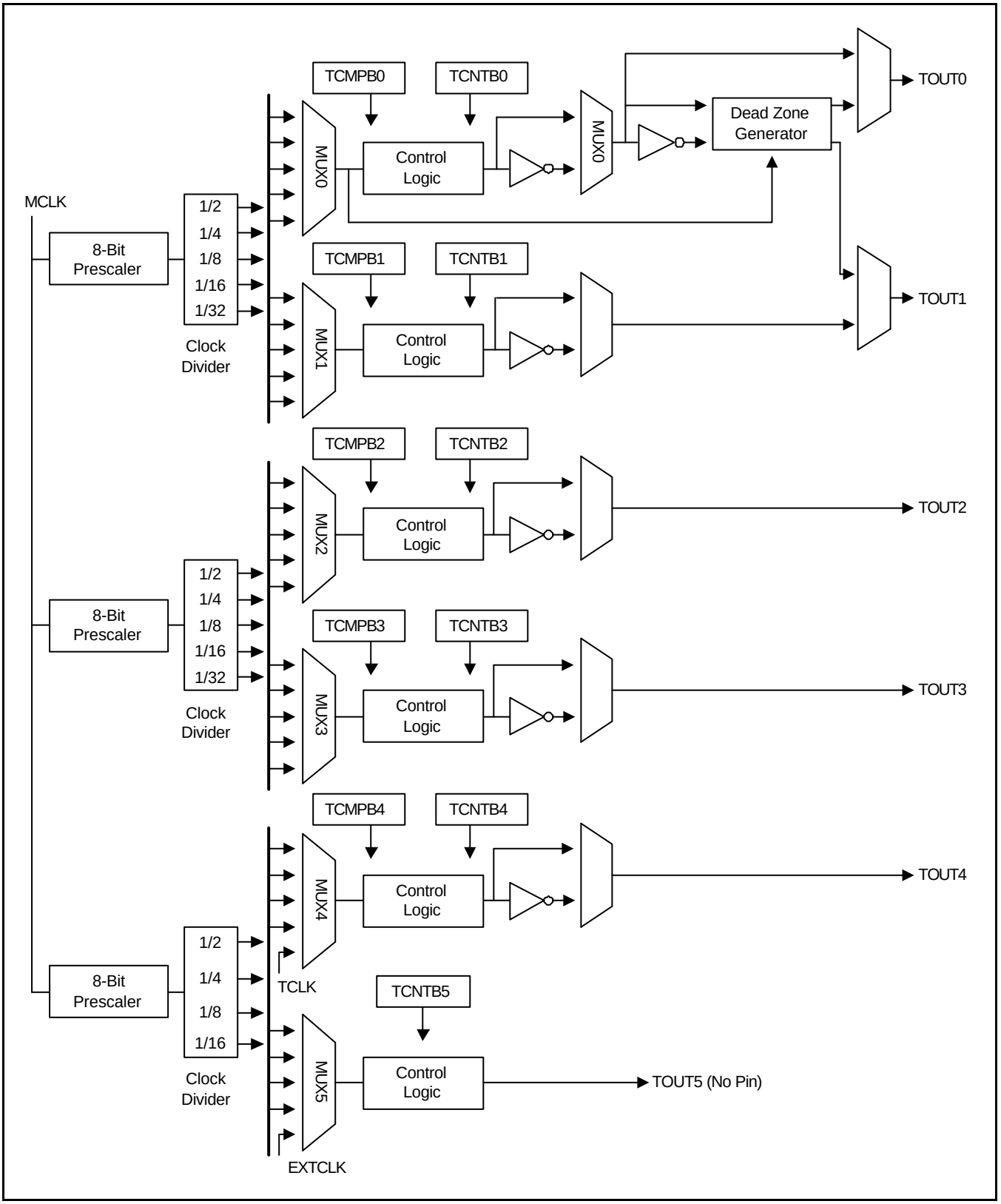


Figure 9-1. 16-bit PWM Timer Block Diagram

PWM TIMER OPERATION

PRESCALER & DIVIDER

An 8-bit prescaler and an independent 4-bit divider make the following output frequencies:

4-bit divider settings	minimum resolution (prescaler = 1)	maximum resolution (prescaler = 255)	maximum interval (TCNTBn = 65535)
1/2 (MCLK = 66 MHz)	0.030 us (33.0 MHz)	7.75 us (58.6 KHz)	0.50 sec
1/4 (MCLK = 66 MHz)	0.060 us (16.5 MHz)	15.5 us (58.6 KHz)	1.02 sec
1/8 (MCLK = 66 MHz)	0.121 us (8.25 MHz)	31.0 us (29.3 KHz)	2.03 sec
1/16 (MCLK = 66 MHz)	0.242 us (4.13 MHz)	62.1 us (14.6 KHz)	4.07 sec
1/32 (MCLK = 66 MHz)	0.485 us (2.06 MHz)	125 us (7.32 KHz)	8.13 sec

BASIC TIMER OPERATION

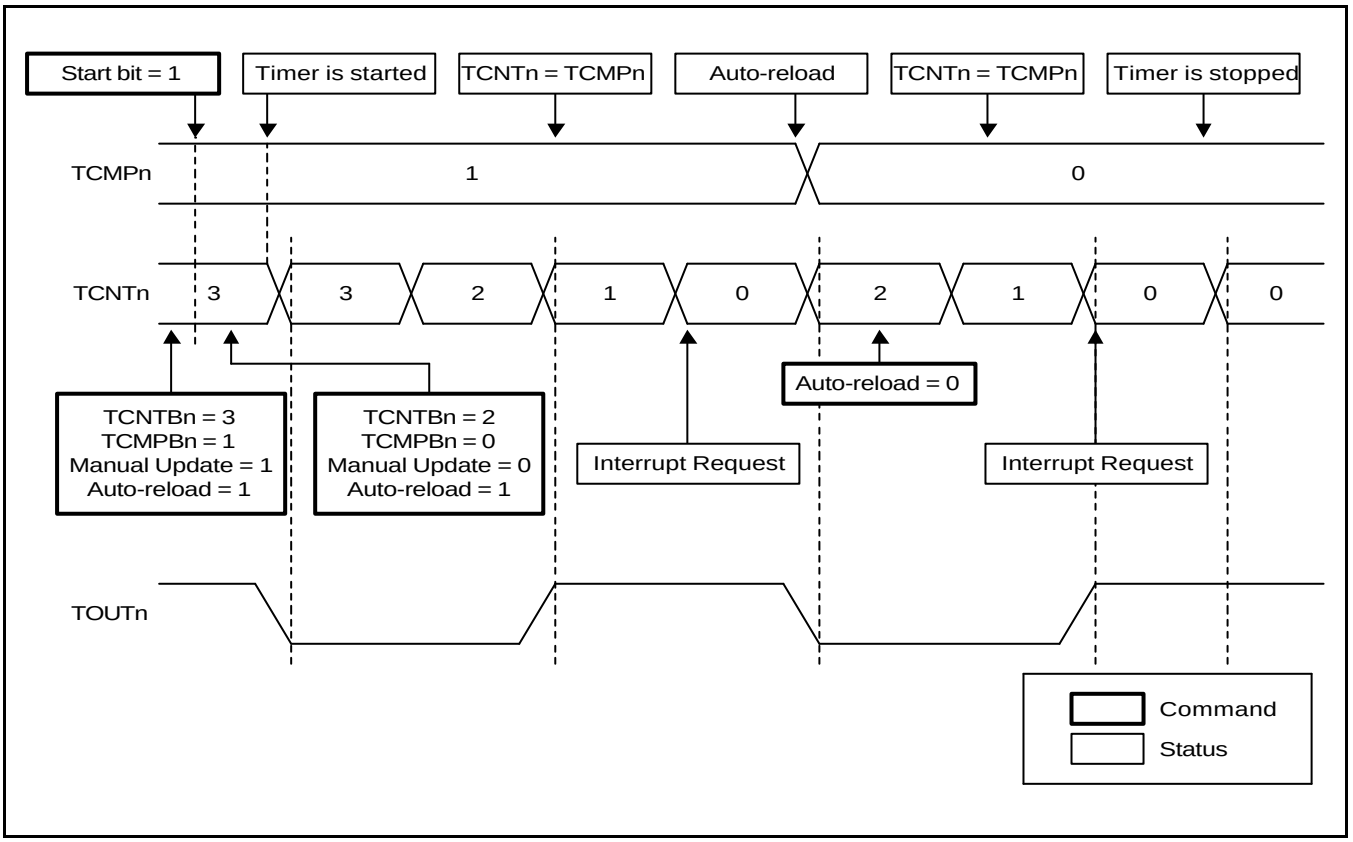


Figure 9-2. Timer operations

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

S3C44B0X 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 counter 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.

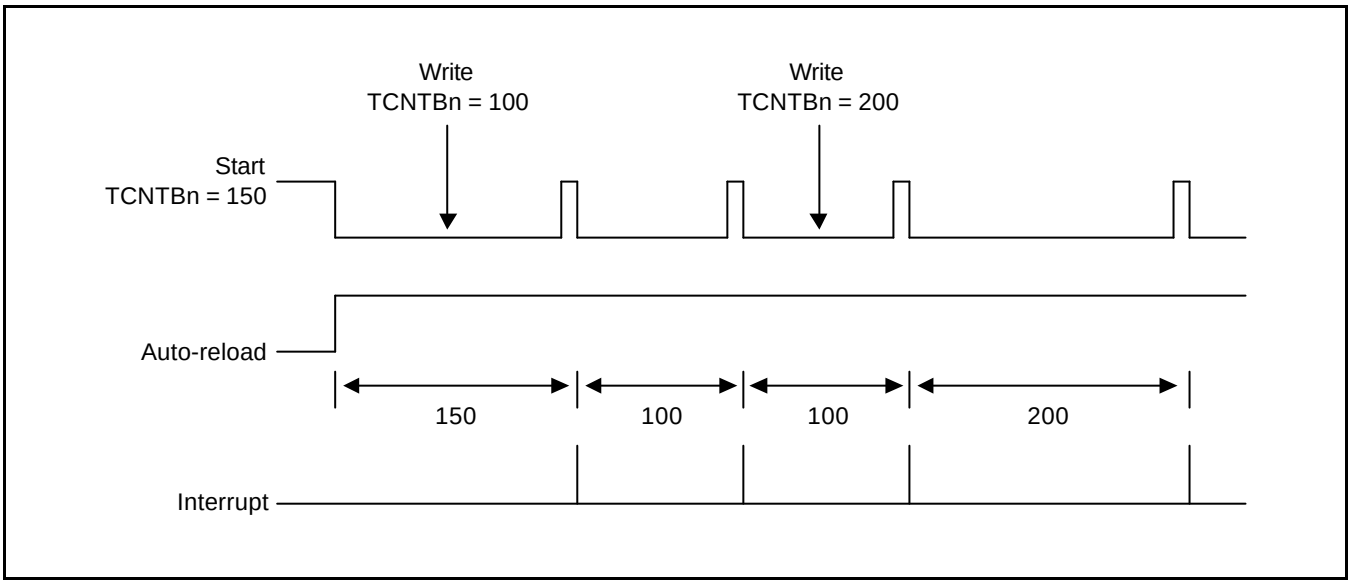


Figure 9-3. Example of Double Buffering Feature

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 is not defined 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.

3) Set the start bit of the 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 or not the timer runs. Therefore, it is desirable that the inverter on/off bit is configured with the manual update bit.

EXAMPLE OF A TIMER OPERATION

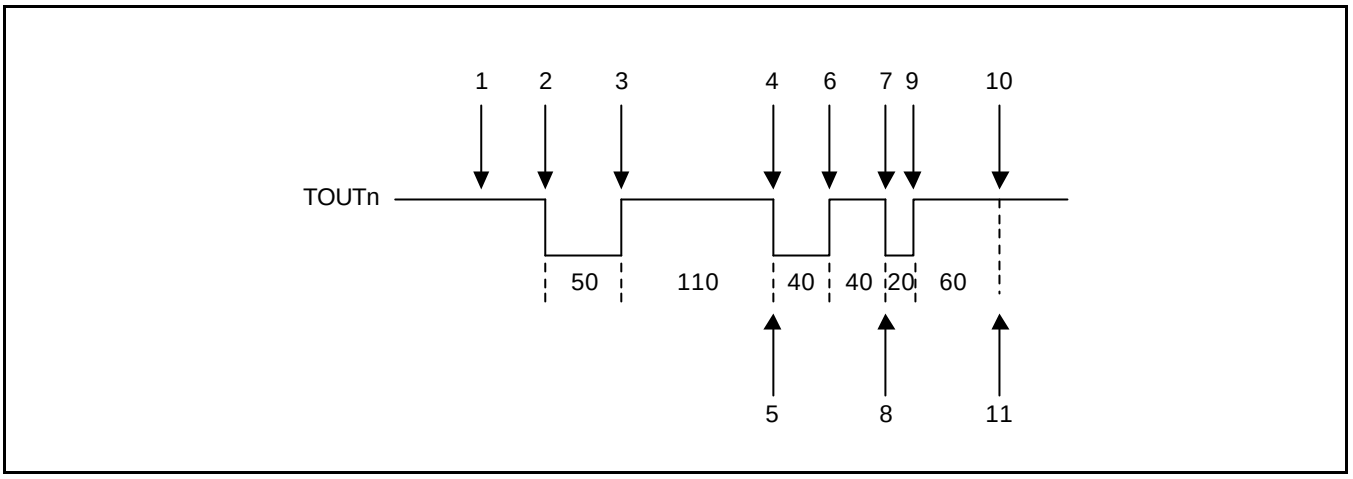


Figure 9-4. Example of a Timer Operation

The result of the following procedure is shown in Figure 9-4.

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)

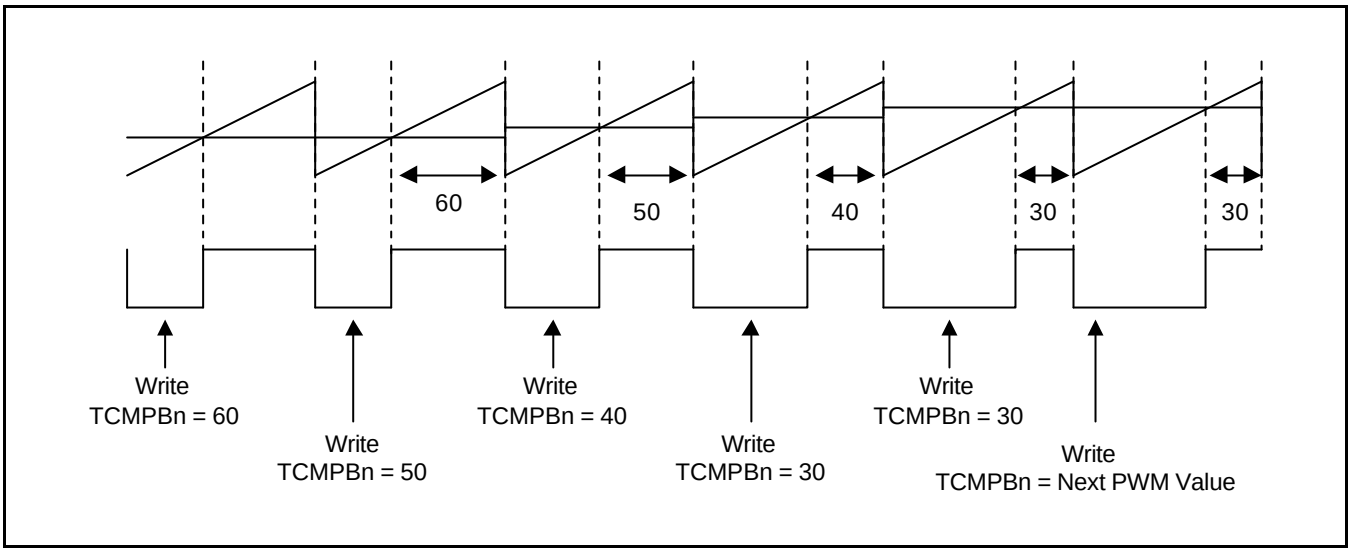


Figure 9-5. Example of PWM

PWM feature can be implemented by using the TCMPBn. PWM frequency is determined by TCNTBn. A PWM value is determined by TCMPBn in figure 9-5.

For a lower PWM output value, decrease the TCMPBn value. For a higher PWM output 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

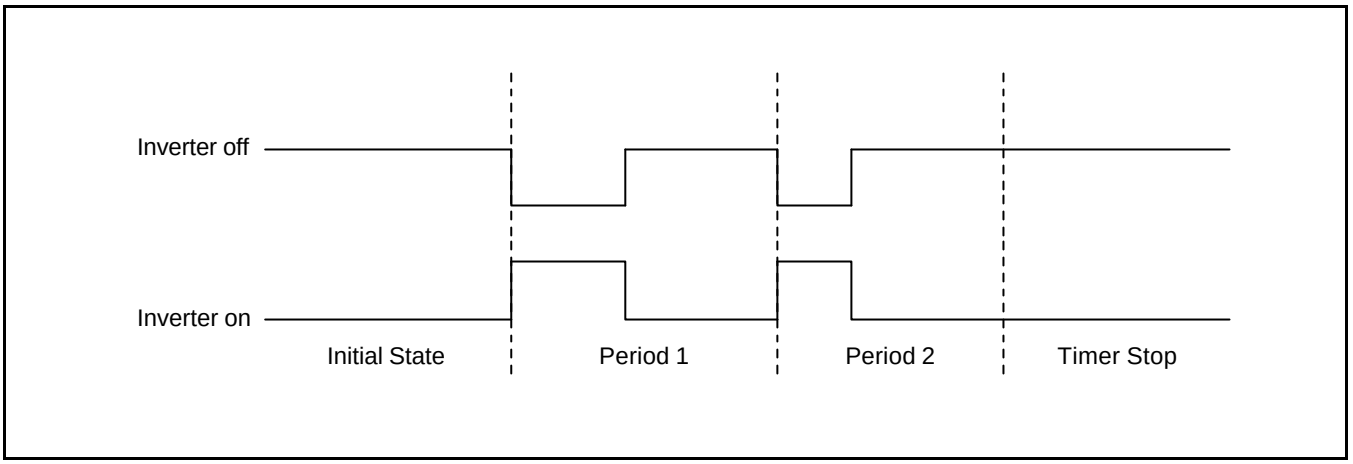


Figure 9-6. Inverter On/Off

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. Write the TCMPBn which is bigger than TCNTBn. This inhibits the TOUTn from going to high because TCMPBn can not have the same value as TCNTn.
4. 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.

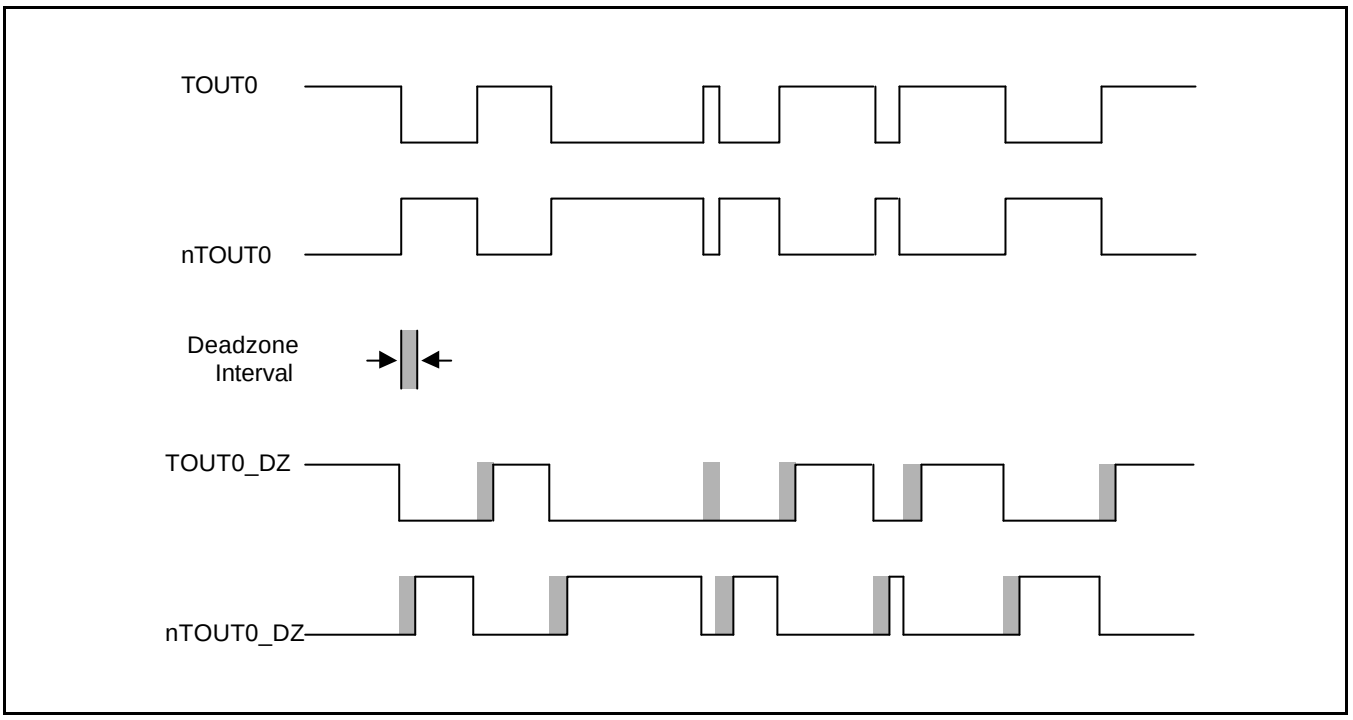


Figure 9-7. The Wave Form When a Dead Zone Feature is Enabled

DMA REQUEST MODE

The PWM timer can generate a DMA request at every specific times. The timer keeps DMA request signal low until the timer receives the ACK signal. When the timer receives the ACK signal, it makes the request signal inactive. One of 6 timers can generate a DMA request. The timer, that generates the DMA request, is determined by setting DMA mode bits(in TCFG1 register). If a timer is configured as DMA request mode, the 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	Timer5 INT
0000	No select	ON	ON	ON	ON	ON	ON
0001	Timer0	OFF	ON	ON	ON	ON	ON
0010	Timer1	ON	OFF	ON	ON	ON	ON
0011	Timer2	ON	ON	OFF	ON	ON	ON
0100	Timer3	ON	ON	ON	OFF	ON	ON
0101	Timer4	ON	ON	ON	ON	OFF	ON
0110	Timer5	ON	ON	ON	ON	ON	OFF
0111	No select	ON	ON	ON	ON	ON	ON

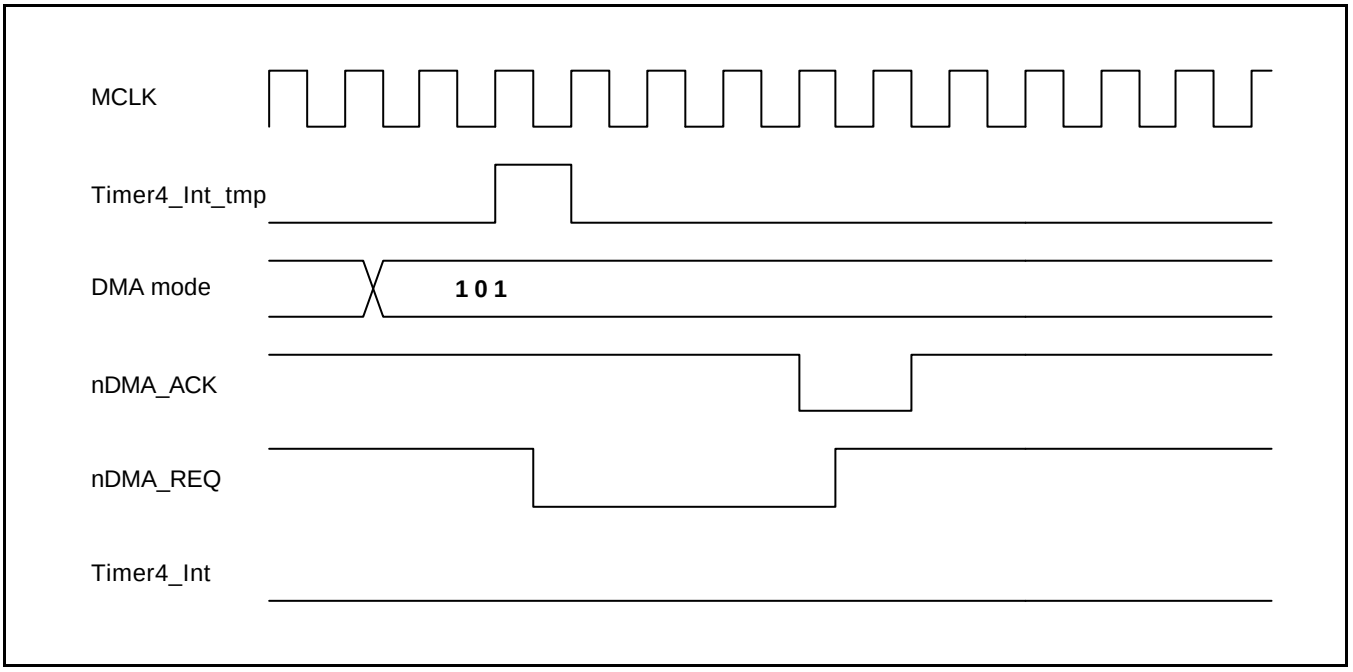


Figure 9-8. The Timer4 DMA mode operation

PWM TIMER CONTROL REGISTERS

TIMER CONFIGURATION REGISTER0 (TCFG0)

Timer input clock Frequency = MCLK / {prescaler value + 1} / {divider value}
{prescaler value} = 0-255
{divider value} = 2, 4, 8, 16, 32

Register	Address	R/W	Description	Reset Value
TCFG0	0x01D50000	R/W	Configures the three 8-bit prescalers	0x00000000

TCFG0	Bit	Description	Initial State
Dead zone length	[31:24]	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 2	[23:16]	These 8 bits determine prescaler value for Timer 4 & 5	0x00
Prescaler 1	[15:8]	These 8 bits determine prescaler value for Timer 2 & 3	0x00
Prescaler 0	[7:0]	These 8 bits determine prescaler value for Timer 0 & 1	0x00

TIMER CONFIGURATION REGISTER1 (TCFG1)

Register	Address	R/W	Description	Reset Value
TCFG1	0x01D50004	R/W	6-MUX & DMA mode selecton register	0x00000000

TCFG1	Bit	Description	Initial State
DMA mode	[27:24]	Select DMA request channel 0000 = No select (all interrupt) 0001 = Timer0 0010 = Timer1 0011 = Timer2 0100 = Timer3 0101 = Timer4 0110 = Timer5 0111 = Reserved	000
MUX 5	[23:20]	Select MUX input for PWM Timer5. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = EXTCLK	000
MUX 4	[19:16]	Select MUX input for PWM Timer4. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = TCLK	000
MUX 3	[15:12]	Select MUX input for PWM Timer3. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 1/32	000
MUX 2	[11:8]	Select MUX input for PWM Timer2. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 1/32	000
MUX 1	[7:4]	Select MUX input for PWM Timer1. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 1/32	000
MUX 0	[3:0]	Select MUX input for PWM Timer0. 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 01xx = 1/32	000

TIMER CONTROL REGISTER (TCON)

Register	Address	R/W	Description	Reset Value
TCON	0x01D50008	R/W	Timer control register	0x00000000

TCON	Bit	Description	initial state
Timer 5 auto reload on/off	[26]	This bit determines auto reload on/off for Timer 5. 0 = One-shot 1 = Interval mode (auto reload)	0
Timer 5 manual update (note)	[25]	This bit determines the manual update for Timer 5. 0 = No operation 1 = Update TCNTB5	0
Timer 5 start/stop	[24]	This bit determines start/stop for Timer 5. 0 = Stop 1 = Start for Timer 5	0
Timer 4 auto reload on/off	[23]	This bit determines auto reload on/off for Timer 4. 0 = One-shot 1 = Interval mode (auto reload)	0
Timer 4 output inverter on/off	[22]	This bit determines output inverter on/off for Timer4. 0 = Inverter off 1 = Inverter on for TOUT4	0
Timer 4 manual update (note)	[21]	This bit determines the manual update for Timer 4. 0 = No operation 1 = Update TCNTB4, TCMPB4	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

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

TIMER CONTROL REGISTER (TCON) (Continued)

TCON	Bit	Description	initial state
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
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	0x01D5000C	R/W	Timer 0 count buffer register	0x00000000
TCMPB0	0x01D50010	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 **NOTE:** This value must be smaller than TCNTB0	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	0x01D50014	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	0x01D50018	R/W	Timer 1 count buffer register	0x00000000
TCMPB1	0x01D5001C	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 **NOTE:** This value must be smaller than TCNTB1	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	0x01D50020	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	0x01D50024	R/W	Timer 2 count buffer register	0x00000000
TCMPB2	0x01D50028	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 **NOTE:** This value must be smaller than TCNTB2	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	0x01D5002C	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	0x01D50030	R/W	Timer 3 count buffer register	0x00000000
TCMPB3	0x01D50034	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 **NOTE:** This value must be smaller than TCNTB3	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	0x01D50038	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 & COMPARE BUFFER REGISTER (TCNTB4, TCMPB4)

Register	Address	R/W	Description	Reset Value
TCNTB4	0x01D5003C	R/W	Timer 4 count buffer register	0x00000000
TCMPB4	0x01D50040	R/W	Timer 4 campare buffer register	0x00000000

TCMPB4	Bit	Description	Initial State
Timer 4 compare buffer register	[15:0]	Setting compare buffer value for Timer 4 **NOTE:** This value must be smaller than TCNTB4	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	0x01D50044	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

TIMER 5 COUNT BUFFER REGISTER (TCNTB5)

Register	Address	R/W	Description	Reset Value
TCNTB5	0x01D50048	R/W	Timer 5 count buffer register	0x00000000

TCNTB5	Bit	Description	Initial State
Timer 5 count buffer register	[15:0]	Setting count buffer value for Timer 5	0x00000000

TIMER 5 COUNT OBSERVATION REGISTER (TCNTO5)

Register	Address	R/W	Description	Reset Value
TCNTO5	0x01D5004C	R	Timer 5 count observation register	0x00000000

TCNTO5	Bit	Description	Initial State
Timer 5 observation register	[15:0]	Setting count observation value for Timer 5	0x00000000

10 UART

OVERVIEW

The S3C44B0X 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 S3C44B0X 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 Figure10-1. The baud-rate generator can be clocked by MCLK. 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

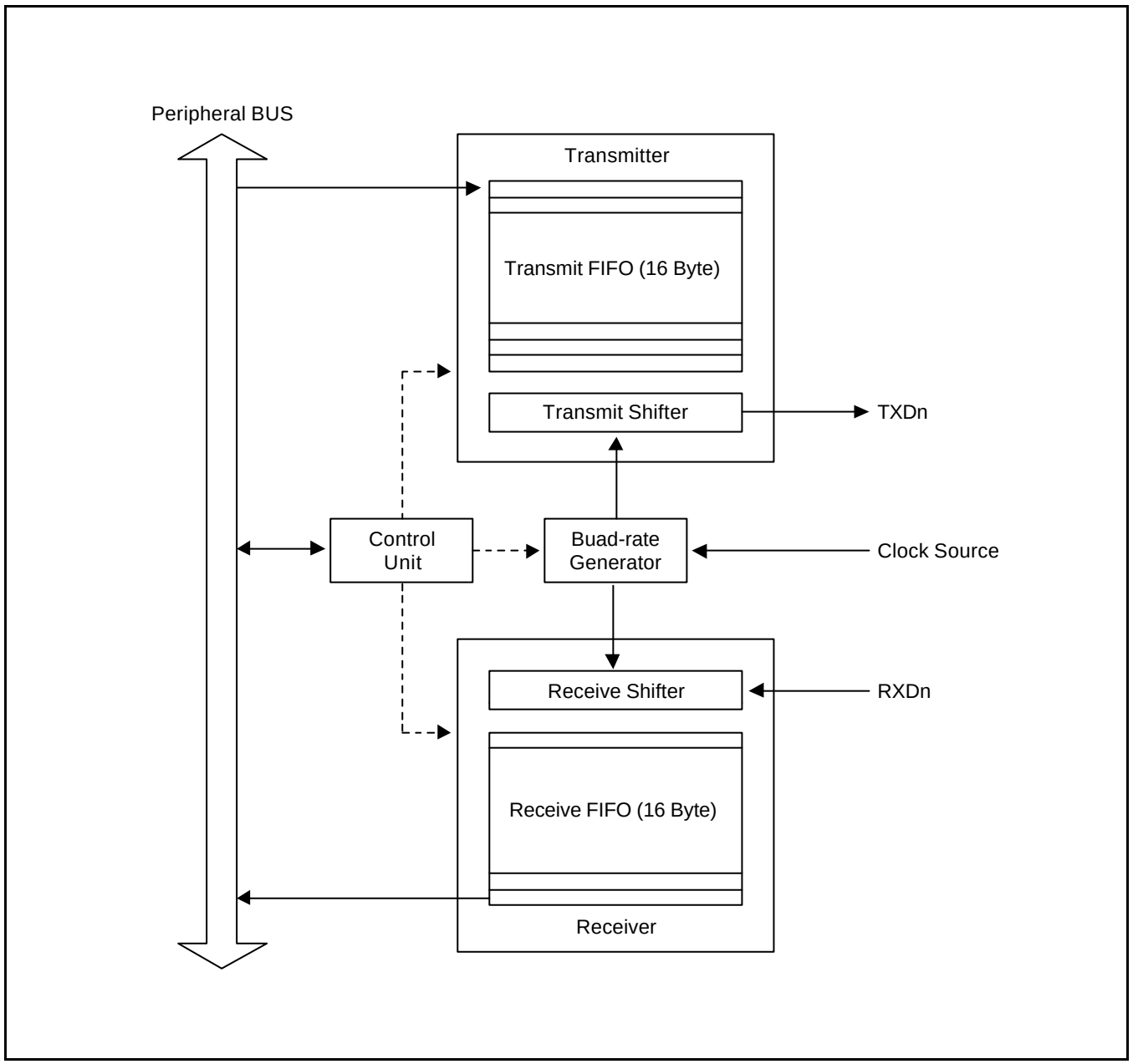


Figure 10-1. UART Block Diagram (with FIFO)

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 (UCONn). The transmitter can also produce the break condition. The break condition forces the serial output to logic 0 state for a duration longer than one frame transmission time. This block transmit break signal after the present transmission word transmits perfectly. After the break signal transmit, continously transmit 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 (UCONn). 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 and the Rx FIFO is not empty in the FIFO mode.

Auto Flow Control(AFC)

S3C44B0X'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).

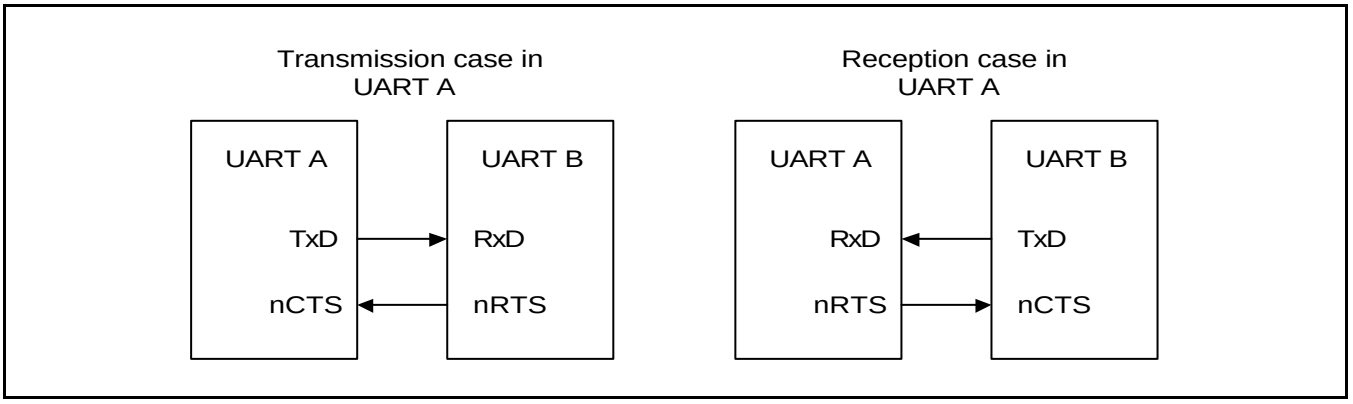


Figure 10-2. UART AFC interface

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

Rx operation

1. Select receive mode(Interrupt or BDMA 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

1. Select transmit mode(Interrupt or BDMA mode)
2. Check the value of UMSTATn[0]. If the value is '1'(nCTS is activated), users write the data to Tx buffer or 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 S3C44B0X has seven status(Tx/Rx/Error) signals: Overrun error, Parity error, Frame error, Break, Receive FIFO/buffer data ready, Transmit FIFO/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, it activates the receive FIFO full status signal which will cause the receive interrupt, if the receive mode in control register is selected as the interrupt mode.

When the transmitter transfers data from its transmit FIFO to its transmit shifter, the transmit FIFO empty status signal is activated. The signal causes the transmit interrupt if the transmit mode in control register is selected as that interrupt mode.

The receive-FIFO-full and transmit-FIFO-empty status signals can also be connected to generate the DMA request signals if the receive/transmit mode is selected as the DMA mode.

Table 10-1. 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 FIFO is not empty and does not receive data during 3 word time, the Rx interrupt will be generated (receive time out).	Each time receive data becomes full, the receive shift register, generates an interrupt.
Tx interrupt	Each time transmit data reaches the trigger level of transmit FIFO, 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 received in bytes, and will generate an error interrupt. When it gets to the top of the receive FIFO, 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 of 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 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	–	

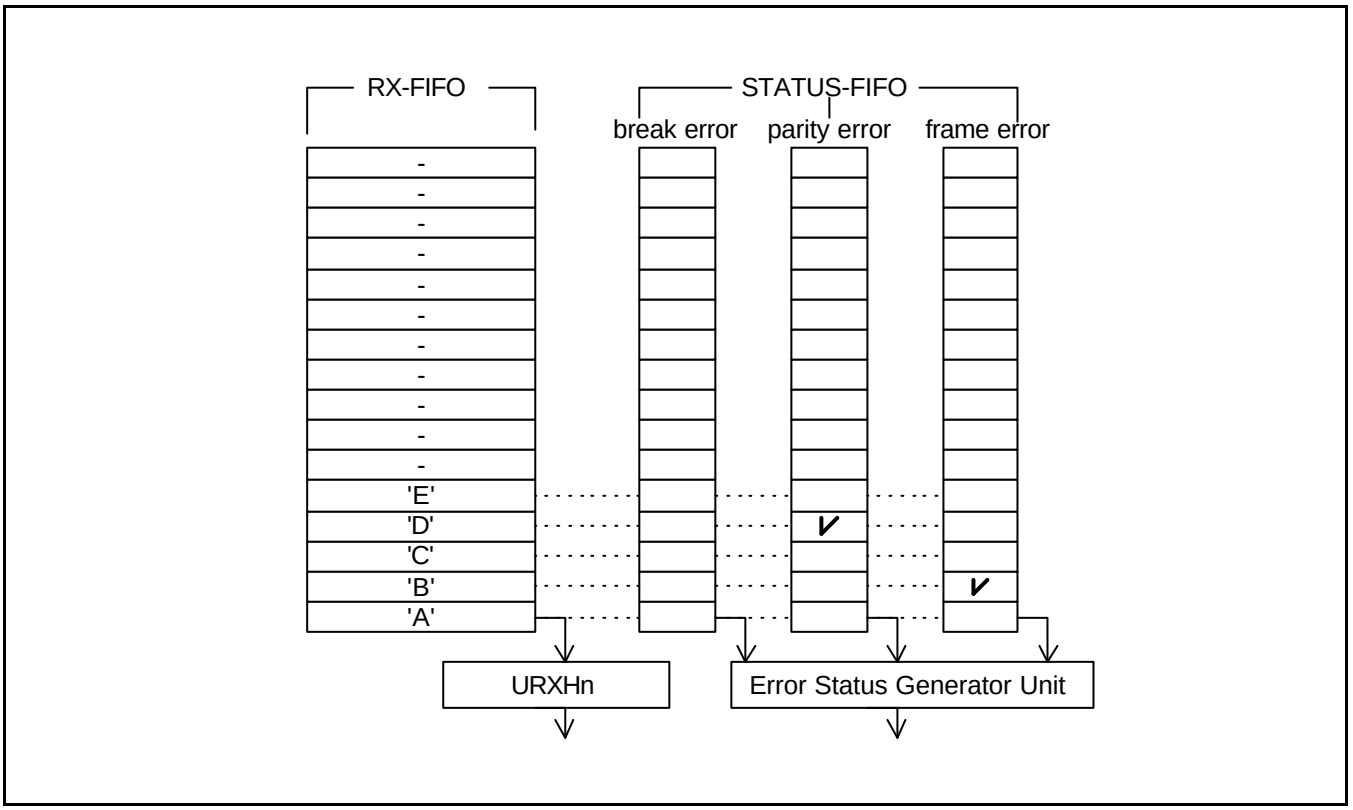


Figure 10-3. A Case showing UART Receiving 5 Characters with 2 Errors

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 S3C44B0X'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 = (round_off)(MCLK/(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 MCLK is 40 MHz , UBRDIVn is:

UBRDIVn = (int)(40000000/(115200 x 16)+0.5) -1
= (int)(21.7+0.5) -1
= 22 -1 = 21

Loop-back Mode

The S3C44B0X 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 more than one frame transmission time on the transmit data output.

IR (Infrared) Mode

The S3C44B0X UART block supports Infrared (IR) transmission and reception, which can be selected by setting the Infrared-mode bit in the UART control register (ULCONn). The implementation of the mode is shown in Figure 10-3.

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 Figures 10-5 and 10-6).

> **Note:** The received pulse is recognized by S3C44B0X which sampling frequency is 1/16 bit frame time, so when it communicates in low speed the Rx pulse must be longer than 1/16 bit frame time. In case of 9600-baud rate, the Rx pulse width must be longer than 6.51us. (Bit frame width = 104.1us, sampling frequency = 6.51us)

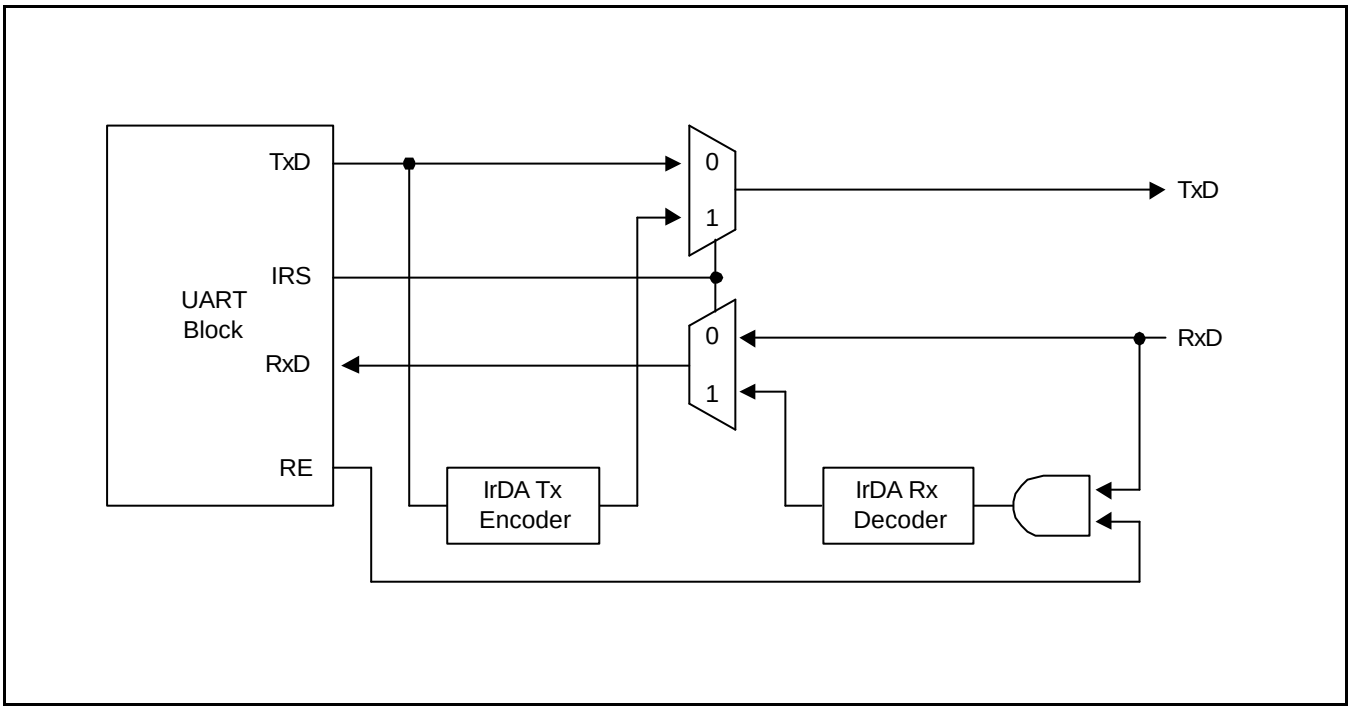


Figure 10-4. IrDA Function Block Diagram

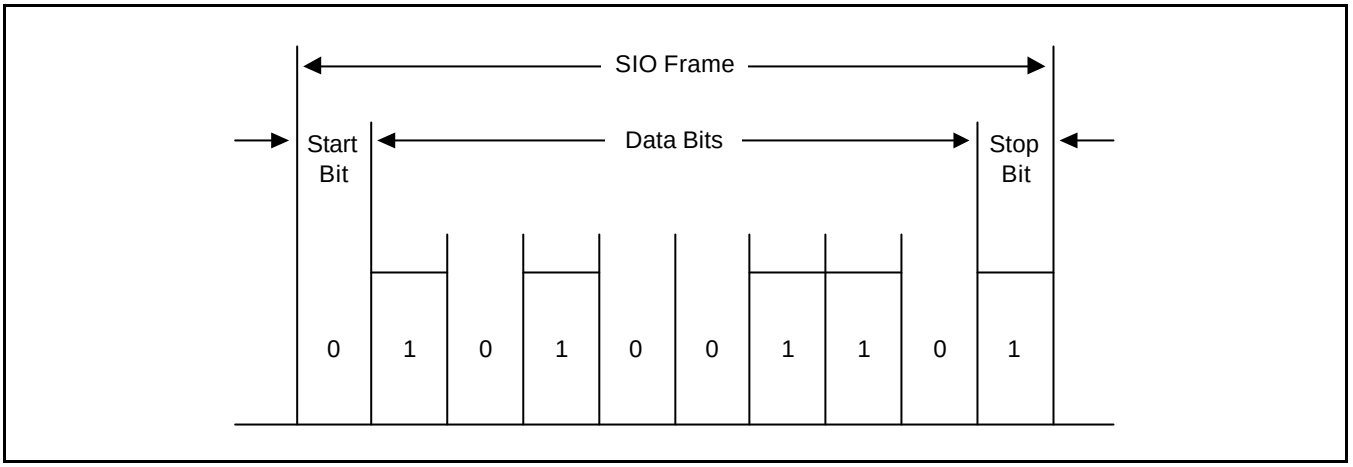


Figure 10-5. Serial I/O Frame Timing Diagram (Normal UART)

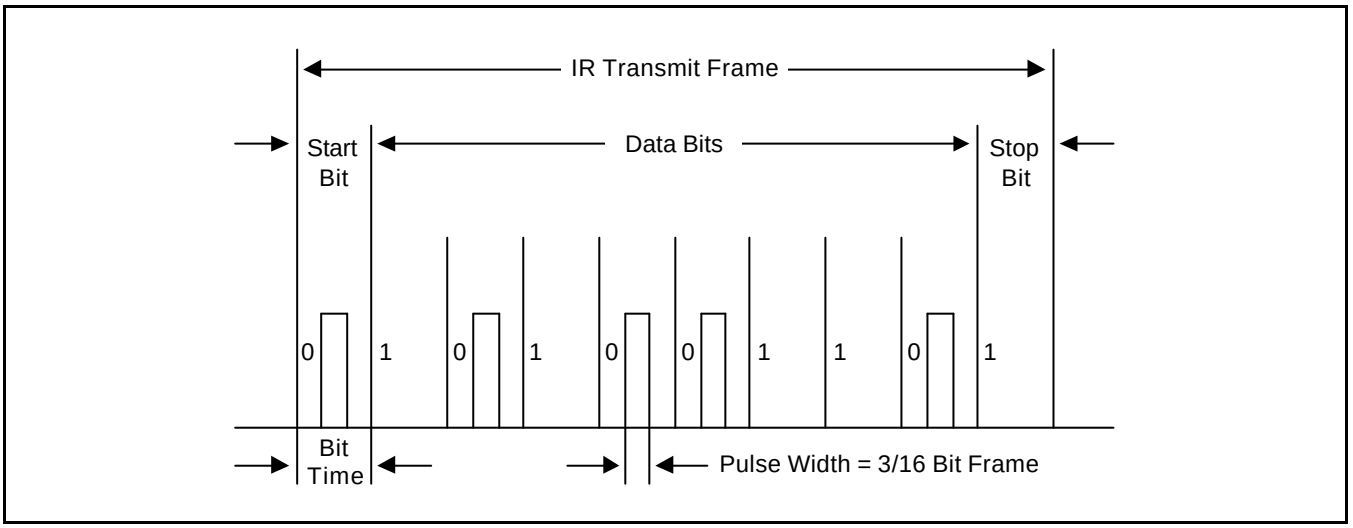


Figure 10-6. Infra-Red Transmit Mode Frame Timing Diagram

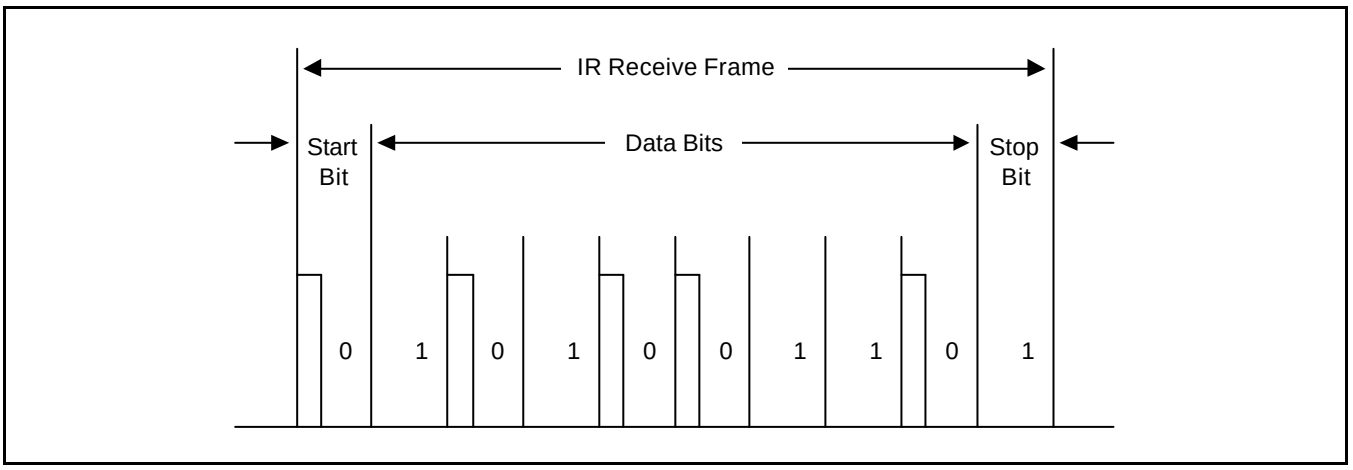


Figure 10-7. Infra-Red Receive Mode Frame Timing Diagram

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	0x01D00000	R/W	UART channel 0 line control register	0x00
ULCON1	0x01D04000	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	0x01D00004	R/W	UART channel 0 control register	0x00
UCON1	0x01D04004	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 the instant Tx buffer becomes empty) 1 = Level (Interrupt is requested while Tx buffer is empty)	0
Rx interrupt type	[8]	Interrupt request type 0 = Pulse (Interrupt is requested the instant Rx buffer receives the data) 1 = Level (Interrupt is requested while Rx buffer is receiving data)	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 holding register. 00 = Disable 01 = Interrupt request or polling mode 10 = BDMA0 request (Only for UART0) 11 = BDMA1 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 = BDMA0 request (Only for UART0) 11 = BDMA1 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	0x01D00008	R/W	UART channel 0 FIFO control register	0x0
UFCON1	0x01D04008	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	0x01D0000C	R/W	UART channel 0 Modem control register	0x0
UMCON1	0x01D0400C	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 S3C44B0X 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	0x01D00010	R	UART channel 0 Tx/Rx status register	0x6
UTRSTAT1	0x01D04010	R	UART channel 1 Tx/Rx status register	0x6

UTRSTATn	Bit	Description	Initial State
Transmit shifter empty	[2]	This bit is automatically set to 1 when the transmit shift register has no valid data to transmit and the transmit shift register is empty. 0 = Not empty 1 = Transmit holding & shifter register empty	1
Transmit buffer empty	[1]	This bit is automatically set to 1 when the transmit buffer register does not contain valid data. 0 =The buffer register is not empty 1 = 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 = Completely empty 1 = The buffer register has a received data If the UART uses the FIFO, users should check Rx FIFO Count bits 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	0x01D00014	R	UART channel 0 Rx error status register	0x0
UERSTAT1	0x01D04014	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	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	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	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	0

NOTE: These bits (UERSATn[3:0]) are automatically cleared to 0 when the UART error status register is read.

UART FIFO STATUS REGISTER

Only the UARTn has a 16-byte transmit FIFO & a 16-byte receive FIFO.

There are two UART FIFO status registers, UFSTAT0 and UFSTAT1, in the UART block.

Register	Address	R/W	Description	Reset Value
UFSTAT0	0x01D00018	R	UART channel 0 FIFO status register	0x00
UFSTAT1	0x01D04018	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 register, UMSTAT0 and UMSTAT1, in the UART block.

Register	Address	R/W	Description	Reset Value
UMSTAT0	0x01D0001C	R	UART channel 0 Modem status register	0x0
UMSTAT1	0x01D0401C	R	UART channel 1 Modem status register	0x0

UMSTATn	Bit	Description	Initial State
Delta CTS	[4]	This bit indicates that the nCTS input to S3C44B0X has changed state since the last time it was read by CPU. (Refer to Fig. 10-7) 0 = Has not changed 1 = Has changed	0
Reserved	[3:1]	Reserved	
Clear to Send	[0]	0 = CTS signal is not activated(nCTS pin is high) 1 = CTS signal is activated(nCTS pin is low)	0

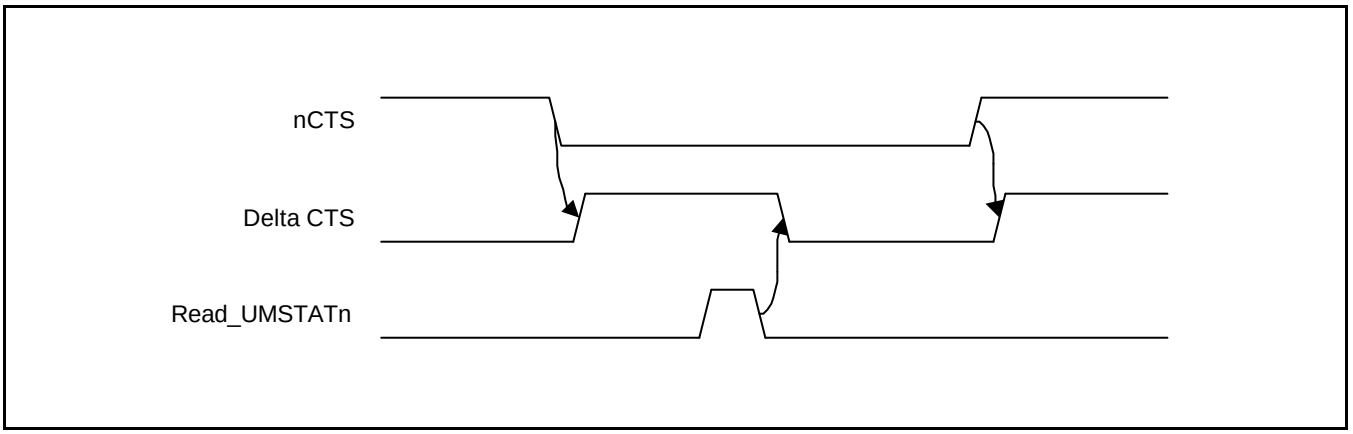


Figure 10-8. nCTS and Delta CTS Timing diagram

UART TRANSMIT HOLDING(BUFFER) REGISTER & FIFO REGISTER

UTXHn has an 8-bit data for transmission data

Register	Address	R/W	Description	Reset Value
UTXH0	0x01D00020(L) 0x01D00023(B)	W (by byte)	UART channel 0 transmit holding register	–
UTXH1	0x01D04020(L) 0x01D04023(B)	W (by byte)	UART channel 1 transmit holding 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 HOLDING (BUFFER) REGISTER & FIFO REGISTER

URXHn has an 8-bit data for received data..

Register	Address	R/W	Description	Reset Value
URXH0	0x01D00024(L) 0x01D00027(B)	R (by byte)	UART channel 0 receive buffer register	–
URXH1	0x01D04024(L) 0x01D04027(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 = (round_off)(MCLK / (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 MCLK is 40 MHz, UBRDIVn is:

UBRDIVn = (int)(40000000 / (115200 x 16)+0.5) -1
= (int)(21.7+0.5) -1
= 22 -1 = 21

Register	Address	R/W	Description	Reset Value
UBRDIV0	0x01D00028	R/W	Baud rate divisior register 0	–
UBRDIV1	0x01D04028	R/W	Baud rate divisior register 1	–

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

11 INTERRUPT CONTROLLER

OVERVIEW

The interrupt controller in S3C44B0X receives the request from 30 interrupt sources. These interrupt sources are provided by internal peripherals such as the DMA controller, UARTand SIO, etc. In these interrupt sources, the four external interrupts(EINT4/5/6/7) are 'OR'ed to the interrupt controller. The UART0 and 1 Error interrupt are 'OR'ed , as well.

The role of the interrupt controller is to ask for the FIQ or IRQ interrupt request to the ARM7TDMI core after making the arbitration process when there are multiple interrupt requests from internal peripherals and external interrupt request pins.

Originally, ARM7TDMI core only permits the FIQ or IRQ interrupt, which is the arbitration process based on priority by software. For example, if you define all interrupt source as IRQ (Interrupt Mode Setting), and, if there are 10 interrupt requests at the same time, you can determine the interrupt service priority by reading the interrupt pending register, which indicates the type of interrupt request that will occur.

This kind of interrupt process requires a long interrupt latency until to jump to the exact service routine.
(The S3C44B0X may support this kind of interrupt processing.)

To solve the above-mentioned problem, S3C44B0X supports a new interrupt processing called vectored interrupt mode, which is a general feature of the CISC type micro-controller, to reduce the interrupt latency. In other words, the hardware inside the S3C44B0X interrupt controller provides the interrupt service vector directly.

When the multiple interrupt sources request interrupts, the hardware priority logic determines which interrupt should be serviced. At same time, this hardware logic applies the jump instruction of the vector table to 0x18(or 0x1c), which performs the jump to the corresponding service routine. Compared with the previous software method, it will reduce the interrupt latency, dramatically.

INTERRUPT CONTROLLER OPERATION

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

If the F-bit of PSR (program status register in ARM7TDMI 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 ARM7TDMI 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 cleared to 0.

Interrupt Mode

ARM7TDMI 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

Indicates whether or not an interrupt request is pending. When a pending bit is set, the interrupt service routine starts whenever the I-flag or F-flag is cleared to 0. Interrupt Pending Register is a read-only register, so the service routine must clear the pending condition by writing a 1 to I_ISPC or F_ISPC.

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 pending bit will be set. If the global mask bit is set to 1, the interrupt pending bit will be set but all interrupts will not be serviced.

INTERRUPT SOURCES

Among 30 interrupt sources, 26 sources are provided for the interrupt controller. Four external interrupt (EINT4/5/6/7) requests are ORed to provide a single interrupt source to the interrupt controller, and two UART error interrupts (UERROR0/1) are the same configuration.

Sources	Descriptions	Master Group	Slave ID
EINT0	External interrupt 0	mGA	sGA
EINT1	External interrupt 1	mGA	sGB
EINT2	External interrupt 2	mGA	sGC
EINT3	External interrupt 3	mGA	sGD
EINT4/5/6/7	External interrupt 4/5/6/7	mGA	sGKA
TICK	RTC Time tick interrupt	mGA	sGKB
INT_ZDMA0	General DMA0 interrupt	mGB	sGA
INT_ZDMA1	General DMA1 interrupt	mGB	sGB
INT_BDMA0	Bridge DMA0 interrupt	mGB	sGC
INT_BDMA1	Bridge DMA1 interrupt	mGB	sGD
INT_WDT	Watch-Dog timer interrupt	mGB	sGKA
INT_UERR0/1	UART0/1 error Interrupt	mGB	sGKB
INT_TIMER0	Timer0 interrupt	mGC	sGA
INT_TIMER1	Timer1 interrupt	mGC	sGB
INT_TIMER2	Timer2 interrupt	mGC	sGC
INT_TIMER3	Timer3 interrupt	mGC	sGD
INT_TIMER4	Timer4 interrupt	mGC	sGKA
INT_TIMER5	Timer5 interrupt	mGC	sGKB
INT_URXD0	UART0 receive interrupt	mGD	sGA
INT_URXD1	UART1 receive interrupt	mGD	sGB
INT_IIC	IIC interrupt	mGD	sGC
INT_SIO	SIO interrupt	mGD	sGD
INT_UTXD0	UART0 transmit interrupt	mGD	sGKA
INT_UTXD1	UART1 transmit interrupt	mGD	sGKB
INT_RTC	RTC alarm interrupt	mGKA	–
INT_ADC	ADC EOC interrupt	mGKB	–

NOTE: EINT4, EINT5, EINT6, and EINT7 share the same interrupt request line. Therefore, the ISR (interrupt service routine) will discriminate these four interrupt sources by reading the EXTINPND[3:0] register. EXTINPND[3:0] must be cleared by writing a 1 in the ISR after the corresponding ISR has been completed.

INTERRUPT PRIORITY GENERATING BLOCK

There is the interrupt priority generating block only for IRQ interrupt request. If the vectored mode is used and an interrupt source is configured as ISR in INTMOD register, the interrupt will be processed by the interrupt priority generating block.

The priority generating block consists of five units, 1 master unit and 4 slave units. Each slave priority generating unit manages six interrupt sources. The master priority generating unit manages 4 slave units and 2 interrupt sources.

Each slave unit has 4 programmable priority sources (sGn) and 2 fixed priotiry sources (sGKn). The priority among the 4 sources in each slave unit is programmable. The other 2 fixed priorities have the lowest priority among the 6 sources.

The master priority generating unit determines the priority between the 4 slave units and 2 interrupt sources. The 2 interrupt sources, INT_RTC and INT_ADC, have the lowest priority among 26 interrupt sources.

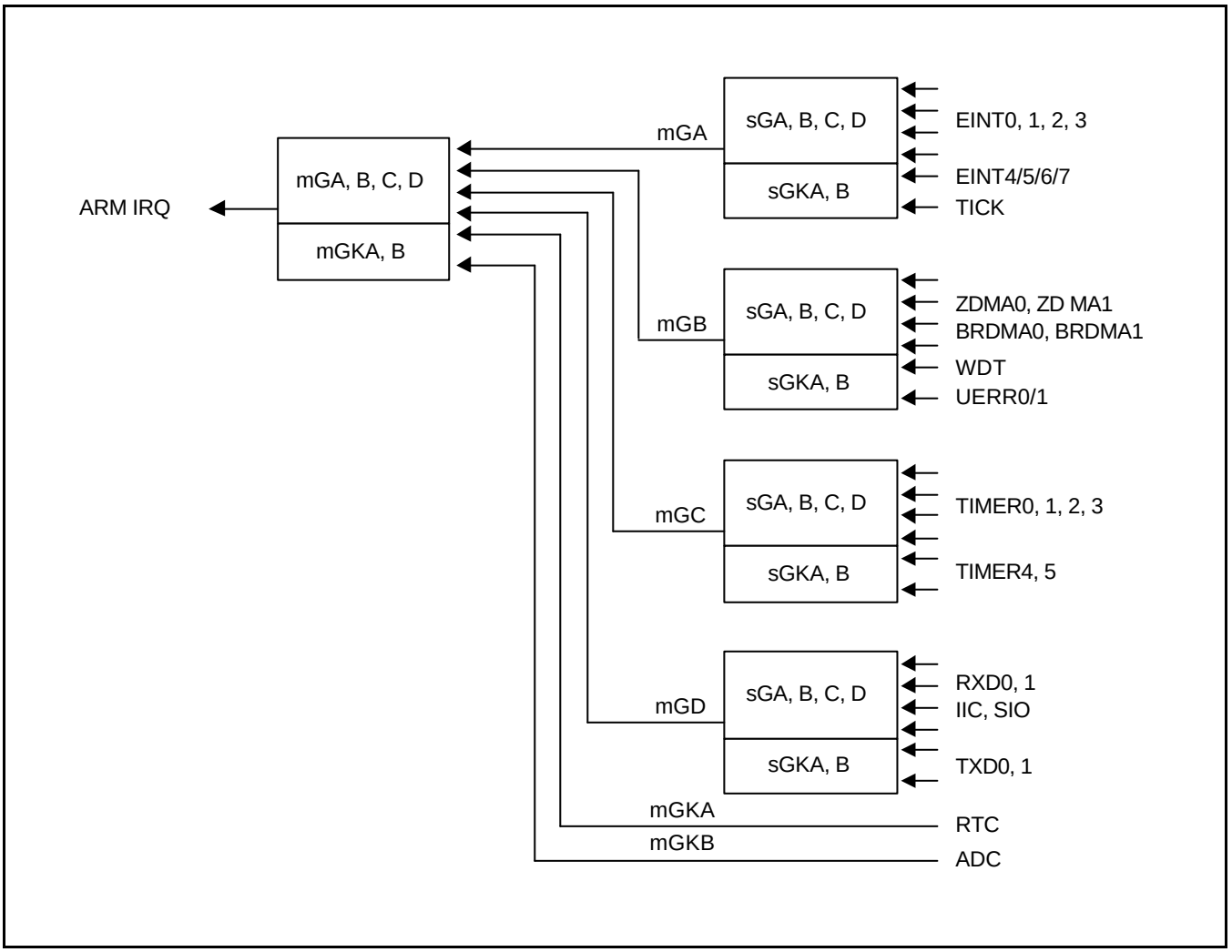


Figure 11-1. Priority Generating Block

INTERRUPT PRIORITY

If source A is configured to FIQ and source B is configured to IRQ, source A has higher priority than source B because a FIQ interrupt has higher priority than an IRQ interrupt in all cases.

If source A and source B are in different master groups and the master group priority of source A is higher than the master group priority of source B, the priority of source A is higher than source B.

If source A and source B are in the same master group and source A has higher priority than source B, source A has the higher priority.

The priorities of sGA, sGB, sGC, and sGD are always higher than those of sGKA and sGKB. The priorities among sGA,sGB,sGC and sGD are programmable or are determined by the round-robin method. Between sGKA and sGKB, sGKA has always the higher priority.

The group priority of mGA, mGB, mGC, and mGD are always higher than that of mGKA and mGKB. So, the priorities of mGKA and mGKB are the lowest among the other interrupt sources. The group priority among mGA, mGB, mGC and, mGD is programmable or is determined by the round-robin method. Between mGKA and mGKB, mGKA always has the higher priority.

VECTORED INTERRUPT MODE (ONLY FOR IRQ)

S3C44B0X has a new feature, the vectored interrupt mode, to reduce the interrupt latency time.

If ARM7TDMI receives the IRQ interrupt request from the interrupt controller, ARM7TDMI executes an instruction at 0x00000018. In vectored interrupt mode, the interrupt controller will load branch instructions on the data bus when ARM7TDMI fetches the instructions at 0x00000018. The branch instructions let the program counter be a unique address corresponding to each interrupt source.

The interrupt controller generates the machine code for branching to the vector address of each interrupt source. For example, If EINT0 is IRQ, the interrupt controller must generate the branch instruction which branches from 0x18 to 0x20. So, the interrupt controller generates the machine code, 0xea000000.

The user program code must locate the branch instruction, which branches to the corresponding ISR (interrupt service routine) at each vector address. The machine code, branch instruction, at the corresponding vector address is calculated as follows;

Branch Instruction machine code for vectored interrupt mode

= 0xea000000 +((<destination address> - <vector address> - 0x8)>>2)

For example, if Timer 0 interrupt to be processed in vector interrupt mode, the branch instruction, which jumps to the ISR, is located at 0x00000060. The ISR start address is 0x10000. The following 32bit machine code is written at 0x00000060.

machine code@0x00000060 : 0xea000000+((0x10000-0x60-0x8)>>2) = 0xea000000+0x3fe6 = 0xea003fe6

The machine code is usually generated automatically by the assembler and therefore the machine code does not have to be calculated as above.

Interrupt Sources	Vector Address
EINT0	0x00000020
EINT1	0x00000024
EINT2	0x00000028
EINT3	0x0000002c
EINT4/5/6/7	0x00000030
INT_TICK	0x00000034
INT_ZDMA0	0x00000040
INT_ZDMA1	0x00000044
INT_BDMA0	0x00000048
INT_BDMA1	0x0000004c
INT_WDT	0x00000050
INT_UERR0/1	0x00000054
INT_TIMER0	0x00000060
INT_TIMER1	0x00000064
INT_TIMER2	0x00000068
INT_TIMER3	0x0000006c
INT_TIMER4	0x00000070
INT_TIMER5	0x00000074
INT_URXD0	0x00000080
INT_URXD1	0x00000084
INT_IIC	0x00000088
INT_SIO	0x0000008c
INT_UTXD0	0x00000090
INT_UTXD1	0x00000094
INT_RTC	0x000000a0
INT_ADC	0x000000c0

EXAMPLE OF VECTORED INTERRUPT MODE

In the vectored interrupt mode, CPU will branch to each interrupt address when an interrupt request is generated. So, there must be the branch instruction to jump each corresponding ISR on it's own address as follows;

```
        ENTRY
        b ResetHandler          ;   0x00
        b HandlerUndef          ;   0x04
        b HandlerSWI            ;   0x08
        b HandlerPabort         ;   0x0c
        b HandlerDabort         ;   0x10
        b .                     ;   0x14
        b HandlerIRQ            ;   0x18
        b HandlerFIQ            ;   0x1c

        ldr pc,=HandlerEINT0    ;   0x20
        ldr pc,=HandlerEINT1
        ldr pc,=HandlerEINT2
        ldr pc,=HandlerEINT3
        ldr pc,=HandlerEINT4567
        ldr pc,=HandlerTICK     ;   0x34
        b .
        b .
        ldr pc,=HandlerZDMA0    ;   0x40
        ldr pc,=HandlerZDMA1
        ldr pc,=HandlerBDMA0
        ldr pc,=HandlerBDMA1
        ldr pc,=HandlerWDT
        ldr pc,=HandlerUERR01   ;   0x54
        b .
        b .
        ldr pc,=HandlerTIMER0   ;   0x60
        ldr pc,=HandlerTIMER1
        ldr pc,=HandlerTIMER2
        ldr pc,=HandlerTIMER3
        ldr pc,=HandlerTIMER4
        ldr pc,=HandlerTIMER5   ;   0x74
        b .
        b .
        ldr pc,=HandlerURXD0    ;   0x80
        ldr pc,=HandlerURXD1
        ldr pc,=HandlerIIC
        ldr pc,=HandlerSIO
        ldr pc,=HandlerUTXD0
        ldr pc,=HandlerUTXD1    ;   0x94
        b .
        b .
        ldr pc,=HandlerRTC      ;   0xa0
        b .
        b .
        b .
        b .
        b .
        b .
        ldr pc,=HandlerADC      ;   0xb4
```

EXAMPLE FOR NON-VECTORED INTERRUPT MODE USING I_ISPR

In the non-vectored interrupt mode, the IRQ/FIQ handler will move the PC to the corresponding ISR by analyzing I_ISPR/F_ISPR register. HandleXXX addresses hold each corresponding ISR routine start addresses. The source code for an IRQ interrupt is as follows;

```
                ENTRY
                b ResetHandler                  ;   for debug
                b HandlerUndef                  ;   handlerUndef
                b HandlerSWI                    ;   SWI interrupt handler
                b HandlerPabort                 ;   handlerPAbort
                b HandlerDabort                 ;   handlerDAbort
                b .                             ;   handlerReserved
                b IsrIRQ
                b HandlerFIQ
                . . . . . .

IsrIRQ
                sub         sp,sp,#4            ;   reserved for PC
                stmfd       sp!,{r8-r9}

                ldr         r9,=I_ISPR
                ldr         r9,[r9]
                mov         r8,#0x0
0               movs        r9,r9,lsr #1
                bcs         %F1
                add         r8,r8,#4
                b           %B0

1               ldr         r9,=HandleADC
                add         r9,r9,r8
                ldr         r9,[r9]
                str         r9,[sp,#8]
                ldmfd       sp!,{r8-r9,pc}
                . . . . . .
HandleADC           #       4
HandleRTC           #       4
HandleUTXD1         #       4
HandleUTXD0         #       4
                . . . . . .
HandleEINT3         #       4
HandleEINT2         #       4
HandleEINT1         #       4
HandleEINT0         #       4               ;   0xc1(c7)fff84
```

INTERRUPT CONTROLLER SPECIAL REGISTERS

INTERRUPT CONTROL REGISTER (INTCON)

Register	Address	R/W	Description	Reset Value
INTCON	0x01E00000	R/W	Interrupt control Register	0x7

INTCON	Bit	Description	initial state
Reserved	[3]	0	0
V	[2]	This bit disables/enables vector mode for IRQ 0 = Vectored interrupt mode 1 = Non-vectored interrupt mode	1
I	[1]	This bit enables IRQ interrupt request line to CPU 0 = IRQ interrupt enable 1 = Reserved **Note : Before using the IRQ interrupt this bit must be cleared.**	1
F	[0]	This bit enables FIQ interrupt request line to CPU 0 = FIQ interrupt enable (Not allowed vectored interrupt mode) 1 = Reserved **Note : Before using the FIQ interrupt this bit must be cleared.**	1

NOTE: FIQ interrupt mode does not support vectored interrupt mode.

INTERRUPT PENDING REGISTER (INTPND)

Each of the 26 bits in the interrupt pending register, INTPND, corresponds to an interrupt source. When an interrupt request is generated, it will be set to 1. The interrupt service routine must then clear the pending condition by writing '1' to the corresponding bit of I_ISPC/F_ISPC. Although several interrupt sources generate requests simultaneously, the INTPND will indicate all interrupt sources that generate an interrupt request. Even if the interrupt source is masked by INTMSK, the corresponding pending bit can be set to 1.

Register	Address	R/W	Description	Reset Value
INTPND	0x01E00004	R	Indicates the interrupt request status. 0 = The interrupt has not been requested 1 = The interrupt source has asserted the interrupt request	0x0000000

INTPND	Bit	Description	Initial State
EINT0	[25]	0 = Not requested, 1 = Requested	0
EINT1	[24]	0 = Not requested, 1 = Requested	0
EINT2	[23]	0 = Not requested, 1 = Requested	0
EINT3	[22]	0 = Not requested, 1 = Requested	0
EINT4/5/6/7	[21]	0 = Not requested, 1 = Requested	0
INT_TICK	[20]	0 = Not requested, 1 = Requested	0
INT_ZDMA0	[19]	0 = Not requested, 1 = Requested	0
INT_ZDMA1	[18]	0 = Not requested, 1 = Requested	0
INT_BDMA0	[17]	0 = Not requested, 1 = Requested	0
INT_BDMA1	[16]	0 = Not requested, 1 = Requested	0
INT_WDT	[15]	0 = Not requested, 1 = Requested	0
INT_UERR0/1	[14]	0 = Not requested, 1 = Requested	0
INT_TIMER0	[13]	0 = Not requested, 1 = Requested	0
INT_TIMER1	[12]	0 = Not requested, 1 = Requested	0
INT_TIMER2	[11]	0 = Not requested, 1 = Requested	0
INT_TIMER3	[10]	0 = Not requested, 1 = Requested	0
INT_TIMER4	[9]	0 = Not requested, 1 = Requested	0
INT_TIMER5	[8]	0 = Not requested, 1 = Requested	0
INT_URXD0	[7]	0 = Not requested, 1 = Requested	0
INT_URXD1	[6]	0 = Not requested, 1 = Requested	0
INT_IIC	[5]	0 = Not requested, 1 = Requested	0
INT_SIO	[4]	0 = Not requested, 1 = Requested	0
INT_UTXD0	[3]	0 = Not requested, 1 = Requested	0
INT_UTXD1	[2]	0 = Not requested, 1 = Requested	0
INT_RTC	[1]	0 = Not requested, 1 = Requested	0

INT_ADC	[0]	0 = Not requested, 1 = Requested	0

INTERRUPT MODE REGISTER (INTMOD)

Each of the 26 bits in the interrupt mode register, INTMOD, corresponds to an interrupt source. When the interrupt mode bit for each source is set to 1, the interrupt is processed by the ARM7TDMI core in the FIQ (fast interrupt) mode. Otherwise, it is processed in the IRQ mode (normal interrupt). The 26 interrupt sources are summarized as follows:

Register	Address	R/W	Description	Reset Value
INTMOD	0x01E00008	R/W	Interrupt mode Register 0 = IRQ mode 1 = FIQ mode	0x0000000

INTMOD	Bit	Description	initial state
EINT0	[25]	0 = IRQ mode 1 = FIQ mode	0
EINT1	[24]	0 = IRQ mode 1 = FIQ mode	0
EINT2	[23]	0 = IRQ mode 1 = FIQ mode	0
EINT3	[22]	0 = IRQ mode 1 = FIQ mode	0
EINT4/5/6/7	[21]	0 = IRQ mode 1 = FIQ mode	0
INT_TICK	[20]	0 = IRQ mode 1 = FIQ mode	0
INT_ZDMA0	[19]	0 = IRQ mode 1 = FIQ mode	0
INT_ZDMA1	[18]	0 = IRQ mode 1 = FIQ mode	0
INT_BDMA0	[17]	0 = IRQ mode 1 = FIQ mode	0
INT_BDMA1	[16]	0 = IRQ mode 1 = FIQ mode	0
INT_WDT	[15]	0 = IRQ mode 1 = FIQ mode	0
INT_UERR0/1	[14]	0 = IRQ mode 1 = FIQ mode	0
INT_TIMER0	[13]	0 = IRQ mode 1 = FIQ mode	0
INT_TIMER1	[12]	0 = IRQ mode 1 = FIQ mode	0
INT_TIMER2	[11]	0 = IRQ mode 1 = FIQ mode	0
INT_TIMER3	[10]	0 = IRQ mode 1 = FIQ mode	0
INT_TIMER4	[9]	0 = IRQ mode 1 = FIQ mode	0
INT_TIMER5	[8]	0 = IRQ mode 1 = FIQ mode	0
INT_URXD0	[7]	0 = IRQ mode 1 = FIQ mode	0
INT_URXD1	[6]	0 = IRQ mode 1 = FIQ mode	0
INT_IIC	[5]	0 = IRQ mode 1 = FIQ mode	0
INT_SIO	[4]	0 = IRQ mode 1 = FIQ mode	0
INT_UTXD0	[3]	0 = IRQ mode 1 = FIQ mode	0
INT_UTXD1	[2]	0 = IRQ mode 1 = FIQ mode	0
INT_RTC	[1]	0 = IRQ mode 1 = FIQ mode	0
INT_ADC	[0]	0 = IRQ mode 1 = FIQ mode	0

INTERRUPT MASK REGISTER (INTMSK)

Each of the 26 bits except the global mask bit in the interrupt mask register, INTMSK, corresponds to an interrupt source. When a source interrupt mask bit is 1 and the corresponding interrupt event occurs, the interrupt is not serviced by the CPU. If the mask bit is 0, the interrupt is serviced upon a request.

If the global mask bit is set to 1, all interrupt requests are not serviced, and the INTPND register is set to 1.

If the INTMSK is changed in ISR(interrupt service routine) and the vectored interrupt is used, an INTMSK bit can not mask an interrupt event, which had been latched in INTPND before the INTMSK bit was set. To clear this problem, clear the corresponding pending bit(INTPND) after changing INTMSK.

The 26 interrupt sources and global mask bit are summarized as follows:

Register	Address	R/W	Description	Reset Value
INTMSK	0x01E0000C	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	0x07ffffff

IMPORTANT NOTES

1. INTMSK register can be masked only when it is sure that the corresponding interrupt does not be requested. If your application should mask any interrupt mask bit(INTMSK) just when the corresponding interrupt is issued, please contact our FAE (field application engineer).
2. If you need that all interrupt is masked, we recommend that I/F bits in CPSR are set using MRS, MSR instructions. The I, F bit in CPSR can be masked even when any interrupt is issued.

INTMSK	Bit	Description	initial state
Reserved	[27]		0
Global	[26]	0 = Service available 1 = Masked	1
EINT0	[25]	0 = Service available 1 = Masked	1
EINT1	[24]	0 = Service available 1 = Masked	1
EINT2	[23]	0 = Service available 1 = Masked	1
EINT3	[22]	0 = Service available 1 = Masked	1
EINT4/5/6/7	[21]	0 = Service available 1 = Masked	1
INT_TICK	[20]	0 = Service available 1 = Masked	1
INT_ZDMA0	[19]	0 = Service available 1 = Masked	1
INT_ZDMA1	[18]	0 = Service available 1 = Masked	1
INT_BDMA0	[17]	0 = Service available 1 = Masked	1
INT_BDMA1	[16]	0 = Service available 1 = Masked	1
INT_WDT	[15]	0 = Service available 1 = Masked	1
INT_UERR0/1	[14]	0 = Service available 1 = Masked	1
INT_TIMER0	[13]	0 = Service available 1 = Masked	1
INT_TIMER1	[12]	0 = Service available 1 = Masked	1
INT_TIMER2	[11]	0 = Service available 1 = Masked	1
INT_TIMER3	[10]	0 = Service available 1 = Masked	1
INT_TIMER4	[9]	0 = Service available 1 = Masked	1
INT_TIMER5	[8]	0 = Service available 1 = Masked	1
INT_URXD0	[7]	0 = Service available 1 = Masked	1
INT_URXD1	[6]	0 = Service available 1 = Masked	1
INT_IIC	[5]	0 = Service available 1 = Masked	1
INT_SIO	[4]	0 = Service available 1 = Masked	1
INT_UTXD0	[3]	0 = Service available 1 = Masked	1
INT_UTXD1	[2]	0 = Service available 1 = Masked	1
INT_RTC	[1]	0 = Service available 1 = Masked	1
INT_ADC	[0]	0 = Service available 1 = Masked	1

IRQ VECTORED MODE REGISTERS

The priority generating block consists of five units, 1 master unit and 4 slave units. Each slave priority generating unit manages six interrupt sources. The master priority generating unit manages 4 slave units and 2 interrupt sources.

Each slave unit has 4 programmable priority source (sGn) and 2 fixed priority sources (kn). The priority among the 4 sources in each slave unit is determined the I_PSLV register. The other 2 fixed priorities have the lowest priority among the 6 sources.

The master priority generating unit determines the priority between 4 slave units and 2 interrupt sources using the I_PMST register. The 2 interrupt sources,INT_RTC and INT_ADC, have the lowest priority among the 26 interrupt sources.

If several interrupts are requested at the same time, the I_ISPR register shows only the requested interrupt source with the highest priority.

Register	Address	R/W	Description	Reset Value
I_PSLV	0x01E00010	R/W	IRQ priority of slave register	0x1b1b1b1b
I_PMST	0x01E00014	R/W	IRQ priority of master register	0x00001f1b
I_CSLV	0x01E00018	R	Current IRQ priority of slave register	0x1b1b1b1b
I_CMST	0x01E0001C	R	Current IRQ priority of master register	0x0000xx1b
I_ISPR	0x01E00020	R	IRQ interrupt service pending register (Only one service bit can be set)	0x00000000
I_ISPC	0x01E00024	W	IRQ interrupt service clear register (Whatever to be set, INTPND will be cleared automatically)	Undef.

IMPORTANT NOTE

In FIQ mode, there is no service pending register like I_ISPR, users must check INTPND resister.

IRQ PRIORITY OF SLAVE REGISTER (I_PSLV)

I_PSLV determines the interrupt priorities among the 4 interrupt sources of each slave group.

Register	Address	R/W	Description	Reset Value
I_PSLV	0x01E00010	R/W	IRQ priority of slave register	0x1b1b1b1b

I_PSLV	Bit	Description	Initial State
PSLAVE@mGA	[31:24]	Determine the priorities among sGA, B, C, D of mGA. Each sGn must have a different priority.	0x1b
PSLAVE@mGB	[23:16]	Determine the priorities among sGA, B, C, D of mGB. Each sGn must have a different priority.	0x1b
PSLAVE@mGC	[15:8]	Determine the priorities among sGA, B, C, D of mGC. Each sGn must have a different priority.	0x1b
PSLAVE@mGD	[7:0]	Determine the priorities among sGA, B, C, D of mGD. Each sGn must have a different priority.	0x1b

PSLAVE@mGA	Bit	Description	Initial State
sGA (EINT0)	[31:30]	00: 1st 01: 2nd 10: 3rd 11: 4th	00
sGB (EINT1)	[29:28]	00: 1st 01: 2nd 10: 3rd 11: 4th	01
sGC (EINT2)	[27:26]	00: 1st 01: 2nd 10: 3rd 11: 4th	10
sGD (EINT3)	[25:24]	00: 1st 01: 2nd 10: 3rd 11: 4th	11

PSLAVE@mGB	Bit	Description	Initial State
sGA (INT_ZDMA0)	[23:22]	00: 1st 01: 2nd 10: 3rd 11: 4th	00
sGB (INT_ZDMA1)	[21:20]	00: 1st 01: 2nd 10: 3rd 11: 4th	01
sGC (INT_BDMA0)	[19:18]	00: 1st 01: 2nd 10: 3rd 11: 4th	10
sGD (INT_BDMA1)	[17:16]	00: 1st 01: 2nd 10: 3rd 11: 4th	11

PSLAVE@mGC	Bit	Description	Initial State
sGA (TIMER0)	[15:14]	00: 1st 01: 2nd 10: 3rd 11: 4th	00
sGB (TIMER1)	[13:12]	00: 1st 01: 2nd 10: 3rd 11: 4th	01
sGC (TIMER2)	[11:10]	00: 1st 01: 2nd 10: 3rd 11: 4th	10
sGD (TIMER3)	[9:8]	00: 1st 01: 2nd 10: 3rd 11: 4th	11

PSLAVE@mGD	Bit	Description	Initial State
sGA (INT_URXD0)	[7:6]	00: 1st 01: 2nd 10: 3rd 11: 4th	00
sGB (INT_URXD1)	[5:4]	00: 1st 01: 2nd 10: 3rd 11: 4th	01
Sgc (INT_IIC)	[3:2]	00: 1st 01: 2nd 10: 3rd 11: 4th	10
sGD (INT_SIO)	[1:0]	00: 1st 01: 2nd 10: 3rd 11: 4th	11

NOTE: The items in I_PSLAVE must be configured with different priorities even if the corresponding interrupt source is not used.

IRQ PRIORITY OF MASTER REGISTER (I_PMST)

I_PMST determines the interrupt priorities among the 4 slave groups.

Register	Address	R/W	Description	Reset Value
I_PMST	0x01E00014	R/W	IRQ priority of master register	0x00001f1b

I_PMST	Bit	Description	Initial State
Reserved	[15:13]		000
M	[12]	Master operating mode 0 = round robin 1 = fix mode	1
FxSLV[A:D]	[11:8]	Slave operating mode 0 = round robin 1 = fix mode	1111
PMASTER	[7:0]	Determine the priorities among 4 slave units.	0x1b

FxSLV	Bit	Description	Initial State
Fx@mGA	[11]	Determines the operating mode of slave unit @mGA	1
Fx@mGB	[10]	Determines the operating mode of slave unit @mGB	1
Fx@mGC	[9]	Determines the operating mode of slave unit @mGC	1
Fx@mGD	[8]	Determines the operating mode of slave unit @mGD	1

PMASTER	Bit	Description	Initial State
mGA	[7:6]	00: 1st 01: 2nd 10: 3rd 11: 4th	00
mGB	[5:4]	00: 1st 01: 2nd 10: 3rd 11: 4th	01
mGC	[3:2]	00: 1st 01: 2nd 10: 3rd 11: 4th	10
mGD	[1:0]	00: 1st 01: 2nd 10: 3rd 11: 4th	11

NOTE: The items in I_PMST must be configured with different priorities even if the corresponding interrupt source is not used.

CURRENT IRQ PRIORITY OF SLAVE REGISTER (I_CSLV)

I_CSLV indicates the current priority status among the sources in each slave group. The I_CSLV may differ from I_PSLV if the round-robin mode is enabled.

Register	Address	R/W	Description	Reset Value
I_CSLV	0x01E00018	R	Current IRQ priorities of slave register	0x1b1b1b1b

I_CSLV	Bit	Description	Initial State
CSLAVE@mGA	[31:24]	Indicate the current priority status of mGA	0x1b
CSLAVE@mGB	[23:16]	Indicate the current priority status of mGB	0x1b
CSLAVE@mGC	[15:8]	Indicate the current priority status of mGC	0x1b
CSLAVE@mGD	[7:0]	Indicate the current priority status of mGD	0x1b

CSLAVE@mGA	Bit	Description	Initial State
sGA (EINT0)	[31:30]	00: 1st 01: 2nd 10: 3rd 11: 4th	00
sGB (EINT1)	[29:28]	00: 1st 01: 2nd 10: 3rd 11: 4th	01
sGC (EINT2)	[27:26]	00: 1st 01: 2nd 10: 3rd 11: 4th	10
sGD (EINT3)	[25:24]	00: 1st 01: 2nd 10: 3rd 11: 4th	11

CSLAVE@mGB	Bit	Description	Initial State
sGA (INT_ZDMA0)	[23:22]	00: 1st 01: 2nd 10: 3rd 11: 4th	00
sGB (INT_ZDMA1)	[21:20]	00: 1st 01: 2nd 10: 3rd 11: 4th	01
sGC (INT_BDMA0)	[19:18]	00: 1st 01: 2nd 10: 3rd 11: 4th	10
sGD (INT_BDMA1)	[17:16]	00: 1st 01: 2nd 10: 3rd 11: 4th	11

CSLAVE@mGC	Bit	Description	Initial State
sGA (TIMER0)	[15:14]	00: 1st 01: 2nd 10: 3rd 11: 4th	00
sGB (TIMER1)	[13:12]	00: 1st 01: 2nd 10: 3rd 11: 4th	01
sGC (TIMER2)	[11:10]	00: 1st 01: 2nd 10: 3rd 11: 4th	10
sGD (TIMER3)	[9:8]	00: 1st 01: 2nd 10: 3rd 11: 4th	11

CSLAVE@mGD	Bit	Description	Initial State
sGA (INT_URXD0)	[7:6]	00: 1st 01: 2nd 10: 3rd 11: 4th	00
sGB (INT_URXD1)	[5:4]	00: 1st 01: 2nd 10: 3rd 11: 4th	01
sGC (INT_IIC)	[3:2]	00: 1st 01: 2nd 10: 3rd 11: 4th	10
sGD (INT_SIO)	[1:0]	00: 1st 01: 2nd 10: 3rd 11: 4th	11

CURRENT IRQ PRIORITY OF MASTER REGISTER (I_CMST)

I_CMST indicates the current priority status among the slave groups

Register	Address	R/W	Description	Reset Value
I_CMST	0x01E0001C	R	Current IRQ priority of master register	0x0000xx1b

I_CMST	Bit	Description	Initial State
Reserved	[15:14]		00
VECTOR	[13:8]	The lower 6 bits of corresponding branch machine code	unknown
CMASTER	[7:0]	Current priority of master	00011011

CMASTER	Bit	Description	Initial State
mGA	[7:6]	00: 1st 01: 2nd 10: 3rd 11: 4th	00
mGB	[5:4]	00: 1st 01: 2nd 10: 3rd 11: 4th	01
mGC	[3:2]	00: 1st 01: 2nd 10: 3rd 11: 4th	10
mGD	[1:0]	00: 1st 01: 2nd 10: 3rd 11: 4th	11

IRQ INTERRUPT SERVICE PENDING REGISTER (I_ISPR)

I_ISPR indicates the interrupt being currently serviced. Although the several interrupt pending bits are all turned on, only one bit will be turned on.

Register	Address	R/W	Description	Reset Value
I_ISPR	0x01E00020	R	IRQ interrupt service pending register	0x00000000

I_ISPR	Bit	Description	Initial State
EINT0	[25]	0 = not serviced 1 = serviced now	0
EINT1	[24]	0 = not serviced 1 = serviced now	0
EINT2	[23]	0 = not serviced 1 = serviced now	0
EINT3	[22]	0 = not serviced 1 = serviced now	0
EINT4/5/6/7	[21]	0 = not serviced 1 = serviced now	0
INT_TICK	[20]	0 = not serviced 1 = serviced now	0
INT_ZDMA0	[19]	0 = not serviced 1 = serviced now	0
INT_ZDMA1	[18]	0 = not serviced 1 = serviced now	0
INT_BDMA0	[17]	0 = not serviced 1 = serviced now	0
INT_BDMA1	[16]	0 = not serviced 1 = serviced now	0
INT_WDT	[15]	0 = not serviced 1 = serviced now	0
INT_UERR0/1	[14]	0 = not serviced 1 = serviced now	0
INT_TIMER0	[13]	0 = not serviced 1 = serviced now	0
INT_TIMER1	[12]	0 = not serviced 1 = serviced now	0
INT_TIMER2	[11]	0 = not serviced 1 = serviced now	0
INT_TIMER3	[10]	0 = not serviced 1 = serviced now	0
INT_TIMER4	[9]	0 = not serviced 1 = serviced now	0
INT_TIMER5	[8]	0 = not serviced 1 = serviced now	0
INT_URXD0	[7]	0 = not serviced 1 = serviced now	0
INT_URXD1	[6]	0 = not serviced 1 = serviced now	0
INT_IIC	[5]	0 = not serviced 1 = serviced now	0
INT_SIO	[4]	0 = not serviced 1 = serviced now	0
INT_UTXD0	[3]	0 = not serviced 1 = serviced now	0
INT_UTXD1	[2]	0 = not serviced 1 = serviced now	0
INT_RTC	[1]	0 = not serviced 1 = serviced now	0
INT_ADC	[0]	0 = not serviced 1 = serviced now	0

IRQ/FIQ INTERRUPT SERVICE PENDING CLEAR REGISTER (I_ISPC/F_ISPC)

I_ISPC/F_ISPC clears the interrupt pending bit (INTPND). I_ISPC/F_ISPC also informs the interrupt controller of the end of corresponding ISR (interrupt service routine). At the end of ISR(interrupt service routine), the corresponding pending bit must be cleared.

The bit of INTPND bit is cleared to zero by writing '1' on I_ISPC/F_ISPC. This feature reduces the code size to clear the INTPND. The corresponding INTPND bit is cleared automatically by I_ISPC/F_ISPC, INTPND register can not be cleared directly.

NOTE

To clear the I_ISPC/F_ISPC, the following two rules has to be obeyed.

1) The I_ISPC/F_ISPC registers are accessed only once in ISR(interrupt service routine).
2) The pending bit in I_ISPR/INTPND register should be cleared by writing I_ISPC register.

If these two rules are not followed, I_ISPR and INTPND register may be 0 although the interrupt has been requested.

Register	Address	R/W	Description	Reset Value
I_ISPC	0x01E00024	W	IRQ interrupt service pending clear register	Undef.
F_ISPC	0x01E0003C	W	FIQ interrupt service pending clear register	Undef.

I_ISPC/F_ISPC	Bit	Description		Initial State
EINT0	[25]	0 = No change	1 = clear the pending bit	0
EINT1	[24]	0 = No change	1 = clear the pending bit	0
EINT2	[23]	0 = No change	1 = clear the pending bit	0
EINT3	[22]	0 = No change	1 = clear the pending bit	0
EINT4/5/6/7	[21]	0 = No change	1 = clear the pending bit	0
INT_TICK	[20]	0 = No change	1 = clear the pending bit	0
INT_ZDMA0	[19]	0 = No change	1 = clear the pending bit	0
INT_ZDMA1	[18]	0 = No change	1 = clear the pending bit	0
INT_BDMA0	[17]	0 = No change	1 = clear the pending bit	0
INT_BDMA1	[16]	0 = No change	1 = clear the pending bit	0
INT_WDT	[15]	0 = No change	1 = clear the pending bit	0
INT_UERR0/1	[14]	0 = No change	1 = clear the pending bit	0
INT_TIMER0	[13]	0 = No change	1 = clear the pending bit	0
INT_TIMER1	[12]	0 = No change	1 = clear the pending bit	0
INT_TIMER2	[11]	0 = No change	1 = clear the pending bit	0
INT_TIMER3	[10]	0 = No change	1 = clear the pending bit	0
INT_TIMER4	[9]	0 = No change	1 = clear the pending bit	0
INT_TIMER5	[8]	0 = No change	1 = clear the pending bit	0
INT_URXD0	[7]	0 = No change	1 = clear the pending bit	0
INT_URXD1	[6]	0 = No change	1 = clear the pending bit	0
INT_IIC	[5]	0 = No change	1 = clear the pending bit	0
INT_SIO	[4]	0 = No change	1 = clear the pending bit	0
INT_UTXD0	[3]	0 = No change	1 = clear the pending bit	0
INT_UTXD1	[2]	0 = No change	1 = clear the pending bit	0
INT_RTC	[1]	0 = No change	1 = clear the pending bit	0
INT_ADC	[0]	0 = No change	1 = clear the pending bit	0

12 LCD CONTROLLER

OVERVIEW

The LCD controller within S3C44B0X 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. It can support 8-bit per pixel (256 level color) for interfacing with a color LCD panel, also.

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

— Supports color/gray/monochrome LCD panels.

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

— Supports Multiple Virtual Display Screen. (Supports Hardware Horizontal/Vertical Scrolling)

— The system memory is used as the display memory.

— Dedicated DMA supports to fetch the image data from video buffer located in system memory.

— Supports multiple screen size.
Typical actual screen sizes: 640x480, 320x240, 160x160 (pixels)
Maximum virtual screen sizes(color mode): 4096x1024, 2048x2048, 1024x4096, etc

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

— Supports 256 level colors for color STN LCD panel.

— Supports the power saving mode(SL_IDLE Mode).

EXTERNAL INTERFACE SIGNAL

VFRAME: This is the frame synchronous signal between the LCD controller and LCD driver. It signals the LCD panel of the start of a new frame. The LCD controller asserts VFRAME after a full frame of display as shown in Fig. 12-3.

VLINE: This is the line synchronous pulse signal between LCD controller and LCD driver, and it is used by the LCD driver to transfer the contents of it's horizontal line shift register to the LCD panel for display. The LCD controller asserts VLINE after an entire horizontal line of data has been shifted into the LCD driver.

VCLK: This pin is the pixel clock signal between the LCD controller and LCD driver, and data is sent by the LCD controller on the rising edge of VCLK and sampled by LCD driver on the falling edge of VCLK.

VM: This is the AC signal for the LCD driver. 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 VM signal can be toggled on every frame or toggled on the programmable number of the VLINE signal.

VD[3:0]: These are LCD pixel data output ports. For a 4-bit or 8-bit single scan display, these 4-bit data are used as the display data as shown in Fig. 12-4. In case of 4-bit dual scan display, these

4-bit

plays into its role of the upper display data as shown in Fig. 12-4.

VD[7:4]: These are LCD pixel data output ports. For a 8-bit single scan display, these data are used as upper dispaly data as shown in Fig. 12-4. For a 4-bit dual scan display, these data are used as lower display data as shown in Fig. 12-4.

BLOCK DIAGRAM

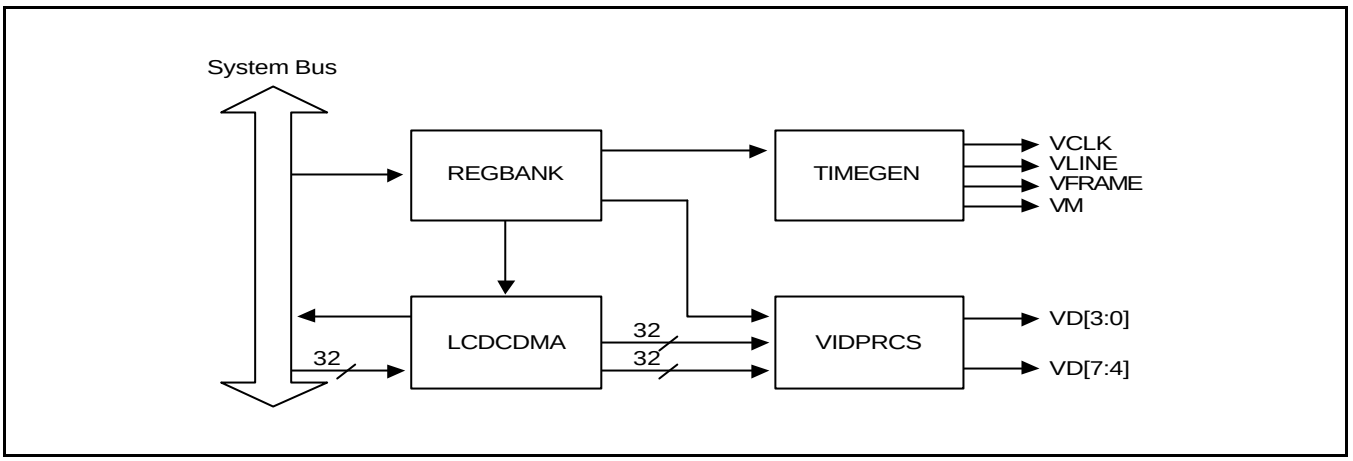


Figure 12-1. LCD Controller Block Diagram

The LCD controller within S3C44B0X is used to transfer the video data and to generate the necessary control signals such as, VFRAME, VLINE, VCLK, and VM. As well as the control signals, S3C44B0X has the data ports of video data, which are VD[7:0] as shown in Fig. 12-1. The LCD controller consists of a REGBANK, LCDCDMA, VIDPRCS, and TIMEGEN (See Figure 12-1 LCD Controller Block Diagram). The REGBANK has 18 programmable register sets 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[7:0] 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. The S3C44B0X 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

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 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[7:0] field of LCDSADDR 2 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 number of VLINE signal by the MVAL[7:0] value. Figure 12-3 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 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 case of 4-bit dual scan display the number of valid VD data line should be 4 and in case of 8-bit signal 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 12-1 defines the relationship of VCLK and CLKVAL. The minimum value of CLKVAL is 2.

VCLK(Hz)=MCLK/(CLKVAL x 2)

The frame rate is the VFRAM signal frequency. The frame rate is closely related to the field of WLH(VLINE pulse width), WHLY(the delay width of VCLK after VLINE pulse), HOZVAL, VLINEBLANK, and LINEVAL in LCDCON1 and LCDCON2 registers as well as VCLK and MCLK. 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/MCLK) x (WLH+WDLY+LINEBLANK)) x (LINEVAL+1)]

VCLK(Hz) = (HOZVAL+1) / [(1 / (frame_rate x (LINEVAL+1))) - ((WLH+WDLY+LINEBLANK) / MCLK)]

Table 12-1. Relation between VCLK and CLKVAL(MCLK=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 S3C44B0X supports 8-bit color mode(256 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 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 S3C44B0X can support the palette 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 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 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.

Gray Mode Operation

Two gray modes are supported by the LCD controller within the S3C44B0X: 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 BULEVAL[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.

Color Mode Operation

The LCD controller in S3C44B0X 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.

DITHERING AND FRC (FRAME RATE CONTROL)

The DITHFRC block has two functions, such as a Time-based Dithering Algorithm for reducing flicker and FRC(Frame Rate Control) for displaying gray level on the STN panel. The main principle of gray 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 12-2. To represent the 14th 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 12-2.

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 15th gray level should always have pixel-on, and the 14th gray level should have 6 times pixel-on and one times pixel-off, and the 13th gray level should have 4 times pixel-on and one times pixel-off, ,,,,,,,, , and the 0th gray level should always have pixel-off as shown in Table 12-2. In Table 12-3, the DP1_2 corresponds to the 7th gray level because it has half the duty cycle from having 2 times pixel-on and 2 times pixel-off. Also, the DP4_7 corresponds to 8th gray level because it has (4/7) duty cycle from having 4 times pixel-on and 3 times pixel-off. Using the same methodology, the DP3_5, DP2_3, DP5_7, DP3_4, DP4_5, and DP6_7 are made to correspond to 9th, 10th, 11th, 12th, 13th, and 14th gray level, respectively. For the gray level from 1st to 6th, the reverse sequence of DP6_7, DP4_5, DP3_4, DP2_3, DP3_5, and DP4_7 should be used; this way, new tables for gray level of 1st to 6th are not needed. The Table 12-7 shows that the same pixel value can not have the same FRC sequence. For example, if the Pi pixel has half gray level in Nth frame, and if adjacent pixel of Pi+1 also has half gray level in Nth frame, and if adjacent pixel of Pi+2 also has half gray level in Nth frame, and if adjacent pixel of Pi+3 also has half gray level in Nth frame, the Pi, Pi+1, Pi+2, and Pi+3 pixel should be 1, 0, 1, and 0 in Nth frame. In (N+1)th frame, the Pi, Pi+1, Pi+2, and Pi+3 pixel should be 0, 1, 0, and 1 as shown in Table 12-3. In case of arbitrary pixel values on arbitrary position, the H/W will select a suitable display value by referring to the corresponding frame number and pixel position. This type of display methodology can randomize the pixel display to reduce the Flicker Noise. The value of table 12-3 is just only reference, and users can specify their own value suitable for the LCD display.

Table 12-2. 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

Pixel Duty Rate

S3C44B0X has eight programmable registers, such as DP6_7, DP4_5, DP5_7, DP3_4, DP2_3, DP3_5, DP4_7, and

DP1_2. The pre-dithered data 1111b has a dithering data '1' because the duty rate is 1. The pre-dithered data 0000b has a dithering data 0 because the duty rate is 0. The pre-dithered data from 0001b to 1110b refer to DP6_7, DP4_5, DP5_7, DP3_4, DP2_3, DP3_5, DP4_7, and DP1_2 registers for dithering data.(The dithering data are used to do FRC.)

The DP6_7, DP4_5, DP5_7, DP3_4, DP2_3, DP3_5, DP4_7, and DP1_2, registers can also determine the duty rates, such as 6/7, 4/5, 5/7, 3/4, 2/3, 3/5, and 4/7, respectively. For examples, 1/7 can be made by inverting 6/7.

Table 12-3. Recommended Dithering Pattern

Pattern Name	Number of Bits	Recommened Pattern
DP1_2	16	1010 0101 1010 0101 (0xA5A5)
DP4_7	28	1011 1010 0101 1101 1010 0110 0101 (0xBA5DA65)
DP3_5	20	1010 0101 1010 0101 1111 (0xA5A5F)
DP2_3	12	1101 0110 1011 (0xD6B)
DP5_7	28	1110 1011 0111 1011 0101 1110 1101 (0xEB7B5ED)
DP3_4	16	0111 1101 1011 1110 (0x7DBE)
DP4_5	20	0111 1110 1011 1101 1111 (0x7EBDF)
DP6_7	28	0111 1111 1101 1111 1011 1111 1110 (0x7FDFBFE)

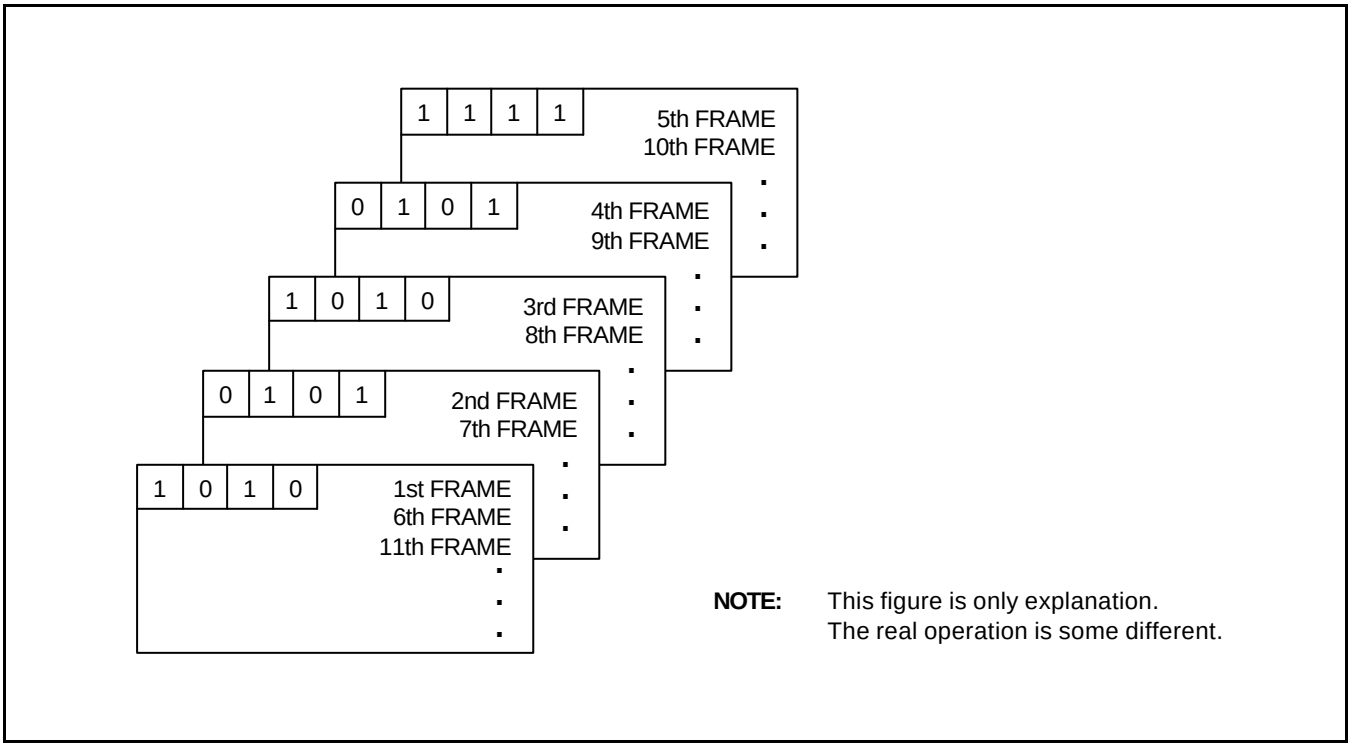


Figure 12-2. The example of DP3_5 pattern

LCD Self Refresh Mode

The LCD controller within S3C44B0X can support the self refresh mode to reduce power comsumption. 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 display memory, which can display the previous stored image in the built-in display memory without image data fetch when the self refresh mode has been invoked. The kind of self refresh mode can be made by writing the control bit of SELFREF in the LCDCON3 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 VD value, but the signal of VM, VFRAME, and VLINE will be generated continuously. To exit the self refresh mode, the user should execute the following path, 1) disable the ENVID bit in LCDCON 1 register , 2) disable SELFREF bit in LCDCON 3 register and 3) enable ENVID bit again in LCDCON 1 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 S3C44B0X should be stopped to reduce the power comsumption, because the power management block inserts divide_by_n input clock only to the LCD controller.

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 12-3 shows the timing requirements for the LCD driver interface.

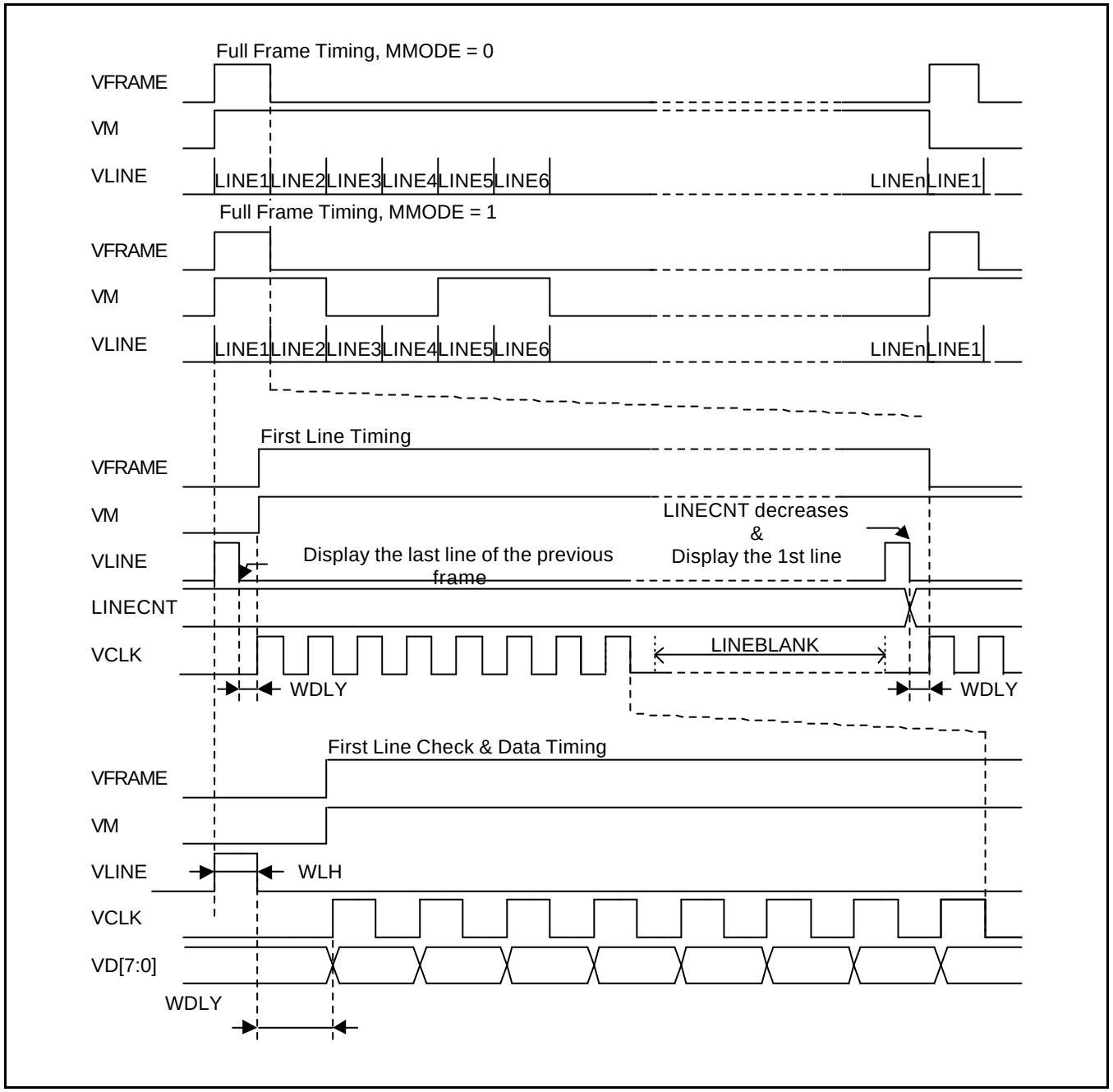


Figure 12-3. 8-bit Single Scan Display Type LCD Timing

Display Types

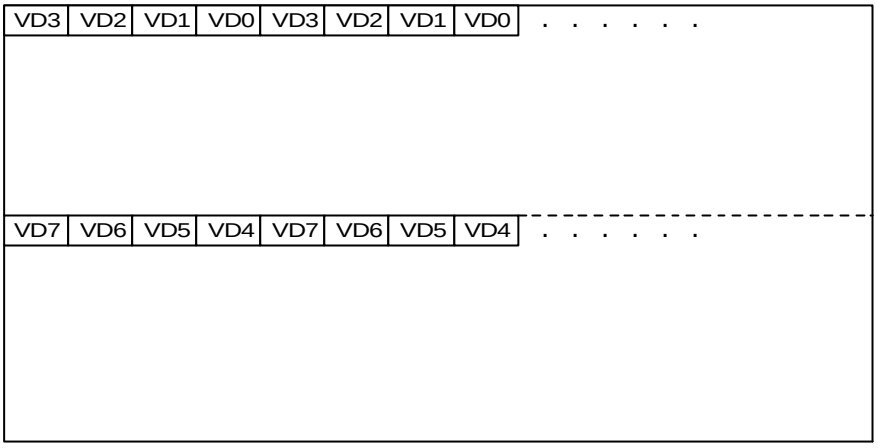
The LCD controller supports 3 types of LCD drivers: 4-bit dual scan, 4-bit single scan, and 8-bit single scan display mode. Figure 12-4 shows these 3 different display types for monochrome displays, and figure 12-5 shows these 3 different display types for color displays.

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 12-4. 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.

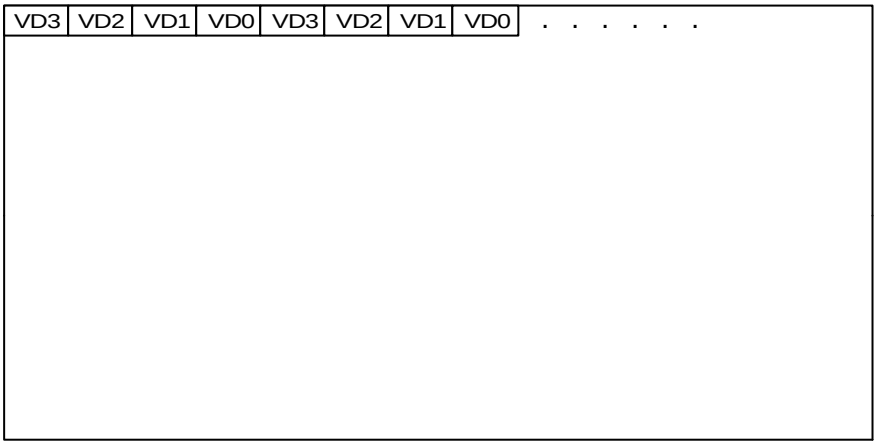
4-bit single scan display type
A 4-bit single scan display uses 4 parallel data lines to shift data to successive single horizontal lines of the display at a time, until the entire frame has been shifted and transferred. The 4 pins(VD[3:0]) for the LCD output from the LCD controller can be directly connected 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.

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 12-5 shows the RGB and order of the pixels in the parallel data lines for the 3 types of color displays.



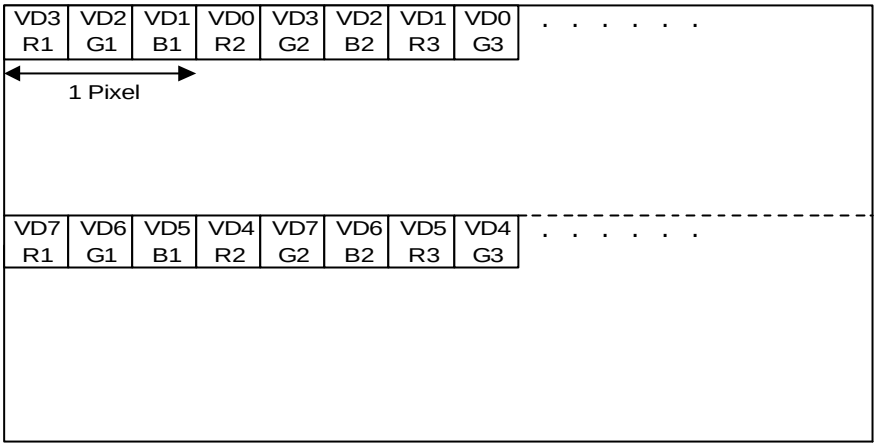
4-bit Dual Scan Display



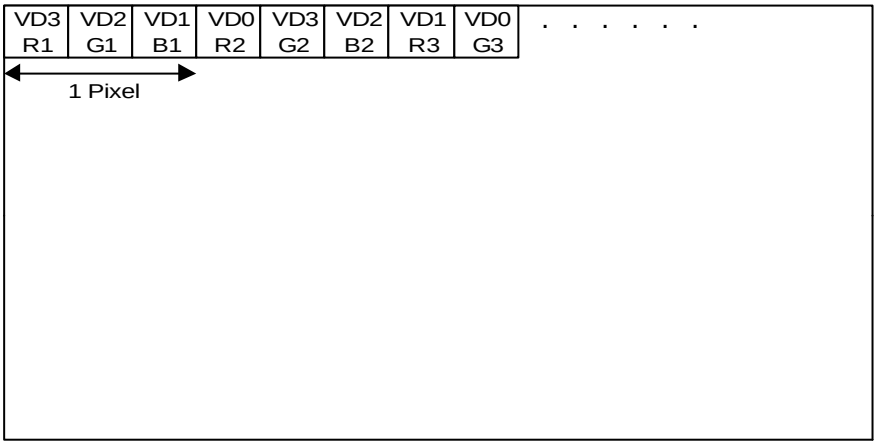
4-bit Single Scan Display

8-bit Single Scan Display

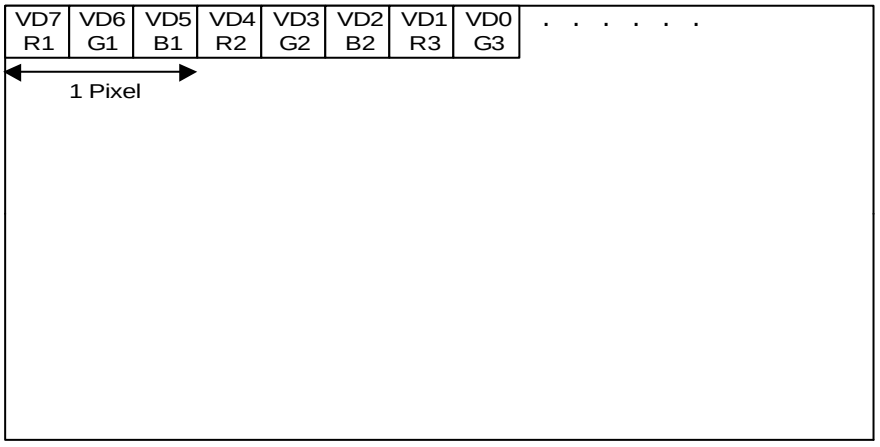
Figure 12-4. Monochrome Display Types



4-bit Dual Scan Display



4-bit Single Scan Display



8-bit Single Scan Display

Figure 12-5. Color Display Types

MEMORY DATA FORMAT (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 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

VIRTUAL DISPLAY

The S3C44B0X 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 Fig. 12-6) 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.

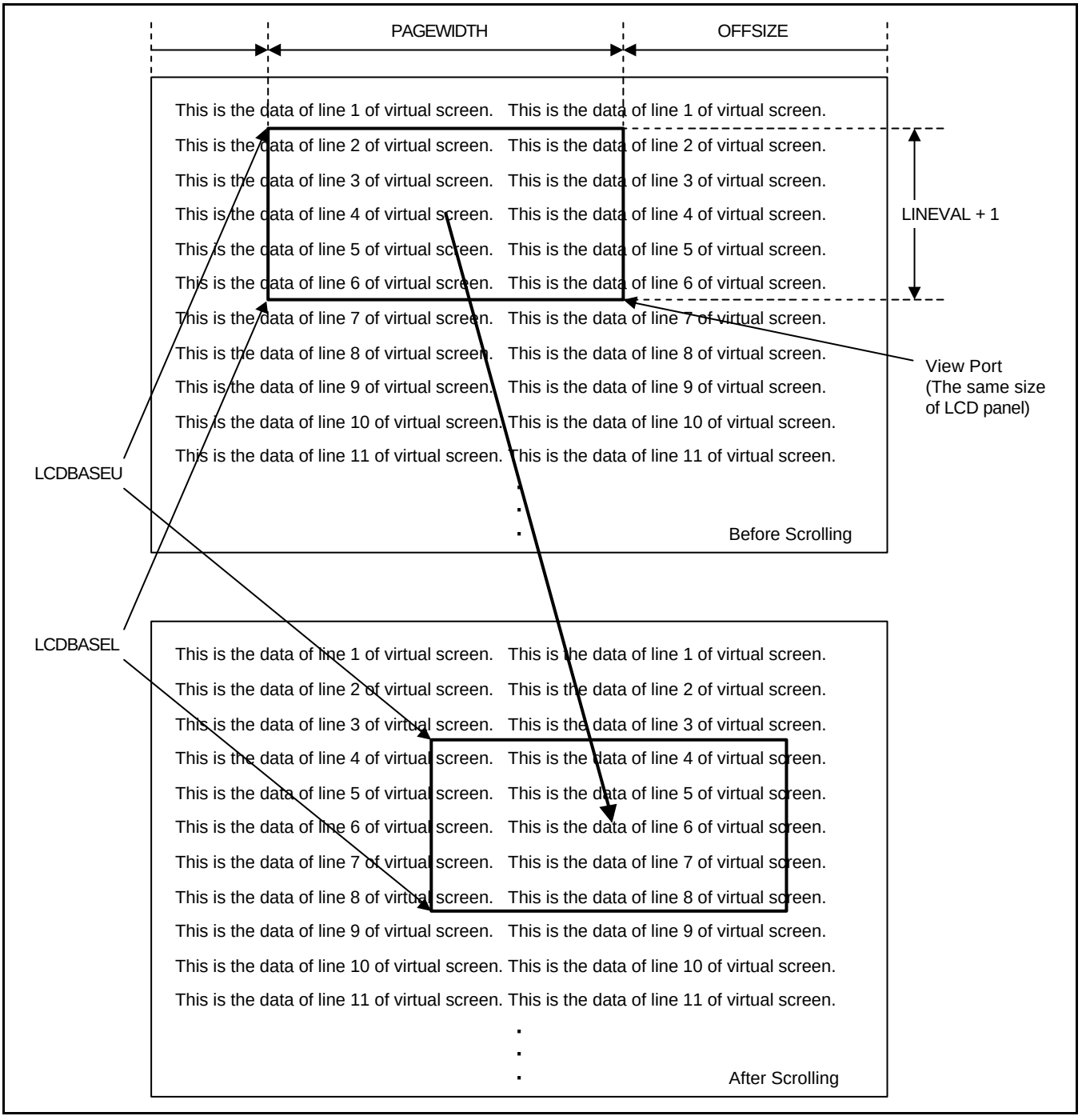


Figure 12-6. Example of Scrolling in Virtual Display(single scan)

LCD CONTROLLER SPECIAL REGISTERS

LCD Control 1 Register

Register	Address	R/W	Description	Reset Value
LCDCON1	0x01F00000	R/W	LCD control 1 register	0x00000000

LCDCON1	Bit	Description	Initial State
LINECNT (read only)	[31:22]	These bits provide the status of the line counter. Down count from LINEVAL to 0	0000000000
CLKVAL	[21:12]	These bits determine the rate of VCLK. If this value can be changed when ENVID=1, the new value will be used next frame. *VCLK = MCLK / (CLKVAL x 2) (CLKVAL ³ 2)*	0000000000
WLH	[11:10]	These bits determine the VLINE pulse's high level width by counting the number of the system clock. 00 = 4 clock, 01 = 8 clock, 10 = 12 clock, 11 = 16 clock	00
WDLY	[9:8]	These bits determine the delay between VLINE and VCLK by counting the number of the system clock 00 = 4clock, 01 = 8 clock, 10 = 12 clock, 11 = 16 clock	00
MMODE	[7]	This bit determines the toggle rate of the VM. 0 = Each Frame, 1 = The rate defined by the MVAL	0
DISMODE	[6:5]	These bits select the display mode. 00 = 4-bit dual scan display mode 01 = 4-bit single scan display mode 10 = 8-bit single scan display mode 11 = Not used	00
INVCLK	[4]	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
INVLINE	[3]	This bit indicates the line pulse polarity. 0 = normal 1 = inverted	0
INVFRAME	[2]	This bit indicates the frame pulse polarity. 0 = normal 1 = inverted	0
INVVD	[1]	This bit indicates the video data(VD[7:0]) polarity. 0 = Normal 1 = VD[7:0] output is inverted.	0
ENVID	[0]	LCD video output and the logic enable/disable. 0 = Disable the video output and the logic. The LCD FIFO is cleared. 1 = Enable the video output and the logic.	0

LCD Control 2 Register

Register	Address	R/W	Description	Reset Value
LCDCON2	0x01F00004	R/W	LCD control 2 register	0x00000000

LCDCON2	Bit	Description	Initial State
LINEBLANK	[31:21]	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 MCLK. Ex) If the value of LINEBLANK is 10, the blank time is inserted to VCLK during 10 system clocks.	0x000
HOZVAL	[20:10]	These bits determine the horizontal size of the LCD panel. HOZVAL has to be determined to meet the condition that total bytes of 1 line be 2n bytes. If the x size of LCD is 120 dots in mono mode, x=120 can not be supported because 1 line consists of 15 bytes. Instead, x=128 in mono mode can be supported because 1 line consists of 16 bytes(2n). The additional 8 dot will be discarded by LCD panel driver.	0x000
LINEVAL	[9:0]	These bits determine the vertical size of LCD panel.	0x000

LCD Control 3 Register

Register	Address	R/W	Description	Reset Value
LCDCON3	0x01F00040	R/W	Test Mode Enable Register	0x00

LCDCON3	Bit	Description	initial state
Reserved	[2:1]	reserved for test	0
SELFREF	[0]	LCD self refresh mode enable bit 0 : LCD self refresh mode disable 1 : LCD self refresh mode enable	0

FRAME Buffer Start Address 1 Register

Register	Address	R/W	Description	Reset Value
LCDSADDR1	0x01F00008	R/W	Frame buffer start address 1 register	0x000000

LCDSADDR1	Bit	Description	Initial State
MODESEL	[28:27]	These bits select the monochrome, gray, or color mode. 00 = monochrome mode 01 = 4-level gray mode 10 = 16-level gray mode 11 = color mode	00
LCDBANK	[26:21]	These bits indicate A[27:22] 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 should not be changed when moving the view port. So, using the malloc function the care should be taken.	0x00
LCDBASEU	[20:0]	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.	0x000000

NOTES:

1. LCDBANK can't be changed while ENVID=1
2. If LCDBASEU,LCDBASEL is changed during ENVID=1, the new value will be used next frame. If you use serveral frame buffer for better display quality and if you write the previous frame memory just after changing LCDBASEU,LCDBASEL, the items drawn on the previous frame memory may be shown. To avoid this undesirable phenomen, you may have to check LINECNT.

FRAME Buffer Start Address 2 Register

Register	Address	R/W	Description	Reset Value
LCDSADDR2	0x01F0000C	R/W	Frame buffer start address 2 register	0x000000

LCDSADDR2	Bit	Description	Initial State
BSWP	[29]	Byte swap control bit 1 : Swap Enable 0 : Swap Disable LCD DMA fetches the frame memory data by 4 word burst access. In little endian mode and BSWP is 0, the frame memory data are displayed in the sequence, 4n+3th, 4n+2th ,4n+1th ,4n-th data. If BSWP is 1, the sequence will be 4n-th, 4n+1th, 4n+2th, 4n+3th. If the CPU is little endian mode, the frame buffer may be accessed by only byte access mode, Because BSWP is 1, the byte accessed data will be shown correctly also in the little endian mode. In the other case, BSWP has to be 0.	0
MVAL	[28:21]	These bits define the rate at which the VM signal will toggle if the MMODE bit is set to logic '1'.	0x00
LCDBASEL	[20:0]	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. LCDBASEL = 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, interrutpt 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	0x01F00010	R/W	Virtual screen address set	0x000000

LCDSADDR3	Bit	Description	Initial State
OFFSIZE	[19:9]	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.	0x0000
PAGEWIDTH	[8:0]	Virtual screen page width(the number of half words) This value defines the width of the view port in the frame	0x000

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	0x01F00014	R/W	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	0x01F00018	R/W	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	0x01F0001C	R/W	Blue lookup table register	0x0000

BULELUT	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

Dithering Pattern DP1_2 Register

Register	Address	R/W	Description	Reset Value
DP1_2	0x01F00020	R/W	Dithering pattern duty 1/2 register (Please, refer to a sample program source for the latest value of this register).	0xa5a5

DP1_2	Bit	Description	Initial state
DP1_2	[15:0]	Recommended pattern value 1010 0101 1010 0101 (0xa5a5)	0xa5a5

Dithering Pattern DP4_7 Register

Register	Address	R/W	Description	Reset Value
DP4_7	0x01F00024	R/W	Dithering pattern duty 4/7 register (Please, refer to a sample program source for the latest value of this register).	0xba5da65

DP4_7	Bit	Description	Initial state
DP4_7	[27:0]	Recommended pattern value 1011 1010 0101 1101 1010 0110 0101 (0xba5da65)	0xba5da65

Dithering Pattern DP3_5 Register

Register	Address	R/W	Description	Reset Value
DP3_5	0x01F00028	R/W	Dithering pattern duty 3/5 register (Please, refer to a sample program source for the latest value of this register).	0xa5a5f

DP3_5	Bit	Description	Initial state
DP3_5	[19:0]	Recommended pattern value 1010 0101 1010 0101 1111 (0xa5a5f)	0xa5a5f

Dithering Pattern DP2_3 Register

Register	Address	R/W	Description	Reset Value
DP2_3	0x01F0002C	R/W	Dithering pattern duty 2/3 register (Please, refer to a sample program source for the latest value of this register).	0xd6b

DP2_3	Bit	Description	Initial state
DP2_3	[11:0]	Recommended pattern value 1101 0110 1011 (0xd6b)	0xd6b

Dithering Pattern DP5_7 Register

Register	Address	R/W	Description	Reset Value
DP5_7	0x01F00030	R/W	Dithering pattern duty 5/7 register (Please, refer to a sample program source for the latest value of this register).	0xeb7b5ed

DP5_7	Bit	Description	Initial state
DP5_7	[27:0]	Recommended pattern value 1110 1011 0111 1011 0101 1110 1101 (0xeb7b5ed)	0xeb7b5ed

Dithering Pattern DP3_4 Register

Register	Address	R/W	Description	Reset Value
DP3_4	0x01F00034	R/W	Dithering pattern duty 3/4 register (Please, refer to a sample program source for the latest value of this register).	0x7dbe

DP3_4	Bit	Description	Initial state
DP3_4	[15:0]	Recommended pattern value 0111 1101 1011 1110 (0x7dbe)	0x7dbe

Dithering Pattern DP4_5 Register

Register	Address	R/W	Description	Reset Value
DP4_5	0x01F00038	R/W	Dithering pattern duty 4/5 register (Please, refer to a sample program source for the latest value of this register).	0x7ebdf

DP4_5	Bit	Description	Initial state
DP4_5	[19:0]	Recommended pattern value 0111 1110 1011 1101 1111 (0x7ebdf)	0x7ebdf

Dithering Pattern DP6_7 Register

Register	Address	R/W	Description	Reset Value
DP6_7	0x01F0003C	R/W	Dithering pattern duty 6/7 register (Please, refer to a sample program source for the latest value of this register).	0x7fdfbfe

DP6_7	Bit	Description	initial state
DP6_7	[27:0]	Recommended pattern value 0111 1111 1101 1111 1011 1111 1110 (0x7fdfbfe)	0x7fdfbfe

Dithering Mode Register

Register	Address	R/W	Description	Reset Value
DITHMODE	0x01F00044	R/W	Dithering Mode Register. This register reset value is 0x00000. But, users will have to change this value to 0x12210. (Please, refer to a sample program source for the latest value of this register).	0x00000

DP8_9	Bit	Description	initial state
DITHMODE	[18:0]	Use one of following value for your LCD 0x12210 or 0x0	0x00000

Register Setting Guide

The maximum VCLK frequency of the LCD controller is 16.5MHz whenever system clock frequency is 66 MHz; therefore the LCD controller supports all existing LCD drivers. 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 12-4. 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 160pixel, 4-level gray, 80 frame/sec, 4-bit single scan display, system clock frequency = 66 MHz,
WLH = 1, WDLY = 1, LCD frame buffer = SDRAM, Bus width = 16bit.

System bus occupation = (LCD data transmission frequency) / (System clock frequency)

LCD data transmission frequency = (Total LCD data during 1sec) x (Transmission cycle / 1byte)

Total LCD data during 1sec = Total LCD data x frame rate
= 160 x 160pixel x (2bit / 1pixel) x 80Hz x (1byte / 8bit) = 512Kbyte

Transmission cycle per 1byte = Transmission clock per 4word / 16

Transmission clock per 4word = Trp(=2clk) + Trcd(=2clk) + C/L(=2clk) + Burst cycle(=8clk) = 14clk

System load (system bus occupation) : 448KHz / 66MHz = 0.68%

NOTE: The higher the system load is, the lower the CPU performance is.

Example 2 (Virtual screen register) :

4 -level gray, 4-bit single scan display, Vertual screen size = 1024 x 1024, LCD size = 320 x 240,
LCDBASEU = 0x64.

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 240 = 0x3C64

Gray Level Selection Guide

S3C44B0X LCD 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 may have 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. If some gray level quality is not good, select the good gray levels, which is displayed well on your LCD.

NOTES

13 A/D CONVERTER

OVERVIEW

The 10-bit CMOS ADC(Analog to Digital Converter) of S3C44B0X consists of a 8-channel analog input multiplexer, auto-zeroing comparator, clock generator, 10 bit successive approximation register (SAR), and output register. This ADC provides software-selection power-down(sleep) mode.

FEATURES

- Resolution: 10-bit
- Differential Linearity Error: ± 1 LSB
- Integral Linearity Error: ± 2 LSB (Max. ± 3 LSB)
- Maximum Conversion Rate: 100 KSPS
- Input voltage range: 0-2.5V
- Input bandwidth: 0-100 Hz (without S/H(sample&hold) circuit)
- Low Power Consumption

A/D CONVERTER OPERATION

BLOCK DIAGRAM

Figure 13-1 shows the functional block diagram of S3C440BX A/D converter. Note that the reference positive voltage REFT and reference negative voltage RETB are applied internally by A/D converter power supply and ground, so no power is applied to REFT and REFB pins. Also REFT, REFB, and analog common voltage VCOM should be connected to bypass capacitors respectively because of voltage level stability.

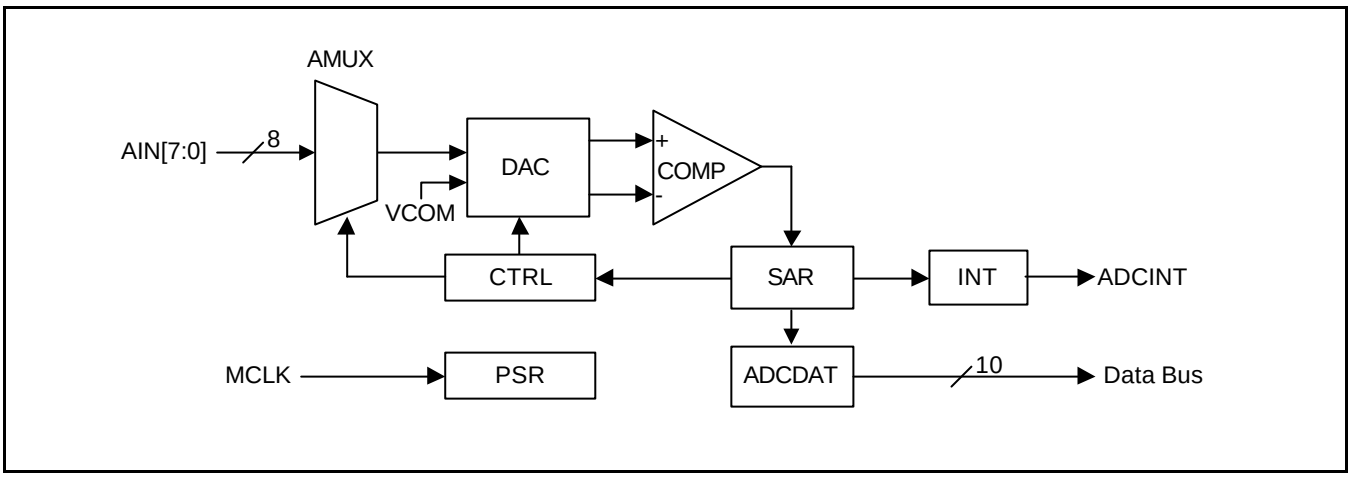


Figure 13-1. A/D Converter Block Diagram

FUNCTION DESCRIPTIONS

SAR (Successive Approximation Register) A/D Converter Operation

A SAR type A/D converter basically consists of the comparator, D/A converter, and SAR logic. At the beginning of the conversion, the MSB is switched ON and the analog input signal is compared the reference signal of D/A converter. Because the A/D converter was designed with differential architecture, D/A converter generates the differential reference signal internally and the two difference signals - one signal is the difference between analog input and positive reference signal, the other is between the analog common voltage(VCOM) and negative reference signal - are delivered to comparator. The comparator then compares the analog input with the reference signal differentially. When the input signal is larger than the reference, then MSB remains ON and the next bit is switched ON and a comparison will be performed. A bit by bit operation is in this system bring the reference signal within 1 LSB of the time discrete input signal.

A/D Conversion Time

When the system clock frequency is 66MHz and the prescaler value is 20, total 10-bit conversion time is as follows.

66 MHz / 2(20+1) / 16(at least 16 cycle by 10-bit operation) = 98.2 KHz = 10.2 us

NOTE: Because this A/D converter has no sample-and-hold circuit, analog input frequency should not exceed 100Hz for accurate conversion although the maximum conversion rate is 100KSPS.

Sleep Mode

The ADC sleep mode is activated by setting the SLEEP bit, ADCCON[5], to '1'. In this mode, the conversion clock is disabled and A/D conversion operation is halted. The A/D converter data register contains the previous data in sleep mode.

NOTE: After the ADC exits the sleep mode(ADCCON[5]=1? 0), there is 10ms wait for the ADC reference voltage stabilization before the first AD conversion.

ADC reference pin configuration

Users must configure S3C44B0X' s reference pins(83, 84, 85) as shown in Fig.13-2.

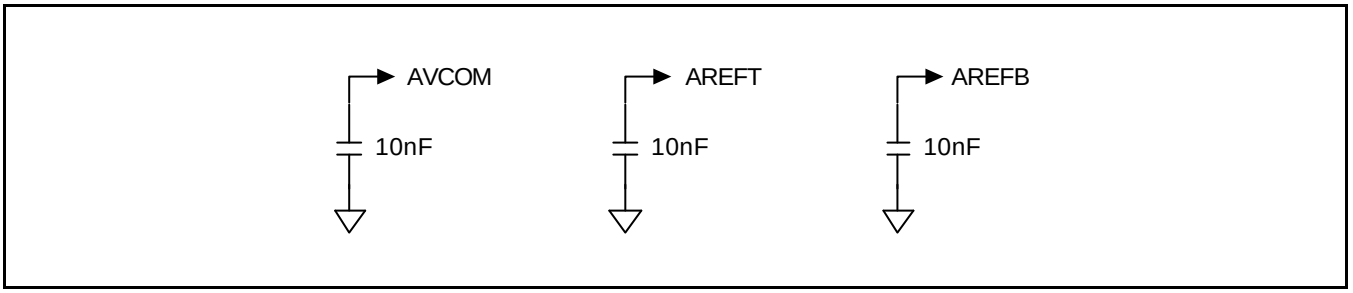


Figure 13-2. External reference pin configuration

Workaround For the ADC Data Reading Problem

The ADC converter state flag(ADCCON[6], FLAG bit) is not correct. The FLAG operates incorrectly in the following cases:

a) The FLAG will be 1 for one ADC clock time just after the ADC conversion is started. This is not correct.
b) The FLAG will be 1 one ADC clock time ago than the ADC conversion is completed. This is not correct.

This problem will be shown conspicuously only if the ADCPSR is large. To read ADC converted data correctly , please refer to the following codes;

```
rADCCON=0x1|(0x0<<2);          //Start A/D conversion
while(rADCCON &0x1);           //To avoid The first FLAG error case.
                               //(The START bit is cleared in one ADC clock.)
while(!(rADCCON & 0x40));
for(i=0;i<rADCPSR;i++);        //To avoid The second FLAG error case
Uart_Printf("A0=%03xh ",rADCDAT);
```

The Programming Technique in ADC

1. There is no sample & hold circuit on the ADC input pin. So, The small current will flow in/out from AINn input pins because of the ADC internal operation. If the output impedance of source signal is high, this current will change the signal voltage. The current is about 7.6uA in the following condition.

Condition	100KSPS, 10K-ohm resister, Vsource=0.0V
Induced Current	7.8uA (78mV)
Induced ADC Error	32

   1) This ADC error will be decreased if the output impedance of the signal source is reduced. For example, If the output impedance of the signal source is 1Kohm, the induced ADC error by the ADC input current is 3(1/10).
   2) The current will be also decreased if ADCPSR is large. If the ADC conversion rate is 30KSPS, the current will be about 1.2uA. The ADCPSR value is higher, the current is lower.

2. The ADC conversion error is decreased if the ADCPSR is large beside the above ADC conversion error. If you want accurate ADC conversion, you let the ADCPSR as large as possible.
3. Because our ADC have no sample&hold circuit, the input frequency bandwidth is 0~100Hz. This limitation is because there is no internal sample&hold circuit. But, If you can ignore the small ADC error(or an external S/H circuit is used), the higher frequency signal can be converted.
4. If the ADC channel is changed, the channel setup time(min. 15us) is needed. So, If the ADC channel is changed, you must wait for 15us and then start AD conversion.
5. After the ADC exits the sleep mode(the initial state is the sleep mode), there is 10ms wait for the ADC reference voltage stabilization before the first AD conversion.
6. Our ADC has ADC start-by-read feature. This feature can be used for DMA to move the ADC data to memory.
7. If you read the ADCDAT by polling method, you must apply the work-around for ADC data reading problem.

A/D CONVERTER SPECIAL REGISTERS

A/D CONVERTER CONTROL REGISTER (ADCCON)

Register	Address	R/W	Description	Reset Value
ADCCON	0x01D40000(Li/W, Li/HW, Li/B, Bi/W) 0x01D40002(Bi/HW) 0x01D40003(Bi/B)	R/W	A/D Converter control Register	0x20

ADCCON	Bit	Description	Initial State
FLAG	[6]	A/D converter state flag (Read Only). 0 = A/D conversion in process 1 = End of A/D conversion If check this bit please refer to workaround in page13-3.	0
SLEEP	[5]	System power down 0 = Normal operation, 1 = Sleep mode	1
INPUT SELECT	[4:2]	Clock source select 000 = AIN0 001 = AIN1 010 = AIN2 011 = AIN3 100 = AIN4 101 = AIN5 110 = AIN6 111 = AIN7	00
READ_ START	[1]	A/D conversion start by read 0 = Disable start by read operation 1 = Enable start by read operation	00
ENABLE_START	[0]	A/D conversion start by enable. If READ_START is enabled, this value is not valid. 0 = No operation 1 = A/D conversion starts and this bit is cleared after the start-up.	0

NOTES:

1. The ADCCON register can be accessed by halfword and word unit using STRB/STRH/STR and LDRB/LDRH/LDR instructions or char/short int/int type pointer in the Little/Big endian mode.
2. (Li/B/HW/W): Access by char/halfword/word unit when the endian mode is Little.
   (Bi/B/HW/W): Access by char/halfword/word unit when the endian mode is Big.

A/D CONVERTER PRESCALER REGISTER (ADCPSR)

Register	Address	R/W	Description	Reset Value
ADCPSR	0x01D40004(Li/W, Li/HW, Li/B, Bi/W) 0x01D40006(Bi/HW) 0x01D40007(Bi/B)	R/W	A/D Converter prescaler Register	0x0

ADCPSR	Bit	Description	Initial State
PRESCALER	[7:0]	Prescaler value (0-255) Division factor = 2 (prescaler_value+1). Total clocks for ADC converstion = 2*(Prescalser_value+1)*16	0

NOTES:
1. The ADCPSR register can be accessed by halfword and word unit using STRB/STRH/STR and LDRB/LDRH/LDR instructions or char/short int/int type pointer in Little/Big endian mode.
2. (Li/HW/W): Access by char/halfword/word unit when the endian mode is Little.
   (Bi/HW/W): Access by char/halfword/word unit when the endian mode is Big.

A/D CONVERTER DATA REGISTER (ADCDAT)

After A/D conversion is completed, the ADCDAT reads the converted data. ADCDAT has to be read after the conversion has been completed.

Register	Address	R/W	Description	Reset Value
ADCDAT	0x01D40008(Li/W, L/HW, Bi/W) 0x01D4000A(Bi/HW)	R	A/D converter data register	–

ADCDAT	Bit	Description	Initial State
ADCDAT	[9:0]	A/D converter output data value	–

NOTES:
1. The ADCDAT 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.

14 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.

FEATURES

— 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 (VDDRTC)

— Supports millisecond tick time interrupt for RTOS kernel time tick.

— Round reset function

REAL TIME CLOCK OPERATION

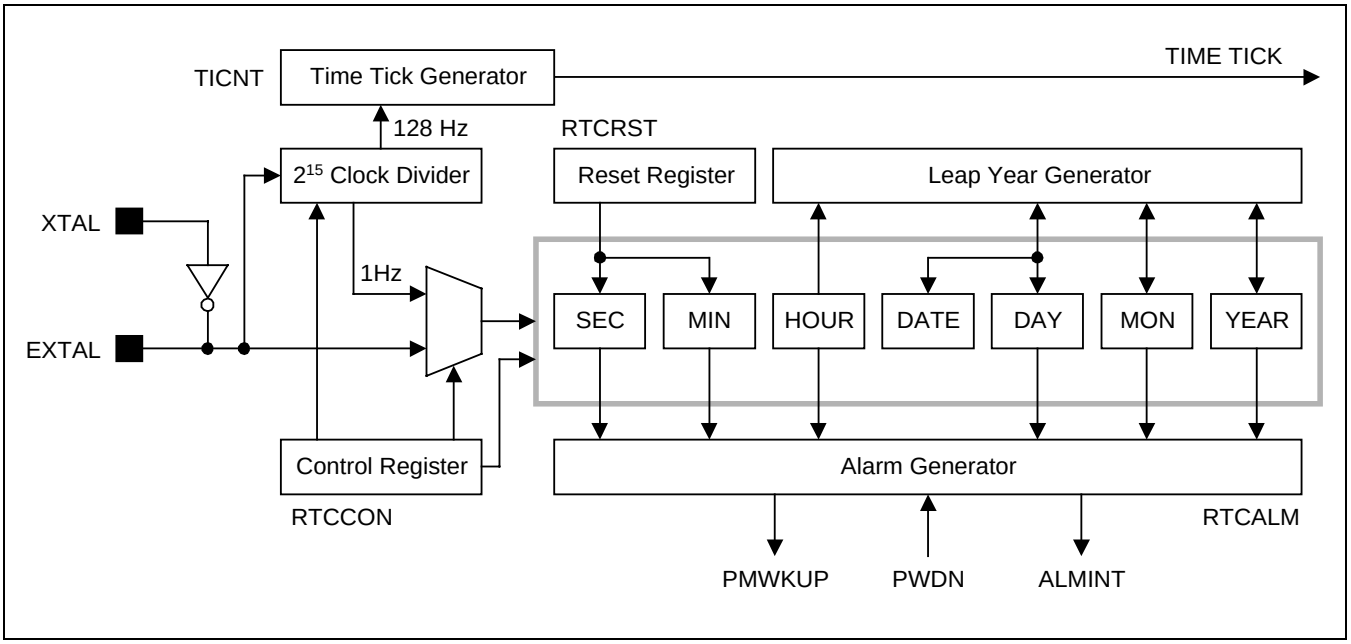


Figure 14-1. Real Time Clock Block Diagram

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 S3C44B0X 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 S3C44B0X denote 2000, not 1900.

READ/WRITE REGISTERS

Bit 0 of the RTCCON register must be set in order to read and write the 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 1959(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 1960(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.

32.768KHZ X-TAL CONNECTION EXAMPLE

The Figure 14-2 is an example circuit of the RTC unit oscillation at 32.768Khz.

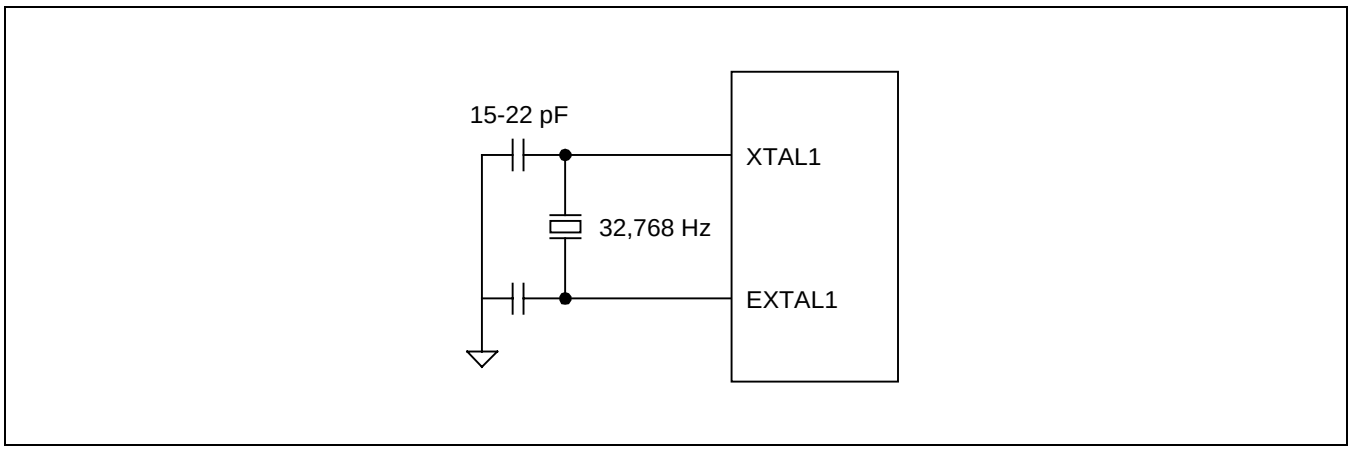


Figure 14-2. Main Oscillator Circuit Examples

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	0x01D70040(L) 0x01D70043(B)	R/W (by byte)	RTC control Register	0x0

RTCCON	Bit	Description	Initial State
CLKRST	[3]	RTC clock count reset 0 = No reset 1 = Reset	0
CNTSEL	[2]	BCD count select 0 = Merge BCD counters 1 = Reserved (Separate BCD counters)	0
CLKSEL	[1]	BCD clock select 0 = XTAL $1/2^{15}$ divided clock 1 = Reserved (XTAL clock only for test)	0
RTCEN	[0]	RTC write enable 0 = Disable 1 = Enable This control bit can reduce the power consumption of RTC in normal mode. If this bit is 0, the CPU cannot write the BCD registers, but these registers can be read out. To reduce RTC current, this bit should be 0 while users do not write to the BCD registers. Although this bit is 0, the RTC block is still running.	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.

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	0x01D70050(L) 0x01D70053(B)	R/W (by byte)	RTC alarm control Register	0x00

RTCALM	Bit	Description	Initial State
Reserved	[7]		0
ALMEN	[6]	Alarm global enable 0 = Disable, 1 = Enable	0
YEAREN	[5]	Year alarm enable 0 = Disable, 1 = Enable	0
MONREN	[4]	Month alarm enable 0 = Disable, 1 = Enable	0
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	0x01D70054(L) 0x01D70057(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	0x01D70058(L) 0x01D7005B(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	0x01D7005C(L) 0x01D7005F(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	0x01D70060(L) 0x01D70063(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	0x01D70064(L) 0x01D70067(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	0x01D70068(L) 0x01D7006B(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	0x01D7006C(L) 0x01D7006F(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. (note) 011 = over than 30 sec 100 = over than 40 sec 101 = over than 50 sec	00

NOTE: Otherwise, no second carry is generated.

BCD SECOND REGISTER (BCDSEC)

Register	Address	R/W	Description	Reset Value
BCDSEC	0x01D70070(L) 0x01D70073(B)	R/W (by byte)	BCD second Register	Undef.

BCDSEC	Bit	Description	Initial State
Reserved	[7]		–
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	0x01D70074(L) 0x01D70077(B)	R/W (by byte)	BCD minute Register	Undef.

BCDMIN	Bit	Description	Initial State
Reserved	[7]		–
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	0x01D70078(L) 0x01D7007B(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	0x01D7007C(L) 0x01D7007F(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	0x01D70080(L) 0x01D70083(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	0x01D70084(L) 0x01D70087(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	0x01D70088(L) 0x01D7008B(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	–

TICK TIME COUNT REGISTER (TICNT)

Register	Address	R/W	Description	Reset Value
TICNT	0x01D7008C(L) 0x01D7008F(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

15 WATCHDOG TIMER

OVERVIEW

The S3C44B0X 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 MCLK cycles.

FEATURES

— Normal interval timer mode with interrupt request

— Internal reset signal is activated for 128 MCLK 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 15-1. The watchdog timer uses MCLK as its only source clock. To generate the corresponding watchdog timer clock, the MCLK frequency is prescaled first, and the resulting frequency is divided again.

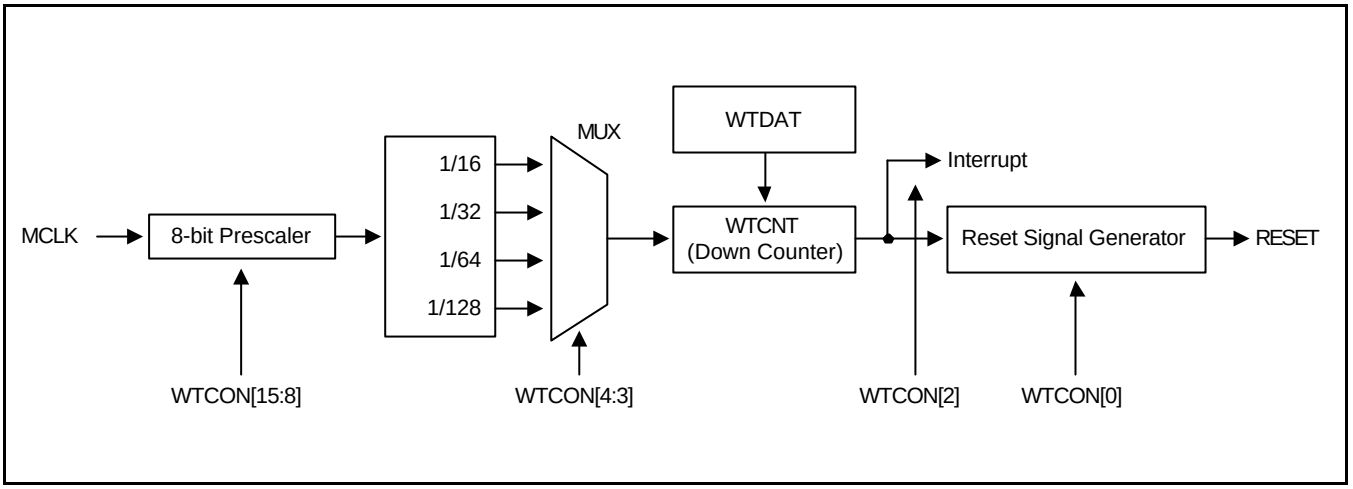


Figure 15-1. Watchdog Timer Block Diagram

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/(MCLK / (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 S3C44B0X 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 S3C44B0X 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	0x01D30000	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: 1/16 01: 1/32 10: 1/64 11: 1/128	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 S3C44B0X 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	0x01D30004	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	0x01D30008	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

16 IIC-BUS INTERFACE

OVERVIEW

The S3C44B0X 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 S3C44B0X RISC microprocessors can receive or transmit serial data to or from slave devices. The master S3C44B0X, 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 S3C44B0X.

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.

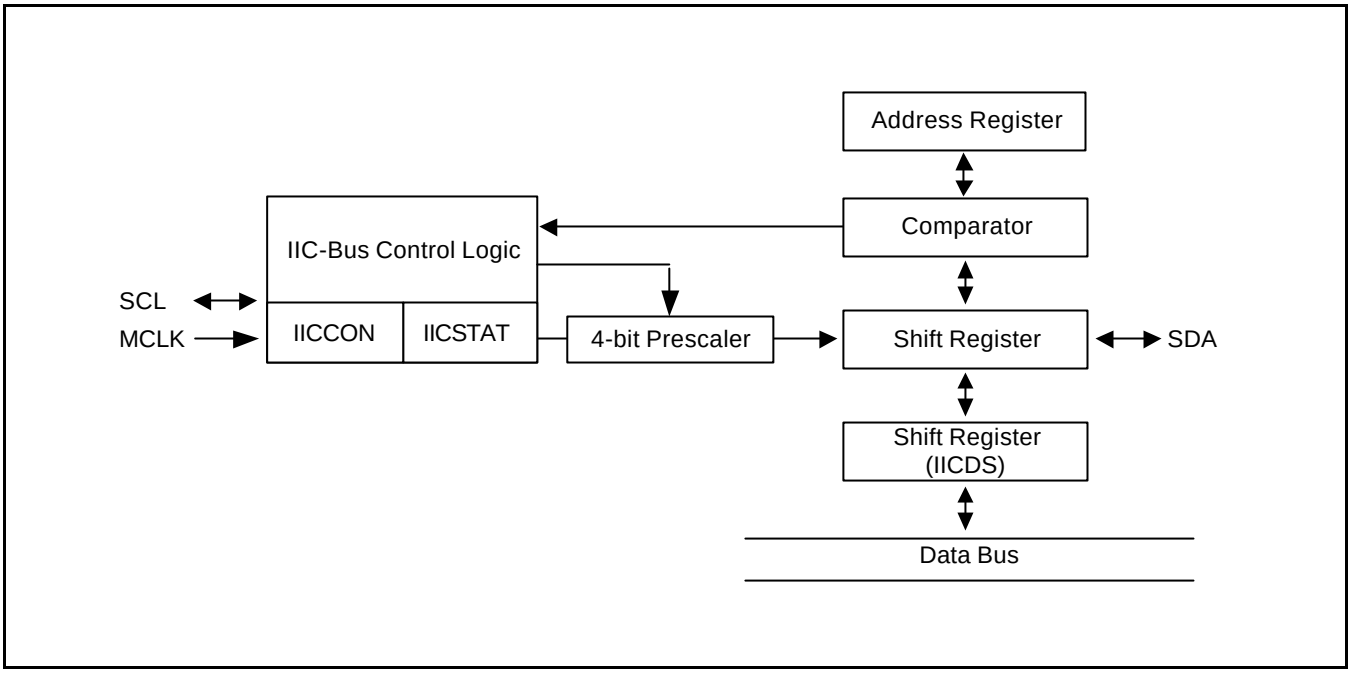


Figure 16-1. IIC-Bus Block Diagram

Note: The IIC data hold time (tSDAH) is minimum 0ns. (Refer to figure 19-52.)

1. The IIC data hold time (tSDAH) is minimum 0ns.
   Please check the data hold time of your IIC device.
   (IIC data hold time is minimum 0ns for standard/fast bus mode in IIC specification v2.1.)
2. The IIC controller supports only IIC bus device (standard/ fast bus mode), not C bus device.

THE IIC-BUS INTERFACE

The S3C44B0X 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.

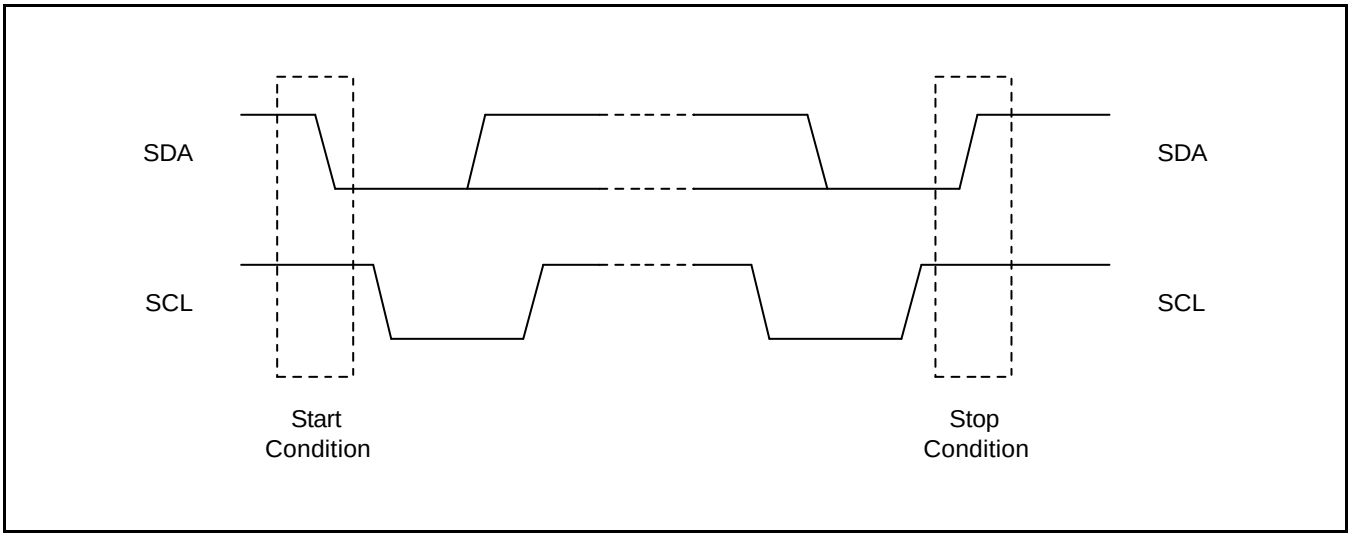


Figure 16-2. Start and Stop Condition

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.

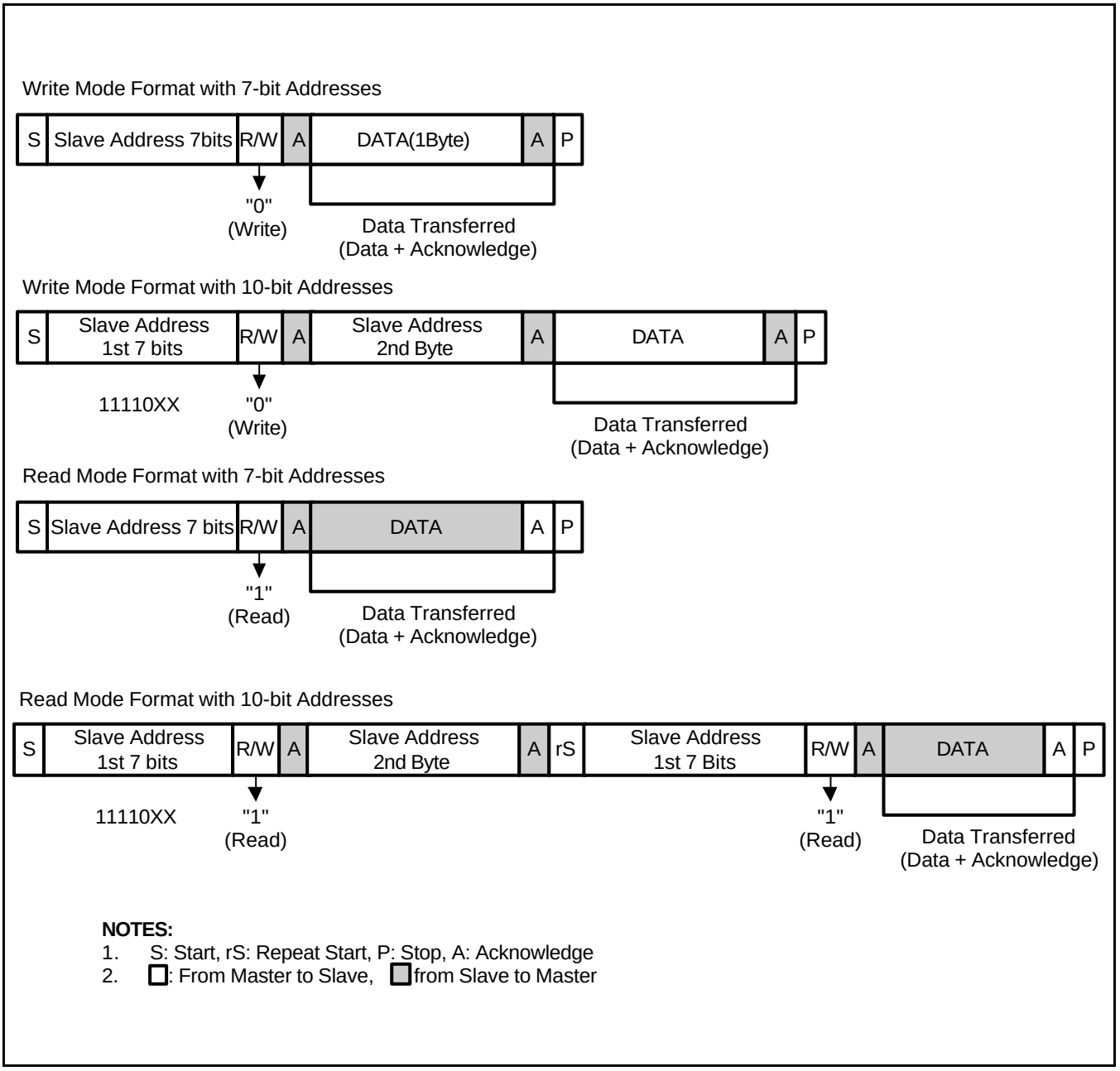


Figure 16-3. IIC-Bus Interface Data Format

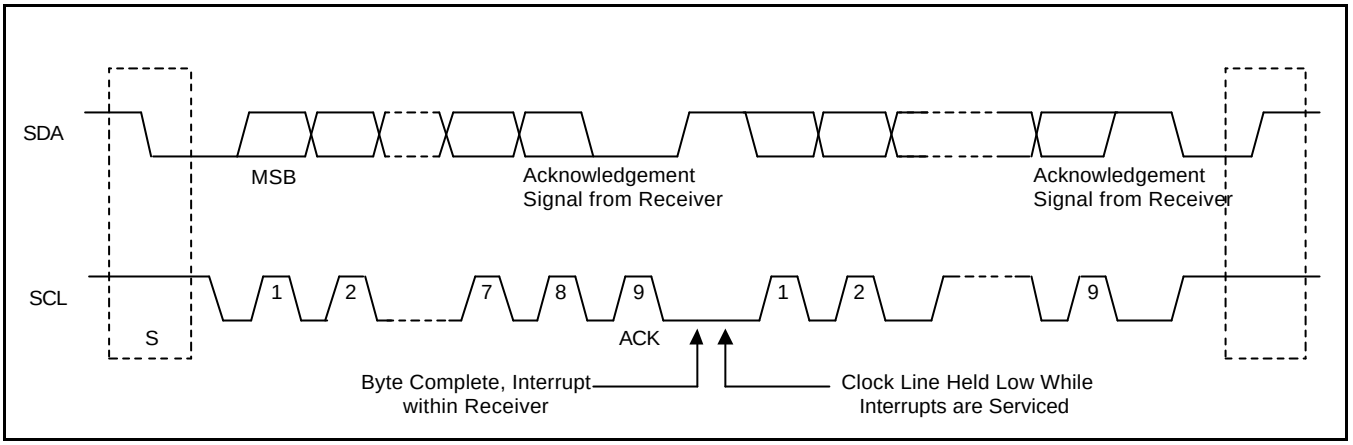


Figure 16-4. Data Transfer on the IIC-Bus

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.

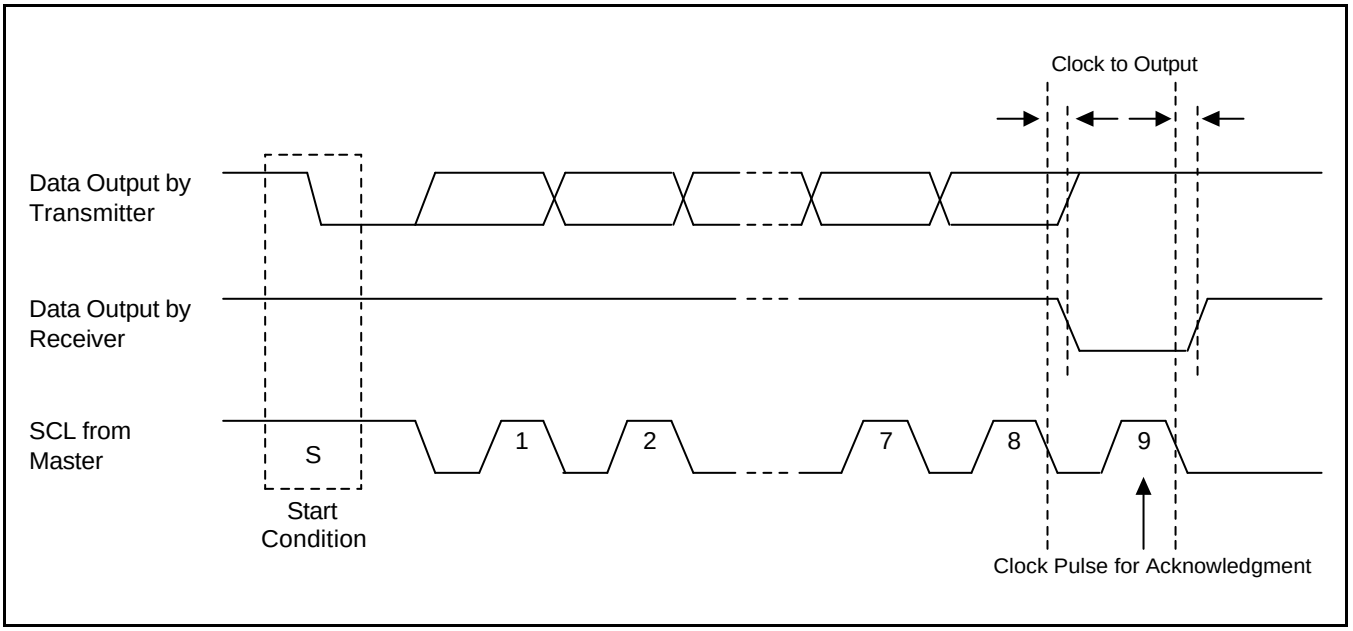


Figure 16-5. Acknowledge on the IIC-Bus

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 S3C44B0X 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 S3C44B0X 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

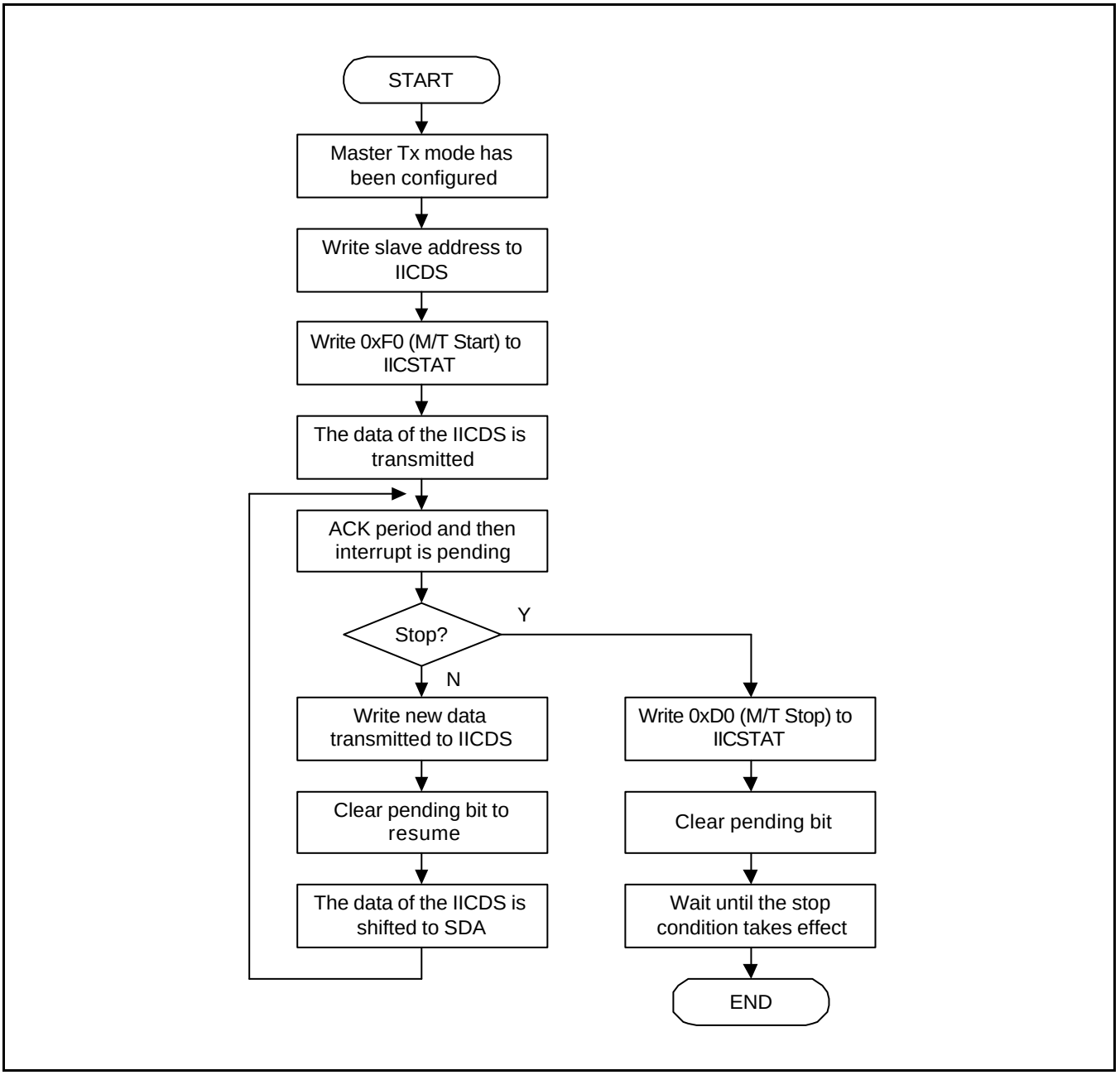


Figure 16-6. Operations for Master/Transmitter Mode

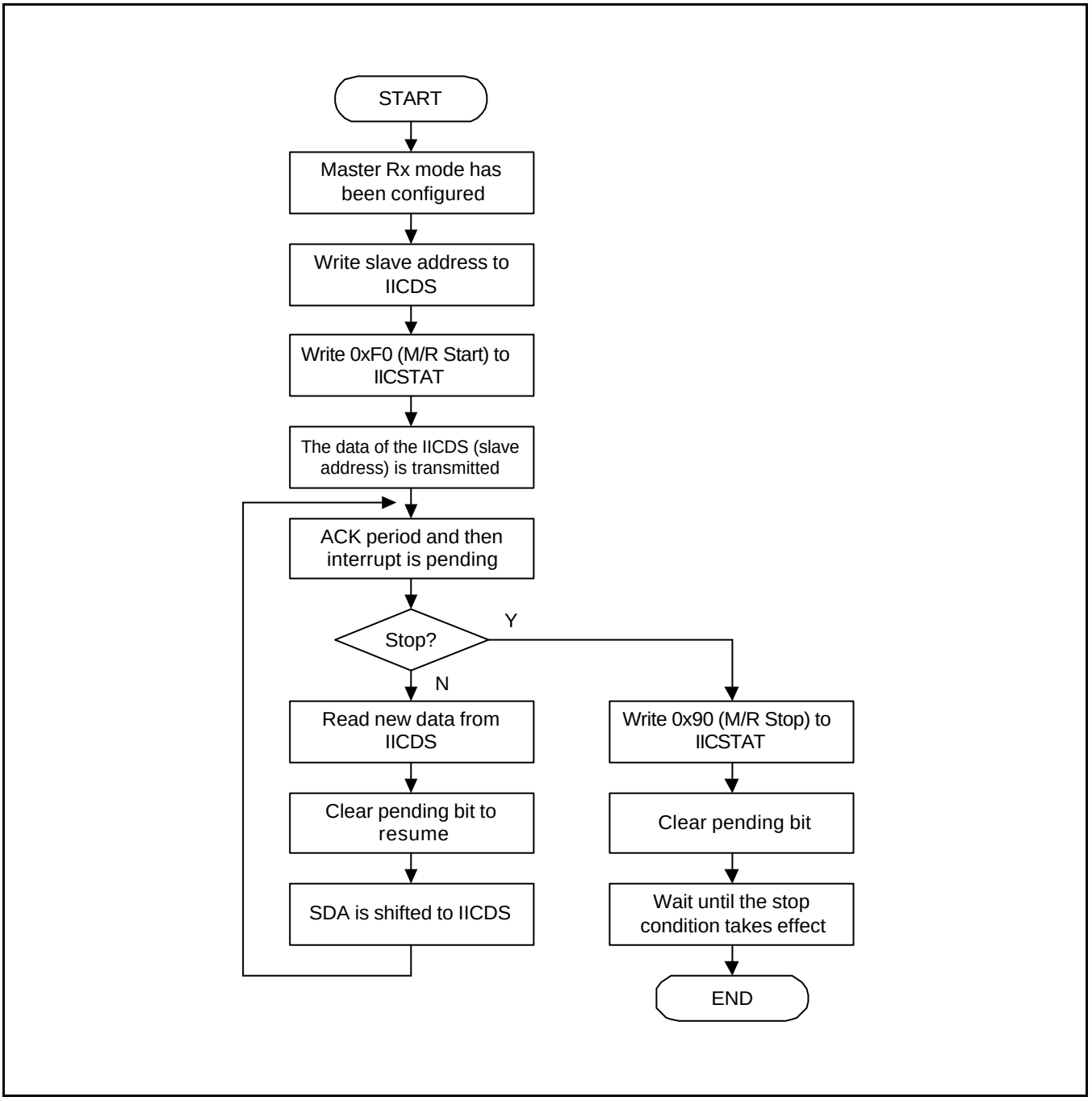


Figure 16-7. Operations for Master/Receiver Mode

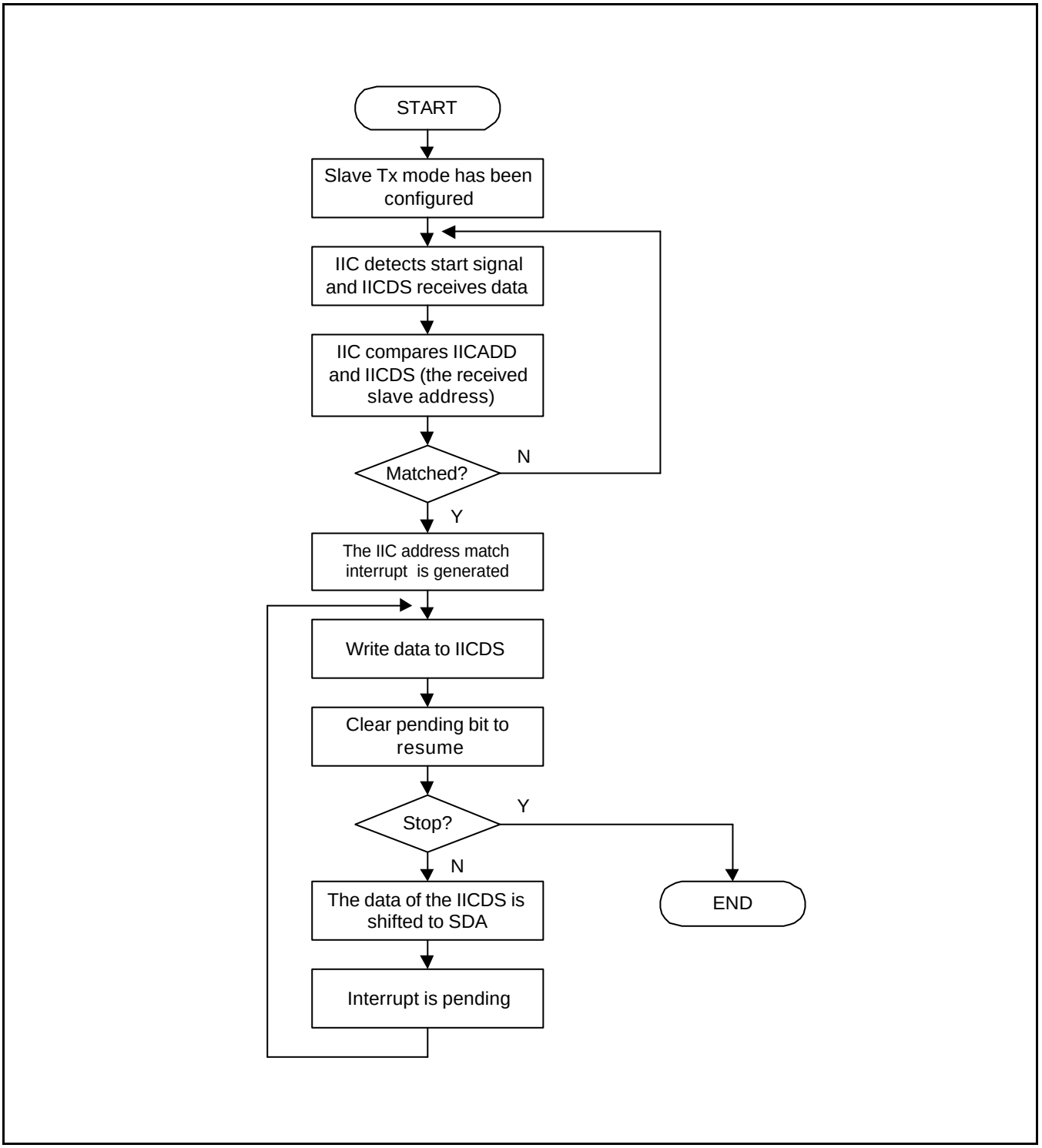


Figure 16-8. Operations for Slave/Transmitter Mode

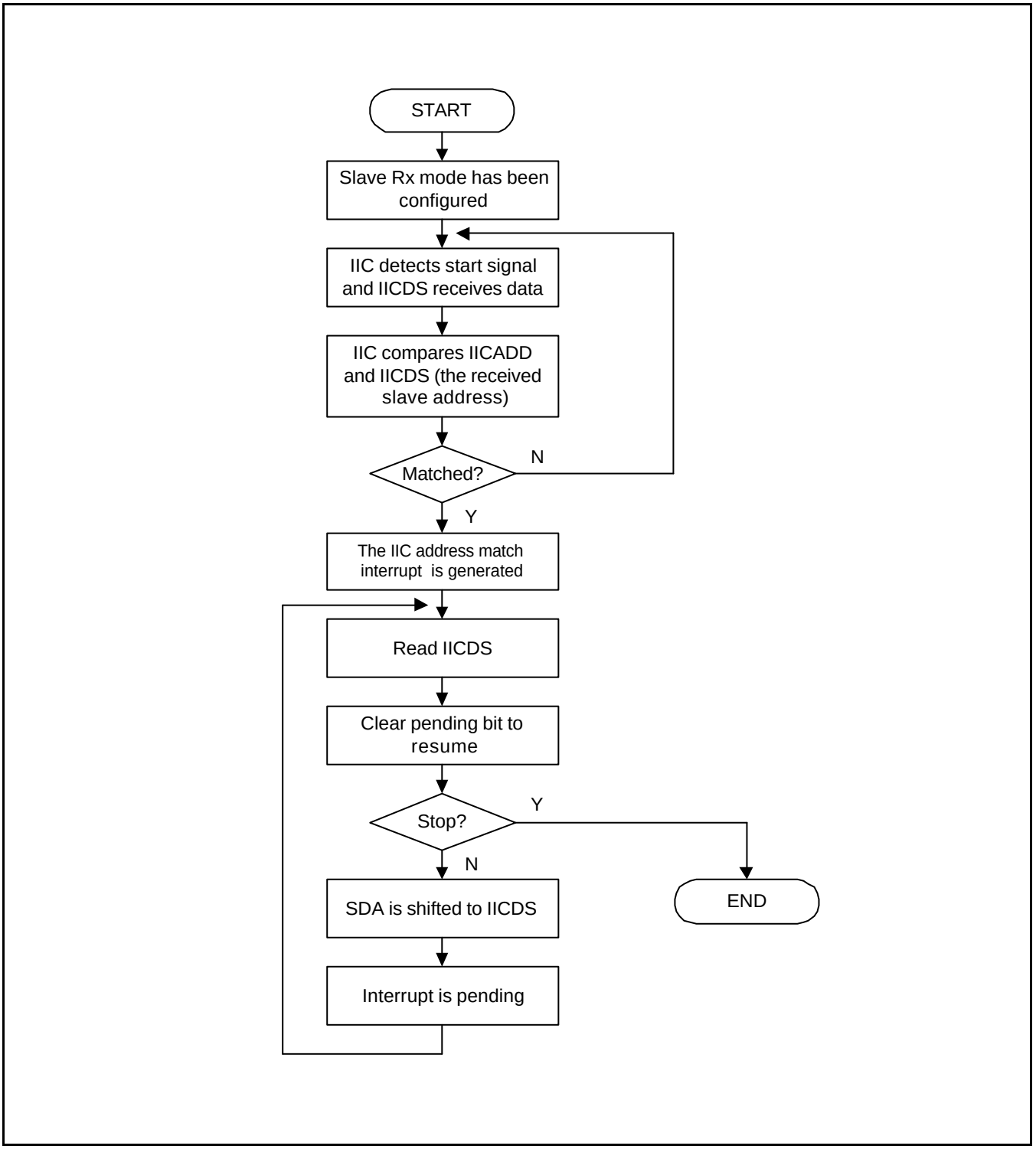


Figure 16-9. Operations for Slave/Receiver Mode

IIC-BUS INTERFACE SPECIAL REGISTERS

MULTI-MASTER IIC-BUS CONTROL REGISTER (IICCON)

Register	Address	R/W	Description	Reset Value
IICCON	0x01D60000	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_{MCLK}/16 1 = IICCLK = f_{MCLK}/512	0
Tx/Rx Interrupt enable	[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) No interrupt pending(when read), 2) Clear pending condition & Resume the operation (when write). 1 = 1) Interrupt is pending (when read) 2) N/A (when write)	0
Transmit clock value (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[5]=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	0x01D60004	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 (when read) write) IIC-bus STOP signal generation 1 = read) IIC-bus busy (when read) write) IIC-bus START signal generation. The data in IICDS will be transferred automatically just after the start signal.	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. 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	0x01D60008	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. 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	0x01D6000C	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	XXXX_XXXX

NOTES

17 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 S3C44B0X 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 (2X16) FIFO for transmit and receive

— Normal and DMA transfer mode

BLOCK DIAGRAM

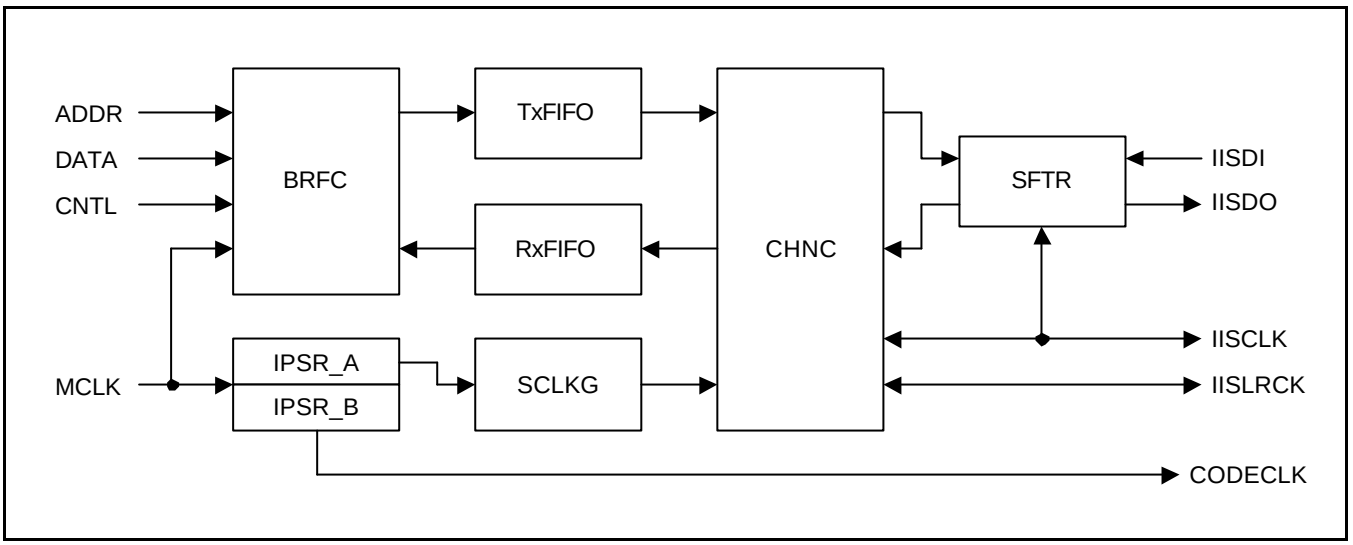


Figure 17-1. IIS-Bus Block Diagram

FUNCTIONAL DESCRIPTIONS

Bus interface, register bank, and state machine(BRFC) - Bus interface logic and FIFO access are controlled by the state machine.

3-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. Because one DMA source is assigned, normal FIFO write is done in the transmit channel, and DMA receive FIFO read is done in the receive channel and vice versa.

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.

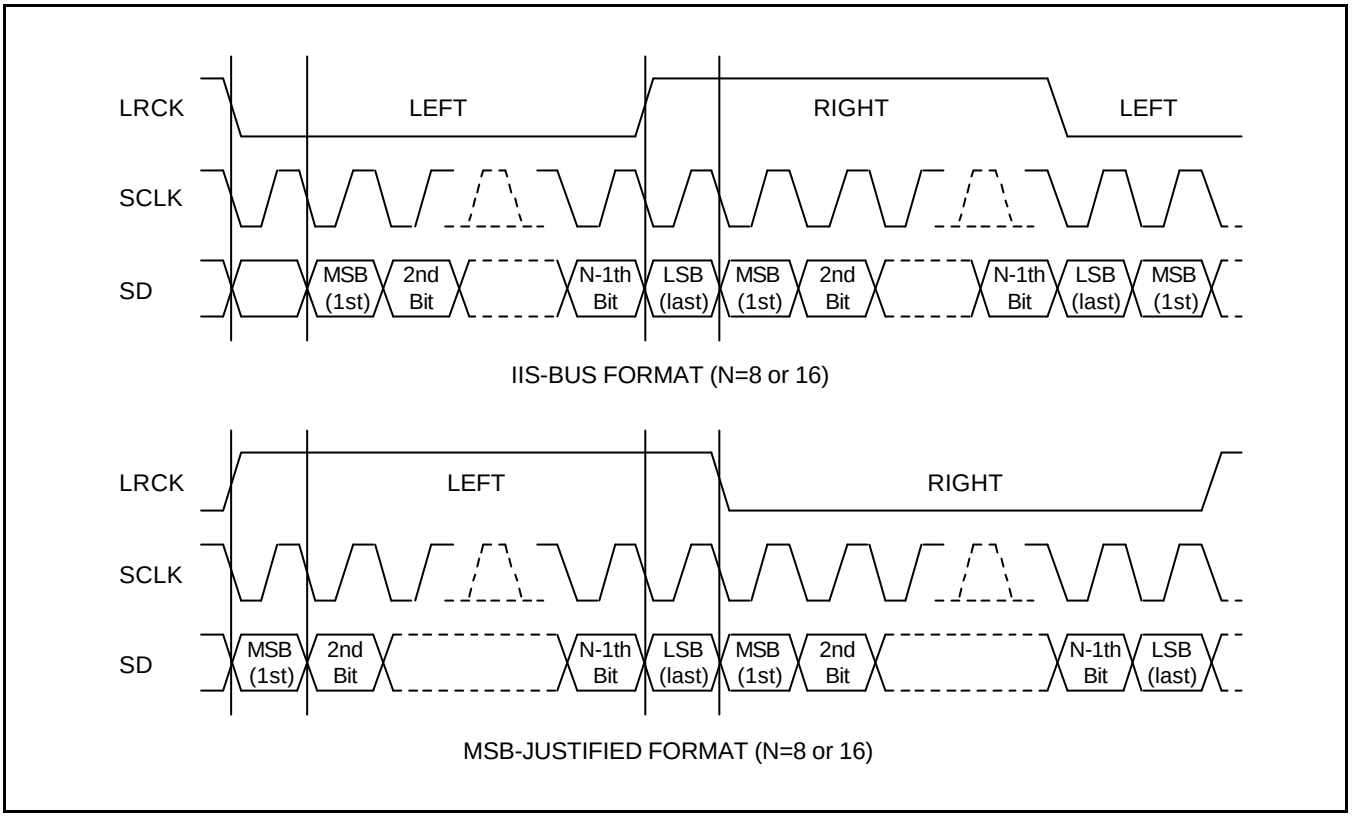


Figure 17-2. IIS-Bus and MSB(Left)-justified Data Interface Formats

SAMPLING FREQUENCY AND MASTER CLOCK

Master clock frequency(MCLK) can be selected by sampling frequency as shown in Table 17-1. Because MCLK is made by IIS prescaler, the prescaler value and MCLK 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 MCLK as shown in Table 17-2.

Table 17-1 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 17-2 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	0x01D18000(Li/HW, Li/W, Bi/W) 0x01D18002(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). This bit is only effective if the IIS is a master. 0 = IISLRCK is generated. 1 = IISLRCK is not generated.	0
Receive channel idle command	[2]	In Idle state the IISLRCK is inactive(pause Rx). This bit is only effective if the IIS is a master. 0 = IISLRCK is generated. 1 = IISLRCK is not generated.	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 halfword and word unit using STRH/STR and LDRH/LDR instructions or char/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 MODE REGISTER (IISMOD)

Register	Address	R/W	Description	Reset Value
IISMOD	0x01D18004(Li/W, Li/HW, Bi/W) 0x01D18006(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(CODECLK) 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 (fs : sampling frequency)	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	0x01D18008(Li/B, Li/HW, Li/W, Bi/W) 0x01D1800A(Bi/HW) 0x01D1800B(Bi/B)	R/W	IIS prescaler register	0x0

IISPSR	Bit	Description	Initial State
Prescaler value A	[7:4]	prescaler division factor for the prescaler A *clock_prescaler_A = MCLK/<division factor>*	0x0
Prescaler value B	[3:0]	prescaler division factor for the prescaler B *clock_prescaler_B = MCLK/<division factor>*	0x0

IISPSR[3:0] / [7:4]	Division Factor	IISPSR[3:0] / [7:4]	Division Factor
0000b	2	1000b	1
0001b	4	1001b	–
0010b	6	1010b	3*
0011b	8	1011b	–
0100b	10	1100b	5*
0101b	12	1101b	–
0110b	14	1110b	7*
0111b	16	1111b	–

NOTES:

1. If the prescaler value is 3,5,7, the duty is not 50%. In this case, the H duration is 0.5 MCLK.
2. 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.
3. (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)

To start IIS operation, the following procedure is needed.

1) Enable the FIFO in IISFCON register
2) Enable DMA request in IISCON register
3) Enable IIS interface start in IISCON register

To end IIS operation, the following procedure is needed.

1) Disable the FIFO. If you want to transmit the data remained in FIFO, you must not disable the FIFO and skip this stop 1.
2) Disable DMA request in IISCON register.
3) Disable IIS interface start in IISCON register.

Register	Address	R/W	Description	Reset Value
IISFCON	0x01D1800C(Li/HW, Li/W, Bi/W) 0x01D1800E(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 0x01D18010.

Register	Address	R/W	Description	Reset Value
IISFIF	0x01D18010(Li/HW) 0x01D18012(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

18 SIO (SYNCHRONOUS I/O)

OVERVIEW

The S3C44B0X SIO (synchronous IO) can interface with various types of external devices that requires serial data transfer. The SIO module can transmit or receive 8bit serial data at a frequency determined by its corresponding control register settings. To ensure flexible data transmission rates, you can select an internal or external clock source.

FEATURES

— 8-bit Data Buffer (SIODAT)
— 12-bit Prescaler (SBRDR)
— 8-bit Interval Counter (ITVCNT)
— Clock Selection Logic
— Serial data I/O pins (SIORXD and SIOTXD)
— External clock input/output pin (SIOCK)
— DMA run mode (auto run/flag run, SIORDY)

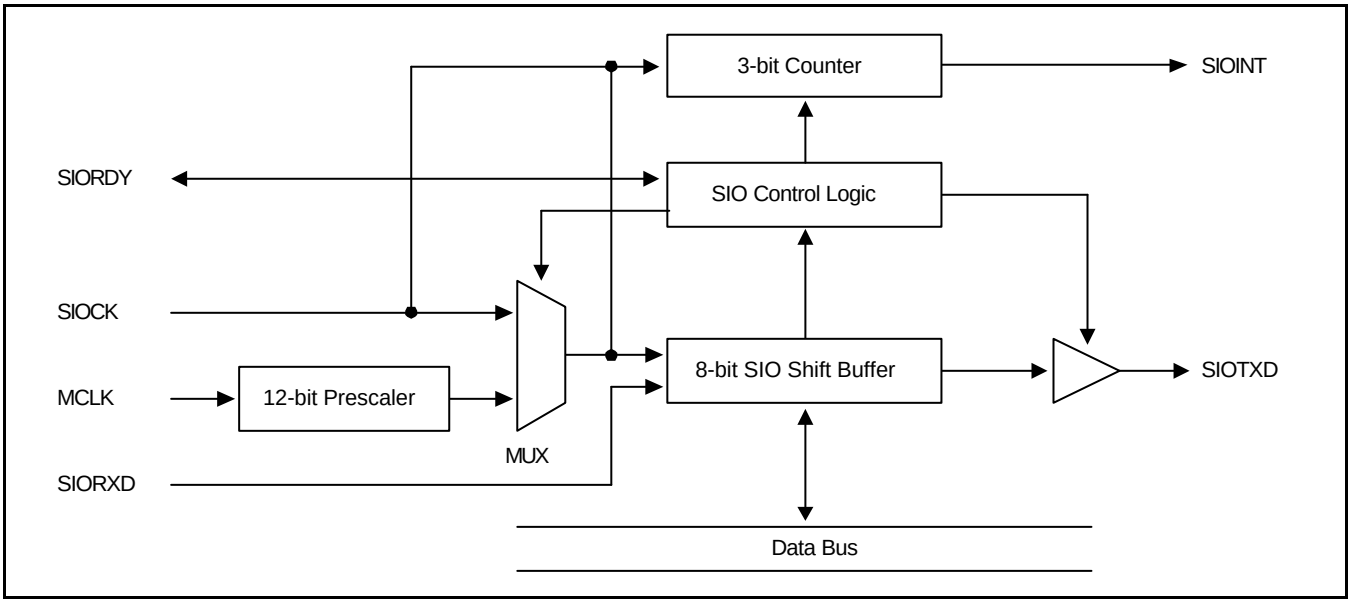


Figure 18-1. SIO Interface Block Diagram

SIO NORMAL OPERATION

Transmit and Receive by Serial Line Synchronously

Using the serial I/O interface, 8-bit data can be exchanged by serial line.

The serial output data comes through a serial input pin(SIORXD) and goes out through a serial output pin, synchronously by serial clock pin (SIOCK). After transmitting or receiving data, the SIO interrupt request is activated if a programmer enables an interrupt source.

Transmitting always occurs with reception. If you want only to transmit, you may treat the received data as dummy.

The transmission frequency is controlled by making the appropriate bit settings to the SIOCON and SBRDR registers. The serial interface can be operated by an internal or external clock source. If the internal clock signal is used, you can modify its frequency to adjust the baud rate data register value.

Programming Procedure

When a byte data is written into the SIODAT register, SIO starts to transmit if the SIO run bit is set and the transmit mode bit is enabled.

To program the SIO modules, follow these basic steps:

1. Configure the I/O pins at port (SIOTXD, SIOCLK, SIORXD).
2. Set SIOCON register to properly configure the serial I/O module.
3. For interrupt generation, set the serial I/O interrupt enable bit and refer the interrupt controller to 1.
4. If you want to transmit data to the serial buffer, write data to SIODAT.
5. For receiving/transmitting, set SIOCON[3] to 1 to start the shift operation.
6. When the shift operation (transmit/receive) is completed, the SIO interrupt is requested and SIODAT has the received data or dummy data.
7. go to step 4

SIO DMA OPERATION

Auto Run Mode (non-hand-shaking mode)

If the SIO is in the auto-run mode (non-hand-shaking mode) and the SIO transmits data using the DMA controller, SIO can wait until the transmitted data is read by the external destination device. Not using hand-shaking, the SIO must wait for a fixed interval between every 8-bit data. The interval is determined by the IVTCNT register. In the auto run mode, the SIO inserts this interval after transmitting every 8-bit data.

Steps for Transmit by DMA(Refer to Fig.18-2)

1. DCNTZ[n] is cleared to 0, which allows the SIO to request DMA service. The SIO is configured properly, but the value of SIOCON[1:0] has to be 00b.
2. DMA is configured properly.
3. The SIO is configured as DMA transmit mode. SIOCON[3] (SIO start bit) will be ignored.
4. The SIO automatically requests DMA service without SIO start bit(SIOCON[3]).
5. The SIO transmits the data.
6. Go to step 4 until DMA count is 0.
7. DCNTZ[n] is set to 1, which stops the SIO from requesting further DMA service.

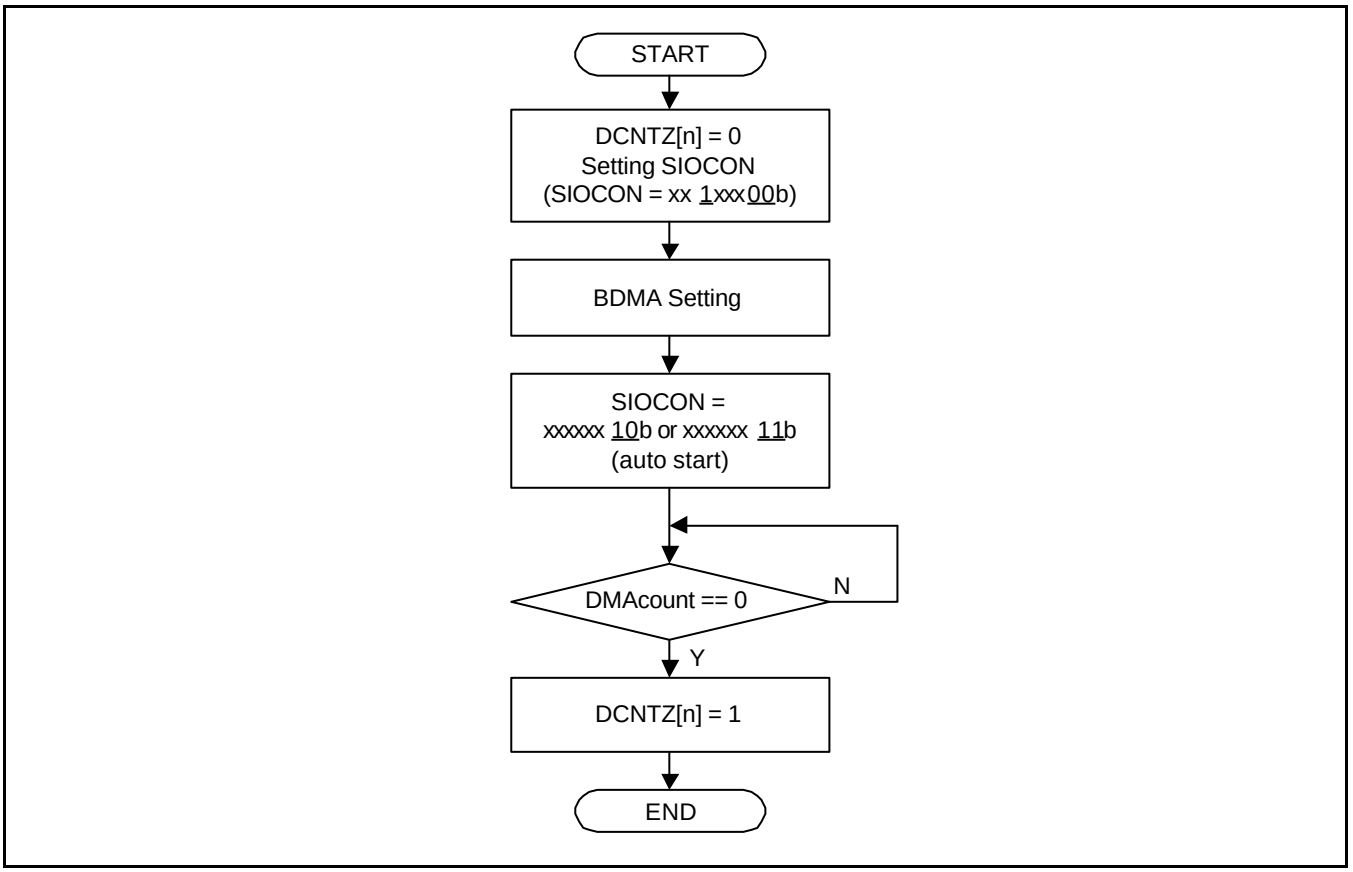


Figure 18-2. SIO Transmit by DMA

Steps for Receive by DMA(Refer to Fig.18-3)

1. DCNTZ[n] is cleared to 0, which allows the SIO to request the DMA service. The SIO is configured properly. But the value of SIOCON[1:0] has to be 00b.
2. DMA is configured properly.
3. The SIO is configured in DMA receive only mode.
4. Set SIOCON[3] (SIO start bit) to start the receiving operation.
5. The SIO requests the DMA service after 8-bit data has been received.
6. Go to step 5 until DMA count is 0.
7. DCNTZ[n] is set to 1, which stops the SIO from requesting further DMA service.

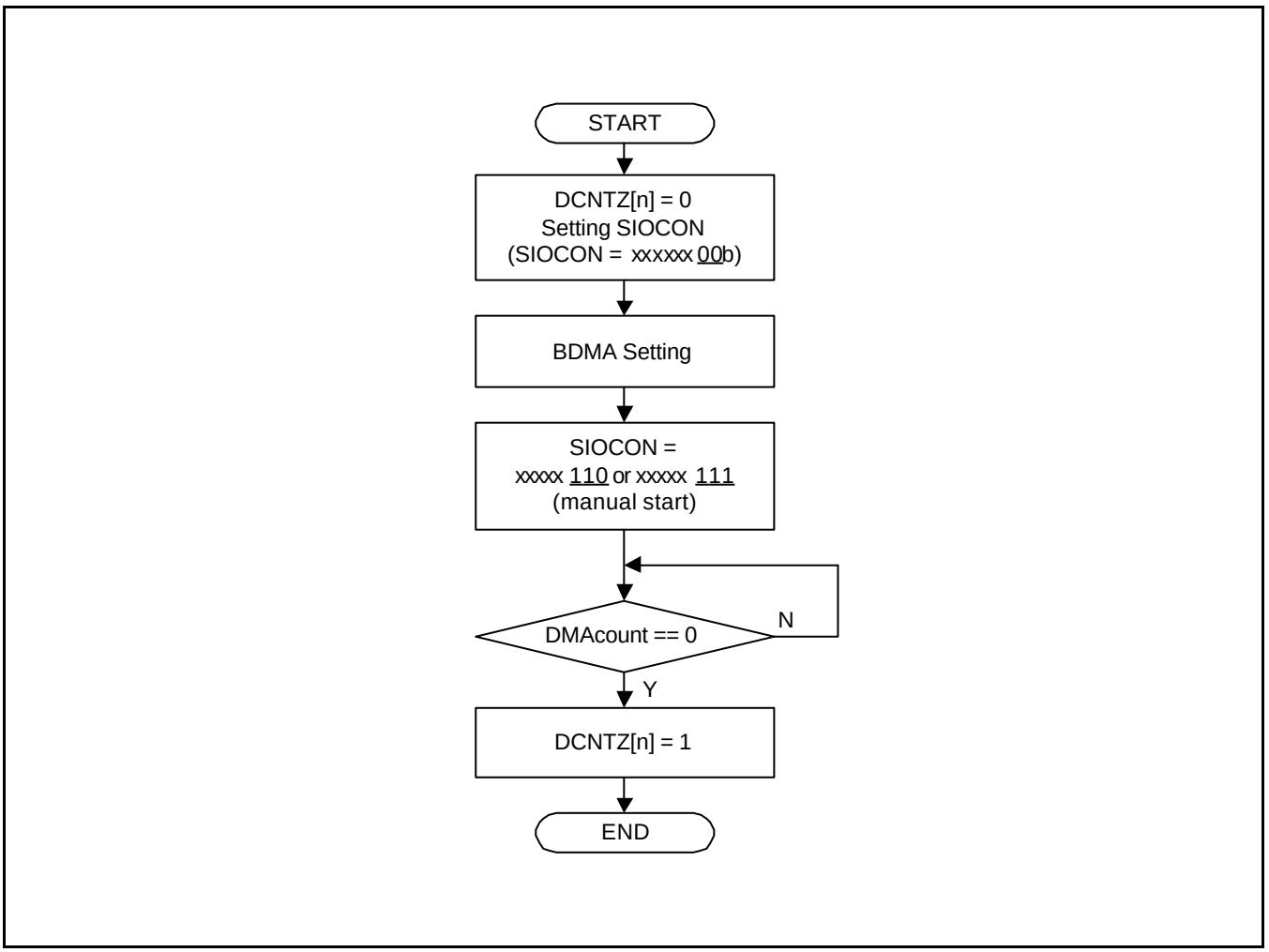


Figure 18-3. SIO Receive by DMA

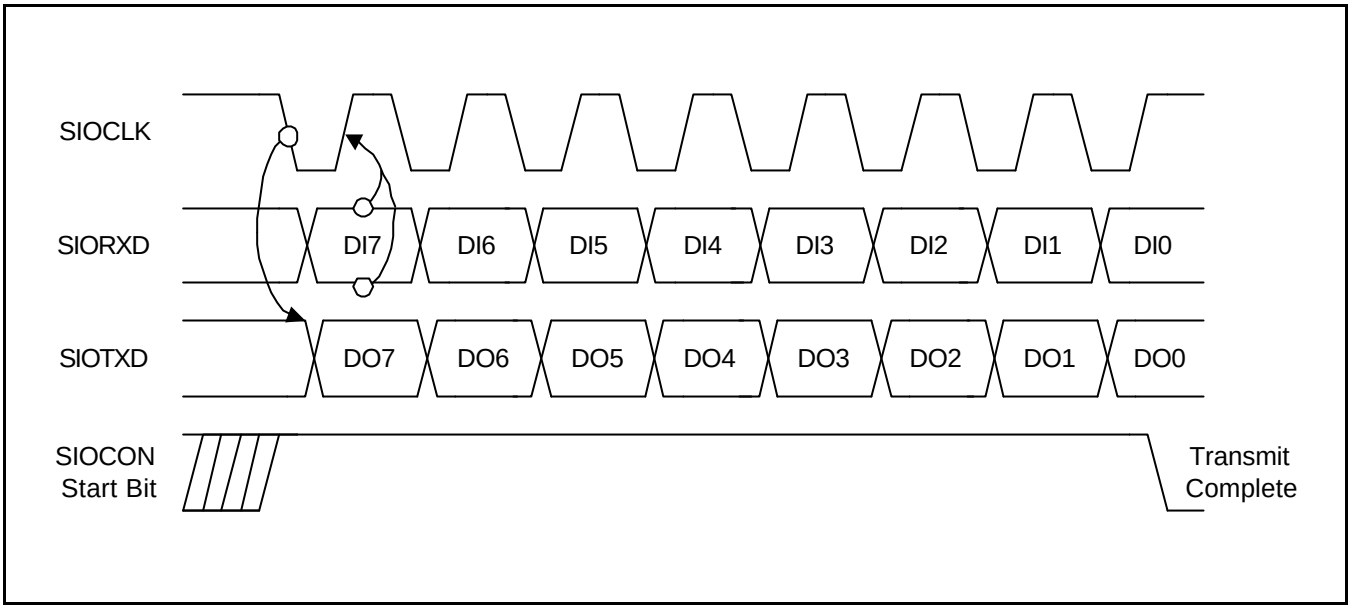


Figure 18-4. SIO Transmit/Receive Mode Timing diagram(Tx at Falling)

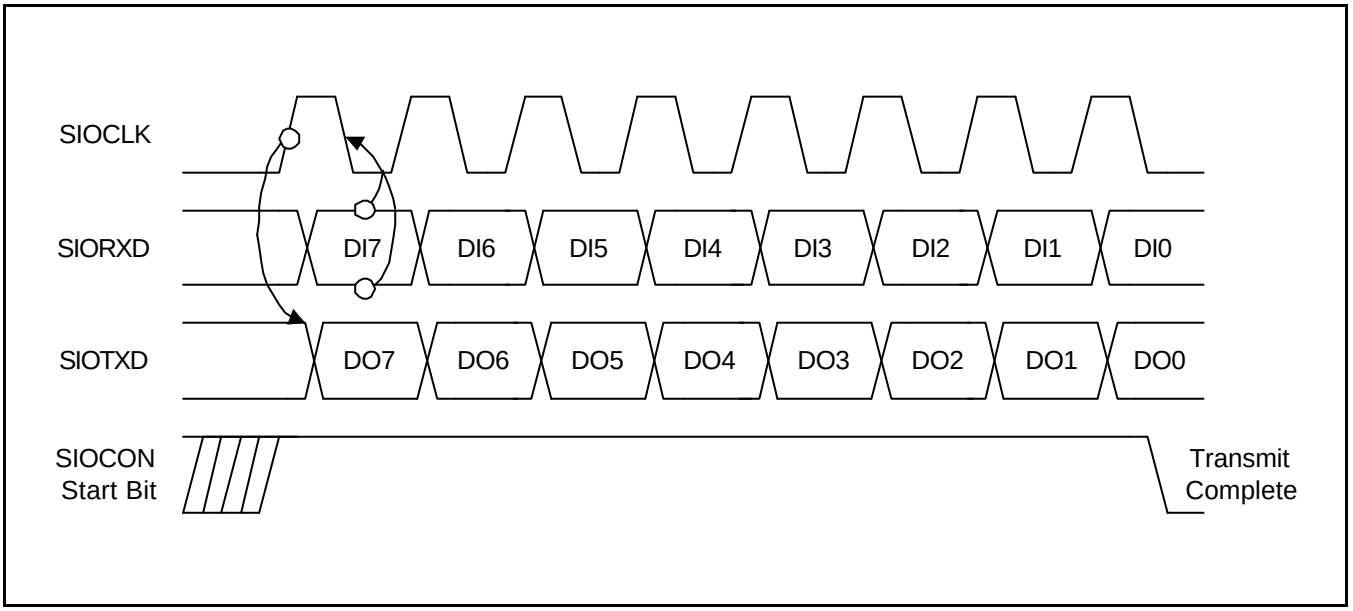
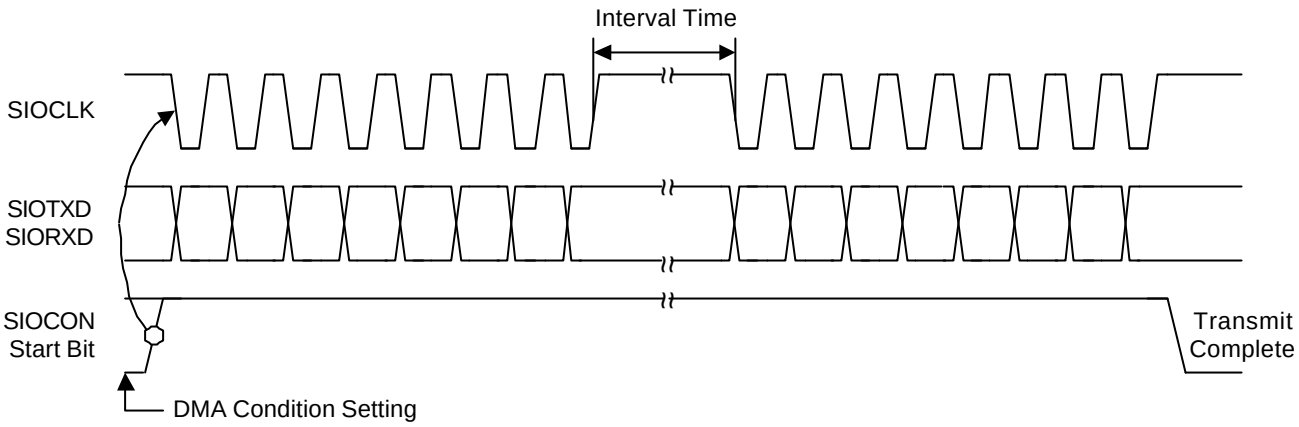


Figure 18-5. SIO Transmit/Receive Mode Timing diagram(Tx at Rising)



NOTE: SIO Tx is auto-start regardless of the SIOCON start bit.

Figure 18-6. SIO in Non-Hand-shaking Mode Timing diagram(Auto Run Mode)

SYNCHRONOUS I/O INTERFACE SPECIAL REGISTERS

SIO CONTROL REGISTER (SIOCON)

Register	Address	R/W	Description	Reset Value
SIOCON	0x01D14000	R/W	SIO control register	0x00

SIOCON	Bit	Description	Initial State
Clock source select	[7]	SIO shift clock source select bit. 0 = Internal clock, 1 = External clock	0
Data direction	[6]	This bit controls whether MSB is transmitted first or LSB is transmitted first. 0 = MSB mode, 1 = LSB mode	0
Tx/Rx selection	[5]	This bit decides whether to enable the transmit operation enabled. If you want to only transmit, the received data in SIODAT will be ignored. If users want to transmit and receive, SIO supports data transmission and reception simultaneously. Users write the data transmitted in the SIODAT register and then SIO will transmit the data serially. At the same time, SIO will receive the data from an external SIO device. After the SIO transmission is completed, the contents of SIODAT, will have the received data. 0 = Receive only mode, 1 = Transmit/Receive mode	0
Clock edge select	[4]	This bit determines the clock to be used for serial transmit or receive operation. 0 = falling edge clock, 1 = rising edge clock	0
SIO start	[3]	This bit determines whether the SIO functions is running or has stopped. When BDMA Tx is used, this bit should be '0'. 0 = No action 1 = Clear 3-bit counter and start shift. This bit is cleared just after writing this bit as 1.	0
Shift operation	[2]	Determines SIO shift operation 0 = Non hand-shaking mode(Auto run mode) 1 = Reserved	0
SIO mode select	[1:0]	Determines how and by what SIODATA is read/written. 00 = no operations 01 = SIO interrupt mode 10 = BDMA0 mode 11 = BDMA1 mode	00

SIO DATA REGISTER (SIODAT)

Before transmitting, the SIO data register (SIODAT) contains an 8-bit data value to be transmitted. After transmitting is completed, the SIODAT has the received data or dummy data.

Register	Address	R/W	Description	Reset Value
SIODAT	0x01D14004	R/W	SIO data register	0x00

SIODAT	Bit	Description	Initial State
SIO DATA	[7:0]	This field contains the data to be transmitted or received over the SIO channel.	0x00

SIO BAUD RATE PRESCALER REGISTER (SBRDR)

The baud rate prescaler register (SBRDR) determines SIO clock rate (baud rate) as follows.

Baud rate = MCLK / 2 /(Prescaler value + 1)

Register	Address	R/W	Description	Reset Value
SBRDR	0x01D14008	R/W	SIO baud rate prescaler register	0x00

SBRDR	Bit	Description	Initial State
SBRDR	[11:0]	This field contains the prescaler value for the baud rate	0x00

SIO INTERVAL COUNT REGISTER (IVTCNT)

In the auto run mode, the SIO inserts this interval after transmitting every 8-bit data.

Intervals (between 8-bit data) = MCLK / 4/ (IVTCNT +1)

Register	Address	R/W	Description	Reset Value
IVTCNT	0x01D1400C	R/W	SIO interval counter register	0x00

IVTCNT	Bit	Description	Initial State
IVTCNT	[7:0]	SIO interval counter register	0x00

SIO DMA COUNT ZERO REGISTER (DCNTZ)

When SIO operates in DMA mode, the corresponding DCNTZ bit has to be 0 initially. When DMA terminal count is reached, the corresponding DCNTZ bit has to be set to 1.

Register	Address	R/W	Description	Reset Value
DCNTZ	0x01D14010	R/W	SIO dma count zero register	0x0

DCNTZ	Bit	Description	Initial State
DCNTZ1	[1]	0: Enables BDMA1 service request. When this bit is 0, the SIO can request the DMA service. 1: Disables BDMA1 service request	0
DCNTZ0	[0]	0: Enables BDMA0 service request When this bit is 0, the SIO can request the DMA service. 1: Disables BDMA0 service request	0

NOTES

19 ELECTRICAL DATA

ABSOLUTE MAXIMUM RATINGS

Table 19-1. Absolute Maximum Rating

Symbol	Parameter	Rating		Unit
V_{DD}	DC Supply Voltage	3.6		V
V_{IN}	DC Input Voltage	3.3 V Input buffer	4.6	
V_{OUT}	DC Input Voltage	3.3 V buffer	4.6	
I_{latch}	Latch-up Current	± 200		mA
T_{STG}	Storage Temperature	- 40 to 125		°C

RECOMMENDED OPERATING CONDITIONS

Table 19-2. Recommended Operating Conditions

Symbol	Parameter	Rating		Unit
V_{DDP}	DC Supply Voltage	3.3 V I/O	3.0 to 3.6	V
V_{DDI}	Internal Voltage	2.5 V tolerant	2.3 to 2.7	
V_{DDA}	Analog core DC Input Voltage	2.5 V Core	2.5 ± 5%	
T_A	Commercial temperature range	0 to 70		°C

D.C. ELECTRICAL CHARACTERISTICS

Table 19-3. Normal I/O PAD DC Electrical Characteristics

V_{DDP} = 3.3 V ± 0.3 V, T_A = 0 to 70 °C

Symbol	Parameters	Condition	Min	Typ	Max	Unit
VT+	Schmitt trigger, positive-going threshold	LVCMOS			2.0	V
VT-	Schmitt trigger, negative-going threshold	LVCMOS	0.8			
V_H	VT+ - VT-	Schmitt-trigger	0.5	0.575	0.65	
I_{IH}	High level input current					uA
	Input buffer	V_{IN} = V_{DDP}	-10		10	
	Input buffer with pull-up		10	33	60	
I_{IL}	Low level input current					uA
	Input buffer	V_{IN} = V_{SS}	-10		10	
	Input buffer with pull-up		-60	-33	-10	
V_{OH}	High level output voltage (note)					V
	Type B6	I_{OH} = -6 mA	2.4			
	Type B8	I_{OH} = -8 mA				
	Type B10	I_{OH} = -10 mA				
	Type B12	I_{OH} = -12 mA				
V_{OL}	Low level output voltage (note)					V
	Type B4	I_{OL} = 4 mA			0.4	
	Type B6	I_{OL} = 6 mA				
	Type B8	I_{OL} = 8 mA				
	Type B10	I_{OL} = 10 mA				
	Type B12	I_{OL} = 12 mA				
I_{DS}	Stop current	V_{IN} = V_{SS} or V_{DD}		5		uA @25 °C
I_{DD}	Operating current				1.2	mA/MHz

NOTE: Type B4 means 4mA output driver cell, and Type B8 means 8mA output driver cells.

Table 19-4. DC Electrical Characteristics

(TA = 0 to 70 °C)

Item	Symbol	Min	Typ	Max	Unit	Remarks
Normal operation	I_{DDCPU}	–	60*	80	mA	* : 66MHz
Idle mode			23*	28		Both oscillators running, CPU static, LCD refresh active
** Slow mode(@1MHz)	I_{TOTAL}		2.1	–		** : Total current consumption. (CPU+I/O)
** SL-Idle mode(@1MHz)			1			
** Stop mode			5		uA (@25 °C)	Just running 32KHz oscillator(for RTC), all other I/O static.
RTC consumption	I_{RTC}		2	5	uA	x-tal = 32.768KHz for RTC

Table 19-5. Typical current decrease percentage by CLKCON register(@66MHz)

(Unit: %)

Peri	IIS	IIC	ADC	RTC	UART1	SIO	ZDMA0/1	Timer012345	LCD	Total
Current saving	1.3%	1.6%	0.7%	0.8%	3.8%	0.9%	2.2%	2.2%	3.2%	16.7

NOTE: This table includes each power consumption of each peripherals. For example, If you do not use IIS and you turned off IIS block by CLKCON register, you can save the 1.3% portion from total power consumption.

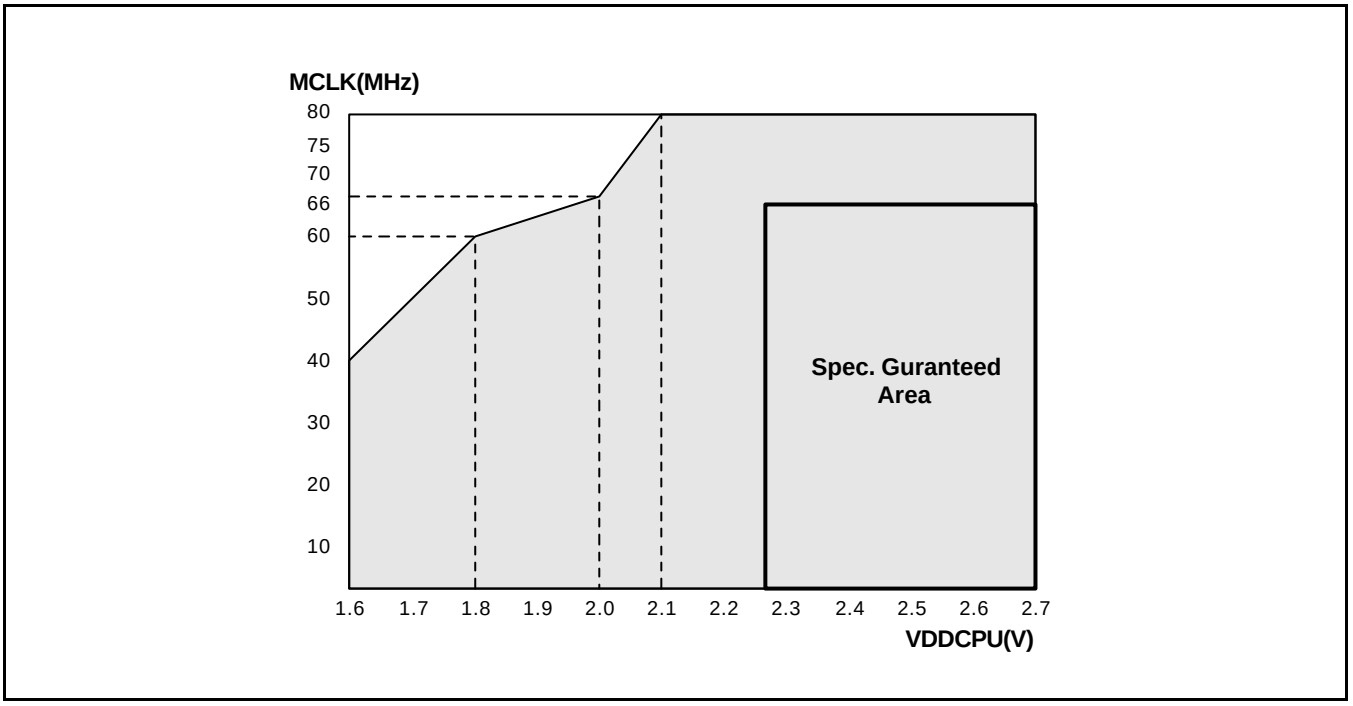
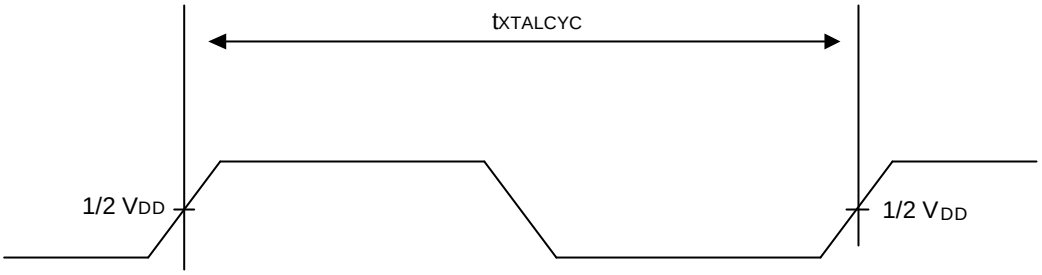


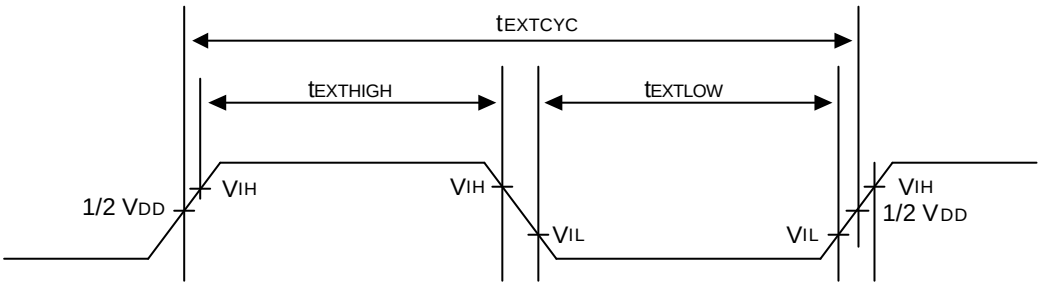
Figure 19-1. Typical Operating Voltage/Frequency Range (VDDIO=3.3V, @Room temperature & SMDK41100 board)

A.C. ELECTRICAL CHARACTERISTICS



NOTE: The clock input from the EXTAL0 pin.

Figure 19-2. EXTAL0 Clock Timing



NOTE: The clock input from the EXTCLK pin.

Figure 19-3. EXTCLK Clock Input Timing

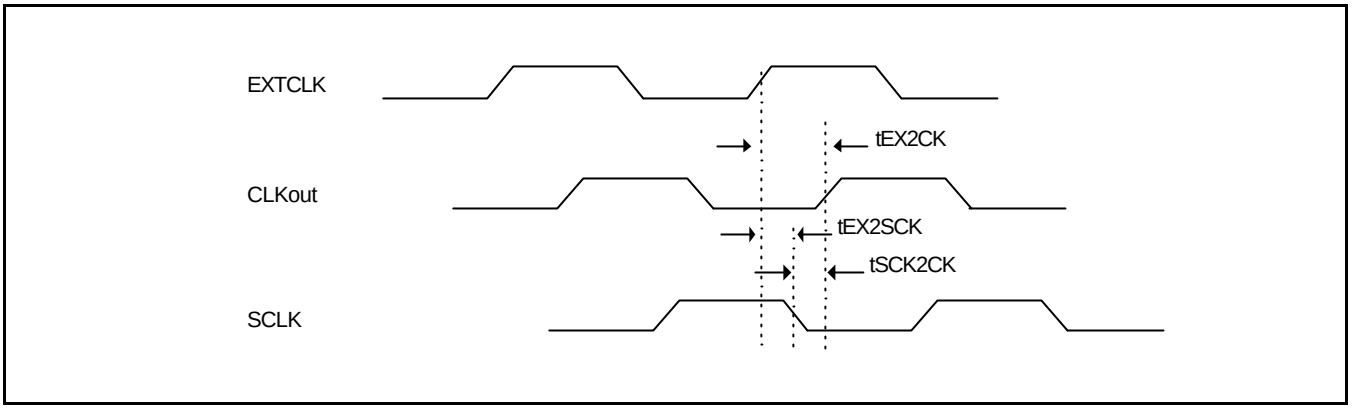


Figure 19-4. EXTCLK/CLKout/SCLK in the case that EXTCLK is used without the PLL

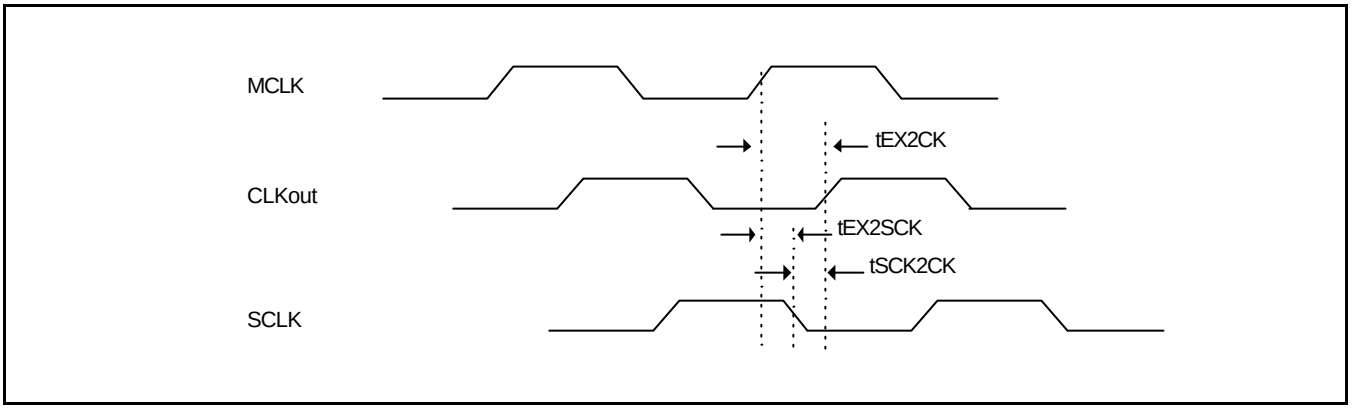


Figure 19-5. MCLK/CLKout/SCLK in the case that EXTCLK is used with the PLL

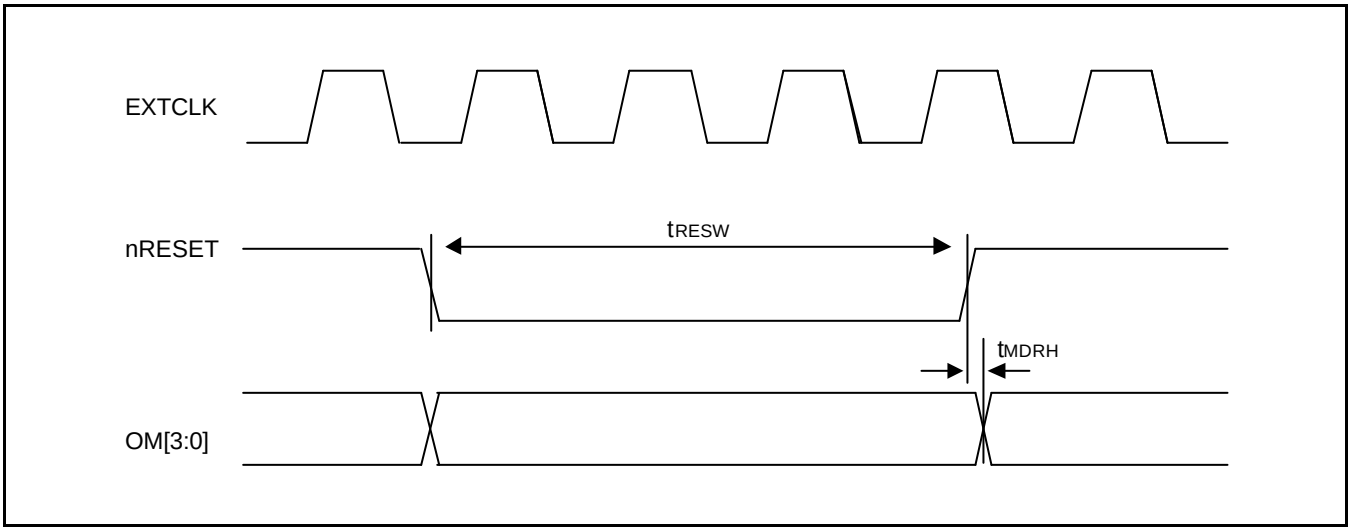


Figure 19-6. Manual Reset and OM[3:0] Input Timing

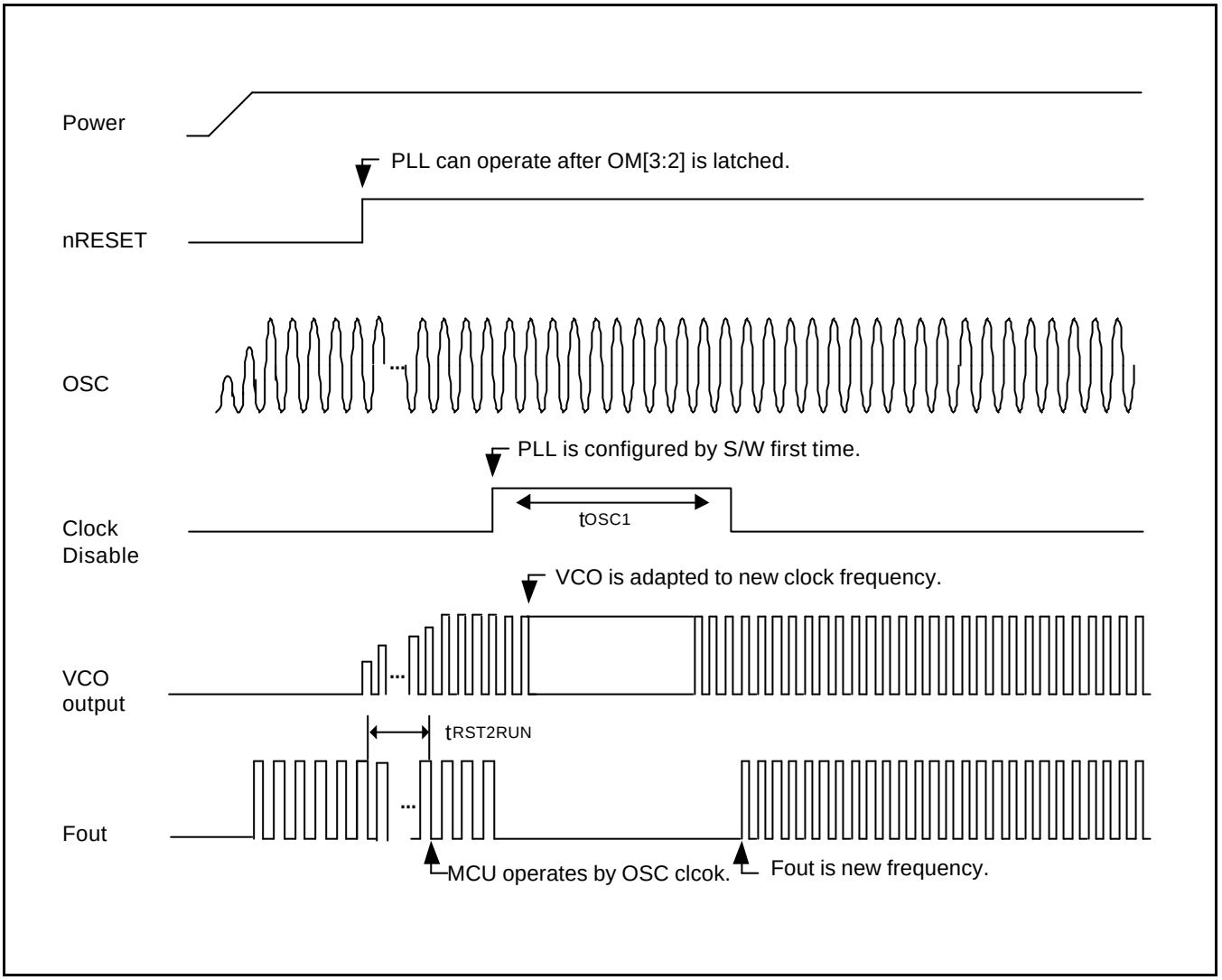


Figure 19-7. Power-On Oscillation Setting Timing

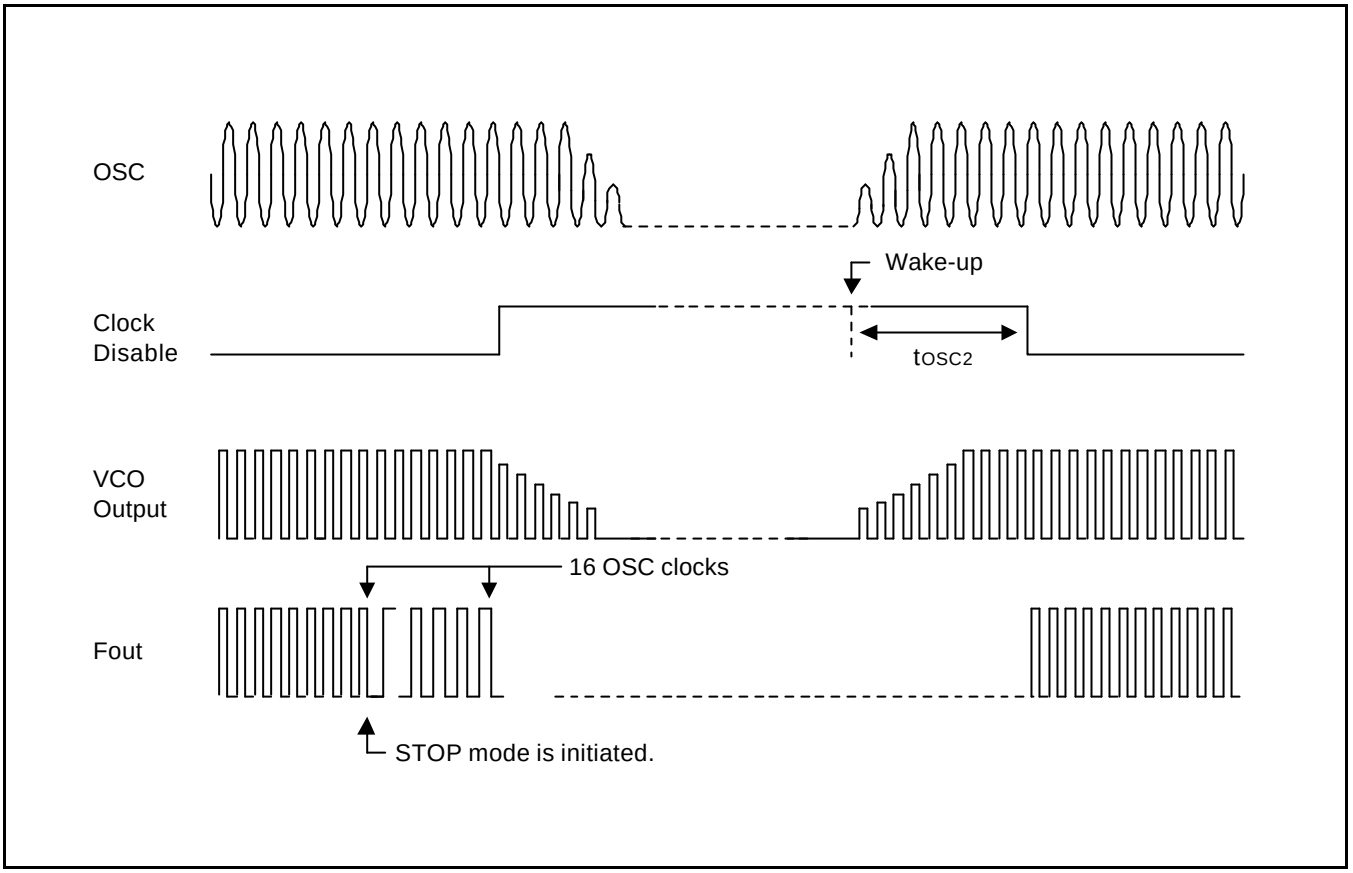


Figure 19-8. STOP Mode Return Oscillation Setting Timing

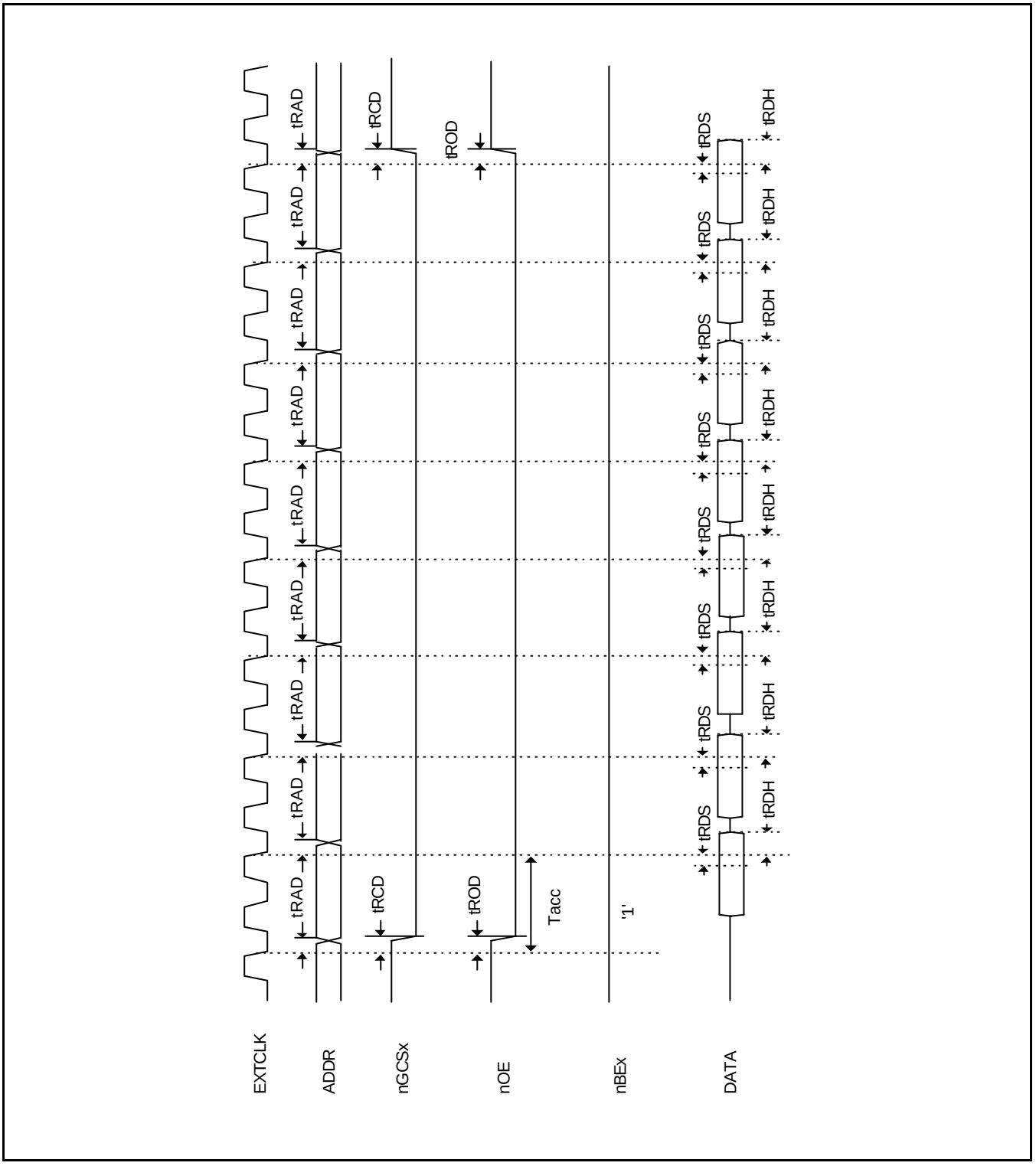


Figure 19-9. ROM/SRAM Burst READ Timing(I)
(Tacs=0, Tcos=0, Tacc=2, Toch=0, Tcah=0, PMC=10b, ST=0, DW=16bit)

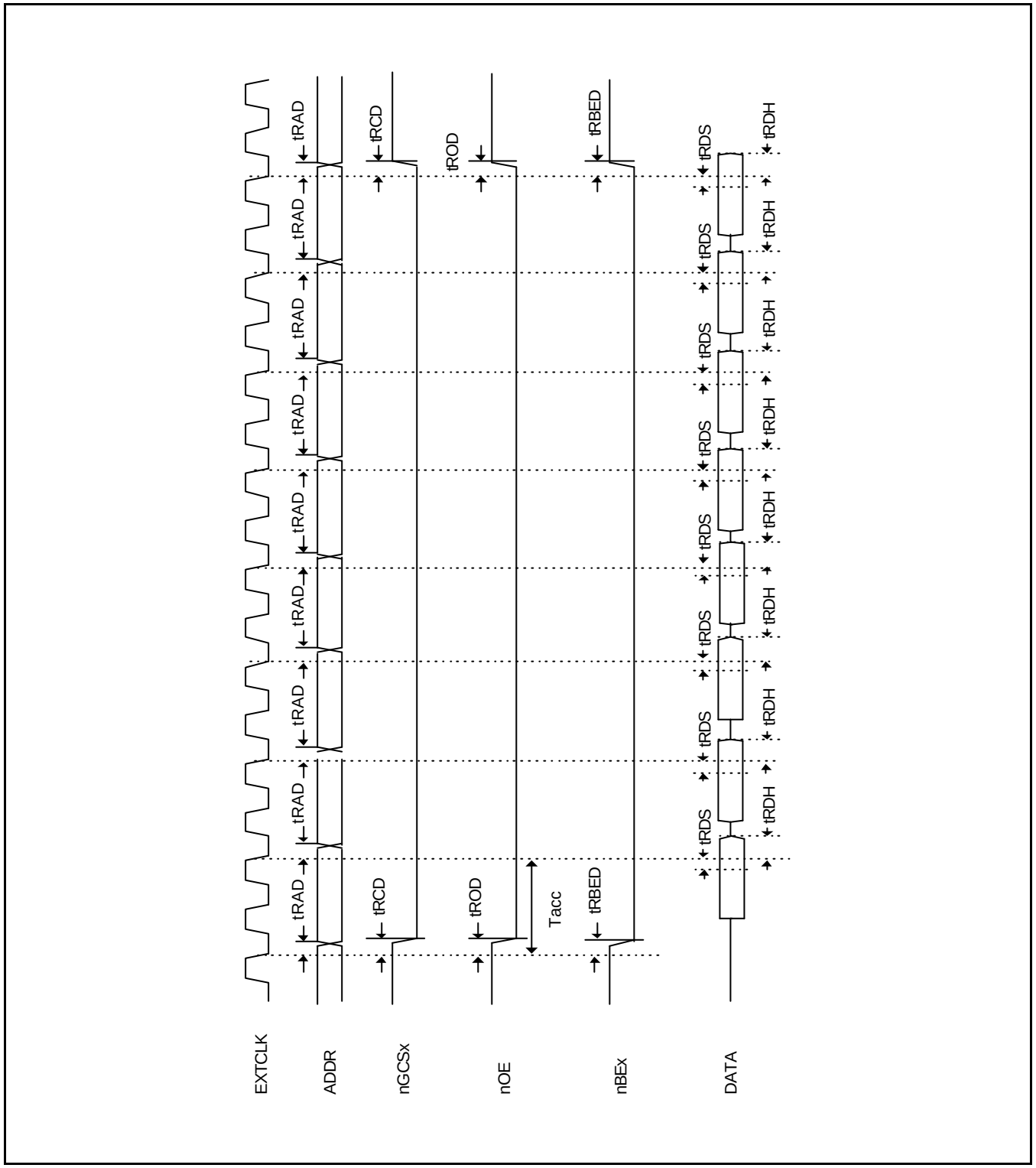


Figure 19-10. ROM/SRAM Burst READ Timing(II)
(Tacs=0, Tcos=0, Tacc=2, Toch=0, Tcah=0, PMC=10b, ST=1, DW=16bit)

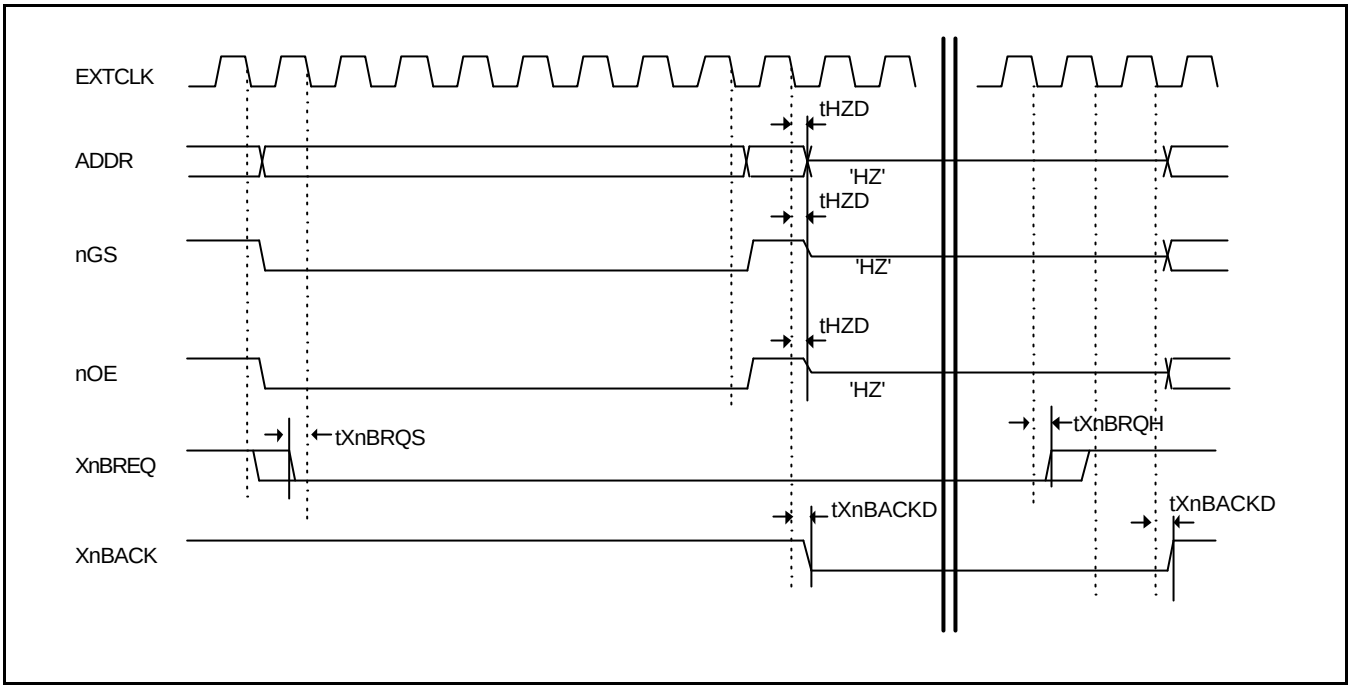


Figure 19-11. External Bus Request in ROM/SRAM Cycle (Tacs=0, Tcos=0, Tacc=8, Toch=0, Tcah=0, PMC=0, ST=0)

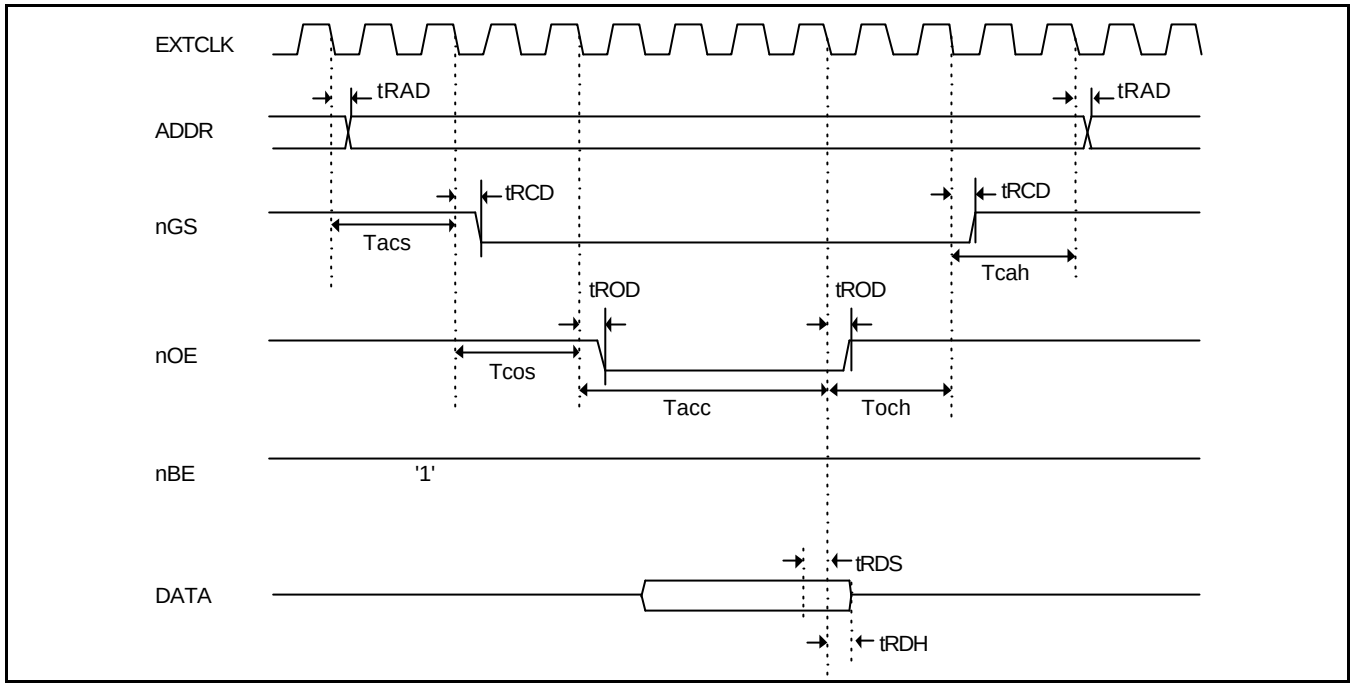


Figure 19-12. ROM/SRAM READ Timing (I)
(Tacs=2,Tcos=2, Tacc=4, Toch=2, Tcah=2, PMC=0, ST=0)

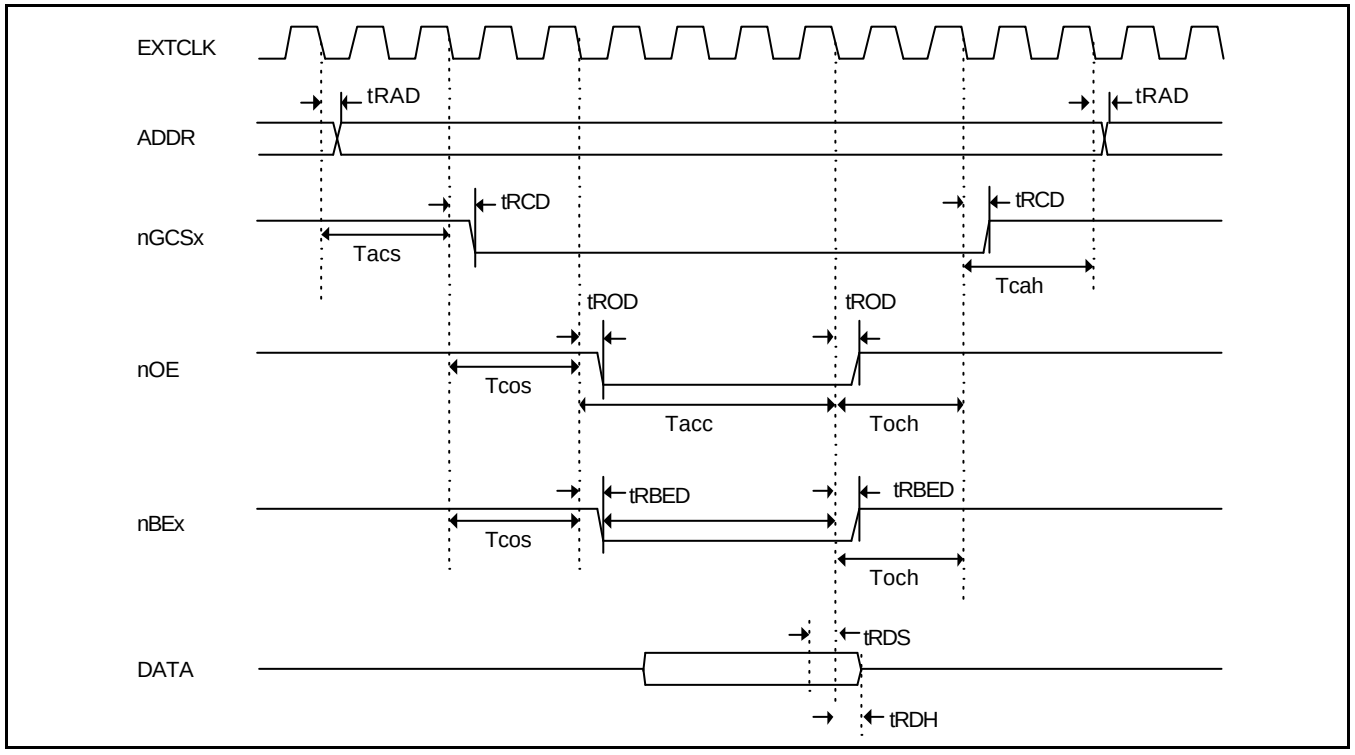


Figure 19-13. ROM/SRAM READ Timing (II)
(Tacs=2, Tcos=2, Tacc=4, Toch=2, Tcah=2cycle, PMC=0, ST=1)

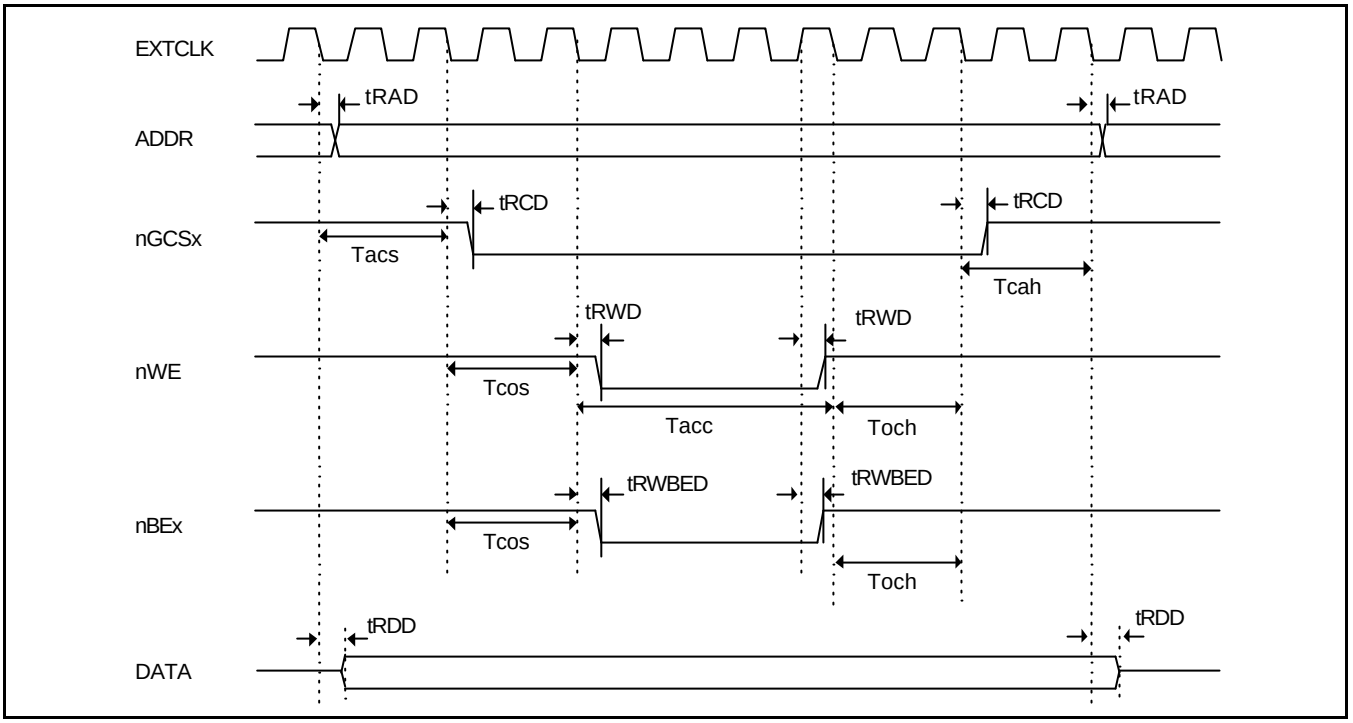


Figure 19-14. ROM/SRAM WRITE Timing (I)
(Tacs=2,Tcos=2,Tacc=4,Toch=2, Tcah=2, PMC=0, ST=0)

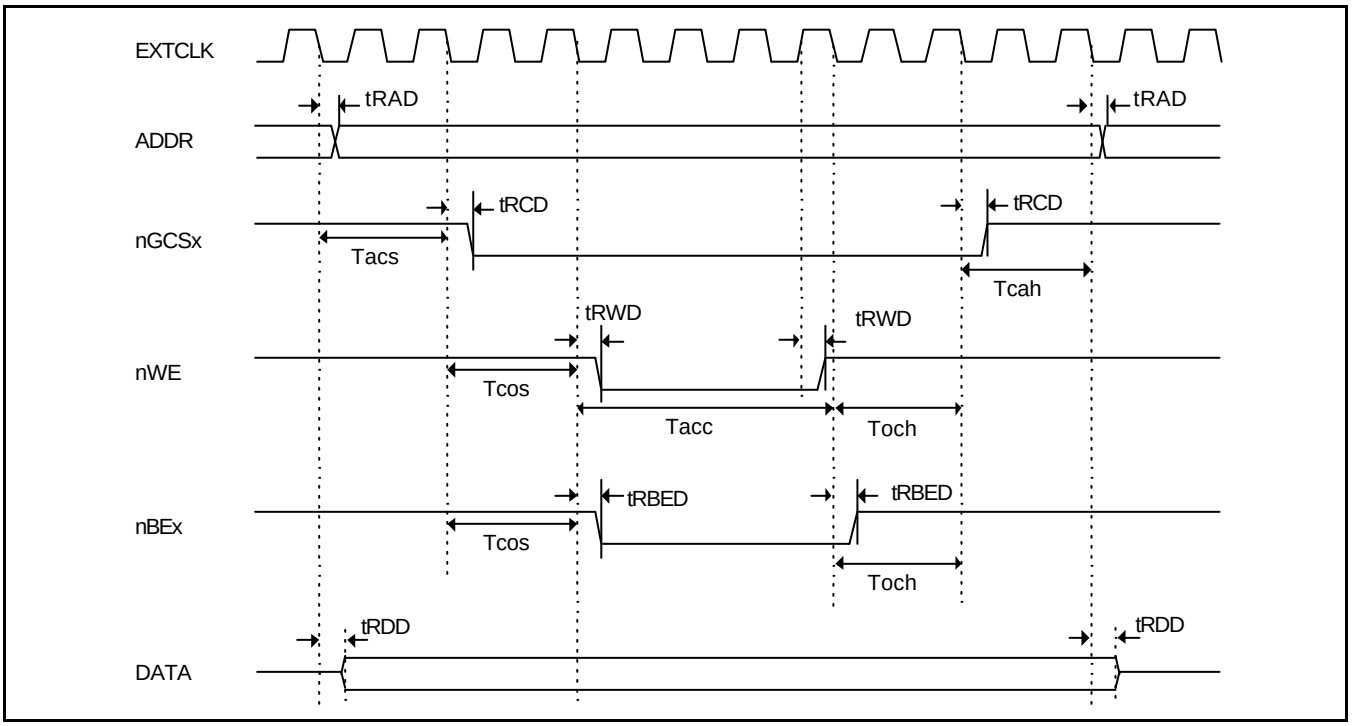


Figure 19-15. ROM/SRAM WRITE Timing (II)
(Tacs=2, Tcos=2, Tacc=4, Toch=2, Tcah=2, PMC=0, ST=1)

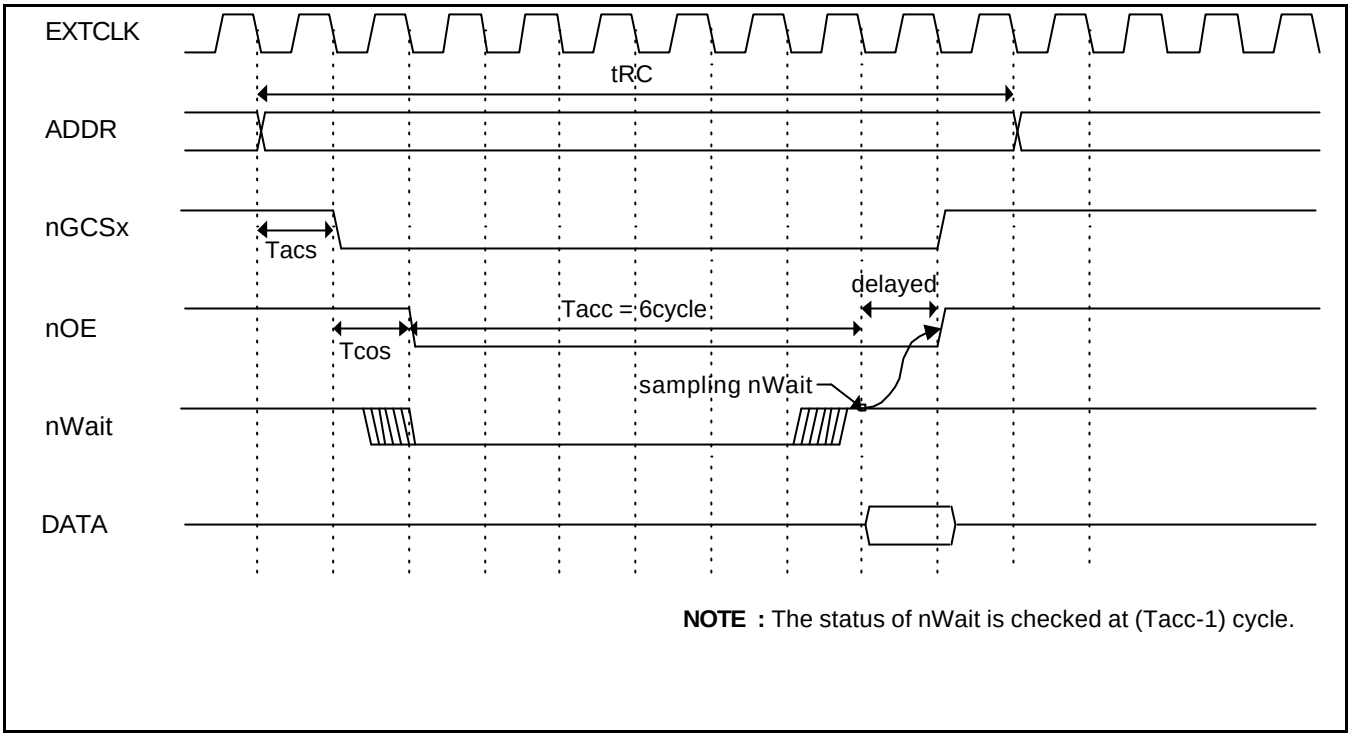


Figure 19-16. External nWAIT READ Timing
(Tacs=0, Tcos=0, Tacc=6, Toch=0, Tcah=0, PMC=0, ST=0)

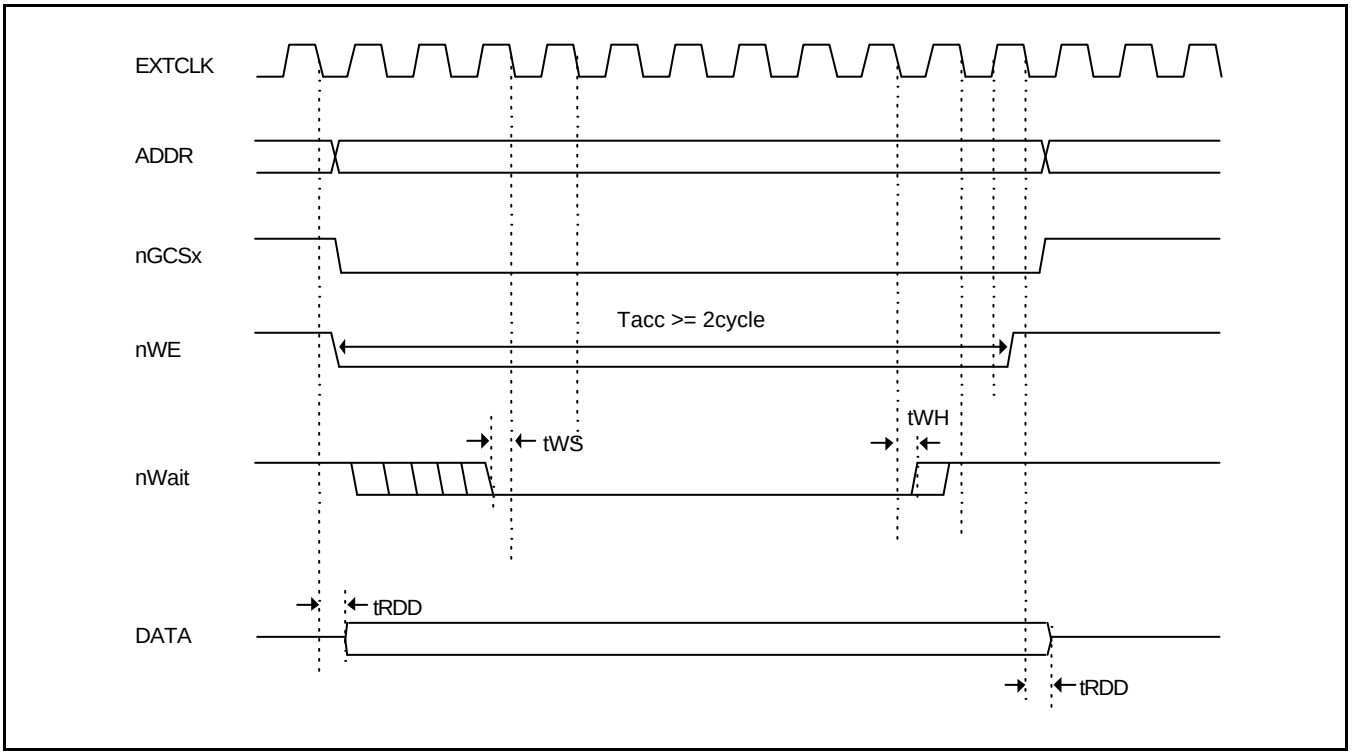


Figure 19-17. External nWAIT WRITE Timing
(Tacs=0, Tcos=0, Tacc=4, Toch=0, Tcah=0, PMC=0, ST=0)

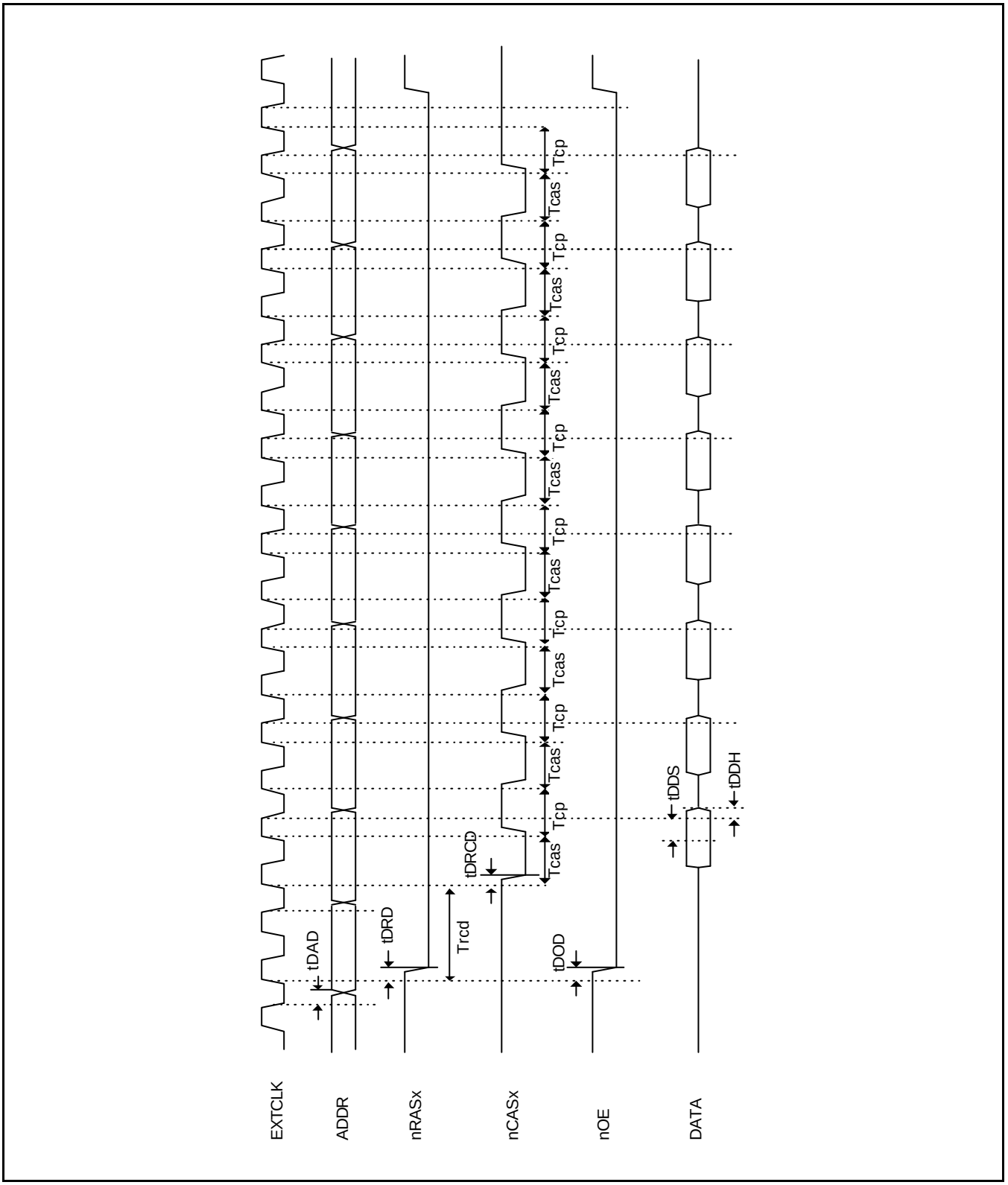


Figure 19-18. DRAM (EDO) Burst READ Timing
(Trcd=2, Tcas=1, Tcp=1, Trp=3.5, MT=10b, DW = 16bit)

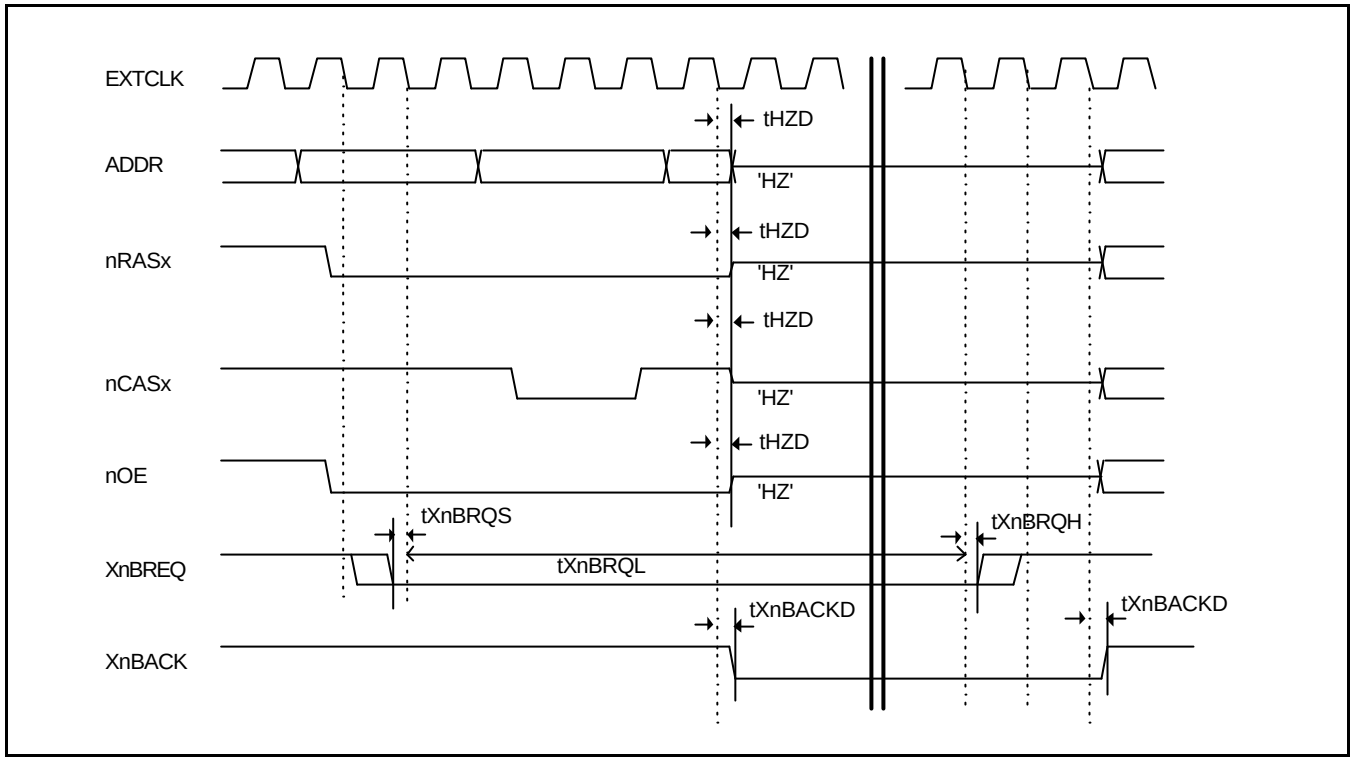


Figure 19-19. External Bus Request in DRAM Cycle (Trcd=3, Tcas=2, Tcp=1, Trp=4.5)

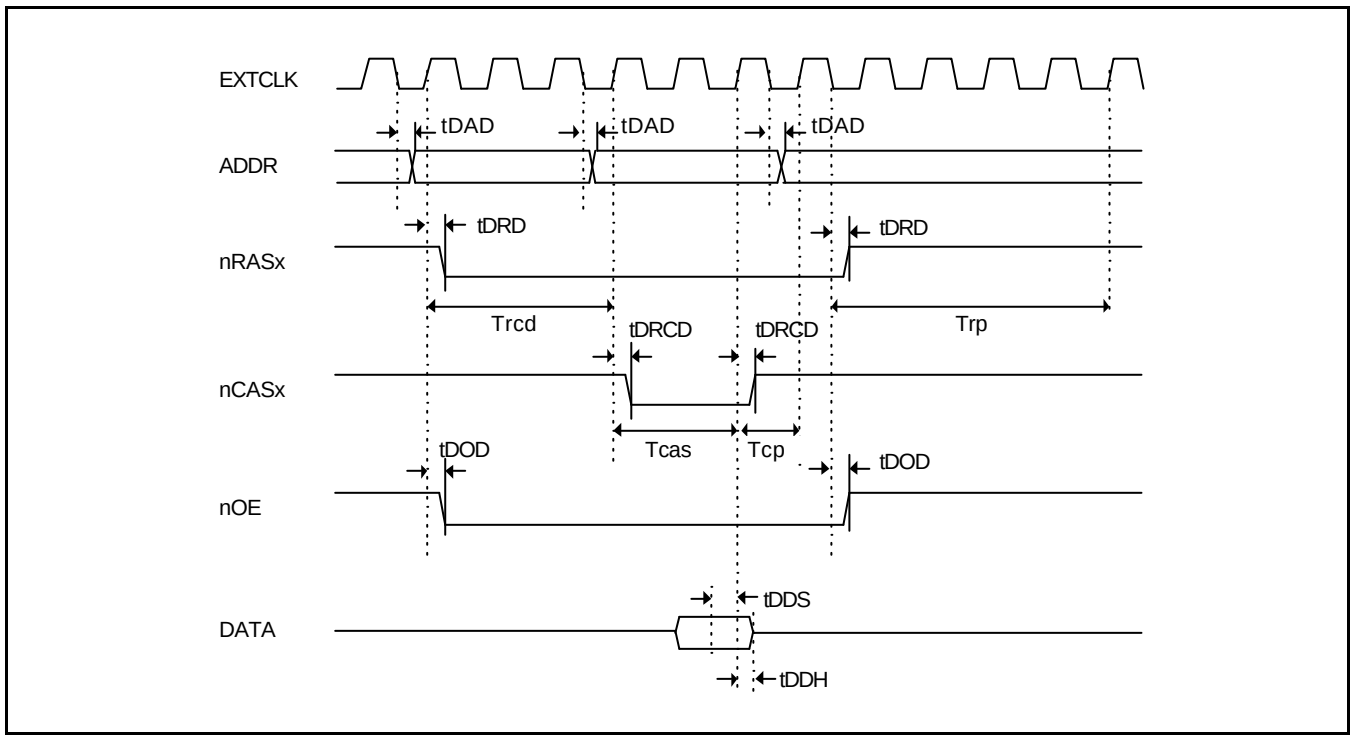


Figure 19-20. DRAM(FP) Single READ Timing (Trcd=3, Tcas=2, Tcp=1, Trp=4.5, MT=01b)

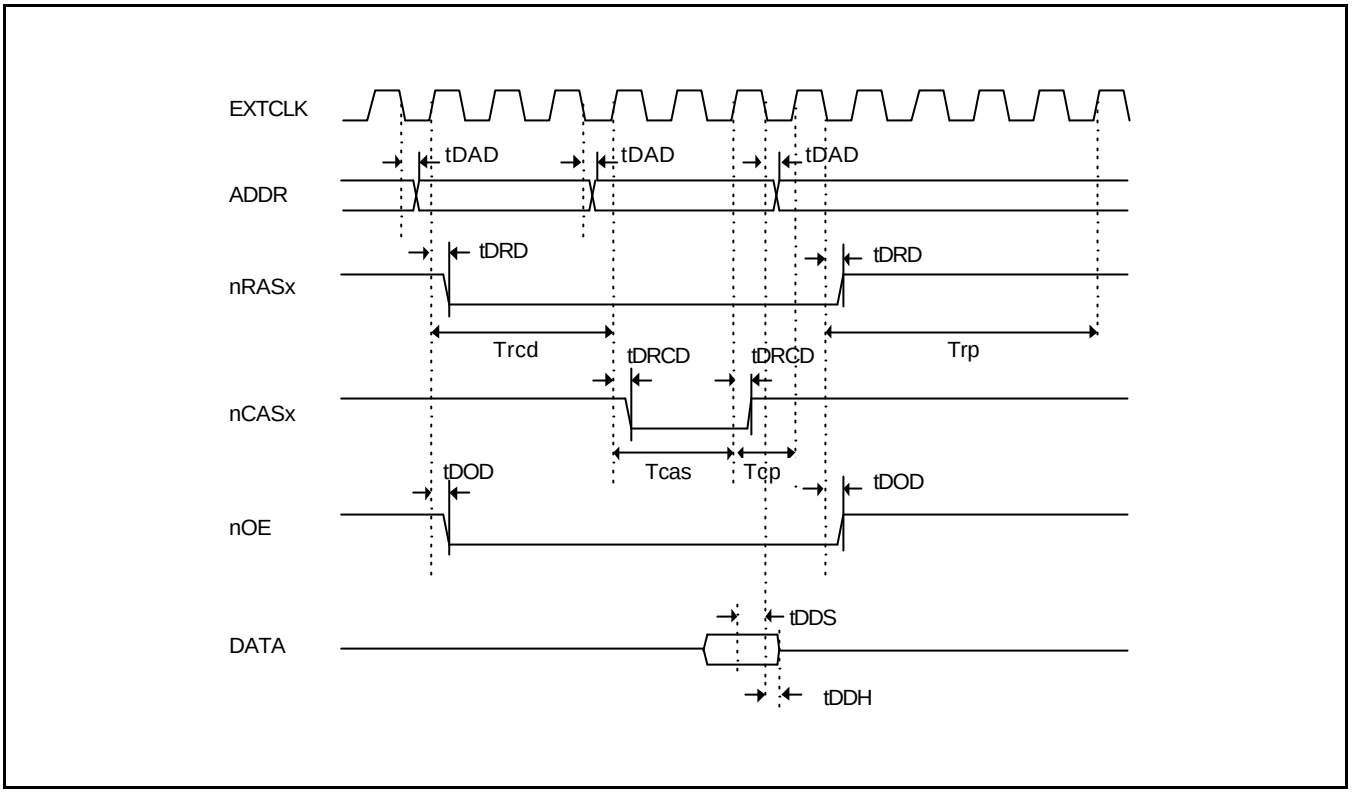


Figure 19-21. DRAM(EDO) Single READ Timing (Trcd=3, Tcas=2, Tcp=1, Trp=4.5, MT=10b)

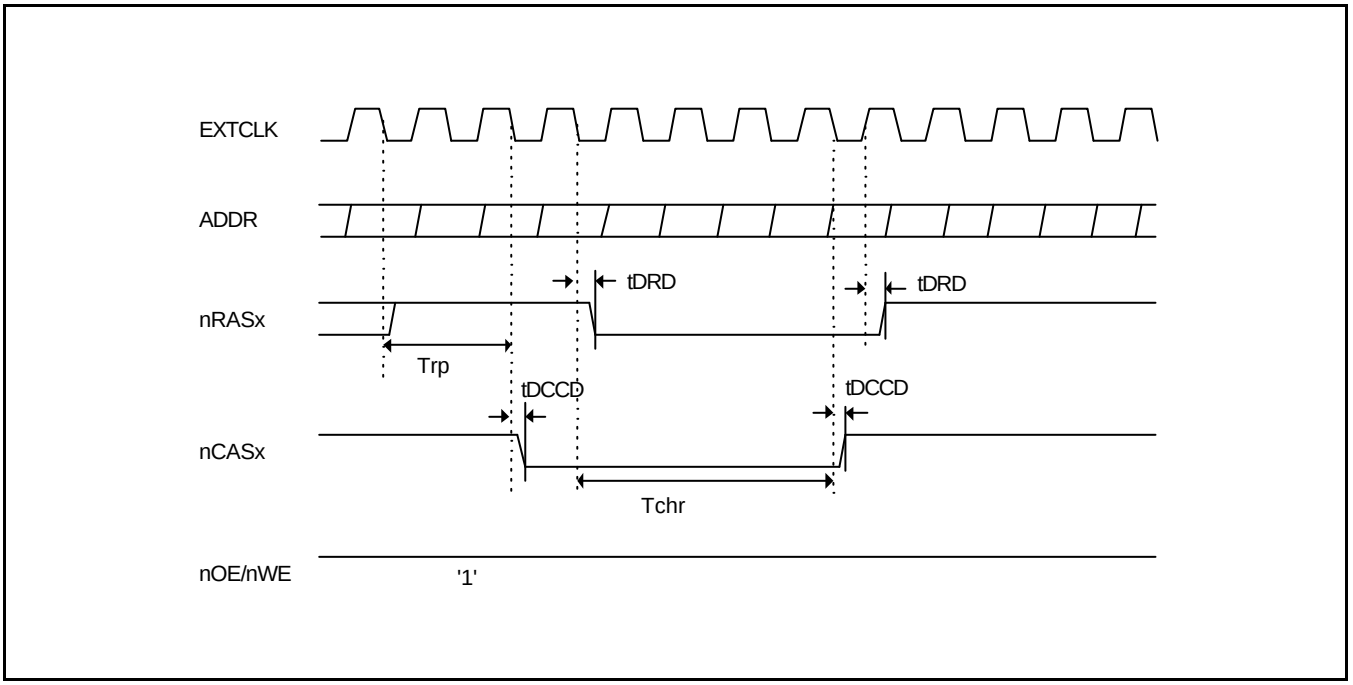


Figure 19-22. DRAM CBR Refresh Timing (Tchr=4)

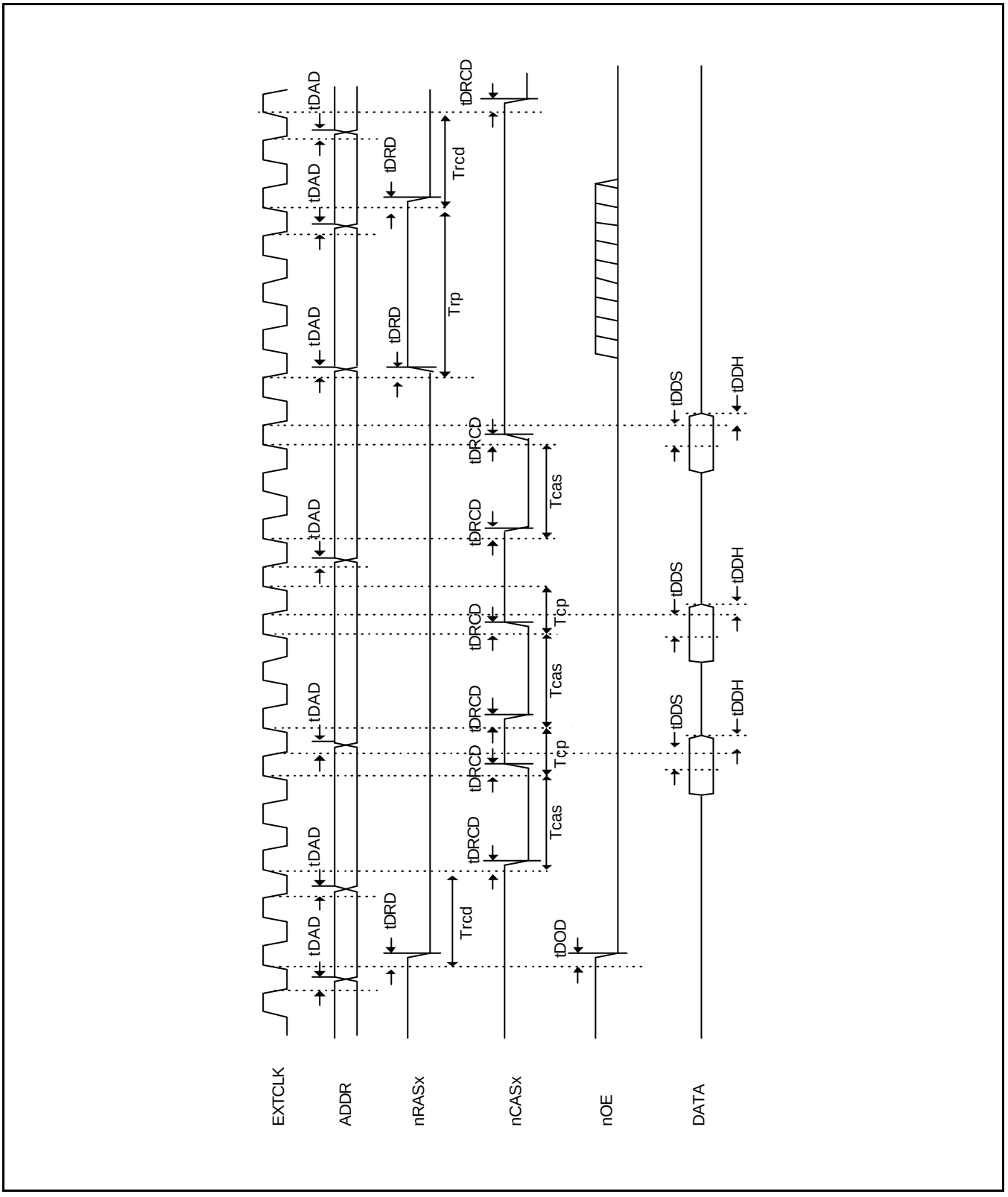


Figure 19-23. DRAM(EDO) Page Hit-Miss READ Timing (Trcd=2, Tcas=2, Tcp=1, Trp=3.5, MT=10b)

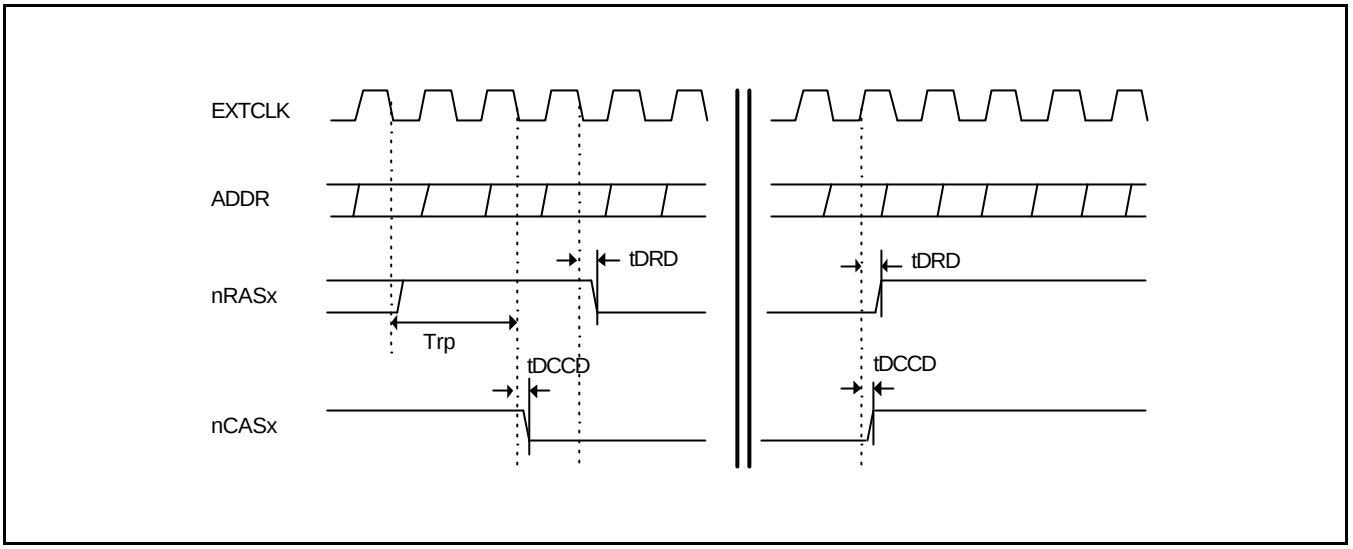


Figure 19-24. DRAM Self Refresh Timing

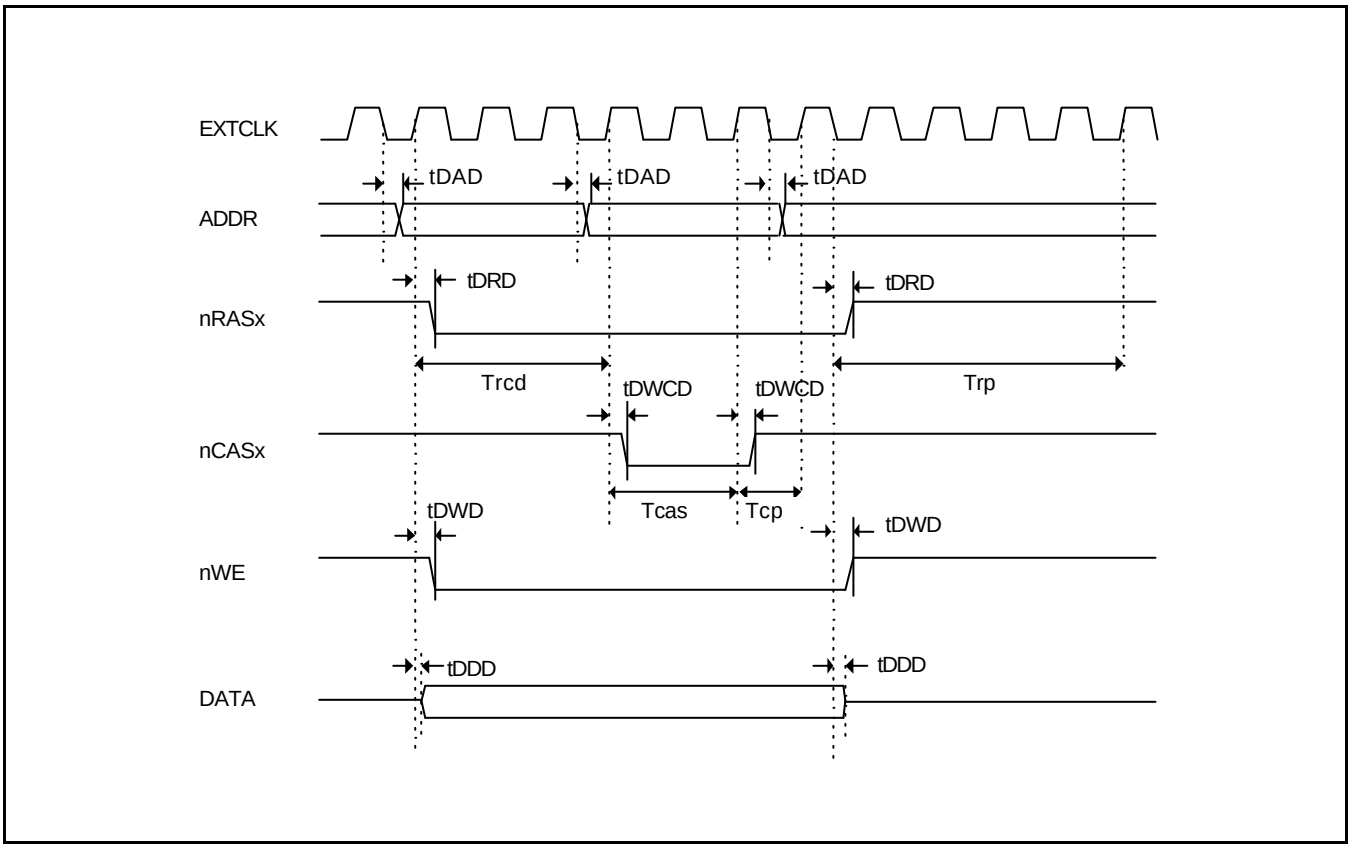


Figure 19-25. DRAM(FP/EDO) Single Write Timing (Trcd=3, Tcas=2, Tcp=1, Trp=4.5, MT=01/10b)

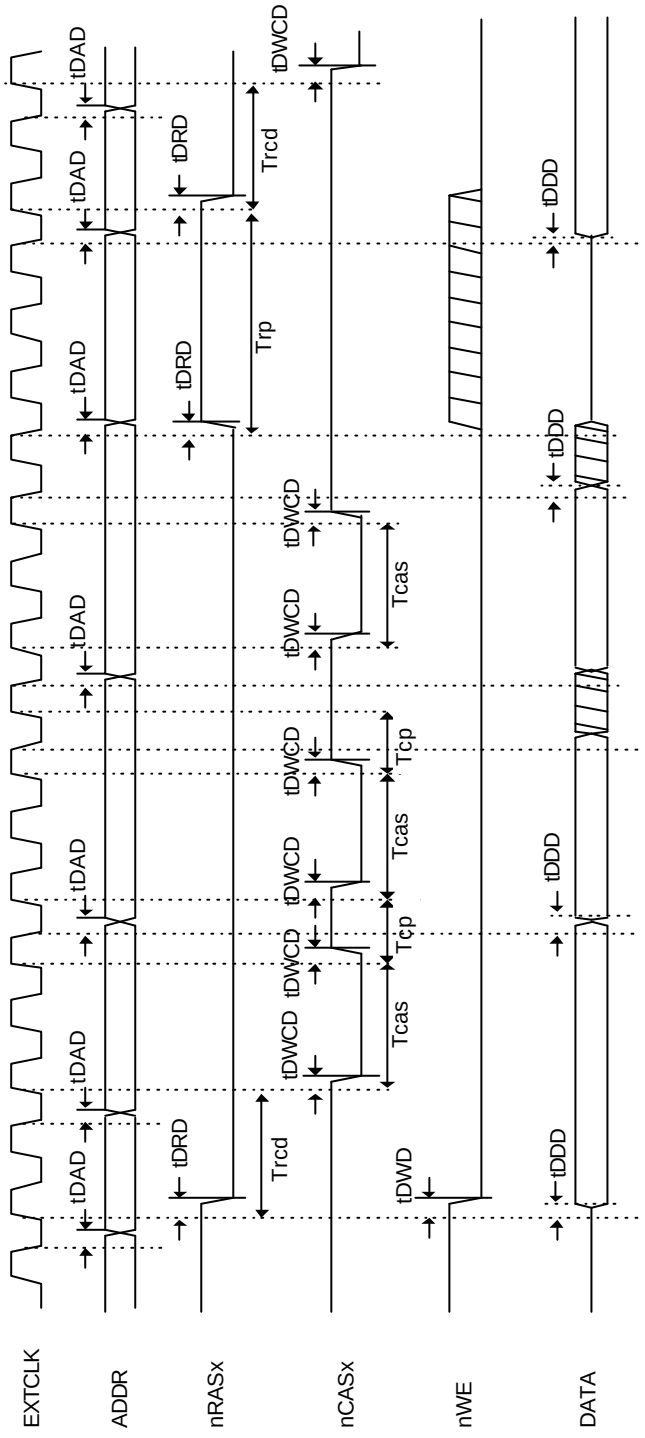


Figure 19-26. DRAM(FP/EDO) Page Hit-Miss Write Timing (Trcd=2, Tcas=2, Tcp=1, Trp=3.5, MT=01/10b)

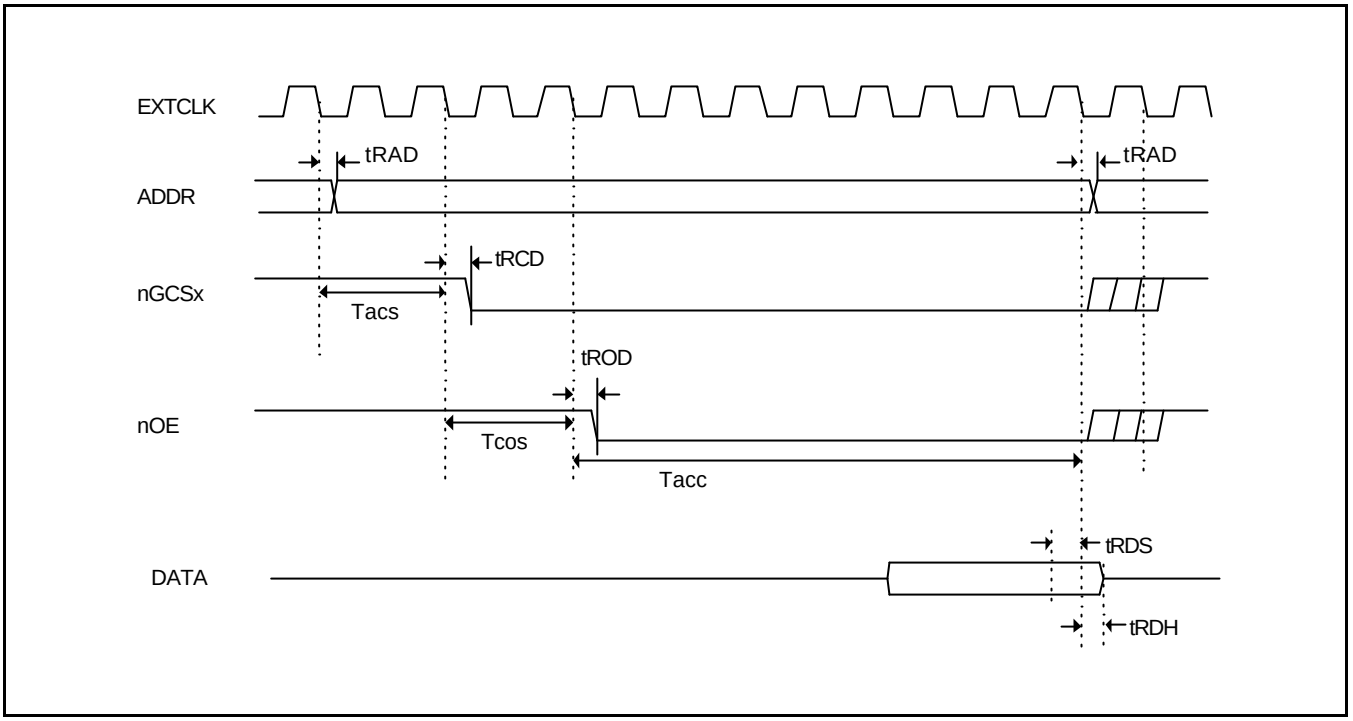


Figure 19-27. Masked-ROM Single READ Timing (Tacs=2, Tcos=2, Tacc=8, PMC=01/10/11b)

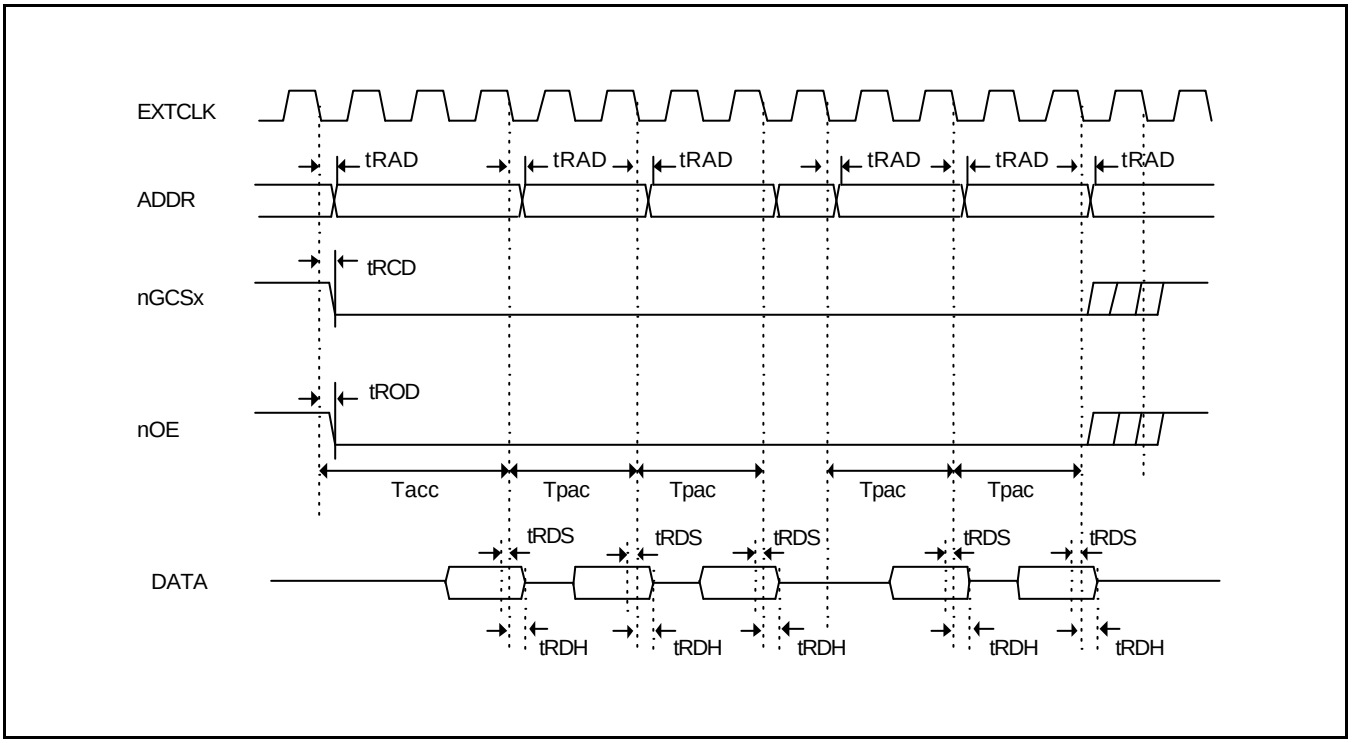


Figure 19-28. Masked-ROM Consecutive READ Timing (Tacs=0, Tcos=0, Tacc=3, Tpac=2, PMC=01/10/11b)

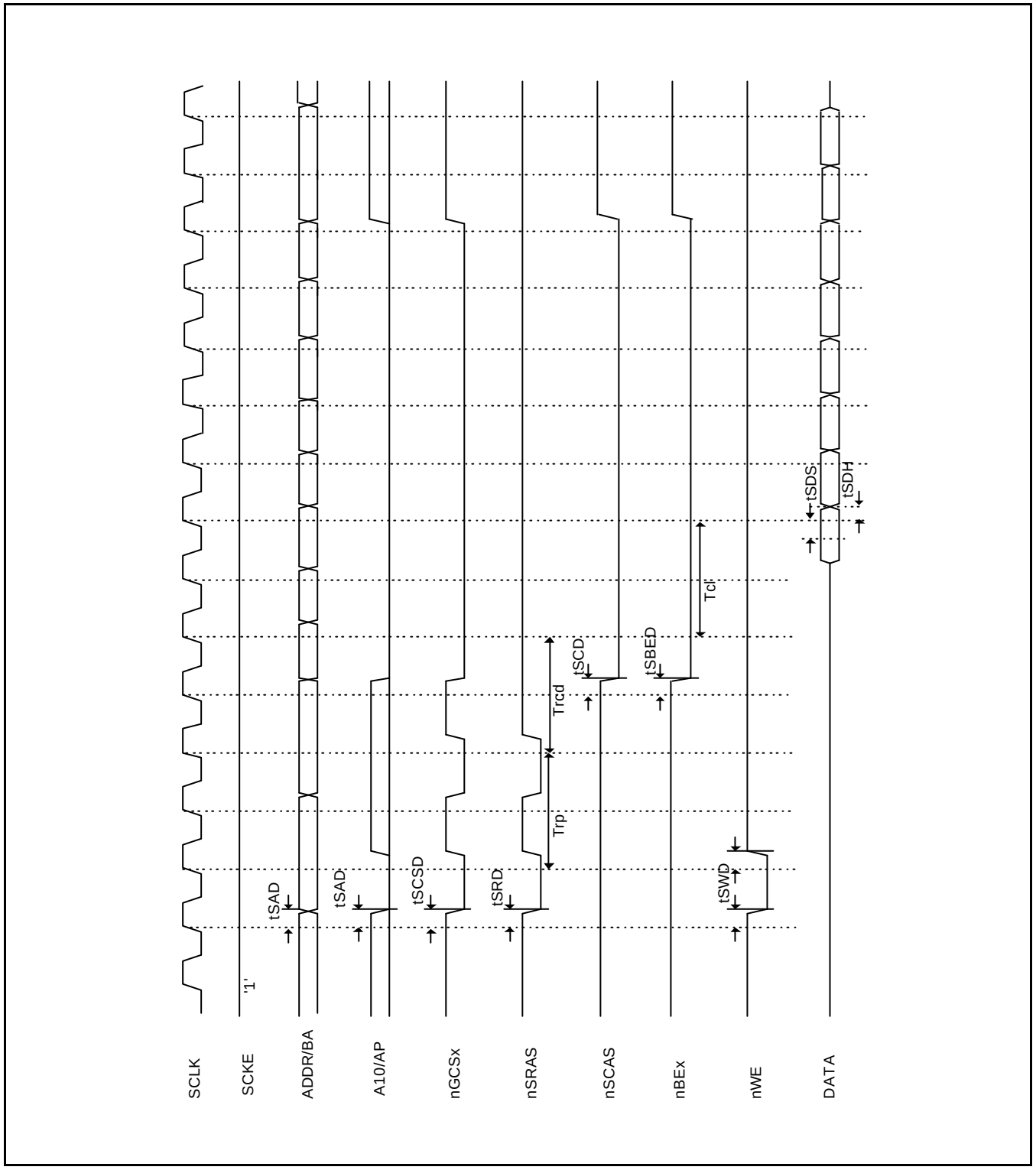


Figure 19-29. SDRAM Single Burst READ Timing (Trp=2, Trcd=2, Tcl=2, DW=16bit)

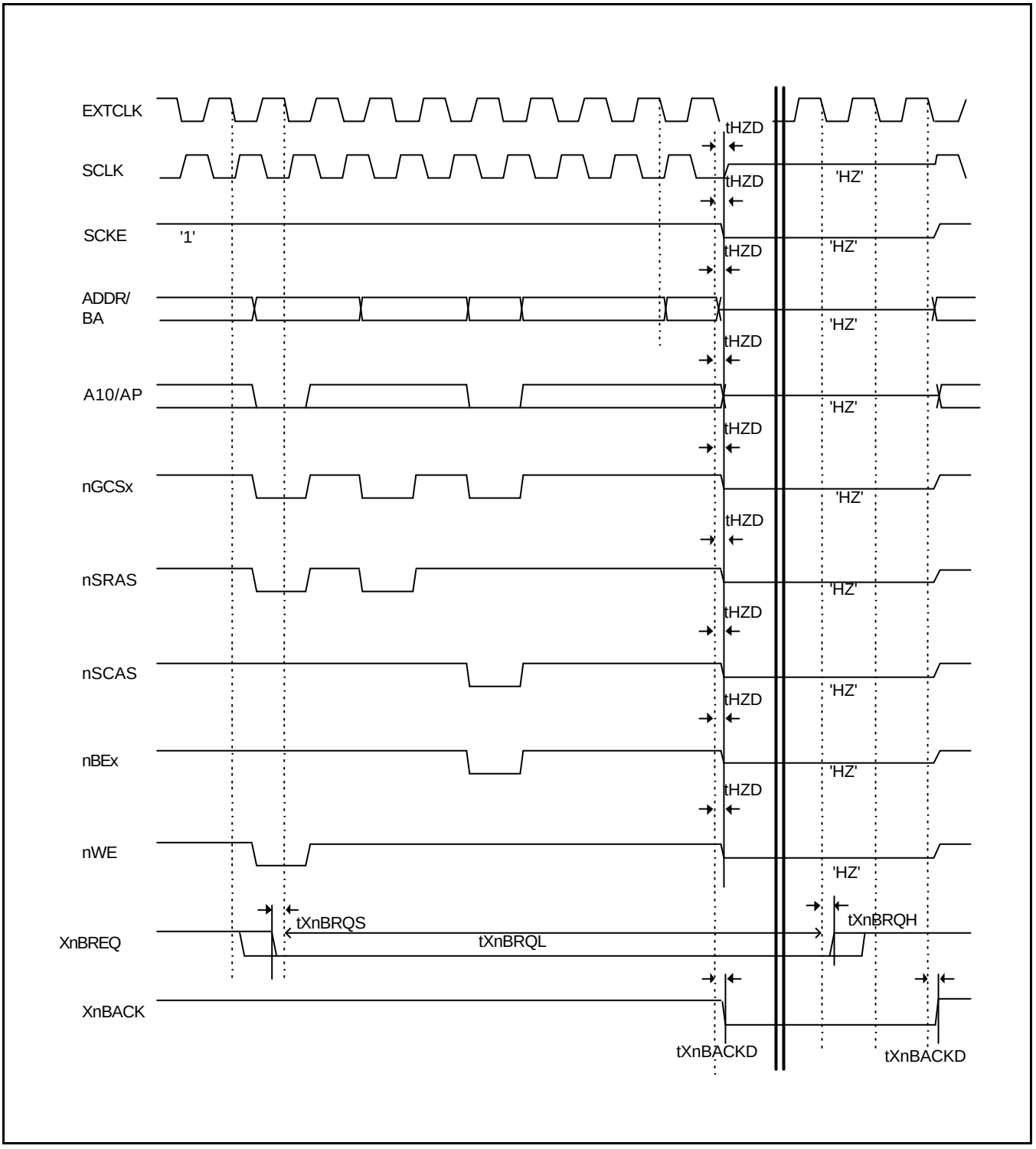


Figure 19-30. External Bus Request in SDRAM Timing (Trp=2, Trcd=2, Tcl=2)

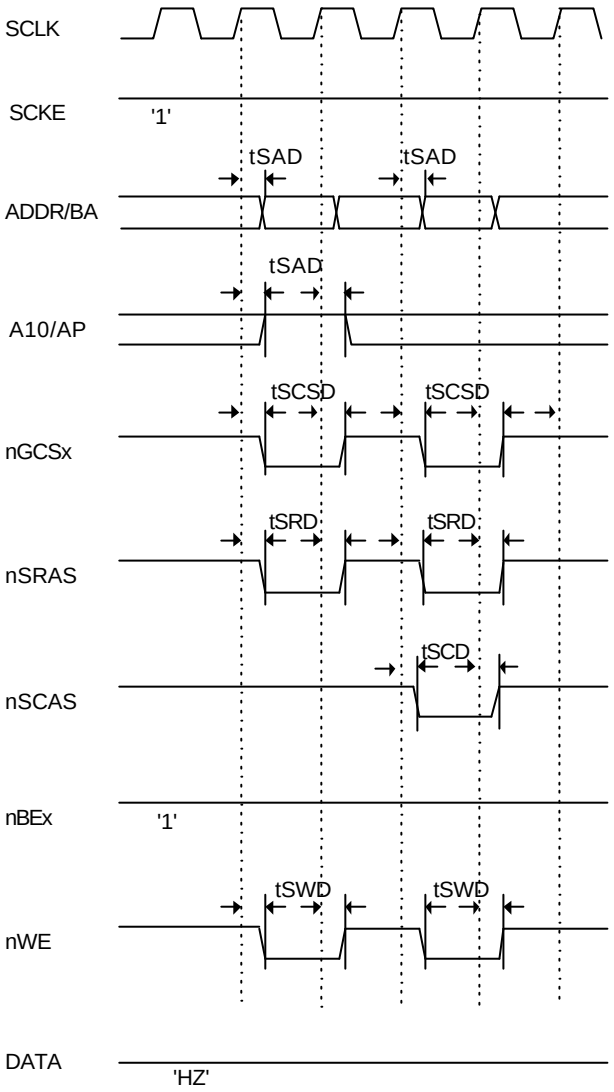


Figure 19-31. SDRAM MRS Timing

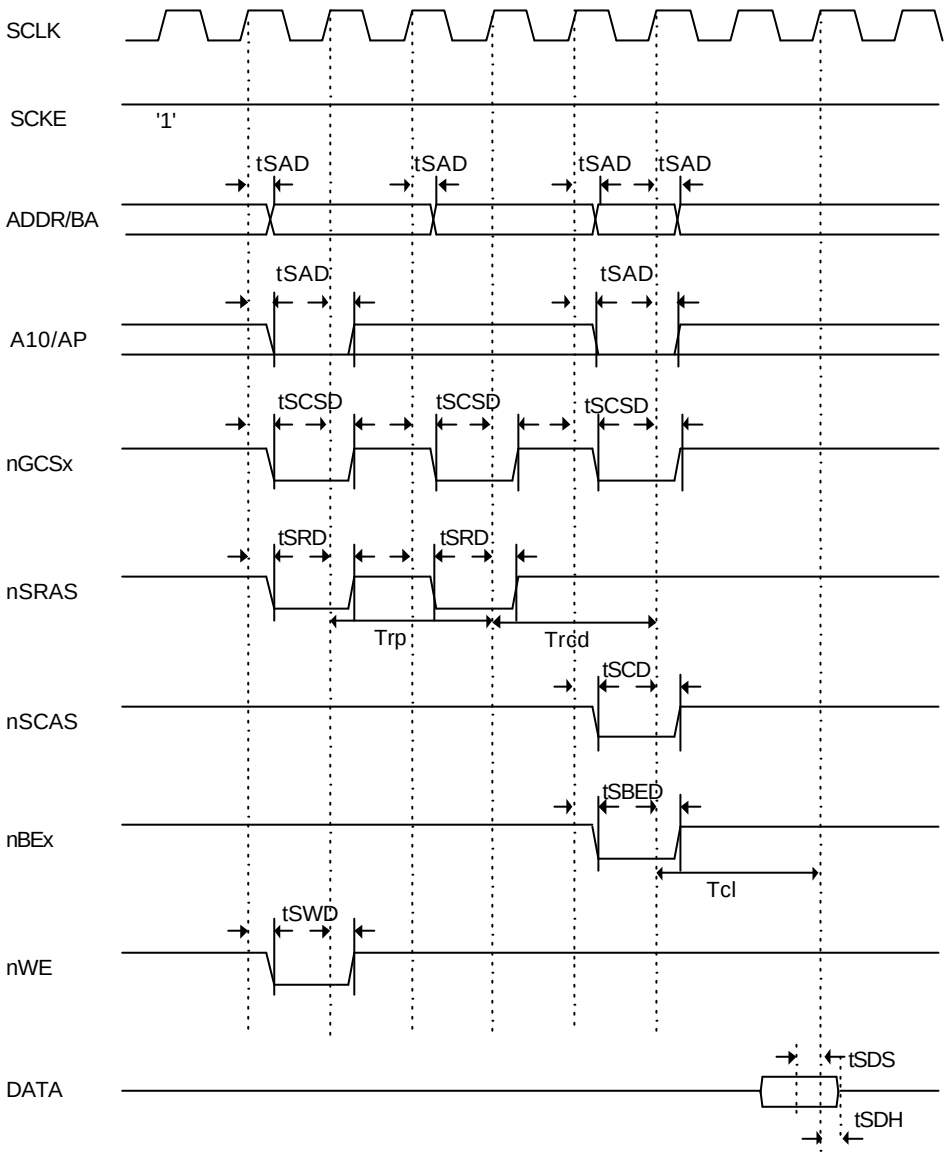


Figure 19-32. SDRAM Single READ Timing(I) (Trp=2, Trcd=2, Tcl=2)

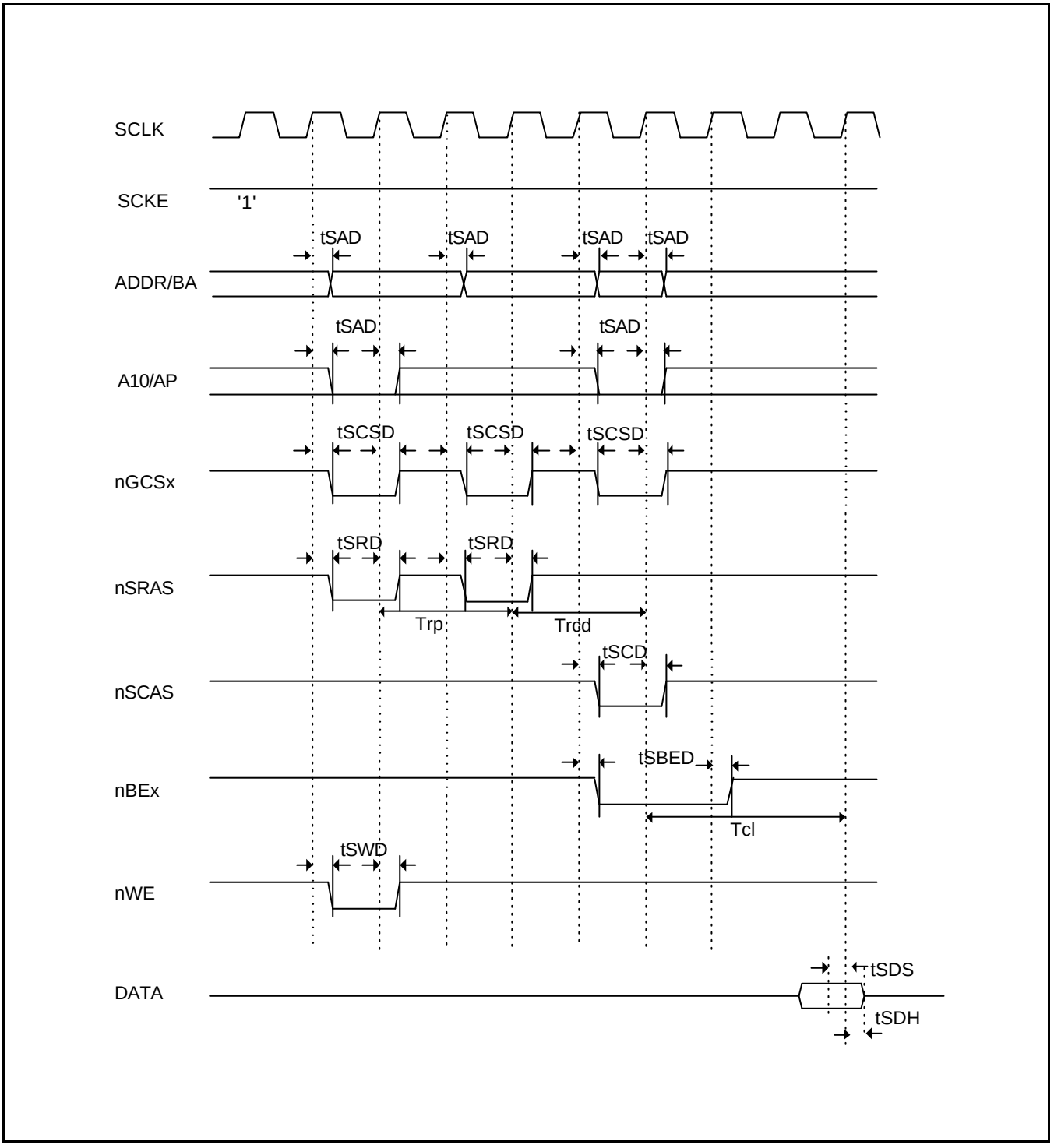
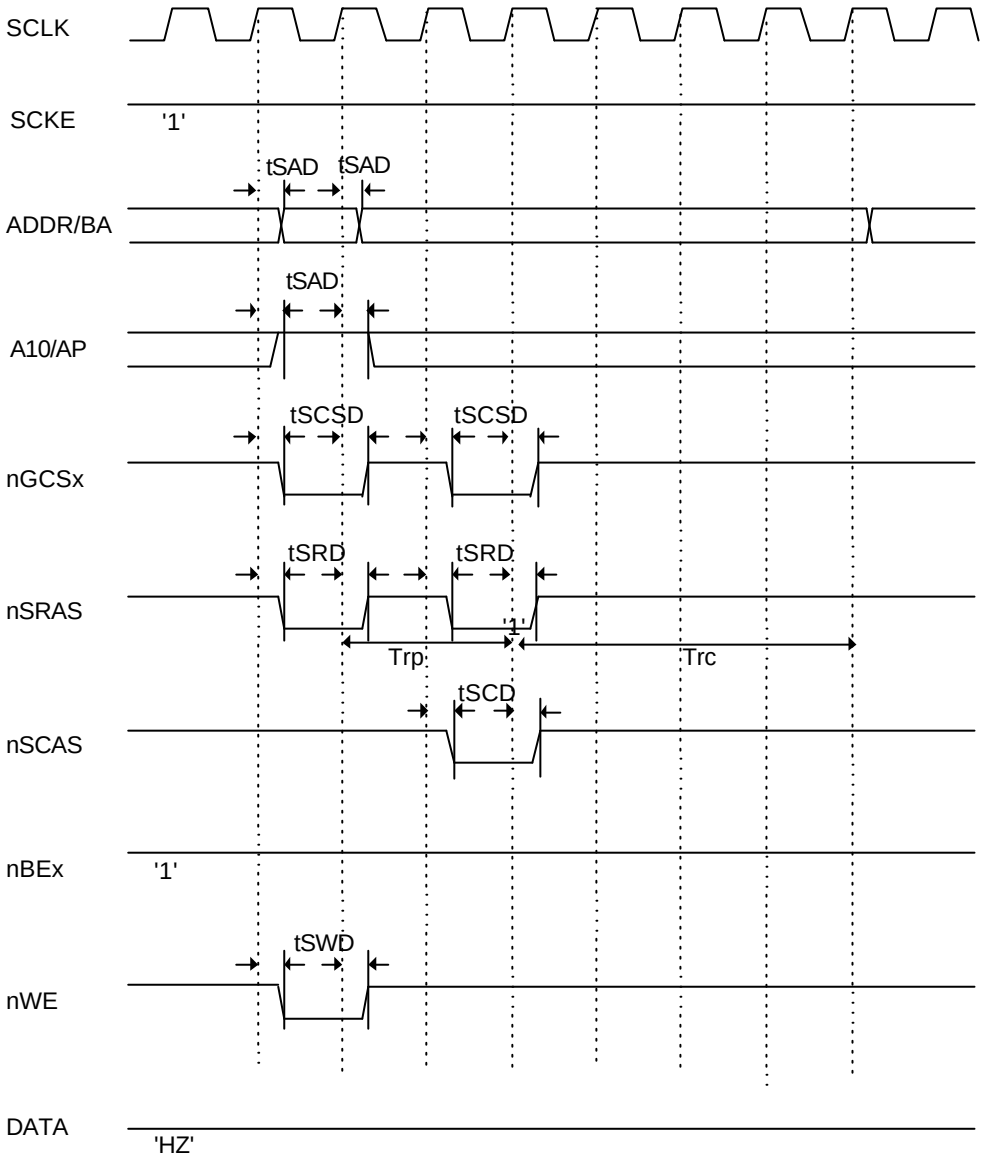


Figure 19-33. SDRAM Single READ Timing(II) (Trp=2, Trcd=2, Tcl=3)



NOTE: Before executing auto/self refresh command, all banks must be idle state.

Figure 19-34. SDRAM Auto Refresh Timing (Trp=2, Trc=4)

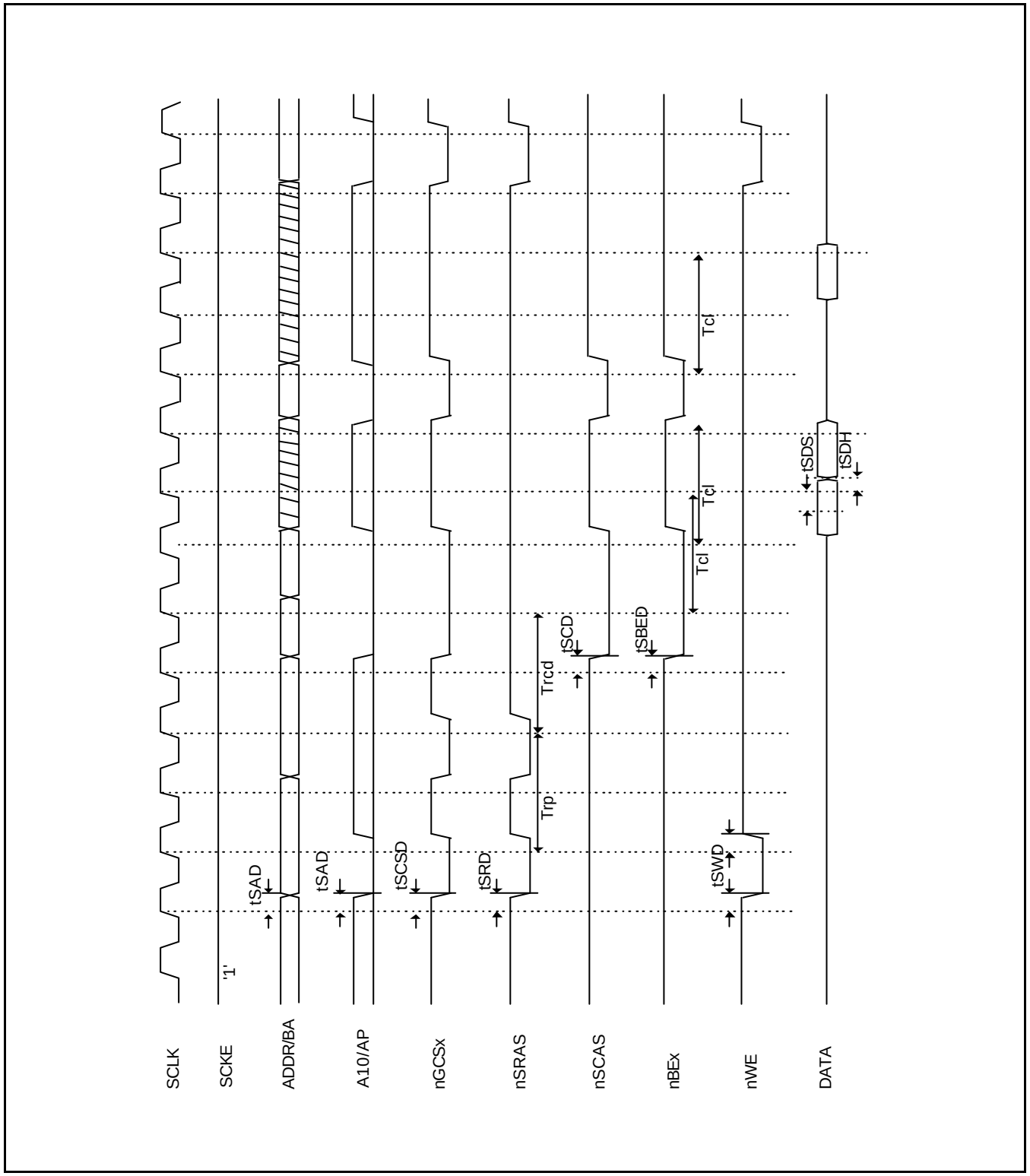
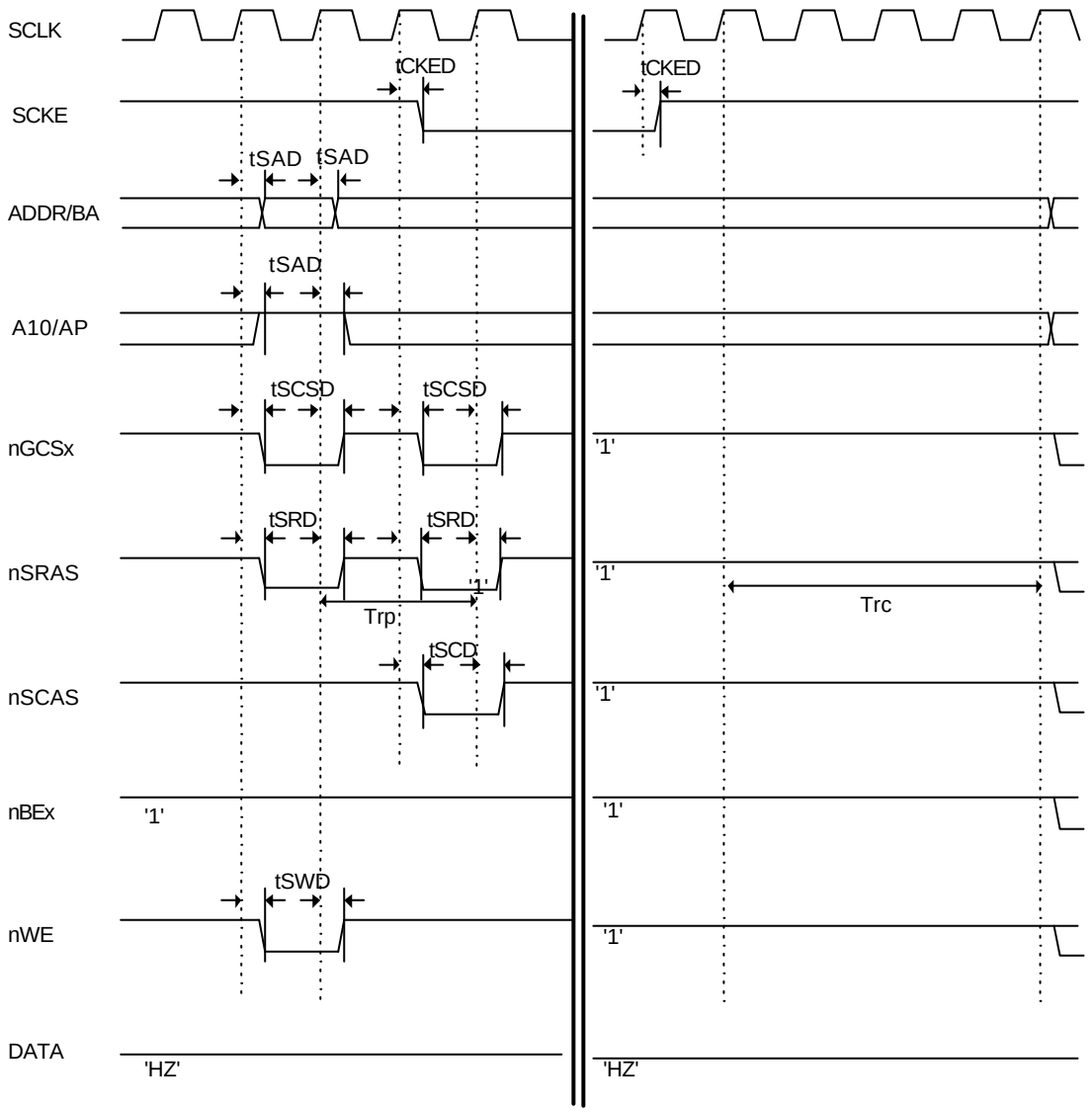


Figure 19-35. SDRAM Page Hit-Miss READ Timing (Trp=2, Trcd=2, Tcl=2)



NOTE: Before executing auto/self refresh command, all banks must be idle state.

Figure 19-36. SDRAM Self Refresh Timing (Trp=2, Trc=4)

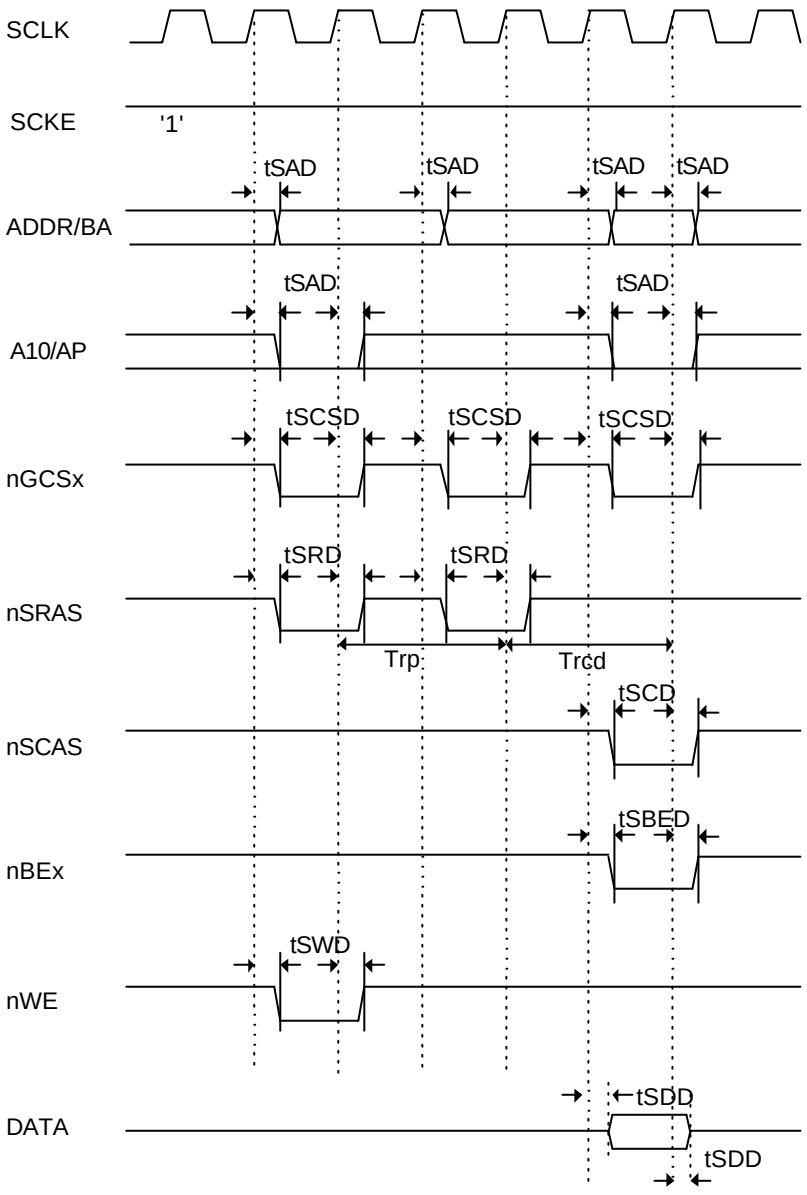


Figure 19-37. SDRAM Single Write Timing (Trp=2, Trcd=2)

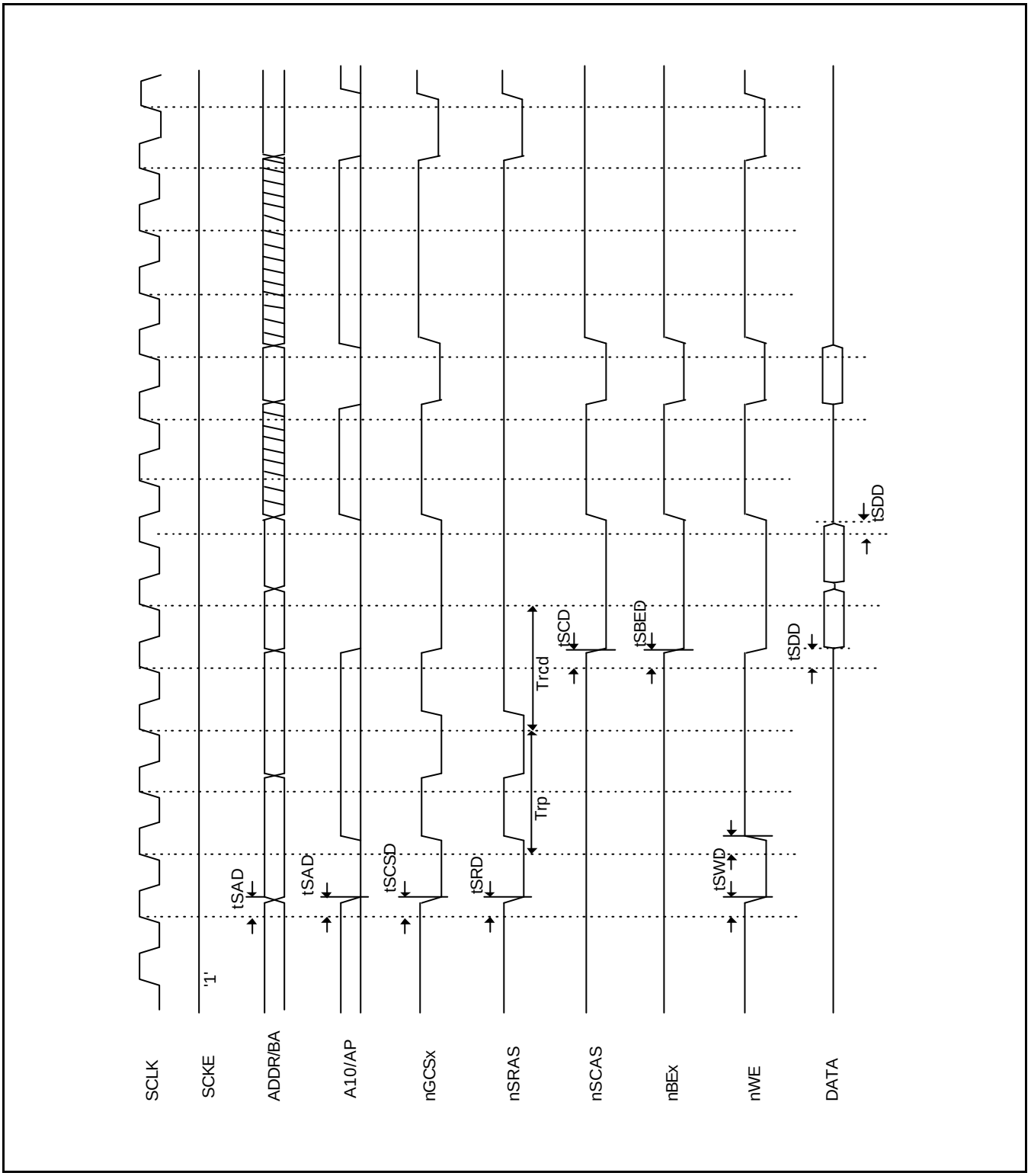


Figure 19-38. SDRAM Page Hit-Miss Write Timing (Trp=2, Trcd=2, Tcl=2)

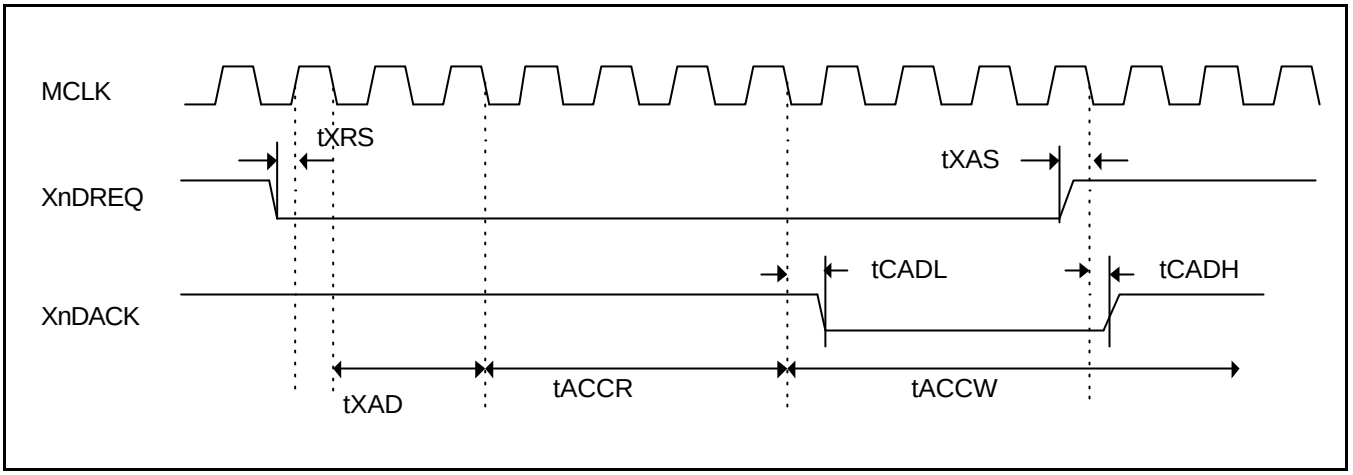


Figure 19-39. External DMA Timing (Handshake, Unit transfer/Block mode I)

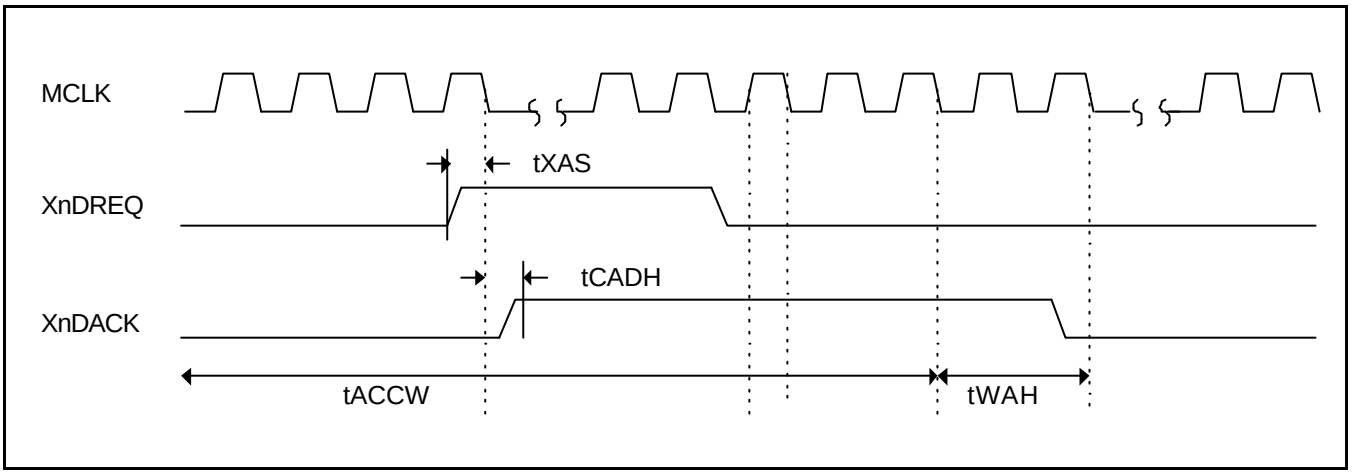


Figure 19-40. External DMA Timing (Handshake, Unit transfer/Block mode II)

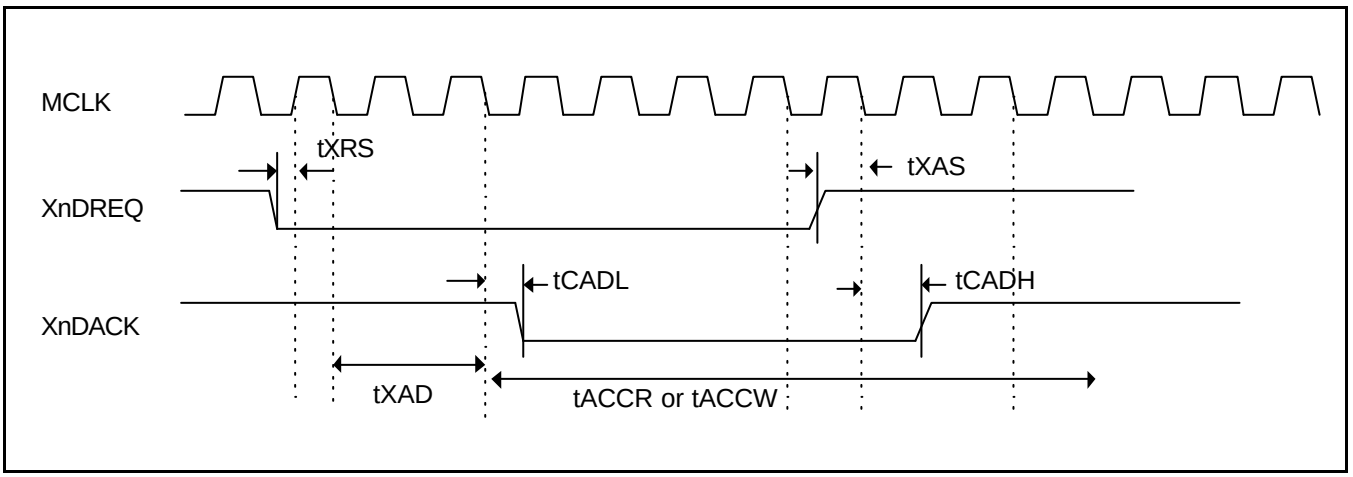


Figure 19-41. External DMA Timing (Handshake, On The Fly mode)

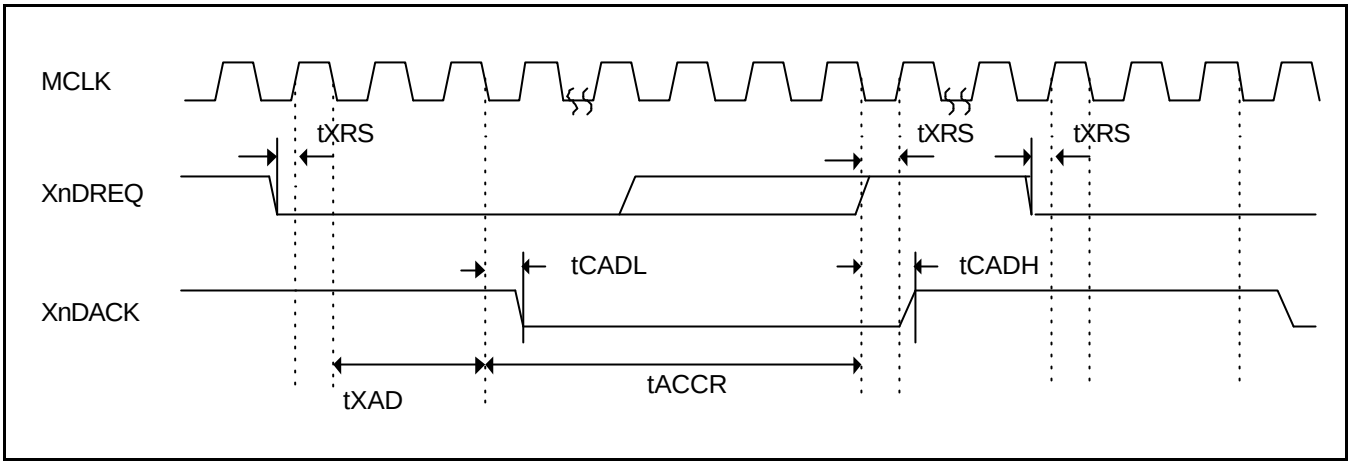


Figure 19-42. External DMA Timing (Single Step, Unit/Block/On-the-fly mode I)

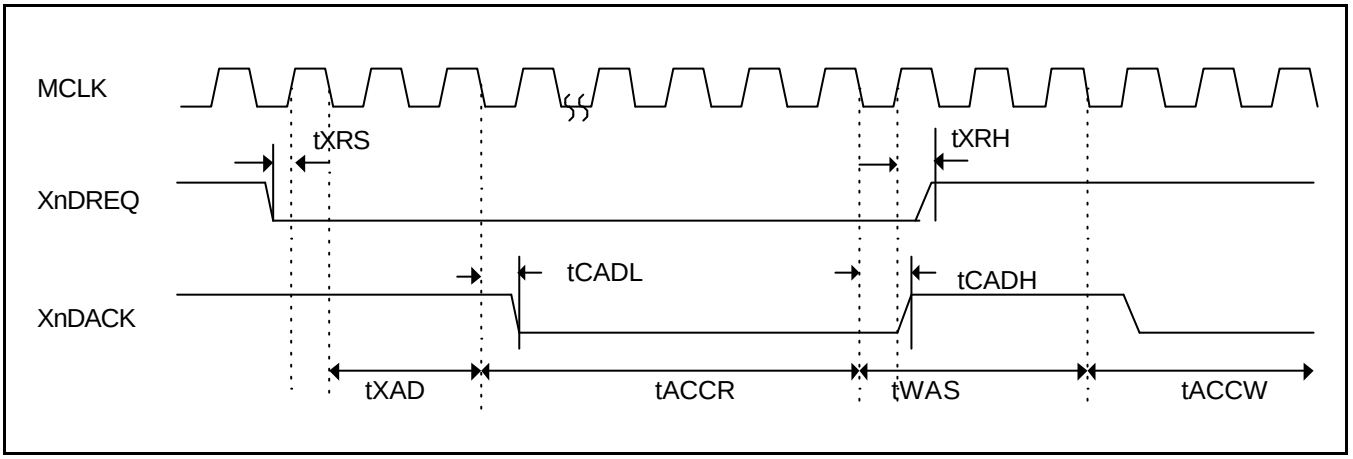


Figure 19-43. External DMA Timing (Single Step , Unit /Block/On-the-fly mode II)

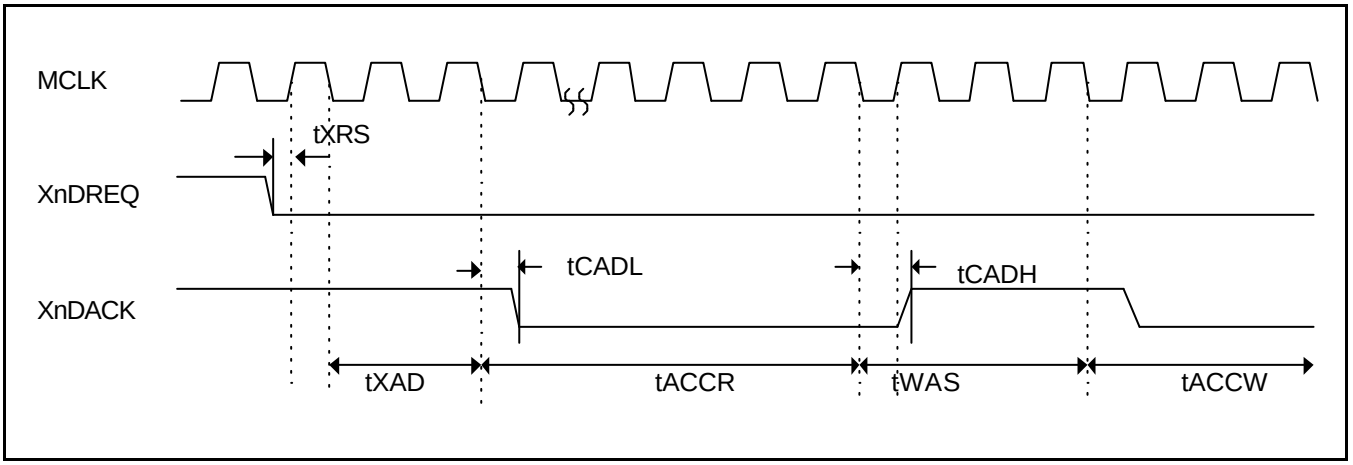


Figure 19-44. External DMA Timing (Single Step , Unit /Block/On-the-fly mode III)

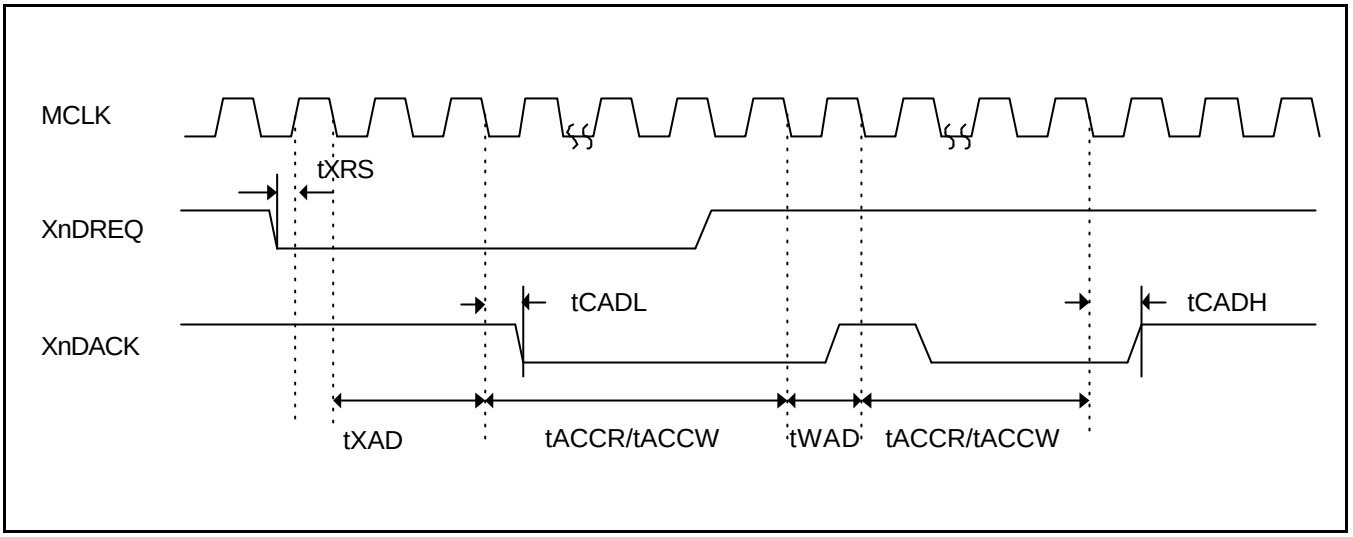


Figure 19-45. External DMA Timing (Demand, On The Fly mode I)

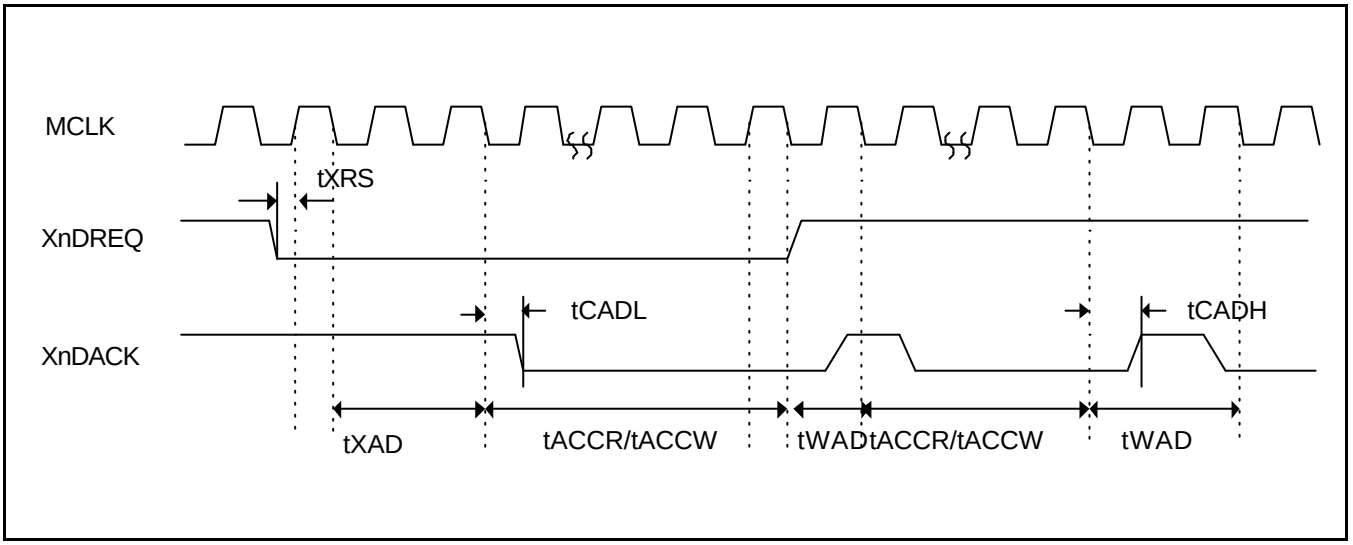


Figure 19-46. External DMA Timing (Demand, On The Fly mode II)

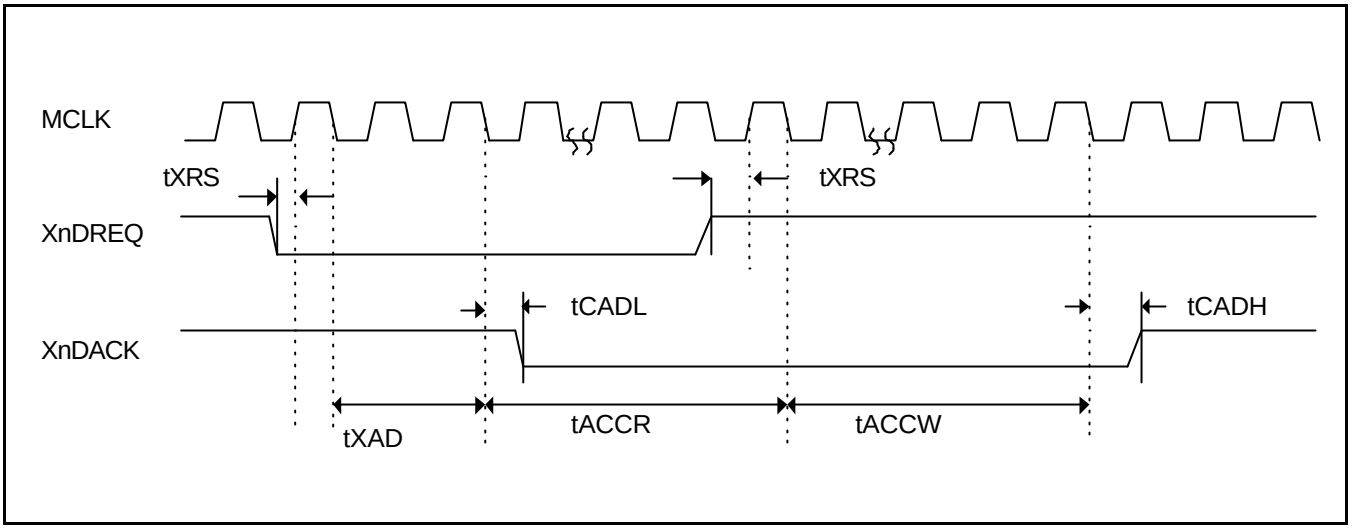


Figure 19-47. External DMA Timing (Demand, Unit transfer/Block mode I)

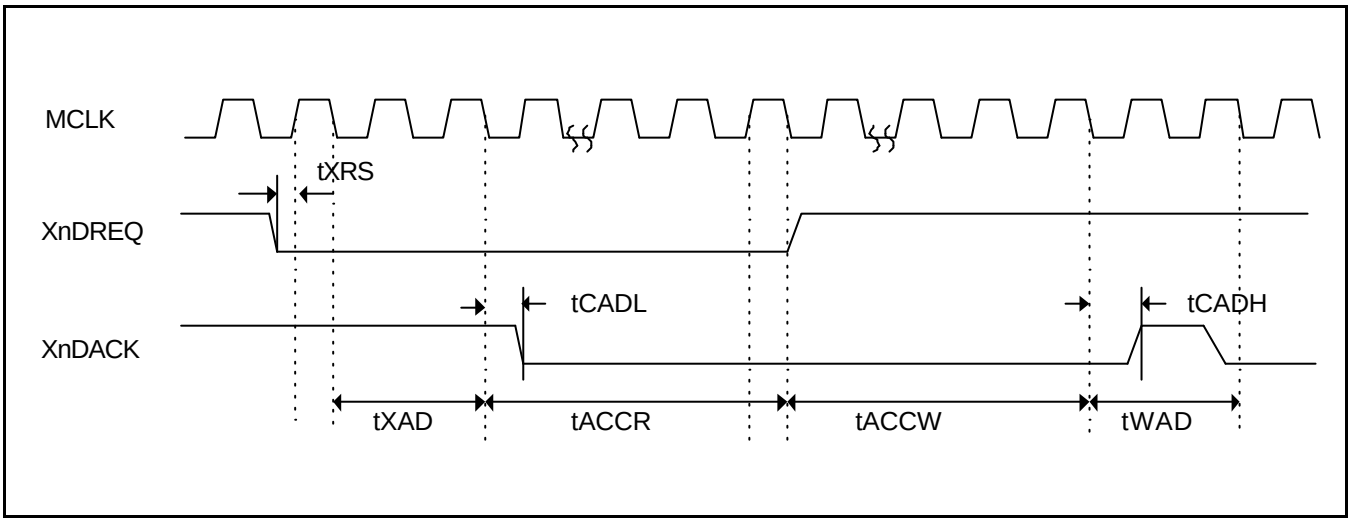


Figure 19-48. External DMA Timing (Demand, Unit transfer/Block mode II)

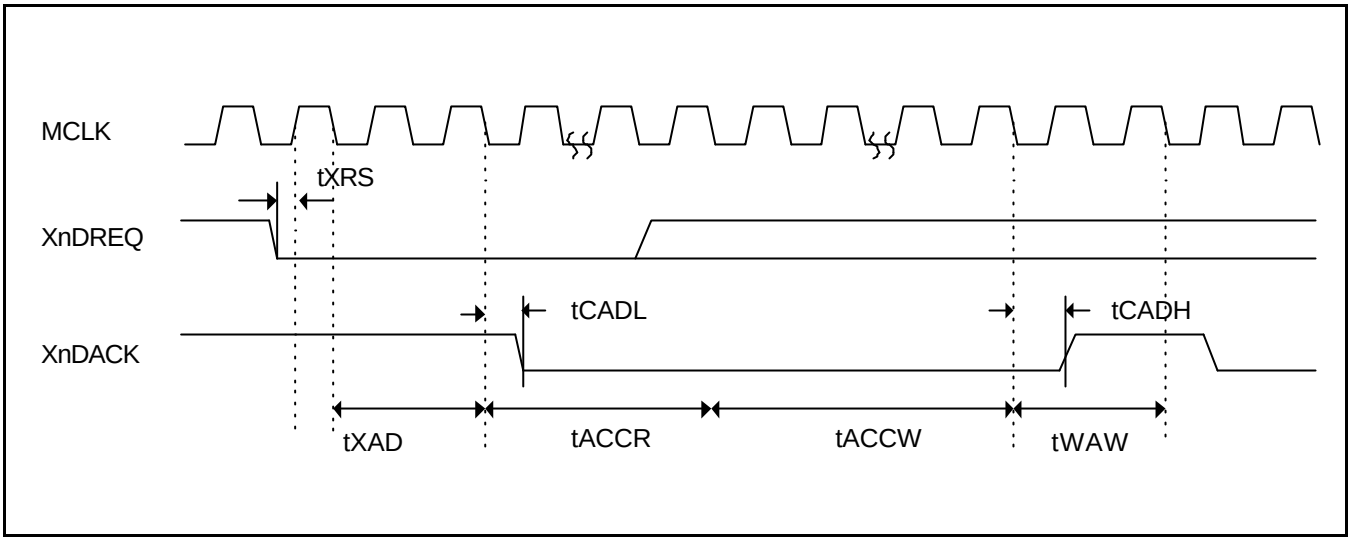


Figure 19-49. External DMA Timing (Whole, Unit transfer/Block mode)

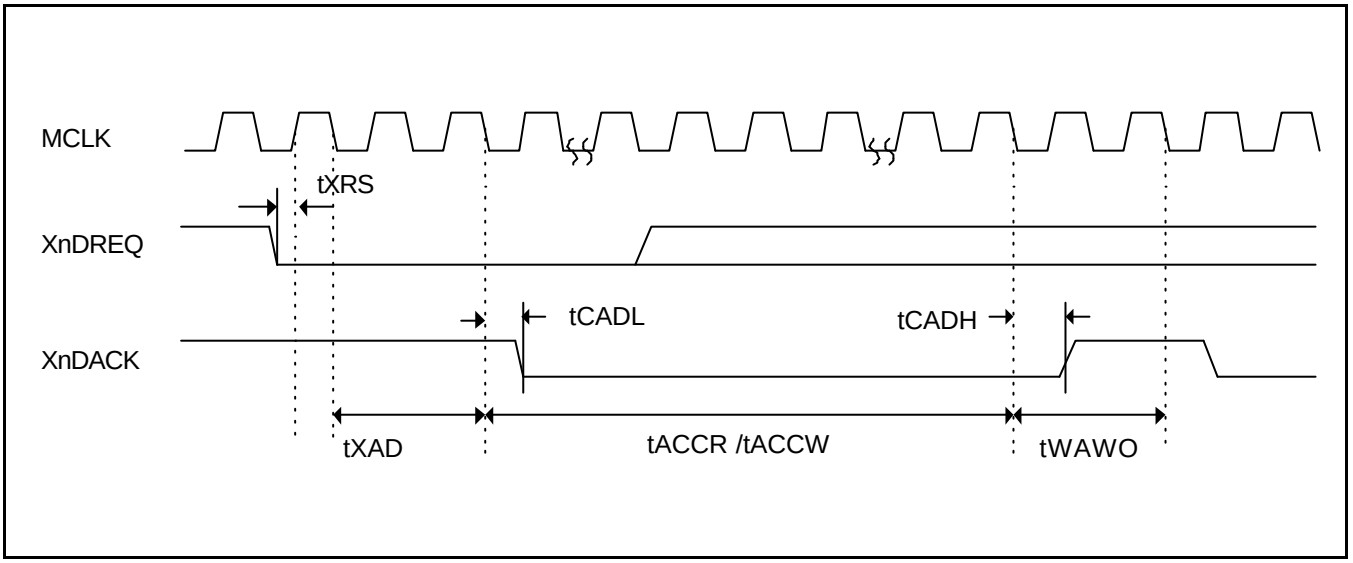


Figure 19-50. External DMA Timing (Whole, On The Fly mode)

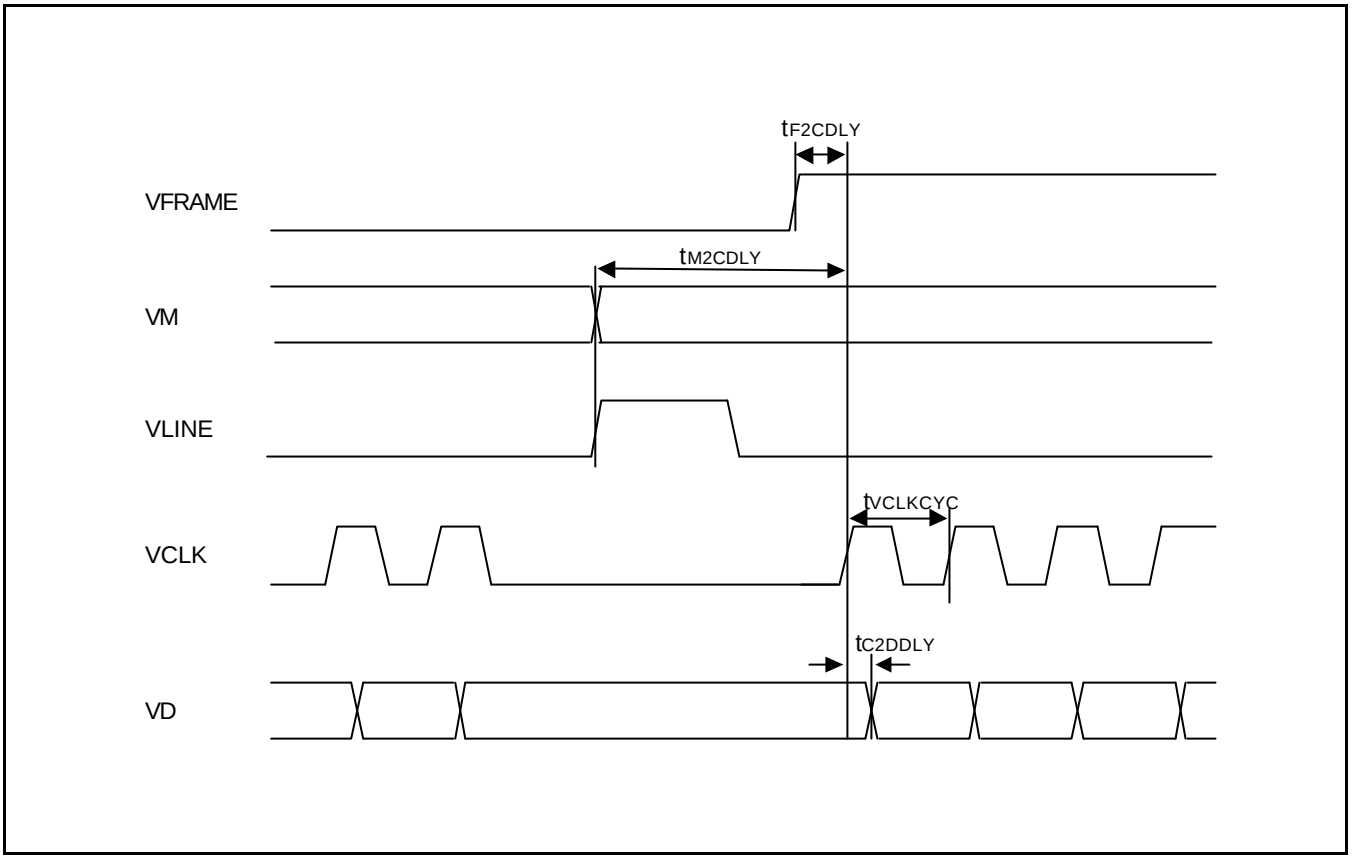


Figure 19-51. LCD Controller Timing

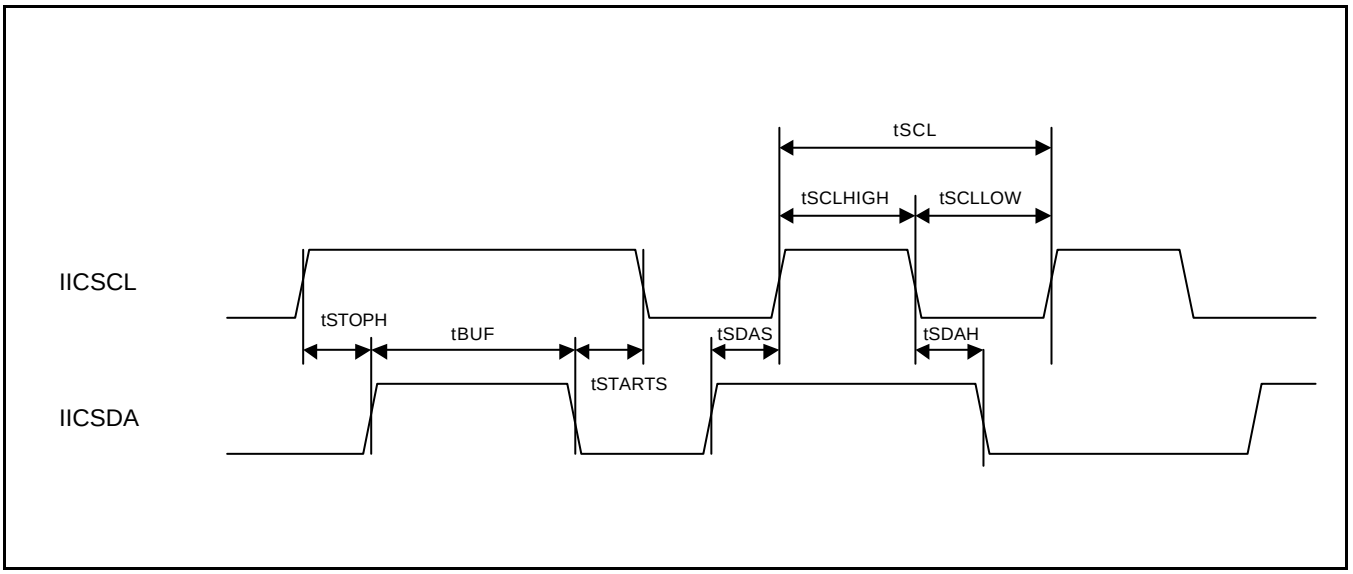


Figure 19-52. IIC Interface Timing

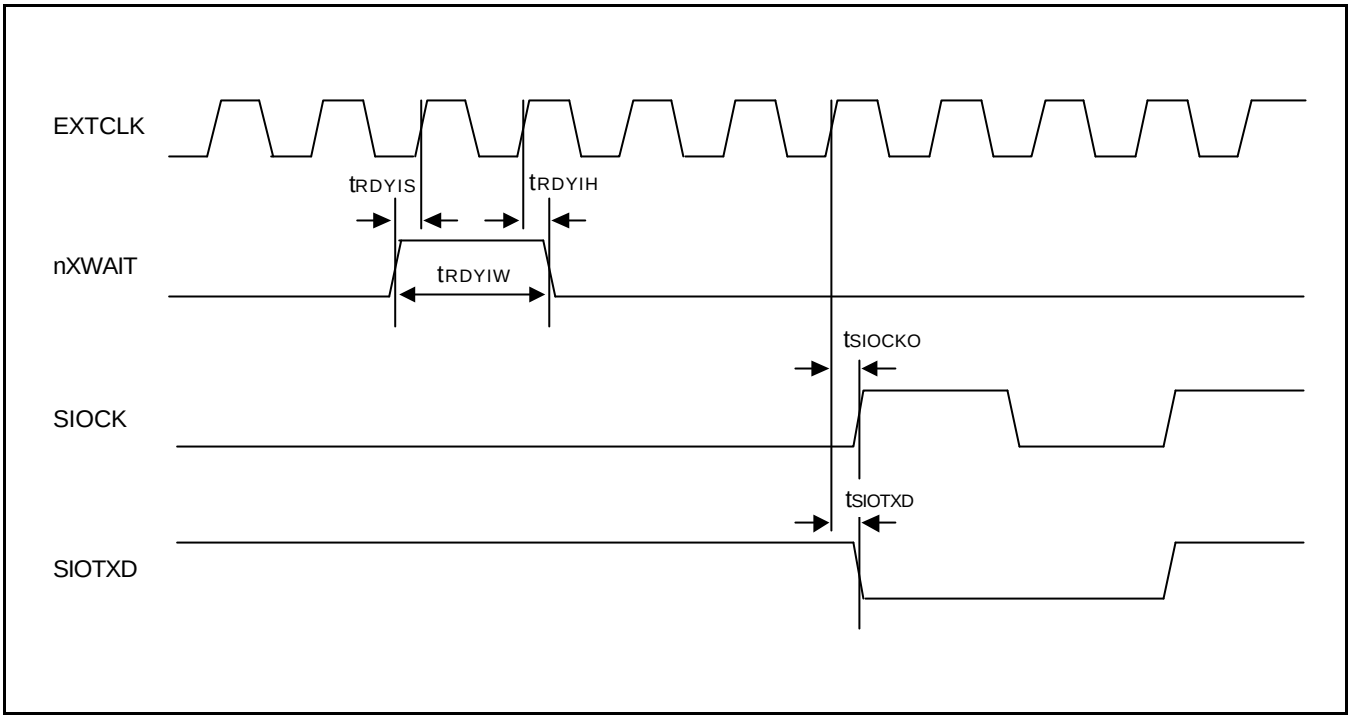


Figure 19-53. SIO Interface Transmit Timing (Rising edge clock)

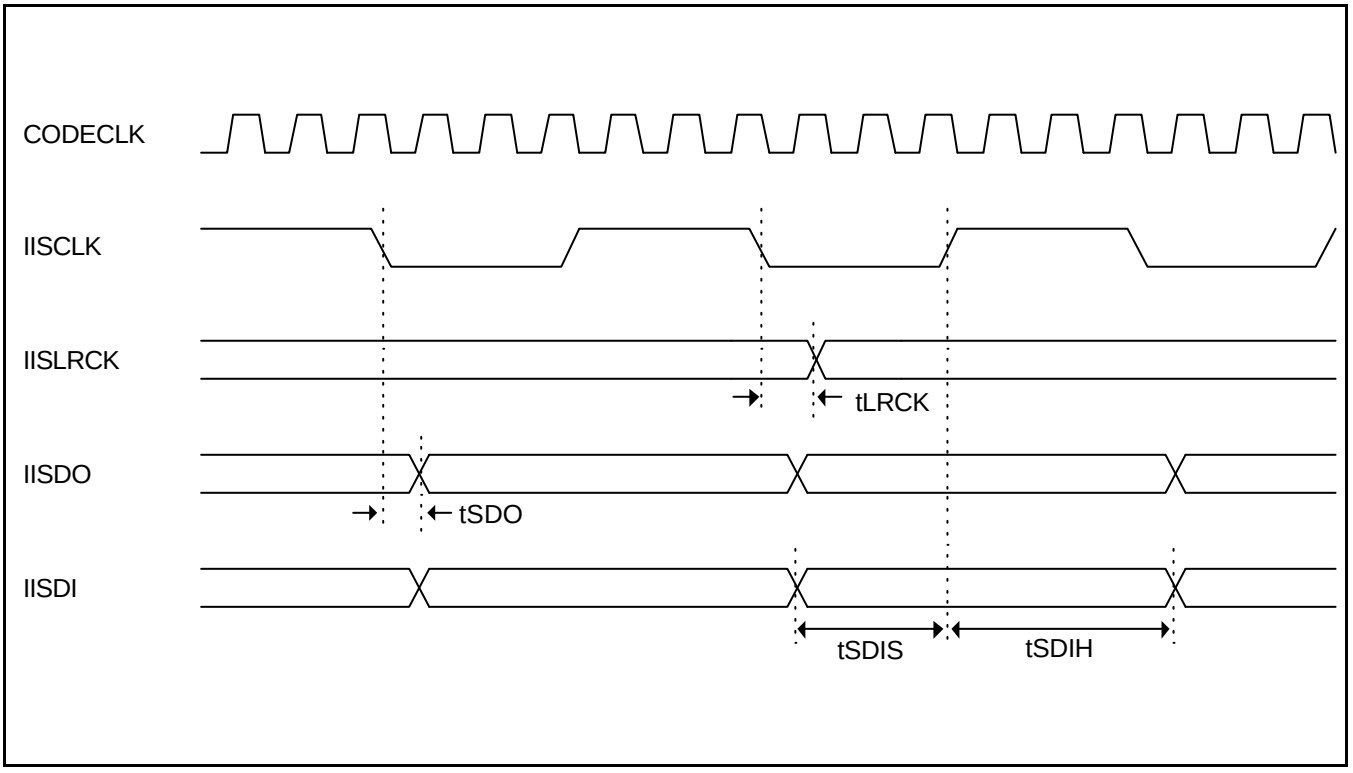


Figure 19-54. SIO Interface Transmit Timing (Rising edge clock)

Table 19-6. Clock Timing Constants

(VDDP: 3.3V, VDDI: 2.5V, Ta = 25 °C, PLCAP = 70pf, max/min = typ ± 30%)

Parameter	Symbol	Min	Typ	Max	Unit
External clock to CKOUT	t_{EX2CK}	–	12	–	ns
External clock to SCLK	$t_{EX2SCLK}$	–	8	–	ns
SCLK to CKOUT	$t_{SCLK2CK}$	–	4	–	ns
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 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	–	–	MCLK
Power-on oscillation setting time	t_{OSC1}	–	4096	–	MCLK
STOP mode return oscillation setting time	t_{OSC2}	–	4096	–	MCLK
the interval before CPU runs after nRESET is released.	$t_{RST2RUN}$	–	132	–	MCLK

Table 19-7. ROM/SRAM Bus Timing Constants

(VDDP: 3.3V, VDDI: 2.5V, Ta = 25°C, PLCAP = 70pf, max/min = typ ± 30%)

Parameter	Symbol	Min	Typ	Max	Unit
ROM/SRAM Address Delay	t_{RAD}	–	12	–	ns
ROM/SRAM Chip select Delay	t_{RCD}	–	11	–	ns
ROM/SRAM Output enable Delay	t_{ROD}	–	11	–	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}	–	13	–	ns
ROM/SRAM Write Byte Enable Delay	t_{RWBED}	–	14	–	ns
ROM/SRAM output Data Delay	t_{RDD}	–	14	–	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}	–	14	–	ns

Table 19-8. Clock Timing Constants

(VDDP: 3.3V, VDDI: 2.5V, Ta = 25°C, PLCAP = 70pf, max/min = typ ± 30%)

Parameter	Symbol	Min	Typ	Max	Unit
DRAM Address Delay	t_{DAD}	–	12	–	ns
DRAM Row active Delay	t_{DRD}	–	11	–	ns
DRAM Read Column active Delay	t_{DRCD}	–	11	–	ns
DRAM Output enable Delay	t_{DOD}	–	12	–	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}	–	14	–	ns
DRAM Cbr Cas active Delay	t_{DCCD}	–	12	–	ns
DRAM Write enable Delay	t_{DWD}	–	13	–	ns
DRAM output Data Delay	t_{DDD}	–	14	–	ns

Table 19-9. Memory Interface Timing Constants

(VDDP :3.3V, VDDI:2.5V, Ta = 25°C, PLCAP = 70pf, max/min = typ ± 30%)

Parameter	Symbol	Min	Typ	Max	Unit
SDRAM Address Delay	t_{SAD}	–	4	–	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}	–	5	–	ns
SDRAM Write enable Delay	t_{SWD}	–	5	–	ns
SDRAM read Data Setup time	t_{SDS}	–	4	–	ns
SDRAM read Data Hold time	t_{SDH}	–	0	–	ns
SDRAM output Data Delay	t_{SDD}	–	8	–	ns
SDRAM Clock Eable Delay	T_{cked}	–	5	–	ns

Table 19-10. External Bus Request Timing Constants

(VDDP: 3.3V, VDDI: 2.5V, Ta = 25°C, PLCAP = 70pf, max/min = typ ± 30%)

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}	–	7	–	ns

Table 19-11. DMA Controller Module Signal Timing Constants

(VDDP: 3.3V, VDDI: 2.5V, Ta = 25°C, PLCAP = 70pf, max/min = typ ± 30%)

Parameter	Symbol	Min	Typ.	Max	Unit
eXternal Request Setup	t_{XRS}	–	3	–	ns
eXternal Acknowledge Setup	t_{XAS}	–	3	–	ns
aCcess to Ack Delay when Low transition	t_{CADL}	–	11	–	ns
aCcess to Ack Delay when High transition	t_{CADH}	–	9	–	ns
eXternal Acknowledge Delay	t_{XAD}	2	–	–	MCLK
Width Acknowledge when Handshake mode	t_{WAH}	0	–	–	MCLK
Width of Acknowledge high when Whole mode	t_{WAW}	2	–	–	MCLK
Width of Acknowledge high when Whole and OTF mode	t_{WAWO}	0	–	–	MCLK
Width Ack of Single	t_{WAS}	3	–	–	MCLK
eXternal Request Hold	t_{XRH}	0	–	–	MCLK
Width of Acknowledge when Demand mode	t_{WAD}	0	–	–	MCLK

Table 19-12. LCD Controller Module Signal Timing Constants

(VDDP: 3.3V, VDDI: 2.5V, Ta = 25°C, PLCAP = 70pf)

Parameter	Symbol	Min	Typ.	Max	Unit
VCLK cycle time	$t_{VCLKCYC}$	4	–	–	MCLK
VCLK to VD delay time	t_{C2DDLY}	–	–	3	ns
VM to VCLK delay time	t_{M2CDLY}	4	–	–	MCLK
VFRAME to VCLK delay time	t_{F2CDLY}	–	–	3	ns

Table 19-13. IIS Controller Module Signal Timing Constants

(VDDP: 3.3V, VDDI: 2.5V, Ta = 25°C, PLCAP = 70pf)

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 19-14. IIC BUS Controller Module Signal Timing

(VDDP: 3.3V, VDDI: 2.5V, Ta = 25°C, PLCAP = 70pf)

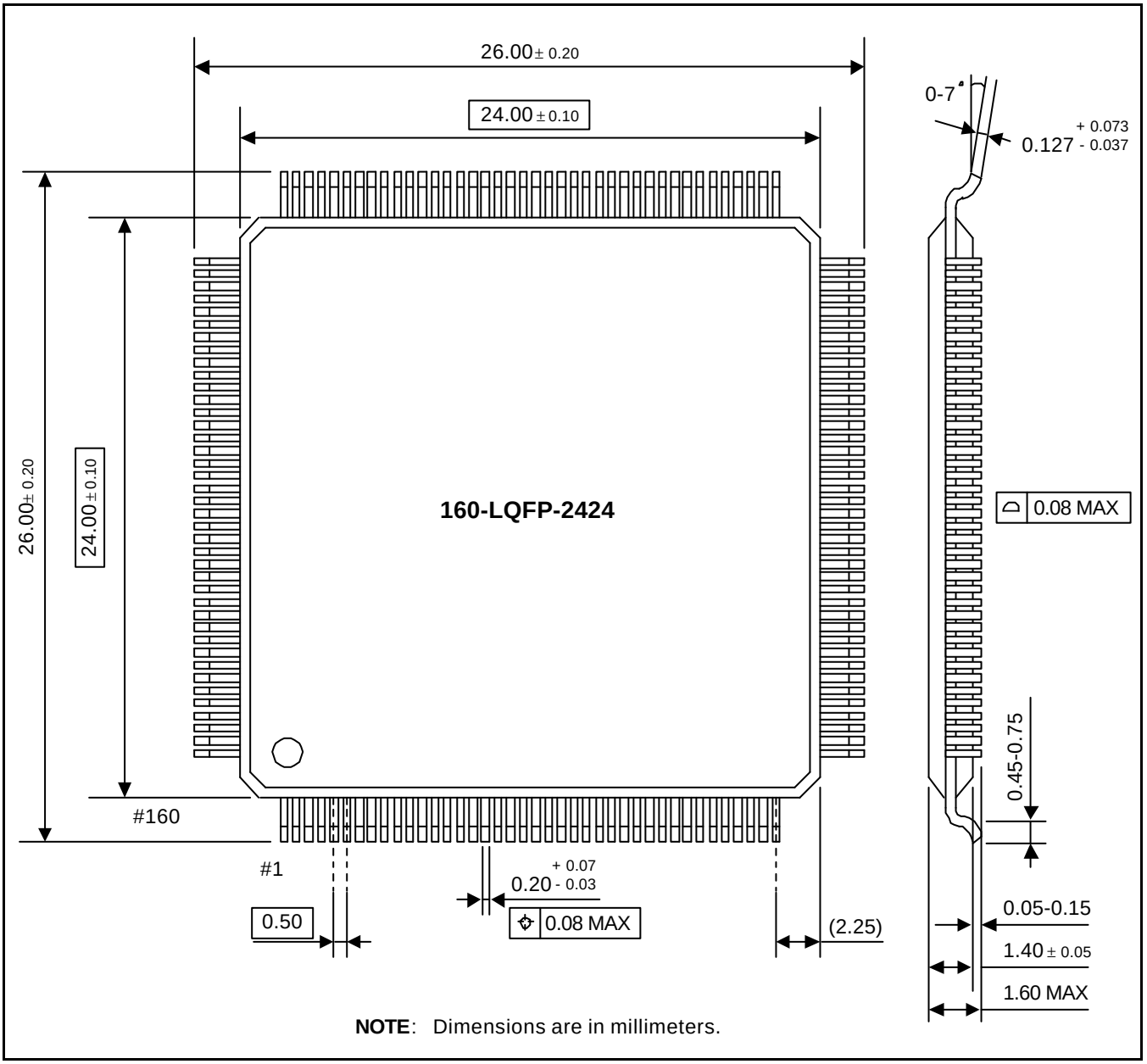
Parameter	Symbol	Min	Typ.	Max	Unit
SCL clock frequency	f_{SCL}	–	–	std. 100 fast 400	KHz
SCL high level pulse width	$t_{SCLHIGH}$	std. 4.0 fast 0.6	–	–	us
SCL low level pulse width	t_{SCLLOW}	std. 4.7 fast 1.3	–	–	us
Bus free time between STOP and START	t_{BUF}	std. 4.7 fast 1.3	–	–	us
START hold time	t_{STARTS}	std. 4.0 fast 0.6	–	–	us
SDA hold time	t_{SDAH}	std. 0 fast 0	–	std. - fast 0.9	us
SDA setup time	t_{SDAS}	std. 250 fast 100	–	–	ns
STOP setup time	T_{stOPH}	std. 4.0 fast 0.6	–	–	us

NOTE: Std. means Standard Mode and fast means Fast Mode.

NOTES

20 MECHANICAL DATA

PACKAGE DIMENSIONS



NOTE: Dimensions are in millimeters.

Figure 20-1. 160-LQFP-2424 Package Dimensions

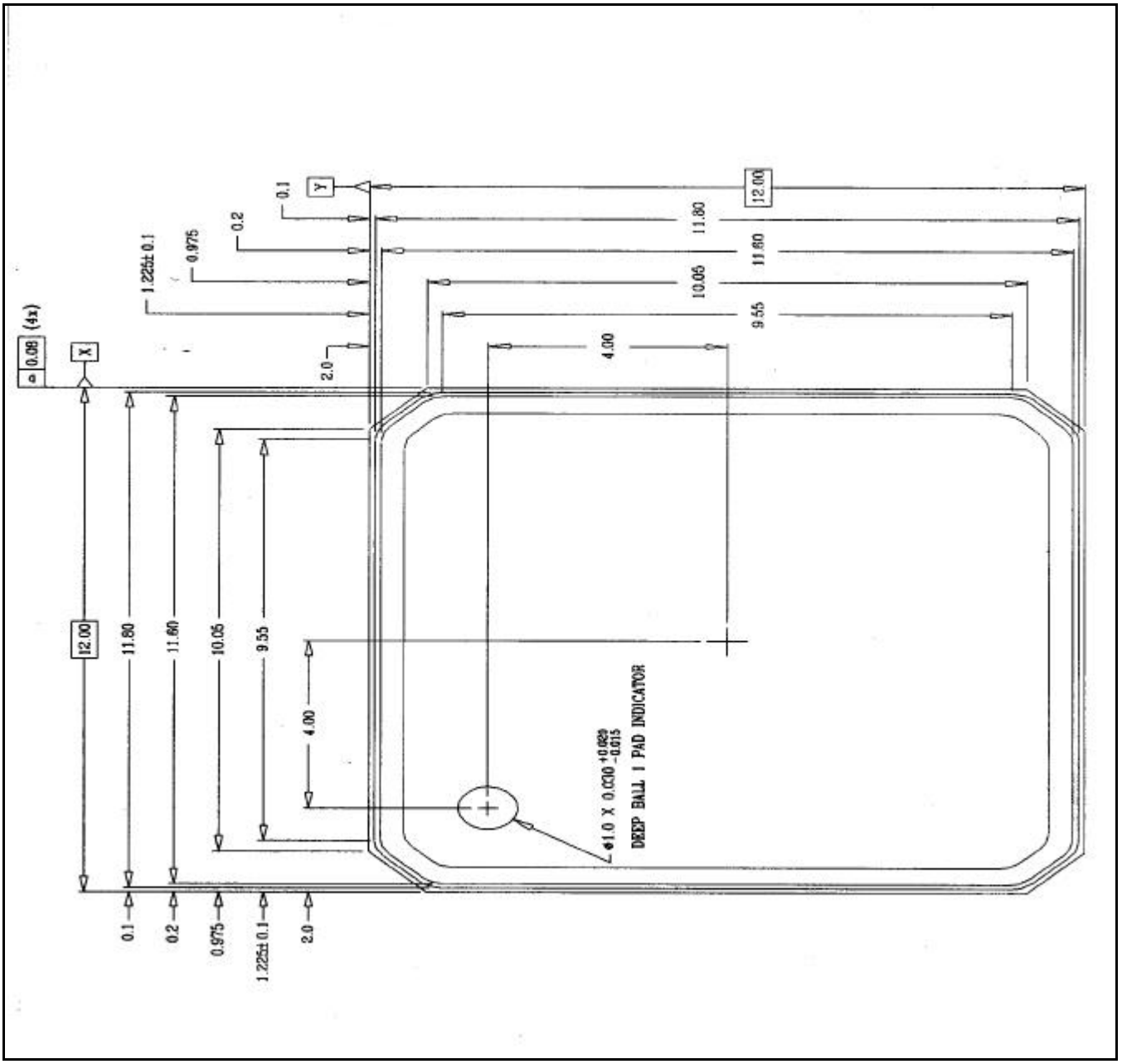


Figure 20-2. 160-FBGA-12.0x12.0 Package Dimensions 1

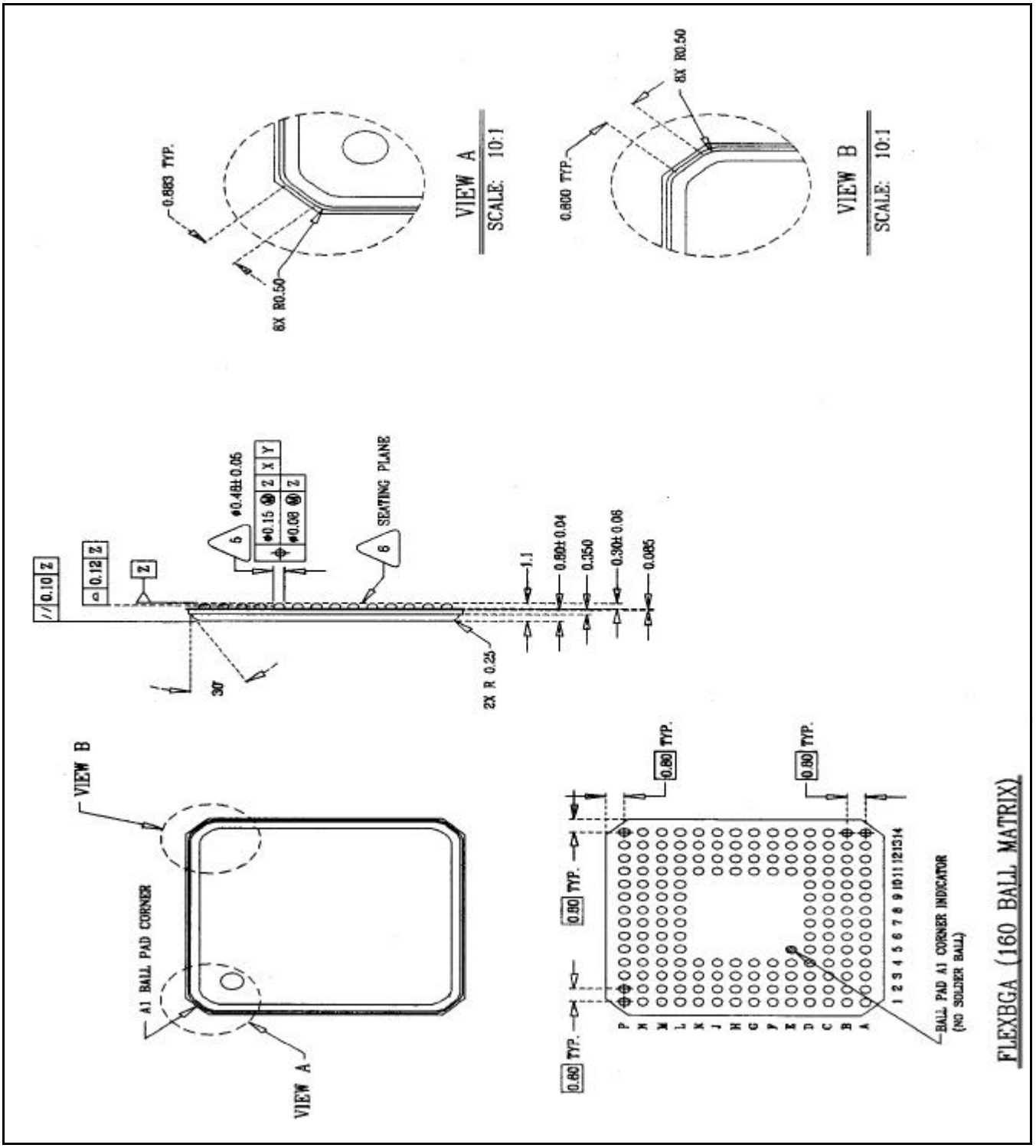


Figure 20-3. 160-FBGA-12.0x12.0 Package Dimensions 2

NOTE: To get more specific information for testing the FBGA/TQFP package using JTAG, Please contact us.

NOTES