

1

PRODUCT OVERVIEW

INTRODUCTION

S5L8700X is an audio player IC supporting various compressed audio format on Flash Memory Media and Hard Disk. S5L8700X provides 256Kbytes of SRAM and it supports optional external SDRAM. An ARM940T™ is provided as a CPU and an 16bit RISC processor (CalmRISC16™) and 24bit MAC(MAC2424™) are provided as a DSP function.

FEATURES

- AMBA BUS Architecture(100Mhz)
- ARM940T (200Mhz)
- CalmADM2E(130Mhz)
CalmRISC16 + MAC2424
with 4KB of Instruction Cache
 - 6KB of X Cache
 - 6KB of Y Cache
- IO DMA (4 ch)
- INTERRUPTS (8ch ext interrupt)
- MEMORIES
 - 256K Byte embedded SRAM
 - 50K Byte embedded ROM
 - SDRAM(16bit, 3.3V), mSDRAM(16bit, 1.8V)
 - DDR SDRAM(16bit, 2.7V), mDDR(16bit, 1.8V)
(support 3 chip select)
 - NOR Flash
(support 3 chip select)
- USB 1.1 Host
- USB2.0 Function
- ATAPI Host Controller
- X-TAL & PLL
 - 32.768kHz X-tal for system clock source.
 - 24MHz(or 48MHz) X-tal for USB2.0 phy
 - Maximum PLL Frequency : 300MHz
- NAND Flash / Smart Media Card
 - (SMC) -MLC type NAND Flash support
- MMC/SD card interface
- Support MMC 4.0 spec, SD 1.0 spec, SDIO
- Memory Stick interface
 - Host controller version 1.3
- LCD INTERFACE
 - 4/8/16 bit parallel & serial interface
 - Color LCD
 - 1/2/4/8bpp Palletized and
16bpp/24-bpp Non-Palletized Color-TFT support
 - 320x240, 640x480 up to 2048x2048
 - Maximum 2K x 2K x 4 virtual screen size
 - OSD(On Screen Display) multiplexing
 - Alpha blending
- TSADC :10bit 4 channel adc
 - (support Touch Screen)
- IIS OUT/ IIS IN
- SPDIF out/ UART / IIC / SPI /
- Chip ID (40bit unique user-id)
- Real Timer Clock (RTC)
- General-Purpose Input/Output (GPIO)
- 16-bit timer(4ch)
- POWER MANAGEMENT
- Digital data recording/playback system utilizing EEPROM and ROM memories as a storage medium – **U.S. Patent No. 5,535,356**
- Digital storage system adopting semiconductor memory device – **U.S. Patent No. 5,623,623**

TYPICAL APPLICATION

- Digital Audio Player

ORDERING INFORMATION

Device	Package	Operating Temperature
S5L8700X	232-FBGA-1010 (0.5)	-20 °C ~ +70 °C

SAMSUNG CONFIDENTIAL

BLOCK DIAGRAM

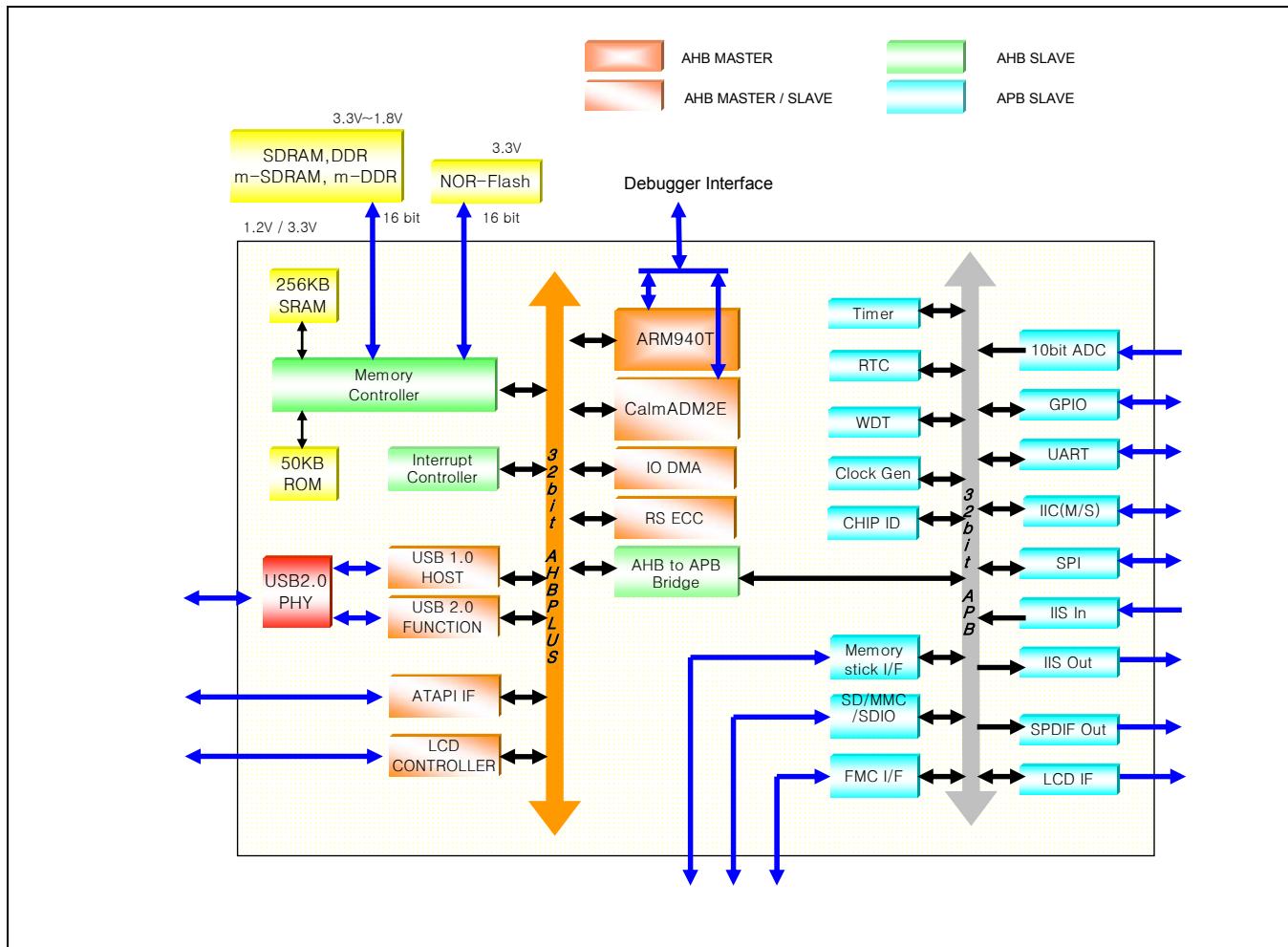


Figure 1-1. S5L8700X Block Diagram

PIN ASSIGNMENTS



Figure 1-2. S5L8700X Pin Assignments (232-FBGA)

PIN DESCRIPTION**Table 1-1. S5L8700X Pin Descriptions**

Pin Name	I/O	Type	Pin Description	Pin	ETC
System					
TEST	I	phic	Test enable	1	
BOOT_MODE	I	phic	Test clock/ Boot bypass mode(active high)	3	
nRESET	I	phic	Global Reset input (active low)	4	
nARM_ADM	I	phic	JTAG select (0:ARM, 1:ADM)	5	
XI	I	phsosck1	Crystal in	22	
XO	O	phsosck1	Crystal out	23	
ARM940T Debugger					
nTRST_DBG	I(up)	phicu	Tap controller reset (active low) for JTAG.	7	
TCK_DBG	I(up)	phicu	JTAG Clock Input	9	
TMS_DBG	I(up)	phicu	Tap controller Machine State control for JTAG	11	
TDI_DBG	I(up)	phicu	Test data input for JTAG	13	
TDO_DBG	O	phob4	Test data output for JTAG	14	
ADM Debugger & External interrupt & Touch Screen					
nTRST_ADM	B	phbct4	P1.3 / ext int3 / TS_PUON	35	
TCK_ADM	B	phbct4	P1.4 / ext int4 / TS_XPON	36	
TMS_ADM	B	phbct4	P1.5 / ext int5 / TS_XMON	37	
TDI_ADM	B	phbct4	P1.6 / ext int6 / TS_YPON	38	
TDO_ADM	B	phbct4	P1.7 / ext int7 / TS_YMON / ExtCLK CLCD	39	
System PLL0, PLL1					
PLL0_VDDA	P	vdd12t_abb	1.2V analog power for PLL0 circuitry	102	
CP0	O	poar50_pll_abb	filter for PLL0 circuitry	103	
PLL0_VSSBBA	P	vssbb_abb	1.2V analog ground and bulk bias for PLL0 circuitry	104	
PLL1_VDDA	P	vdd12t_abb	1.2V analog power for PLL1 circuitry	105	
CP1	O	poar50_pll_abb	filter for PLL1 circuitry	106	
PLL1_VSSBBA	P	vssbb_abb	1.2V analog ground and bulk bias for PLL1 circuitry	107	
PLL_VBBD	P	vbb_abb	1.2V digital bulk bias for PLL0 , PLL1	108	
PLL_VSSD	P	vsst_abb	1.2V digital ground for PLL0 , PLL1(Double bonding)	109	
PLL_VDDD	P	vdd12t_abb	1.2V digital power for PLL0 , PLL1(Double bonding)	110	

Table 1-1. S5L8700X Pin Descriptions (Continued)

Pin Name	I/O	Type	Pin Description	Pin	ETC
NOR Flash & SDRAM					
nSOE	B	pvbct8cd	Nor flash output enable	50	
nSWE	B	pvbct8cd	Nor flash write enable	51	
nSCS0	B	pvbct8cd	Nor flash Chip select for static memory space 0	52	
nSCS1	B	pvbct8cd	Chip select for static memory space 1	53	
nSCS2	B	pvbct8cd	Chip select for static memory space 2	54	
SA0	B	pvbct8cd	Address 22 to 0 for NOR Flash	55	
SA1	B	pvbct8cd	Address 22 to 0 for NOR Flash	56	
SA2	B	pvbct8cd	Address 22 to 0 for NOR Flash	57	
SA3	B	pvbct8cd	Address 22 to 0 for NOR Flash	58	
SA4	B	pvbct8cd	Address 22 to 0 for NOR Flash	59	
SA5	B	pvbct8cd	Address 22 to 0 for NOR Flash	60	
SA6	B	pvbct8cd	Address 22 to 0 for NOR Flash	61	
SA7	B	pvbct8cd	Address 22 to 0 for NOR Flash	62	
SA8	B	pvbct8cd	Address 22 to 0 for NOR Flash	63	
SA9	B	pvbct8cd	Address 22 to 0 for NOR Flash	64	
SA10	B	pvbct8cd	Address 22 to 0 for NOR Flash	65	
SA11	B	pvbct8cd	Address 22 to 0 for NOR Flash	66	
SA12	B	pvbct8cd	Address 22 to 0 for NOR Flash	67	
SA13	B	pvbct8cd	Address 22 to 0 for NOR Flash	68	
SA14	B	pvbct8cd	Address 22 to 0 for NOR Flash	69	
SA15	B	pvbct8cd	Address 22 to 0 for NOR Flash	70	
SA16	B	pvbct8cd	Address 22 to 0 for NOR Flash	71	
SA17	B	pvbct8cd	Address 22 to 0 for NOR Flash	72	
SA18	B	pvbct8cd	Address 22 to 0 for NOR Flash	73	
SA19	B	pvbct8cd	Address 22 to 0 for NOR Flash	74	
SA20	B	pvbct8cd	Address 22 to 0 for NOR Flash	75	
SA21	B	pvbct8cd	Address 22 to 0 for NOR Flash	76	
SA22	B	pvbct8cd	Address 22 to 0 for NOR Flash	77	
SDB0	B	pvbct8cd	Data 15 to 0 for NOR flash	82	
SDB1	B	pvbct8cd	Data 15 to 0 for NOR flash	83	
SDB2	B	pvbct8cd	Data 15 to 0 for NOR flash	84	
SDB3	B	pvbct8cd	Data 15 to 0 for NOR flash	85	
SDB4	B	pvbct8cd	Data 15 to 0 for NOR flash	86	
SDB5	B	pvbct8cd	Data 15 to 0 for NOR flash	87	
SDB6	B	pvbct8cd	Data 15 to 0 for NOR flash	88	

Table 1-1. S5L8700X Pin Descriptions (Continued)

Pin Name	I/O	Type	Pin Description	Pin	ETC
SDB7	B	pvbct8cd	Data 15 to 0 for NOR flash	89	
SDB8	B	pvbct8cd	Data 15 to 0 for NOR flash	90	
SDB9	B	pvbct8cd	Data 15 to 0 for NOR flash	91	
SDB10	B	pvbct8cd	Data 15 to 0 for NOR flash	92	
SDB11	B	pvbct8cd	Data 15 to 0 for NOR flash	93	
SDB12	B	pvbct8cd	Data 15 to 0 for NOR flash	94	
SDB13	B	pvbct8cd	Data 15 to 0 for NOR flash	95	
SDB14	B	pvbct8cd	Data 15 to 0 for NOR flash	96	
SDB15	B	pvbct8cd	Data 15 to 0 for NOR flash	97	
DCLK2	B	psbsstl_8700	SDRAM clock	136	
VREF	I	phia_8700	DDR SDRAM VREF	146	
DQS0	B	psbsstl_8700	DDR SDRAM LDQS	142	
DQS1	B	psbsstl_8700	DDR SDRAM UDQS	143	
DCLKN	O	psosstldiff_8700	DDR SDRAM CLKB	145	
DCLK	O	psosstldiff_8700	DDR SDRAM CLK	144	
DCKE	B	psbsstl_8700	SDRAM clock enable	147	
DQM0	B	psbsstl_8700	SDRAM Data mask [7:0]	156	
DQM1	B	psbsstl_8700	SDRAM Data mask [15:8]	155	
nDCS0	B	psbsstl_8700	SDRAM Chip select0	137	
nDCS1	B	psbsstl_8700	SDRAM Chip select1	138	
nDCS2	B	psbsstl_8700	SDRAM Chip select2	139	
nDWE	B	psbsstl_8700	SDRAM write enable	150	
nRAS	B	psbsstl_8700	SDRAM raw address strobe	148	
nCAS	B	psbsstl_8700	SDRAM column address strobe	149	
MDA0	B	psbsstl_8700	Address 12 to 0 for SDRAM	132	
MDA1	B	psbsstl_8700	Address 12 to 0 for SDRAM	133	
MDA2	B	psbsstl_8700	Address 12 to 0 for SDRAM	134	
MDA3	B	psbsstl_8700	Address 12 to 0 for SDRAM	135	
MDA4	B	psbsstl_8700	Address 12 to 0 for SDRAM	119	
MDA5	B	psbsstl_8700	Address 12 to 0 for SDRAM	118	
MDA6	B	psbsstl_8700	Address 12 to 0 for SDRAM	123	
MDA7	B	psbsstl_8700	Address 12 to 0 for SDRAM	121	
MDA8	B	psbsstl_8700	Address 12 to 0 for SDRAM	120	
MDA9	B	psbsstl_8700	Address 12 to 0 for SDRAM	117	
MDA10	B	psbsstl_8700	Address 12 to 0 for SDRAM	131	
MDA11	B	psbsstl_8700	Address 12 to 0 for SDRAM	122	
MDA12	B	psbsstl_8700	Address 12 to 0 for SDRAM	124	

Table 1-1. S5L8700X Pin Descriptions (Continued)

Pin Name	I/O	Type	Pin Description	Pin	ETC
MDBA0	B	psbsstl_8700	Bank 0 address for SDRAM	129	
MDBA1	B	psbsstl_8700	Bank 1 address for SDRAM	130	
MDB0	B	psbsstl_8700	Data 15 to 0 for SDRAM	174	
MDB1	B	psbsstl_8700	Data 15 to 0 for SDRAM	173	
MDB2	B	psbsstl_8700	Data 15 to 0 for SDRAM	172	
MDB3	B	psbsstl_8700	Data 15 to 0 for SDRAM	171	
MDB4	B	psbsstl_8700	Data 15 to 0 for SDRAM	170	
MDB5	B	psbsstl_8700	Data 15 to 0 for SDRAM	169	
MDB6	B	psbsstl_8700	Data 15 to 0 for SDRAM	168	
MDB7	B	psbsstl_8700	Data 15 to 0 for SDRAM	167	
MDB8	B	psbsstl_8700	Data 15 to 0 for SDRAM	164	
MDB9	B	psbsstl_8700	Data 15 to 0 for SDRAM	163	
MDB10	B	psbsstl_8700	Data 15 to 0 for SDRAM	162	
MDB11	B	psbsstl_8700	Data 15 to 0 for SDRAM	161	
MDB12	B	psbsstl_8700	Data 15 to 0 for SDRAM	160	
MDB13	B	psbsstl_8700	Data 15 to 0 for SDRAM	159	
MDB14	B	psbsstl_8700	Data 15 to 0 for SDRAM	158	
MDB15	B	psbsstl_8700	Data 15 to 0 for SDRAM	157	
USB					
VDDA33T1	P	vdd33th_abb	Transceiver supply	191	
VSSA33T2	P	vssbbh_abb	Transceiver supply	192	
DP	B	phtoa_abb	Data pin (special pad , single-in-line)	193	
VSSA33T3	P	vssbbh_abb	Transceiver supply	194	
DM	B	phtoa_abb	Data pin (special pad , single-in-line)	195	
VSSA33T4	P	vssbbh_abb	Transceiver supply	196	
VDDA33T5	P	vdd33th_abb	Transceiver supply	197	
REXT	O	phoarext_abb	analog signal (special pad , single-in-line)	198	
VDDA33C6	P	vdd33th_abb	Common supply	199	
VSSA33C7	P	vssbbh_abb	Common supply	200	
USBXO	I	phoa_abb	X-tal out	201	
USBXI	I	phoa_abb	X-tal in	202	
ANALOGTEST	O	phoa_abb	Analog Test in/out	203	
VBUS	I	phtic	USB2.0 Power detect / ext int 8	45	

Table 1-1. S5L8700X Pin Descriptions (Continued)

Pin Name	I/O	Type	Pin Description	Pin	ETC
FMC/ MMC/ MS/ ATAPI/LCD If					
HD0	B	pvbct8cd	P2.0/ATAPI HD0/LCDif D0/MSD/SMC IO0	175	
HD1	B	pvbct8cd	P2.1/ATAPI HD1/LCDif D1/ - /SMC IO1	176	
HD2	B	pvbct8cd	P2.2/ATAPI HD2/LCDif D2/ - /SMC IO2	177	
HD3	B	pvbct8cd	P2.3/ATAPI HD3/LCDif D3/ - /SMC IO3	178	
HD4	B	pvbct8cd	P2.4/ATAPI HD4/LCDif D4/ - /SMC IO4	179	
HD5	B	pvbct8cd	P2.5/ATAPI HD5/LCDif D5/ - /SMC IO5	180	
HD6	B	pvbct8cd	P2.6/ATAPI HD6/LCDif D6/ - /SMC IO6	181	
HD7	B	pvbct8cd	P2.7/ATAPI HD7/LCDif D7/ - /SMC IO7	182	
HD8	B	pvbct8cd	P3.0/ATAPI HD8/LCDif D8/SDC D0/SMC IO8	187	
HD9	B	pvbct8cd	P3.1/ATAPI HD9/LCDif D9/ - /SMC IO9	188	
HD10	B	pvbct8cd	P3.2/ATAPI HD10/LCDif D10/SDC D2/SMC IO10	204	
HD11	B	pvbct8cd	P3.3/ATAPI HD11/LCDif D11/SDC D3/SMC IO11	205	
HD12	B	pvbct8cd	P3.4/ATAPI HD12/LCDif D12/ - /SMC IO12	206	
HD13	B	pvbct8cd	P3.5/ATAPI HD13/LCDif D13/SDC WP/SMC IO13	207	
HD14	B	pvbct8cd	P3.6/ATAPI HD14/LCDif D14/ - /SMC IO14	208	
HD15	B	pvbct8cd	P3.7/ATAPI HD15/LCDif D15/ - /SMC IO15	209	
CBLID	B	pvbct8cd	P4.0/ATAPI CBLID/ - / - / -	210	
DMACK	B	pvbct8cd	P4.1/ATAPI DMACK/LCDif REG/ - / SMC CLE	211	
DMARQ	B	pvbct8cd	P4.2/ATAPI DMARQ/ - / - / -	212	
IORDY	B	pvbct8cd	P4.3/ATAPI IORDY/ - / - / -	213	
DIOW	B	pvbct8cd	P4.4/ATAPI DIOW/LCDif nWR/MS BS/SMC nWR	214	
DIOR	B	pvbct8cd	P4.5/ATAPI DIOR/LCDif nRD/ - /SMC nRD	215	
ATA_RESET	B	pvbct8cd	P4.6/ATAPI RESET/ - / - /SMC nWP	216	
ATA_INTRQ	B	pvbct8cd	P4.7/ATAPI INTRQ/ - / - / -	217	
DA0	B	pvbct8cd	P5.0/ATAPI DA0/ - / - / -	218	
DA1	B	pvbct8cd	P5.1/ATAPI DA1/ - / - / -	219	
DA2	B	pvbct8cd	P5.2/ATAPI DA2/ - / - / -	220	
CS0	B	pvbct8cd	P5.3/ATAPI CS0/ - / - / -	221	
CS1	B	pvbct8cd	P5.4/ATAPI CS1/ - / - / -	222	
VDEN	B	phbct4	P5.5/VDEN	47	
VSYND	B	phbct4	P5.6/VSYNC / SDC CMD	48	
HSYNC	B	phbct4	P5.7/HSYNC / SDC D1	49	

Table 1-1. S5L8700X Pin Descriptions (Continued)

Pin Name	I/O	Type	Pin Description	Pin	ETC
nSMRDY	B	pvbct8cd	P6.0/SMC Ready & Busy state input/SMBUSY	225	
nSMCS0	B	pvbct8cd	P6.1/nNFCS0	226	
nSMCS1	B	pvbct8cd	P6.2/nNFCS1	227	
nSMCS2	B	pvbct8cd	P6.3/nNFCS2 / TxD1 (Uart1)	228	
nSMCS3	B	pvbct8cd	P6.4/nNFCS3 / RxD1 (Uart1)	229	
SDMMCLK	B	pvbct8cd	P6.5/SDCLK and MMCLK	230	
MSCLK	B	pvbct8cd	P6.6/MS clock	231	
ALE	B	pvbct8cd	P6.7/SMC ALE	232	
LCDnRST	B	phbct4	P7.0/LCDif nRST/ VCLK	2	
LCDnCS	B	phbct4	P7.1/LCDif nCS	6	
Ext interrupt / GPIO					
P1_0	B	phbct4	P1.0 / ext int 0	32	
P1_1	B	phbct4	P1.1 / ext int 1	33	
P1_2	B	phbct4	P1.2 / ext int 2	34	
IIS					
IISMCK	O(t)	phbct4	SAIU master clock(WM8731L20 XT/MCLK) / P7.2	8	
IISWS	O(t)	phbct4	SAIU word select(WM8731L20 DACLRC) / P7.3	10	
IISBCLK	O(t)	phbct4	SAIU bit clock(WM8731L20 BCLK) / P7.4	12	
IISD_OUT	O(t)	phbct4	SAIU data out(WM8731L20 DACDAT) / P7.5	16	
IISD_IN	B	phbct4	SAIU data in(WM8732L20 ADCDATA) / P7.6	18	
SPDIF OUT					
SPDIFOUT	B	phbct4	SPDIF out / P7.7	20	
IIC					
IICCLK	B(OD)	phtbcd4	IIC clock / WM8731L20 SCLK / P10.0	41	
IICDAT	B(OD)	phtbcd4	IIC data / WM8731L20 SDIN / P10.1	43	

Table 1-1. S5L8700X Pin Descriptions (Continued)

Pin Name	I/O	Type	Pin Description	Pin	ETC
TIMER & SPI					
P0_0	B	phbct4	P0.0 / TAOUT / nSSI	24	
P0_1	B	phbct4	P0.1 / TBOUT / SPICLK	25	
P0_2	B	phbct4	P0.2 / TCOUT / MOSI	26	
P0_3	B	phbct4	P0.3 / TDOUT / MISO	27	
P0_4	B	phbct4	P0.4 / TEXTCLK / pll0 out for test	28	
P0_5	B	phbct4	P0.5 / TCAP / pll1 out for test	29	
UART					
P0_6	B	phbct4	P0.6 / UART0 transmit pin / IISD1_OUT	30	
P0_7	B	phbct4	P0.7 / UART0 receive pin / IISD2_OUT	31	
ADC					
ADC_VSSA	P	vssbbh_abb	ADC agnd,avss33a1,avbb33a1,avss33a2	116	
ADC_VCC	P	vdd33th_abb	ADC VREF,AVDD33A1,AVDD33A2 연결	115	
ADC0	I	phiar50_abb	10-bit ADC input channel 0	114	
ADC1	I	phiar50_abb	10-bit ADC input channel 1	113	
ADC2	I	phiar50_abb	10-bit ADC input channel 2	112	
ADC3	I	phiar50_abb	10-bit ADC input channel 3	111	
Digital Power & GND					
Digital Power	P	vdd12ih	15, 40, 78, 98, 125, 151, 183	-	
Digital GND	P	vssiph	16, 41, 79, 99, 126, 152, 184	-	
USB 2.0 PHY	P	vdd12ih	190	-	
USB 2.0 PHY	P	vssiph	189	-	
PAD Power * GND					
PAD Power 1	P	vdd33oph	19, 44, 185, 223	-	
PAD GND 1	P	vssoh	20, 45, 186, 224	-	
PAD Power 2	P	vdd33oph	80, 100 (For NOR I/O) * 2	-	
PAD GND 2	P	vssoh	81, 101 (For NOR I/O) * 2	-	
PAD Power 3	P	vdd33oph	127, 140, 153, 165 (For SDRAM I/O) * 4	-	
PAD GND 3	P	vssoh	128, 141, 154, 166 (For SDRAM I/O) * 4	-	

I/O BALL MAP

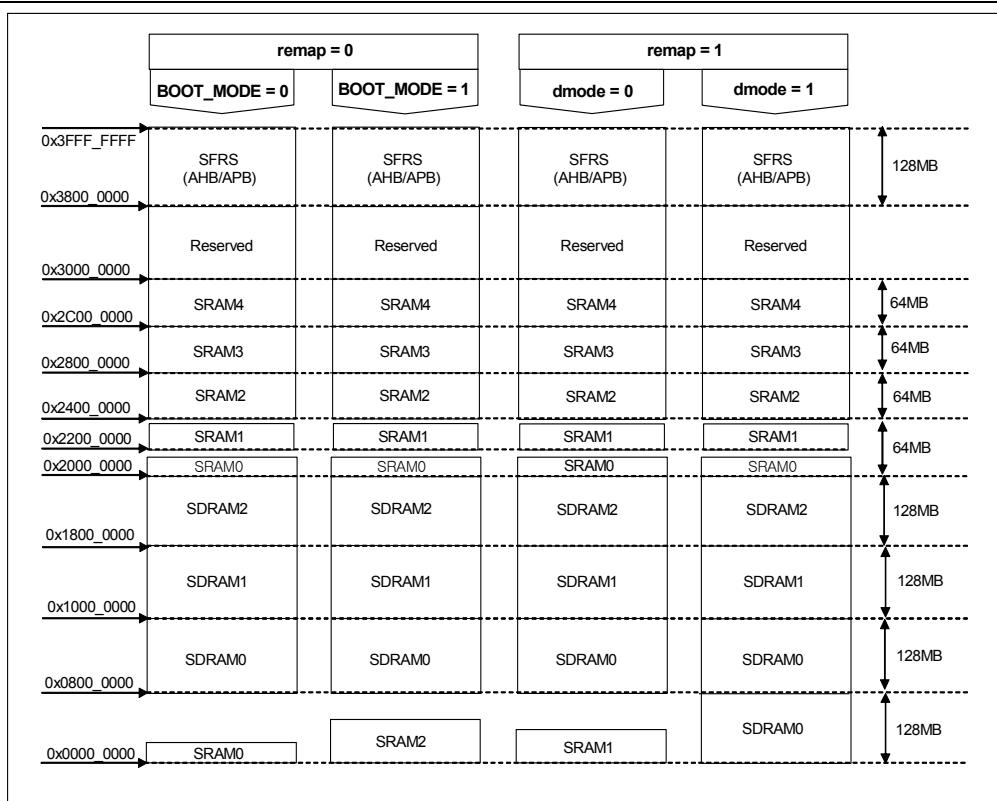
Table 1-2. S5L8700X I/O BALL MAP

Ball map	core pin number	Pin Assignment	Ball map	core pin number	Pin Assignment	Ball map	core pin number	Pin Assignment
B1	1	TEST	R2	52	nSCS0	R12	103	CP0
C2	2	LCDnRST	T1	53	nSCS1	U13	104	PLL0_VSSBBA
E4	3	BOOT_MODE	P3	54	nSCS2	V14	105	PLL1_VDDA
D3	4	nRESET	P4	55	SDA0	T13	106	CP1
E3	5	nARM_ADM	R3	56	SDA1	R13	107	PLL1_VSSBBA
C1	6	LCDnCS	T2	57	SDA2	U14	108	PLL_VBBD
D2	7	nTRST_DBG	U1	58	SDA3	V15	109	PLL_VSSD
D1	8	IISMCK	V2	59	SDA4	T14	110	PLL_VDDD
E2	9	TCK_DBG	U3	60	SDA5	R14	111	ADC3
F4	10	IISWS	T4	61	SDA6	V16	112	ADC2
E1	11	TMS_DBG	R5	62	SDA7	U15	113	ADC1
G5	12	IISBCLK	R6	63	SDA8	T15	114	ADC0
F3	13	TDI_DBG	V3	64	SDA9	U16	115	ADC_VCO
F2	14	TDO_DBG	P7	65	SDA10	V17	116	ADC_VSSA
F1	15	VDDIO	U4	66	SDA11	U18	117	MDA9
G4	16	IISD_OUT	T5	67	SDA12	T17	118	MDA5
H5	17	VSSI0	V4	68	SDA13	R16	119	MDA4
G3	18	IISD_IN	R7	69	SDA14	P15	120	MDA8
G1	19	VDDE0	U5	70	SDA15	T18	121	MDA7
G2	20	SPDIFOut	V5	71	SDA16	P16	122	MDA11
H4	21	VSSE0	T6	72	SDA17	R17	123	MDA6
H2	22	XI	P8	73	SDA18	R18	124	MDA12
H1	23	XO	U6	74	SDA19	M14	125	VDDI4
H3	24	P0_0	V6	75	SDA20	N15	126	VSSI4
J5	25	P0_1	T7	76	SDA21	M15	127	VDDE4
J4	26	P0_2	P9	77	SDA22	P18	128	VSSE4
J1	27	P0_3	U7	78	VDDI2	P17	129	MDAB0
J2	28	P0_4	V7	79	VSSI2	N16	130	MDAB1
J3	29	P0_5	R8	80	VDDE2	L14	131	MDA10
K1	30	P0_6	T8	81	VSSE2	N18	132	MDA0
K5	31	P0_7	U8	82	SDB0	N17	133	MDA1
K4	32	P1_0	V8	83	SDB1	M16	134	MDA2
K2	33	P1_1	R9	84	SDB2	M17	135	MDA3
L1	34	P1_2	T9	85	SDB3	M18	136	DCLK2
K3	35	P1_3	V9	86	SDB4	L15	137	nDCS0
L5	36	P1_4	U9	87	SDB5	L16	138	nDCS1
L2	37	P1_5	P10	88	SDB6	L18	139	nDCS2
M1	38	P1_6	R10	89	SDB7	L17	140	VDDE8
L4	39	P1_7	V10	90	SDB8	K14	141	VSSE8
L3	40	VDDI1	T10	91	SDB9	K18	142	DQS0
N1	41	IICCLK	U10	92	SDB10	K17	143	DQS1
M2	42	VSSI1	P11	93	SDB11	K16	144	DCLK
M3	43	IICDAT	V11	94	SDB12	J16	145	DCLKN
N2	44	VDDE1	U11	95	SDB13	J18	146	VREF
P1	45	VBUS	T11	96	SDB14	K15	147	DCKE
M4	46	VSSE1	R11	97	SDB15	J15	148	nRAS
M5	47	VDEN	V12	98	VDDI3	H18	149	nCAS
P2	48	VSYNC	P12	99	VSSI3	J14	150	nDWE
R1	49	HSYNC	U12	100	VDDE3	J17	151	VDDI5
N3	50	nSOE	V13	101	VSSE3	H15	152	VSSI5
N4	51	nSWE	T12	102	PLL0_VDDA	G18	153	VDDE5

Table 1-2. S5L8700X I/O BALL MAP (Continued)

Ball map	core pin number	Pin Assignment	Ball map	core pin number	Pin Assignment	Ball map	core pin number	Pin Assignment
H14	154	VSSE5	A15	181	HD6	B8	208	HD14
H17	155	DQM1	D14	182	HD7	C9	209	HD15
H16	156	DQM0	D13	183	VDDI6	A8	210	CBLID
F18	157	MDB15	E12	184	VSSI6	D8	211	DMACK
G15	158	MDB14	A14	185	VDDE6	C8	212	DMARQ
G17	159	MDB13	B14	186	VSSE6	B7	213	IORDY
G14	160	MDB12	C13	187	HD8	A7	214	DIOW
E18	161	MDB11	B13	188	HD9	E8	215	DIOR
G16	162	MDB10	A13	189	VSSI7	D7	216	ATA_RESET
F17	163	MDB9	D12	190	VDDI7	A6	217	ATA_INTRQ
D18	164	MDB8	E11	191	VDDA33T1	B6	218	DA0
F15	165	VDDE9	C12	192	VSSA33T2	C7	219	DA1
F16	166	VSSE9	A12	193	DP	E7	220	DA2
E17	167	MDB7	B12	194	VSSA33T3	A5	221	CS0
C18	168	MDB6	A11	195	DM	B5	222	CS1
E15	169	MDB5	B11	196	VSSA33T4	D6	223	VDDE7
E16	170	MDB4	D11	197	VDDA33T5	C6	224	VSSE7
D17	171	MDB3	C11	198	REXT	A4	225	nSMRDY
D16	172	MDB2	B10	199	VDDA33C6	B4	226	nSMCS0
C17	173	MDB1	D10	200	VSSA33C7	D5	227	nSMCS1
B18	174	MDB0	E10	201	USBX0	C5	228	nSMCS2
A17	175	HD0	A10	202	USBXI	A3	229	nSMCS3
A16	176	HD1	B9	203	ANALOGTEST	C4	230	SDMMCLK
B16	177	HD2	C10	204	HD10	B3	231	MSCLK
B15	178	HD3	E9	205	HD11	A2	232	ALE
C15	179	HD4	A9	206	HD12			
C14	180	HD5	D9	207	HD13			

MEMORY MAP



NOTES : 1.
SRAM0 : ROM(50KB)
SRAM1 : Internal SRAM(256KB)
SRAM2 : NOR-FLASH(8Mbit)
SRAM3,4: External SRAM type memory

2. In case of **remap=0**,
boot_mode=0 : boot from ROM
boot_mode=1 : boot from NOR FLASH

Figure 1-3. Overall Physical Address Configuration

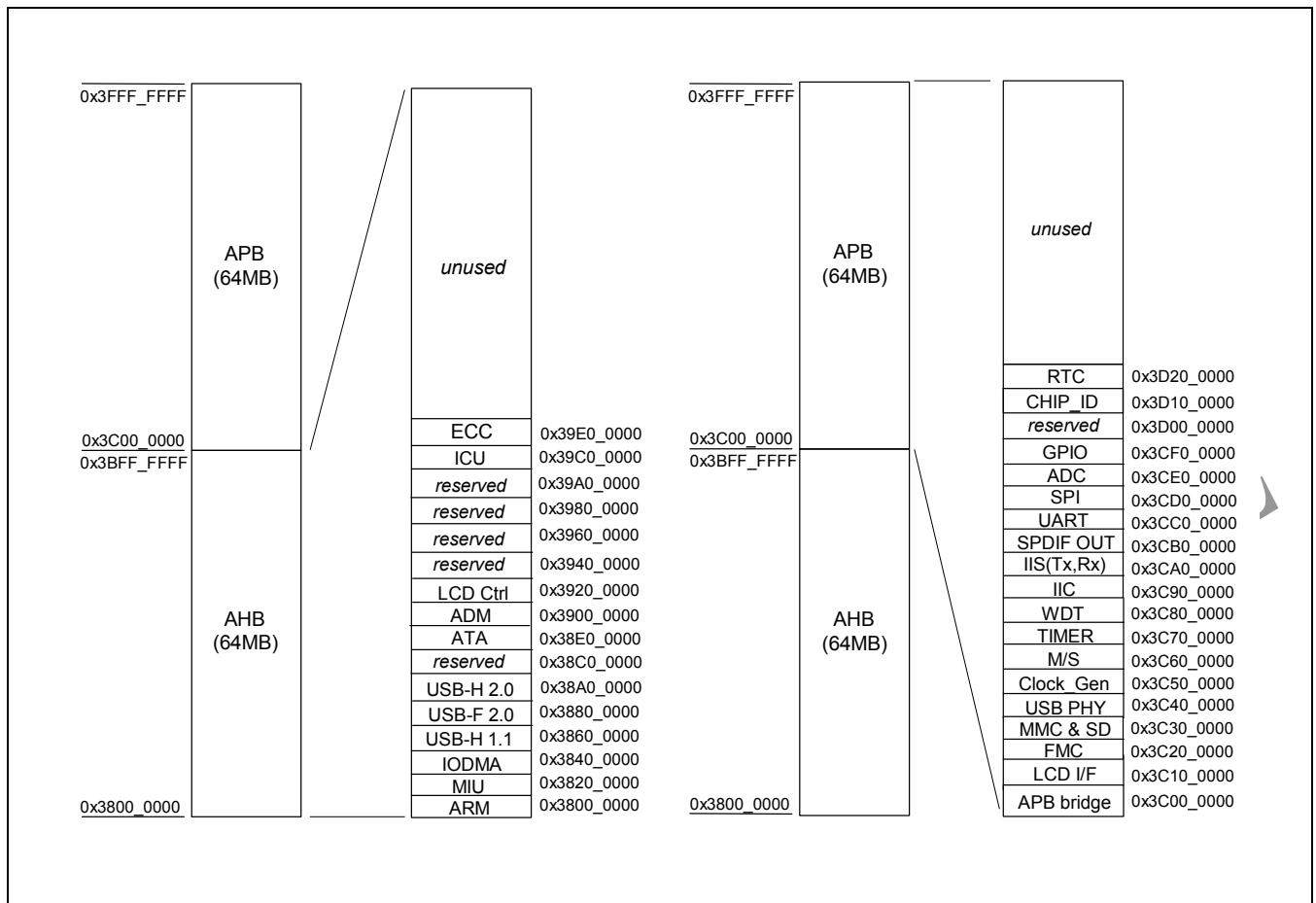


Figure 1-4. 32bit Address Mapped I/O Configuration

NOTES

SAMSUNG CONFIDENTIAL

2 PROGRAMMER'S MODEL

OVERVIEW

S5L8700X was developed using the advanced ARM9TDMI core designed by advanced RISC machines, Ltd.

- Processor Operating States

From the programmer's point of view, the ARM9TDMI can be in one of two states:

- ARM state which executes 32-bit, word-aligned ARM instructions.
- THUMB state which operates with 16-bit, half-word-aligned THUMB instructions. In this state, the PC uses bit 1 to select between alternate half-words.

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:

1. On execution of the BX instruction with the state bit clear in the operand register.
2. 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

ARM9TDMI 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. ARM9TDMI 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.

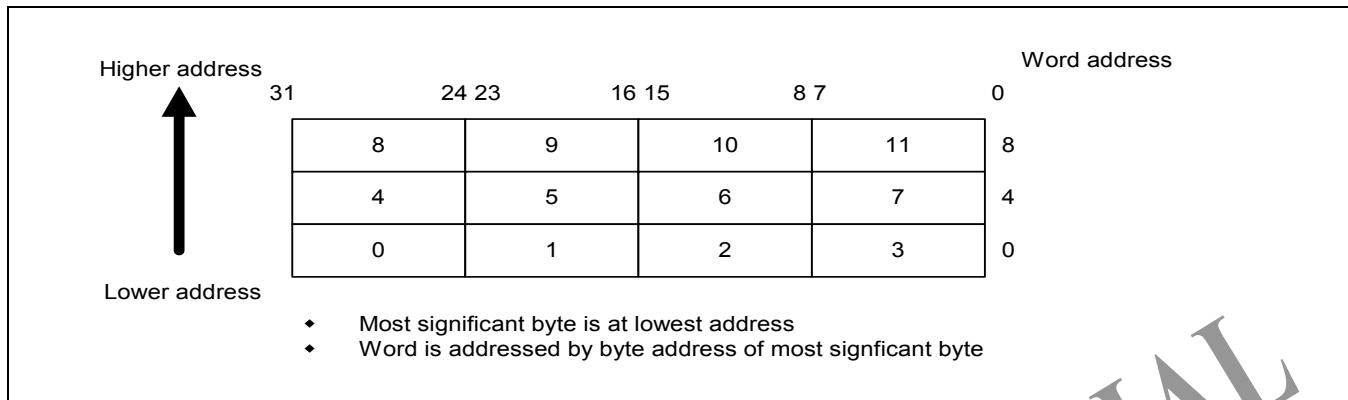


Figure 2-1. Big-Endian Addresses of Bytes within Words

NOTE

The data locations in the external memory are different with Figure 2-1 in the S5L8700X.

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.

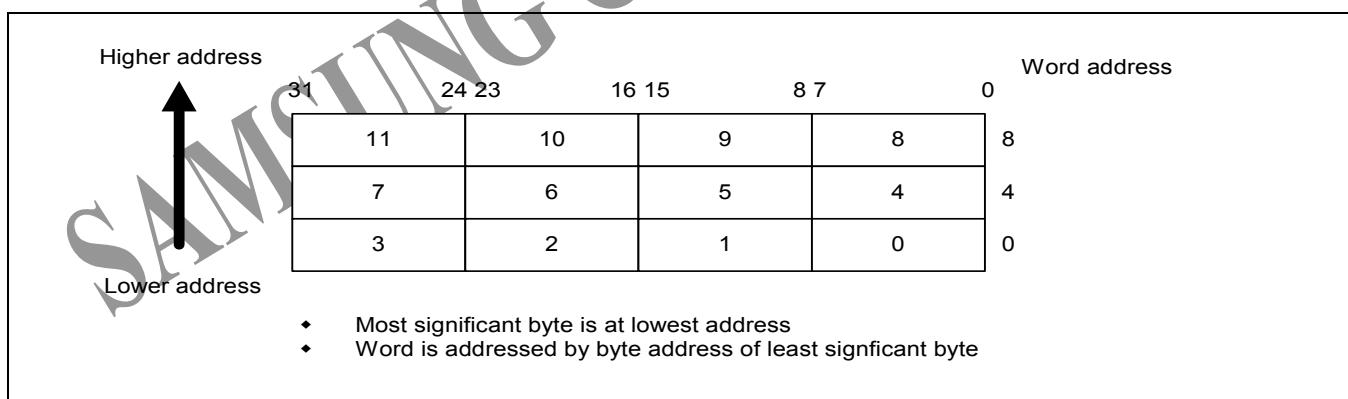


Figure 2-2. Little-Endian Addresses of Bytes Words

INSTRUCTION LENGTH

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

Data Types

ARM9TDMI supports byte (8-bit), half-word (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

ARM9TDMI 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

ARM9TDMI 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.

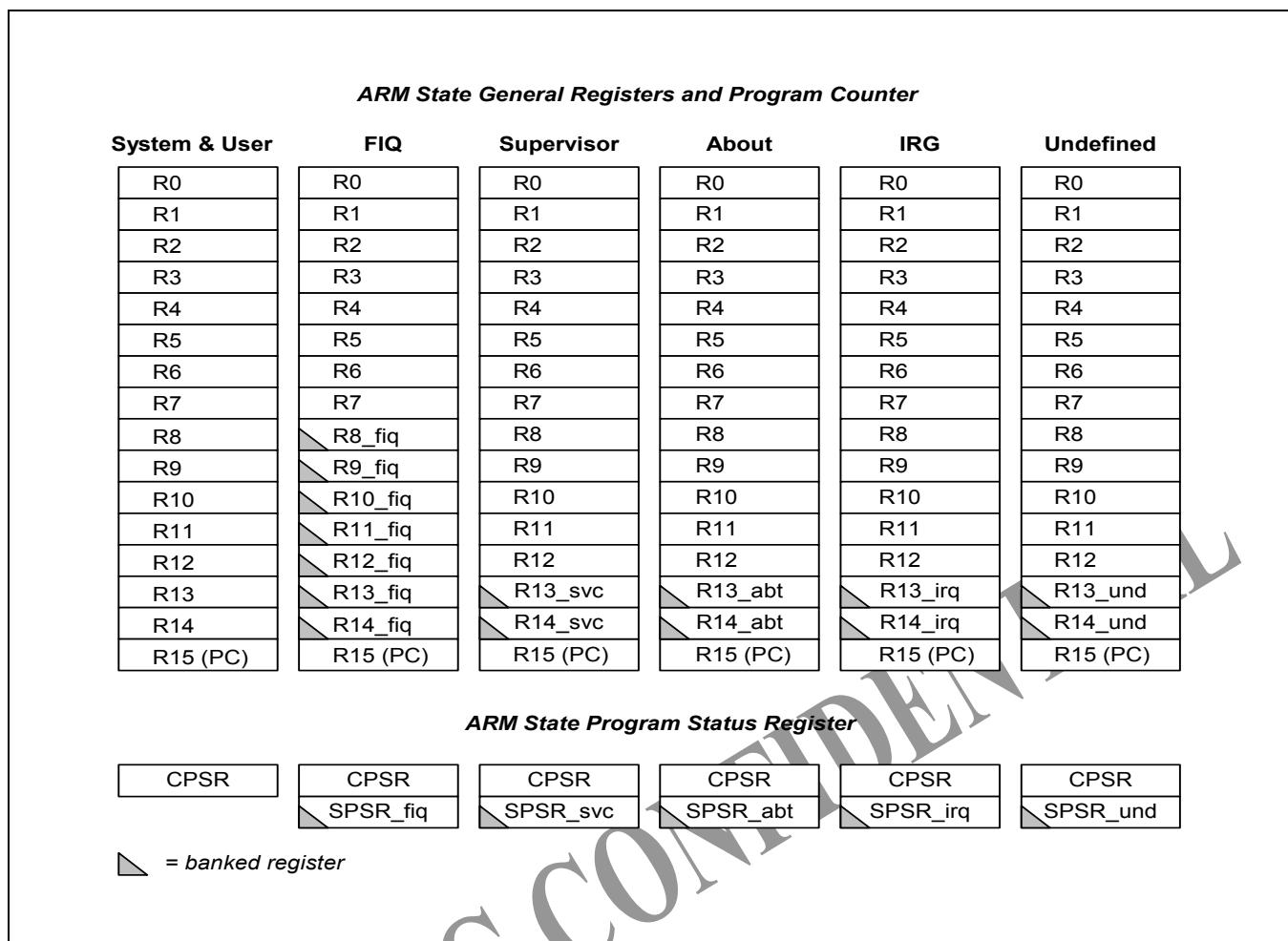


Figure 2-3. Register Organization in ARM State

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.

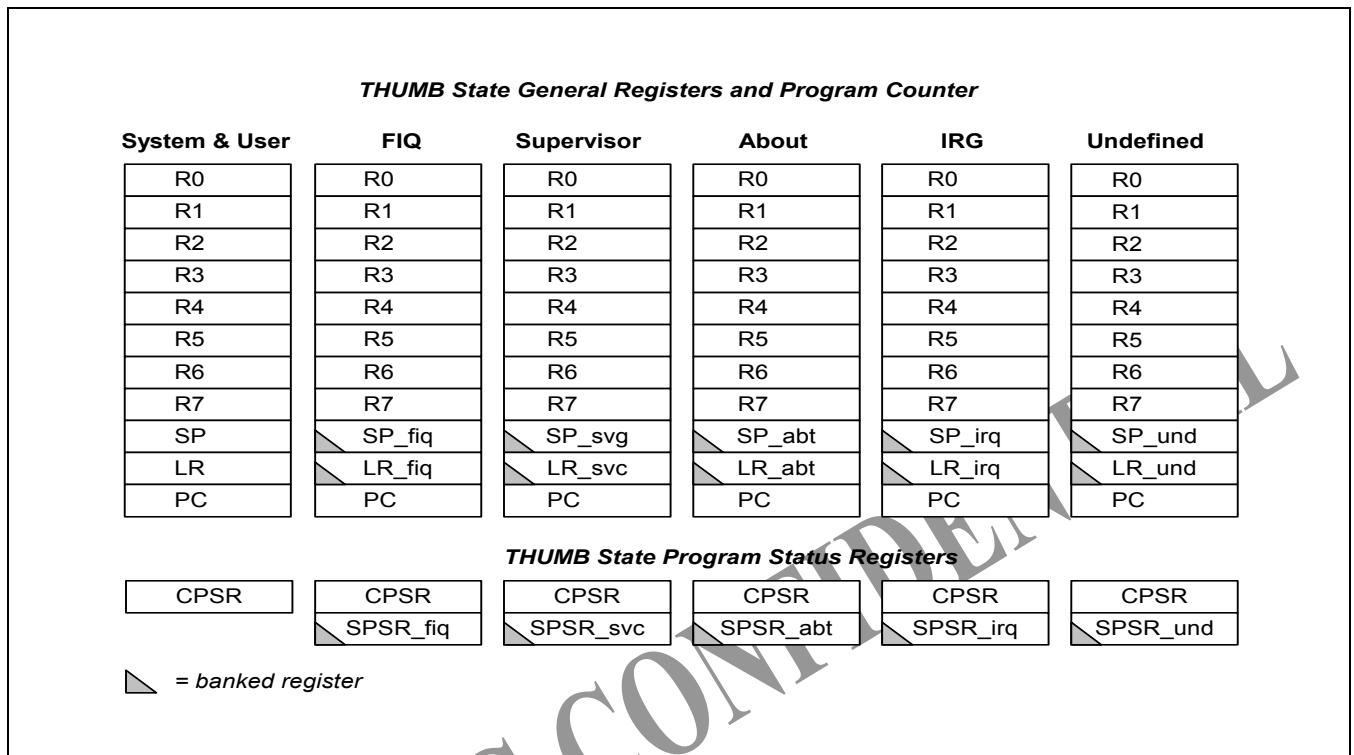


Figure 2-4. Register Organization in THUMB State

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.

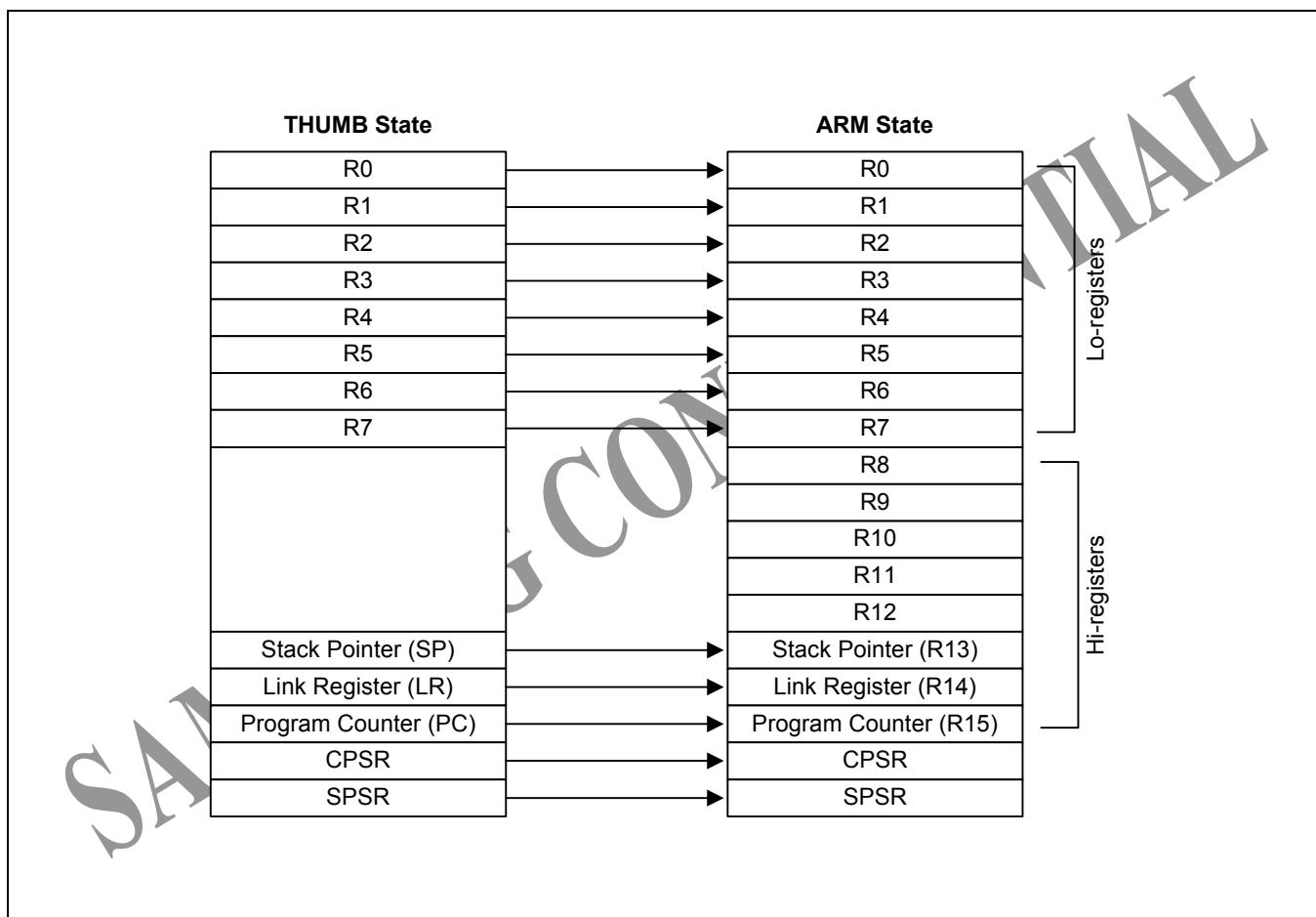


Figure 2-5. Mapping of THUMB State Registers onto ARM State Registers

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 ARM9TDMI 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.

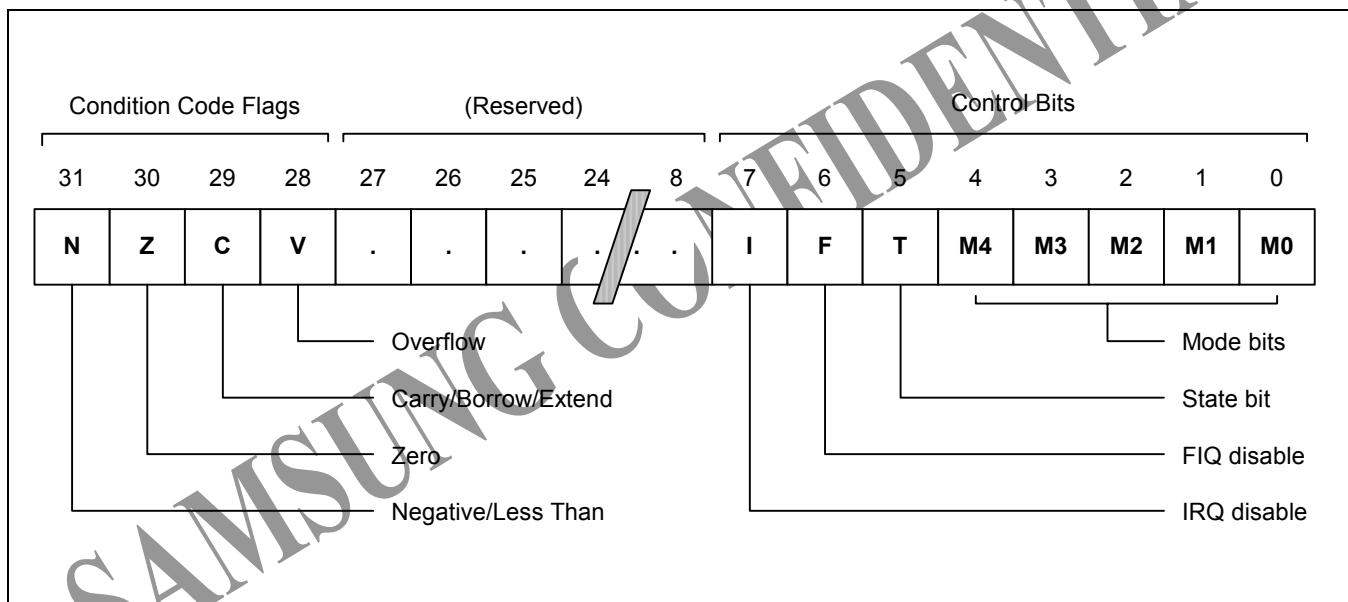


Figure 2-6. Program Status Register Format

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 change 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.

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 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.

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 ARM9TDMI:

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 nesting of exceptions.

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

Action on Leaving an Exception

On completion, the exception handler:

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

NOTE

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

Exception Entry/Exit Summary

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

Table 2-2. Exception Entry/Exit

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

NOTES:

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

FIQ

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

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

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

SUBS PC,R14_fiq,#4

FIQ may be disabled by setting the CPSR's F flag (but note that this is not possible from User mode). If the F flag is clear, ARM9TDMI 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 signaled by the external ABORT input. ARM9TDMI 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 ARM9TDMI CPU core.

Undefined Instruction

When ARM9TDMI comes across an instruction which it cannot handle, it takes the undefined instruction trap. This mechanism may be used to extend either the THUMB or ARM instruction set by software emulation.

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

```
MOVS     PC,R14_und
```

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

Exception Vectors

The following table shows the exception vector addresses.

Table 2-3. Exception Vectors

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

Exception Priorities

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

Highest priority:

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

Lowest priority:

6. Undefined Instruction, Software interrupt.

Not All Exceptions Can Occur at Once:

Undefined Instruction and Software Interrupt are mutually exclusive, since they each correspond to particular (non-overlapping) decoding 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), ARM9TDMI 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 ($T_{syncmax}$ if asynchronous), plus the time for the longest instruction to complete (T_{ldm} , the longest instruction is an LDM which loads all the registers including the PC), plus the time for the data abort entry (T_{exc}), plus the time for FIQ entry (T_{fiq}). At the end of this time ARM9TDMI will be executing the instruction at 0x1C.

$T_{syncmax}$ is 3 processor cycles, T_{ldm} is 20 cycles, T_{exc} is 3 cycles, and T_{fiq} 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 ($T_{syncmin}$) plus T_{fiq} . This is 4 processor cycles.

Reset

When the nRESET signal goes LOW, ARM9TDMI abandons the executing instruction and then continues to fetch instructions from incrementing word addresses.

When nRESET goes HIGH again, ARM9TDMI:

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.

INTRODUCTION FOR ARM940T

The ARM940T cached processor macrocell is a member of the ARM9 Thumb Family of high-performance 32-bit system-on-a-chip processor solutions. It is targeted at a wide range of embedded control applications where high performance, low system cost, small die size, and low power are key considerations.

The ARM940T processor macrocell provides a complete high performance CPU subsystem, including ARM9TDMI RISC integer CPU, caches, write buffer, and protection unit, with an AMBA ASB bus interface. Providing a complete high-frequency CPU subsystem frees the system-on-a-chip designer to concentrate on design issues unique to their system.

The ARM9TDMI core within the ARM940T macrocell executes both the 32-bit ARM and 16-bit Thumb instruction sets, allowing the user to trade off between high performance and high code density. It is binary compatible with ARM7TDMI, ARM10TDMI, and StrongARM processors, and is supported by a wide range of tools, operating systems, and application software.

The ARM940T processor macrocell is designed to be integrated into larger chips. It supports EmbeddedICE software and hardware debug and efficient production test when embedded in larger devices. The Advanced Microcontroller Bus Architecture (AMBA) provides a high performance 32-bit System Bus (ASB) and a low power peripheral bus (APB). The ASB is re-used to provide a channel for production test vectors with low silicon and pin overhead. The ASB is a multi-master on-chip bus interface designed specifically to address the needs of system-on-a-chip designs.

The EmbeddedICE software and hardware debug features of the ARM940T macrocell are accessed via a standard 5-pin JTAG port, and are supported by ARM's Software Development Toolkit and Multi-ICE interface hardware. The EmbeddedICE features allow software download and debug of the final production system with no cost overhead (there is no monitor code or other use of target resident RAM or ROM).

The ARM940T processor has a Harvard cache architecture with separate 4KB instruction and 4KB data caches, each with a 4-word line length. A protection unit allows 8 regions of memory to be defined, each with individual cache and write buffer configurations and access permissions. The cache system is software configurable to provide highest average performance or to meet the needs of real-time systems.

Software configurable options include:

- Random or round robin replacement algorithm
- Write-through or write-back cache operation (independently selectable for each memory region)
- Cache locking with granularity 1/64 th of cache size.

Overall, the cache and write buffers improve CPU performance and minimize accesses to the AMBA bus and to any off-chip memory, thus reducing overall system power consumption.

The ARM940T includes support for coprocessors, allowing a floating point unit or other application specific hardware acceleration to be added.

To minimize die size and power consumption the ARM940T does not provide virtual to physical address mapping as this is not required in most embedded applications. For systems requiring virtual memory capability, ARM provides an alternative product, the ARM920T cached processor macrocell. The ARM940T also features a TrackingICE mode which allows an approach similar to a conventional ICE mode of operation.

ARM940T BLOCK DIAGRAM

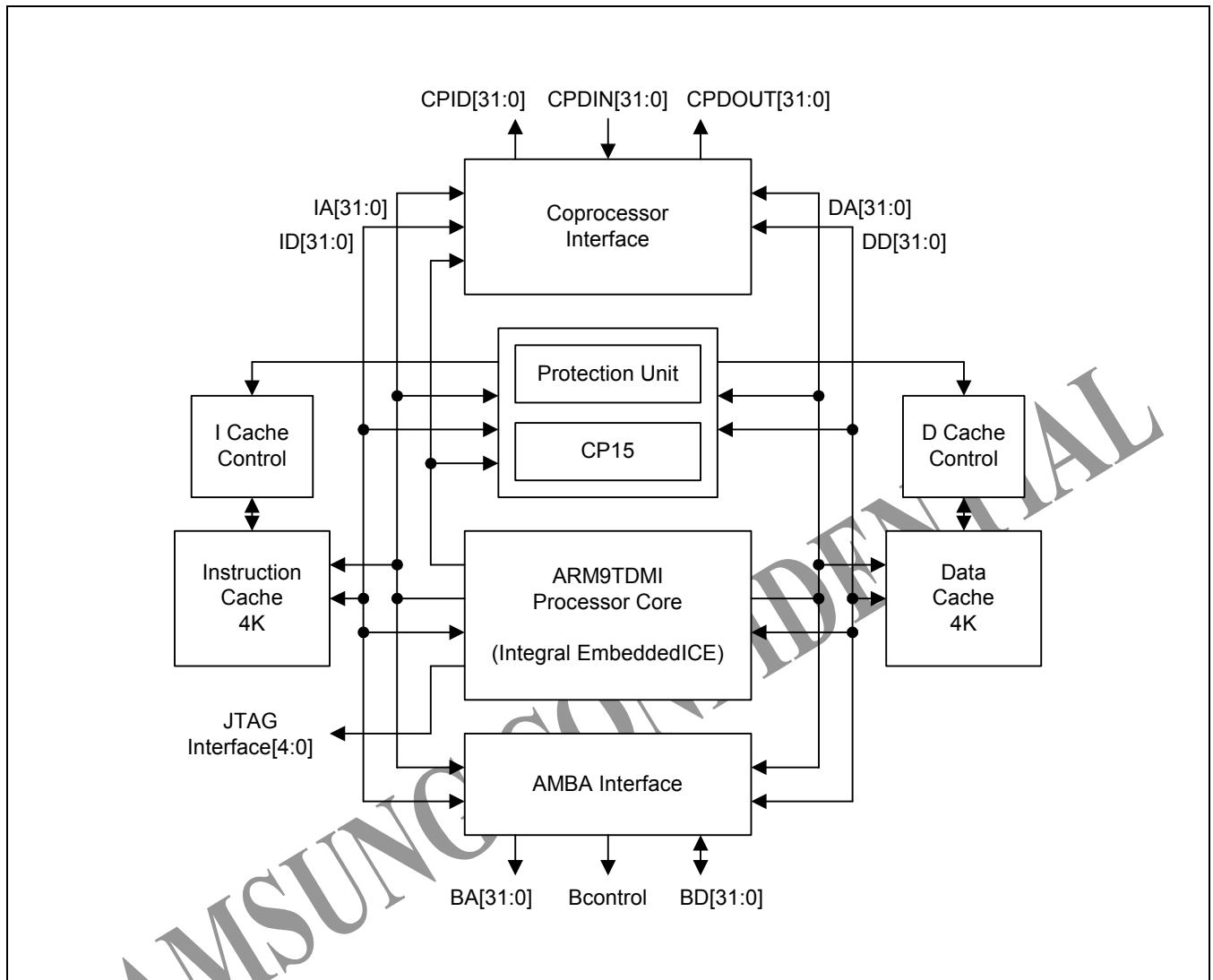


Figure 2-7. ARM940T Block Diagram

ABOUT THE ARM940T PROGRAMMER'S MODEL

The ARM940T cached processor macrocell includes the ARM9TDMI microprocessor core, instruction and data caches, a write-buffer, and a protection unit for defining the attributes of regions of memory.

The ARM940T incorporates two coprocessors:

- CP14 which allows software access to the debug communications channel
- CP15 which allows configuration of the caches, protection unit, and other system options such as big or little endian operation.

The ARM940T also features an external coprocessor interface which allows the attachment of a closely coupled coprocessor on the same chip, for example, a floating point unit.

The programmer's model of the ARM940T consists of the programmer's model of the ARM9TDMI (see About the ARM9TDMI programmer's model on page 2-3) with the following additions and modifications:

- Memory accesses for instruction fetches and data loads and stores may be cached or buffered. Cache and write buffer configuration and operation is described in detail in following chapters.
- The registers defined in CP14 are accessible with MCR and MRC instructions. These are described in Debug communications channel on page 8-46.
- The registers defined in CP15 are accessible with MCR and MRC instructions. These are described in ARM940T CP15 registers on page 2-5.
- Registers and operations provided by any coprocessors attached to the external coprocessor interface will be accessible with appropriate coprocessor instructions.

The ARM9TDMI processor core implements ARM Architecture v4T, and so executes the ARM 32-bit instruction set and the compressed Thumb 16-bit instruction set. The programmer's model is fully described in the ARM Architecture Reference Manual.

The ARM v4T architecture specifies a small number of implementation options. The options selected in the ARM9TDMI implementation are listed in Table 2-4. For comparison, the options selected for the ARM9TDMI implementation are also shown.

Table 2-4. ARM9TDMI Implementation Option

Processor Core	ARM Architecture	Data Abort Mode	Value Stored by Direct STR, STRT, STM of PC
ARM7TDMI	v4T	Base updated	Address of Inst + 12
ARM9TDMI	v4T	Base restored	Address of Inst + 12

The ARM9TDMI is code-compatible with the ARM7TDMI, with two exceptions:

- The ARM9TDMI implements the base restored data abort model, which significantly simplifies the software data abort handler.
- The ARM9TDMI fully implements the instruction set extension spaces added to the ARM (32-bit) instruction set in architecture v4 and v4T.

These differences are explained in more detail below.

DATA ABORT MODEL

The base restored data abort model differs from the base updated data abort model implemented by ARM7TDMI. The difference in the data abort model affects only a very small section of operating system code, the data abort handler. It does not affect user code. With the base restored data abort model, when a data abort exception occurs during the execution of a memory access instruction, the base register is always restored by the processor hardware to the value the register contained before the instruction was executed. This removes the need for the data abort handler to unwind any base register update which may have been specified by the aborted instruction. The base restored data abort model significantly simplifies the software data abort handler.

Instruction set extension spaces

All ARM processors implement the undefined instruction space as one of the entry mechanisms for the undefined instruction exception. That is, ARM instructions with opcode[27:25] = 0b011 and opcode[4] = 1 are undefined on all ARM processors including the ARM9TDMI and ARM7TDMI.

ARM Architecture v4 and v4T also introduced a number of instruction set extension spaces to the ARM instruction set. These are:

- Arithmetic instruction extension space
- Control instruction extension space
- Coprocessor instruction extension space
- Load/store instruction extension space.

Instructions in these spaces are undefined (they cause an undefined instruction exception). The ARM9TDMI fully implements all the instruction set extension spaces defined in ARM Architecture v4T as undefined instructions, allowing emulation of future instruction set additions.

ARM940T CP15 REGISTERS

CP15 Register Map Summary

The ARM940T incorporates CP15 for system control. The register map for C15 is shown in Table 2-5.

Table 2-5. CP15 Register Map

Register	Function	Access
0	ID code/Cache type	See note below
1	Control	Read/write
2	Cacheable	See note below
3	Write buffer control	Read/write
4	Reserved	Undefined
5	Protection region access permissions	See note below
6	Protection region base/size control	See note below
7	Cache operations	Write only. Reads unpredictable
8	Reserved	Undefined
9	Cache lockdown	Read/write
10-14	Reserved	Undefined
15	Test	Not accessed in normal operations

NOTE: Register locations 0, 2, 5, and 6 each provide access to more than one register. The register accessed depends upon the value of the opcode_2 field. See the register descriptions that follow for further information.

Register 0: ID code

This is a read-only register which returns a 32-bit device ID code. The ID code register is accessed by reading CP15 register 0 with the opcode_2 field set to any value other than 1. For example:

MRC cp15, 0, rd, c0, c0,{0,2-7}; returns ID register

The contents of the ID code are shown in Table 2-6.

Table 2-6. ID Code Register

Register Bits	Function	Value
31:12	Implementor	0x41 (identifies ARM)
23:16	Architecture version	0x2
15:4	Part number	0x940
3:0	Version	0x1

Register 0: Cache type

This is a read-only register which allows operating systems to establish how to perform operations such as cache cleaning and lockdown. Future ARM cached processors will contain this register, allowing RTOS vendors to produce future-proof versions of their operating systems.

The cache type register is accessed by reading CP15 register 0 with the opcode_2 field set to 1. For example:

MRC cp15, 0, rd, c0, c0, 1; returns Cache type register

The register contains information about the size and architecture of the caches. The format of the register is shown in Table 2-7.

Table 2-7. Cache Type Register Format

Register Bits	Meaning	Value
31:29	Reserved	000
28:25	Cache type	0111
24	Harvard/Unified	1 (defines Harvard cache)
23:21	Reserved	000
20:18	DCache size	011 (defines 4KB)
17:15	DCache associativity	110 (defines 64 way)
14	DCache base size	0 (defines 1x base parameters)
13:12	DCache words per line	01 (defines 4 words per line)
11:9	Reserved	000
8:6	ICache size	011 (defines 4KB)
5:3	ICache Associativity	110 (defines 64 way)
2	ICache base size	0 (defines 1x base parameters)
1:0	ICache words per line	01 (defines 4 words per line)

Register 1: Control register

This contains the global control bits of the ARM940T. All reserved bits should either be written with zero or one, as indicated, or written using read-modify-write. The reserved bits have an unpredictable value when read.

Table 2-8. CP15 Register 1

Register Bits	Functions
31	Asynchronous clocking select (iA)
30	nFastBus select (nF)
29:14	Reserved (should be zero)
13	Alternate vectors select (V)
12	ICache enable bit (1)
11:8	Reserved (should be zero)
7	Big-end bit (E)
6:3	Reserved (should be one)
2	DCache enable bit (D)
1	Reserved (should be zero)
0	Protection unit enable (P)

The bits in the control register have the following functions:

- Bits 31:30 Control the clocking mode of the processor, as shown in Table 2-9. Clocking modes are discussed in Chapter5 Clock Modes.

Table 2-9. Clocking Modes

Clockin Mode	Bit 31	Bit 30
FastBus mode	0	0
Reserved	1	0
Synchronous	0	1
Asynchronous	1	1

- Bit 13 Selects the location of the vector table. During reset, the bit is cleared and the vector table is located at address 0x00000000. When bit 13 is set, the vector table is relocated to address 0xfffff0000.
- Bits 12 and 2 Enable the caches (see Chapter4 Caches and Write Buffer).
- Bit 7 Selects the endian configuration of the ARM940T. Setting bit 7 selects a big-endian configuration. Clearing bit 7 selects a little-endian configuration. Bit 7 is cleared during reset.
- Bit 0 Enables the protection unit.

Register 2: Instruction and data cacheable registers

This location provides access to two registers which contain the cacheable attributes for each of eight memory areas. The two registers provide individual control for the I and D address spaces. The opcode_2 field determines whether the instruction-or data-cacheable attributes are programmed:

If the opcode_2 field = 0, the data-cacheable bits are programmed. For example:

MCR p15,0,Rd,c2,c0,0; Write data-cacheable bits

MRC p15,0,Rd,c2,c0,0; Read data-cacheable bits

If the opcode_2 field = 1 the instruction-cacheable bits are programmed. For example:

MCR p15,0,Rd,c2,c0,1; Write instruction cacheable bits

MRC p15,0,Rd,c2,c0,1; Read instruction cacheable bits

The format for the data and instruction cacheable bits is similar, as shown in Table2-7. Setting a bit makes an area cacheable, clearing it makes it non-cacheable.

Table 2-10. Cacheable Register Format

Register Bits	Functions
7	Cacheable bit (C_7) for area 7
6	Cacheable bit (C_6) for area 6
5	Cacheable bit (C_5) for area 5
4	Cacheable bit (C_4) for area 4
3	Cacheable bit (C_3) for area 3
2	Cacheable bit (C_2) for area 2
1	Cacheable bit (C_1) for area 1
0	Cacheable bit (C_0) for area 0

The use of register 2 is discussed in Chapter 3 Protection Unit.

Register 3: Write buffer control register

This register contains a write buffer control (bufferable) attribute bit for each of the eight areas of memory. Each bit is used in conjunction with the cacheable bit to control write-buffer operation. For a description of buffer behavior, see The write buffer on page 4-11.

Setting a bit makes an area bufferable, clearing a bit makes an area unbuffered. For example:

MCR p15,0,Rd,c3,c0,0; Write data-bufferable bits

MRC p15,0,Rd,c3,c0,0; Read data-bufferable bits

NOTE

The opcode_2 field should be 0 because the write buffer only operates on data regions. The following table, therefore, only applies to the DCache.

Table 2-11. Write Buffer Control Register

Register Bits	Functions
7	Write buffer control bit (B_d7) for data area 7
6	Write buffer control bit (B_d6) for data area 6
5	Write buffer control bit (B_d5) for data area 5
4	Write buffer control bit (B_d4) for data area 4
3	Write buffer control bit (B_d3) for data area 3
2	Write buffer control bit (B_d2) for data area 2
1	Write buffer control bit (B_d1) for data area 1
0	Write buffer control bit (B_d0) for data area 0

Register 5: Instruction and data space protection registers

These registers contain the access permission bits for the instruction and data protection regions. The opcode_2 field determines whether the instruction or data access permissions are programmed.

If the opcode_2 field = 0, the data space bits are programmed. For example:

MCR p15,0,Rd,c5,co,0; Write data space access permissions

If the opcode_2 field =1, the instruction space bits are programmed. For example:

MCR p15,0,Rd,c5,co,1; Write instruction space access permissions

MRC p15,0,Rd,c5,co,1; Read instruction space access permissions

Each register contains the access permission bits, apn[1:0], for the eight areas of instruction or data memory, as shown in Table 2-12.

Table 2-12. Protection Space Register Format

Register Bits	Functions
15:14	ap7[1:0] bits of area 7
13:12	ap6[1:0] bits of area 6
11:10	ap5[1:0] bits of area 5
9:8	ap4[1:0] bits of area 4
7:6	ap3[1:0] bits of area 3
5:4	ap2[1:0] bits of area 2
3:2	ap1[1:0] bits of area 1
1:0	ap0[1:0] bits of area 0

The values of the lapn[1:0] and Dapn[1:0] bits define the access permission for each area of memory. The encoding is shown in Table 2-13.

NOTE

On reset, the values of the lapn[1:0] and Dapn[1:0] bits for all areas are undefined. However, as on reset, the protection unit is disabled and all areas are effectively set to no access. The protection space registers therefore, must be programmed before the protection unit is enabled.

Table 2-13. Permission Encoding

I/Dapn[1:0]	Permission
00	No access.
01	Privileged mode access only.
10	Privileged mode full access, user mode read only.
11	Full access.

Register 6: Protection region base and size registers

This register is used to define 16 programmable regions (eight instruction, eight data) in memory. These registers define the base and size of each of the eight areas of memory. Individual control is provided for the instruction and data memory regions. The values are ignored when the protection unit is disabled.

On reset, only the region enable bit for each region is reset to 0, all other bits are undefined. At least one instruction and data memory region must be programmed before the protection unit is enabled.

The opcode_2 field defines whether the data or instruction protection regions are to be programmed. The CRm field selects the region number.

Table 2-14. CP15 Data Protection Region Registers

ARM instruction	Protection region register
MCR/MRC p15, 0, Rd, c6, c7, 0	Data memory region 7
MCR/MRC p15, 0, Rd, c6, c6, 0	Data memory region 6
MCR/MRC p15, 0, Rd, c6, c5, 0	Data memory region 5
MCR/MRC p15, 0, Rd, c6, c4, 0	Data memory region 4
MCR/MRC p15, 0, Rd, c6, c3, 0	Data memory region 3
MCR/MRC p15, 0, Rd, c6, c2, 0	Data memory region 2
MCR/MRC p15, 0, Rd, c6, c1, 0	Data memory region 1
MCR/MRC p15, 0, Rd, c6, c0, 0	Data memory region 0

Table 2-15. CP15 Instruction Protection Region Registers

ARM instruction	Protection region register
MCR/MRC p15, 0, Rd, c6, c7, 1	Instruction memory region 7
MCR/MRC p15, 0, Rd, c6, c6, 1	Instruction memory region 6
MCR/MRC p15, 0, Rd, c6, c5, 1	Instruction memory region 5
MCR/MRC p15, 0, Rd, c6, c4, 1	Instruction memory region 4
MCR/MRC p15, 0, Rd, c6, c3, 1	Instruction memory region 3
MCR/MRC p15, 0, Rd, c6, c2, 1	Instruction memory region 2
MCR/MRC p15, 0, Rd, c6, c1, 1	Instruction memory region 1
MCR/MRC p15, 0, Rd, c6, c0, 1	Instruction memory region 0

Each protection region register has the format shown in Table 2-16.

Table 2-16. CP15 Protection Region Register Format

Register bit	Function
31:12	Base address
11:6	Unused
5:1	Area size (See Table2-17)
0	Region enable. Reset to disable (0).

The region base must be aligned to an area size boundary, where the area size is defined in its respective protection region register. The behavior is undefined if this is not the case. Area sizes are given in Table 2-17.

Table 2-17. Area Size Encoding

Bit encoding	Area size	Bit encoding	Area size
00000 to 01010	Reserved	10101	4MB
01011	4KB	10110	8MB
01100	8KB	10111	16MB
01101	16KB	11000	32MB
01110	32KB	11001	64MB
01111	64KB	11010	128MB
10000	128KB	11011	256MB
10001	256KB	11100	512MB
10010	512KB	11101	1GB
10011	1MB	11110	2GB
10100	2MB	11111	4GB

Example Base Setting

An 8KB size region aligned to the 8KB boundary at 0x00002000 (covering the address range 0x00002000–0x00003fff) would be programmed to 0x00002019.

Register 7: Cache operations

A write to this register can be used to perform the following operations:

- Flush ICache and Dcache
- Prefetch an ICache line
- Wait for interrupt
- Drain the write buffer
- Clean and flush the DCache.

The ARM940T uses a subset of the architecture V4 functions (defined in the ARM Architecture Reference Manual). The available operations are summarized in Table 2-18 and described below.

Table 2-18. Cache Operations Writing to Register 7

ARM instruction	Data	Protection region register
MCR p15, 0, Rd, c7, c5, 0	should be zero	Flush ICache.
MCR p15, 0, Rd, c7, c5, 2	Index/segment	Flush ICache single entry.
MCR p15, 0, Rd, c7, c6, 0	should be zero	Flush DCache.
MCR p15, 0, Rd, c7, c6, 2	Index/segment	Flush DCache single entry.
MCR p15, 0, Rd, c7, c10, 2	Index/segment	Clean DCache single entry.
MCR p15, 0, Rd, c7, c13, 1	Address	Prefetch ICache line.
MCR p15, 0, Rd, c7, c14, 2	Index/segment	Clean and flush DCache single entry.
MCR p15, 0, Rd, c7, c8, 2	should be zero	Wait for interrupt.
MCR p15, 0, Rd, c7, c10, 2	should be zero	Drain write buffer.

"Should be zero" means the value transferred in the Rd.

A read from this register returns an unpredictable value.

Index/Segment Format

Where the required value is an index/segment, the format is:

Table 2-19. CP15 Register 7 Index/Segment Data Format

Rd bit position	Function
31:26	Index
25:6	Should be zero
5:4	Segment
3:0	Should be zero

ICache Prefetch Data Format

For the ICache prefetch operation, the data format is:

Table 2-20. CP15 Register 7 Prefetch Address Format

Rd bit position	Function
31:6	Address bits 31:6
5:4	Cache segment
3:0	Should be zero

Wait for interrupt

This operation allows the ARM940T to be placed in a low-power standby mode. When the operation is invoked, all clocks in the processor are frozen until either an interrupt or a debug request occurs. This function is invoked by a write to register 7. The following ARM instruction causes this to occur:

MCR p15, 0, Rd, c7, c0, 4

The following instruction causes the same affect and has been added for backward compatibility with StrongARM SA-1

MCR p15, 0, Rd, c15, c8, 2

This stalls the processor, with internal clocks held high from the time that this instruction is executed until one of the signals nFIQ, nIRQ, or EDBGREQ is asserted. Also, if the debugger sets the debug request bit in the EmbeddedICE unit control register, the wait-for-interrupt condition is terminated.

In the case of nFIQ and nIRQ, the processor is woken up regardless of whether the interrupts are enabled or disabled (that is, independent of the I and F bits in the processor CPSR). The debug related waking only occurs if DBGEN is HIGH, that is, only when debug is enabled.

If the interrupts are enabled, the ARM is guaranteed to take the interrupt before executing the instruction after the wait-for-interrupt. If debug request is used to wake up the system, the processor will enter debug-state before executing any further instructions.

Drain Write Buffer

This CP15 operation causes instruction execution to be stalled until the write buffer is emptied. This operation is useful in real time applications where the processor needs to be sure that a write to a peripheral has completed before program execution continues. An example would be where a peripheral in a bufferable region is the source of an interrupt. Once the interrupt has been serviced, the request must be removed before interrupts can be re-enabled. This can be ensured if a drain write buffer operation separates the store to the peripheral and the enable interrupt functions.

The drain write buffer function is invoked by a write to CP15 register 7 using the following ARM instruction:

MCR p15, 0, Rd, c7, c10, 4

This stalls the processor core, with CPnWAIT asserted until any outstanding accesses in the write buffer have been completed (that is, until all data has been written to memory).

Register 9: Instruction and data lockdown registers

These registers allow regions of the cache to be locked down. The opcode_2 field determines whether the instruction or data caches are programmed.

- If the opcode_2 field = 0, the data lockdown bits are programmed. For example:

MCR/MRC p15, 0, Rd, c9, c0, 0; data lockdown control

- If the opcode_2 field = 1, the instruction lockdown bits are programmed. For example:

MCR/MRC p15, 0, Rd, c9, c0, 1; instruction lockdown control

The format of the registers, Rd, transferred during this operation, is shown below:

Table 2-21. Lockdown Register Format

Register bit	Function
31	Load bit
30:6	Reserved
5:0	Cache index

NOTE: The segment number is not specified because cache lines are locked down across all four segments (16-word granularity).

Register 15: Test/debug register

The DTRRobin and ITRRobin bits set the respective caches into a pseudo round-robin replacement mode.

Table 2-22. CP15 Register 15

Register bit	Function
31:4	Reserved
3	ITRRobin
2	DTRRobin
1:0	Reserved

Reserved registers

Accessing a reserved register is unpredictable.

NOTES

SAMSUNG CONFIDENTIAL

3 INSTRUCTIONS SET

INSTRUCTION SET SUMMARY

This chapter describes the ARM instruction set and the THUMB instruction set in the ARM9TDMI core.

The ARM940T implements uses a five-stage pipeline design. These five stages are:

- instruction fetch (F)
- instruction decode (D)
- execute (E)
- data memory access (M)
- register write (W)

ARM implementations are fully interlocked, so that software will function identically across different implementations without concern for pipeline effects. Interlock do affect instruction times. For example, the following sequence suffers a single cycle penalty due to a load-use interlock on register r0:

LDR	R0,[R7]
ADD	R5,R0,R1

ABOUT THE INSTRUCTION CYCLE SUMMARY

All signals quoted in this chapter are ARM9TDMI signals, and are internal to the ARM940T. In all cases it is assumed that all accesses are from cached regions of memory.

If an instruction causes an external access, either when prefetching instructions or when accessing data, the instruction takes more cycles to complete execution. The additional number of cycles is dependent on the system implementation.

INSTRUCTION CYCLE TIMES

Table 3-1 shows a key to the symbols used in tables in this section

Table 3-1. Symbol Used in Tables

Symbol	Meaning
B	The number of busy-wait states during coprocessor accesses
M	Is in the range 0 to 3, depending on early termination
N	The number of words transferred in an LDM/STM/LDC/STC
C	Coprocessor register transfer (C-cycle)
I	Internal cycle (I-cycle)
N	Non-sequential cycle (N-cycle)
S	Sequential cycle (S-cycle)

Table 3-2 summarize the ARM940T instruction cycle counts and bus activity when executing the ARM instruction set.

Table 3-2. Instruction Cycle Bus Times

Instruction	Cycle	Instruction Bus	Data Bus	Comment
Data Op	1	1S	1I	Normal case
Data Op	2	1S+1I	2I	With register controlled shift
LDR	1	1S	1N	Normal case, not loading PC
LDR	2	1S+1I	1N+1I	Not loading PC and following instruction uses loaded word (1 cycle load-use interlock)
LDR	3	1S+2I	1N+2I	Loaded byte, halfword, or unaligned word used by following instruction (2 cycle load-use interlock)
LDR	5	2S+2I+1N	1N+4I	PC is destination register
STR	1	1S	1N	All cases
LDM	2	1S+1I	1S+1I	Loading 1 Register, not the PC
LDM	N	1S+(n-1)I	1N+(n-1)S	Loading n registers, n > 1, not loading the PC
LDM	n+4	2S+1N+(n+1)I	1N+(n-1)S+4I	Loading n registers including the PC, n > 0
STM	2	1S+1I	1N+1I	Storing 1 Register
STM	N	1S+(n-1)I	1N+(n-1)S	Storing n registers, n > 1
SWP	2	1S+1I	2N	Normal case
SWP	3	1S+2I	2N+1I	Loaded byte used by following instruction
B, BL, BX	3	2S+1N	3I	All cases
SWI, Undefined	3	2S+1N	3I	All cases
CDP	b+1	1S+bl	(1+b)I	All cases
LDC, STC	b+n	1S+(b+n-1)I	bl+1N+(n-1)S	All cases
MCR	b+1	1S+bl	bl+1C	All cases
MRC	b+1	1S+bl	bl+1C	Normal case
MRC	b+2	1S+(b+1)I	(b+l)l+1C	Following instruction uses transferred data
MUL, MLA	2+m	1S+(1+m)I	(2+m)I	All cases
SMULL, UMULL, SMLAL, UMLAL	3+m	1S+(2+m)I	(3+m)I	All cases

Table 3-3 shows the instruction cycle times from the perspective of the data bus.

Table 3-3. Data Bus Instruction Times

Instruction	Cycle time
LDR	1N
STR	1N
LDM, STM	1N+(n-1)S
SWP	1N+1S
LDC, STC	1N+(n-1)S
MCR, MRC	1C

MULTIPLIER CYCLE COUNTS

The number of cycles that a multiply instruction takes to complete depends on which instruction, and on the value of the multiplier-operand. The multiplier-operand is the contents of the register specified by bits [11:8] of the ARM multiply instructions, or bits [2:0] of the Thumb multiply instructions.

For ARM MUL, MLA, SMULL, SMLAL, and Thumb MUL, m is:

1. if bits [31:8] of the multiplier operand are all 0 or all 1
2. if bits [31:16] of the multiplier operand are all 0 or all 1
3. if bits [31:24] of the multiplier operand are all 0 or all 1
4. otherwise.

For ARM UMULL, UMLAL, m is:

1. if bits [31:8] of the multiplier operand are all 0
2. if bits [31:16] of the multiplier operand are all 0
3. if bits [31:24] of the multiplier operand are all 0
4. otherwise.

INTERLOCKS

Pipeline interlocks occur when the data required for an instruction is not available due to the incomplete execution of an earlier instruction. When an interlock occurs, instruction fetches stop on the instruction memory interface of the ARM940T. Four examples are given in.

- Example 3-1
- Example 3-2
- Example 3-3
- Example 3-4

Example 3-1: Single load interlock

In this example, the following code sequence is executed:

```
LDR R0, [R1]
ADD R2, R0, R1
```

The ADD instruction cannot start until the data is returned from the load. The ADD instruction therefore, has to delay entering the Execute stage of the pipeline by one cycle. The behavior on the instruction memory interface is shown in Figure 3-1.

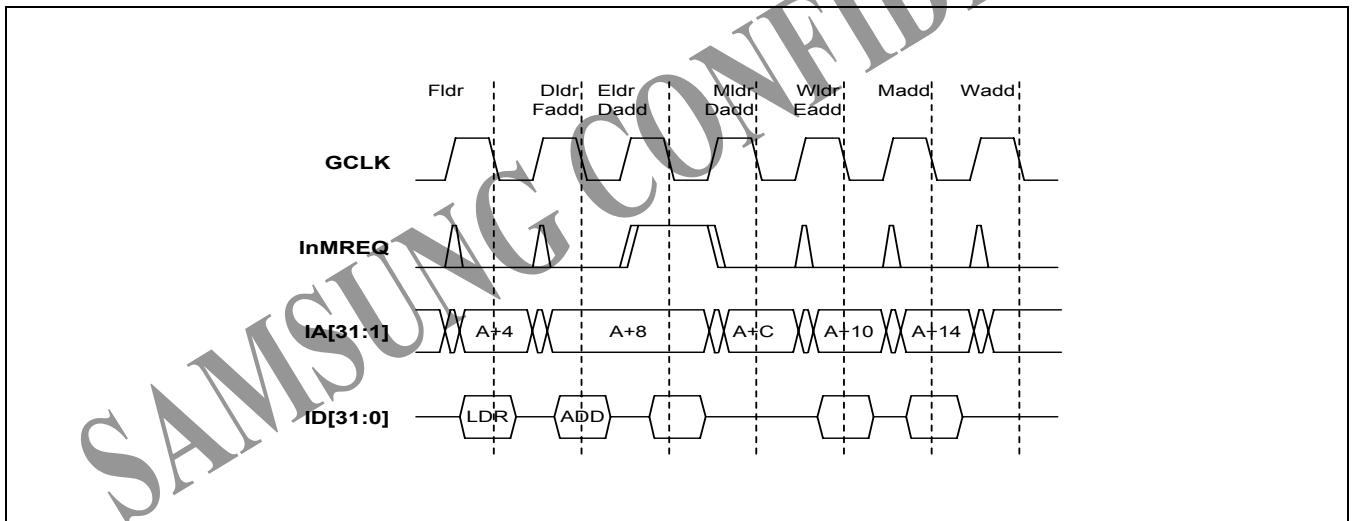


Figure 3-1. Single Load Interlock Timing

Example 3-2: Two cycle load interlock

In this example, the following code sequence is executed:

LDRB R0, [R1, #1]

ADD R2, R0, R1

Now, because a rotation must occur on the loaded data, there is a second interlock cycle. The behavior on the instruction memory interface is shown in Figure 3-2.

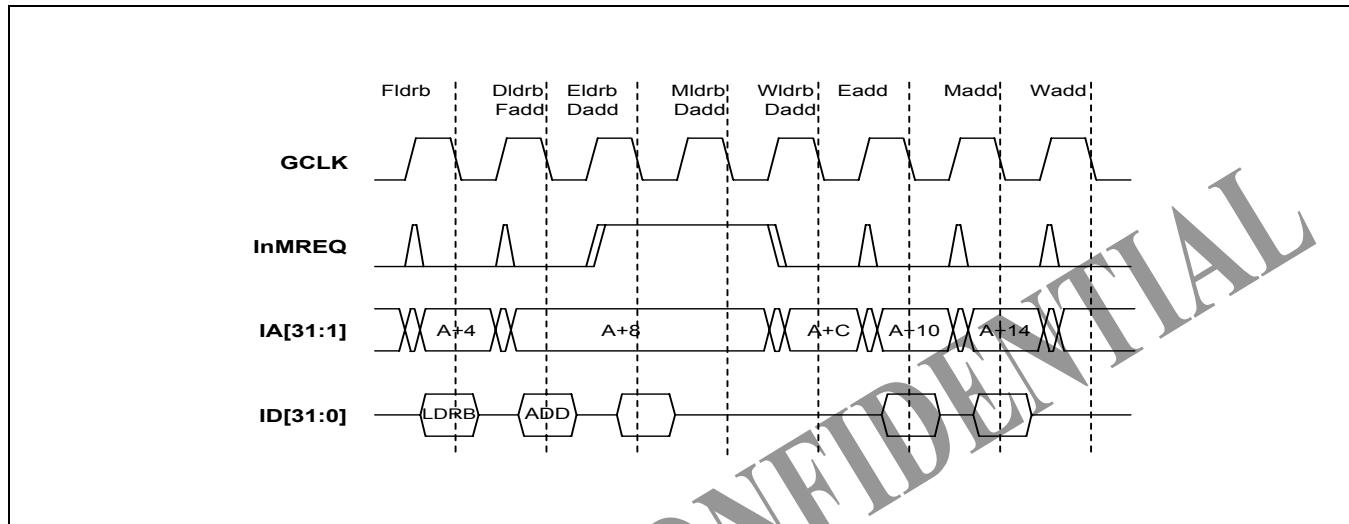


Figure 3-2. Tow Cycle Load Interlock Timing

Example 3-3: Ldm interlock

In this example, the following code sequence is executed:

LDM R12, {R1–R3}

ADD R2, R2, R1

The LDM takes three cycle to execute in the Memory stage of the pipeline. The ADD is therefore delayed until the LDM begins its final memory fetch. The behavior on the instruction memory interface is shown in Figure 3-3.

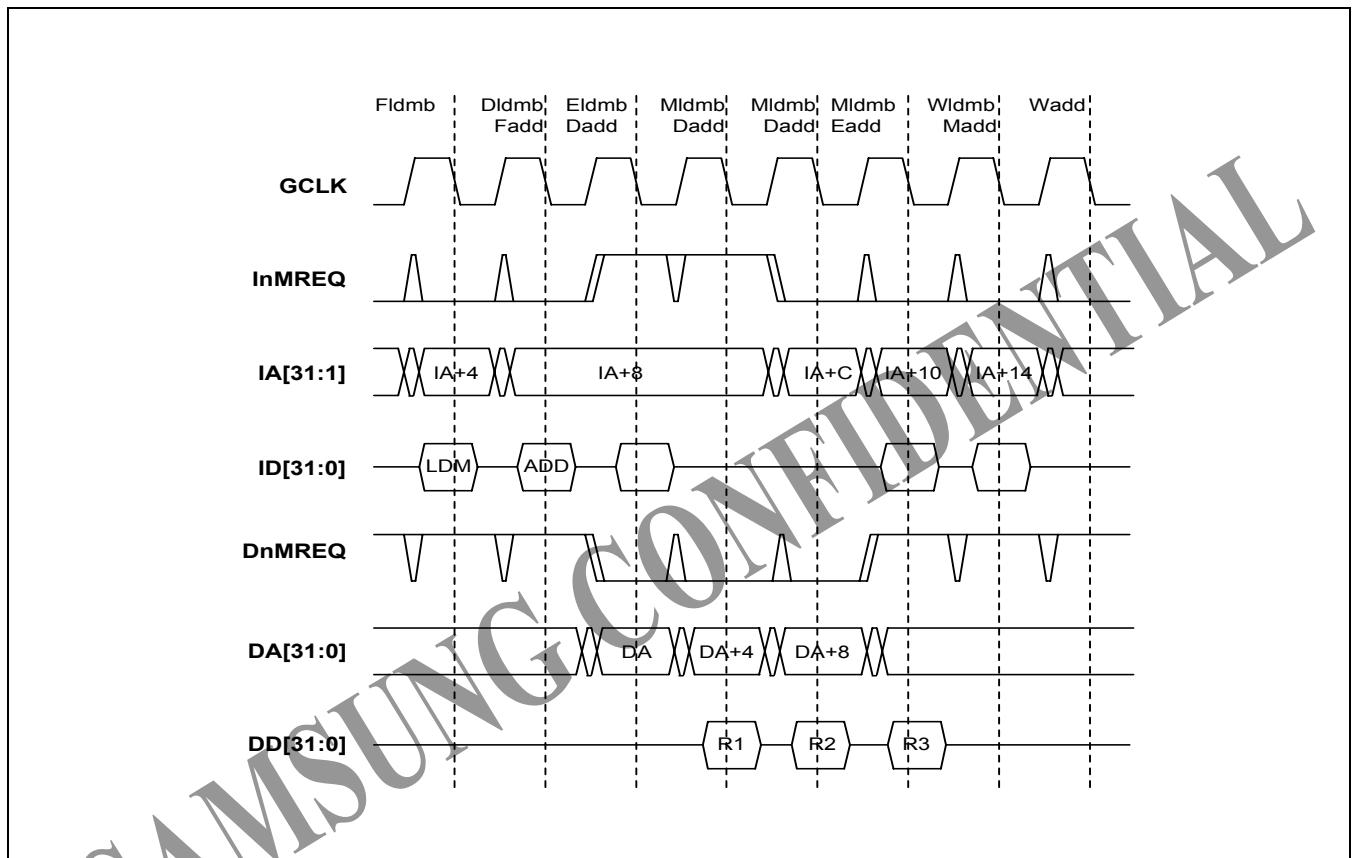


Figure 3-3. LDM Interlock Timing

Example 3-4: LDM Dependent Interlock

In this example, the following code sequence is executed:

LDM R12, {R1–R3}

ADD R4, R3, R1

The code is the same code as in example 3-3, but in this instance the ADD instruction uses R3. Due to the nature of load multiples, the lowest register specified is transferred first, and the highest specified register last. Because the ADD is dependent on R3, there must be another cycle of interlock while R3 is loaded. The behavior on the instruction and data memory interface is shown in Figure 3-4.

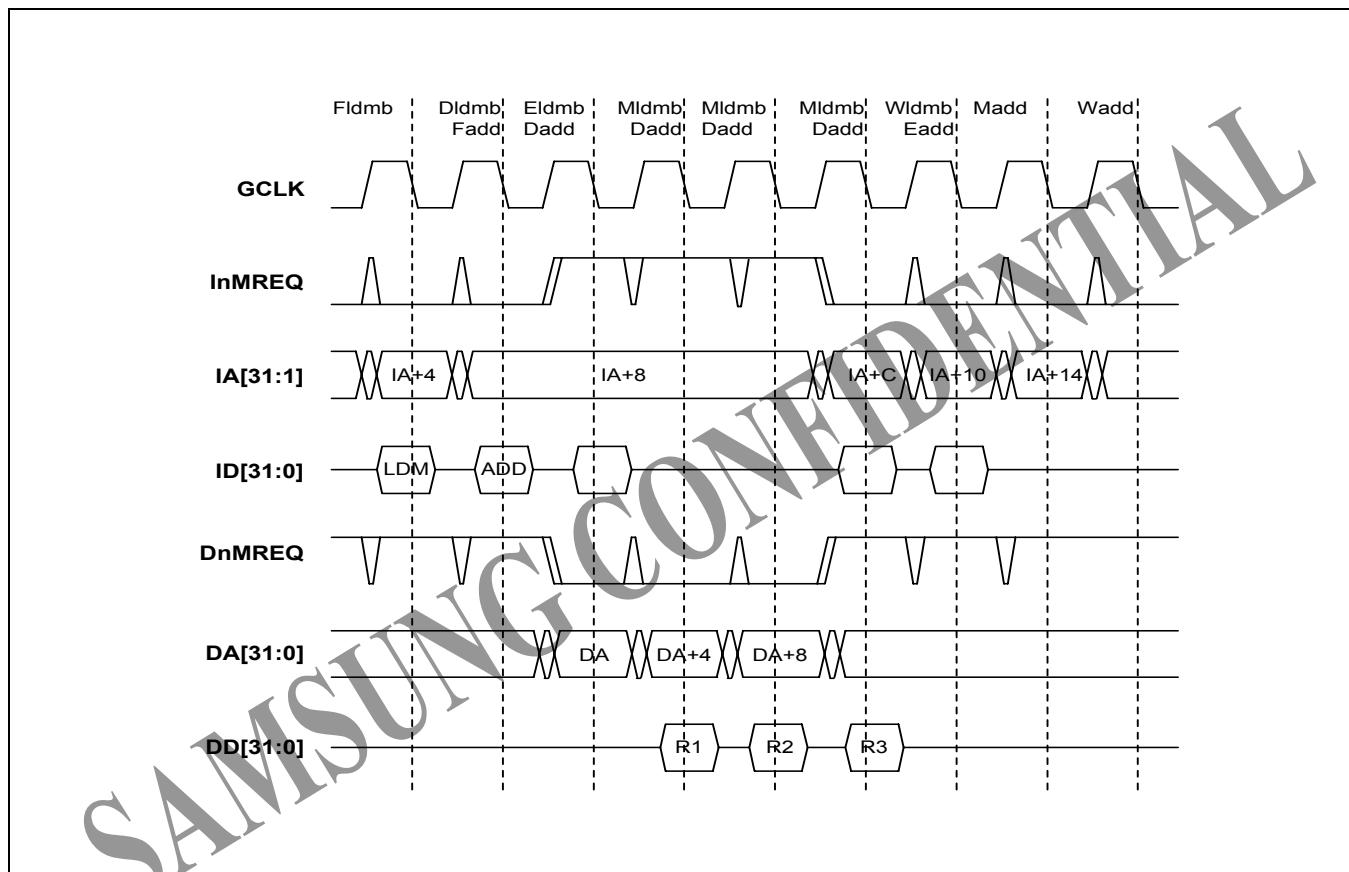
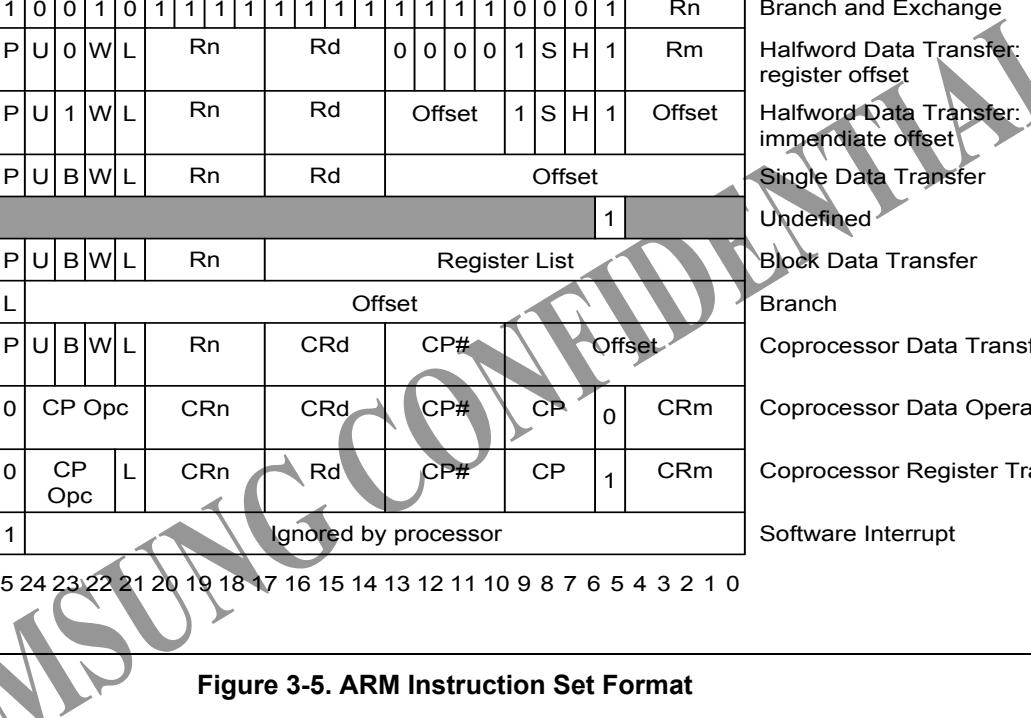


Figure 3-4. LDM Dependent Interlock Timing

FORMAT SUMMARY

The ARM instruction set formats are shown below.

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

INSTRUCTION SUMMARY

Table 3-4. The ARM Instruction Set

Mnemonic	Instruction	Action
ADC	Add with carry	Rd: = Rn + Op2 + Carry
ADD	Add	Rd: = Rn + Op2
AND	ANDo	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
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 meets 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: they are listed in Table 3-5. 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 3-5. 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 executed only if the condition is true. The various conditions are defined in Table 3-5.

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 would be decoded as ARM or THUMB instructions.

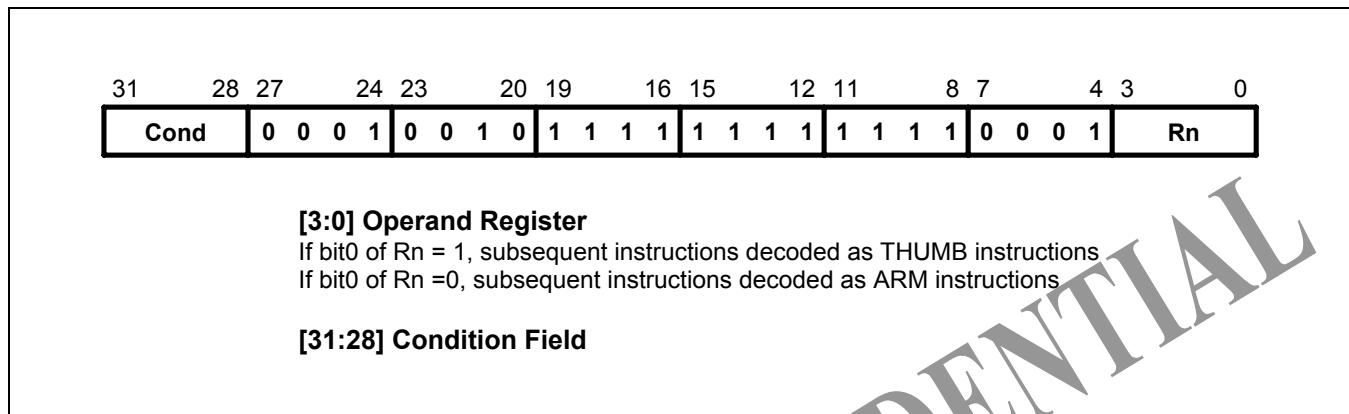


Figure 3-6. Branch and Exchange Instructions

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 3-5.

Rn ; is an expression evaluating to a valid register number.

USING R15 AS AN OPERAND

If R15 is used as an operand, its behaviour 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
Into_THUMB
.
.
.
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
.
.
.
ALIGN
CODE32
Back_to_ARM
```

BRANCH AND BRANCH WITH LINK (B, BL)

The instruction is executed only if the condition is true. The various conditions are defined Table 3-5. The instruction encoding is shown in Figure 3-7, below.

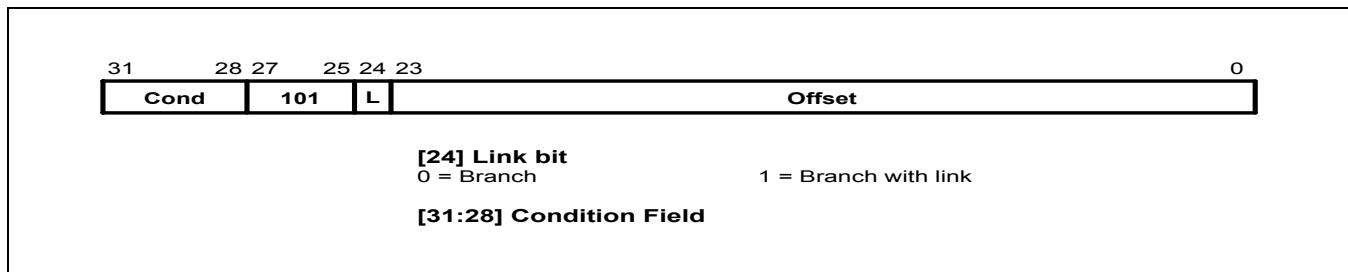


Figure 3-7. Branch Instructions

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 was 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 it 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 3-5. If absent, AL (ALways) will be used.

<expression> The destination. The assembler calculates the offset.

Examples

here	BAL	here	; Assembles to 0xEFFFFFFE (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 is held 0xFFFFFFFF.

DATA PROCESSING

The data processing instruction is executed only if the condition is true. The conditions are defined in Table 3-5. The instruction encoding is shown in Figure 3-8.

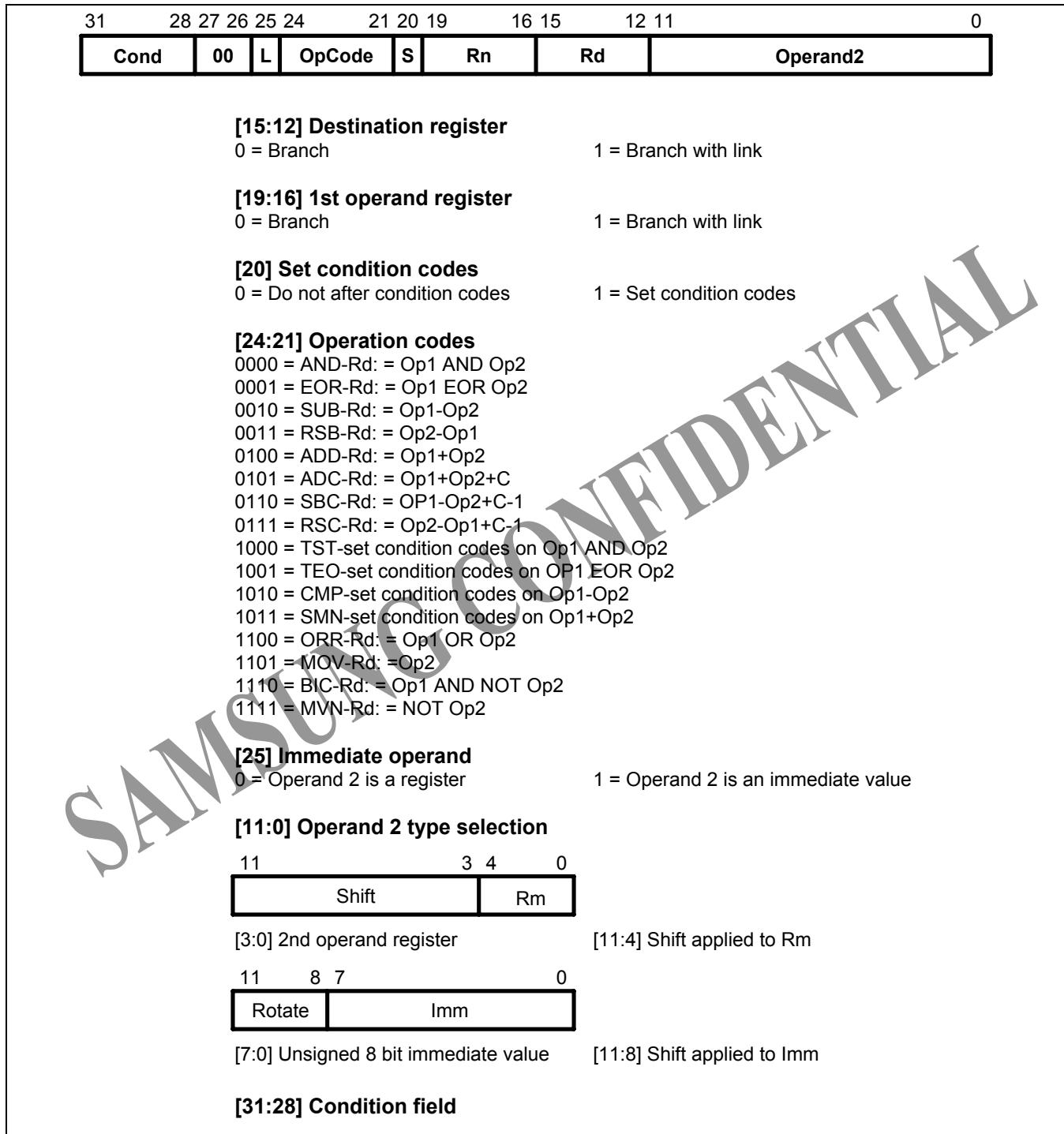


Figure 3-8. Data Processing Instructions

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 maintained 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 3-6.

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 maintained 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 3-6. 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). When the S bit is set (and Rd is not R15), the V flag in the CPSR will be set if an overflow occurs in bit 31 of the result; this may be ignored if the operands were considered unsigned, but there will be a warning 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 3-9.

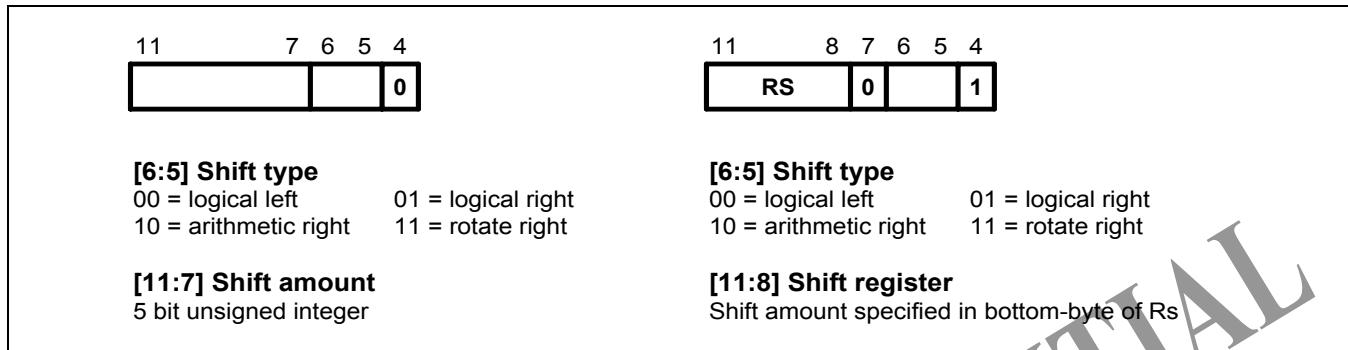


Figure 3-9. ARM Shift Operations

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. The least significant bit discarded 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 an example, the effect of LSL #5 is shown in Figure 3-10.

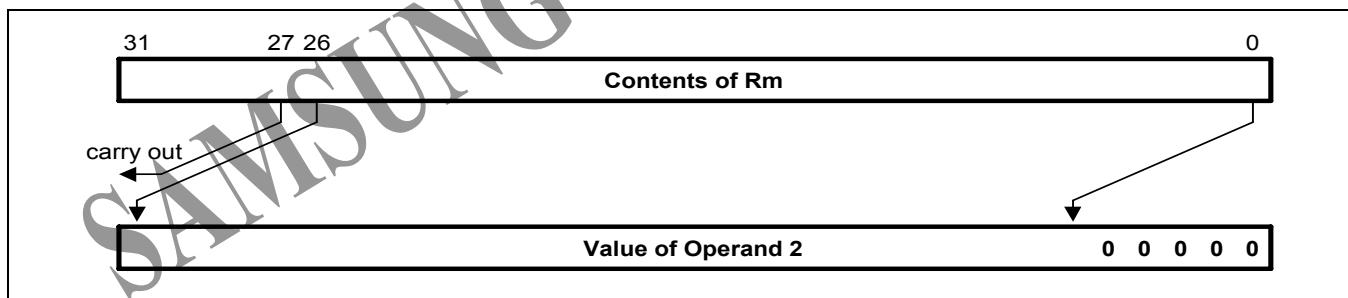


Figure 3-10. Logical Shift Left

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 to this except that the contents of Rm are moved to less significant positions in the result. LSR #5 has the effect shown in Figure 3-11.

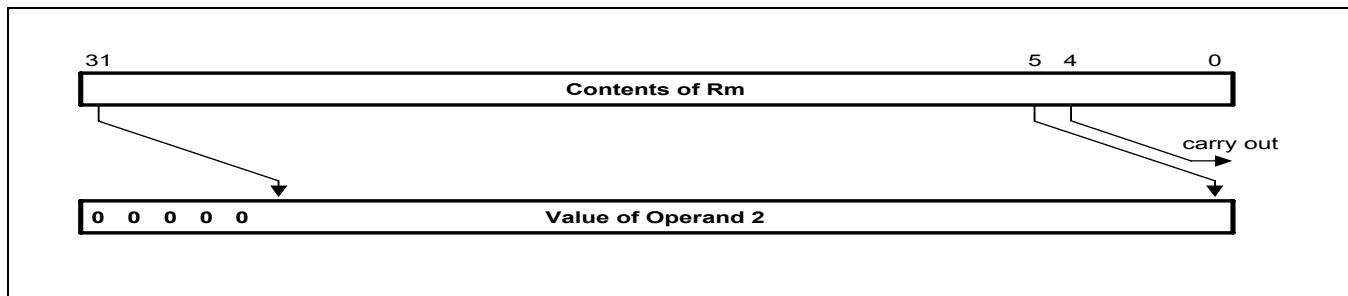


Figure 3-11. Logical Shift Right

The form of the shift field 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 contains the sign in 2's complement notation. For an example, ASR #5 is shown in Figure 3-12.

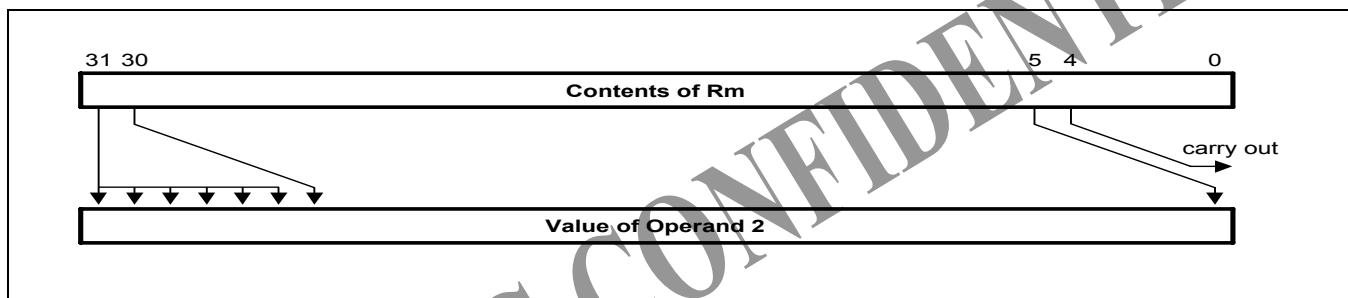


Figure 3-12. Arithmetic Shift Right

The form of the shift field 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 an example, ROR #5 is shown in Figure 3-13.

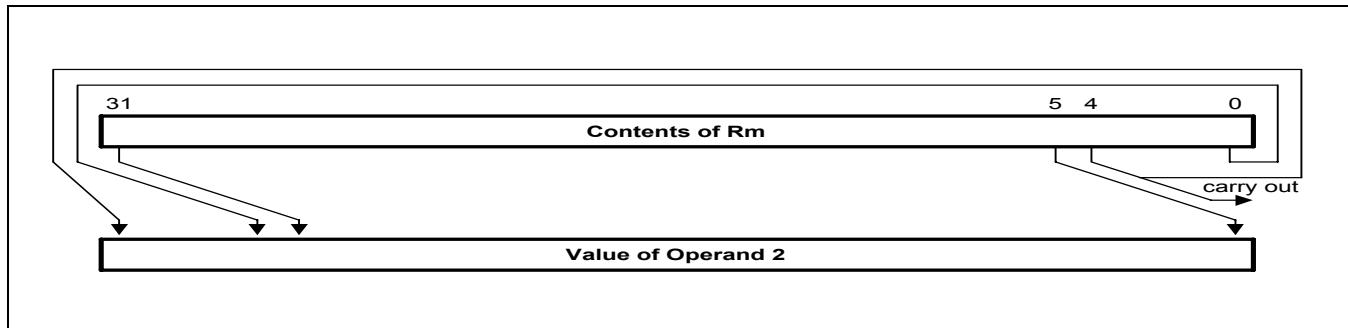


Figure 3-13. Rotate Right

The form of the shift field expected to give ROR #0 is used to encode a special function of the barrel shifter, rotate right extended (RRX). It enables rotating 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 3-14.

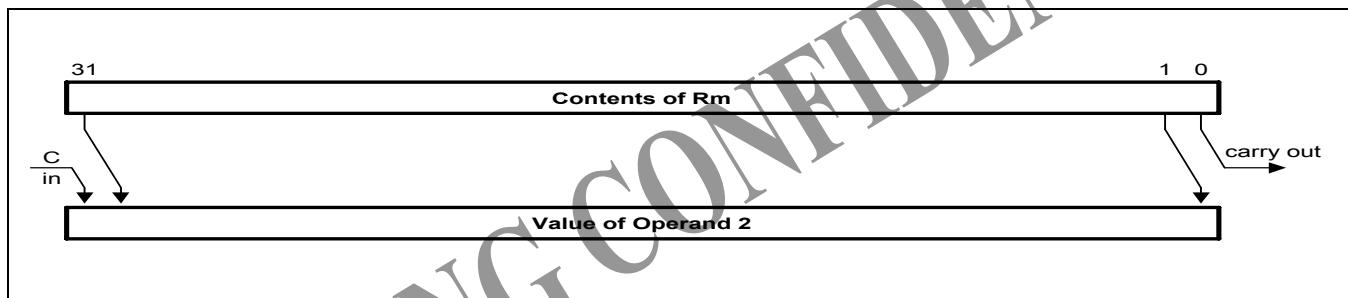


Figure 3-14. Rotate Right Extended

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, 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 OPERAND

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 TEQP in the ARM9TDMI moves SPSR_<mode> to the CPSR if the processor is in a privileged mode and does nothing in User mode.

INSTRUCTION CYCLE TIMES

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

Table 3-7. 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 mean sequential (S-cycle), non-sequential (N-cycle), and internal (I-cycle), respectively.

ASSEMBLER SYNTAX

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

where:

$\langle\text{Op2}\rangle$ $\text{Rm}\{\langle\text{shift}\rangle\}$ or, $\langle\#\text{expression}\rangle$

$\{\text{cond}\}$ A two-character condition mnemonic. See Table 3-5.

$\{\text{S}\}$ Set condition codes if S is present (implied for CMP, CMN, TEQ, TST).

Rd, Rn and Rm Expressions evaluating to a register number.

$\langle\#\text{expression}\rangle$ 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.

$\langle\text{shift}\rangle$ $\langle\text{Shiftname}\rangle \langle\text{register}\rangle$ or $\langle\text{Shiftname}\rangle \#expression$, or RRX (rotate right one bit with extend).

$\langle\text{shiftname}\rangle\{\text{s}$ ASL, LSL, LSR, ASR, ROR. (ASL is a synonym for LSL, they assemble to the same code.)

Examples

ADDEQ R2,R4,R5
 TEQS R4,#3

; If the Z flag is set, make R2:=R4+R5

; Test R4 for equality with 3.

; (The S is in fact redundant as the assembler inserts it automatically.)

; Logical right shift R7 by the number in

SUB R4,R5,R7,LSR R2

; the bottom byte of R2, subtract result from R5, and put the answer into R4.

; Return from subroutine.

; Return from exception and restore CPSR from SPSR_mode.

MOV PC,R14
 MOVS PC,R14

PSR TRANSFER (MRS, MSR)

The instruction is executed only if the condition is true. The various conditions are defined in Table 3-5.

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 3-15.

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 values 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 a 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, otherwise the processor would 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)

31	28 27	23 22 21	16 15	12 11	0
Cond	00010	Ps	001111	Rd	00000000000000

[15:21] Destination Register**[19:16] Source PSR**

0 = CPSR 1 = SPSR_<current mode>

[31:28] Condition Field**MRS (transfer register contents to PSR)**

31	28 27	23 22 21	12 11	4 3	0
Cond	00010	Pd	101001111	00000000	Rm

[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)**

31	28 27	26 25	24 23	22 21	12 11	0
Cond	00	I	10	Pd	101001111	Source operand

[22] Destination PSR

0 = CPSR 1 = SPSR_<current mode>

[25] Immediate Operand0 = Source operand is a register
1 = SPSR_<current mode>**[11:0] Source Operand**

11	4 3	0
00000000	Rm	

[3:0] Source Register

[11:4] Source operand is an immediate value

11	8 7	0
Rotate	Imm	

[7:0] Unsigned 8 bit immediate value

[11:8] Shift applied to Imm

[31:28] Condition Field**Figure 3-15. PSR Transfer**

RESERVED BITS

Only twelve bits of the PSR are defined in ARM9TDMI (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 in Chapter 2 *Programmer's model* for a full description of the PSR bits.

To ensure the maximum compatibility between ARM9TDMI 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 symbolize 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 3-5..
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 ous with 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 executed only if the condition is true. The various conditions are defined in Table 3-5. The instruction encoding is shown in Figure 3-16.

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

31	28 27	22 21 20 19	16 15	12 11	8 7	4 3	0
Cond	0 0 0 0 0 0	A S Rd	Rn	Rs	1 0 0 1	Rm	

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

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

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

[31:28] Condition Field

Figure 3-16. 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	0xFFFFFFF38

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 0xFFFFFFF38.

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 0xFFFFFFF38.

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.

SAMSUNG CONFIDENTIAL

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 $1S + ml$ and MLA $1S + (m+1)l$ cycles to execute, where S and l are defined as sequential (S-cycle) and internal (l-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. Available 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 3-5..

{S} Set condition codes if S is 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 executed only if the condition is true. The various conditions are defined in Table 3-5. The instruction encoding is shown in Figure 3-17.

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.

31	28 27	23 22 21 20 19	16 15	12 11	8 7	4 3	0
Cond	0 0 0 0 1	U A S	RdHi	RdLo	Rs	1 0 0 1	Rm
[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 3-17. 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 2's-complement signed numbers and write a 2'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.

Available 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 3-8. Assembler Syntax Descriptions

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

where:

{cond} Two-character condition mnemonic. See Table 3-5.

{S} Set condition codes if S is 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 executed only if the condition is true. The various conditions are defined in Table 3-5. The instruction encoding is shown in Figure 3-18. 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.

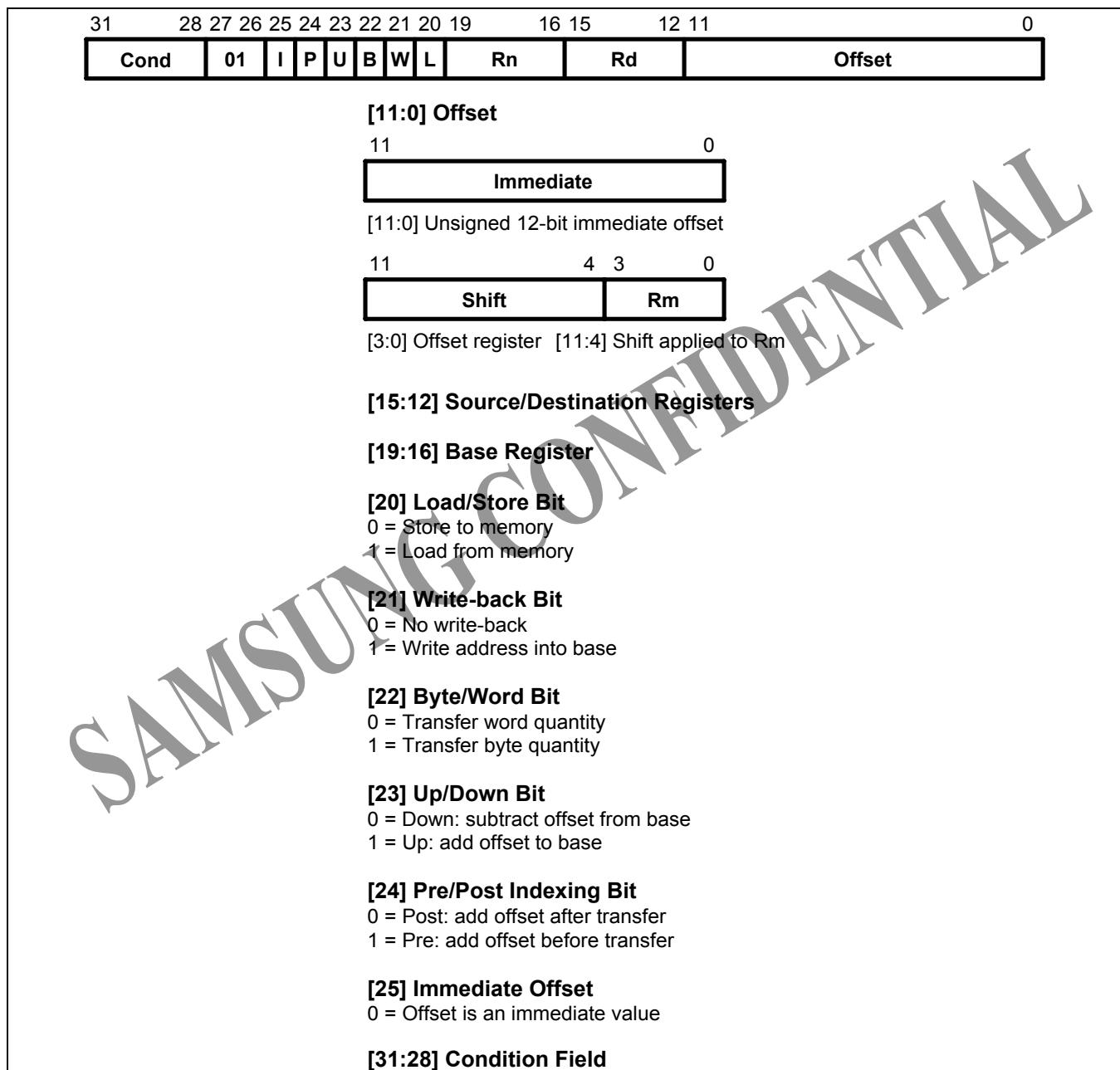


Figure 3-18. Single Data Transfer Instructions

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 3-6.

BYTES AND WORDS

This instruction class may be used to transfer a byte ($B=1$) or a word ($B=0$) between an ARM9TDMI register and memory.

The action of LDR(B) and STR(B) instructions is influenced by the **BIGEND** control signal of ARM9TDMI 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 in Chapter 2 *Programmer's model*.

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.

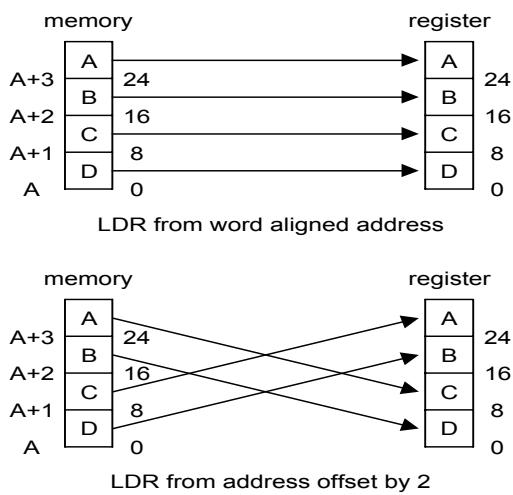


Figure 3-19. Little-Endian Offset Addressing

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 in Chapter 2 *Programmer's model*.

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 that 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

where:

LDR	Load from memory into a register
STR	Store from a register into memory
{cond}	Two-character condition mnemonic. See Table 3-5.
{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		

SAMSUNG CONFIDENTIAL

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

The instruction is executed only if the condition is true. The various conditions are defined in Table 3-5. The instruction encoding is shown in Figure 3-20.

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.

31	28	27	25	24	23	22	21	20	19	16	15	12	11	8	7	6	5	4	3	0
Cond	000	P	U	0	W	L	Rn		Rd	0000	1	S	H	1	Rm					

[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

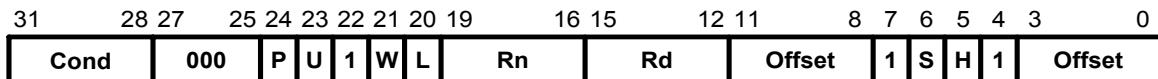
[21] Write-back
 0 = No write-back
 1 = Write address into base

[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 before transfer

[31:28] Condition Field

Figure 3-20. 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 before transfer

[31:28] Condition Field

Figure 3-21. Halfword and Signed Data Transfer with Immediate Offset and Auto-Indexing

OFFSETS AND AUTO-INDEXING

The offset from the base may be either an 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, so 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 ARM9TDMI 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.

ENDIANCE 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 bits 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 in Chapter 2 *Programmer's model*.

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, the ARM9TDMI 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 bits 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 in Chapter 2 *Programmer's model*.

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, the ARM9TDMI 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 that 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 continues.

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 3-5..
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 ARM9TDMI 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 R4+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 executed only if the condition is true. The various conditions are defined in Table 3-5. The instruction encoding is shown in Figure 3-22.

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 they 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.

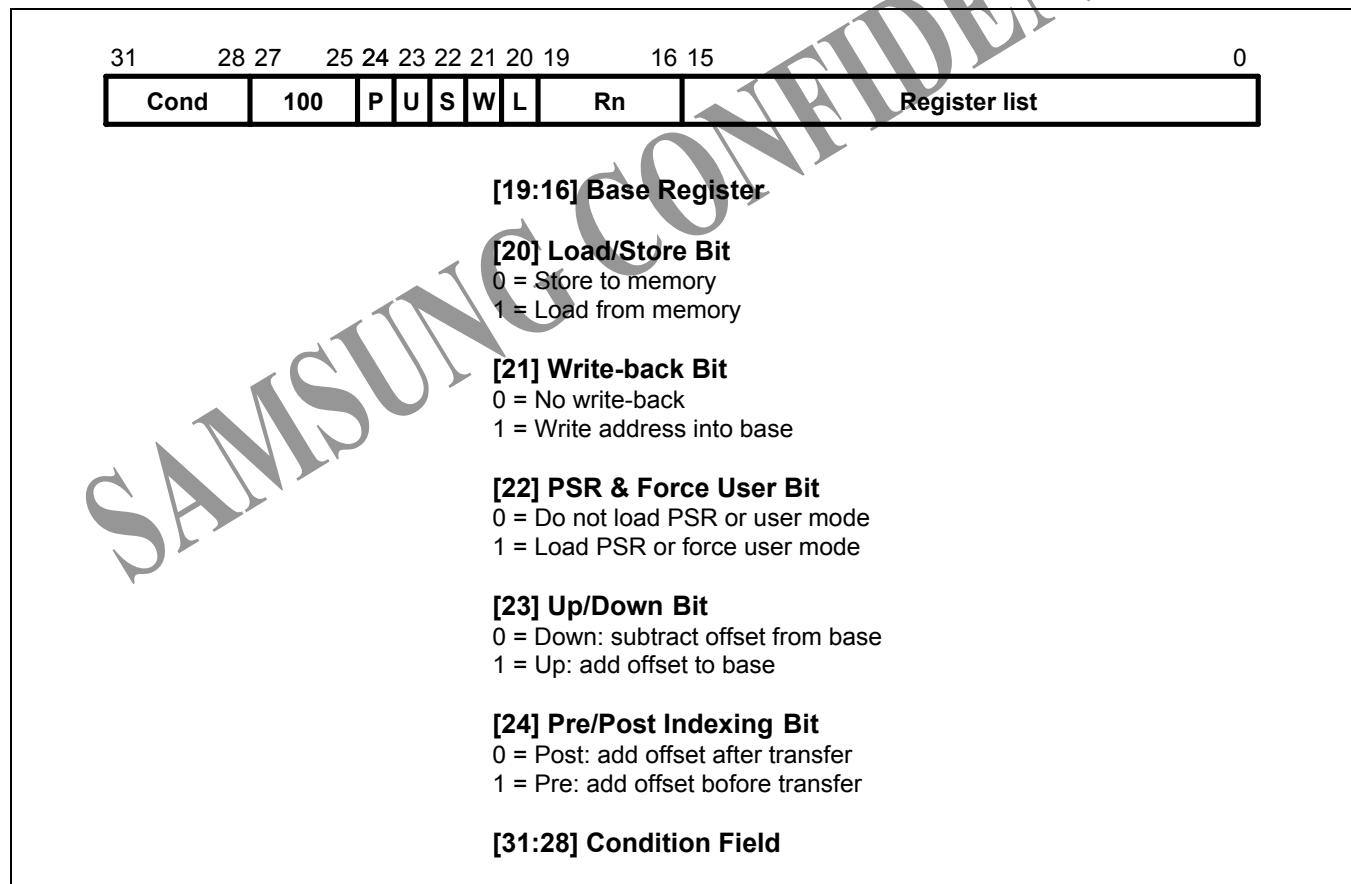


Figure 3-22. Block Data Transfer Instructions

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 3-23 ~ 3-26 shows 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.

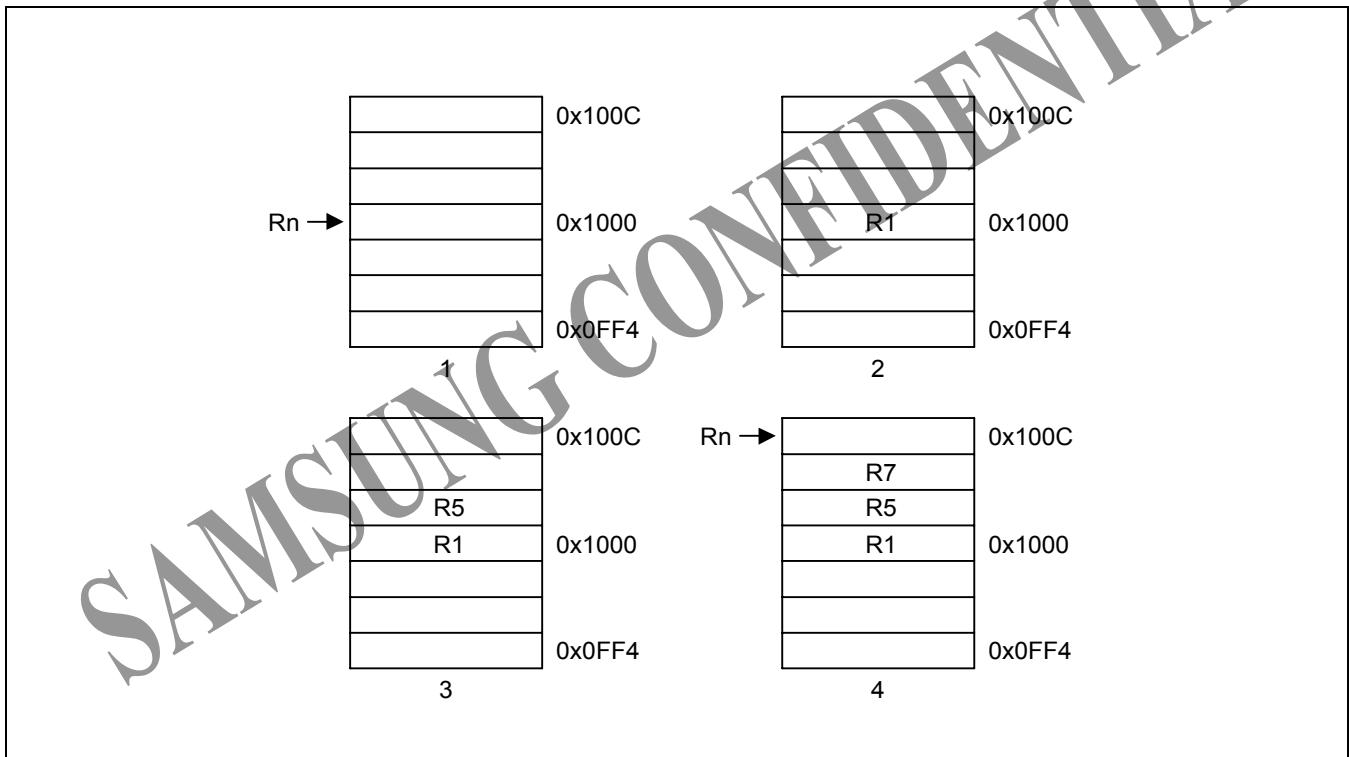


Figure 3-23. Post-Increment Addressing

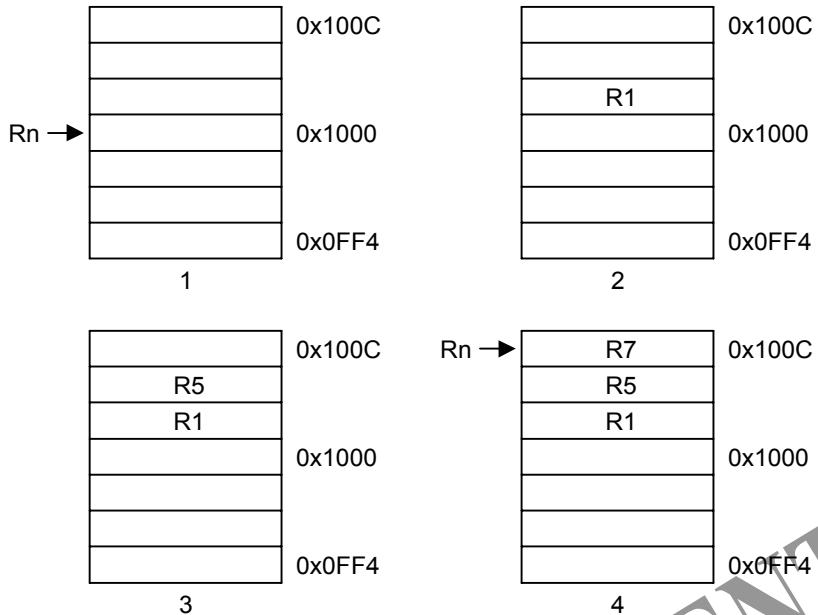


Figure 3-24. Pre-Increment Addressing

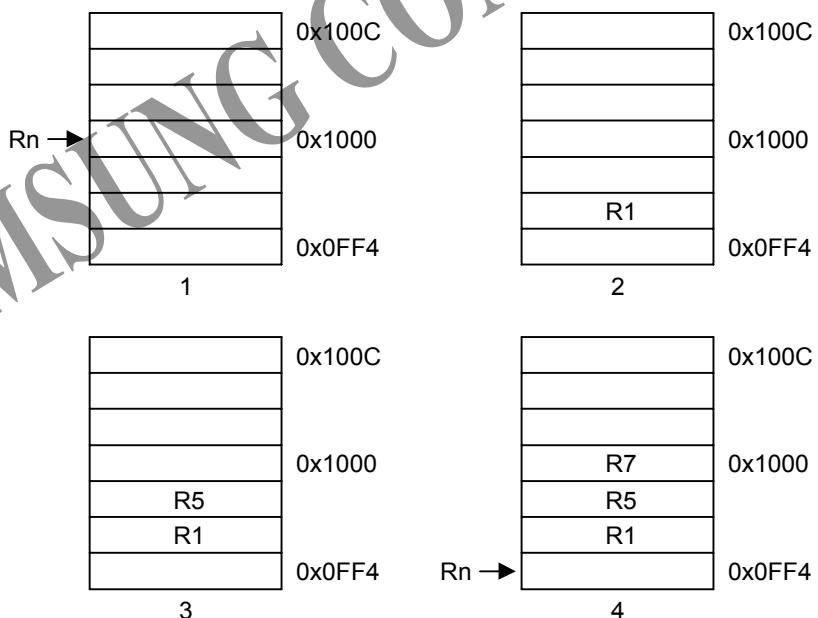


Figure 3-25. Post-Decrement Addressing

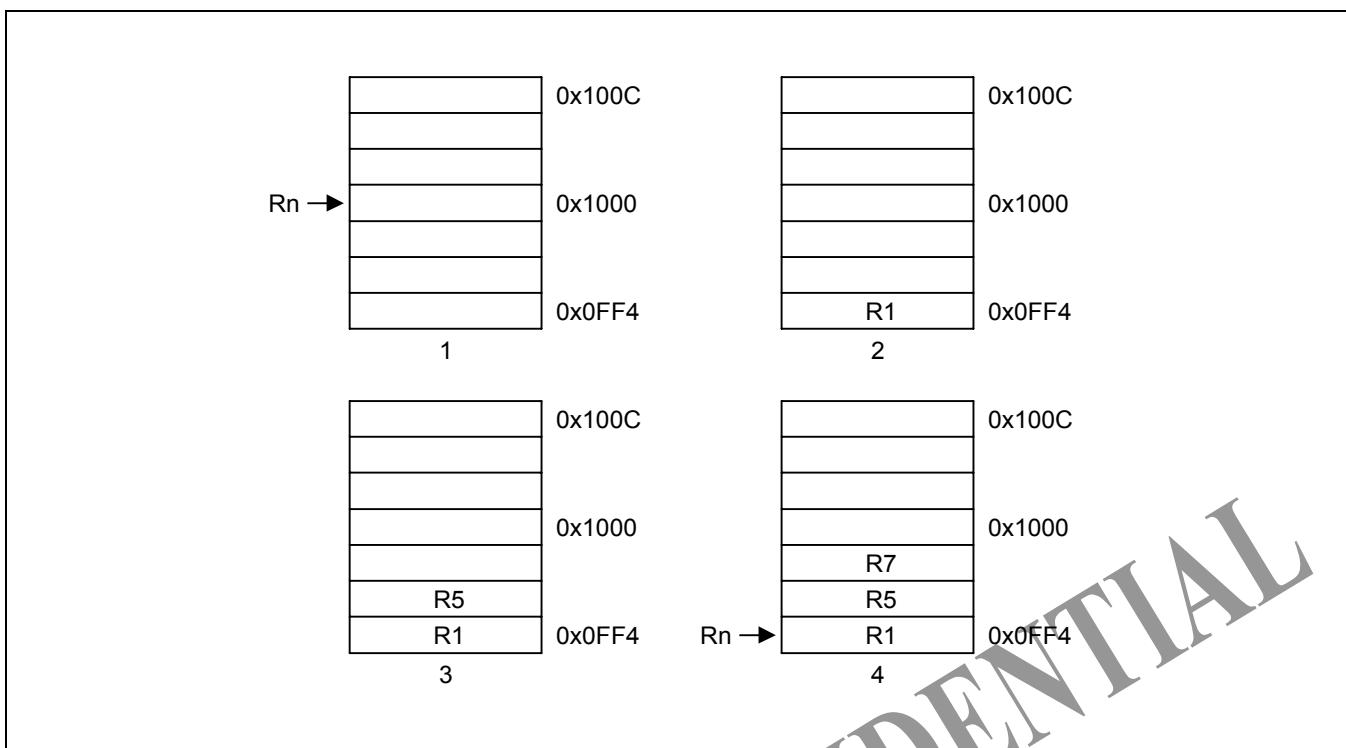


Figure 3-26. Pre-Decrement Addressing

USE OF THE S BIT

When the S bit is set in an 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 an LDM, 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 an 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 an STM, the first register is written out at the start of the second cycle. An 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. An 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 ARM9TDMI is to be used in a virtual memory system.

Abort during STM Instructions

If the abort occurs during a store multiple instruction, ARM9TDMI 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 ARM9TDMI 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 3-5.
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 3-9.

Table 3-9. 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-Increment Load	LDMEA	LDMDB	1	1	0
Post-Increment Load	LDMFA	LDMDA	1	0	0
Pre-Increment Load	STMFA	STMIB	0	1	1
Post-Increment Load	STMEA	STMIA	0	0	1
Pre-Increment Load	STMFD	STMDB	0	1	0
Post-Increment Load	STMED	STMDA	0	0	0

FD, ED, FA, and EA define pre/post indexing and the up/down bit by referencing to the form of the 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, an STM will go up and an LDM down, if descending, vice versa.

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

Examples

LDMFD	SPI!,{R0,R1,R2}	; Unstack 3 registers.
STMIA	R0,{R0-R15}	; Save all registers.
LDMFD	SPI!,{R15}	; R15 ← (SP), CPSR unchanged.
LDMFD	SPI!,{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	SPI!,{R0-R3,R14}	; Save R0 to R3 to use as workspace ; and R14 for returning.
BL	somewhere	; This nested call will overwrite R14
LDMED	SPI!,{R0-R3,R15}	; Restore workspace and return.

SINGLE DATA SWAP (SWP)

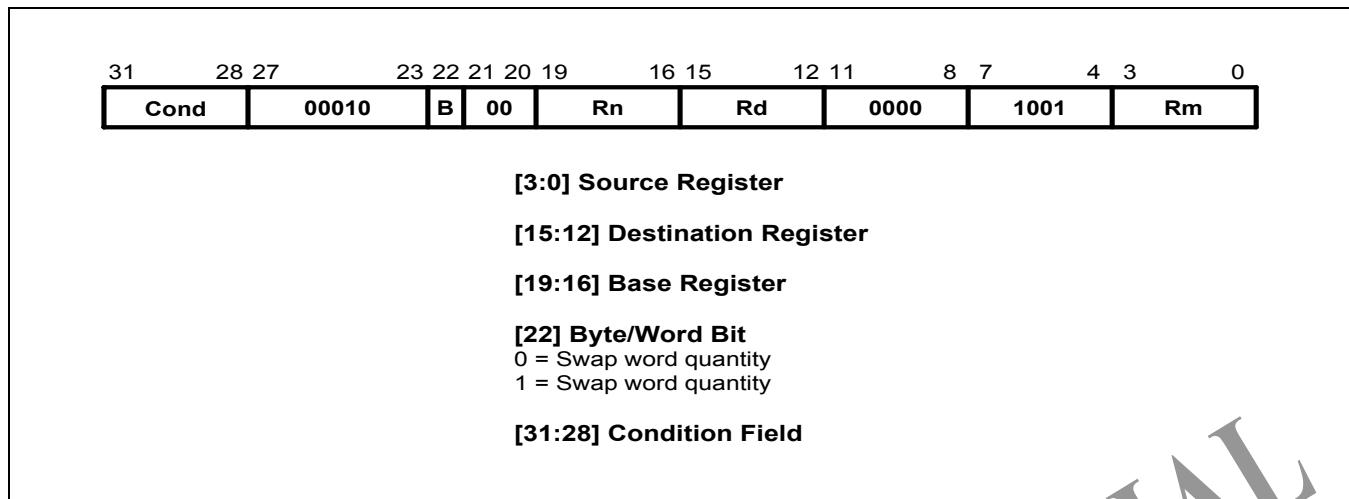


Figure 3-27. Swap Instruction

The instruction is executed only if the condition is true. The various conditions are defined in Table 3-5. The instruction encoding is shown in Figure 3-27.

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 are 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 ARM9TDMI register and memory. The SWP instruction is implemented as a LDR followed by an 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 an 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 continues.

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 3-5.
{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 executed only if the condition is true. The various conditions are defined in Table 3-5. The instruction encoding is shown in Figure 3-28 below.

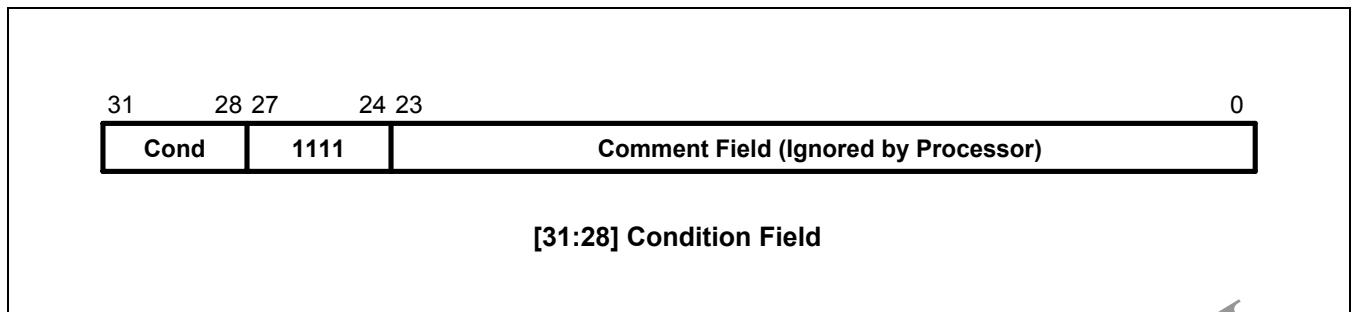


Figure 3-28. Software Interrupt Instruction

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 3-5.
- <expression> Evaluated and placed in the comment field (which is ignored by ARM9TDMI).

Examples

SWI	ReadC	; Get next character from read stream.
SWI	Writel+"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		
Writel EQU 512		
Supervisor		; SWI has routine required in bits 8-23 and data (if any) in bits 0-7. Assume 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 executed only if the condition is true. The various conditions are defined in Table 3-5. The instruction encoding is shown in Figure 3-29.

This class of instruction is used to tell a coprocessor to perform some internal operation. No result is communicated back to ARM9TDMI, 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 ARM9TDMI to perform independent tasks in parallel.

COPROCESSOR INSTRUCTIONS

The S5L8700X, unlike some other ARM-based processors, does not have an external coprocessor interface. It does not have an on-chip coprocessor also.

So then all coprocessor instructions will cause the undefined instruction trap to be taken on the S5L8700X. These coprocessor instructions can be emulated by the undefined trap handler. Even though an external coprocessor can not be connected to the S5L8700X, the coprocessor instructions are still described here in full for completeness. (Remember that any external coprocessor described in this section is a software emulation.)

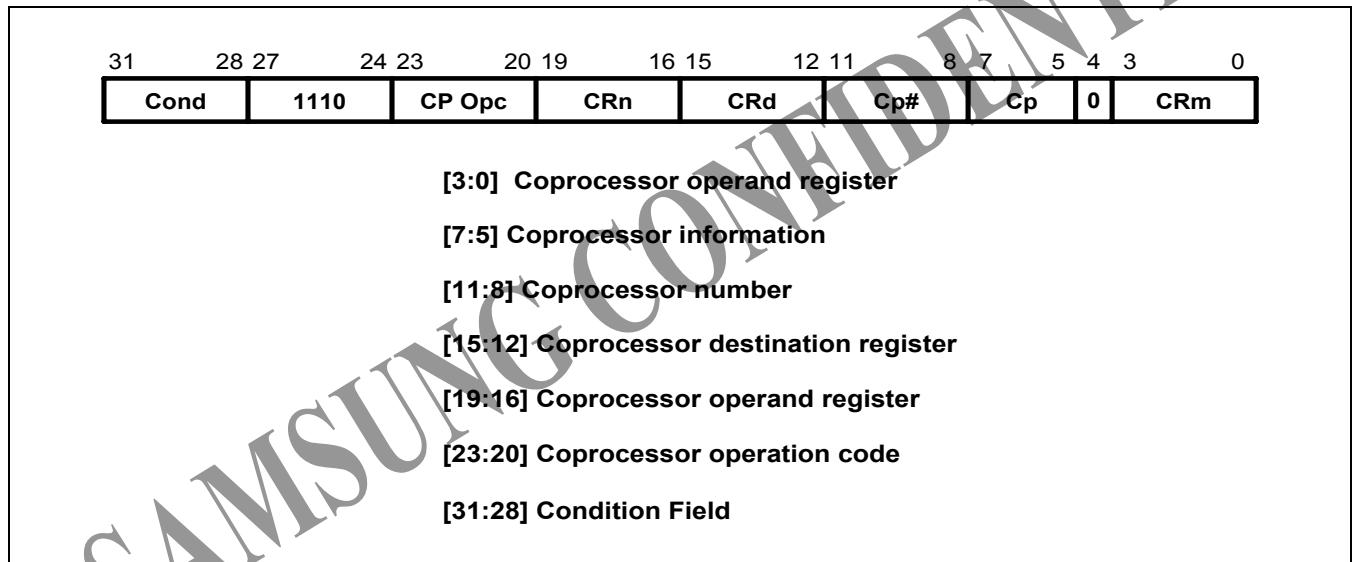


Figure 3-29. Coprocessor Data Operation Instruction

THE COPROCESSOR FIELDS

Only bit 4 and bits 24 to 31 are significant to ARM9TDMI. 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 as appropriate except CP#. 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 3-5.
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 executed only if the condition is true. The various conditions are defined in Table 3-5. The instruction encoding is shown in Figure 3-30.

This class of instruction is used to load (LDC) or store (STC) a subset of a coprocessor's registers directly to memory. ARM9TDMI is responsible for supplying the memory address, and the coprocessor supplies or accepts the data and controls the number of words transferred.

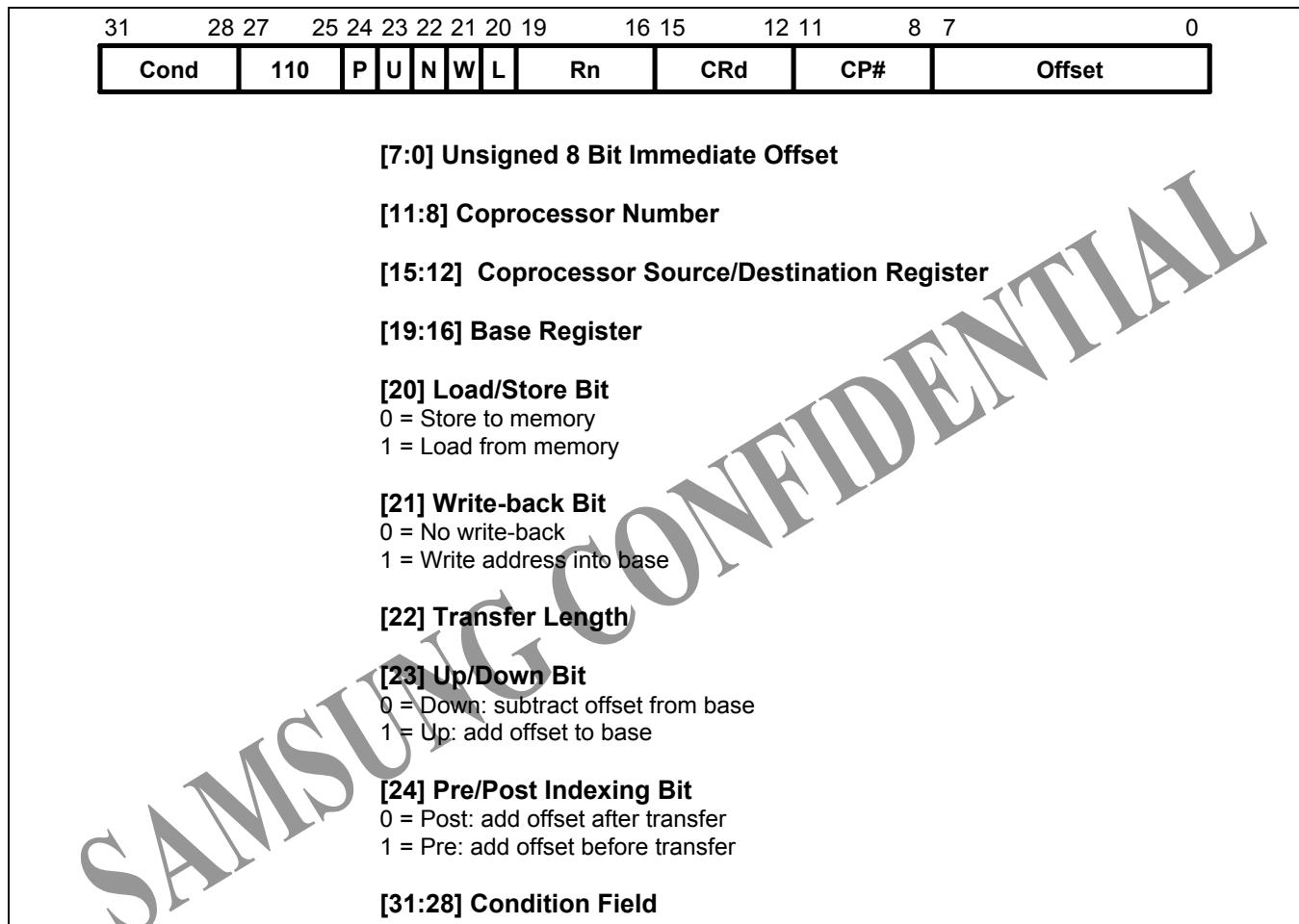


Figure 3-30. Coprocessor Data Transfer Instructions

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 respond only 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

ARM9TDMI 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 is resolved, and must ensure that any subsequent action 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 3-5..

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 ARM9TDMI register number.

NOTE

If Rn is R15, the assembler will subtract 8 from the offset value to allow for ARM9TDMI 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 executed only if the condition is true. The various conditions are defined in Table 3-5. The instruction encoding is shown in Figure 3-31.

This class of instruction is used to communicate information directly between ARM9TDMI and a coprocessor. An example of a coprocessor to ARM9TDMI 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 ARM9TDMI register. A FLOAT of a 32-bit value in ARM9TDMI register into a floating point value within the coprocessor illustrates the use of ARM9TDMI register to coprocessor transfer (MCR).

An important use of this instruction is to communicate control information directly from the coprocessor into the ARM9TDMI CPSR flags. As an example, the result of a comparison of two floating point values within a coprocessor can be moved to the CPSR to control the subsequent flow of execution.

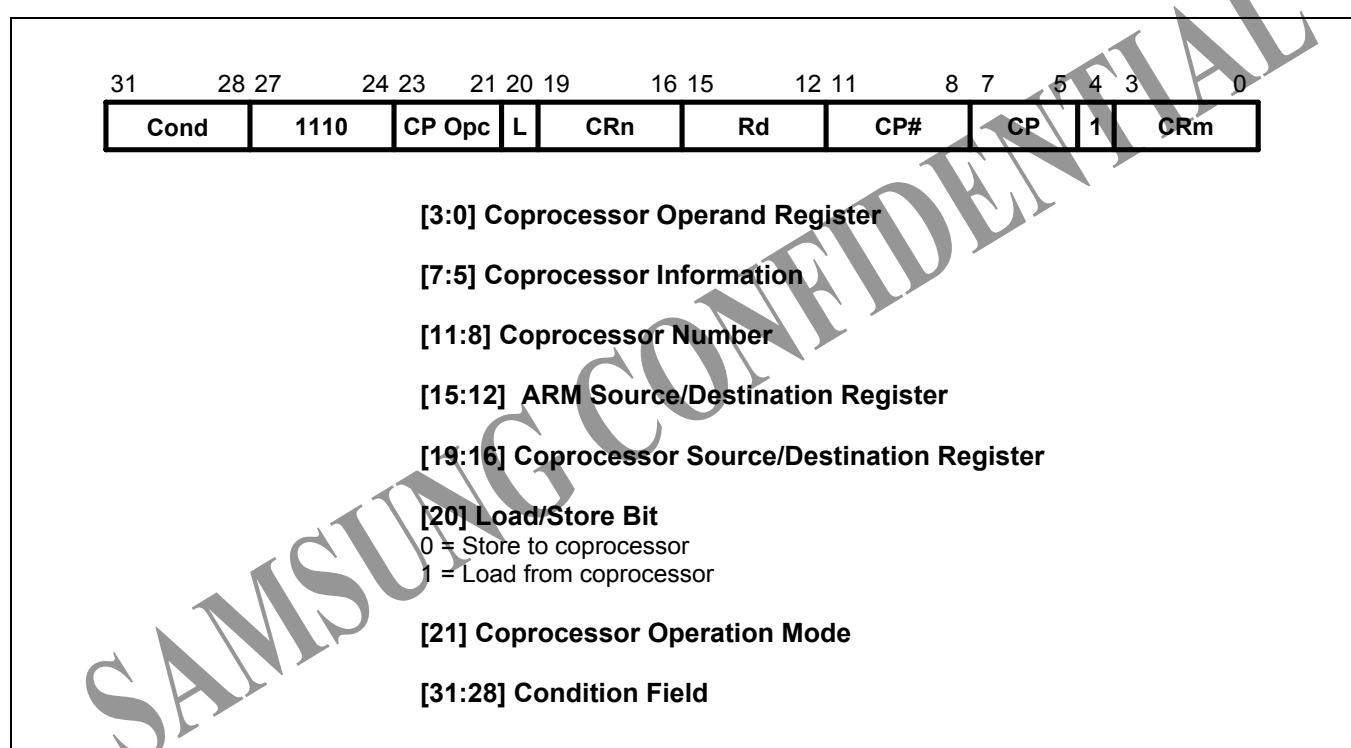


Figure 3-31. Coprocessor Register Transfer Instructions

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 ARM9TDMI 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 ARM9TDMI 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 ARM9TDMI register (L=1)
MCR	Move from ARM9TDMI register to coprocessor (L=0)
{cond}	Two-character condition mnemonic. See Table 3-5
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 ARM9TDMI 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 executed only if the condition is true. The various conditions are defined in Table 3-5. The instruction format is shown in Figure 3-32.

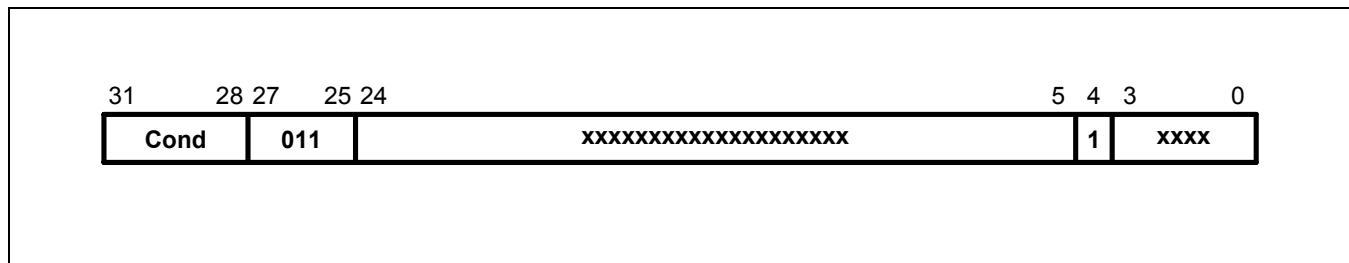


Figure 3-32. Undefined Instruction

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 then, this instruction must not be used.

INSTRUCTION SET EXAMPLES

The following examples show ways in which the basic ARM9TDMI instructions can combine to give efficient codes. None of these methods saves a great deal of execution time (although they may save some), mostly they just save codes.

Using the conditional instructions

Using Conditional instruction 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	;	If condition not satisfied try other test.
CMPNE	Rm,#q		
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 is as follows.

	MOV	Rcnt,#1	; Enter with numbers in Ra and Rb.
Div1	CMP	Rb,#0x80000000	; Bit to control the division.
	CMPCC	Rb,Ra	; Move Rb until greater than 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 ARM9TDMI

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	RI,Rh,Rm,Rn	; 3 to 6 cycles
ADDS	RI,RI,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	RI,Rh,Rm,Rn	; 3 to 6 cycles
ADDS	RI,RI,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 an 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 or 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:

TST	Rb,Rb,LSR#1	; Enter with seed in Ra (32-bits),
MOVS	Rc,Ra,RRX	; Rb (1 bit in Rb lsb), uses Rc.
ADC	Rb,Rb,Rb	; Top bit into carry
EOR	Rc,Rc,Ra,LSL#12	; 33-bit rotate right
		; Carry into lsb of Rb
EOR	Ra,Rc,Rc,LSR#20	; (involved!)
		; (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 $C \text{ MOD } 4 = 1$, 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 $C \text{ MOD } 4 = 3$, 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

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

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 ARM9TDMI 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												
2	0	0	0	1	1	I	Op	Rn/offset3								
3	0	0	1	Op			Rd									Offset8
4	0	1	0	0	0	0	Op									
5	0	1	0	0	0	1	Op	H1	H2		Rs/Hs					Rd/Hd
6	0	1	0	0	1	Rd										Word8
7	0	1	0	1	L	B	0	Ro		Rb						Rd
8	0	1	0	1	H	S	1	Ro		Rb						PC-relative load
9	0	1	1	B	L		Offset5			Rb						Load/store with register offset
10	1	0	0	0	L		Offset5			Rb						Load/store sign-extended byte/halfword
11	1	0	0	1	L	Rd										Load/store with immediate offset
12	1	0	1	0	SP	Rd										Load/store halfword
13	1	0	1	1	0	0	0	0	S							SP-relative load/store
14	1	0	1	1	L	1	0	R								Add offset to stack pointer
15	1	1	0	0	L	Rb										Push/pop register
16	1	1	0	1	Cond											Multiple load/store
17	1	1	0	1	1	1	1	1								Conditional branch
18	1	1	1	0	0											Software interrupt
19	1	1	1	1	H											Unconditional branch
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 3-33. 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 3-10. THUMB Instruction Set Opcodes

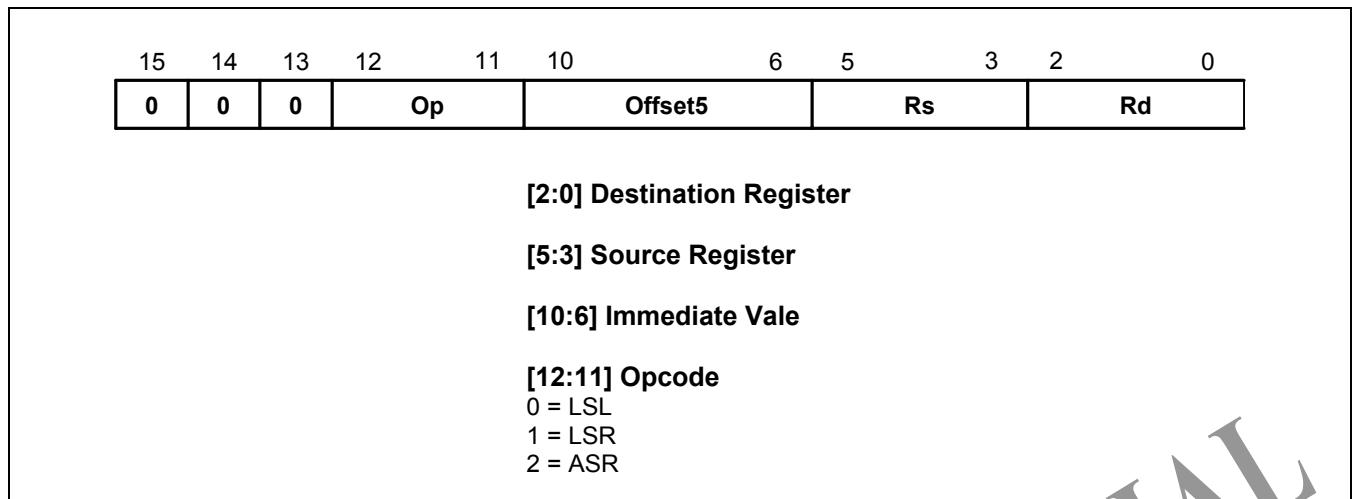
Mnemonic	Instruction	Lo-Register Operand	Hi-Register Operand	Condition Codes Set
ADC	Add with Carry	4	–	4
ADD	Add	4	–	4 (1)
AND	AND	4	–	4
ASR	Arithmetic Shift Right	4	–	4
B	Unconditional branch	4	–	–
Bxx	Conditional branch	4	–	–
BIC	Bit Clear	4	–	4
BL	Branch and Link	–	–	–
BX	Branch and Exchange	4	4	–
CMN	Compare Negative	4	–	4
CMP	Compare	4	4	4
EOR	EOR	4	–	4
LDMIA	Load multiple	4	–	–
LDR	Load word	4	–	–
LDRB	Load byte	4	–	–
LDRH	Load halfword	4	–	–
LSL	Logical Shift Left	4	–	4
LDSB	Load sign-extended byte	4	–	–
LDSH	Load sign-extended halfword	4	–	–
LSR	Logical Shift Right	4	–	4
MOV	Move register	4	4	4 (2)
MUL	Multiply	4	–	4
MVN	Move Negative register	4	–	4
ADC	Add with Carry	4	–	4
ADD	Add	4	–	4 (1)
AND	AND	4	–	4
ASR	Arithmetic Shift Right	4	–	4
B	Unconditional branch	4	–	–

Table 3-10. THUMB Instruction Set Opcodes (Continued)

Mnemonic	Instruction	Lo-Register Operand	Hi-Register Operand	Condition Codes Set
Bxx	Conditional branch	4	–	–
BIC	Bit Clear	4	–	4
BL	Branch and Link	–	–	–
BX	Branch and Exchange	4	4	–
CMN	Compare Negative	4	–	4
CMP	Compare	4	4	4
EOR	EOR	4	–	4
LDMIA	Load multiple	4	–	–

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**Figure 3-34. Format 1****OPERATION**

These instructions move a shifted value between Lo registers. The THUMB assembler syntax is shown in Table 3-11.

NOTE

All instructions in this group set the CPSR condition codes.

Table 3-11. 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 3-11. 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

15	14	13	12	11	10	9	8	6	5	3	2	0
0	0	0	1	1	I	Op	Rn/Offset3		Rs		Rd	

[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 3-35. 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 3-12.

NOTE

All instructions in this group set the CPSR condition codes.

Table 3-12. 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 3-12. 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

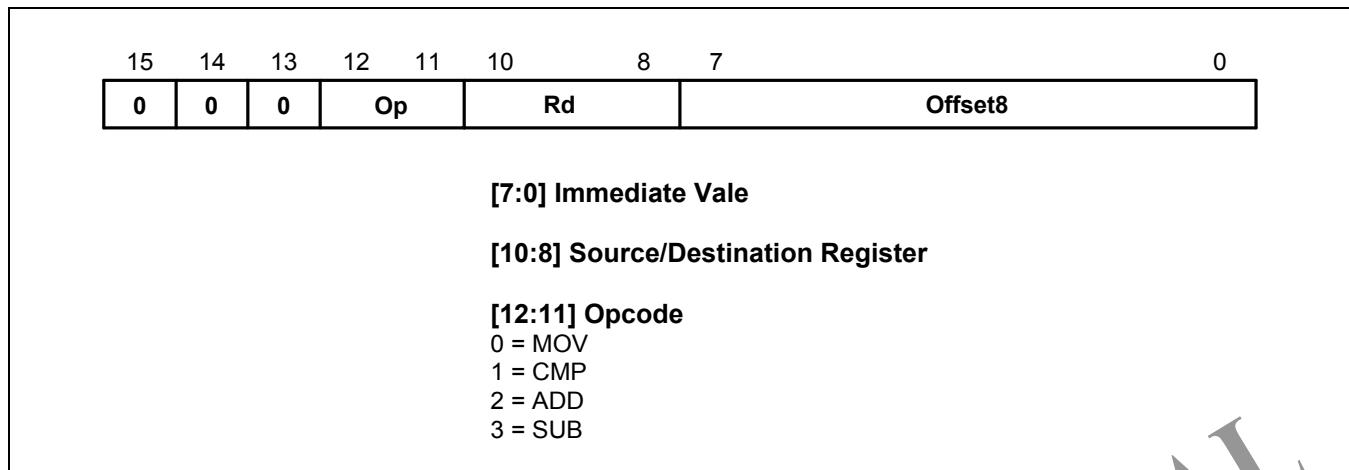


Figure 3-36. Format 3

OPERATION

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 3-13.

NOTE

All instructions in this group set the CPSR condition codes.

Table 3-13. 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 3-13. 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

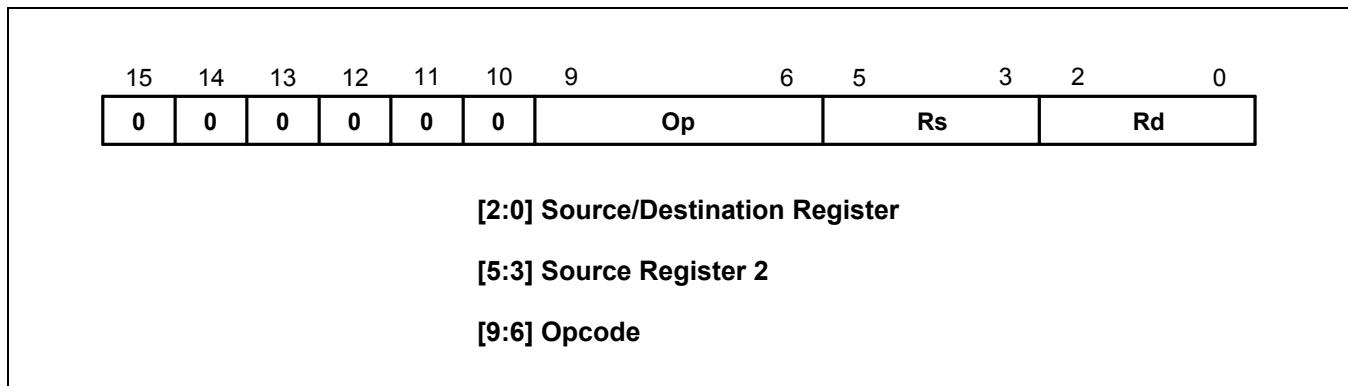


Figure 3-37. Format 4

OPERATION

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

NOTE

All instructions in this group set the CPSR condition codes.

Table 3-14. 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 3-14. 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

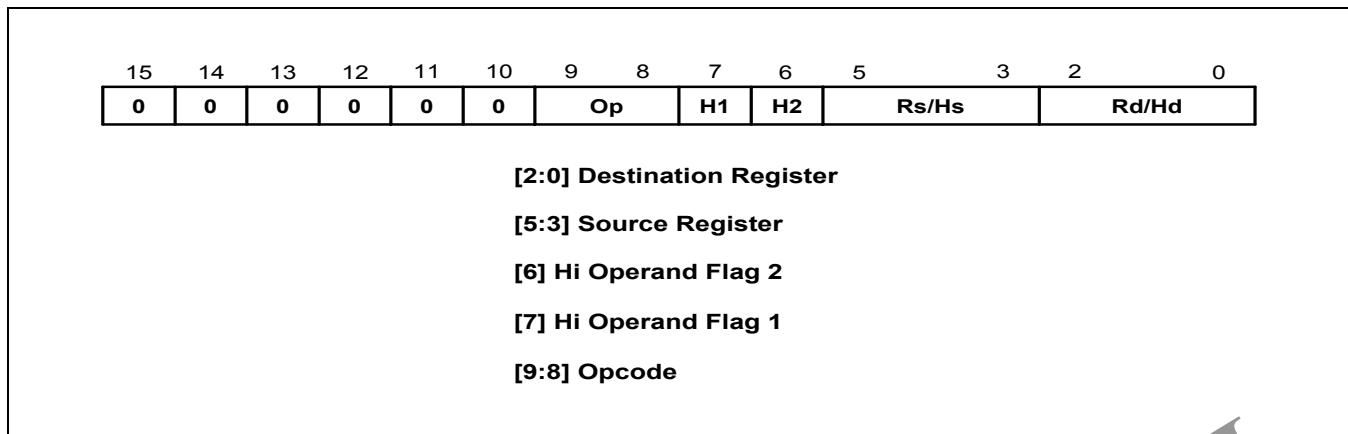


Figure 3-38. Format 5

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 3-15.

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 3-15. 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 3-15. 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 3-15. 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

ADR	R1,outofTHUMB	; Switch from THUMB to ARM state.
MOV	R11,R1	; Load address of outofTHUMB into 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

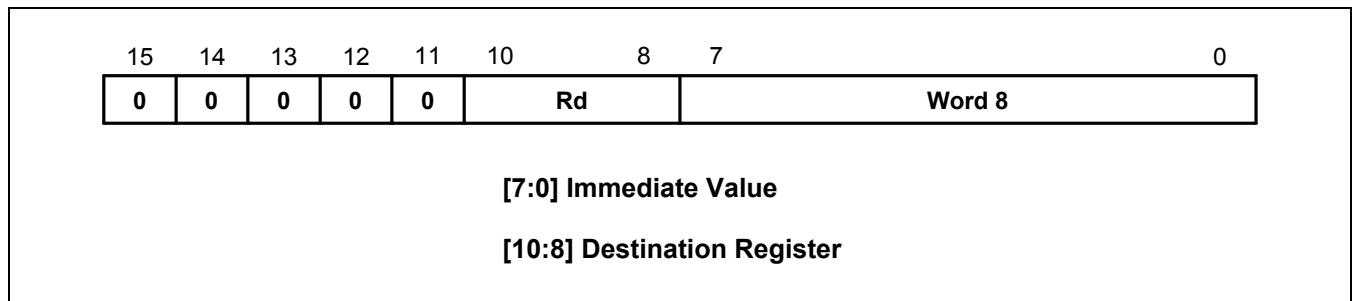


Figure 3-39. Format 6

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 3-16. 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.

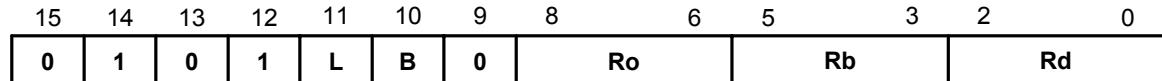
NOTE: The value specified by #Imm is a full 10-bit address, but must always be word-aligned (ie with bits 1:0 set to 0), since the assembler places #Imm >> 2 in field Word 8. The value of the PC will be 4 bytes greater than the address of this instruction, but bit 1 of the PC is forced to 0 to ensure 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 3-40. Format 7



ELECTRONICS

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 3-17.

Table 3-17. 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 3-17. 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.

```

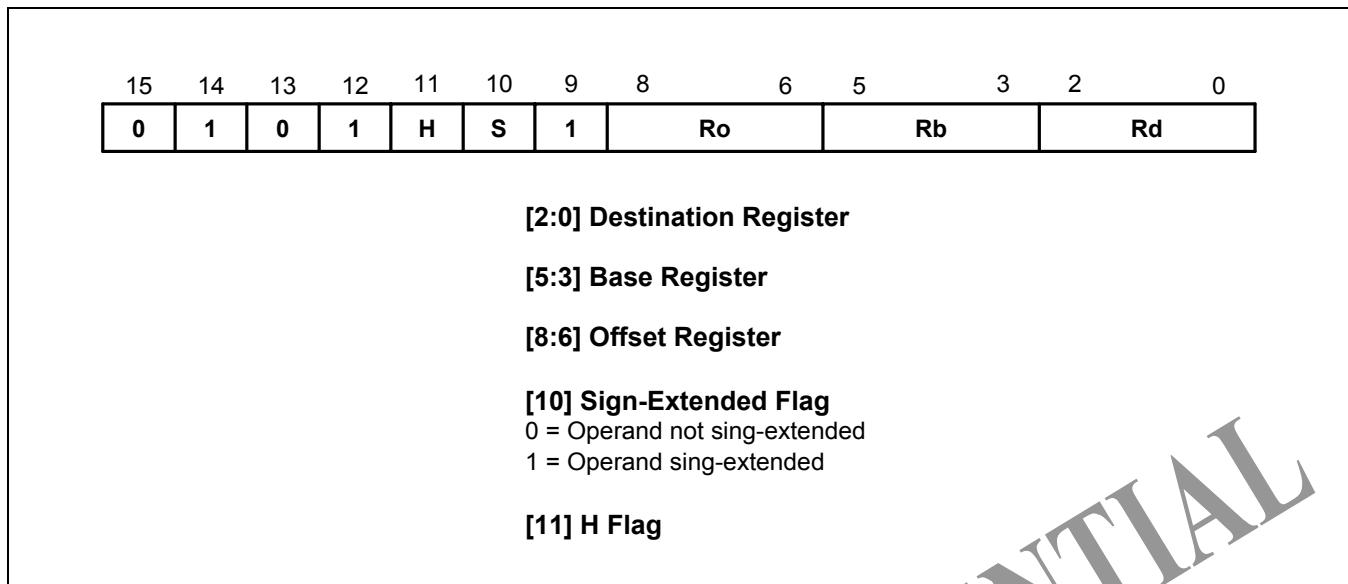
FORMAT 8: LOAD/STORE SIGN-EXTENDED BYTE/HALFWORD

Figure 3-41. Format 8

OPERATION

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

Table 3-18. Summary of format 8 instructions

S	H	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 3-18. 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

15	14	13	12	11	10	6	5	3	2	0
0	1	1	B	L	Offset5	Rb		Rd		

[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 Flag

0 = Transfer word quantity
1 = Transfer byte quantity

Figure 3-42. Format 9



ELECTRONICS

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 3-19.

Table 3-19. 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.

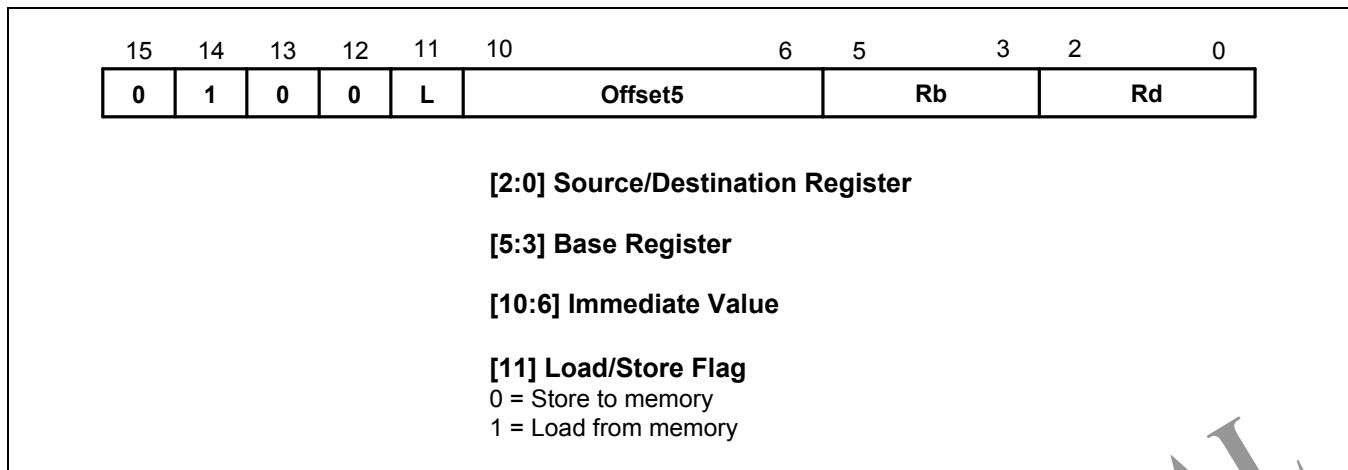
NOTE: For word accesses (B = 0), the value specified by #Imm is a full 7-bit address, but must be word-aligned (ie with bits 1:0 set to 0), since the assembler places #Imm >> 2 in the Offset5 field.

Instruction cycle times

All instructions in this format have an equivalent ARM instruction as shown in Table 3-19. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

Examples

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

FORMAT 10: LOAD/STORE HALFWORD**Figure 3-43. 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 3-20.

Table 3-20. 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 3-20. 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

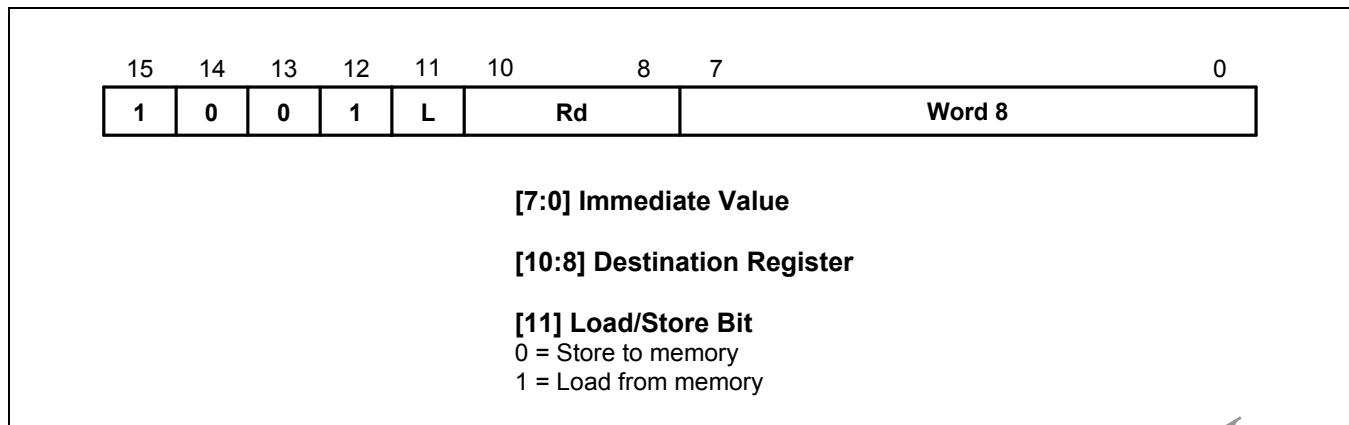


Figure 3-44. Format 11

OPERATION

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

Table 3-21. 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 3-21. 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

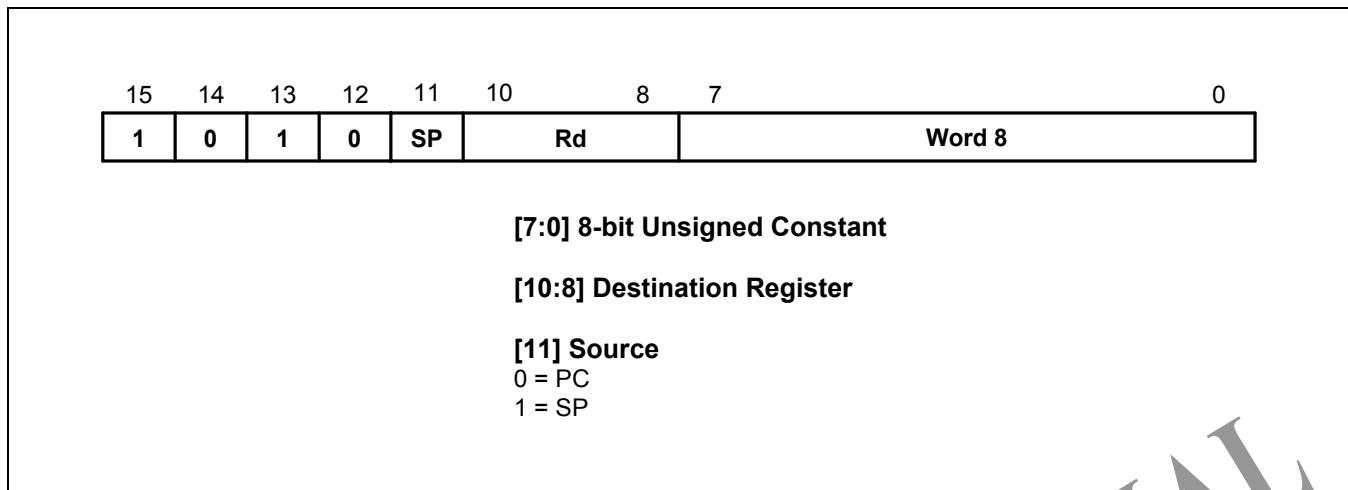


Figure 3-45. Format 12

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 3-22. Load Address

SP	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 3-22. 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.

SAMSUNG CONFIDENTIAL

FORMAT 13: ADD OFFSET TO STACK POINTER

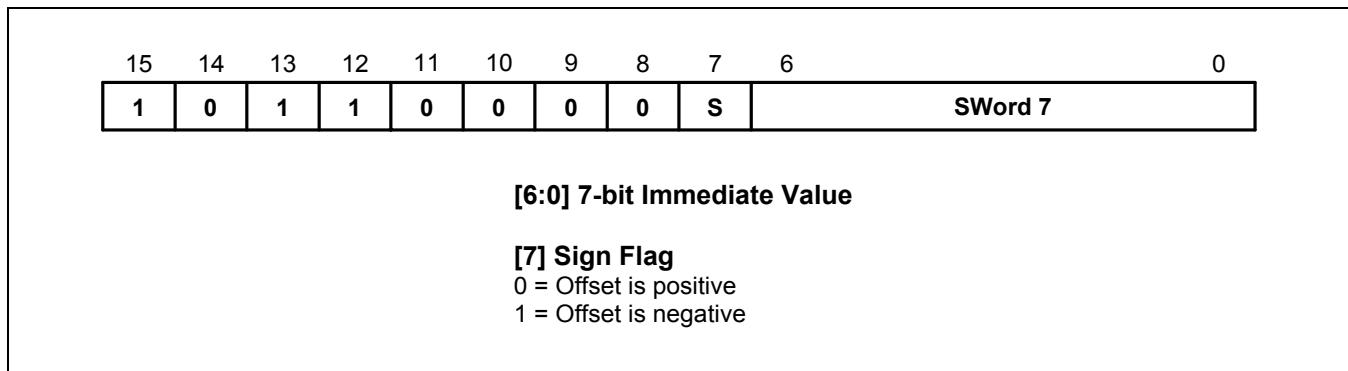


Figure 3-46. Format 13

OPERATION

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

Table 3-23. The ADD SP Instruction

S	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 3-23. The instruction cycle times for the THUMB instruction are identical to that of the equivalent ARM instruction.

Examples

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

FORMAT 14: PUSH/POP REGISTERS

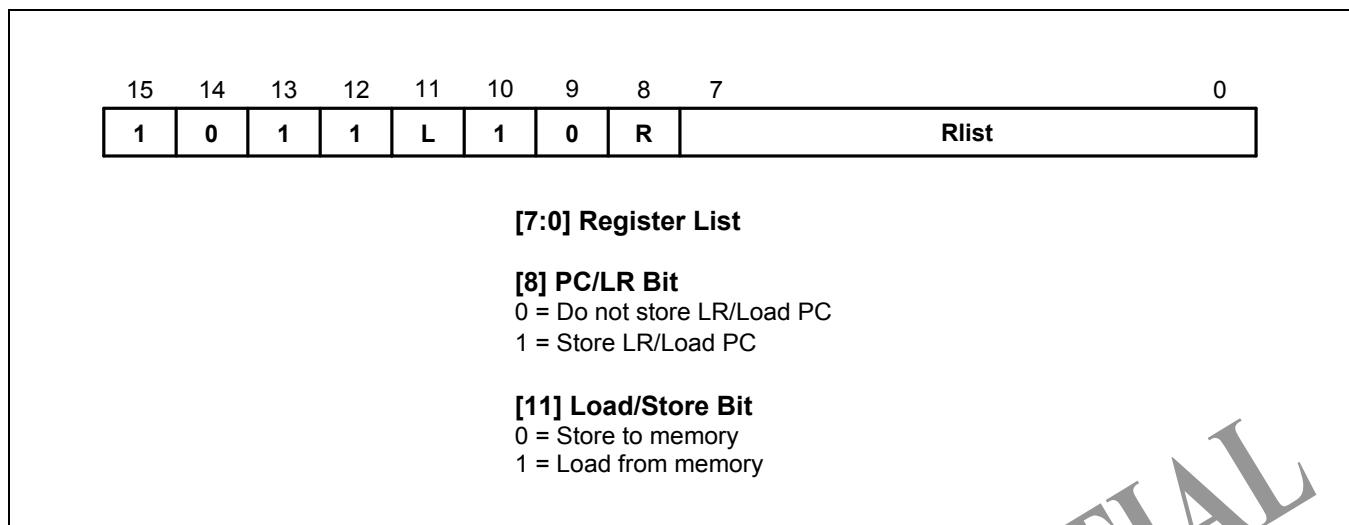


Figure 3-47. Format 14

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 3-24.

NOTE

The stack is always assumed to be Full Descending.

Table 3-24. PUSH and POP Instructions

L	R	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 3-24. 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

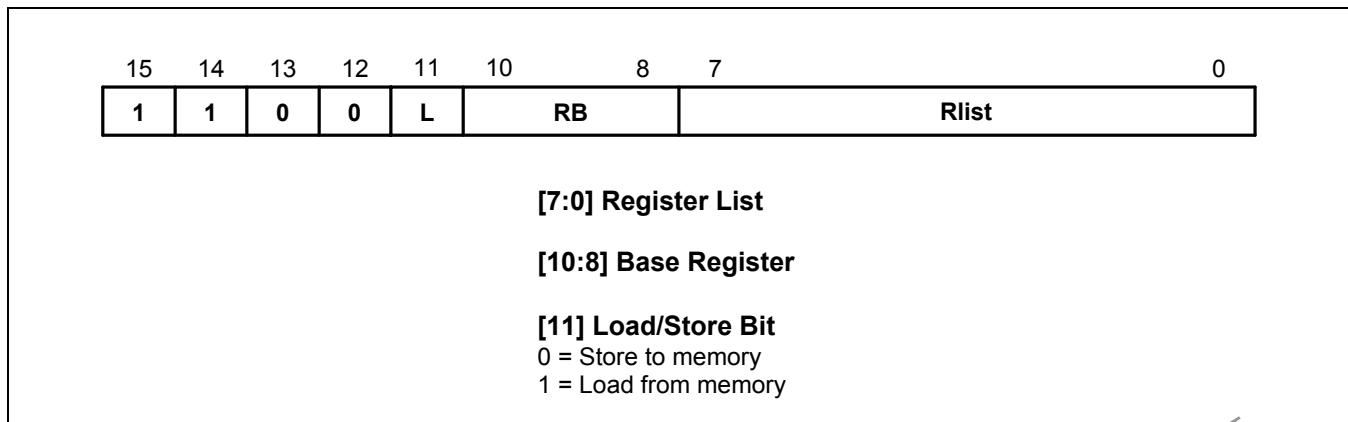


Figure 3-48. Format 15

OPERATION

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

Table 3-25. 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 3-25. 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.

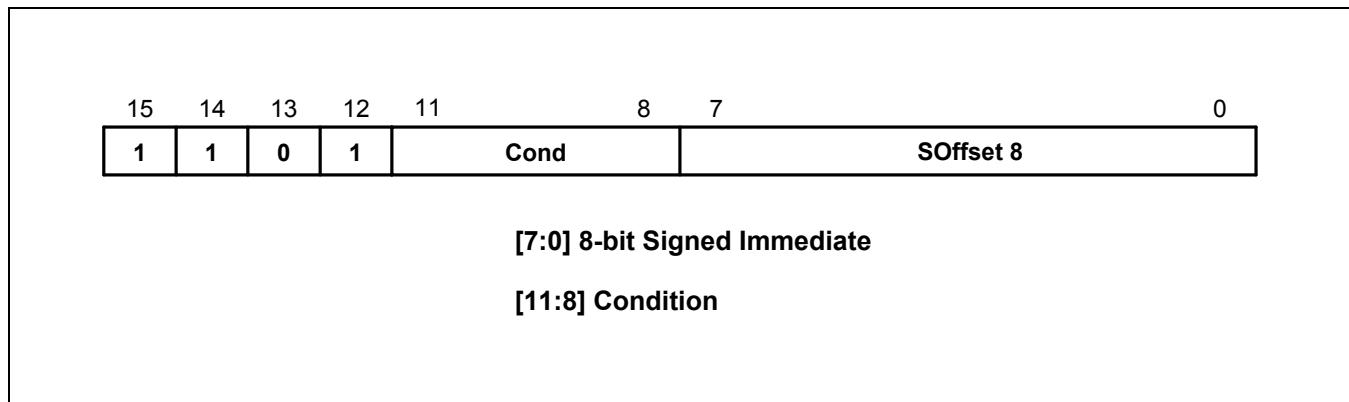
FORMAT 16: CONDITIONAL BRANCH

Figure 3-49. 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 3-26. The Conditional Branch Instructions

Cond	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)
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 3-26. 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

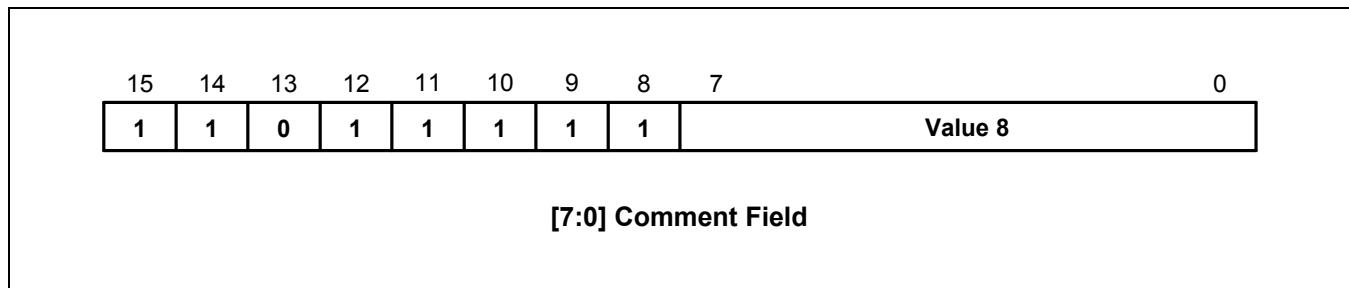


Figure 3-50. Format 17

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 3-27. 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 3-27. 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

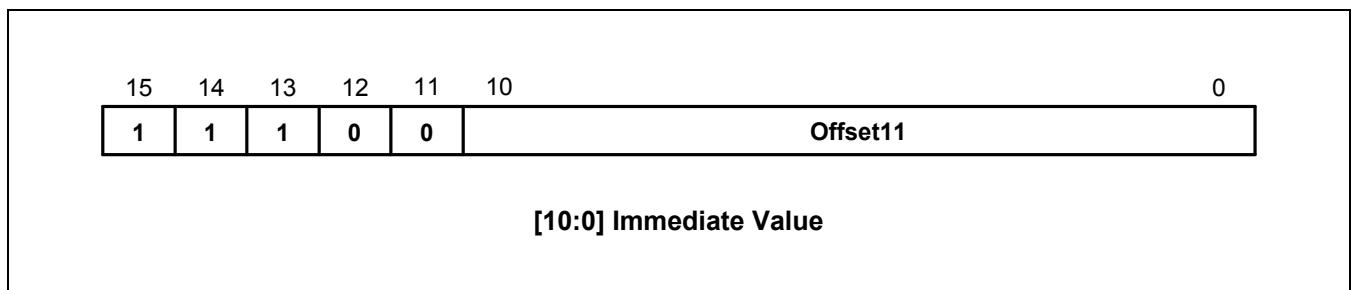


Figure 3-51. Format 18

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 3-28. Summary of Branch Instruction

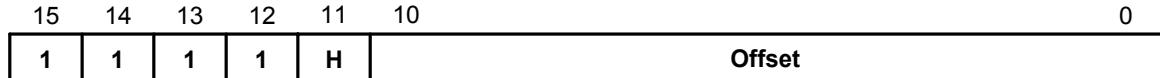
THUMB assembler	ARM equivalent	Action
B label	BAL label (halfword offset)	Branch PC relative +/- Offset11 << 1, where label is PC +/- 2048 bytes.

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



[10:0] Long Branch and Link Offset High/Low

[11] Low/High Offset Bit

0 = Offset high
1 = Offset low

Figure 3-52. 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 3-29. 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^k

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 (0xFFFFFFFF) 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 xorring 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 (0xFFFFFFFF) 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 >.

just_I 0	LSR	R0, R1, #1	
	MOV	R2, R3	
	B	%FT0	
	LSL	R2, #1	
	CMP	R2, R0	
	BLS	just_I	
MOV	R0, #0	; Set accumulator to 0	
B	%FT0	; Branch into division loop	
div_I 0	LSR	R2, #1	
	CMP	R1, R2	; Test subtract
	BCC	%FT0	
	SUB	R1, R2	; If successful do a real subtract
	ADC	R0, R0	; Shift result and add 1 if subtract succeeded
CMP	R2, R3	; Terminate when R2 == R3 (ie we have just	
BNE	div_I	; tested subtracting the 'ones' value).	

Now fixup 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		
CMP	a3, a2, LSR #1	; Justification stage shifts 1 bit at a time
MOVLS	a3, a3, LSL #1	
BLO	s_loop	; NB: LSL #1 is always OK if LS succeeds
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, #100  
    MOV     pc, lr
```

ARM Code

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

NOTES

SAMSUNG CONFIDENTIAL

4 CALMADM2E

INTRODUCTION

CalmADM2E, a CalmRISC based audio DSP module, is designed for high-quality audio processing. It includes Samsung's 16-bit microprocessor, CalmRISC16E, and 24-bit DSP engine, CalmMAC24E. CalmADM2E also includes three caches, one instruction cache and two data caches. To keep high performance with optimal die area, two data caches are adopted instead of large on-chip data memories typically used in other audio processors. Since it is used to encode/decode large audio data frames, CalmADM2E includes two sequential buffers to handle input/output data efficiently. These sequential buffers can be act as a kind of ring buffer also. CalmADM2E communicate with host CPU through four communication channels. Through two of four channels, Host and CalmADM2E can send large communication data to each other. Other two channels are just for sending events to each other. Since CalmADM2E is a master module on AHB+ bus, it accesses external system memory through AHB+ bus. Since CalmADM2E is also a slave module on AHB+ bus, Host can control it through AHB+ bus. CalmADM2E supports both synchronous and asynchronous AHB+ bus interfaces. In this document, we call CalmADM2E as an abbreviation, ADM.

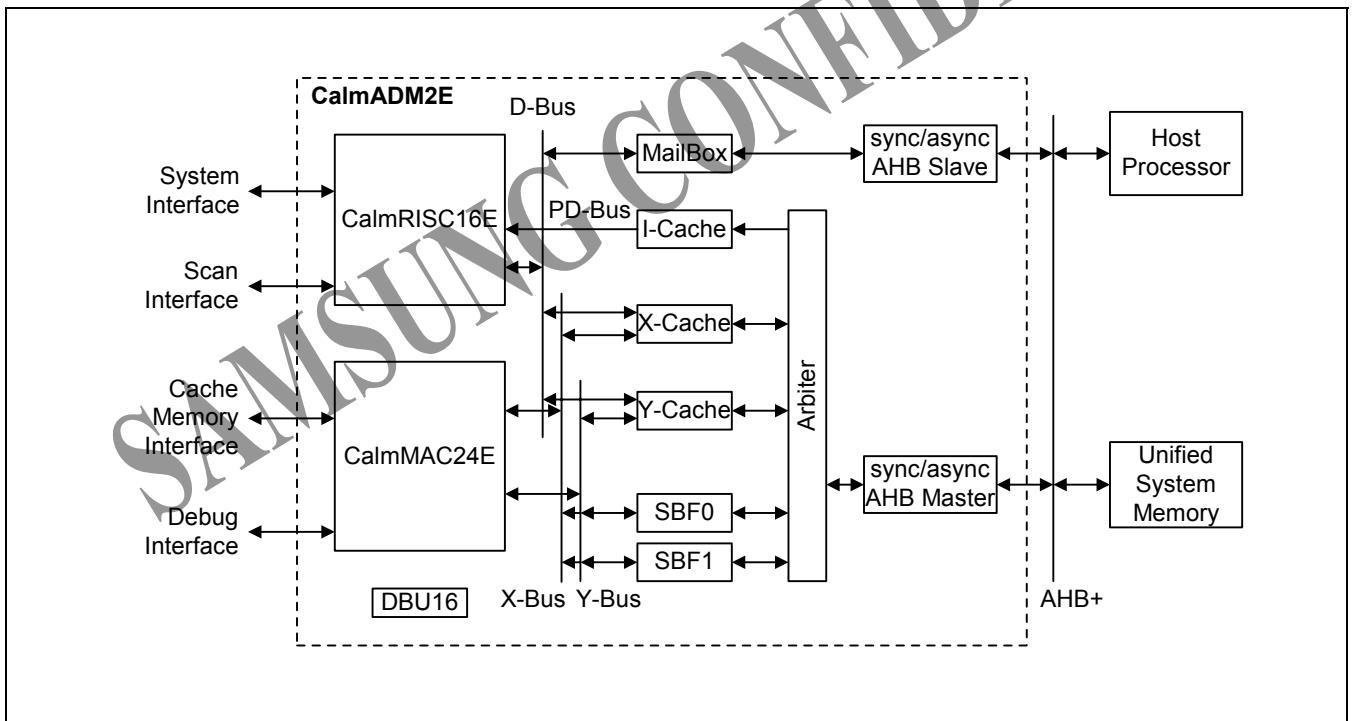


Figure 4-1. CalmADM2E Block Diagram

FEATURES

CalmRISC16E

- 16-bit low power & high performance RISC micro-controller
- Harvard style architecture:
 - 4M-byte program memory space,
 - 4M-byte data memory space
- 5-stage pipelined instruction execution
- 16-bit (half-word) instruction set
- Sixteen 16-bit general-purpose registers with eight 6-bit extension registers
- Program address bus with normal timing distinguishes CalmRISC16E from CalmRISC16F that has one-cycle-delayed PA bus.
- In this document, we call CalmRISC16E as an abbreviation, Calm.

CalmMAC24E

- 24-bit (aud-word) high performance fixed-point DSP coprocessor for CalmRISC16E
- Single cycle 24×24 MAC operations
- 32K aud-word X data memory space & 32K aud-word Y data memory space
- Two multiplier accumulator registers, four general accumulator registers, and eight pointer registers
- One-cycle-delayed EC[2:0] output pins distinguish CalmMAC24E from CalmMAC24F that has EC[2:0] with normal timing.
- In this document, we call CalmMAC24E as an abbreviation, Mac.

Internal Memory

- Instruction cache: 256-bit line, 4K-byte, direct mapped cache
- X Data cache: 192-bit line, 6K-byte, 2-way set associative cache
- Y Data cache: 192-bit line, 6K-byte, 2-way set associative cache
- Two 16-byte sequential buffers: configurable sequential ring buffer mode or sequential linear buffer mode
- Two 16-byte communication data registers.

Register Sets

- AFRS (ADM Function Register Set):
ADM internal registers for control flags, status flags, system memory base addresses, and communication mailbox registers.
- SFRs (Special Function Registers) in ADM:
Part of system level SFRS (Special Function Register Set) that Host can read/write.
For control flags, status flags, system memory base addresses and communication mailbox registers.

FEATURES (Continued)**AHB+ interfaces**

- Both asynchronous and synchronous AHB+ master interfaces for system memory access.
- Both asynchronous and synchronous AHB+ slave interfaces for host interface

Clocks

- Dual Clock: One processor clock and One bus clock

Performance

- Max. Operating Frequency = 140MHz
- @ (Voltage: 1.1V ~ 1.3V, Temperature: -40°C ~ 125°C, Process: Samsung L13

SAMSUNG CONFIDENTIAL

DSP PROGRAMMING MODEL

MEMORY CONFIGURATIONS

Both program and data of ADM are in off-chip memory. All of program memory space is cacheable with an instruction cache. Most of data memory space is cacheable also with two data caches. The non-cacheable data memory space is reserved for function registers and sequential blocks.

Memory Configuration in Calm Side

Data Memory: 4M-byte space, 6K-byte X-Cache and 6K-byte Y-Cache
(X-Cache and Y-Cache for Mac are shared as data cache of Calm)

Program Memory: 4M-byte space, 4K-byte I-Cache

Memory Configuration in Mac Side

X-data Memory: 32K aud-word space, 6K-byte X-Cache

Y-data Memory: 32K aud-word space, 6K-byte Y-Cache

Program Memory

Program memory configuration is shown in Figure 4-2. 4M-byte cacheable program memory space can be placed any where in 4G-byte off-chip memory space.

Calm accesses program memory with its 21-bit program address bus, PA[20:0]. The 4K-byte instruction cache (I-Cache) covers whole program memory space. PA, which is a half-word address, is converted as a byte address (multiplied by 2), added with ADM_IBASE and mapped to 32-bit off-chip memory address. ADM_IBASE is a 32-bit register containing base address of program memory.

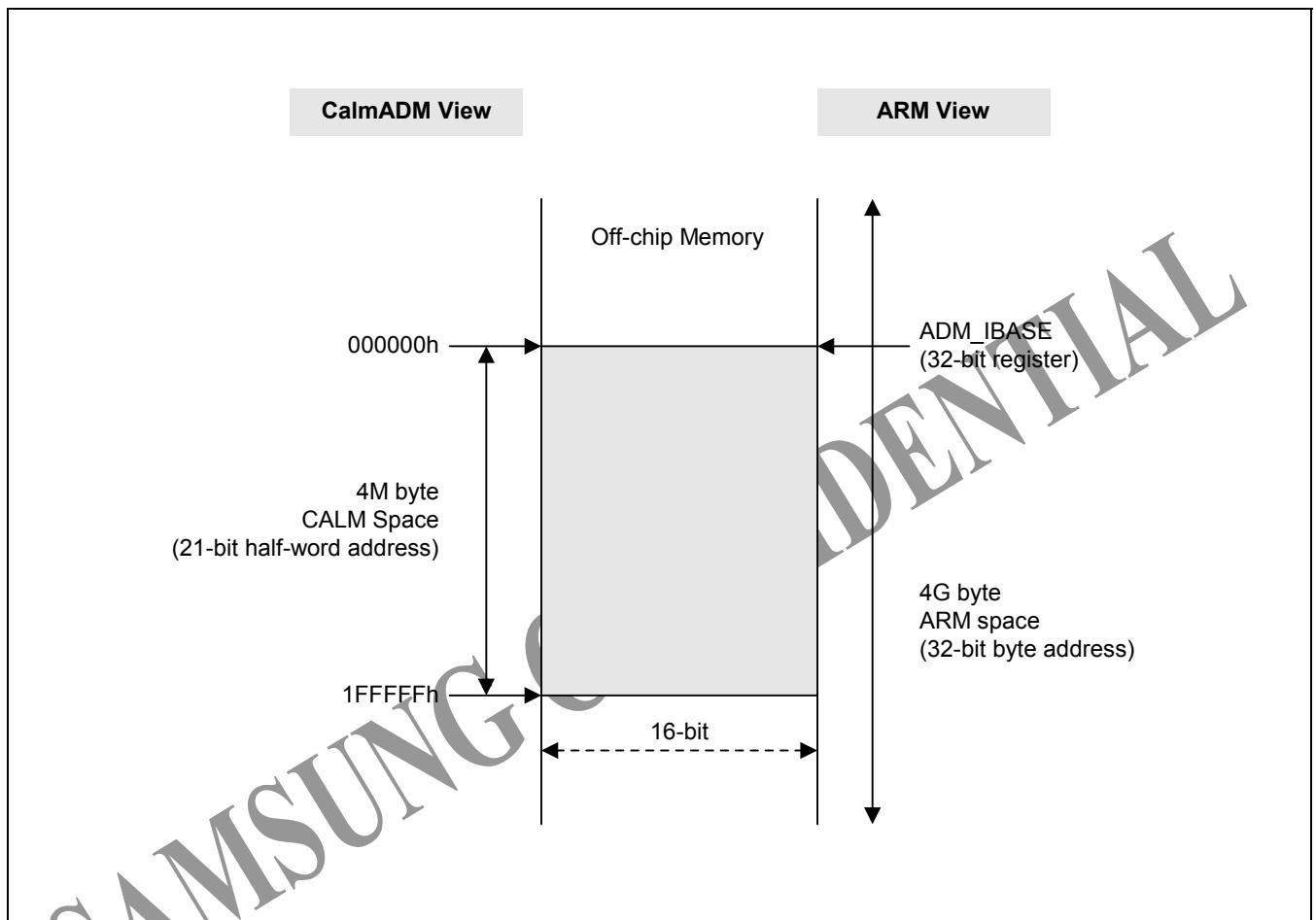


Figure 4-2. Program Memory Configuration

Data Memory

Data memory configuration is shown in Figure 4-3 and Figure 4-4. Data memory areas can be placed anywhere in 4G-byte off-chip memory space. 4M-byte data memory space is divided into 7 areas, Calm area, MAC X area, MAC Y area, two sequential block areas, Unused area and I/O area.

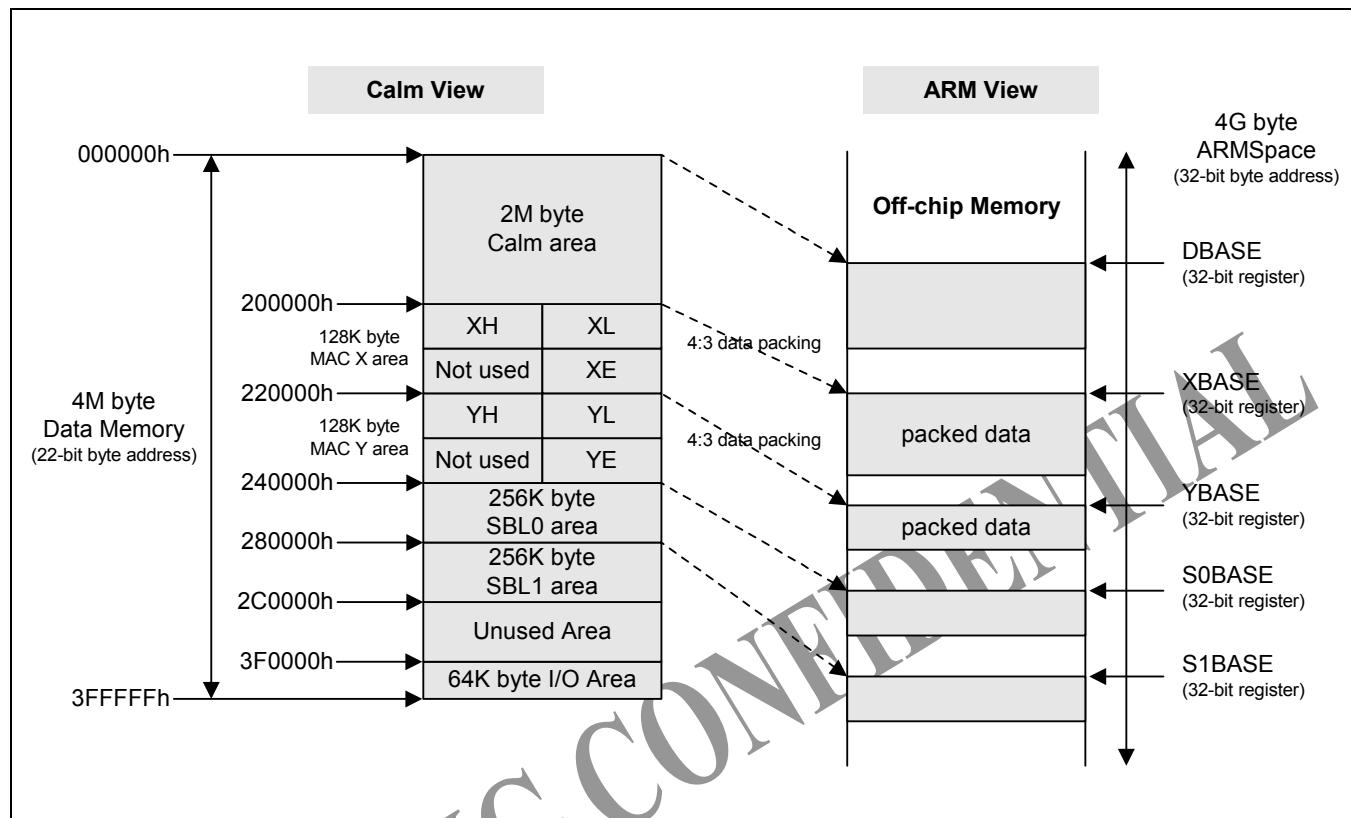


Figure 4-3. CalmRISC16E Data Memory Configuration

Calm Area

Calm area is 2M-byte data area that can be accessed by Calm only. If DA[21] (MSB of data address output from Calm) is 0, Calm accesses Calm area. To access this area, Calm uses two data caches (X-Cache and Y-Cache) as a two way set associative cache. DA from the cache is added with ADM_DBASE and mapped to 32-bit off-chip memory address. ADM_DBASE is 32-bit register containing base address of Calm area.

MAC X/Y Area

MAC X area and MAC Y area are dual memory for dual load capability of Mac. Each area is 32K-aud-word. Both two cores (Calm and Mac) can access these two areas. If DA[21:17] is 10000b, Calm accesses MAC X area. To access this area, Calm uses X-Cache as data cache. DA[16:0] from Calm is converted as MAC X area address before get into X-Cache. If DA[21:17] is 10001b, Calm accesses MAC Y area. To access this area, Calm uses Y-Cache as data cache. DA[16:0] from Calm is converted as MAC Y area address before get into Y-Cache. Mac accesses two MAC areas with its two 15-bit data addresses, XA and YA (equivalent to DA[16:2] in byte addressing). The 6K-byte X-Cache covers 96K-byte MAC X area. The 6K-byte Y-Cache covers 96K-byte MAC Y area. 15-bit XA from X-cache is 2-bit left shifted (conversion to byte address), converted as 4:3 reduction, added with ADM_XBASE and finally mapped to 32-bit off-chip memory address. YA mapping is similar to XA. Two words in highest address of MAC X and two words in highest address of MAC Y are reserved for accessing sequential block areas.

SBL0/SBL1 Area

Sequential block areas are used for input and output of audio data in well-defined and well-aligned data frames.

Calm can access these areas randomly but Mac can access them in sequential way only. Since these areas are out of Mac's memory space, Mac cannot access these areas in normal way. In ADM, a special logic was added so that Mac can access these areas sequentially. When DA[21:18] is 1001b, Calm accesses SBL0 area. Before mapped to 32-bit off-chip memory address, DA[17:0] from Calm is added with ADM_S0BASE, a 32-bit register containing base address of SBL0 area. When DA[21:18] is 1010b, Calm accesses SBL1 area similarly to SBL0 area. Since ADM has no caches for these areas, Calm accesses off-chip memory directly when it accesses these areas. If XA or YA are 7FFEh, Mac accesses SBL0 area. If XA or YA are 7FFFh, Mac accesses SBL1 area. More of sequential data accessing is described in later of this document.

UnUsed Area

Unused area is a part of Calm's data memory space addressed from 2C0000h to 3EFFFFh. Since there is no physical memory for unused area, reading this area returns zero and writing results no effects.

I/O Area

I/O area is reserved for memory-mapped registers of ADM. I/O area resides above 3F0000h address space in Calm's data memory space.

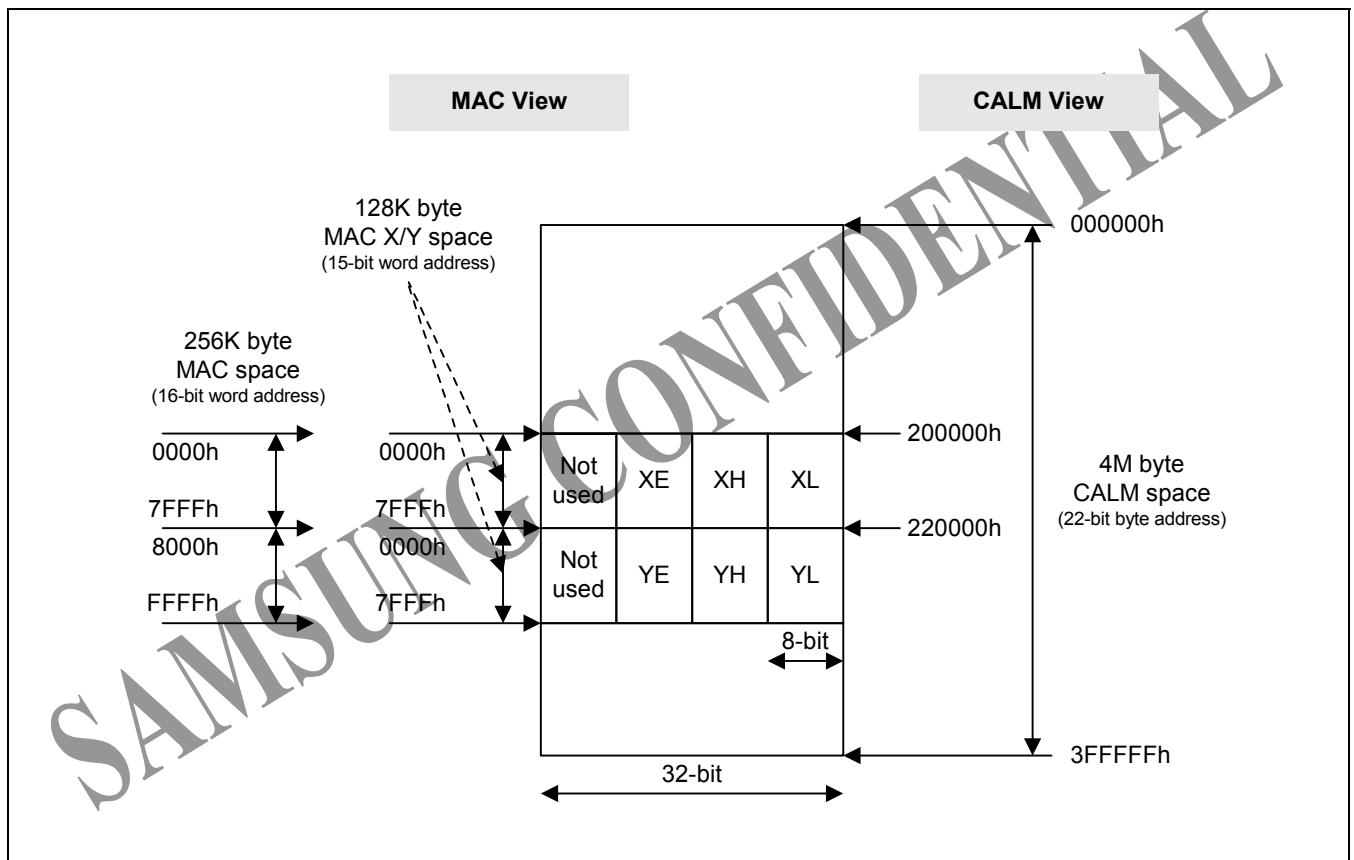


Figure 4-4. CalmMAC24E Data Memory Configuration

MAC Area Considerations

The target Host of ADM is 32-bit machine, Mac is a 24-bit DSP and Calm is a 16-bit RISC processor. Because of different size of unit data among three processors, following two address conversions are needed for MAC X/Y areas.

DA Conversion for MAC Area

Because of the data unit difference, Calm and Mac have different address maps of MAC X/Y area. (Look at the difference between MAC X/Y areas in Figure 4-3 and Figure 4-4) In addition, proper address conversion is needed when Calm accesses MAC X/Y area. DA[21:0] from Calm is converted to DA_mem[21:0] in following manner before it goes to caches or off-chip memory.

*DA to DA_mem conversion:

```
if ((DA[21:17] == 0b10000) or (DA[21:17] == 0b10001))
    DA_mem[21:0] = "DA[21:17],DA[15:1],~DA[16],DA[0]"
else
    DA_mem[21:0] = DA[21:0]
```

4:3 Reductions for unUsed Bytes

Since Mac is a 24-bit DSP and its memory data are word (32-bit) aligned, there are unused bytes in data words of MAC X/Y area. Since there is no physical memory for these bytes, two things should be considered. The first is accessing these bytes by Calm. Calm may access these unused bytes, but a write to these bytes cannot be done and a read from these bytes always results 00h. The second is waste of off-chip memory area. If MAC X/Y areas are mapped to off-chip memory directly, it causes waste 1/4 of off-chip memory area. So, in ADM, addresses of MAC X/Y area from X/Y-caches are converted as 4:3 reduction before mapped to off-chip memory addresses. As a result of this conversion, size of each MAC X area and MAC Y area will be 96K-bytes in off-chip memory instead of 128K-bytes.

SEQUENTIAL DATA ACCESS

If a data block in off-chip memory is well aligned in well-known structure, it can be accessed in sequential. In ADM, user can define two sequential blocks in Calm data memory area and can access them sequentially through sequential buffers. There are two sequential access modes, linear mode and ring mode.

Concepts and Definitions

Sequential Block Areas

- Data memory area in which sequential blocks can be defined.
- In Calm data memory area, two 256K-byte sequential block areas are defined. One starts from 240000h address and the other starts from 280000h.
- Calm can access these areas randomly but Mac can't access in normal way because these areas are defined out of MAC X/Y areas. ADM offers special way for Mac to access these areas sequentially.

Sequential Blocks

- Well-aligned data memory area that can be sequentially accessed by Mac through sequential buffers.
- Even though unit data in a sequential block can be word or half-word, a sequential block should be word-aligned for efficient off-chip memory access.
- Data in a sequential block are addressed with Offset register.
- These are soft blocks created, used and removed dynamically by software.
- In linear mode, the size and boundary of a sequential block are implicitly defined by how user program accesses the sequential block. In ring mode, the size and boundary of a sequential block are explicitly defined by the values of Begin Offset register and End Offset register.

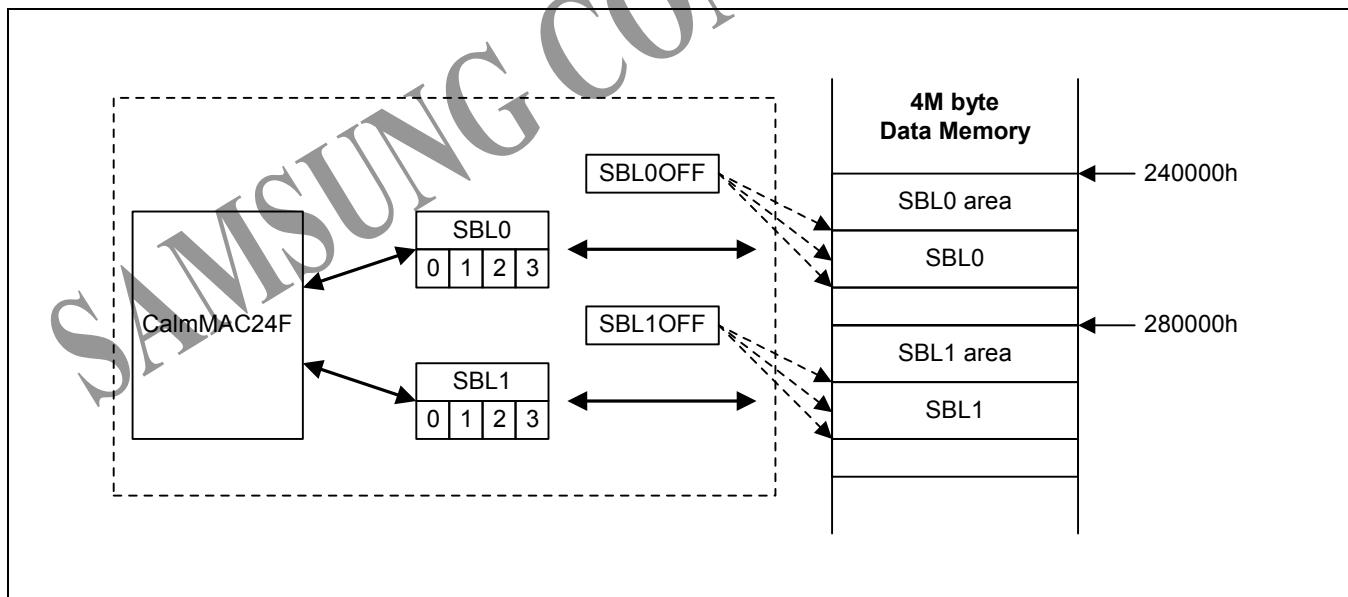


Figure 4-5. Sequential Access Data Flow

Sequential Buffers

- A kind of FIFO that is used as buffer memory for Mac to access sequential blocks sequentially.
- These buffers can be both read buffer and write buffer.
- Pre-loading and post-storing capability are supported.
These capabilities can be done explicitly by issuing special commands defined in AFRS.
- Size of sequential buffers
 - 128-bits
 - 4 data for word data, 8 data for half-word data
- Address of sequential buffers
 - Sequential buffers are defined as a memory data located in special address.
 - SBF0: 21FFF8h (in X area) or 23FFF8h (in Y area)
 - SBF1: 21FFFCh (in X area) or 23FFFCh (in Y area)

Sequential Block Offset Registers

- 18-bit registers that contain offset address of memory data to be accessed at next sequential access.
- Offset registers are automatically increased just after a sequential access.
- In ring mode, increased Offset register is compared with the End Offset register. If two register values are same, Offset register is set as the value of Begin Offset register.
- Offset registers can be read or written directly since these are memory mapped in I/O area of Calm.
- When Offset register is written, the value can be any address in sequential block area. The written value is independent from the values of boundary-offset registers even in ring mode.
- The bases of offsets are start addresses of sequential block areas.
- The offset values are byte-address.

Sequential Block Boundary-offset Registers

- 18-bit registers containing boundary-offset values (begin/end) of a sequential block used in ring mode.
- In ring mode, a sequential block is define as below:
 - *Begin Offset* =< SBL < *End Offset*
 - **CAUTION: End Offset is not included in sequential block.**
- In linear mode, boundary-offsets are unused.
- These registers can be read or written directly since these are memory mapped in I/O area of Calm.
- The bases of boundary-offsets are start addresses of sequential block areas.
- The boundary-offset values are byte-address.

Ring Mode vs. Linear Mode

- In linear mode, the size and boundary of a sequential block is not fixed. Defining a sequential block and checking its boundary totally depend on user program.
- In ring mode, the size and boundary of a sequential block is defined by the values of Begin Offset register and End Offset register. Boundary checking is performed by hardware.
- In ring mode, when Offset register is increase to meet the value of End Offset register, two operations are automatically performed by hardware.
 - Offset wrapping:
Offset register is set as the value of Begin Offset register. (We call it ‘wrapping’ in this document)
 - Interrupt:
Corresponding interrupt flag is set and interrupt is forced to Calm if the interrupt is enabled.

SAMSUNG CONFIDENTIAL

OPERATIONS in linear mode

Following tables show how to access SBL0 in linear mode. SBL1 can be accessed similarly.

Table 4-1. SBL0 as a Sequential Read Block in Linear Mode

Explicit Operations	Implicit Operations
Set SBL0OFF	<ul style="list-style-type: none"> - Set SBL0OFF as start offset of sequential block - Invalidate all data words in SBF0
Run buffer fill command	<ul style="list-style-type: none"> - Fetch words located from {SBL0OFF[17:4],SBL0OFF[3:2],00} to {SBL0OFF[17:4],11,00} in SBL0 into SBF0 - Validate fetched words in SBF0
Read SBF0	<ul style="list-style-type: none"> - Data read - Invalidate read word and Increase SBL0OFF - If SBF0 is empty <ul style="list-style-type: none"> - fetch new 4 words in (SBL0OFF[17:4],0000b) location of SBL0
Repeating read from SBF0 results sequential reads	

Table 4-2. SBL0 as a Sequential Write Block in Linear Mode

Explicit Operations	Implicit Operations
Set SBL0OFF	<ul style="list-style-type: none"> - Set SBL0OFF as start offset of sequential block - Invalidate all data words in SBF0
Write to SBF0	<ul style="list-style-type: none"> - Data write - Validate written word and Increase SBL0OFF - If SBF0 is full <ul style="list-style-type: none"> - flush valid words in SBF0 into (SBL0OFF[17:4]-1,0000b) location of SBL0 and invalidate flushed words.
Repeating write into SBF0 results sequential writes	
Run buffer flush command	Flush valid words and invalidate flushed words

Table 4-3. SBL0 Area as Randomly Accessed Data Memory

Explicit Operations	Implicit Operations
Calm accesses SBL0 area in off-chip memory directly	<p>To keep data consistency,</p> <ul style="list-style-type: none"> - ADM checks if it is hit on one-lined-cache (SCACHE0) (SCACHE0 consists of SBL0OFF as tag and SBF0 as a data line.) - Write policy: write-through - No replacement is done on miss

OPERATIONS in Ring Mode

Following tables show how to access SBL0 in ring mode. SBL1 can be accessed similarly.

Table 4-4. SBL0 as a Sequential Read Block in Ring Mode

Explicit Operations	Implicit Operations
Set SBL0BEGIN and SBL0END	* No special implicit operations
Set SBL0OFF	* Same as descriptions in Table 4-1.
Run buffer fill command	
Read SBF0	<ul style="list-style-type: none"> - Data read - Invalidate read word and Increase SBL0OFF - If SBL0OFF is equal to SBL0END, <ul style="list-style-type: none"> - set SBF0 interrupt flag - invalidate entire SBF0 - set SBL0OFF as SBL0BEGIN - Fetch words located from {SBL0OFF[17:4],SBL0OFF[3:2],00} to {SBL0OFF[17:4],11,00} in SBL0 into SBF0 - Validate fetched words in SBF0 - If SBF0 is empty <ul style="list-style-type: none"> - fetch new 4 words from (SBL0OFF[17:4],0000b) location of SBL0
Repeating read from SBF0 results sequential reads in ring mode	

Table 4-5. SBL0 as a Sequential write Block in Ring Mode

Explicit Operations	Implicit Operations
Set SBL0BEGIN and SBL0END	* No special implicit operations
Set SBL0OFF	* Same as descriptions in Table 4-2.
Write to SBF0	<ul style="list-style-type: none"> - Data write - Validate written word and Increase SBL0OFF - If SBL0OFF is equal to SBL0END, <ul style="list-style-type: none"> - set SBF0 interrupt flag - Flush valid words and invalidate flushed words - set SBL0OFF as SBL0BEGIN - If SBF0 is full <ul style="list-style-type: none"> - flush valid words in SBF0 into (SBL0OFF[17:4]-1,0000b) location of SBL0 and invalidate flushed words.
Repeating write into SBF0 results sequential writes	
Run buffer flush command	* Same as descriptions in Table 4-2.

In ring mode, Calm randomly accesses SBL0 area in the same way as the case of linear mode. Refer Table 4-3.

CALMADM2E INTERNAL CONTROLS

Interrupts

ADM has no external interrupt inputs. It has five internal interrupt sources. The five sources are gathered as an interrupt signal that Calm gets as source of its IRQ. Calm has two interrupt input pins, nFIQ and nIRQ. nFIQ is tied to high in ADM. Only nIRQ is used.

IRQ

In ADM are five internal interrupt sources. Three of them are defined as interrupt pending flags of COMMUN register in AFRS. These are parts of communication channels between Host and ADM. In Section 'System Programming Model', communication channels will be described in detail. User can enable/disable these interrupt sources by set/resetting corresponding interrupt enable flags of CONFIG register in AFRS. Remained two of them are defined as interrupt pending flags of SBFSTAT register in AFRS. These are for wrapping interrupts caused by ring mode operations of sequential buffers described in Section 'Sequential Data Access'. These interrupts are enabled while corresponding sequential buffer is in ring mode. If one or more among enabled interrupt pending flags are set, IRQ of Calm is active. User should clear an interrupt pending flag by issuing proper interrupt flag clear command defined in COMMUN or SBFCON register in AFRS before returning from the corresponding interrupt service routine.

SYS Commands

In ADM, one SYS command is used internally.

SYS_IDLE

SYS #5h is defined as 'sys_idle' meaning that Calm goes to idle mode after execution of this command. Upon the interrupts or wake up command from debugger unit, Calm exits from idle mode.

ADM FUNCTION REGISTER SET

Following registers are mapped on IO Area in data memory area of Calm. These registers are addressed with 7bit off set address. A full address value is an addition of an off set address and the base address. In ADM, the base address was defined as 0x3F0000.

Register 1: CONFIG

CONFIG0: configuration/control register 0

Reset Value: 0000h

BASE + 0h		Mode: read/write
Bit	Name	Description
15	SBF0 mode [3]	Sequential access mode selection bit 0 : linear mode 1 : ring mode
14:12	SBF0 mode [2:0]	* When SBF0 mode [2] is 0, access unit of external memory is 32 bit. * When SBF0 mode [2] is 1, access unit of external memory is 16 bit. <as a read buffer> 00x : mac input [23:0] <- external input [23:0] 01x : mac input [23:0] <- external input [32:8] 100 : mac input [23:0] <- zero extension of external input [15:0] 101 : mac input [23:0] <- sign extension of external input [15:0] 11x : mac input [23:8] <- external input [15:0], mac input [7:0] <- 00h <as a write buffer> 000 : external output [31:0] <- zero extension of mac output [23:0] 001 : external output [31:0] <- sign extension of mac output[23:0] 01x : external output [31:8] <- mac output [23:0], external output [7:0] <- 00h 10x : external output [15:0] <- mac output [15:0] 11x : external output [15:0] <- mac output [23:8]
11:8	SBF1 mode [3:0]	* Similar to the description of 'SBF0 mode [3:0]'
7	—	Unused (reading returns zero)
6	Host Event Interrupt Enable	If 1, host event interrupt generation is allowed.
5	Upload Complete Interrupt Enable	If 1, upload complete interrupt is generated when upload is completed.
4	New Download Interrupt Enable	If 1, download request interrupt is generated when a new download request is generated.
3:2	—	Unused (reading returns zero)
1	SBF1 Interrupt Enable	If 1, SBF1 interrupt is forced to Calm when SBF1 interrupt flag is set.
0	SBF0 Interrupt Enable	If 1, SBF0 interrupt is forced to Calm when SBF0 interrupt flag is set.

NOTE: When sbf0mode/sbf1mode flags are written, newly updated values are not effective to the operation of sequential buffers. New values are effective after the sequential buffer is newly initialized, which means that SBL0OFF/SBL1OFF registers are newly written.

CONFIG1: configuration/control register 1

Reset Value: 0000h

BASE + 2h		Mode: read/write
Bit	Name	Description
15:6	—	Unused (reading returns zero)
5:4	xyrr	XY-Cache round robin code. When a Calm area access is cache-missed, one of X- or Y-Cache is replaced according to this code. 00: X- and Y-Cache are selected one after another (round-robin). At the first miss, X-Cache is selected. 01: X-Cache is selected. 1x: Y-Cache is selected.
3:1	—	Unused (reading returns zero)
0	ldcinv	This flag is for partial invalidation. If it is set, all LDC instruction of Calm invalidates the target address line in I-Cache.

NOTE: xyrr bits should change while both X- and Y-Caches are disabled. Otherwise, data coherence may not be kept.

Register 2: COMMUN

Communication Control Register

Reset Value: 0000h

BASE + 4h		Mode: read/write
Bit	Name	Description
15:7	—	Unused (reading returns zero)
6	Host Event Interrupt Flag	Read only bit. Writing does not affect this bit. It is set when host generates an event. It is cleared when ADM commands 'clear host event interrupt flag'.
5	Upload Complete Interrupt Flag	Read only bit. Writing does not affect this bit. It is set when host has done an upload transaction. It is cleared when ADM commands 'new upload' or 'clear upload complete interrupt flag'.
4	New Download Interrupt Flag	Read only bit. Writing does not affect this bit. It is set when host requests a new upload transaction. It is cleared when ADM commands 'download complete' or 'clear new download interrupt flag'.
3	—	Unused (reading returns zero)
2:0	ADM Command Flag	Write only bits. Reading returns zero. 000 : new upload 001 : download complete 01x : ADM event generation 100 : clear new download interrupt flag 101 : clear upload complete interrupt flag 11x : clear host event interrupt flag

Register 3: DDATA0 ~ DDATA7

Download Data Registers

Reset Value: xxxxh

BASE + 6h ~ 14h		Mode: Read only. Writing does not affect these registers.
Bit	Name	Description
[15:0]	Download Data	Communication data from host to ADM

Register 4: UDATA0 ~ UDATA7

Upload Data Registers

Reset Value: xxxxh

BASE + 16h ~ 24h		Mode : read/write
Bit	Name	Description
[15:0]	Upload Data	Communication data from ADM to host

Register 5: IBASE

IBASE_H: Higher half of start address for ADM instruction area

Reset Value: xxxxh

BASE + 26h		Mode: Read only. Writing does not affect these registers.
Bit	Name	Description
[15:0]	Ibase_h	Higher 16 bits of CalmRISC instruction area base address

IBASE_L: Lower half of start address for ADM instruction area

Reset Value: xxxxh

BASE + 28h		Mode: Read only. Writing does not affect these registers.
Bit	Name	Description
[15:4]	Ibase_l	Middle 12 bits of CalmRISC instruction area base address
[3:0]	-	Lower 4 bits of CalmRISC instruction area base address, fixed to 0

Register 6: DBASE

DBASE_H: Higher half of start address for CalmRISC data area

Reset Value: xxxxh

BASE + 2Ah		Mode: Read only. Writing does not affect these registers
Bit	Name	Description
[15:0]	dbase_h	Higher 16 bits of CalmRISC data area base address

DBASE_L: Lower half of start address for CalmRISC data area

Reset Value: xxxxh

BASE + 2Ch		Mode: Read only. Writing does not affect these registers
Bit	Name	Description
[15:4]	dbase_l	Middle 12 bits of CalmRISC data area base address
[3:0]	-	Lower 4 bits of CalmRISC data area base address, fixed to 0

Register 7: XBASE

XBASE_H: Higher half of start address for Mac X area

Reset Value: xxxxh

BASE + 2Eh		Mode : read/write
Bit	Name	Description
[15:0]	xbase_h	Higher 16 bits of Mac X area base address

XBASE_L: Lower half of start address for Mac X area

Reset Value: xxxxh

BASE + 30h		Mode : read/write
Bit	Name	Description
[15:10]	xbase_l	Middle 6 bits of Mac X area base address
[9:0]	-	Lower 10 bits of Mac X area base address, fixed to 0

Register 8: YBASE

YBASE_H: Higher half of start address for Mac Y area

Reset Value: xxxxh

BASE + 32h		Mode : read/write
Bit	Name	Description
[15:0]	ybase_h	Higher 16 bits of Mac Y area base address

YBASE_L: Lower half of start address for Mac Y area

Reset Value: xxxxh

BASE + 34h		Mode : read/write
Bit	Name	Description
[15:10]	ybase_l	Middle 6 bits of Mac Y area base address
[9:0]	-	Lower 10 bits of Mac Y area base address, fixed to 0

Register 9: S0BASE

S0BASE_H: Higher half of start address for sequential buffer 0 area

Reset Value: xxxxh

BASE + 36h		Mode : read/write
Bit	Name	Description
[15:0]	s0base_h	Higher 16 bits of sequential block 0 area base address

S0BASE_L: Lower half of start address for sequential buffer 0 area

Reset Value: xxxxh

BASE + 38h		Mode : read/write
Bit	Name	Description
[15:4]	s0base_l	Middle 12 bits of sequential block 0 area base address
[3:0]	-	Lower 4 bits of sequential block 0 area base address, fixed to 0

Register 10: S1BASE

S1BASE_H: Higher half of start address for sequential buffer 1 area

Reset Value: xxxxh

BASE + 3Ah		Mode : read/write
Bit	Name	Description
[15:0]	s1base_h	Higher 16 bits of sequential block 1 area base address

S1BASE_L: Lower half of start address for sequential buffer 1 area

Reset Value: xxxxh

BASE + 3Ch		Mode : read/write
Bit	Name	Description
[15:4]	s1base_l	Middle 12 bits of sequential block 1 area base address
[3:0]	-	Lower 4 bits of sequential block 1 area base address, fixed to 0

Register 11: CACHECON

Cache Control Register

Reset Value: 0000h

BASE + 3Eh		Mode: write only. Reading returns zero.
Bit	Name	Description
[15:10]	-	Unused (reading returns zero)
[9:8]	ic command flag	00: No operation 01: I-Cache invalidation 10: I-Cache enable. 11: I-Cache disable (go to bypass mode).
[7]	-	Unused (reading returns zero)
[6:4]	xc command flag	000: No operation 001: X-Cache invalidation 010: X-Cache enable. 011: X-Cache disable (go to bypass mode). 1xx: X-Cache flush
[3]	-	Unused (reading returns zero)
[2:0]	yc command flag	* Similar to the description of 'xc command flag'

Register 12: CACHESTAT

Cache status register

Reset Value: 0000h

BASE + 40h		Mode: read only. Writing does not affect this register.
Bit	Name	Description
[15:10]	—	Unused (reading returns zero)
[9:8]	ic state flag	00: Undefined 01: I-Cache is in invalidation state. 10: I-Cache is in normal state. 11: I-Cache is in bypass state.
[7]	—	Unused (reading returns zero)
[6:4]	xc state flag	000: Undefined 001: X-Cache is in invalidation state. 010: X-Cache is in normal state. 011: X-Cache is in bypass state. 1xx: X-Cache is in flush state.
[3]	—	Unused (reading returns zero)
[2:0]	yc state flag	* Similar to the description of 'xc command flag'

Register 13: SBFCON

Sequential Buffer Control Register

Reset Value: 0000h

BASE + 42h		Mode: write only. Reading returns zero.
Bit	Name	Description
[15:14]	—	Unused (reading returns zero)
[13:12]	sbf0 command flag	00,11: No operation 01: buffer fill command 10: buffer flush command
[11:10]	—	Unused (reading returns zero)
[9:8]	sbf1 command flag	* Similar to the description of 'sbf0 command flag'
[7:5]	—	Unused (reading returns zero)
4	Sbf0 interrupt clear command	0: no operation 1: clear SBF0 interrupt flag command
[3:1]	—	Unused (reading returns zero)
0	Sbf1 interrupt clear command	0: no operation 1: clear SBF1 interrupt flag command

Register 14: SBFSTAT

Sequential Buffer Status Register

Reset Value: 0000h

BASE + 44h		Mode: read only. Writing does not affect this register.
Bit	Name	Description
[15]	-	Unused (reading returns zero)
[14:12]	sbf0 state flag	000: Sequential buffer 0 is in non-sequential access mode 001: Sequential buffer 0 is being filled 010: Sequential buffer 0 is being flushed 100: Sequential buffer 0 is in initialized mode 101: Sequential buffer 0 is in sequential read mode 110: Sequential buffer 0 is in sequential write mode others: undefined
[11]	-	Unused (reading returns zero)
[10:8]	sbf1 state flag	* Similar to the description of 'sbf0 state flag'
[7:5]	-	Unused (reading returns zero)
4	Sbf0 interrupt flag	It is set when offset address wrapping occurs in SBF0 in sequential ring buffer mode. It is cleared when ADM commands 'clear SBF0 interrupt flag'.
[3:1]	-	Unused (reading returns zero)
0	Sbf1 interrupt flag	* Similar to the description of 'sbf0 interrupt flag'

Register 15: SBL0OFF

SBL0OFF_H: Higher bits of Offset register of sequential block 0 area

Reset Value: 0000h

BASE + 46h		Mode: read/write
Bit	Name	Description
[15:2]	-	Unused (reading returns zero)
[1:0]	sbl0off_h	Higher 2 bits of 18-bit sequential block 0 offset (SBL0OFF[17:16])

SBL0OFF_L: lower bits of Offset register of sequential block 0 area

Reset Value: 0000h

BASE + 48h		Mode: read/write
Bit	Name	Description
[15:2]	sbl0off_l	Middle 14 bits of 18-bit sequential block 0 offset (SBL0OFF[15:2])
[1]	sbl0off_1	SBL0OFF[1]. It is fixed to 0 when SBF0 is working in 32-bit access mode.
[0]	sbl0off_0	SBL0OFF[0]. It is fixed to 0.

NOTE: When SBL0OFF is written SBL0OFF[1:0] is set as '00' because the boundary of SBL0 was fixed to be word (32-bit data) aligned for efficient off-chip memory access.

Register 16: SBL1OFF

SBL1OFF_H: Higher bits of Offset register of sequential block 1 area

Reset Value: 0000h

BASE + 4Ah		Mode: read/write
Bit	Name	Description
[15:2]	-	Unused (reading returns zero)
[1:0]	Sbl1off_h	Higher 2 bits of 18-bit sequential block 1 offset (SBL1OFF[17:16])

SBL1OFF_L: lower bits of Offset register of sequential block 1 area

Reset Value: 0000h

BASE + 4Ch		Mode: read/write
Bit	Name	Description
[15:2]	Sbl1off_l	Middle 14 bits of 18-bit sequential block 1 offset (SBL1OFF[15:2])
[1]	Sbl1off_1	SBL1OFF[1]. It is fixed to 0 when SBF1 is working in 32-bit access mode.
[0]	Sbl1off_0	SBL1OFF[0]. It is fixed to 0.

NOTE: When SBL1OFF is written SBL1OFF[1:0] is set as '00' because the boundary of SBL1 was fixed to be word (32-bit data) aligned for efficient off-chip memory access.

Register 17: SBL0BEGIN

SBL0BEGIN_H: Higher bits of Begin Offset of sequential block 0 area in ring mode

Reset Value: 0000h

BASE + 4Eh		Mode : read/write
Bit	Name	Description
[15:2]	-	Unused (reading returns zero)
[1:0]	sbl0begin_h	Higher 2 bits of 18-bit sequential block 0 Begin Offset (SBL0BEGIN[17:16])

SBL0BEGIN_L: lower bits of Begin Offset of sequential block 0 area in ring mode

Reset Value: 0000h

BASE + 50h		Mode: read/write
Bit	Name	Description
[15:2]	sbl0begin_l	Middle 14 bits of 18-bit sequential block 0 Begin Offset (SBL0BEGIN[15:2])
[1:0]	sbl0begin_10	Lower 2 bits of 18-bit sequential block 0 Begin Offset (SBL0BEGIN[1:0]). These are fixed to 00.

Register 18: SBL1BEGIN

SBL1BEGIN_H: Higher bits of Begin Offset of sequential block 1 area in ring mode Reset Value: 0000h

BASE + 52h		Mode: read/write
Bit	Name	Description
[15:2]	-	Unused (reading returns zero)
[1:0]	Sbl1begin_h	Higher 2 bits of 18-bit sequential block 1 Begin Offset (SBL1BEGIN[17:16])

SBL1BEGIN_L: lower bits of Begin Offset of sequential block 1 area in ring mode Reset Value: 0000h

BASE + 54h		Mode: read/write
Bit	Name	Description
[15:2]	Sbl1begin_l	Middle 14 bits of 18-bit sequential block 1 Begin Offset (SBL1BEGIN[15:2])
[1:0]	Sbl1begin_10	Lower 2 bits of 18-bit sequential block 1 Begin Offset (SBL1BEGIN[1:0]). These are fixed to 00.

Register 19: SBL0END

SBL0END_H: Higher bits of End Offset of sequential block 0 area in ring mode Reset Value: 0000h

BASE + 56h		Mode: read/write
Bit	Name	Description
[15:2]	-	Unused (reading returns zero)
[1:0]	sbl0end_h	Higher 2 bits of 18-bit sequential block 0 End Offset (SBL0END[17:16])

SBL0END_L: lower bits of End Offset of sequential block 0 area in ring mode Reset Value: 0000h

BASE + 58h		Mode: read/write
Bit	Name	Description
[15:2]	sbl0end_l	Middle 14 bits of 18-bit sequential block 0 End Offset (SBL0END[15:2])
[1:0]	sbl0end_10	Lower 2 bits of 18-bit sequential block 0 End Offset (SBL0END[1:0]). These are fixed to 00.

Register 20: SBL1END

SBL1END_H: Higher bits of End Offset of sequential block 1 area in ring mode Reset Value: 0000h

BASE + 5Ah		Mode: read/write
Bit	Name	Description
[15:2]	-	Unused (reading returns zero)
[1:0]	Sbl1end_h	Higher 2 bits of 18-bit sequential block 1 End Offset (SBL1END[17:16])

SBL1END_L: lower bits of End Offset of sequential block 1 area in ring mode Reset Value: 0000h

BASE + 5Ch		Mode: read/write
Bit	Name	Description
[15:2]	Sbl1end_l	Middle 14 bits of 18-bit sequential block 1 End Offset (SBL0END[15:2])
[1:0]	Sbl1end_10	Lower 2 bits of 18-bit sequential block 1 End Offset (SBL0END[1:0]). These are fixed to 00.

SYSTEM PROGRAMMING MODEL

STATE CONTROLS

Figure 4-6 shows main state diagram of ADM. Host can on, off and reset ADM by setting flags of ADM_CONFIG register in SFRs in ADM. ADM can go to IDLE state by itself if Calm issues 'sys idle' command. In 'Core Stop State' and 'Core Idle State', the clocks can be stopped or Host can update SFRs in ADM. In 'Core Run State', stopping clock and updating SFRs in ADM are not recommended.

COMMUNICATION CHANNELS

Host and ADM communicate with each other through four channels, download, upload, Host event and ADM event. Each channel consists of command-interrupt pairs. Through download and upload channels, Host and ADM can send large communication data to each other.

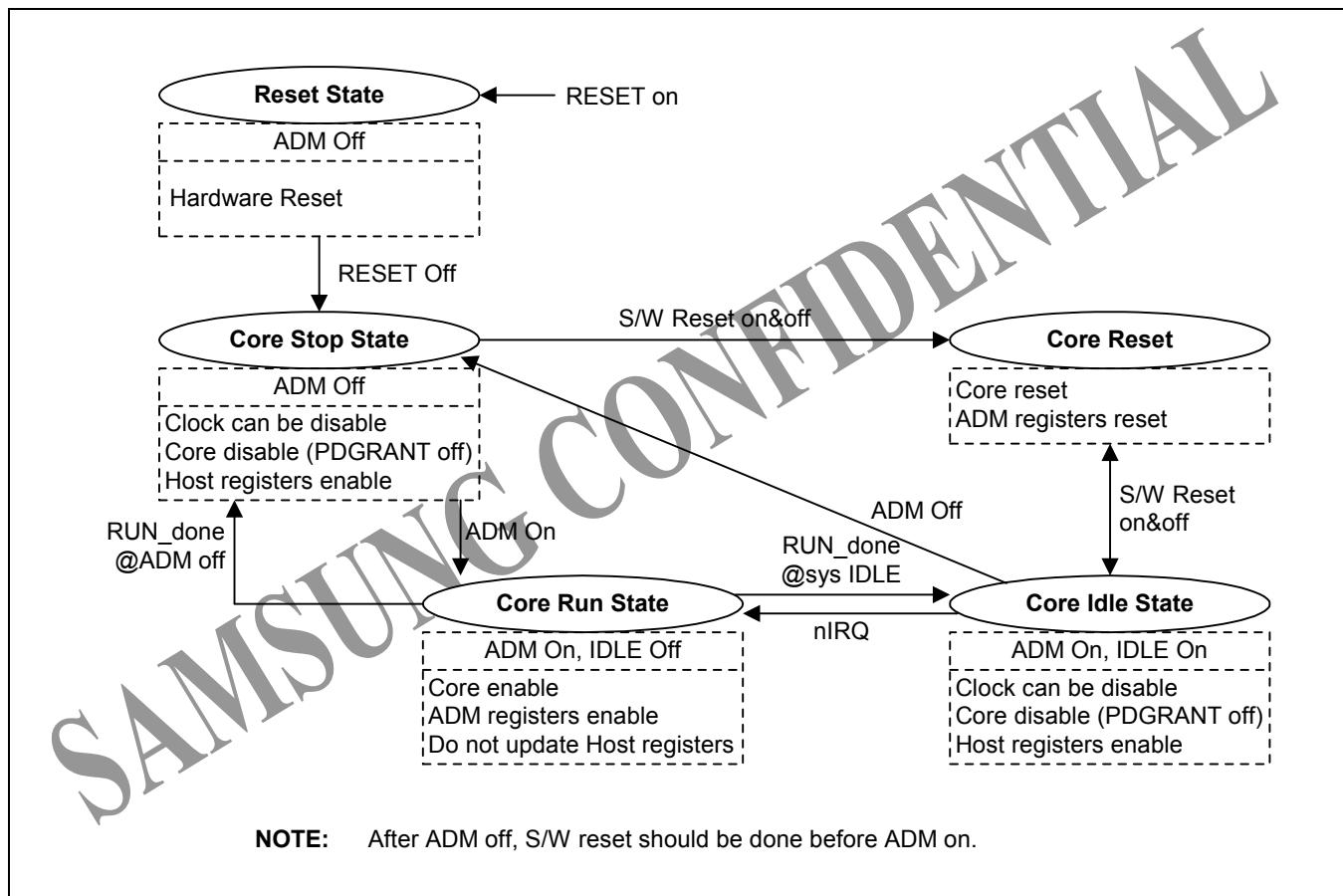


Figure 4-6. CalmADM2E State Diagram

CHANNELS

4 Channels and 6 Channel Commands

Download channel is used to transfer communication data from Host to ADM. Host writes communication data into download register and sends 'new download' command to ADM. ADM reads download data and sends 'download complete' command to Host.

Upload channel is used to transfer communication data from ADM to Host. ADM writes communication data into upload register and sends 'new upload' command to Host. Host reads upload data and sends 'upload complete' command to ADM.

Host event channel and ADM event channel do not transfer any communication data. Host issues 'host event' command so that ADM can execute pre-defined Host event service routine. ADM issues 'adm event' command so that Host can execute pre-defined ADM event service routine.

6 Channel Interrupts

When one of three Host channel commands issued, its corresponding interrupt flag is set. ADM can indicate that a command is issued by taking the flag as an interrupt or by polling the interrupt flags.

When one of three ADM channel commands issued, its corresponding interrupt flag is set. Host can indicate that a command is issued by taking the flag as an interrupt or by polling the interrupt flags.

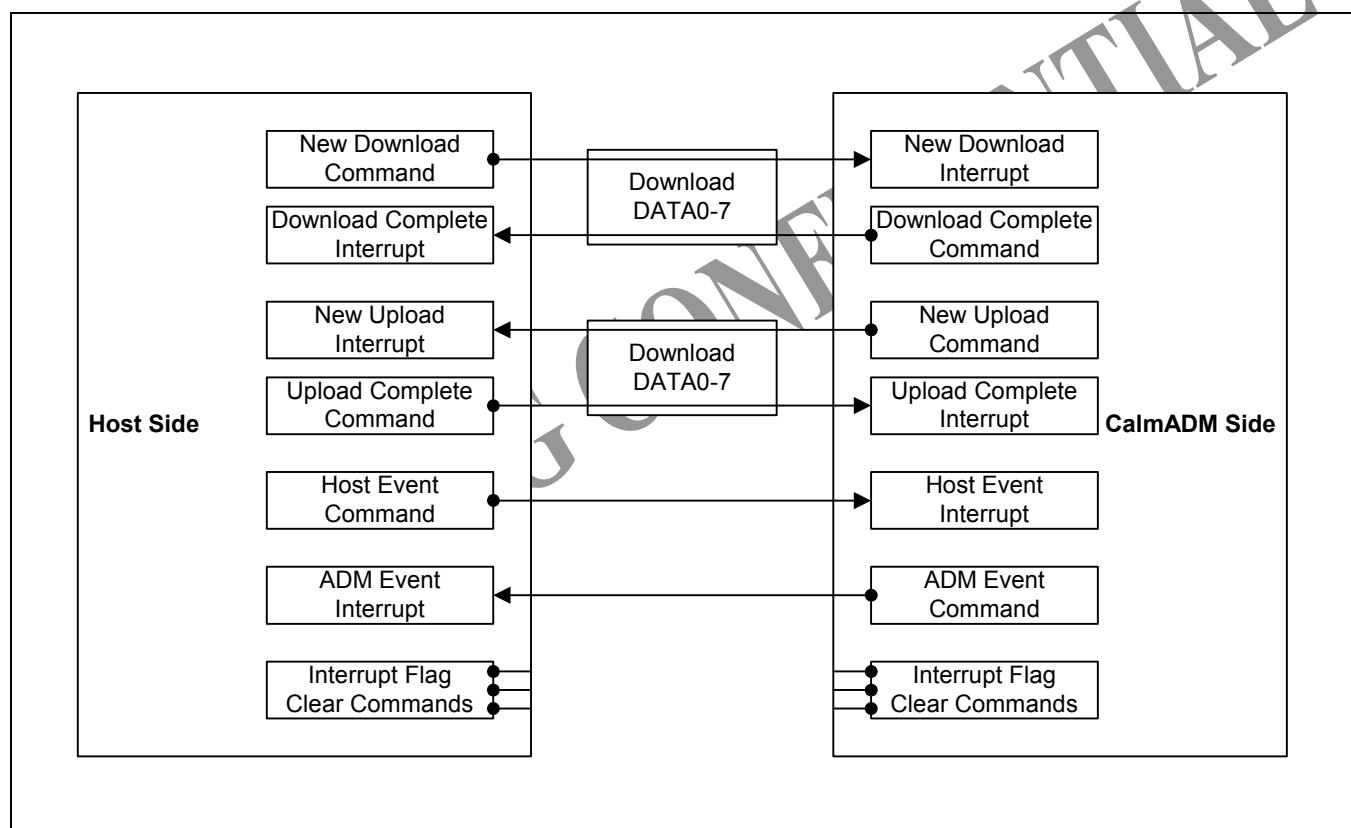


Figure 4-7. Host-ADM Communications

6 Clear Commands

Host issues one of 3 'clear interrupt flag' commands to reset corresponding interrupt flag set by ADM channel command.

ADM issues one of 3 'clear interrupt flag' commands to reset corresponding interrupt flag set by Host channel command.

Channel Operations

Table 4-6. Typical Channel Operation Sequence

Channel	Steps
Download	Host writes communication data into download register.
	Host sets host command flag as new download. New download interrupt flag is set automatically.
	ADM indicates new download requested by taking interrupt or polling the interrupt flags. ADM may clear the interrupt flag by asserting 'clear new download interrupt flag' command.
	ADM reads download register.
	ADM sets ADM command flag as download complete. Download complete interrupt flag is set automatically.
	Host indicates download completed by taking interrupt or polling the interrupt flags. Host may clear the interrupt flag by asserting 'clear download complete interrupt flag' command.
	ADM runs in accordance with communication data in download register.
Upload	* Similar to download sequence
Host Event	Host sets host command flag as host event. Host event interrupt flag is set automatically.
	ADM indicates host event occurred by taking interrupt or polling the interrupt flags. ADM may clear the interrupt flag by asserting 'clear host event interrupt flag' command.
	ADM executes pre-defined host event service routine.
ADM Event	* Similar to host event sequence

HARDWARE COMPONENTS

Communication Data Registers

128-bit download register contains communication data for download channel. It is write-only to Host and read-only to ADM. It is mapped on eight 16-bit registers in I/O area.

128-bit upload register contains communication data upload channel. It is write-only to ADM and read-only to Host. It is mapped on eight 16-bit registers in I/O area.

Command Flags

To represent six Host commands, two for up and down load download and one for Host event and three for interrupt flag reset, 3-bit Host command flag is defined in SFRs in ADM, ADM_COMMUN. Because writing a code into this flag is asserting a command, physical registers are not needed for this flag. This flag is write-only. Reading it returns all zero.

To represent six ADM commands, two for up and down load and one for ADM event and three for interrupt flag reset, 3-bit ADM command flag is defined in AFRS, COMMUN. Because writing a code into this flag is asserting a command, physical registers are not needed for this flag. This flag is write-only. Reading it returns all zero.

Interrupt Flags

Three interrupt flags are defined in SFRs in ADM, ADM_COMMUN. Each flag corresponds to one of three ADM commands - download complete, new upload and ADM event. An interrupt flag is set when ADM issues corresponding command and is reset when Host issues its clear command. These are read-only. Writing them results nothing.

Three interrupt flags are defined in AFRS, COMMUN. Each flag corresponds to one of three Host commands - new download, upload complete and Host event. An interrupt flag is set when Host issues corresponding command and is reset when ADM issues its clear command. These are read-only. Writing them results nothing.

Interrupt Enable Flags

Three interrupt enable flags are defined in SFRs in ADM, ADM_CONFIG. Each flag corresponds to one of three interrupt flags in ADM_COMMUN. When an interrupt flag is set, it causes interrupt request to Host only while its corresponding interrupt enable flag is on.

Three interrupt enable flags are defined in AFRS, CONFIG0. Each flag corresponds to one of three interrupt flags in COMMUN. When an interrupt flag is set, it causes interrupt request to ADM only while its corresponding interrupt enable flag is turned on.

SPECIAL FUNCTION REGISTERS IN ADM

Following registers are components of SFRS of the target system. These registers are addressed with 7bit off set address. A full address value is an addition of an off set address and the base address. The base address is target system dependent.

In S5L8700X and its followers, the base address was defined as **0x3900_0000**.

Register 1: ADM_CONFIG

Configuration/Control Register

Reset Value: 00000002h

BASE + 0h		Mode: read/write
Bit	Name	Description
31:7	–	Unused (reading returns zero)
6	ADM Event Interrupt Enable	If 1, ADM event interrupt generation is allowed.
5	Download Complete Interrupt Enable	If 1, download complete interrupt is generated when download is completed.
4	New Upload Interrupt Enable	If 1, upload request interrupt is generated when a new upload request is generated.
3	–	Unused (reading returns zero)
2	ADM swrst	<p>Software reset of ADM.</p> <p>0: No reset 1: Reset. Two cores and all registers in ADM, except Host Control Registers, are reset to their hardware-reset status. After reset is done, this bit will be de-asserted automatically.</p> <p>Software reset should be asserted in ADM off state. If it is asserted when ADM on, correct completion of the instructions in pipeline is not guaranteed.</p>
1	ADM ready_clk_down	<p>Read only status bit (write operation does not affect this bit)</p> <p>0: ADM is running 1: ADM is ready to accept “no clock”.</p> <p>If host disables ADM (clearing ADM on bit), ADM finishes all pending actions including external memory access, and set this flag high to let host know it is ready to accept “no clock”.</p> <p>If host enables ADM (set ADM on bit), ADM reset this flag.</p>
0	ADM on	<p>0 : ADM is disabled 1 : ADM is enabled.</p> <p>ADM cannot be stop-and-resumed. Once, Host disables ADM, It should be reset either by software or by hardware before Host enables ADM.</p>

Register 2: ADM_COMMUN

Communication Control Register

Reset Value: 00000000h

BASE + 4h		Mode: read/write
Bit	Name	Description
31:7	-	Unused (reading returns zero)
6	ADM Event Interrupt Flag	Read only bit. Writing does not affect this bit. It is set when ADM generates ADM event. It is cleared when host commands 'clear ADM event interrupt flag'.
5	Download Complete Interrupt Flag	Read only bit. Writing does not affect this bit. It is set when ADM has done a download transaction. It is cleared when host commands 'new download' or 'clear download complete interrupt flag'.
4	New Upload Interrupt Flag	Read only bit. Writing does not affect this bit. It is set when ADM requests a new upload transaction. It is cleared when host commands 'upload complete' or 'clear new upload interrupt flag'.
3	-	Unused (reading returns zero)
2:0	Host Command Flag	These bits are write-only. Reading returns zero. 000 : new download 001 : upload complete 01x : host event generation 100 : clear new upload interrupt flag 101 : clear download complete interrupt flag 11x : clear ADM event interrupt flag

Register 3: ADM_DDATA0 ~ ADM_DDATA7

Download Data Registers

Reset Value: xxxxxxxxh

BASE + 10h ~ 2Ch		Mode: read/write
Bit	Name	Description
[31:16]	-	Unused (reading returns zero)
[15:0]	Download Data	Communication data from host to ADM

Register 4: ADM_UDATA0 ~ ADM_UDATA7

Upload Data Registers

Reset Value: xxxxxxxxh

BASE + 30h ~ 4Ch		Mode: read
Bit	Name	Description
[31:16]	-	Unused (reading returns zero)
[15:0]	upload data	Communication data from ADM to host

Register 5: ADM_IBASE

Start Address for ADM Instruction Area

Reset Value: xxxxxxxxh

BASE + 50h		Mode: read/write
Bit	Name	Description
[31:4]	adm_ibase	Higher 28 bits of CalmRISC instruction area base address
[3:0]	–	Lower 4 bits of CalmRISC instruction area base address, fixed to 0

Register 6: ADM_DBASE

Start Address for CalmRISC Data Area

Reset Value: xxxxxxxxh

BASE + 54h		Mode: read/write
Bit	Name	Description
[31:4]	adm_dbase	Higher 28 bits of CalmRISC data area base address
[3:0]	–	Lower 4 bits of CalmRISC data area base address, fixed to 0

Register 7: ADM_XBASE

Start Address for Mac X Area

Reset Value: xxxxxxxxh

BASE + 58h		Mode: read/write
Bit	Name	Description
[31:10]	adm_xbase	Higher 22 bits of Mac X area base address
[9:0]	–	Lower 10 bits of Mac X area base address, fixed to 0

Register 8: ADM_YBASE

Start Address for Mac Y Area

Reset Value: xxxxxxxxh

BASE + 5Ch		Mode: read/write
Bit	Name	Description
[31:10]	adm_ybase	Higher 22 bits of Mac Y area base address
[9:0]	–	Lower 10 bits of Mac Y area base address, fixed to 0

Register 9: ADM_S0BASE

Start Address for Sequential Block 0 Area

Reset Value: xxxxxxxxh

BASE + 60h		Mode: read/write
Bit	Name	Description
[31:4]	adm_s0base	Higher 28 bits of SB0 area base address
[3:0]	–	Lower 4 bits of SB0 area base address, fixed to 0

Register 10: ADM_S1BASE

Start Address for Sequential Block 1 Area

Reset Value: xxxxxxxxh

BASE + 64h		Mode: read/write
Bit	Name	Description
[31:4]	adm_s1base	Higher 28 bits of SB1 area base address
[3:0]	-	Lower 4 bits of SB1 area base address, fixed to 0

SAMSUNG CONFIDENTIAL

HARDWARE DESCRIPTIONS

This section contains short descriptions about the components of ADM. Calm, Mac, DBU (debugger unit) are not mentioned here because technical manuals of them are available.

PIN DESCRIPTIONS

Table 4-7. CalmADM2E Pin Description

Group	Signal	Direction	Polarity	Description
System Interface	MCLK	I	Rising	Processor clock input
	MCLKO	O	Rising	Processor clock output to be used as the clock of cache memories
	HCLK	I	Rising	Bus clock
	HRESETN	I	Low	Reset
	IRQ2ARM	O	High	Interrupt request for Host used for issuing a command from ADM to Host
	FAST_BUS_MODE	I	High	Bus operation mode selection 1: adm is in fast_bus_mode. HCLK is used as the main clock. MCLK is unused. 0: adm is in async_bus_mode. HCLK works as the bus clock only. MCLK works as the main processor clock. MCLK should be faster than HCLK.
AHB+ Master Interface	HBUSREQM	O	High	
	HGRANTM	I	High	
	HREADYM	I	High	
	HRESPM [1:0]	I	-	
	HADDRM [31:0]	O	-	
	HTRANSM [1:0]	O	-	
	HBURSTM [3:0]	O	-	
	HSIZEM [2:0]	O	-	
	HWRITEIM	O	High	
	HWDATAM[31:0]	O	-	
	HRDATAM[31:0]	I	-	

Table 4-7. CalmADM2E Pin Description (Continued)

Group	Signal	Direction	Polarity	Description
AHB+ Slave Interface	HSELS	I	High	
	HREADYS	I	High	
	HADDRS [31:0]	I	-	
	HTRANSS [1:0]	I	-	
	HWRITES	I	High	
	HWDATAS[31:0]	I	-	
	HRDATAS[31:0]	O	-	
	HREADYOUTS	O	High	
Debug Interface	nTRST	I	Low	
	TCK	I	Rising	
	TMS	I	-	
	TDI	I	-	
	TDO	O	-	
Scan Test Interface	SCAN_TEST_MODE	I	High	
	SCAN_ENABLE	I	High	
	SCAN_IN	I	-	
	SCAN_OUT	O	-	
Memory Interface	IC_DRAM_A[9:0]	O	-	
	IC_DRAM_CSN	O	Low	
	IC_DRAM_WEN	O	Low	
	IC_DRAM_DI[31:0]	O	-	
	IC_DRAM_DO[31:0]	I	-	
	IC_TRAM_A[6:0]	O	-	
	IC_TRAM_CSN	O	Low	
	IC_TRAM_WEN	O	Low	
	IC_TRAM_BWEN[10:0]	O	Low	
	IC_TRAM_DI[10:0]	O	-	
	IC_TRAM_DO[10:0]	I	-	
	XC_DRAM_A[8:0]	O	-	
	XC_DRAM_CSN	O	Low	
	XC_DRAM_WEN	O	Low	
	XC_DRAM_BWEN[95:0]	O	Low	
	XC_DRAM_DI[95:0]	O	-	
	XC_DRAM_DO[95:0]	I	-	
	XC_TRAM_A[6:0]	O	-	
	XC_TRAM_CSN	O	Low	

Table 4-7. CalmADM2E Pin Description (Continued)

Group	Signal	Direction	Polarity	Description
Memory Interface	XC_TRAM_WEN	O	Low	
	XC_TRAM_BWEN[23:0]	O	Low	
	XC_TRAM_DI[23:0]	O	-	
	XC_TRAM_DO[23:0]	I	-	
	YC_DRAM_A[8:0]	O	-	
	YC_DRAM_CSN	O	Low	
	YC_DRAM_WEN	O	Low	
	YC_DRAM_BWEN[95:0]	O	Low	
	YC_DRAM_DI[95:0]	O	-	
	YC_DRAM_DO[95:0]	I	-	
	YC_TRAM_A[6:0]	O	-	
	YC_TRAM_CSN	O	Low	
	YC_TRAM_WEN	O	Low	
	YC_TRAM_BWEN[23:0]	O	Low	
	YC_TRAM_DI[23:0]	O	-	
	YC_TRAM_DO[23:0]	I	-	

ARBITER

Features

Prioritized scheduling of bus requests from 8 sources

SBF0 (the highest priority)

> SBF1

> X-Cache

> Y-Cache

> I-Cache

> X-Cache Write-Back

> Y-Cache Write-Back (the lowest priority)

AHB+ INTERFACE UNITS

Overview

ADM is designed as an IP module on AHB+ working as both master and slave of AHB+. When ADM accesses system memory, AHBMIU performs AHB+ master access. When the host CPU accesses SFRs in ADM, AHBSIU performs AHB slave access.

Features

AHBMIU (AHB+ master interface unit) supports

- Both asynchronous and synchronous AHB+ master interfaces for system memory access
- Eight burst word access to system memory for I-Cache
- Six burst word access to system memory for X/Y-Caches
- Single burst half-word and byte access to system memory for cache bypass mode.
- One ~ four burst word access for sequential accesses to sequential blocks
- Single burst half-word access for random accesses to sequential blocks

AHBSIU (AHB+ slave interface unit) supports

- Both asynchronous and synchronous AHB+ slave interfaces for host interface
- Write and read of system function registers by Host processor that results
 - On/off/reset of ADM
 - Communications between Host and ADM
 - Setting base addresses

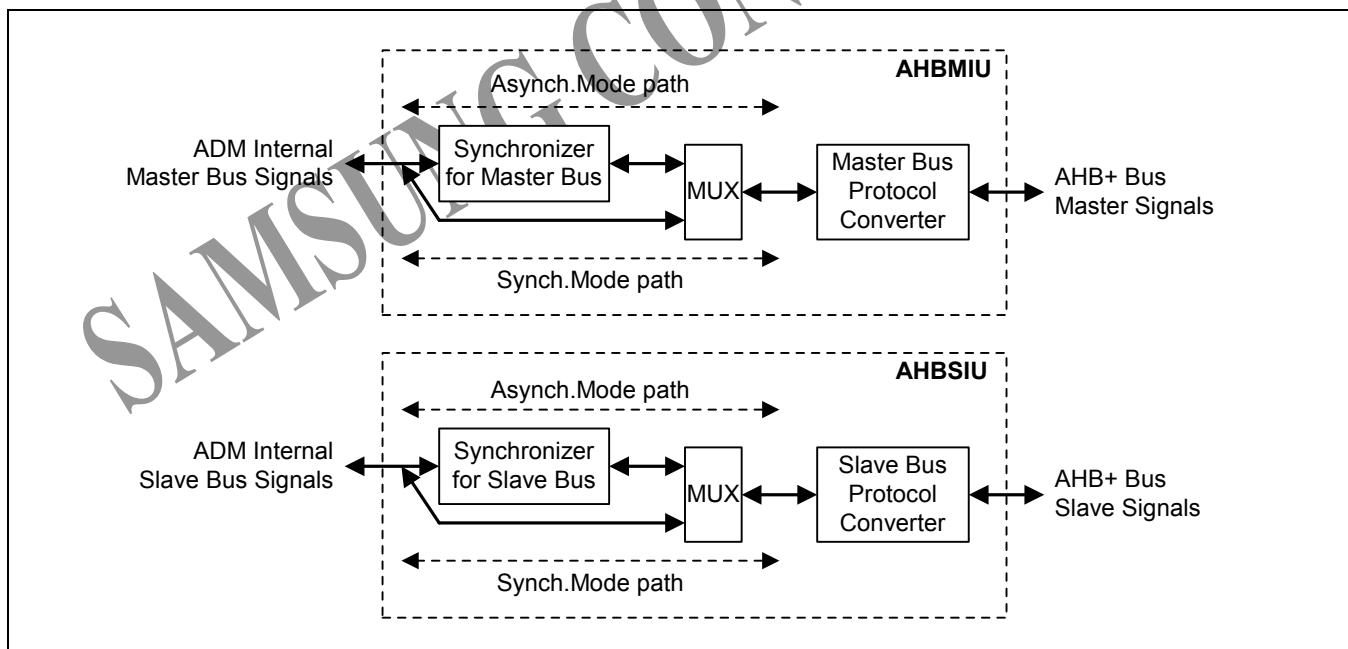


Figure 4-8. Structure of AHB+ Interface Units

INSTRUCTION CACHE

Overview

I-Cache is a direct-mapped, 256x16-instruction size cache. Three commands, on, off and all invalidation, are supported for normal cache function. Before I-Cache on, all invalidation command should be done for valid cache operation. I-Cache also supports one-line invalidation function for debugging convenience. If Calm performs LDC instruction while 'ldcinv' flag in CONFIG1 register is on, one I-Cache line selected by program address of Calm is invalidated. Since I-Cache performs caching with virtual address (Calm program memory address), it is need to convert to physical address (Host memory address). This conversion is done inside of I-Cache by adding cache address with instruction cache base register (ADM_IBASE).

Features

- Direct-mapped cache
- **128 cache data entries** (Cache Lines)
- 256 bit wide Cache Data Memory (16*16-bit instructions in a Cache Line)
- 11 bit wide Cache Tag Memory (9 bits for address, 1 bit for valid bit)
- 32 bit Base Register for mapping to physical address
- Supports all invalidation and one line invalidation

DATA CACHES

In ADM are two data caches. One is X-Cache caching data in MAC X area. The other is Y-Cache caching data in MAC Y area. Since, X-Cache and Y-Cache are exactly same, only X-Cache is described more detail in this section. These two data caches also work as the data cache for Calm as a kind of 4-way set-associative cache.

X-Cache

Overview

X-Cache is a 2-way set associative cache with a set sized 128x8-data. Four commands (on, off, invalidation and flush) are supported for normal cache function. Before X-Cache on, all-validation command should be done for valid cache operation. After X-Cache off, flush command should be done where data consistency should be kept. Since X-Cache performs caching with virtual address (MAC X memory address), it is need to be converted to physical address (Host memory address). The physical address is an addition of 4:3-reduced virtual address with X-area base register (ADM_XBASE). X-Cache works as not only the cache for MAC X area data accessed by both Calm and Mac but also the cache for Calm area data accessed by Calm.

Features

- 2-way set associative cache
- 128 data entries (cache lines) in a set
- 8 aud-words (192 bit) in a cache line
- Only 128 bits of a cache line are used as Calm area cache
- 24-bit wide Cache Tag Memory ((11 bits for tag address, 1 bit for valid bit) * 2 ways)
- 32-bit Base Register for mapping to physical address
- Supports invalidation and flushing.

X/Y-CACHE AS CALM AREA DATA CACHE

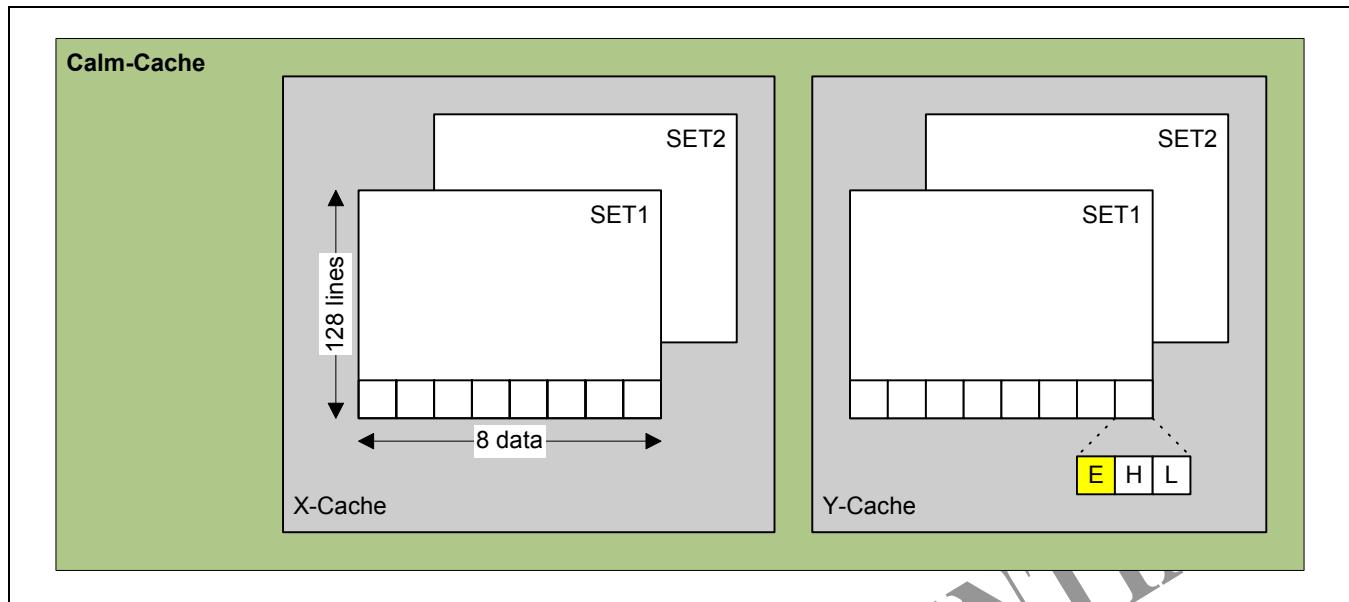


Figure 4-9. Data Cache Structure

Overview

X/Y-Caches work as not only the cache for MAC X/Y area data accessed by both Calm and Mac but also the cache for Calm area data accessed by Calm. By setting XYRR code in CONFIG1 register properly, X- and Y-Cache can be enabled/disabled as the cache of Calm. Since X/Y-Caches perform caching with virtual address (Calm data memory address), virtual address to physical address (Host memory address) conversion is needed. This conversion is done inside of X/Y-Cache by adding cache address with instruction cache base register (ADM_DBASE).

Operations

When X/Y-Caches work as the cache for Calm area data, the operation depends on the value of XYRR bits in CONFIG1 register.

- When X-Cache is selected (XYRR == 01)
 - Calm area data are accessed through X-Cache block whether its cache function is enabled or not.
- When Y-cache is selected (XYRR == 1x)
 - Calm area data are accessed through Y-Cache block whether its cache function is enabled or not.
- When Round robin scheme is selected (XYRR == 00)
 - If both of X- and Y- Caches are enabled, X- and Y-Cache are selected for cache data replacement one after another (round robin). At the first miss, X-Cache is selected for replacement. When next miss occurred, Y-Cache is selected for replacement. At the next miss, X-cache is selected, and so on.
 - If both of X- and Y- Caches are disabled, Calm area data are accessed through Y-Cache block.
 - If one of X- and Y- Caches is enabled and the other one is disabled, Calm area data are accessed through enabled Cache block.

Architecture

When X/Y-Caches work as the cache for Calm area data, the architecture of the cache is depends on the value of XYRR bits in CONFIG1 register. Following description is for the case when both X- and Y- caches are enabled.

— When one of X- or Y-Cache is selected (XYRR == 01 or XYRR == 1x)

- 4K byte, 2-way set associative cache
- 128 data entries (cache lines) in a set
- 8 half-words (128 bit) in a cache line

— When Round robin scheme is selected (XYRR == 00)

- 8K byte, 4-way set associative cache
- 128 data entries (cache lines) in a set
- 8 half-words (128 bit) in a cache line

SAMSUNG CONFIDENTIAL

SEQUENTIAL BUFFER UNITS

Overview

In ADM, two sequential buffers are supported, SBF0 and SBF1. A sequential buffer consists of a 128-bit FIFO, an 18-bit Offset register, two 18-bit boundary-offset registers and a 16-bit data register. 128-bit FIFO is used as buffer when Mac sequentially accesses sequential block. Access mode, unit data size and data size conversion scheme in a SBU are defined in CONFIG0 register. 18-bit Offset register contains offset address of sequential block data to be accessed. Since it is defined as one of control registers, user can write the start address of sequential block into this register. This register is added with sequential block area base register (ADM_S0BASE/ADM_S1BASE) to generate physical address of sequential data. After sequential access to sequential block, this register value is increased automatically. In ring mode, auto-incremented register value is compared with End Offset register value. If these are same, Offset register is set as the value of Begin Offset register and interrupt flag in SBFSTAT register is set. A 16-bit data register is used as buffer when Calm randomly accesses sequential block. Two special commands (fill and flush commands) are supported for a sequential buffer. Before user starts sequential read on sequential block, fill command should be done. Flush command should be done after sequential writes end. More of SBU operation is described in Section 'Sequential Data Access'.

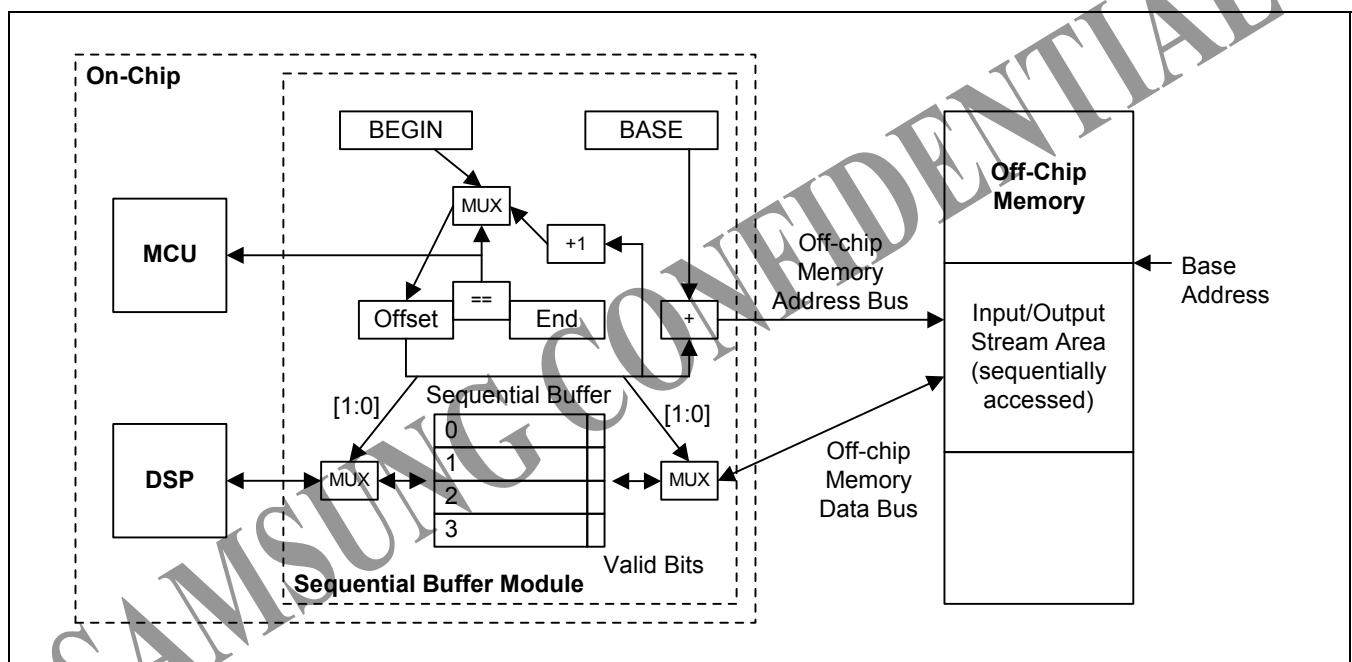


Figure 4-10. Sequential Buffer Function Diagram

Features

- 4-word FIFO used as a sequential buffer
- 18-bit offset address register
- 16-bit data buffer for random access to sequential block area
- Fill and Flush commands
- Unit data size and data size conversion scheme selected by setting control flags.
- Two access modes supported (linear mode and ring mode)
- Two boundary-offset registers (Begin Offset and End Offset) are used to access a sequential block as a ring buffer.

APPLICATION GUIDES

ON CONTROLLING ADM

Turning Off and On ADM

As mentioned in Figure 4-6, S/W Reset command should be issued to ADM before you turn on ADM. Therefore when you turn off and turn on ADM, following sequence is recommended.

```
reset adm_on flag of ADM_CONFIG register  
check if adm_ready_clk_down flag of ADM_CONFIG register is on  
set adm_swrst flag of ADM_CONFIG register  
<wait until ADM is needed to work>  
set adm_on flag of ADM_CONFIG register
```

Stopping Clocks

For the purpose of power saving, we can stop the clocks. In ‘Core Stop State’ and ‘Core Idle State’ in Figure 4-6, we can stop the clocks. In ‘Core Run State’, stopping clocks are not recommended. When you stop and resume the clocks, you should follow the sequences below.

In Fast_Bus_mode:

```
force ADM into ‘Core Stop State’ or ‘Core Idle State’  
check if adm_ready_clk_down flag of ADM_CONFIG register is on  
stop toggling HCLK  
<wait until ADM is needed to work>  
resume toggling HCLK  
force ADM into ‘Core Run State’
```

In Async_Bus_mode:

```
force ADM into ‘Core Stop State’ or ‘Core Idle State’  
check if adm_ready_clk_down flag of ADM_CONFIG register is on  
stop toggling HCLK  
stop toggling MCLK  
<wait until ADM is needed to work>  
resume toggling MCLK  
resume toggling HCLK  
force ADM into ‘Core Run State’
```

Updating SFRs in ADM

Before you update a register in SFRs in ADM, forcing ADM into ‘Core Stop State’ and ‘Core Idle State’ in Figure 4-6 is recommended. In a general way, updating SFRs in ADM while ADM is in ‘Core Run State’ is not recommended.

ON PROGRAM SEQUENCE

Constraints on CalmRISC16E Program

Delay slot instructions

As an instruction at delay slot, following three types of instructions are prohibited.

- Break instruction
- Branch instructions
- Two word instructions

Constraints on CalmMAC24E Program

Loading RAM Pointer

Because of the nature of pipeline scheme and data memory accessing scheme of Mac, data memory accessing with the RAM pointer, that is loaded from the outside of Mac just before, is prohibited. An ENOP instruction or another instruction should be inserted between RAM pointer load instruction and data memory access instruction using that RAM pointer as data address. Instructions loading RAM pointer are listed in Table 4-8.

Example code

```
ELD RP0, RPD1.0    ; load RAM pointer
ENOP                  ; inserted ENOP
ELD A, @RP0          ; using loaded RAM pointer as data address
```

Table 4-8. List of Instructions Loading RAM Pointers

opc	op1	op2	Function	Flag
ELD	rpui	rpd1.adr:2	op1<-op2	-

NOTE: opc— opcode, opi— operand i

Accessing MIN/MAX data

For easy searching MIN/MAX data, Mac offers special instructions, EMIN and EMAX. The example code below shows how to search MIN/MAX data with EMIN/EMAX instructions. After the execution of the code, the address of MIN/MAX data is latched in RP3. Because of the nature of pipeline scheme and data memory accessing scheme of Mac, data memory accessing with RP3, that is latched by execution of EMIN/EMAX instruction just before, is prohibited. An ENOP instruction or another instruction should be inserted between EMIN/EMAX instruction and data memory access instruction using RP3 as data address. EMIN/EMAX Instructions are listed in Table 4-9.

Example code

```

ELD C, @RP0+S0      ; 1st data load
Loop_start:
    EMAX A, C, C, @RP0+S0 ; 1st MAX evaluation, 2nd data load
    JP Loop_start
    EMAX A, C           ; last MAX evaluation
    ENOP                ; inserted ENOP
    ELD A, @RP3          ; using RP3 as data address

```

Table 4-9. List of EMIN/EMAX Instructions

opc	op1	op2	op3	op4	Function	Flag
EMAX	Ai	Ci	Ci	@rps	Ai<-max(Ai,Ci), op3<-op4, RP3<-address	V,N,Z,C
EMIN					Ai<-min(Ai,Ci), op3<-op4, RP3<-address	V,N,Z,C

NOTE: opc – opcode, opi- operand i

Branch on EC instructions

Dissimilarly to CalmMAC24F, CalmMAC24E outputs EC[2:0] with one cycle delay. Because of delayed EC[2:0], branch on EC[2:0] instructions cannot follow on a Mac instruction. A Calm instruction should be inserted between a Mac instruction and one of following instructions.

- BRA/BRAD EC0/EC1/EC2 #offset

Example code

```

EMAD MA0, X0Y0      ; any Mac instruction
NOP                 ; inserted Calm instruction
BRA EC1, #10h        ; branch on EC

```

ON ADM OPERATIONS

Constraints on Accessing Sequential Buffers

- DATA alignment

When Calm accesses sequential block area, the data can be byte or half word. When Mac access sequential buffer, the data can be half word aligned or word aligned. However, when fill or flush is performed in sequential buffer, ADM assumes the data is word aligned. This assumption was taken for efficient external memory access. When user accesses a sequential buffer in half word mode, odd number of sequential accesses causes miss-aligned external memory access. There is no hardware to prevent odd number of sequential accesses in ADM. Therefore user program should consider data alignment especially for sequential writes in half word mode. SBL0OFF and SBL1OFF can be half word aligned or word aligned when these are automatically increased. However, these are fixed to be word aligned when these are written. SBL0BEGIN, SBL0END, SBL1BEGIN are SBL1END are fixed to be word aligned.

- Confirm Flush

After a flush command is preformed, it is recommended to check the status flag in SBFSTAT register if the flush has done physically. This status check is not needed in most cases. However, when ADM transmits certain data to Host processor through sequential buffer, data coherence may be corrupted if this confirmation is omitted.

Constraints on accessing caches

- Setting XYRR

The XYRR flags in CONFIG1 register should be set while both X-Cache and Y-Cache are turned off. If the XYRR flags are changed while one or both of X-Cache and Y-Cache are turned on, following data reads and writes may be performed incorrectly. Because of that, the data in off-chip memory may corrupt.

- Waiting Completion of Commands

After issuing a special command (invalidation or flushing) on a cache, you should test the value of corresponding cache state flags in AFRS. We recommend waiting completion of the command before going to the next step.

DEFINITIONS OF ABBREVIATIONS

Instruction tables in this chapter are extracted from “Quick Reference*” instruction table in “CalmMAC24 DSP Coprocessor Architecture Reference Manual”. Definitions of abbreviations used in instruction tables are listed following tables. This tables are copy of “Instruction Coding”, (1) Abbreviation Definition and Encoding” of “CalmMAC24 DSP Coprocessor Architecture Reference Manual”.

rps

Mnemonic	Encoding	Description
RP0+S0	000	RP0 post-modified by SD0 S0 field
RP1+S0	001	RP1 post-modified by SD1 S0 field
RP2+S0	010	RP2 post-modified by SD2 S0 field
RP3+S0	011	RP3 post-modified by SD3 S0 field
RP0+S1	100	RP0 post-modified by SD0 S1 field
RP1+S1	101	RP1 post-modified by SD1 S1 field
RP2+S1	110	RP2 post-modified by SD2 S1 field
RP3+S1	111	RP3 post-modified by SD3 S1 field

rpd

Mnemonic	Encoding	Description
RP0+D0	000	RP0 post-modified by SD0 D0 field
RP1+D0	001	RP1 post-modified by SD1 D0 field
RP2+D0	010	RP2 post-modified by SD2 D0 field
RP3+D0	011	RP3 post-modified by SD3 D0 field
RP0+D1	100	RP0 post-modified by SD0 D1 field
RP1+D1	101	RP1 post-modified by SD1 D1 field
RP2+D1	110	RP2 post-modified by SD2 D1 field
RP3+D1	111	RP3 post-modified by SD3 D1 field

rp01s

Mnemonic	Encoding	Description
RP0+S0	00	RP0 post-modified by SD0 S0 field
RP1+S0	01	RP1 post-modified by SD1 S0 field
RP0+S1	10	RP0 post-modified by SD0 S1 field
RP1+S1	11	RP1 post-modified by SD1 S1 field

rp3s

Mnemonic	Encoding	Description
RP3+S0	0	RP3 post-modified by SD3 S0 field
RP3+S1	1	RP3 post-modified by SD3 S1 field

mg1

Mnemonic	Encoding	Description
Y0	000	Y0[23:0] register
Y1	001	Y1[23:0] register
X0	010	X0[23:0] register
X1	011	X1[23:0] register
MA0(H)	100	MA0[51:0] / MA0[47:24] register
MA0L	101	MA0[23:0] register
MA1(H)	110	MA1[51:0] / MA1[47:24] register
MA1L	111	MA1[23:0] register

mg2

Mnemonic	Encoding	Description
RP0	000	current bank RP0[15:0] register
RP1	001	current bank RP1[15:0] register
RP2	010	current bank RP2[15:0] register
RP3	011	current bank RP3[15:0] register
RPD0	100	RPD0[15:0] register
RPD1	101	RPD1[15:0] register
MC0	110	MC0[15:0] register
MC1	111	MC1[15:0] register

sdi

Mnemonic	Encoding	Description
SD0	00	current bank SD0[15:0] register (SD0 or SD0E)
SD1	01	current bank SD1[15:0] register
SD2	10	current bank SD2[15:0] register
SD3	11	current bank SD3[15:0] register (SD3 or SD3E)

Ai

Mnemonic	Encoding	Description
A	0	A[23:0] register
B	1	B[23:0] register

Ci

Mnemonic	Encoding	Description
C	0	C[23:0] register
D	1	D[23:0] register

An

Mnemonic	Encoding	Description
A	00	A[23:0] register
B	01	B[23:0] register
C	10	C[23:0] register
D	11	D[23:0] register

rpui

Mnemonic	Encoding	Description
RP0	0000	current bank RP0[15:0] register
RP1	0001	current bank RP1[15:0] register
RP2	0010	current bank RP2[15:0] register
RP3	0011	current bank RP3[15:0] register
MC0_0	0100	MC0[15:0] register (set 0)
MC1_0	0101	MC1[15:0] register (set 0)
MC0_1	0110	MC0[15:0] register (set 1)
MC1_1	0111	MC1[15:0] register (set 1)
SD0_0	1000	current bank SD0[15:0] register (set 0)
SD1_0	1001	current bank SD1[15:0] register (set 0)
SD2_0	1010	current bank SD2[15:0] register (set 0)
SD3_0	1011	current bank SD3[15:0] register (set 0)
SD0_1	1100	current bank SD0[15:0] register (set 1)
SD1_1	1101	current bank SD1[15:0] register (set 1)
SD2_1	1110	current bank SD2[15:0] register (set 1)
SD2_1	1111	current bank SD3[15:0] register (set 1)

mga

Mnemonic	Encoding	Description
MA0	00	MA0[51:0] / MA0[47:24] register
MA1	01	MA1[51:0] / MA1[47:24] register
A	10	A[23:0] register
B	11	B[23:0] register

mgx

Mnemonic	Encoding	Description
Y0	00	Y0[23:0] register
Y1	01	Y1[23:0] register
X0	10	X0[23:0] register
X1	11	X1[23:0] register

INFORMATION FOR CALM SHINE DEVELOPMENT

ASSEMBLER RELATED

Branch on EC instructions

Assembler should check if one of branch instructions below is following on a Mac instruction. This case should be treated as a syntax error. Refer Section 'Application Guides' for more information.

- BRA/BRAD EC0/EC1/EC2 #offset

Simulator Related

Memory Map

Simulator should keep track of memory configuration. Simulator should check if Calm accesses the locations that have no physical memory (Unused area and unused I/O area). Data of these locations are fixed to all zero.

Sequential Buffers

Simulator should keep track of the sequential accesses in terms of the functionality only. Modeling exact hardware structure is not needed.

- Sequential buffer models are not needed in simulator.
- Sequential Offset registers should be modeled in simulator. Being read/written as a function register, addressing sequential block area and auto-increment capability of these registers should be modeled. When written, these registers should be word-aligned. In other words, lower 2 bits of these registers should be forced as zero when Calm writes values to these registers.
- When auto-incremented, the incrementing step should follow the mode defined in CONFIG0 register. In ring mode, equivalence between Offset register and End Offset register should be checked in simulator. In addition, wrapping and flag setting should be done in simulator when those are same.
- Data format mapping defined by SBF0/1 mode flags should be modeled in simulator. SBF0/1 mode flags are part of CONFIG0 register.
- Modeling commands on sequential buffers, defined by SBFCON register, is not needed in simulator.
- Access mode defined by [3] bits of SBF0/1 mode flags should be modeled in simulator.
- Begin Offset registers and End Offset registers should be modeled in simulator as one of memory mapped I/O registers. These should be used for wrapping operation of Offset register in ring mode.
- When sbf0mode/sbf1mode flags are written, newly updated values are not effective to the operation of sequential buffers. New values are effective after the sequential buffer is newly initialized, which means that SBL0OFF/SBL1OFF registers are newly written.

NOTES

SAMSUNG CONFIDENTIAL

5

CLOCK & POWER MANAGEMENT

OVERVIEW

The clock & power management unit consists of clock control, power control and reset control.

The clock control logic in S5L8700X generates various system clock signals: FCLK(FCLK_CPU, FCLK_DSP) for CPU core and DSP core and HCLK for the AHB bus and PCLK for the APB bus. The clock control logic allows bypassing of PLL for slow clock and connection/disconnection of the clock to each peripheral block by software, which results in power reduction.

Also, S5L8700X has the power control logic to support various power management schemes for optimal power consumption for a given application. The power management provides five power modes: NORMAL mode, SLOW mode, IDLE mode, STANDBY mode and STOP mode.

In NORMAL mode clock is supplied to CPU as well as all peripherals in S5L8700X. The power consumption will be a maximum when all peripherals are turned on. Also, user is allowed to control supply of the clock to peripherals by software. For example, if user does not need timer and DMA, user can disconnect the clock to timer and DMA to reduce the power consumption.

The SLOW mode is a non-PLL mode. Only difference to NORMAL mode is that the SLOW mode uses the external clock as a master clock in S5L8700X rather than the internal PLL clock. In this case, the power consumed by PLL itself is eliminated, and the power consumption will depend on the frequency of the external clock.

The IDLE mode disconnects the clock to CPU core while maintaining the clock to all peripherals. By using this IDLE mode, we can further reduce the power consumption by the CPU core. The wake-up from IDLE mode is done by an interrupt request to CPU.

STOP mode freezes all clocks to the CPU as well as peripherals by disabling PLLs. The power consumption is only due to the leakage current in S5L8700X. The wake-up from STOP mode can be done by activating external interrupt pins.

The reset controller in S5L8700X consists of three reset types: hardware reset, software reset and watchdog reset. These types of reset are described in detail on page 5-16 *Reset Controller*.

Feature

- Input frequency : 32.768 kHz.
- Output frequency range : 20MHz – 100MHz.
- Programmable frequency divider
- Power management : Normal, Slow, Idle, Standby and Stop.
- Reset controller : Hardware, Software, and Watchdog reset.

FUNCTION DESCRIPTION

CLOCK GENERATION

Figure 5-1 shows a block diagram of the clock generator. An external crystal clock is connected to the oscillation amplifier, and the PLL (Phase-Locked-Loop) converts the low input frequency into a high-frequency clock required by S5L8700X. The clock generator block also has a built-in logic to stabilize the clock frequency after each system reset since the clock takes time before stabilized.

Maximum Bus Frequencies

Table 5-1 lists the maximum operating frequencies for the S5L8700X.

Table 5-1. Maximum Bus Frequencies

Internal Bus	Maximum Frequency	Module on the Internal Bus	Symbol
CPU	200MHz	CPU Core, I/D cache, Write buffer	FCLK_CPU
DSP	130MHz	DSP Core, I/D cache	FCLK_DSP
AHB	100MHz	DMA,Interrupt,Clock & Power, Memory controller, AHB Master & Slave	HCLK
APB	100MHz	IIC, GPIO, UART, Timer, RTC, IIS, Watchdog timer.	PCLK

Note : Max frequency of DRAM(SDRAM, Mobile SDARM, DDR, Mobile DDR) is 100MHz

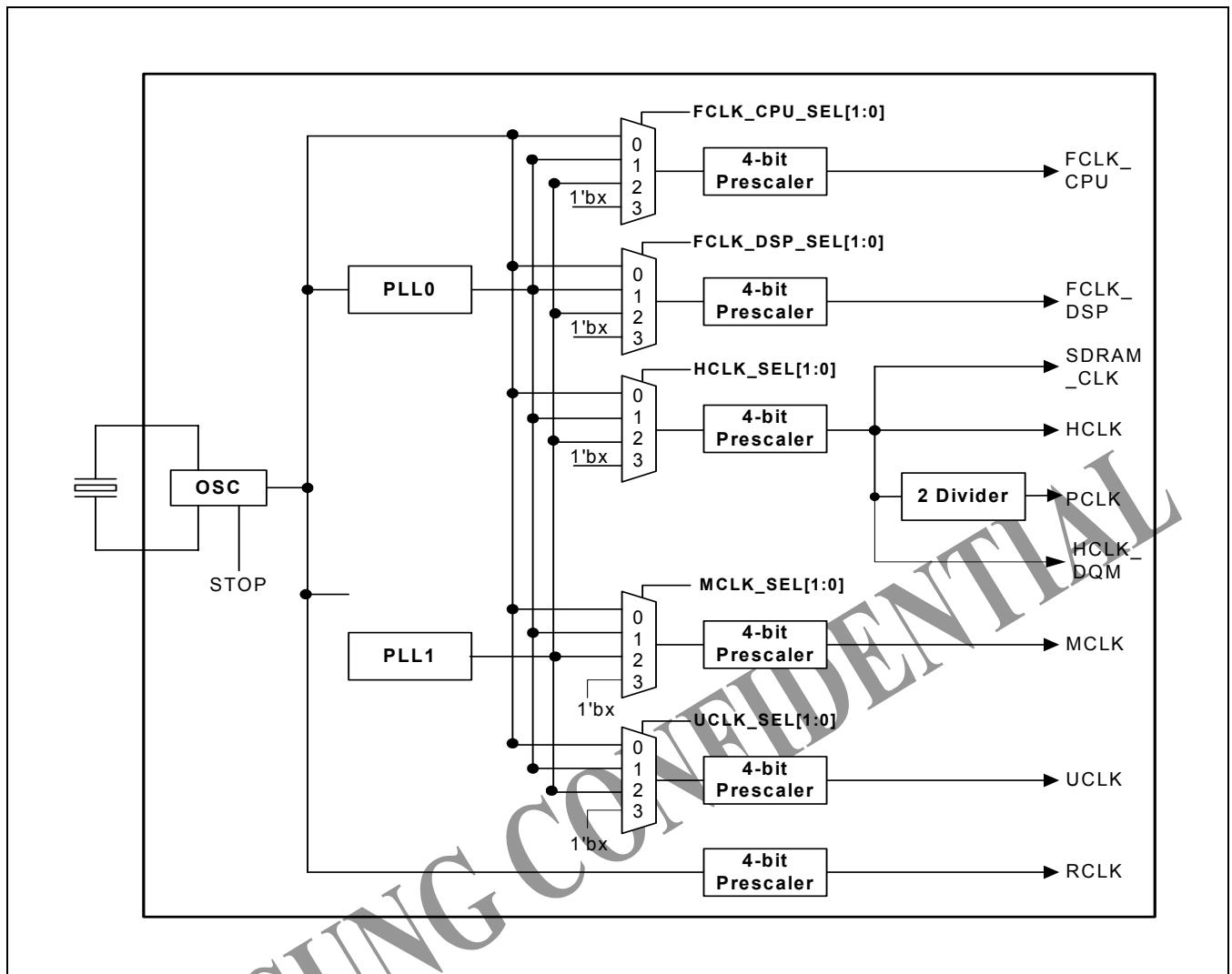


Figure 5-1. Clock Generator Block Diagram

NOTE : Until PLLPMS register is configured for desired clock frequency by user, OSC clock (Fin) is supplied to the system.

PLL (PHASE LOCKED LOOP)

The PLL in the clock generator synchronizes the output signal with the input reference signal in terms of frequency as well as phase. The output clock frequency F_{Pll} is related to the reference input clock frequency F_{in} by the following equation:

$$F_{\text{out}} = F_{\text{in}} * (\text{PDIV} + 2) * (\text{MDIV} + 8) / \text{pow}(2, \text{SDIV})$$

Change PLL Settings in Normal Operation Mode

During the operation of S5L8700X in NORMAL mode, if users want to change the frequency by modifying PMS value, the PLL lock time is automatically inserted. During the lock time, the clock will not be supplied to internal blocks in S5L8700X. The timing diagram is as follow.

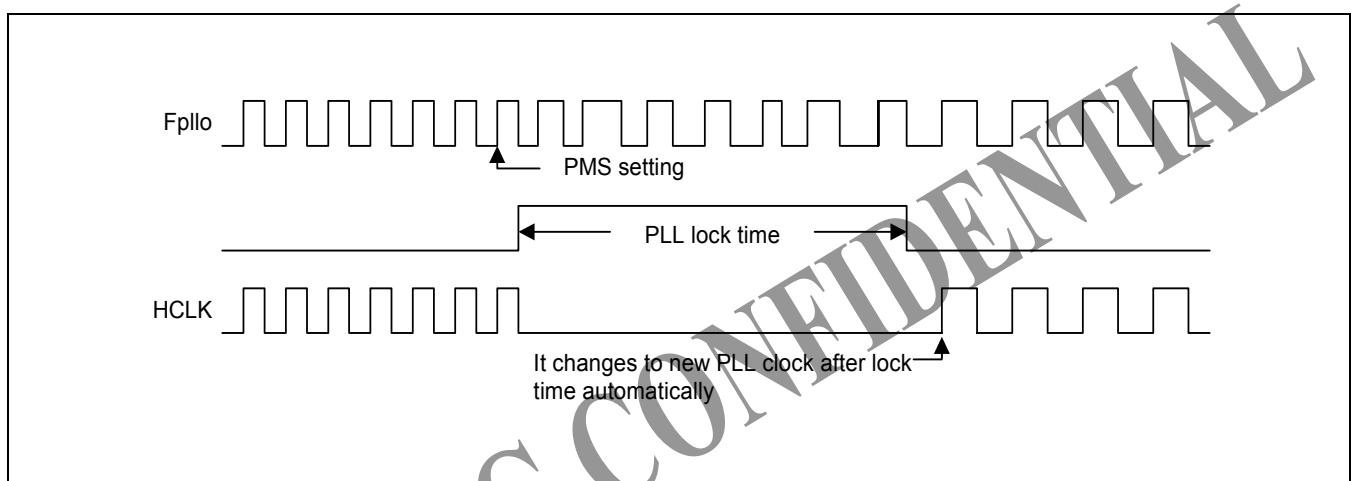


Figure 5-2. Timing Diagram of Clock Change in NORMAL Mode

POWER MANAGEMENT

All clock signals for each AHB and APB device can be maskable. The CPU controls each of the clocks to enable or disable it to perform local power management. The CPU itself have its own clock to be masked to enter IDLE mode which is one of the power saving mode of S5L8700X and can be woken up by interrupt. Including IDLE power saving mode S5L8700X provides 4 global power saving modes which are SLOW, IDLE, STANBY, and STOP.

NORMAL Mode :

All clocks are alive.

SLOW Mode :

Non-PLL mode. Unlike the Normal mode, the Slow mode uses an X-tal oscillator directly as HCLK in the S5L8700X without PLL. In this mode, the power consumption depends on the frequency of the external clock only. The power consumption due to PLL is excluded. This is mainly for the purpose of displaying time on the LCD screen while the mp3 player is not operating. To display the time the CPU needs to be engaged but it doesn't need to run as fast as it runs to play audio. The CPU, RTC, and a number of peripherals are needed to run and display time on an LCD and they are required to operate with the X-tal frequency to consume minimum power. And all other peripherals that doesn't need to operate are powered-down by having their clocks masked. The X-tal frequency should be 32.768kHz for the real time clock. To further reduce the power consumption in this case, the CPU may be in IDLE mode and woken-up, so to speak, at every 0.5sec to update the time display on the LCD while the RTC operates always.

IDLE Mode :

The CPU can mask its own clock, *hclk_cpu* to enter IDLE mode and later interrupt input (nIRQ or nFIQ) can wake the CPU up. Before CPU enters IDLE Mode it should guarantee there will be no further AHB+ transaction.

The nIRQ and nFIQ comes to the clock generation unit also to unmask *hclk_cpu* that enables the CPU to be woken-up and to recognize the interrupt consequently.

STOP Mode :

The X-tal OSC is disabled to enter STOP Mode. The external interrupt may cause return to NORMAL Mode. When it entered STOP Mode the S5L8700X maintains statically its latest state. The SDRAM(if exist) should be in self-refresh mode before entering STOP Mode. To support alarm function, Real Time Clock(RTC) also should be able to wake up S5L8700X from STOP mode to NORMAL mode.

STANDBY Mode :

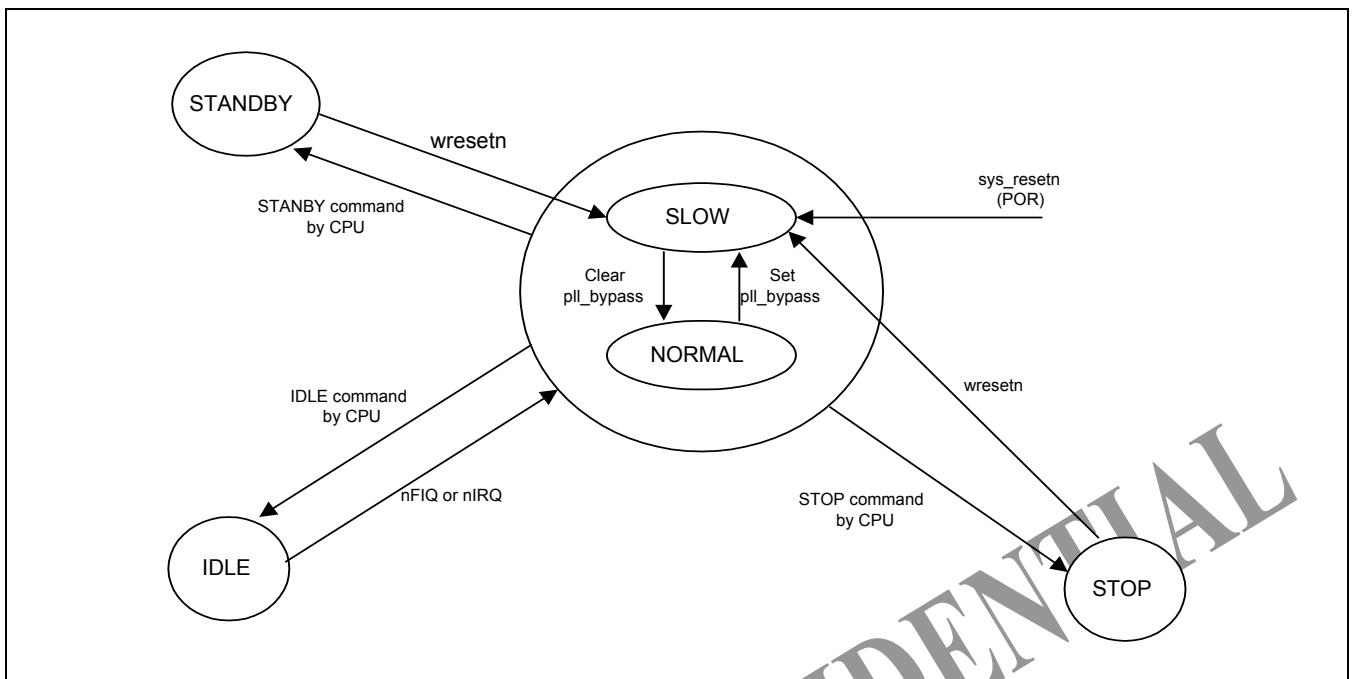
At this power saving mode, just RTC and X-tal for RTC is operating and all others are powered-down. We can't display TIME on the LCD because the CPU and the LCD interface unit will be powered-down at this mode. But the time is correctly maintained in the RTC and when woken up, the TIME can be displayed on the LCD. RTC operation with TIME display on an LCD requires RTC, CPU, BUSes and LCDIF to be alive, so this is the case of SLOW power saving mode.

To enter STOP/STANDBY mode follow the sequence :

1. If $\text{pclk} == \text{hclk} / 2$, make pclk is equal to hclk
2. CalmADM3 sends the STOP/STANDBY power saving command to clock generation unit. The FSM in the clock generation unit executes the following sequence (step 3~step6).
3. Masks all clocks.
4. Changes clock source to X-tal OSC from PLL.
5. Controls the PLL to be disabled (power-down).
6. Disable X-tal OSC.(STOP mode only)

To recover to NORMAL Mode : (Initiated by external interrupt coming.)

1. External interrupt received.
2. Upon receiving the external interrupt, *The clock and reset generation unit asserts wresetn to "LOW"* that makes the X-tal OSC to be enabled.
3. *The clock and reset generation unit releases the wresetn to "HIGH" after 128 counts of the $\text{extint_cnt}[6:0]$. The wresetn is to reset the WDT(watch dog timer) and possibly released before the oscillator becomes stable.*
4. The WDT counts the clock which is possibly unstable and sends the clock generation unit the wdt_send signal that indicates the oscillator has been stabilized. All clocks are disabled until wdt_start is received . (except the clock for the WDT of source which uses the clock hclk_pre that never be masked.)
5. The CPU enables the PLL.
6. The CPU waits until the PLL settles monitoring the Lock flag of the PLL.
7. Changes the clock source to PLL from X-tal OSC.

Global Power Management:**Figure 5-3. The Global Power Management****Local Power Management:**

While staying in **NORMAL** Mode the CPU can mask each of the clocks for AHB and APB peripheral devices to exploit power save as following :

Besides the global power saving mode stated above, we provide local power management scheme so that the CPU can disable each peripheral device by masking the clock to the device when it is not needed to operate.

Classification	Devices	Events
No external events involved	IIS UART IIC SPI SPDIF LCD I/F ADC RTC	
External events involved	USB MS I/F SMC I/F SD Card I/F MMC Card I/F	USB connection Card Insertion Card Insertion Card Insertion Card Insertion

1. RTC Power Issues

S5L8700X does not provide a separate power and ground for RTC.

2. Memory Stick Interface and Smart Media Card Interface Power Management

The Memory Stick Interface and Smart Media Card Interface Unit may be powered- down when they don't have cards in their slots.

When a card is inserted in it's slot the external circuitry (with a circuit inside the card) might give a dc voltage level which is different to the level shown when the card is disconnected. This signal might be connected to external interrupt pin for CPU to recognize the card insertion.

The CPU may enable the clocks for Memory Stick Interface and Smart Media Card Interface each time when it needs to access the cards and disable the clocks after each access. The CPU might need to check the status of the card interface before disables the clock. (We need to check if there is any limitation on when the clock can be disabled safely to be woken-up and work properly later time.)

3. SD Card and Multi-Media Card Power Management

The CPU polls the SD Card Interface to identify if a card is installed or not. If not installed it masks the clock for the SD Card Interface. When the CPU does not want to access the SD Card it may mask the clock for the SD Card Interface.

RESET CONTROLLER

The reset controller manages the various reset sources in the S5L8700X. For a programmer, two reset control registers are provided: one used to invoke software reset and one to read the status showing why the processor is reset after the reset sequence. After booting from the reset, software can examine the reset status register (RSTSR) to determine which types of reset has caused the reset condition.

Three types of reset in the S5L8700X are described below:

Hardware Reset

Hardware reset is invoked when the nRESET pin is asserted, and all units in the S5L8700X are initialized to a known state. Hardware reset is intended to be used for power-up only. Because the memory controller receives a full reset, all dynamic memory(DRAM/SDRAM) contents will be lost during hardware reset.

The nRESET_OUT pin is asserted during hardware reset.

Software Reset

Software reset is invoked when the software reset (SWR) bit in the SWRCON is set by software. After the SWR bit is set, the S5L8700X stays in reset state for 128 APB bus clocks (PCLK) and then is allowed to boot again.

The nRESET_OUT pin is asserted during software reset

Watchdog Reset

Watchdog reset is invoked when the watchdog enable bits in the WDTCON[7:0] are set and the watchdog timer counter (WDTCNT) overflows. The sequence of watchdog reset is identical to software reset. When the WTCNT overflows, the S5L8700X stays in reset state for 128 APB bus clocks (PCLK) and then is allowed to boot again.

The nRESET_OUT pin is asserted during watchdog reset.

CLOCK AND POWER MANAGEMENT SPECIAL FUNCTION REGISTERS

PLL PMS VALUE REGISTER (PLLPMS)

$$F_{out} = F_{in} * (PDIV + 2) * (MDIV + 8) / \text{pow}(2, SDIV)$$

Table 5-2. Recommended Value of MDIV, PDIV and SDIV for Audio Clock

Application	Mode	Fs (kHz)	F _{in} (kHz)	PDIV	MDIV	SDIV	Customer's Divide	F _{out} (MHz)	F _{available} (MHz)
normal	384	44.1	32.768	37	151	1	6	16.9344	16.9329
		32	32.768	28	192	2	4	12.288	12.2880
		48	32.768	43	117	1	5	18.432	18.4320
	256	44.1	32.768	37	151	1	9	11.2896	11.2886
		32	32.768	28	192	2	6	8.192	8.1920
		48	32.768	28	192	2	4	12.288	12.2880
	512	44.1	32.768	50	98	2	2	22.5792	22.5772
		32	32.768	28	192	2	3	16.384	16.3840
		48	32.768	28	192	1	4	24.576	24.5760
half freq	384	22.05	32.768	37	151	2	6	8.4672	8.4664
		16	32.768	28	192	2	8	6.144	6.1440
		24	32.768	43	117	2	5	9.126	9.1260
	256	22.05	32.768	50	98	2	8	5.6448	5.6443
		16	32.768	28	192	3	6	4.096	4.0960
		24	32.768	28	192	2	8	6.144	6.1440
	512	22.05	32.768	50	98	2	4	11.2896	12.2886
		16	32.768	28	192	2	6	8.192	8.1920
		24	32.768	28	192	2	4	12.288	12.2880
quarter freq	384	11.03	32.768	42	133	2	12	4.23552	4.2353
		8	32.768	28	192	3	8	3.072	3.0720
		12	32.768	43	117	2	10	4.608	4.6030
	256	11.03	32.768	26	189	3	8	2.82368	2.8242
		8	32.768	28	192	3	12	2.048	2.0480
		12	32.768	28	192	3	8	3.072	3.0720
	512	11.03	32.768	26	189	2	8	5.64736	5.6484
		8	32.768	28	192	3	6	4.096	4.0960
		12	32.768	28	192	2	8	6.144	6.1440

Table 5-2. Recommended Value of MDIV, PDIV and SDIV for Audio Clock (Continued)

Application	Mode	Fs (kHz)	Fin (kHz)	PDIV	MDIV	SDIV	Customer's Divide	Fout (MHz)	F_available (MHz)
double freq	384	88.2	32.768	37	151	1	3	33.8688	33.6657
		64	32.768	28	192	1	4	24.576	24.5760
		96	32.768	43	117	0	5	36.864	36.8640
	256	88.2	32.768	50	98	1	4	22.5792	22.5772
		64	32.768	28	192	1	6	16.384	16.3840
		96	32.768	28	192	1	4	24.576	24.5760
	512	88.2	32.768	50	98	1	2	45.1584	45.1543
		64	32.768	28	192	1	3	32.768	32.7680
		96	32.768	28	192	0	4	49.152	49.1520

Table 5-3. Recommended Value of MDIV, PDIV and SDIV for System Clock

Fin (kHz)	PDIV	MDIV	SDIV	Customer's Divide	Fout (MHz)	F_available (MHz)
32.768	26	210	2	1	50	50.0040
32.768	40	101	1	1	75	75.0060
32.768	26	210	1	1	100	100.0079
32.768	33	210	1	1	125	125.0099
32.768	40	101	0	1	150	150.0119
32.768	26	210	0	1	200	200.0159

NOTE: One PLL output frequency is selected to be used for system clock as Table 6-6. The system clock includes CPU clock, DSP clock and BUS clock which are derived from corresponding Customer's Divide at the output of the PLL. The implementation of the PLL output frequency for the system clock don't have to be exactly the same with the nominal value suggested at the Table 6-6, but can be selected closest value instead.

Table 5-4. Recommended Value of MDIV, PDIV and SDIV for USB Clock

Fin (kHz)	PDIV	MDIV	SDIV	Customer's Divide	Fout (MHz)	F_available (MHz)
32.768	45	179	1	3	48	47.9997
32.768	25	209	0	4	48	47.9969
32.768	31	214	0	5	48	48.0117
32.768	45	179	0	6	48	47.9997

NOTE: This value may be calculated using PLLSET.EXE utility from Samsung. This PLL is not guaranteed that the PMS values are all zeros.

Table 5-5. Application Requirement on System Clock

PLL OUTPUT Frequency	Customer's Divide (1~16)	System Clock (MHz)	PLL OUTPUT Frequency	Customer's Divide (1~16)	System Clock (MHz)	PLL OUTPUT Frequency	Customer's Divide (1~16)	System Clock (MHz)
50	1	50.0	100	1	100.0	150	1	150.0
	2	25.0		2	50.0		2	75.0
	3	16.7		3	33.3		3	50.0
	4	12.5		4	25.0		4	37.5
	5	10.0		5	20.0		5	30.0
	6	8.3		6	16.7		6	25.0
	7	7.1		7	14.3		7	21.4
	8	6.3		8	12.5		8	18.8
	9	5.6		9	11.1		9	16.7
	10	5.0		10	10.0		10	15.0
	11	4.5		11	9.1		11	13.6
	12	4.2		12	8.3		12	12.5
	13	3.8		13	7.7		13	11.5
	14	3.6		14	7.1		14	10.7
	15	3.3		15	6.7		15	10.0
	16	3.1		16	6.3		16	9.4
75	1	75.0	125	1	125.0	200	1	200.0
	2	37.5		2	62.5		2	100.0
	3	25.0		3	41.7		3	66.7
	4	18.8		4	31.3		4	50.0
	5	15.0		5	25.0		5	40.0
	6	12.5		6	20.8		6	33.3
	7	10.7		7	17.9		7	28.6
	8	9.4		8	15.6		8	25.0
	9	8.3		9	13.9		9	22.2
	10	7.5		10	12.5		10	20.0
	11	6.8		11	11.4		11	18.2
	12	6.3		12	10.4		12	16.7
	13	5.8		13	9.6		13	15.4
	14	5.4		14	8.9		14	14.3
	15	5.0		15	8.3		15	13.3
	16	4.7		16	7.8		16	12.5

REGISTER MAP

CLOCK CONTROL REGISTER (CLKCON)

Register	Address	R/W	Description	Reset Value
CLKCON	0x3C50_0000	R/W	Clock control Register	0x0000 0000

CLKCON	Bit	Description	Initial State
FCLK_CPU_MASK	[31]	Mask FCLK into CPU core 0 = enable 1 = disable	0
FCLK_CPU_SEL	[30:29]	00 = OSC 01 = PLL0 10 = PLL1 11 = reserved	0x0
FCLK_CPU_DIV_ON	[28]	0 = prescaler off 1 = prescaler on	0
FCLK_CPU_DIV_VAL	[27:24]	FCLK_CPU = input clock / (FCLK_CPU_DIV_VAL+1) when FCLK_CPU_DIV_ON == 1	0x0
FCLK_DSP_MASK	[23]	Mask FCLK into DSP core 0 = enable 1 = disable	0
FCLK_DSP_SEL	[22:21]	00 = OSC 01 = PLL0 10 = PLL1 11 = reserved	0x0
FCLK_DSP_DIV_ON	[20]	0 = prescaler off 1 = prescaler on	0
FCLK_DSP_DIV_VAL	[19:16]	FCLK_DSP = input clock / (FCLK_DSP_DIV_VAL+1) when FCLK_DSP_DIV_ON == 1	0x0
HCLK_MASK	[15]	-	0
HCLK_SEL	[14:13]	00 = OSC 01 = PLL0 10 = PLL1 11 = reserved	0x0
HCLK_DIV_ON	[12]	0 = prescaler off 1 = prescaler on	0
HCLK_DIV_VAL	[11:8]	HCLK = input clock / (HCLK_DIV_VAL+1) when HCLK_DIV_ON == 1	0x0
MCLK_MASK	[7]	Mask audio clock into SPDIF or IIS block 0 = enable 1 = disable	0
MCLK_SEL	[6:5]	00 = OSC 01 = PLL0 10 = PLL1 11 = reserved	0x0
MCLK_DIV_ON	[4]	0 = prescaler off 1 = prescaler on	0
MCLK_DIV_VAL	[3:0]	MCLK = input clock / (MCLK_DIV_VAL+1) when MCLK_DIV_ON == 1	0x0

PLL CONTROL REGISTER (PLLCON)

Register	Address	R/W	Description	Reset Value
PLLCON	0x3C50_0024	R/W	PLL control register	0x0000_0000

PLLCON	Bit	Description	Initial State
PLL1_PWD	[1]	PLL1 Power Down 0 : PLL1 is turned off. 1 : PLL1 is turned on.	0
PLL0_PWD	[0]	PLL0 Power Down 0 : PLL0 is turned off. 1 : PLL0 is turned on.	0

PLL0 PMS VALUE REGISTER (PLL0PMS)

Register	Address	R/W	Description	Reset Value
PLL0PMS	0x3C50_0004	R/W	PLL PMS value Register	0x001A_D201

PLLPMS	Bit	Description	Initial State
PDIV	[21:16]	Pre divider control	6'h1a (6'h25)
MDIV	[15:8]	Main divider control	8'hd2 (8'h97)
	[7:2]	Reserved	
SDIV	[1:0]	Post-divider control	2'h1 (2'h1)

PLL1 PMS VALUE REGISTER (PLL1PMS)

Register	Address	R/W	Description	Reset Value
PLL1PMS	0x3C50_0008	R/W	PLL PMS value Register	0x0025_9701

PLLPMS	Bit	Description	Initial State
PDIV	[21:16]	Pre divider control	6'h1a (6'h25)
MDIV	[15:8]	Main divider control	8'hd2 (8'h97)
	[7:2]	Reserved	
SDIV	[1:0]	Post-divider control	2'h1 (2'h1)

PLL0 LOCK COUNT REGISTER (PLL0LCNT)

Register	Address	R/W	Description	Reset Value
PLL0LCNT	0x3C50_0014	R/W	PLL0 lock count register	0x0000_0000

PLLLCNT	Bit	Description	Initial State
LOCK_CNT	[12:0]	PLL lock count value (down counter)	0x0

NOTE: Maximum PLL locking time = 150 us

PLL1 LOCK COUNT REGISTER (PLL1LCNT)

Register	Address	R/W	Description	Reset Value
PLL1LCNT	0x3C50_0018	R/W	PLL1 lock count register	0x0000_0000

PLLLCNT	Bit	Description	Initial State
LOCK_CNT	[12:0]	PLL lock count value (down counter)	0x0

NOTE: Maximum PLL locking time = 150 us

PLL LOCK STATUS REGISTER (PLLLOCK)

Register	Address	R/W	Description	Reset Value
PLLLOCK	0x3C50_0020	R	PLL lock status register	0x0000_0000

PLLLOCK	Bit	Description	Initial State
PLL1_LOCK	[1]	PLL1 Lock Status 0 : Progress 1 : Locking done	0
PLL0_LOCK	[0]	PLL0 Lock Status 0 : Progress 1 : Locking done	0

CLOCK POWER CONTROL REGISTER (PWRCON)

Register	Address	R/W	Description	Reset Value
PWRCON	0x3C50_0028	R/W	Clock power control register	0x0000_0000

PWRCON	Bit	Description	Initial State
RCLKclkOn HCLK[7]	[22]	Controls real-time clock(RCLK) into RTC block 0 = enable 1 = disable	0
SDRAMclkOn HCLK[6]	[21]	Controls clock(sdram_clk) into external sdram 0 = enable 1 = disable	0
ECCclkOn HCLK[5]	[20]	Controls HCLK into ECC block 0 = enable 1 = disable	0
ATAclkOn HCLK[4]	[19]	Controls HCLK into ATA controller block 0 = enable 1 = disable	0
LCDclkOn HCLK[3]	[18]	Controls HCLK into LCD controller block 0 = enable 1 = disable	0
DSPclkOn HCLK[2]	[17]	Controls HCLK into DSP block 0 = enable 1 = disable	0
USB1.1hostclkOn HCLK[1]	[16]	Controls HCLK into USB1.1 host block 0 = enable 1 = disable	0
USB2.0funclkOn HCLK[0]	[15]	Controls HCLK into USB2.0 function block 0 = enable 1 = disable	0
USB2.0phyclkOn PCLK[15]	[14]	Controls PCLK into USB2.0 Phy controller block 0 = enable 1 = disable	0
RTCclkOn PCLK[14]	[13]	Controls PCLK into RTC block 0 = enable 1 = disable	0
ChipIDclkOn PCLK[13]	[12]	Controls PCLK into ChipID block 0 = enable 1 = disable	0
GPIOclkOn PCLK[12]	[11]	Controls PCLK into GPIO block 0 = enable 1 = disable	0
ADCclkOn PCLK[11]	[10]	Controls PCLK into ADC block 0 = enable 1 = disable	0
SPIclkOn PCLK[10]	[9]	Controls PCLK into SPI block 0 = enable 1 = disable	0
UARTclkOn PCLK[9]	[8]	Controls PCLK into UART block 0 = enable 1 = disable	0
SPDIFclkOn PCLK[8]	[7]	Controls PCLK into SPDIF block 0 = enable 1 = disable	0
IISclkOn	[6]	Controls PCLK into IIS block	0

PCLK[7]		0 = enable 1 = disable	
IICclkOn PCLK[6]	[5]	Controls PCLK into IIC block 0 = enable 1 = disable	0
TIMERclkOn PCLK[5]	[4]	Controls PCLK into TIMER block 0 = enable 1 = disable	0
MSCclkOn PCLK[4]	[3]	Controls PCLK into Memory Stick Controller block 0 = enable 1 = disable	0
SDCMMCclkOn PCLK[3]	[2]	Controls PCLK into SDC & MMCblock 0 = enable 1 = disable	0
FMCclkOn PCLK[2]	[1]	Controls PCLK into Flash Memory Controller block 0 = enable 1 = disable	0
LCDIFclkOn PCLK[1]	[0]	Controls PCLK into LCD interface block 0 = enable 1 = disable	0

POWER MODE CONTROL REGISTER (PWRMODE)

Register	Address	R/W	Description	Reset Value
PWRMODE	0x3C50_002C	R/W	Power mode control register	0x0000_0000

PLLCON	Bit	Description	Initial State
STOP MODE	[11:8]	Enter STOP mode 1010 = Transition to STOP mode	0x0
STANDBY MODE	[7:4]	Enter STANDBY mode 0011 = Transition to STANDBY mode	0x0
IDLE MODE	[3:0]	Enter IDLE mode 0101 = Transition to IDLE mode	0x0

NOTE: The power mode control register can't set simultaneously each mode.

SOFTWARE RESET CONTROL REGISTER (SWRCON)

Register	Address	R/W	Description	Reset Value
SWRCON	0x3C5_0030	W	Software reset control register	0x0000 0000

SWRCON	Bit	Description	Initial State
SWR	[7:0]	Software reset. 1010_0101 = Invoke a software reset of the chip. Other value = Do not invoke a software reset of the chip. This bit is self-clearing, and is automatically cleared several system clock cycles after it has been set.	0x0

RESET STATUS REGISTER (RSTSR)

Register	Address	R/W	Description	Reset Value
RSTSR	0x3C50_0034	R	Reset status register	0x0000 0001

RSTSR	Bit	Description	Initial State
WDR	[2]	Watchdog reset.(Read only) 0 = Watchdog reset has not occurred. 1 = Watchdog reset has occurred This bit is cleared automatically when one of the other reset status bit is set.	0
SWR	[1]	Software reset.(Read only) 0 = Software reset has not occurred. 1 = Software reset has occurred This bit is cleared automatically when one of the other reset status bit is set.	0
HWR	[0]	Hardware reset.(Read only) 0 = Hardware reset has not occurred. 1 = Hardware reset has occurred This bit is cleared automatically when one of the other reset status bit is set.	1

DSP CLOCK MODE REGISTER (DSPCLKMD)

Register	Address	R/W	Description	Reset Value
DSPCLKMD	0x3C50_0038	R/W	DSP clock mode register	0x0000_0000

PLLCNT	Bit	Description	Initial State
CLKMODE	[0]	0 = DSP has async mode which dsp_fclk is different to hclk 1 = DSP has fast-bus mode which use hclk instead of dsp_fclk	0

CLOCK CONTROL REGISTER 2 (CLKCON2)

Register	Address	R/W	Description	Reset Value
CLKCON2	0x3C50_003C	R/W	clock control register 2	0x0000_0000

PLLCON	Bit	Description	Initial State
PCLK_DIV_ON	[8]	0 = PCLK has the clock same as the HCLK 1 = PCLK has the clock same as the HCLK/2	0
UCLK_MASK	[7]	Controls UCLK into USB host block 0 = UCLK enable 1 = UCLK disable	0
UCLK_SEL	[6:5]	00 = OSC 01 = PLL0 10 = PLL1 11 = PLL2	0x0
UCLK_DIV_ON	[4]	0 = prescaler off 1 = prescaler on	0
UCLK_DIV_VAL	[3:0]	UCLK = input clock / (UCLK_DIV_VAL+1) when UCLK_DIV_ON == 1	0x0

NOTES

SAMSUNG CONFIDENTIAL

6

INTERRUPT CONTROLLER UNIT

FUNCTIONAL DESCRIPTION

The interrupt controller in S5L8700X receives the request from 32 interrupt sources. These interrupt sources are provided by internal peripheral such as the DMA controller, UART, IIC, external interrupts, etc.

The role of the interrupt controller is to ask for the FIQ or IRQ interrupt requests to the ARM940T core after the arbitration process when there are multiple interrupt requests from internal peripherals and external interrupt request pins.

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

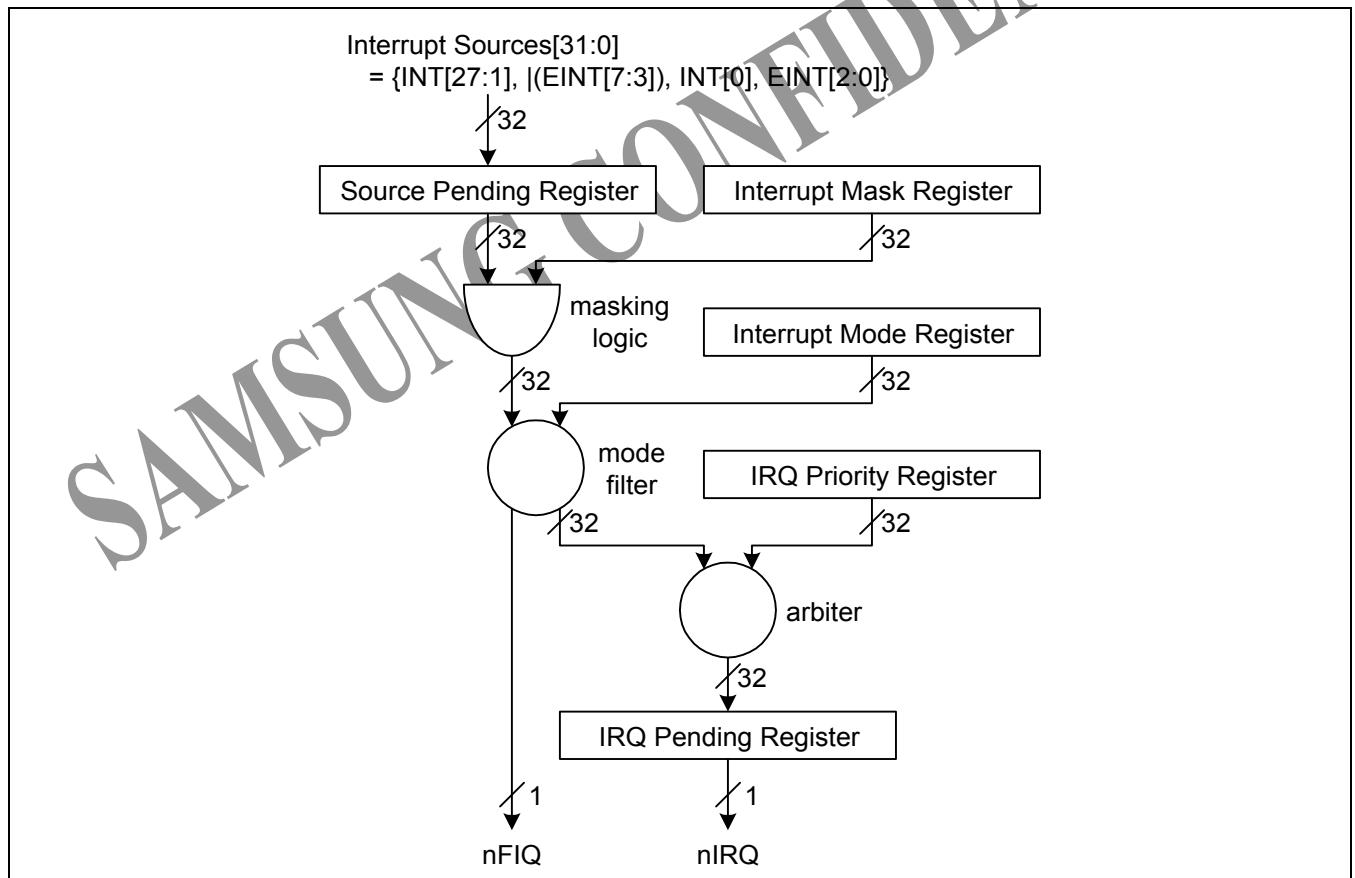


Figure 6-1. Interrupt Process Diagram

INTERRUPT SOURCES

ICU supports 28 internal interrupt sources and 8 external interrupt sources. But 5 external interrupt sources are ored to 1 source internally. Therefore, 28 internal interrupt sources, 1 ored external source and 3 external sources participate in arbitration.

All interrupt sources should be high active and more than 1 cycle pulse signals. Therefore, additional logic is needed for external interrupts. Additional logic can make external interrupts change signal polarity and be distinguished from invalid sources that were generated by noise or masked by user control register

INTERRUPT CONTROLLER OPERATION

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

If the F-bit of PSR (program status register in ARM940T 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 ARM940T CPU) is set to 1, the CPU does not accept the IRQ (interrupt request) from the interrupt controller. So, to enable the interrupt reception, the F-bit or I-bit of PSR has to be cleared to 0 and also the corresponding bit of INTMSK has to be set to 1.

Interrupt Mode

ARM940T 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

S5L8700X has two interrupt pending registers. The one is source pending register(**SRCPND**) and the other is interrupt pending register(**INTPND**). These pending registers indicate whether or not an interrupt request is pending. When the interrupt sources request interrupt service the corresponding bits of SRCPND register are set to 1, at the same time the only one bit of INTPND register is set to 1 automatically after arbitration process. If interrupts are masked, the corresponding bits of SRCPND register are set to 1, but the bit of INTPND register is not changed. When a pending bit of INTPND register is set, the interrupt service routine starts whenever the I-flag or F-flag is cleared to 0. The SRCPND and INTPND registers can be read and written, so the service routine must **clear the pending condition by writing a 1 to the corresponding bit in SRCPND register first** and then clear the pending condition in INTPND registers with same method.

Interrupt Mask Register

Indicates that an interrupt has been disabled if the corresponding mask bit is 0. If an interrupt mask bit of INTMSK is 1, the interrupt will be serviced normally. If the corresponding mask bit is 0 and the interrupt is generated, the source pending bit will be set.

INTERRUPT PRIORITY GENERATING BLOCK

The priority logic for 32 interrupt requests is composed of seven rotation-based arbiters: six first-level arbiters and one second-level arbiter as shown in the following figure.

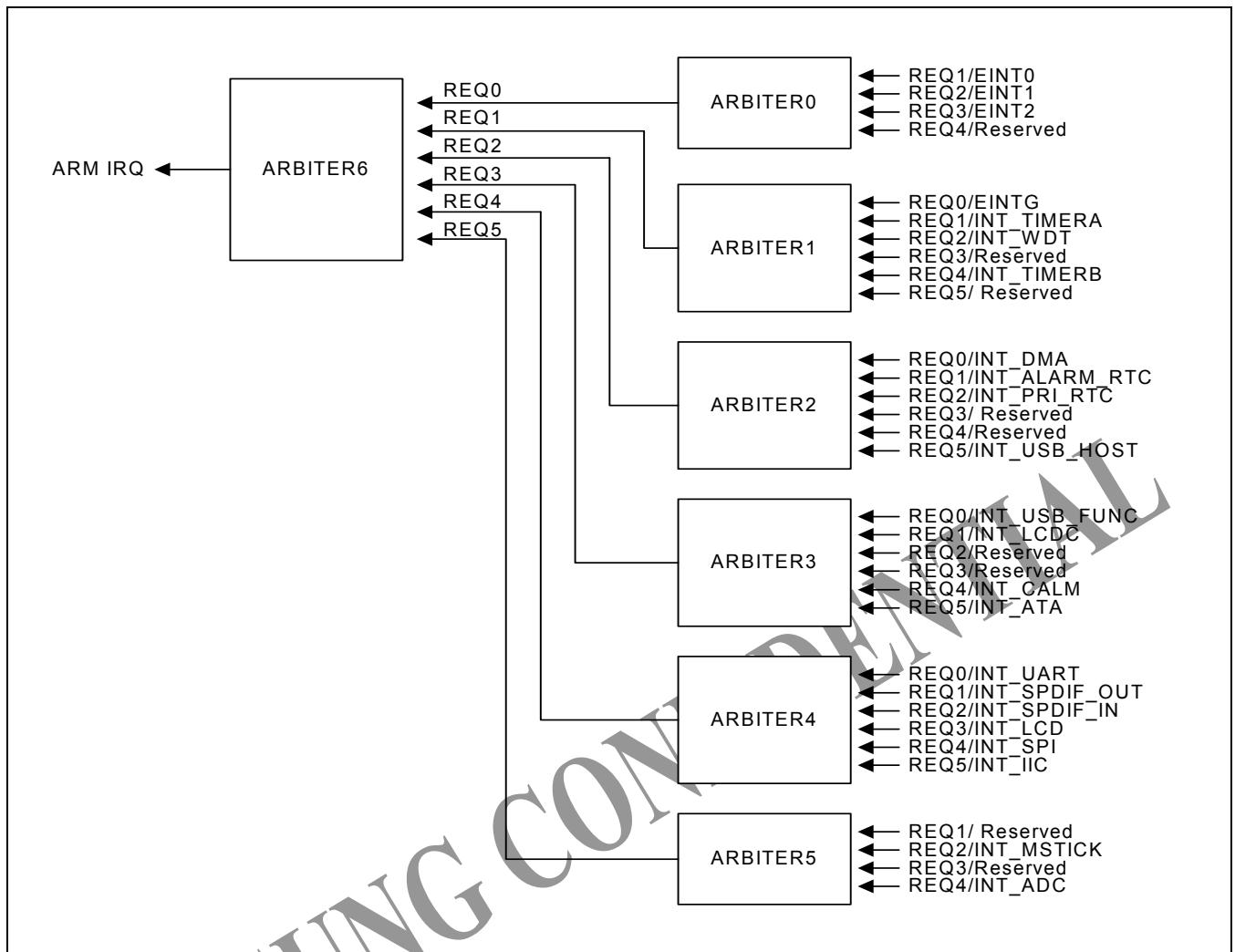


Figure 6-2. Priority Generating Block

INTERRUPT PRIORITY

Each arbiter can handle six interrupt requests based on the one bit arbiter mode control(**ARB_MODE**) and two bits of selection control signals(**ARB_SEL**) as follows:

If **ARB_SEL** bits are 00b, the priority order is REQ0, REQ1, REQ2, REQ3, REQ4, and REQ5.

If **ARB_SEL** bits are 01b, the priority order is REQ0, REQ2, REQ3, REQ4, REQ1, and REQ5.

If **ARB_SEL** bits are 10b, the priority order is REQ0, REQ3, REQ4, REQ1, REQ2, and REQ5.

If **ARB_SEL** bits are 11b, the priority order is REQ0, REQ4, REQ1, REQ2, REQ3, and REQ5.

Note that REQ0 of an arbiter is always the highest priority, and REQ5 is the lowest one. In addition, by changing the ARB_SEL bits, we can rotate the priority of REQ1 - REQ4.

Here, if ARB_MODE bit is set to 0, ARB_SEL bits are not automatically changed, thus the arbiter operates in the fixed priority mode. (Note that even in this mode, we can change the priority by manually changing the ARB_SEL bits.). On the other hand, if ARB_MODE bit is 1, ARB_SEL bits are changed in rotation fashion, e.g., if REQ1 is serviced, ARB_SEL bits are changed to 01b automatically so as to make REQ1 the lowest priority one. The detailed rule of ARB_SEL change is as follows.

If REQ0 or REQ5 is serviced, ARB_SEL bits are not changed at all.

If REQ1 is serviced, ARB_SEL bits are changed to 01b.

If REQ2 is serviced, ARB_SEL bits are changed to 10b.

If REQ3 is serviced, ARB_SEL bits are changed to 11b.

If REQ4 is serviced, ARB_SEL bits are changed to 00b.

DBG OPERATION MODE

DBGACK occurs during operation of Debugger unit in ARM940T. DBGACK signal makes that pending interrupts is not propagated to CalmADM3. If DBGACK signal is active(set to 0), both nIRQ and nFIQ is masked to 1. DBGACK_HCLK is synchronized with HCLK and stop the Watchdogtimer operation.

REGISTER MAP

There are 9 control registers in the interrupt controller: source pending register, interrupt mode register, mask register, priority register, interrupt pending register, offset register, external interrupt polarity register, external interrupt mask register and external interrupt pending register.

All the interrupt requests from the interrupt sources are first registered in the source pending register. They are divided into two groups based on the interrupt mode register, i.e., one FIQ request and the remaining IRQ requests. Arbitration process is performed for the multiple IRQ requests based on the priority register.

SOURCE PENDING REGISTER (SRCPND)

SRCPND register is composed of 32 bits each of which is related to an interrupt source. Each bit is set to 1 if the corresponding interrupt source generates the interrupt request and waits for the interrupt to be serviced. By reading this register, we can see the interrupt sources waiting for their requests to be serviced. Note that each bit of SRCPND register is automatically set by the interrupt sources regardless of the masking bits in the INTMASK register. In addition, it is not affected by the priority logic of interrupt controller.

In the interrupt service routine for a specific interrupt source, the corresponding bit of SRCPND register has to be cleared to get the interrupt request from the same source correctly. If you return from the ISR without clearing the bit, interrupt controller operates as if another interrupt request comes in from the same source. In other words, if a specific bit of SRCPND register is set to 1, it is always considered as a valid interrupt request waiting to be serviced.

The specific time to clear the corresponding bit depends on the user's requirement. The bottom line is that if you want to receive another valid request from the same source you should clear the corresponding bit first, and then enable the interrupt.

You can clear a specific bit of SRCPND register by writing a data to this register. It clears only the bit positions of SRCPND corresponding to those set to one in the data. The bit positions corresponding to those that are set to 0 in the data remains as they are with no change.

Register	Address	R/W	Description	Reset Value
SRCPND	0x39C0_0000	R/W	Indicates the interrupt request status. 0 = The interrupt has not been requested. 1 = The interrupt source has asserted the interrupt request.	0x00000000

SRCPND	Bit	Description	Initial State
INT_ADC	[31]	0 = Not requested, 1 = Requested	0
INT_ADC_WAKEUP	[30]	0 = Not requested, 1 = Requested	0
INT_MSTICK	[29]	0 = Not requested, 1 = Requested	0
Reserved	[28]	Not used	0
INT_IIC	[27]	0 = Not requested, 1 = Requested	0
INT_SPI	[26]	0 = Not requested, 1 = Requested	0
INT_LCD	[25]	0 = Not requested, 1 = Requested	0
INT_SDCI	[24]	0 = Not requested, 1 = Requested	0
INT_SPDIF_OUT	[23]	0 = Not requested, 1 = Requested	0
INT_UART0	[22]	0 = Not requested, 1 = Requested	0
INT_ATA	[21]	0 = Not requested, 1 = Requested	0
INT_CALM	[20]	0 = Not requested, 1 = Requested	0
INT_ECC	[19]	0 = Not requested, 1 = Requested	0
INT_LCDC[1]	[18]	0 = Not requested, 1 = Requested	0
INT_LCDC[0]	[17]	0 = Not requested, 1 = Requested	0
INT_USB_FUNC	[16]	0 = Not requested, 1 = Requested	0
INT_USB_HOST	[15]	0 = Not requested, 1 = Requested	0
INT_UART1	[14]	0 = Not requested, 1 = Requested	0
Reserved	[13]	Not used	0
INT_PRI_RTC	[12]	0 = Not requested, 1 = Requested	0
INT_ALARM_RTC	[11]	0 = Not requested, 1 = Requested	0
INT_DMA	[10]	0 = Not requested, 1 = Requested	0
INT_TIMER D	[9]	0 = Not requested, 1 = Requested	0
INT_TIMER C	[8]	0 = Not requested, 1 = Requested	0
INT_TIMER B	[7]	0 = Not requested, 1 = Requested	0
INT_WDT	[6]	0 = Not requested, 1 = Requested	0
INT_TIMER A	[5]	0 = Not requested, 1 = Requested	0
EINTG	[4]	0 = Not requested, 1 = Requested	0
EINT_VBUS	[3]	0 = Not requested, 1 = Requested	0
EINT2	[2]	0 = Not requested, 1 = Requested	0
EINT1	[1]	0 = Not requested, 1 = Requested	0
EINT0	[0]	0 = Not requested, 1 = Requested	0

INTERRUPT MODE REGISTER (INTMOD)

This register is composed of 32 bits each of which is related to an interrupt source. If a specific bit is set to 1, the corresponding interrupt is processed in the FIQ (fast interrupt) mode. Otherwise, it is processed in the IRQ mode (normal interrupt).

Note that at most **only one** interrupt source can be serviced in the FIQ mode in the interrupt controller. (You should use the FIQ mode only for the urgent interrupt.) Thus, only one bit of INTMOD can be set to 1 at most.

Register	Address	R/W	Description	Reset Value
INTMOD	0x39C0_0004	R/W	Interrupt mode register. 0 = IRQ mode 1 = FIQ mode	0x00000000

NOTE: If an interrupt mode is set to FIQ mode in INTMOD register, FIQ interrupt will not affect INTPND and INTOFFSET registers. The INTPND and INTOFFSET registers are valid only for IRQ mode interrupt source.

INTMOD	Bit	Description	Initial State
INT ADC	[31]	0 = IRQ, 1 = FIQ	0
INT_ADC_WAKEUP	[30]	0 = IRQ, 1 = FIQ	0
INT MSTICK	[29]	0 = IRQ, 1 = FIQ	0
Reserved	[28]	Not used	0
INT IIC	[27]	0 = IRQ, 1 = FIQ	0
INT SPI	[26]	0 = IRQ, 1 = FIQ	0
INT LCD	[25]	0 = IRQ, 1 = FIQ	0
INT_SDCI	[24]	0 = IRQ, 1 = FIQ	0
INT_SPDIF_OUT	[23]	0 = IRQ, 1 = FIQ	0
INT_UART0	[22]	0 = IRQ, 1 = FIQ	0
INT ATA	[21]	0 = IRQ, 1 = FIQ	0
INT_CALM	[20]	0 = IRQ, 1 = FIQ	0
INT_ECC	[19]	0 = IRQ, 1 = FIQ	0
INT_LCDC[1]	[18]	0 = IRQ, 1 = FIQ	0
INT_LCDC[0]	[17]	0 = IRQ, 1 = FIQ	0
INT_USB_FUNC	[16]	0 = IRQ, 1 = FIQ	0
INT_USB_HOST	[15]	0 = IRQ, 1 = FIQ	0
INT_UART1	[14]	0 = IRQ, 1 = FIQ	0
Reserved	[13]	Not used	0
INT_PRI_RTC	[12]	0 = IRQ, 1 = FIQ	0
INT_ALARM_RTC	[11]	0 = IRQ, 1 = FIQ	0
INT_DMA	[10]	0 = IRQ, 1 = FIQ	0
INT_TIMER_D	[9]	0 = IRQ, 1 = FIQ	0
INT_TIMER_C	[8]	0 = IRQ, 1 = FIQ	0
INT_TIMER_B	[7]	0 = IRQ, 1 = FIQ	0
INT_WDT	[6]	0 = IRQ, 1 = FIQ	0
INT_TIMER_A	[5]	0 = IRQ, 1 = FIQ	0
EINTG	[4]	0 = IRQ, 1 = FIQ	0
EINT_VBUS	[3]	0 = IRQ, 1 = FIQ	0
EINT2	[2]	0 = IRQ, 1 = FIQ	0
EINT1	[1]	0 = IRQ, 1 = FIQ	0
EINT0	[0]	0 = IRQ, 1 = FIQ	0

INTERRUPT MASK REGISTER (INTMSK)

Each of the 32 bits in the interrupt mask register is related to an interrupt source. If you set a specific bit to 0, the interrupt request from the corresponding interrupt source is not serviced by the CPU. (Note that even in such a case, the corresponding bit of SRCPND register is set to 1). If the mask bit is 1, the interrupt request can be serviced.

Register	Address	R/W	Description	Reset Value
INTMSK	0x39C0_0008	R/W	Determines which interrupt source is masked. The masked interrupt source will not be serviced. 1 = Interrupt service is available 0 = Interrupt service is masked	0x00000000

INTMSK	Bit	Description	Initial State
INT ADC	[31]	1 = Service available, 0 = Masked	0
INT_ADC_WAKEUP	[30]	1 = Service available, 0 = Masked	0
INT MSTICK	[29]	1 = Service available, 0 = Masked	0
Reserved	[28]	Not used	0
INT IIC	[27]	1 = Service available, 0 = Masked	0
INT SPI	[26]	1 = Service available, 0 = Masked	0
INT LCD	[25]	1 = Service available, 0 = Masked	0
INT_SDCI	[24]	1 = Service available, 0 = Masked	0
INT_SPDIF_OUT	[23]	1 = Service available, 0 = Masked	0
INT_UART0	[22]	1 = Service available, 0 = Masked	0
INT ATA	[21]	1 = Service available, 0 = Masked	0
INT_CALM	[20]	1 = Service available, 0 = Masked	0
INT_ECC	[19]	1 = Service available, 0 = Masked	0
INT_LCDC[1]	[18]	1 = Service available, 0 = Masked	0
INT_LCDC[0]	[17]	1 = Service available, 0 = Masked	0
INT_USB_FUNC	[16]	1 = Service available, 0 = Masked	0
INT_USB_HOST	[15]	1 = Service available, 0 = Masked	0
INT_UART1	[14]	1 = Service available, 0 = Masked	0
Reserved	[13]	Not used	0
INT_PRI RTC	[12]	1 = Service available, 0 = Masked	0
INT_ALARM RTC	[11]	1 = Service available, 0 = Masked	0
INT_DMA	[10]	1 = Service available, 0 = Masked	0
INT_TIMER D	[9]	1 = Service available, 0 = Masked	0
INT_TIMER C	[8]	1 = Service available, 0 = Masked	0
INT_TIMER B	[7]	1 = Service available, 0 = Masked	0
INT_WDT	[6]	1 = Service available, 0 = Masked	0
INT_TIMER A	[5]	1 = Service available, 0 = Masked	0
EINTG	[4]	1 = Service available, 0 = Masked	0
EINT_VBUS	[3]	1 = Service available, 0 = Masked	1
EINT2	[2]	1 = Service available, 0 = Masked	0
EINT1	[1]	1 = Service available, 0 = Masked	0
EINT0	[0]	1 = Service available, 0 = Masked	0

PRIORITY REGISTER (PRIORITY)

Register	Address	R/W	Description	Reset Value
PRIORITY	0x39C0_000 C	W	IRQ priority control register	0x7f

PRIORITY	Bit	Description	Initial State
ARB_SEL6	[20:19]	Arbiter 6 group priority order set 00 = REQ 0-1-2-3-4-5 01 = REQ 0-2-3-4-1-5 10 = REQ 0-3-4-1-2-5 11 = REQ 0-4-1-2-3-5	0
ARB_SEL5	[18:17]	Arbiter 5 group priority order set 00 = REQ 1-2-3-4 01 = REQ 2-3-4-1 10 = REQ 3-4-1-2 11 = REQ 4-1-2-3	0
ARB_SEL4	[16:15]	Arbiter 4 group priority order set 00 = REQ 0-1-2-3-4-5 01 = REQ 0-2-3-4-1-5 10 = REQ 0-3-4-1-2-5 11 = REQ 0-4-1-2-3-5	0
ARB_SEL3	[14:13]	Arbiter 3 group priority order set 00 = REQ 0-1-2-3-4-5 01 = REQ 0-2-3-4-1-5 10 = REQ 0-3-4-1-2-5 11 = REQ 0-4-1-2-3-5	0
ARB_SEL2	[12:11]	Arbiter 2 group priority order set 00 = REQ 0-1-2-3-4-5 01 = REQ 0-2-3-4-1-5 10 = REQ 0-3-4-1-2-5 11 = REQ 0-4-1-2-3-5	0
ARB_SEL1	[10:9]	Arbiter 1 group priority order set 00 = REQ 0-1-2-3-4-5 01 = REQ 0-2-3-4-1-5 10 = REQ 0-3-4-1-2-5 11 = REQ 0-4-1-2-3-5	0
ARB_SEL0	[8:7]	Arbiter 0 group priority order set 00 = REQ 1-2-3-4 01 = REQ 2-3-4-1 10 = REQ 3-4-1-2 11 = REQ 4-1-2-3	0
ARB_MODE6	[6]	Arbiter 6 group priority rotate enable 0 = Priority does not rotate, 1 = Priority rotate enable	1
ARB_MODE5	[5]	Arbiter 5 group priority rotate enable 0 = Priority does not rotate, 1 = Priority rotate enable	1
ARB_MODE4	[4]	Arbiter 4 group priority rotate enable 0 = Priority does not rotate, 1 = Priority rotate enable	1
ARB_MODE3	[3]	Arbiter 3 group priority rotate enable 0 = Priority does not rotate, 1 = Priority rotate enable	1
ARB_MODE2	[2]	Arbiter 2 group priority rotate enable 0 = Priority does not rotate, 1 = Priority rotate enable	1
ARB_MODE1	[1]	Arbiter 1 group priority rotate enable 0 = Priority does not rotate, 1 = Priority rotate enable	1
ARB_MODE0	[0]	Arbiter 0 group priority rotate enable 0 = Priority does not rotate, 1 = Priority rotate enable	1

INTERRUPT PENDING REGISTER (INTPND)

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

Like the SRCPND, this register has to be cleared in the interrupt service routine after clearing SRCPND register. We can clear a specific bit of INTPND register by writing a data to this register. It clears only the bit positions of INTPND corresponding to those set to one in the data. The bit positions corresponding to those that are set to 0 in the data remains as they are with no change.

Register	Address	R/W	Description	Reset Value
INTPND	0x39C0_0010	R/W	Indicates the interrupt request status. 0 = The interrupt has not been requested 1 = The interrupt source has asserted the interrupt request	0x00000000

NOTE: If the FIQ mode interrupt is occurred, the corresponding bit of INTPND will not be turned on. Because the INTPND register is available only for IRQ mode interrupt.

INTPND	Bit	Description	Initial State
INT_ADC	[31]	0 = Not requested, 1 = Requested	0
INT_ADC_WAKEUP	[30]	0 = Not requested, 1 = Requested	0
INT_MSTICK	[29]	0 = Not requested, 1 = Requested	0
Reserved	[28]	Not used	0
INT_IIC	[27]	0 = Not requested, 1 = Requested	0
INT_SPI	[26]	0 = Not requested, 1 = Requested	0
INT_LCD	[25]	0 = Not requested, 1 = Requested	0
INT_SDCI	[24]	0 = Not requested, 1 = Requested	0
INT_SPDIF_OUT	[23]	0 = Not requested, 1 = Requested	0
INT_UART0	[22]	0 = Not requested, 1 = Requested	0
INT_ATA	[21]	0 = Not requested, 1 = Requested	0
INT_CALM	[20]	0 = Not requested, 1 = Requested	0
INT_ECC	[19]	0 = Not requested, 1 = Requested	0
INT_LCDC[1]	[18]	0 = Not requested, 1 = Requested	0
INT_LCDC[0]	[17]	0 = Not requested, 1 = Requested	0
INT_USB_FUNC	[16]	0 = Not requested, 1 = Requested	0
INT_USB_HOST	[15]	0 = Not requested, 1 = Requested	0
INT_UART1	[14]	0 = Not requested, 1 = Requested	0
Reserved	[13]	Not used	0
INT_PRI_RTC	[12]	0 = Not requested, 1 = Requested	0
INT_ALARM_RTC	[11]	0 = Not requested, 1 = Requested	0
INT_DMA	[10]	0 = Not requested, 1 = Requested	0
INT_TIMER D	[9]	Not used	0
INT_TIMER C	[8]	0 = Not requested, 1 = Requested	0
INT_TIMER B	[7]	0 = Not requested, 1 = Requested	0
INT_WDT	[6]	0 = Not requested, 1 = Requested	0
INT_TIMER A	[5]	0 = Not requested, 1 = Requested	0
EINTG	[4]	0 = Not requested, 1 = Requested	0
EINT_VBUS	[3]	0 = Not requested, 1 = Requested	0
EINT2	[2]	0 = Not requested, 1 = Requested	0
EINT1	[1]	0 = Not requested, 1 = Requested	0
EINT0	[0]	0 = Not requested, 1 = Requested	0

INTERRUPT OFFSET REGISTER (INTOFFSET)

The number in the interrupt offset register shows that interrupt request of IRQ mode is in the INTPND register. This bit can be cleared automatically by clearing SRCPND and INTPND.

Register	Address	R/W	Description	Reset Value
INTOFFSET	0x39C0_0014	R	Indicates the IRQ interrupt request source	0x00000000

INT Source	The OFFSET value	INT Source	The OFFSET value
INT_ADC	31	INT_USB_HOST	15
INT_ADC_WAKEUP	30	INT_UART1	14
INT_MSTICK	29	Reserved	13
Reserved	28	INT_PRI_RTC	12
INT_IIC	27	INT_ALARM_RTC	11
INT_SPI	26	INT_DMA	10
INT_LCD	25	INT_TIMER D	9
INT_SDCI	24	INT_TIMER C	8
INT_SPDIF_OUT	23	INT_TIMER B	7
INT_UART0	22	INT_WDT	6
INT_ATA	21	INT_TIMER A	5
INT_CALM	20	EINTG	4
INT_ECC	19	EINT_VBUS	3
INT_LCDC[1]	18	EINT2	2
INT_LCDC[0]	17	EINT1	1
INT_USB_FUNC	16	EINT0	0

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

EXTERNAL INTERRUPT POLARITY SELECTION REGISTER

Register	Address	R/W	Description	Reset Value
EINTROL	0x39C0_0018	R/W	Indicates external interrupt polarity	0x00000000

INTPND	Bit	Description	Initial State
External Interrupt VBUS	[8]	1 = Falling edge interrupt, 0 = Rising edge interrupt	0
External Interrupt 7	[7]	1 = Falling edge interrupt, 0 = Rising edge interrupt	0
External Interrupt 6	[6]	1 = Falling edge interrupt, 0 = Rising edge interrupt	0
External Interrupt 5	[5]	1 = Falling edge interrupt, 0 = Rising edge interrupt	0
External Interrupt 4	[4]	1 = Falling edge interrupt, 0 = Rising edge interrupt	0
External Interrupt 3	[3]	1 = Falling edge interrupt, 0 = Rising edge interrupt	0
External Interrupt 2	[2]	1 = Falling edge interrupt, 0 = Rising edge interrupt	0
External Interrupt 1	[1]	1 = Falling edge interrupt, 0 = Rising edge interrupt	0
External Interrupt 0	[0]	1 = Falling edge interrupt, 0 = Rising edge interrupt	0

EXTERNAL INTERRUPT PENDING REGISTER

Register	Address	R/W	Description	Reset Value
EINTPEND	0x39C0_001C	R/W	Indicates whether external interrupts are pending.	0x00000000

INTPND	Bit	Description	Initial State
Reserved	[8]	Not used	0
External Interrupt 7	[7]	0 = No interrupt request pending, 1 = Interrupt request pending	0
External Interrupt 6	[6]	0 = No interrupt request pending, 1 = Interrupt request pending	0
External Interrupt 5	[5]	0 = No interrupt request pending, 1 = Interrupt request pending	0
External Interrupt 4	[4]	0 = No interrupt request pending, 1 = Interrupt request pending	0
External Interrupt 3	[3]	0 = No interrupt request pending, 1 = Interrupt request pending	0
Reserved	[2]	Not used	0
Reserved	[1]	Not used	0
Reserved	[0]	Not used	0

EXTERNAL INTERRUPT MASK REGISTER

Register	Address	R/W	Description	Reset Value
EINTMSK	0x39C0_0020	R/W	Indicates whether external interrupts are masked	0x00000000

INTPND	Bit	Description	Initial State
External Interrupt VBUS	[8]	0 = External interrupt disable 1 = External interrupt enable	0
External Interrupt 7	[7]	0 = External interrupt disable 1 = External interrupt enable	0
External Interrupt 6	[6]	0 = External interrupt disable 1 = External interrupt enable	0
External Interrupt 5	[5]	0 = External interrupt disable 1 = External interrupt enable	0
External Interrupt 4	[4]	0 = External interrupt disable 1 = External interrupt enable	0
External Interrupt 3	[3]	0 = External interrupt disable 1 = External interrupt enable	0
External Interrupt 2	[2]	0 = External interrupt disable 1 = External interrupt enable	0
External Interrupt 1	[1]	0 = External interrupt disable 1 = External interrupt enable	0
External Interrupt 0	[0]	0 = External interrupt disable 1 = External interrupt enable	0

NOTES

SAMSUNG CONFIDENTIAL

7

MEMORY INTERFACE UNIT (MIU)

FUNCTIONAL DESCRIPTION

FEATURES

MIU supports various **SDR/DDR** SDRAM interfaces and SRAM-like device interfaces. Specially, MIU makes an effort in hiding SDRAM access overheads. Also, MIU supports synchronous ROM and SRAM interface. Main features of MIU are as followings.

- supports 4 SDRAM access areas and 4 SRAM access areas. Each SDRAM access areas have 128Mbyte address space respectively and each SRAM access areas have 32Mbyte address space respectively.
- supports 32bit and 16bit data width in SDR and 16bit data width in DDR SDRAM access.
- supports various SDRAM modules. MIU has configuration registers for column bit width and the number of banks.
- supports 8 burst mode and full-page mode.
- schedules SDRAM commands to hide overheads of pending SDRAM accesses.
- supports 16bit and 8bit SRAM accesses.
- supports SRAM-like device interfaces. SRAM-like devices have the same access protocols with chip select, output enable and write enable signals as SRAM module.
- supports 1 ROM access area and 1 SRAM access area. Each ROM and SRAM access area have 32Mbyte address space respectively.
- supports 32bit data width in SRAM and ROM access.

SDR/DDR SDRAM Access

An access to SDRAM requires latencies as RAS to CAS delay, CAS latency, and precharge time besides data transfer time. MIU focuses on hiding these latencies between consecutive accesses to different banks.

To hide these latencies, SDRAM controller should have a pending SDRAM access. First, to hide the RAS to CAS delay, MIU uses one of good features of SDRAM, that is, SDRAM can accept the RAS command of a new SDRAM access to different banks during SDRAM access phase. As using this SDRAM feature, MIU can hide the RAS to CAS delay of a pending SDRAM access to other banks. Second, to hide precharge time, MIU serves the pending SDRAM access before issuing the precharge command of a current access, and then issues the precharge command of the current access on the idle SDRAM command cycle. Last, the pending access can also help SDRAM controller hide CAS latency. Because, SDRAM controller arranges properly the RAS and CAS command of the pending access and may transfer the data of the pending access successively to the data of the current access. As the above description, if MIU has a pending access to other banks before the completion of a current SDRAM access, SDRAM controller can hide the latencies. To have a pending access, SDRAM controller supports the new interface protocol with AHB+ that a new SDRAM access can be issued to SDRAM controller before the completion of a preceding access.

SDRAM access time is optimized when the burst length of SDRAM accesses is equal to or greater than 3. The burst length of an SDRAM access is an important factor to hide the above latencies. An SDRAM access needs 3 basic commands – RAS command, CAS command, and precharge command. To hide the latencies, the 3 basic

commands are executed in parallel with data transfer phases of its own access or the other accesses. This means that when the burst size of an SDRAM access is less than 3, the number of cycles for issuing SDRAM command is greater than that of cycles to transfer the data and therefore, the possibility of hiding the latencies decreases. So, MIU recommends the SDRAM access requester to use the burst length with equal to or greater than 3.

Even though the burst length of SDRAM access is greater than or equal to 3, there exists an overhead that cannot be hidden. The transition of the direction of data transfer – from SDRAM read access to SDRAM write access or from SDRAM write access to SDRAM read access – needs additional cycles for preventing data conflict and these are unavoidable overheads.

MIU should have the ability of interleaving the SDRAM commands among continuous SDRAM accesses. SDRAM controller arranges these SDRAM commands by the following rules.

- SDRAM commands keep the sufficient conditions of SDRAM operating parameters.
- The priority of the same commands among different accesses is the same with the order of the SDRAM accesses that bus masters issue through AHB+ bus interface block.
- The priority of the different SDRAM commands for different banks follows the rule that (RAS command > CAS command > Precharge command).

The following is the summary of main features for more efficient SDRAM access.

- MIU hides the latency of consecutive SDRAM accesses to different banks.
- MIU supports the burst length of variable size (≤ 16).
- MIU supports multiple pending SDRAM accesses.
- MIU has 6×32 bit write buffers.
- MIU completes the SDRAM access in the request order of AHB+ masters.
- MIU supports 16bit and 32bit data interface with external SDRAM for HDD-type mpeg decoder processor.

To support the burst length of variable size, MIU has the burst length counter. This counter increases when SDRAM controller samples a confirmed request at the positive edge of clock. Based on the counter, MIU accesses external SDRAM in full-page mode and stops the access after the needed burst length.

Memory Configuration With Interleaved Bank System

To hide overheads of SDRAM accesses, consecutive SDRAM accesses should be to different banks. If SDRAM accesses are uniformly distributed to different banks at a time-interval, the possibility of consecutive accesses to different banks increase and this is helpful for hiding overhead. For the uniform-distribution of accesses to different banks, block-interleaved bank method is used in SAMSUNG DVDP chip. In block-interleaved bank method, bank address is *block address % number of banks* where block address is *byte-address / block size*, and therefore, consecutive blocks have different bank address. In our chip, block size is 1K bytes because one line of a 720x480 image has 736 bytes data.

Figure 7-1 shows overall address configuration in S5L8700X.

Figure 7-2 shows core SDRAM address configuration for block-interleaved bank system.

Figure 7-3 shows memory mapped IO configuration of SFRS area in S5L8700X.

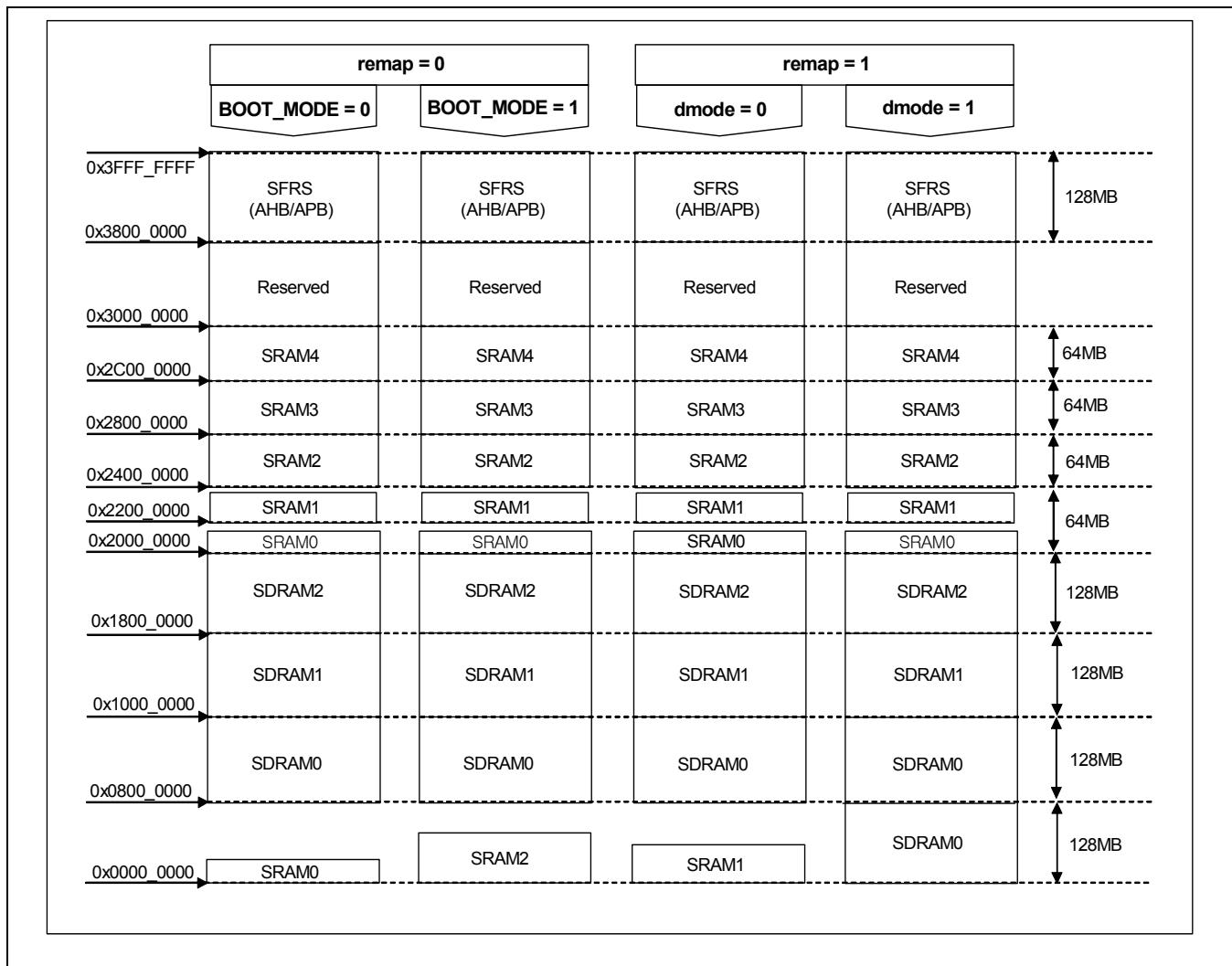


Figure 7-1. Overall Address Configuration

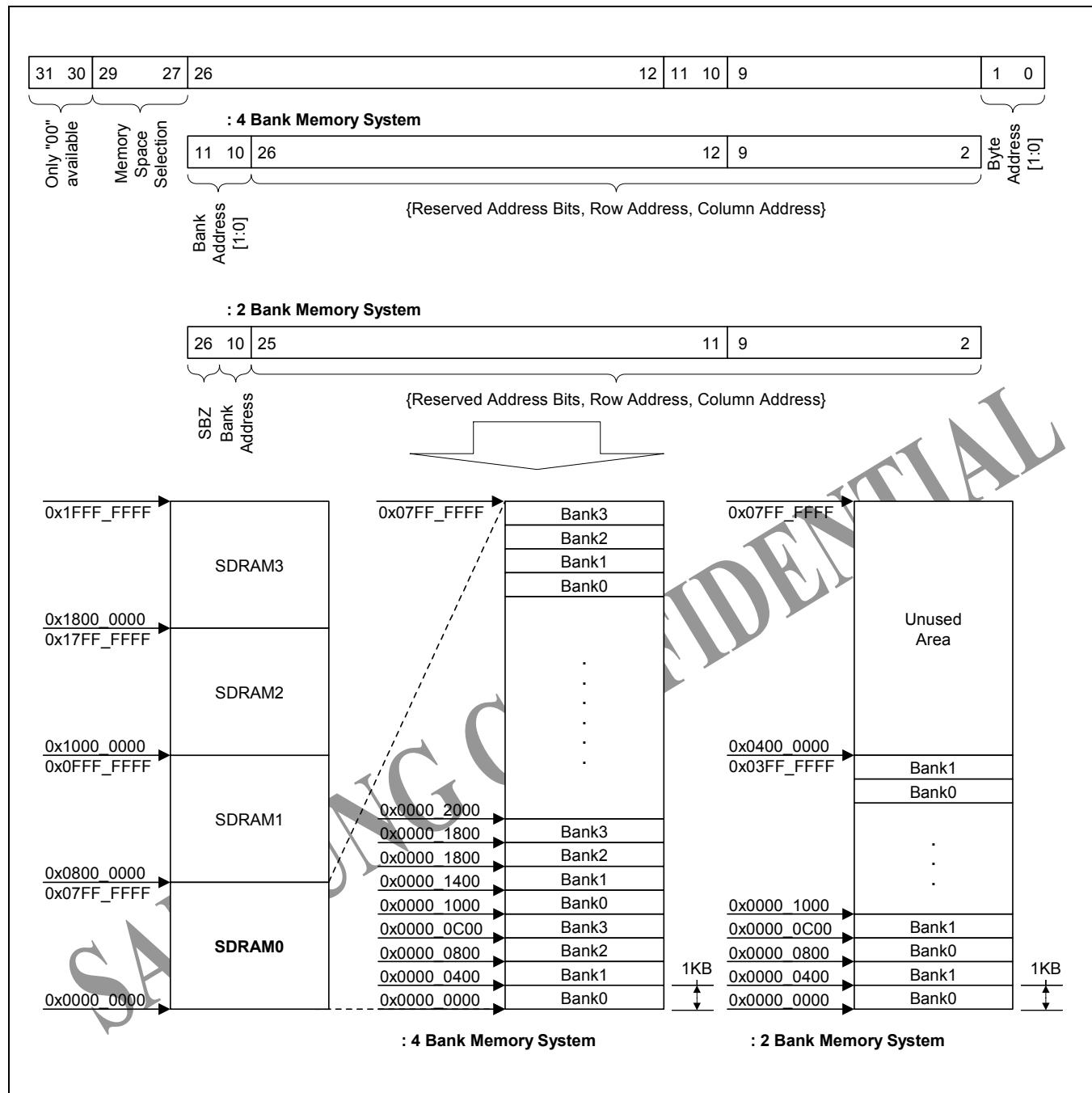


Figure 7-2. SDR/DDR Memory Configuration

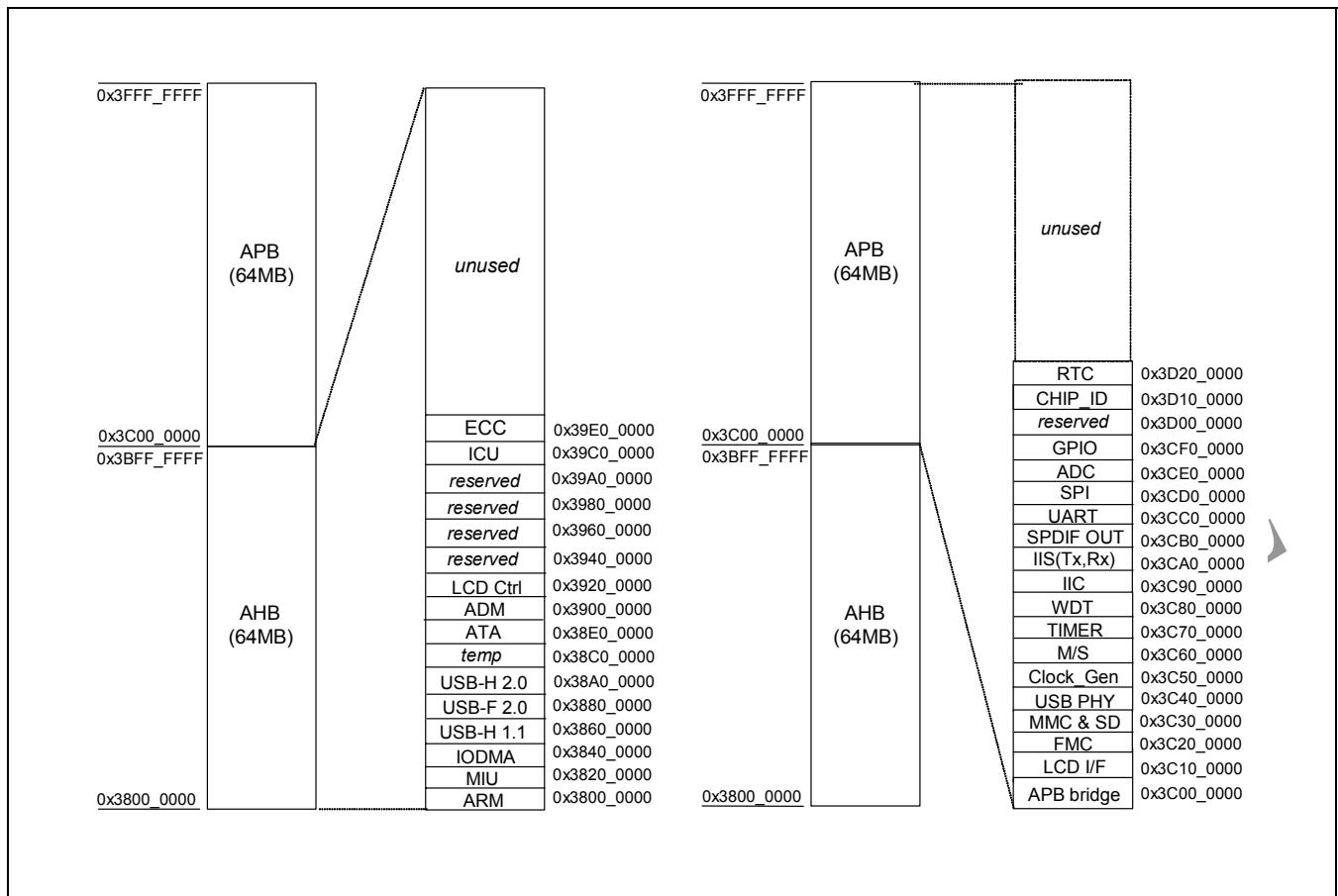


Figure 7-3. Memory Mapped IO Configuration

BLOCK DIAGRAM

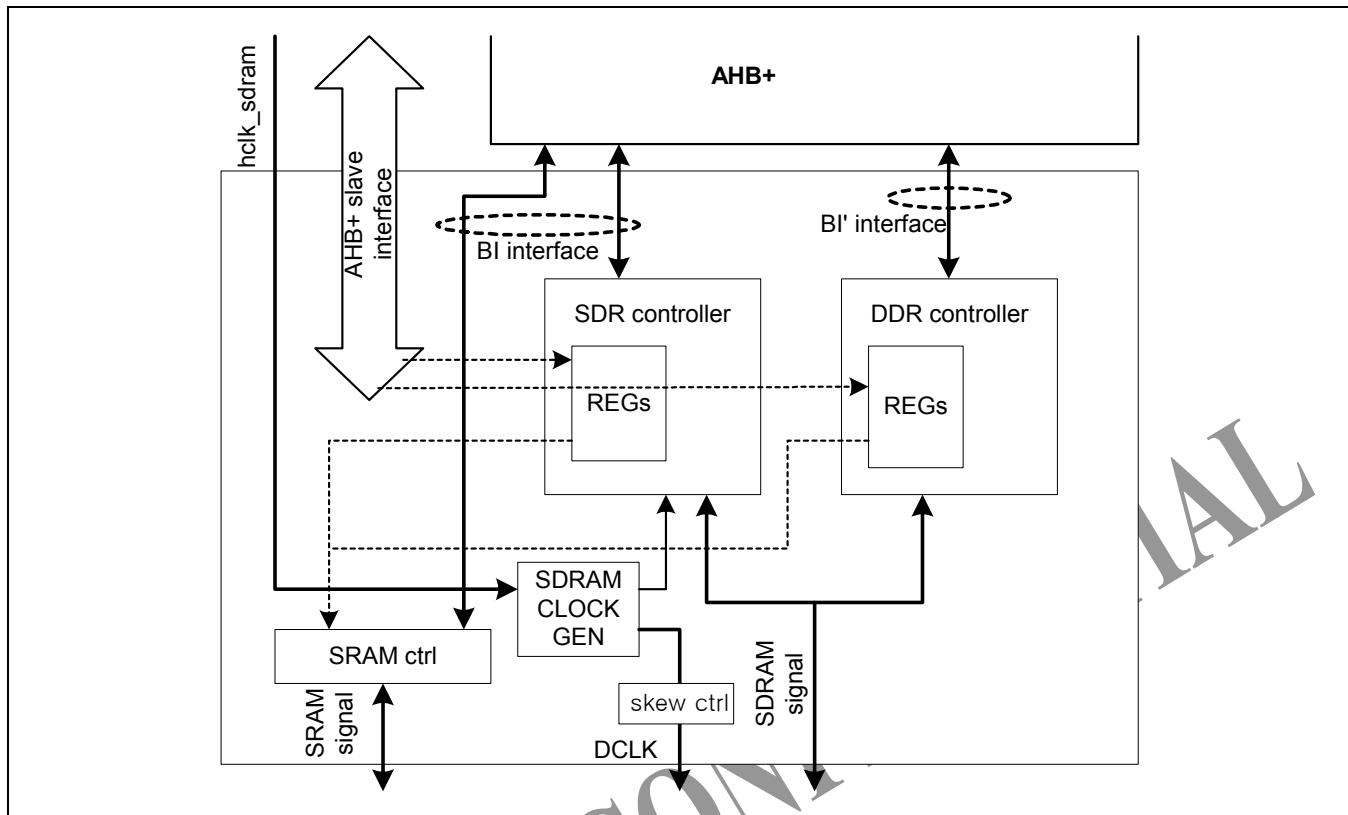


Figure 7-4. Architectural Block Diagram

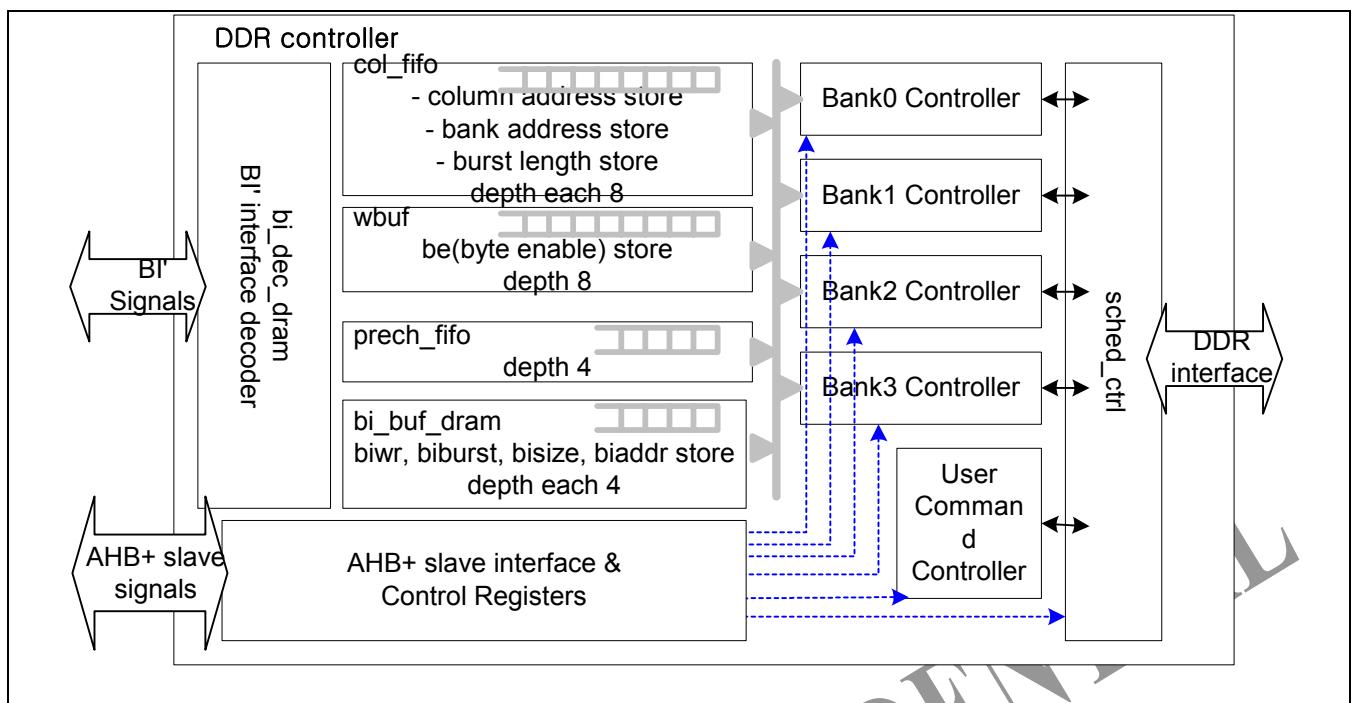


Figure 7-5. DDR SDRAM controller Block Diagram

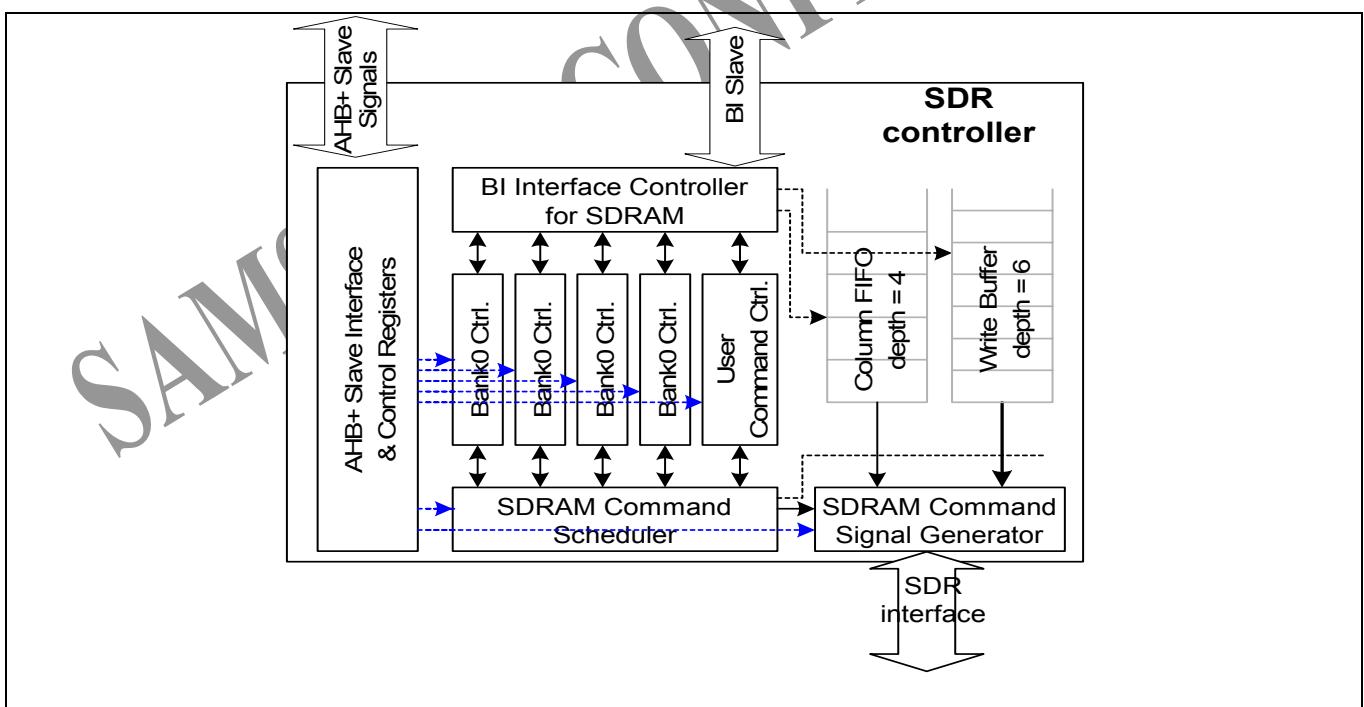


Figure 7-6. SDR SDRAM controller Block Diagram

REGISTER MAP(SDR)**External Memory Configuration Register**

Register	Address	R/W	Description	Width
MIUCON	0x3820_0000	R/W	External Memory configuration register	32 bits
Bits	Field	Description		Reset Value
[31:17]	-	Don't use. These bits appear as zero when read.		-
[16]	DMODE	If remap is 1, DMODE select the SRAM or DRAM for address "0000_0000" (Reference Figure 1, overall address configuration.)		0
[15]	-	Don't use. These bits appear as zero when read.		-
[14]	SDRAM clock invert	If necessary, you can invert the SDRAM clock for high-speed sram		0
[13]	SDRAM clock divided by 2	If 0, SDRAM clock period is similar to BUS clock. If 1, SDRAM clock period is 2*(BUS clock).		0
[12]	-	Don't use. This bit must be set to LOW .		0
[11:10]	SDRAM column width	The size of SDRAM column bit width		00b
		00	10 bits	
		01	9 bits	
		10	8 bits	
		11	7 bits	
[9]	SDRAM not full page	When using full page mode SDRAM, set 0. Otherwise, set 1.		0b
[8]	SDRAM data bus width	0	This value is not supported.	0b
		1	16 bit external data bus ** This bit must be set to HIGH	
[7]	-	Don't use. These bits appear as zero when read.		-
[6:4]	reserved	Don't use.		000b
[3]	-	Don't use. These bits appear as zero when read.		-

(Continued)

Bits	Field	Description	Reset Value
[2:1]	SDRAM bank configuration	This field indicates SDRAM bank configuration.	00b
		00 4 bank memory system with 4 bank SDRAM modules	
		01 Reserved.	
		10 4 bank memory system with 2 bank SDRAM modules	
		11 2 bank memory system with 2 bank SDRAM modules	
[0]	remap	If 1, address space of 0x0000_0000~0x07FF_FFFF is interchanged with address space of SRAM1 or SDRAM0	0b

Command and Status Register

Register	Address	R/W	Description	Width
MIUCOM	0x3820_0004	R/W	Command and status register	32 bits

Bits	Field	Description			Reset Value	
[31:19]	–	Don't use. These bits appear as zero when read.			–	
[18]	SDRAM self-refresh mode	0	Disabled	Read-only status bits	0b	
		1	Enabled			
[17]	SDRAM power down mode	0	Disabled		0b	
		1	Enabled			
[16]	User SDRAM command state	0	Idle		0b	
		1	Busy			
[15:11]	–	Don't use. These bits appear as zero when read.			–	
[10:8]	User SDRAM command	000	No action		000b	
		001	Mode register set			
		010	Precharge all banks			
		011	Auto-refresh			
		100	Enter self-refresh mode			
		101	Enter power down mode			
		110	Exit from self-refresh mode			
		111	Exit from power down mode			
[7:2]	–	Don't use. These bits appear as zero when read.			–	
[1]	SDRAM CKE enable	0	CKE is always low		0b	
		1	CKE can be controlled by SDRAM controller			
[0]	SDRAM clock output enable	When 0, CK and CK# is tied to 0 and 1, respectively.			0b	

Auto-refresh control register

Register	Address	R/W	Description	Width
MIUAREF	0x3820_0008	R/W	Auto-refresh control register	32 bits

Bits	Field	Description		Reset Value
[31:13]	–	Don't use. These bits appear as zero when read.		–
[12]	SDRAM auto-refresh enable	0	Disable	0b
		1	Enable	
[11:0]	SDRAM auto-refresh count	$\leq \left\lceil \frac{\text{minimum refresh period}}{\# \text{ of row entry} \times \text{clock period}} \right\rceil - 1$ This field should not be zero.		XXh

The example of auto-refresh count value

If the size of SDRAM row address bit width is 12, minimum refresh period is 63msec and clock period is 10 ns,

$$\text{SDRAM auto-refresh count is less than or equal to } \left\lceil \frac{63\text{ms}}{2^{12} \times 10\text{ns}} \right\rceil - 1 = 1538.$$

SDRAM Mode Register Set Value Register

Register	Address	R/W	Description	Width
MIUMRS	0x3820_000C	R/W	SDRAM Mode register set value register	32 bits

Bits	Field	Description	Reset Value
[31:15]	–	Don't use. These bits appear as zero when read.	–
[14:0]	OP code	{BA[1:0], ADDR[12:0]}	XXXXh

SDRAM Parameter Register

Register	Address	R/W	Description	Width
MIUSDPARA	0x3820_0010	R/W	SDRAM parameter register	32 bits

Bits	Field	Description		Reset Value
[31:24]	–	Don't use. These bits appear as zero when read.		–
[23:21]	CAS latency	000	Reserved	011b
		001	1	
		010	2	
		011	3	
		100	4	
		101	5	
		110	Reserved	
		111	Reserved	
[20:19]	Row active to row active delay	$t_{RRD(min)} - 1$		01b
[18:16]	RAS to read CAS delay	$t_{RCDR(min)} - 1$		010b
[15:13]	RAS to write CAS delay	$t_{RCDW(min)} - 1$		010b
[12:10]	Row precharge time	$t_{RP(min)} - 1$		010b
[9:6]	Row active time	$t_{RAS(min)} - 1$		0110b
[5:2]	Refresh cycle	$t_{RFC(min)} - 1$		1001b
[1:0]	Mode register set time	$t_{MRD(min)} - 1$		01b

REGISTER MAP (DDR)**External Memory configuration register**

Register	Address	R/W	Description	Width
MEMCONF	38200020H	R/W	External Memory configuration register	32 bits

Bits	Field	Description	Reset value
[31:12]	-	Don't use. These bits appear as zero when read.	-
[11:10]	DRAM Column Address Bit Width	The size of DRAM column address bit width	01b
		00 : 10 bits (ex) 128Mb x8, 256Mb x8, 512Mb x16	
		01 : 9 bits (ex) 128Mb x16, 256Mb x16	
		10 : 8 bits	
		11 : 7 bits	
[9:7]	-	Don't use. These bits appear as zero when read.	
[6]	Enable forcing high-Z at DM's	If 1, DM output will be high-Z when there is no operation between DRAM0. If 0, DM output will be 0 when there is no operation between DRAM.	0b
[5]	Enable forcing high-Z at DQ's	If 1, DQ output will be high-Z when there is no operation between DRAM0. If 0, DQ output will be 0 when there is no operation between DRAM.	0b
[4]	Enable forcing high-Z at DQS's	If 1, DQS output will be high-Z when there is no operation between DRAM. If 0, DQS output will be 0 when there is no operation between DRAM.	0b
[3]	-	Don't use. These bits appear as zero when read.	
[2]	Enable DQS-FF reset -	If 1, flip-flops of DQS clock domain pointer signals regarding DRAM0 will be reset every start of read operation. (When a board system is not glitch-free on DQS signal pattern, you may set this register field.)	0b
[1]	-	Don't use. This bit appears as zero when read.	-
[0]	Remap	If 1, address space of 0x0000_0000~0x07FF_FFFF is interchanged with address space of SRAM1 or SDRAM0	0b

Command and Status register

Register	Address	R/W	Description	Width
USERCMD	38200024H	R/W	Command and status register	32 bits

Bits	Field	Description			Reset value
[31:19]	-	Don't use. These bits appear as zero when read.			-
[18]	DRAM Self-refresh Mode	0 :	Disabled	Read-only status bits	0b
		1 :	Enabled		
[17]	DRAM Power Down Mode	0 :	Disabled		0b
		1 :	Enabled		

[16]	User DRAM Command State	0 : Idle		0b
		1 : Busy		
[15:11]	-	Don't use. These bits appear as zero when read.		-
[10:8]	User DRAM Command	000 :	No action	
		001 :	Mode register set	
		010 :	Precharge all banks	
		011 :	Auto-refresh	
		100 :	Enter self-refresh mode	
		101 :	Enter power down mode	
		110 :	Exit from self-refresh mode	
		111 :	Exit from power down mode	
[7:2]	-	Don't use. These bits appear as zero when read.		-
[1]	DRAM CKE Enable	0 :	CKE is always low	
		1 :	CKE can be controlled by DRAM controller	
[0]	DRAM Clock Output Enable	When 0, CK and CK# is tied to 0 and 1, respectively.		0b

Auto-refresh control register

Register	Address	R/W	Description	Width
AREF	38200028H	R/W	Auto-refresh control register	32 bits

DRAM mode register set value register

Register	Address	R/W	Description	Width
MRS	3820002CH	R/W	DRAM Mode register set value register	32 bits

Bits	Field	Description	Reset value
[31:15]	-	Don't use. These bits appear as zero when read.	-
[14:0]	OP code	{BA[1:0], ADDR[12:0]}	XXXXh

DRAM parameter register (Unit of 'tXXX' : tCK)

Register	Address	R/W	Description	Width
DPARAM	38200030H	R/W	DRAM parameter register	32 bits

Bits	Field	Description		Reset value
[31:21]	-	Don't use. These bits appear as zero when read.		-
[20:19]	CAS Latency	00	1	0b
		01	2	
		10	2.5	
		11	3	
[18:17]	RAS to CAS delay	tRCD(min) – 2		01b

[16:15]	Row Precharge Time	tRP(min) – 2	01b
[14:12]	Row Active Time	tRAS(min) – 1	101b
[11:10]	Row Active to Row Active Delay	tRRD(min) – 1	01b
[9:8]	Write recovery time	tWR(min) - 1	01b
[7:6]	Last data in to read command	tWTR(min) - 1	00b
[5:2]	Refresh Cycle Time	tRFC(min) – 2	1000b
[1:0]	Mode Register Set Time	tMRD(min) – 1	01b

SAMSUNG CONFIDENTIAL

Static memory mode register set value register

Register	Address	R/W	Description	Width
SMEMCONF	38200034H	R/W	Static Memory Configuration	32 bits

Bits	Field	Description	Reset value
[31:30]	-	Don't use. These bits appear as zero when read.	-
[29]	SRAM1 wait signal	0 = nWAIT disable, 1 = nWAIT enable	0b
[28]	SRAM1 data bus width	If 0, SRAM1 is interfaced with 8-bit external data bus. If 1, SRAM1 is interfaced with 16-bit external data bus.	0b
[27:22]	-	Don't use. These bits appear as zero when read.	-
[21]	SRAM0 wait signal	0 = nWAIT disable, 1 = nWAIT enable	0b
[20]	SRAM0 data bus width	If 0, SRAM0 is interfaced with 8-bit external data bus. If 1, SRAM0 is interfaced with 16-bit external data bus. Reset value is dependent on MODE[0] pin.	0
[19:0]		Don't use. These bits appear as zero when read.	-

Static memory parameter registers

Register	Address	R/W	Description	Width
MIUS01PARA	38200038H	R/W	SRAM0, SRAM1 static memory parameter register (In S5L8700, SRAM0 is Nor Flash)	32 bits
MIUS23PARA	3820003CH	R/W	SRAM2 and SRAM3 static memory parameter register	32 bits

Bits	Field	Description	Reset value
[31:24]	-	Don't use. These bits appear as zero when read.	-
[23:20]	T _{ACS}	Address set-up time before chip select	1h
[19:16]	T _{COS}	Chip select set-up time before output enable	1h
[15:8]	T _{ACC}	Access cycles. This field must not be zero. Otherwise, the result of access is unpredictable.	MIUS01PARA 07h MIUS23PARA 01h
[7:4]	T _{OCH}	Chip select hold time after output enable	1h
[3:0]	T _{CAH}	Address hold time after chip select	1h

* The Unit of MIUSPARA, MIUS0PARA and MIUS1PARA fields is clock cycle.

Extra parameter register

SDR/DDR selection

Register	Address	R/W	Description	Width
MIUORG	40~43H	R/W	SDR/DDR selection	32 bits

Bits	Field	Description	Reset value
[31:18]	-	Don't use. These bits appear as zero when read.	-
[17]	SDR/DDR selection	If 1, DDR controller is issued.	0h
[16:1]	-	Don't use. These bits appear as zero when read.	-
[0]	SSTL2/LVTTL type select	1 : LVTTL, 0: SSTL2	1h

DQS/DQS-rst delay parameter

Register	Address	R/W	Description	Width
MIUDLYDQS	44~47H	R/W	DQS/DQS-rst delay parameter	32 bits

Bits	Field	Description	Reset value
[31:29]	-	Don't use. These bits appear as zero when read.	-
[28:24]	DQS-rst1 delay	Each step has ?? ns time value in typical condition	0h
[23:21]	-	Don't use. These bits appear as zero when read.	-
[20 :16]	DQS-rst0 delay	Each step has ?? ns time value in typical condition	0h
[15:13]	-	Don't use. These bits appear as zero when read.	-
[12:8]	DQS1 delay	Each step has ?? ns time value in typical condition	
[7:5]	-	Don't use. These bits appear as zero when read.	-
[4:0]	DQS0 delay	Each step has ?? ns time value in typical condition	0h

SDR/DDR Clock delay parameter

Register	Address	R/W	Description	Width
MIUDLYCLK	48~4BH	R/W	SDR/DDR Clock delay parameter	32 bits

Bits	Field	Description	Reset value
[31:5]	-	Don't use. These bits appear as zero when read.	-
[4 : 0]	SDR/DDR Clock delay	Each step has ?? ns time value in typical condition	0h

SSTL2 Drive Strength parameter for Bi-direction signal

Register	Address	R/W	Description	Width
MIU_DSS_SEL_B	4C~4FH	R/W	SSTL2 Drive Strength parameter for Bi-direction signal	32 bits

Bits	Field	Description	Reset value
[31 : 3]	-	Don't use. These bits appear as zero when read.	-
[2 : 0]	SSTL2 Drive Strength	Reset value is 50%	1h

SSTL2 Drive Strength parameter for Output signal

Register	Address	R/W	Description	Width
MIU_DSS_SEL_O	50~53H	R/W	SSTL2 Drive Strength parameter for Output signal	32 bits

Bits	Field	Description	Reset value
[31 : 3]	-	Don't use. These bits appear as zero when read.	-
[2 : 0]	SSTL2 Drive Strength	Reset value is 50%	1h

SSTL2 Drive Strength parameter for Clock signal

Register	Address	R/W	Description	Width
MIU_DSS_SEL_O	54~57H	R/W	SSTL2 Drive Strength parameter for Clock signal	32 bits

Bits	Field	Description	Reset value
[31 : 3]	-	Don't use. These bits appear as zero when read.	-
[2 : 0]	SSTL2 Drive Strength	Reset value is 50%	1h

Wide range I/O Drive Strength parameter for NOR interface

Register	Address	R/W	Description	Width
PAD_DSS_SEL_NOR	58~5BH	R/W	Wide range I/O Drive Strength parameter for NOR interface	32 bits

Bits	Field	Description	Reset value
[31 : 3]	-	Don't use. These bits appear as zero when read.	-
[2 : 0]	Drive Strength	Reset value is 4mA	7h

Wide range I/O Drive Strength parameter for ATA interface

Register	Address	R/W	Description	Width
PAD_DSS_SEL_ATA	5C~5FH	R/W	Wide range I/O Drive Strength parameter for ATA interface	32 bits

Bits	Field	Description	Reset value
[31 : 3]	-	Don't use. These bits appear as zero when read.	-
[2 : 0]	Drive Strength	Reset value is 4mA	7h

SSTL2 pad ON/OFF select

Register	Address	R/W	Description	Width
SSTL2_PAD_ON	60~63H	R/W	SSTL2 pad ON/OFF	32 bits

Bits	Field	Description	Reset value
[31 : 7]	-	Don't use. These bits appear as zero when read.	-

[6 : 4]	Clock PAD ENB	[2] SDRAM clock – 0 : ON, 1 : OFF [1] DCLKN(DDR) clock – 0 : ON, 1 : OFF [0] DCLK(DDR) clock – 0 : ON, 1 : OFF	3h
[3]	-	Don't use. These bits appear as zero when read.	-
[2 : 0]	PAD ON/OFF according to direction		2h

SAMSUNG CONFIDENTIAL

FLASH MEMORY PROGRAM GUIDE (SIP)**Software Command Sequence**

SA58700X consists of S5L8700X, 8Mbit NOR Flash memory and audio CODEC. (See Figure 1-5)
 CSO pin of S5L8700X ties NOR Flash chip enable pin and allocate address area 0x2400_0000 – 0x2410_0000 to this NOR flash.

Command Sequence		Program	Sector Erase	Block Erase	Chip Erase
1 st BUS Write Cycle	ADDR ¹	0x2400_AAAA	0x2400_AAAA	0x2400_AAAA	0x2400_AAAA
	DATA ²	0xAA	0xAA	0xAA	0xAA
2 nd BUS Write Cycle	ADDR ¹	0x2400_5554	0x2400_5554	0x2400_5554	0x2400_5554
	DATA ²	0x55	0x55	0x55	0x55
3 rd BUS Write Cycle	ADDR ¹	0x2400_AAAA	0x2400_AAAA	0x2400_AAAA	0x2400_AAAA
	DATA ²	0xA0	0x80	0x80	0x80
4 th BUS Write Cycle	ADDR ¹	WA ³	0x2400_AAAA	0x2400_AAAA	0x2400_AAAA
	DATA ²	Data	0xAA	0xAA	0xAA
5 th BUS Write Cycle	ADDR ¹		0x2400_5554	0x2400_5554	0x2400_5554
	DATA ²		0x55	0x55	0x55
6 th BUS Write Cycle	ADDR ¹		Sax ⁴	BAx ⁴	0x2400_AAAA
	DATA ²		0x30	0x50	0x10

NOTE: Address format A[31:26] means the selection of Nor Flash memory, A[25:16] must has a DC value, A[15:0].



ELECTRONICS

8

IODMA CONTROLLER

OVERVIEW

S5L8700X supports four-channel IODMA (I/O Direct Memory Access) controller that performs data transfer without core intervention. Each channel of DMA controller has two data transfer directions, which are memory-to-IO and IO-to-memory. In other words, each channel can handle the following data transfers:

1. Source is in the system bus (AHB) while destination is in the peripheral bus (APB) (ex, memory to an I/O device transfer)
2. Source is in the peripheral bus (APB) while destination is in the system bus (AHB) (ex, an I/O device to memory transfer)

In IODMA, DMA is made of repetition of read-and-write and this read-and-write is atomic and is called as a single transfer. A single transfer is a minimum indivisible unit of DMA's.

The DMA operation can be requested by either software or hardware including external DMA source.

IODMA channel

Each channel of DMA controller has four IO peripherals. We can select IO peripheral to set DMACONn[31:30] bits as table selection value.

In table, Hex value means data address of IO peripheral.

Select Channel	Sel 00	Sel 01	Sel10	Sel11
Channel 0	I2S out 0x3ca0_0010		LCD I/F 0x3c10_0040	UART0 out 0x3cc0_0020
Channel 1	LCD I/F 0x3c10_0040	FMC 0x3c20_0080	Mstick 0x3c60_1008	SDC (MMC) 0x3c30_0040
Channel 2	I2S in 0x3ca0_0038		SPI in 0x3cd0_0014	LCD I/F 0x3c10_0040
Channel 3	SPDIF out 0x3cb0_0010		SPI out 0x3cd0_0010	LCD I/F 0x3c10_0040

SINGLE TRANSFER PROTOCOL

There are 2 signals for DMA protocol between IODMA controller and an IO peripheral.

Signal Name	In/Out	Description
DMA_REQ_n	In	Low-active single transfer request signal.
DMA_ACK_n	Out	Low-active single transfer acknowledge signal.

The following FSM (Finite State Machine) is a recommended FSM for IO peripheral supporting for IODMA.

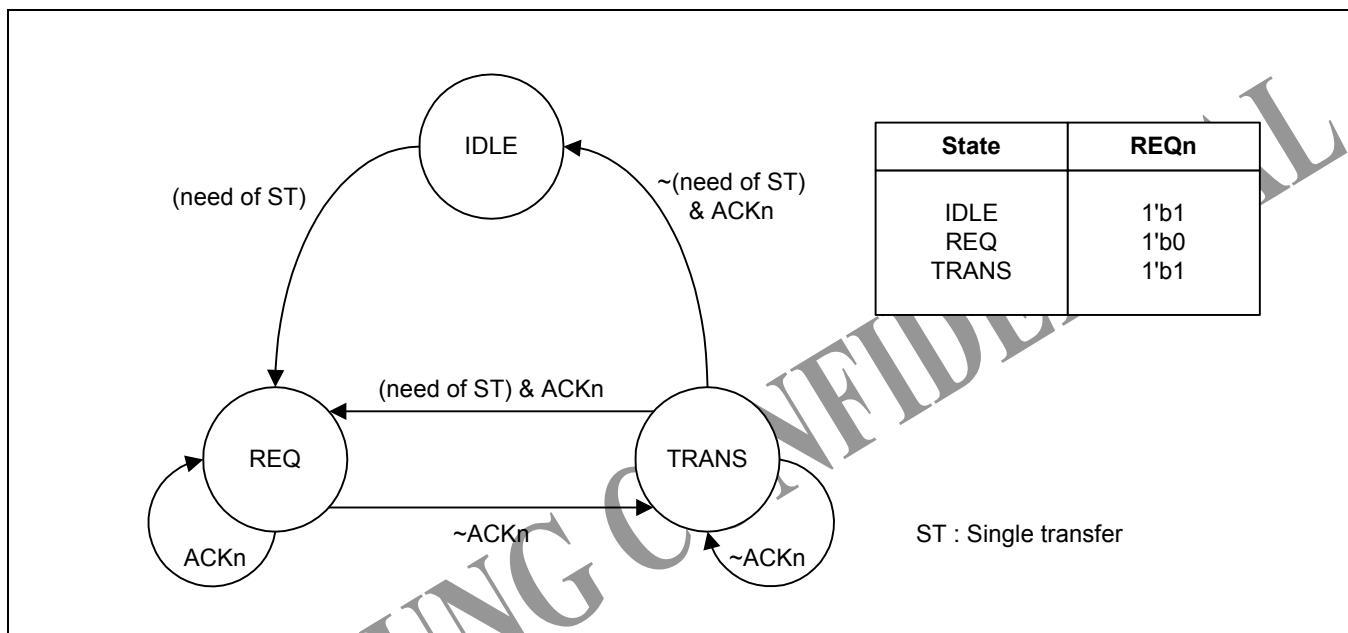


Figure 8-1. FSM for IODMA

- IDLE. As an initial state, it waits for the DMA request. If it comes, go to REQ state. At this state, DMA_ACK_n and DMA_REQ_n are high.
- REQ. In this state, DMA_REQ_n becomes low and wait for DMA_ACK_n signal asserted.
- TRANS. In this state, the IODMA executes just one single transfer.

IODMA can execute only one single transfer of 4 channels at one moment. Therefore, arbitration among channels is needed and this arbitration is executed before a single transfer is started. In arbitration, a channel with higher priority will acquire an ownership of IODMA for its single transfer. The priority of each channel is fixed. Channel 0 has the highest priority and Channel 3 has the lowest priority.

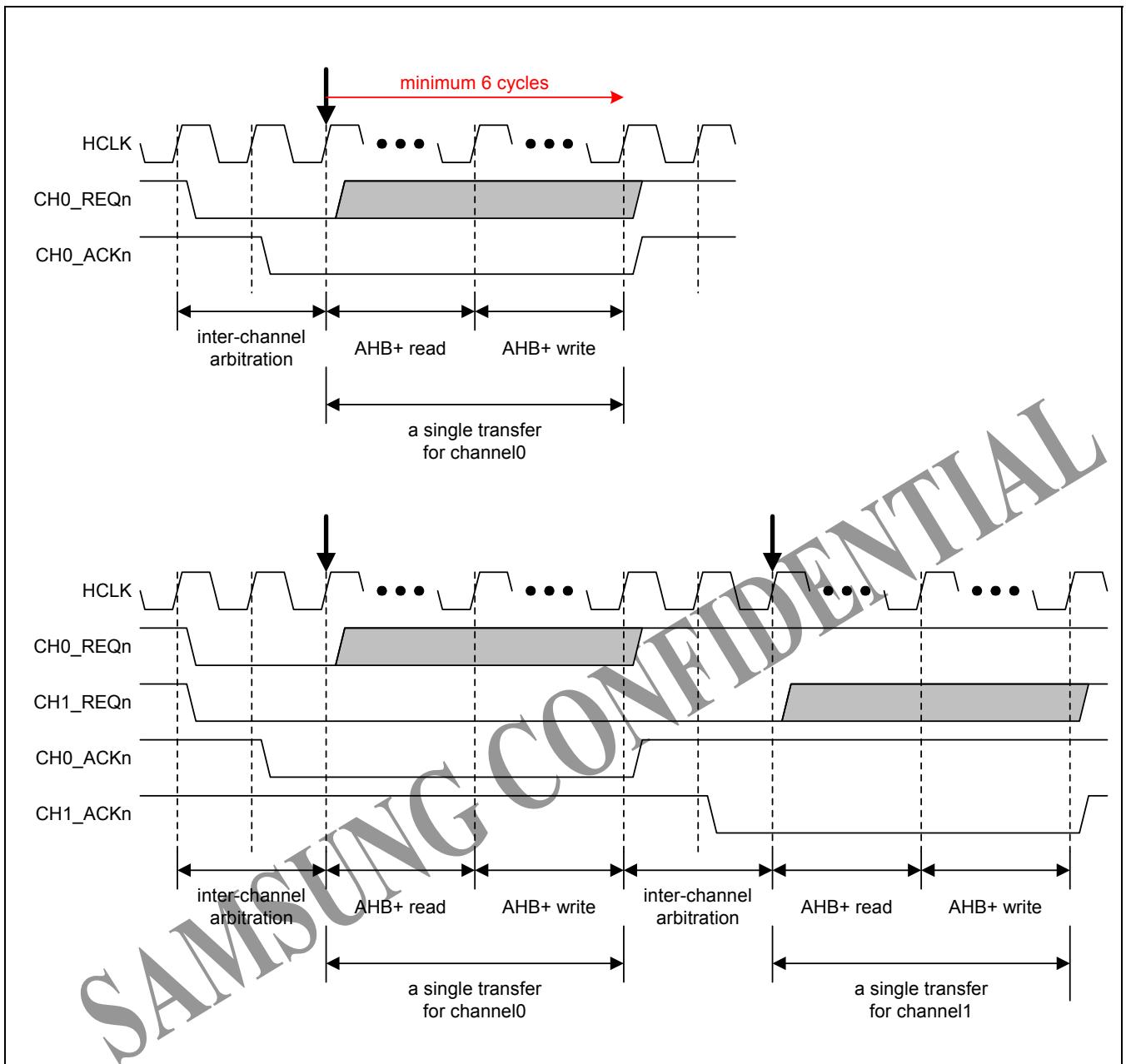


Figure 8-2. Single Data Transfer Timing

IODMA REGISTERS

BASE ADDRESS REGISTERS FOR EACH CHANNEL

Register	Address	R/W	Description	Width
DMABASE0	0x3840_0000	R/W	Base address register for channel 0	32 bits
DMABASE1	0x3840_0020	R/W	Base address register for channel 1	32 bits
DMABASE2	0x3840_0040	R/W	Base address register for channel 2	32 bits
DMABASE3	0x3840_0060	R/W	Base address register for channel 3	32 bits

Bits	Field	Description	Reset Value
[31:0]	BA[31:0]	This field is memory base address for DMA and is represented in byte addressing with 32bits. This address should be aligned to a data size of a single transfer.	00000000h

Transfer Count Registers for each Channel

Register	Address	R/W	Description	Width
DMATCNT0	0x3840_0008	R/W	Transfer count register for channel 0	32 bits
DMATCNT1	0x3840_0028	R/W	Transfer count register for channel 1	32 bits
DMATCNT2	0x3840_0048	R/W	Transfer count register for channel 2	32 bits
DMATCNT3	0x3840_0068	R/W	Transfer count register for channel 3	32 bits

Bits	Field	Description	Reset Value
[31:20]	-	This field is reserved and appears as zero when read.	-
[19:0]	TCNT[19:0]	This field is total number of single transfers.	XXXXh

CURRENT MEMORY ACCESS ADDRESS REGISTERS FOR EACH CHANNEL

Register	Address	R/W	Description	Width
DMACADDR0	0x3840_000C	R	Current memory address register for channel 0	32 bits
DMACADDR1	0x3840_002C	R	Current memory address register for channel 1	32 bits
DMACADDR2	0x3840_004C	R	Current memory address register for channel 2	32 bits
DMACADDR3	0x3840_006C	R	Current memory address register for channel 3	32 bits

Bits	Field	Description	Reset Value
[31:0]	CADDR[31:0]	This field is a current memory access address of DMA and is represented in byte addressing with 32bits.	XXXXXXXXh

Configuration Registers for each Channel

Register	Address	R/W	Description	Width
DMACON0	0x3840_0004	R/W	Configuration register for channel 0	32 bits
DMACON1	0x3840_0024	R/W	Configuration register for channel 1	32 bits
DMACON2	0x3840_0044	R/W	Configuration register for channel 2	32 bits
DMACON3	0x3840_0064	R/W	Configuration register for channel 3	32 bits

Bits	Field	Description	Reset Value																									
[31:30]	DEVSEL[1:0]	Device selection.	XXb																									
		<table border="1"> <tr> <td>Ch Sel</td> <td>00</td> <td>01</td> <td>10</td> <td>11</td> </tr> <tr> <td>0</td> <td>I2S out</td> <td></td> <td>LCD I/F</td> <td>UART0 out</td> </tr> <tr> <td>1</td> <td>LCD I/F</td> <td>FMC</td> <td>Mstick</td> <td>SDC/MMC</td> </tr> <tr> <td>2</td> <td>I2S in</td> <td></td> <td>SPI in</td> <td>LCD I/F</td> </tr> <tr> <td>3</td> <td>SPDIF out</td> <td></td> <td>SPI out</td> <td>LCD I/F</td> </tr> </table>	Ch Sel	00	01	10	11	0	I2S out		LCD I/F	UART0 out	1	LCD I/F	FMC	Mstick	SDC/MMC	2	I2S in		SPI in	LCD I/F	3	SPDIF out		SPI out	LCD I/F	
Ch Sel	00	01	10	11																								
0	I2S out		LCD I/F	UART0 out																								
1	LCD I/F	FMC	Mstick	SDC/MMC																								
2	I2S in		SPI in	LCD I/F																								
3	SPDIF out		SPI out	LCD I/F																								
[29]	DIR	DMA direction 0 : IO to Memory DMA, 1 : Memory to IO DMA	Xb																									
[28]	ADDRCON	Memory address control bit. This bit has to be "0" This field indicates how to determine memory access address for next single transfer. <table border="1"> <tr> <td>0</td> <td>Normal mode. The relationship of next memory access address (NADDR) with current memory access address (CADDR) is NADDR = CADDR + 1 * size_of(ST³).</td> </tr> </table>	0	Normal mode. The relationship of next memory access address (NADDR) with current memory access address (CADDR) is NADDR = CADDR + 1 * size_of(ST ³).	Xb																							
0	Normal mode. The relationship of next memory access address (NADDR) with current memory access address (CADDR) is NADDR = CADDR + 1 * size_of(ST ³).																											

(Continued)

Bits	Field	Description	Reset Value
[27:24]	SCHCNT[3:0]	Sub-channel count. Note that channel0, channel 1, channel 2 and channel 3 should have 0 or 1 in this field. In other words, channel 1, channel 2 and channel 3 cannot have more than 2 sub-channels. Otherwise, the result is unpredictable.	XXXXb
[23:22]	DSIZE[1:0]	Data size. 00 : byte, 01 : half word, 10 : word, 11 : reserved	XXb
[21:19]	BLEN[2:0]	Burst length (BL). 000 : (BL = 1), 001 : (BL = 2), 010 : (BL = 3), 011 : (BL = 4), 100 : (BL = 5), 101 : (BL = 6), 110 : (BL = 7), 111 : (BL = 8)	XXXb
[18]	RELOAD	Reload enable. If “1”, after whole completion of DMA, channel controller automatically restart the same DMA without commands.	Xb
[17]	HCOMINT	Half completion interrupt enable. If “1”, channel controller make interrupt to inform core on half completion of DMA. This is used for double buffering. Half completion is checked with following inequality. $(DMASTATx[19:0] \geq DMATCNTx[19:1])$	Xb
[16]	WCOMINT	Whole completion interrupt enable. If “1”, channel controller inform core on the completion of DMA by interrupt.	Xb
[15:0]	OFFSET[15:0]	Offset value. This field is used to calculating next memory access address in sub-channel mode.	XXXXh

CURRENT TRANSFER COUNT REGISTERS FOR EACH CHANNEL

Register	Address	R/W	Description	Width
DMACTCNT0	0x3840_0010	R	Current transfer count register for channel 0	32 bits
DMACTCNT1	0x3840_0030	R	Current transfer count register for channel 1	32 bits
DMACTCNT2	0x3840_0050	R	Current transfer count register for channel 2	32 bits
DMACTCNT3	0x3840_0070	R	Current transfer count register for channel 3	32 bits

Bits	Field	Description	Reset Value
[31:20]	—	This field is reserved and appears as zero when read.	—
[19:0]	CTCNT[19:0]	This field shows the number of single transfer to be remained for whole DMA transfer.	XXXXXh

NOTE: Normally, Completion of single transfer decreases CTCNT by 1.

Channel Command Registers

Register	Address	R/W	Description	Width
DMACOM0	0x3840_0014	W	Channel 0 command register	32 bits
DMACOM1	0x3840_0034	W	Channel 1 command register	32 bits
DMACOM2	0x3840_0054	W	Channel 2 command register	32 bits
DMACOM3	0x3840_0074	W	Channel 3 command register	32 bits

Bits	Field	Description	Reset Value
[31:3]	—	This field is reserved and appears as zero when read.	—
[2:0]	COM[2:0]	000 – 001 : No operation 010 : HOLD command 011 : SKIP command * HOLD command and SKIP command is only for channel 0. Other channels consider these commands as no operations. 100 : DMA channel on 101 : DMA channel off 110 : clear half completion state bit 111 : clear whole completion state bit and half completion state bit.	000b

CHANNEL 0 OFFSET2 REGISTER

Register	Address	R/W	Description	Width
DMANOFF0	0x3840_0018	R/W	Channel 0 offset2 register	32 bits

Bits	Field	Description	Reset Value
[31:16]	–	This field is reserved and appears as zero when read.	–
[15:0]	OFF2CH0[15:0]	Offset2 value. This field is used for calculating next memory access address in Multi-sub channel mode.(Not Used)	XXXXh

NOTE: This register is for channel 0 and the other channels do not have offset2 register.

ALL Channel Status Register

Register	Address	R/W	Description	Width
DMAALLST	0x3840_0100	R	All channel status register	32 bits

Bits	Field	Description		Reset value
[31:15]	–	This field is reserved and appears as zero when read.		–
[14]	DMABUSY3	Channel 3	DMA busy.	0b
[13]	HCOM3		Half completion.	0b
[12]	WCOM3		Whole completion.	0b
[11]	–	This field is reserved and appears as zero when read.		–
[10]	DMABUSY2	Channel 2	DMA busy.	0b
[9]	HCOM2		Half completion.	0b
[8]	WCOM2		Whole completion.	0b
[7]	–	This field is reserved and appears as zero when read.		–
[6]	DMABUSY1	Channel 1	DMA busy.	0b
[5]	HCOM1		Half completion.	0b
[4]	WCOM1		Whole completion.	0b
[3]	HOLD_SKIP	Channel 0	When 1, this field indicates that HOLD or SKIP command is executing and is not completed.	0b
[2]	DMABUSY0		DMA busy.	0b
[1]	HCOM0		Half completion.	0b
[0]	WCOM0		Whole completion.	0b

NOTE: This register is for observing all DMA channels by just a single word read.

9

WATCHDOG TIMER

OVERVIEW

Watchdog timer does two main functions: watchdog function and start signal generation after main clock oscillation stabilization.

Watchdog function is for a situation where the system goes to an abnormal state. In this situation, if the system cannot clear the counter in the watchdog timer, the counter will count up to overflow and generate a reset signal to initialize all the system. So normal system should periodically clear the counter in the watchdog timer.

The second function of the watchdog timer is oscillation stabilization. When the system power is on, the main clock from the PLL is not a stable signal. So the system should wait until this main clock is stabilized. Waiting for an enough time to stabilize the main clock, watchdog timer generates the start signal that will start all the system.

FEATURES

- Periodic interrupt generation
- Global reset generation by watchdog timer
- Start signal generation for oscillation stabilization
- 8-bits enable/disable register

BLOCK DIAGRAM

Watchdog timer consists of internal clock divider, pre-scaler and internal counter. The clock divider divides the main clock with pre-defined divisors which are 2048, 1024, 256, and 128. The divided clocks are selected by the WDT_CS flag in the control register and pre-scaled again by 4-bits pre-scaler. The clock generated from the pre-scaler is supported to the internal 11-bits counter that will generate start signal, reset signal, and interrupt signal.

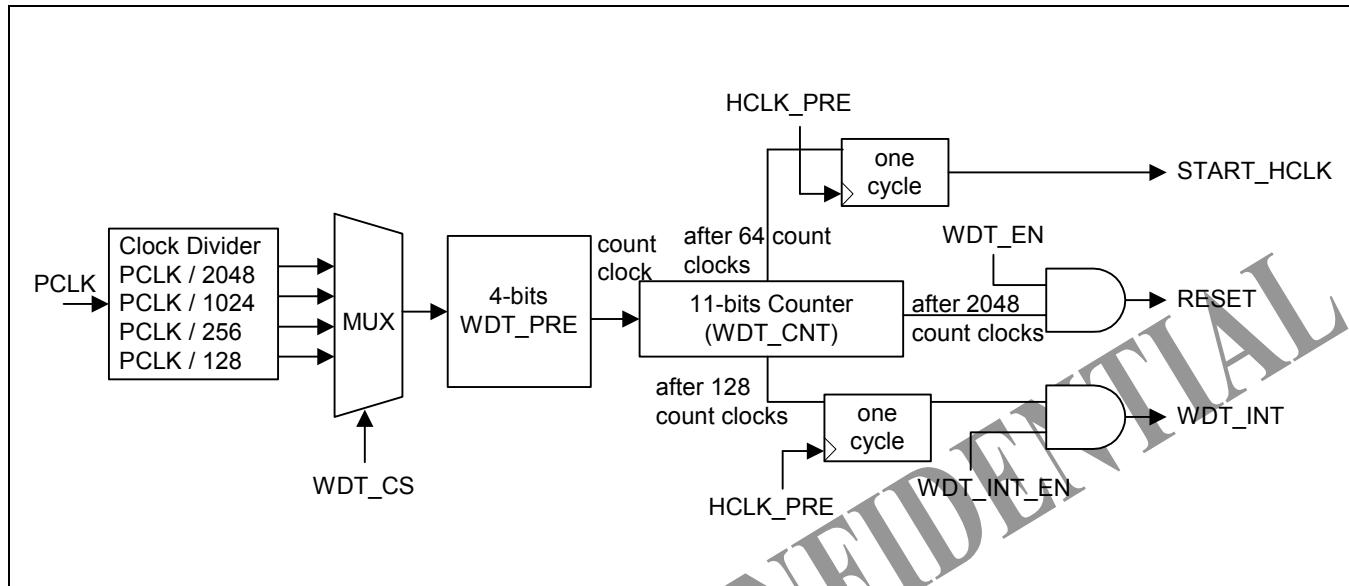


Figure 9-1. Watchdog Timer Block Diagram

REGISTERS

Name	Width	Address(Virtual)	R/W	Description	Reset
WDTCON	32	0x3C80_0000	R/W	Control Register	0x0000 0000
WDTCNT	32	0x3C80_0004	R	11-bits internal counter	0x0000 0000

WDTCON

Name	Width	Address	R/W	Description	Reset
WDTCON	32	0x3C80_0000	R/W	Control Register	0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
															WDT_PRE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															WDT_EN

Bits	Name	Type	Description
19:16	WDT_PRE	R/W	Pre-scale value
15	WDT_INT_EN	R/W	Enable or disable the periodic interrupt generation when watchdog timer is enabled. 0 = Disable the periodic interrupt generation 1 = Enable the periodic interrupt generation
14:12	WDT_CS	R/W	Clock selection 100 = PCLK / 2048 001 = PCLK / 1024 010 = PCLK / 256 011 = PCLK / 128 000 = PCLK / 8
11:8	WDT_CLR	W	Clear the internal 11-bits counter register. As this field is write-only, the read value is always zero. 1010 = Clear the internal 11-bits counter register Others = Nothing
7:0	WDT_EN	R/W	Enable the watch-dog timer 10100101 = Disable the reset/start signal generation by the watchdog timer. Others = Nothing

WDTCNT

Name	Width	Address	R/W	Description	Reset
WDTCNT	32	0x3C80_0004	R	11-bits internal counter	0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

WDT_CNT

Bits	Name	Type	Description
10:0	WDT_CNT	R	11-bits internal counter

SAMSUNG CONFIDENTIAL

PRESCALING

System clock is scaled in two stages: WDT_PRE and WDT_CS. The WDT_CS has 8 types of pre-scale value and the WDT_PRE has a 4-bits resolution for pre-scaling.

When the power is on, the clock generated from the PLL is not clean signal that should not be used for the operation of the chip. It takes 3 ~ 4 ms to stabilize the PLL. The watchdog timer in S5L8700X waits for 4.85 ms after its reset is released. The start signal will trigger the release of the all the other reset signals when the clock from the PLL is stabilized enough.

When the watchdog timer interrupt is enabled, periodic interrupt signal is generated that will be used to clear the watchdog counter.

When the chip falls to abnormal state, it should be reset. When the watchdog timer is not cleared and counts up to overflow, it generates the reset signal and reset the chip. The time is defined by the WDT_CS and WDT_PRE as shown in table 9-1, table 9-2, table 9-3 and table 9-4.

Table 9-1. Pre-scaled Clock Frequency and Periods When WDT_PRE = 0 and PCLK = 27 MHz

WDT_CS	Frequency/ Period	WDT_PRE = 0	WDT_START (x64)	WDT_INT (x128)	WDT_RESET (x2048)
		Frequency/ Period	Frequency/ Period	Frequency/ Period	Frequency/ Period
PCLK / 2048	13.18 KHz	13.18 KHz	0.21 KHz	0.10 KHz	6.44 Hz
	75.85 us	75.85 us	4.85 ms	9.71 ms	155.34 ms
PCLK / 1024	26.37 KHz	26.37 KHz	0.41 KHz	0.21 KHz	12.87 Hz
	37.93 us	37.93 us	2.43 ms	4.85 ms	77.67 ms
PCLK / 256	105.47 KHz	105.47 KHz	1.65 KHz	0.82 KHz	51.50 Hz
	9.48 us	9.48 us	0.61 ms	1.21 ms	19.42 ms
PCLK / 128	210.94 KHz	210.94 KHz	3.30 KHz	1.65 KHz	103.00 Hz
	4.74 us	4.74 us	0.30 ms	0.61 ms	9.71 ms
PCLK / 8	3.375MHz	3,375MHz	52,73KHz	26,37KHz	1,65KHz
	296.3 ms	296.3ms	18.96us	37.93us	606.81us

Table 9-2. Pre-scaled Clock Frequency and Periods When WDT_PRE = 15 and PCLK = 27 MHz

WDT_CS	Frequency/ Period	WDT_PRE = 15	WDT_START (x64)	WDT_INT (x128)	WDT_RESET (x2048)
		Frequency/ Period	Frequency/ Period	Frequency/ Period	Frequency/ Period
PCLK / 2048	13.18 KHz 75.85 us	0.82 KHz 1,213.63 us	12.87 KHz 77.67 ms	6.44 KHz 155.34 ms	0.40 Hz 2,485.51 ms
PCLK / 1024	26.37 KHz 37.93 us	1.65 KHz 606.81 us	25.75 KHz 38.84 ms	12.87 KHz 77.67 ms	0.80 Hz 1,242.76 ms
PCLK / 256	105.47 KHz 9.48 us	6.59 KHz 151.70 us	103.00 KHz 9.71 ms	51.50 KHz 19.42 ms	3.22 Hz 310.69 ms
PCLK / 128	210.94 KHz 4.74 us	13.18 KHz 75.85 us	205.99 KHz 4.85 ms	103.00 KHz 9.71 ms	6.44 Hz 155.34 ms
PCLK / 8	3.375MHz 296.3 ms	0.21MHz 5.037us	3.296KHz 322.37us	1.648KHz 644.74us	102.99Hz 10.3ms

Table 9-3. Pre-scaled Clock Frequency and Periods When WDT_PRE = 0 and PCLK = 32,768Hz

WDT_CS	Frequency/ Period	WDT_PRE = 0	WDT_START (x64)	WDT_INT (x128)	WDT_RESET (x2048)
		Frequency/ Period	Frequency/ Period	Frequency/ Period	Frequency/ Period
PCLK / 2048	16Hz 62.5ms	16Hz 62.5ms	0.25 Hz 4s	0.125 Hz 8s	0.00781Hz 128s
PCLK / 1024	32Hz 31.25ms	32Hz 31.25ms	0.5Hz 2s	0.25 Hz 4s	0.0156Hz 64s
PCLK / 256	128Hz 7.81ms	128Hz 7.81ms	2Hz 0.5s	1Hz 1s	0.0625Hz 16s
PCLK / 128	256Hz 3.906ms	256Hz 3.906ms	4Hz 0.25s	2Hz 0.5s	0.125Hz 8s
PCLK / 8*	4096Hz 244.14us	4096Hz 244.14us	64Hz 15.63ms	32Hz 31.25ms	2Hz 0.5s

NOTE: PCLK/8 mode is default after reset.

Table 9-4. Pre-scaled Clock Frequency and Periods When WDT_PRE = 15 and PCLK = 32,768Hz

WDT_CS	Frequency/ Period	WDT_PRE = 15	WDT_START (x64)	WDT_INT (x128)	WDT_RESET (x2048)
		Frequency/ Period	Frequency/ Period	Frequency/ Period	Frequency/ Period
PCLK / 2048	16Hz	1Hz	0.0156 Hz	0.00781 Hz	0.00048Hz
	62.5ms	1s	64s	128s	2048s
PCLK / 1024	32Hz	2Hz	0.03125Hz	0.0156 Hz	0.000976Hz
	31.25ms	0.5s	32s	64s	1024s
PCLK / 256	128Hz	8Hz	0.125Hz	0.0625Hz	0.0039Hz
	7.81ms	125ms	8s	16s	256s
PCLK / 128	256Hz	16Hz	0.25Hz	0.125Hz	0.00781Hz
	3.906ms	62.5ms	45s	8s	128s
PCLK / 8	4096Hz	256Hz	4Hz	2Hz	0.125Hz
	244.14us	3.906ms	0.25s	0.5ms	8s

Simulation Mode

We add simulation mode for fast and stable simulation and add two signals to wdt module. wdt_fast, wdt_noreset are these signals. If wdt_fast is set to high, wdt module output signals (wdt_start, wdt_int, wdt_reset) occurred more frequently than normal. And wdt_noreset is set to high, wdt module don't generation wdt_reset. In simulation mode, these two signals are all high.

To go to simulation mode, see chapter of mode control part.

NOTES

SAMSUNG CONFIDENTIAL

10

REAL TIMER CLOCK (RTC)

OVERVIEW

The Real Time Clock (RTC) unit can operate by the backup battery although the system power turns off. The RTC transmits data to CPU as BCD (binary coded decimal) values. The data include second, minute, hour, date, day of the week, month, and year. The RTC unit works with an external 32.768kHz crystal and also can perform the alarm function. The block diagram is shown in Figure 10-1.

Features

- 2 data transfer modes – transmission, reception
- Clock and calendar functions (BCD display) : seconds, minutes, hours, date, day of week, month, year
- Leap year generator
- Wake-up (PMWKUP) signal generation: support on the power-down mode
- Alarm interrupt (ALMINT) in normal operation mode
- Cyclic interrupts: the interrupt cycle may be 1/256 second, 1/64 second, 1/16second, 1/4 second, 1/2 second, or 1 second
- Round reset function: 30-, 40-, 50-second

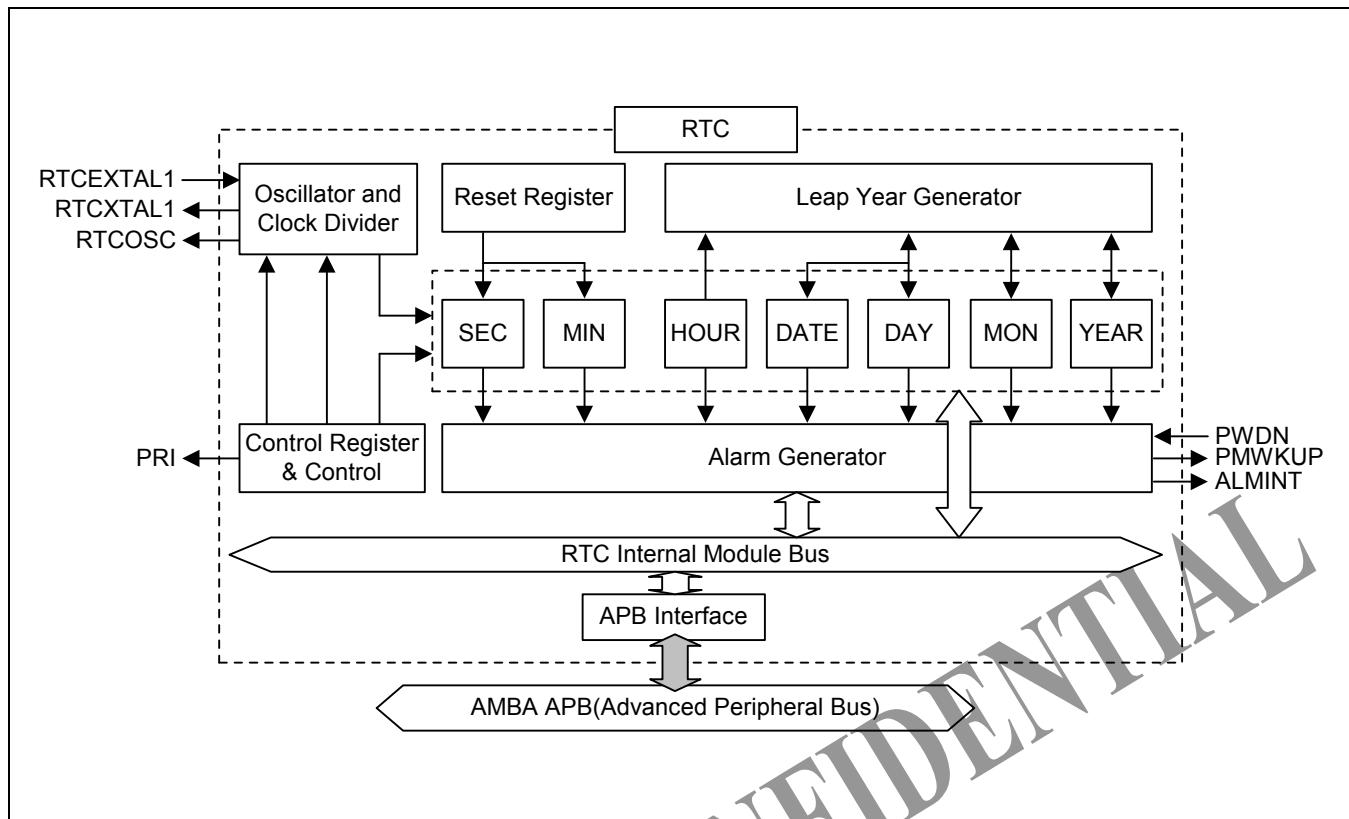


Figure 10-1. Top Block Diagram

FUNCTION DESCRIPTION

System Clock Frequency Control

The leap year generator calculates which the last date of each month is 28,29,30 or 31 that is based on data from BCDDAY, BCDMON, and BCDYEAR. This also considers leap years in deciding the last date. A 16 bit counter can just represent four BCD digits, so it can decide whether any year is a leap year or not.

System Power Operation

It is required to set bit 1 of the RTCCON register for interfacing between CPU and RTC logic. An one second error can occur when the CPU reads or writes data into BCD counters and this can cause the change of the higher time units. When the CPU reads/writes data to/from the BCD counters, another time unit may be changed if BCDSEC register is overflowed. To avoid this problem, the CPU should reset BCDSEC register to 00h. The reading sequence of the BCD counters is BCDYEAR, BCDMON, BCDDATE, BCDDAY, BCDHOUR, BCDMIN, and BCDSEC. It is required to read it again from BCDYEAR to BCDSEC if BCDSEC is zero.

Alarm Function

The RTC generates alarm signal at specified time in the power down mode or normal operation mode. In normal operation mode, the alarm interrupt (ALMINT) is activated and in the power down mode the power management wake up (PMWKUP) signal is activated. The RTC alarm register, RTCALM, determines the alarm enable and the condition of the alarm time setting. Note that the PWDN signal determines whether the normal operation or power down mode.

Round Reset Function

The round reset function can be performed by the RTC round reset register, RTCRST. You can select the round boundary (30, 40, or 50 sec) of the second carry generation and the second value is rounded to zero value in the round reset operation. For example, when the current time is 23:37:47 and the round boundary is selected as 40 sec, the round reset operation changes the current time with 23:38:00.

RTC OPERATION

Initial Settings of Registers after Power-on

Almost of all registers (except BCD registers) have initial value after the power is turned on.

Setting the Time

Figure 10-2 shows how to set the time when clock is stopped. This works when the entire calendar or clock is to be set.

First, set CLKRST and STARTB in RTCCON register and set CLKSEL, RTCEN and STARTB bits in RTCCON register. Next write BCDYEAR, BCDMON, BCDDAY, BCDHOUR, BCDMIN and BCDSEC registers.

Then reset STARTB bit.

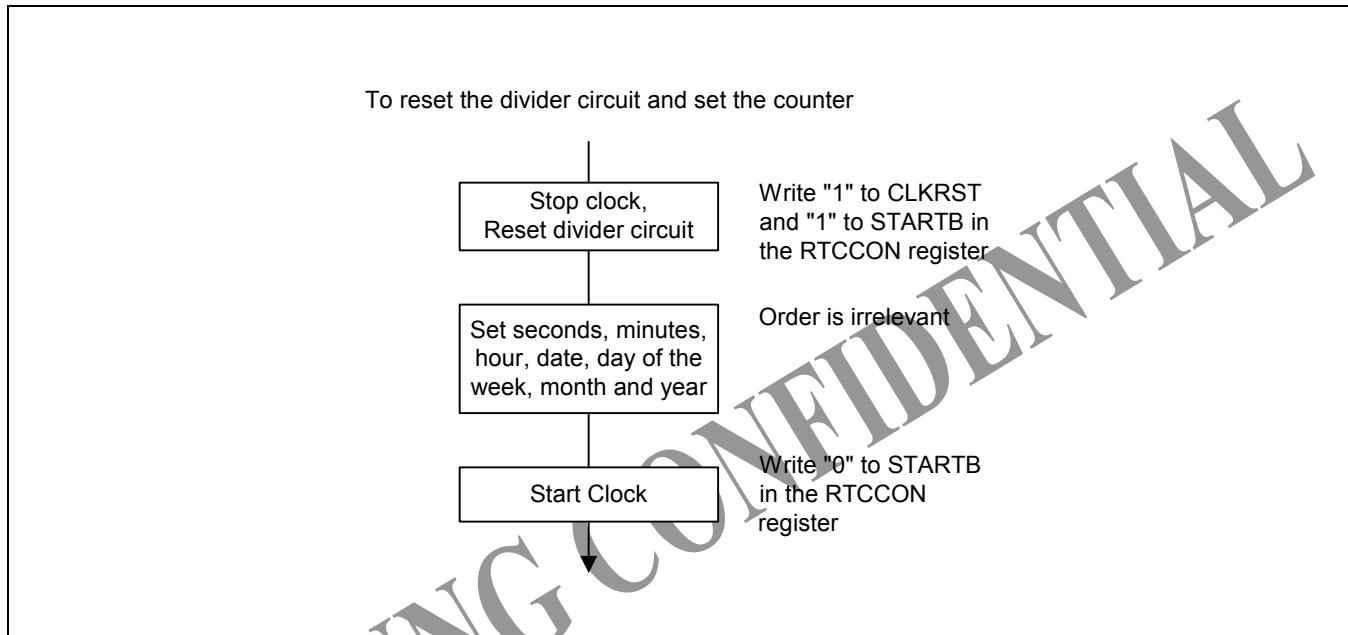


Figure 10-2. Setting the Time

Alarm Function

Figure 10-3 shows how to use the alarm function. Alarms can be generated using the seconds, minutes, hours, day of week, date, month, year or any combination of these. Set the ALMEN bit (bit 7) in the RTCALM register on which the alarm is placed to "1", and then set the alarm time. Clear the ALMEN bit in the register on which the alarm is placed to "0".

When the INTMODE bit of RTCIM register is high, and the clock and alarm times match, "1" is set in the PEND bit of RTCPEND register. The detection of alarm can be checked with reading the PEND bit.

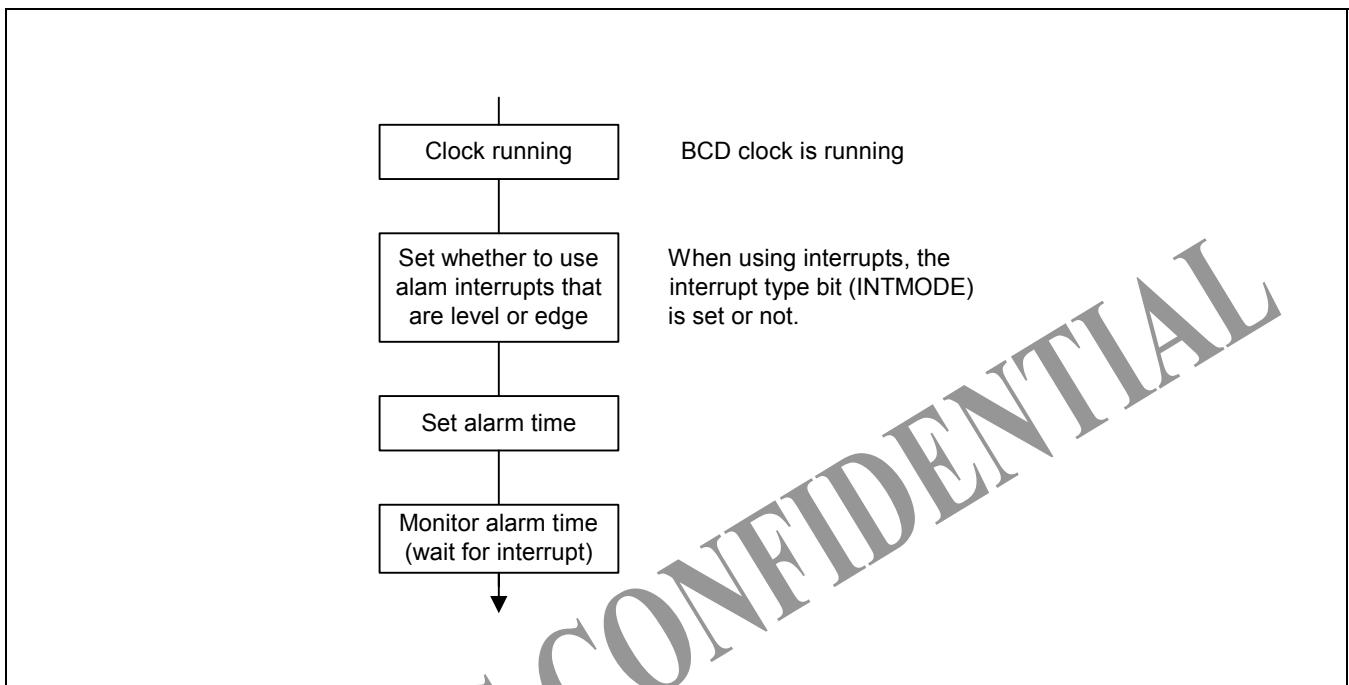


Figure 10-3. Using the Alarm Function

PROGRAMMER'S MODEL**Register Memory Map**

Register	Address	R/W	Description	Reset Value
RTCCON	0x3D20_0000	R/W	RTC Control Register	0x00
RTCrst	0x3D20_0004	R/W	RTC Round Reset Register	0x00
RTCALM	0x3D20_0008	R/W	RTC Alarm Control Register	0x00
ALMSEC	0x3D20_000C	R/W	Alarm Second Data Register	0x00
ALMMIN	0x3D20_0010	R/W	Alarm Minute Data Register	0x00
ALMHOUR	0x3D20_0014	R/W	Alarm Hour Data Register	0x00
ALMDATE	0x3D20_0018	R/W	Alarm Date Data Register	0x00
ALMDAY	0x3D20_001C	R/W	Alarm Day of Week Data Register	0x00
ALMMON	0x3D20_0020	R/W	Alarm Month Data Register	0x00
ALMYEAR	0x3D20_0024	R/W	Alarm Year Data Register	0x0000
BCDSEC	0x3D20_0028	R/W	BCD Second Register	-
BCDMIN	0x3D20_002C	R/W	BCD Minute Register	-
BCDHOUR	0x3D20_0030	R/W	BCD Hour Register	-
BCDDATE	0x3D20_0034	R/W	BCD Date Register	-
BCDDAY	0x3D20_0038	R/W	BCD Day of Week Register	-
BCDMON	0x3D20_003C	R/W	BCD Month Register	-
BCDYEAR	0x3D20_0040	R/W	BCD Year Register	-
RTCIM	0x3D20_0044	R/W	RTC Interrupt Mode Register	0x00
RTCPEND	0x3D20_0048	R/W	RTC Interrupt Pending Register	0x00

RTC Control Register (RTCCON)

RTCCON register consists of 6-bits such as STARTB that controls to run the normal counters, RTCEN that controls the read/write enable of the BCD registers, CLKSEL, CNTSEL, and CLKRST for BCD counters testing. RTCEN bit controls all interfaces between the CPU and the RTC, so it should be set to '1' in an initialization routine to enable data transfer after a system reset. Instead of working BCD with 1Hz, CLKSEL bit enables the operation of BCD counters with an external clock which is applied through the pin EXTAL1 to the test BCD counters. CNTSEL bit converts the dependent operation of BCD counters into independent counters for the test. CLKRST resets the frequency divided logic in the RTC.

OSCEN bit controls the path from input of Crystal to the output of divider logic. If this bit is high, the output of divider is 1 Hz clock. To purpose of testing that oscillator circuit and divider block, this bit is implemented.

RTCCON	Bit	Description	Initial State
STARTB	[0]	RTC start bit 0 : RUN 1: Halt	0
RTCEN	[1]	RTC write enable bit 0 : Disable 1: Enable	0
CLKSEL	[2]	BCD counter test clock set bit 0 : EXTAL1 divided clock (1 Hz) 1 : Reserved (EXTAL1 clock : 32.768 kHz)	0
CNTSEL	[3]	BCD count test type set bit 0 : Merge BCD counters 1: Reserved (Separate BCD counters)	0
CLKRST	[4]	RTC clock count set bit 0 : No reset 1: reset	0
OSCEN	[5]	Oscillator and Divider circuit test enable bit 0 : Disable 1 : Enable	0

RTC Alarm Control Register (RTCALM)

RTCALM register determines the alarm enable and the condition of the alarm time setting. Note that RTCALM register generates the alarm signal through both ALMINT and PMWKUP in power down mode, while only ALMINT in normal operation mode.

RTCALM	Bit	Description	Initial State
SECEN	[0]	Second alarm enable bit 0 : Disable 1 : Enable	0
MINEN	[1]	Minute alarm enable bit 0 : Disable 1 : Enable	0
HOUREN	[2]	Hour alarm enable bit 0 : Disable 1 : Enable	0
DATEEN	[3]	Date alarm enable bit 0 : Disable 1 : Enable	0
DAYEN	[4]	Day of week alarm enable bit 0 : Disable 1 : Enable	0
MONEN	[5]	Month alarm enable bit (reserved bit) For alarm function, this bit should be set.	0
YEAREN	[6]	Year alarm enable bit (reserved bit) For alarm function, this bit should be set.	0
ALMEN	[7]	Alarm global enable bit 0 : Disable 1 : Enable	0

Alarm Second Data Register (ALMSEC)

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

Alarm Minute Data Register (ALMMIN)

ALMMIN	Bit	Description	Initial State
MINDATA	[6:0]	BCD value for alarm minute bits [3:0] bit is from 0 to 9 [6:4] bit is from 0 to 5	0
Reserved	[7]		0

Alarm Hour Data Register (ALMHOUR)

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

Alarm Date Data Register (ALMDATE)

ALMDATE	Bit	Description	Initial State
DATEDATA	[5:0]	BCD value for alarm date, from 0 to 28, 29, 30, 31 (decimal : 01 ~ 31) [3:0] bit is from 0 to 9 [5:4] bit is from 0 to 3	0
Reserved	[7:6]		0

Alarm Day of Week Data Register (ALMDAY)

ALMDAY	Bit	Description	Initial State
DAYDATA	[2:0]	BCD value for alarm day bits [2:0] bit is from 0 to 6 000 : Sunday 001 : Monday 010 : Tuesday 011 : Wednesday 100 : Thursday 101 : Friday 110 : Saturday	0
Reserved	[7:3]		0

Alarm Month Data Register (ALMMON)

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

Alarm Year Data Register (ALMYEAR)

ALMYEAR	Bit	Description	Initial State
YEARDATA	[15:0]	BCD value for alarm year bits [7:0] bit is from 0 to 99 [15:8] bit is from 0 to 99	0

RTC Round Reset Register (RTCRST)

RTCRST	Bit	Description	Initial State
SECCR	[2:0]	Round boundary for second carry generation bits 011 : over than 30 sec 100 : over than 40 sec 101 : over than 50 sec	0
SRSTEN	[3]	Round second reset enable bit 0 : Disable 1 : Enable	0
Reserved	[7:5]		0

BCD Second Data Register (BCDSEC)

BCDSEC	Bit	Description	Initial State
SECDATA	[6:0]	BCD value for second bits [3:0] bit is from 0 to 9 [6:4] bit is from 0 to 5	Undef.
Reserved	[7]		-

BCD Minute Data Register (BCDMIN)

BCDMIN	Bit	Description	Initial State
MINDATA	[6:0]	BCD value for minute bits [3:0] bit is from 0 to 9 [6:4] bit is from 0 to 5	Undef.
Reserved	[7]		-

BCD Hour Data Register (BCDHOUR)

BCDHOUR	Bit	Description	Initial State
HOURDATA	[5:0]	BCD value for hour bits [3:0] bit is from 0 to 9 [5:4] bit is from 0 to 2	Undef.
Reserved	[7:6]		-

BCD Date Data Register (BCDDATE)

BCDDATE	Bit	Description	Initial State
DATEDATA	[5:0]	BCD value for date bits(decimal : 01 ~ 31) [3:0] bit is from 0 to 9 [5:4] bit is from 0 to 3	Undef.
Reserved	[7:6]		-

BCD Day of Week Data Register (BCDDAY)

BCDDAY	Bit	Description	Initial State
DATEDAY	[2:0]	BCD value for date bits [2:0] bit is from 0 to 6 000 : Sunday 001 : Monday 010 : Tuesday 011 : Wednesday 100 : Thursday 101 : Friday 110 : Saturday	Undef.
Reserved	[7:3]		-

BCD Month Data Register (BCDMON)

BCDMON	Bit	Description	Initial State
MONDATA	[4:0]	BCD value for month bits [3:0] bit is from 0 to 9 [4] bit is from 0 to 1	Undef.
Reserved	[7:5]		-

BCD Year Data Register (BCDYEAR)

BCDYEAR	Bit	Description	Initial State
YEARDATA	[15:0]	BCD value for year bits [7:0] bit is from 0 to 99 [15:8] bit is from 0 to 99	Undef.

RTC Interrupt Mode Register (RTCIM)

Periodic Interrupt Mode (PEIMODE): Indicates generation of interrupt with the period designated by the PES bits.
When set to "1" PEIMODE generates periodic interrupts.

RTCIM	Bit	Description	Initial State
INTMODE	[1:0]	Interrupt mode selection bit x0 : Disable alarm interrupt mode. x1 : Enable alarm interrupt mode. 01 : Supports on the edge alarm interrupt. 11 : Supports on the level alarm interrupt.	0
PEIMODE	[2]	Periodic interrupt mode bit 0 : Interrupts not generated with the period designated by the PES bits. 1 : Interrupts generated with the period designated by the PES bits.	0
PES	[5:3]	These bits specify the periodic interrupt. 000 : No periodic interrupt generated 001 : Periodic interrupt generated every 1/256 second 010 : Periodic interrupt generated every 1/64 second 011 : Periodic interrupt generated every 1/16 second 100 : Periodic interrupt generated every 1/4 second 101 : Periodic interrupt generated every 1/2 second 110 : Periodic interrupt generated every 1 second 111 : reserved	0
Reserved	[7:6]		0

RTC Interrupt Pending Register (RTCPEND)

RTCPEND	Bit	Description	Initial State
PEND	[0]	Interrupt pending enable bit 0 : PEND bit is cleared. 1 : PEND bit is pending.	0
Reserved	[7:1]		0

11

16-BIT TIMER

OVERVIEW

S5L8700X has internally four timers. They are named as timer A, timer B, timer C, timer D. The supported modes of all timers are interval mode, PWM (Pulse Width Modulation) mode, one-shot PWM mode, and capture mode. The timer has internally one counter register (TM_CNT), one pre-scale register (TM_PRE), and two data registers (TM_DATA0, TM_DATA1). The TM_CNT is incremented by a pre-scaled clock that is pre-scaled by TM_PRE value from an external clock or internally selected clock. The role of TM_DATA0 and TM_DATA1 is different for each mode.

FEATURES

- Two 16-bits timer: Timer A, Timer B, Timer C, Timer D
- All timers support interval mode, PWM mode, one-shot mode, and capture mode
- Pre-scaling the counting clock with the 10-bits pre-scale register (TM_PRE)
- 0 ~ 100 % duty ratio PWM signal generation

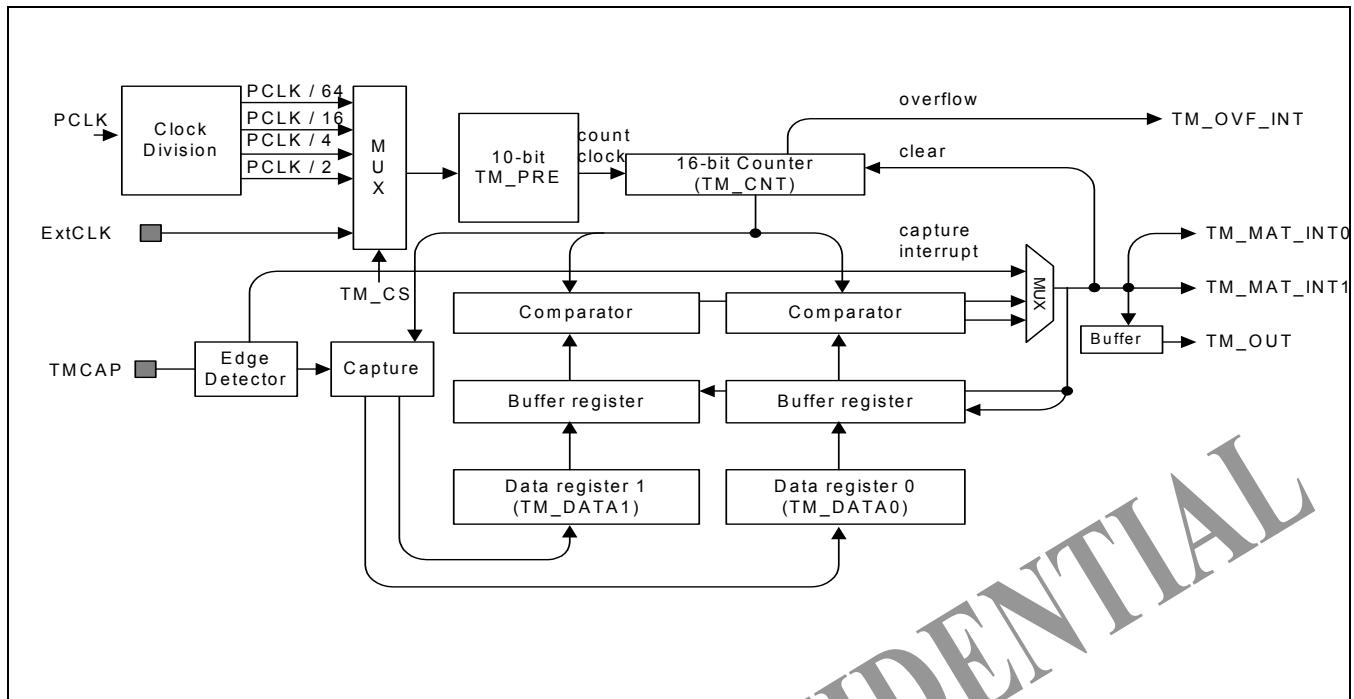
BLOCK DIAGRAM

Figure 11-1. Block Diagram for Timer

REGISTER MAP

Name	Width	Address	R/W	Description	Reset
Timer A Registers					
TACON	32	0x3C70_0000	R/W	Control Register	0x0000 0000
TACMD	32	0x3C70_0004	R/W	Command Register	0x0000 0000
TADATA0	32	0x3C70_0008	R/W	Data0 Register	0x0000 0000
TADATA1	32	0x3C70_000C	R/W	Data1 Register	0x0000 0000
TAPRE	32	0x3C70_0010	R/W	Prescale Register	0x0000 0000
TACNT	32	0x3C70_0014	R	Counter Register	0x0000 0000
Timer B Registers					
TBCON	32	0x3C70_0020	R/W	Control Register	0x0000 0000
TBCMD	32	0x3C70_0024	R/W	Command Register	0x0000 0000
TBDDATA0	32	0x3C70_0028	R/W	Data0 Register	0x0000 0000
TBDDATA1	32	0x3C70_002C	R/W	Data1 Register	0x0000 0000
TBPRE	32	0x3C70_0030	R/W	Prescale Register	0x0000 0000
TBCNT	32	0x3C70_0034	R	Counter Register	0x0000 0000
Timer C Registers					
TCCON	32	0x3C70_0040	R/W	Control Register	0x0000 0000
TCCMD	32	0x3C70_0044	R/W	Command Register	0x0000 0000
TCDATA0	32	0x3C70_0048	R/W	Data0 Register	0x0000 0000
TCDATA1	32	0x3C70_004C	R/W	Data1 Register	0x0000 0000
TCPRE	32	0x3C70_0050	R/W	Prescale Register	0x0000 0000
TCCNT	32	0x3C70_0054	R	Counter Register	0x0000 0000
Timer D Registers					
TDCON	32	0x3C70_0060	R/W	Control Register	0x0000 0000
TDCMD	32	0x3C70_0064	R/W	Command Register	0x0000 0000
TDDATA0	32	0x3C70_0068	R/W	Data0 Register	0x0000 0000
TDDATA1	32	0x3C70_006C	R/W	Data1 Register	0x0000 0000
TDPRE	32	0x3C70_0070	R/W	Prescale Register	0x0000 0000
TDCNT	32	0x3C70_0074	R	Counter Register	0x0000 0000

TIMER A REGISTERS**TACON**

Name	Width	Address	R/W	Description	Reset
TACON	32	0x3C70_0000	R/W	Control Register for timer A	0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits	Name	Type	Description
20	TA_OUT	R	The PWM output in interval/PWM/one-shot modes
18	TA_OVF	R/W	Overflow interrupt status. Although the interrupt is disabled, this field is updated when an overflow interrupt occurs. - Writing one clears this flag - Writing zero has no effect
17	TA_INT1	R/W	Match interrupt 1 status. Although the interrupt is disabled, this field is updated when a match interrupt 1 occurs. - Writing one clears this flag - Writing zero has no effect
16	TA_INT0	R/W	Match interrupt 0 status. Although the interrupt is disabled, this field is updated when a match interrupt 0 occurs. - Writing one clears this flag - Writing zero has no effect
14	TA_OVF_EN	R/W	Enable the overflow interrupt. When disabled, overflow interrupt sets the TA_OVF but it does not generate an interrupt request signal to external interrupt control unit. 0 = Disable the overflow interrupt 1 = Enable the overflow interrupt
13	TA_INT1_EN	R/W	Enable the match interrupt 1. When disabled, match interrupt 1 sets the TA_INT1 but it does not generate an interrupt request signal to external interrupt control unit. 0 = Disable the match interrupt 1 = Enable the match interrupt Rising clear mode Clear the counter when falling edge is detected.

(Continued)

Bits	Name	Type	Description
12	TA_INT0_EN	R/W	<p>Enable the match interrupt 0. When disabled, match interrupt 0 sets the TA_INT0 but it does not generate an interrupt request signal to external interrupt control unit.</p> <p>0 = Disable the match interrupt 1 = Enable the match interrupt</p> <p>Rising clear mode Clear the counter when falling edge is detected</p>
11	TA_START	R/W	In interval/PWM/one-shot modes, this field is set to the TA_OUT by clear operation (writing one to TA_CLR). In PWM and one-shot mode, whenever an MAT_INT1 occurs, this value is also updated to the TA_OUT. In interval mode, if MAT_INT0 occurs, this value is updated to the TA_OUT.
10:8	TA_CS	R/W	<p>Timer clock source selection</p> <p>000 = PCLK / 2 001 = PCLK / 4 010 = PCLK / 16 011 = PCLK / 64 10x = External clock 0 11x = External clock 1</p>
7	TA_CAP_MODE	R/W	<p>In capture mode,</p> <p>0 = Rising clear mode, Clear the counter when a rising edge is detected. 1 = Falling clear mode. Clear the counter when falling edge is detected.</p>
5:4	TA_MODE_SEL	R/W	<p>Operation mode selection</p> <p>00 = Interval mode 01 = PWM mode 10 = One-shot mode 11 = Capture mode</p>
3:0			Reserved

TACMD

Name	Width	Address	R/W	Description	Reset
TACMD	32	0x3C70_0004	R/W	Command Register for timer A	0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Bits	Name	Type	Description
1	TA_CLR	W	<p>Clear operation. This field is always zero when read.</p> <p>0 = Nothing occurs</p> <p>1 = Initialize the timer.</p> <ul style="list-style-type: none"> - Clear the counter register. - The value of TA_START is set to TA_OUT. - TA_DATA0 and TA_DATA1 are updated to the internal buffers - Initialize the state of the previously captured signal.
0	TA_EN	R/W	<p>Timer enable command</p> <p>0 = Disable the timer</p> <p>1 = Enable the timer</p>

TADATA0

Name	Width	Address	R/W	Description		Reset
TADATA0	32	0x3C70_0008	R/W	Data register		0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA_DATA0															

Bits	Name	Type	Description	
15:0	TA_DATA0	R/W	This field is used differently by the operation mode.	

Interval mode The target counting value is stored in this field. When the counter value equals to this register, a MAT_INT0 interrupt is generated. Whenever an MAT_INT0 occurs or clear operation is executed, this value is updated to the internal data buffer 0.

PWM/One-shot mode The target counting value is stored in this field. When the counter value equals to this register, a MAT_INT0 interrupt is generated. This field is updated to the internal data buffer 0 when MAT_INT0 occurs or when clear operation is executed.

Capture mode In rising-edge clear mode, the captured data at the falling edge is stored to this register.
In falling-edge clear mode, the captured data at the rising edge is stored to this register.

TADATA1

Name	Width	Address	R/W	Description		Reset
TADATA1	32	0x3C70_000C	R/W	Data register		0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA_DATA1															

Bits	Name	Type	Description												
15:0	TA_DATA1	R/W	This field is used differently by the operation mode. Interval mode Not used. PWM/ One-shot mode The target counting value is stored in this field. When the counter value equals to this register, a MAT_INT1 interrupt is generated. This field is updated to the internal data buffer 1 when MAT_INT1 occurs or when clear operation is executed. Capture mode In rising-edge clear mode, the captured data at the rising edge is stored to this register. In falling-edge clear mode, the captured data at the falling edge is stored to this register.												

TAPRE

Name	Width	Address	R/W	Description		Reset
TAPRE	32	0x3C70_0010	R/W	Pre-scale register		0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA_PRE															

Bits	Name	Type	Description												
9:0	TA_PRE	R/W	Pre-scale value												

TACNT

Name	Width	Address	R/W	Description	Reset
TACNT	32	0x3C70_0014	R	Counter register	0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA_CNT															

Bits	Name	Type	Description
15:0	TA_CNT	R	Counter register

TIMER B~D REGISTERS**TBCON, TCCON, TDCON**

Name	Width	Address	R/W	Description	Reset
TBCON	32	0x3C70_0020	R/W	Control Register for timer B	0x0000 0000
TCCON	32	0x3C70_0040	R/W	Control Register for timer C	0x0000 0000
TDCON	32	0x3C70_0060	R/W	Control Register for timer D	0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits	Name	Type	Description
20	OUT	R	The PWM output in interval/PWM/one-shot modes
18	OVF	R/W	Overflow interrupt status. Although the interrupt is disabled, this field is updated when an overflow interrupt occurs. - Writing one clears this flag - Writing zero has no effect
17	INT1	R/W	Match interrupt 1 status. Although the interrupt is disabled, this field is updated when a match interrupt 1 occurs. - Writing one clears this flag - Writing zero has no effect
16	INT0	R/W	Match interrupt 0 status. Although the interrupt is disabled, this field is updated when a match interrupt 0 occurs. - Writing one clears this flag - Writing zero has no effect
14	OVF_EN	R/W	Enable the overflow interrupt. When disabled, overflow interrupt sets the OVF but it does not generate an interrupt request signal to external interrupt control unit. 0 = Disable the overflow interrupt 1 = Enable the overflow interrupt
13	INT1_EN	R/W	Enable the match interrupt 1. When disabled, match interrupt 1 sets the INT1 but it does not generate an interrupt request signal to external interrupt control unit. 0 = Disable the match interrupt 1 = Enable the match interrupt Rising clear mode Clear the counter when falling edge is detected.

(Continued)

Bits	Name	Type	Description
12	INT0_EN	R/W	<p>Enable the match interrupt 0. When disabled, match interrupt 0 sets the INT0 but it does not generate an interrupt request signal to external interrupt control unit.</p> <p>0 = Disable the match interrupt 1 = Enable the match interrupt</p> <p>Rising clear mode Clear the counter when falling edge is detected</p>
11	START	R/W	In interval/PWM/one-shot modes, this field is set to the OUT by clear operation (writing one to CLR). In PWM and one-shot mode, whenever an MAT_INT1 occurs, this value is also updated to the OUT. In interval mode, if MAT_INT0 occurs, this value is updated to the OUT.
10:8	CS	R/W	<p>Timer clock source selection</p> <p>000 = PCLK / 2 001 = PCLK / 4 010 = PCLK / 16 011 = PCLK / 64 10x = External clock 11x = reserved</p>
7	CAP_MODE	R/W	<p>In capture mode,</p> <p>0 = Rising clear mode, Clear the counter when a rising edge is detected. 1 = Falling clear mode. Clear the counter when falling edge is detected.</p>
5:4	MODE_SEL	R/W	<p>Operation mode selection</p> <p>00 = Interval mode 01 = PWM mode 10 = One-shot mode 11 = Capture mode</p>
3:0			Reserved

TBCMD, TCCMD, TDCMD

Name	Width	Address	R/W	Description	Reset
TBCMD	32	0x3C70_0024	R/W	Command Register for timer B	0x0000 0000
TCCMD	32	0x3C70_0044	R/W	Command Register for timer C	0x0000 0000
TDCMD	32	0x3C70_0064	R/W	Command Register for timer D	0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Bits	Name	Type	Description
1	CLR	W	<p>Clear operation. This field is always zero when read.</p> <p>0 = Nothing occurs</p> <p>1 = Initialize the timer.</p> <ul style="list-style-type: none"> - Clear the counter register. - The value of START is set to OUT. - DATA0 and DATA1 are updated to the internal buffers - Initialize the state of the previously captured signal.
0	EN	R/W	<p>Timer enable command</p> <p>0 = Disable the timer</p> <p>1 = Enable the timer</p>

TBDATA0, TCDATA0, TDDATA0

Name	Width	Address	R/W	Description	Reset
TBDATA0	32	0x3C70_0028	R/W	Data register	0x0000 0000
TCDATA0	32	0x3C70_0048	R/W	Data register	0x0000 0000
TDDATA0	32	0x3C70_0068	R/W	Data register	0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA0															

Bits	Name	Type	Description	
15:0	DATA0	R/W	This field is used differently by the operation mode.	

Interval mode The target counting value is stored in this field. When the counter value equals to this register, a MAT_INT0 interrupt is generated. Whenever an MAT_INT0 occurs or clear operation is executed, this value is updated to the internal data buffer 0.

PWM/
One-shot
mode The target counting value is stored in this field. When the counter value equals to this register, a MAT_INT0 interrupt is generated. This field is updated to the internal data buffer 0 when MAT_INT0 occurs or when clear operation is executed.

Capture mode In rising-edge clear mode, the captured data at the falling edge is stored to this register.
In falling-edge clear mode, the captured data at the rising edge is stored to this register.

TBDDATA1, TCDATA1, TDDATA1

Name	Width	Address	R/W	Description	Reset
TBDDATA1	32	0x3C70_002C	R/W	Data register	0x0000 0000
TCDATA1	32	0x3C70_004C	R/W	Data register	0x0000 0000
TDDATA1	32	0x3C70_006C	R/W	Data register	0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

DATA1

Bits	Name	Type	Description
15:0	DATA1	R/W	<p>This field is used differently by the operation mode.</p> <p>Interval mode Not used.</p> <p>PWM/ One-shot mode The target counting value is stored in this field. When the counter value equals to this register, a MAT_INT1 interrupt is generated. This field is updated to the internal data buffer 1 when MAT_INT1 occurs or when clear operation is executed.</p> <p>Capture mode In rising-edge clear mode, the captured data at the rising edge is stored to this register. In falling-edge clear mode, the captured data at the falling edge is stored to this register.</p>

TBPRE, TCPRE, TDPRE

Name	Width	Address	R/W	Description	Reset
TBPRE	32	0x3C70_0030	R/W	Pre-scale register	0x0000 0000
TCPRE	32	0x3C70_0050	R/W	Pre-scale register	0x0000 0000
TDPRE	32	0x3C70_0070	R/W	Pre-scale register	0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits	Name	Type	Description
9:0	PRE	R/W	Pre-scale value

TBCNT, TCCNT, TDCNT

Name	Width	Address	R/W	Description	Reset
TBCNT	32	0x3C70_0034	R	Counter register	0x0000 0000
TCCNT	32	0x3C70_0054	R	Counter register	0x0000 0000
TDCNT	32	0x3C70_0074	R	Counter register	0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

CNT

Bits	Name	Type	Description
15:0	CNT	R	Counter register

INTERVAL MODE

This mode is selected by setting TM_MODE_SEL to 0b00x. When the (TM_CNT + 1) value equals to the buffer corresponding to TM_DATA0 in the operation, an interrupt (TM_MAT_INT0) occurs, the value of the TM_CNT is cleared to zero and TM_CNT counts up again. In this mode, TM_DATA1 is not used. The TM_OUT pin of a timer is used to generate a signal that is toggled by TM_MAT_INT0. The waveform of the signal generated in this mode is in Figure 11-2. As you can see, changing the value of TM_DATA0 varies the width of the pulse. TM_DATA0 can be updated during the operation but the effect of that value occurs at the next phase.

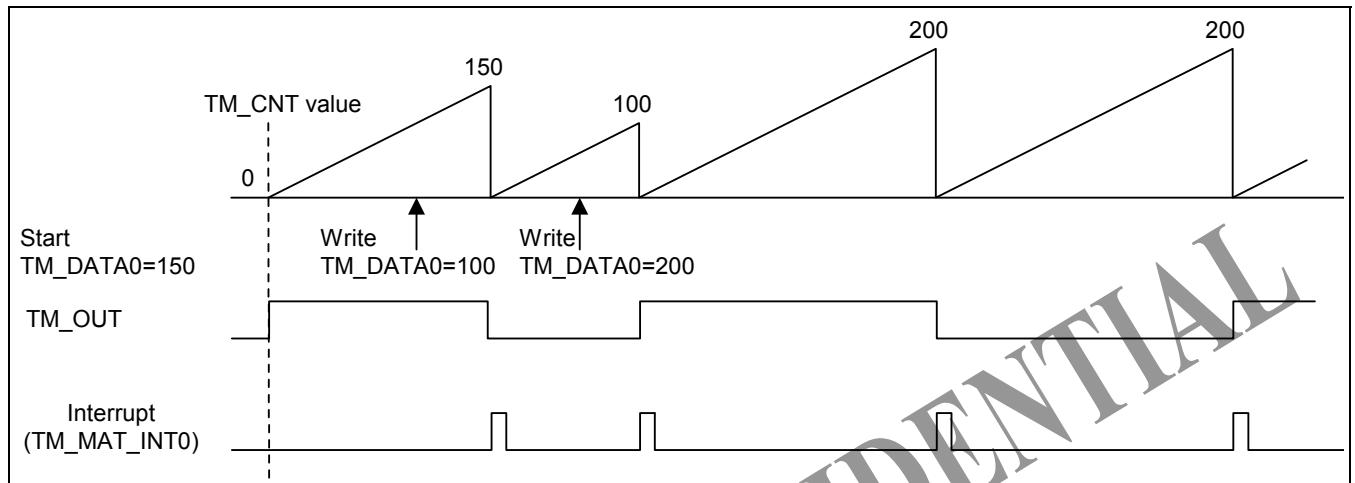


Figure 11-2. Interval Mode Operation

The detailed waveform is shown in Figure 11-3.

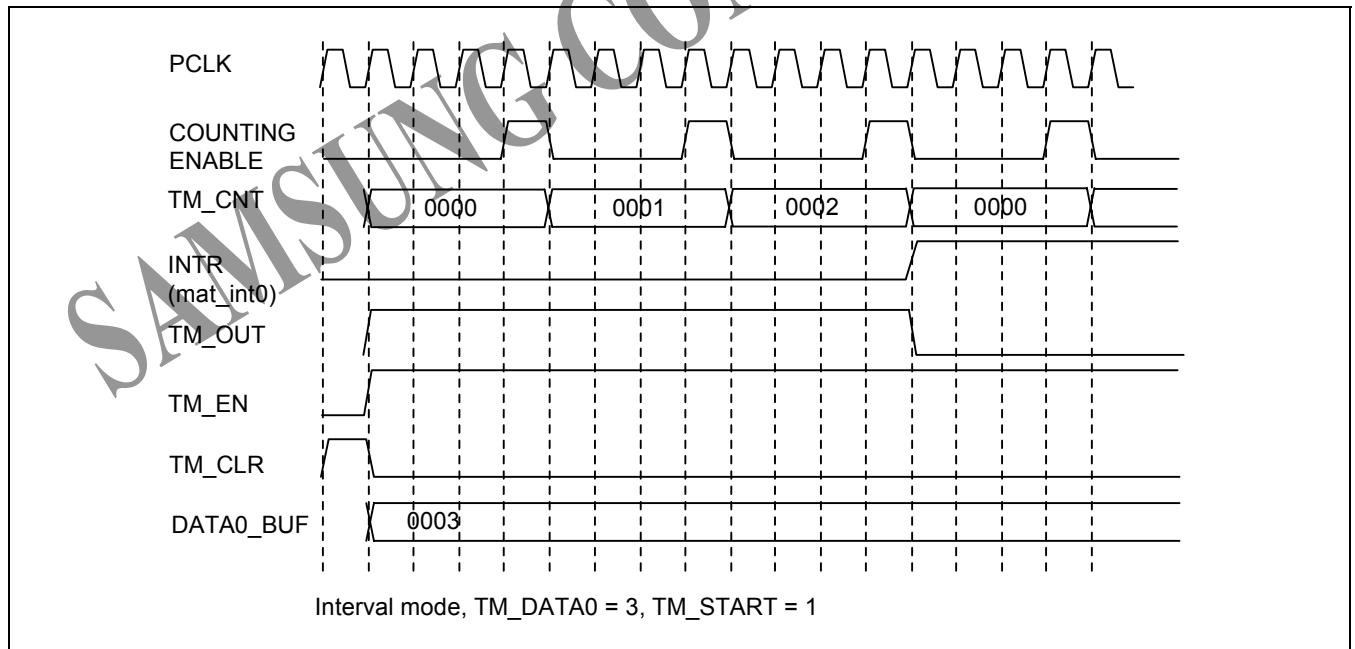


Figure 11-3. Waveform in Interval Mode

PWM (PULSE WIDTH MODULATION) MODE

Like the interval mode, the TM_CNT value is compared to the two buffers that are updated with the values of TM_DATA0 and TM_DATA1 register. When $(TM_CNT + 1)$ is equal to the buffer of TM_DATA0, TM_MAT_INT0 interrupt occurs and the timer continues the counting operation without clearing the counter. When $(TM_CNT + 1)$ is equal to the buffer of TM_DATA1, the timer generates TM_MAT_INT1 interrupt, clears the TM_CNT register, and updates the internal buffers with the values of TM_DATA0 register and TM_DATA1 register. For each interrupt, the TM_OUT is toggled. As the values of TM_DATA0 and TM_DATA1 register are updated after the TM_MAT_INT1 interrupt, the new values in TM_DATA0 and TM_DATA1 registers have an effect after TM_MAT_INT1 occurs.

This mode is used to generate a configurable PWM signal. The clock period of PWM signal can be set in TM_DATA1 register and the duty ratio can be set in TM_DATA0 register. The operation of this mode is described in the Figure 11-4.

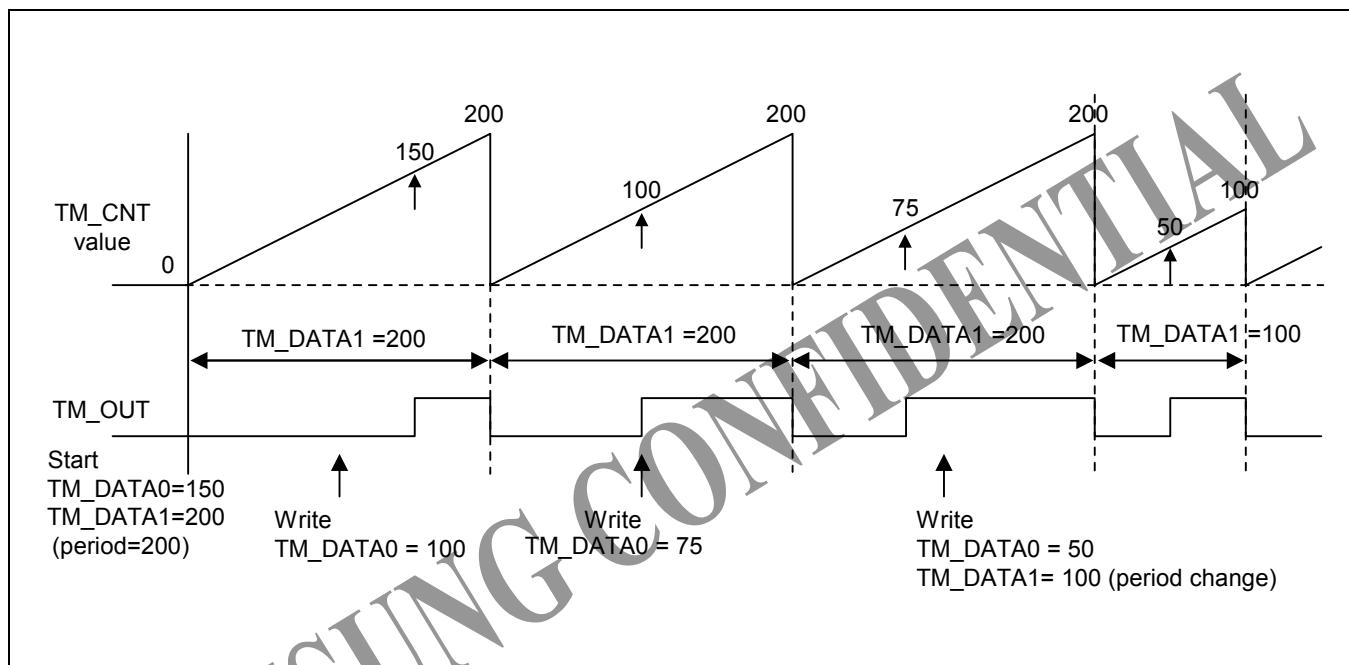


Figure 11-4. PWM Mode Operation

Depending on the values of the TM_DATA0 and TM_DATA1, the shapes of the PWM signals are different. The detailed waveforms in PWM mode are shown in Figure 11-5, Figure 11-6, Figure 11-7, and Figure 11-8.

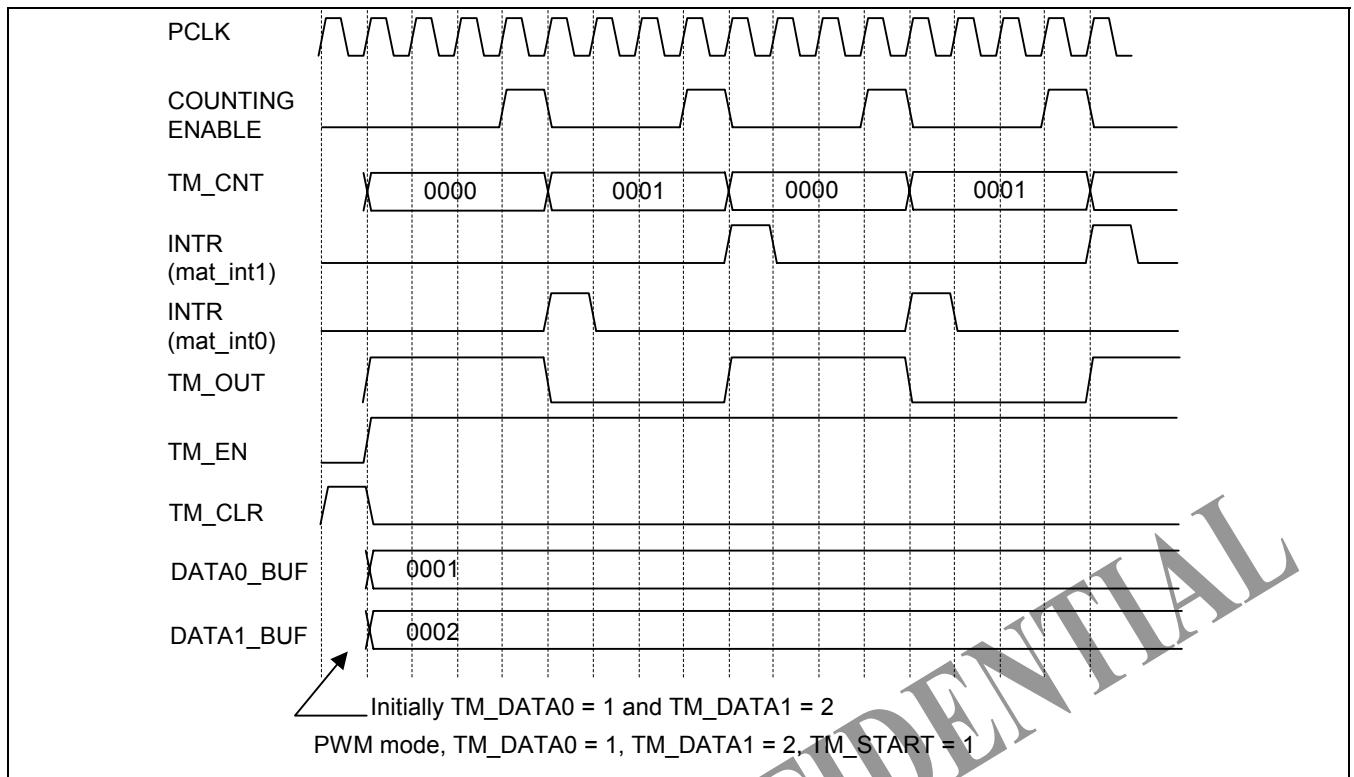


Figure 11-5. PWM Signal When TM_DATA0 = 1 and TM_DATA1 = 2

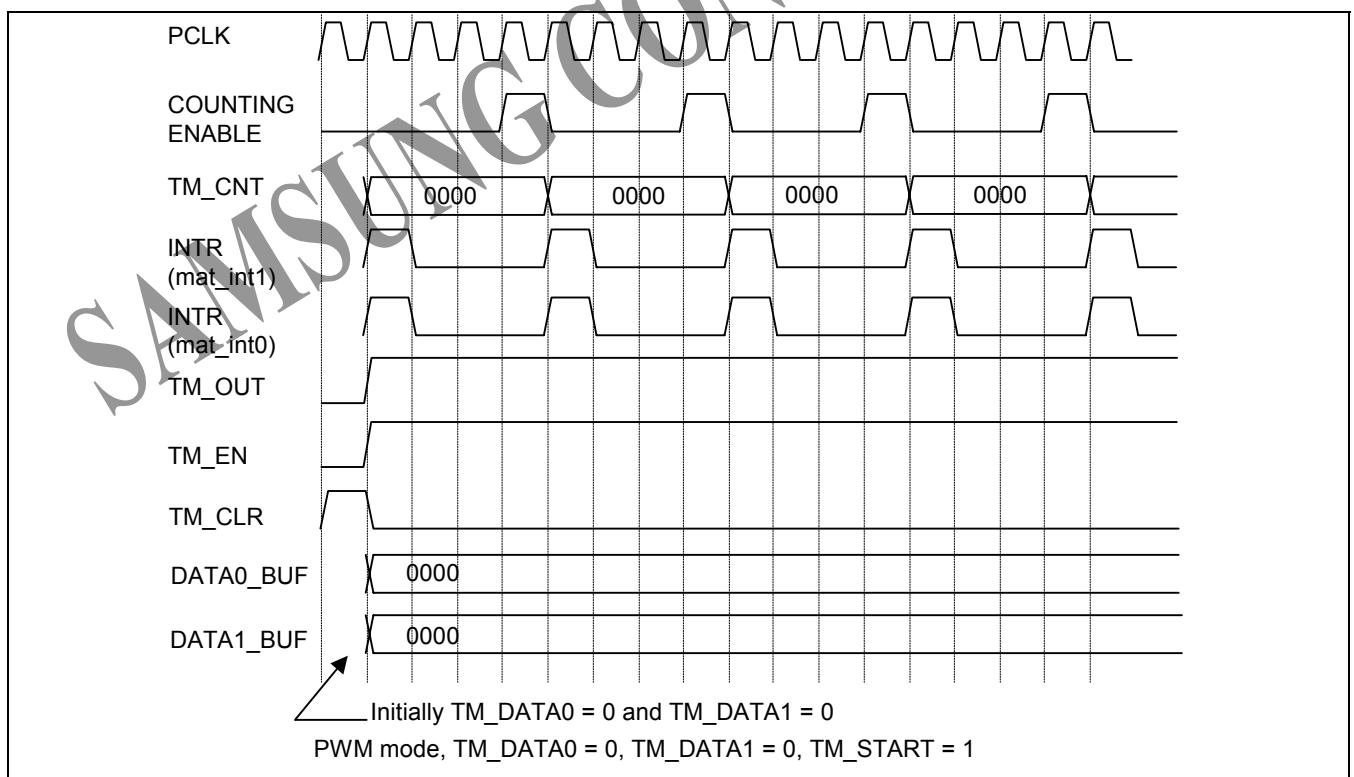


Figure 11-6. PWM Signal When TM_DATA0 = 0 and TM_DATA1 = 0

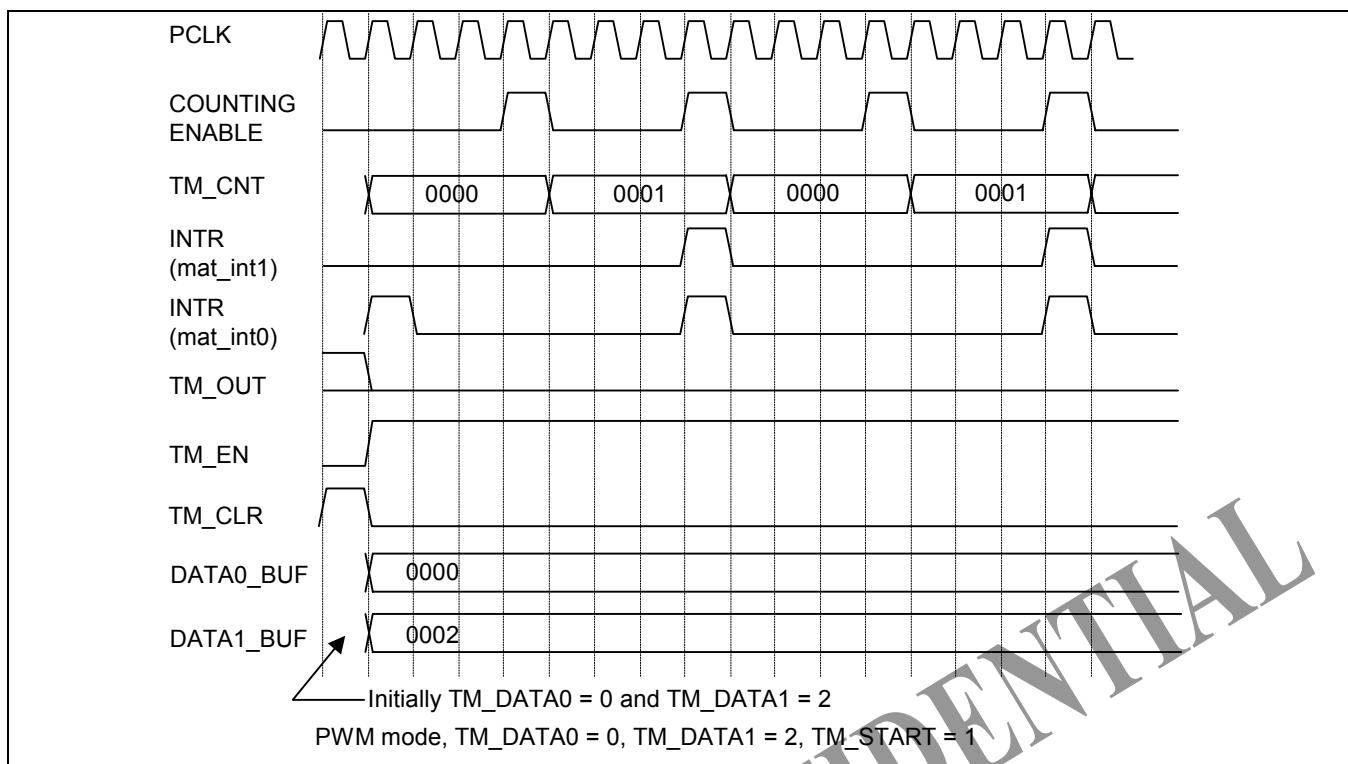


Figure 11-7. PWM Signal When TM_DATA0 = 0 and TM_DATA1 = 2 (Duty Ratio = 0 %)

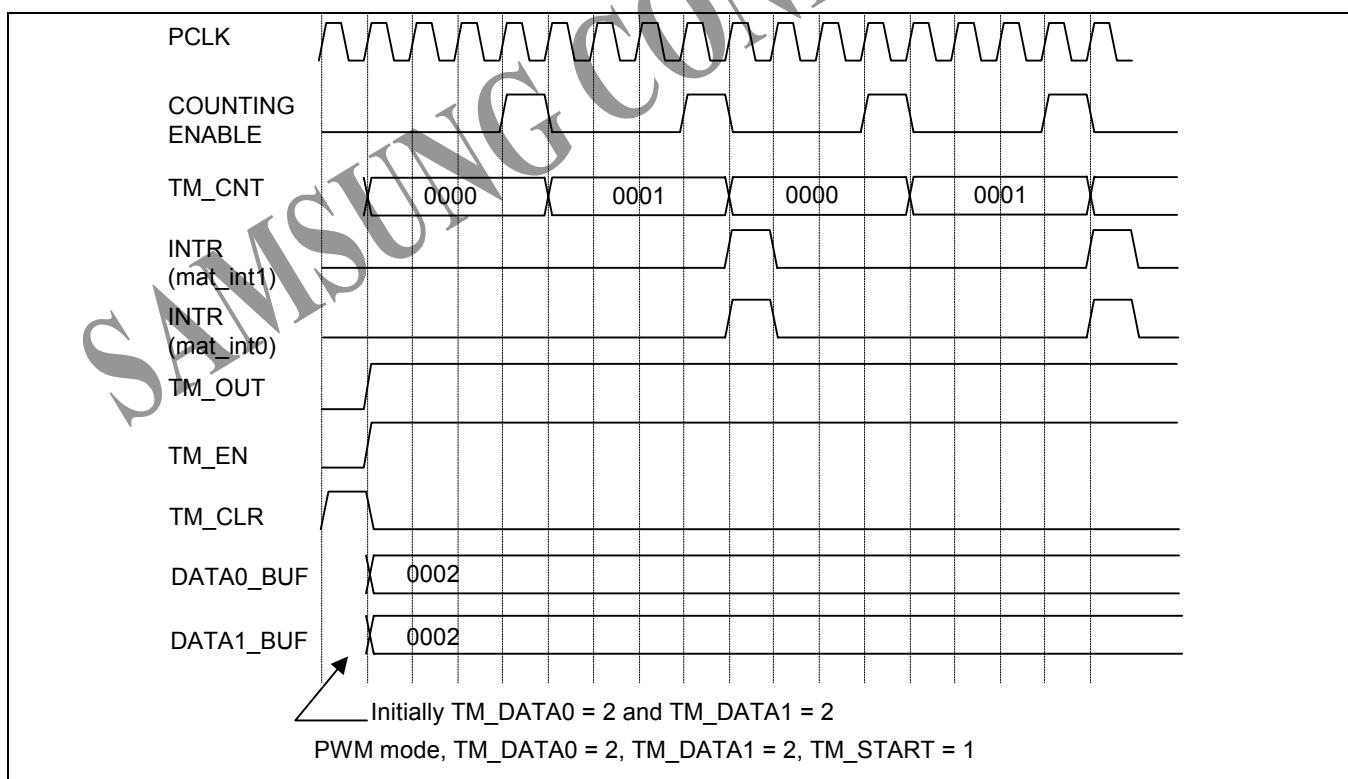


Figure 11-8. PWM Signal When TM_DATA0 = 2 and TM_DATA1 = 2 (Duty Ratio = 100 %)

ONE-SHOT MODE

One-shot mode is same as the PWM mode except that only one PWM signal pulse is generated. After generating one PWM signal, the flag, TM_EN, in TM_COM register is cleared to disable the timer. The operation of the one-shot mode is described in Figure 11-9.

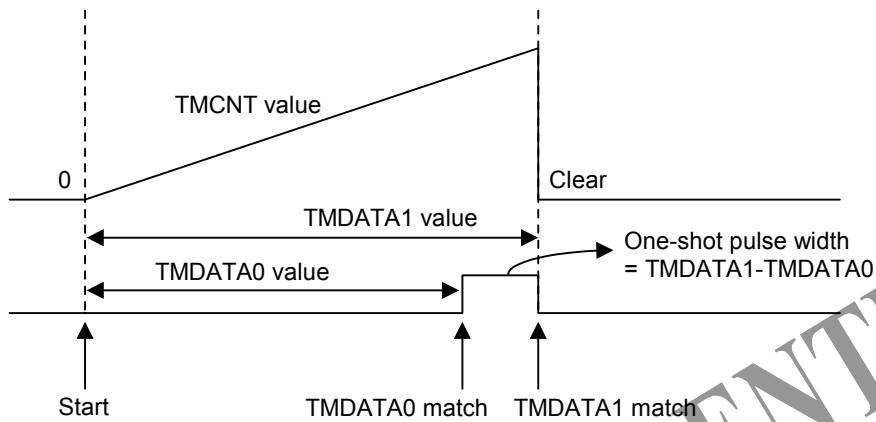


Figure 11-9. One-shot Mode Operation

CAPTURE MODE

Capture mode is used to capture the external signal from the TM_CAP. When a timer is enabled in this mode, the internal counter continues the counting and the timer waits an event on the TM_CAP port. When a falling or rising transition occurs on the TM_CAP port, the value of the counter register is captured to the data registers (TM_DATA0 or TM_DATA1) and an interrupt is generated (TM_MAT_INT0 or TM_MAT_INT1). Capture mode has two distinct modes: rising edge clear mode and falling edge clear mode. The TM_CAP_MODE flag in the TMCON register sets these modes.

In rising edge clear mode by setting TM_CAP_MODE to 0, when a falling edge is detected, the count value is captured to the TM_DATA0 and an interrupt TM_MAT_INT0 is generated. On the other hand, when a rising edge on the TM_CAP port is detected, the count value is also captured to the TM_DATA1, an interrupt TM_MAT_INT1 is generated and the count register is cleared to zero. In this mode, the internal counter is cleared when an rising event is detected on the TM_CAP port. The detailed description is shown in Figure 11-10.

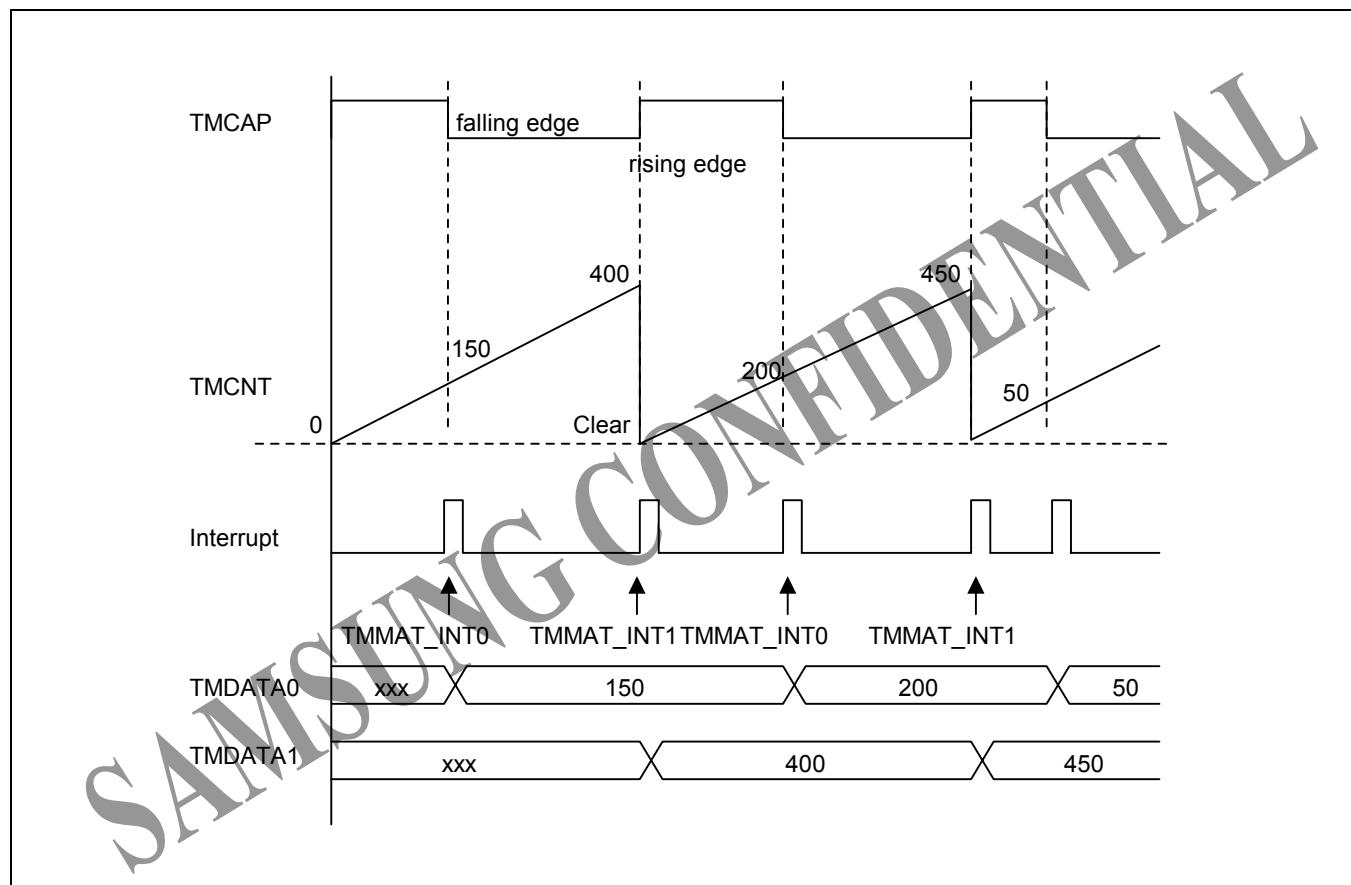


Figure 11-10. Capture Mode in Rising Edge Clear Mode

The falling edge clear mode is reverse to the rising edge clear mode. When a rising edge is detected on the TM_CAP port, the count value is captured to TM_DATA0 and an interrupt, TM_MAT_INT0, is generated. When a falling edge occurs, the count value is stored to TM_DATA1, TM_MAT_INT1 occurs and the count register is cleared.

By using the capture mode, you can get all the needed information of a PWM signal: duty ratio and the period.

COUNTING CLOCK DIVISION

The internal clock is counted by a clock which is prescaled by TM_PRE register and clock selection. The clock selected by TM_CS is predefined clocks or external clocks. The predefined clocks are derived from the main clock.

There are two external clocks. They can be used for accurate sync with an external logic that is not synchronous with the main clock. The internal counter counts up with the rising edge of the external clock.

The clock selected by the TM_CS is scaled by the TM_PRE register. The frequency of the scaled clock is as follows:

$$\text{Scaled clock [Hz]} = \text{Source clock [Hz]} / (\text{TM_PRE} + 1)$$

The source clock is the selected clock by the TM_CS. Table 11-1 shows the frequency and the period for each clock selection. It assumes that the main clock is 121.5 MHz.

Table 11-1. Pre-scaled Clock Frequency and Period When Main Clock = 121.5 MHz

TM_CS	Frequency/ Period	TM_PRE = 0		TM_PRE = 1024	
		Count Clock	Overflow (x65536)	Count Clock	Overflow (x65536)
		Frequency/ Period	Frequency/ Period	Frequency/ Period	Frequency/ Period
Main Clock / 64	1.90 MHz	1.90 MHz	28.97 Hz	1.85 KHz	0.03 Hz
	526.75 ns	526.75 ns	34.52 ms	539.39 ms	35.35 s
Main Clock / 16	7.59 MHz	7.59 MHz	115.87 Hz	7.42 KHz	0.11 Hz
	131.69 ns	131.69 ns	8.63 ms	134.85 ms	8.84 s
Main Clock / 4	30.38 MHz	30.38 MHz	463.49 Hz	29.66 KHz	0.45 Hz
	32.92 ns	32.92 ns	2.16 ms	33.71 ms	2.21 s
Main Clock / 2	60.75 MHz	60.75 MHz	926.97 Hz	59.33 KHz	0.91 Hz
	16.46 ns	16.46 ns	1.08 ms	16.86 ms	1.10 s

NOTES

SAMSUNG CONFIDENTIAL

12

NAND FLASH MEMORY CONTROLLER

OVERVIEW

This module can control the interface of the external NAND Flash memory

FEATURES

Flash Memory Controller (FMC) supports the following function

- NAND Flash Memory Interface by CPU / IODMA method
- Support APB interface
- On-the-fly Hamming Code over 256 bytes
- On-the-fly Parity, Syndrome Calculator about RS (472, 464, 9) over GF (2^9)
- Support 4-way interleaving for enhance write performance
- Special Function Register
 - ✓ Control Register 0
 - ✓ Control Register 1
 - ✓ NAND Flash Command Register
 - ✓ NAND Flash Address Register 0, 1, 2, 3, 4, 5, 6, 7
 - ✓ Address Counter Register
 - ✓ Data Counter Register
 - ✓ Write Data Register 0, 1, 2, 3
 - ✓ Status Register
 - ✓ Syndrome Register 0, 1, 2, 3, 4, 5, 6, 7 by Hamming-Code
 - ✓ Write / Read FIFO Register
 - ✓ Reed-Solomon Control Register
 - ✓ On-the-fly Parity Register 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 by RS-Code
 - ✓ On-the-fly Syndrome Register 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 by RS-Code
 - ✓ On-the-fly ECC Result Flag Register

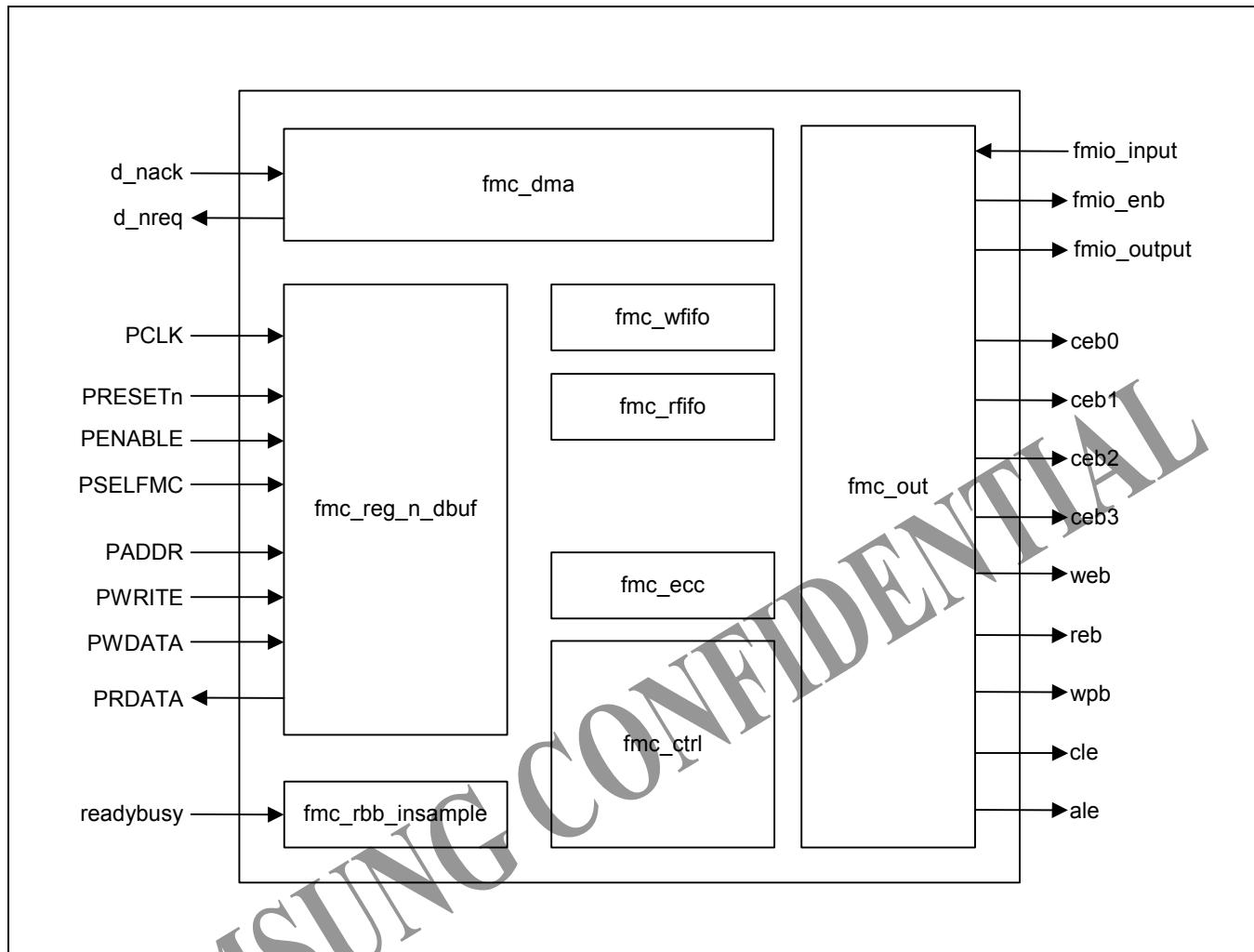
BLOCK DIAGRAM**Figure 12-1. NAND Flash Memory Block Diagram**

Table 12-1. Pin Description for Flash Memory Diagram

Part	Port Name	Description
APB	PCLK	AMBA APB protocol
	PRESETn	
	PSELFMC	
	PADDR[5:0]	
	PWRITE	
	PWDATA[15:0]	
	PRDATA[15:0]	
Flash	fmio_enb	Flash IO output enable, low active.
	fmio_input[15:0]	Flash IO input
	fmio_output[15:0]	Flash IO output
	ceb0	Chip0 select
	ceb1	Chip1 select
	ceb2	Chip2 select
	ceb3	Chip3 select
	web	Write enable
	reb	Read enable
	wpb	Wrote protect
	cle	Command latch enable
	ale	Address latch enable
DMA	D_nack	DMA acknowledge
	D_nreq	DMA request

Table 12-2. Flash Memory Controller Register

Register	Address	R/W	Description	Reset
FMCTRL0	0x3C20_0000	R/W	Control Register0	0x0000
FMCTRL1	0x3C20_0004	R/W	Control Register1	0x00
FMCMD	0x3C20_0008	R/W	Command Register	0x00
FMADDR0	0x3C20_000C	R/W	Address Register0	0x00
FMADDR1	0x3C20_0010	R/W	Address Register1	0x00
FMADDR2	0x3C20_0014	R/W	Address Register2	0x00
FMADDR3	0x3C20_0018	R/W	Address Register3	0x00
FMADDR4	0x3C20_001C	R/W	Address Register4	0x00
FMADDR5	0x3C20_0020	R/W	Address Register5	0x00
FMADDR6	0x3C20_0024	R/W	Address Register6	0x00
FMADDR7	0x3C20_0028	R/W	Address Register7	0x00
FMANUM	0x3C20_002C	R/W	Address Counter Register	0x00
FMDNUM	0x3C20_0030	R/W	Data Counter Register	0x00
FMDATAW0	0x3C20_0034	R/W	Write Data Register0	0x00
FMDATAW1	0x3C20_0038	R/W	Write Data Register1	0x00
FMDATAW2	0x3C20_003C	R/W	Write Data Register2	0x00
FMDATAW3	0x3C20_0040	R/W	Write Data Register3	0x00
FMCSTAT	0x3C20_0048	R	Status Register	0x00
FMSYND0	0x3C20_004C	R	Hamming Syndrome0	0x0000000
FMSYND1	0x3C20_0050	R	Hamming Syndrome1	0x0000000
FMSYND2	0x3C20_0054	R	Hamming Syndrome2	0x0000000
FMSYND3	0x3C20_0058	R	Hamming Syndrome3	0x0000000
FMSYND4	0x3C20_005C	R	Hamming Syndrome4	0x0000000
FMSYND5	0x3C20_0060	R	Hamming Syndrome5	0x0000000
FMSYND6	0x3C20_0064	R	Hamming Syndrome6	0x0000000
FMSYND7	0x3C20_0068	R	Hamming Syndrome7	0x0000000
FMFIFO	0x3C20_0080 ~ 0x3C20_00FC	R/W	WRITE/READ FIFO	0x00
RSCRTL	0x3C20_0100	W	Reed-Solomon Control Register	0x00
RSPArty0-0	0x3C20_0110	R/W	On-the-fly Parity Register0[31:0]	0x00
RSPArty0-1	0x3C20_0114	R/W	On-the-fly Parity Register0[63:32]	0x00
RSPArty0-2	0x3C20_0118	R/W	On-the-fly Parity Register0[71:64]	0x00
RSPArty1-0	0x3C20_0120	R/W	On-the-fly Parity Register1[31:0]	0x00
RSPArty1-1	0x3C20_0124	R/W	On-the-fly Parity Register1[63:32]	0x00
RSPArty1-2	0x3C20_0128	R/W	On-the-fly Parity Register1[71:64]	0x00
RSPArty2-0	0x3C20_0130	R/W	On-the-fly Parity Register2[31:0]	0x00
RSPArty2-1	0x3C20_0134	R/W	On-the-fly Parity Register2[63:32]	0x00

RSPaity2-2	0x3C20_0138	R/W	On-the-fly Parity Register2[71:64]	0x00
RSPaity3-0	0x3C20_0140	R/W	On-the-fly Parity Register3[31:0]	0x00
RSPaity3-1	0x3C20_0144	R/W	On-the-fly Parity Register3[63:32]	0x00
RSPaity3-2	0x3C20_0148	R/W	On-the-fly Parity Register3[71:64]	0x00
RSSynd0-0	0x3C20_0150	R/W	On-the-fly Synd Register0[31:0]	0x00
RSSynd0-1	0x3C20_0154	R/W	On-the-fly Synd Register0[63:32]	0x00
RSSynd0-2	0x3C20_0158	R/W	On-the-fly Synd Register0[71:64]	0x00
RSSynd1-0	0x3C20_0160	R/W	On-the-fly Synd Register1[31:0]	0x00
RSSynd1-1	0x3C20_0164	R/W	On-the-fly Synd Register1[63:32]	0x00
RSSynd1-2	0x3C20_0168	R/W	On-the-fly Synd Register1[71:64]	0x00
RSSynd2-0	0x3C20_0170	R/W	On-the-fly Synd Register2[31:0]	0x00
RSSynd2-1	0x3C20_0174	R/W	On-the-fly Synd Register2[63:32]	0x00
RSSynd2-2	0x3C20_0178	R/W	On-the-fly Synd Register2[71:64]	0x00
RSSynd3-0	0x3C20_0180	R/W	On-the-fly Synd Register3[31:0]	0x00
RSSynd3-1	0x3C20_0184	R/W	On-the-fly Synd Register3[63:32]	0x00
RSSynd3-2	0x3C20_0188	R/W	On-the-fly Synd Register3[71:64]	0x00
FlagSynd	0x3C20_0190	R/W	On-the-fly ECC Result Flag	0x00

- Special Function Register



ELECTRONICS

1. Control Register 0 (FMCTRL0, BASE + 0x000, R/W)

Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Default
DataWidth	WEB/REB Pulse Width			WPB	WEB/REB High Hold Time			8'h00
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Default
ModeSEL	-	ECCOn	CEB3	CEB2	CEB1	CEB0	FMCOOn	8'h00

- ✓ DataWidth : 1'b0 -> X8 Flash
: 1'b1 -> X16 Flash
- ✓ WEB/REB Pulse Width : tWP
- ✓ WPB : Flash Write Protect
- ✓ WEB/REB High Hold Time : tWH
- ✓ ModeSEL : 1'b0 -> CPU Mode
: 1'b1 -> IODMA Mode
- ✓ ECCOn : Hamming Code, On-the-fly Parity/Syndrome Enable
- ✓ CEB0, 1, 2, 3 : Flash Chip Enable
- ✓ FMCOOn : NAND Flash Memory Controller Enable

2. Control Register 1 (FMCTRL1, BASE + 0x004, R/W)

Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Default
-	-	-	-	-	-	ClearSyndPtr	ClearParityPtr	8'h00
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Default
ClearRFIFO	ClearWFIFO	ClearSyndPtr	WriteREQSEL	-	DoWriteData	DoReadData	DoTransAddr	8'h00

- ✓ ClearSyndPtr : Clear on-the-fly Syndrome Register Pointer
- ✓ ClearParityPtr : Clear on-the-fly Parity Register Pointer
- ✓ ClearRFIFO : Flush Read FIFO
- ✓ ClearWFIFO : Flush Write FIFO
- ✓ CleaeSyndPtr : Clear Hamming Syndrome Register Pointer
- ✓ WriteREQSEL : 1'b0 -> Request when WFIFO Empty
: 1'b1 -> when WFIFO Half Empty
- ✓ DoWriteData : Write Data to Flash (Page Program)
- ✓ DoReadData : Read Data from Flash (Page Read)
- ✓ DoTransAddr : Transfer Address to Flash

3. NAND Flash Command Register (FMCMD, BASE + 0x008, R/W)

Bit7 ~ Bit0	Default
FMCMD	8'h00

- ✓ Command to be transferred to Flash
- ✓ Refer to Command sets in NAND Spec

4. NAND Flash Address Register 0 (FMCADDR0, BASE + 0x00C, R/W)

Bit7 ~ Bit0	Default
FMCADDR	8'h00

- ✓ Address to be transferred to Flash
- ✓ Depend on the density of Flash Memory

5. NAND Flash Address Register 1 (FMCADDR1, BASE + 0x010, R/W)

6. NAND Flash Address Register 2 (FMCADDR2, BASE + 0x014, R/W)

7. NAND Flash Address Register 3 (FMCADDR3, BASE + 0x018, R/W)

8. NAND Flash Address Register 4 (FMCADDR4, BASE + 0x01C, R/W)

9. NAND Flash Address Register 5 (FMCADDR5, BASE + 0x020, R/W)

10. NAND Flash Address Register 6 (FMCADDR6, BASE + 0x024, R/W)

11. NAND Flash Address Register 7 (FMCADDR7, BASE + 0x028, R/W)

12. Address Counter Register (FMCANUM, BASE + 0x02C, R/W)

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2 ~ Bit0	Default
RESERVED					FMCANUM	8'h00

- ✓ The number of the transferred address
- ✓ Ex) FMCANUM == 0 : FMCADDR0, FMCANUM == 1 : FMCADDR0, 1 ...

13. Data Counter Register (FMCDNUM, BASE + 0x030, R/W)

Bit11 ~ Bit0	Default
FMCDNUM	12'h000

- ✓ The number of the transferred data

14. Status Register (FCMSTAT, BASE + 0x048, R/W)

Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Default
RESERVED					EndECC	RFIFO_FULL	WFIFO_EMPTY	8'h00
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Default
-	RFIFO_HFUL L	WFIFO_HEM PTY	TransDone	AddrDone	CMDDone	RBBDone	RBB	8'h00

- ✓ EndECC : High means that Parity/Syndrome Operation end
- ✓ RFIFO_FULL : High means that Read FIFO is Full
- ✓ WRFIFO_EMPTY : High means that Write FIFO is Empty
- ✓ RFIFO_HFULL : High means that Read FIFO is Half-Full
- ✓ WFIFO_HEMPTY : High means that Write FIFO is Half-Empty

- ✓ TransDone : High means that the data transfer done
: This bit is cleared when CPU write this bit.
- ✓ AddrDone : High means that the address transfer done
: This bit is cleared when CPU write this bit.
- ✓ CMDDone : High means that the command transfer done
- ✓ RBBDone : High means that the RBB is high
: This bit is cleared when CPU write this bit.
- ✓ RBB : Flash Ready/Busy output

15. Syndrome Register 0 by Hamming-Code (HammingSYND0, BASE + 0x04C, R)

Bit23 ~ Bit0	Default
HammingSYND0	24'h000000

- ✓ Syndrome by Hamming-Code over 256 bits
- ✓ When page program, Firmware read this value and write Flash Spare Area. When page read, Firmware compare this value to Flash Spare Area's Syndrome which written in previous page program.

16. Syndrome Register 1 by Hamming-Code (HammingSYND1, BASE + 0x050, R)

17. Syndrome Register 2 by Hamming-Code (HammingSYND2, BASE + 0x054, R)

18. Syndrome Register 3 by Hamming-Code (HammingSYND3, BASE + 0x058, R)

19. Syndrome Register 4 by Hamming-Code (HammingSYND4, BASE + 0x05C, R)

20. Syndrome Register 5 by Hamming-Code (HammingSYND5, BASE + 0x060, R)

21. Syndrome Register 6 by Hamming-Code (HammingSYND6, BASE + 0x064, R)

22. Syndrome Register 7 by Hamming-Code (HammingSYND7, BASE + 0x068, R)

23. Write / Read FIFO Register (FMCFIFO, BASE + 0x080~0x0FC, R/W)

Bit31 ~ Bit0	Default
FMCFIFO	32'h00000000

- ✓ Register used to store the transmitted/received data

24. Reed-Solomon Control Register (RS_ECC_CTRL, BASE + 0x100, W)

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Default
RESERVED						SyndromeOn	ParityOn	8'h00

- ✓ SyndromeOn : On-the-fly Syndrome Enable
- ✓ ParityOn : On-the-fly Parity Enable

25. On-the-fly Parity Register 0 (PARITY0_31_0, BASE + 0x110, R/W)

Bit31 ~ Bit0	Default
PARITY0_31_0	32'h00000000

- ✓ On-the-fly Parity Register
- ✓ In page program case, Firmware read this register and write to Flash Spare Area after write operation. In page read case, Firmware read the value from Flash Spare Area and write to this register before read operation.

26. On-the-fly Parity Register 1 (PARITY0_63_32, BASE + 0x114, R/W)

Bit31 ~ Bit0	Default
PARITY0_63_32	32'h00000000 0

27. On-the-fly Parity Register 2 (PARITY0_71_64, BASE + 0x118, R/W)

Bit7 ~ Bit0	Default
PARITY0_71_64	8'h00

28. On-the-fly Parity Register 3 (PARITY1_31_0, BASE + 0x120, R/W)

29. On-the-fly Parity Register 4 (PARITY1_63_32, BASE + 0x124, R/W)

30. On-the-fly Parity Register 5 (PARITY1_71_64, BASE + 0x128, R/W)

31. On-the-fly Parity Register 6 (PARITY2_31_0, BASE + 0x130, R/W)

32. On-the-fly Parity Register 7 (PARITY2_63_32, BASE + 0x134, R/W)

33. On-the-fly Parity Register 8 (PARITY2_71_64, BASE + 0x138, R/W)

34. On-the-fly Parity Register 9 (PARITY3_31_0, BASE + 0x140, R/W)

35. On-the-fly Parity Register 10 (PARITY3_63_32, BASE + 0x144, R/W)

36. On-the-fly Parity Register 11 (PARITY3_71_64, BASE + 0x148, R/W)

37. On-the-fly Syndrome Register 0 (SYNDROME0_31_0, BASE +0x150, R)

Bit31 ~ Bit0	Default
SYNDROME0_31_0	32'h00000000 0

✓ On-the-fly Syndrome Register

✓ If Syndrome is not zero, Firmware write this value to Flash ECC Engine's Syndrome Poly register. After that, invoke Flash ECC Engine without calculating syndrome.

38. On-the-fly Syndrome Register 1 (SYNDROME0_63_32, BASE +0x154, R)

Bit31 ~ Bit0	Default
SYNDROME0_63_32	32'h00000000 0

39. On-the-fly Syndrome Register 2 (SYNDROME0_71_64, BASE +0x158, R)

Bit7 ~ Bit0	Default
SYNDROME0_71_64	8'h00

40. On-the-fly Syndrome Register 3 (SYNDROME1_31_0, BASE +0x160, R)

41. On-the-fly Syndrome Register 4 (SYNDROME1_63_32, BASE +0x164, R)

42. On-the-fly Syndrome Register 5 (SYNDROME1_71_64, BASE +0x168, R)

43. On-the-fly Syndrome Register 6 (SYNDROME2_31_0, BASE +0x170, R)

44. On-the-fly Syndrome Register 7 (SYNDROME2_63_32, BASE +0x174, R)

45. On-the-fly Syndrome Register 8 (SYNDROME2_71_64, BASE +0x178, R)

46. On-the-fly Syndrome Register 9 (SYNDROME3_31_0, BASE +0x180, R)

47. On-the-fly Syndrome Register 10 (SYNDROME3_63_32, BASE +0x184, R)

48. On-the-fly Syndrome Register 11 (SYNDROME3_71_64, BASE +0x188, R)

49. On-the-fly ECC Result Flag Register (FLAGSYND, BASE + 0x190, R)

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Default
RESERVED				FLAGSYND				8'h00

- ✓ Syndrome Result Register
- ✓ If not zero, invoke Flash ECC Engine.

- Reset Sequence
 - 1. WRITE FMCTRL0
 - 2. WRITE FMCMD [:=00FFh]
 - 3. WAIT until FMCSTAT[1]==1
 - 4. WRITE FMCSTAT [:=0002h]

SAMSUNG CONFIDENTIAL

- READ ID Sequence
 - 1. WRITE FMCMD [:=0090h]
 - 2. WRITE FMCANUM [:=0000h]
 - 3. WRITE FMCADDR0 [:=0000h]
 - 4. WRITE FMCTRL1 [:=0001h]
 - 5. WAIT until FMCSTAT[3]==1
 - 6. WRITE FMCSTAT [:=0008h]
 - 7. WRITE FMCTRL0
 - 8. WRITE FMCDNUM [:=0003h]
 - 9. WRITE FMCTRL1 [:=0002h]
 - 10. WAIT until FMCSTAT[4]==1
 - 11. WRITE FMCSTAT [:=0010h]
 - 12. READ FMCFIFO0, ... (depend on the target flash)
 - 13. WRITE FMCTRL1 [:=0080h]

SAMSUNG CONFIDENTIAL

- READ Status Sequence
 - 1. WRITE FMCTRL1 [:=0080]
 - 2. WRITE FMCMD [:=0070h]
 - 3. WRITE FMCDNUM [:=0000h]
 - 4. WRITE FMCTRL1 [:=0002h]
 - 5. WAIT until FMCSTAT[4]==1
 - 6. WRITE FMCSTAT [:=0010h]
 - 7. READ FMCFIFO and CHECK (32'h00E0_0000 @ X16, 32'hE000_0000 @ X8)
 - 8. WRITE FMCTRL1 [:=0080h]

SAMSUNG CONFIDENTIAL

- Block Erase Sequence
 1. WRITE FMCTRL0
 2. WRITE FMCMD [:=0060h]
 3. WRITE FMCNUM (depend on the target flash)
 4. WRITE FMCADDR0, FMCADDR1, ... (depend on the target flash)
 5. WRITE FMCTRL1 [:=0001h]
 6. WAIT until FMCSTAT[3]==1
 7. WRITE FMCSTAT [:=0008h]
 8. WRITE FMCMD [:=00D0h]
 9. WAIT until FMCSTAT[1]==1
 10. WRITE FMCSTAT [:=0002h]
 11. CALL READ Status Routine

SAMSUNG CONFIDENTIAL

- Page Program @ CPU mode Sequence
 - 1. WRITE FMCTRL0
 - 2. WRITE FMCMD [:=0080h]
 - 3. WRITE FMCANUM (depend on the target flash)
 - 4. WRITE FMCADDR0, FMCADDR1, ... (depend on the target flash)
 - 5. WRITE FMCTRL1 [:=0001h]
 - 6. WAIT until FMCSTAT[3]==1
 - 7. WRITE FMCSTAT [:=0008h]
 - 8. WRITE FMCDNUM (depend on the target flash)
 - 9. WRITE FMCTRL1 [:=0004h]
 - 10. While (the number of write data) {
 - WRITE FMCFIFO [:=data]
 - WAIT until FMCSTAT[5]==1}
 - 11. WAIT until FMCSTAT[4]==1
 - 12. WRITE FMCSTAT [:=0010h]
 - 13. WRITE FMCMD [:=0010h]
 - 14. WAIT until FMCSTAT[1]==1
 - 15. WRITE FMCSTAT [:=0002h]
 - 16. CALL READ Status Routine
 - 17. WRITE FMCTRL1 [:=0040h]



- Page Program [with on-the-fly parity] @ CPU mode Sequence
 1. WRITE FMCTRL0
 2. WRITE RS_ECC_CTRL [:=0001h]
 3. WRITE FMCMD [:=0080h]
 4. WRITE FMCANUM (depend on the target flash)
 5. WRITE FMCADDR0, FMCADDR1, ... (depend on the target flash)
 6. WRITE FMCTRL1 [:=0001h]
 7. WAIT until FMCSTAT[3]==1
 8. WRITE FMCSTAT [:=0008h]
 9. WRITE FMCDNUM (depend on the target flash)
 10. WRITE FMCTRL1 [:=0004h]
 11. While (the number of write data) {
 WRITE FMCFIFO [:=data]
 WAIT until FMCSTAT[5]==1
}
 12. WAIT until FMCSTAT[4]==1
 13. WRITE FMCSTAT [:=0010h]
 14. WRITE RS_ECC_CTRL [:=0000h]
 15. WRITE FMCMD [:=0010h]
 16. WAIT until FMCSTAT[1]==1
 17. WRITE FMCSTAT [:=0002h]
 18. CALL READ Status Routine
 19. READ PARITY0_31_0, PARITY0_63_32, PARITY0_71_64
 20. READ PARITY1_31_0, PARITY1_63_32, PARITY1_71_64
 21. READ PARITY2_31_0, PARITY2_63_32, PARITY2_71_64
 22. READ PARITY3_31_0, PARITY3_63_32, PARITY3_71_64
 23. WRITE PARITY0, 1, 2, 3 To Flash Spare Area Pointer
 24. WRITE FMCTRL1 [:=0040h]

- Page READ @ CPU mode Sequence
 1. WRITE FMCTRL0
 2. WRITE FMCMD [:=0000h]
 3. WRITE FMCANUM (depend on the target flash)
 4. WRITE FMCADDR0, FMCADDR1, ... (depend on the target flash)
 5. WRITE FMCTRL1 [:=0001h]
 6. WAIT until FMCSTAT[3]==1
 7. WRITE FMCSTAT [:=0008h]
 8. WRITE FMCMD [:0030h]
 9. WRITE FMCDNUM (depend on the target flash)
 10. WAIT until FMCSTAT[1]==1
 11. WRITE FMCSTAT [:=0002h]
 12. WRITE FMCTRL1 [:=0002h]
 13. While (the number of read data) {
 READ FMCFIFO
 WAIT until (FMCSTAT[6]==1 or FMCSTAT[9]==1)
}
 14. WAIT until FMCSTAT[4]==1
 15. WRITE FMCSTAT [:=0010h]
 16. WRITE FMCTRL1 [:=00C0h]

- Page READ [with on-the-fly syndrome] @ CPU mode Sequence
 1. READ PARITY0, 1, 2, 3 From Flash Spare Area Pointer
 2. WRITE PARITY0_31_0, PARITY0_63_32, PARITY0_71_64
 3. WRITE PARITY1_31_0, PARITY1_63_32, PARITY1_71_64
 4. WRITE PARITY2_31_0, PARITY2_63_32, PARITY2_71_64
 5. WRITE PARITY3_31_0, PARITY3_63_32, PARITY3_71_64
 6. WRITE FMCTRL0
 7. WRITE RS_ECC_CTRL [:=0002h]
 8. WRITE FMCMD [:=0000h]
 9. WRITE FMCANUM (depend on the target flash)
 10. WRITE FMCADDR0, FMCADDR1, ... (depend on the target flash)
 11. WRITE FMCTRL1 [:=0001h]
 12. WAIT until FMCSTAT[3]==1
 13. WRITE FMCSTAT [:=0008h]
 14. WRITE FMCMD [:0030h]
 15. WRITE FMCDNUM (depend on the target flash)
 16. WAIT until FMCSTAT[1]==1
 17. WRITE FMCSTAT [:=0002h]
 18. WRITE FMCTRL1 [:=0002h]
 19. While (the number of read data) {
 READ FMCFIFO
 WAIT until (FMCSTAT[6]==1 or FMCSTAT[9]==1)
}
 20. WAIT until FMCSTAT[4]==1
 21. WRITE FMCSTAT [:=0010h]
 22. WRITE RS_ECC_CTRL [:=0000h]
 23. WRITE FMCTRL1 [:=00C0h]
 24. READ SYNDROME0_31_0, SYNDROME0_63_32, SYNDROME0_71_64
 25. READ SYNDROME1_31_0, SYNDROME1_63_32, SYNDROME1_71_64
 26. READ SYNDROME2_31_0, SYNDROME2_63_32, SYNDROME2_71_64
 27. READ SYNDROME3_31_0, SYNDROME3_63_32, SYNDROME3_71_64
 28. READ FLAGSYND
 29. WRITE SYNDROME0, 1, 2, 3 To Memory Spare Area Pointer

- Page Program @ IODMA mode Sequence
 1. WRITE FMCTRL0
 2. WRITE FMCMD [:=0080h]
 3. WRITE FMCANUM (depend on the target flash)
 4. WRITE FMCADDR0, FMCADDR1, ... (depend on the target flash)
 5. WRITE FMCTRL1 [:=0001h]
 6. WAIT until FMCSTAT[3]==1
 7. WRITE FMCSTAT [:=0008h]
 8. WRITE DMABASE1
 9. WRITE DMACON1
 10. WRITE DMATCNT1 (according to DMACON1's BL)
 11. WRITE DMACOM1
 12. WRITE FMCDNUM (depend on the target flash)
 13. WRITE FMCTRL1 [:=0014]
 14. WAIT INTERRUPT (IODMA)
 15. WAIT until FMCSTAT[4]==1
 16. WRITE FMCSTAT [:=0010h]
 17. IRQ HANDLER (IODMA)
 18. CLEAR INTERRUPT PENDING (IODMA)
 19. WRITE FMCMD [:=0010h]
 20. WAIT until FMCSTAT[1]==1
 21. WRITE FMCTRL1 [:=0002h]
 22. CALL READ Status Routine
 23. WRITE FMCTRL1 [:=0040h]

- Page Program [with on-the-fly parity] @ IODMA mode Sequence
 1. WRITE FMCTRL0
 2. WRITE RS_ECC_CTRL [:=0001h]
 3. WRITE FMCMD [:=0080h]
 4. WRITE FMCANUM (depend on the target flash)
 5. WRITE FMCADDR0, FMCADDR1, ... (depend on the target flash)
 6. WRITE FMCTRL1 [:=0001h]
 7. WAIT until FMCSTAT[3]==1
 8. WRITE FMCSTAT [:=0008h]
 9. WRITE DMABASE1
 10. WRITE DMACON1
 11. WRITE DMATCNT1 (according to DMACON1's BL)
 12. WRITE DMACOM1
 13. WRITE FMCDNUM (depend on the target flash)
 14. WRITE FMCTRL1 [:=0014]
 15. WAIT INTERRUPT (IODMA)
 16. WAIT until FMCSTAT[4]==1
 17. WRITE FMCSTAT [:=0010h]
 18. IRQ HANDLER (IODMA)
 19. CLEAR INTERRUPT PENDING (IODMA)
 20. WRITE RS_ECC_CTRL [:=0000h]
 21. READ PARITY0_31_0, PARITY0_63_32, PARITY0_71_64
 22. READ PARITY1_31_0, PARITY1_63_32, PARITY1_71_64
 23. READ PARITY2_31_0, PARITY2_63_32, PARITY2_71_64
 24. READ PARITY3_31_0, PARITY3_63_32, PARITY3_71_64
 25. WRITE FMCTRL0
 26. WRITE FMCMD [:=0085h]
 27. WRITE FMCANUM [:=0001h]
 28. WRITE FMCADDR0, FMCADDR1
 29. WRITE FMCTRL1 [:=0001h]
 30. WAIT until FMCSTAT[3]==1
 31. WRITE FMCSTAT [:=0008h]
 32. WRITE FMDNUM [:=001Fh] (depend on the number of spare data)
 33. WRITE FMCTRL1 [:=0004h]
 34. While (the number of spare data) {
 WRITE FMCFIFO [:=spare data]
 WAIT until FMCSTAT[5]==1
}
 35. WAIT until FMCSTAT[4]==1
 36. WRITE FMCSTAT [:=0010h]
 37. WRITE FMCMD [:=0010h]
 38. WAIT until FMCSTAT[1]==1
 39. WRITE FMCSTAT [:=0002h]
 40. CALL READ Status Routine
 41. WRITE FMCTRL1 [:=0040h]

- Page READ @ IODMA mode Sequence
 1. WRITE DMABASE1
 2. WRITE DMACON1
 3. WRITE DMATCNT1 (according to DMACON1's BL)
 4. WRITE DMACOM1
 5. WRITE FMCTRL0
 6. WRITE FMCMD [:=0000h]
 7. WRITE FMCANUM (depend on the target flash)
 8. WRITE FMCADDR0, FMCADDR1, ... (depend on the target flash)
 9. WRITE FMCTRL1 [:=0001h]
 10. WAIT until FMCSTAT[3]==1
 11. WRITE FMCSTAT [:=0008h]
 12. WRITE FMCDNUM (depend on the target flash)
 13. WRITE FMCMD [:=0030h]
 14. WAIT until FMCSTAT[1]==1
 15. WRITE FMCSTAT [:=0002h]
 16. WRITE FMCTRL1 [:=0002h]
 17. WAIT INTERRUPT (IODMA)
 18. IRQ HANDLER (IODMA)
 19. CLEAR INTERRUPT PENDING (IODMA)
 20. WAIT until FMCSTAT[4]==1
 21. WRITE FMCSTAT [:=0001h]
 22. WRITE FMCTRL1 [:=00C0h]



- Page READ [with on-the-fly syndrome] @ IODMA mode Sequence
 1. READ PARITY0, 1, 2, 3 From Flash Spare Area Pointer
 2. WRITE PARITY0_31_0, PARITY0_63_32, PARITY0_71_64
 3. WRITE PARITY1_31_0, PARITY1_63_32, PARITY1_71_64
 4. WRITE PARITY2_31_0, PARITY2_63_32, PARITY2_71_64
 5. WRITE PARITY3_31_0, PARITY3_63_32, PARITY3_71_64
 6. WRITE DMABASE1
 7. WRITE RS_ECC_CTRL [:=0002h]
 8. WRITE DMACON1
 9. WRITE DMATCNT1 (according to DMACON1's BL)
 10. WRITE DMACOM1
 11. WRITE FMCTRL0
 12. WRITE FMCMD [:=0000h]
 13. WRITE FMCANUM (depend on the target flash)
 14. WRITE FMCADDR0, FMCADDR1, ... (depend on the target flash)
 15. WRITE FMCTRL1 [:=0001h]
 16. WAIT until FMCSTAT[3]==1
 17. WRITE FMCSTAT [:=0008h]
 18. WRITE FMCDNUM (depend on the target flash)
 19. WRITE FMCMD [:=0030h]
 20. WAIT until FMCSTAT[1]==1
 21. WRITE FMCSTAT [:=0002h]
 22. WRITE FMCTRL1 [:=0002h]
 23. WAIT INTERRUPT (IODMA)
 24. IRQ HANDLER (IODMA)
 25. CLEAR INTERRUPT PENDING (IODMA)
 26. WAIT until FMCSTAT[4]==1
 27. WRITE FMCSTAT [:=0001h]
 28. WRITE RS_ECC_CTRL [:=0000h]
 29. WRITE FMCTRL1 [:=00C0h]
 30. READ SYNDROME0_31_0, SYNDROME0_63_32, SYNDROME0_71_64
 31. READ SYNDROME1_31_0, SYNDROME1_63_32, SYNDROME1_71_64
 32. READ SYNDROME2_31_0, SYNDROME2_63_32, SYNDROME2_71_64
 33. READ SYNDROME3_31_0, SYNDROME3_63_32, SYNDROME3_71_64
 34. READ FLAGSYND
 35. WRITE FMCMD [:=0005h]
 36. WRITE FMCANUM [:=0001h]
 37. WRITE FMCADDR0, FMCADDR1
 38. WRITE FMCTRL1 [:=0001h]
 39. WAIT until FMCSTAT[3]==1
 40. WRITE FMCSTAT [:=0008h]
 41. WRITE FMCMD [:=00E0h]
 42. WRITE FMCDNUM [:=001Fh] (depend on the number of spare data)
 43. WRITE FMCTRL1 [:=0002h]
 44. WRITE DMABASE1
 45. WRITE DMACON1
 46. WRITE DMATCNT1 (according to DMACON1's BL)
 47. WRITE DMACOM1
 48. WAIT INTERRUPT (IODMA)
 49. IRQ HANDLER (IODMA)
 50. CLEAR INTERRUPT PENDING (IODMA)
 51. WAIT until FMCSTAT[4]==1
 52. WRITE FMCSTAT [:=0001h]



ELECTRONICS

13

SECURE DIGITAL CARD INTERFACE (SDCI)

OVERVIEW

The S5L8700X Secure Digital Card Interface (SDCI) can interface for SD memory card, Multi-Media Card(MMC) and SDIO device

FEATURES

- Supports MultiMediaCard Specification Version 4.0 and previous Versions
- Supports SD Memory Card Specification Version 1.0
- Supports SDIO Card Specification Version 1.1
- Cards Clock Rate up to System Clock(PCLK) Divided by 2
- 64 bytes FIFO for data Transmit (depth 16)
- 32 bytes FIFO for data Receive (depth 8)
- CRC7 & CRC16 Generator/Checker
- Normal, and DMA Data Transfer Mode
- Support for Block and Multi-block Data Read and Write
- Support for three different data bus width modes: 1bit(default), 4bit and 8bit
 - Multi-Media Card : 1bit(all Versions), 4bit and 8bit(only Version 4.0)
 - Secure Digital Card : 1bit and 4bit (Version 1.0)
- Can Be Directly Connected to the AMBA Peripheral Bus (APB) Version 2.0
- This SD card interface only supports 1 burst and 8 burst operation when using IODMA.

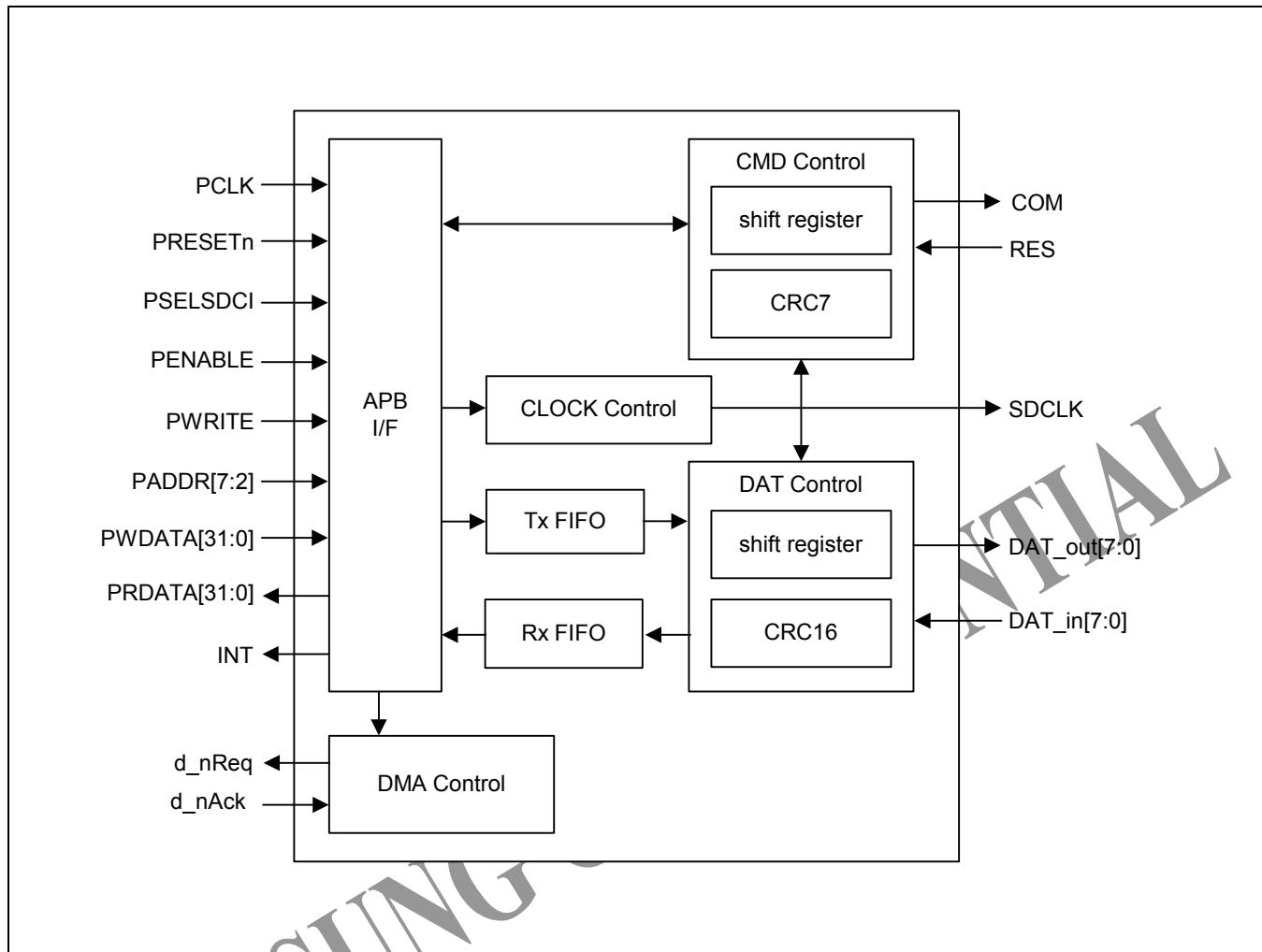
BLOCK DIAGRAM

Figure 13-1. SDCI Block Diagram

Table 13-1. Pin Description

Pin Name	Width	I/O	Description
System Signals			
PCLK	1	I	Clock
PRESETn	1	I	Reset
APB Signals			
PSELSDCI	1	I	Selection on APB
PENABLE	1	I	Enable in APB
PWRITE	1	I	Write/Read in APB
PADDR	6	I	Address in APB
PWADDR	32	I	Write data in APB
PRADDR	32	O	Read data in APB
External Signal			
wp_dect_input	1	I	Write protect pin input
response_input	1	I	Response input (CMD)
dat0_input	1	I	Data0 input (DAT0)
dat1_input	1	I	Data1 input (DAT1)
dat2_input	1	I	Data2 input (DAT2)
dat3_input	1	I	Data3 input (DAT3)
dat4_input	1	I	Data4 input (DAT4)
dat5_input	1	I	Data5 input (DAT5)
dat6_input	1	I	Data6 input (DAT6)
dat7_input	1	I	Data7 input (DAT7)
clk_input	1	I	Clock Input (SDCLK)
clk_output	1	O	Clock output (SDCLK)
command_output	1	O	Command output (CMD)
command_enable	1	O	Command output enable
dat0_output	1	O	Data0 output (DAT0)
dat1_output	1	O	Data1 output (DAT1)
dat2_output	1	O	Data2 output (DAT2)
dat3_output	1	O	Data3 output (DAT3)
dat4_output	1	O	Data4 output (DAT4)
dat5_output	1	O	Data5 output (DAT5)
dat6_output	1	O	Data6 output (DAT6)
dat7_output	1	O	Data7 output (DAT7)
dat0_enable	1	O	Data0 output enable
dat1_enable	1	O	Data1 output enable
dat2_enable	1	O	Data2 output enable

dat3_enable	1	O	Data3 output enable
dat4_enable	1	O	Data4 output enable
dat5_enable	1	O	Data5 output enable
dat6_enable	1	O	Data6 output enable
dat7_enable	1	O	Data7 output enable
INT	1	O	Interrupt signal
DMA relative Signal			
d_nAck	1	I	DMA request signal
d_nReq	1	O	DMA acknowledge signal

SAMSUNG CONFIDENTIAL

SDCI OPERATION

A serial clock line is synchronized with the command and 4 data lines for shifting and sampling of the information. Making the appropriate bit settings to the SDCI_CTRL register depends on the transmission frequency.

SDCI Configuration

After a hardware reset, by default, the SDCI pins are deselected and the user must configure the GPIO controller to assign PIOs to SDCI peripheral functions. For details, refer to the GPIO datasheet.

CMD Path Programming

- Operation of broadcast commands (bc, bcr) and addressed commands (ac)
 1. Write the argument value to SDCI_ARGU register.
 2. Write the command information to SDCI_CMD register.
 - Confirm the ready of command transmission when the specific flag of SDCI_DSTA[0].
 3. Write the command start bit to SDCI_CMD register.
 4. Check the status of SDCI command operation.
 - Command transmission is in progress, the SDCI_DSTA[1] is set.
 - Command transmission is completed, the SDCI_DSTA[2] is set.
 - Response reception is in progress, the SDCI_DSTA[3] is set.
 - Response reception is completed, the SDCI_DSTA[4] is set.
 5. Check the Status of response
 - If command response time-out error occur, the SDCI_DSTA[15] is set.
 - If response end bit error occur, the SDCI_DSTA[16] is set.
 - If response index error occur, the SDCI_DSTA[17] is set.
 - If response CRC error occur, the SDCI_DSTA[18] is set.
 6. Check the Card Response, read from SDCI_RESP0~3 register.
 7. Clear the corresponding flag of the SDCI_DSTA register by write the SDCI_STAC register.

DAT Path Programming (not use DMA)

- Operation of addressed data transfer commands (adtc)
 1. Write the Data block length to the SDCL_DCTRL register.
 2. FIFO reset by writing the SDCL_DCTRL register.
 3. **Do CMD path Programming**
 4. Write the Data transmission start bit to SDCL_DCTRL[5:4] register(only data write operation).
 5. When Write Operation, Write Tx data to SDCL_TXRX register while TxFIFO is available by checking SDCL_FSTA[3:2].
 6. When Read Operation, Read Rx data to SDCL_TXRX register while RxFIFO is available by checking SDCL_FSTA[1:0].
 7. Check the status of SDCI data operation.
 - Data transmission/reception is in progress, the SDCL_DSTA[5] is set.
 - Data block transmission/reception is completed, the SDCL_DSTA[6] is set
 - Data CRC data transmission/reception is completed, the SDCL_DSTA[7] is set.
 - CRC status token reception is completed, the SDCL_DSTA[8] is set, only write command.
 - Card busy state, the SDCL_DSTA[9] is set.
 8. Check the Status of data transfer
 - If write data CRC status token has error, set the SDCL_DSTA[22] is set, CRC status token value is set SDCL_DSTA[21:19].
 - If read data CRC error occur, set the SDCL_DSTA[23] is set.
 - If read data end bit error occur, set the SDCL_DSTA[31:24] are set.
 9. Clear the corresponding flag of the SDCL_DSTA register by write the SDCL_STAC register.
 10. If Multiple data block transfer, repeat 4~9.

DAT Path Programming (Using DMA)

1. Configure SDCI as DMA mode, control register set in DMA module.
2. Write the Data block length to the SDCL_DCTRL register.
3. FIFO reset by writing the SDCL_DCTRL register.
4. **Do CMD path Programming**
5. Write the Data Transmission start bit to SDCL_DCTRL[5:4] register(only data write operation).
6. The SDCI requests DMA service.
7. DMA transmits data to the SDCI as DMA configuration.
8. Check the Status of data Transfer.
9. If Multiple data block transfer, repeat 1, 5~8.

NOTE

1. In case of 136bit response, the CRC error should be detected after receiving exact response data from SDCard or MMC. User should check the CRC of receive response by software.
2. User should check the Read, Write and Erase Time-out error.

Bus Testing Procedure (only use MMC Version 4.0 or higher)

1. Send CMD19 to the card.
2. Start test pattern transmission by writing the SDCI_DCTRL register.
Do not need to select the other four(BLK_LEN, TRCONT, RXFIFORST, TXFIFORST).
3. Write specific data pattern in SDCI_DATA0 register. (See Table13-3)
4. Start test pattern reception by writing the SDCI_DCTRL register.
5. Send CMD14 to the card.
6. Detect bus width of MMC by reading data pattern in SDCI_DATA0 register. (See Table13-3)

Table 13-2. Data Pattern for Bus Test

Write/Read	Data Pattern
Write	32'h0000_AA55
Read	32'h0000_55AA (MMC support 1bit, 4bit and 8bit) 32'h0000_050A (MMC support 1bit and 4bit) 32'h0000_0100 (MMC only support 1bit)

SDIO Operation Procedure

There are two functions of SDIO operation: SDIO interrupt receiving and Read Wait Request generation. There two functions can operate when SDIO_INT_EN bit and SDIO_INT_EN bit of SDIO_CSR register is activated respectively. And two functions have the steps and conditions like below.

SDIO Interrupt

In SD 1bit mode, Interrupt is received through all range from DAT_in[1] pin.

In SD 4bit mode, DAT_in[1] pin is shared between data receiving and interrupt receiving.

When interrupt detection range(Interrupt Period) is :

1. Single Block : the time between A and B
 - A : 2clocks after the completion of a data packet
 - B : The completion of sending the end bit of the next withdata command
2. Multi Block, SDIO_INT_PERIOD = 1 : the time between A and B, restart at C
 - A : 2clocks after the completion of a data packet

- B : 2clocks after A
 - C : 2clocks after the end bit of the abort command response
3. Multi Block, SDIO_INT_PERIOD = 0 : the time between A and B, restart at A
- A : 2clocks after the completion of a data packet
 - B : 2clocks after A
 - In case of last block, interrupt period begins at A, but not ends at B(CMD53 case)

Read Wait Request

Regardless of 1bit or 4bit mode, Read Wait Request signal transmits to DAT_out[2] pin in condition of below.

1. In read multiple operation, request signal transmission begins at 2clocks after the end of the data block
2. Transmission ends when user sets to one SDIO_RW_STOP bit of SDIO_CSR register.

SAMSUNG CONFIDENTIAL

Table 13-3. SDCI Register

Name	Address	Width	R/W	Description	Initial Value
SDCI_CTRL	0x3C30_0000	32bit	R/W	Control Register	0x0000_0000
SDCI_DCTRL	0x3C30_0004	32bit	R/W	Data Control Register	0x0200_0000
SDCI_CMD	0x3C30_0008	32bit	R/W	Command Register	0x0000_0000
SDCI_ARGU	0X3C30_000C	32bit	R/W	Argument Register	0x0000_0000
SDCI_STATE	0x3C30_0010	32bit	R	State Register	0x0000_0000
SDCI_STAC	0x3C30_0014	32bit	W	Status Clear Register	0x0000_0000
SDCI_DSTA	0x3C30_0018	32bit	R	Data Status Register	0x0000_2000
SDCI_FSTA	0x3C30_001C	32bit	R	FIFO Status Register	0x0000_0005
SDCI_RESP0	0x3C30_0020	32bit	R	Response0 Register	0x0000_0000
SDCI_RESP1	0x3C30_0024	32bit	R	Response1 Register	0x0000_0000
SDCI_RESP2	0x3C30_0028	32bit	R	Response2 Register	0x0000_0000
SDCI_RESP3	0x3C30_002C	32bit	R	Response3 Register	0x0000_0000
SDCI_CLKDIV	0x3C30_0030	32bit	R/W	Clock Divider Register	0x0000_0000
SDIO_CSR	0x3C30_0034	32bit	R/W	SDIO Control & Status Register	0x0000_0000
SDIO_IRQ	0x3C30_0038	32bit	R/W	Interrupt Source Register	0x0000_0000
SDIO_IRQ_MAS K	0x3C30_003C	32bit	R/W	Interrupt Mask Register	0x0000_0000
SDCI_DATA0 ~ SDCI_DATA7	0x3C30_0040 ~ 0x3C30_005C	32bit	R/W	Data Register	0x0000_0000

SDCI Control Register (SDCI_CTRL)

Bit	Name	Description
31:8	Reserved	
7	CLK_SEL	Response/Data input detect clock select 0 Response/Data input detected by PCLK. 1 Response/Data input detected by feedback SDCLK.
6	DMA_REQ_CON	Select DMA request condition, only Read Command 0 DMA request when Receive FIFO is not empty. 1 DMA request when Receive FIFO is full.
5	L_ENDIAN	Select Host endian 0 Big endian 1 Little endian
4	DMA_EN	DMA enable 0 DMA disable 1 DMA enable
3:2	BUS_WIDTH	Select the card bus width. 00 1-bit 01 4-bit 10 8-bit 11 Reserved Caution : 8-bit only use in MMC version 4.0 or higher
1	CARD_TYPE	Select the card type 0 SD type (SD memory card and SDIO card) 1 MMC type
0	SDCIEN	SDCI block enable 0 Disable the SDCI 1 Enable the SDCI

SDCI Data Control Register (SDCI_DCTRL)

Bit	Name	Description
31:16	BLK_LEN	Determine the data block size (unit : byte)
15:8	Reserved	
7:6	BUS_TEST	Start Bus Testing Procedure (only use MMC Version 4.0 or higher) 00 No effect 01 Test Pattern Transmission Start 10 Test Pattern Reception Start 11 Reserved Caution : BUS_TEST value must use along with CMD14 or CMD19.
5:4	TRCONT	Start the data transfer, automatically cleared. Only write command (CMD_CLASS ==2'b11). 00 No effect 01 Data Transmission Start 1x Reserved
3:2	Reserved	
1	RXFIFORST	Reset the receive FIFO. 0 No effect 1 Reset the Receive FIFO
0	TXFIFORST	Reset the transmit FIFO. 0 No effect 1 Reset the Transmit FIFO

SDCI Clock Divider Register (SDCI_CDIV)

Bit	Name	Description
31:8	Reserved	
7:0	CLKDIV	Select the SDCLK 0000_0001 SDCLK = PCLK/2 0000_0010 SDCLK = PCLK/4 0000_0100 SDCLK = PCLK/8 0000_1000 SDCLK = PCLK/16 0001_0000 SDCLK = PCLK/32 0010_0000 SDCLK = PCLK/64 0100_0000 SDCLK = PCLK/128 1000_0000 SDCLK = PCLK/256 ELSE Reserved

SDCI Command Register (SDCI_CMD)

Bit	Name	Description
31	CMDSTR	Start the command transfer, automatically cleared. Only writable when SDCI_DSTA[0] = 1. 0 No effect 1 Command transmit start
30:22	Reserved	
21	NCR_NID	Select the clock cycle command to response. 0 N _{CR} (64 clock cycle) 1 N _{ID} (5 clock cycle)
20	RES_SIZE	Select the response size 0 48 bit size response 1 136 bit size response
19	RES_BUSY	Response with an optional busy signal transmitted on data line. 0 no busy signal 1 busy signal
18:16	RES_TYPE	Select response type 000 No Response 001 R1, R1b 010 R2 011 R3 100 R4 101 R5 110 R6 111 Reserved
15:9	Reserved	
8	CMD_RD_WR	Select read/write of commands. Only use Addressed data transfer command (CMD_TYPE == 2'b11) 0 Read command 1 Write command
7:6	CMD_TYPE	Select the command type. 00 Broadcast commands(bc), no response 01 Broadcast commands with response(bcr) 10 Addressed commands(ac), no data transfer on dat lines 11 Addressed data transfer commands(adtc), data transfer on dat lines Caution : Use CMD13 during read/write operation, CMD_TYPE value must maintain the existing value.
5:0	CMD_NUM	Set the command number

SDCI Argument Register (SDCI_ARGU)

Bit	Name	Description
31:0	ARGUMENT	Set the argument value

SDCI State Register (SDCI_STATE)

Bit	Name	Description																																		
31:6	Reserved																																			
5:4	CMD_STATE	<p>State of CMD line</p> <table> <tr><td>00</td><td>CMD_IDLE</td></tr> <tr><td>0</td><td></td></tr> <tr><td>00</td><td>CMD_CMDO: Write command data to the Card.</td></tr> <tr><td>1</td><td></td></tr> <tr><td>01</td><td>CMD_CRCO: Write CRC7 data to the Card.</td></tr> <tr><td>0</td><td></td></tr> <tr><td>01</td><td>CMD_TOUT: Wait response from the Card.</td></tr> <tr><td>1</td><td></td></tr> <tr><td>10</td><td>CMD_RESR: Received response from the Card.</td></tr> <tr><td>0</td><td></td></tr> <tr><td>10</td><td>CMD_INTV: Interval time for N_{RC} cycle.</td></tr> <tr><td>1</td><td></td></tr> <tr><td>11</td><td>Reserved</td></tr> <tr><td>0</td><td></td></tr> <tr><td>11</td><td>Reserved</td></tr> <tr><td>1</td><td></td></tr> </table>	00	CMD_IDLE	0		00	CMD_CMDO: Write command data to the Card.	1		01	CMD_CRCO: Write CRC7 data to the Card.	0		01	CMD_TOUT: Wait response from the Card.	1		10	CMD_RESR: Received response from the Card.	0		10	CMD_INTV: Interval time for N_{RC} cycle.	1		11	Reserved	0		11	Reserved	1			
00	CMD_IDLE																																			
0																																				
00	CMD_CMDO: Write command data to the Card.																																			
1																																				
01	CMD_CRCO: Write CRC7 data to the Card.																																			
0																																				
01	CMD_TOUT: Wait response from the Card.																																			
1																																				
10	CMD_RESR: Received response from the Card.																																			
0																																				
10	CMD_INTV: Interval time for N_{RC} cycle.																																			
1																																				
11	Reserved																																			
0																																				
11	Reserved																																			
1																																				
3:0	DAT_STATE	<p>State of DAT line</p> <table> <tr><td>000</td><td>IDLE</td></tr> <tr><td>0</td><td></td></tr> <tr><td>000</td><td>DAT_RCV: Read data from the Card.</td></tr> <tr><td>1</td><td></td></tr> <tr><td>001</td><td>CRC_RCV: Read CRC data from the Card</td></tr> <tr><td>0</td><td></td></tr> <tr><td>001</td><td>DAT_END: Read data end bit form the Card.</td></tr> <tr><td>1</td><td></td></tr> <tr><td>010</td><td>DAT_SET: Check CMD12 to terminate operation</td></tr> <tr><td>0</td><td></td></tr> <tr><td>010</td><td>DAT_OUT: Write data to the Card.</td></tr> <tr><td>1</td><td></td></tr> <tr><td>011</td><td>CRC_TIME: Ready to write CRC16 data to the Card.</td></tr> <tr><td>0</td><td></td></tr> <tr><td>011</td><td>CRC_OUT: Write CRC16 data to the Card.</td></tr> <tr><td>1</td><td></td></tr> <tr><td>100</td><td>ENDB_OUT: Write data end bit to the Card.</td></tr> </table>	000	IDLE	0		000	DAT_RCV: Read data from the Card.	1		001	CRC_RCV: Read CRC data from the Card	0		001	DAT_END: Read data end bit form the Card.	1		010	DAT_SET: Check CMD12 to terminate operation	0		010	DAT_OUT: Write data to the Card.	1		011	CRC_TIME: Ready to write CRC16 data to the Card.	0		011	CRC_OUT: Write CRC16 data to the Card.	1		100	ENDB_OUT: Write data end bit to the Card.
000	IDLE																																			
0																																				
000	DAT_RCV: Read data from the Card.																																			
1																																				
001	CRC_RCV: Read CRC data from the Card																																			
0																																				
001	DAT_END: Read data end bit form the Card.																																			
1																																				
010	DAT_SET: Check CMD12 to terminate operation																																			
0																																				
010	DAT_OUT: Write data to the Card.																																			
1																																				
011	CRC_TIME: Ready to write CRC16 data to the Card.																																			
0																																				
011	CRC_OUT: Write CRC16 data to the Card.																																			
1																																				
100	ENDB_OUT: Write data end bit to the Card.																																			

		0
	100	ENDB_STOD
	1	: Write data end bit to the Card for stop transmission
	101	DAT_CRCR: Read CRC status from the Card.
	0	
	101	CARD_PRG: Data line busy(write command)
	1	
	110	DAT_BUSY
	0	: Data line is busy(erase command, busy command).
	110	Reserved
	1	
	111	Reserved
	0	
	111	Reserved
	1	

SAMSUNG CONFIDENTIAL

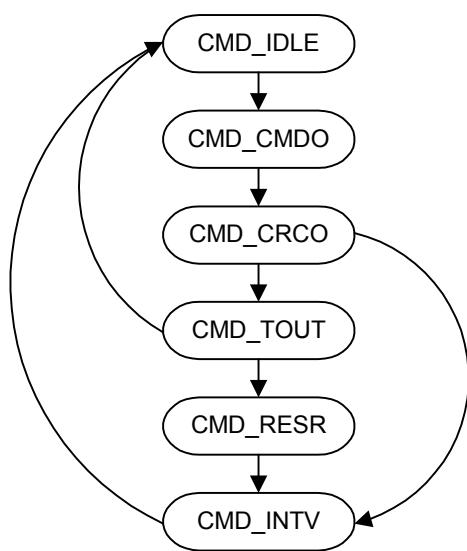


Figure 13-2. CMD_STATE Diagram

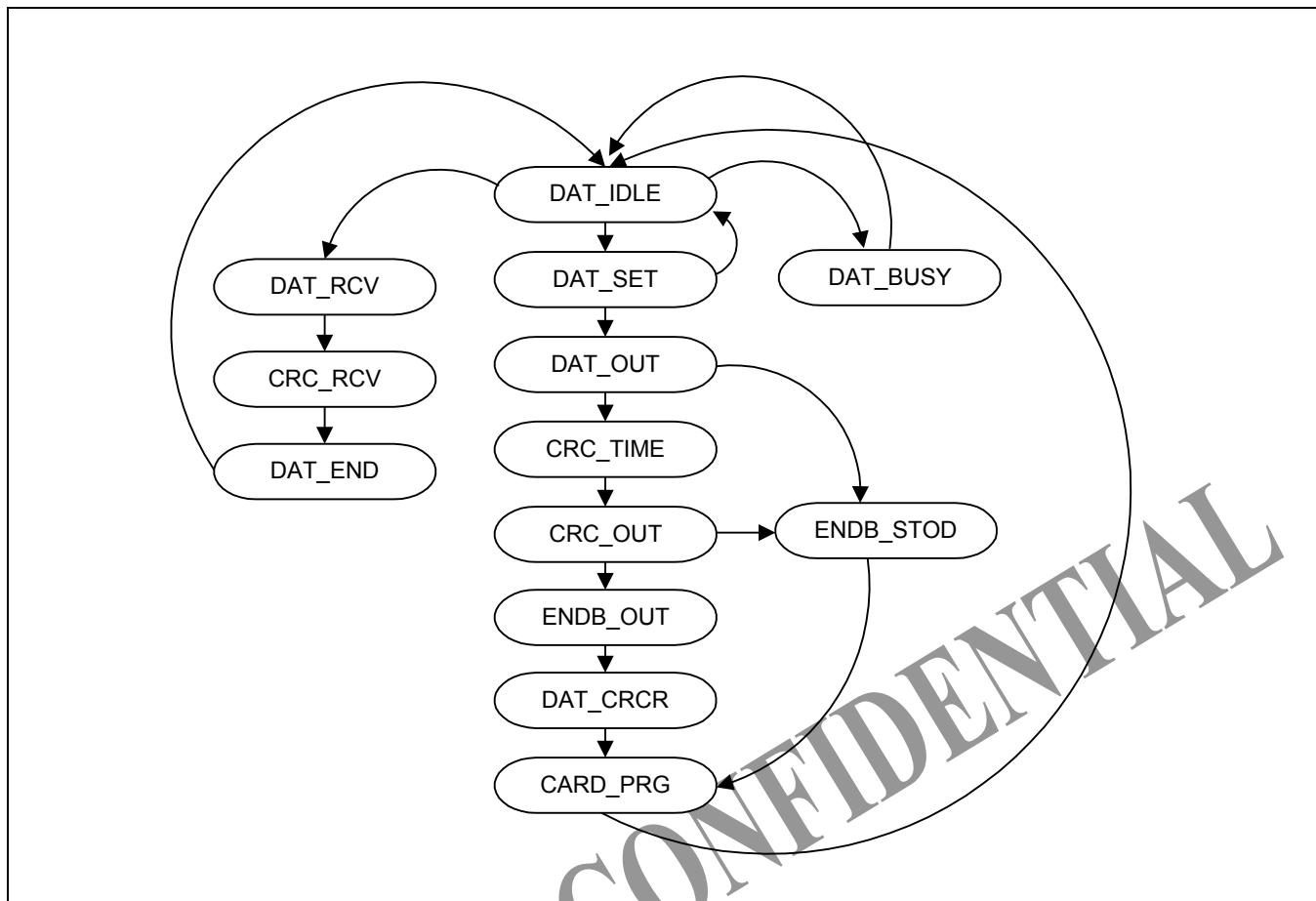


Figure 13-3. DAT_STATE Diagram

SDCI Status Clear Register (SDCI_STAC)

Bit	Name	Description
31	CLR_RD_DATENDE7	Clear RD_DATENDE7 at SDCI_DSTA 0 No effect 1 Clear the RD_DATEMDE7 bit
30	CLR_RD_DATENDE6	Clear RD_DATENDE6 at SDCI_DSTA 0 No effect 1 Clear the RD_DATEMDE6 bit
29	CLR_RD_DATENDE5	Clear RD_DATENDE5 at SDCI_DSTA 0 No effect 1 Clear the RD_DATEMDE5 bit
28	CLR_RD_DATENDE4	Clear RD_DATENDE4 at SDCI_DSTA 0 No effect 1 Clear the RD_DATEMDE4 bit
27	CLR_RD_DATENDE3	Clear RD_DATENDE3 at SDCI_DSTA 0 No effect 1 Clear the RD_DATEMDE3 bit
26	CLR_RD_DATENDE2	Clear RD_DATENDE2 at SDCI_DSTA 0 No effect 1 Clear the RD_DATEMDE2 bit
25	CLR_RD_DATENDE1	Clear RD_DATENDE1 at SDCI_DSTA 0 No effect 1 Clear the RD_DATEMDE1 bit
24	CLR_RD_DATENDE0	Clear RD_DATENDE0 at SDCI_DSTA 0 No effect 1 Clear the RD_DATEMDE0 bit
23	CLR_RD_DATCRCE	Clear RD_DATCRCE at SDCI_DSTA 0 No effect 1 Clear the RD_DATCRCE bit
22	CLR_WR_DATCRCE	Clear WR_DATCRCE at SDCI_DSTA 0 No effect 1 Clear the WR_DATCRCE bit
21:19	Reserved	
18	CLR_RESCRCE	Clear RESCRCE at SDCI_DSTA 0 No effect 1 Clear the RESCRCE bit
17	CLR_RESINDE	Clear RESINDE at SDCI_DSTA 0 No effect 1 Clear the RESINDE bit

16	CLR_RESENDE	Clear RESENDE at SDCI_DSTA 0 No effect 1 Clear the RESENDE bit
15	CLR_RESTROUTE	Clear RESTROUTE at SDCR_DSTA 0 No effect 1 Clear the RESTROUTE bit
14:9	Reserved	
8	CLR_CRC_STAEND	Clear CRC_STAEND at SDCI_DSTA 0 No effect 1 Clear the CRC_STAEND bit
7	CLR_DAT_CRCEND	Clear DAT_CRCEND at SDCI_DSTA 0 No effect 1 Clear the DAT_CRCEND bit
6	CLR_DATEND	Clear DAT_END at SDCI_DSTA 0 No effect 1 Clear the DATEND bit
5	Reserved	
4	CLR_RESEND	Clear RESEND at SDCI_DSTA 0 No effect 1 Clear the RESEND bit
3	Reserved	
2	CLR_CMDEND	Clear CMDEND at SDCI_DSTA 0 No effect 1 Clear the CMDEND bit
1:0	Reserved	

SDCI Data Status Register (SDCI_DSTA)**Note:** Clear Condition

A : According to the SDCI interface state.

C : Clear by write '1' to corresponding bit of SDCI_DSTA.

Bit	Name	Description	Clear Condition
31	RD_DATENDE7	Read data end bit error on dat7 0 No Read Data End Bit Error of DAT7 occurs 1 Read Data End Bit Error of DAT7 occurs	C
30	RD_DATENDE6	Read data end bit error on dat6 0 No Read Data End Bit Error of DAT6 occurs 1 Read Data End Bit Error of DAT6 occurs	C
29	RD_DATENDE5	Read data end bit error on dat5 0 No Read Data End Bit Error of DAT5 occurs 1 Read Data End Bit Error of DAT5 occurs	C
28	RD_DATENDE4	Read data end bit error on dat4 0 No Read Data End Bit Error of DAT4 occurs 1 Read Data End Bit Error of DAT4 occurs	C
27	RD_DATENDE3	Read data end bit error on dat3 0 No Read Data End Bit Error of DAT3 occurs 1 Read Data End Bit Error of DAT3 occurs	C
26	RD_DATENDE2	Read data end bit error on dat2 0 No Read Data End Bit Error of DAT2 occurs 1 Read Data End Bit Error of DAT2 occurs	C
25	RD_DATENDE1	Read data end bit error on dat1 0 No Read Data End Bit Error of DAT1 occurs 1 Read Data End Bit Error of DAT1 occurs	C
24	RD_DATENDE0	Read data end bit error on dat0 0 No Read Data End Bit Error of DAT0 occurs 1 Read Data End Bit Error of DAT0 occurs	C
23	RD_DATCRCE	Read data has CRC error 0 No Read Data CRC Error occurs 1 Read Data CRC Error occurs	C
22	WR_DATCRCE	Write data CRC status response has error 0 No CRC status Token Error occurs 1 CRC status Token Error occurs	C
21:19	WR_CRC_STATUS	Write data CRC status response value 010 Non erroneous transmission 101 Transmission error 111 Card error else Reserved	A



18	RESCRCE	Response CRC error. Don't care, when RES_CLASS = 3'001. 0 No Response CRC Error occurs 1 Response CRC Error occurs	C
17	RESINDE	Response index error 0 No Response Index Error occurs 1 Response Index Error occurs	C
16	RESENDE	Response end bit error 0 No Response End Bit Error occurs 1 Response End Bit Error occurs	C
15	RESTROUTE	Response timeout error (Ncr, Nid) 0 No Command to Response timeout error occurs 1 Command to Response timeout error occurs	C
14	WP_DECT_INPUT	WP pin status 0 WP Pin status is Low 1 WP Pin status is High	A
13	DAT0_STATUS	DAT0 pin status 0 DAT0 pin is Low 1 DAT0 pin is High	A
12	SDCLK_HOLD	SD Clock status 0 SD Clock is not holding 1 SD Clock is holding To clear SDCLK_HOLD, disable SDCIEN in SDCI control register	A
11:10	Reserved		
9	DAT_BUSY	Data line busy 0 card is not busy 1 card is busy This status is direct connected inverted DAT0 of Card.	A
8	CRC_STAEND	Write data CRC status token receive end 0 CRC status token reception is not ended 1 CRC status token reception is ended	C
7	DAT_CRCEND	Data CRC transmit/receive end 0 CRC transmission/reception is not ended 1 CRC transmission/reception is ended	C
6	DATEND	Data transmit/receive end 0 Data transmission/reception is not ended 1 Data transmission/reception is ended	C
5	DATPRO	Data transfer in progress	A

		0 Data transmission/reception is not in progress 1 Data transmission/reception is in progress	
4	RESEND	Response receive end 0 Response reception is not ended 1 Response reception is ended	C
3	RESPRO	Response receive in progress 0 Response reception is not in progress 1 Response reception is in progress	A
2	CMDEND	Command transfer end 0 Command transmission is not ended 1 Command transmission is ended	C
1	CMDPRO	Command transfer in progress 0 Command transmission is not in progress 1 Command transmission is in progress	A
0	CMDRDY	Command ready 0 Command transfer is not ready 1 Command transfer is ready (SDCI_CMD, SDCI_ARGU set complete)	A

SDCI FIFO Status Register (SDCI_FSTA)

Note: Clear Condition -> According to the SDCI interface state

Bit	Name	Description
31:4	Reserved	
3	TX_FIFO_FULL	Tx FIFO full 0 Transmit FIFO is not full 1 Transmit FIFO is full
2	TX_FIFO_EMPTY	Tx FIFO empty 0 Transmit FIFO is not empty 1 Transmit FIFO is empty
1	RX_FIFO_FULL	Rx FIFO full 0 Receive FIFO is not full 1 Receive FIFO is full
0	RX_FIFO_EMPTY	Rx FIFO empty 0 Receive FIFO is not empty 1 Receive FIFO is empty

SDIO Control & Status Register (SDIO_CSR) - (optional for SDIO card)

Bit	Name	Description
31:5	Reserved	
4	SDIO_INT_PERIOD	SDIO interrupt period. 0 more 2 cycle 1 exactly 2 cycle (only multiple block in SD 4-bit mode)
3	SDIO_RW_STOP	SDIO read wait request stop 0 not active 1 stop the read wait request (if this bit sets, the stalled transfer is released) Caution: If SDIO_RW_STOP is high, it is cleared automatically when SDIO_STA bit is changed from high(=1'b1) to low(=1'b0) value.
2	SDIO_RW_REQ	SDIO read wait request 0 not active 1 occur the read wait request(if this bit sets, the transfer is stalled) Caution: If SDIO_RW_REQ is high, it is cleared automatically when user set to one SDIO_RW_STOP bit.
1	SDIO_INT_EN	SDIO interrupt enable (optional) 0 disable (not receive interrupt from SDIO card) 1 enable (receive interrupt from SDIO card)
0	SDIO_RW_EN	SDIO read wait enable (optional) 0 disable (not signal read wait to SDIO card) 1 enable (signal read wait to SDIO card)

SDCI Interrupt Source Register (SDCI_IRQ)

Bit	Name	Description
31:3	Reserved	
2	READ_WAIT_INT	When Read Wait Request signal transmit to card. (for SDIO) CPU can clear this interrupt by writing "1" 0 not occur 1 read wait request occur
1	IOCARD_IRQ_INT	When host receive SDIO Interrupt from card. (for SDIO) CPU can clear this interrupt by writing "1"

		0 not receive 1 SDIO interrupt receive
0	DAT_DONE_INT	When data of 1block transfer are finished. CPU can clear this interrupt by writing "1" 0 not finish 1 data transfer finish

SDCI Interrupt Mask Register (SDCI_IRQ_MASK)

Bit	Name	Description
31:3	Reserved	
2	MASK_READ_WAIT_INT	Determines SDCI generate an interrupt if read wait request occur. 0 disable 1 interrupt enable
1	MASK_IOCARD_IRQ_INT	Determines SDCI generate an interrupt if host receive SDIO Interrupt from card. 0 disable 1 interrupt enable
0	MASK_DAT_DONE_INT	Determines SDCI generate an interrupt if all data transfer are finished 0 disable 1 interrupt enable

SDCI Response3 Register (SDCI_RESP3)

Bit	Name	Description						
31:0	RESPONSE3	<p>The RES_SIZE determines the value</p> <table border="1"> <tr> <td>RES_SIZE</td> <td>RESPONSE3</td> </tr> <tr> <td>0</td> <td>32'h0000_0000</td> </tr> <tr> <td>1</td> <td>response[127:96]</td> </tr> </table>	RES_SIZE	RESPONSE3	0	32'h0000_0000	1	response[127:96]
RES_SIZE	RESPONSE3							
0	32'h0000_0000							
1	response[127:96]							

SDCI Response2 Register (SDCI_RESP2)

Bit	Name	Description						
31:0	RESPONSE2	<p>The RES_SIZE determines the value</p> <table border="1"> <tr> <td>RES_SIZE</td> <td>RESPONSE2</td> </tr> <tr> <td>0</td> <td>32'h0000_0000</td> </tr> <tr> <td>1</td> <td>response[95:64]</td> </tr> </table>	RES_SIZE	RESPONSE2	0	32'h0000_0000	1	response[95:64]
RES_SIZE	RESPONSE2							
0	32'h0000_0000							
1	response[95:64]							

SDCI Response1 Register (SDCI_RESP1)

Bit	Name	Description						
31:0	RESPONSE1	<p>The RES_SIZE determines the value</p> <table border="1"> <tr> <td>RES_SIZE</td> <td>RESPONSE1</td> </tr> <tr> <td>0</td> <td>32'h0000_0000</td> </tr> <tr> <td>1</td> <td>response[63:32]</td> </tr> </table>	RES_SIZE	RESPONSE1	0	32'h0000_0000	1	response[63:32]
RES_SIZE	RESPONSE1							
0	32'h0000_0000							
1	response[63:32]							

SDCI Response0 Register (SDCI_RESP0)

Bit	Name	Description						
31:0	RESPONSE0	<p>The RES_SIZE determines the value</p> <table border="1"> <tr> <td>RES_SIZE</td> <td>RESPONSE0</td> </tr> <tr> <td>0</td> <td>response[39:8]</td> </tr> <tr> <td>1</td> <td>response[31:0]</td> </tr> </table>	RES_SIZE	RESPONSE0	0	response[39:8]	1	response[31:0]
RES_SIZE	RESPONSE0							
0	response[39:8]							
1	response[31:0]							

SDCI Data0~7 Register (SDCI_DATA0 ~ SDCI_DATA7)

Bit	Name	Description
31:0	DATA	Data buffer for transmit/receive

NOTE: If data is not aligned to word (4byte), SDCI_DATA register value is following table (big_endian).

When Data Read

Write Data	SDCI_DATA[31:24]	SDCI_DATA[13:16]	SDCI_DATA[15:8]	SDCI_DATA[7:0]
4n+1 byte	Read data[7:0]	stuff bits	stuff bits	stuff bits
4n+2 byte	Read data[7:0]	Read data[15:8]	stuff bits	stuff bits
4n+3 byte	Read data[7:0]	Read data[15:8]	Read data[23:16]	stuff bits

When Data Write

Read Data	SDCI_DATA[31:24]	SDCI_DATA[13:16]	SDCI_DATA[15:8]	SDCI_DATA[7:0]
4n+1 byte	0x00	0x00	0x00	Write data[7:0]
4n+2 byte	0x00	0x00	Write data[7:0]	Write data[15:8]
4n+3 byte	0x00	Write data[7:0]	Write data[15:8]	Write data[23:16]

NOTES

SAMSUNG CONFIDENTIAL

14 MEMORY STICK HOST CONTROLLER

OVERVIEW

- Protocol is started by writing to the command register from the CPU
- Data is requested by DMA or Interrupt to the CPU when entering the data period
- RDY timeout period can be set by the number of serial clock
- Interrupt can also be output to the CPU when a timeout occurs
- CRC can be turned off during test mode
- 16-bit access
- The output from FIFO is little endian

FEATURE

- Built-in 8-byte (4-word) FIFO buffers for transmit and receive respectively
- Built-in CRC circuit
- Transfer clock up to 80 MHz (External input. When using the SONY C5MX-HB-T library.)
- DMA supported
- Automatic command execution (can be turned on/off) when an INT from the Memory Stick is detected
- MagicGate supported
- Approx. 8K gates (When using the SONY C5MX-HB-T library)
- All registers are described positive edge flip flop (for SCAN)

BLOCK DIAGRAM

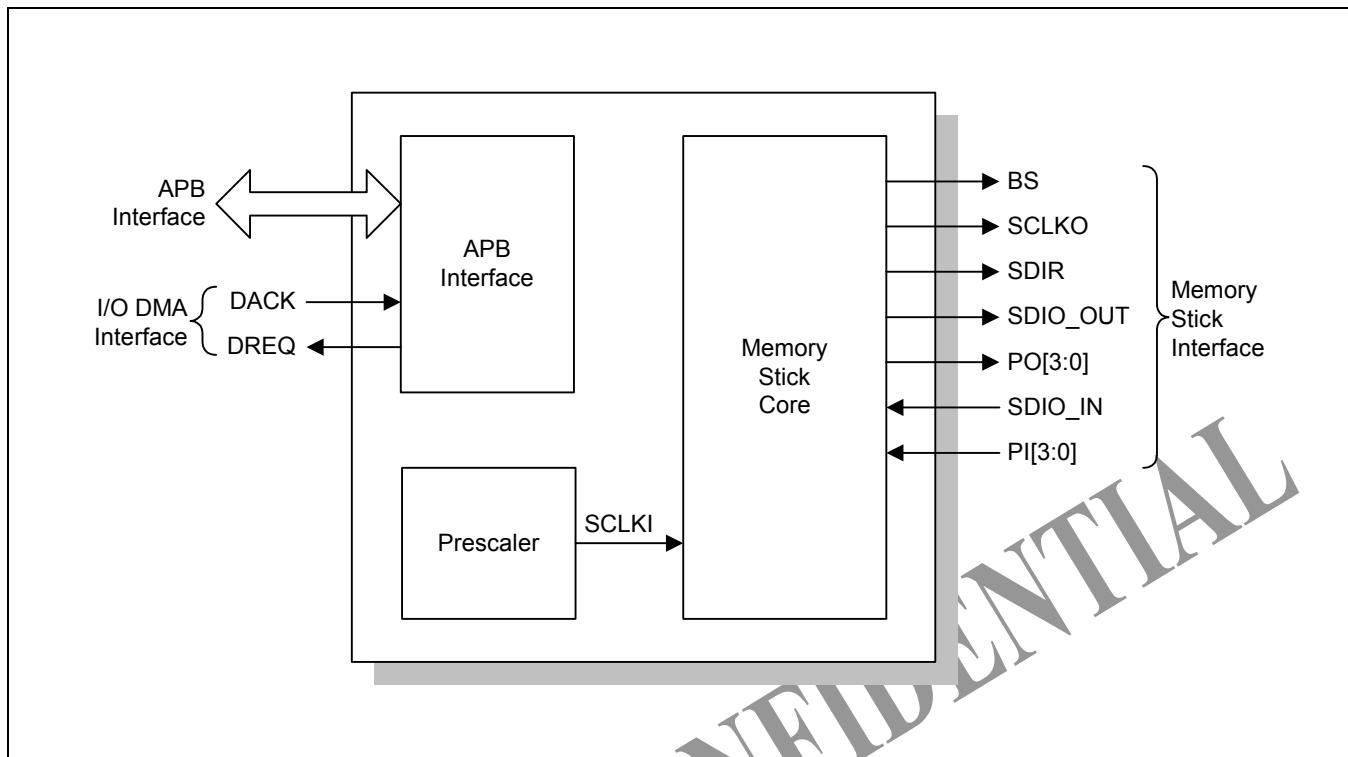


Figure 14-1. Memory Stick Host Controller Block Diagram

REGISTERS

PRESCALER CONTROL REGISTER(MSPRE)

Register	Address	R/W	Description	Reset Value
MSPRE	0x3C60_0000	R/W	Prescaler Register	32 bits

MSPRE	Bit	Description	Initial State
CLK_EN	[8]	SCLK (XSCLK) output or not 0 = Disable the SCLK (XSCLK) output 1 = Enable the SCLK (XSCLK) output	0
PRESCALE	[7:0]	Prescale register to generate the SCLK (XSCLK) from the main clock. $SCLK [Hz] = \{\text{main clock [Hz]} / (\text{MS_PRE} + 1)\} / 2$	0

INTERRUPT ENABLE REGISTER(MSINTEN)

Register	Address	R/W	Description	Reset Value
MSINTEN	0x3C60_0004	R/W	Interrupt Enable Register	32 bits

MSINTEN	Bit	Description	Initial State
CORE_INT_STAT	[12]	Interrupt generated by the memory stick host controller. Writing one clears this flag. User should write one to this field and read the MSINT register to deactivate the interrupt signal from the memory stick host controller. The sequence is not important.	0
RBE_INT_STAT	[11]	RBE interrupt status. Writing one clears this flag and deactivates the interrupt.	1
RBF_INT_STAT	[10]	RBF interrupt status. Writing one clears this flag and deactivates the interrupt.	0
TBE_INT_STAT	[9]	TBE interrupt status. Writing one clears this flag and deactivates the interrupt.	1
TBF_INT_STAT	[8]	TBF interrupt status. Writing one clears this flag and deactivates the interrupt.	0
RBE_INT_EN	[3]	Interrupt enable/disable for the RBE. 0 = Disable the interrupt from RBE 1 = Enable the interrupt from RBE	0
RBF_INT_EN	[2]	Interrupt enable/disable for the RBF. 0 = Disable the interrupt from RBF 1 = Enable the interrupt from RBF	0
TBE_INT_EN	[1]	Interrupt enable/disable for the TBE. 0 = Disable the interrupt from TBE 1 = Enable the interrupt from TBE	
TBF_INT_EN	[0]	Interrupt enable/disable for the TBF. 0 = Disable the interrupt from TBF 1 = Enable the interrupt from TBF	0

COMMAND REGISTER(MSCMD)

Register	Address	R/W	Description	Reset Value
MSCMD	0x3C60_1000	R/W	Command Register	32 bits

ADCPARA	Bit	Description	Initial State
PID	[15:12]	Packet ID Writing to Command register starts the protocol. CRC16bit is transferred during the data period even if the data size is 0. PID command is disabled if the data size is 0 and the NOCRC bit of the Control register 1 is 1. Data cannot be written to the Command register when the RDY bit of the Interrupt Control and Data register is 0. (While the protocol is executing.)	0
DATA_SIZE	[9:0]	Data size for each transfer. The PID determines the value.	0

CONTROL/STATUS REGISTER(MSCTRLSTAT)

Register	Address	R/W	Description	Reset Value
MSCTRLSTAT	0x3C60_100 4	R/W	Control/Status Register	32 bits

MSCTRLSTAT	Bit	Description	Initial State
RST	[15]	Internal reset 0 = Nothing 1 = Internal reset	0h
Reserved	[14]	Reserved to 0	0h
SIEN	[13]	Serial interface enable 0 = Disable the serial interface 1 = Enable the serial interface	0h
Reserved	[12]	Reserved to 0.	0h
NOCRC	[11]	Data is transmitted/received without adding a CRC (16bit) at the end of the data. Normally this bit is set to zero during the operation. 0 = Enable the CRC 1 = Disable the CRC	0h
BSYCNT	[10:8]	RDY timeout time setting (serial clock count). This field is set to the maximum BSY timeout time (BSYCNT x 4 + 2) to wait until the RDY signal is output from the memory stick. RDY timeout error detection is not performed when BSYCNT = 0. Initial value is 05h (22 SCLK).	05h
INT	[7]	Indicates the interrupt status 0 = When an interrupt condition is not generated 1 = When an interrupt condition is generated	0h
DRQ	[6]	Indicates DMA request status 0 = When data is not requested 1 = When data is requested	0h
RBE	[3]	Receive buffer empty. 0 = The receive data buffer is not empty 1 = The receive data buffer is empty	1h
RBF	[2]	Receive buffer full. 0 = The receive data buffer is not full 1 = The receive data buffer is full	0h
TBE	[1]	Transmit buffer empty. 0 = The transmit data buffer is not empty 1 = The transmit data buffer is empty	1h
TBF	[0]	Transmit buffer full. 0 = The transmit data buffer is not full 1 = The transmit data buffer is full	0h

OPERATION PERFORMED DURING RESET

Operations when RST is "1".

A value of "1" should be maintained for the RST bit for system clock 2 clocks and SCLKI 2 clocks and then must be returned to "0" in order to perform reset in sync with the clock. If the software-reset operation is executed, the following condition is set.

Register operation (Status after RST = 1 and immediately after RST = 0)

MSCMD register	= 0x00000000
MSCTRLSTAT register	= 0x0000050a
MSFIFO register	= 0x00000000
MSINT register	= 0x00000080
MSPP register	= 0x00000000
MSCTRL2 register	= 0x00000000
MSACD register	= 0x00007001

Output signal

BS (bus state)	= low level
SDIO_OUT (SDIO output)	= low level
SCLKO (memory stick clock)	= low level
INT	= low level
nDRQ (DMA request)	= high level

Internal Operation

- The Transmit/Receive Data Buffers are cleared
- TBE, RBE = 1
- TBF, RBF = 0
- PO = 0

The executing protocol is terminated.

RECEIVE/TRANSMIT DATA BUFFER (MSFIFO)

Register	Address	R/W	Description	Reset Value
MSFIFO	0x3C60_1008	R/W	Receive/Transmit Register	32 bits

MSFIFO	Bit	Description	Initial State
RDATA_BUF	[15:0]	Data buffer for receive – When RBE is one, invalid data is read.	0h
TDATA_BUF	[15:0]	Data buffer for transmit – When TBF is one, invalid data is ignored.	0h

INTERRUPT CONTROL/DATA REGISTER(MSINT)

Register	Address	R/W	Description	Reset Value
MSINT	0x3C60_100C	R/W	Interrupt Control/Data Register	32 bits

MSINT	Bit	Description	Initial State
INTEN	[15]	XINT interrupt enable 0 = XINT interrupt signal output is disabled. 1 = XINT interrupt signal output is enabled.	0h
DRQSL	[14]	0 = XINT output is disabled during data transfer request. 1 = XINT output is enabled during data transfer request.	0h
PINEN	[13]	0 = XINT output is disabled by the change of the value in the XPI[3:0]. 1 = XINT output is enabled by the change of the value in XPI[3:0].	0h
RDY	[7]	Protocol status 0 = Protocol with the memory stick is not ended. 1 = Protocol with the memory stick is ended. This flag is cleared to zero when writing to the Command register.	1h
SIF	[6]	Serial interface interrupt 0 = Serial I/F does not receive an interrupt 1 = Serial I/F receives an interrupt. An interrupt signal is output separately from RDY interrupt.	0h
DRQ	[5]	DMA request 0 = XDRQ (DMA request) is not requested. 1 = When XDRQ (DMA request) output is requested and DRQSL bit is 1.	0h
PIN	[4]	Parallel port input change 0 = Parallel inputs are not changed. 1 = When parallel inputs are changed and PINEN is 1.	0h
CRC	[1]	CRC error 0 = No CRC error occurs. 1 = CRC error occurs. Cleared to zero when data is written to command register (MSCMD). BS output is set to zero, RDY becomes to one, and an interrupt signal is generated.	0h
TOE	[0]	Time out error 0 = No time out error occurs 1 = BSY timeout error occurs Cleared to zero when data is written to the command register (MSCMD) RDY becomes to one and an interrupt signal is output	0h

PARALLEL PORT CONTROL/DATA REGISTER (MSPP)

Register	Address	R/W	Description	Reset Value
MSPP	0x3C60_1010	R/W	Parallel Port Control/Data Register	32 bits

MSPP	Bit	Description	Initial State
PIEN	[15:12]	Parallel port input enable (This bit shall be cleared to 0) 0 = Parallel port inputs are disabled 1 = Parallel port inputs are enabled	0h
POEN	[11:8]	Parallel port output enable (This bit shall be cleared to 0) 0 = Parallel port outputs are disabled 1 = Parallel port outputs are enabled	0h
XPIN	[7:4]	Parallel port input data – XPIN[n] is 1 when the Pln pin is low level and 0 when high level. – It takes 32 SCLK cycles for a value from the parallel input pin PI[3:0] to be reflected at the XPIN[3:0]	0h
POUT	[3:0]	Parallel port output data – High level is output when the POUT[n] is set to 1 – Low level is output when the POUT[n] is set to 0	0h

control register 2 (msctrl2)

Register	Address	R/W	Description	Reset Value
MSCTRL2	0x3C60_1014	R/W	Control Register 2	32 bits

MSCTRL2	Bit	Description	Initial State
ACD	[15]	Auto command enable 0 = Disable the auto command 1 = Enable the auto command after an interrupt is detected. This flag is automatically cleared to zero after Auto Command processing is ended.	0h
RED	[14]	Edge selection for data loading 0 = Serial data is loaded at the rising edge of the clock 1 = Serial data is loaded at the falling edge of the clock	0h

Auto Command is a function used to automatically execute the GET_INT or READ_REG on the host interface. With this function, the interrupt signal from the Memory Stick is detected and the command set in the ACD Command Register is executed. When CRC error or TOE occurs, the Auto Command processing is terminated without performing ACD and an interrupt signal is generated.

ACD COMMAND REGISTER (MSACD)

Register	Address	R/W	Description	Reset Value
MSACD	0x3C60_1018	R/W	ACD Command Register	32 bits

MSACD	Bit	Description	Initial State
APID	[15:12]	PID code (Transport Protocol Command)	7h
ADATA_SIZE	[9:0]	Data size for each transfer. The PID determines the value.	1h

Auto Command is a function used to automatically execute the GET_INT or READ_REG on the host interface. With this function, the interrupt signal from the Memory Stick is detected and the command set in the ACD Command Register is executed.

SAMSUNG CONFIDENTIAL

NOTES

SAMSUNG CONFIDENTIAL

15

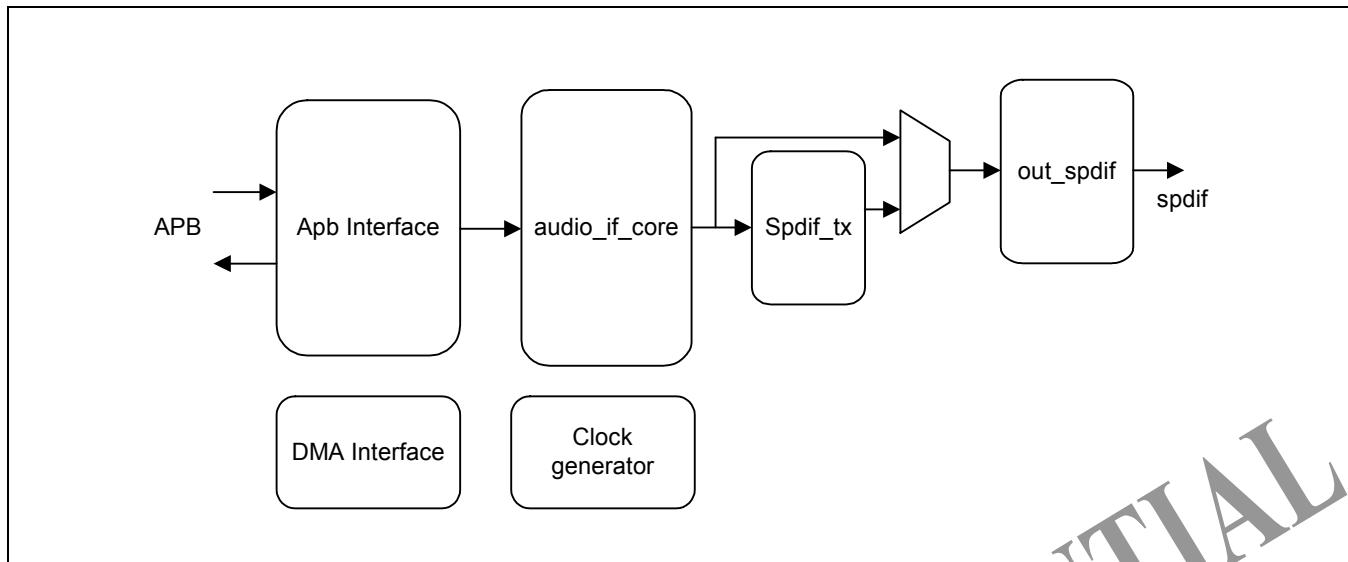
SPDIF TRANSMITTER (SPDIFOUT)

OVERVIEW

This standard describes a serial, un-directional, self-clocking interface for the interconnection of digital audio equipment for consumer and professional applications. When used in a consumer digital processing environment, the interface is primarily intended to carry stereophonic programs, with a resolution of up to 20 bits per sample, an extension to 24 bits per sample being possible. When used in a broadcasting studio environment, the interface is primarily intended to carry monophonic or stereophonic programs, at a 48kHz sampling frequency and with a resolution of up to 24 bits per sample; it may alternatively be used to carry one or two signals sampled at 32 kHz. In both cases, the clock references and auxiliary information are transmitted along with the program. Provision is also made to allow the interface to carry data related to computer software.

FEATURE

- SPDIFOUT module only supports the consumer application in S5L8700X.
- Linear PCM up to 24-bit per sample is supported.
- Non-linear PCM formats such as AC3, MPEG1 and MPEG2 are also supported.
- 2 x 24-bit buffers which is alternately filled with data

BLOCK DIAGRAM**Figure 15-1. SPDIFOUT Block Diagram**

- APB interface block : defines register banks to control the driving of SPDIFOUT module and data buffers to store linear or non-linear PCM data.
- DMA interface block : requests DMA service to IODMA depending on the status of data buffer in APB Interface block
- Clock Generator block : makes 128fs(sampling frequency) clock used in out_spdif block from system audio clock(MCLK)
- audio_if_core block : acts as interface block between data buffer and out_spdif block. Finite-state machine controls the flow of PCM data.
- spdif_tx block : inserts burst preamble and executes zero-stuffing in the nonlinear PCM stream. Linear PCM data are bypassed by spdif_tx module.
- out_spdif block : makes spdif format. It inserts 4-bit preamble, 16- or 20- or 24-bit data, user-data bit, validity bit, channel status bit and parity bit into the appropriate position of 32-bit word. It modulates each bit to bi-phase format.

PIN DESCRIPTION

Pin Name	Width	I/O	Description
PCLK	1	I	Global clock
i-dac-clock	1	I	Global audio main clock
PRESETn	1	I	Global reset
APB Interface			
PSEL	1	I	Selection in APB
PENABLE	1	I	Enable in APB
PWRITE	1	I	Write/Read in APB
PADDR	3	I	Address in APB
PWDATA	32	I	Write data in APB
PRDATA	32	O	Read data in APB
SPDIFOUT Interface			
o-spdif-data	1	O	SPDIFOUT data output
DMA Request & Interrupt			
nDMAREQ	1	O	Active low DMA request signal
nDMAACK	1	I	Active low DMA acknowledgement signal
spdif-int	1	O	Interrupt of SPDIFOUT module

REGISTERS

Name	Address (virtual)	R/W	Description	Reset
SPDCLKCON	0x3CB0_0000	R/W	Clock Control Register	0x0000 0002
SPDCON	0x3CB0_0004	R/W	Control Register	0x0000 0020
SPDBSTAS	0x3CB0_0008	R/W	Burst Status Register	0x0000 0000
SPDCSTAS	0x3CB0_000C	R/W	Channel Status Register	0x2000 8000
SPDDAT	0x3CB0_0010	W	SPDIFOUT Data Buffer	0x0000 0000
SPDCNT	0x3CB0_0014	R/W	Repetition Count Register	0x0000 0000

SPDIFOUT Clock Control Register (SPDCLKCON)

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

SPDCLKCON	Bit	Description		Initial State
	[31:2]	Reserved		0
SPDIFOUT clock down ready (read only)	[1]	0 = clock-down not ready 1 = clock-down ready		1
SPDIFOUT power on	[0]	0 = power off 1 = power on		0

SPDIFOUT Control Register (SPDCON)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SPDCON	Bit	Description	Initial State
	[31:9]	Reserved	0
Data Buffer Empty Flag (read only)	[8]	0 = not empty 1 = empty	1
Interrupt Status bit	[7]	0 = no interrupt pending 1 = interrupt pending (Interrupt status bit can be reset by writing '1' to this bit. Writing '0' to interrupt status bit has no effect.)	0
Interrupt Enable bit	[6]	0 = interrupt masked 1 = interrupt enable	0
software reset bit	[5]	0 = normal operation 1 = software reset	0
Main Audio Clock Frequency	[4:3]	00 = 256fs 10 = 512fs 01 = 384fs 11 = reserved	0
PCM Data Size	[2:1]	00 = 16 bit 10 = 24 bit 01 = 20 bit 11 = reserved	0
PCM or Stream	[0]	0 = stream 1 = PCM	0

SPDIFOUT Burst Status Register (SPDBSTAS)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SPDBSTAS	Bit	Description	Initial State
Burst Data Length Bit	[31:16]	ES size in bits (Burst Preamble Pd)	0
Bitstream number	[15:13]	Bit_stream_number, shall be set to 0	0
Data Type Dependent Info	[12:8]	Data type dependent information	0
Error Flag	[7]	0 = error flag indicating a valid burst_payload 1 = error flag indicating that the burst payload may contain errors	0
	[6:5]	Reserved	0
Compressed Data Type	[4:0]	0000 = Pause data 0001 = AC-3 0010 = MPEG1 (layer1) 0011 = MPEG1 (layer2, 3), MPEG2-bc 0100 = MPEG2 – extension 0101 = MPEG2 (layer1 – lsf) 0110 = MPEG2 (layer2, layer3 – lsf) others = Reserved	0

SPDIFOUT Channel Status Register (SPDCSTAS)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SPDCSTAS	Bit	Description	Initial State
	[31:30]	Reserved	0
Clock Accuracy	[29:28]	10 = Level I, ±50 ppm 00 = Level II, ±1000 ppm" 01 = Level III, variable pitch shifted	10
Sampling Frequency	[27:24]	0000 = 44.1 kHz 0100 = 48 kHz 1100 = 32 kHz	0
Channel Number	[23:20]	Bit 20 is LSB	0
Source Number	[19:16]	Bit 16 is LSB	0
Category Code	[15:8]	Equipment type CD player = 1000_0000 DAT player = 1100_000L DCC player = 1100_001L Mini disc = 1001_001L (L : information about generation status of the material)	80
Channel Status Mode	[7:6]	00 = Mode 0 others = Reserved	0
Emphasis	[5:3]	000 = emphasis not indicated 100 = emphasis – CD type	0
Copyright Assertion	[2]	0 = copyright 1 = no copyright	0
Audio Sample Word	[1]	0 = linear PCM 1 = non-linear PCM	0
Channel Status Block	[0]	0 = consumer format 1 = professional format	0

SPDIFOUT Data Buffer (SPDDAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SPDDAT	Bit	Description	Initial State
	[31:24]	Reserved	0
SPDIFOUT Data	[23:0]	PCM or stream data	0

SPDIFOUT Repetition Count Register (SPDCNT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SPDCNT	Bit	Description	Initial State
	[31:13]	Reserved	0
Stream Repetition Count	[12:0]	Repetition count according to data type. This bit is valid only for stream data.	0

DATA FORMAT OF SPDIF

Frame Format

A frame is uniquely composed of two sub-frames. The transmission rate of frames corresponds exactly to the source sampling frequency.

In the 2-channel operation mode, the samples taken from both channels are transmitted by time multiplexing in consecutive sub-frames. Sub-frames related to channel 1(left or "A" channel in stereophonic operation and primary channel in monophonic operation) normally use preamble M. However, the preamble is changed to preamble B once every 192 frame. This unit composed of 192 frames defines the block structure used to organize the channel status information. Sub-frames of channel 2(right or "B" in stereophonic operation and secondary channel in monophonic operation) always use preamble W.

In the single channel operation mode in a broadcasting studio environment the frame format is identical to the 2-channel mode. Data is carried only in channel 1. In the sub-frames allocated to channel 2, time slot 28(validity flag) shall be set to logical "1"(not valid).

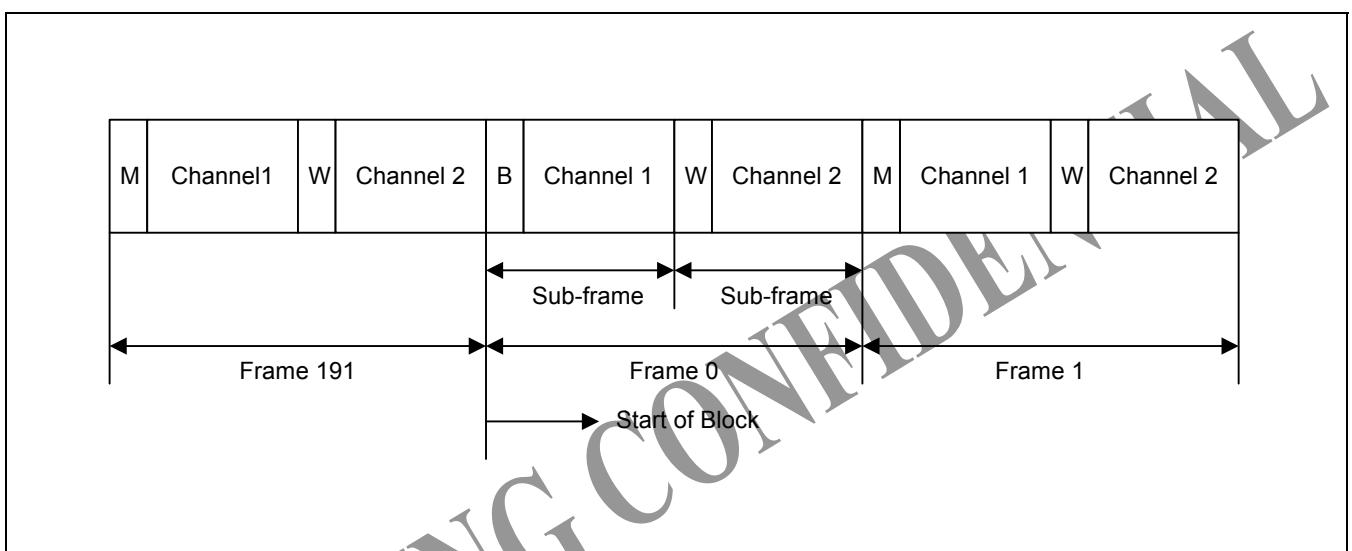


Figure 15-2. SPDIF Frame Format

Sub-frame Format (IEC 60958)

Each sub-frame is divided into 32 time slot, numbered from 0 to 31. Time slot 0 to 3 carry one of the three permitted preambles. These are used to affect synchronization of sub-frames, frames and blocks. Time slots 4 to 27 carry the audio sample word in linear 2's complement representation. The most significant bit is carried by time slot 27. When a 24-bit coding range is used, the least significant bit is in time slot 4. When a 20-bit coding range is sufficient, the least significant bit is in time slot 8 and time slot 4 to 7 may be used for other application. Under these circumstances, the bits in the time slot 4 to 7 are designated auxiliary sample bits.

If the source provides fewer bits than the interface allows (24 or 20), the unused least significant bits shall be set to a logical "0". By this procedure, equipment using different numbers of bits may be connected together. Time slot 28 carries the validity flag associated with the audio sample word. This flag is set to logical "0" if the audio sample is reliable. Time slot 29 carries one bit of the user data associated with the audio channel transmitted in the same sub-frame. The default value of the user bit is logical "0". Time slot 30 carries one bit of the channel status words associated with the audio channel transmitted in the same sub-frame. Time slot 31 carries a parity bit such that time slots 4 to 31 inclusive will carry an even number of ones and an even number of zeros.

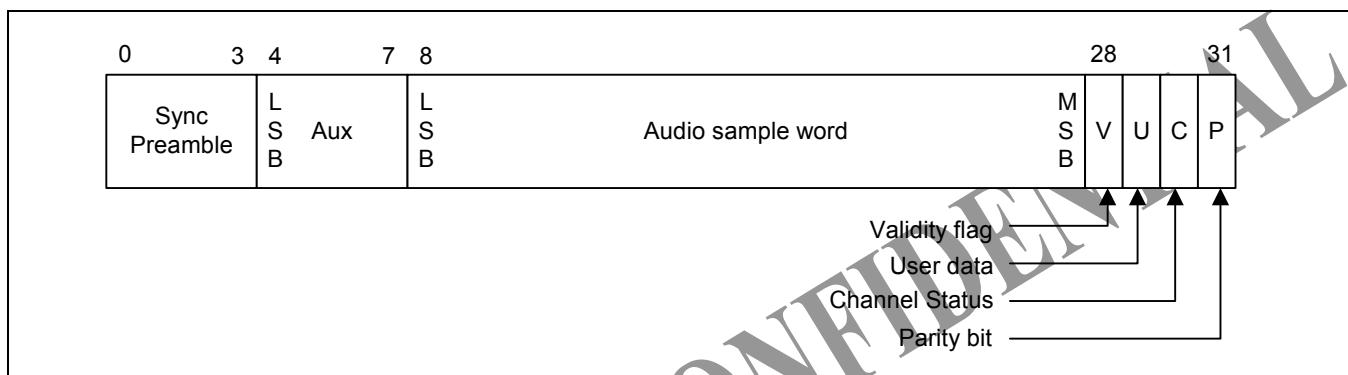


Figure 15-3. SPDIF Sub-frame Format

Channel Coding

To minimize the dc component on the transmission line, to facilitate clock recovery from the data stream and to make the interface insensitive to the polarity of connections, time slots 4 to 31 are encoded in biphase-mark. Each bit to be transmitted is represented by a symbol comprising two consecutive binary states. The first state of a symbol is always different from the second state of the previous symbol. The second state of the symbol is identical to the first if the bit to be transmitted is logical "0", is different from the first if the bit is logical "1".

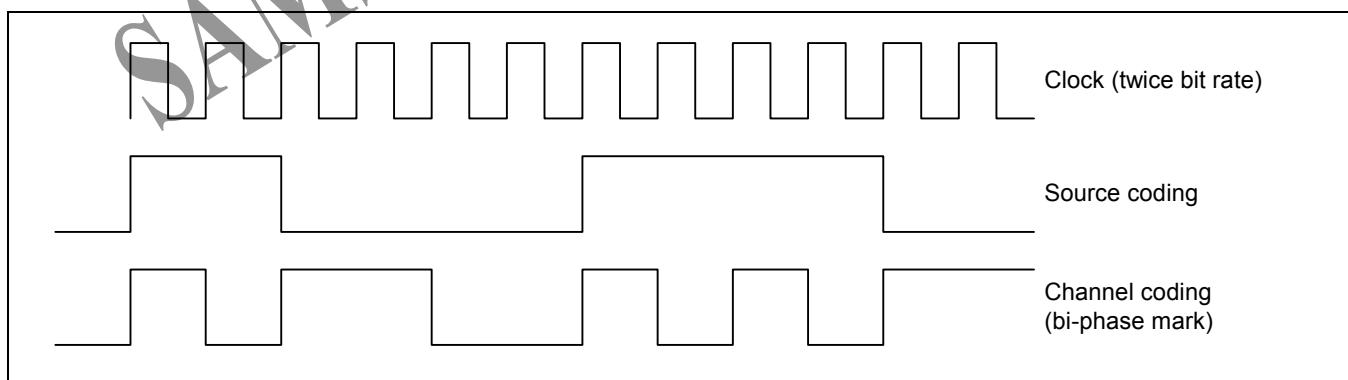


Figure 15-4. Channel Coding

Preamble

Preambles are specific patterns providing synchronization and identification of the sub-frames and blocks. A set of three preambles is used. These preambles are transmitted in the time allocated to four time slots (time slots 0 to 3) and are represented by eight successive states. The first state of the preamble is always different from the second state of the previous symbol.

Like bi-phase code, these preambles are dc free and provide clock recovery. They differ in at least two states from any valid biphasic sequence.

Non-Linear PCM Encoded Source(IEC 61937)

The non-linear PCM encoded audio bitstream is transferred using the basic 16-bit data area of the IEC 60958 subframes, i.e. in time slots 12 to 27. Each IEC 60958 frame can transfer 32 bits of the non-PCM data in consumer application mode.

When the SPDIF bitstream conveys linear PCM audio, the symbol frequency is 64 times the PCM sampling frequency(32 time slots per PCM sample times two channels). When a non-linear PCM encoded audio bitstream is conveyed by the interface, the symbol frequency shall be 64 times the sampling rate of the encoded audio within that bitstream. But in the case where a non-linear PCM encoded audio bitstream is conveyed by the interface containing audio with low sampling frequency, the symbol frequency shall be 128 times the sampling rate of the encoded audio within that bitstream.

Each data burst contains a burst-preamble consisting of four 16-bit words (Pa, Pb, Pc, Pd), followed by the burst-payload which contains data of an encoded audio frame.

The burst-preamble consists of four mandatory fields. Pa and Pb represent a synchronization word; Pc gives information about the type of data and some information/control for the receiver; Pd gives the length of the burst-payload, limited to 216(=65,535) bits.

The four preamble words are contained in two sequential SPDIF frames. The frame beginning the data-burst contains preamble word Pa in subframe 1 and Pb in subframe 2. The next frame contains Pc in subframe 1 and Pd in subframe 2. When placed into a SPDIF subframe, the MSB of a 16-bit burst-preamble is placed into time slot 27 and the LSB is placed into time slot 12.

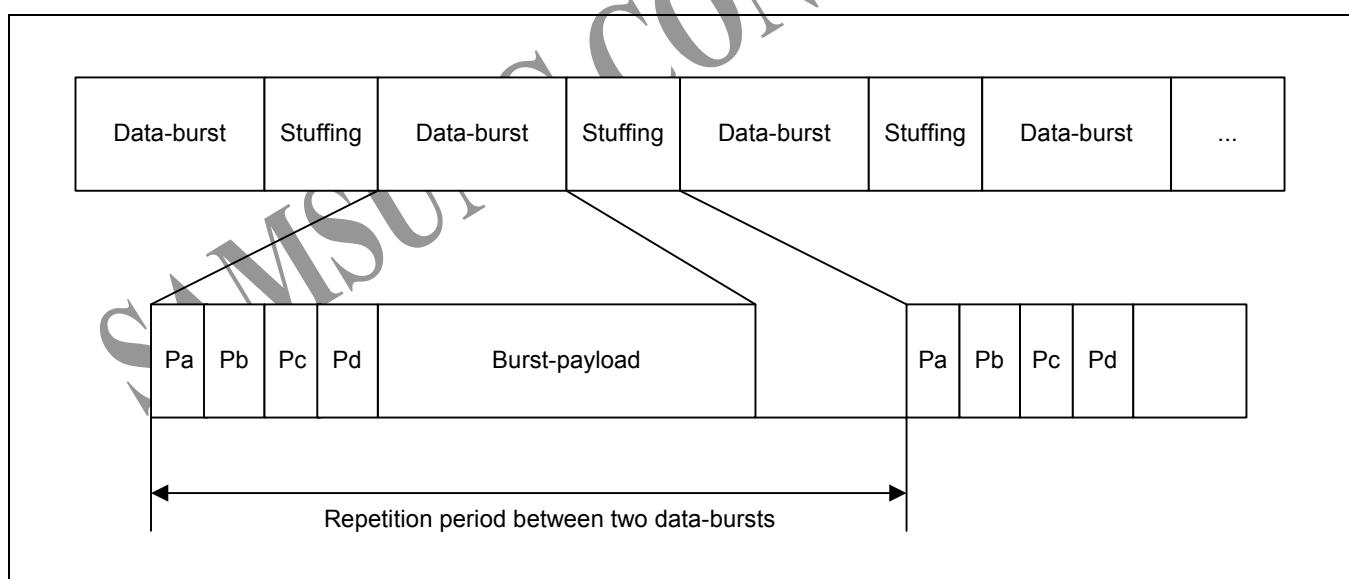


Figure 15-5. Format of Burst Payload

SPDIF OPERATION

Since the bit frequency of SPDIF is 128fs(fs: sampling frequency), the main clock of SPDIF is made by dividing audio main clock(MCLK) depending on the frequency of MCLK. MCLK is divided by 2 in case of 256fs, by 3 in case of 384fs and by 4 in case of 512fs.

SPDIF module in S5L8700X plays the role of transforming audio sample data into the format of SPDIF. To do this, SPDIF module inserts preamble data, channel status data, user data, error check bit and parity bit into the appropriate time slots. Preamble data are fixed in the module and inserted depending on subframe counter. Channel status data are set in the SPDCSTAS register and used by one bit per frame. User data always have zero values.

For non-linear PCM data, burst-preamble which consists of Pa, Pb, Pc and Pd must be inserted before burst-payload and zero is stuffed from the end of burst-payload to the repetition count. Pa(=16'hF872) and Pb(=16'h4E1F) is fixed in the module and Pc and Pd is set in the register SPDBSTAS. To stuff zero, the end of burst-payload is calculated from Pd value and repetition count which depends on data type in the preamble Pc is acquired from register SPDCNT.

Audio data are justified to the LSB. 16-, 20- or 24-bit PCM data and 16-bit stream data are supported. The unoccupied upper bits of 32-bit word are ignored.

Data are fetched through DMA request. When one of two data buffers is empty, DMA service is requested. Audio data stored in the data buffers are transformed into SPDIF format and output to the port. For non-linear PCM data, interrupt is generated after audio data are output up to the value specified in the SPDCNT register. Interrupt makes the registers such as SPDBSTAS and SPDCNT be set to new values when data type of new bitstream is different from the previous one.

16

REED-SOLOMON ECC CODEC

OVERVIEW

This module can perform ECC (Error Correction Control) for support MLC (Multi-Level Cell) NAND.

FEATURES

- Supports AMBA AHB+ (8 Burst Only)
- Use RS Code(472, 464, 8) on GF(512)
- Primitive Polynomial :
 $P(x) = X^9 + X^4 + 1$
- Generator Polynomial :
 $G(x) = X^8 + \alpha^{399}X^7 + \alpha^{207}X^6 + \alpha^{118}X^5 + \alpha^{108}X^4 + \alpha^{125}X^3 + \alpha^{221}X^2 + \alpha^{420}X + \alpha^{28}$
- Supports 4 Error Correction(Not Erasure Correction)
- Supports to start ECC Decoding Without Syndrome Calculation when the On-the-Fly Syndrome Engine is operated.

1. Flash ECC Block Diagram

1) Top Block Diagram

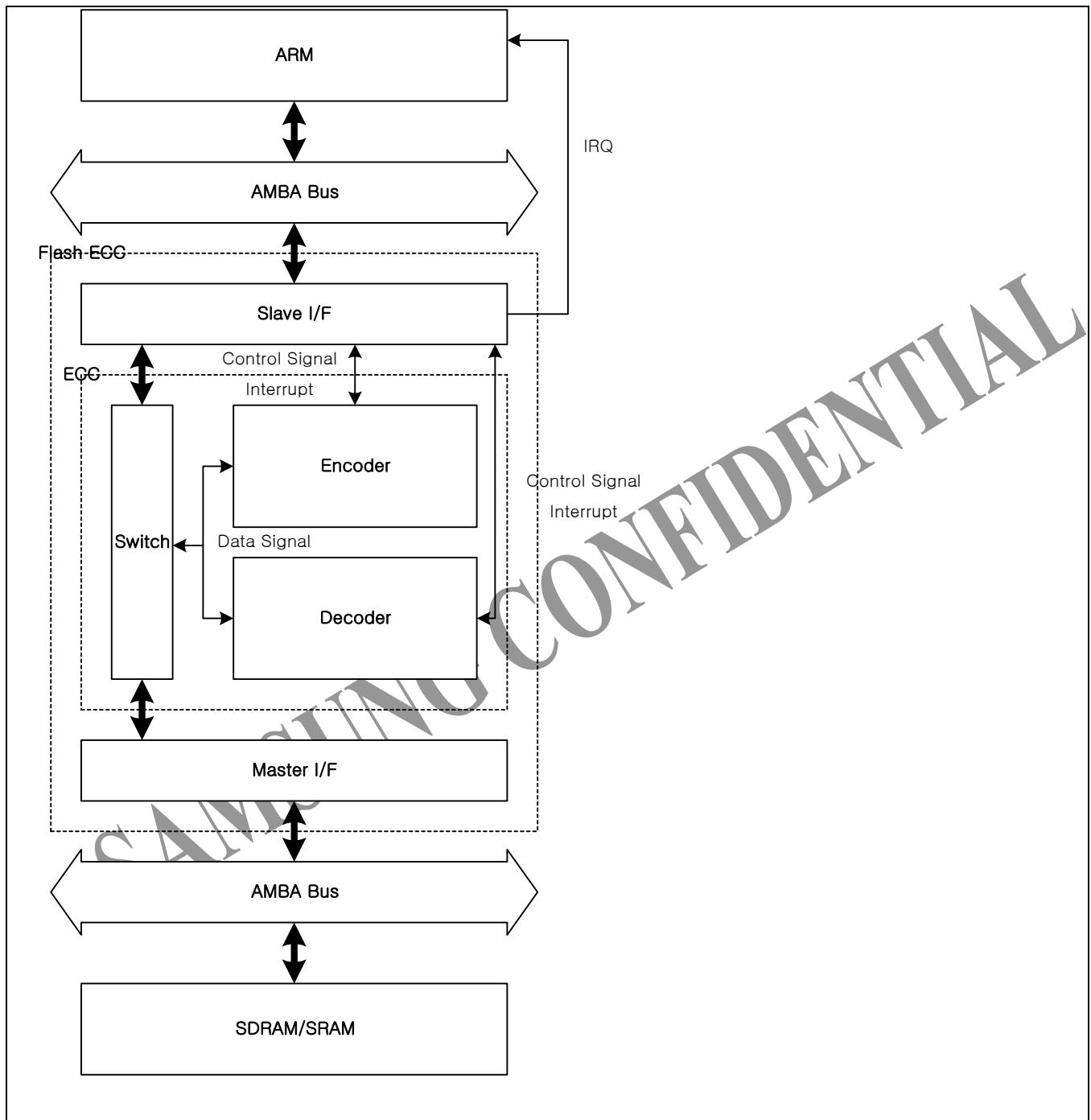


Figure 16-1. Top Block Diagram

2) Encoder Block Diagram

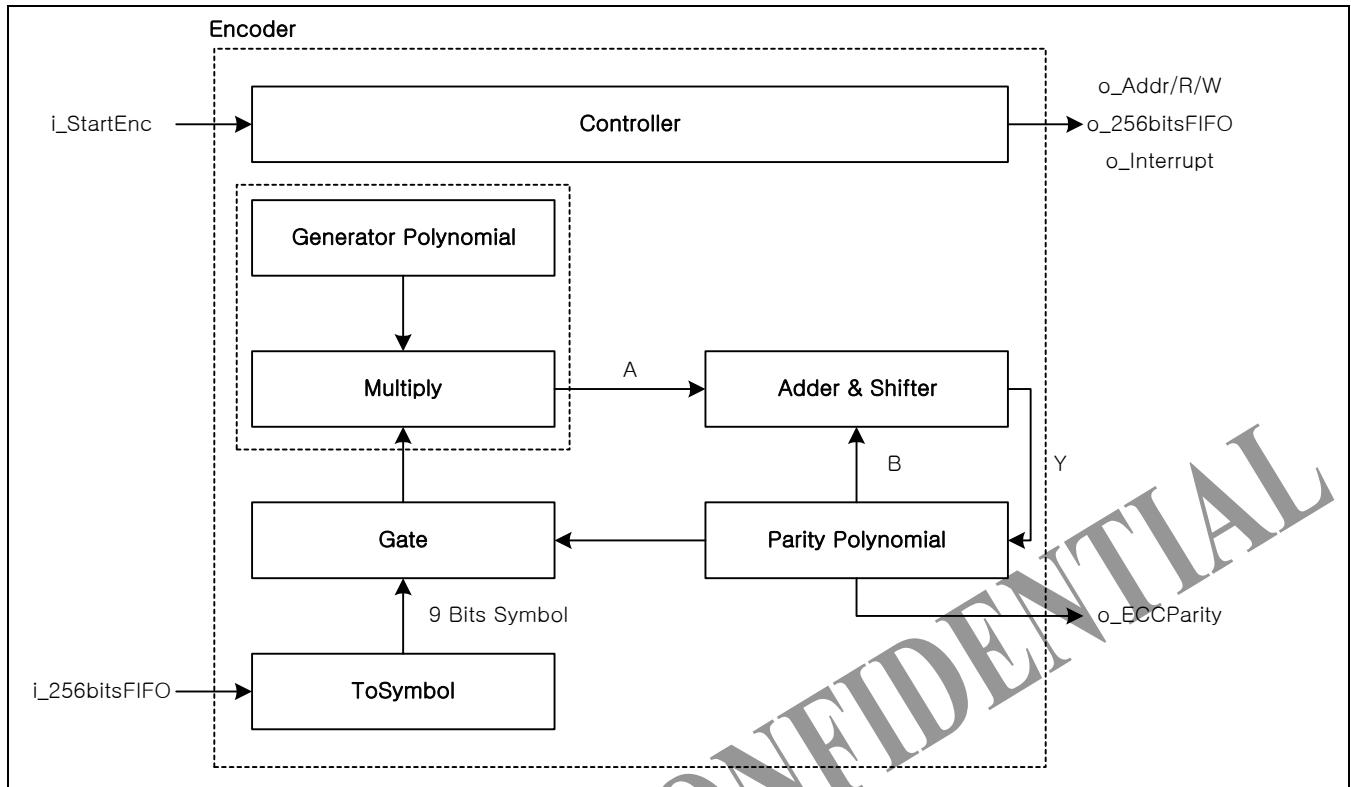


Figure 16-2. Encoder Block Diagram

3) Decoder Block Diagram

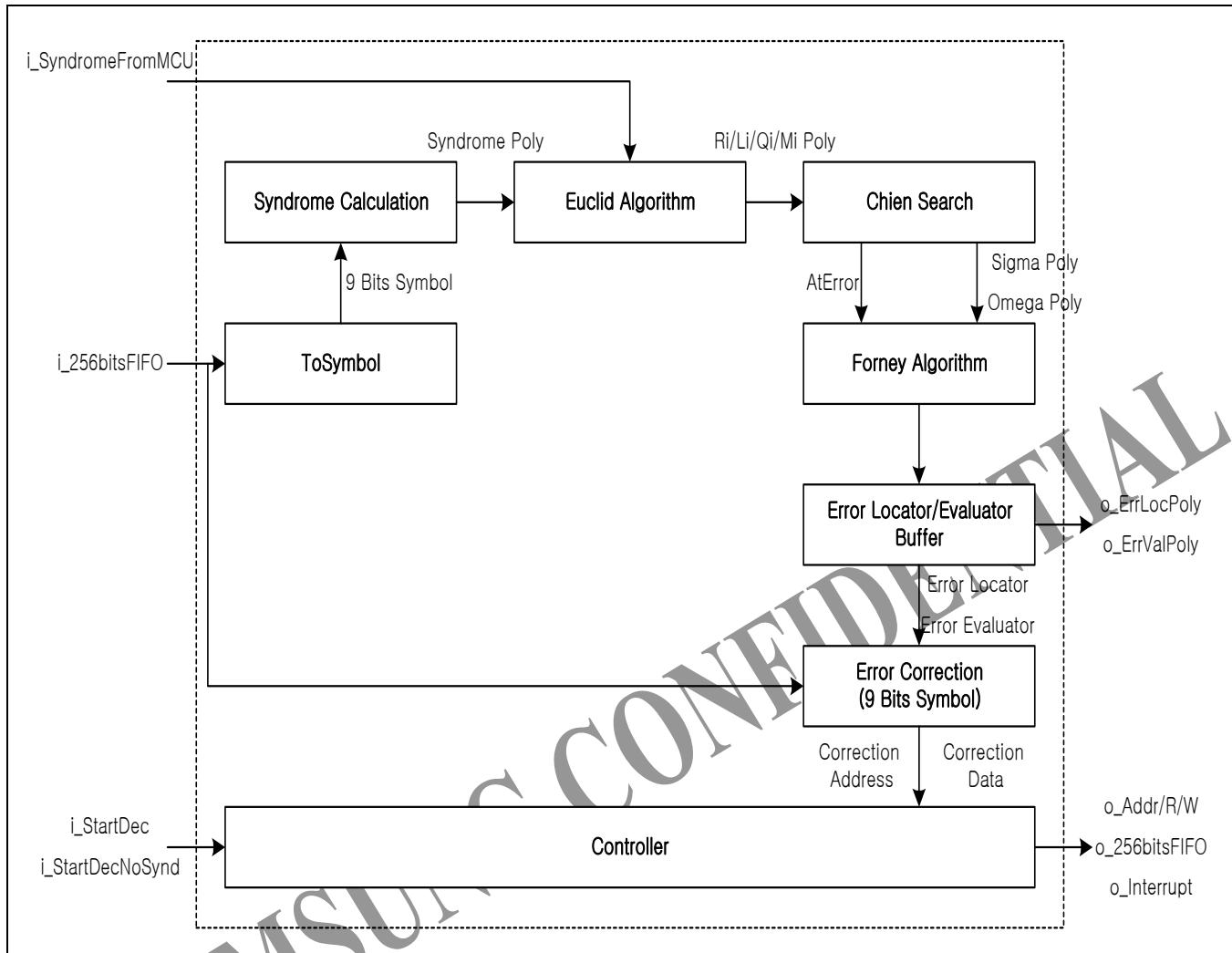


Figure 16-3. Decoder Block Diagram

2. Flash ECC Data Flow

1) On-the-Fly Parity Off

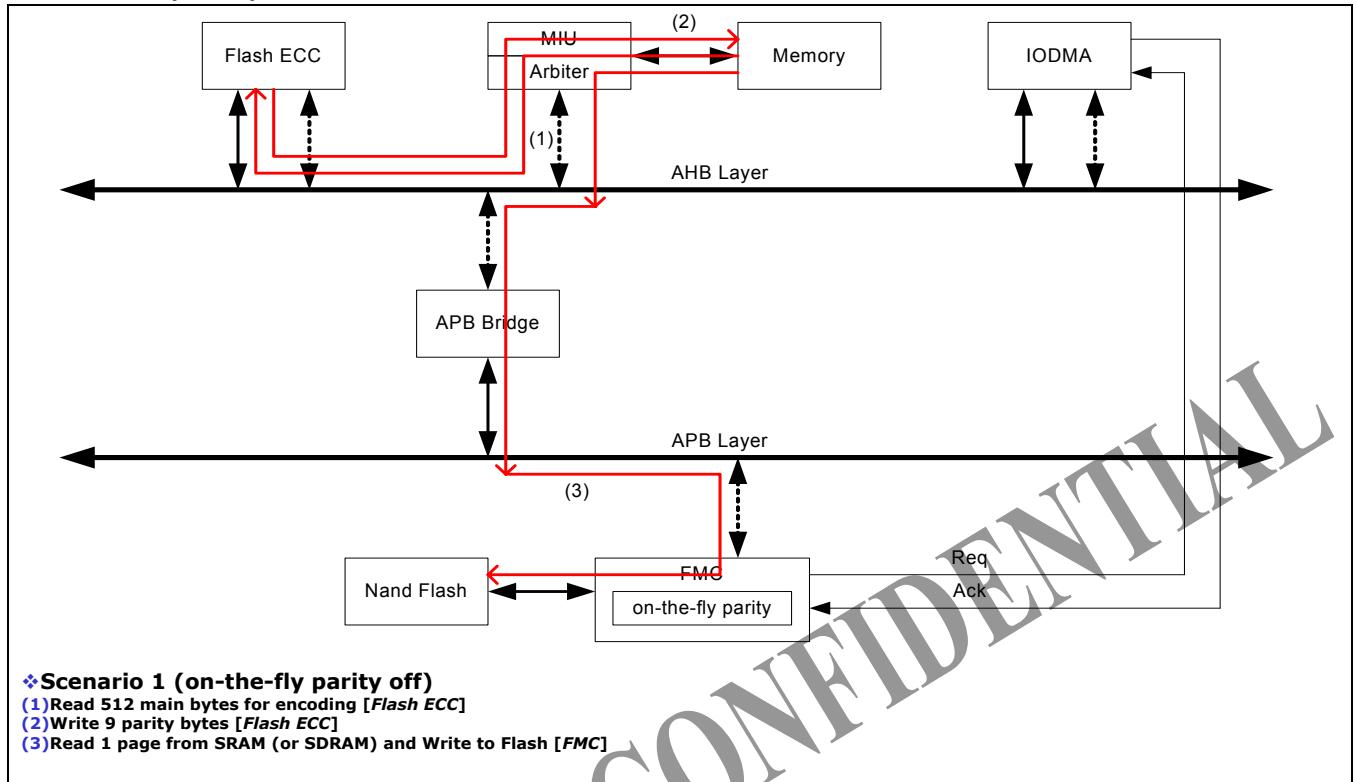


Figure 16-4. On-the-Fly Parity Off

2) On-the-Fly Parity On

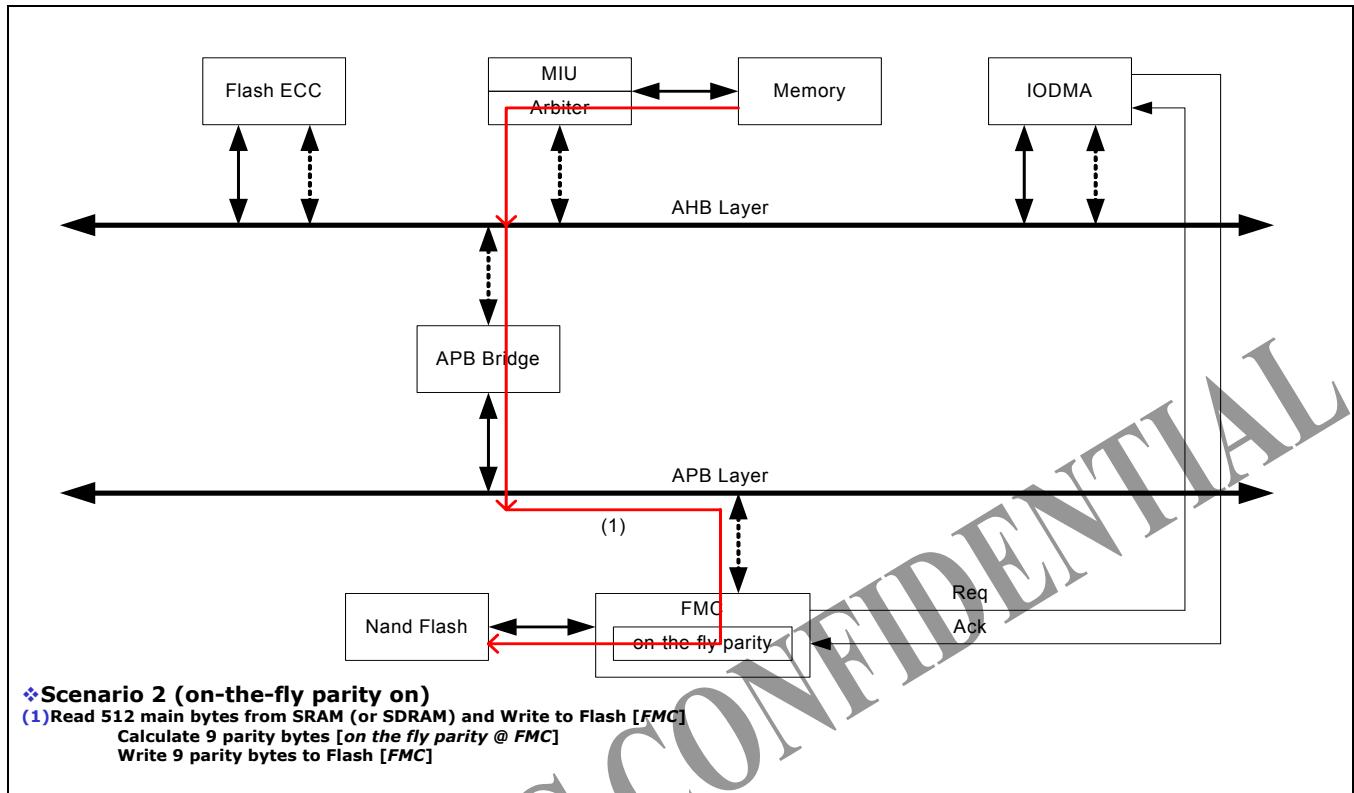


Figure 16-5. On-the-Fly Parity On

3) On-the-Fly Syndrome Off

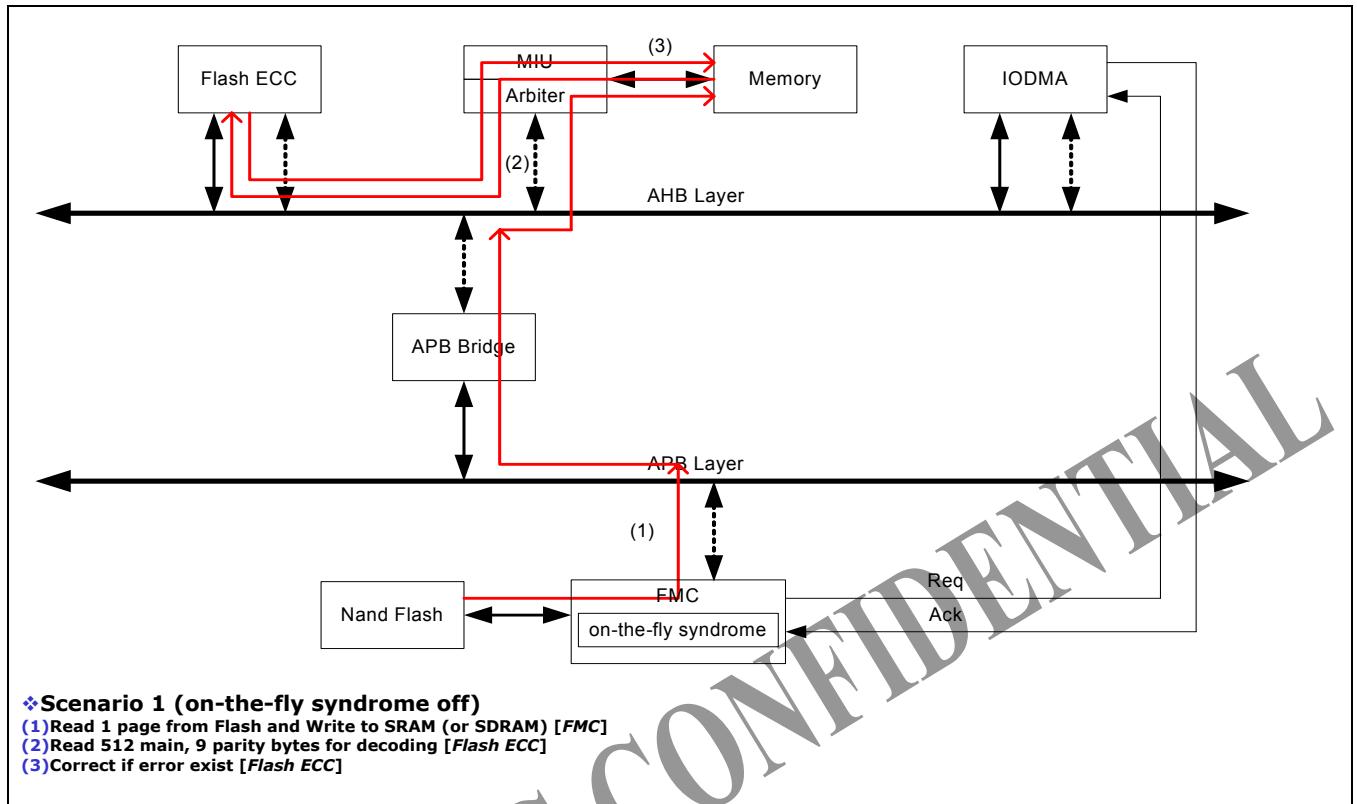
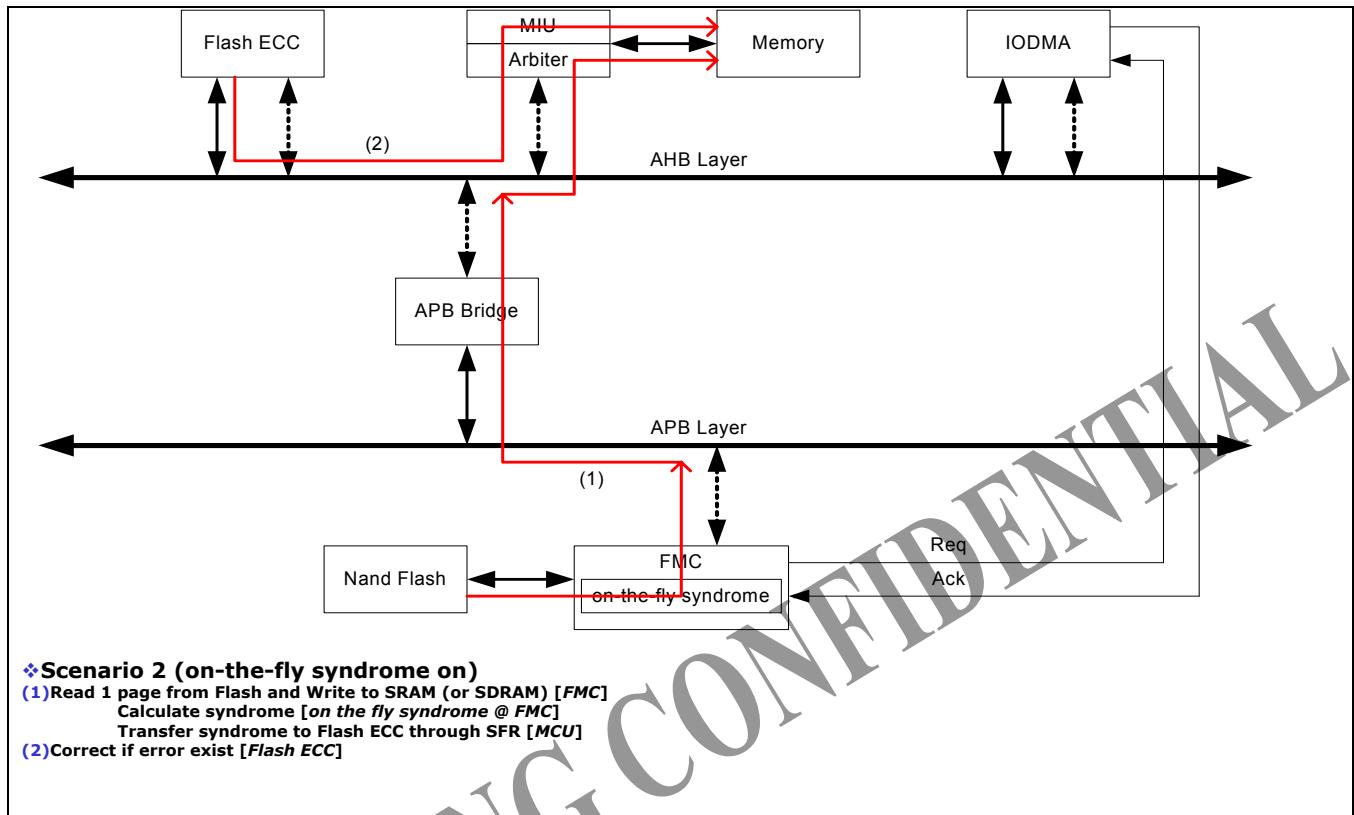


Figure 16-6. On-the-Fly Syndrome Off

4) On-the-Fly Syndrome On



3. Flash ECC Scenario

1) Encoding Scenario(On-the-Fly Parity Off)

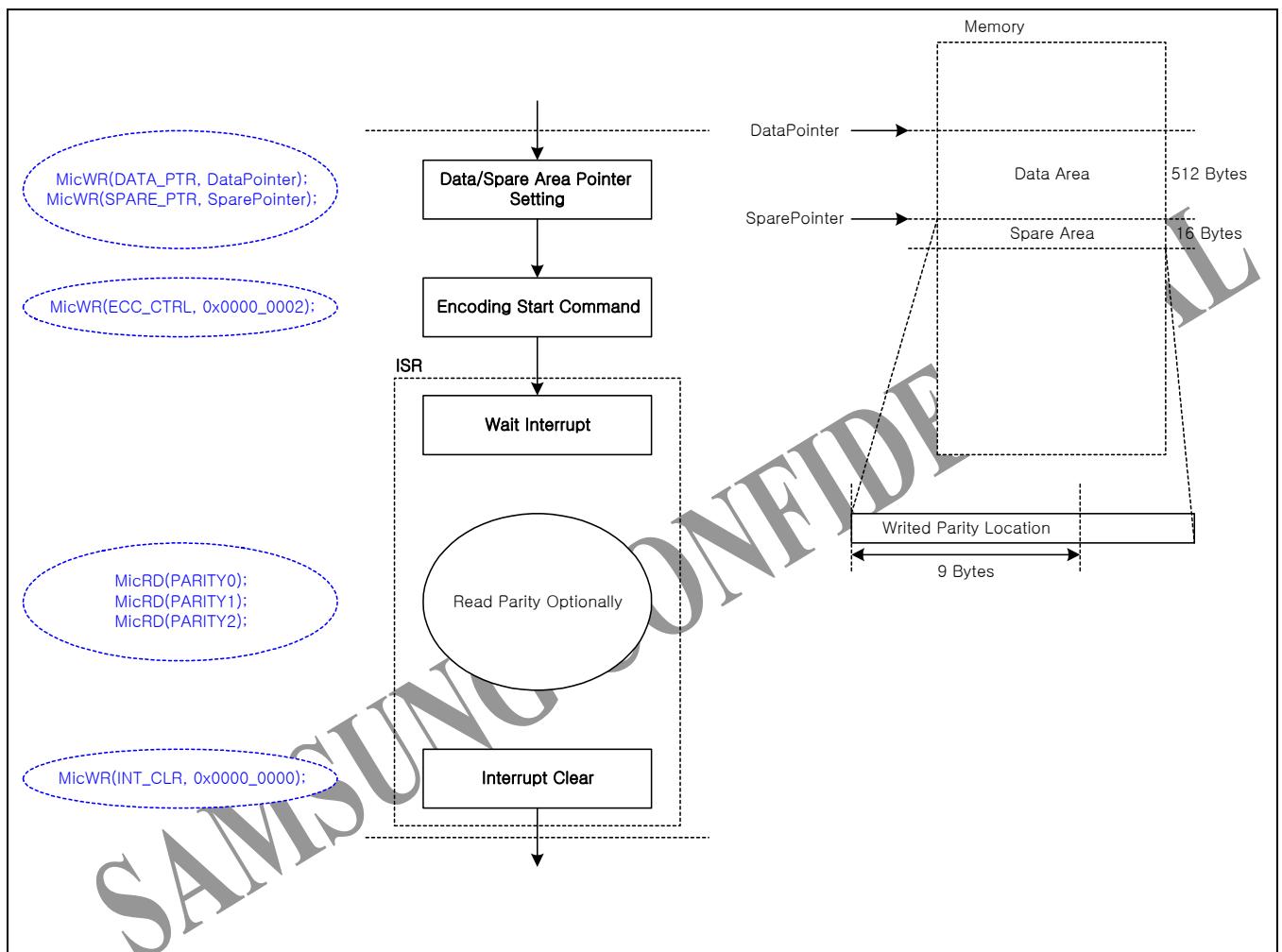


Figure16-8. Encoding Scenario

2) Decoding Scenario(On-the-Fly Syndrome Off)

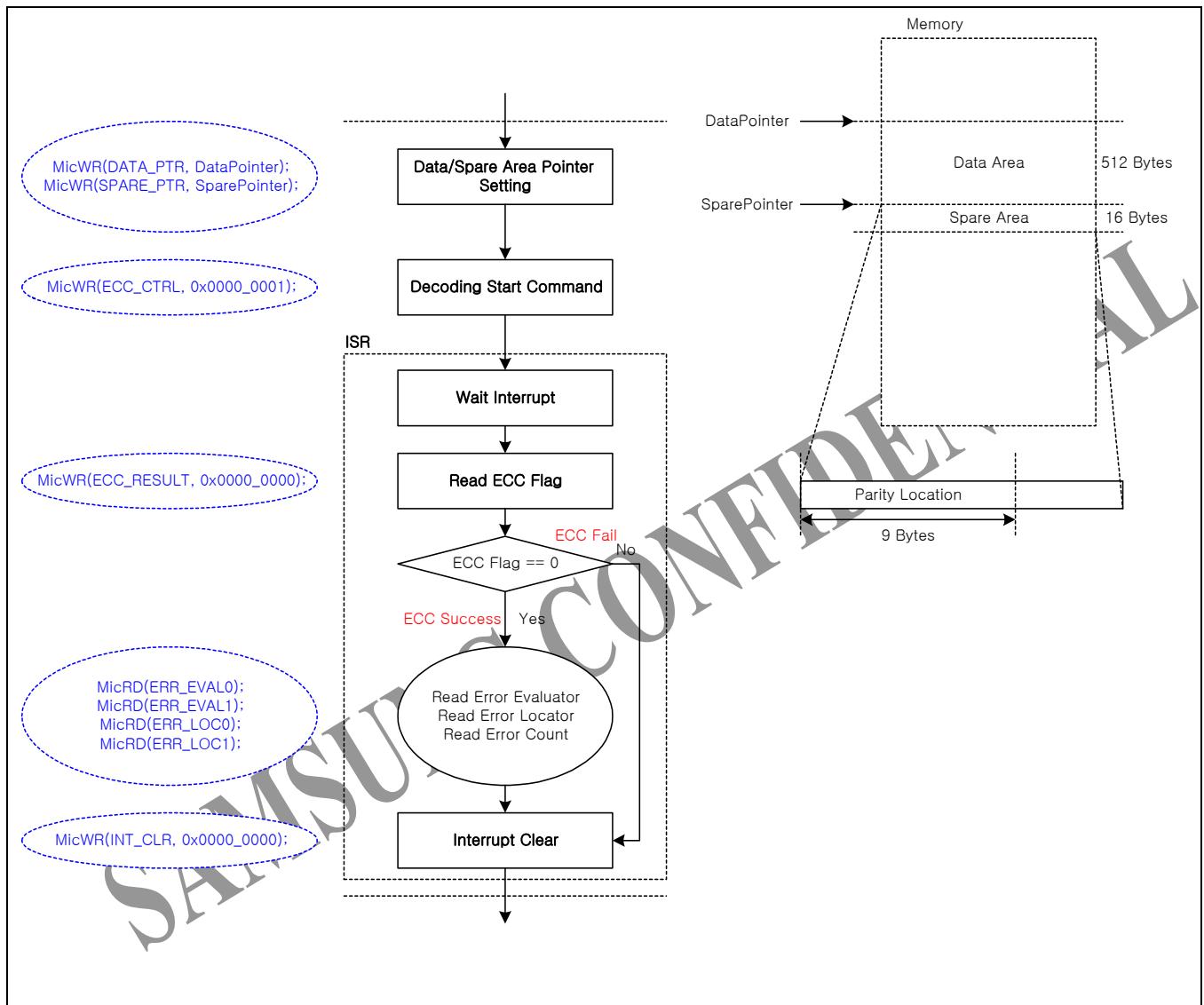


Figure16-9. Decoding Scenario

3) Decoding Scenario for No Syndrome Operation(On-the-Fly Syndrome On)

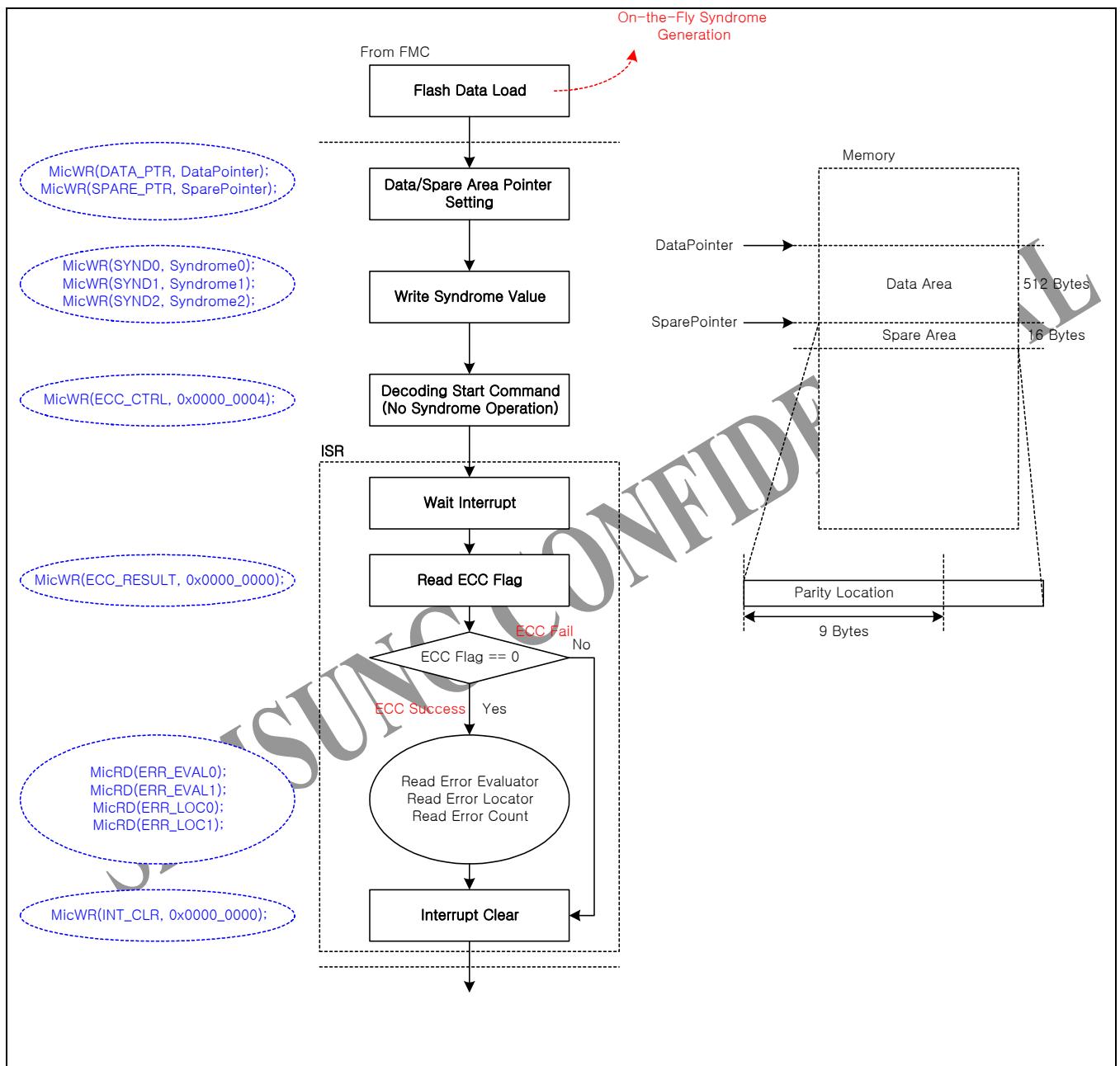


Figure 16-10. Decoding Scenario for No Syndrome Operation

4. MCU Register

REGISTERS

Base address: 0x39E0_0000

Name	Address (virtual)	R/W	Description	Reset
Data Area Start Pointer	Base + 04h	R/W	Data Area Start Pointer	0
Spare Area Start Pointer	Base + 08h	R/W	Spare Area Start Pointer	0
ECC Control Register	Base + 0Ch	W	ECC Control Register	0
ECC Result	Base + 10h	R	ECC Result	0
Error Eval0 Poly	Base + 20h	R	Error Eval0 Poly	0
Error Eval1 Poly	Base + 24h	R	Error Eval1 Poly	0
Error Loc0 Poly	Base + 28h	R	Error Loc0 Poly	0
Error Loc1 Poly	Base + 2Ch	R	Error Loc1 Poly	0
Encode Parity0 Poly	Base + 30h	R	Encode Parity0 Poly	0
Encode Pariyt1 Poly	Base + 34h	R	Encode Pariyt1 Poly	0
Encode Parity2 Poly	Base + 38h	R	Encode Parity2 Poly	0
Interrupt Clear Register	Base + 40h	W	Interrupt Clear Register	0
Syndrom0 Poly	Base + 44h	W	Syndrom0 Poly	0
Syndrom1 Poly	Base + 48h	W	Syndrom1 Poly	0
Syndrom2 Poly	Base + 4Ch	W	Syndrom2 Poly	0

1) Data Area Start Pointer

Name	Addr	R/W	Bit31	...	Bit0	df.
DATA_PTR	04	R/W		Data Area Start Pointer		-

Bit 31~0 : Data Area Start Pointer(Data Unit : 512 Bytes)

2) Spare Area Start Pointer

Name	Addr	R/W	Bit31	...	Bit0	df.
SPARE_PT_R	08	R/W		Spare Area Start Pointer		-

Bit 31~0 : Spare Area Start Pointer(Data Unit : 16 Bytes)

3) ECC Control(Command) Register

Name	Addr	R/W	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	df.
ECC_CTRL	0c	W						StartNoSyndDec	StartEnc	StartDec	-

Bit 2 : Start Decoding Command Without Syndrome Calculation

Bit 1 : Start Encoding Command

Bit 0 : Start Decoding Command

4) ECC Result

Name	Addr	R/W	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	df.
ECC_RESULT	10	R					Number of Error		ECCFlag		-

Bit 3 ~ 1 : Indicate the Number of Error. The Value is meaningless when ECC Flag is set.

Bit 0: ECC Flag. When the Flag is low, the ECC System is successfully operated. This means that the current Codeword didn't have a Error or corrected a Error. But, it is possible to take Wrong Correction for excessive error number and random error. Otherwise, the Flag is high, the ECC System fails Error Correction.

5) Error Evaluation 0 Polynomial

Name	Addr	R/W	Bit31	...	Bit0	df.
ECC_EVAL_0	20	R		ECC Evaluation Polynomial[31: 0]		-

Bit 31 ~ 0 : ECC Evaluation 0 Polynomial. The polynomial is composed of 4 Symbols(9 Bits – 4 Unit).

Evaluator3	Evaluator2	Evaluator1	Evaluator0
Poly1[3:0],Poly0[31:27]	Poly0[26:18]	Poly0[17:9]	Poly0[8:0]

6) Error Evaluation 1 Polynomial

Name	Addr	R/W	Bit31	...	Bit4	Bit3	...	Bit0	df.
ECC_EVAL1	24	R				ECC Evaluation Polynomial[35:32]			-

Bit 3 ~ 0 : ECC Evaluation 1 Polynomial.

7) Error Location 0 Polynomial

Name	Addr	R/W	Bit31	...	Bit0	df.
ECC_LOC0	28	R		ECC Location Polynomial[31: 0]		-

Bit 31 ~ 0 : ECC Location 0 Polynomial. The polynomial is composed of 4 Symbols(9 Bits – 4 Unit).

Locator3	Locator2	Locator1	Locator0
Poly1[3:0],Poly0[31:27]	Poly0[26:18]	Poly0[17:9]	Poly0[8:0]

8) Error Location 1 Polynomial

Name	Addr	R/W	Bit31	...	Bit4	Bit3	...	Bit0	df.
ECC_LOC1	2c	R				ECC Location Polynomial[35:32]			-

Bit 3 ~ 0 : ECC Location 1 Polynomial.

9) Encoding Parity 0 Polynomial

Name	Addr	R/W	Bit31	...	Bit0	df.
PARITY0	30	R		Encoding Parity Polynomial[31: 0]		-

Bit 31 ~ 0 : Encoding Parity 0 Polynomial. The polynomial is composed of 8 Symbols(9 Bits – 8 Unit).

Parity7	Parity6	Parity5	Parity4	Parity3	Parity2	Parity1	Parity0
---------	---------	---------	---------	---------	---------	---------	---------

Poly2[7:0],Poly1[31:]	Poly1[30:22]	Poly1[21:13]	Poly1[12:4]	Poly1[3:0],Poly0[31:27]	Poly0[26:18]	Poly0[17:9]	Poly0[8:0]
-----------------------	--------------	--------------	-------------	-------------------------	--------------	-------------	------------

10) Encoding Parity 1 Polynomial

Name	Addr	R/W	Bit31	...	Bit0	df.
PARITY1	34	R		Encoding Parity Polynomial[63: 32]		-

Bit 31 ~ 0 : Encoding Parity 1 Polynomial

11) Encoding Parity 2 Polynomial

Name	Addr	R/W	Bit31	...	Bit8	Bit7	...	Bit0	df.
PARITY2	38	R					Encoding Parity Polynomial[71:64]		-

Bit 7 ~ 0 : Encoding Parity 2 Polynomial

12) Interrupt Clear Register

Name	Addr	R/W	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	df.
INT_CLR	40	W									-

Any Bit don't care to clear interrupt. Write the Interrupt Clear Register to clear interrupt.

13) Syndrome 0 Polynomial for No Syndrome Operation

Name	Addr	R/W	Bit31	...	Bit0	df.
SYND0	44	W		Syndrome Polynomial[71:40]		-

Bit 31 ~ 0 : Syndrome 0 Polynomial. The polynomial is composed of 8 Symbols(9 Bits – 8 Unit).

Syndrome7	Syndrome6	Syndrome5	Syndrome4	Syndrome3	Syndrome2	Syndrome1	Syndrome0
Poly0[31:23]	Poly0[22:14]	Poly0[13:5]	Poly0[4:0],Poly1[31:28]	Poly1[27:19]	Poly1[18:10]	Poly1[9:1]	Poly1[0],Poly2[7:0]

14) Syndrome 1 Polynomial for No Syndrome Operation

Name	Addr	R/W	Bit31	...	Bit0	df.
SYND1	48	W		Syndrome Polynomial[39:8]		-

Bit 31 ~ 0 : Syndrome 1 Polynomial

15) Syndrome 2 Polynomial for No Syndrome Operation

Name	Addr	R/W	Bit31	...	Bit8	Bit7	...	Bit0	df.
SYND2	4c	W				Syndrome Polynomial[7:0]			-

Bit 7 ~ 0 : Syndrome 2 Polynomial

SAMSUNG CONFIDENTIAL

17

IIS (Tx/Rx) INTERFACE

OVERVIEW

The IIS bus transmits PCM audio data to external DAC and receives PCM audio data from external ADC. To minimize the number of pins required and to keep wiring simple, a 3-line serial bus which consists of a data line for time-multiplexed two-channel data (left/right), a word select line and a clock line is used. In S5L8700X, IIS bus has 6 data lines(one for reception and 5 for transmission), 2 word select lines and 2 clock lines(one for reception and the other for transmission).

IIS has two modes for data transfer. In transmission mode, IIS module makes a request for PCM audio data to IODMA module and IODMA module brings audio data decoded by ADM from SRAM. In reception mode, IIS module gets audio data from external source and stores them into SRAM by requesting DMA to IODMA module. 1 data lines are synchronized with 1 word select line and 1 clock line for transmission mode and 1 data line is synchronized with 1 word select line and 1 clock line for reception mode.

In IIS bus, the chip which generates the bit clock is called master. Either transmitter or receiver of audio data has to generate the bit clock and word select clock as a master since they use the same clock signal for data transfer. S5L8700X acts only as a master in IIS bus, that is, always provides two clock signals (bit clock and word select clock) for a slave even when it receives audio data.

FEATURE

- 2 data transfer modes – transmission, reception
- DMA mode transfer only
- 1 x 24-bit buffer for Transmission and 1 x 24-bit buffer for reception
- 16/20/24 bit data per channel
- 1 channel for transmission and 1 channel for reception
- MSB-first or LSB-first transfer mode
- IIS, left-justified and right-justified format compatible
- Burst transfer mode, which makes Tx buffer filled by burst length in one request.
- Master mode only
- Programmable frequency divider for serial bit clock
- LRCK polarity change at both posedge and negedge of serial bit clock
- 32, 48, 64fs(sampling frequency) serial bit clock per frame (left channel + right channel)
- 256, 384, 512fs master clock (DAC clock)

BLOCK DIAGRAM

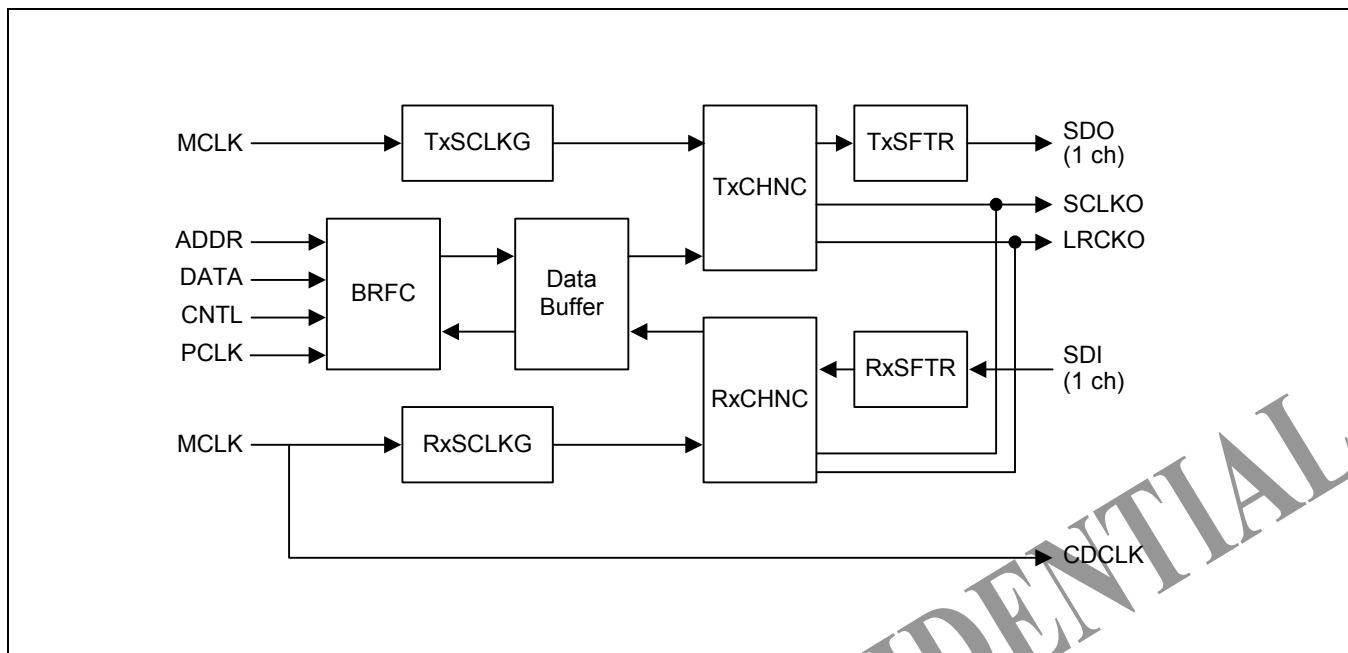


Figure 17-1. IIS Block Diagram

- BRFC : register bank, APB interface, finite state machine for DMA request
- SCLK : generation of serial bit clock for Tx and Rx, respectively
- DATA BUFFER : 24-bit data buffer for Tx and Rx
- TxCHNC, RxCHNC : generation of control signals which connect data buffer with shift register, data alignment depending on various data transfer mode
- TxSFTR, RxSFTR : shift register which transfers parallel data serially

REGISTERS

Name	Width	Address	R/W	Description	Reset
I2SCLKCON	32	0x3CA0_0000	R/W	Clock Control Register	0x0000 0002
I2STXCON	32	0x3CA0_0004	R/W	Tx configuration Register	0x0000 0000
I2STXCOM	32	0x3CA0_0008	R/W	Tx command Register	0x0000 0000
I2STXDB0	32	0x3CA0_0010	W	Tx data buffer	0x0000 0000
I2SRXCON	32	0x3CA0_0030	R/W	Rx configuration Register	0x0000 0000
I2SRXCOM	32	0x3CA0_0034	R/W	Rx command Register	0x0000 0000
I2SRXDB	32	0x3CA0_0038	R	Rx data buffer	0x0000 0000
I2SSTATUS	32	0x3CA0_003C	R	status register	0x0000 000D

I2S Clock Control Register (I2SCLKCON)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

I2SCLKCON	Bit	Description	Initial State
	[31:2]	Reserved	0
IIS clock down ready (read only)	[1]	0 = clock-down not ready 1 = clock-down ready	1
IIS power on	[0]	0 = power off 1 = power on	0

I2S Tx Configuration Register (I2STXCON)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

I2STXCON	Bit	Description	Initial State
	[31:19]	Reserved	0
Burst mode	[18:16]	Burst length(BL) selection Burst Length = (BL[2:0] + 1)	0
LRCK polarity change	[15]	0 = SCLK falling edge 1 = SCLK rising edge	0
Audio interface format	[14:13]	0x = IIS (basic format) 10 = Left justified 11 = Right justified	00
MSB first or LSB firstIn Serial Interface	[12]	0 = MSB first (Normal audio interface mode) 1 = LSB first	0
Left/Right channel polarity (LRCK)	[11]	0 = Left Channel for Low polarity 1 = Left Channel for High polarity	0
4-bit scaler for SCLK generation	[10:8]	SCLK = MCLK / {(3-bit value +1) * 2}	000
	[7]	Reserved (always recognized as zero)	0
Serial Data Bit per Channel	[6:5]	00 = 16 bit 10 = 24 bit	00
		01 = 20 bit 11 = N/A	
Bit Clock per Frame (Frame = Left + Right)	[4:3]	00 = 32 fs 10 = 64 fs	00
		01 = 48 fs 11 = N/A	
Channel index	[2:0]	Channel index=number of channels / 2 (<=5)	0

I2S TX Command Register (I2STXCOM)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

I2STXCOM	Bit	Description	Initial State
	[31:4]	Reserved	0
Tx enable select2	[3]	0 = No transfer 1 = Transmit Mode On	0
I2S interface enable	[2]	0 = I2S interface disable (stop) 1 = I2S interface enable (start)	0
DMA service request enable	[1]	0 = DMA request disable 1 = DMA request enable	0
Channel Idle Command	[0]	0 = Channel not idle (LRCK On) 1 = Channel idle (LRCK Off)	0

I2S Data Buffer Register (I2STXDB)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

I2STXDB	Bit	Description	Initial State
I2S Transmit Data	[31:0]	Transmit Data to DAC	0

I2S RX Configuration Register (I2SRXCON)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

I2SRXCON	Bit	Description	Initial State
	[31:13]	Reserved	0
LRCK polarity change	[12]	0 = SCLK falling edge 1 = SCLK rising edge	0
Audio interface format	[11:10]	0x = IIS (basic format) 10 = Left justified 11 = Right justified	0
MSB first or LSB first in Serial Interface	[9]	0 = MSB first (Normal audio interface mode) 1 = LSB first	0
Left/Right channel polarity	[8]	0 = Left Channel for Low polarity 1 = Left Channel for High polarity	0
4-bit scaler for SCLK generation	[7:5]	SCLK = MCLK / {(3-bit value+1) * 2}	0
	[4]	Reserved (always recognized as zero)	0
Serial Data Bit per Channel	[3:2]	00 = 16 bit 10 = 24 bit 11 = N/A	0
Bit Clock per Frame (Frame = Left + Right)	[1:0]	00 = 32 fs 10 = 64 fs 01 = 48 fs 11 = N/A	0

I2S Rx Command Register (I2SRXCOM)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

I2SRXCOM	Bit	Description	Initial State
	[31:4]	Reserved	0
Rx enable select	[3]	0 = No transfer 1 = Receive Mode On	0
I2S interface enable	[2]	0 = I2S interface disable (stop) 1 = I2S interface enable (start)	0
DMA service request enable	[1]	0 = DMA Request disable 1 = DMA Request enable	0
Channel Idle Command	[0]	0 = Channel not idle (LRCK On) 1 = Channel idle (LRCK Off)	0

I2S Rx Data Buffer Register (I2SRXDB)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															

I2SRXDB	Bit	Description	Initial State
I2S Receive Data	[31:0]	Receive Data from ADC	0

I2S Status Register (I2SSTATUS)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

I2SSTATUS	Bit	Description		Initial State
	[31:4]	Reserved		0
RXDBEMP	[3]	Rx data buffer state flag 0 = not empty 1 = empty		1
RXLRIDX	[2]	Rx Left/Right channel index 0 = left channel 1 = right channel		1
TXDBFUL	[1]	Tx data buffer state flag 0 = not full 1 = full		0
TXLRIDX	[0]	Tx Left/Right channel index 0 = left channel 1 = right channel		1

IIS OPERATIONS

IIS Clock Frequency

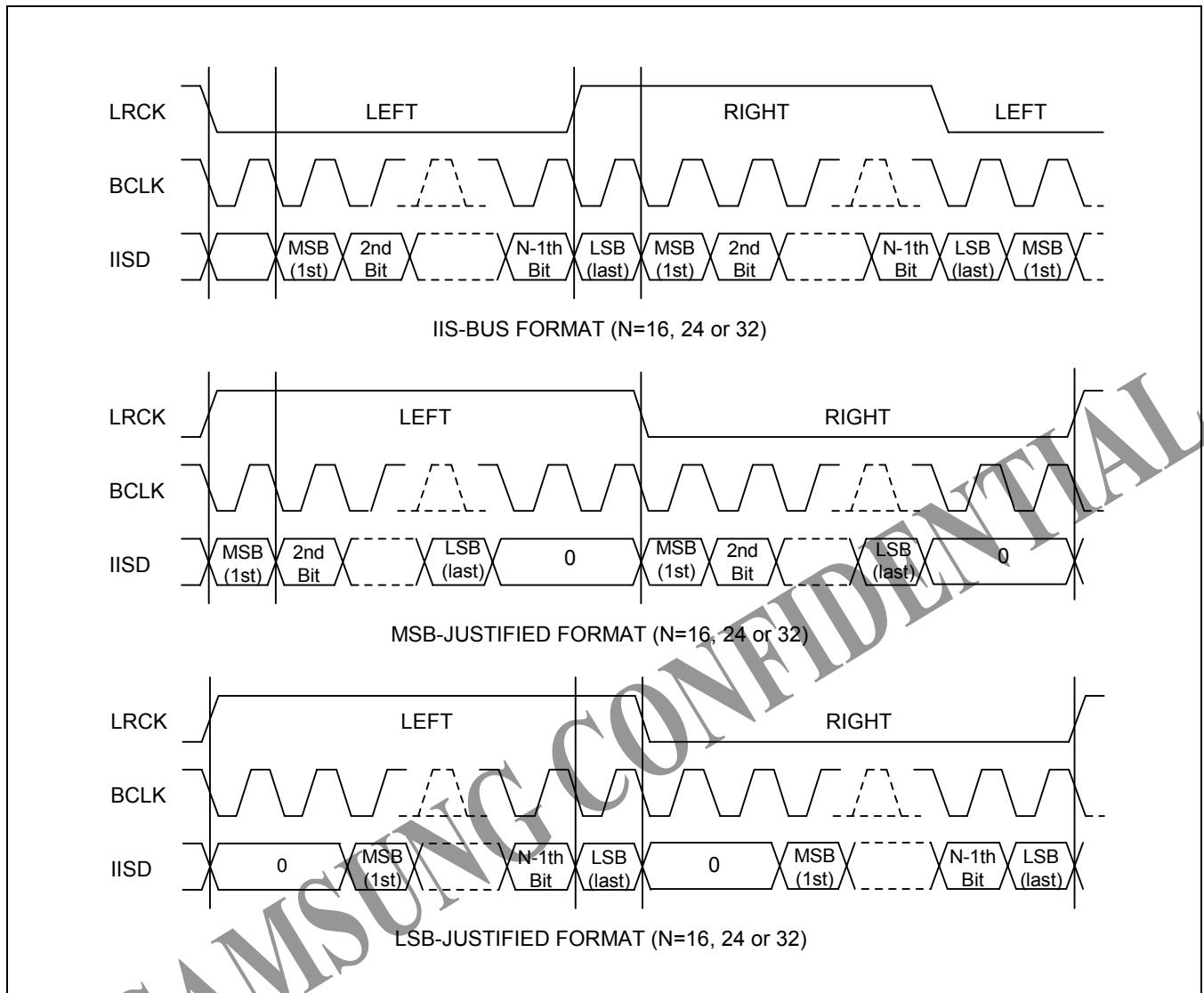
The main clock of IIS(PCLK) is the same as system clock's frequency. Audio main clock(MCLK) is made by PLL2 and is determined considering sampling frequency of audio data. The relationship between sampling frequency(fs) and audio main clock is shown in Table 17-1. Serial bit clock(SCLK) is determined depending on MCLK and data bit per channel(Table 17-2) and is set by the value of configuration register(I2STXCON, I2SRXCON). Word select signal(LRCK) has the same frequency as sampling frequency(fs).

Table 17-1. The Frequency of Audio Main Clock

LRCK (fs)	8.000 KHz	11.03 KHz	16.00 KHz	22.05 KHz	32.00 KHz	44.10 KHz	48.00 KHz	64.00 KHz	88.20 KHz	96.0 KHz
256fs										
MCLK (MHz)	2.048	2.822	4.096	5.645	8.192	11.29	12.29	16.38	22.58	24.58
384fs										
	3.072	4.234	6.144	8.467	12.29	16.93	18.43	24.58	33.87	36.86
512fs										
	4.096	5.645	8.192	11.29	16.38	22.58	24.58	32.77	45.16	49.15

Table 17-2. The Frequency of Serial Bit Clock

Serial Bit Per Channel	16-Bit	20-Bit	24-Bit
Serial Clock Frequency (BCLK)			
@CODECLK=256fs	32fs, 64fs	32fs, 64fs	32fs, 64fs
@CODECLK=384fs	32fs, 48fs	32fs, 48fs	32fs, 48fs
@CODECLK=512fs	32fs, 64fs	32fs, 64fs	32fs, 64fs

AUDIO INTERFACE FORMAT**Figure 17-2. Audio Data Interface Format**

In S5L8700X, IIS module is applicable to various interface format as shown above. IIS-Bus format starts data transfer at the next BCLK clock after word select signal(LRCK) changes the polarity. But MSB-Justified format transfers data from the very clock that LRCK changes the polarity. LSB-justified format completes one channel transfer at the same time as the change of LRCK. If there exists more serial clock bits in one channel than serial data bits, the remaining clock bits are stuffed with zero's in each case.

Depending on register values, MSB of audio data or LSB can be transferred first and data transfer can be synchronized with the rising edge or falling edge of LRCK.

Start and Stop Condition

To make IIS module active, I2S_power_on bit in I2SCLKCON must be set to '1'. After IIS module becomes active, Setting command register to '0x0000_000E' drives IIS to its function mode. IIS_interface_enable bit in command register decides the generation of serial bit clock(SCLK) and makes IIS bus stop immediately after it is set to '0'. IIS_channel_idle_command bit decides the generation of word select signal(LRCK) and makes IIS transfer one pair datum after it is set to '0' and then makes it stop. Therefore, IIS_interface_enable bit can be used to refresh current SDRAM data and IIS_channel_idle_command bit be used to maintain current SDRAM data.

To make IIS inactive, the procedure is as follows.

1. Wait LRCK 1cycle period.
2. Set I2S_power_on to '0'.
3. Wait until I2S_power_down_ready becomes set to '1'.

DMA Data Transfer

IIS module gives audio data to MEMORY or gets from MEMORY through IODMA module. Therefore, IIS module sends the request signal to IODMA module for audio data and receives acknowledgement signal after the completion of data transfer. Both request and acknowledgement signal have 2 bits, one for transmission and the other for reception.

For transmission, IIS module sends request signal when Tx data buffers are not occupied and receives acknowledgement signal after it receives audio data by burst length. Since IIS module acquires audio data by burst length per request and acknowledgement counter also increases by burst length per request, burst mode bit in I2STXCON must be set with the same burst length as in IODMA module to increase acknowledgement counter correctly. If acknowledgement counter is less than the number of channels, that is, data buffer is not full after one request, IIS module sends request signal until data_buffer_full flag becomes set to '1'. After data_buffer_full flag is set to '1', internal signal which indicates the start of next channel transfers audio data in buffers into Tx shift registers and resets data_buffer_full flag to '0'.

For reception, the data buffer is empty at first(buffer_empty flag = '1'). The audio data are received into Rx shift register and is transferred to data buffer with the internal channel_start signal. After the buffer_empty flag becomes set to '0', IIS module sends request signal for Rx to IODMA and then receives acknowledgement signal after audio data in data buffer is read. Data reception is independent of data transmission and therefore the two modes can function simultaneously.

Program Guide of Tx Mode

In case of TURN-ON,

- S1. DMA channel 0 set (IIS Tx channel)
DMABASE0, DMACON0, DMATCNT0 registers set
- S2. IIS Tx mode set
I2STXCON register set
- S3. DMA channel 0 on
DMACOM0 register set (0x0000_0004)
- S4. IIS Tx clock on
I2SCLKCON register set (0x0000_0001)
- S5. IIS Tx on
I2STXCOM register set(0x0000_000E)

In case of TURN-OFF,

- E1. DMA channel 0 off
DMACOM0 register set (0x0000_0005)
- E2. Some time wait
LRCK half cycle wait
- E3. IIS Tx off
I2STXCOM register set(0x0000_000A)

Program guide of Rx mode

IIS module is consist of IIS RX part and TX part. Each part has IIS interface signals (LRCK, BCK, ADATA). And we support only master mode of IIS Rx. So S5L8700X share two pin LRCK, BCK.

As this restriction, there is some problem of receiving wrong data.

This is a program guide to use Rx mode.

In case of TURN-ON,

- S1. DMA channel 0 set (IIS Tx channel)
DMABASE0, DMACON0, DMATCNT0 registers set
- S2. DMA channel 2 set (IIS Rx channel)
DMABASE0, DMACON0, DMATCNT0 registers set
- S3. IIS Rx mode set
I2SRXCON register set
- S4. IIS Tx mode set
I2STXCON register set
- S5. DMA channel 2 on
DMACOM2 register set (0x0000_0004)
- S6. DMA channel 0 on
DMACOM0 register set (0x0000_0004)
- S7. IIS clock on
I2SCLKCON register set (0x0000_0001)
- S8. IIS Rx on but DMA request off
I2SRXCOM register set(0x0000_000C)

- S9. IIS Tx on
I2STXCOM register set(0x0000_000E)
- S10. Wait BCLK 1cycle
100 ~200 us wait
- S11. Wait IIS status (LRCK == 1)
while((I2SSTATUS &1));
- S12. Wait IIS status (LRCK == 0)
while(!(I2SSTATUS &1));
- S13. IIS Rx on
I2SRXCOM register set(0x0000_000E)

In case of TURN-OFF,
E1. DMA channel 2 off

DMACOM2 register set (0x0000_0005)

E2. DMA channel 0 off

DMACOM0 register set (0x0000_0005)

E3. Some time wait

LRCK half cycle wait

E4. IIS Rx off

I2SRXCOM register set(0x0000_000A)

E5. IIS Tx off

I2STXCOM register set(0x0000_000A)

NOTES

SAMSUNG CONFIDENTIAL

18 IIC BUS INTERFACE

OVERVIEW

S5L8700X has 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 that are connected to the IIC-bus. The SDA and SCL lines are bi-directional.

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 line initiates a start condition while SCL line remains at high level. A low-to-high transition of SDA line with SCL line high generates a stop condition with SCL line high.

The start and stop condition should always be generated by the master devices. A 7-bit address value in the first data transfer, which is put onto the bus after the start condition, determines the slave device that is addressed by the 7-bit address. The 8th bit in the first data transfer determines the direction of the transfer (read or write).

Every data onto the SDA line should be eight bits or one byte. The number of bytes that can be transferred during the bus transfer is unlimited. Data is always sent from the MSB bit and acknowledge (ACK) bit should be asserted after one byte transfer (NOTE).

NOTE: This is dependent on the devices. Some other devices, for example EEPROM, need not to generate ACK signals.

FEATURE

- Four operation mode
 1. Master transmitter mode
 2. Master receive mode
 3. Slave transmitter mode
 4. Slave receive mode
- Configurable slave address
- Operation pending until the interrupt pending flag is cleared by the software

BLOCK DIAGRAM

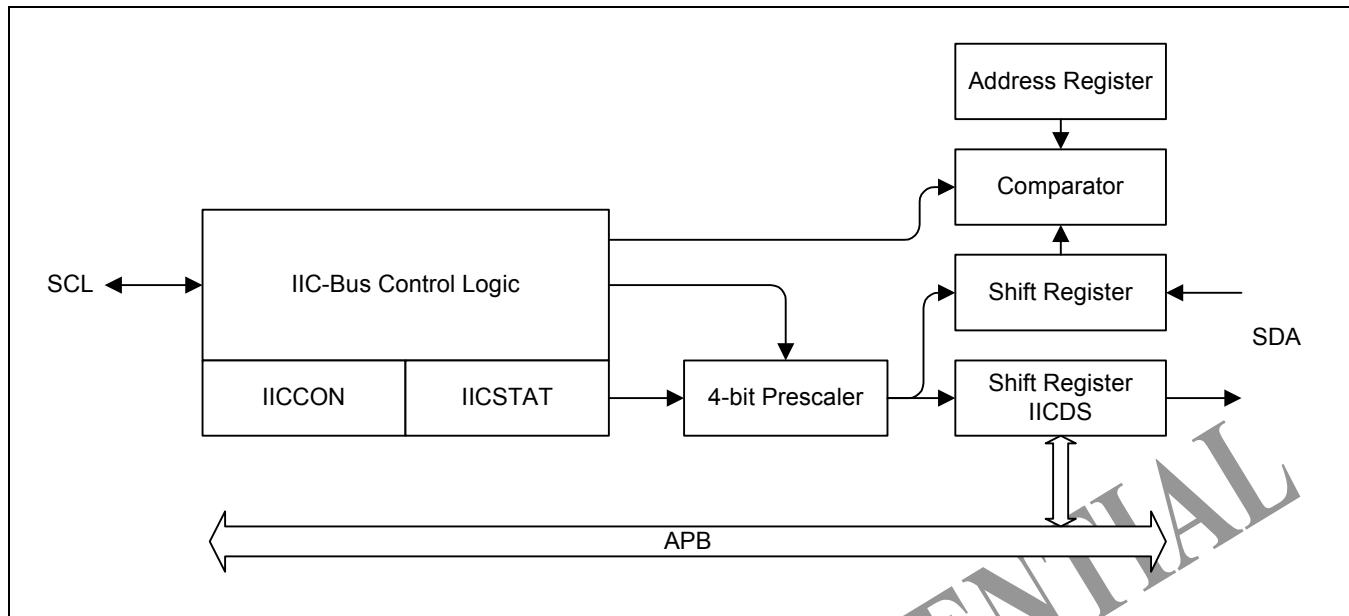


Figure 18-1. IIC Block Diagram

REGISTERS

Name	Width	Address	R/W	Description	Reset
IICC CON	32	0x3C90_0000	R/W	Control Register	0x0000 000x
IICSTAT	32	0x3C90_0004	R/W	Control/Status Register	0x0000 0000
IICADD	32	0x3C90_0008	R/W	Bus Address Register	-
IICDS	32	0x3C90_000C	R/W	Transmit/Receive Data Shift Register	-

Multi-Master IIC-Bus Control Register (IICCON)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits	Name	Type	Description	Reset
7	Acknowledge Generation (ACK_GEN)	R/W	IIC-bus acknowledge enable bit 0 = Disable 1 = Enable In Tx mode, the IICSDA is free in the ack time. In Rx mode, the IICSDA is low in the ack time.	0
6	Tx Clock Source Selection (CKSEL)	R/W	Source clock of IIC-bus transmit clock pre-scaler selection bit 0 = IICCLK = PCLK / 16 1 = IICCLK = PCLK / 512	0
5	Tx/Rx Interrupt (INT_EN)	R/W	IIC-bus Tx/Rx interrupt enable/disable bit 0 = Disable 1 = Enable	0
4	Interrupt Pending Flag (NOTE) (IRQ)	R/W	IIC-bus Tx/Rx interrupt pending flag. Read Operation 0 = No interrupt pending 1 = Interrupt is pending. In this condition, the IICSCL is tied to low and the IIC is stopped. Write Operation 0 = Nothing occurs 1 = Clear the pending condition and resume the operation	0
3:0	Transmit Clock Value (CK_REG)	R/W	IIC-bus transmit clock prescaler IIC-bus transmit clock frequency is determined by this 4-bit prescaler value, according to the following formula: $\text{Tx clock} = \text{IICCLK} / (\text{IICCON}[3:0] + 1)$ - Should be $\text{CK_REG}[3:0] > 0$	-

- The value of CK_REG[3:0] should be more than 0 because if it is 0, SDA outputs 0 continuously although SCL outputs regularly.
- The value of clock is not concerned whether set or not in Slave mode.

NOTE: An IIC-bus interrupt occurs

- When one-byte transmit or receive operation is completed.
- When a general call or a slave address match occurs.
- If bus arbitration fails.

Multi-Master IIC-Bus Control/Status Register (IICSTAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Bits	Name	Type	Description	Reset
7:6	Mode Selection (MODE_SEL)	R/W	IIC-bus master/slave Tx/Rx mode selection 00 = Slave receive mode 01 = Slave transmit mode 10 = Master receive mode 11 = Master transmit mode	0
5	Busy Signal Status/Start-Stop Generation (BB)	R/W	Read Operation: IIC-bus busy signal status bit 0 = Not busy 1 = Busy Write Operation: START-STOP signal generation 0 = STOP signal generation 1 = START signal generation	0
4	Serial Output (SOE)	R/W	IIC-bus data output enable/disable bit 0 = Disable Tx/Rx 1 = Enable Tx/Rx	0
3	Arbitration status Flag (LBA)	R	IIC-bus arbitration procedure status flag 0 = Bus arbitration successful 1 = Bus arbitration failed during serial I/O	0
2	Address-as-slave Status Flag (AAS)	R	IIC-bus address-as-slave status flag 0 = When START/STOP condition was detected 1 = Received slave address matches the address value in the IICADD	0
1	Address Zero Status Flag (ADDR_ZERO)	R	IIC-bus address zero status flag 0 = When START/STOP condition was detected 1 = Received slave address is 00000000b	0
0	Last Received Bit Status Flag (LRB)	R	IIC-bus last received bit status flag 0 = Last received bit is 0 (ACK was received) 1 = Last received bit is 1 (ACK was not received)	0

NOTES

- When IIC sends 1 byte, interrupt is generated regardless of receiving ack. During this interrupt period, SCL is maintained low and holds BUS.
- All IIC devices generate ack to "0" slave address by default. This feature is used when one master wants to write same value to several IIC slaves simultaneously.

Multi-Master IIC-Bus Address Register (IICADD)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															S_ADDR

Bits	Name	Type	Description													Reset
7:1	Slave Address (S_ADDR)	R/W	7-bit slave address. When serial output is disabled, IICADD is writable. Slave address = IICADD[7:1]													-

NOTE: Serial output must be disabled to write self-slave address to IICADD. If serial output is enabled, it is not possible to write slave address and after reset value "0" remains.

Multi-Master IIC-Bus Transmit/Receive Register (IICDS)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															DATA

Bits	Name	Type	Description													Reset
7:0	Data Shift (DATA)	R/W	8-bit data shift register for IIC-bus Tx/Rx operation: When serial output is enabled, IICDS is writable.													-

NOTE: When serial output is enabled, IICDS can be set. Otherwise, IICDS cannot be set so unknown value outputs through SDA.

IIC OPERATION

IIC Clock Frequency

PCLK	CKSEL	CK_REG = 0	CK_REG = 15
121.5 MHz	PCLK / 16	7.593 MHz	474 kHz
	PCLK / 512	237 kHz	14 kHz

For example, the main clock is 121.5 MHz. So the available operation frequency of IIC is as shown in the above table. User can adjust the operation frequency by the CK_REG field in the IICCON register.

Start and Stop Condition

When the IIC-bus interface is inactive, it is usually in Slave mode. In other words, the interface should be in Slave mode before detecting a Start condition on the SDA line (a Start condition can be initiated with a High-to-Low transition of the SDA line while the clock signal of SCL is high). When the interface state is changed to Master mode, a data transfer on the SDA line can be initiated and SCL signal generated.

A Start condition can transfer one-byte serial data over the SDA line, and a Stop condition can terminate the data transfer. A Stop condition is a Low-to-High transition of the SDA line while SCL is high. Start and Stop conditions are always generated by the master. The IIC-bus gets busy when a Start condition is generated. A Stop condition will make the IIC-bus free.

When a master initiates a Start condition, it should send a slave address to notify the slave device. One byte of address field consists of a 7-bit address and a 1-bit transfer direction indicator (showing write or read).

If bit 8 is 0, it indicates a write operation (transmit operation); if bit 8 is 1, it indicates a request for data read (receive operation).

The master will finish the transfer operation by transmitting a Stop condition. If the master wants to continue the data transmission to the bus, it should generate another Start condition as well as a slave address. In this way, the read-write operation can be performed in various formats.

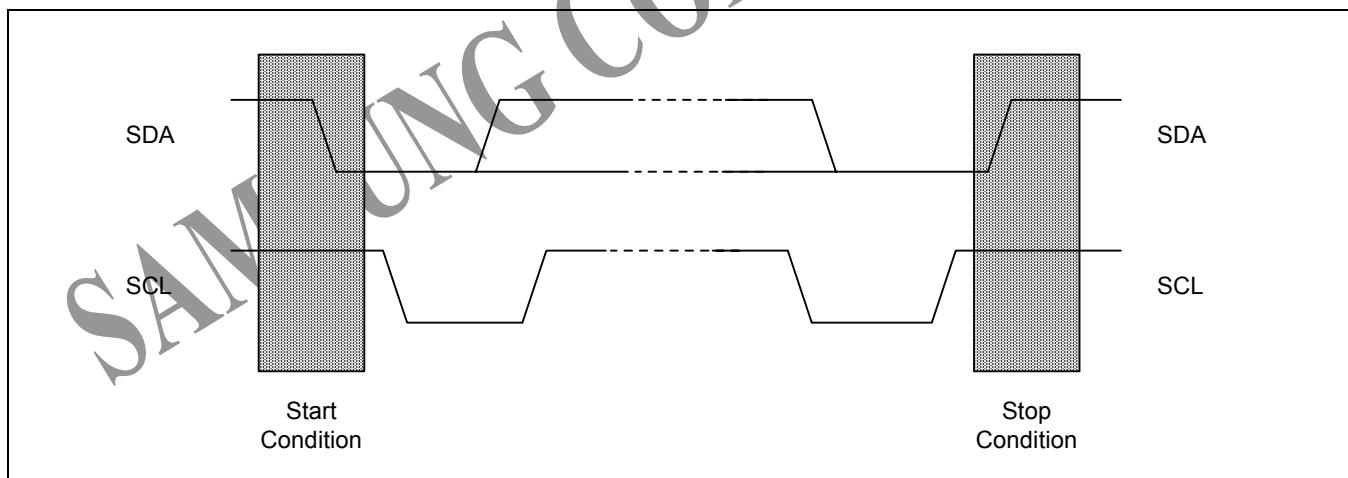
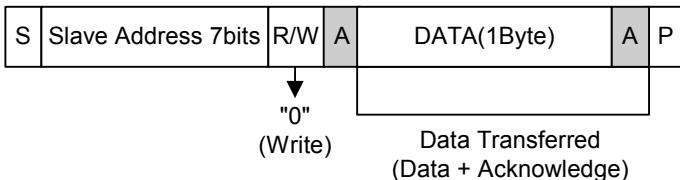


Figure 18-2. Start and Stop Condition

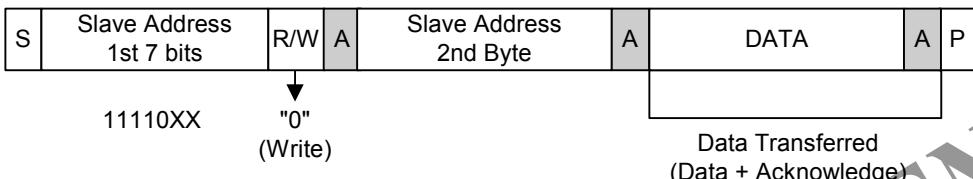
Data Transfer Format

Every byte placed on the SDA line should be eight bits in length. The bytes can be unlimitedly transmitted per transfer. The first byte following a Start condition should have the address field. The master can transmit the address field 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.

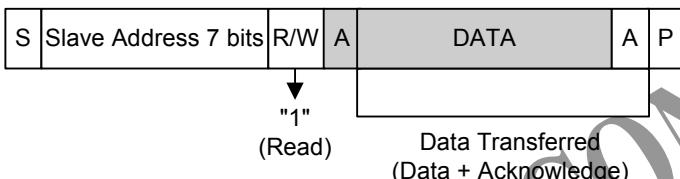
Write Mode Format with 7-bit Addresses



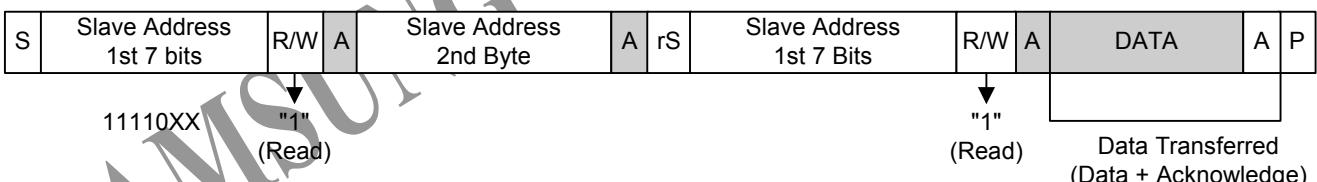
Write Mode Format with 10-bit Addresses



Read Mode Format with 7-bit Addresses



Read Mode Format with 10-bit Addresses



NOTES:

1. S : Start, rS: Repeat Start, P : Stop, A: Acknowledge
2. : From Master to Slave, : From Slave to Master

Figure 18-3. IIC-Bus Interface Data Format

NOTE: When SCL is low and start occurs, restart begins. That is, when SCL is high (BUS is not held) and write 1 to start bit, start begins. When BUS is held and writes 1 to start bit, restart begins.

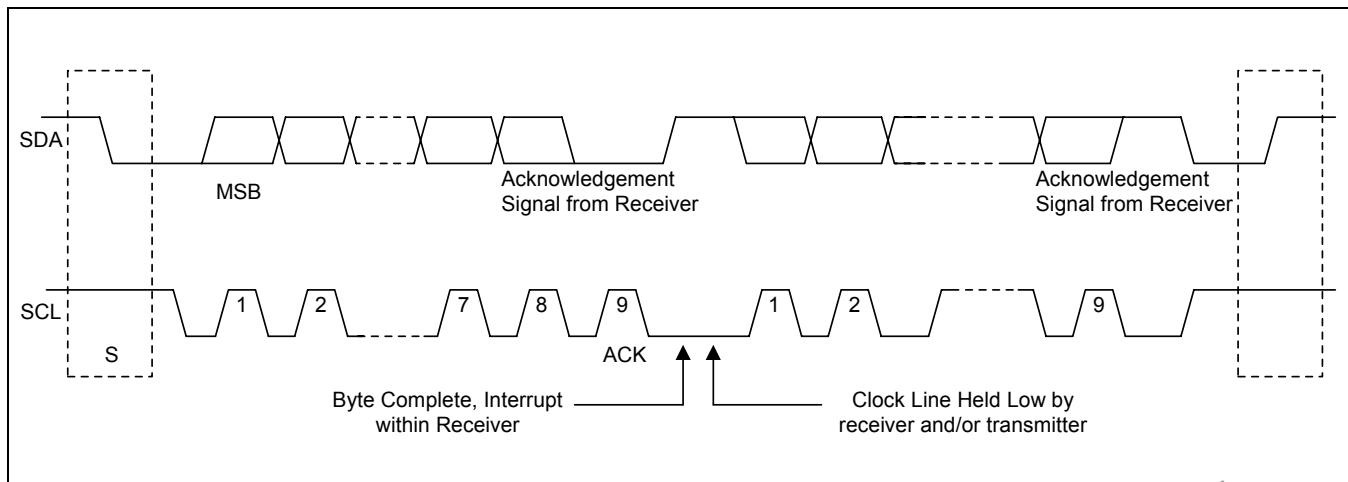


Figure 18-4. Data Transfer on the IIC-Bus

Ack Signal Transmission

To complete a one-byte transfer operation, the receiver should send an ACK bit to the transmitter. The ACK pulse should occur at the ninth clock of the SCL line. Eight clocks are required for the one-byte data transfer. The master should generate the clock pulse required to transmit the ACK bit.

The transmitter should release the SDA line by making the SDA line high when the ACK clock pulse is received. The receiver should also drive the SDA line Low during the ACK clock pulse so that the SDA keeps Low during the high period of the ninth SCL pulse.

The ACK bit transmit function can be enabled or disabled by software (IICSTAT). However, the ACK pulse on the ninth clock of SCL is required to complete the one-byte data transfer operation.

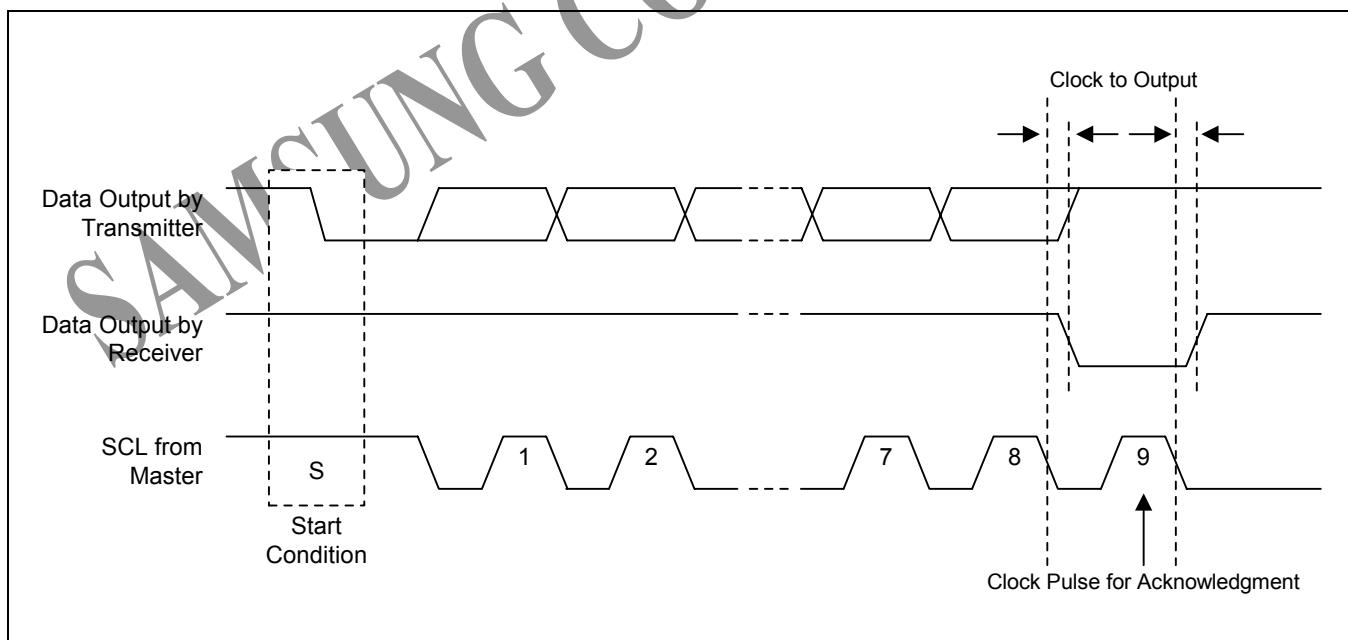


Figure 18-5. Acknowledge on the IIC-Bus

Read-Write Operation

In transmitter mode, when the data is transferred, the IIC-bus interface will wait until IIC-bus Data Shift (IICDS) register receives a new data. Before the new data is written to the register, the SCL line should be held low, and then released after it is written. The IIC controller should hold the interrupt to identify the completion of current data transfer. After the CPU receives the interrupt request, it should write a new data into the IICDS register, again.

In Receive mode, when a data is received, the IIC-bus interface will wait until IICDS register is read. Before the new data is read out, the SCL line will be held low and then released after it is read. The IIC controller should hold the interrupt to identify the completion of the new data reception. After the CPU receives the interrupt request, it should read the data from the IICDS register.

NOTE: Master sends 7-bit address and R/W information on 8th bit to slave and slave is set to Transmit or Receive mode according to R/W information. That is, master decides slave's mode.

Bus Arbitration Procedures

Arbitration takes place on the SDA line to prevent the contention on the bus between two masters. If a master with a SDA high level detects the other master with a SDA active low level, it will not initiate a data transfer because the current level on the bus does not correspond to its own. The arbitration procedure will be extended until the SDA line turns high.

However, when the masters simultaneously lower the SDA line, each master should evaluate whether or not the mastership is allocated to itself. For the purpose of evaluation, each master should detect the address bits. While each master generates the slave address, it should also detect the address bit on the SDA line because the SDA line is likely to get low rather than to keep high. Assume that one master generates a low as first address bit, while the other master is maintaining high. In this case, both masters will detect low on the bus because the low status is superior to the high status in power. When this happens, low (as the first bit of address) generating master will get the mastership while high (as the first bit of address) generating master should withdraw the mastership. If both masters generate low as the first bit of address, there should be arbitration for the second address bit, again. This arbitration will continue to the end of last address bit.

Abort Conditions

If a slave receiver cannot acknowledge the confirmation of the slave address, it should hold the level of the SDA line high. In this case, the master should generate a Stop condition and to abort the transfer.

If a master receiver is involved in the aborted transfer, it should signal the end of the slave transmit operation by canceling the generation of an ACK after the last data byte received from the slave. The slave transmitter should then release the SDA to allow a master to generate a Stop condition.

Configuring IIC-Bus

To control the frequency of the serial clock (SCL), the 4-bit prescaler value can be programmed in the IICCON register. The IIC-bus interface address is stored in the IIC-bus address (IICADD) register. (By default, the IIC-bus interface address has an unknown value.)

FLOWCHARTS OF OPERATIONS IN EACH MODE

The following steps must be executed before any IIC Tx/Rx operations.

1. Write own slave address on IICADD register, if needed.
2. Set IICCON register
 - a) Enable interrupt
 - b) Define SCL period
3. Set IICSTAT to enable Serial Output

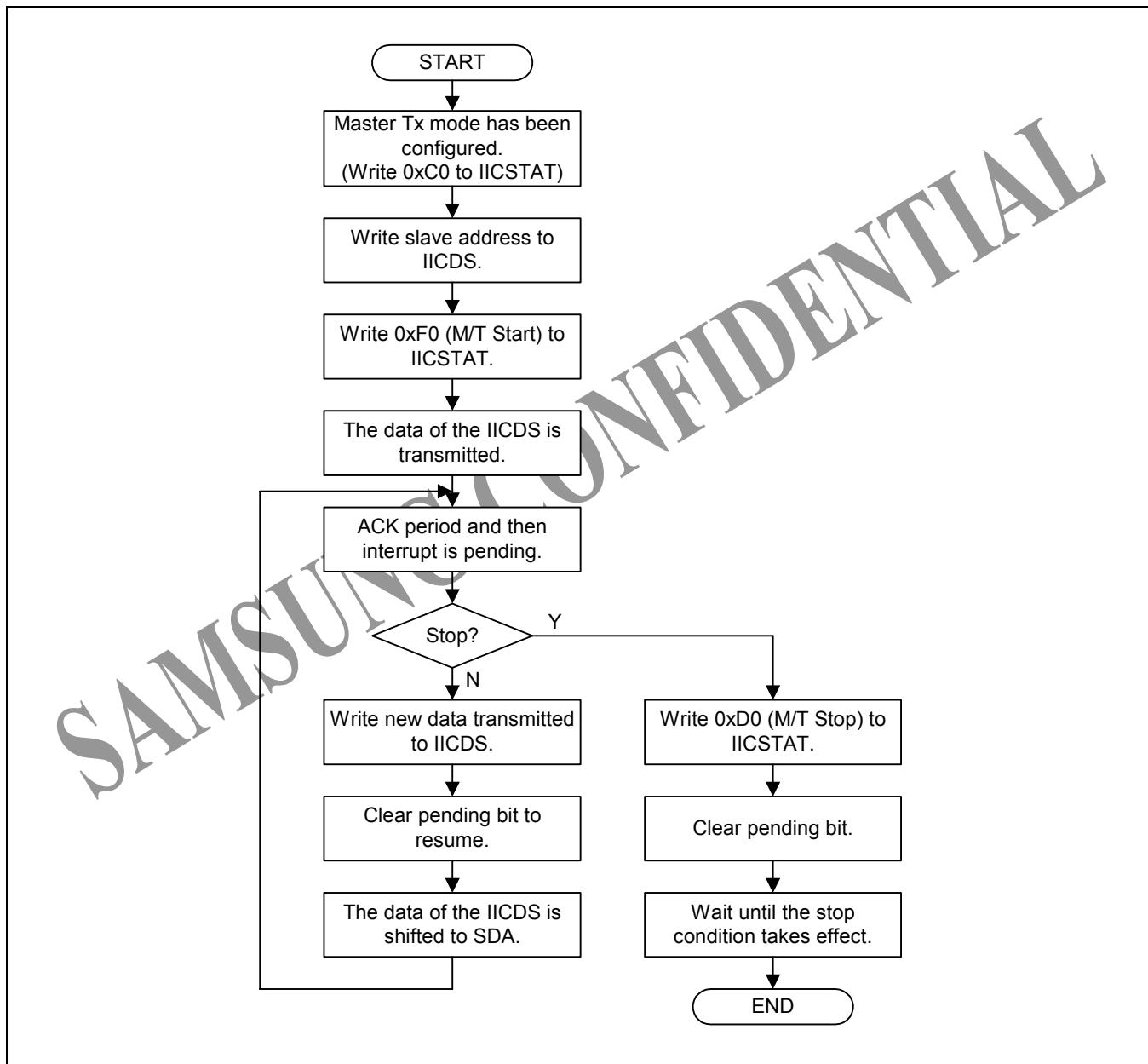


Figure 18-6. Operations for Master/Transmitter Mode

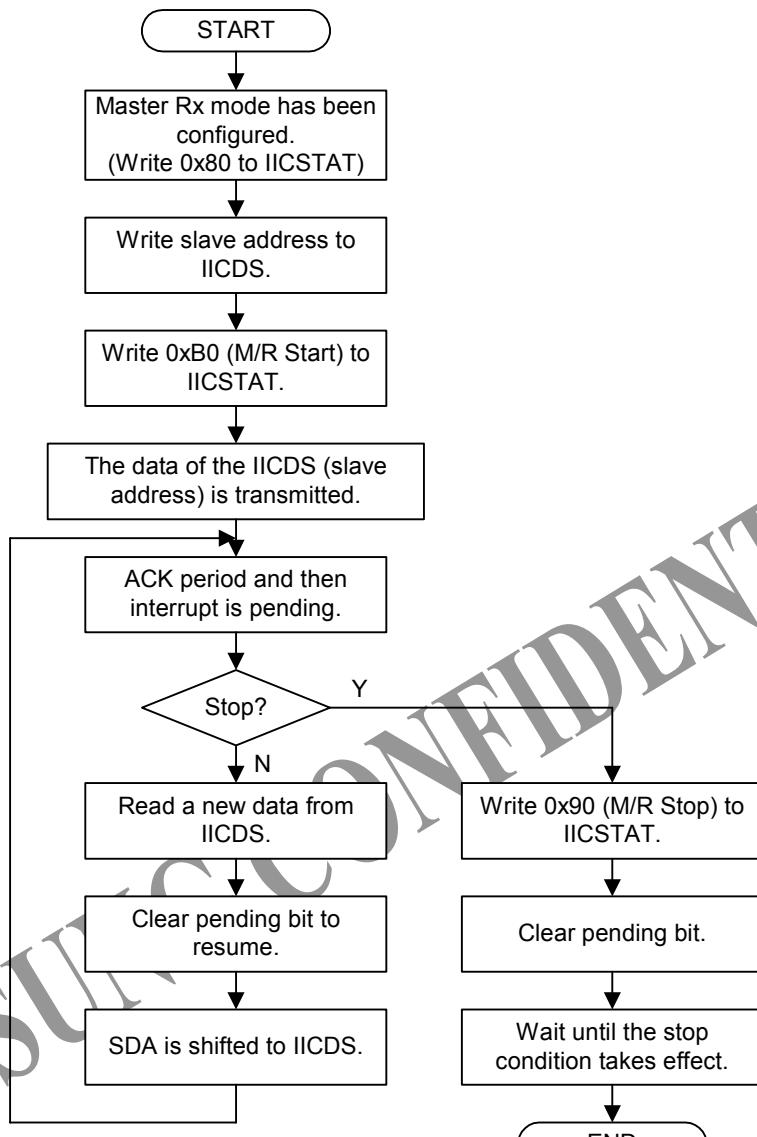


Figure 18-7. Operations for Master/Receiver Mode

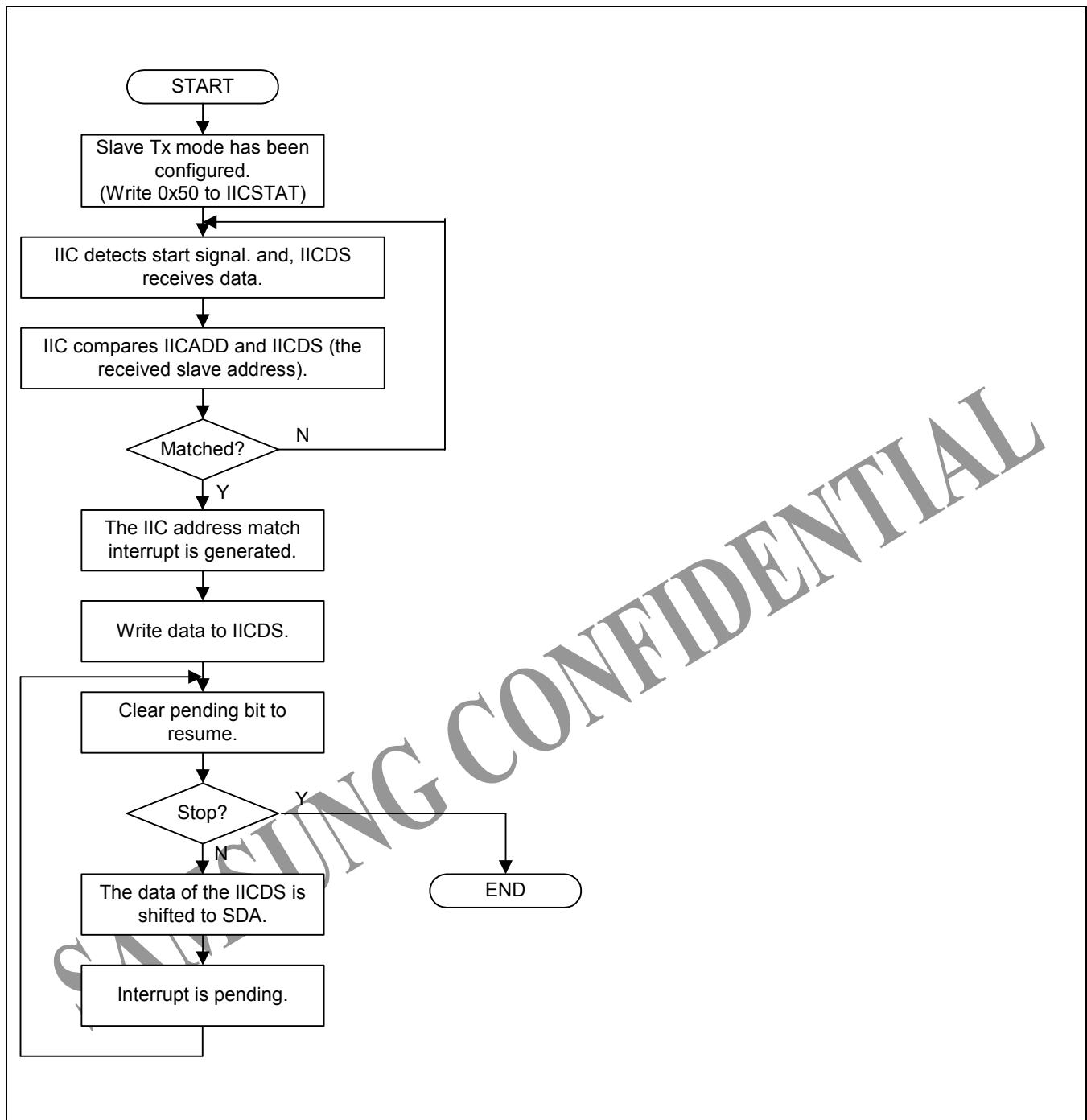


Figure 18-8. Operations for Slave/Transmitter Mode

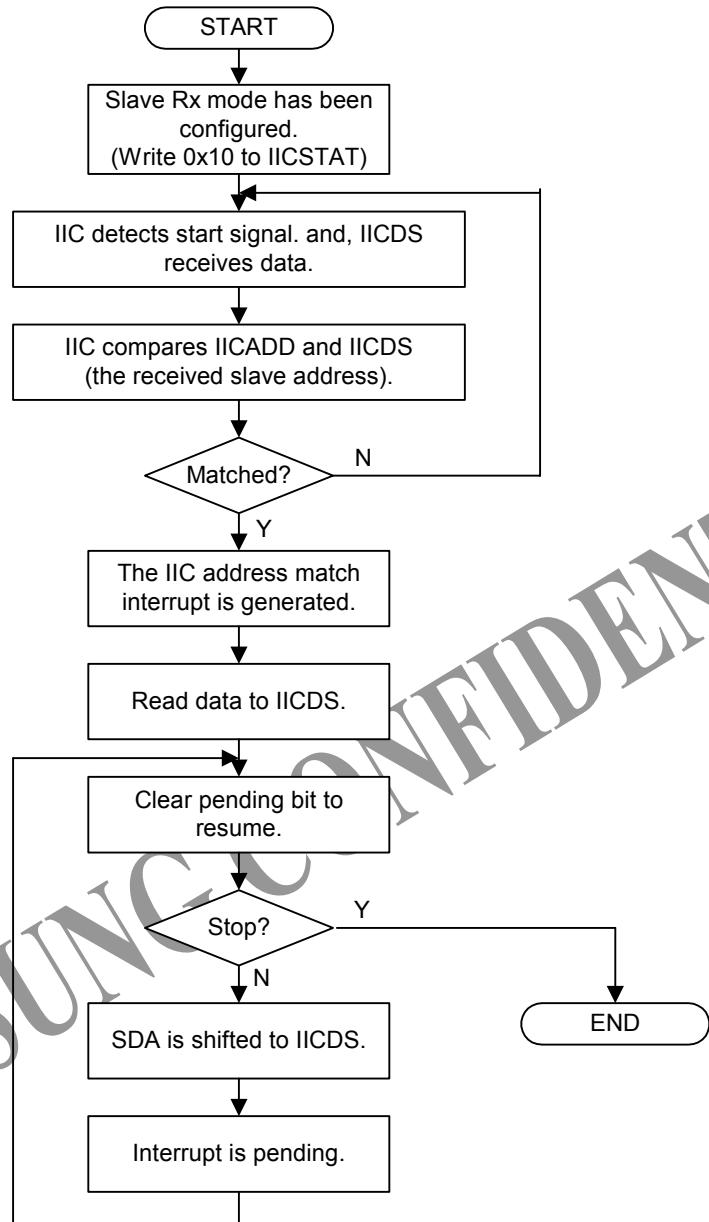


Figure 18-9. Operations for Slave/Receiver Mode

NOTES

SAMSUNG CONFIDENTIAL

19

SERIAL PERIPHERAL INTERFACE (SPI)

OVERVIEW

SPI (Serial Peripheral Interface) in S5L8700X can transfer serial data with various peripherals. SPI has two 8bit shift registers for transmission and reception, respectively. During transfer, SPI receives 8bit serial data at the same time as it transmits at a frequency determined by prescaler register setting. If you want only to transmit data through SPI, you may treat the received data as dummy. On the contrary, if you want only to receive data, you should write dummy '0xFF' data to Tx buffer since transmission and reception happen at the same time. There are 4 I/O pins associated with SPI transfer: SPI clock pin (SCK), MISO (master_in_slave_out) data pin, MOSI (master_out_slave_in) data pin and active low /nSS pin. SCK, MISO and MOSI pins act as inputs or outputs depending on register setting and are stitched in the PAD module using associated signals from SPI module. /nSS input pin indicates that the SPI module is used as a slave by an external master.

FEATURES

- SPI protocol (Ver. 2.11) compatible
- Two 8-bit Shift Registers for transmission and reception
- 10-bit prescaler logic
- Polling, Interrupt and DMA transfer mode
- Master and slave mode
- 4 combinations of clock polarity and clock phase

BLOCK DIAGRAM

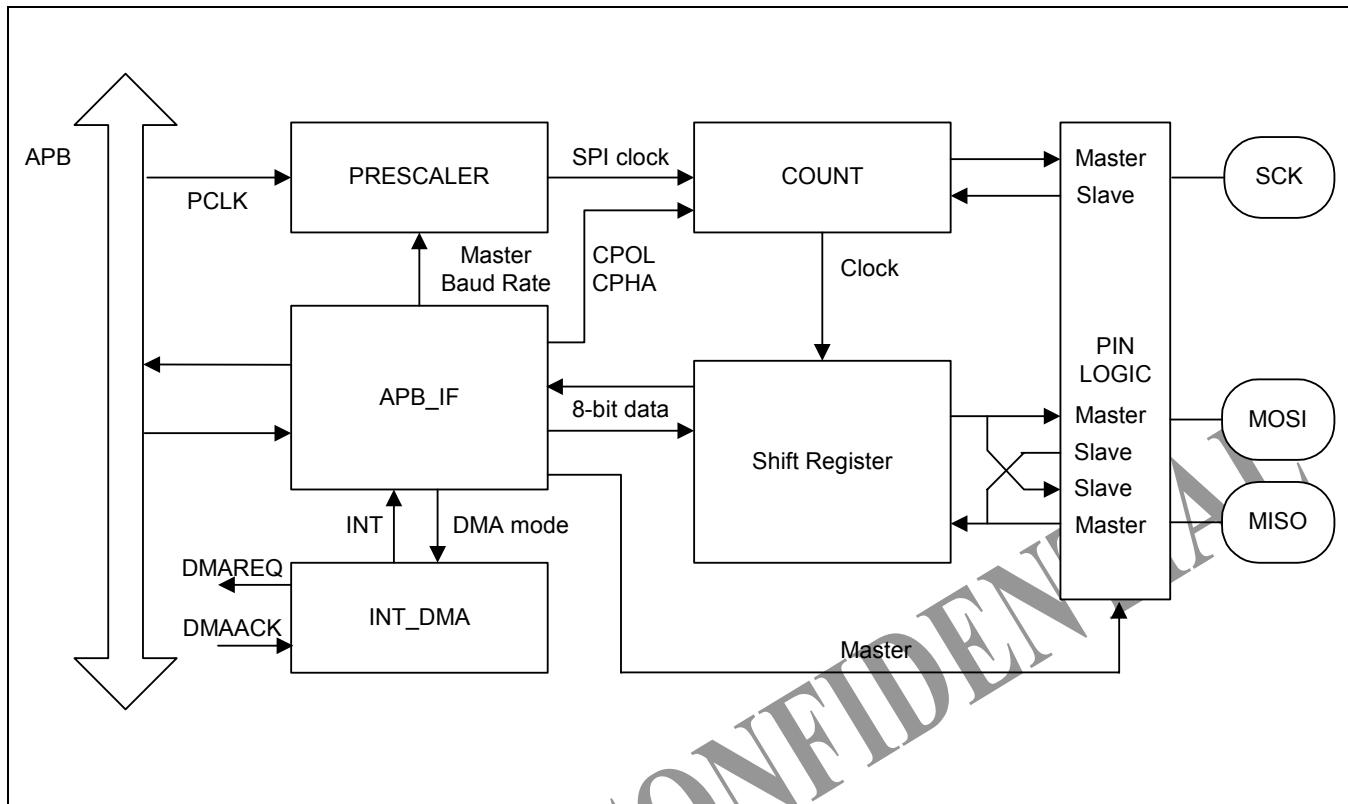


Figure 19-1. SPI Block Diagram

- APB_IF: register bank, interrupt generation
- INT_DMA: DMA request
- PRESCALER: generation of prototype SPI clock from PCLK
- COUNT: generation of SPI clock depending on clock mode, clock count
- Shift Register: two 8-bit shift registers for transmission and reception
- PIN LOGIC: generation of signals needed to make 3 basic SPI pins

REGISTERS

Name	Address (virtual)	R/W	Description	Reset
SPCLKCON	0x3CD0_0000	R/W	Clock Control Register	0x0000 0002
SPCON	0x3CD0_0004	R/W	Control Register	0x0000 0000
SPSTA	0x3CD0_0008	R/W	Status Register	0x0000 0001
SPPIN	0x3CD0_000C	R/W	Pin Control Register	0x0000 0000
SPTDAT	0x3CD0_0010	R/W	Tx Data Register	0x0000 0000
SPRDAT	0x3CD0_0014	R	Rx Data Register	0x0000 0000
SPPRE	0x3CD0_0018	R/W	Baud Rate Prescaler Register	0x0000 0000

SPI Clock Control Register (SPCLKCON)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPCLKCON															
		Bit		Description								Initial State			
		[31:2]		Reserved								0			
SPI clock down ready (read only)		[1]		0 = clock-down not ready 1 = clock-down ready								1			
SPI power on		[0]		0 = SPI power off on 1 = SPI power								0			

SPI Control Register (SPCON)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SPCON	Bit	Description	Initial State
	[31:11]	Reserved	0
Interrupt Enable (Data Collision error)	[10]	0 = interrupt masked 1 = interrupt enable	0
Interrupt Enable (Multi Master error)	[9]	0 = interrupt masked 1 = interrupt enable	0
Interrupt Enable (Transfer Ready)	[8]	0 = interrupt masked 1 = interrupt enable	0
DMA direction	[7]	0 = Tx DMA 1 = Rx DMA	0
SPI mode select (SMOD)	[6:5]	Determine how SPTDAT/SPRDAT is written/read 00 = polling mode 01 = interrupt mode 10 = DMA mode 11 = reserved	00
SPI clock enable	[4]	SPI clock enable (master) or not (slave) 0 = disable 1 = enable	0
Master or Slave mode (MSTR)	[3]	0 = slave mode 1 = master mode NOTE: In slave mode, set-up time is required for master to initiate Tx/Rx	0
Clock Polarity Select (CPOL)	[2]	Select an active high or active low clock 0 = active high 1 = active low	0
Clock Phase Select (CPHA)	[1]	This bit selects one of two fundamentally different transfer formats. 0 = first clock edge missing 1 = last clock edge missing	0
Tx Auto Garbage Data mode enable (TAGD)	[0]	This bit decides whether receiving data automatically transmits dummy 0xFF data or not. When you only want to receive data, you don't have to transmit dummy 0xFF data if this bit is set. 0 = normal mode 1 = Tx auto garbage data mode	0

SPI Status Register (SPSTA)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SPSTA	Bit	Description	Initial State
	[31:4]	Reserved	0
Interrupt Status Bit (Data Collision Error)	[3]	0 = no interrupt 1 = interrupt pending (This interrupt is generated if SPTDAT is written or SPRDAT is read while transfer is in progress. This bit can be reset by writing '1'. Writing '0' has no effect.)	0
Interrupt Status Bit (Multi Master Error)	[2]	0 = no interrupt 1 = interrupt pending (This interrupt is generated if nSS signal goes to low while SPI is configured as a master and ENMUL bit of SPPIN register is set. This bit can be reset by writing '1'. Writing '0' has no effect.)	0
Interrupt Status Bit (Transfer Ready)	[1]	0 = no interrupt 1 = interrupt pending (This interrupt is generated if transfer ready flag is high in the state of interrupt mode. This bit can be reset by writing '1'. Writing '0' has no effect.)	0
Transfer Ready Flag (Read only)	[0]	This flag indicates that SPTDAT or SPRDAT is ready to transmit or receive. It is automatically cleared by writing data to SPTDAT. 0 = not ready 1 = data Tx/Rx ready	1

SPI Pin Control Register (SPPIN)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SPPIN	Bit	Description	Initial State
	[31:4]	Reserved	0
nSSI pin enable	[3]	This bit determine whether to use nSSI pin or not 0 = don't use (slave condition: MSTR bit is off) 1 = use (slave condition: MSTR bit is off and nSSI is low)	0
Multi Master Error Detect Enable (ENMUL)	[2]	This bit enables Multi Master Error Flag of SPSTA to be set when multi master error occurs. 0 = disable (general purpose) 1 = multi master error detect enable	0
	[1]	Reserved	0
Master Out Keep (KEEP)	[0]	Determine whether state is kept or released in MOSI when 1 byte transfer is finished. (only master) 0 = release 1 = keep the state	0

SPI Tx Data Register (SPTDAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SPTDAT	Bit	Description	Initial State
	[31:8]	Reserved	0
Tx Data	[7:0]	This field contains the data to be transmitted over SPI channel	0

SPI RX Data Register (SPRDAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SPRDAT	Bit	Description	Initial State
	[31:8]	Reserved	0
Rx Data	[7:0]	This field contains the data to be received over SPI channel	0

SPI Baud Rate Prescaler (SPPRE)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SPPRE	Bit	Description	Initial State
	[31:8]	Reserved	0
Prescaler Value	[9:0]	Determines SPI clock rate with the following equation. Baud Rate = PCLK / (2 * (prescaler value +1))	0

SPI OPERATIONS

SPI Transfer Format

SPI has 4 transfer formats depending on CPOL and CPHA values in SPCON register. SPI generates clock only when SPI transfers or receives data. CPHA indicates whether SPICLK has a leading edge of the first clock (CPHA = 1) or trailing edge of the last clock(CPHA = 0). CPOL determines the polarity of SPICLK when SPI bus is idle.

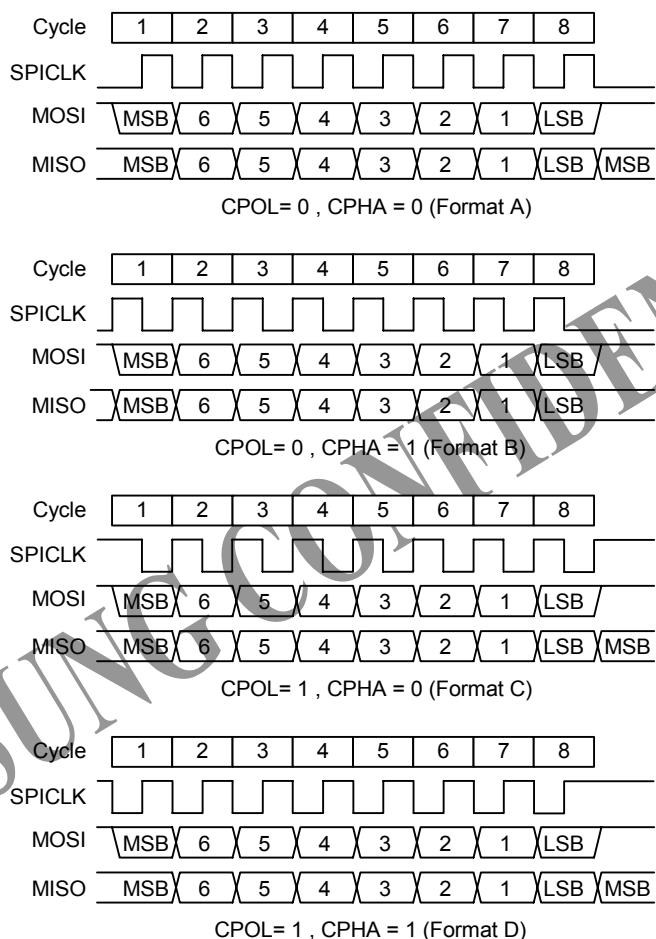


Figure 19-2. SPI Transfer Format

SPI OPERATION

Using the SPI operation, 8-bit data can be sent to and received from external device simultaneously. A serial clock line synchronizes shifting and sampling of the data on the two serial data line. When SPI acts as a master, transmission frequency can be controlled by setting the appropriate bit to SPRPE register. You can modify its frequency to adjust the baud rate data register value. When SPI acts as a slave, the other master supplies the clock. SPI starts its operation as soon as a byte data is written to SPTDAT register. If 0th bit of SPCON is set, reading SPRDAT register makes data transfer start without needing to write 0xFF to SPTDAT.

Programming Procedure

When a byte data is written to the SPTDAT register, SPI starts to transmit data if clock-enable bit and master-slave bit of SPCON register are set. To operate SPI module, refer to the following steps.

1. Set Baud Rate Prescaler Register (SPPRE).
2. Set SPCON to configure SPI module properly
3. Set the GPIO pin which acts as nSS of external SPI device to low to make external SPI device slave.
4. To transmit data, check the status of Transfer Ready flag of SPSTA register is high and then write data to SPTDAT.
5. To receive data, if 0th bit of SPCON (TAGD) is inactive, write 0xFF to SPTDAT register, confirm REDY to set and then read data from SPRDAT register. If TAGD bit is active, confirm REDY to set and then read data from SPRDAT register. In this case, there is no need to write data to SPTDAT.
6. Set the GPIO pin which acts as nSS to high to deactivate external SPI device.

Steps for DMA Mode Transmission

1. Configure SPI as DMA mode.
2. The SPI requests DMA service.
3. DMA transmits 1 byte data to the SPI.
4. The SPI transmits data to external SPI module
5. Go to step 2 until DMA count is 0.
6. The SPI is configured as interrupt or polling mode.

Steps for DMA mode Reception

1. Configure SPI as DMA mode and set TAGD bit to high
2. The SPI receives 1 byte data from external SPI module.
3. The SPI requests DMA service.
4. DMA receives the data from the SPI.
5. Write data 0xFF automatically to SPTDAT.
6. Go to step 4 until DMA count is 0.
7. The SPI is configured as polling mode and TAGD bit is cleared.
8. When REDY flag of SPSTA register is set, read the last byte data.

NOTE: Total received data = DMA TC values + the last datum received in polling mode (step 9).

The first received datum is dummy, so user may neglect that.

NOTES

SAMSUNG CONFIDENTIAL

20 ADC CONTROLLER

OVERVIEW

The 10-bit CMOS ADC(Analog to Digital Converter) is a recycling type device with 4-channel analog inputs. It converts the analog input signal into 10-bit binary digital codes at a maximum conversion rate of 500KSPS with 2.5MHz A/D converter clock. A/D converter operates with on-chip sample-and-hold function and power down mode is supported.

Touch Screen Interface is controlling pads(XP, XM, YP, YM) of Touch Screen and selecting pads(XP, XM, YP, YM) of The Touch Screen for X-position conversion, Y-position conversion. The Touch Screen Interface contains Touch Screen Pads control logic and ADC interface logic with an interrupt generation logic.

Features

- Resolution: 10-bit
- Differential Linearity Error: ± 1.0 LSB
- Integral Linearity Error: ± 2.0 LSB
- Maximum Conversion Rate: 500 KSPS
- Low Power Consumption
- Power Supply Voltage: 3.3V
- Analog Input Range: 0 ~ 3.3V
- On-chip sample-and-hold function
- Normal Conversion Mode
- Separate X/Y position conversion Mode
- Auto(Sequential) X/Y Position Conversion Mode
- Waiting for Interrupt Mode
- CPU IDLE mode wakeup source

ADC & TOUCH SCREEN INTERFACE OPERATION

BLOCK DIAGRAM

Figure 1 shows the functional block diagram of A/D converter and Touch Screen Interface. Note that the A/D converter device is a recycling type.

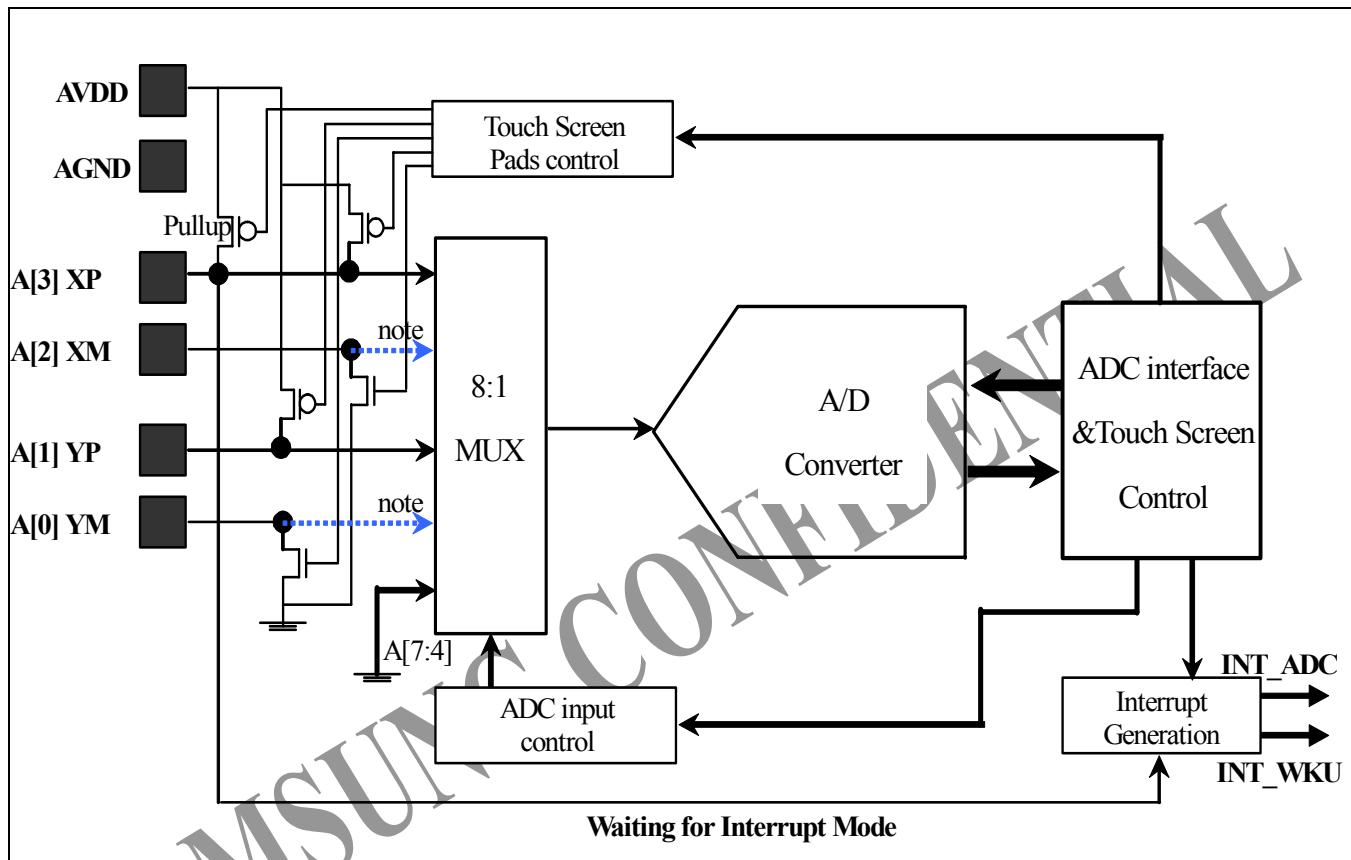


Figure 20-1. ADC and Touch Screen Interface Functional Block Diagram

*note (symbol)

When Touch Screen device is used, XM or PM is only connected ground for Touch Screen I/F.

When Touch Screen device is not used, XM or PM is connecting Analog Input Signal for Normal ADC conversion.



FUNCTION DESCRIPTIONS

A/D Conversion Time

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

$$\text{A/D converter freq.} = 50\text{MHz}/(49+1) = 1\text{MHz}$$

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

NOTE:

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

Touch Screen Interface Mode**1. Normal Conversion Mode**

Single Conversion Mode is the most likely used for General Purpose ADC Conversion. This mode can be initialized by setting the ADCCON (ADC Control Register) and completed with a read and a write to the ADCDAT0 (ADC Data Register 0).

2. Separate X/Y position conversion Mode

Touch Screen Controller can be operated by one of two Conversion Modes. Separate X/Y Position Conversion Mode is operated as the following way. X-Position Mode writes X-Position Conversion Data to ADCDAT0, so Touch Screen Interface generates the Interrupt source to Interrupt Controller. Y-Position Mode writes Y-Position Conversion Data to ADCDAT1, so Touch Screen Interface generates the Interrupt source to Interrupt Controller.

3. Auto(Sequential) X/Y Position Conversion Mode

Auto(Sequential) X/Y Position Conversion Mode is operated as the following. Touch Screen Controller sequentially converts X-Position and Y-Position that is touched. After Touch controller writes X-measurement data to ADCDAT0 and writes Y-measurement data to ADCDAT1, Touch Screen Interface is generating Interrupt source to Interrupt Controller in Auto Position Conversion Mode.

4. Waiting for Interrupt Mode

Touch Screen Controller is generating wake-up (WKU) signal when the system is CPU IDLE mode (Power Down). The Waiting for Interrupt Mode of Touch Screen Controller must be set state of Pads(XP, XM, YP, YM) in Touch Screen Interface.

After Touch Screen Controller is generating Wake-Up signal (INT_WKU), Waiting for interrupt Mode must be cleared. (XY_PST sets to the No operation Mode)

Standby Mode

Standby mode is activated when ADCCON [2] is set to '1'. In this mode, A/D conversion operation is halted and ADCDAT0, ADCDAT1 register contains the previous converted data.

Programming Notes

1. The A/D converted data can be accessed by means of interrupt or polling method. With interrupt method the overall conversion time - from A/D converter start to converted data read - may be delayed because of the return time of interrupt service routine and data access time. With polling method, by checking the ADCCON[15] - end of conversion flag-bit, the read time from ADCDAT register can be determined.
2. Another way for starting A/D conversion is provided. After ADCCON[1] - A/D conversion start-by-read mode-is set to 1, A/D conversion starts simultaneously whenever converted data is read.

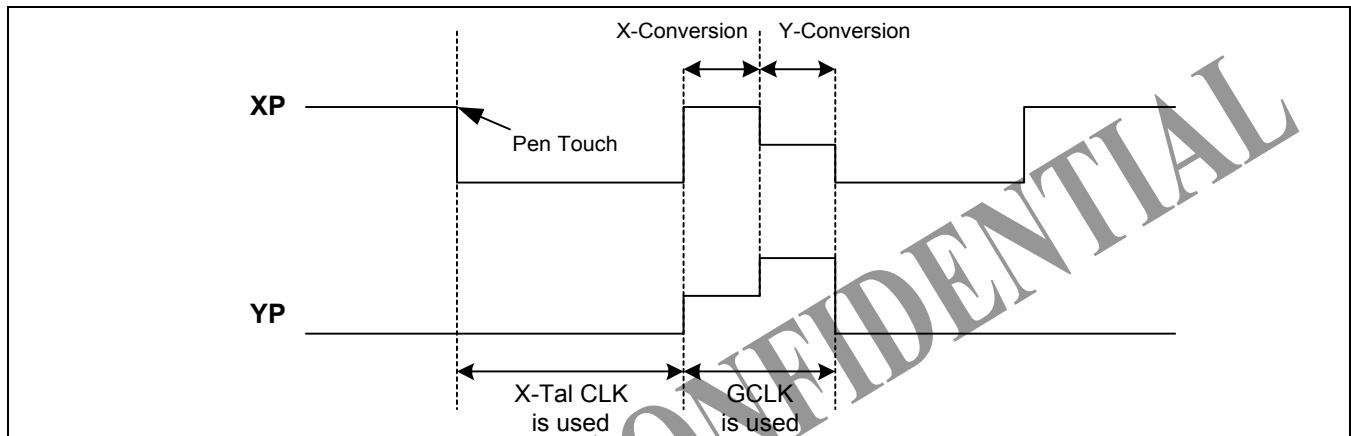


Figure 20-2 ADC and Touch Screen Operation signal

3. If use wakeup source in CPU IDLE mode, XY_PST bit(ADCTSC[1:0]) set Waiting for interrupt mode(2'b11).
stylus pen up/down wakeup is selected UD_SEN bit(ADCTSC[8]).

ADC AND TOUCH SCREEN INTERFACE SPECIAL REGISTERS

ADC CONTROL REGISTER (ADCCON)

Register	Address	R/W	Description	Reset Value
ADCCON	0x3CE00000	R/W	ADC Control Register	0x3FC4

ADCCON	Bit	Description	Initial State
ECFLG	[15]	End of conversion flag(Read only) 0 = A/D conversion in process 1 = End of A/D conversion	0
PRSCEN	[14]	A/D converter prescaler enable 0 = Disable 1 = Enable	0
PRSCVL	[13:6]	A/D converter prescaler value Data value: 0 ~ 255 NOTE: ADC Frequency should be set less than PCLK by 5times. (Ex. PCLK=10MHZ, ADC Freq.< 2MHz)	0xFF
SEL_MUX	[5:3]	Analog input channel select 000 = AIN 0, YM 001 = AIN 1, YP 010 = AIN 2, XM 011 = AIN 3, XP 1xx = Not used channel	0
STDBM	[2]	Standby mode select 0 = Normal operation mode 1 = Standby mode	1
READ_START	[1]	A/D conversion start by read 0 = Disable start by read operation 1 = Enable start by read operation	0
ENABLE_START	[0]	A/D conversion starts by 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

ADC TOUCH SCREEN CONTROL REGISTER (ADCTSC)

Register	Address	R/W	Description	Reset Value
ADCTSC	0x3CE00004	R/W	ADC Touch Screen Control Register	0x58

ADCTSC	Bit	Description	Initial State
UD_SEN	[8]	Detect Stylus Up or Down status. 0 = Detect Stylus Down Interrupt Signal. 1 = Detect Stylus Up Interrupt Signal.	0
YM_SEN	[7]	YM Switch Enable 0 = YM Output Driver Disable. 1 = YM Output Driver Enable.	0
YP_SEN	[6]	YP Switch Enable 0 = YP Output Driver Enable. 1 = YP Output Driver Disable.	1
XM_SEN	[5]	XM Switch Enable 0 = XM Output Driver Disable. 1 = XM Output Driver Enable.	0
XP_SEN	[4]	XP Switch Enable 0 = XP Output Driver Enable. 1 = XP Output Driver Disable.	1
PULL_UP	[3]	Pull-up Switch Enable 0 = XP Pull-up Enable. 1 = XP Pull-up Disable.	1
AUTO_PST	[2]	Automatically sequencing conversion of X-Position and Y-Position 0 = Normal ADC conversion. 1 = Auto Sequential measurement of X-position, Y-position.	0
XY_PST	[1:0]	Manually measurement of X-Position or Y-Position. 00 = No operation mode 01 = X-position measurement 10 = Y-position measurement 11 = Waiting for Interrupt Mode	0

NOTE: 1) While waiting for Touch screen Interrupt, XP_SEN bit should be set to '1', namely 'XP Output disable' and PULL_UP bit should be set to '0', namely 'XP Pull-up enable'.

2) AUTO_PST bit should be set '1' only in Automatic & Sequential X/Y Position conversion.

Touch screen pin conditions in X/Y position conversion.

	XP	XM	YP	YM	ADC ch. select
X Position	Vref	GND	Hi-Z	Hi-Z	YP
Y Position	Hi-Z	Hi-Z	Vref	GND	XP

ADC START DELAY REGISTER (ADCDLY)

Register	Address	R/W	Description	Reset Value
ADCDLY	0x3CE00008	R/W	ADC Start or Interval Delay Register	0x00ff

ADCDLY	Bit	Description	Initial State
DELAY	[15:0]	<p>1) Normal Conversion Mode, XY Position Mode, Auto Position Mode. → ADC conversion start delay value.</p> <p>2) Waiting for Interrupt Mode. When Stylus Down occurs at CPU IDLE MODE, generates Wake-Up signal, having interval(several ms), for Exiting CPU IDLE MODE.</p> <p>Note) Don't use Zero value(0x0000)</p>	00ff

ADC CONVERSION DATA REGISTER (ADCDAT0)

Register	Address	R/W	Description	Reset Value
ADCDAT0	0x3CE0000C	R	ADC Conversion Data Register	-

ADCDAT0	Bit	Description	Initial State
UPDOWN	[15]	Up or Down state of Stylus at Waiting for Interrupt Mode. 0 = Stylus down state. 1 = Stylus up state.	-
AUTO_PST	[14]	Automatic sequencing conversion of X-Position and Y-Position 0 = Normal ADC conversion. 1 = Sequencing measurement of X-position, Y-position.	-
XY_PST	[13:12]	Manually measurement of X-Position or Y-Position. 00 = No operation mode 01 = X-position measurement 10 = Y-position measurement 11 = Waiting for Interrupt Mode	-
Reserved	[11:10]	Reserved	
XPDATA (Normal ADC)	[9:0]	X-Position Conversion data value (include Normal ADC Conversion data value) Data value : 0 ~ 3FF	-

ADC CONVERSION DATA REGISTER (ADCDAT1)

Register	Address	R/W	Description	Reset Value
ADCDAT1	0x3CE00010	R	ADC Conversion Data Register	-

ADCDAT1	Bit	Description	Initial State
UPDOWN	[15]	Up or Down state of Stylus at Waiting for Interrupt Mode. 0 = Stylus down state. 1 = No stylus down state.	-
AUTO_PST	[14]	Automatically sequencing conversion of X-Position and Y-Position 0 = Normal ADC conversion. 1 = Sequencing measurement of X-position, Y-position.	-
XY_PST	[13:12]	Manually measurement of X-Position or Y-Position. 00 = No operation mode 01 = X-position measurement 10 = Y-position measurement 11 = Waiting for Interrupt Mode	-
Reserved	[11:10]	Reserved	
YPDATA	[9:0]	Y-Position Conversion data value Data value : 0 ~ 3FF	-

ADC TOUCH SCREEN UP-DOWN REGISTER (ADCUPDN)

Register	Address	R/W	Description	Reset Value
ADCUPDN	0x3CE00014	R/W	Stylus Up or Down Interrpt Register	0x0

ADCDAT1	Bit	Description	Initial State
TSC_DN	[1]	Stylus Down Interrupt. 0 = No stylus down status. 1 = Stylus down status.	0
TSC_UP	[0]	Stylus Up Interrupt. 0 = No stylus up status. 1 = Stylus up status.	0

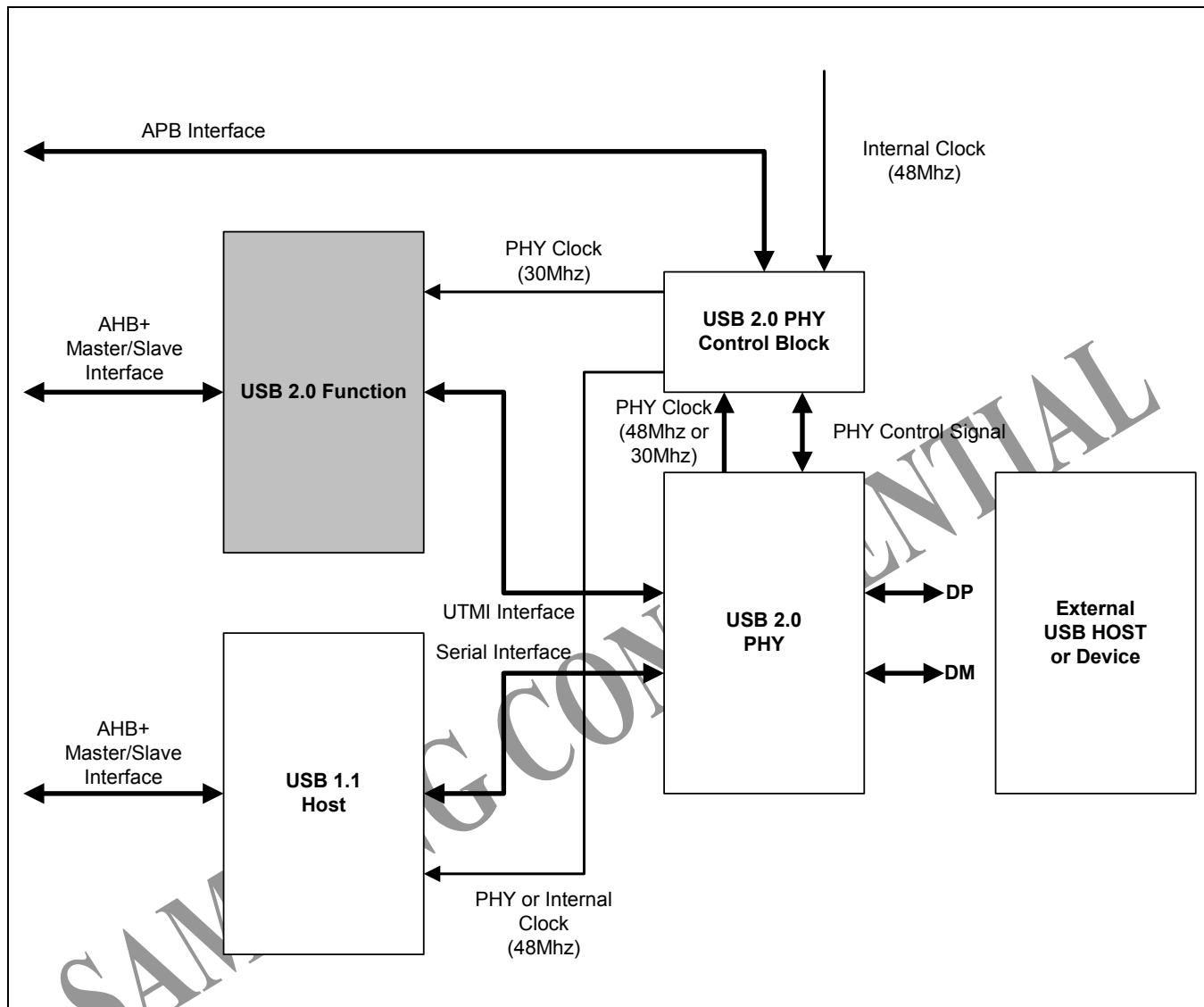
21 USB 2.0 FUNCTION

OVERVIEW

The Samsung USB 2.0 Controller is designed to aid the rapid implementation of the USB 2.0 peripheral device. The controller supports both High and Full speed mode. With the standard UTMI interface and AHB interface. The USB 2.0 Controller can support up to 7 Endpoints (including Endpoint0) with programmable INTERRUPT, BULK and ISOCHRONOUS transfer mode.

FEATURE

- Compliant to USB 2.0 specification
- Supports FS/HS dual mode operation
- EP 0 FIFO: 64 bytes.
- EP 1/2/3/4 FIFO: 512 bytes double buffering
- EP 5/6 FIFO: 64 bytes double buffering
- Convenient Debugging
- Support Interrupt, Bulk, Isochronous Transfer

BLOCK DIAGRAM**Figure 21-1. USB2.0 Block Diagram**

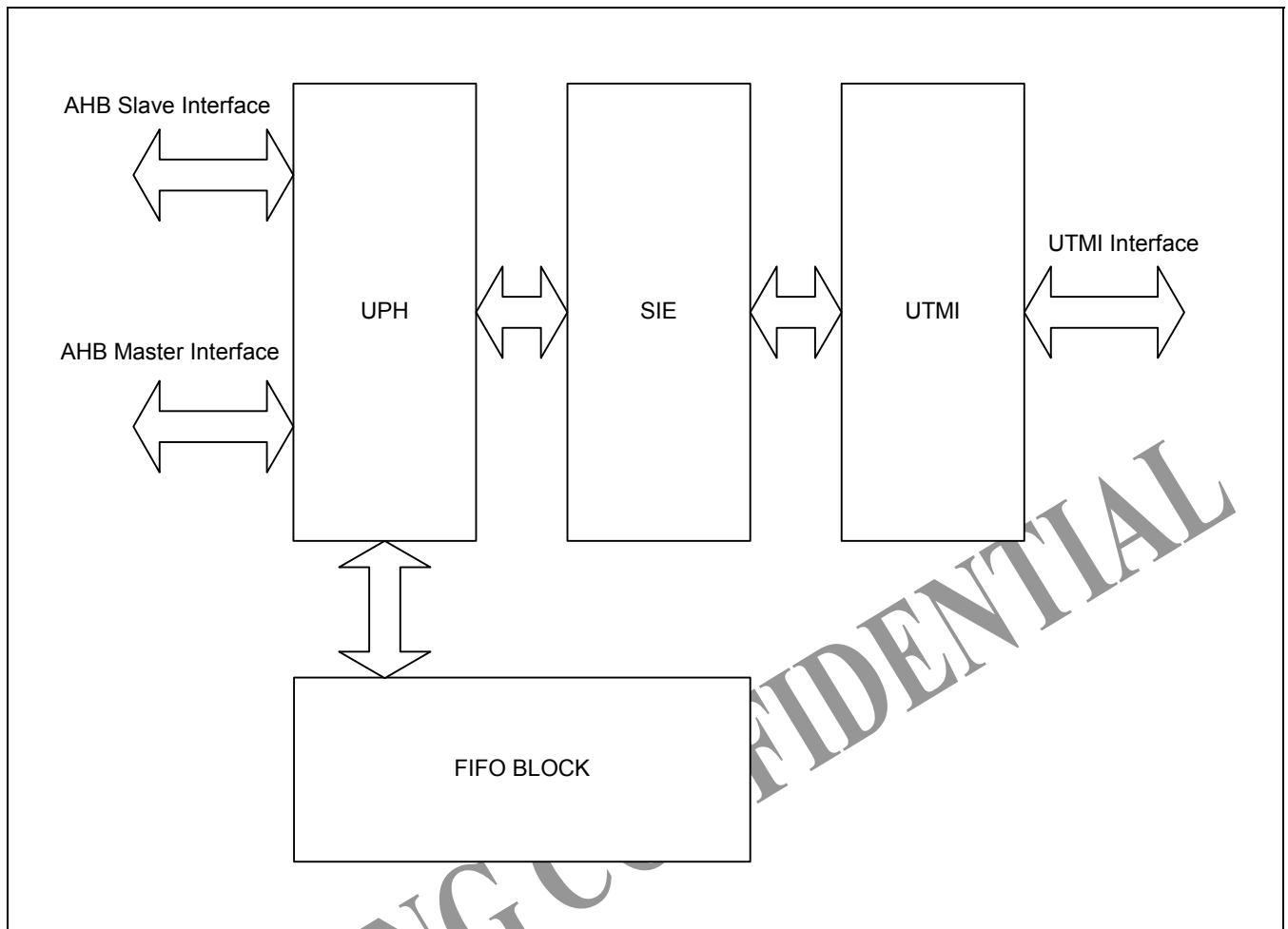


Figure 21-2. USB2.0 Function Block Diagram

USB 2.0 FUNCTION CONTROLLER SPECIAL REGISTERS

The USB 2.0 controller includes several 16-bit registers for the endpoint programming and debugging. The registers can be grouped into two categories. Few of the indexed registers are related to endpoint 0, but most of them are utilized for the control and status monitoring of each data endpoint, including FIFO control and packet size configuration. The buffer register for TX/RX data buffering also belong to the indexed register. The non-indexed registers are mainly used for the control and status checking of the system. The control and status registers of endpoint0 belong to these non-indexed registers.

Table 21-1. Non-Indexed Registers

Register	Address	R/W	Description
IR	0x3880_0000	R/W	Index Register
EIR	0x3880_0004	R/W	Endpoint Interrupt Register
EIER	0x3880_0008	R/W	Endpoint Interrupt Enable Register
FAR	0x3880_000C	R	Function Address Register
FNR	0x3880_0010	R	Frame Number Register
EDR	0x3880_0014	R/W	Endpoint Direction Register
TR	0x3880_0018	R/W	Test Register
SSR	0x3880_001C	R/W	System Status Register
SCR	0x3880_0020	R/W	System Control Register
EP0SR	0x3880_0024	R/W	EP0 Status Register
EP0CR	0x3880_0028	R/W	EP0 Control Register
EP0BR	0x3880_0060	R/W	EP0 Buffer Register
EP1BR	0x3880_0064	R/W	EP1 Buffer Register
EP2BR	0x3880_0068	R/W	EP2 Buffer Register
EP3BR	0x3880_006C	R/W	EP3 Buffer Register
EP4BR	0x3880_0070	R/W	EP4 Buffer Register
EP5BR	0x3880_0074	R/W	EP5 Buffer Register
EP6BR	0x3880_0078	R/W	EP6 Buffer Register

Table 21-2. Indexed Registers

Register	Address	R/W	Description
ESR	0x3880_002C	R/W	Endpoints Status Register
ECR	0x3880_0030	R/W	Endpoints Control Register
BRCR	0x3880_0034	R	Byte Read Count Register
BWCR	0x3880_0038	R/W	Byte Write Count Register
MPR	0x3880_003C	R/W	Max Packet Register
MCR	0x3880_0040	R/W	Master Control Register
MTCR	0x3880_0044	R/W	Master Transfer Counter Register
MFCR	0x3880_0048	R/W	Master FIFO Counter Register
MTTCR1	0x3880_004C	R/W	Master Total Transfer Counter1 Register
MTTCR2	0x3880_0050	R/W	Master Total Transfer Counter2 Register
MICR	0x3880_0084	R/W	Master Interface Control Register
MBAR1	0x3880_0088	R/W	Memory Base Address Register1
MBAR2	0x3880_008C	R/W	Memory Base Address Register2
MCAR1	0x3880_0094	R/W	Memory Current Address Register1
MCAR2	0x3880_0098	R/W	Memory Current Address Register2

REGISTERS

INDEX REGISTER (IR)

The index register is used for indexing a specific endpoint. In most cases, setting the index register value should precede any other operation.

Register	Address	R/W	Description	Reset Value
IR	0x3880_0000	R/W	Index Register	32 bits

IR	Bit	R/W	Description	Initial State
	[31:3]		Reserved	
INDEX	[2:0]	R/W	Endpoint Number Select (0~6) 000 = Endpoint0 001 = Endpoint1 010 = Endpoint2 011 = Endpoint3 100 = Endpoint4 101 = Endpoint5 110 = Endpoint6	000

ENDPOINT INTERRUPT REGISTER (EIR)

The endpoint interrupt register lets the MCU knows what endpoint generates the interrupt. The source of an interrupt could be various, but, when an interrupt is detected, the endpoint status register should be checked to identify if it's related to specific endpoint. Clearing the bits can be accomplished by writing "1" to the bit position where the interrupt is detected.

Register	Address	R/W	Description	Reset Value
EIR	0x3880_0004	R/C	Endpoint Interrupt Register	32 bits

EIR	Bit	R/W	Description	Initial State
	[31:7]		Reserved	
EP6I	[6]	R/C	Endpoint 6 Interrupt Flag	0
EP5I	[5]	R/C	Endpoint 5 Interrupt Flag	0
EP4I	[4]	R/C	Endpoint 4 Interrupt Flag	0
EP3I	[3]	R/C	Endpoint 3 Interrupt Flag	0
EP2I	[2]	R/C	Endpoint 2 Interrupt Flag	0
EP1I	[1]	R/C	Endpoint 1 Interrupt Flag	0
EP0I	[0]	R/C	Endpoint 0 Interrupt Flag	0

ENDPOINT INTERRUPT ENABLE REGISTER (EIER)

Pairing with interrupt register, this register enables interrupt for each endpoints.

Register	Address	R/W	Description	Reset Value
EIER	0x3880_0008	R/W	Endpoint Interrupt Enable Register	32 bits

EIER	Bit	R/W	Description	Initial State
	[31:7]		Reserved	
EP6IE	[6]	R/W	Endpoint 6 Interrupt Enable Flag	0
EP5IE	[5]	R/W	Endpoint 5 Interrupt Enable Flag	0
EP4IE	[4]	R/W	Endpoint 4 Interrupt Enable Flag	0
EP3IE	[3]	R/W	Endpoint 3 Interrupt Enable Flag	0
EP2IE	[2]	R/W	Endpoint 2 Interrupt Enable Flag	0
EP1IE	[1]	R/W	Endpoint 1 Interrupt Enable Flag	0
EP0IE	[0]	R/W	Endpoint 0 Interrupt Enable Flag 1 = EP0 Interrupt flag enable 0 = EP0 Interrupt flag disable	0

FUNCTION ADDRESS REGISTER (FAR)

This register holds the address of USB device.

Register	Address	R/W	Description	Reset Value
FAR	0x3880_000C	R	Function Address Register	32 bits

FAR	Bit	R/W	Description	Initial State
	[31:7]		Reserved	
FA	[6:0]	R	MCU can read a unique USB function address from this register. The address is transferred from USB Host through "set_address" command.	

FRAME NUMBER REGISTER (FNR)

The frame number register is used for the isochronous transaction.

Register	Address	R/W	Description	Reset Value
FNR	0x3880_0010	R	Frame Number Register	32 bits

FNR	Bit	R/W	Description	Initial State
	[31:15]		Reserved	
FTL	[14]	R	Frame Timer Lock FTL is activated when the device frame timer is locked to the host frame timer. When this bit is set, Frame Number is valid. This bit is set by USB after device receives two valid SOF.	0
SM	[13]	R	SOF Missing SM is activated when frame timer locking between device frame timer and Host frame timer fails.	0
	[12:11]		Reserved	
FN	[10:0]	R	[10:0] bits store the frame count number, which increases per every SOF packet.	0

ENDPOINT DIRECTION REGISTER (EDR)

USB 2.0 Core supports IN/OUT direction control for each endpoint. This direction can't be changed dynamically. Only by new enumeration, the direction can be altered. Since the endpoint 0 is bi-directional, there is no direction bit assigned to it.

Register	Address	R/W	Description	Reset Value
EDR	0x3880_0014	R/W	Endpoint Direction Register	32 bits

EDR	Bit	R/W	Description	Initial State
	[31:7]		Reserved	
EP6DS	[6]	R/W	Endpoint 6 Direction Select (Only Using OUT Endpoint, so must set to "0")	0
EP5DS	[5]	R/W	Endpoint 5 Direction Select (Only Using IN Endpoint, so must set to "1")	0
EP4DS	[4]	R/W	Endpoint 4 Direction Select (Only Using OUT Endpoint, so must set to "0")	0
EP3DS	[3]	R/W	Endpoint 3 Direction Select (Only Using IN Endpoint, so must set to "1")	0
EP2DS	[2]	R/W	Endpoint 2 Direction Select (Only Using OUT Endpoint, so must set to "0")	0
EP1DS	[1]	R/W	Endpoint 1 Direction Select (Only Using IN Endpoint, so must set to "1")	0
	[0]		Reserved	

TEST REGISTER (TR)

The test register is used for the diagnostics. All bit are activated when 1 is written to and is cleared by 0 on them. Bit[3:0] are for the high speed device only.

Register	Address	R/W	Description	Reset Value
TR	0x3880_0018	R/W	Test Register	32 bits

TR	Bit	R/W	Description	Initial State
	[31:16]		Reserved	
VBUS	[15]	R	0 = Vbus OFF 1 = Vbus ON	0
	[14]		Reserved	
EUERR	[13]	R/C	EB UNDERRUN Error If error interrupt enable bit of SCR register is set to 1, EUERR is set to 1 when EB underrun error in transceiver is detected.	0
PERR	[12]	R/C	PID Error If error interrupt enable bit of SCR register is set to 1, PERR is set to 1 when PID error is detected.	0
	[11:5]		Reserved	
TMD	[4]	R/W	Test Mode. When TMD is set to 1. The core is forced into the test mode. Following TPS, TKS, TJS, TSNS bits are meaningful in test mode.	0
TPS	[3]	R/W	Test Packets. If this bit is set, the USB repetitively transmit the test packets to Host. The test packets are explained in 7.1.20 of USB 2.0 specification. This bit can be set when TMD bit is set.	0
TKS	[2]	R/W	Test K Select. If this bit is set, the transceiver port enters into the high-speed K state. This bit can be set when TMD bit is set.	0
TJS	[1]	R/W	Test J Select. If this bit is set, the transceiver port enters into the high-speed J state. This bit can be set when TMD bit is set.	0
TSNS	[0]	R/W	Test SE0 NAK Select If this bit is set, the transceiver enters into the high speed receive mode and must respond to any IN token with NAK handshake. This bit can be set when TMD bit is set.	0

SYSTEM STATUS REGISTER (SSR)

This register reports operational status of the USB 2.0 Function Core, especially about error status and power saving mode status. Except the line status, every status bits in the System Status Register could be an interrupt sources. When the register is read after an interrupt due to certain system status changes, MCU should write back 1 to the corresponding bits to clear it.

Register	Address	R/W	Description	Reset Value
SSR	0x3880_001C	R/C	Test Register	32 bits

SSR	Bit	R/W	Description	Initial State
	[31:16]		Reserved	
BAERR	[15]	R/C	Byte Align Error If error interrupt enable bit of SCR register is set to 1, BAERR is set to 1 when byte alignment error is detected.	0
TMERR	[14]	R/C	Timeout Error If error interrupt enable bit of SCR register is set to 1, TMERR is set to 1 when timeout error is detected.	0
BSERR	[13]	R/C	Bit Stuff Error If error interrupt enable bit of SCR register is set to 1, BSERR is set to 1 when bit stuff error is detected.	0
TCERR	[12]	R/C	Token CRC Error If error interrupt enable bit of SCR register is set to 1, TCERR is set to 1 when CRC error in token packet is detected.	
DCERR	[11]	R/C	Data CRC Error If error interrupt enable bit of SCR register is set to 1, DCERR is set to 1 when CRC error in data packet is detected.	0
EOERR	[10]	R/C	EB OVERRUN Error If error interrupt enable bit of SCR register is set to 1, EOERR is set to 1 when EB overrun error in transceiver is detected.	0
VBUSOFF	[9]	R/C	VBUS OFF If vbus off interrupt enable bit of SCR register is set to 1, VBUSOFF is set to 1 when VBUS is Low.	0
VBUSON	[8]	R/C	VBUS ON If vbus on interrupt enable bit of SCR register is set to 1, VBUSON is set to 1 when VBUS is High.	0
TBM	[7]	R/C	Toggle Bit Mismatch. If error interrupt enable bit of SCR register is set to 1, TBM is set to 1 when Toggle mismatch is detected.	0
DP	[6]	R	DP Data Line State DP informs the status of D+ Line	0

(Continued)

SSR	Bit	R/W	Description	Initial State
DM	[5]	R	DM Data Line State DM informs the status of D- Line	0
HSP	[4]	R	Host Speed 0 = Full Speed 1 = High Speed	0
SDE	[3]	R/C	Speed Detection End. SDE is set by the core when the HS Detect Handshake process is ended.	0
HFRM	[2]	R/C	Host Forced Resume. HFRM is set by the core in suspend state when host sends resume signaling.	0
HFSUSP	[1]	R/C	Host Forced Suspend HFSUSP is set by the core when the SUSPEND signaling from host is detected.	0
HFRES	[0]	R/C	Host Forced Reset. HFRES is set by the core when the RESET signaling from host is detected.	0

SYSTEM CONTROL REGISTER (SCR)

This register enables top-level control of the core. MCU should access this register for controls such as Power saving mode enable/disable.

Register	Address	R/W	Description	Reset Value
SCR	0x3880_0020	R/W	System Control Register	32 bits

SCR	Bit	R/W	Description	Initial State
	[31:15]		Reserved	
DTZIEN	[14]	R/W	Master Total Counter Zero Interrupt Enable 0 = Disable 1 = Enable When set to 1, DMA total counter zero interrupt is generated.	0
	[13]		Reserved	
DIEN	[12]	R/W	DUAL Interrupt Enable 0 = Disable 1 = Enable When set to 1, Interrupt is activated until Interrupt source is cleared.	0
VBUSOFFEN	[11]	R/W	VBUS OFF Enable 0 = Disable 1 = Enable	0
VBUSONEN	[10]	R/W	VBUS ON Enable 0 = Disable 1 = Enable	0
RWDE	[9]	R/W	Reverse Write Data Enable 0 = Low byte data is first sent to Host 1 = High byte data is first send to Host (We don't support Low byte first, so must set to "1")	1
EIE	[8]	R/W	Error Interrupt Enable This bit must be set to 1 to enable error interrupt.	0
BIS	[7]	R/W	Bus Interface Select, The MCU bus width is selected by BIS. When set to 0, bus width is 8bit. When set to 1, bus width is set to 16bit. (Only Using 16bit mode)	1
SPDEN	[6]	R/W	Speed Detect End Interrupt Enable When set to 1, Speed detection interrupt is generated.	0
RRDE	[5]	R/W	Reverse Read Data Enable 0 = First received byte is loaded in High byte field. 1 = First received byte is loaded in Low byte field. (We don't support High byte first, so must set to "0")	0

(Continued)

SCR	Bit	R/W	Description	Initial State
IPS	[4]	R/W	Interrupt Polarity Select (must write "0", don't set to "1")	0
	[3]		Reserved	0
MFRM	[2]	R/W	Resume by MCU If this bit is set, the suspended core generates a resume signal. This bit is set when MCU writes 1. This bit is cleared when MCU writes 0.	0
HSUSPE	[1]	R/W	Suspend Enable When set to 1, core can respond to the suspend signaling by USB host.	0
HRESE	[0]	R/W	Reset Enable When set to 1, core can respond to the reset signaling by USB host.	0



ELECTRONICS

EP0 STATUS REGISTER (EP0SR)

The test register is used for the diagnostics. All bit are activated when 1 is written to and is cleared by 0 on them. Bit[3:0] are for the high speed device only.

Register	Address	R/W	Description	Reset Value
EP0SR	0x3880_0024	R/W	EP0 Status Register	32 bits

EP0SR	Bit	R/W	Description	Initial State
	[31:7]		Reserved	
LWO	[6]	R	Last Word Odd LWO informs that the last word of a packet in FIFO has an invalid upper byte. This bit is cleared automatically after the MCU reads it from the FIFO.	0
	[5]		Reserved	
SHT	[4]	R/C	Stall Handshake Transmitted SHT informs that STALL handshake due to stall condition is sent to Host. This bit is an interrupt source. This bit is cleared when the MCU writes 1.	0
	[3:2]		Reserved	
TST	[1]	R/C	Tx successfully received TST is set by core after core sends TX data to Host and receives ACK successfully. TST is one of the interrupt sources.	0
RSR	[0]	R/C	Rx successfully received. RSR is set by core after core receives error free packet from Host and sent ACK back to Host successfully. RSR is one of the interrupt sources.	0

EP0 CONTROL REGISTER (EP0CR)

EP0 control register is used for the control of endpoint 0. Controls such as enabling ep0 related interrupts and toggle controls can be handled by EP0 control register.

Register	Address	R/W	Description	Reset Value
EP0CR	0x3880_0028	R/W	EP0 Control Register	32 bits

EP0CR	Bit	R/W	Description	Initial State
	[31:7]		Reserved	
TTE	[3]	R/W	Tx Test Enable This bit can be used for Test. TTE enables MCU to control Tx data toggle bit and Tx zero length set bit. The Tx data toggle bit and Tx zero length set bit do not need to be set in normal operation. 0 = disable 1 = enable	0
TSS	[2]	R/W	Tx Toggle Set. TTS is useful for Test. TTS can be managed when Tx Test Enable (TTE) bit is set. 0 = DATA PID 0 1 = DATA PID 1	0
ESS	[1]	R/W	Endpoint Stall Set ESS is set by MCU when it intends to send STALL handshake to Host. This bit is cleared when the MCU writes 0 on it. ESS is needed to be set 0 after MCU writes 1 on it.	0
TZLS	[0]	R/W	Tx Zero Length Set. TZLS is set by MCU when it intends to send Tx zero length data to Host. TZLS is useful for core Test. TZLS can be managed when Tx Test Enable (TTE) bit is set. This bit is cleared when the MCU writes 0 on it	0

ENDPOINT# BUFFER REGISTER (EP#BR)

The buffer register is used to hold data for TX/RX transfer.

Register	Address	R/W	Description	Reset Value
EP0BR	0x3880_0060	R/W	EP0 Buffer Register	
EP1BR	0x3880_0064	R/W	EP1 Buffer Register	
EP2BR	0x3880_0068	R/W	EP2 Buffer Register	
EP3BR	0x3880_006C	R/W	EP3 Buffer Register	
EP4BR	0x3880_0070	R/W	EP4 Buffer Register	
EP5BR	0x3880_0074	R/W	EP5 Buffer Register	
EP6BR	0x3880_0078	R/W	EP6 Buffer Register	

EP#BR	Bit	R/W	Description	Initial State
	[31:16]		Reserved	
	[15:0]	R/W	Buffer register holds TX/RX data between MCU and the core	

ENDPOINT STATUS REGISTER (ESR)

The endpoint status register reports current status of an endpoint (except EP0) to the MCU

Register	Address	R/W	Description	Reset Value
ESR	0x3880_002C	R/W	Endpoint Status Register	32 bits
ESR	Bit	R/W	Description	Initial State
	[31:16]		Reserved	
FUDR	[15]	R/C	FIFO underflow. FUDR is only used for ISO mode. FUDR is set when FIFO is empty and Host sends IN token. This bit is cleared when the MCU write 1.	0
FOVF	[14]	R/C	FIFO overflow. FOVF is only used for ISO mode. FOVF is set when FIFO is full and Host sends OUT data. This bit is cleared when the MCU writes 1.	
	[13:12]		reserved	
FPID	[11]	R/W	First OUT Packet interrupt Disable in OUT Master operation. First Received OUT packet generates interrupt if this bit is disabled and DEN in Master control register is enabled 0 = Disable 1 = Enable	0
OSD	[10]	R/C	OUT Start Master Operation. OSD is set when First OUT packet is received after Registers related Master Operation are set.	
DTCZ	[9]	R/C	Master Total Count Zero DTCZ is set when Master Operation Total Counter reach to 0. This bit is cleared when the MCU writes 1 on it.	0
SPT	[8]	R/C	Short Packet Received. SPT informs that OUT endpoint receives short packet during OUT Master Operation. This bit is cleared when the MCU writes 1 on it.	
DOM	[7]	R	Dual Operation Mode DOM is set when the max packet size of corresponding endpoint is equal to a half FIFO size. This bit is read only. Endpoint0 does not support dual mode.	

(Continued)

ESR	Bit	R/W	Description	Initial State
FFS	[6]	R/C	FIFOFlushed. FFS informs that FIFO is flushed. This bit is an interrupt source. This bit is cleared when the MCU clears FLUSH bit in Endpoint Control Register.	
FSC	[5]	R/C	Function Stall Condition. FSC informs that STALL handshake due to functional stall condition is sent to Host. This bit is set when endpoint stall set bit is set by the MCU. This bit is cleared when the MCU writes 1 on it.	
LWO	[4]	R	Last Word Odd. LWO informs that the lower byte of last word is only valid. This bit is automatically cleared after the MCU reads packet data received Host.	
PSIF	[3:2]	R	Packet Status In FIFO. 00 = No packet in FIFO 01 = One packet in FIFO 10 = Two packet in FIFO 11 = Invalid value	
TPS	[1]	R/C	Tx Packet Success TPS is used for Single or Dual transfer mode. TPS is activated when one packet data in FIFO was successfully transferred to Host and received ACK from Host. This bit should be cleared by writing 1 on it after being read by the MCU.	
RPS	[0]	R	Rx Packet Success. RPS is used for Single or Dual transfer mode. RPS is activated when the FIFO has a packet data to receive. RPS is automatically cleared when MCU reads all packets (one or two) from FIFO. MCU can identify the packet size through BYTE READ COUNT REGISTER (BRCR).	0

ENDPOINT CONTROL REGISTER (ECR)

The endpoint control register is useful for controlling an endpoint both in normal operation and test case. Putting an endpoint in specific operation mode can be accomplished through the endpoint control register.

Register	Address	R/W	Description	Reset Value
ECR	0x3880_0030	R/W	Endpoint Control Register	32 bits

ECR	Bit	R/W	Description	Initial State
	[31:13]		Reserved	
INPKTHLD	[12]	R/W	The MCU can control Tx FIFO status through this bit. If this bit is set to one, USB does not send IN data to Host. 0 = The USB can send IN data to Host according to IN FIFO status(normal operation) 1 = The USB sends NAK handshake to Host regardless of IN FIFO status.	0
OUTPKTHLD	[11]	R/W	The MCU can control Rx FIFO Status through this bit. If this bit is set to one, USB does not accept OUT data from Host. 0 = The USB can accept OUT data from Host according to OUT FIFO status(normal operation) 1 = The USB does not accept OUT data from Host.	
TNPMF	[10:9]	R/W	Transaction Number Per Micro Frame. TNPMF is useful for ISO transfer. 00 = Invalid value 01 = 1 Transaction Per MicroFrame 10 = 2 Transaction Per MicroFrame 11 = 3 Transaction Per MicroFrame	
IME	[8]	R/W	ISO mode Endpoint IME determines the transfer type of an endpoint. 0 = Bulk(Interrupt) mode 1 = ISO mode	
DUEN	[7]	R/W	Dual FIFO mode Enable 0 = Dual Disable(Single mode) 1 = Dual Enable	
FLUSH	[6]	R/W	FIFO Flush FIFO is flushed when this bit is set to 1. This bit is automatically cleared after MCU writes 1.	
TTE	[5]	R/W	TX Toggle Enable. The MCU can force TX data toggle bit with TTE. This bit is useful for test. The TX data toggle bit changes automatically in normal operation. 0 = Disable 1 = Enable	

(Continued)

ECR	Bit	R/W	Description	Initial State
TTS	[4:3]	R/W	<p>TX Toggle Select</p> <p>TTS is used for test. This is valid when TX Toggle Enable (TTE) is set.</p> <p>00 = DATA PID 0 01 = DATA PID 1 10 = DATA PID 2 (Only in ISO mode) 11 = DATA PID M (Only in ISO mode)</p>	
CDP	[2]	R/W	<p>Clear Data PID</p> <p>In RX mode When this bit is set to 1, data toggle bit in core to be compared with the data PID of received packet is reset to 0. This bit is automatically cleared after MCU writes 1.</p> <p>In TX mode TX data PID to be transmitted to host is reset to 0 when this bit is set to 1. This bit is automatically cleared after MCU writes 1.</p>	
ESS	[1]	R/W	<p>Endpoint Stall Set</p> <p>ESS is set by the MCU when the MCU intends to send STALL handshake to Host.</p> <p>This bit is cleared when the MCU writes 0 in it.</p>	
TZLS	[0]	R/W	<p>TX Zero Length Set.</p> <p>This bit is used for Test.</p> <p>TZLS is set by the MCU when the MCU intends to send zero length TX data to Host.</p> <p>This bit is cleared when the MCU writes 0 in it.</p>	

BYTE READ COUNT REGISTER (BRCR)

The byte read count register keeps byte (half word) counts of a RX packet from USB host.

Register	Address	R/W	Description	Reset Value
BRCR	0x3880_0034	R	Byte Read Count Register	32 bits

BRCR	Bit	R/W	Description	Initial State
	[31:10]		Reserved	
RDCNT	[9:0]	R	<p>FIFO Read Byte Count[9:0] RDCNT is read only. The BRCR inform the amount of received data from host.</p> <p>In 16bit Interface, RDCNT informs the amount of data in half word (16bit) unit. Through the LWO bit of EP0SR, the MCU can determine valid byte in last data word.</p> <p>In 8bit interface, RDCNT keeps the byte size of received data.</p>	

BYTE WRITE COUNT REGISTER (BWCR)

The byte write count register keeps the byte (half word) count value of a TX packet from MCU. The counter value will be used to determine the end of TX packet.

Register	Address	R/W	Description	Reset Value
BWCR	0x3880_0038	R/W	Byte Write Count Register	32 bits

BWCR	Bit	R/W	Description	Initial State
	[31:10]		Reserved	
WRCNT	[9:0]	R/W	Through BWCR, the MCU must load the byte counts of a TX data packet to the core. The core uses this count value to determine the end of packet. The count value to this register must be less than MAXP.	

MAX PACKET REGISTER (MPR)

The byte write count register keeps the byte (half word) count value of a TX packet from MCU. The counter value will be used to determine the end of TX packet.

Register	Address	R/W	Description	Reset Value
MPR	0x3880_003C	R/W	MAX Packet Register	32 bits

MPR	Bit	R/W	Description	Initial State
	[31:11]		Reserved	
MAXP	[10:0]	R/W	MAX Packet [9:0] The max packet size of each endpoint is determined by MAX packet register. The range of max packet is from 0 to 512 bytes. 000_0000_0000 = Max Packet 0 byte. 000_0000_1000 = Max Packet 8 bytes. 000_0001_0000 = Max Packet 16 bytes. 000_0010_0000 = Max Packet 32 bytes. 000_0100_0000 = Max Packet 64 bytes. 000_1000_0000 = Max Packet 128 bytes. 001_0000_0000 = Max Packet 256 bytes. 010_0000_0000 = Max Packet 512 bytes.	

MASTER CONTROL REGISTER (MCR)

The AHB Master Operation is controlled by the programming Master Control Register and Master IF Control Register..

Register	Address	R/W	Description	Reset Value
MCR	0x3880_0040	R/W	Master Control Register	32 bits

MCR	Bit	R/W	Description	Initial State
	[31:6]		Reserved	
ARDRD	[5]	R/W	Auto Rx Master Run set disable. 0 = set 1 = disable This bit is cleared when Master operation is ended.	0
FMDE	[4]	R/W	Burst Mode Enable. This bit is used to run Burst Mode Master Operation. 0 = Burst mode disable 1 = Burst mode enable	
	[3]		Reserved	
TDR	[2]	R/W	Tx Master Operation Run This bit is used to set start Master operation for Tx Endpoint (IN endpoint) 0 = Master operation stop 1 = Master operation run	
RDR	[1]	R/W	Rx Master Operation Run This bit is used to start Master operation for Rx Endpoint (OUT endpoint). This bit is automatically set when USB receives OUT packet data and DEN bit is set to 1 and ARRD RD bit is set to 0. To operate Master operation after OUT packet data received, MCU must set RDR to 1. 0 = Master operation stop. 1 = Master operation run.	
DEN	[0]	R/W	Master Operation Mode Enable This bit is used to set the Master Operation mode 0 = Interrupt Operation mode 1 = Master Operation mode	

MASTER TRANSFER COUNTER REGISTER (MTCR)

The byte write count register keeps the byte (half word) count value of a TX packet from MCU. The counter value will be used to determine the end of TX packet.

Register	Address	R/W	Description	Reset Value
MTCR	0x3880_0044	R/W	Master Transfer Counter Register	32 bits

MTCR	Bit	R/W	Description	Initial State
	[31:16]		Reserved	
MTCR	[15:0]	R/W	To operate single mode transfer, MTCR is needed to be set 16'h0002. In case of Burst mode, the MCU should set max packet value.	

MASTER FIFO COUNTER REGISTER (MFCR)

This register has the byte number of data per Master operation.
The max packet size is loaded in this register.

Register	Address	R/W	Description	Reset Value
MFCR	0x3880_0048	R/W	Master FIFO Counter Register	32 bits

MFCR	Bit	R/W	Description	Initial State
	[31:16]		Reserved	
MFCR	[15:0]	R/W	In case of OUT Endpoint, the size value of received packet will be loaded in this register automatically when Rx Master Run is enabled. In case of IN Endpoint, the MCU should set max packet value.	

MASTER TOTAL TRANSFER COUNTER REGISTER 1/2 (MTTCR 1/2)

This register has the total byte number of data to transfer using Master Interface.
When this counter register value is zero, Master operation is ended.

Register	Address	R/W	Description	Reset Value
MTTCR1	0x3880_004C	R/W	Master Total Transfer Counter Register 1/2	32 bits
MTTCR2	0x3880_0050			

MTTCR#	Bit	R/W	Description	Initial State
	[31:16]		Reserved	
MTTCR	[15:0]	R/W	This register should have total byte size to be transferred using Master Interface. Master Total Transfer Counter1 : Low half word value Master Total Transfer Counter2 : High half word value. The max value is up to 2^{32}	

MASTER INTERFACE COUNTER REGISTER (MICR)

The AHB Master Operation is controlled by the programming Master Control Register and Master IF Control Register..

Register	Address	R/W	Description	Reset Value
MICR	0x3880_0084	R/W	Master Interface Counter Register	32 bits

MICR	Bit	R/W	Description	Initial State
	[31:6]		Reserved	
BLENGTH	[5:2]	R/W	Burst Length 0000 = Single transfer 0011 = 4-beat incrementing burst transfer 0101 = 8-beat incrementing burst transfer 0111 = 16-beat incrementing burst transfer 1110 = 256-beat incrementing burst transfer	0000
DSIZE	[1:0]	R/W	Transfer data size of AHB interface 00 = 16 bit 01 = 32 bit	00

MEMORY BASE ADDRESS REGISTER 1/2 (MBAR1/2)

Register	Address	R/W	Description	Reset Value
MBAR1	0x3880_0088	R/W	Memory Base Address Register1/2	
MBAR2	0x3880_008C			32 bits

MBAR#	Bit	R/W	Description	Initial State
	[31:16]		Reserved	
MBAR	[15:0]	R/W	This register should have memory base address to be transferred using Master Interface. Memory Base Address 1 : Low half word value Memory Base Address [15:0] Memory Base Address 2 : High half word value. Memory Base Address [31:16]	

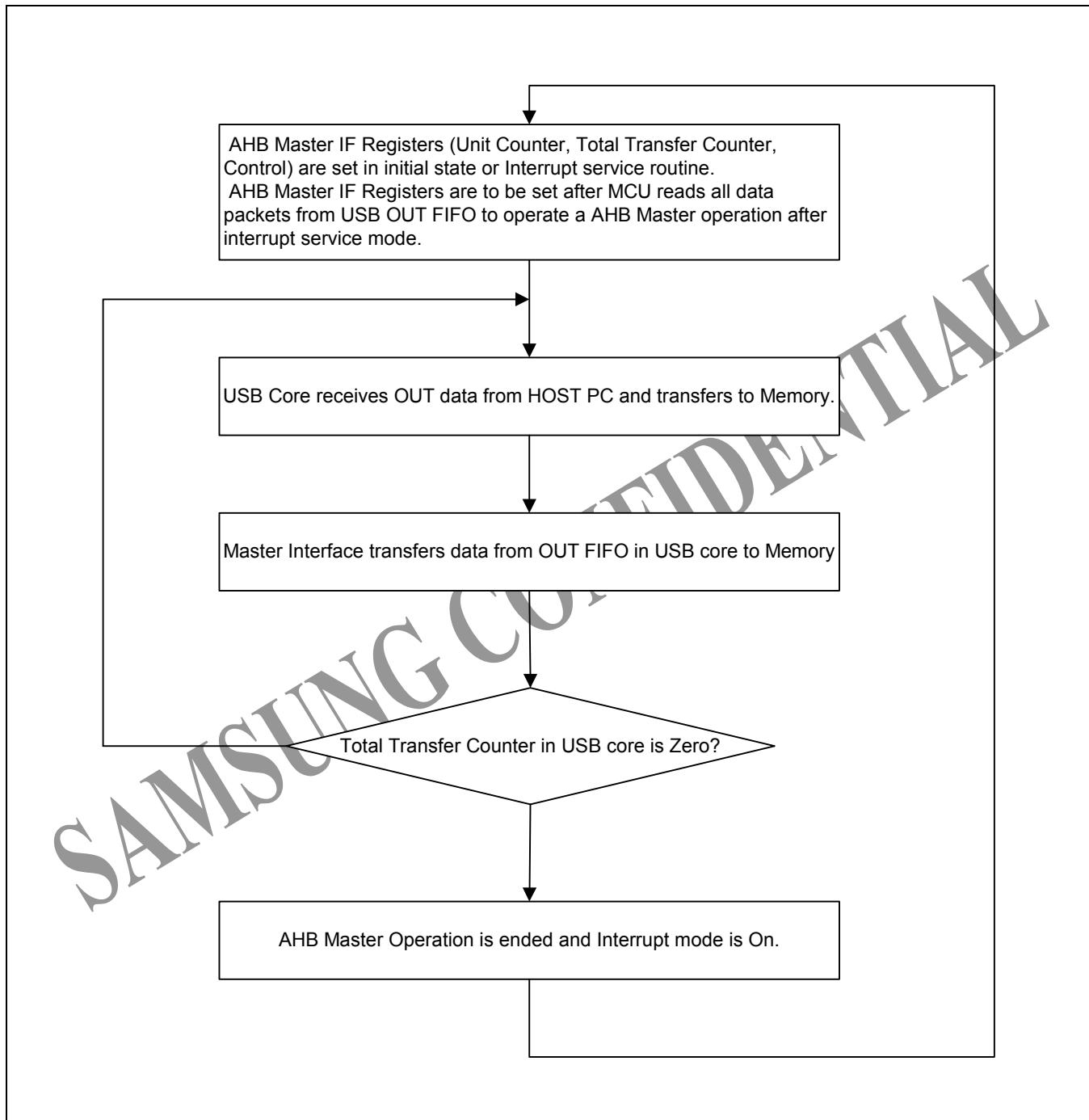
MEMORY CURRENT ADDRESS REGISTER 1/2 (MCAR1/2)

Register	Address	R/W	Description	Reset Value
MCAR1	0x3880_0094	R/W	Memory Current Address Register1/2	32 bits
MCAR2	0x3880_0098			

MCAR#	Bit	R/W	Description	Initial State
	[31:16]		Reserved	
MCAR	[15:0]	R/W	<p>This register should have memory current address to be transferred using Master Interface.</p> <p>Memory Current Address 1 : Low half word value Memory Current Address [15:0]</p> <p>Memory Current Address 2 : High half word value. Memory Current Address [31:16]</p>	

Ahb MASTER OPERATION FLOW CHART

A. OUT Transfer Operation Flow

**Figure 21-3. OUT Transfer Operation Flow**

B. IN Transfer Operation Flow

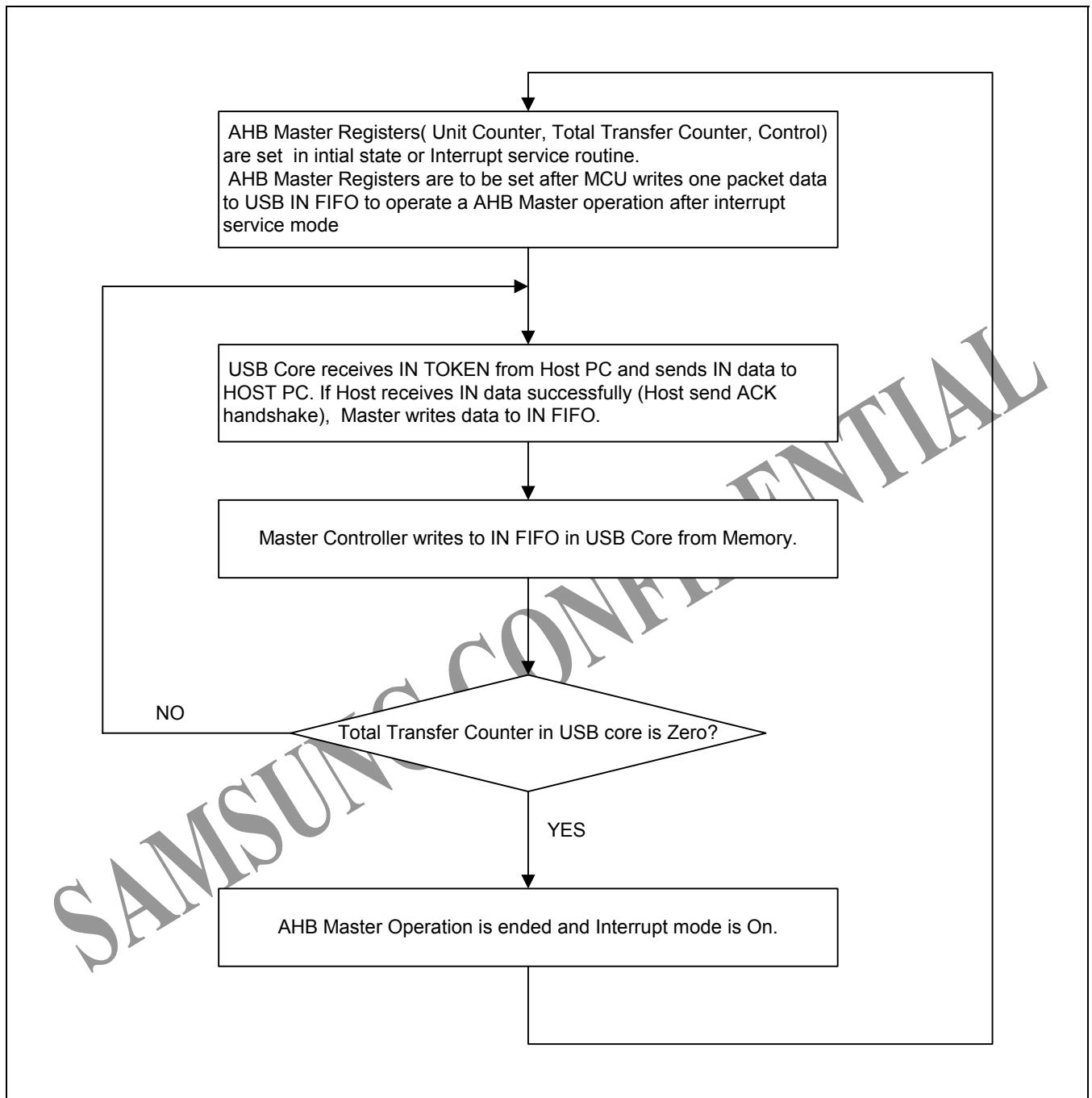


Figure 21-4. IN Transfer Operation Flow

NOTES

SAMSUNG CONFIDENTIAL

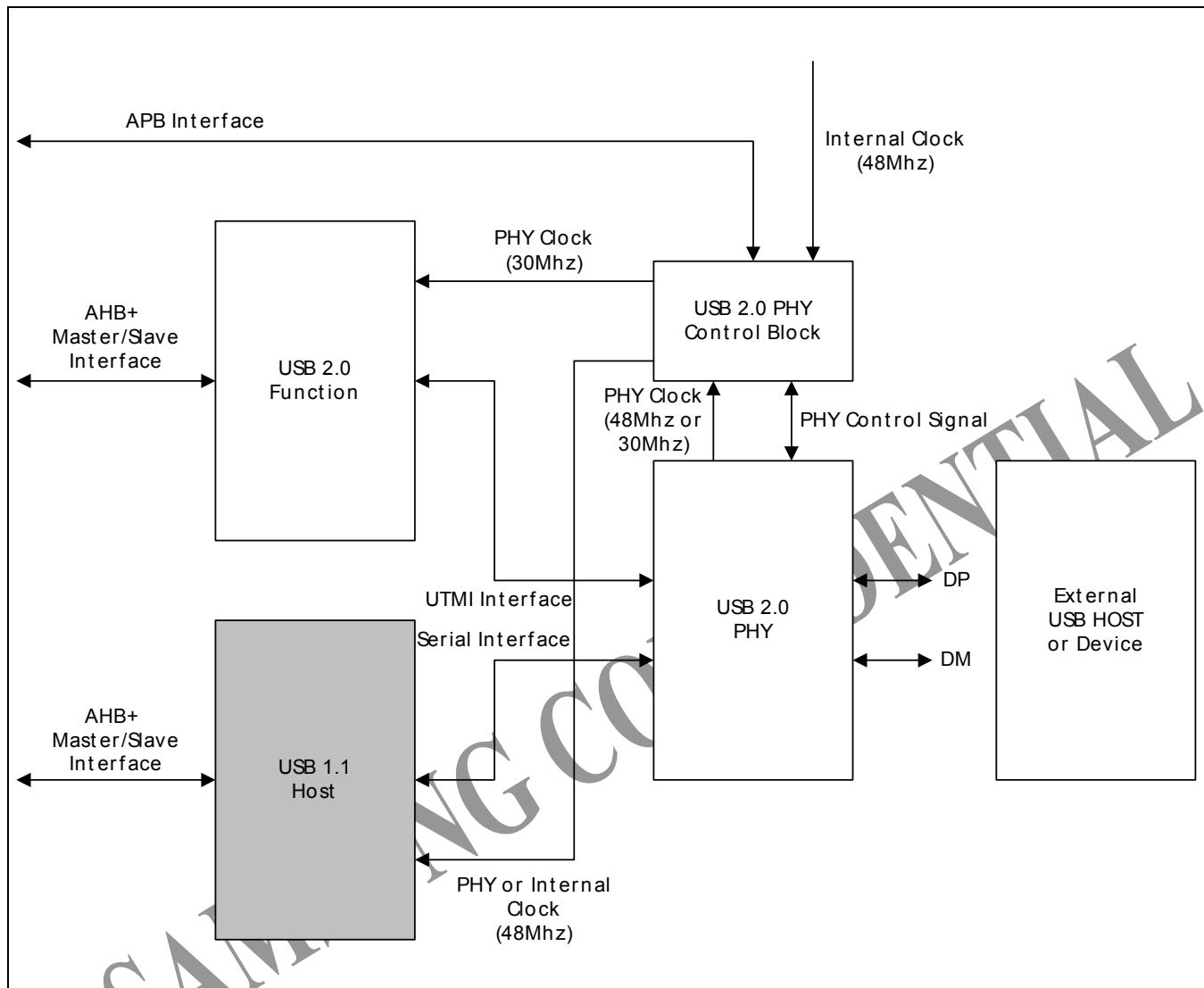
22 USB 1.1 HOST

OVERVIEW

The USB 1.1 Host Controller is Open HCI Rev1.0 compatible device. The controller supports both Low and Full speed mode. It supports two type interface of the standard serial interface and AHB interface. If you want detail specification, see OpenHCI Rev1.0 in “www.usb.org”.

FEATURE

- Open HCI Rev 1.0 compatible
- USB Rev 1.1 compatible
- Support for both Low Speed and Full Speed USB Devices
- Support One Downstream USB Port Transceiver

BLOCK DIAGRAM**Figure 22-1. USB 1.1 Block Diagram**

USB 1.1 HOST CONTROLLER SPECIAL REGISTERS

The Host Controller contains a set of on-chip operational registers which are mapped into a non-cacheable portion of the system addressable space. These registers are used by the Host Controller Driver. According to the function of registers, they are divided into four partitions, specifically for Control and Status, Memory Pointer, Frame Counter and Root Hub. All of the registers should be read and written as Dwords.

Table 22-1. Host Controller Operation Registers

Name	Address
HcRevision	0x3860_0000
HcControl	0x3860_0004
HcCommandStatus	0x3860_0008
HcInterruptStatus	0x3860_000C
HcInterruptEnable	0x3860_0010
HcInterruptDisable	0x3860_0014
HcHCCA	0x3860_0018
HcPeriodCurrentED	0x3860_001C
HcControlHeadED	0x3860_0020
HcControlCurrentED	0x3860_0024
HcBulkHeadED	0x3860_0028
HcBulkCurrentED	0x3860_002C
HcDoneHead	0x3860_0030
HcFmInterval	0x3860_0034
HcFmRemaining	0x3860_0038
HcFmNumber	0x3860_003C
HcPeriodicStart	0x3860_0040
HcLSThreshold	0x3860_0044
HcRhDescriptorA	0x3860_0048
HcRhDescriptorB	0x3860_004C
HcRhStatus	0x3860_0050
HcRhPortStatus	0x3860_0054

NOTES

SAMSUNG CONFIDENTIAL

23

USB 2.0 PHY CONTROL

OVERVIEW

This controller is designed to control USB 2.0 PHY. USB 2.0 PHY in S5L8700X supports two-type interface for host and Function. One is serial interface for Host 1.1, another is UTMI for Function 2.0.

We have S/W reset for each block (host, function and PHY) and select clock that is internal PLL clock or PHY clock for host1.1.

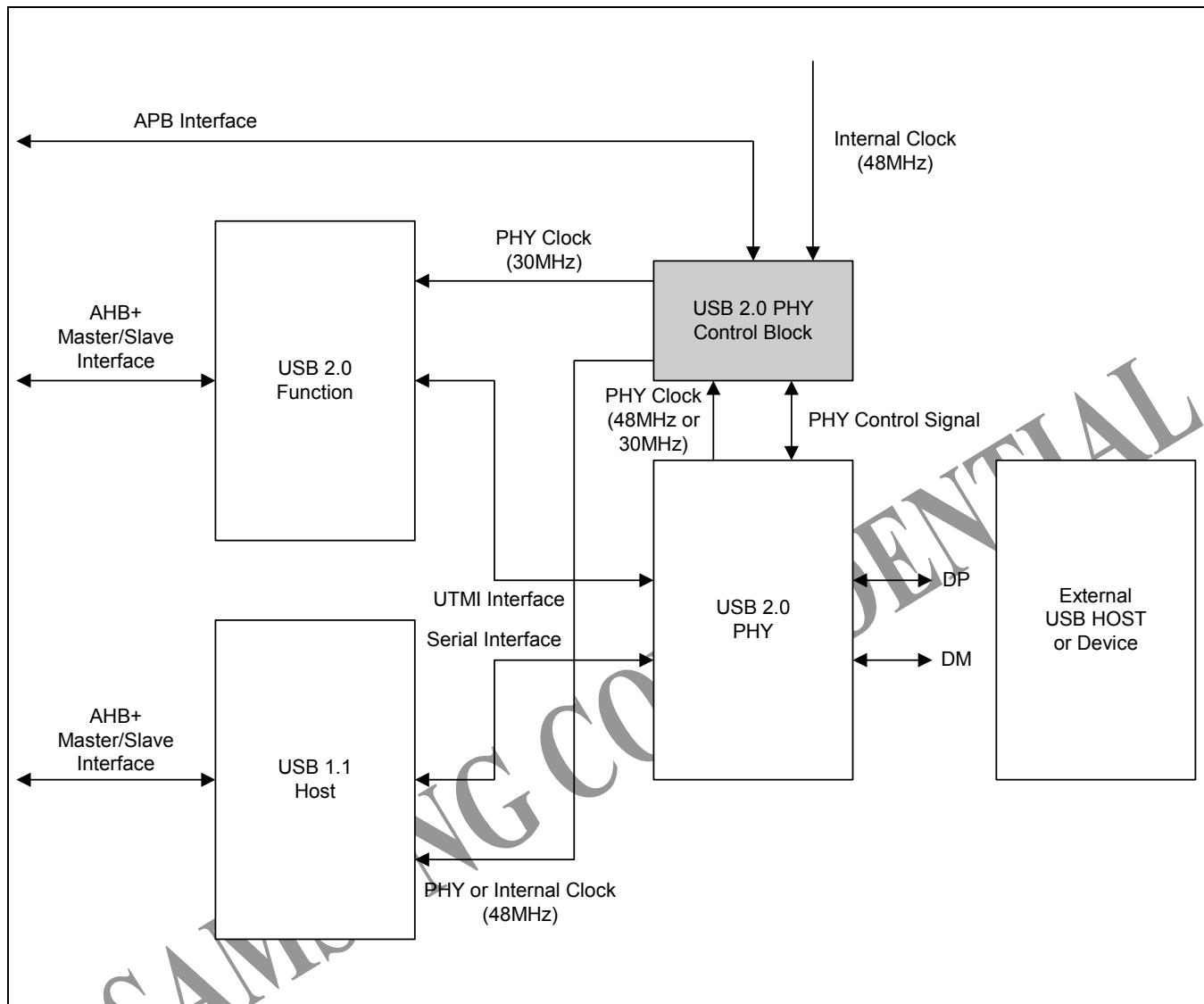
Each USB 2.0 PHY ports has three distinct external interface:

1. USB Data Plus (DP) and Data Minus (DM) Lines
2. USB 2.0 Transceiver Macrocell Interface (UTMI) for USB 2.0 Function
 - The USB 2.0 PHY supports the following modes through the UTMI
 - High Speed(HS)
 - Full Speed(FS)
3. Serial Interface for USB 1.1 Host

This interface supports full-speed and low-speed data transmission rate to and from a controller.

FEATURE

- Supports for 480Mbps HS, 12Mbps FS, and 1.5 Mbps LS data transmission rates.
- Dual (HS/FS) mode device supports.
- USB 1.1 Host and USB 2.0 Function support.

BLOCK DIAGRAM**Figure 23-1. USB2.0 Block Diagram**

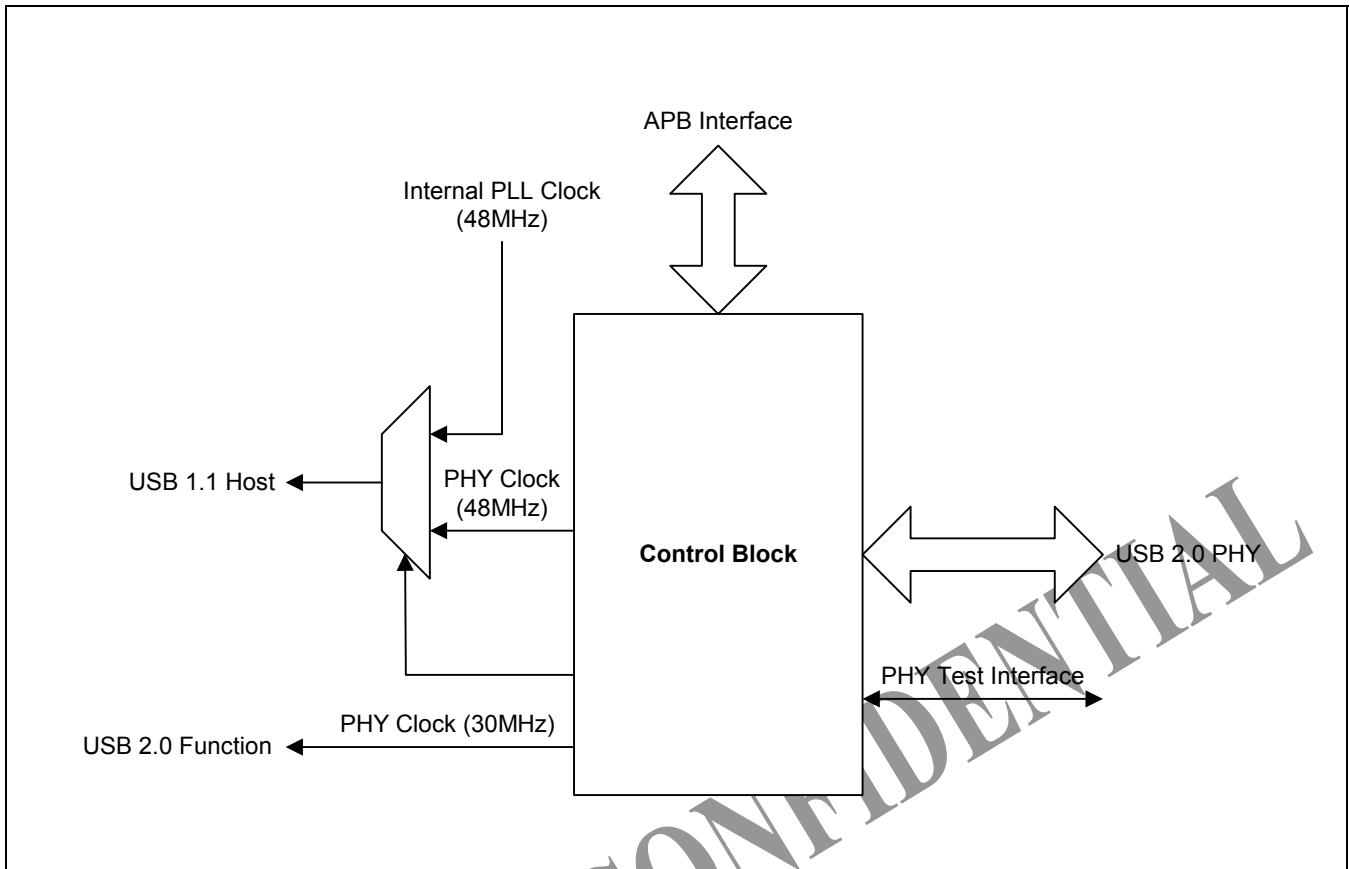


Figure 23-2. USB2.0 PHY Control Block Diagram

REGISTERS

PHY CONTROL REGISTER (PHYCTRL)

Register	Address	R/W	Description	Reset Value
PHYCTRL	0x3C40_0000	R/W	USB2.0 PHY Control Register	32 bits

PHYCTRL	Bit	Description	Initial State
downstream_port	[0]	Downstream Port Select 0 = Device (Function) Mode 1 = Host Mode	0
power_save	[1]	PLL Power down in USB 2.0 PHY 0 = Non Power Save Mode 1 = Power Save mode	0
xo_ext_clk_enbn	[2]	Clock Select for XO Block 1 = The XO Block uses the clock from a crystal 0 = The XO Block uses an external clock supplied on the xo pin	1
xo_refclk_enb	[3]	XO Block is power down 1 = XO Block is powered-up 0 = XO Block is powered-down	1
clk_sel	[5:4]	Reference Clock Frequency Select 11 = Reserved 10 = 48MHz 01 = 24MHz 00 = 12MHz	2'b10
xo_on	[6]	Force XO Block on During a Suspend 1 = This input is inactive, and the XO block is powered down when all ports are suspended 0 = If all ports are suspended, the XO block is powered up	0
int_pll_en	[7]	Host 1.1 uses Internal PLL Clock (48Mhz) 1 = Internal PLL Clock 0 = External X-tal (Only using 48Mhz X-tal) The clk_sel[1:0] bus must be set to 2'b10	0

USB CLOCK CONTROL REGISTER (UCLKCON)

Register	Address	R/W	Description	Reset Value
UCLKCON	0x3C40_0004	R/W	USB Clock Control Register	32 bits

MSINTEN	Bit	Description	Initial State
tclk_en	[0]	USB 2.0 PHY Test Clock Enable 0 = disable 1 = enable	0h
host_clk_en	[1]	USB 1.1 Host Clock Enable 0 = disable 1 = enable	1h
func_clk_en	[2]	USB 2.0 Function Clock Enable 0 = disable 1 = enable	1h

USB RESET CONTROL REGISTER (URSTCON)

Register	Address	R/W	Description	Reset Value
URSTCON	0x3C40_0008	R/W	USB Reset Control Register	32 bits

ADCPARA	Bit	Description	Initial State
phy_reset	[0]	PHY 2.0 S/W Reset (Active Level is Low) The phy_reset signal must be asserted for at least 10us	1
host_reset	[1]	Host 1.1 S/W Reset (Active Level is Low)	1
func_reset	[2]	Function 2.0 S/W Reset (Active Level is Low)	1

NOTES

SAMSUNG CONFIDENTIAL

24 GPIO PORTS

OVERVIEW

S5L8700X has 66 multi-functional GPIO (general-purpose input/output) port pins organized into 9 port groups.

Each port can be easily configured by software to meet various system configuration and design requirements. These multi-functional pins need to be properly configured before their use. If a multiplexed pin is not used as a dedicated functional pin, this pin can be configured as GPIO ports.

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

Each port has different width. P0, P1, P2, P3, P4, P7 has 8 bit width and P10 has 2-bit width.

(P0[7:0], P1[7:0], P2[7:0], P3[7:0], P4[7:0], P5[4:0], P6[4:0], P7[7:0], P10[1:0])

S5L8700X has 44 NOR Flash ports and 45 SDRAM ports.

Each Data ports has 16 bit width and we can use that ports to Normal GPIO if you set to “2'b11” of PCON_ASRAM, PCON_SDRAM. Also we can set that ports to CLCD module port or LCD if module port.

Table 24-1. Port Configuration Overview

Port 0		Selectable Pin functions						
		Function 1	Function 2			Function 3		
Pin Name	Port	GPIO	Function	I/O	Module	Function	I/O	Module
P0_0	P0.0	Input/output	TAOUT	I	Timer	nSSI	I	SPI
P0_1	P0.1	Input/output	TBOUT	O	Timer	SPICLK	I/O	SPI
P0_2	P0.2	Input/output	TCOUT	I	Timer	MOSI	I/O	SPI
P0_3	P0.3	Input/output	TDOUT	I	Timer	MISO	I/O	SPI
P0_4	P0.4	Input/output	TECLK	O	Timer	pll0 out	O	PLL
P0_5	P0.5	Input/output	TCAP	I	Timer	pll1 out	O	PLL
P0_6	P0.6	Input/output	TxD0	O	UART	IISD1_OUT	O	I2S
P0_7	P0.7	Input/output	RxD1	O	UART	IISD2_OUT	O	I2S

Port 1		Selectable Pin functions						
		Function 1	Function 2			Function 3		
Pin Name	Port	GPIO	Function	I/O	Module	Function	I/O	Module
P1_0	P1.0	Input/output	Ext int0	I	ICU			
P1_1	P1.1	Input/output	Ext int1	I	ICU			
P1_2	P1.2	Input/output	Ext int2	I	ICU			
P1_3	P1.3	Input/output	Ext int3	I	ICU	ntrst_adm	I	ADM2E
P1_4	P1.4	Input/output	Ext int4	I	ICU	tck_adm	I	ADM2E
P1_5	P1.5	Input/output	Ext int5	I	ICU	tms_adm	I	ADM2E
P1_6	P1.6	Input/output	Ext int6	I	ICU	tdi_adm	I	ADM2E
P1_7	P1.7	Input/output	Ext int7	I	ICU	tdo_adm	O	ADM2E

Selectable Pin functions					
Function 4			Function 5		
Function	I/O	Module	Function	I/O	Module
adc_puon	O	ADC			
adc_xpon	O	ADC			
adc_xmon	O	ADC			
adc_ypon	O	ADC			
adc_ymon	O	ADC	ext_clk_clcd	I	CLCD

Port 2		Selectable Pin functions						
		Function 1	Function 2			Function 3		
Pin Name	Port	GPIO	Function	I/O	Module	Function	I/O	Module
HD0	P2.0	Input/output	HD[0]	I/O	ATA	LCDD[0]	I/O	LCD if
HD1	P2.1	Input/output	HD[1]	I/O	ATA	LCDD[1]	I/O	LCD if
HD2	P2.2	Input/output	HD[2]	I/O	ATA	LCDD[2]	I/O	LCD if
HD3	P2.3	Input/output	HD[3]	I/O	ATA	LCDD[3]	I/O	LCD if
HD4	P2.4	Input/output	HD[4]	I/O	ATA	LCDD[4]	I/O	LCD if
HD5	P2.5	Input/output	HD[5]	I/O	ATA	LCDD[5]	I/O	LCD if
HD6	P2.6	Input/output	HD[6]	I/O	ATA	LCDD[6]	I/O	LCD if
HD7	P2.7	Input/output	HD[7]	I/O	ATA	LCDD[7]	I/O	LCD if

Selectable Pin functions								
Function 4			Function 5			Function 6		
Function	I/O	Module	Function	I/O	Module	Function	I/O	Module
MSD	I/O	MS	smc_IO0	I/O	SMC	VD[0]	O	CLCD
			smc_IO1	I/O	SMC	VD[1]	O	CLCD
			smc_IO2	I/O	SMC	VD[2]	O	CLCD
			smc_IO3	I/O	SMC	VD[3]	O	CLCD
D4	I/O	MMC	smc_IO4	I/O	SMC	VD[4]	O	CLCD
D5	I/O	MMC	smc_IO5	I/O	SMC	VD[5]	O	CLCD
D6	I/O	MMC	smc_IO6	I/O	SMC	VD[6]	O	CLCD
D7	I/O	MMC	smc_IO7	I/O	SMC	VD[7]	O	CLCD

Port 3		Selectable Pin functions						
		Function 1		Function 2			Function 3	
Pin Name	Port	GPIO	Function	I/O	Module	Function	I/O	Module
HD8	P3.0	Input/output	HD[8]	I/O	ATA	LCDD[8]	I/O	LCD if
HD9	P3.1	Input/output	HD[9]	I/O	ATA	LCDD[9]	I/O	LCD if
HD10	P3.2	Input/output	HD[10]	I/O	ATA	LCDD[10]	I/O	LCD if
HD11	P3.3	Input/output	HD[11]	I/O	ATA	LCDD[11]	I/O	LCD if
HD12	P3.4	Input/output	HD[12]	I/O	ATA	LCDD[12]	I/O	LCD if
HD13	P3.5	Input/output	HD[13]	I/O	ATA	LCDD[13]	I/O	LCD if
HD14	P3.6	Input/output	HD[14]	I/O	ATA	LCDD[14]	I/O	LCD if
HD15	P3.7	Input/output	HD[15]	I/O	ATA	LCDD[15]	I/O	LCD if

Selectable Pin functions								
Function 4			Function 5			Function 6		
Function	I/O	Module	Function	I/O	Module	Function	I/O	Module
D0	I/O	MMC/SDC	smc_IO8	I/O	SMC	VD[8]	O	CLCD
			smc_IO9	I/O	SMC	VD[9]	O	CLCD
D2	I/O	MMC/SDC	smc_IO10	I/O	SMC	VD[10]	O	CLCD
D3	I/O	MMC/SDC	smc_IO11	I/O	SMC	VD[11]	O	CLCD
			smc_IO12	I/O	SMC	VD[12]	O	CLCD
WP	I	MMC/SDC	smc_IO13	I/O	SMC	VD[13]	O	CLCD
			smc_IO14	I/O	SMC	VD[14]	O	CLCD
			smc_IO15	I/O	SMC	VD[15]	O	CLCD

Port 4		Selectable Pin functions						
		Function 1	Function 2			Function 3		
Pin Name	Port		GPIO	Function	I/O	Module	Function	I/O
CBLID	P4.0	Input/output	CBLID	I	ATA			
DMACK	P4.1	Input/output	DMACK	O	ATA	LCDREG	O	LCD if
DMARQ	P4.2	Input/output	DMARQ	I	ATA			
IORDY	P4.3	Input/output	IORDY	I	ATA			
DIOW	P4.4	Input/output	DIOW	O	ATA	LCDnWR	O	LCD if
DIOR	P4.5	Input/output	DIOR	O	ATA	LCDnRD	O	LCD if
ATA_RESET	P4.6	Input/output	ATA_RESET	O	ATA			
ATA_INTRQ	P4.7	Input/output	INTRQ	I	ATA			

Selectable Pin functions								
Function 4			Function 5			Function 6		
Function	I/O	Module	Function	I/O	Module	Function	I/O	Module
						VD[16]	O	CLCD
			CLE	O	SMC	VD[17]	O	CLCD
						VD[18]	O	CLCD
						VD[19]	O	CLCD
MS_BS	O	MS	nWR	O	SMC	VD[20]	O	CLCD
			nRD	O	SMC	VD[21]	O	CLCD
			nWP	O	SMC	VD[22]	O	CLCD
						VD[23]	O	CLCD

Port 5		Selectable Pin functions						
		Function 1		Function 2			Function 3	
Pin Name	Port	GPIO	Function	I/O	Module	Function	I/O	Module
DA0	P5.0	Input/output	DA0	O	ATA			
DA1	P5.1	Input/output	DA1	O	ATA			
DA2	P5.2	Input/output	DA2	O	ATA			
CS0	P5.3	Input/output	CS0	O	ATA			
CS1	P5.4	Input/output	CS1	O	ATA			
VDEN	P5.5	Input/output						
VSYNC	P5.5	Input/output						
HSYNC	P5.6	Input/output						

Selectable Pin functions								
Function 4			Function 5			Function 6		
Function	I/O	Module	Function	I/O	Module	Function	I/O	Module
						VDEN	O	CLCD
CMD	I/O	MMC/SDC				VSYNC	O	CLCD
D1	I/O	MMC/SDC				HSYNC	O	CLCD

Port 6		Selectable Pin functions						
		Function 1	Function 2			Function 3		
Pin Name	Port	GPIO	Function	I/O	Module	Function	I/O	Module
nSMRDY	P6.0	Input/output						
nSMCS0	P6.1	Input/output						
nSMCS1	P6.2	Input/output						
nSMCS2	P6.3	Input/output						
nSMCS3	P6.4	Input/output						
SDMMCLK	P6.5	Input/output						
MSCLK	P6.6	Input/output						
ALE	P6.7	Input/output						

Selectable Pin functions					
Function 4			Function 5		
Function	I/O	Module	Function	I/O	Module
			nf_rbn	I	SMC
			smc_CE0	O	SMC
			smc_CE1	O	SMC
Tx1	O	UART1	smc_CE2	O	SMC
Rx1	I	UART1	smc_CE3	O	SMC
SDMMCLK	O	SDC/MMC			
MSCLK	O	MS	ALE	O	SMC

Port 7		Selectable Pin functions						
		Function 1	Function 2			Function 3		
Pin Name	Port	GPIO	Function	I/O	Module	Function	I/O	Module
LCDnRST	P7.0	Input/output				LCDnRST	O	LCD if
LCDnCS	P7.1	Input/output				LCDnCS	O	LCD if
IISMCK	P7.2	Input/output	IISMCK	O	I2S			
IISWS	P7.3	Input/output	IISWS	O	I2S			
IISBCLK	P7.4	Input/output	IISBCLK	O	I2S			
IISD_OUT	P7.5	Input/output	IISD_OUT	O	I2S			
IISD_IN	P7.6	Input/output	IISD_IN	I	I2S			
SPDIFOUT	P7.7	Input/output	SPDIFOUT	O	SPDIFOU T			

Port 10		Selectable Pin functions						
		Function 1	Function 2			Function 3		
Pin Name	Port	GPIO	Function	I/O	Module	Function	I/O	Module
IICCLK	P10.0	Input/output	IICCLK	I/O	I2C			
IICDAT	P10.1	Input/output	IICDAT	I/O	I2C			

PORT CONTROL DESCRIPTIONS

Port Configuration Register (PCON0, PCON10)

In S5L8700X, most pins are multiplexed, and the PCONn(port control register) determines which function is used for each pin. These ports have 4 different modes. So we can configure 2bit of PCONn to control each port one pin.

For example, If you want to use P0_0 of TA_CK(timer A clock) , you must configure that PCON0[1:0] has 2'b10.

Port Configuration Register (PCON1, PCON2, PCON3, PCON4, PCON5, PCON6, PCON7)

In S5L8700X, most pins are multiplexed. Specially from P1 to P7 has 6 different function. So we can configure 4bit of PCONn to control each port one pin.

For example, If you want to use P2_0 of HD0 (ATAPI data 0) , you must configure that PCON2[3:0] has 4'b0010.

Port Configuration Register (PCON_ASRAM, PCON_SDRAM)

After reset, PCON_ASRAM and PCON_SDRAM has '0' value. Then NOR Flash port works as nor ports and SDRAM port works as sdram ports. These ports have 4 different function. One is it's own port, second function is CLCD port, third function is LCD if port, and fourth function is GPIO.

For example, If you want to use NOR flash ports of CLCD ports, you must configure that PCON_ASRAM has 32'h00000001.

Port Data Register (PDAT0 – PDAT10)

If 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.

External Interrupt Control Register

The 8 external interrupts support various trigger mode: the trigger mode can be configured as falling-edge trigger and rising-edge trigger.

Because each external interrupt pin has an integrated digital noise filter, the interrupt controller can recognize the request signal that lasts longer than 3 clocks.

GPIO PORT SPECIAL FUNCTION REGISTERS

Port 0 Control Registers (PCON0, PDAT0)

Register	Address	R/W	Description	Reset Value
PCON0	0x3CF0 0000	R/W	Configures the pins of port 0	0x00000055
PDAT0	0x3CF0 0004	R/W	The data register for port 0	0xF0

Pin Name	GPIO Name	PCON Bit	Description	
P0_0	P0_0	PCON0[1:0]	00 = Input 10 = TACK(Timer)	01 = Output 11 = Not used
P0_1	P0_1	PCON0[3:2]	00 = Input 10 = TAOOUT(Timer)	01 = Output 11 = Not used
P0_2	P0_2	PCON0[5:4]	00 = Input 10 = TACAP(Timer)	01 = Output 11 = Not used
P0_3	P0_3	PCON0[7:6]	00 = Input 10 = TBCK(Timer)	01 = Output 11 = Not used
P0_4	P0_4	PCON0[9:8]	00 = Input 10 = TBOUT(Timer)	01 = Output 11 = Not used
P0_5	P0_5	PCON0[11:10]	00 = Input 10 = TBCAP(Timer)	01 = Output 11 = Not used
P0_6	P0_6	PCON0[13:12]	00 = Input 10 = PLL0OUT(PII)	01 = Output 11 = Not used
P0_7	P0_7	PCON0[15:14]	00 = Input 10 = PLL1OUT(PII)	01 = Output 11 = Not used

PDAT0	Bit	Description
P0[7:0]	[7:0]	When the port is configured as output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

Port 1 Control Registers (PCON1, PDAT1)

Register	Address	R/W	Description	Reset Value
PCON1	0x3CF0 0010	R/W	Configures the pins of port 1	0x0000 0000
PDAT1	0x3CF0 0014	R/W	The data register for port 1	Undefined

Pin Name	GPIO Name	PCON Bit	Description		
P1_0	P1_0	PCON1[3:0]	0000 = Input 0010 = Ext Interrupt0(ICU) 0100 = Not Used	0001 = Output 0011 = Not Used 0101 = Not Used	
P1_1	P1_1	PCON1[7:4]	0000 = Input 0010 = Ext Interrupt1(ICU) 0100 = Not Used	0001 = Output 0011 = Not Used 0101 = Not Used	
P1_2	P1_2	PCON1[11:8]	0000 = Input 0010 = Ext Interrupt2(ICU) 0100 = Not used	0001 = Output 0011 = Not Used 0101 = Not Used	
P1_3	P1_3	PCON1[15:12]	0000 = Input 0010 = Ext Interrupt3(ICU) 0100 = PUON(ADC)	0001 = Output 0011 = nTRST_ADM 0101 = Not Used	
P1_4	P1_4	PCON1[19:16]	0000 = Input 0010 = Ext Interrupt4(ICU) 0100 = XPON(ADC)	0001 = Output 0011 = TCK_ADM 0101 = Not Used	
P1_5	P1_5	PCON1[23:20]	0000 = Input 0010 = Ext Interrupt5(ICU) 0100 = XMON(ADC)	0001 = Output 0011 = TMS_ADM 0101 = Not Used	
P1_6	P1_6	PCON1[27:24]	0000 = Input 0010 = Ext Interrupt6(ICU) 0100 = YPON(ADC)	0001 = Output 0011 = TDI_ADM 0101 = Not Used	
P1_7	P1_7	PCON1[31:28]	0000 = Input 0010 = Ext Interrupt7(ICU) 0100 = YMON(ADC)	0001 = Output 0011 = TDO_ADM 0101 = Ext Clock(CLCD)	

PDAT1	Bit	Description
P1[7:0]	[7:0]	When the port is configured as output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

Port 2 Control Registers (PCON2, PDAT2)

Register	Address	R/W	Description	Reset Value
PCON2	0x3CF0 0020	R/W	Configures the pins of port 2	0x0000 0000
PDAT2	0x3CF0 0024	R/W	The data register for port 2	Undefined

Pin Name	GPIO Name	PCON Bit	Description	
HD0	P2_0	PCON2[3:0]	0000 = Input 0010 = HD[0] (ATAPI) 0100 = MSD (MS) 0110 = VD[0] (CLCD)	0001 = Output 0011 = LCDD[0] (LCD if) 0101 = smcio[0] (SMC)
HD1	P2_1	PCON2[7:4]	0000 = Input 0010 = HD[1] (ATAPI) 0100 = Not used 0110 = VD[1] (CLCD)	0001 = Output 0011 = LCDD[1] (LCD if) 0101 = smcio[1] (SMC)
HD2	P2_2	PCON2[11:8]	0000 = Input 0010 = HD[2] (ATAPI) 0100 = Not used 0110 = VD[2] (CLCD)	0001 = Output 0011 = LCDD[2] (LCD if) 0101 = smcio[2] (SMC)
HD3	P2_3	PCON2[15:12]	0000 = Input 0010 = HD[3] (ATAPI) 0100 = Not used 0110 = VD[3] (CLCD)	0001 = Output 0011 = LCDD[3] (LCD if) 0101 = smcio[3] (SMC)
HD4	P2_4	PCON2[19:16]	0000 = Input 0010 = HD[4] (ATAPI) 0100 = D4(MMC) 0110 = VD[4] (CLCD)	0001 = Output 0011 = LCDD[4] (LCD if) 0101 = smcio[4] (SMC)
HD5	P2_5	PCON2[23:20]	0000 = Input 0010 = HD[5] (ATAPI) 0100 = D5(MMC) 0110 = VD[5] (CLCD)	0001 = Output 0011 = LCDD[5] (LCD if) 0101 = smcio[5] (SMC)
HD6	P2_6	PCON2[27:24]	0000 = Input 0010 = HD[6] (ATAPI) 0100 = D6(MMC) 0110 = VD[6] (CLCD)	0001 = Output 0011 = LCDD[6] (LCD if) 0101 = smcio[6] (SMC)
HD7	P2_7	PCON2[31:28]	0000 = Input 0010 = HD[7] (ATAPI) 0100 = D7(MMC) 0110 = VD[7] (CLCD)	0001 = Output 0011 = LCDD[7] (LCD if) 0101 = smcio[7] (SMC)

PDAT2	Bit	Description
P2[7:0]	[7:0]	When the port is configured as output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

Port 3 Control Registers (PCON3, PDAT3)

Register	Address	R/W	Description	Reset Value
PCON3	0x3CF0 0030	R/W	Configures the pins of port 3	0x0000 0000
PDAT3	0x3CF0 0034	R/W	The data register for port 3	Undefined

Pin Name	GPIO Name	PCON Bit	Description	
HD8	P3_0	PCON3[3:0]	0000 = Input 0010 = HD[8] (ATAPI) 0100 = D0 (MMC/SDC) 0110 = VD[8] (CLCD)	0001 = Output 0011 = LCDD[8] (LCD if) 0101 = smcio[8] (SMC)
HD9	P3_1	PCON3[7:4]	0000 = Input 0010 = HD[9] (ATAPI) 0100 = Not used 0110 = VD[9] (CLCD)	0001 = Output 0011 = LCDD[9] (LCD if) 0101 = smcio[9] (SMC)
HD10	P3_2	PCON3[11:8]	0000 = Input 0010 = HD[10] (ATAPI) 0100 = D2(SDC) 0110 = VD[10] (CLCD)	0001 = Output 0011 = LCDD[10] (LCD if) 0101 = smcio[10] (SMC)
HD11	P3_3	PCON3[15:12]	0000 = Input 0010 = HD[11] (ATAPI) 0100 = D3(SDC) 0110 = VD[11] (CLCD)	0001 = Output 0011 = LCDD[11] (LCD if) 0101 = smcio[11] (SMC)
HD12	P3_4	PCON3[19:16]	0000 = Input 0010 = HD[12] (ATAPI) 0100 = Not used 0110 = VD[12] (CLCD)	0001 = Output 0011 = LCDD[12] (LCD if) 0101 = smcio[12] (SMC)
HD13	P3_5	PCON3[23:20]	0000 = Input 0010 = HD[13] (ATAPI) 0100 = WP (SDC) 0110 = VD[13] (CLCD)	0001 = Output 0011 = LCDD[13] (LCD if) 0101 = smcio[13] (SMC)
HD14	P3_6	PCON3[27:24]	0000 = Input 0010 = HD[14] (ATAPI) 0100 = Not Used 0110 = VD[14] (CLCD)	0001 = Output 0011 = LCDD[14] (LCD if) 0101 = smcio[14] (SMC)
HD15	P3_7	PCON3[31:28]	0000 = Input 0010 = HD[15] (ATAPI) 0100 = Not Used 0110 = VD[15] (CLCD)	0001 = Output 0011 = LCDD[15] (LCD if) 0101 = smcio[15] (SMC)

PDAT3	Bit	Description
P3[7:0]	[7:0]	When the port is configured as output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

Port 4 Control Registers (PCON4, PDAT4)

Register	Address	R/W	Description	Reset Value
PCON4	0x3CF0 0040	R/W	Configures the pins of port 4	0x0000 0000
PDAT4	0x3CF0 0044	R/W	The data register for port 4	Undefined

Pin Name	GPIO Name	PCON Bit	Description	
CBLID	P4_0	PCON4[3:0]	0000 = Input 0010 = CBLID (ATAPI) 0100 = Not Used 0110 = VD[16] (CLCD)	0001 = Output 0011 = Not Used 0101 = Not Used
DMACK	P4_1	PCON4[7:4]	0000 = Input 0010 = DMACK (ATAPI) 0100 = Not Used 0110 = VD[17] (CLCD)	0001 = Output 0011 = LCDREG 0101 = CLE (SMC)
DMARQ	P4_2	PCON4[11:8]	0000 = Input 0010 = DMARQ (ATAPI) 0100 = Not Used 0110 = VD[18] (CLCD)	0001 = Output 0011 = Not Used 0101 = Not Used
IORDY	P4_3	PCON4[15:12]	0000 = Input 0010 = IORDY (ATAPI) 0100 = Not Used 0110 = VD[19] (CLCD)	0001 = Output 0011 = Not Used 0101 = Not Used
DIOW	P4_4	PCON4[19:16]	0000 = Input 0010 = DIOW (ATAPI) 0100 = MS_BS (MS) 0110 = VD[20] (CLCD)	0001 = Output 0011 = LCDnWR (LCD if) 0101 = nWR (SMC)
DIOR	P4_5	PCON4[23:20]	0000 = Input 0010 = DIOR (ATAPI) 0100 = Not Used 0110 = VD[21] (CLCD)	0001 = Output 0011 = LCDnRD (LCD if) 0101 = nRD (SMC)
ATA_RESET	P4_6	PCON4[27:24]	0000 = Input 0010 = ATA_RESET (ATAPI) 0100 = Not Used 0110 = VD[22] (CLCD)	0001 = Output 0011 = Not Used 0101 = nWP (SMC)
INTRQ	P4_7	PCON4[31:28]	0000 = Input 0010 = INTRQ (ATAPI) 0100 = Not Used 0110 = VD[23] (CLCD)	0001 = Output 0011 = Not Used 0101 = Not Used

PDAT4	Bit	Description
P4[7:0]	[7:0]	When the port is configured as output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

Port 5 Control Registers (PCON5, PDAT5)

Register	Address	R/W	Description	Reset Value
PCON5	0x3CF0 0050	R/W	Configures the pins of port 5	0x0000 0000
PDAT5	0x3CF0 0054	R/W	The data register for port 5	Undefined

Pin Name	GPIO Name	PCON Bit	Description	
DA0	P5_0	PCON5[3:0]	0000 = Input 0010 = DA0 (ATAPI) 0100 = Not Used 0110 = Not Used	0001 = Output 0011 = Not Used 0101 = Not Used
DA1	P5_1	PCON5[7:4]	0000 = Input 0010 = DA1 (ATAPI) 0100 = Not Used 0110 = Not Used	0001 = Output 0011 = Not Used 0101 = Not Used
DA2	P5_2	PCON5[11:8]	0000 = Input 0010 = DA2 (ATAPI) 0100 = Not Used 0110 = Not Used	0001 = Output 0011 = Not Used 0101 = Not Used
CS0	P5_3	PCON5[15:12]	0000 = Input 0010 = CS0 (ATAPI) 0100 = Not Used 0110 = Not Used	0001 = Output 0011 = Not Used 0101 = Not Used
CS1	P5_4	PCON5[19:16]	0000 = Input 0010 = CS1 (ATAPI) 0100 = Not Used 0110 = Not Used	0001 = Output 0011 = Not Used 0101 = Not Used
VDEN	P5_5	PCON5[23:20]	0000 = Input 0010 = Not Used 0100 = Not Used 0110 = VDEN (CLCD)	0001 = Output 0011 = Not Used 0101 = Not Used
VSYNC	P5_6	PCON5[27:24]	0000 = Input 0010 = Not Used 0100 = CMD (SDC) 0110 = VSYNC (CLCD)	0001 = Output 0011 = Not Used 0101 = Not Used
HSYNC	P5_7	PCON5[31:28]	0000 = Input 0010 = Not Used 0100 = D1(int) (SDC) 0110 = HSYNC (CLCD)	0001 = Output 0011 = Not Used 0101 = Not Used

PDAT5	Bit	Description
P5[7:0]	[7:0]	When the port is configured as output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

Port 6 Control Registers (PCON6, PDAT6)

Register	Address	R/W	Description	Reset Value
PCON6	0x3CF0 0060	R/W	Configures the pins of port 6	0x0000 0000
PDAT6	0x3CF0 0064	R/W	The data register for port 6	Undefined

Pin Name	GPIO Name	PCON Bit	Description	
nSMRDY	P6_0	PCON6[3:0]	0000 = Input 0010 = Not Used 0100 = Not Used	0001 = Output 0011 = Not Used 0101 = nf_rbn (SMC)
nSMCS0	P6_1	PCON6[7:4]	0000 = Input 0010 = Not Used 0100 = Not Used	0001 = Output 0011 = Not Used 0101 = smc_ce0 (SMC)
nSMCS1	P6_2	PCON6[11:8]	0000 = Input 0010 = Not Used 0100 = Not Used	0001 = Output 0011 = LCDREG (LCD If) 0101 = smc_ce1 (SMC)
nSMCS2	P6_3	PCON6[15:12]	0000 = Input 0010 = Not Used 0100 = TxD (Uart1)	0001 = Output 0011 = Not Used 0101 = smc_ce2 (SMC)
nSMCS3	P6_4	PCON6[19:16]	0000 = Input 0010 = Not Used 0100 = RxD (Uart1)	0001 = Output 0011 = Not Used 0101 = smc_ce3 (SMC)
SDMMCLK	P6_5	PCON6[23:20]	0000 = Input 0010 = Not Used 0100 = SDMMCLK	0001 = Output 0011 = Not Used 0101 = Not Used
MSCLK	P6_6	PCON6[27:24]	0000 = Input 0010 = Not Used 0100 = MSCLK	0001 = Output 0011 = Not Used 0101 = Not Used
ALE	P6_7	PCON6[31:28]	0000 = Input 0010 = Not Used 0100 = Not Used	0001 = Output 0011 = Not Used 0101 = ALE (SMC)

PDAT6	Bit	Description
P6[7:0]	[7:0]	When the port is configured as output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

Port 7 Control Registers (PCON7, PDAT7)

Register	Address	R/W	Description	Reset Value
PCON7	0x3CF0 0070	R/W	Configures the pins of port 7	0x00222200
PDAT7	0x3CF0 0074	R/W	The data register for port 7	Undefined

Pin Name	GPIO Name	PCON Bit	Description		
LCDnRST	P7_0	PCON7[3:0]	0000 = Input 0010 = Not Used 0100 = Not Used	0001 = Output 0011 = LCDnRST (LCDif) 0101 = Not Used	
LCDnCS	P7_1	PCON7[7:4]	0000 = Input 0010 = Not Used 0100 = Not Used	0001 = Output 0011 = LCDnCS (LCDif) 0101 = Not Used	
IISMCK	P7_2	PCON7[11:8]	0000 = Input 0010 = IISMCK (IIS) 0100 = Not Used	0001 = Output 0011 = Not Used 0101 = Not Used	
IISWS	P7_3	PCON7[15:12]	0000 = Input 0010 = IISWS (IIS) 0100 = Not Used	0001 = Output 0011 = Not Used 0101 = Not Used	
IISBCLK	P7_4	PCON7[19:16]	0000 = Input 0010 = IISBCLK (IIS) 0100 = Not Used	0001 = Output 0011 = Not Used 0101 = Not Used	
IISD_OUT	P7_5	PCON7[23:20]	0000 = Input 0010 = IISD_OUT (IIS) 0100 = Not Used	0001 = Output 0011 = Not Used 0101 = Not Used	
IISD_IN	P7_6	PCON7[27:24]	0000 = Input 0010 = IISD_IN (IIS) 0100 = Not Used	0001 = Output 0011 = Not Used 0101 = Not Used	
SPDIFOUT	P7_7	PCON7[31:28]	0000 = Input 0010 = SPDIFOut (SPDIF) 0100 = Not Used	0001 = Output 0011 = Not Used 0101 = Not Used	

PDAT7	Bit	Description
P7[7:0]	[7:0]	When the port is configured as output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

Port 10 Control Registers (PCON10, PDAT10)

Register	Address	R/W	Description	Reset Value
PCON10	0x3CF0 00A0	R/W	Configures the pins of port 10	0x0000 0000
PDAT10	0x3CF0 00A4	R/W	The data register for port 10	Undefined

Pin Name	GPIO Name	PCON Bit	Description
IICCLK	P10_0	PCON10[1:0]	00 = Input 10 = IICCLK (IIC) 01 = Output 11 = Not used
IICDAT	P10_1	PCON10[3:2]	00 = Input 10 = IICDAT (IIC) 01 = Output 11 = Not used

PDAT10	Bit	Description
P10[1:0]	[1:0]	When the port is configured as output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

Port ASRAM, SDRAM control Registers (PCON_ASRAM, PCON_SDRAM)

Register	Address	R/W	Description	Reset Value
PCON_ASRAM	0x3CF0 00F0	R/W	Configures the pins of port nor flash	0x00000000
PCON_SDRAM	0x3CF0 00F4	R/W	Configures the pins of port sram	0x00000000
PCON11	0x3CF0 00F8	R/W	Configures the pins of port 11	0x00000000
PDAT11	0x3CF0 00FC	R/W	The data register for port 11	Undefined

PCON_ASRAM	0x00000000	0x00000001	0x00000002	0x00000003
Port	Function 0	Function 1 (CLCD)	Function 2(LCD if)	Function 3(GPIO 11)
SDA8	SDA8	VCLK		
SDA9	SDA9	VDEN		
SDA10	SDA10	VSYNC		
SDA11	SDA11	H SYNC		
SDA12	SDA12	D16		
SDA13	SDA13	D17		
SDA14	SDA14	D18		
SDA15	SDA15	D19		
SDA16	SDA16	D20		
SDA17	SDA17	D21		
SDA18	SDA18	D22		
SDA19	SDA19	D23		
SDA20	SDA20	nWR		
SDA21	SDA21	nRD		
SDA22	SDA22	REG		
SDB0	SDB0	D0	D0	PDAT11[0]
SDB1	SDB1	D1	D1	PDAT11[1]
SDB2	SDB2	D2	D2	PDAT11[2]
SDB3	SDB3	D3	D3	PDAT11[3]
SDB4	SDB4	D4	D4	PDAT11[4]
SDB5	SDB5	D5	D5	PDAT11[5]
SDB6	SDB6	D6	D6	PDAT11[6]
SDB7	SDB7	D7	D7	PDAT11[7]
SDB8	SDB8	D8	D8	PDAT11[8]
SDB9	SDB9	D9	D9	PDAT11[9]
SDB10	SDB10	D10	D10	PDAT11[10]
SDB11	SDB11	D11	D11	PDAT11[11]
SDB12	SDB12	D12	D12	PDAT11[12]
SDB13	SDB13	D13	D13	PDAT11[13]
SDB14	SDB14	D14	D14	PDAT11[14]
SDB15	SDB15	D15	D15	PDAT11[15]

NOTES

SAMSUNG CONFIDENTIAL

25

UART

OVERVIEW

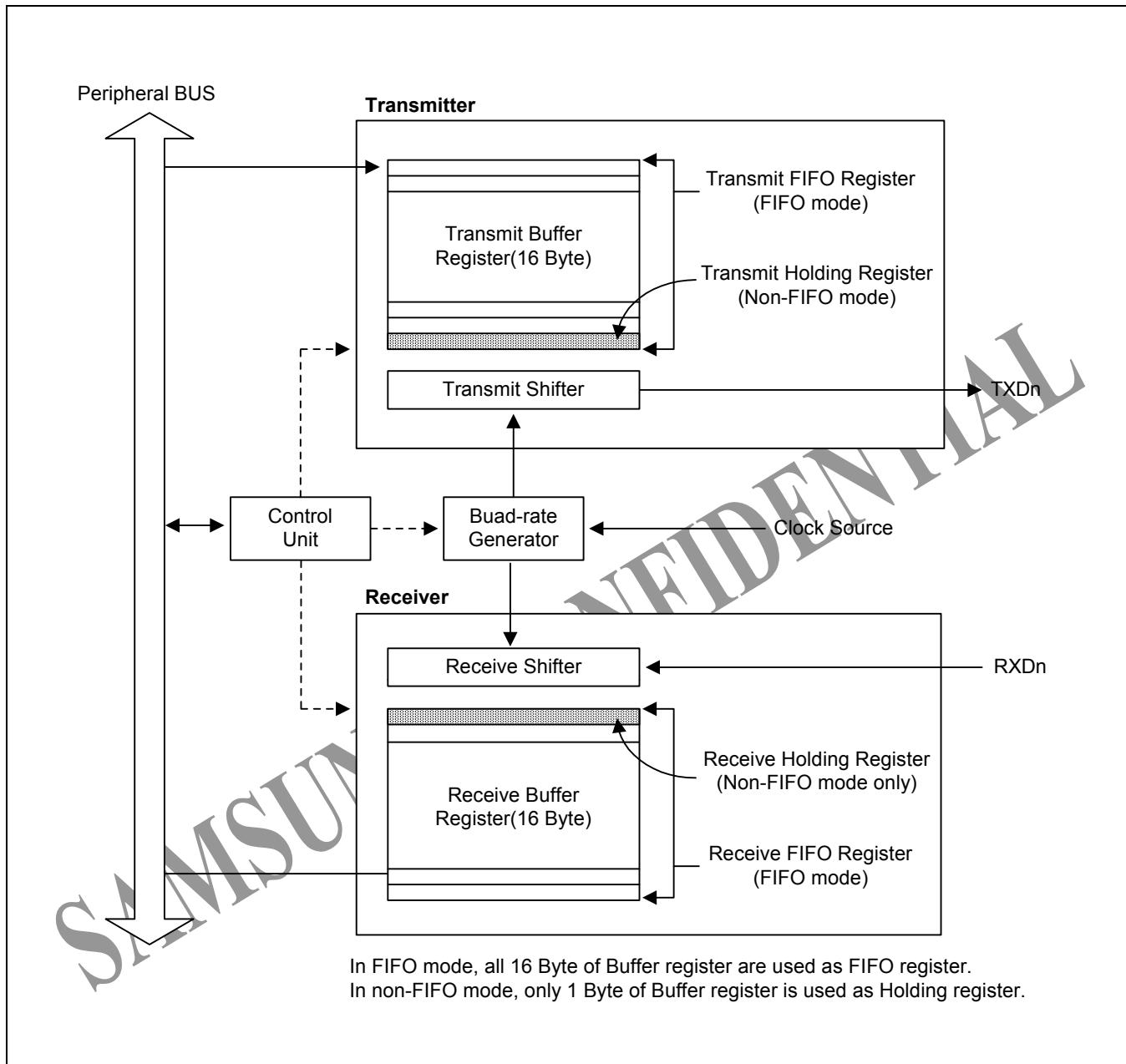
The S5L8700X UART (Universal Asynchronous Receiver and Transmitter) unit provides two independent asynchronous serial I/O ports, each of which can operate in interrupt-based or DMA-based mode. In other words, UART can generate an interrupt or DMA request to transfer data between CPU and UART. It can support bit rates up to 115.2Kbps with the system clock. If external device provides UART with UCLK, then UART can operates at higher speed. Each UART channel contains two 16-byte FIFOs for receive and transmit.

The 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 Figure 25-1. The baud-rate generator can be clocked by system clock or external clock. The transmitter and the receiver contain 16-byte FIFOs and data shifters. Data to be transmitted is written to FIFO and then copied to the transmit shifter. It is then shifted out by the transmit data pin (TxD). The received data is shifted from the receive data pin (RxD), and then copied to FIFO from the shifter.

FEATURES

- Configurable baud-rate up to 115.2 Kbps
- Transmit/Receive with DMA-based or interrupt-based operation
- Internal 16-byte FIFOs for receiver/transmitter

BLOCK DIAGRAM**Figure 25-1. UART Block Diagram with FIFO**

PIN DESCRIPTION

Pin Name	Width	I/O	Description
GnRESET	1	I	Global reset
GCLK	1	I	Global clock (PCLK)
EXT_SCLK	1	I	External reference clock for Tx/Rx (Not used)
APB Interface			
PSEL	1	I	Selection in APB
PENABLE	1	I	Enable in APB
PWRITE	1	I	Write/Read in APB
PADDR	16	I	Address in APB
PWDATA	16	I	Write data in APB
PRDATA	32	O	Read data in APB
I/O DMA Interface			
PDACK	4	I	DMA acknowledgement PDACK[0] PDACK[1] PDACK[2] PDACK[3] not used DMA ch0_ack3 port for UART(Tx) not used not used
PDREQ	4	O	DMA request PDREQ[0] PDREQ[1] PDREQ[2] PDREQ[3] not used DMA ch0_req3 port for UART(Tx) not used not used
UART Interface			
RXD	1	I	Serial data input to UART
nCTS	1	I	not used
TXD	1	O	Serial data output from UART
nRTS	1	O	not used
Interrupt Interface			
INTR	2	O	Interrupt
MISC Interface			
BIG	1	I	Big endian (1) / little endian (0) (Big endian used)
STMODE	1	I	Scan test mode

REGISTERS

UART0

Name	Width	Address	R/W	Description	Reset
Registers					
ULCON	32	0x3CC0_0000	R/W	Line Control Register	0x0000 0000
UCON	32	0x3CC0_0004	R/W	Control Register	0x0000 0000
UFCON	32	0x3CC0_0008	R/W	FIFO Control Register	0x0000 0000
UMCON	32	0x3CC0_000C	R/W	Modem Control Register	0x0000 0000
UTRSTAT	32	0x3CC0_0010	R/W	Tx/Rx Status Register	0x0000 0006
UERSTAT	32	0x3CC0_0014	R/W	Rx Error Status Register	0x0000 0000
UFSTAT	32	0x3CC0_0018	R/W	FIFO Status Register	0x0000 0000
UMSTAT	32	0x3CC0_001C	R/W	Modem Status Register	0x0000 0000
UTXH	32	0x3CC0_0020	R/W	Transmit Buffer Register	–
URXH	32	0x3CC0_0024	R/W	Receive Buffer Register	–
UBRDIV	32	0x3CC0_0028	R/W	Baud Rate Divisor Register	–

UART1

Name	Width	Address	R/W	Description	Reset
Registers					
ULCON	32	0x3CC0_8000	R/W	Line Control Register	0x0000 0000
UCON	32	0x3CC0_8004	R/W	Control Register	0x0000 0000
UFCON	32	0x3CC0_8008	R/W	FIFO Control Register	0x0000 0000
UMCON	32	0x3CC0_800C	R/W	Modem Control Register	0x0000 0000
UTRSTAT	32	0x3CC0_8010	R/W	Tx/Rx Status Register	0x0000 0006
UERSTAT	32	0x3CC0_8014	R/W	Rx Error Status Register	0x0000 0000
UFSTAT	32	0x3CC0_8018	R/W	FIFO Status Register	0x0000 0000
UMSTAT	32	0x3CC0_801C	R/W	Modem Status Register	0x0000 0000
UTXH	32	0x3CC0_8020	R/W	Transmit Buffer Register	–
URXH	32	0x3CC0_8024	R/W	Receive Buffer Register	–
UBRDIV	32	0x3CC0_8028	R/W	Baud Rate Divisor Register	–

UART Line Control Register (ULCON)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits	Name	Type	Description	Reset
[31:7]			Reserved	0
[6]	Infra-Red Mode	R/W	IR_MODE determines whether or not to use the Infra-Red mode. 0 = Normal operation mode 1 = Infra-Red Tx/Rx mode	0
[5:3]	Parity Mode	R/W	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
[2]	Number of Stop Bit	R/W	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
[1:0]	Word Length	R/W	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 (UCON)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Bits	Name	Type	Description	Reset
[31:16]			Reserved	0
[15]	Modem Interrupt Enable	R/W	Modem interrupt enable 0 = Disable 1 = Enable	0
[14]	Error Interrupt Enable	R/W	Error interrupt enable 0 = Disable 1 = Enable	0
[13]	Transmit Interrupt Enable	R/W	Transmit interrupt enable 0 = Disable 1 = Enable	0
[12]	Receive Interrupt Enable	R/W	Receive interrupt enable 0 = Disable 1 = Enable	0
[11]			Reserved	0
[10]	Clock Selection	R/W	Select PCLK or UCLK for the UART baud rate 0 = PCLK - UBRDIVn = (int) (PCLK / (bps x 16)) – 1 1 = UCLK - UBRDIVn = (int) (UCLK / (bps x 16)) - 1	0
[9:8]			Reserved	00
[7]	Rx Time Out Enable	R/W	Enable/disable Rx time out interrupt when UART FIFO is enabled. The interrupt is a receive interrupt. 0 = Disable 1 = Enable	0
[5]	Loop-Back Mode	R/W	Loop back mode. 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
4	Send Break Signal	R/W	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
[3:2]	Transmit Mode (NOTE)	R/W	These two bits determine which function is currently able to write Tx data to the UART transmit buffer register. 00 = Disable 01 = Interrupt request or polling mode 10 = Not defined 11 = DMA ch0 request	0

(Continued)

Bits	Name	Type	Description	Reset
[1:0]	Receive Mode (NOTE)	R/W	These two bits determine which function is currently able to read data from the UART receive buffer register. 00 = Disable 01 = Interrupt request or polling mode 10 = Not used 11 = Not defined	0

NOTE: Before starting the DMA operation, it is recommended first to disable the UART Tx/Rx mode. This will initialize the internal state of the signals for IO DMA interface.

UART FIFO Control Register (UFCON)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Bits	Name	Type	Description	Reset
[31:8]			Reserved	0
[7:6]	Tx FIFO Trigger Level	R/W	These two bits determine the trigger level of the transmit FIFO. 00 = Empty 01 = 4-byte 10 = 8-byte 11 = 12-byte	00
[5:4]	Rx FIFO Trigger Level	R/W	These two bits determine the trigger level of the receive FIFO 00 = 4-byte 01 = 8-byte 10 = 12-byte 11 = 16-byte	00
[3]			Reserved	0
[2]	Tx FIFO Reset	R/W	This bit is auto-cleared after resetting the FIFO 0 = Normal 1 = Tx FIFO reset	0
[1]	Rx FIFO Reset	R/W	This bit is auto-cleared after resetting the FIFO 0 = Normal 1 = Rx FIFO reset	0
[0]	FIFO Enable	R/W	0 = FIFO disable 1 = FIFO enable	0

UART Modem Control Register (UMCON)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits	Name	Type	Description	Reset
[31:8]			Reserved	0
[7:5]	Reserved	–	These bits must be 0's	000
[4]	AFC (Auto Flow Control)	R/W	0 = Disable the AFC 1 = Enable the AFC	0
[3:1]			Reserved	000
[0]	Request to Send	R/W	If AFC bit is enabled, this value will be ignored. In this case the core 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 FIFO Status Register (UFSTAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Bits	Name	Type	Description	Reset
[31:10]			Reserved	0
[9]	Tx FIFO Full	R	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
[8]	Rx FIFO Full	R	This bit is automatically set to 1 whenever receive FIFO is full during transmit operation 0 = 0-byte <= Rx FIFO data <= 15-byte 1 = Full	0
[7:4]	Tx FIFO Count	R	Number of data in Tx FIFO. You should also check the Tx FIFO Full flag to determinate between the full state and the empty state.	0000
[3:0]	Rx FIFO Count	R	Number of data in Rx FIFO. You should also check the Rx FIFO Full flag to determinate between the full state and the empty state.	0000

UART Tx/Rx Status Register (UTRSTAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits	Name	Type	Description	Reset
[31:8]			Reserved	0
[7]	Modem Interrupt Status	R/W	Read 0 = No modem interrupt occurs 1 = Modem interrupt occurs Write 0 = Nothing occurs 1 = Clear this status flag	0
[6]	Error Interrupt Status	R/W	Read 0 = No error interrupt occurs 1 = Modem interrupt occurs Write 0 = Nothing occurs 1 = Clear this status flag	0
[5]	Transmit Interrupt Status	R/W	Read 0 = No transmit interrupt occurs 1 = Modem interrupt occurs Write 0 = Nothing occurs 1 = Clear this status flag	0
[4]	Receive Interrupt Status	R/W	Read 0 = No receive interrupt occurs 1 = Modem interrupt occurs Write 0 = Nothing occurs 1 = Clear this status flag	0
[3]			Reserved	0
[2]	Transmitter Empty	R	This bit is automatically set to 1 when the transmit buffer register has no valid data to transmit and the transmit shift register is empty. 0 = Not empty 1 = Transmitter is empty	0

(Continued)

Bits	Name	Type	Description	Reset
[1]	Transmit Buffer Empty	R	<p>This bit is automatically set to 1 when transmit buffer register is empty.</p> <p>0 = The buffer register is not empty</p> <p>1 = Empty (In Non-FIFO mode, interrupt or DMA is requested. In FIFO mode, interrupt or DMA is requested, when Tx FIFO Trigger Level is set to 00).</p> <p>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.</p>	0
[0]	Receive Buffer Data Ready	R	<p>This bit is automatically set to 1 when receive buffer register is ready.</p> <p>0 = The buffer register is empty</p> <p>1 = The buffer register has a received data (In Non-FIFO mode, interrupt or DMA is requested)</p> <p>If the UART uses the FIFO, users should check Rx FIFO count bits and Rx FIFO full bit in the UFSTAT register instead of this bit.</p>	0

UART Error Status Register (UERSTAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits	Name	Type	Description	Reset
[31:4]			Reserved	0
[3]	Break Detect	R	This bit is automatically set to 1 to indicate that a break signal has been received 0 = No break receive 1 = Break receive (Interrupt is requested)	0
[2]	Frame Error	R	This bit is automatically set to 1 whenever a frame error occurs during receive operation 0 = No frame error during receive 1 = Frame error (Interrupt is requested)	0
[1]	Parity Error	R	This bit is automatically set to 1 whenever a parity error occurs during receive operation 0 = No parity error during receive 1 = Parity error (Interrupt is requested)	0
[0]	Overrun Error	R	This bit is automatically set to 1 whenever an overrun error occurs during receive operation 0 = No overrun error during receive 1 = Overrun error (Interrupt is requested)	0

NOTE: These bits are automatically cleared to zero when UERSTAT register is read.

UART Modem Status Register (UMSTAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Bits	Name	Type	Description	Reset
[31:5]			Reserved	0
[4]	Delta CTS	R	This bit indicates that the nCTS input to core has changed state since the last time it was read by CPU. 0 = Has not changed 1 = Has changed	0
[3:1]			Reserved	0
[0]	Clear to Send	R	0 = CTS signal is not activated (nCTS pin is high) 1 = CTS signal is activated (nCTS pin is low)	0

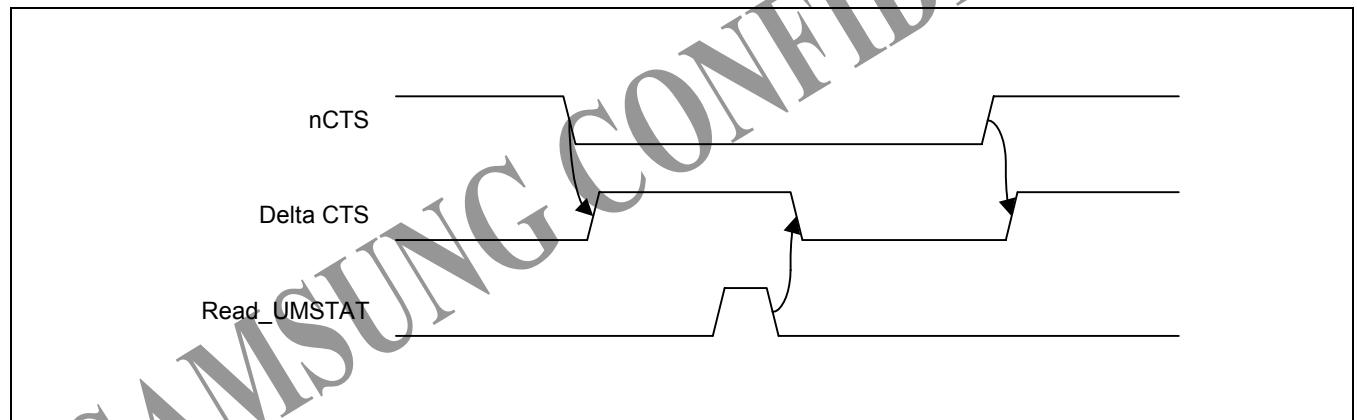


Figure 25-2. nCTS and Delta CTS Timing Diagram

UART Transmit Buffer Register (UTXH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDATA															

Bits	Name	Type	Description													Reset
[31:8]			Reserved													0
[7:0]	TXDATA	W	Transmit data for UART													0

UART Receive Buffer Register (URXH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits	Name	Type	Description													Reset
[31:8]			Reserved													0
[7:0]	RXDATA	R	Received data for UART													0

UART Baud Rate Divisor Register (UBRDIV)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UBRDIV															

Bits	Name	Type	Description													Reset
[31:16]			Reserved													0
[15:0]	UBRDIV	R/W	Baud rate division value UBRDIV > 0													0

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 (ULCON). The transmitter can also produce the break condition. The break condition forces the serial output to logic 0 state for one frame transmission time. This block transmits break signal after the present transmission word transmits perfectly. After the break signal transmission, it continuously transmits data into the Tx FIFO (Tx holding register in the case of Non-FIFO mode).

Data Reception

Like the transmission, the data frame for reception is also programmable. It consists of a start bit, 6 to 8 data bits, an optional parity bit and 1 to 2 stop bits in the line control register (ULCON). 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 RxD input is held in the logic 0 state for longer than one frame transmission time.

Receive time-out condition occurs when it does not receive data during 3 word time (This interval follows the setting of word length bit) and the Rx FIFO is not empty in the FIFO mode.

Auto Flow Control (AFC)

UART0 and UART1 support auto flow control with nRTS and nCTS signals. It would have to connect UART to UART. If users connect UART to a Modem, disable auto flow control bit in UMCN 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 URT'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).

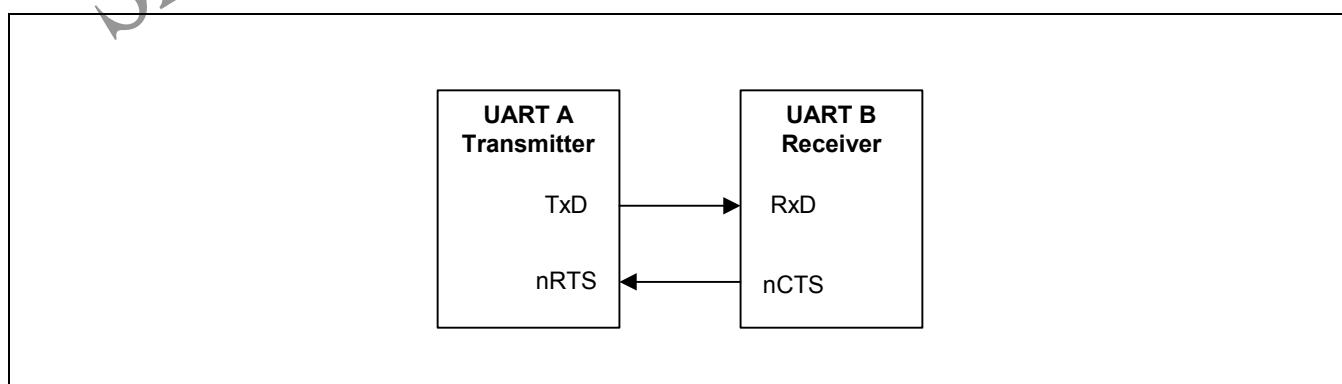


Figure 25-3. UART AFC Interface

Non Auto-Flow control (Controlling nRTS and nCTS by software) Example

Ex) Rx operation with FIFO

1. Select receive mode (interrupt or DMA mode)
2. Check the value of Rx FIFO count in UFSTAT register. If the value is less than 15, users have to set the value of UMCON [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.

Ex) Tx operation with FIFO

1. Select transmit mode (Interrupt or DMA mode)
2. Check the value of UMSTAT [0]. If the value is '1' (nCTS is activated), users write the data to Tx FIFO register.

RS-232C Interface

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

Interrupt/DMA Request Generation

Each UART has seven status (Tx/Rx/Error) flags: Overrun error, Parity error, Frame error, Break, Receive buffer data ready, Transmit buffer empty, and Transmit shifter empty. All of the conditions are indicated by the corresponding UART status register (UTRSTAT/UERSTAT).

The overrun error, parity error, frame error and break condition are referred as the receive error status. If the receive error status interrupt enable flag is set to one in the control register, UCON, the error will generate the receive error interrupt. When the interrupt is detected, user can identify the source of interrupt by reading the value of error status register, UERSTAT.

When the receiver transfers the data of the receive shifter to the receive FIFO register in FIFO mode and the number of received data reaches Rx FIFO Trigger Level, Rx interrupt is generated, if the receive mode in control register (UCON) is set to 0x1. In the non-FIFO mode, transferring the data of the receive shifter to receive holding register will cause Rx interrupt under the interrupt mode.

When the transmitter transfers data from its transmit FIFO register to its transmit shifter and the number of data left in transmit FIFO reaches Tx FIFO Trigger Level, Tx interrupt is generated, if the transmit mode in control register is set to 0x1. In the non-FIFO mode, transferring data from transmit holding register to the transmit shifter will case Tx interrupt under the interrupt mode.

If the receive mode and transmit mode are selected as the DMA request mode, DMA request operation occurs instead of Rx or Tx interrupt in the above situation.

Type	FIFO Mode	Non-FIFO Mode
Rx interrupt	<p>Generated whenever receive data reaches the trigger level of receive FIFO.</p> <p>Generated when the number of data in FIFO does not reaches Rx FIFO trigger Level and does not receive any data during 3 word time (receive time out). One word length is determined by the line control register.</p>	Generated by the receive holding register whenever receive buffer becomes full.
Tx interrupt	Generated whenever transmit data reaches the trigger level of transmit FIFO (Tx FIFO trigger Level).	Generated by the transmit holding register whenever transmit buffer becomes empty.
Error interrupt	<p>Generated when frame error, parity error, or break signal are detected.</p> <p>Generated when it gets to the top of the receive FIFO without reading out data in it (overrun error).</p>	Generated by all errors. However if another error occurs at the same time, only one interrupt is generated.

UART Error Status FIFO

UART has the error status FIFO besides the Rx FIFO register. The error status FIFO indicates which data, among FIFO registers, is received with an error. The error interrupt will be issued only when the data, which has an error in the error status FIFO, is ready to read out. To clear the error status FIFO, the URXHn with an error and UERSTATn must be read out.

For example,

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

The actual UART receive error will not generate any error interrupt because the character, which was received with an error, has not been read yet. The error interrupt will occur when the character is read out.

Time	Sequence Flow	Error Interrupt	Note
#0	When no character is read out	-	
#1	A, B, C, D, and E is received	-	
#2	After A is read out	The frame error (in B) interrupt occurs.	The 'B' has to be read out.
#3	After B is read out	-	
#4	After C is read out	The parity error (in D) interrupt occurs.	The 'D' has to be read out.
#5	After D is read out	-	
#6	After E is read out	-	

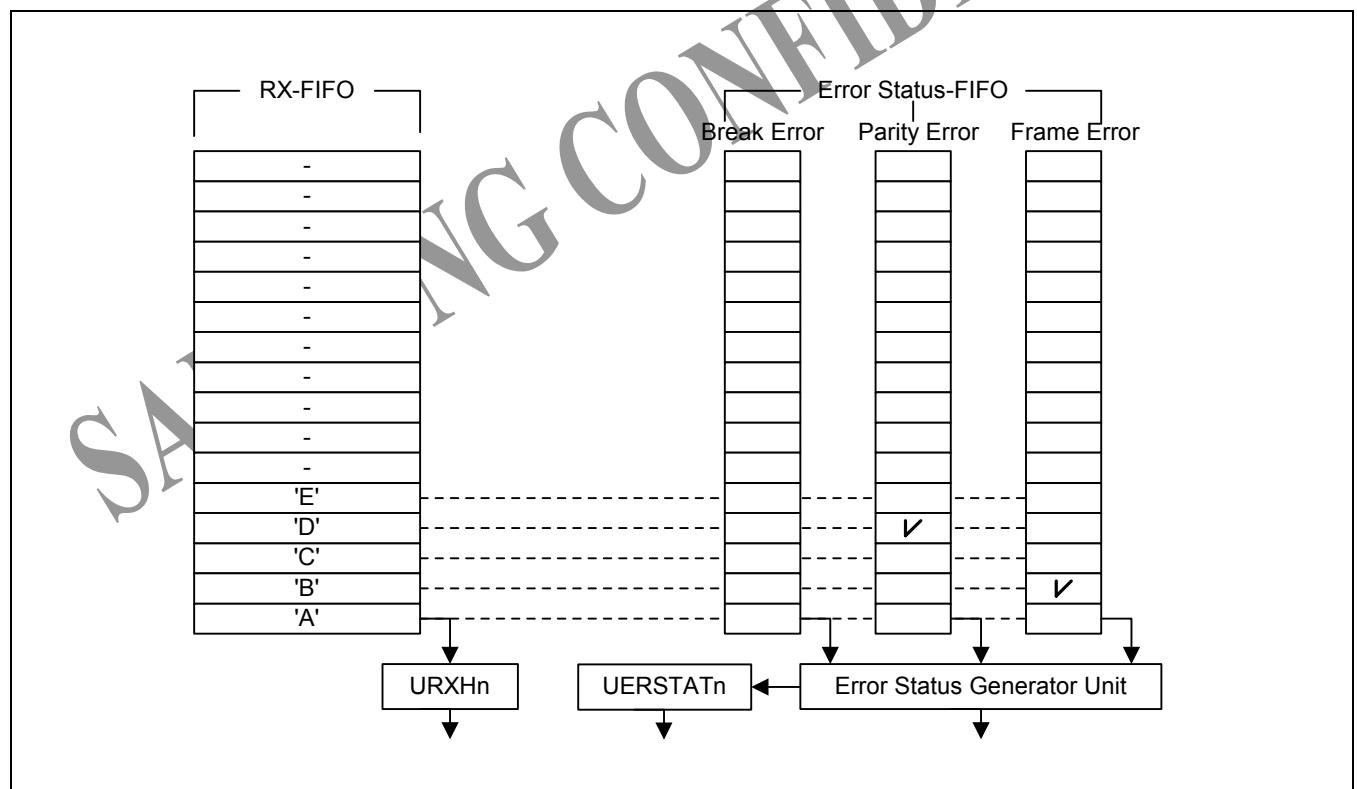


Figure 25-4. UART Receiving 5 Characters with 2 Errors

Baud-Rate Generation

Each UART's baud-rate generator provides the serial clock for the transmitter and the receiver. The source clock for the baud-rate generator can be selected with the internal system clock or external clock (UCLK). In other words, dividend is selectable by setting Clock Selection of UCON register. The baud-rate clock is generated by dividing the source clock (PCLK or UCLK) by 16 and a 16-bit divisor specified in the UART baud-rate divisor register (UBRDIV). The UBRDIV can be determined by the following expression:

$$\text{UBRDIV} = (\text{PCLK}/(\text{bps} \times 16)) - 1$$

Where, the divisor should be from 1 to $(2^{16}-1)$.

For accurate UART operation, the UART also supports external clock (UCLK) as a dividend.

If the UART uses UCLK, which is supplied by an external UART device or system, then the serial clock of UART is exactly synchronized with UCLK. So, the user can get the more precise UART operation. The UBRDIV can be determined:

$$\text{UBRDIV} = (\text{UCLK} / (\text{bps} \times 16)) - 1$$

Where, the divisor should be from 1 to $(2^{16}-1)$ and UCLK should be smaller than PCLK.

For example, if the baud-rate is 115200 bps and PCLK or UCLK is 40 MHz, UBRDIV is determined:

$$\begin{aligned}\text{UBRDIV} &= (40000000/(115200 \times 16)) - 1 \\ &= (\text{int})(21.7) - 1 \\ &= 21 - 1 = 20\end{aligned}$$

Loopback Mode

The UART provides a test mode referred to as the Loopback mode, to detect the faults in the communication link. This mode structurally enables the direct connection of RXD and TXD in the UART. In this mode, therefore, transmitted data is received to the receiver, via RXD. This feature allows the processor to verify the internal transmit and to receive the data path of each SIO channel. This mode can be selected by setting the loopback bit in the UART control register (UCON).

Break Condition

The break is defined as a continuous low-level signal for one frame transmission time on the transmit data output. This signal is distinguished from the zero value transmission by the stop bit. In the stop bit period, the break signal has low-level signal but the zero value transmission has high-level signal.

Infra-Red (IR) Mode

The UART block supports infra-red (IR) transmission and reception, which can be selected by setting the Infra-red-mode bit in the UART line control register (ULCON). Figure 5 illustrates how to implement the IR mode. In IR transmit mode, the transmit pulse comes out at a rate of 3/16, the normal serial transmit rate (when the transmit data bit is zero); In IR receive mode, the receiver must detect the 3/16 pulsed period to recognize a zero value (see the frame timing diagrams shown in Figure 7 and 8).

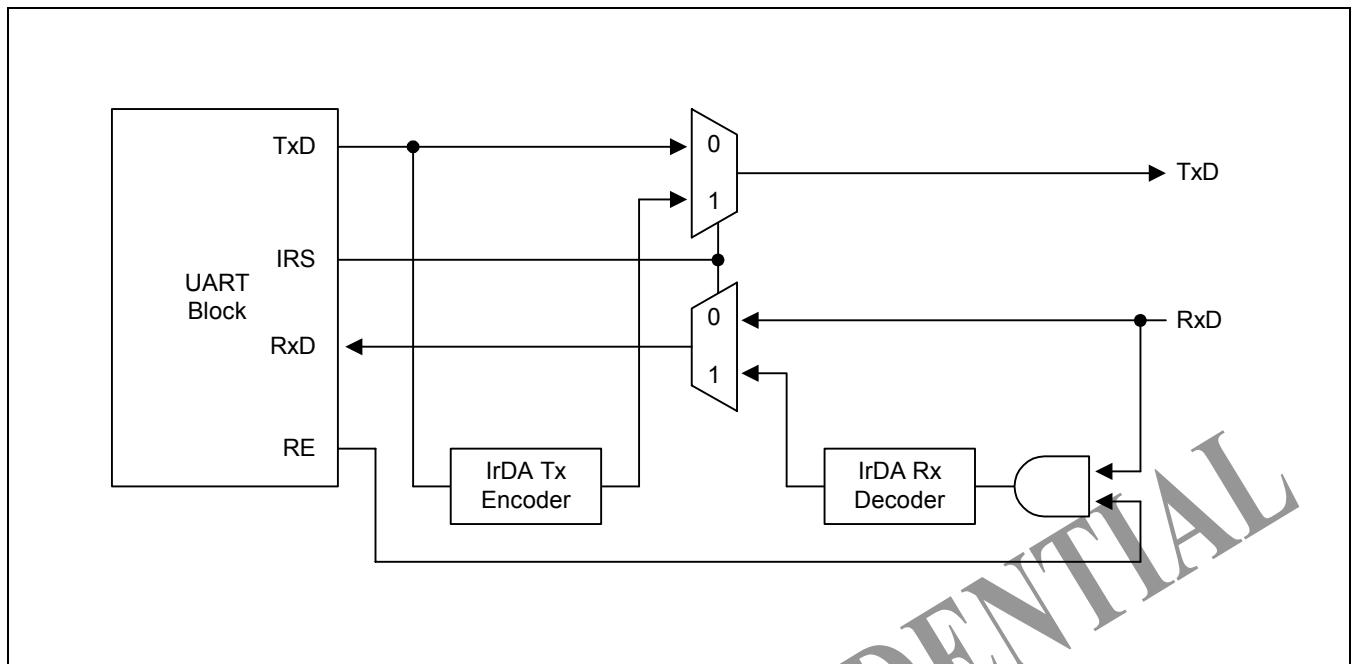


Figure 25-5. IrDA Function Block Diagram

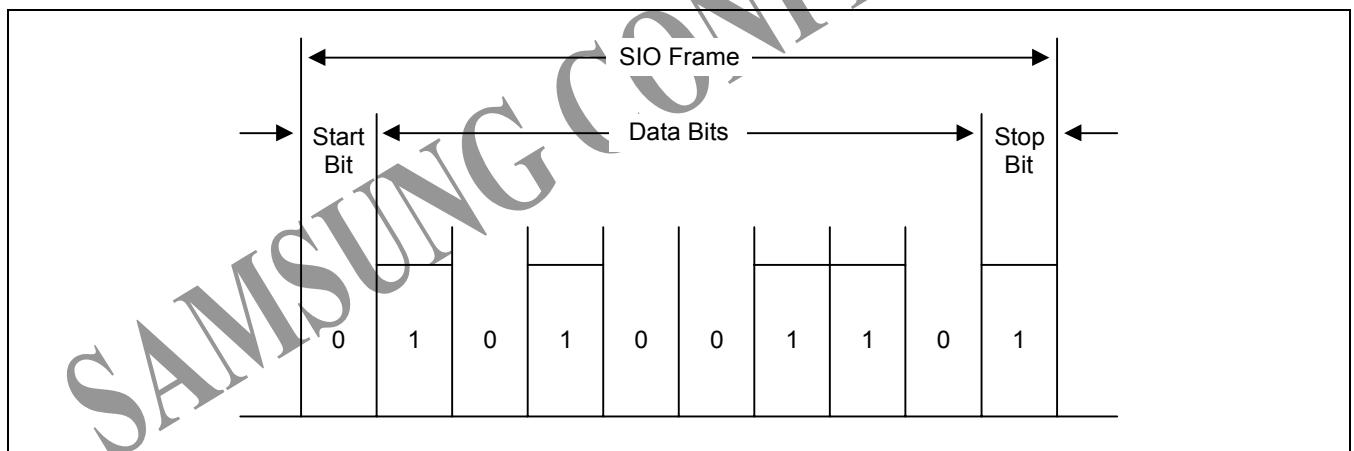


Figure 25-6. Serial I/O Frame Timing Diagram (Normal UART)

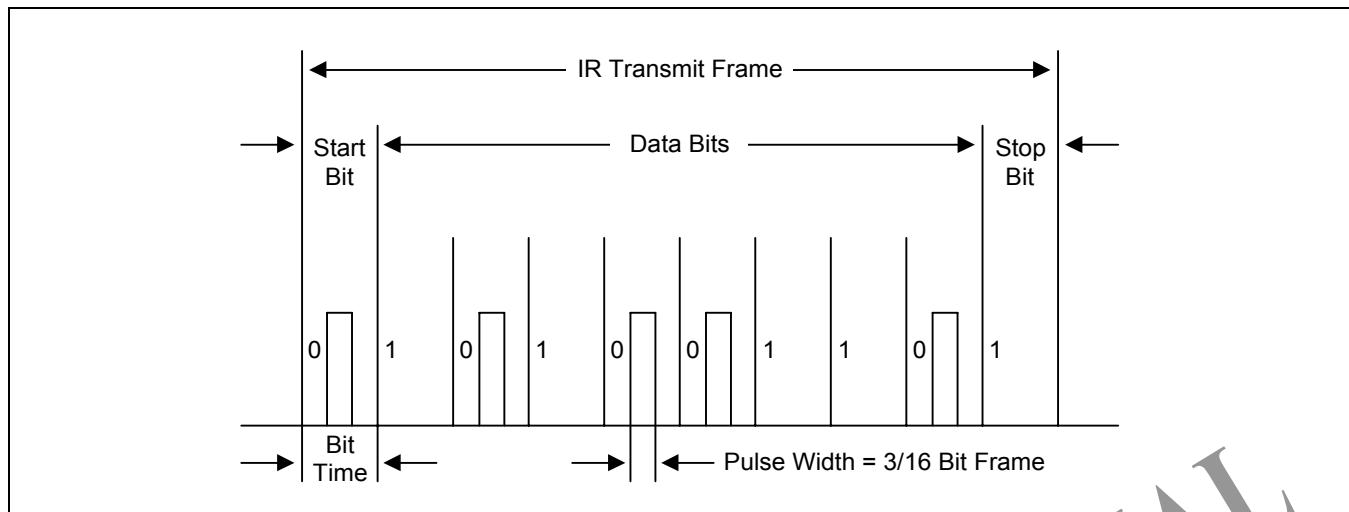


Figure 25-7. Infra-Red Transmit Mode Frame Timing Diagram

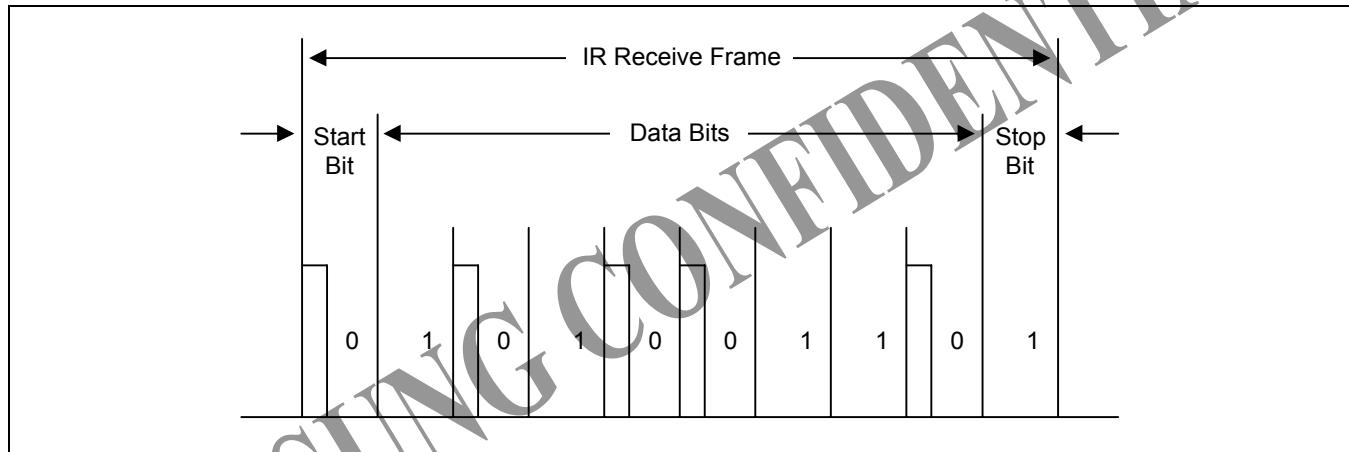


Figure 25-8. Infra-Red Receive Mode Frame Timing Diagram

26 LCD INTERFACE CONTROLLER

OVERVIEW

LCD Interface controller supports the interface between LCD controller with 6800series (Motorola) and 8080serise(intel).

FEATURE

- 16-/8-/4-bit parallel interface mode (6800/8080).
- 3-/4-Pin serial Interface mode.
- Support multiple frequencies internal clock for high and low speed controller.
- Contains a 16 bytes FIFO for sending control and data information to the LCD controller.
- Apply 32 Bits, 16bit and 8bit write APB Bus on FIFO.
- Contains maskable interrupts.
- This LCD interface supports 1-burst and 2-burst data operation when using IODMA.

BLOCK DIAGRAM

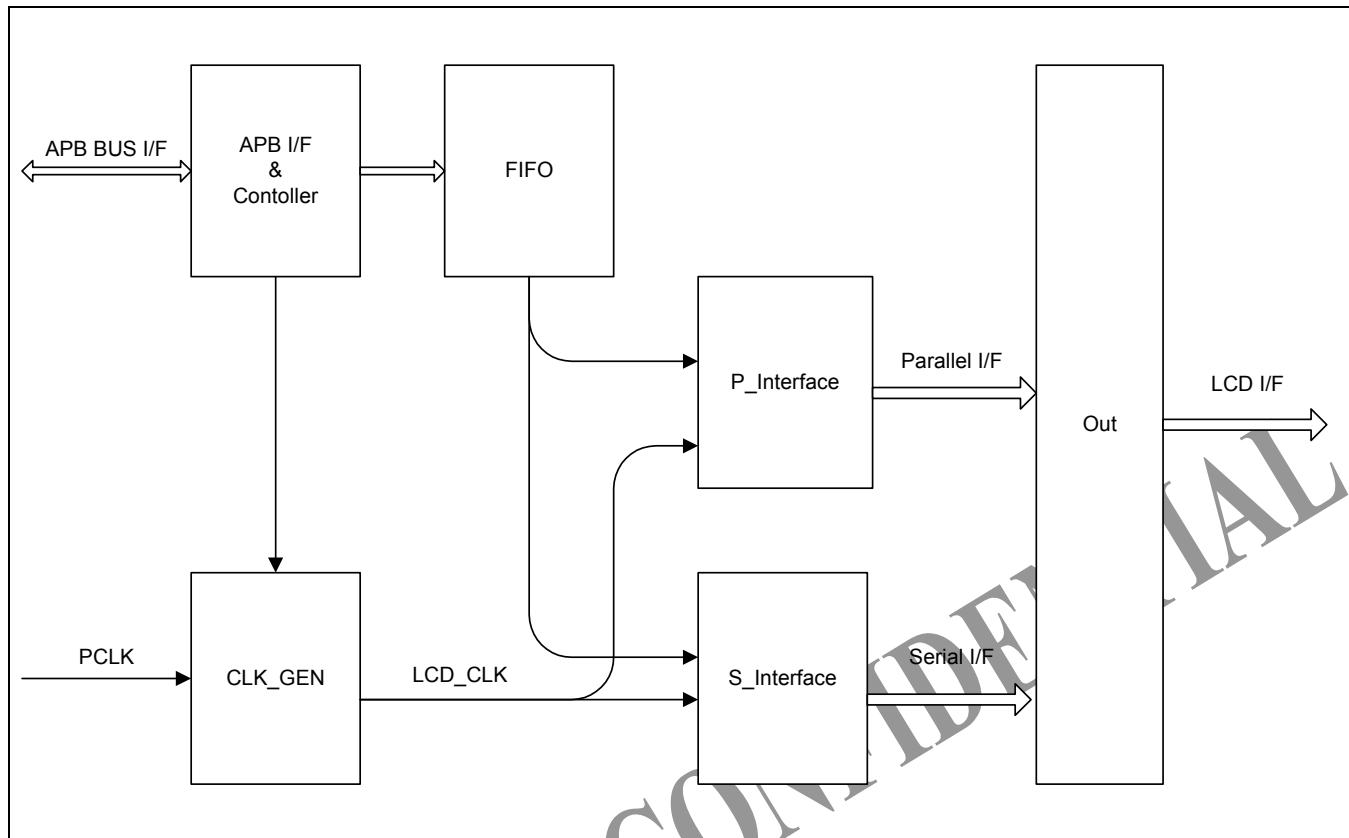


Figure 26-1. LCD Interface Controller Block Diagram

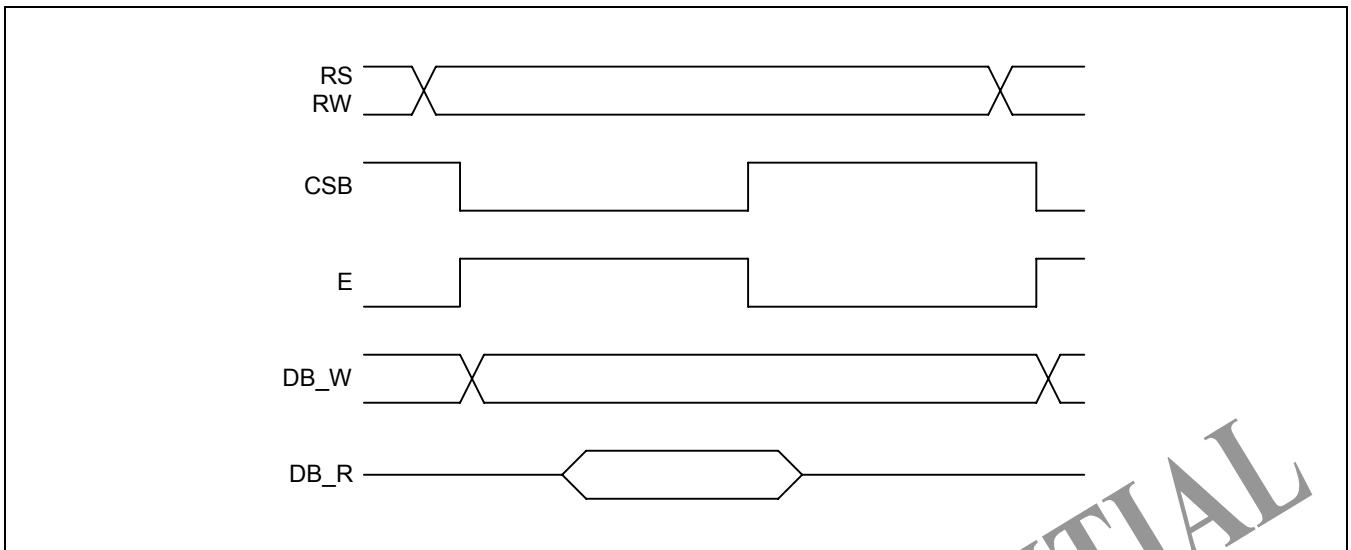
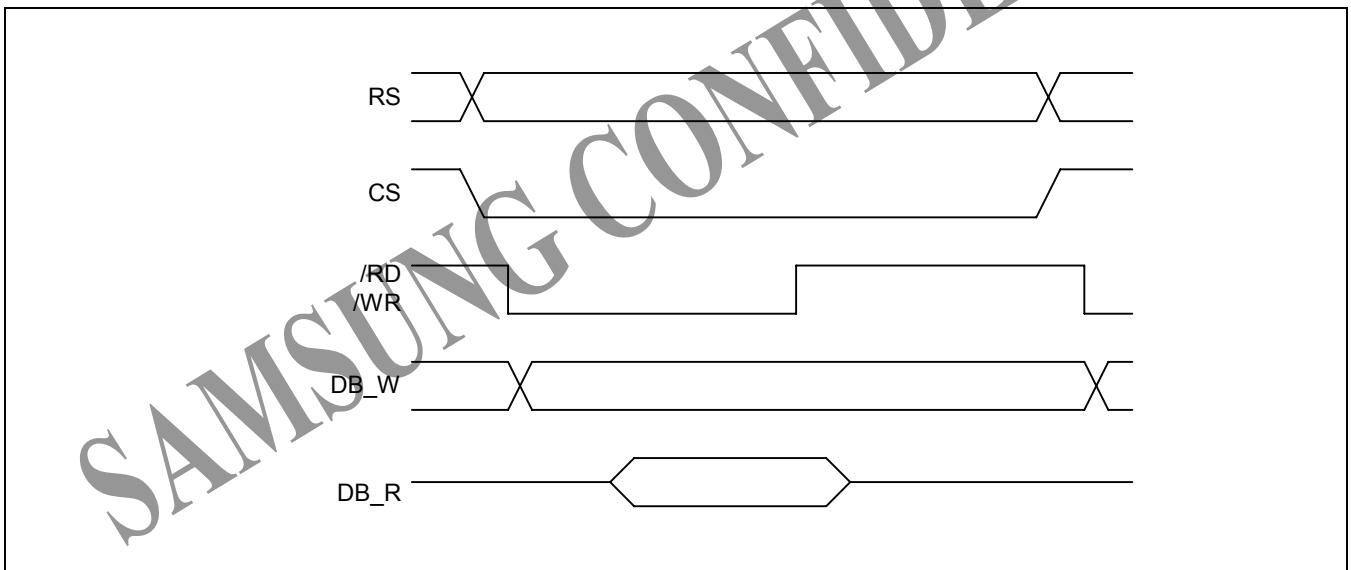
1. APB I/F & Controller: This block supports the interface with HOST and generate the control signal, interrupt and status.
2. CLK_GEN: As to the value set in register, this block generate the LCD clock which is used as clock source in this block. (2/4/8/16/32/64/128/256 divider).
3. FIFO: 9 bits x 16 depths FIFO. MSB 1bit is used as RS Signal (Command/Data) LSB 8 bits is used for sending data. Each data is send to P_interface or S_interface block as to control register.
4. P_Interface: This block control the data transfer including control signal in parallel mode. In serial mode, this block is disabled.
5. S_Interface: This block control the data transfer including control signal in serial mode. In parallel mode, this block is disabled.

PIN DESCRIPTION

	Width	I/O	Description
TestMode	1	I	For scan enable signal
PCLK	1	I	Global clock
PRESETn	1	I	Global reset
APB Interface			
PSEL	1	I	Selection in APB
PENABLE	1	I	Enable in APB
PWRITE	1	I	Write/Read in APB
PADDR	4	I	Address in APB
PWDATA	9	I	Write data in APB
PRDATA	9	O	Read data in APB
LCD driver Interface			
LCD_RST	1	O	LCD driver reset
LCD_CSB	1	O	LCD driver chip select
LCD_RS	1	O	LCD driver register select.
LCD_RW_WR	1	O	Read/Write execution control pin
LCD_E_RD	1	O	Read/Write execution control pin
LCD_DB[3:0]	4	I/O	Bi-directional data bus (16/8bit mode)
LCD_DB[5:4]	2	I/O	Bi-directional data bus (16/8/4bit mode)
LCD_DB[6]/SCLK	1	I/O	Bi-directional data bus (16/8/4bit mode) & SCLK
LCD_DB[7]/SDO	1	I/O	Bi-directional data bus (16/8/4bit mode) & SDO
LCD_DB[15:8]	8	I/O	Bi-directional data bus (16bit mode)
Interrupt Interface			
INT_LCD	1	O	Interrupt Signal

LCD Driver Interface

- LCD_RST : LCD driver reset signal.
- LCD_CSB : LCD driver chip select signal.
- LCD_RS : LCD driver register select signal.
RS = 1 : DB0 to DB15 are display data.
RS = 0 : DB0 to DB15 are control data.
- LCD_RW_WR : Read / Write execution control Pin.
In 6800 Mode, High : Read, Low : Write.
In 8080 Mode, Write enable signal.
- LCD_E_RD : Read / Write execution control Pin.
In 6800 Mode, Read / Write Enable signal.
In 8080 Mode, Read enable signal.
- DB[3:0] : Bi-directional data bus low 4bits.
- DB[5:4] : Bi-directional data bus data[5:4] bit & data[1:0] bit at 4bit mode.
- DB[6] : Bi-directional data bus data[6], data[2] at Parallel Mode, serial clock(SCLK) at serial mode.
- DB[7] : Bi-directional data bus data[7], data[3] at Parallel Mode, serial output data(SDO) at serial mode.
- DB[15:8] : Bi-directional data bus data[15:8].

TIMING DIAGRAM**Figure 26-2. 6800 Mode Timing Diagram****Figure 26-3. 8080 Mode Timing Diagram**

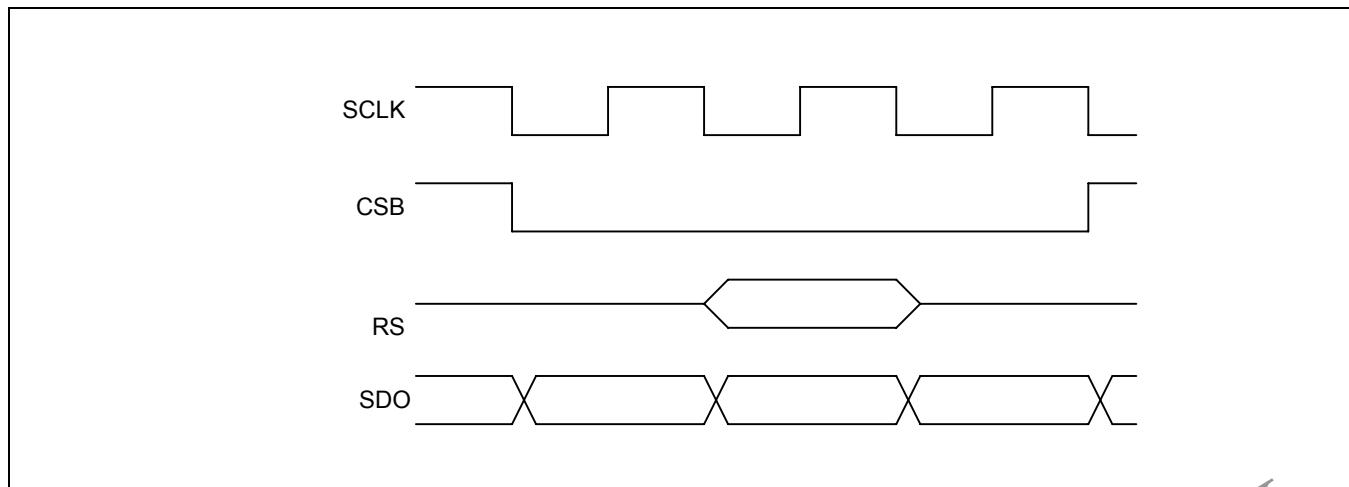


Figure 26-4. Serial Mode Timing Diagram

REGISTERS

Name	Width	Address	R/W	Description	Reset
LCD_CON	32	0x3C10_0000	R/W	Control register.	0x0000 0000
LCD_WCMD	32	0x3C10_0004	W	Write command register.	0x0000 0000
LCD_RCMD	32	0x3C10_000C	W	Read command register.	0x0000 0000
LCD_RDATA	32	0x3C10_0010	R	Read data register.	0x0000 0000
LCD_DBUFF	32	0x3C10_0014	R	Read Data buffer	0x0000 0000
LCD_INTCON	32	0x3C10_0018	R/W	Interrupt control register	0x0000 0000
LCD_STATUS	32	0x3C10_001C	R	LCD Interface status	0x0000 0106
LCD_PHTIME	32	0x3C10_0020	R/W	Phase time register	0x0000 0060
LCD_RST_TIME	32	0x3C10_0024	R/W	Reset active period	0x0000 07FF
LCD_DRV_RST	32	0x3C10_0028	R/W	Reset drive signal	0x0000 0001
LCD_WDATA	32	0x3C10_0040 ~ 0x3C10_005C	W	Write data register	0x0000 0000

LCD_CON

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bit	Name	Type	Description	Reset Value
11	DMA_EN	R/W	DMA control enable 0 : DMA disable 1 : DMA enable	0
10	LCD_ON	R/W	LCD Interface On.	0
9:8	W_LEN	R/W	APB Bus interface word length. 00 : 8 bit. 01 : 16 bit 10 : 32 bit	00
7	ENDIAN	R/W	APB Bus word Endian. 0 : Little endian. 1 : Big endian. When W_LEN = 0, No effect.	0
6	PS1	R/W	Microprocessor interface select. When PS0 = 0 PS1 = 0 : 3 pin-SPI MPU interface. PS1 = 1 : 4 pin-SPI MPU interface. When PS0 = 1 PS1 = 0 : 8080-series parallel MPU interface. PS2 = 1 : 6800-series parallel MPU interface.	0
5	PS0	R/W	Parallel/serial data select. When 0 Serial MPU Interface. When 1 Parallel MPU interface.	0
4:3	BUS_M	R/W	Bus mode Select at parallel Interface. 00 : 8bit Data Bus. 01 : 4 bit Data Bus. 1x : 16 bit Data Bus.	00
2:0	CLK_SEL	R/W	LCD clock select. 000 : PCLK / 2, 001 : PCLK / 4 010 : PCLK / 8, 011 : PCLK / 16 100 : PCLK / 32, 101 : PCLK / 64 110 : PCLK / 128, 111 : PCLK / 256	000

LCD_WCMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bit	Name	Type	Description												Reset Value
7:0	WCMD	W	Write LCD driver command.												0x00000000

LCD_WDATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bit	Name	Type	Description												Reset Value
31:0*	WDATA	W	Write LCD driver data.												0x00000000

NOTE: In 8bit Length mode, 31~ 8 data bits are not valid.

LCD_RCMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bit	Name	Type	Description												Reset Value
7:0	RCMD	W	Read command LCD driver command. Dummy value.												0x00000000

LCD_RDATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bit	Name	Type	Description										Reset Value
7:0	RDATA	R	Read LCD driver data										0x00000000

LCD_DBUFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bit	Name	Type	Description										Reset Value
15:0	DBUFF	R	Read LCD driver data buff.										0x00000000

LCD_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bit	Name	Type	Description										Reset Value
8	bus_idle	R	Data bus idle										1
7	Interrupt	R	Interrupt Flag.										0
6	Reserved	R											0
5	OW	R	FIFO is Over write.										0
4	FULL	R	FIFO is Full.										0
3	HFULL	R	FIFO is half full.										0
2	HEMPTY	R	FIFO is half empty.										1
1	EMPTY	R	FIFO is empty.										1
0	READON	R	Read operation done.										0

LCD_INT_CON

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bit	Name	Type	Description										Reset Value
7	INT_EN	R/W	Interrupt enable.										0
6	F_CLEAR	R/W	FIFO Clear.										0
5	OW_EN	R/W	Over write interrupt enable.										0
4	F_EN	R/W	FIFO Full interrupt enable.										0
3	HF_EN	R/W	FIFO half full interrupt enable.										0
2	HE_EN	R/W	FIFO half empty interrupt enable.										0
1	EM_EN	R/W	FIFO empty interrupt enable.										0
0	RED_EN	R/W	Read done interrupt enable.										0

LCD_PHTIME

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bit	Name	Type	Description												Reset Value
7:4	PH1_TIME	R/W	Phase 1 time register												6
3:0	PH2_TIME	R/W	Phase 2 time register												0

If serial mode is selected, place more than 22h in LCD_PHTIME register. For example,

LCD_PHTIME = 00h (No recommended), LCD_PHTIME = 11h (No recommended)

LCD_PHTIME = 22h (ok), LCD_PHTIME = 33h (ok),

LCD_PHTIME = more than 33h (ok)

LCD_RST_TIME

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bit	Name	Type	Description												Reset Value
10:0	RST_TIME	R/W	RESET TIME register												0x7FF

LCD_DRV_RST

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bit	Name	Type	Description												Reset Value
1	Rst_out	R	Indicates the current reset state.												
0	Drv_rst	W	Driving reset signal.												1

OPERATION

Clock Select

As to the operation frequency of LCD Driver IC, CPU selects the frequency of clock in LCD interface controller. There is the register for this operation. Using this internal clock, the following signals, RD_RW, E_RD and CSB are generated. At this time, the high pulse width (tPWH) and the low pulse width (tPWL) is controlled also by LCD_PHTIME Register.

PCLK	CLK_SEL	tPW Time Unit (ns)
100 MHz	000 (x 1/2)	20
	001 (x 1/4)	40
	010 (x 1/8)	80
	011 (x 1/16)	160
	100 (x 1/32)	320
	101 (x 1/64)	640
	110 (x 1/128)	1280
	111 (x 1/256)	2560
50 MHz	000 (x 1/2)	40
	001 (x 1/4)	80
	--	--
	110 (x 1/128)	2560
	111 (x 1/256)	5120

$$tPW = (\text{PH1_TIME Value} + 1) \times tPW \text{ time unit (Phase 1)}$$

$$tPW = (\text{PH2_TIME Value} + 1) \times tPW \text{ time unit (Phase 2)}$$

Mode Select

The bit PS1 and PS0 of control register set the Interface mode.

PS0	PS1	Interface Mode	Data/Instruction	Data	Read/Write	Serial Clock
0	0	Serial (3 pin)	None	SDO(DB7)	Write only	SCLK(DB6)
0	1	Serial (4 pin)	RS	SDO(DB7)	Write only	SCLK(DB6)
1	0	Parallel (8080)	RS	DB[7:0]	Read: E_RD Write: RW_WR	None
1	1	Parallel (6800)	RS	DB[7:0]	Read: RW_WR = high Write: RW_WR = low	None

In parallel Interface mode, using the BUS_M bit of control register, 8bit data bus mode or 4bit data bus mode is set.

Interrupt Control

There are 5 Interrupt sources as the following:

- 1) Read done
- 2) FIFO empty
- 3) FIFO Half empty
- 4) FIFO Full
- 5) FIFO Over run

Each interrupt source can be masked each maskable control bit or common interrupt mask bit. If the Interrupt is occurred at once, CPU can find out the source by reading status register.

The way of clearing each interrupt source is as following:

- 1) FIFO empty or FIFO Half empty: Writing the data to FIFO.
- 2) FIFO Full or FIFO Over run: clearing FIFO
- 3) Read done: reading LCD_DBUFF register.

LCD Command & Data

The bit position 7 in FIFO named RS bit is used as indicating the data or command.

This value is set automatically as to writing to which register.

To write command to LCD Driver, CPU has to write to LCD_WCMD. At this time, the value of RS is set automatically as zero. LCD interface controller outputs the command to LCD Driver IC.

To write data to LCD Driver, CPU has to write to LCD_WDATA. At this time, the value of RS is set automatically as one. LCD interface controller outputs the data to LCD Driver IC.

To read the status of LCD Driver, CPU writes the dummy data to LCD_RCMD register.

Then, The status value is saved in LCD_DBUFF.

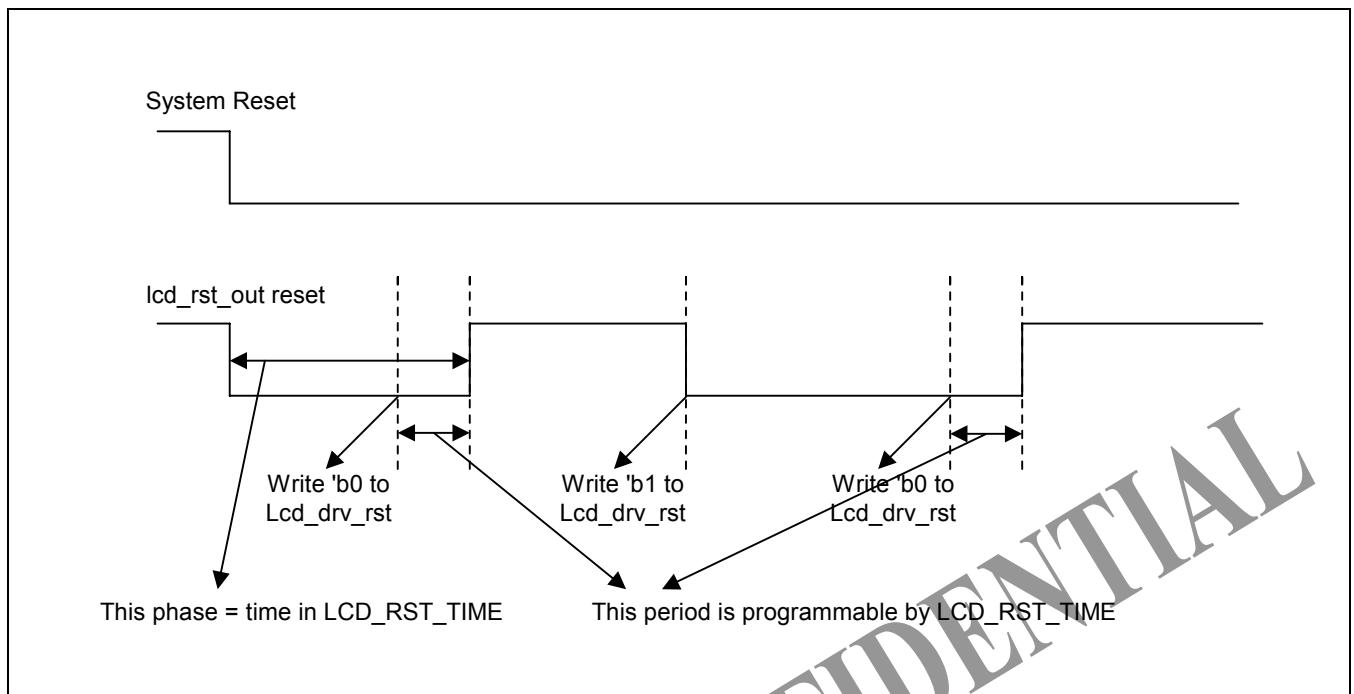
To read the data of LCD Driver, CPU writes the dummy data to LCD_RDAT register.

Then, The status value is saved in LCD_DBUFF.

After reading the status or data is finished, the read_done_flag is set to high.

CPU read this value using interrupt or polling.

SAMSUNG CONFIDENTIAL

LCD Controller Reset Signal Generation**Figure 26-5. LCD Controller Reset Signal Generation**

NOTES

SAMSUNG CONFIDENTIAL

27 CLCD CONTROLLER

OVERVIEW

The LCD controller 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 1-bit per pixel, 2-bit per pixel, 4-bit per pixel, 8-bit per pixel for interfacing with the palettized TFT color LCD panel, 16-bit per pixel and 24-bit per pixel non-palettized true-color display.

The LCD controller can be programmed to support the different requirements on the screen related to the number of horizontal and vertical pixels, data line width for the data interface, interface timing, and refresh rate.

FEATURES

Video Clock Source	HCLK, ECLK (PLL or External clock)
External Panel Interface	Supports 8 bit serial Panel (8/6 bit RGB-IF mode) Supports RGB/BGR both mode
OSD(Overlay)	Supports 8-bpp (bit per pixel) palettized color displays for TFT Supports 16-bpp non-palettized true-color displays for color TFT Supports 24-bpp non-palettized true-color displays for color TFT Supports X,Y indexed position Supports 8 bit Alpha blending : per Plan or per Pixel (24bpp only)
Color Level of TFT	Supports 1, 2, 4 or 8-bpp (bit per pixel) palettized color displays for TFT Supports 16-bpp non-palettized high-color displays for color TFT Supports 24-bpp non-palettized true-color displays for color TFT
Configurable Burst Length	Support programmable 4 / 8 / 16 Burst DMA operation
Dual Palette	256 x 24 bit palette (2ea for each Back-ground and OSD image)
Soft Scrolling	Horizontal : 1 Byte resolution Vertical : 1 pixel resolution
Virtual Screen	Virtual image can has up to 16MB image size.
Double Buffering	Frame buffer alternating by one control bit
Color Key (Chroma Key)	Support 24 bit color key function
Dithering	Patented 4x4 dither matrix implemetation

EXTERNAL INTERFACE SIGNAL

Name	Type	Source/Destination	Description
VCLK	Output	Pad	Video Clock Signal
HSYNC	Output	Pad	Horizontal Sync. Signal
VSYNC	Output	Pad	Vertical Sync. Signal
VDEN	Output	Pad	Video Data Enable/Valid
VD[23:0]	Output	Pad	LCD pixel data output

SAMSUNG CONFIDENTIAL

BLOCK DIAGRAM

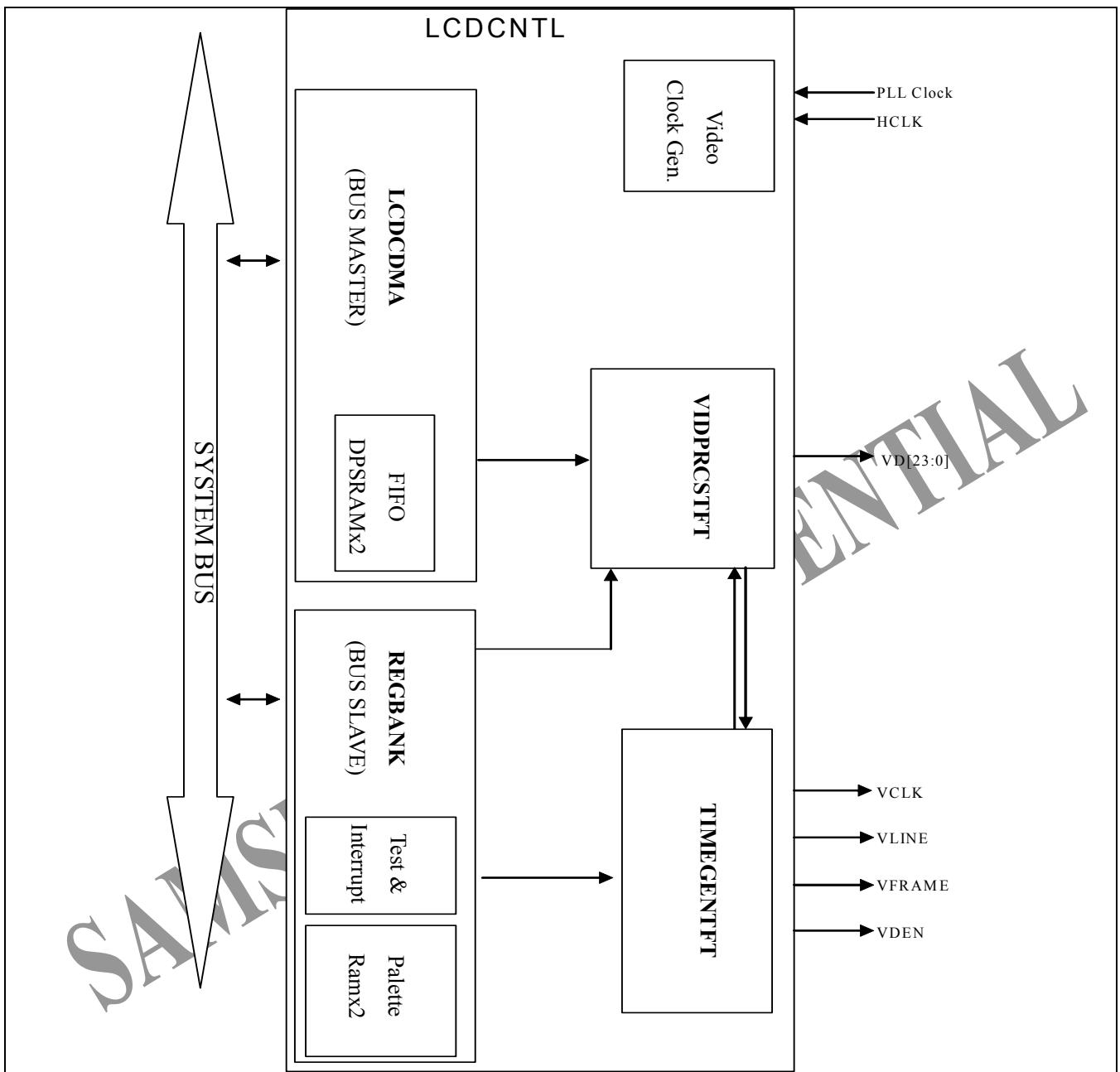


Figure 27-1. LCD Controller Block Diagram

The LCD controller is used to transfer the video data and to generate the necessary control signals such as, VSYNC, HSYNC, VCLK, and VDEN. As well as the control signals, the data ports for video data are VD[23:0] as shown in Figure 21-1. The LCD controller consists of a REGBANK, LCDLCDMA, VIDPRCS, TIMEGEN, and video clock generator (See Figure 21-1 LCD Controller Block Diagram). The REGBANK has 26 programmable register sets and 256x24 palette memory which are used to configure the LCD controller. The LCDLCDMA 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 LCDLCDMA and sends the video data through the VD[23:0] data ports to the LCD driver after changing

them into a suitable data format, for example 8-bit per pixel mode(8 BPP Mode) or 16-bit per pixel mode(16 BPP 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 VSYNC, HSYNC, VCLK, and VDEN.

The description of data flow is as follows:

FIFO memory is present in the LCDDMA. When FIFO is empty or partially empty, LCDDMA requests data fetching from the frame memory based on the burst memory transfer mode(Consecutive memory fetching of 4 / 8 / 16 words 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 4 /8 /16 successive word data transfers from system memory to internal FIFO. The total size of FIFO is 128x2 words, which consists of 128 words for back-ground FIFO and 128 words fore-ground FIFO, respectively. The LCD controller has two FIFOs because it needs to support the OSD display mode. In case of one screen display mode, the back-ground FIFO could only be used.

LCD controller supports overlay function which enabling overlay any image (OSD, fore ground image) which is smaller or same size can be blending with back ground image with programmable alpha blending or color (chroma) key function.

SAMSUNG CONFIDENTIAL

TIMING CONTROLLER OPERATION

The TIMEGEN generates the control signals for LCD driver such as, VSYNC, HSYNC, VCLK, and VDEN signal. These control signals are highly related with the configuration on the LCDTCON1/2/3 registers in the REGBANK. Base on these programmable configurations on the LCD control registers in REGBANK, the TIMEGEN can generate the programmable control signals suitable for the support of many different types of LCD drivers.

The VSYNC signal is asserted to cause the LCD's line pointer to start over at the top of the display.

The VSYNC and HSYNC pulse generation is controlled by the configuration of both the HOZVAL field and the LINEVAL registers. The HOZVAL and LINEVAL can be determined by the size of the LCD panel according to the following equations:

$$\text{HOZVAL} = (\text{Horizontal display size}) - 1$$

$$\text{LINEVAL} = (\text{Vertical display size}) - 1$$

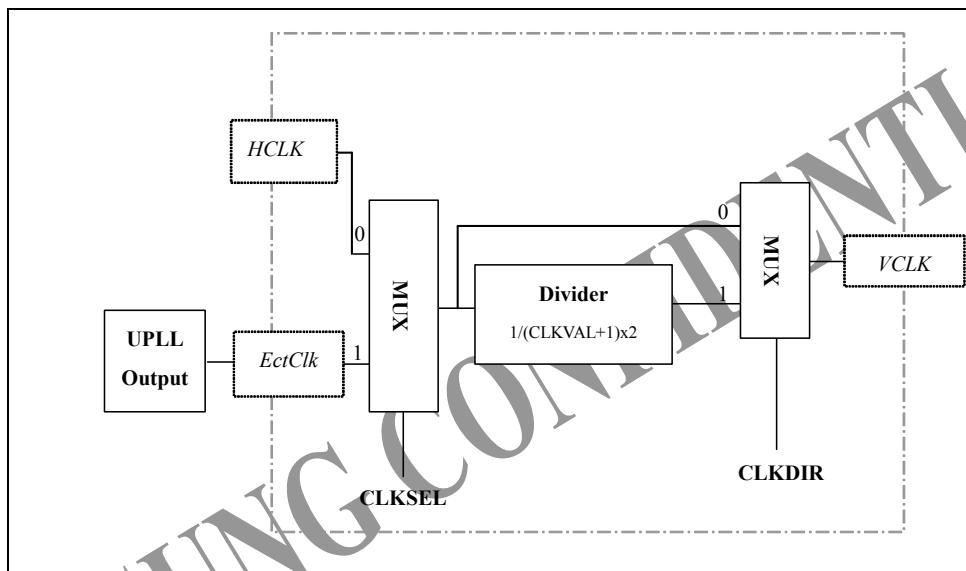


Figure 27-2. Clock selection

The rate of VCLK signal can be controlled by the CLKVAL field in the LCDCON1 register. The table below defines the relationship of VCLK and CLKVAL. The minimum value of CLKVAL is 0.

$$\text{VCLK (Hz)} = \text{HCLK} / [(CLKVAL+1) \times 2]$$

The frame rate is VSYNC signal frequency. The frame rate is related with the field of VSYNC, VBDP, VFDP, LINEVAL, HSYNC, HBPD, HFDP, HOZVAL, CLKVAL registers. Most LCD driver needs their own adequate frame rate. The frame rate is calculated as follows;

$$\begin{aligned} \text{Frame Rate} = 1 / [& \{ (\text{VSPW}+1) + (\text{VBPD}+1) + (\text{LINEVAL} + 1) + (\text{VFDP}+1) \} \times \{ (\text{HSPW}+1) + (\text{HBPD} + 1) \\ & + (\text{HFDP}+1) + (\text{HOZVAL} + 1) \} \times \{ 2 \times (\text{CLKVAL}+1) / (\text{Frequency of Clock source}) \}] \end{aligned}$$

Table 27-1. Relation between VCLK and CLKVAL (TFT, Freq. of Video Clock Source=60MHz)

CLKVAL	60MHz/X	VCLK
1	60 MHz/4	15.0 MHz
2	60 MHz/6	10.0 MHz
:	:	:
63	60 MHz/128	492 kHz

SAMSUNG CONFIDENTIAL

VIDEO OPERATION

The TFT LCD controller supports 1, 2, 4 or 8 bpp(bit per pixel) palettized color displays and 8, 16 non-palettized high-color or 24 bpp non-palettized true-color displays. The TFT LCD controller also supports On-Screen Display with 256-level alpha blending and color (chroma) key functions. The Back-ground image and Fore-ground image (OSD image) should have a frame buffer of each image.

OSD (ON-SCREEN DISPLAY) : OVERLAY

OSD (On Screen Display) and blending operation as shown in Fig 21-x are established for video overlay or other graphics applications. Two blending schemes are provided according to the control bit of OSD_BLD_PIX. One is per-pixel blending for 24bpp mode display (OSD_BLD_PIX = 1) and the other is per-plane bending for 8/16/24bpp mode display (OSD_BLD_PIX = 0). LCDB1ADDR1/2/3, LCDB2ADDR1/2/3 register are defined to perform DMA for OSD image. Four screen coordinates such as OSD_LEFT_TOP_X, OSD_LEFT_TOP_Y, OSD_RIGHT_BOT_X and OSD_RIGHT_BOT_Y determines where the OSD image is located on the whole background image. The level of blending is controller by OSD_ALPHA as following manner.

$$\text{New Pixel} = \text{Alpha} \times \text{Background Pixel} + (1 - \text{Alpha}) \times \text{Foreground Pixel}$$

Where, Alpha = 0 $\sum_{i=1,2,\dots,7} 2^{-i} \times \text{OSD_ALPHA}[7-i]$, if OSD_ALPHA[7:0] = 0
 , other

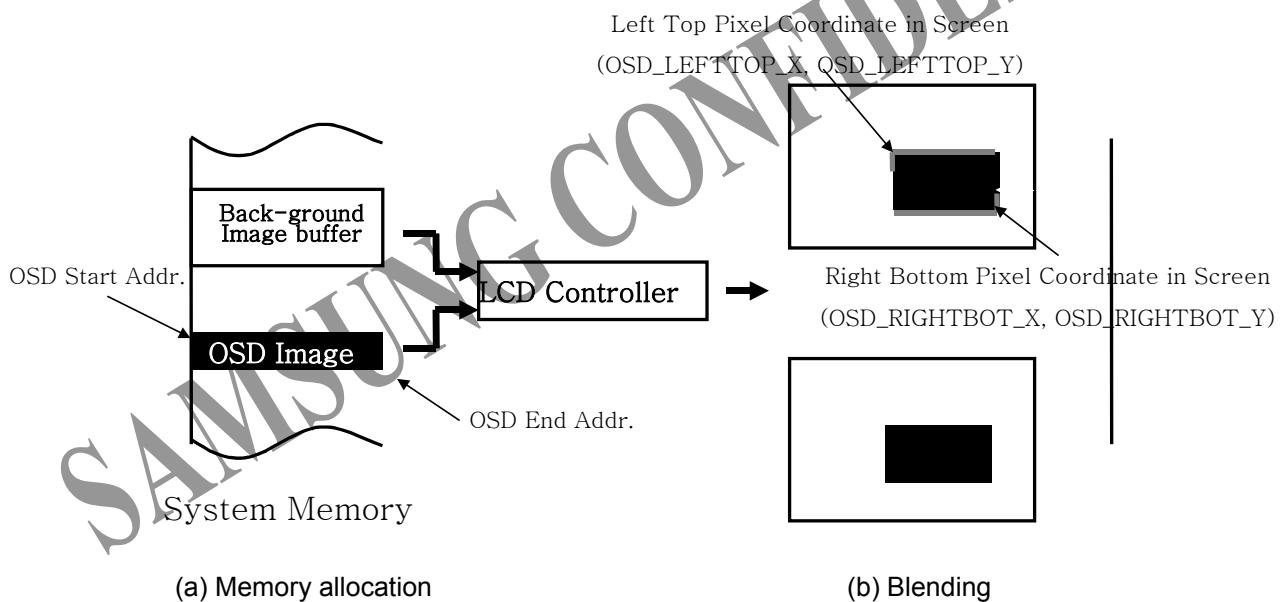


Fig 27-3 OSD and blending procedure

COLOR-KEY FUNCTION

The LCD controller can support color-key function for the various effect of image mapping. Color image, which is specified by COLOR-KEY register, of OSD layer will be substituted by back-ground image for special functionality, as cursor image or pre-view image of the camera.

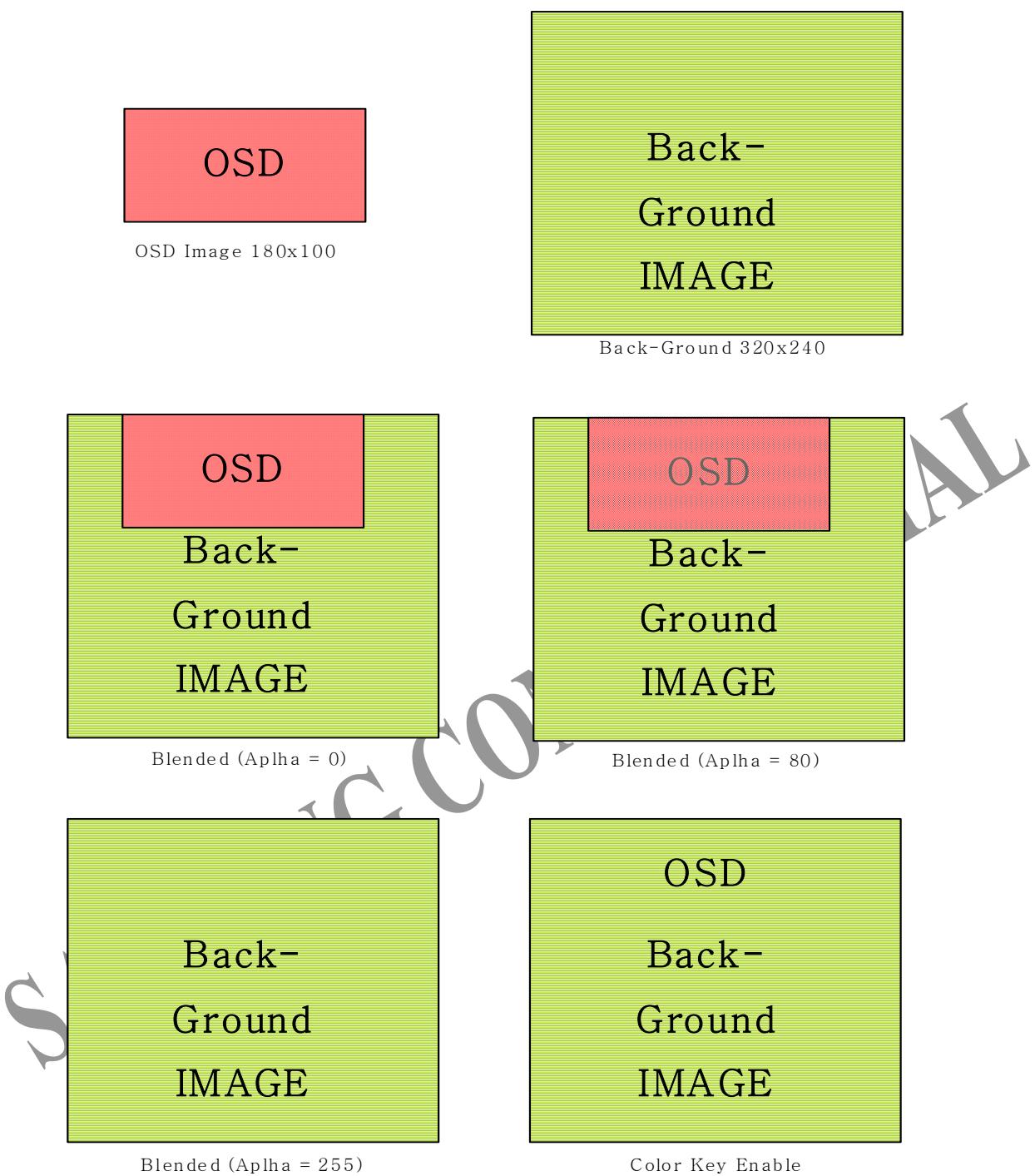


Figure 27-4. Color-key function configuration

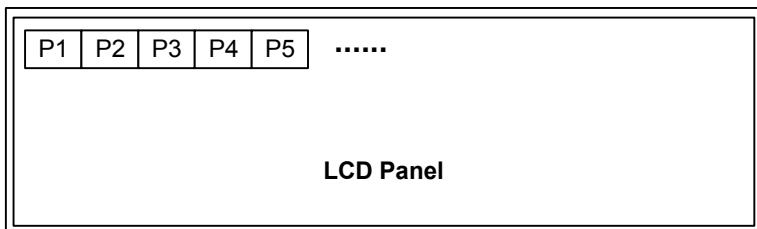
MEMORY DATA FORMAT (TFT)

The LCD controller request the specified memory format of frame buffer. The next table shows some examples of each display mode.

24BPP Display

(BSWP = 0, HWSWP = 0)

	D[31:24]	D[23:0]
000H	Dummy Bit	P1
004H	Dummy Bit	P2
008H	Dummy Bit	P3
...		



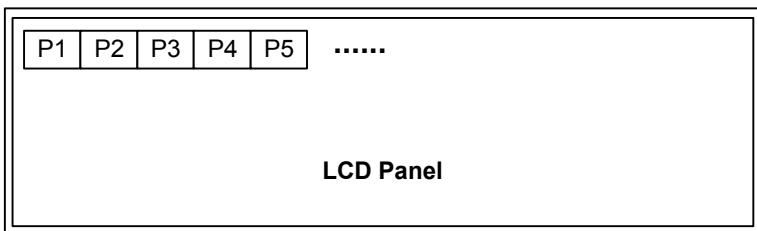
16BPP Display

(BSWP = 0, HWSWP = 0)

	D[31:16]	D[15:0]
000H	P1	P2
004H	P3	P4
008H	P5	P6
...		

(BSWP = 0, HWSWP = 1)

	D[31:16]	D[15:0]
000H	P2	P1
004H	P4	P3
008H	P6	P5
...		



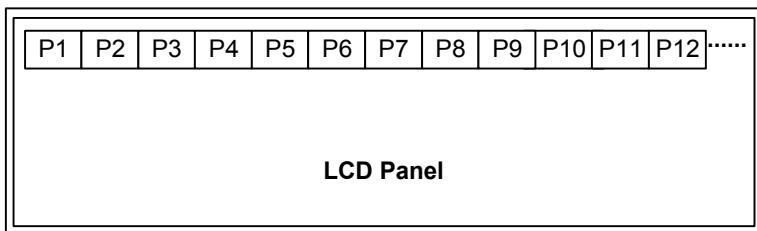
8BPP Display

(BSWP = 0, HWSWP = 0)

	D[31:24]	D[23:16]	D[15:8]	D[7:0]
000H	P1	P2	P3	P4
004H	P5	P6	P7	P8
008H	P9	P10	P11	P12
...				

(BSWP = 1, HWSWP = 0)

	D[31:24]	D[23:16]	D[15:8]	D[7:0]
000H	P4	P3	P2	P1
004H	P8	P7	P6	P5
008H	P12	P11	P10	P9
...				



4BPP Display

(BSWP = 0, HWSWP = 0)

	D[31:28]	D[27:24]	D[23:20]	D[19:16]	D[15:12]	D[11:8]	D[7:4]	D[3:0]
000H	P1	P2	P3	P4	P5	P6	P7	P8
004H	P9	P10	P11	P12	P13	P14	P15	P16
008H	P17	P18	P19	P20	P21	P22	P23	P24
...								

(BSWP = 1, HWSWP = 0)

	D[31:28]	D[27:24]	D[23:20]	D[19:16]	D[15:12]	D[11:8]	D[7:4]	D[3:0]
000H	P7	P8	P5	P6	P3	P4	P1	P2
004H	P15	P16	P13	P14	P11	P12	P9	P10
008H	P23	P24	P21	P22	P19	P20	P17	P18
...								

2BPP Display

(BSWP = 0, HWSWP = 0)

D	[31:30]	[29:28]	[27:26]	[25:24]	[23:22]	[21:20]	[19:18]	[17:16]
000H	P1	P2	P3	P4	P5	P6	P7	P8
004H	P17	P18	P19	P20	P21	P22	P23	P24
008H	P33	P34	P35	P36	P37	P38	P39	P40
...								

D	[15:14]	[13:12]	[11:10]	[9:8]	[7:6]	[5:4]	[3:2]	[1:0]
000H	P9	P10	P11	P12	P13	P14	P15	P16
004H	P25	P26	P27	P28	P29	P30	P31	P32
008H	P41	P42	P43	P44	P45	P46	P47	P48
...								

256 PALETTE USAGE (TFT)

Palette Configuration and Format Control

The LCD controller can support the 256 color palette for various selection of color mapping.

The user can select 256 colors from the 24-bit colors through these four formats.

256 color palette consist of the 256(depth) × 24-bit SPSRAM. Palette supports 8:8:8, 6:6:6, 5:6:5(R:G:B), and 5:5:5:1(R:G:B:I) format.

When the user use 5:5:5:1 format, the intensity data(I) is used as a common LSB bit of each RGB data. So, 5:5:5:1 format is same as R(5+I):G(5+I):B(5+I) format.

For example of 5:5:5:1 format, write palette like Table 21-7 and then connect VD pin to TFT LCD panel(R(5+I)=VD[23:19]+VD[18], VD[10] or VD[2], G(5+I)=VD[15:11]+VD[18], VD[10] or VD[2], B(5+I)=VD[7:3]+VD[18], VD[10] or VD[2].) At the last, Set PALFRM register to 0x3.

Table 27-2. 8:8:8 Palette Data Format

IND EX\ Bit Pos.	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00H	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0
01H	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0
.....
FFH	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0
Nu mbe r of VD	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 27-3. 6:6:6 Palette Data Format

IND EX\ Bit Pos.	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00H	-	-	-	-	-	-	R5	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B5	B4	B3	B2	B1	B0
01H	-	-	-	-	-	-	R5	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B5	B4	B3	B2	B1	B0
.....	-	-	-	-	-	-
FFH	-	-	-	-	-	-	R5	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B5	B4	B3	B2	B1	B0
Nu mbe r of VD	-	-	-	-	-	-	23	22	21	20	19	18	15	14	13	12	15	14	7	6	5	4	3	2



Table 27-4. 5:6:5 Palette Data Format

IND EX\ Bit Pos.	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00H	-	-	-	-	-	-	-	-	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
01H	-	-	-	-	-	-	-	-	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
.....	-	-	-	-	-	-	-	-
FFH	-	-	-	-	-	-	-	-	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
Nu mbe r of VD	-	-	-	-	-	-	-	-	23	22	21	20	19	15	14	13	12	11	10	7	6	5	4	3

Table 27-5. 5:5:5:1 Palette Data Format

IND EX\ Bit Pos.	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00H	-	-	-	-	-	-	-	-	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	I
01H	-	-	-	-	-	-	-	-	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	I
.....	-	-	-	-	-	-	-	-
FFH	-	-	-	-	-	-	-	-	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	I
Nu mbe r of VD	-	-	-	-	-	-	-	-	23	22	21	20	19	15	14	13	12	11	7	6	5	4	3	2)

NOTES:

1. VD18, VD10 and VD2 has same output value, I.
2. DATA[31:24] is invalid.

Palette Read/Write

It is prohibited to access Palette memory during the ACTIVE status of the VSTATUS (vertical status) register. When the user going to do Read/Write operation on the palette, VSTATUS register must be checked.

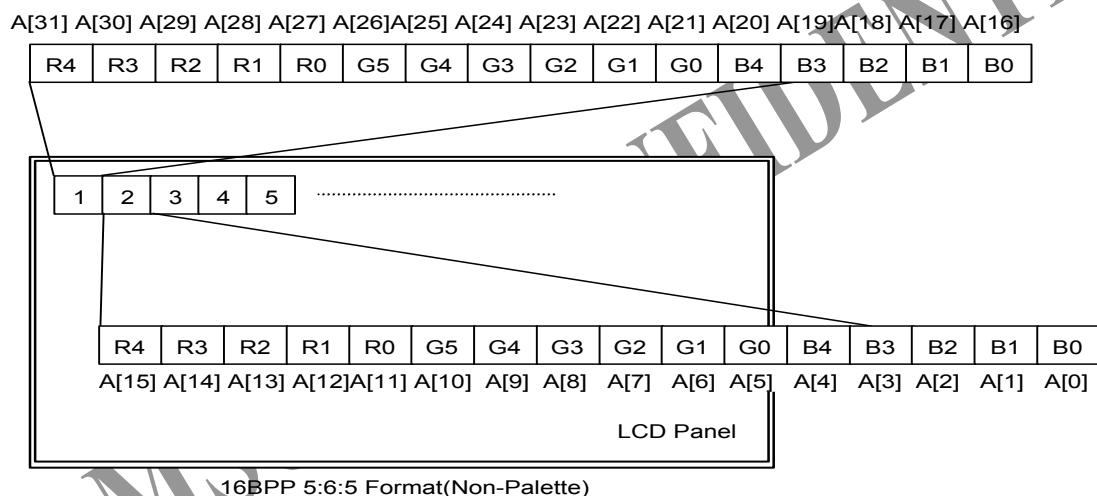
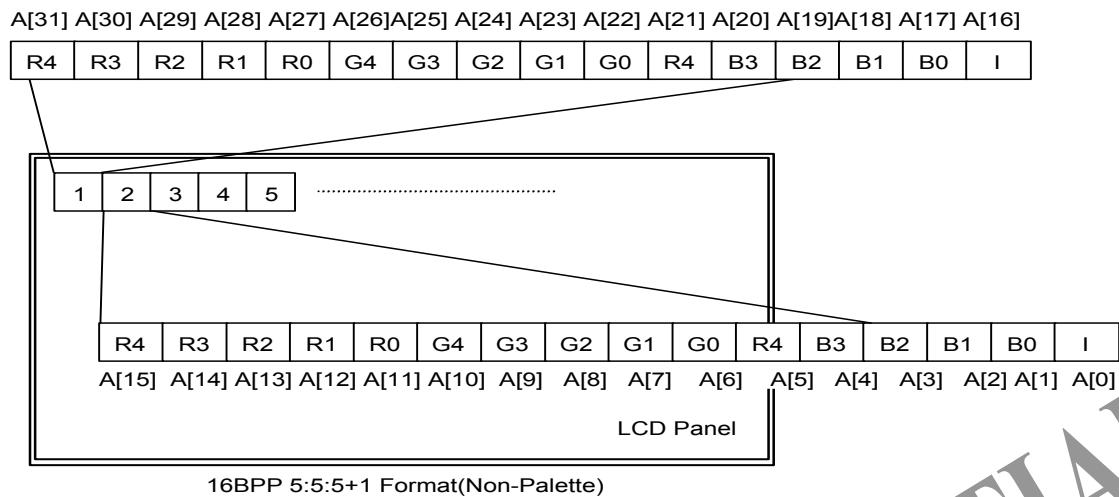


Figure 27-5. 16BPP Display Types

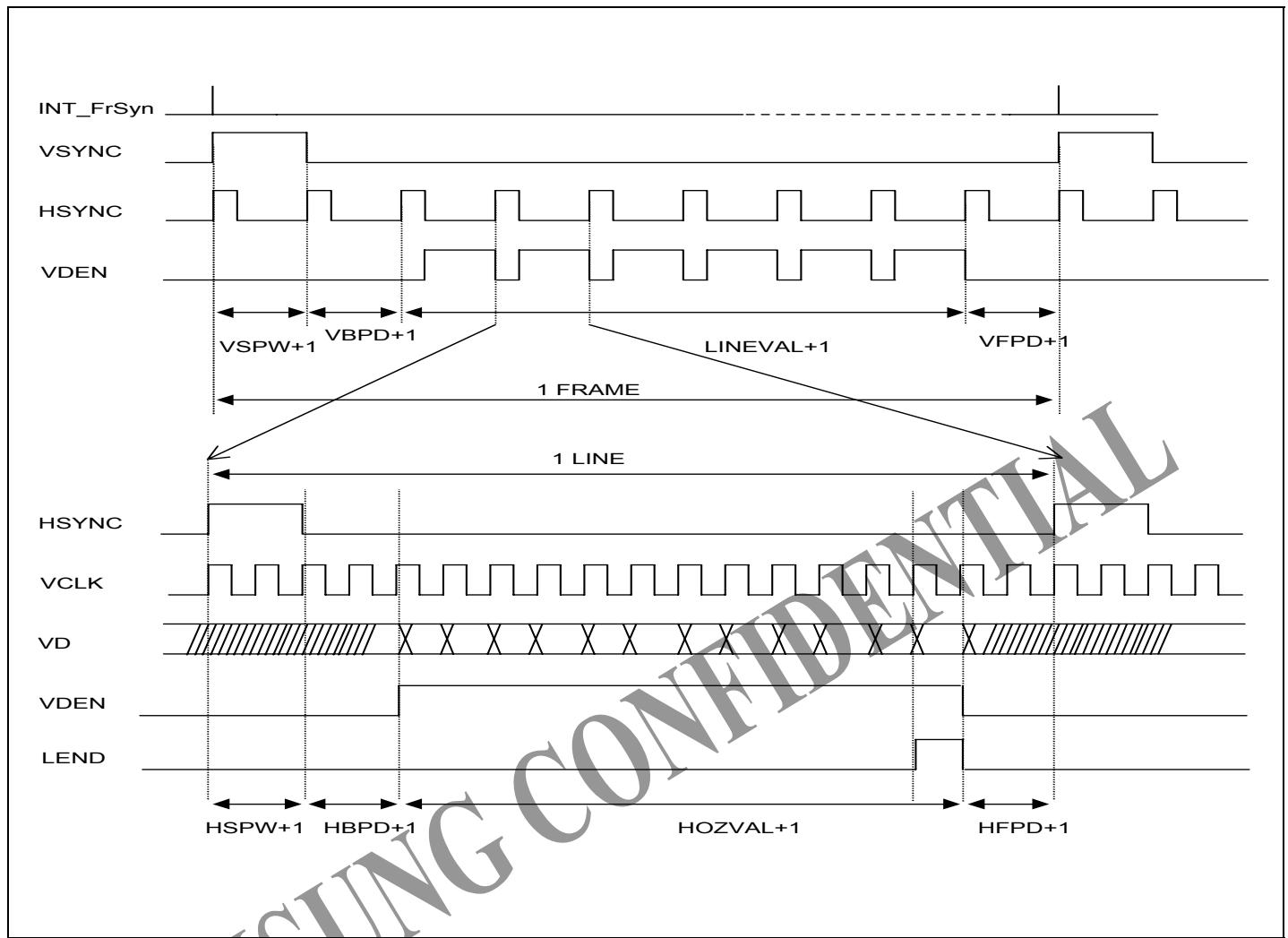


Figure 27-6. TFT LCD Timing Example

VIRTUAL DISPLAY

The LCD controller supports hardware horizontal or vertical scrolling. If the screen is scrolled, the fields of LCDBASEU and LCDBASEL registers need to be changed(refer to Figure 21-8) but not the values of PAGEWIDTH and OFFSIZE. The size of video buffer in which the image is stored should be larger than LCD panel screen size.

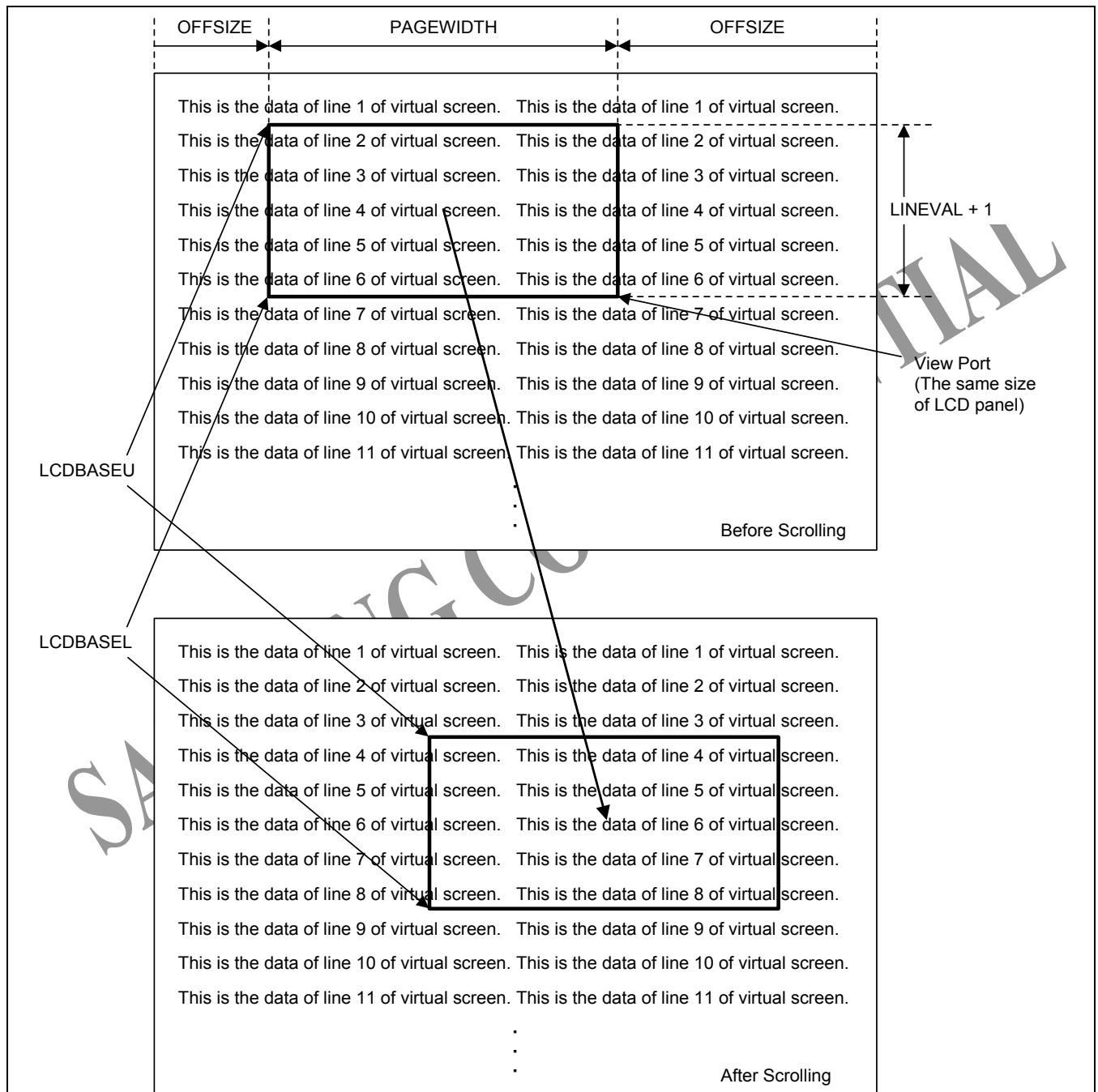


Figure 27-7. Example of Scrolling in Virtual Display

REGISTER DESCRIPTION

MEMORY MAP

Table 27-6. Configuration registers

Register	Address	R/W	Description	Reset Value
LCDCON1	0X39200000	R/W	LCD Control 1	0x00000000
LCDCON2	0X39200004	R/W	LCD Control 2	0x00000000
LCDTCON1	0X39200008	R/W	LCD Time Control 1	0x00000000
LCDTCON2	0X3920000C	R/W	LCD Time Control 2	0x00000000
LCDTCON3	0X39200010	R/W	LCD Time Control 3	0x00000000
LCDOSD1	0X39200014	R/W	LCD OSD Control Register	0x00000000
LCDOSD2	0X39200018	R/W	Foreground image(OSD Image) Left top position set	0x00000000
LCDOSD3	0X3920001C	R/W	Foreground image(OSD Image) Right Bottom position set	0x00000000
LCDB1SADDR1	0X39200020	R/W	Frame Buffer Start Address 1 (Background buffer 1)	0x00000000
LCDB2SADDR1	0X39200024	R/W	Frame Buffer Start Address 2 (Background buffer 2)	0x00000000
LCDF1SADDR1	0X39200028	R/W	Frame Buffer Start Address 1 (foreground buffer 1)	0x00000000
LCDF2SADDR1	0X3920002C	R/W	Frame Buffer Start Address 2 (foreground buffer 2)	0x00000000
LCDB1SADDR2	0X39200030	R/W	Frame Buffer End Address 1 (Background buffer 1)	0x00000000
LCDB2SADDR2	0X39200034	R/W	Frame Buffer End Address 2 (Background buffer 2)	0x00000000
LCDF1SADDR2	0X39200038	R/W	Frame Buffer End Address 1 (foreground buffer 1)	0x00000000
LCDF2SADDR2	0X3920003C	R/W	Frame Buffer End Address 2 (foreground buffer 2)	0x00000000
LCDB1SADDR3	0X39200040	R/W	Virtual Screen Address Set (Background buffer 1)	0x00000000
LCDB2SADDR3	0X39200044	R/W	Virtual Screen Address Set (Background buffer 2)	0x00000000
LCDF1SADDR3	0X39200048	R/W	Virtual Screen Address Set (foreground buffer 1)	0x00000000
LCDF2SADDR3	0X3920004C	R/W	Virtual Screen Address Set (foreground buffer 2)	0x00000000
LCDINTCON	0X39200050	R/W	LCD Interrupt Control	0x00000000
LCDKEYCON	0X39200054	R/W	COLOR KEY Control 1	0x00000000
LCDKEYVAL	0X39200058	R/W	COLOR KEY Control 2	0x00000000
LCDBGCON	0x3920005C	R/W	Back-ground color Control	0x00000000
LCDFGCON	0x39200060	R/W	Fore-ground color Control	0x00000000
LCDDITHCON	0X39200064	R/W	LCD Dithering Control for Active Matrix	0x00000000

LCD CONTROL 1 REGISTER

Register	Address	R/W	Description	Reset Value
LCDCON1	0X00000	R/W	LCD control 1 register	0x00000000

LCDCON1	Bit	Description	Initial State
BURSTLEN	[29:28]	DMA's Burst Length selection : 00 : 16 word- burst 01 : 8 word- burst 10 : 4 word- burst 11 : reserved	0
Reserved	[29:22]	Reserved	0
BDBCON	[21]	Active Frame Select control for back-ground image It will be adopted from next frame data. 0 = Buffer1 1 = Buffer2	0
FDBCON	[20]	Active Frame Select control for fore-ground image (OSD image) It will be adopted from next frame data. 0 = Buffer1 1 = Buffer2	0
DIVEN	[19]	VCLK Divider(CLKVAL) counter enable control bit 0 = Disable (for Power saving) 1 = Enable	0
CLKVAL_F	[18:13]	Determine the rates of VCLK and CLKVAL[5:0]. VCLK = HCLK / [(CLKVAL+1) x 2] (CLKVAL ≥ 0)	0
CLKDIR	[12]	Select the clock source as direct or divide using CLKVAL_F register. 0 = Direct clock (frequency of VCLK = frequency of Clock source) 1 = Divided using CLKVAL_F	0
CLKSEL_F	[11]	Select the Video Clock source 0 = HCLK 1 = UPLL Clock	0
PNRMODE	[10:9]	Select the display mode. 00 = RGB Parallel format (RGB) 01 = RGB Parallel format (BGR) 10 = Serial Format (R->G->B) 11 = Serial Format (B->G->R)	0
BPPMODEF_F (for Fore-ground)	[8:6]	Select the BPP (Bits Per Pixel) mode for fore-ground image (OSD). 011 = 8 bpp (palletized) 100 = 8 bpp (non-palletized, R:3-G:3-B:2) 101 = 16 bpp (non-palletized, R:5-G:6-B:5) 110 = 16 bpp (non-palletized, R:5-G:5-B:5-I:1)	0



		111 = unpacked 24 bpp (non-palletized)	
BPPMODEB_F (for Back-ground)	[5:2]	Select the BPP (Bits Per Pixel) mode for back-ground image. 0000 = 1 bpp 0001 = 2 bpp 0010 = 4 bpp 0011 = 8 bpp (palletized) 0100 = 8 bpp (non-palletized, R:3-G:3-B:2) 0101 = 16 bpp (non-palletized, R:5-G:6-B:5) 0110 = 16 bpp (non-palletized, R:5-G:5-B:5-I:1) 0111 = unpacked 24 bpp (non-palletized) 1xxx = Reserved	0
ENVID	[1]	LCD video output and the logic immediately enable/disable. 0 = Disable the video output and the LCD control signal. 1 = Enable the video output and the LCD control signal.	0
ENVID_F	[0]	LCD video output and the logic enable/disable at current frame end. 0 = Disable the video output and the LCD control signal. 1 = Enable the video output and the LCD control signal. * If you on and off this bit, then you will read "H" and video controller enable until the end of current frame.	0

Note. Per Frame video on-off : ENVID & ENVID_F on-off simultaneously.

Direct video on-off : ENVID on-off only. (where, ENVID_F = 0)

LCD CONTROL 2 REGISTER

Register	Address	R/W	Description	Reset Value
LCDCON2	0X000004	R/W	LCD control 2 register	0x00000000

LCDCON2	Bit	Description	Initial state
LINECNT (read only)	[25:15]	Provide the status of the line counter (read only) Up count from 0 to LINEVAL	0
VSTATUS	[14:13]	Vertical Status (read only). 00 = VSYNC Porch 10 = ACTIVE Porch	01 = BACK 11 = FRONT
HSTATUS	[12:11]	Horizontal Status (read only). 00 = HSYNC Porch 10 = ACTIVE Porch	01 = BACK 11 = FRONT
PALFRM	[10:9]	This bit determines the size of the palette data format 00 = 24 bit (8:8:8) (6:6:6) 10 = 16 bit (5:6:5)	01 = 18 bit 11 = 16 bit

		(5:5:5:1)	
Reserved	[8]	This bit must be "0".	0
IVCLK	[7]	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
IHSYNC	[6]	This bit indicates the HSYNC pulse polarity. 0 = normal 1 = inverted	0
IVSYNC	[5]	This bit indicates the VSYNC pulse polarity. 0 = normal 1 = inverted	0
Reserved	[4]	For future use	0
IVDEN	[3]	This bit indicates the VDEN signal polarity. 0 = normal 1 = inverted	0
BITSWP	[2]	Bit swap control bit. 0 = Swap Disable 1 = Swap Enable	0
BYTSPW	[1]	Byte swap control bit. 0 = Swap Disable 1 = Swap Enable	0
HAWSPW	[0]	Half-Word swap control bit. 0 = Swap Disable 1 = Swap Enable	0

LCD TIME CONTROL 1 REGISTER

Register	Address	R/W	Description	Reset Value
LCDTCON1	0X00008	R/W	LCD control 2 register	0x00000000

LCDTCON2	Bit	Description	Initial State
VBPD	[23:16]	Vertical back porch is the number of inactive lines at the start of a frame, after vertical synchronization period.	0
VFPD	[15:8]	Vertical front porch is the number of inactive lines at the end of a frame, before vertical synchronization period.	0
VSPW	[7:0]	Vertical sync pulse width determines the VSYNC pulse's high level width by counting the number of inactive lines.	0

LCD TIME CONTROL 2 REGISTER

Register	Address	R/W	Description	Reset Value
LCDTCON2	0X0000C	R/W	LCD time control 2 register	0x00000000

LCDTCON2	Bit	Description	Initial state
HBPD	[23:16]	Horizontal back porch is the number of VCLK periods between the falling edge of HSYNC and the start of active data.	0000000
HFPD	[15:8]	Horizontal front porch is the number of VCLK periods between the end of active data and the rising edge of HSYNC.	0X00
HSPW	[7:0]	Horizontal sync pulse width determines the HSYNC pulse's high level width by counting the number of the VCLK.	0X00



LCD TIME CONTROL 3 REGISTER

Register	Address	R/W	Description	Reset Value
LCDTCON3		R/W	LCD time control 3 register	0x00000000
LCDTCON3		Bit	Description	Initial state
LINEVAL		[21:11]	These bits determine the vertical size of LCD panel.	0
HOZVAL		[10:0]	These bits determine the horizontal size of LCD panel.	0

LCD OSD CONTROL 1 REGISTER

Register	Address	R/W	Description	Reset Value
LCDOSD1		R/W	LCD OSD control 1 register	0x00000000
LCDOSD1		Bit	Description	Initial state
OSDEN_F		[9]	OSD(On-screen display) control bit. 0 = OSD Disable 1 = OSD Enable	0
OSD_BLD_PIX		[8]	Select blending category 0 = Per plane blending (8/16/24bpp mode) 1 = Per pixel blending (24bpp only)	0
OSD_ALPHA		[7:0]	8-bit Alpha value defined by Eq. X	0

LCD OSD CONTROL 2 REGISTER

Register	Address	R/W	Description	Reset Value
LCDOSD2		R/W	LCD OSD control 2 register	0x0

LCDOSD2	Bit	Description	initial state
OSD_LeftTopX_F	[21:11]	Horizontal screen coordinate for left top pixel of OSD image	0
OSD_LeftTopY_F	[10:0]	Vertical screen coordinate for left top pixel of OSD image	0

LCD OSD CONTROL 3 REGISTER

Register	Address	R/W	Description	Reset Value
LCDOSD3	0X0001C	R/W	LCD OSD control 3 register	0x0

LCDOSD3	Bit	Description	initial state
OSD_RightBotX_F	[21:11]	Horizontal screen coordinate for right bottom pixel of OSD image	0
OSD_RightBotY_F	[10:0]	Vertical screen coordinate for right bottom pixel of OSD image	0

Note. LCDOSD2 and LCDOSD3 must be word boundary X position.

So, 24 Bpp mode can has X position by 1 pixel. (ex, X = 0,1,2,3....)

16 Bpp mode can has X position by 2 pixel. (ex, X = 0,2,4,6....)

8 Bpp mode can has X position by 4 pixel. (ex, X = 0,4,8,12....)

FRAME BUFFER ADDRESS 1 REGISTER

Register	Address	R/W	Description	Reset Value
LCDF1ADDR1	0X00020	R/W	Frame buffer start address register for Back-Ground buffer 1	0x0
LCDF2ADDR1	0X00024	R/W	Frame buffer start address register for Back-Ground buffer 2	0x0
LCDB1ADDR1	0X00028	R/W	Frame buffer start address register for Fore-Ground (OSD) buffer 1	0x0
LCDB2ADDR1	0X0002C	R/W	Frame buffer start address register for Fore-Ground (OSD) buffer 2	0x0

LCDxADDR1	Bit	Description	Initial State
LCDBANK_F	[31:24]	These bits indicate A[31:24] of the bank location for the video buffer in the system memory.	0
LCDBASEU_F	[23:0]	These bits indicate A[23:0] of the start address of the LCD frame buffer.	0

FRAME BUFFER ADDRESS 2 REGISTER

Register	Address	R/W	Description	Reset Value
LCDF1ADDR2	0X00030	R/W	Frame buffer end address register for Back-Ground buffer 1	0x0
LCDF2ADDR2	0X00034	R/W	Frame buffer end address register for Back-Ground buffer 2	0x0
LCDB1ADDR2	0X00038	R/W	Frame buffer end address register for Fore-Ground (OSD) buffer	0x0



			1	
LCD2ADDR2	0X0003 C	R/W	Frame buffer end address register for Fore-Ground (OSD) buffer 2	0x0

LCDxADDR2	Bit	Description	Initial State
LCDBASEL_F	[23:0]	These bits indicate A[23:0] of the end address of the LCD frame buffer. LCDBASEL = LCDBASEU + (PAGEWIDTH+OFFSIZE) x (LINEVAL+1)	0x0000

FRAME BUFFER ADDRESS 3 REGISTER

Register	Address	R/W	Description	Reset Value
LCDB1ADDR3	0X00040	R/W	Virtual screen address set for Back-Ground buffer 1	0x00000000
LCDB2ADDR3	0X00044	R/W	Virtual screen address set for Back-Ground buffer 2	0x00000000
LCDF1ADDR3	0X00048	R/W	Virtual screen address set for Fore-Ground(OSD) buffer 1	0x00000000
LCDF2ADDR3	0X0004C	R/W	Virtual screen address set for Fore-Ground(OSD) buffer 2	0x00000000

LCDxADDR3	Bit	Description	Initial State
OFFSIZE_F	[25:13]	Virtual screen offset size (the number of byte). This value defines the difference between the address of the last byte displayed on the previous LCD line and the address of the first byte to be displayed in the new LCD line. OFFSIZE_F must has value more than burst size value or 0.	0
PAGEWIDTH_F	[12:0]	Virtual screen page width (the number of byte). This value defines the width of the view port in the frame. PAGEWIDTH must has value which is multiple of the burst size.	0

LCD INTERRUPT CONTROL REGISTER

Register	Address	R/W	Description	Reset Value
LCDINTCON	0X00050	R/W	Indicate the LCD interrupt control register	0x0

LCDINTCON	Bit	Description	Initial state
FRAMESEL0	[11:10]	LCD Frame Interrupt 2 at start of : 00 = BACK Porch 10 = ACTIVE 01 = VSYNC 11 = FRONT Porch	0
FRAMESEL1	[9:8]	LCD Frame Interrupt 1 at start of : 00 = None 10 = VSYNC 01 = BACK Porch 11 = FRONT Porch	0
INTFRMEN	[7]	LCD Frame interrupt Enable control bit. 0 = LCD Frame Interrupt Disable 1 = LCD Frame Interrupt Enable	0

FIFOSEL	[6:5]	FIFO Interrupt select bit 00 = All FIFO (OR case) 01 = Back-ground FIFO only 10 = Fore-ground (OSD) FIFO only	0
FIFOLEVEL	[4:2]	LCD FIFO Interrupt Level Select (1 ~ 128 word) 000 = 32 word left 010 = 96 word left left (OR case) 1xx = reserved	001 = 64 word left 011 = 32/64/96 word
INTFIFOEN	[1]	LCD FIFO interrupt Enable control bit. 0 = LCD FIFO Level Interrupt Disable 1 = LCD FIFO Level Interrupt Enable	0
INTEN	[0]	LCD interrupt Enable control bit. 0 = LCD Interrupt Disable 1 = LCD Interrupt Enable	0

COLOR KEY 1 REGISTER

Register	Address	R/W	Description	Reset Value
KEYCON	0X000054	R/W	Color key control register	0x00000

KEYCON	Bit	Description	Initial state
KEYEN_F	[25]	Color Key (Chroma key) Enable control 0 = color key disable 1 = color key enable	0
DIRCON	[24]	Color key (Chroma key)direction control 0 = If the pixel value match fore-ground image with COLVAL, pixel from back-ground image is displayed (only in OSD area) 1 = If the pixel value match back-ground with COLVAL, pixel from fore-ground image is displayed (only in OSD area)	0
COMPKEY	[23:0]	Each bit is correspond to the COLVAL[23:0]. If some position bit is set then that position bit of COLVAL will be disabled. <i>Compare[0] = IMG[0] ~^ COLVAL[0] & ~COMPKEY[0]</i> <i>Compare[23] = IMG[23] ~^ COLVAL[23] & ~COMPKEY[23]</i> <i>Compare_OK = Compare[23:0]</i>	0



COLOR KEY 2 REGISTER

Register	Address	R/W	Description	Reset Value
COLVAL	0X00058	R/W	Color key value (transparent value) register	0x00000000

COLVAL	Bit	Description	Initial state
COLVAL	[23:0]	Color key value for the transparent pixel effect.	0

Note. COLVAL and COMPKEY use 24bit color data at all bpp mode.

@ BPP24 mode : 24 bit color value is valid.

A. COLVAL

- e Red : COLVAL[23:17]
- e Green: COLVAL[15: 8]
- e Blue : COLVAL[7:0]

B. COMPKEY

- e Red : COMPKEY[23:17]
- e Green: COMPKEY[15: 8]
- e Blue : COMPKEY[7:0]

@ BPP16 (5:6:5) mode : 16 bit color value is valid

A. COLVAL

- e Red : COLVAL[23:19]
- e Green: COLVAL[15: 10]
- e Blue : COLVAL[7:3]

B. COMPKEY

- e Red : COMPKEY[23:19]
- e Green: COMPKEY[15: 10]
- e Blue : COMPKEY[7:3]
- e COMPKEY[18:16] must be 0x7.
- e COMPKEY[9: 8] must be 0x3.
- e COMPKEY[2:0] must be 0x7.
- e COMPKEY register must be set properly for the each bpp mode.

BACK-GROUND COLOR MAP

Register	Address	R/W	Description	Reset Value
BGCON	0X0005C	R/W	Back-Ground color control	0x00000

BGCON	Bit	Description	Initial state
BGCOLEN_F	[24]	Back-Ground color mapping control bit . If this bit is enabled then lcd back-ground DMA will stop, and GBCOLOR will be appear on back-ground image instead of original image. 0 = disable 1 = enable	0
BGCOLOR	[23:0]	Color Value	0

FORE-GROUND COLOR MAP

Register	Address	R/W	Description	Reset Value
FGCON	0X000060	R/W	Fore-Ground color control	0x00000

BGCON	Bit	Description	Initial state
FGCOLEN_F	[24]	Fore-Ground color mapping control bit . If this bit is enabled then lcd fore-ground DMA will stop, and FGCOLOR will be appear on fore-ground image instead of original image. 0 = disable 1 = enable	0
FGCOLOR	[23:0]	Color Value	0

DITHERING CONTROL 1 REGISTER

Register	Address	R/W	Description	Reset Value
DITHMODE	0X000064	R/W	Dithering mode register.	0x00000

DITHMODE	Bit	Description	Initial state
RDithPos	[6:5]	Red Dither bit control 00 : 8bit 01 : 6bit 1x : 5bit	0
GDithPos	[4:3]	Green Dither bit control 00 : 8bit 01 : 6bit 1x : 5bit	0
BDithPos	[2:1]	Blue Dither bit control 00 : 8bit 01 : 6bit 1x : 5bit	0
DITHEN_F	[0]	Dithering Enable bit 0 = dithering disable 1 = dithering enable	0

BACK-GROUND PALETTE RAM ACCESS ADDRESS (NOT SFR)

Register	Address	R/W	Description	Reset Value
00	0X01000	R/W	Back-ground Palette entry 0 address	undefined
01	0X01004	R/W	Back-ground Palette entry 1 address	undefined
-	-	-	-	-
FF	0x013FC	R/W	Back-ground Palette entry 255 address	undefined

FORE-GROUND PALETTE RAM ACCESS ADDRESS (NOT SFR)

Register	Address	R/W	Description	Reset Value
00	0X02000	R/W	Fore-ground Palette entry 0 address	undefined
01	0X02004	R/W	Fore-ground Palette entry 1 address	undefined
-	-	-	-	-
FF	0x023FC	R/W	Fore-ground Palette entry 255 address	undefined

28 ATA CONTROLLER

OVERVIEW

This ATA Controller can control the Hard Disk and the Compact Flash that can operate in true IDE mode.

FEATURES

- Support all ATA PIO modes.
- Support ATA UDMA from mode 0 to mode 4.
- ECC supported for UDMA data transfer.
- Connect up to two devices (HDD or CF card).

The ATA controller directly accesses the system RAM when it implements UDMA data transfer. And it may work in DMA mode when it implements PIO data transfer, which is named the PIO DMA mode, or the PDMA mode. Therefore, there are three operating modes called PIO, PDMA, and UDMA in the ATA Controller.

BLOCK DIAGRAM

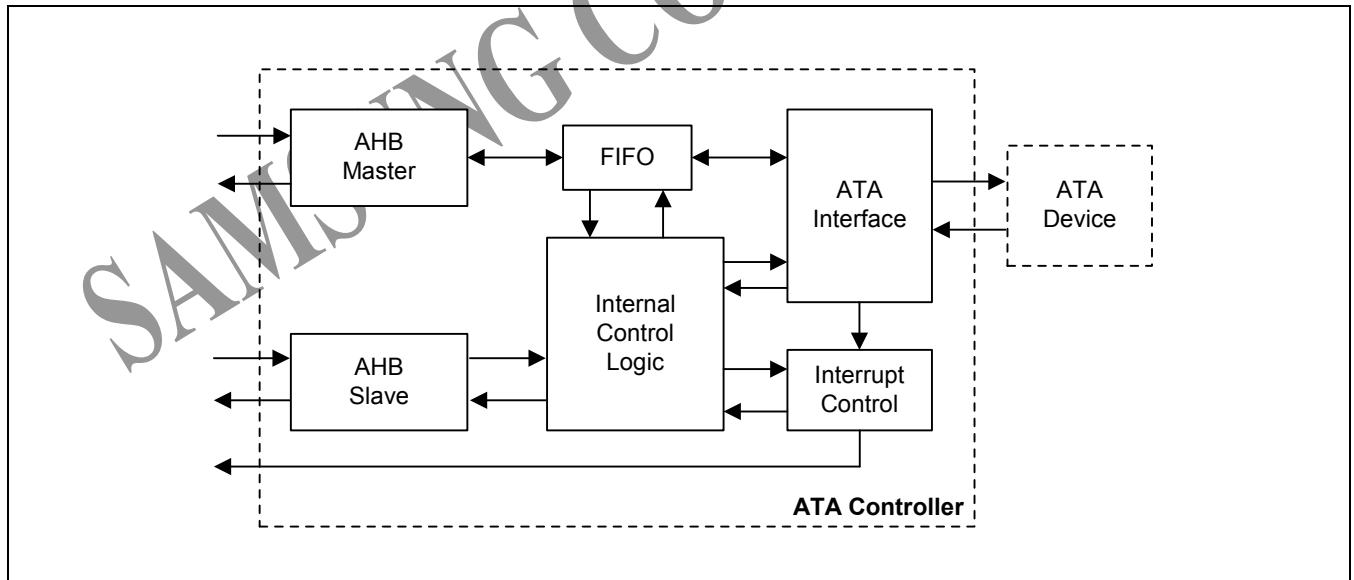


Figure 28-1. Block Diagram

PIN DESCRIPTION

Name	I/O	Description	Name	I/O	Description
HD0	IO	Data bus bit 0	IORDY	I	I/O ready
HD1	IO	Data bus bit 1			DMA ready during Ultra DMA data-out bursts
HD2	IO	Data bus bit 2			Data strobe during Ultra DMA data-in bursts
HD3	IO	Data bus bit 3			
HD4	IO	Data bus bit 4			
HD5	IO	Data bus bit 5	DIOW	O	I/O write
HD6	IO	Data bus bit 6			Stop during Ultra DMA data bursts
HD7	IO	Data bus bit 7			
HD8	IO	Data bus bit 8	DIOR	O	I/O read
HD9	IO	Data bus bit 9			DMA ready during Ultra DMA data-in bursts
HD10	IO	Data bus bit 10			Data strobe during Ultra DMA data-out bursts
HD11	IO	Data bus bit 11			
HD12	IO	Data bus bit 12			
HD13	IO	Data bus bit 13	DMACK	O	DMA acknowledge
HD14	IO	Data bus bit 14	DMARQ	I	DMA request
HD15	IO	Data bus bit 15	ATA_RESET	O	Reset
DA0	O	Device address bit0	ATA_INTRQ	I	Interrupt request
DA1	O	Device address bit1	CBLID	I	Cable assembly type identifier
DA2	O	Device address bit2			
CS0	O	Chip select 0			
CS1	O	Chip select 1			

ATA Controller Register Map

Name	Address	R/W	Reset	Description
ATA_CONTROL	0x38E0_0000	R/W	0x00000000	Enable and clock down status
ATA_STATUS	0x38E0_0004	R	0x01070000	Status
ATA_COMMAND	0x38E0_0008	R/W	0x00000000	Command
ATA_SWRST	0x38E0_000C	R/W	0x00000001	Software reset
ATA_IRQ	0x38E0_0010	R/W	0x00000000	Interrupt sources
ATA_IRQ_MASK	0x38E0_0014	R/W	0x00000000	Interrupt mask
ATA_CFG	0x38E0_0018	R/W	0x00000000	Configuration for ATA interface
ATA PIO_TIME	0x38E0_002C	R/W	0x0001c238	PIO timing
ATA_UDMA_TIME	0x38E0_0030	R/W	0x020b1362	UDMA timing
ATA_XFR_NUM	0x38E0_0034	R/W	0x00000000	Transfer number
ATA_XFR_CNT	0x38E0_0038	R	0x00000000	Current transfer count
ATA_TBUF_START	0x38E0_003C	R/W		Start address of track buffer
ATA_TBUF_SIZE	0x38E0_0040	R/W		Size of track buffer
ATA_SBUF_START	0x38E0_0044	R/W		Start address of Source buffer1
ATA_SBUF_SIZE	0x38E0_0048	R/W		Size of source buffer1
ATA_CADR_TBUF	0x38E0_004C	R		Current write address of track buffer
ATA_CADR_SBUF	0x38E0_0050	R		Current read address of source buffer
ATA_PIO_DTR	0x38E0_0054	R/W		PIO device data register
ATA_PIO_FED	0x38E0_0058	R/W		PIO device Feature/Error register
ATA_PIO_SCR	0x38E0_005C	R/W		PIO sector count register
ATA_PIO_LLR	0x38E0_0060	R/W		PIO device LBA low register
ATA_PIO_LMR	0x38E0_0064	R/W		PIO device LBA middle register
ATA_PIO_LHR	0x38E0_0068	R/W		PIO device LBA high register
ATA_PIO_DVR	0x38E0_006C	R/W		PIO device register
ATA_PIO_CSD	0x38E0_0070	R/W		PIO device command/status register
ATA_PIO_DAD	0x38E0_0074	R/W		PIO control/alternate status register
ATA_PIO_READY	0x38E0_0078	R		PIO data read/write ready
ATA_PIO_RDATA	0x38E0_007C	R		PIO read data from device register
BUS_FIFO_STATUS	0x38E0_0080			Reserved
ATA_FIFO_STATUS	0x38E0_0084			Reserved

ATA Enable and Clock Down Status (ATA_CONTROL)

Bit	Name	R/W	Description
31:2	-	-	-
1	clk_down_ready	R	Status for clock down This bit is asserted in idle state when ATA_CONTROL bit [0] is zero. 0 : not ready for clock down 1 : ready for clock down
0	ata_enable	R/W	ATA enable 0 : ATA is disabled and preparation for clock down maybe in progress 1 : ATA is enabled.

ATA Status (ATA_STATUS)

Bit	Name	R/W	Description
31:6	-	-	-
5	atadev_cblid	R	ATAPI cable identification
4	atadev_irq	R	ATAPI interrupt signal line
3	atadev_iordy	R	ATAPI iordy signal line
2	atadev_dmareq	R	ATAPI dmareq signal line
1:0	xfr_state	R	Transfer state 00 : idle state 01 : transfer state 1x : wait for completion state

ATA Command (ATA_COMMAND)

Bit	Name	R/W	Description
31:2	-	-	-
1:0	xfr_command	R/W	<p>ATA transfer command</p> <p>Four command types (START, STOP, ABORT and CONTINUE) are supported for data transfer control. The “START” command is used to start data transfer. The “STOP” command can pause transfer temporarily. The “CONTINUE” command shall be used after “STOP” command or internal state of “pause” when track buffer is full or UDMA hold state. The “ABORT” command terminated current data transfer sequences and make ATA host controller move to idle state.</p> <p>00 : command stop 01 : command start (Only available in idle state) 10 : command abort 11 : command continue (Only available in pause state)</p>

ATA Software Reset (ATA_SWRST)

Bit	Name	R/W	Description
31:1	-	-	-
0	ata_swrstn	R/W	<p>Software reset for this ATAPI host</p> <p>0: No reset 1: Software reset for all this ATAPI host module</p>

ATA Interrupt Sources (ATA_IRQ)

Bit	Name	R/W	Description
31:5	-	-	-
4	sbuf_empty_int	R/W	When source buffer is empty. CPU can clear this interrupt by writing “1”.
3	tbuf_full_int	R/W	When track buffer is half full. CPU can clear this interrupt by writing “1”.
2	atadev_irq_int	R/W	When ATAPI device generates interrupt. CPU can clear this interrupt by writing “1”.
1	udma_hold_int	R/W	When ATAPI device makes early termination in UDMA class. CPU can clear this interrupt by writing “1”.
0	xfr_done_int	R/W	When all data transfers are finished. CPU can clear this interrupt by writing “1”.

ATA Interrupt Mask (ATA_IRQ_MASK)

Bit	Name	R/W	Description
31:5	-	-	-
4	mask_sbut_empty_int	R/W	0 : mask sbuf_empty_int; disable 1 : unmask sbuf_empty_int; enable
3	mask_tbuf_full_int	R/W	0 : mask tbuf_full_int; disable 1 : unmask tbuf_full_int; enable
2	mask_atadev_irq_int	R/W	0 : mask atadev_irq_int; disable 1 : unmask ata_irq_int; enable
1	mask_udma_hold_int	R/W	0 : mask udma_hold_int; disable 1 : unmask udma_hold_int; enable
0	mask_xfr_done_int	R/W	0 : mask xfr_done_int; disable 1 : unmask xfr_done_int; enable

ATA Configuration of ATA Interface (ATA_CFG)

Bit	Name	R/W	Description
31:10	-	-	-
9	udma_auto_mode	R/W	Determines whether to continue automatically in case of early termination in UDMA mode by device. Please keep this bit zero for the ATA controller does not support AUTO UDMA feature. 0: stay in pause state and wait for CPU's action. 1: continue automatically
8	sbuf_full_mode	R/W	Determines whether to continue automatically when source buffer is empty. This bit should not be changed during runtime operation. 0: continue automatically with new source buffer address. 1: stay in pause state and wait for CPU's action.
7	tbuf_full_mode	R/W	Determines whether to continue automatically when track buffer is full. This bit should not be changed during runtime operation. 0: continue automatically with new track buffer address. 1: stay in pause state and wait for CPU's action.
6	byte_swap	R/W	Determines whether data endian is little or big in 16bit data. 0 : little endian (ata_data[7:0],ata_data[15:8]) 1 : big endian (ata_data[15:8],ata_data[7:0])
5	atadev_irq_al	R/W	Device interrupt signal level 0: active high 1: active low
4	dma_dir	R/W	DMA transfer direction 0 : Host read data from device 1 : Host write data to device
3:2	ata_class	R/W	ATA transfer class select 00 : transfer class is PIO 01 : transfer class is PIO DMA 1x : transfer class is UDMA
1	ata_iordy_en	R/W	Determines whether IORDY input can extend data transfer. 0 : IORDY disable(ignored) 1 : IORDY enable (can extend)
0	ata_RST	R/W	ATAPI device reset by this host. 0 : no reset 1 : reset

ATA PIO Timing (ATA_PIO_TIME)

Bit	Name	R/W	Description
31:20	—	—	—
19:12	pio_teoc	R/W	PIO timing parameter, teoc, end of cycle time teoc = [pio_teoc] / [bus frequency] Shall not have zero value.
11:4	pio_t2	R/W	PIO timing parameter, t2, DIOR/Wn pulse width t2 = [pio_t2] / [bus frequency] Shall not have zero value.
3:0	pio_t1	R/W	PIO timing parameter, t1, address valid to DIOR/Wn t1 = [pio_t1] / [bus frequency]

ATA UDMA Timing (ATA_UDMA_TIME)

Bit	Name	R/W	Description
31:28	—	—	—
27:24	udma_tdvh	R/W	UDMA timing parameter tDVH tDVH = [udma_tdvh] / [bus frequency]
23:16	udma_tdvs	R/W	UDMA timing parameter tDVS tDVS = [udma_tdvs] / [bus frequency] Shall not have zero value.
15:8	udma_trp	R/W	UDMA timing parameter tRP tRP = [udma_trp] / [bus frequency]
7:4	udma_tss	R/W	UDMA timing parameter, tSS tSS = [udma_tss] / [bus frequency]
3:0	udma_tackenv	R/W	UDMA timing parameter tENV(envelope time(from DMACKn to STOP and HDMARDYn), tACK(setup and hold time for DMACKn) tENV = tACK = [udma_tackenv] / [bus frequency]

ATA Transfer Number (ATA_XFR_NUM)

Bit	Name	R/W	Description
31:1	xfr_num	R/W	Data transfer number.
0			Reserved

ATA Current transfer Count (ATA_XFR_CNT)

Bit	Name	R/W	Description
31:1	xfr_cnt	R	Current remaining transfer counter. This value counts down from ATA_XFR_NUM. It goes to zero when pre-defined all data has been transferred.
0			Reserved

ATA Start Address of Track Buffer (ATA_TBUF_START)

Start address of track buffer when host read data from device

Bit	Name	R/W	Description
31:2	track_buffer_start	R/W	Start address of track buffer when host read data from device (4byte unit)
1:0	-	R	00

ATA Size of Track Buffer (ATA_TBUF_SIZE)

Bit	Name	R/W	Description
31:5	track_buffer_size	R/W	Size of track buffer when host read data from device (32byte unit)
4:0	-	R	00000

ATA Start Address of Source Buffer 1 (ATA_SBUF_START)

Bit	Name	R/W	Description
31:2	source_buffer_start	R/W	Start address of source buffer when host write data to device (4byte unit)
1:0	-	R	00000

ATA Size of Source Buffer1 (ATA_SBUF1_SIZE)

Bit	Name	R/W	Description
31:5	source_buffer_size	R/W	Size of source buffer when host write data to device (32byte unit)
4:0			Reserved

ATA Current Write Address of Track Buffer (ATA_CADR_TBUF)

Bit	Name	R/W	Description
31:2	track_buf_cur_adr	R	Current address of track buffer
1:0			Reserved

ATA Current Read Address of Source Buffer (ATA_CADR_SBUF)

Bit	Name	R/W	Description
31:2	source_buf_cur_adr	R	Current address of source buffer
1:0			Reserved

ATA PIO Device Data Register (ATA_PIO_DTR)

Bit	Name	R/W	Description
31:16	-	-	-
15:0	pio_dev_dtr	R/W	16-bit PIO data register.

ATA PIO Device Feature/Error Register (ATA_PIO_FED)

Bit	Name	R/W	Description
31:8	-	-	-
7:0	pio_dev_fed	R/W	8-bit PIO device feature, error (command block) register

ATA PIO Sector Count Register (ATA_PIO_SCR)

Bit	Name	R/W	Description
31:8	-	-	-
7:0	pio_dev_scr	R/W	8-bit PIO device sector count (command block) register

ATA PIO Device LBA Low Register (ATA_PIO_LLR)

Bit	Name	R/W	Description
31:8	—	—	—
7:0	pio_dev_llr	R/W	8-bit PIO device LBA low (command block) register

ATA PIO Device LBA Middle Register (ATA_PIO_LMR)

Bit	Name	R/W	Description
31:8	—	—	—
7:0	pio_dev_lmr	R/W	8-bit PIO device LBA mid (command block) register

ATA PIO Device LBA High Register (ATA_PIO_LHR)

Bit	Name	R/W	Description
31:8	—	—	—
7:0	pio_dev_lhr	R/W	8-bit PIO device LBA high (command block) register

ATA PIO Device Register (ATA_PIO_DVR)

Bit	Name	R/W	Description
31:8	—	—	—
7:0	pio_dev_dvr	R/W	8-bit PIO device (command block) register bit 4 (DEV) : Cleared to zero selects Device 0. Set to one selects Device 1.

ATA PIO Device Command/Status Register (ATA_PIO_CSD)

Bit	Name	R/W	Description
31:8	—	—	—
7:0	pio_dev_dvr	R/W	<p>Write:</p> <p>8-bit PIO device command (command block) register</p> <p>Read: (status register)</p> <p>7 : BSY 6 : DRDY 5 : DF 4 : # 3 : DRQ 2 : # 1 : # 0 : ERR</p>

ATA PIO Device Control/Alternate Status Register (ATA_PIO_DAD)

Bit	Name	R/W	Description
31:8	—	—	—
7	HOB (High Order Byte)	R/W	Set to one for 480-bit Address feature.
6:3	—	R/W	—
2	SRST	R/W	Software reset bit
1	nIEN	R/W	It enables the device asserts INTRQ to the ATA controller if it is zero.
0	0	R/W	Bit 0 shall be cleared to zero.

ATA PIO Data Read/Write Ready (ATA_PIO_READY)

Bit	Name	R/W	Description
31:2	-	-	-
1	dev_acc_ready	R	Indicates whether host can start access to device register 0 : not ready to start access ATA device register 1 : ready to start access ATA device register
0	pio_data_ready	R	Indicates whether data is valid in ATA_PIO_DATA register 0 : no valid data in ATA_PIO_DATA register 1 : valid data in ATA_PIO_DATA register

ATA PIO Read Data from Device Data Register (ATA_PIO_RDATA)

Bit	Name	R/W	Description
31:16	-	-	-
15:0	pio_rdata	R	PIO read data register while HOST read from ATA device register

ACCESSING THE ATA DEVICE COMMAND BLOCK REGISTERS

There is an address map for ATA device CBR to read status from a device or write a command to a device. In order to program these registers, the ATA_PIO_READY shall be read to determine whether the ATA controller is ready to access a device's CBR.

For a reading access, the ATA_PIO_READY shall also be checked first. Furthermore, after performing the read command, the ATA_PIO_READY checking is required once more to find whether the data from a device is available. Finally a host processor reads the data via the ATA_PIO_RDATA.

To perform a PIO data transfer phase, the Device Data Register works in the same way as the CBR accessing.

PIO TRANSFER MODE

Step 1: initialization

Various transfer protocols can be implemented when the ATA controller core registers are programmed. The ATA controller supports PIO (PDMA) and UDMA protocol. Whereas the user shall know which modes the attached devices support.

Tow things are concerned for programming the ATA_PIO_TIME. One is the PIO transfer mode and the other is bus operating frequency. The timing parameters shall not violate ATA timing requirements.

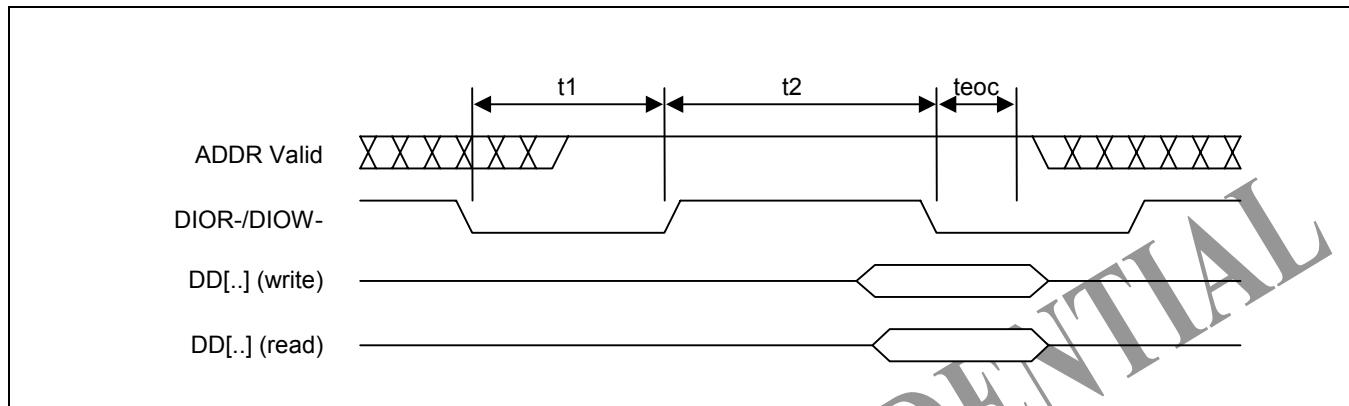


Figure 28-2. PIO Timing

Table 28-1. ATA PIO Timing Parameters ($t_0 = t_1 + t_2 + t_{eoc}$)

	PIO Mode 0	PIO Mode 1	PIO Mode 2	PIO Mode 3	PIO Mode 4
t_0 (min)	600	383	330	180	120
t_1 (min)	70	50	30	30	25
t_2 (min)	290	290	290	80	70

Table 28-2. ATA_PIO_TIME Register Value

PIO Modes	Bus Frequency		
	100 MHz	50 MHz	33 MHz
PIO Mode 0	0x000_12_20_8	0x000_0a_0e_3	0x000_07_09_2
PIO Mode 1	0x000_03_1c_4	0x000_01_0e_2	0x000_02_09_1
PIO Mode 2	0x000_01_1c_2	0x000_01_0e_1	0x000_01_09_1
PIO Mode 3	0x000_06_07_2	0x000_05_03_1	0x000_02_02_1
PIO Mode 4	0x000_02_06_2	0x000_01_03_1	0x000_01_02_1
note	ATA_PIO_TIME = 0x000_teoc_t2_t1		

After programming the timing parameters, the interrupt registers shall be initialized to complete the PIO initialization step.

Step 2: sending command

Write the data to the ATA Device Command Block Registers to start a PIO command. The ATA_PIO_READY register shall be read in advance to determine if the host can start access to a device register.

Step 3: check status

The ATA Controller may wait the device interrupt if the device interrupt function is enabled. When it receives the interrupt from a device, the ATA Controller checks the device's status, and then go to step 4 to perform a data transfer phase, or go to step 5 to complete the PIO command. If the device interrupt is disabled, the ATA controller checks the device status directly if it is necessary.

Normally a status checking phase is between the command block transfer phase and the data transfer phase. And the device status shall be checked when a command is completed to find the command finished correctly or not. It may also be implemented at the completion of a DRQ data block transfer if all the data for the command has not been transferred.

Step 4: handling the data transfer (optional)

If this step is entered, that means the command contains a data transfer phase. The ATA controller then accesses the ATA_PIO_DTR to perform the data transmission. Please refer to the [Accessing the ATA Device Command Block Registers](#) for more information.

Step 5: command completion

Check status after all data transfer has completed. See step 3.

PDMA TRANSFER MODE

Step 1: initialization

The PDMA transfer mode uses the same timing parameters as PIO transfer mode. In order to access the system RAM via AHB master, the RAM base address and the Buffer size shall be initialized. There two set of these registers named Track Buffer Registers and Source Buffer Registers. For data in command, only track buffer shall be initialized, otherwise the source buffer.

The ATA_IRQ_MASK shall be initialized corresponding to the DMA direction.

Step 2: sending command

See PIO Transfer Mode.

Step 3: check status

See PIO Transfer Mode.

Step 4: handling the data transfer

The ATA controller automatically transfers data between the RAM buffer and the selected device. The Host CPU shall write the ATA_COMMAND register to control the DMA START, STOP, CONTINUE, or ABORT. When the INTRQ from device has been detected, the ATA controller shall abort the PDMA data transfer, and go to step 3 to check the device status. After the device status checking, a START command shall be written to the ATA_COMMAND register to complete the rest data transfer. In this case, the transfer mode shall alter from PDMA and PIO, for the status checking works under the PIO mode.

Step 5: command completion

See PIO Transfer Mode.

UDMA TRANSFER MODE

Step 1: initialization

Initializing the UDMA Timing Parameters, DMA buffer registers, and interrupt mask register.

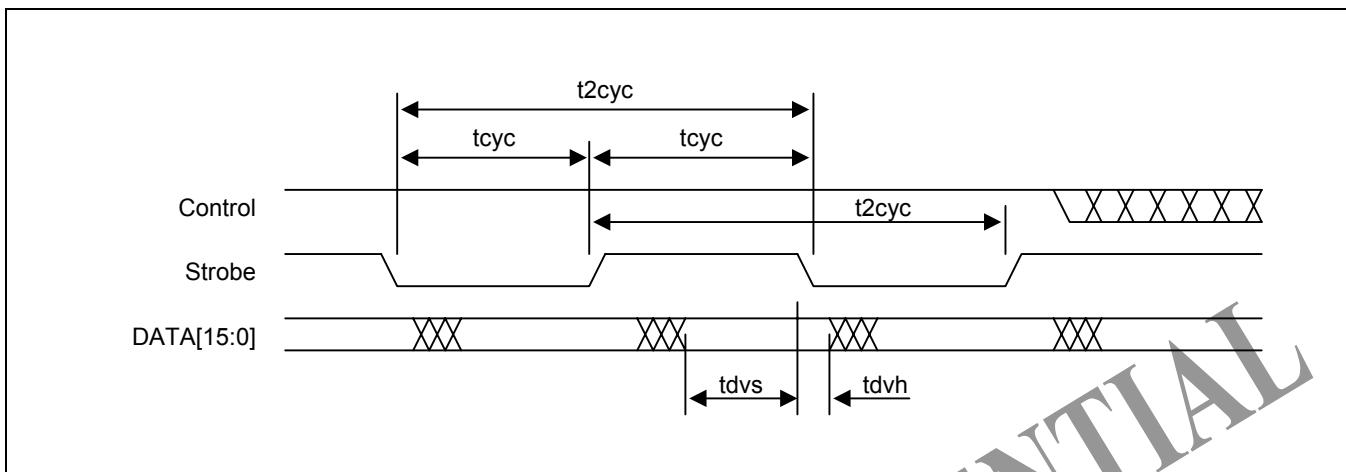


Figure 28-3. UDMA Timing

Table 28-3. ATA UDMA Timing Parameters (ns)

	UDMA Mode 0	UDMA Mode 1	UDMA Mode 2	UDMA Mode 3	UDMA Mode 4
t2cyc (min)	240	160	120	90	60
tcyc (min)	112	73	54	39	25
tdvs (min)	70	48	31	20	6.7
tdvh (min)			6.2		
trp (min)	160	125	100	100	100
tack (min)			20		
tss (min)			50		

Table 28-4. ATA_UDMA_TIME Register Value

UDMA Modes	Bus frequency		
	100 MHz	50 MHz	33 MHz
UDMA Mode 0	0x000_12_20_8	0x000_0a_0e_3	0x000_07_09_2
UDMA Mode 1	0x000_03_1c_4	0x000_01_0e_2	0x000_02_09_1
UDMA Mode 2	0x000_01_1c_2	0x000_01_0e_1	0x000_01_09_1
UDMA Mode 3	0x000_06_07_2	0x000_05_03_1	0x000_02_02_1
UDMA Mode 4	0x000_02_06_2	0x000_01_03_1	0x000_01_02_1
note	ATA_UDMA_TIME = 0x0_tdhv_tdvs_trp_tss_tack		

Step 2: send command

See PIO Transfer Mode.

Step 3: check status

See PIO Transfer Mode.

Step 4: handling the data transfer

See PDMA Transfer Mode.

Step 5: command completion

See PIO Transfer Mode.

SAMSUNG CONFIDENTIAL

29 CHIP ID

OVERVIEW

In the future MP3 audio providers might request that users should submit unique ID to download MP3 music on the net. That is why each die possesses one unique ID and users use the ID. CPU can read the unique ID, which is stored in a fuse box of Die. By reading data in two registers, CPU can capture 40-bit length of the unique ID. REG_ONE register captures the most right 32 bit from the fuse box and REG_TWO register then receives the next 8 bits following the 32 bits. CPU must read two registers 90 clock cycles after reset is high. In other words, 90 clock cycles is time requirement to read the unique ID in two registers.

REGISTERS

Name	Width	Address	R/W	Description
REG_ONE	32	0x3D10_0000	R/W	Receive the first 32 bits from a fuse box
REG_TWO	32	0x3D10_0004	R/W	Receive the other 8 bits from a fuse box

NOTES

SAMSUNG CONFIDENTIAL

30 ELECTRICAL DATA

ABSOLUTE MAXIMUM RATINGS

Table 30-1. Absolute Maximum Rating

Symbol	Parameter	Rating			Unit
			Min	Max	
V_{DD}	DC Supply Voltage	1.2V V_{DD}	- 0.5	1.8	V
		3.3V V_{DD}	- 0.5	4.8	
V_{IN}	DC Input Voltage	3.3V input buffer	- 0.5	4.8	V
		3.3V interface/5V tolerant input buffer	- 0.5	6.5	
V_{OUT}	DC Output Voltage	3.3V output buffer	- 0.5	4.8	V
		3.3V interface/5V tolerant output buffer	- 0.5	6.5	
I_{IO}	Input/Output current	\pm 20 mA			mA
T_A	Storage temperature	- 65 to 150			°C

RECOMMENDED OPERATING CONDITIONS

Table 30-2. Recommended Operating Conditions

Symbol	Parameters	Rating			Unit
			Min	Max	
V_{DD}	DC Supply Voltage for internal ($= V_{DDIN}$)	1.2V V_{DD}	1.1	1.3	V
	DC Supply Voltage for I/O block ($= V_{DDIO}$)	3.3V V_{DD}	3.0	3.6	
	DC Supply Voltage for Analog core ($= V_{DDA}$)	3.3V V_{DD}	3.3 - 5%	3.3 + 5%	
V_{IN}	DC Input Voltage	3.3V Input buffer	-0.3	$V_{DDIO} + 0.3$	V
		3.3V interface/5V tolerant Input buffer	-0.3	5.5	
V_{OUT}	DC Output Voltage	3.3V Output buffer	-0.3	$V_{DDIO} + 0.3$	V
		3.3V interface/5V tolerant Output buffer	-0.3	5.5	
T_A	Temperature Range		-20 to 70	oC	



ELECTRONICS

D.C. ELECTRICAL CHARACTERISTICS

Table 30-3. Normal I/O PAD D.C. Electrical Characteristics

($V_{DD} = 3.3V \pm 0.3V$, $T_A = -20$ to $70^\circ C$, $V_{EXT} = 5V \pm 0.25$ (In Case of 5V Tolerant))

Symbol	Parameters	Condition	Min	Type	Max	Unit
V_{IH}	High level input voltage		2.0			V
	LVCMS interface					
V_{IL}	Low level input voltage			0.8		V
	LVCMS interface					
VT	Switching threshold			$0.5V_{DD}$		V
$VT+$	Schmitt trigger, positive-going threshold	CMOS			2.0	V
$VT-$	Schmitt trigger, negative-going threshold	CMOS	0.8			
I_{IH}	High level input current					μA
	Input buffer					
	Input buffer with pull-down					
I_{IL}	Low level input current					μA
	Input buffer					
	Input buffer with pull-up					
V_{OH}	High level output voltage					V
	Type B1 to B12		$I_{OH} = -1 \mu A$	$V_{DD}-0.05$		
	Type B1		$I_{OH} = -1 mA$	2.4		
	Type B2		$I_{OH} = -2 mA$			
	Type B4		$I_{OH} = -4 mA$			
	Type B8		$I_{OH} = -8 mA$			
	Type B12		$I_{OH} = -12 mA$			
V_{OL}	Low level output voltage					V
	Type B1 to B12		$I_{OL} = 1 \mu A$			
	Type B1		$I_{OL} = 1 mA$			
	Type B2		$I_{OL} = 2 mA$			
	Type B4		$I_{OL} = 4 mA$			
	Type B8		$I_{OL} = 8 mA$			
I_{OZ}	Tri-state output leakage current	$V_{OUT} = V_{SS}$ or V_{DD}	-10		10	μA
I_{DD}	Quiescent supply current				100	μA
C_{IN}	Input capacitance	Any Input and Bi-directional Buffers			5	pF

C_{OUT}	Output capacitance	Any Output Buffers			5	pF
-----------	--------------------	--------------------	--	--	---	----

Table 30-4. A.C. Electrical Characteristics(T_A = -20°C to +70°C, V_{DD} = 3.0 V to 3.6 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Interrupt input high, low width	t _{INTL} , t _{INTH}	P1.0–P1.7 at V _{DD} = 3 V	200	—	—	ns
nRESET input low width	t _{RSL}	V _{DD} = 3 V	10	—	—	us

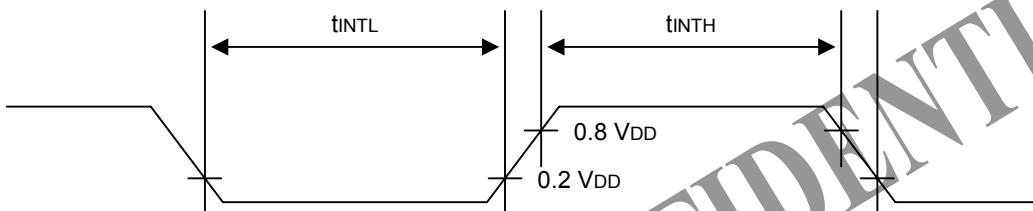
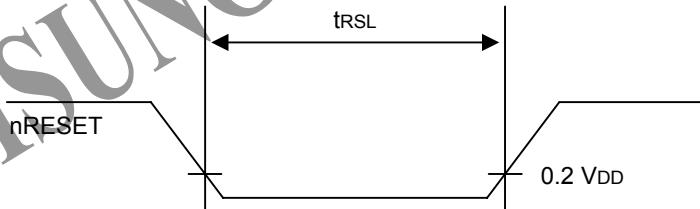
NOTE: User must keep a larger value than the Min value.**Figure 30-1. Input Timing for External Interrupts (Port 4, Port5)****Figure 30-2. Input Timing for nRESET**

Table 30-5. Input/Output Capacitance(TA = - 20 °C to + 70 °C, V_{DD} = 0 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input capacitance	C _{IN}	f = 1 MHz; unmeasured pins are returned to V _{SS}	—	—	10	pF
Output capacitance	C _{OUT}		—	—	—	—
I/O capacitance	C _{IO}		—	—	—	—

Table 30-6. A/D Converter Electrical Characteristics(TA = - 20 °C to + 70 °C, V_{DD} = 3.0 V to 3.6 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Resolution	—	—	—	8	—	bit
Total accuracy	—	V _{DD} = 3.3 V Conversion time = 20 us	—	—	± 2	LSB
Integral Linearity Error	ILE	A _{VREF} = 3.3 V	—	—	± 1	
Differential Linearity Error	DLE	A _{VSS} = 0 V	—	—	± 1	
Offset Error of Top	EOT	—	—	± 1	± 2	
Offset Error of Bottom	EOB	—	—	± 0.5	± 2	
Conversion time (1)	t _{CON}	—	20	—	—	us
Analog input voltage	V _{IAN}	—	A _{VSS}	—	A _{VREF}	V
Analog input impedance	R _{AN}	—	2	1000	—	MΩ
Analog reference voltage	A _{VREF}	—	V _{DD}	—	V _{DD}	V
Analog ground	A _{VSS}	—	V _{SS}	—	V _{SS}	V
Analog input current	I _{ADIN}	A _{VREF} = V _{DD} = 3.3 V	—	—	10	uA
Analog block current (2)	I _{ADC}	A _{VREF} = V _{DD} = 3.3 V		1	3	mA
		A _{VREF} = V _{DD} = 3 V		0.5	1.5	mA

NOTES:

- 'Conversion time' is the time required from the moment a conversion operation starts until it ends.
- I_{ADC} is an operating current during A/D conversion.

Table 30-7. I²S Master Transmitter with Data Rate of 2.5 MHz (10%) (Unit: ns)

Parameter	Min	Typ	Max	Condition
Clock period T	360	400	440	Ttr = 360
Clock HIGH tHC	160	—	—	min > 0.35T = 140 (at typical data rate)
Clock LOW tLC	160	—	—	min > 0.35T = 140 (at typical data rate)
Delay tdtr	—	—	300	max < 0.80T = 320 (at typical data rate)
Hold time thtr	100	—	—	min > 0
Clock rise-time tRC	—	—	60	max > 0.15T = 54 (at relevant in slave mode)

Table 30-8. I²S Slave Receiver with Data Rate of 2.5 MHz (10%) (Unit: ns)

Parameter	Min	Typ	Max	Condition
Clock period T	360	400	440	Ttr = 360
Clock HIGH tHC	110	—	—	min < 0.35T = 126
Clock LOW tLC	110	—	—	min < 0.35T = 126
Set-up time tsr	60	—	—	min < 0.20T = 72
Hold time thtr	0	—	—	min < 0

Table 30-9. Data Retention Supply Voltage in Stop Mode(TA = -20°C to +70°C, V_{DD} = 3.0 V to 3.6V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data retention supply voltage	V _{DDDR}	Normal operation	2	—	3.6	V
Data retention supply current	I _{DDDR}	V _{DDDR} = 2 V	—	—	1	µA

NOTE: Supply current does not include a current which drawn through internal pull-up resistors or external output current loads.



ELECTRONICS

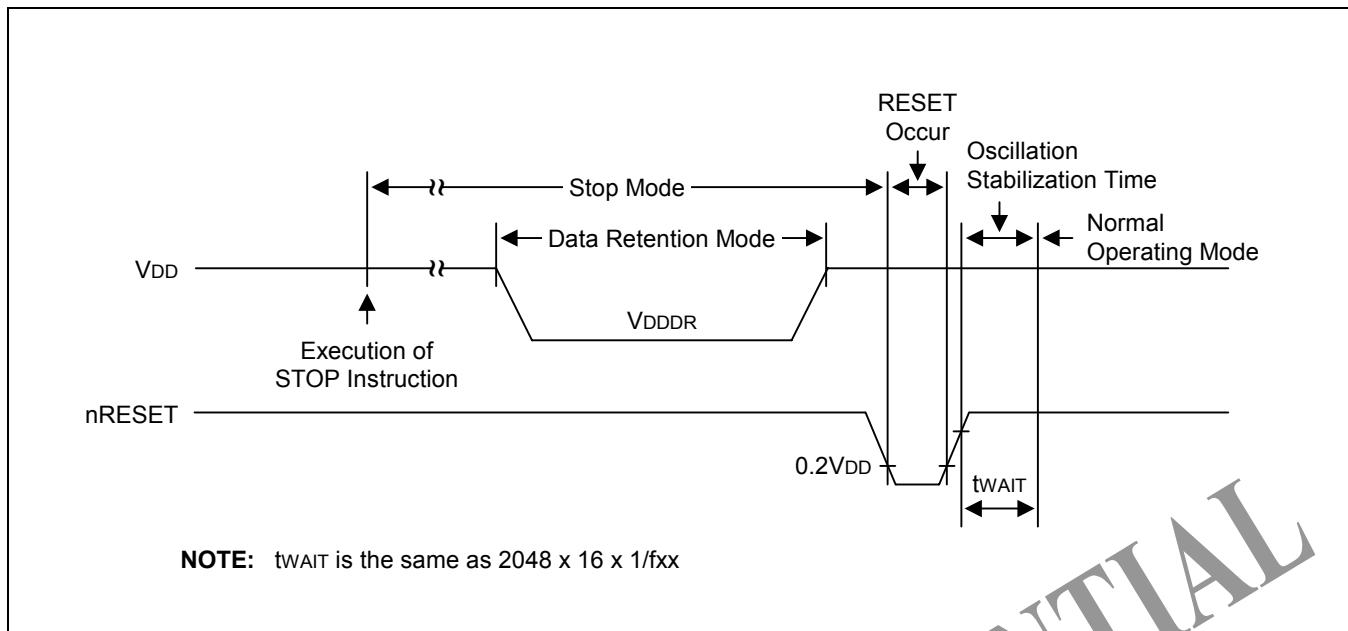


Figure 30-3. Stop Mode Release Timing When Initiated by a RESET

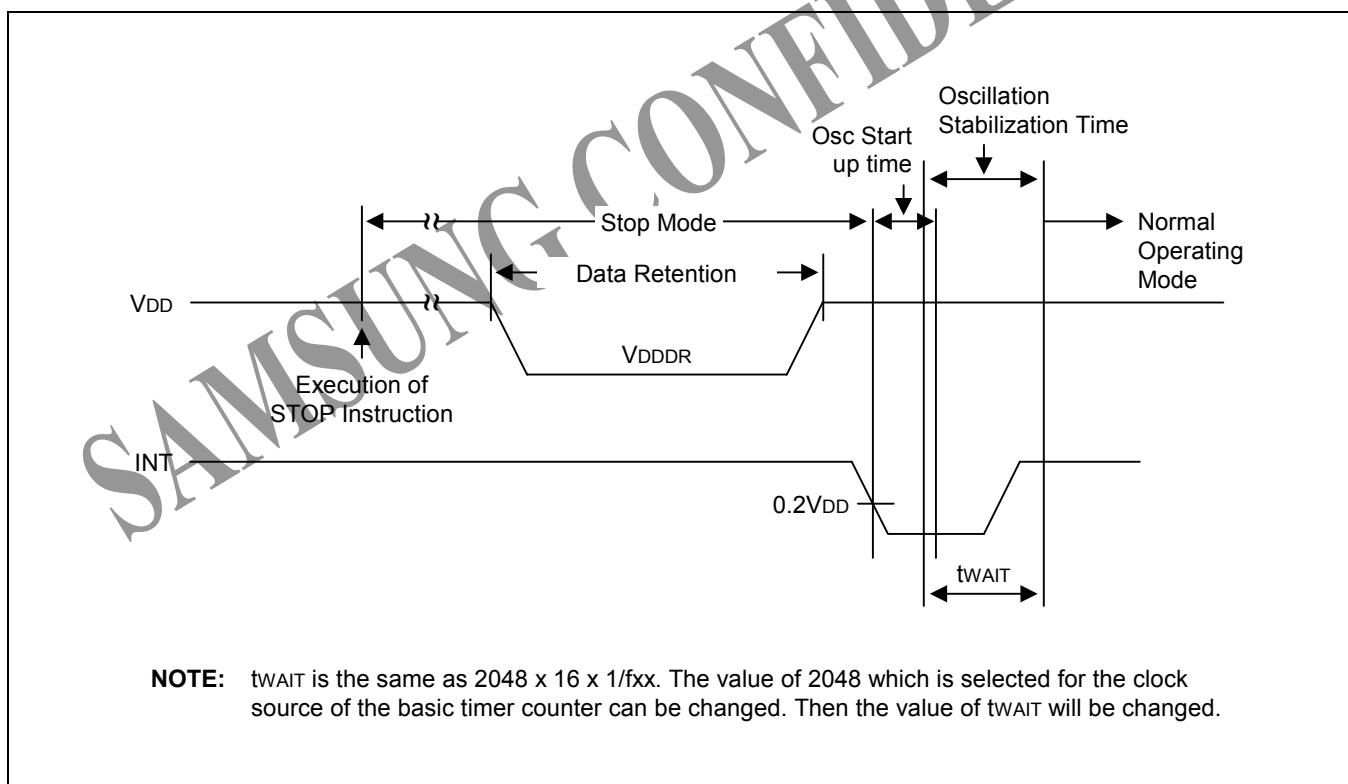


Figure 30-4. Stop Mode Release Timing When Initiated by Interrupts

Table 30-10. Synchronous SIO Electrical Characteristics(TA = -20°C to +70°C V_{DD} = 3.0 V to 3.6 V, V_{SS} = 0 V, f_{xx} = 80 MHz oscillator)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
SCK Cycle time	t _{CYC}	—	200	—	—	ns
Serial Clock High Width	t _{SCKH}	—	60	—	—	
Serial Clock Low Width	t _{SCKL}	—	60	—	—	
Serial Output data delay time	t _{OD}	—	—	—	50	
Serial Input data setup time	t _{ID}	—	40	—	—	
Serial Input data Hold time	t _{IH}	—	100	—	—	

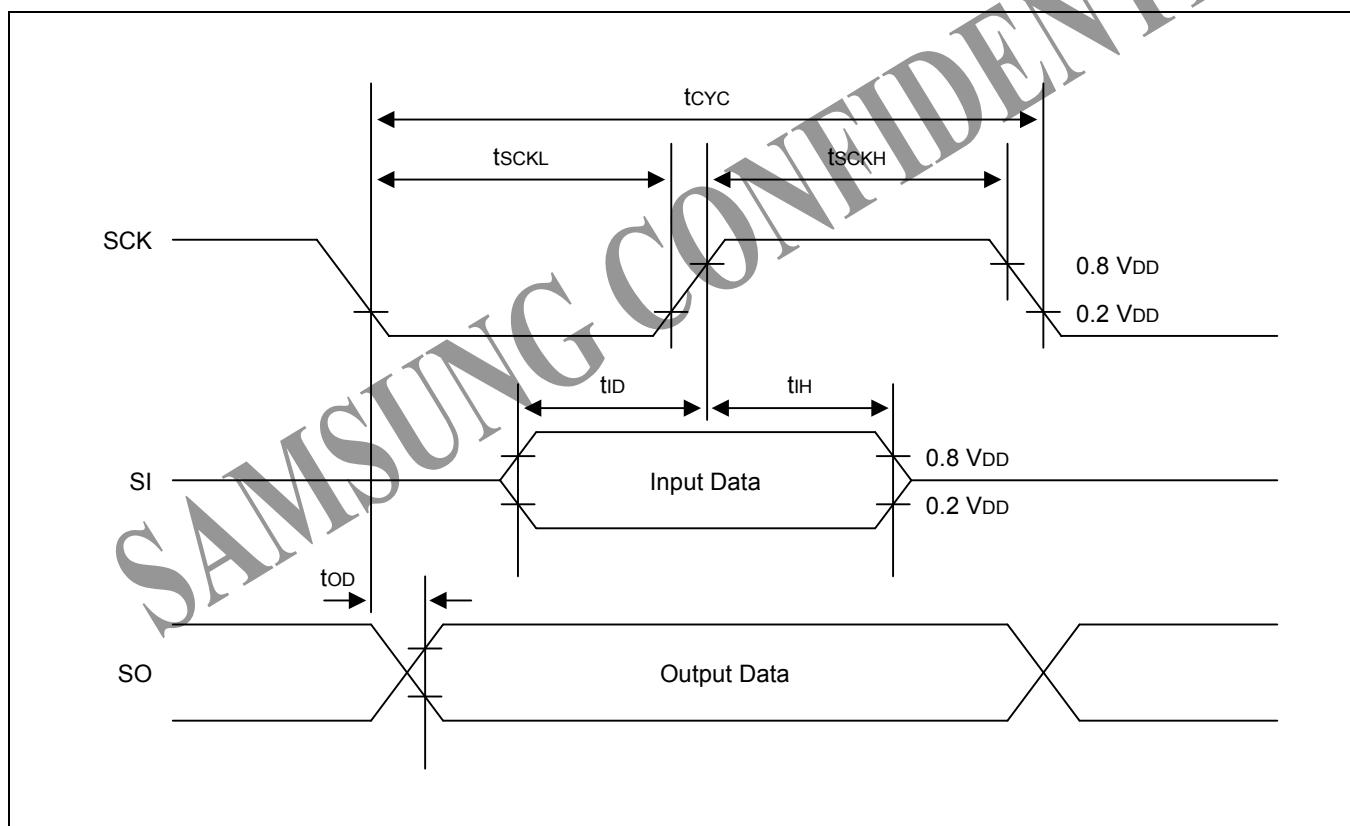
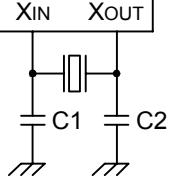
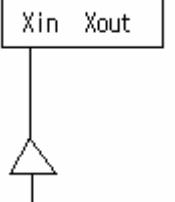
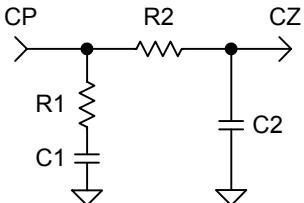
**Figure 30-5. Serial Data Transfer Timing**

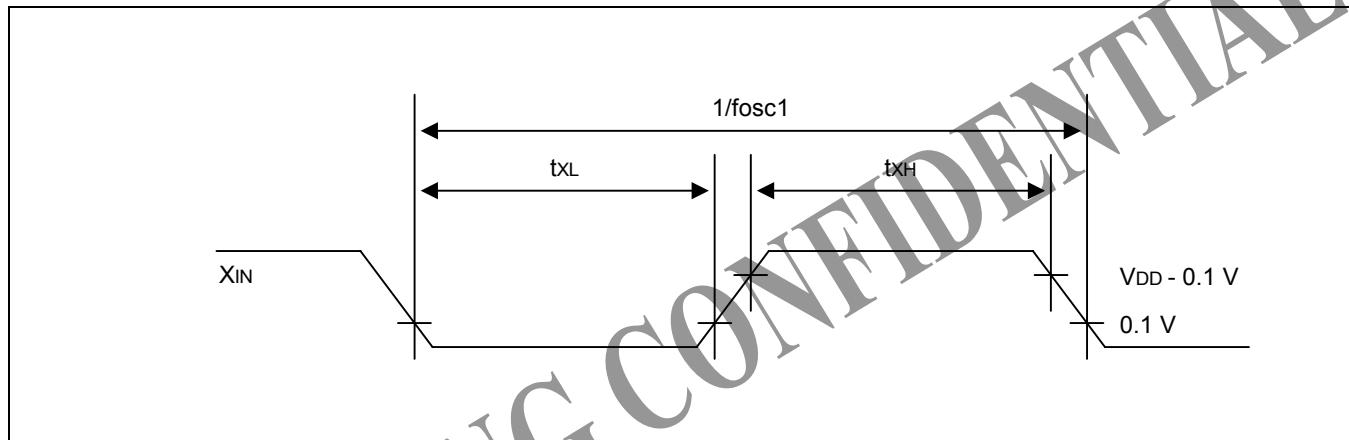
Table 30-11. Main Oscillator Frequency (f_{OSC1})(TA = -20°C to +70°C V_{DD} = 3.0 V to 3.6 V)

Oscillator	Clock Circuit	Test Condition	Min	Typ	Max	Unit
Crystal		Oscillation frequency	32	32.768	35	kHz
		Stabilization time	-	1	3	s
External clock		X _{IN} input frequency	32	-	35	kHz
		X _{IN} input high and low level width (t _{XH} , t _{XL})	14	-	16	us

NOTE: Oscillation stabilization time (t_{ST1}) is the time that the amplitude of an oscillator input rises to 0.8 V_{DD}, after a power-on occurs, or when Stop mode is ended by a nRESET or a interrupt signal.

Table 30-12. PLL Clock Synthesizer Frequency (f_{OUT})(TA = -20°C to +70°C V_{DD} = 3.0 V to 3.6 V)

Oscillator	External Loop Filter Circuit	Min	Typ	Max	Unit
PLL f _{OUT0} clock		-	-	200	MHz
PLL f _{OUT1} clock		-	-	200	

Figure 30-6. Clock Timing Measurement at X_{IN}

31 MECHANICAL DATA

OVERVIEW

The S5L8700X microcontroller is currently available in a 232-pin FBGA package.

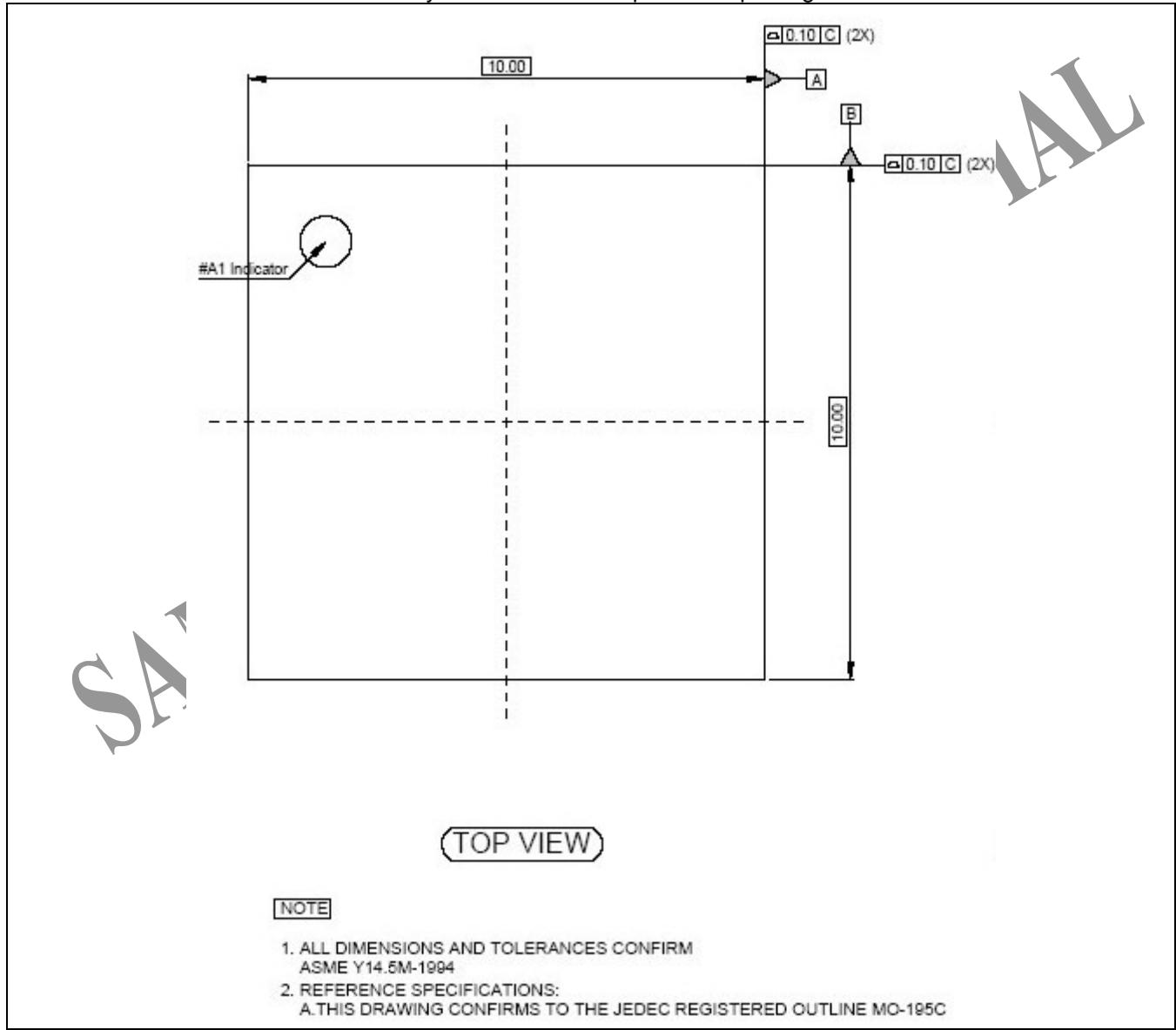


Figure 31-1. 232-FBGA-1010 Package Dimensions (Top View)

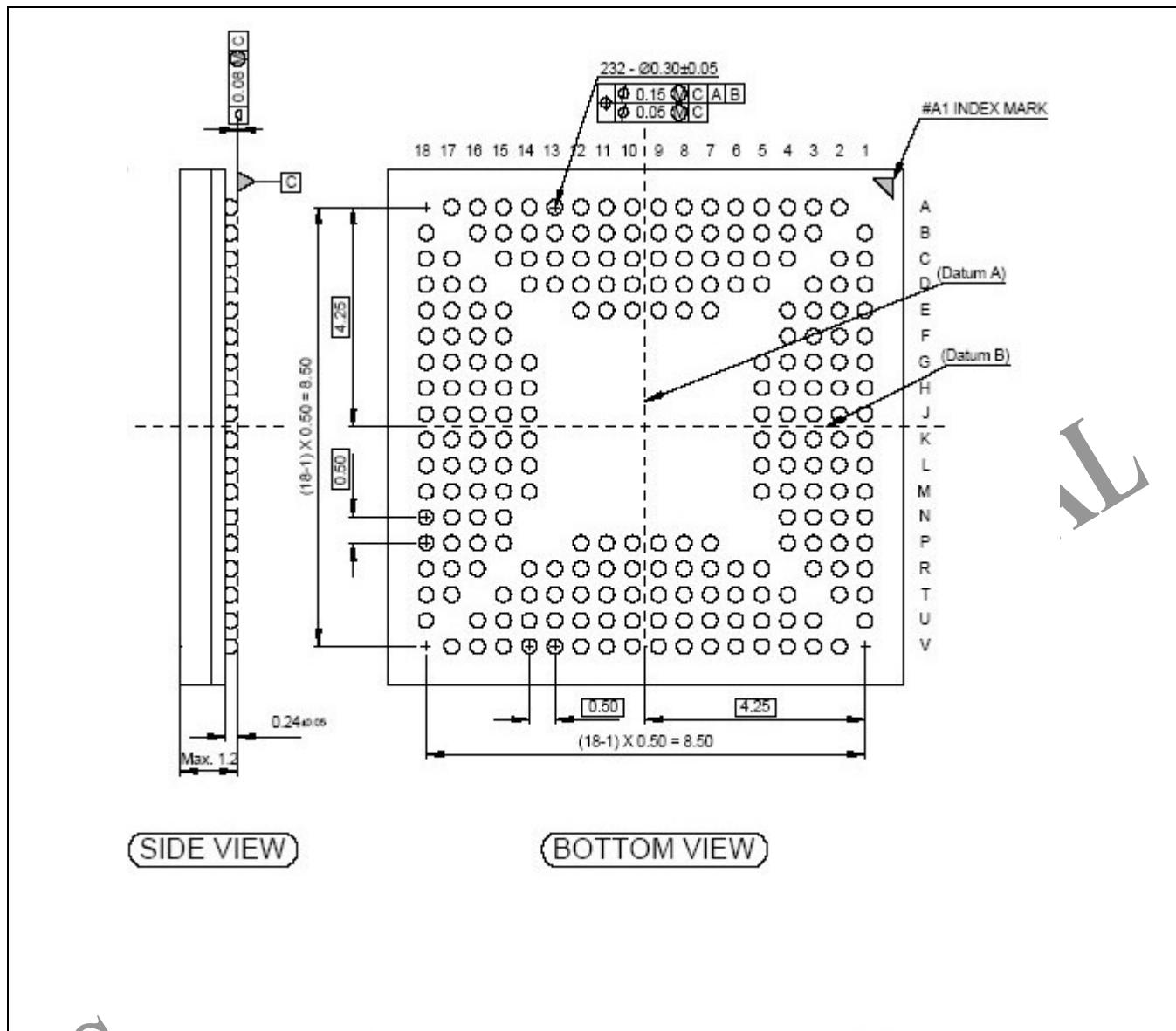


Figure 31-2. 232-FBGA-1010 Package Dimensions (Bottom View)