SAMSUNG ELECTRONICS

CPAD-WALTZ(S5L840F)

Internet Audio Decoder for Flash Memory Media

Data Sheet

INTRODUCTION

S5L840F is a single chip digital audio player IC supporting various compressed audio format on Flash Memory Media. S5L840F provides 2Mbits of embedded NOR flash memory and 76Kbytes of SRAM requiring no external memory. A 16bit RISC processor (CALMRISC16^{TM}) and 24bit MAC(MAC2424 ^{TM}) are provided as a CPU and DSP function.

FEATURES

- Supply voltage range:
  - Supply Voltage (Core) : 1.8V
  - Supply Voltage (IO) : 3.0V
- X-tal Oscillator: 32.768 KHz
- 16bit RISC(CalmRISC16) & 24bit MAC
  with 4KB of Instruction Cache
  6KB of X Cache
  6KB of Y Cache
- 2Mbit NOR Flash & 76KB SRAM
- IO DMA
- Supports SMC/MMC/SD/Memory Stic
- LCD Controller Interface
- 2 Channels of IIS
- IIC / SPDIF Output / UART / SPI
- USB1.1
- 5 channel 10bit ADC
- RTC
- GPIO

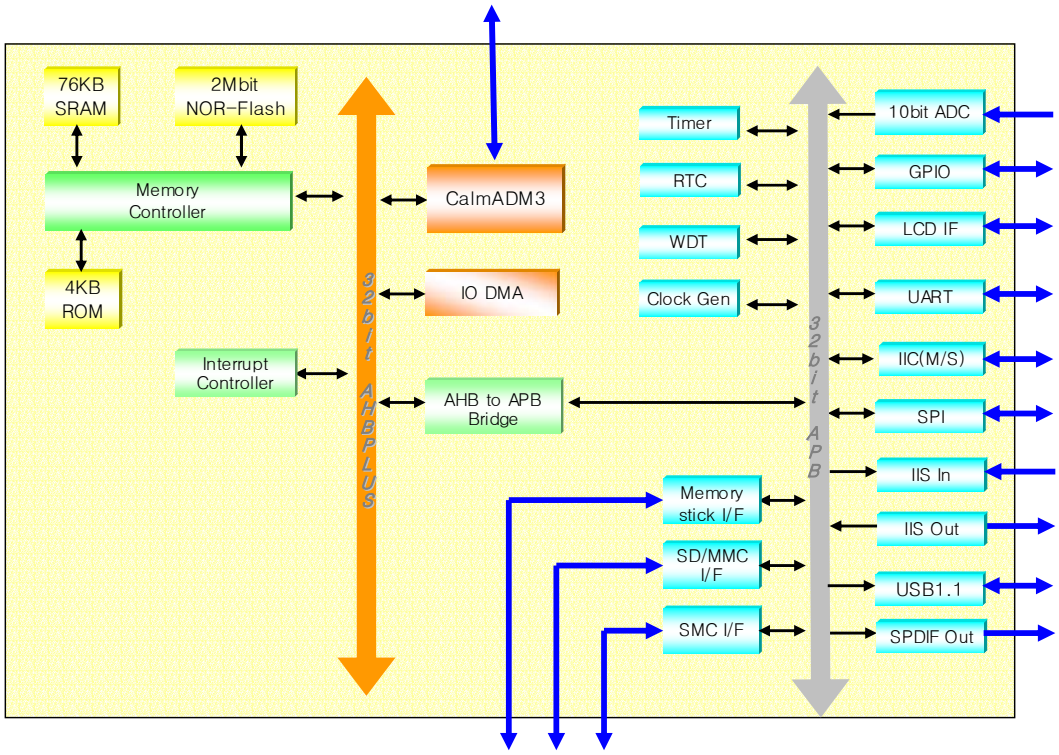
TYPICAL APPLICATION

- MP3/WMA/etc Player

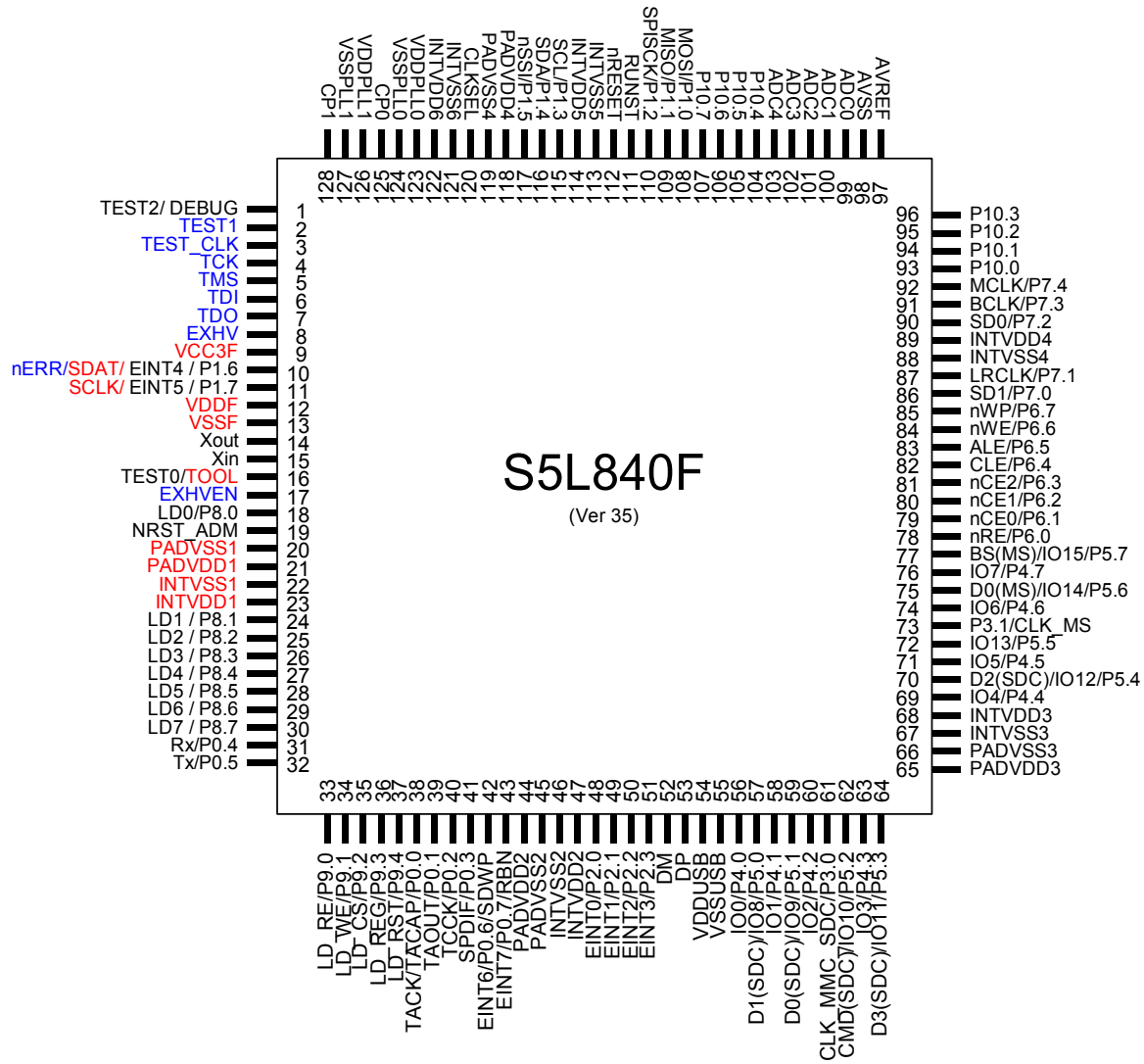
ORDERING INFORMATION

Device	Package	Operating Temperature
S5L840F	128-TQFP-1414	–40 °C – +85 °C

BLOCK DIAGRAM



PIN CONFIGURATION



PIN DESCRIPTION

Pin Number	Pin Assignment	I/O	Pin Description
1	TEST2/DEBUG	I	DEBUG(internal F/F value dump) control
2	TEST1	I	Test mode
3	TEST_CLK(for debug)	I	Test clock
4	TCK	I	JTAG clock. Pull-Up.
5	TMS	I	JTAG mode selection. Pull-Up.
6	TDI	I	JTAG input
7	TD0	O	JTAG output
8	EXHV	B	Flash high voltage test
9	VCC3F	P	Flash memory internal 3.3V Power.
10	nERR/EINT4/P1.6	B	UART, GPIO
11	EINT5/P1.7	B	UART, GPIO
12	VDDF	P	Flash memory internal 1.8V Power.
13	VSSF	P	GND for flash core
14	Xout	O	Crystal oscillator signal (~100KHz)
15	Xin	I	Crystal oscillator signal (~100KHz)
16	TEST0/TOOL_MODE	I	BUS/Serial Controller Selection. Pull-Down
17	EXHVEN	I	Flash high volatge test enable. Pull-Down
18	LD0/P8.0	B	LCD I/F
19	NTRST_ADM	I	Adm reset. Pull-Up.
20	PADVSS1	P	Pad power GND
21	PADVDD1	P	Pad power VDD 3.3V
22	INTVSS1	P	Internal logic GND
23	INTVDD1	P	Internal logic Power VDD 1.8V
24	LD1/P8.1	B	LCD I/F

25	LD2/P8.2	B	LCD I/F
26	LD3/P8.3	B	LCD I/F
27	LD4/P8.4	B	LCD I/F
28	LD5/P8.5	B	LCD I/F
29	LD6/P8.6	B	LCD I/F
30	LD7/P8.7	B	LCD I/F
31	Rx/P0.4	B	INT, GPIO
32	Tx/P0.5	B	INT, GPIO
33	LD_RE/P9.0	B	LCD I/F
34	LD_WE/P9.1	B	LCD I/F
35	LD_CS/P9.2	B	LCD I/F
36	LD_REG/P9.3	B	LCD I/F
37	LD_RST/P9.4	B	LCD I/F
38	TACK/TACAP/P0.0	B	Timer A, GPIO
39	TAOUT/P0.1	B	Timer A, GPIO
40	TCCK/P0.2	B	Timer C, GPIO
41	SPDIF/P0.3	B	SPDIF, GPIO
42	EINT6/P0.6/SDWP	B	INT, GPIO, SDC_WP
43	EINT7/P0.7/RBN	B	INT, GPIO, RBN(SMC)
44	PADVDD2	P	Pad power VDD 3.3V
45	PADVSS2	P	Pad power GND
46	INTVSS2	P	Internal logic GND
47	INTVDD2	P	Internal logic Power VDD 1.8V
48	EINT0/P2.0	B	INT, GPIO
49	EINT1/P2.1	B	INT, GPIO
50	EINT2/P2.2	B	INT, GPIO

51	EINT3/P2.3	B	INT, GPIO
52	DM	B	USB transceive/receive port
53	DP	B	USB transceive/receive port
54	VDDUSB	P	USB Power 3.3V
55	VSSUSB	P	USB Ground
56	IO0/P4.0	B	IO0 for SMC /Debug scan in
57	D1(SDC)/IO8/P5.0	B	IO8 for SMC, D0 for SDC
58	IO1/P4.1	B	IO1 for SMC
59	D0(SDC)/IO9/P5.1	B	IO9 for SMC, D1 for SDC
60	IO2/P4.2	B	IO2 for SMC
61	CLK_MMC_SDC/P3.0	B	CLK for MMC/SDC
62	CMD(SDC)/IO10/P5.2	B	IO10 for SMC, CMD/RESP for SDC
63	IO3/P4.3	B	IO3 for SMC
64	D3(SDC)/IO11/P5.3	B	IO11 for SMC, D3 for SDC
65	PADVDD3	P	Pad power VDD 3.3V
66	PADVSS3	P	Pad power GND
67	INTVSS3	P	Internal logic GND
68	INTVDD3	P	Internal logic Power VDD 1.8V
69	IO4/P4.4	B	IO4 for SMC
70	D2(SDC)/IO12/P5.4	B	IO12 for SMC, D2 for SDC
71	IO5/P4.5	B	IO5 for SMC
72	IO13/P5.5	B	IO13 for SMC
73	P3.1/CLK_MS	B	GPIO, CLK for MS
74	IO6/P4.6	B	IO6 for SMC
75	D0(MS)/IO14/P5.6	B	IO14 for SMC, D0 for MS
76	IO7/P4.7	B	IO7 for SMC

77	BS(MS)/IO15/P5.7	B	IO15 for SMC, BS for MS
78	nRE/P6.0	B	SMC control
79	nCE0/P6.1	B	SMC control
80	nCE1/P6.2	B	SMC control
81	nCE2/P6.3	B	SMC control
82	CLE/P6.4	B	SMC control
83	ALE/P6.5	B	SMC control
84	nWE/P6.6	B	SMC control /Debug scan out
85	nWP/P6.7	B	SMC control
86	SD1/P7.0	B	Serial Data In for IIS
87	LRCLK/P7.1	B	Left-Right Clock for IIS
88	INTVSS4	P	Internal logic GND
89	INTVDD4	P	Internal logic Power VDD 1.8V
90	SD0/P7.2	B	Serial Data Out for IIS
91	BCLK/P7.3	B	Bit Clock for IIS
92	MCLK/P7.4	B	Over-sampling Clock for IIS
93	P10.0	B	GPIO
94	P10.1	B	GPIO
95	P10.2	B	GPIO
96	P10.3	B	GPIO
97	AVREF	P	ADC VREF,AVDD33A1,AVDD33A2 연결. 3.3V Power
98	AVSS	P	ADC analog GND. avss33a1,avbb33a1,avss33a2
99	ADC0	I	ADC
100	ADC1	I	ADC
101	ADC2	I	ADC
102	ADC3	I	ADC

103	ADC4	I	ADC
104	P10.4	B	GPIO
105	P10.5	B	GPIO
106	P10.6	B	GPIO
107	P10.7	B	GPIO
108	MOSI/P1.0/tclk0	B	SPI, GPIO, TCLK0(not open)
109	MISO/P1.1/tclk1	B	SPI, GPIO, TCLK1(not open)
110	SPISCK/P1.2	B	SPI, GPIO
111	RUNST	B	JTAG RUNID / TEST mode select
112	nRESET	I	System RESET. Pull-Up.
113	INTVSS5	P	Internal logic GND
114	INTVDD5	P	Internal logic Power VDD 1.8V
115	SCL/P1.3	B	IIC, GPIO
116	SDA/P1.4	B	IIC, GPIO
117	nSSI/P1.5	B	SPI, IIC, UART
118	PADVDD4	P	Pad power VDD 3.3V
119	PADVSS4	P	Pad power GND
120	CLKSEL	I	Clock selection signal. Pull-Up.
121	INTVSS6	P	Internal logic GND
122	INTVDD6	P	Internal logic Power VDD 1.8V
123	VDDPLL0	P	PLL Power supply VDD 1.8V
124	VSSPLL0	P	PLL GND
125	CP0	O	Low pass filter circuit for pll0
126	VDDPLL1	P	PLL Power supply VDD 1.8V
127	VSSPLL1	P	PLL GND
128	CP1	O	Low pass filter circuit for pll1

ABSOLUTE MAXIMUM RATINGS

Characteristic	Symbol	Value	Unit
Supply Voltage	VDD	3.8	V
Input Voltage	VIN	6.5	V
Storage Temperature Range	Tstg	-65–150	°C

ELECTRICAL CHARACTERISTICS

Recommended Operating Conditions

Characteristic	Symbol	Value	Unit
Supply Voltage	VDD	1.65~1.95(Core), 2.7~3.3(IO)	V
Operating Temperature Range	Topr	-40–85	°C

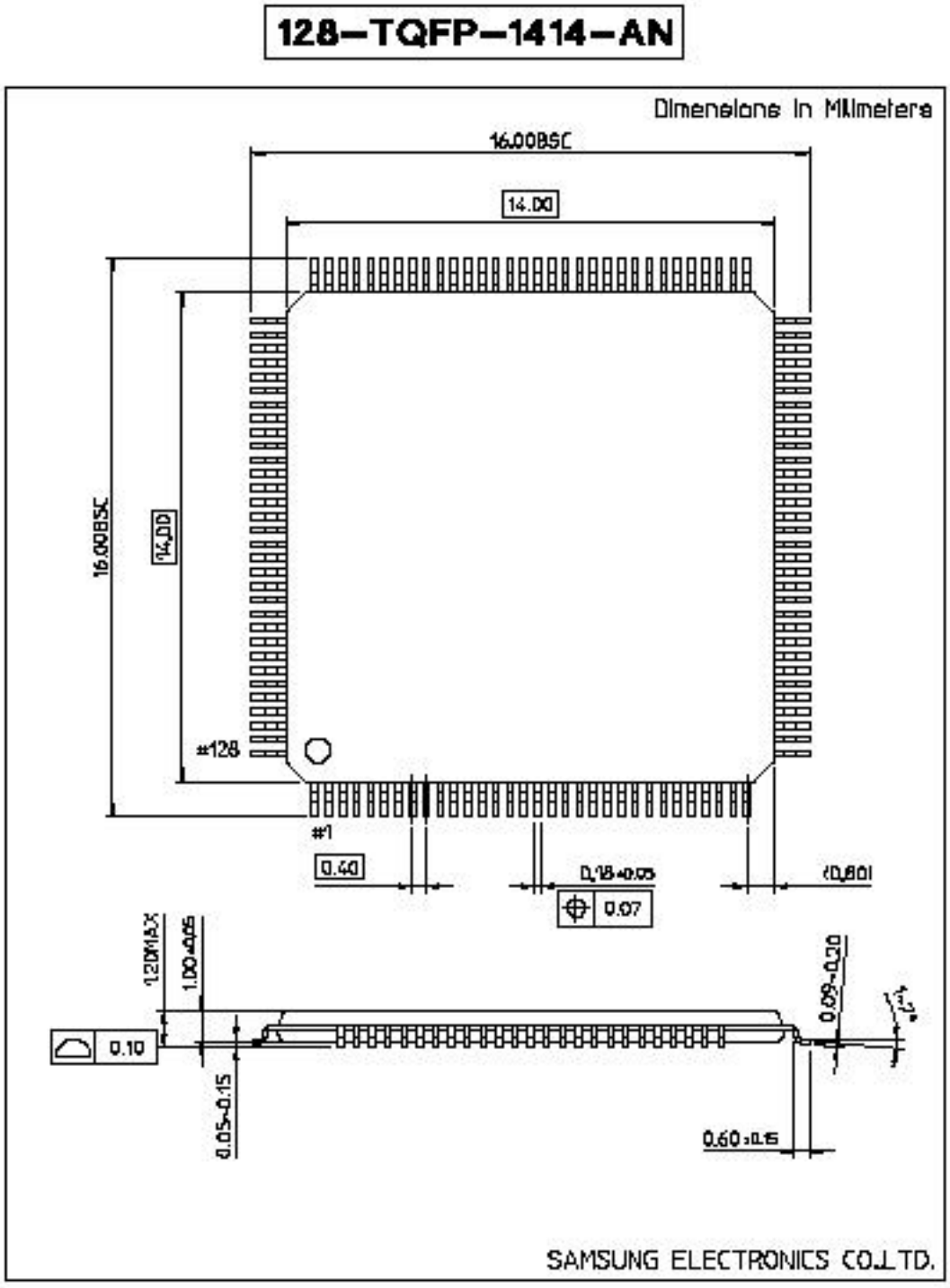
DC Characteristics

(Ta = 25°C, VDD(IO) = 3.3V, Unless otherwise specified)

Symbol	Characteristic	Test Conditions	Min	Typ	Max	Unit
VIH	High level input voltage	–	2.0	–	–	V
VIL	Low level input voltage	–	–	–	0.8	V
VT	Switching threshold			1.4		V
VT+	Schmitt trigger, positive –going threshold	CMOS			2.0	V
VT-	Schmitt trigger, negative-going threshold	CMOS	0.8			V
VOH	High level output voltage	IOH = -2mA	2.4	–	–	V
VOL	Low level output voltage	IOL = 2mA	–	–	0.4	V
IOZ	Tri-state output leakage current	Vout = Vss or GND	-10	–	10	µA

NOTES:

PACKAGE DIMENSIONS



Low-Power & High-Performance RISC Core

CalmRISC16

Technical Reference Manual

MCU Team
LSI Division
System LSI Business
Samsung Electronics Co.

1. Introduction

1.1 Feature

The main features of CalmRISC16, a 16-bit embedded RISC MCU core, are high performance, low power consumption, and efficient coprocessor interface. It can operate up to 100MHz, and consumes 100μA/MHz @3.3V. When operating with MAC2424, a 24-bit fixed point DSP coprocessor, CalmRISC16 can operate up to 80MHz. Through efficient coprocessor interface, CalmRISC16 provides a powerful and flexible MCU+DSP solution. The following gives brief summary of main features of CalmRISC16.

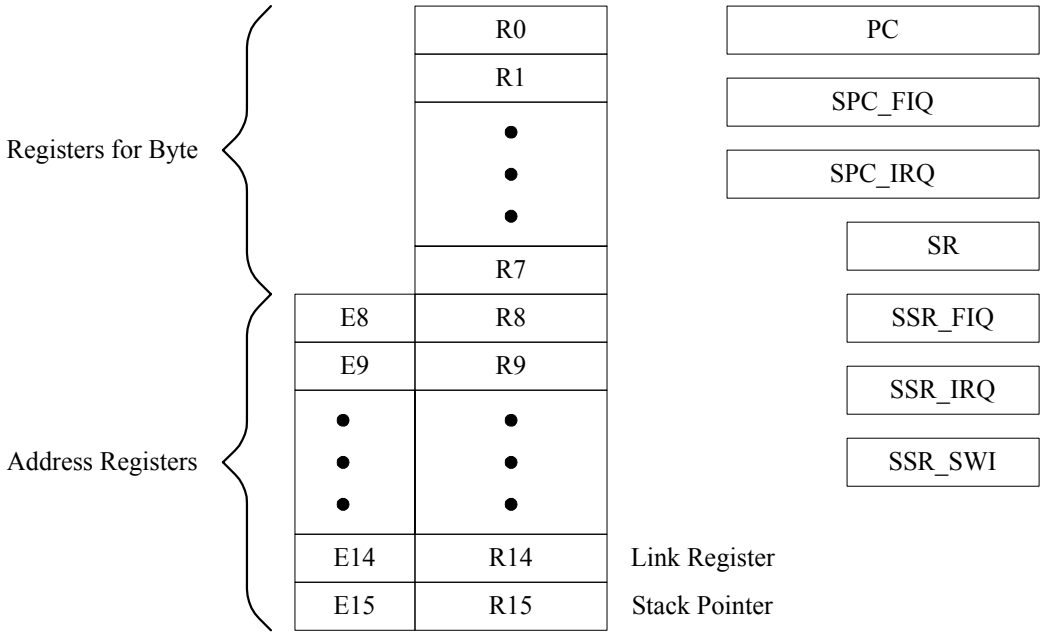
- H/W Feature
  - Power consumption : 100μA per MHz @3.3V, 0.35μ process
  - Maximum frequency : 100MHz @3.3V
  - 0.78 mm^2 die size
- Architecture
  - Harvard RISC architecture
  - 5-Stage pipeline
- Registers
  - Sixteen 16-bit general registers
  - Eight 6-bit extension registers
  - 22-bit Program Counter (PC)
  - 16-bit Status Register (SR)
  - Seven saved registers for interrupts.
- Instruction Set
  - 16-bit instruction width for 1-word instructions
  - 32-bit instruction width for 2-word instructions
  - Load/Store instruction architecture
  - Delayed branch support
  - C-language/OS support
  - Bit operation for I/O process
- Instruction Execution Time
  - One instruction/cycle for basic instructions
- Address Space
  - 4M byte for Program Memory
  - 4M byte for Data Memory

1.2. Registers

In CalmRISC16 there are sixteen 16-bit general registers, eight 6-bit extension registers, a 16-bit Status Register(SR), a program counter (PC), and seven saved registers.

General Registers & Extension Registers

The following figure shows the structure of the general registers and the extension registers.



Register Structure in CalmRISC16

The general registers (from R0 to R15) can be either a source register or a destination register for almost all ALU operations, and can be used as an index register for memory load/store instructions (e.g., LDW R3, @[A8+R2]). The 6-bit extension registers (from E8 to E15) are used to form a 22-bit address register (from A8 to A15) by concatenating with a general register (from R8 to R15). The address registers are used to generate 22-bit program and data addresses.

Special Registers

The special registers consist of 16-bit SR (Status Register), 22-bit PC (Program Counter), and saved registers for IRQ(interrupt), FIQ(fast interrupt), and SWI(software interrupt). When IRQ interrupt occurs, the most significant 6 bits of the return address are saved in SPCH_IRQ, the least significant 16 bits of

the return address are saved in SPCL_IRQ, and the status register is saved in SSR_IRQ. When FIQ interrupt occurs, the most significant 6 bits of the return address are saved in SPCH_FIQ, the least significant 16 bits of the return address are saved in SPCL_FIQ, and the status register is saved in SSR_FIQ. When a SWI instruction is executed, the return address is saved in A14 register (E14 concatenated with R14), and the status register is saved in SSR_SWI. The least significant bit of PC, SPCL_IRQ and SPCL_FIQ is read only and its value is always 0.

The 16-bit register SR has the following format.

15							8	7							0
T	-	-	-	-	-	-	-	-	PM	Z1	Z0	V	TE	IE	FE

- FE : FIQ enable bit, FIQ is enabled when FE is set.
- IE : IRQ enable bit, IRQ is enabled when IE is set.
- TE : TRQ enable bit, Trace is enabled when TE is set.
- V : overflow flag, set/clear accordingly when arithmetic instructions are executed.
- Z0 : zero flag of R6, set when R6 equals zero and used as the branch condition when BNZD instruction with R6 is executed.
- Z1 : zero flag of R7, set when R7 equals zero and used as the branch condition when BNZD instruction with R7 is executed.
- PM : privilege mode bit. PM = 1 for privilege mode and PM = 0 for user mode
- T : true flag, set/clear as a result of an ALU operation.

FE, IE, TE, and PM bits can be modified only when PM = 1 (privilege mode). The only way to change from user mode to privilege mode is via interrupts including SWI instructions. The reserved bit of SR (from bit 7 to bit 14) can be used for other purposes without any notice. Hence programmers should not depend on the value of the reserved bits in their programming. The reserved bits are read as 0 value.

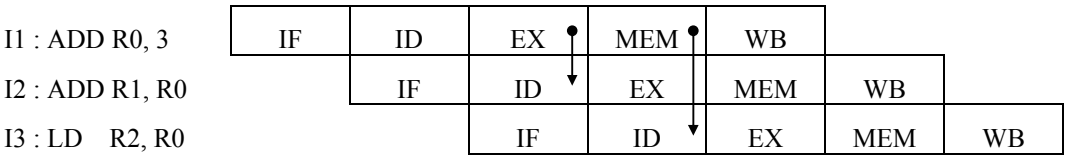
1.3. Pipeline Structure

CalmRISC16 has a 5-stage pipeline architecture. It takes 5 cycles for an instruction to do its operation. In a pipeline architecture, instructions are executed overlapped, hence the throughput is one instruction per cycle. Due to data dependency, control dependency, and 2 word instructions, the throughput is about 1.2 on the average. The following diagram depicts the 5-stage pipeline structure.

IF	ID	EX	MEM	WB

In the first stage, which is called IF (Instruction Fetch) stage, an instruction is fetched from program

memory. In the second stage, which is called ID (Instruction Decoding) stage, the fetched instruction is decoded, and the appropriate operands, if any, for ALU operation are prepared. In the case of branch or jump instructions, the target address is calculated in ID stage. In the third stage, which is called EX (Execution) stage, ALU operation and data address calculation are executed. In the fourth stage, which is called MEM (Memory) stage, data transfer from/to data memory or program memory is executed. In the fifth stage, which is called WB (Write Back) stage, a write-back to register file can be executed. The following figure shows an example of pipeline progress when 3 consecutive instructions are executed.



In the above example, the instruction I2 needs the result of the instruction I1 before I1 completes. To resolve this problem, the EX stage result of I1 is forwarded to ID stage of I2. Similar forwarding mechanism occurs from MEM stage of I1 to ID stage of I3.

The pipeline cannot progress (called a pipeline stall) due to a data dependency, a control dependency, or a resource conflict.

When a source operand of an ALU instruction is from a register, which is loaded from memory in the previous instruction, 1 cycle of pipeline stall occurs (called load stall). Such load stalls can be avoided by smart reordering of the instruction sequences. CalmRISC16 has 2 classes of branch instructions, those with a delay slot and without a delay slot. Non-delay slot branch instructions incurs a 1 cycle pipeline stall if the branch is taken, due to a control dependency. For branch instructions with a delay slot, no cycle waste is incurred if the delay slot is filled with a useful instruction (or non NOP instruction). Pipeline stalls due to resource conflicts occurs when two different instructions access at the same cycle the same resource such as the data memory and the program memory. LDC (data load from program memory) instruction causes a resource conflict on the program memory. Bit operations such as BITR and BITS (read-modify-write instructions) cause a resource conflict on the data memory.

1.4 Interrupts

In CalmRISC16, there are five interrupts: RESET, FIQ, IRQ, TRQ, SWI. The RESET, FIQ, and IRQ interrupts correspond to external requests. TRQ and SWI interrupts are initiated by an instruction (therefore, in a deterministic way). The following table shows a summary of interrupts.

Name	Priority	Address	Description
RESET	1	000000h	Hardware Reset
FIQ	3	000002h	Fast Interrupt Request

IRQ	5	000004h	Interrupt Request
TRQ	2	000006h	Trace Request
SWI	4	000008h ~ 0000feh	Software Interrupt

When nRES (an input pin CalmRISC16 core) signal is released (transition from 0 to 1), "JMP addr:22" is automatically executed by CalmRISC16. Among the 22-bit address addr:22, the most significant 6 bits are forced to 0, and the least significant 16 bits are the contents of 000000h (i.e., reset vector address) of the program memory. In other words, "JMP {6'h00[1], PM[000000h]}" instruction is forced to the pipeline. The initial value of PM bit is 1 (that is, in privilege mode) and the initial values of other bits in SR register are 0. All other registers are not initialized (i.e., unknown).

When nFIQ (an input pin CalmRISC16 core) signal is active (transition from 1 to 0), "JMP addr:22" instruction is automatically executed by CalmRISC16. The address of FIQ interrupt service routine is in 000002h (i.e., FIQ vector address) of the program memory (i.e., "JMP {6'h00, PM[000002h]}"). The return address is saved in {SPCH_FIQ, SPCL_FIQ} register pair, and the SR value is saved in SSR_FIQ register. PM bit is set. FE, IE, and TE bits are cleared. When RET_FIQ instruction is executed, SR value is restored from SSR_FIQ, and the return address is restored into PC from {SPCH_FIQ, SPCL_FIQ}.

When nIRQ signal (an input pin CalmRISC16 core) is active (transition from 1 to 0), "JMP {6'h00, PM[000004h]}" instruction is forced to the instruction pipeline. The return address is saved in {SPCH_IRQ, SPCL_IRQ} register pair, and the SR value is saved in SSR_IRQ register. PM bit is set. IE and TE bits are cleared. When RET_IRQ instruction is executed, SR value is restored from SSR_IRQ, and return address is restored to PC from {SPCH_IRQ, SPCL_IRQ}.

When TE bit is set, TRQ interrupt happens and "JMP {6'h00, PM[000006h]}" instruction is executed right after each instruction is executed. TRQ interrupt uses the saved registers of IRQ(that is, {SPCH_IRQ, SPCL_IRQ} register pair and SSR_IRQ) to save the return address and SR, respectively. PM bit is set. IE, TE bits are cleared.

When "SWI imm:6[2]" instruction is executed, the return address is saved in the register A14, and the value of SR is saved in SSR_SWI. Then the program sequence jumps to the address (imm:6 * 4). PM bit is set. IE and TE bits are cleared. "SWI 0" and "SWI 1" are prohibited because the addresses are reserved for other interrupts. When RET_SWI instruction is executed, SR is restored from SSR_SWI, and the return address is restored to PC from A14.

1.5 Memory Formats

[1] 6'h00 is defined as 00 (or zero) in 6 bits

[2] imm:6 is defined as 6-bit immediate number

CalmRISC16 adopts a big endian memory format. In a big endian memory format, the most significant byte of word data is stored at an even address, and the least significant byte is stored at an odd address. For example let us assume that the word data "1234h" is stored at the address 100h. Then the higher byte "12h" is stored at the address 100h, and the lower byte "34h" is stored at the address 101h. When the 22-bit data "123456h" is stored at the address 100h by "LDW @An, Ai" instruction, "00h" is at the address 100h, "12h" is at the address 101h, "34h" is at the address 102h, and "56h" is at the address 103h.

1.6 Signal Description

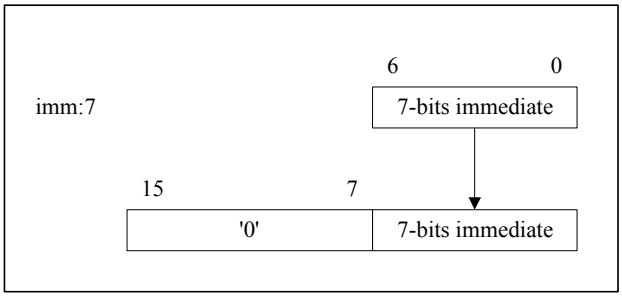
Name	Direction	Description
PA[20:0]	O	Program Memory Address, equivalent to PC[21:1]
PD[15:0]	I	Program Data
nPMCS	O	Program Memory Chip Selection
nLDC	O	Data load from program memory indicator
DA[21:0]	O	Data Memory Address DA[4:0] is shared with SYS and CLD instructions
DI[15:0]	I	Input from Data Memory, Input from coprocessor for CLD instruction.
DO[15:0]	O	Output to Data Memory, Output to coprocessor for CLD instruction.
nDMCSH	O	Chip Selection for Higher Byte Data Memory
nDMCSL	O	Chip Selection for Lower Byte Data Memory
DMWR	O	Data Memory Write, 1 means transfer from Core to Memory
nDME	O	Data Bus Enable Signal.
nRES	I	Hardware Reset
nFIQ	I	Fast Interrupt Request
nIRQ	I	Interrupt Request
nEXPACK	O	Exception Acknowledge
nWAIT	I	Wait signal, core is stopped when active.
nSYSID	O	SYS instruction indicator
MCLK	I	Main Clock Input
ECLK	O	Early Clock Output
ICLK	O	Clock Output
nCOPID	O	Coprocessor instruction indicator
nCLDID	O	Coprocessor Load instruction indicator
CLDWR	O	Write to Coprocessor indicator

COPIR[12:0]	O	Instruction to coprocessor, 13-bit immediate field in COP instruction.
EC[3:0]	I	External Conditions from coprocessor or peripherals.
nBRK	O	Software break indicator
nBKACK	O	Break Acknowledge
BKMODE[2:0]	O	Break Mode, indicates core state when core breaks.
BKREQ	I	Break Request
nGIDIS	I	Global interrupt disable, when active, all interrupt is disabled.
PDGRANT	I	Indicates program memory access is permitted.
PDWAIT	I	Indicates current program memory access is not complete.
DBGRANT	I	Indicates data memory access is permitted.
DBWAIT	I	Indicates current data memory access is not complete.
DBREQ	O	Signal asking for data bus permission.
PMODE	O	Privilege Mode Indicator
CGRANT	O	Indicates that coprocessor may use data bus.
CSTALL	I	Coprocessor indicates that coprocessor pipeline stall occurs.
CMW	I	Coprocessor indicates that coprocessor instruction is multiple word.
nSEQ	O	Indicates that the next program address is sequential.
nINCPC	I	If it is 1, PC value is not incremented when sequential execution.
CCLK	O	Clock output to coprocessor

2. Instructions

2.1. ALU instructions

In operations between a 16-bit general register and an immediate value, the immediate value is zero-extended to 16-bit. The following figure shows an example of 7-bit immediate numbers.



In operations between a 22-bit register and an immediate value, the immediate value is zero-extended to

22-bit. In operations between a 22-bit register and a 16-bit register, the 16-bit register is zero-extended to 22-bit. The overflow flag in a 16-bit arithmetic operation is saved to V flag in SR register. ALU instructions are classified into 3 classes as follows.

- ALUop Register, Immediate
- ALUop Register, Register
- ALUop Register

ALUop Register, Immediate

ADD/ADC/SUB/SBC/AND/OR/XOR/TST/CMP/CMPU Rn, #imm:16

The instructions perform an ALU operation of which source operands are a 16-bit general register Rn and a 16-bit immediate value. In the instructions TST/CMP/CMPU, only T flag is updated accordingly as the result. In the instructions ADD/ADC/SUB/SBC, the value of T flag is the carry flag of the operations, and the value of V flag indicates whether overflow or underflow occurs. In the instructions AND/OR/XOR/TST, the value of T flag indicates whether the result is zero (T=1). "CMP {GT|GE|EQ}, Rn, #imm:16[3]" instructions are for signed comparison operations (GT for greater than, GE for greater than or equal to and EQ for equal to), and "CMPU {GT|GE}, Rn, #imm:16" instructions are for unsigned comparison operations.

ADD/SUB An, #imm:16

The immediate value is zero-extended to 22-bit value. No flag update occurs.

ADD/SUB Rn, #imm:7

The immediate value is zero-extended to 16-bit value. T flag is updated to the carry of the operation. V flag is updated.

AND/OR/XOR/TST R0, #imm:8

The immediate value is zero-extended to 16-bit value. T flag indicates whether the lower 8-bit of the logical operation result is zero.

CMP EQ, Rn, #imm:8

The immediate value is zero-extended to 16-bit value. Rn is restricted to R0 to R7. T flag is updated as the result of the instruction.

[3] imm:16 is defined as a 16-bit immediate number

CMP GE, Rn, #imm:6

The immediate value is zero-extended to 16-bit value. The instruction is for signed compare. T flag is updated as the result of the instruction.

ADD/SUB An, #imm:5

The immediate value is zero-extended to 22-bit value. No flag is updated.

ALUop Register, Register

ADD/SUB/ADC/SBC/AND/OR/XOR/TST/CMP/CMPU Rn, Ri

The instructions perform an ALU operation of which source operands are a pair of 16-bit general registers. In the instructions TST/CMP/CMPU, only T flag is updated as the result. In the instructions ADD/ADC/SUB/SBC, the value of T flag is the carry of the operations, and the value of V flag indicates whether overflow or underflow occurs. In the instructions AND/OR/XOR/TST, the value of T flag indicates whether the result is zero. "CMP {GT|GE|EQ}, Rn, Ri" instructions are for signed comparison, and "CMPU {GT|GE}, Rn, Ri" instructions are for unsigned comparison.

ADD/SUB An, Ri

16-bit general register Ri is zero-extended to 22-bit value. The result is saved in the 22-bit register An. No flag update occurs.

CMP EQ, An, Ai

The instruction compares two 22-bit registers.

MUL {SS|SU|US|UU}, Rn, Ri

The general registers Rn and Ri can be one of R0 to R7. The instruction multiplies the lower byte of Rn and the lower byte of Ri, and the 16-bit result is saved in Rn. The optional field, SS, SU, US, and UU, indicates whether the source operands are signed value or unsigned value. The first letter of the two letter qualifiers corresponds to Rn, and the second corresponds to Ri. For example, in the instruction "MUL SU, R0, R1", the 8-bit signed value in the lower byte of R0 and the 8-bit unsigned value in the lower byte of R1 are multiplied, and the 16-bit result is saved in R0.

RR/RL/RRC/SR/SRA/SLB/SRB/DT/INCC/DECC/COM/COM2/COMC/EXT Rn

For "DT Rn"(Decrement and Test) and "COM Rn"(Complement) instructions, T flag indicates whether the result is zero. In the instruction of "EXT Rn"(Sign Extend), no flag update occurs. In all other instructions, carry-out of the operation is transferred to T flag. In the instruction of DT, INCC, and DECC,

V flag indicates whether overflow or underflow occurs.

2.2. Load instructions

"Load instructions" move data from register/memory/immediate to register/memory. When the destination is a memory location, only general registers and extension registers can be the source. We can classify "Load instructions" into the following 4 classes.

- LD Register, Register
- LD Register, Immediate
- LD Data Memory, Register / LD Register, Data Memory
- LD Register, Program Memory

LD Register, Register

LD Rn, Ri / LD An, Ai

The instructions move 16-bit or 22-bit data from the source register to the destination register. When the destination register is R6/R7, the zero flag Z0/Z1 is updated. In all other cases, no flag update occurs.

LD Rn, Ei / LD En, Ri

In the instruction "LD Rn, Ei", the 6-bit data in Ei is zero-extended to 16-bit data, and then transferred to Rn. When the destination register is R6/R7, the zero flag Z0/Z1 is updated. In the instruction "LD En, Ri", least significant 6 bits of Ri are transferred to En. Rn/Ri is one of the registers from R0 to R7.

LD R0, SPR / LD SPR, R0

SPR : SR, SPCL_FIQ, SPCH_FIQ, SSR_FIQ, SPCL_IRQ, SPCH_IRQ, SSR_IRQ, SSR_SWI

The instructions transfer data between SPR (Special Purpose Registers) and R0. No flag update occurs except the case that the destination register is SR.

LD An, PC

The instruction moves the value of (PC+4) to An.

LD Register, Data Memory / LD Data Memory, Register

LDW Rn, @[SP+edisp:9] / LDW @[SP+edisp:9], Rn

The instructions transfer 16-bit data between a general register Rn and the memory location at the address of (SP+edisp:9). Note SP is another name of A15. edisp:9 is an even positive displacement from 0 to 510.

edisp:9 is encoded into an 8-bit displacement value in the instruction map because the LSB is always 0. When the address is calculated, the 8-bit displacement field is shifted to the left by one bit, and then the result is added to the value of SP. Even if the address might be specified as odd in assembly mnemonic, the LSB of the address should be truncated to zero for word alignment.

LDW Rn, @[Ai+edisp:5] / LDW @[Ai+edisp:5], Rn

The instructions transfer 16-bit data between a general register Rn and the memory location at the address of (Ai+edisp:5). edisp:5 is an even positive displacement from 0 to 30. edisp:5 is encoded into an 4-bit displacement value in the instruction map because the LSB is always 0. When the address is calculated, the 4-bit displacement field is shifted to the left by one bit, and then the result is added to the value of Ai. Even if the address might be specified as odd in assembly mnemonic, the LSB of the address should be truncated to zero for word alignment.

LDW Rn, @[Ai+disp:16] / LDW @[Ai+disp:16], Rn

The instructions transfer 16-bit data between a general register Rn and the memory location at the address of (Ai+disp:16). disp:16 is an positive displacement from 0 to FFFFh. If the address is odd, the LSB of the address is set to zero for word alignment.

LDW Rn, @[Ai+Rj] / LDW @[Ai+Rj], Rn

The instructions transfer 16-bit data between a general register Rn and the memory location at the address of (Ai+Rj). The value of Rj is zero-extended to 22-bit value. If the address is odd, the LSB of the address is set to zero for word alignment.

LDW An, @[Ai+edisp:5] / LDW @[Ai+edisp:5], An

The instructions transfer 22-bit data between an address register An and the memory location at the address of (Ai+edisp:5). edisp:5 is an even positive displacement from 0 to 30. edisp:5 is encoded into an 4-bit displacement value in the instruction map because the LSB is always 0. When the address is calculated, the 4-bit displacement field is shifted to the left by one bit, and then the result is added to the value of Ai. Even if the address might be specified as odd in assembly mnemonic, the LSB of the address should be truncated to zero for word alignment.

LDW An, @[Ai+disp:16] / LDW @[Ai+disp:16], An

The instructions transfer 22-bit data between an address register An and the memory location at the address of (Ai+disp:16). disp:16 is an positive displacement from 0 to FFFFh. If the address is odd, the LSB of the address is set to zero for word alignment.

LDW An, @[Ai+Rj] / LDW @[Ai+Rj], An

The instructions transfer 22-bit data between an address register An and the memory location at the address of (Ai+Rj). The value of Rj is zero-extended to 22-bit value. If the address is odd, the LSB of the address is set to zero for word alignment.

PUSH Rn/PUSH Rn, Rm/PUSH An/ PUSH An, Am

The instruction "PUSH Rn" transfers 16-bit data from the register Rn to the memory location at the address of SP, and then increments the value of SP by 2. The register Rn should not be R15. The operation of "PUSH R15" is undefined. The instruction "PUSH Rn, Rm" pushes Rn and then Rm. The registers Rn and Rm should not be the same. The registers Rn and Rm should not be R15. The instruction "PUSH An" pushes Rn and then En. When the extension register En is pushed, the value of En is zero-extended to 16-bit data. The register An should not be A15. The instruction "PUSH An, Am" pushes An and then Am. The registers An and Am should not be the same

POP Rn/POP Rn, Rm/POP An/ POP An, Am

The instruction "POP Rn" decrements the value of SP by 2, and then transfers 16-bit data to the register Rn from the memory location at the address of SP. The register Rn should not be R15. The operation of "POP R15" is undefined. The instruction "POP Rn, Rm" pops Rn and then Rm. The registers Rn and Rm should not be the same. The registers Rn and Rm should not be R15. The instruction "POP An" pops En and then Rn. When the extension register En is popped, the least significant 6 bits are transferred to En. The register An should not be A15. The instruction "POP An, Am" pops An and then Am. The registers An and Am should not be the same

LDB Rn, @[Ai+disp:4] / LDB @[Ai+disp:4], Rn

The instructions transfer 8-bit data between the general register Rn and the memory location at the address of (Ai+disp:4). disp:4 is a positive displacement from 0 to 15. The general register Rn is one R0 to R7. In the instruction "LDB Rn, @[Ai+disp:4]", the 8-bit data is zero-extended to 16-bit data, and then written into Rn. In the instruction "LDB @[Ai+disp:8], Rn", the least significant byte of Rn is transferred to the memory.

LDB Rn, @[Ai+disp:16] / LDB @[Ai+disp:16], Rn

The instructions transfer 8-bit data between the general register Rn and the memory location at the address of (Ai+disp:16). disp:16 is a positive displacement from 0 to FFFFh. The general register Rn is one of R0 to R7. In the instruction "LDB Rn, @[Ai+disp:16]", the 8-bit data is zero-extended to 16-bit data, and then written into Rn. In the instruction "LDB @[Ai+disp:16], Rn", the least significant byte of Rn is transferred to the memory.

LDB R0, @[A8+disp:8] / LDB @[A8+disp:8], Rn

The instructions transfer 8-bit data between the general register R0 and the memory location at the address of (A8+disp:8). disp:8 is a positive displacement from 0 to 255. In the instruction "LDB R0, @[A8+disp:8]", the 8-bit data is zero-extended to 16-bit data, and then written into R0. In the instruction "LDB @[A8+disp:8], R0", the least significant byte of R0 is transferred to the memory.

LDB Rn, @[Ai+Rj] / LDB @[Ai+Rj], Rn

The instructions transfer 8-bit data between the general register Rn and the memory location at the address of (Ai+Rj). The value of Rj is zero-extended to 22-bit value. The general register Rn is one of the 8 registers from R0 to R7. In the instruction "LDB Rn, @[Ai+Rj]", the 8-bit data is zero-extended to 16-bit data, and then written into R0. In the instruction "LDB @[Ai+Rj], Rn", the least significant byte of Rn is transferred to the memory.

LD Register, Program Memory

LDC Rn, @Ai

The instruction transfers 16-bit data to Rn from program memory at the address of Ai.

LD Register, # immediate

LD Rn, #imm:8 / LD Rn, #imm:16 / LD An, #imm:22

The instructions move an immediate data to a register. In the instruction "LD Rn, #imm:8", the immediate value is zero-extended to 16-bit value.

2.3. Branch instructions

CalmRISC16 has 2 classes of branch instructions: with a delay slot and without a delay slot. If a delay slot is filled with a useful instruction (or an instruction which is not NOP), then the performance degradation due to the control dependency can be minimized. However, if the delay slot cannot be used, then it should be NOP instruction, which can increase the program code size. In this case, the corresponding branch instruction without a delay slot can be used to avoid using NOP.

Some instructions are not permitted to be in the delay slot. The prohibited instructions are as follows.

- All 2-word instructions
- All branch and jump instructions including SWI, RETD, RET_SWI, RET_IRQ, RET
- BREAK instructions

When a prohibited instruction is in the delay slot, the operation of CalmRISC16 is undefined or unpredictable.

BSRD eoffset:13

In the instruction, called branch subroutine with a delay slot, the value (PC + 4) is saved into A14 register, the instruction in the delay slot is executed, and then the program sequence is moved to (PC + 2 + eoffset:13), where PC is the address of the instruction "BSRD eoffset:13". The immediate value eoffset:13 is sign-extended to 22-bit and then added to (PC+2). In general, the 13-bit offset field appears as a label in assembly programs. If the instruction in the delay slot reads the value of A14, the value (PC+4) is read. The even offset eoffset:13 is encoded to 12bit signed offset in instruction map by dropping the least significant bit.

BRA/BRAD/BRT/BRTD/BRF/BRFD eoffset:11

In the branch instructions, the target address is (PC + 2 + eoffset:11). The immediate value eoffset:11 is sign-extended to 22-bit and then added to (PC+2). The "D" in the mnemonic stands for a delay slot. In general, the 11-bit offset field appears as a label in assembly programs. BRA and BRAD instructions always branch to the target address. BRT and BRTD instructions branch to the target address if T flag is set. BRF and BRFD instructions branch to the target address if T flag is cleared. BRAD/BRTD/BRFD instructions are delay slot branch instructions, therefore the instruction in the delay slot is executed before the branch to the target address or the branch decision is made. The even offset eoffset:11 is encoded to 10-bit signed offset in instruction map by dropping the least significant bit.

BRA/BRAD EC:2, eoffset:8

In the branch instructions, the target address is (PC + 2 + eoffset:8). The immediate value eoffset:8 is sign-extended to 22-bit and then added to (PC+2). The EC:2 field indicates one of the 4 external conditions from EC0 to EC3 (input pin signals to CalmRISC16). When the external condition corresponding to EC:2 is set, the program branches to the target address. BRAD has a delay slot. The even offset eoffset:8 is encoded to 7-bit signed offset in instruction map by dropping the least significant bit.

BNZD R6/R7, eoffset:8

In the branch instruction, the target address is (PC + 2 + eoffset:8). The immediate value eoffset:8 is sign-extended to 22-bit and then added to (PC+2). "BNZD R6, eoffset:8" instruction branches to the target address if Z0 flag is cleared. "BNZD R7, eoffset:8" instruction branches if Z1 flag is cleared. Before the branch operation, the instruction decrements R6/R7, updates Z0/Z1 flag according to the decrement result, and then executes the instruction in the delay slot. The instruction is used to manage loop counter with just one cycle overhead. In the end of the loop, the value of R6/R7 is –1. When the instruction in the delay slot read the Z0/Z1 flag, the result after the decrement is read. The even offset eoffset:8 is encoded to 7-bit signed offset in instruction map by dropping the least significant bit.

JMP/JPT/JPF/JSR addr:22

The target address of the instructions is addr:22. JMP always branches to the target address. JPT branches to the target address if the T flag is set. JPF branches if the T flag is cleared. JSR always branches to the target address with saving the return address (PC+4) into A14. The instructions are 2 word instructions.

JMP/JPT/JPF/JSR Ai

The target address of the instructions is the value of Ai. JMP always branches to the target address. JPT branches to the target address if the T flag is set. JPF branches if the T flag is cleared. JSR always branches to the target address with saving the return address (PC+2) into A14.

SWI #imm:6/ RET_SWI/RET_IRQ/RET_FIQ

refer to the section for interrupts.

RETD

The instruction branches to the address in A14 after the execution of the instruction in the delay slot. When there is no useful instruction adequate to the delay slot, "JMP A14" can be used instead of "RETD".

2.4. Bit Operation

The bit operations manipulate a bit in SR register or in a memory location.

BITR/BITS/BITC/BITT @[A8+R1], #imm:3

The source as well as the destination is the 8-bit data in the data memory at the address (A8 + R1). The #imm:3 field chooses a bit position among the 8 bits. BITR resets the bit #imm:3 of the source, and then writes the result to the destination, the same memory location. BITS sets the bit #imm:3 of the source, and then writes the result to the destination. BITC complements the bit #imm:3 of the source, and then writes the result to the destination. BITT does not write any data to the destination. T flag indicates whether the bit #imm:3 of the source is zero. In other words, when the bit #imm:3 of the source is zero, T flag is set. BITR and BITS can be used to implement a semaphore mechanism or lock acquisition/release.

CLRSR/SETSR/TSTSR bit

bit : FE, IE, TE, Z0, Z1, V, PM

CLRSR instruction clears the corresponding bit of SR. SETSR instruction sets the corresponding bit of SR. TSTSR tests whether the corresponding bit is zero, and stores the result in T flag. For example, when IE flag is zero, "TSTSR IE" instruction sets the T flag. We can clear the T flag by the instruction "CMP GT, R0, R0". We can set the T flag by the instruction "CMP EQ, R0, R0".

2.5. Miscellaneous instructions

SYS #imm:5

The instruction activates the output port nSYSID. The #imm:5 is transferred to outside on DA[4:0]. The most significant 17 bits remain unchanged. The instruction is for system command to outside such as power down modes.

COP #imm:13

The instruction activates the output port nCOPID. The #imm:13 is transferred to outside on COPIR[12:0]. The instruction is used to transfer instruction to coprocessor. The #imm:13 may be from 200h to 1FFFh.

CLD Rn, #imm:5 / CLD #imm:5, Rn

The instruction activates the output port nCOPID, nCLDID, and CLDWR. The least significant 13 bits of the instruction is transferred to outside on COPIR[12:0]. The #imm:5 is transferred to outside on DA[4:0]. The instructions move 16-bit data between Rn and a coprocessor register implied by the #imm:5 field. CLDWR signal indicates whether the data movement is from CalmRISC16 to coprocessor. The register Rn is one 8 registers from R0 to R7.

NOP

No operation.

BREAK

The software break instruction activates nBRK signal, and holds PA for one cycle. It's for debugging operation.

3. CalmRISC16 Instruction Map

	15				8	7		0
ADD Rn, #imm:7	0	0	0	0	Rn	0	Imm:7	
SUB Rn, #imm:7	0	0	0	0	Rn	1	Imm:7	
LD Rn, #imm:8	0	0	0	1	Rn	Imm:8		
LDW Rn, @[SP + edisp:9]	0	0	1	0	Rn	Edisp:9		
LDW @[SP + edisp:9], Ri	0	0	1	1	Ri	Edisp:9		
LDW Rn, @[Ai + edisp:5]	0	1	0	0	Rn	0	Ai	Edisp:5
LDW Rn, @[Ai + Rj]	0	1	0	0	Rn	1	Ai	Rj
LDW @[An + edisp:5], Ri	0	1	0	1	Ri	0	An	Edisp:5

LDW @[An + Rm], Ri	0	1	0	1	Ri				1	An			Rm			
LDB Dn, @[Ai + disp:4]	0	1	1	0	0	Dn			0	Ai			Disp:4			
LDB Dn, @[Ai + Rj]	0	1	1	0	0	Dn			1	Ai			Rj			
LDW An, @[Ai + disp:4]	0	1	1	0	1	An			0	Ai			Disp:4			
LDW An, @[Ai + Rj]	0	1	1	0	1	An			1	Ai			Rj			
LDB @[An + disp:4], Di	0	1	1	1	0	Di			0	An			Disp:4			
LDB @[An + Rm], Di	0	1	1	1	0	Di			1	An			Rm			
LDW @[An + disp:4], Ai	0	1	1	1	1	Ai			0	An			Disp:4			
LDW @[An + Rm], Ai	0	1	1	1	1	Ai			1	An			Rm			
ADD Rn, Ri	1	0	0	0	Rn				0	0	0	0	Ri			
SUB Rn, Ri	1	0	0	0	Rn				0	0	0	1	Ri			
ADC Rn, Ri	1	0	0	0	Rn				0	0	1	0	Ri			
SBC Rn, Ri	1	0	0	0	Rn				0	0	1	1	Ri			
AND Rn, Ri	1	0	0	0	Rn				0	1	0	0	Ri			
OR Rn, Ri	1	0	0	0	Rn				0	1	0	1	Ri			
XOR Rn, Ri	1	0	0	0	Rn				0	1	1	0	Ri			
TST Rn, Ri	1	0	0	0	Rn				0	1	1	1	Ri			
CMP GE, Rn, Ri	1	0	0	0	Rn				1	0	0	0	Ri			
CMP GT, Rn, Ri	1	0	0	0	Rn				1	0	0	1	Ri			
CMPU GE, Rn, Ri	1	0	0	0	Rn				1	0	1	0	Ri			
CMPU GT, Rn, Ri	1	0	0	0	Rn				1	0	1	1	Ri			
CMP EQ, Rn, Ri	1	0	0	0	Rn				1	1	0	0	Ri			
LD Rn, Ri	1	0	0	0	Rn				1	1	0	1	Ri			
RR Rn	1	0	0	0	0	0	0	0	1	1	1	0	Rn			
RL Rn	1	0	0	0	0	0	0	1	1	1	1	0	Rn			
RRC Rn	1	0	0	0	0	0	1	0	1	1	1	0	Rn			
SRB Rn	1	0	0	0	0	0	1	1	1	1	1	0	Rn			
SR Rn	1	0	0	0	0	1	0	0	1	1	1	0	Rn			
SRA Rn	1	0	0	0	0	1	0	1	1	1	1	0	Rn			
JPF Ai	1	0	0	0	0	1	1	0	1	1	1	0	0	Ai		
JPT Ai	1	0	0	0	0	1	1	0	1	1	1	0	1	Ai		
JMP Ai	1	0	0	0	0	1	1	1	1	1	1	0	0	Ai		
JSR Ai	1	0	0	0	0	1	1	1	1	1	1	0	1	Ai		
SLB Rn	1	0	0	0	1	0	0	0	1	1	1	0	Rn			
DT Rn	1	0	0	0	1	0	0	1	1	1	1	0	Rn			

INCC Rn	1	0	0	0	1	0	1	0	1	1	1	0	Rn		
DECC Rn	1	0	0	0	1	0	1	1	1	1	1	0	Rn		
COM Rn	1	0	0	0	1	1	0	0	1	1	1	0	Rn		
COM2 Rn	1	0	0	0	1	1	0	1	1	1	1	0	Rn		
COMC Rn	1	0	0	0	1	1	1	0	1	1	1	0	Rn		
EXT Rn	1	0	0	0	1	1	1	1	1	1	1	0	Rn		
ADD Rn, #imm:16	1	0	0	0	0	0	0	0	1	1	1	1	Rn		
ADD An, #imm:16	1	0	0	0	0	0	0	1	1	1	1	1	0	An	
SUB An, #imm:16	1	0	0	0	0	0	0	1	1	1	1	1	1	An	
ADC Rn, #imm:16	1	0	0	0	0	0	1	0	1	1	1	1	Rn		
SBC Rn, #imm:16	1	0	0	0	0	0	1	1	1	1	1	1	Rn		
AND Rn, #imm:16	1	0	0	0	0	1	0	0	1	1	1	1	Rn		
OR Rn, #imm:16	1	0	0	0	0	1	0	1	1	1	1	1	Rn		
XOR Rn, #imm:16	1	0	0	0	0	1	1	0	1	1	1	1	Rn		
TST Rn, #imm:16	1	0	0	0	0	1	1	1	1	1	1	1	Rn		
CMP GE, Rn, #imm:16	1	0	0	0	1	0	0	0	1	1	1	1	Rn		
CMP GT, Rn, #imm:16	1	0	0	0	1	0	0	1	1	1	1	1	Rn		
CMPU GE, Rn, #imm:16	1	0	0	0	1	0	1	0	1	1	1	1	Rn		
CMPU GT, Rn, #imm:16	1	0	0	0	1	0	1	1	1	1	1	1	Rn		
CMP EQ, Rn, #imm:16	1	0	0	0	1	1	0	0	1	1	1	1	Rn		
LD Rn, #imm:16	1	0	0	0	1	1	0	1	1	1	1	1	Rn		
Reserved	1	0	0	0	1	1	1		1	1	1	1			
CMP EQ, Dn, #imm:8	1	0	0	1	0	Dn			Imm:8						
AND R0, #imm:8	1	0	0	1	1	0	0	0	Imm:8						
OR R0, #imm:8	1	0	0	1	1	0	0	1	Imm:8						
XOR R0, #imm:8	1	0	0	1	1	0	1	0	Imm:8						
TST R0, #imm:8	1	0	0	1	1	0	1	1	Imm:8						
LDB R0, @[A8+ disp:8]	1	0	0	1	1	1	0	0	Disp:8						
LDB @[A8+ disp:8],R0	1	0	0	1	1	1	0	1	Disp:8						
BITR @[A8+R1], bs:3	1	0	0	1	1	1	1	0	0	0	0	0	0	Bs:3	
BITS @[A8+R1], bs:3	1	0	0	1	1	1	1	0	0	0	0	0	1	Bs:3	
BITC @[A8+R1], bs:3	1	0	0	1	1	1	1	0	0	0	0	1	0	Bs:3	
BITT @[A8+R1], bs:3	1	0	0	1	1	1	1	0	0	0	0	1	1	Bs:3	
SYS #imm:5	1	0	0	1	1	1	1	0	0	0	1	Imm:5			
SWI #imm:6	1	0	0	1	1	1	1	0	0	1	Imm:6				

CLRSR bs:3	1	0	0	1	1	1	1	0	1	0	0	0	0	Bs:3			
SETSR bs:3	1	0	0	1	1	1	1	0	1	0	0	0	1	Bs:3			
TSTSR bs:3	1	0	0	1	1	1	1	0	1	0	0	1	0	Bs:3			
NOP	1	0	0	1	1	1	1	0	1	0	0	1	1	0	0	0	
BREAK	1	0	0	1	1	1	1	0	1	0	0	1	1	0	0	1	
LD R0, SR	1	0	0	1	1	1	1	0	1	0	0	1	1	0	1	0	
LD SR, R0	1	0	0	1	1	1	1	0	1	0	0	1	1	0	1	1	
RET_FIQ	1	0	0	1	1	1	1	0	1	0	0	1	1	1	0	0	
RET_IRQ	1	0	0	1	1	1	1	0	1	0	0	1	1	1	0	1	
RET_SWI	1	0	0	1	1	1	1	0	1	0	0	1	1	1	1	0	
RETD	1	0	0	1	1	1	1	0	1	0	0	1	1	1	1	1	
LD R0, SPCL_FIQ	1	0	0	1	1	1	1	0	1	0	1	0	0	0	0	0	
LD R0, SPCH_FIQ	1	0	0	1	1	1	1	0	1	0	1	0	0	0	0	1	
LD R0, SSR_FIQ	1	0	0	1	1	1	1	0	1	0	1	0	0	0	1	0	
Reserved	1	0	0	1	1	1	1	0	1	0	1	0	0	0	1	1	
LD R0, SPCL_IRQ	1	0	0	1	1	1	1	0	1	0	1	0	0	1	0	0	
LD R0, SPCH_IRQ	1	0	0	1	1	1	1	0	1	0	1	0	0	1	0	1	
LD R0, SSR_IRQ	1	0	0	1	1	1	1	0	1	0	1	0	0	1	1	0	
Reserved	1	0	0	1	1	1	1	0	1	0	1	0	0	1	1	1	
Reserved	1	0	0	1	1	1	1	0	1	0	1	0	1	0	0		
LD R0, SSR_SWI	1	0	0	1	1	1	1	0	1	0	1	0	1	0	1	0	
Reserved	1	0	0	1	1	1	1	0	1	0	1	0	1	0	1	1	
Reserved	1	0	0	1	1	1	1	0	1	0	1	0	1	1			
LD SPCL_FIQ, R0	1	0	0	1	1	1	1	0	1	0	1	1	0	0	0	0	
LD SPCH_FIQ, R0	1	0	0	1	1	1	1	0	1	0	1	1	0	0	0	1	
LD SSR_FIQ, R0	1	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0	
Reserved	1	0	0	1	1	1	1	0	1	0	1	1	0	0	1	1	
LD SPCL_IRQ, R0	1	0	0	1	1	1	1	0	1	0	1	1	0	1	0	0	
LD SPCH_IRQ, R0	1	0	0	1	1	1	1	0	1	0	1	1	0	1	0	1	
LD SSR_IRQ, R0	1	0	0	1	1	1	1	0	1	0	1	1	0	1	1	0	
Reserved	1	0	0	1	1	1	1	0	1	0	1	1	0	1	1	1	
Reserved	1	0	0	1	1	1	1	0	1	0	1	1	1	0	0		
LD SSR_SWI, R0	1	0	0	1	1	1	1	0	1	0	1	1	1	0	1	0	
Reserved	1	0	0	1	1	1	1	0	1	0	1	1	1	0	1	1	
Reserved	1	0	0	1	1	1	1	0	1	0	1	1	1	1			

Reserved	1	0	0	1	1	1	1	0	1	1	0					
Reserved	1	0	0	1	1	1	1	0	1	1	1	0				
LD An, PC	1	0	0	1	1	1	1	0	1	1	1	1	0	An		
Reserved	1	0	0	1	1	1	1	0	1	1	1	1	1			
JPF adr:22	1	0	0	1	1	1	1	1	0	0	Adr[21:16]					
JPT adr:22	1	0	0	1	1	1	1	1	0	1	Adr[21:16]					
JMP adr:22	1	0	0	1	1	1	1	1	1	0	Adr[21:16]					
JSR adr:22	1	0	0	1	1	1	1	1	1	1	Adr[21:16]					
LDC Rn, @Ai	1	0	1	0	Rn				0	0	0	0	0	Ai		
Reserved	1	0	1	0					0	0	0	0	1			
LD Dn, Ei	1	0	1	0	0	Dn			0	0	0	1	0	Ei		
LD En, Di	1	0	1	0	0	Di			0	0	0	1	1	En		
CMP EQ, An, Ai	1	0	1	0	1	An			0	0	0	1	0	Ai		
LD An, Ai	1	0	1	0	1	An			0	0	0	1	1	Ai		
LDW Rn, @[Ai+disp:16]	1	0	1	0	Rn				0	0	1	0	0	Ai		
LDW @[An+disp:16], Ri	1	0	1	0	Ri				0	0	1	0	1	An		
LDB Dn, @[Ai+disp:16]	1	0	1	0	0	Dn			0	0	1	1	0	Ai		
LDB @[An+disp:16], Di	1	0	1	0	0	Di			0	0	1	1	1	An		
LDW An, @[Ai+disp:16]	1	0	1	0	1	An			0	0	1	1	0	Ai		
LDW @[An+disp:16], Ai	1	0	1	0	1	Ai			0	0	1	1	1	An		
CMP GE, Dn, #imm:6	1	0	1	0	0	Dn			0	1	Imm:6					
ADD An, #imm:5	1	0	1	0	1	An			0	1	0	imm:5				
SUB An, #imm:5	1	0	1	0	1	An			0	1	1	imm:5				
CMP EQ, An, #imm:22	1	0	1	0	0	An			1	0	Imm[21:16]					
LD An, #imm:22	1	0	1	0	1	An			1	0	Imm[21:16]					
ADD An, Ri	1	0	1	0	0	An			1	1	0	0	Ri			
SUB An, Ri	1	0	1	0	1	An			1	1	0	0	Ri			
MUL UU, Dn, Di	1	0	1	0	0	Dn			1	1	0	1	0	Di		
MUL US, Dn, Di	1	0	1	0	0	Dn			1	1	0	1	1	Di		
MUL SU, Dn, Di	1	0	1	0	1	Dn			1	1	0	1	0	Di		
MUL SS, Dn, Di	1	0	1	0	1	Dn			1	1	0	1	1	Di		
POP Rn[, Rm]	1	0	1	0	Rm				1	1	1	0	0	Rn		
Reserved	1	0	1	0	0				1	1	1	0	1			
POP An[, Am]	1	0	1	0	1	Am			1	1	1	0	1	An		
PUSH Rn[, Rm]	1	0	1	0	Rm				1	1	1	1	0	Rn		

Reserved	1	0	1	0	0				1	1	1	1	1			
PUSH An[, Am]	1	0	1	0	1	Am			1	1	1	1	1	An		
BSRD eoffset:13	1	0	1	1	Eoffset:13											
BRA EC:2, eoffset:8	1	1	0	0	0	0	0	EC:2		Eoffset:8						
Reserved	1	1	0	0	0	0	1									
BRAD EC:2, eoffset:8	1	1	0	0	0	1	0	EC:2		Eoffset:8						
BNZD H, eoffset:8	1	1	0	0	0	1	1	H	0	Eoffset:8						
Reserved	1	1	0	0	0	1	1		1							
BRA eoffset:11	1	1	0	0	1	0	Eoffset:11									
BRAD eoffset:11	1	1	0	0	1	1	Eoffset:11									
BRF eoffset:11	1	1	0	1	0	0	Eoffset:11									
BRFD eoffset:11	1	1	0	1	0	1	Eoffset:11									
BRT eoffset:11	1	1	0	1	1	0	Eoffset:11									
BRTD eoffset:11	1	1	0	1	1	1	Eoffset:11									
CLD Dn, imm:5	1	1	1	0	0	0	0	imm:5						0	Dn	
CLD imm:5, Di	1	1	1	0	0	0	0	imm:5						1	Di	
COP imm:13	1	1	1	Imm:13												

- Dn[15:0] : R0 ~ R7
- H[15:0] : R6, R7
- An[21:0] : A8 ~ A15, concatenation of En and Rn
- En[5:0] : E8 ~ E15, MS 6-bit of An
- SP : equal to A15
- EC:2 : EC0,EC1,EC2,EC3
- Disp : unsigned displacement
- Eoffset : even signed offset
- Edisp : even unsigned displacement

4. Quick Reference

Instruction	op1	op2	operation	flag
ADD SUB	Rn	#imm:7 Ri	op1 <- op1 + op2 op1 <- op1 + ~op2 + 1	T=C, Z0, Z1,V
LD	Rn	#imm:8 #imm:16 Ri	op1 <- op2	Z0, Z1

LDW	Rn	@[SP+edisp:9] @[Ai+edisp:5] @[Ai+Rj] @[Ai+disp:16]	op1 <- op2	-
LDW	@[SP+edisp:9] @[An+edisp:5] @[An+Rm] @[Ai+disp:16]	Ri	op1 <- op2	-
LDW	An	@[Ai+edisp:5] @[Ai+Rj] @[Ai+disp:16]	op1 <- op2	-
LDW	@[An+edisp:5] @[An+Rm] @[Ai+disp:16]	Ai	op1 <- op2	-
LDB	Dn	@[SP+disp:8] @[Ai+disp:4] @[Ai+Rj] @[Ai+disp:16]	op1<-{8'h0,op2[7:0]}	-
LDB	R0	@[A8+disp:8]	op1<-{8'h0,op2[7:0]}	-
LDB	@[SP+disp:8] @[An+disp:4] @[Ai+Rj] @[Ai+disp:16]	Di	op1 <- op2[7:0]	-
LDB	@[A8+disp:8]	R0	op1 <- op2[7:0]	-
ADC SBC	Rn	Ri #imm:16	op1 <- op1 + op2 + T op1 <- op1 + ~op2 + T	T=C,V, Z0,Z1
AND OR XOR	Rn	Ri #imm:16	op1 <- op1 & op2 op1 <- op1 | op2 op1 <- op1 ^ op2	T=Z, Z0,Z1
TST	Rn	Ri #imm:16	op1 & op2	T=Z
CMP GE CMP GT CMPU GE CMPU GT CMP EQ	Rn	Ri #imm:16	op1 + ~op2 + 1, T=~N op1 + ~op2 + 1, T=~N&~Z op1 + ~op2 + 1, T=C op1 + ~op2 + 1, T=C&~Z op1 + ~op2 + 1, T=Z	T

RR RL RRC SRB SR SRA SLB	Rn	-	op1 <- {op1[0],op1[15:1]} op1 <- {op1[14:0],op1[15]} op1 <- {T,op1[15:1]} op1 <- {8'h00,op1[15:8]} op1 <- {0,op1[15:1]} op1 <- {op1[15],op1[15:1]} op1 <- {op1[7:0],8'h00}	T=op1[0] T=op1[15] T=op1[0] T=op1[7] T=op1[0] T=op1[0] T= op1[8]
DT	Rn		op1 <- op1 + 0xffff	T=Z, Z0,Z1,V
COM	Rn		Op1 <- ~op1	T=Z,Z0, Z1
INCC DECC COM2 COMC	Rn		op1 <- op1 + T op1 <- op1 + 0xffff + T op1 <- ~op1 + 1 op1 <- ~op1 + T	T=C,Z0, Z1
EXT	Rn		op1<-{8{op1[7]},op1[7:0]}	Z0, Z1

JPF
JPT
JMP
JSR
	Ai
	addr:22
			if(T==0) PC <- op1
			if(T==1) PC <- op1
			PC <- op1
			A14 <- PC+(2|4), PC<-op1
				-

ADD	Rn	#imm:16	op1 <- op1 + op2	T=C, Z0,Z1,V
ADD SUB	An	#imm:16 #imm:5 Ri	op1 <- op1 + op2 op1 <- op1 – op2	-
CMP EQ	Dn	#imm:8	op1 + ~op2 + 1	T=Z
AND OR XOR TST	R0	#imm:8	Op1 <- op1 & {8'h00,op2} op1 <- op1 | {8'h00,op2} op1 <- op1 ^ {8'h00,op2} op1 & {8'h00,op2}	T=Z8
BITR BITS BITC BITT	@[A8+R1]	bs:3	op1[op2] <- 0 op1[op2] <- 1 op1[op2] <- ~op1[op2] op1[op2] <- op1[op2]	T= op1[op2]
SYS	#imm:5	-	DA[4:0] <- op1	-
SWI	#imm:6	-	A14 <- PC+2, PC <- op2*4	IE, TE

CLRSR SETSR TSTSR	bs:3	-	SR[op1] <- 0 SR[op1] <- 1 T <- ~SR[op1]	-
RETD	-	-	PC <- A14	-
LD	R0	SR SPCL_FIQ SPCH_FIQ SSR_FIQ SPCL_IRQ SPCH_IRQ SSR_IRQ SSR_SWI	op1 <- op2	-
LD	SR SPCL_FIQ SPCH_FIQ SSR_FIQ SPCL_IRQ SPCH_IRQ SSR_IRQ SSR_SWI	R0	op1 <- op2	
LD	An	PC Ai #imm:22	op1 <- op2 + 4 op1 <- op2 op1 <- op2	-
CMP EQ	An	Ai #imm:22	op1 + ~op2 + 1	T=Z22
LDC	Rn	@Ai	op1 <- PM[op2]	-
LD	Rn	Ei	op1 <- {10'h000, op2}	-
LD	En	Ri	op1 <- op2[5:0]	-
CMP GE	Dn	#imm:6	op1 + ~op2 + 1	T=~N
MUL UU MUL US MUL SU MUL SS	Dn	Di	op1<-{0,op1[7:0]} * {0,op2[7:0]} op1<-{0,op1[7:0]}*{op2[7],op2[7:0]} op1<-{op1[7],op1[7:0]}*{0,op2[7:0]} op1 <-{op1[7],op1[7:0]}* {op2[7],op2[7:0]}	-
POP	Rn	Rm	op1<-@[SP+2], op2<-@[SP+4], SP<-SP+4	-

PUSH	Rn	Rm	@[SP]<-op1,@[SP-2]<-op2,SP<-SP-4	-
POP	An	Am	En<-@[SP+2], Rn<-@[SP+4], Em<-@[SP+6], Rm<-@[SP+8], SP<-SP+8	-
PUSH	An	Am	@[SP]<-Rn, @[SP-2]<-En, @[SP-4]<-Rm, @[SP-6]<-Em, SP<-SP-8	-
BSRD	eoffset:13	-	A14 <- PC+2, PC <- PC + 2 + op1	-
BRA/BRAD	EC:2	eoffset:8	if(EC:2 == 1) PC <- PC + 2 + op2	-
BNZD	R6	eoffset:8	if(Z0 == 0) PC <- PC + 2 + op2 R6 <- R6 – 1	Z0
BNZD	R7	eoffset:8	if(Z1 == 0) PC <- PC + 2 + op2 R7 <- R7 – 1	Z1
BRA/BRAD	eoffset:11	-	PC <- PC + 2 + op1	-
BRF/BRFD	eoffset:11	-	if(T==0) PC <- PC + 2 + op1	-
BRT/BRTD	eoffset:11	-	if(T==1) PC <- PC + 2+op1	-
CLD	Dn	imm:5	op1 <- Coprocessor[op2]	-
CLD	imm:5	Di	Coprocessor[op1] <- op2	
COP	imm:13	-	COPIR <- op2	

ADC (1)

Add with Carry Register

ADC Rn, Ri

Description

The ADC (Add with Carry Register) instruction is used to synthesize 32-bit addition. If register pairs R0, R1 and R2, R3 hold 32-bit values (R0 and R2 hold the least-significant word), the following instructions leave the 32-bit sum in R0, R1:

```
ADD R0, R2
ADC R1, R3
```

The instruction ADC R0, R0 produces a single-bit Rotate Left with Carry (17-bit rotate through the carry) on R0.

ADC adds the value of register Rn, and the value of the Carry flag (stored in the T bit), and the value of register Ri, and stores the result in register Rn. The T bit and the V flag are updated based on the result.

15	14	13	12	11 … 8	7	6	5	4	3 … 0
1	0	0	0	Rn	0	0	1	0	Ri

Operation

Rn := Rn + Ri + T bit

T bit := Carry from (Rn + Ri + T bit)

V flag := Overflow from (Rn + Ri + T bit)

if(Rn == R6/R7) Z0/Z1 flag := ((Rn + Ri + T) == 0)

Exceptions

None.

Notes

None.

ADC (2)

Add with Carry Immediate

ADC Rn, #<imm:16>

Description

The ADC (Add with Carry Immediate) instruction is used to synthesize 32-bit addition with an immediate operand. If register pair R0, R1 holds a 32-bit value (R0 holds the least-significant word), the following instructions leave the 32-bit sum with 87653456h in R0, R1:

```
ADD R0, #3456h
ADC R1, #8765h
```

ADC adds the value of register Rn, and the value of the Carry flag (stored in the T bit), and the 16-bit immediate operand, and stores the result in register Rd. The T bit and the V flag are updated based on the result.

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

Operation

Rn := Rn + <imm:16> + T bit

T bit := Carry from (Rn + <imm:16> + T bit)

V flag := Overflow from (Rn + <imm:16> + T bit)

if(Rn == R6/R7) Z0/Z1 flag := ((Rn + <imm:16>) == 0)

Exceptions

None.

Notes

This is a 2-word instruction, where the 16-bit immediate follows the instruction word shown above. Unlike 1-word instructions, therefore, fetching of ADC Rn, <imm:16> takes 2 cycles.

ADD (1)

Add Register

ADD Rn, Ri

Description

The ADD (Add Register) instruction is used to add two 16-bit values in registers. 32-bit addition can be achieved by executing ADC instruction in pair with this instruction (see page 27).

ADD adds the value of register Rn, and the value of register Ri, and stores the result in register Rn. The T bit and the V flag are updated based on the result.

15	14	13	12	11 … 8	7	6	5	4	3 … 0
1	0	0	0	Rn	0	0	0	0	Ri

Operation

Rn := Rn + Ri

T bit := Carry from (Rn + Ri)

V flag := Overflow from (Rn + Ri)

if(Rn == R6/R7) Z0/Z1 flag := ((Rn + Ri) == 0)

Exceptions

None.

Notes

None.

ADD (2)

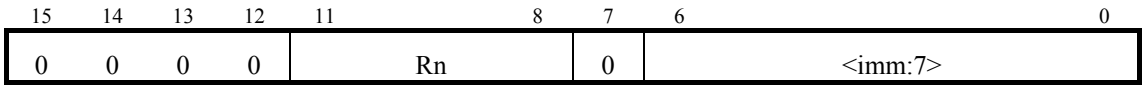
Add Small Immediate

ADD Rn, #<imm:7>

Description

This form of ADD instruction is used to add a 7-bit (positive) immediate value to a register

ADD adds the value of register Rn, and the value of <imm:7>, and stores the result in register Rn. The T bit and the V flag are updated based on the result.



15	14	13	12	11 … 8	7	6 … 0
0	0	0	0	Rn	0	<imm:7>

Operation

Rn := Rn + <imm:7>

T bit := Carry from (Rn + <imm:7>)

V flag := Overflow from (Rn + <imm:7>)

if(Rn == R6/R7) Z0/Z1 flag := ((Rn + <imm:7>) == 0)

Exceptions

None.

Notes

<imm:7> is an unsigned amount.

ADD (3)

Add Immediate

ADD Rn, #<imm:16>

Description

The ADD (Add Immediate) instruction is used to add a 16-bit immediate value to a register. 32-bit addition or subtraction can be achieved by executing ADC or SBC instruction in pair with this instruction (see page 28 and 99 for examples).

ADD adds the value of register Rn, and the value of <imm:16>, and stores the result in register Rn. The T bit and the V flag are updated based on the result.

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

Operation

Rn := Rn + <imm:16>

T bit := Carry from (Rn + <imm:16>)

V flag := Overflow from (Rn + <imm:16>)

if(Rn == R6/R7) Z0/Z1 flag := ((Rn + <imm:16>) == 0)

Exceptions

None.

Notes

This is a 2-word instruction, where the 16-bit immediate follows the instruction word shown above. Unlike 1-word instructions, therefore, fetching of ADD Rn, <imm:16> takes 2 cycles. The instruction “SUB Rn, #<imm:16>” does not exist.

The result of “SUB Rn, #<imm:16>” instruction is identical with the result of “ADD Rn, #(2’s complement of <imm:16>)” except when <imm:16> is zero. In that case, “SUB Rn, #<imm:7>” can be used.

ADD (4)

Add Extended Register

ADD An, Ri

Description

The ADD (Add Extended Register) instruction is used to add a 16-bit unsigned register value to a 22-bit register.

This instruction adds the value of 16-bit register Ri, and the value of 22-bit register An, and stores the result in register An.

15	14	13	12	11	10 … 8	7	6	5	4	3 … 0
1	0	1	0	0	An	1	1	0	0	Ri

Operation

An := An + Ri

Exceptions

None.

Notes

None.

ADD (5)

Add Immediate to Extended Register

ADD An, #<imm:16>

Description This form of ADD instruction is used to add a 16-bit unsigned immediate value to a 22-bit register.

This instruction adds the value of <imm:16> to the value of An, and stores the result in register An

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

Operation An := An + <imm:16>

Exceptions None.

Notes This is a 2-word instruction, where the 16-bit immediate follows the instruction word shown above. Unlike 1-word instructions, therefore, fetching of this instruction takes 2 cycles.

ADD (6)

Add 5-bit Immediate to Extended Register

ADD An, #<imm:5>

Description This form of ADD instruction is used to add a 5-bit unsigned immediate value to a 22-bit register.

This instruction adds the value of 5-bit immediate <imm:5>, and the value of 22-bit register An, and stores the result in register An.

15	14	13	12	11	10 … 8	7	6	5	4 … 0
1	0	1	0	1	An	0	1	0	<imm:5>

Operation An := An + <imm:5>

Exceptions None.

Notes <imm:5> is an unsigned amount.

AND (1)

AND Register

AND Rn, Ri

Description

The AND (AND Register) instruction is used to perform bitwise AND operation on two values in registers, Rn and Ri.

The result is stored in register Rn. The T bit is updated based on the result.

15	14	13	12	11 ... 8	7	6	5	4	3 ... 0
1	0	0	0	Rn	0	1	0	0	Ri

Operation

Rn := Rn & Ri

T bit := ((Rn & Ri) == 0)

if(Rn == R6/R7) Z0/Z1 flag := ((Rn & Ri) == 0)

Exceptions

None.

Notes

None.

AND (2)

AND Small Immediate

AND R0, #<imm:8>

Description

The AND (AND Small Immediate) instruction is used to perform an 8-bit bitwise AND operation on two values in register R0 and <imm:8>.

The result is stored in register R0. The T bit is updated based on the result.

15	14	13	12	11	10	9	8	7 … 0
1	0	0	1	1	0	0	0	<imm:8>

Operation

R0 := R0 & <imm:8>

T bit := ((R0 & <imm:8>)[7:0] == 0)

Exceptions

None.

Notes

The register used in this operation is fixed to R0. Therefore, the operand should be placed in R0 before this instruction executes. <imm:8> is zero-extended to a 16-bit value before operation.

AND (3)

AND Large Immediate

AND Rn, #<imm:16>

Description

This type of AND instruction is used to perform bitwise AND operation on two values in register Rn and <imm:16>.

The result is stored in register Rn. The T bit is updated based on the result.

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

Operation

Rn := Rn & <imm:16>

T bit := ((Rn & <imm:16>) == 0)

if(Rn == R6/R7) Z0/Z1 flag := ((Rn & <imm:16>) == 0)

Exceptions

None.

Notes

This is a 2-word instruction, where the 16-bit immediate follows the instruction word shown above. Unlike 1-word instructions, therefore, fetching of this instruction takes 2 cycles.

BITop

BIT Operation

BITop @[A8+R1], #<bs:3>

Description

The BITop (Bit Operation) instruction is used to perform a bit operation on an 8-bit memory value. The allowed operations include reset (BITR), set (BITS), complement (BITC), and test (BITT).

BITop fetches the value of memory location specified by @(A8+R1), performs the specified operation on the specified bit, and stores the result back into the same memory location

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	1	1	0	0	0	0	OP		<bs:3>		

Operation

```
Temp := MEM[A8+R1]
T bit := ~Temp[<bs:3>]
if (BITop != BITT) {
   Result := BITop(Temp, <bs:3>)
   MEM[A8+R1] := Result
}
```

Here, BITop is BITR (OP == 00) | BITS (01) | BITC (10) | BITT (11). The bit location of these operations is specified by <bs:3>.

Exceptions

None.

Notes

The address used to access data memory is obtained from the addition of two registers A8 and R1. No other registers can be used for this address calculation.

BNZD

Branch Not Zero with Autodecrement

BNZD H, <eoffset:8>

Description

The BNZD (Branch Not Zero with Delay Slot) instruction is used to change the program flow when the specified register value does not evaluate to zero. After evaluation, the value in register is automatically decremented. A typical usage of this instruction is as a backward branch at the end of a loop.

```
LOOP:
        ...
        BNZD R6, LOOP       // if (Z0 != 0) go back to LOOP
        ADD R4, 3           // delay slot
```

In the above example, R6 is used as the loop counter. After specified loop iterations, BNZD is not taken and the control will come out of the loop, and R6 will have -1. For a loop with "N" iterations, the counter register used should be initially set to "(N-1)". BNZD has a single delay slot; the instruction that immediately follows BNZD will be executed always regardless of whether BNZD is taken or not.

15	14	13	12	11	10	9	8	7	6 … 0
1	1	0	0	0	1	1	H	0	<eoffset:8>

Operation

```
if(H == R6) {
   if(Z0 != 0)    PC := PC + 2 + <eoffset:8>
   R6 := R6 – 1
   Z0 := ((R6-1) == 0)
} else {      // H == R7
  Same mechanism as the case R6
}
```

H is a register specifier denoting either R6 or R7.

Exceptions

None.

Notes

When BNZD checks if H is zero by looking up the Z0 (for R6) or Z1 (for R7) bit in SR, these flags are updated as BNZD decrements the value of the register. For the first iteration, however, the user is responsible for resetting the flag, Z0 or Z1, before the loop starts execution.

BR

Conditional Branch

BRtype <eoffset:11>

Description

The BR (Conditional Branch) instruction is used to change the program flow conditionally or unconditionally. The allowed forms of the instruction include BRA (always), BRAD (always with delay slot), BRT (when T bit is set), BRTD (when T bit is set, with delay slot), BRF (when T bit is clear), and BRFD (when T bit is clear, with delay slot).

The branch target address is calculated by

1. sign-extending <offset:10> to 22 bits
2. adding this to the PC (which contains the address of the branch instruction plus 1)

15	14	13	12	11	10	9	0
1	1	0	<Type>		D	<eoffset:11>	

Operation

if (Condition)

PC := PC + 2 + <eoffset:11>

Here, the <Type> field determines whether this branch is BRA (01), BRF (10), or BRT (11). If D is set, the branch instruction has one branch delay slot, meaning that the instruction following the branch will be executed always, regardless of the branch outcome. If D is clear, the immediately following instruction is NOT executed if the branch is taken.

Exceptions

None.

Notes

None.

BRA EC

Branch on External Condition

BRA(D) EC:2 <eoffset:8>

Description

The BRA EC (Branch on External Condition) instruction is used to change the program flow when a certain external condition is set. A typical usage of this instruction is to branch after a coprocessor operation as shown below:

```
        COP <operation>
        NOP
        NOP
        BRA EC0    OVERFLOW
        ...
OVERFLOW: ...
        ...
```

The BRA EC instruction checks the specified external condition (instead of checking the T bit as other branch instructions, see page 40) and branch to the specified program address. There can be up to 4 external conditions, specified by the <EC:2> field in the instruction.

15	14	13	12	11	10	9	8	7	6	0
1	1	0	0	0	D	0	<EC:2>		<eoffset:8>	

Operation

```
if (ExternalCondition_n == True)
  PC := PC + 2 + <eoffset:8>
```

Exceptions

None.

Notes

None.

BREAK

BREAK

BREAK

Description

The BREAK instruction suspends the CalmRISC core for 1 cycle by keeping PC from increasing. Processor resumes execution after 1 cycle. This instruction is used for debugging purposes only and thus should not be used in normal operating modes. A core signal nBRK is asserted low for the cycle.

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

Operation

No operation with PC suspended for a single cycle.

Exceptions

None.

Notes

None.

BSRD

Branch Subroutine with Delay Slot

BSRD <eoffset:13>

Description

The BSRD (Branch Subroutine with Delay slot) instruction is used to change the program flow to a subroutine by assigning the address of the subroutine to PC after saving the return address (PC+4) in the link register, or A14.

The address of the subroutine is calculated by:

1. sign-extending <eoffset:13> to 22 bits
2. adding this to the PC (which contains the address of the branch instruction plus 1)

After executing the subroutine, the program flow can return back to the instruction that follows the BSRD instruction by setting PC with the value stored in A14 (see JMP Ai instruction in page 58 and RET instruction in page 91). This instruction has a delay slot; the instruction that immediately follows BSRD will be always executed.

15	14	13	12	11 … 0
1	0	1	1	<eoffset:13>

Operation

A14 := PC + 4

PC := PC + 2 + <eoffset:13>

Exceptions

None.

Notes

None.

CLD

Coprocessor Load

CLD Dn, <imm:5> / CLD <imm:5>, Di

Description

The CLD (Coprocessor Load) instruction is used to transfer data from and to coprocessor by generating the core signals nCLDID and CLDWR. The content of DA[4:0] is <imm:5>, the address of coprocessor register to be read or written.

When a data item is read from coprocessor (CLD Dn, <imm:5>), it is stored in Dn.

When a data item is written to coprocessor, it should be prepared in Di.

15	14	13	12	11	10		8	7	6	5	4		0
1	1	1	0	0	0	0	imm:5					M	Dn/Di

Operation

(M == 0, read)

DA[4:0] := <imm:5>

nCLDID := 0

CLDWR := 0

Dn := (<imm:5>)

(M == 1, write)

DA[4:0] := <imm:5>

nCLDID := 0

CLDWR := 1

(<imm:5>) := Di

Exceptions

None.

Notes

None.

CLRSR

Clear SR

CLRSR bs:3

Description The CLRSR (Clear SR) instruction is used to clear a specified bit in SR as follows:

CLRSR FE / IE / TE / V / Z0 / Z1 / PM

To clear the T bit, one can do as follows:

CMP GT, R0, R0

To turn on a specified bit in SR, the SETSR instruction (in page 100) is used.

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

Operation SR[<bs:3>] := 0

Exceptions None.

Notes None.

CMP (1)

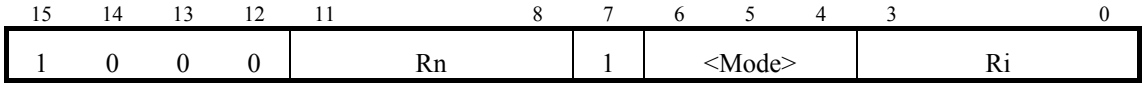
Compare Register

CMPmode Rn, Ri

Description

The CMP (Compare Register) instruction is used to compare two values in registers Rn and Ri. The allowed modes include GE (Greater or Equal), GT (Greater Than), UGE (Unsigned Greater or Equal), UGT (Unsigned Greater Than), and EQ (Equal).

CMP subtracts the value of Ri from the value of Rn and performs comparison based on the result. The contents of Rn and Ri are not changed after this operation. The T bit is updated for later reference.



Operation

Temp := Rn - Ri

T bit := ~Negative if (<Mode> == GE)

~Negative && ~Zero if (<Mode> == GT)

Carry if (<Mode> == UGE)

Carry && ~Zero if (<Mode> == UGT)

Zero if (<Mode> == EQ)

<Mode> encoding: GE (000), GT (001), UGE (010), UGT (011), and EQ (100).

Exceptions

None.

Notes

None.

CMP (2)

Compare Immediate

CMPmode Rn, #<imm:16>

Description

The CMP (Compare Immediate) instruction is used to compare two values in register Rn and <imm:16>. The allowed modes include GE (Greater or Equal), GT (Greater Than), UGE (Unsigned Greater or Equal), UGT (Unsigned Greater Than), and EQ (Equal).

CMP subtracts the value of <imm:16> from the value of Rn and performs comparison based on the result. The contents of Rn is not changed, however, after this operation. The T bit is updated for later reference.

15	14	13	12	11	10 ... 8	7	6	5	4	3 ... 0
1	0	0	0	1	<Mode>	1	1	1	1	Rn

Operation

Temp := Rn - <imm:16>

T bit := ~Negative if (<Mode> == GE)
~Negative && ~Zero if (<Mode> == GT)
Carry if (<Mode> == UGE)
Carry && ~Zero if (<Mode> == UGT)
Zero if (<Mode> == EQ)

<Mode> encoding: GE (000), GT (001), UGE (010), UGT (011), and EQ (100).

Exceptions

None.

Notes

This is a 2-word instruction, where the 16-bit immediate follows the instruction word shown above. Unlike 1-word instructions, therefore, fetching of CMPmode #<imm:16> takes 2 cycles.

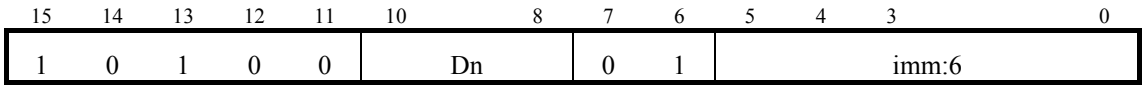
CMP (3)

Compare Short Immediate

CMP GE, Dn, #<imm:6>

Description

The CMP (Compare Immediate) instruction is used to perform signed-comparison of the register Dn and an unsigned immediate value <imm:6>. Dn is one of the registers from R0 to R7. CMP subtracts the value of <imm:6> from the value of Dn and performs signed-comparison based on the result. The contents of Dn is not changed, however, after this operation. The T bit is updated for later reference.



Operation

T bit := ~Negative of (Rn - <imm:6>)

Exceptions

None.

Notes

None

CMPEQ (1)

Compare Equal Extended Register

CMP EQ, An, Ai

Description

The CMP EQ (Compare Equal Extended Register) instruction is used to compare two values in registers An and Ai.

This instruction is a restricted form of more general CMPmode instructions for a 22-bit equality comparison between register values.

15	14	13	12	11	10 … 8	7	6	5	4	3	2 … 0
1	0	1	0	1	An	0	0	0	1	0	Ai

Operation

T bit := (An == Ai)

An or Ai refers to registers from A8 to A15 with their 6-bit extensions.

Exceptions

None.

Notes

None.

CMPEQ (2)

Compare Equal Small Immediate

CMP EQ, Dn, #<imm:8>

Description

The CMP EQ (Compare Equal Small Immediate) instruction is used to compare two values in register Dn and <imm:8>. <imm:8> is zero-extended to 16 bits before comparison.

This instruction is a restricted form of more general CMPmode instructions for an 8-bit equality comparison between a register value and an immediate value.

15	14	13	12	11	10 … 8	7 … 0
1	0	0	1	0	Dn	<imm:8>

Operation

T bit := ((Dn - <imm:8>) == 0)

Dn refers to registers R0 - R8.

Exceptions

None.

Notes

None.

CMPEQ (3)

Compare Equal Large Immediate

CMP EQ An, #<imm:22>

Description

The CMP EQ (Compare Equal Large Immediate) instruction is used to compare two values in register An and <imm:22>.

This instruction is a restricted form of more general CMPmode instructions for a 22-bit equality comparison between a register value and an immediate value.

15	14	13	12	11	10 … 8	7	6	5 … 0
1	0	1	0	0	An	1	0	<imm:22>[21:16]

Operation

T bit := Zero from (An - <imm:22>)

An refers to registers from A8 to A15 with their 6-bit extensions.

Exceptions

None.

Notes

This is a 2-word instruction, where the 16-bit immediate (<imm:22>[15:0]) follows the instruction word shown above. Unlike 1-word instructions, therefore, fetching of CMP EQ <imm:22> takes 2 cycles.

COM

Complement

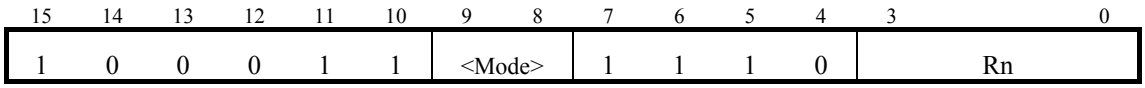
COMmode Rn

Description

The COM (Complement) instruction is used to compute 1's or 2's complement of a register value Rn. Utilizing various modes, 32-bit complement operation can be done. If register pair R0, R1 holds a 32-bit value (R0 holds the least-significant word), the following instructions leave the 32-bit 2's complement in R0, R1:

```
COM2 R0          // 2's complement
COMC R1          // 2's complement with carry
```

COM computes the 1's complement of the value of register Rn. COM2 computes the 2's complement, and COMC computes the 2's complement value when T bit has been set. If T bit is clear, COM2 is equivalent to COM.



15	14	13	12	11	10	9	8	7	6	5	4	3	0
1	0	0	0	1	1	<Mode>		1	1	1	0	Rn	

Operation

```
if (<Mode> == 00) {      // COM
   Rn := ~Rn
   T bit := (Rn == 0)
}
if (<Mode> == 01) {      // COM2
   Rn := ~Rn + 1
   T bit := Carry from (~Rn + 1)
}
if (<Mode> == 10) {      // COMC
   Rn := ~Rn + T bit
   T bit := Carry from (~Rn + T)
}
```

Encoding of <Mode>:

00: COM, 01: COM2, 10: COMC

if(Rn == R6/R7) Z0/Z1 := Zero flag of the result.

Exceptions

None.

Notes

None.

COP

Coprocessor

COP <imm:13>

Description The COP (Coprocessor) instruction is used to perform a coprocessor operation, specified by <imm:13>. Certain coprocessor operations set external conditions, upon which branches can be executed (see BRECn instructions, from page 41).

The <imm:13> should be greater or equal to 0x200.

15	14	13	12 ... 0
1	1	1	<imm:13>

Operation Perform a coprocessor operation by placing signals on core output pins as follows:

Core output signal COPIR[12:0] := <imm:13>

Core output signal nCOPID := LOW

Exceptions None.

Notes None.

DECC

Decrement with Carry

DECC Rn

Description

The DECC (Decrement with Carry) instruction is used to synthesize 32-bit decrement. If register pair R0, R1 holds a 32-bit value (R0 holds the least-significant word), the following instructions leave the 32-bit decremented value in R0, R1:

```
DEC R0          // this is implemented by ADD R0, -1
DECC R1
```

DECC decrements the value of Rn by 1 only if the Carry flag (stored in the T bit) is clear, and stores the result back in register Rn. The T bit and the V flag are updated based on the result.

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

Operation

Rn := Rn - 1 + T bit

T bit := Carry from (Rn - 1 + T bit)

V flag := Overflow from (Rn -1 + T bit)

if(Rn == R6/R7) Z0/Z1 := ((Rn – 1 + T) == 0)

Exceptions

None.

Notes

None.

DT

Decrement and Test

DT Rn

Description

The DT (Decrement and Test) instruction is used to decrement the value of a specified register and test it. This instruction provides a compact way to control register indexing for loops. The T bit and the V flag are updated based on the result.

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

Operation

Rn := Rn - 1

T bit := ((Rn - 1) == 0)

V flag := Overflow from (Rn - 1)

if(Rn == R6/R7) Z0/Z1 := ((Rn – 1) == 0)

Exceptions

None.

Notes

None.

EXT

Sign-Extend

EXT Rn

Description The EXT (Sign Extend) instruction is used to sign-extend an 8-bit value in Rn. This instruction copies Rn[7] to Rn[15:8].

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

Operation All bits from Rn[15] to Rn[8] := Rn[7]

if(Rn == R6/R7) Z0/Z1 := (Result == 0)

Exceptions None.

Notes None.

INCC

Increment with Carry

INCC Rn

Description

The INCC (Increment with Carry) instruction is used to synthesize 32-bit increment. If register pair R0, R1 holds a 32-bit value (R0 holds the least-significant word), the following instructions leave the 32-bit incremented value in R0, R1:

```
INC R0          // will be replaced by ADD R0, 1
INCC R1
```

INCC increments the value of Rn by 1 only if the Carry flag (stored in the T bit) is set, and stores the result back in register Rn. The T bit and the V flag are updated based on the result.

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

Operation

Rn := Rn + T bit

T bit := Carry from (Rn + T bit)

V flag := Overflow from (Rn + T bit)

if(Rn == R6/R7) Z0/Z1 := ((Rn + T0) == 0)

Exceptions

None.

Notes

None.

JMP (1)

Jump Register

JPF/JPT/JMP/JSR Ai

Description

The Jump Register instructions change the program flow by assigning the value of register Ai into PC.

JPF and JPT are conditional jumps that check the T bit to determine whether or not to jump to the target address. JMP unconditionally jumps to the target. JSR is an unconditional jump but saves the return address (the immediately following instruction to JSR) in the link register, A14. At the end of each subroutine, JMP A14 will change the program flow back to the original call site.

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

Operation

(M == 00, JPF)
if (T bit == FALSE)
  PC := Ai
(M == 01, JPT)
if (T bit == TRUE)
  PC := Ai
(M == 10, JMP)
PC := Ai
(M == 11, JSR)
A14 := PC + 2
PC := Ai

Exceptions

None.

Notes

There is no delay slot for these instructions. Therefore, when conditional branch JPF or JPT is taken, the instruction in the pipeline which is fetched from PC+2 will be squashed. In case of JMP and JSR (always taken), the following instruction fetched will be always squashed.

JMP (2)

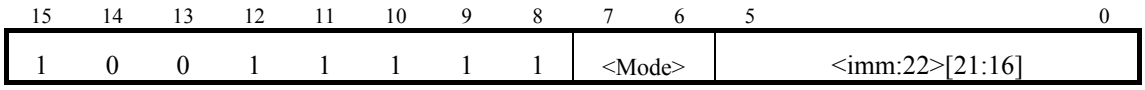
Jump Immediate

JPF/JPT/JMP/JSR <imm:22>

Description

The Jump Immediate instructions change the program flow by assigning the value of <imm:22> into PC.

JPF and JPT are conditional jumps that check the T bit to determine whether or not to jump to the target address. JMP unconditionally jumps to the target. JSR is an unconditional jump but saves the return address (the immediately following instruction to JSR) in the link register, A14. At the end of each subroutine, JMP A14 will change the program flow back to the original call site.



Operation

(<Mode> == 00, JPF)
if (T bit == FALSE)
  PC := <imm:22>
(<Mode> == 01, JPT)
if (T bit == TRUE)
  PC := <imm:22>
(<Mode> == 10, JMP)
PC := <imm:22>
(<Mode> == 11, JSR)
A14 := PC + 4
PC := <imm:22>

Exceptions

None.

Notes

These are 2-word instructions, where the 16-bit immediate (<imm:22>[15:0]) follows the instruction word shown above. As fetching of a 2-word instruction takes 2 cycles, no later instructions will be in processor pipeline when the branch is taken (thus no squashing).

LD (1)

Load Register

LD Rn, Ri

Description The LD (Load Register) instruction is used to transfer a register value to a register.

15	14	13	12	11 … 8	7	6	5	4	3 … 0
1	0	0	0	Rn	1	1	0	1	Ri

Operation Rn := Ri

if(Rn == R6/R7) Z0/Z1 := (Ri == 0)

Exceptions None.

Notes None.

LD (2)

Load Extended Register

LD An, Ai

Description This form of LD instruction (Load Extended Register) is used to load a 22-bit register value to a 22-bit register.

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

Operation An := Ai

Exceptions None.

Notes None.

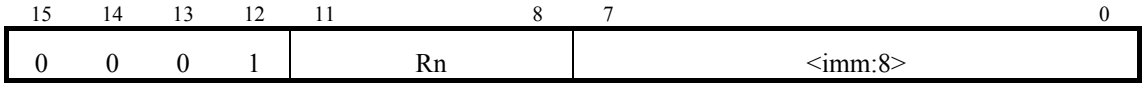
LD (3)

Load

Short Immediate

LD Rn, #<imm:8>

Description The LD (Load Short Immediate) instruction is used to load an 8-bit immediate value to a register.



Operation Rn[15:8] := 0, Rn[7:0] := <imm:8>

if(Rn == R6/R7) Z0/Z1 := (<imm:8> == 0)

Exceptions None.

Notes None.

LD (4)

Load Immediate

LD Rn, #<imm:16>

Description This form of LD instruction (Load Immediate) is used to load a 16-bit immediate value to a register.

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

Operation Rn := <imm:16>
if(Rn == R6/R7) Z0/Z1 := (<imm:16> == 0)

Exceptions None.

Notes This is a 2-word instruction, where the 16-bit immediate follows the instruction word shown above. Unlike 1-word instructions, therefore, fetching of this instruction takes 2 cycles.

LD (5)

Load Large Immediate

LD An, #<imm:22>

Description This form of LD instruction (Load Large Immediate) is used to load a 22-bit immediate value to an extended register An.

15	14	13	12	11	10 … 8	7	6	5 … 0
1	0	1	0	1	An	1	0	<imm:22>[21:16]

Operation An := <imm:22>

Exceptions None.

Notes This is a 2-word instruction, where the 16-bit immediate (<imm:22>[15:0]) follows the instruction word shown above. Unlike 1-word instructions, therefore, fetching of this instruction takes 2 cycles.

LD RExt

Load Register Extension

LD Dn, Ei / LD En, Di

Description

The LD RExt (Load Register Extension) instructions are used to transfer a register value to and from a 6-bit extension register.

15	14	13	12	11	8	7	6	5	4	3	2	0
1	0	1	0	0	Dn(or Di)	0	0	0	1	M	Ei (or En)	

Operation

(M == 0, LD Dn, Ei)

Dn := Ei (zero-extended to 16 bits)

(M == 1, LD En, Di)

En := Di (lower 6 bits only)

Exceptions

None.

Notes

None.

LDB (1)

Load Byte
Register Disp.

LDB Dn, @[Ai+<disp:4>] / LDB @[An+<disp:4>], Di

Description

The LDB (Load Byte Register Displacement) instruction is used to load a byte from or to data memory at the location specified by the register Ai and a 4-bit displacement.

15	14	13	12	11	10 ... 8	7	6 ... 4	3 ... 0
0	1	1	M	0	Dn or Di	0	Ai or An	<disp:4>

Operation

(M == 0, LDB Dn, @[Ai+<disp:4>])

Dn := DM[(Ai+<disp:4>)]

(M == 1, LDB @[An+<disp:4>], Di)

DM[(An+<disp:4>)] := Di

Exceptions

None.

Notes

None.

LDB (2)

Load Byte
Register Large Disp.

LDB Dn, @[Ai+<disp:16>] / LDB @[An+<disp:16>], Di

Description

The LDB (Load Byte Register Large Displacement) instruction is used to load a byte from or to data memory at the location specified by the register Ai and a 16-bit displacement.

15	14	13	12	11	10 ... 8	7	6	5	4	3	2 ... 0
1	0	1	0	0	Dn or Di	0	0	1	1	M	Ai or An

Operation

(M == 0, LDB Dn, @[Ai+<disp:16>])
Dn := DM[(Ai+<disp:16>)]
(M == 1, LDB @[An+<disp:16>], Di)
DM[(An+<disp:16>)] := Di

Exceptions

None.

Notes

This is a 2-word instruction, where the 16-bit immediate follows the instruction word shown above. Unlike 1-word instructions, therefore, fetching of this instruction takes 2 cycles.

LDB (3)

Load Byte
Register Indexed

LDB Dn, @[Ai+Rj] / LDB @[An+Rm], Di

Description

The LDB (Load Byte Register Indexed) instruction is used to load a byte from or to data memory at the location specified by the register Ai (or An) and the second register Rj (or Rm).

15	14	13	12	11	10 ... 8	7	6 ... 4	3 ... 0
0	1	1	M	0	Dn or Di	1	Ai or An	Rj or Rm

Operation

(M == 0, LDB Dn, @[Ai+Rj])
Dn := DM[(Ai+Rj]
(M == 1, LDB @[An+Rm], Di)
DM[(An+Rm)] := Di

Exceptions

None.

Notes

None.

LDB (4)

Load Byte to R0 Register Disp.

LDB R0, @[A8+<disp:8>] / LDB @[A8+<disp:8>], A8

Description

The LDB (Load Byte to R0 Register Displacement) instruction is used to load a byte from or to data memory at the location specified by the register A8 and an 8-bit displacement.

15	14	13	12	11	10	9	8	7 ... 0
1	0	0	1	1	1	0	M	<disp:8>

Operation

(M == 0, LDB R0, @[A8+<disp:8>])

R0 := DM[(A8+<disp:8>]

(M == 1, LDB @[A8+<disp:8>], R0)

DM[(A8+<disp:8>)] := R0

Exceptions

None.

Notes

This single-word instruction allows a user to access a wider range of data memory than the LDB (1) instruction by providing a larger displacement, at the expense of the restrictions that only the R0 and A8 registers are used for data transfer and address computation.

LDC

Load Code

LDC Rn, @Ai

Description The LDC instruction is used to transfer a register value from the program memory. The program memory address is specified by the 22-bit register An. LDC is useful to look up the data stored in program memory, such as the coefficient table for certain numerical algorithms.

15	14	13	12	11 … 8	7	6	5	4	3	2 … 0
1	0	1	0	Rn	0	0	0	0	0	Ai

Operation Rn := PM[Ai]

Exceptions None.

Notes None.

LD PC

Load
Program Counter

LD An, PC

Description

The LD PC (Load Program Counter) instruction is used to transfer the value of PC into a 22-bit register An. This instruction provides a way to implement position independent code (PIC) on CalmRISC16 even in the absence of general virtual memory support. After executing this instruction, An will be used to compute a PC-relative location of a data item or a code section.

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

Operation

An := PC + 4

Exceptions

None.

Notes

None.

LD SvR (1)

Load from Saved Register

LD R0, SPCL_* / LD R0, SPCH_* / LD R0, SSR_*

Description

The LD SvR (Load from Saved Register) instructions are used to transfer a value from the specified interrupt register, e.g., SSR_FIQ. Only R0 register is used for this data transfer.

15	14	13	12	11	10	9	8	7	6	5	4	3 ... 0
1	0	0	1	1	1	1	0	1	0	1	0	<RS>

Operation

R0 := <specified_saved_register>

Encoding for <RS> (Register Specifier):

0000: SPCL_FIQ, 0001: SPCH_FIQ, 0010: SSR_FIQ,

0100: SPCL_IRQ, 0101: SPCH_IRQ, 0110: SSR_IRQ,

1010: SSR_SWI

Exceptions

None.

Notes

None.

LD SvR (2)

Load to Saved Register

LD SPCL_*, R0 / LD SPCH_*, R0 / LD SSR_*, R0

Description

The LD SvR (Load to Saved Register) instructions are used to transfer a value to the specified interrupt register, e.g., SSR_FIQ. Only R0 register is used for this data transfer.

15	14	13	12	11	10	9	8	7	6	5	4	3 ~ 0
1	0	0	1	1	1	1	0	1	0	1	1	<RS>

Operation

<specified_saved_register> := R0

Encoding for <RS> (Register Specifier):

0000: SPCL_FIQ, 0001: SPCH_FIQ, 0010: SSR_FIQ,

0100: SPCL_IRQ, 0101: SPCH_IRQ, 0110: SSR_IRQ,

1010: SSR_SWI

Exceptions

None.

Notes

None.

LD SR

Load
Status Register

LD R0, SR / LD SR, R0

Description The LD SR (Load Status Register) instruction is used to transfer a value to and from SR. Only R0 register is used for this operation.

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

Operation

(M == 0, LD R0, SR)

R0 := SR

(M == 1, LD SR, R0)

SR := R0

Exceptions None.

Notes None.

LDW (1)

Load Word
Stack Disp.

LDW Rn, @[SP+<edisp:9>] / LDW @[SP+<edisp:9>], Ri

Description

The LDW (Load Word Stack Displacement) instruction is used to load a word from or to data memory at the location specified by the SP register (or A15) and an even 9-bit displacement. <edisp:9>, from 0 to 510, is encoded into 8-bit displacement by dropping the least significant bit.

15	14	13	12	11 ... 8	7 ... 0
0	0	1	M	Rn or Ri	<edisp:9>

Operation

(M == 0, LDW Rn, @[SP+<edisp:9>])
Rn := DM[(SP + <edisp:9>)]
(M == 1, LDW @[SP+<edisp:9>], Ri)
DM[(SP + <edisp:9>)] := Ri

Exceptions

None.

Notes

For memory transfer per word, the (byte) address need to be aligned to be even. Thus, if (SP + <edisp:9>) is an odd number, it will be made even by clearing the least significant bit. <edisp:9> can denote an even number from 0 to 510.

LDW (2)

Load Word Register

Small Disp.

LDW Rn, @[Ai+<edisp:5>] / LDW @[An+<edisp:5>], Ri

Description

The LDW (Load Word Register Displacement) instruction is used to load a word from or to data memory at the location specified by the register Ai and a 5-bit even displacement from 0 to 30. <edisp:5> is encoded to 4-bit number by dropping the least significant bit.

15	14	13	12	11 ... 8	7	6 ... 4	3 ... 0
0	1	0	M	Rn or Ri	0	Ai or An	<edisp:5>

Operation

(M == 0, LDW Rn, @[Ai+<edisp:5>])

Rn := DM[(Ai + <edisp:5>)]

(M == 1, LDW @[An+<edisp:5>], Ri)

DM[(An + <edisp:5>)] := Ri

Exceptions

None.

Notes

For memory transfer per word, the (byte) address need to be aligned to be even. Thus, if (Ai + <edisp:5>) is an odd number, it will be made even by clearing the least significant bit. <edisp:5> can denote an even number from 0 to 30.

LDW (3)

Load Word
Register Disp.

LDW Rn, @[Ai+<disp:16>] / LDW @[An+<disp:16>], Ri

Description

The LDW (Load Word Register Large Displacement) instruction is used to load a word from or to data memory at the location specified by the register Ai and a 16-bit displacement.

15	14	13	12	11 ... 8	7	6	5	4	3	2 ... 0
1	0	1	0	Rn or Ri	0	0	1	0	M	Ai or An

Operation

(M == 0, LDW Rn, @[Ai+<disp:16>])
Rn := DM[(Ai + <disp:16>)]
(M == 1, LDW @[An+<disp:16>], Ri)
DM[(An + <disp:16>)] := Ri

Exceptions

None.

Notes

This is a 2-word instruction, where the 16-bit immediate follows the instruction word shown above. Unlike 1-word instructions, therefore, fetching of this instruction takes 2 cycles. For memory transfer per word, the (byte) address need to be aligned to be even. Thus, if (Ai + <disp:16>) is an odd number, it will be made even by clearing the least significant bit.

LDW (4)

Load Word
Register Indexed

LDW Rn, @[Ai+Rj] / LDW @[An+Rm], Ri

Description

The LDW (Load Word Register Indexed) instruction is used to load a word from or to data memory at the location specified by the register Ai (or An) and the second register Rj (or Rm), which is an unsigned value.

15	14	13	12	11 ... 8	7	6 ... 4	3 ... 0
0	1	0	M	Rn or Ri	1	Ai or An	Rj or Rm

Operation

(M == 0, LDW Rn, @[Ai+Rj])
Rn := DM[(Ai+Rj]
(M == 1, LDW @[An+Rm], Ri)
DM[(An+Rm)] := Ri

Exceptions

None.

Notes

For memory transfer per word, the (byte) address needs to be aligned to be even. Thus, if (Ai + Rj) or (An + Rm) is an odd number, it will be made even by clearing the least significant bit.

LDW (5)

Load Word Register
Small Disp.

LDW An, @[Ai+<edisp:5>] / LDW @[Ai+<edisp:5>], An

Description

The LDW (Load Word Register Displacement) instruction is used to load 2 word from or to data memory at the location specified by the register Ai and a 5-bit even displacement from 0 to 30. <edisp:5> is encoded to 4-bit number by dropping the least significant bit.

15	14	13	12	11	8	7	6	4	3	0
0	1	1	M	1	An	0	Ai		<edisp:5>	

Operation

(M == 0, LDW An, @[Ai+<edisp:5>])

En := DM[(Ai + <edisp:5>)]

Rn := DM[(Ai + <edisp:5> + 2)]

(M == 1, LDW @[Ai+<edisp:5>], An)

DM[(Ai + <edisp:5>)] := En

DM[(Ai + <edisp:5> + 2)] := Rn

Exceptions

None.

Notes

For memory transfer per word, the (byte) address need to be aligned to be even. Thus, if (Ai + <edisp:5>) is an odd number, it will be made even by clearing the least significant bit. <edisp:5> can denote an even number from 0 to 30.

LDW (6)

Load Word
Register Disp.

LDW An, @[Ai+<disp:16>] / LDW @[Ai+<disp:16>], An

Description

The LDW (Load Word Register Large Displacement) instruction is used to load 2 word from or to data memory at the location specified by the register Ai and a 16-bit displacement.

15	14	13	12	11	8	7	6	5	4	3	2	0
1	0	1	0	1	An	0	0	1	1	M	Ai	

Operation

(M == 0, LDW An, @[Ai+<disp:16>])

En := DM[(Ai + <disp:16>)]

Rn := DM[(Ai + <disp:16> + 2)]

(M == 1, LDW @[Ai+<disp:16>], An)

DM[(Ai + <disp:16>)] := En

DM[(Ai + <disp:16> + 2)] := Rn

Exceptions

None.

Notes

This is a 2-word instruction, where the 16-bit immediate follows the instruction word shown above. Unlike 1-word instructions, therefore, fetching of this instruction takes 2 cycles. For memory transfer per word, the (byte) address need to be aligned to be even. Thus, if (Ai + <disp:16>) is an odd number, it will be made even by clearing the least significant bit.

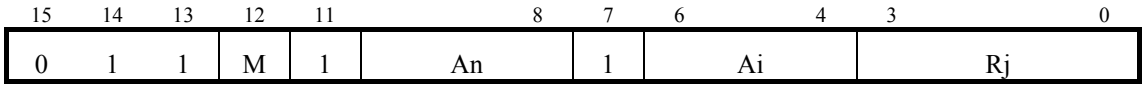
LDW (7)

Load Word
Register Indexed

LDW An, @[Ai+Rj] / LDW @[Ai+Rj], An

Description

The LDW (Load Word Register Indexed) instruction is used to load 2 word from or to data memory at the location specified by the register Ai and the second register Rj, which is an unsigned value.



Operation

(M == 0, LDW An, @[Ai + Rj])

En := DM[(Ai + Rj)]

Rn := DM[(Ai + Rj + 2)]

(M == 1, LDW @[Ai + Rj], An)

DM[(Ai + Rj)] := En

DM[(Ai + Rj + 2)] := Rn

Exceptions

None.

Notes

For memory transfer per word, the (byte) address needs to be aligned to be even. Thus, if (Ai + Rj) is an odd number, it will be made even by clearing the least significant bit.

MUL

Multiplication

MUL Mode, Dn, Di

Description

The instruction MUL performs 8x8 multiplication of the least significant byte of Dn and the least significant byte of Di. Dn and Di are registers from R0 to R7. The 16-bit multiplication result is written back to Dn. The mode is one of UU, US, SU, SS. The mode indicates each operand is signed value or unsigned value.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	M1		Dn		1	1	0	1	M2		Di	

Operation

```
if(M1 == 0 && M2 == 0)          // mode = UU
   Dn := lower 16 bits of ({0,Dn[7:0]} * {0, Di[7:0]})
else if(M1 == 0 && M2 == 1)     // mode == US
   Dn := lower 16 bits of ({0,Dn[7:0]} * {Di[7],Di[7:0]})
else if(M1 == 1 && M2 == 0)     // mode == SU
   Dn := lower 16 bits of ({Dn[7],Dn[7:0]} * {0,Di[7:0]})
else // mode == SS
   Dn := lower 16 bits of ({Dn[7],Dn[7:0]} * {Di[7],Di[7:0]})
```

Exceptions

None.

Notes

None.

NOP

No Operation

Description The NOP (No Operation) instruction does not perform any operation.

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

Operation None.

Exceptions None.

Notes None.

OR (1)

OR Register

OR Rn, Ri

Description

The OR (OR Register) instruction is used to perform bitwise OR operation on two values in registers, Rn and Ri.

The result is stored in register Rn. The T bit is updated based on the result.

15	14	13	12	11 ... 8	7	6	5	4	3 ... 0
1	0	0	0	Rn	0	1	0	1	Ri

Operation

Rn := Rn | Ri

T bit := ((Rn | Ri) == 0)

if(Rn == R6/R7) Z0/Z1 := ((Rn|Ri) == 0)

Exceptions

None.

Notes

None.

OR (2)

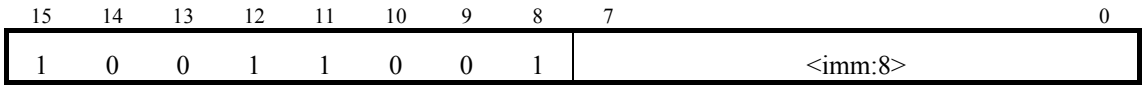
OR Small Immediate

OR R0, #<imm:8>

Description

The OR (OR Small Immediate) instruction is used to perform bitwise OR operation on two values in register R0 and <imm:8>.

The result is stored in register R0. The T bit is updated based on the result.



15	14	13	12	11	10	9	8	7 ... 0
1	0	0	1	1	0	0	1	<imm:8>

Operation

R0 := R0 | <imm:8>

T bit := ((R0 | <imm:8>)[7:0] == 0)

Exceptions

None.

Notes

The register used in this operation is fixed to R0. Therefore, the operand should be placed in R0 before this instruction executes. <imm:8> is zero-extended to a 16-bit value before operation.

OR (3)

OR Large Immediate

OR Rn, #<imm:16>

Description This type of OR instruction is used to perform bitwise OR operation on two values in register Rn and <imm:16>.

The result is stored in register R0. The T bit is updated based on the result.

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

Operation

Rn := Rn | <imm:16>

T bit := ((Rn | <imm:16>) == 0)

if(Rn == R6/R7) Z0/Z1 := ((Rn | <imm:16>) == 0)

Exceptions None.

Notes This is a 2-word instruction, where the 16-bit immediate follows the instruction word shown above. Unlike 1-word instructions, therefore, fetching of this instruction takes 2 cycles.

POP(1)

Load register from Stack

POP Rn, Rm / POP Rn

Description

The POP instruction load one or two 16-bit data from software stack to general registers. In the instruction of "POP Rn, Rm", there are some restrictions on Rn and Rm.

- Rn and Rm should not be R15.
- If Rn is one of the 8 registers from R0 to R7, Rm should also be one of them. If Rn is one of the registers from R8 to R14, Rm should also be one of them. For example, "POP R7, R8" is illegal.
- If Rn is the same as Rm, pop operation occurs only once. "POP Rn, Rn" is equivalent to "POP Rn".

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

Operation

```
if(Rn == Rm) {    // POP Rn
   Rn := DM[SP + 2]
   SP := SP + 2
} else {
   Rn := DM[SP + 2]
   Rm := DM[SP + 4]
   SP := SP + 4
}
```

Exceptions

None.

Notes

None.

POP(2)

Load register from Stack

POP An, Am / POP An

Description

The POP instruction load one or two 22-bit data from software stack to extended registers. In the instruction of "POP An, Am", there are some restrictions on An and Am.

- An and Am should not be A15.
- If An is the same as Am, pop operation occurs only once. "POP An, An" is equivalent to "POP An".

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

Operation

```
if(An == Am) {    // POP An
   En := lower 6 bits of DM[SP + 2]
   Rn := DM[SP + 4]
   SP := SP + 4
} else {
   En := lower 6 bits of DM[SP + 2]
   Rn := DM[SP + 4]
   Em := lower 6 bits of DM[SP + 6]
   Rm := DM[SP + 8]
   SP := SP + 8
}
```

Exceptions

None.

Notes

None.

PUSH(1)

Load register to Stack

PUSH Rn, Rm / PUSH Rn

Description

The PUSH instruction load one or two 16-bit data from general registers to software stack. In the instruction of "PUSH Rn, Rm", there are some restrictions on Rn and Rm.

- Rn and Rm should not be R15.
- If Rn is one of the 8 registers from R0 to R7, Rm should also be one of them. If Rn is one of the registers from R8 to R14, Rm should also be one of them. For example, "PUSH R7, R8" is illegal.
- If Rn is the same as Rm, push operation occurs only once. "PUSH Rn, Rn" is equivalent to "PUSH Rn".

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

Operation

```
if(Rn == Rm) {    // PUSH Rn
   DM[SP] := Rn
   SP := SP – 2
} else {
   DM[SP] := Rn
   DM[SP – 2] := Rm
   SP := SP – 4
}
```

Exceptions

None.

Notes

None.

PUSH(2)

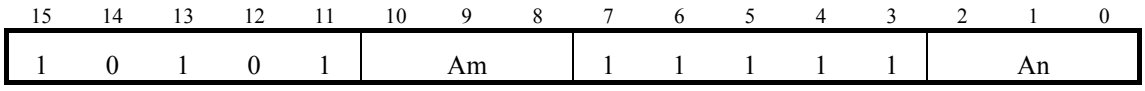
Load register to Stack

PUSH An, Am / PUSH An

Description

The PUSH instruction load one or two 22-bit data to software stack from extended registers. In the instruction of "PUSH An, Am", there are some restrictions on An and Am.

- An and Am should not be A15.
- If An is the same as Am, push operation occurs only once. "PUSH An, An" is equivalent to "PUSH An".



Operation

```
if(An == Am) {    // PUSH An
   DM[SP] := Rn
   DM[SP – 2] := {10'h000, En}
   SP := SP – 4
} else {
   DM[SP] := Rn
   DM[SP – 2] := {10'h000, En}
   DM[SP – 4] := Rm
   DM[SP – 6] := {10'h000, Em}
   SP := SP – 8
}
```

Exceptions

None.

Notes

None.

RETD

Ret. from Subroutine with Delay Slot

RETD

Description

The RETD (Return from Subroutine with Delay Slot) instruction is used to finish a subroutine and return by jumping to the address specified by the link register or A14. The difference between RETD and JMP A14 is that RETD has a delay slot, which allows efficient implementation of small subroutines.

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

Operation

PC := A14

Exceptions

None.

Notes

None.

RET_FIQ

Return from Fast Interrupt

RET_FIQ

Description

The RET_FIQ (Return from Fast Interrupt) instruction is used to finish a FIQ handler and resume the normal program execution. When this instruction is executed, SSR_FIQ (saved SR) is restored into SR, and the program control transfers to (SPCH_FIQ:SPCL_FIQ).

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

Operation

SR := SSR_FIQ

PC := (SPCH_FIQ:SPCL_FIQ)

Exceptions

None.

Notes

Fast Interrupt is requested through the core signal nFIQ. When the request is acknowledged, SR and current PC are saved in the designated registers (namely SSR_FIQ and SPCH_FIQ:SPCL_FIQ) assigned for FIQ processing. Such bits in SR as FE, IE, and TE are cleared, and PM is set.

RET_IRQ

Return from Interrupt

RET_IRQ

Description

The RET_IRQ (Return from Interrupt) instruction is used to finish an IRQ handler and resume the normal program execution. When this instruction is executed, SSR_IRQ (saved SR) is restored into SR, and the program control transfers to (SPCH_IRQ:SPCL_IRQ).

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

Operation

SR := SSR_IRQ

PC := (SPCH_IRQ:SPCL_IRQ)

Exceptions

None.

Notes

Interrupt is requested through the core signals nIRQ. When the request is acknowledged, SR and current PC are saved in the designated registers (namely SSR_IRQ and SPCH_FIQ:SPCL_IRQ) assigned for IRQ processing. Such bits in SR as IE and TE are cleared, and PM is set.

RET_SWI

Return from Software Interrupt

RET_SWI

Description

The RET_SWI (Return from Software Interrupt) instruction is used to finish a SWI handler and resume the normal program execution. When this instruction is executed, SSR_FIQ (saved SR) is restored into SR, and the program control transfers to the address A14 (link register).

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

Operation

SR := SSR_SWI

PC := A14

Exceptions

None.

Notes

Software interrupt is initiated by executing a SWI instruction from applications. When SWI instruction is executed, SR and current PC are saved in the designated registers (namely SSR_SWI and A14) assigned for SWI processing.

RL

Rotate Left

RL Rn

Description The RL (Rotate Left) instruction rotates the value of Rn left by one bit and stores the result back in Rn. T bit is updated as a result of this operation.

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

Operation Rn := Rn << 1, Rn[0] = MSB of Rn before rotation
T bit := MSB of Rn before rotation

Exceptions None.

Notes None.

RR

Rotate Right

RR Rn

Description The RR (Rotate Right) instruction rotates the value of Rn right by one bit and stores the result back in Rn. T bit is updated as a result of this operation.

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

Operation Rn := Rn >> 1, MSB of Rn = Rn[0] before rotation
T bit := Rn[0] before rotation

Exceptions None.

Notes None.

RRC

Rotate Right with Carry

RRC Rn

Description The RRC (Rotate Right with Carry) instruction rotates the value of (Rn:T bit) right by one bit and stores the result back in Rn. T bit is updated as a result of this operation.

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

Operation Rn := Rn >> 1, MSB of Rn = T bit before rotation
T bit := Rn[0] before rotation

Exceptions None.

Notes None.

SBC (1)

Subtract with Carry Register

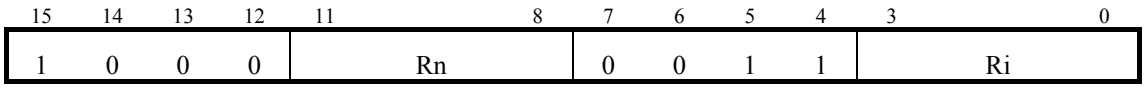
SBC Rn, Ri

Description

The SBC (Subtract with Carry) instruction is used to synthesize 32-bit subtraction. If register pairs R0, R1 and R2, R3 hold 32-bit values (R0 and R2 hold the least-significant word), the following instructions leave the 32-bit result in R0, R1:

```
SUB R0, R2
SBC R1, R3
```

SBC subtracts the value of register Ri, and the value of the Carry flag (stored in the T bit), from the value of register Rn, and stores the result in register Rn. The T bit and the V flag are updated based on the result.



Operation

Rn := Rn + ~Ri + T bit

T bit := Carry from (Rn + ~Ri + T bit)

V flag := Overflow from (Rn + ~Ri + T bit)

if(Rn == R6/R7) Z0/Z1 := ((Rn + ~Ri + T) == 0)

Exceptions

None.

Notes

None.

SBC (2)

Subtract with Carry Immediate

SBC Rn, #<imm:16>

Description

The SBC (Subtract with Carry immediate) instruction is used to synthesize 32-bit subtraction with an immediate operand. If register pair R0, R1 holds a 32-bit value (R0 holds the least-significant word), the following instructions leave the 32-bit subtraction result with 34157856h in R0, R1:

```
SUB R0, #7856h
SBC  R1, #3415h
```

SBC subtracts the value of <imm:16>, and the value of the Carry flag (stored in the T bit), from the value of Rn, and stores the result in register Rd. The T bit and the V flag are updated based on the result.

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

Operation

Rn := Rn + ~<imm:16> + T bit

T bit := Carry from (Rn + ~<imm:16> + T bit)

V flag := Overflow from (Rn + ~<imm:16> + T bit)

if(Rn == R6/R7) Z0/Z1 := ((Rn + ~<imm:16> + T) == 0)

Exceptions

None.

Notes

This is a 2-word instruction, where the 16-bit immediate follows the instruction word shown above. Unlike 1-word instructions, therefore, fetching of this instruction takes 2 cycles.

SETSR

Set SR

SETSR bs:3

Description The SETSR (Set SR) instruction is used to set a specified bit in SR as follows:

SETSR FE / IE / TE / V / Z0 / Z1 / PM

To set the T bit, one can do as follows:

CMP EQ, R0, R0

To clear a specified bit in SR, the CLRSR instruction (in page 45) is used.

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

Operation SR[<bs:3>] := 1

Exceptions None.

Notes None.

SR

Shift Right

SR Rn

Description The SR (Shift Right) instruction shifts the value of Rn right by one bit and stores the result back in Rn. T bit is updated as a result of this operation.

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

Operation Rn := Rn >> 1, with Rn[15] set to 0
T bit := Rn[0] before shifting

Exceptions None.

Notes None.

SRA

Shift Right Arithmetic

SRA Rn

Description

The SRA (Shift Right Arithmetic) instruction shifts the value of Rn right by one bit and stores the result back in Rn. While doing so, the original sign bit (most significant bit) is copied to the most significant bit of the result. T bit is updated as a result of this operation.

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

Operation

Rn := Rn >> 1, with Rn[15] set to the original value
T bit := Rn[0] before shifting

Exceptions

None.

Notes

None.

SRB

Shift Right Byte

SRB Rn

Description The SRB (Shift Right Byte) instruction shifts the value of Rn right by 8 bit and stores the result back in Rn. The high 8 bit positions are filled with 0's. T bit is updated as a result of this operation.

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

Operation Rn[7:0] := Rn[15:8] and Rn[15:8] := 8'h00
T bit := Rn[7] before shifting

Exceptions None.

Notes None.

SUB (1)

Subtract Register

SUB Rn, Ri

Description

The SUB (Subtract Register) instruction is used to subtract a 16-bit register value from another 16-bit register value. 32-bit subtraction can be achieved by executing SBC instruction in pair with this instruction (see page 98).

SUB subtracts the value of register Ri from the value of Rn, and stores the result in register Rn. The T bit and the V flag are updated based on the result.

15	14	13	12	11 … 8	7	6	5	4	3 … 0
1	0	0	0	Rn	0	0	0	1	Ri

Operation

Rn := Rn - Ri

T bit := Carry from (Rn - Ri)

V flag := Overflow from (Rn - Ri)

if(Rn == R6/R7) Z0/Z1 := ((Rn – Ri) == 0)

Exceptions

None.

Notes

None.

SUB (2)

Subtract Small Immediate

SUB Rn, #<imm:7>

Description

This form of SUB instruction is used to subtract a 7-bit immediate value from a register

It subtracts the value of <imm:7> from the value of register Rn, and stores the result in register Rn. The T bit and the V flag is updated based on the result.

15	14	13	12	11 ... 8	7	6 ... 0
0	0	0	0	Rn	1	<imm:7>

Operation

Rn := Rn - <imm:7>

T bit := Carry from (Rn - <imm:7>)

V flag := Overflow from (Rn - <imm:7>)

if(Rn == R6/R7) Z0/Z1 := ((Rn - <imm:7>) == 0)

Exceptions

None.

Notes

<imm:7> is an unsigned amount.

SUB (3)

Subtract

Extended Register

SUB An, Ri

Description This form of SUB instruction (Subtract Extended Register) is used to add a 16-bit unsigned register value from a 22-bit value in register.

This instruction subtracts the value of 16-bit register Ri from the value of 22-bit register An, and stores the result in register An.

15	14	13	12	11	10 … 8	7	6	5	4	3 … 0
1	0	1	0	1	An	1	1	0	0	Ri

Operation An := An - Ri

Exceptions None.

Notes None.

SUB (4)

Subtract Large Immediate

SUB An, #<imm:16>

Description

The SUB (Subtract Large Immediate) instruction is used to subtract a 16-bit unsigned immediate value from a 22-bit register.

SUB subtracts the value of <imm:16> from the value of An, and stores the result in register An.

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

Operation

An := An - <imm:16>

Exceptions

None.

Notes

This is a 2-word instruction, where the 16-bit immediate follows the instruction word shown above. Unlike 1-word instructions, therefore, fetching of this instruction takes 2 cycles.

SUB (5)

Subtract

5-bit Immediate

SUB An, #<imm:5>

Description

This form of SUB instruction (Subtract Extended Register) is used to subtract a 5-bit unsigned immediate value from a 22-bit register.

This instruction subtracts the value of 5-bit immediate <imm:5> from the value of 22-bit register An, and stores the result in register An.

15	14	13	12	11	10 … 8	7	6	5	4 … 0
1	0	1	0	1	An	0	1	1	<imm:5>

Operation

An := An - <imm:5>

Exceptions

None.

Notes

None.

SWI

Software Interrupt

SWI #<imm:6>

Description

The SWI (Software Interrupt) instruction performs a specified set of operations (i.e., an SWI handler). This instruction can be used as an interface to the low-level system software such as operating system.

Executing this instruction is similar to performing a function call. However, interrupts (IRQ and TRQ) will be masked off so that when a software interrupt is handled, it can be seen as an uninterruptible operation. Note that FIQ can still be triggered when an SWI is serviced. Return from a SWI handler is done by RET_SWI unlike normal function calls.

15	14	13	12	11	10	9	8	7	6	5	0
1	0	0	1	1	1	1	0	0	1	<imm:6>	

Operation

A14 := PC + 2

SSR_SWI := SR

IE := 0, TE := 0

PC := <imm:6> << 2

Exceptions

None.

Notes

Program addresses from 000000h to 0000feh are reserved for SWI handlers. SWI vectors 0 and 1 are not used, as the addresses from 000000h to 000007h are reserved for other interrupts.

SYS

System

SYS #<imm:5>

Description The SYS (System) instruction is used for system peripheral interfacing using DA[4:0] and nSYSID core signals.

15	14	13	12	11	10	9	8	7	6	5	4	0
1	0	0	1	1	1	1	0	0	0	1	<imm:5>	

Operation core output signal DA[4:0] := <imm:5>, DA[21:5] := (unchanged)
core output signal nSYSID := LOW

Exceptions None.

Notes None.

TST (1)

Test Register

TST Rn, Ri

Description

The TST (TST Register) instruction is used to determine if many bits of a register are all clear, or if at least one bit of a register is set.

TST performs a comparison by logically ANDing the value of register Rn with the value of Ri. T bit is set according to the result.

15	14	13	12	11 … 8	7	6	5	4	3 … 0
1	0	0	0	Rn	0	1	1	1	Ri

Operation

Temp := Rn & Ri

T bit := ((Rn & Ri) == 0)

Exceptions

None.

Notes

None.

TST (2)

Test Small Immediate

TST R0, #<imm:8>

Description This type of TST instruction is used to determine if many bits of a register are all clear, or if at least one bit of a register is set.

TST performs a comparison by logically ANDing the value of register Rn with the value of Ri. T bit is set according to the result.

15	14	13	12	11	10	9	8	7 ... 0
1	0	0	1	1	0	1	1	<imm:8>

Operation

Temp n := Rn & <imm:8>

T bit := ((Rn & <imm:8>)[7:0] == 0)

Exceptions None.

Notes The register used in this operation is fixed to R0. Therefore, the operand should be placed in R0 before this instruction executes.

TST (3)

Test Large Immediate

TST Rn, #<imm:16>

Description This type of TST instruction is used to determine if many bits of a register are all clear, or if at least one bit of a register is set.

TST performs a comparison by logically ANDing the value of register Rn with the value of Ri. T bit is set according to the result.

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

Operation

Temp := Rn & <imm:16>

T bit := ((Rn & <imm:16>) == 0)

Exceptions None.

Notes This is a 2-word instruction, where the 16-bit immediate follows the instruction word shown above. Unlike 1-word instructions, therefore, fetching of this instruction takes 2 cycles.

TSTSR

Test SR

TSTSR bs:3

Description The TSTSR (Test SR) instruction is used to test a specified bit in SR as the following example shows:

TST FE / IE / TE / V / Z0 / Z1 / PM

To set or clear a specified bit, the SETSR (in page 100) or CLRSR (in page 45) instruction is used.

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

Operation T bit := ~SR[<bs:3>]

Exceptions None.

Notes None.

XOR (1)

XOR Register

XOR Rn, Ri

Description

The XOR (XOR Register) instruction is used to perform bitwise XOR operation on two values in registers, Rn and Ri.

The result is stored in register Rn. The T bit is updated based on the result.

15	14	13	12	11 ... 8	7	6	5	4	3 ... 0
1	0	0	0	Rn	0	1	1	0	Ri

Operation

Rn = Rn ^ Ri

T bit = ((Rn ^ Ri) == 0)

if(Rn == R6/R7) Z0/Z1 := ((Rn^Ri) == 0)

Exceptions

None.

Notes

None.

XOR (2)

XOR Small Immediate

XOR R0, #<imm:8>

Description

This type of XOR instruction is used to perform bitwise XOR operation on two values in register R0 and <imm:8>.

The result is stored in register R0. The T bit is updated based on the result.

15	14	13	12	11	10	9	8	7 ... 0
1	0	0	1	1	0	1	0	<imm:8>

Operation

R0 = R0 ^ <imm:8>

T bit = ((R0 ^ <imm:8>)[7:0] == 0)

Exceptions

None.

Notes

The register used in this operation is fixed to R0. Therefore, the operand should be placed in R0 before this instruction executes. <imm:8> is zero-extended to a 16-bit value before operation.

XOR (3)

XOR Large Immediate

XOR Rn, #<imm:16>

Description

This type of XOR instruction is used to perform bitwise XOR operation on two values in register Rn and <imm:16>.

The result is stored in register Rn. The T bit is updated based on the result.

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

Operation

Rn = Rn ^ <imm:16>

T bit = ((Rn ^ <imm:16>) == 0)

if(Rn == R6/R7) Z0/Z1 := ((Rn^<imm:16>) == 0)

Exceptions

None.

Notes

This is a 2-word instruction, where the 16-bit immediate follows the instruction word shown above. Unlike 1-word instructions, therefore, fetching of this instruction takes 2 cycles.

5 CLOCK & POWER MANAGEMENT

OVERVIEW

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

The clock control logic in S5L840F generates various system clock signals: HCLK_CPU for CPU, HCLK for the AHB bus peripherals and PCLK for the APB bus peripherals. 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, S5L840F 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 S5L840F. 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 S5L840F 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 S3C2400, which is uA unit. The wake-up from STOP mode can be done by activating external interrupt pins.

The reset controller in S5H5002 consists of three reset types: hardware reset, software reset and watchdog reset. These types of reset are described in detail on page 6-9 *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 6-1 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 S5L840F. 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 6-1 lists the maximum operating frequencies for the S5L840F. When selecting strap settings, make sure that the bus divider ratios do not result in the bus frequencies that exceed these maximums.

Table 6-1. Maximum Bus Frequencies

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

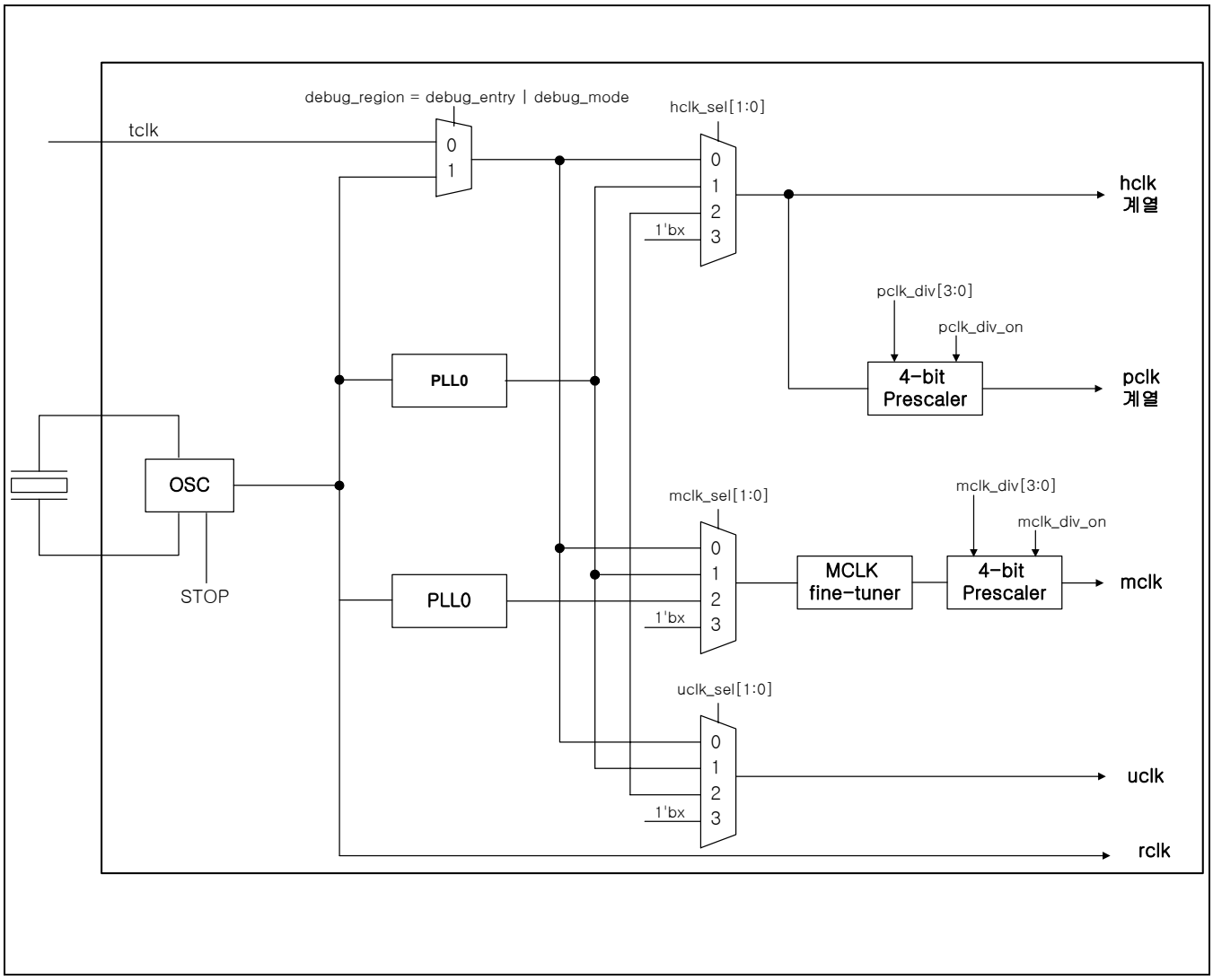


Figure 6-1. Clock Generator Block Diagram

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 Fpllo is related to the reference input clock frequency Fin by the following equation:

$$Fpllo = Fin * (m + 1) / 2^{(s+1)}$$

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

Change PLL Settings in Normal Operation Mode

During the operation of S5L840F 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 S5L840F. The timing diagram is as follow.

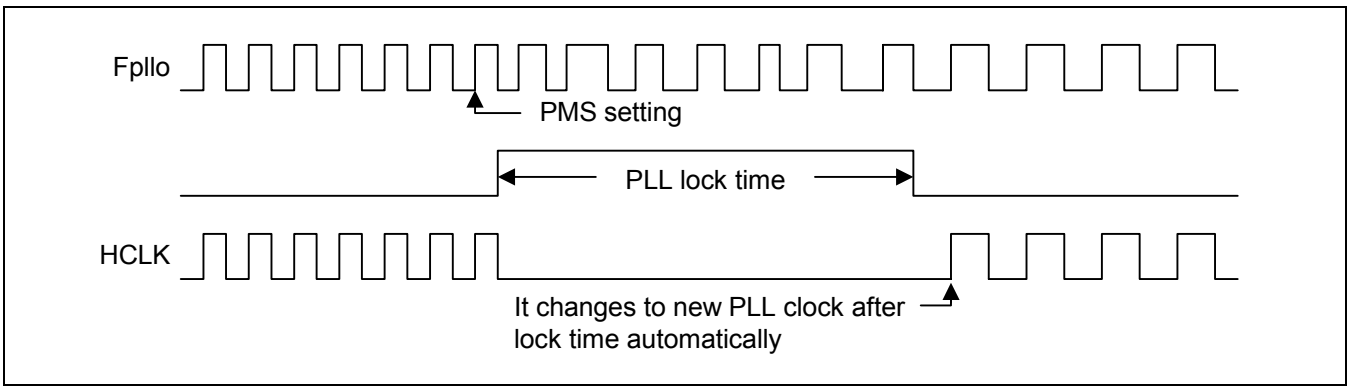


Figure 6-2. Timing Diagram of Clock Change in NORMAL Mode

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 it's own clock to be masked to enter IDLE mode which is one of the power saveing mode of S5L840F and can be woken up by interrupt. Including IDLE power saving mode S5L840F 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_cpu in the S5L840F 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 minimun power. And all other pheriphrals 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 it's 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 guerantee 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 S5L9260X maintains statically it's 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 S5L840F 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 pclk == 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 excutes 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 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 stablized. All clocks are disbled until *wdt_start* is received . (except the clock for the WDT of cource 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. (The CPU changes it ?)

Global Power Management :

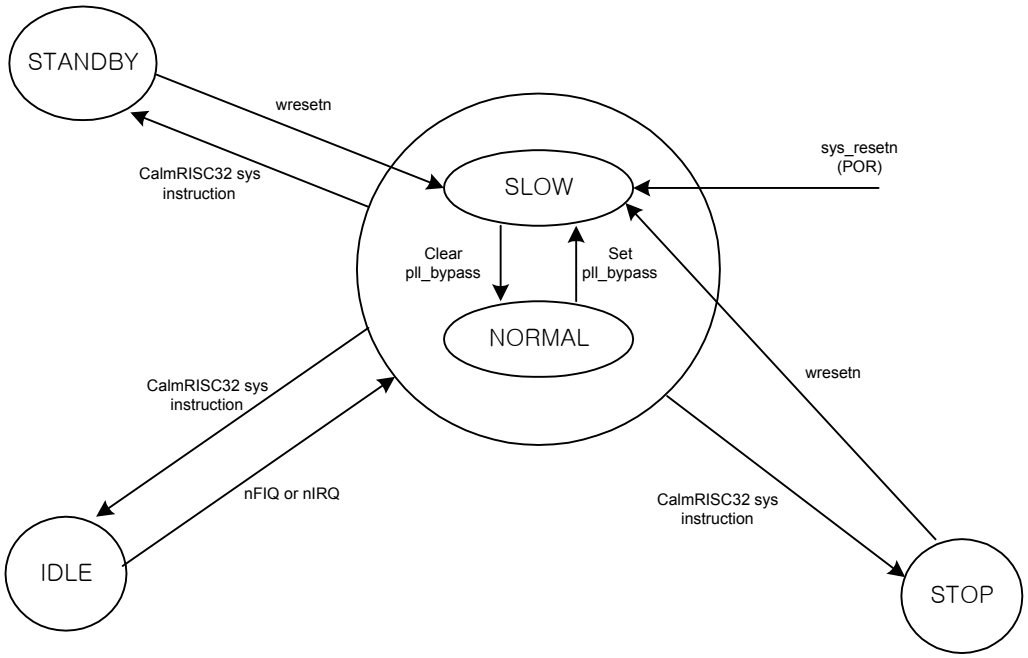


Figure 6-5. The Global Power management

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

S5L840F does not provide a separate power and ground for RTC.

2. USB power management :

The USB don't have to operate when it's not connected to USB cable. Therefore the USB clocks can be masked after boot by CPU when it's not used..

According to USB standard USB device goes to Suspend state when there is no USB Bus activity during more than 2.5us. S5L840F USB Core sets D1 bit to "1" when it comes to Suspend state.

USB Power management Register

Name	bit	CPU (ClamRISC32)	USB Core	Description
SUSPEND_MODE	D1	R	R/W	This bit is set by the USB when it enters suspend mode
UC_RESUME	D2	R/W	R	The MCU sets this bit for a duration of 10ms to initiate a resume signaling
USB_RESET	D3	R	Set	The USB sets this bit if reset signaling is received from the host

When USB cable is disconnected or there is no USB signal detected USB goes to suspend state. The CPU should check D1 bit periodically to recognize that USB goes to suspend state. Then CPU controls to disable both of USB master clock and USB 48Mhz clock.

During the USB is in it's Suspend state, when USB signal from host is detected USB core generates interrupt. This interrupt is generated even when there is no clock supplied to USB core. The CPU recognizes the USB connection by the interrupt, enables USB clocks and sets the bit D2 of the USB power management register to "1". Then the USB changes it's state to Resume state.

3. 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. The following is a senario about how a card can be detected automatically in it's slot. This is based on guess!

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 (or GPIO ??) 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.)

4. SD Card and Multi-Media Card Power Management :

THE CPU POLLS THE SC CARD INTERFACE TO IDENTIFY IF A CARD IS INSTALLED OR NOT. IF NOT INSTALLED IT MASKS THE CLOCK FOR THE SC CARD INTERFACE. WHEN THE CPU DOES NOT WANT TO ACCESS THE SC CARD IT MAY MASK THE CLOCK FOR THE SC CARD INTERFACE. (IS THIS CORRECT ??)

RESET CONTROLLER

The reset controller manages the various reset sources in the S5L840F. 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 S5L840F are described below:

Hardware Reset

Hardware reset is invoked when the nRESET pin is asserted, and all units in the S5H5002 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 S5L840F 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 WTCON[7:0] are set and the watchdog timer counter (WTCNT) overflows. The reset sequence of watchdog initiated reset is identical to software reset. When the WTCNT overflows, the S5L840F 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)

Fpllo = Fin * (m+1) / $2^{(s+1)}$

Table 6-3. Recommended Value of MDIV, SDIV

M	S	Fvco [MHz]	Fout [MHz]
5859	0	192.02048	96.01024
5859	1	192.02048	48.00512
5166	0	169.312256	84.656128
5166	1	169.312256	42.328064
5166	2	169.312256	21.164032
5511	0	180.617216	90.308608
5511	1	180.617216	45.154304
5511	2	180.617216	22.577152
5511	3	180.617216	11.288576
5512	0	180.649984	90.324992
5624	0	184.32	92.16
5624	1	184.32	46.08
5624	2	184.32	23.04
5999	0	196.608	98.304
5999	1	196.608	49.152
5999	2	196.608	24.576
5999	3	196.608	12.288
6201	0	203.227136	101.613568
6749	0	221.184	110.592

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

values are all zeros.

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
-	[31:27]	Reserved	0
UCLK_MASK	[26]	0 = UCLK enable 1 = UCLK disable	0
RCLK_MASK	[25]	0 = RCLK enable 1 = RCLK disable	0
MCLK_MASK	[24]	0 = MCLK enable 1 = MCLK disable	0
-	[23]	Reserved	0
PCLK_DIVON	[22]	0 = prescaler off 1 = prescaler on	0
UCLK_DIVON	[21]	0 = prescaler off 1 = prescaler on	0
MCLK_DIVON	[20]	0 = prescaler off 1 = prescaler on	0
-	[19:18]	Reserved	00
HCLK_SEL	[17:16]	00 = OSC 01 = PLL0 10 = PLL1 11 = Not Used	00
UCLK_SEL	[15:14]	00 = OSC 01 = PLL0 10 = PLL1 11 = Not Used	00
MCLK_SEL	[13:12]	00 = OSC 01 = PLL0 10 = PLL1 11 = Not Used	00
-	[11:8]	Reserved	0000
UCLK_DIV	[7:4]	4-bit prescaler value UCLK = input clock / (UCLK_DIV + 1) when UCLK_DIVON == 1	0000
MCLK_DIV	[3:0]	4-bit prescaler value MCLK = input clock / (MCLK_DIV + 1) when MCLK_DIVON == 1	0000

PLL PMS VALUE REGISTER (PLLPMS)

Register	Address	R/W	Description	Reset Value
PLL0PMS	0x3C50 0004	R/W	PLL PMS value Register	Undefined
PLL1PMS	0x3C50 0008	R/W	PLL PMS value Register	Undefined

PLLPMS	Bit	Description	Initial State
Reserved	[31:30]	Reserved	
MDIV	[29:16]	Main divider control	Undefined
Reserved	[15:2]	Reserved	
SDIV	[1:0]	Post-divider control	Undefined

PLL LOCK COUNT REGISTER (PLLLCNT)

Register	Address	R/W	Description	Reset Value
PLL0LCNT	0x3C50_0014	R/W	PLL0 lock count register	0x0000 1FFF
PLL1LCNT	0x3C50_0018	R/W	PLL1 lock count register	0x0000 1FFF

PLLLCNT	Bit	Description	Initial State
Reserved	[31:13]	Reserved	
LOCK_CNT	[12:0]	PLL lock count value (down counter)	0x1FFF

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	0

PLLLOCK	Bit	Description	Initial State
Reserved	[31:2]	Reserved	
PLL1_LOCK	[1]	PLL1 Lock Status 0 : Progress 1 : Locking done	0
PLL0_LOCK	[0]	PLL0 Lock Status 0 : Progress 1 : Locking done	0

PLL CONTROL REGISTER (PLLCON)

Register	Address	R/W	Description	Reset Value
PLLCON	0x3C50 0024	R/W	PLL control register	0

PLLCON	Bit	Description	Initial State
Reserved	[31:2]	Reserved	
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

CLOCK POWER CONTROL REGISTER (PWRCON)

Register	Address	R/W	Description	Reset Value
PWRCON	0x3C50_0028	R/W	Clock power control register	0x0020 4000

PWRCON	Bit	Description	Initial State
-	[31:15]	Reserved	0000
CLK14 : GPIO	[14]	0 = Disable 1 = Enable	1
CLK13 : TIMER	[13]	0 = Disable 1 = Enable	0
CLK12 : ADC IF	[12]	0 = Disable 1 = Enable	0
CLK11 : LCD IF	[11]	0 = Disable 1 = Enable	0
CLK10 : RTC	[10]	0 = Disable 1 = Enable	0
CLK9 : IIS	[9]	0 = Disable 1 = Enable	0
CLK8 : SPDIF	[8]	0 = Disable 1 = Enable	0
CLK7 : IIC	[7]	0 = Disable 1 = Enable	0
CLK6 : SPI	[6]	0 = Disable 1 = Enable	0
CLK5 : UART	[5]	0 = Disable 1 = Enable	0
CLK4 : SDC/MMC	[4]	0 = Disable 1 = Enable	0
CLK3 : Memory Stick	[3]	0 = Disable 1 = Enable	0
CLK2 : SMC	[2]	0 = Disable 1 = Enable	0
CLK1 : USB	[1]	0 = Disable 1 = Enable	0
CLK0 : APBIF	[0]	0 = Disable 1 = Enable	0

SOFTWARE RESET CONTROL REGISTER (SWRCON)

The software reset control register has a software reset bit, which when set, causes a reset of the S5L840F. The software-reset bit (SWR) is located within the least significant bit of the write-only software reset register (SWRCON). Writing a one to this bit causes all on-chip resources to reset but does not cause the PLL to go out of lock. The software reset bit is self-clearing. It is automatically cleared to zero after a few system clock cycles once it is set. Writing zero to the software reset bit has no effect. Care should be taken to restrict access to this register by programming MMU permissions.

The following table shows the SWRCON.

Register	Address	R/W	Description	Reset Value
SWRCON	0x3C50_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.	0

RESET STATUS REGISTER (RSTSR)

To determine the last cause or causes of the reset, the CPU can refer to the reset status register (RSTSR). The S5L840F has three sources of reset:

- Hardware reset
- Software reset
- Watchdog reset

Each RSTSR status bit is set by a different source of reset, and can be cleared by setting a one of the other reset status bits. Note that the hardware reset state of software and watchdog reset bit is zero.

The table below shows the status bits within RSTSR.

Register	Address	R/W	Description	Reset Value
RSTSR	0x3C50_0034	R/W	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

CalmADM3-0.2-030715

CalmADM3

CalmRISC16 and CalmMAC24 Based Audio DSP Module with Host Capability

Functional Specification 1.0

Ver-030715

JoongEon Lee, Optical Player P/J

REVISION HISTORY

Verion	Engineer	Date	Description
0.0	J.E.Lee	2003-05-31	First version. Copied & modified from CalmADM2_spec_v01.
0.1	J.E.Lee	2003-06-11	Major changes from the last version: - added nFIQ input pin - changed fast interrupt generation mechanism - added chapter 5, “Information for CalmShine Development”
1.0	J.E.Lee	2003-07-15	Major changes from the last version: - added SYS_IDLE command - included BIST modules - added RIACE flag in FIECFG of HFRS - added DBU output ports - fixed some errata Green colored characters denote changes.

TABLE OF CONTENTS

1 Product Overview

INTRODUCTION

CalmADM3, a CalmRISC16 and CalmMAC24 based audio DSP module with host capability, is designed for high-quality audio processing and micro system control. It includes Samsung's 16-bit MCU, CalmRISC16, and 24-bit DSP, CalmMAC24. CalmADM3 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, CalmADM3 includes two sequential buffers to handle input/output data efficiently. These sequential buffers can be act as a kind of ring buffer also. Since CalmADM3 is based on a 16-bit MCU, CalmRISC16, but its target system bus, AMBA, is a 32-bit bus, CalmADM3 cannot access slave registers on the system bus in usual way. In CalmADM3, a specially designed slave register interface unit handles the 16-bit/32-bit data width conversion. In this document, we call CalmADM3 as an abbreviation, ADM.

Features

CalmRISC16F

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
'F' denotes full scan version of CalmRISC16
In this document, we call CalmRISC16F as an abbreviation, Calm.

CalmMAC24F

24-bit (audword) high performance fixed-point DSP coprocessor for CalmRISC16 micro-controller
1 cycle 24×24 MAC operation
32K audword X data memory space & 32K audword Y data memory space
2 multiplier accumulator registers, 4 general accumulator registers, and 8 pointer registers
'F' denotes full scan version of CalmMAC24
In this document, we call CalmMAC24F as an abbreviation, Mac.

**NOTE: WE DEFINED 24-BIT DATA UNIT OF CALMMAC24F AS aud-word IN THIS DOCUMENT.*

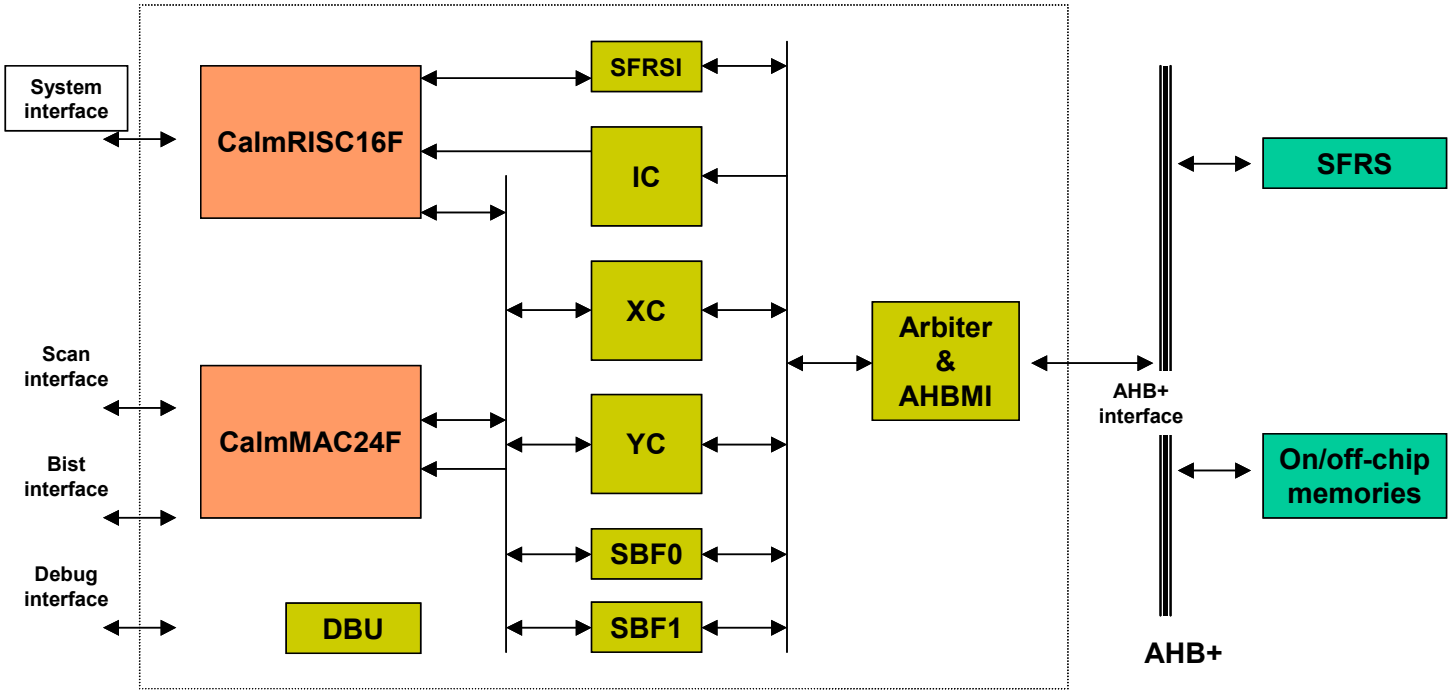


Figure 1-1. CalmADM3 Block Diagram

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

AHB+ INTERFACE

CalmADM3 acts as an AHB+ master
to access on/off-chip memories including memory mapped slave registers.

SFRS INTERFACE

Converts 16-bit dual-access from Calm to 32-bit bus access
one-entry cache with auto-invalidation and auto-backup

Clock

One AHB+ clock input

PERFORMANCE

Max. operating Frequency = 100MHz
@ (Voltage: 1.65V – 1.95V, Temperature: -40 °C ~ 125 °C, Process: Samsung L18

2 CalmADM3 Programming Model

2-1 memory configuration

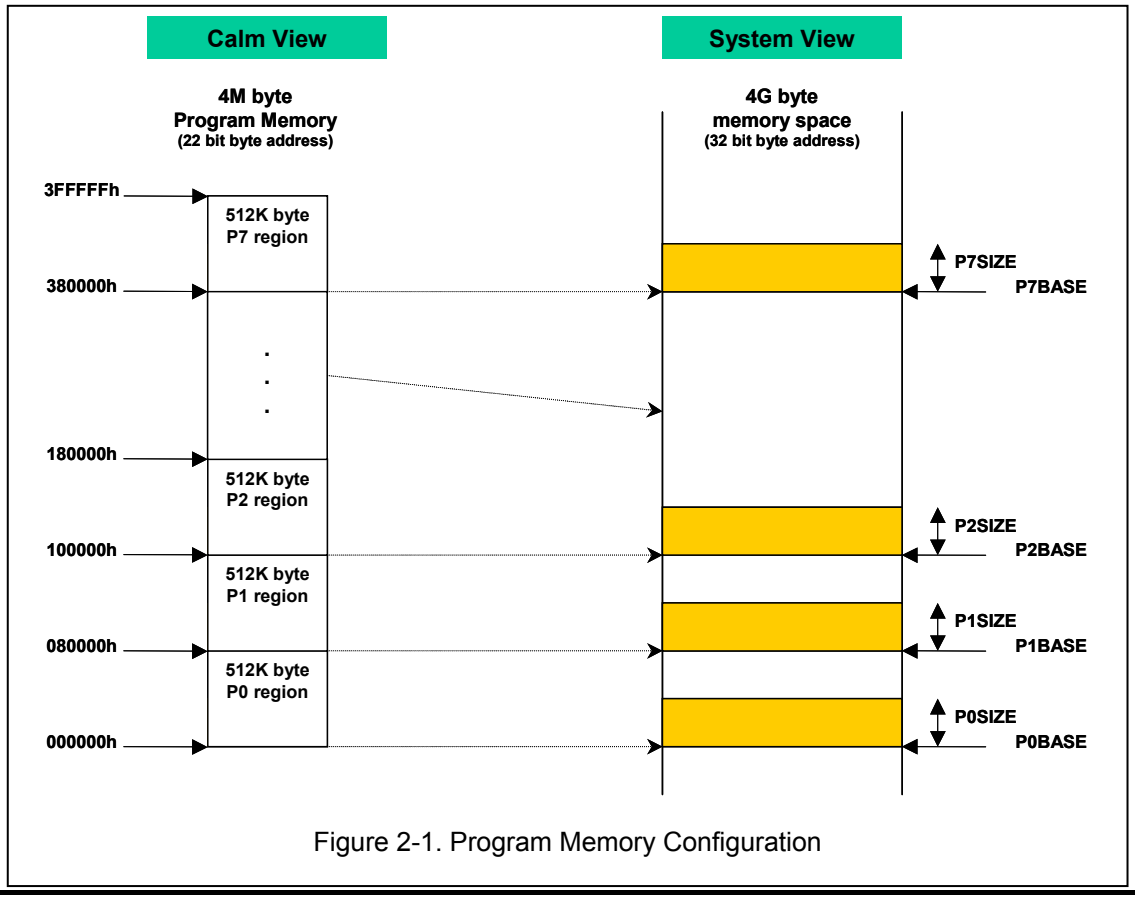


Figure 2-1. Program Memory Configuration

2-1-1 Program Memory

Program memory configuration is shown in Figure 2-1. 4M-byte cacheable instruction memory space is divided into 8 regions. Calm accesses these regions through an instruction cache, I-Cache. Each region is individually mapped to 4G-byte system memory space as defined by its BASE and SIZE fields of its configuration register in HFRS. For example, if [20:18] bits of PA, the 21-bit program address bus from Calm, equals “000b”, it is in P0 region in Calm view. PA[17:0] is converted as a byte address (multiplied by 2), added with P0BASE and mapped to 32-bit system memory space. Before this address mapping, PA[17:0] is checked whether it is addressing the location beyond size limit defined by P0SIZE.

2-1-2 Data Memory

Data memory configuration is shown in Figure 2-2. 4M-byte data memory space is divided into 8 memory regions and 2 register areas. Each region is individually mapped to 4G-byte system memory space as defined by its BASE and SIZE fields of its configuration register in HFRS.

Calm Regions

Lower 2M-byte area of 4M-byte data memory space is divided into 4 regions, accessed by Calm only.

Calm accesses these regions through two data caches (X-Cache and Y-Cache). As the cache for Calm regions,

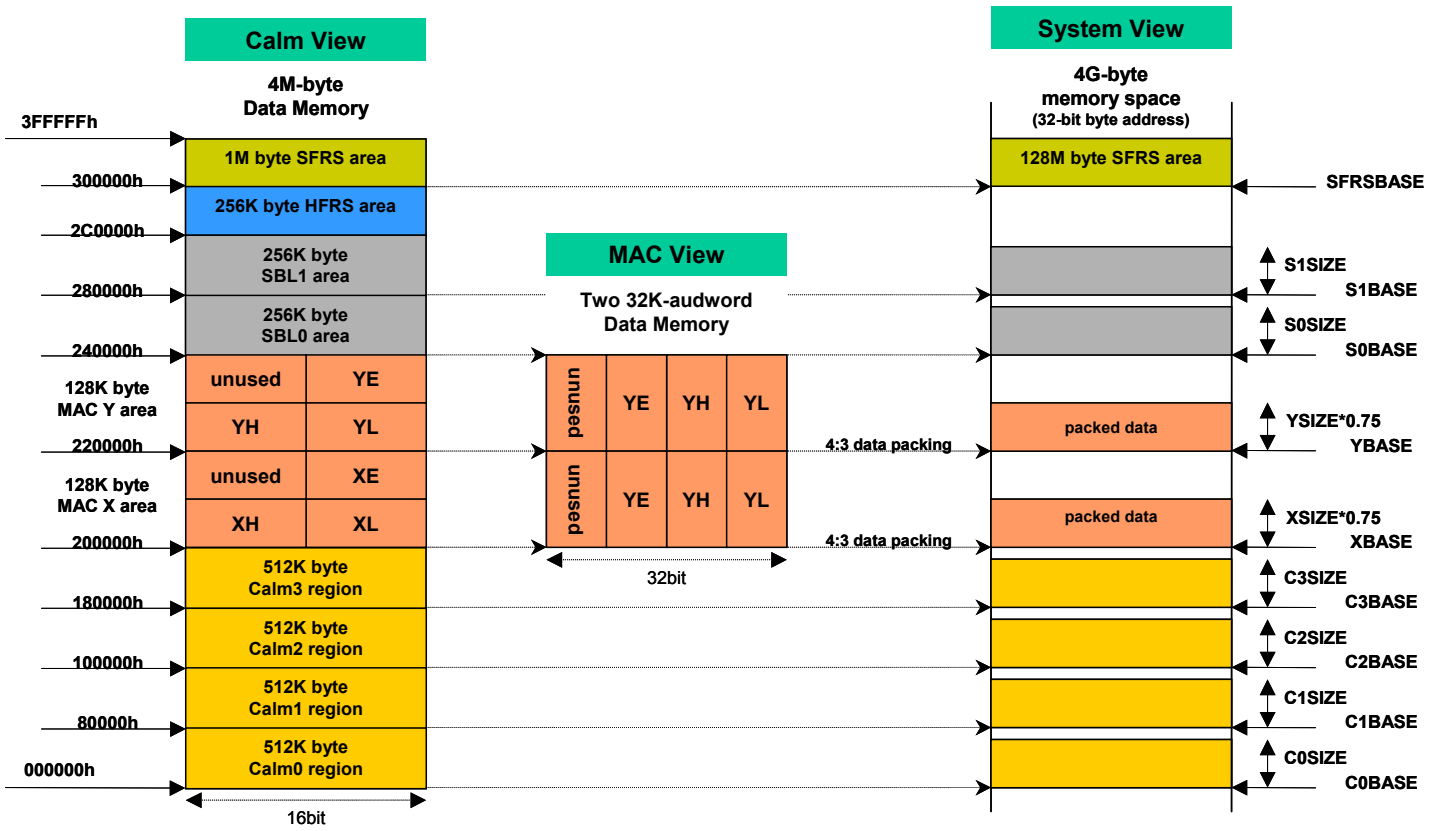


Figure 2-2. Data Memory Configuration

these two data caches operate similar to a two way set associative cache.

If [21:19] bits of DA, the 22-bit data address bus from Calm, equals “000b”, it is in C0 region in Calm view. DA[18:0] is added with C0BASE and mapped to 32-bit system memory space. Before this address mapping DA[18:0] is checked whether it is addressing the location beyond size limit defined by C0SIZE.

MAC Regions

MAC X and MAC Y are dual memory for dual load capability of Mac. Each is 32K-audword. Both of Calm and Mac can access these two regions.

When DA[21:17] equals “10000b”, Calm accesses X region. To access this region, Calm uses X-Cache as data cache. DA[16:0] from Calm is converted as X region address before get into X-Cache. When DA[21:17] is 10001b, Calm accesses Y region. To access this region, Calm uses Y-Cache as data cache. DA[16:0] from Calm is converted as Y region address before get into Y-Cache. Mac accesses two MAC areas with its two 15-bit audword data addresses, XA and YA. XA and YA are 2-bit left shifted (conversion to byte address) before get into the caches. The 6K-byte X-Cache covers 32K-audword MAC X region. The 6K-byte Y-Cache covers 32K-audword MAC Y region.

The addresses from X-Cache and Y-Cache are converted as 4:3 reduction, added with XBASE/YBASE and mapped to 32-bit system memory space. Before this address mapping the addresses are checked whether they are addressing the location beyond size limit defined by XSIZE/YSIZE.
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 REGIONS

Sequential block regions are used mainly for input and output of audio stream data frames.

Calm can access these regions randomly but Mac can access them in sequential way only. Since these regions are out of Mac’s memory space, Mac cannot access these randomly. In ADM, a special logic was added so that Mac can access these regions sequentially.

When DA[21:18] is 1001b, Calm accesses SBL0 area. DA[17:0] is added with S0BASE and mapped to 32-bit system memory address. Before this address mapping DA[17:0] is checked whether it is addressing the location beyond size limit defined by S0SIZE. When DA[21:18] is 1010b, Calm accesses SBL1 area similarly to SBL0 area. Since ADM has no caches for these areas, Calm accesses system memory directly when it accesses these areas. When XA or YA are 7FFEh, Mac accesses SBL0 area sequentially. When XA or YA are 7FFFh, Mac accesses SBL1 area sequentially. The sequential accesses are modulated with the size limit. Therefore, sequential accesses do not cause the size limit violation. More of sequential data accessing is described in later of this document.

HFRS AREA

HFRS (Host Function Register Set) area is reserved for memory-mapped registers of ADM. HFRS area resides 2C0000h to 2FFFFFh address space in Calm’s data memory space. Registers in HFRS can be accessed by ADM only. Outside of ADM cannot access these.

SFRS Areas

SFRS (Special Function Register Set) area is reserved for system level memory-mapped registers. SFRS area is divided into two areas, PSFRS area for slave registers in APB IPs and HSFRS area for slave registers in AHB IPs.

PSFRS area resides 380000h to 3FFFFFh address space in Calm’s data memory space. 512K-byte PSFRS area is divided into 64 8K-byte blocks and mapped to 64M-byte system memory space. ADM can handle up to 64 APB

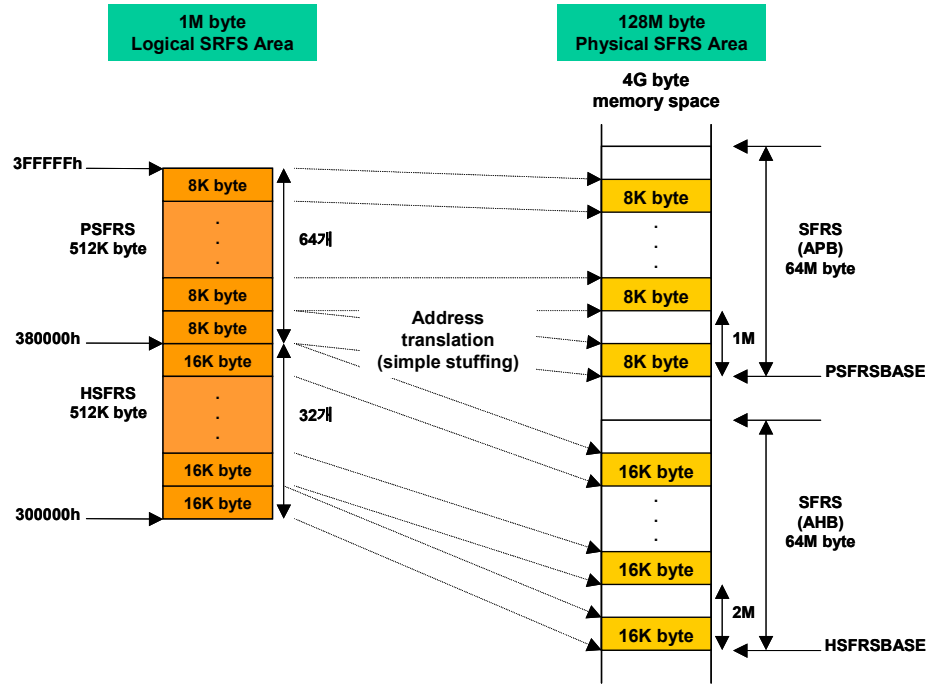


Figure 2-3. SFRS Memory Map

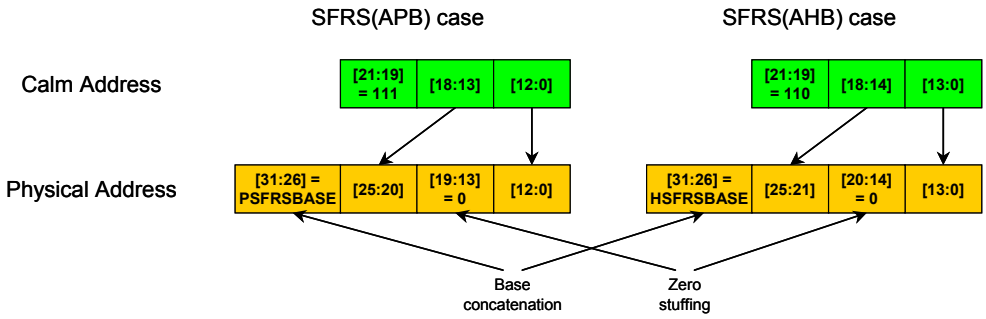


Figure 2-4. SFRS Address Translation

IPs that have up to 8K-byte slave register set.
HSFRS area resides 300000h to 37FFFFh address space in Calm's data memory space. 512K-byte HSFRS area is divided into 32 16K-byte blocks and mapped to 64M-byte system memory space. ADM can handle up to 32 AHB IPs that have up to 16K-byte slave register set. Memory map and address translation of SFRS are shown in Fig.2-3 and Fig.2-4.

2-1-3 MEMORY REGIONS

In ADM are 16 memory regions. The regions are assigned as Fig.2-5.

Each region has corresponding region control registers and flag in HFRS. The region control is performed by hardware in ADM based on a region enable flag of RECFG register and base address field and region size field of region configuration register in HFRS. Because the base address field is 24-bit corresponding to [31:8] bit location of 32-bit physical memory address, each region can be aligned 256-byte step boundaries in physical memory

R15	R14	R13	R12	R11:R8	R7:R0
S1	S0	Y	X	C3:C0	P7:P0

Figure 2-5. Region Assignment

SIZE Code	Size			
	R0~R11	R12,R13 (X,Y)		R14,R15 (S0,S1)
		logical	physical	
0000	1KB	1KB	0.75KB	1KB
0001	2KB	2KB	1.5KB	2KB
0010	4KB	4KB	3KB	4KB
0011	8KB	8KB	6KB	8KB
0100	16KB	16KB	12KB	16KB
0101	32KB	32KB	24KB	32KB
0110	64KB	64KB	48KB	64KB
0111	128KB	128KB	96KB	128KB
1000	256KB	128KB	96KB	256KB
1001	512KB	128KB	96KB	256KB
others	512KB	128KB	96KB	256KB

Figure 2-6. Region Size Definitions

map. The SIZE fields are defined as Fig.2-6.

When Calm or Mac issues a memory access, ADM checks if the accessed region is enabled and if the accessed location is in valid region area defined by SIZE field. If one of these two checks results false, ADM issues an exception to Calm instead of issuing the physical memory access. Before ADM issues the physical memory access, it performs an address translation by adding BASE field to the logical address issued from processors.

2-1-4 MAC AREA CONSIDERATIONS

**NOTE: THIS SECTION IS EXACTLY SAME AS THE ONE IN CALMADM2 SPECIFICATION. IF YOU ARE FAMILIAR WITH CALMADM2, YOU DON'T NEED TO READ THIS SECTION.*

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 structure difference of MAC X/Y area between Calm View and Mac View in Figure2-2) Because of the difference, 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 system 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 Reduction for Unused Bytes

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

2-2 Sequential Data Access

**NOTE: THIS SECTION IS EXACTLY SAME AS THE ONE IN CALMADM2 SPECIFICATION. IF YOU ARE FAMILIAR WITH CALMADM2, YOU DON'T NEED TO READ THIS SECTION.*

If a data block in the system memory is well-aligned and well-known stream data, 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.

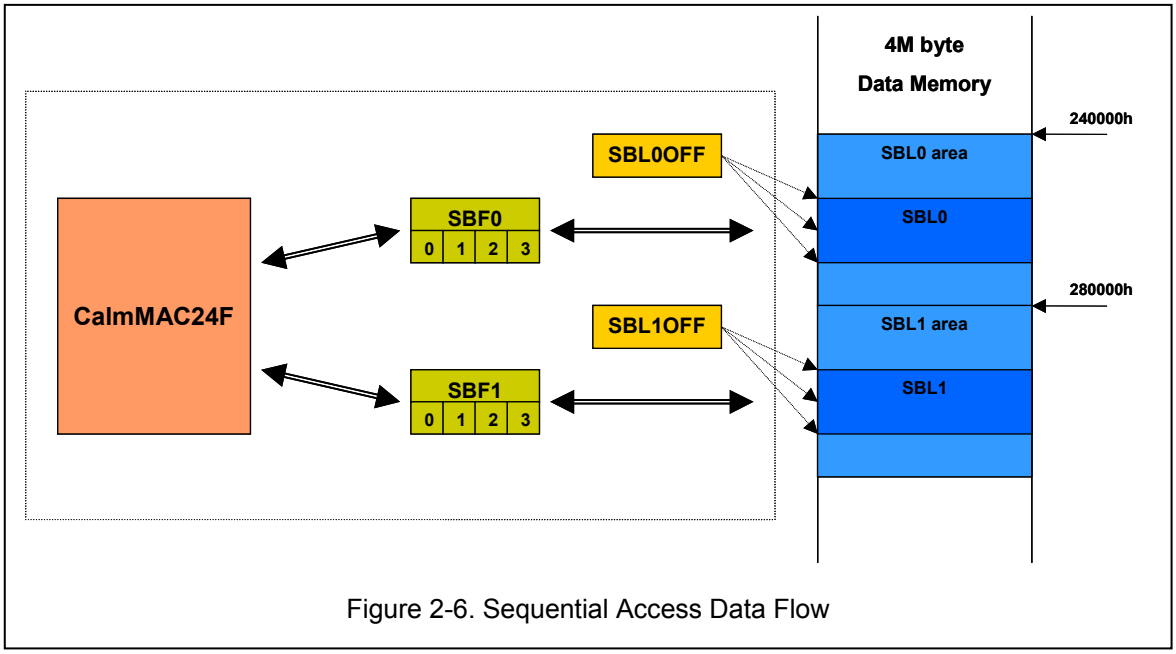


Figure 2-6. Sequential Access Data Flow

2-2-1 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.
- Unit data in a sequential block can be word or half-word. However, a sequential block should be word aligned for efficient system memory access.
- Data in a sequential block are addressed with offset register.
- These blocks are dynamic blocks that can be created, used and removed 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.

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 giving special command into control registers.
- 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, incremented offset register is compared with the end offset register. If two register values are same, an offset register is set as the value of the 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 the 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 that contain 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 not used.
- 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 the offset register is increase to meet the value of end offset register, two operations are automatically performed by hardware.
  1. offset wrapping
     The offset register is set as the value of begin offset register. (we call it ‘wrapping’ in this document)
  2. interrupt
     Corresponding interrupt flag is set and interrupt is forced to Calm if the interrupt is enabled.

2-2-2 OPERATIONS in linear mode

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

Table 2-1. SBL0 AS A SEQUENTIAL READ BLOCK IN LINEAR MODE

Explicit Operations	Implicit Operations
Set SBL0OFF	- Set SBL0OFF as start offset of sequential block - Invalidate all data words in SBF0
Run buffer fill command	- 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	- Data read - Invalidate read word and Increase SBL0OFF - If SBF0 is empty - fetch new 4 words in (SBL0OFF[17:4],0000b) location of SBL0
Repeating read from SBF0 results sequential reads	

Table 2-2. SBL0 AS A SEQUENTIAL WRITE BLOCK IN LINEAR MODE

Explicit Operations	Implicit Operations
Set SBL0OFF	- Set SBL0OFF as start offset of sequential block - Invalidate all data words in SBF0
Write to SBF0	- Data write - Validate written word and Increase SBL0OFF - If SBF0 is full - 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 2-3. SBL0 area as randomly accessed data memory

Explicit Operations	Implicit Operations
Calm accesses SBL0 area in system memory directly	To keep data consistency, - 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

2-2-3 OPERATIONS in ring mode

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

Table 2-4. 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 table2-1
Run buffer fill command	
Read SBF0	- Data read - Invalidate read word and Increase SBL0OFF - If SBL0OFF is equal to SBL0END, - 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 - fetch new 4 words from (SBL0OFF[17:4],0000b) location of SBL0
Repeating read from SBF0 results sequential reads in ring mode	

Table 2-5. 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 table2-2
Write to SBF0	- Data write - Validate written word and Increase SBL0OFF - If SBL0OFF is equal to SBL0END, - set SBF0 interrupt flag - Flush valid words and invalidate flushed words - set SBL0OFF as SBL0BEGIN - If SBF0 is full - 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 table2-2

In ring mode, Calm randomly accesses SBL0 area in the same way as the case of linear mode. Refer table2-3.

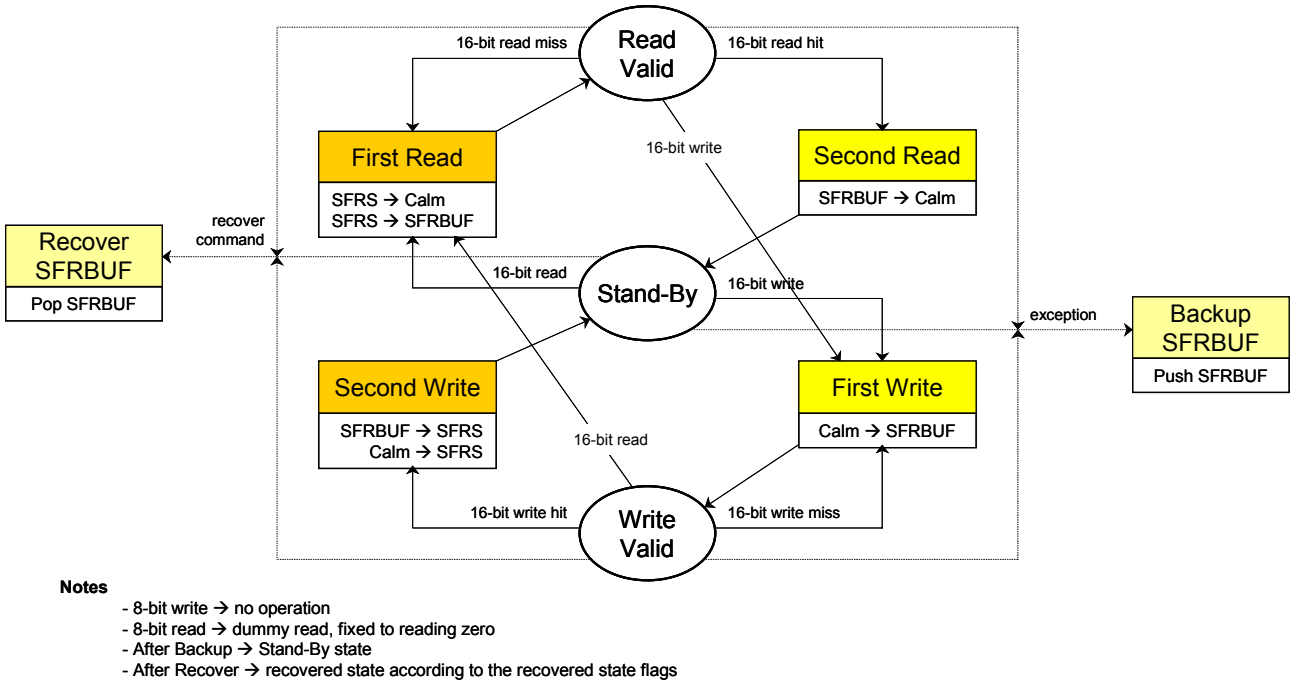


Figure 2-7. SFRSI State & Function Diagram

2-3 SFRS Interface

Since ADM works as a host processor of 32-bit AMBA bus based system, ADM should read and write 32-bit SFRS registers in the IPs on the 32-bit AMBA buses. Because ADM is based on a 16-bit MCU, CalmRISC16, it cannot access SFRS registers in usual way. In ADM, a specially designed slave register interface unit, SFRSI, handles the 16-bit/32-bit data width conversion.

To access a 32-bit register, Calm accesses the register twice. One is for higher half of the register. the other is for lower half of the register. We call this accessing twice as dual-access (dual-read, dual-write). Main function of SFRSI is conversion of dual-access from Calm to one 32-bit access to AMBA bus.

To perform this conversion, SFRSI is designed as an one-entry cache composed with a 16-bit data register, an address register tagged to the data, a valid flag and a read/write mode flag. In addition to the one-entry cache function, SFRSI performs two special functions. One is auto-invalidation on hit. It guarantees that ADM converts two 16-bit Calm accesses to one 32-bit bus access. Without this auto-invalidation, 3 or more Calm accesses may cause only 1 bus access when Calm repeats accessing same SFRS register. The other one is auto-backup on exception. Since dual-access is a single access to a SFRS register, it is an atomic operation. However, an exception (interrupt) may occur during Calm performs a dual-access. It may separate an atomic operation into two operations and may cause a failure on system operation. To prevent the separation of atomic dual-access, the contents of one-entry cache in SFRSI should be backup on exception so that they can be recovered before returning from the exception handling routine. In ADM, the backup on exception is done automatically by SFRSI hardware and the recovery is done on special command. We defined one of SYS commands of Calm as the recovery command so that the system programmers can insert the command into the end of exception handling routines. Fig.2-7 shows the operation of SFRSI.

When reading a SFRS register, dual-read is not mandatory. Calm can read any SFRS register freely in 16-bit mode. When writing, dual-write is mandatory. Calm can write any SFRS register in 16-bit mode, but only when the write hit occurs, the SFRS register is physically written. The SFR buffer data written by Calm is written to the

SFRS register if next SFRS access results write hit. The data is discarded if the next SFRS access is read or write-miss.

2-4 CalmADM3 Internal Controls

2-4-1 CalmADM3 Fast Interrupt

ADM has two fast interrupt signals. One is nFIQ, one of ADM input pins. The other is internal fast interrupt signal. When one or two of these are active, fast interrupt request is issued to Calm.

INTERNAL FIQS

In ADM are four sources of internal fast interrupt. These sources are defined as interrupt pending flags of ADMSTAT register in HFRS. Two of them, S0WIF and S1WIF, are wrapping interrupt flags set by sequential buffer unit described in section 2-2-1. Remaining two of them, PRIAIF and DRIAIF, are results of region check described in section 2-1-3. When one or more of 8 program region checks failed, PRIAIF is set. When one or more of 8 data region checks failed, DRIAIF is set.

User can enable/disable each interrupt source by set/resetting corresponding interrupt enable flag of FIQCFG in HFRS. If one or more among enabled interrupt pending flags are set, the internal fast interrupt signal is active. User should clear the interrupt pending flag by writing 1 into that flag before returning from the corresponding interrupt service routine.

2-4-2 CalmADM3 SYS commands

In ADM, two SYS commands are used internally.

SYS #5h is defined as ‘sys_idle’ meaning that Calm goes to idle mode after execution of this command. Note that, before ‘sys_idle’ is issued, ‘ready_clk_down’ flag in ADMSTAT register should checked if it’s on. Upon the interrupts or wake up command from debugger unit, Calm exits from idle mode.

SYS #18h is defined as ‘sys_rsfrb’ meaning recovery of SFR buffer described in section 2-3.

2-5 CalmADM3 Host Function Register Set

Address Base

The address values below are 7bit off set values. The full address values are additions of the offsets and the base address.

Base address of ADM function register set: 2C0000h

Register Set 1: CONFIG

ADMCFG: ADM configuration register 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]'
7:6	-	not used (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	-	not used (reading returns zero)
0	Idcinv	This flag is for partial invalidation. If it is set, all LDC instruction of Calm invalidates the target address line in I-Cache.

* 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.
* Note: xyrr bits should change while both X- and Y-Caches are disabled. Otherwise, data coherence may corrupt.

FIECFG: Fast Interrupt Enable/disable configuration register reset value : 0000h

BASE + 2h		Mode : read/write
Bit	Name	Description
15:9	-	not used (reading returns zero)
8	RIACE	Region Invalid Access Check Enable 0 : region check disable, 1 : region check enable
7:4	-	not used (reading returns zero)
3	DRIAIE	Data Region Invalid Access Interrupt Enable 0 : interrupt disable, 1 : interrupt enable
2	PRIAIE	Program Region Invalid Access Interrupt Enable 0 : interrupt disable, 1 : interrupt enable
1	S1WIE	Sequential Buffer 1 Wrapping Interrupt Enable 0 : interrupt disable, 1 : interrupt enable
0	S0WIE	Sequential Buffer 0 Wrapping Interrupt Enable 0 : interrupt disable, 1 : interrupt enable

* Note: Enabling 'region invalid access check' may cause performance degradation of ADM in terms of both speed and power. You can set RIACE flag only in debug mode, and reset it in normal operation mode that needs the full performance of ADM.

RECFG: Region Enable/disable configuration register reset value : 0001h

BASE + 4h		Mode : read/write
Bit	Name	Description
15:0	RE	0 : region disable, 1 : region enable

* Note: RE[0] is reset as high (enabled) because Region0 contains boot code(reset vector).

R0CFG_H/L: Region0 Base & Size configuration register reset value : 0000h_0002h

BASE + 6h		Mode : read/write
Bit	Name	Description

[15:0]	BASE[23:8]	Base address of Region0 in physical memory space. Mapped to [31:16] of physical memory address.

BASE + 8h		Mode : read/write
Bit	Name	Description
[15:8]	BASE[7:0]	Base address of Region0 in physical memory space. Mapped to [15:8] of physical memory address.
[7:4]	-	not used (reading returns zero)
[3:0]	SIZE	Size code of Region 0. 0000: 1K-byte 0001: 2K-byte 0010: 4K-byte 0011: 8K-byte 0100: 16K-byte 0101: 32K-byte 0110: 64K-byte 0111: 128K-byte 1000: 256K-byte 1001: 512K-byte others: 512K-byte

R1CFG_H/L ~ R15CFG_H/L: Region1~15 Base & Size configuration register reset value: xxxx_xxxxh

BASE + Ah, Eh, …, 42h		Mode : read/write
Bit	Name	Description
[15:0]	BASE[23:8]	Base address of Region1~15 in physical memory space. Mapped to [31:16] of physical memory address.

BASE + Ch, 12h, …, 44h		Mode : read/write
Bit	Name	Description
[15:8]	BASE[7:0]	Base address of Region1~15 in physical memory space. Mapped to [15:8] of physical memory address.
[7:4]	-	not used (reading returns zero)
[3:0]	SIZE	Size code of Region 0. *code definition is same as SIZE definition of R0CFG

SFRSCFG: SFRS Area Base configuration register reset value : xxxxh

BASE + 46h		Mode : read/write
Bit	Name	Description
[15:10]	PSFRSBASE	Base address of SFRS(APB) area in physical memory space. Mapped to [31:26] of physical memory address.
[9:8]	-	not used (reading returns zero)
[7:2]	HSFRSBASE	Base address of SFRS(AHB) area in physical memory space. Mapped to [31:26] of physical memory address.
[1:0]	-	not used (reading returns zero)

Register Set 2: CONTROL registers

CACHECON: Cache control register reset value : 0333h

BASE + 48h		Mode : write only. Reading returns zero.
Bit	Name	Description
[15:10]	-	not used (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]	-	not used (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]	-	not used (reading returns zero)
[2:0]	yc command flag	* Similar to the description of ‘xc command flag’

SBFCON: Sequential buffer control register reset value : 0000h

BASE + 4Ah		Mode : write only. Reading returns zero.
Bit	Name	Description
[15:14]	-	not used (reading returns zero)

[13:12]	sbf0 command flag	00,11: No operation 01: buffer fill command 10: buffer flush command
[11:10]	-	not used (reading returns zero)
[9:8]	sbf1 command flag	* Similar to the description of 'sbf0 command flag'
[7:0]	-	not used (reading returns zero)

Register Set 3: STATE registers

ADMSTAT: ADM state register reset value : 0100h

BASE + 4Ch		Mode : read/write
Bit	Name	Description
15:9	-	not used (reading returns zero)
8	ready_clk_down	Read only. Writing does not affect this flag. 0: sub-blocks in adm are running 1: sub-blocks in adm are ready to accept "no clock". This flag is reset as zero when one or more of ADM sub-blocks are doing at least one pending action like external memory access. ADM should check if this flag is set before it issues system idle or software reset.
7:4	-	not used (reading returns zero)
3	DRIAIF	Data Region Invalid Access Interrupt Flag 0 : interrupt not issued, 1 : interrupt issued
2	PRIAIF	Program Region Invalid Access Interrupt Flag 0 : interrupt not issued, 1 : interrupt issued
1	S1WIF	Sequential Buffer 1 Wrapping Interrupt Flag 0 : interrupt not issued, 1 : interrupt issued
0	S0WIF	Sequential Buffer 0 Wrapping Interrupt Flag 0 : interrupt not issued, 1 : interrupt issued

* Note: You can clear a specific bit of interrupt flags by writing a data to this register. It clears only the bit positions corresponding to those set to one in the written data. The bit positions corresponding to those that are set to zero in the written data remains as they are.

CACHESTAT: Cache status register reset value : 0000h

BASE + 4Eh		Mode : read only. Writing does not affect this register.
Bit	Name	Description
[15:10]	-	not used (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]	-	not used (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]	-	not used (reading returns zero)
[2:0]	yc state flag	* Similar to the description of 'xc command flag'

SBFSTAT: Sequential buffer status register reset value : 0000h

BASE + 50h		Mode : read only. Writing does not affect this register.
Bit	Name	Description
[15]	-	not used (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]	-	not used (reading returns zero)
[10:8]	sbf1 state flag	* Similar to the description of 'sbf0 state flag'
[7:0]	-	not used (reading returns zero)

Register Set 4: Sequential Buffer Function Registers

SBL0OFF_H: Higher bits of offset register of sequential block 0 area reset value : 0000h

BASE + 52h	Mode : read/write

Bit	Name	Description
[15:2]	-	not used (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 + 54h		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]. Its is fixed to 0 when SBF0 is working in 32-bit access mode.
[0]	sbl0off_0	SBL0OFF[0]. Its 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 system memory access.

SBL1OFF_H: Higher bits of offset register of sequential block 1 area reset value : 0000h

BASE + 56h		Mode : read/write
Bit	Name	Description
[15:2]	-	not used (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 + 58h		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]. Its is fixed to 0 when SBF1 is working in 32-bit access mode.
[0]	Sbl1off_0	SBL1OFF[0]. Its 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 system memory access.

SBL0BEGIN_H: Higher bits of begin offset of sequential block 0 area in ring mode reset value : 0000h

BASE + 5Ah	Mode : read/write

Bit	Name	Description
[15:2]	-	not used (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 + 5Ch		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.

SBL1BEGIN_H: Higher bits of begin offset of sequential block 1 area in ring mode reset value : 0000h

BASE + 5Eh		Mode : read/write
Bit	Name	Description
[15:2]	-	not used (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 + 60h		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.

SBL0END_H: Higher bits of end offset of sequential block 0 area in ring mode reset value : 0000h

BASE + 62h		Mode : read/write
Bit	Name	Description
[15:2]	-	not used (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 + 64h		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.

SBL1END_H: Higher bits of end offset of sequential block 1 area in ring mode reset value : 0000h

BASE + 66h		Mode : read/write
Bit	Name	Description
[15:2]	-	not used (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 + 68h		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.

Register Set 5: SFRSI Function Registers

SFRBSTAT0/1/2: SFR buffer state register reset value : 0xxxh

BASE + 6Ah, 70h, 76h		Mode : read/write
Bit	Name	Description
[15:13]	-	not used (reading returns zero)
[12]	sfrbvalid	This flag is set as high only when SFR buffer contains valid read/write data.
[11:9]	-	not used (reading returns zero)
[8]	sfrbmode	0: read mode 1: write mode This flag is effective only when sfrbvalid flag is set as high.
[7:6]	-	not used (reading returns zero)

[5:0]	Sfrbtag[21:16]	Higher 6 bits of SFR buffer tag address

SFRBTAG0/1/2: SFR tag address register reset value : xxxxh

BASE + 6Ch, 72h, 78h		Mode : read/write
Bit	Name	Description
[15:0]	Sfrbtag[15:0]	Lower 16 bits of SFR buffer tag address

SFRBUF0/1/2: SFR data register reset value : xxxxh

BASE + 6Eh, 74h, 7Ah		Mode : read/write
Bit	Name	Description
[15:0]	Sfrbuf	SFR data buffer

Register Set 6: DHCLK Control

DHCLK_CON: DHCLK delay control register reset value : 0000h

BASE + 7Ch		Mode : read/write
Bit	Name	Description
[15:3]	-	not used (reading returns zero)
[2:0]	Dhclk_con	DHCLK (clock of I-Cache memories) delay control value

3 CalmADM3 Hardware Specifications

3-1 Interface Spec.

3-1-1 pin descriptions

Table 3-1. CalmADM3 Pin Description

Group	Signal	direction	polarity	Description
System Interface	nRES	I	low	reset signal
	MCLK	I	rising	clock
	nSYSID	O	low	SYS command indicator
	DA[4:0]	O	-	DA output for SYS commands
	PMODE	O	high	Privileged mode indicator
	nIRQ	I	low	Interrupt request
	nFIQ	I	low	Fast interrupt request
	nEXPACK	O	low	Acknowledge for exceptions
AHB Master Interface	HBUSREQM	O	high	
	HGRANTM	I	high	
	HREADYM	I	high	
	HADDRM [31:0]	O	-	
	HTRANSM [1:0]	O	-	
	HBURSTM [3:0]	O	-	
	HSIZEM [2:0]	O	-	
	HWRITEM	O	high	
	HWDATAM[31:0]	O	-	
	HRDATAM[31:0]	I	-	
Debug Interface	nTRST	I	low	
	TCK	I	rising	
	TMS	I	-	
	TDI	I	-	
	TDO	O	-	
	nres_dbu	O	low	Reset signal from DBU in ADM
	runst_dbu	O	high	Indicating Calm is not stopped by DBU
BIST Interface	BISTMODE	I	-	BIST input signals
	MEMSEL[2:0]	I	-	
	BCLK	I	-	
	BIST_ON	I	high	
	DIAG_BIST_XYC	O	-	BIST output signals for X and Y-Cache memories
	DONE_BIST_XYC	O	-	
	ERROB_BIST_XYC	O	-	
	PAUSE_BIST_XYC	O	-	
	DIAG_BIST_IC	O	-	BIST output signals for I-Cache memories
	DONE_BIST_IC	O	-	
	ERROB_BIST_IC	O	-	
	PAUSE_BIST_IC	O	-	
Scan Test Interface	SCAN_TEST_MODE	I	high	
	SCAN_ENABLE	I	high	
	SCAN_IN	I	-	

	SCAN_OUT	O	-	

3-1-2 pin timing diagrams

**WILL BE AVAILABLE IN LATER RELEASE IF NECESSARY*

3-2 Arbiter

FEATURES

Prioritized scheduling of bus requests from 8 sources

SFRSI (the highest priority)
> SBF0
> SBF1
> X-Cache
> Y-Cache
> I-Cache
> X-Cache Write-Back
> Y-Cache Write-Back (the lowest priority)

3-3 AHBMIU (AHB Master Interface Unit)

FEATURES

- 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
- Single burst word access to SFRS

3-4 INSTRUCTION CACHE

OVERVIEW

I-Cache is a direct-mapped, 128x16-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. I-Cache also supports one-line-invalidation function for debugging convenience. If Calm performs LDC instruction while 'ldcinv' flag of ADMCFG 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 (system memory address). This conversion is done inside of I-Cache by adding cache address with the BASE field of corresponding region configuration register (R0CFG ~ R7CFG).

FEATURES

- Direct-mapped cache
- 128 data entries (cache lines)
- 256-bit wide data memory (16 half-word instructions in a cache line)
- 11-bit wide tag memory (10-bit tag address + 1 valid bit)
- 24-bit base fields for mapping to physical address
- Supports all invalidation and one line invalidation

3-5 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 works as the data cache for Calm area as a kind of 4-way set-associative cache.

3-5-1 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-invalidation command should be done. After X-Cache off, flush command should be done to keep system memory data consistency.
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 (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 only.

FEATURES

- 2-way set associative cache
- 128 data entries (cache lines) in a set
- 8 audwords (192 bit) in a cache line
- only 128 bits of 192-bit cache line are used as Calm area cache
- 24-bit wide Cache Tag Memory ((11-bit tag address + 1 valid bit) * 2 sets)
- 256 dirty flags (for 128 lines * 2 sets)
- 24-bit base fields for mapping to physical address
- Supports invalidation and flushing.

3-5-2 X/Y-Cache as calm area data cache

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 only. By setting XYRR code in ADMCFG 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 the BASE field of corresponding region configuration register (R8CFG ~ R11CFG).

OPERATIONS

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

3-6 SBFU (S-buffer unit)

OVERVIEW

In ADM, two sequential buffers, SBF0 and SBF1, are included. A sequential buffer consists of a 128-bit FIFO, a 18-bit offset register, two 18-bit boundary offset registers and a 16-bit data buffer.

128-bit FIFO is used as buffer when Mac sequentially accesses sequential block. A 16-bit data register is used as buffer when Calm randomly accesses sequential block. 18-bit offset register contains offset address of sequential block data to be accessed. Since it is defined in HFRS, user can write the start address of sequential block into

this register. This register is added with BASE field of the region configuration register (S0CFG, S1CFG) to generate physical address of sequential data. After a sequential access to sequential block has done, this register value is increased automatically. This increment is modulated by the size limit defined in SIZE field of the region configuration register (S0CFG, S1CFG). In ring mode, auto-incremented register value is compared with 18-bit end offset register value. If these are same, offset register is set as 18-bit begin offset register value and interrupt flag in ADMSTAT register is set.

Two special commands, fill and flush, 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. Access mode, unit data size and data size conversion scheme in a SBFU are defined in ADMCFG register. More of SBFU operation is described in 2-2.

FEATURES

- 4-word FIFO used as a sequential buffer
- 18-bit offset address register
  : automatically increased during sequential accesses
  : the increment is modulated according to the SIZE field
- 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 in the ring mode.

3-7 SFRSI (SFRS interface unit)

FEATURES

- One-entry cache composed of
  : SFRBUF[15:0]: SFR read/write Buffer
  : SFRBTAG[21:0]: SFRBUF tag address
  : SFRBVALID: “0” – invalid, “1” – valid
  : SFRBMODE: “0” – read mode, “1” – write mode
- One-entry cache with
  : auto-invalidation on hit
  : auto-backup on exception
  : recovery on ‘sys_rsfrb’ command
- Backup stack
  : depth = 2 (for two exceptions, IRQ and FIQ)

: contains all components of the one-entry cache

- All registers and flags in the one-entry cache and the stack are defined in HFRS
  : for debugging and context switching purpose

4 Constraints on using CalmADM3

**NOTE: THIS CHAPTER IS EXACTLY SAME AS THE ONE IN CALMADM2 SPECIFICATION. IF YOU ARE FAMILIAR WITH CALMADM2, YOU DON'T NEED TO READ THIS CHAPTER.*

4-1 Constraints on using CalmRISC16F

DELAY SLOT INSTRUCTION

As an instruction at delay slot, following 3 types of instructions are prohibited.

- Break instruction
- Branch instructions
- two word instructions

4-2 Constraints on using CalmMAC24F

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 table6-1.

Example code

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

Table 6-1. 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 a EMIN/EMAX instruction, 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 table6-2.

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 6-2. 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

4-3 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 performed, 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.

4-4 Constraints on accessing X/Y-Caches

SETTING XYRR

The XYRR flags in ADMCFG 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 system memory may corrupt.

4-5 Definitions of Abbreviations

Instruction tables in this chapter are extracted from “3.3 Quick Reference” instruction table in “CalmMAC24 DSP Coprocessor Architecture Reference Manual”. Definition of abbreviations used in instruction tables is listed following tables. This tables are copy of “3.2 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

5 Information for CalmShine Development

5-1 Simulator Spec.

MEMORY MAP

Simulator should keep track of memory configuration in section 2-1 from Calm and Mac point of view.

- Program regions, Calm regions, X/Y/S0/S1 regions can be modeled same as the CalmADM2 case.

- Modeling the region control and the address translation to system memory space is not needed.

- HFRS Registers can be modeled those read/write function only. Modeling precise functionality of them described in this document is not needed.
  - Exception:
    Some sequential buffer related registers (flags) should be modeled precisely as described in next paragraph.

- SFRS area can be modeled as a normal memory area.

MODELING SEQUENTIAL BUFFERS

**NOTE: THIS SECTION IS EXACTLY SAME AS THE ONE IN CALMADM2 SPECIFICATION. IF YOU ARE FAMILIAR WITH CALMADM2, YOU DON'T NEED TO READ THIS SECTION.*

Simulator should keep track of the sequential accesses in 2-2 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 control 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 0 when Calm writes values to these registers.
  When auto-incremented, the incrementing step should follow the mode defined by flags of ADMCFG register in HFRS.
  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 ADMCFG register in HFRS.
- Modeling commands on sequential buffers, defined by SBFCON register in HFRS, 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.

5-2 Emulator Spec.

ACCESSING SFRS

Because of hardware limitation of SFRSI block, SFRS area should be accessed in 16-bit mode only. Byte-read from SFRS area always returns zero. Byte-write to SFRS area is same as no-operation.

10 GPIO PORTS

OVERVIEW

S5L840F has 72 multi-functional GPIO (general-purpose input/output) port pins organized into ten 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.

Table 10-1. Port Configuration Overview

Port 0		Selectable Pin Functions						
		Function 1	Function 2			Function 3		
	Pin No.	GPIO	Pin_name	I/O	Module	Pin _name	I/O	Module
P0.0	38	Input/output	TACK/TACAP	I	Timer		–	
P0.1	39	Input/output	TAOUT	O	Timer		–	
P0.2	40	Input/output	TCCK	I	Timer		–	
P0.3	41	Input/output	SPDIF	O	SPDIF		–	
P0.4	31	Input/output	RX	I	UART		–	
P0.5	32	Input/output	TX	O	UART		–	
P0.6	42	Input/output	EINT6	I	ICU	SDWP	I	SDC
P0.7	43	Input/output	EINT7	I	ICU	NF_RBN	I	NF

Port 1		Selectable Pin Functions						
		Function 1	Function 2			Function 3		
	Pin No.	GPIO	Pin _name	I/O	Module	Pin _name	I/O	Module
P1.0	108	Input/output	MOSI	I/O	SPI		–	
P1.1	109	Input/output	MISO	I/O	SPI		–	
P1.2	110	Input/output	SPICLK	I/O	SPI		–	
P1.3	115	Input/output	SCL	I/O	IIC		–	
P1.4	116	Input/output	SDA	I/O	IIC		–	
P1.5	117	Input/output	nSSI	I	SPI		–	
P1.6	10	Input/output	EINT4	I	ICU		–	
P1.7	11	Input/output	EINT5	I	ICU		–	

Port 2		Selectable Pin Functions						
		Function 1	Function 2			Function 3		
	Pin No.	GPIO	Pin _name	I/O	Module	Pin _name	I/O	Module
P2.0	48	Input/output	EINT0	I	ICU		–	
P2.1	49	Input/output	EINT1	I	ICU		–	
P2.2	50	Input/output	EINT2	I	ICU		–	
P2.3	51	Input/output	EINT3	I	ICU		–	

Table 10-1. Port Configuration Overview (Continued)

Port 3		Selectable Pin Functions						
		Function 1	Function 2			Function 3		
	Pin No.	GPIO	Pin_name	I/O	Module	Pin _name	I/O	Module
P3.0	61	Input/output	CK_SDC_MMC	O	SDC/MMC		–	
P3.1	73	Input/output	CK_MS	O	MS		–	

Port 4		Selectable Pin Functions						
		Function 1	Function 2			Function 3		
	Pin No.	GPIO	Pin _name	I/O	Module	Pin _name	I/O	Module
P4.0	56	Input/output	IO0	I/O	SMC		–	
P4.1	58	Input/output	IO1	I/O	SMC		–	
P4.2	60	Input/output	IO2	I/O	SMC		–	
P4.3	63	Input/output	IO3	I/O	SMC		–	
P4.4	69	Input/output	IO4	I/O	SMC		–	
P4.5	71	Input/output	IO5	I/O	SMC		–	
P4.6	74	Input/output	IO6	I/O	SMC		–	
P4.7	76	Input/output	IO7	I/O	SMC		–	

Port 5		Selectable Pin Functions						
		Function 1	Function 2			Function 3		
	Pin No.	GPIO	Pin _name	I/O	Module	Pin _name	I/O	Module
P5.0	57	Input/output	IO8	I/O	SMC	D1	–	SDC/MMC
P5.1	59	Input/output	IO9	I/O	SMC	D0	–	SDC
P5.2	62	Input/output	IO10	I/O	SMC	CMD	–	SDC/MMC
P5.3	64	Input/output	IO11	I/O	SMC	D3	–	SDC
P5.4	70	Input/output	IO12	I/O	SMC	D2	–	SDC
P5.5	72	Input/output	IO13	I/O	SMC		–	
P5.6	75	Input/output	IO14	I/O	SMC	D0	–	MS
P5.7	77	Input/output	IO15	I/O	SMC	BS	–	MS

Table 10-1. Port Configuration Overview (Continued)

Port 6		Selectable Pin Functions						
		Function 1	Function 2			Function 3		
	Pin No.	GPIO	Pin_name	I/O	Module	Pin _name	I/O	Module
P6.0	78	Input/output	nRE	O	SMC		–	
P6.1	79	Input/output	nCE0	O	SMC		–	
P6.2	80	Input/output	nCE1	O	SMC		–	
P6.3	81	Input/output	nCE2	O	SMC		–	
P6.4	82	Input/output	CLE	O	SMC		–	
P6.5	83	Input/output	ALE	O	SMC		–	
P6.6	84	Input/output	nWE	O	SMC		–	
P6.7	85	Input/output	nWP	O	SMC		–	

Port 7		Selectable Pin Functions						
		Function 1	Function 2			Function 3		
	Pin No.	GPIO	Pin _name	I/O	Module	Pin _name	I/O	Module
P7.0	86	Input/output	SDI	I	IIS		–	
P7.1	87	Input/output	WS	O	IIS		–	
P7.2	90	Input/output	SDO	O	IIS		–	
P7.3	91	Input/output	SCLK	O	IIS		–	
P7.4	92	Input/output	MCLK	O	IIS		–	

Port 8		Selectable Pin Functions						
		Function 1	Function 2			Function 3		
	Pin No.	GPIO	Pin _name	I/O	Module	Pin _name	I/O	Module
P8.0	18	Input/output	LD0	O	LCD		–	
P8.1	24	Input/output	LD1	O	LCD		–	
P8.2	25	Input/output	LD2	O	LCD		–	
P8.3	26	Input/output	LD3	O	LCD		–	
P8.4	27	Input/output	LD4	O	LCD		–	
P8.5	28	Input/output	LD5	O	LCD		–	
P8.6	29	Input/output	LD6	O	LCD		–	
P8.7	30	Input/output	LD7	O	LCD		–	

Table 10-1. Port Configuration Overview (Continued)

Port 9		Selectable Pin Functions						
		Function 1	Function 2			Function 3		
	Pin No.	GPIO	Pin _name	I/O	Module	Pin _name	I/O	Module
P9.0	33	Input/output	RE	O	LCD		–	
P9.1	34	Input/output	WE	O	LCD		–	
P9.2	35	Input/output	CS	O	LCD		–	
P9.3	36	Input/output	REG	O	LCD		–	
P9.4	37	Input/output	RESET	O	LCD		–	

Port 10		Selectable Pin Functions						
		Function 1	Function 2			Function 3		
	Pin No.	GPIO	Pin _name	I/O	Module	Pin _name	I/O	Module
P10.0	93	Input/output		–			–	
P10.1	94	Input/output		–			–	
P10.2	95	Input/output		–			–	
P10.3	96	Input/output		–			–	
P10.4	104	Input/output		–			–	
P10.5	105	Input/output		–			–	
P10.6	106	Input/output		–			–	
P10.7	107	Input/output		–			–	

PORT CONTROL DESCRIPTIONS

Port Configuration Register (PCON0 – PCON10)

In S5L840F, most pins are multiplexed, and the PCONn (port control register) determines which function is used for each pin.

If P0.6-P0.7 is used for the wakeup signal in power down mode, these ports must be configured for interrupt mode.

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	0x0000 0000
PDAT0	0x3CF0 0004	R/W	The data register for port 0	Undefined

PCON0	Bit	Description
P0.0	[1:0]	00 = Input 01 = Output 10 = TACK/TACAP(Timer) 11 = Not used
P0.1	[3:2]	00 = Input 01 = Output 10 = TAOUT(Timer) 11 = Not used
P0.2	[5:4]	00 = Input 01 = Output 10 = TCCK(Timer) 11 = Not used
P0.3	[7:6]	00 = Input 01 = Output 10 = SPDIF(SPDIF) 11 = Not used
P0.4	[9:8]	00 = Input 01 = Output 10 = RX(UART) 11 = Not used
P0.5	[11:10]	00 = Input 01 = Output 10 = TX(UART) 11 = Not used
P0.6	[13:12]	00 = Input 01 = Output 10 = EINT6(ICU) 11 = SDWP(SDC)
P0.7	[15:14]	00 = Input 01 = Output 10 = EINT7(ICU) 11 = NF_RBN(SMC)

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

PCON1	Bit	Description
P1.0	[1:0]	00 = Input 01 = Output 10 = MOSI(SPI) 11 = Not used
P1.1	[3:2]	00 = Input 01 = Output 10 = MISO(SPI) 11 = Not used
P1.2	[5:4]	00 = Input 01 = Output 10 = SPICLK(SPI) 11 = Not used
P1.3	[7:6]	00 = Input 01 = Output 10 = SCL(IIC:Open Drain) 11 = Not used
P1.4	[9:8]	00 = Input 01 = Output 10 = SDA(IIC:Open Drain) 11 = Not used
P1.5	[11:10]	00 = Input 01 = Output 10 = nSSI(SPI) 11 = Not used
P1.6	[13:12]	00 = Input 01 = Output 10 = EINT4(ICU) 11 = Not used
P1.7	[15:14]	00 = Input 01 = Output 10 = EINT5(ICU) 11 = Not used

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

PCON2	Bit	Description
P2.0	[1:0]	00 = Input 01 = Output 10 = EINT0(ICU) 11 = Not used
P2.1	[3:2]	00 = Input 01 = Output 10 = EINT1(ICU) 11 = Not used
P2.2	[5:4]	00 = Input 01 = Output 10 = EINT2(ICU) 11 = Not used
P2.3	[7:6]	00 = Input 01 = Output 10 = EINT3(ICU) 11 = Not used

PDAT2	Bit	Description
P2[3:0]	[3:0]	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

PCON3	Bit	Description
P3.0	[1:0]	00 = Input 01 = Output 10 = CK_SDC(SDC/MMC) 11 = Not used
P3.1	[3:2]	00 = Input 01 = Output 10 = CK_MS(MS) 11 = Nout used

PDAT3	Bit	Description
P3[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 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

PCON4	Bit	Description
P4.0	[1:0]	00 = Input 01 = Output 10 = IO0(SMC) 11 = Not used
P4.1	[3:2]	00 = Input 01 = Output 10 = IO1(SMC) 11 = Not used
P4.2	[5:4]	00 = Input 01 = Output 10 = IO2(SMC) 11 = Not used
P4.3	[7:6]	00 = Input 01 = Output 10 = IO3(SMC) 11 = Not used
P4.4	[9:8]	00 = Input 01 = Output 10 = IO4(SMC) 11 = Not used
P4.5	[11:10]	00 = Input 01 = Output 10 = IO5(SMC) 11 = Not used
P4.6	[13:12]	00 = Input 01 = Output 10 = IO6(SMC) 11 = Not used
P4.7	[15:14]	00 = Input 01 = Output 10 = IO7(SMC) 11 = 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

PCON5	Bit	Description	
P5.0	[1:0]	00 = Input 10 = IO8(SMC)	01 = Output 11 = D1(SDC/MMC)
P5.1	[3:2]	00 = Input 10 = IO9(SMC)	01 = Output 11 = D0(SDC)
P5.2	[5:4]	00 = Input 10 = IO10(SMC)	01 = Output 11 = CMD(SDC/MMC)
P5.3	[7:6]	00 = Input 10 = IO11(SMC)	01 = Output 11 = D3(SDC)
P5.4	[9:8]	00 = Input 10 = IO12(SMC)	01 = Output 11 = D2(SDC)
P5.5	[11:10]	00 = Input 10 = IO13(SMC)	01 = Output 11 = Not used
P5.6	[13:12]	00 = Input 10 = IO14(SMC)	01 = Output 11 = D0(MS)
P5.7	[15:14]	00 = Input 10 = IO15(SMC)	01 = Output 11 = BS(MS)

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

PCON6	Bit	Description
P6.0	[1:0]	00 = Input 01 = Output 10 = nRE(SMC) 11 = Not used
P6.1	[3:2]	00 = Input 01 = Output 10 = nCE0(SMC) 11 = Not used
P6.2	[5:4]	00 = Input 01 = Output 10 = nCE1(SMC) 11 = Not used
P6.3	[7:6]	00 = Input 01 = Output 10 = nCE2(SMC) 11 = Not used
P6.4	[9:8]	00 = Input 01 = Output 10 = CLE(SMC) 11 = Not used
P6.5	[11:10]	00 = Input 01 = Output 10 = ALE(SMC) 11 = Not used
P6.6	[13:12]	00 = Input 01 = Output 10 = nWE(SMC) 11 = Not used
P6.7	[15:14]	00 = Input 01 = Output 10 = nWP(SMC) 11 = Not used

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	0x0000 0000
PDAT7	0x3CF0 0074	R/W	The data register for port 7	Undefined

PCON7	Bit	Description
P7.0	[1:0]	00 = Input 01 = Output 10 = SDI(IIS) 11 = Not used
P7.1	[3:2]	00 = Input 01 = Output 10 = WS(IIS) 11 = Not used
P7.2	[5:4]	00 = Input 01 = Output 10 = SDO(IIS) 11 = Not used
P7.3	[7:6]	00 = Input 01 = Output 10 = SCLK(IIS) 11 = Not used
P7.4	[9:8]	00 = Input(EINT4) 01 = Output 10 = MCLK(IIS) 11 = Not used

PDAT7	Bit	Description
P7[4:0]	[4:0]	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 8 CONTROL REGISTERS (PCON8, PDAT8)

Register	Address	R/W	Description	Reset Value
PCON8	0x3CF0 0080	R/W	Configures the pins of port 8	0x0000 0000
PDAT8	0x3CF0 0084	R/W	The data register for port 8	Undefined

PCON8	Bit	Description
P8.0	[1:0]	00 = Input 01 = Not used 10 = LD0(LCD) 11 = Not used
P8.1	[3:2]	00 = Input 01 = Not used 10 = LD1(LCD) 11 = Not used
P8.2	[5:4]	00 = Input 01 = Not used 10 = LD2(LCD) 11 = Not used
P8.3	[7:6]	00 = Input 01 = Not used 10 = LD3(LCD) 11 = Not used
P8.4	[9:8]	00 = Input 01 = Not used 10 = LD4(LCD) 11 = Not used
P8.5	[11:10]	00 = Input 01 = Not used 10 = LD5(LCD) 11 = Not used
P8.6	[13:12]	00 = Input 01 = Not used 10 = LD6(LCD) 11 = Not used
P8.7	[15:14]	00 = Input 01 = Not used 10 = LD7(LCD) 11 = Not used

PDAT8	Bit	Description
P8[7:0]	[3:0]	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 9 CONTROL REGISTERS (PCON9, PDAT9)

Register	Address	R/W	Description	Reset Value
PCON9	0x3CF0 0090	R/W	Configures the pins of port 9	0x0000 0000
PDAT9	0x3CF0 0094	R/W	The data register for port 9	Undefined

PCON5	Bit	Description
P9.0	[1:0]	00 = Input 01 = Output 10 = RE(LCD) 11 = Not used
P9.1	[3:2]	00 = Input 01 = Output 10 = WE(LCD) 11 = Not used
P9.2	[5:4]	00 = Input 01 = Output 10 = CS(LCD) 11 = Not used
P9.3	[7:6]	00 = Input 01 = Output 10 = REG(LCD) 11 = Not used
P9.4	[9:8]	00 = Input 01 = Output 10 = RESET(LCD) 11 = Not used

PDAT5	Bit	Description
P9[4:0]	[4:0]	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 5	0x0000 0000
PDAT10	0x3CF0 00A4	R/W	The data register for port 5	Undefined

PCON5	Bit	Description
P10.0	[1:0]	00 = Input 01 = Output 1x = Not used
P10.1	[3:2]	00 = Input 01 = Output 1x = Not used
P10.2	[5:4]	00 = Input 01 = Output 1x = Not used
P10.3	[7:6]	00 = Input 01 = Output 1x = Not used
P10.4	[9:8]	00 = Input 01 = Output 1x = Not used
P10.5	[11:10]	00 = Input 01 = Output 1x = Not used
P10.6	[13:12]	00 = Input 01 = Output 1x = Not used
P10.7	[15:14]	00 = Input 01 = Output 1x = Not used

PDAT5	Bit	Description
P10[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.

EINTPOL (EXTERNAL INTERRUPT POLARITY CONTROL REGISTER)

The EXTINTR register selects the trigger types among various level or edge trigger mode of the external interrupt.

Register	Address	R/W	Description	Reset Value
EXTPOL	0x39C0 0018	R/W	External Interrupt control register	0x0000 0000

EXTINTR	Bit	Description
P2.0/INT0	[0]	Trigger mode of the EXTINT0. 0 = Falling edge triggered 1 = Rising edge triggered
P2.1/INT1	[1]	Trigger mode of the EXTINT1. 0 = Falling edge triggered 1 = Rising edge triggered
P2.2/INT2	[2]	Trigger mode of the EXTINT2. 0 = Falling edge triggered 1 = Rising edge triggered
P2.3/INT3	[3]	Trigger mode of the EXTINT3. 0 = Falling edge triggered 1 = Rising edge triggered
P1.6/INT4	[4]	Trigger mode of the EXTINT4. 0 = Falling edge triggered 1 = Rising edge triggered
P1.7/INT5	[5]	Trigger mode of the EXTINT5. 0 = Falling edge triggered 1 = Rising edge triggered
P0.6/INT6	[6]	Trigger mode of the EXTINT6. 0 = Falling edge triggered 1 = Rising edge triggered
P0.7/INT7	[7]	Trigger mode of the EXTINT7. 0 = Falling edge triggered 1 = Rising edge triggered

NOTES:

1. Because each external interrupt pins has a digital filter, the interrupt controller can recognize a request signal that is longer than 3 clocks.
2. If users want to change the trigger mode in the external interrupt mode, users are first required to switch the corresponding pin to input mode and then change the trigger mode.

NOTES

ICU (Interrupt Control Unit)

USER'S MANUAL

REV0.0

Kim, Hye-Ryeong

MEDIA Player PJ

System LSI Business

Device Solution Network Division

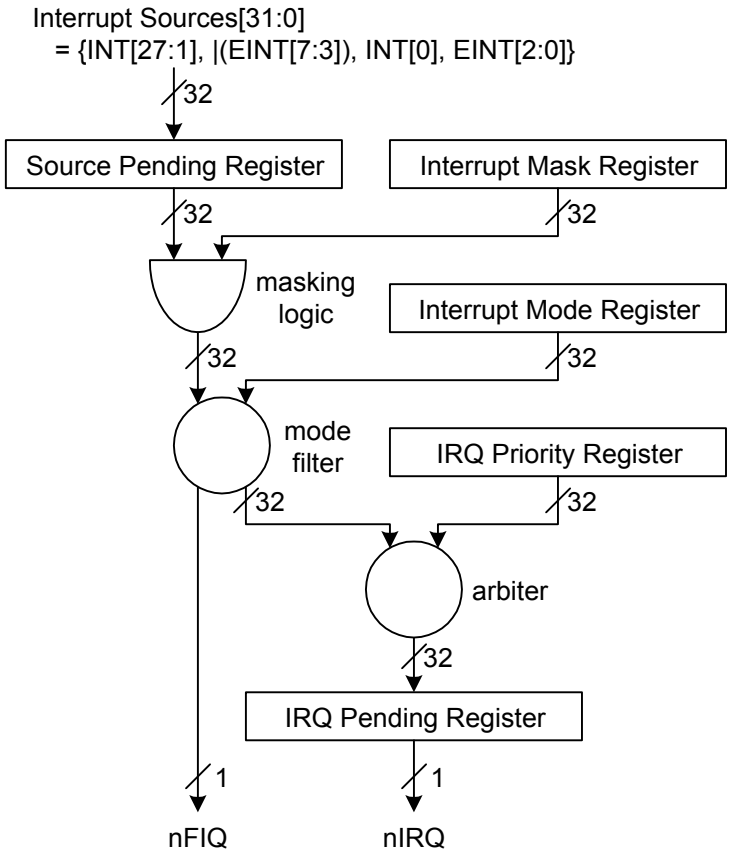
Samsung Electronics

2.17.1 FUNCTIONAL DESCRIPTION

The interrupt controller in S5L840F receives the request from 32interrupt 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 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

S5L840F has two interrupt pending resisters. 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.

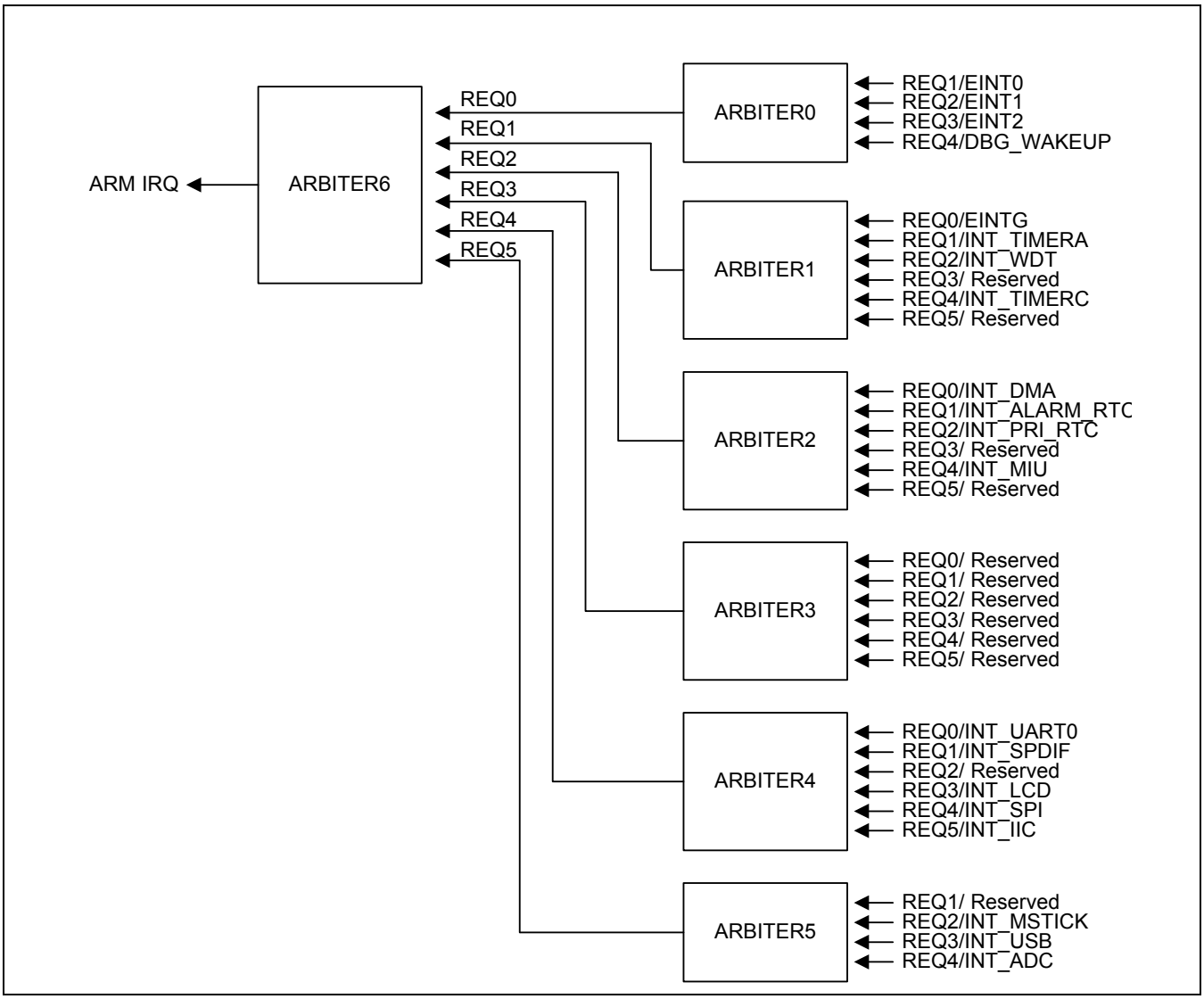
Interrupt Mask Register

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 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 CALMADM3. 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 WATCHDOG-TIMER OPERATION.

2.17.3 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_USB	[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
Reserved	[24]	Not used		0
INT_SPDIF	[23]	0 = Not requested,	1 = Requested	0
INT_UART0	[22]	0 = Not requested,	1 = Requested	0
Reserved	[21]	Not used		0
Reserved	[20]	Not used		0
Reserved	[19]	Not used		0
Reserved	[18]	Not used		0
Reserved	[17]	Not used		0
Reserved	[16]	Not used		0
Reserved	[15]	Not used		0
INT_MIU	[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]	Not used		0
INT_DMA	[10]	0 = Not requested,	1 = Requested	0
Reserved	[9]	Not used		0
INT_TIMER C	[8]	0 = Not requested,	1 = Requested	0
Reserved	[7]	Not used		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
DBG_WAKEUP	[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_USB	[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
Reserved	[24]	Not used	0
INT_SPDIF	[23]	0 = IRQ, 1 = FIQ	0
INT_UART0	[22]	0 = IRQ, 1 = FIQ	0
Reserved	[21]	Not used	0
Reserved	[20]	Not used	0
Reserved	[19]	Not used	0
Reserved	[18]	Not used	0
Reserved	[17]	Not used	0
Reserved	[16]	Not used	0
Reserved	[15]	Not used	0
INT_MIU	[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
Reserved	[9]	Not used	0
INT_TIMER C	[8]	0 = IRQ, 1 = FIQ	0
Reserved	[7]	Not used	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
DBG_WAKEUP	[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_USB	[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
Reserved	[24]	Not used	0
INT_SPDIF	[23]	1 = Service available, 0 = Masked	0
INT_UART0	[22]	1 = Service available, 0 = Masked	0
Reserved	[21]	Not used	0
Reserved	[20]	Not used	0
Reserved	[19]	Not used	0
Reserved	[18]	Not used	0
Reserved	[17]	Not used	0
Reserved	[16]	Not used	0
Reserved	[15]	Not used	0
INT_MIU	[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
Reserved	[9]	Not used	0
INT_TIMER C	[8]	1 = Service available, 0 = Masked	0
Reserved	[7]	Not used	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
DBG_WAKEUP	[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_000C	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_USB	[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
Reserved	[24]	Not used	0
INT_SPDIF	[23]	0 = Not requested, 1 = Requested	0

INT_UART0	[22]	0 = Not requested, 1 = Requested	0
Reserved	[21]	Not used	0
Reserved	[20]	Not used	0
Reserved	[19]	Not used	0
Reserved	[18]	Not used	0
Reserved	[17]	Not used	0
Reserved	[16]	Not used	0
Reserved	[15]	Not used	0
INT_MIU	[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
Reserved	[9]	Not used	0
INT_TIMER C	[8]	0 = Not requested, 1 = Requested	0
Reserved	[7]	Not used	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
DBG_WAKEUP	[3]	0 = Not requested, 1 = Requested	0
EINT2	[2]	0 = Not requested, 1 = Requested	0
EINT1	[1]	0 = Not requested, 1 = Requested	0
EINT0	[0]	0 = Not requested, 1 = Requested	0

INTERRUPT OFFSET REGISTER (INTOFFSET)

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

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

INT Source	The OFFSET value	INT Source	The OFFSET value
INT_ADC	31	Reserved	15
INT_USB	30	INT_MIU	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	Reserved	9

Reserved	24	INT_TIMER C	8
INT_SPDIF	23	Reserved	7
INT_UART0	22	INT_WDT	6
Reserved	21	INT_TIMER A	5
Reserved	20	EINTG	4
Reserved	19	DBG_WAKEUP	3
Reserved	18	EINT2	2
Reserved	17	EINT1	1
Reserved	16	EINT0	0

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

EXTERNAL INTERRUPT POLARITY SELECTION REGISTER

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

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

MIU (Memory Interface Unit)

User's mamual

REV0.0

Kim, Hye-ryeong

MEDIA Plyer P/J
System LSI Business
Device Solution Network Division
Samsung Electronics

2.16.1 FUNCTIONAL DESCRIPTION

Features

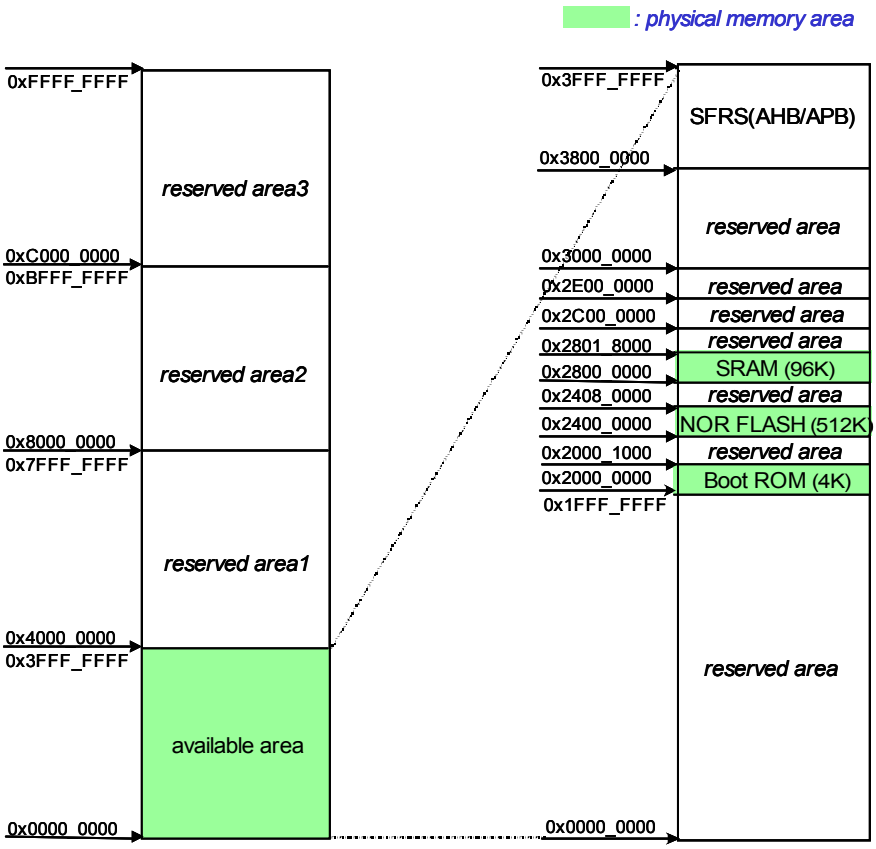
MIU supports ROM interface, SRAM interfaces and NOR FLASH memory device interfaces. Main features of MIU are as followings.

- supports 1 ROM access areas and 1 SRAM access areas. Each ROM and SRAM access areas have 32Mbyte address space respectively.
- supports 8/16/32bit access to SRAM and ROM.
- supports 4Mbit(512Kbyte) NOR flash memory.
- Supports 8/16/32bit access in NOR flash memory READ operation. But, only 32bit access is allowed in WRITE operation.
- not supports burst mode. Only single transfer is supported.

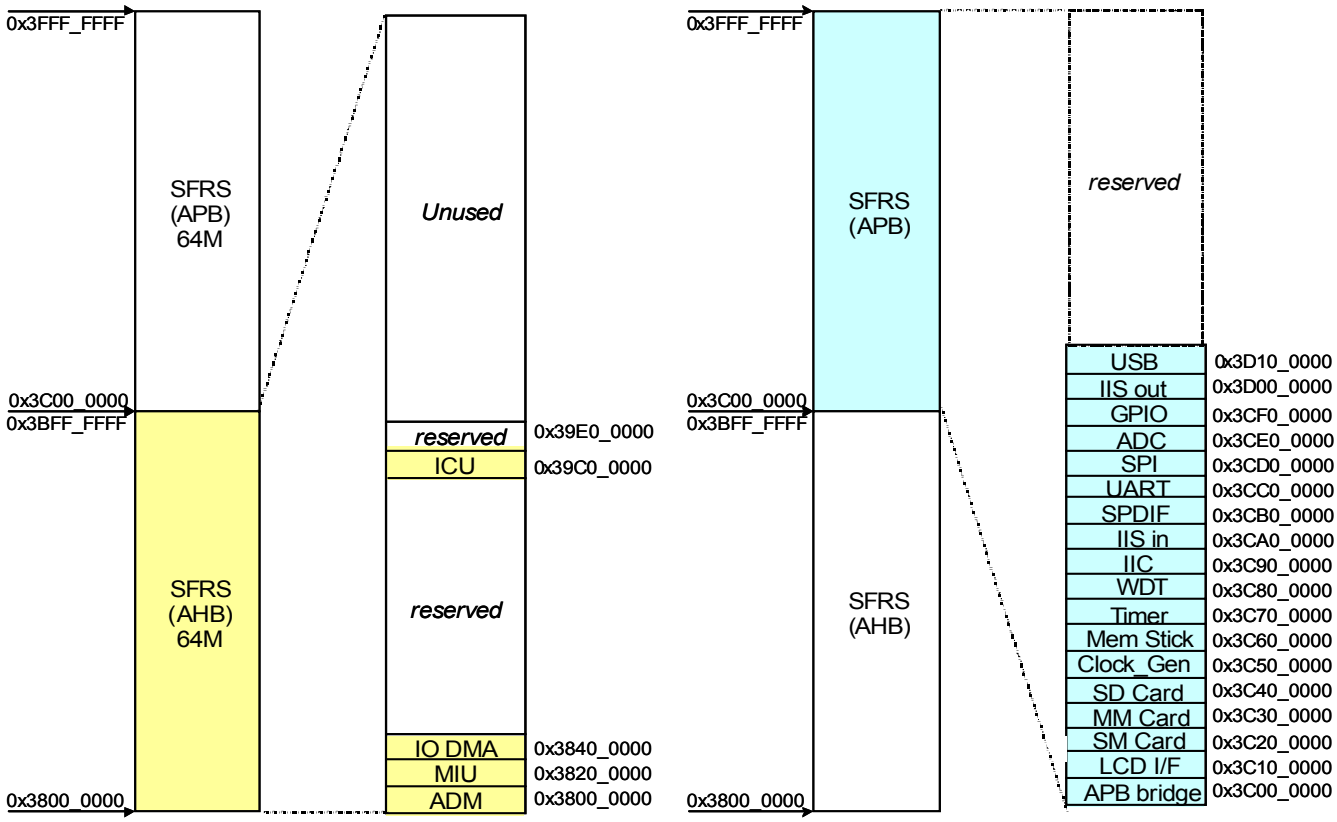
Memory configuration of S5L840FX

<Figure 1> shows overall address configuration in S5L840FX.

<Figure 2> shows memory mapped IO configuration of SFRS area in S5L840FX.



<Figure 1> Overall Address Configuration



<Figure 2> Memory Mapped IO Configuration

SRAM/ROM controller

Register map

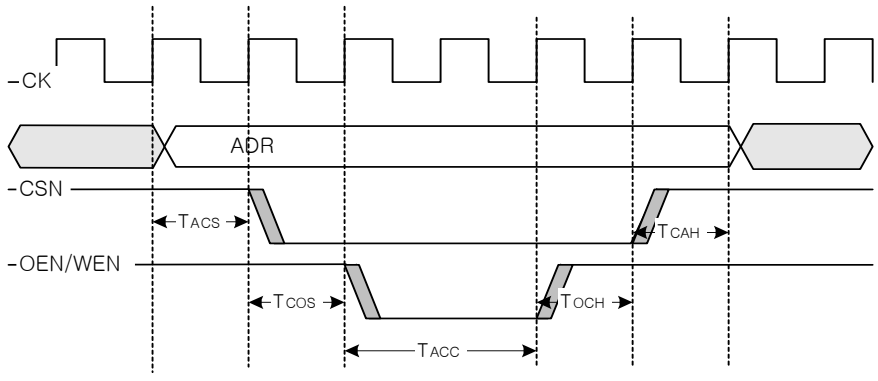
Static(ROM/SRAM) memory access parameter registers

Register	Address	R/W	Description	Width
MIURPARA	14~17H	R/W	ROM parameter register	32 bits
MIUSPARA	18~1BH	R/W	SRAM 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.	03h
[7:4]	T_{OCH}	Chip select hold time after output enable	1h
[3:0]	T_{CAH}	Address hold time after chip select	1h

* Recommend : T_{COS}, T_{OCH} cycle is recommended to keep up more than one cycle time. Otherwise, It makes a problem at memory read cycle if you adopt a synchronous SRAM.

*The Unit of MIURPARA and MIUSPARA fields is clock cycle.



Parameter Diagram

Static(ROM/SRAM) memory size parameter registers

Register	Address	R/W	Description	Width
MIUSSIZE	1C~1FH	W	Static memory size pre-setting	32 bits

Bits	Field	Description	Reset value
[31:24]	SIZE3	Static memory 3 size (reserved in S5L840FX)	00h
[23:16]	SIZE2	Static memory 2 size (SRAM area in S5L840FX)	80h
[15:8]	SIZE1	Static memory 1 size (NOR FLASH in S5L840FX)	40h
[7:0]	SIZE0	Static memory 0 size (ROM in S5L840FX)	01h

Parameter 1step equals 4Kbyte of memory size.

00h = reserved

01h = 4Kbyte size.

……

10h = 64Kbyte size

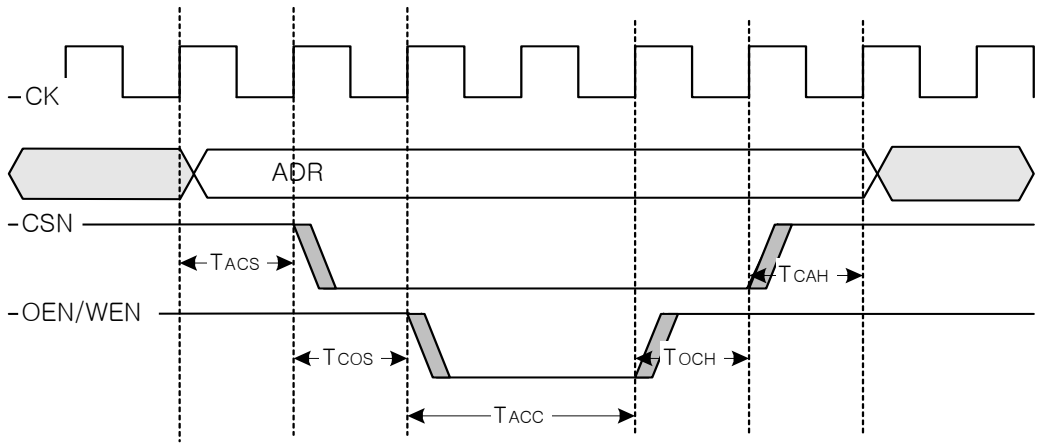
……

80h = 512Kbyte size.

memory size는 4Kbyte 의 정수배이며, 3Kbyte와 같은 경우는 4Kbyte로 가장 근접한 값을 세팅한다.

Access Parameter

SRAM/ROM controller, access cycle time parameter set.



If you use synchronous SRAM using the clock which same phase frequency with HCLK, you must keep up the T_{ACC} and T_{OCH} for more than one cycle time.

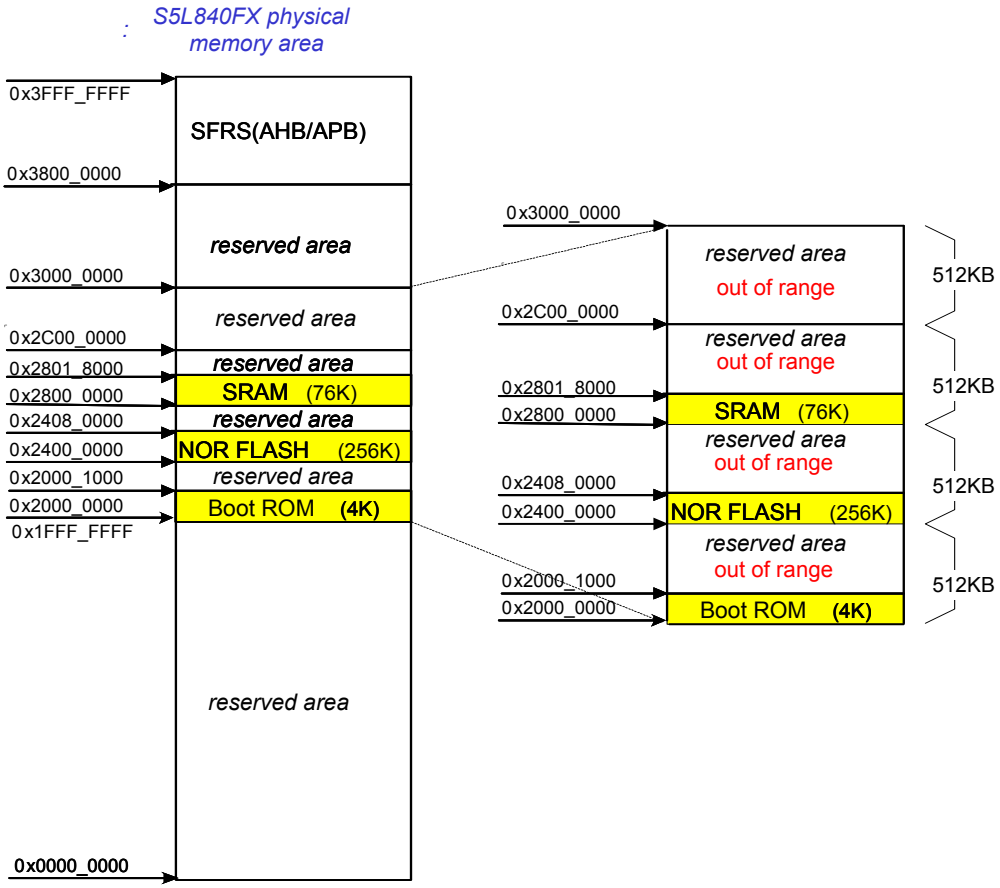
Minimum data phase cycle time of SRAM/ROM controller is 2cycle. First 1cycle is 'Chip select setup cycle time' and second 1cycle is 'Access cycle time'. 이때의 MIURPARA/MIUSPARA의 값은 32'h0001_0100이다. 2cycle access 의 경우에는 read 한 데이터를 F/F을 거치지 않은 데이터가 bus에 실린다. 만약 동작 주파수가 빠른 이유로 타이밍이 넉넉하지 않을 시에는 정상 동작이 불가능 할 수 있으므로 2cycle access는 피해야 한다. 이런 경우에는 Parameter 값을 32'h0001_0110(minimum value)으로 설정한다면 F/F을 거친 데이터가 bus에 실리게 되면 access 는 3cycle 만에 이루어지게 된다.

access parameter를 setting하는 assemble 의 예제는 다음과 같다.

Example. Access time parameter setting (3cycle access)

```
ld      A8, #rMIUBASE
//minimum cycle time parameter
ld      R0, #0001h
ld      R1, #0110h
ldw     @[A8+rMIUSPARA_H], R0
ldw     @[A8+rMIUSPARA_L], R1 //SRAM PARAMETER SET
ldw     @[A8+rMIURPARA_H], R0
ldw     @[A8+rMIURPARA_L], R1 //ROM PARAMETER SET.
```

Size parameter & Interrupt



S5L840FX out-of-range interrupt diagram from MIU

MIU의 out-of-range signal은 4개의 SRAM-like 영역에 대해서만 유효하다. user가 MIUSSIZE를 통해 설정한 값을 벗어난 메모리 영역을 access 하였을 때 toggle된다. 각 SRAM-like 영역의 size 는 MIUSSIZE register의 8비트씩에 할당되어 있으며, out-of-range 신호는 address 입력 신호와 register에 설정된 값을 비교함으로써 결정이 된다. 이 신호가 필요치 않을 시에는 maximum address size 인 8'h80(512KByte)을 MIUSSIZE의 해당 부분에 write해 준다. MIUSSIZE register에 write 할 때에는 32bit 전체가 update 되므로 기존 쓰여진 값이 유효할 경우에는 주의를 기울여야 한다. reset 후 MIUSSZE의 default 값은 32'h0080_8001이다.

9 IODMA CONTROLLER

OVERVIEW

S5L840F 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.

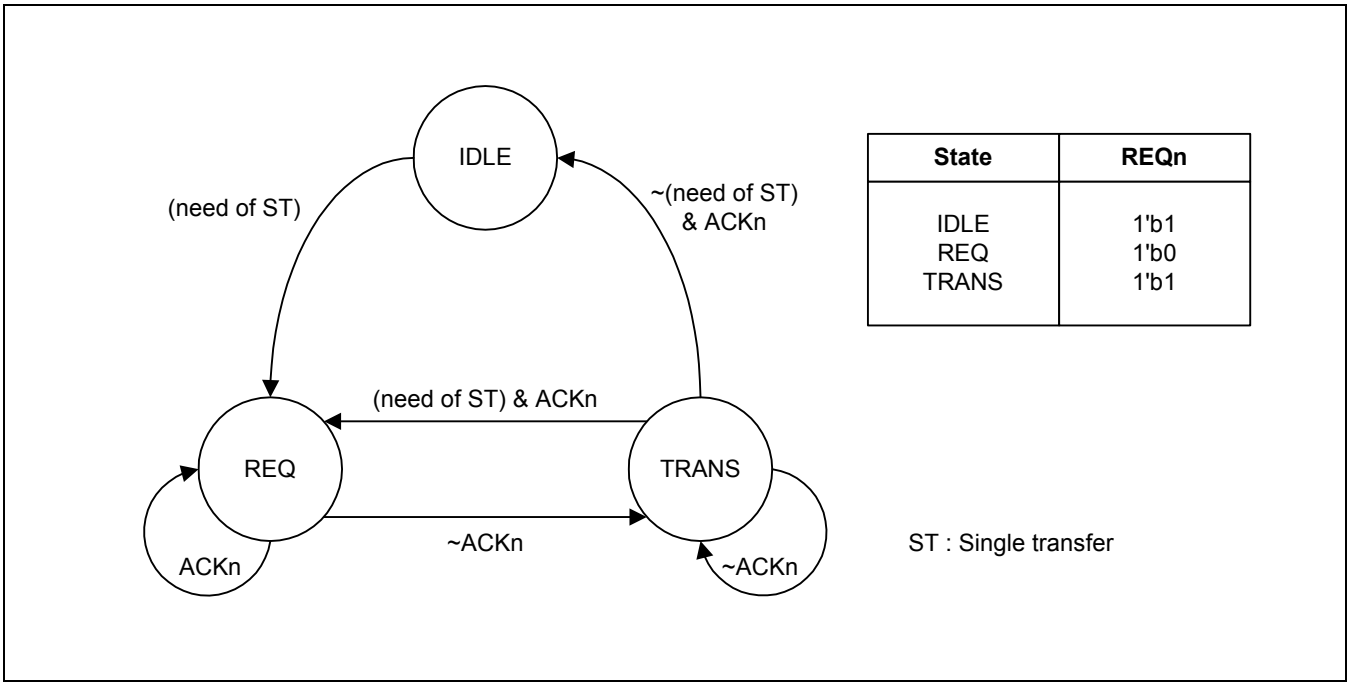
The DMA operation can be requested by either software or hardware including external DMA source.

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.



State	REQn
IDLE	1'b1
REQ	1'b0
TRANS	1'b1

Figure 9-1. 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.

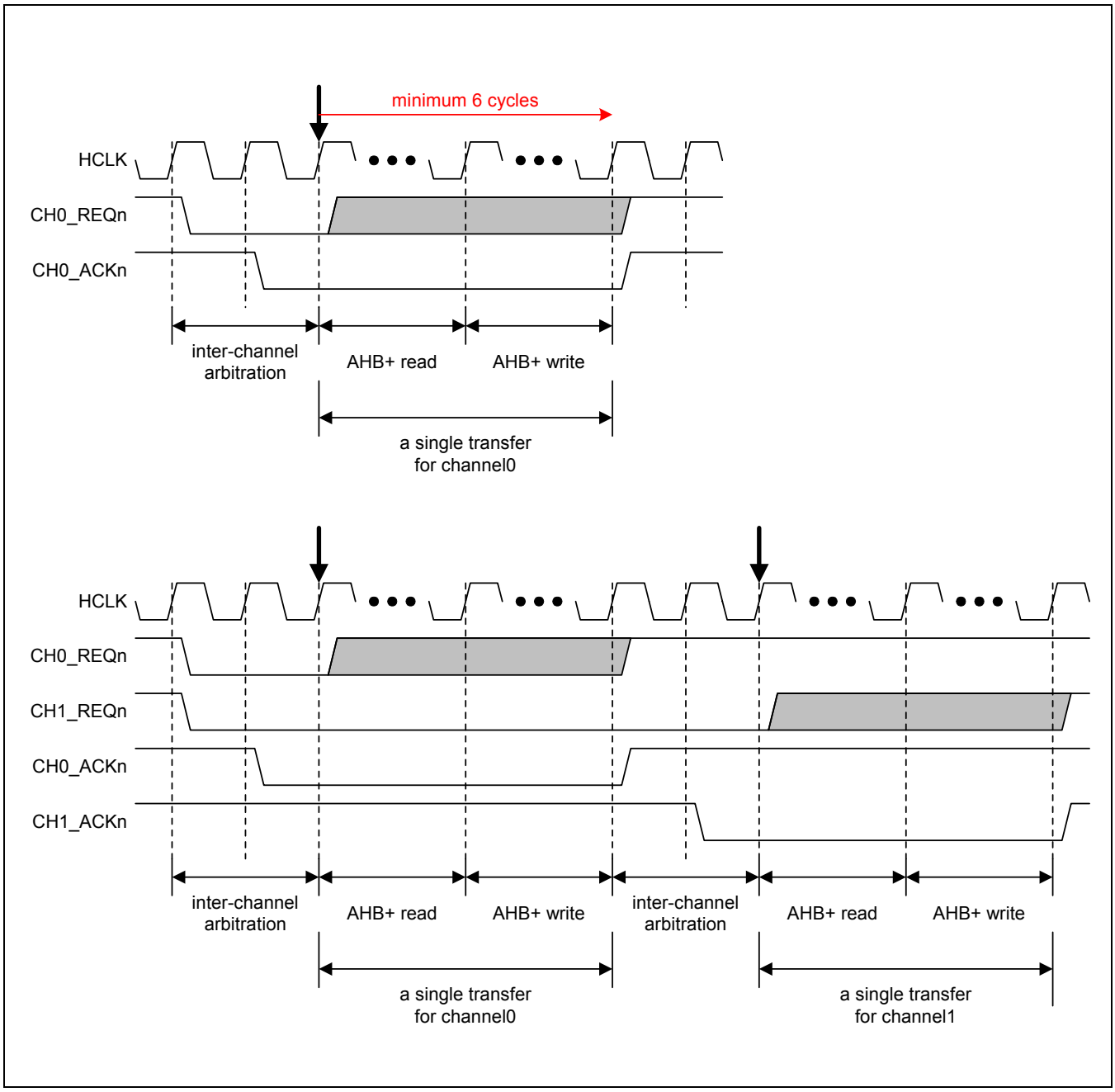


Figure 9-2. Single Data Transfer Timing

IODMA SPECIAL REGISTERS

BASE ADDRESS REGISTERS FOR EACH CHANNEL

Register	Address	R/W	Description	Width
DMABASE0	38400000H	R/W	Base address register for channel 0	32 bits
DMABASE1	38400020H	R/W	Base address register for channel 1	32 bits
DMABASE2	38400040H	R/W	Base address register for channel 2	32 bits
DMABASE3	38400060H	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	38400008H	R/W	Transfer count register for channel 0	32 bits
DMATCNT1	38400028H	R/W	Transfer count register for channel 1	32 bits
DMATCNT2	38400048H	R/W	Transfer count register for channel 2	32 bits
DMATCNT3	38400068H	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.	XXXXXh

CURRENT MEMORY ACCESS ADDRESS REGISTERS FOR EACH CHANNEL

Register	Address	R/W	Description	Width
DMACADDR0	3840000CH	R	Current memory address register for channel 0	32 bits
DMACADDR1	3840002CH	R	Current memory address register for channel 1	32 bits
DMACADDR2	3840004CH	R	Current memory address register for channel 2	32 bits
DMACADDR3	3840006CH	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	38400004H	R/W	Configuration register for channel 0	32 bits
DMACON1	38400024H	R/W	Configuration register for channel 1	32 bits
DMACON2	38400044H	R/W	Configuration register for channel 2	32 bits
DMACON3	38400064H	R/W	Configuration register for channel 3	32 bits

<table>
<tr><th>Bits</th><th>Field</th><th>Description</th><th>Reset Value</th></tr>
<tr><td>[31:30]</td><td>DEVSEL[1:0]</td><td>Device selection.
<table>
<tr><th>Ch \ Sel</th><th>00</th><th>01</th><th>10</th><th>11</th></tr>
<tr><td>0</td><td>IIS out</td><td>USB Ch1</td><td>LCD</td><td>UART0 out</td></tr>
<tr><td>1</td><td>LCD</td><td>SM</td><td>MS</td><td>SDC</td></tr>
<tr><td>2</td><td>IIS in</td><td>USB Ch2</td><td>SPI in</td><td>LCD</td></tr>
<tr><td>3</td><td>SPDIF</td><td>USB Ch3</td><td>SPI out</td><td>LCD</td></tr>
</table></td><td>XXb</td></tr>
<tr><td>[29]</td><td>DIR</td><td>DMA direction
0 : IO to Memory DMA, 1 : Memory to IO DMA</td><td>Xb</td></tr>
<tr><td>[28]</td><td>ADDRCON</td><td>Memory address control bit. This bit has to be “0”
This field indicates how to determine memory access address for next single transfer.
<table>
<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(ST[3]).</td></tr>
</table></td><td>Xb</td></tr>
</table>

(Continued) 0X2047_0000

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] >= 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 subchannel mode.	XXXXh

CURRENT TRANSFER COUNT REGISTERS FOR EACH CHANNEL

Register	Address	R/W	Description	Width
DMACTCNT0	38400010H	R	Current transfer count register for channel 0	32 bits
DMACTCNT1	38400030H	R	Current transfer count register for channel 1	32 bits
DMACTCNT2	38400050H	R	Current transfer count register for channel 2	32 bits
DMACTCNT3	38400070H	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	38400014H	W	Channel 0 command register	32 bits
DMACOM1	38400034H	W	Channel 1 command register	32 bits
DMACOM2	38400054H	W	Channel 2 command register	32 bits
DMACOM3	38400074H	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	38400018H	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-subchannel 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	38400100H	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.

Chapter 10. WDT module

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 in 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 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.

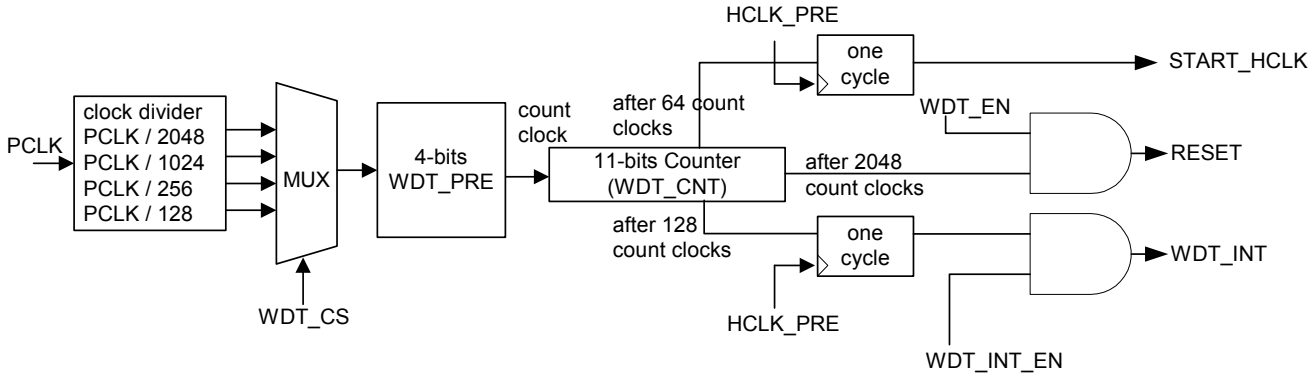


Figure 1. Watch-dog timer block diagram

PIN DESCRIPTION

Pin Name	Width	I/O	Description
clk	1	I	Global clock
reset_n	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
Control Signals			
st_mode	1	I	Scan test mode
wdt_stop	1	I	Stop the function of the watchdog timer
wdt_int	1	O	Basic timer interrupt
wdt_start	1	O	Basic timer start signal for oscillation stabilization
wdt_reset	1	O	Watch-dog timer reset signal
wdt_fast	1	I	Simulation option signal for wdt output signals
wdt_noreset	1	I	Simulation option signal for disabling wdt_reset
wdt_start_hclk	1	O	Hclk one cycle signal of wdt_start

clk
Main clock. The clock after the power-on is 27 MHz directed from the PLL. In normal operation, APB clock is supported.

reset_n
Global reset to reset the internal register

APB interface signals (psel, penable, pwrite, paddr, pwdata, prdata)
AMBA APB interface signals

st_mode
Scan test mode signal. In scan test mode, the internally generated or gated clocks are disabled and the main clock is used for all the internal flip-flops. This signal is used the selection signal for the clock muxing.

wdt_stop
Watchdog timer stop signal. In debugging mode, the watchdog timer should not operate as normal. The wdt_stop signal is used to temporarily stop the watchdog timer function.

wdt_int
Watchdog timer interrupt signal. When the interrupt operation is enabled, the interrupt signal is generated periodically. The time can be adjusted by the WDT_CS and WDT_PRE values.

wdt_start
Start signal. This signal will be long enough to stabilize the clock from the PLL after the power-on. After the stabilization of the clock, the reset signals of all the other modules are released and they start the normal operation.

wdt_reset

When the chip goes into abnormal state, it will not execute the normal operation. In this case, the normal clear operation for the watchdog timer may not be executed. When the watchdog timer is not cleared periodically, the reset signal is generated that will reset all the chip to go into the power-on state.

wdt_fast
For fast simulation, it makes wdt_start, wdt_int, wdt_reset to be generated at more short time than normal operations.

wdt_noreset
For long time simulation, it sets wdt global reset(wdt_reset) to zero

wdt_start_hclk
This signal is hclk_pre one cycle signal of wdt_start.

REGISTERS

Name	Width	Address(Virtual)	R/W	Description	Reset
WDTCON	32	0x3c80 0000(0x39 0000)	R/W	Control Register	0x0000 0000
WDTCNT	32	0x3c80 0000(0x39 0004)	R/W	11-bits internal counter	0x0000 0000

WDTCON

Name	Width	Address	R/W	Description	Reset
WDTCON	32	0x3c80 0000(0x39 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_CS			WDT_CLR				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 3'b100 PCLK / 2048 3'b001 PCLK / 1024 3'b010 PCLK / 256 3'b011 PCLK / 128 3'b000 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. 4'b1010 Clear the internal 11-bits counter register Others Nothing
7:0	WDT_EN	R/W	Enable the watch-dog timer 8'b10100101 Disable the reset/start signal generation by the watchdog timer. Others Nothing

WDTCNT

Name	Width	Address	R/W	Description	Reset
WDTCNT	32	0x3c80 0000(0x39 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

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 S5L840F 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 1, table 2, table 3 and table4.

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 75.85 us	13.18 KHz 75.85 us	0.21 KHz 4.85 ms	0.10 KHz 9.71 ms	6.44 Hz 155.34 ms
PCLK / 1024	26.37 KHz 37.93 us	26.37 KHz 37.93 us	0.41 KHz 2.43 ms	0.21 KHz 4.85 ms	12.87 Hz 77.67 ms
PCLK / 256	105.47 KHz 9.48 us	105.47 KHz 9.48 us	1.65 KHz 0.61 ms	0.82 KHz 1.21 ms	51.50 Hz 19.42 ms
PCLK / 128	210.94 KHz 4.74 us	210.94 KHz 4.74 us	3.30 KHz 0.30 ms	1.65 KHz 0.61 ms	103.00 Hz 9.71 ms
PCLK / 8	3.375MHz 296.3 ms	3,375MHz 296.3ms	52,73KHz 18.96us	26,37KHz 37.93us	1,65KHz 606.81us

Table 1. Pre-scaled clock frequency and periods when WDT_PRE = 0 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 2. Pre-scaled clock frequency and periods when WDT_PRE = 15 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	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

*PCLK/8 mode is default after reset

Table 3. Pre-scaled clock frequency and periods when WDT_PRE = 0 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 62.5ms	1Hz 1s	0.0156 Hz 64s	0.00781 Hz 128s	0.00048Hz 2048s
PCLK / 1024	32Hz 31.25ms	2Hz 0.5s	0.03125Hz 32s	0.0156 Hz 64s	0.000976Hz 1024s
PCLK / 256	128Hz 7.81ms	8Hz 125ms	0.125Hz 8s	0.0625Hz 16s	0.0039Hz 256s
PCLK / 128	256Hz 3.906ms	16Hz 62.5ms	0.25Hz 45s	0.125Hz 8s	0.00781Hz 128s
PCLK / 8	4096Hz 244.14us	256Hz 3.906ms	4Hz 0.25s	2Hz 0.5ms	0.125Hz 8s

Table 4. Pre-scaled clock frequency and periods when WDT_PRE = 15 and PCLK = 32,768Hz

Simulation Mode

We add simulation mode for fast and stable simulation and add two signals to wdt module. wdt_fatst, 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.

Chapter 11. RTC module

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 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
- Current Consumption: 3.5 μA (typical)
- Operation Temperature Range: 0 °C ~ 70 °C

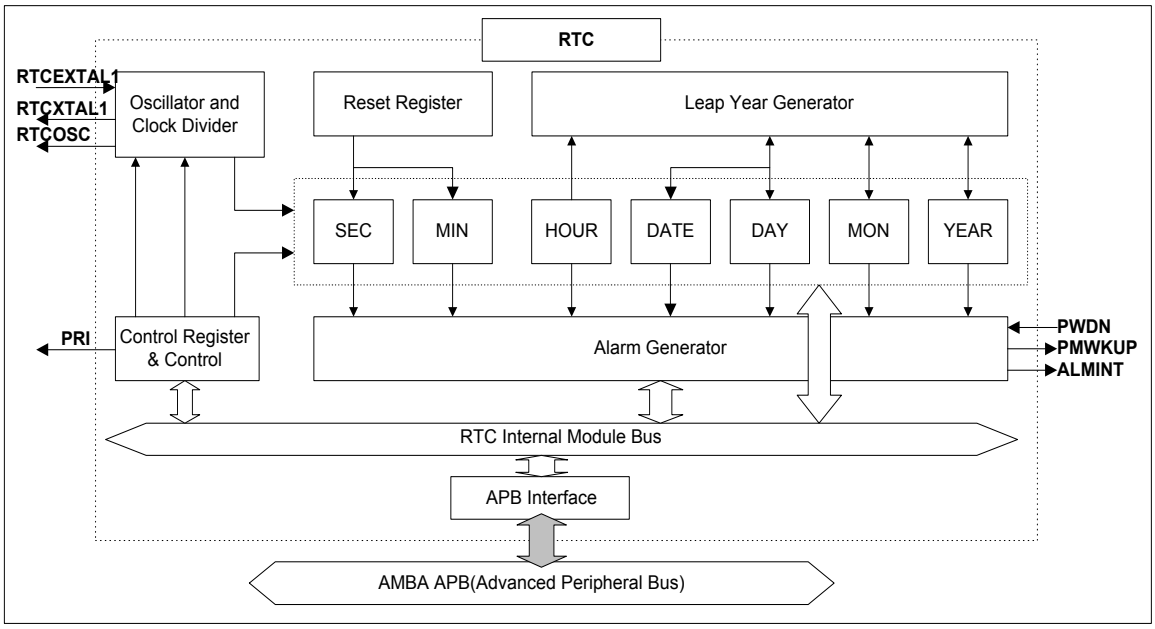


Figure 1. Top block diagram

Pin Description

The signals of Real Time Clock are divided into three sets. The first set includes the APB signals to configure the control registers. Another is to handshake with the Power Management block while in power down mode and relating to interrupt controller. The final is a related crystal signal that will be applied to off-chip crystal.

Table 1-1a. APB Interface Signals

Name	Type	Source/ Destination	Description
PCLK	In	AMBA APB	APB bus clock.
PRESETn	In	AMBA APB	Active low. Global reset signal.
PENABLE	In	AMBA APB	APB enable signal.
PSELPTC	In	AMBA APB	APB selection signal.
PWRITE	In	AMBA APB	APB read write signal.
PADDR[7:0]	In	AMBA APB	APB address bus.

Table 1-1b. APB Interface Signals

Name	Type	Source/ Destination	Description
PWDATA[15:0]	In	AMBA APB	APB write data bus.
PRDATA[15:0]	Out	AMBA APB	APB read data bus.

Table 1-2. Power Management and Interrupt Controller Interface Signals

Name	Type	Source/ Destination	Description
PWDN	In	Power Management	When this signal is high, RTC is operated by backup battery.
PMWKUP	Out	Power Management	This signal is to wake up the power management. While power down mode, this signal can be used to wake up power management unit.
ALMINT	Out	Interrupt Controller	Active high. When not power down mode, alarm enable bit in RTC was set and alarm time is matched by this signal that RTC requests a interrupt to interrupt controller.
PRI	Out	Interrupt Controller	Active high. This signal is to indicate 1/256 second, 1/64 second, 1/16 second, 1/4 second , 1/2 second, or 1 second.

Table 1-3. Crystal Interface Signals

Name	Type	Source/ Destination	Description
RTCEXTAL1	In	Clkgen Unit	Clock from clkgen Unit (i_rclk)

Table 1-4. Scan Test Interface Signals

Name	Type	Source/ Destination	Description
TMS	In	Testmode	RTC scan mode selection signal
SCAN_IN	In	Testmux	Scan serial input data
SCAN_EN	In	Testmux	Scan serial data enable
SCAN_OUT	Out	Testmux	Scan serial output data

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

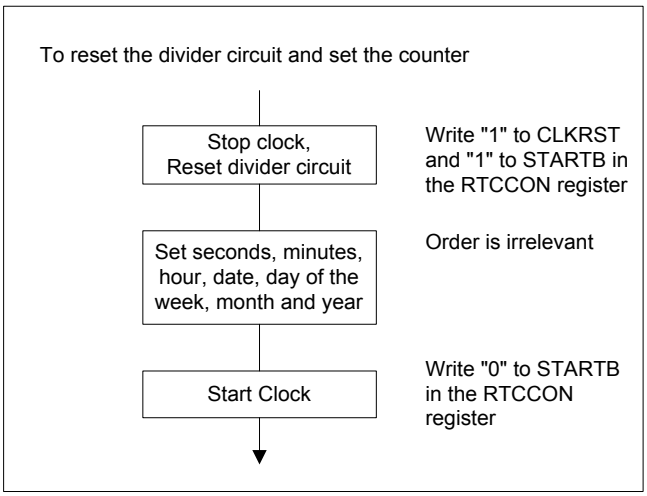


Figure 2. Setting the Time

Alarm Function

Figure 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 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.

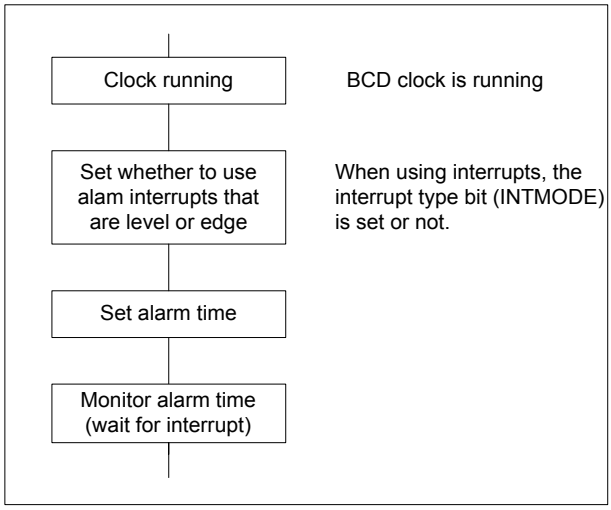


Figure 3. Using the Alarm Function

Programmer's Model

Register memory map

RTC_BASE : 0x 3D20_0000 (Virtual base address : 0x3A 4000)

Register	Address	R/W	Description	Reset value
RTCCON	Base + 0x00	R/W	RTC Control Register	0x00
RTCRST	Base + 0x04	R/W	RTC Round Reset Register	0x00
RTCALM	Base + 0x08	R/W	RTC Alarm Control Register	0x00
ALMSEC	Base + 0x0C	R/W	Alarm Second Data Register	0x00
ALMMIN	Base + 0x10	R/W	Alarm Minute Data Register	0x00
ALMHOUR	Base + 0x14	R/W	Alarm Hour Data Register	0x00
ALMDATE	Base + 0x18	R/W	Alarm Date Data Register	0x00
ALMDAY	Base + 0x1C	R/W	Alarm Day of Week Data Register	0x00
ALMMON	Base + 0x20	R/W	Alarm Month Data Register	0x00
ALMYEAR	Base + 0x24	R/W	Alarm Year Data Register	0x0000
BCDSEC	Base + 0x28	R/W	BCD Second Register	-
BCDMIN	Base + 0x2C	R/W	BCD Minute Register	-
BCDHOUR	Base + 0x30	R/W	BCD Hour Register	-
BCDDATE	Base + 0x34	R/W	BCD Date Register	-
BCDDAY	Base + 0x38	R/W	BCD Day of Week Register	-
BCDMON	Base + 0x3C	R/W	BCD Month Register	-
BCDYEAR	Base + 0x40	R/W	BCD Year Register	-
RTCIM	Base + 0x44	R/W	RTC Interrupt Mode Register	0x00
RTCPEND	Base + 0x48	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.

Table 1-6. RTC Control Register (RTCCON)

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.

Table 1-7a. RTC Alarm Control Register (RTCALM)

RTCCON	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

Table 1-7b. RTC Alarm Control Register (RTCALM)

RTCCON	Bit	Description	Initial State
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)

Table 1-8. 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)

Table 1-9. Alarm Minute Data Register (ALMMIN)

ALMMIN	Bit	Description	Initial State
MINDATA	[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 Hour Data Register (ALMHOUR)

Table 1-10. 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)

Table 1-11. 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)

Table 1-12. 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)

Table 1-13. 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)

Table 1-14. 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)

Table 1-15. 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)

Table 1-16. BCD Second Data Register (BCDSEC)

BCDSEC	Bit	Description	Initial State
SECDATA	[6:0]	BCD value for second bits	Undef.

		[3:0] bit is from 0 to 9 [6:4] bit is from 0 to 5	
Reserved	[7]		-

BCD Minute Data Register (BCDMIN)

Table 1-17. 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)

Table 1-18. 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)

Table 1-19. 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)

Table 1-20. 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)

Table 1-20. 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)

Table 1-21. 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.

Table 1-22. RTC Interrupt Mode Register (RTCIM)

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)

Table 1-23. 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

Timing Diagram

Interfacing RTC to APB

Interfacing the RTC to the APB is described in:

- Read transfer
- Write transfer

Figure 5. illustrates a read transfer.

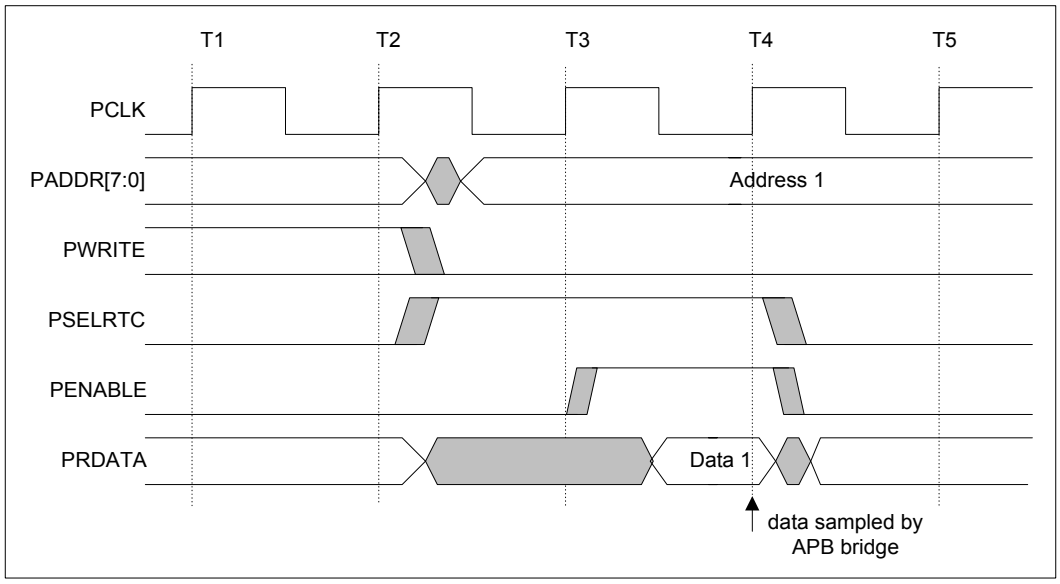


Figure 5. A Read Transfer

Figure 6. illustrates a write transfer.

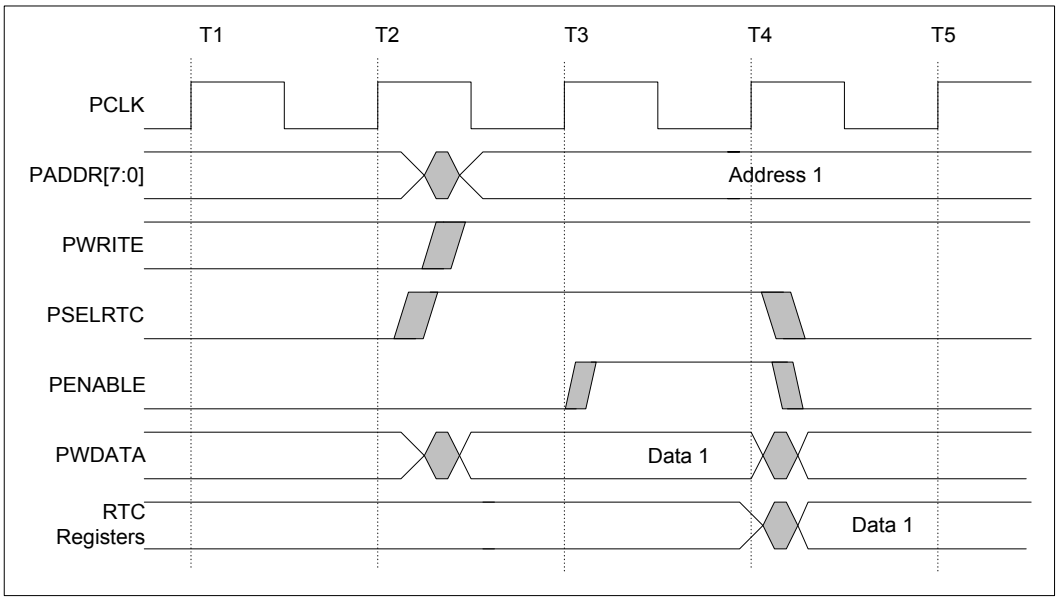


Figure 6. A Write Transfer

Chapter 12. Timer module

S5L840F has internally two timers. They are named as timer A, timer C. Timer A has the full function. However, timer C has a limited function that is derived from the full function of timer A. The supported modes of the full functional timer are interval mode, PWM (Pulse Width Modulation) mode, one-shot PWM mode, and capture mode.
The full functional 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. The timer with paritial funcion (Timer C) does not have the TM_DATA1 register because of its functional limit.

FEATURES

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

BLOCK DIAGRAM

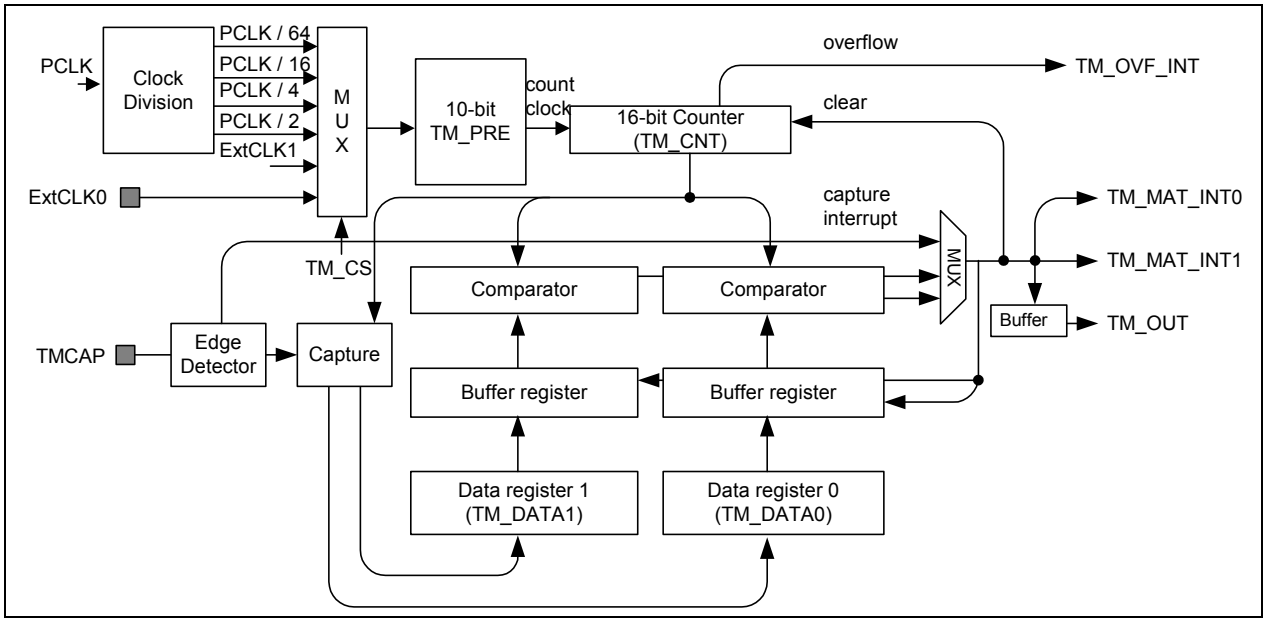


Figure 1. Block diagram for timer A

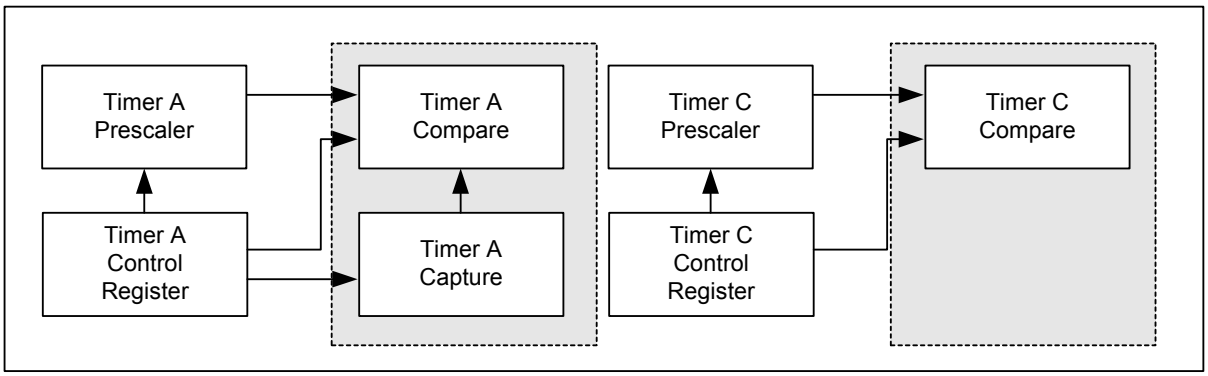


Figure 2. Timer Block Diagram

PIN DESCRIPTION

Pin Name	Width	I/O	Description
reset_n	1	I	Global reset
Clk	1	I	Global clock
Clk_ext0	4	I	External clock 0 for internal counting 0 External clock 0 for Timer A 1 External clock 0 for Timer B 2 External clock 0 for Timer C 3 External clock 0 for Timer D
Clk_ext1	4	I	External clock 1 for internal counting 0 External clock 1 for Timer A 1 External clock 1 for Timer B

			2 External clock 1 for Timer C 3 External clock 1 for Timer D
intr	4	O	Interrupt
APB Interface			
psel	1	I	APB selection signal
penable	1	I	APB enable signal
pwrite	1	I	APB write/read signal
paddr	7	I	APB address
pwdata	32	I	APB data input
prdata	32	O	APB data output
External Interface			
cap_port	2	I	External signal to be captured
pwm_out	2	O	PWM signal output pwm_out[0] Timer A PWM output pwm_out[1] Timer B PWM output

reset_n
Global reset to reset the internal register

clk
APB clock (main clock)

clk_ext0/1
Clk_ext0 and clk_ext1 are the external clock ports to count the internal counter. As the divided clock from the main clock mainly counts the internal counter, the counter value may not be accurate for some application. In that case, external accurate clock is used for more accurate counting operation. The positive edge of external clocks is internally detected by the main clock. So the external clock frequency should be lower than the main clock.

intr
When an interrupt condition occurs, an interrupt request is generated to the CPU core.

APB interface signals (psel, penable, pwrite, paddr, pwdata, prdata)
AMBA APB interface signals

cap_port
Positive edge or negative edge of external signals is detected and the time of these events are captured to the internal data registers. As these cap ports get external signals, they are digitally filtered in the internal capture block with 5 tabs. So glitches within 5 main clock periods in the cap port are digitally filtered out.

pwm_out
In interval mode and PWM mode, when the value of counter equals to that of data register, a match interrupt occurs and the pwm_out signal is toggled. This port is used to generate the toggling signal in interval mode or PWM signal in PWM mode.

REGISTER MAP

Name	Width	Address(Virtual)	R/W	Description	Reset
Timer A Registers					
TACON	32	0x3c70 0000(0x38 e000)	R/W	Control Register	0x0000 0000
TACMD	32	0x3c70 0004(0x38 e004)	R/W	Command Register	0x0000 0000
TADATA0	32	0x3c70 0008(0x38 e008)	R/W	Data0 Register	0x0000 0000
TADATA1	32	0x3c70 000c(0x38 e00c)	R/W	Data1 Register	0x0000 0000
TAPRE	32	0x3c70 0010(0x38 e010)	R/W	Prescale Register	0x0000 0000
TACNT	32	0x3c70 0014(0x38 e014)	R/W	Counter Register	0x0000 0000
Timer C Registers					
TCCON	32	0x3c70 0040(0x38 e040)	R/W	Control Register	0x0000 0000
TCCMD	32	0x3c70 0044(0x38 e044)	R/W	Command Register	0x0000 0000
TCDATA0	32	0x3c70 0048(0x38 e048)	R/W	Data0 Register	0x0000 0000
TCRE	32	0x3c70 0050(0x38 e050)	R/W	Prescale Register	0x0000 0000
TCCNT	32	0x3c70 0054(0x38 e054)	R/W	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
					TA_CS										

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 overflow interrupt 1 Enable the overflow interrupt
12	TA_INT0_EN	R/W	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. 0 Disable the overflow interrupt 1 Enable the overflow interrupt
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	Timer clock source selection 3'b000 PCLK / 2 3'b001 PCLK / 4 3'b010 PCLK / 16 3'b011 PCLK / 64 3'b10x External clock 0 3'b11x External clock 1
7	TA_CAP_MODE	R/W	In capture mode, 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
5:4	TA_MODE_SEL	R/W	Operation mode selection 2'b00 Interval mode 2'b01 PWM mode 2'b10 One-shot mode 2'b11 Capture mode

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	Clear operation. This field is always zero when read. 0 Nothing occurs 1 Initialize the timer. - 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	Timer enable command 0 Disable the timer 1 Enable the timer

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 mode/ 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_INT1 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 mode/ 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

			Capture mode	generated. This field is updated to the internal data buffer 1 when MAT_INT1 occurs or when clear operation is executed. 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 C REGISTERS

TCCON

Name	Width	Address	R/W	Description	Reset
TCCON	32	0x3c70 0040	R/W	Control Register for timer C	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
					TC_CS										

Bits	Name	Type	Description
18	Reserved	-	This flag should be set to 0.
16	TC_INT0	R/W	Match interrupt 0 status - Writing one clears this flag - Writing zero has no effect
14	Reserved	-	This flag should be set to 0
12	TC_INT0_EN	R/W	Enable the match interrupt 0. When disabled, match interrupt 0 sets the TC_INT0 but it does not generate an interrupt request signal to external interrupt control unit. 0 Disable the overflow interrupt 1 Enable the overflow interrupt
10:8	TC_CS	R/W	Timer clock source selection 3'b000 PCLK / 2 3'b001 PCLK / 4 3'b010 PCLK / 16 3'b011 PCLK / 64 3'b10x External clock 0 3'b11x External clock 1

Rising clear
Clear the cou

TCCMD

Name	Width	Address	R/W	Description	Reset
TCCMD	32	0x3c70 0044	R/W	Command Register for timer C	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	TC_CLR	W	Initialize the timer 0 Nothing occurs 1 Initialize the timer. - Clear the counter register. - TC_DATA0 is updated to the internal buffers - Initialize the state of the previously captured signal.
0	TC_EN	R/W	Timer enable command 0 Disable the timer 1 Enable the timer

TCDATA0

Name	Width	Address	R/W	Description	Reset
TCDATA0	32	0x3c70 0048	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
TC_DATA0															

Bits	Name	Type	Description
15:0	TC_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 for next comparison.

TCPRE

Name	Width	Address	R/W	Description	Reset
TCPRE	32	0x3c70 0050	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
						TC_PRE									

Bits	Name	Type	Description
9:0	TC_PRE	R/W	Pre-scale value

TCCNT

Name	Width	Address	R/W	Description	Reset
TCCNT	32	0x3c70 0054	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
TC_CNT															

Bits	Name	Type	Description
15:0	TC_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 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.

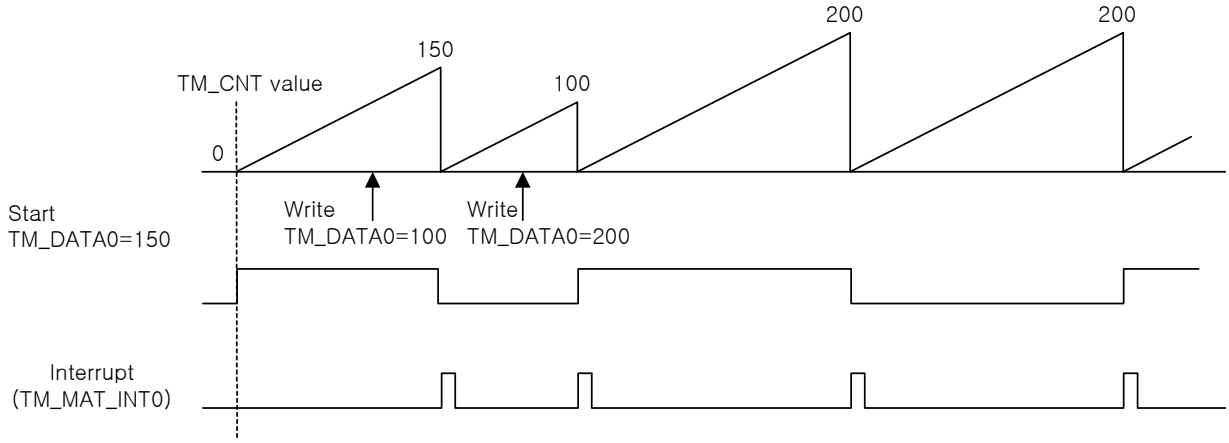


Fig 2. Interval mode operation

The detailed waveform is shown in Figure 3.

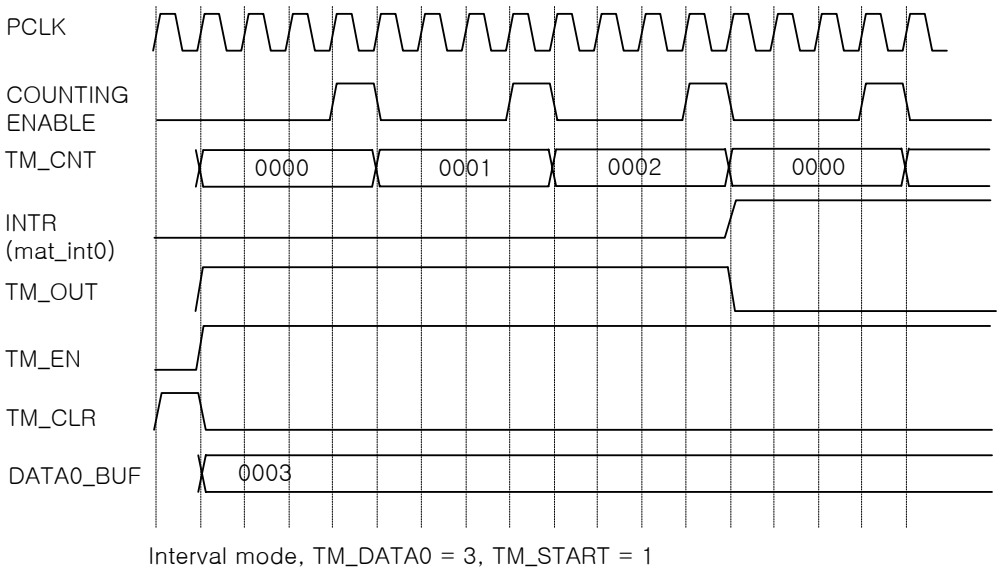


Figure 3. Waveform in interval mode

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 theTM_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 4.

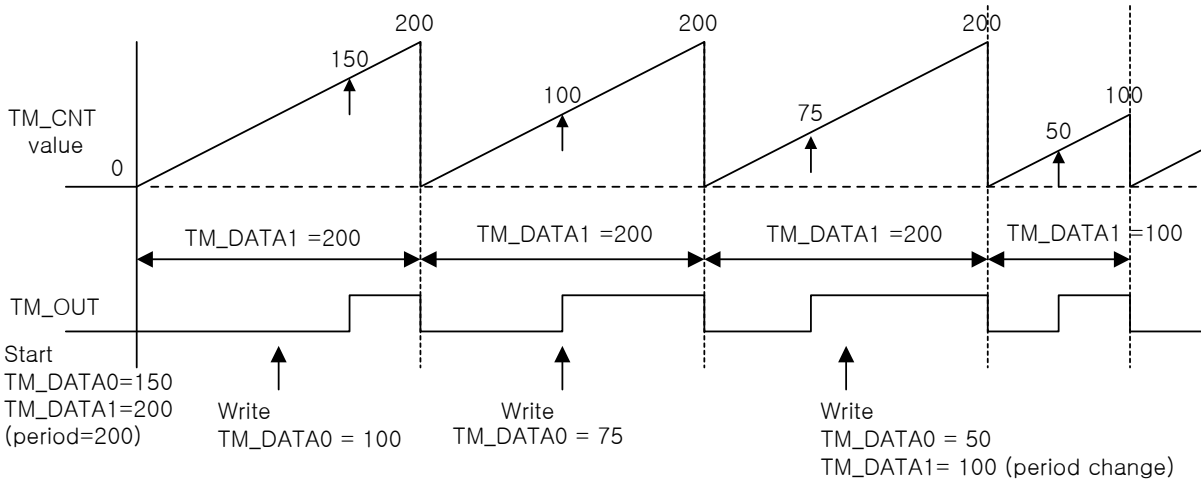


Figure 4. PWM mode operation

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 6, Figure 7, Figure 8, and Figure 9.

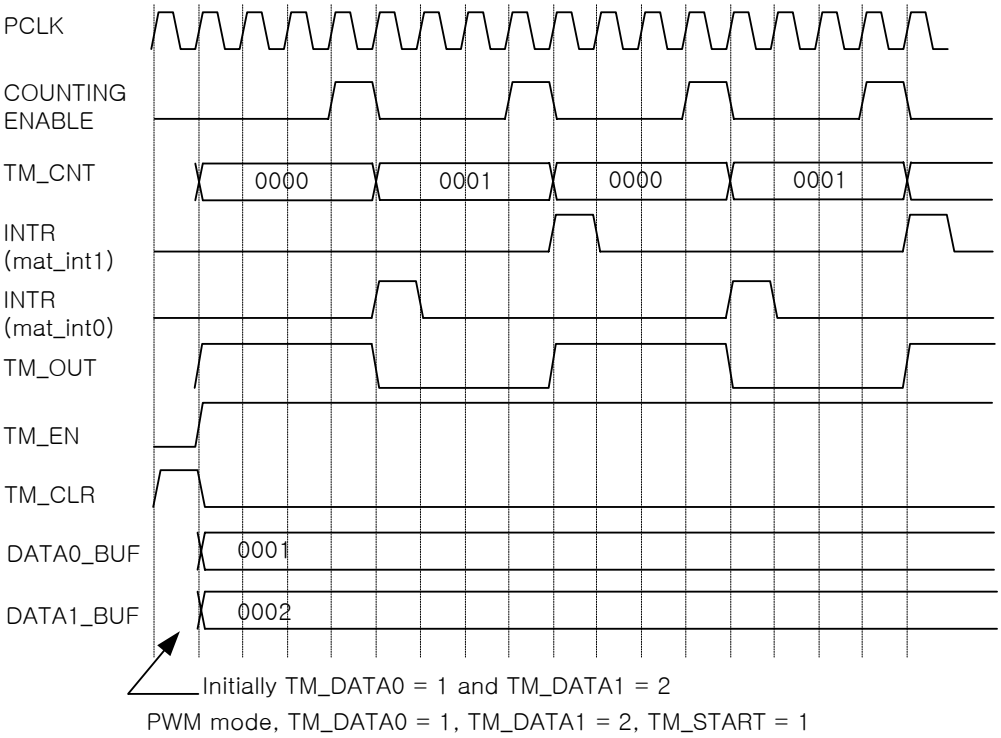


Figure 6. PWM signal when TM_DATA0 = 1 and TM_DATA1 = 2

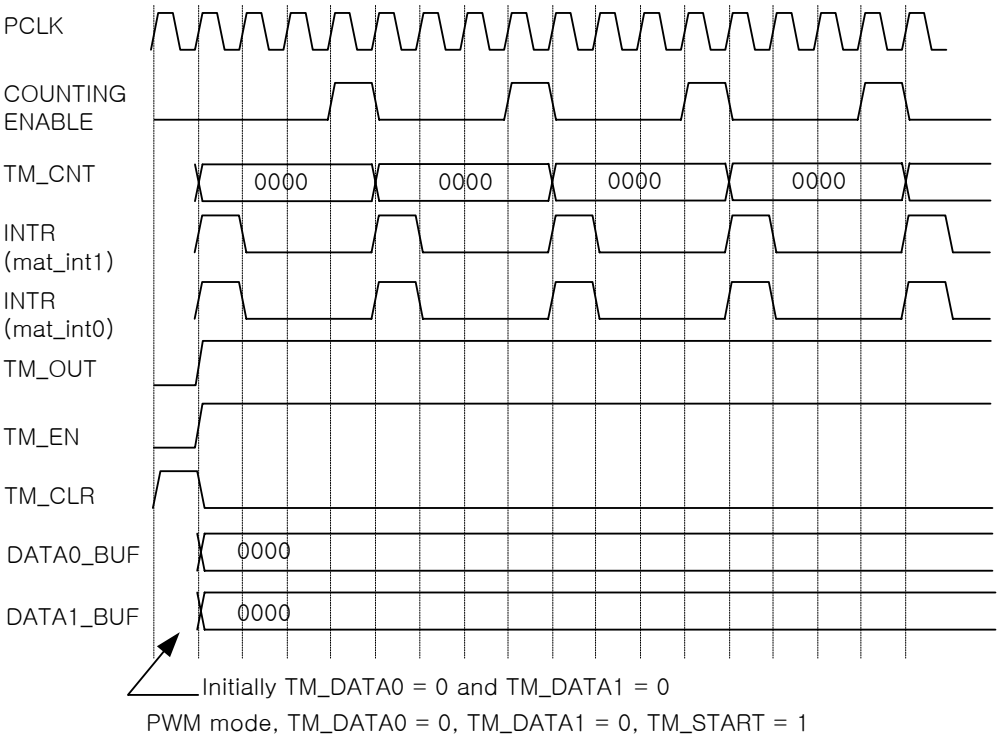


Figure 7. PWM signal when TM_DATA0 = 0 and TM_DATA1 = 0

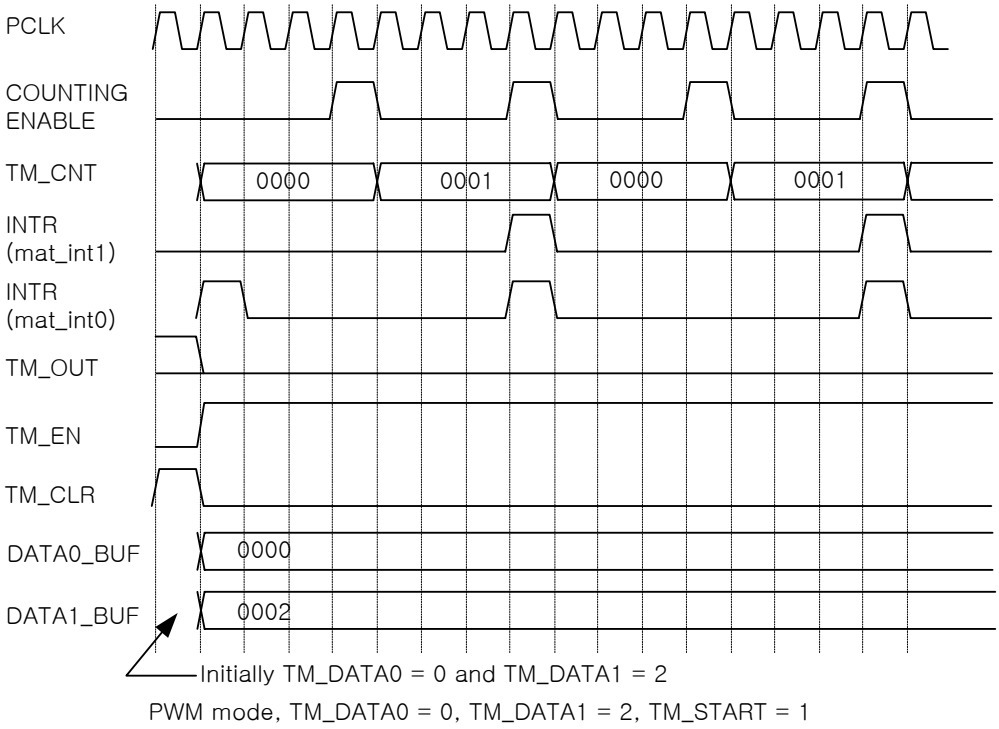


Figure 8. PWM signal when TM_DATA0 = 0 and TM_DATA1 = 2 (Duty Ratio = 0 %)

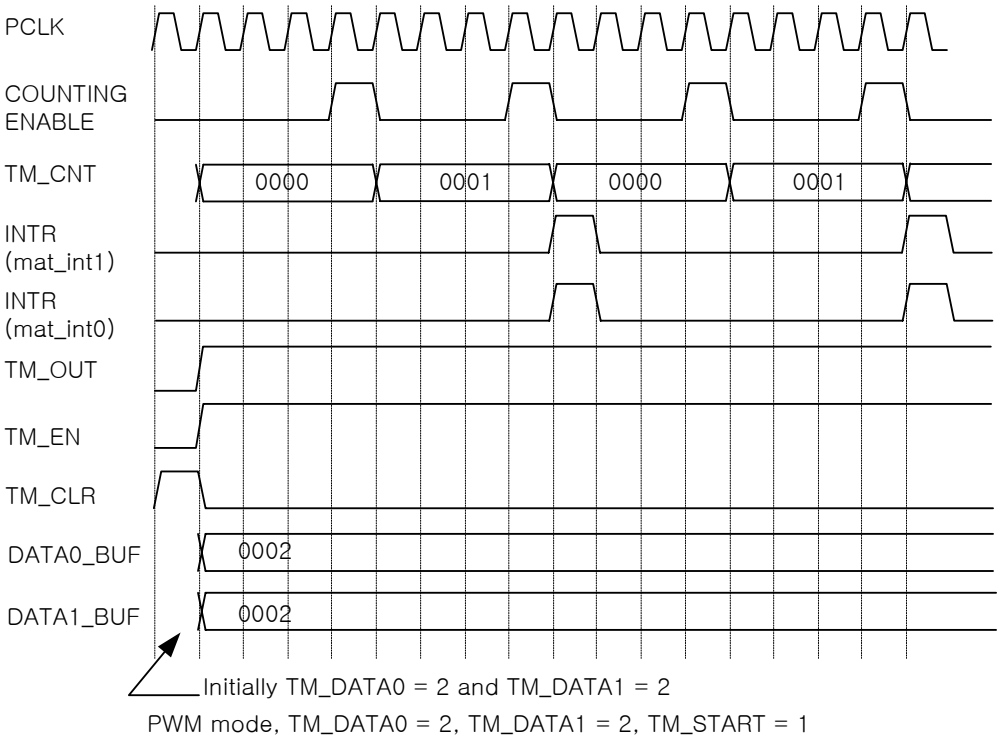


Figure 9. PWM signal when TM_DATA0 = 2 and TM_DATA1 = 2 (Duty Ratio = 100 %)

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 10.

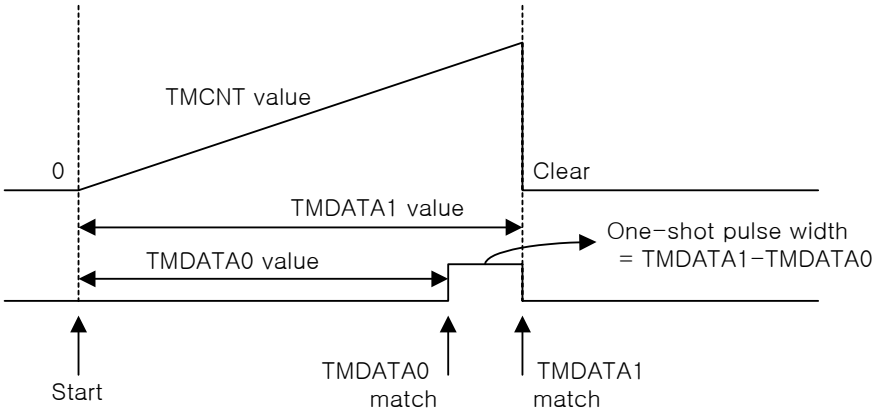


Figure 10. One-shot mode operation

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 Fig 8.

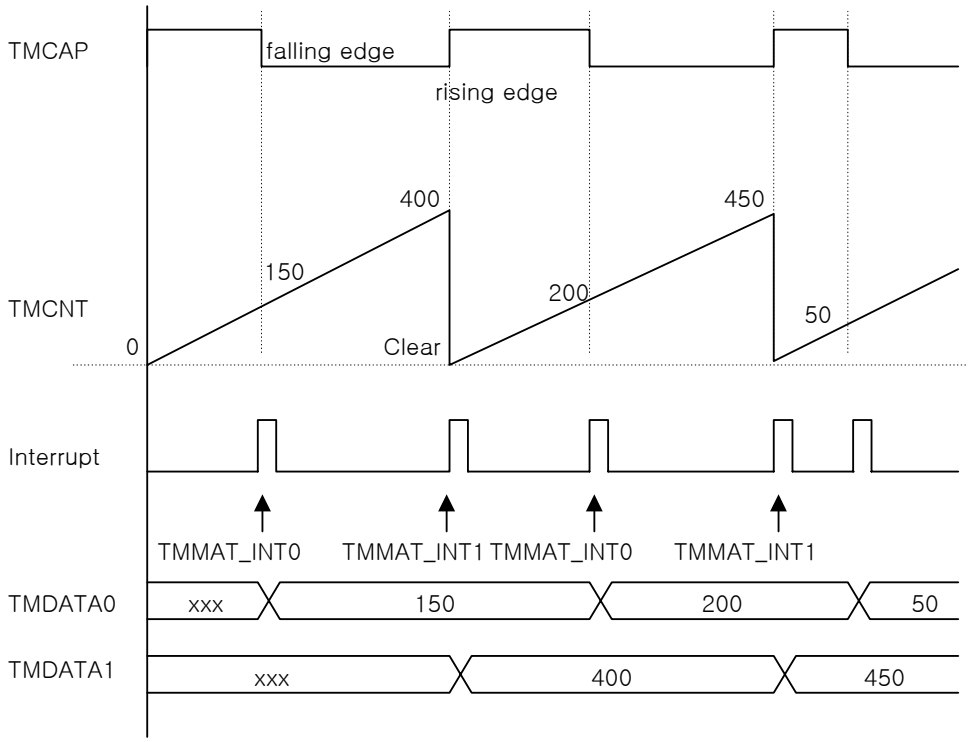


Fig 8. Capture mode in rising edge clear mode

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:

Scaled clock [Hz] = Source clock [Hz] / (TM_PRE + 1)

The source clock is the selected clock by the TM_CS. Table 1 and Table 2 shows the frequency and the period for each clock selection. It assumes that the main clock is 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 526.75 ns	1.90 MHz 526.75 ns	28.97 Hz 34.52 ms	1.85 KHz 539.39 ms	0.03 Hz 35.35 s
Main Clock / 16	7.59 MHz 131.69 ns	7.59 MHz 131.69 ns	115.87 Hz 8.63 ms	7.42 KHz 134.85 ms	0.11 Hz 8.84 s
Main Clock / 4	30.38 MHz 32.92 ns	30.38 MHz 32.92 ns	463.49 Hz 2.16 ms	29.66 KHz 33.71 ms	0.45 Hz 2.21 s
Main Clock / 2	60.75 MHz 16.46 ns	60.75 MHz 16.46 ns	926.97 Hz 1.08 ms	59.33 KHz 16.86 ms	0.91 Hz 1.10 s

Table 1. Pre-scaled clock frequency and period when main clock = 121.5 MHz

NAND Flash Memory Controller

OVERVIEW

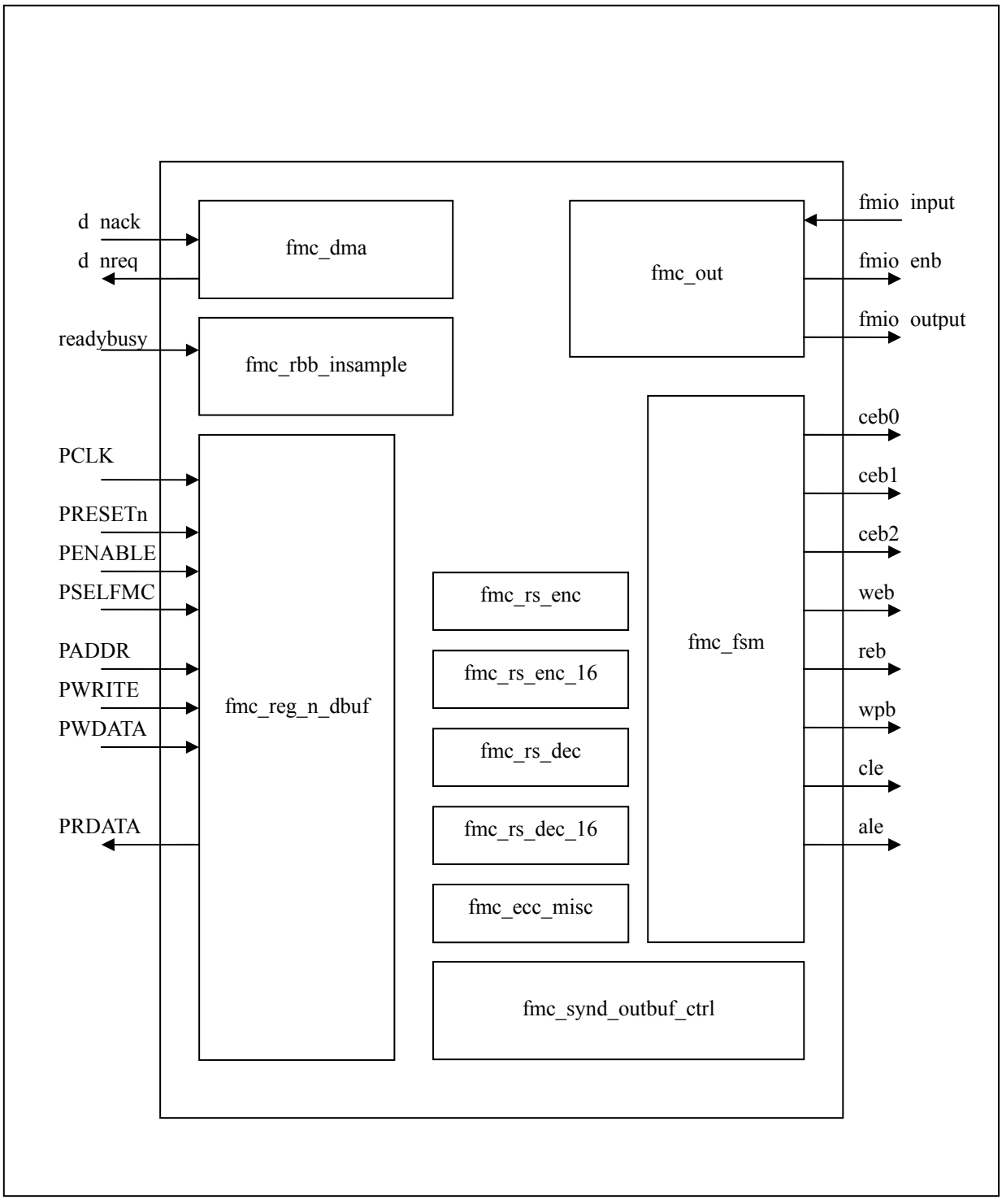
This module can control the interface of the external NAND Flash memory.

Features

Flash Memory Controller(FMC) supports the following function.

- supports 64/128/256/512Mbit,1G, 2G,4Gbit NAND flash memory made by Samsung.
- can be connected up to 3 flash memory.
- supports only one flash memory at a time.
- ECC supported
- For ECC, the parity bit encoding and decoding

Block Diagram



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
	web	Write enable
	reb	Read enable
	~~seb~~	~~Spare area enable (deleted)~~
	wpb	Wrote protect
	cle	Command latch enable
	ale	Address latch enable
DMA	D_nack	DMA acknowledge
	D_nreq	DMA request

Function description

OVERVIEW

This FMC support 64Mb, 128Mb, 256Mb, 512Mb, 1Gb, 2Gb and 4Gb NAND flash memory which are made by SAMSUNG. User can do that FMC is enabled or disabled using control register in FMC. The ECC scheme used in this FMC is RS-CODE.

The main features are :

- NAND Flash Controller
  - Support 64Mb/128Mb/256Mb/512Mb/1Gb/2Gb/4Gb NAND Flash components
  - Support 8bit or 16bit interface.
- Embedded 1-symbol ECC (RS-CODE) Encoder/Decoder
- There are two ways for data transfer, by CPU or DMA.

.

Flash Memory Controller Operation description

After reset is active, FMC will be disabled.
All of registers are cleared in this state. FMC is enabled or disabled by control register.
The operation sequence is the following :

1) Read the ID register for getting the information of the connected external Flash memory. From the content of ID register, User can also get the version of FM.
2) Set the density register and control register.
   The signal named by fmc_en in control register is the signal for FMC to be enabled or disabled. After this signal is toggled from low to high, the density register has to be set again. In case of ECC mode, after write operation, the syndrome data generated has to be saved for read operation. In case that read operation is executed in ECC mode, User has to write the syndrome data in syndrome register prior to reading.
3) User can get the status of the current operation by reading the status register.

Flash Memory Controller register

ADDRESS	R/W	REGISTER NAME	
Base address + 0x00	R/W	Command Register	FMCMD
Base address + 0x04	R/W	Column Address Register1	FMCADDR1
Base address + 0x08	R/W	Column Address Register2	FMCADDR2
Base address + 0x0C	R/W	Row Address Register1	FMRADDR1
Base address + 0x10	R/W	Row Address Register2	FMRADDR2
Base address + 0x14	R/W	Row Address Register3	FMRADDR3
Base address + 0x18	R	Flash Memory Status Register (CMD = 70h)	FMRSTATUS
Base address + 0x1C	R	Flash Memory ID Register0 (CMD = 90h)	FMRID0
Base address + 0x20	R	Flash Memory ID Register1 (CMD = 90h)	FMRID1
Base address + 0x24	R	Flash Memory ID Register2 (CMD = 90h)	FMRID2
Base address + 0x28	R	Flash Memory ID Register3 (CMD = 90h)	FMRID3
Base address + 0x2C	R	Flash Memory ID Register4 (CMD = 90h)	FMRID4
Base address + 0x30	R/W	Control Register	FMCTRL
Base address + 0x34	R/W	ECC Decoder Control Register	FMCDECTRL
Base address + 0x38	R	Controller Status Register	FMCSTATUS
Base address + 0x3C	R	ECC Decoder Status Register	FMCDECSTATUS
Base address + 0x40	R/W	Data Register0 ~ 7	FMDATARDWR0 ~ 7
Base address + 0x60	R/W	Syndrome data0 generated at write operation	FMSYNDOUTRW0
Base address + 0x64	R/W	Syndrome data1 generated at write operation	FMSYNDOUTRW1
Base address + 0x68	R/W	Syndrome data2 generated at write operation	FMSYNDOUTRW2
Base address + 0x6C	R/W	Syndrome data3 generated at write operation	FMSYNDOUTRW3
Base address + 0x80	R/W	Syndrome data0 used for read operation	FMSYNDINRW0
Base address + 0x84	R/W	Syndrome data1 used for read operation	FMSYNDINRW1
Base address + 0x88	R/W	Syndrome data2 used for read operation	FMSYNDINRW2
Base address + 0x8C	R/W	Syndrome data3 used for read operation	FMSYNDINRW3
Base address + 0x90	R/W	Syndrome data0 used for read operation	FMDEC_RESULT0
Base address + 0x94	R/W	Syndrome data1 used for read operation	FMDEC_RESULT1
Base address + 0x98	R/W	Syndrome data2 used for read operation	FMDEC_RESULT2
Base address + 0x9C	R/W	Syndrome data3 used for read operation	FMDEC_RESULT3
Base address + 0xA0	R/W	Density of Flash that will be used	FMDENSITY
Base address + 0xA4	R/W	FSM of Controller	FMTEST

Flash Memory Controller Register Descriptions

FMCMD

Initial value is 0x00.

The command will be sent to FM is written in this register.

Bit[15:8]	Bit[7:0]
Not used	Command

Address_n Register (FMCADDRn/FMRADDRn)

Initial valures are 0x00.

There are 5 address registers. The column address is written in FMCADDRn and the row address is written in FMRADDRn as to the density type of external Flash Memory.

(Ex : 512Mb Read : FMCADDR1→ FMRADDR1 → FMRADDR2 → FMRADDR3

512Mb Erase : FMRADDR1 → FMRADDR2 → FMRADDR3)

Bit[15:8]	Bit[7:0]
Not used	Address

External Flash Memory Status Register (FMRSTATUS)

Initial value is 0x00.

After writing 0x70 command(READ STATUS) to FM, FM outputs the contents of FM status. Those content are written in FMRSTATUS register. User can find out whether program or erase operation is completed, and whether the program or erase operation is completed successfully.

Bit[15:8]	Bit[7:0]
Not used	Status of Flash

External Flash Memory IDn Register (FMRIDn)

Initial value is 0x00.

After writing 0x90 command (READ ID) and an address input, FM(flash memory) send the product identification. Those information is written in FMRIDn register.

Bit[15:8]	Bit[7:0]
Not used	Flash ID

Control Register (FMCTRL)

BIT	OPERATION	FUNCTION
0 (ceb1 enable)	0 = disable 1 = enable	The ceb1 is enabled when need to Flash1. The inverse value of this bit outputs to external Flash Memoy/
1 (fmc_en)	0 = disable 1 = enable	User can initiate all of registers in FMC.
2		reserved
3 (random_cben)	0 = disable 1 = enable	Select whether to use the Copy-back with random data input function
4.(copybacken)	0 = disable 1 = enable	Select whether to use the Copy-back function
5 (ecc_en)	0 = disable 1 = enable	Select whether to use the ECC function
6		reserved
7 (nWP)	0 = enable 1 = disable	nWP must be high in write mode and don't care in read mode
8 (ceb2 enable)	0 = disable 1 = enable	The ceb2 is enabled when need to Flash2. The inverse value of this bit outputs to external Flash Memoy/
9(ceb0 enable)	0 = disable 1 = enable	The ceb0 is enabled when need to Flash0. The inverse value of this bit outputs to external Flash Memoy/
10 (little endian)	0 = little endian 1 = big endian	Select whether the data transfer type is little endian or big endian.
11(version _inf)	0 = single die Flash 1 = Double die Flash	Select whether the memoy type is single die or double die.
14:12 (sig_wait)	Ex) PCLK 100MHz = 4 (sig_wait) 50MHz = 2 (sig_wait) 30MHz = 0 (sig_wait)	Flash control signal hold the current level for (sig_wait +1) PCLK
15 (data_width)	0 : flash IO = x8 1: Flash IO = x16	Select whether the data bit width transferred is 8bit or 16bit.

ECC Control Register (FMCDECTRL)

The initial value is 0x0000.

After ECC function is enabled, the 4bit named dec_synd_flagN have to be enabled to do read operation with ECC before the contents of syndrome input buffer is changed.

The description of each bit is as following:

BIT	OPERATION	FUNCTION
0 (dec_synd_flag0) 1 (dec_synd_flag0) 2 (dec_synd_flag0) 3 (dec_synd_flag0)	0 = Disable ECC decoder when read operation 1 = Enable ECC decoder when read operation	Select whether to use the ECC decoder. Each bit is used for being enable the syndrome data , bit0 is used for 1st 128byte data, bit1 is used for 2nd 128byte data, bit3 is used for 3rd 128 byte data bit4 is used for 4th 128byte data
4 rd_end	Read stop	While the read operation is executed with ECC, if this bit is high, read operation will be stopped after decode is finished.
5 (clr_inbuf_wr_ptr)	0 = don't care 1 = clear the pointer to 3'b000	When user don't read data by 512byte, this bit must be used before read operation
6 dmaen	0 = disable 1 = enable	
7		Reserved
8		Reserved
15:8		Not used

FMC Status Register (FMCSTATUS)

The initial value is 0x00.

User has to check this register while the flow of each program.

BIT	OPERATION	FUNCTION
0 (readybusyb)	0 = Busy 1 = Ready	The status of current operation in FM is busy or ready.
1		reserved
2 (rdid_end_flag)	1 = Flash ID read end	After Flash ID command, all Flash ID bytes are read from Flash
3		Reserved
4		Reserved
5 (status_data_set)	0 = not ready 1= ready	When the status data transfers to **FMRSTATUS** by 70h command, this bit is set. After cpu read **FMRSTATUS,** this bit will be cleared
6 (last_column)	0 = not last column 1 = last column	When '1' is setting, it indicates the last column at that page.
7 (add_send_end)	1 = address-send-end	Indicate that all address bytes are send to Flash as to command.
8(outbuf_rdend)	1=output buffer empty	Indicates Data Buffer(FIFO) are empty during write operation, CPU should write 8 data(width 16) if this bit is high
9(inbuf0_ready)	1=input buffer full	Indicates the first FIFO are full when read operation. CPU can read 8 data
10(inbuf1_ready)	1=input buffer full	Indicates the second FIFO are full when read operation. CPU can read 8 data
11		Reserved
12		Reserved
15:13(column_cnt)		Column_cnt[2:0] While read or write operation is doing, This counter indicate the data number transferred.

ECC Status Register (FMCDECSTATUS)

The initial value is 0x0000..

BIT		FUNCTION
0 (dec_no_error0) 1 (dec_no_error0) 2 (dec_no_error0) 3 (dec_no_error0)	0 = error 1 = no error	Each bit is used for indicating whether to occur the ECC decoding error. The bit0 indicates the occurrence of the error for 1st 128byte data, bit1 for 2nd 128byte data, bit3 for 3rd 128 byte data and bit4 for 4th 128byte data
4 (dec_end0) 5 (dec_end0) 6 (dec_end0) 7 (dec_end0)	0 = progressing 1 = decoding end	Each bit is used for indicating whether to end ECC decoding operation, bit4 for 1st 128byte data, bit5 for 2nd 128byte data, bit6 for 3rd 128 byte data and bit7 for 4th 128byte data
15:8		Not used

Data Register (FMDATARDWRn)

The initial value is 0x00. The width is 16bit. There are 8 registers.

These registers are used to store data transmitted or received.

CPU have to write the data when the bit[8] of thisregisteris high in write mode.

In read mode, CPU have to read the data from this register after checking that the bit[9] or bit[10] is high. At first, User check the bit[10[after checking bit[9]. Then check those bits in turn.

X16, X8	512byte	FMCDATARDWR0 ~7

Syndrome Data output Register (FMSYNDOUTRWn)

The initial value is 0x00.

There are 4 registers whose width is 16bit.

The syndrome data generated from ECC encoder in write mode enter to thess registers.

FMCSYNDOUTRW0	1st 128byte data syndrome data(2byte)
FMCSYNDOUTRW1	2nd 128byte data syndrome data
FMCSYNDOUTRW2	3rd 128byte data syndrome data
FMCSYNDOUTRW3	4th 128byte data syndrome data

To transfer these value to external FM, at first, CPU read these register.

After that, CPU write these data to FMDATARDWRn.

The way of transfer is same to normal

Syndrome Data input Register (FMSYNDINRWn)

The initial value is 0x00.

There are 4 register whose width is 16bit.

Before read operation in ECC mode, CPU have to write the syndrome data to these register in just order. If the read operation is finished after writing syndrome data, User can the occurrence of error by checking dec_status register.

FMCSYNDINRW0	1st 128byte data syndrome data(2byte)
FMCSYNDINRW1	2nd 128byte data syndrome data
FMCSYNDINRW2	3rd 128byte data syndrome data
FMCSYNDINRW3	4th 128byte data syndrome data

ECC Decoding Result Register (FMDEC_RESULT)

The initial value is 0xFFFF.

There are 4 register whose width is 16bit.

FMDEC_RESULT0	The result of 1st 128byte data decoding(2byte)
FMDEC_RESULT1	The result of 2nd 128byte data decoding
FMDEC_RESULT2	The result of 3rd 128byte data decoding
FMDEC_RESULT3	The result of 4th 128byte data decoding

Flash Density Register (FMDENSITY)

After reading Flash ID, CPU have to set the density register.

BIT	FUNCTION
0	Below 256M Bit
1	512M bit
2	1G Bit
3	2G Bit
4	4G Bit
15:5	Not used

Flash Controller FMC (FMTEST)

The state of fsm in FMC is written in this register..

BIT	FUNCTION
4:0	FSM
15:5	'0'

*** The Flash type and command supported.**

(Vcc=2.7 ~ 3.6V, Organization = x8, x16)

Density (bits)	Command Sets				ID
	Function	1'st cycle	2'nd cycle	3'rd cycle	
64M	Read1 Read2 Read ID Reset Page Program Block Erase Status Read	00h/01h 50h 90h FFh 80h 60h 70h	 10h D0h		E6h
128M	Read1 Read2 Read ID Reset Page Program Block Erase Status Read	00h/01h 50h 90h FFh 80h 60h 70h	 10h D0h		73h
256M	Read1 Read2 Read ID Reset Page Program Block Erase Status Read Copy-back	00h/01h 50h 90h FFh 80h 60h 70h 00h	 10h D0h 8Ah		75h

512M (single chip)	Read1 Read2 Read ID Reset Page Program Block Erase Status Read Copy-back	00h/01h 50h 90h FFh 80h 60h 70h 00h	 10h D0h 8Ah	 10h	76h
512M (Double chip)	Read1 Read2 Read ID Reset Page Program Block Erase Status Read Copy-back	00h/01h 50h 90h FFh 80h 60h 70h 00h	 10h D0h 8Ah		76h
1G (single chip)	Read Read for copy-back Read ID Reset Page Program Cache Program Copy-back program Block Erase Random input Random output Read Status	00h 00h 90h FFh 80h 80h 85h 60h 85h 05h 70h	30h 35h 10h 15h 10h D0h E0h		F1h
1G (Double chip)	Read1 Read2 Read ID Reset Page Program Block Erase Status Read Copy-back	00h/01h 50h 90h FFh 80h 60h 70h 00h	 10h D0h 8Ah	 10h	79h
2G (single,Double chip)	Read Read for copy-back Read ID Reset Page Program Cache Program Copy-back program Block Erase Random input Random output Read Status	00h 00h 90h FFh 80h 80h 85h 60h 85h 05h 70h	30h 35h 10h 15h 10h D0h E0h		DAh
4G (Double chip)	Read Read for copy-back Read ID Reset Page Program Cache Program Copy-back program Block Erase Random input Random output Read Status	00h 00h 90h FFh 80h 80h 85h 60h 85h 05h 70h	30h 35h 10h 15h 10h D0h E0h		DCh

SECURE DIGITAL CARD INTERFACE (SDCI)

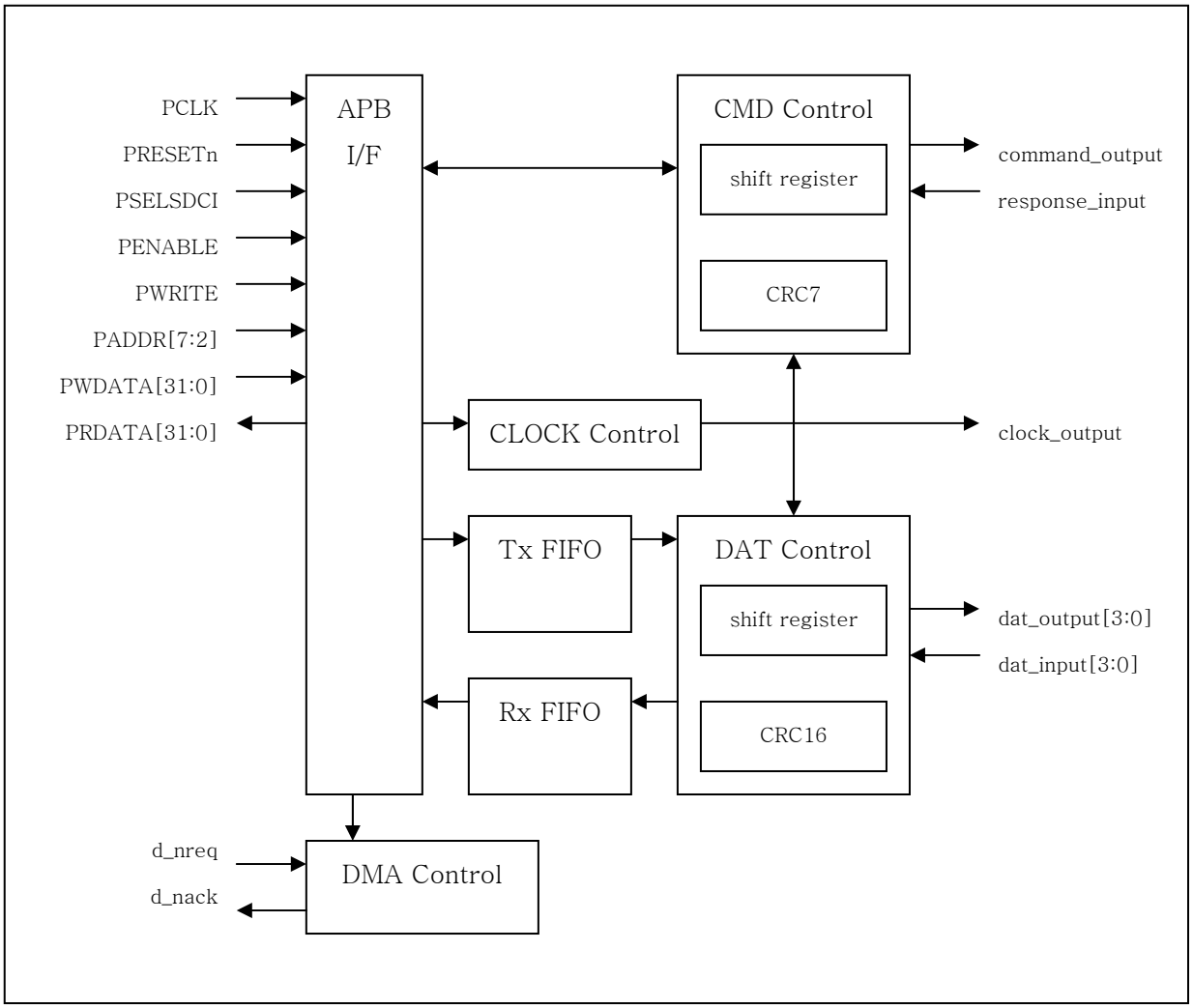
OVERVIEW

The Secure Digital Card Interface (SDCI) can interface for SD memory card and Multi-Media Card(MMC).

FEATURES

- Supports MultiMediaCard Specification Version 3.1
- Supports SD Memory Card Specification Version 1.0
- Cards Clock Rate up to System Clock(PCLK) Divided by 2
- 16 words (64 bytes) FIFO (depth 16) for data Transmit
- 8 words (32 bytes) FIFO (depth 8) for data Receive
- CRC7 & CRC16 Generator/Checker
- Normal, and DMA Data Transfer Mode
- Support for Block and Multi-block Data Read and Write
- 1bit/4bit(wide bus) Mode switch support
- Can Be Directly Connected to the AMBA Peripheral Bus (APB) Version 2.0

BLOCK DIAGRAM



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)
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)
dat0_enable	1	O	Data0 output enable
dat1_output	1	O	Data1 output(DAT1)
dat1_enable	1	O	Data1 output enable
dat2_output	1	O	Data2 output(DAT2)
dat2_enable	1	O	Data2 output enable
dat3_output	1	O	Data3 output(DAT3)
dat3_enable	1	O	Data3 output enable
DMA relative Signal			
d_nack	1	I	DMA request signal
d_nreq	1	O	DMA acknowledge signal

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

1. 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_ARG register.
2. Write the command information to SDCI_CMD register.
   - Confirm the ready of command transmission when the specific flag of SDCI_STA[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_STA[1] is set.
   - Command transmission is completed, the SDCI_STA[2] is set.
   - Response reception is in progress, the SDCI_STA[3] is set.
   - Response reception is completed, the SDCI_STA[4] is set.
5. Check the Status of response
   - If command response time-out error occur, the SDCI_STA[16] is set.
   - If response end bit error occur, the SDCI_STA[17] is set.
   - If response index error occur, the SDCI_STA[18] is set.
   - If response CRC error occur, the SDCI_STA[19] is set.
6. Check the Card Response, read from SDCI_RESP3~0 register.
7. Clear the corresponding flag of the SDCI_STA 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 SDCI_DCTRL register.
2. FIFO reset by writing the SDCI_DCTRL register.
3. Do **CMD path Programming**
4. Write the Data transmission start bit to SDCI_DCTRL[5:4] register(only data write operation).

5. When Write Operation, Write Tx data to SDCI_TXRX register while TxFIFO is available by checking SDCI_STA[13:12].
6. When Read Operation, Read Rx data to SDCI_TXRX register while RxFIFO is available by checking SDCI_STA[11:10].
7. Check the status of SDCI data operation.
    - Data transmission/reception is in progress, the SDCI_STA[5] is set.
    - Data block transmission/reception is completed, the SDCI_STA[6] is set
    - Data CRC data transmission/reception is completed, the SDCI_STA[7] is set.
    - CRC status token reception is completed, the SDCI_STA[8] is set, only write command.
    - Card busy state, the SDCI_STA[9] is set.
8. Check the Status of data transfer
    - If write data CRC status token has error, set the SDCI_STA[23] is set, CRC status token value is set SDCI_STA[22:10].
    - If read data CRC error occur, set the SDCI_STA[24] is set.
    - If read data end bit error occur, set the SDCI_STA[28:25] are set.
9. Clear the corresponding flag of the SDCI_STA register by write the SDCI_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 SDCI_DCTRL register.
3. FIFO reset by writing the SDCI_DCTRL register.
4. Do CMD path Programming
5. Write the Data Transmission start bit to SDCI_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 form 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.

SDCI Register

Offset	Register Name	Width	Read/Write	Register name	Initial Value
0x00	SDCI_CTRL	32bit	Read/Write	Control Register	0x0000_0000
0x04	SDCI_DCTRL	32bit	Read/Write	Data Control Register	0x0200_0000
0x08	SDCI_CMD	32bit	Read/Write	Command Register	0x0000_0000
0x0c	SDCI_ARG	32bit	Read/Write	Argument Register	0x0000_0000
0x10				Reserved	
0x14	SDCI_STAC	32bit	Write	Status Clear Register	0x0000_0000
0x18	SDCI_STA	32bit	Read	Status Register	0x8000_1400
0x1c				Reserved	
0x20	SDCI_RESP0	32bit	Read	Response0 Register	0x0000_0000
0x24	SDCI_RESP1	32bit	Read	Response1 Register	0x0000_0000
0x28	SDCI_RESP2	32bit	Read	Response2 Register	0x0000_0000
0x2c	SDCI_RESP3	32bit	Read	Response3 Register	0x0000_0000
0x30	SDCI_TXRX	32bit	Read/Write	Data Register	0x0000_0000

SDCI Control (SDCI_CTRL) Register

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>31:16</td><td>Reserved</td><td></td></tr>
<tr><td>15:8</td><td>CLKDIV</td><td>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</td></tr>
<tr><td>7:6</td><td>Reserved</td><td></td></tr>
<tr><td>5</td><td>DMA_REQ_CON</td><td>Select DMA request condition, only Read Command
0 DMA request when Receive FIFO is not empty.
1 DMA request when Receive FIFO is full.</td></tr>
<tr><td>4</td><td>L_ENDIAN</td><td>Select Endian</td></tr>
</table>

		0 Big endian 1 Little endian
3	DMAEN	DMA enable 0 DMA disable 1 DMA enable
2	BUS_WIDTH	Select the SDCard bus width. Only use SDCard (CARD_TYPE == 1'b0) 0 1bit data bus 1 4bit data bus
1	CARD_TYPE	Select the card type 0 SD Card 1 MM Card
0	SDCIEN	SDCI block enable 0 Disable the SDCI 1 Enable then SDCI

SDCI Data Control (SDCI_DCTRL) Register

Bit	Name	Description
31:16	BLK_LEN	Determine the data block size (unit : byte)
15:6	Reserved	
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	TXFIFIRST	Reset the transmit FIFO. 0 No effect 1 Reset the Transmit FIFO

SDCI Command (SDCI_CMD) Register

Bit	Name	Description

31	CMDSTR	Start the command transfer, automatically cleared. Only writable when SDCI_SR[0] = 1. 0 No effect 1 Command transmit start
30:21	Reserved	
20	NCR_NID	Select the clock cycle command to response. 0 N_{CR} (64 clock cycle) 1 N_{ID} (5 clock cycle)
19	RES_SIZE	Select the response size 0 48 bit size response 1 136 bit size response
18:16	RES_CLASS	Select response class 000 No Response 001 R1, R1b 010 R2 011 R3 100 R4 101 R5 110 R6 111 Reserved
15:9	Reserved	
8	CMD_TYPE	Select the command type 0 General command 1 Application command
7:6	CMD_CLASS	Select the command class 00 Broadcast commands 01 Addressed commands 10 Addressed data transfer commands – read command 11 Addressed data transfer commands – write command ※Caution : Use CMD13 during read/write operation, CMD_CLASS value must maintain the existing value.
5:0	CMD_NUM	Set the command number

SDCI Argument (SDCI_ARG) Register

Bit	Name	Description

31:0	ARGUMENT	Set the argument value

SDCI STATUS Clear (SDCI_STAC) Register

Bit	Name	Description
31:29	Reserved	
28	CLR_RD_DATENDE3	Clear RD_DATENDE3 at SDCI_SR 0 No effect 1 Clear the RD_DATEMDE3 bit
27	CLR_RD_DATENDE2	Clear RD_DATENDE2 at SDCI_SR 0 No effect 1 Clear the RD_DATEMDE2 bit
26	CLR_RD_DATENDE1	Clear RD_DATENDE1 at SDCI_SR 0 No effect 1 Clear the RD_DATEMDE1 bit
25	CLR_RD_DATENDE0	Clear RD_DATENDE0 at SDCI_SR 0 No effect 1 Clear the RD_DATEMDE0 bit
24	CLR_RD_DATCRCE	Clear RD_DATCRCE at SDCI_SR 0 No effect 1 Clear the RD_DATCRCE bit
23	CLR_WR_DATCRCE	Clear WR_DATCRCE at SDCI_SR 0 No effect 1 Clear the WR_DATCRCE bit
22:20	Reserved	
19	CLR_RESCRCE	Clear RESCRCE at SDCI_SR 0 No effect 1 Clear the RESCRCE bit
18	CLR_RESINDE	Clear RESINDE at SDCI_SR 0 No effect 1 Clear the RESINDE bit
17	CLR_RESENDE	Clear RESENDE at SDCI_SR 0 No effect 1 Clear the RESENDE bit
16	CLR_RESTOUTE	Clear RESTOUTE at SDCR_SR 0 No effect

		1 Clear the RESTOUTE bit
15:9	Reserved	
8	CLR_CRC_STAEND	Clear CRC_STAEND at SDCI_SR 0 No effect 1 Clear the CRC_STAEND bit
7	CLR_DAT_CRCEND	Clear DAT_CRCEND at SDCI_SR 0 No effect 1 Clear the DAT_CRCEND bit
6	CLR_DATEND	Clear DAT_END at SDCI_SR 0 No effect 1 Clear the DATEND bit
5	Reserved	
4	CLR_RESEND	Clear RESEND at SDCI_SR 0 No effect 1 Clear the RESEND bit
3	Reserved	
2	CLR_CMDEND	Clear CMDEND at SDCI_SR 0 No effect 1 Clear the CMDEND bit
1:0	Reserved	

SDCI STATUS (SDCI_STA) Register

※Clear Condition:

A : According to the SDCI interface state.

C : Clear by write '1' to corresponding bit of SDCI_SCR.

Bit	**Name**	**Description**	**Clear Condition**
31	WP_DECT_INPUT	WP pin status 0 WP Pin status is Low 1 WP Pin status is High	A
30:29	Reserved		
28	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

27	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
26	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
25	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
24	RD_DATCRCE	Read data has CRC error 0 No Read Data CRC Error occurs 1 Read Data CRC Error occurs	C
23	WR_DATCRCE	Write data CRC status response has error 0 No CRC status Token Error occurs 1 CRC status Token Error occurs	C
22:20	WR_CRC_STATUS	Write data CRC status response value 010 Non erroneous transmission 101 Transmission error 111 Card error else Reserved	A
19	RESCRCE	Response CRC error. Not valid, when RES_CLASS = 3'001. 0 No Response CRC Error occurs 1 Response CRC Error occurs	C
18	RESINDE	Response index error 0 No Response Index Error occurs 1 Response Index Error occurs	C
17	RESENDE	Response end bit error 0 No Response End Bit Error occurs 1 Response End Bit Error occurs	C
16	RESTOUTE	Response timeout error (Ncr, Nid) 0 No Command to Response timeout error occurs 1 Command to Response timeout error occurs	C
15:14	Reserved		

13	TX_FIFO_FULL	Tx FIFO full 0 Transmit FIFO is not full 1 Transmit FIFO is full	A
12	TX_FIFO_EMPTY	Tx FIFO empty 0 Transmit FIFO is not empty 1 Transmit FIFO is empty	A
11	RX_FIFO_FULL	Rx FIFO full 0 Receive FIFO is not full 1 Receive FIFO is full	A
10	RX_FIFO_EMPTY	Rx FIFO empty 0 Receive FIFO is not empty 1 Receive FIFO is empty	A
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 0 Data transmission/reception is not in progress 1 Data transmission/reception is in progress	A
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	A

		1 Response reception is in progress	
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_ARG set complete)	A

SDCI Response3 (SDCI_RESP3) Register

Bit	Name	Description
31:0	RESPONSE3	The RES_SIZE determines the value RES_SIZE: RESPONSE3 0: 32'h0000_0000 1: response[127:96]

SDCI Response2 (SDCI_RESP2) Register

Bit	Name	Description
31:0	RESPONSE2	The RES_SIZE determines the value RES_SIZE: RESPONSE2 0: 32'h0000_0000 1: response[95:64]

SDCI Response1 (SDCI_RESP1) Register

Bit	Name	Description
31:0	RESPONSE1	The RES_SIZE determines the value RES_SIZE: RESPONSE1 0: 32'h0000_0000 1: response[63:32]

SDCI Response0 (SDCI_RESP0) Register

Bit	Name	Description
31:0	RESPONSE0	The RES_SIZE determines the value RES_SIZE: 0 → response[39:8]; 1 → response[31:0]

SDCI Data (SDCI_TXRX) Register

Bit	Name	Description
31:0	SDCI_TXRX	Data buffer for transmit/receive

NOTE: If data is not aligned to word(4byte), SDCI_TXRX register value is following table(big_endian).

When Data read

write data	SDCI_TXRX[31:24]	SDCI_TXRX[13:16]	SDCI_TXRX[15:8]	SDCI_TXRX[7:0]
4n+1 byte	Write data[7:0]	stuff bits	stuff bits	stuff bits
4n+2 byte	Write data[7:0]	Write data[15:8]	stuff bits	stuff bits
4n+3 byte	Write data[7:0]	Write data[15:8]	Write data[23:16]	stuff bits

When Data write

read data	SDCI_TXRX[31:24]	SDCI_TXRX[13:16]	SDCI_TXRX[15:8]	SDCI_TXRX[7:0]
4n+1 byte	0x00	0x00	0x00	Read data[7:0]
4n+2 byte	0x00	0x00	Read data[7:0]	Read data[15:8]
4n+3 byte	0x00	Read data[7:0]	Read data[15:8]	Read data[23:16]

15 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
- Built-in 4-bit parallel port
- 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 (Exnternal 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

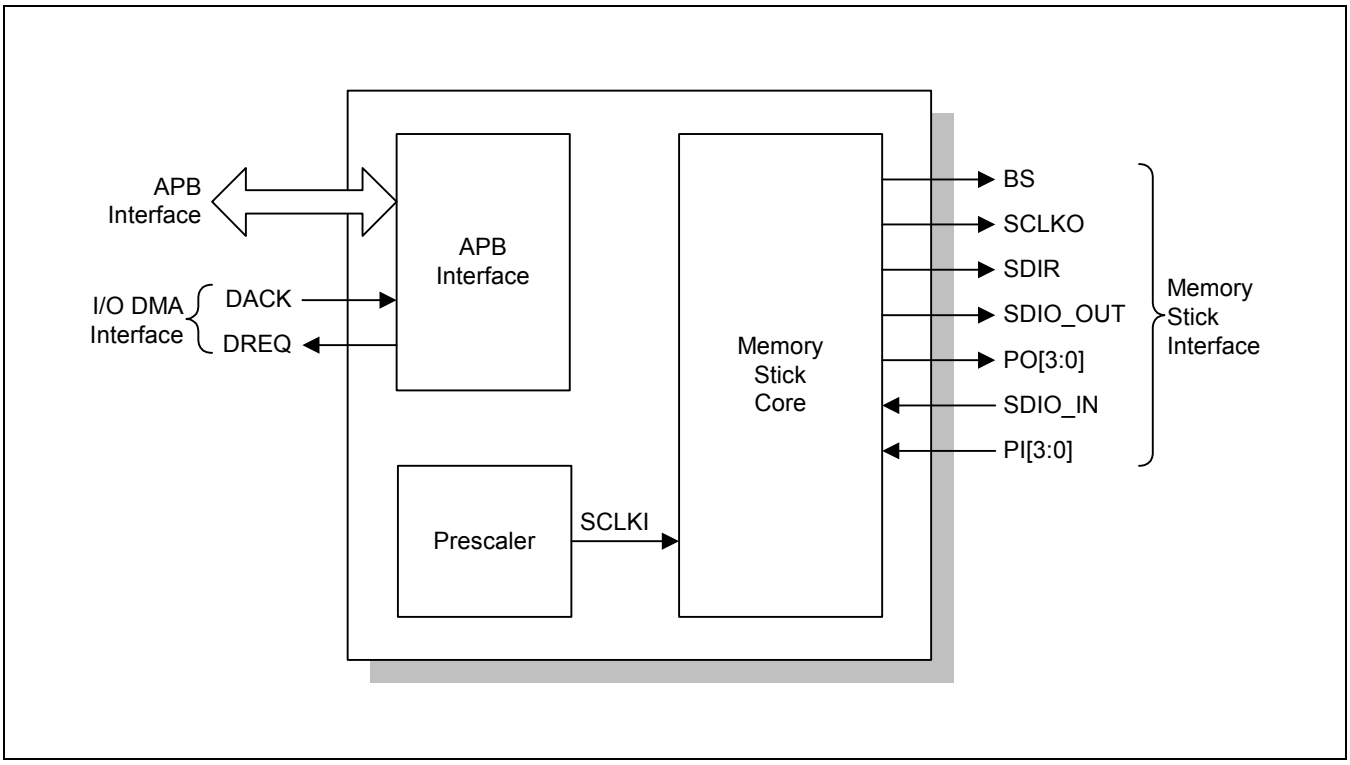


Figure 15-1. Memory Stick Host Controller Block Diagram

REGISTERS

PRESCALER CONTROL REGISTER(MSPRE)

Register	Address	R/W	Description	Reset Value
MSPRE	3C600000H	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	0h
PRESCALE	[7:0]	Prescale register to generate the SCLK (XSCLK) from the main clock. SCLK [Hz] = {main clock [Hz] / (MS_PRE + 1)} / 2	0h

INTERRUPT ENABLE REGISTER(MSINTEN)

Register	Address	R/W	Description	Reset Value
MSINTEN	3C600004H	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.	0h
RBE_INT_STAT	[11]	RBE interrupt status. Writing one clears this flag and deactivates the interrupt.	1h
RBF_INT_STAT	[10]	RBF interrupt status. Writing one clears this flag and deactivates the interrupt.	0h
TBE_INT_STAT	[9]	TBE interrupt status. Writing one clears this flag and deactivates the interrupt.	1h
TBF_INT_STAT	[8]	TBF interrupt status. Writing one clears this flag and deactivates the interrupt.	0h
RBE_INT_EN	[3]	Interrupt enable/disable for the RBE. 0 = Disable the interrupt from RBE 1 = Enable the interrupt from RBE	0h
RBF_INT_EN	[2]	Interrupt enable/disable for the RBF. 0 = Disable the interrupt from RBF 1 = Enable the interrupt from RBF	0h
TBE_INT_EN	[1]	Interrupt enable/disable for the TBE. 0 = Disable the interrupt from TBE 1 = Enable the interrupt from TBE	0h
TBF_INT_EN	[0]	Interrupt enable/disable for the TBF. 0 = Disable the interrupt from TBF 1 = Enable the interrupt from TBF	0h

COMMAND REGISTER(MSCMD)

Register	Address	R/W	Description	Reset Value
MSCMD	3C601000H	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 transfered 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.)	0h
DATA_SIZE	[9:0]	Data size for each transfer. The PID determines the value.	0h

CONTROL/STATUS REGISTER(MSCTRLSTAT)

Register	Address	R/W	Description	Reset Value
MSCTRLSTAT	3C601004H	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	3C601008H	R/W	Receie/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	3C60100cH	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	3C601010H	R/W	Parallel Port Control/Data Register	32 bits

MSPP	Bit	Description	Initial State
PIEN	[15:12]	Parallel port input enable 0 = Parallel port inputs are disabled 1 = Parallel port inputs are enabled	0h
POEN	[11:8]	Parallel port output enable 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 PIn 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	3C601014H	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	3C601018H	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.

NOTES

16 USB DEVICE CONTROLLER

OVERVIEW

Universal Serial Bus (USB) device controller is designed to provide a high performance full speed function controller solution with DMA interface. USB device controller allows bilk transfer with DMA, interrupt transfer and control transfer

USB device controller support :

- Full speed USB device controller compatible with the USB specification version 1.1
- DMA interface for bulk transfer
- Four endpoint with FIFO

  EP0 : 16 bytes (Register)

  EP1 : 64 bytes IN/OUT FIFO

  EP2 : 64 X 2 bytes IN/OUT FIFO

  EP3 : 64 X 2 bytes IN/OUT FIFO
- Integrated USB Transceiver

FEATURE

- Fully compliant with USB Specification Version 1.1
- Full speed (12Mbps) device
- Integrated USB Transceiver
- Supports control, interrupt, Isochronous and bulk transfer
- Four endpoints with FIFO
- Supports DMA interface for receive and transmit bulk endpoint. (EP1, EP2 and EP3)
- Support suspend and remote wakeup function.

BLOCK DIAGRAM

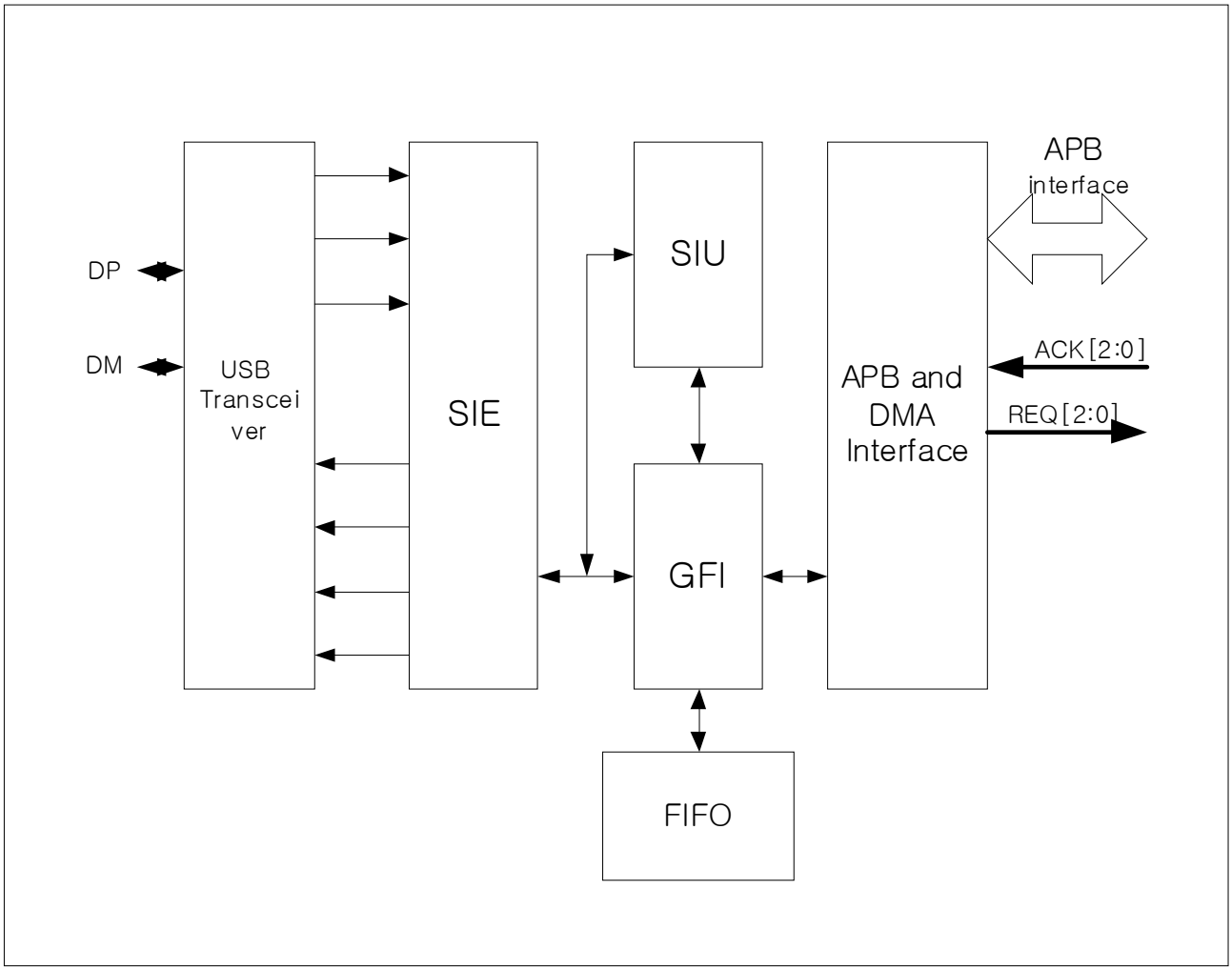


Figure -1. USB Device Controller Block Diagram

USB DEVICE CONTROLLER SPECIAL REGISTERS

This section describes detailed functionalities about register sets of USB device controller.

All special function register is byte-accessible.

Common indexed registers depend on INDEX register(INDEX_REG) value. For example if you want to write EP0 CSR register, you must write '0X00' on the INDEX_REG before writing IN_CSR1 register.

Register	Address	Description
NON INDEXED REGISTER		
FUNC_ADDR_REG	3D100000h	Function Address Register
PWR_MNGNT_REG	3D100010h	Power Management Register
EP_INT_REG	3D100020h	Endpoint Interrupt Register
USB_INT_REG	3D100060h	USB Interrupt Register
EP_INT_EN_REG	3D100070h	Endpoint Interrupt Enable Register
USB_INT_EN_REG	3D1000B0h	USB Interrupt Enable Register
FRAME_NUM1_REG	3D1000C0h	Frame Number 1 Register
FRAME_NUM2_REG	3D1000D0h	Frame Number 2 Register
INDEX_REG	3D1000E0h	Index Register
DMA_EN_REG	3D1000F0h	DMA Enable Register
EP0_FIFO_REG	3D100200h	Endpoint0 FIFO Register
EP1_FIFO_REG	3D100210h	Endpoint1 FIFO Register
EP2_FIFO_REG	3D100220h	Endpoint2 FIFO Register
EP3_FIFO_REG	3D100230h	Endpoint3 FIFO Register
COMMON INDEXED REGISTERS		
MAXP_REG	3D100130h	Endpoint MAX packet Register
IN INDEXED REGISTERS		
IN_CSR1_REG/EP0_CSR	3D100110h	EP In Control Status Register1/ EP0 Control Status Register
IN_CSR2_REG	3D100120h	EP In Control Status Register2
OUT INDEX REGISTERS		
OUT_CSR1_REG	3D100140h	EP Out Control Status Register1
OUT_CSR2_REG	3D100150h	EP Out Control Status Register2
OUT_WRT_CNT1	3D100160h	EP Out Write Count Register1
OUT_WRT_CNT2	3D100170h	EP Out Write Count Register2

FUNCTION ADDRESS REGISTER (FUNC_ADDR_REG)

This register maintains the USB device controller address assigned by the host. The MCU writes the value received through a SET_ADDRESS descriptor to this register. This address is used for the next token.

Register	Address	R/W	Description	Reset Value
FUNC_ADDR_REG	0X3D100000	R/W	Function address register	0x00

Symbol	Bit	MCU	USB	Description	Reset Value
ADDR_UPDATE	[7]	R/ SET	R/ CLEAR	Set by the MCU whenever if updates the FUNC_ADDR field in the register. This bit will be cleared by USB when DATA_END bit in EP0_CSR register.	0
FUNCTION_ADDR	[6:0]	R/W	R	The MCU writes the address to these bits	00

POWER MANAGEMENT REGISTER (PWR_MNGNT_REG)

This register is used for suspend, resume and reset signaling.

Register	Address	R/W	Description	Reset Value
PWR_MNGNT_REG	0X3D100010	R/W	Power management register	0X00

Symbol	Bit	MCU	USB	Description	Reset Value
ISO_UPDATE	[7]	R/W	R	Used for ISO mode only. If set, GFI waits for a SOF token to set IN_PKT_RDY even though a packet to send is already loaded by MCU. If an IN token is received before a SOF token, then a zero length data packet will be sent.	0
Reserved	[6:4]				
USB_RESET	[3]	R	SET	Set by the USB if reset signaling is received from the host. This bit remains set as long as reset signaling persists on the bus.	0
MCU_RESUME	[2]	R/W	R/ CLEAR	Set by the MCU for MCU Resume. The USB generates the resume signaling during 10ms, if this bit is set in suspend mode.	
SUSPEND_MODE	[1]	R	SET/ CLEAR	Set by USB automatically when the device enter into suspend mode. It is cleared under the following conditions: 1) The MCU clears the MCU_RESUME bit by	0
SUSPEND_EN	[0]	R/W	R	Suspend mode enable control bit 0 = Disable (default) The device will not enter suspend mode 1 = Enable suspend mode	0

INTERRUPT REGISTER1 (EP_INT_REG)

The USB core has two interrupt registers.

These registers act as status registers for the MCU when it is interrupted. The bits are cleared by writing a '1'(not '0') to each bit that was set.

Once the MCU is interrupted. MCU should read the contents of interrupt-related registers and write back to clear the contents if it is necessary.

Register	Address	R/W	Description	Reset Value
EP_INT_REG	0X3D100020	R/W	EP interrupt pending / clear register	0X00

Register	Bit	MCU	USB	Description	Reset Value
Reserved	[8:4]				
EP1~EP3	[3:1]	R/ CLEAR	SET	**For BULK/INTERRUPT IN endpoints** : Set by the USB under the following conditions: 1. IN_PKT_RDY bit is cleared 2. FIFO is flushed 3. SENT_STALL set **For BULK/INTERRUPT OUT endpoints** : Set by the USB under the following conditions: 1. Sets OUT_PKT_RDY bit 2. Sets SENT_STALL bit 3. **For ISO IN endpoints** : Set by the USB under the following conditions: 1. UNDER_RUN bit is set 2. IN_PKT_RDY bit is cleared 3. FIFO is flushed **Note** : Condition 1 and 2 are mutually exclusive **For ISO OUT endpoints** : Set by the USB under the following conditions: 1. OUT_PKT_RDY bit is set 2. OVER RUN bit is set **Note** : Condition 1 and 2 are mutually exclusive	0

EP0 Interrupt	[0]	R/ CLEAR	SET	Correspond to endpoint 0 interrupt. Set by the USB under the following conditions: 1. OUT_PKT_RDY bit is set. 2. IN_PKT_RDY bit is cleared. 3. SENT_STALL bit is set. 4. SETUP_END bit is set. 5. DATA_END bit is cleared. (it indicates the end of control transfer)	0

INTERRUPT REGISTER2 (USB_INT_REG)

Register	Address	R/W	Description	Reset Value
USB_INT_REG	0X3D100060	R/W	USB interrupt pending/clear register	0X00

Symbol	Bit	MCU	USB	Description	Reset Value
USB RESET Interrupt	[2]	R/ CLEAR	SET	Set by the USB when it receives reset signaling	0
RESUME Interrupt	[1]	R/ CLEAR	SET	Set by the USB when it receives resume signaling, while in Suspend mode. If the resume occurs due to a USB reset, then the MCU is first interrupted with a RESUME interrupt. Once the clocks resume and the SE0 condition persists for 3ms, USB RESET interrupt will be asserted.	
SUSPEND Interrupt	[0]	R/ CLEAR	SET	Set by the USB when it receives suspend signalizing. The bit is set whenever there is no activity for 3ms on the bus. Thus, if the MCU does not stop the clock after the first suspend interrupt, it will continue to be interrupted every 3ms as long as there is no activity on the USB bus By default, this interrupt is disabled.	0

Note : If the RESET interrupt is occurred, all USB device registers should be re-configured.

INTERRUPT ENABLE REGISTER (EP_INT_EN_REG/USB_INT_EN_REG)

Corresponding to each interrupt register. The USB device controller also has two interrupt enable registers (except resume interrupt enable). By default usb reset interrupt is enabled.

If bit = 0, the interrupt is disabled.

If bit = 1, the interrupt is enabled.

Register	Address	R/W	Description	Reset Value
EP_INT_EN_REG	0X3D100070	R/W	Determine which interrupt is enabled	0XFF

Symbol	Bit	MCU	USB	Description	Reset Value
EP3_INT_EN	[3]	R/W	R	EP3 interrupt enable bit 0 = interrupt disable, 1 = enable	1
EP2_INT_EN	[2]	R/W	R	EP2 interrupt enable bit 0 = interrupt disable, 1 = enable	1
EP1_INT_EN	[1]	R/W	R	EP1 interrupt enable bit 0 = interrupt disable, 1 = enable	1
EP0_INT_EN	[0]	R/W	R	EP0 interrupt enable bit 0 = interrupt disable, 1 = enable	1

Register	Address	R/W	Description	Reset Value
USB_INT_EN_REG	0X3D1000B0	R/W	Determine which interrupt is enabled	0X04

Symbol	Bit	MCU	USB	Description	Reset Value
RESET_INT_EN	[2]	R/W	R	Reset interrupt enable bit 0 = interrupt disable, 1 = enable	1
Reserved	[1]				0
SUSPEND_INT_EN	[0]	R/W	R	Suspend interrupt enable bit 0 = interrupt disable, 1 = enable	0

FRAME NUMBER REGISTER (FRAME_NUM1_REG/FRAME_NUM2_REG)

When the host transfer USB packets, each Start Of Frame(SOF) packet includes a frame number. The USB device controller catches this frame number and loads it into this register automatically.

Register	Address	R/W	Description	Reset Value
FRAME_NUM1_REG	0X3D1000C0	R	Frame number lower byte register	00

Symbol	Bit	MCU	USB	Description	Reset Value
FRAME_NUM1	[7:0]	R	W	Frame number lower byte value	00

Register	Address	R/W	Description	Reset Value
FRAME_NUM2_REG	0X3D1000D0	R	Frame number higher byte register	00

Symbol	Bit	MCU	USB	Description	Reset Value
FRAME_NUM2	[7:0]	R	W	Frame number higher byte value	00

DMA TRANSFER ENABLE REGISTER (DMA_EN_REG)

These registers maintain the number of bytes in the packet as the number is unloaded by the MCU

Register	Address	R/W	Description	Reset Value
DMA_EN_REG	0X3D1000F0	R/W	DMA transfer enable register	00

Symbol	Bit	MCU	USB	Description	Reset Value
EP3_DMA_ENABLE	[3]	R/W	R	Endpoint 3 DMA transfer enable 0 : DMA disable 1 : DMA enable	00
EP2_DMA_ENABLE	[2]	R/W	R	Endpoint 2 DMA transfer enable 0 : DMA disable 1 : DMA enable	00
EP1_DMA_ENABLE	[1]	R/W	R	Endpoint 1 DMA transfer enable 0 : DMA disable 1 : DMA enable	00
Reserved	[0]				

INDEX REGISTER (INDEX_REG)

The INDEX register is used to indicate certain endpoint registers effectively. The MCU can access the endpoint register (MAXP_REG, IN_CSR1_REG, IN_CSR2_REG, OUT_CSR1_REG, OUT_CSR2_REG, OUT_WRT_CNT1_REG and OUT_WRT_CNT2_REG) for an endpoint inside the core using the INDEX register.

Register	Address	R/W	Description	Reset Value
INDEX_REG	0X3D1000E0	R/W	Register index register	00

Symbol	Bit	MCU	USB	Description	Reset Value
INDEX	[7:0]	R/W	R	Indicate a certain endpoint	00

ENDPOINT0 CONTROL STATUS REGISTER (EP0_CSR)

This register has the control and status bits for Endpoint0. Since a control transaction is involved with both IN and OUT tokens, there is only one CSR register, mapped to the IN CSR1 register. (share IN1_CSR and can access by writing index register “0” and read/write IN1_CSR)

Register	Address	R/W	Description	Reset Value
EP0_CSR	0X3D100110	R/W	Endpoint 0 status register	00

Symbol	Bit	MCU	USB	Description	Reset Value
SERVICED_SETUP_END	[7]	CLEAR	CLEAR	The MCU should write a "1" to this bit to clear SETUP_END.	0
SERVICED_OUT_PKT_RDY	[6]	CLEAR	CLEAR	The MCU should write a "1" to this bit to clear OUT_PKT_RDY.	0
SEND_STALL	[5]	R/W	CLEAR	MCU should write a “1” to this bit at the same time it clears OUT_PKT_RDY, if it decodes and invalid token. 0 = Finish the STALL condition 1 = The USB issues a STALL and shake to the current control transfer	0
SETUP_END	[4]	R	SET	Set by the USB when a control transfer ends before DATA_END is set. When the USB sets this bit, an interrupt is generated to the MCU. When such a condition occurs, the USB flushes the FIFO and invalidates MCU access to the FIFO.	0
DATA_END	[3]	SET/ R	CLEAR	Set by the MCU on the conditions below : 1. After loading the last packet of data into the FIFO, at the same time IN_PKT_RDY is set. 2. While it clears OUT_PKT_RDY after unloading the last packet of data. 3. For a zero length data phase.	0
SENT_STALL	[2]	CLEAR /R	SET	Set by the USB if a control transaction is stopped due to a protocol violation. An interrupt is generated when this bit is set. The MCU should write “0” to clear this bit.	0
IN_PKT_RDY	[1]	SET/R	CLEAR	Set by the MCU after writing a packet of data into EP0 FIFO. The USB clears this bit once the packet has been successfully sent to the clears this bit, so as the MCU to load the next packet. For a zero length data phase, the MCU sets DATA_END at the same time.	0
OUT_PKT_RDY	[0]	R	SET	Set by the USB once a valid token is written to the FIFO. An interrupt is generated when the USB sets this bit. The MCU clears this bit by writing a “1” to the SERVICED_OUT_PKT_RDY bit	0

ENDPOINT IN CONTROL STATUS 1 REGISTER (IN_CSR1_REG)

Register	Address	R/W	Description	Reset Value
IN_CSR1_CSR	0X3D100110	R/W	IN Endpoint control status 1 register	00

Symbol	Bit	MCU	USB	Description	Reset Value
Reserved	[7]				
CLR_DATA_TOGGLE	[6]	R	R	Used in Set-up procedure. 0 : There are alternation of DATA0 and DATA1 1 : The data toggle bit is cleared and PID in packet will maintain DATA0	0
SENT_STALL	[5]	R/ CLEAR	SET	Set by the USB when an IN token issues a STALL handshake, after the MCU sets SEND_STALL bit to start STALL handshaking. When the USB issues a STALL handshake, IN_PKT_RDY is cleared	0
SEND_STALL	[4]	W/R	R	0 : The MCU clears this bit to finish the STALL condition. 1 : The MCU issues a STALL handshake to the USB.	0
FIFO_FLUSH	[3]	R/W	CLEAR	Set by the MCU if it intends to flush the packet in Input-related FIFO. This bit is cleared by the USB when the FIFO is flushed. The MCU is interrupted when this happens. If a token is in process, the USB waits until the transmission is complete before FIFO flushing. If two packets are loaded into the FIFO, only first packet(The packet is intended to be sent to the host) is IN_PKT_RDY bit is cleared	0
UNDER_RUN	[2]	R/ CLEAR	Set	Valid only For ISO mode Set by the USB when in ISO mode, an IN token is received and the IN_PKT_RDY bit is not set. The USB sends a zero length data packet for such conditions, and the next packet that is loaded into the FIFO is flushed. This bit is cleared by writing 0.	0
FIFO_NOT_EMPTY	[1]	R	Set	Indicates there is at least one packet of data in FIFO Bit[0] = 0, Bit[1] = 0 : No packet in the FIFO Bit[0] = 1, Bit[1] = 0 : 1 packet in the FIFO (when FIFO size is 2X MAXP) Bit[0] = 1, Bit[1] = 1 : 2 packet in the FIFO (when FIFO size is 2X MAXP) or Bit[0] = 1, Bit[1] = 1 : 1 packet in the FIFO (when FIFO size is MAXP)	0

IN_PKT_RDY	[0]	R/SET	CLEAR	Set by the MCU after writing a packet of data into the FIFO. The USB clears this bit once the packet has been successfully sent to the host. An interrupt is generated when the USB clears this bit, so the MCU can load the next packet. While this bit is set, the MCU will not be able to write to the FIFO. If the MCU sets SEND_STALL bit, this bit can not be set.	0

ENDPOINT IN CONTROL STATUS 2 REGISTER (IN_CSR2_REG)

Register	Address	R/W	Description	Reset Value
IN_CSR2_CSR	0X3D100120	R/W	IN Endpoint control status 2 register	20

Symbol	Bit	MCU	USB	Description	Reset Value
AUTO_SET	[7]	R/W	R	If set, whenever the MCU writes MAXP data, IN_PKT_RDY will automatically be set by the core without any intervention from MCU. If the MCU writes less than MAXP data, IN_PKT_RDY bit has to be set by the MCU.	0
ISO	[6]	R/W	R	Used only for endpoints whose transfer type is programmable. 1 : Configures endpoint to ISO mode 0 : Configures endpoint to Bulk mode	0
MODE_IN	[5]	R/W	R	Used only for endpoints whose direction is programmable. 1 : Configures endpoint direction as IN 0 : Configures endpoint direction an OUT	1
DMA_MODE	[4]	W/R	R	This bit is used only for endpoints whose interface has DMA. 1 : DMA enable 0 : DMA disable	0
Reserved	[3:0]				0

ENDPOINT OUT CONTROL STATUS 1 REGISTER (OUT_CSR1_REG)

Register	Address	R/W	Description	Reset Value
OUT_CSR1_CSR	0X3D100140	R/W	OUT Endpoint control status 1 register	00

Symbol	Bit	MCU	USB	Description	Reset Value
CLR_DATA_TOGGLE	[7]	R	SET	When the MCU writes a 1 to this bit, the data toggle sequence bit si reset to DATA0	0
SENT_STALL	[6]	R/ CLEAR	SET	Set by the USB when an OUT token is ended with a STALL handshake. The USB issues a stall handshake to the host if it sends more than MAXP data for the OUT_TOKEN, the MCU clears this bit by writing 0.	0
SEND_STALL	[5]	R/W	R	0 : The MCU clears this bit to end the STALL condition handshake, IN_PKT_RDY is cleared 1 : The MCU issues a STALL handshake to the USB. The MCU clears this bit to end the STALL condition handshake, IN_PKT_RDY is cleared.	0
FIFO_FLUSH	[4]	R/W	CLEAR	The MCU writes a 1 to flush the FIFO. This bit can be set only when OUT_PKT_RDY (D0) is set. The packet due to be unloaded by the MCU will be flushed.	0
DATA_ERROR	[3]	R	R/W	Valid only in ISO mode This bit should be sampled with OUT_PKT_RDY. When set, it indicates the data packet due to be unloaded by the MCU has an error(either bit stuffing or CRC). If two packets are loaded into the FIFO, and the second packet has an error, then this bit gets set only after the first packet is unloaded. This bit is automatically cleared when OUT_PKT_RDY gets cleared.	0
OVER_RUN	[2]	R/ CLEAR	R/W	Valid only in ISO mode. This bit is set if the core is not able to load an OUT ISO token into the FIFO. MCU clears this bit by writing 0.	0
FIFO_FULL	[1]	R	R/W	Indicate no more packets can be accepted Bit[0] = 0, Bit[1] = 0 : No packet in the FIFO Bit[0] = 1, Bit[1] = 0 : 1 packet in the FIFO (when FIFO size is 2X MAXP) Bit[0] = 1, Bit[1] = 1 : 2 packet in the FIFO (when FIFO size is 2X MAXP) or Bit[0] = 1, Bit[1] = 1 : 1 packet in the FIFO (when FIFO size is MAXP)	0

OUT_PKT_RDY	[0]	R/ CLEAR	SET	Set by the USB after it has loaded a packet of data into the FIFO. Once the MCU reads the packet from FIFO, this bit should be cleared by MCU (write a “0”)	

ENDPOINT OUT CONTROL STATUS 2 REGISTER (OUT_CSR2_REG)

Register	Address	R/W	Description	Reset Value
OUT_CSR2_CSR	0X3D100150	R/W	OUT Endpoint control status 2 register	00

Symbol	Bit	MCU	USB	Description	Reset Value
AUTO_CLR	[7]	R/W	R	If the MCU is set, whenever the MCU reads data from the OUT FIFO, OUT_PKT_RDY will automatically be cleared by the logic without any intervention from the MCU	0
ISO	[6]	R/W	R	Determine endpoint transfer type. 0 : Configures endpoint to Bulk mode 1 : Configures endpoint to ISO mode Default = 0	0
Reserved	[5:0]	R	R		

ENDPOINT FIFO REGISTER (EPn_FIFO_REG)

Register	Address	R/W	Description	Reset Value
EP0_FIFO	0X3D100200	R/W	Endpoint0 FIFO register	XX
EP1_FIFO	0X3D100210	R/W	Endpoint1 FIFO register	XX
EP2_FIFO	0X3D100220	R/W	Endpoint2 FIFO register	XX
EP3_FIFO	0X3D100230	R/W	Endpoint3 FIFO register	XX

Symbol	Bit	MCU	USB	Description	Reset Value
FIFO_DATA	[7:0]	R/W	R/W	FIFO data value	XX

MAX PACKET REGISGER (MAXP_REG)

Register	Address	R/W	Description	Reset Value
MAXP_REG	0X3D100130	R/W	Endpoint MAX packet register	00

Symbol	Bit	MCU	USB	Description	Reset Value
MAXP	[7:0]	R/W	R	0000_0000 : MAXP = 8 (default value) 0000_0010 : MAXP = 16 (endpoint 0) 0000_1000 : MAXP = 64 (endpoint 1,2,3)	00

ENDPOINT OUT WRITE COUNT REGISGER (OUT_WRT_CNT1/OUT_WRT_CNT2)

These registers maintain the number of bytes in the packet as the number is unloaded by the MCU

Register	Address	R/W	Description	Reset Value
OUT_WRT_CNT1	0X3D100160	R/W	Endpoint out write count register1	00

Symbol	Bit	MCU	USB	Description	Reset Value
OUT_CNT_LOW	[7:0]	R	W	Lower byte of write count	00

Register	Address	R/W	Description	Reset Value
OUT_WRT_CNT2	0X3D100170	R/W	Endpoint out write count register1	00

Symbol	Bit	MCU	USB	Description	Reset Value
OUT_CNT_HIGH	[7:0]	R	W	Higher byte of write count	00

Chapter 17. IIS(Tx/Rx) module

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 S5L840F, 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 SDRAM. In reception mode, IIS module gets audio data from external source and stores them into SDRAM by requesting DMA to IODMA module. 5 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. S5L840F 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
- 5 x 24-bit buffers for Transmission and 1 x 24-bit buffer for reception
- 16/20/24 bit data per channel
- Up to 10 channels for transmission and 2 channels 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

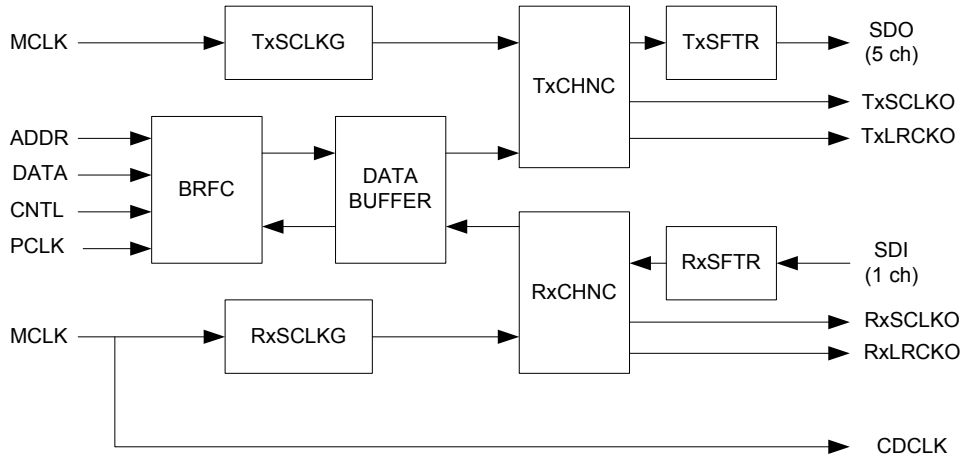


Fig 1. IIS Block Diagram

- BRFC : register bank, APB interface, finite state machine for DMA request
- TxSCLKG, RxSCLKG : 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

Pin Description

Pin Name	Width	I/O	Description
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	24	I	Write data in APB
PRDATA	32	O	Read data in APB
IIS Interface			
MCLK	1	I	Audio main clock
CDCLK	1	O	External DAC clock
TXSCLKO	1	O	Serial bit clock for transmission
TXLRCKO	1	O	Word select signal for transmission
SDO0	1	O	Audio data output
SDO1	1	O	Audio data output
SDO2	1	O	Audio data output
SDO3	1	O	Audio data output
SDO4	1	O	Audio data output
RXSCLKO	1	O	Serial bit clock for reception
RXLRCKO	1	O	Word select signal for reception
SDI	1	I	Audio data input
DMA Request			
DMAREQn	2	O	DMA request signal [Tx, Rx]
DMAACKn	2	I	DMA acknowledge signal [Tx, Rx]

Registers

Name	Width	Address (virtual)	R/W	Description	Reset
I2SCLKCON	32	0x3ca0 0000(0x39 4000)	R/W	Clock Control Register	0x0000 0002
I2STXCON	32	0x3ca0 0004(0x39 4004)	R/W	Tx configuration Register	0x0000 0000
I2STXCOM	32	0x3ca0 0008(0x39 4008)	R/W	Tx command Register	0x0000 0000
I2STXDB0	32	0x3ca0 0010(0x39 4010)	W	Tx data buffer	0x0000 0000
I2SRXCON	32	0x3ca0 0030(0x39 4030)	R/W	Rx configuration Register	0x0000 0000
I2SRXCOM	32	0x3ca0 0034(0x39 4034)	R/W	Rx command Register	0x0000 0000
I2SRXDB	32	0x3ca0 0038(0x39 4038)	R	Rx data buffer	0x0000 0000
I2SSTATUS	32	0x3ca0 003c(0x39 403c)	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
														1	0

I2SCLKCON	Bit	Description	Initial State
	[31:2]	Reserved	0
I2S clock down ready (read only)	[1]	0 = clock-down not ready 1 = clock-down ready	1
I2S 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
0	1	1	0	0	0	1	1	0	1	0	0	1	0	0	1

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 01 = 20 bit 10 = 24 bit 11 = N/A	00
Bit Clock per Frame (Frame = Left + Right)	[4:3]	00 = 32 fs 01 = 48 fs 10 = 64 fs 11 = N/A	00
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
												1	1	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

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
			0	1	1	0	0	0	1	1	0	1	0	0	1

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 01 = 20 bit 10 = 24 bit 11 = N/A	0
Bit Clock per Frame (Frame = Left + Right)	[1:0]	00 = 32 fs 01 = 48 fs 10 = 64 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
												1	1	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

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
												1	1	0	1

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 whose frequency is 117MHz. 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 1. Serial bit clock(SCLK) is determined depending on MCLK and data bit per channel(Table 2) and is set by the value of configuration register(I2STXCON, I2SRXCON). Word select signal(LRCK) has the same frequency as sampling frequency(fs).

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
MCLK (MHz)	256fs									
	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 1. the frequency of audio main 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

Table 2. the frequency of serial bit clock

Audio Interface Format

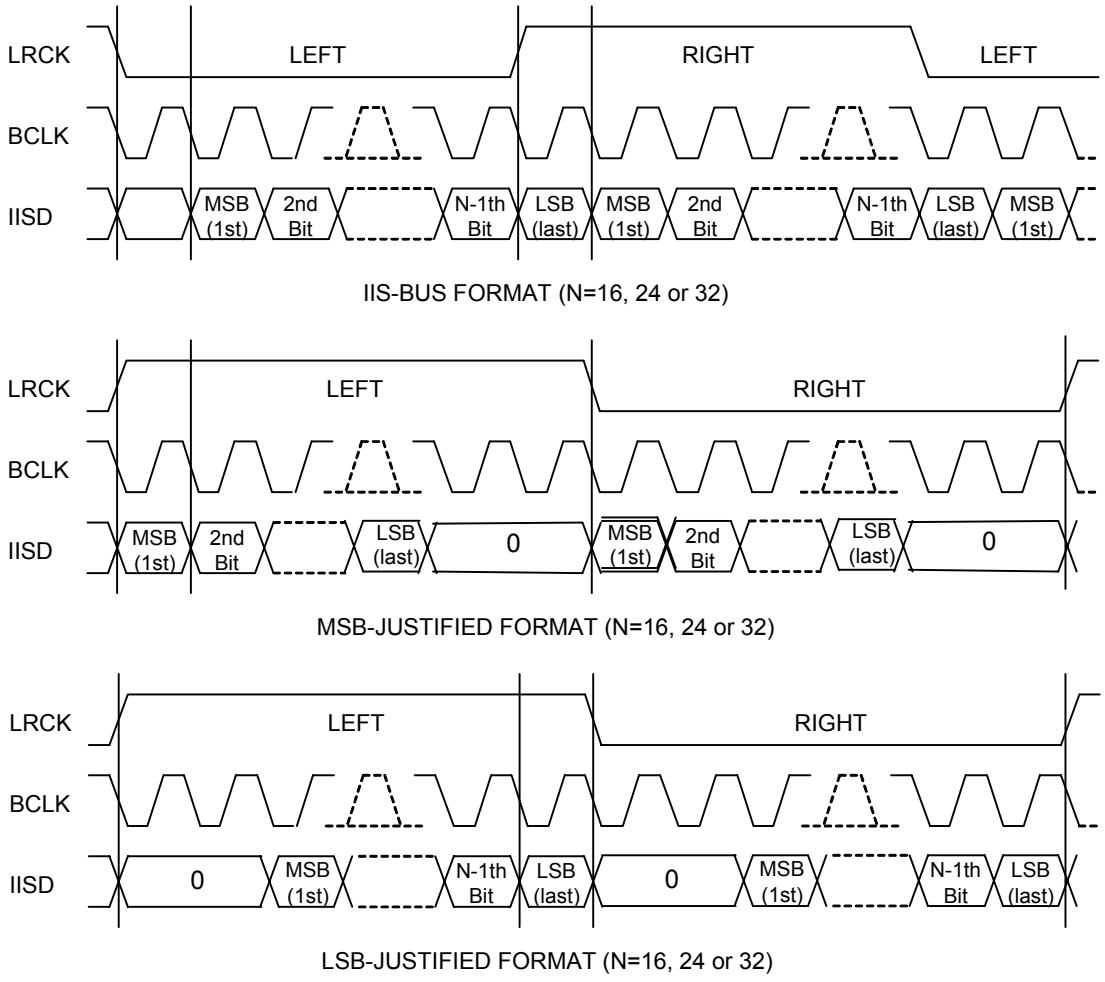


Fig 2. audio data interface format

In S5L840F, 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. Set I2S_power_on to '0'.
2. 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 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 S5L840F 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.

1. DMA channel 0 set (IIS Tx channel)
   DMABASE0, DMACON0, DMATCNT0, DMACOM0 registers set
2. DMA channel 1 set (IIS Rx channel)
   DMABASE0, DMACON0, DMATCNT0, DMACOM0 registers set
3. IIS Rx mode set (you must set Rx mode first and then set Tx mode)

I2SRXCON register set

4. IIS Tx mode set
   I2STXCON register set
5. IIS Rx clock on
   I2SRXCOM register set
6. IIS Tx clock on
   I2STXCOM register set

Chapter 18. IIC module

S5L840F 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.[1]

FEATURE

- Four operation mode
  - Master transmitter mode
  - Master receive mode
  - Slave transmitter mode
  - Slave receive mode
- Configurable slave address
- Operation pending until the interrupt pending flag is cleared by the software

[1] This is dependent on the devices. Some other devices, for example EEPROM, need not to generate ACK signals.

BLOCK DIAGRAM

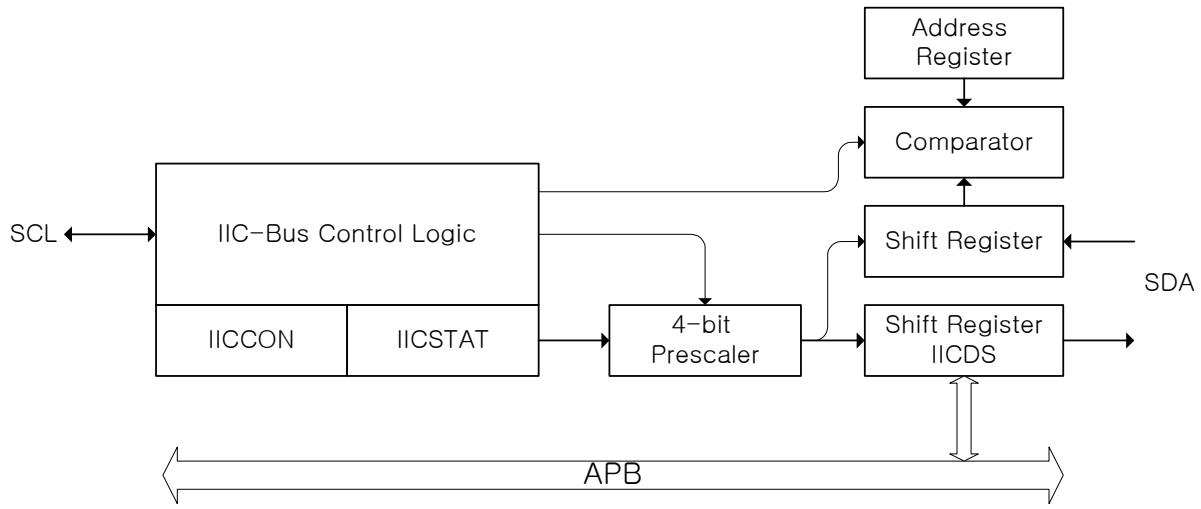


Fig 1. IIC Block Diagram

PIN DESCRIPTION

Pin Name	Width	I/O	Description
GCLK	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	6	I	Address in APB
PWDATA	8	I	Write data in APB
PRDATA	32	O	Read data in APB
IIC Interface			
SCLIN	1	I	IIC clock line input
SDAIN	1	I	IIC data line input
SCLOUT	1	O	IIC clock line output
SDAOUT	1	O	IIC data line output
Interrupt Interface			
INTC	1	O	Interrupt

GCLK
APB clock (main clock)

PRESETn
Global reset to reset the internal register

APB interface signals (psel, penable, pwrite, paddr, pwdata, prdata)
AMBA APB interface signals

SCLIN

IIC clock input line.

SDAIN
IIC data input line

SCLOUT
IIC clock output line

SDAOUT
IIC data output line

INTC
Interrupt signal.

REGISTERS

Name	Width	Address(Virtual)	R/W	Description	Reset
IICCON	32	0x3c90 0000(0x39 2000)	R/W	Control Register	0x0000 000x
IICSTAT	32	0x3c90 0004(0x39 2004)	R/W	Control/Status Register	0x0000 0000
IICADD	32	0x3c90 0008(0x39 2008)	R/W	Bus Address Register	-
IICDS	32	0x3c90 000c(0x39 200c)	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[1] (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 formular: Tx clock = IICCLK / (IICCON[3:0] + 1) - Shuld be CK_REG[3:0] > 0	-

***-CK_REG[3:0]** 의 값을 **0** 으로 써주게 되면 **SCL** 은 정상적으로 발생하지만 **SDA** 가 계속 **0** 으로 나가게 된다. 따라서 **0** 이상의 값을 써주어야 한다.*

- ***Slave Mode** 일때도 **CK_SEL, CK_REG** 값을 **Master** 와 같은 값으로 셋팅해주어야 하는가?*
  ***=> Slave** 일때는 **clock** 값을 셋팅 하지 않아도 된다.*

[1] 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

% I2C 가 1byte 를 보내면 ack 를 받던 받지 않던 interrupt 를 발생 시킨다. 이 interrupt 구간에서는 SCL 이 LOW 로 유지 되면서 BUS 를 계속 잡고 있는다.

% 모든 I2C Device 는 default 로 "0" slave address 에 ack 를 발생시키게 되는데 이는 하나의 Master 가 동시에 여러 I2C Slave 에 동일한 값을 write 하고자 할때 사용한다.

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]	-

- ***IICADD 에 자기 자신의 Slave Address 를 쓰기 위해서는 serial output 이 반드시 disable 상태여야 한다. 만약에 serial output 을 enable 시켜 놓고 Slave Address 를 쓸려고 하면 아무리 써도 써지지 않고 Reset 이후의 초기값 "0" 의 값이 보존된다.***

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.	-

- ***Serial Output 을 Enable 시켜놓고 IICDS 에 값을 써주어야 한다 .***
  그렇지 않으면 IICDS 에 값이 써지지 않아서 Unkown 값이 SDA 로 나가게 된다.

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

In S5L840F, 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.

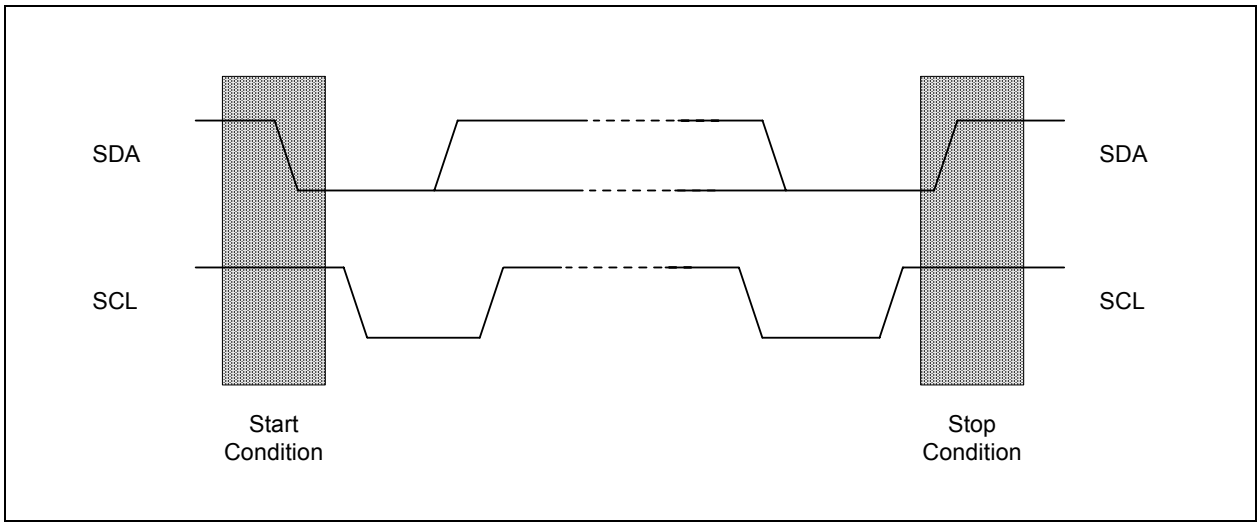


Figure 2. Start and Stop Condition

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.

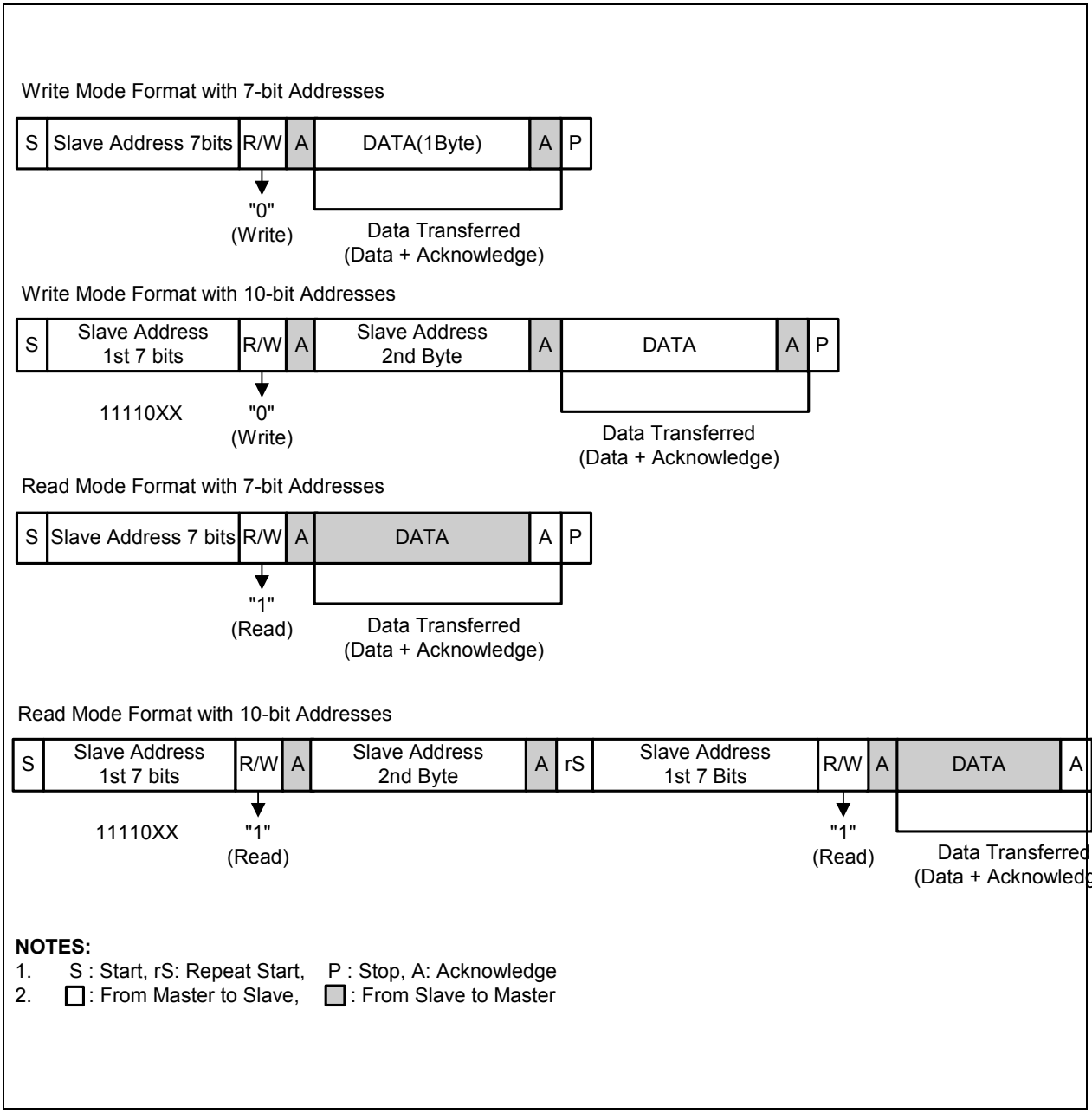


Figure 3. IIC-Bus Interface Data Format

% restart 는 scl 이 low 일때 start 가 발생하면 restart 가 된다. 즉 처음엔 scl 이 high 일때는 아직 버스를 잡고 있지 않은 상태인데 이때 start bit 을 1 로 write 하면 start 가 되는 것이고 버스를 잡은 상태(scl low) 일때 다시 start bit 를 1 로 write 하면 restart 가 된다.

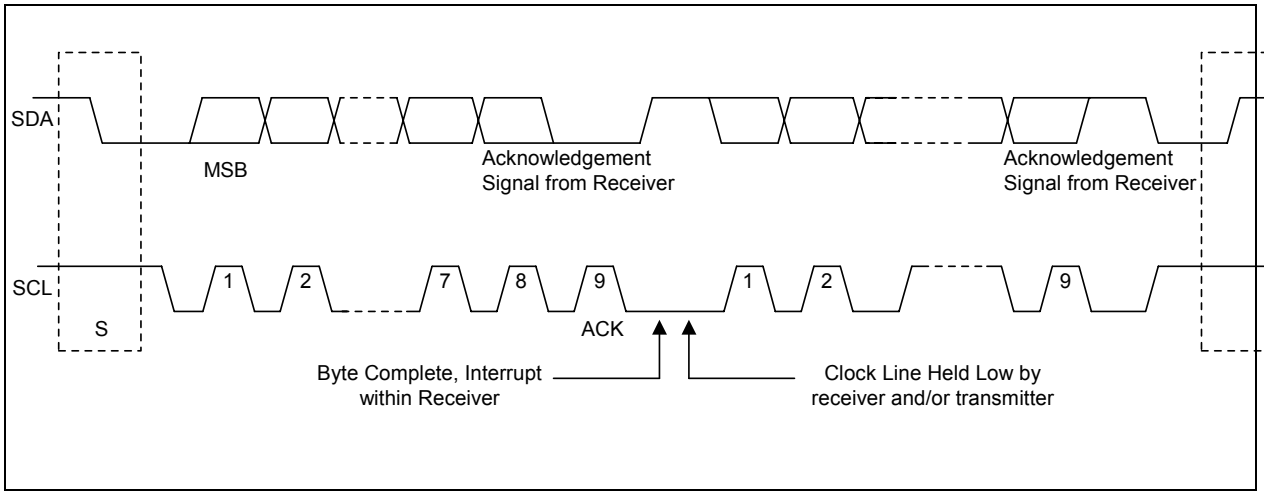


Figure 4. Data Transfer on the IIC-Bus

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.

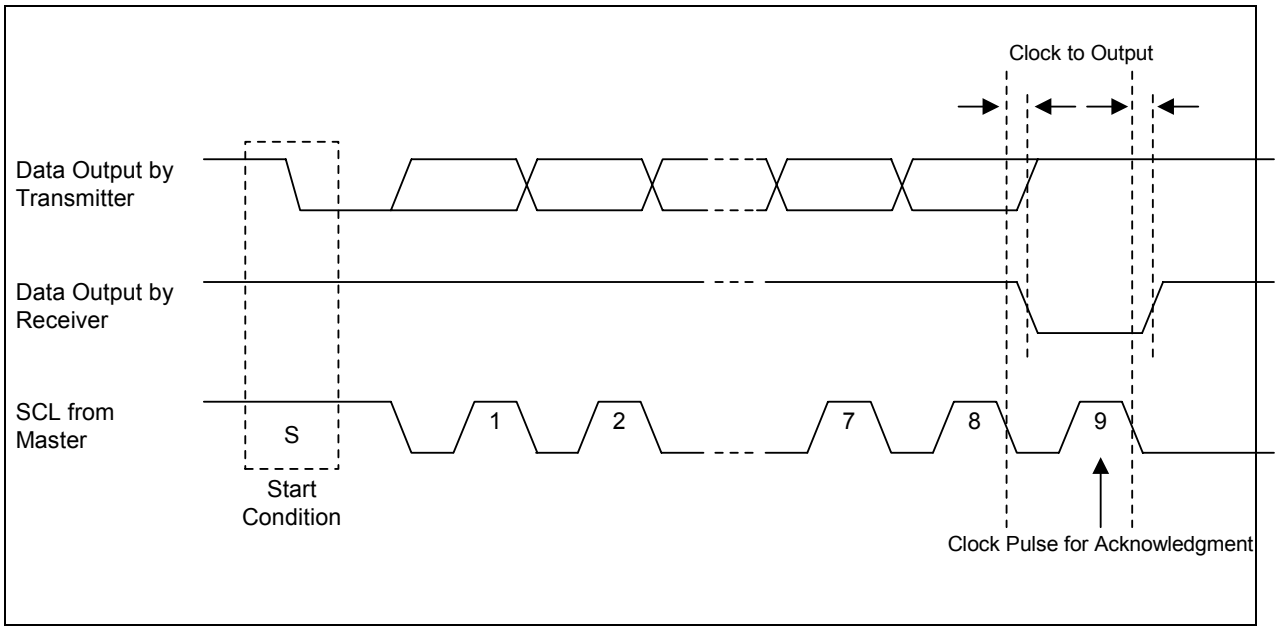


Figure 5. Acknowledge on the IIC-Bus

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.

-Master 가 7bit 어드레스를 Slave 에게 보내고 8 번째 bit 에서 R/W 정보를 주면 Slave 는 이 R/W 비트에 따라서 Transmit or Receive Mode 로 설정이 되게 된다. 즉 Master 가 Slave 의 Transmit or Receive 를 결정 시켜 준다.

Bus Arbitration Procedures

Arbitration takes place on the SDA line to prevent the contention on the bus between two masters. If a master with a SDA high level detects the other master with a SDA active low level, it will not initiate a data transfer because the current level on the bus does not correspond to its own. The arbitration procedure will be extended until the SDA line turns high.
However, when the masters simultaneously lower the SDA line, each master should evaluate whether or not the mastership is allocated to itself. For the purpose of evaluation, each master should detect the address bits. While each master generates the slaver address, it should also detect the address bit on the SDA line because the SDA line is likely to get low rather than to keep high. Assume that one master generates a low as first address bit, while the other master is maintaining high. In this case, both masters will detect low on the bus because the low status is superior to the high status in power. When this happens, low (as the first bit of address) generating master will get the mastership while high (as the first bit of address) generating master should withdraw the mastership. If both masters generate low as the first bit of address, there should be 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

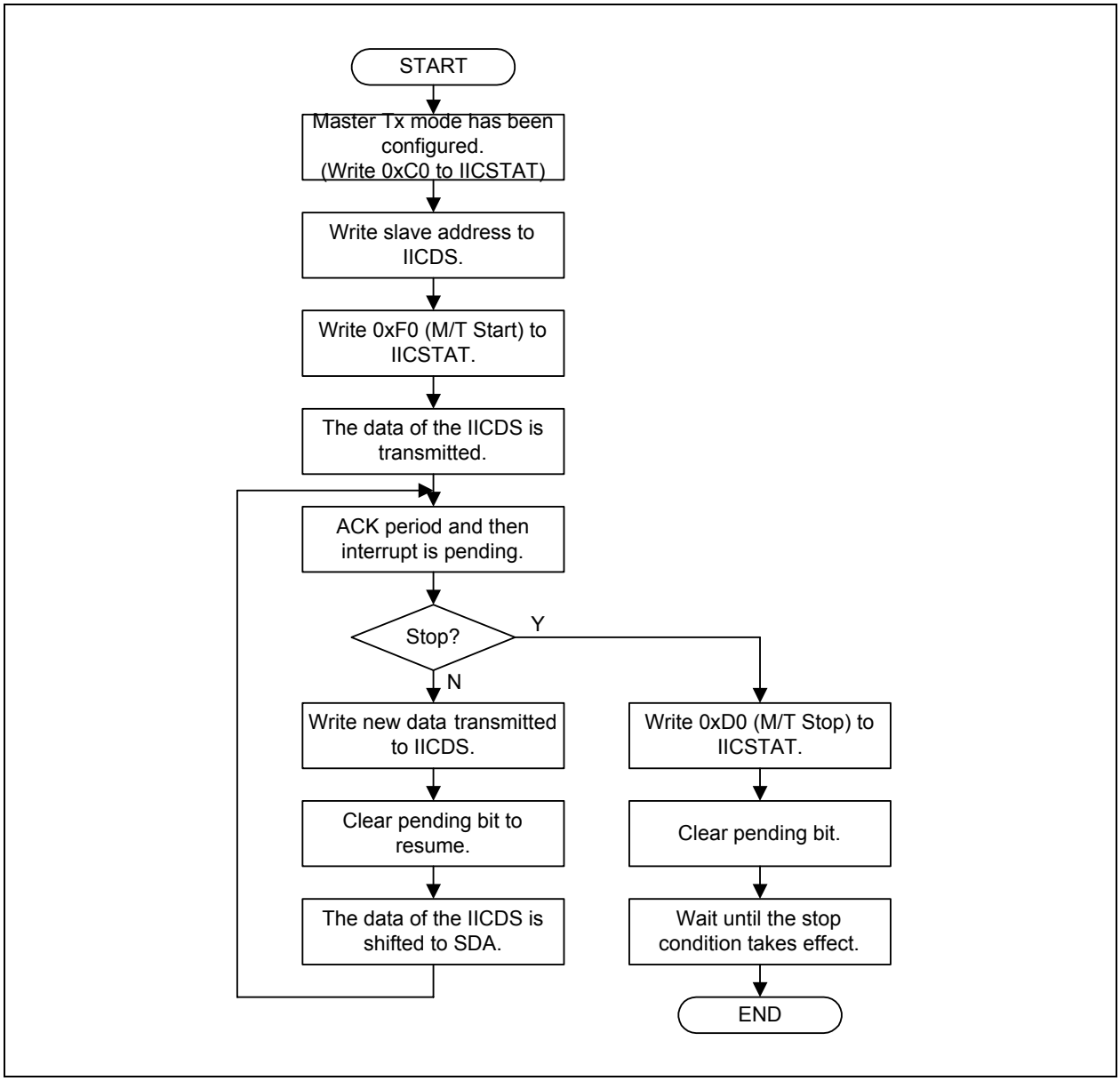


Figure 6. Operations for Master/Transmitter Mode

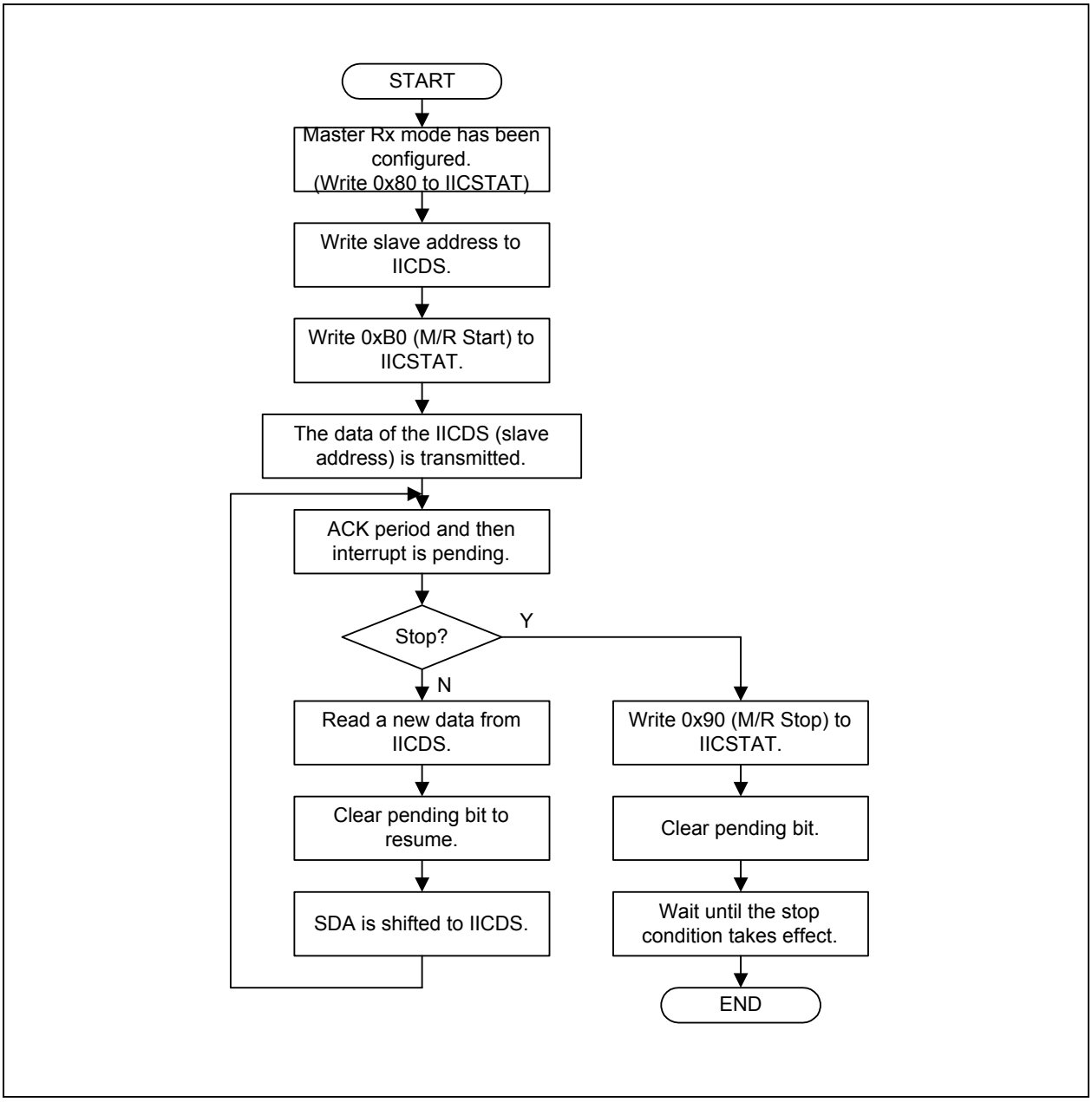


Figure 7. Operations for Master/Receiver Mode

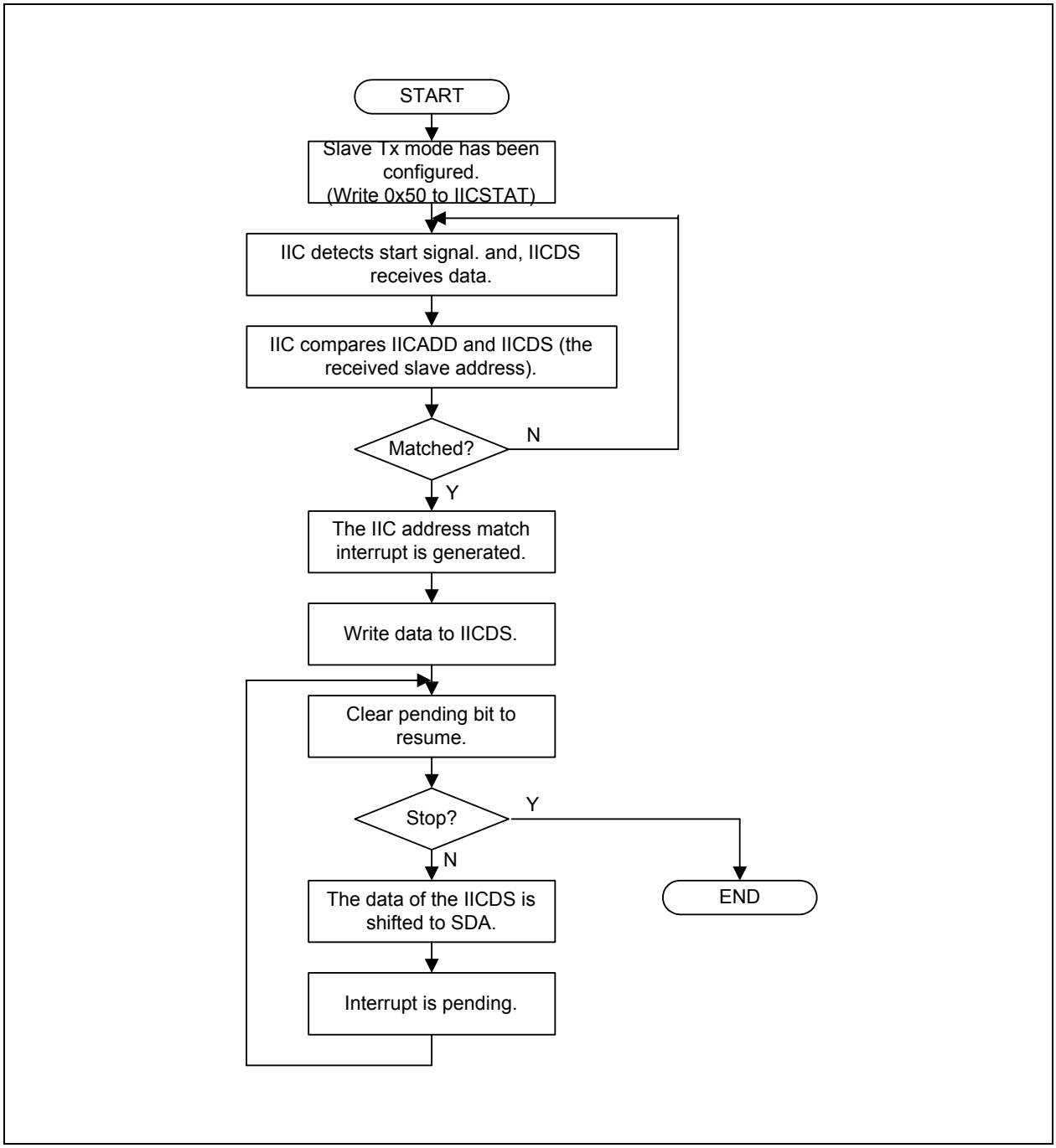


Figure 8. Operations for Slave/Transmitter Mode

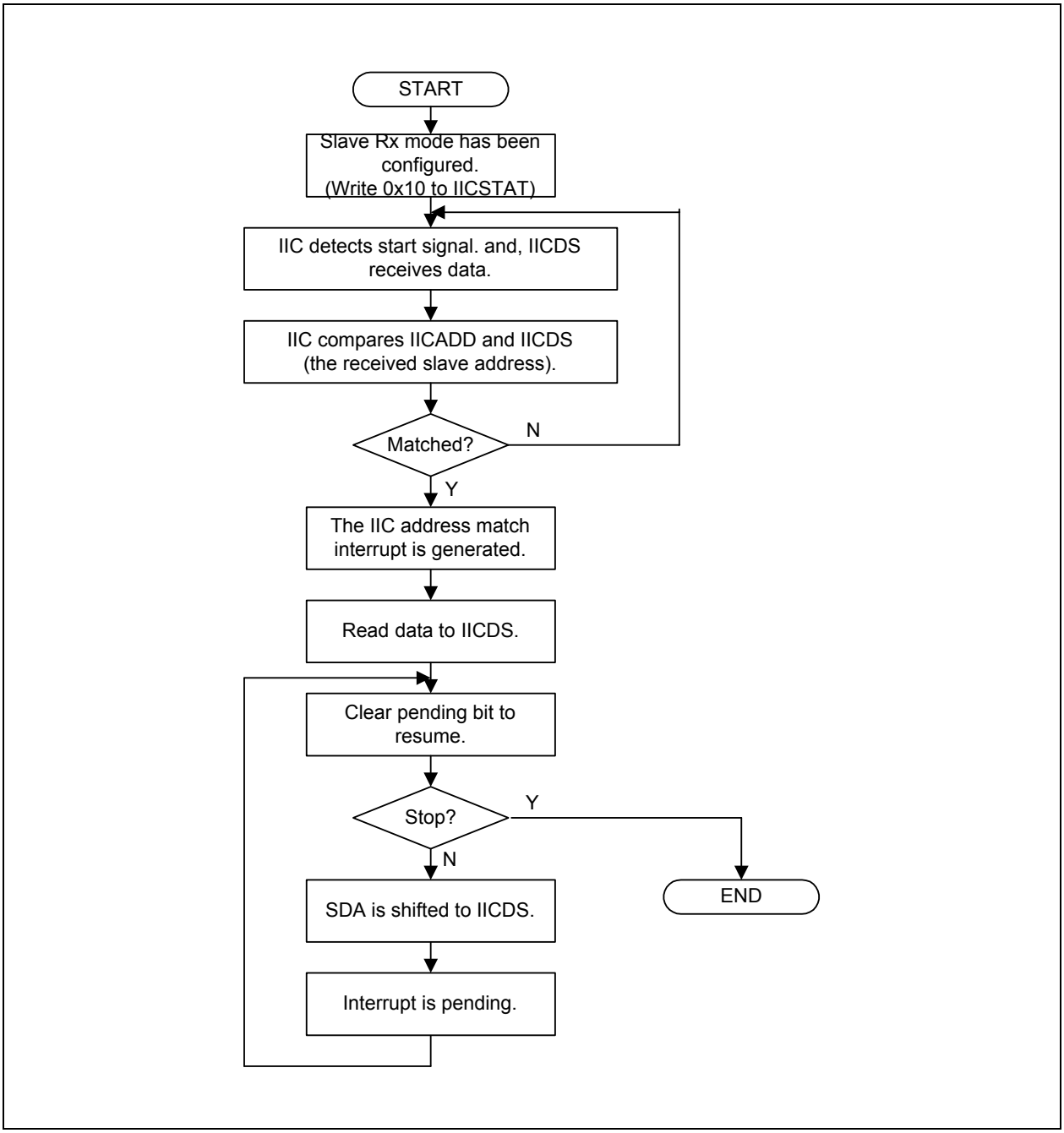


Figure 9. Operations for Slave/Receiver Mode

SPI

Preliminary Spec

Version 0.1

Aug. 29, 2003

Byeong-Woo Jeon

Media Player P/T

System LSI Division

Device Solution Network

Contents

REVISION HISTORY

Date	Version Number	Update Contents
2003.08.29	0.1	Initial version

Overview

SPI(Serial Peripheral Interface) in S5L840FX can transfer serial data with various peripherals. SPI has two 8bit shift registers for transmission and reception, respectively. During an 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 : an SPI clock pin(SCK), an MISO(master_in_slave_out) data pin, an MOSI(master_out_slave_in) data pin and an 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.

Feature

- SPI protocol(Ver. 2.11) compatible
- 2 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

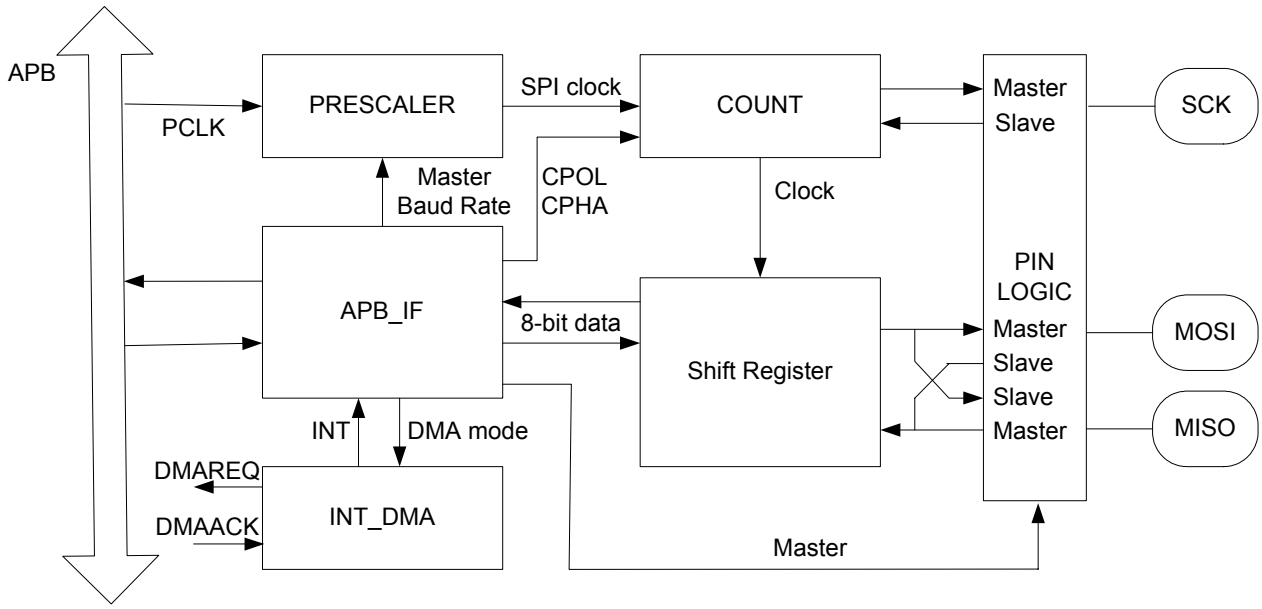


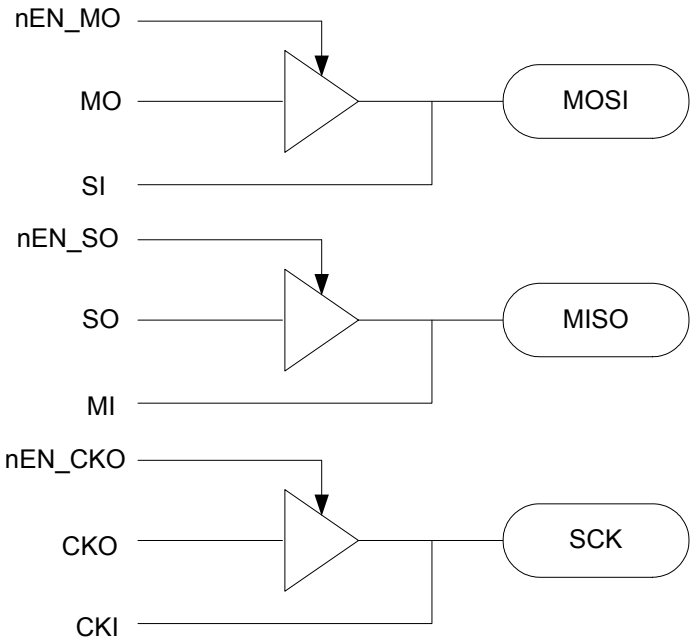
Fig 1. SPI Block Diagram

- 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 : 2 8-bit shift registers for transmission and reception
- PIN LOGIC : generation of signals needed to make 3 basic SPI pins

Pin Description

Pin Name	Width	I/O	Description
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	3	I	Address in APB
PWDATA	12	I	Write data in APB
PRDATA	12	O	Read data in APB
SPI Interface			
MI	1	I	Data input for master mode
SI	1	I	Data input for slave mode
CKI	1	I	Clock input for slave mode
nSSI	1	I	Active low signal indicating whether SPI module is slave or not
MO	1	O	Data output for master mode
nEN_MO	1	O	Data output enable for master mode
SO	1	O	Data output for slave mode
nEN_SO	1	O	Data output enable for slave mode
CKO	1	O	Clock output for master mode
nEN_CKO	1	O	Clock output enable for master mode
DMA Request & Interrupt			
nDREQ	1	O	Active low DMA request signal
nDACK	1	I	Active low DMA acknowledgement signal
INT_SPI	1	O	Interrupt of SPI module

3 basic SPI pins(MOSI, MISO, SCK) are organized as follows in pad module using signals above.



Registers

Base address : 0x3cd0_0000 (in 32bit), 0x39_a000 (in 24bit)

Name	Width	Address	R/W	Description	Reset
SPCLKCON	32 24	Base + 00h	R/W	Clock Control Register	0x0000 0002
SPCON	32 24	Base + 04h	R/W	Control Register	0x0000 0000
SPSTA	32 24	Base + 08h	R/W	Status Register	0x0000 0001
SPPIN	32 24	Base + 0ch	R/W	Pin Control Register	0x0000 0000
SPTDAT	32 24	Base + 10h	R/W	Tx Data Register	0x0000 0000
SPRDAT	32 24	Base + 14h	R	Rx Data Register	0x0000 0000
SPPRE	32 24	Base + 18h	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 = power off 1 = power on	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
	[31:10]	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	[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	[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	[2]	Select an active high or active low clock 0 = active high 1 = active low	0
Clock Phase Select	[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	[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	[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 indcates 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.

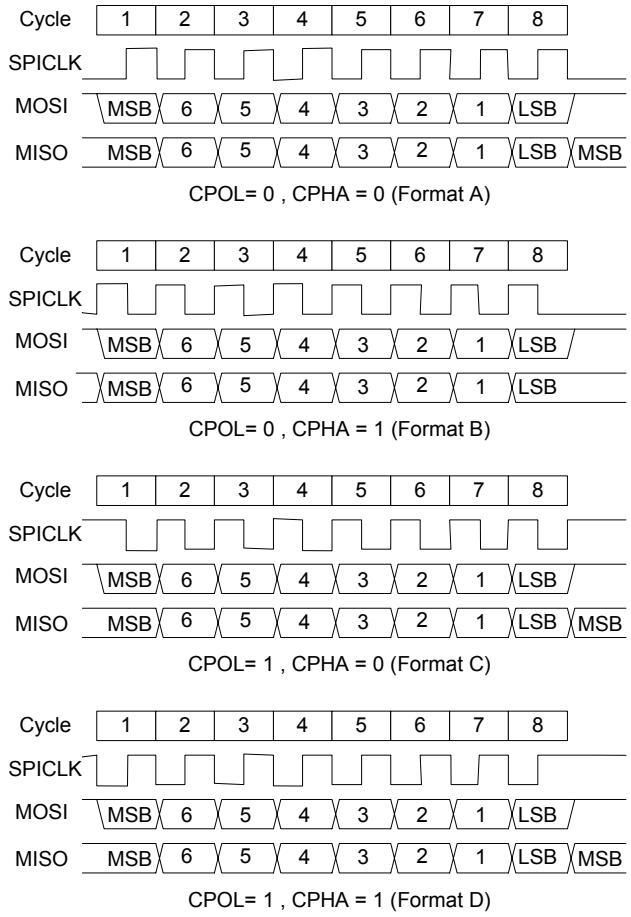


Fig. 1. SPI Transfer Format

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 0^{th} bit of SPCON is set, reading SPRDAT register makes data transfer start without needing to write 8'hFF 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 reginster 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 0^{th} bit of SPCON (TAGD) is inactive, write 8'hFF 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 8'hFF 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.

SPDIF

Preliminary Spec

Version 0.1

Aug. 29, 2003

Byeong-Woo Jeon

Media Player PT

System LSI Division

Device Solution Network

Contents

REVISION HISTORY

Date	Version Number	Update Contents
2003.08.29	0.1	Initial version

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

- SPDIF module only supports the consumer application in S5L840FX.
- 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

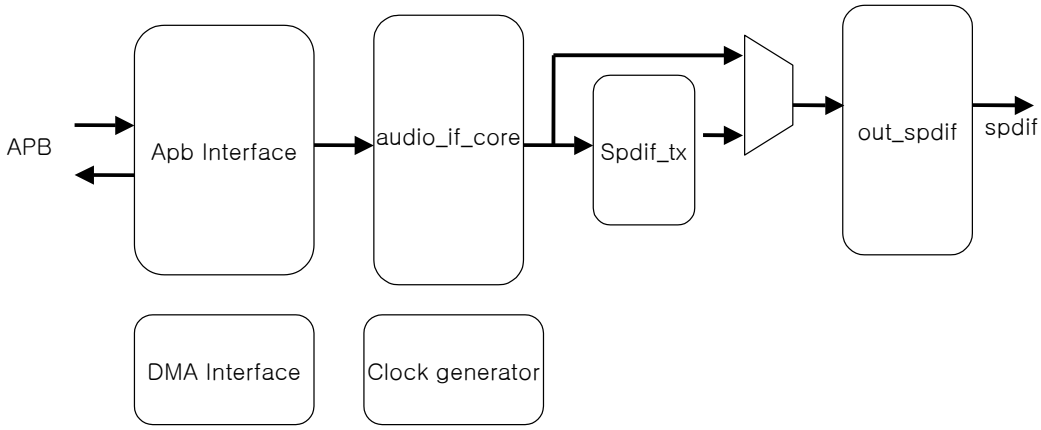


Fig 1. SPDIF Block Diagram

- APB interface block : defines register banks to control the driving of SPDIF 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
SPDIF Interface			
o-spdif-data	1	O	Spdif 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 SPDIF module

Registers

Base address : 0x3cb0_0000 (in 32bit), 0x39_6000 (in 24bit)

Name	Width	Address	R/W	Description	Reset
SPDCLKCON	32 24	Base + 00h	R/W	Clock Control Register	0x0000 0002
SPDCON	32 24	Base + 04h	R/W	Control Register	0x0000 0020
SPDBSTAS	32 24	Base + 08h	R/W	Burst Status Register	0x0000 0000
SPDCSTAS	32 24	Base + 0ch	R/W	Channel Status Register	0x2000 8000
SPDDAT	32 24	Base + 10h	W	SPDIF Data Buffer	0x0000 0000
SPDCNT	32 24	Base + 14h	R/W	Repetition Count Register	0x0000 0000

SPDIF 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

SPCLKCON	Bit	Description	Initial State
	[31:2]	Reserved	0
SPDIF clock down ready (read only)	[1]	0 = clock-down not ready 1 = clock-down ready	1
SPDIF power on	[0]	0 = power off 1 = power on	0

SPDIF 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 01 = 384fs 10 = 512fs 11 = reserved	0
PCM Data Size	[2:1]	00 = 16 bit 01 = 20 bit 10 = 24 bit 11 = reserved	0
PCM or Stream	[0]	0 = stream 1 = PCM	0

SPDIF 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

SPDIF 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

SPDIF 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
SPDIF Data	[23:0]	PCM or stream data	0

SPDIF Repetition Count Register (SPDCNT)

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

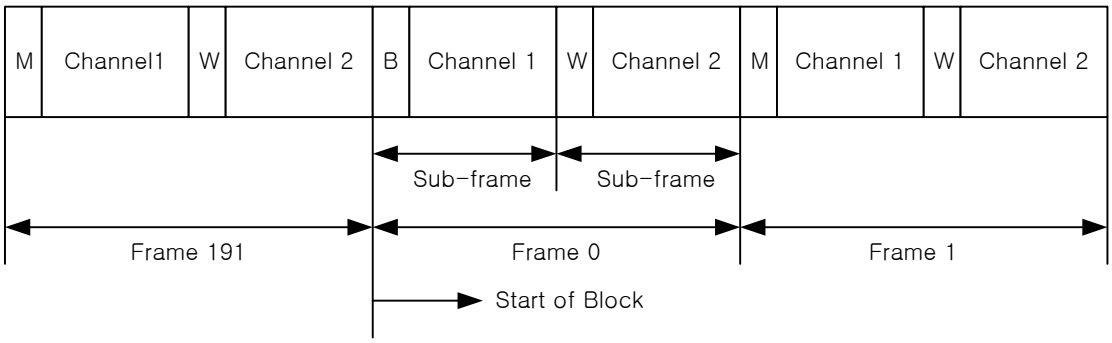
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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

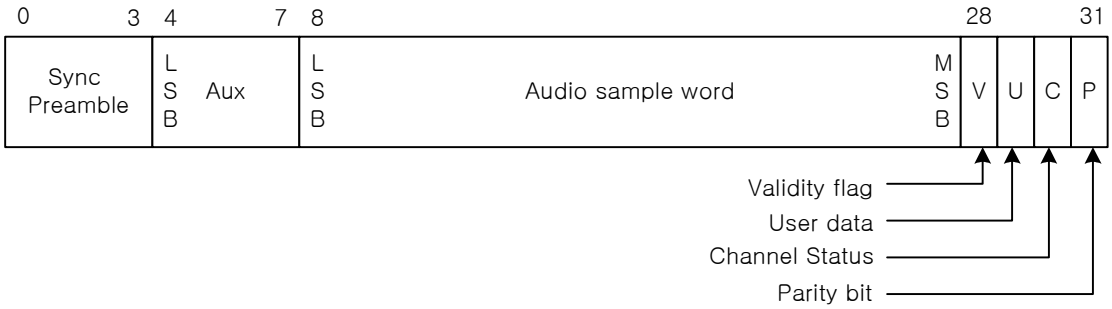
Frame Format

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).



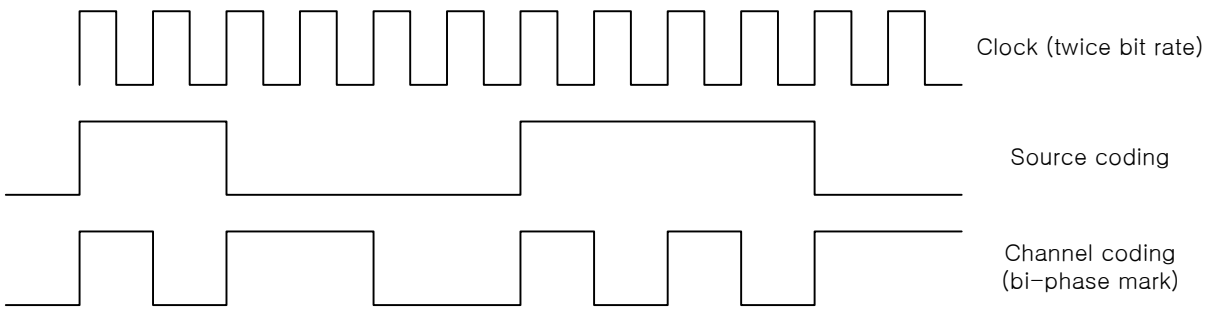
Sub-frame Format (IEC 60958)

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.



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".



Preamble

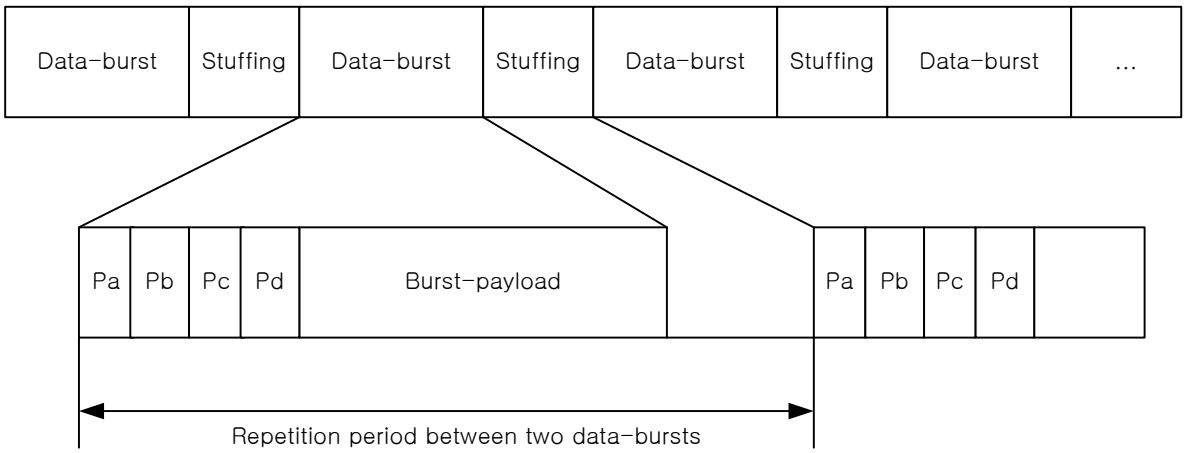
Preambles are specific patterns providing synchronization and identification of the sub-frames and blocks. A set of three preambles is used. These preambles are transmitted in the time allocated to four time slots(time slots 0 to 3) and are represented by eight successive states. The first state of the 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 biphase 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 2^{16}(=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.



SPDIF operation

Since the bit frequency of SPDIF is 128fs(fs : samling 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 S5L840FX 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.

LCD INTERFACE CONTROLLER

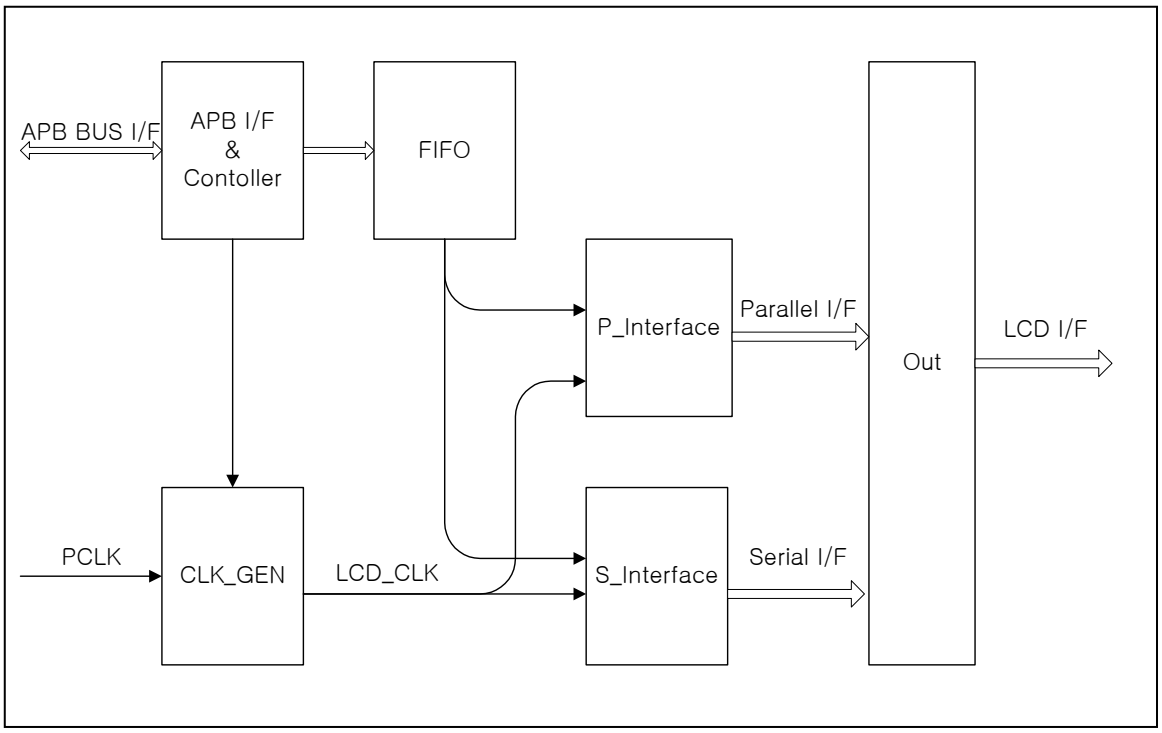
OVERVIEW

LCD Interface controller supports the interface between LCD controller with 6800series (Motorola) and 8080serise(intel).

FEATHER

- 8/4 bit parallel interface mode(6800/8080).
- 3 Pin / 4 Pin serial Interface mode.
- Support multiple frequencies internal clock for high and low speed controller.
- Contains an 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.

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. (1/2/4/8/16/32/64/128 divider).
3. FIFO : 9 bits x 16 depth 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

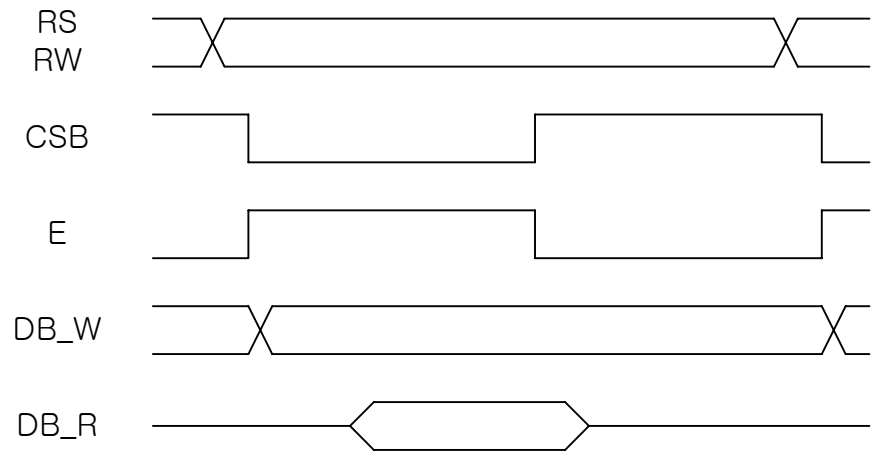
	Width	I/O	Description
TestMode	1	I	For scan enable signal
PCLK	1	I	Global clock
PRESETn	1	I	Global reset
APB Interface			
PSEL	1	I	Selection in APB
PENABLE	1	I	Enable in APB
PWRITE	1	I	Write / Read in APB
PADDR	4	I	Address in APB
PWDATA	9	I	Write data in APB
PRDATA	9	O	Read data in APB
LCD driver Interface			
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(8bit mode)
LCD_DB[5:4]	2	I/)	Bi-directional data bus(8bit/4bit mode)
LCD_DB[6]/SCLK	1	I/O	Bi-directional data bus & SCLK
LCD_DB[7]/SDO	1	I/O	Bi-directional data bus & SDO
Interrupt Interface			
INT_LCD	1	O	Interrupt Signal

LCD driver Interface

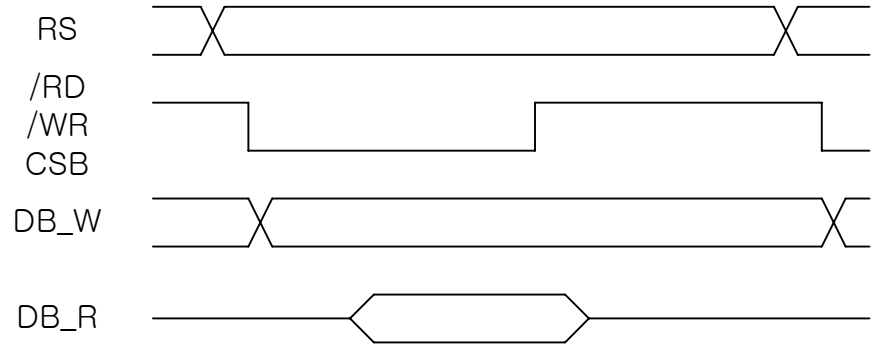
- LCD_RST : LCD driver reset signal.
- LCD_CSB : LCD driver chip select signal.
- LCD_RS: LCD driver register select signal.
  RS = 1 : DB0 to DB7 are display data.
  RS = 0 : DB0 to DB7 are control data.
- LCD_RW_WR : Read / Write execution control Pin.
  In 6800 Mode, High : Read, Low : Write.
  In 8080 Mode, Write enable sognal.
- 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.

TIMING DIAGRAM

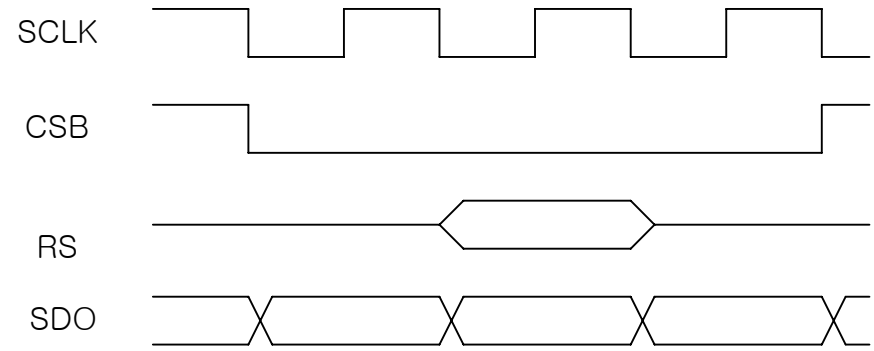
6800 Mode



8080 Mode



Serial Mode



REGISTERS

Name	Width	Address	R/W	Description	Rest Value
LCD_CON	32	0xhhhh 0000	R/W	Control register.	0x0000 0000
LCD_WCMD	32	0xhhhh 0004	W	Write command register.	0x0000 0000
LCD_WDATA	32	0xhhhh 0008	W	Write data register	0x0000 0000
LCD_RCMD	32	0xhhhh 000C	W	Read command register.	0x0000 0000
LCD_RDATA	32	0xhhhh 0010	W	Read data register.	0x0000 0000
LCD_DBUFF	32	0xhhhh 0014	R	Read Data buffer	0x0000 0000
LCD_INTCON	32	0xhhhh 0018	R/W	Interrupt control register	0x0000 0000
LCD_STATUS	32	0xhhhh 001C	R	LCD Interface status	0x0000 0000
LCD_PHTIME	32	0xhhhh 0020	R/W	Phase time register	0x0000 0066

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
9	LCD_ON	R/W	LCD Interface On.	0
8:7	W_LEN	R/W	APB Bus interface word length. 00 : 8 bit. 01 : 16 bit 10 : 32 bit	00
6	ENDIAN	R/W	APB Bus word Endian. 0 : Little endian. 1 : Big endian. When W_LEN = 0, No effect.	0
5	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	0

			PS1 = 0 : 8080-series parallel MPU interface. PS2 = 1 : 6800-series parallel MPU interface.	
4	PS0	R/W	Parallel/serial data select. When 0 Serial MPU Interface. When 1 Parallel MPU interface.	0
3	BUS_M	R/W	Bus mode Select at parallel Interface. BUS_M = 0 : 8bit Data Bus. BUS_M = 1 : 4 bit Data Bus.	0
2:0	CLK_SEL	R/W	LCD clock select. 000 : PCLK / 1, 001 : PCLK / 2 010 : PCLK / 4, 011 : PCLK / 8 100 : PCLK / 16, 101 : PCLK / 32 110 : PCLK / 64, 111 : PCLK / 128	0

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	Rest Value
7:0	WCMD	W	Write LCD driver command.	0x000

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	Rest Value
31:0*	WDATA	W	Write LCD driver data.	0x000

- 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	Rest Value
7:0	RCMD	W	Read command LCD driver command.] Dummy value.	0x000

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	Rest Value
7:0	RDATA	W	Read command LCD driver data. Dummy value.	0x000

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	Rest Value
7:0	DBUFF	W	Read LCD driver data buff.	0x000

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	Rest Value
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	EMPTY	R	FIFO is empty.	0
0	READON	R	Read operation done.	1

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	Rest 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	Rest Value
7:4	PH1_TIME	R/W	Phase 1 time register	6
3:0	PH2_TIME	R/W	Phase 1 time register	0

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 Registe.

PCLK	CLK_SEL	tPW time unit (ns)
100 MHz	000 (x 1)	10
	001 (x 1/2)	20
	010 (x 1/4)	40
	011 (x 1/8)	80
	100 (x 1/16)	160
	101 (x 1/32)	320
	110 (x 1/64)	640
	111 (x 1/128)	1280
50 MHz	000 (x 1)	20
	001 (x 1/2)	40
	--	--
	110 (x 1/64)	1280
	111 (x 1/128)	2560

tPW = (PH1_TIME Value + 1) x tPW time unit (Phase 1)

tPW = (PH2_TIME Value + 1) x tPW 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 indiacating the data or command.
This value is set automatically as to writing to which register.

To write command to LCD Driver, CPU have to write to LCD_WCMD. At this time, The value of RS is set automatically as zero. LCD interfase 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.