1 PRODUCT OVERVIEW

CALMRISC OVERVIEW

The S3CB018/FB018 single-chip CMOS microcontroller is designed for high performance using Samsung’ s newest 8-bit CPU core, CalmRISC.

CalmRISC is an 8-bit low power RISC microcontroller. Its basic architecture follows Harvard style, that is, it has separate program memory and data memory. Both instruction and data can be fetched simultaneously without causing a stall, using separate paths for memory access. Represented below is the top block diagram of the CalmRISC microcontroller.

S3CB018/FB018 OVERVIEW

FEATURES SUMMARY

CPU

- 8-Bit RISC architecture

Memory

- ROM: 4 Kword (8 K-byte)
- RAM: 3072 (1024+2048) byte
  1024 (X-memory) byte
  2048 (Y-memory) byte

Stack

- size: maximum 16 (word)-level

26 I/O Pins

- I/O: 26 pins, including 8 S/W open drain pins

8-Bit Basic Timer

- Programmable interval timer
- 8 kinds of clock source

Watchdog Timer

- System reset when 11-bit counter overflows

16-Bit Timer/Counter

- Programmable interval timer
- Two 8-bit timer counter mode and one 16-bit timer counter mode, selectable by S/W

Watch Timer

- Real time clock or interval time measurement
- Four frequency outputs for buzzer sound

8-Bit Serial I/O Interface

- 8-bit transmit/receive mode
- 8-bit receive mode
- LSB first or MSB first transmission selectable
- Internal and external clock source

16-Bit Serial I/O Interface

- 16-bit transmit/receive mode
- External clock source

Coprocessor

- MAC 816
- 8 x 16, 16 x 16 Multiply and Accumulation
- Arithmetic operation

Two Power-Down Modes

- Idle mode: only CPU clock stop
- Stop mode: selected system clock and CPU clock stop

Oscillation Sources

- Crystal and Ceramic (0.4-20MHz), RC Oscillation
- Programmable oscillation source

Instruction Execution Times

- 50ns at 20MHz for 1 cycle instruction
- 100ns at 20MHz for 2 cycle instruction

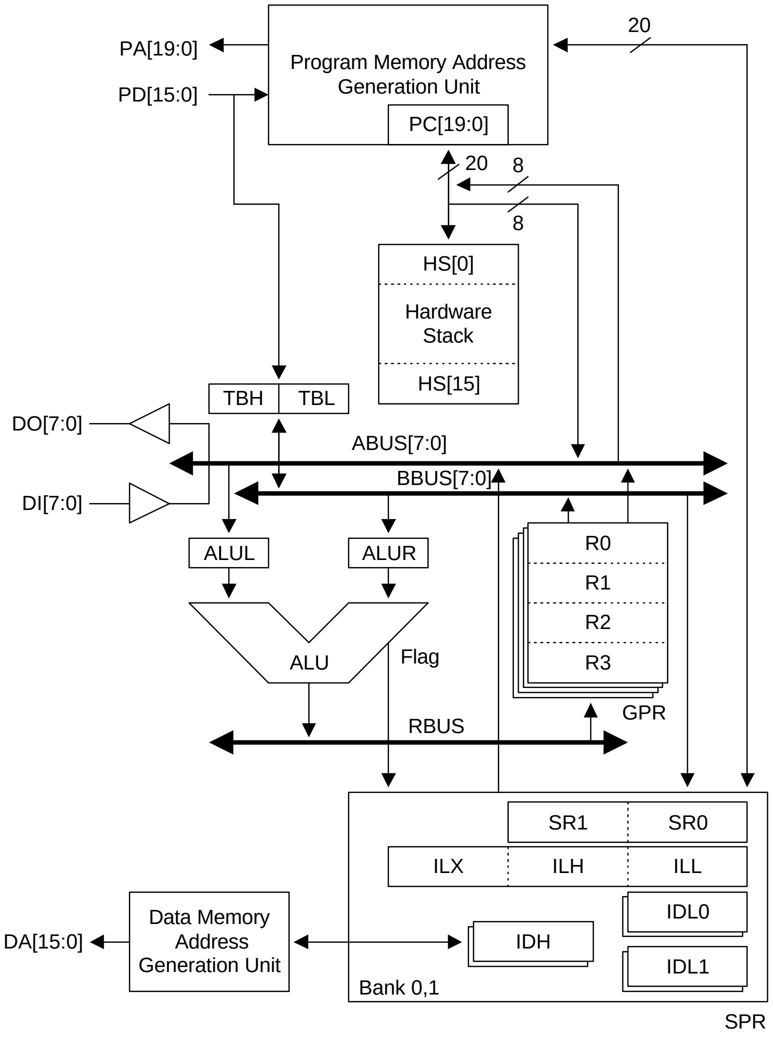


Figure 1-1. Top Block Diagram of CalmRISC

The CalmRISC building blocks consist of:

- An 8-bit ALU
- 16 general purpose registers (GPR)
- 11 special purpose registers (SPR)
- 16-level hardware stack
- Program memory address generation unit
- Data memory address generation unit

16 GPR’ s are grouped into four banks (Bank0 to Bank3) and each bank has four 8-bit registers (R0, R1, R2, and R3). SPR’ s, designed for special purposes, include status registers, link registers for branch-link instructions, and data memory index registers. The data memory address generation unit provides the data memory address (denoted as *DA[15:0]* in the top block diagram) for a data memory access instruction. Data memory contents are accessed through *DI[7:0]* for read operations and *DO[7:0]* for write operations. The program memory address generation unit contains a program counter, PC[19:0], and supplies the program memory address through *PA[19:0]* and fetches the corresponding instruction through *PD[15:0]* as the result of the program memory access. CalmRISC has a 16-level hardware stack for low power stack operations as well as a temporary storage area.

CalmRISC has a 3-stage pipeline as described below:

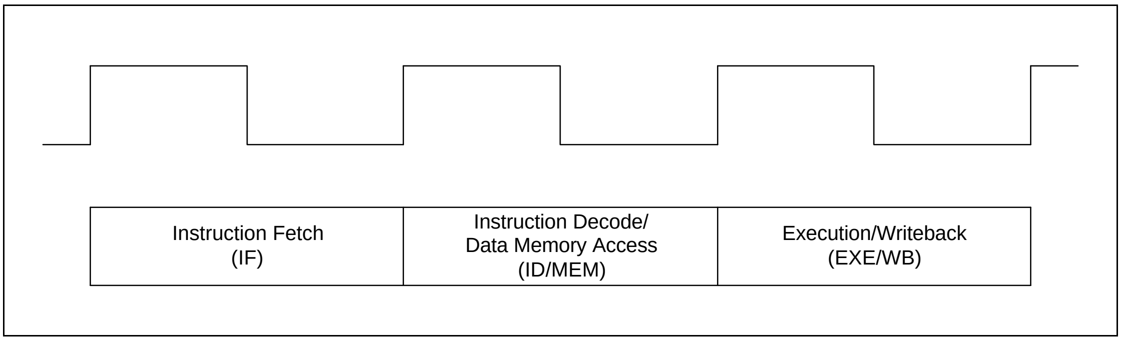


Figure 1-2. CalmRISC Pipeline Diagram

As can be seen in the pipeline scheme, CalmRISC adopts a register-memory instruction set. In other words, data memory where *R* is a GPR, can be one operand of an ALU instruction as shown below:

The first stage (or cycle) is Instruction Fetch stage (IF for short), where the instruction pointed to by the program counter, PC[19:0] , is read into the Instruction Register (IR for short). The second stage is Instruction Decode and Data Memory Access stage (ID/MEM for short), where the fetched instruction (stored in IR) is decoded and data memory access is performed, if necessary. The final stage is Execute and Write-back stage (EXE/WB), where the required ALU operation is executed and the result is written back into the destination registers.

Since CalmRISC instructions are pipelined, the next instruction fetch is not postponed until the current instruction is completely finished, but is performed immediately after the current instruction fetch is done. The pipeline stream of instructions is illustrated in the following diagram.

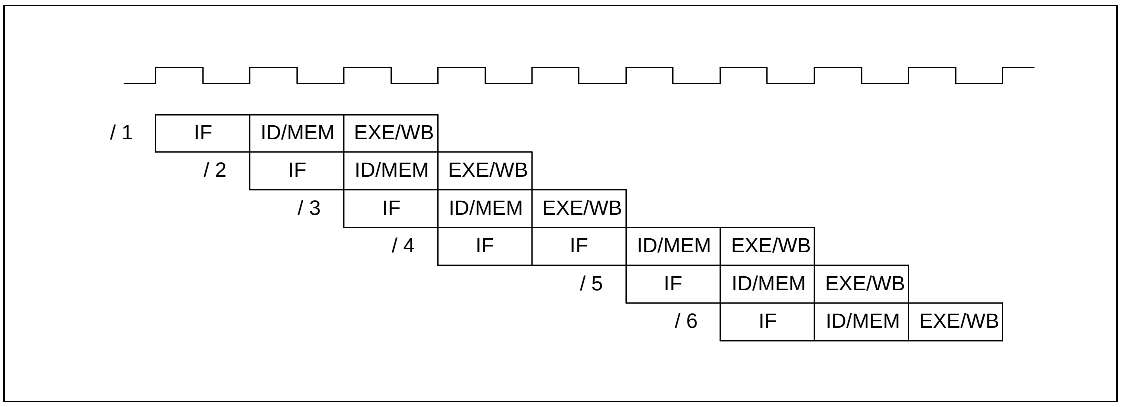


Figure 1-3. CalmRISC Pipeline Stream Diagram

Most CalmRISC instructions are 1-word instructions, while same branch instructions such as “LCALL” and “LJT” instructions are 2-word instructions. In Figure 1-3, the instruction, I_4, is a long branch instruction and it takes two clock cycles to fetch the instruction. As indicated in the pipeline stream, the number of clocks per instruction (CPI) is 1 except for long branches, which take 2 clock cycles per instruction.

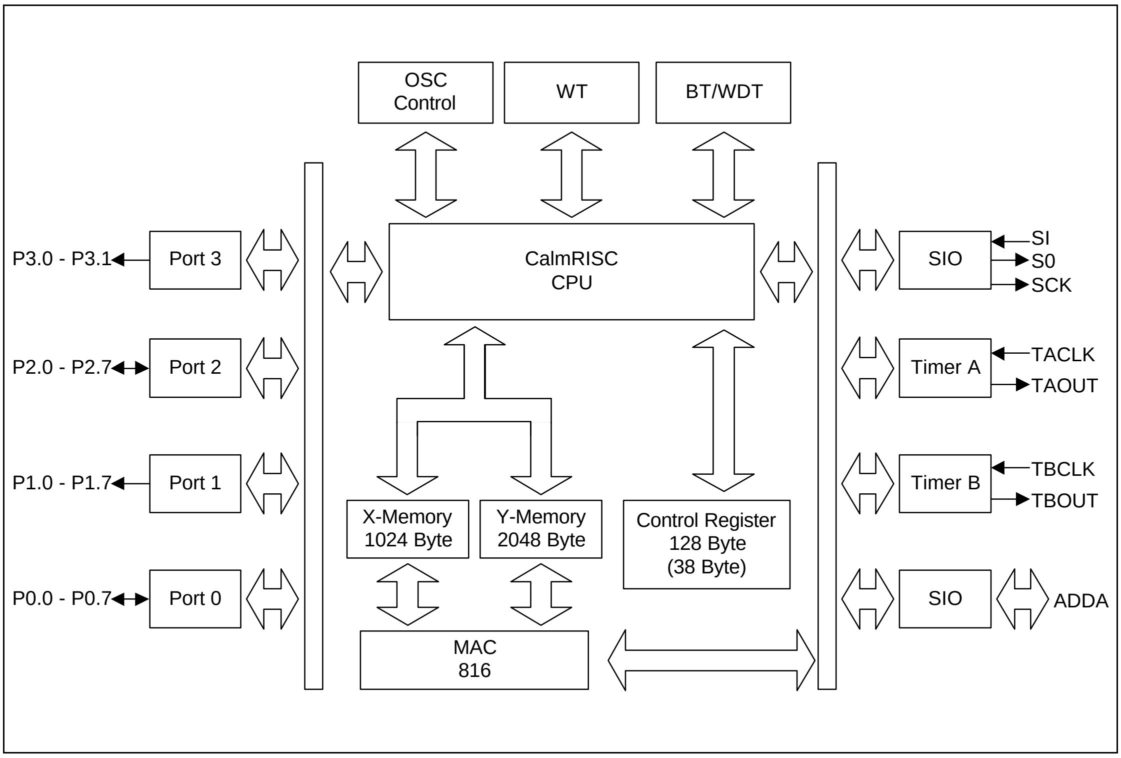


Figure 1-4. S3CB018/FB018 Block Diagram

PIN ASSIGNMENTS

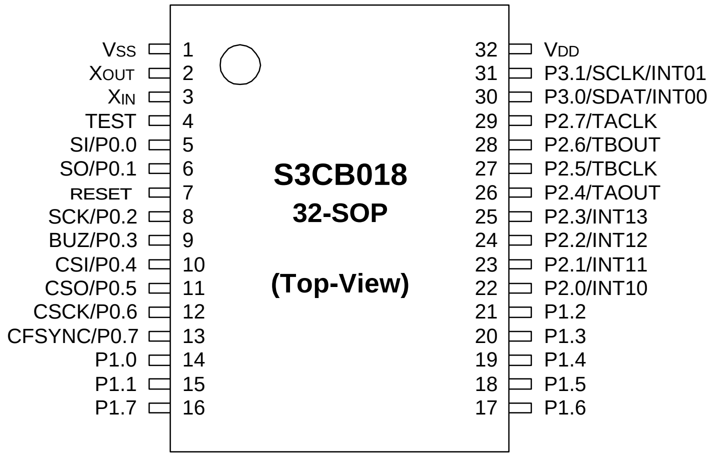


Figure 1-5. 32-SOP Pin Assignment

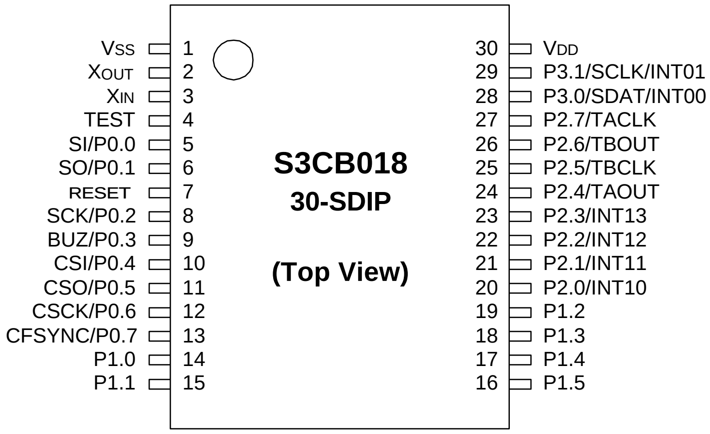


Figure 1-6. 30-SDIP Pin Assignment

I/O PIN DESCRIPTION

Table 1-1. S3CB018/FB018 Pin Descriptions (32-SOP)

Pin Name	Pin Type	Pin Description	Circuit Type	Share Pins
P0.0-P0.7	I/O	I/O port with bit programmable pins; Input and output mode are selectable by software; Software assignable pull-up. P0.4-P0.7 can be used as inputs for comparator input CIN0-CIN3.; Alternately they can be used as SI, SO, SCK, BUZ, CSI, CSO, CSCK, CFSYNC.	D-2 F-10	SI, SO, SCK BUZ, CSI, CSO, CSCK, CFSYNC
P1.0-P1.7	O	Output port with bit programmable pins; Push-pull output mode and open-drain output mode are selected by software; Software assignable pull-up.	E-2	
P2.0-P2.7	I/O	I/O port with bit programmable pins; Input and output mode are selectable by software; Software assignable pull-up; P2.0-P2.3 can be used as inputs for external interrupts INT10-INT13. (with noise filter) ; Alternately they can be used as TAOUT, TACLK or TBOUT, TBCLK.	D-4 D-2	INT10-INT13 TAOUT TACLK TBOUT TBCLK
P3.0-P3.1	I/O	I/O port with bit programmable pins; Input or output mode selected by software; software assignable pull-up; P3.0-P3.1 can be used as inputs for external interrupts INT00-INT01. (with noise filter and interrupt polarity control)	D-4	INT00-INT01

Table 1-2. S3CB018/FB018 Pin Descriptions (30-SDIP)

Pin Name	Pin Type	Pin Description	Circuit Type	Share Pins
P0.0-P0.7	I/O	I/O port with bit programmable pins; Input and output mode are selectable by software; Software assignable pull-up. P0.4-P0.7 can be used as SI, SO, SCK, BUZ, CSI, CSO, CSCK, CFSYNC, Alternately.	D-2 F-10	SI, SO, SCK BUZ, CSI, CSO, CSCK, CFSYNC
P1.0-P1.5	O	O port with bit programmable pins; Push-pull output mode and open-drain output mode are selected by software; Software assignable pull-up.	E-2	
P2.0-P2.7	I/O	I/O port with bit programmable pins; Input and output mode are selectable by software; Software assignable pull-up; P2.0-P2.3 can be used as inputs for external interrupts INT10-INT13. (with noise filter); Alternately they can be used as TAOUT, TACLK or TBOUT, TBCLK.	D-4 D-2	INT10-INT13 TAOUT TACLK TBOUT TBCLK
P3.0-P3.1	I/O	I/O port with bit programmable pins; Input or output mode selected by software; software assignable pull-up; P3.0-P3.1 can be used as inputs for external interrupts INT00-INT01. (with noise filter and interrupt polarity control)	D-4	INT00-INT01

NOTE: In S3CB018/FB018, the CSI, CSO, CSCK, CFSYNC pins are shared with P0.7-P0.4.

Table 1-3. I/O Pin Description

Pin Name	Pin Type	Description
CSI	I	AD/DA Serial Input (from codec)
CSO	O	AD/DA Serial Output (to codec)
CSCK	I	AD/DA Serial Clock (from codec)
CFSYNC	I	AD/DA Sync signal (from codec)
SI	I/O	Serial data input
SO	I/O	Serial data output
SCK	I/O	Serial I/O interface clock signal
BUZ	I/O	0.5 kHz, 1 kHz, 2 kHz, or 4 kHz frequency output at 4.19 MHz for buzzer sound
INT10-INT13	I	External interrupts. Stop release. Can’ t be masked by S/W individually but wholly.
TAOUT	I/O	Timer A interval mode output
TACLK	I/O	Timer A counter external clock input
TBOUT	I/O	Timer B interval mode output
TBCLK	I/O	Timer B counter external clock input
INT00-INT01	I	External interrupts. Stop release. Can be masked by S/W individually.
SDAT	I	Serial data for Programmable memory
SCLK	I	Serial clock for Programmable memory
VDD	–	Power supply
VSS	–	Ground
TEST	–	Test signal input
RESET	I	Reset signal
X_{IN}, X_{OUT}	–	Crystal, ceramic and RC oscillator signal for system clock (For external clock input, use X_{IN} and input X_{IN}'s reverse phase to X_{OUT})

PIN ASSIGNMENTS

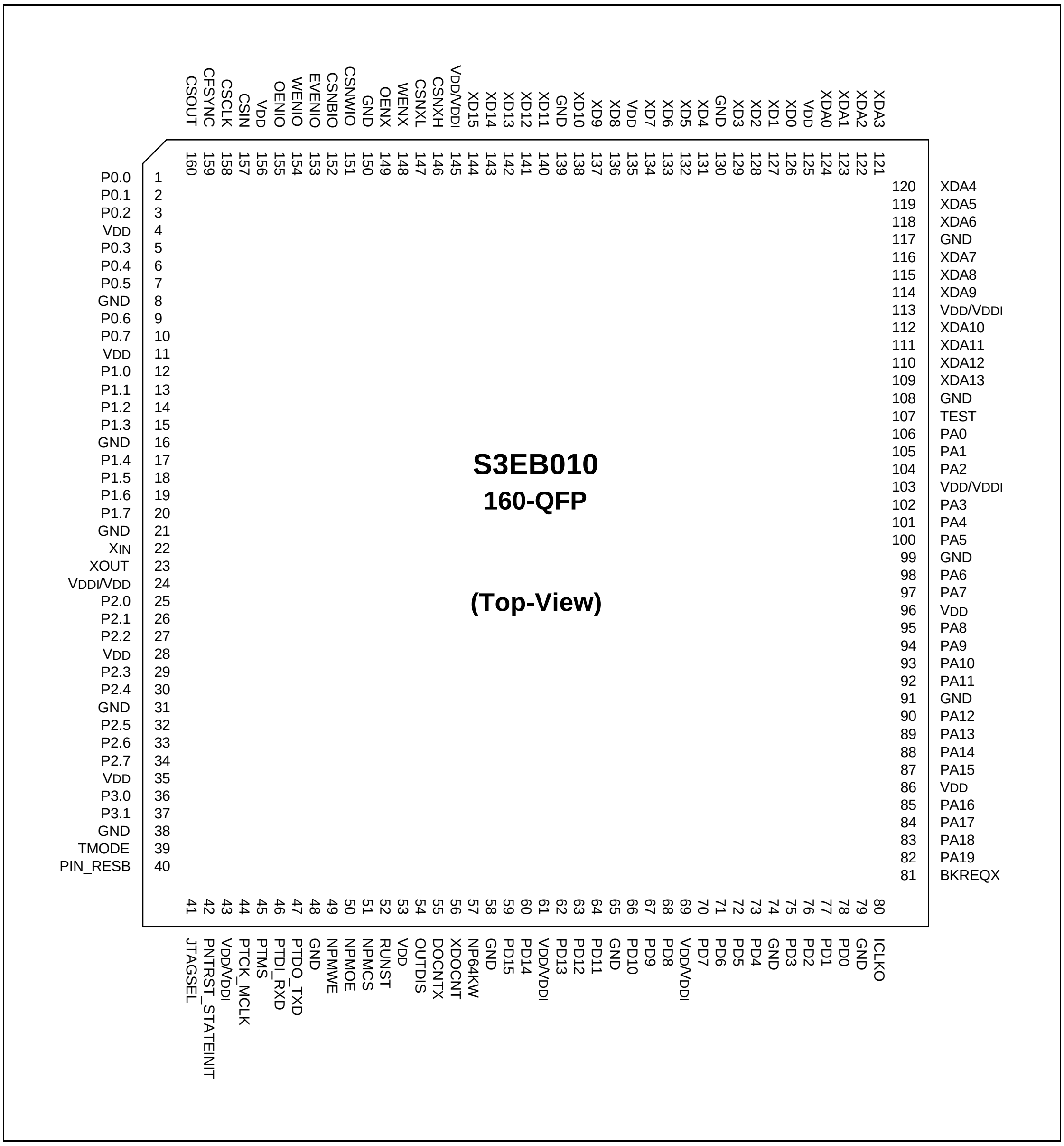


Figure 1-7. S3EB010 Pin Diagram

Table 1-4. Evaluation Chip Pin Descriptions

No.	Pin Name	Pin Type	Description
1-3 5-9	P0.0-P0.2 P0.3-P0.7	I/O	Port 0
12-15 17-20	P1.0-P1.3 P1.4-P1.7	O	Port 1
22	X_{IN}	I	Clock In
23	X_{OUT}	O	Clock Out
25-27 29, 30 32, 34	P2.0-P2.2 P2.3, P2.4 P2.5-P2.7	I/O	Port 2
36, 37	P3.0, P3.1	I/O	Port 3
39	TMODE	I	Test Mode pin; 1: skip warm-up time, 0: normal mode
40	PIN_RESB	I	Asynchronous reset, active low
41	JTAGSEL	I	JTAG mode select; 1: parallel, 0: serial
42	PNTRST_STSTEINIT	I	JTAG/UART pin
44	PTCK_MCLK	I	JTAG/UART pin
45	PTMS	I	JTAG/UART pin
46	PTDI_RXD	I	JTAG/UART pin
47	PTDO_TXD	O	JTAG/UART pin
49	NPMWE	O	Program Memory Write Enable, active low
50	NPMOE	O	Program Memory Output Enable, active low
51	NPMCS	O	Program Memory Chip Select, active low
52	RUNST	O	Run Status Indicator
54	OUTDIS	I	I/O PAD Disable for debugger
55	DOCNTX	I	Data Bus Output Control
56	XDOCNTX	I	External X-Memory Data Bus Output Control
57	NPM64KW	I	Up to 64KW Program Memory, active low
59, 60 62-64 66-68 70-73 75-78	PD15-PD14 PD13-PD11 PD10-PD8 PD7-PD4 PD3-PD0	I/O	Program Memory Data Bus

Table 1-4. Evaluation Chip Pin Descriptions (Continued)

No.	Pin Name	Pin Type	Description
80	ICLKO	O	ICLK Output
81	BKREQX	I	Break input for debugger
82-85 87-90 92-95 97, 98 100-102 104-106	PA19-PA16 PA15-PA12 PA11-PA8 PA7, PA6 PA5-PA3 PA2-PA0	O	Program Memory Address
107	TEST	I	Test pin for debugger
109-112 114-116 118-124	XDA13-XDA10 XDA9-XDA7 XDA6-XDA0	O	External X-Memory Address
126-129 131-134 136-138 140-144	XD0-XD3 XD4-XD7 XD8-XD10 XD11-XD15	I/O	External X-Memory Data Bus
146	CSNXH	O	External X-Memory High Byte Chip Select, active low
147	CSNXL	O	External X-Memory Low Byte Chip Select, active low
148	WENX	O	External X-Memory Write Enable, active low
149	OENX	O	External X-Memory Output Enable, active low
151	CSNWIO	O	External I/O Word Chip Select, active low
152	CSNBIO	O	External I/O Byte Chip Select, active low
153	EVENIO	O	External I/O Even Indicator; 1:Even, 0: Odd
154	WENIO	O	External I/O Write Enable, active low
155	OENIO	O	External I/O Output Enable, active low
157	CSIN	I	AD / DA Serial Input (from codec)
158	CSCLK	I	AD / DA Serial Clock (from codec)
159	CFSYNC	I	AD / DA Sync signal (from codec)
160	CSOUT	O	AD / DA Serial Output (to codec)
VDD	Power supply	–	4, 11, 24, 28, 35, 43, 53, 61, 69, 86, 96, 103, 113 125, 135, 145, 156
GND	Ground	–	8, 16, 21, 31, 38, 48, 58, 65, 74, 79, 91, 99, 108, 117 130, 139, 150

PIN CIRCUIT DIAGRAMS

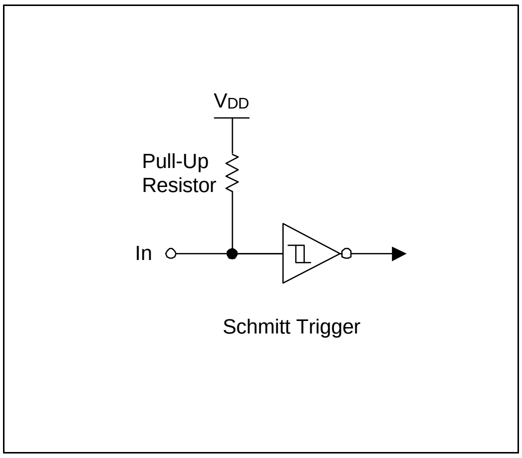


Figure 1-8. Pin Circuit Type B (RESET)

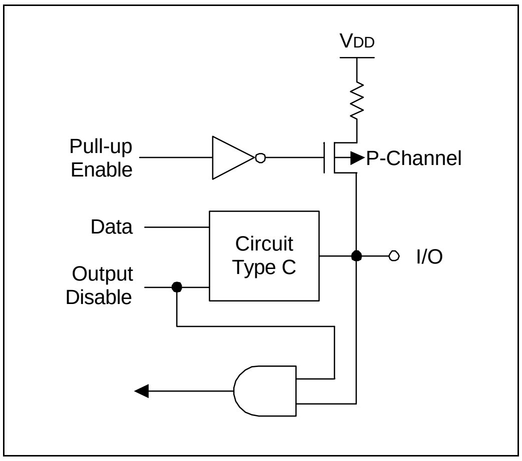


Figure 1-10. Pin Circuit Type D-2
(P0.0-P0.3, P2.4-P2.7)

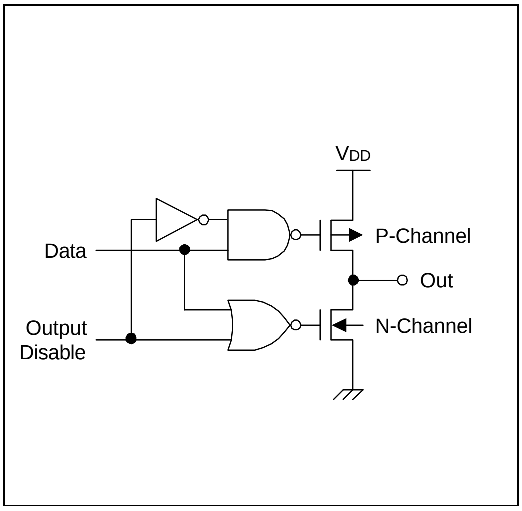


Figure 1-9. Pin Circuit Type C

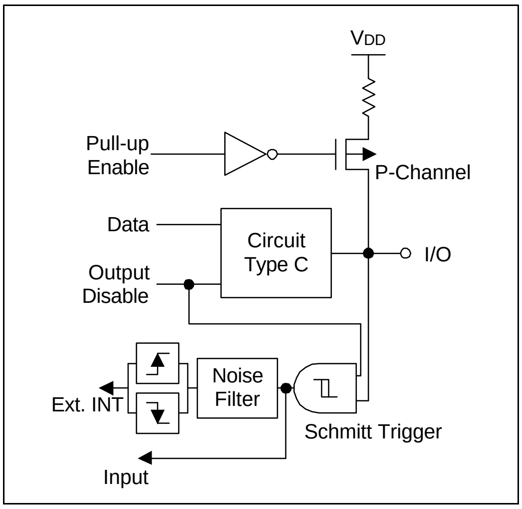


Figure 1-11. Pin Circuit Type D-4 (P2.0-P2.3, P3)

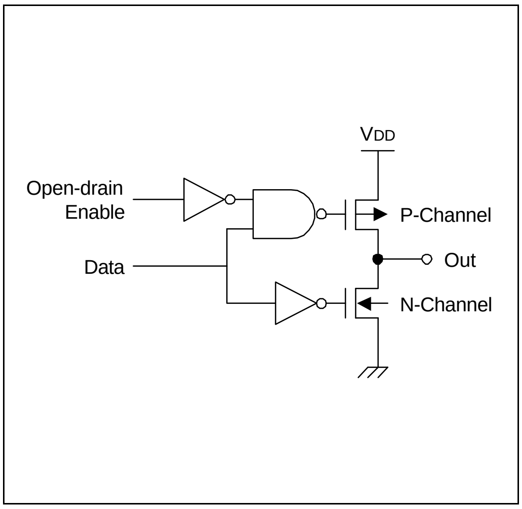


Figure 1-12. Pin Circuit Type E-2 (P1)

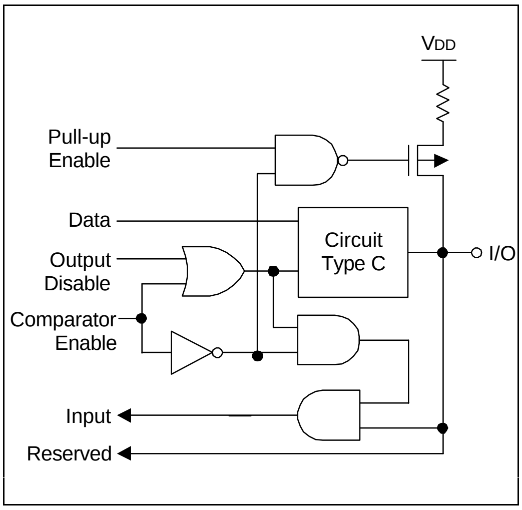


Figure 1-13. Pin Circuit Type F-10 (P0.4-P0.7)

2 ADDRESS SPACE

OVERVIEW

CalmRISC has 20-bit program address lines, *PA[19:0]*, which support up to 1M words of program memory. The 1 M word program memory space is divided into 256 pages, and each page is 4K word long, as shown in the next page. The upper 8 bits of the program counter, PC[19:12], point to a specific page and the lower 12 bits, PC[11:0], specify the offset address of the page.

CalmRISC also has 16-bit data memory address lines, DA[15:0], which support up to 64K bytes of data memory. The 64K byte data memory space is divided into 256 pages, and each page has 256 bytes. The upper 8 bits of the data address, DA[15:8], point to a specific page, and the lower 8 bits, DA[7:0], specify the offset address of the page.

PROGRAM MEMORY (ROM)

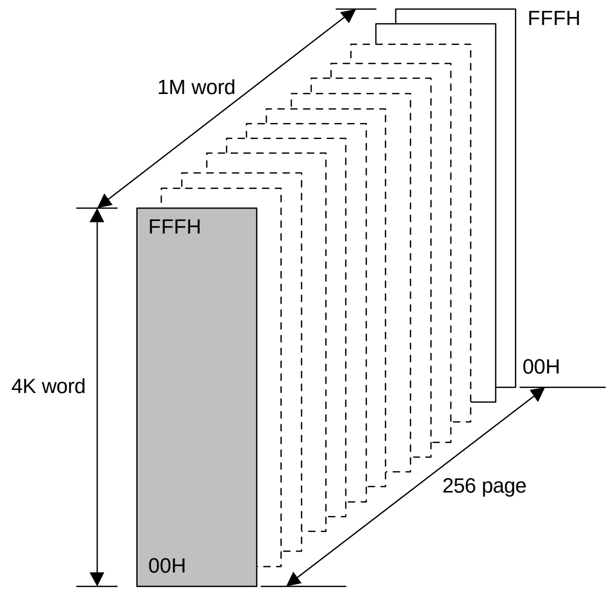


Figure 2-1. Program Memory Organization

For example, if PC[19:0] = 5F79AH, the page index pointed to by PC is 5FH and the offset in the page is 79AH. If the current PC[19:0] = 5EFFFH and the instruction pointed to by the current PC, i.e., the instruction at the address 5EFFFH is *not* a branch instruction, the next PC becomes 5E000H, *not* 5F000H. In other words, the instruction sequence wraps around at the page boundary, unless the instruction at the boundary (in the above example, at 5EFFFH) is a long branch instruction. The only way to change the program page is by long branches (LCALL, LLNK, and LJP), where the absolute branch target address is specified. For example, if the current PC[19:0] = 047ACH (the page index is 04H and the offset is 7ACH) and the instruction pointed to by the current PC, i.e., the instruction at the address 047ACH, is “LJP A507FH” (jump to the program address A507FH), then the next PC[19:0] = A507FH, which means that the page and the offset are changed to A5H and 07FH, respectively. On the other hand, the short branch instructions cannot change the page indices.

Suppose the current PC is 6FFFEH and its instruction is “JR 5H” (jump to the program address PC + 5H). Then the next instruction address is 6F003H, not 70003H. In other words, the branch target address calculation also wraps around with respect to a page boundary. This situation is illustrated below:

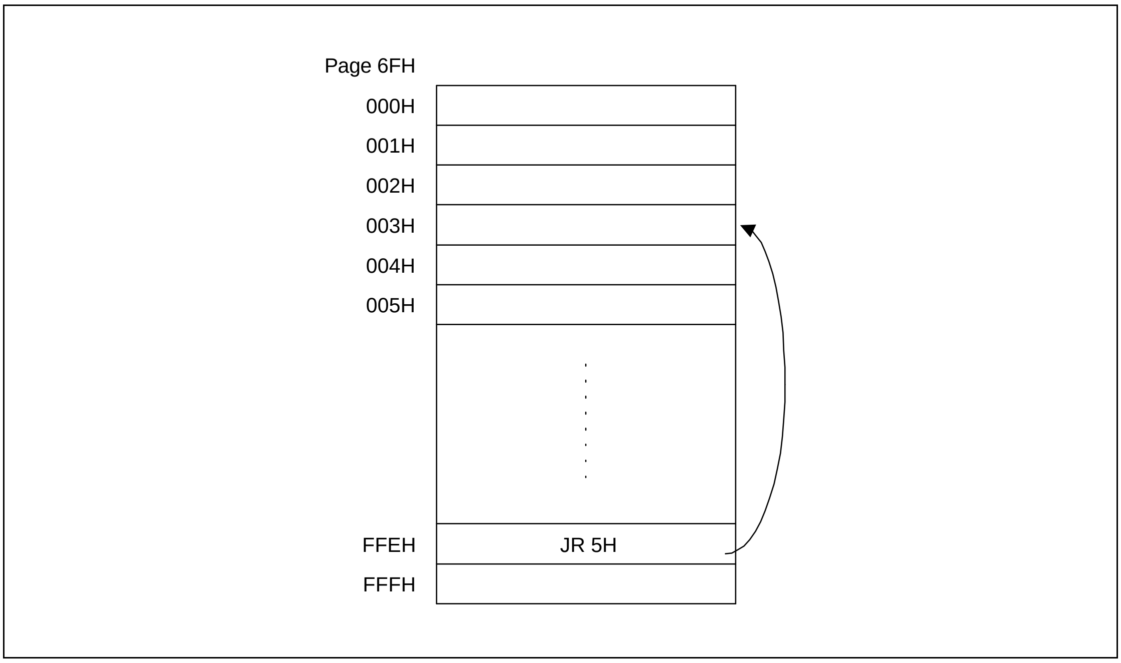
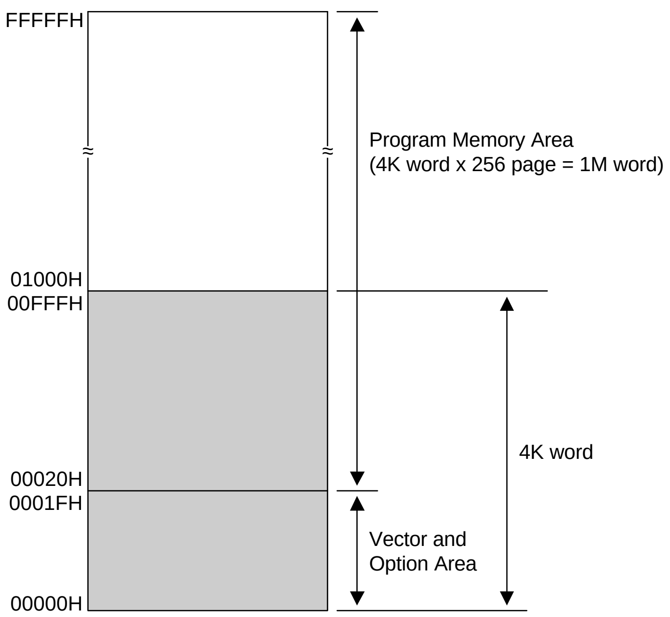


Figure 2-2. Relative Jump Around Page Boundary

Programmers do not have to manually calculate the offset and insert extra instructions for a jump instruction across page boundaries. The compiler and the assembler for CalmRISC are in charge of producing appropriate codes for it.



NOTE: S3CB018 has in total 4K word (8K byte) program memory area.

Figure 2-3. Program Memory Layout

Addresses from 00000H to 00004H are used as the vector addresses for the exceptions, and 0001EH and 0001FH are used for the option only. Aside from these addresses others are reserved in the vector and option areas. Program memory area from addresses 00020H to FFFFFH can be used for normal programs.

S3CB018/FB018's program memory is 4K word (8K byte) long, so addresses from 00020H to 00FFFH are the program memory area.

ROM CODE OPTION (RCOD_OPT)

Just after power on, the oscillator becomes one of the four types of oscillators, RC, LSX, MSX and HSX**,** by the value of RCOD_OPT**,** which is located at 0001EH and 0001FH in program memory. The address 0001EH and the high byte of address 0001FH are not used in S3CB018/FB018.

For example, if you program as below:

```
rcod_opt      001EH, 0FFFFH
rcod_opt      001FH, 0FF80H
```

— If the watchdog timer overflows, the hardware reset exception would be serviced.

— The basic timer counter clock input is fxx/2048 after reset (by bit.6, .5, .4).

— BTCNT.5 is the CPU stop release signal (by bit.3).

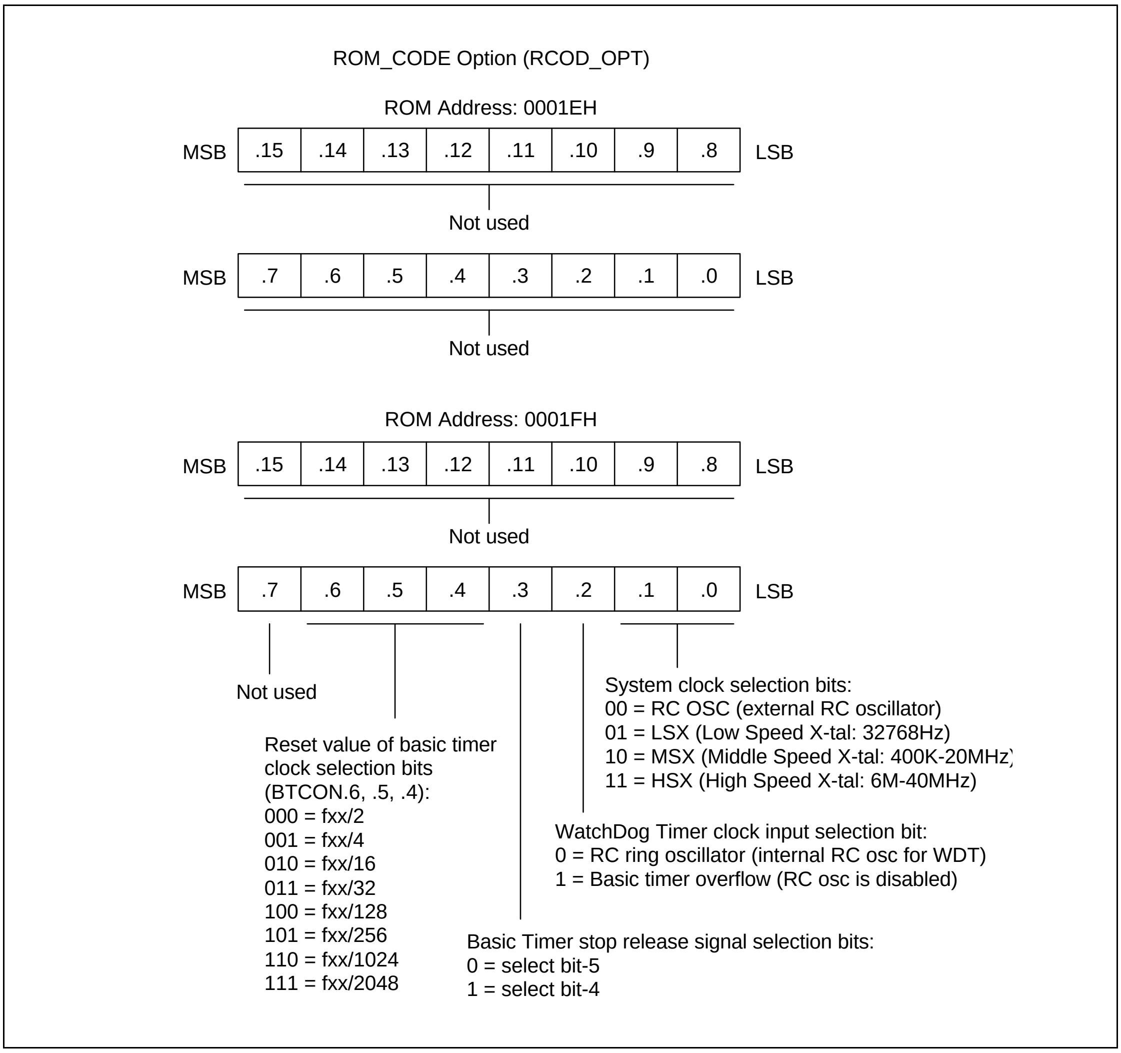
— Basic timer overflow is the watchdog timer clock input (by bit.2).

— HSX (high speed oscillator) is the main system oscillator (by bit.1, .0).

If you do not program any values in these option, then the default value is “1”. Therefore, in these cases the address 0001EH becomes the value of “FFH”.

In S3EB010, the RCOD_OPT is not available. Here the basic timer counter’s clock input is fxx/2048; the stop release signal is the bit.5 of basic timer counter. The main system oscillator is the crystal oscillator(0.4–20MHz). The watchdog timer overflow produces the reset operation. In this state, it is fixed and not programmable in S3EB010.

The RCOD_OPT is programmable only in S3CB018/FB018.



System clock selection bits:
00 = RC OSC (external RC oscillator)
01 = LSX (Low Speed X-tal: 32768Hz)
10 = MSX (Middle Speed X-tal: 400K-20MHz)
11 = HSX (High Speed X-tal: 6M-40MHz)

Reset value of basic timer clock selection bits (BTCON.6, .5, .4):
000 = fxx/2
001 = fxx/4
010 = fxx/16
011 = fxx/32
100 = fxx/128
101 = fxx/256
110 = fxx/1024
111 = fxx/2048

WatchDog Timer clock input selection bit:
0 = RC ring oscillator (internal RC osc for WDT)
1 = Basic timer overflow (RC osc is disabled)

Basic Timer stop release signal selection bits:
0 = select bit-5
1 = select bit-4

Figure 2-4. ROM Code Option (RCOD_OPT)

DATA MEMORY ORGANIZATION

The total data memory address space is 64K byte, addressed by *DA[15:0],* which is also divided into 256 pages, Each page consists of 256 bytes as shown below.

The X and Y memories are for the mac816 coprocessor. CalmRISC finds no meaning in dividing memories such as X and Y memories.
Mac816 views the peripheral control register area from 00H to 3FH; the X memory is from 40H to 23FH and the Y memory is from 4000H to 43FFH.

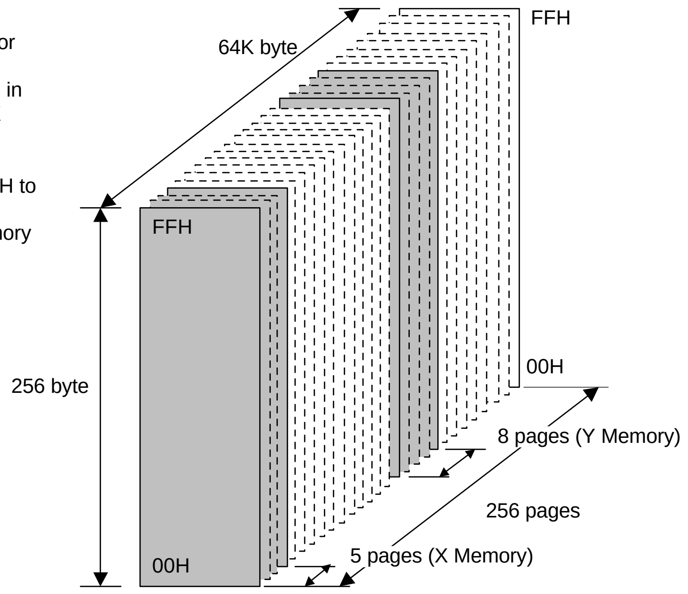
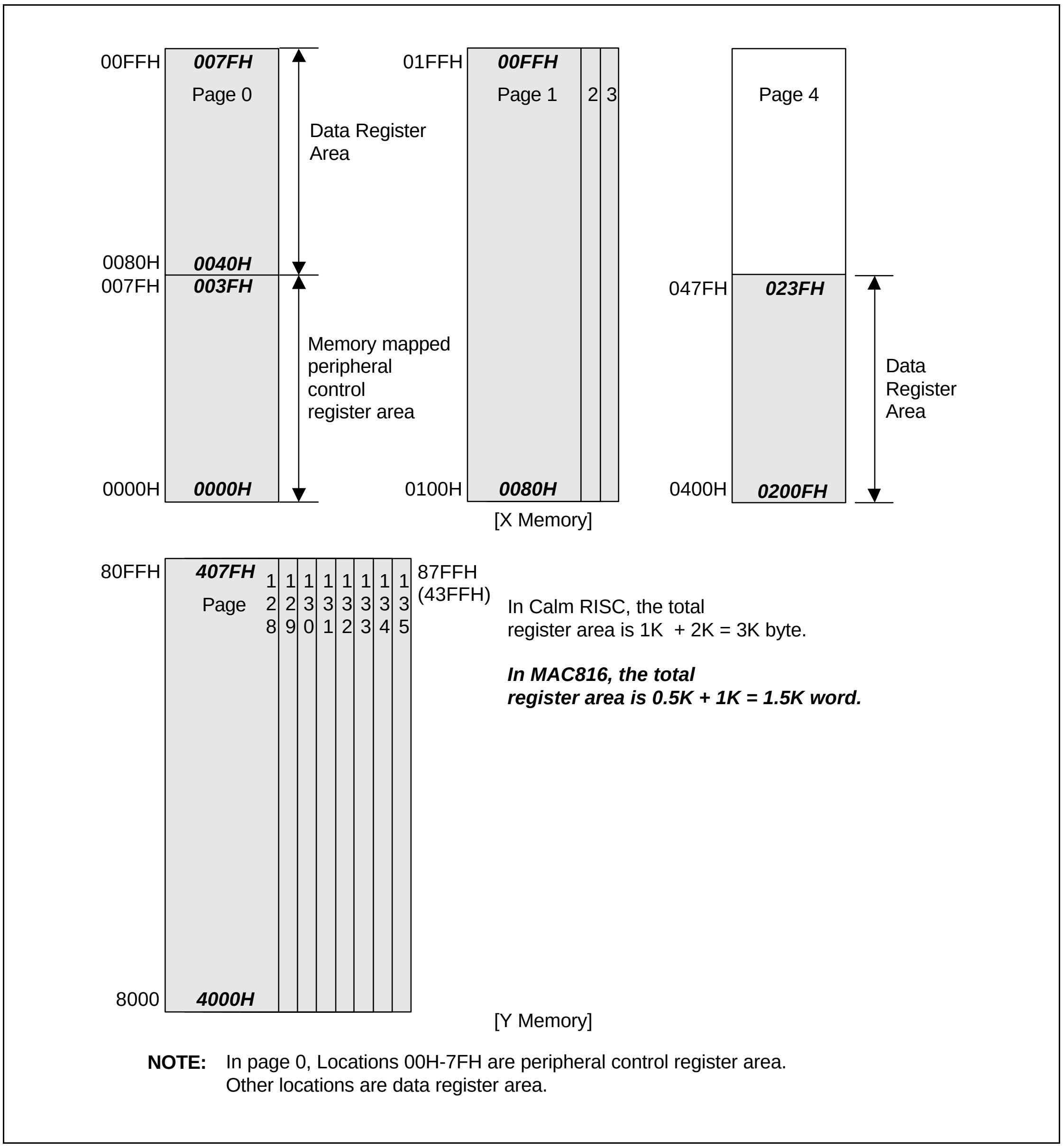


Figure 2-5. Data Memory Map

The data memory page is indexed by an SPR, IDH. In data memory index addressing mode, 16-bit data memory address is composed of two 8-bit SPR's, IDH[7:0] and IDL0[7:0] (or IDH[7:0] and IDL1[7:0]). IDH[7:0] points to a page index and IDL0[7:0] (or IDL1[7:0]) represents the page offset. In data memory direct addressing mode, an 8-bit direct address, adr[7:0], specifies the page offset pointed to by IDH[7:0] (See the details for direct addressing mode in the instruction sections). Unlike the program memory organization, data memory address does *not* wrap around. In other words, data memory index addressing with modification performs an addition or a subtraction operation on the whole 16-bit addresses of IDH[7:0] and IDL0[7:0] (or IDL1[7:0]) and updates IDH[7:0] and IDL0[7:0] (or IDL1[7:0]) accordingly. Suppose IDH[7:0] is 0FH and IDL0[7:0] is FCH and the modification on the index registers, IDH[7:0] and IDL0[7:0], is increment by 5H. Then after the modification (i.e., 0FFCH + 5 = 1001H), IDH[7:0]and IDL0[7:0] become 10H and 01H, respectively.

The S3CB018/FB018 has a total of 3072 bytes of data register address from 0000H to 047FH and from 8000H to 87FFH . Here the area from 0000H to 007FH is for peripheral control, Aside from this area the other area is for data memory. *Mac816 views the peripheral control register area from 0000H to 003FH; X memory is from 0040H to 23FH, and Y memory is from 4000H to 43FFH.*



NOTE: In page 0, Locations 00H-7FH are peripheral control register area. Other locations are data register area.

Figure 2-6. S3CB018/FB018 Data Memory Map

3 REGISTERS

OVERVIEW

The registers of CalmRISC are grouped into 2 parts: general purpose registers and special purpose registers.

Table 3-1. General and Special Purpose Registers

Registers		Mnemonics	Description	Reset Value
General Purpose Registers (GPR)		R0	General Register 0	Unknown
		R1	General Register 1	Unknown
		R2	General Register 2	Unknown
		R3	General Register 3	Unknown
Special Purpose Registers (SPR)	Group 0 (SPR0)	IDL0	Lower Byte of Index Register 0	Unknown
		IDL1	Lower Byte of Index Register 1	Unknown
		IDH	Higher Byte of Index Register	Unknown
		SR0	Status Register 0	00H
	Group 1 (SPR1)	ILX	Instruction Pointer Link Register for Extended Byte	Unknown
		ILH	Instruction Pointer Link Register for Higher Byte	Unknown
		ILL	Instruction Pointer Link Register for Lower Byte	Unknown
		SR1	Status Register 1	Unknown

GPR's can be used in most instructions such as ALU instructions, stack instructions, load instructions, *etc* (See the instruction set sections). From the programming standpoint, they have almost no restriction whatsoever. CalmRISC has 4 banks of GPR's and each bank has 4 registers, R0, R1, R2, and R3. Hence, 16 GPR's in total are available. The GPR bank switching can be done by setting an appropriate value in SR0[4:3] (See SR0 for details). The ALU operations among GPR's from different banks are *not* allowed.

SPR's are designed for their own dedicated purposes. They have some restrictions in terms of instructions that can access them. For example, direct ALU operations cannot be performed on SPR's. However, data transfers between a GPR and an SPR are allowed and stack operations with SPR's are also possible (See the instruction sections for details).

INDEX REGISTERS: IDH, IDL0 AND IDL1

IDH in concatenation with IDL0 (or IDL1) forms a 16-bit data memory address. Note that CalmRISC's data memory address space is 64 K byte (addressable by 16-bit addresses). Basically, IDH points to a page index and IDL0 (or IDL1) corresponds to an offset of the page. Like GPR's, the index registers are 2-way banked. There are 2 banks in total, each of which has its own index registers, IDH, IDL0 and IDL1. The banks of index registers can be switched by setting an appropriate value in SR0[2] (See SR0 for details). Normally, programmers can reserve an index register pair, IDH and IDL0 (or IDL1), for software stack operations.

LINK REGISTERS: ILX, ILH AND ILL

The link registers are specially designed for link-and-branch instructions (See LNK and LRET instructions in the instruction sections for details). When an LNK instruction is executed, the current PC[19:0] is saved into ILX, ILH and ILL registers, i.e., PC[19:16] into ILX[3:0], PC[15:8] into ILH [7:0], and PC[7:0] into ILL[7:0], respectively. When an LRET instruction is executed, the return PC value is recovered from ILX, ILH, and ILL, i.e., ILX[3:0] into PC[19:16], ILH[7:0] into PC[15:8] and ILL[7:0] into PC[7:0], respectively. These registers are used to access program memory by LDC/LDC+ instructions. When an LDC or LDC+ instruction is executed, the (code) data residing at the program address specified by ILX:ILH:ILL will be read into TBH:TBL. LDC+ also increments ILL after accessing the program memory.

There is a special core input pin signal, *nP64KW*, which is reserved for indicating that the program memory address space is only 64 K word. By grounding the signal pin to zero, the upper 4 bits of PC, PC[19:16], are deactivated and therefore the upper 4 bits , PA[19:16], of the program memory address signals from CalmRISC core are also deactivated. By doing so, the power consumption due to manipulating the upper 4 bits of PC can be totally eliminated (See the core pin description section for details). From the programmer's standpoint, when *nP64KW* is tied to the ground level, PC[19:16] is *not* saved into ILX for LNK instructions and ILX is *not* read back into PC[19:16] for LRET instructions. Therefore, ILX is totally unused for LNK and LRET instructions when *nP64KW* = 0.

STATUS REGISTER 0: SR0

SR0 is mainly reserved for system control functions and each bit of SR0 has its own dedicated function.

Table 3-2. Status Register 0 Configuration

Flag Name	Bit	Description
eid	0	Data memory page selection in direct addressing
ie	1	Global interrupt enable
idb	2	Index register banking selection
grb[1:0]	4,3	GPR bank selection
exe	5	Stack overflow/underflow exception enable
ie0	6	Interrupt 0 enable
ie1	7	Interrupt 1 enable

SR0[0] (or eid) selects which page index is **to be** used in direct addressing. If eid = 0, then page 0 (page index = 0) is used. Otherwise (eid = 1), IDH of the current index register bank is used for page index. SR0[1] (or ie) is the global interrupt enable flag. As explained in the interrupt/exception section, CalmRISC has 3 interrupt sources (non-maskable interrupt, interrupt 0, and interrupt 1) and 1 stack exception. Both interrupt 0 and interrupt 1 are masked by setting SR0[1] to 0 (i.e., ie = 0). When an interrupt is serviced, the global interrupt enable flag ie is automatically cleared. The execution of an IRET instruction (return from an interrupt service routine) automatically sets ie = 1. SR0[2] (or idb) and SR0[4:3] (or grb[1:0]) ~~selects~~ **select** an appropriate bank for index registers and GPR’s, respectively**(,)** as shown below:

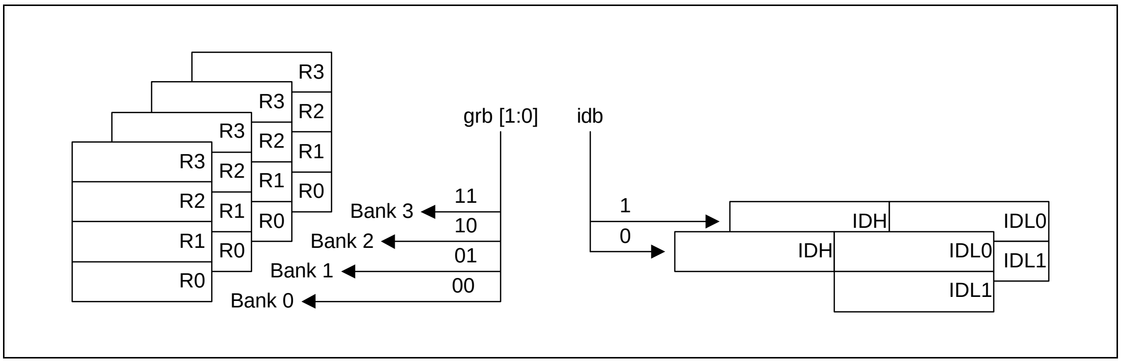


Figure 3-1. Bank Selection by Setting of GRB Bits and IDB Bit

SR0[5] (or exe) enables the stack exception, that is, the stack overflow/underflow exception. If exe = 0, the stack exception is disabled. The stack exception can be used for program debugging in the software development stage. SR0[6] (or ie0) and SR0[7] (or ie1) are enabled, by setting them to 1. Even though ie0 ~~or~~ **are** ie1 are enabled, the interrupts are ignored (not serviced) if the global interrupt enable flag ie is set to 0.

STATUS REGISTER 1: SR1

SR1 is the register for status flags such as the ALU execution flag and stack full flag.

Table 3-3. Status Register 1: SR1

Flag Name	Bit	Description
C	0	Carry flag
V	1	Overflow flag
Z	2	Zero flag
N	3	Negative flag
SF	4	Stack Full flag
–	5, 6, 7	Reserved

SR1[0] (or C) is the carry flag of ALU executions. SR1[1] (or V) is the overflow flag of ALU executions. It is set to 1 if and only if the carry-in into the 8-th bit position of addition/subtraction differs from the carry-out from the 8-th bit position. SR1[2] (or Z) is the zero flag, which is set to 1 if and only if the ALU result is zero. SR1[3] (or N) is the negative flag. Basically, the most significant bit (MSB) of ALU results becomes the N flag. Note a load instruction into a GPR is considered an ALU instruction. However, if an ALU instruction touches the overflow flag (V) like ADD, SUB, CP, *etc*, N flag is updated as exclusive-OR of V and the MSB of the ALU result. This implies that even if an ALU operation results in overflow, N flag is still valid. SR1[4] (or SF) is the stack overflow flag. It is set when the hardware stack is overflowed or underflowed. Programmers can check if the hardware stack has any abnormalities by checking the stack exception or testing if SF is set (See the hardware stack section for more details).

NOTE: When an interrupt occurs SR0 and SR1 are not saved by hardware, so the SR1 register values must be saved by software.

4 CONTROL REGISTERS

OVERVIEW

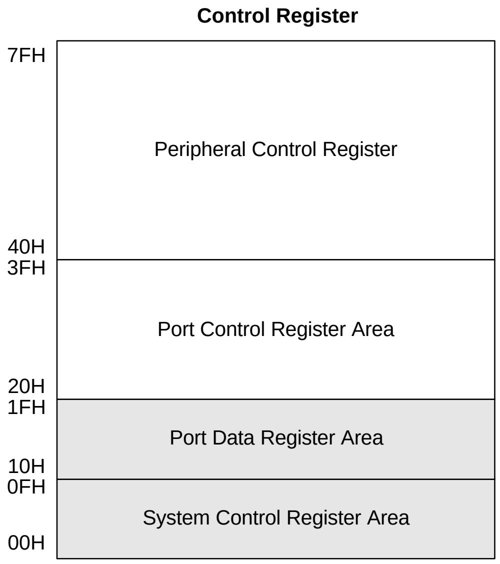
To support the control of peripheral hardware, the addresses for peripheral control registers are memory-mapped to page 0 of the RAM. Memory mapping lets you use a mnemonic as the operand of an instruction in place of the specific memory location.
In this section, detailed descriptions of the S3CB018/FB018 control registers are presented in an easy-to-read format.
You can use this section as a quick-reference source when you write application programs.

This memory area can be accessed with the whole method of data memory access.

— If SR0 bit 0 is “0” **,** then the accessed register area is always page 0.

— If SR0 bit 0 is “1” **,** then the accessed register page is controlled by the proper IDH register’ s value.

So if you want to access the memory map area, clear the SR0.0 and use the direct addressing mode.
This method is used for most cases.
This control register is divided into four areas. Here, the system control register area is the same in every device.


Control Register
7FH
Peripheral Control Register
40H
3FH
Port Control Register Area
20H
1FH
Port Data Register Area
10H
0FH
System Control Register Area
00H

Locations from 60H to 7FH can be used for an external device in S3EB010.
You can control the pheperals that you want to use externally by setting this area.

For example, if you use a peripheral and make it's control register addresss 62H and load data to this address, the data would be output through XDA[13:0] + EVENIO. The address would be come out, and the data through XD[7:0].
At this time the WENIO signal is output for writing and OENIO signal is for reading.

But the above example is possiable at S3EB010. S3CB018/FB018 has the same memory map but does not have interface pins. More peripherals in the main MCU can be requested through Samsung.

□ Standard exhortative area

▨ Standard area

Figure 4-1. Memory Map Areas

Table 4-1. Registers

Register Name	Mnemonic	Decimal	Hex	Reset	R/W
Locations 14H-1FH are not mapped					
Port 3 data register	P3	19	13H	00H	R/W
Port 2 data register	P2	18	12H	00H	R/W
Port 1 data register	P1	17	11H	00H	R/W
Port 0 data register	P0	16	10H	00H	R/W
Watchdog timer control register	WDTCON	15	0FH	00H	R/W
Watchdog timer enable register	WDTEN	14	0EH	A5H	R/W
Basic timer counter	BTCNT	13	0DH	00H	R
Basic timer control register	BTCON	12	0CH	**70H**	R/W
Interrupt ID register 1	IIR1	11	0BH	–	R/W
Interrupt priority register 1	IPR1	10	0AH	–	R/W
Interrupt mask register 1	IMR1	9	09H	00H	R/W
Interrupt request register 1	IRQ1	8	08H	00H	R
Interrupt ID register 0	IIR0	7	07H	–	R/W
Interrupt priority register 0	IPR0	6	06H	–	R/W
Interrupt mask register 0	IMR0	5	05H	00H	R/W
Interrupt request register 0	IRQ0	4	04H	00H	R
Oscillator control register	OSCCON	3	03H	00H	R/W
Power control register	PCON	2	02H	**04H**	R/W
Locations 00H-01H are not mapped					

NOTES:

1. ‘-’ means undefined.
2. If you want to clear the bit of IRQx, then write the number which you want to clear to IIRx. For example, when clear IRQ0.4 then LD Rx, #04H and LD IIRQ, Rx.

Table 4-1. Registers (continued)

Register Name	Mnemonic	Decimal	Hex	Reset	R/W
Locations 51H-7FH are not mapped (reserved for extra peripherals externally)					
Watch timer control register	WTCON	80	50H	00H	R/W
ADDA data register low byte	ADDATAL	79	4FH	00H	R/W
ADDA data register high byte	ADDATAH	78	4EH	00H	R/W
Locations 4DH is not mapped					
ADDASIO control register	ADDACON	76	4CH	00H	R/W
Locations 4BH is not mapped					
Serial I/O data register	SIODATA	74	4AH	00H	R/W
Serial I/O pre-scale register	SIOPS	73	49H	00H	R/W
Serial I/O control register	SIOCON	72	48H	00H	R/W
Location 47H is not mapped					
Timer B counter	TBCNT	70	46H	–	R
Timer B data register	TBDATA	69	45H	00H	R/W
Timer B control register	TBCON	68	44H	00H	R/W
Location 43H is not mapped					
Timer A counter	TACNT	66	42H	–	R
Timer A data register	TADATA	65	41H	00H	R/W
Timer A control register	TACON	64	40H	00H	R/W
Locations 2DH-3FH are not mapped					
Port 3 control register	P3CON	44	2CH	00H	R/W
Location 2BH is not mapped					
Port 2 interrupt control register	P2INT	42	2AH	00H	R/W
Port 2 control register low	P2CONL	41	29H	00H	R/W
Port 2 control register high	P2CONH	40	28H	00H	R/W
Locations 25H-27H are not mapped					
Port 1 control register	P1CON	36	24H	00H	R/W
Locations 22H-23H are not mapped					
Port 0 control register low	P0CONL	33	21H	00H	R/W
Port 0 control register high	P0CONH	32	20H	00H	R/W

5 HARDWARE STACK

OVERVIEW

The hardware stack in CalmRISC has two usages:

- To save and restore the return PC[19:0] on LCALL, CALLS, RET, and IRET instructions.
- To use as temporary storage space for registers on PUSH and POP instructions.

When PC[19:0] is saved into or restored from the hardware stack, the access should be 20 bits wide. On the other hand, when a register is pushed into or popped from the hardware stack, the access should be 8 bits wide. Hence, to maximize the efficiency of the stack usage, the hardware stack is divided into 3 parts: the extended stack bank (XSTACK, 4-bits wide), the odd bank (8-bits wide) and the even bank (8-bits wide).

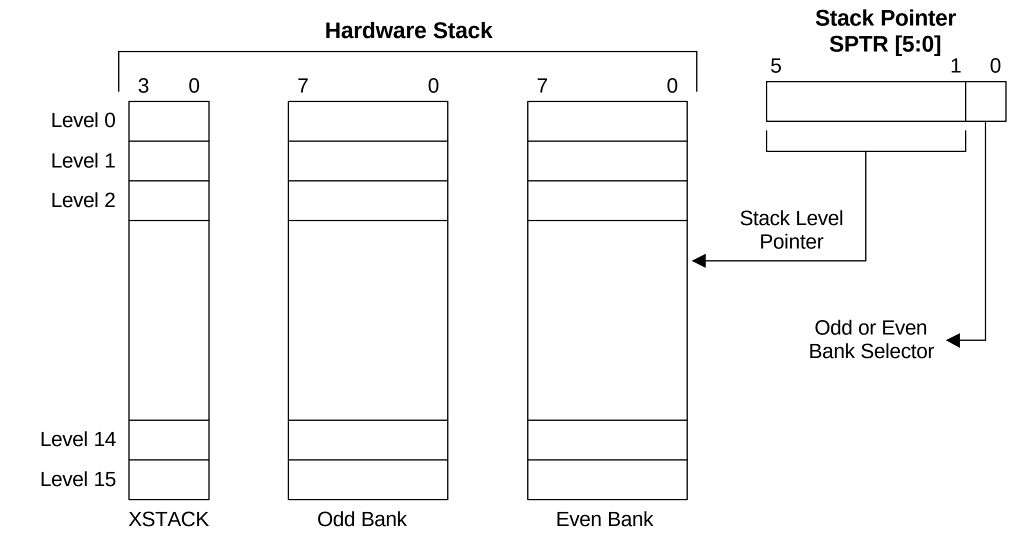


Figure 5-1. Hardware Stack

The stack pointer, called sptr[5:0], points to the top of the stack (TOS). The upper 5 bits of the stack pointer, sptr[5:1], points to the stack level, where either PC[19:0] or a register is saved. For example, if sptr[5:1] is 5H or TOS is 5, then level 5 of XSTACK is empty and either level 5 of the odd bank or level 5 of the even bank is empty. In fact, sptr[0], the stack bank selection bit, indicates which bank(s) is empty. If sptr[0] = 0, both level 5 of the even and the odd banks are empty. On the other hand, if sptr[0] = 1, level 5 of the odd bank is empty, but level 5 of the even bank is occupied. This situation is well illustrated in the figure below.

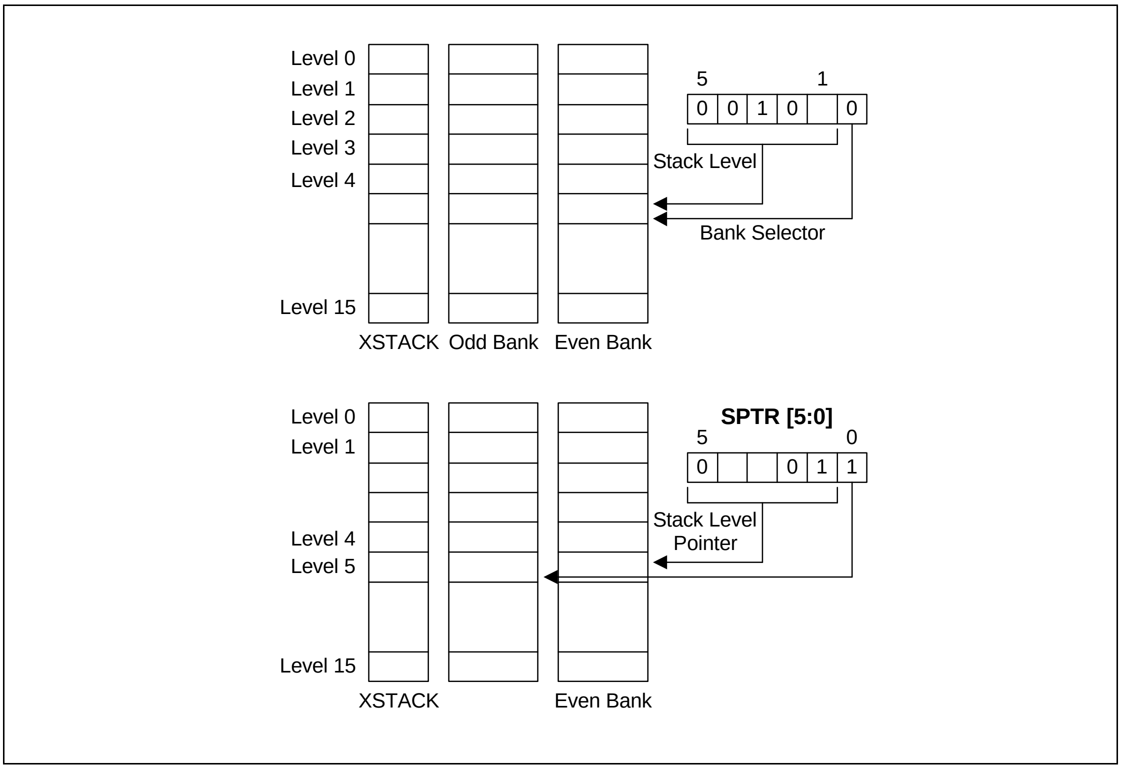
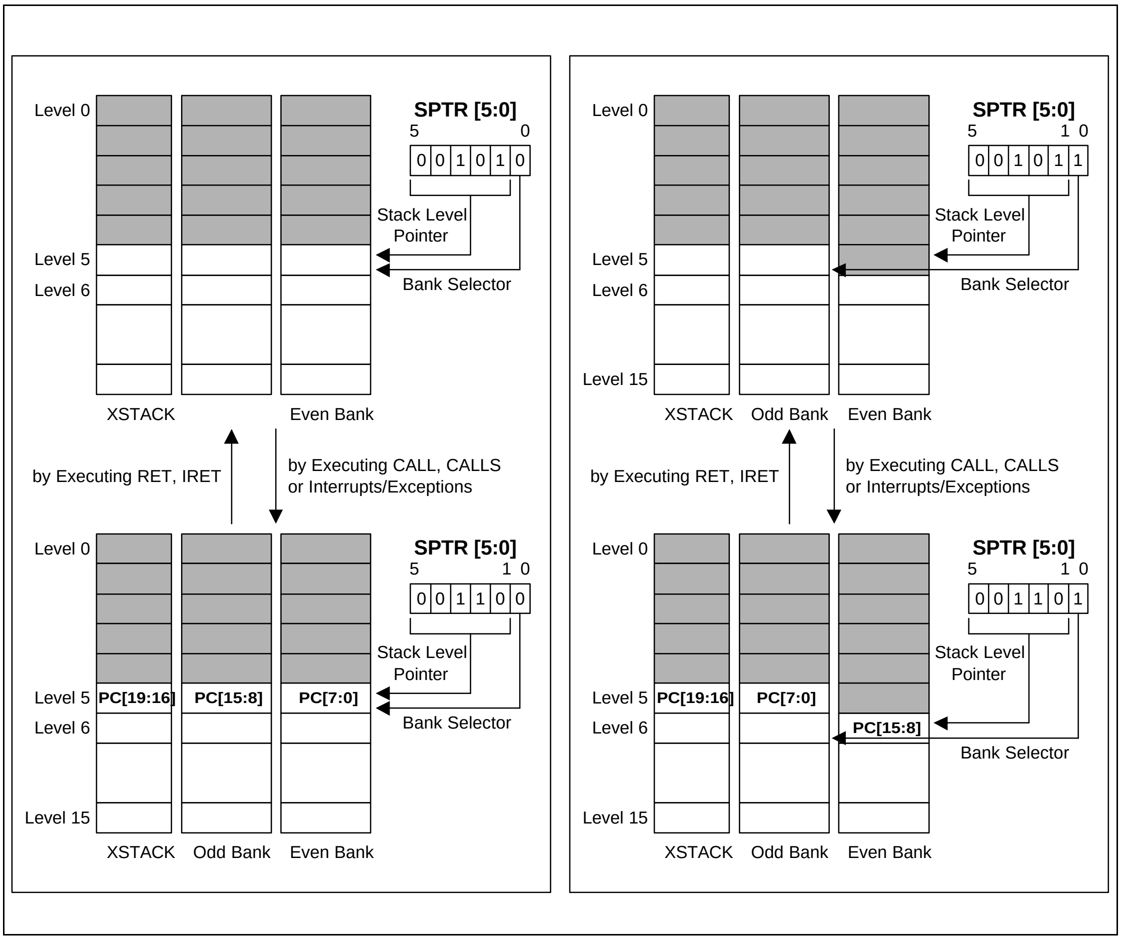


Figure 5-2. Even and Odd Bank Selection Example

As can be seen in the above example, sptr[5:1] is used as the hardware stack pointer when PC[19:0] is pushed or popped and sptr[5:0] as the hardware stack pointer when a register is pushed or popped. Note that XSTACK is used only for storing and retrieving PC[19:16]. Let us consider the cases where PC[19:0] is pushed into the hardware stack (by executing LCALL/CALLS instructions or by interrupts/exceptions being served) or is retrieved from the hardware stack (by executing RET/IRET instructions). Regardless of the stack bank selection bit (sptr[0]), TOS of the even bank and the odd bank store or return PC[7:0] or PC[15:8], respectively. This is illustrated in the following figures.



As can be seen in the figures, when stack operations with PC[19:0] are performed, the stack level pointer sptr[5:1] (*not* sptr[5:0]) is either incremented by 1 (when PC[19:0] is pushed into the stack) or decremented by 1 (when PC[19:0] is popped from the stack). The stack bank selection bit (sptr[0]) is unchanged. If a CalmRISC core input signal *nP64KW* is 0, signifying that only PC[15:0] is meaningful, then any access to XSTACK is totally deactivated from the stack operations with PC. Therefore, XSTACK has no meaning when the input pin signal, *nP64KW*, is tied to 0. In that case, XSTACK does not have to even exist. As a matter of fact, XSTACK is not included in CalmRISC core itself**,** and it is interfaced through some specially reserved core pin signals (*nPUSH, nSTACK*, *XHSI[3:0]*, *XSHO[3:0]*), if the program address space is more than 64 K words (See the core pin signal section for details).

A similar argument can be made with regards to stack operations with registers. A stack pointer, called sptr[5:0], points to the top of the stack (TOS). The only difference is that the data written into or read from the stack are a byte. Hence, the even bank and the odd bank are accessed alternately as shown below.

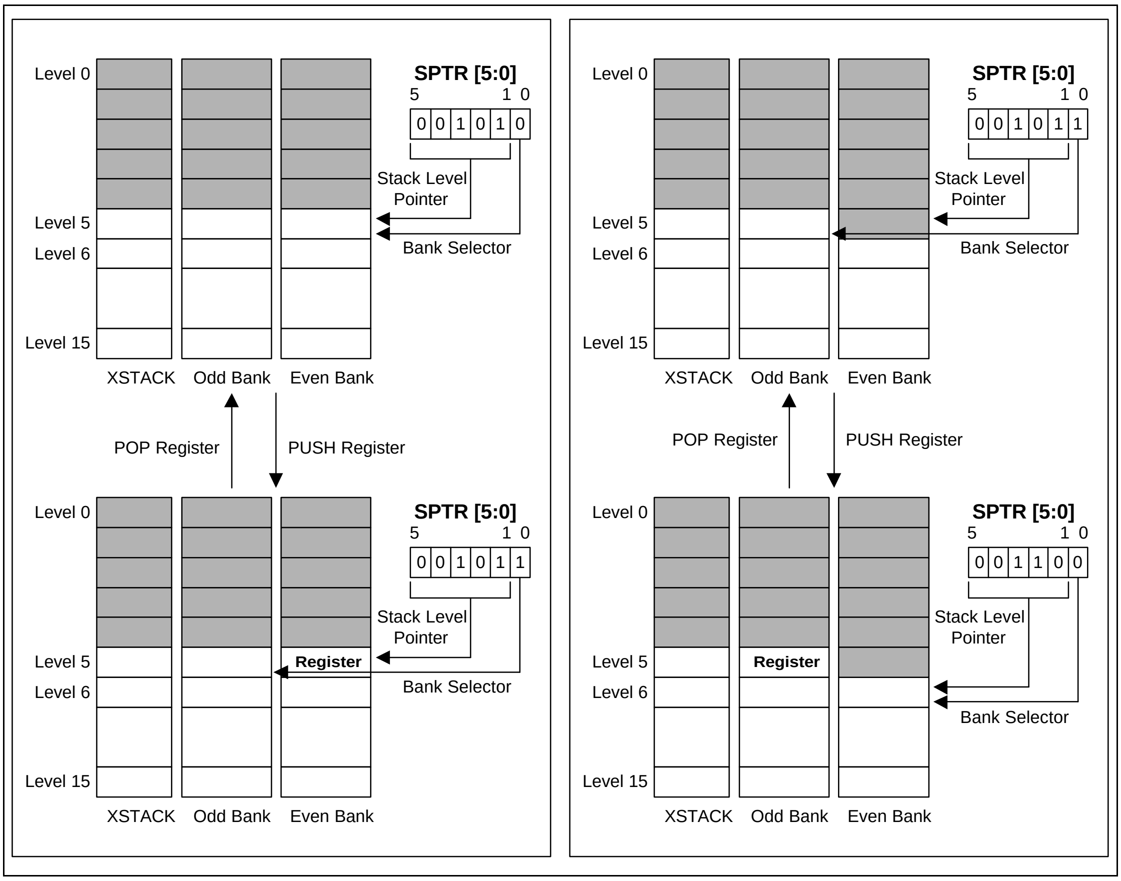


Figure 5-4. Stack Operation with Registers

When the bank selection bit (sptr[0]) is 0, the register is pushed into the even bank and the bank selection bit is set to 1. In this case, the stack level pointer is unchanged. When the bank selection bit (sptr[0]) is 1, then the register is pushed into the odd bank, the bank selection bit is set to 0, and the stack level pointer is incremented by 1. Unlike the push operations of PC[19:0], no data are written into XSTACK in the register push operations. This is illustrated in the example figures. When a register is pushed into the stack, sptr[5:0] is incremented by 1 (*not* the stack level pointer sptr[5:1]). The register pop operations are the reverse processes of the register push operations. When a register is popped out of the stack, sptr[5:0] is decremented by 1 (*not* the stack level pointer sptr[5:1]).

Hardware stack overflows/underflows when the MSB of the stack level pointer, sptr[5], is 1. This is obvious from the fact that the hardware stack has only 16 levels and the following relationship holds for the stack level pointer in a normal case.

Suppose the stack level pointer sptr[5:1] = 15 (or 01111B in binary format) and the bank selection bit sptr[0] = 1. Here if either PC[19:0] or a register is pushed, the stack level pointer is incremented by 1. Therefore, sptr[5:1] = 16 (or 10000B in binary format) and sptr[5] = 1, which imply that the stack is overflowed. The situation is depicted in the following.

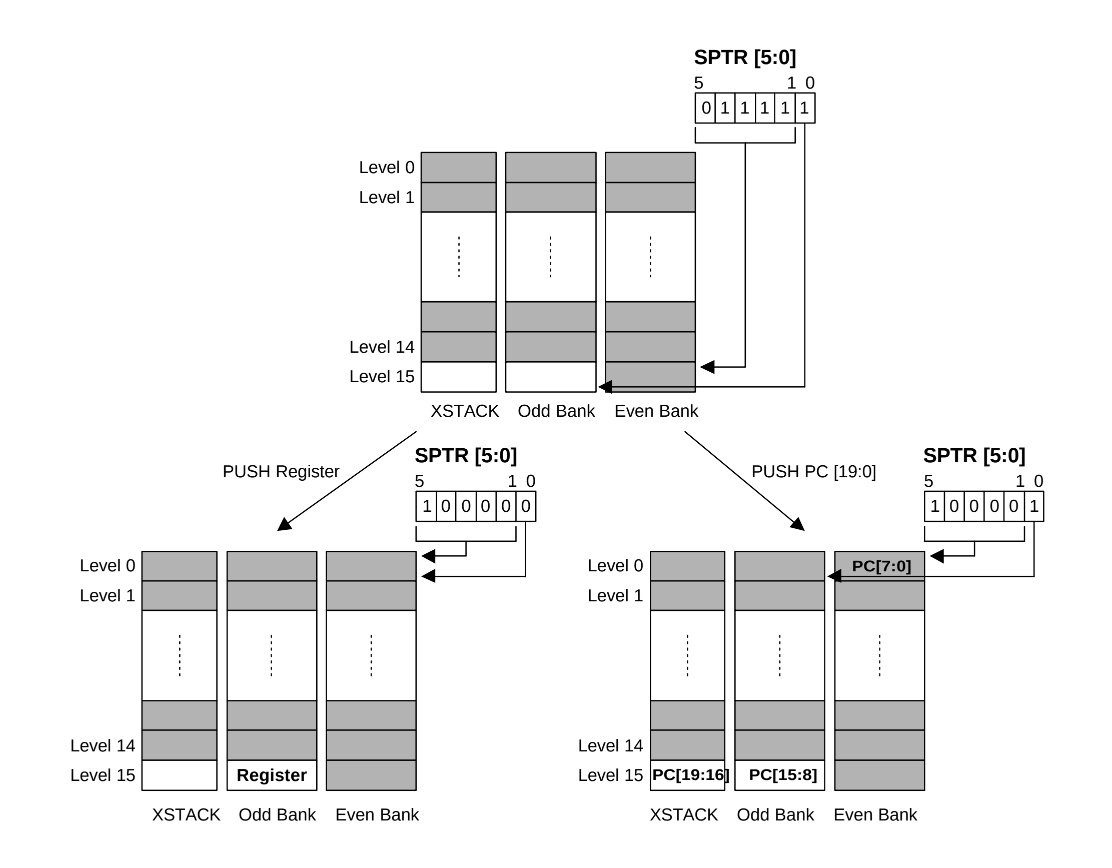


Figure 5-5. Stack Overflow

The first overflow happens due to a register push operation. As explained earlier, a register push operation increments sptr[5:0] (not sptr[5:1]) , which results in sptr[5] = 1, sptr[4:1] = 0 and sptr[0] = 0. As indicated by sptr[5] = 1, an overflow happens. Note that this overflow does not overwrite any data in the stack. On the other hand, when PC[19:0] is pushed, sptr[5:1] is incremented by 1 instead of sptr[5:0], and as expected, an overflow results. Unlike the first overflow, PC[7:0] is pushed into level 0 of the even bank and the data that have been there before the push operation are *overwritten*. A similar argument can be made about stack underflows. Note that any stack operation, which causes the stack to overflow or underflow, does not necessarily mean that any data in the stack are lost, as is observed in the first example.

In SR1, there is a status flag, SF (Stack Full Flag), which is exactly the same as sptr[5]. In other words, the value of sptr[5] can be checked by reading SF (or SR1[4]). SF is set to 1 if there is an stack overflow/underflow. If POP operation is executed after overflow, SF is cleared to 0. So programmers cannot tell whether there was a stack overflow/underflow just by reading SF. For example, if a program pushes a register 64 times in a row, sptr[5:0] is exactly the same as sptr[5:0] before the push sequence. Therefore, special attention should be paid.

Another mechanism to detect a stack overflow/underflow is through a stack exception. A stack exception happens only when the execution of any stack access instruction results in SF = 1 (or sptr[5] = 1). Suppose a register push operation makes SF = 1 (the SF value before the push operation does not matter). Then the stack exception due to the push operation is immediately generated and served, if the stack exception enable flag (exe of SR0) is 1. If the stack exception enable flag is 0, then the generated interrupt is not served but pending. Sometime later when the stack exception enable flag is set to 1, the pending exception request is served even if SF = 0. More details are available in the stack exception section.

6 EXCEPTIONS

OVERVIEW

Exceptions in CalmRISC are listed in the table below. Exception handling routines, residing at the given addresses in the table, are invoked when the corresponding exception occurs. The starting address of each exception routine is specified by concatenating 0H (leading 4 bits of 0) and the 16-bit data in the exception vector listed in the table. For example, the interrupt service routine for IRQ[0] starts from 0H:PM[00002H]. Note that ":" means concatenation and PM[*] stands for the 16-bit content at the address * of the program memory. Aside from the exception due to reset release, the current PC is pushed in the stack on an exception. When an exception is executed due to IRQ[1:0]/IEXP, the global interrupt enable flag, bit (SR0[1]), is set to 0, and ie is set to 1 when IRET or an instruction that explicitly sets ie is executed.

Table 6-1. Exceptions

Name	Address	Priority	Description
Reset	00000H	1 st	Exception due to rest release.
–	00001H	–	Reserved.
IRQ[0]	00002H	3 rd	Exception due to *nIRQ[0]* signal. Maskable by setting ie/ie0.
IRQ[1]	00003H	4 th	Exception due to *nIRQ[1]* signal. Maskable by setting ie/ie1.
IEXP	00004H	2 nd	Exception due to stack full. Maskable by setting exe.
–	00005H	–	Reserved.
–	00006H	–	Reserved.
–	00007H	–	Reserved.

NOTE: Break mode due to BKREQ has a higher priority than all the exceptions above. That is, when BKREQ is active, even the exception due to reset release is not executed.

HARDWARE RESET

When Hardware Reset is active (the reset input signal pin *nRES* = 0), the control pins in the CalmRISC core are initialized to be disabled, and SR0 and sptr (the hardware stack pointer) are initialized to be 0. Additionally, the interrupt sensing block is cleared. When Hardware Reset is released (nRES = 1), the reset exception is executed by loading the JP instruction in IR (Instruction Register) and 0h:0000h in PC. Therefore, when Hardware Reset is released, the "JP {0h:PM[00000h]}" instruction is executed. When the reset exception is executed, a core output signal *nEXPACK* is generated to acknowledge the exception.

IRQ[0] EXCEPTION (LEVEL-SENSITIVE)

When a core input signal *nIRQ[0]* is low, SR0[6] (ie0) is high, and SR0[1] (ie) is high and IRQ[0] exception is generated; this will load the CALL instruction in IR (Instruction Register) and 0h:0002h in PC. Therefore, on an IRQ[0] exception, the “CALL {0h:PM[00002h]}” instruction is executed. When the IRQ[0] exception is executed, SR0[1] (ie) is set to 0 and a core output signal *nEXPACK* is generated to acknowledge the exception.

IRQ[1] EXCEPTION (LEVEL-SENSITIVE)

When a core input signal *nIRQ[1]* is low, SR0[7] (ie1) is high, and SR0[1] (ie) is high and IRQ[1] exception is generated; this will load the CALL instruction in IR (Instruction Register) and 0h:0003h in PC. Therefore, on an IRQ[1] exception, the “CALL {0h:PM[00003h]}” instruction is executed. When the IRQ[1] exception is executed, SR0[1] (ie) is set to 0 and a core output signal *nEXPACK* is generated to acknowledge the exception.

HARDWARE STACK FULL EXCEPTION

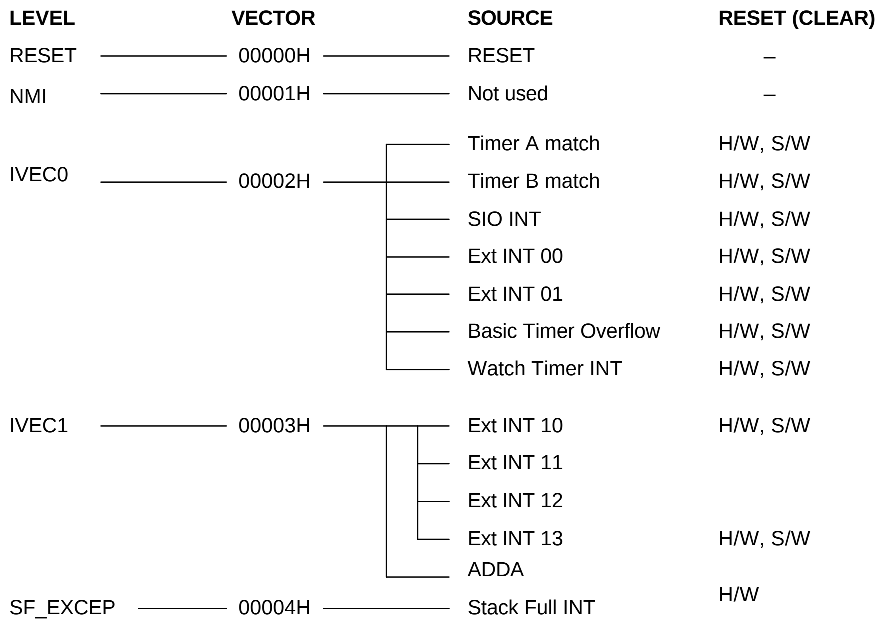
A Stack Full exception occurs when a stack operation is performed and as a result of the stack operation sptr[5] (SF) is set to 1. If the stack exception enable bit, exe (SR0[5]), is 1, the Stack Full exception is served. One exception to this rule is when nNMI causes a stack operation that sets sptr[5] (SF), since it has higher priority.

Handling a Stack Full exception may cause another Stack Full exception. In this case, the new exception is ignored. On a Stack Full exception, the CALL instruction is loaded in IR (Instruction Register) and 0h:0004h in PC. Therefore, when the Stack Full exception is activated, the “CALL {0h:PM[00004h]}” instruction is executed. When the exception is executed, SR0[1] (ie) is set to 0, and a core output signal *nEXPACK* is generated to acknowledge the exception.

BREAK EXCEPTION

Break exception is reserved only for an in-circuit debugger. When a core input signal, *BKREQ*, is high, the CalmRISC core is halted or entered in the break mode, until *BKREQ* is deactivated. Another way to drive the CalmRISC core into the break mode is by executing a break instruction, BREAK. When BREAK is fetched, it is decoded in the fetch cycle (IF stage) and the CalmRISC core output signal *nBKACK* is generated in the second cycle (ID/MEM stage). An in-circuit debugger makes *BKREQ* active by monitoring whether *nBKACK* is active or not. BREAK instruction is exactly the same as the NOP (no operation) instruction except that it does not increase the program counter. It activates nBKACK in the second cycle (or ID/MEM stage of the pipeline). There, once BREAK is encountered in the program execution, it falls into a deadlock. BREAK instruction is reserved for in-circuit debuggers only, so it should not be used in user programs.

EXCEPTIONS (or INTERRUPTS)

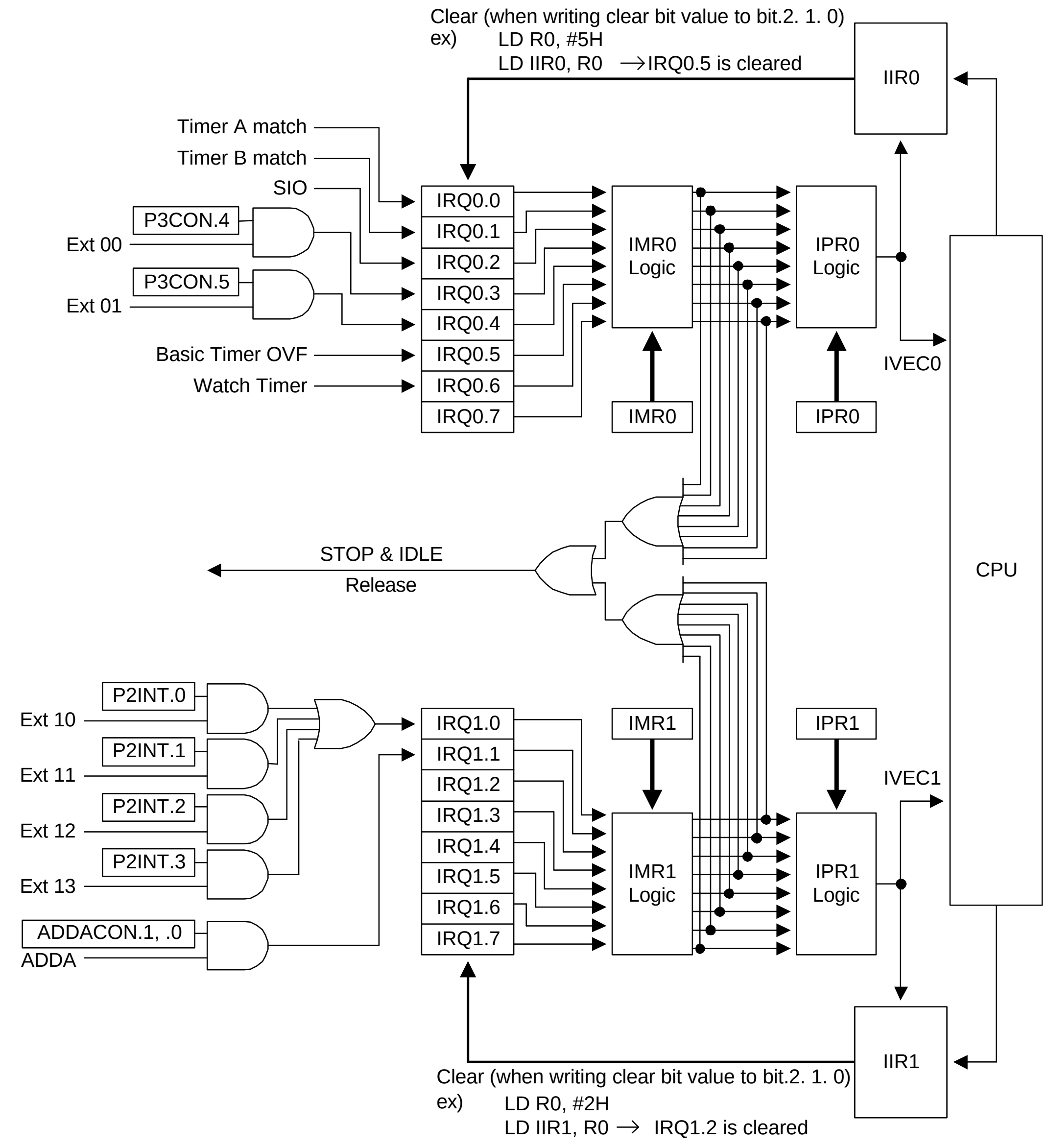


LEVEL	VECTOR	SOURCE	RESET (CLEAR)
RESET	00000H	RESET	–
NMI	00001H	Not used	–
IVEC0	00002H	Timer A match	H/W, S/W
		Timer B match	H/W, S/W
		SIO INT	H/W, S/W
		Ext INT 00	H/W, S/W
		Ext INT 01	H/W, S/W
		Basic Timer Overflow	H/W, S/W
		Watch Timer INT	H/W, S/W
IVEC1	00003H	Ext INT 10	H/W, S/W
		Ext INT 11	
		Ext INT 12	
		Ext INT 13	H/W, S/W
		ADDA	
SF_EXCEP	00004H	Stack Full INT	H/W

NOTES:

1. In cases IVEC0 and IVEC1, one interrupt vector has several interrupt sources. The priority of the sources is controlled by setting the IPR register.
2. External interrupts are triggered by a rising or falling edge, depending on the corresponding control register setting. Ext INT 10 - Ext INT 13 have no interrupt pending bit but have an enable bit.
3. After system reset, the IPR register is in unknown status, so the user must set the IPR register with proper value.
4. The interrupt priority can be changed by setting the IPR register.
5. The pending bit is cleared by Hard Wearely when CPU reads the IIR registser value.

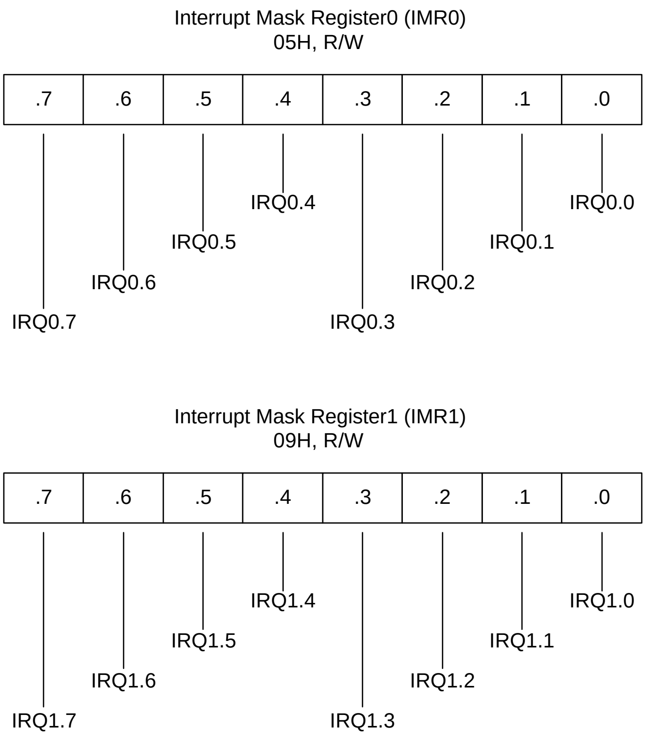
Figure 6-1. Interrupt Structure



NOTE: The IRQ register value is cleared by H/W when the IIR register is read by the programmer in an interrupt service routine. However, if you want to clear by S/W, then write the proper value to the IIR register like as in the example above. To clear all the bits of IRQx register at one time write "#08h" to the IIRx register.

Figure 6-2. Interrupt Block Diagram

INTERRUPT MASK REGISTERS

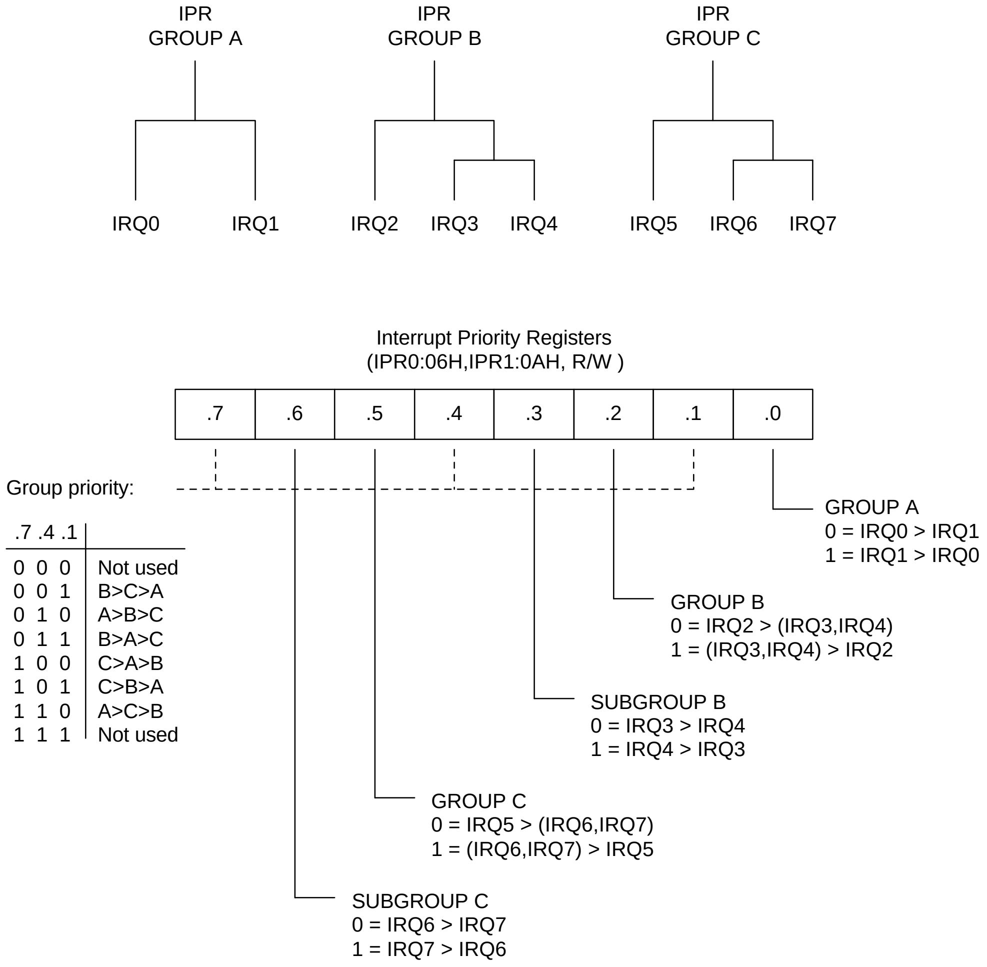


Interrupt request enable bits:
0 = Disable interrupt request
1 = Enable interrupt request

NOTE: If you want to change the value of the IMR register, then you should first disable global INT using the DI instruction and change the value of the IMR register.

Figure 6-3. Interrupt Mask Register

INTERRUPT PRIORITY REGISTER



.7 .4 .1	
0 0 0	Not used
0 0 1	B>C>A
0 1 0	A>B>C
0 1 1	B>A>C
1 0 0	C>A>B
1 0 1	C>B>A
1 1 0	A>C>B
1 1 1	Not used

NOTE: If you want to change the value of the IPR register, then you should first disable the global INT using the DI instruction and change the value of the IPR register. After reset, the IPR register is unknown status, so user must set the IPR register with proper value.

Figure 6-4. Interrupt Priority Register

☞ PROGRAMMING TIP —Interrupt Programming Tip 1

Jumped from vector 2

```
           PUSH     SR1

           PUSH     R0
           LD       R0, IIR0
           CP       R0, #03h
           JR       ULE, LTE03
           CP       R0, #05h
           JR       ULE, LTE05
           CP       R0, #06h
           JP       EQ, IRQ6_srv
           JP       T, IRQ7_srv
LTE05      CP       R0, #04
           JP       EQ, IRQ4_srv
           JP       T, IRQ5_srv
LTE03      CP       R0, #01
           JR       ULE, LTE01
           CP       R0, #02
           JP       EQ, IRQ2_srv
           JP       T, IRQ3_srv
LTE01      CP       R0, #00h
           JP       EQ, IRQ0_srv
           JP       T, IRQ1_srv

IRQ0_srv   →  service for IRQ0
           •

           POP      R0
           POP      SR1
           IRET

IRQ1_srv   →  service for IRQ1
           •
           •
           POP      R0
           POP      SR1
           IRET
           •
           •

IRQ7_srv   →  service for IRQ7
           •
           •
           POP      R0
           POP      SR1
           IRET
```

NOTE: If the SR0 register is changed in the interrupt service routine, then the SR0 register must be pushed and popped in the interrupt service routine.

☞ PROGRAMMING TIP —Interrupt Programming Tip 2

Jumped from vector 2

```
            PUSH        SR1
            PUSH        R0
            PUSH        R1
            LD          R0, IIRx

            SL          R0
            LD          R1, < TBL_INTx
            ADD         R0, > TBL_INTx
            PUSH        R1
            PUSH        R0
            RET
TBL_INTx    LJP         IRQ0_svr
            LJP         IRQ1_svr
            LJP         IRQ2_svr
            LJP         IRQ3_svr
            LJP         IRQ4_svr
            LJP         IRQ5_svr
            LJP         IRQ6_svr
            LJP         IRQ7_svr
IRQ0_srv    → service for IRQ0
            •
            •
            POP         R1
            POP         R0
            POP         SR1
            IRET
IRQ1_srv    → service for IRQ1
            •
            •
            POP         R1
            POP         R0
            POP         SR1
            IRET
            •
            •
IRQ7_srv    → service for IRQ7
            •
            •
            POP         R1
            POP         R0
            POP         SR1
            IRET
```

NOTES:

1. If the SR0 register is changed in the interrupt service routine, then the SR0 register must be pushed and popped in the interrupt service routine.
2. The above example assumes that the ROM size is less than 64K word and all the LJP instructions in the jump table (TBL-INTx) is in the same page.

7 COPROCESSOR INTERFACE

OVERVIEW

CalmRISC supports an efficient and seamless interface with coprocessors. By integrating a MAC (multiply and accumulate) DSP coprocessor engine with the CalmRISC core, not only the microcontroller functions but also complex digital signal processing algorithms can be implemented in a single development platform (or MDS). CalmRISC has a set of dedicated signal pins, through which data/command/status are exchanged to and from a coprocessor. Depicted below are the coprocessor signal pins and the interface between two processors.

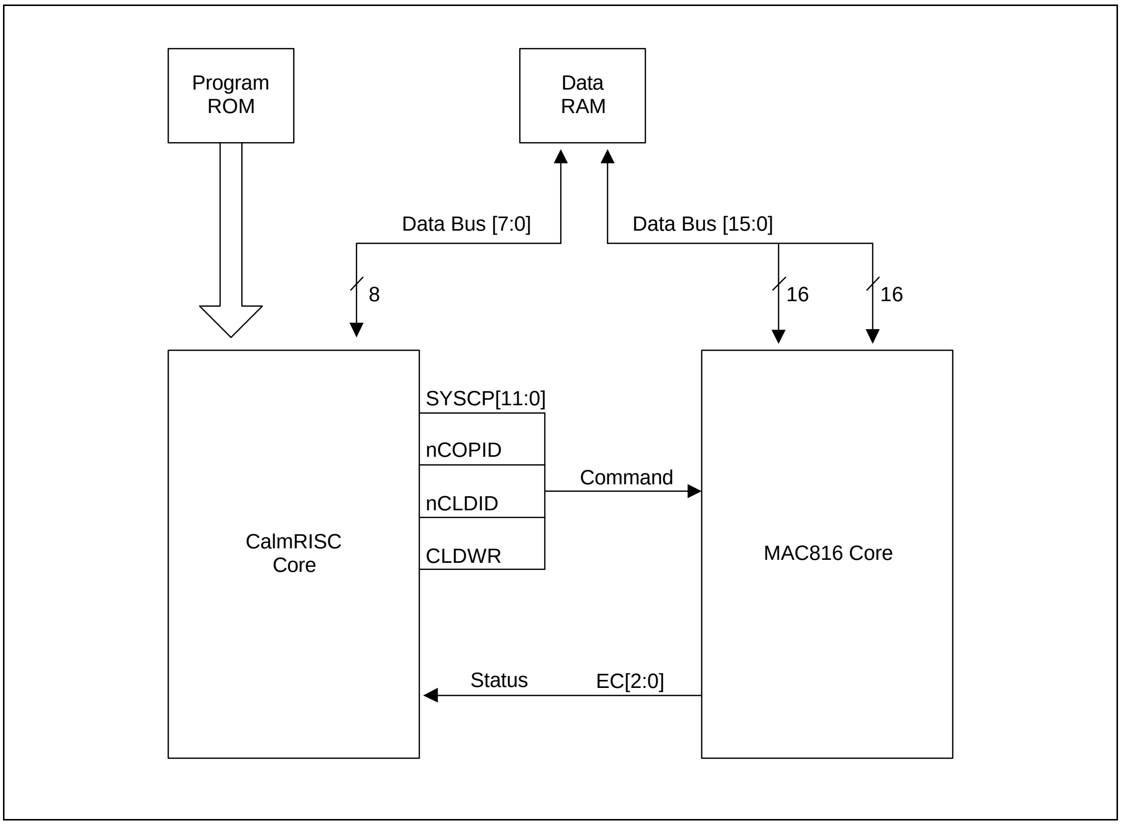


Figure 7-1. Coprocessor Interface Diagram

As shown in the coprocessor interface diagram above, the coprocessor interface signals of CalmRISC are: *SYSCP[11:0]*, *nCOPID*, *nCLDID*, *nCLDWR*, and *EC[2:0]*. The data are exchanged through data buses, *DI[7:0]* and *DO[7:0]*. A command is issued from CalmRISC to a coprocessor through *SYSCP[11:0]* in COP instructions. The status of a coprocessor can be sent back to CalmRISC through EC[2:0] and these flags can be checked in the condition codes of branch instructions. The coprocessor instructions are listed in the following table

Table 7-1. Coprocessor instructions

Mnemonic	Op 1	Op 2	Description
COP	#imm:12	–	Coprocessor operation
CLD	GPR	imm:8	Data transfer from coprocessor into GPR
CLD	imm:8	GPR	Data transfer of GPR to coprocessor
JP(or JR) CALL LNK	EC2–EC0	label	Conditional branch with coprocessor status flags

The coprocessor of CalmRISC does not have its own program memory (i.e., it is a passive coprocessor) as shown in Figure 7 -1. In fact, the coprocessor instructions are fetched and decoded by CalmRISC, and CalmRISC issues the command to the coprocessor through interface signals. For example, if “COP #imm:12” instruction is fetched, then the 12-bit immediate value (imm:12) is loaded on *SYSCP[11:0]* signal with *nCOPID* active in ID/MEM stage, to request the coprocessor to perform the designated operation. The interpretation of the 12-bit immediate value is totally up to the coprocessor. By arranging the 12-bit immediate field, the instruction set of the coprocessor is determined. In other words, CalmRISC only provides a set of generic coprocessor instructions, and its installation to a specific coprocessor instruction set can differ from one coprocessor to another. CLD Write instructions (“CLD imm:8, GPR”) put the content of a GPR register of CalmRISC on the data bus (*DO[7:0]*) and issue the address(imm:8) of the coprocessor internal register on *SYSCP[7:0]* with *nCLDID* active and *CLDWR* active. CLD Read instructions (“CLD GPR, imm:8” in Table 7-1) work similarly, except that the content of the coprocessor internal register addressed by the 8-bit immediate value is read into a GPR register through *DI[7:0]* with *nCLDID* active and *CLDWR* deactivated.

The timing diagram given below is a coprocessor instruction pipeline and shows when the coprocessor performs the required operations. Suppose I_2 is a coprocessor instruction. First, it is fetched and decoded by CalmRISC (at t = T(i-1)). Once it is identified as a coprocessor instruction, CalmRISC indicates to the coprocessor the appropriate command through the coprocessor interface signals (at t = T(i)). Then the coprocessor performs the designated tasks at t = T(i) and t = T(i+1). Hence IF from CalmRISC and then ID/MEM and EX from the coprocessor constitute the pipeline for I_2. Similarly, if I_3 is a coprocessor instruction, the coprocessor’s ID/MEM and EX stages replace the corresponding stages of CalmRISC.

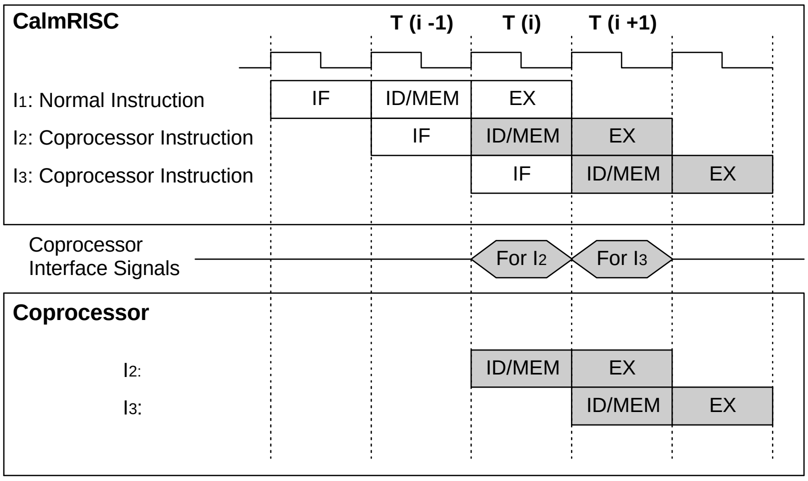


Figure 7-2. Coprocessor Instruction Pipeline

In a multi-processor system, the data transfer between processors is an important factor in determining the efficiency of the overall system. Suppose an input data stream is accepted by a processor, in order for the data to be shared by another processors. There should be some efficient mechanism to transfer the data to the processors. In CalmRISC, data transfers are accomplished through a single shared data memory. The shared data memory in a multi-processor has some inherent problems such as data hazards and deadlocks. However, the coprocessor in CalmRISC accesses the shared data memory only at the designated time at which time CalmRISC is guaranteed not to access the data memory, and therefore there is no contention over the shared data memory. Another advantage of the scheme is that the coprocessor can access the data memory in its own bandwidth.

9 CLOCK CIRCUIT

OVERVIEW

The clock frequency generated for the S3CB018/FB018 by an external crystal can range from 0.4MHz to 20MHz. The Xin and Xout pins connect the external oscillator or clock source to the on-chip clock circuit.

SYSTEM CLOCK CIRCUIT

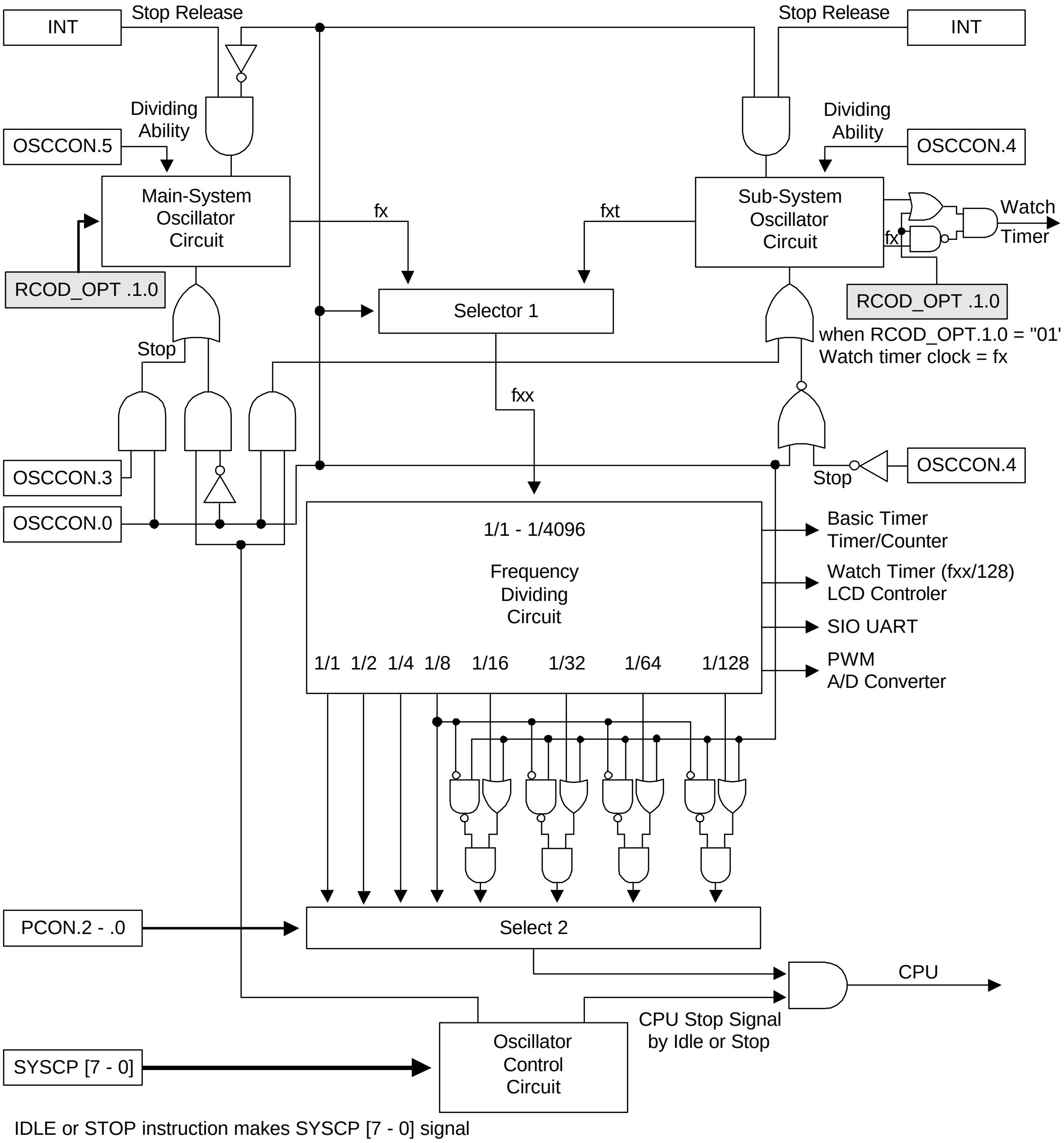
The system clock circuit has the following components:

— External crystal or ceramic resonator oscillation source (or an external clock source)

— Oscillator stop and wake-up functions

— Programmable frequency divider for the CPU clock (Fosc divided by 1, 2, 4, 8, 16, 32, 64, 128)

— System clock control register, PCON



NOTE: S3CB018/FB018 has no sub-oscillator, so there is no effect for selecting a sub-oscillator. The main oscillation is always available.

Figure 9-1. System Clock Circuit Diagram

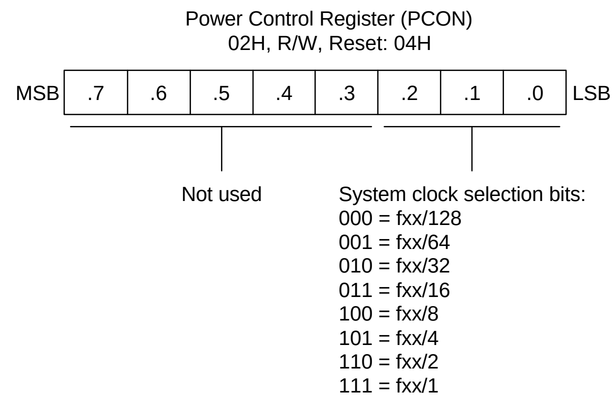
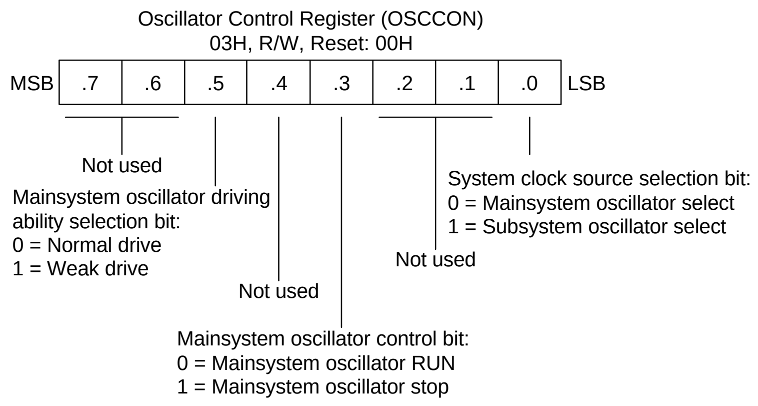


Figure 9-2. Power Control Register (PCON)


Oscillator Control Register (OSCCON)
03H, R/W, Reset: 00H
MSB
.7
.6
.5
.4
.3
.2
.1
.0
LSB
Not used
Mainsystem oscillator driving
ability selection bit:
0 = Normal drive
1 = Weak drive
Not used
Mainsystem oscillator control bit:
0 = Mainsystem oscillator RUN
1 = Mainsystem oscillator stop
Not used
System clock source selection bit:
0 = Mainsystem oscillator select
1 = Subsystem oscillator select

NOTE: This device doesn't have a subsystem oscillator, so you should keep the value of bit 0 and bit 3 to "0".

Figure 9-3. Oscillator Control Register (OSCCON)

10 RESET AND POWER-DOWN

OVERVIEW

During a power-on reset, the voltage at V_{DD} goes to High level and the RESET pin is forced to Low level. The RESET signal is input through a Schmitt trigger circuit where it is then synchronized with the CPU clock. This procedure brings S3CB018/FB018 into a known operating status.

During the time required for the CPU clock oscillation to stabilize, the RESET pin must be held to low level for a minimum time, after the power supply comes within tolerance. This minimum time interval is identified in the electrical characteristics.

In summary, the following sequence of events occurs during a reset operation:

— All interrupts are disabled.

— The watchdog function (basic timer) is enabled.

— Ports are set to input mode except port 1 which is set to output mode.

— Peripheral control and data registers are disabled and reset to their default hardware values.

— The program counter (PC) is loaded with the program reset address in the ROM, 00000H.

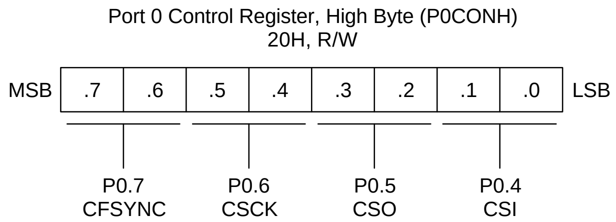
— When the programmed oscillation stabilization time interval has elapsed, the instruction stored in ROM location 00000H is fetched and executed.

NOTE

To program the duration of the oscillation stabilization interval, you make the appropriate settings to the basic timer control register, BTCON, before entering STOP mode. Also, if you do not want to use the basic timer watchdog function (which causes a system reset if a basic timer counter overflow occurs), you can disable it by writing ‘1010 0101b’ to the WDTEN register.

11 I/O PORTS

PORT 0

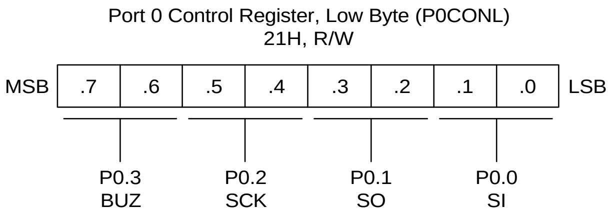


P0CONH bit-pair pin configuration settings:

0 0	Input mode (CSI, CSCK, CFSYNC)
0 1	Input mode, pull-up
1 0	Alternative mode (CSO the others are push pull output)
1 1	Output mode, push-pull

NOTE: For S3CB018/FB018, this port is shared with ADDA codec interface; however for S3EB010, all pins for the ADDA codec interface have their own pins, so in mode "10", these pins are floated.

Figure 11-1. Port 0 High-byte Control Register (P0CONH)

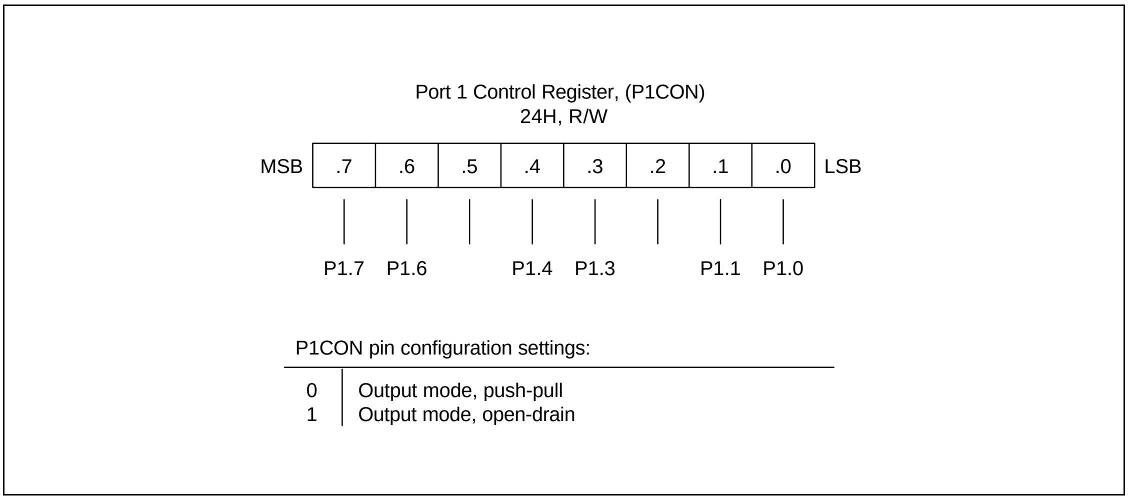


P0CONL bit-pair pin configuration settings:

0 0	Input mode (SI, SCK input)
0 1	Input mode, pull-up
1 0	Alternative mode (SCK output, SO, BUZ output, P0.0 is Push-pull output mode)
1 1	Output mode, push-pull

Figure 11-2. Port 0 Low-byte Control Register (P0CONL)

PORT 1

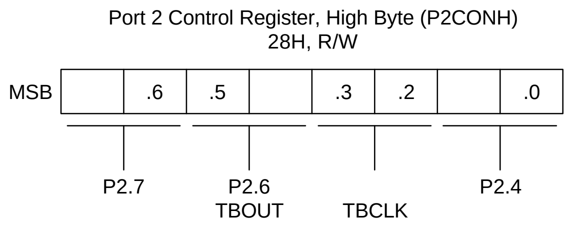


P1CON pin configuration settings:

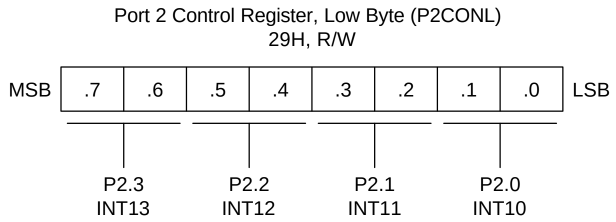
0	Output mode, push-pull
1	Output mode, open-drain

Figure 11-3. Port 1 High-byte Control Register (P1CON)

PORT 2



0 0	Input mode (TACLK, TBCLK)
0 1	Input mode, pull-up
	are push-pull output mode)
1 1	Output mode, push-pull



P2CONL bit-pair pin configuration

0 0	Input mode, rising edge interrupt
0 1	Input mode, falling edge interrupt
1 0	Input mode, pull-up, falling edge interrupt
1 1	Output mode, push-pull

Figure 11-5. Port 2 Low-byte Control Register (P2CONL)

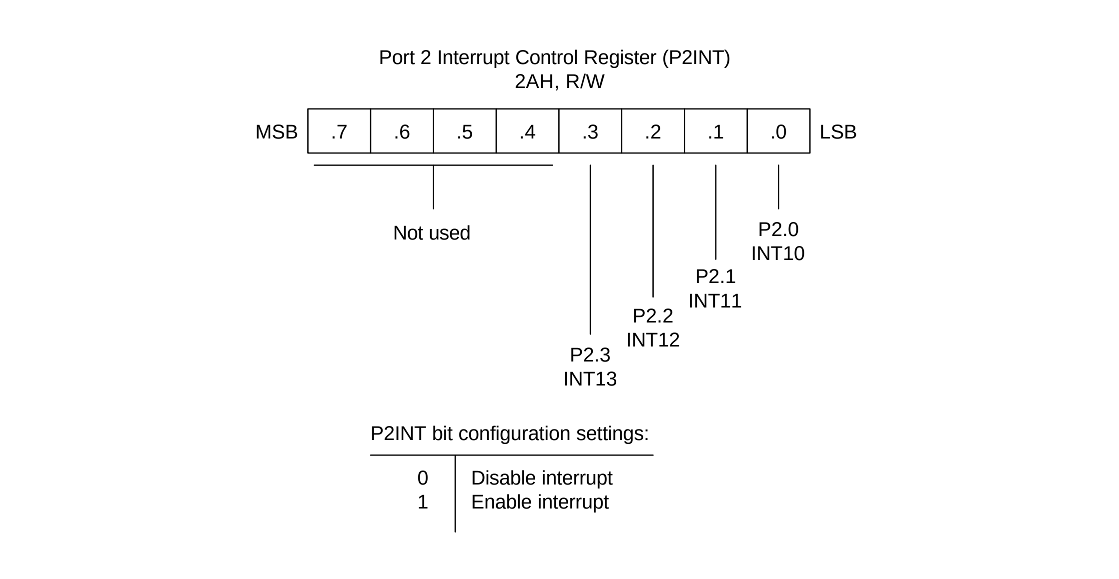
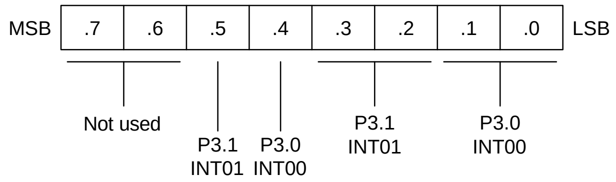


Figure 11-6. Port 2 Interrupt Control Register (P2INT)

PORT 3

Port 3 Control Register (P3CON)
2CH, R/W



P3CON bit-pair(.3 - .0) pin configuration

0 0	Input mode, rising edge interrupt
0 1	Input mode, falling edge interrupt
1 0	Input mode, pull-up, falling edge interrupt
1 1	Output mode, push-pull

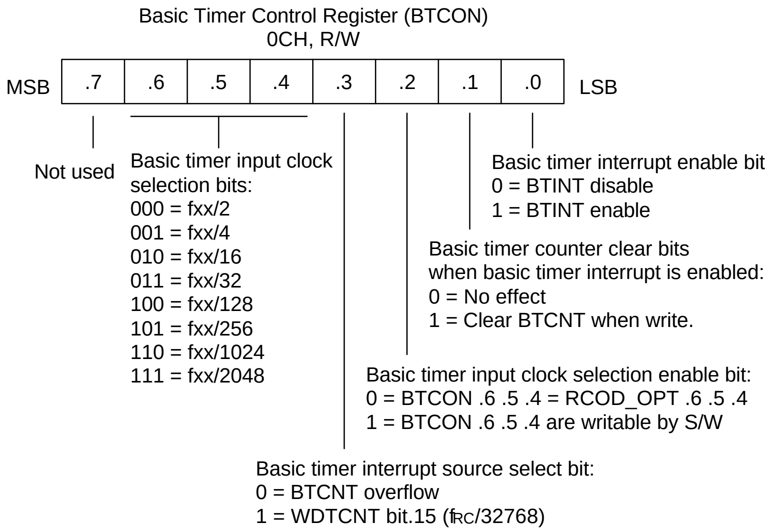
P3CON bit-pair(.5 - .4) pin configuration

0	Disable interrupt
1	Enable interrupt

Figure 11-7. Port 3 Control Register (P3CON)

12 BASIC TIMER

OVERVIEW


Basic Timer Control Register (BTCON)
0CH, R/W
MSB
.7
.6
.5
.4
.3
.2
.1
.0
LSB
Not used
Basic timer input clock selection bits:
000 = fxx/2
001 = fxx/4
010 = fxx/16
011 = fxx/32
100 = fxx/128
101 = fxx/256
110 = fxx/1024
111 = fxx/2048
Basic timer interrupt enable bit
0 = BTINT disable
1 = BTINT enable
Basic timer counter clear bits when basic timer interrupt is enabled:
0 = No effect
1 = Clear BTCNT when write.
Basic timer input clock selection enable bit:
0 = BTCON .6 .5 .4 = RCOD_OPT .6 .5 .4
1 = BTCON .6 .5 .4 are writable by S/W
Basic timer interrupt source select bit:
0 = BTCNT overflow
1 = WDTCNT bit.15 (f_{RC}/32768)

NOTE: Under the reset operation BTCON's value is affected by RCOD_OPT. After reset if you set the BTCON.2, then the BTCON .6 .5 .4 written by software.

Figure 12-1. Basic Timer Control Register (BTCON)

WATCHDOG TIMER

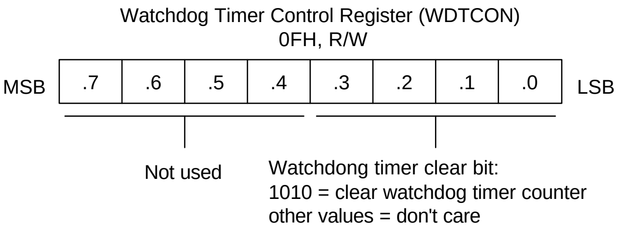

Watchdog Timer Control Register (WDTCON)
0FH, R/W
MSB
.7
.6
.5
.4
.3
.2
.1
.0
LSB
Not used
Watchdong timer clear bit:
1010 = clear watchdog timer counter
other values = don't care

Figure 12-2. Watchdog Timer Control Register (WDTCON)

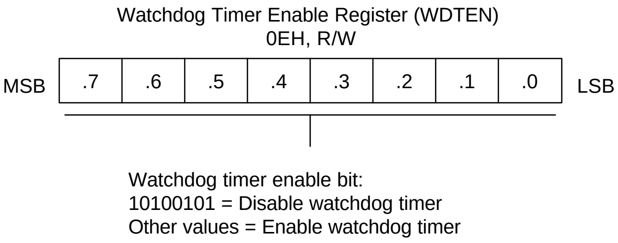
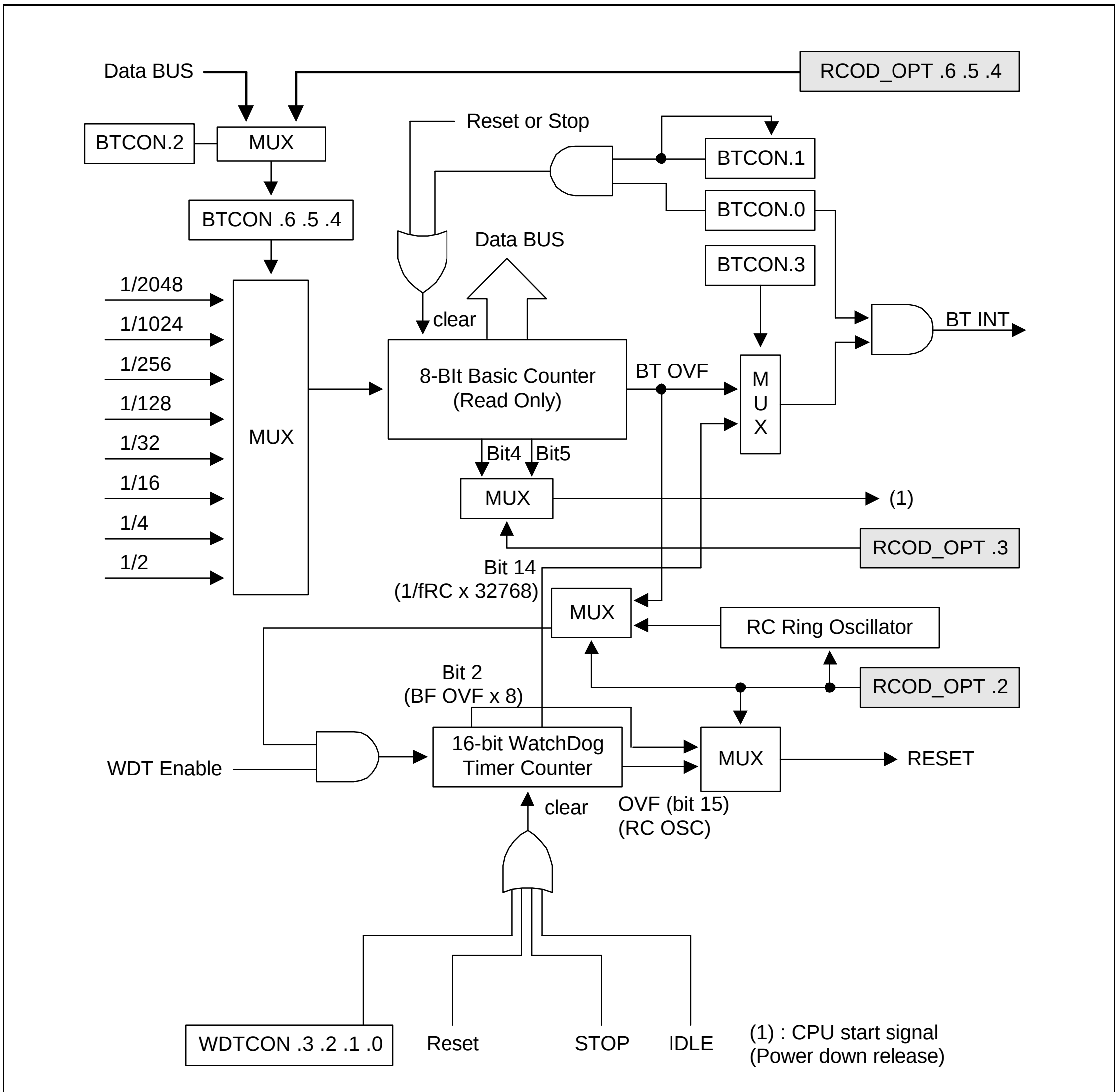

Watchdog Timer Enable Register (WDTEN)
0EH, R/W
MSB
.7
.6
.5
.4
.3
.2
.1
.0
LSB
Watchdog timer enable bit:
10100101 = Disable watchdog timer
Other values = Enable watchdog timer

Figure 12-3. Watchdog Timer Enable Register (WDTEN)

BLOCK DIAGRAM



NOTE: The frequency of RC ring oscillator is 10MHz ± 30% at V_{DD} =5.5 V

Figure 12-4. Basic Timer & Watchdog Timer Functional Block Diagram

13 WATCH TIMER

OVERVIEW

Table 13-1. Watch Timer Control Register (WTCON): 8-Bit R/W

Bit Name	Values		Function	Address
WTCON.7	–		Not used	50H
WTCON.6	–		Not used	
WTCON .5 - .4	0	0	0.5 kHz buzzer (BUZ) signal output	
	0	1	1 kHz buzzer (BUZ) signal output	
	1	0	2 kHz buzzer (BUZ) signal output	
	1	1	4 kHz buzzer (BUZ) signal output	
WTCON .3 - .2	0	0	Set watch timer interrupt to 1S.	
	0	1	Set watch timer interrupt to 0.5S.	
	1	0	Set watch timer interrupt to 0.25S.	
	1	1	Set watch timer interrupt to 3.91mS.	
WTCON.1	0		Select (fx/128) as the watch timer clock	
	1		No effect, fx/128 is selected.	
WTCON.0	0		Stop watch timer counter; clear frequency dividing circuits	
	1		Run watch timer counter	

NOTE: Main system clock frequency (fx) is assumed to be 4.195 MHz

WATCH TIMER CIRCUIT DIAGRAM

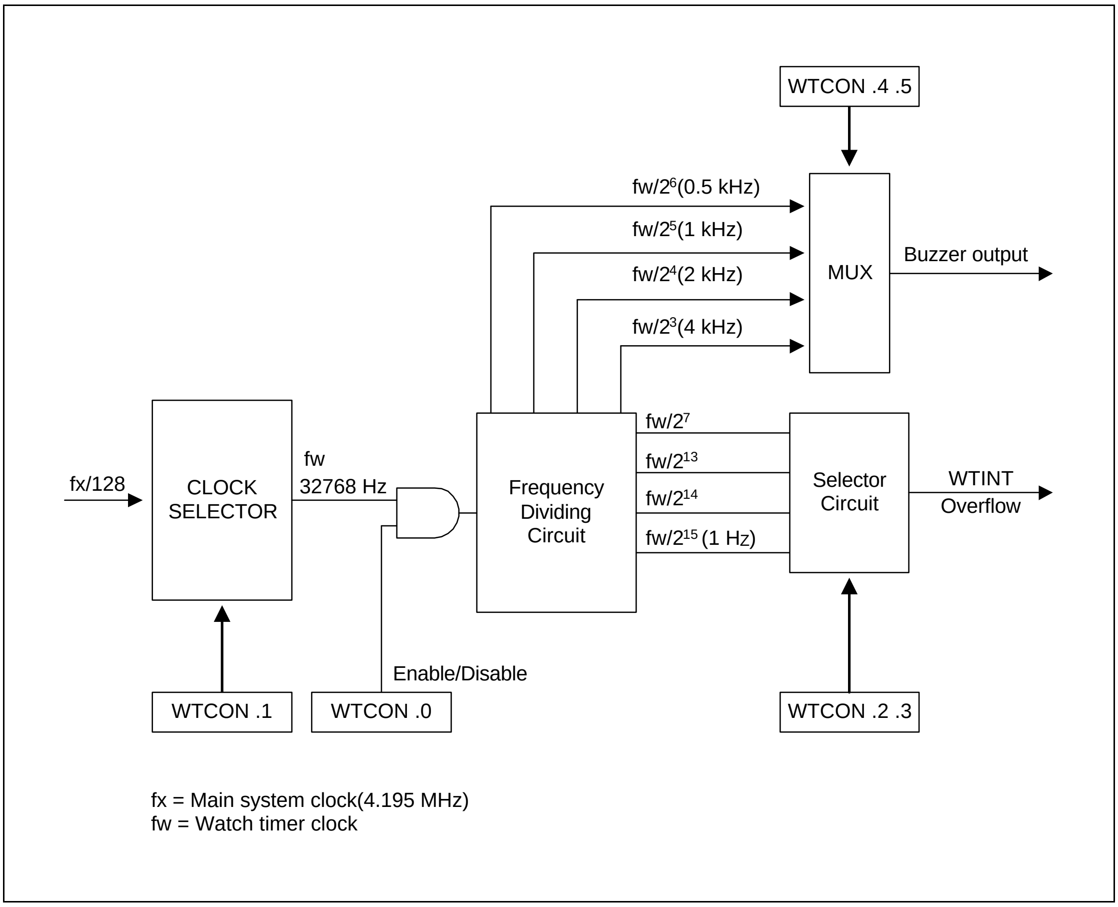
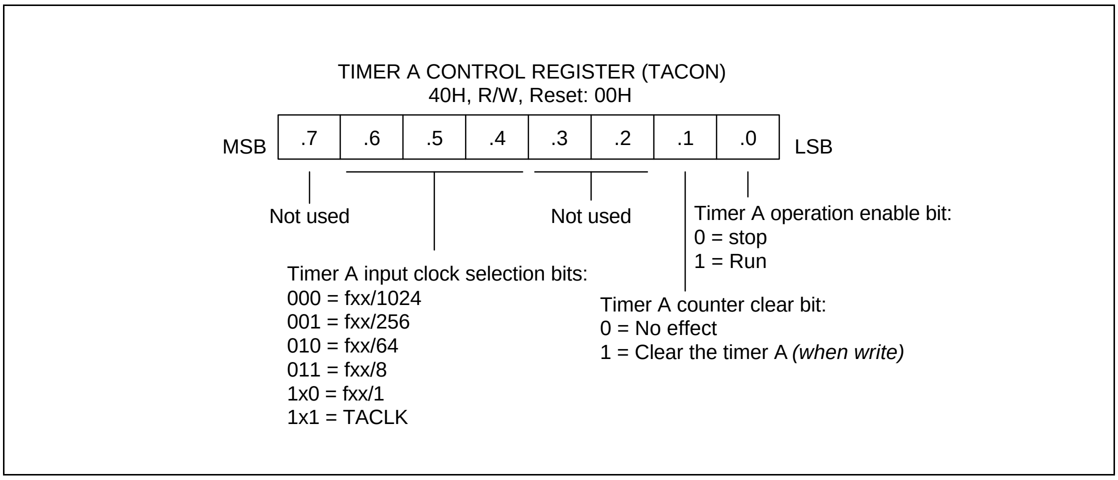


Figure 13-1. Watch Timer Circuit Diagram

14 16-BIT TIMER (8-BIT TIMER A & B)

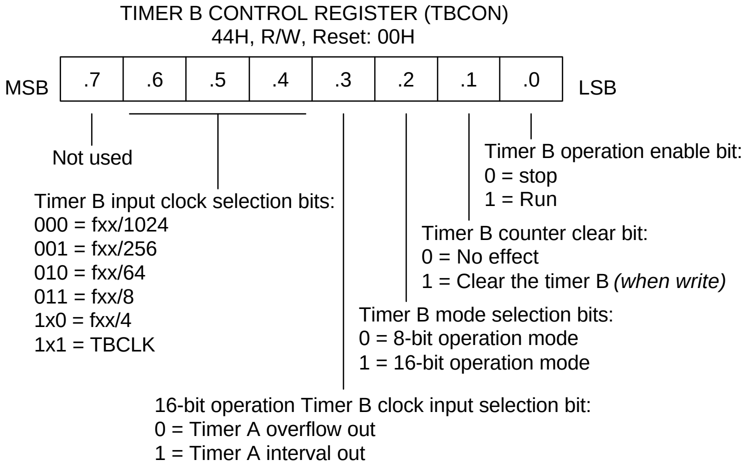
OVERVIEW

This 16-bit timer has two modes. One is 16-bit timer mode and the other is two 8-bit timer mode. When Bit 2 of TBCON is “0”, it operates with the 16-bit timer. When it is “1”, it operates with two 8-bit timers. When it operates with the 16-bit timer, the TBCNT’s clock source can be selected by setting TBCON.3. If TBCON.3 is “0”, the timer A’s overflow would be TBCNT’s clock source. If it is “1”, the timer A’s interval out would be TBCNT’s clock source. The timer clock source can be selected by S/W.



TIMER A CONTROL REGISTER (TACON)
40H, R/W, Reset: 00H
MSB
.7
.6
.5
.4
.3
.2
.1
.0
LSB
Not used
Not used
Timer A operation enable bit:
0 = stop
1 = Run
Timer A input clock selection bits:
000 = fxx/1024
001 = fxx/256
010 = fxx/64
011 = fxx/8
1x0 = fxx/1
1x1 = TACLK
Timer A counter clear bit:
0 = No effect
1 = Clear the timer A *(when write)*

Figure 14-1. Timer A Control Register (TACON)


TIMER B CONTROL REGISTER (TBCON)
44H, R/W, Reset: 00H
MSB
.7
.6
.5
.4
.3
.2
.1
.0
LSB
Not used
Timer B input clock selection bits:
000 = fxx/1024
001 = fxx/256
010 = fxx/64
011 = fxx/8
1x0 = fxx/4
1x1 = TBCLK
16-bit operation Timer B clock input selection bit:
0 = Timer A overflow out
1 = Timer A interval out
Timer B mode selection bits:
0 = 8-bit operation mode
1 = 16-bit operation mode
Timer B counter clear bit:
0 = No effect
1 = Clear the timer B *(when write)*
Timer B operation enable bit:
0 = stop
1 = Run

NOTE: At 16-bit operation mode 16-bit counter clock input is selected by TACON .6,.5, .4

Figure 14-2. Timer B Control Register (TBCON)

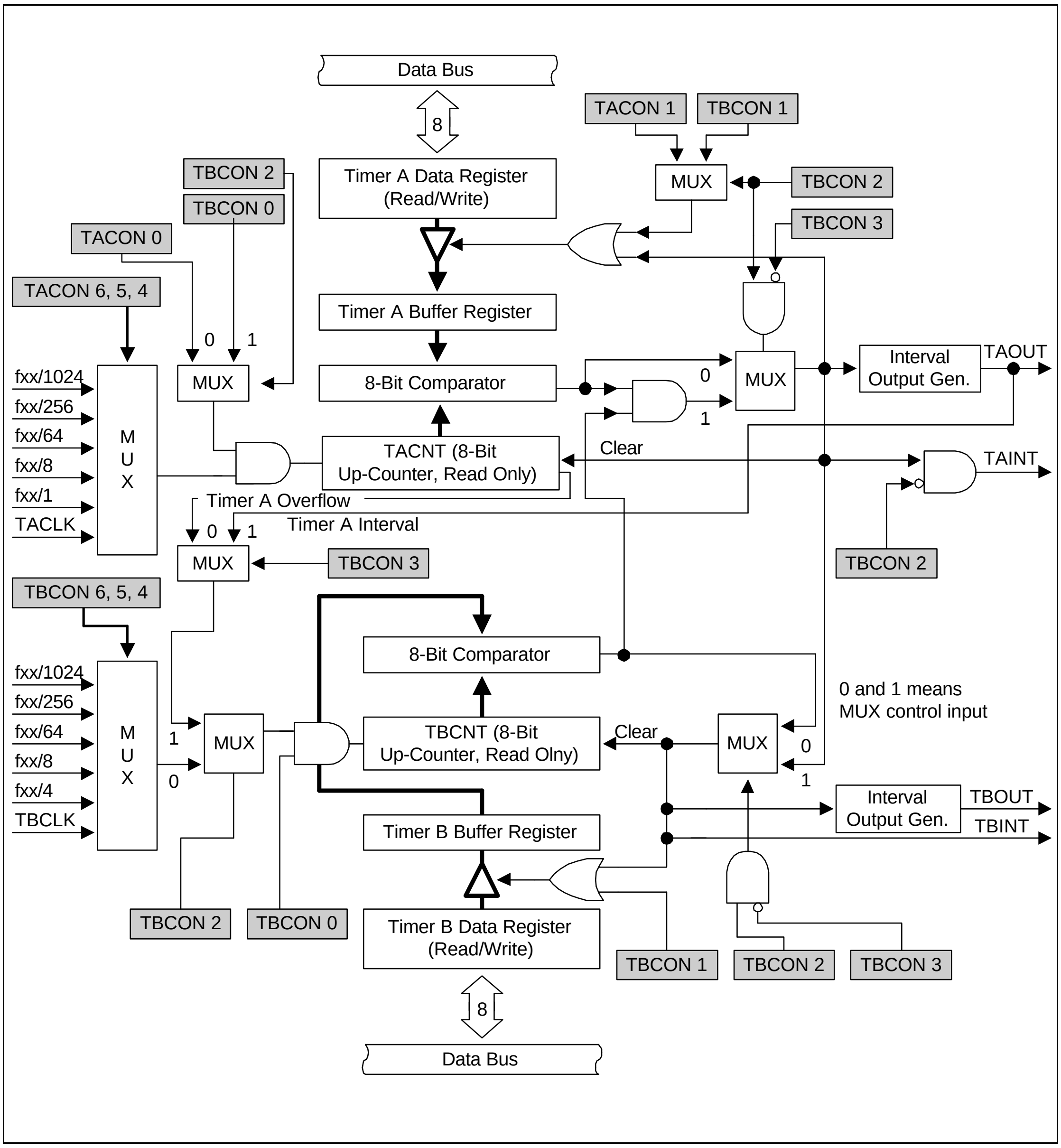


Figure 14-3. Timer A, B Function Block Diagram

15 SERIAL I/O INTERFACE

OVERVIEW

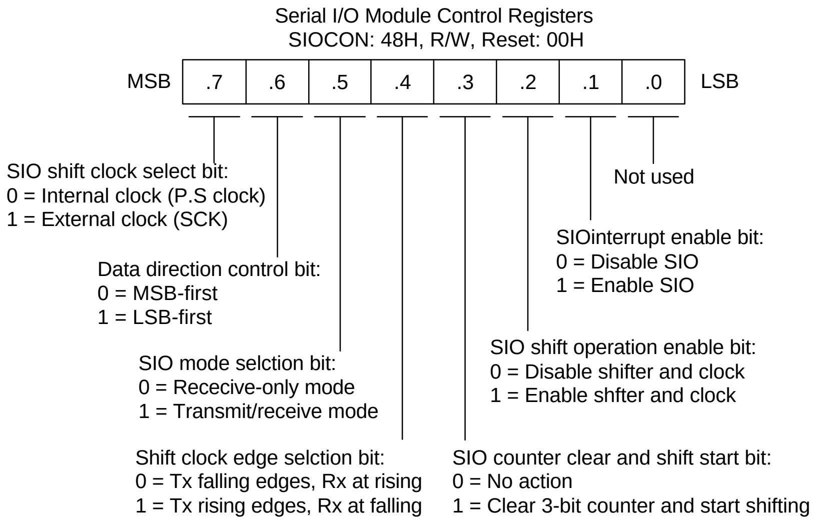

Serial I/O Module Control Registers
SIOCON: 48H, R/W, Reset: 00H
MSB
.7
.6
.5
.4
.3
.2
.1
.0
LSB
SIO shift clock select bit:
0 = Internal clock (P.S clock)
1 = External clock (SCK)
Not used
SIOinterrupt enable bit:
0 = Disable SIO
1 = Enable SIO
Data direction control bit:
0 = MSB-first
1 = LSB-first
SIO mode selction bit:
0 = Rececive-only mode
1 = Transmit/receive mode
SIO shift operation enable bit:
0 = Disable shifter and clock
1 = Enable shfter and clock
Shift clock edge selction bit:
0 = Tx falling edges, Rx at rising
1 = Tx rising edges, Rx at falling
SIO counter clear and shift start bit:
0 = No action
1 = Clear 3-bit counter and start shifting

Figure 15-1. Serial I/O Module Control Registers (SIOCON)

SIO PRE-SCALER REGISTER (SIOPS)

The control register for serial I/O interface module, SIOPS, is located at 49H. The value stored in the SIO pre-scaler registers, SIOPS, lets you determine the SIO clock rate (baud rate) as follows:

Baud rate = Input clock(X_{IN}/8)/(Pre-scaler value + 1), or, SCLK input clock

where the input clock is X_{IN} / 4.

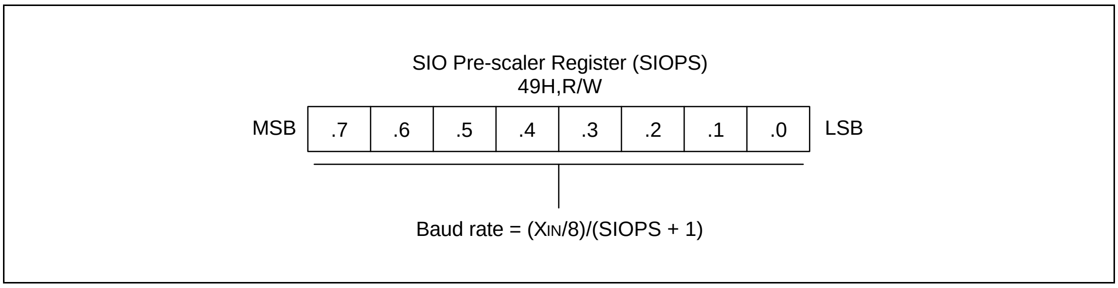

SIO Pre-scaler Register (SIOPS)
49H,R/W
MSB
.7
.6
.5
.4
.3
.2
.1
.0
LSB
Baud rate = (XIN/8)/(SIOPS + 1)

Figure 15-2. SIO Pre-scaler Register (SIOPS)

BLOCK DIAGRAM

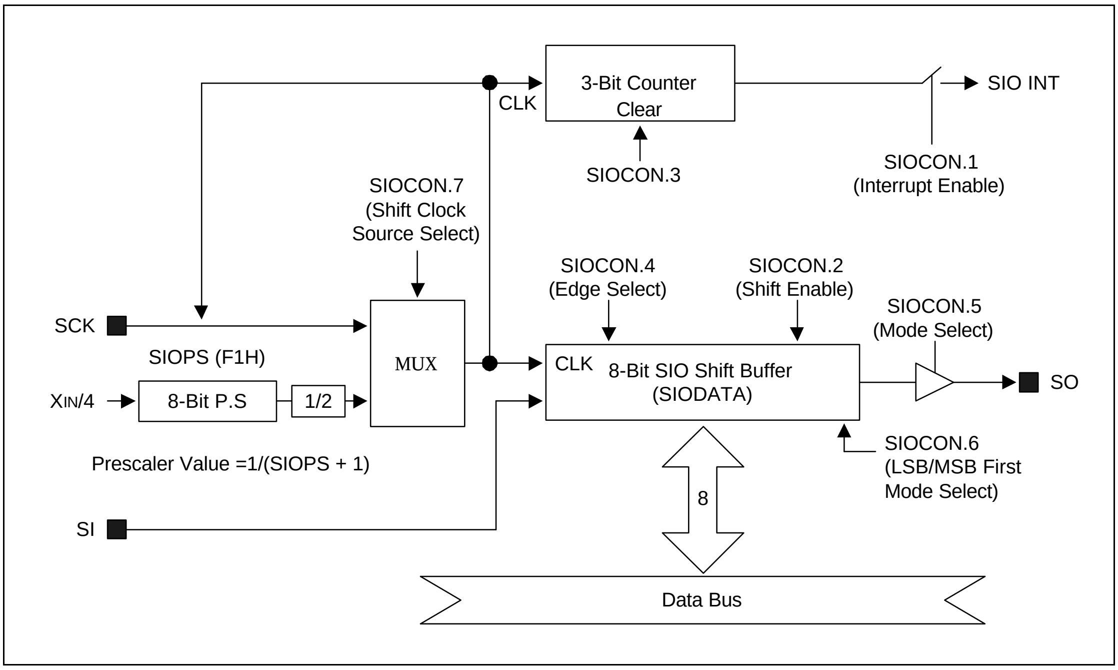


Figure 15-3. SIO Function Block Diagram

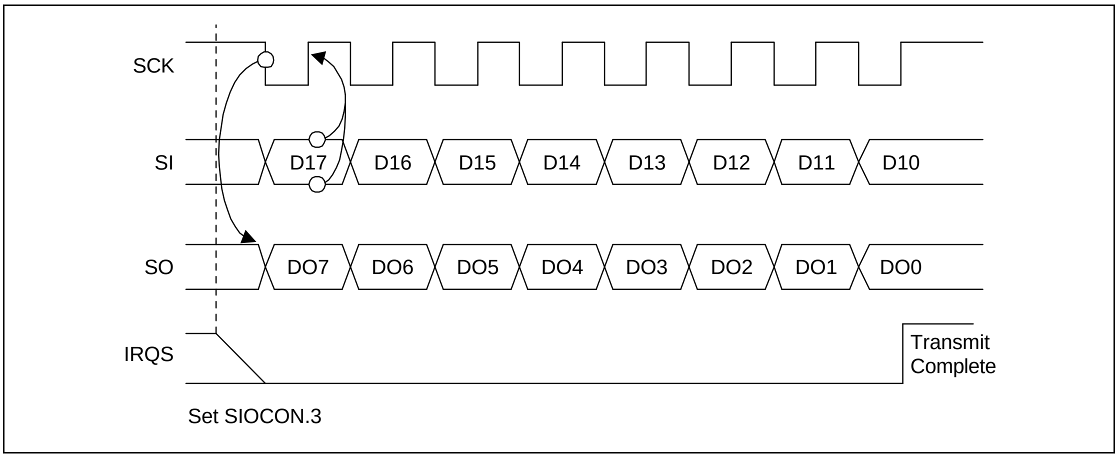


Figure 15-4. Serial I/O Timing in Transmit/Receive Mode(Tx at falling, SIOCON.4=0)

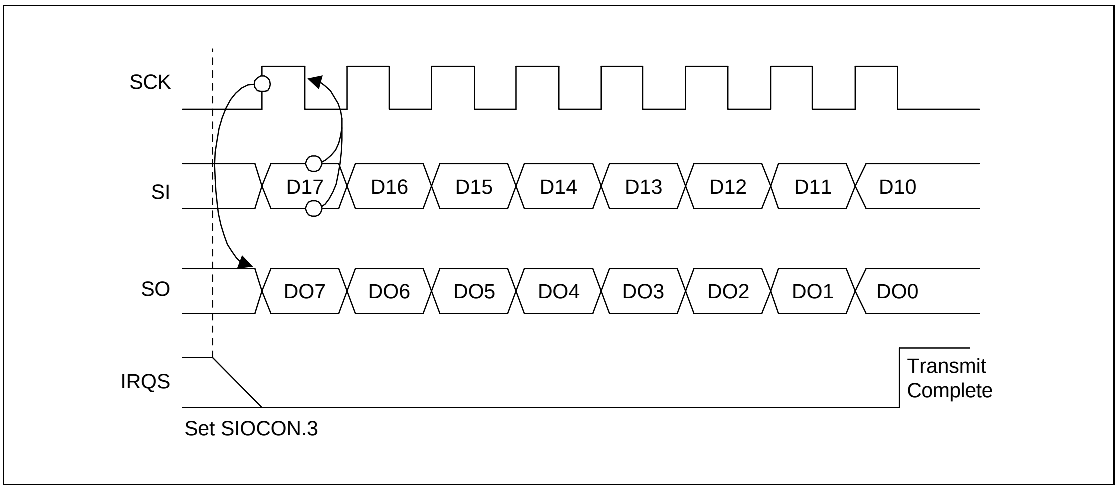


Figure 15-5. Serial I/O Timing in Transmit/Receive Mode(Tx at rising, SIOCON.4=1)

16 SERIAL I/O FOR ADC/DAC

OVERVIEW

ADC/DAC Serial I/O module, ADDASIO, can interface with SM4 Master, 32 BPF mode of CS4216 (note). The components of each ADDASIO function block are :

— 8-bit control register (ADDACON)

— 16-bit Data buffer (ADCHIGH, ADCLOW, DACHIGH, DACLOW)

— Serial data I/O pins (CSIN, CSOUT)

— Frame sync. pin (CFSYNC)

— External clock input/out pin (CSCLK)

The ADDASIO module can transmit or receive 2 channels of 16-bit serial data configured by its corresponding control register settings. The ADDASIO module operates with slave mode only.

Programming Procedure

To program the ADDASIO modules, follow these basic steps:

1. Load an 8-bit value to the ADDACON control register to properly configure the ADDASIO module. (When using CS4216, ADDACON.3 must be cleared and CS4216 is configured to SM4 Master, 32 BPF mode).
2. Enable control (Channel 0/1 Enable, Shift Enable)
3. The ADDASIO interrupt request is automatically generated at the end of 16-bit shifting.
4. In the ADDASIO interrupt routine, read the channel information (ADDACON.5) and read/write ADC/DAC data corresponding channel.
5. Repeat steps 3 to 4.

NOTE: CS4216: analog-to-digital/digital-to-analog converter device supplied by Crystal Semiconductor Corporation in U.S.A.

ADDASIO CONTROL REGISTER (ADDACON)

The control register for ADDASIO interface module, ADDACON, is located at 4CH. It has the control settings for the ADDASIO module.

- Channel 0 Interrupt enable
- Channel 1 Interrupt enable
- Edge selection for shift operation
- IIS format
- Shift operation (transmit/receive) enable
- Channel information.

A reset clears the ADDACON value to '00H'. This means that Tx and Rx are at rising edges and selects the IIS format so that the transition of CFSYNC (Frame sync.) occurs one clock cycle earlier(i.e. together with the last bit of the previous data word). The data shift operation and the interrupt are disabled.

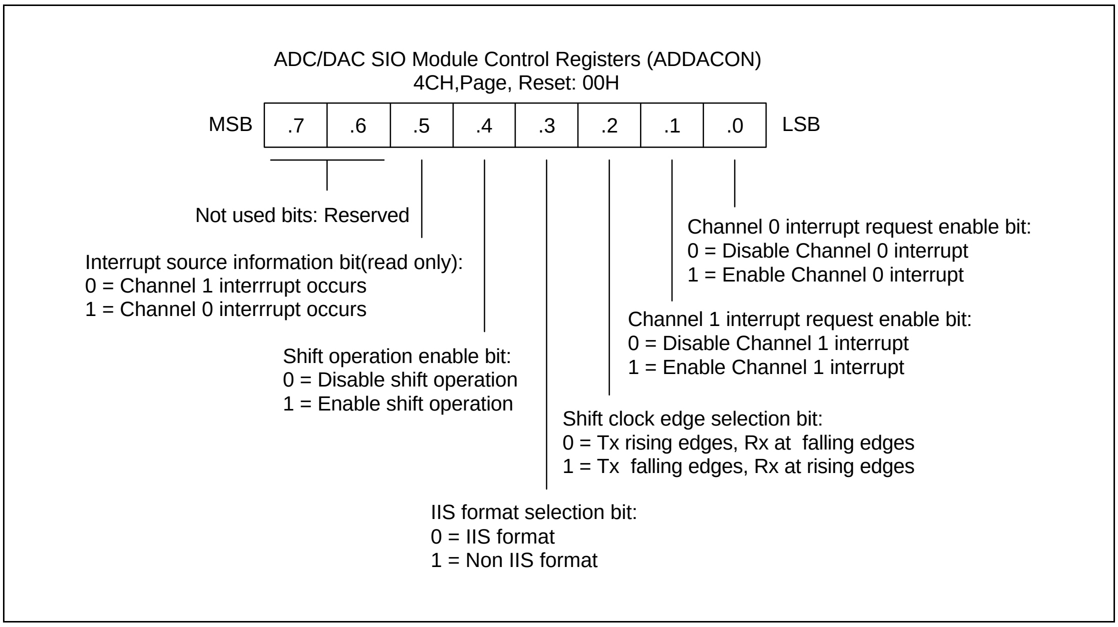

ADC/DAC SIO Module Control Registers (ADDACON)
4CH,Page, Reset: 00H
MSB
.7
.6
.5
.4
.3
.2
.1
.0
LSB
Not used bits: Reserved
Interrupt source information bit(read only):
0 = Channel 1 interrrupt occurs
1 = Channel 0 interrrupt occurs
Shift operation enable bit:
0 = Disable shift operation
1 = Enable shift operation
IIS format selection bit:
0 = IIS format
1 = Non IIS format
Shift clock edge selection bit:
0 = Tx rising edges, Rx at falling edges
1 = Tx falling edges, Rx at rising edges
Channel 1 interrupt request enable bit:
0 = Disable Channel 1 interrupt
1 = Enable Channel 1 interrupt
Channel 0 interrupt request enable bit:
0 = Disable Channel 0 interrupt
1 = Enable Channel 0 interrupt

Figure 16-1. ADC/DAC Serial I/O Module Control Register (ADDACON)

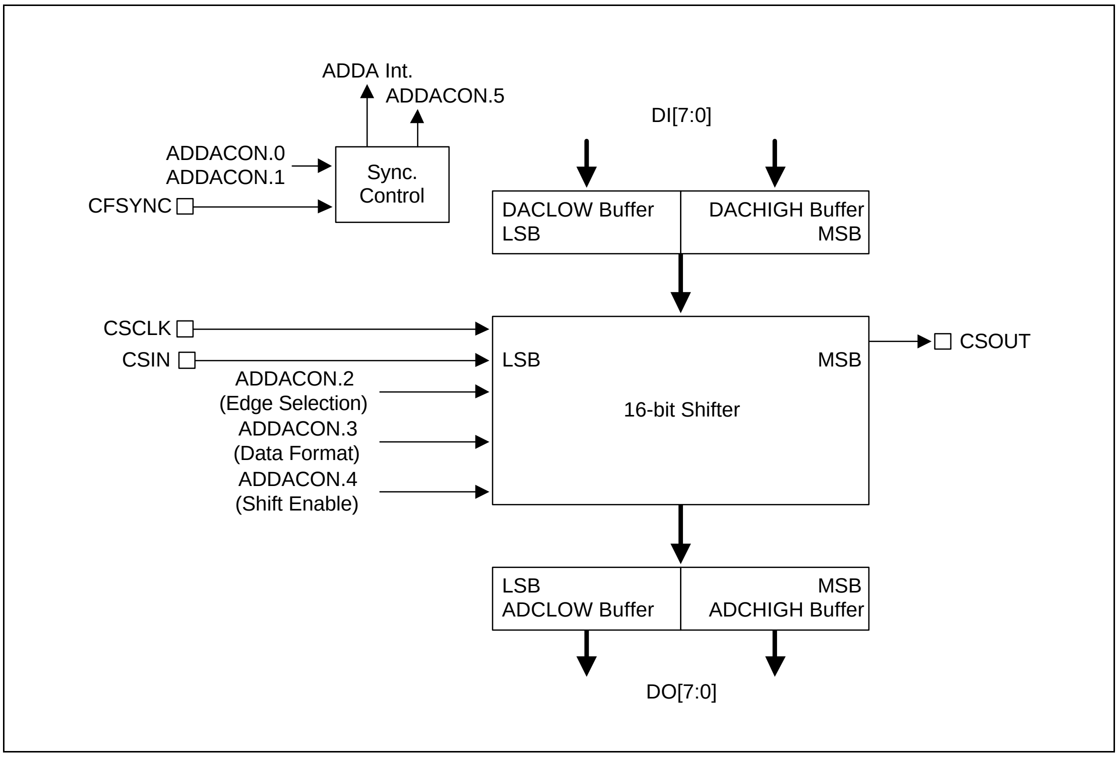


Figure 16-2. ADC/DAC Serial I/O Module Block Diagram

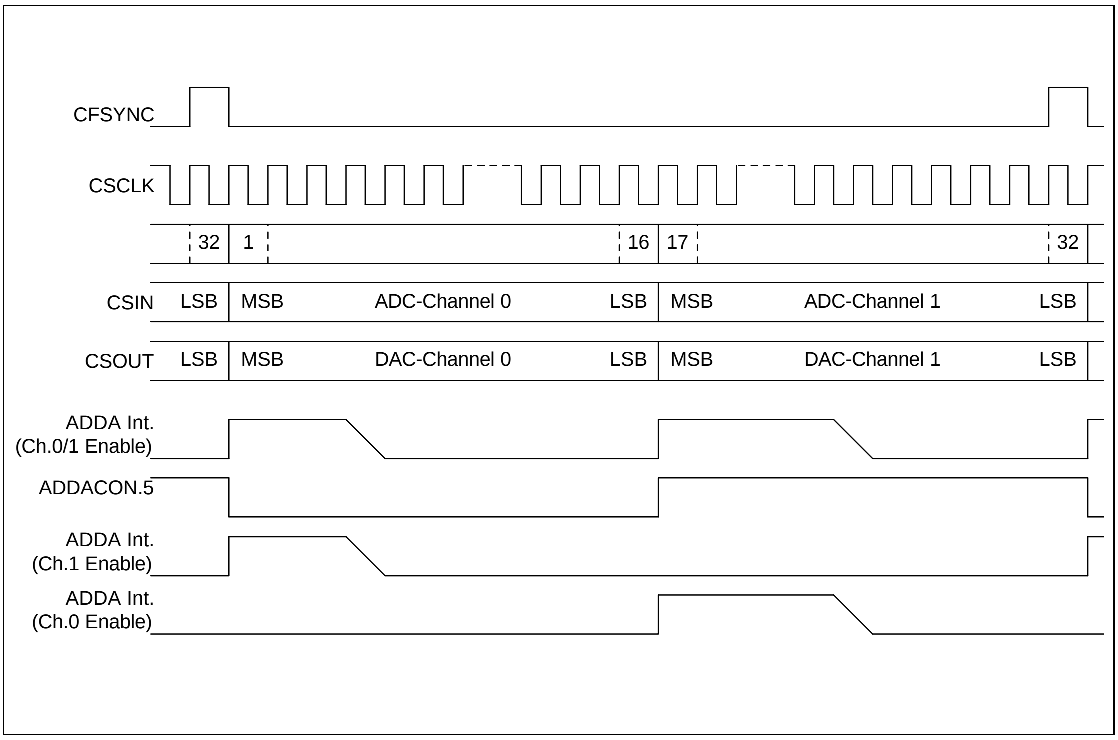


Figure 16-3. ADC/DAC Serial I/O Module Waves

17 MAC816

MAC816 ARCHITECTURE OVERVIEW

MAC816 is a 16-bit fixed-point DSP coprocessor for low-end DSP applications. It is designed as one of the DSP coprocessor engines for

intended operations on MAC816, including the DSP data type, and the DSP addressing mode. Below represented is the top block diagram of MAC816.

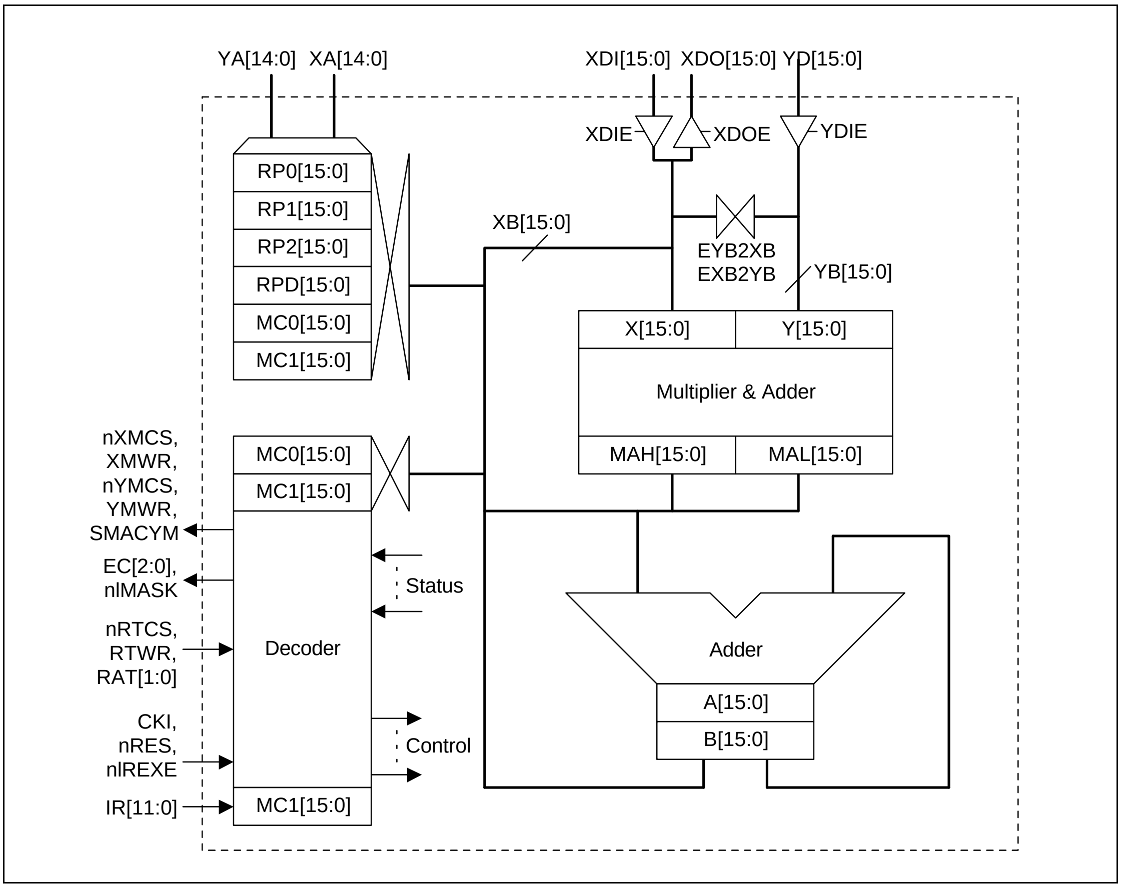


Figure 17-1. Top Block Diagram

The MAC816 building blocks consist of:

— Multiplier and Accumulator Unit (MAU)

— Arithmetic Unit (AU)

— RAM Pointer Unit (RPU)

— Interface Unit (IU)

Basically, MAU (Multiplier and Accumulator Unit) is built around an 8-bit by 16-bit parallel multiplier and a 32-bit adder for multiply-and-accumulate (MAC) operations. Hence, 16-bit by 16-bit MAC operations are performed in two cycles in MAC816. AU performs 16-bit arithmetic and shift operations for DSP. RPU of MAC816 consists of 3 data memory pointers and 2 control blocks for the pointer modulo calculation. The pointers are used for accessing the data memory for a 16-bit data operand. Since two 16-bit data operands can be fetched simultaneously in a single cycle through XD[15:0] and YD[15:0] for MAC operation, the data memory should be partitioned into two parts: X and Y memory. IU is used for the communication between CalmRISC and MAC816. It decodes coprocessor interface signals from CalmRISC and controls the data paths in MAC816, according to the decoding result.

Most of MAC816 instructions are 1-word instruction, while several instructions which need 16-bit immediate value are 2-word instruction.

PROGRAMMER' S MODEL

In this chapter, the important features of MAC816 are discussed in detail. How the data memory is organized is discussed and the explanation of registers follows. Last, the host interface with CalmRISC will be explained.

DATA MEMORY ACCESSES

The total data memory address space for MAC816 is 32 kword. The 32 kword data memory space is physically divided into XM (X area memory) and YM (Y area memory). This memory space is actually shared with the host processor (CalmRISC). The host processor accesses the 64K byte data memory in byte width, otherwise MAC816 accesses it in 2-byte width. MAC816 has two types of addressing modes. RPU can generate two 15-bit addresses every instruction cycle which can be post-modified.

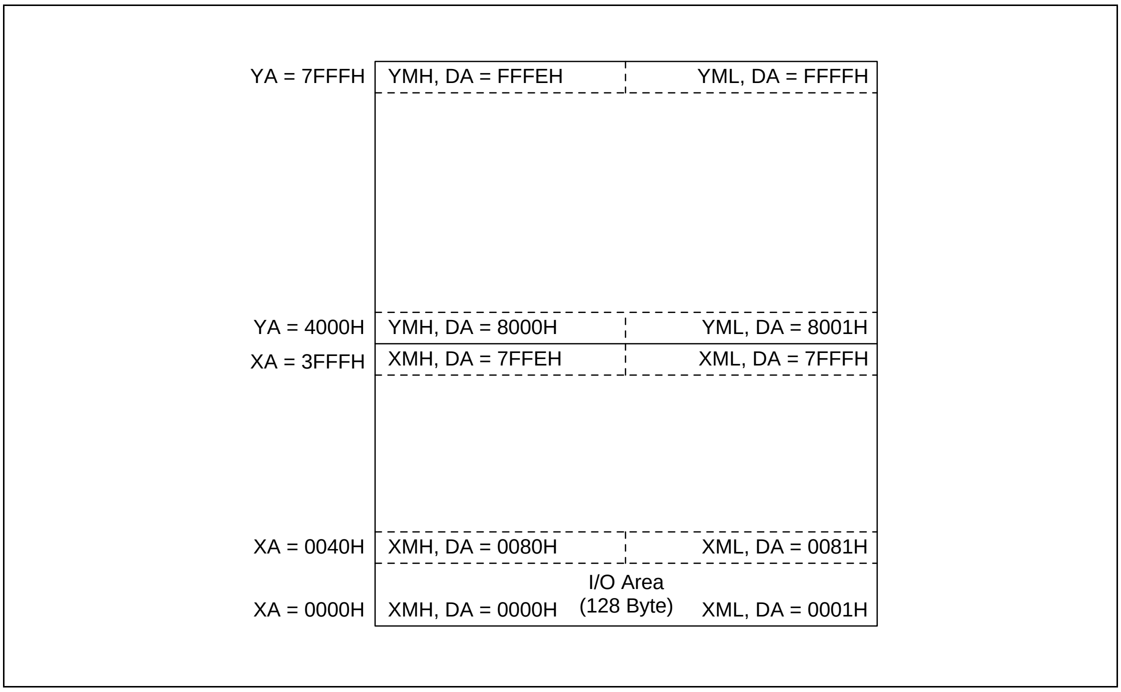


Figure 17-2. Data Memory Organization

Table 18-1. RPU(RAM Pointer Unit) Registers

Registers		Mnemonics	Description	Reset Value
Mreg1	RPi	**RP0**	**R**AM **P**ointer register **0**	Unknown
		RP1	**R**AM **P**ointer register **1**	Unknown
		RP2	**R**AM **P**ointer register **2**	Unknown
		RPD	**R**AM **P**ointer for short direct addressing	Unknown
MCi		**MC0**	**M**odulo **C**ontrol register **0** for RP0/RP1	Unknown
		MC1	**M**odulo **C**ontrol register **1** for RP2	Unknown

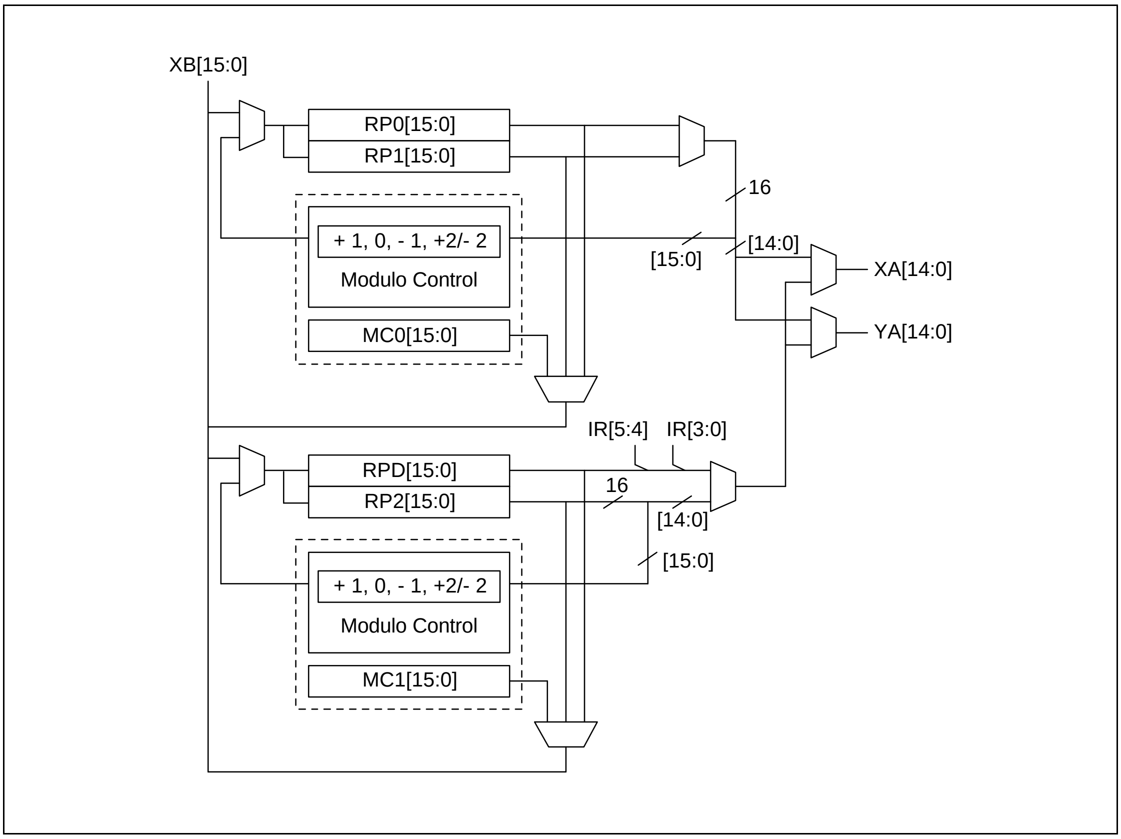


Figure 17-3. RPU(RAM Pointer Unit) Block Diagram

Short Direct Memory Addressing Mode

Six-bits embedded in the instruction code as LSBs and 9-bits from the RPD[14:6] of RPD register as MSB compose the 15-bit address to the data memory address. This can be used with some instructions operating an Ai (A/B register in AU) operand. In "load/store *mreg1*" instruction, a 4-bit embedded in the instruction code as LSBs and 11-bits from the RPD[14:4] of RPD register as MSB compose the 15-bit address to the data memory address. This can be used to load/store RAM pointer register from/to data memory.

Indirect Memory Addressing Mode

The RPi registers of RPU are used as a 15-bit address for indirect addressing XM (X area memory) or YM (Y area memory). Some instructions can simultaneously access the XM and YM; then RP0 is used for XM and RP2 for YM. In indirect addressing mode, the RPi register is modified by +1,-1,-2, and +2 after the addressing. The MSB of the RPi register enables modulo operation of the RPi modification. The RPU registers are divided into two groups of simultaneous addressing over XA and YA: X-memory is addressed by RP0 and RP1 with MC0, and Y-memory is addressed by RP2 with MC1. The RPi from both groups can be used for both XA and YA for instruction, which uses only one address register. In this instruction the XM and YM can be viewed as a single continuous data memory space.

Table 17-2. RPi register bit information

Bit position	Value	Description
[14:0]	0H–7FFFH	Data memory(XM/YM) address
[15]	0	Modulo mode disable
	1	Modulo mode enable

Modulo Control Registers (MCi)

MCi controls RP0, RP1 and RP2 register modifications after indirect memory accessing. MCi has an upper boundary value in MCi[9:0], a step size in MCi[12:10] and a modulo size information in MCi[15:13]. The upper boundary determines the upper limit of the modulo body. The modulo size information determines the lower limit and size of the modulo body as shown below. For example, assume RP0 = 87FFH and MC0 = 03FFH: If "@RP0+" is used on the operand of the instruction, the data memory contents pointed by "07FFH" is accessed, and RP0 is updated to "8400H" after memory accessing. Assume RP0 = 07FFH and MC0 = 03FFH: If "@RP0+" is used on the operand of the instruction, the data memory contents pointed by "07FFH" is accessed, and RP0 is updated to "0800H" after memory accessing.

Bit position	Value	Description
[9:0]	0H–3FFH	Upper boundary
[12:10]	000	Step size = + 2
	001	Step size = - 2
	010–111	Reserved
[15:13]	000	Maximum modulo size = 1024 (0H to 3FFH), Modulo body = RPi[14:10]:0000000000 to RPi[14:10]:MCi[9:0]
	001	Maximum modulo size = 8 (0H to 7H), Modulo body = RPi[14:3]:000 to RPi[14:3]:MCi[2:0]
	010	Maximum modulo size = 16 (0H to 0FH), Modulo body = RPi[14:4]:0000 to RPi[14:4]:MCi[3:0]
	011	Maximum modulo size = 32 (0H to 1FH), Modulo body = RPi[14:5]:00000 to RPi[14:5]:MCi[4:0]
	100	Maximum modulo size = 64 (0H to 3FH), Modulo body = RPi[14:6]:000000 to RPi[14:6]:MCi[5:0]
	101	Maximum modulo size = 128 (0H to 7FH), Modulo body = RPi[14:7]:0000000 to RPi[14:7]:MCi[6:0]
	110	Maximum modulo size = 256 (0H to 0FFH), Modulo body = RPi[14:8]:00000000 to RPi[14:8]:MCi[7:0]
	111	Maximum modulo size = 512 (0H to 1FFH), Modulo body = RPi[14:9]:000000000 to RPi[14:9]:MCi[8:0]

COMPUTATION UNIT

The computation unit contains two main units, the Multiplier and Accumulator Unit (MAU) and Arithmetic Unit (AU).

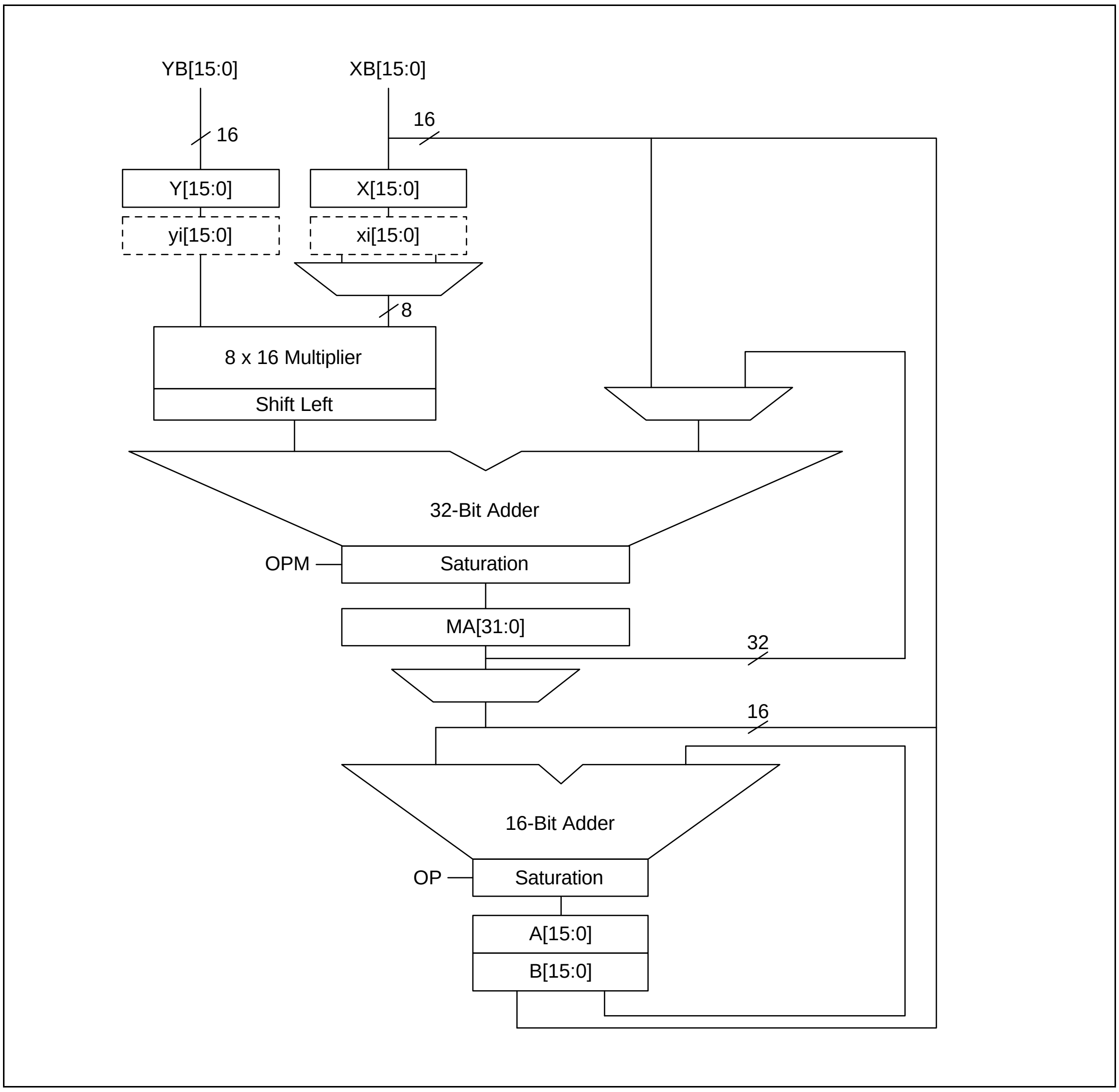


Figure 17-4. Computation Unit Block Diagram

Multiplier and Accumulator Unit (MAU)

The MAU consists of a 8 by 16 to 24 bit parallel multiplier, two 16-bit input registers(X and Y), a product output shifter, and 32-bit product and accumulator register(MA). The multiplier performs signed by signed, signed by unsigned, unsigned by signed, or unsigned by unsigned multiplication. By clearing “MSR1[2] (or M816)”, the MAU can perform 16 by 16 to 32 bit parallel multiplication in 2 cycles. After the multiplier instruction, if a read instruction of MA is followed, previous MA register value will be read out because during a 16x16 multiplication in the second cycle of multiplication, the instruction of MA modification can cause illegal multiplication results. Thus, multiplier instruction should not be followed by MA register writing. The “MV” flag is set if arithmetic overflow occurs after an arithmetic operation in the MA register, and if set “OPM”, the MA register is saturated to a 32-bit positive (7FFFFFFFH) or negative (80000000H). The MA register is not updated by loading X and Y registers. Hence, the X and Y registers can be used as a temporary data registers. The registers in MAU are as shown in the table.

Mnemonics	Description	Reset Value
X	MAU X input register	Unknown
Y	MAU Y input register	Unknown
MAL	MAU Accumulator register, MA[15:0]	Unknown
MAH or MA	MAU Accumulator register, MA[31:16]	Unknown

Arithmetic Unit (AU)

The AU consists of a 16-bit adder, 1-bit shifter, and two result registers (A and B). The AU receives one operand from Ai and another operand from XB or Ai. Operations between the two Ai registers are also possible. The source and destination Ai registers of an AU instruction are always the same. The XB bus is used for transferring one of the register content, an immediate operand, or the content of a data memory location, a source operand. The AU results are stored in one of the Ai registers. The AU can perform add, subtract, compare, and shift operations. It uses two’ s complement arithmetic operations. The AU evaluates the status flags of an arithmetic result. The “V” flag is set if arithmetic overflow occurs after an arithmetic operation in A or B register, and if set to “OPA” or “OPB”, the A or B register is saturated to a 16-bit positive (7FFFH) or negative (8000H). Data transfer between MAC816 and the host processor can be achieved via A or B register. The host processor (CalmRISC) can directly access A and B registers of MAC816 through “CLD GPR,imm” or “CLD imm,GPR” instruction.

STATUS REGISTERS

Status Register 0 : MSR0

MSR0 is mainly reserved for flagging an AU result, for protecting control overflow, and for indicating test results.

Bit Name	Bit	Description
C	0	Carry flag
V	1	Overflow flag
Z	2	Zero flag
N	3	Negative flag
T	4	Test result flag
OPA	5	Overflow Protection control for A register
OPB	6	Overflow Protection control for B register
–	15–7	Reserved

MSR0[0] (or C) is the carry of AU executions. MSR0[1] (or V) is the overflow flag of AU executions. It is set to 1 if and only if the carry-in into the 16-th bit position of addition/subtraction differs from the carry-out from the 16-th bit position. MSR0[2] (or Z) is the zero flag, which is set to 1 if and only if the AU result is zero. MSR0[3] (or N) is the negative flag. Basically, the most significant bit (MSB) of AU results becomes the N flag. However, if an AU instruction touches the overflow flag (V) like ADD, SUB, CP, *etc*, N flag is updated as exclusive-OR of V and the MSB of the AU result. This implies that even if an AU operation results in overflow, N flag is still valid. T flag is set to 1 if the result of "ETST *cond*." Instruction is true. MSR0[5] (or OPA) or MSR0[6] (or OPB) enables arithmetic saturation when an arithmetic overflow occurs in A or B register.

Status Register 1 : MSR1

MSR1 consists of status flags of MAU operation, control bit for MAU, and selection bits of EC[I].

Bit Name	Bit	Description
PSH1	0	Multiplier product 1 bit shift control
OPM	1	Overflow Protection control for MA register
M816	2	Multiplication mode control
MV	3	MA overflow flag
SEC0	7–4	EC[0] selection
SEC1	11–8	EC[1] selection
SEC2	15–12	EC[2] selection

MSR1[0] (or PSH1) enables the product to shift by one bit to the left. MSR1[1] (or OPM) controls MA saturation. MSR1[2] (or M816) selects the operating mode for the multiplier. If M816=1, then the multiplier performs 8 by 16 bit to 24 bit multiplication. Otherwise (M816=0), the multiplier performs 16 by 16 bit to 32 bit multiplication in two cycles. MSR1[3] (or MV) is the overflow flag of MAU executions. It is set to 1 if an arithmetic overflow (32-bit overflow) occurs after an arithm3etic operation in MAU. It is cleared by a processor reset or “ECR MV” and modified by writing to MSR1. SECi selects the combination of EC[I]. The flag information for the host processor is selected by setting SECi.

Value(of SECi)	Description
0000	EC[I] = Z, Set to 1 if Z flag is 1.
0001	EC[I] = not Z
0010	EC[I] = N
0011	EC[I] = not N
0100	EC[I] = C
0101	EC[I] = not C
0110	EC[I] = V
0111	EC[I] = not V
1000	EC[I] = T
1001	EC[I] = GT
1010	EC[I] = LE
1011	EC[I] = MV
1100	EC[I] = not MV
1101–1111	reserved

INSTRUCTION SET

GLOSSARY

This chapter describes the MAC816 instruction set, and the details of each instruction are listed in alphabetical order . The following notations are used for the description and mnemonics of assembler.

Table 17-4. Notation and Convention

Notation	Interpretation
<opN>	Operand N. N can be omitted if there is only one operand. Typically, <op1> is the destination (and source) operand and <op2> is the source operand.
adr:N	Content of memory location specified by N-bit address
#imm:N	N-bit immediate number
&	Bit-wise AND
|	Bit-wise OR
~	Bit-wise NOT
^	Bit-wise XOR
N**M	Mth power of N
$(N)_M$	M-based number N

Table 17-5. MAC816 Registers

Notation	Operand Code	Mnemonic	Descriptions
Mreg	0000–0010	–	Reserved
	0011	MARN	MA[31:16] + MA[15], MA higher word with round-off
	0100	Y	Y[15:0], multiplier Y input register
	0101	X	X[15:0], multiplier X input register
	0110	MAL	MA[15:0], multiplier accumulator lower 16-bits
	0111	MAH	MA[31:16], multiplier accumulator higher 16-bits
	1000	RP0	RP0[15:0], RAM pointer register 0
	1001	RP1	RP0[15:0], RAM pointer register 1
	1010	RP2	RP0[15:0], RAM pointer register 2
	1011	RPD	RAM pointer for short direct addressing
	1100	MC0	Modulo control register 0 for RP0/RP1
	1101	MC1	Modulo control register 1 for RP2
	1110	MSR0	MAC816 status register 0
	1111	MSR1	MAC816 status register 1
Ai	0	A	A[15:0], AU result register A
	1	B	B[15:0], AU result register B
Am	00	A	A[15:0], AU result register A
	01	B	B[15:0], AU result register B
	10	AC	A[15:0], AU result register A with Carry
	11	BC	B[15:0], AU result register B with Carry
MAm	00	A	A[15:0], AU result register A
	01	B	B[15:0], AU result register B
	10	MAL	MA[15:0], multiplier accumulator lower 16-bits
	11	MAH	MA[31:16], multiplier accumulator higher 16-bits

Table 17-5. MAC816 Registers (Continued)

Notation	Operand Code	Mnemonic	Descriptions
Mreg2 Mreg2s Mreg2d	000–011	–	Reserved
	100	Y	Y[15:0], multiplier Y input register
	101	X	X[15:0], multiplier X input register
	110	MAL	MA[15:0], multiplier accumulator lower 16-bits
	111	MAH	MA[31:16], multiplier accumulator higher 16-bits
Mreg1	00	RP0	RP0[15:0], RAM pointer register 0
	01	RP1	RP0[15:0], RAM pointer register 1
	10	RP2	RP0[15:0], RAM pointer register 2
	11	RPD	RAM pointer for short direct addressing
Mreg3	00	MC0	Modulo control register 0 for RP0/RP1
	01	MC1	Modulo control register 1 for RP2
	10	MSR0	MAC816 status register 0
	11	MSR1	MAC816 status register 1

Table 17-6. Data Transfer Registers

Notation	Register Address	Descriptions
Creg	00	A[7:0], AU result register A lower 8-bits
	01	A[15:8], AU result register A higher 8-bits
	10	B[7:0], AU result register B lower 8-bits
	11	B[15:8], AU result register B higher 8-bits

Table 17-7. Memory Access Mode Information

Notation	Operand Code	Mnemonic	Descriptions
@rpm	0000	@rp0+	Content of memory location specified by RP0, RP0 post-increment by 1 with modulo mode
	0001	@rp0-	Content of memory location specified by RP0, RP0 post-decrement by 1 with modulo mode
	0010	@rp0s	Content of memory location specified by RP0, RP0 post-modification by +2 or –2 with modulo mode
	0011	@rp0	Content of memory location specified by RP0
	0100	@rp1+	Content of memory location specified by RP1, RP1 post-increment by 1 with modulo mode
	0101	@rp1-	Content of memory location specified by RP1, RP1 post-decrement by 1 with modulo mode
	0110	@rp1s	Content of memory location specified by RP1, RP1 post-modification by +2 or –2 with modulo mode
	0111	@rp1	Content of memory location specified by RP1
	1000	@rp2+	Content of memory location specified by RP2, RP2 post-increment by 1 with modulo mode
	1001	@rp2-	Content of memory location specified by RP2, RP2 post-decrement by 1 with modulo mode
	1010	@rp2s	Content of memory location specified by RP2, RP2 post-modification by +2 or –2 with modulo mode
	1011	@rp2	Content of memory location specified by RP2
	1100–1111	-	Reserved
@rp0m	00	@rp0+	Content of memory location specified by RP0, RP0 post-increment by 1 with modulo mode
	01	@rp0-	Content of memory location specified by RP0, RP0 post-decrement by 1 with modulo mode
	10	@rp0s	Content of memory location specified by RP0, RP0 post-modification by +2 or –2 with modulo mode
	11	@rp0	Content of memory location specified by RP0
@rp2m	00	@rp2+	Content of memory location specified by RP2, RP2 post-increment by 1 with modulo mode
	01	@rp2-	Content of memory location specified by RP2, RP2 post-decrement by 1 with modulo mode
	10	@rp2s	Content of memory location specified by RP2, RP2 post-modification by +2 or –2 with modulo mode
	11	@rp2	Content of memory location specified by RP2

Table 17-8. Condition Code Information

Notation	Operand Code	Mnemonic	Descriptions
cc	0000	Z	Z = 1
	0001	NZ	Z = 0
	0010	C	C = 1
	0011	NC	C = 0
	0100	NEG	N = 1
	0101	POS	N = 0
	0110	V1	V = 1
	0111	V0	V = 0
	1000	–	Reserved
	1001	GT	N = 0 and Z = 0
	1010	LE	N = 1 and Z = 1
	1011	MV1	MV = 1
	1100	MV0	MV = 0
	1101–1111	–	Reserved

Table 17-9. Control Bit Code Information

Notation	Operand Code	Mnemonic	Descriptions
bs	000	OPM	MSR1[1]
	001	PSH1	MSR1[0]
	010	ME0	RP0[15], RP0 modulo mode enable
	011	ME1	RP1[15], RP1 modulo mode enable
	100	M816	MSR1[2]
	101	ME2	RP2[15], RP2 modulo mode enable
	110	OPA	MSR0[5]
	111	OPB	MSR0[6]

Table 17-10. AU operation code information

Notation	Operand Code	Mnemonic	Descriptions
EMOD0	00	ELD/ELDT	Load
	01	EADD/EADDT	Addition
	10	ESUB/ESUBT	Subtraction
	11	ECP/ECPT	Comparison
EMOD1	0000	ERR/ERRT	Rotate right
	0001	ERL/ERLT	Rotate left
	0010	ESR/ESRT	Arithmetic shift right
	0011	ESL/ESLT	Arithmetic shift left
	0100	EINC/EINCT	Increment
	0101	EDEC/EDECT	Decrement
	0110	ENEG/ENEGT	Negation
	0111	ECR/ECRT	Clear
	1000	ENORM/ENORMT	Normalization
	1001	EABS/EABST	Absolution
	1010–1111	–	reserved

Table 17-11. Others

Notation	Operand Code	Mnemonic	Descriptions
sXsY	00	uu	Unsigned by unsigned multiplication
	01	us	Unsigned by signed multiplication
	10	su	Signed by unsigned multiplication
	11	none	Signed by signed multiplication
rs	0	ER	Reset
	1	ES	Set
ts	0	ELD/ EMOD1/ EMOD0	Execute mnemonic always
	1	ELDT/ EMOD1T/ EMOD0T	Execute mnemonic when test result flag (MSR0[4] or T) is set. If T = 0, act as nop.

INSTRUCTION ENCODING

Table 17-12. Instruction Encoding

Instruction	11	10	9	8	7	6	5	4	3	2	1	0	2^{nd} Word
ELD Mreg2,@rpm	00		00		0	Mreg2			rpm				–
ELD @rpm,Mreg2					1								
ELD Mreg3,@rpm			01		00		Mreg3						
ELD @rpm,Mreg3					01								
ELD Mreg1,adr:4					10		Mreg1		adr[3:0]				
ELD adr:4,Mreg1					11		Mreg1						
ESEC0 #imm:4			10		00		00		Imm[3:0]				
ESEC0 #imm:4							01						
ESEC0 #imm:4							10						
ECR MV							11						
ELD Mreg2d,Mreg2s					01		Mreg2d			Mreg2s			
EMOD0 A,#imm:5					1	EMOD0		Imm[4:0]					
ELD adr:6,MAm			11		adr[5:4]		MAm		adr[3:0]				
ELD MAm,adr:6	01		00				MAm						
EADD Am,adr:6			01				Am						
ESUB Am,adr:6			10										
ECP Am,adr:6			11										
ELD Mreg,Am	10		00		00				Mreg				
ELD Am,Mreg					01								
ELD/ELDT @rpm,Am					1	ts			rpm				
EMOD1/EMOD1T Am			01		0				EMOD1				
EMOD0/EMOD0T Am,MAm					1				MAm		EMOD0		
EMOD0/EMOD0T Am,@rpm			1	EMOD0					rpm				

Table 17-12. Instruction Encoding (Continued)

Instruction	11	10	9	8	7	6	5	4	3	2	1	0	2nd Word
ELD Mreg,#imm:16	11		00		Mreg				Imm[15:12]				Imm[11:0]
EMOD0 Am,#imm:16			01		EMOD0		Am						
EMAD @rp0m,@rp2m,sXsY			10		00		rp0m		rp2m		sXsY		–
EMSB @rp0m,@rp2m,sXsY					01								
EMUL @rp0m,@rp2m,sXsY					10								
EMUL Ai,@rp2m,sXsY					11		0	Ai					
EMUL X,@rp2m,sXsY							10						
EMUL @rp0mY,,sXsY							11		rp0m				
EMAD Ai,@rp2m,sXsY			11		00		0	Ai	rp2m				
EMAD X,@rp2m,sXsY							10						
EMAD @rp0m,Y,sXsY							11		rp0m				
EMSB Ai,@rp2m,sXsY					01		0	Ai	rp2m				
EMSB X,@rp2m,sXsY							10						
EMSB @rp0m,Y,sXsY							11		rp0m				
EMAD X,Y,sXsY					10		00		00				
EMSB X,Y,sXsY									01				
EMUL X,Y,sXsY									10				
ESR MA							01		00		xx		
ESL MA									01				
ERND MA									10				
ENOP							1	xxxxx					
ERPM rpm							00		rpm				
ER/ES bs					11		01		rs	bs			
ETST cc							10		cc				
ELD RPDN,#imm:4							11		Imm[3:0]				

NOTES:

1. "X" means not applicable.
2. There are several codes of EMOD0 or EMOD1, as described in table 8-10.

QUICK REFERENCE

Table 17-13. Quick Reference

Operation	Operand1	Operand2	Function	Flag

ELD
EADD
ESUB
ECP
	A	#imm:5	op1 ← op2
			op1 ← op1 + op2
			op1 ← op1 - op2
			op1 - op2
				–
				c.z,v,n
				c,z,v,n
				c,z,v,n

ELD	RPDN	#imm:4	RPD[7:4] ← op2	
ELD	Adr:6	Am/MAm	op1 ← op2	
ELD	Am/MAm	Adr:6	op1 ← op2	

EADD
ESUB
ECP
	Am	Adr:6	op1 ← op1 + op2
			op1 ← op1 - op2
			op1 - op2
				c.z,v,n
				c,z,v,n
				c,z,v,n

ELD	Mreg1	Adr:4	op1 ← op2	–
ELD	Adr:4	Mreg1	op1 ← op2	–
ELD	Am	Mreg	op1 ← op2	–
ELD	mreg	Am	op1 ← op2	–
ELD	Mreg2d	Mreg2s	op1 ← op2	–
ELD	Mreg2	@rpm	op1 ← op2	–
ELD	@rpm	Mreg2	op1 ← op2	–
ELD EADD ESUB ECP ELDT EADDT ESUBT ECPT	Am	MAm	op1 ← op2 op1 ← op1 + op2 op1 ← op1 - op2 op1 - op2 If T=1, same as ELD If T=1, same as EADD If T=1, same as ESUB If T=1, same as ECP	– c.z,v,n c,z,v,n c,z,v,n – c.z,v,n c,z,v,n c,z,v,n
ELD ELDT	@rpm	Am	op1 ← op2 If T=1, same as ELD	–
ELD EADD ESUB ECP ELDT EADDT ESUBT ECPT	Am	@rpm	op1 ← op2 op1 ← op1 + op2 op1 ← op1 - op2 op1 - op2 If T=1, same as ELD If T=1, same as EADD If T=1, same as ESUB If T=1, same as ECP	– c.z,v,n c,z,v,n c,z,v,n – c.z,v,n c,z,v,n c,z,v,n

Table 17-13. Quick Reference (Continued)

Operation	Operand1	Operand2	Function	Flag
ETST	cc	–	MSR0[4] ← cc (condition check)	–
ELD	mreg	#imm:16	op1 ← op2	–

ELD
EADD
ESUB
ECP
	A	#imm:16	op1 ← op2
			op1 ← op1 + op2
			op1 ← op1 - op2
			op1 - op2
				–
				c.z,v,n
				c,z,v,n
				c,z,v,n

ERPM	rpm	–	RP ← modified RP	–
ER	bs	–	op1 ← 0	–
ES	bs	–	op1 ← 10	–
ESEC0 ESEC1 ESEC2	MSR1	#imm:4	MSR1[7:4] ← imm[3:0] MSR1[11:8] ← imm[3:0] MSR1[15:12] ← imm[3:0]	–
ERR ERRT	Am	–	when Am!=AC/BC, op ← {op1}>>1, op1[15] ← op1[0], c ← op1[0] when Am=AC/BC, op1← {c:op1}>>1, c ← op1[0] when t=1, same as ERR	c,z,v,n c,z,v,n
ERL ERLT	Am	–	when Am!=AC/BC, op←{op1}<<1, op1[0]←op[15], c←op[15], when Am=AC/BC, op1←{op1:c}<<1, c←op[15] when t=1, same as ERL	c,z,v,n c,z,v,n
ESR ESRT	Am	–	when Am!=AC/BC, op ← {op1}>>1, c ← op1[0] when Am=AC/BC, op1 ← {c:op1}>>1, c ← op1[0] when t=1, same as ESR	c,z,v,n c,z,v,n
ESL ESLT	Am	–	when Am!=AC/BC, op1 ← {op1}<<1, op1[0] ← 0, c ← op[15], when Am=AC/BC, op1 ← {op1:c}<<1, c ← op[15] when t=1, same as ESL	c,z,v,n c,z,v,n
EINC EINCT	Am	–	when Am!=AC/BC, op1 ← op1+1 when Am=AC/BC, op1 ← op1+c when t=1, same as EINC	c,z,v,n c,z,v,n
EDEC EDECT	Am	–	when Am!=AC/BC, op1 ← op1+ffffh when Am=AC/BC, op1 ← op1+ffffh+c when t=1, same as EDEC	c,z,v,n c,z,v,n
ENEG ENEGT	Am	–	when Am!=AC/BC, op1 ← ~op1+1 when Am=AC/BC, op1 ← ~op1+c when t=1, same as ENEG	c,z,v,n c,z,v,n
EABS EABST	Am	–	when Am!=AC/BC, if op[15]=1, op1 ← ~op1+1 when Am=AC/BC, op[15]=1, op1 ← ~op1+c when t=1, same as EABS	c,z,v,n c,z,v,n
ENORM ENORMT	Am	–	when Am!=AC/BC, if op1[15]^op1[14]=0, op1 ← {op1}<<1, op1[0] ← 0, RP0 ← RP0+1 when Am=AC/BC, if op1[15]^op1[14]=0, op1 ← {op1:c}<<1, RP0 ← RP0+1 when t=1, same as ENORMT	c,z,v,n c,z,v,n
ECR ECRT	Am	–	op1 ← 0 when t=1, same as ECR	–

Table 17-13. Quick Reference (Concluded)

Operation	Operand1	Operand2	Operand3	Function	Flag
ESR	MA	–	–	op1 ← op1>>1	–
ESL	MA	–	–	op1 ← op1<<1	MV
ERND	MA	–	–	MA[31:16] ← MA[31:16] + MA[15]	MV
EMAD	MA	@rp0m	@rp2m	X-reg ← @rp0m, Y-reg ← @rp2m, MA ← MA+X*Y	MV
EMSB	MA	@rp0m	@rp2m	X-reg ← @rp0m, Y-reg ← @rp2m, MA ← MA-X*Y	MV
EMUL	MA	@rp0m	@rp2m	X-reg ← @rp0m, Y-reg ← @rp2m, MA ← (X*Y)	–
EMAD	MA	Ai	@rp2m	X-reg ← op2, Y-reg ← @rp2m, MA ← MA+X*Y	MV
EMSB	MA	Ai	@rp2m	X-reg ← op2, Y-reg ← @rp2m, MA ← MA-X*Y	MV
EMUL	MA	Ai	@rp2m	X-reg ← op2, Y-reg ← @rp2m, MA ← (X*Y)	–
EMAD	MA	X	@rp2m	Y-reg ← @rp2m, MA ← MA+X*Y	MV
EMSB	MA	X	@rp2m	Y-reg ← @rp2m, MA ← MA-X*Y	MV
EMUL	MA	X	@rp2m	Y-reg ← @rp2m, MA ← (X*Y)	–
EMAD	MA	@rp0m	Y	X-reg ← @rp0m, MA ← MA+X*Y	MV
EMSB	MA	@rp0m	Y	X-reg ← @rp0m, MA ← MA-X*Y	MV
EMUL	MA	@rp0m	Y	X-reg ← @rp0m, MA ← (X*Y)	–
EMAD	MA	X	Y	MA ← MA+X*Y	MV
EMSB	MA	X	Y	MA ← MA-X*Y	MV
EMUL	MA	X	Y	MA ← (X*Y)	–

MAC816 INSTRUCTION DESCRIPTION

EABS —Absolute

Format: EABS <op>
<op>: Am

Operation: If the MSB of <op> is 1, <op> ← ~<op> +1 when <op> is A or B.
If the MSB of <op> is 1, <op> ← ~<op> +C when <op> is AC or BC.
EABS adds the values 0 and the 2' s complement of <op>.

Flags:
C: set if the borrow of result is zero. Reset if not.
Z: set if result is zero. Reset if not.
V: set if overflow is generated. Reset if not.
N: exclusive OR of V and MSB of result.

EABST —Absolute conditional

Format:	EABST <op> <op>: Am
Operation:	If T=1, then same as EABS, else no operation
Flags:	If T=1, then same as EABS, else no operation

EADD —Add

Format: EADD <op1>, <op2>
<op1>: Am: A, B, AC, BC
<op2>: adr:6, @rpm, Ai, Mreg, #imm:16, #imm:5

Operation: <op1> ← <op1 + <op2> when <op1> is A or B.
<op1> ← <op1 + <op2> + C when <op1> is AC or BC.
EADD adds the values in <op1> and <op2> and stores the result in <op1>.

Flags:
C: set if the carry of result is 1. Reset if not.
Z: set if result is zero. Reset if not.
V: set if overflow is generated. Reset if not.
N: exclusive OR of V and MSB of result.

NOTE: If <op1> is B, <op2> can not be #imm:5.

EADDT —Add conditional

Format: EADDT <op1>, <op2>
<op1>: Am: A, B, AC, BC
<op2>: @rpm, Ai, MAH,MAL

Operation: If T=1, then same as EADD, else no operation

Flags: If T=1, then same as EADD, else no operation

ECP —Compare

Format: ECP <op1>, <op2>
<op1>: Am
<op2>: adr:6, @rpm, Ai, Mreg, #imm:16, #imm:5

Operation: <op1> + ~<op2> +1 when <op1> is A or B.
<op1> + ~<op2> +C when <op1> is AC or BC.
ECP compares the values of <op1> and <op2> by subtracting <op2> from <op1>.
Contents of <op1> and <op2> are not changed.

Flags: **C:** set if the borrow of result is zero. Reset if not.
Z: set if result is zero. Reset if not.
V: set if overflow is generated. Reset if not.
N: exclusive OR of V and MSB of result.

NOTE: If <op1> is B, <op2> can not be #imm:5.

ECPT — Compare conditional

Format: ECPT <op1>, <op2>
<op1>: Am: A, B, AC, BC
<op2>: @rpm, Ai, MAH,MAL

Operation: If T=1, then same as ECP, else no operation

Flags: If T=1, then same as ECP, else no operation

ECR —Clear

Format: ECRT <op>
<op>: Ai, MV

Operation: <op> ← 0
ECRT clears Ai or MV.

ECRT —Clear

Format: ECRT <op>
<op>: Ai

Operation: If T=1, <op> ← 0
ECRT clears Ai when T=1.

EDEC —Decrement

Format: EDEC <op>
<op>: Am

Operation: <op> ← <op> + 0xffff when <op> is A or B.
<op> ← <op> + 0xffff + C when <op> is AC or BC.
EDEC decrements the value in <op>.

Flags:
C: set if carry is generated. Reset if not.
Z: set if result is zero. Reset if not.
V: set if overflow is generated. Reset if not.
N: exclusive OR of V and MSB of result.

EDECT —Decrement conditional

Format: EDECT <op>
<op>: Am

Operation: If T=1, then same as EDEC, else no operation

Flags: If T=1, then same as EDEC, else no operation

EINC —Increment

Format: EINC <op>
<op>: Am

Operation: <op> ← <op> + 1 when <op> is A or B.
<op> ← <op> + C when <op> is AC or BC.
EINC increments the value in <op>.

Flags:
C: set if carry is generated. Reset if not.
Z: set if result is zero. Reset if not.
V: set if overflow is generated. Reset if not.
N: exclusive OR of V and MSB of result.

EINCT —Increment conditional

Format: EINCT <op>
<op>: Am

Operation: If T=1, then same as EINC, else no operation

Flags: If T=1, then same as EINC, else no operation

ELD Adr —Load Adr

Format: ELD <op1>, <op2>
<op1>,<op2>: adr:6, MAi / adr:4,Mreg1

Operation: <op1> ← <op2>
ELD Adr loads a value specified by <op2> into the memory location determined by <op1>

ELD Ai —Load Ai

Format: ELD <op1>, <op2>
<op1>: Ai: A, B
<op2>: adr:6, @rpm, Ai, Mreg, #imm:5, #imm:16

Operation: Ai ← <op2>
ELD Ai loads a value specified by <op2> into the register designated by Ai.

NOTE: If <op1> is B, <op2> can not be #imm:5.

ELD Mreg —Load Mreg

Format: ELD <op1>, <op2>
<op1>: Mreg
<op2>: Ai

Operation: Mreg ← Ai
ELD Mreg loads a value specified by <op2> into the register designated by Mreg.

ELD Mreg1 —Load Mreg1

Format: ELD <op1>, <op2>
<op1>: Mreg1: RP0, RP1, RP2, RPD
<op2>: adr:4

Operation: Mreg1 ← adr:4
ELD Mreg1 loads the content of memory location determined by adr:4 into the register designated by Mreg1.

ELD Mreg2 —Load Mreg2

Format: ELD <op1>, <op2>
<op1>: Mreg2: X, Y, MAH, MAL
<op2>: @rpm

Operation: Mreg2 ← @rpm, rpi ← post-modified rpi
ELD Mreg2 loads the content of memory location determined by @rpm into the register designated by Mreg2.

ELD Mreg3 —Load Mreg3

Format: ELD <op1>, <op2>
<op1>: Mreg3: MC0, MC1, MSR0, MSR1
<op2>: @rpm

Operation: Mreg3 ← @rpm
ELD Mreg3 loads the content of memory location determined by @rpm into the register designated by Mreg3.

ELD @rpm —Load into memory indexed

Format: ELD <op1>, <op2>
<op1>: @rpm
<op2>: Ai, Mreg2, Mreg3

Operation: @rpm ← <op2>, rpi ← post-modified rpi
ELD @rpm loads the value of <op2> into the memory location determined by @rpm.

EMAD —Multiplication and Addition

Format: EMAD <op1>, <op2>,sXsY
<op1>,<op2>: @rp0m,@rp2m / Ai,@rp2m / X,@rp2m / @rp0m,Y / X,Y

Operation: X ← <op1>, Y ← <op2>, MA ← MA + {sign,X}*{sign,Y}
EMAD multiplies the values in <op1> and <op2> and adds the result to MA in MA.

Flags: **MV:** Set if the arithmetic overflow occurs in MA after this instruction.

EMSB —Multiplication and Subtraction

Format: EMSB <op1>, <op2>,sXsY
<op1>,<op2>: @rp0m,@rp2m / Ai,@rp2m / X,@rp2m / @rp0m,Y / X,Y

Operation: X ← <op1>, Y ← <op2>, MA ← MA –{sign,X}*{sign,Y}
EMAD multiplies the values in <op1> and <op2> together and subtracts the result from MA and stores the result in MA.

Flags: **MV:** Set if the arithmetic overflow occurs in MA after this instruction.

EMUL —Multiply

Format: EMUL <op1>, <op2>,sXsY
<op1>,<op2>: @rp0m,@rp2m / Ai,@rp2m / X,@rp2m / @rp0m,Y / X,Y

Operation: X ← <op1>, Y ← <op2>, MA ← {sign,X}*{sign,Y}
EMUL multiplies the values in <op1> and <op2> and stores the result in MA.

ENEG —Negate

Format: ENEG <op>
<op>: Am

Operation: <op> ← ~<op> +1 when <op> is A or B.
<op> ← ~<op> +C when <op> is AC or BC.
ESUB adds the values 0 and the 2’ s complement of <op> to negate <op>.

Flags:
C: set if the borrow of result is zero. Reset if not.
Z: set if result is zero. Reset if not.
V: set if overflow is generated. Reset if not.
N: exclusive OR of V and MSB of result.

ENEGT —Negate conditional

Format: ENEGT <op>
<op>: Am

Operation: If T=1, then same as ENEG, else no operation

Flags: If T=1, then same as ENEG, else no operation

ENOP —No operation

Format: ENOP

Operation: No operation

Flags: No operation

ENORM —Normalization step

Format: ENORM <op>
<op>: Am

Operation: If <op>[15] == <op>[14], <op> ← <op> << 1, RP0 ← RP0+1 when <op> is A or B.
If <op>[15] == <op>[14], <op> ← {<op>,C} <<1, RP0 ← RP0+1 when <op> is AC or BC.

Flags:
C: <op>[15] ^ <op>[14]
Z: set if result is zero. Reset if not.
V: reset to zero.
N: set if the MSB of result is 1. Reset if not.

ENORMT —Normalization step conditional

Format: ENORMT <op>
<op>: Am

Operation: If T=1, then same as ENORM, else no operation

Flags: If T=1, then same as ENORM, else no operation

ER —Bit Reset

Format: ER bs

Operation: bs ← 0
ES resets the specified bit.

ERL —Rotate Left

Format: ERL <op>
<op>: Am

Operation: <op> ← {<op>[14:0],<op>[15]}, C ← <op>[15] when Am is A or B.
<op> ← {<op>[14:0],C}, C ← <op>[15] when Am is AC or BC.
ERL rotates the value of <op> to the left and stores the result back into <op>.
The original MSB of <op> is copied into carry (C).

Flags:
C: set if the MSB of <op> (before shifting) is 1. Reset if not.
Z: set if result is zero. Reset if not.
V: reset to zero.
N: set if the MSB of result is 1. Reset if not.

ERLT —Rotate Left conditional

Format: ERLT <op>
<op>: Am

Operation: If T=1, then same as ERL, else no operation

Flags: If T=1, then same as ERL, else no operation

ERND —Round off

Format: ERND MA

Operation: MA[31:16] ← MA[31:16] + MA[15], MA[15:0] ← 0
ERND adds 0x8000 to the lower 16-bit position of MA and stores the result in MA.

Flags: **MV:** set if overflow is generated. Reset if not.

ERPM —Modify Ram pointer

Format: ERPM rpm

Operation: rpi ← modified rpi
ERPM modifies a rpi by rpm.

NOTE: It does not generate a cycle of RAM access.

ERR —Rotate Right

Format: ERR <op>
<op>: Am

Operation: <op> ← {<op>[0], <op>[15:1]}, C ← <op>[0] when Am is A or B.
<op> ← {C, <op>[15:1]}, C ← <op>[0] when Am is AC or BC.
RR rotates the value of <op> to the right and stores the result back into <op>.
The original LSB of <op> is copied into carry (C).

Flags:
C: set if the LSB of <op>(before shifting) is 1. Reset if not.
Z: set if result is zero. Reset if not.
V: reset to zero.
N: set if the MSB of result is 1. Reset if not.

ERRT —Rotate Right conditional

Format: ERRT <op>
<op>: Am

Operation: If T=1, then same as ERR, else no operation

Flags: If T=1, then same as ERR, else no operation

ES —Bit Set

Format: ES bs

Operation: bs ← 1
ES sets the specified bit.

ESEC0 / ESEC1 / ESEC2 —Set SECi

Format:

ESEC0 #imm:4
ESEC1 #imm:4
ESEC2 #imm:4

Operation:

ESEC0: SEC0[3:0] ← #imm:4
ESEC1: SEC1[3:0] ← #imm:4
ESEC2: SEC2[3:0] ← #imm:4

ESL —Shift Left

Format: ESL <op>
<op>:Am

Operation: <op> ← {<op>[14:0],0}, C ← <op>[15] when <op> is A or B.
<op> ← {<op>[14:0],C}, C ← <op>[15] when <op> is AC or BC.
ESL shifts to the left by 1 bit. The MSB of the original <op> is copied into carry(C).

Flags: **C:** set if the MSB of <op>(before shifting) is 1. Reset if not.
Z: set if result is zero. Reset if not.
V: set if overflow is generated. Reset if not.
N: exclusive OR of V and MSB of result.

ESLT —Shift Left conditional

Format: ESLT <op>
<op>: Am

Operation: If T=1, then same as ESL, else no operation

Flags: If T=1, then same as ESL, else no operation

ESR —Shift Right

Format: ESR <op>
<op>:Am

Operation: <op> ← {<op>[15],<op>[15:1]}, C ← <op>[0] when <op> is A or B.
<op> ← {C,<op>[15:1]}, C ← <op>[0] when <op> is AC or BC.
ESR shifts to the right by 1 bit. The LSB of the original <op> is copied into carry(C).

Flags:
C: set if the LSB of <op>(before shifting) is 1. Reset if not.
Z: set if result is zero. Reset if not.
V: set to zero
N: set if result is negative. Reset if not.

ESRT — Shift Right conditional

Format: ESRT <op>
<op>: Am

Operation: If T=1, then same as ESR, else no operation

Flags: If T=1, then same as ESR, else no operation

ESUB —Subtract

Format: ESUB <op1>, <op2>
<op1>: Am
<op2>: adr:6, @rpm, Ai, Mreg, #imm:16, #imm:5

Operation: <op1> ← <op1> + ~<op2> +1 when <op1> is A or B.
<op1> ← <op1> + ~<op2> +C when <op1> is AC or BC.
ESUB adds the values in <op1> and the 2' s complement of <op2> to perform subtraction on <op1> and <op2>.

Flags: **C:** set if the borrow of result is zero. Reset if not.
Z: set if result is zero. Reset if not.
V: set if overflow is generated. Reset if not.
N: exclusive OR of V and MSB of result.

NOTE: If <op1> is B, <op2> can not be #imm:5.

ESUBT —Subtract conditional

Format: ESUBT <op1>, <op2>
<op1>: Am
<op2>: @rpm, Ai, MAH, MAL

Operation: If T=1, then same as ESUB, else no operation

Flags: If T=1, then same as ESUB, else no operation

ETST —Test Condition

Format: ETST cc
cc: Z, NZ, C, NC, NEG, POS, V1, V0, GT, LE, MV1, MV0

Operation: T ← test result
ETST tests the specified condition of a flag.

Flags: **T:** set if test result is true. Reset if not.

18 ELECTRICAL DATA

OVERVIEW

Table 18-1. Absolute Maximum Ratings

(T_A = 25°C)

Parameter	Symbol	Conditions	Rating	Unit
Supply voltage	V_{DD}	–	– 0.3 to + 6.0	V
Input voltage	V_I	–	– 0.3 to V_{DD} + 0.3	
Output voltage	V_O	–	– 0.3 to V_{DD} + 0.3	
Output current high	I_{OH}	One I/O pin active	– 18	mA
		All I/O pins active	– 60	
Output current low	I_{OL}	One I/O pin active	+ 30	
		Total pin current for ports 1, 2, 3	+ 100	
Operating temperature	T_A	–	– 40 to + 85	°C
Storage temperature	T_{STG}	–	– 65 to + 150	

Table 18-2. D.C. Electrical Characteristics

(T_A = – 40°C to + 85°C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Operating Voltage (HSX mode)	V_{DD}	F_{CPU} = 20 MHz	4.5	–	5.5	V
		F_{CPU} = 3 MHz	1.8		5.5	
Operating Voltage (MSX mode)	V_{DD}	F_{CPU} = 10 MHz	4.5	–	5.5	
		F_{CPU} = 3 MHz	1.8		5.5	

(T_A = 40° °C, V = 1.8 V to 5.5 V)

Parameter		Conditions	Min		Max	Unit
	V_{IH1}	IH2	0.8 V	–	DD	V
	IH2	X	V_{DD}- 0.1			
Input low voltage	V_{IL1}	All input pins except V_{IL2}	–	–	0.2 V_{DD}	
	V_{IL2}	X_{IN}			0.1	
Output high voltage	V_{OH1}	V_{DD} = 5V; I_{OH} = -1 mA All output pins	V_{DD}-1.0	–	–	V
Output low voltage	V_{OL1}	V_{DD} = 5V; I_{OL} = 8 mA All output pins except V_{OL2}	–		2	
	V_{OL2}	V_{DD} = 5V; I_{OL} = 15 mA, Port 1			2	
Input high leakage current	I_{LIH1}	V_{IN} = V_{DD} All input pins except I_{LIH2}	–	–	3	uA
	I_{LIH2}	V_{IN} = V_{DD} X_{IN}, XT_{IN}			20	
Input low leakage current	I_{LIL1}	V_{IN} = 0 V All input pins except I_{LIL2}	–	–	-3	
	I_{LIL2}	V_{IN} = 0 V X_{IN}, XT_{IN}, RESET			-20	
Output high leakage current	I_{LOH}	V_{OUT} = V_{DD} All I/O pins and Output pins	–	–	3	uA
Output low leakage current	I_{LOL}	V_{OUT} = 0 V All I/O pins and Output pins	–	–	-3	
Oscillator feed back resistors	R_{osc1} (HSX)	V_{DD} = 5.0 V, T_A = 25°C, X_{IN} = V_{DD}, X_{OUT} = 0V	510	710	910	kΩ
	R_{osc2} (MSX)	V_{DD} = 5.0 V,T_A = 25°C, X_{IN} = V_{DD}, X_{OUT} = 0V	510	710	910	
	R_{osc3} (LSX)	V_{DD} = 5.0 V, T_A = 25°C, X_{IN} = V_{DD}, X_{OUT} = 0V	2.0	2.7	3.5	MΩ
Pull-up resistor	R_{L1}	V_{IN} = 0 V; V_{DD} = 5 V ± 10% Ports 0,1,2,3,4,5 T_A=25°C	30	50	70	kΩ
	R_{L2}	V_{IN} = 0 V; V_{DD} = 5 V ± 10% T_A=25°C, RESET only	110	210	310	

Table 18-2. D.C. Electrical Characteristics (Continued)

(T_A = –40°C to + 85°C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Supply current [1]	I_{DD1}[2]	Operating mode: V_{DD} = 5 V ± 10% **20 MHz** crystal oscillator(HSX)	–	10	20	mA
		5 MHz crystal oscillator(MSX)		4	8	
		V_{DD} = 3 V ± 10% **5 MHz** crystal oscillator(MSX)		2	4	
	I_{DD2}[3]	Idle mode: V_{DD} = 5 V ± 10% **20 MHz** crystal oscillator(HSX)	–	2.5	5	mA
		5 MHz crystal oscillator(MSX)		1	2	
		V_{DD} = 3 V± 10% **5 MHz** crystal oscillator(MSX)		0.4	0.8	
	I_{DD3}	Stop mode V_{DD} = 5 V ± 10%	–	0.5	3	uA
		V_{DD} = 3 V ± 10%		0.2	1.2	

NOTES:

1. Supply current does not include current drawn through internal pull-up resistors or external output current loads.
2. In operating current test mode Timer A and Timer B are running.
3. In idle current test mode the Watch timer is running.
4. The operating and idle currents are measured at weak mode.

Table 18-3. A. C. Electrical Characteristics

(T_A = –40°C to + 85°C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Interrupt input high, low width	t_{INTH}, t_{INTL}	P2.0 - P2.3, P3.0 - P3.1 V_{DD} = 5V	200	–	–	ns
RESET input low width	t_{RSL}	V_{DD} = 5V ± 10%	1	–	–	us

NOTE: User must keep a value larger than the min value.

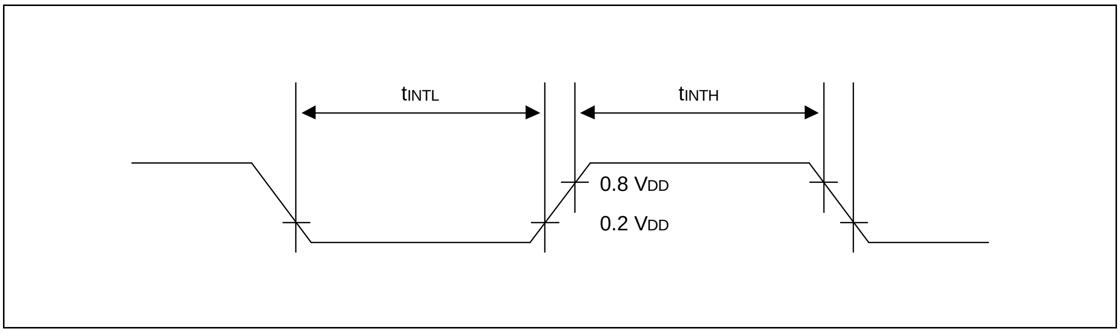


Figure 18-1. Input Timing for External Interrupts

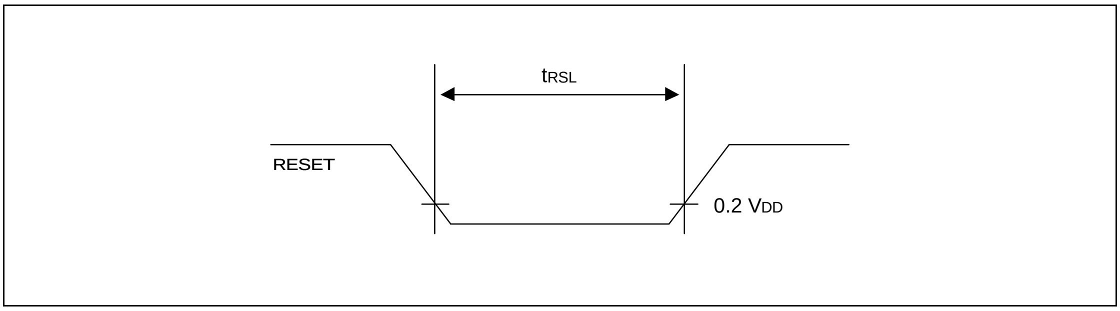


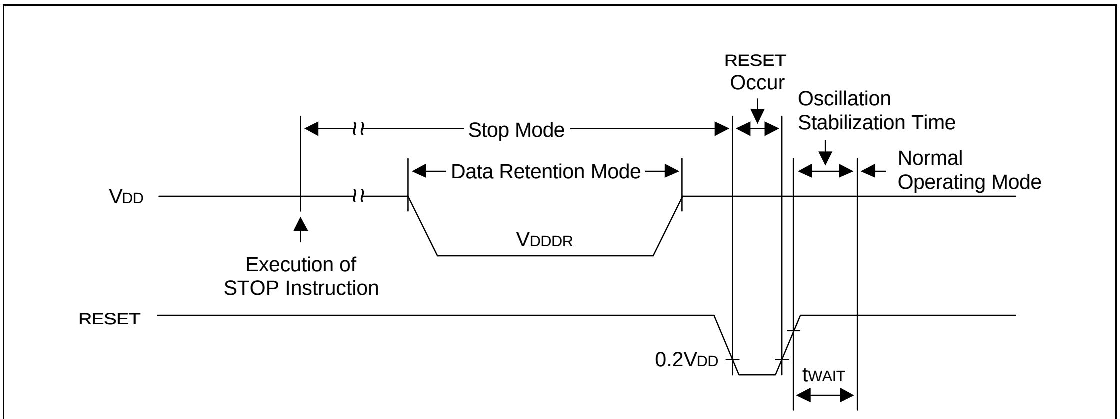
Figure 18-2. Input Timing for RESET

Table 18-4. Data Retention Supply Voltage in Stop Mode

(T_A = –40°C to + 85°C, V_{DD} = 1.8 V to 5.5V)

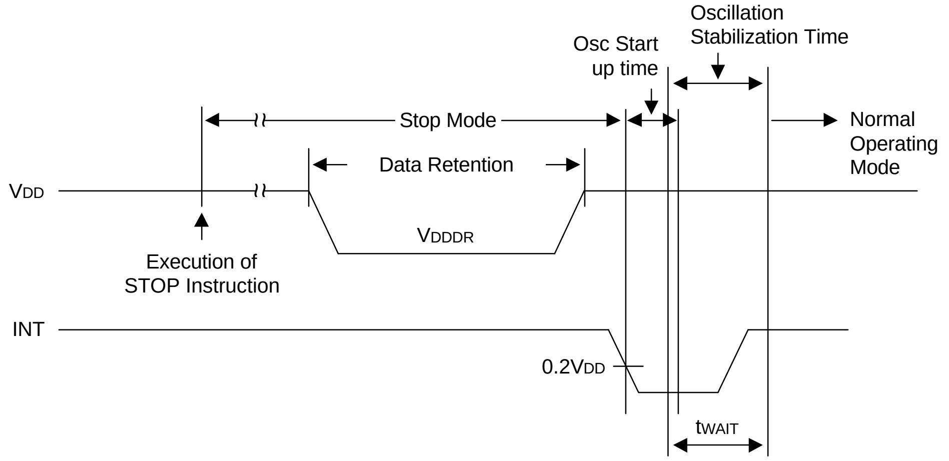
Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data retention supply voltage	V_{DDDR}		1.5	–	5.5	V
Data retention supply current	I_{DDDR}	V_{DDDR} = 1.5V	–	–	2	µA

NOTE: Supply current does not include current drawn through internal pull-up resistors or external output current loads.



NOTE: t_{WAIT} is the same as 2048 x 32 x 1/fxx

Figure 18-3. Stop Mode Release Timing When Initiated by a RESET



NOTE: t_{WAIT} is the same as 2048 x 32 x 1/fxx. The value of 2048 which is selected for the clock source of the basic timer counter can be changed. Then the value of t_{WAIT} will be changed and ,when you select 16 instead of 32, the value of twait will also be changed.

Figure 18-4. Stop Mode Release Timing When Initiated by Interrupts

Table 18-5. Synchronous SIO Electrical Characteristics

(T_A = –40°C to + 85°C V_{DD} = 4.5 V to 5.5 V, V_{SS} = 0 V, fxx = 10 MHz oscillator)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
SCK Cycle time	t_{CYC}	–	200	–	–	ns
Serial Clock High Width	t_{SCKH}	–	60	–	–	
Serial Clock Low Width	t_{SCKL}	–	60	–	–	
Serial Output data delay time	t_{OD}	–	–	–	50	
Serial Input data setup time	t_{ID}	–	40	–	–	
Serial Input data Hold time	t_{IH}	–	100	–	–	

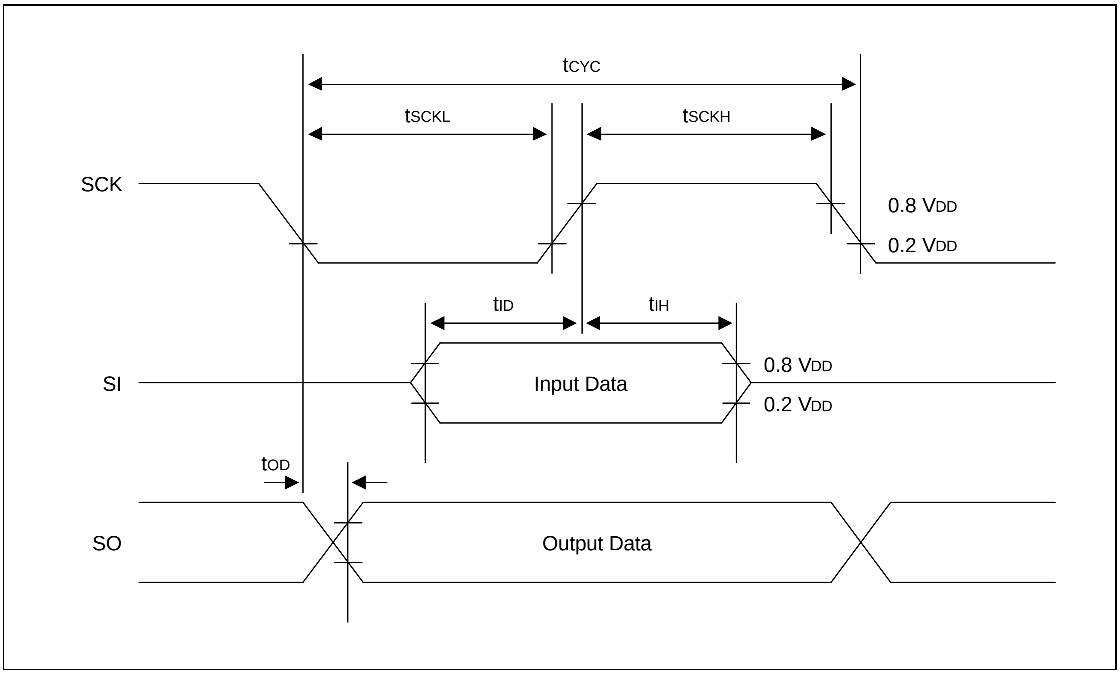
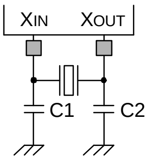
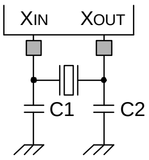
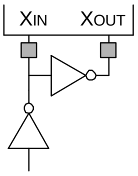


Figure 18-5. Serial Data Transfer Timing

Table 18-6. Main Oscillator Frequency

(T_A = -40°C + 85°C, V_{DD} = 1.8 V to 5.5 V)

Oscillator	Clock Circuit	Test Condition	Min	Typ	Max	Unit
Crystal		LSX mode	32	32.768	35	kHz
		MSX mode	0.4	–	10	MHz
		HSX mode	0.4	–	20	
Ceramic		LSX mode	32	32.768	35	kHz
		MSX mode	0.4	–	10	MHz
		HSX mode	0.4	–	20	
External clock		LSX mode	32	32.768	35	kHz
		MSX mode	0.4	–	10	MHz
		HSX mode	0.4	–	20	
RC		r = 22Kohm, V_{DD} = 5 V Direct soldering	1.4	2	2.6	MHz

NOTES:

1. Keep the wiring length as short as possible.
2. Do not cross the wiring with the other signal lines.
3. Do not route the wiring near a signal line through which a high fluctuating current flows.
4. Always make the ground point of the oscillator capacitor the same potential as VSS.
5. Do not ground the capacitor to a ground pattern through which a high current flows.
6. Do not fetch signals from the oscillator.

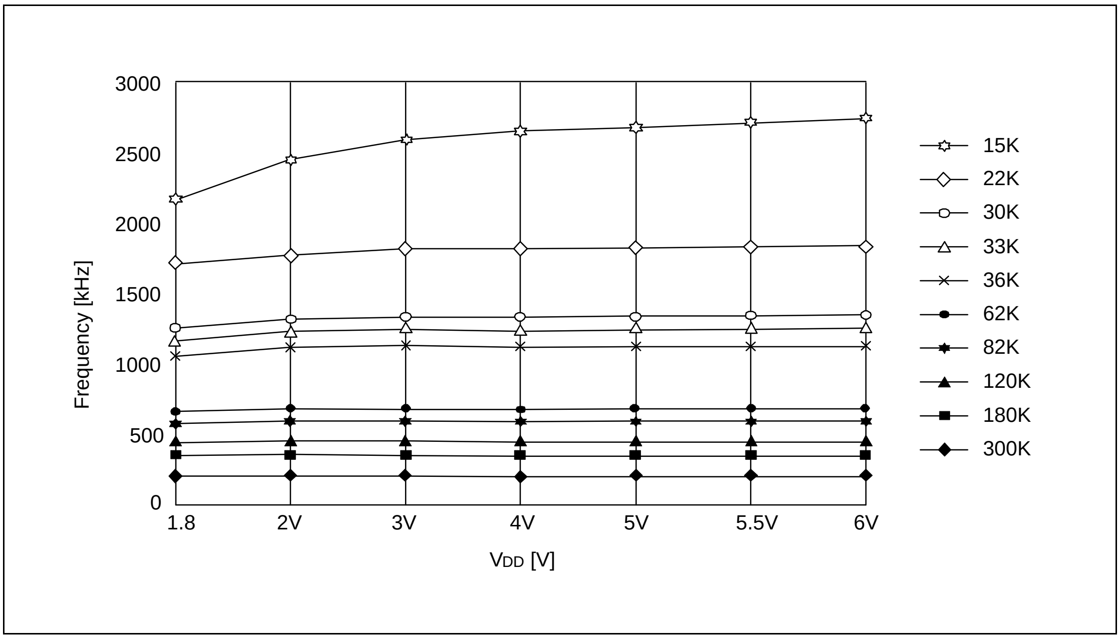


Figure 18-6. RC Oscillator Characteristic Curve

Table 18-7. Main Oscillator Oscillation Stabilization Time (t_{ST1})

(T_A = -40°C + 85°C, V_{DD} = 4.5 V to 5.5 V)

Oscillator		Test Condition(Normal mode)	Min	Typ	Max	Unit
HSX	Crystal	V_{DD} = minimum oscillation voltage range.	–	–	10	ms
	Ceramic		–	–	4	ms
	External clock	X_{IN} input high and low level width (t_{XH}, t_{XL})	50	–	–	ns
MSX	Crystal	V_{DD} = minimum oscillation voltage range.	–	–	100	ms
	Ceramic		–	–	50	ms
	External clock	X_{IN} input high and low level width (t_{XH}, t_{XL})	50	–	–	ns
LSX	32768Hz Crystal	V_{DD} = minimum oscillation voltage range.	–	200	500	ms

NOTE: Oscillation stabilization time (t_{ST1}) is the time that is required to stabilize oscillation after a reset or STOP mode release.

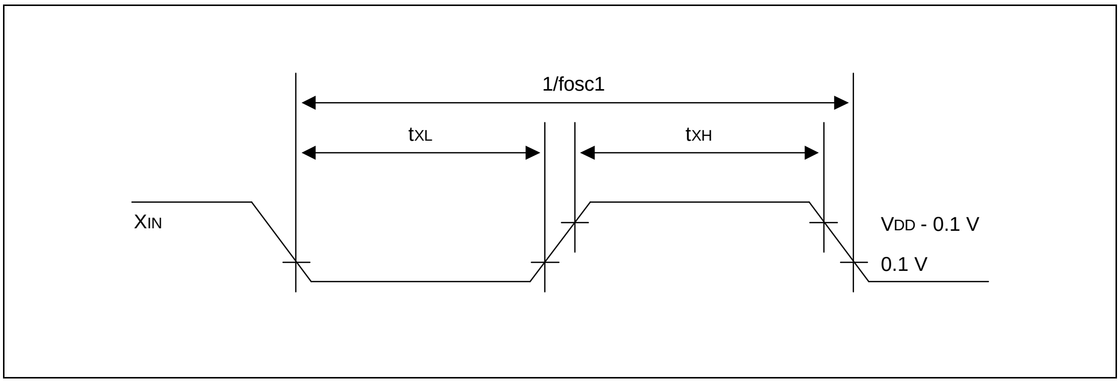
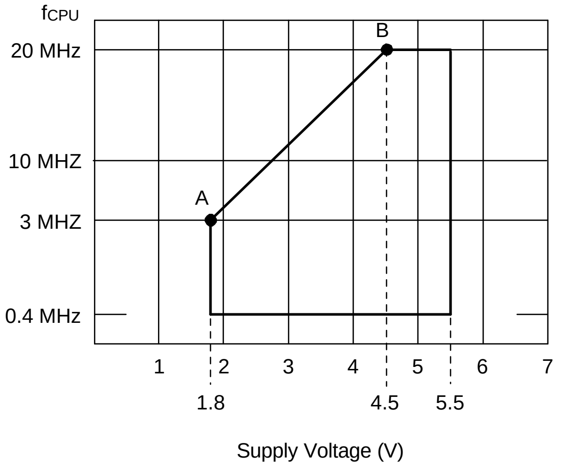
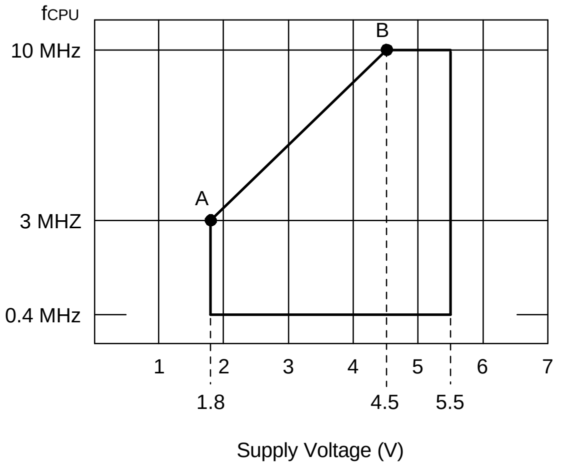


Figure 18-7. Clock Timing Measurement at X_{IN}



Minimum instruction clock = 1/1 x oscillator frequency

Figure 18-8. HSX Mode Operating Voltage Range



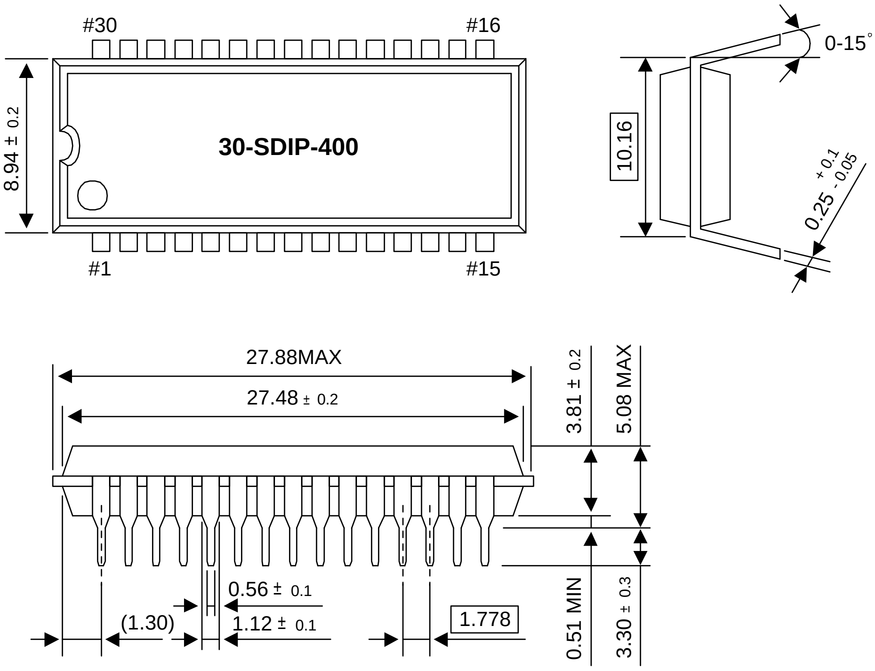
Minimum instruction clock = 1/1 x oscillator frequency

Figure 18-9. MSX Mode Operating Voltage Range

19 MECHANICAL DATA

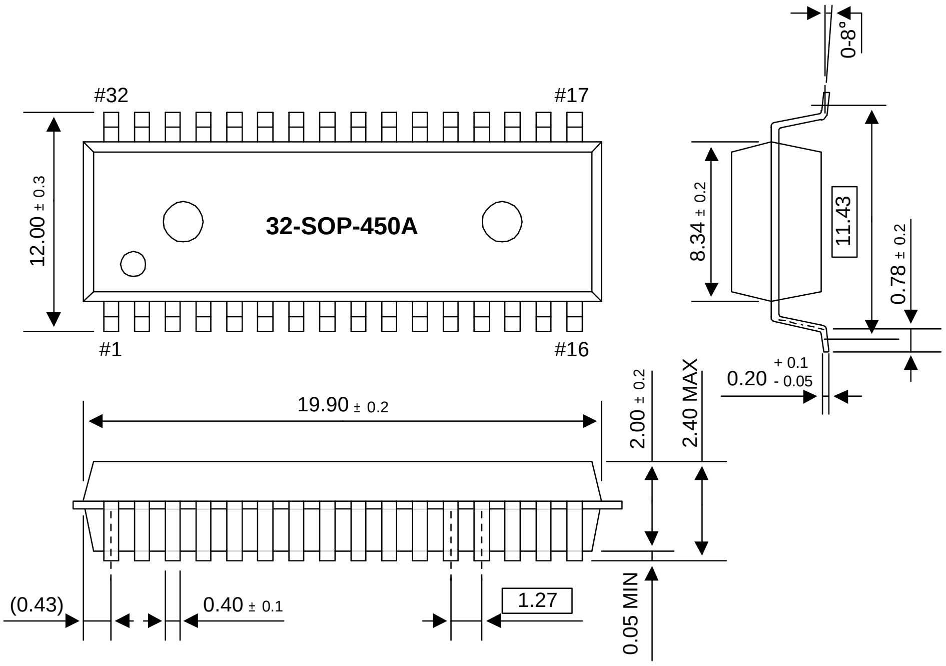
OVERVIEW

The S3CB018/FB018 is available in a 30-pin SDIP package (Samsung: 30-SDIP-400) and a 32-pin SOP package (32-SOP-450A). Package dimensions are shown in Figures 20-1 and 20-2.



NOTE: Dimensions are in millimeters.

Figure 19-1. 30-Pin SDIP Package Dimensions



NOTE: Dimensions are in millimeters

Figure 19-2. 32-SOP-450A Package Dimensions

20 S3FB018 FLASH MCU

OVERVIEW

The S3FB018 single-chip CMOS microcontroller is the FLASH version of the S3CB018 microcontroller.
It has an on-chip FLASH ROM instead of masked ROM. The FLASH ROM is accessed in serial data format.

The S3FB018 is fully compatible with the S3CB018, both in function and in pin configuration. Because of its simple programming requirements, the S3FB018 is ideal for use as an evaluation chip for the S3CB018.

PIN ASSIGNMENTS

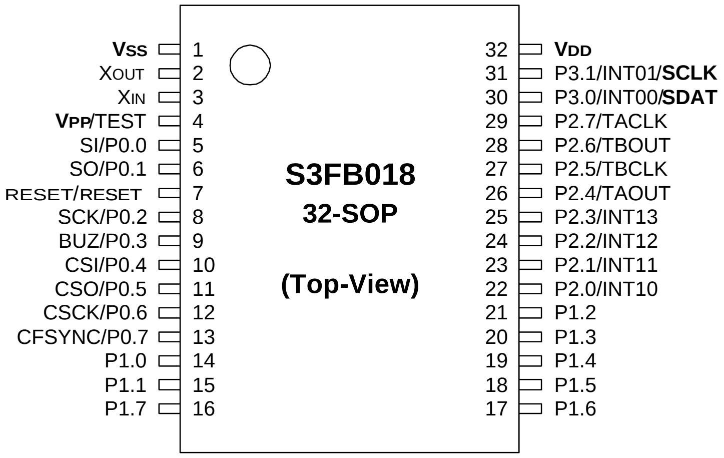


Figure 20-1. 32-SOP Pin Assignment

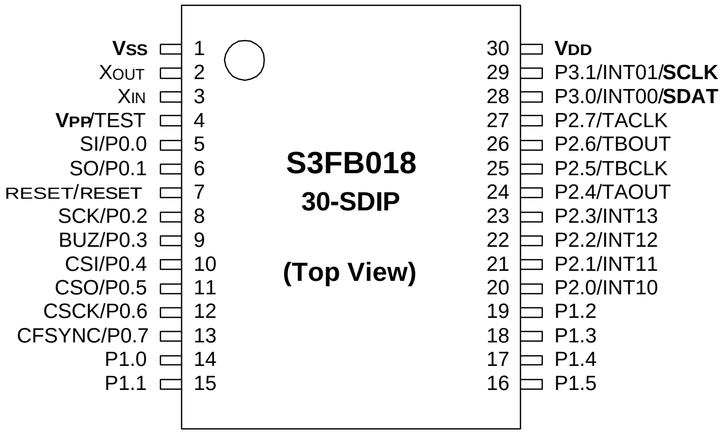


Figure 20-2. 30-SDIP Pin Assignment

Table 20-1. Descriptions of Pins Used to Read/Write the FLASH ROM

Main Chip	During Programming			
Pin Name	Pin Name	Pin No.	I/O	Function
P3.0	SDAT	30(28)	I/O	Serial data pin. Output port when reading and input port when writing. Can be assigned as a Input/push-pull output port.
P3.1	SCLK	31(29)	I/O	Serial clock pin. Input only pin.
TEST	Vpp (TEST)	4	I	Power supply pin for FLASH ROM cell writing (indicates that FLASH enters into the writing mode). When 12.5 V is applied, FLASH is in writing mode and, when 5 V is applied, FLASH is in the reading mode. When FLASH is operating , hold GND.
RESET	RESET	7	I	Chip Initialization
V_{DD}/V_{SS}	V_{DD}/V_{SS}	32/1(30/1)	–	Logic power supply pin. V_{DD} should be tied to +5 V during programming.

NOTE: Pin No. is for 100 QFP type package. (for 100 TQFP, the pins with the same name have same functions).

Table 20-2. Comparison of S3FB018 and S3CB018 Features

Characteristic	S3FB519	S3CB519
Program Memory	4K word (8K byte) FLASH ROM	4K word (8K byte) FLASH ROM
Operating Voltage (V_{DD})	1.8 V to 5.5 V	1.8 V to 5.5 V
OTP Programming Mode	V_{DD} = 5 V, V_{PP} (TEST) = 12.5 V	
Pin Configuration	32-SOP/30-SDIP	32-SOP/30-SDIP
FLASH ROM Programmability	User programmable	Programmed at the factory

Table 20-3. Absolute Maximum Ratings

(T_A = 25°C)

Parameter	Symbol	Conditions	Rating	Unit
Supply voltage	V_{DD}	–	– 0.3 to + 6.0	V
Input voltage	V_I	–	– 0.3 to V_{DD} + 0.3	
Output voltage	V_O	–	– 0.3 to V_{DD} + 0.3	
Output current high	I_{OH}	One I/O pin active	– 18	mA
		All I/O pins active	– 60	
Output current low	I_{OL}	One I/O pin active	+ 30	
		Total pin current for ports 1, 2, 3	+ 100	
Operating temperature	T_A	–	– 40 to + 85	°C
Storage temperature	T_{STG}	–	– 65 to + 150	

Table 20-4. D.C. Electrical Characteristics

(T_A = – 40°C to + 85°C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Operating Voltage (HSX mode)	V_{DD}	F_{CPU} = 20 MHz	4.5	–	5.5	V
		F_{CPU} = 3 MHz	1.8		5.5	
Operating Voltage (MSX mode)	V_{DD}	F_{CPU} = 10 MHz	4.5	–	5.5	
		F_{CPU} = 3 MHz	1.8		5.5	

Table 20-4. D.C. Electrical Characteristics (Continued)

(T_A = –40°C to + 85°C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input high voltage	V_{IH1}	All input pins except V_{IH2}	0.8 V_{DD}	–	V_{DD}	V
	V_{IH2}	X_{IN}	V_{DD}- 0.1			
Input low voltage	V_{IL1}	All input pins except V_{IL2}	–	–	0.2 V_{DD}	
	V_{IL2}	X_{IN}			0.1	
Output high voltage	V_{OH1}	V_{DD} = 5V; I_{OH} = -1 mA All output pins	V_{DD}-1.0	–	–	V
Output low voltage	V_{OL1}	V_{DD} = 5V; I_{OL} = 8 mA All output pins except V_{OL2}	–	–	2	
	V_{OL2}	V_{DD} = 5V; I_{OL} = 15 mA, Port 1			2	
Input high leakage current	I_{LIH1}	V_{IN} = V_{DD} All input pins except I_{LIH2}	–	–	3	uA
	I_{LIH2}	V_{IN} = V_{DD} X_{IN}, XT_{IN}			20	
Input low leakage current	I_{LIL1}	V_{IN} = 0 V All input pins except I_{LIL2}	–	–	-3	
	I_{LIL2}	V_{IN} = 0 V X_{IN}, XT_{IN}, RESET			-20	
Output high leakage current	I_{LOH}	V_{OUT} = V_{DD} All I/O pins and Output pins	–	–	3	uA
Output low leakage current	I_{LOL}	V_{OUT} = 0 V All I/O pins and Output pins	–	–	-3	
Oscillator feed back resistors	R_{osc1} (HSX)	V_{DD} = 5.0 V, T_A = 25°C X_{IN} = V_{DD}, X_{OUT} = 0V	510	710	910	kΩ
	R_{osc2} (MSX)	V_{DD} = 5.0 V, T_A = 25°C X_{IN} = V_{DD}, X_{OUT} = 0V	510	710	910	
	R_{osc3} (LSX)	V_{DD} = 5.0 V, T_A = 25°C X_{IN} = V_{DD}, X_{OUT} = 0V	2.0	2.7	3.5	MΩ
Pull-up resistor	R_{L1}	V_{IN} = 0 V; V_{DD} = 5 V ± 10% Ports 0,1,2,3,4,5 T_A=25°C	30	50	70	kΩ
	R_{L2}	V_{IN} = 0 V; V_{DD} = 5 V ± 10% T_A=25°C, RESET only	110	210	310	

Table 20-4. D.C. Electrical Characteristics (Continued)

(T_A = –40°C to + 85°C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Supply current [1]	I_{DD1}[2]	Operating mode: V_{DD} = 5 V ± 10% **20 MHz** crystal oscillator(HSX)	–	10	20	mA
		5 MHz crystal oscillator(MSX)		4	8	
		V_{DD} = 3 V ± 10% **5 MHz** crystal oscillator(MSX)		2	4	
	I_{DD2}[3]	Idle mode: V_{DD} = 5 V ± 10% **20 MHz** crystal oscillator(HSX)	–	2.5	5	mA
		5 MHz crystal oscillator(MSX)		1	2	
		V_{DD} = 3 V± 10% **5 MHz** crystal oscillator(MSX)		0.4	0.8	
	I_{DD3}	Stop mode V_{DD} = 5 V ± 10%	–	0.5	3	uA
		V_{DD} = 3 V ± 10%		0.2	1.2	

NOTES:

1. Supply current does not include current drawn through internal pull-up resistors or external output current loads.
2. In operating current test mode Timer A and Timer B are running.
3. In idle current test mode the Watch timer is running.
4. The operating and idle currents are measured at weak mode.