1 PRODUCT OVERVIEW

OVERVIEW

The S3C7295 single-chip CMOS microcontroller has been designed for high performance using Samsung's newest 4-bit CPU core, SAM47 (Samsung Arrangeable Microcontrollers).

With an up-to-704-dot LCD direct drive capability, and flexible 8-bit timer/counter, the S3C7295 offers an excellent design solution for a mid-end LCD game.

Up to 8 pins of the 80-pin QFP package can be dedicated to I/O. Six vectored interrupts provide fast response to internal and external events. In addition, the S3C7295's advanced CMOS technology provides for low power consumption.

OTP

The S3C7295 microcontroller is also available in OTP (One Time Programmable) version, S3P7295. S3P7295 microcontroller has an on-chip 16K-byte one-time-programable EPROM instead of masked ROM.
The S3P7295 is comparable to S3C7295, both in function and in pin configuration.

FEATURES

Memory

- 256 × 4-bit RAM (excluding LCD display RAM)
- 16,384 × 8-bit ROM

8 I/O Pins

- I/O: 8 pins

LCD Controller/Driver

- 44 segments and 16 common terminals
  (8, 12 and 16 common selectable)
- Internal resistor circuit for LCD bias
- Voltage doubler
- All dot can be switched on/off

8-bit Basic Timer

- 4 interval timer functions
- Watch-dog timer

8-bit Timer/Counter

- Programmable 8-bit timer
- Arbitrary clock output (TCLO0)
- Inverted clock output (TCLO0)

Watch Timer

- Time interval generation: 0.5 s, 3.9 ms at 32768 Hz
- Four frequency outputs to BUZ pin and BUZ pin
- Clock source generation for LCD

Interrupts

- Two internal vectored interrupts
- Four external vectored interrupts
- Two quasi-interrupts

Memory-Mapped I/O Structure

- Data memory bank 15

Power-Down Modes

- Idle mode (only CPU clock stops)
- Stop mode (main system oscillation stops)
- Sub system clock stop mode

Oscillation Sources

- Crystal, ceramic, or RC for main system clock
- Crystal oscillator for subsystem clock
- Main system clock frequency: 4.19 MHz (typical)
- Subsystem clock frequency: 32.768 kHz
- CPU clock divider circuit (by 4, 8, or 64)

Instruction Execution Times

- 0.95, 1.91, 15.3 µs at 4.19 MHz (main)
- 122 µs at 32.768 kHz (subsystem)

Operating Temperature

- – 40 °C to 85 °C

Operating Voltage Range

- 2.2 V to 3.4 V (0.4 MHz to 4.19 MHz)

Package Type

- 80-pin QFP or pellet

BLOCK DIAGRAM

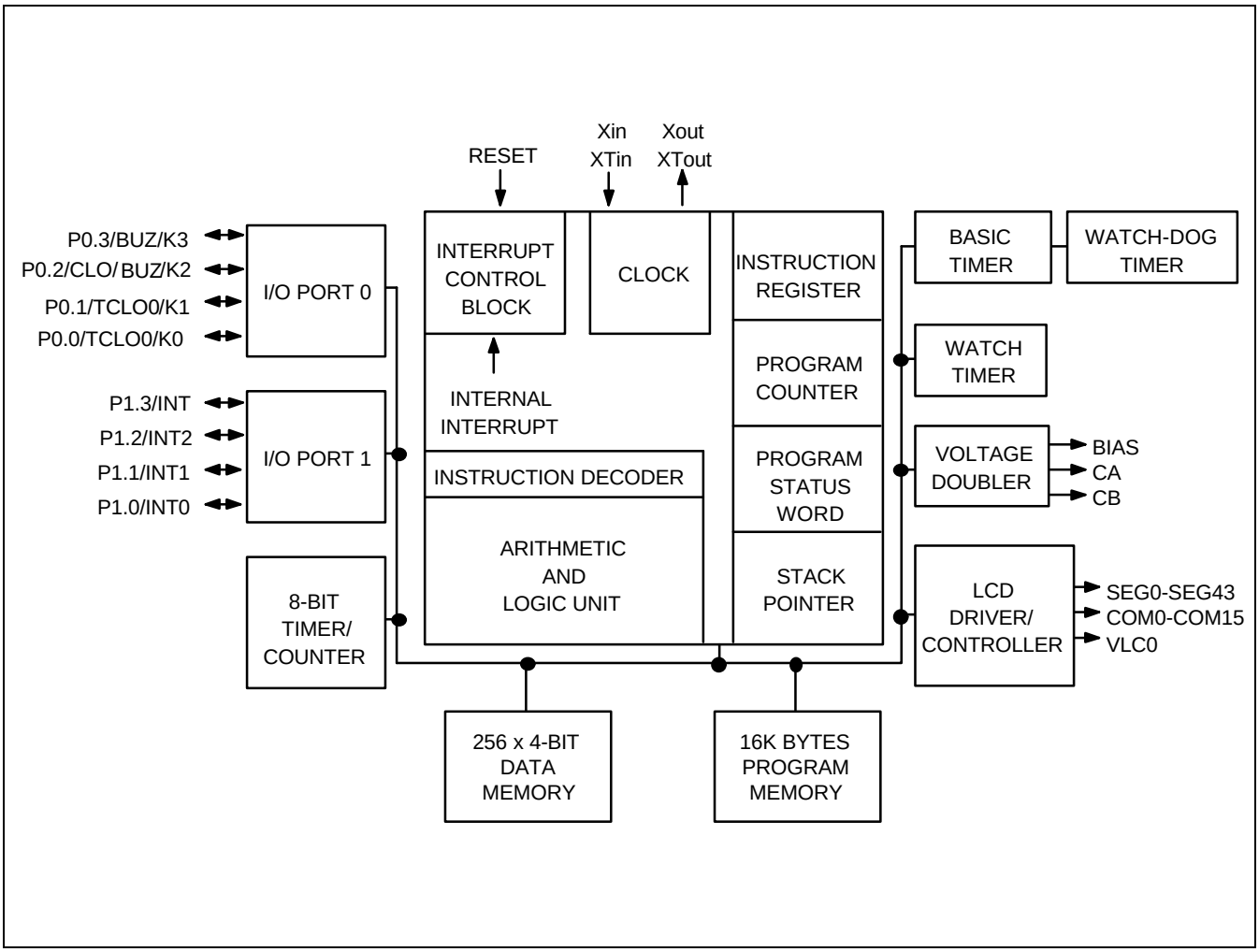


Figure 1-1. S3C7295 Simplified Block Diagram

PIN ASSIGNMENTS

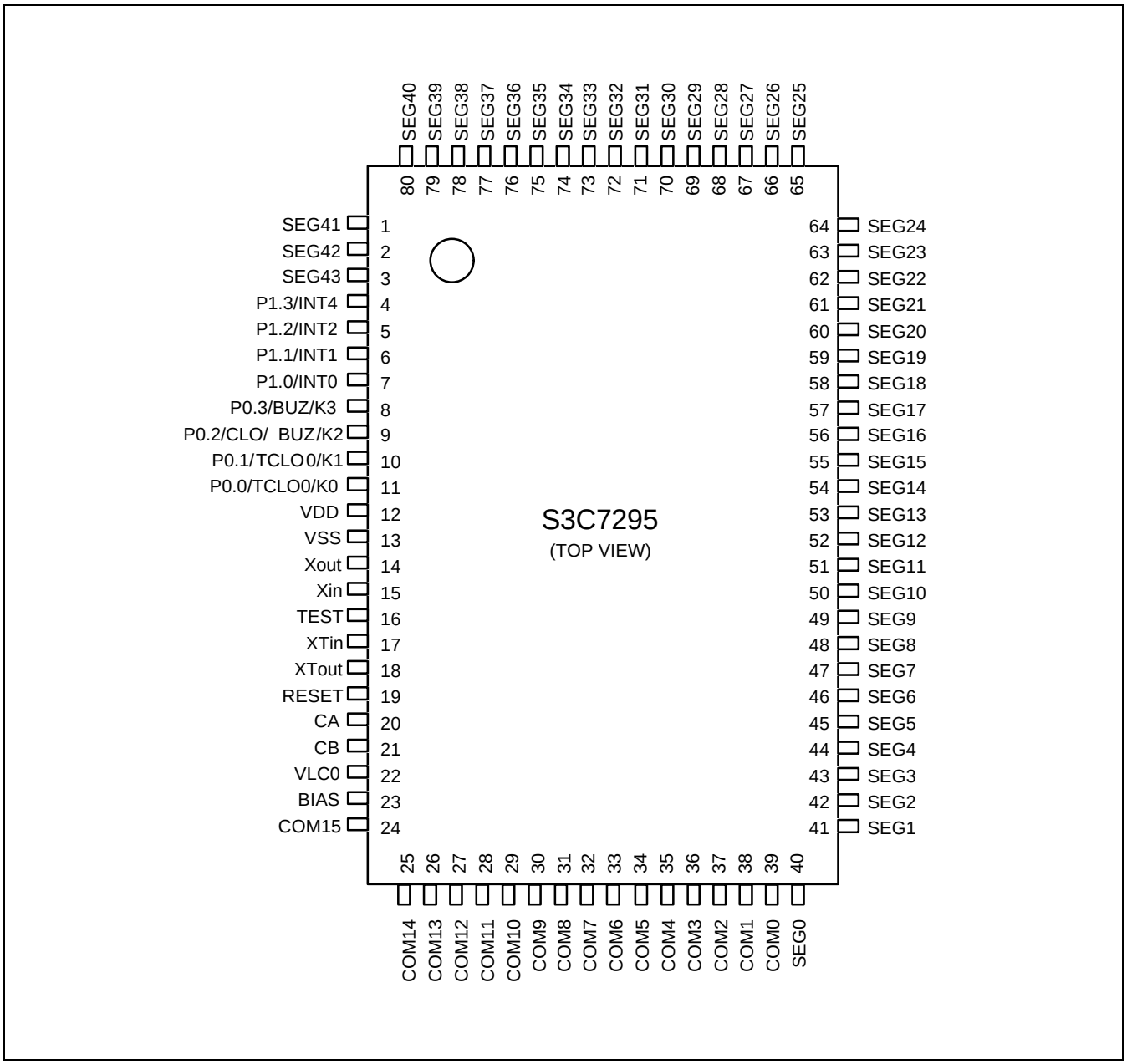


Figure 1-2. S3C7295 80-QFP Pin Assignment Diagram

PIN DESCRIPTIONS

Table 1-1. S3C7295 Pin Descriptions

Pin Name	Pin Type	Description	Circuit Type	Number	Share Pin
P0.0 P0.1 P0.2 P0.3	I/O	4-bit I/O port. 1-bit and 4-bit read/write and test are possible. Individual pins are software configurable as input or output. Individual pins are software configurable as open-drain or push-pull output. Individual pull-up resistors are software assignable; pull-up resistors are automatically disabled for output pins.	E-1	11 10 9 8	TCLO0/K0 TCLO0/K1 CLO/BUZ/K2 BUZ/K3
P1.0 P1.1 P1.2 P1.3	I/O	Same as port 0.	E-1	7 6 5 4	INT0 INT1 INT2 INT4
INT0, INT1	I/O	External interrupts. The triggering edge for INT0 and INT1 is selectable.		7, 6	P1.0, P1.1
INT2	I/O	Quasi-interrupt with detection of rising or falling edges		5	P1.2
INT4	I/O	External interrupt with detection of rising or falling edges.		4	P1.3
BUZ	I/O	2 kHz, 4 kHz, 8 kHz or 16 kHz frequency output for buzzer sound.		8	P0.3/K3
BUZ	I/O	Inverted BUZ signal		9	P0.2/CLO/K2
CLO	I/O	Clock output		9	P0.2/BUZ/K2
TCLO0	I/O	Inverted Timer/counter 0 clock output		10	P0.1/K1
TCLO0	I/O	Timer/counter 0 clock output		11	P0.0/K0
COM0–COM15	O	LCD common signal output	H-6	39–24	–
SEG0–SEG43	O	LCD segment signal output	H-6	40–80, 1–3	–

Table 1-1. S3C7295 Pin Descriptions (Continued)

Pin Name	Pin Type	Description	Circuit Type	Number	Share Pin
K0–K3	I/O	External interrupt (triggering edge is selectable)	E-1	11–8	P0.0–P0.3
V_{DD}	–	Power supply	–	12	–
V_{SS}	–	Ground	–	13	–
RESET	I	Reset input (active low)	B	19	–
CA, CB	–	Capacitor terminal for voltage doubling	–	20, 21	–
VCL0	–	LCD power supply input	–	22	–
BIAS	O	Doubling voltage level output	–	23	–
X_{in}, X_{out}	–	Crystal, ceramic or RC oscillator pins for system clock	–	15, 14	–
XT_{in}, XT_{out}	–	Crystal oscillator pins for subsystem clock	–	17, 18	–
TEST	I	Test input (must be connected to V_{SS})	–	16	–

NOTE: Pull-up resistors for all I/O ports are automatically disabled if they are configured to output mode.

PIN CIRCUIT DIAGRAMS

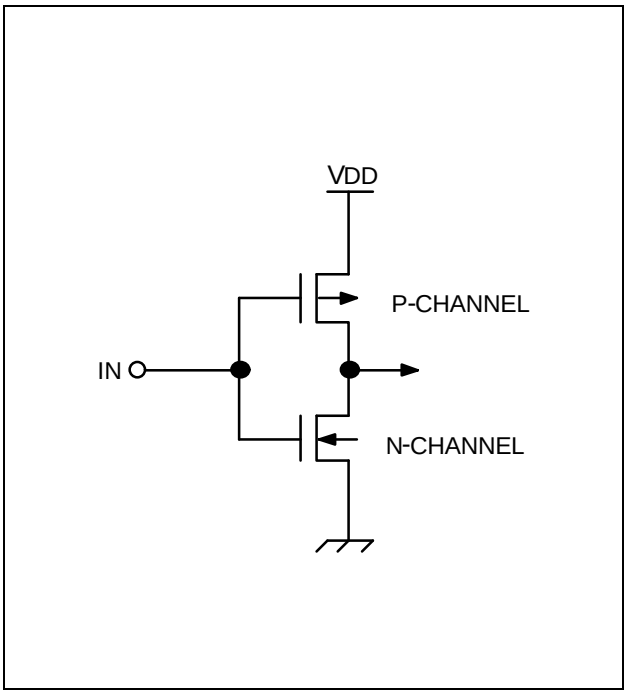


Figure 1-3. Pin Circuit Type A

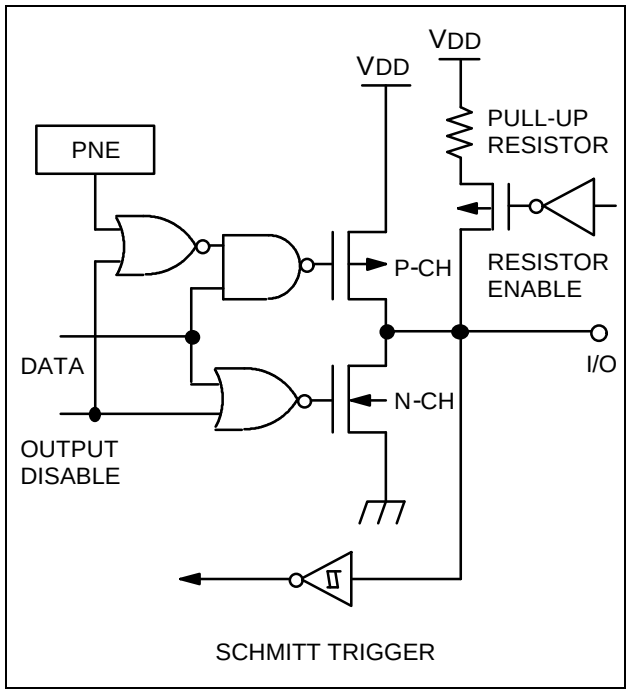


Figure 1-5. Pin Circuit Type E-1

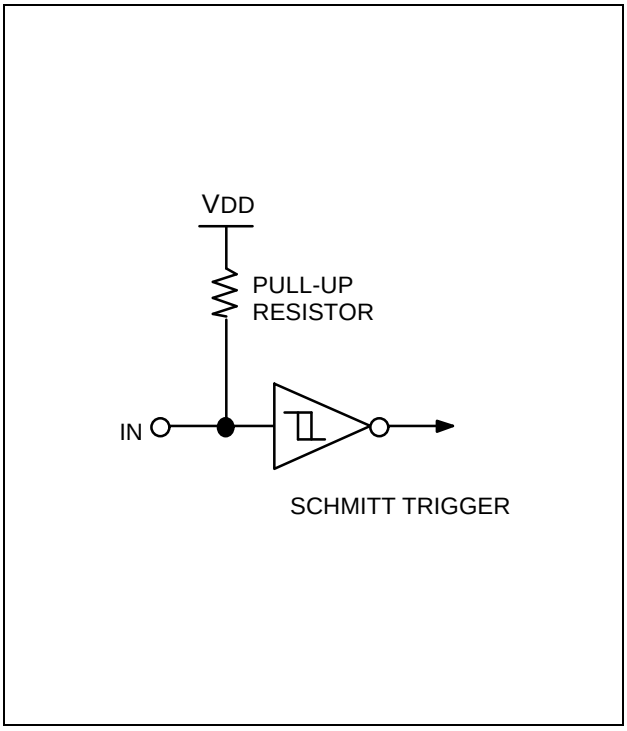


Figure 1-4. Pin Circuit Type B

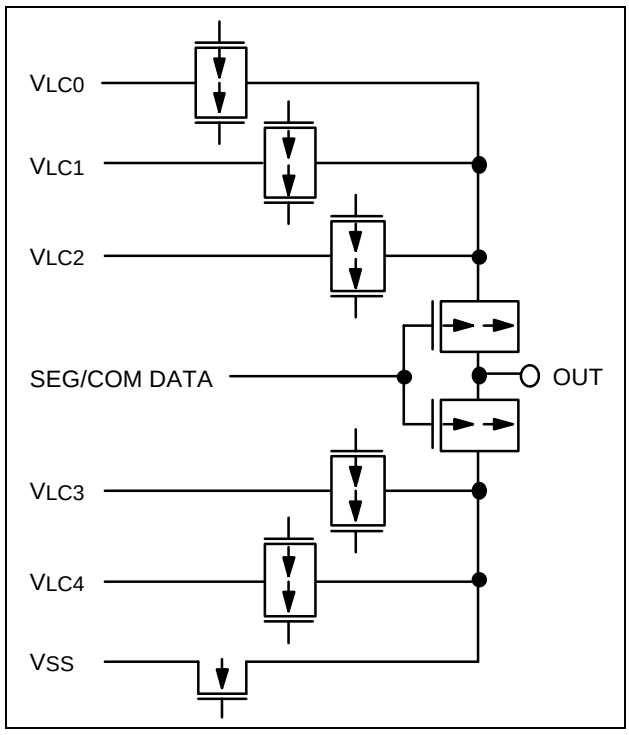


Figure 1-6. Pin Circuit Type H-6

2 ADDRESS SPACES

PROGRAM MEMORY (ROM)

OVERVIEW

ROM maps for S3C7295 devices are mask programmable at the factory. In its standard configuration, the device's 16,384 × 8-bit program memory has three areas that are directly addressable by the program counter (PC):

— 16-byte area for vector addresses

— 96-byte instruction reference area

— 16-byte general-purpose area

— 16,256-byte general-purpose area

General-purpose Program Memory

Two program memory areas are allocated for general-purpose use: One area is 16 bytes in size and the other is 16,256 bytes.

Vector Addresses

A 16-byte vector address area is used to store the vector addresses required to execute system reset and interrupts. Start addresses for interrupt service routines are stored in this area, along with the values of the enable memory bank (EMB) and enable register bank (ERB) flags that are used to set their initial value for the corresponding service routines. The 16-byte area can be used alternately as general-purpose ROM.

REF Instructions

Locations 0020H–007FH are used as a reference area (look-up table) for 1-byte REF instructions. The REF instruction reduces the byte size of instruction operands. REF can reference one 2-byte instruction, two 1-byte instructions, and one 3-byte instruction which are stored in the look-up table. Unused look-up table addresses can be used as general-purpose ROM.

Table 2-1. Program Memory Address Ranges

ROM Area Function	Address Ranges	Area Size (in Bytes)
Vector address area	0000H–000FH	16
General-purpose program memory	0010H–001FH	16
REF instruction look-up table area	0020H–007FH	96
General-purpose program memory	0080H–3FFFH	16,256

GENERAL-PURPOSE MEMORY AREAS

The 16-byte area at ROM locations 0010H–001FH and the 16,256-byte area at ROM locations 0080H–3FFFH are used as general-purpose program memory. Unused locations in the vector address area and REF instruction look-up table areas can be used as general-purpose program memory. However, care must be taken not to overwrite live data when writing programs that use special-purpose areas of the ROM.

VECTOR ADDRESS AREA

The 16-byte vector address area of the ROM is used to store the vector addresses for executing system resets and interrupts. The starting addresses of interrupt service routines are stored in this area, along with the enable memory bank (EMB) and enable register bank (ERB) flag values that are needed to initialize the service routines. 16-byte vector addresses are organized as follows:

EMB	ERB	PC13	PC12	PC11	PC10	PC9	PC8
PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

To set up the vector address area for specific programs, use the instruction VENTn. The programming tips on the next page explain how to do this.

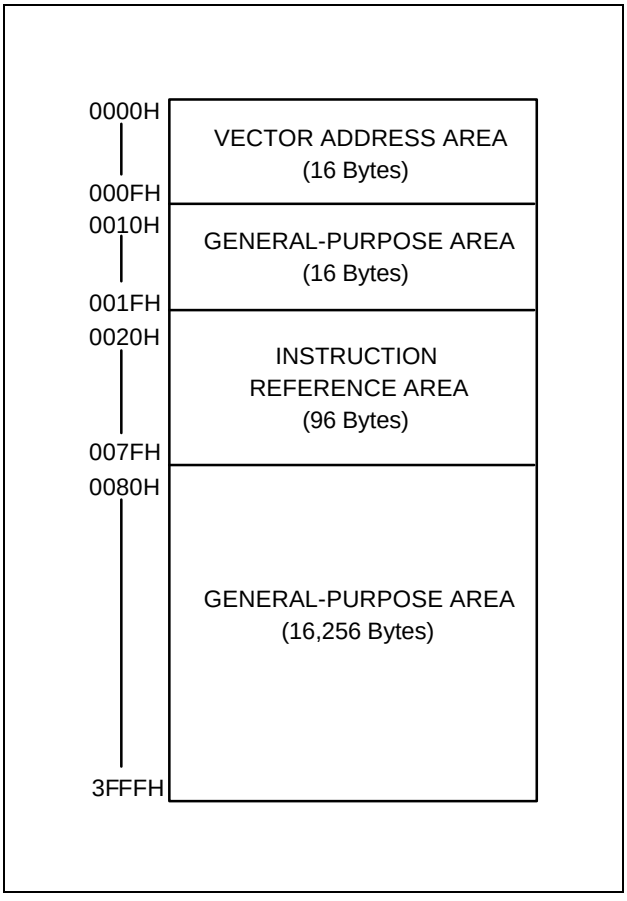


Figure 2-1. ROM Address Structure

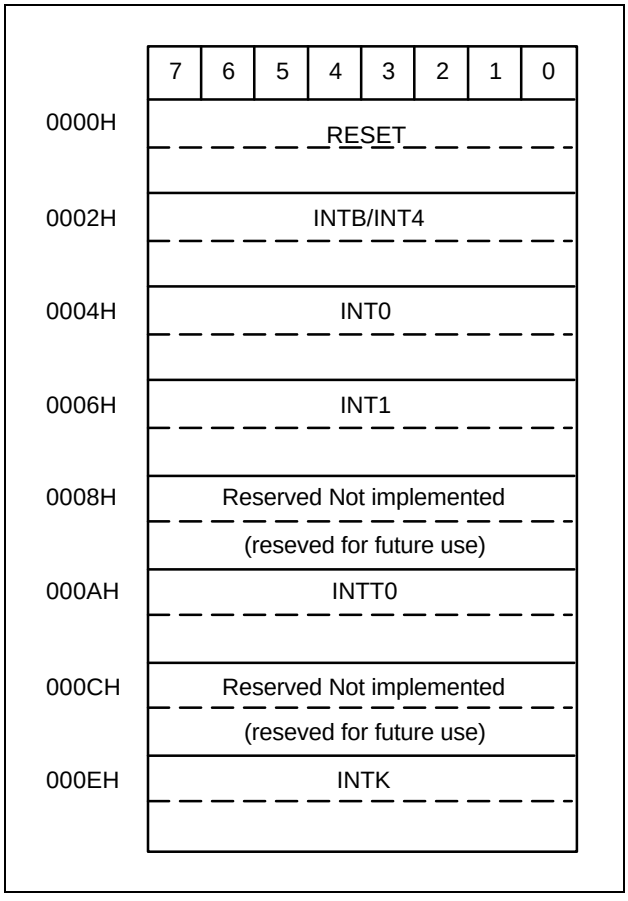


Figure 2-2. Vector Address Map

☞ PROGRAMMING TIP — Defining Vectored Interrupts

The following examples show you several ways you can define the vectored interrupt and instruction reference areas in program memory:

1. When all vector interrupts are used:

```
            ORG     0000H
;
            VENT0   1,0,RESET       ;  EMB ← 1, ERB ← 0; Jump to RESET address by RESET
            VENT1   0,0,INTB        ;  EMB ← 0, ERB ← 0; Jump to INTB address by INTB
            VENT2   0,0,INT0        ;  EMB ← 0, ERB ← 0; Jump to INT0 address by INT0
            VENT3   0,0,INT1        ;  EMB ← 0, ERB ← 0; Jump to INT1 address by INT1
            DB      00, 00          ;
            VENT4   0,0,INTT0       ;  EMB ← 0, ERB ← 0; Jump to INTT0 address by INTT0
            DB      00, 00          ;
            VENT5   0,0,INTK        ;  EMB ← 0, ERB ← 0; Jump to INTK address by INTK
```

2. When a specific vectored interrupt such as INT0, and INTT0 is not used, the unused vector interrupt locations must be skipped with the assembly instruction ORG so that jumps will address the correct locations:

```
            ORG     0000H
;
            VENT0   1,0,RESET       ;  EMB ← 1, ERB ← 0; Jump to RESET address by RESET
            VENT1   0,0,INTB        ;  EMB ← 0, ERB ← 0; Jump to INTB address by INTB
            ORG     0006H           ;  INT0 interrupt not used
            VENT3   0,0,INT1        ;  EMB ← 0, ERB ← 0; Jump to INT1 address by INT1
;
            ORG     000EH           ;  INTT0 interrupt not used
;
            VENT5   0,0,INTK        ;  EMB ← 0, ERB ← 0; Jump to INTK address by INTK
;
            ORG     0010H
```

☞ PROGRAMMING TIP — Defining Vectored Interrupts (Continued)

3. If an INT0 interrupt is not used and if its corresponding vector interrupt area is not fully utilized, or if it is not written by a ORG instruction as in Example 2, a CPU malfunction will occur:

```
                ORG         0000H
;
                VENT0       1,0,RESET       ;  EMB ← 1, ERB ← 0; Jump to RESET address by RESET
                VENT1       0,0,INTB        ;  EMB ← 0, ERB ← 0; Jump to INTB address by INTB
                VENT3       0,0,INT1        ;  EMB ← 0, ERB ← 0; Jump to INT1 address by INT0
                DB          00, 00          ;  Malfunction will occur when INT1 interrupt is generated
                VENT4       0,0,INTT0       ;  Not available
                DB          00, 00          ;  Malfunction will occur when INT0 interrupt is generated
                VENT5       0,0,INTK        ;  Not available
;
                ORG         0010H
;
                General-purpose ROM area
;
```

In this example, when an INTT0 interrupt is generated, the corresponding vector area is not VENT4 INTT0, but 0AH address. This causes an INTT0 interrupt to jump incorrectly to the 0AH address and causes a CPU malfunction to occur.

INSTRUCTION REFERENCE AREA

Using 1-byte REF instructions, you can easily reference instructions with larger byte sizes that are stored in addresses 0020H–007FH of program memory. This 96-byte area is called the REF instruction reference area, or look-up table. Locations in the REF look-up table may contain two 1-byte instructions, a one 2-byte instruction, or one 3-byte instruction such as a JP (jump) or CALL. The starting address of the instruction you are referencing must always be an even number. To reference a JP or CALL instruction, it must be written to the reference area in a two-byte format: for JP, this format is TJP; for CALL, it is TCALL. In summary, there are three ways to the REF instruction:

By using REF instructions, you can execute instructions larger than one byte. In summary, there are three ways you can use the REF instruction:

— Using the 1-byte REF instruction to execute one 2-byte or two 1-byte instructions.

— Branching to any location by referencing a branch instruction stored in the look-up table.

— Calling subroutines at any location by referencing a call instruction stored in the look-up table.

☞ PROGRAMMING TIP — Using The REF Look-up Table

Here is one example of how to use the REF instruction look-up table:

```
            ORG     0020H
;
JMAIN       TJP     MAIN            ;  0, MAIN
KEYCK       BTSF    KEYFG           ;  1, KEYFG CHECK
WATCH       TCALL   CLOCK           ;  2, Call CLOCK
INCHL       LD      @HL,A           ;  3, (HL) ← A
            INCS    HL
            •
            •
            •
ABC         LD      EA,#00H         ;  47, EA ← #00H
            ORG     0080
;
MAIN        NOP
            NOP
            •
            •
            •
            REF     KEYCK           ;  BTSF KEYFG (1-byte instruction)
            REF     JMAIN           ;  KEYFG  =  1, jump to MAIN (1-byte instruction)
            REF     WATCH           ;  KEYFG  =  0, CALL   CLOCK (1-byte instruction)
            REF     INCHL           ;  LD @HL,A
                                    ;  INCS HL
            REF     ABC             ;  LD EA,#00H (1-byte instruction)
            •
            •
            •
```

DATA MEMORY (RAM)

OVERVIEW

In its standard configuration, the data memory has four areas:

— 32 × 4-bit working register area in bank 0

— 224 × 4-bit general-purpose area in bank 0 which is also used as the stack area

— 176 × 4-bit area for LCD data in bank 1

— 128 × 4-bit area in bank 15 for memory-mapped I/O addresses

To make it easier to reference, the data memory area has three memory banks — bank 0, bank 1, and bank 15. The select memory bank instruction (SMB) is used to select the bank you want to select as working data memory. Data stored in RAM locations are 1-, 4-, and 8-bit addressable.

Initialization values for the data memory area are not defined by hardware and must therefore be initialized by program software after the power RESET. However, when RESET signal is generated in power-down mode, most of the data memory contents are held.

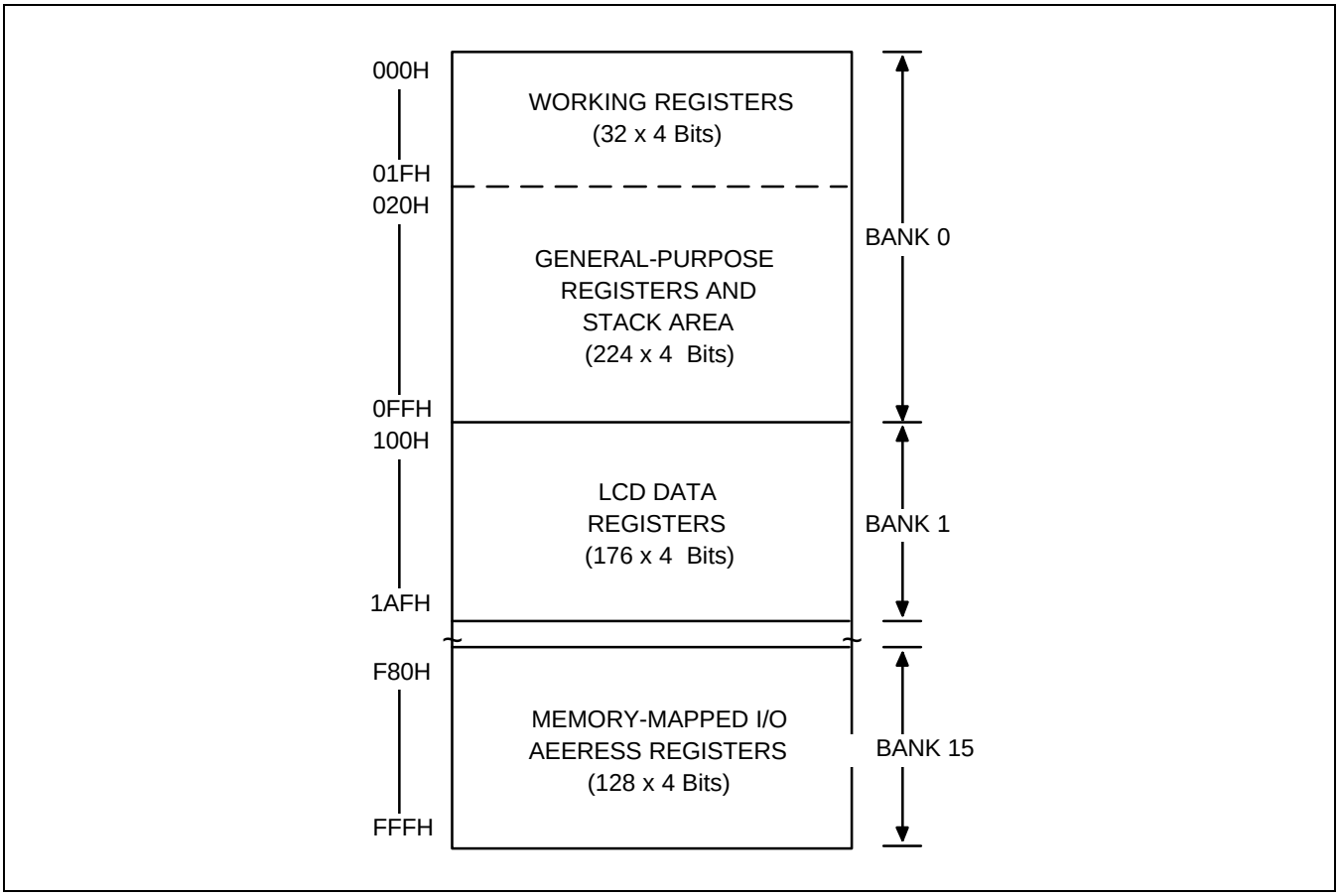


Figure 2-3. Data Memory (RAM) Map

Memory Banks 0, 1, and 15

Bank 0	(000H–0FFH)	The lowest 32 nibbles of bank 0 (000H–01FH) are used as working registers; the next 224 nibbles (020H–0FFH) can be used both as stack area and as general-purpose data memory. Use the stack area for implementing subroutine calls and returns, and for interrupt processing.
Bank 1	(100H–1AFH)	The 176 nibbles of bank 1 are for display registers or general-purpose use; Detailed map on bank 1 is shown in Section 12 LCD Controller/Driver.
Bank 15	(F80H–FFFH)	The microcontroller uses bank 15 for memory-mapped peripheral I/O. Fixed RAM locations for each peripheral hardware address are mapped into this area.

Data Memory Addressing Modes

The enable memory bank (EMB) flag controls the addressing mode for data memory banks 0, 1, or 15. When the EMB flag is logic zero, the addressable area is restricted to specific locations, depending on whether direct or indirect addressing is used. With direct addressing, you can access locations 000H–07FH of bank 0 and bank 15. With indirect addressing, only bank 0 (000H–0FFH) can be accessed. When the EMB flag is set to logic one, all three data memory banks can be accessed according to the current SMB value.

For 8-bit addressing, two 4-bit registers are addressed as a register pair. Also, when using 8-bit instructions to address RAM locations, remember to use the even-numbered register address as the instruction operand.

Working Registers

The RAM working register area in data memory bank 0 is further divided into four *register* banks (bank 0, 1, 2, and 3). Each register bank has eight 4-bit registers and paired 4-bit registers are 8-bit addressable.

Register A is used as a 4-bit accumulator and register pair EA as an 8-bit extended accumulator. The carry flag bit can also be used as a 1-bit accumulator. Register pairs WX, WL, and HL are used as address pointers for indirect addressing. To limit the possibility of data corruption due to incorrect register addressing, it is advisable to use register bank 0 for the main program and banks 1, 2, and 3 for interrupt service routines.

LCD Data Register Area

Bit values for LCD segment data are stored in data memory bank 1. Register locations in this area that are not used to store LCD data can be assigned to general-purpose use.

Table 2-2. Data Memory Organization and Addressing

Addresses	Register Areas	Bank	EMB Value	SMB Value
000H–01FH	Working registers	0	0, 1	0
020H–0FFH	Stack and general-purpose registers			
100H–1AFH	Display registers and general-purpose registers	1	1	1
F80H–FFFH	I/O-mapped hardware registers	15	0, 1	15

☞ **PROGRAMMING TIP — Clearing Data Memory Banks 0 and 1**

Clear banks 0 and 1 of the data memory area:

```
RAMCLR  SMB   1          ;  RAM (100H–1FFH) clear
        LD    HL,#00H
        LD    A,#0H
RMCL1   LD    @HL,A
        INCS  HL
        JR    RMCL1
;
        SMB   0          ;  RAM (010H–0FFH) clear
        LD    HL,#10H
RMCL0   LD    @HL,A
        INCS  HL
        JR    RMCL0
```

WORKING REGISTERS

Working registers, mapped to RAM address 000H-01FH in data memory bank 0, are used to temporarily store intermediate results during program execution, as well as pointer values used for indirect addressing. Unused registers may be used as general-purpose memory. Working register data can be manipulated as 1-bit units, 4-bit units or, using paired registers, as 8-bit units.

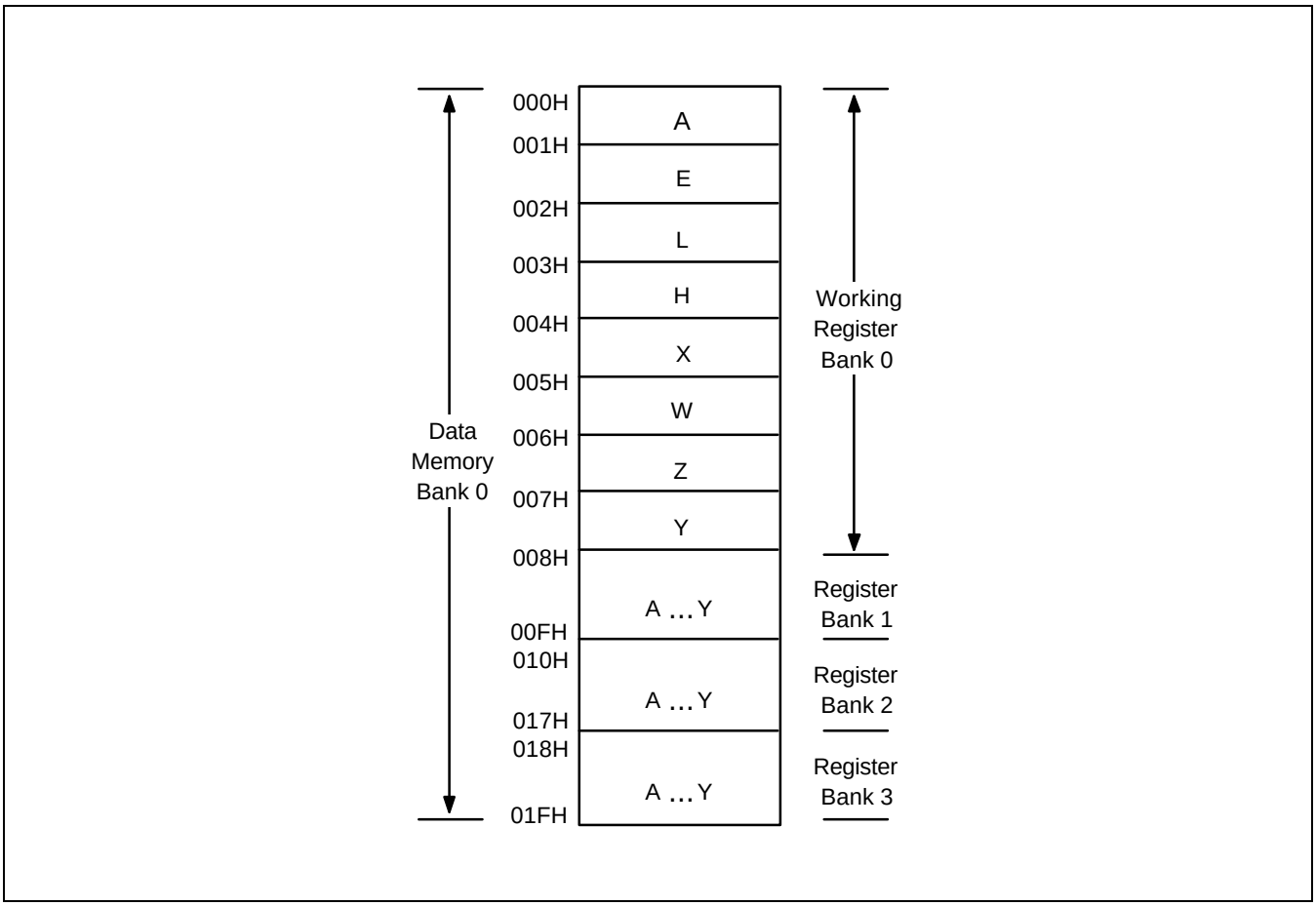


Figure 2-4. Working Register Map

Working Register Banks

For addressing purposes, the working register area is divided into four register banks — bank 0, bank 1, bank 2, and bank 3. Any one of these banks can be selected as the working register bank by the register bank selection instruction (SRB n) and by setting the status of the register bank enable flag (ERB).

Generally, working register bank 0 is used for the main program, and banks 1, 2, and 3 for interrupt service routines. Following this convention helps to prevent possible data corruption during program execution due to contention in register bank addressing.

Table 2-3. Working Register Organization and Addressing

ERB Setting	SRB Settings				Selected Register Bank
	3	2	1	0	
0	0	0	x	x	Always set to bank 0
1	0	0	0	0	Bank 0
			0	1	Bank 1
			1	0	Bank 2
			1	1	Bank 3

NOTE: 'x' means don't care.

Paired Working Registers

Each of the register banks is subdivided into eight 4-bit registers. These registers, named Y, Z, W, X, H, L, E and A, can either be manipulated individually using 4-bit instructions, or together as register pairs for 8-bit data manipulation.

The names of the 8-bit register pairs in each register bank are EA, HL, WX, YZ and WL. Registers A, L, X and Z always become the lower nibble when registers are addressed as 8-bit pairs. This makes a total of eight 4-bit registers or four 8-bit double registers in each of the four working register banks.

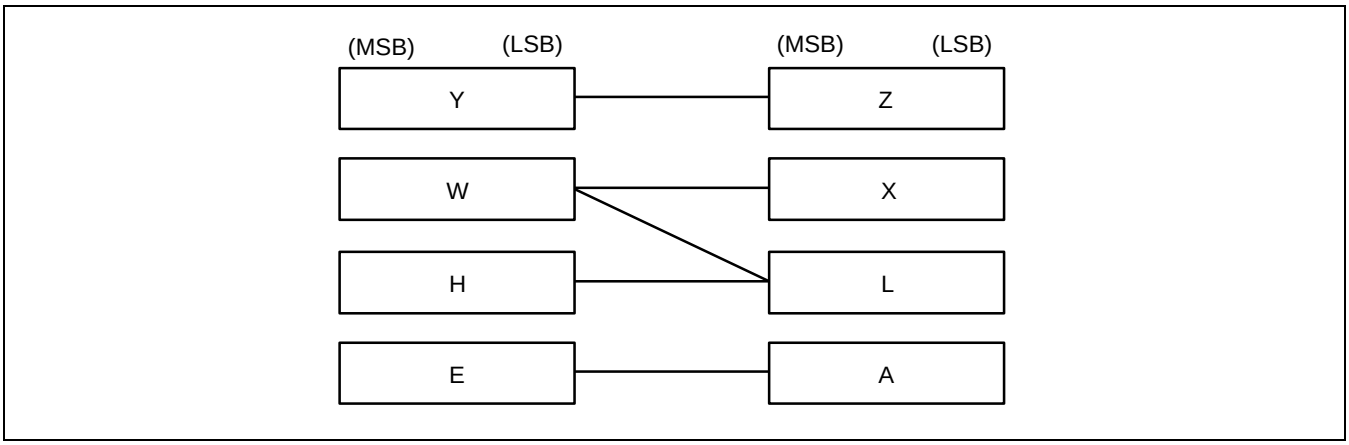


Figure 2-5. Register Pair Configuration

Special-Purpose Working Registers

Register A is used as a 4-bit accumulator and double register EA as an 8-bit accumulator. The carry flag can also be used as a 1-bit accumulator.

8-bit double registers WX, WL and HL are used as data pointers for indirect addressing. When the HL register serves as a data pointer, the instructions LDI, LDD, XCHI, and XCHD can make very efficient use of working registers as program loop counters by letting you transfer a value to the L register and increment or decrement it using a single instruction.

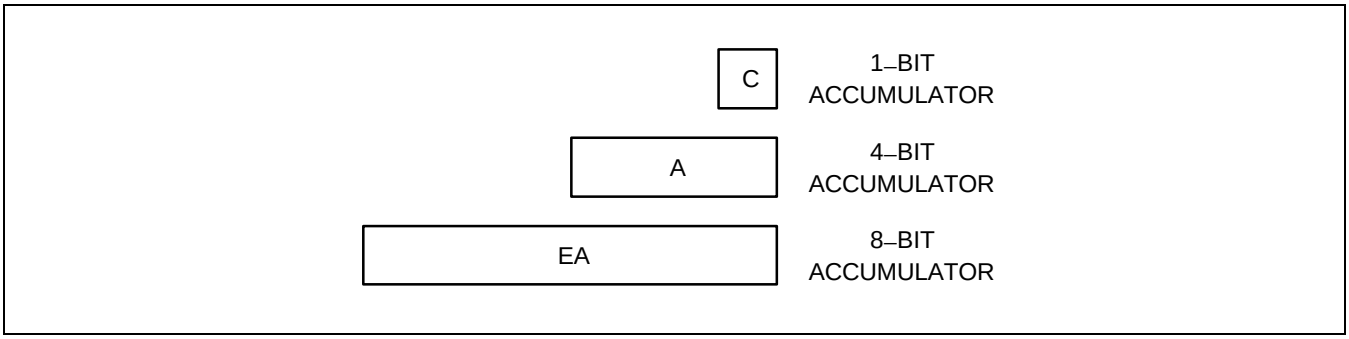


Figure 2-6. 1-Bit, 4-Bit, and 8-Bit Accumulator

Recommendation for Multiple Interrupt Processing

If more than four interrupts are being processed at one time, you can avoid possible loss of working register data by using the PUSH RR instruction to save register contents to the stack before the service routines are executed in the same register bank. When the routines have executed successfully, you can restore the register contents from the stack to working memory using the POP instruction.

☞ PROGRAMMING TIP — Selecting The Working Register Area

The following examples show the correct programming method for selecting working register area:

1. When ERB = "0":

```
VENT2   1,0,INT0                ;  EMB ← 1, ERB ← 0, Jump to INT0 address
;
INT0    PUSH    SB              ;  PUSH current SMB, SRB
        SRB     2               ;  Instruction does not execute because ERB = "0"
        PUSH    HL              ;  PUSH HL register contents to stack
        PUSH    WX              ;  PUSH WX register contents to stack
        PUSH    YZ              ;  PUSH YZ register contents to stack
        PUSH    EA              ;  PUSH EA register contents to stack
        SMB     0
        LD      EA,#00H
        LD      80H,EA
        LD      HL,#40H
        INCS    HL
        LD      WX,EA
        LD      YZ,EA
        POP     EA              ;  POP EA register contents from stack
        POP     YZ              ;  POP YZ register contents from stack
        POP     WX              ;  POP WX register contents from stack
        POP     HL              ;  POP HL register contents from stack
        POP     SB              ;  POP current SMB, SRB
        IRET
```

The POP instructions execute alternately with the PUSH instructions. If an SMB n instruction is used in an interrupt service routine, a PUSH and POP SB instruction must be used to store and restore the current SMB and SRB values, as shown in Example 2 below.

2. When ERB = "1":

```
VENT2   1,1,INT0                ;  EMB ← 1, ERB ← 1, Jump to INT0 address
;
INT0    PUSH    SB              ;  Store current SMB, SRB
        SRB     2               ;  Select register bank 2 because of ERB = "1"
        SMB     0
        LD      EA,#00H
        LD      80H,EA
        LD      HL,#40H
        INCS    HL
        LD      WX,EA
        LD      YZ,EA
        POP     SB              ;  Restore SMB, SRB
        IRET
```

STACK OPERATIONS

STACK POINTER (SP)

The stack pointer (SP) is an 8-bit register that stores the address used to access the stack, an area of data memory set aside for temporary storage of data and addresses. The SP can be read or written by 8-bit control instructions. When addressing the SP, bit 0 must always remain cleared to logic zero.

F80H	SP3	SP2	SP1	"0"
F81H	SP7	SP6	SP5	SP4

There are two basic stack operations: writing to the top of the stack (push), and reading from the top of the stack (pop). A push decrements the SP and a pop increments it so that the SP always points to the top address of the last data to be written to the stack.

The program counter contents and program status word are stored in the stack area prior to the execution of a CALL or a PUSH instruction, or during interrupt service routines. Stack operation is a LIFO (Last In-First Out) type. The stack area is located in general-purpose data memory bank 0.

During an interrupt or a subroutine, the PC value and the PSW are saved to the stack area. When the routine has completed, the stack pointer is referenced to restore the PC and PSW, and the next instruction is executed.

The SP can address stack registers in bank 0 (addresses 000H-0FFH) regardless of the current value of the enable memory bank (EMB) flag and the select memory bank (SMB) flag. Although general-purpose register areas can be used for stack operations, be careful to avoid data loss due to simultaneous use of the same register(s).

Since the reset value of the stack pointer is not defined in firmware, we recommend that you initialize the stack pointer by program code to location 00H. This sets the first register of the stack area to 0FFH.

NOTE

A subroutine call occupies six nibbles in the stack; an interrupt requires six. When subroutine nesting or interrupt routines are used continuously, the stack area should be set in accordance with the maximum number of subroutine levels. To do this, estimate the number of nibbles that will be used for the subroutines or interrupts and set the stack area correspondingly.

☞ PROGRAMMING TIP — Initializing The Stack Pointer

To initialize the stack pointer (SP):

1. When EMB = "1":

```
SMB     15          ;  Select memory bank 15
LD      EA,#00H     ;  Bit 0 of SP is always cleared to "0"
LD      SP,EA       ;  Stack area initial address (0FFH) ← (SP) – 1
```

2. When EMB = "0":

```
LD      EA,#00H
LD      SP,EA       ;  Memory addressing area (00H–7FH, F80H–FFFH)
```

PUSH OPERATIONS

Three kinds of push operations reference the stack pointer (SP) to write data from the source register to the stack: PUSH instructions, CALL instructions, and interrupts. In each case, the SP is decremented by a number determined by the type of push operation and then points to the next available stack location.

PUSH Instructions

A PUSH instruction references the SP to write two 4-bit data nibbles to the stack. Two 4-bit stack addresses are referenced by the stack pointer: one for the upper register value and another for the lower register. After the PUSH has executed, the SP is decremented *by two* and points to the next available stack location.

CALL Instructions

When a subroutine call is issued, the CALL instruction references the SP to write the PC's contents to six 4-bit stack locations. Current values for the enable memory bank (EMB) flag and the enable register bank (ERB) flag are also pushed to the stack. Since six 4-bit stack locations are used per CALL, you may nest subroutine calls up to the number of levels permitted in the stack.

Interrupt Routines

An interrupt routine references the SP to push the contents of the PC and the program status word (PSW) to the stack. Six 4-bit stack locations are used to store this data. After the interrupt has executed, the SP is decremented by six and points to the next available stack location. During an interrupt sequence, subroutines may be nested up to the number of levels which are permitted in the stack area.

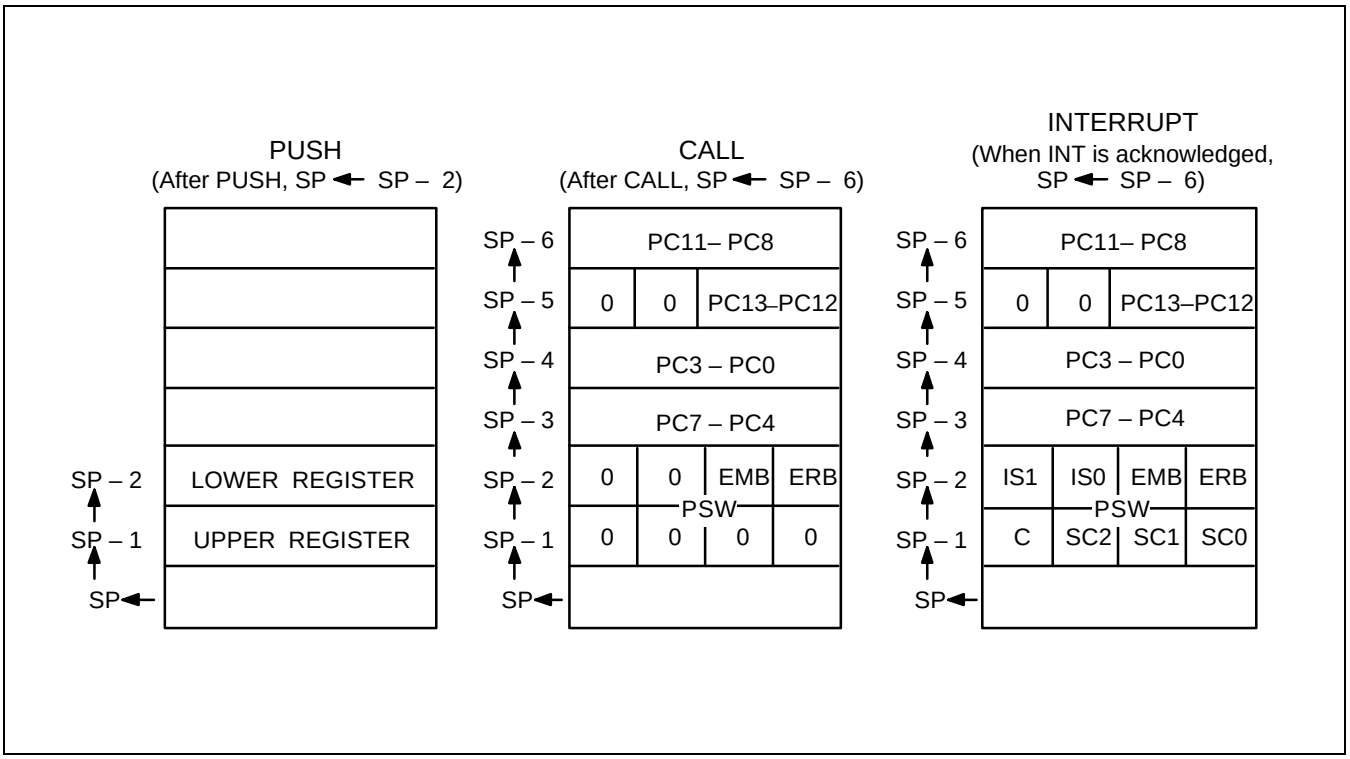


Figure 2-7. Push-Type Stack Operations

POP OPERATIONS

For each push operation there is a corresponding pop operation to write data from the stack back to the source register or registers: for the PUSH instruction it is the POP instruction; for CALL, the instruction RET or SRET; for interrupts, the instruction IRET. When a pop operation occurs, the SP is *incremented* by a number determined by the type of operation and points to the next free stack location.

POP Instructions

A POP instruction references the SP to write data stored in two 4-bit stack locations back to the register pairs and SB register. The value of the lower 4-bit register is popped first, followed by the value of the upper 4-bit register. After the POP has executed, the SP is incremented *by two* and points to the next free stack location.

RET and SRET Instructions

The end of a subroutine call is signaled by the return instruction, RET or SRET. The RET or SRET uses the SP to reference the six 4-bit stack locations used for the CALL and to write this data back to the PC, the EMB, and the ERB. After the RET or SRET has executed, the SP is incremented *by six* and points to the next free stack location.

IRET Instructions

The end of an interrupt sequence is signaled by the instruction IRET. IRET references the SP to locate the six 4-bit stack addresses used for the interrupt and to write this data back to the PC and the PSW. After the IRET has executed, the SP is incremented *by six* and points to the next free stack location.

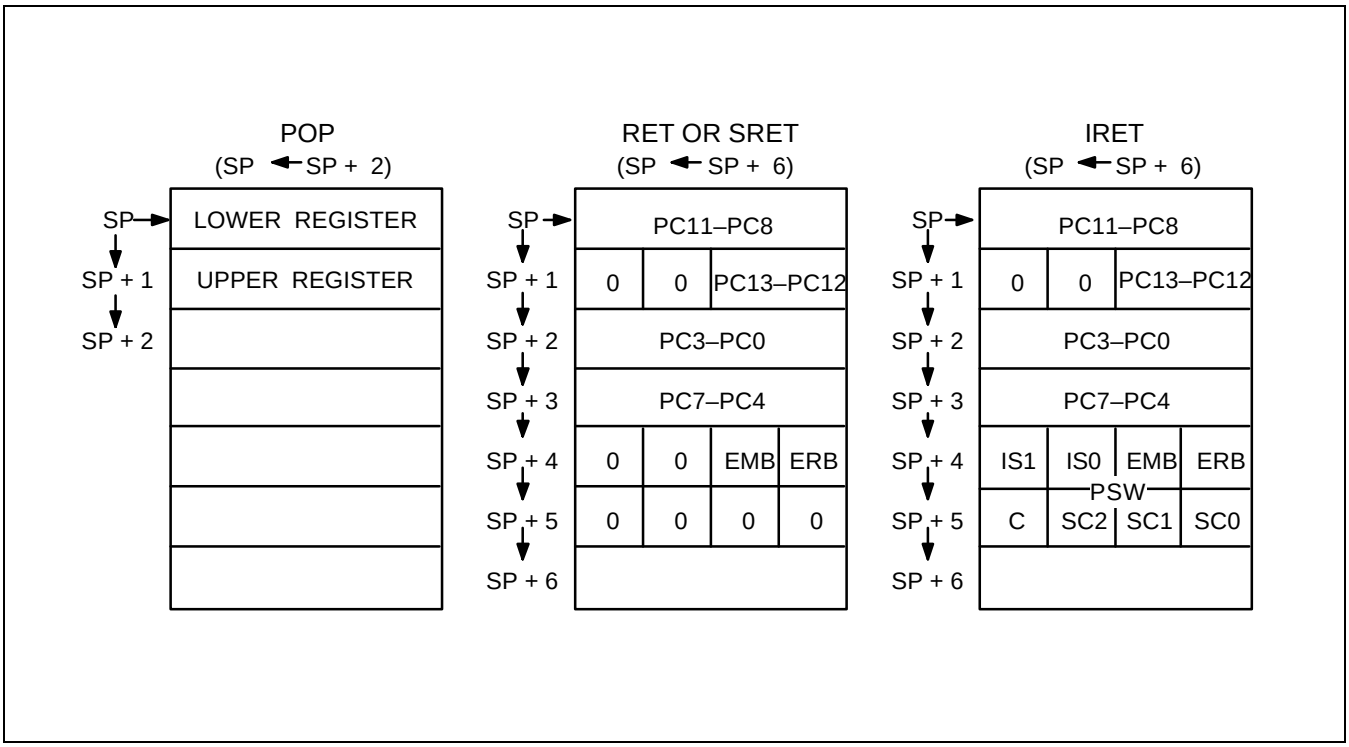


Figure 2-8. Pop-Type Stack Operations

PROGRAM COUNTER (PC)

A 14-bit program counter (PC) stores addresses for instruction fetches during program execution. Whenever a RESET operation or an interrupt occurs, bits PC13 through PC0 are set to the vector address.

Usually, the PC is incremented by the number of bytes of the instruction being fetched. One exception is the 1-byte REF instruction which is used to reference instructions stored in the ROM.

PROGRAM STATUS WORD (PSW)

The program status word (PSW) is an 8-bit word that defines system status and program execution status and which permits an interrupted process to resume operation after an interrupt request has been serviced. PSW values are mapped as follows:

	(MSB)			(LSB)
FB0H	IS1	IS0	EMB	ERB
FB1H	C	SC2	SC1	SC0

The PSW can be manipulated by 1-bit or 4-bit read/write and by 8-bit read instructions, depending on the specific bit or bits being addressed. The PSW can be addressed during program execution regardless of the current value of the enable memory bank (EMB) flag.

Part or all of the PSW is saved to stack prior to execution of a subroutine call or hardware interrupt. After the interrupt has been processed, the PSW values are popped from the stack back to the PSW address.

When a RESET is generated, the EMB and ERB values are set according to the RESET vector address, and the carry flag is left undefined (or the current value is retained). PSW bits IS0, IS1, SC0, SC1, and SC2 are all cleared to logical zero.

Table 2-4. Program Status Word Bit Descriptions

PSW Bit Identifier	Description	Bit Addressing	Read/Write
IS1, IS0	Interrupt status flags	1, 4	R/W
EMB	Enable memory bank flag	1	R/W
ERB	Enable register bank flag	1	R/W
C	Carry flag	1	R/W
SC2, SC1, SC0	Program skip flags	8	R

INTERRUPT STATUS FLAGS (IS0, IS1)

PSW bits IS0 and IS1 contain the current interrupt execution status values. You can manipulate IS0 and IS1 flags directly using 1-bit RAM control instructions

By manipulating interrupt status flags in conjunction with the interrupt priority register (IPR), you can process multiple interrupts by anticipating the next interrupt in an execution sequence. The interrupt priority control circuit determines the IS0 and IS1 settings in order to control multiple interrupt processing. When both interrupt status flags are set to "0", all interrupts are allowed. The priority with which interrupts are processed is then determined by the IPR.

When an interrupt occurs, IS0 and IS1 are pushed to the stack as part of the PSW and are automatically incremented to the next higher priority level. Then, when the interrupt service routine ends with an IRET instruction, IS0 and IS1 values are restored to the PSW. Table 2-5 shows the effects of IS0 and IS1 flag settings.

Table 2-5. Interrupt Status Flag Bit Settings

IS1 Value	IS0 Value	Status of Currently Executing Process	Effect of IS0 and IS1 Settings on Interrupt Request Control
0	0	0	All interrupt requests are serviced
0	1	1	Only high-priority interrupt(s) as determined in the interrupt priority register (IPR) are serviced
1	0	2	No more interrupt requests are serviced
1	1	–	Not applicable; these bit settings are undefined

Since interrupt status flags can be addressed by write instructions, programs can exert direct control over interrupt processing status. Before interrupt status flags can be addressed, however, you must first execute a DI instruction to inhibit additional interrupt routines. When the bit manipulation has been completed, execute an EI instruction to re-enable interrupt processing.

☞ PROGRAMMING TIP — Setting LSX Flags For Interrupt Processing

The following instruction sequence shows how to use the IS0 and IS1 flags to control interrupt processing:

```
INTB    DI                  ;  Disable interrupt
        BITR    IS1         ;  IS1 ← 0
        BITS    IS0         ;  Allow interrupts according to IPR priority level
        EI                  ;  Enable interrupt
```

EMB FLAG (EMB)

The EMB flag is used to allocate specific address locations in the RAM by modifying the upper 4 bits of 12-bit data memory addresses. In this way, it controls the addressing mode for data memory banks 0, 1, or 15.

When the EMB flag is "0", the data memory address space is restricted to bank 15 and addresses 000H–07FH of memory bank 0, regardless of the SMB register contents. When the EMB flag is set to "1", the general-purpose areas of bank 0, 1, and 15 can be accessed by using the appropriate SMB value.

☞ PROGRAMMING TIP — Using the EMB Flag to Select Memory Banks

EMB flag settings for memory bank selection:

1. When EMB = "0":

```
SMB     1           ;  Non-essential instruction since EMB = "0"
LD      A,#9H
LD      90H,A       ;  (F90H) ← A, bank 15 is selected
LD      34H,A       ;  (034H) ← A, bank 0 is selected
SMB     0           ;  Non-essential instruction since EMB = "0"
LD      90H,A       ;  (F90H) ← A, bank 15 is selected
LD      34H,A       ;  (034H) ← A, bank 0 is selected
SMB     15          ;  Non-essential instruction, since EMB = "0"
LD      20H,A       ;  (020H) ← A, bank 0 is selected
LD      90H,A       ;  (F90H) ← A, bank 15 is selected
```

2. When EMB = "1":

```
SMB     1           ;  Select memory bank 1
LD      A,#9H
LD      90H,A       ;  (190H) ← A, bank 1 is selected
LD      34H,A       ;  (134H) ← A, bank 1 is selected
SMB     0           ;  Select memory bank 0
LD      90H,A       ;  (090H) ← A, bank 0 is selected
LD      34H,A       ;  (034H) ← A, bank 0 is selected
SMB     15          ;  Select memory bank 15
LD      20H,A       ;  Program error, but assembler does not detect it
LD      90H,A       ;  (F90H) ← A, bank 15 is selected
```

ERB FLAG (ERB)

The 1-bit register bank enable flag (ERB) determines the range of addressable working register area. When the ERB flag is "1", the working register area from register banks 0 to 3 is selected according to the register bank selection register (SRB). When the ERB flag is "0", register bank 0 is the selected working register area, regardless of the current value of the register bank selection register (SRB).

When an internal RESET is generated, bit 6 of program memory address 0000H is written to the ERB flag. This automatically initializes the flag. When a vectored interrupt is generated, bit 6 of the respective address table in program memory is written to the ERB flag, setting the correct flag status before the interrupt service routine is executed.

During the interrupt routine, the ERB value is automatically pushed to the stack area along with the other PSW bits. Afterwards, it is popped back to the FB0H.0 bit location. The initial ERB flag settings for each vectored interrupt are defined using VENTn instructions.

☞ PROGRAMMING TIP — Using The ERB Flag To Select Register Banks

ERB flag settings for register bank selection:

1. When ERB = "0":

```
SRB     1           ;  Register bank 0 is selected (since ERB = "0", the
                    ;  SRB is configured to bank 0)
LD      EA,#34H     ;  Bank 0 EA ← #34H
LD      HL,EA       ;  Bank 0 HL ← EA
SRB     2           ;  Register bank 0 is selected
LD      YZ,EA       ;  Bank 0 YZ ← EA
SRB     3           ;  Register bank 0 is selected
LD      WX,EA       ;  Bank 0 WX ← EA
```

2. When ERB = "1":

```
SRB     1           ;  Register bank 1 is selected
LD      EA,#34H     ;  Bank 1 EA ← #34H
LD      HL,EA       ;  Bank 1 HL ← Bank 1 EA
SRB     2           ;  Register bank 2 is selected
LD      YZ,EA       ;  Bank 2 YZ ← BANK2 EA
SRB     3           ;  Register bank 3 is selected
LD      WX,EA       ;  Bank 3 WX ← Bank 3 EA
```

SKIP CONDITION FLAGS (SC2, SC1, SC0)

The skip condition flags SC2, SC1, and SC0 in the PSW indicate the current program skip conditions and are set and reset automatically during program execution. Skip condition flags can only be addressed by 8-bit read instructions. Direct manipulation of the SC2, SC1, and SC0 bits is not allowed.

CARRY FLAG (C)

The carry flag is used to save the result of an overflow or borrow when executing arithmetic instructions involving a carry (ADC, SBC). The carry flag can also be used as a 1-bit accumulator for performing Boolean operations involving bit-addressed data memory.

If an overflow or borrow condition occurs when executing arithmetic instructions with carry (ADC, SBC), the carry flag is set to "1". Otherwise, its value is "0". When a RESET occurs, the current value of the carry flag is retained during power-down mode, but when normal operating mode resumes, its value is undefined.

The carry flag can be directly manipulated by predefined set of 1-bit read/write instructions, independent of other bits in the PSW. Only the ADC and SBC instructions, and the instructions listed in Table 2-6, affect the carry flag.

Table 2-6. Valid Carry Flag Manipulation Instructions

Operation Type	Instructions	Carry Flag Manipulation
Direct manipulation	SCF	Set carry flag to "1"
	RCF	Clear carry flag to "0" (reset carry flag)
	CCF	Invert carry flag value (complement carry flag)
	BTST C	Test carry and skip if C = "1"
Bit transfer	LDB (operand) (1),C	Load carry flag value to the specified bit
	LDB C,(operand) (1)	Load contents of the specified bit to carry flag
Boolean manipulation	BAND C,(operand) (1)	AND the specified bit with contents of carry flag and save the result to the carry flag
	BOR C,(operand) (1)	OR the specified bit with contents of carry flag and save the result to the carry flag
	BXOR C,(operand) (1)	XOR the specified bit with contents of carry flag and save the result to the carry flag
Interrupt routine	INTn (2)	Save carry flag to stack with other PSW bits
Return from interrupt	IRET	Restore carry flag from stack with other PSW bits

NOTES:
1. The operand has three bit addressing formats: mema.a, memb.@L, and @H + DA.b.
2. 'INTn' refers to the specific interrupt being executed and is not an instruction.

☞ PROGRAMMING TIP — Using The Carry Flag as a 1-Bit Accumulator

1. Set the carry flag to logic one:

```
SCF                     ;  C ← 1
LD      EA,#0C3H        ;  EA ← #0C3H
LD      HL,#0AAH        ;  HL ← #0AAH
ADC     EA,HL           ;  EA ← #0C3H + #0AAH + #1H, C ← 1
```

2. Logical-AND bit 3 of address 3FH with P1.3 and output the result to P0.0:

```
LD      H,#3H           ;  Set the upper four bits of the address to the H register value
LDB     C,@H+0FH.3      ;  C ← bit 3 of 3FH
BAND    C,P1.3          ;  C ← C  AND  P1.3
LDB     P0.0,C          ;  Output result from carry flag to P0.0
```

3 ADDRESSING MODES

OVERVIEW

The enable memory bank flag, EMB, controls the two addressing modes for data memory. When the EMB flag is set to logic one, you can address the entire RAM area; when the EMB flag is cleared to logic zero, the addressable area in the RAM is restricted to specific locations.

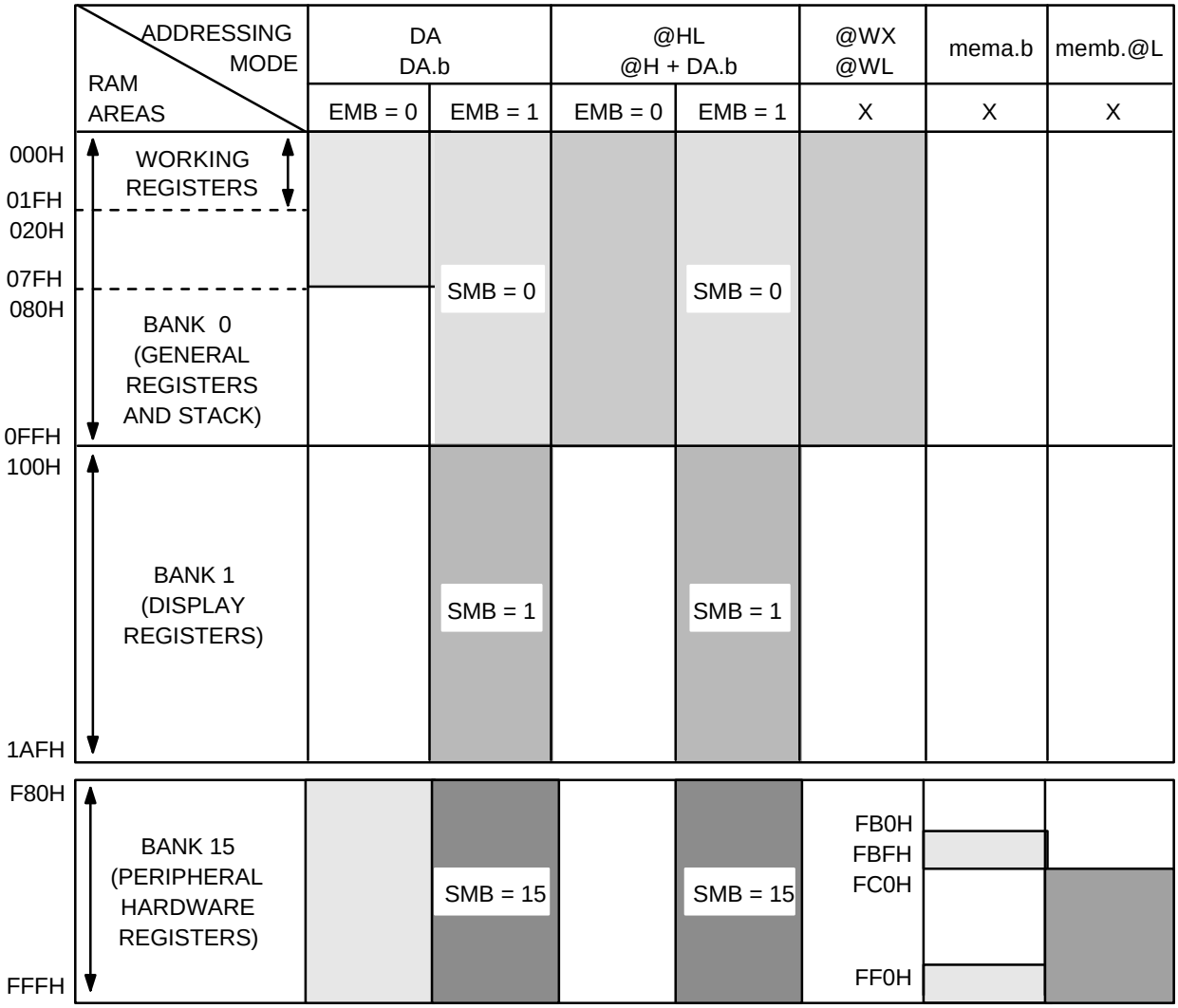
The EMB flag works in connection with the select memory bank instruction, SMBn. You will recall that the SMBn instruction is used to select RAM bank 0, 1, or 15. The SMB setting is always contained in the upper four bits of a 12-bit RAM address. For this reason, both addressing modes (EMB = "0" and EMB = "1") apply specifically to the memory bank indicated by the SMB instruction, and any restrictions to the addressable area within banks 0, 1, or 15. Direct and indirect 1-bit, 4-bit, and 8-bit addressing methods can be used. Several RAM locations are addressable at all times, regardless of the current EMB flag setting.

Here are a few guidelines to keep in mind regarding data memory addressing:

— When you address peripheral hardware locations in bank 15, the mnemonic for the memory-mapped hardware component can be used as the operand in place of the actual address location.

— Always use an even-numbered RAM address as the operand in 8-bit direct and indirect addressing.

— With direct addressing, use the RAM address as the instruction operand; with indirect addressing, the instruction specifies a register which contains the operand's address.



NOTES

1. 'X' means don't care.
2. Blank columns indicate RAM areas that are not addressable, given the addressing method and enable memory bank (EMB) flag setting shown in the column headers.

Figure 3-1. RAM Address Structure

EMB AND ERB INITIALIZATION VALUES

The EMB and ERB flag bits are set automatically by the values of the RESET vector address and the interrupt vector address. When a RESET is generated internally, bit 7 of program memory address 0000H is written to the EMB flag, initializing it automatically. When a vectored interrupt is generated, bit 7 of the respective vector address table is written to the EMB. This automatically sets the EMB flag status for the interrupt service routine. When the interrupt is serviced, the EMB value is automatically saved to stack and then restored when the interrupt routine has completed.

At the beginning of a program, the initial EMB and ERB flag values for each vectored interrupt must be set by using VENT instruction. The EMB and ERB can be set or reset by bit manipulation instructions (BITS, BITR) despite the current SMB setting.

☞ PROGRAMMING TIP — Initializing the EMB and ERB Flags

The following assembly instructions show how to initialize the EMB and ERB flag settings:

```
        ORG     0000H       ; ROM address assignment
        VENT0   1,0,RESET   ; EMB ← 1, ERB ← 0; Jump to RESET address by
                            RESET
        VENT1   0,1,INTB    ; EMB ← 0, ERB ← 1; Jump to INTB address by INTB
        VENT2   0,1,INT0    ; EMB ← 0, ERB ← 1; Jump to INT0 address by INT0
        VENT3   0,1,INT1    ; EMB ← 0, ERB ← 1; Jump to INT1 address by INT1
        DB      00, 00      ;
        VENT4   0,1,INTT0   ; EMB ← 0, ERB ← 1; Jump to INTT0 address by INTT0
        DB      00, 00      ;
        VENT5   0,1,INTK    ; EMB ← 0, ERB ← 1; Jump to INTK address by INTK
          •
          •
          •
RESET   BITR    EMB
```

ENABLE MEMORY BANK SETTINGS

EMB = "1"

When the enable memory bank flag EMB is set to logic one, you can address the data memory bank specified by the select memory bank (SMB) value (0, 1, or 15) using 1-, 4-, or 8-bit instructions. You can use both direct and indirect addressing modes. The addressable RAM areas when EMB = "1" are as follows:

If SMB = 0, 000H–0FFH

If SMB = 1, 100H–1AFH

If SMB = 15, F80H–FFFH

EMB = "0"

When the enable memory bank flag EMB is set to logic zero, the addressable area is defined independently of the SMB value, and is restricted to specific locations depending on whether a direct or indirect address mode is used.

If EMB = "0", the addressable area is restricted to locations 000H–07FH in bank 0 and to locations F80H–FFFH in bank 15 for direct addressing. For indirect addressing, only locations 000H–0FFH in bank 0 are addressable, regardless of SMB value.

To address the peripheral hardware register (bank 15) using indirect addressing, the EMB flag must first be set to "1" and the SMB value to "15". When a RESET occurs, the EMB flag is set to the value contained in bit 7 of ROM address 0000H.

EMB-Independent Addressing

At any time, several areas of the data memory can be addressed independent of the current status of the EMB flag. These exceptions are described in Table 3-1.

Table 3-1. RAM Addressing Not Affected by the EMB Value

Address	Addressing Method	Affected Hardware	Program Examples
000H–0FFH	4-bit indirect addressing using WX and WL register pairs; 8-bit indirect addressing using SP	Not applicable	LD A,@WX PUSH POP
FB0H–FBFH FF0H–FFFH	1-bit direct addressing	PSW, SCMOD, IEx, IRQx, I/O	BITS EMB BITR IE4
FC0H–FFFH	1-bit indirect addressing using the L register	I/O	BTST FF0H.@L BAND C,P1.@L

SELECT BANK REGISTER (SB)

The select bank register (SB) is used to assign the memory bank and register bank. The 8-bit SB register consists of the 4-bit select register bank register (SRB) and the 4-bit select memory bank register (SMB), as shown in Figure 3-2.

During interrupts and subroutine calls, SB register contents can be saved to stack in 8-bit units by the PUSH SB instruction. You later restore the value to the SB using the POP SB instruction.

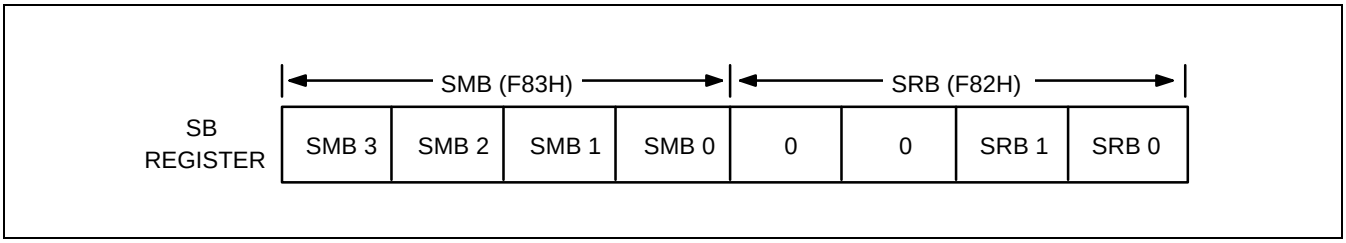


Figure 3-2. SMB and SRB Values in the SB Register

Select Register Bank (SRB) Instruction

The select register bank (SRB) value specifies which register bank is to be used as a working register bank. The SRB value is set by the 'SRB n' instruction, where n = 0, 1, 2, 3.

One of the four register banks is selected by the combination of ERB flag status and the SRB value that is set using the 'SRB n' instruction. The current SRB value is retained until another register is requested by program software. PUSH SB and POP SB instructions are used to save and restore the contents of SRB during interrupts and subroutine calls. RESET clears the 4-bit SRB value to logic zero.

Select Memory Bank (SMB) Instruction

To select one of the four available data memory banks, you must execute an SMB n instruction specifying the number of the memory bank you want (0, 1, or 15). For example, the instruction 'SMB 1' selects bank 1 and 'SMB 15' selects bank 15. (And remember to enable the selected memory bank by making the appropriate EMB flag setting.

The upper four bits of the 12-bit data memory address are stored in the SMB register. If the SMB value is not specified by software (or if a RESET does not occur) the current value is retained. RESET clears the 4-bit SMB value to logic zero.

The PUSH SB and POP SB instructions save and restore the contents of the SMB register to and from the stack area during interrupts and subroutine calls.

DIRECT AND INDIRECT ADDRESSING

1-bit, 4-bit, and 8-bit data stored in data memory locations can be addressed directly using a specific register or bit address as the instruction operand.

Indirect addressing specifies a memory location that contains the required direct address. The KS57 instruction set supports 1-bit, 4-bit, and 8-bit indirect addressing. For 8-bit indirect addressing, an even-numbered RAM address must always be used as the instruction operand.

1-BIT ADDRESSING

Table 3-2. 1-Bit Direct and Indirect RAM Addressing

Operand Notation	Addressing Mode Description	EMB Flag Setting	Addressable Area	Memory Bank	Hardware I/O Mapping
DA.b	Direct: bit is indicated by the RAM address (DA), memory bank selection, and specified bit number (b).	0	000H–07FH	Bank 0	–
			F80H–FFFH	Bank 15	All 1-bit addressable peripherals (SMB = 15)
		1	000H–FFFH	SMB = 0, 1, 15	
mema.b	Direct: bit is indicated by addressable area (mema) and bit number (b).	x	FB0H–FBFH FF0H–FFFH	Bank 15	IS0, IS1, EMB, ERB, IEx, IRQx, Pn.n
memb.@L	Indirect: address is indicated by the upper ten bits of RAM area (memb) and the upper two bits of register L, bit is indicated by the lower two bits of register L.	x	FC0H–FFFH	Bank 15	Pn.n
@H + DA.b	Indirect: bit is indicated by the lower four bits of the address (DA), memory bank selection, and the H register identifier.	0	000H–0FFH	Bank 0	–
		1	000H–FFFH	SMB = 0, 1, 15	All 1-bit addressable peripherals (SMB = 15)

NOTE: 'x' means don't care.

☞ PROGRAMMING TIP — 1-Bit Addressing Modes

1-Bit Direct Addressing

1. If EMB = "0"

```
AFLAG   EQU     34H.3
BFLAG   EQU     85H.3
CFLAG   EQU     0BAH.0
        SMB     0
        BITS    AFLAG           ;  34H.3 ← 1
        BITS    BFLAG           ;  F85H.3 ← 1
        BTST    CFLAG           ;  If FBAH.0 = 1, skip
        BITS    BFLAG           ;  Else if, FBAH.0 = 0, F85H.3 (BMOD.3) ← 1
        BITS    P1.0            ;  FF1H.0 (P1.0) ← 1
```

2. If EMB = "1":

```
AFLAG   EQU     34H.3
BFLAG   EQU     85H.3
CFLAG   EQU     0BAH.0
        SMB     0
        BITS    AFLAG           ;  34H.3 ← 1
        BITS    BFLAG           ;  85H.3 ← 1
        BTST    CFLAG           ;  If 0BAH.0 = 1, skip
        BITS    BFLAG           ;  Else if 0BAH.0 = 0, 085H.3 ← 1
        BITS    P1.0            ;  FF1H.0 (P1.0) ← 1
```

☞ PROGRAMMING TIP — 1-Bit Addressing Modes (Continued)

1-Bit Indirect Addressing

1. If EMB = "0":

```
AFLAG   EQU     34H.3
BFLAG   EQU     85H.3
CFLAG   EQU     0BAH.0
        SMB     0
        LD      H,#0BH          ;  H ← #0BH
        BTSTZ   @H+CFLAG        ;  If 0BAH.0 = 1, 0BAH.0 ← 0 and skip
        BITS    CFLAG           ;  Else if 0BAH.0 = 0, FBAH.0 ← 1
```

2. If EMB = "1":

```
AFLAG   EQU     34H.3
BFLAG   EQU     85H.3
CFLAG   EQU     0BAH.0
        SMB     0
        LD      H,#0BH          ;  H ← #0BH
        BTSTZ   @H+CFLAG        ;  If 0BAH.0 = 1, 0BAH.0 ← 0 and skip
        BITS    CFLAG           ;  Else if 0BAH.0 = 0, 0BAH.0 ← 1
```

4-BIT ADDRESSING

Table 3-3. 4-Bit Direct and Indirect RAM Addressing

Operand Notation	Addressing Mode Description	EMB Flag Setting	Addressable Area	Memory Bank	Hardware I/O Mapping
DA	Direct: 4-bit address indicated by the RAM address (DA) and the memory bank selection	0	000H–07FH	Bank 0	–
			F80H–FFFH	Bank 15	All 4-bit addressable peripherals
		1	000H–FFFH	SMB = 0, 1, 15	(SMB = 15)
@HL	Indirect: 4-bit address indicated by the memory bank selection and register HL	0	000H–0FFH	Bank 0	–
		1	000H–FFFH	SMB = 0, 1, 15	All 4-bit addressable peripherals (SMB = 15)
@WX	Indirect: 4-bit address indicated by register WX	x	000H–0FFH	Bank 0	–
@WL	Indirect: 4-bit address indicated by register WL	x	000H–0FFH	Bank 0	

NOTE: 'x' means don't care.

☞ PROGRAMMING TIP — 4-Bit Addressing Modes

4-Bit Direct Addressing

1. If EMB = "0":

```
ADATA   EQU   46H
BDATA   EQU   8EH
        SMB   15          ;  Non-essential instruction, since EMB = "0"
        LD    A,P0        ;  A ← (P0)
        SMB   0           ;  Non-essential instruction, since EMB = "0"
        LD    ADATA,A     ;  (046H) ← A
        LD    BDATA,A     ;  (F8EH (LCON)) ← A
```

2. If EMB = "1":

```
ADATA   EQU   46H
BDATA   EQU   8EH
        SMB   15
        LD    A,P0        ;  A ← (P0)
        SMB   0
        LD    ADATA,A     ;  (046H) ← A
        LD    BDATA,A     ;  (08EH) ← A
```

☞ PROGRAMMING TIP — 4-Bit Addressing Modes (Continued)

4-Bit Indirect Addressing (Example 1)

1. If EMB = "0", compare bank 0 locations 040H–046H with bank 0 locations 060H–066H:

```
ADATA   EQU     46H
BDATA   EQU     66H
        SMB     1               ;  Non-essential instruction, since EMB = "0"
        LD      HL,#BDATA
        LD      WX,#ADATA
COMP    LD      A,@WL           ;  A ← bank 0 (040H–046H)
        CPSE    A,@HL           ;  If bank 0 (060H–066H) = A, skip
        SRET
        DECS    L
        JR      COMP
        RET
```

2. If EMB = "1", compare bank 0 locations 040H–046H to bank 1 locations 160H–166H:

```
ADATA   EQU     46H
BDATA   EQU     66H
        SMB     1
        LD      HL,#BDATA
        LD      WX,#ADATA
COMP    LD      A,@WL           ;  A ← bank 0 (040H–046H)
        CPSE    A,@HL           ;  If bank 1 (160H–166H) = A, skip
        SRET
        DECS    L
        JR      COMP
        RET
```

☞ PROGRAMMING TIP — 4-Bit Addressing Modes (Concluded)

4-Bit Indirect Addressing (Example 2)

1. If EMB = "0", exchange bank 0 locations 040H–046H with bank 0 locations 060H–066H:

```
ADATA   EQU   46H
BDATA   EQU   66H
        SMB   1              ;  Non-essential instruction, since EMB = "0"
        LD    HL,#BDATA
        LD    WX,#ADATA
TRANS   LD    A,@WL          ;  A ← bank 0 (040H–046H)
        XCHD  A,@HL          ;  Bank 0 (060H–066H) ↔ A
        JR    TRANS
```

2. If EMB = "1", exchange bank 0 locations 040H–046H to bank 1 locations 160H–166H:

```
ADATA   EQU   46H
BDATA   EQU   66H
SMB     1
        LD    HL,#BDATA
        LD    WX,#ADATA
TRANS   LD    A,@WL          ;  A ← bank 0 (040H–046H)
        XCHD  A,@HL          ;  Bank 1 (160H–166H) ↔ A
        JR    TRANS
```

8-BIT ADDRESSING

Table 3-4. 8-Bit Direct and Indirect RAM Addressing

Instruction Notation	Addressing Mode Description	EMB Flag Setting	Addressable Area	Memory Bank	Hardware I/O Mapping
DA	Direct: 8-bit address indicated by the RAM address (DA = even number) and memory bank selection	0	000H–07FH	Bank 0	–
			F80H–FFFH	Bank 15	All 8-bit addressable peripherals (SMB = 15)
		1	000H–FFFH	SMB = 0, 1, 15	
@HL	Indirect: the 8-bit address indicated by the memory bank selection and register HL; (the 4-bit L register value must be an even number)	0	000H–0FFH	Bank 0	–
		1	000H–FFFH	SMB = 0, 1, 15	All 8-bit addressable peripherals (SMB = 15)

☞ PROGRAMMING TIP — 8-Bit Addressing Modes

8-Bit Direct Addressing

1. If EMB = "0":

```
ADATA   EQU   46H
BDATA   EQU   8EH
        SMB   15              ;  Non-essential instruction, since EMB  =  "0"
        LD    A,P0            ;  A ← (P0)
        SMB   0
        LD    ADATA,EA        ;  (046H) ← A, (047H) ← E
        LD    BDATA,EA        ;  (F8EH) ← A, (F8FH) ← E
```

2. If EMB = "1":

```
ADATA   EQU   46H
BDATA   EQU   8EH
        SMB   15
        LD    A,P0            ;  A ← (P0)
        SMB   0
        LD    ADATA,EA        ;  (046H) ← A, (047H) ← E
        LD    BDATA,EA        ;  (08EH) ← A, (08FH) ← E
```

☞ PROGRAMMING TIP — 8-Bit Addressing Modes (Continued)

8-Bit Indirect Addressing

1. If EMB = "0":

```
ADATA   EQU   46H
        SMB   1              ;  Non-essential instruction, since EMB = "0"
        LD    HL,#ADATA
        LD    EA,@HL         ;  A ← (046H), E ← (047H)
```

2. If EMB = "1":

```
ADATA   EQU   46H
        SMB   1
        LD    HL,#ADATA
        LD    EA,@HL         ;  A ← (146H), E ← (147H)
```

4 MEMORY MAP

OVERVIEW

To support program control of peripheral hardware, I/O addresses for peripherals are memory-mapped to bank 15 of the RAM. Memory mapping lets you use a mnemonic as the operand of an instruction in place of the specific memory location.

Access to bank 15 is controlled by the select memory bank (SMB) instruction and by the enable memory bank flag (EMB) setting. If the EMB flag is "0", bank 15 can be addressed using direct addressing, regardless of the current SMB value. 1-bit direct and indirect addressing can be used for specific locations in bank 15, regardless of the current EMB value.

I/O MAP FOR HARDWARE REGISTERS

Table 4-1 contains detailed information about I/O mapping for peripheral hardware in bank 15 (register locations F80H–FFFH). Use the I/O map as a quick-reference source when writing application programs. The I/O map gives you the following information:

— Register address
— Register name (mnemonic for program addressing)
— Bit values (both addressable and non-manipulable)
— Read-only, write-only, or read and write addressability
— 1-bit, 4-bit, or 8-bit data manipulation characteristics

Table 4-1. I/O Map for Memory Bank 15

Memory Bank 15							Addressing Mode		
Address	**Register**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**R/W**	**1-Bit**	**4-Bit**	**8-Bit**
F80H	SP	.3	.2	.1	"0"	R/W	No	No	Yes
F81H		.7	.6	.5	.4				
Locations F82H–F84H are not mapped.									
F85H	BMOD	.3	.2	.1	.0	W	.3	Yes	No
F86H	BCNT					R	No	No	Yes
F87H									
F88H	WMOD	.3	.2	.1	.0	W	.3 (1)	No	Yes
F89H		.7	"0"	.5	.4				
Locations F8AH–F8BH are not mapped.									
F8CH	LMOD	.3	.2	.1	.0	W	No	No	Yes
F8DH		.7	.6	.5	.4				
F8EH	LCON	"0"	.2	.1	.0	W	No	Yes	No
Location F8FH is not mapped.									
F90H	TMOD0	.3	.2	"0"	"0"	W	.3	No	Yes
F91H		"0"	.6	.5	.4				
F92H		TOEB	TOE	"U"	BUZB	R/W	Yes	Yes	No
Location F93H is not mapped.									
F94H	TCNT0					R	No	No	Yes
F95H									
F96H	TREF0					W	No	No	Yes
F97H									
F98H	WDMOD	.3	.2	.1	.0	W	No	No	Yes
F99H		.7	.6	.5	.4				
F9AH	WDFLAG (2)	WDTCF	“0”	“0”	“0”	W	.3	Yes	No
Locations F9BH–FAFH are not mapped.									
FB0H	PSW	IS1	IS0	EMB	ERB	R/W	Yes	Yes	Yes
FB1H		C (3)	SC2	SC1	SC0	R	No	No	
FB2H	IPR	IME	.2	.1	.0	W	IME	Yes	No
FB3H	PCON	.3	.2	.1	.0	W	No	Yes	No
FB4H	IMOD0	“0”	"0"	.1	.0	W	No	Yes	No
FB5H	IMOD1	"0"	"0"	"0"	.0				
Location FB6H is not mapped.									

Table 4-1. I/O Map for Memory Bank 15 (Continued)

Memory Bank 15							Addressing Mode		
Address	**Register**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**R/W**	**1-Bit**	**4-Bit**	**8-Bit**
FB7H	SCMOD	.3	.2	"0"	.0	W	Yes	No	No
FB8H		IE4	IRQ4	IEB	IRQB	R/W	Yes	Yes	No
Location FB9H is not mapped.									
FBAH		"0"	"0"	IEW	IRQW	R/W	Yes	Yes	No
FBBH		IEK	IRQK	"0"	"0"				
FBCH		"0"	"0"	IET0	IRQT0				
Location FBDH is not mapped.									
FBEH		IE1	IRQ1	IE0	IRQ0	R/W	Yes	Yes	No
FBFH		"0"	"0"	IE2	IRQ2				
Locations FC0H–FCBH are not mapped.									
FCCH	IMODK (4)	IMODK.3	IMODK.2	IMODK.1	IMODK.0	W	No	No	Yes
FCDH		IMODK.7	"0"	"0"	"0"				
Location FCEH is not mapped.									
FCFH	IMOD2	"0"	"0"	"0"	.0	W	No	Yes	No
FD0H	CLMOD	.3	"0"	.1	.0	W	No	Yes	No
Locations FD1H–FD9H are not mapped.									
FDAH	PNE1	PNE0.3	PNE0.2	PNE0.1	PNE0.0	W	No	No	Yes
FDBH		PNE1.3	PNE1.2	PNE1.1	PNE1.0				
FDCH	PUMOD0	PUR0.3	PUR0.2	PUR0.1	PUR0.0	W	No	No	Yes
FDDH		PUR1.3	PUR1.2	PUR1.1	PUR1.0				
Locations FDFH–FE7H are not mapped.									
FE8H	PMG1	PM0.3	PM0.2	PM0.1	PM0.0	W	No	No	Yes
FE9H		PM1.3	PM1.2	PM1.1	PM1.0				
Locations FEAH–FEFH are not mapped.									
FF0H	P0	.3	.2	.1	.0	R/W	Yes	Yes	No
FF1H	P1	.3	.2	.1	.0	R/W			
Locations FF2H–FFFH are not mapped.									

NOTES:

1. Bit 3 in the WMOD register is read-only.
2. F9AH.0, F9AH.1 and F9AH.2 are fixed to "0".
3. The carry flag can be read or written by specific bit manipulation instruction only.
4. IMODK (External Key Interrupt Mode Register) can be enabled/disabled by 1-bit unit.
5. The "U" means a undefined register bit.

REGISTER DESCRIPTIONS

In this section, register descriptions are presented in a consistent format to familiarize you with the memory-mapped I/O locations in bank 15 of the RAM. Figure 4-1 describes features of the register description format. Register descriptions are arranged in alphabetical order. Programmers can use this section as a quick-reference source when writing application programs.

Counter registers, buffer registers, and reference registers, as well as the stack pointer and port I/O latches, are not included in these descriptions. More detailed information about how these registers are used is included in Part II of this manual, "Hardware Descriptions," in the context of the corresponding peripheral hardware module descriptions.

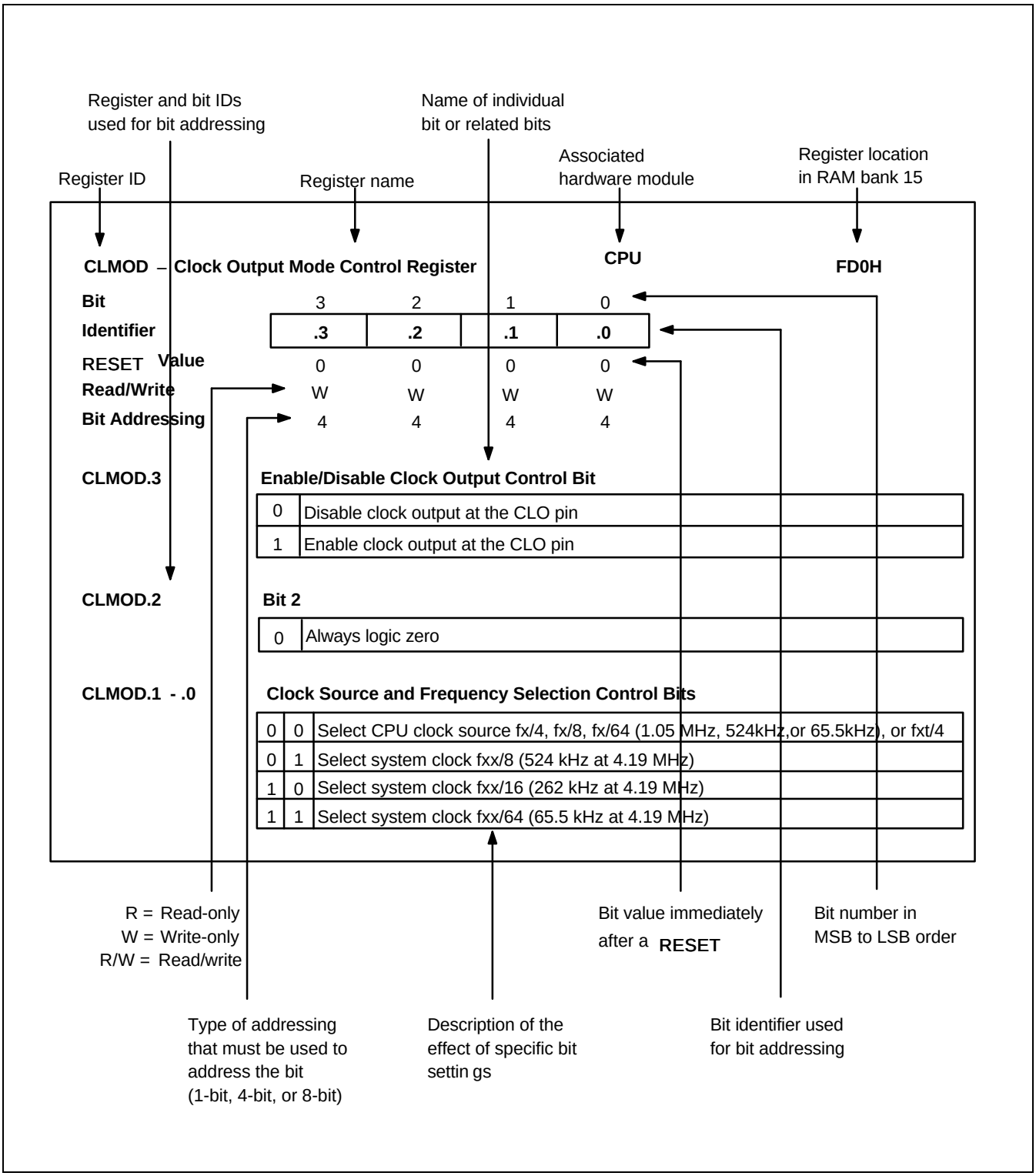


Figure 4-1. Register Description Format

BMOD — Basic Timer Mode Register BT F85H

Bit	3	2	1	0
Identifier	**.3**	**.2**	**.1**	**.0**
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	1/4	4	4	4

BMOD.3 **Basic Timer Restart Bit**

1	Restart basic timer, then clear IRQB flag, BCNT and BMOD.3 to logic zero

BMOD.2 – .0 **Input Clock Frequency and Interrupt Interval Time Control Bits**

0	0	0	Input clock frequency: Interrupt interval time (wait time):	fxx / 2^{12} (1.02 kHz) 2^{20} / fxx (250 ms)
0	1	1	Input clock frequency: Interrupt interval time (wait time):	fxx / 2^{9} (8.18 kHz) 2^{17} / fxx (31.3 ms)
1	0	1	Input clock frequency: Interrupt interval time (wait time):	fxx / 2^{7} (32.7 kHz) 2^{15} / fxx (7.82 ms)
1	1	1	Input clock frequency: Interrupt interval time (wait time):	fxx / 2^{5} (131 kHz) 2^{13} / fxx (1.95 ms)

NOTES:
1. When a RESET occurs, the oscillator stabilization wait time is 31.3 ms (2^{17}/fxx) at 4.19 MHz.
2. 'fxx' is the system clock frequency (assume that fxx is 4.19 MHz).

BUZB — Inverted BUZ Signal Enable Flag Register W/T F92H

Bit	3	2	1	0
Identifier	**TOEB**	**TOE**	**"U"**	**BUZB**
RESET Value	0	0	U	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

TOEB **Timer/Counter 1 Output Enable Flag**

0	Disable inverted Timer/counter 0 clock output at the TCLO0 pin
1	Enable inverted Timer/counter 0 clock output at the TCLO0 pin

TOE **Timer/Counter 0 Output Enable Flag**

0	Disable Timer/counter 0 clock output at the TCLO0 pin
1	Enable Timer/counter 0 clock output at the TCLO0 pin

.1 **Bits 1**

U	This bit is undefined

BUZB **Inverted BUZ Signal Enable Flag**

0	Disable inverted BUZ signal output at the BUZ pin
1	Enable inverted BUZ signal output at the BUZ pin

NOTES:

1. If you set TOE flag to "0", the contents of TOL0 and inverted TOL0 can not be output to the TCLO0 pin and TCLO0 pin, respectively.
2. If you set WMOD.7 to "0", the signals of BUZ and inverted BUZ can not be output to the BUZ pin and BUZ pin, respectively.
3. The “U” means a undefined register bit.

CLMOD — Clock Output Mode Register CPU FD0H

Bit	3	2	1	0
Identifier	**.3**	**"0"**	**.1**	**.0**
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

CLMOD.3 **Enable/Disable Clock Output Control Bit**

0	Disable clock output at the CLO pin
1	Enable clock output at the CLO pin

CLMOD.2 **Bit 2**

0	Always logic zero

CLMOD.1 – .0 **Clock Source and Frequency Selection Control Bits**

0	0	Select CPU clock sourcefx/4, fx/8, fx/64 (1.05 MHz, 524 kHz, or 65.5 kHz), or fxt/4
0	1	Select system clock fxx/8 (524 kHz)
1	0	Select system clock fxx/16 (262 kHz)
1	1	Select system clock fxx/64 (65.5 kHz)

NOTE: 'fxx' is the system clock frequency (assume that fxx is 4.19MHz).

IE0, IRQ0 — INT0 Interrupt Enable/Request Flags CPU FBEH

IE1, IRQ1 — INT1 Interrupt Enable/Request Flags CPU FBEH

Bit	3	2	1	0
Identifier	**IE1**	**IRQ1**	**IE0**	**IRQ0**
RESET Value	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

IE1 **INT1 Interrupt Enable Flag**

0	Disable interrupt requests at the INT1 pin
1	Enable interrupt requests at the INT1 pin

IRQ1 **INT1 Interrupt Request Flag**

–	Generate INT1 interrupt (This bit is set and cleared by hardware when rising or falling edge detected at INT1 pin.)

IE0 **INT0 Interrupt Enable Flag**

0	Disable interrupt requests at the INT0 pin
1	Enable interrupt requests at the INT0 pin

IRQ0 **INT0 Interrupt Request Flag**

–	Generate INT0 interrupt (This bit is set and cleared automatically by hardware when rising or falling edge detected at INT0 pin.)

IE2, IRQ2 — INT2 Interrupt Enable/Request Flags CPU FBFH

Bit	3	2	1	0
Identifier	**"0"**	**"0"**	**IE2**	**IRQ2**
RESET Value	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

.3 – .2 **Bits 3–2**

0	Always logic zero

IE2 **INT2 Interrupt Enable Flag**

0	Disable INT2 interrupt requests at the INT2 pin
1	Enable INT2 interrupt requests at the INT2 pin

IRQ2 **INT2 Interrupt Request Flag**

–	Generate INT2 quasi-interrupt (This bit is set and is not cleared automatically by hardware when a rising or falling edge is detected at INT2. Since INT2 is a quasi-interrupt, IRQ2 flag must be cleared by software.)

IE4, IRQ4 — INT4 Interrupt Enable/Request Flags CPU FB8H

IEB, IRQB — INTB Interrupt Enable/Request Flags CPU FB8H

Bit	3	2	1	0
Identifier	**IE4**	**IRQ4**	**IEB**	**IRQB**
RESET Value	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

IE4 **INT4 Interrupt Enable Flag**

0	Disable interrupt requests at the INT4 pin
1	Enable interrupt requests at the INT4 pin

IRQ4 **INT4 Interrupt Request Flag**

–	Generate INT4 interrupt (This bit is set and cleared automatically by hardware when rising and falling signal edge detected at INT4 pin.)

IEB **INTB Interrupt Enable Flag**

0	Disable INTB interrupt requests
1	Enable INTB interrupt requests

IRQB **INTB Interrupt Request Flag**

–	Generate INTB interrupt (This bit is set and cleared automatically by hardware when reference interval signal received from basic timer.)

IET0, IRQT0 — INTT0 Interrupt Enable/Request Flags CPU FBCH

Bit	3	2	1	0
Identifier	**"0"**	**"0"**	**IET0**	**IRQT0**
RESET Value	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

.3 – .2 **Bits 3–2**

0	Always logic zero

IET0 **INTT0 Interrupt Enable Flag**

0	Disable INTT0 interrupt requests
1	Enable INTT0 interrupt requests

IRQT0 **INTT0 Interrupt Request Flag**

–	Generate INTT0 interrupt (This bit is set and cleared automatically by hardware when contents of TCNT0 and TREF0 registers match.)

IEK, IRQK — INTK Interrupt Enable/Request Flags CPU FBBH

Bit	3	2	1	0
Identifier	**IEK**	**IRQK**	**"0"**	**"0"**
RESET Value	0	0	0	0
Read/Write	R/W	R/W	–	–
Bit Addressing	1/4	1/4	–	–

IEK **INTK Interrupt Enable Flag**

0	Disable interrupt requests at the K0–K3 pins
1	Enable interrupt requests at the K0–K3 pins

IRQK **INTK Interrupt Request Flag**

–	Generate INTK interrupt (This bit is set and cleared automatically by hardware when rising or falling edge detected at K0–K3 pins.)

.1 – .0 **Bits 1 – 0**

0	Always logic zero

IEW, IRQW — INTW Interrupt Enable/Request Flags CPU FBAH

Bit	3	2	1	0
Identifier	**"0"**	**"0"**	**IEW**	**IRQW**
RESET **Value**	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

.3 – .2 **Bits 3–2**

0	Always logic zero

IEW **INTW Interrupt Enable Flag**

0	Disable INTW interrupt requests
1	Enable INTW interrupt requests

IRQW **INTW Interrupt Request Flag**

–	Generate INTW interrupt (This bit is set when the timer interval is set to 0.5 seconds or 3.91 milliseconds.)

NOTE: Since INTW is a quasi-interrupt, the IRQW flag must be cleared by software.

IMOD0 — External Interrupt 0 (INT0) Mode Register CPU FB4H

Bit	3	2	1	0
Identifier	**"0"**	**"0"**	**.1**	**.0**
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

IMOD0.3–.2 **Bit 3–2**

0	Always logic zero

IMOD0.1 – .0 **External Interrupt Mode Control Bits**

0	0	Interrupt request is triggered by a rising signal edge
0	1	Interrupt request is triggered by a falling signal edge
1	0	Interrupt request is triggered by both rising and falling signal edges
1	1	Interrupt request flag (IRQ0) cannot be set to logic one

IMOD1 — External Interrupt 1 (INT1) Mode Register CPU FB5H

Bit	3	2	1	0
Identifier	**"0"**	**"0"**	**"0"**	**IMOD1.0**
RESET **Value**	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

IMOD1.3 – .1 **Bits 3–1**

0	Always logic zero

IMOD1.0 **External Interrupt 1 Edge Detection Control Bit**

0	Rising edge detection
1	Falling edge detection

IMOD2 — External Interrupt 2 (INT2) Mode Register CPU FCFH

Bit	3	2	1	0
Identifier	**"0"**	**"0"**	**"0"**	**IMOD2.0**
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

IMOD2.3 – .1 **Bits 3–1**

0	Always logic zero

IMOD2.0 **External Interrupt 2 Edge Detection Selection Bit**

0	Interrupt request at INT2 pin trigged by rising edge
1	Interrupt request at INT2 pin trigged by falling edge

IMODK — External Key Interrupt Mode Register CPU FCDH, FCCH

Bit	7	6	5	4	3	2	1	0
Identifier	IMODK.7	"0"	"0"	"0"	IMODK.3	IMODK.2	IMODK.1	IMODK.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

IMODK.7 **Bit 7**

0	Falling edge detection
1	Rising edge detection

IMODK.6 – .4 **Bit 6–4**

0	Always logic zero

IMODK.3 **Bit 3**

0	Disable key interrupt
1	Enable edge detection at the K3 pins

IMODK.2 **Bit 2**

0	Disable key interrupt
1	Enable edge detection at the K2 pins

IMODK.1 **Bit 1**

0	Disable key interrupt
1	Enable edge detection at the K1 pins

IMODK.0 **Bit 0**

0	Disable key interrupt
1	Enable edge detection at the K0 pins

NOTES:

1. To generate a key interrupt, all of the selected pins must be configured to input mode.
2. To generate a key interrupt, all of the selected pins must be at input high state for falling edge detection, or all of the selected pins must be at input low state for rising edge detection. If any one of them or more is at input low state or input high state, the interrupt may be not occurred at falling edge or rising edge.
3. To generate a key interrupt, first, configure pull-up resistors or external pull-down resistors. And then, select edge detection and pins by setting IMODK register.

IPR — Interrupt Priority Register

CPU FB2H

Bit	3	2	1	0
Identifier	**IME**	**.2**	**.1**	**.0**
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	1/4	4	4	4

IME **Interrupt Master Enable Bit**

0	Disable all interrupt processing
1	Enable processing for all interrupt service requests

IPR.2 – .0 **Interrupt Priority Assignment Bits**

0	0	0	Process all interrupt requests at low priority
0	0	1	Only INTB and INT4 interrupts are at high priority
0	1	0	Only INT0 interrupt is at high priority
0	1	1	Only INT1 interrupt is at high priority
1	0	0	Not available
1	0	1	Only INTT0 interrupt is at high priority
1	1	0	Not available
1	1	1	Only INTK interrupt is at high priority

LCON — LCD Output Control Register LCD F8EH

Bit	3	2	1	0
Identifier	"0"	.2	.1	.0
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

LCON.3 **Bit 3**

0	Always logic zero

LCON.2 **LCD Dividing Resistor Selection Bit**

0	Normal LCD dividing resistors
1	Diminish LCD dividing resistors to strengthen LCD drive

LCON.1 **Voltage Pumping Circuit On/Off Bit**

0	Only V_{DD} for V_{LC0} (voltage pumping circuit is off)
1	Only BIAS for V_{LC0} (voltage pumping circuit is operated)

LCON.0 **LCD Control Bit**

0	Display off (cut off the voltage dividing resisters)
0	Normal display on

NOTE: When the LCON.2 is selected to "logic one", the leakage current of LCD bias resistor will be increased.

LMOD — LCD Mode Register

LCD F8DH, F8CH

Bit	7	6	5	4	3	2	1	0
Identifier	**.7**	**.6**	**.5**	**.4**	**.3**	**.2**	**.1**	**.0**
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

LMOD.7 **LCD bias Selection Bit**

0	1/4 bias
1	1/5 bias

LMOD.6 **Watch Timer Clock Selection Bit (when main system clock is supplied to the watch timer source.)**

0	f_{LCD} = 4096 Hz when fw = fx/128 (32.768 kHz @ fx = 4.19 MHz)
1	f_{LCD} = 8192 Hz when fw = fx/64 (65.563 kHz @ fx = 4.19 MHz)

LMOD.5 – .4 **LCD Clock Selection Bits**

.5	.4	1/8 duty (COM0–COM7)	1/12 duty (COM0–COM11)	1/16 duty (COM0–COM15)
0	0	fw/2^7 (256 Hz)	fw/2^6 (512 Hz)	fw/2^6 (512 Hz)
0	1	fw/2^6 (512 Hz)	fw/2^5 (1024 Hz)	fw/2^5 (1024 Hz)
1	0	fw/2^5 (1024 Hz)	fw/2^4 (2048 Hz)	fw/2^4 (2048 Hz)
1	1	fw/2^4 (2048 Hz)	fw/2^3 (4096 Hz)	fw/2^3 (4096 Hz)

LMOD.3 – .2 **Duty Selection Bits**

.3	.2	Duty
0	0	1/8 duty (COM0–COM7 select)
0	1	1/12 duty (COM0–COM11 select)
1	0	1/16 duty (COM0–COM15 select)

LMOD.1 – .0 **LCD Display Mode Selection Bits**

0	0	All LCD dots off
0	1	All LCD dots on
1	1	Normal display

PCON — Power Control Register CPU FB3H

Bit	3	2	1	0
Identifier	**.3**	**.2**	**.1**	**.0**
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

PCON.3 – .2 **CPU Operating Mode Control Bits**

0	0	Enable normal CPU operating mode
0	1	Initiate idle power-down mode
1	0	Initiate stop power-down mode

PCON.1 – .0 **CPU Clock Frequency Selection Bits**

0	0	If SCMOD.0 = "0", fx/64; if SCMOD.0 = "1", fxt/64
1	0	If SCMOD.0 = "0", fx/8; if SCMOD.0 = "1", fxt/8
1	1	If SCMOD.0 = "0", fx/4; if SCMOD.0 = "1", fxt/4

NOTE: 'fx' is the main system clock; 'fxt' is the subsystem clock.

PMG1 — Port I/O Mode Register 1 I/O FE9H, FE8H

Bit	7	6	5	4	3	2	1	0
Identifier	PM1.3	PM1.2	PM1.1	PM1.0	PM0.3	PM0.2	PM0.1	PM0.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PM1.3 **P1.3 I/O Mode Selection Flag**

0	Set P1.3 to input mode
1	Set P1.3 to output mode

PM1.2 **P1.2 I/O Mode Selection Flag**

0	Set P1.2 to input mode
1	Set P1.2 to output mode

PM1.1 **P1.1 I/O Mode Selection Flag**

0	Set P1.1 to input mode
1	Set P1.1 to output mode

PM1.0 **P1.0 I/O Mode Selection Flag**

0	Set P1.0 to input mode
1	Set P1.0 to output mode

PM0.3 **P0.3 I/O Mode Selection Flag**

0	Set P0.3 to input mode
1	Set P0.3 to output mode

PM0.2 **P0.2 I/O Mode Selection Flag**

0	Set P0.2 to input mode
1	Set P0.2 to output mode

PM0.1 **P0.1 I/O Mode Selection Flag**

0	Set P0.1 to input mode
1	Set P0.1 to output mode

PM0.0 **P0.0 I/O Mode Selection Flag**

0	Set P0.0 to input mode
1	Set P0.0 to output mode

PNE1 — N-Channel Open-Drain Mode Register 1 I/O FDBH, FDAH

Bit	7	6	5	4	3	2	1	0
Identifier	PNE1.3	PNE1.2	PNE1.1	PNE1.0	PNE0.3	PNE0.2	PNE0.1	PNE0.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PNE1.7 **P1.3 N-Channel Open-Drain Configurable Bit**

0	Configure P1.3 as a push-pull
1	Configure P1.3 as a n-channel open-drain

PNE1.6 **P1.2 N-Channel Open-Drain Configurable Bit**

0	Configure P1.2 as a push-pull
1	Configure P1.2 as a n-channel open-drain

PNE1.5 **P1.1 N-Channel Open-Drain Configurable Bit**

0	Configure P1.1 as a push-pull
1	Configure P1.1 as a n-channel open-drain

PNE1.4 **P1.0 N-Channel Open-Drain Configurable Bit**

0	Configure P1.0 as a push-pull
1	Configure P1.0 as a n-channel open-drain

PNE1.3 **P0.3 N-Channel Open-Drain Configurable Bit**

0	Configure P0.3 as a push-pull
1	Configure P0.3 as a n-channel open-drain

PNE1.2 **P0.2 N-Channel Open-Drain Configurable Bit**

0	Configure P0.2 as a push-pull
1	Configure P0.2 as a n-channel open-drain

PNE1.1 **P0.1 N-Channel Open-Drain Configurable Bit**

0	Configure P0.1 as a push-pull
1	Configure P0.1 as a n-channel open-drain

PNE1.0 **P0.0 N-Channel Open-Drain Configurable Bit**

0	Configure P0.0 as a push-pull
1	Configure P0.0 as a n-channel open-drain

PSW — Program Status Word CPU FB1H, FB0H

Bit	7	6	5	4	3	2	1	0
Identifier	**C**	**SC2**	**SC1**	**SC0**	**IS1**	**IS0**	**EMB**	**ERB**
RESET Value	(1)	0	0	0	0	0	0	0
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W
Bit Addressing	(2)	8	8	8	1/4/8	1/4/8	1/4/8	1/4/8

C **Carry Flag**

0	No overflow or borrow condition exists
1	An overflow or borrow condition does exist

SC2 – SC0 **Skip Condition Flags**

0	No skip condition exists; no direct manipulation of these bits is allowed
1	A skip condition exists; no direct manipulation of these bits is allowed

IS1, IS0 **Interrupt Status Flags**

0	0	Service all interrupt requests
0	1	Service only the high-priority interrupt(s) as determined in the interrupt priority register (IPR)
1	0	Do not service any more interrupt requests
1	1	Undefined

EMB **Enable Data Memory Bank Flag**

0	Restrict program access to data memory to bank 15 (F80H–FFFH) and to the locations 000H–07FH in the bank 0 only
1	Enable full access to data memory banks 0, 1, and 15

ERB **Enable Register Bank Flag**

0	Select register bank 0 as working register area
1	Select register banks 0, 1, 2, or 3 as working register area in accordance with the select register bank (SRB) instruction operand

NOTES:

1. The value of the carry flag after a RESET occurs during normal operation is undefined. If a RESET occurs during power-down mode (IDLE or STOP), the current value of the carry flag is retained.
2. The carry flag can only be addressed by a specific set of 1-bit manipulation instructions. See Section 2 for detailed information.

PUMOD0 — Pull-Up Resistor Mode Register 0 I/O FDDH, FDCH

Bit	7	6	5	4	3	2	1	0
Identifier	PUR1.3	PUR1.2	PUR1.1	PUR1.0	PUR0.3	PUR0.2	PUR0.1	PUR0.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PUR1.3 **Connect/Disconnect Port 1.3 Pull-Up Resistor Control Bit**

0	Disconnect port 1.3 pull-up resistor
1	Connect port 1.3 pull-up resistor

PUR1.2 **Connect/Disconnect Port 1.2 Pull-Up Resistor Control Bit**

0	Disconnect port 1.2 pull-up resistor
1	Connect port 1.2 pull-up resistor

PUR1.1 **Connect/Disconnect Port 1.1 Pull-Up Resistor Control Bit**

0	Disconnect port 1.1 pull-up resistor
1	Connect port 1.1 pull-up resistor

PUR1.0 **Connect/Disconnect Port 1.0 Pull-Up Resistor Control Bit**

0	Disconnect port 1.0 pull-up resistor
1	Connect port 1.0 pull-up resistor

PUR0.3 **Connect/Disconnect Port 0.3 Pull-Up Resistor Control Bit**

0	Disconnect port 0.3 pull-up resistor
1	Connect port 0.3 pull-up resistor

PUR0.2 **Connect/Disconnect Port 0.2 Pull-Up Resistor Control Bit**

0	Disconnect port 0.2 pull-up resistor
1	Connect port 0.2 pull-up resistor

PUR0.1 **Connect/Disconnect Port 0.1 Pull-Up Resistor Control Bit**

0	Disconnect port 0.1 pull-up resistor
1	Connect port 0.1 pull-up resistor

PUR0.0 **Connect/Disconnect Port 0.0 Pull-Up Resistor Control Bit**

0	Disconnect port 0.0 pull-up resistor
1	Connect port 0.0 pull-up resistor

SCMOD — System Clock Mode Control Register CPU FB7H

Bit	3	2	1	0
Identifier	**.3**	**.2**	**"0"**	**.0**
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	1	1	1	1

SCMOD.3 **Bit 3**

0	Enable main system clock
1	Disable main system clock

SCMOD.2 **Bit 2**

0	Enable sub system clock
1	Disable sub system clock

SCMOD.1 **Bit 1**

0	Always logic zero

SCMOD.0 **Bit 0**

0	Select main system clock
1	Select sub system clock

NOTES:

1. Sub-oscillation goes into stop mode only by SCMOD.2. PCON which revokes stop mode cannot stop the sub-oscillation.
2. You can use SCMOD.2 as follows (ex; after data bank was used, a few minutes have passed):
   Main operation → sub-operation → sub-idle (LCD on, after a few minutes later without any external input) → sub-operation → main operation → SCMOD.2 = 1 → main stop mode (LCD off).
3. SCMOD bit3–0 can not be modified simultaneously by a 4-bit instruction; They can only be modified by separate 1-bit instructions.

TMOD0 — Timer/Counter 0 Mode Register T/C0 F91H, F90H

Bit	7	6	5	4	3	2	1	0
Identifier	**"0"**	**.6**	**.5**	**.4**	**.3**	**.2**	**"0"**	**"0"**
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	1/8	8	8	8

TMOD0.7 **Bit 7**

0	Always logic zero

TMOD0.6 – .4 **Timer/Counter 0 Input Clock Selection Bits**

0	0	0	Not available
0	0	1	Not available
1	0	0	Select clock: fxx/2^{10} (4.09 kHz)
1	0	1	Select clock: fxx/2^{6} (65.5 kHz)
1	1	0	Select clock: fxx/2^{4} (262 kHz)
1	1	1	Select clock: fxx (4.19 MHz)

NOTE: 'fxx' is the system clock frequency (assume that fxx is 4.19 MHz).

TMOD0.3 **Clear Counter and Resume Counting Control Bit**

1	Clear TCNT0, IRQT0, and TOL0 and resume counting immediately (This bit is cleared automatically when counting starts)

TMOD0.2 **Enable/Disable Timer/Counter 0 Bit**

0	Disable timer/counter 0; retain TCNT0 contents
1	Enable timer/counter 0

TMOD0.1 **Bit 1**

0	Always logic zero

TMOD0.0 **Bit 0**

0	Always logic zero

TOE, TOEB — Timer Output Enable Flag Register T/C F92H

Bit	3	2	1	0
Identifier	**TOEB**	**TOE**	**"U"**	**BUZB**
RESET Value	0	0	U	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

TOEB **Timer/Counter 1 Output Enable Flag**

0	Disable inverted Timer/counter 0 clock output at the TCLO0 pin
1	Enable inverted Timer/counter 0 clock output at the TCLO0 pin

TOE **Timer/Counter 0 Output Enable Flag**

0	Disable Timer/counter 0 clock output at the TCLO0 pin
1	Enable Timer/counter 0 clock output at the TCLO0 pin

.1 **Bit 1**

U	This bit is undefined

BUZB **Inverted BUZ Signal Enable Flag**

0	Disable inverted BUZ signal output at the BUZ pin
1	Enable inverted BUZ signal output at the BUZ pin

NOTES:

1. If you set TOE flag to "0", the contents of TOL0 and inverted TOL0 can not be output to the TCLO0 pin and TCLO0 pin, respectively.
2. If you set WMOD.7 to "0", the signals of BUZ and inverted BUZ cannot be output to the BUZ pin and BUZ pin, respectively.
3. The “U” means a undefined register bit.

WDFLAG — Watch-Dog Timer's Counter Clear Flag WT F9AH.3

Bit	3	2	1	0
Identifier	**WDTCF**	**"0"**	**"0"**	**"0"**
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	1/4	1/4	1/4	1/4

WDTCF **Watch-dog Timer's Counter Clear Bit**

0	–
1	Clear the WDT's counter to zero and restart the WDT's counter

WDFLAG.2 – .0 **Bit2 – 0**

0	Always logic zero

WDMOD — Watch-Dog Timer Mode Control Register WT F99H, F98H

Bit	7	6	5	4	3	2	1	0
Identifier	**.7**	**.6**	**.5**	**.4**	**.3**	**.2**	**.1**	**.0**
RESET Value	1	0	1	0	0	1	0	1
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

WMOD.7 – .0 **Watch-Dog Timer Enable/Disable Control**

0	1	0	1	1	0	1	0	Disable watch-dog timer function
Other Values								Enable watch-dog timer function

WMOD — Watch Timer Mode Register WT F89H, F88H

Bit	7	6	5	4	3	2	1	0
Identifier	.7	"0"	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	(NOTE)	0	0	0
Read/Write	W	W	W	W	R	W	W	W
Bit Addressing	8	8	8	8	1	8	8	8

WMOD.7 **Enable/Disable Buzzer Output Bit**

0	Disable buzzer (BUZ) signal output at the BUZ pin
1	Enable buzzer (BUZ) signal output at the BUZ pin

WMOD.6 **Bit 6**

0	Always logic zero

WMOD.5 – .4 **Output Buzzer Frequency Selection Bits**

0	0	2 kHz buzzer (BUZ/BUZ) signal output
0	1	4 kHz buzzer (BUZ/BUZ) signal output
1	0	8 kHz buzzer (BUZ/BUZ) signal output
1	1	16 kHz buzzer (BUZ/BUZ) signal output

WMOD.3 **XT_{in} Input Level Control Bit**

0	Input level to XT_{in} pin is low; 1-bit read-only addressable for tests
1	Input level to XT_{in} pin is high; 1-bit read-only addressable for tests

WMOD.2 **Enable/Disable Watch Timer Bit**

0	Disable watch timer and clear frequency dividing circuits
1	Enable watch timer

WMOD.1 **Watch Timer Speed Control Bit**

0	Normal speed; set IRQW to 0.5 seconds
1	High-speed operation; set IRQW to 3.91 ms

WMOD.0 **Watch Timer Clock Selection Bit**

0	Select main system clock (fx)/128 as the watch timer clock
1	Select a subsystem clock as the watch timer clock

NOTES:

1. RESET sets WMOD.3 to the current input level of the subsystem clock, XTin. If the input level is high, WMOD.3 is set to logic one; if low, WMOD.3 is cleared to zero along with all the other bits in the WMOD register.
2. If you set WMOD.7 to "0", the signals of BUZ and inverted BUZ can not be output to the BUZ pin and BUZ pin, respectively.

6 OSCILLATOR CIRCUITS

OVERVIEW

The S3C7295/P7295 microcontroller has two oscillator circuits: a main system clock circuit, and a subsystem clock circuit. The CPU and peripheral hardware operate on the system clock frequency supplied through these circuits. Specifically, a clock pulse is required by the following peripheral modules:

— LCD controller
— Basic timer
— Timer/counter
— Watch timer
— Clock output circuit

CPU Clock Notation

In this document, the following notation is used for descriptions of the CPU clock:

fx Main system clock
fxt Subsystem clock
fxx Selected system clock

Clock Control Registers

When the system clock mode control register, SCMOD, and the power control register, PCON, are both cleared to zero after RESET, the normal CPU operating mode is enabled, a main system clock is selected as fx/64, and main system clock oscillation is initiated.,

The PCON is used to select normal CPU operating mode or one of two power down mode stop or idle. Bits 3 and 2 of the PCON register can be manipulated by STOP or IDLE instruction to engage stop or idle power down mode.

The SCMOD, lets you select the main system clock (fx) or a subsystem clock (fxt) as the CPU clock and start (or stop) main/sub system clock oscillation. The resulting clock source, either main system clock or subsystem clock, is referred to the selected system clock (fxx).

The main system clock is selected and oscillation started when all SCMOD bits are cleared to “0”. By setting SCMOD.3, SCMOD.2 and SCMOD.0 to different values, you can select a subsystem clock source and start or stop main/sub system clock oscillation. To stop main system clock oscillation, you must use the STOP instruction (assuming the main system clock is selected) or manipulate SCMOD.3 to "1" (assuming the sub system clock is selected).

The main system clock frequencies can be divided by 4, 8, or 64 and a subsystem clock frequencies can only be divided by 4. By manipulating PCON bits 1 and 0, you select one of the following frequencies as the CPU clock.

fx/4, fxt/4, fx/8, fx/64

Using a Subsystem Clock

If a subsystem clock is being used as the selected system clock, the idle power-down mode can be initiated by executing an IDLE instruction.

The watch timer, buzzer and LCD display operate normally with a subsystem clock source, since they operate at very low speed (as low as 122 µs at 32.768 kHz) and with very low power consumption.

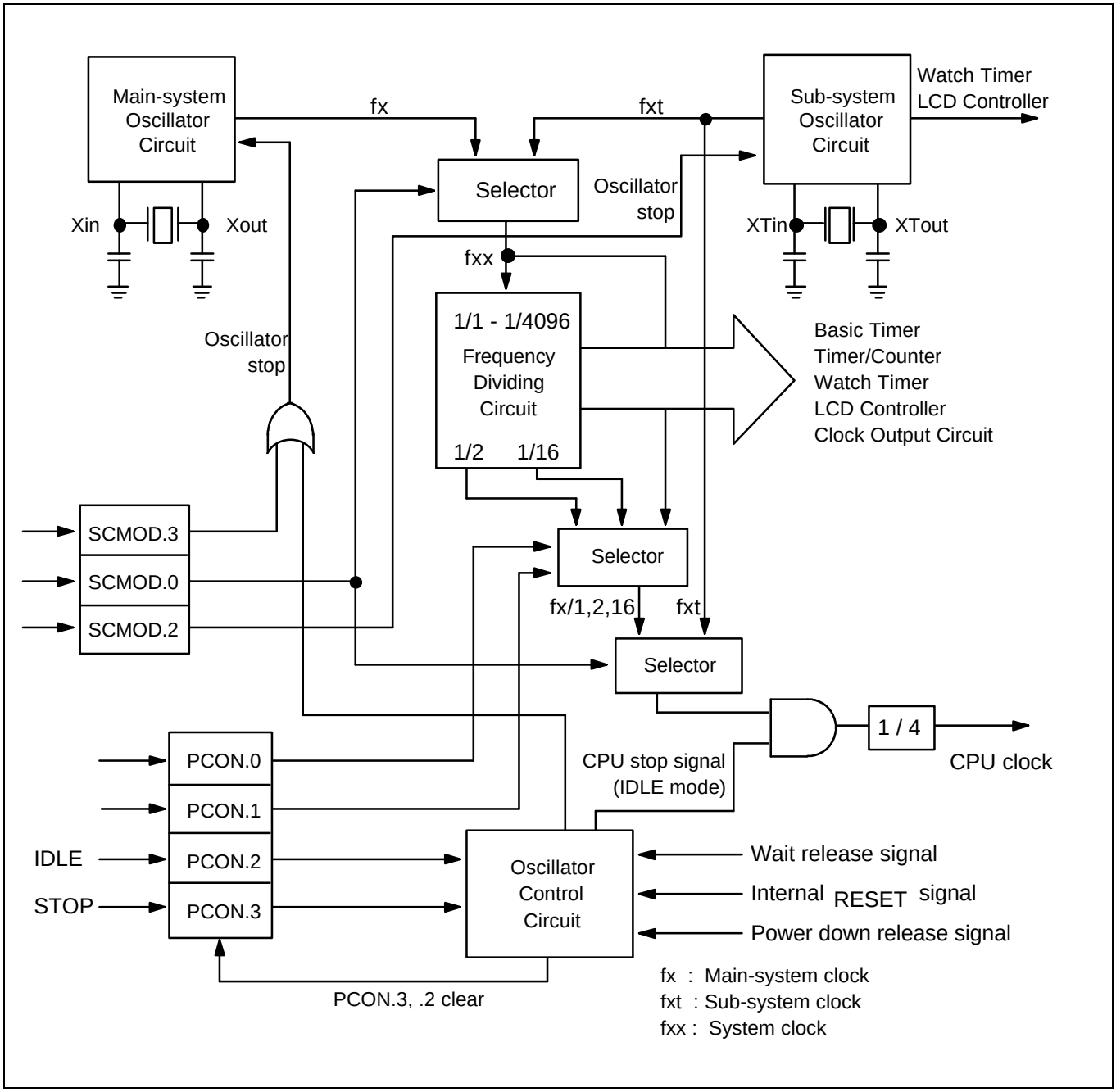


Figure 6-1. Clock Circuit Diagram

MAIN SYSTEM OSCILLATOR CIRCUITS

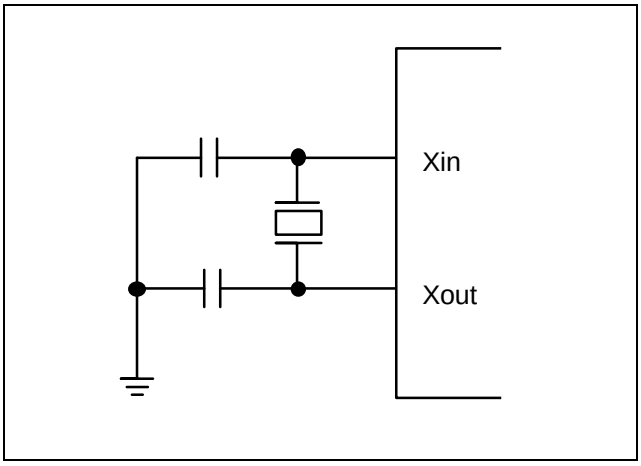


Figure 6-2. Crystal/Ceramic Oscillator

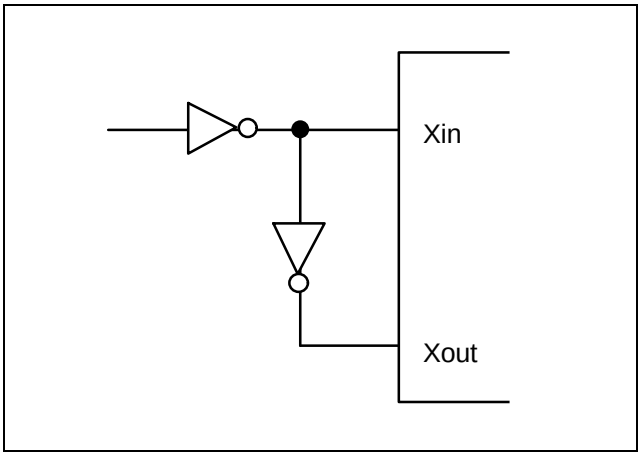


Figure 6-3. External Oscillator

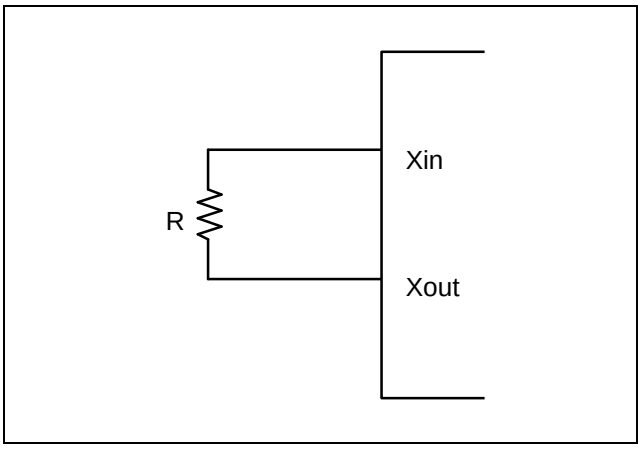


Figure 6-4. RC Oscillator

SUBSYSTEM OSCILLATOR CIRCUITS

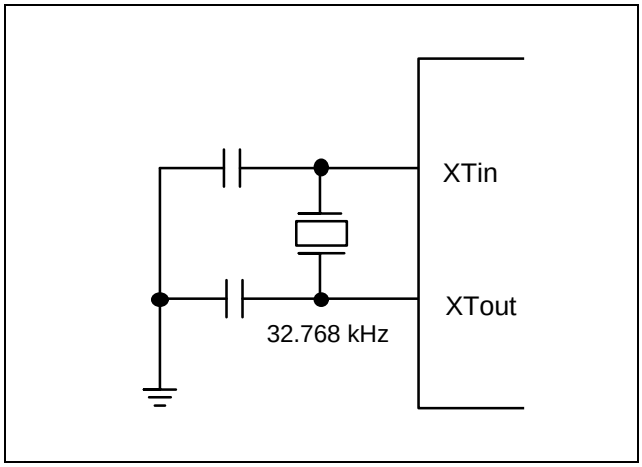


Figure 6-5. Crystal/Ceramic Oscillator

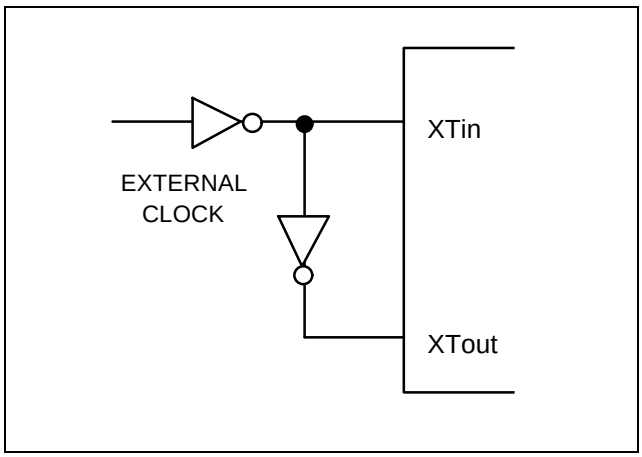


Figure 6-6. External Oscillator

POWER CONTROL REGISTER (PCON)

The power control register, PCON, is a 4-bit register that is used to select the CPU clock frequency and to control CPU operating and power-down modes. PCON can be addressed directly by 4-bit write instructions or indirectly by the instructions IDLE and STOP.

FB3H	PCON.3	PCON.2	PCON.1	PCON.0

PCON bits 3 and 2 are addressed by the STOP and IDLE instructions, respectively, to engage the idle and stop power-down modes. Idle and stop modes can be initiated by these instruction despite the current value of the enable memory bank flag (EMB). PCON bits 1 and 0 are used to select a specific system clock frequency. There are two basic choices:

— Main system clock (fx) or subsystem clock (fxt);
— Divided fx/4, 8, 64 or fxt/4 clock frequency.

PCON.1 and PCON.0 settings are also connected with the system clock mode control register, SCMOD. If SCMOD.0 = "0" the main system clock is always selected by the PCON.1 and PCON.0 setting; if SCMOD.0 = "1" the subsystem clock is selected.

RESET sets PCON register values (and SCMOD) to logic zero: SCMOD.3 and SCMOD.0 select the main system clock (fx) and start clock oscillation; PCON.1 and PCON.0 divide the selected fx frequency by 64, and PCON.3 and PCON.2 enable normal CPU operating mode.

Table 6-1. Power Control Register (PCON) Organization

PCON Bit Settings		Resulting CPU Operating Mode
PCON.3	PCON.2	
0	0	Normal CPU operating mode
0	1	Idle power-down mode
1	0	Stop power-down mode

PCON Bit Settings		Resulting CPU Clock Frequency	
PCON.1	PCON.0	If SCMOD.0 = "0"	If SCMOD.0 = "1"
0	0	fx/64	fxt/4
1	0	fx/8	
1	1	fx/4	

☞ PROGRAMMING TIP — Setting the CPU Clock

To set the CPU clock to 0.95 µs at 4.19 MHz:

```
BITS    EMB
SMB     15
LD      A,#3H
LD      PCON,A
```

INSTRUCTION CYCLE TIMES

The unit of time that equals one machine cycle varies depending on whether the main system clock (fx) or a subsystem clock (fxt) is used, and on how the oscillator clock signal is divided (by 4, 8, or 64). Table 6-2 shows corresponding cycle times in microseconds.

Table 6-2. Instruction Cycle Times for CPU Clock Rates

Selected CPU Clock	Resulting Frequency	Oscillation Source	Cycle Time (μsec)
fx/64	65.5 kHz	fx = 4.19 MHz	15.3
fx/8	524.0 kHz		1.91
fx/4	1.05 MHz		0.95
fxt/4	8.19 kHz	fxt = 32.768 kHz	122.0

SYSTEM CLOCK MODE REGISTER (SCMOD)

The system clock mode register, SCMOD, is a 4-bit register that is used to select the CPU clock and to control main and sub-system clock oscillation. The SCMOD is mapped to the RAM address FB7H.

The main clock oscillation is stopped by setting SCMOD.3 when the clock source is subsystem clock and subsystem clock can be stopped by setting SCMOD.2 when the clock source is main system clock. SCMOD.0, SCMOD.3 cannot be simultaneously modified.

The subsystem clock is stopped only by setting SCMOD.2, and PCON which revokes stop mode cannot stop the subsystem clock. The stop of subsystem clock is released by RESET when the selected system clock is main system clock or subsystem clock and is released by setting SCMOD.2 when the selected system clock is main system clock.

RESET clears all SCMOD values to logic zero, selecting the main system clock (fx) as the CPU clock and starting clock oscillation. The reset value of the SCMOD is “0”

SCMOD.0, SCMOD.2, and SCMOD.3 bits can be manipulated by 1-bit write instructions (In other words, SCMOD.0, SCMOD.2, and SCMOD.3 cannot be modified simultaneously by a 4-bit write).
Bit 1 is always logic zero.

FB7H	SCMOD.3	SCMOD.2	"0"	SCMOD.0

A subsystem clock (fxt) can be selected as the system clock by manipulating the SCMOD.3 and SCMOD.0 bit settings. If SCMOD.3 = "0" and SCMOD.0 = "1", the subsystem clock is selected and main system clock oscillation continues. If SCMOD.3 = "1" and SCMOD.0 = "1", fxt is selected, but main system clock oscillation stops.

Even if you have selected fx as the CPU clock, setting SCMOD.3 to "1" will stop main system clock oscillation, and malfunction may be occured. To operate safely, main system clock should be stopped by a stop instruction is main system clock mode.

Table 6-3. System Clock Mode Register (SCMOD) Organization

SCMOD Register Bit Settings		Resulting Clock Selection	
SCMOD.3	**SCMOD.0**	**CPU Clock Source**	**fx Oscillation**
0	0	fx	On
0	1	fxt	On
1	1	fxt	Off

NOTE: fx: Main–system clock, fxt: Sub–system clock

Table 6-4. SCMOD.2 for Sub-oscillation On/Off

SCMOD.2	Sub-oscillation on/off
0	Enable sub system clock
1	Disable sub system clock

NOTE: You can use SCMOD.2 as follows (ex; after data bank was used, a few minutes have passed): Main operation → sub-operation → sub-idle (LCD on, after a few minutes later without any external input) → sub-operation → main operation → SCMOD.2 = 1 → main stop mode (LCD off).

Table 6-5. Main/Sub Oscillation Stop Mode

Mode	Condition	Method to issue Osc Stop	Osc Stop Release Source (2)
Main Oscillation STOP Mode	Main oscillator runs. Sub oscillator runs (stops). System clock is the main oscillation clock.	STOP instruction: Main oscillator stops. CPU is in idle mode. Sub oscillator still runs (stops).	Interrupt and reset: After releasing stop mode, main oscillation starts and oscillation stabilization time is elapsed. And then the CPU operates. Oscillation stabilization time is 1/ {256 x BT clock (fx)}.
		When SCMOD.3 is set to “1” (1), main oscillator stops, halting the CPU operation. Sub oscillator still runs (stops).	Reset: Interrupt can’t start the main oscillation. Therefore, the CPU operation can never be restarted.
	Main oscillator runs. Sub oscillator runs. System clock is the sub oscillation clock.	STOP instruction stops (1) main oscillator and the CPU is left in idle mode. Sub oscillator still runs.	BT overflow, interrupt, and reset: After the overflow of basic timer [1/ {256 x BT clock (fxt)}], CPU operation and main oscillation automatically start.
		When SCMOD.3 is set to “1”, main oscillator stops. The CPU, however, would still operate. Sub oscillator still runs.	Set SCMOD.3 to “0” or reset
Sub oscillation STOP Mode	Main oscillator runs. Sub oscillator runs. System clock is the main oscillation clock.	When SCMOD.2 to “1”, main oscillator and the CPU would still operate, while sub oscillator stops.	Set SCMOD.2 to “0” or reset
	Main oscillator runs (stops). Sub oscillator runs. System clock is the sub oscillation clock.	When SCMOD.2 to “1”, sub oscillator stops, halting the CPU operation. Main oscillator still runs (stops).	Reset

NOTES:
1. This mode must not be used.
2. Oscillation stabilization time by interrupt is 1/(256 x BT clocks). Oscillation stabilization time by a reset is 31.3ms at 4.19Mhz, main oscillation clock.

SWITCHING THE CPU CLOCK

Together, bit settings in the power control register, PCON, and the system clock mode register, SCMOD, determine whether a main system or a subsystem clock is selected as the CPU clock, and also how this frequency is to be divided. This makes it possible to switch dynamically between main and subsystem clocks and to modify operating frequencies.

SCMOD.3, SCMOD.2, and SCMOD.0 select the main system clock (fx) or a subsystem clock (fxt) and start or stop main or subsystem clock oscillation. PCON.1 and PCON.0 control the frequency divider circuit, and divide the selected fx clock by 4,8,64 or fxt clock by 4.

NOTE

A clock switch operation does not go into effect immediately when you make the SCMOD and PCON register modifications — the previously selected clock continues to run for a certain number of machine cycles.

For example, you are using the default CPU clock (normal operating mode and a main system clock of fx/64) and you want to switch from the fx clock to a subsystem clock and to stop the main system clock. To do this, you first need to set SCMOD.0 to "1". This switches the clock from fx to fxt but allows main system clock oscillation to continue. Before the switch actually goes into effect, a certain number of machine cycles must elapse. After this time interval, you can then disable main system clock oscillation by setting SCMOD.3 to "1".

This same 'stepped' approach must be taken to switch from a subsystem clock to the main system clock: First, clear SCMOD.3 to "0" to enable main system clock oscillation. Then, after a certain number of machine cycles has elapsed, select the main system clock by clearing all SCMOD values to logic zero.

Following a RESET, CPU operation starts with the lowest main system clock frequency of 15.3 µsec at 4.19 MHz after the standard oscillation stabilization interval of 31.3 ms has elapsed. Table 6-4 details the number of machine cycles that must elapse before a CPU clock switch modification goes into effect.

Table 6-6. Elapsed Machine Cycles During CPU Clock Switch

<table>
<tr><td colspan="2" rowspan="2">AFTER
BEFORE</td><td colspan="4">SCMOD.0 = 0</td><td colspan="2">SCMOD.0 = 1</td></tr>
<tr><td>PCON.1 = 0</td><td>PCON.0 = 0</td><td>PCON.1 = 1</td><td>PCON.0 = 0</td><td>PCON.1 = 1</td><td>PCON.0 = 1</td></tr>
<tr><td rowspan="6">SCMOD.0 = 0</td><td>PCON.1 = 0</td><td colspan="2" rowspan="2">N/A</td><td colspan="2" rowspan="2">1 MACHINE CYCLE</td><td colspan="2" rowspan="2">1 MACHINE CYCLE</td><td rowspan="2">N/A</td></tr>
<tr><td>PCON.0 = 0</td></tr>
<tr><td>PCON.1 = 1</td><td colspan="2" rowspan="2">8 MACHINE CYCLES</td><td colspan="2" rowspan="2">N/A</td><td colspan="2" rowspan="2">1 MACHINE CYCLES</td><td rowspan="2">N/A</td></tr>
<tr><td>PCON.0 = 0</td></tr>
<tr><td>PCON.1 = 1</td><td colspan="2" rowspan="2">16 MACHINE CYCLES</td><td colspan="2" rowspan="2">1 MACHINE CYCLES</td><td colspan="2" rowspan="2">N/A</td><td rowspan="2">fx/4fxt</td></tr>
<tr><td>PCON.0 = 1</td></tr>
<tr><td>SCMOD.0 = 1</td><td></td><td colspan="2">N/A</td><td colspan="2">N/A</td><td colspan="2">1MACHINE CYCLES</td><td>N/A</td></tr>
</table>

NOTES:
1. Even if oscillation is stopped by setting SCMOD.3 during main system clock operation, the stop mode is not entered.
2. Since the Xin input is connected internally to V_{SS} to avoid current leakage due to the crystal oscillator in stop mode, do not set SCMOD.3 to "1" or do not use STOP instruction when an external clock is used as the main system clock.
3. When the system clock is switched to the subsystem clock, it is necessary to disable any interrupts which may occur during the time intervals shown in Table 6-4.
4. 'N/A' means 'not available'.
5. fx: Main–system clock, fxt: Sub–system clock, M/C: Machine Cycle.
   When fx is 4.19 MHz, and fxt is 32.768 kHz.

☞ PROGRAMMING TIP — Switching Between Main System And Subsystem Clock

1. Switch from the main system clock to the subsystem clock:

```
MA2SUB   BITS   SCMOD.0          ;  Switches to subsystem clock
         CALL   DLY80            ;  Delay 80 machine cycles
         BITS   SCMOD.3          ;  Stop the main system clock
         RET
DLY80    LD     A,#0FH
DEL1     NOP
         NOP
         DECS   A
         JR     DEL1
         RET
```

2. Switch from the subsystem clock to the main system clock:

```
SUB2MA   BITR   SCMOD.3          ;  Start main system clock oscillation
         CALL   DLY80            ;  Delay 160 machine cycles
         CALL   DLY80
         BITR   SCMOD.0          ;  Switch to main system clock
         RET
```

CLOCK OUTPUT MODE REGISTER (CLMOD)

The clock output mode register, CLMOD, is a 4-bit register that is used to enable or disable clock output to the CLO pin and to select the CPU clock source and frequency. CLMOD is addressable by 4-bit write instructions only.

FD0H	CLMOD.3	"0"	CLMOD.1	CLMOD.0

RESET clears CLMOD to logic zero, which automatically selects the CPU clock as the clock source (without initiating clock oscillation), and disables clock output.

CLMOD.3 is the enable/disable clock output control bit; CLMOD.1 and CLMOD.0 are used to select one of four possible clock sources and frequencies: normal CPU clock, fxx/8, fxx/16, or fxx/64.

Table 6-7. Clock Output Mode Register (CLMOD) Organization

CLMOD Bit Settings		Resulting Clock Output	
CLMOD.1	**CLMOD.0**	**Clock Source**	**Frequency**
0	0	CPU clock (fx/4, fx/8, fx/64, or fxt/4)	1.05 MHz, 524 kHz, 65.5 kHz, 8.19 kHz
0	1	fxx/8	524 kHz
1	0	fxx/16	262 kHz
1	1	fxx/64	65.5 kHz

CLMOD.3	Result of CLMOD.3 Setting
0	Disable clock output at the CLO pin
1	Enable clock output at the CLO pin

NOTE: Frequencies assume the fxx, fx = 4.19 MHz and fxt = 32.768 kHz

CLOCK OUTPUT CIRCUIT

The clock output circuit, used to output clock pulses to the CLO pin, has the following components:

— 4-bit clock output mode register (CLMOD)
— Clock selector
— Port mode flag
— CLO output pin

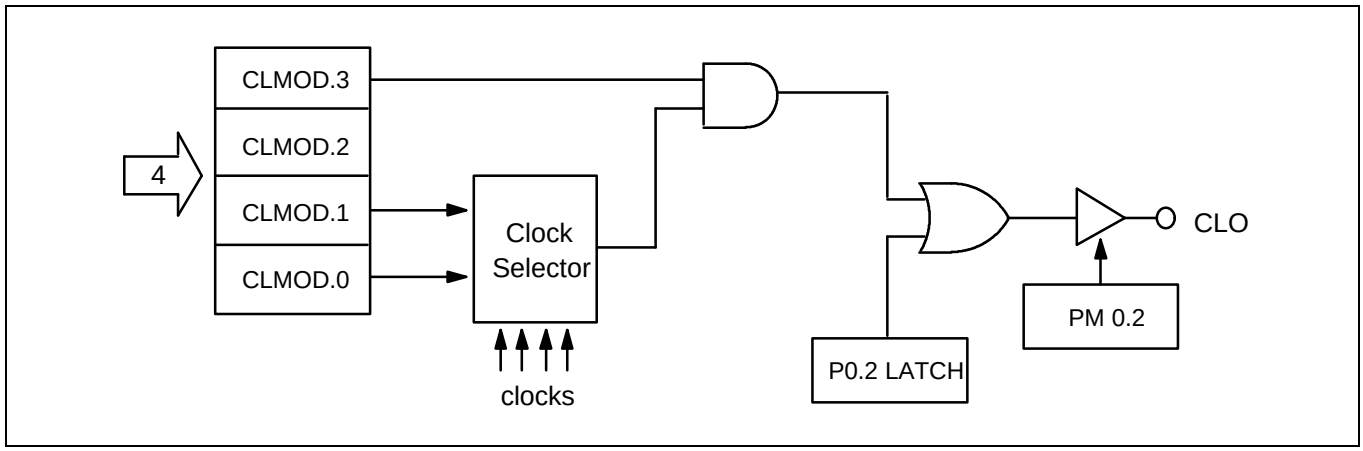


Figure 6-7. CLO Output Pin Circuit Diagram

CLOCK OUTPUT PROCEDURE

The procedure for outputting clock pulses to the CLO pin may be summarized as follows:

1. Disable clock output by clearing CLMOD.3 to logic zero.
2. Set the clock output frequency (CLMOD.1, CLMOD.0).
3. Load "0" to the output latch of the CLO pin.
4. Set the port mode flag to output mode.
5. Load "0" to the BUZB register bit.
6. Enable clock output by setting CLMOD.3 to logic one.

☞ PROGRAMMING TIP — CPU CLOCK OUTPUT TO THE CLO PIN

To output the CPU clock to the CLO pin:

```
BITS    EMB
SMB     15
LD      EA,#04H
LD      PMG1,EA          ;  P0.2 ← Output mode
BITR    P0.2             ;  Clear the CLO pin output latch
LD      A,#9H
LD      CLMOD,A
```

7 INTERRUPTS

OVERVIEW

The S3C7295 `s interrupt control circuit has five functional components:

— Interrupt enable flags (IEx)
— Interrupt request flags (IRQx)
— Interrupt master enable register (IME)
— Interrupt priority register (IPR)
— Power-down release signal circuit

Three kinds of interrupts are supported:

— Internal interrupts generated by on-chip processes
— External interrupts generated by external peripheral devices
— Quasi-interrupts used for edge detection and as clock sources

Table 7-1. Interrupt Types and Corresponding Port Pin(s)

Interrupt Type	Interrupt Name	Corresponding Port Pins
External interrupts	INT0, INT1, INT4, INTK	P1.0, P1.1, P1.3, P0 (K0-K3)
Internal interrupts	INTB, INTT0	Not applicable
Quasi-interrupts	INT2	P1.2
	INTW	Not applicable

Vectored Interrupts

Interrupt requests may be processed as vectored interrupts in hardware, or they can be generated by program software. A vectored interrupt is generated when the following flags and register settings, corresponding to the specific interrupt (INTn) are set to logic one:

— Interrupt enable flag (IEx)
— Interrupt master enable flag (IME)
— Interrupt request flag (IRQx)
— Interrupt status flags (IS0, IS1)
— Interrupt priority register (IPR)

If all conditions are satisfied for the execution of a requested service routine, the start address of the interrupt is loaded into the program counter and the program starts executing the service routine from this address.

EMB and ERB flags for RAM memory banks and registers are stored in the vector address area of the ROM during interrupt service routines. The flags are stored at the beginning of the program with the VENT instruction. The initial flag values determine the vectors for resets and interrupts. Enable flag values are saved during the main routine, as well as during service routines. Any changes that are made to enable flag values during a service routine are not stored in the vector address.

When an interrupt occurs, the EMB and the ERB flags before the interrupt is initiated are saved along with the program status word (PSW), and the EMB and the ERB flags for the interrupt are fetched from the respective vector address. Then, if necessary, you can modify the enable flags during the interrupt service routine. When the interrupt service routine is returned to the main routine by the IRET instruction, the original values saved in the stack are restored and the main program continues program execution with these values.

Software-Generated Interrupts

To generate an interrupt request from software, the program manipulates the appropriate IRQx flag. When the interrupt request flag value is set, it is retained until all other conditions for the vectored interrupt have been met, and the service routine can be initiated.

Multiple Interrupt

By manipulating the two interrupt status flags (IS0 and IS1), you can control service routine initialization and thereby process multiple interrupts simultaneously.

If more than four interrupts are being processed at one time, you can avoid possible loss of working register data by using the PUSH RR instruction to save register contents to the stack before the service routines are executed in the same register bank. When the routines have executed successfully, you can restore the register contents from the stack to working memory using the POP instruction.

Power-Down Mode Release

An interrupt can be used to release power-down mode (stop or idle). Interrupts for power-down mode release are initiated by setting the corresponding interrupt enable flag. Even if the IME flag is cleared to zero, power-down mode will be released by an interrupt request signal when the interrupt enable flag has been set. In such cases, the interrupt routine will not be executed since IME = "0".

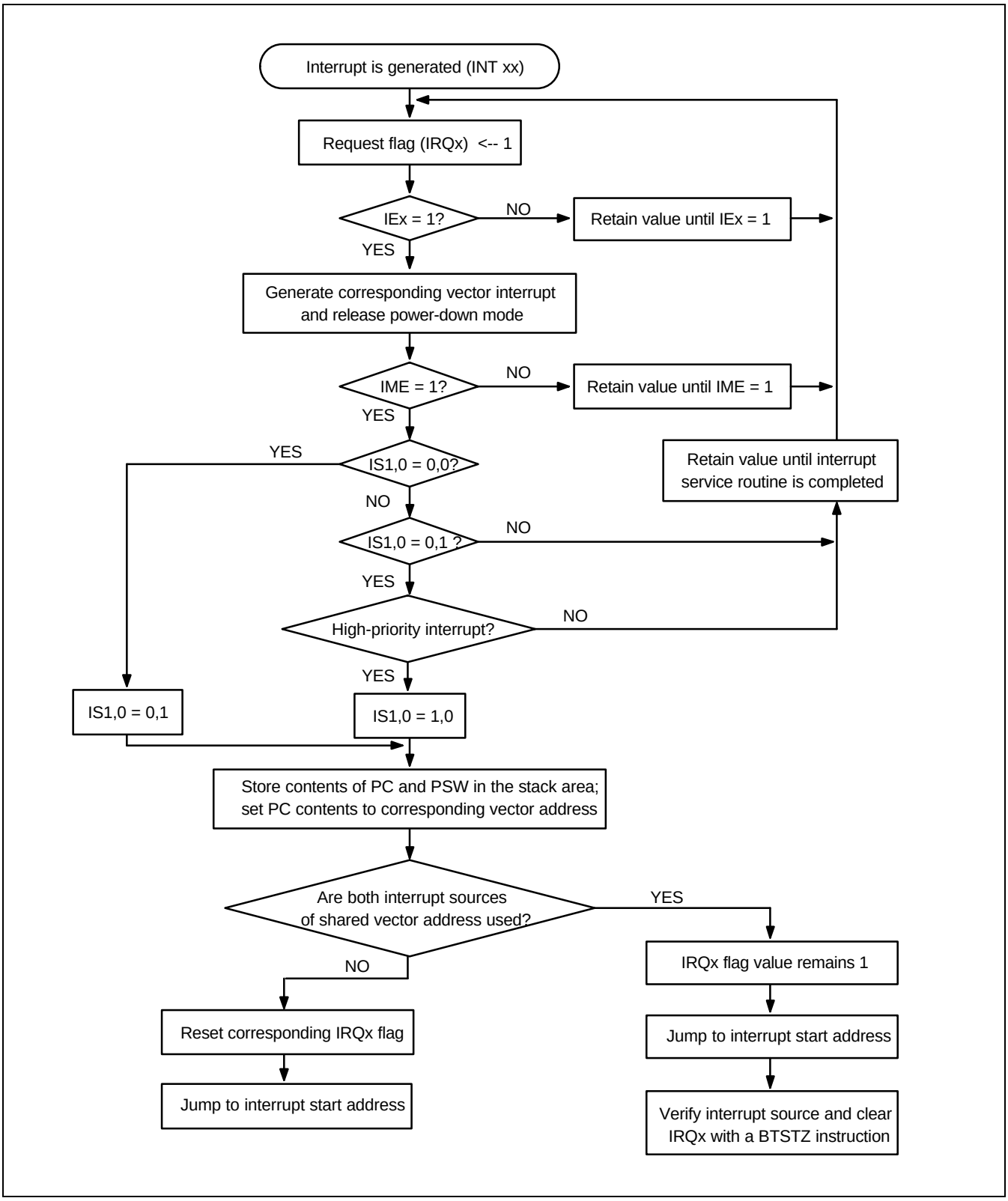


Figure 7-1. Interrupt Execution Flowchart

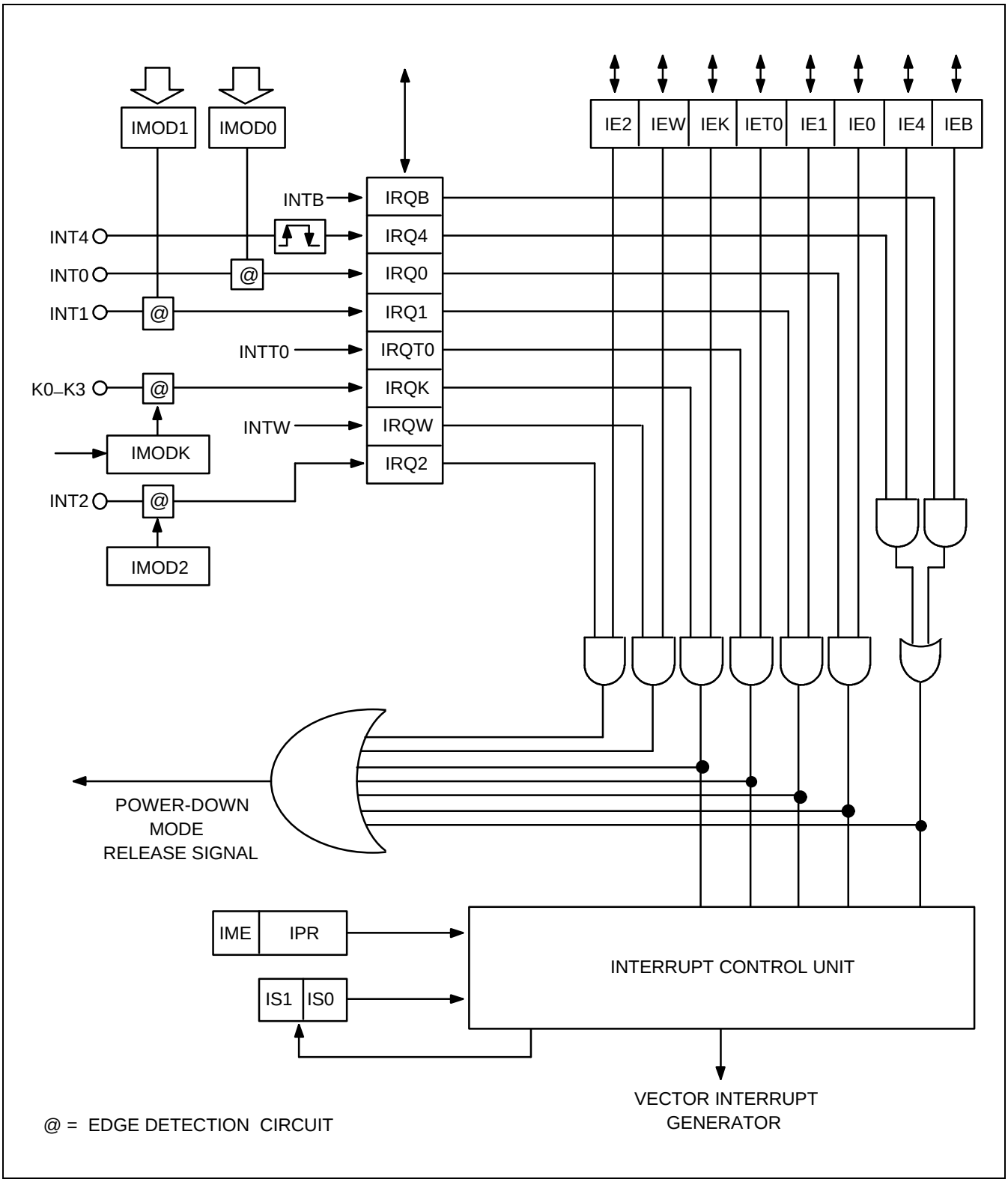


Figure 7-2. Interrupt Control Circuit Diagram

MULTIPLE INTERRUPTS

The interrupt controller can service multiple interrupts in two ways: as two-level interrupts, where either all interrupt requests or only those of highest priority are serviced, or as multi-level interrupts, when the interrupt service routine for a lower-priority request is accepted during the execution of a higher priority routine.

Two-Level Interrupt Handling

Two-level interrupt handling is the standard method for processing multiple interrupts. When the IS1 and IS0 bits of the PSW (FB0H.3 and FB0H.2, respectively) are both logic zero, program execution mode is normal and all interrupt requests are serviced (see Figure 7-3).
Whenever an interrupt request is accepted, IS1 and IS0 are incremented by one and the values are stored in the stack along with the other PSW bits. After the interrupt routine has been serviced, the modified IS1 and IS0 values are automatically restored from the stack by an IRET instruction.
IS0 and IS1 can be manipulated directly by 1-bit write instructions, regardless of the current value of the enable memory bank flag (EMB). Before you can modify an interrupt service flag, however, you must first disable interrupt processing with a DI instruction.
When IS1 = "0" and IS0 = "1", all interrupt service routines are inhibited except for the highest priority interrupt currently defined by the interrupt priority register (IPR).

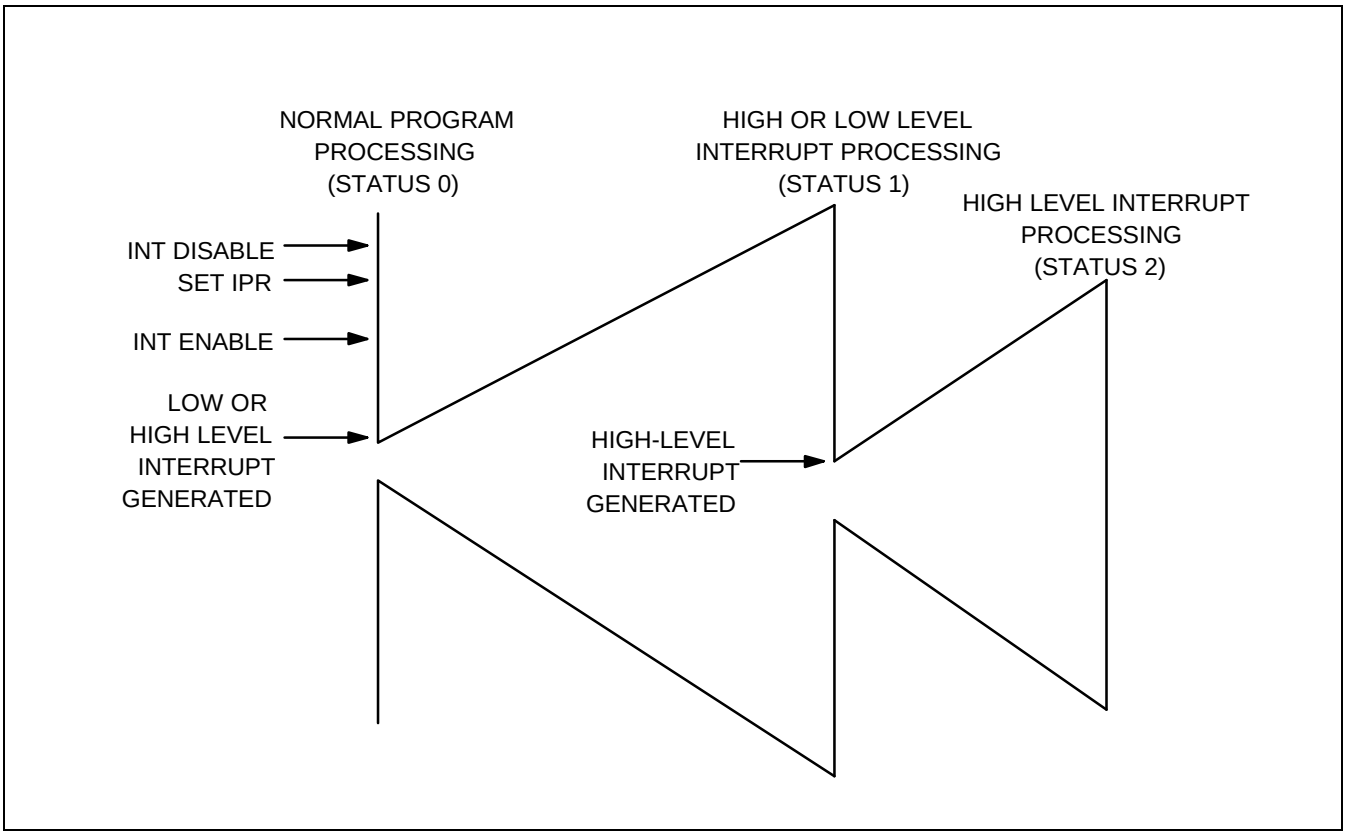


Figure 7-3. Two-Level Interrupt Handling

Multi-Level Interrupt Handling

With multi-level interrupt handling, a lower-priority interrupt request can be executed while a high-priority interrupt is being serviced. This is done by manipulating the interrupt status flags, IS0 and IS1 (see Table 7-2). When an interrupt is requested during normal program execution, interrupt status flags IS0 and IS1 are set to "1" and "0", respectively. This setting allows only highest-priority interrupts to be serviced. When a high-priority request is accepted, both interrupt status flags are then cleared to "0" by software so that a request of any priority level can be serviced. In this way, the high- and low-priority requests can be serviced in parallel (see Figure 7-4).

Table 7-2. IS1 and IS0 Bit Manipulation for Multi-Level Interrupt Handling

Process Status	Before INT		Effect of ISx Bit Setting	After INT ACK	
	IS1	IS0		IS1	IS0
0	0	0	All interrupt requests are serviced.	0	1
1	0	1	Only high-priority interrupts as determined by the current settings in the IPR register are serviced.	1	0
2	1	0	No additional interrupt requests will be serviced.	–	–
–	1	1	Value undefined	–	–

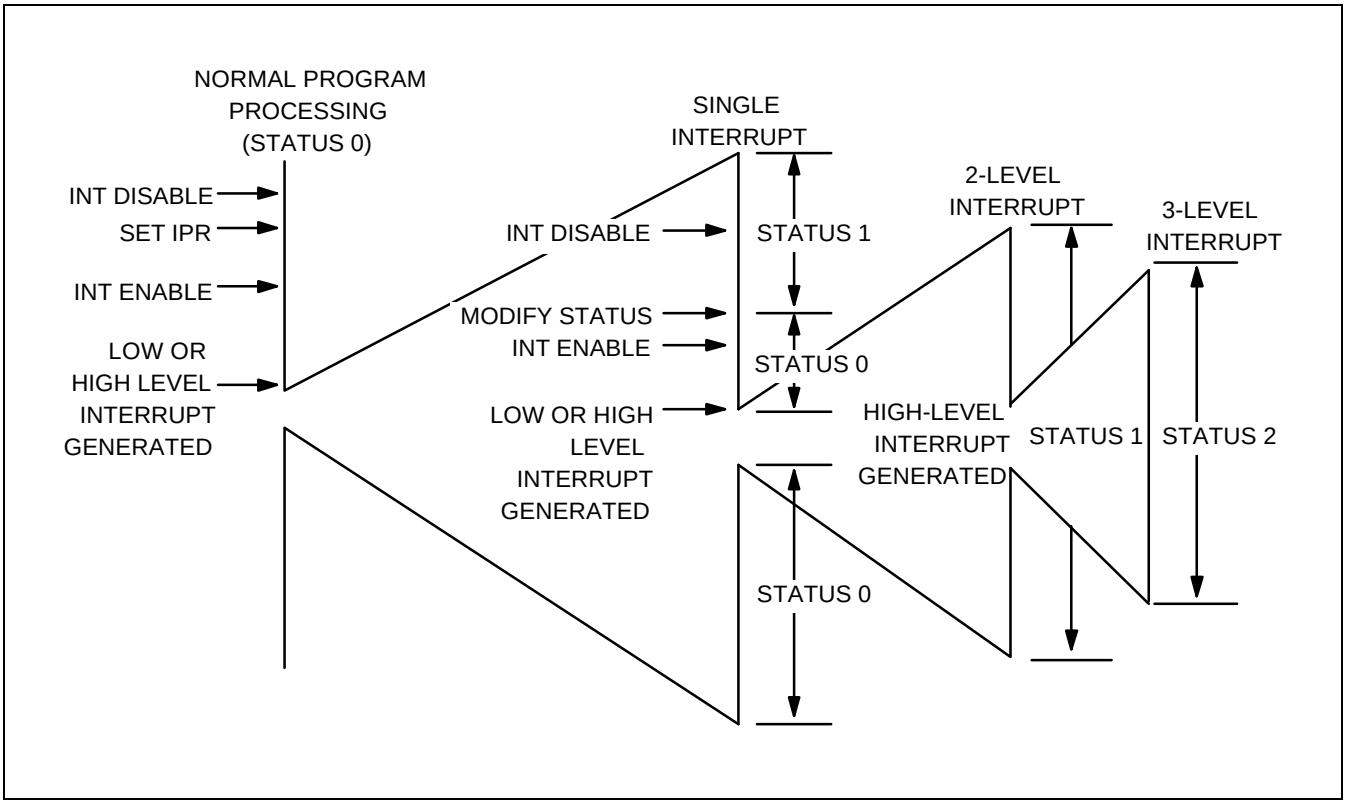


Figure 7-4. Multi-Level Interrupt Handling

INTERRUPT PRIORITY REGISTER (IPR)

The 4-bit interrupt priority register (IPR) is used to control multi-level interrupt handling and its reset value is logic zero. Before the IPR can be modified by 4-bit write instructions, all interrupts must first be disabled by a DI instruction.

FB2H	IME	IPR.2	IPR.1	IPR.0

By manipulating the IPR settings, you can choose to process all interrupt requests with the same priority level, or you can select one type of interrupt for high-priority processing. A low-priority interrupt can itself be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt cannot be interrupted by any other interrupt source.

Table 7-3. Standard Interrupt Priorities

Interrupt	Default Priority
INTB, INT4	1
INT0	2
INT1	3
INTT0	4
INTK	5

The MSB of the IPR, the interrupt master enable flag (IME), enables and disables all interrupt processing. Even if an interrupt request flag and its corresponding enable flag are set, a service routine cannot be executed until the IME flag is set to logic one. The IME flag can be directly manipulated by EI and DI instructions, regardless of the current enable memory bank (EMB) value.

Table 7-4. Interrupt Priority Register Settings

IPR.2	IPR.1	IPR.0	Result of IPR Bit Setting
0	0	0	Process all interrupt requests at low priority (NOTE)
0	0	1	Only INTB and INT4 interrupts are at high priority
0	1	0	Only INT0 interrupts is at high priority
0	1	1	Only INT1 interrupts is at high priority
1	0	0	Not available
1	0	1	Only INTT0 interrupts is at high priority
1	1	0	Not available
1	1	1	Only INTK interrupts is at high priority

NOTE: When all interrupts are low priority (the lower three bits of the IPR register are logic zero), the interrupt requested first will have high priority. Therefore, the first- request interrupt cannot be superceded by any other interrupt. If two or more interrupt requests are received simultaneously, the priority level is determined according to the standard interrupt priorities in Table 7-3 (the default priority assigned by hardware when the lower three IPR bits = "0"). In this case, the higher-priority interrupt request is serviced and the other interrupt is inhibited. Then, when the high-priority interrupt is returned from its service routine by an IRET instruction, the inhibited service routine is started.

☞ PROGRAMMING TIP —Priority Setting the INT Interrupt

The following instruction sequence sets the INT1 interrupt to high priority:

```
BITS    EMB
SMB     15
DI                          ;  IPR.3 (IME) ← 0
LD      A,#3H
LD      IPR,A
EI                          ;  IPR.3 (IME) ← 1
```

EXTERNAL INTERRUPT 0, 1 AND 2 MODE REGISTERS (IMOD0, IMOD1 AND IMOD2)

The external interrupt mode registers (IMOD0, IMOD1, and IMOD2) are used to control the triggering edge of the input signal at INT0, INT1 and INT2 respectively. The INT4 interrupt is an exception because its input signal generates an interrupt request on both rising and falling edges. Because INT2 is a quasi-interrupt, the interrupt request flag IRQ2 must be cleared to “0” by software.

Table 7-5. IMOD0, 1 and 2 Register Organization

IMOD0.3	IMOD0.2	IMOD0.1	IMOD0.0	Effect of IMOD Settings
“0”	“0”	0	0	Rising edge detection
		0	1	Falling edge detection
		1	0	Both rising and falling edge detection
		1	1	IRQ0 flag cannot be set to "1"

IMOD1.3 IMOD2.3	IMOD1.2 IMOD2.2	IMOD1.1 IMOD2.1	IMOD1.0 IMOD2.0	Effect of IMOD1 and IMOD2 Settings
			0	Rising edge detection
			1	Falling edge detection

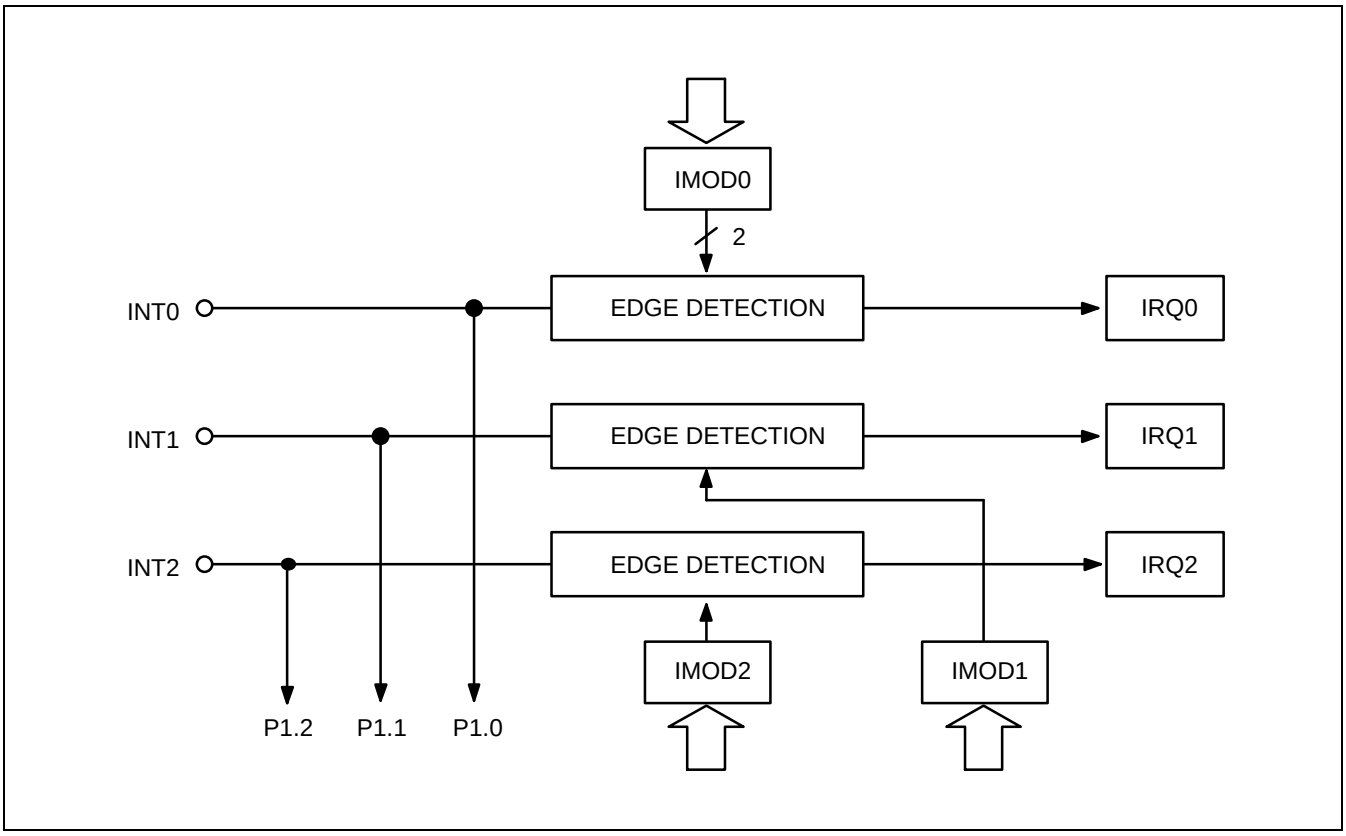


Figure 7-5. Circuit Diagram for INT0, INT1, and INT2 Pins

When modifying the IMOD0 and IMOD1 registers, it is possible to accidentally set an interrupt request flag. To avoid unwanted interrupts, take these precautions when writing your programs:

1. Disable all interrupts with a DI instruction.
2. Modify the IMOD0 or IMOD1 register.
3. Clear all relevant interrupt request flags.
4. Enable the interrupt by setting the appropriate IEx flag.
5. Enable all interrupts with an EI instructions.

EXTERNAL KEY INTERRUPT MODE REGISTER (IMODK)

The external key interrupt (INTK) mode register (IMODK) is used to select the K0–3 pins as interrupt inputs. If a rising or falling edge is detected at any one of these pins, the IRQK flag is set to “1”. This generates an interrupt request and a release signal for power-down mode.

If one or more of the pins which are configured as key interrupt (K0–3) are in low input state, the key interrupt can not be occurred.

Table 7-6. IMODK Register Bit Settings

IMODK.3	IMODK.2	IMODK.1	IMODK.0	Effect of IMODK Settings
0	0	0	0	Disable key interrupt
1				Enable edge detection at the Kx pins

IMODK.7	0	Falling edge detection
	1	Rising edge detection

NOTES:

1. If IMODK.x is set to logic “1”, the external key interrupts are enabled as edge detection at the corresponding pins. Where “x” is 0, 1, 2, and 3.
2. To generate a key interrupt, all of the selected pins must be at input high state for falling edge detection, or all of the selected pins must be at input low state for rising edge detection. If any one of them or more is at input low state or input high state, the interrupt may be not occurred at falling edge or rising edge.
3. To generate a key interrupt, first, configure pull-up resistors or external pull-down resistors. And then, select edge detection and pins by setting IMODK register.

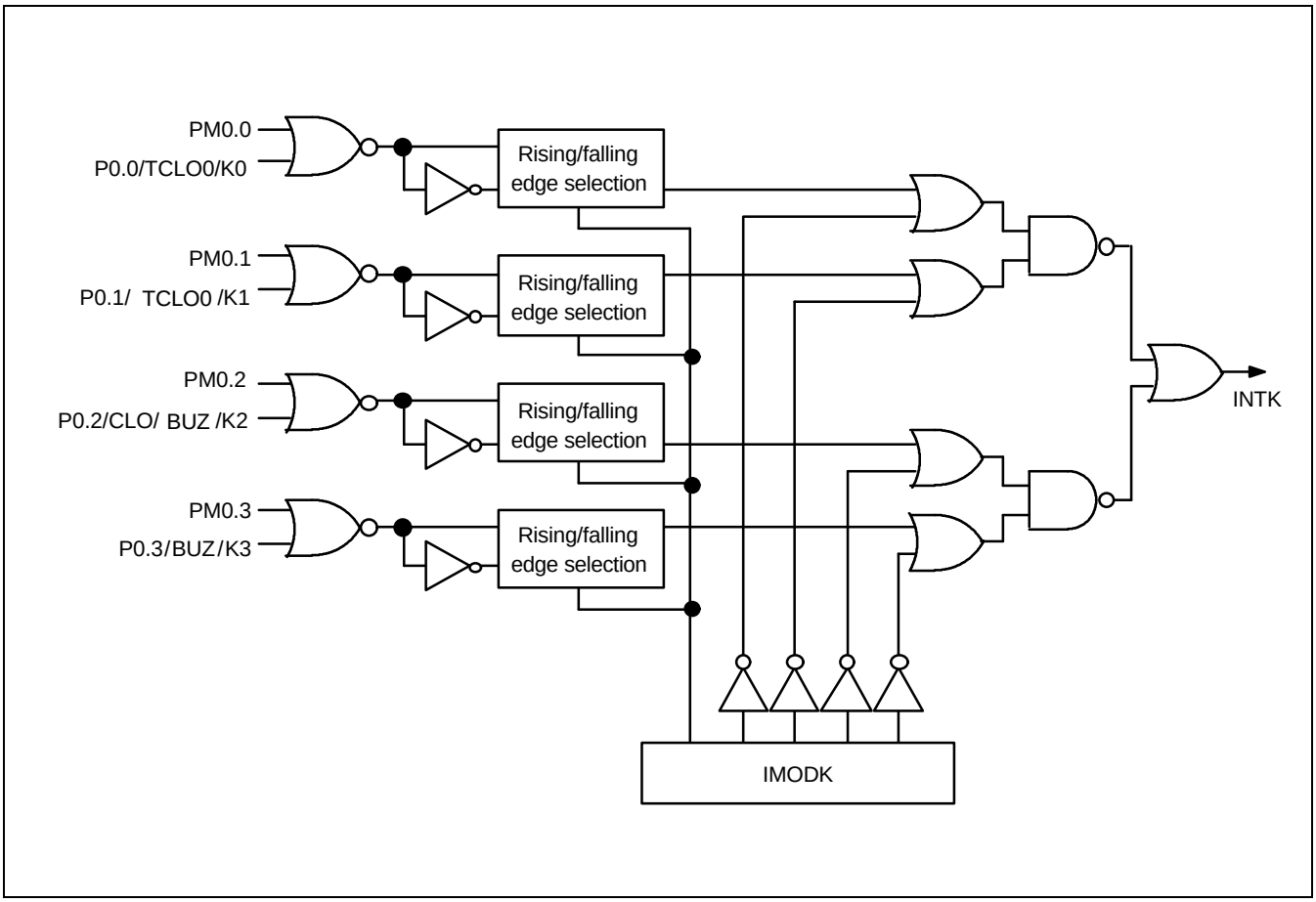


Figure 7-6. Circuit Diagram for INTK

☞ PROGRAMMING TIP — Using INTK as a Key Input Interrupt

When the key interrupt is used, the selected key interrupt source pin must be set to input:

1. When K0–3 are selected (four pins):

```
BITS    EMB
SMB     15
LD      EA,#0FH
LD      IMODK,EA        ;  (IMODK) ← #0FH, K0–3 falling edge select
LD      EA,#00H
LD      PMG1,EA         ;  P0, P1 ← input mode
LD      EA,#0FFH
LD      PUMOD0,EA       ;  Enable P0 and P1 pull-up resistors
```

2. When K0 is selected (one pin):

```
BITS    EMB
SMB     15
LD      EA,#01H
LD      IMODK,EA        ;  (IMODK) ← #1H, K0 falling edge select
LD      EA,#00H
LD      PMG1,EA         ;  P0, P1 ← input mode
LD      EA,#01H
LD      PUMOD0,EA       ;   Enable P0.0 pull-up resistor
```

INTERRUPT FLAGS

There are three types of interrupt flags: interrupt request and interrupt enable flags that correspond to each interrupt, the interrupt master enable flag, which enables or disables all interrupt processing.

Interrupt Master Enable Flag (IME)

The interrupt master enable flag, IME, enables or disables all interrupt processing. Therefore, even when an IRQx flag is set and its corresponding IEx flag is enabled, the interrupt service routine is not executed until the IME flag is set to logic one.
The IME flag is located in the IPR register (IPR.3). It can be directly be manipulated by EI and DI instructions, regardless of the current value of the enable memory bank flag (EMB).

IME	IPR.2	IPR.1	IPR.0	Effect of Bit Settings
0				Inhibit all interrupts
1				Enable all interrupts

Interrupt Enable Flags (IEx)

IEx flags, when set to logical one, enable specific interrupt requests to be serviced. When the interrupt request flag is set to logical one, an interrupt will not be serviced until its corresponding IEx flag is also enabled. Interrupt enable flags can be read, written, or tested directly by 1-bit instructions. IEx flags can be addressed directly at their specific RAM addresses, despite the current value of the enable memory bank (EMB) flag.

Table 7-7. Interrupt Enable and Interrupt Request Flag Addresses

Address	Bit 3	Bit 2	Bit 1	Bit 0
FB8H	IE4	IRQ4	IEB	IRQB
FBAH	0	0	IEW	IRQW
FBBH	IEK	IRQK	0	0
FBCH	0	0	IET0	IRQT0
FBDH	–			
FBEH	IE1	IRQ1	IE0	IRQ0
FBFH	0	0	IE2	IRQ2

NOTES:
1. IEx refers to all interrupt enable flags.
2. IRQx refers to all interrupt request flags.
3. IEx = 0 is interrupt disable mode.
4. IEx = 1 is interrupt enable mode.

Interrupt Request Flags (IRQx)

Interrupt request flags are read/write addressable by 1-bit or 4-bit instructions. IRQx flags can be addressed directly at their specific RAM addresses, regardless of the current value of the enable memory bank (EMB) flag. When a specific IRQx flag is set to logic one, the corresponding interrupt request is generated. The flag is then automatically cleared to logic zero when the interrupt has been serviced. Exceptions are the watch timer interrupt request flags, IRQW, and the external interrupt 2 flag IRQ2, which must be cleared by software after the interrupt service routine has executed. IRQx flags are also used to execute interrupt requests from software. In summary, follow these guidelines for using IRQx flags:

1. IRQx is set to request an interrupt when an interrupt meets the set condition for interrupt generation.
2. IRQx is set to "1" by hardware and then cleared by hardware when the interrupt has been serviced (with the exception of IRQW and IRQ2).
3. When IRQx is set to "1" by software, an interrupt is generated.

When two interrupts share the same service routine start address, interrupt processing may occur in one of two ways:

— When only one interrupt is enabled, the IRQx flag is cleared automatically when the interrupt has been serviced.

— When two interrupts are enabled, the request flag is not automatically cleared so that the user has an opportunity to locate the source of the interrupt request. In this case, the IRQx setting must be cleared manually using a BTSTZ instruction.

Table 7-8. Interrupt Request Flag Conditions and Priorities

Interrupt Source	Internal / External	Pre-condition for IRQx Flag Setting	Interrupt Priority	IRQ Flag Name
INTB	I	Reference time interval signal from basic timer	1	IRQB
INT4	E	Both rising and falling edges detected at INT4	1	IRQ4
INT0	E	Rising or falling edge detected at INT0 pin	2	IRQ0
INT1	E	Rising or falling edge detected at INT1 pin	3	IRQ1
INTT0	I	Signals for TCNT0 and TREF0 registers match	4	IRQT0
INTK	E	When a rising or falling edge detected at any one of the K0–3 pins	5	IRQK
INT2 (NOTE)	E	Rising or falling edge detected at INT2	–	IRQ2
INTW	I	Time interval of 0.5 s or 3.19 ms	–	IRQW

NOTE: The quasi-interrupt INT2 is only used for testing incoming signals.

☞ PROGRAMMING TIP — Enabling the INTB and INT4 Interrupts

To simultaneously enable INTB and INT4 interrupts:

```
INTB    DI
        BTSTZ   IRQB        ;  IRQB  =  1 ?
        JR      INT4        ;  If no, INT4 interrupt; if yes, INTB interrupt is processed
        .
        .
        .
        EI
        IRET
INT4    BITR    IRQ4        ;  INT4 is processed
        .
        .
        .
        EI
        IRET
```

8 POWER-DOWN

OVERVIEW

The S3C7295 microcontroller has two power-down modes to reduce power consumption: idle and stop. Idle mode is initiated by the IDLE instruction and stop mode by the instruction STOP. (Several NOP instructions must always follow an IDLE or STOP instruction in a program.) In idle mode, the CPU clock stops while peripherals and the oscillation source continue to operate normally.

When RESET occurs during normal operation or during a power-down mode, a reset operation is initiated and the CPU enters idle mode. When the standard oscillation stabilization time interval (31.3 ms at 4.19 MHz) has elapsed, normal CPU operation resumes.

In stop mode, main system clock oscillation is halted (assuming it is currently operating), and peripheral hardware components are powered-down. The effect of stop mode on specific peripheral hardware components — CPU, basic timer, timer/counters 0, watch timer, and LCD controller — and on external interrupt requests, is detailed in Table 8-1.

Idle or stop modes are terminated either by a RESET, or by an interrupt which is enabled by the corresponding interrupt enable flag, IEx. When power-down mode is terminated by RESET, a normal reset operation is executed. Assuming that both the interrupt enable flag and the interrupt request flag are set to "1", power-down mode is released immediately upon entering power-down mode.

When an interrupt is used to release power-down mode, the operation differs depending on the value of the interrupt master enable flag (IME):

— If the IME flag = “0”; If the power down mode release signal is generated, after releasing the power-down mode, program execution starts immediately under the instruction to enter power down mode without execution of interrupt service routine. The interrupt request flag remains set to logic one.

— If the IME flag = "1"; If the power down mode release signal is generated, after releasing the power down mode, two instructions following the instruction to enter power down mode are executed first and the interrupt service routine is executed, finally program is resumed.
However, when the release signal is caused by INT2 or INTW, the operation is identical to the IME = “0” condition because INT2 and INTW are a quasi-interrupt.

NOTE

Do not use stop mode if you are using an external clock source because X_{in} input must be restricted internally to V_{SS} to reduce current leakage.

Table 8-1. Hardware Operation During Power-Down Modes

Operation	Stop Mode (STOP)	Idle Mode (IDLE)
System clock status	Stop mode can be used only if the main system clock is selected as system clock (CPU clock)	Idle mode can be used if the main system clock or subsystem clock is selected as system clock (CPU clock)
Clock oscillator	Main system clock oscillation stops	CPU clock oscillation stops (main and subsystem clock oscillation continues)
Basic timer	Basic timer stops	Basic timer operates (with IRQB set at each reference interval)
Timer/counter 0	Operates only if TCL0 is selected as the counter clock	Timer/counter 0 operates
Watch timer	Operates only if subsystem clock (fxt) is selected as the counter clock	Watch timer operates
LCD controller	Operates only if a subsystem clock is selected as LCDCK	LCD controller operates
External interrupts	INT0, INT1, INT2, INT4, and INTK are acknowledged	INT0, INT1, INT2, INT4, and INTK are acknowledged
CPU	All CPU operations are disabled	All CPU operations are disabled
Mode release signal	Interrupt request signals are enabled by an interrupt enable flag or by RESET input	Interrupt request signals are enabled by an interrupt enable flag or by RESET input

Table 8-2. System Operating Mode Comparison

Mode	Condition	STOP/IDLE Mode Start Method	Current Consumption
Main operating mode	Main oscillator runs. Sub oscillator runs (stops). System clock is the main oscillation clock.	–	A
Main Idle mode	Main oscillator runs. Sub oscillator runs (stops). System clock is the main oscillation clock.	IDLE instruction	B
Main Stop mode	Main oscillator runs. Sub oscillator runs. System clock is the main oscillation clock.	STOP instruction	D
Sub operating mode	Main oscillator is stopped by SCMOD.3. Sub oscillator runs. System clock is the sub oscillation clock.	–	C
Sub Idle Mode	Main oscillator is stopped by SCMOD.3. Sub oscillator runs. System clock is the sub oscillation clock.	IDLE instruction	D
Sub Stop mode	Main oscillator is stopped by SCMOD.3. Sub oscillator runs. System clock is the sub oscillation clock.	Setting SCMOD.2 to “1”: This mode can be released only by an external RESET.	E
Main/Sub Stop mode	Main oscillator runs. Sub oscillator is stopped by SCMOD.2. System clock is the main oscillation clock.	STOP instruction: This mode can be released by an interrupt and RESET.	E

NOTE: The current consumption is: A > B > C > D > E.

IDLE MODE TIMING DIAGRAMS

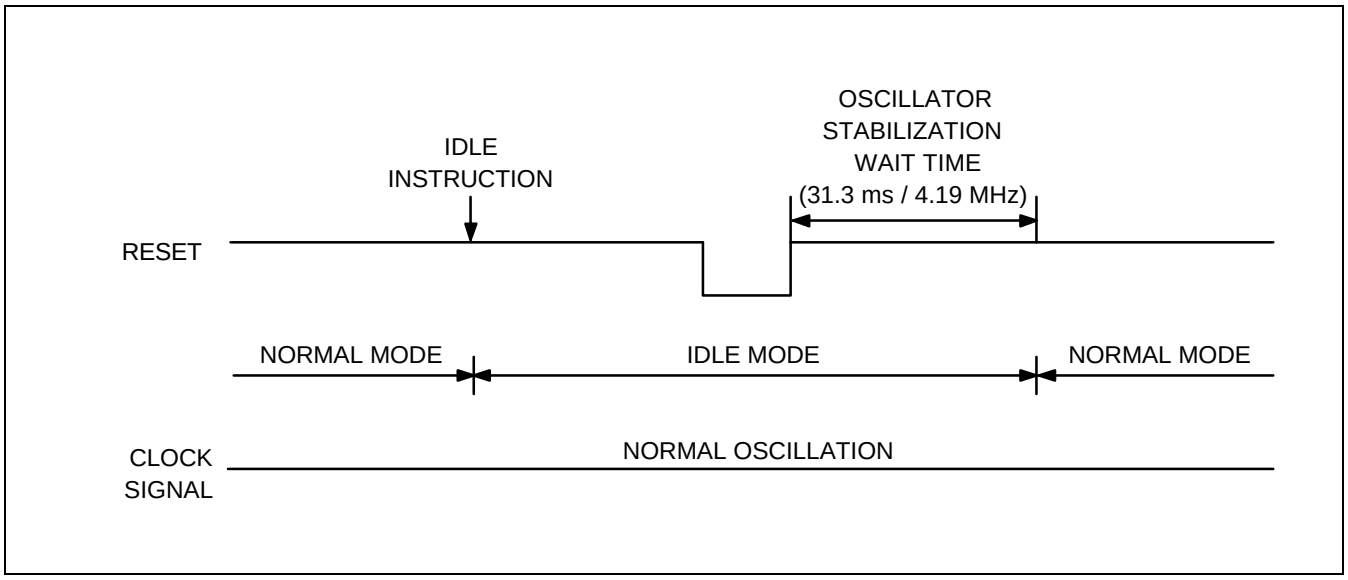


Figure 8-1. Timing When Idle Mode is Released by RESET

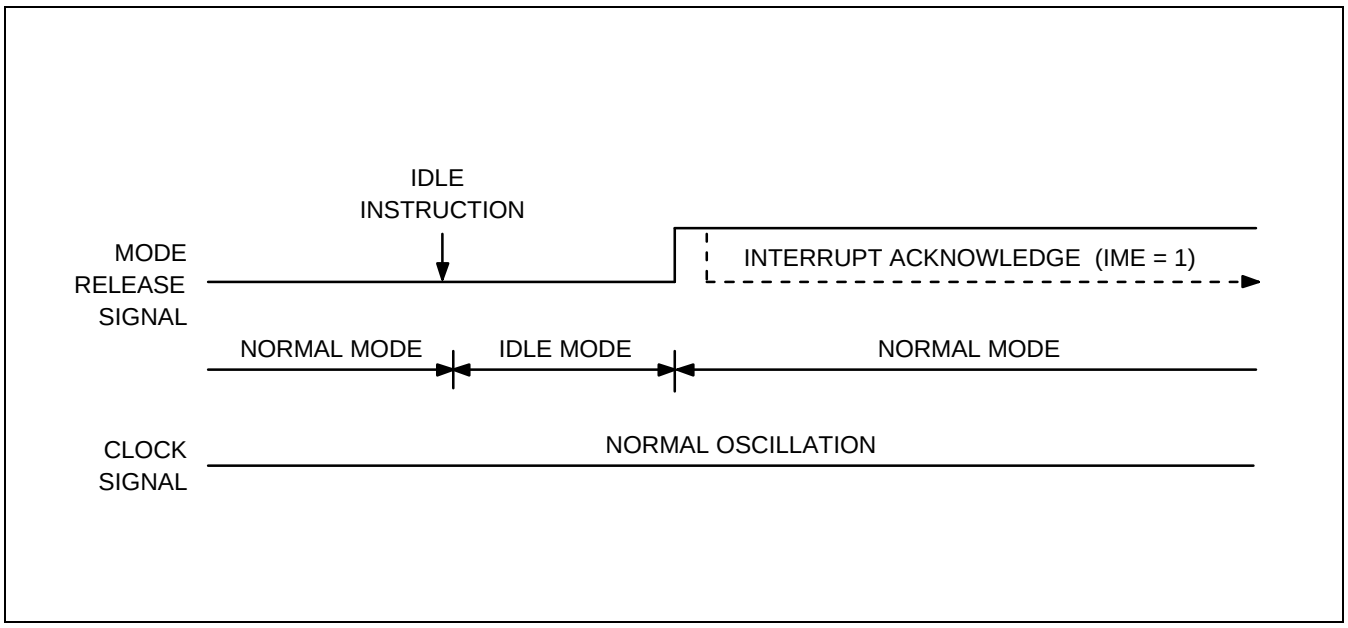


Figure 8-2. Timing When Idle Mode is Released by an Interrupt

STOP MODE TIMING DIAGRAMS

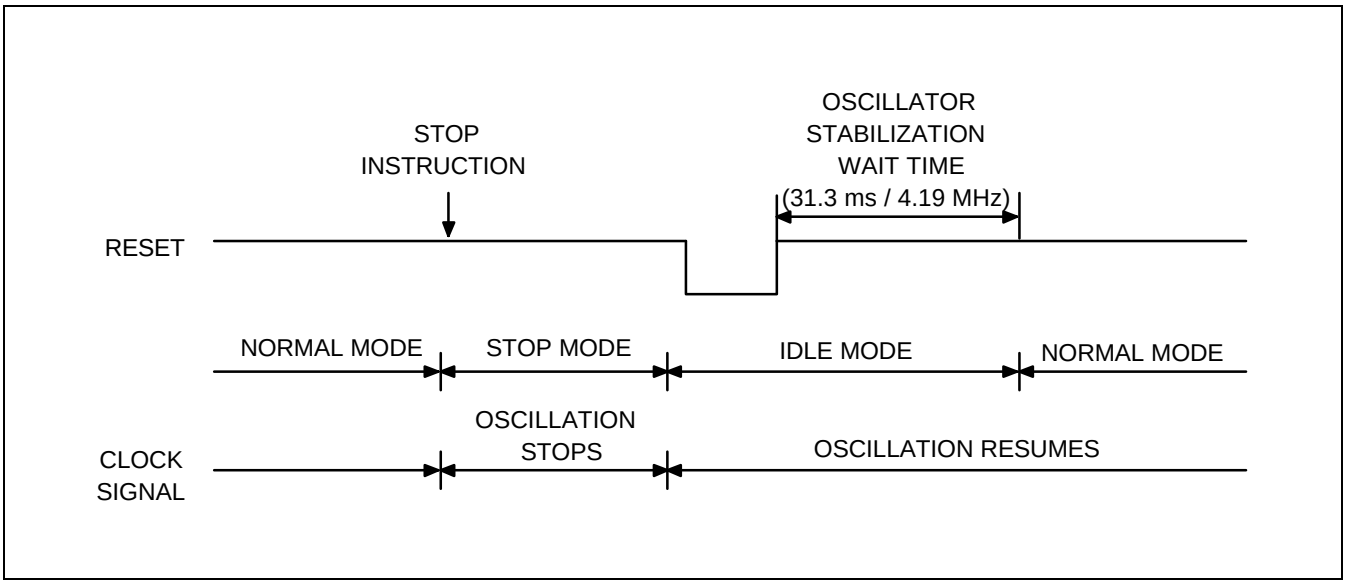


Figure 8-3. Timing When Stop Mode is Released by RESET

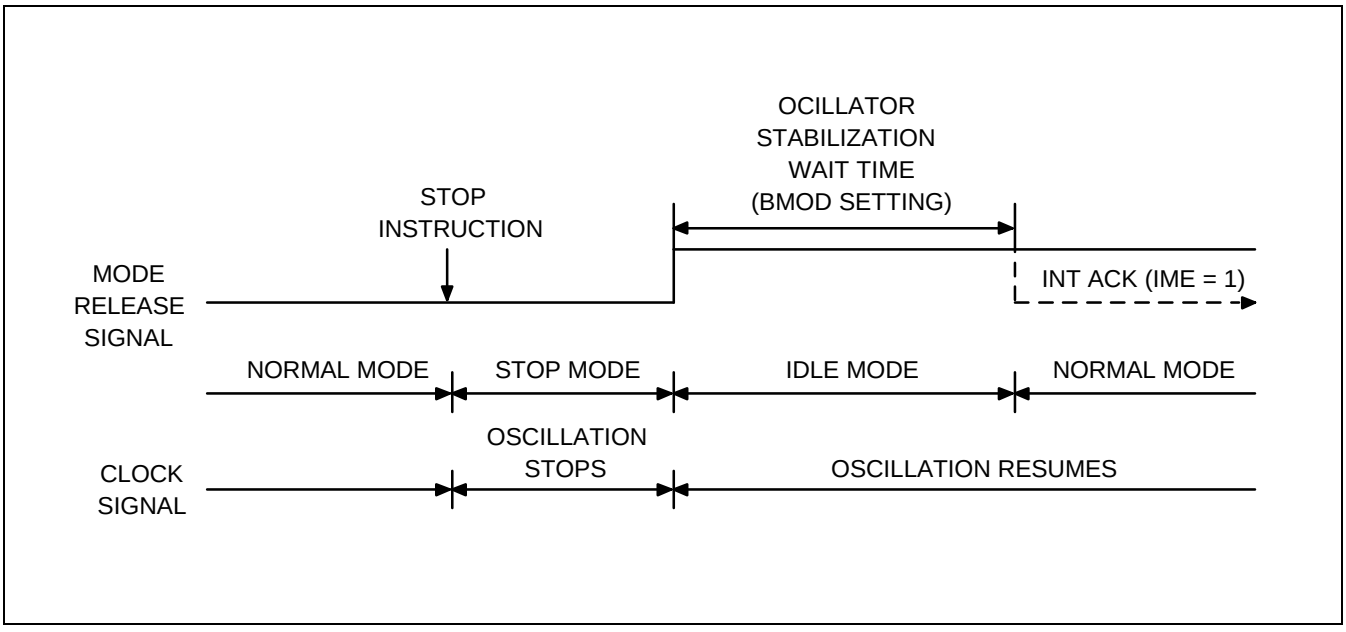


Figure 8-4. Timing When Stop Mode is Released by an Interrupt

☞ PROGRAMMING TIP — Reducing Power Consumption For Key Input Interrupt Processing

The following code shows real-time clock and interrupt processing for key inputs to reduce power consumption. In this example, the system clock source is switched from the main system clock to a subsystem clock and the LCD display is turned on:

```
KEYCLK  DI
        CALL    MA2SUB          ;  Main system  clock → subsystem clock switch
                                ;  subroutine
        SMB     15
        LD      A,#00H
        LD      P1,A            ;  All key strobe outputs to low level
        LD      EA,#00001111B
        LD      IMODK,EA        ;  Select K0–K3 enable
        SMB     0
        BITR    IRQW
        BITR    IRQK
        BITS    IEW
        BITS    IEK
CLKS1   CALL    WATDIS          ;  Execute clock and display changing subroutine
        BTSTZ   IRQK
        JR      CIDLE
        CALL    SUB2MA          ;  Subsystem clock → main system clock switch
                                ;  subroutine
        EI
        RET
CIDLE   IDLE                    ;  Engage idle mode
        NOP
        NOP
        NOP
        JPS     CLKS1
```

RECOMMENDED CONNECTIONS FOR UNUSED PINS

To reduce overall power consumption, please configure unused pins according to the guidelines described in Table 8-2.

Table 8-3. Unused Pin Connections for Reducing Power Consumption

Pin/Share Pin Names	Recommended Connection
P0.0/K0/TCLO0 P0.1/K1/TCLO0 P0.2/K2/BUZ/CLO P0.3/K3/BUZ	Input mode: Connect to V_{DD} Output mode: No connection
P1.0/INT0 – P1.3/INT4	Input mode: Connect to V_{DD} Output mode: No connection
SEG0–SEG39 COM0–COM7	No connection
V_{LC0}	No connection
BIAS	No connection
CA, CB	No connection
XT_{in}	Stop sub-oscillator by setting the SCMOD.2 to logic “1”
XT_{out}	No connection
TEST	Connect to V_{SS}

9 RESET

OVERVIEW

When a RESET signal is input during normal operation or power-down mode, a hardware reset operation is initiated and the CPU enters idle mode. Then, when the standard oscillation stabilization interval of 31.3 ms at 4.19 MHz has elapsed, normal system operation resumes.

Regardless of when the RESET occurs — during normal operating mode or during a power-down mode — most hardware register values are set to the reset values described in Table 9-1. The current status of several register values is, however, always retained when a RESET occurs during idle or stop mode; If a RESET occurs during normal operating mode, their values are undefined. Current values that are retained in this case are as follows:

— Carry flag
— Data memory values
— General-purpose registers E, A, L, H, X, W, Z, and Y

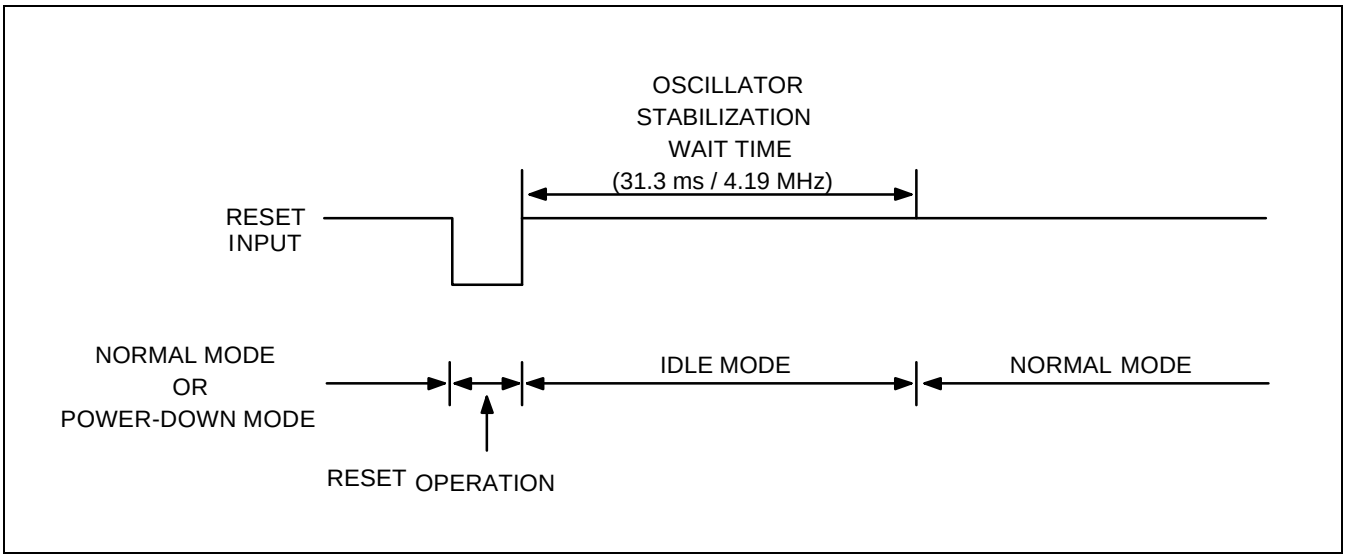


Figure 9-1. Timing for Oscillation Stabilization after RESET

HARDWARE REGISTER VALUES AFTER RESET

Table 9-1 gives you detailed information about hardware register values after a RESET occurs during power-down mode or during normal operation.

Table 9-1. Hardware Register Values after RESET

Hardware Component or Subcomponent	If RESET Occurs During Power-Down Mode	If RESET Occurs During Normal Operation
Program counter (PC)	Lower six bits of address 0000H are transferred to PC13–8, and the contents of 0001H to PC7–0.	Lower six bits of address 0000H are transferred to PC13–8, and the contents of 0001H to PC7–0.
Program Status Word (PSW):		
Carry flag (C)	Retained	Undefined
Skip flag (SC0–SC2)	0	0
Interrupt status flags (IS0, IS1)	0	0
Bank enable flags (EMB, ERB)	Bit 6 of address 0000H in program memory is transferred to the ERB flag, and bit 7 of the address to the EMB flag.	Bit 6 of address 0000H in program memory is transferred to the ERB flag, and bit 7 of the address to the EMB flag.
Stack pointer (SP)	Undefined	Undefined
Data Memory (RAM):		
General registers E, A, L, H, X, W, Z, Y	Values retained	Undefined
General-purpose registers	Values retained (note)	Undefined
Bank selection registers (SMB, SRB)	0, 0	0, 0
Clocks:		
Power control register (PCON)	0	0
Clock output mode register (CLMOD)	0	0
System clock mode register (SCMOD)	0	0
Interrupts:		
Interrupt request flags (IRQx)	0	0
Interrupt enable flags (IEx)	0	0
Interrupt priority flag (IPR)	0	0
Interrupt master enable flag (IME)	0	0
INT0 mode register (IMOD0)	0	0
INT1 mode register (IMOD1)	0	0
INT2 mode register (IMOD2)	0	0
INTK mode register (IMODK)	0	0

NOTE: The values of the 0F8H–0FDH are not retained when a RESET signal is input.

Table 9-1. Hardware Register Values after RESET (Continued)

Hardware Component or Subcomponent	If RESET Occurs During Power-Down Mode	If RESET Occurs During Normal Operation
I/O Ports:		
Output buffers	Off	Off
Output latches	0	0
Port mode flags (PM)	0	0
Pull-up resistor mode reg (PUMOD0)	0	0
Basic Timer:		
Count register (BCNT)	Undefined	Undefined
Mode register (BMOD)	0	0
Mode register (WDMOD)	A5H	A5H
Counter clear flag (WDTCF)	0	0
Timer/Counter 0:		
Count registers (TCNT0)	0	0
Reference registers (TREF0)	FFH, FFFFH	FFH, FFFFH
Output enable flags (TOE, TOEB)	0	0
Watch Timer:		
Watch timer mode register (WMOD)	0	0
Inverted buzzer enable flag (BUZB)	0	0
LCD Driver/Controller:		
LCD mode register (LMOD)	0	0
LCD control register (LCON)	0	0
Display data memory	Values retained	Undefined
Output buffers	Off	Off
N-Channel Open-Drain Mode Register		
PNE1	0	0

10 I/O PORTS

OVERVIEW

The S3C7295 has 2 ports. There are total of 8 configurable I/O pins. Pin addresses for all ports are mapped to bank 15 of the RAM. The contents of I/O port pin latches can be read, written, or tested at the corresponding address using bit manipulation instructions.

Port Mode Flags

Port mode flags (PM) are used to configure I/O ports to input or output mode by setting or clearing the corresponding I/O buffer.

Pull-up Resistor Mode Register (PUMOD)

The pull-up resistor mode register (PUMOD0) is used to assign internal pull-up resistors by software to specific ports. When a configurable I/O port pin is used as an output pin, its assigned pull-up resistor is automatically disabled, even though the pin's pull-up is enabled by a corresponding PUMOD bit setting.

N-channel open-drain mode register (PNE1)

The n-channel open-drain mode register (PNE1) is used to configure outputs as n-channel open-drain outputs or as push-pull outputs.

Table 10-1. I/O Port Overview

Port	I/O	Pins	Pin Names	Address	Function Description
0	I/O	4	P0.0–P0.3	FF0H	4-bit I/O port. 1-bit and 4-bit read/write and test is possible. Individual pins are software configurable as input or output. Individual pins are software configurable as open-drain or push-pull output. Individual pull-up resistors are software assignable; pull-up resistors are automatically disabled for output pins.
1	I/O	4	P1.0–P1.3	FF1H	Same as Port0

Table 10-2. Port Pin Status During Instruction Execution

Instruction Type	Example	Input Mode Status	Output Mode Status

1-bit test
1-bit input
4-bit input
	BTST P0.1
	LDB C,P1.3
	LD A,P1
		Input or test data at each pin	Input or test data at output latch

1-bit output	BITR P0.3	Output latch contents undefined	Output pin status is modified
4-bit output	LD P0,A	Transfer accumulator data to the output latch	Transfer accumulator data to the output pin

PORT MODE FLAGS (PM FLAGS)

Port mode flags (PM) are used to configure I/O ports to input or output mode by setting or clearing the corresponding I/O buffer.

If a PM bit is "0", the corresponding I/O pin is set to input mode. If the PM bit is "1", the pin is set to output mode: PM0.0 for P0.0, PM0.1 for P0.1, and so on.

Table 10-3. Port Mode Group Flags

PM Group ID	Address	Bit 3	Bit 2	Bit 1	Bit 0
PMG1	FE8H	PM0.3	PM0.2	PM0.1	PM0.0
	FE9H	PM1.3	PM1.2	PM1.1	PM1.0

☞ PROGRAMMING TIP — Configuring I/O Ports to Input or Output

Configure port 0 as an output port:

```
BITS    EMB
SMB     15
LD      EA, #00001111B
LD      PMG1, EA            ;  P0 ← output, P1 ← input
```

PULL-UP RESISTOR MODE REGISTER (PUMOD)

The pull-up resistor mode registers PUMOD0 is used to assign internal pull-up resistors to specific ports.

When a PUMOD0.n bit is "1", a pull-up resistor is assigned to the corresponding I/O port: PUR0.3 for port 0.3, PUR0.2 for port0.2, and so on.

When a configurable I/O port pin is used as an output pin, its assigned pull-up resistor is automatically disabled, even though the pin's pull-up may be enabled by a corresponding PUMOD0 bit setting.

Table 10-4. Pull-up Resistor Mode Register (PUMOD) Organization

PUMOD ID	Address	Bit 3	Bit 2	Bit 1	Bit 0
PUMOD0	FDCH	PUR0.3	PUR0.2	PUR0.1	PUR0.0
	FDDH	PUR1.3	PUR1.2	PUR1.1	PUR1.0

☞ PROGRAMMING TIP — Enabling and Disabling I/O Port Pull-up Resistors

P1 enable pull-up resistor:

```
BITS    EMB
SMB     15
LD      EA, #11110000B
LD      PUMOD0, EA          ;  P1 enable
```

N-CHANNEL OPEN-DRAIN MODE REGISTER (PNE)

The n-channel, open-drain mode register, PNE, is used to configure ports 0 and 1 to n-channel open-drain mode or as push-pull outputs.

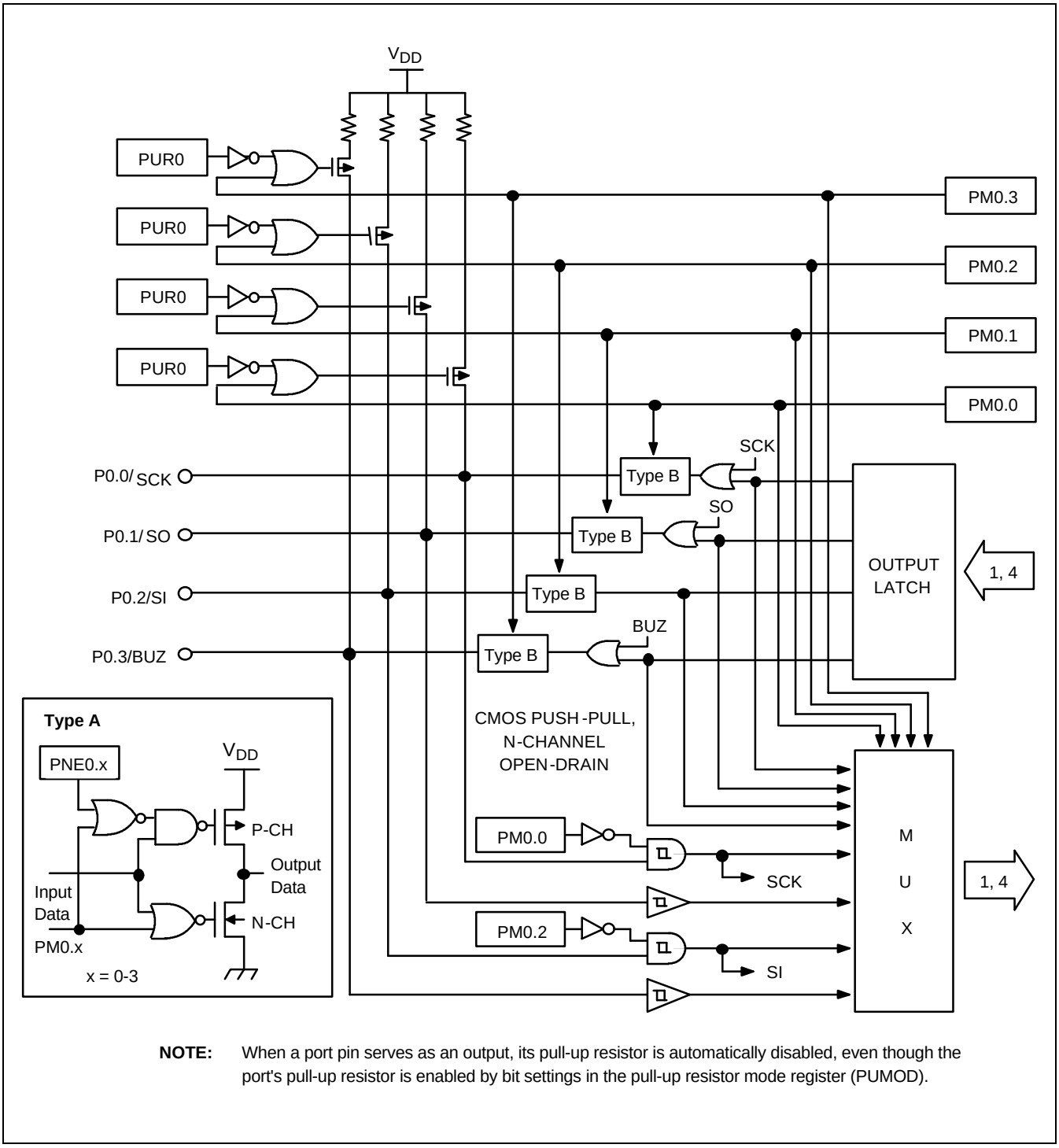
When a bit in the PNE register is set to "1", the corresponding output pin is configured to n-channel open-drain; when set to "0", the output pin is configured to push-pull.

The PNE register consists of an 8-bit register as shown below. PNE1 can be addressed by 8-bit write instructions only.

Table 10-5. N-channel, Open-drain Mode Register (PNE)

PNE ID	Address	Bit 3	Bit 2	Bit 1	Bit 0
PNE1	FDAH	PNE0.3	PNE0.2	PNE0.1	PNE0.0
	FDBH	PNE1.3	PNE1.2	PNE1.1	PNE1.0

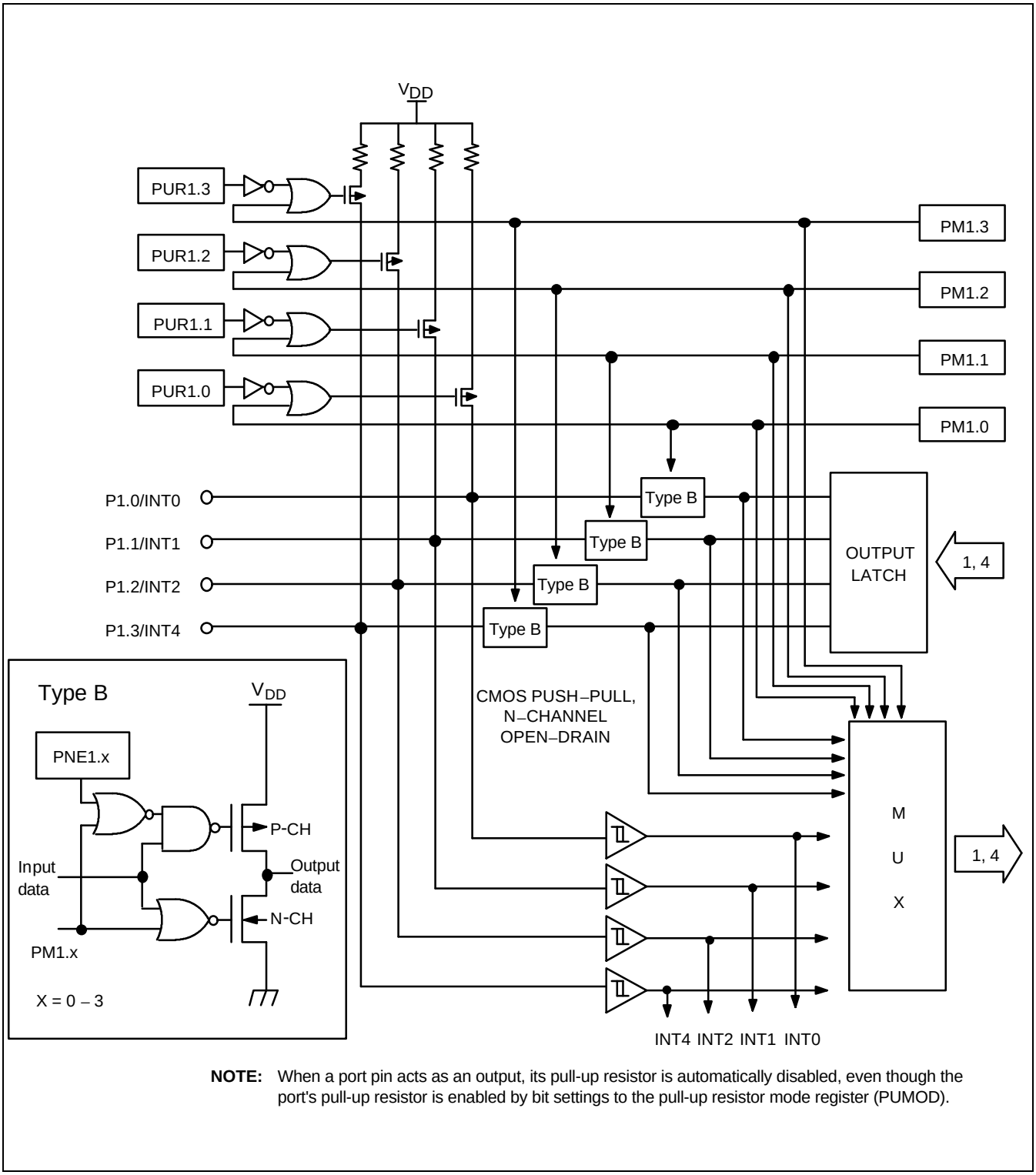
PORT 0 CIRCUIT DIAGRAM



NOTE: When a port pin serves as an output, its pull-up resistor is automatically disabled, even though the port's pull-up resistor is enabled by bit settings in the pull-up resistor mode register (PUMOD).

Figure 10-1. Port 0 Circuit Diagram

PORT 1 CIRCUIT DIAGRAM



NOTE: When a port pin acts as an output, its pull-up resistor is automatically disabled, even though the port's pull-up resistor is enabled by bit settings to the pull-up resistor mode register (PUMOD).

Figure 10-2. Port 1 Circuit Diagram

11 TIMERS and TIMER/COUNTERS

OVERVIEW

The S3C7295 microcontroller has three timer and timer/counter modules:

— 8-bit basic timer (BT)
— 8-bit timer/counter (TC0)
— Watch timer (WT)

The 8-bit basic timer (BT) is the microcontroller's main interval timer. It generates an interrupt request at a fixed time interval when the appropriate modification is made to its mode register. The basic timer also functions as a 'watchdog' timer and is used to determine clock oscillation stabilization time when stop mode is released by an interrupt and after a RESET.

The 8-bit timer/counter (TC0) is programmable timer/counter that is used primarily for clock frequency modification and output.

The watch timer (WT) module consists of an 8-bit watch timer mode register, a clock selector, and a frequency divider circuit. Watch timer functions include real-time and watch-time measurement, system clock interval timing, buzzer and inverted output generation.

BASIC TIMER (BT)

OVERVIEW

The 8-bit basic timer (BT) has six functional components:

— Clock selector logic

— 4-bit mode register (BMOD)

— 8-bit counter register (BCNT)

— 8-bit watchdog timer mode register (WDMOD)

— Watchdog timer counter clear flag (WDTCF)

The basic timer generates interrupt requests at precise intervals, based on the frequency of the system clock. You can use the basic timer as a "watchdog" timer for monitoring system events or use BT output to stabilize clock oscillation when stop mode is released by an interrupt and following RESET. Bit settings in the basic timer mode register BMOD turns the BT module on and off, selects the input clock frequency, and controls interrupt or stabilization intervals.

Interval Timer Function

The basic timer's primary function is to measure elapsed time intervals. The standard time interval is equal to 256 basic timer clock pulses.

To restart the basic timer, one bit setting is required: bit 3 of the mode register BMOD should be set to logic one. The input clock frequency and the interrupt and stabilization interval are selected by loading the appropriate bit values to BMOD.2–BMOD.0.

The 8-bit counter register, BCNT, is incremented each time a clock signal is detected that corresponds to the frequency selected by BMOD. BCNT continues incrementing as it counts BT clocks until an overflow occurs (≥ 255). An overflow causes the BT interrupt request flag (IRQB) to be set to logic one to signal that the designated time interval has elapsed. An interrupt request is than generated, BCNT is cleared to logic zero, and counting continues from 00H.

Watchdog Timer Function

The basic timer can also be used as a "watchdog" timer to signal the occurrence of system or program operation error. For this purpose, instruction that clear the watchdog timer (BITS WDTCF) should be executed at proper points in a program within given period. If an instruction that clears the watchdog timer is not executed within the given period and the watchdog timer overflows, reset signal is generated and the system restarts with reset status. An operation of watchdog timer is as follows:

— Write some values (except #5AH) to watchdog timer mode register, WDMOD

— If WDCNT overflows, system reset is generated.

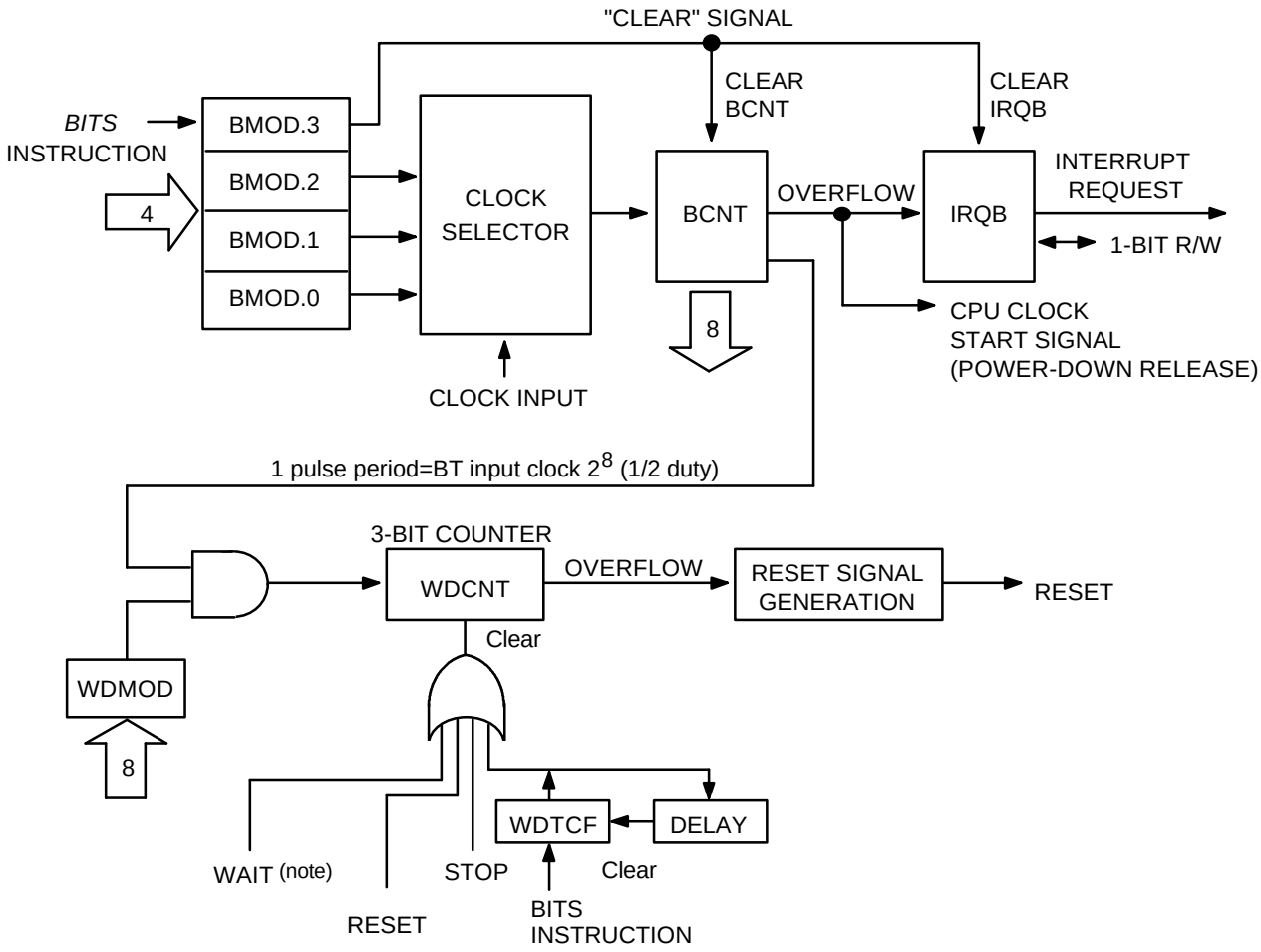
Oscillation Stabilization Interval Control

Bits 2–0 of the BMOD register are used to select the input clock frequency for the basic timer. This setting also determines the time interval (also referred to as 'wait time') required to stabilize clock signal oscillation when stop mode is released by an interrupt. When a RESET signal is inputted, the standard stabilization interval for system clock oscillation following the RESET is 31.3 ms at 4.19 MHz.

Table 11-1. Basic Timer Register Overview

Register Name	Type	Description	Size	RAM Address	Addressing Mode	Reset Value
BMOD	Control	Controls the clock frequency (mode) of the basic timer; also, the oscillation stabilization interval after stop mode release or RESET	4-bit	F85H	4-bit write-only; BMOD.3: 1-bit writeable	"0"
BCNT	Counter	Counts clock pulses matching the BMOD frequency setting	8-bit	F86H–F87H	8-bit read-only	U(note)
WDMOD	Control	Controls watchdog timer operation.	8-bit	F98H–F99H	8-bit write-only	A5H
WDTCF	Control	Clears the watchdog timer's counter.	1-bit	F9AH.3	1-, 4-bit write	"0"

NOTE: 'U' means the value is undetermined after a RESET.



NOTES:

1. WAIT means stabilization time after RESET or stabilization time after STOP mode release.
2. The RESET signal can be generated if the WDMOD is toggled for 8 times where “toggle” means change from 5AH to other value and vice versa.
3. When the watchdog timer is enabled or the 3-bit counter of the watchdog timer is cleared to “0”, the BCNT value is not cleared but increased continuously.
   As a result, the 3-bit counter of the watchdog timer (WDCNT) can be increased by 1.
   For example, when the BMOD value is x000B and the watchdog timer is enabled, the watchdog timer interval time is either $2^3 \times 2^{12} \times 2^8/fxx$ or $(2^3 - 1) \times 2^{12} \times 2^8/fxx$.

Figure 11-1. Basic Timer Circuit Diagram

BASIC TIMER MODE REGISTER (BMOD)

The basic timer mode register, BMOD, is a 4-bit write-only register. Bit 3, the basic timer start control bit, is also 1-bit addressable. All BMOD values are set to logic zero following RESET and interrupt request signal generation is set to the longest interval. (BT counter operation cannot be stopped.) BMOD settings have the following effects:

— Restart the basic timer;

— Control the frequency of clock signal input to the basic timer;

— Determine time interval required for clock oscillation to stabilize following the release of stop mode by an interrupt.

By loading different values into the BMOD register, you can dynamically modify the basic timer clock frequency during program execution. Four BT frequencies, ranging from fxx/2^{12} to fxx/2^5, are selectable. Since BMOD's reset value is logic zero, the default clock frequency setting is fxx/2^{12}.

The most significant bit of the BMOD register, BMOD.3, is used to restart the basic timer. When BMOD.3 is set to logic one by a 1-bit write instruction, the contents of the BT counter register (BCNT) and the BT interrupt request flag (IRQB) are both cleared to logic zero, and timer operation restarts.

The combination of bit settings in the remaining three registers — BMOD.2, BMOD.1, and BMOD.0 — determine the clock input frequency and oscillation stabilization interval.

Table 11-2. Basic Timer Mode Register (BMOD) Organization

BMOD.3	Basic Timer Start Control Bit
1	Start basic timer; clear IRQB, BCNT, and BMOD.3 to "0"

BMOD.2	BMOD.1	BMOD.0	Basic Timer Input Clock	Interrupt Interval Time (Wait Time)
0	0	0	fxx/2^{12} (1.02 kHz)	2^{20}/fxx (250 ms)
0	1	1	fxx/2^9 (8.18 kHz)	2^{17}/fxx (31.3 ms)
1	0	1	fxx/2^7 (32.7 kHz)	2^{15}/fxx (7.82 ms)
1	1	1	fxx/2^5 (131 kHz)	2^{13}/fxx (1.95 ms)

NOTES:

1. Clock frequencies and interrupt interval time assume a system oscillator clock frequency (fx) of 4.19 MHz.
2. fxx = system clock frequency.
3. Wait time is the time required to stabilize clock signal oscillation after stop mode is released. The data in the table column "Interrupt Interval Time" can also be interpreted as "Oscillation Stabilization."
4. The standard stabilization time for system clock oscillation following a RESET is 31.3 ms at 4.19 MHz.

BASIC TIMER COUNTER (BCNT)

BCNT is an 8-bit counter for the basic timer. It can be addressed by 8-bit read instructions. RESET leaves the BCNT counter value undetermined. BCNT is automatically cleared to logic zero whenever the BMOD register control bit (BMOD.3) is set to "1" to restart the basic timer. It is incremented each time a clock pulse of the frequency determined by the current BMOD bit settings is detected.

When BCNT has incrementing to hexadecimal ‘FFH’ (≥ 255 clock pulses), it is cleared to ‘00H’ and an overflow is generated. The overflow causes the interrupt request flag, IRQB, to be set to logic one. When the interrupt request is generated, BCNT immediately resumes counting incoming clock signals.

NOTE

Always execute a BCNT read operation twice to eliminate the possibility of reading unstable data while the counter is incrementing. If, after two consecutive reads, the BCNT values match, you can select the latter value as valid data. Until the results of the consecutive reads match, however, the read operation must be repeated until the validation condition is met.

BASIC TIMER OPERATION SEQUENCE

The basic timer's sequence of operations may be summarized as follows:

1. Set BMOD.3 to logic one to restart the basic timer
2. BCNT is then incremented by one after each clock pulse corresponding to BMOD selection
3. BCNT overflows if BCNT = 255 (BCNT = FFH)
4. When an overflow occurs, the IRQB flag is set by hardware to logic one
5. The interrupt request is generated
6. BCNT is then cleared by hardware to logic zero
7. Basic timer resumes counting clock pulses

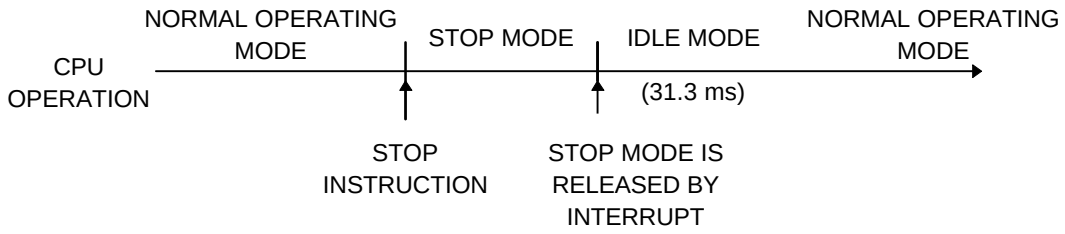
☞ PROGRAMMING TIP — Using The Basic Timer

1. To read the basic timer count register (BCNT):

```
        BITS    EMB
        SMB     15
BCNTR   LD      EA,BCNT
        LD      YZ,EA
        LD      EA,BCNT
        CPSE    EA,YZ
        JR      BCNTR
```

2. When stop mode is released by an interrupt, set the oscillation stabilization interval to 31.3 ms:

```
        BITS    EMB
        SMB     15
        LD      A,#0BH
        LD      BMOD,A          ;  Wait time is 31.3 ms
        NOP
        STOP                    ;  Set stop power-down mode
        NOP
        NOP
        NOP
```



3. To set the basic timer interrupt interval time to 1.95 ms (at 4.19 MHz):

```
        BITS    EMB
        SMB     15
        LD      A,#0FH
        LD      BMOD,A
        EI
        BITS    IEB             ;  Basic timer interrupt enable flag is set to "1"
```

4. Clear BCNT and the IRQB flag and restart the basic timer:

```
        BITS    EMB
        SMB     15
        BITS    BMOD.3
```

WATCHDOG TIMER MODE REGISTER (WDMOD)

The watchdog timer mode register, WDMOD, is a 8-bit write-only register. WDMOD register controls to enable or disable the watchdog function. WDMOD values are set to logic “A5H” following RESET and this value enables the watchdog timer. Watchdog timer is set to the longest interval because BT overflow signal is generated with the longest interval.

WDMOD	Watchdog Timer Enable/Disable Control
5AH	Disable watchdog timer function
Any other value	Enable watchdog timer function

WATCHDOG TIMER COUNTER (WDCNT)

The watchdog timer counter, WDCNT, is a 3-bit counter. WDCNT is automatically cleared to logic zero, and restarts whenever the WDTCF register control bit is set to “1”. RESET, stop, and wait signal clears the WDCNT to logic zero also.

WDCNT increments each time a clock pulse of the overflow frequency determined by the current BMOD bit setting is generated. When WDCNT has incremented to hexadecimal ‘07H’, it is cleared to ‘00H’ and an overflow is generated. The overflow causes the system RESET. When the interrupt request is generated, BCNT immediately resumes counting incoming clock signals.

WATCHDOG TIMER COUNTER CLEAR FLAG (WDTCF)

The watchdog timer counter clear flag, WDTCF, is a 1-bit write instruction. When WDTCF is set to one, it clears the WDCNT to zero and restarts the WDCNT. WDTCF register bits 2–0 are always logic zero.

Table 11-3. Watchdog Timer Interval Time

BMOD	BT Input Clock	WDT Interval Time (3)	
x000b	$fxx/2^{12}$	$2^3 \times 2^{12} \times 2^8/fxx$ or $(2^3–1) \times 2^{12} \times 2^8/fxx$	1.75–2.0 sec
x011b	$fxx/2^9$	$2^3 \times 2^9 \times 2^8/fxx$ or $(2^3–1) \times 2^9 \times 2^8/fxx$	218.7–250 ms
x101b	$fxx/2^7$	$2^3 \times 2^7 \times 2^8/fxx$ or $(2^3–1) \times 2^7 \times 2^8/fxx$	54.6–62.5ms
x111b	$fxx/2^5$	$2^3 \times 2^5 \times 2^8/fxx$ or $(2^3–1) \times 2^5 \times 2^8/fxx$	13.6–15.6 ms

NOTES:

1. Clock frequencies assume a system oscillator clock frequency (fx) of 4.19 MHz
2. fxx = system clock frequency.
3. When the watchdog timer is enabled or the 3-bit counter of the watchdog timer is cleared to “0”, the BCNT value is not cleared but increased continuously. As a result, the 3-bit counter of the watchdog timer (WDCNT) can be increased by 1. For example, when the BMOD value is x000b and the watchdog timer is enabled, the watchdog timer interval time is either $2^3 \times 2^{12} \times 2^8/fxx$ or $(2^3–1) \times 2^{12} \times 2^8/fxx$.

☞ PROGRAMMING TIP — Using the Watchdog Timer

```
RESET   DI
        LD      EA,#00H
        LD      SP,EA
        •
        •
        •
        LD      A,#0DH          ;  WDCNT input clock is 7.82 ms
        LD      BMOD,A
        •
        •
        •
MAIN    BITS    WDTCF           ;  Main routine operation period must be shorter than
                                ;  watchdog timer's period
        •
        •
        •
        JP      MAIN
```

8-BIT TIMER/COUNTER 0 (TC0)

OVERVIEW

Timer/counter 0 (TC0) is used to count system 'events' by identifying the transition (high-to-low or low-to-high) of incoming square wave signals. To indicate that an event has occurred, or that a specified time interval has elapsed, TC0 generates an interrupt request. By counting signal transitions and comparing the current counter value with the reference register value, TC0 can be used to measure specific time intervals.

TC0 has a reloadable counter that consists of two parts: an 8-bit reference register (TREF0) into which you write the counter reference value, and an 8-bit counter register (TCNT0) whose value is automatically incremented by counter logic.

An 8-bit mode register, TMOD0, is used to activate the timer/counter and to select the basic clock frequency to be used for timer/counter operations. To dynamically modify the basic frequency, new values can be loaded into the TMOD0 register during program execution.

TC0 FUNCTION SUMMARY

8-bit programmable timer	Generates interrupts at specific time intervals based on the selected clock frequency.
Arbitrary frequency output	Outputs selectable clock frequencies to the TC0 output pin, TCLO0/TCLO0.

TC0 COMPONENT SUMMARY

Mode register (TMOD0)	Activates the timer/counter and selects the internal clock frequency or the external clock source at the TCL0 pin.
Reference register (TREF0)	Stores the reference value for the desired number of clock pulses between interrupt requests.
Counter register (TCNT0)	Counts internal or external clock pulses based on the bit settings in TMOD0 and TREF0.
Clock selector circuit	Together with the mode register (TMOD0), lets you select one of four internal clock frequencies or an external clock.
8-bit comparator	Determines when to generate an interrupt by comparing the current value of the counter register (TCNT0) with the reference value previously programmed into the reference register (TREF0).
Output latch (TOL0)	Where a clock pulse is stored pending output to the serial I/O circuit or to the TC0 output pin, TCLO0. When the contents of the TCNT0 and TREF0 registers coincide, the timer/counter interrupt request flag (IRQT0) is set to "1", the status of TOL0 is inverted, and an interrupt is generated.
Output enable flag (TOE/TOEB)	Must be set to logic one before the contents of the TOL0 latch can be output to TCLO0/TCLO0.
Interrupt request flag (IRQT0)	Cleared when TC0 operation starts and the TC0 interrupt service routine is executed and set to 1 whenever the counter value and reference value coincide.
Interrupt enable flag (IET0)	Must be set to logic one before the interrupt requests generated by timer/counter 0 can be processed.

Table 11-4. TC0 Register Overview

Register Name	Type	Description	Size	RAM Address	Addressing Mode	Reset Value
TMOD0	Control	Controls TC0 enable/disable (bit 2); clears and resumes counting operation (bit 3); sets input clock and clock frequency (bits 6–4)	8-bit	F90H–F91H	8-bit write only; (TMOD0.3 is also 1-bit writeable)	"0"
TCNT0	Counter	Counts clock pulses matching the TMOD0 frequency setting	8-bit	F94H–F95H	8-bit read-only	"0"
TREF0	Reference	Stores reference value for the timer/counter 0 interval setting	8-bit	F96H–F97H	8-bit write-only	FFH
TOE	Flag	Controls timer/counter 0 output to the TCLO0 pin	1-bit	F92H.2	1/4-bit read/write	"0"
TOEB	Flag	Controls converted timer/counter 0 output to the TCLO0 pin	1-bit	F92H.3	1/4-bit read/write	"0"

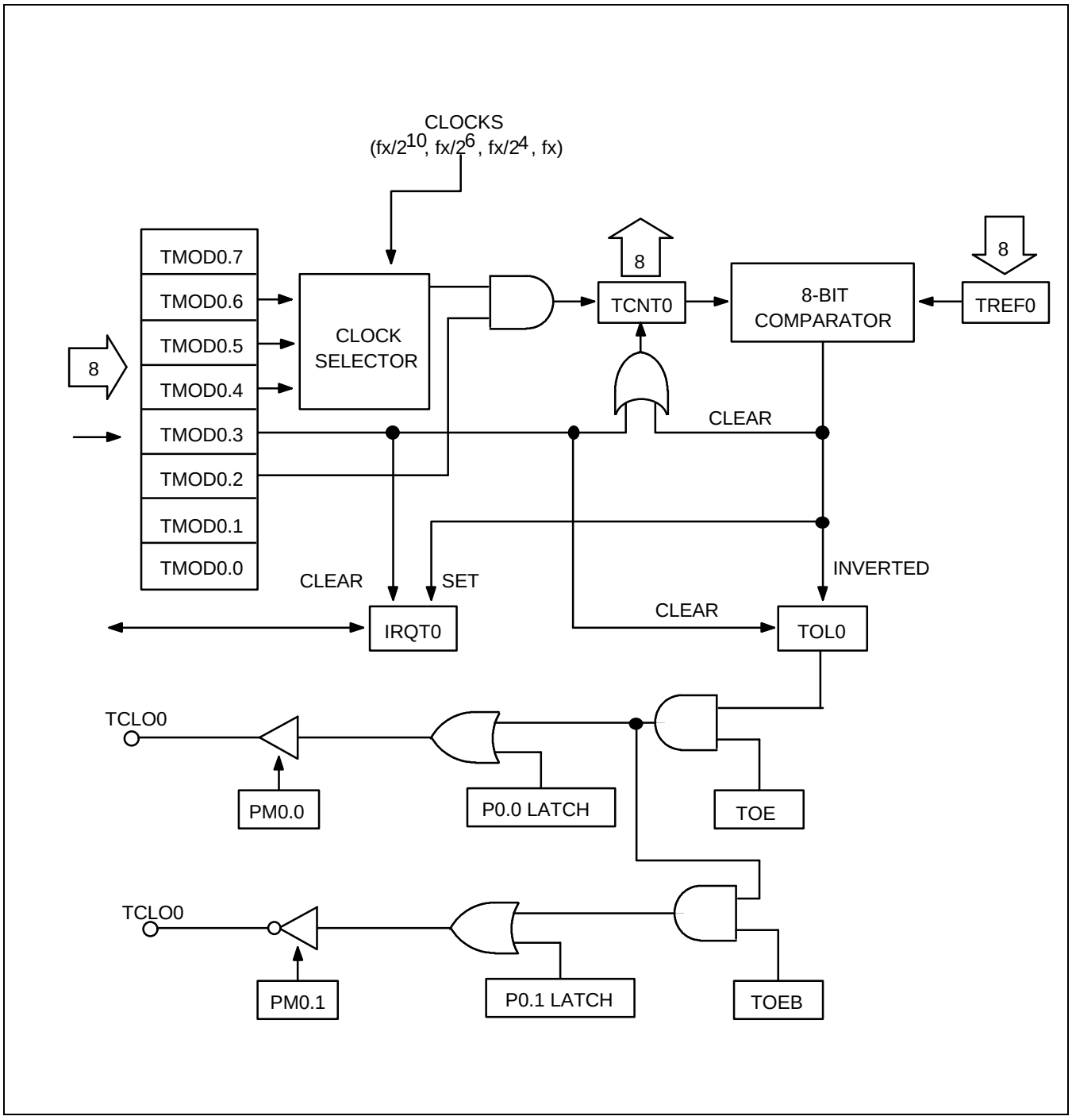


Figure 11-2. TC0 Circuit Diagram

TC0 ENABLE/DISABLE PROCEDURE

Enable Timer/Counter 0

— Set TMOD0.2 to logic one

— Set the TC0 interrupt enable flag IET0 to logic one

— Set TMOD0.3 to logic one

TCNT0, IRQT0, and TOL0 are cleared to logic zero, and timer/counter operation starts.

Disable Timer/Counter 0

— Set TMOD0.2 to logic zero

Clock signal input to the counter register TCNT0 is halted. The current TCNT0 value is retained and can be read if necessary.

TC0 PROGRAMMABLE TIMER/COUNTER FUNCTION

Timer/counter 0 can be programmed to generate interrupt requests at various intervals based on the selected system clock frequency. Its 8-bit TC0 mode register TMOD0 is used to activate the timer/counter and to select the clock frequency. The reference register TREF0 stores the value for the number of clock pulses to be generated between interrupt requests. The counter register, TCNT0, counts the incoming clock pulses, which are compared to the TREF0 value as TCNT0 is incremented. When there is a match (TREF0 = TCNT0), an interrupt request is generated.

To program timer/counter 0 to generate interrupt requests at specific intervals, choose one of four internal clock frequencies (divisions of the system clock, fxx) and load a counter reference value into the TREF0 register. TCNT0 is incremented each time an internal counter pulse is detected with the reference clock frequency specified by TMOD0.4–TMOD0.6 settings. To generate an interrupt request, the TC0 interrupt request flag (IRQT0) is set to logic one, the status of TOL0 is inverted, and the interrupt is generated. The content of TCNT0 is then cleared to 00H and TC0 continues counting. The interrupt request mechanism for TC0 includes an interrupt enable flag (IET0) and an interrupt request flag (IRQT0).

TC0 OPERATION SEQUENCE

The general sequence of operations for using TC0 can be summarized as follows:

1. Set TMOD0.2 to "1" to enable TC0.
2. Set TMOD0.6 to "1" to enable the system clock (fxx) input.
3. Set TMOD0.5 and TMOD0.4 bits to desired internal frequency ($fxx/2^n$).
4. Load a value to TREF0 to specify the interval between interrupt requests.
5. Set the TC0 interrupt enable flag (IET0) to "1".
6. Set TMOD0.3 bit to "1" to clear TCNT0, IRQT0, and TOL0, and start counting.
7. TCNT0 increments with each internal clock pulse.
8. When the comparator shows TCNT0 = TREF0, the IRQT0 flag is set to "1" and an interrupt request is generated.
9. Output latch (TOL0) logic toggles high or low.
10. TCNT0 is cleared to 00H and counting resumes.
11. Programmable timer/counter operation continues until TMOD0.2 is cleared to "0".

TC0 Clock Frequency Output

Using timer/counter 0, a modifiable clock frequency can be output to the TC0 clock output pin, TCLO0. To select the clock frequency, load the appropriate values to the TC0 mode register, TMOD0. The clock interval is selected by loading the desired reference value into the reference register TREF0. To enable the output to the TCLO0 pin, the following conditions must be met:

— TC0 output enable flag TOE must be set to "1"

— I/O mode flag for P0.0 (PM0.0) must be set to output mode ("1")

— Output latch value for P0.0 must be set to "0"

In summary, the operational sequence required to output a TC0-generated clock signal to the TCLO0 pin is as follows:

1. Load a reference value to TREF0.
2. Set the internal clock frequency in TMOD0.
3. Initiate TC0 clock output to TCLO0 (TMOD0.2 = "1").
4. Set P0.0 mode flag (PM0.0) to "1".
5. Set P0.0 output latch to "0".
6. Set TOE flag to "1".

Each time TCNT0 overflows and an interrupt request is generated, the state of the output latch TOL0 is inverted and the TC0-generated clock signal is output to the TCLO0 pin.

☞ PROGRAMMING TIP — TC0 Signal Output to The TCLO0 Pin

Output a 30 ms pulse width signal to the TCLO0 pin:

```
BITS    EMB
SMB     15
LD      EA,#79H
LD      TREF0,EA
LD      EA,#4CH
LD      TMOD0,EA
LD      EA,#01H
LD      PMG2,EA          ;  P0.0 ← output mode
BITR    P0.0             ;  P0.0 clear
BITS    TOE
```

TC0 MODE REGISTER (TMOD0)

TMOD0 is the 8-bit mode control register for timer/counter 0. It is addressable by 8-bit write instructions. One bit, TMOD0.3, is also 1-bit writeable. RESET clears all TMOD0 bits to logic zero and disables TC0 operations.

F90H	TMOD0.3	TMOD0.2	"0"	"0"
F91H	"0"	TMOD0.6	TMOD0.5	TMOD0.4

TMOD0.2 is the enable/disable bit for timer/counter 0. When TMOD0.3 is set to "1", the contents of TCNT0, IRQT0, and TOL0 are cleared, counting starts from 00H, and TMOD0.3 is automatically reset to "0" for normal TC0 operation. When TC0 operation stops (TMOD0.2 = "0"), the contents of the TC0 counter register TCNT0 are retained until TC0 is re-enabled.

The TMOD0.6, TMOD0.5, and TMOD0.4 bit settings are used together to select the TC0 clock source. This selection involves two variables:

— Synchronization of timer/counter operations with either the rising edge or the falling edge of the clock signal input at the TCL0 pin, and

— Selection of one of four frequencies, based on division of the incoming system clock frequency, for use in internal TC0 operation.

Table 11-5. TC0 Mode Register (TMOD0) Organization

Bit Name	Setting	Resulting TC0 Function	Address
TMOD0.7	0	Always logic zero	F91H
TMOD0.6 TMOD0.5 TMOD0.4	0,1	Specify input clock edge and internal frequency	
TMOD0.3	1	Clear TCNT0, IRQT0, and TOL0 and resume counting immediately (This bit is automatically cleared to logic zero immediately after counting resumes.)	F90H
TMOD0.2	0	Disable timer/counter 0; retain TCNT0 contents	
	1	Enable timer/counter 0	
TMOD0.1	0	Always logic zero	
TMOD0.0	0	Always logic zero	

Table 11-6. TMOD0.6, TMOD0.5, and TMOD0.4 Bit Settings

TMOD0.6	TMOD0.5	TMOD0.4	Resulting Counter Source and Clock Frequency
0	0	0	Not available
0	0	1	Not available
1	0	0	Select clock fxx/2^{10} (4.09 kHz)
1	0	1	Select clock fxx /2^{6} (65.5 kHz)
1	1	0	Select clock fxx/2^{4} (262 kHz)
1	1	1	Select clock fxx (4.19 MHz)

NOTE: 'fxx' = system clock frequency (assume that fxx is 4.19 MHz).

☞ PROGRAMMING TIP — Restarting TC0 Counting Operation

1. Set TC0 timer interval to 4.09 kHz:

```
BITS    EMB
SMB     15
LD      EA,#4CH
LD      TMOD0,EA
EI
BITS    IET0
```

2. Clear TCNT0, IRQT0, and TOL0 and restart TC0 counting operation:

```
BITS    EMB
SMB     15
BITS    TMOD0.3
```

TC0 COUNTER REGISTER (TCNT0)

The 8-bit counter register for timer/counter 0, TCNT0, is read-only and can be addressed by 8-bit RAM control instructions. RESET sets all TCNT0 register values to logic zero (00H).

Whenever TMOD0.3 is enabled, TCNT0 is cleared to logic zero and counting resumes. The TCNT0 register value is incremented each time an incoming clock signal is detected that matches the signal edge and frequency setting of the TMOD0 register (specifically, TMOD0.6, TMOD0.5, and TMOD0.4).

Each time TCNT0 is incremented, the new value is compared to the reference value stored in the TC0 refer-ence buffer, TREF0. When TCNT0 = TREF0, an overflow occurs in the TCNT0 register, the interrupt request flag, IRQT0, is set to logic one, and an interrupt request is generated to indicate that the specified timer/counter interval has elapsed.

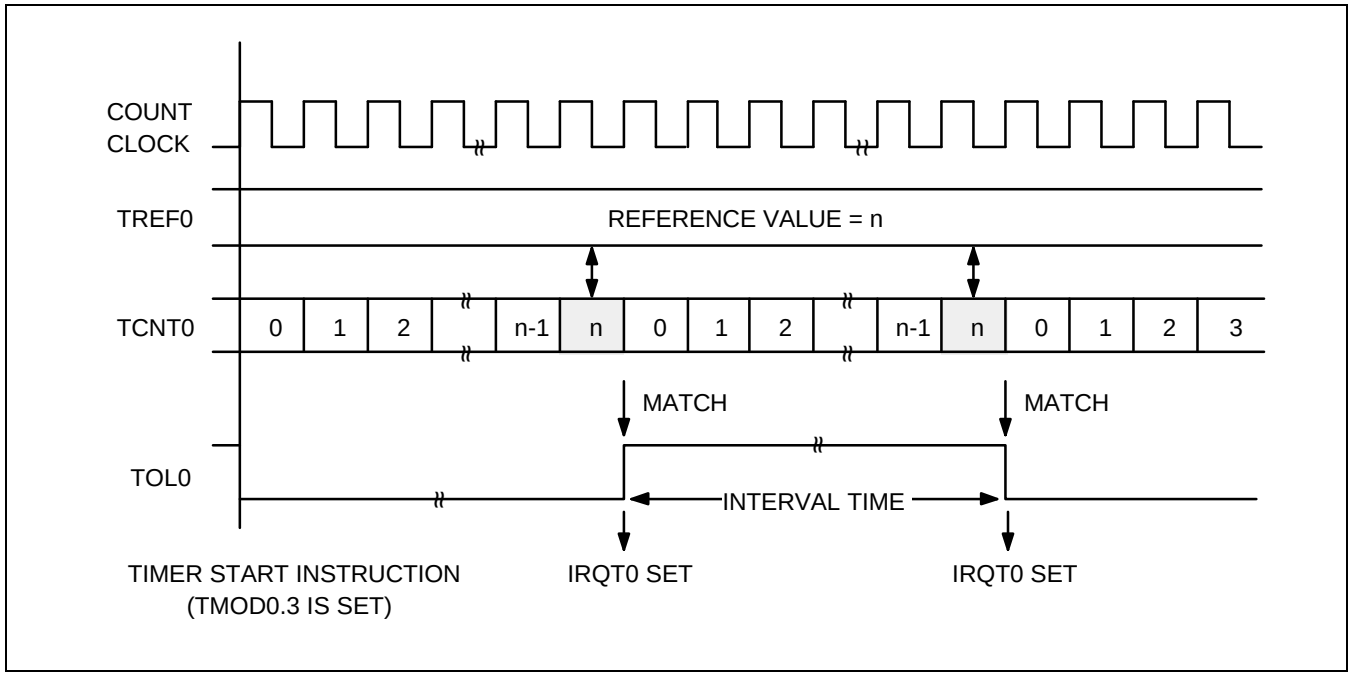


Figure 11-3. TC0 Timing Diagram

TC0 REFERENCE REGISTER (TREF0)

The TC0 reference register TREF0 is an 8-bit write-only register. It is addressable by 8-bit RAM control instructions. RESET initializes the TREF0 value to 'FFH'.

TREF0 is used to store a reference value to be compared to the incrementing TCNT0 register in order to identify an elapsed time interval. Reference values will differ depending upon the specific function that TC0 is being used to perform — as a programmable timer/counter, event counter, clock signal divider, or arbitrary frequency output source.

During timer/counter operation, the value loaded into the reference register is compared to the TCNT0 value. When TCNT0 = TREF0, the TC0 output latch (TOL0) is inverted and an interrupt request is generated to signal the interval or event. The TREF0 value, together with the TMOD0 clock frequency selection, determines the specific TC0 timer interval. Use the following formula to calculate the correct value to load to the TREF0 reference register:

$$\text{TC0 timer interval} = (\text{TREF0 value} + 1) \times \frac{1}{\text{TMOD0 frequency setting}}$$

$$(\text{TREF0 value} \neq 0)$$

TC0 OUTPUT ENABLE FLAG (TOEB, TOE)

The 1-bit timer/counter 0 output enable flag TOE controls output from timer/counter 0 to the TCLO0 pin, and 1-bit inverted timer/counter 0 output enable flag TOEB controls output from timer/counter 0 to the TCLO0 pin.

F92H	**TOEB**	**TOE**	"U"	**BUZB**

NOTE: The “U” means a undefined register bit.

When you set the TOE flag to "1", the contents of TOL0 can be output to the TCLO0. When you set the TOEB flag to "1", the contents of inverted TOL0 can be output to the TCLO0.

NOTE:

If you set TOE flag to "0", the contents of TOL0 and inverted TOL0 can not be output to the TCLO0 and TCLO0, respectively.

TC0 OUTPUT LATCH (TOL0)

TOL0 is the output latch for timer/counter 0. When the 8-bit comparator detects a correspondence between the value of the counter register TCNT0 and the reference value stored in the TREF0 register, the TOL0 value is inverted — the latch toggles high-to-low or low-to-high. Whenever the state of TOL0 is switched, the TC0 signal is output. TC0 output may be directed to the TCLO0 pin.

Assuming TC0 is enabled, when bit 3 of the TMOD0 register is set to "1", the TOL0 latch is cleared to logic zero, along with the counter register TCNT0 and the interrupt request flag, IRQT0, and counting resumes immediately. When TC0 is disabled (TMOD0.2 = "0"), the contents of the TOL0 latch are retained and can be read, if necessary.

☞ PROGRAMMING TIP — Setting a TC0 Timer Interval

To set a 30 ms timer interval for TC0, given fxx = 4.19 MHz, follow these steps.

1. Select the timer/counter 0 mode register with a maximum setup time of 62.5 ms (assume the TC0 counter clock = fxx/2^{10}, and TREF0 is set to FFH):

2. Calculate the TREF0 value:

$$30 \text{ ms} = \frac{\text{TREF0 value} + 1}{4.09 \text{ kHz}}$$

$$\text{TREF0} + 1 = \frac{30 \text{ ms}}{244 \ \mu\text{s}} = 122.9 = 7\text{AH}$$

$$\text{TREF0 value} = 7\text{AH} - 1 = 79\text{H}$$

3. Load the value 79H to the TREF0 register:

```
BITS    EMB
SMB     15
LD      EA,#79H
LD      TREF0,EA
LD      EA,#4CH
LD      TMOD0,EA
```

WATCH TIMER

OVERVIEW

The watch timer is a multi-purpose timer which consists of three basic components:

— 8-bit watch timer mode register (WMOD)

— Clock selector

— Frequency divider circuit

Watch timer functions include real-time and watch-time measurement and interval timing for the main and sub-system clock. It is also used as a clock source for the LCD controller and for generating buzzer (BUZ) output.

Real-Time and Watch-Time Measurement

To start watch timer operation, set bit 2 of the watch timer mode register (WMOD.2) to logic one. The watch timer starts, the interrupt request flag IRQW is automatically set to logic one, and interrupt requests commence in 0.5-second intervals.

Since the watch timer functions as a quasi-interrupt instead of a vectored interrupt, the IRQW flag should be cleared to logic zero by program software as soon as a requested interrupt service routine has been executed.

Using a Main System or Subsystem Clock Source

The watch timer can generate interrupts based on the main system clock frequency or on the subsystem clock. When the zero bit of the WMOD register is set to "1", the watch timer uses the subsystem clock signal (fxt) as its source; if WMOD.0 = "0", the main system clock (fx) is used as the signal source, according to the following formula:

$$\text{Watch timer clock (fw)} = \frac{\text{Main system clock (fx)}}{128} = 32.768\ \text{kHz (fx = 4.19 MHz)}$$

This feature is useful for controlling timer-related operations during stop mode. When stop mode is engaged, the main system clock (fx) is halted, but the subsystem clock continues to oscillate. By using the subsystem clock as the oscillation source during stop mode, the watch timer can set the interrupt request flag IRQW to "1", thereby releasing stop mode.

Clock Source Generation for LCD Controller

The watch timer supplies the clock frequency for the LCD controller (f_{LCD}). Therefore, if the watch timer is disabled, the LCD controller does not operate.

Buzzer Output Frequency Generator

The watch timer can generate a steady 2 kHz, 4 kHz, 8 kHz, or 16 kHz signal to the BUZ and BUZ pins. To select the desired BUZ frequency , load the appropriate value to the WMOD register. This output can then be used to actuate an external buzzer sound. To generate a BUZ signal, three conditions must be met:

— The WMOD.7 register bit is set to "1"

— The output latch for I/O port 0.3 is cleared to "0"

— The port 0.3 output mode flag (PM0.3) set to 'output' mode

To generate a BUZ signal, four conditions must be met:

— The WMOD.7 register bit is set to "1"

— The output latch for I/O port 0.2 is cleared to "0"

— The port 0.2 output mode flag (PM0.2) is set to output mode

— BUZB flag is set to "1"

Timing Tests in High-Speed Mode

By setting WMOD.1 to "1", the watch timer will function in high-speed mode, generating an interrupt every 3.91 ms. At its normal speed (WMOD.1 = '0'), the watch timer generates an interrupt request every 0.5 seconds. High-speed mode is useful for timing events for program debugging sequences.

Check Subsystem Clock Level Feature

The watch timer can also check the input level of the subsystem clock by testing WMOD.3. If WMOD.3 is "1", the input level at the XT_{in} pin is high; if WMOD.3 is "0", the input level at the XT_{in} pin is low.

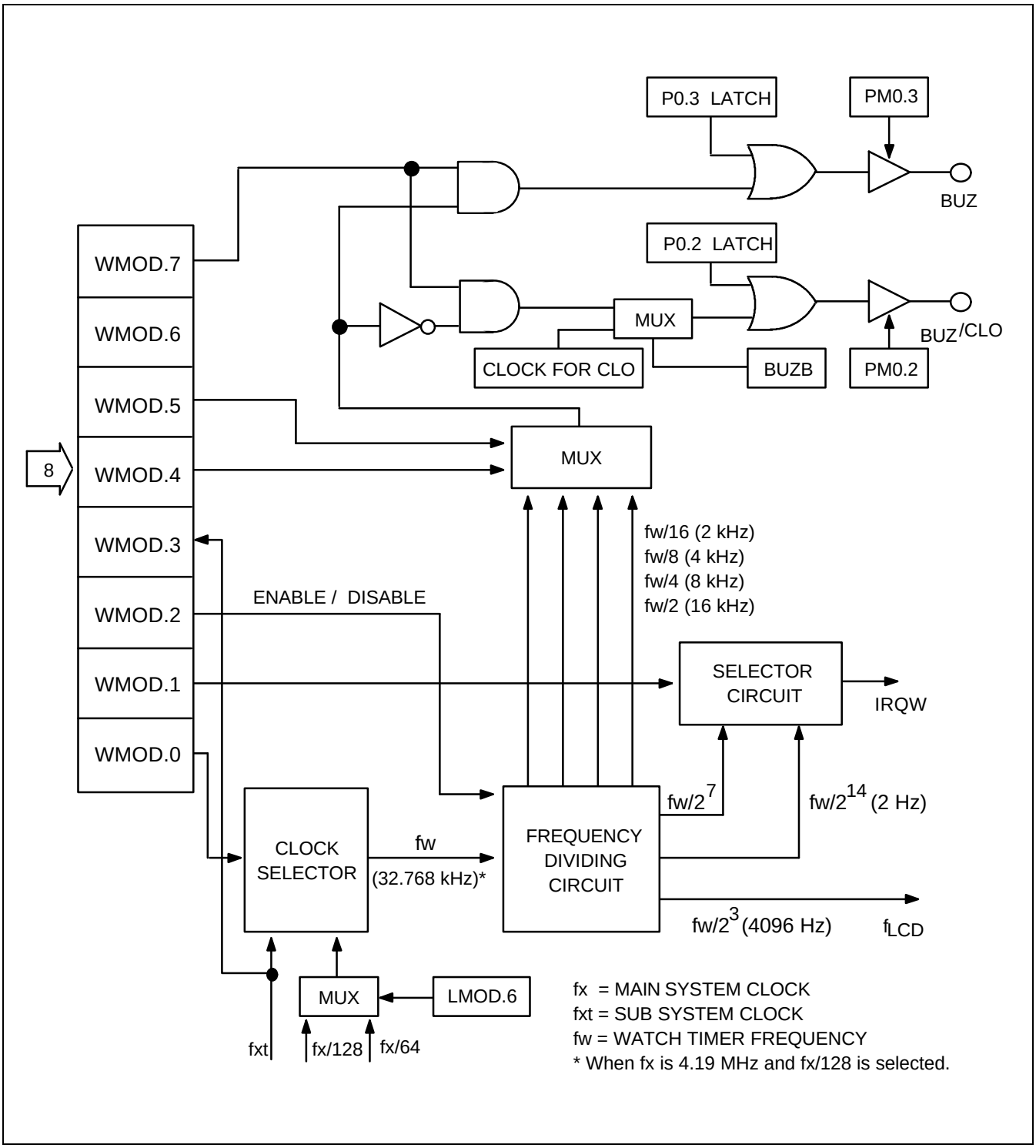


Figure 11-4. Watch Timer Circuit Diagram

WATCH TIMER MODE REGISTER (WMOD)

The watch timer mode register WMOD is used to select specific watch timer operations. It is 8-bit write-only addressable. An exception is WMOD bit 3 (the XT_{in} input level control bit) which is 1-bit read-only addressable. A RESET automatically sets WMOD.3 to the current input level of the subsystem clock, XT_{in} (high, if logic one; low, if logic zero), and all other WMOD bits to logic zero.

F88H	WMOD.3	WMOD.2	WMOD.1	WMOD.0
F89H	WMOD.7	"0"	WMOD.5	WMOD.4

In summary, WMOD settings control the following watch timer functions:

— Watch timer clock selection (WMOD.0)
— Watch timer speed control (WMOD.1)
— Enable/disable watch timer (WMOD.2)
— XT_{in} input level control (WMOD.3)
— Buzzer frequency selection (WMOD.4 and WMOD.5)
— Enable/disable buzzer output (WMOD.7)

Table 11-7. Watch Timer Mode Register (WMOD) Organization

Bit Name	Values		Function	Address
WMOD.7	0		Disable buzzer (BUZ) signal output at the BUZ pin	F89H
	1		Enable buzzer (BUZ) signal output at the BUZ pin	
WMOD.6	0		Always logic zero	
WMOD.5 – .4	0	0	2 kHz buzzer (BUZ/BUZ) signal output	
	0	1	4 kHz buzzer (BUZ/BUZ) signal output	
	1	0	8 kHz buzzer (BUZ/BUZ) signal output	
	1	1	16 kHz buzzer (BUZ/BUZ) signal output	
WMOD.3	0		Input level to XT_{in} pin is low	F88H
	1		Input level to XT_{in} pin is high	
WMOD.2	0		Disable watch timer; clear frequency dividing circuits	
	1		Enable watch timer	
WMOD.1	0		Normal mode; sets IRQW to 0.5 seconds	
	1		High-speed mode; sets IRQW to 3.91 ms	
WMOD.0	0		Select (fx/128) as the watch timer clock (fw)	
	1		Select subsystem clock as watch timer clock (fw)	

NOTES:
1. Main system clock frequency (fx) is assumed to be 4.19 MHz; subsystem clock (fxt) is assumed to be 32.768 kHz.
2. If you set WMOD.7 to "0", the signals of BUZ and inverted BUZ can not be output to the BUZ pin and BUZ pin,

respectively.

☞ PROGRAMMING TIP — Using The Watch Timer

1. Select a subsystem clock as the LCD display clock, a 0.5 second interrupt, and 2 kHz buzzer enable:

```
        BITS    EMB
        SMB     15
        LD      EA,#8H
        LD      PMG1,EA         ;  P0.3 ← output mode
        BITR    P0.3
        LD      EA,#85H
        LD      WMOD,EA
        BITS    IEW
```

2. Sample real-time clock processing method:

```
CLOCK   BTSTZ   IRQW             ;  0.5 second check
                RET              ;  No, return
        •                        ;  Yes, 0.5 second interrupt generation
        •
        •                        ;  Increment HOUR, MINUTE, SECOND
```

12 LCD CONTROLLER/DRIVER

OVERVIEW

The S3C7295 microcontroller can directly drive an up-to-704-dot (44 segments x 16 commons) LCD panel. Its LCD block has the following components:

— LCD controller/driver
— Display RAM for storing display data
— 44 segment output pins (SEG0–SEG43)
— 16 common output pins (COM0–COM15)
— Internal resistor circuit for LCD bias
— V_{LC0} pin for controlling the driver and bias voltage
— Voltage doubler for LCD (BIAS)

To use the LCD controller, bit 2 in the watch mode register WMOD must be set to 1 because LCDCK is supplied by the watch timer.

The frame frequency, duty and bias, and the segment pins used for display output, are determined by bit settings in the LCD mode register, LMOD.

The LCD control register, LCON, is used to turn the LCD display on and off, to switch current to the dividing resistors for the LCD display, and to turn the voltage doubler on and off for LCD bias voltage. Data written to the LCD display RAM can be transferred to the segment signal pins automatically without program control.

When a subsystem clock is selected as the LCD clock source, the LCD display is enabled even during main clock stop and idle modes.

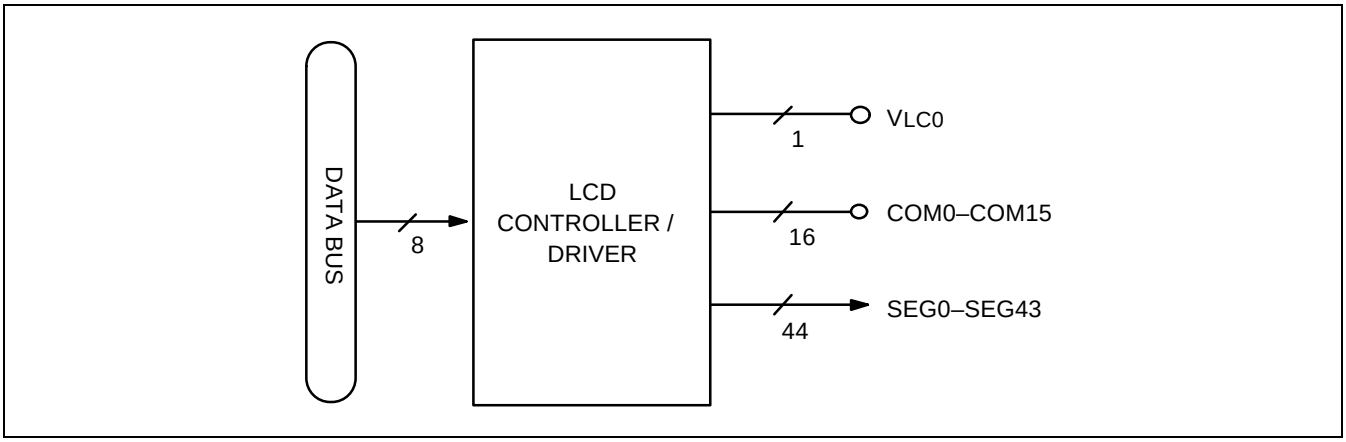


Figure 12-1. LCD Function Diagram

LCD CIRCUIT DIAGRAM

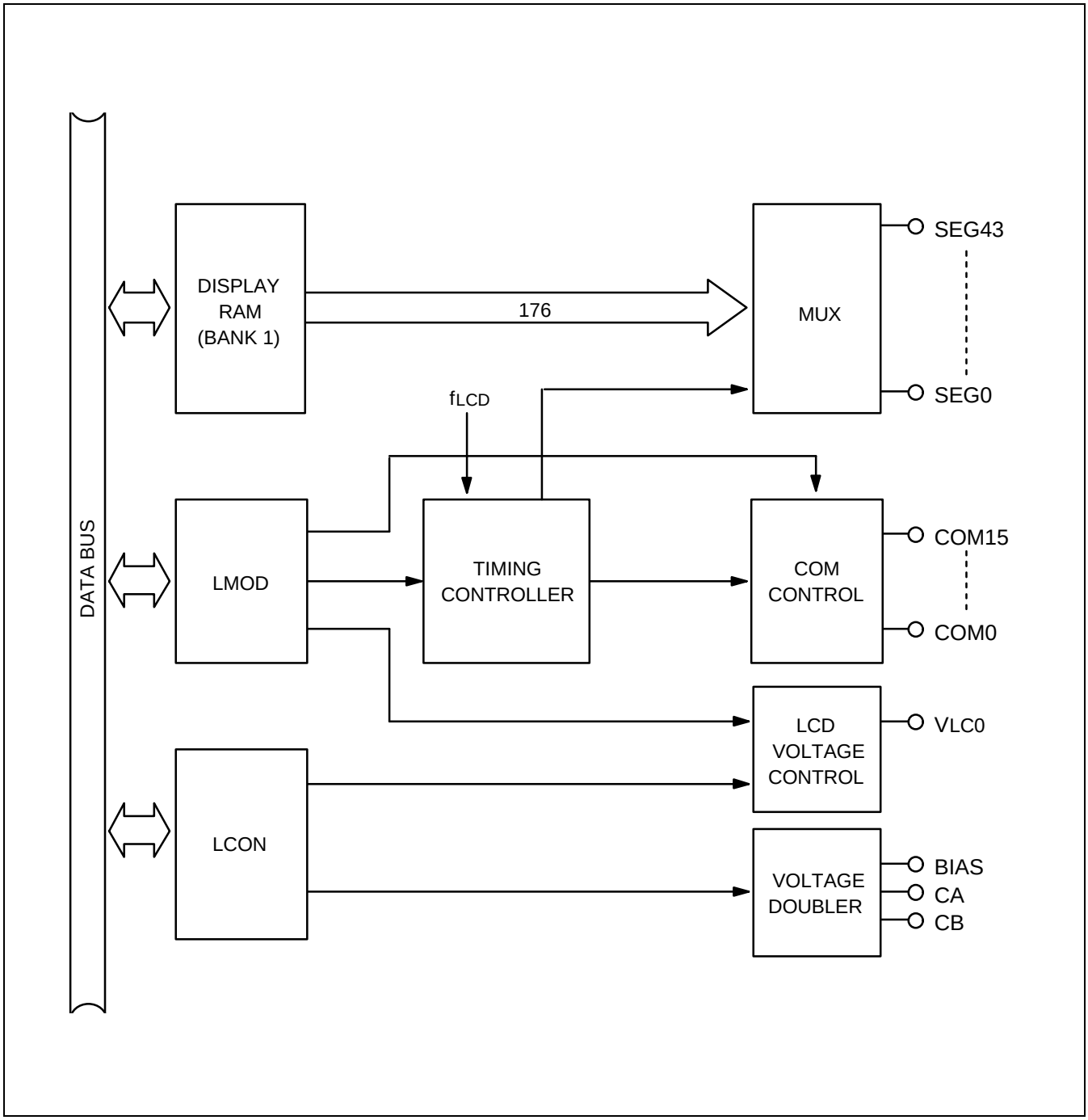


Figure 12-2. LCD Circuit Diagram

LCD RAM ADDRESS AREA

RAM addresses of bank 1 are used as LCD data memory. These locations can be addressed by 1-bit or 4-bit instructions. When the bit value of a display segment is "1", the LCD display is turned on; when the bit value is "0", the display is turned off.

Display RAM data are sent out through segment pins SEG0–SEG43 using a direct memory access (DMA) method that is synchronized with the f_{LCD} signal. RAM addresses in this location that are not used for LCD display can be
allocated to general-purpose use.

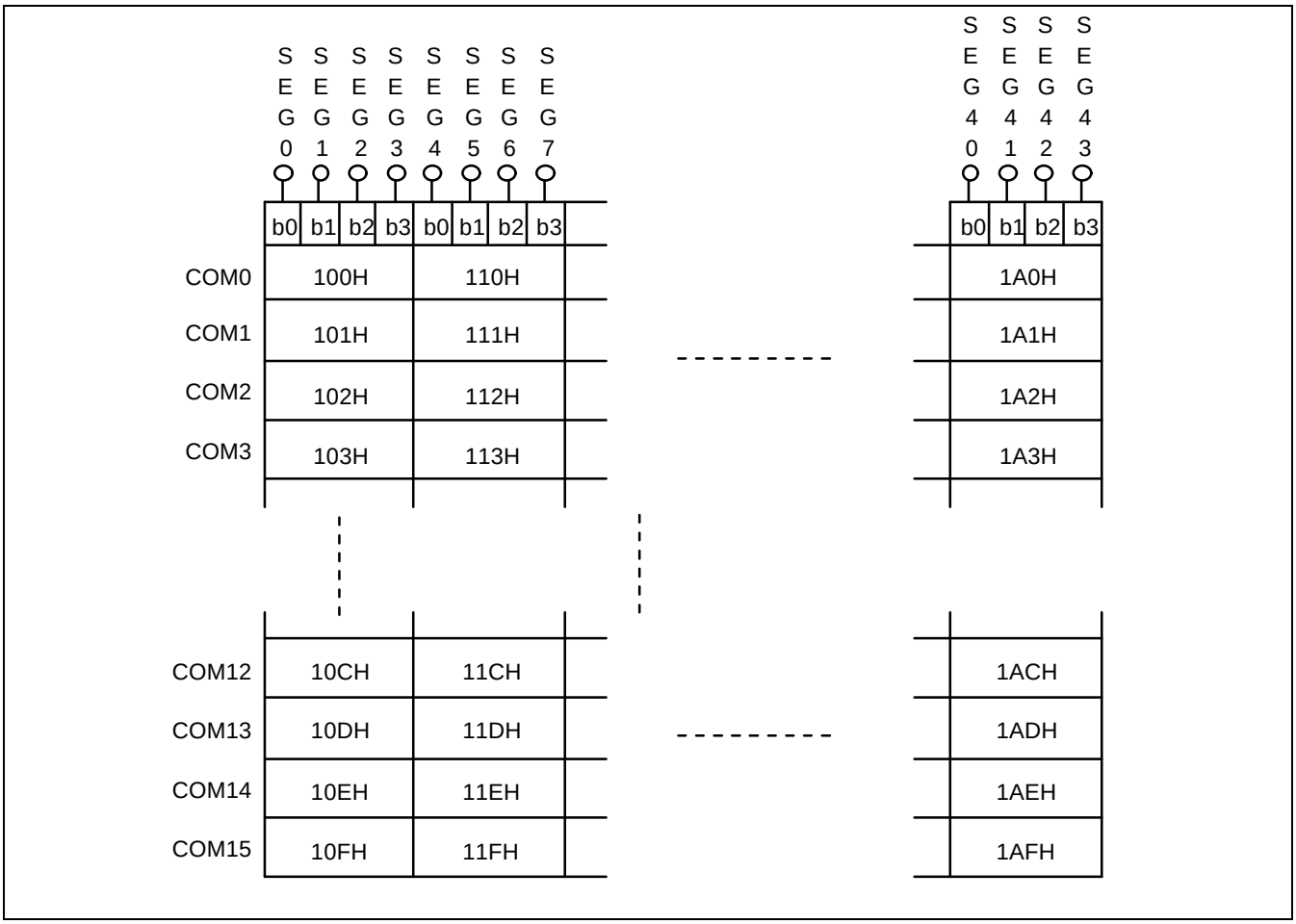


Figure 12-3. LCD Display Data RAM Organization

Table 12-1. Common and Segment Pins per Duty Cycle

Duty	Common Pins	Segment Pins	Dot Number
1/16	COM0–COM15	44 pins	704 dots
1/12	COM0–COM11		528 dots
1/8	COM0–COM7		352 dots

LCD CONTROL REGISTER (LCON)

The LCD control register (LCON) is used to turn the LCD display on and off, and to control the flow of current to dividing resistors in the LCD circuit. Following a RESET, all LCON values are cleared to "0". This turns the LCD display off and stops the flow of current to the dividing resistors.

F8EH	"0"	LCON.2	LCON.1	LCON.0

The effect of the LCON.0 setting is dependent upon the current setting of bits LMOD.0 and LMOD.1.

Table 12-2. LCD Control Register (LCON) Organization

LCON Bit	Setting	Description
LCON.3	0	Always logic zero.
LCON.2	0	Normal LCD dividing resistors
	1	Diminish a LCD dividing resistor to strengthen LCD drive
LCON.1 (note)	0	Only V_{DD} for V_{LC0} (voltage pumping circuit is off)
	1	Only BIAS for V_{LC0} (voltage pumping circuit is operated)
LCON.0	0	Display off (cut off the voltage dividing resistors)
	1	Normal display on

NOTE: The voltage pumping circuit requires additional current to change the external capacitor.

Table 12-3. LMOD.1–0 Bits Settings

LMOD.1–0	COM0–COM15	SEG0–SEG43	Power Supply to the Dividing Resistor
0, 0	All of the LCD dots off		On
0, 1	All of the LCD dots on		
1, 1	Common and segment signal output corresponds to display data (normal display mode)		

LCD MODE REGISTER (LMOD)

The LCD mode control register LMOD is used to control display mode; LCD clock, and display on/off. LMOD can be manipulated using 8-bit write instructions.

F8CH	LMOD.3	LMOD.2	LMOD.1	LMOD.0
F8DH	LMOD.7	LMOD.6	LMOD.5	LMOD.4

The LCD clock signal, LCDCK, determines the frequency of COM signal scanning of each segment output. This is also referred to as the 'frame frequency. Since LCDCK is generated by dividing the watch timer clock (fw), the watch timer must be enabled when the LCD display is turned on. RESET clears the LMOD register values to logic zero.

The LCD display can continue to operate during idle and stop modes if a subsystem clock is used as the watch timer source.

Table 12-4. LCD Clock Signal (LCDCK) Frame Frequency

LCDCK / Display Duty Cycle	256 Hz	512 Hz	1024 Hz	2048 Hz	4096 Hz
1/8	32	64	128	256	–
1/12	–	42.7	85.3	170.7	341.3
1/16	–	32	64	128	256

NOTE:

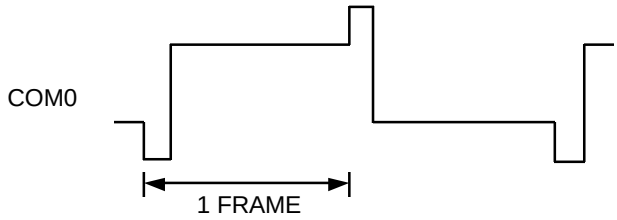


Table 12-5. LCD Mode Register (LMOD) Organization (8-bit W)

LCD bias Selection Bit

LMOD.7	Bias Selection for LCD Display
0	1/4 bias
1	1/5 bias

Watch Timer (fw) Clock Selection Bits (When main system clock is supplied to the watch timer source.)

LMOD.6	Watch timer clock when main system clock is selected as watch timer clock
0	f_{LCD} = 4096 Hz when fw = fx/128 (32.768 kHz @fx = 4.19 MHz)
1	f_{LCD} = 8192 Hz when fw = fx/64 (65.563 kHz @fx = 4.19 MHz)

LCD Clock Selection Bits

LMOD.5	LMOD.4	LCD Clock (LCDCK)		
		1/8 duty (COM0–COM7)	1/12 duty (COM0–COM11)	1/16 duty (COM0–COM15)
0	0	fw / 2^7 (256 Hz)	fw / 2^6 (512 Hz)	fw / 2^6 (512 Hz)
0	1	fw/ 2^6 (512 Hz)	fw/ 2^5 (1024 Hz)	fw/ 2^5 (1024 Hz)
1	0	fw / 2^5 (1024 Hz)	fw / 2^4 (2048 Hz)	fw / 2^4 (2048 Hz)
1	1	fw / 2^4 (2048 Hz)	fw / 2^3 (4096 Hz)	fw / 2^3 (4096 Hz)

NOTES:
1. LCDCK is supplied only when the watch timer is operating. To use the LCD controller, you must set bit 2 in the watch mode register WMOD to "1".
2. When fw = 32.768 kHz, f_{LCD} = 4096 Hz.

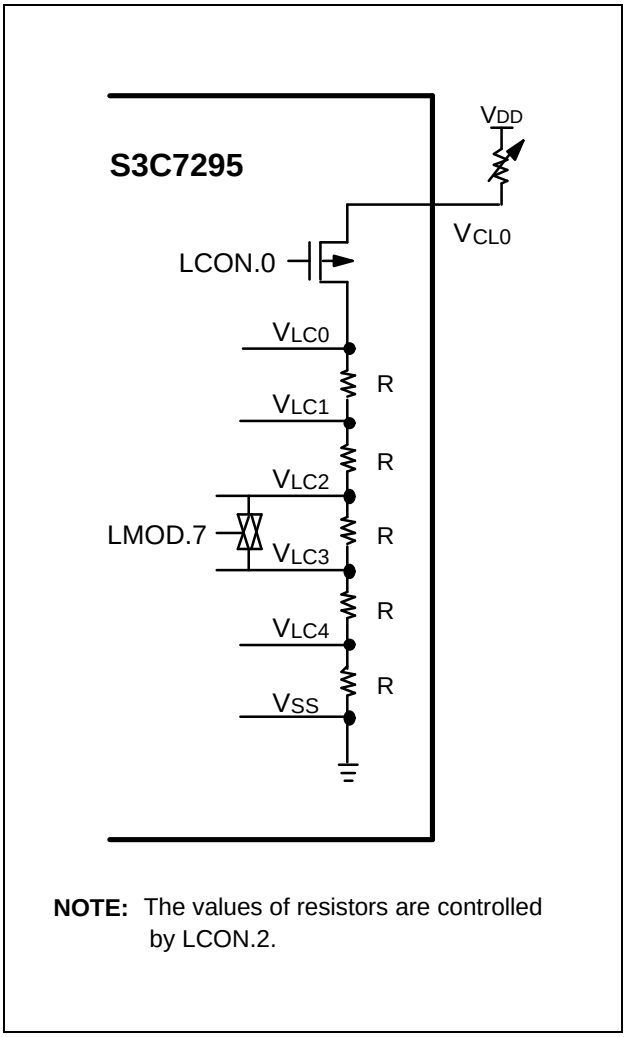
Duty Selection Bits

LMOD.3	LMOD.2	Duty
0	0	1/8 duty (COM0–COM7 select)
0	1	1/12 duty (COM0–COM11 select)
1	0	1/16 duty (COM0–COM15 select)

Display Mode Selection Bits

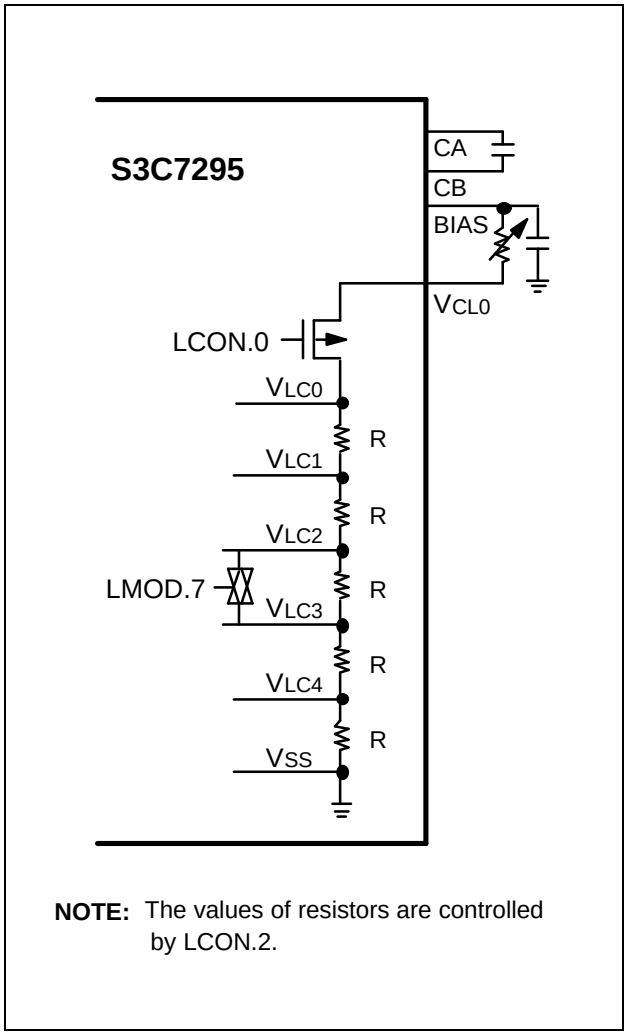
LMOD.1	LMOD.0	COM0–COM15	SEG0–SEG43	Power Supply to the Dividing Resistor
0	0	All of the LCD dots off		On
0	1	All of the LCD dots on		
1	1	Common and segment signal output corresponds to display data (normal display mode)		

LCD VOLTAGE DIVIDING RESISTORS



NOTE: The values of resistors are controlled by LCON.2.

Figure 12-4. Internal Voltage Dividing Resistor Connection (LCON.1 = 0, Voltage Pumping Circuit is Off)



NOTE: The values of resistors are controlled by LCON.2.

Figure 12-5. Internal Voltage Dividing Resistor Connection (LCON.1 = 1, Voltage Pumping Circuit is On)

COMMON (COM) SIGNALS

The common signal output pin selection (COM pin selection) varies according to the selected duty cycle.

— In 1/8 duty mode, COM0–COM7 pins are selected

— In 1/12 duty mode, COM0–COM11 pins are selected

— In 1/16 duty mode, COM0–COM15 pins are selected

SEGMENT (SEG) SIGNALS

The 44 LCD segment signal pins are connected to corresponding display RAM locations at bank 1. Bits of the display RAM are synchronized with the common signal output pins.

When the bit value of a display RAM location is "1", a select signal is sent to the corresponding segment pin. When the display bit is "0", a 'no-select' signal is sent to the corresponding segment pin.

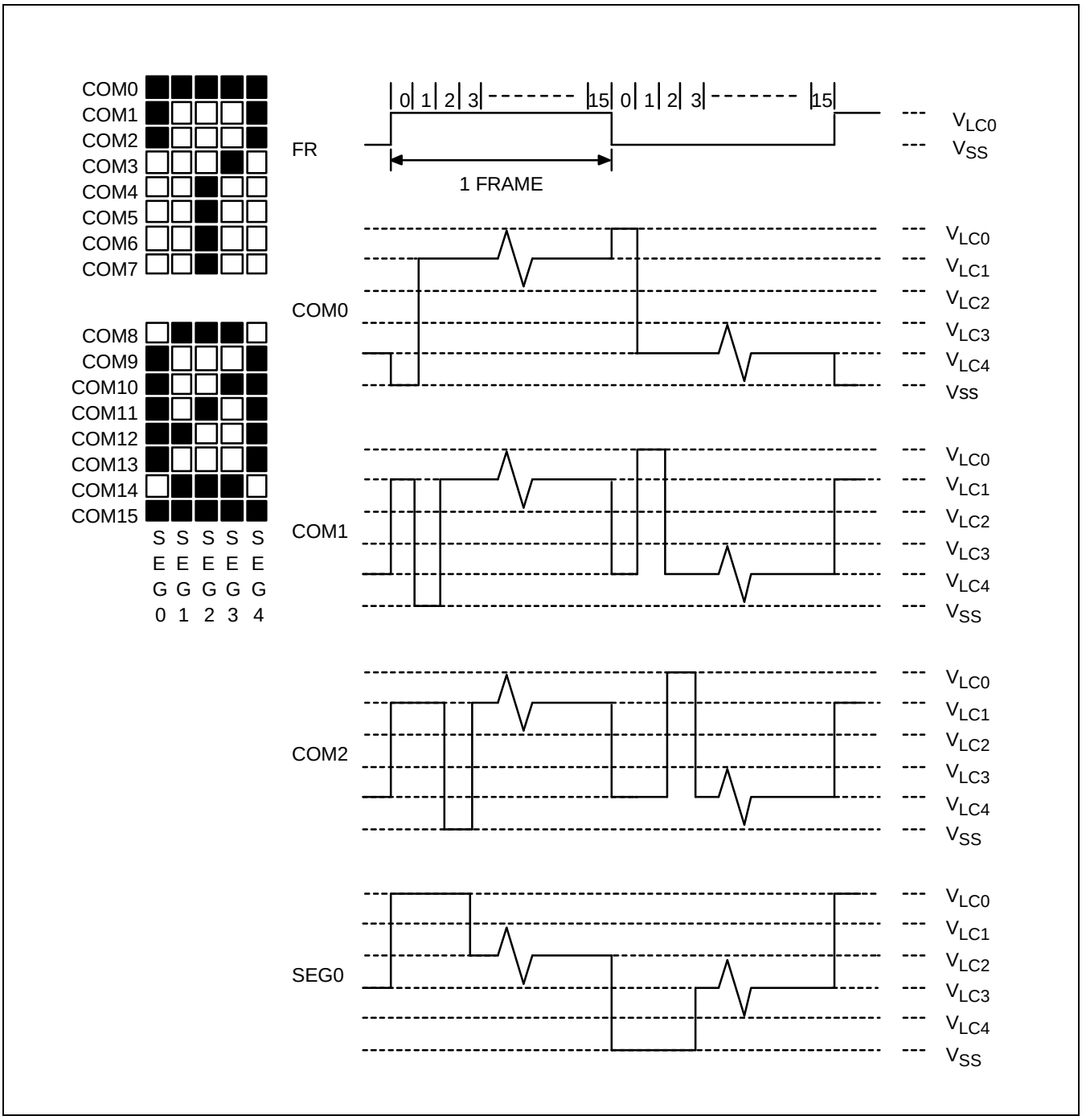


Figure 12-6. LCD Signal Waveforms (1/16 Duty, 1/5 Bias)

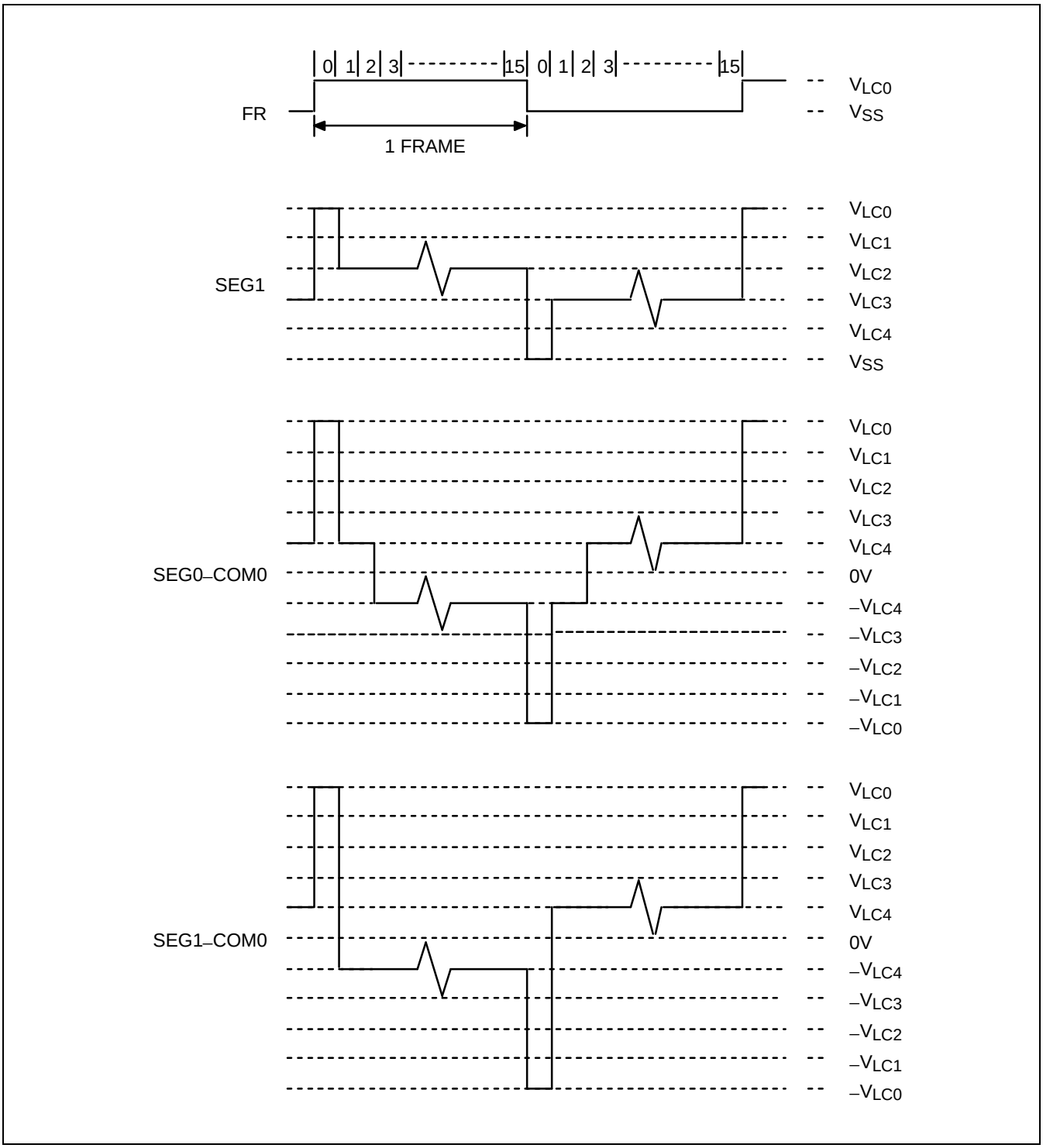


Figure 12-6. LCD Signal Waveforms (1/16 Duty, 1/5 Bias) (Continued)

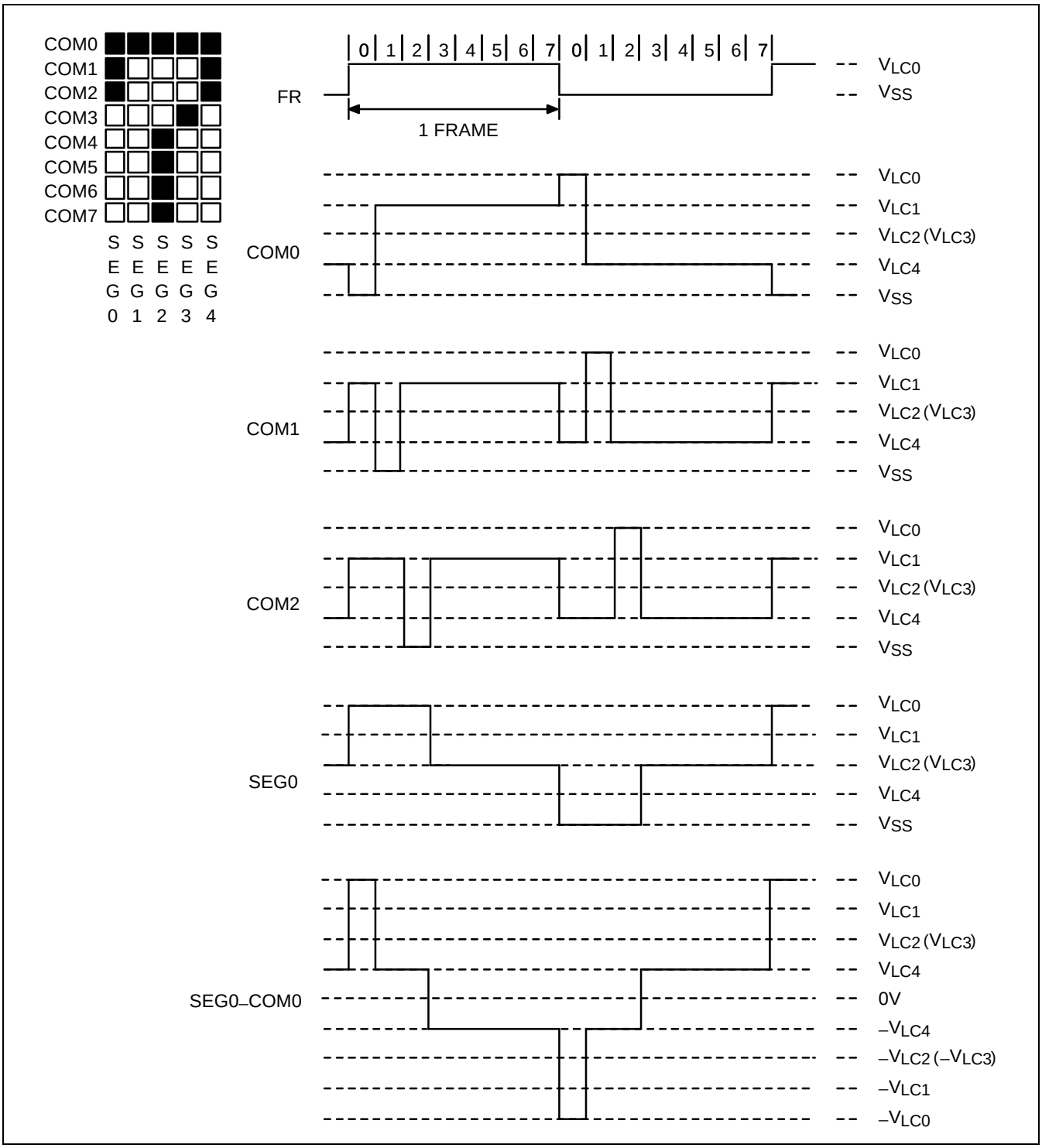


Figure 12-7. LCD Signal Waveforms (1/8 Duty, 1/4 Bias)

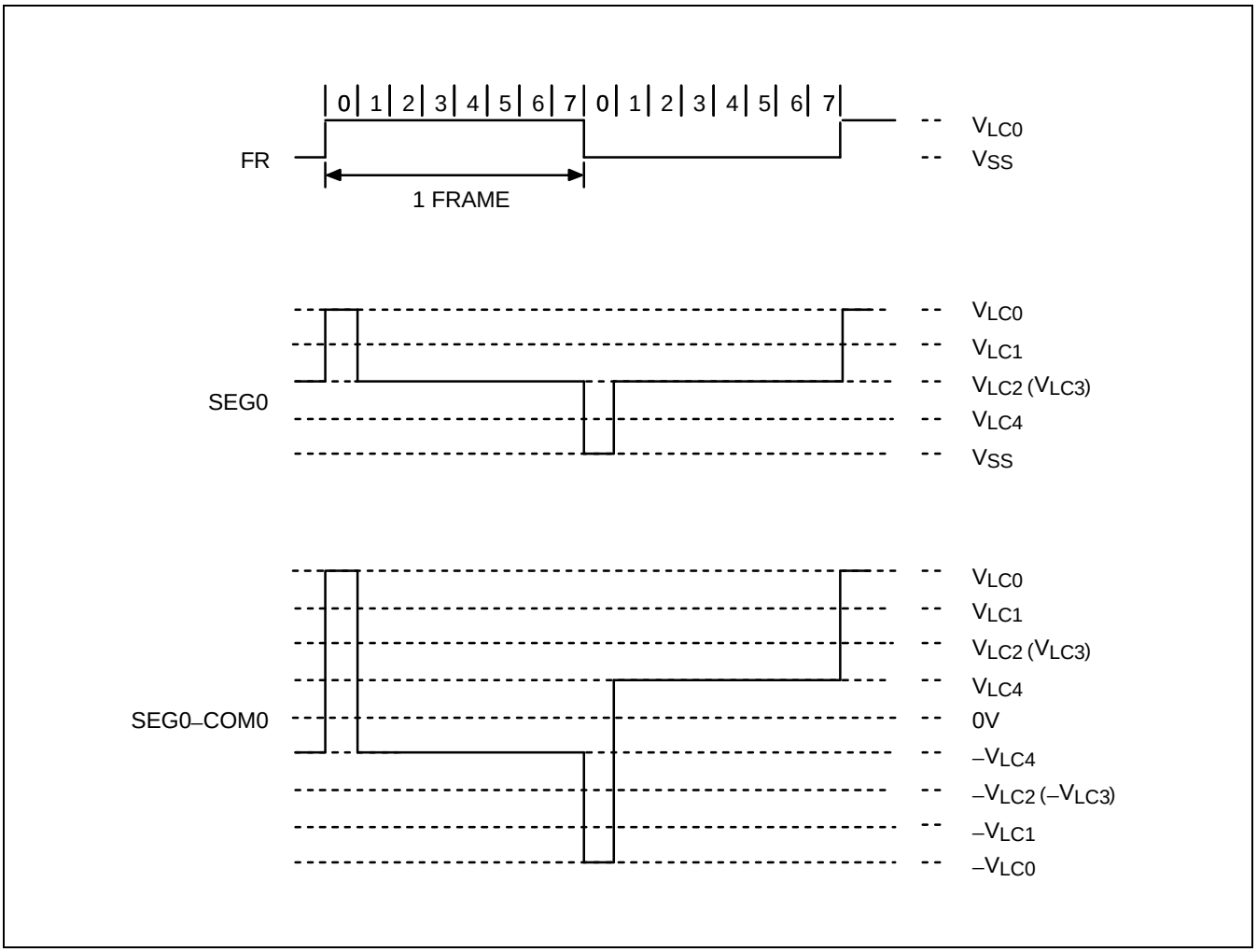


Figure 12-7. LCD Signal Waveforms (1/8 Duty, 1/4 Bias) (Continued)

13 ELECTRICAL DATA

OVERVIEW

In this section, information on S3C7295 electrical characteristics is presented as tables and graphics. The information is arranged in the following order:

Standard Electrical Characteristics

— Absolute maximum ratings
— D.C. electrical characteristics
— Main system clock oscillator characteristics
— Subsystem clock oscillator characteristics
— I/O capacitance
— A.C. electrical characteristics
— Operating voltage range

Miscellaneous Timing Waveforms

— A.C timing measurement point
— Clock timing measurement at X_{in}
— Clock timing measurement at XT_{in}
— TCL timing
— Input timing for RESET
— Input timing for external interrupts
— Serial data transfer timing

Stop Mode Characteristics and Timing Waveforms

— RAM data retention supply voltage in stop mode
— Stop mode release timing when initiated by RESET
— Stop mode release timing when initiated by an interrupt request

Table 13-1. Absolute Maximum Ratings

(T_A = 25 °C)

Parameter	Symbol	Conditions	Rating	Units
Supply Voltage	V_{DD}	–	– 0.3 to + 4.5	V
Input Voltage	V_I	Ports 0, 1	– 0.3 to VDD + 0.3	V
Output Voltage	V_O	–	– 0.3 to VDD + 0.3	V
Output Current High	I_{OH}	One I/O pin active	– 15	mA
		All I/O pins active	– 30	
Output Current Low	I_{OL}	One I/O pin active	+ 30 (Peak value)	mA
			+ 15(note)	
		Total for pins 0, 1	+ 100 (Peak value)	
			+ 60(note)	
Operating Temperature	T_A	–	– 40 to + 85	°C
Storage Temperature	T_{stg}	–	– 65 to + 150	°C

NOTE: The values for Output Current Low (I_{OL}) are calculated as Peak Value $\times \sqrt{\text{Duty}}$.

Table 13-2. D.C. Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 2.2 V to 3.4 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Input High Voltage	V_{IH1}	Ports 0, 1, and RESET	$0.8V_{DD}$	–	V_{DD}	V
	V_{IH2}	X_{in}, X_{out}, and XT_{in}	V_{DD} – 0.1		V_{DD}	
Input Low Voltage	V_{IL1}	Ports 0, 1, and RESET	–	–	$0.2V_{DD}$	V
	V_{IL2}	X_{in}, X_{out}, and XT_{in}			0.1	
Output High Voltage	V_{OH}	V_{DD} = 2.2 V to 3.4 V I_{OH} = – 1 mA Ports 0, 1	V_{DD} – 1.0	–	–	V
Output Low Voltage	V_{OL}	V_{DD} = 2.2 V to 3.4 V I_{OL} = 5 mA Ports 0, 1	–	–	1.0	V

Table 13-2. D.C. Electrical Characteristics (Continued)

(T_A = – 40 °C to + 85 °C, V_{DD} = 2.2 V to 3.4 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Input High Leakage Current	I_{LIH1}	$V_I = V_{DD}$ All input pins except those specified below for I_{LIH2}	–	–	3	µA
	I_{LIH2}	$V_I = V_{DD}$ X_{in}, X_{out} and XT_{in}			20	
Input Low Leakage Current	I_{LIL1}	V_I = 0 V All input pins except RESET X_{in}, X_{out} and XT_{in}	–	–	– 3	µA
	I_{LIL2}	V_I = 0 V RESET, X_{in}, X_{out} and XT_{in}			– 20	
Output High Leakage Current	I_{LOH}	$V_O = V_{DD}$ All output pins	–	–	3	µA
Output Low Leakage Current	I_{LOL}	V_O = 0 V All output pins	–	–	– 3	µA
Pull-Up Resistor	R_{L1}	V_I = 0 V; V_{DD} = 3V Ports 0, 1	50	100	200	kΩ
	R_{L2}	V_I = 0 V; V_{DD} = 3V; RESET	200	450	800	
LCD Voltage Dividing Resistor (1)	R_{LCD1}	Ta = + 25 °C	50	100	150	kΩ
	R_{LCD2}	Ta = + 25 °C	25	50	75	
$\lvert V_{DD}$-COMi$\rvert$ Voltage Drop (i = 0–15)	V_{DC}	V_{LCD} = 3.0 V – 15 µA per common pin	–	–	120	mV
$\lvert V_{LCD}$-SEGx$\rvert$ Voltage Drop (x = 0–43)	V_{DS}	V_{LCD} = 3.0 V – 15 µA per common pin	–	–	120	
Middle Output Voltage (2)	V_{LC0}	V_{LC0} = 5.0 V	V_{LC0}-0.2	V_{LC0}	V_{LC0}+0.2	V
	V_{LC1}		$0.8V_{LC0}$-0.2	$0.8V_{LC0}$	$0.8V_{LC0}$+0.2	
	V_{LC2}		$0.6V_{LC0}$-0.2	$0.6V_{LC0}$	$0.6V_{LC0}$+0.2	
	V_{LC3}		$0.4V_{LC0}$-0.2	$0.4V_{LC0}$	$0.4V_{LC0}$+0.2	
	V_{LC4}		$0.2V_{LC0}$-0.2	$0.2V_{LC0}$	$0.2V_{LC0}$+0.2	

NOTES:

1. RLCD1 is LCD voltage dividing resistor when LCON.2 = "0", and RLCD2 when LCON.2 = "1".
2. It is middle output voltage when 1/16 duty and 1/5 bias.

Table 13-2. D.C. Electrical Characteristics (Concluded)

(T_A = – 40 °C to + 85 °C, V_{DD} = 2.2 V to 3.4 V)

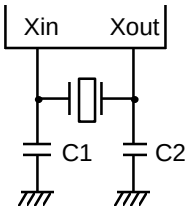
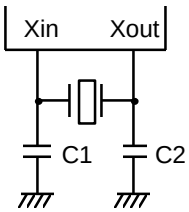
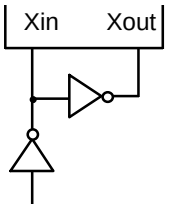
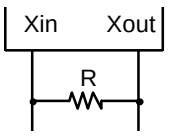
Parameter	Symbol	Conditions		Min	Typ	Max	Units
Supply Current (1)	I_{DD1}	V_{DD} = 3V ± 10% 4.19 MHz (PCON=3H) crystal oscillator C1 = C2 = 22 pF		–	1.3	3.0	mA
	I_{DD2}	Idle mode; V_{DD} = 3 V ± 10% 4.19 MHz (PCON=3H) crystal oscillator C1 = C2 = 22 pF			0.4	1.0	
	I_{DD3} (2)	V_{DD} = 3 V ± 10% 32 kHz crystal oscillator		–	15	30	µA
	I_{DD4} (2)	Idle mode; V_{DD} = 3 V ± 10% 32 kHz crystal oscillator			5	15	
	I_{DD5}	Stop mode; V_{DD} = 3 V ± 10%	SCMOD=0000B, XTin=0V		0.5	3	
		Stop mode; V_{DD} = 3 V ± 10%	SCMOD=0100B		0.2	2	

NOTES:

1. Current in the following circuits are not included; on-chip pull-up resistors, internal LCD voltage dividing resistors, voltage doubler, and output port drive currents.
2. Data includes power consumption for subsystem clock oscillation.
3. When the system clock control register, SCMOD, is set to 1001B, main system clock oscillation stops and the subsystem clock is used.

Table 13-3. Main System Clock Oscillator Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 2.2 V to 3.4 V)

Oscillator	Clock Configuration	Parameter	Test Condition	Min	Typ	Max	Units
Ceramic Oscillator		Oscillation frequency (1)	–	0.4	–	4.19	MHz
		Stabilization time (2)	Stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range; V_{DD} = 3.0 V	–	–	4	ms
Crystal Oscillator		Oscillation frequency (1)	–	0.4	–	4.19	MHz
		Stabilization time (2)	V_{DD} = 3.0 V	–	–	10	ms
External Clock		X_{in} input frequency (1)	–	0.4	–	4.19	MHz
		X_{in} input high and low level width (t_{XH}, t_{XL})	–	83.3	–	1250	ns
RC Oscillator		Frequency	V_{DD} = 3 V	0.4	–	1.5	MHz

NOTES:

1. Oscillation frequency and X_{in} input frequency data are for oscillator characteristics only.
2. Stabilization time is the interval required for oscillator stabilization after a power-on occurs, or when stop mode is terminated.

Table 13-4. Recommended Oscillator Constants

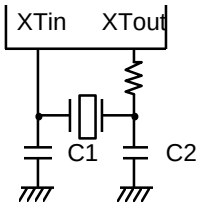
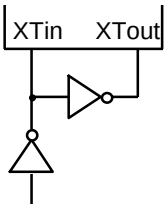
(T_A = – 40 °C to + 85 °C, V_{DD} = 2.2 V to 3.4 V)

Manufacturer	Series Number (1)	Frequency Range	Load Cap (pF)		Oscillator Voltage Range (V)		Remarks
			C1	C2	MIN	MAX	
TDK	FCR ðÿM5	3.58 MHz–4.2 MHz	33	33	2.2	3.4	Leaded Type
	FCR ðÿMC5	3.58 MHz–4.2 MHz	(2)	(2)	2.2	3.4	On-chip C Leaded Type
	CCR ðÿMC3	3.58 MHz–4.2 MHz	(3)	(3)	2.2	3.4	On-chip C SMD Type

NOTES:
1. Please specify normal oscillator frequency.
2. On-chip C: 30pF built in.
3. On-chip C: 38pF built in.

Table 13-5. Subsystem Clock Oscillator Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 2.2 V to 3.4 V)

Oscillator	Clock Configuration	Parameter	Test Condition	Min	Typ	Max	Units
Crystal Oscillator		Oscillation frequency (1)	–	32	32.768	35	kHz
		Stabilization time (2)	V_{DD} = 2.2 V to 3.4 V	–	1.0	3	s
External Clock		XT_{in} input frequency (1)	–	32	–	100	kHz
		XT_{in} input high and low level width (t_{XTL}, t_{XTH})	–	5	–	15	µs

NOTES:
1. Oscillation frequency and XT_{in} input frequency data are for oscillator characteristics only.
2. Stabilization time is the interval required for oscillating stabilization after a power-on occurs.

Table 13-6. Input/Output Capacitance

(T_A = 25 °C, V_{DD} = 0 V)

Parameter	Symbol	Condition	Min	Typ	Max	Units
Input Capacitance	C_{IN}	f = 1 MHz; Unmeasured pins are returned to V_{SS}	–	–	15	pF
Output Capacitance	C_{OUT}		–	–	15	pF
I/O Capacitance	C_{IO}		–	–	15	pF

Table 13-7. Voltage Doubler Output

(T_A = -40 °C to + 85 °C, V_{DD} = 2.2 V to 3.4 V)

Parameter	Symbol	Condition	Min	Typ	Max	Units
Voltage Doubler Output	Vbias	V_{DD} = 2.2 V to 3.4 V	–	2 V_{DD}	–	V

Table 13-8. A.C. Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 2.2 V to 3.4 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (note)	t_{CY}	V_{DD} = 2.2 V to 3.4 V	0.95	–	64	µs
		With subsystem clock (fxt)	114	122	125	
Interrupt Input High, Low Width	f_{INTH}, f_{INTL}	INT0–INT2, INT4 K0–K3	10	–	–	
RESET Input Low Width	t_{RSL}	Input	10	–	–	

NOTE: Unless otherwise specified, Instruction Cycle Time condition values assume a main system clock (fx) source.

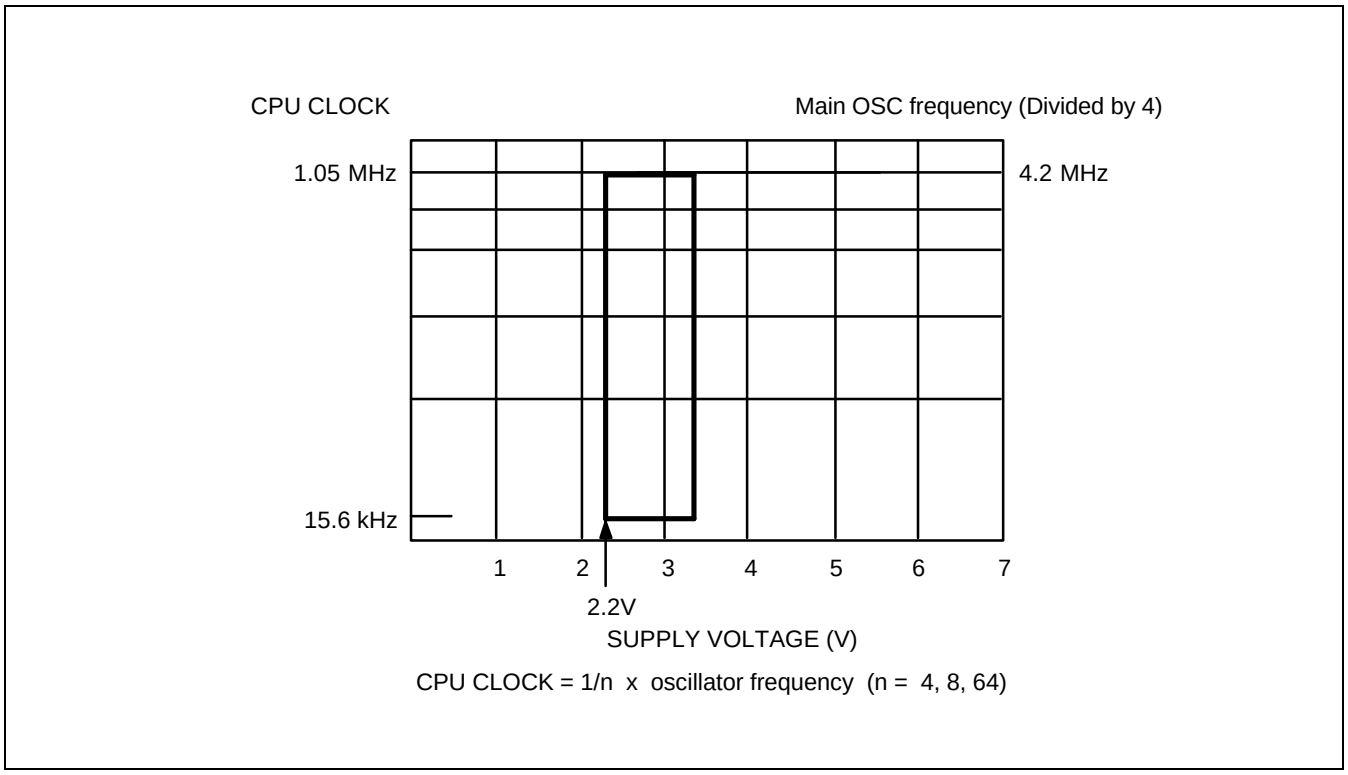


Figure 13-1. Standard Operating Voltage Range

Table 13-9. RAM Data Retention Supply Voltage in Stop Mode

(T_A = – 40 °C to + 85 °C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data retention supply voltage	V_{DDDR}	–	2.2	–	3.4	V
Data retention supply current	I_{DDDR}	V_{DDDR} = 2.2 V	–	0.1	10	µA
Release signal set time	t_{SREL}	–	0	–	–	µs
Oscillator stabilization wait time (1)	t_{WAIT}	Released by RESET	–	2^{17} / fx	–	ms
		Released by interrupt	–	(2)	–	

NOTES:
1. During oscillator stabilization wait time, all CPU operations must be stopped to avoid instability during oscillator start-up.
2. Use the basic timer mode register (BMOD) interval timer to delay execution of CPU instructions during the wait time.

TIMING WAVEFORMS

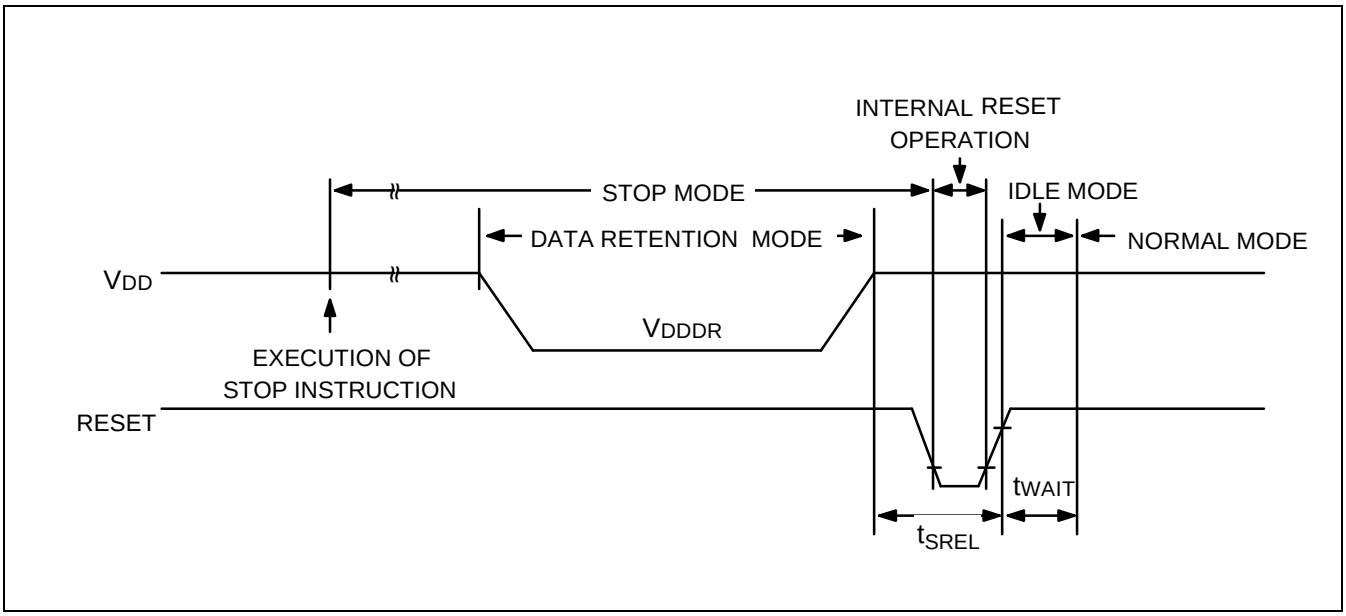


Figure 13-2. Stop Mode Release Timing When Initiated by RESET

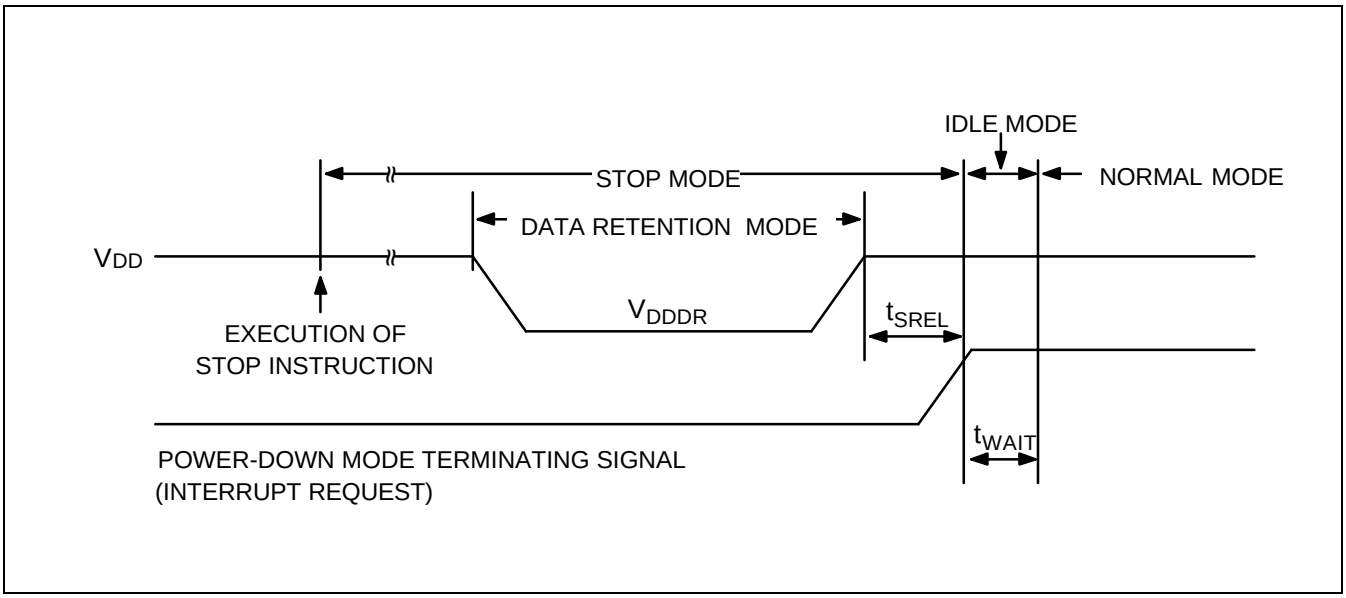


Figure 13-3. Stop Mode Release Timing When Initiated by Interrupt Request

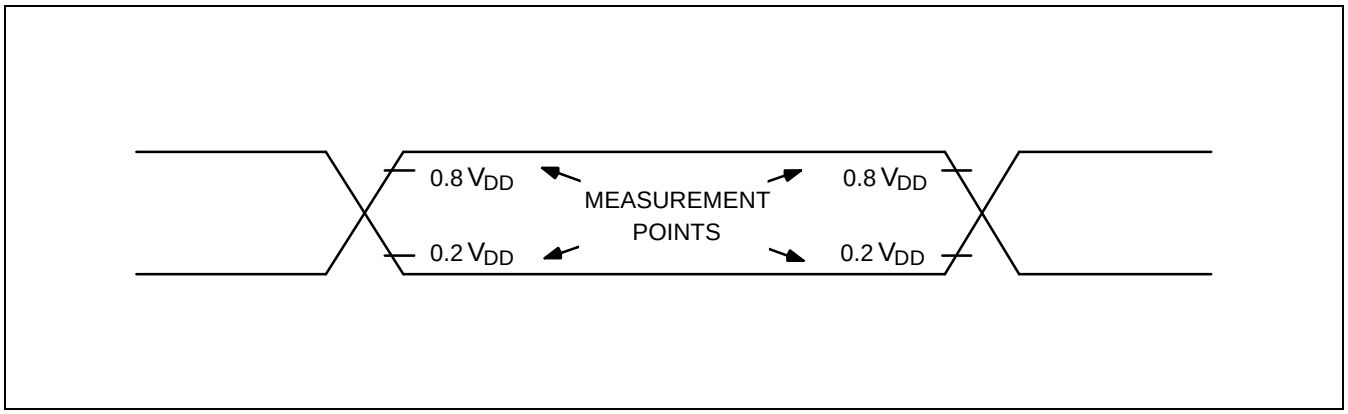


Figure 13-4. A.C. Timing Measurement Points (Except for X_{in} and XT_{in})

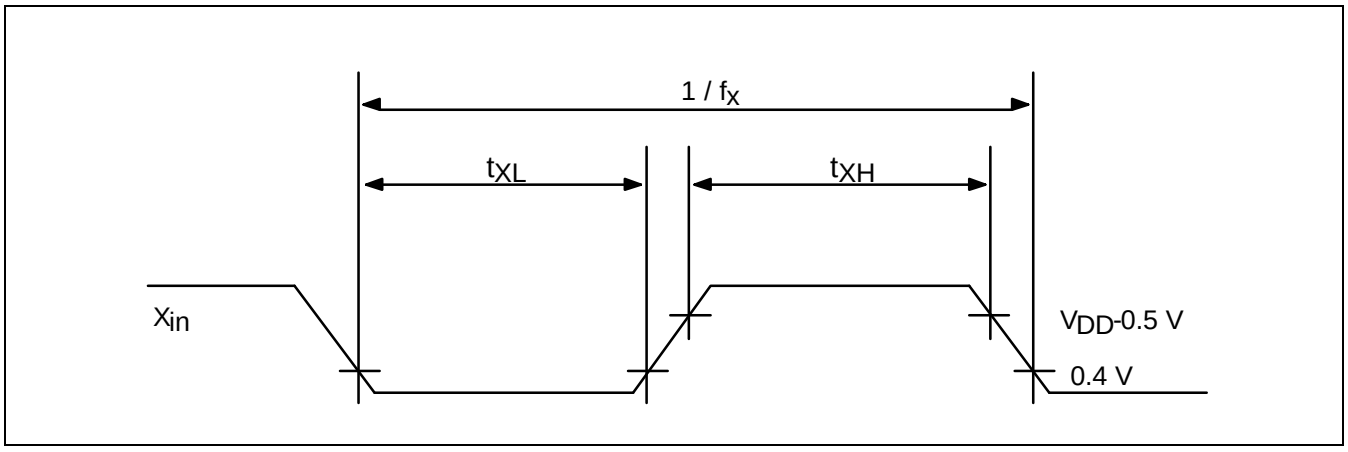


Figure 13-5. Clock Timing Measurement at X_{in}

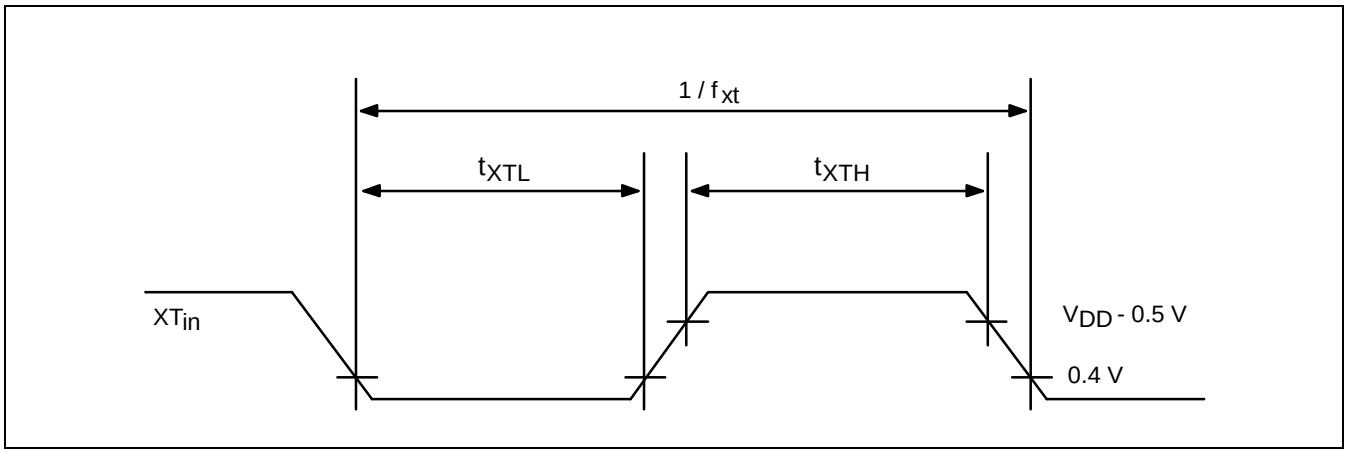


Figure 13-6. Clock Timing Measurement at XT_{in}

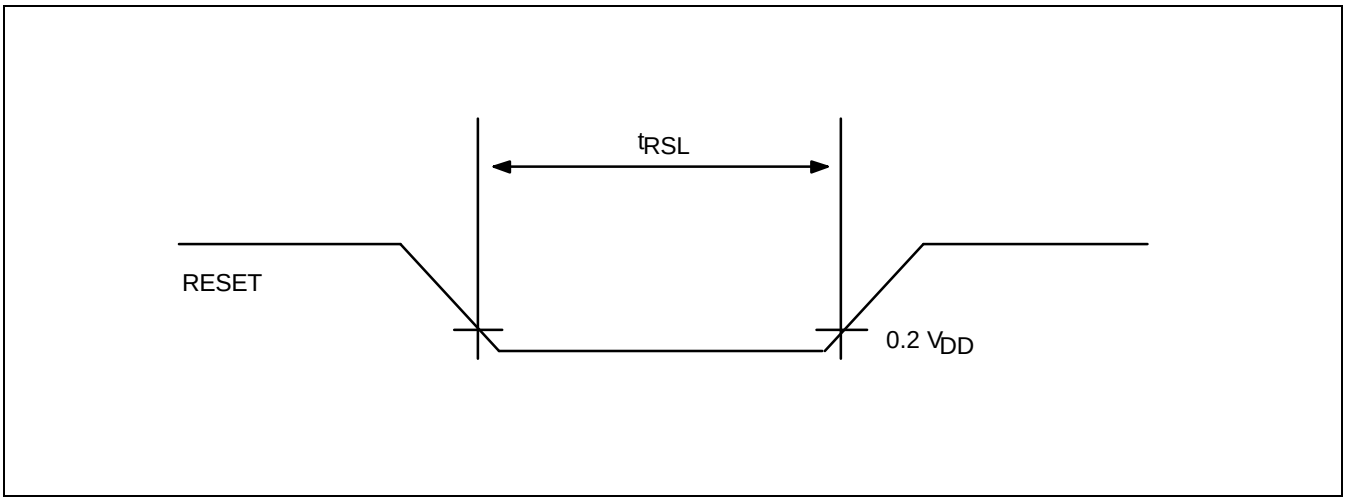


Figure 13-7. Input Timing for RESET Signal

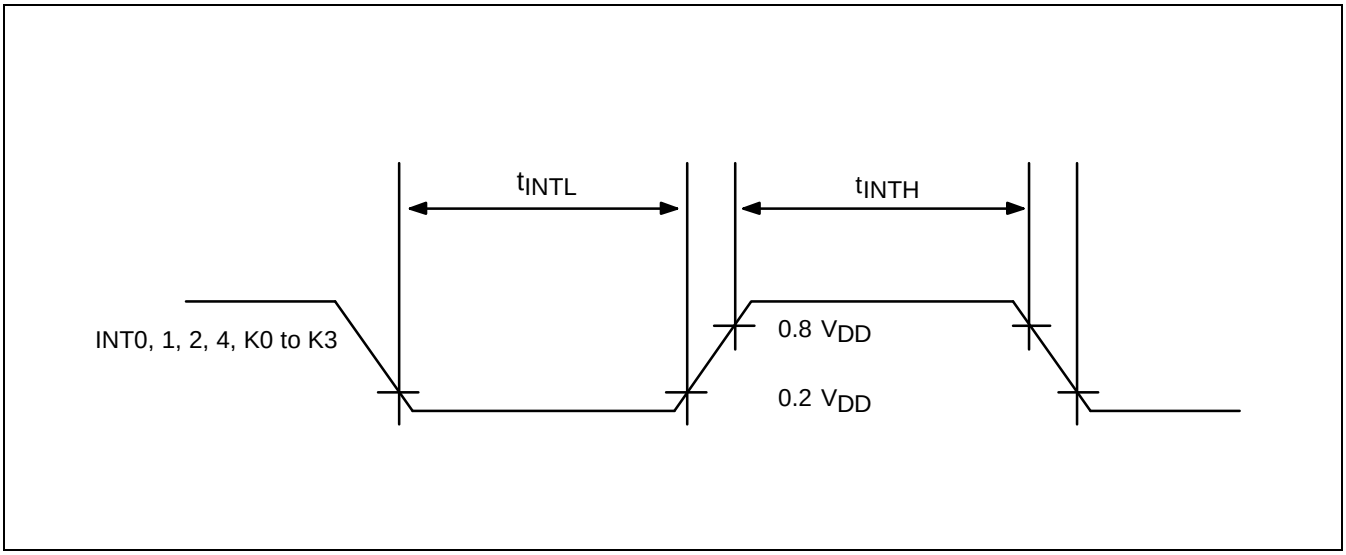


Figure 13-8. Input Timing for External Interrupts

NOTES

CHARACTERISTIC CURVES

NOTE

The characteristic values shown in the following graphs are based on actual test measurements.

They do not, however, represent guaranteed operating values.

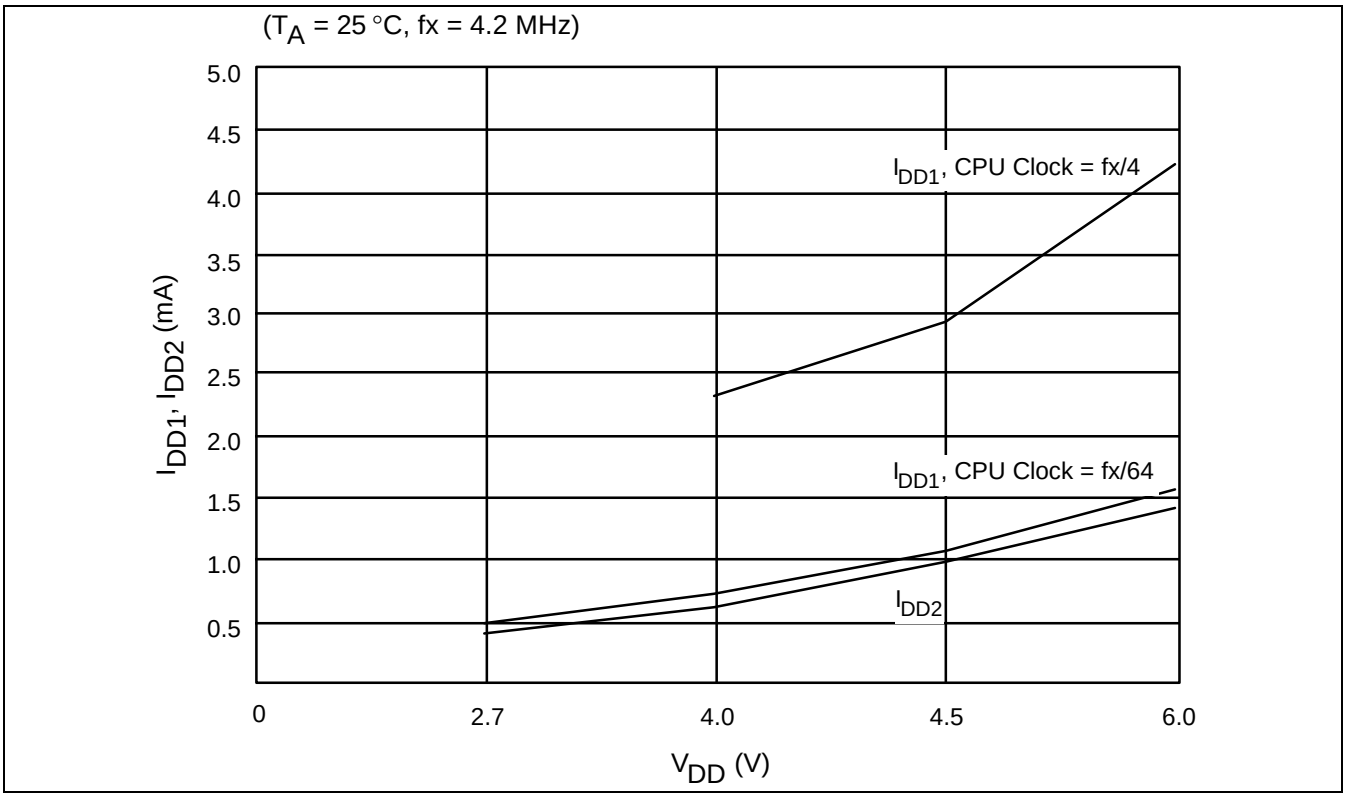


Figure 13-11. I_{DD1}, I_{DD2} VS. V_{DD}

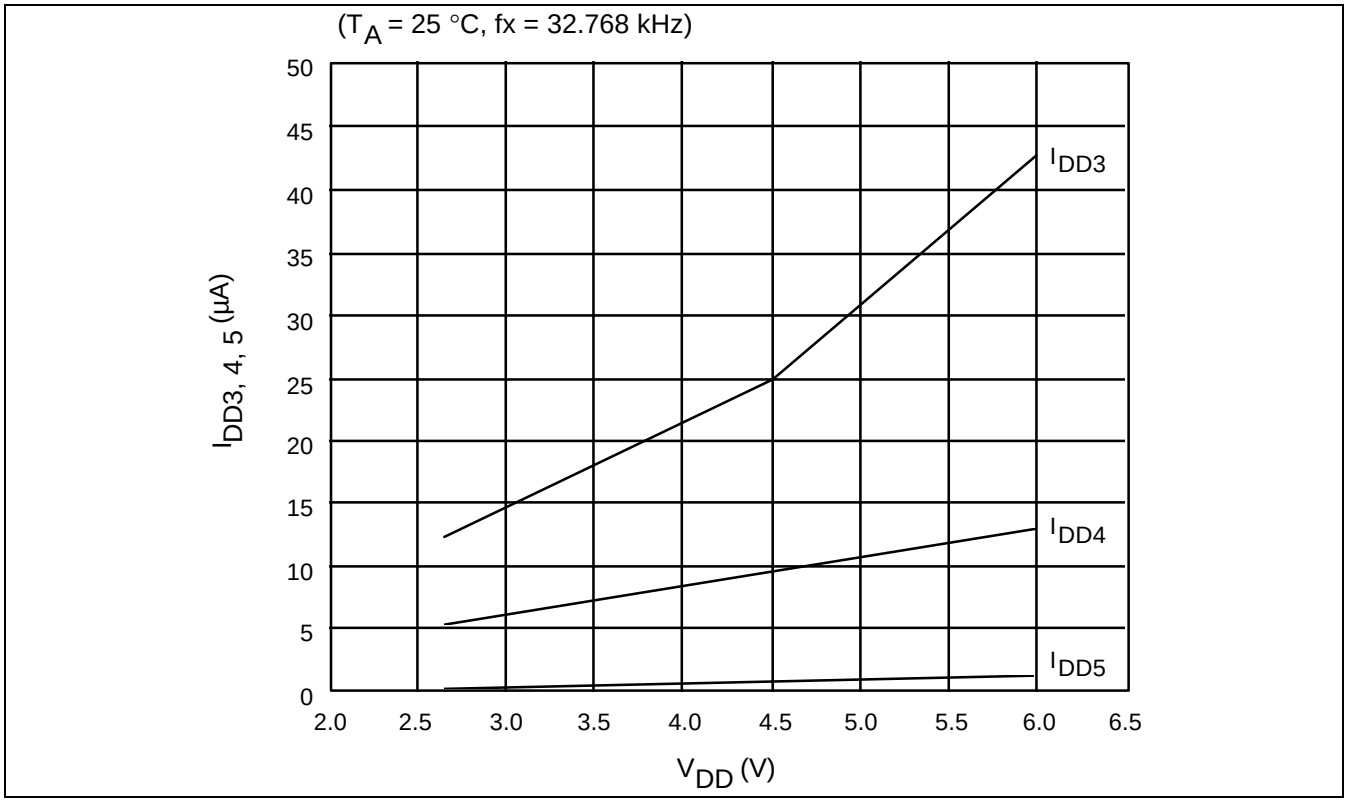


Figure 13-12. I_{DD3}, I_{DD4}, I_{DD5} VS. V_{DD}

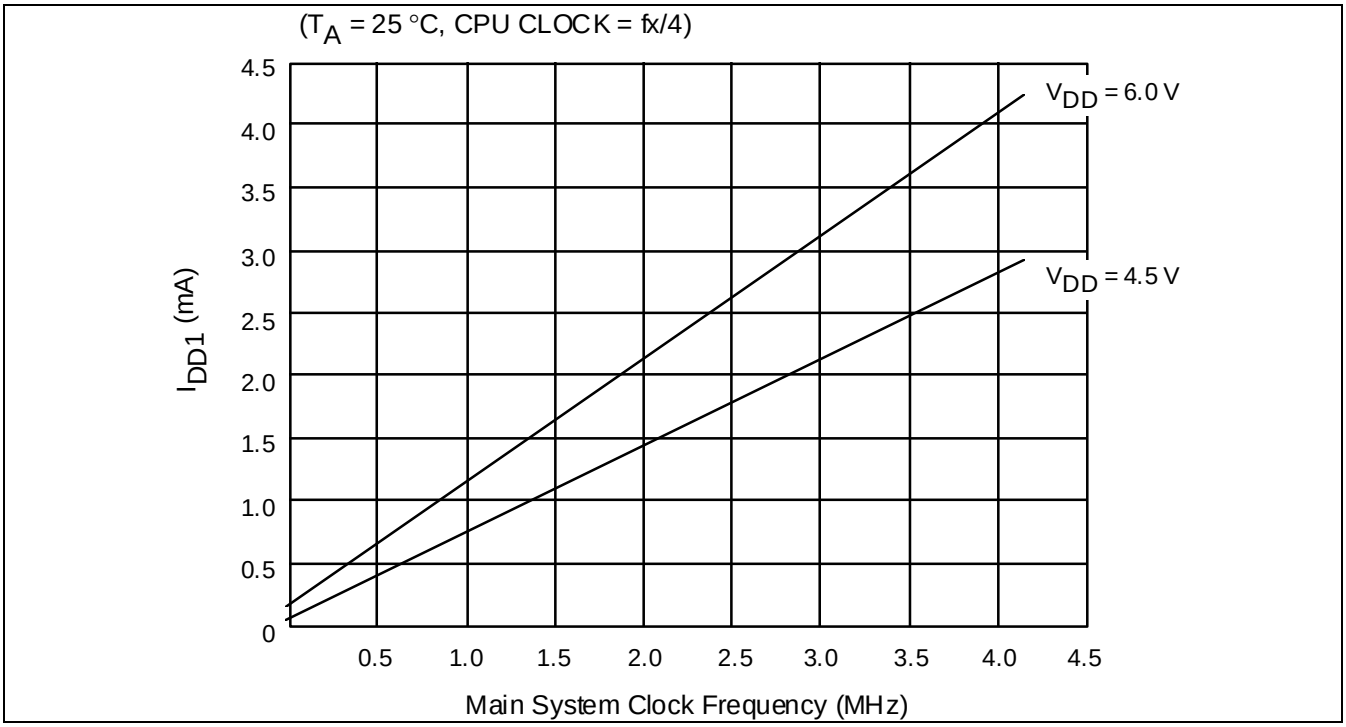


Figure 13-13. I_{DD1} VS. Main System Clock Frequency

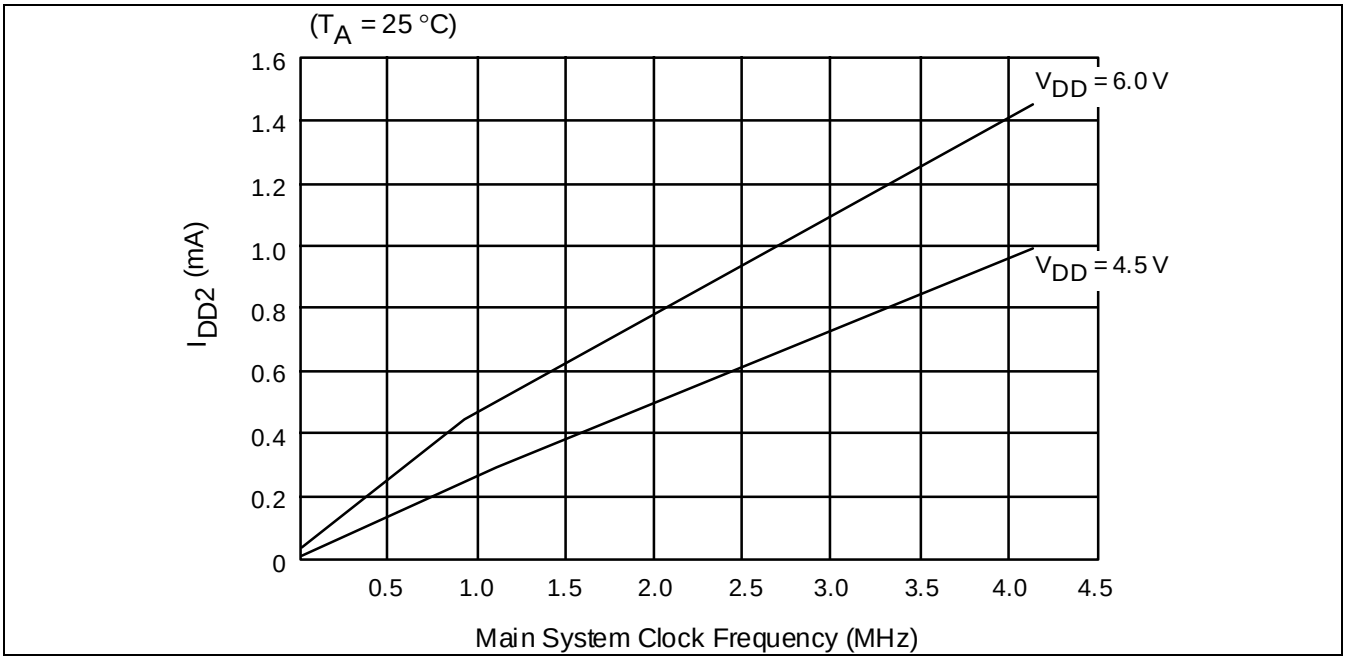


Figure 13-13. I_{DD2} VS. Main System Clock Frequency

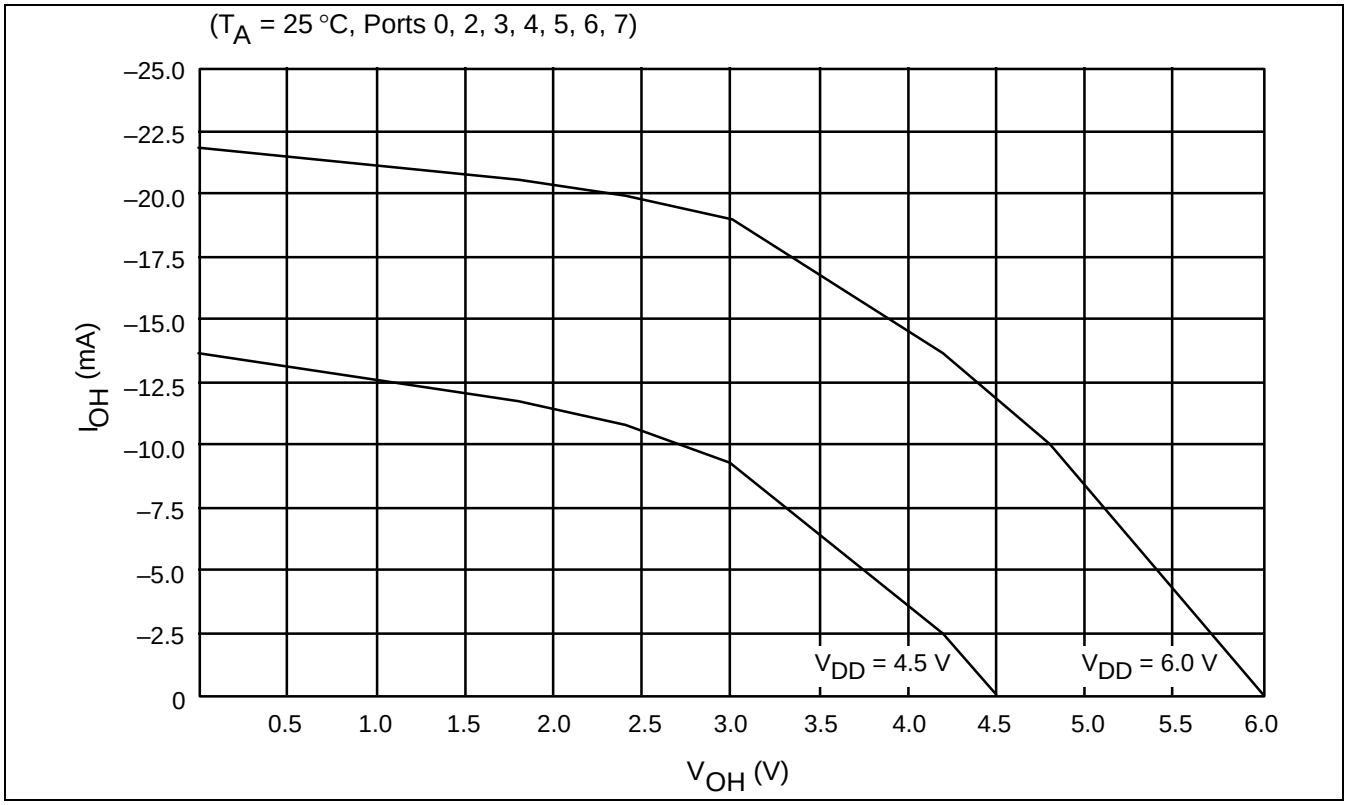


Figure 13–15. I_{OH} VS. V_{OH} (P0, 2, 3, 4, 5, 6, 7)

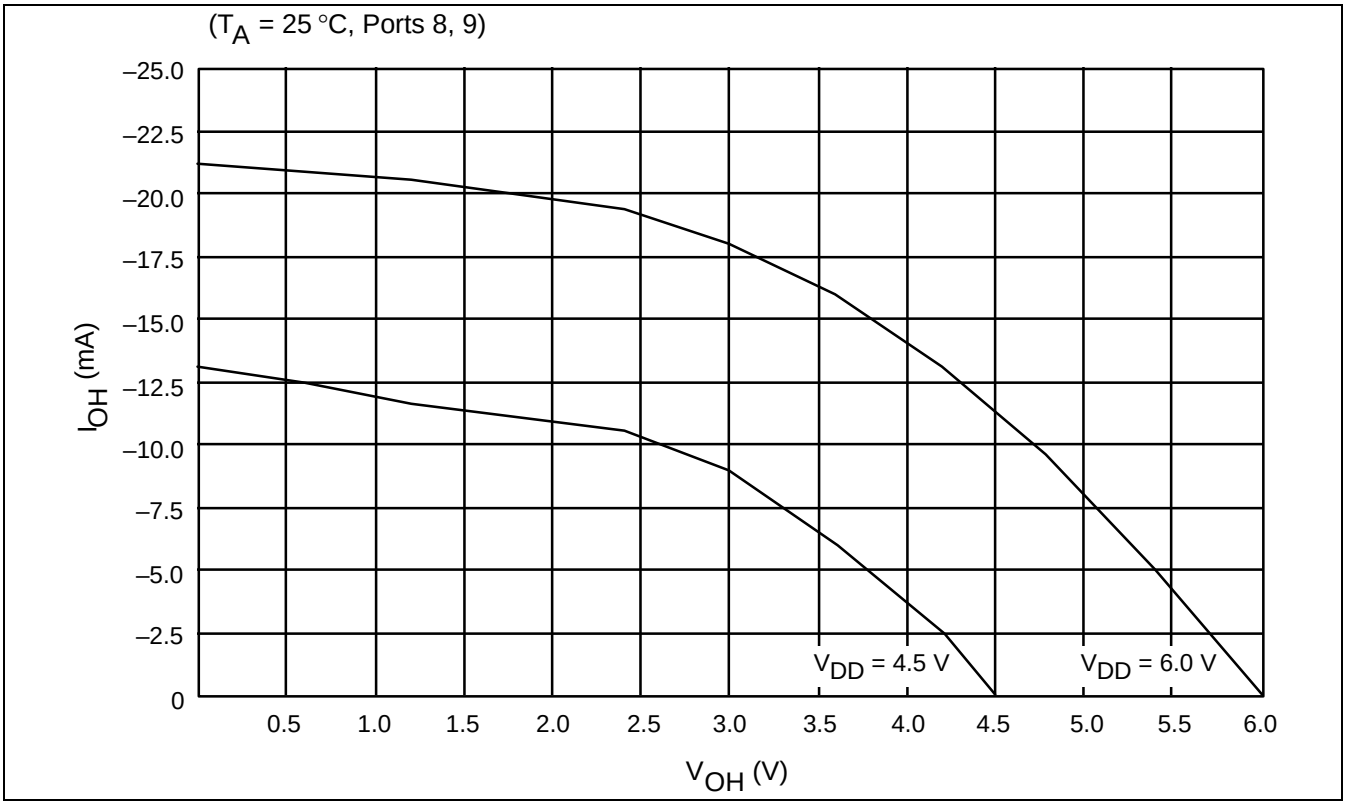


Figure 13–16. I_{OH} VS. V_{OH} (P8, 9)

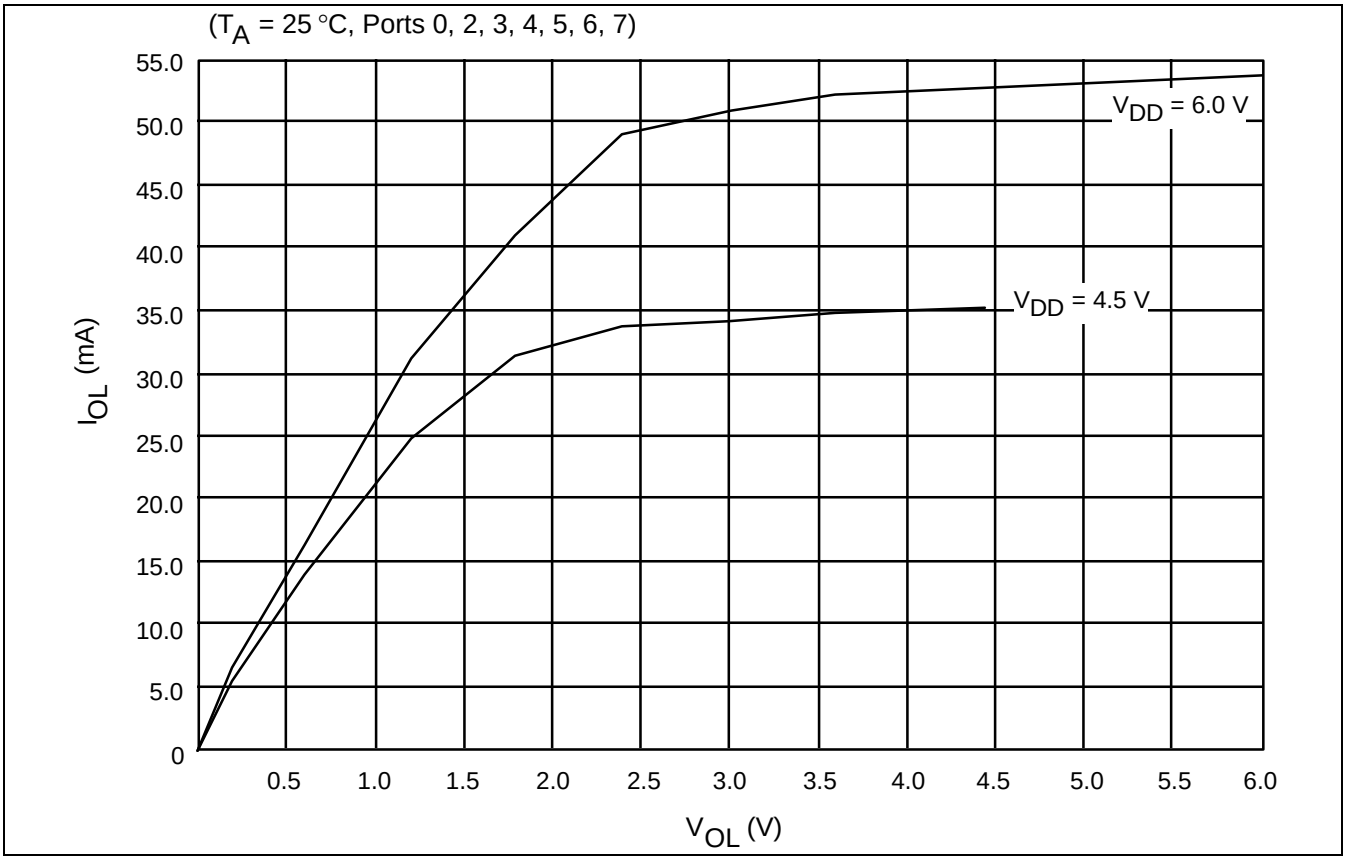


Figure 13–17. I_{OL} VS. V_{OL} (P0, 2, 3, 4, 5, 6, 7)

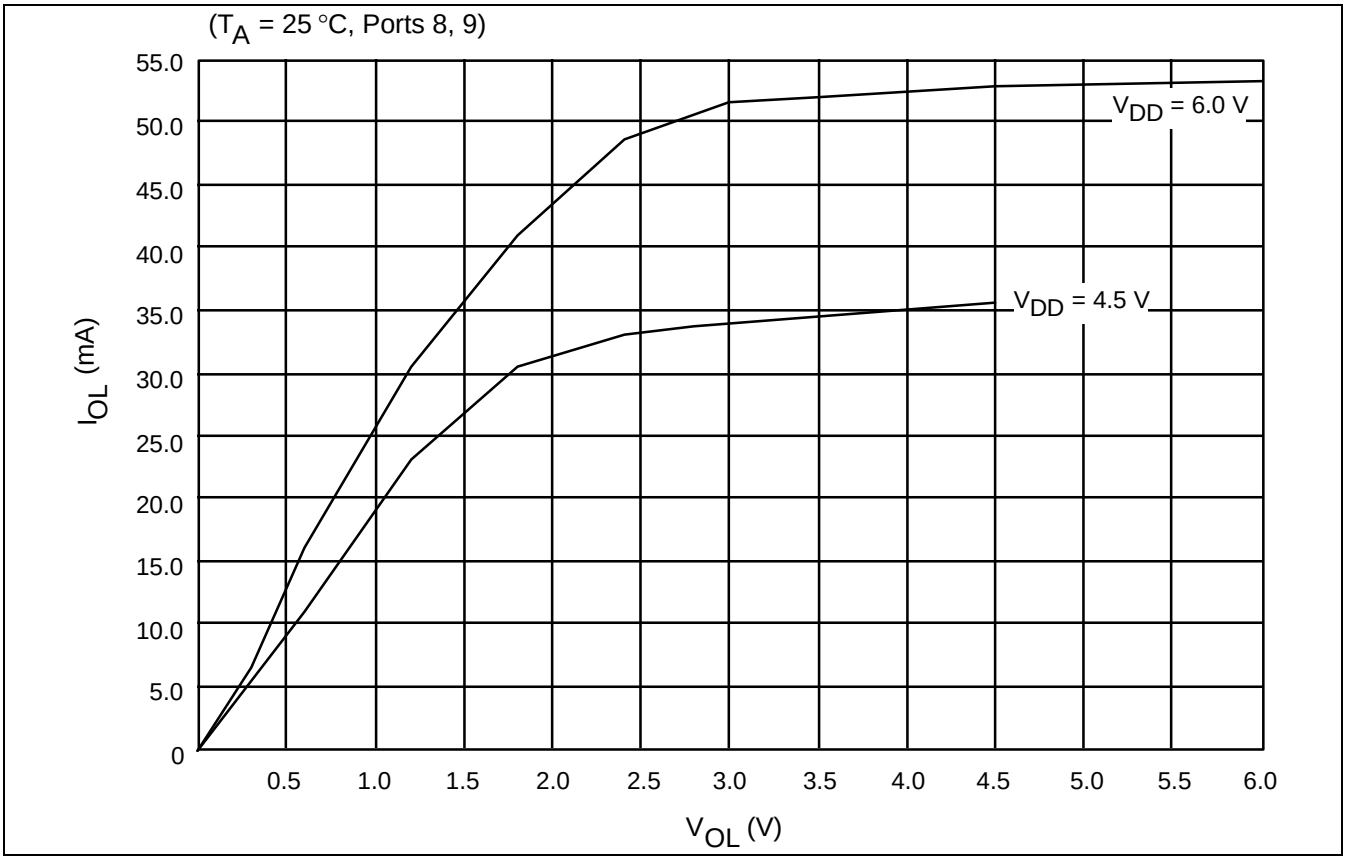
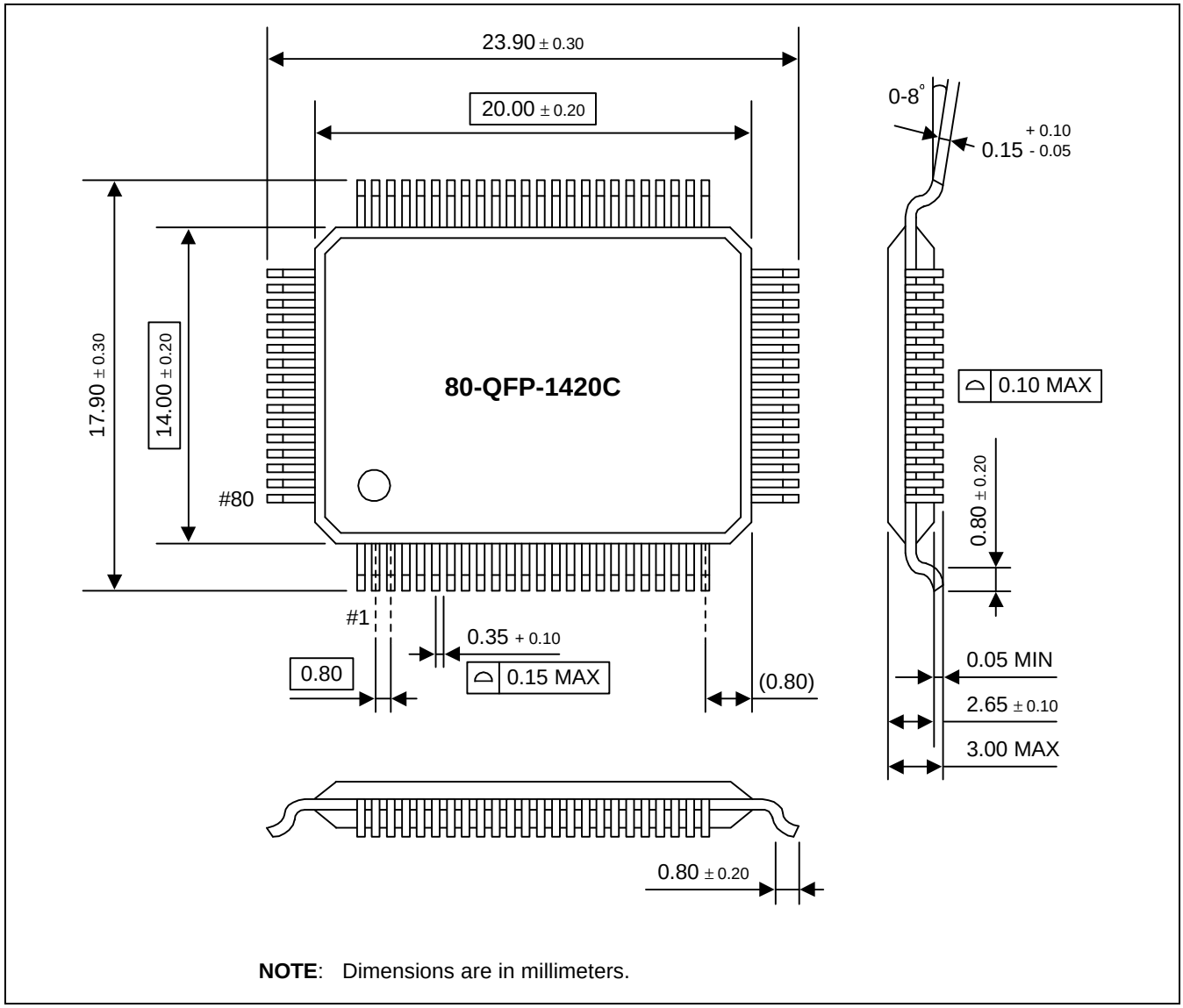


Figure 13–18. I_{OL} VS. V_{OL} (P8, 9)

14 MECHANICAL DATA

OVERVIEW

The S3C7295/P7295 is available in a 80-QFP-1420 package.



NOTE: Dimensions are in millimeters.

Figure 14-1. 80-QFP-1420C Package Dimensions

15 S3P7295 OTP

OVERVIEW

The S3P7295 single-chip CMOS microcontroller is the OTP (One Time Programmable) version of the S3C7295 microcontroller. It has an on-chip OTP ROM instead of masked ROM. The EPROM is accessed by serial data format.

The S3P7295 is fully compatible with the S3C7295, both in function and in pin configuration. Because of its simple programming requirements, the S3P7295 is ideal for use as an evaluation chip for the S3C7295.

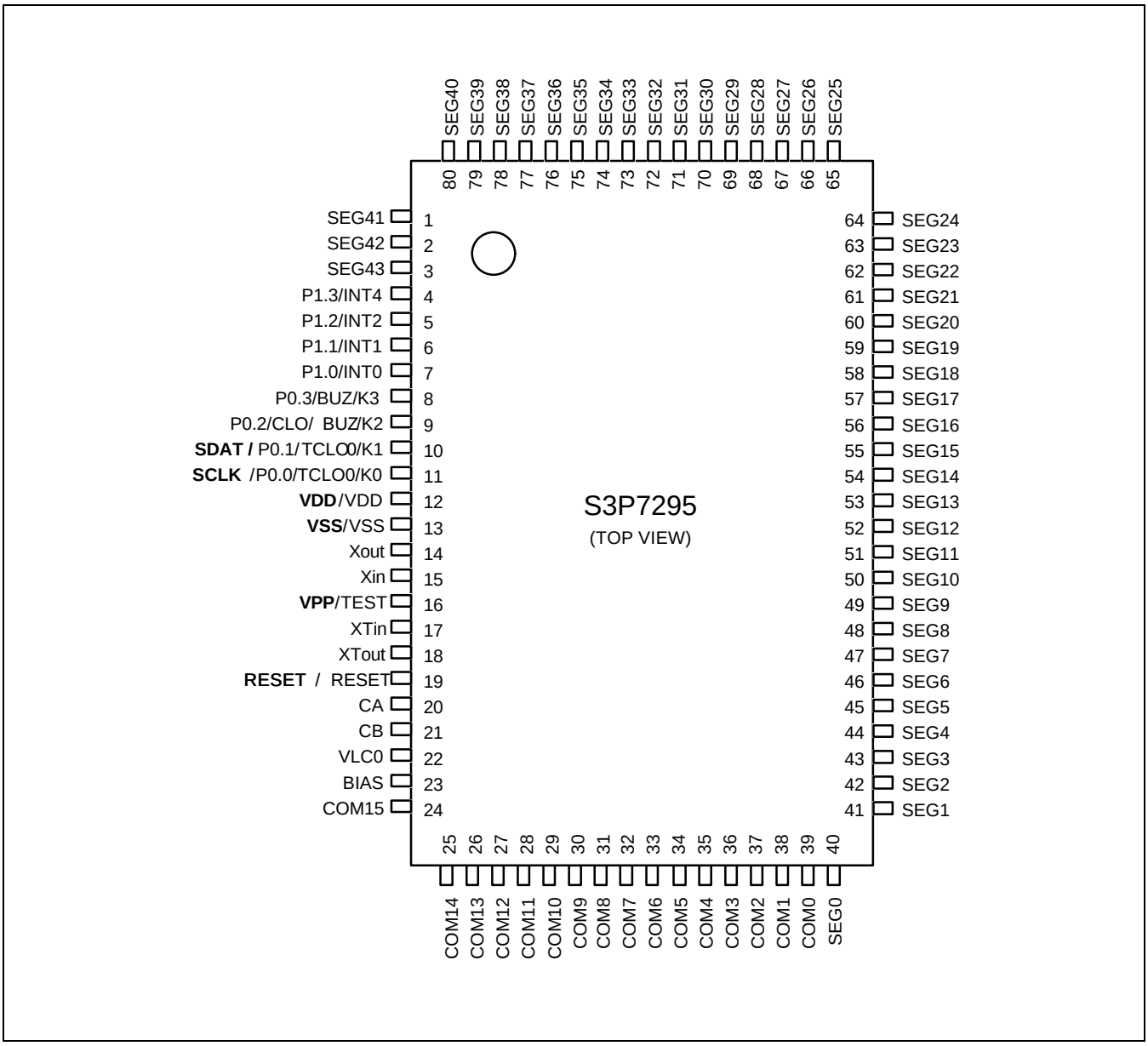


Figure 15-1. S3P7295 Pin Assignments (80-QFP Package)

Table 15-1. Descriptions of Pins Used to Read/Write the EPROM

Main Chip	During Programming			
Pin Name	Pin Name	Pin No.	I/O	Function
P0.1	SDAT	10	I/O	Serial data pin. Output port when reading and input port when writing. Can be assigned as a Input/push-pull output port.
P0.0	SCLK	11	I/O	Serial clock pin. Input only pin.
TEST	V_{PP}(TEST)	16	I	Power supply pin for EPROM cell writing (indicates that OTP enters into the writing mode). When 12.5 V is applied, OTP is in writing mode and when 5 V is applied, OTP is in reading mode. (Option)
RESET	RESET	19	I	Chip initialization
V_{DD}/V_{SS}	V_{DD}/V_{SS}	12/13	I	Logic power supply pin. VDD should be tied to +5 V during programming.

Table 15-2. Comparison of S3P7295 and S3C7295 Features

Characteristic	S3P7295	S3C7295
Program Memory	16 Kbyte EPROM	16 Kbyte mask ROM
Operating Voltage (V_{DD})	2.2 V to 3.4 V	2.2 V to 3.4 V
OTP Programming Mode	V_{DD} = 5 V, V_{PP}(TEST)=12.5V	
Pin Configuration	80 QFP	80 QFP
EPROM Programmability	User Program 1 time	Programmed at the factory

OPERATING MODE CHARACTERISTICS

When 12.5 V is supplied to the V_{PP}(TEST) pin of the S3P7295, the EPROM programming mode is entered. The operating mode (read, write, or read protection) is selected according to the input signals to the pins listed in Table 15–3 below.

Table 15-3. Operating Mode Selection Criteria

V_{DD}	V_{PP} (TEST)	REG/MEM	Address (A15–A0)	R/W	Mode
5 V	5 V	0	0000H	1	EPROM read
	12.5 V	0	0000H	0	EPROM program
	12.5 V	0	0000H	1	EPROM verify
	12.5 V	1	0E3FH	0	EPROM read protection

NOTE: "0" means Low level; "1" means High level.

Table 15-4. D.C. Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 2.2 V to 3.4 V)

Parameter	Symbol	Conditions		Min	Typ	Max	Units
Supply Current (1)	IDD1	V_{DD} = 3V ± 10% 4.19 MHz (PCON=3H) crystal oscillator C1 = C2 = 22 pF		–	1.3	3.0	mA
	IDD2	Idle mode; V_{DD} = 3 V ± 10% 4.19 MHz (PCON=3H) crystal oscillator C1 = C2 = 22 pF			0.4	1.0	
	IDD3 (2)	V_{DD} = 3 V ± 10% 32 kHz crystal oscillator		–	15	30	µA
	IDD4 (2)	Idle mode; V_{DD} = 3 V ± 10% 32 kHz crystal oscillator			5	15	
	IDD5	Stop mode; V_{DD} = 3 V ± 10%	SCMOD=0000B, XTin=0V		0.5	3	
		Stop mode; V_{DD} = 3 V ± 10%	SCMOD=0100B		0.2	2	

NOTES:

1. Data includes power consumption for subsystem clock oscillation.
2. When the system clock control register, SCMOD, is set to 1001B, main system clock oscillation stops and the subsystem clock is used.
3. Current in the following circuits are not included; on-chip pull-up resistors, internal LCD voltage dividing resistors, voltage doubler, and output port drive currents.

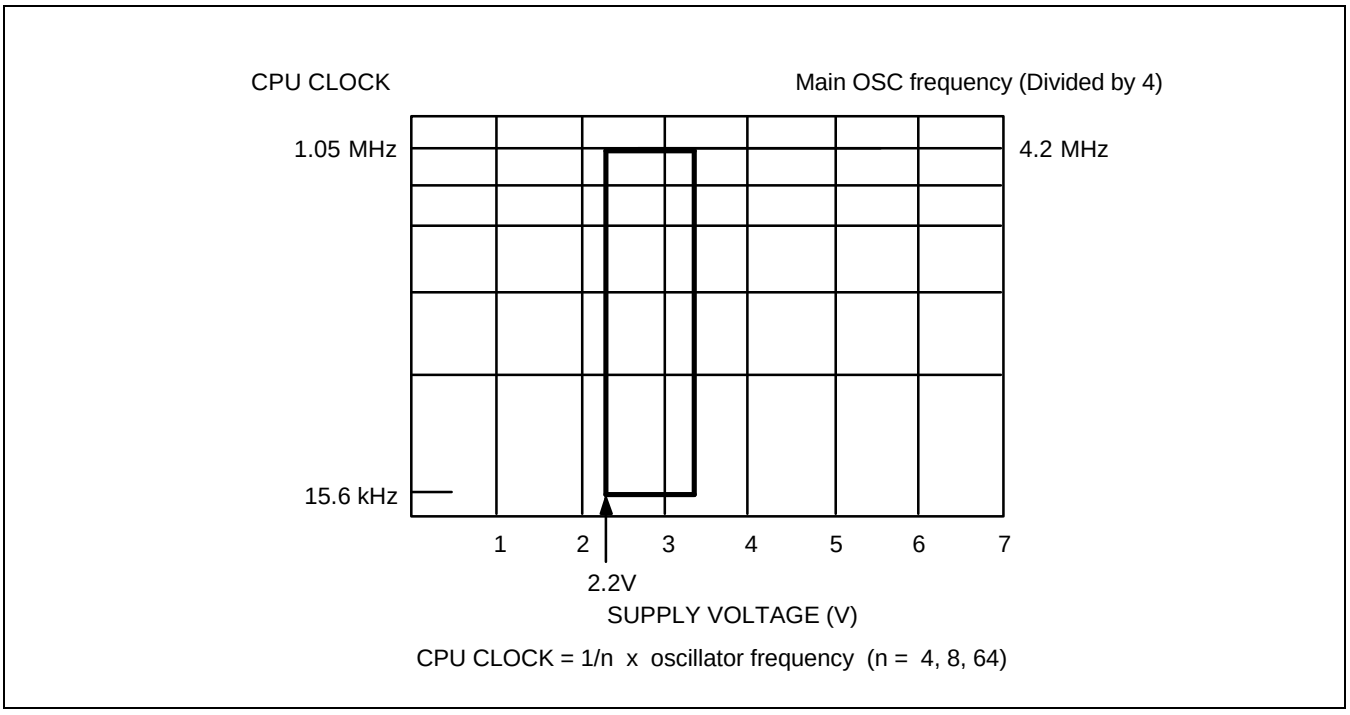


Figure 15-2. Standard Operating Voltage Range

16 DEVELOPMENT TOOLS

OVERVIEW

Samsung provides a powerful and easy-to-use development support system in turnkey form. The development support system is configured with a host system, debugging tools, and support software. For the host system, any standard computer that operates with MS-DOS as its operating system can be used. One type of debugging tool including hardware and software is provided: the sophisticated and powerful in-circuit emulator, SMDS2+, for KS57, KS86, KS88 families of microcontrollers. The SMDS2+ is a new and improved version of SMDS2. Samsung also offers support software that includes debugger, assembler, and a program for setting options.

SHINE

Samsung Host Interface for In-Circuit Emulator, SHINE, is a multi-window based debugger for SMDS2+. SHINE provides pull-down and pop-up menus, mouse support, function/hot keys, and context-sensitive hyper-linked help. It has an advanced, multiple-windowed user interface that emphasizes ease of use. Each window can be sized, moved, scrolled, highlighted, added, or removed completely.

SAMA ASSEMBLER

The Samsung Arrangeable Microcontroller (SAM) Assembler, SAMA, is a universal assembler, and generates object code in standard hexadecimal format. Assembled program code includes the object code that is used for ROM data and required SMDS program control data. To assemble programs, SAMA requires a source file and an auxiliary definition (DEF) file with device specific information.

SASM57

The SASM57 is an relocatable assembler for Samsung's KS57-series microcontrollers. The SASM57 takes a source file containing assembly language statements and translates into a corresponding source code, object code and comments. The SASM57 supports macros and conditional assembly. It runs on the MS-DOS operating system. It produces the relocatable object code only, so the user should link object file. Object files can be linked with other object files and loaded into memory.

HEX2ROM

HEX2ROM file generates ROM code from HEX file which has been produced by assembler. ROM code must be needed to fabricate a microcontroller which has a mask ROM. When generating the ROM code (.OBJ file) by HEX2ROM, the value 'FF' is filled into the unused ROM area upto the maximum ROM size of the target device automatically.

TARGET BOARDS

Target boards are available for all KS57-series microcontrollers. All required target system cables and adapters are included with the device-specific target board.

OTPs

One time programmable microcontroller (OTP) for the S3C7295 microcontroller and OTP programmer (Gang) are now available.

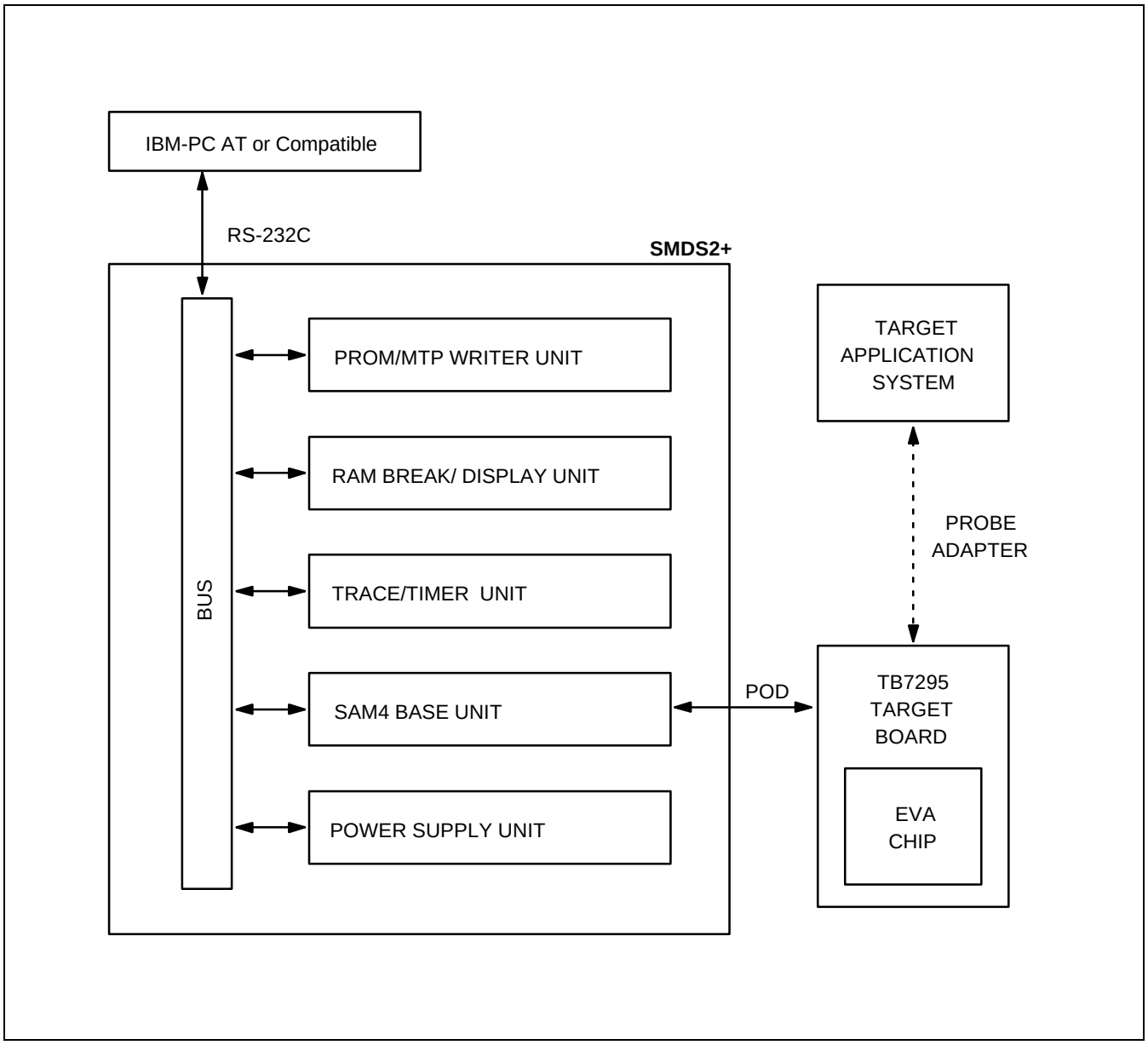


Figure 16-1. SMDS Product Configuration (SMDS2+)

TB7295 TARGET BOARD

The TB7295 target board is used for the S3C7295/P7295 microcontroller. It is supported by the SMDS2+ development system.

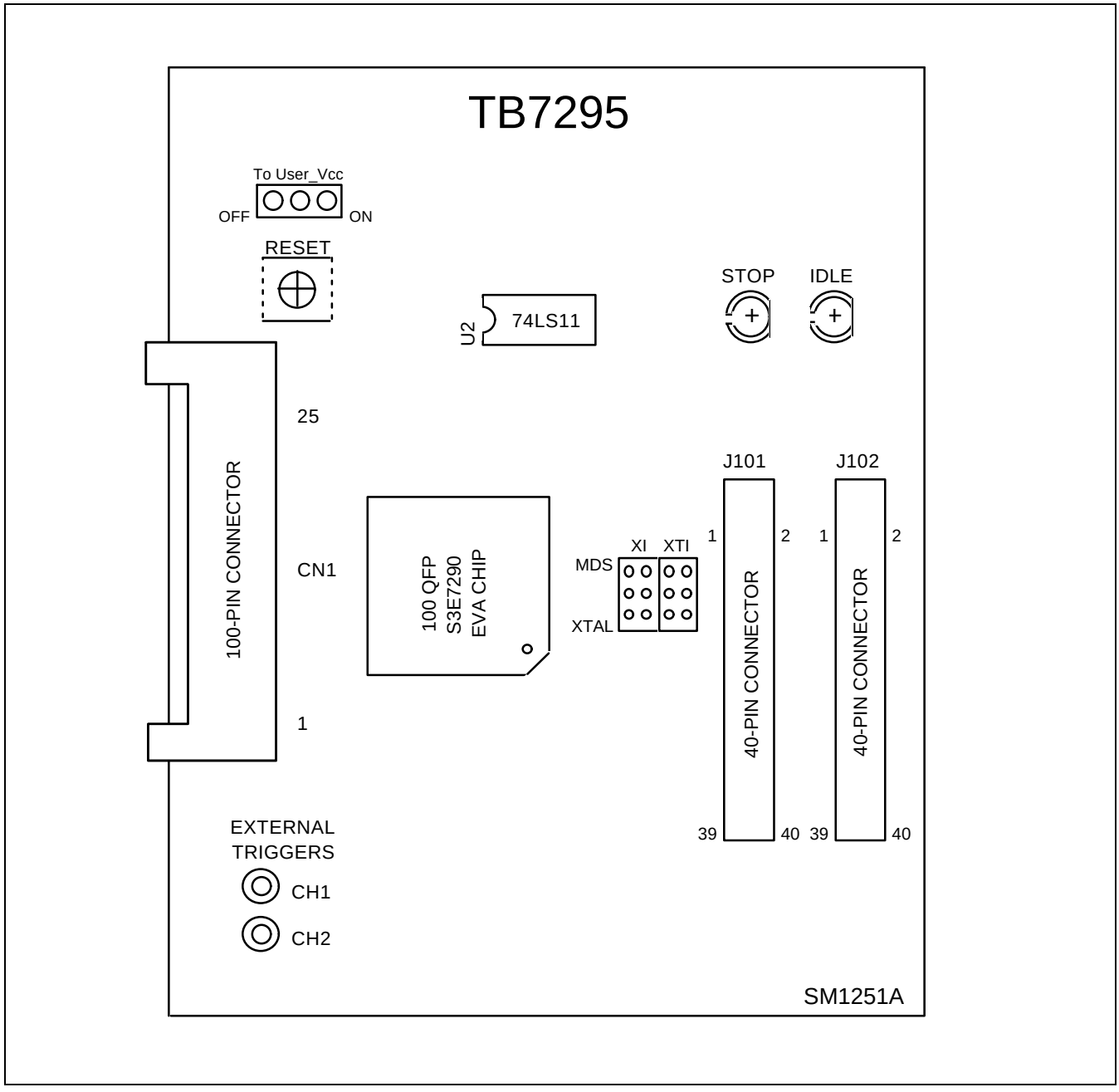


Figure 16-2. TB7295 Target Board Configuration

Table 16-1. Power Selection Settings for TB7295

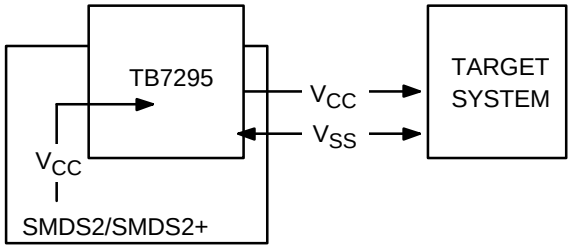
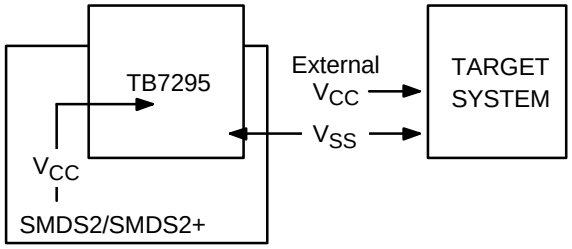
'To User_Vcc' Settings	Operating Mode	Comments
To User_Vcc OFF  ON		The SMDS2/SMDS2+ supplies V_{CC} to the target board (evaluation chip) and the target system.
To User_Vcc OFF  ON		The SMDS2/SMDS2+ supplies V_{CC} only to the target board (evaluation chip). The target system must have its own power supply.

Table 16-2. Main-clock Selection Settings for TB7295

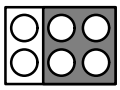
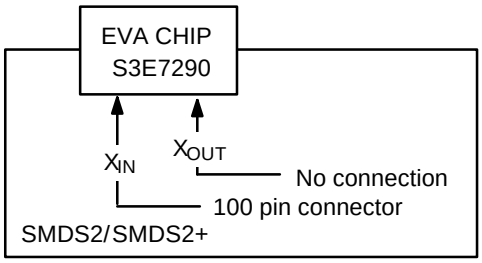
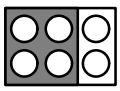
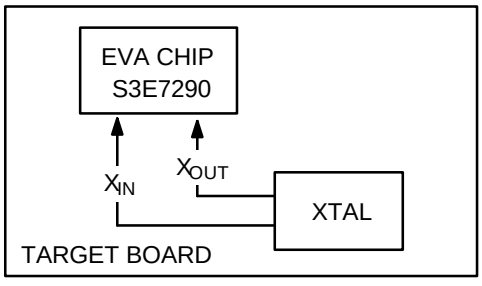
Sub Clock Setting	Operating Mode	Comments
XI XTAL  MDS		Set the XI switch to “MDS” when the target board is connected to the SMDS2/SMDS2+.
XI XTAL  MDS		Set the XI switch to “XTAL” when the target board is used as a standalone unit, and is not connected to the SMDS2/SMDS2+.

Table 16-3. Sub-clock Selection Settings for TB7295

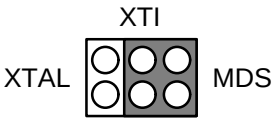
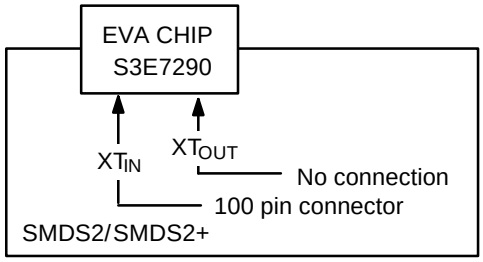
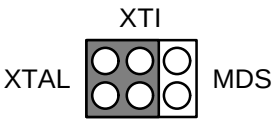
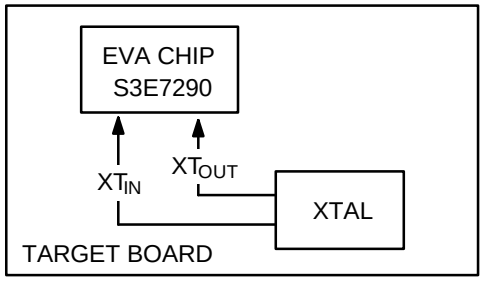
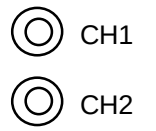
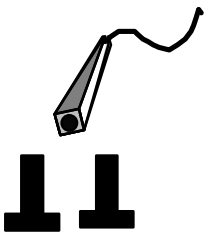
Sub Clock Setting	Operating Mode	Comments
		Set the XTI switch to “MDS” when the target board is connected to the SMDS2/SMDS2+.
		Set the XTI switch to “XTAL” when the target board is used as a standalone unit, and is not connected to the SMDS2/SMDS2+.

Table 16-4. Using Single Header Pins as the Input Path for External Trigger Sources

Target Board Part	Comments
	 You can connect an external trigger source to one of the two external trigger channels (CH1 or CH2) for the SMDS2+ breakpoint and trace functions.

IDLE LED

This LED is ON when the evaluation chip (S3E7290) is in idle mode.

STOP LED

This LED is ON when the evaluation chip (S3E7290) is in stop mode.

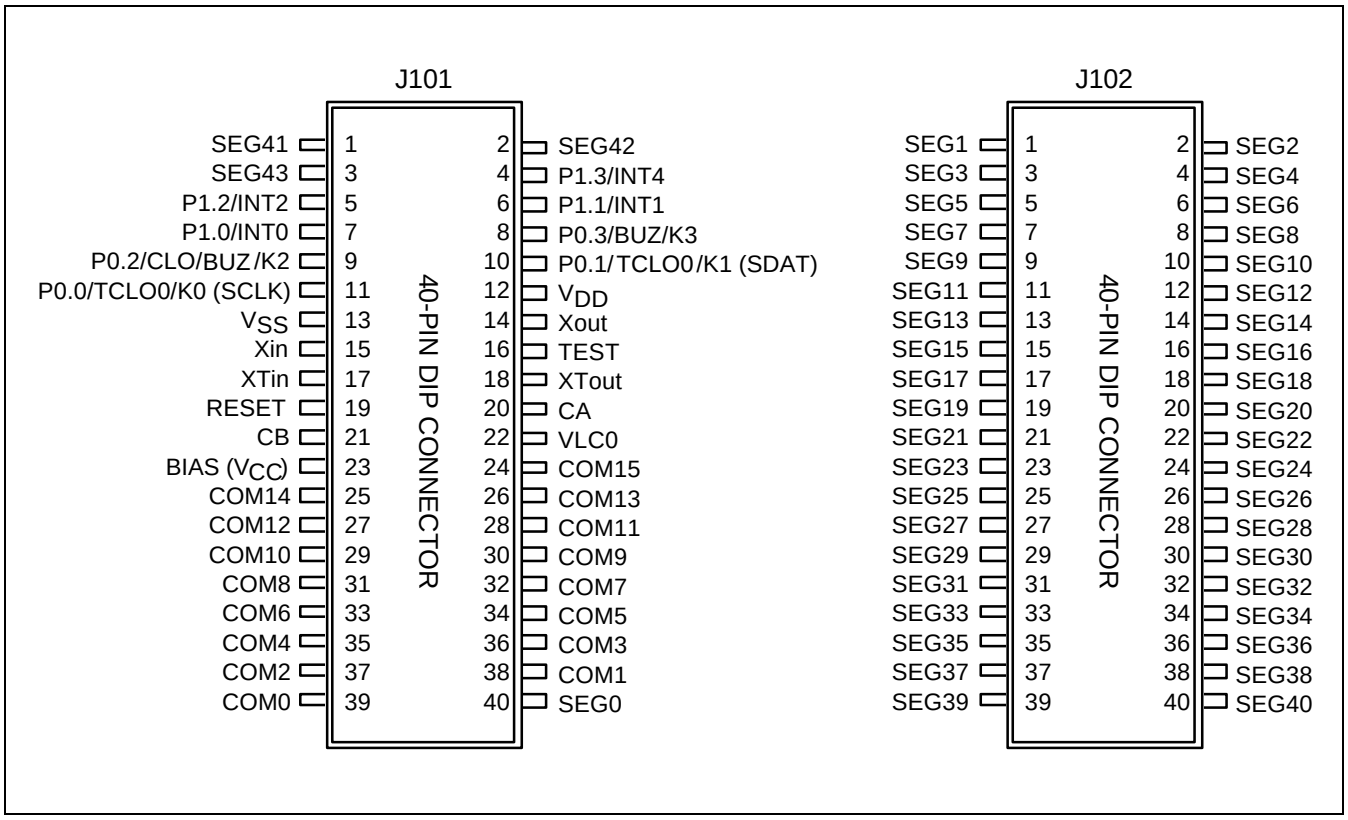


Figure 16-3. 40-Pin Connectors for TB7295

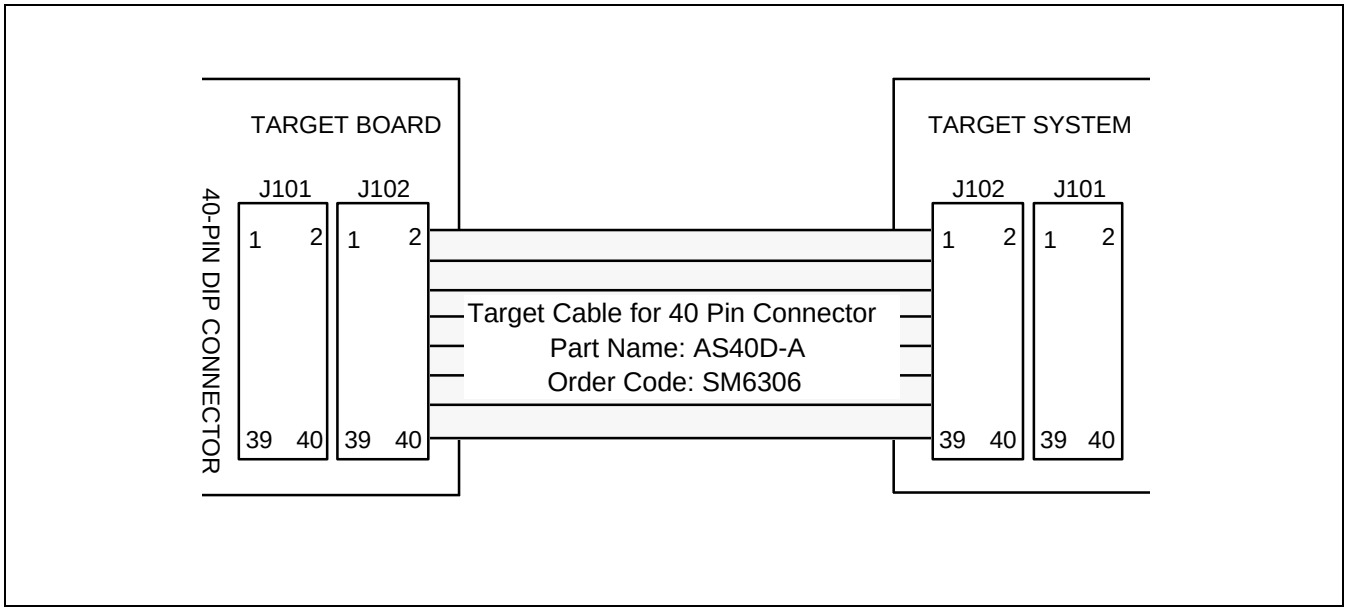


Figure 16-4. TB7295 Adapter Cable for 80-QFP Package (S3C7295/P7295)