1 PRODUCT OVERVIEW

OVER VIEW

The S3C7031/7032 single-chip CMOS microcontroller has been designed for high performance using Samsung's newest 4-bit CPU core.

With comparator inputs, high-current LED direct-drive pins, serial I/O interface, and a versatile 8-bit timer/counter, the S3C7031/7032 offers an excellent design solution for a wide range of applications such as mouse controllers, subsystem controllers, and toys.

Up to 15 pins of the 20-pin DIP or 20-pin SOP package can be dedicated to I/O. Pull-up resistors are assignable to all of the pins by software. Four vectored interrupts provide fast response to internal and external events.

In addition, the S3C7031/7032's advanced CMOS technology provides for very low power consumption and a wide operating voltage range.

DEVELOPMENT SUPPORT

The Samsung Microcontroller Development System, SMDS, provides you with a complete PC-based development environment for KS57-series microcontrollers that is powerful, reliable, and portable. In addition to its easy to use window-oriented program development structure, the SMDS toolset includes versatile debugging, trace, instruction timing, and performance measurement applications.

The Samsung Generalized Assembler (SAMA) has been designed specifically for the SMDS environment and accepts assembly language sources in a variety of microprocessor formats.

SAMA generates industry-standard object files that also contain program control data for SMDS compatibility.

FEATURES

Memory

— 1024 × 8-bit program memory (S3C7031) (ROM)

— 2048 × 8-bit program memory (S3C7032) (ROM)

— 128 × 4-bit data memory (S3C7031) (RAM)

— 256 × 4-bit data memory (S3C7032) (RAM)

I/O Pins

— Up to 15 pins for 20-DIP and 20-SOP package

Comparator Inputs

— 4-channel mode
  Internal reference: 4-bit resolution

— 3-channel mode
  External reference

8-Bit Basic Timer

— Programmable interval timer

8-Bit Timer/Counter

— Programmable interval timer

— External event counter function

— Timer clock output to TIO pin

Watch Timer

— Time interval generation: 0.5 s, 3.9 ms at 4.19 MHz

— Four frequency outputs to BUZ pin

Bit Sequential Carrier

— 16-bit serial data transfer in arbitrary format

8-Bit Serial I/O Interface

— 8-bit transmit/receive mode

— 8-bit receive-only mode

— LSB-first or MSB-first transmission selectable

— Internal or external clock source

Interrupts

— One external interrupt vector

— Three internal interrupt vectors

— Two quasi-interrupts

Memory-Mapped I/O Structure

Two Power-Down Modes

— Idle mode: Only the CPU clock stops

— Stop mode: Main system clock stops

On-Chip Crystal, Ceramic, Or RC Oscillator

— Crystal/ceramic: 4.19 MHz (typical)

— RC: 1 MHz (typical)

— CPU clock divider circuit (by 4, 8, or 64)

Frequency Outputs

— Eight frequency outputs to the CLO pin

Instruction Execution Times

— 0.95, 1.91, 15.3 µs at 4.19 MHz (5 V),
  4 µs at 1 MHz (2.7 V)

Operating Temperature:

— – 40°C to 85°C

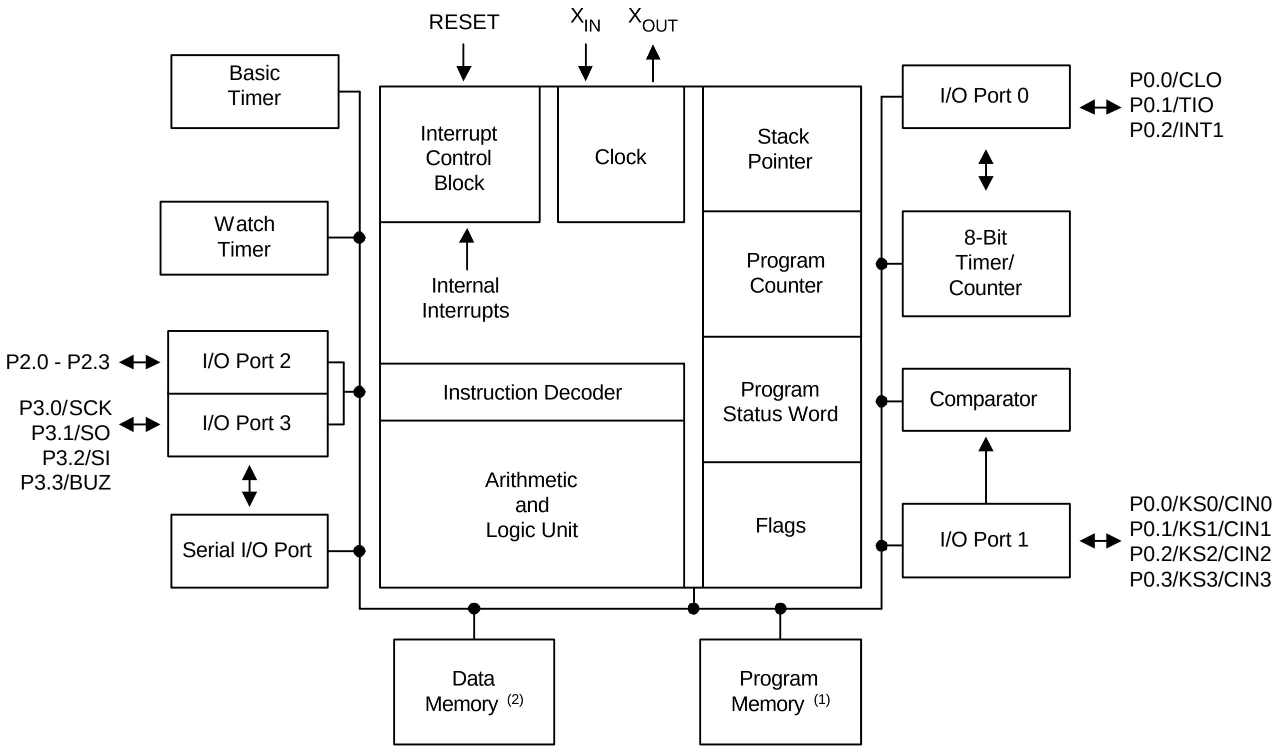
Operating Voltage Range:

— 2.7 V to 6.0 V

Package Type:

— 20-DIP, 20-SOP

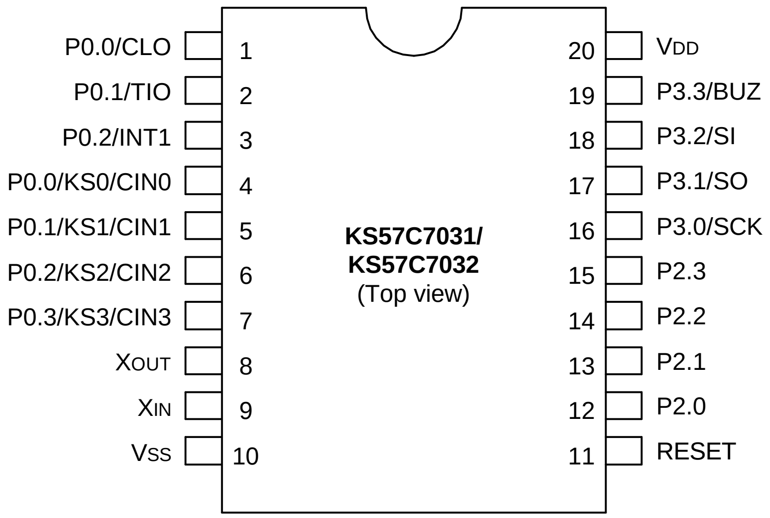
BLOCK DIAGRAM



NOTES:
1. Program Memory is 1-KByte (S3C7031) and 2-KByte (S3C7032).
2. Data Memory is 128 x 4bit (S3C7031) and 256 x 4bit (S3C7032).

Figure 1-1. S3C7031/7032 Block Diagram

PIN ASSIGNMENTS



NOTE: Pin assignments are identical for the 20-pin DIP and SOP package.

Figure 1-2. S3C7031/7032 Pin Assignment Diagram (20-pin DIP/SOP Package)

PIN DESCRIPTIONS

Table 1-1. S3C7031/7032 Pin Descriptions

Pin Name	Pin Type	Description	Number	Share Pin
P0.0 P0.1 P0.2	I/O	3-bit I/O port. 1-bit or 3-bit read/write and test is possible. Pull-up resistors are individually assignable to input pins by software and are automatically disabled for output pins. Pins are individually configurable as input or output.	1 2 3	CLO TIO INT1
P1.0 P1.1 P1.2 P1.3	I/O	Same as port 0 except that port 1 is a 4-bit I/O port.	4 5 6 7	KS0/CIN0 KS1/CIN1 KS2/CIN2 KS3/CIN3

Table 1-1. S3C7031/7032 Pin Descriptions (Continued)

Pin Name	Pin Type	Description	Number	Share Pin
P2.0-P2.3 P3.0 P3.1 P3.2 P3.3	I/O	4-bit I/O port. 1-bit, 4-bit or 8-bit read/write and test is possible. Pins are individually configurable as input or output. Pull-up resistors are individually assignable to input pins by software and are automatically disabled for output pins. Ports are software configurable as n-channel open-drain outputs or push-pull output by software. Ports 2 and 3 can be paired to enable 8-bit data transfer.	12-15 16 17 18 19	–- SCK SO SI BUZ
CLO	I/O	Eight frequency outputs	1	P0.0
TIO	I/O	External clock input or timer clock output	2	P0.1
INT1	I/O	External interrupts with rising or falling edge detection	3	P0.2
KS0-KS3	I/O	Quasi-interrupts with falling edge detection	4-7	P1.0-P1.3
CIN0-CIN3	I/O	4-channel comparator input. CIN0-CIN2: comparator input only. CIN3: comparator input or external reference input	4-7	P1.0-P1.3
SCK	I/O	Serial interface clock signal	16	P3.0
SO	I/O	Serial data output	17	P3.1
SI	I/O	Serial data input	18	P3.2
BUZ	I/O	2 kHz, 4 kHz, 8 kHz, or 16 kHz frequency output at 4.19 MHz for buzzer sound	19	P3.3
X_{IN}, X_{OUT}	–-	Crystal, ceramic, or RC signal for system clock	9, 8	–-
RESET	I	Reset signal	11	–-
V_{DD}	–-	Power supply	20	–-
V_{SS}	–-	Ground	10	–-

Table 1-2. Overview of S3C7031/7032 Pin Data

Pin Numbers	Pin Names	Share Pins	I/O Type	Reset Value	Circuit Type
1-3	P0.0-P0.2	CLO, TIO, INT1	I/O	Input	2
4-7	P1.0-P1.3	KS0/CIN0-KS3/CIN3	I/O	Input	4
12-5	P2.0-P2.3	–	I/O	Input	3
16-19	P3.0-P3.3	SCK, SO, SI, BUZ	I/O	Input	3
11	RESET	–-	I	–-	1
20, 10	V_{DD}, V_{SS}	–-	–-	–-	–-
9, 8	X_{IN}, X_{OUT}	–-	–-	–-	–-

PIN CIRCUIT DIAGRAMS

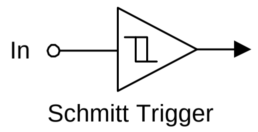


Figure 1-3. Pin Circuit Type 1

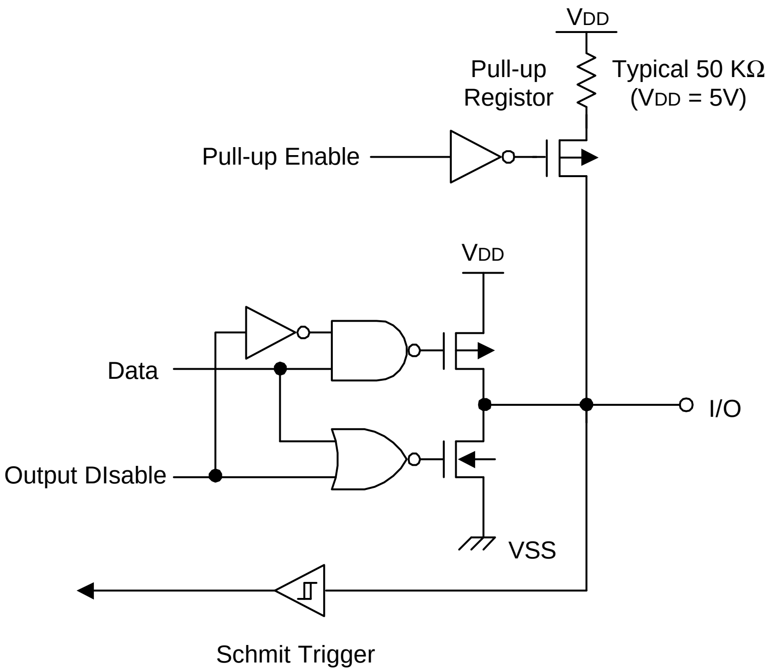


Figure 1-4. Pin Circuit Type 2

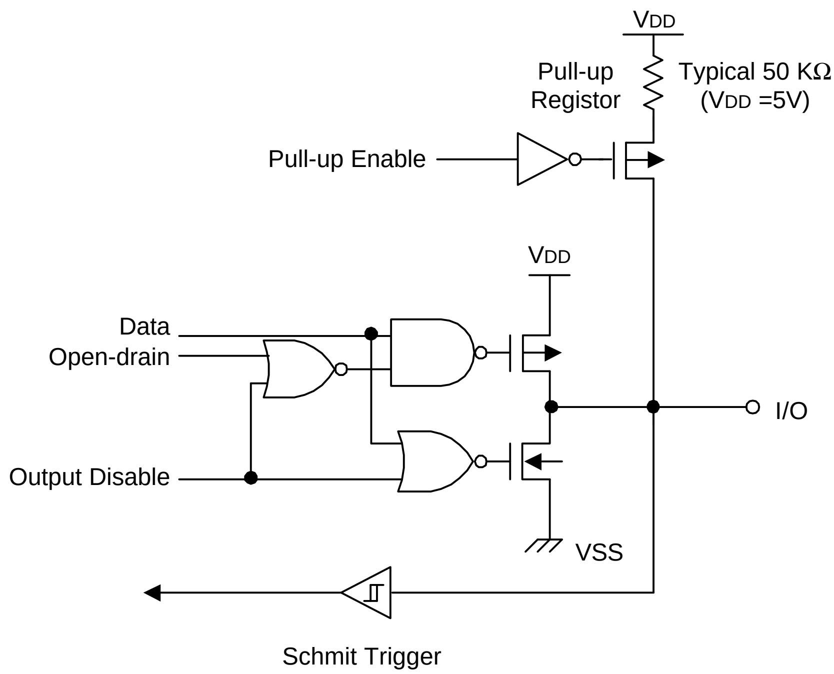


Figure 1-5. Pin Circuit Type 3

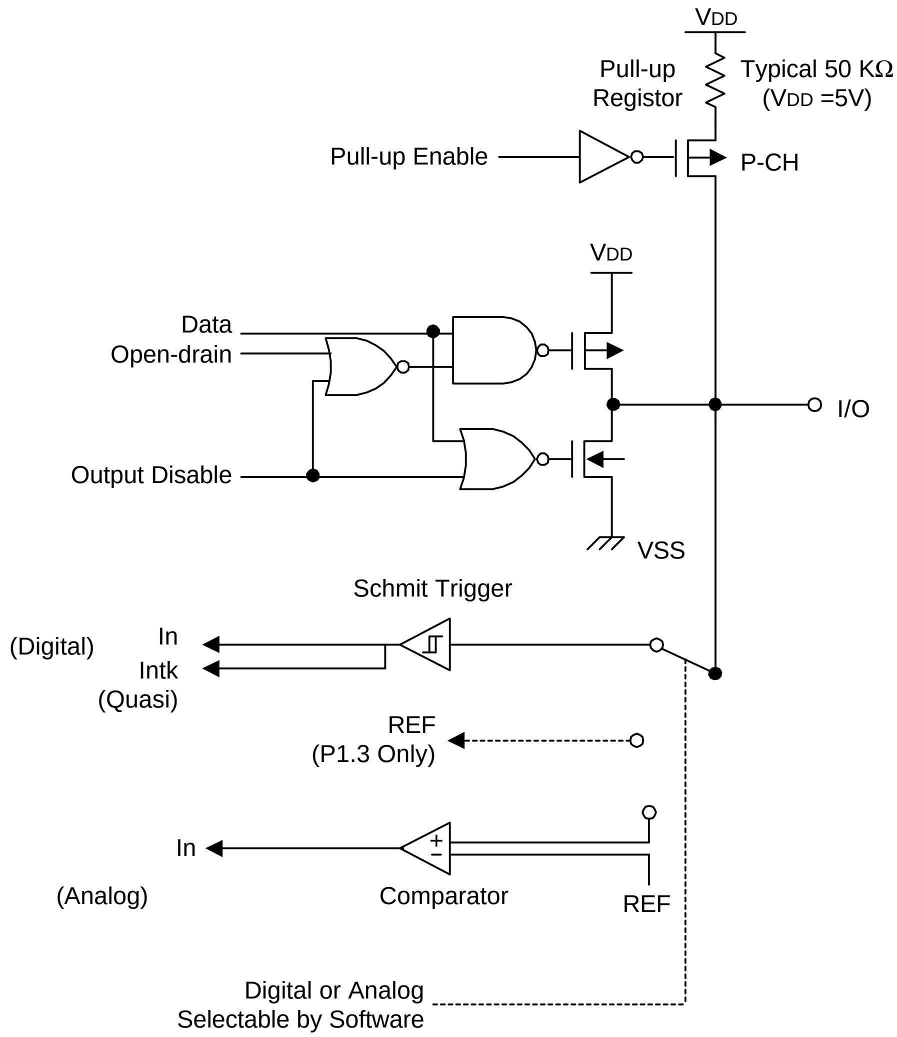


Figure 1-6. Pin Circuit Type 4

2 ADDRESS SPACES

PROGRAM MEMORY (ROM)

OVERVIEW

ROM maps for S3C7031/7032 devices are mask programmable at the factory. In its standard configuration, the device's 2048 × 8-bit program memory has three areas that are directly addressable by the program counter:

— 16-byte general-purpose area

— 896-byte general-purpose area (S3C7031)

— 1920-byte general-purpose area (S3C7032)

— 16-byte area for vector addresses

— 96-byte instruction reference area

General-Purpose Memory

Two program memory areas are allocated for general-purpose use: One area is 16 bytes in size and the other is 896 bytes (S3C7031) and 1920 bytes (S3C7032).

Vector Addresses

You use the 16-byte vector address area to store the vector addresses required to execute system resets 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. Using REF instructions, you can reduce the byte size of instruction operands. REF can reference either one 2-byte or two 1-byte instructions 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-03FFH 0080H-07FFH	896 (S3C7031) 1920 (S3C7032)

GENERAL-PURPOSE MEMORY AREAS

The 16-byte area at ROM locations 0010H-001FH and the 896-byte area at ROM locations 0080H-03FFH are used as general-purpose program memory. (S3C7031)
The 16-byte area at ROM locations 0010H-001FH and the 1920-byte area at ROM locations 0080H-07FFH are used as general-purpose program memory. (S3C7032)

You can also use vacant locations in the vector address area and REF instruction look-up table areas as general-purpose program memory. But please be careful not to overwrite live data when writing programs that use special-purpose areas of the ROM.

VECTOR ADDRESS AREA

Use the 16-byte vector address area of the ROM 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 set EMB and ERB's initial values for the service routines. A 16-byte vector address is organized as follows:

EMB	ERB	0	0	0	PC10*	PC9	PC8
PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

* This mark is only for the S3C7032

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

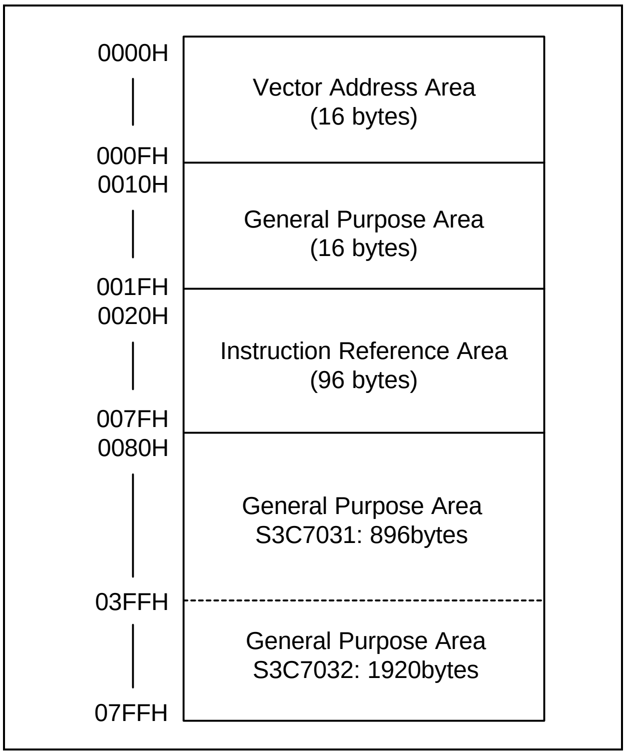


Figure 2-1. ROM Structure

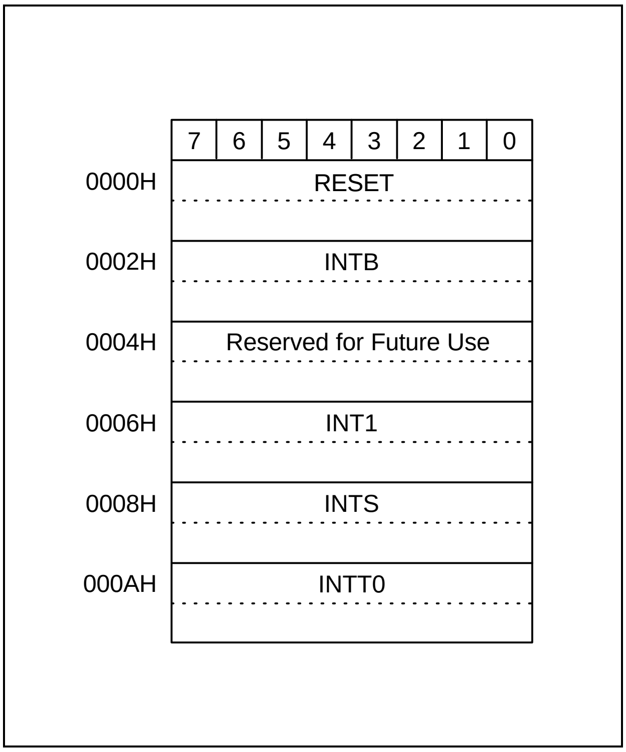


Figure 2-2. Vector Address Map

☞ PROGRAMMING TIP — Defining Vectored Interrupt Areas

The following examples show several ways that 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
            VENT1   0,0,INTB        ;  EMB ← 0, ERB ← 0; Jump to INTB address
            ORG     0006H
            VENT3   0,0,INT1        ;  EMB ← 0, ERB ← 0; Jump to INT1 address
            VENT4   0,0,INTS        ;  EMB ← 0, ERB ← 0; Jump to INTS address
            VENT5   0,0,INTT0       ;  EMB ← 0, ERB ← 0; Jump to INTT0 address
```

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

```
            ORG     0000H
;
            VENT0   1,0,RESET       ;  EMB ← 1, ERB ← 0; Jump to RESET address
            VENT1   0,0,INTB        ;  EMB ← 0, ERB ← 0; Jump to INTB address
            ORG     0006H
            VENT3   0,0,INT1        ;  EMB ← 0, ERB ← 0; Jump to INT1 address
            VENT4   0,0,INTS        ;  EMB ← 0, ERB ← 0; Jump to INTS address
;
            ORG     0010H           ;  INTT0 interrupt not used
```

3. 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
            VENT1   0,0,INTB        ;  EMB ← 0, ERB ← 0; Jump to INTB address
            VENT3   0,0,INT1        ;  EMB ← 0, ERB ← 0; Jump to 0004/5H Data
            VENT4   0,0,INTS        ;  EMB ← 0, ERB ← 0; Jump to INT1 address
            VENT5   0,0,INTT0       ;  EMB ← 0, ERB ← 0; Jump to INTS address
;
            ORG     0010H
;
            General-purpose ROM area
;
```

In this example, when an INTS interrupt is generated, the corresponding vector area is not VENT4 INTS, but VENT5 INTT0. This makes an INTS interrupt jump incorrectly to the INTT0 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 one-byte instructions, a single two-byte instruction, or three-byte instructions such as a JP 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:

— 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 256 × 4-bit data memory has three areas:

— 32 × 4-bit working register area
— 96 x 4-bit working register area (also used as stack area) (S3C7031)
— 224 × 4-bit general-purpose area (also used as stack area) (S3C7032)
— 128 × 4-bit area for memory-mapped I/O addresses

To simplify referencing, the data memory area has two memory banks — bank 0 and bank 15. The select memory bank instruction (SMB) selects the bank you want to use as working data memory. Data stored in RAM locations are 1-bit, 4-bit, and 8-bit addressable. Initialization values for the data memory area are not defined by hardware and must therefore be initialized by program software following RESET. When the RESET signal is generated in power-down mode, the data memory contents are retained.

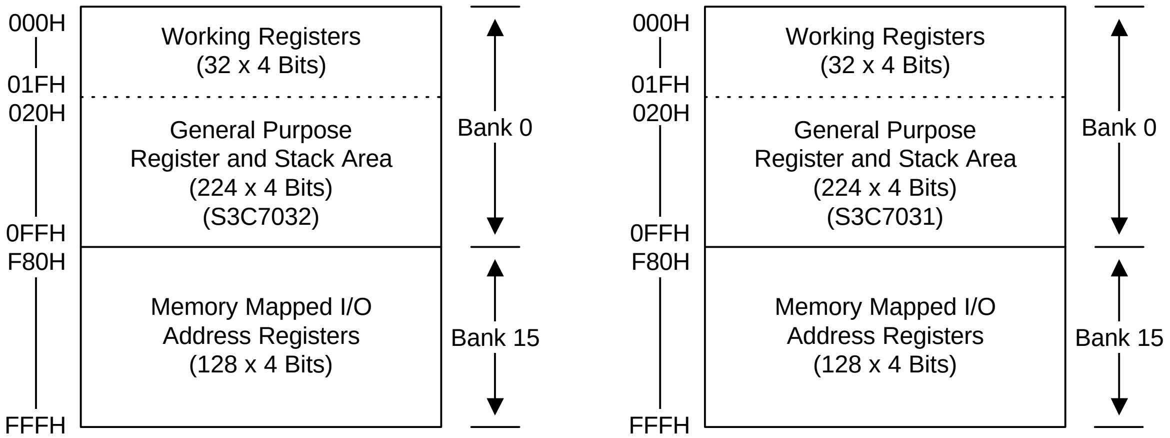


Figure 2-3. Data Memory (RAM) Map

Memory Banks 0 and 15

Bank 0	(000H-0FFH)	The lowest 32 nibbles of bank 0 (000H-01FH) are used as working registers; the next 96 nibbles (020H-07FH) 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. (S3C7031)
Bank 0	(000H-FFFH)	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. (S3C7032)
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 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 000H107FH 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 two 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. 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 is 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.

Bit Sequential Carrier (BSC)

The bit sequential carrier (BSC) is a 16-bit general register mapped to RAM addresses FC0H-FC3H that can be manipulated by 1-bit, 4-bit, and 8-bit RAM control instructions. RESET clears all bit values to logic zero.

You can specify addresses and bit locations sequentially using a 1-bit indirect addressing instruction. In this way, a program can process 16-bit data by moving the bit location sequentially, incrementing or decrementing the value of the L register. BSC data can also be manipulated by direct addressing. For 8-bit manipulations, you must address the upper and lower eight bits separately.

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 (S3C7031) 020-0FFH (S3C7032)	Stack and general-purpose registers			
F80H-FFFH	I/O-mapped hardware registers	15	0, 1	15

☞ PROGRAMMING TIP — Clearing Data Memory Bank 0

The following instructions clear bank 0 of the data memory area:

```
RAMCLR   BITS   EMB
         SMB    0
         LD     HL,#10H
         LD     A,#0H
RMCL0    LD     @HL,A            ;   RAM (010H-0FFH) clear
         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. You can manipulate working register data as 1-bit units, 4-bit units or, using paired registers, as 8-bit units.

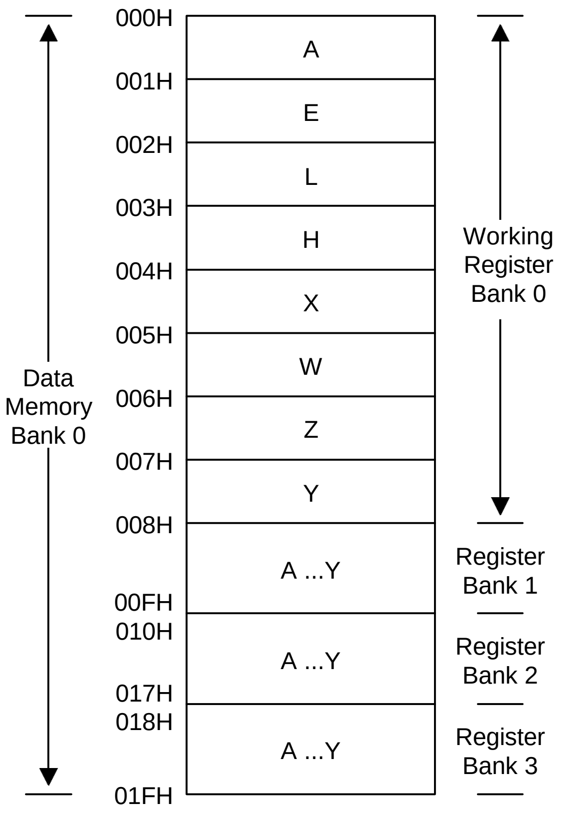


Figure 2-4. Working Register Map

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 (SRBn) 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 are named Y, Z, W, X, H, L, E and A. You can manipulate them individually using 4-bit instructions, or 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.

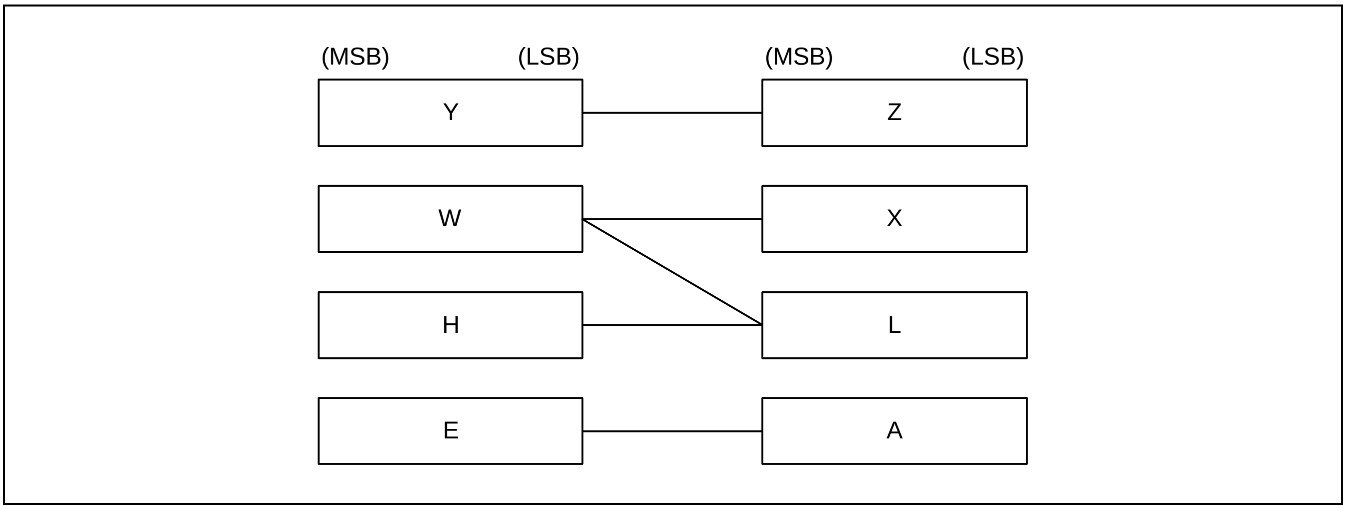


Figure 2-5. Register Pair Configuration

Special-Purpose Working Registers

You use register A as a 4-bit accumulator and double register EA as an 8-bit accumulator. The carry flag can 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 and increment or decrement L register value using a single instruction.

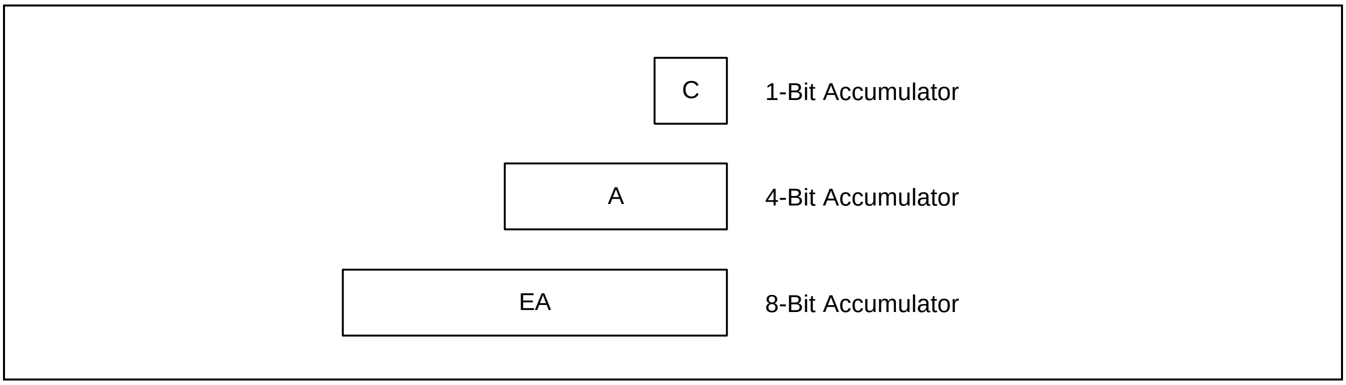


Figure 2-6. 1-Bit, 4-Bit, and 8-Bit Accumulator

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 Your Working Register Area

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

1. When ERB = "0":

```
VENT3   1,0,INT1           ;  EMB ← 1, ERB ← 0, Jump to INT1 address
;
INT0    PUSH    SB         ;  PUSH current SMB, SRB
        SRB     2          ;  Non-essential instruction, since ERB = "0"
        PUSH    HL         ;  PUSH HL register to stack
        PUSH    WX         ;  PUSH WX register to stack
        PUSH    YZ         ;  PUSH YZ register to stack
        PUSH    EA         ;  PUSH EA register 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 from stack
        POP     YZ         ;  POP YZ register from stack
        POP     WX         ;  POP WX register from stack
        POP     HL         ;  POP HL register 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":

```
VENT3   1,1,INT1           ;  EMB ← 1, ERB ← 1, Jump to INT1 address
;
INT0    PUSH    SB         ;  Store current SMB, SRB
        SRB     2          ;  Select register bank 2
        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 is mapped to RAM addresses F80H-F81H, and 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. Because the reset value of the stack pointer is not defined in hardware, 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.

Although you may use general-purpose register areas for stack operations, be careful to avoid data loss due to simultaneous use of the same register(s).

☞ PROGRAMMING TIP — Initializing the Stack Pointer

To initialize the stack pointer (SP):

1. When EMB = "1":

```
        SMB     15          ;  Select memory bank 15
        LD      EA,#7EH     ;  Bit 0 of accumulator A is always cleared to "0" (For the S3C7031)
        LD      EA,#00H     ;  Bit 0 of accumulator A is always cleared to "0" (For the S3C7032)
        LD      SP,EA       ;  Stack area initial address (0FFH) ← (SP) – 1
```

2. When EMB = "0":

```
        LD      EA,#7EH (for the S3C7031)
        LD      EA,#00H (for the S3C7032)
        LD      SP,EA       ;  Memory addressing area  (00H-7FH, F80H-FFFH)
```

PUSH Operations

Three kinds of push operations reference the stack pointer (SP) in order 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 from the PC 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 four 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. After the CALL has executed, the SP is decremented *by six* and points to the next available stack location. Because six 4-bit stack locations are used per CALL, you can 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, as well as current values for 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.

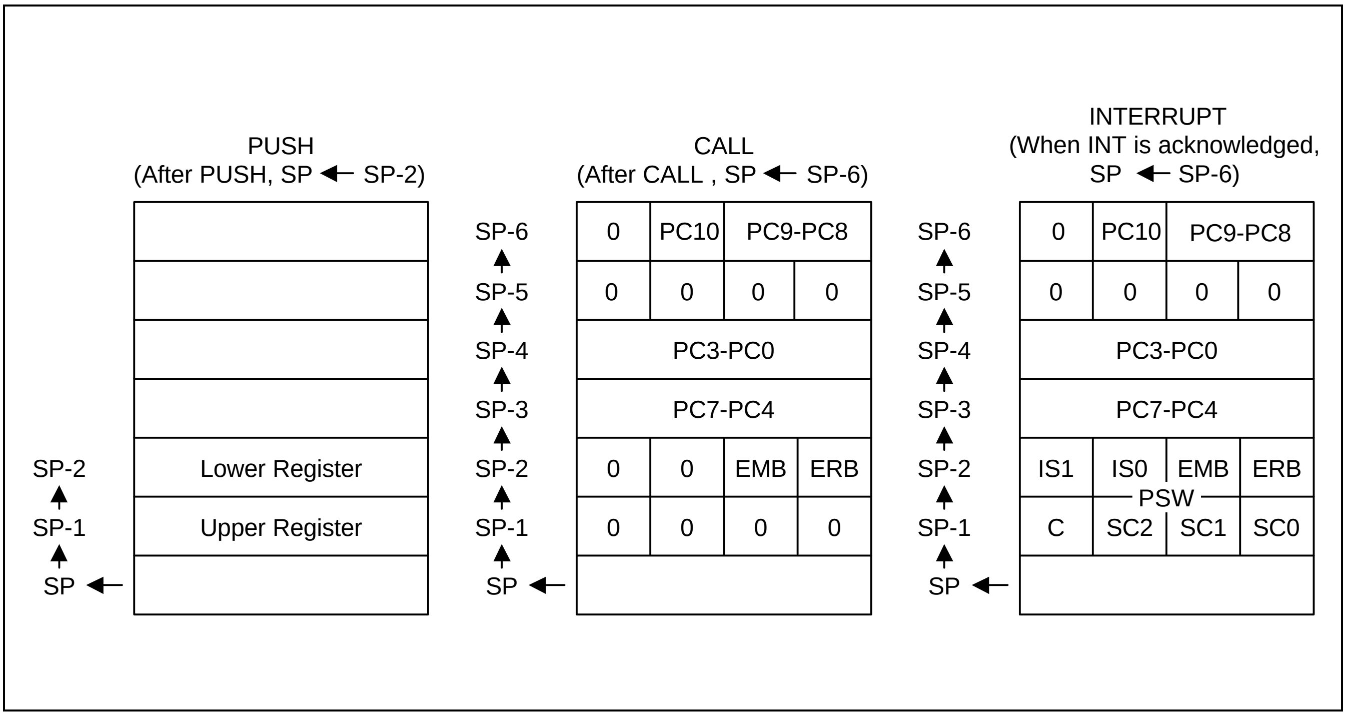


Figure 2-7. Push-Type Stack Operations

SAMSUNG ELECTRONICS

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, it is 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 for the lower 4-bit register is popped first, followed by the value for 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 four 4-bit stack locations used for the CALL and to write this data back to the PC, the EMB, and 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.

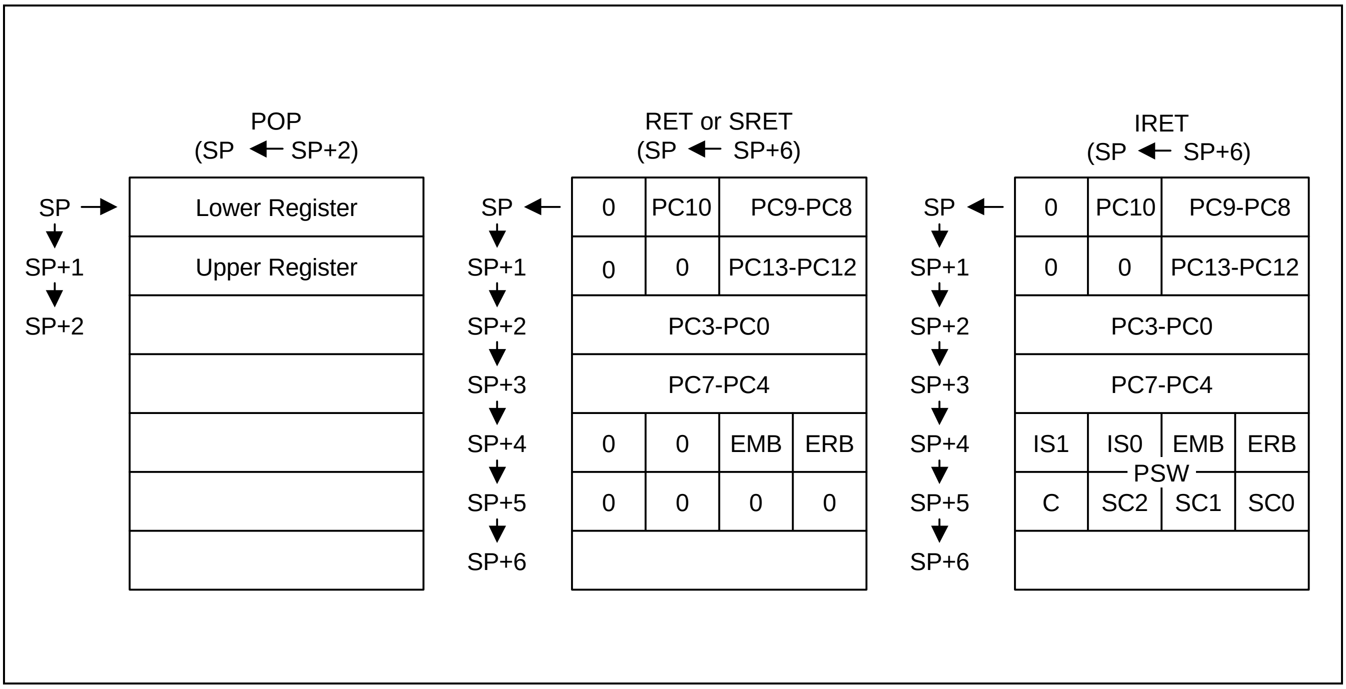


Figure 2-8. Pop-Type Stack Operations

BIT SEQUENTIAL CARRIER (BSC)

The bit sequential carrier (BSC) is a 16-bit register that is mapped to RAM addresses FC0H-FC3H. You can manipulate the BSC register using 1-bit, 4-bit, and 8-bit RAM control instructions. RESET clears all BSC bit values to logic zero.

Using the BSC, you can specify addresses and bit locations sequentially using 1-bit indirect addressing (memb.@L). Bit addressing is independent of the current EMB value. In this way, programs can process 16-bit data by moving the bit location sequentially and then incrementing or decrementing the value of the L register.

BSC data can also be manipulated using direct addressing. For 8-bit manipulations, specify the 4-bit register names BSC0 and BSC2 and manipulate the upper and lower 8 bits manipulated separately.

If the values of the L register are 0H at BSC0.@L, the address and bit location assignment is FC0H.0. If the L register content is FH at BSC0.@L, the address and bit location assignment is FC3H.3.

Table 2-4. BSC Register Organization

Name	Address	Bit 3	Bit 2	Bit 1	Bit 0
BSC0	FC0H	BSC0.3	BSC0.2	BSC0.1	BSC0.0
BSC1	FC1H	BSC1.3	BSC1.2	BSC1.1	BSC1.0
BSC2	FC2H	BSC2.3	BSC2.2	BSC2.1	BSC2.0
BSC3	FC3H	BSC3.3	BSC3.2	BSC3.1	BSC3.0

☞ PROGRAMMING TIP — Using the BSC Register to Output 16-Bit Data

To use the bit sequential carrier (BSC) register to output 16-bit data (5937H) to the P3.0 pin:

```
        BITS    EMB
        SMB     15
        LD      EA,#37H
        LD      BSC0,EA         ;  BSC0 ← A, BSC1 ← E
        LD      EA,#59H
        LD      BSC2,EA         ;  BSC2 ← A, BSC3 ← E
        SMB     0
        LD      L,#0H
AGN     LDB     C,BSC0.@L
        LDB     P3.0,C          ;  P3.0 ← C
        INCS    L
        JR      AGN
        RET
```

PROGRAM COUNTER (PC)

A 11-bit program counter (PC) stores addresses for instruction fetches during program execution. Whenever a reset operation or an interrupt occurs, bits PC10 through PC0 are set to the vector address. Bit PC11-PC13 is reserved to support future expansion of the device's ROM size.

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, mapped to RAM locations FB0H-FB1H, 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:

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 logic zero.

Table 2-5. 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. They are mapped to RAM bit locations FB0H.2 and FB0H.3, respectively. 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 2-6 shows the effects of IS0 and IS1 flag settings.

Because 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.

Table 2-6. 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 by interrupt priority register (IPR) settings are serviced
1	0	2	No more interrupt requests are serviced
1	1	—	Not applicable; these bit settings are undefined

☞ PROGRAMMING TIP — Setting ISx Flags for Interrupt Processing

The following instructions show 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 enable memory bank flag EMB is mapped to registers FB0H-FB1H in bank 15 of the RAM. The EMB flag occupies bit location 1 in register FB0H.

The EMB flag is used to allocate specific address locations in the RAM by modifying the upper four bits of 12-bit data memory addresses. In this way, it controls the addressing mode for data memory banks 0 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", you can access general-purpose areas of bank 0 and bank 15 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     0           ;  Non-essential instruction because 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 because 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     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", you select the working register area from register banks 0 to 3 according to the register bank selection register (SRB). When the ERB flag is "0", you select register bank 0 as the 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 vector 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 (because 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 ¨ BANK 2 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 indicate the current program skip conditions and are set and reset automatically during program execution. These flags are mapped to RAM bit locations FB1H.0, FB1H.1, and FB1H.2 of the PSW.

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 mapped to bit location FB1H.3 in the PSW. It 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 2-7, affect the carry flag.

Table 2-7. 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 P3.3 and output the result to P2.0:

```
LD      H,#3H         ;  Set upper four bits of address to the H register value
LDB     C,@H+0FH.3    ;  C ← bit 3 of 3FH
BAND    C,P3.3        ;  C ← C AND P3.3
LDB     P2.0,C        ;  Output result from carry flag to P2.0
```

3 ADDRESSING MODES

OVERVIEW

The enable memory bank flag, EMB, controls the two addressing modes for data memory. When you enable the EMB flag, you can address the entire RAM area. When you clear the EMB flag to logic zero, the addressable RAM is restricted to specific areas.

The EMB flag works in connection with the select memory bank instruction, SMB n. You will recall that the SMB n instruction is used to select RAM bank 0 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 or 15. Direct and indirect 1-bit, 4-bit, and 8-bit addressing methods can be used.

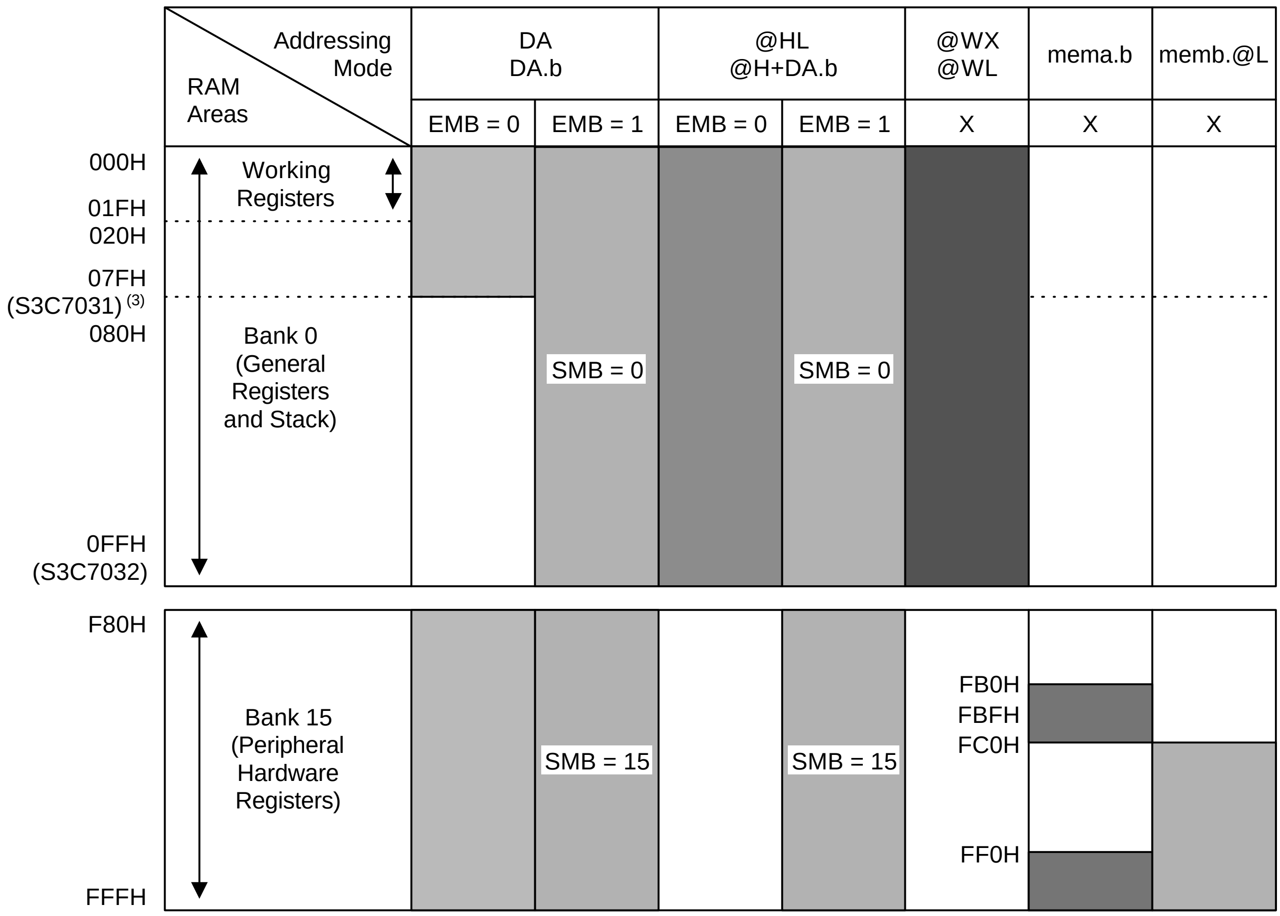
In addition, there are several RAM locations that can always be addressed using specific addressing methods, regardless of the current EMB flag setting.

Here are a few things to remember about addressing data memory areas:

— When you address peripheral hardware locations in bank 15, you can use the mnemonic for the memory-mapped hardware component 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.
3. The bank 0 Area in the S3C7031 is from location 000H to 07FH

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 been completed.

At the beginning of a program, the initial EMB flag value for each vectored interrupt must be set by using VENT instruction. The EMB can be set or reset by bit manipulation instructions (BITS, BITR), regardless of 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, branch RESET
        VENT1   0,1,INTB        ;  EMB ← 0, ERB ← 1. branch INTB
        ORG     0006H
        VENT3   0,1,INT1        ;  EMB ← 0, ERB ← 1, branch INT1
        VENT4   0,1,INTS        ;  EMB ← 0, ERB ← 1, branch INTS
        VENT5   0,1,INTT0       ;  EMB ← 0, ERB ← 1, branch INTT0
        •
        •
        •
RESET   BITR    EMB
```

ENABLE MEMORY BANK SETTINGS

EMB = "1"

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

If SMB = 0, 000H-07FH (S3C7031)

If SMB = 0, 000H-0FFH (S3C7032)

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

You can address several areas of the data memory at any time, despite the 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, IEx, IRQx, I/O	BITS EMB BITR IE1
FC0H-FFFH	1-bit indirect addressing using the L register	BSC, I/O	BTST FC3H.@L BAND C,P3.@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.

	SMB				SRB			
SB Register	SMB 3	SMB 2	SMB 1	SMB 0	0	0	SRB 1	SRB 0

Figure 3-2. 4-Bit SMB and SRB Values in the SB Register

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.

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 you 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 two available data memory banks, you must execute an SMB n instruction specifying the number of the memory bank you want (0 or 15). For example, the instruction 'SMB 1' selects bank 1 and 'SMB 15' selects bank 15. You must also remember to enable the memory bank you select by the appropriate enable memory bank flag (EMB) 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.

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

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

In indirect addressing the instruction specifies a specific register pair which contain the address of the operand. The KS57 instruction set supports 1-bit, 4-bit, and 8-bit indirect addressing. For 8-bit indirect addressing, you must always use an even-numbered RAM address as the instruction operand. The address register can be HL, WX, or WL of the selected register bank.

1-BIT ADDRESSING

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

Instruction 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	Bank15	All 1-bit addressable peripherals (SMB = 15)
		1	000H-FFFH	SMB = 0, 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: lower two bits of register L as indicated by the upper 10 bits of RAM area (memb) and the upper two bits of register L.	x	FC0H-FFFH	Bank 15	BSCn.x Pn.n
@H + DA.b	Indirect: bit indicated by the lower four bits of the address (DA), memory bank selection, and the H register identifier.	0	000H-0FFH	Bank 0	All 1-bit addressable peripherals (SMB = 15)
		1	000H-FFFH	SMB = 0, 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           ; Non-essential instruction, since EMB = "0"
        BITS    AFLAG       ; 34H.3 ← 1
        BITS    BFLAG       ; F85H.3 (BMOD.3) ← 1
        BTST    CFLAG       ; If FBAH.0 (IRQW) = 1, skip
        BITS    BFLAG       ; Else if FBAH.0 (IRQW) = 0, F85H.3 (BMOD.3) ← 1
        BITS    P3.0        ; FF3H.0 (P3.0) ← 1
;
```

2. If EMB = "1":

```
AFLAG   EQU     34H.3
BFLAG   EQU     85H.3
CFLAG   EQU     0BAH.0
        SMB     0           ; Select memory bank 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    P3.0        ; FF3H.0 (P3.0) ← 1
;
```

1-Bit Indirect Addressing

1. If EMB = "0":

```
AFLAG   EQU     34H.3
BFLAG   EQU     85H.3
CFLAG   EQU     0BAH.0
        SMB     0           ; Non-essential instruction, since EMB = "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 (IRQW) ← 1
;
```

2. If EMB = "1":

```
AFLAG   EQU     34H.3
BFLAG   EQU     85H.3
CFLAG   EQU     0BAH.0
        SMB     0           ; Select memory bank 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

Instruction 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	Bank15	All 4-bit addressable peripherals (SMB = 15)
		1	000H-FFFH	SMB = 0, 15	
@HL	Direct: 4-bit address indicated by the memory bank selection and register HL	0	000H-0FFH	Bank 0	
		1	000H-FFFH	SMB = 0, 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 because EMB = "0"
        LD    A,P3        ; A ← (P3)
        SMB   0           ; Non-essential instruction because EMB = "0"
        LD    ADATA,A     ; (046H) ← A
        LD    BDATA,A     ; (F8EH) ← A
;
```

2. If EMB = "1":

```
ADATA   EQU   46H
BDATA   EQU   8EH
        SMB   15          ; Select memory bank 15
        LD    A,P3        ; A ← (P3)
        SMB   0           ; Select memory bank 0
        LD    ADATA,A     ; (046H) ← A
        LD    BDATA,A     ; (08EH) ← A
;
```

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

4-Bit Indirect Addressing

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

```
ADATA   EQU     46H
BDATA   EQU     66H
        SMB     15              ; Non-essential instruction because 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", exchange bank 0, 040H–046H with 060H–066H:

```
ADATA   EQU     46H
BDATA   EQU     66H
        SMB     0               ; Select memory bank 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
;
```

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	Bank15	All 8-bit addressable peripherals (SMB = 15)
		1	000H-FFFH	SMB = 0, 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, 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 because EMB = "0"
        LD    EA,P4       ; E ← (P5), A ← (P4)
        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          ; Select memory bank 15
        LD    EA,P4       ; E ← (P5), A ← (P4)
        SMB   0           ; Select memory bank 0
        LD    ADATA,EA    ; (046H) ← A, (047H) ← E
        LD    BDATA,EA    ; (08EH) ← A, (08FH) ← E
;
```

8-Bit Indirect Addressing

1. If EMB = "0":

```
ADATA   EQU   8EH
        LD    HL,#ADATA
        LD    EA,@HL      ; A ← (08EH), E ← (08FH)
```

2. If EMB = "1":

```
ADATA   EQU   46H
        SMB   0
        LD    HL,#ADATA
        LD    EA,@HL      ; A ← (046H), E ← (047H)
;
```

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. You can use 1-bit direct and indirect addressing, however, 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				
•									
•									
•									
F85H	BMOD	.3	.2	.1	.0	W	.3	Yes	No
F86H	BCNT	.3	.2	.1	.0	R	No	No	Yes
F87H		.7	6	.5	.4				
F88H	WMOD	"0"	.2	.1	(1)	W	No	No	Yes
F89H		.7	"0"	.5	.4				
•									
•									
•									
F90H	TMOD0	.3	.2	"0"	"0"	W	.3	No	Yes
F91H		"0"	.6	.5	.4				
F92H		"0"	TOE0	"0"	"0"	R/W	Yes	Yes	No
F93H									
F94H	TCNT0	.3	.2	.1	.0	R	No	No	Yes
F95H		.7	.6	.5	.4				
F96H	TREF0	.3	.2	.1	.0	W	No	No	Yes
F97H		.7	.6	.5	.4				
•									
•									
•									
FB0H	PSW	IS1	IS0	EMB	ERB	R/W	Yes	Yes	Yes
FB1H		C (2)	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									
FB5H	IMOD1	"0"	"0"	"0"	.0	W	No	Yes	No
FB6H	IMODK	.3	"0"	.1	.0	W	N o	Yes	No
FB7H									

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**
FB8H		"0"	"0"	IEB	IRQB	R/W	Yes	Yes	No
FB9H									
FBAH		"0"	"0"	IEW	IRQW	R/W	Yes	Yes	No
FBBH									
FBCH		"0"	"0"	IET0	IRQT0	R/W	Yes	Yes	No
FBDH		"0"	"0"	IES	IRQS				
FBEH		IE1	IRQ1	"0"	"0"				
FBFH		"0"	"0"	IEK	IRQK				
FC0H	BSC0	.3	.2	.1	.0	R/W	Yes	Yes	Yes
FC1H	BSC1	.3/.7	.2/.6	.1/.5	.0/.4	R/W		Yes	
FC2H	BSC2	.3	.2	.1	.0	R/W		Yes	Yes
FC3H	BSC3	.3/.7	.2/.6	.1/.5	.0/.4	R/W		Yes	
•									
•									
•									
FD0H	CLMOD	.3	.2	.1	.0	W	No	Yes	No
•									
•									
•									
FD4H	CMPREG	.3	.2	.1	.0	R	No	Yes	No
FD5H									
FD6H	CMOD	.3	.2	.1	.0	R/W	No	No	Yes
FD7H		.7	.6	.5	"0"				
•									
•									
•									
FDAH	PNE	.3	.2	.1	.0	W	No	No	Yes
FDBH		.7	.6	.5	.4				
FDCH	PUMOD1	"0"	.2	.1	.0	W	No	No	Yes
FDDH		.7	.6	.5	.4				
FDEH	PUMOD2	.3	.2	.1	.0	W	No	No	Yes
FDFH		.7	.6	.5	.4				

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

Memory Bank 15							Addressing Mode		
Address	**Register**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**R/W**	**1-Bit**	**4-Bit**	**8-Bit**
FE0H	SMOD	.3	.2	.1	.0	W	.3	No	Yes
FE1H		.7	.6	.5	"0"				
FE2H	P1MOD	.3	.2	.1	.0	W	No	Yes	No
FE3H									
FE4H	SBUF	.3	.2	.1	.0	R/W	No	No	Yes
FE5H		.7	.6	.5	.4				
FE6H									
FE7H									
FE8H	PMG1	"0"	PM0.2	PM0.1	PM0.0	W	No	No	Yes
FE9H		PM1.3	PM1.2	PM1.1	PM1.0				
FEAH	PMG2	PM2.3	PM2.2	PM2.1	PM2.0	W	No	No	Yes
FEBH		PM3.3	PM3.2	PM3.1	PM3.0				
•									
•									
•									
FF0H	PORT 0	"0"	.2	.1	.0	R/W	Yes	Yes	No
FF1H	PORT 1	.3	.2	.1	.0	R/W			
FF2H	PORT 2	.3	.2	.1	.0	R/W			Yes
FF3H	PORT 3	.3/.7	.2/.6	.1/.5	.0/.4	R/W			
•									
•									
•									
FFFH									

NOTES:
1. Bit 0 in the WMOD register must be set to logic "0".
2. The carry flag can be read or written by specific bit manipulation instructions only.

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. You can use this section as a quick-reference source when writing application programs.

Figure 4-1 describes features of the register description format. Register descriptions are arranged in alphabetical order. 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 each of these registers is included in Part II of this manual, "Hardware Descriptions," in the context of the corresponding peripheral hardware module descriptions.

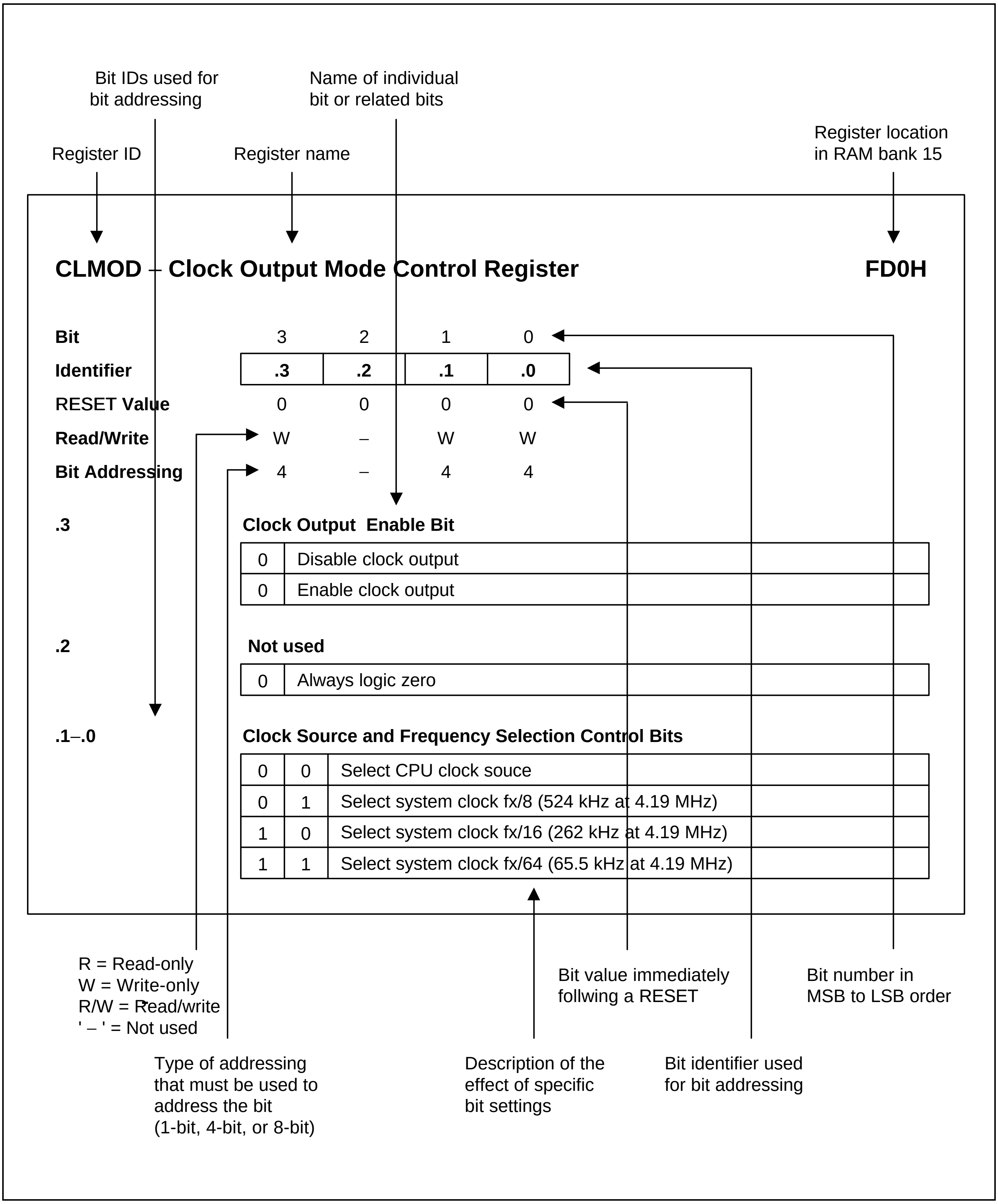


Figure 4-1. Register Description Format

BMOD — Basic Timer Mode Register 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

.3 **Basic Timer Restart Bit**

1	Restart basic timer, then clear IRQB flag, BCNT and BMOD.3 to "0"

.2–.0 **Input Clock Frequency and Signal Stabilization Interval Control Bits**

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

NOTES:
1. Signal stabilization interval is the time required to stabilize clock signal oscillation after stop mode is released by an interrupt.
2. When a RESET occurs, the oscillation stabilization time is 31.3 ms (at 4.19 MHz).
3. 'fx' is the system clock rate, assuming a clock frequency of 4.19 MHz.

CLMOD — Clock Output Mode Register
FD0H

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

.3 **Clock Output Enable Bit**

0	Disable clock output
1	Enable clock output

.2 **Clock Divide-By Frequency Selection Bit**

0	Non-divided clock
1	Divided by three clock

.1 and .0 **Clock Source and Frequency Selection Bits**

0	0	Select CPU clock source fx/4, fx/8, or fx/64 (1.05 MHz, 524 kHz, or 65.6 kHz) as determined by PCON.1-PCON.0
0	1	Select system clock fx/8 (524 kHz)
1	0	Select system clock fx/16 (262 kHz)
1	1	Select system clock fx/64 (65.5 kHz)

NOTE: fx' is the system clock, assuming a clock frequency of 4.19 MHz.

CMOD — Comparator Mode Register FD7H, FD6H

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

.7 **Comparator Enable Bit**

0	Disable comparator operation
1	Enable comparator operation

.6 **Conversion Time Control Bit**

0	4×2^7 / fx, 121.6 µs at 4.19 MHz
1	4×2^4 / fx, 15.2 µs at 4.19 MHz

.5 **External/Internal Reference Selection Bit**

0	Internal reference; CIN0-CIN3 set to analog input
1	External reference at CIN3; CIN0-CIN2 set to analog input

.4 **Not used**

0	Always logic zero

.3–.0 **Reference Voltage (V_{REF}) Selection Bits**

0	0	0	0	'n' = 0
0	0	0	1	'n' = 1
•	•	•		
1	1	1	1	'n' = 15

$V_{REF} = V_{DD} \times \frac{(n + 0.7)}{16}$, where, 'n' = 0 to 15

IE1, IRQ1 — INT1 Interrupt Enable/Request Flags FBEH

Bit	3	2	1	0
Identifier	**IE1**	**IRQ1**	**"0"**	**"0"**
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 automatically set and then cleared by hardware whenever a rising or falling edge is detected at the INT1 pin.)

.1 and .0 **Not used**

0	Always logic zero

IEK, IRQK — Key Interrupt Enable/Request Flags FBFH

Bit	3	2	1	0
Identifier	**"0"**	**"0"**	**IEK**	**IRQK**
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 and .2 **Not used**

0	Always logic zero

IEK **Key Interrupt Request Enable Flag**

0	Disable INTK interrupt requests at the KS0-KS3 pins
1	Enable INTK interrupt requests at the KS0-KS3 pins

IRQK **Key Interrupt Request Flag**

–	Generate INTK interrupt. (This bit is set whenever a falling edge is detected at any at of the KS0-KS3 pins. INTK is a quasi-interrupt and IRQK must be cleared by software.)

IEB, IRQB — INTB Interrupt Enable/Request Flags FB8H

Bit	3	2	1	0
Identifier	**“0”**	**“0”**	**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

.3 and .2 — **Not used**

0	Always logic zero

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 whenever a reference interval signal is received from the basic timer.)

IES, IRQS — INTS Interrupt Enable/Request Flags FBDH

Bit	3	2	1	0
Identifier	"0"	"0"	IES	IRQS
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 and .2 **Not used**

0	Always logic zero

IES **INTS Interrupt Enable Flag**

0	Disable INTS interrupt requests
1	Enable INTS interrupt requests

IRQS **INTS Interrupt Request Flag**

–	Generate INTS interrupt. (This bit is set and cleared automatically by hardware whenever a transmit or receive operation is completed.)

IET0, IRQT0 — INTT0 Interrupt Enable/Request Flags 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 and .2 **Not used**

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 whenever the contents of theTCNT0 and TREF0 registers match.)

IEW, IRQW — INTW Interrupt Enable/Request Flags 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 and .2 — **Not used**

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 (bit is set when timer interval = 0.5 s or 3.19 ms)

NOTE: INTW is a quasi-interrupt and its request flag must be cleared by software.

IMOD1 — External Interrupt 1 (INT1) Mode Register FB5H

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

.3 –.1 **Not used**

0	Always logic zero

.0 **INT1 Edge Detection Control Bit**

0	Rising edge detection
1	Falling edge detection

IMODK — Key Interrupt Mode Register FB6H

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

.3 **Key Interrupt Enable Bit**

0	Disable key interrupt request
1	Enable key interrupt request

.2 **Not used**

0	Always logic zero

.1 and .0 **Key Interrupt Edge Detection Control Bits**

0	0	Trigger key interrupt request at KS0-KS3 on falling edge
0	1	Trigger key interrupt request at KS0-KS1 on falling edge
1	0	Trigger key interrupt request at KS2-KS3 on falling edge
1	1	Trigger key interrupt request at KS0-KS2 on falling edge

IPR — Interrupt Priority Register 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

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

0	0	0	Process all interrupt requests at low priority
0	0	1	Process INTB interrupt only
0	1	0	Process INT0 interrupts only
0	1	1	Process INT1 interrupts only
1	0	0	Process INTS interrupts only
1	0	1	Process INTT0 interrupts only

PCON — Clock Power Control Register 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

.3 and .2 **CPU Operating Mode Control Bits**

0	0	Enable normal CPU operating mode
0	1	Enter Idle power-down mode
1	0	Enter Stop power-down mode

.1 and .0 **CPU Clock Frequency Selection Bits**

0	0	CPU clock = fx/64
1	0	CPU clock = fx/8
1	1	CPU clock = fx/4

NOTE: 'fx' is the system clock frequency.

PMG1 — Port I/O Mode Flags (Group 1: Ports 0, 1) FE9H, FE8H

Bit	3	2	1	0	3	2	1	0
Identifier	**PM1.3**	**PM1.2**	**PM1.1**	**PM1.0**	**"0"**	**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 Mode Selection Bit**

0	Input mode
1	Output mode

PM1.2 **P1.2 Mode Selection Bit**

0	Input mode
1	Output mode

PM1.1 **P1.1 Mode Selection Bit**

0	Input mode
1	Output mode

PM1.0 **P1.0 Mode Selection Bit**

0	Input mode
1	Output mode

.3 **Not used**

0	Always logic zero

PM0.2 **P0.2 Mode Selection Bit**

0	Input mode
1	Output mode

PM0.1 **P0.1 Mode Selection Bit**

0	Input mode
1	Output mode

PM0.0 **P0.0 Mode Selection Bit**

0	Input mode
1	Output mode

PMG2 — Port I/O Mode Flags (Group 2: Ports 2, 3) FEBH, FEAH

Bit	3	2	1	0	3	2	1	0
Identifier	**PM3.3**	**PM3.2**	**PM3.1**	**PM3.0**	**PM2.3**	**PM2.2**	**PM2.1**	**PM2.0**
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

PM3.3 **P3.3 Mode Selection Bit**

0	Input mode
1	Output mode

PM3.2 **P3.2 Mode Selection Bit**

0	Input mode
1	Output mode

PM3.1 **P3.1 Mode Selection Bit**

0	Input mode
1	Output mode

PM3.0 **P3.0 Mode Selection Bit**

0	Input mode
1	Output mode

PM2.3 **P2.3 Mode Selection Bit**

0	Input mode
1	Output mode

PM2.2 **P2.2 Mode Selection Bit**

0	Input mode
1	Output mode

PM2.1 **P2.1 Mode Selection Bit**

0	Input mode
1	Output mode

PM2.0 **P2.0 Mode Selection Bit**

0	Input mode
1	Output mode

PNE — Port Open-Drain Enable Register FDBH, FDAH

Bit	3	2	1	0	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

.7 **P3.3 Open-Drain Output Enable**

0	Push-pull output
1	Open-drain output

.6 **P3.2 Open-Drain Output Enable**

0	Push-pull output
1	Open-drain output

.5 **P3.1 Open-Drain Output Enable**

0	Push-pull output
1	Open-drain output

.4 **P3.0 Open-Drain Output Enable**

0	Push-pull output
1	Open-drain output

.3 **P2.3 Open-Drain Output Enable**

0	Push-pull output
1	Open-drain output

.2 **P2.2 Open-Drain Output Enable**

0	Push-pull output
1	Open-drain output

.1 **P2.1 Open-Drain Output Enable**

0	Push-pull output
1	Open-drain output

.0 **P2.0 Open-Drain Output Enable**

0	Push-pull output
1	Open-drain output

PSW — Program Status Word FB1H, FB0H

Bit	3	2	1	0	3	2	1	0
Identifier	**C**	**SC2**	**SC1**	**SC0**	**IS1**	**IS0**	**EMB**	**ERB**
RESET Value	(Note 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	(Note 2)	8	8	8	1/4	1/4	1	1

C — **Carry Flag**

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

SC2–SC0 — **Skip Condition Flags (Read-Only)**

0	No skip condition exists (no direct manipulation of these bits is allowed)
1	Skip condition exists (no direct manipulation of these bits is allowed)

IS1 and IS0 — **Interrupt Status Flags**

0	0	Service all interrupt requests
0	1	Service only the high-priority interrupt(s) as determined by current interrupt priority register (IPR) settings
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, "Addressing Modes," for detailed information.

P1MOD — Port 1 Mode Register FE2H

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

.3 **P1.3 Analog or Digital Input Selection Bit**

0	Digital input
1	Analog input

.2 **P1.2Analog or Digital Input Selection Bit**

0	Digital input
1	Analog input

.1 **P1.1 Analog or Digital Input Selection Bit**

0	Digital input
1	Analog input

.0 **P1.0 Analog or Digital Input Selection Bit**

0	Digital input
1	Analog input

PUMOD1— Pull-Up Resistor Mode Register (Ports 0 and 1) FDDH, FDCH

Bit	3	2	1	0	3	2	1	0
Identifier	**.7**	**.6**	**.5**	**.4**	**"0"**	**.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

.7 **P1.3 Pull-Up Resistor Enable Bit**

0	Disable pull-up
1	Enable pull-up

.6 **P1.2 Pull-Up Resistor Enable Bit**

0	Disable pull-up
1	Enable pull-up

.5 **P1.1 Pull-Up Resistor Enable Bit**

0	Disable pull-up
1	Enable pull-up

.4 **P1.0 Pull-Up Resistor Enable Bit**

0	Disable pull-up
1	Enable pull-up

.3 **Not used**

0	Always logic zero

.2 **P0.2 Pull-Up Resistor Enable Bit**

0	Disable pull-up
1	Enable pull-up

.1 **P0.1 Pull-Up Resistor Enable Bit**

0	Disable pull-up
1	Enable pull-up

.0 **P0.0 Pull-Up Resistor Enable Bit**

0	Disable pull-up
1	Enable pull-up

PUMOD2 — Pull-Up Resistor Mode Register (Ports 2 and 3) FDDH, FDCH

Bit	3	2	1	0	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

.7 **P3.3 Pull-Up Resistor Enable Bit**

0	Disable pull-up
1	Enable pull-up

.6 **P3.2 Pull-Up Resistor Enable Bit**

0	Disable pull-up
1	Enable pull-up

.5 **P3.1 Pull-Up Resistor Enable Bit**

0	Disable pull-up
1	Enable pull-up

.4 **P3.0 Pull-Up Resistor Enable Bit**

0	Disable pull-up
1	Enable pull-up

.3 **P2.3 Pull-Up Resistor Enable Bit**

0	Disable pull-up
1	Enable pull-up

.2 **P2.2 Pull-Up Resistor Enable Bit**

0	Disable pull-up
1	Enable pull-up

.1 **P2.1 Pull-Up Resistor Enable Bit**

0	Disable pull-up
1	Enable pull-up

.0 **P2.0 Pull-Up Resistor Enable Bit**

0	Disable pull-up
1	Enable pull-up

SMOD — Serial I/O Mode Register FE1H, FE0H

Bit	3	2	1	0	3	2	1	0
Identifier	**.7**	**.6**	**.5**	**“0”**	**.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	1/8	8	8	8

.7–.5 **Serial I/O Clock Selection and SBUF Read/Write Status Control Bits**

0	0	0	Use an external clock at the SCK pin; Enable SBUF when SIO operation is halted or when SCK goes high
0	0	1	Use the TOL0 clock from timer/counter 0; Enable SBUF when SIO operation is halted or when SCK goes high
0	1	x	Use the selected CPU clock (fx/4, 8, or 64; 'fx' is the system clock) then, enable SBUF read/write operation. 'x' means 'don't care.'
1	0	0	4.09-kHz clock ($fx/2^{10}$)
1	1	1	262-kHz clock ($fx/2^{4}$); Note: You cannot select a $fx/2^{4}$ clock frequency if you have selected a CPU clock of fx/64

.4 **Not used**

0	Always logic zero

.3 **Serial I/O Start Bit**

1	Clear IRQS flag and 3-bit clock counter and start serial transmission. When the SIO transmission starts, this bit is automatically cleared by hardware.

.2 **SIO Data Shifter and Clock Counter Enable Bit**

0	Disable the data shifter and clock counter; retain contents of IRQS flag when serial transmission is complete
1	Enable the data shifter and clock counter; The IRQS flag is set to logic one when serial transmission is complete

.1 **Serial I/O Transmission Mode Selection Bit**

0	Receive-only mode
1	Transmit-and-receive mode

.0 **LSB/MSB Transmission Mode Selection Bit**

0	Transmit the most significant bit (MSB) first
1	Transmit the least significant bit (LSB) first

NOTE: All kHz frequency ratings assume a system clock of 4.19 MHz.

TMOD0 — Timer/Counter 0 Mode Register F91H, F90H

Bit	3	2	1	0	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

.7 **Not used**

0	Always logic zero

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

0	0	0	External clock input at TCL0 pin on rising edges
0	0	1	External clock input at TCL0 pin on falling edges
1	0	0	Internal system clock (fx) of 4.19 MHz / 2^{10} (4.09 kHz)
1	0	1	Selected clock: fx/2^{6} (65.5 kHz)
1	1	0	Selected clock: fx/2^{4} (262 kHz)
1	1	1	Selected clock: fx (4.19 MHz)

.3 **Counter Clear Bit**

1	Clear TCNT0, IRQT0, and TOL0 and resume counting immediately. (This bit is cleared automatically to "0" when counting starts.)

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

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

.1 **Not used**

0	Always logic zero

.0 **Not used**

0	Always logic zero

NOTE: System clock frequency (fx) is assumed to be 4.19 MHz.

TOE0 — Timer Output Enable Flag F92H

Bit	3	2	1	0
Identifier	**"0"**	**TOE0**	**"0"**	**"0"**
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 **Not used**

0	Always logic zero

TOE0 **Timer/Counter 0 Output Enable Flag**

0	Disable output to the TCLO0 pin
1	Enable output to the TCLO0 pin

.1 **Not used**

0	Always logic zero

.0 **Not used**

0	Always logic zero

WMOD — Watch Timer Mode Register F89H, F88H

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

.7 **Buzzer Output Enable Bit**

0	Disable buzzer (BUZ) signal output
1	Enable buzzer (BUZ) signal output

.6 **Not used**

0	Always logic zero

.5 and .4 **Output Buzzer Frequency Selection Bits**

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

.3 **Not used**

0	Always logic zero

.2 **Watch Timer Enable Bit**

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

.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

.0 **Not used**

0	Always logic zero

6 OSCILLATOR CIRCUIT

OVERVIEW

The S3C7031/7032 has an on-chip system clock circuit. The CPU and peripheral hardware operate on the system clock frequency. Specifically, a clock is required by the following peripheral modules:

— Basic timer
— Timer/counter 0
— Watch timer
— Serial I/O interface
— Clock output circuit

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

$$\frac{fx}{4}, \frac{fx}{8}, \frac{fx}{64}$$

When the PCON register is cleared to zero after RESET, the normal CPU operating mode is enabled, a system clock of fx/64 is selected.

Bits 3 and 2 of the PCON register can be manipulated by a STOP or IDLE instruction to engage Stop or Idle power-down mode.

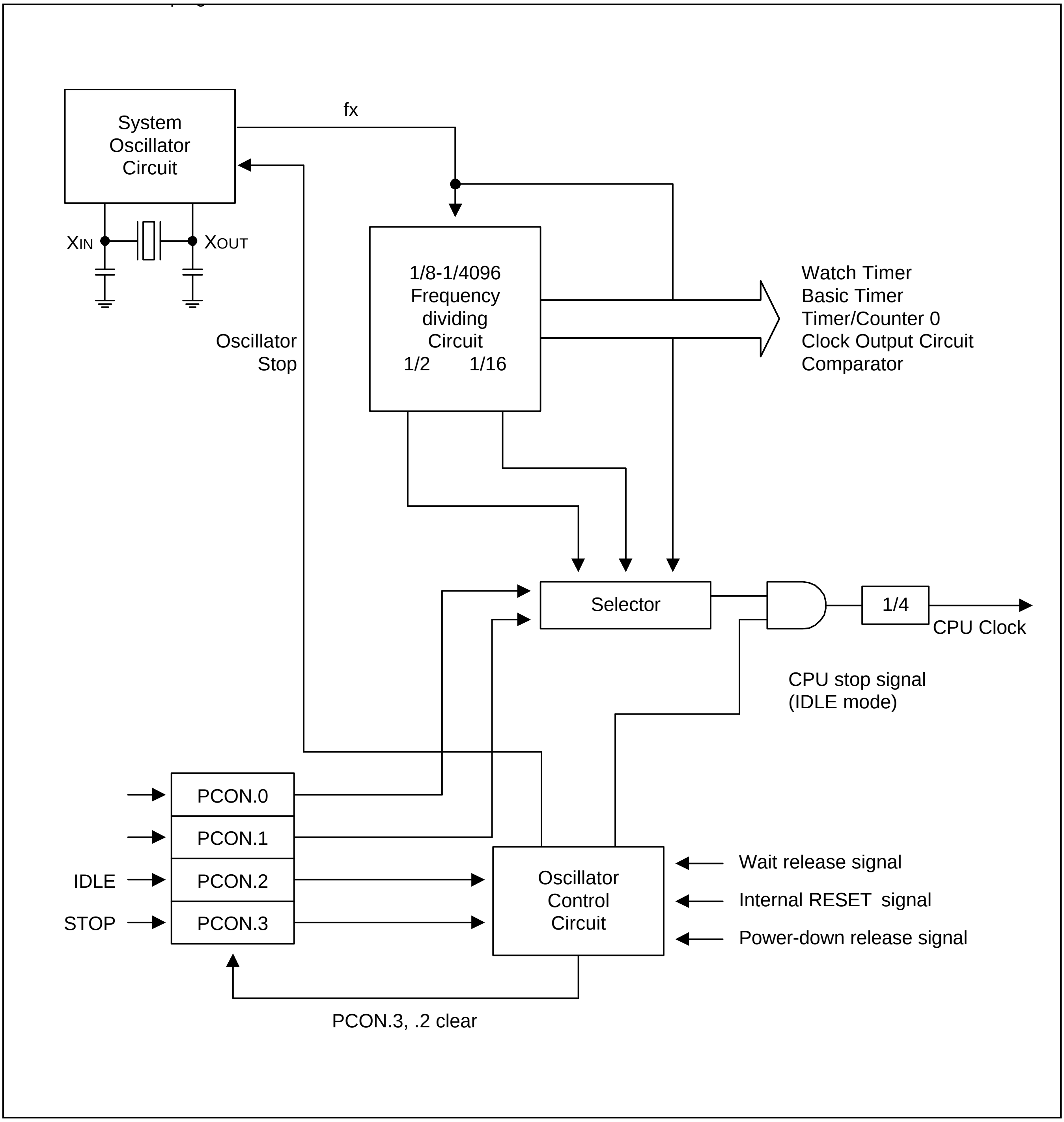


Figure 6-1. Clock Circuit Diagram

SYSTEM OSCILLATOR CIRCUIT TYPES

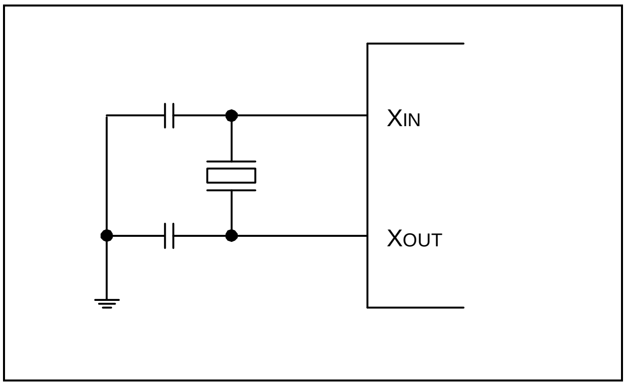


Figure 6-2. Crystal/Ceramic Oscillator

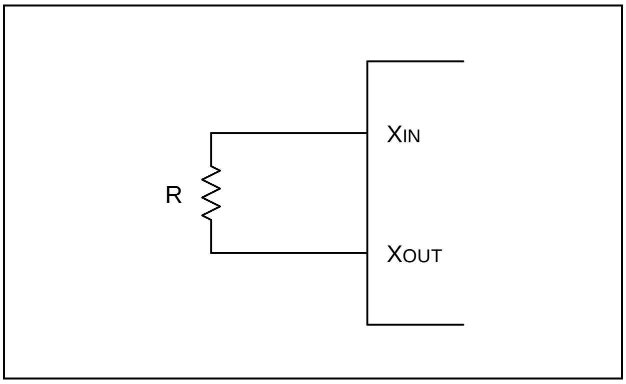


Figure 6-4. RC Oscillator

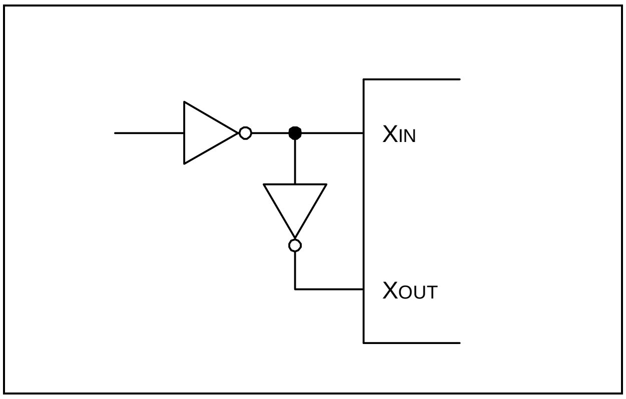


Figure 6-3. External Clock

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 is mapped to RAM address FB3H and can be addressed directly by 4-bit write instructions or by the instructions IDLE and STOP.

PCON bits 3 and 2 are controlled by the STOP and IDLE instructions that initiate the Idle and Stop power-down modes. Idle and Stop mode can be initiated by these instructions at any time, regardless of the current value of the enable memory bank flag (EMB). PCON bits 1 and 0 are used to select a specific system clock frequency.

RESET sets PCON register values to logic zero. PCON.1 and PCON.0 divide the frequency (fx) by 64, 8, or 4. 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	
0	0	fx/64
1	0	fx/8
1	1	fx/4

☞ PROGRAMMING TIP — Setting the CPU Clock

To set the CPU clock to 0.95µs 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 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

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 mapped to RAM address FD0H and is addressable by 4-bit write instructions only.

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, fx/8, fx/16, or fx/64. The clock selected by CLMOD.2 can be output after being divided by three.

Table 6-3. Clock Output Mode Register (CLMOD) Organization

CLMOD Bit Settings			Resulting Clock Output		
CLMOD.2	CLMOD.1	CLMOD.0	Clock Source	Frequency	Duty
0	0	0	CPU clock (fx/4, fx/8, fx/64)	1.05 MHz, 524 kHz, 65.5 kHz	1/2
	0	1	fx/8	524 kHz	
	1	0	fx/16	262 kHz	
	1	1	fx/64	65.5 kHz	
1	0	0	CPU clock/3 (fx/12, fx/24, fx/192)	350MHz, 175kHz, 21.8 kHz	1/3
	0	1	fx/24	175 kHz	
	1	0	fx/48	87.4 kHz	
	1	1	fx/192	21.8 kHz	

CLMOD.3	Result of CLMOD.3 Setting
0	Clock output is disabled
1	Clock output is enabled

NOTE: All fequencies assume that fx = 4.19 MHz.

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
— Output latch
— Port mode flag
— CLO output pin (P0.0)

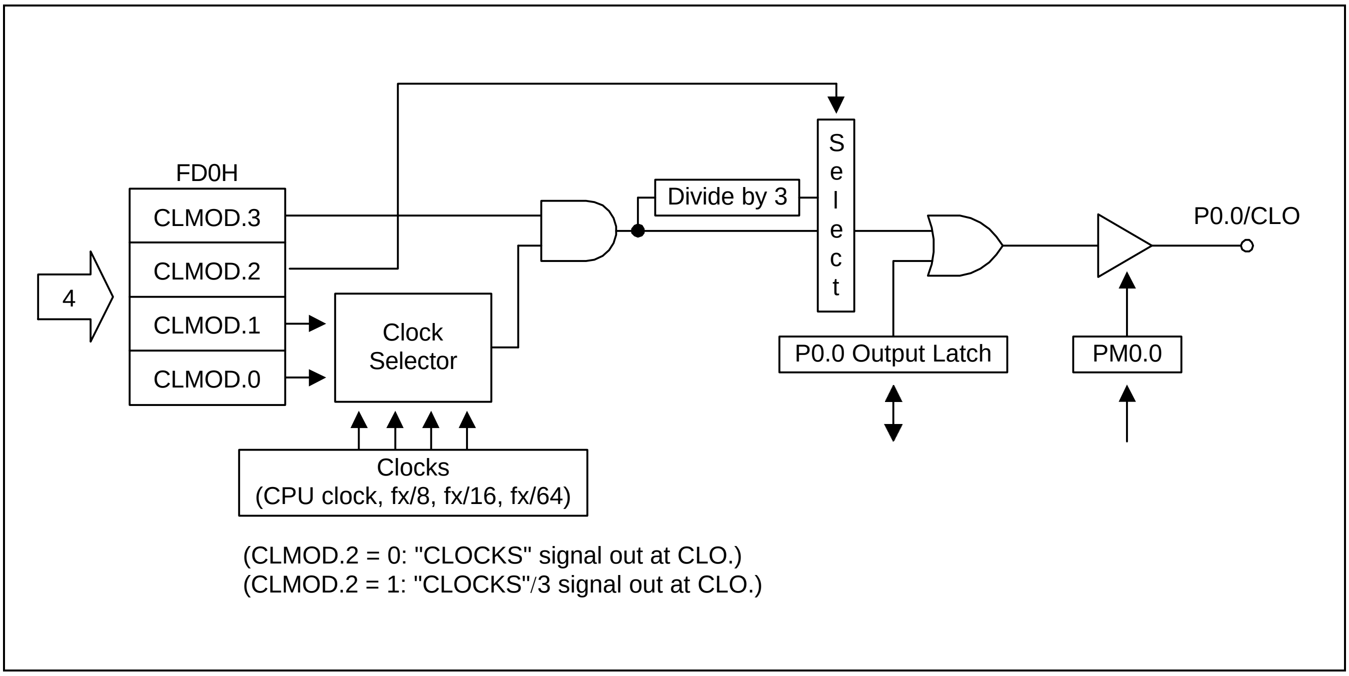


Figure 6-5. CLO Output Pin Circuit Diagram

CLOCK OUTPUT PROCEDURE

To output clock pulses to the CLO pin, follow this general procedure:

1. Disable clock output by clearing CLMOD.3 to logic zero.
2. Set the clock output frequency (CLMOD.1, CLMOD.0, CLMOD.2).
3. Load a "0" to the output latch of the CLO pin (P0.0).
4. Set the P0.0 mode flag (PM0.0) to output mode.
5. 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             ; or BITR EMB
SMB     15
LD      EA,#07H
LD      PMG1,EA         ; Port 0 ← Output mode
BITR    P0.0            ; Clear P0.0 output latch
LD      A,#9H
LD      CLMOD,A
```

7 INTERRUPTS

OVERVIEW

S3C7031/7032 microcontrollers process three types of interrupts:

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

Table 7-1. Interrupts and Corresponding I/O Pin(s)

Interrupt Type	Interrupt Name	I/O Port Pin(s)
External Interrupt	INT1	P0.2
Internal Interrupts	INTB, INTT0, INTS	Not applicable
Quasi-interrupts	INTK	P1.0-P1.3 (KS0-KS3)
	INTW	Not applicable

The interrupt control circuit has four functional components:

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

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, are enabled (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, 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 and register banks 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. Enable flag values are saved during the main routine, as well as during service routines. Any changes you make to enable flag values during a service routine are not stored in the vector address.

When an interrupt occurs, the enable flag values before the interrupt is initiated are saved along with the program status word (PSW), and the enable flag values for the interrupt is fetched from the respective vector address. Then, if required, 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, however, 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 value in the IRQx flag is set, it is retained until all other conditions for the interrupt have been met, and the service routine can be initiated.

Multiple Interrupts

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

Power-Down Mode Release

An interrupt (with the exception of INT0) 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".

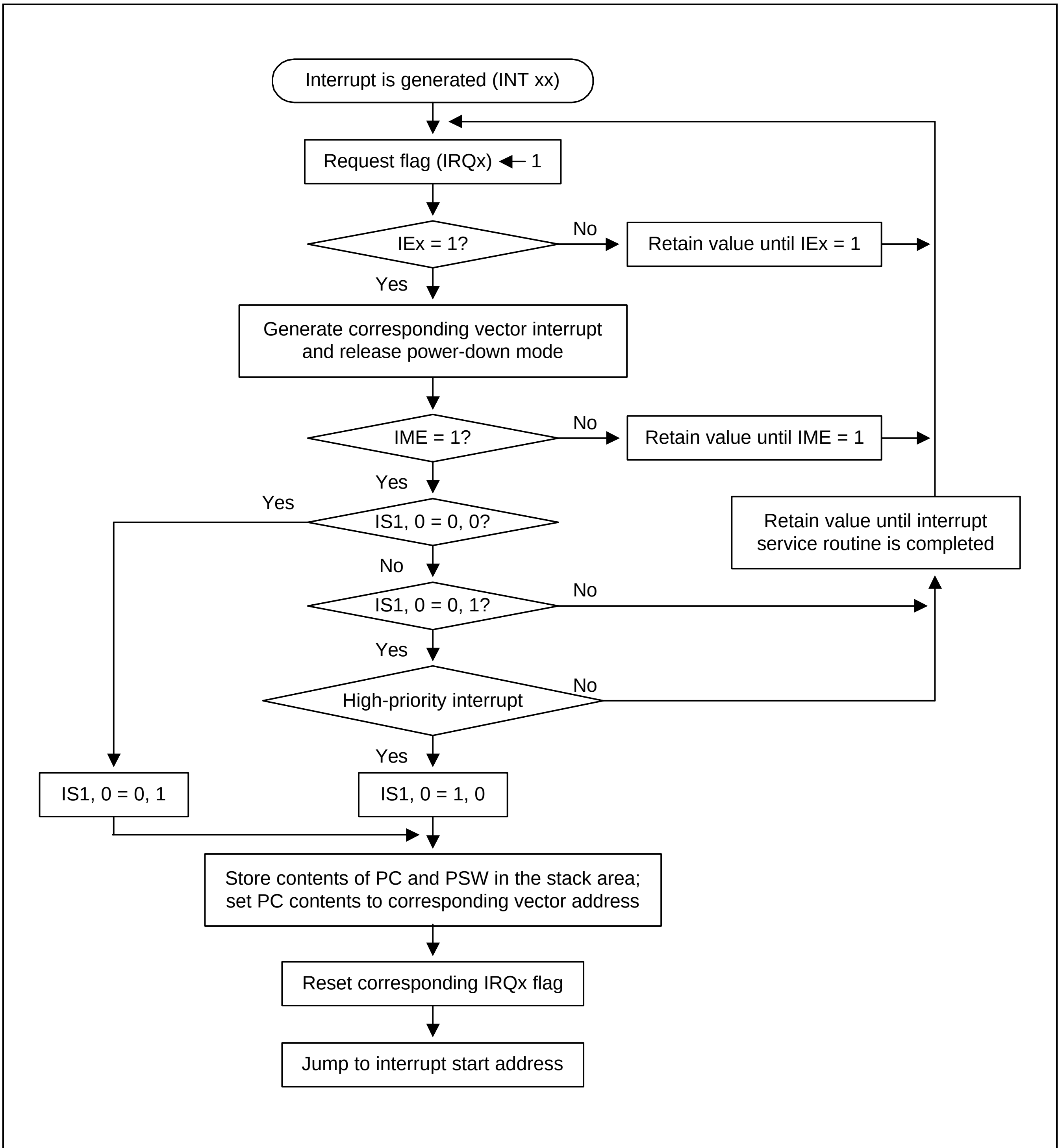


Figure 7-1. Interrupt Execution Flowchart

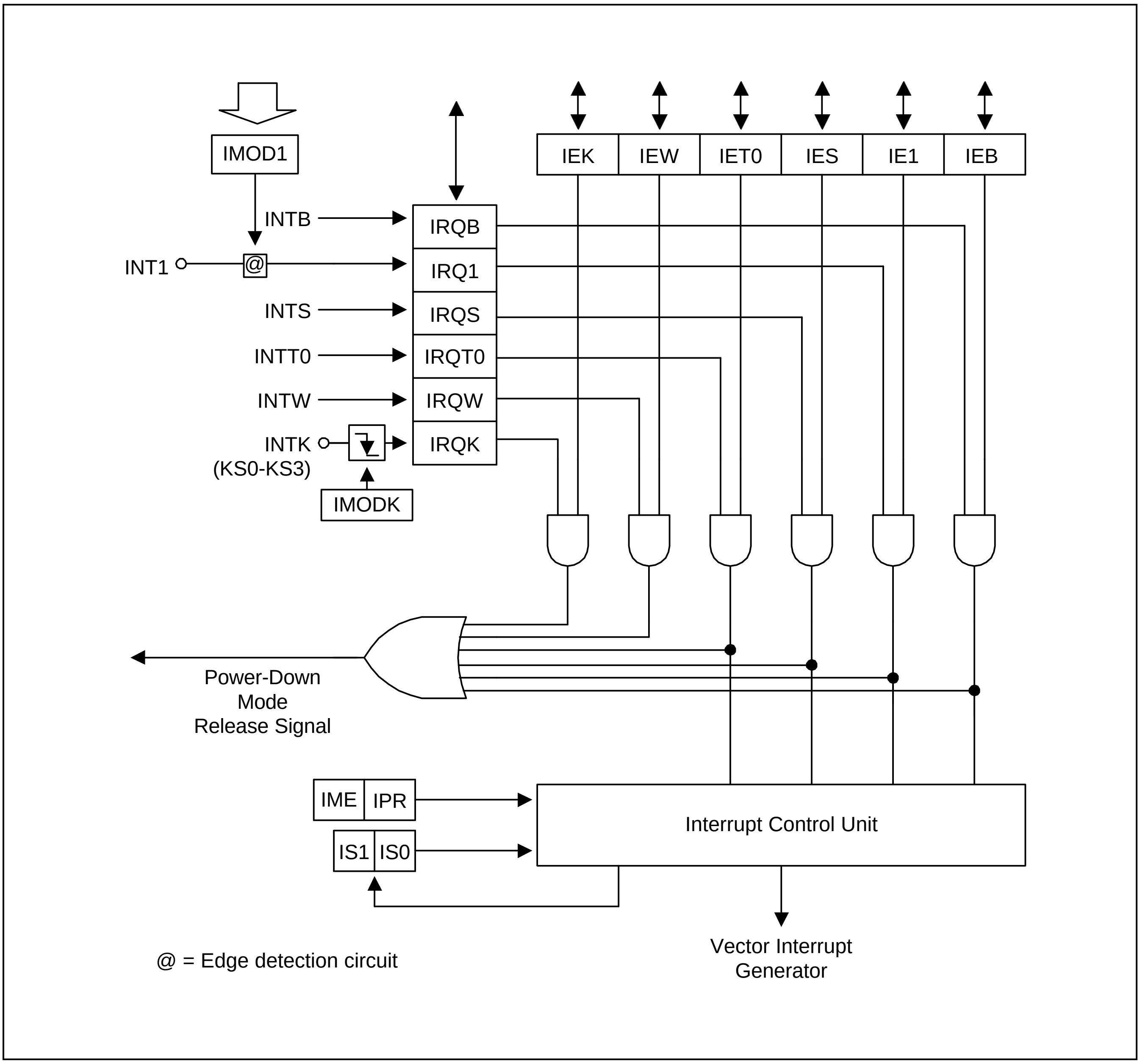


Figure 7-2. Interrupt Control Circuit Diagram

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 ("0" → "1" or "1" → "0"), 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 you set IS1 to "0" and IS0 to "1", you inhibit all interrupt service routines except for the highest priority interrupt currently defined by the interrupt priority register (IPR).

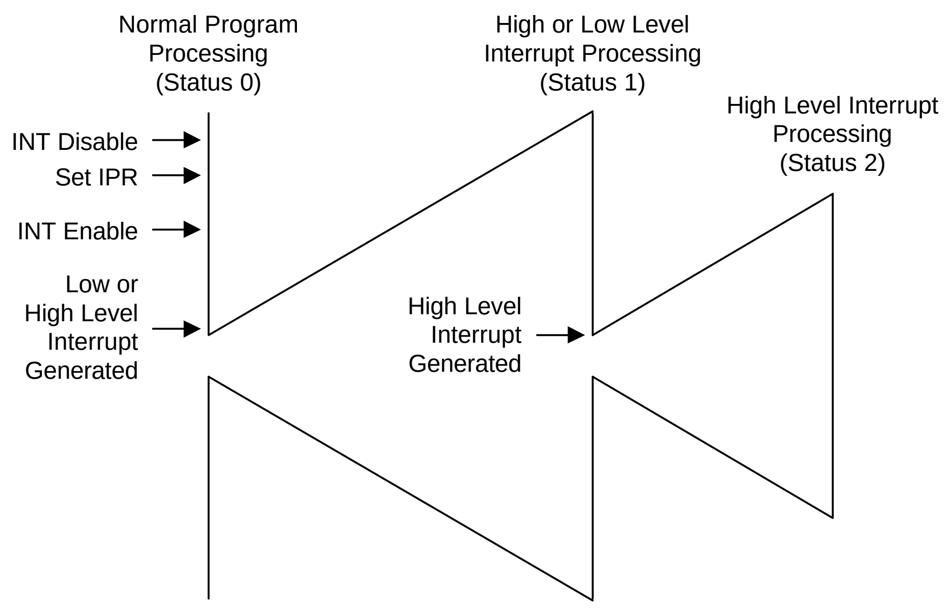


Figure 7-3. Two-Level Interrupt Handling

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 Figure 7-4.

When an interrupt is requested during normal program execution, the interrupt status flags IS0 and IS1 are set to "0" and "1", 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-priority and low-priority requests will be serviced in parallel.

Table 7-2. IS1 and IS0 Function

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	–	–

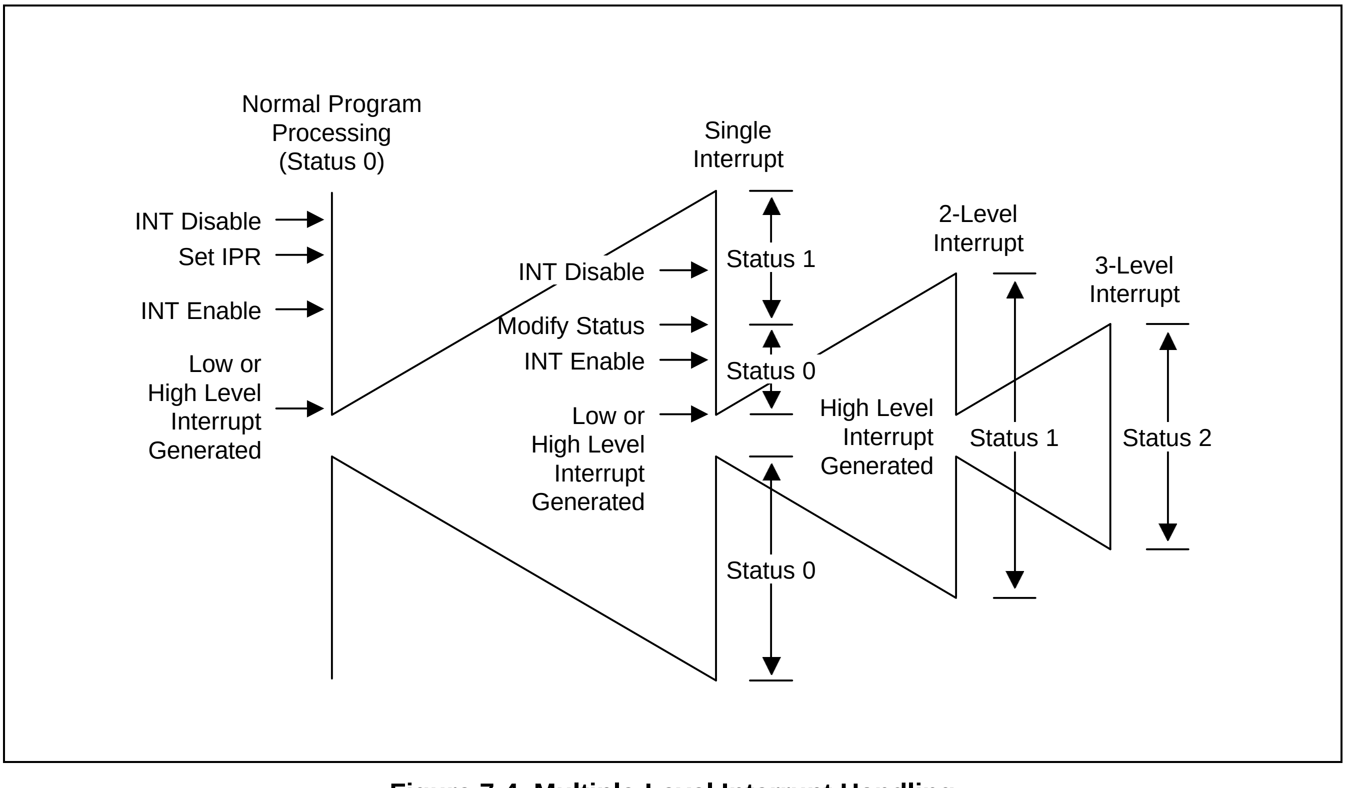


Figure 7-4. Multiple-Level Interrupt Handling

INTERRUPT PRIORITY REGISTER (IPR)

The 4-bit interrupt priority register (IPR) is used to control multi-level interrupt handling. The IPR is mapped to RAM address FB2H, 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	1
INT1	2
INTS	3
INTT0	4

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 is mapped to FB2H.3 and 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.
0	0	1	Process INTB interrupt only.
0	1	1	Process INT1 interrupts only.
1	0	0	Process INTS interrupts only.
1	0	1	Process INTT0 interrupts only.

NOTE: When all interrupts are low priority (the lower three bits of the IPR register are logic zero), the interrupt generated first will become high priority. Therefore, the first generated 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 7.4 (e.g., 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 interrupt service routine is started.

☞ PROGRAMMING TIP — Setting the INT Interrupt Priority

Set 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 MODE REGISTER (IMOD1)

The following components are used to process external interrupts at the INT1 pin:

— Edge detection circuit

— One mode register, IMOD1

The mode register is used to control the triggering edge of the input signal. IMOD settings let you choose either the rising or falling edge of the incoming signal at the INT1 pins as the interrupt request trigger.

IMOD1 bit is mapped to RAM addresses FB5H (IMOD1), and are addressable by 4-bit write instructions. RESET clears all IMOD values to logic zero, selecting rising edges as the trigger for incoming interrupt requests.

Table 7-5. IMOD1 Register Organization

IMOD1	0	0	0	IMOD1.0	Effect of IMOD1 Settings
				0	Rising edge detection
				1	Falling edge detection

EXTERNAL INTERRUPT1 MODE REGISTERS (Continued)

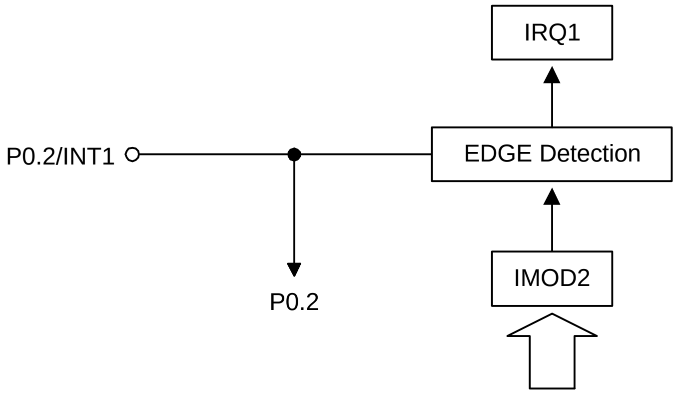


Figure 7-5. Circuit Diagram for INT1 Pins

When modifying the 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 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.

NOTE

Because the INT1 pin can be used as an output pin (P0.2), if the pin is configured to output mode, IRQ1 will be set to "1" by the edge of the output signal .

KEY INTERRUPT MODE REGISTER (IMODK)

The mode register for external interrupts at the KS0-KS3 pins, IMODK, is a 4-bit register at RAM address FB6H. IMODK is addressable only by 4-bit write instructions. RESET clears all IMODK bits to logic zero.

When bits in the IMODK register are set to logic one, INTK uses the falling edge of an incoming signal at corresponding pins as the interrupt request trigger. When a falling edge is detected at any one of the KS0-KS3 pins, the IRQK flag is set to logic one and a release signal for power-down mode is generated.

Table 7-6. IMODK Register Bit Settings

IMODK	IMODK.3	IMODK.1	IMODK.0	Effect of IMODK Settings
	0			Interrupt request is disabled
	1			Interrupt request is enabled
		0	0	Select falling edge at KS0-KS3
		0	1	Select falling edge at KS0-KS1
		1	0	Select falling edge at KS2-KS3
		1	1	Select falling edge at KS0-KS2

NOTE: Refer to the port 1 circuit diagram in Chapter 10.

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), and is mapped to bit address FB2H.3. It can be directly be manipulated by EI and DI instructions, regardless of the current value of the enable memory bank flag (EMB).

Interrupt Enable Flags (IEx)

IEx flags, when set to logic one, enable specific interrupt requests to be serviced. When the interrupt request flag is set to logic one, an interrupt will not be serviced until its corresponding IEx flag is also enabled.

Interrupt enable flags are mapped to the RAM address area FB8H-FBFH, and can be read, written, or tested directly by 1-bit instructions (BITS and BITR). IEx flags can be addressed directly at their specific RAM addresses, despite the current value of the enable memory bank (EMB) flag.

Interrupt Request Flags (IRQx)

Interrupt request flags, located in the RAM area FB8H-FBFH, 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 lag is set to logic one, the corresponding interrupt request is generated. The flag is then automatically cleared to logic zero by hardware when the interrupt has been serviced. Exceptions are the watch timer interrupt request flag, IRQW, and key interrupt request flag IRQK, 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 IRQK).
3. When IRQx is set to "1" by software, an interrupt is generated.

INTERRUPT MASTER ENABLE FLAG (IME)

The interrupt master enable flag, IME, inhibits or enables all interrupt processing. Therefore, even when an IRQx flag and its corresponding IEx flag is enabled, an interrupt request will not be serviced until the IME flag is set to logic one. The IME flag is the most significant bit of the 4-bit IPR register at RAM location FB2H.

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

You can manipulate the IME flag using EI and DI instructions, despite the current value of the enable memory bank (EMB) flag.

INTERRUPT ENABLE FLAGS (IEx)

Interrupt enable flags are used to control the execution of service routines for specific interrupt requests. The enable flag has priority over a request flag – even if the IRQx flag is enabled, the interrupt request will not be serviced until the corresponding IEx flag is set to logic one.

Using 1-bit or 4-bit instructions and direct addressing, you can read, write, or test IEx (and IRQx) flags despite the current enable memory bank (EMB) value. The IEx and IRQx flags are mapped to RAM area FB8H-FBFH.

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

Address	Bit 3	Bit 2	Bit 1	Bit 0
FB8H	0	0	IEB	IRQB
FBAH	0	0	IEW	IRQW
FBBH	0	0	0	0
FBCH	0	0	IET0	IRQT0
FBDH	0	0	IES	IRQS
FBEH	IE1	IRQ1	0	0
FBFH	0	0	IEK	IRQK

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

INTERRUPT REQUEST FLAGS (IRQx)

When an interrupt request flag (IRQx) is set, a software-generated interrupt is enabled for the corresponding interrupt. IRQx flags can be written by 1-bit or 4-bit RAM control instructions. IRQx flags are then cleared automatically when the interrupt has been serviced. Exceptions to the general rule are the watch timer interrupt request flag, IRQW and key interrupt request flag, IRQK; they must be cleared by software after the interrupt service routine has executed.

Table 7-8. Interrupt Request Flag Conditions and Priorities

Interrupt Source	Internal/ External	Pre-condition for IRQx Flag Setting	Interrupt Priority	IRQx Flag Name
INTB	I	Reference time interval signal from basic timer	1	IRQB
INT0	E	Rising or falling edge detected at INT0 pin	2	IRQ0
INTS	I	Completion signal for serial transmit-and-receive or receive-only operation	3	IRQS
INTT0	I	Signals for TCNT0 and TREF0 registers coincide	4	IRQT0
INTK (note)	E	A falling edge is detected at any one of the KS0-KS3 pins	–	IRQK
INTW (note)	I	Time interval of 0.5 secs or 3.19 msecs	–	IRQW

NOTE: INTK and INTW are quasi-interrupts; INTK is used only for testing incoming signals.

8 POWER-DOWN

OVERVIEW

The S3C7031/7032 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 halts 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, 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, serial I/O, timer/ counters 0, and watch timer – and on external interrupt requests, is detailed in Table 8-1.

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.

Idle or Stop modes are terminated either by a RESET, or by an interrupt which are enabled by the corresponding interrupt enable flag, IEx. When power-down mode is terminated by RESET input, 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", program execution is started immediately after the instruction which issues the request to enter power-down mode. The interrupt request flag remains set to logic one.

— If the IME flag = "1", two instructions are executed after the power-down mode release. Then, the vectored interrupt is initiated. However, when the release signal is caused by INTK or INTW, the operation is identical to the IME = 0 condition. That is, a vector interrupt is not generated.

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

Operation	Stop Mode (STOP)	Idle Mode (IDLE)
Clock oscillator	System clock oscillation Stops	CPU clock oscillation Stops (system clock oscillation continues)
Basic timer	Basic timer Stops	Basic timer operates (with IRQB set at each reference interval)
Serial interface	Operates only if external SCK input is selected as the serial I/O clock	Operates if a clock other than the CPU clock is selected as the serial I/O clock
Timer/counter 0	Operates only if TCL0 is selected as the counter clock	Timer/counter 0 operates
Comparator	Comparator operation is Stopped	Comparator operates
Watch timer	Watch timer operation is Stopped	Watch timer operates
External interrupts	INT1 and INTK are acknowledged	INT1 and INTK are acknowledged
CPU	All CPU operations are disabled	All CPU operations are disabled
Power-down mode release signal	Interrupt request signals are enabled by an interrupt enable flag or by RESET input	Interrupt request signalsare enabled by an interrupt enable flag or by RESET input

IDLE MODE TIMING DIAGRAMS

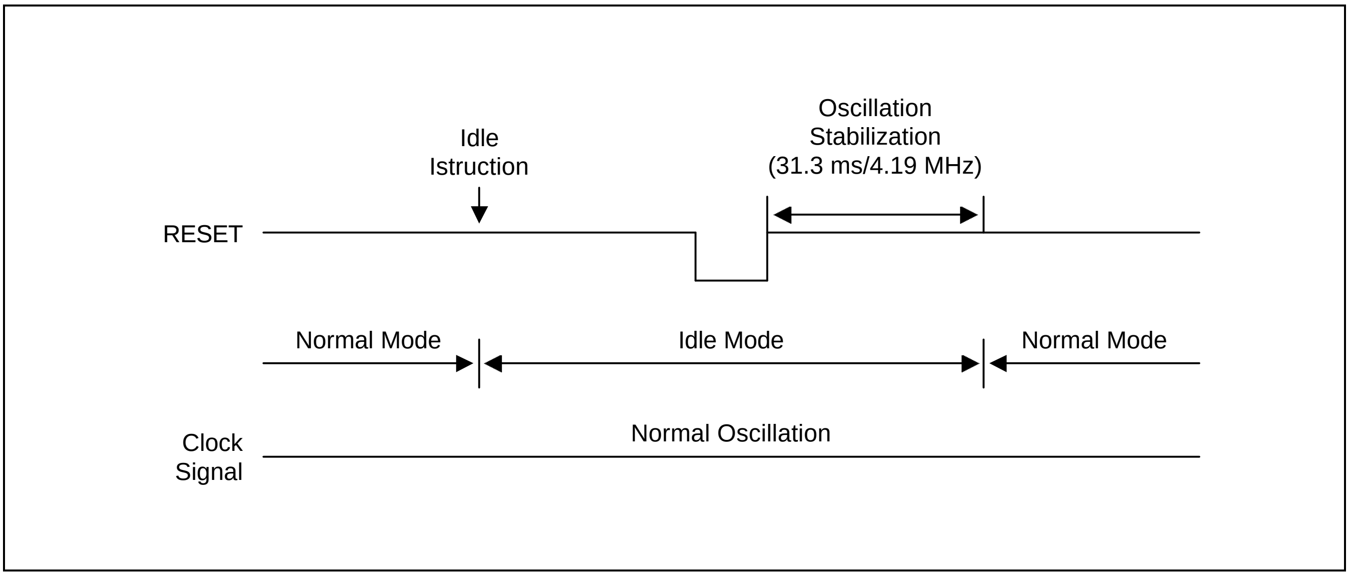


Figure 8-1. Timing When Idle Mode is Released by RESET

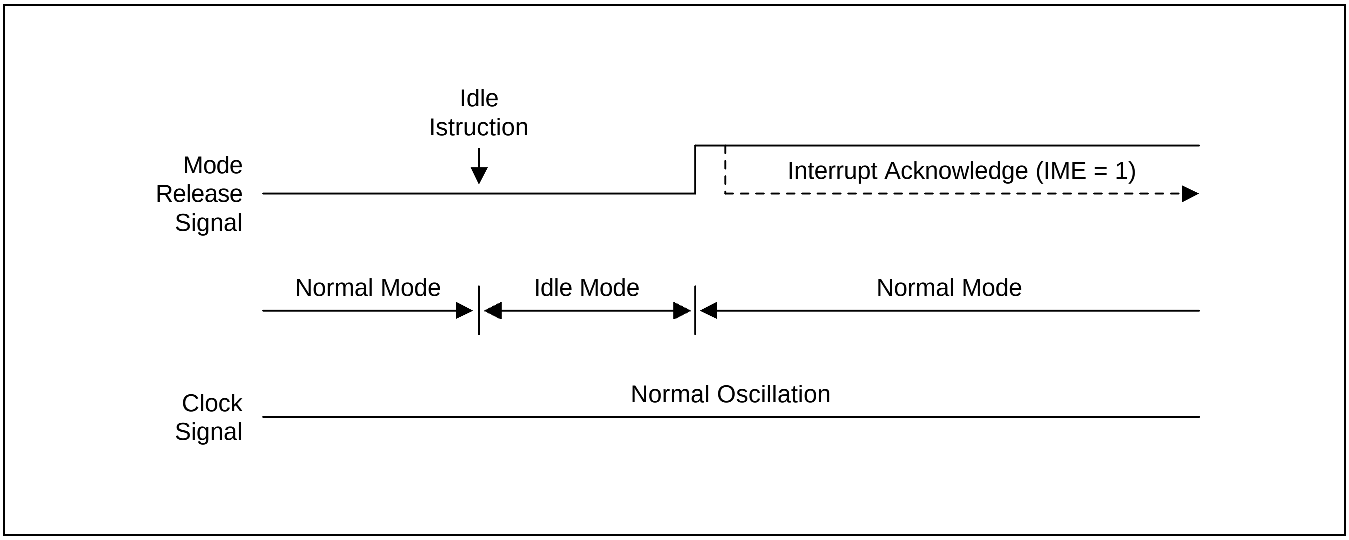


Figure 8-2. Timing When Idle Mode is Released by an Interrupt

STOP MODE TIMING DIAGRAMS

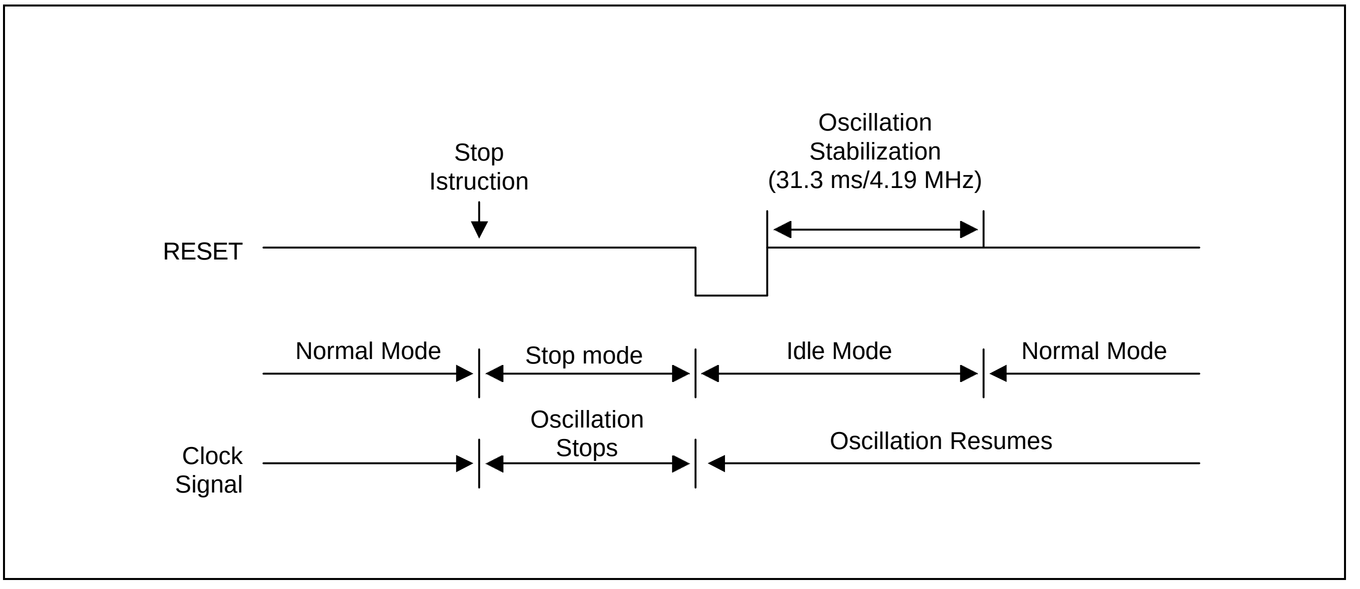


Figure 8-3. Timing When Stop Mode is Released by RESET

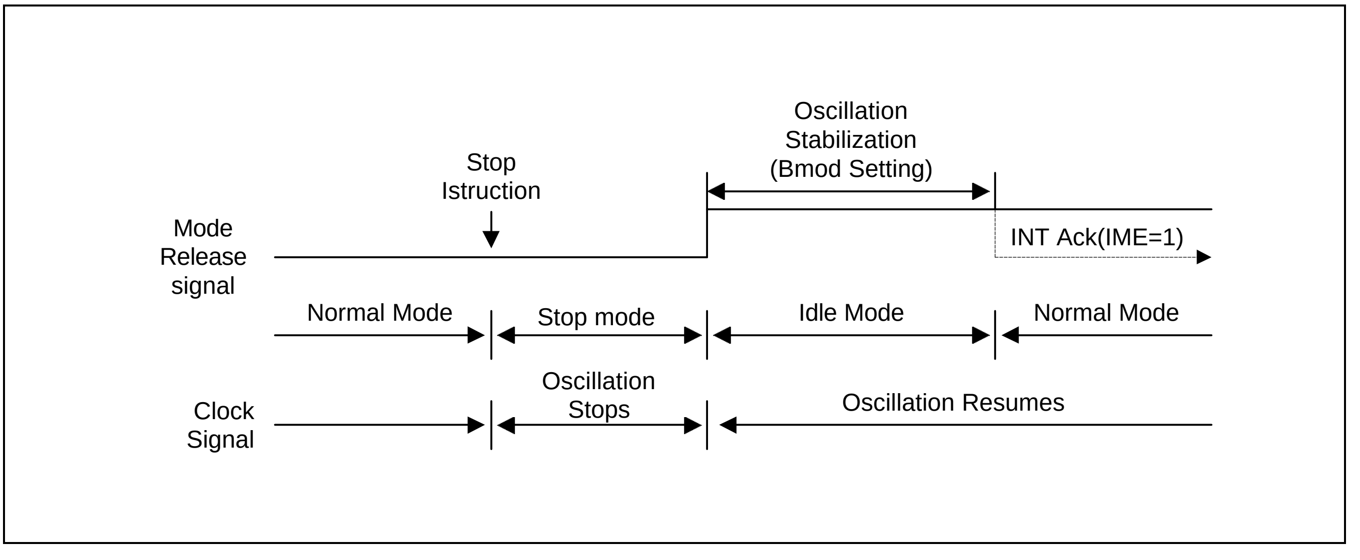


Figure 8-4. Timing When Stop Mode is Release by an Interrupt

I/O PORT PIN CONFIGURATION FOR POWER-DOWN

The following method describes how to configure I/O port pins to reduce power consumption during power-down modes (Stop, Idle):

Condition 1: If the microcontroller is not configured to an external device:

1. Connect unused port pins according to the information in Table 8-2.
2. Disable all pull-up resistors for output pins by making the appropriate modifications to the pull-up resistor mode register, PUMOD. Reason: If output goes low when the pull-up resistor is enabled, there may be unexpected surges of current through the pull-up.
3. Disable pull-up resistors for input pins configured to V_{DD} or V_{SS} levels in order to check the current input option. Reason: If the input level of a port pin is set to V_{SS} when a pull-up resistor is enabled, it will draw an unnecessarily large current.

Condition 2: If the microcontroller is configured to an external device and the external device's V_{DD} source is turned off in power-down mode.

1. Connect unused port pins according to the information in Table 8-2.
2. Disable the pull-up resistors of output pins by making the appropriate modifications to the pull-up resistor mode register, PUMOD. Reason: If output goes low when the pull-up resistor is enabled, there may be unexpected surges of current through the pull-up.
3. Disable pull-up resistors for input pins configured to V_{DD} or V_{SS} levels in order to check the current input option. Reason: If the input level of a port pin is set to V_{SS} when a pull-up resistor is enabled, it will draw an unnecessarily large current.
4. Disable the pull-up resistors of input pins connected to the external device by making the necessary modifications to the PUMOD register.
5. Configure the output pins that are connected to the external device to low level. Reason: When the external device's V_{DD} source is turned off, and if the microcontroller's output pins are set to high level, V_{DD} – 0.7 V is supplied to the V_{DD} of the external device through its input pin. This causes the device to operate at the level V_{DD} – 0.7 V. In this case, total current consumption would not be reduced.
6. Determine the correct output pin state necessary to block current pass in according with the external transistors (PNP, NPN).

RECOMMENDED CONNECTIONS FOR UNUSED PINS

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

Table 8-2. Unused Pin Connections for Reduced Power Consumption

Pin/Share Pin Names	Recommended Connection
P0.0 / CIO P0.1 / TIO P0.2 / INT1	Input mode: Connect to V_{DD} Output mode: Do not use
P1.0 /KS0/CIN0– P1.3/KS3/CIN3	Input mode: Connect to V_{DD} Output mode: Do not use
P2.0 P2.1 P2.2 P2.3	Input mode: Connect to V_{DD} Output mode: Do not connect
P3.0 / SCK P3.1 /SO P3.2 / SI P3.3 / BUZ	Input mode: Connect to V_{DD} Output mode: Do not connect

9 RESET

OVERVIEW

When a RESET signal is input during normal operation or power-down mode, a 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 – the effect on most hardware register values is almost identical. The exceptions are as follows:

— Carry flag

— Data memory values

— General-purpose registers A, E, L, H, X, W, Z, and Y

— Serial I/O buffer register (SBUF)

If a RESET occurs during Idle or Stop mode, the current values in these registers are retained. Otherwise, their values are undefined.

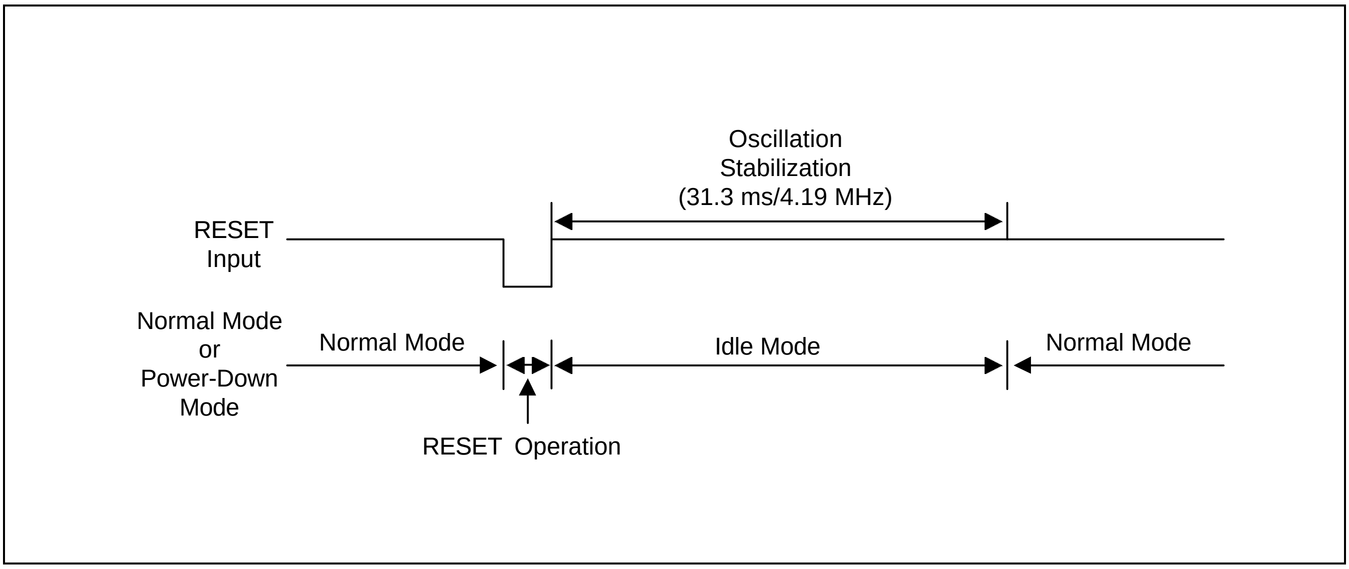


Figure 9-1. Timing for Oscillation Stabilization After RESET

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 three bits of address 0000H are transferred to PC10-8, and the contents of 0001H to PC7-0.	Lower three bits of address 0000H are transferred to PC10-8, and the contents of 0001H to PC7-0.
Bank selection registers (SMB, SRB)	0, 0	0, 0
BSC register (BSC0-BSC3)	0	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):		
Working registers A, E, L, H, X, W, Z, Y	Values retained	Undefined
General-purpose registers	Values retained	Undefined
Clock:		
Power control register (PCON)	0	0
Clock output mode register (CLMOD)	0	0
Interrupt:		
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
INT1 mode register (IMOD1)	0	0
INTK mode register (IMODK)	0	0

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 Port:		
Output buffers	Off	Off
Output latches	0	0
Port mode flags (PM)	0	0
Pull-up resistor mode reg (PUMOD)	0	0
Port 1 mode register (P1MOD)	0	0
Port open drain enable (PNE)	0	0
Basic Timer:		
Count register (BCNT)	Undefined	Undefined
Mode register (BMOD)	0	0
Timer/Counter 0:		
Count registers (TCNT0)	0	0
Reference registers (TREF0)	FFH, FFFFH	FFH, FFFFH
Mode registers (TMOD0)	0	0
Output enable flags (TOE0)	0	0
Watch Timer:		
Watch timer mode register (WMOD)	0	0
Comparator		
Comparator mode register (CMOD)	0	0
Comparison result register	Undefined	Undefined
Serial I/O Interface:		
SIO mode register (SMOD)	0	0
SIO interface buffer (SBUF)	Values retained	Undefined

10 I/O PORTS

OVERVIEW

The S3C7031/7032 has four I/O ports. Pin addresses for all I/O ports are mapped to locations FF0H-FF3H in 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. There are15 configurable I/O pins, including one high-current I/O pin.

PORT MODE FLAGS

Port mode flags (PM flags) are used to configure I/O ports 0 and 1 (as port mode group 1) and ports 2 and 3 (as port mode group 2). Port mode group 1 or 2 is configured to input or output mode by setting or clearing the corresponding I/O buffer. PM flags are stored in two 8-bit registers in RAM area FE8H-FEBH, and are addressable by 8-bit write instructions only.

PORT 1 MODE REGISTER

Port 1 (P1.0-P1.3) can be used for either digital or analog input. P1MOD register settings determine the input mode (digital or analog) for specific port 1 pins.

PULL-UP RESISTORS

Pull-up resistors are assignable to the input pins of ports 0, 1, 2, and 3 (in bit units). When a configurable I/O port pin serves as an output pin, its assigned pull-up resistor is automatically disabled, even though the pin's pull-up resistor may be enabled by a corresponding bit setting in the pull-up resistor mode register (PUMOD).

PUMOD CONTROL REGISTERS (PUMOD1/PUMOD2)

The pull-up mode registers PUMOD1 and PUMOD2 are 8-bit register pairs that are used to assign internal pull-up resistors by software to specific I/O ports in bit units.

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 PUMOD bit setting.

The two PUMOD registers are mapped to RAM addresses FDCH-FDDH and FDEH-FDFH respectively, and are addressable by 8-bit write instructions only. RESET clears the PUMOD1/2 register values to logic zero, automatically disconnecting all software-assignable port pull-up resistors.

Table 10-1. I/O Port Overview

Port	I/O	Pins	Pin Names	Address	Function Description
0	I/O	3	P0.0-P0.2	FF0H	3-bit I/O port. 1-bit and 3-bit read/write and test is possible. Individual pins are software configurable as input or output. 1-bit unit pull-up resistors are assignable by software.
1	I/O	4	P1.0-P1.3	FF1H	Same as port 0 except that port1 is 4-bit I/O port.
2, 3	I/O	8	P2.0-P2.3 P3.0-P3.3	FF2H FF3H	4-bit I/O port. 1-bit, 4-bit, or 8-bit read/write and test is possible. Individual pins are software configurable as input or output. 1-bit unit pull-up resistors are assignable by software. Port can be configurable as n-channel open-drain or pushs-pull output by software. Ports 2 and 3 can be paired to support 8-bit data transfer.

Table 10-2. I/O Port Pin Status During Instruction Execution

Instruction Type	Example	Input Mode Status	Output Mode Status

1-bit test
1-bit input
4-bit input
8-bit input
	BTST P0.1
	LDB C,P1.3
	LD A,P3
	LD EA,P2
		Input or test data at each pin	Input or test data at output latch

1-bit output	BITR P3.0	Output latch contents undefined	Output pin status is modified
4-bit output 8-bit output	LD P3,A LD P2,EA	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 0 and 1-3 to input or output mode by setting or clearing the corresponding I/O buffer. PM flags are stored in two 8-bit registers in RAM area FE8H-FEBH, and are addressable by 8-bit write instructions only.

For convenient program reference, PM flags are organized into two groups – PMG1 and PMG2 as shown in Table 10-3.

Table 10-3. Port Mode Groups and Corresponding I/O Ports

Port Mode Group ID	Corresponding I/O Ports	Port Mode Group Address
PMG1	Ports 0 and 1	FE8H-FE9H
PMG2	Ports 2 and 3	FEAH-FEBH

When a PM flag is "0", the port is set to input mode; when it is "1", the port is enabled for output. RESET clears all port mode flags to logic zero, automatically configuring the corresponding I/O ports to input mode.

Table 10-4. Port Mode Flag Map

PM Group ID	Address	Bit 3	Bit 2	Bit 1	Bit 0
PMG1	FE8H	"0"	PM0.2	PM0.1	PM0.0
	FE9H	PM1.3	PM1.2	PM1.1	PM1.0
PMG2	FEAH	PM2.3	PM2.2	PM2.1	PM2.0
	FEBH	PM3.3	PM3.2	PM3.1	PM3.0

NOTE: If bit = "0", the corresponding I/O pin is set to input mode. If bit = "1", the pin is set to output mode. All flags are cleared to "0" following RESET.

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

Configure P0.0 and P3.0 as an output port and the other ports as input ports:

```
BITS    EMB
SMB     15
LD      EA,#01H
LD      PMG1,EA         ;  P0.0 ← Output
LD      EA,#10H
LD      PMG2,EA         ;  P3.0 ← Output
```

PORT 1 MODE REGISTER (P1MOD)

P1MOD register settings determine if port 1 is used for digital input or for analog input. The P1MOD register is a 4-bit write only register. P1MOD is mapped to address FE2H. A reset operation initializes all P1MOD values to logic zero, configuring port 1 as a digital input port.

When a P1MOD bit is "0", the corresponding pin is configured as a digital input pin. When set to "1", it is configured as an analog input pin: P1MOD.0 corresponds to P1.0, P1MOD.1 to P1.1, P1MOD.2 to P1.2, and P1MOD.3 to P1.3.

NOTE

Port 1 pins can be configured individually as inputs or outputs. When P1.0 is configured as output, P1.1 as input, P1.2/P1.3 as comparator input, and IMODK is set to 1000B, IRQK can be set by only P1.1.

PULL-UP RESISTOR MODE REGISTERS (PUMOD1, PUMOD2)

The pull-up resistor mode registers PUMOD1 and PUMOD2 are 8-bit registers used to assign internal pull-up resistors by software to specific I/O 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 may be enabled by a corresponding PUMOD bit setting.

PUMOD1 and PUMOD2 are mapped to RAM addresses FDCH-FDDH and FDEH-FDFH, respectively, and are addressable by 8-bit write instructions only. RESET clears PUMOD register values to logic zero, automatically disconnecting all software-assignable port pull-up resistors.

Table 10-5. Pull-Up Resistor Mode Register (PUMOD) Organization

Register	Address	Bit 3	Bit 2	Bit 1	Bit 0
PUMOD1	FDCH	"0"	PUR0.2	PUR0.1	PUR0.0
	FDDH	PUR1.3	PUR1.2	PUR1.1	PUR1.0
PUMOD2	FDEH	PUR2.3	PUR2.2	PUR2.1	PUR2.0
	FDFH	PUR3.3	PUR3.2	PUR3.1	PUR3.0

NOTE: When bit = "1", a pull-up resistor is assigned to the corresponding I/O port: PUR0.0 for P0.0, PUR0.1 for P0.1, and so on.

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

P2 enable pull-up resistors, P0, P1, and P3 disable pull-up resistors.

```
BITS    EMB
SMB     15
LD      EA,#00H
LD      PUMOD1,EA
LD      EA,#0FH
LD      PUMOD2,EA        ;   P2 enable
```

PORT 0 CIRCUIT DIAGRAM

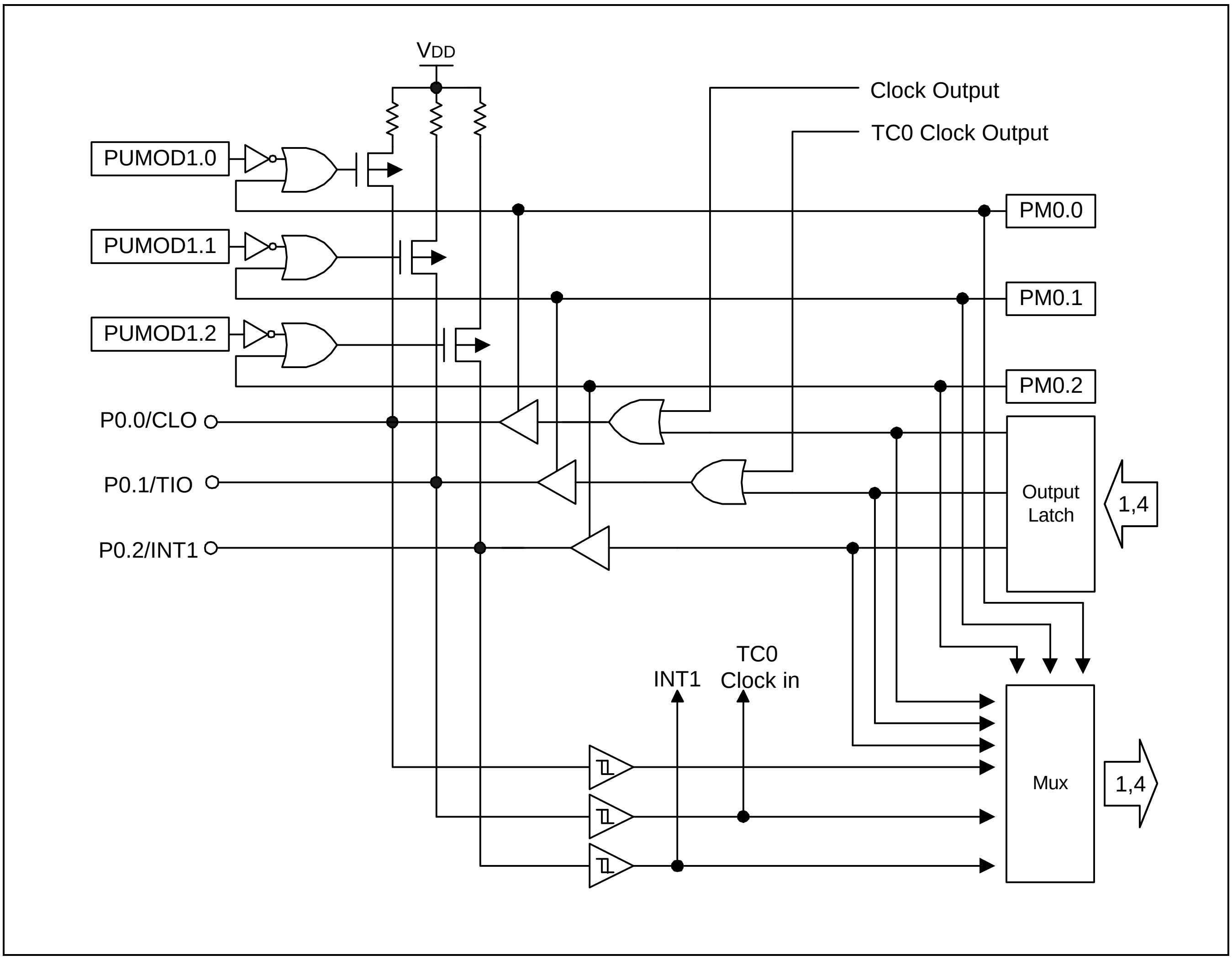


Figure 10-1. I/O Port 0 Circuit Diagram

PORT 1 CIRCUIT DIAGRAM

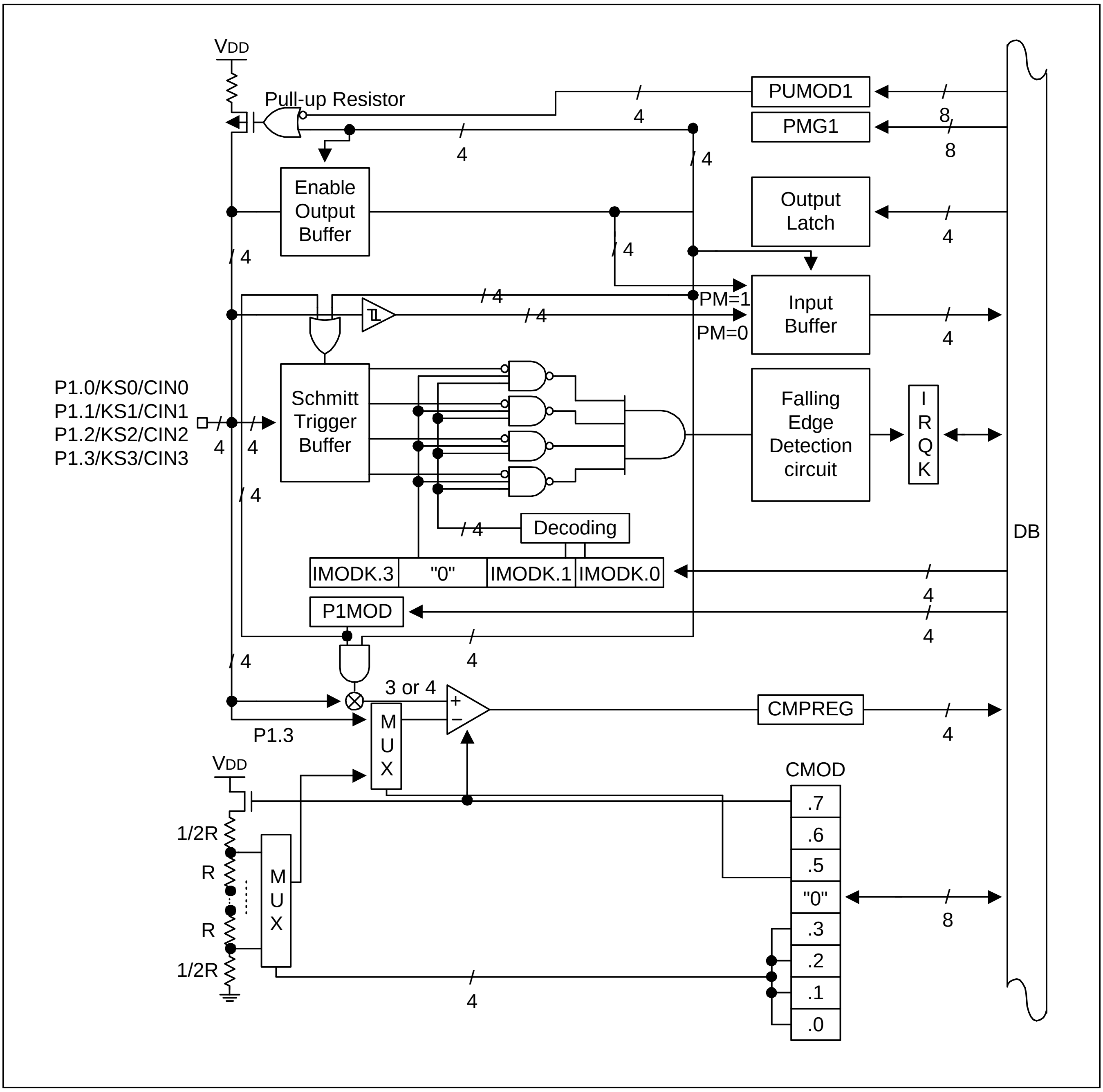


Figure 10-2. Input Port 1 Circuit Diagram

PORT 2 CIRCUIT DIAGRAM

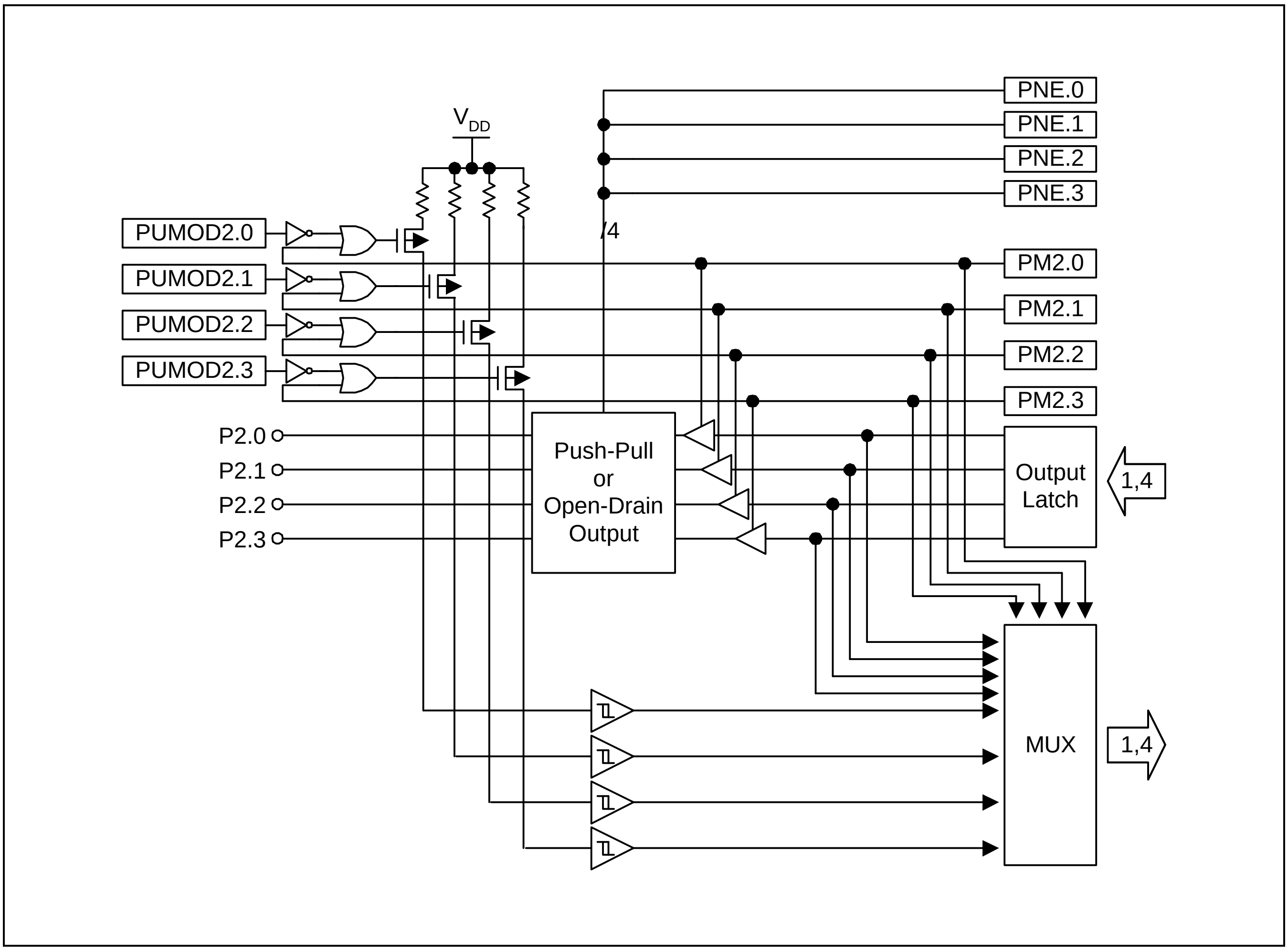


Figure 10-3. Port 2 Circuit Diagram

PORT 3 CIRCUIT DIAGRAM

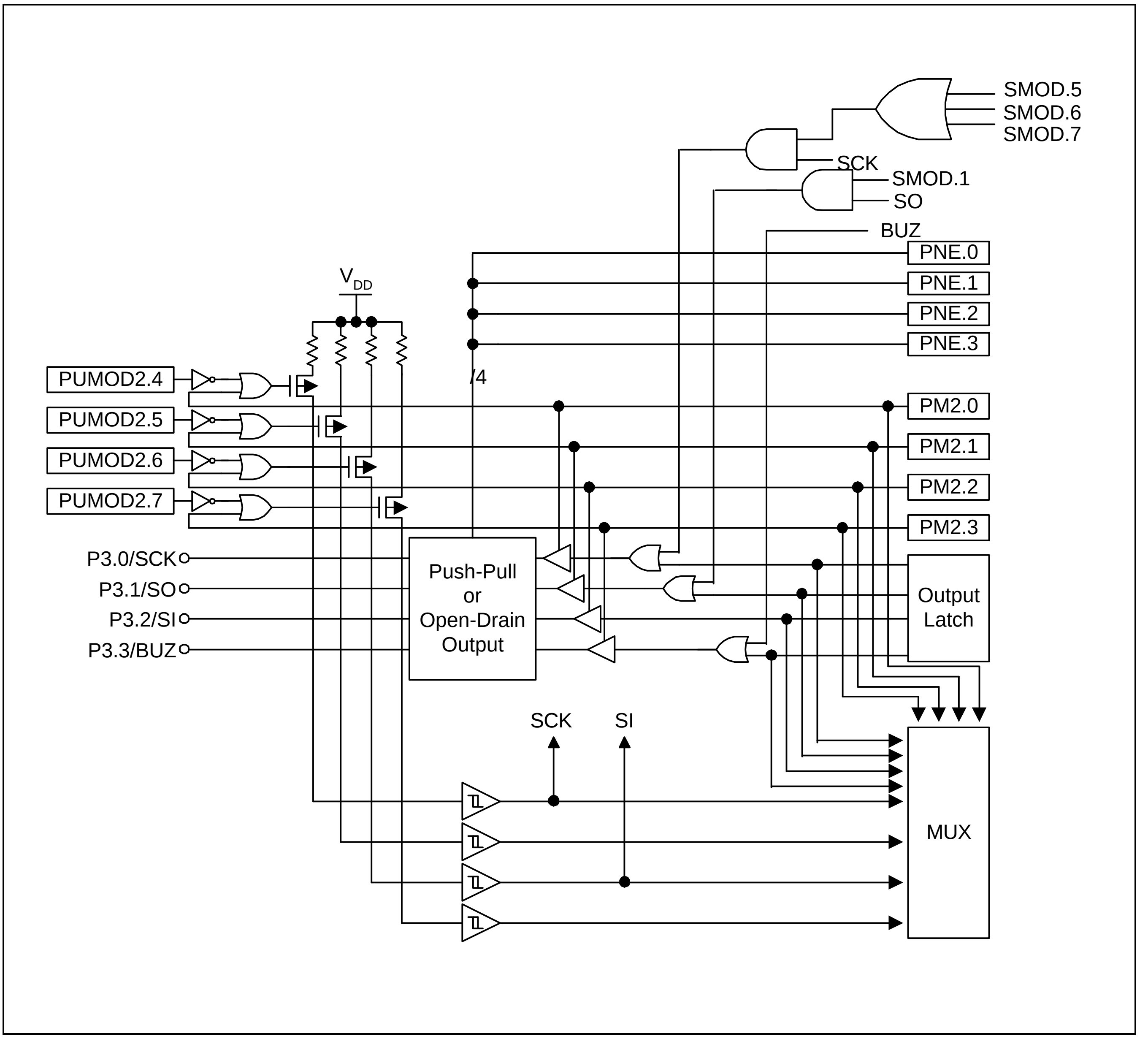


Figure 10-4. Port 3 Circuit Diagram

11 TIMERS and TIMER/COUNTER

OVERVIEW

The S3C7032 microcontroller has three timer and timer/counter function modules:

— 8-bit basic timer (BT)
— 8-bit timer/counter 0 (TC0)
— Watch timer (WT)

The 8-bit basic timer (BT) is the microcontroller's main interval timer. It generates a interrupt request at a fixed time interval by making the appropriate modification to the 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 0 (TC0) is programmable timer/counter that is used primarily for event counting and for clock frequency modification and output. In addition, TC0 generates a clock signal that can be used by the serial I/O interface.

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

BASIC TIMER (BT)

OVERVIEW

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

— Clock selector logic
— 4-bit mode register (BMOD)
— 8-bit counter register (BCNT)

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.

Use the basic timer mode register, BMOD, to select input clock frequency, and to control stabilization intervals.

Interval Timer Function

The measurement of elapsed time intervals is the basic timer's primary function. The standard interval is 256 BT clock pulses.

To restart the basic timer, set bit 3 of the mode register BMOD 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.

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 then generated, BCNT is cleared to logic zero, and counting continues from 00H.

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 power-down mode is released by an interrupt. When a RESET signal is generated, the standard stabilization interval for system clock oscillation following a 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 power-down mode release or RESET	4-bit	F85H	4-bit write-only; BMOD.3: also 1-bit writeable	"0"
BCNT	Counter	Counts clock pulses matching the BMOD frequency setting	8-bit	F86H-F87H	8-bit read-only	U [1)]

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

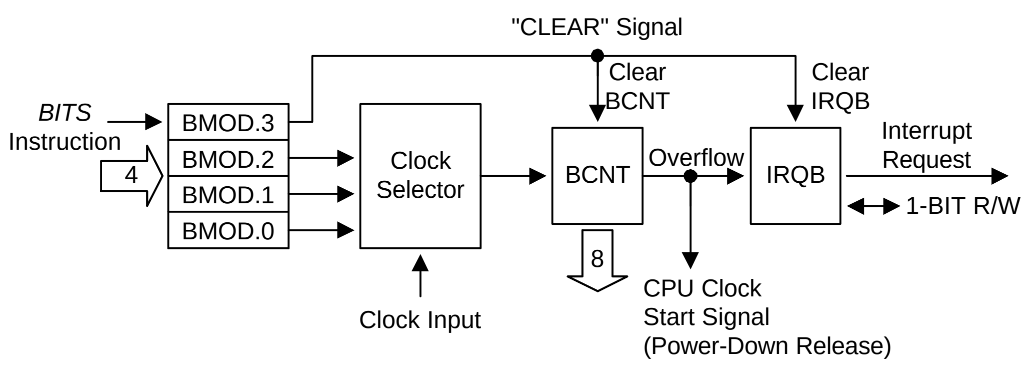


Figure 11-1. Basic Timer Circuit Diagram

BASIC TIMER MODE REGISTER (BMOD)

The basic timer mode register, BMOD, is a 4-bit write-only register located at RAM address F85H. 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 halted.) BMOD settings have the following effects:

— Restart the basic timer,

— Control the frequency of clock signal input to the basic timer, and

— Determine time interval required for clock oscillation to stabilize following the release of Stop modes 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 $fx/2^{12}$ (1.02 kHz) to $fx/2^{5}$ (131 kHz), are selectable. Since BMOD's reset value is logic zero, the default clock frequency setting is $fx/2^{12}$. (All kHz frequency values assume a system clock (fx) frequency of 4.19 MHz.)

The most significant bit of the BMOD register, BMOD.3, is used to start the basic timer again. When BMOD.3 is set to logic one (enabled) 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 is restarted.

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 Enable/Disable Control Bit
1	Start basic timer; clear IRQB, BCNT, and BMOD.3 to "0"

BMOD.2	BMOD.1	BMOD.0	Basic Timer Input Clock	Oscillation Stabilization
0	0	0	$fx/2^{12}$ (1.02 kHz)	$2^{20}/fx$ (250 ms)
0	1	1	$fx/2^{9}$ (8.18 kHz)	$2^{17}/fx$ (31.3 ms)
1	0	1	$fx/2^{7}$ (32.7 kHz)	$2^{15}/fx$ (7.82 ms)
1	1	1	$fx/2^{5}$ (131 kHz)	$2^{13}/fx$ (1.95 ms)

NOTES:
1. Clock frequencies and stabilization intervals assume a system oscillator clock frequency (fx) of 4.19 MHz.
2. fx = system clock frequency.
3. Oscillation stabilization time is the time required to stabilize clock signal oscillation after Stop mode is released.
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 register for the basic timer. It is mapped to RAM addresses F86H-F87H and can be addressed by 8-bit read instructions.

RESET leaves the BCNT register 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 incremented to hexadecimal 'FFH' (256 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 bit BMOD.3 to logic one to restart basic timer operation
2. BCNT is incremented by one after each clock pulse corresponding to BMOD selection
3. BCNT overflows if BCNT≥255 (FFH)
4. When an overflow occurs, the IRQB flag is set to logic one by hardware
5. The interrupt request is generated
6. BCNT is automatically cleared to logic zero (BCNT = 00H)
7. BCNT resumes counting BT clock pulse

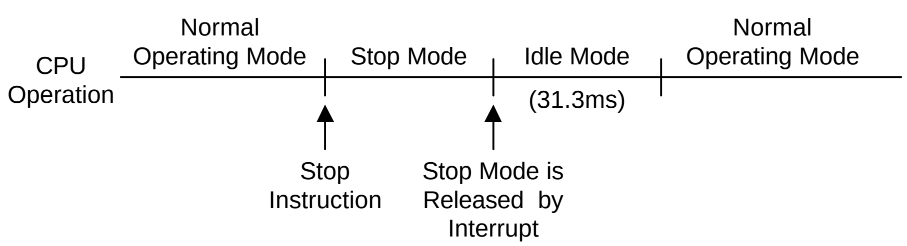
☞ PROGRAMMING TIP — Using the Basic Timer

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
        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
```

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. You can modify the basic frequency dynamically by loading new values into TMOD0 during program execution.

TC0 FUNCTION SUMMARY

8-bit programmable timer	Generates interrupts at specific time intervals based on the selected clock frequency.
External event counter	Counts various system "events" based on edge detection of external clock signals at the TC0 input pin, TIO. To start the event counting operation, TMOD0.2 is set to "1" and TMOD0.6 is cleared to "0".
Arbitrary frequency output	Outputs selectable clock frequencies to the TC0 output pin, TIO.
Serial I/O clock source	Outputs a modifiable clock signal for use as the SCK clock source.

TC0 COMPONENT SUMMARY

Mode register (TMOD0)	Activates the timer/counter and selects the internal clock frequency or the external clock source at the TIO 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 external clock frequency.
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 TC0 interrupt request or clock pulse is stored pending output to the serial I/O circuit or to the TC0 I/O pin, TIO. 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 (TOE0)	You must set this flag to logic one before the contents of the TOL0 latch can be output to TIO.
Interrupt request flag (IRQT0)	This flag is cleared when TC0 operation starts and the TC0 interrupt service routine is executed and is enabled 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 can be processed.

Table 11-3. TC0 Register Overview

Register Name	Type	Description	Size	RAM Address	Addressing Mode	Reset Value
TMOD0	Control	Controls TC0 restart (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 write-only)	"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
TOE0	Flag	Controls timer/counter 0 output to the TIO pin	1-bit	F92H.2	1-bit read/write	"0"

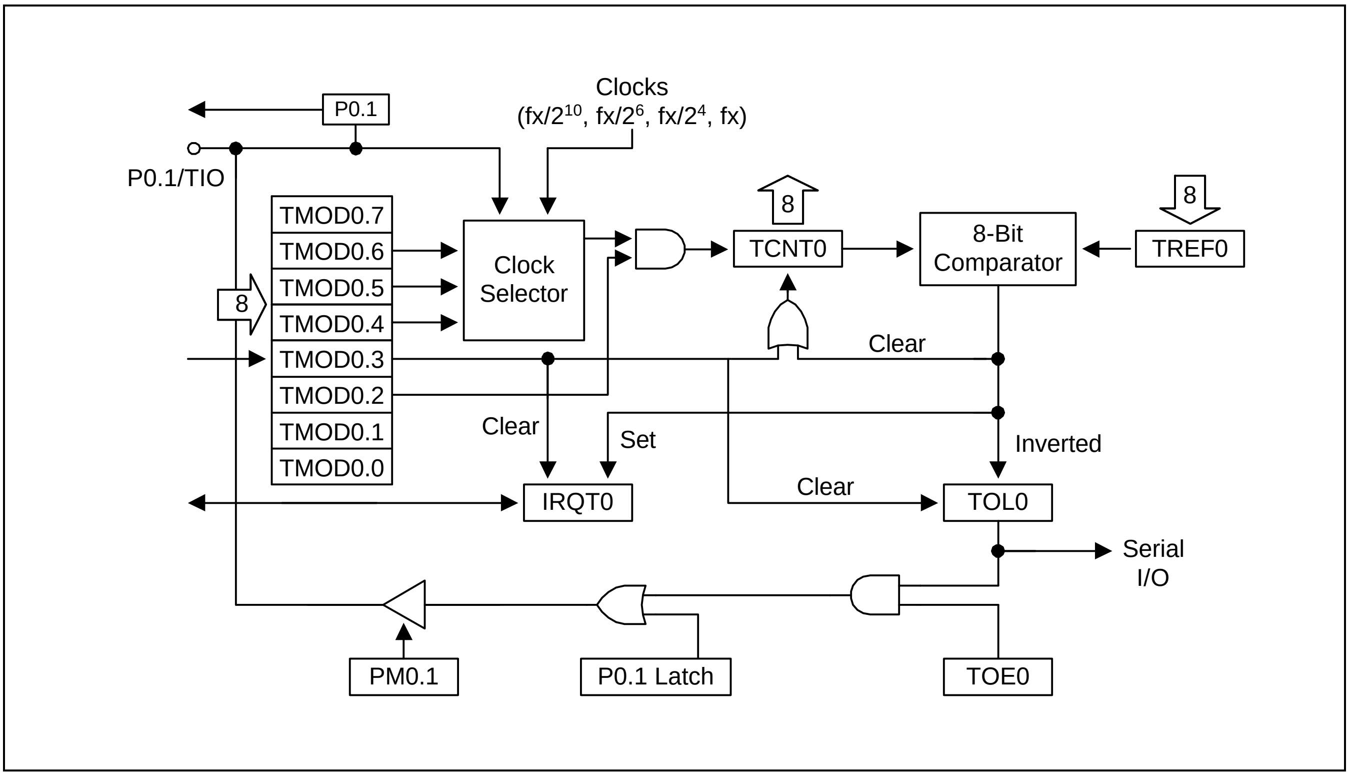


Figure 11-2. TC0 Circuit Diagram

TC0 ENABLE/DISABLE PROCEDURE

Enable Timer/Counter 0

— Set TMOD.2 to logic one (RAM address F90H.2)

— Set the TC0 interrupt enable flag IET0 to logic one (RAM address FBCH.1)

— Set TMOD0.3 to logic one (RAM address F90H.3)

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

Disable Timer/Counter

— Set TMOD0.2 to logic zero (RAM address F90H.2)

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 system clock frequency you select.

The 8-bit TC0 mode register, TMOD0, is used to activate the timer/counter and to select the clock frequency. The reference register, TREF0, stores your 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 to generate interrupt requests at specific intervals, you choose one of four internal clock frequencies (divisions of the system clock, fx) and load your own 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 the programmable timer/counter consists of the TC0 interrupt enable flag IET0 and the TC0 interrupt request flag IRQT0.

TC0 OPERATION SEQUENCE

The general sequence of operations when using TC0 as a programmable timer/counter 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 (fx) input
3. Set TMOD0.5 and TMOD0.4 bits to desired internal frequency ($fx/2^n$)
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"
9. Output latch (TOL0) logic toggles high or low
10. Interrupt request is generated
11. TCNT0 is cleared to 00H and counting resumes
12. Programmable timer/counter operation continues until TMOD0.2 is cleared to "0".

TC0 EVENT COUNTER FUNCTION

Timer/counter 0 can be used to monitor or detect system 'events' by using the external clock input at the TIO pin (I/O port 0.1) as the counter source. The TC0 mode register is used to specify rising or falling edge detection for incoming clock signals. The counter register TCNT0 is incremented each time the selected state transition of the external clock signal occurs. To activate the TC0 event counter function,

— Set TMOD0.2 to "1" to enable TC0
— Clear TMOD0.6 to "0" to select the external clock source at the TIO pin
— Select TIO edge detection for rising or falling signal edges by loading the appropriate values to TMOD0.5 and TMOD0.4.
— P0.1 must be set to input mode.

Table 11-4. TMOD0 Settings for TIO Edge Detection

TMOD0.5	TMOD0.4	TIO Edge Detection
0	0	Rising edges
0	1	Falling edges

With the exception of the different TMOD0.4-TMOD0.6 settings, the operation sequence for TC's event counter function is identical to its programmable counter/timer function.

NOTE

If event count function is selected when P0.1 is configured as output mode, TC0 will count the output signal to set IRQT0.

TC0 CLOCK FREQUENCY OUTPUT

Using timer/counter, you can output a modifiable clock frequency to the TC0 clock output pin, TIO. To select the clock frequency, you load appropriate values to the TC0 mode register, TMOD0. The clock interval is determined by loading the desired reference value into the reference register TREF0. Then, to enable the output to the TIO pin at I/O port 0.1, the following conditions must be met:

— TC0 output enable flag TOE0 must be set to "1"

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

— Output latch value for P0.1 must be set to "0"

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

1. Load your reference value to TREF0
2. Set the clock frequency in TMOD0
3. Initiate TC0 clock output to TIO (TMOD0.2 = "1")
4. Set port 3 mode flag (PM0.1) to "1"
5. Set P0.1 output latch to "0"
6. Set TOE0 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 TIO pin.

☞ PROGRAMMING TIPS — TC0 Signal Output to the TIO Pin

Output a 30 ms pulse width signal to the TIO pin: (fx = 4.19 MHz)

```
BITS    EMB
SMB     15
LD      EA,#7AH
LD      TREF0,EA
LD      EA,#4CH
LD      TMOD0,EA
LD      EA,#02H
LD      PMG1,EA        ;  P0.1 ← Output mode
BITR    P0.1           ;  P0.1 clear
BITS    TOE0
```

TC0 SERIAL I/O CLOCK GENERATION

Timer/counter 0 can supply a clock signal to the clock selector circuit of the serial I/O interface for data shifter and clock counter operations. (These internal SIO operations are controlled in turn by the SIO mode register, SMOD). This clock generation function enables you to adjust data transmission rates across the serial interface.

Use TMOD0 and TREF0 register settings to select the frequency and interval of the TC0 clock signals to be used as SCK input to the serial interface. The generated clock signal is then sent directly to the serial I/O clock selector circuit – not through the port 0.1 latch and TIO pin.

TC0 MODE REGISTER (TMOD0)

TMOD0 is the 8-bit mode control register for timer/counter. It is located at RAM addresses F90H–F91H and 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. 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.

Use TMOD0.6, TMOD0.5, and TMOD0.4 bit settings 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 TIO 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	MSB value 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; retain TCNT0 contents	
	1	Enable timer/counter	
TMOD0.1	0	Value always logic zero	
TMOD0.0	0	LSB value always logic zero	

Table 11-6. TMOD0.6, TMO0.5, and TMOD0.4 Bit Settings

TMOD0.6	TMOD0.5	TMOD0.4	Resulting Counter Source and Clock Frequency
0	0	0	External clock input (TCL0) on rising edges
0	0	1	External clock input (TCL0) on falling edges
1	0	0	$fx/2^{10}$ = 4.09 kHz
1	0	1	$fx/2^{6}$ = 65.5 kHz
1	1	0	$fx/2^{4}$ = 262 kHz
1	1	1	fx = 4.19 MHz

NOTE: 'fx' = system clock

☞ 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, TCNT0, is mapped to RAM addresses F94H-F95H. It 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 are enabled, TCNT0 is cleared to logic zero and counting begins. 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 reference register, 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.

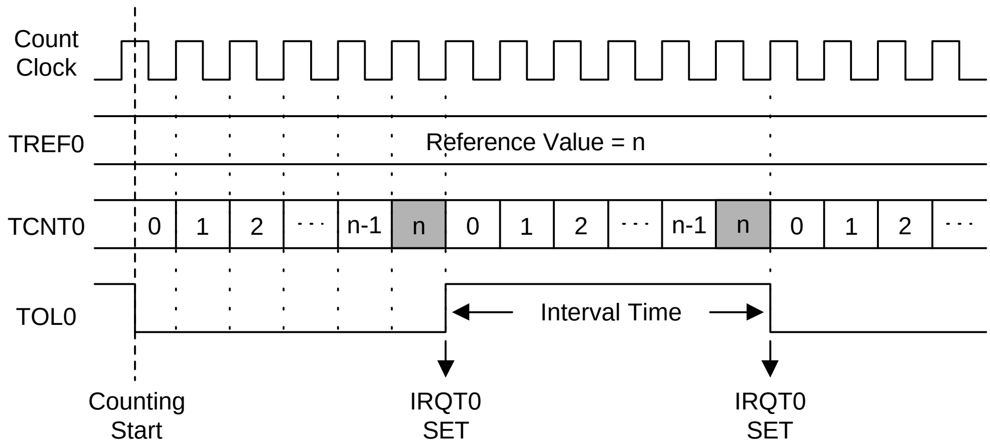


Figure 11-3. TC0 Timing Diagram

TC0 REFERENCE REGISTER (TREF0)

The TC0 reference register TREF0 is an 8-bit write-only register that is mapped to RAM locations F96H and F97H. 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 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:

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

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

TC0 OUTPUT ENABLE FLAG (TOE0)

The 1-bit timer/counter 0 output enable flag TOE0 controls output from timer/counter 0 to the TIO pin. TOE0 is mapped to RAM location F92H.2 and is addressable by 1-bit read and write instructions.

	Bit 3	**Bit 2**	Bit 1	Bit 0
F92H	0	**TOE0**	0	0

When you set the TOE0 flag to "1", the contents of TOL0 can be output to the TIO pin. Whenever a RESET occurs, TOE0 is automatically set to logic zero, disabling all TC0 output. Even when the TOE0 flag is disabled, timer/counter can continue to output an internally-generated clock frequency, via TOL0, to the serial I/O clock selector circuit.

TC0 OUTPUT LATCH (TOL0)

TOL0 is the output latch for timer/counter. When the 8-bit comparator detects a correspondence between the value of the counter register TCNT0 and the reference value stored in the TREF0 buffer, 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 TIO pin at P0.1, or it can be output directly to the serial I/O clock selector circuit as the SCK signal.

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 fx = 4.19 MHz, follow these steps:

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

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

$$\text{TREF0} + 1 = \frac{30 \text{ ms}}{244\ \mu\text{s}} = 123 = 7BH$$

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

3. Load the value 7AH to the TREF0 register:

```
BITS    EMB
SMB     15
LD      EA,#7AH
LD      TREF0,EA
LD      EA,#4CH
LD      TMOD0,EA
```

WATCH TIMER

OVERVIEW

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

— 8-bit watch timer mode register (WMOD)

— Frequency divider circuit

Watch timer functions include real-time and watch-time measurement and interval timing for the system clock. It is also used as a clock source for generating buzzer 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 System Clock Source

The watch timer can generate interrupts based on the system clock frequency. The system clock (fx) is used as the signal source, according to the following formula:

$$\text{Watch timer clock (fw)} = \frac{\text{System clock (fx)}}{128} = 32.768 \text{ kHz}$$

(assuming fx = 4.19 MHz)

Buzzer Output Frequency Generator

The watch timer can generate a steady 2 kHz, 4 kHz, 8 kHz, or 16 kHz signal to the BUZ pin. To select the BUZ frequency you want, 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 at F89H.3 is set to "1"

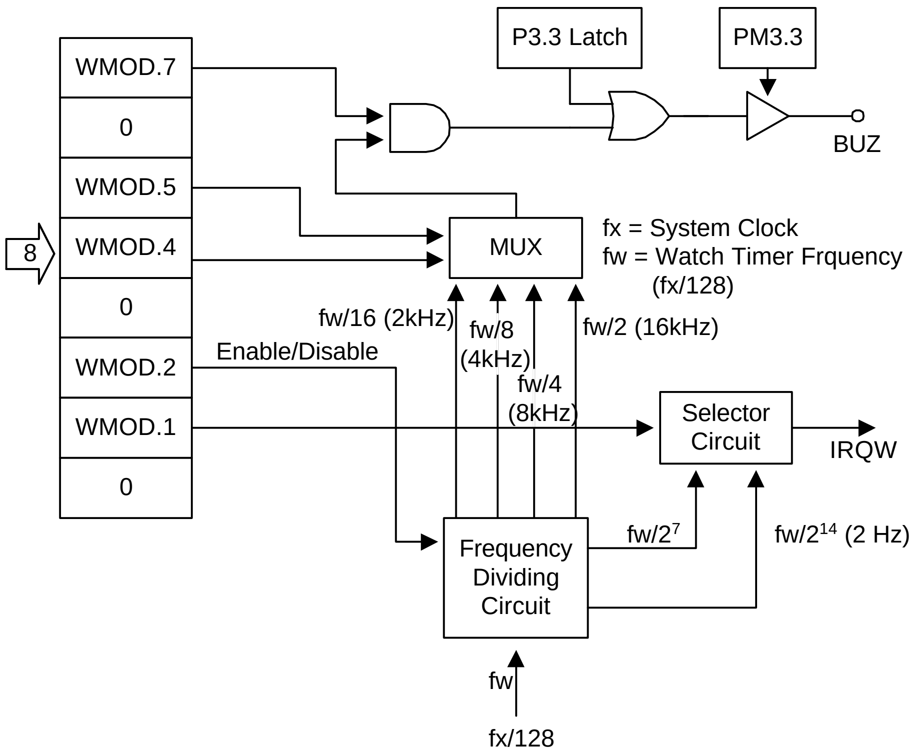
— The output latch for I/O port 3.3 is cleared to "0"

— The port 3.3 output mode flag (PM3.3) set to 'output' mode

Timing Tests in High-Speed Mode

By setting WMOD.1 (F88H.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.

WATCH TIMER CIRCUIT



NOTE: System clock frequency (fx) is assumed to be 4.19 MHz

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 mapped to RAM locations F88H-F89H and is 8-bit write-only addressable.

RESET sets all WMOD bits to logic zero.

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

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

— Watch timer speed control (WMOD.1)
— Enable/disable watch timer (WMOD.2)
— Buzzer frequency selection (WMOD.4)
(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	F89H
	1		Enable buzzer (BUZ) signal output	
WMOD.6	"0"		Always logic zero	
WMOD.5 – .4	0	0	2 kHz buzzer (BUZ) signal output	
	0	1	4 kHz buzzer (BUZ) signal output	
	1	0	8 kHz buzzer (BUZ) signal output	
	1	1	16 kHz buzzer (BUZ) signal output	
WMOD.3	"0"		Always logic zero	F88H
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		Always logic zero	

NOTE: System clock frequency (fx) is assumed to be 4.19 MHz. WMOD.0 must be set to "0" to operate the watch timer.

☞ PROGRAMMING TIP — Using the Watch Timer

1. Select a 0.5 second interrupt, and 2 kHz buzzer enable:

```
        BITS    EMB
        SMB     15
        LD      EA,#80H
        LD      PMG2,EA         ; P3.3 ← Output mode
        BITR    P3.3            ; Clear P3.3 output latch
        LD      EA,#84H
        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 COMPARATOR

OVERVIEW

Port 1 can be used as a analog input port for a comparator. The reference voltage for the 4-channel comparator can be supplied either internally or externally at P1.3. When an internal reference voltage is used, four channels (P1.0-P1.3) are used for analog inputs and the internal reference voltage is varied in 16 levels. If an external reference voltage is input at P1.3, the other three port 1 pins (P1.0-P1.2) are used for analog input.

When a conversion is completed, the result is saved in the comparison result register CMPREG. The initial values of the CMPREG are undefined and the comparator operation is disabled by a RESET. The comparator module has the following components:

— Comparator
— Internal reference voltage generator (4-bit resolution)
— External reference voltage source at P1.3
— Comparator mode register (CMOD)
— Comparison result register (CMPREG)

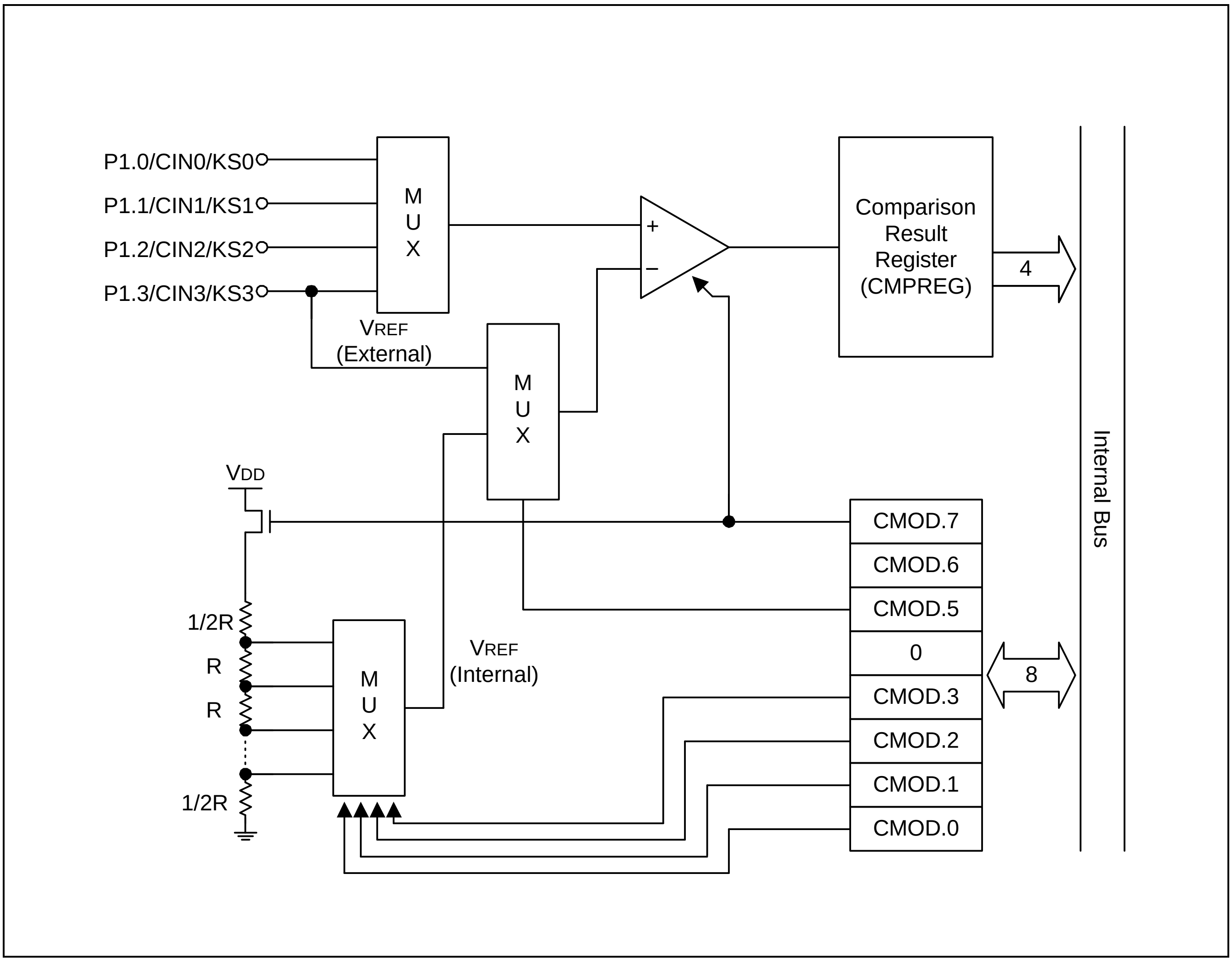


Figure 12-1. Comparator Circuit Diagram

COMPARATOR MODE REGISTER (CMOD)

The comparator mode register CMOD is an 8-bit register that is used to select the operation mode of the comparator. It is mapped to addresses FD6H-FD7H and can be manipulated using 8-bit memory instructions. Based on the CMOD.5 bit setting, an internal or an external reference voltage is input for the comparator, as follows:

When CMOD.5 is set to logic zero:

— A reference voltage is selected by the CMOD.0 to CMOD.3 bit settings.

— P1.0 to P1.3 are used as analog input pins.

— The internal digital to analog converter generates 16 reference voltages.

— The comparator can detect 150-mV differences between the reference voltage and the analog input voltages.

— Comparator results are written into 4-bit comparison result register (CMPREG).

When CMOD.5 is set to logic one:

— An external reference voltage is supplied from P1.3/CIN3.

— P1.0 to P1.2 are used as the analog input pins.

— The comparator can detect 50-mV differences between the reference voltage and the analog input voltages.

— Bits 0-2 in the CMPREG register contain the results; the content of bit 3 is not used.

Bit 6 in the CMOD register controls conversion time while bit 7 enables or disables comparator operation to reduce power consumption. A RESET signal clears all bits to logic zero, causing the comparator to enter stop mode.

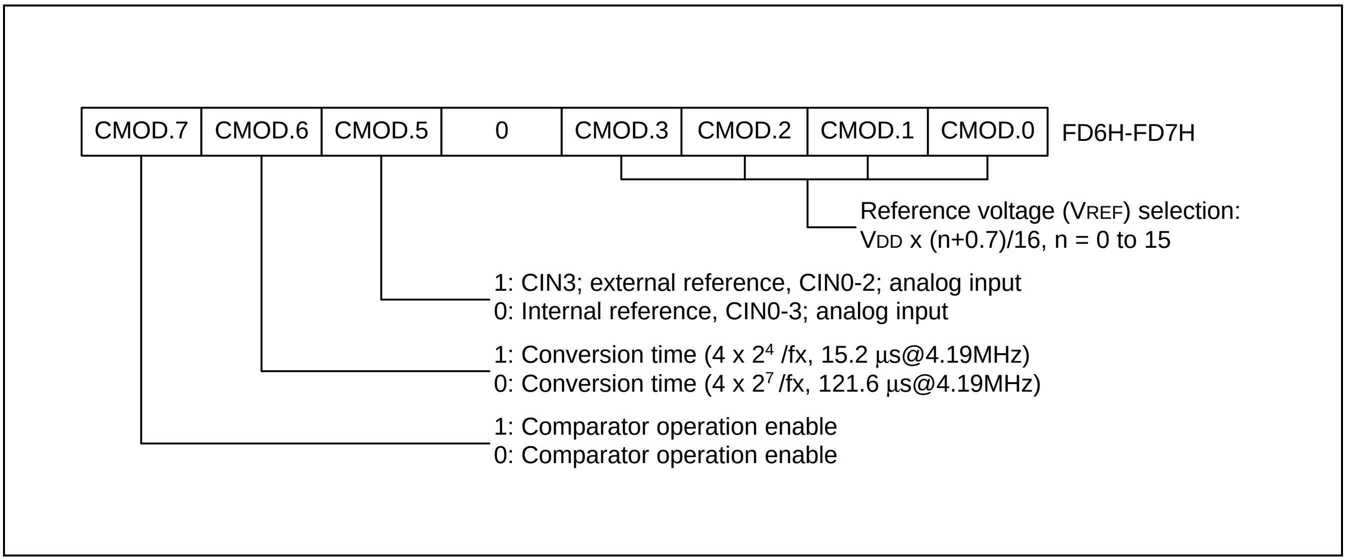


Figure 12-2. Comparator Mode Register (CMOD) Organization

PORT 1 MODE REGISTER (P1MOD)

P1MOD register settings determine if port 1 is used for analog or digital input. The P1MOD register is 4-bit write-only register. P1MOD is mapped to address FE2H. A reset operation initializes all P1MOD register values to zero, configuring port 1 as a digital input port.

FE2H	P1MOD.3	P1MOD.2	P1MOD.1	P1MOD.0

When a P1MOD bit is set to "0", the corresponding pin is configured as a digital input pin. When set to "1", it is configured as an analog input pin: P1MOD.0 for P1.0, P1MOD.1 for P1.1, P1MOD.2 for P1.2, and P1MOD.3 for P1.3.

COMPARATOR OPERATION

The comparator compares analog voltage input at CIN0-CIN3 with an external or internal reference voltage (V_{REF}) that is selected by the CMOD register. The result is written to the comparison result register CMPREG at address FD4H. The comparison result at internal reference is calculated as follows:

If "1" Analog input voltage $\geq V_{REF}$ + 150 mV (+50 mV at external reference)

If "0" Analog input voltage $\leq V_{REF}$ - 150 mV (-50 mV at external reference)

To obtain a comparison result, the data must be read out from the CMPREG register after V_{REF} is updated by changing the CMOD value after a conversion time has elapsed.

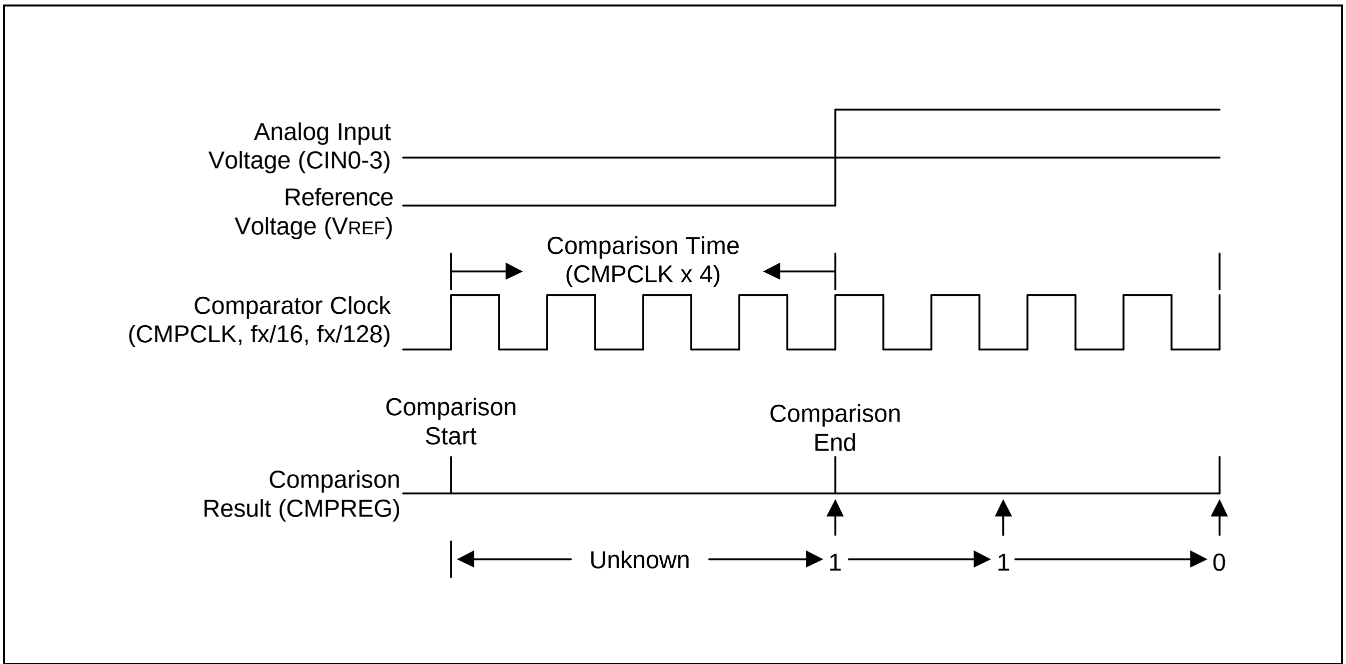


Figure 12-3. Conversion Characteristics

☞ PROGRAMMING TIP — Programming the Comparator

The following code converts the analog voltage input at the CIN0–CIN3 pins into 4-bit digital code.

```
        BITR    EMB
        LD      A,#0F
        LD      P1MOD,A         ;  Analog input selection (CIN0-CIN3)
        LD      EA,#0CXH        ;  x = 0-F, comparator enable
                                ;  Internal reference, conversion time
                                   (15.2 µs at 4.19 MHz)
        LD      CMOD,EA
        LD      A,#0H
WAIT    INCS    A
        JR      WAIT
        LD      A,CMPREG        ;  Read the result
        LD      P2,A            ;  Output the result from port 2
```

13 SERIAL I/O INTERFACE

OVERVIEW

The serial I/O interface (SIO) has the following functional components:

— 8-bit mode register (SMOD)

— Clock selector circuit

— 8-bit buffer register (SBUF)

— 3-bit serial clock counter

Using the serial I/O interface, you can exchange 8-bit data with an external device. You control the transmission frequency by the appropriate bit settings to the SMOD register.

The serial interface can run off an internal or an external clock source, or the TOL0 signal that is generated by the 8-bit timer/counter 0, TC0. If you use the TOL0 clock signal, you can modify its frequency to adjust the serial data transmission rate.

SIO OPERATION SEQUENCE

The general sequence of operations for the serial I/O interface may be summarized as follows:

1. Set SIO mode to transmit-and-receive or to receive-only.
2. Select MSB-first or LSB-first transmission mode.
3. Set the SCK clock signal in the mode register, SMOD.
4. Set SIO interrupt enable flag (IES) to "1".
5. Initiate SIO transmission by setting bit 3 of the SMOD to "1".
6. When the SIO operation is complete, IRQS flag is set and an interrupt is generated.

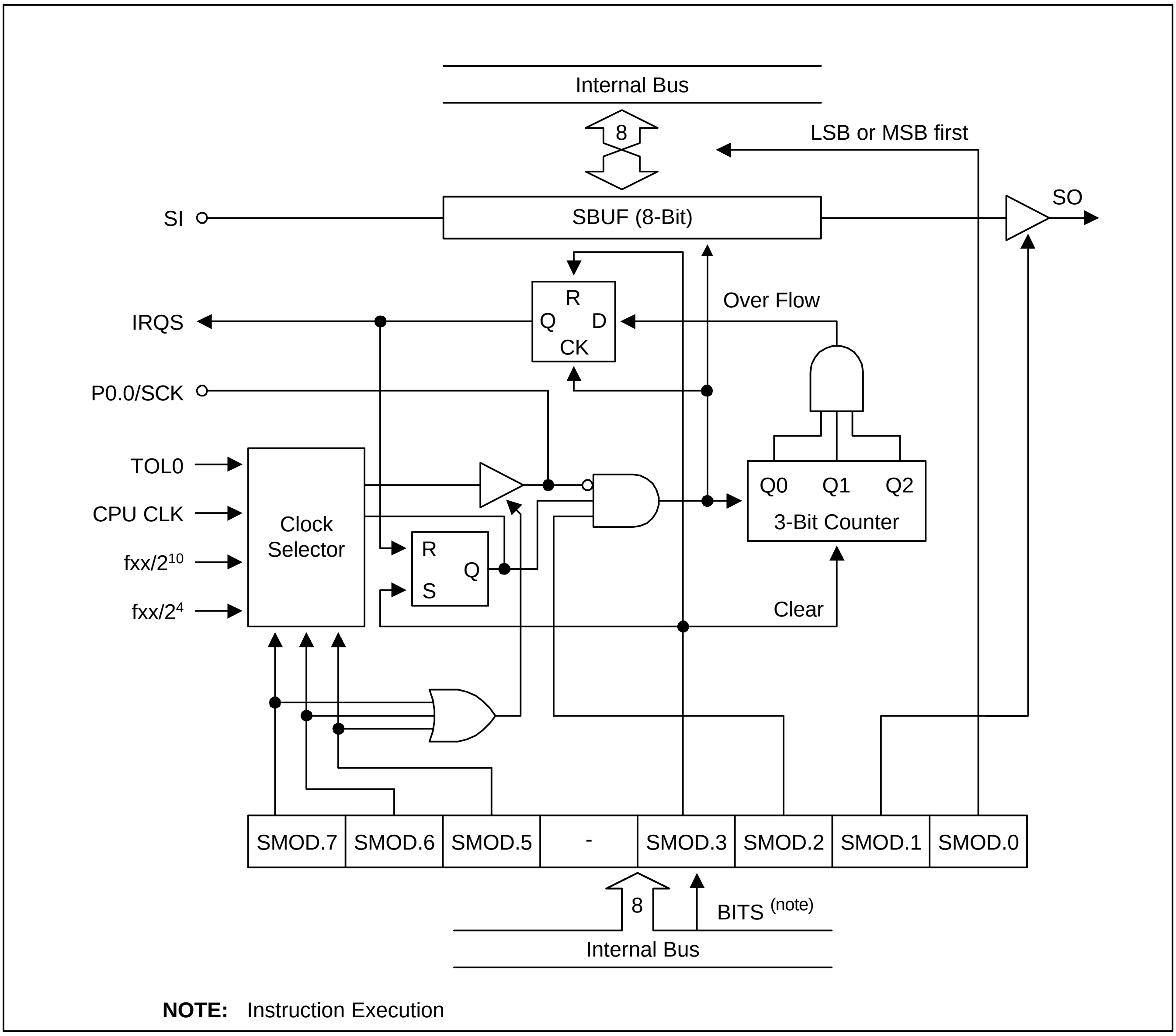


Figure 13-1. Serial I/O Interface Circuit Diagram

SERIAL I/O MODE REGISTER (SMOD)

The serial I/O mode register, SMOD, is an 8-bit register that specifies the operation mode of the serial interface. SMOD is mapped to RAM address FE0H-FE1H and its reset value is logic zero. SMOD is organized in two 4-bit registers, as follows:

FE0H	SMOD.3	SMOD.2	SMOD.1	SMOD.0
FE1H	SMOD.7	SMOD.6	SMOD.5	"0"

SMOD register settings enable you to select either MSB-first or LSB-first serial transmission, and to operate in transmit-and-receive mode or receive-only mode.

SMOD is a write-only register and can be addressed only by 8-bit RAM control instructions. One exception to this is SMOD.3, which can be written by a 1-bit RAM control instruction. When SMOD.3 is set to 1, the contents of the serial interface interrupt request flag, IRQS, and the 3-bit serial clock counter are cleared, and SIO operations are initiated. When the SIO transmission starts, SMOD.3 is cleared to logic zero.

Table 13-1. SIO Mode Register (SMOD) Organization

SMOD.0	0	Most significant bit (MSB) is transmitted first
	1	Least significant bit (LSB) is transmitted first
SMOD.1	0	Receive-only mode; output buffer is off
	1	Transmit-and-receive mode
SMOD.2	0	Disable the data shifter and clock counter; retain contents of IRQS flag when serial transmission is halted
	1	Enable the data shifter and clock counter; set IRQS flag to "1" when serial transmission is halted
SMOD.3	1	Clear IRQS flag and 3-bit clock counter to "0"; initiate transmission and then reset this bit to logic zero
SMOD.4	"0"	Bit not used; value is always "0"

SMOD.7	SMOD.6	SMOD.5	Clock Selection	R/W Status of SBUF
0	0	0	External clock at SCK pin	SBUF is enabled when SIO operation is halted or when SCK goes high.
0	0	1	Use TOL0 clock from TC0	
0	1	x	CPU clock: fx/4, fx/8, fx/64	Enable SBUF read/write
1	0	0	4.09 kHz clock: $fx/2^{10}$	SBUF is enabled when SIO operation is halted or when SCK goes high.
1	1	1	262 kHz clock: $fx/2^{4}$	

NOTES:
1. 'fx' = system clock; 'x' means 'don't care.'
2. kHz frequency ratings assume a system clock (fx) running at 4.19 MHz.
3. The SIO clock selector circuit cannot select a $fx/2^{4}$ clock if the CPU clock is fx/64.

SERIAL I/O TIMING DIAGRAMS

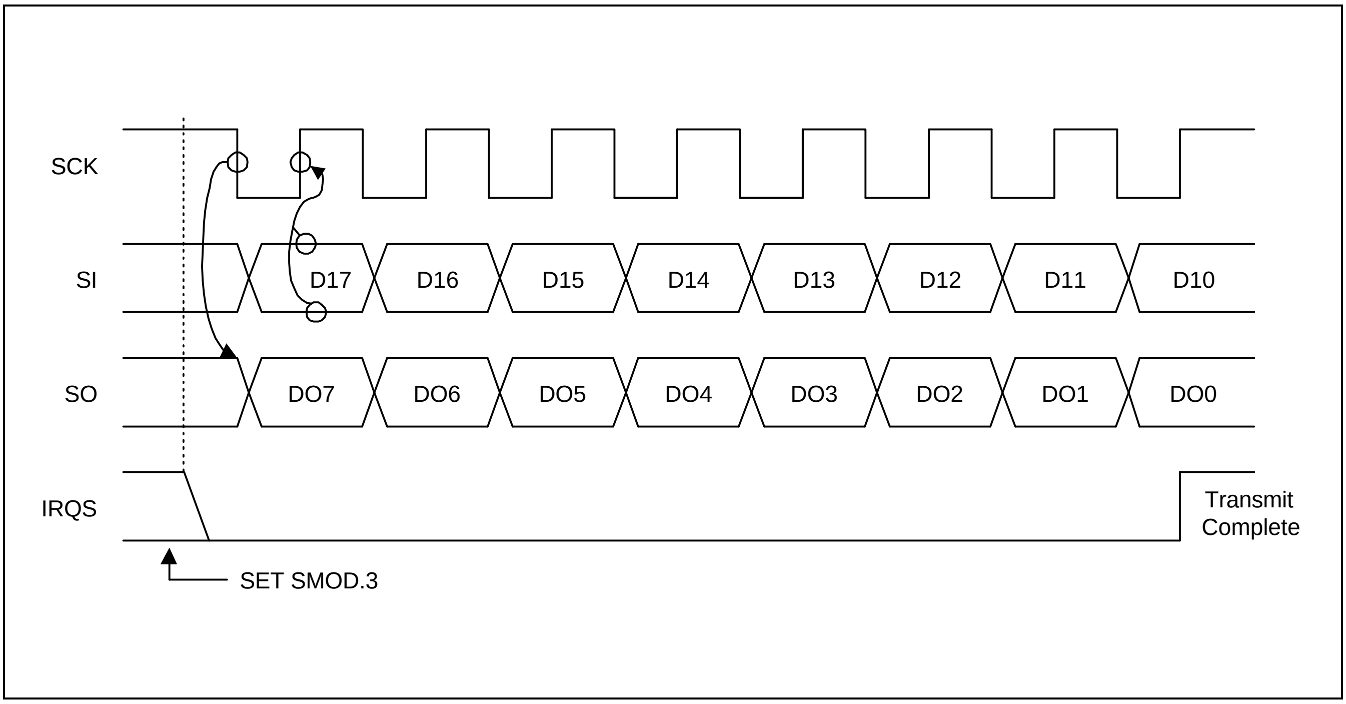


Figure 13-2. SIO Timing in Transmit/Receive Mode

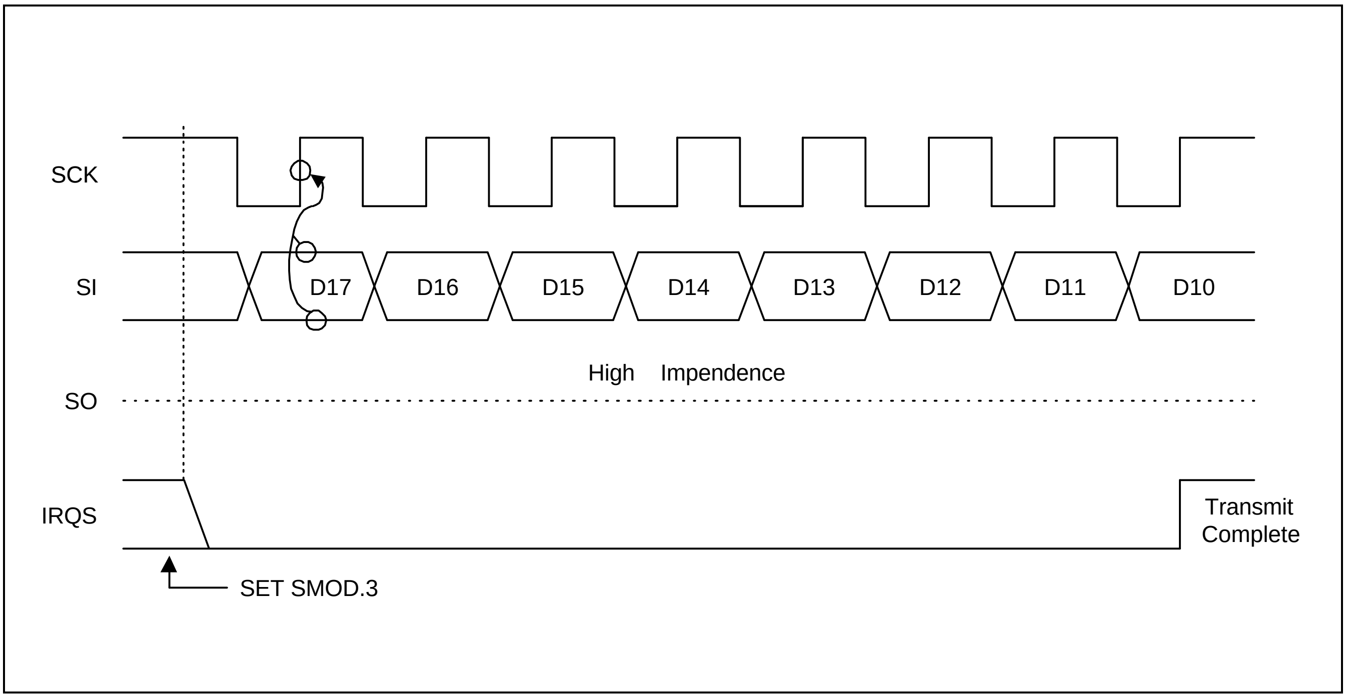


Figure 13-3. SIO Timing in Receive-Only Mode

SERIAL I/O BUFFER REGISTER (SBUF)

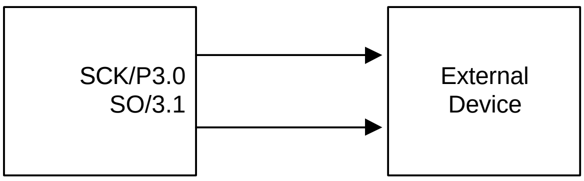
When the serial interface operates in transmit-and-receive mode (SMOD.1 = "1"), transmit data in the SIO buffer register are output to the SO pin (P3.1) at the rate of one bit for each falling edge of the SIO clock. Receive data is simultaneously input from the SI pin (P3.2) to SBUF at the rate of one bit for each rising edge of the SIO clock.

When receive-only mode is used, incoming data is input to the SIO buffer at the rate of one bit for each rising edge of the SIO clock. SBUF can be read or written using 8-bit RAM control instructions. It is mapped to addresses FE4H-FE5H.

☞ PROGRAMMING TIP — Setting Transmit/Receive Modes for Serial I/O

1. Transmit the data value 48H through the serial I/O interface using an internal clock frequency of fx/2^4 and in MSB-first mode:

```
        BITS    EMB
        SMB     15
        LD      EA,#30H
        LD      PMG2,EA         ;  P3.0 / SCK and P3.1 / SO  ←  Output
        LD      EA,#48H         ;
        LD      SBUF,EA         ;
        LD      EA,#0EEH
        LD      SMOD,EA         ;  SIO data transfer
```



S3C7031/C7032

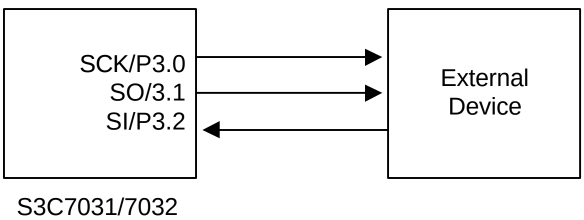
2. Use CPU clock to transfer and receive serial data at high speed:

```
        BITR    EMB
        LD      EA,#30H
        LD      PMG2,EA         ;  P3.0 / SCK and P3.1 / SO  ← Output, P3.2 / SI  ← Input
        LD      EA,TDATA        ;  TDATA address = Bank0(20H-7FH)
        LD      SBUF,EA
        LD      EA,#4FH
        LD      SMOD,EA         ;  SIO start
        BITS    IES
STEST   BTSTZ   IRQS
        JR      STEST
        LD      EA,SBUF
        SMB     0
        LD      RDATA,EA        ;  RDATA address = Bank0 (20H-7FH)
```

☞ PROGRAMMING TIP — Setting Transmit/Receive Modes for Serial I/O (Continued)

3. Transmit and receive an internal clock frequency of 4.09 kHz (at 4.19 MHz) in LSB-first mode:

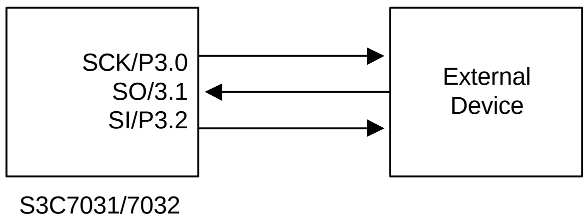
```
        BITR    EMB
        LD      EA,#30H
        LD      PMG2,EA         ; P3.0 / SCK and P3.1 / SO ← Output, P3.2/SI ← Input
        LD      EA,TDATA        ; TDATA address = Bank0 (20H-7FH)
        LD      SBUF,EA
        LD      EA,#8FH
        LD      SMOD,EA         ; SIO start
        EI
        BITS    IES
        •
        •
        •
INTS    PUSH    SB              ; Store SMB, SRB
        PUSH    EA              ; Store EA
        BITR    EMB
        LD      EA,TDATA        ; EA ← Transmit data
                                ; TDATA address = Bank0 (20H-7FH)
        XCH     EA,SBUF         ; Transmit data ↔ Receive data
        LD      RDATA,EA        ; RDATA address = Bank0 (20H-7FH)
        BITS    SMOD.3          ; SIO start
        POP     EA
        POP     SB
        IRET
```



☞ PROGRAMMING TIP — Setting Transmit/Receive Modes for Serial I/O (Concluded)

4. Transmit and receive an external clock in LSB-first mode:

```
        BITR    EMB
        LD      EA,#20H
        LD      PMG2,EA         ; P3.1 / SO ← Output, P3.0 / SCK and P3.2 / SI ←Input
        LD      EA,TDATA        ; TDATA address = Bank0 (20H-7FH)
        LD      SBUF,EA
        LD      EA,#0FH
        LD      SMOD,EA         ; SIO start
        EI
        BITS    IES
        •
        •
        •
INTS    PUSH    SB              ; Store SMB, SRB
        PUSH    EA              ; Store EA
        BITR    EMB
        LD      EA,TDATA        ; EA ← Transmit data
                                ; TDATA address = Bank0 (20H-7FH)
        XCH     EA,SBUF         ; Transmit data ↔ Receive data
        LD      RDATA,EA        ; RDATA address = Bank0 (20H-7FH)
        BITS    SMOD.3          ; SIO start
        POP     EA
        POP     SB
        IRET
```



14 ELECTRICAL DATA

OVERVIEW

In this section, information on S3C7031/7032 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
— Oscillators characteristics
— I/O capacitance
— Comparator electrical characteristics
— A.C. electrical characteristics
— Operating voltage range

Oscillation Characteristics

— System clock oscillator frequencies and stabilization time

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

Miscellaneous Timing Waveforms

— Clock timing measurement at X_{IN}

— TIO timing

— Input timing for RESET

— Input timing for external interrupts and quasi-interrupts

— Serial data transfer timing

Characteristic Curves

— I_{DD} vs Frequency

— I_{DD} vs V_{DD}

— I_{OL} vs V_{OL} (P0.0)

— I_{OL} vs V_{OL} (P1.1)

— I_{OL} vs V_{OL} (P2.0)

— I_{OH} vs V_{OH} (P0.0)

— I_{OH} vs V_{OH} (P1.1)

Table 14-1. Absolute Maximum Ratings

(T_A = 25 °C)

Parameter	Symbol	Conditions	Rating	Units
Supply Voltage	V_{DD}	–	- 0.3 to + 7.0	V
Input Voltage	V_I	All I/O ports	- 0.3 to V_{DD} + 0.3	V
Output Voltage	V_O	–	- 0.3 to V_{DD} + 0.3	V
Output Current High	I_{OH}	One I/O port active	- 5	mA
		All I/O ports active	- 15	
Output Current Low	I_{OL}	One I/O port active	25	mA
		All I/O port, total	100	
Operating Temperature	T_A	–	- 40 to + 85	°C
Storage Temperature	T_{stg}	–	- 65 to + 150	°C

Table 14-2. D.C. Electrical Characteristics

(T_A = - 40 °C to + 85 °C, V_{DD} = 2.7 V to 6.0 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Input High Voltage	$V_{IH}1$	Ports 0, 1, 2, 3, RESET	0.7 V_{DD}	–	V_{DD}	V
	$V_{IH}2$	X_{IN}, X_{OUT}	V_{DD} - 0.5	–	V_{DD}	
Input Low Voltage	$V_{IL}1$	Ports 0, 1, 2, 3, RESET	–	–	0.3 V_{DD}	V
	$V_{IL}2$	X_{IN}, X_{OUT}			0.4	
Output High Voltage	$V_{OH}1$	V_{DD} = 4.5 V to 6.0 V I_{OH} = - 3 mA Ports 0, 1, 2, 3 except P0.0	V_{DD} - 1.0	V_{DD} - 0.4	–	V
		V_{DD} = 4.5 V to 6.0 V I_{OH} = - 6 mA Ports 0, 1, 2, 3 except P0.0	V_{DD} - 2.0	V_{DD} - 0.9	–	
	$V_{OH}2$	V_{DD} = 4.5 V to 6.0 V I_{OH} = - 10 mA P0.0	V_{DD} - 2.0	–	–	
Output Low Voltage	$V_{OL}1$	V_{DD} = 4.5 V to 6.0 V I_{OL} = 25 mA Ports 0, 1, 2, 3 except P0.0	–	1.4	2.0	V
	$V_{OL}2$	V_{DD} = 4.5 V to 6.0 V I_{OL} = 50 mA P0.0	–	1.6	2.0	V

Table 14-2. D.C. Electrical Characteristics (Continued)

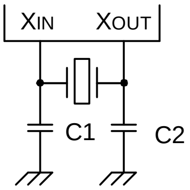
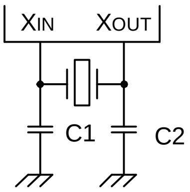
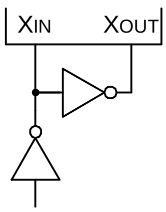
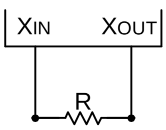
(T_A = - 40 °C to + 85 °C, V_{DD} = 2.7 V to 6.0 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Input High Leakage Current	$I_{LIH}1$	$V_{IN} = V_{DD}$ All input pins except $I_{LIH}2$	–	–	3	µA
	$I_{LIH}2$	$V_{IN} = V_{DD}$ X_{IN}, X_{OUT}		15	20	
Input Low Leakage Current	$I_{LIL}1$	V_{IN} = 0 V All input pins except $I_{LIL}2$	–	–	- 3	µA
	$I_{LIL}2$	V_{IN} = 0 V X_{IN}, X_{OUT}		-15	- 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_L	V_{IN} = 0 V; V_{DD} = 5 V - 10 % Ports 0, 1, 2, 3	15	50	80	KΩ
		V_{IN} = 0 V; V_{DD} = 3 V - 10 % Ports 0, 1, 2, 3	30	100	200	
Supply Current (2)	$I_{DD}1$	V_{DD} = 5 V ± 10 % (2) 4.19 MHz crystal oscillator C1 = C2 = 22 pF	–	1.7	8.0	mA
		V_{DD} = 3 V ± 10 % (3) 4.19 MHz crystal oscillator C1 = C2 = 22 pF		0.6	1.2	
	$I_{DD}2$	Idle mode; V_{DD} = 5 V ± 10 % 4.19 MHz crystal oscillator C1 = C2 = 22 pF	–	0.5	1.8	mA
		Idle mode; V_{DD} = 3 V ± 10 % 4.19 MHz crystal oscillator C1 = C2 = 22 pF		0.2	1.0	
	$I_{DD}3$	Stop mode V_{DD} = 5 V - 10 %		0.2	5	µA
		Stop mode V_{DD} = 3 V - 10 %		0.1	3	

NOTES:
1. D.C. electrical values for Supply Current ($I_{DD}1$ to $I_{DD}3$) do not include current drawn through internal pull-up resistors.
2. For high-speed controller operation, set the PCON register to 0011B.
3. For low-speed controller operation, set the PCON register to 0000B.

Table 14-3. Oscillators Characteristics

(T_A = - 40 °C to + 85 °C, V_{DD} = 5 V)

Oscillator	Clock Configuration	Parameter	Test Condition	Min	Typ	Max	Units
Ceramic Oscillator		Oscillation frequency (1)	–	0.4	–	4.5	MHz
		Stabilization time (2)	After V_{DD} reaches the minimum level of its variable range	–	–	4	ms
Crystal Oscillator		Oscillation frequency (1)	–	0.4	4.19	4.5	MHz
		Stabilization time (2)	V_{DD} = 2.7 V to 4.5 V	–	–	30	ms
			V_{DD} = 4.5 V to 6.0 V	–	–	10	
External Clock		X_{IN} input frequency (1)	–	0.4	–	4.5	MHz
		X_{IN} input high and low level width (t_{XH}, t_{XL})	–	111	–	1250	ns
RC Oscillator		Frequency	V_{DD} = 5 V	0.6	1	2.3	MHz
			V_{DD} = 3 V	0.4	0.8	1.5	

NOTES:

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

Table 14-4. 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 14-5. Comparator Electrical Characteristics

(T_A = - 40 °C to + 85 °C, V_{DD} = 4.0 V to 6.0 V)

<table>
<tr><th colspan="2">Parameter</th><th>Symbol</th><th>Condition</th><th>Min</th><th>Typ</th><th>Max</th><th>Units</th></tr>
<tr><td colspan="2">Input Voltage Range</td><td>–</td><td>–</td><td>0</td><td>–</td><td>V_{DD}</td><td>V</td></tr>
<tr><td colspan="2">Reference Voltage Range</td><td>V_{REF}</td><td>–</td><td>0</td><td>–</td><td>V_{DD}</td><td>V</td></tr>
<tr><td rowspan="2">Input Voltage Accuracy</td><td>Internal Reference</td><td>$V_{CIN}1$</td><td>–</td><td>–</td><td>–</td><td>- 150</td><td rowspan="2">mV</td></tr>
<tr><td>External Reference</td><td>$V_{CIN}2$</td><td>–</td><td>–</td><td>–</td><td>- 50</td></tr>
<tr><td colspan="2">Input Leakage Current</td><td>I_{CIN}, I_{REF}</td><td>–</td><td>- 3</td><td>–</td><td>3</td><td>μA</td></tr>
</table>

Table 14-6. A.C. Electrical Characteristics

(T_A = - 40 °C to + 85 °C, V_{DD} = 2.7 V to 6.0 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Instruction Cycle Time	t_{CY}	V_{DD} = 4.5 V to 6.0 V	0.95	–	64	µs
		V_{DD} = 2.7 V to 4.5 V	3.8			
TIO Input Frequency	f_{TI}	V_{DD} = 4.5 V to 6.0 V	0	–	1	MHz
		V_{DD} = 2.7 V to 4.5 V			275	kHz
TIO Input High, Low Width	t_{TIH}, t_{TIL}	V_{DD} = 4.5 V to 6.0 V	0.48	–	–	µs
		V_{DD} = 2.7 V to 4.5 V	1.8			
SCK Cycle Time	t_{KCY}	V_{DD} = 4.5 V to 6.0 V; Input	800	–	–	ns
		V_{DD} = 4.5 V to 6.0 V; Output	950			
		V_{DD} = 2.7 V to 4.5 V; Input	3200			
		V_{DD} = 2.7 V to 4.5 V; Output	3800			
SCK High, Low Width	t_{KH}, t_{KL}	V_{DD} = 4.5 V to 6.0 V; Input	400	–	–	ns
		V_{DD} = 4.5 V to 6.0 V; Output	t_{KCY}/2-50			
		V_{DD} = 2.7 V to 4.5 V; Input	1600			
		V_{DD} = 2.7 V to 4.5 V; Output	t_{KCY}/2-50			
SI Setup Time to SCK High	t_{SIK}	Input	100	–	–	ns
		Output	150			
SI Hold Time to SCK High	t_{KSI}	Input	400	–	–	ns
		Output	400			
Output Delay for SCK to SO	t_{KSO}	V_{DD} = 4.5 V to 6.0 V; Input	–	–	300	ns
		V_{DD} = 4.5 V to 6.0 V; Output			250	
		V_{DD} = 2.7 V to 4.5 V; Input			1000	
		V_{DD} = 2.7 V to 4.5 V; Output			1000	
Interrupt Input High, Low Width	t_{INTH}, t_{INTL}	INT1, KS0-KS3	10	–	–	µs
RESET Input Low Width	t_{RSL}	Input	10	–	–	µs

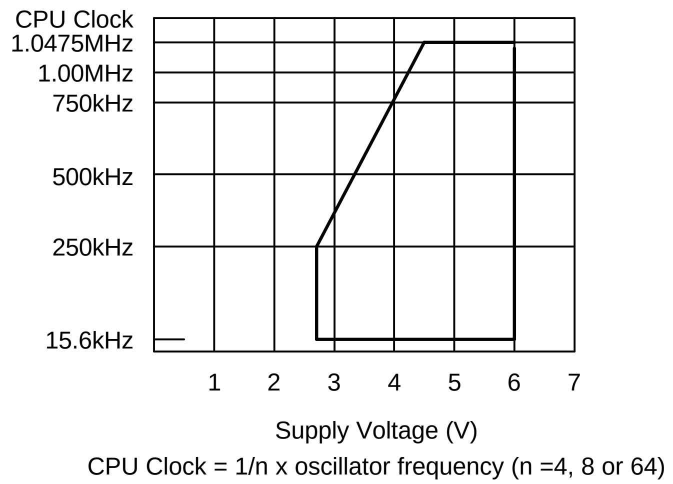


Figure 14-1. Standard Operating Voltage Range

Table 14-7. RAM Data Retention Supply Voltage in Stop Mode

(T_A = - 40 °C to + 85 °C)

Parameter	Symbol	Condition	Min	Typ	Max	Units
Data Retention Supply voltage	V_{DDDR}	–	2.0	—	6.0	V
Data Retention Supply Current	I_{DDDR}	V_{DDDR} = 2.0 V	–	0.1	10	µA
Release Signal Set Time	t_{SREL}	–	0	—	–	µs
Oscillation 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

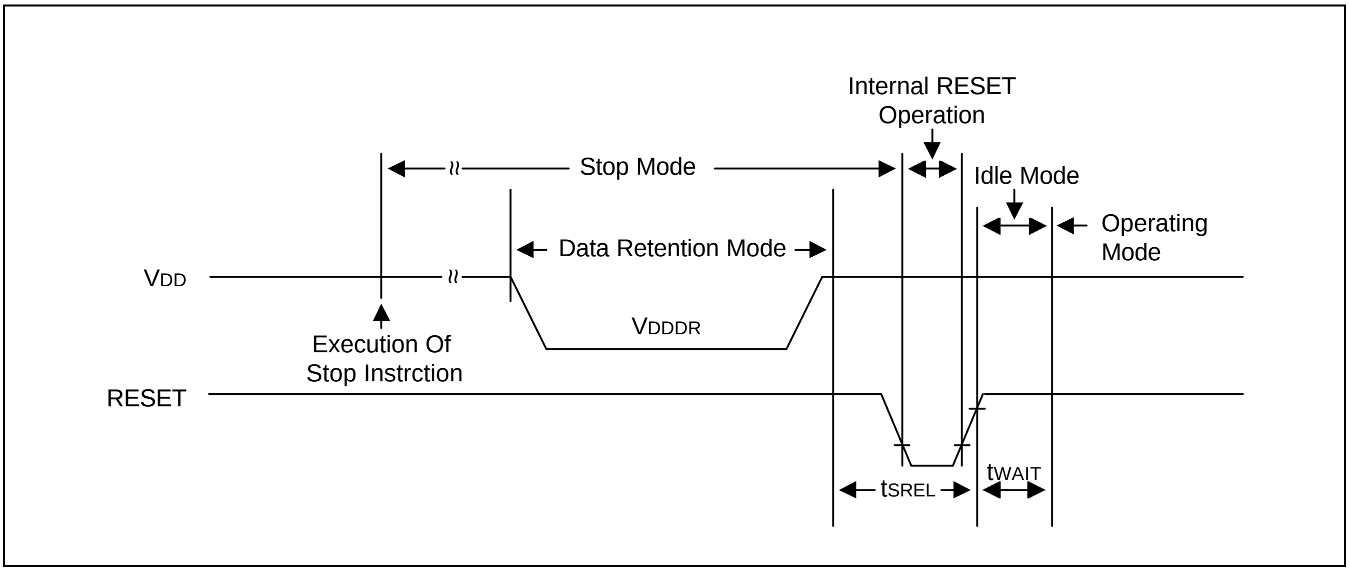


Figure 14-2.Stop Mode Release Timing When Initiated By RESET

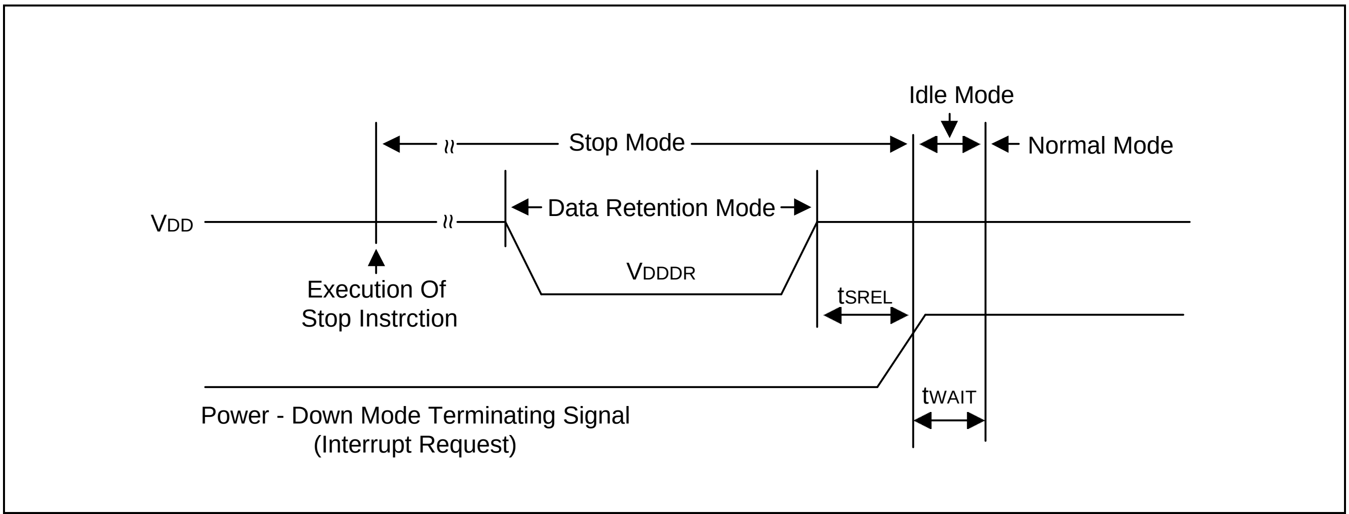


Figure 14-3. Stop Mode Release Timing When Initiated By Interrupt Request

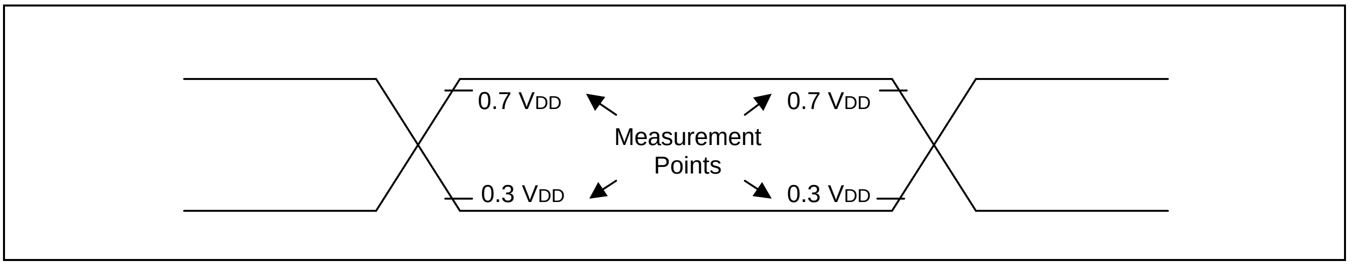


Figure 14-4. A.C. Timing Measure Points (Except for X_{IN})

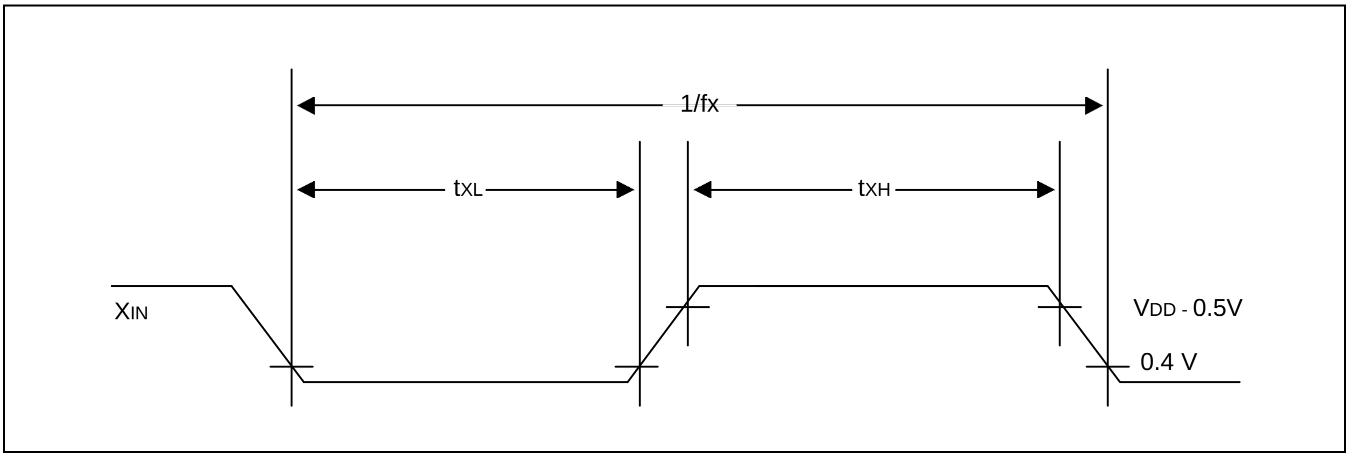


Figure 14-5. Clock Timing Measurement at X_{IN}

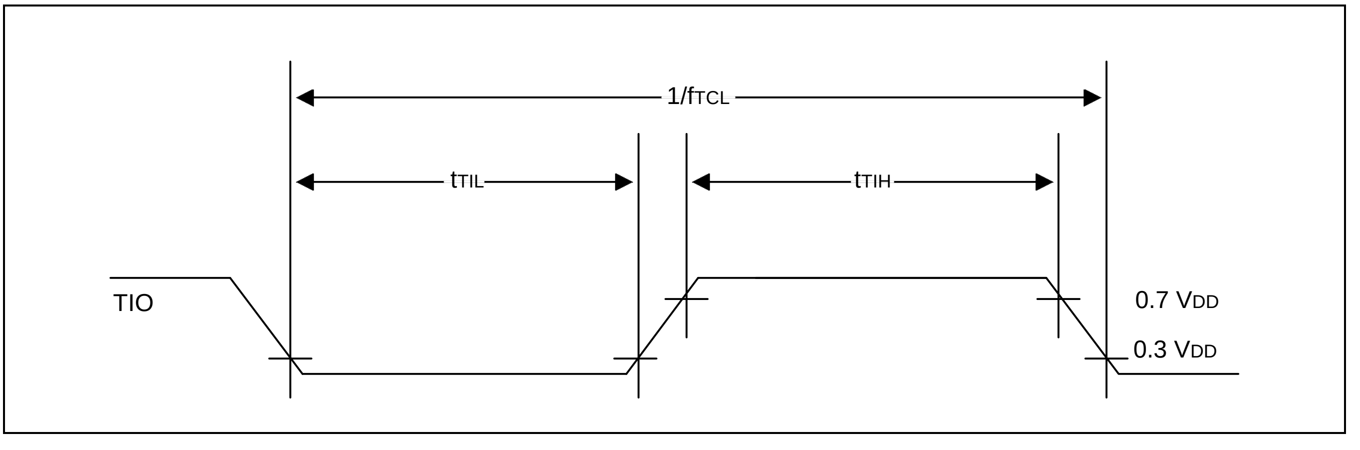


Figure 14-6. TIO Timing

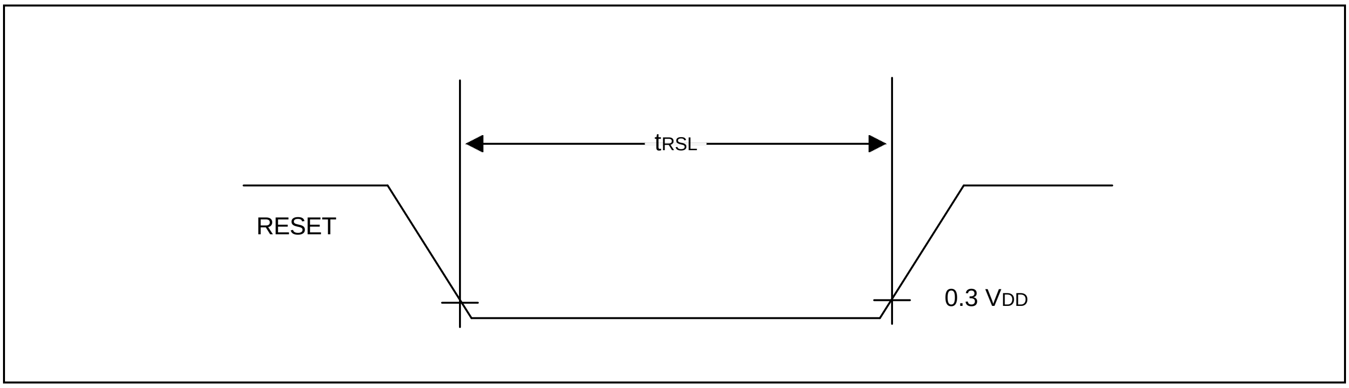


Figure 14-7. Input Timing for RESET Signal

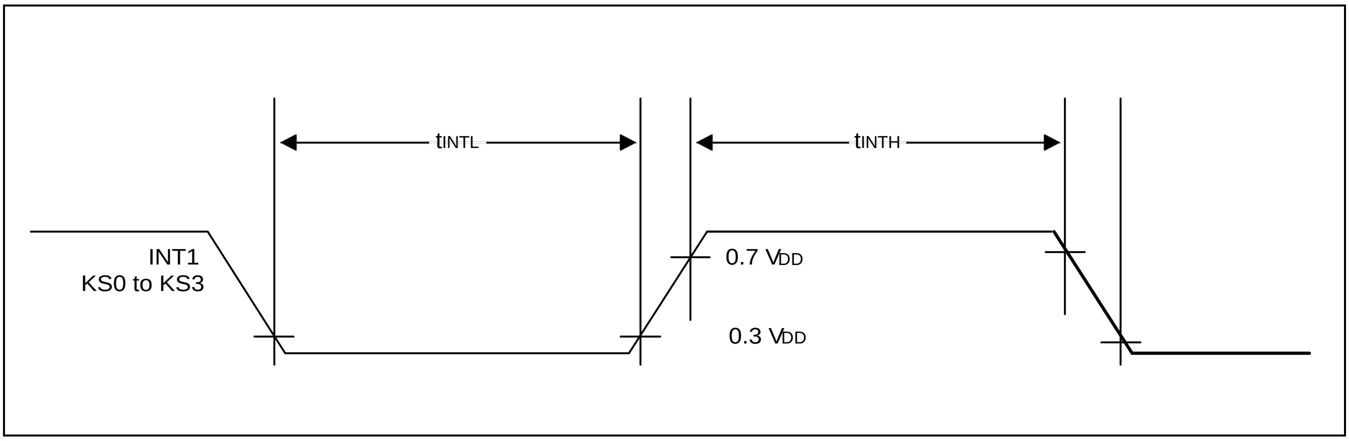


Figure 14-8. Input Timing for External Interrupts

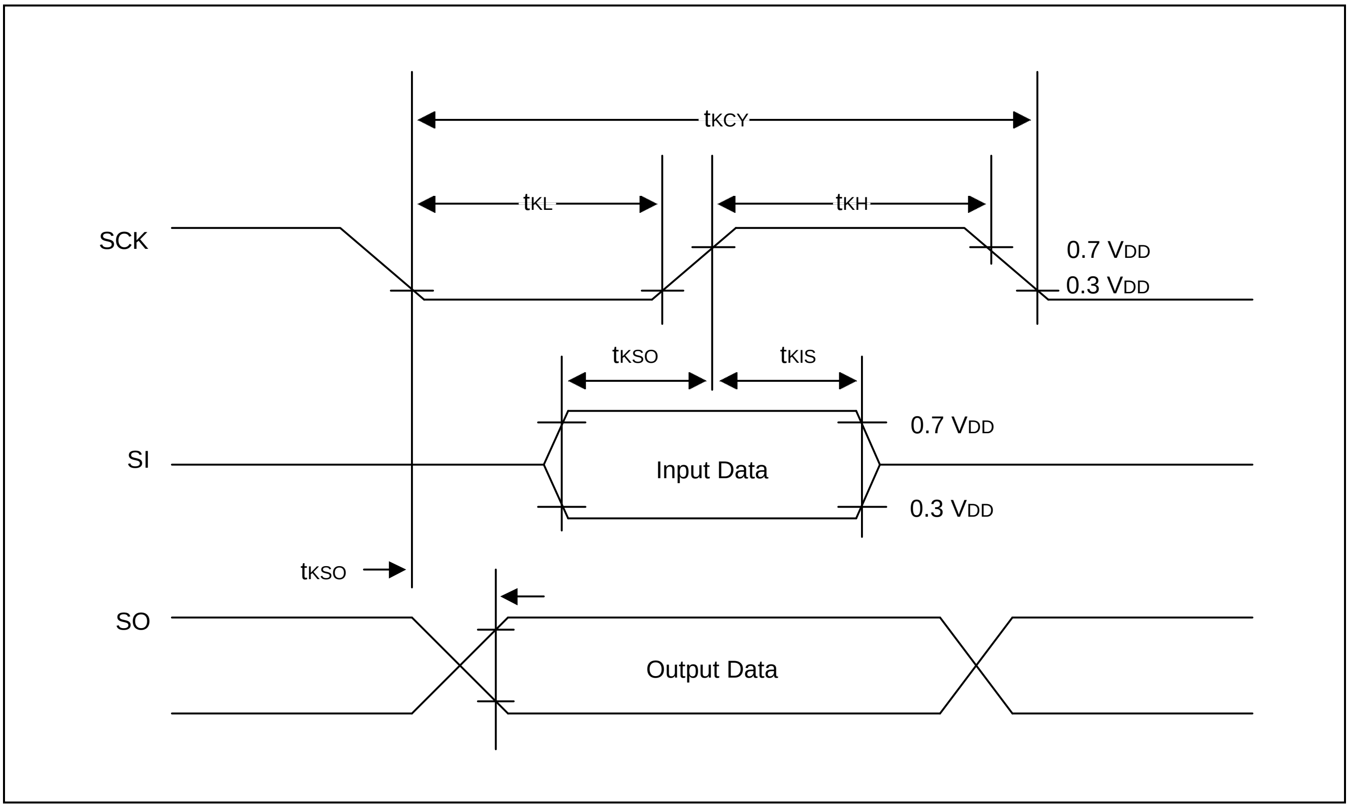


Figure 14-9. Serial Data Transfer Timing

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.

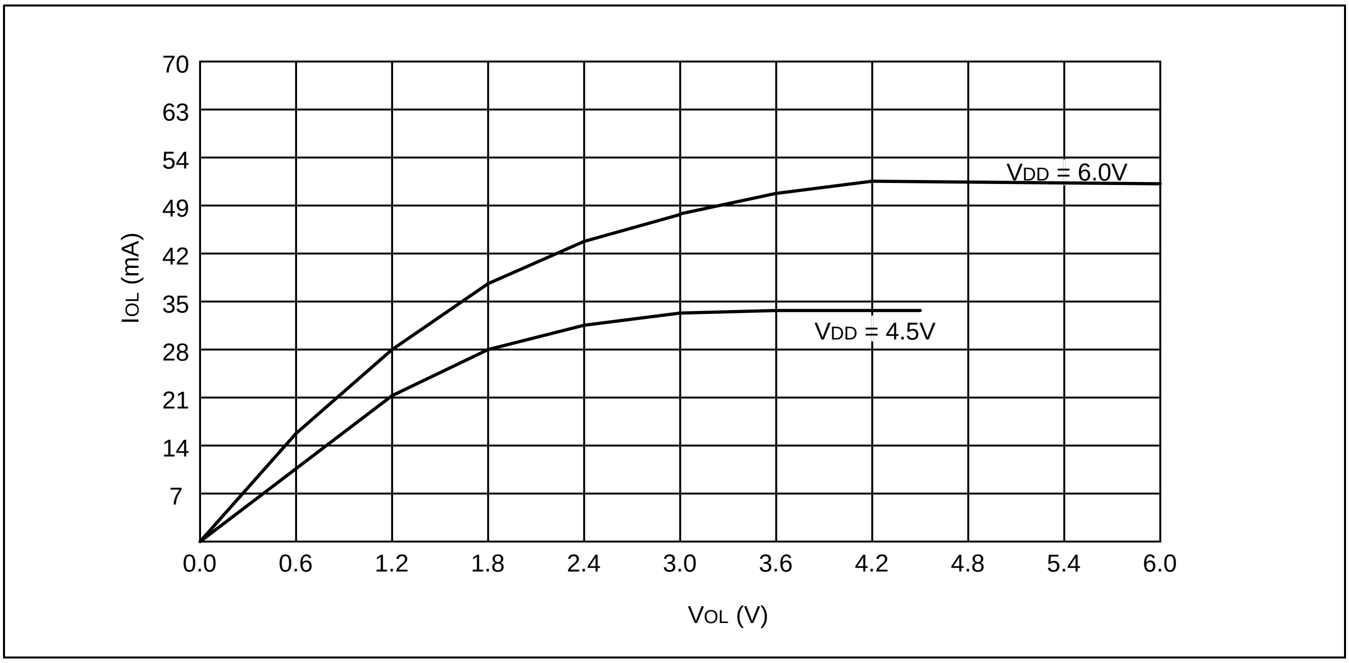


Figure 14-10. I_{OL} vs. V_{OL} (Port 0,1,2,3)

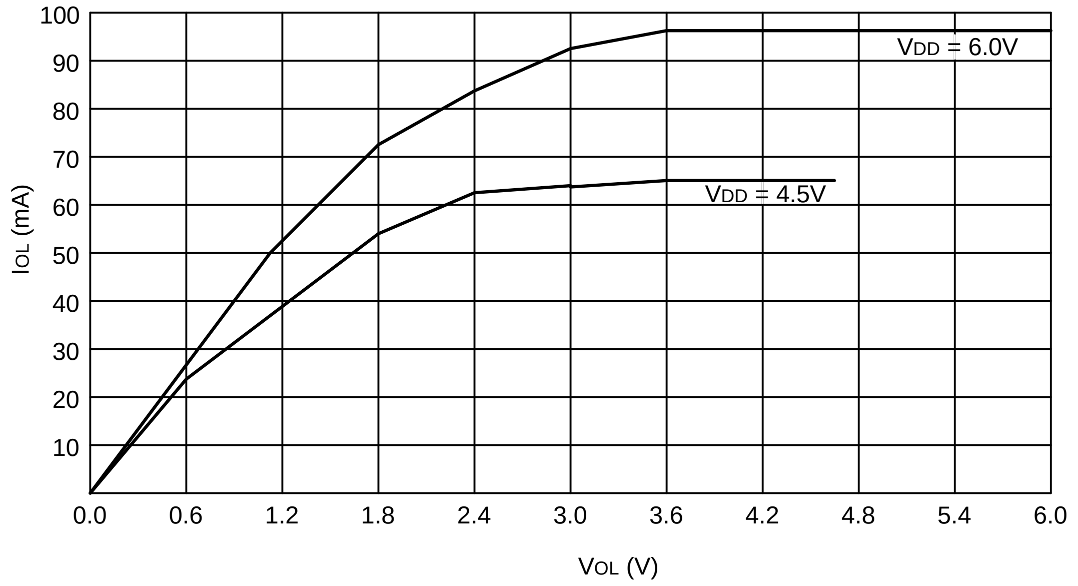


Figure 14-11. I_{OL} vs. V_{OL} (Port 0.0)

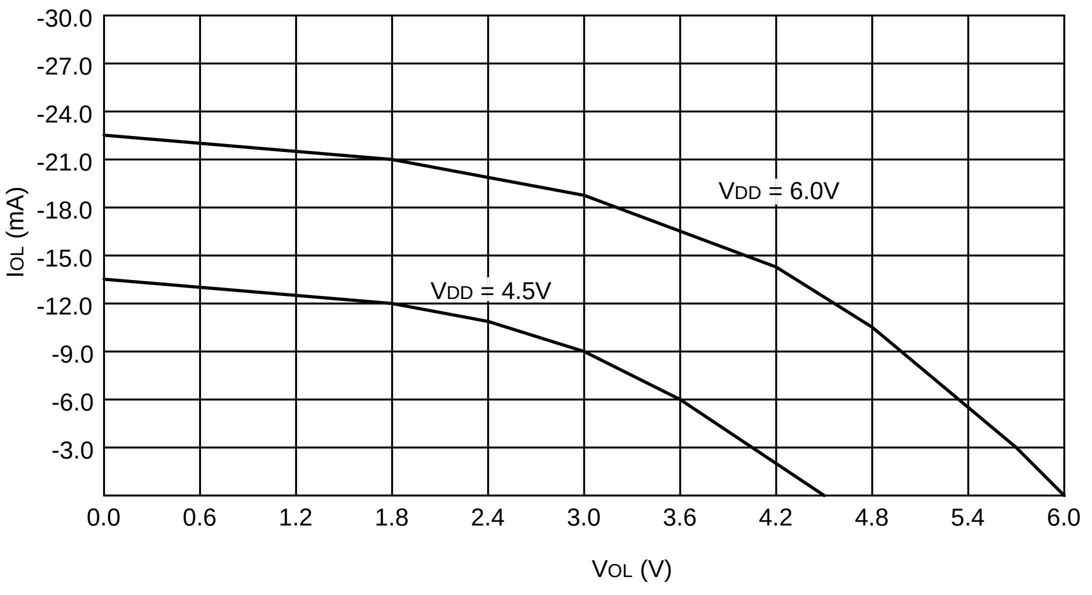


Figure 14-12. I_{OH} vs. V_{OH} (Port 0,1,2,3except P0.0)

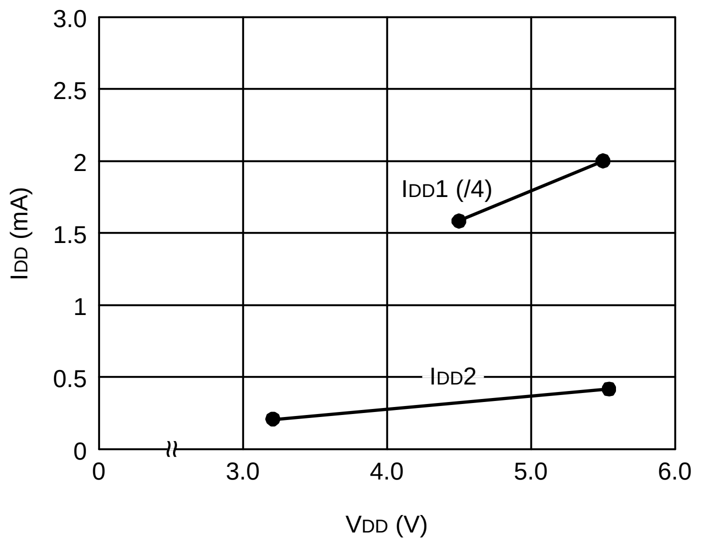


Figure 14-13. I_{DD} vs. V_{DD}

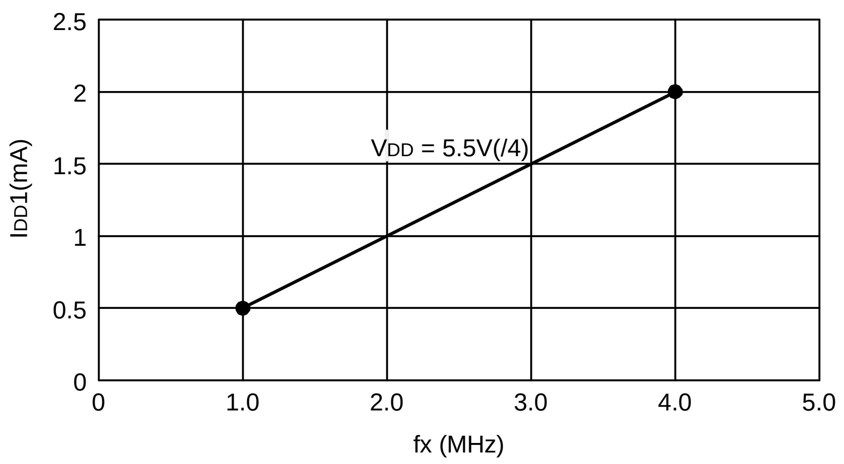


Figure 14-14. I_{DD} vs. Frequency

15 MECHANICAL DATA

This section contains the following information about the device package:

— A 20-pin DIP package is available for S3C7031/7032.

— A 20-pin SOP package is available for S3C7031/7032.

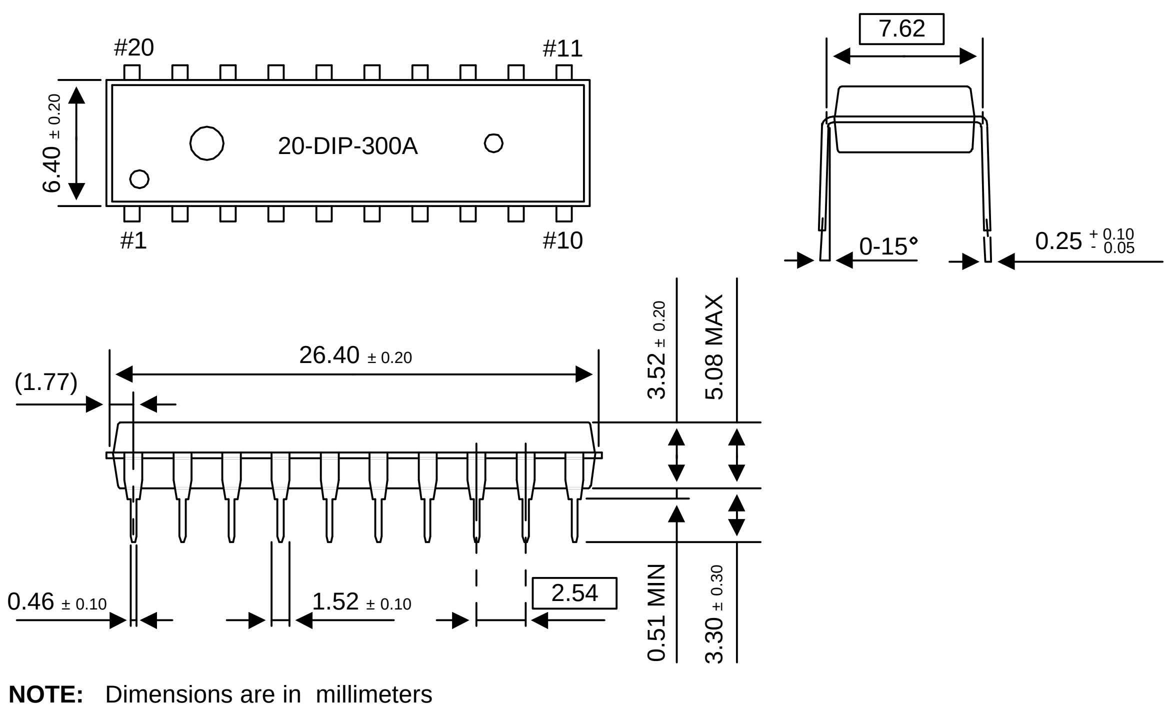


Figure 15-1. 20-pin DIP-300A Package Dimensions

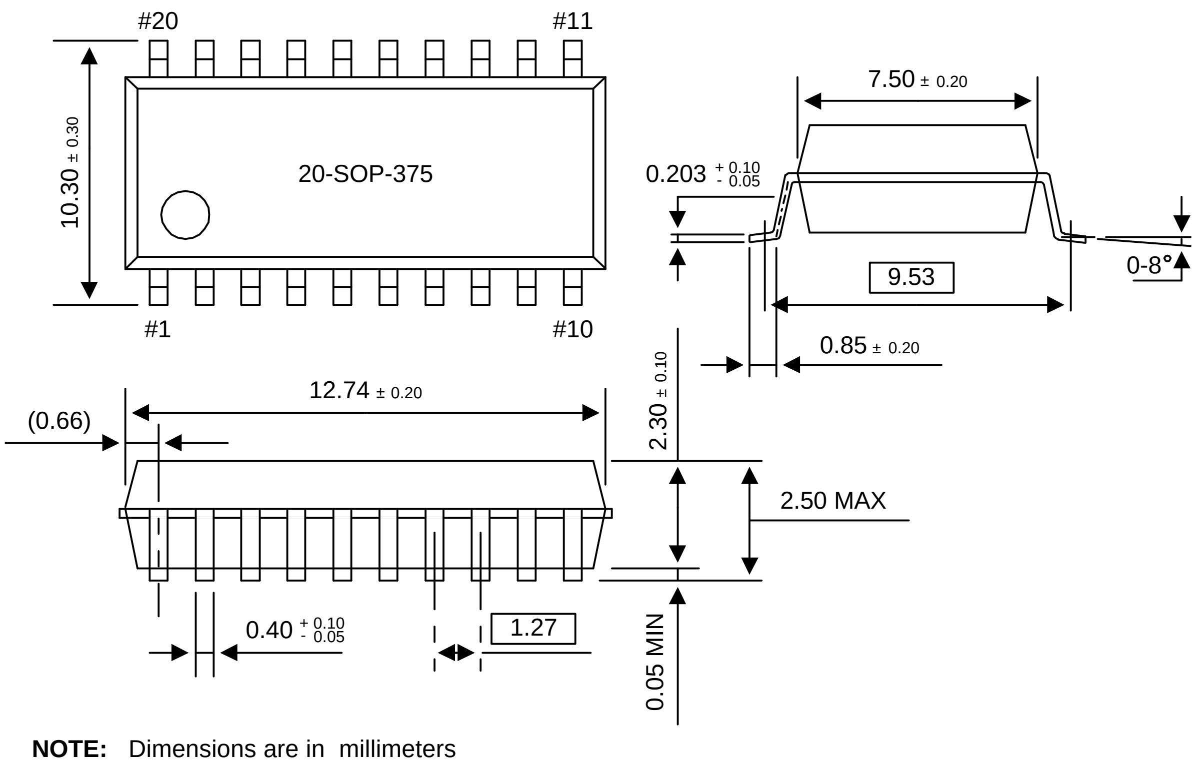


Figure 15-2. 20-pin SOP-375 Package Dimensions

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 S3C7031/7032 microcontroller and OTP programmer (Gang) are now available.

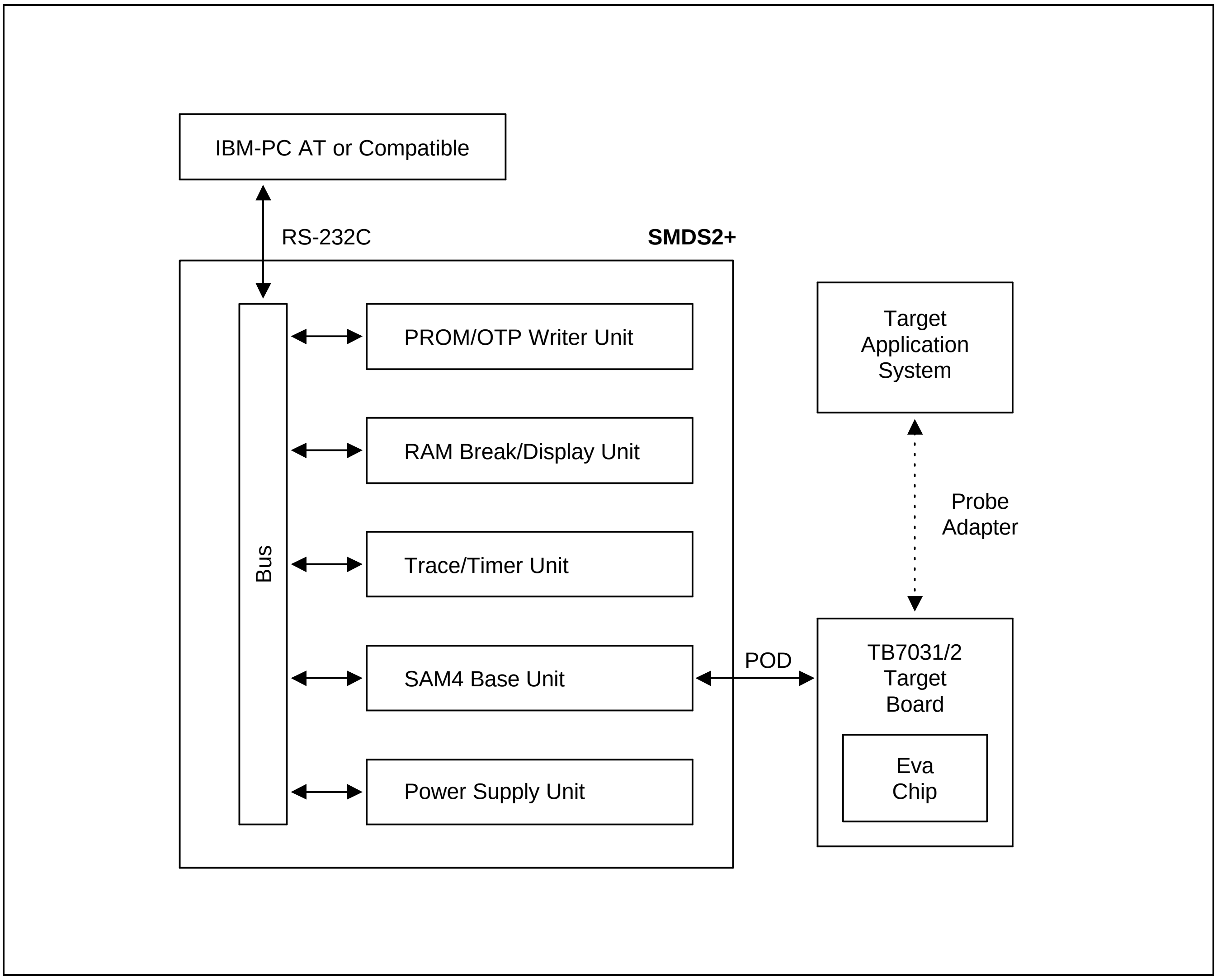


Figure 16-1. SMDS Product Configuration (SMDS2+)

TB7031/2 TARGET BOARD

The TB7031/2 target board is used for the S3C7031/7032 microcontroller. It is supported by the SMDS2+ development system.

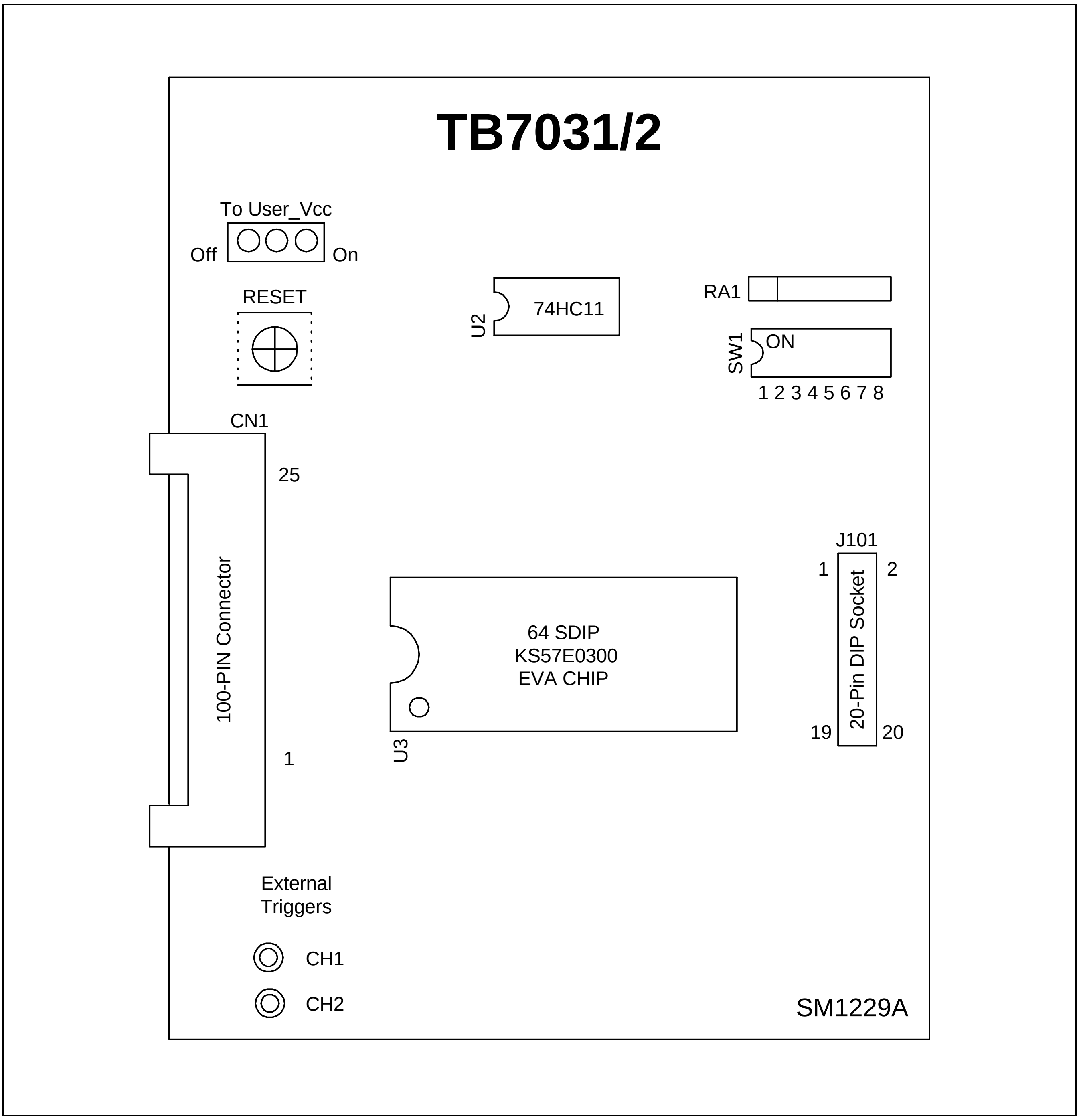


Figure 16-2. TB7301/2A Target Board Configuration

Table 16-1. Power Selection Settings for TB7301/2A

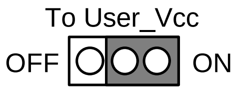
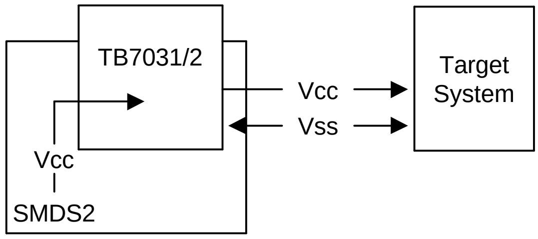
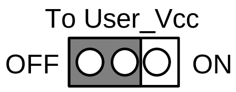
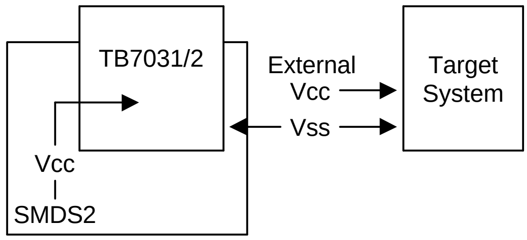
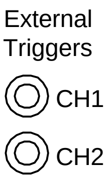
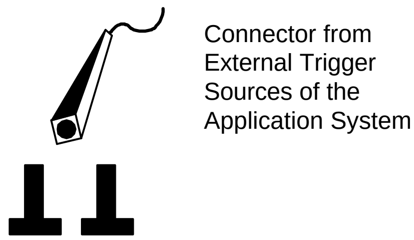
'To User_Vcc' Settings	Operating Mode	Comments
		The SMDS2/SMDS2+ supplies V_{CC} to the target board (evaluation chip) and the target system.
		The SMDS2/SMDS2+ supplies V_{CC} only to the target board (evaluation chip). The target system must have its own power supply.

Table 16-2. 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.

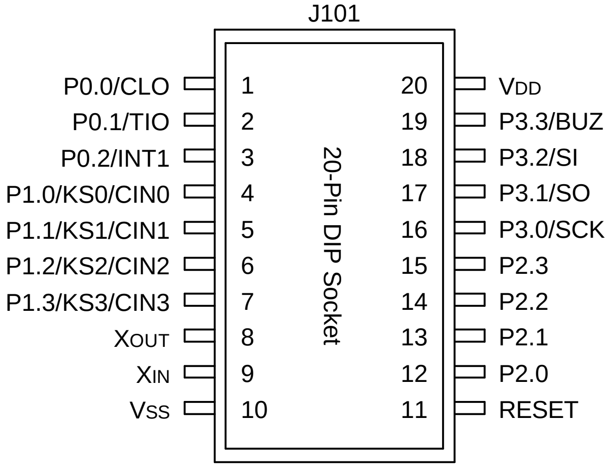


Figure 16-3. 20-pin DIP Socket for TB7031/2

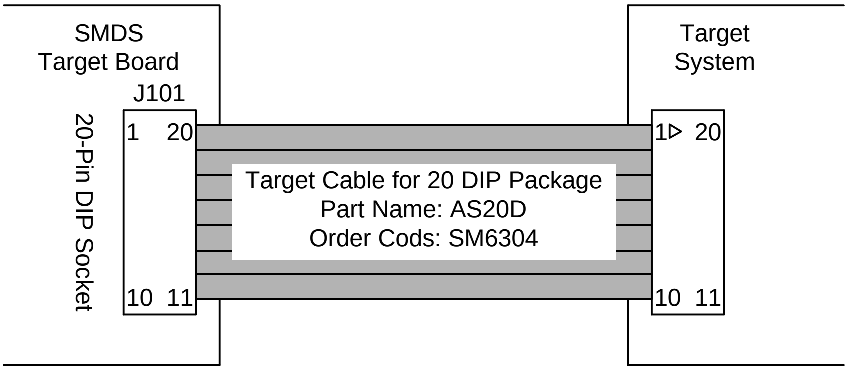


Figure 16-4. TB7031/2 Adapter Cable for 20-DIP Package (S3C7031/7032)