Product Overview

Address Spaces

Addressing Modes

Memory Map

1 PRODUCT OVERVIEW

OVERVIEW

The KS57C3204 single-chip CMOS microcontroller has been designed for high performance using Samsung's newest 4-bit CPU core, SAM47 (Samsung Arrangeable Microcontrollers).

With features such as LCD direct drive capability, 4-channel A/D converter, 24-bit AM/FM frequency counter and watch timer, the KS57C3204 offers an excellent design solution for a wide variety of applications that require LCD functions and audio applications.

Up to 32 pins of the 64-pin QFP package, it can be dedicated to I/O. Five vectored interrupts provide fast response to internal and external events. In addition, the KS57C3204 's advanced CMOS technology provides for low power consumption and a wide operating voltage range.

OTP

The KS57C3204 microcontroller is also available in OTP (One Time Programmable) version, KS57P3204 . The KS57P3204 microcontroller has an on-chip 4-Kbyte one-time-programmable EPROM instead of masked ROM. The KS57P3204 is comparable to KS57C3204, both in function and in pin configuration.

FEATURES

Memory

— 256 × 4-bit RAM

— 4096 × 8-bit ROM

I/O Pins

— Input only: 8 pins

— I/O: 16 pins

— Output only: 8 pins sharing with segment driver outputs

LCD Controller/Driver

— Maximum 14-digit LCD direct drive capability

— 28 segment and 4 common pins

— Display modes: Static, 1/2 duty (1/2 bias)
1/3 duty (1/2 or 1/3 bias), 1/4 duty (1/3 bias)

— Internal resistor circuit for LCD bias

8-Bit Basic Timer

— Programmable interval timer

— Watchdog timer

8-Bit Timer

— Programmable 8-bit timer

Watch Timer

— Real-time and interval time measurement

— Four frequency outputs to BUZ pin

— Clock source generation for LCD

24-Bit Frequency Counter (FC)

— Level = 300mVpp (Min.)

— AMF input range = 0.5 MHz to 10 MHz

— FMF input range = 30 MHz to 150 MHz

A/D Converter

— 4-channels with 8-bit resolution

— 17 µs (Min.) conversion speed

Bit Sequential Carrier

— Support 16-bit serial data transfer in arbitrary format

Interrupts

— Two internal vectored interrupts

— Three external vectored interrupts

— Two quasi-interrupts

Memory-Mapped I/O Structure

— Data memory bank 15

Two Power-Down Modes

— Idle mode (only CPU clock stops)

— Stop mode (main system clock stops)

— Subsystem clock stops

Oscillation Sources

— Crystal, ceramic, or RC for main system clock

— Crystal or external oscillator for subsystem clock

— Main system clock frequency: 4.19 MHz (typical)

— Subsystem clock frequency: 32.768 kHz

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

Instruction Execution Times

— 0.95, 1.91, 15.3 µs at 4.19 MHz (main)

— 122 µs at 32.768 kHz (subsystem)

Operating Temperature

— – 40 °C to 85 °C

Operating Voltage Range

— 1.8 V to 5.5 V at 3 MHz

— 3.0 V to 5.5 V at FC mode

Package Type

— 64-pin QFP

BLOCK DIAGRAM

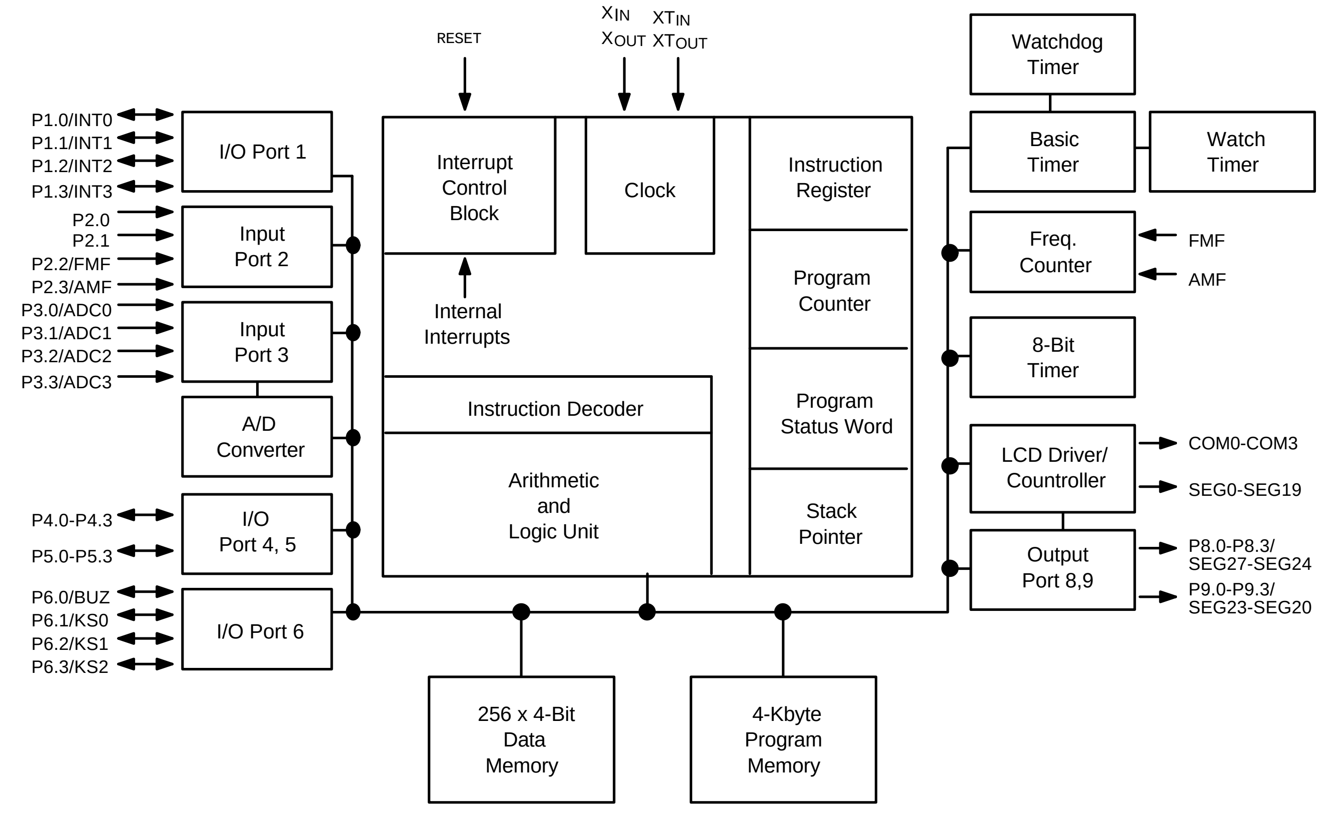


Figure 1-1. KS57C3204 Simplified Block Diagram

PIN ASSIGNMENTS

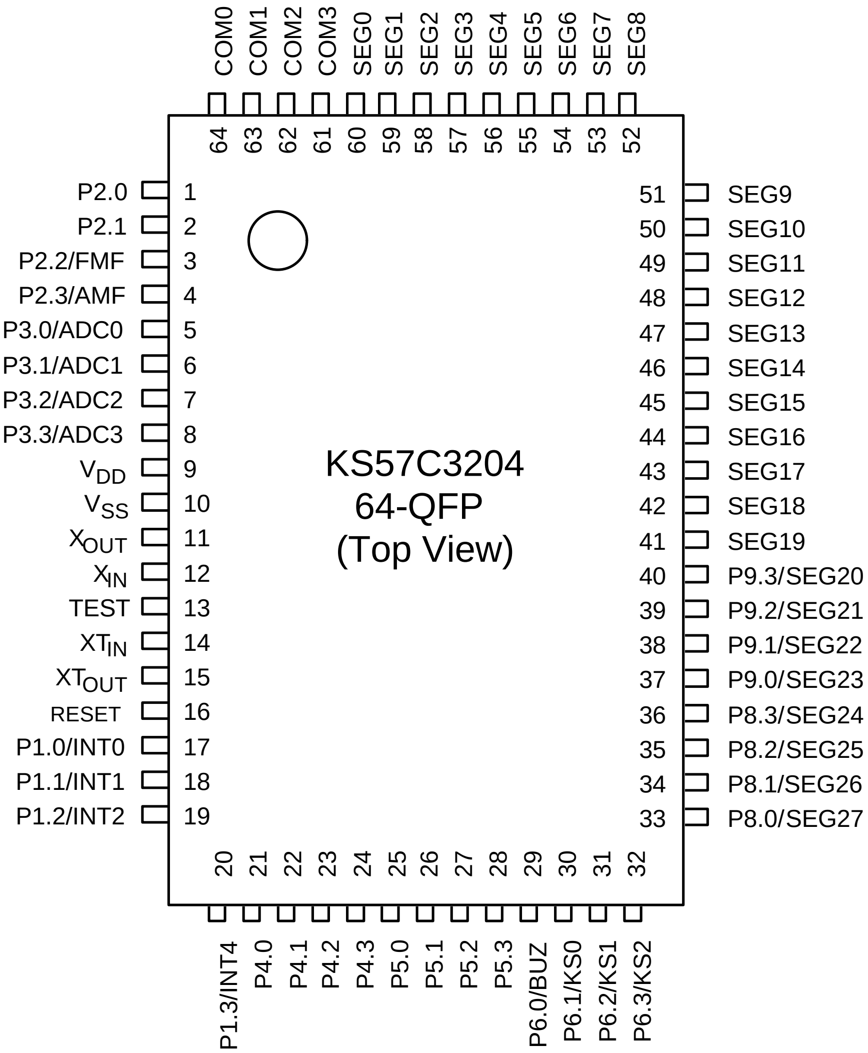


Figure 1-2. KS57C3204 64-QFP Pin Assignment

PIN DESCRIPTIONS

Table 1-1. KS57C3204 Pin Descriptions

Pin Name	Pin Type	Description	Number	Share Pin	Reset Value	Circuit Type
P1.0 P1.1 P1.2 P1.3	I/O	4-bit I/O port. 1-bit or 4-bit read, write, and test are possible. Each pin can be specified as input or output port. Pull-up resistors can be configured by software.	17 18 19 20	INT0 INT1 INT2 INT4	Input	D-4
P2.0 P2.1 P2.2 P2.3	I	4-bit input port. 1-bit and 4-bit read and test are possible. Pull-up resistors can be configured by software.	1 2 3 4	– – FMF AMF	Input	A-4 A-4 B-4 B-4
P3.0 P3.1 P3.2 P3.3	I	4-bit input port. 1-bit and 4-bit read and test are possible Pull-up resistors can be configured by software.	5 6 7 8	ADC0 ADC1 ADC2 ADC3	Input	F-13
P4.0–P4.3 P5.0–P5.3	I/O	4-bit I/O ports. N-channel open-drain output up to 5 V. 1-bit and 4-bit read, write, and test are possible. Ports 4 and 5 can be paired to support 8-bit data. Pull-up resistors can be configured by software.	21–24 25–28	– –	Input	E-2
P6.0 P6.1 P6.2 P6.3	I/O	1-bit and 4-bit read, write, and test are possible. Each pin can be specified as input or output port. Pull-up resistors can be configured by software.	29 30 31 32	BUZ KS0 KS1 KS2	Input	D-2 D-4 D-4 D-4
SEG0–SEG19	O	LCD segment signal output	60–41	–	Output	H-16
P8.0–P8.3 P9.0–P9.3	O	4-bit output ports. 1-bit and 4-bit write and test are possible. Ports 8 and 9 can be paired to support 8-bit data.	33–36 37–40	SEG27– SEG20	Output	H-16
COM0–COM3	O	LCD common signal output	64–61	–	Output	H-16
V_{DD}	–	Main power supply	9	–	–	–
V_{SS}	–	Main ground	10	–	–	–
X_{OUT}, X_{IN}	–	Crystal, ceramic, or RC oscillator pins for main system clock. (For external clock input, use X_{IN} and input X_{IN}'s reverse phase to X_{OUT})	11,12	–	–	–
XT_{OUT}, XT_{IN}	–	Crystal oscillator pin for a subsystem clock. (For external clock input, use XT_{IN} and input XT_{IN}'s reverse phase to XT_{OUT})	15,14	–	–	–

Table 1-1. KS57P3204 Pin Descriptions (Continued)

Pin Name	Pin Type	Description	Number	Share Pin	Reset Value	Circuit Type
SEG20–SEG27	O	LCD segment signal output	40–33	P9.0–P9.3 P8.0–P8.3	Output	H-16
ADC0–ADC3	I	ADC input ports	5–8	P3.0–P3.3	Input	F-13
FMF AMF	I	External FM/AM frequency inputs	3 4	P2.2 P2.3	Input	B-4
INT4	I	External interrupt input with detection of rising or falling edges.	20	P1.3	Input	A-4
INT2	I	Quasi-interrupt with detection of rising edge signals.	19	P1.2	Input	A-4
INT1 INT0	I	External interrupt. The triggering edges for INT0 and INT1 are able to be selected. Only INT0 is synchronized with the system clock.	18 17	P1.1 P1.0	Input	A-4
BUZ	O	2, 4, 8, or 16 kHz frequency output for buzzer sound with 4.19 MHz main system clock.	29	P6.0	Input	D-2
KS0–KS2	I	Quasi-interrupt input with falling edge detection.	30–32	P6.1–P6.3	Input	D-4
RESET	I	System reset signal	16	–	Input	B
TEST	–	System test pin(must be connected to $V_{SS)}$	13	–	–	–

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

PIN CIRCUIT DIAGRAMS

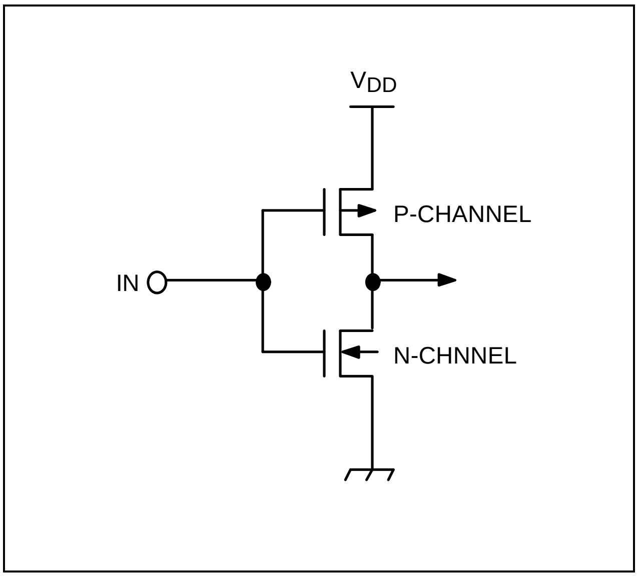


Figure 1-3. Pin Circuit Type A

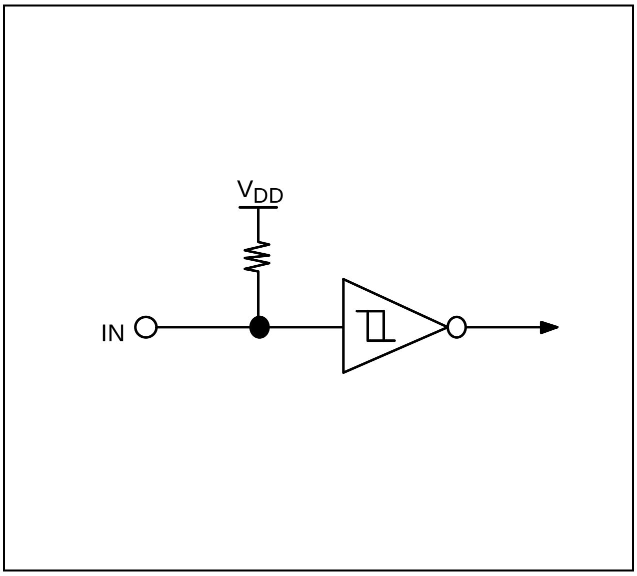


Figure 1-5. Pin Circuit Type B

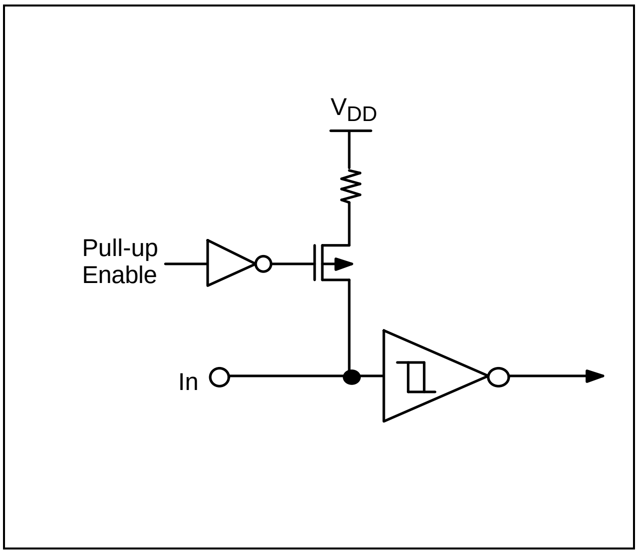


Figure 1-4. Pin Circuit Type A-4

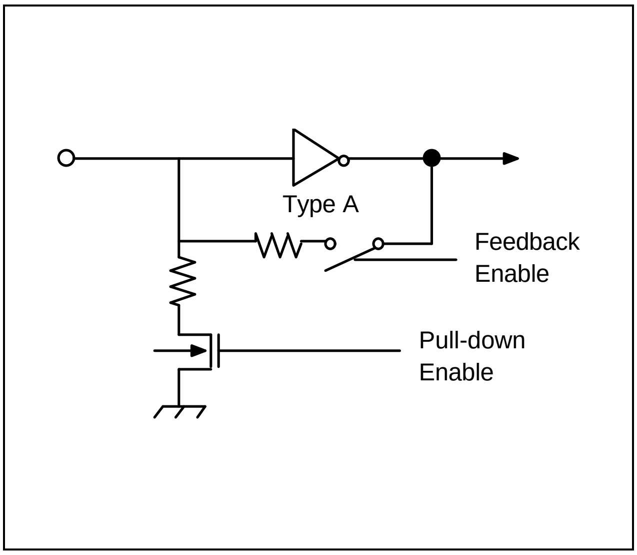


Figure 1-6. Pin Circuit Type B-4

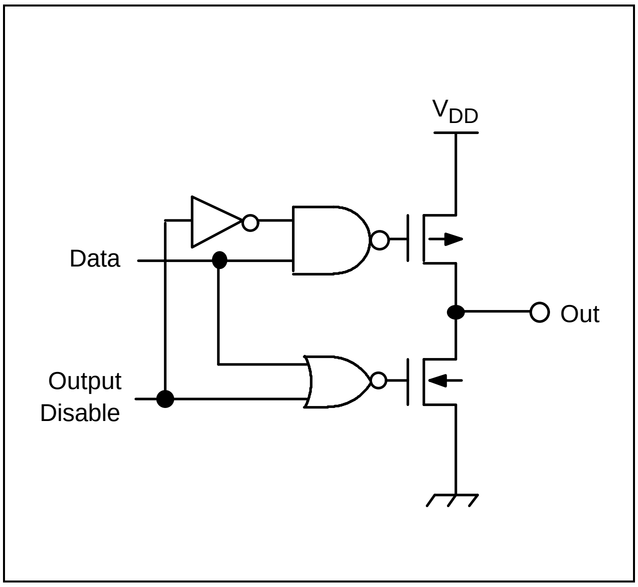


Figure 1-7. Pin Circuit Type C

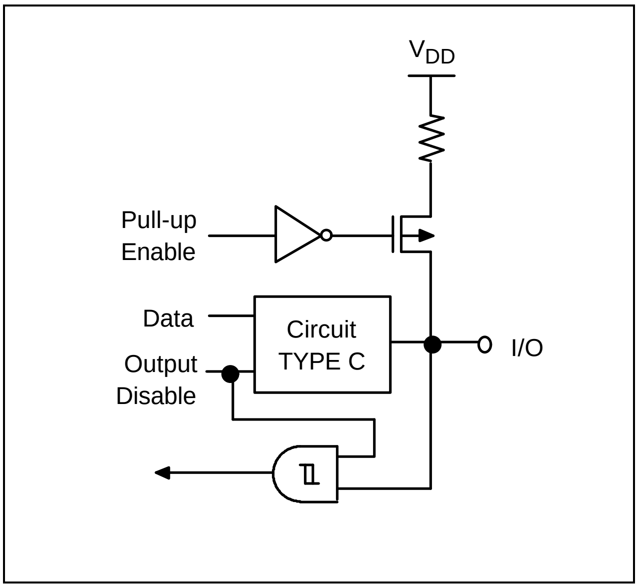


Figure 1-9. Pin Circuit Type D-4

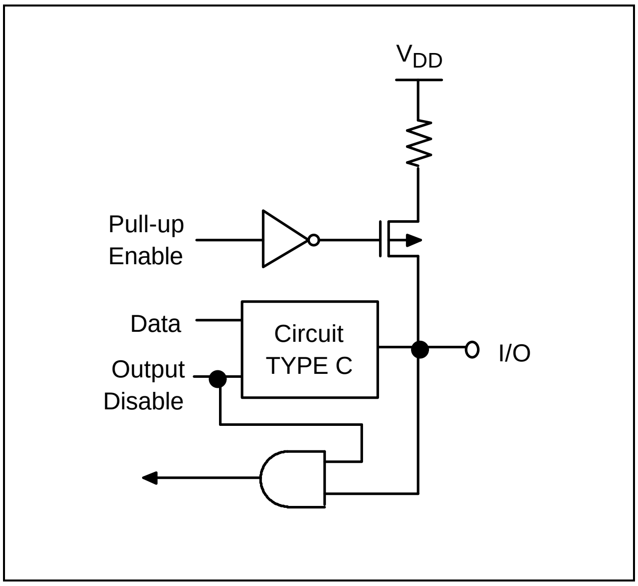


Figure 1-8. Pin Circuit Type D-2

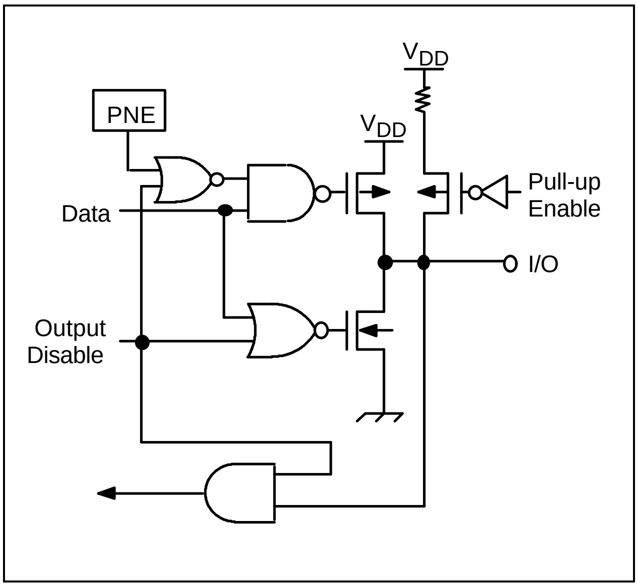


Figure 1-10. Pin Circuit Type E-2

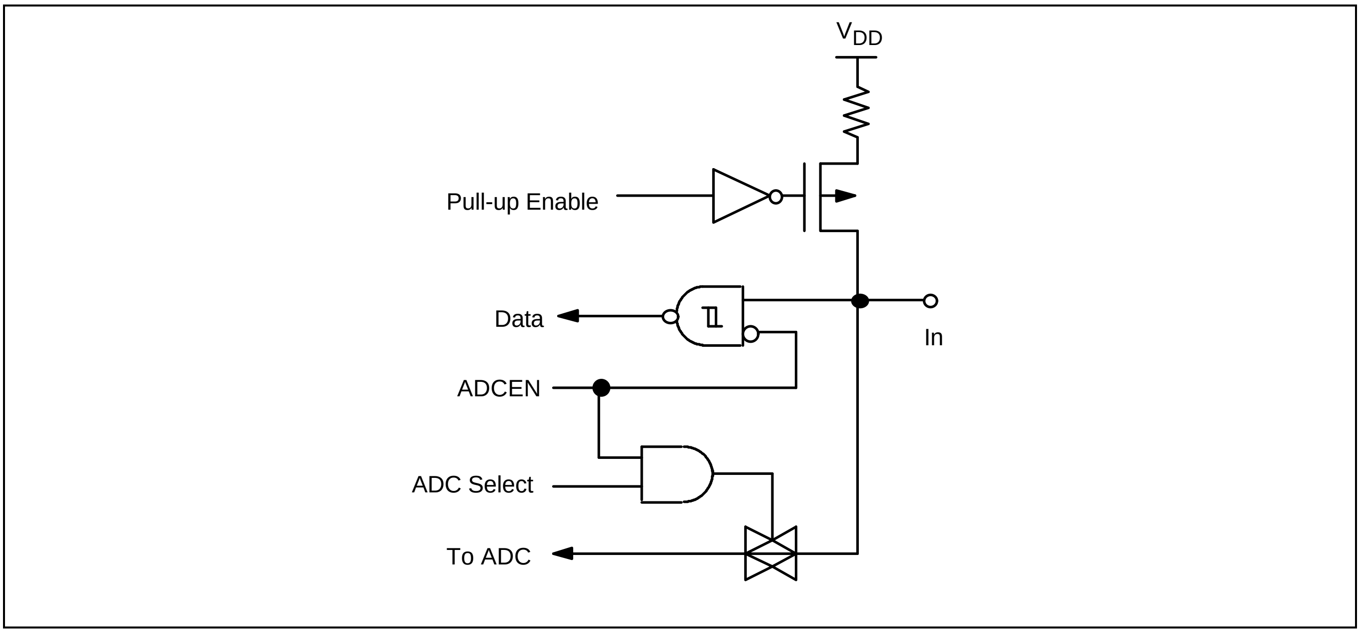


Figure 1-11. Pin Circuit Type F-13

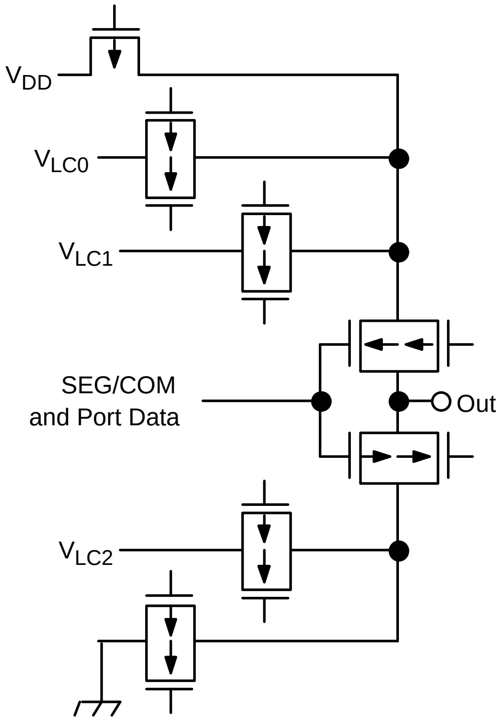


Figure 1-12. Pin Circuit Type H-16

2 ADDRESS SPACES

PROGRAM MEMORY (ROM)

OVERVIEW

ROM maps for KS57C3204 devices are mask programmable at the factory. KS57C3204 has 4K × 8-bit program memory. In its standard configuration, the device's 4,096 × 8-bit program memory has four areas that are directly addressable by the program counter (PC):

— 12-byte area for vector addresses

— 20-byte general-purpose area

— 96-byte instruction reference area

— 3968-byte general-purpose area

General-Purpose Program Memory

Two program memory areas are allocated for general-purpose use: One area is 20 bytes in size and the other is 3,968 bytes.

Vector Addresses

A 12-byte vector address area is used 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 12-byte area can be used alternately as general-purpose ROM.

REF Instructions

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

Table 2-1. Program Memory Address Ranges

ROM Area Function	Address Ranges	Area Size (in Bytes)
Vector address area	0000H–000BH	12
General-purpose program memory	000CH–001FH	20
REF instruction look-up table area	0020H–007FH	96
General-purpose program memory	0080H–0FFFH	3968

GENERAL-PURPOSE MEMORY AREAS

The 20-byte area at ROM locations 000CH–001FH and the 3,968-byte area at ROM locations 0080H–0FFFH are used as general-purpose program memory. Unused locations in the vector address area and REF instruction look-up table areas can be used as general-purpose program memory. However, care must be taken not to overwrite live data when writing programs that use special-purpose areas of the ROM.

VECTOR ADDRESS AREA

The 12-byte vector address area of the ROM is used to store the vector addresses for executing system resets and interrupts. The starting addresses of interrupt service routines are stored in this area, along with the enable memory bank (EMB) and enable register bank (ERB) flag values that are needed to initialize the service routines. 12-byte vector addresses are organized as follows:

EMB	ERB	0	0	PC11	PC10	PC9	PC8
PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

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

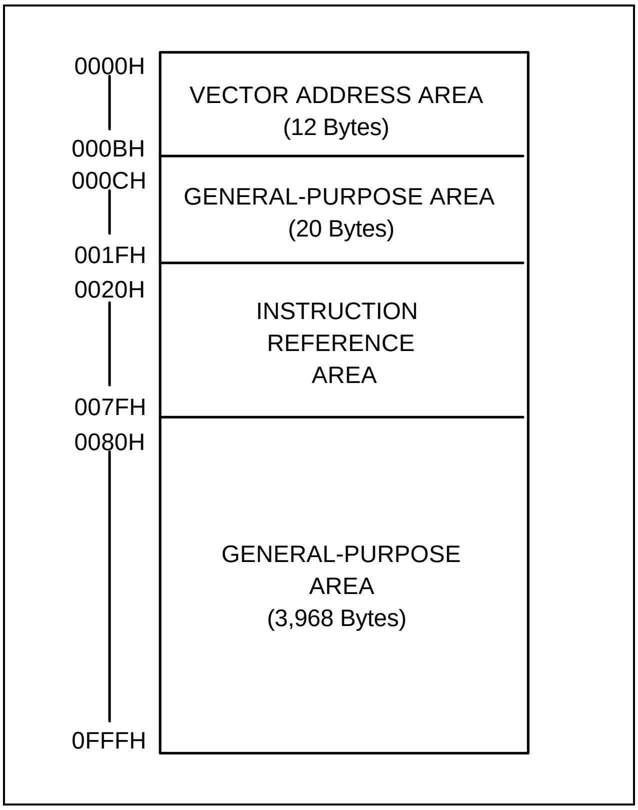


Figure 2-1. ROM Address Structure

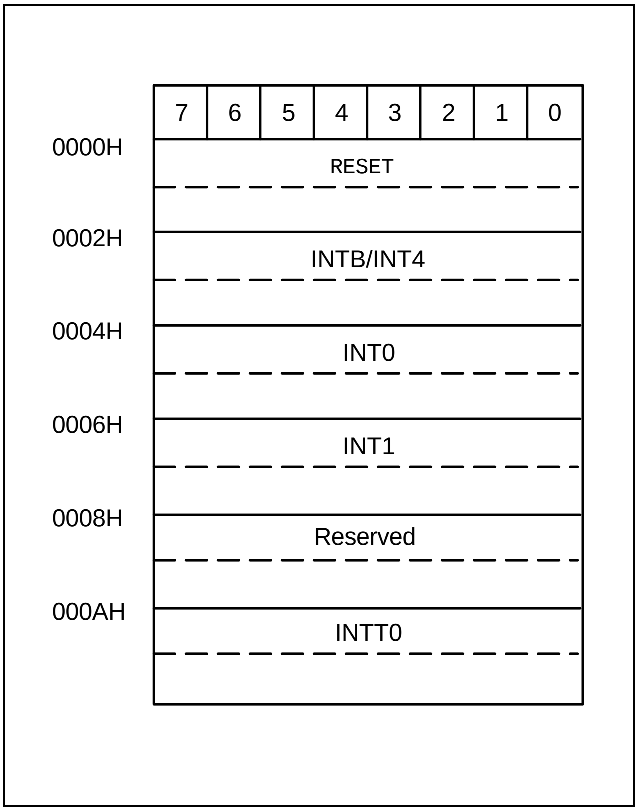


Figure 2-2. Vector Address Structure

☞ PROGRAMMING TIP — Defining Vectored Interrupts

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

1. When all vector interrupts are used:

```
        ORG       0000H

        VENT0     1,0,RESET          ;  EMB ← 1, ERB ← 0; Jump to RESET address
        VENT1     0,0,INTB           ;  EMB ← 0, ERB ← 0; Jump to INTB address
        VENT2     0,0,INT0           ;  EMB ← 0, ERB ← 0; Jump to INT0 address
        VENT3     0,0,INT1           ;  EMB ← 0, ERB ← 0; Jump to INT1 address


        ORG       000AH

        VENT5     0,0,INTT0          ;  EMB ← 0, ERB ← 0; Jump to INTT0 address
```

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

```
        ORG       0000H

        VENT0     1,0,RESET          ;  EMB ← 1, ERB ← 0; Jump to RESET address
        VENT1     0,0,INTB           ;  EMB ← 0, ERB ← 0; Jump to INTB address
        ORG       0006H              ;  INT0 interrupt not used
        VENT3     0,0,INT1           ;  EMB ← 0, ERB ← 0; Jump to INT1 address


        ORG       0010H
```

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

```
        ORG       0000H

        VENT0     1,0,RESET          ;  EMB ← 1, ERB ← 0; Jump to RESET address
        VENT1     0,0,INTB           ;  EMB ← 0, ERB ← 0; Jump to INTB address
        VENT3     0,0,INT1           ;  EMB ← 0, ERB ← 0; Jump to INT0 address
        VENT5     0,0,INTT0          ;  EMB ← 0, ERB ← 0; Jump to INT1 address

        ORG       0010H

        General-purpose ROM area
```

In this example, when an INT1 interrupt is generated, the corresponding vector area is not VENT3 INT1, but VENT5 INTT0. This causes an INT1 interrupt to 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 instruction such as a JP (jump) or CALL. The starting address of the instruction you are referencing must always be an even number. To reference a JP or CALL instruction, it must be written to the reference area in a two-byte format: for JP, this format is TJP; for CALL, it is TCALL.

By using REF instructions to execute instructions larger than one byte, you can improve program execution time considerably by reducing the number of program steps. In summary, there are three ways you can use the REF instruction:

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

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

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

☞ PROGRAMMING TIP — Using the REF Look-Up Table

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

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

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 x 4-bit data memory has three areas:

— 32 × 4-bit working register area

— 196 × 4-bit general-purpose area (also used as the stack area)

— 28 × 4-bit area for LCD data

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

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

Initialization values for the data memory area are not defined by hardware and must therefore be initialized by program software following power reset. However, when RESET signal is generated in power-down mode, the data memory contents are held (except for the values of the addresses, 0F8H–0FDH).

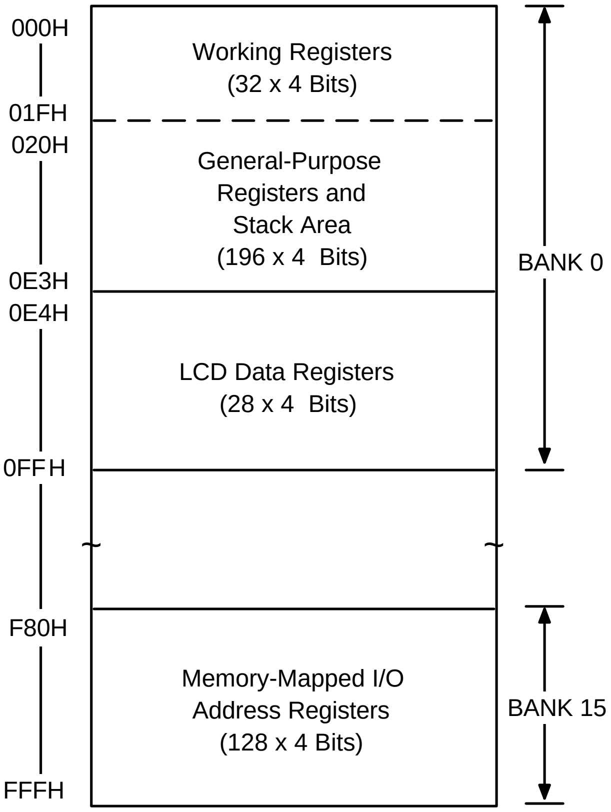


Figure 2-3. Data Memory (RAM) Map

Memory Banks 0 and 15

Bank 0	(000H–0E3H)	The lowest 32 nibbles of bank 0 (000H–01FH) are used as working registers; the next 196 nibbles (020H–0E3H) can be used both as stack area and as general-purpose data memory. Use the stack area for implementing subroutine calls and returns, and for interrupt processing.
Bank 0	(0E4H–0FFH)	28 nibbles of bank 0 are used as display registers or general purpose memory.
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 000H–07FH of bank 0 and bank 15. With indirect addressing, only bank 0 (000H–0FFH) can be accessed. When the EMB flag is set to logic one, all three data memory banks can be accessed according to the current SMB value.

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

Working Registers

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

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

LCD Data Register Area

Bit values for LCD segment data are stored in data memory bank 0. Register locations in this area that are not used to store LCD data can be assigned to general-purpose use.

Table 2-2. Data Memory Organization and Addressing

Addresses	Register Areas	Bank	EMB Value	SMB Value
000H–01FH	Working registers	0	0, 1	0
020H–0E3H	Stack and general-purpose registers			
0E4H–0FFH	LCD Data registers			
F80H–FFFH	I/O-mapped hardware registers	15	0, 1	15

☞ PROGRAMMING TIP — Clearing Data Memory Banks 0 and 1

Clear banks 0 of the data memory area:

```
          BITS      EMB
          SMB       0                   ;  RAM (010H–0FFH) clear
          LD        HL, #10H
RMCL0     LD        @HL, A
          INCS      HL
          JR        RMCL0
```

WORKING REGISTERS

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

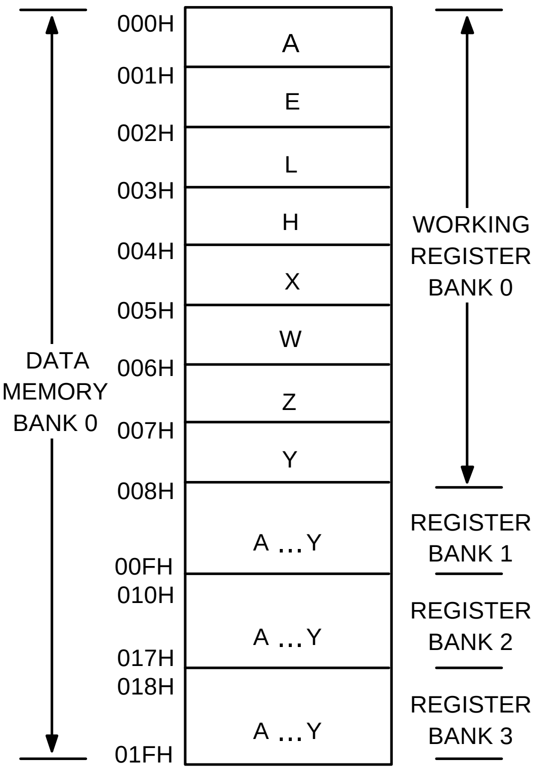


Figure 2-4. Working Register Map

Working Register Banks

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

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

Table 2-3. Working Register Organization and Addressing

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

Paired Working Registers

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

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

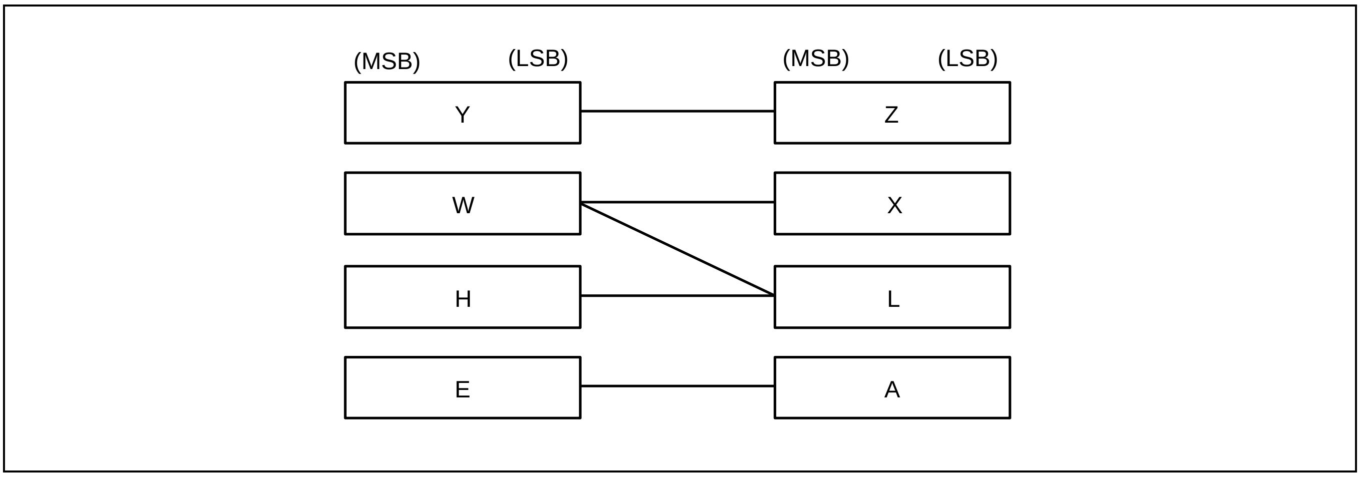


Figure 2-5. Register Pair Configuration

Special-Purpose Working Registers

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

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

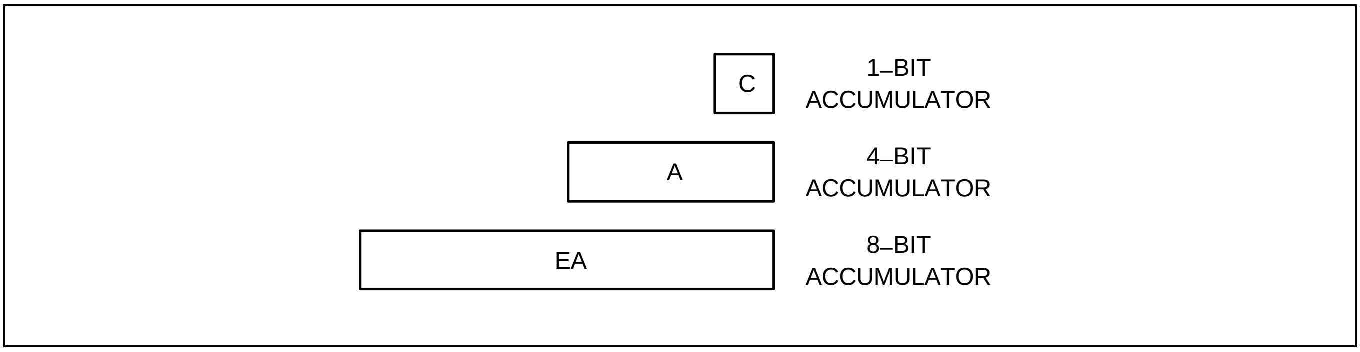


Figure 2-6. 1-Bit, 4-Bit, and 8-Bit Accumulator

Recommendation for Multiple Interrupt Processing

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

☞ PROGRAMMING TIP — Selecting the Working Register Area

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

1. When ERB = "0":

```
VENT2   1,0,INT0                        ;  EMB ← 1, ERB ← 0, Jump to INT0 address

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

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

2. When ERB = "1":

```
VENT2   1,1,INT0                        ;  EMB ← 1, ERB ← 1, Jump to INT0 address

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

STACK OPERATIONS

STACK POINTER (SP)

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

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

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

The program counter contents and program status word (PSW) 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–0E3H) regardless of the current value of the enable memory bank (EMB) flag and the select memory bank (SMB) flag. Although general-purpose register areas can be used for stack operations, be careful to avoid data loss due to simultaneous use of the same register(s).

Since the reset value of the stack pointer is not defined in firmware, we recommend that you initialize the stack pointer by program code to location 0E4H. This sets the first register of the stack area to 0E3H.

NOTE

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

☞ PROGRAMMING TIP — Initializing the Stack Pointer

To initialize the stack pointer (SP):

1. When EMB = "1":

```
        SMB     15              ;  Select memory bank 15
        LD      EA,#0E4H        ;  Bit 0 of SP is always cleared to "0"
        LD      SP,EA           ;  Stack area initial address (0E3H) ← (SP) – 1
```

2. When EMB = "0":

```
        LD      EA,#0E4H
        LD      SP,EA           ;  Memory addressing area (00H–7FH, F80H–FFFH)
```

PUSH OPERATIONS

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

PUSH Instructions

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

CALL Instructions

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

Interrupt Routines

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

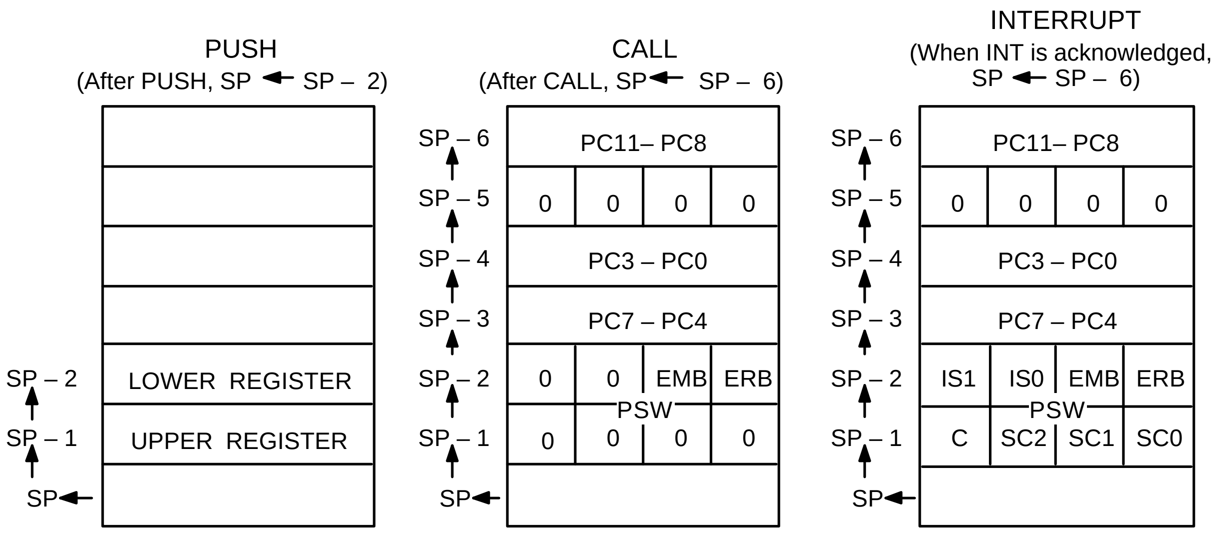


Figure 2-7. Push-Type Stack Operations

POP OPERATIONS

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

POP Instructions

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

RET and SRET Instructions

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

IRET Instructions

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

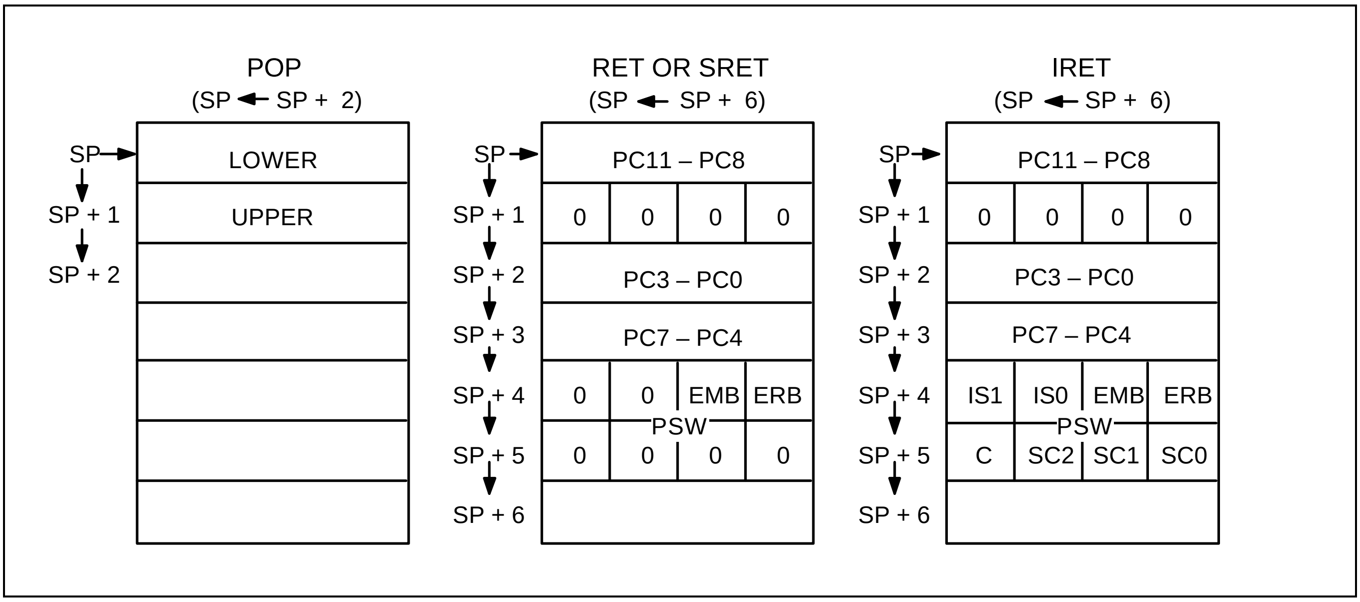


Figure 2-8. Pop-Type Stack Operations

BIT SEQUENTIAL CARRIER (BSC)

The bit sequential carrier (BSC) is a 16-bit general register that can be manipulated using 1-, 4-, and 8-bit RAM control instructions. A system reset clears all BSC bit values to logic zero.

Using the BSC, you can specify sequential addresses and bit locations 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 decreasing the value of the L register.

BSC data can also be manipulated using direct addressing. For 8-bit manipulations, the 4-bit register names BSC0 and BSC2 must be specified and 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 P1.0 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       P1.0,C           ;  P1.0 ← C
          INCS      L
          JR        AGN
          RET
```

PROGRAM COUNTER (PC)

A 12-bit program counter (PC) stores addresses for instruction fetches during program execution. Whenever a reset operation or an interrupt occurs, bits PC11 through PC0 are set to the vector address.

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

PROGRAM STATUS WORD (PSW)

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

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

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

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

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

Table 2-5. 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. 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.

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 in the interrupt priority register (IPR) are serviced
1	0	2	No more interrupt requests are serviced
1	1	–	Not applicable; these bit settings are undefined

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

☞ PROGRAMMING TIP — Setting ISx Flags for Interrupt Processing

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

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

EMB FLAG (EMB)

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

When the EMB flag is "0", the data memory address space is restricted to and addresses 000H–07FH of memory bank 0 and addresses F80H–FFFH of memory bank 15, regardless of the SMB register contents. When the EMB flag is set to "1", the general-purpose areas of bank 0 and 15 can be accessed by using the appropriate SMB value.

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

EMB flag settings for memory bank selection:

1. When EMB = "0":

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

2. When EMB = "1":

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

ERB FLAG (ERB)

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

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

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

☞ PROGRAMMING TIP — Using the ERB Flag to Select Register Banks

ERB flag settings for register bank selection:

1. When ERB = "0":

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

2. When ERB = "1":

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

SKIP CONDITION FLAGS (SC2, SC1, SC0)

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

CARRY FLAG (C)

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

If an overflow or borrow condition occurs when executing arithmetic instructions with carry (ADC, SBC), the carry flag is set to "1". Otherwise, its value is "0". When a system 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 the upper four bits of the 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     P1.0,C               ;  Output result from carry flag to P1.0
```

NOTES

3 ADDRESSING MODES

OVERVIEW

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

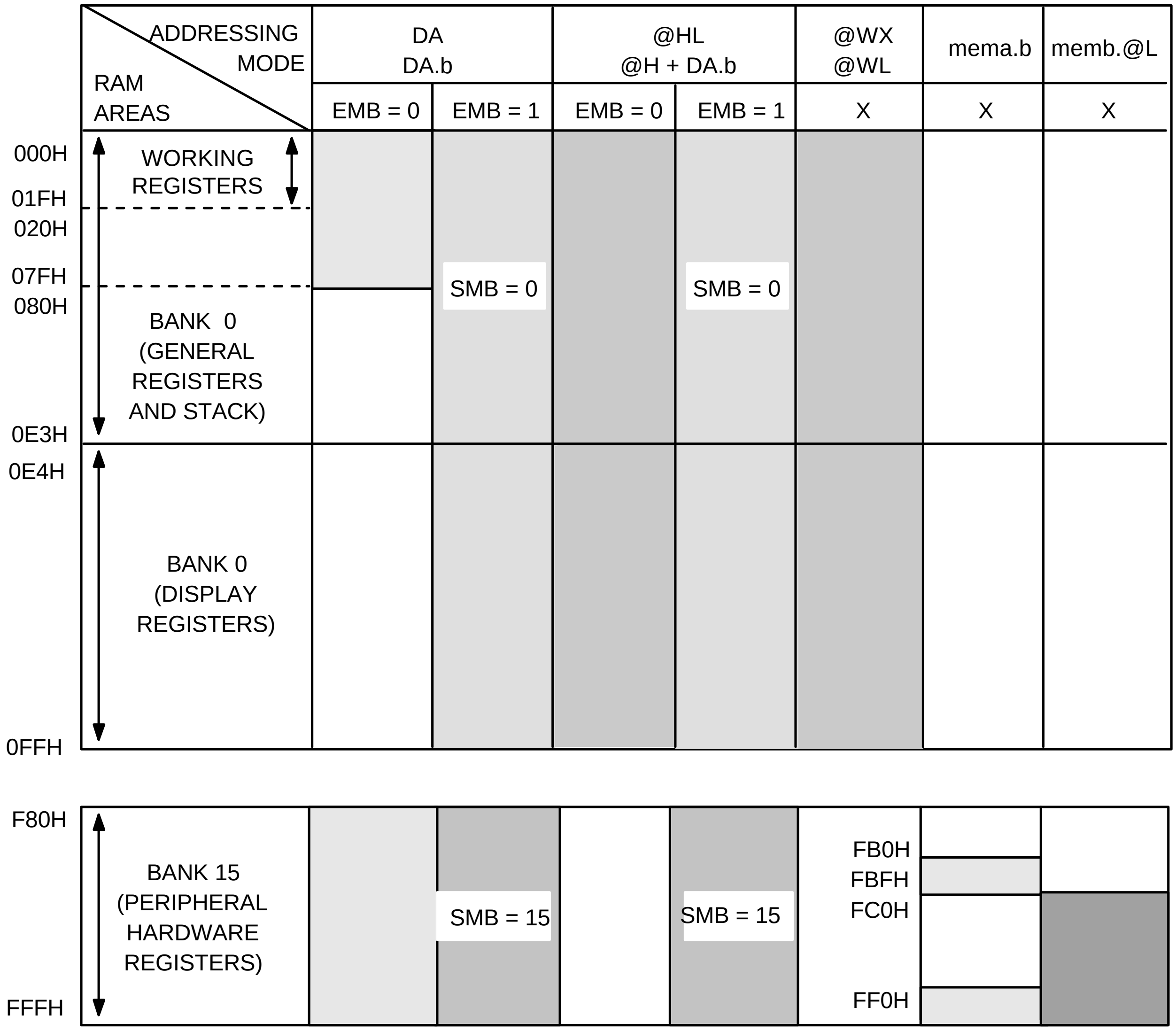
The EMB flag works in connection with the select memory bank instruction, SMB n. You will recall that the SMB n instruction is used to select RAM bank 0 or 15. The SMB setting is always contained in the upper four bits of a 12-bit RAM address. For this reason, both addressing modes (EMB = "0" and EMB = "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. Several RAM locations are addressable at all times, regardless of the current EMB flag setting.

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

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

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

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



NOTES

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

Figure 3-1. RAM Address Structure

EMB AND ERB INITIALIZATION VALUES

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

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

☞ PROGRAMMING TIP — Initializing the EMB and ERB Flags

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

```
        ORG     0000H           ;  ROM address assignment
        VENT0   1,0,RESET       ;  EMB ← 1, ERB ← 0, branch RESET
        VENT1   0,1,INTB        ;  EMB ← 0, ERB ← 1, branch INTB
        VENT2   0,1,INT0        ;  EMB ← 0, ERB ← 1, branch INT0
        VENT3   0,1,INT1        ;  EMB ← 0, ERB ← 1, branch INT1
        ORG     000AH           ;  ROM address assignment
        VENT5   0,1,INTT0       ;  EMB ← 0, ERB ← 1, branch INTT0
        •
        •
        •
RESET   BITR    EMB
```

ENABLE MEMORY BANK SETTINGS

EMB = "1"

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

If SMB = 0,	000H–0FFH
If SMB = 15,	F80H–FFFH

EMB = "0"

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

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

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

EMB-Independent Addressing

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

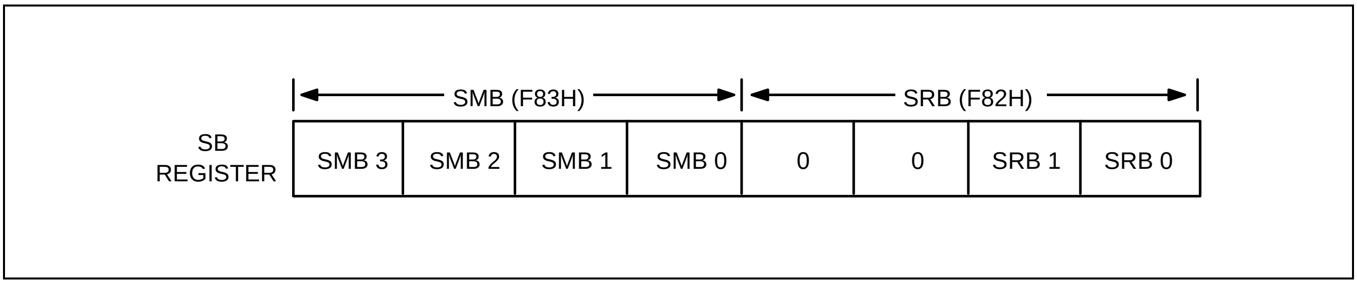
Table 3-1. RAM Addressing Not Affected by the EMB Value

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

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.



SMB 3	SMB 2	SMB 1	SMB 0	0	0	SRB 1	SRB 0

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

Select Register Bank (SRB) Instruction

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

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

Select Memory Bank (SMB) Instruction

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

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

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

DIRECT AND INDIRECT ADDRESSING

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

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

1-BIT ADDRESSING

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

Operand Notation	Addressing Mode Description	EMB Flag Setting	Addressable Area	Memory Bank	Hardware I/O Mapping
DA.b	Direct: bit is indicated by the RAM address (DA), memory bank selection, and specified bit number (b).	0	000H–07FH	Bank 0	–
			F80H–FFFH	Bank 15	All 1-bit addressable peripherals (SMB = 15)
		1	000H–0FFH	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 6 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	–
		1	000H–0FFH	SMB = 0, 15	All 1-bit addressable peripherals (SMB = 15)

NOTE: x = not applicable.

☞ PROGRAMMING TIP — 1-Bit Addressing Modes

1-Bit Direct Addressing

1. If EMB = "0":

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

2. If EMB = "1":

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

1-Bit Indirect Addressing

1. If EMB = "0":

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

2.If EMB = "1":

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

4-BIT ADDRESSING

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

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

NOTE: x = not applicable.

☞ PROGRAMMING TIP — 4-Bit Addressing Modes

4-Bit Direct Addressing

1. If EMB = "0":

```
ADATA   EQU     46H
BDATA   EQU     8EH
        SMB     15              ;  Non-essential instruction, since EMB = "0"
        LD      A,P3            ;  A ← (P3)
        SMB     0               ;  Non-essential instruction, since EMB = "0"
        LD      ADATA,A         ;  (046H) ← A
        LD      BDATA,A         ;  (F8EH (LCON)) ← A
```

4-Bit Indirect Addressing

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

```
ADATA   EQU     46H
BDATA   EQU     66H
        SMB     15              ;  Non-essential instruction, since EMB = "0"
        LD      HL,#BDATA
        LD      WX,#ADATA
COMP    LD      A,@WL           ;  A ← bank 0 (040H–046H)
        CPSE    A,@HL           ;  If bank 0 (060H–066H) = A, skip
        SRET
        DECS    L
        JR      COMP
        RET
```

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

```
ADATA   EQU     46H
BDATA   EQU     66H
        SMB     15              ;  Non-essential instruction, since EMB = "0"
        LD      HL,#BDATA
        LD      WX,#ADATA
TRANS   LD      A,@WL           ;  A ← bank 0 (040H–046H)
        XCHD    A,@HL           ;  Bank 0 (060H–066H) ↔ A
        JR      TRANS
```

8-BIT ADDRESSING

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

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

☞ PROGRAMMING TIP — 8-Bit Addressing Modes

8-Bit Direct Addressing

1. If EMB = "0":

```
ADATA   EQU     46H
BDATA   EQU     8EH
        LD      EA, #0FFH
        SMB     0
        LD      ADATA,EA            ;  (46H)  ← A, (47H) ← E
        LD      BDATA,EA            ;  (F8EH) ← A, (F8FH) ← E
```

2. If EMB = "1":

```
ADATA   EQU     46H
BDATA   EQU     8EH
        SMB     0
        LD      EA, #0FFH
        LD      ADATA,EA            ;  (46H) ← A, (47H) ← E
        LD      BDATA,EA            ;  (8EH) ← A, (8FH) ← E
```

8-Bit Indirect Addressing

1. If EMB = "0":

```
ADATA   EQU     46H
        SMB     15                  ;  Non-essential instruction, since EMB = "0"
        LD      HL,#ADATA
        LD      EA,@HL              ;  A ← (46H), E ← (47H)
```

NOTES

4 MEMORY MAP

OVERVIEW

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

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

I/O MAP FOR HARDWARE REGISTERS

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

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

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

Memory Bank 15							Addressing Mode		
Address	**Register**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**R/W**	**1-Bit**	**4-Bit**	**8-Bit**
F80H	SP	.3	.2	.1	"0"	R/W	No	No	Yes
F81H		.7	.6	.5	.4				
Locations F82H–F84H are not mapped.									
F85H	BMOD	.3	.2	.1	"0"	W	.3 (W)	Yes	No
F86H	BCNT	.3	.2	.1	.0	R	No	No	Yes
F87H		.7	.6	.5	.4				
F88H	WMOD	.3	.2	.1	.0	W [1]	.3 (R)	No	Yes
F89H		.7	"0"	.5	.4				
Locations F8AH–F8BH are not mapped.									
F8CH	LMOD	.3	.2	.1	.0	W	.3 (W)	No	Yes
F8DH		.7	.6	.5	.4				
F8EH	LCON	"0"	.2	.1	.0	W	No	Yes	No
Location F8FH is not mapped.									
F90H	TMOD0	.3	.2	"0"	"0"	W	.3 (W)	No	Yes
F91H		"0"	"0"	.5	.4				
Locations F92H–F93H are not mapped.									
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				
F98H	WDMOD	.3	.2	.1	.0	W	No	No	Yes
F99H		.7	.6	.5	.4				
F9AH	WDFLAG	WDTCF	"0"	"0"	"0"	W	Yes	Yes	No
Locations F9BH–F9FH are not mapped.									
FA0H	FCNT0	.3	.2	.1	.0	R	No	No	Yes
FA1H		.7	.6	.5	.4				
FA2H	FCNT1	.3	.2	.1	.0	R	No	No	Yes
FA3H		.7	.6	.5	.4				
FA4H	FCNT2	.3	.2	.1	.0	R	No	No	Yes
FA5H		.7	.6	.5	.4				
FA6H	FCMOD	"0"	"0"	"0"	.0	W	No	Yes	No
FA7H	FCCON	.3	.2	.1	.0	W	Yes	Yes	No

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**
Location FA8H–FADH are not mapped.									
FAEH	APCON	.3	.2	.1	.0	W	No	Yes	No
Location FAFH is not mapped.									
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	IMOD0	.3	"0"	.1	.0	W	No	Yes	No
FB5H	IMOD1	"0"	"0"	"0"	.0	W	No	Yes	No
FB6H	IMOD2	"0"	"0"	"0"	.0	W	No	Yes	No
FB7H	SCMOD	.3	.2	"0"	.0	W	Yes	No	No
FB8H	INT (A)	IE4	IRQ4	IEB	IRQB	R/W	Yes	Yes	No
Location FB9H is not mapped									
FBAH	INT (B)	"0"	"0"	IEW	IRQW	R/W	Yes	Yes	No
Location FBBH is not mapped.									
FBCH	INT (C)	"0"	"0"	IET0	IRQT0	R/W	Yes	Yes	No
Location FBDH is not mapped.									
FBEH	INT (E)	IE1	IRQ1	IE0	IRQ0	R/W	Yes	Yes	No
FBFH	INT (F)	"0"	"0"	IE2	IRQ2	R/W	Yes	Yes	No
FC0H	BSC0	.3	.2	.1	.0	R/W	Yes	Yes	Yes
FC1H	BSC1	.3	.2	.1	.0		Yes	Yes	
FC2H	BSC2	.3	.2	.1	.0		Yes	Yes	Yes
FC3H	BSC3	.3	.2	.1	.0		Yes	Yes	
Locations FC4H–FD5H are not mapped.									
FD6H	PNE	PNE4.3	PNE4.2	PNE4.1	PNE4.0	W	No	No	Yes
FD7H		PNE5.3	PNE5.2	PNE5.1	PNE5.0				
FD8H	ADATA	.3	.2	.1	.0	R	No	No	Yes
FD9H		.7	.6	.5	.4				
FDAH	ADMOD	"0"	"0"	.1	.0	R/W	Yes	Yes	No
FDBH	AFLAG	ADSTR	EOC	"0"	ADCLK	[3]	Yes	Yes	No
FDCH	PUMOD	PUR3	PUR2	PUR1	"0"	W	No	No	Yes
FDDH		"0"	PUR6	PUR5	PUR4				
Locations FDEH–FE5H are not mapped.									

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

Memory Bank 15							Addressing Mode		
Address	**Register**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**R/W**	**1-Bit**	**4-Bit**	**8-Bit**
FE6H	PMG0	"0"	"0"	"0"	"0"	W	No	No	Yes
FE7H		PM1.3	PM1.2	PM1.1	PM1.0				
FE8H	PMG1	PM5	PM4	"0"	"0"	W	No	No	Yes
FE9H		PM6.3	PM6.2	PM6.1	PM6.0				
Locations FEAH–FF0H are not mapped.									
FF1H	Port 1	.3	.2	.1	.0	R/W	Yes	Yes	No
FF2H	Port 2	.3	.2	.1	.0	R	Yes	Yes	No
FF3H	Port 3	.3	.2	.1	.0	R	Yes	Yes	No
FF4H	Port 4	.3	.2	.1	.0	R/W	Yes	Yes	Yes
FF5H	Port 5	.7/.3	.6/.2	.5/.1	.4/.0		Yes	Yes	
FF6H	Port 6	.3	.2	.1	.0	R/W	Yes	Yes	No
Location FF7H is not mapped.									
FF8H	Port 8	.3	.2	.1	.0	W	Yes	Yes	Yes
FF9H	Port 9	.7/.3	.6/.2	.5/.1	.4/.0		Yes	Yes	
Locations FFAH–FFFH are not mapped.									

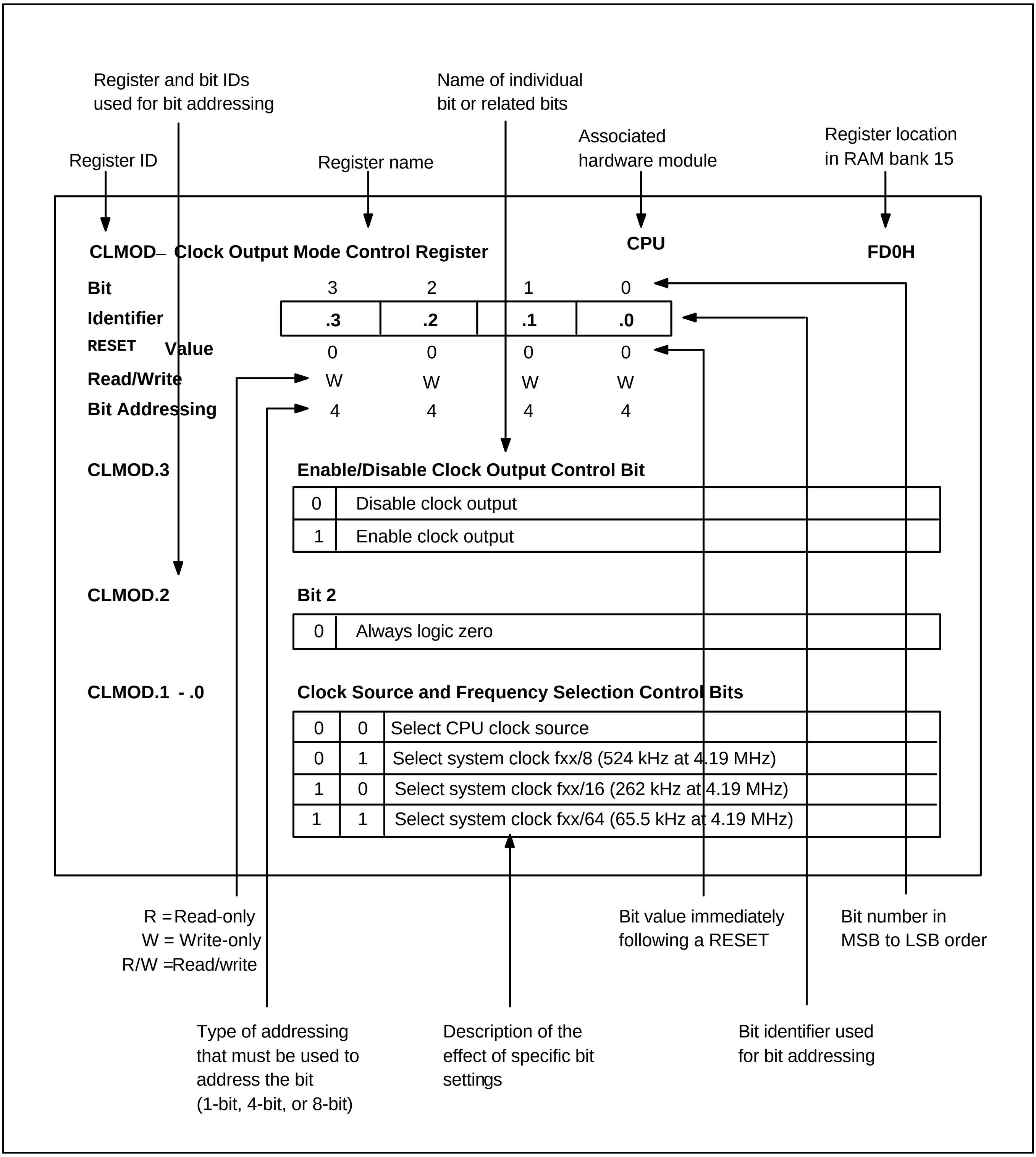
NOTES:

1. All bits except WMOD.3 (F88H.3) can be manipulated by 8-bit write instruction only. WMOD.3 (F88H.3) can be manipulated by only 1-bit read instruction.
2. The carry flag can be read or written by specific bit manipulation instructions only.
3. ADSTR and ADCLK bits of the AFLAG register are 1-or 4-bit write only, but the EOC bit is 1-or 4-bit read 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. Figure 4-1 describes features of the register description format. Register descriptions are arranged in alphabetical order. Programmers can use this section as a quick-reference source when writing application programs.

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


Register and bit IDs used for bit addressing
Name of individual bit or related bits
Associated hardware module
Register location in RAM bank 15
Register ID
Register name
R = Read-only
W = Write-only
R/W = Read/write
Bit value immediately following a RESET
Bit number in MSB to LSB order
Type of addressing that must be used to address the bit (1-bit, 4-bit, or 8-bit)
Description of the effect of specific bit settings
Bit identifier used for bit addressing

CLMOD – Clock Output Mode Control Register **CPU** **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

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

0	Disable clock output
1	Enable clock output

CLMOD.2 **Bit 2**

0	Always logic zero

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

0	0	Select CPU clock source
0	1	Select system clock fxx/8 (524 kHz at 4.19 MHz)
1	0	Select system clock fxx/16 (262 kHz at 4.19 MHz)
1	1	Select system clock fxx/64 (65.5 kHz at 4.19 MHz)

Figure 4-1. Register Description Format

ADMOD — ADC Mode Register FDAH

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

.3–.2 **Bits 3–2**

0	Always logic zero

.1 – .0 **ADC Analog Input Pin Selection Bits**

0	0	Select input channel ADC0 (P3.0)
0	1	Select input channel ADC1 (P3.1)
1	0	Select input channel ADC2 (P3.2)
1	1	Select input channel ADC3 (P3.3)

AFLAG — ADC Flag Register FDBH

Bit	3	2	1	0
Identifier	**ADSTR**	**EOC**	**.1**	**ADCLK**
RESET Value	0	0	0	0
Read/Write	W	R	–	W
Bit Addressing	1/4	1/4	–	1/4

ADSTR **ADC Conversion Start Control Flag**

1	Enable ADC (when the ADSTR bit is set to "1", the ADC starts operating and the ADSTR bit is cleared automatically)

EOC **End-of-Conversion Bit (Read-only)**

0	A/D conversion operation is in progress
1	A/D conversion operation is complete

.1 **Bit 1**

0	Always logic zero

ADCLK **ADC Clock Source Selection**

0	Conversion clock = $f_{xx}/2$
1	Conversion clock = $f_{xx}/4$

NOTES: 'fxx' stands for the system clock .

APCON — ADC and Port Control Register FAEH

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 **Pin (P3.0) input Selection Bit (ADC input or Normal Port Input)**

0	Connect to a normal input block (digital signal input)
1	Connect to a ADC block (analog signal input)

.2 **Pin (P3.1) input Selection Bit (ADC input or Normal Port Input)**

0	Connect to a normal input block (digital signal input)
1	Connect to a ADC block (analog signal input)

.1 **Pin (P3.2) input Selection Bit (ADC input or Normal Port Input)**

0	Connect to a normal input block (digital signal input)
1	Connect to a ADC block (analog signal input)

.0 **Pin (P3.3) input Selection Bit (ADC input or Normal Port Input)**

0	Connect to a normal input block (digital signal input)
1	Connect to a ADC block (analog signal input)

NOTE:If the specific ports were set as a normal input mode, don’t connect an analog signals.

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	–
Bit Addressing	1/4	4	4	–

.3 **Basic Timer Restart Bit**

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

.2–.1 **Input Clock Frequency and Signal Stabilization Interval Control Bits**

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

.0

0	Always logic zero

NOTES:
1. Signal stabilization interval is the time required to stabilize clock signal oscillation after stop mode is terminated by an interrupt. The stabilization interval can also be interpreted as "Interrupt Interval Time".
2. When a system reset occurs, the oscillation stabilization time is 31.3 ms (2^{17}/fxx) at 4.19 MHz.
3. 'fxx' is the system clock rate given a clock frequency of 4.19 MHz.

FCCON — Frequency Counter Control Register FA7H

Bit	3	2	1	0
Identifier	**.3**	**.2**	**.1**	**.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 – .2 **Counter Input Signal Selection Bits**

FCMOD.0	FCCON.3,2		Function
0	–	–	P2.2 and P2.3 are used as general input port; also used to count DC pulse signal.
	x	0	Select P2.3 signal as counter input
	x	1	Select P2.2 signal as counter input; non-divided
1	–	–	P2.2 or P2.3 are used to count AC frequency signal
	0	0	Select AMF signal; disable FMF pin (feedback resistor is off, pull-down through a resistor)
	0	1	Select FMF signal, 1/16-divided; disable AMF pin (feedback resistor is off, pull-down through a resistor)
	1	x	Disable all inputs and frequency counter; feedback resistors are off, pull-down through a resistor.

.1 **Counter Stop or Start Control Bit**

0	Stop increasing a counter value
1	Start increasing a counter value

.0 **Clear Counter Value Control Bit**

1	Clear frequency values (FCNT0-2)and 1/16-divider

FCMOD — Frequency Counter Mode Register FA6H

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

.3 – .1 **Bits 3–1**

0	Always logic zero

.0 **Pin (P2.2, P2.3) Mode Control Bit**

0	General input mode
1	AM and FM frequencies input mode

IE0, 1, IRQ0, 1 — INT0, 1 Interrupt Enable/Request Flags FBEH

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

IE1 **INT1 Interrupt Enable Flag**

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

IRQ1 **INT1 Interrupt Request Flag**

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

IE0 **INT0 Interrupt Enable Flag**

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

IRQ0 **INT0 Interrupt Request Flag**

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

IE2, IRQ2 — INT2 Interrupt Enable/Request Flags FBFH

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

.3–.2 **Bits 3–2**

0	Always logic zero

IE2 **INT2 Interrupt Enable Flag**

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

IRQ2 **INT2 Interrupt Request Flag**

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

IEB, IRQB — INTB Interrupt Enable/Request Flags FB8H

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

IE4 **Bit 3**

0	Disable INT4 interrupt requests
1	Enable INT4 interrupt requests

IRQ4 **Bits 2**

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

IEB **INTB Interrupt Enable Flag**

0	Disable INTB interrupt requests
1	Enable INTB interrupt requests

IRQB **INTB Interrupt Request Flag**

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

IET0, IRQT0 — INTT0 Interrupt Enable/Request Flags FBCH

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

.3–.2 **Bits 3–2**

0	Always logic zero

IET0 **INTT0 Interrupt Enable Flag**

0	Disable INTT0 interrupt requests
1	Enable INTT0 interrupt requests

IRQT0 **INTT0 Interrupt Request Flag**

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

IEW, IRQW — INTW Interrupt Enable/Request Flags FBAH

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

.3–.2 **Bits 3–2**

0	Always logic zero

IEW **INTW Interrupt Enable Flag**

0	Disable INTW interrupt requests
1	Enable INTW interrupt requests

IRQW **INTW Interrupt Request Flag**

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

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

IMOD0 — External Interrupt 0 (INT0) Mode Register FB4H

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

.3 **Interrupt Sampling Clock Selection Bit**

0	Select CPU clock as a sampling clock
1	Select sampling clock frequency of the selected system clock (fxx/64)

.2 **Bit 2**

0	Always logic zero

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

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

IMOD1 — External Interrupt 1 (INT1) Mode Register FB5H

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

.3–.1 **Bits 3–1**

0	Always logic zero

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

0	Rising edge detection
1	Falling edge detection

IMOD2 — External Interrupt 2 (INT2) Mode Register FB6H

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

.3–.1 **Bits 3–1**

0	Always logic zero

.0 **IRQ2 Source Selection Bit**

0	Select rising edge at INT2 pin
1	Select falling edge at KS0–KS2

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	Normal interrupt handling according to default priority settings
0	0	1	Process INTB and INT4 interrupts at the highest priority
0	1	0	Process INT0 interrupt at the highest priority
0	1	1	Process INT1 interrupt at the highest priority
1	0	0	Not used in KS57C3204
1	0	1	Process INTT0 interrupt at the highest priority
1	1	0	Not used in KS57C3204
1	1	1	

LCON — LCD Output Control Register F8EH

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

.3 **Bit 3**

0	Always logic zero.

.2 **LCD Dividing Register Control Bit**

0	Disconnect V_{LC1} to V_{LC2} internally, (open)
1	Connect V_{LC1} to V_{LC2} internally. (short)

.1 **LCD Dividing Register Control Bit**

0	Disconnect V_{LC0} to bias internally. (open)
1	Connect V_{LC0} to bias internally. (short)

.0 **LCD Display Control Bit**

0	LCD output low, turns display off: cut off current to dividing resistor
1	If LMOD.3 = “0”: turn display off; If LMOD.3 = “1”: COM and SEG output in display mode; LCD display on.

NOTES: To select the LCD bias, you must properly configure both LCON.2 and LCON.1.

LMOD — LCD Mode Register F8DH, F8CH

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

.7–.6 **LCD Output Segment and Pin Configuration Bits**

0	0	Segments 27–24 and 23–20
0	1	Segment 27–24 and normal output at P9.3–P9.0
1	0	Normal output at P8.3–P8.0 and segments 23–20
1	1	Normal output only at P9.3–P9.0 and P8.3–P8.0

.5–.4 **LCD Clock (LCDCK) Frequency Selection Bits**

0	0	$fw/2^9$ = 64 Hz
0	1	$fw/2^8$ = 128 Hz
1	0	$fw/2^7$ = 256 Hz
1	1	$fw/2^6$ = 512 Hz

.3–.0 **Duty and Bias Selection for LCD Display**

0	–	–	–	LCD display off
1	0	0	0	1/4 duty, 1/3 bias (note)
1	0	0	1	1/3 duty, 1/3 bias (note)
1	0	1	0	1/2 duty, 1/2 bias (note)
1	0	1	1	1/3 duty, 1/2 bias
1	1	0	0	Static

NOTE: Bias can be configured as setting LCON register.

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

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

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

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

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

PMG0 — Port I/O Mode Flags (Group 0: Port 1) FE7H, FE6H

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

.7 **P1.3 I/O Mode selection Flag**

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

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

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

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

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

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

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

.3–.0 **Bits 3–0**

0	Always logic zero

PMG1 — Port I/O Mode Flags (Group 1: Ports 4, 5, and 6) FE9H, FE8H

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

.7 **P6.3 I/O Mode selection Flag**

0	Set P6.3 to input mode
1	Set P6.3 to output mode

.6 **P6.2 I/O Mode Selection Flag**

0	Set P6.2 to input mode
1	Set P6.2 to output mode

.5 **P6.1 I/O Mode Selection Flag**

0	Set P6.1 to input mode
1	Set P6.1 to output mode

.4 **P6.0 I/O Mode Selection Flag**

0	Set P6.0 to input mode
1	Set P6.0 to output mode

.3 **P5 I/O Mode Selection Flag**

0	Set P5 to input mode
1	Set P5 to output mode

.2 **P4 I/O Mode Selection Flag**

0	Set P4 to input mode
1	Set P4 to output mode

.1–.0 **Bits 1–0**

0	Always logic zero

PSW — Program Status Word FB1H, FB0H

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

C **Carry Flag**

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

SC2–SC0 **Skip Condition Flags**

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

IS1, IS0 **Interrupt Status Flags**

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

EMB **Enable Data Memory Bank Flag**

0	Restrict program access to data memory to bank 15 (F80H–FFFH) and to the locations 000H–07FH in the bank 0 only
1	Enable full access to data memory banks 0, 1, 2, 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 system reset occurs during normal operation is undefined. If a RESET occurs during power-down mode (IDLE or STOP), the current value of the carry flag is retained.
2. The carry flag can only be addressed by a specific set of 1-bit manipulation instructions. See Section 2 for detailed information.

PUMOD — Pull-Up Resistor Mode Register FDDH, FDCH

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

.7 **Bit 7**

0	Always logic zero

.6 **Connect/Disconnect Port 6 Pull-up Resistor Control Bit**

0	Disconnect port 6 pull-up resistor
1	Connect port 6 pull-up resistor

.5 **Connect/Disconnect Port 5 Pull-up Resistor Control Bit**

0	Disconnect port 5 pull-up resistor
1	Connect port 5 pull-up resistor

.4 **Connect/Disconnect Port 4 Pull-up Resistor Control Bit**

0	Disconnect port 4 pull-up resistor
1	Connect port 4 pull-up resistor

.3 **Connect/Disconnect Port 3 Pull-up Resistor Control Bit**

0	Disconnect port 3 pull-up resistor
1	Connect port 3 pull-up resistor

.2 **Connect/Disconnect Port 2 Pull-up Resistor Control Bit**

0	Disconnect port 2 pull-up resistor
1	Connect port 2 pull-up resistor

.1 **Connect/Disconnect Port 1 Pull-up Resistor Control Bit**

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

.0 **Bit 0**

0	Always logic zero

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

SCMOD — System Clock Mode Control Register FB7H

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

.3, .2 and .0 **CPU Clock Selection and Main System Clock Oscillation Control Bits**

0	0	0	Select main system clock (fx); enable main system clock
0	1	0	Select main system clock (fx); disable sub system clock
0	0	1	Select sub system clock (fxt); enable main system clock
1	0	1	Select sub system clock (fxt); disable main system clock

.1 **Bit 1**

0	Always logic zero

NOTE: SCMOD bits 3 and 0 cannot be modified simultaneously by a 4-bit instruction; they can only be modified by separate 1-bit instructions.

TMOD0 — Timer 0 Mode Register F91H, F90H

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

.7–.6 **Bits 7–6**

0	Always logic zero

.5–.4 **Timer 0 Input Clock Selection Bits**

0	0	Select clock: fxx/2^{10} (4.09 kHz at 4.19 MHz)
0	1	Select clock: fxx/2^{6} (65.5 kHz at 4.19 MHz)
1	0	Select clock: fxx/2^{4} (26.2 kHz at 4.19 MHz)
1	1	Not used in KS57C3204

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

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

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

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

.1–.0 **Bits 1–0**

0	Always logic zero

WDFLAG — Watchdog Timer Counter Clear Flag Register F9AH

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

WDTCF **Watchdog Timer Counter Clear Flag**

1	Clears the watchdog timer counter

.2–.0 **Bits 2–0**

0	Always logic zero

NOTE: After watchdog timer is cleared by writing “1”, this bit is cleared to “0” automatically. Instruction that clear the watchdog timer (“BITS WDTCF”) should be executed at proper points in a program within a given period. If not executed within a given period and watchdog timer overflows, A reset is generated internally and system is restarted with reset status.

WDMOD — Watchdog Timer Mode Register F99H, F98H

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

WDMOD **Watchdog Timer Enable/Disable Control**

5AH	Disable watchdog timer function
Any other value	Enable watchdog timer function

WMOD — Watch Timer Mode Register F89H, F88H

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

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

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

.6 **Bit 6**

0	Always logic zero

.5–.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 **XT_{in} Input Level Control Bit**

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

.2 **Enable/Disable Watch Timer 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 **Watch Timer Clock Selection Bit**

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

NOTE: A system reset sets WMOD.3 to the current input level of the subsystem clock, XT_{in}. If the input level is high, WMOD.3 is set to logic one; if low, WMOD.3 is cleared to zero along with all the other bits in the WMOD register.

NOTES

Oscillator Circuits

Interrupts

Power-Down

RESET

I/O Ports

Timers

LCD Controller/Driver

Electrical Data

Mechanical Data

KS57P3204 OTP

Development Tools

6 OSCILLATOR CIRCUITS

OVERVIEW

The KS57C3204 microcontroller has two oscillator circuits: a main system clock circuit, and a subsystem clock circuit. The CPU and peripheral hardware operate on the system clock frequency supplied through these circuits. Specifically, a clock pulse is required by the following peripheral modules:

— LCD controller

— Basic timer

— Timer 0

— Watch timer

— A/D converter

CPU Clock Notation

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

fx	Main system clock
fxt	Subsystem clock
fxx	Selected system clock

Clock Control Registers

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

PCON is used to select normal CPU operating mode or one of two power-down modes — stop or idle. 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.

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

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

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

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

Using a Subsystem Clock

If a subsystem clock is being used as the selected system clock, the idle power-down mode can be initiated by executing an IDLE instruction. The subsystem clock can be stopped by setting SCMOD.2 to “1”.

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

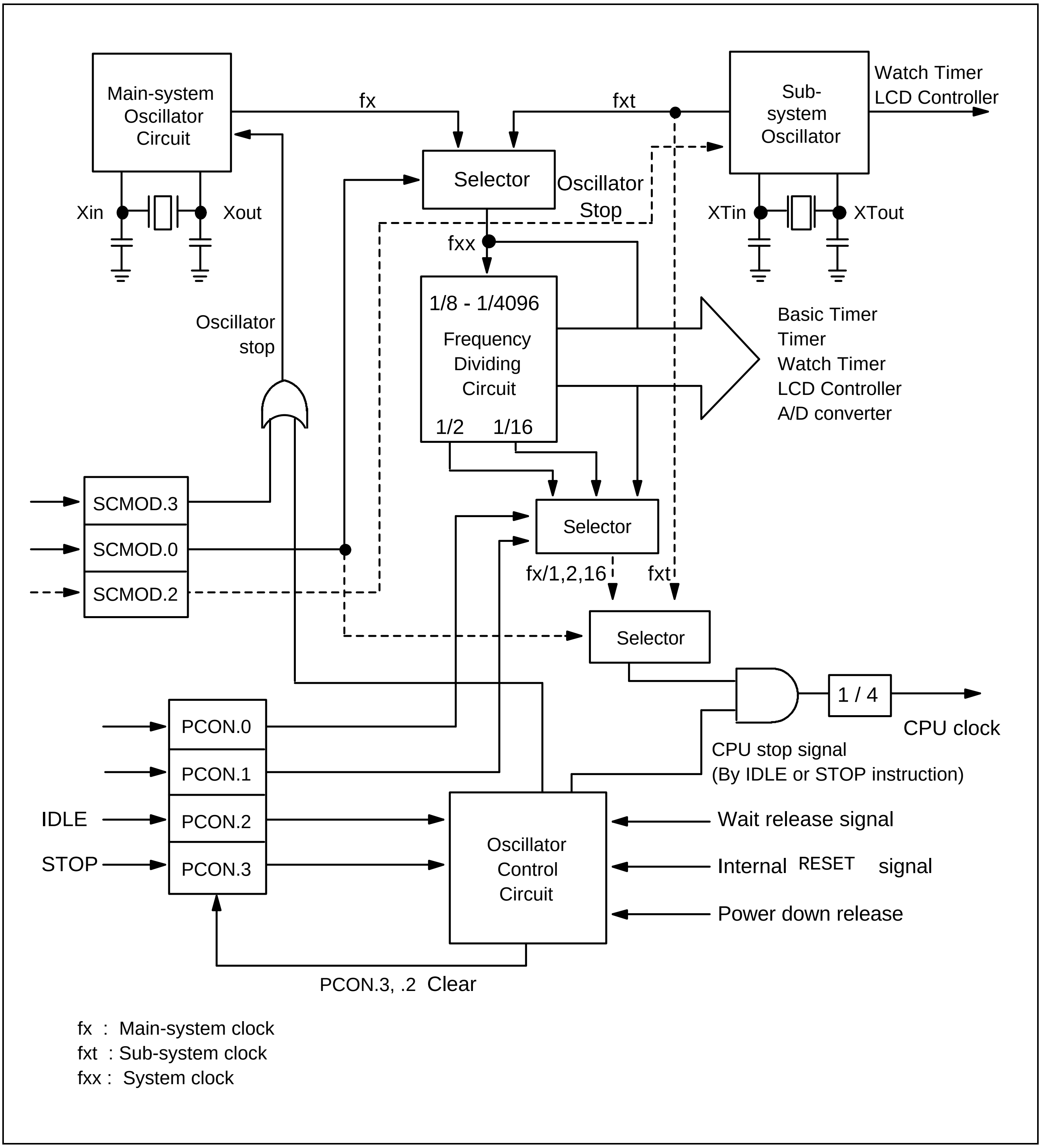


Figure 6-1. Clock Circuit Diagram

MAIN SYSTEM OSCILLATOR CIRCUITS

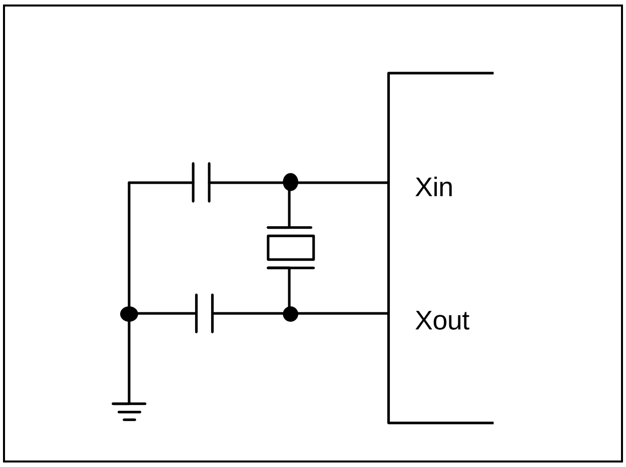


Figure 6-2. Crystal/Ceramic Oscillator

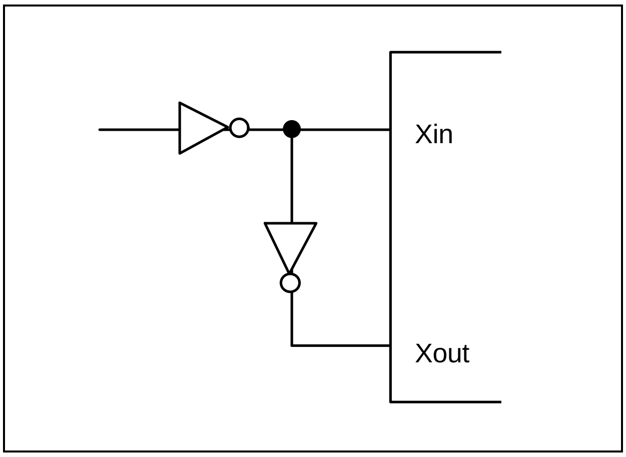


Figure 6-3. External Oscillator

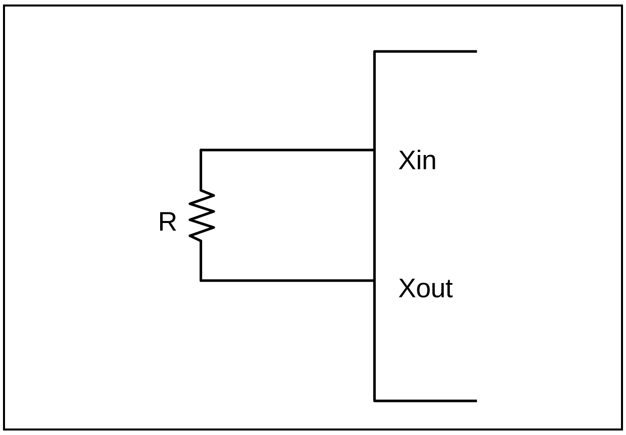


Figure 6-4. RC Oscillator

SUBSYSTEM OSCILLATOR CIRCUITS

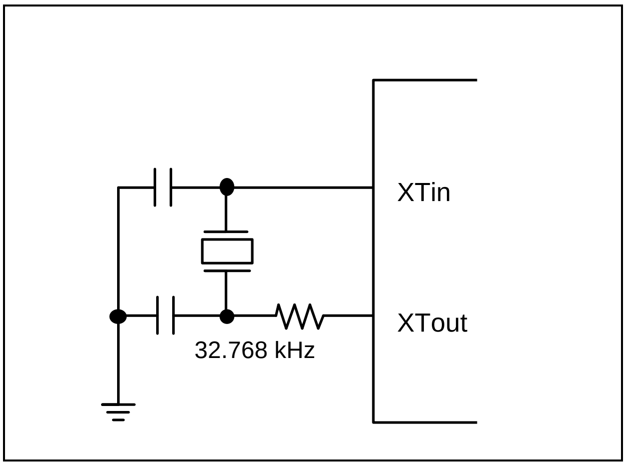


Figure 6-5. Crystal/Ceramic Oscillator

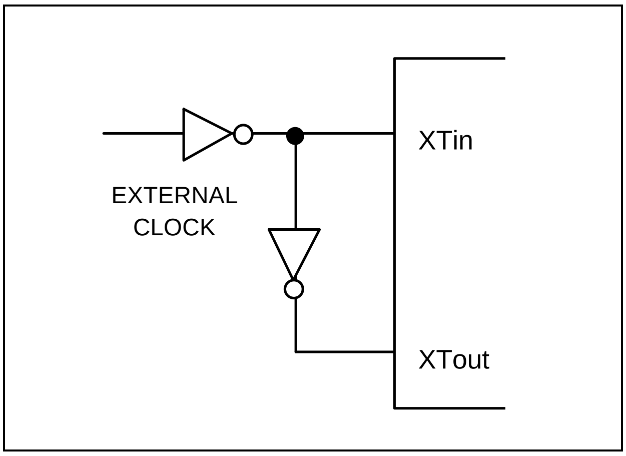


Figure 6-6. External Oscillator

POWER CONTROL REGISTER (PCON)

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

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

PCON.3 and PCON.2 can be addressed only by the STOP and IDLE instructions, respectively, to engage the idle and stop power-down modes. Idle and stop modes can be initiated by these instruction despite the current value of the enable memory bank flag (EMB). PCON bits 1 and 0 can be written only by 4-bit RAM control instruction. PCON is a write-only register. There are three basic choices:

— Main system clock (fx) or subsystem clock (fxt);

— Divided fx clock frequency of 4, 8, or 64

— Divided fxt clock frequency of 4.

PCON.1 and PCON.0 settings are also connected with the system clock mode control register, SCMOD. If SCMOD.0 = "0", the main system clock is always selected by the PCON.1 and PCON.0 setting; if SCMOD.0 = "1" the subsystem clock is selected.
A system reset sets PCON register values (and SCMOD) to logic zero.

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

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

PCON Bit Settings		**Resulting CPU Operating Mode**
PCON.3	**PCON.2**	
0	0	Normal CPU operating mode
0	1	IDLE
1	0	STOP mode

☞ PROGRAMMING TIP — Setting the CPU Clock

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

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

INSTRUCTION CYCLE TIMES

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

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

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

SYSTEM CLOCK MODE REGISTER (SCMOD)

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

When main system clock is used as clock source, main system clock oscillation can be stopped by STOP instruction or setting SCMOD.3 (not recommended).

When the clock source is subsystem clock, main system clock oscillation is stopped by setting SCMOD.3. SCMOD.0, SCMOD2 and SCMOD.3 cannot be simultaneously modified. Sub-oscillation goes into stop mode only by SCMOD.2. PCON which revokes stop mode cannot stop the sub-oscillation. The stop of sub-oscillation is released only by a system reset.

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

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

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

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

If you have selected fx as the CPU clock, setting SCMOD.3 to "1" will stop main system clock oscillation. But this mode must not be used. To stop main system clock oscillation safely, main oscillation clock should be stopped only by a STOP instruction in main system clock mode.

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

SCMOD Register Bit Settings			Resulting Clock Selection		
SCMOD.3	**SCMOD.2**	**SCMOD.0**	**fx Oscillation**	**fxt Oscillation**	**CPU Clock** (note)
0	0	0	On	On	fx
0	1	0	On	Off	fx
0	0	1	On	On	fxt
1	0	1	Off	On	fxt

NOTE: CPU clock is selected by PCON register settings.

Table 6-4. Main or Sub Oscillation Stop Mode

Mode	Condition	Method to issue Osc Stop	Oscillator's Stop Release Source [2]
Main Oscillation STOP Mode	*Main oscillator runs.* *Sub oscillator runs* (or stops). System is operating with the main clock.	*STOP instruction*: Only main oscillator stops, CPU is in idle mode. Sub oscillator still runs (or stops).	*Interrupts or RESET signal*: After stop mode released, main oscillation starts and oscillation stabilization time is elapsed. And then the CPU operates. Oscillation stabilization time is 1/ {256 x BT clock (fx)}.
		Set SCMOD.3 to "1" [1] Only main oscillator stops, CPU is in idle mode. Sub oscillator still runs (or stops).	Only RESET signal: Interrupt can't start the main oscillation. Therefore, the CPU operation can never be restarted.
	Main oscillator runs. Sub oscillator runs. *System is operating with sub clock.*	STOP instruction: [1] Only main oscillator stops. CPU is in idle mode. *Sub oscillator still runs.*	Basic timer overflow or RESET signal: After the overflow of basic timer [1/ {256 x BT clock (fxt)}], CPU operation and main oscillation automatically start.
		Set SCMOD.3 to "1" Only main oscillator stops. CPU still operates. Sub oscillator still runs.	Set SCMOD.3 to "0" or a system reset.
Sub Oscillation STOP Mode	Main oscillator runs. Sub oscillator runs. *System is operating with the main clock*	Set SCMOD.2 to "1": Main oscillator still runs. CPU operates. Only sub oscillator stops.	Set SCMOD.3 to "0" or a system reset.
	Main oscillator runs (or stops). Sub oscillator runs. *System is operating with sub clock.*	Set SCMOD.2 to "1": Main oscillator still runs (or stops). CPU is in idle mode. Only sub oscillator stops.	Only RESET signal

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

Table 6-5. System Operating Mode Comparison

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

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

SWITCHING THE CPU CLOCK

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

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

NOTE

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

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

This same 'stepped' approach must be taken to switch from a subsystem clock to the main system clock: First, clear SCMOD.3 to "0" to enable main system clock oscillation. Until main osc is stabilized, system clock must not be changed. Then, after a certain number of machine cycles has elapsed, select the main system clock by clearing all SCMOD values to logic zero.

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

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

	AFTER	SCMOD.0 = 0						SCMOD.0 = 1
BEFORE		PCON.1 = 0	PCON.0 = 0	PCON.1 = 1	PCON.0 = 0	PCON.1 = 1	PCON.0 = 1	
SCMOD.0 = 0	PCON.1 = 0 PCON.0 = 0	N/A		1 MACHINE CYCLE		1 MACHINE CYCLE		N/A
	PCON.1 = 1 PCON.0 = 0	8 MACHINE CYCLES		N/A		8 MACHINE CYCLES		N/A
	PCON.1 = 1 PCON.0 = 1	16 MACHINE CYCLES		16 MACHINE CYCLES		N/A		fx / 4fxt MACHINE CYCLE
SCMOD.0 = 1		N/A		N/A		fx / 4fxt (M/C)		N/A

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

☞ PROGRAMMING TIP — Switching Between Main System and Subsystem Clock

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

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

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

```
SUB2MA    BITR     SCMOD.3          ;   Start main system clock oscillation
          CALL     DLY80            ;   Delay 80 machine cycles
          CALL     DLY80            ;   Delay 80 machine cycles
          BITR     SCMOD.0          ;   Switch to main system clock
          RET
```

NOTES

7 INTERRUPTS

OVERVIEW

The KS57C3204 interrupt control circuit has five functional components:

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

Three kinds of interrupts are supported:

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

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

Interrupt Type	Interrupt Name	Corresponding Port Pins
External interrupts	INT0, INT1, INT4	P1.0, P1.1, P1.3
Internal interrupts	INTB, INTT0	Not applicable
Quasi-interrupts	INT2, KS0–KS2	P1.2, KS0–KS2 (P6.1–P6.3)
	INTW	Not applicable

Vectored Interrupts

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

— Interrupt enable flag (IEx)

— Interrupt master enable flag (IME)

— Interrupt request flag (IRQx)

— Interrupt status flags (IS0, IS1)

— Interrupt priority register (IPR)

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

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

When an interrupt occurs, the 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 necessary, you can modify the enable flags during the interrupt service routine. When the interrupt service routine is returned to the main routine by the IRET instruction, the original values saved in the stack are restored and the main program continues program execution with these values.

Software-Generated Interrupts

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

Multiple Interrupts

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

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

Power-Down Mode Release

An interrupt can be used to release power-down mode (stop or idle), but INT0 is possible to release only idle when using fxx/64 clock. 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".

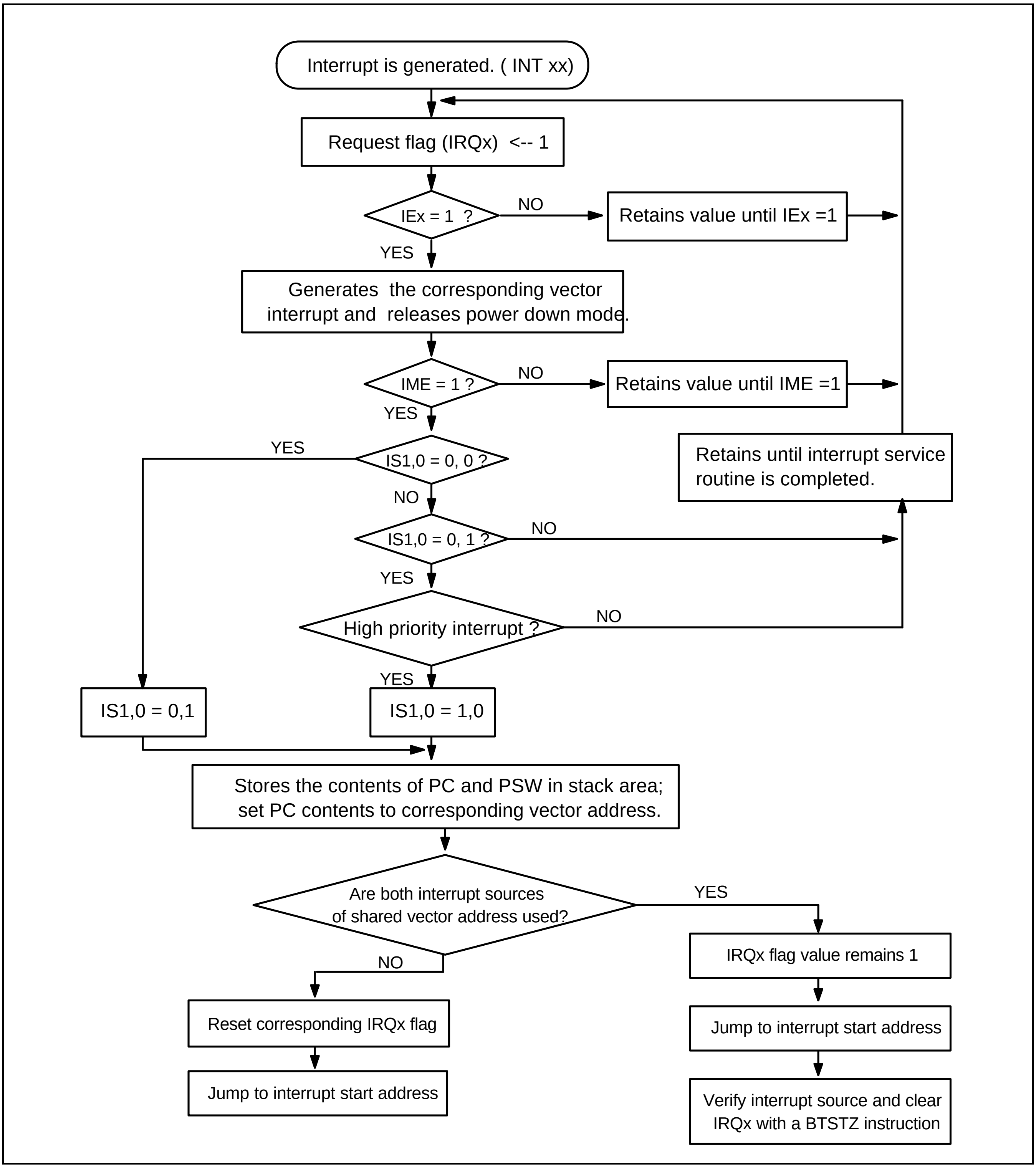

Interrupt is generated. (INT xx)
Request flag (IRQx) <-- 1
IEx = 1 ?
NO
Retains value until IEx =1
YES
Generates the corresponding vector interrupt and releases power down mode.
IME = 1 ?
NO
Retains value until IME =1
YES
YES
IS1,0 = 0, 0 ?
Retains until interrupt service routine is completed.
NO
IS1,0 = 0, 1 ?
NO
YES
High priority interrupt ?
NO
YES
IS1,0 = 0,1
IS1,0 = 1,0
Stores the contents of PC and PSW in stack area; set PC contents to corresponding vector address.
Are both interrupt sources of shared vector address used?
YES
IRQx flag value remains 1
NO
Reset corresponding IRQx flag
Jump to interrupt start address
Jump to interrupt start address
Verify interrupt source and clear IRQx with a BTSTZ instruction

Figure 7-1. Interrupt Execution Flowchart

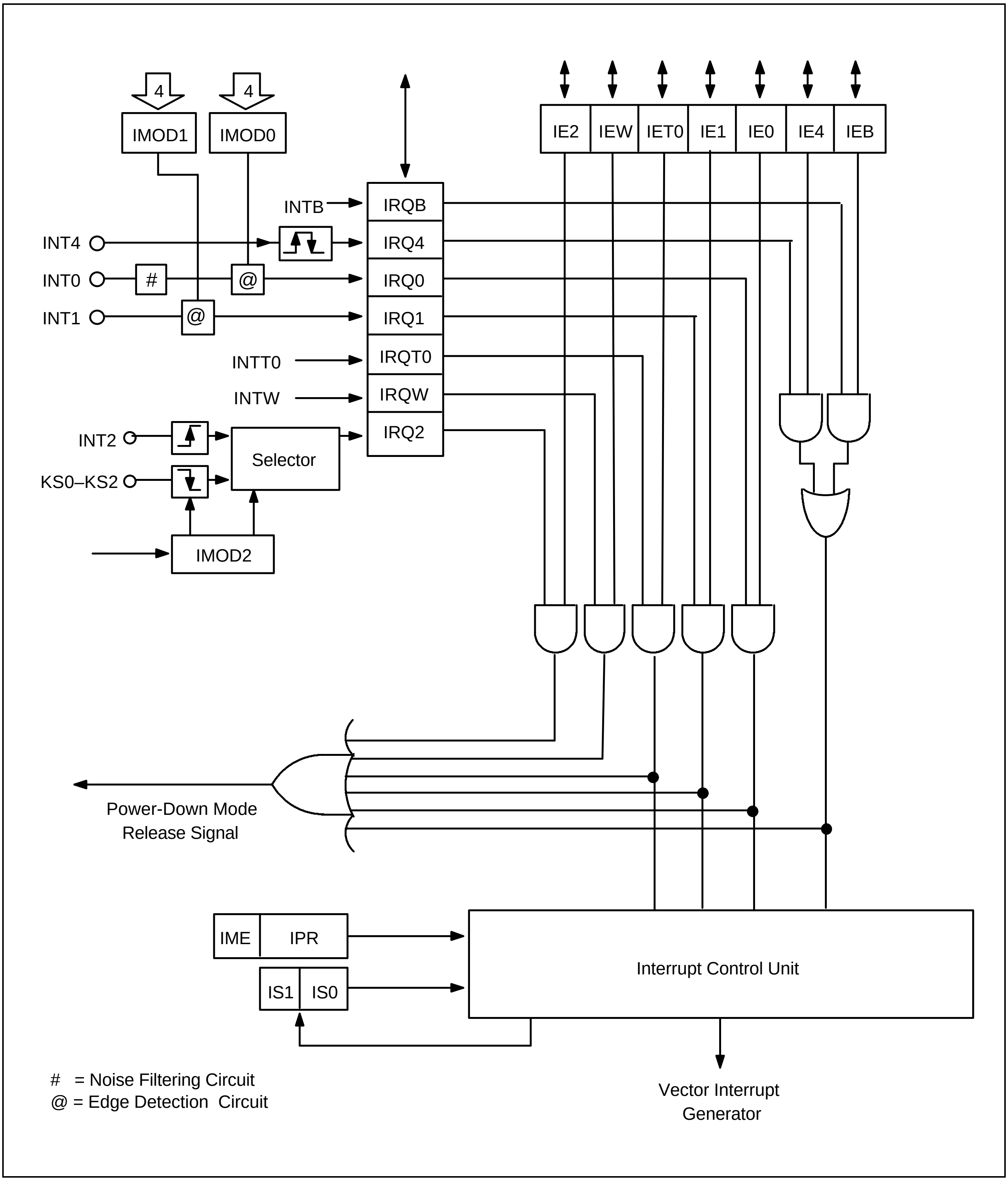


Figure 7-2. Interrupt Control Circuit Diagram

MULTIPLE INTERRUPTS

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

Two-Level Interrupt Handling

Two-level interrupt handling is the standard method for processing multiple interrupts. When the IS1 and IS0 bits of the PSW (FB0H.3 and FB0H.2, respectively) are both logic zero, program execution mode is normal and all interrupt requests are serviced (see Figure 7-3).

Whenever an interrupt request is accepted, IS1 and IS0 are incremented by one, and the values are stored in the stack along with the other PSW bits. After the interrupt routine has been serviced, the modified IS1 and IS0 values are automatically restored from the stack by an IRET instruction.

IS0 and IS1 can be manipulated directly by 1-bit write instructions, regardless of the current value of the enable memory bank flag (EMB). Before you modify an interrupt service flag, however, you must first disable interrupt processing with a DI instruction.

When IS1 = "0" and IS0 = "1", all interrupt service routines are inhibited except for the highest priority interrupt currently defined by the interrupt priority register (IPR).

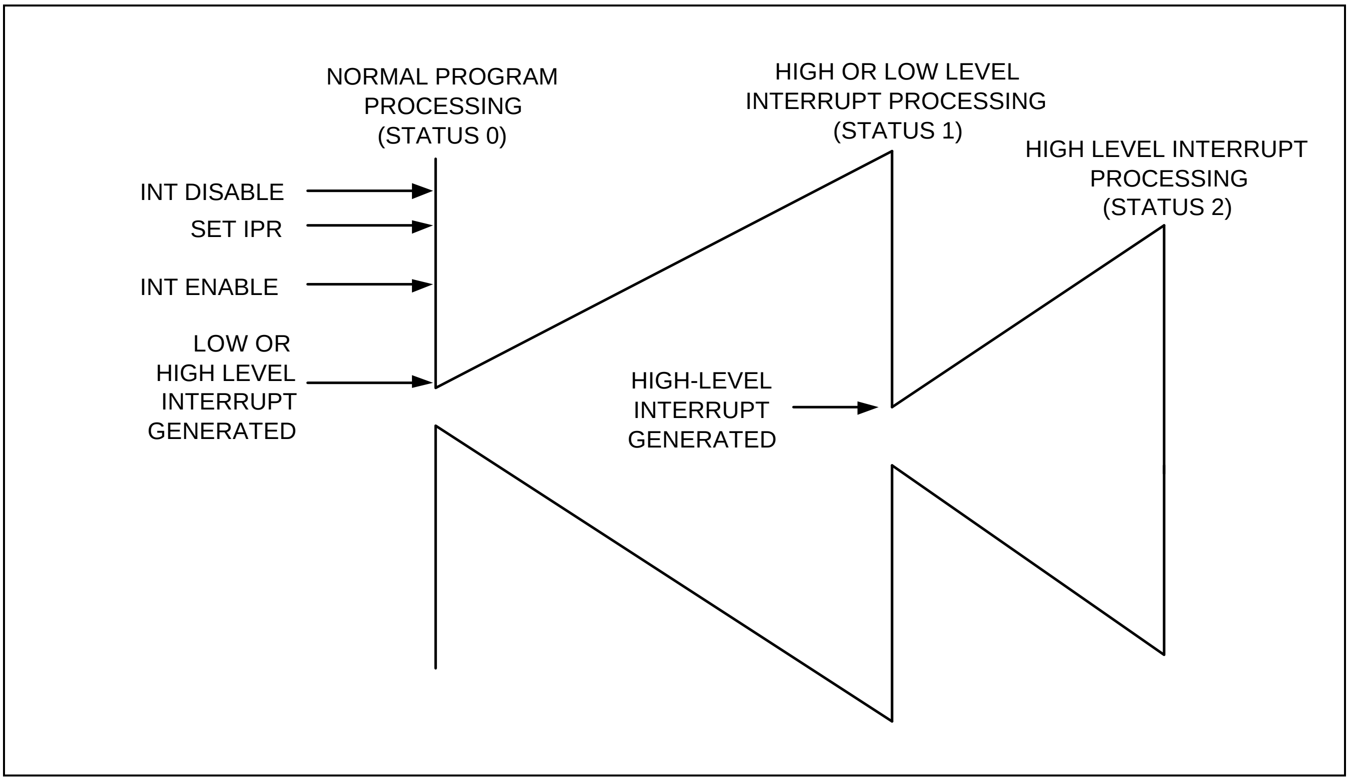

NORMAL PROGRAM PROCESSING (STATUS 0)
HIGH OR LOW LEVEL INTERRUPT PROCESSING (STATUS 1)
HIGH LEVEL INTERRUPT PROCESSING (STATUS 2)
INT DISABLE
SET IPR
INT ENABLE
LOW OR HIGH LEVEL INTERRUPT GENERATED
HIGH-LEVEL INTERRUPT GENERATED

Figure 7-3. Two-Level Interrupt Handling

Multi-Level Interrupt Handling

With multi-level interrupt handling, a lower-priority interrupt request can be executed by manipulating the interrupt status flags, IS0 and IS1 while a high-priority interrupt is being serviced (see Table 7-2).

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

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

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

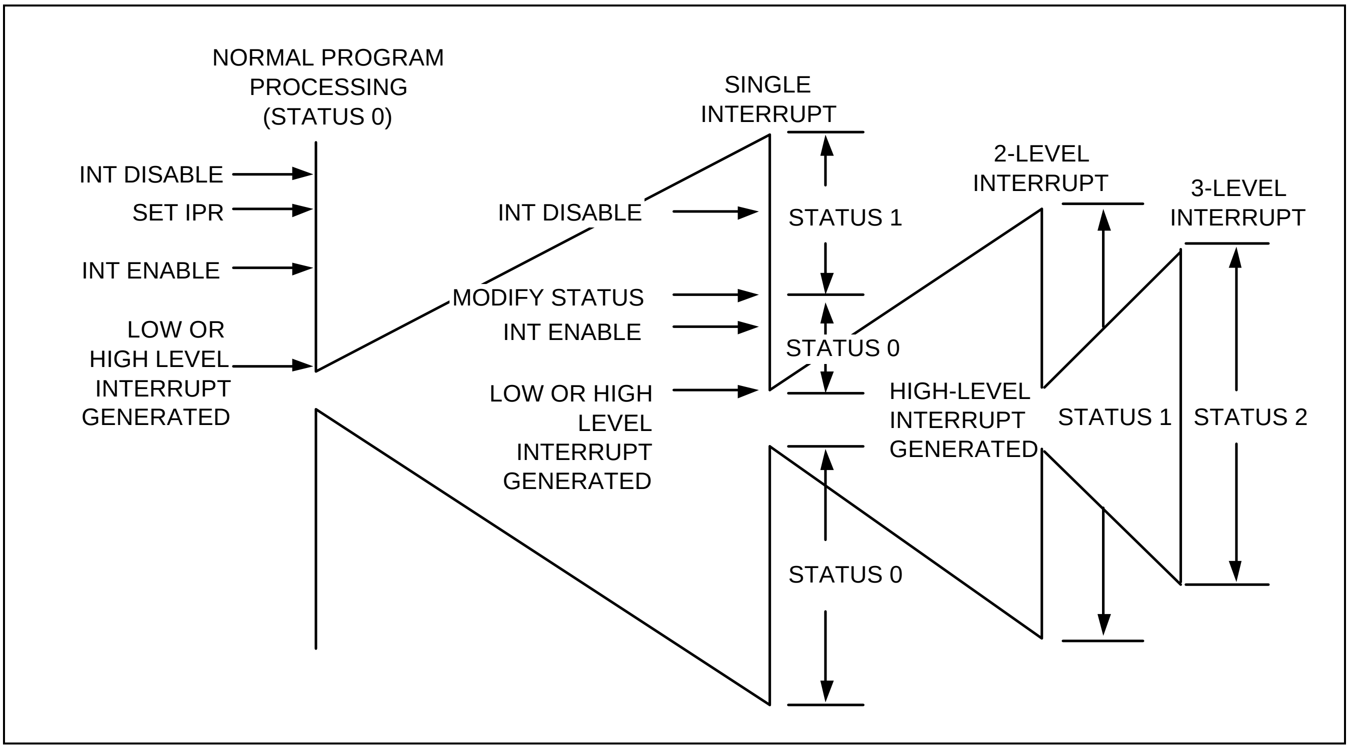


Figure 7-4. Multi-Level Interrupt Handling

INTERRUPT PRIORITY REGISTER (IPR)

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

FB2H	IME	IPR.2	IPR.1	IPR.0

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

Table 7-3. Standard Interrupt Priorities

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

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 (mapped FB2H.3) 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	Normal interrupt handling according to default priority settings
0	0	1	Process INTB and INT4 interrupt at highest priority
0	1	0	Process INT0 interrupt at highest priority
0	1	1	Process INT1 interrupt at highest priority
1	0	0	Not used in the KS57C3204
1	0	1	Process INTT0 interrupt at highest priority
1	1	0	Not used in the KS57C3204
1	1	1	

NOTE: During normal interrupt processing, interrupts are processed in the order in which they occur. If two or more interrupt requests are received simultaneously, the priority level is determined according to the standard interrupt priorities in Table 7-3 (the default priority assigned by hardware when the lower three IPR bits = "0"). In this case, the higher-priority interrupt request is serviced and the other interrupt is inhibited. Then, when the high-priority interrupt is returned from its service routine by an IRET instruction, the inhibited service routine is started.

☞ PROGRAMMING TIP — Setting the INT Interrupt Priority

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

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

EXTERNAL INTERRUPT 0 AND 1 MODE REGISTERS (IMOD0 and IMOD1)

The following components are used to process external interrupts at the INT0 and INT1 pins:

- Noise filtering circuit for INT0
- Edge detection circuit
- Two mode registers, IMOD0 and IMOD1

The mode registers are used to control the triggering edge of the input signal. IMOD0 and IMOD1 settings let you choose either the rising or falling edge of the incoming signal as the interrupt request trigger. The INT4 interrupt is an exception since its input signal generates an interrupt request on both rising and falling edges.

FB4H	IMOD0.3	"0"	IMOD0.1	IMOD0.0
FB5H	"0"	"0"	"0"	IMOD1.0

IMOD0 and IMOD1 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. IMOD0, 1 and 2 Register Organization

IMOD0	**IMOD0.3**	**0**	**IMOD0.1**	**IMOD0.0**	**Effect of IMOD0 Settings**
	0				Select CPU clock for sampling
	1				Select fxx/64 sampling clock
			0	0	Rising edge detection
			0	1	Falling edge detection
			1	0	Both rising and falling edge detection
			1	1	IRQ0 flag cannot be set to "1"

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

EXTERNAL INTERRUPT 0 AND 1 MODE REGISTERS (Continued)

When a sampling clock rate of fxx/64 is used for INT0, an interrupt request flag must be cleared before 16 machine cycles have elapsed. Since the INT0 pin has a clock-driven noise filtering circuit built into it, please take the following precautions when you use it:

— To trigger an interrupt, the input signal width at INT0 must be at least two times wider than the pulse width of the clock selected by IMOD0.

— You can use INT0 to release IDLE mode, when fxx/64 is selected as a sampling clock.

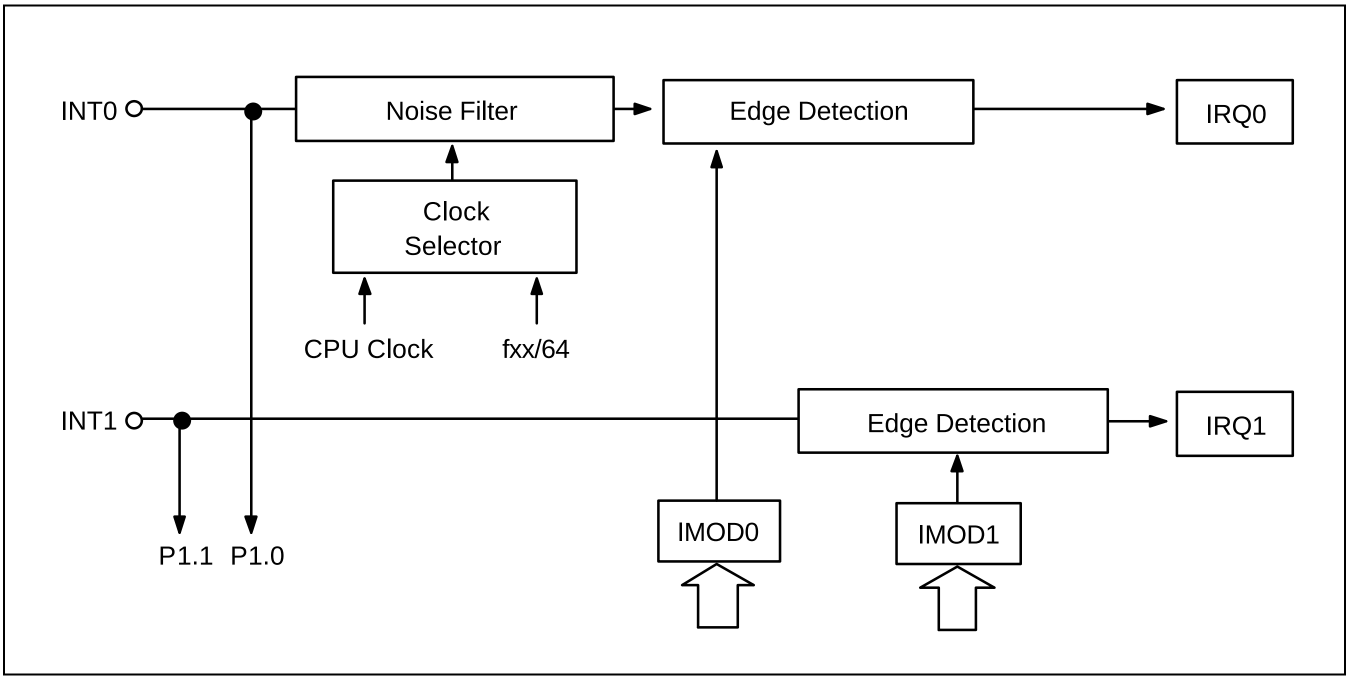


Figure 7-5. Circuit Diagram for INT0 and INT1 Pins

When modifying the IMOD 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 IMOD register.
3. Clear all relevant interrupt request flags.
4. Enable the interrupt by setting the appropriate IEx flag.
5. Enable all interrupts with an EI instructions.

EXTERNAL INTERRUPT 2 MODE REGISTER (IMOD2)

The mode register for external interrupt 2 at the KS0–KS2 pins, IMOD2, is addressable only by 4-bit write instructions. A system reset clears all IMOD2 bits to logic zero.

FB6H	"0"	"0"	"0"	IMOD2.0

If a rising or falling edge is detected the KS0–KS2 pin or P1.2 by the IMOD2 register, the IRQ2 flag is set to logic one and a release signal for power-down mode is generated. Since INT2 is a quasi-interrupt, the interrupt request flag (IRQ2) must be cleared by software.

Table 7-6. IMOD2 Register Bit Settings

IMOD2	**0**	**0**	**0**	**IMOD2.0**	**Effect of IMOD2 Settings**
				0	Select rising edge at INT2 pin
				1	Select falling edge at KS0–KS2

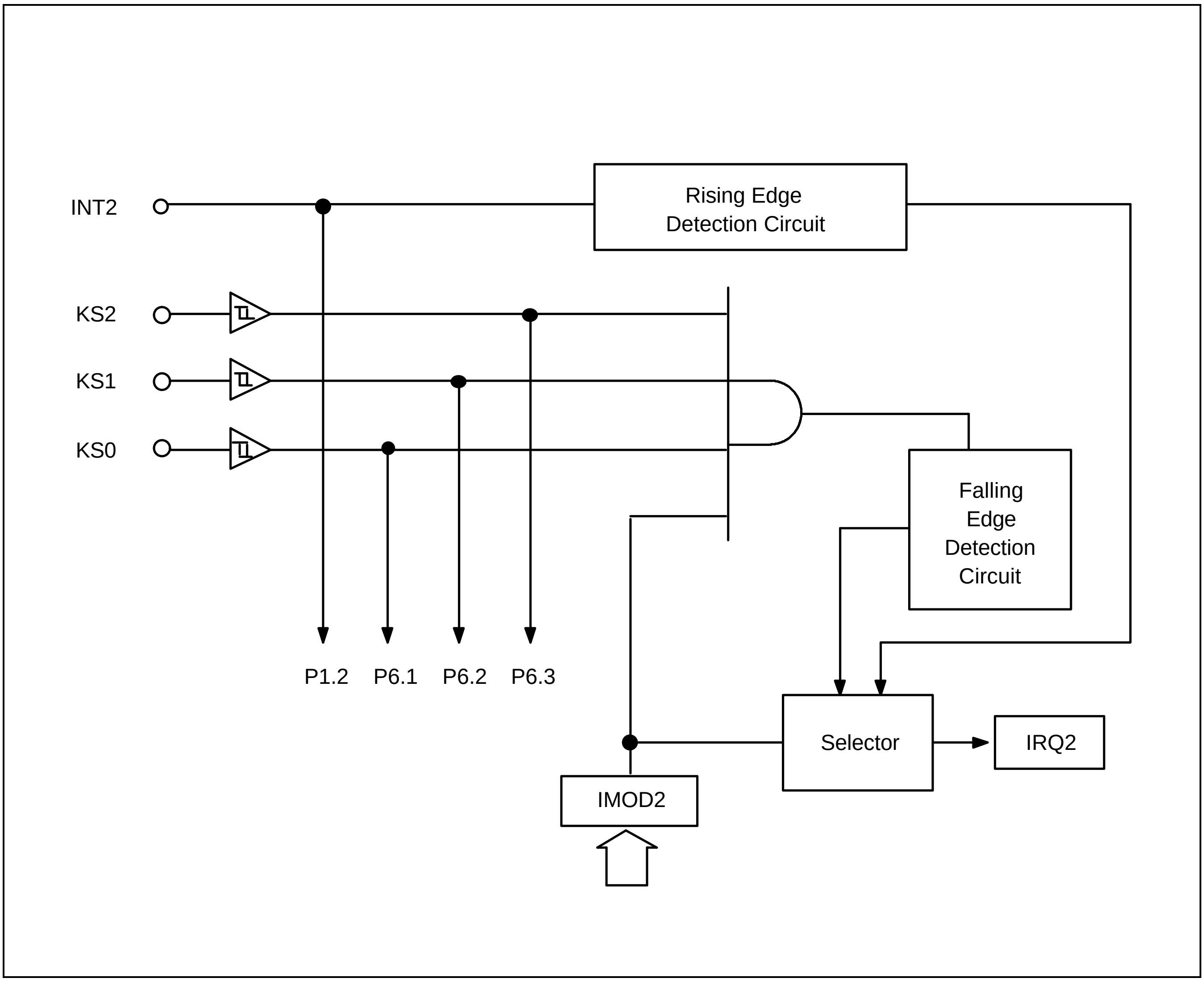


Figure 7-6. Circuit Diagram for INT2 and KS0–KS2 Pins

INTERRUPT FLAGS

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

Interrupt Master Enable Flag (IME)

The interrupt master enable flag, IME, enables or disables all interrupt processing. Therefore, even when an IRQx flag is set and its corresponding IEx flag is enabled, the interrupt service routine is not executed until the IME flag is set to logic one.

The IME flag is located in the IPR register (IPR.3). It can be directly be manipulated by EI and DI instructions, regardless of the current value of the enable memory bank flag (EMB).

FB2H

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

Interrupt Enable Flags (IEx)

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

Interrupt enable flags can be read, written, or tested directly by 1-bit instructions. IEx flags can be addressed directly at their specific RAM addresses, despite the current value of the enable memory bank (EMB) flag.

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

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

NOTES:

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

Interrupt Request Flags (IRQx)

Interrupt request flags are read/write addressable by 1-bit or 4-bit instructions. IRQx flags can be addressed directly at their specific RAM addresses, regardless of the current value of the enable memory bank (EMB) flag.

When a specific IRQx flag is set to logic one, the corresponding interrupt request is generated. The flag is then automatically cleared to logic zero when the interrupt has been serviced. Exceptions are the watch timer interrupt request flags, IRQW, and the external interrupt 2 flag IRQ2, which must be cleared by software after the interrupt service routine has executed. IRQx flags are also used to execute interrupt requests from software. In summary, follow these guidelines for using IRQx flags:

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

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

— When only one interrupt is enabled, the IRQx flag is automatically cleared when entering to the interrupt service routine.

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

Table 7-8. Interrupt Request Flag Conditions and Priorities

Interrupt Source	Internal / External	Pre-condition for IRQx Flag Setting	Interrupt Priority	IRQ Flag Name
INTB	I	Reference time interval signal from basic timer	1	IRQB
INT4	E	Both rising and falling edges detected at INT4	1	IRQ4
INT0	E	Rising or falling edge detected at INT0 pin	2	IRQ0
INT1	E	Rising or falling edge detected at INT1 pin	3	IRQ1
INTT0	I	Signals for TCNT0 and TREF0 registers match	4	IRQT0
INT2 (note) (KS0–KS2)	E	Rising edge detected at INT2 or falling edge detected at KS0–KS2	–	IRQ2
INTW	I	Time interval of 0.5 s or 3.19 ms	–	IRQW

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

NOTES

8 POWER-DOWN

OVERVIEW

The KS57C3204 microcontroller has two power-down modes to reduce power consumption: idle and stop. Idle mode is initiated by the IDLE instruction and stop mode by the instruction STOP. (Several NOP instructions must always follow an IDLE or STOP instruction in a program.) In idle mode, the CPU clock stops while peripherals and the oscillation source continue to operate normally.

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

In main stop mode, main system clock oscillation is halted (assuming main clock is selected as system clock and it is currently operating), and peripheral hardware components are powered-down. In sub stop mode, (assuming sub clock is selected) sub system clock oscillation is halted by setting SCMOD.2 to “1”. The effect of stop mode on specific peripheral hardware components — CPU, basic timer, timer/ counter 0, watch timer, and LCD controller — 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 main stop modes are terminated either by a RESET, or by an interrupt which is enabled by the corresponding interrupt enable flag, IEx. When power-down mode is terminated by RESET, a normal reset operation is executed. Assuming that both the interrupt enable flag and the interrupt request flag are set to "1", power-down mode is released immediately upon entering power-down mode. Sub stop mode can be terminated by RESET only.

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 starts immediately after the instruction issuing a request to enter power-down mode is executed. The interrupt request flag remains set to logical one.

— If the IME flag = "1", two instructions are executed after the power-down mode release and the vectored interrupt is then initiated. However, when the release signal is caused by INT2 or INTW, the operation is identical to the IME = "0" condition. Assuming that both interrupt enable flag and interrupt request flag are set to "1", the release signal is generated when power-down mode is entered.

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

Mode	Main Stop	Main and Sub Stop		Idle
System is operating with	Main clock (fx)	Sub clock (fxt)	Main clock (fx)	Main (fx) or sub clock (fxt)
Instruction	STOP	Setting SCMOD.2 to “1”, after setting SCMOD.3 to “1”	STOP, after setting SCMOD.2 to “1”	IDLE
Clock oscillators	Only main clock oscillation stops	Both main and sub clock oscillation stop		Only CPU clock stops. (1)
Basic timer	Operation stops	Operation stops		Basic timer operates.
Timer 0	Operation stops	Operation stops		Timer 0 operates.
Watch timer	Operates only if sub clock (fxt) is selected as timer clock.	Operation stops		Watch timer operates.
A/D converter	Operation stops	Operation stops		Operation stops
Frequency counter	Operation stops. If Port2.2 and Port2.3 are used as only frequency input, the feedback resistors are off and pull-down is on.	Operation stops. If Port2.2 and Port2.3 are used as only frequency input, the feedback resistors are off and pull-down is on.		Operation stops. If Port2.2 and Port2.3 are used as only frequency input, the feedback resistors are off and pull-down is on.
LCD controller	Operates only if sub clock (fxt) is selected as watch timer clock	Operation stops.		Operates.
External interrupts	INT1, INT2 and INT4 are acknowledged	INT1, INT2 and INT4 is not serviced.	INT1, INT2 and INT4 are acknowledged	INT0, INT1, INT2 and INT4 are acknowledged (2)
CPU	All CPU operations are disabled.			
Mode release signal	Interrupt request signals (except INT0) pre-enabled by IEx or RESET input.	Only RESET input	Interrupt request signals pre-enabled by IEx(2) or RESET input.	

NOTE: 1. Main and sub clock oscillation continues.
2. INT0 can be serviced in the IDLE mode only when fxx/64 is selected as sampling clock source.

IDLE MODE TIMING DIAGRAMS

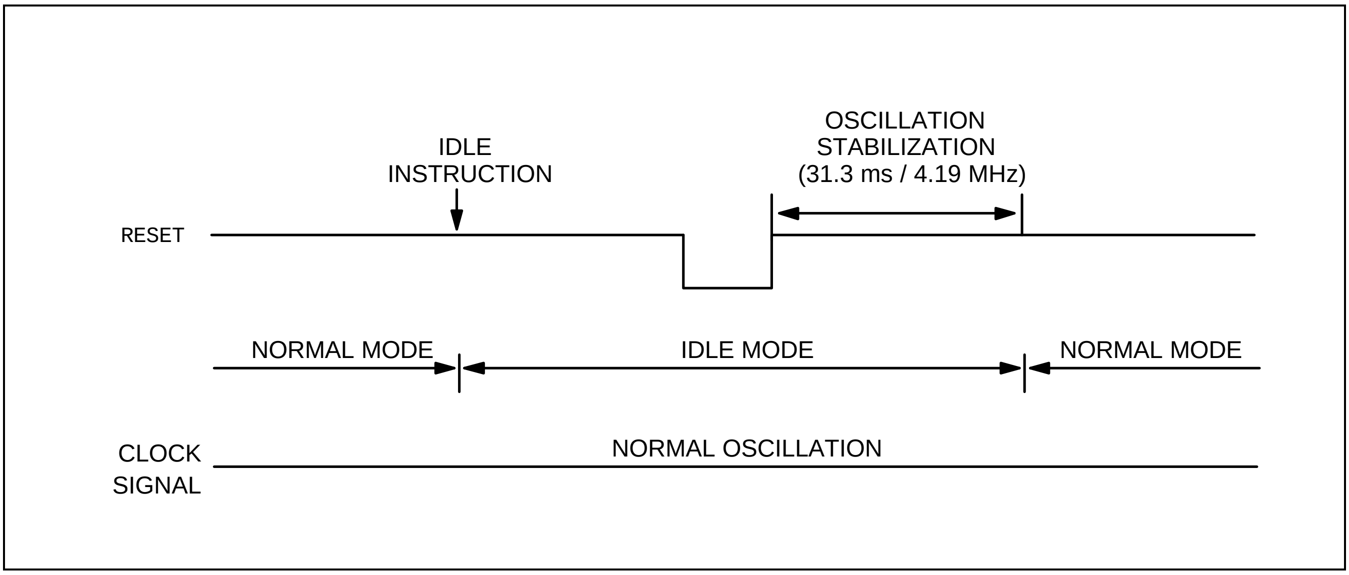


Figure 8-1. Timing When Idle Mode is Released by RESET

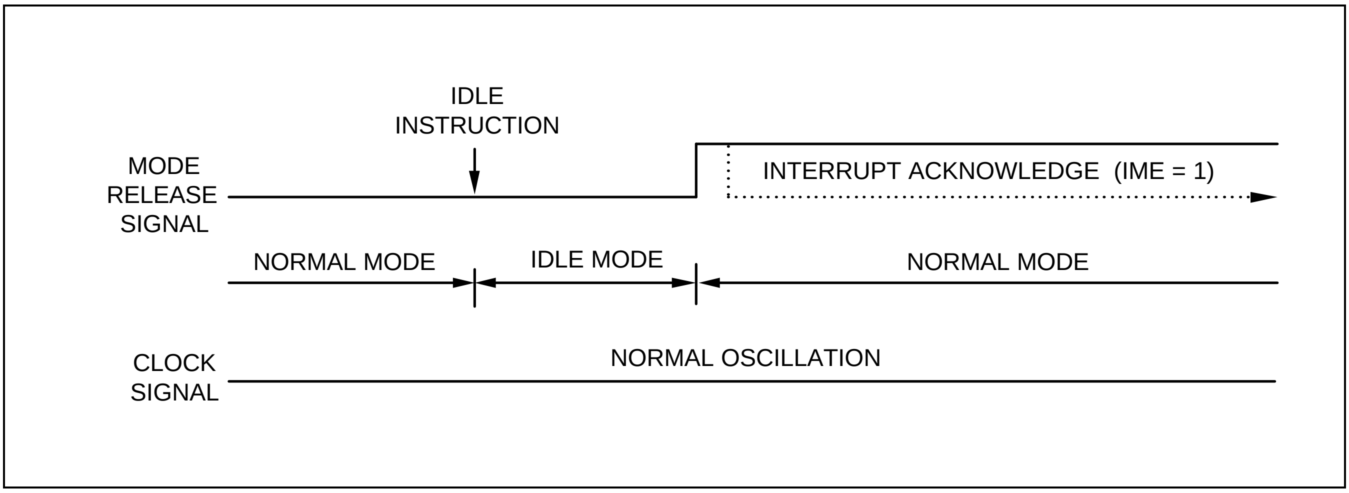


Figure 8-2. Timing When Idle Mode is Released by an Interrupt

STOP MODE TIMING DIAGRAMS

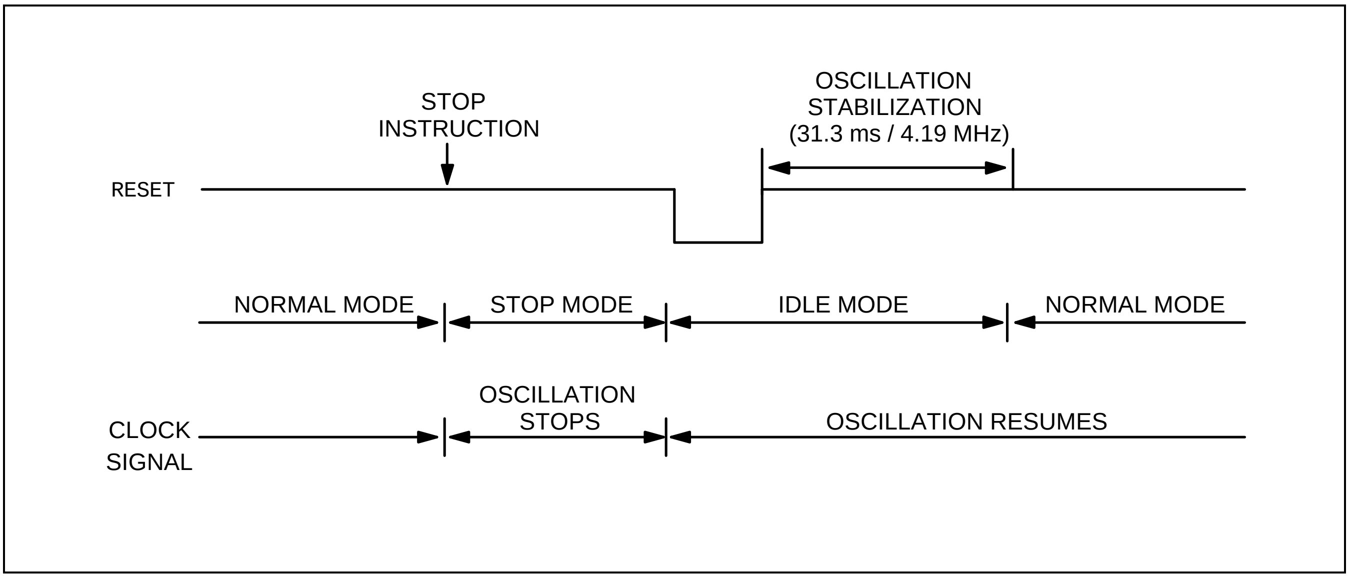


Figure 8-3. Timing When Stop Mode is Released by RESET

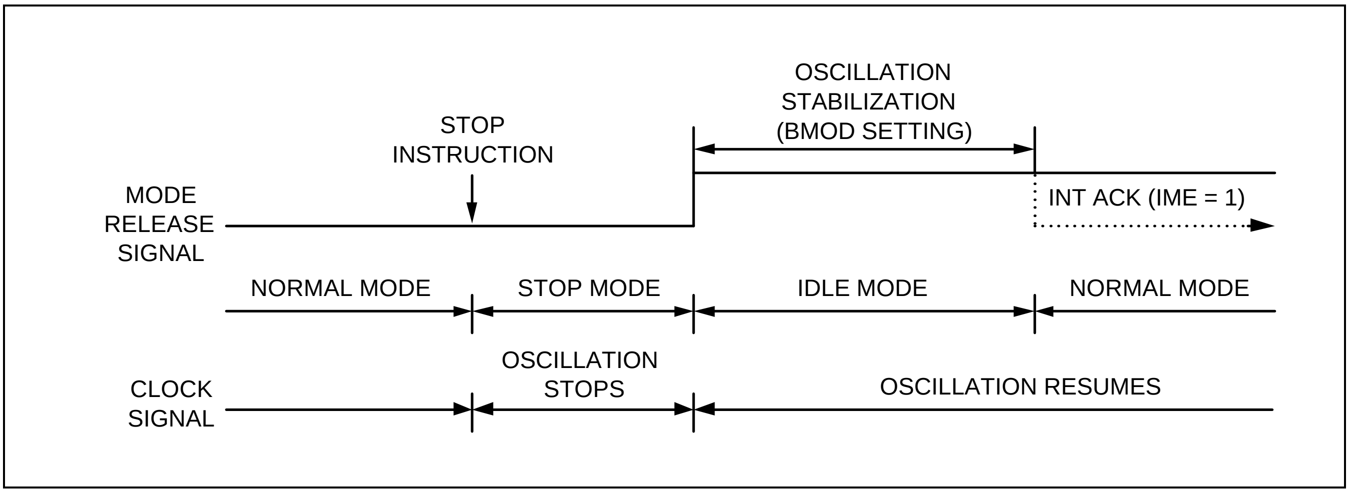


Figure 8-4. Timing When Main Stop or Main/Sub Stop Mode is Release by an Interrupt

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

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

```
KEYCLK   DI
         CALL     MA2SUB             ;  Main system clock → subsystem clock switch subroutine
         SMB      15
         LD       EA,#00H
         LD       P4,EA              ;  All key strobe outputs to low level
         LD       A,#1H
         LD       IMOD2,A            ;  Select KS0–KS2 interrupt
         BITS     IEW
         BITS     IE2
         BITR     IRQW
         BITR     IRQ2
         SMB      0

CLKS1    CALL     WATDIS             ;  Execute clock and display changing subroutine
         BTSTZ    IRQ2
         JR       CIDLE
         CALL     SUB2MA             ;  Subsystem clock→main system clock switch subroutine
         EI
         RET
CIDLE    IDLE                        ;  Engage idle mode
         NOP
         NOP
         NOP
         JPS      CLKS1
```

NOTE

You must execute three NOP instructions after IDLE and STOP instructions, to avoid flowing of leakage current due to the floating state in the internal bus.

PORT PIN CONFIGURATION FOR POWER-DOWN MODE

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 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 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.
3. Disable the pull-up resistors of input pins connected to the external device by making the necessary modifications to the PUMOD register.
4. 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.
5. 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 Reducing Power Consumption

Pin/Share Pin Names	Recommended Connection
P1.0/INT0 P1.1/INT1 P1.2/INT2 P1.3/INT4	Input mode: connect to V_{DD} Output mode: No connection
P2.0 P2.1 P2.2/FMF P2.3/AMF	Connect to V_{DD} P2.2 and P2.3 are also a normal input mode
P3.0/ADC0 P3.1/ADC1 P3.2/ADC2 P3.3/ADC3	Connect to V_{DD}
P4.0–P4.3 P5.0–P5.3	Input mode: connect to V_{DD} Output mode: No connection
P6.0/BUZ P6.1/KS0 P6.2/KS1 P6.3/KS2	Input mode: connect to V_{DD} Output mode: No connection
P8.0/SEG27–P9.3/SEG20	No connection (1)
SEG0–SEG19 COM0–COM3	No connection
XT_{in}	Connect XT_{in} to V_{SS} and set SCMOD.2 to “1”
XT_{out}	No connection
TEST	Connect to V_{SS}

NOTES:

1. Used as segment

NOTES

9 RESET

OVERVIEW

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

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

- Carry flag
- Data memory values (except for the addresses, 0F8H to 0FDH)
- General-purpose registers E, A, L, H, X, W, Z, and Y

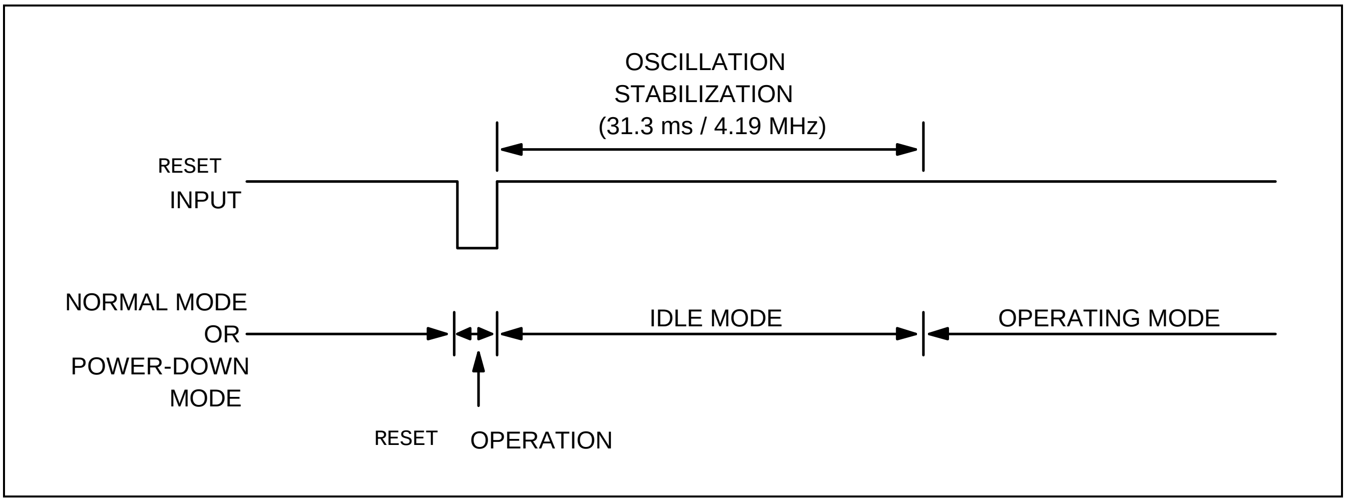


Figure 9-1. Timing for Oscillation Stabilization After RESET

HARDWARE REGISTER VALUES AFTER A SYSTEM RESET

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

Table 9-1. Hardware Register Values After a system reset

Hardware Component or Subcomponent	If reset Occurs During Operating Mode	If RESET Occurs After Power-On
Program counter (PC)	PC11–8 ← ROM [00H], PC7–0 ← ROM [01H]	PC11–8 ← ROM [00H], PC7–0 ← ROM [01H]
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 E, A, L, H, X, W, Z, Y	Values retained	Undefined
General-purpose registers	Values retained (note)	Undefined
Bank selection registers (SMB, SRB)	0, 0	0, 0
BSC register (BSC0–BSC3)	0	0
Clocks:		
Power control register (PCON)	0	0
System clock control reg (SCMOD)	0	0
Interrupts:		
Interrupt request flags (IRQx)	0	0
Interrupt enable flags (IEx)	0	0
Interrupt priority flag (IPR)	0	0
Interrupt master enable flag (IME)	0	0
INT0 mode register (IMOD0)	0	0
INT1 mode register (IMOD1)	0	0
INT2 mode register (IMOD2)	0	0
Watch-Dog Timer:		
WDT mode register (WDMOD)	A5H	A5H
WDT clear flag (WDTCF)	0	0

Table 9-1. Hardware Register Values After a system reset (Continued)

Hardware Component or Subcomponent	If reset Occurs During Operating Mode	If RESET Occurs During Power-On
I/O Ports:		
Output buffers	Off	Off
Output latches	0	0
ADC and port control reg (APCON)	0	0
Port mode registers (PMGx)	0	0
Pull-up resistor mode reg (PUMOD)	0	0
Basic Timer:		
Count register (BCNT)	Undefined	Undefined
Mode register (BMOD)	0	0
Timer 0:		
Count registers (TCNT0)	0	0
Reference register (TREF0)	FFH	FFH
Mode register (TMOD0)	0	0
Watch Timer:		
Watch timer mode register (WMOD)	0	0
LCD Driver/Controller:		
LCD mode register (LMOD)	0	0
LCD control register (LCON)	0	0
Display data memory	Values retained (note)	Undefined
Frequency counter		
FC mode register (FCMOD)	0	0
FC control register (FCCON)	0	0
FC registers (FCNT2, FCNT1, FCNT0)	0	0
A/D converter:		
A/D mode register (ADMOD)	0	0
A/D control register (AFLAG)	0	0
A/D convert data register (ADATA)	0	0

Note; The values of the addresses, 0F8H to 0FDH, are not retained when a system reset is generated.

NOTES

10 I/O PORTS

OVERVIEW

The KS57C3204 has 8 ports. There are total of 8 input pins, 8 output pins and 16 configurable I/O pins, for a maximum number of 32 pins.

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

Port Mode Flags

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

Pull-Up Resistor Mode Register (PUMOD)

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

Table 10-1. I/O Port Overview

Port	I/O	Pins	Pin Names	Address	Function Description
1	I/O	4	P1.0–P1.3	FF1H	4-bit I/O port. 1-bit and 4-bit read/write and test are possible. 4-bit pull-up resistors are software assignable. Each pin can be specified as input or output port.
2	I	4	P2.0–P2.3	FF2H	4-bit input port. 1-bit and 4-bit read and test are possible. 4-bit pull-up resistors are software assignable.
3	I	4	P3.0–P3.3	FF3H	4-bit input port. 1-bit and 4-bit read and test are possible. 4-bit pull-up resistors are software assignable.
4, 5	I/O	8	P4.0–P4.3 P5.0–P5.3	FF4H FF5H	4-bit I/O ports. N-channel open-drain output up to 5 volts. 1-bit, 4-bit, and 8-bit read/write and test are possible. Ports 4 and 5 can be paired to support 8-bit data transfer. Pull-up resistors are software assignable
6	I/O	4	P6.0–P6.3	FF6H	4-bit I/O port. 1-bit and 4-bit read/write and test are possible. 4-bit pull-up resistors are software assignable. Each pin can be specified as input or output port.
8, 9	O	8	P8.0–P8.3 P9.0–P9.3	FF8H FF9H	4-bit output ports. 1-bit and 4-bit write and test are possible. Port 8 and 9 can be paired for 8-bit data transfer.

Table 10-2. Port Pin Status During Instruction Execution

Instruction Type	Example	Input Mode Status	Output Mode Status
1-bit test 1-bit input 4-bit input 8-bit input	BTST P1.1 LDB C,P1.3 LD A,P1 LD EA,P4	Input or test data at each pin	Input or test data at output latch
1-bit output	BITR P1.3	Output latch contents undefined	Output pin status is modified
4-bit output 8-bit output	LD P1,A LD P4,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 to input or output mode by setting or clearing the corresponding I/O buffer.

For convenient program reference, PM flags are organized into two groups — PMG0 and PMG1 as shown in Table 10-3. They are addressable by 8-bit write instructions only.

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

Table 10-3. Port Mode Group Flags

PM Group ID	Address	Bit 3/7	Bit 2/6	Bit 1/5	Bit 0/4
PMG0	FE6H	"0"	"0"	"0"	"0"
	FE7H	PM1.3	PM1.2	PM1.1	PM1.0
PMG1	FE8H	PM5	PM4	“0”	“0”
	FE9H	PM6.3	PM6.2	PM6.1	PM6.0

NOTE: If bit = "0", the corresponding pin is set to input mode. All flags are cleared to "0" after a system reset. If bit = "1", the pin is set to output mode.

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

Configure ports 4, 5 and 6 as an output port:

```
BITS    EMB
SMB     15
LD      EA,#0F4H
LD      PMG1,EA             ;  P4 and P6 ← Output, P5 ← Input
```

PULL-UP RESISTOR MODE REGISTER (PUMOD)

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

PUMOD is addressable by 8-bit write instructions only. A system reset clears PUMOD register values to logic zero, automatically disconnecting all software-assignable port pull-up resistors.

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

PUMOD ID	Address	Bit 3	Bit 2	Bit 1	Bit 0
PUMOD	FDCH	PUR.3	PUR.2	PUR.1	“0”
	FDDH	“0”	PUR.6	PUR.5	PUR.4

NOTE: When bit = "1", a pull-up resistor is assigned to the corresponding I/O port: PUMOD.1 for port 1, PUMOD.2 for port 2, and so on.

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

P2 and P3 are enabled to have pull-up resistors.

```
BITS     EMB
SMB      15
LD       EA,#0CH
LD       PUMOD,EA            ;   Enable P2 and P3 to have pull-up resistors
```

ADC AND PORT CONTROL REGISTER (APCON)

FAEH

.3	.2	.1	.0	Effect of Bit Settings
0	0	0	0	Set P3 to connect the normal input
Other settings				Each bit corresponds with P3.0, P3.1, P3.2, and P3.3 respectively. If the specific bits are set to logic "1", the corresponding pins are connected to ADC block, but disconnected from the normal input and automatically the pull-up registers off.

NOTE: All bits are cleared to "0" after a system reset.

N-CHANNEL OPEN-DRAIN MODE REGISTER (PNE)

The N-channel, open-drain mode register, PNE, is used to configure port4, 5 to N-channel open-drain modes or push-pull modes.

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

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

Table 10-5. N-Channel Open Drain Mode Register (PNE) Setting

ID	Address	Bit 3	Bit 2	Bit 1	Bit 0
PNE	FD6H	PNE4.3	PNE4.2	PNE4.1	PNE4.0
	FD7H	PNE5.3	PNE5.2	PNE5.1	PNE5.0

PIN ADDRESSING FOR OUTPUT PORT 8, 9

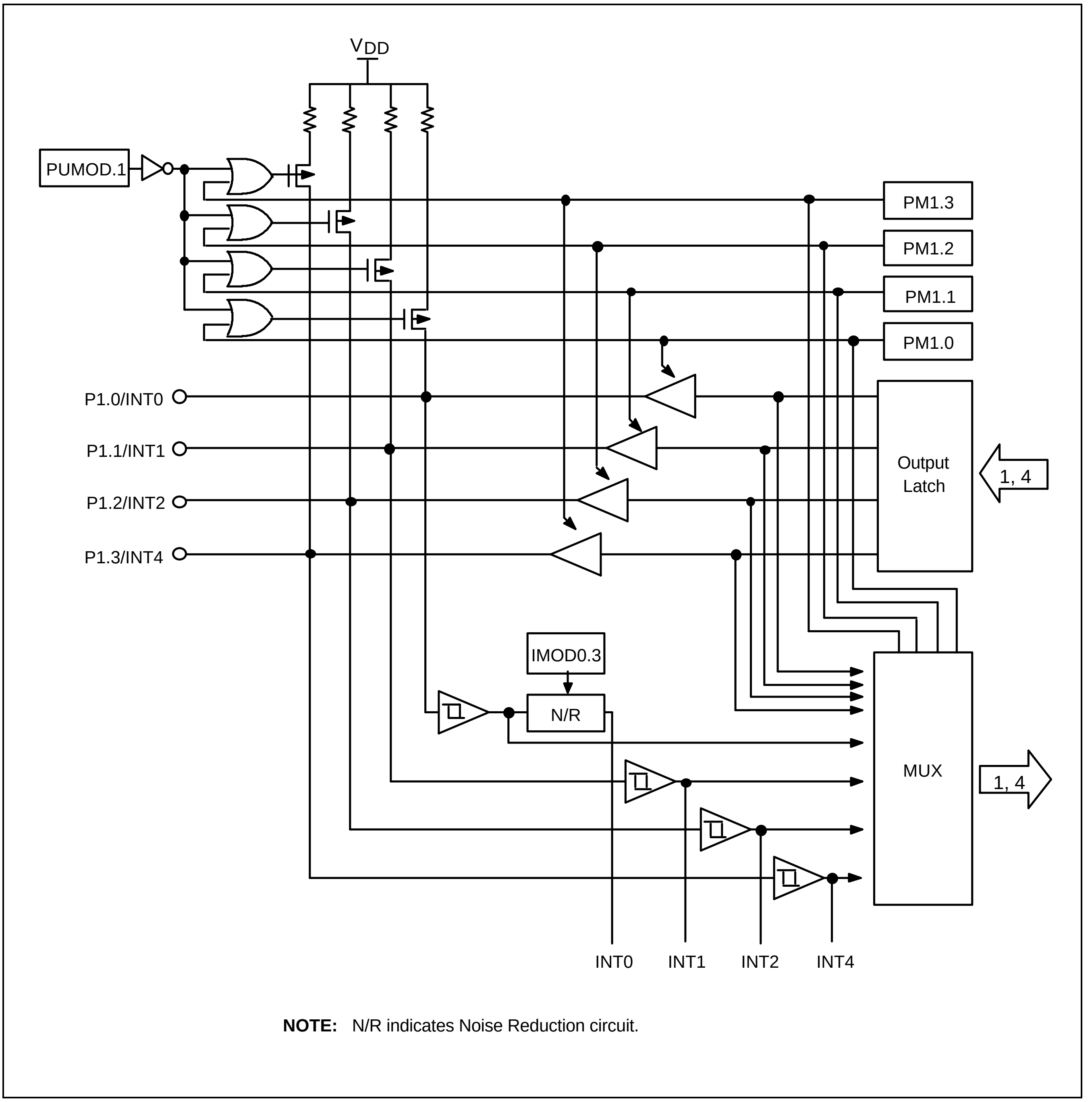
The buffer addresses for the port 8, 9 pins are located in both bank0 and bank15. To output the port 8, 9 in bank15, use the setting SMB = 15. Otherwise, to output SEG20–27 in bank0, use the setting EMB = 1 and SMB = 0. The LCD mode register, LMOD, is used to control whether the pin outputs are the SEG20–27 or the port 8, 9 data.

Table 10-5. LMOD.7 and LMOD.6 Setting for Port 8, 9 Output Control

LMOD.7	LMOD.6	Port 8 Pin Output	Port 9 Pin Output
0	0	SEG27 (0FFH)–SEG24 (0FCH)	SEG23 (0FBH)–SEG20 (0F8H)
0	1	SEG27 (0FFH)–SEG24 (0FCH)	P9.0–P9.3 (FF9H)
1	0	P8.0–P8.3 (FF8H)	SEG23 (0FBH)–SEG20 (0F8H)
1	1	P8.0–P8.3 (FF8H)	P9.0–P9.3 (FF9H)

Locations that are unused for LCD or port I/O can be used as normal data memory. After a system reset, the values connected in the port 8, 9 data are left undetermined.

PORT 1 CIRCUIT DIAGRAM



NOTE: N/R indicates Noise Reduction circuit.

Figure 10-1. Port 1 Circuit Diagram

PORT 2 CIRCUIT DIAGRAM

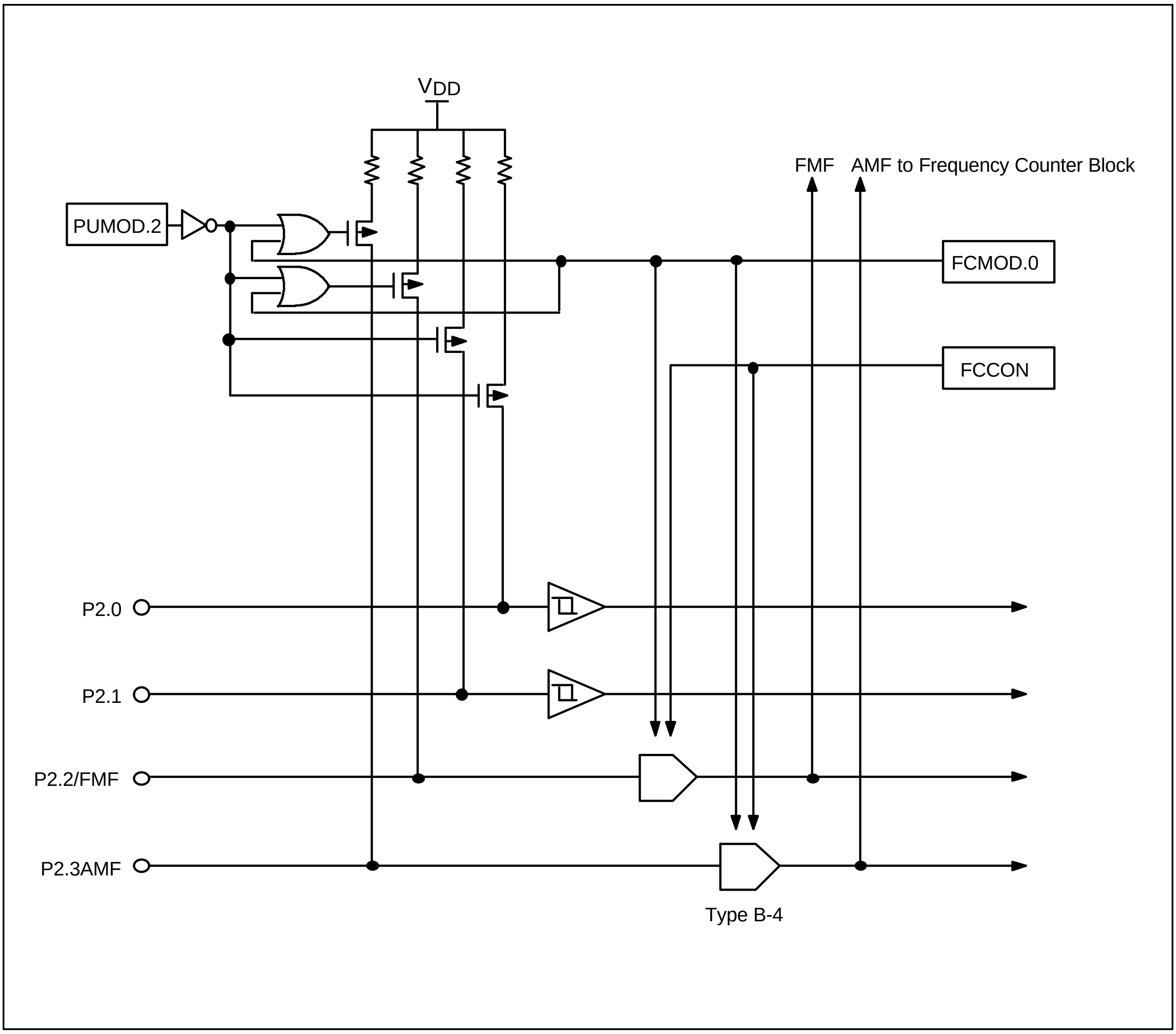


Figure 10-2. Port 2 Circuit Diagram

PORT 3 CIRCUIT DIAGRAM

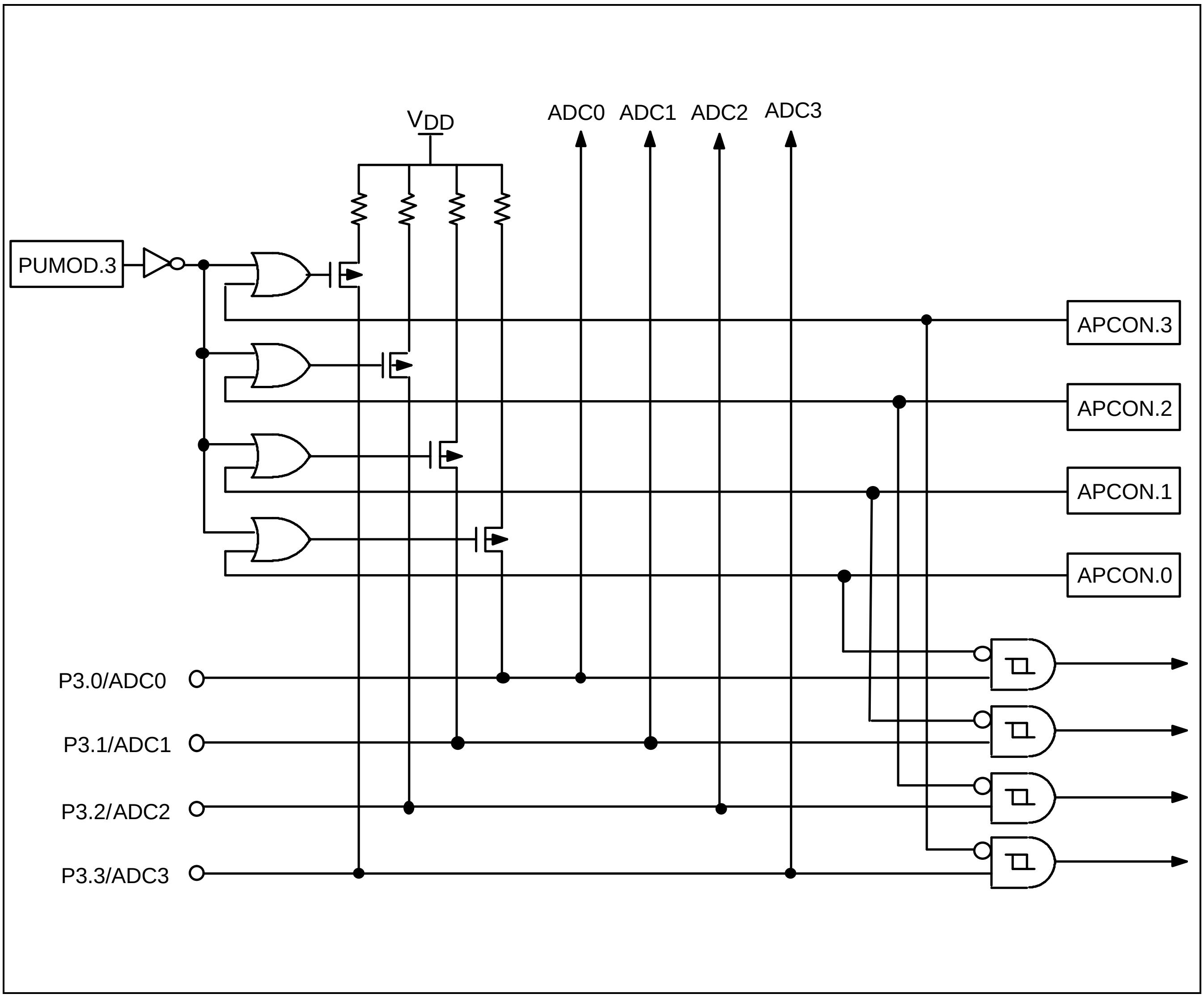


Figure 10-3. Ports 3 Circuit Diagram

PORTS 4, 5 CIRCUIT DIAGRAM

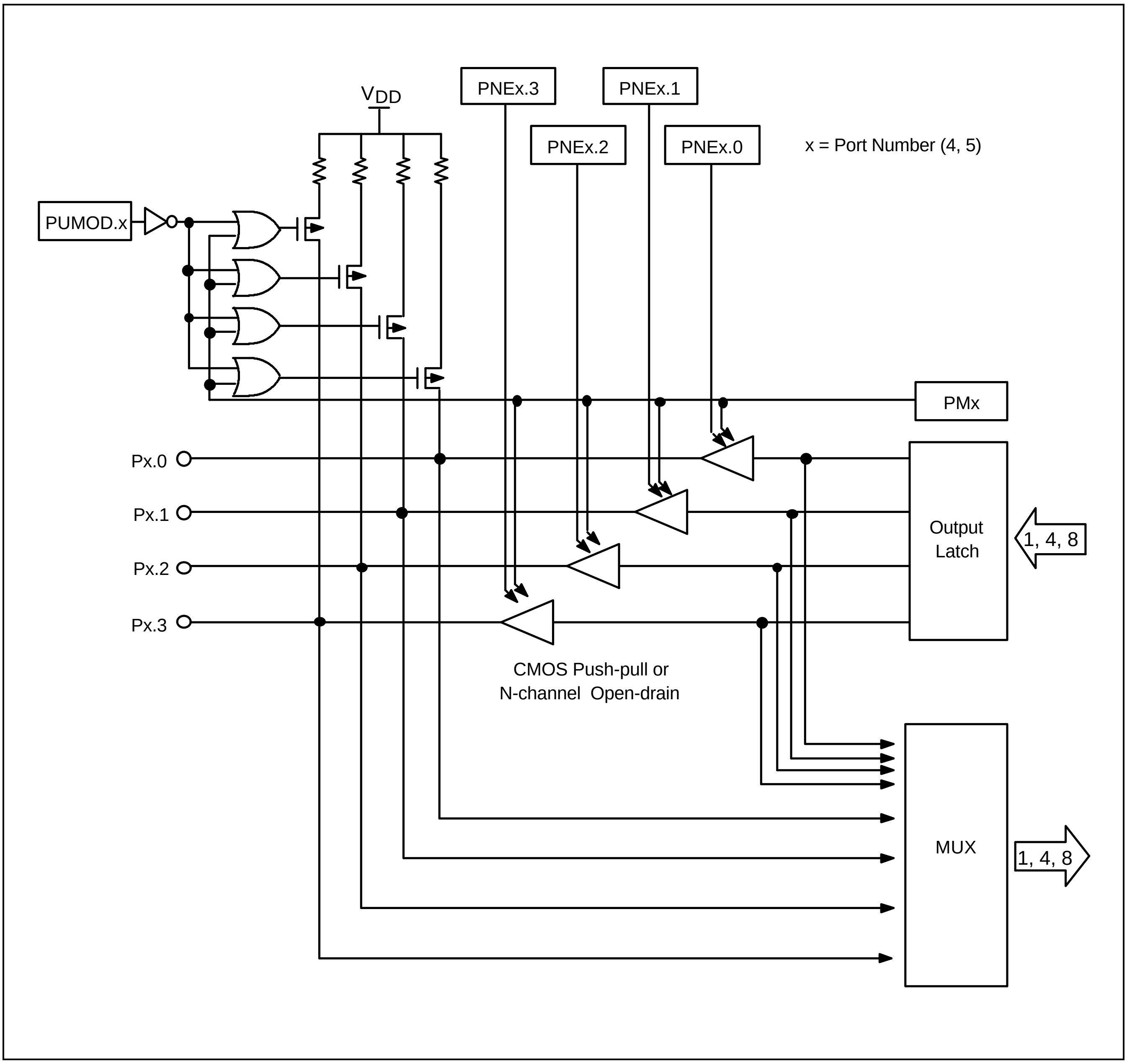


Figure 10-4. Ports 4, 5 Circuit Diagram

PORT 6 CIRCUIT DIAGRAM

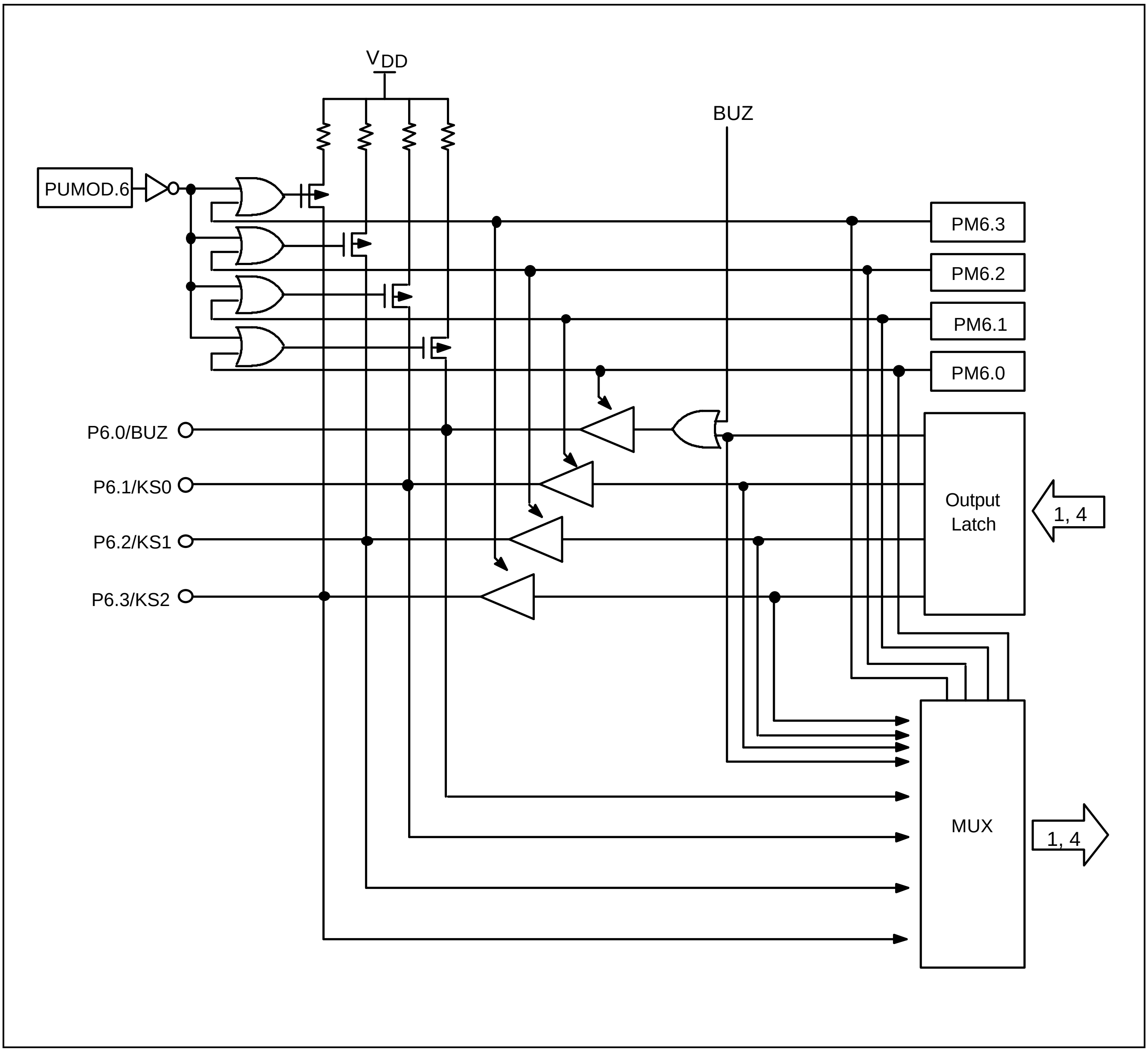


Figure 10-5. Port 6 Circuit Diagram

NOTES

11 TIMERS

OVERVIEW

The KS57C3204 microcontroller has three timer modules:

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

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

The 8-bit timer0 (TC0) is programmable timer that is used primarily for clock frequency modification.

The watch timer (WT) module consists of an 8-bit watch timer mode register, a clock selector, and a frequency divider circuit. Watch timer functions include real-time and watch-time measurement, main and subsystem clock interval timing, buzzer output generation. It also generates a clock signal for the LCD controller.

BASIC TIMER (BT)

OVERVIEW

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

- Clock selector logic
- 4-bit mode register (BMOD)
- 8-bit counter register (BCNT)
- 8-bit watchdog timer mode register (WDMOD)
- Watchdog timer counter clear flag (WDTCF)

The basic timer generates interrupt requests at precise intervals, based on the frequency of the system clock. Basic timer's counter register, BCNT, outputs timer pulses to the watchdog timer's counter register, WDTCNT when an overflow occurs in BCNT. 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 chip reset. Bit settings in the basic timer mode register BMOD turns the BT on and off, selects the input clock frequency, and controls interrupt or 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 chip reset is generated, the standard stabilization interval for system clock oscillation is 31.3 ms at 4.19 MHz.

Watchdog Timer Function

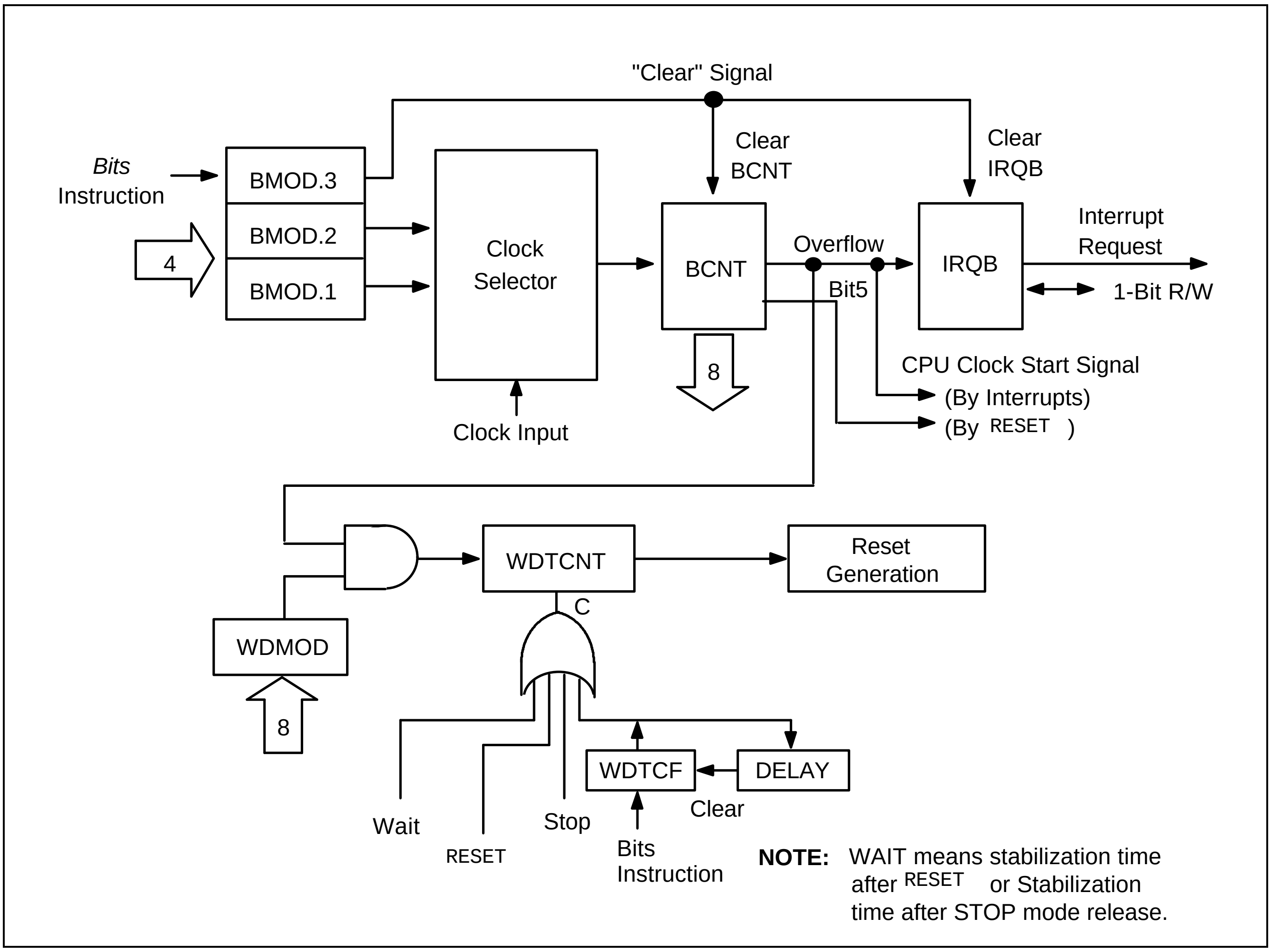
The basic timer can also be used as a "watchdog" timer to detect an inadvertent program loop, that is, system or program operation error. For this purpose, instruction that clears the watchdog timer (BITS WDTCF) within a given period should be executed at proper points in a program. If an instruction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

- Write some value (except #5AH) to Watchdog Timer Mode register, WDMOD.
- Each time BCNT overflows, an overflow signal is sent to the watchdog timer counter, WDCNT.
- If WDTCNT overflows, system reset will be generated.

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 is possible 1-bit write.	"0"
BCNT	Counter	Counts clock pulses matching the BMOD frequency setting	8-bit	F86H–F87H	8-bit read-only	"u" (note)
WDMOD	Control	Controls watchdog timer operation.	8-bit	F98H–F99H	8-bit write-only	A5H
WDTCF	Control	Clear the watchdog timer's counter.	1-bit	F9AH.3	1-bit write-only	"0"

NOTE: "u" means that the value is undetermined after a chip reset.



NOTE: WAIT means stabilization time after RESET or Stabilization time after STOP mode release.

Figure 11-1. Basic Timer Circuit Diagram

BASIC TIMER MODE REGISTER (BMOD)

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

— Restart the basic timer;

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

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

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

The most significant bit of the BMOD register, BMOD.3, is used to restart the basic timer. When BMOD.3 is set to logic one (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 and BMOD.1 — determines the clock input frequency and oscillation stabilization interval.

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

BMOD.3	Restart basic timer; clear IRQB, BCNT, and BMOD.3 to "0"
BMOD.0	Always zero

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

NOTES:
1. fxx = system clock frequency.
2. Oscillation stabilization time is the time required to stabilize clock signal oscillation after a chip reset or stop mode are released.
3. The standard stabilization time for main clock oscillation following a RESET signal is 31.3 ms at 4.19 MHz.

BASIC TIMER COUNTER (BCNT)

BCNT is an 8-bit counter for the basic timer. It can be addressed by 8-bit read instructions.

A chip reset leaves the BCNT counter value undetermined. BCNT is automatically cleared to logic zero whenever the BMOD register control bit (BMOD.3) is set to "1" to restart the basic timer. It is incremented each time a clock pulse of the frequency determined by the current BMOD bit settings is detected.

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

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

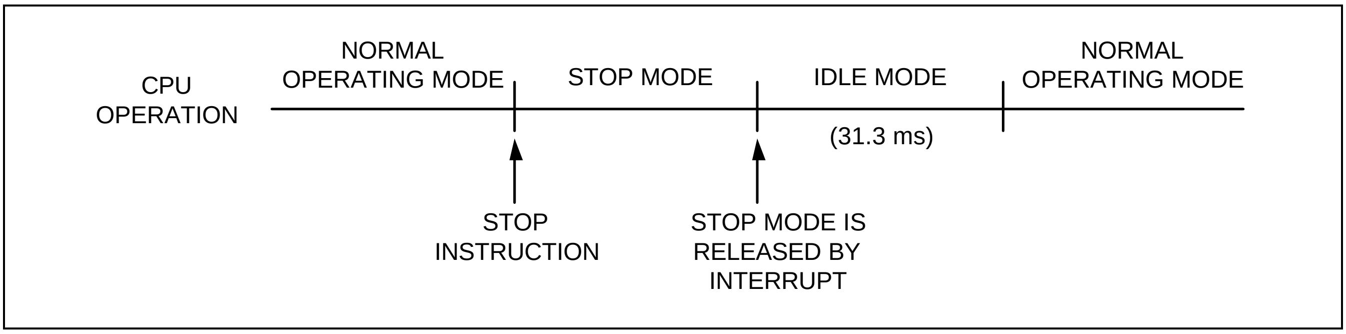
☞ PROGRAMMING TIP — Using the Basic Timer

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

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

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

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



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

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

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

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

WATCHDOG TIMER MODE REGISTER (WDMOD)

The watchdog timer mode register, WDMOD, is a 8-bit write-only register located at RAM address F98H–F99H. WDMOD register controls to enable or disable the watchdog function. WDMOD values are set to logic “A5H” following a chip reset and this value enables the watchdog timer, and watchdog timer is set to the longest interval because BT overflow signal is generated with the longest interval.

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

WATCHDOG TIMER COUNTER (WDCNT)

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

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

WATCHDOG TIMER COUNTER CLEAR FLAG (WDTCF)

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

Table 11-3. Watchdog Timer Interval Time

BMOD	BT Input Clock (frequency)	WDCNT Input Clock (frequency)	WDT Interval Time	Main Clock	Sub Clock
x000b	fxx/2^{12}	fxx/($2^{12} \times 2^8$)	$2^{12} \times 2^8 \times 2^3$/fxx	1.75–2 sec	224–256 sec
x010b	fxx/2^9	fxx/($2^9 \times 2^8$)	$2^9 \times 2^8 \times 2^3$/fxx	218.7–250 ms	28–32 sec
x100b	fxx/2^7	fxx/($2^7 \times 2^8$)	$2^7 \times 2^8 \times 2^3$/fxx	54.6–62.5 ms	7–8 sec
x110b	fxx/2^5	fxx/($2^5 \times 2^8$)	$2^5 \times 2^8 \times 2^3$/fxx	13.6–15.6 ms	1.75–2 sec

NOTES:

1. Assuming that fxx is main system clock, 4.19 MHz or subsystem clock, 32.768 KHz.
2. If the WDMOD changes such as disable and enable, you must set WDTCF flag to “1" for starting WDCNT from zero state.

☞ PROGRAMMING TIP — Using the Watchdog Timer

```
RESET   DI
        BITS    EMB
        SMB     15
        LD      EA,#00H
        LD      SP,EA
                •
                •
                •
        LD      A,#0CH          ;  WDCNT input clock is 7.82 ms
        LD      BMOD,A
                •
                •
                •
MAIN    BITS    WDTCF           ;  Main routine operation period must be shorter than
                                ;  watchdog
                •               ;  timer's period
                •
                •
        JP      MAIN
```

8-BIT TIMER0 (TC0)

OVERVIEW

Timer0 (TC0) is used to count system 'events' by identifying the transition (high-to-low or low-to-high) of incoming square wave signals. 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 and to select the basic clock frequency to be used for timer operations. To dynamically modify the basic frequency, new values can be loaded into the TMOD0 register during program execution.

TC0 FUNCTION SUMMARY

8-bit programmable timer Generates interrupts at specific time intervals based on the selected clock frequency.

TC0 COMPONENT SUMMARY

Mode register (TMOD0)
Activates the timer and selects the internal clock frequency.

Reference register (TREF0)
Stores the reference value for the desired number of clock pulses between interrupt requests.

Counter register (TCNT0)
Counts internal pulses based on the bit settings in TMOD0 and TREF0.

Clock selector circuit
Together with the mode register (TMOD0), lets you select one of three internal clock frequencies or an external clock.

8-bit comparator
Determines when to generate an interrupt by comparing the current value of the counter register (TCNT0) with the reference value previously programmed into the reference register (TREF0).

Interrupt request flag (IRQT0)
Cleared when TC0 operation starts and the TC0 interrupt service routine is executed and set to 1 whenever the counter value and reference value coincide.

Interrupt enable flag (IET0)
Must be set to logic one before the interrupt requests generated by timer 0 can be processed.

Table 11-4. TC0 Register Overview

Register Name	Type	Description	Size	RAM Address	Addressing Mode	Reset Value
TMOD0	Control	Controls TC0 enable/disable (bit 2); clears and resumes counting operation (bit 3); selects clock frequency (bits 5–4)	8-bit	F90H–F91H	8-bit write-only; (TMOD0.3 is also 1-bit writeable)	"0"
TCNT0	Counter	Counts clock pulses matching the TMOD0 frequency setting	8-bit	F94H–F95H	8-bit read-only	"0"
TREF0	Reference	Stores reference value for the timer0 interval setting	8-bit	F96H–F97H	8-bit write-only	FFH

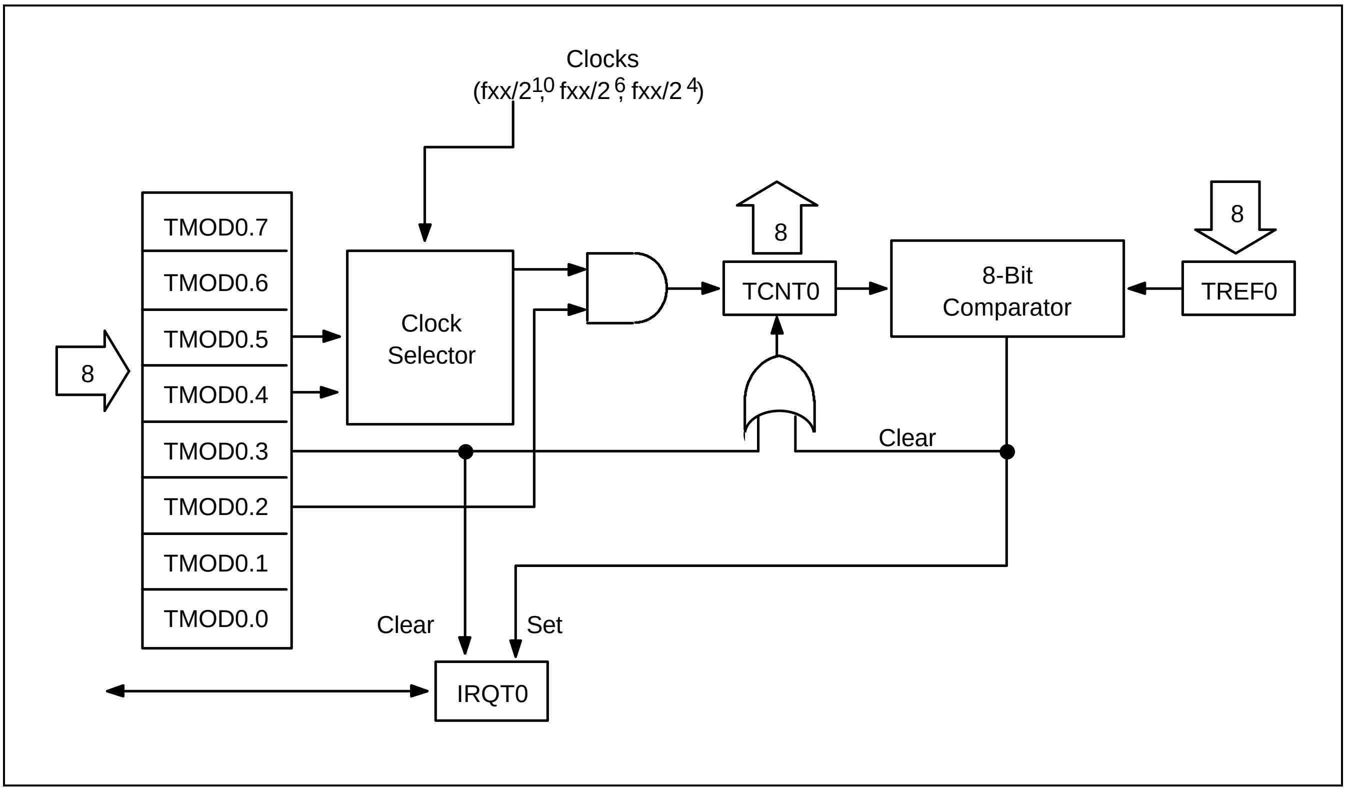


Figure 11-2. TC0 Circuit Diagram

TC0 ENABLE/DISABLE PROCEDURE

Enable Timer 0

- Set TMOD0.2 to logic one
- Set the TC0 interrupt enable flag IET0 to logic one
- Set TMOD0.3 to logic one

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

Disable Timer 0

- Set TMOD0.2 to logic zero

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

TC0 PROGRAMMABLE TIMER FUNCTION

Timer can be programmed to generate interrupt requests at various intervals based on the selected system clock frequency. Its 8-bit TC0 mode register TMOD0 is used to activate the timer and to select the clock frequency.

The reference register TREF0 stores the value for the number of clock pulses to be generated between interrupt requests. The counter register, TCNT0, counts the incoming clock pulses, which are compared to the TREF0 value as TCNT0 is incremented. When there is a match (TREF0 = TCNT0), an interrupt request is generated.

To program timer to generate interrupt requests at specific intervals, choose one of four internal clock frequencies (divisions of the system clock, fxx) and load a counter reference value into the TREF0 register. TCNT0 is incremented each time an internal counter pulse is detected with the reference clock frequency specified by TMOD0.4–TMOD0.5 settings.

To generate an interrupt request, the TC0 interrupt request flag (IRQT0) is set to logic one and the interrupt is generated. The content of TCNT0 is then cleared to 00H and TC0 continues counting. The interrupt request mechanism for TC0 includes an interrupt enable flag (IET0) and an interrupt request flag (IRQT0).

TC0 OPERATION SEQUENCE

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

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

TC0 MODE REGISTER (TMOD0)

TMOD0 is the 8-bit mode control register for timer0. It is addressable by 8-bit write instructions. One bit, TMOD0.3, is also 1-bit writeable. A chip reset clears all TMOD0 bits to logic zero and disables TC0 operations.

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

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

The TMOD0.5 and TMOD0.4 bit settings are used together to select the TC0 clock source.

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

Table 11-6. Timer 0 Mode Register Organization

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

Table 11-7. TMOD0.5, and TMOD0.4 Bit Settings

TMOD0.5	TMOD0.4	Resulting Counter Source and Clock Frequency
0	0	fxx/2^{10} (4.09 kHz)
0	1	fxx /2^{6} (65.5 kHz)
1	0	fxx/2^{4} (262 kHz)
1	1	Not used in KS57C3204

NOTE: Assuming that fxx is a main system clock, 4.19 MHz.

☞ PROGRAMMING TIP — Restarting TC0 Counting Operation

1. Set TC0 timer interval to 4.09 kHz:

```
BITS      EMB
SMB       15
LD        EA,#0CH
LD        TMOD0,EA
EI
BITS      IET0
```

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

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

TC0 COUNTER REGISTER (TCNT0)

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

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

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

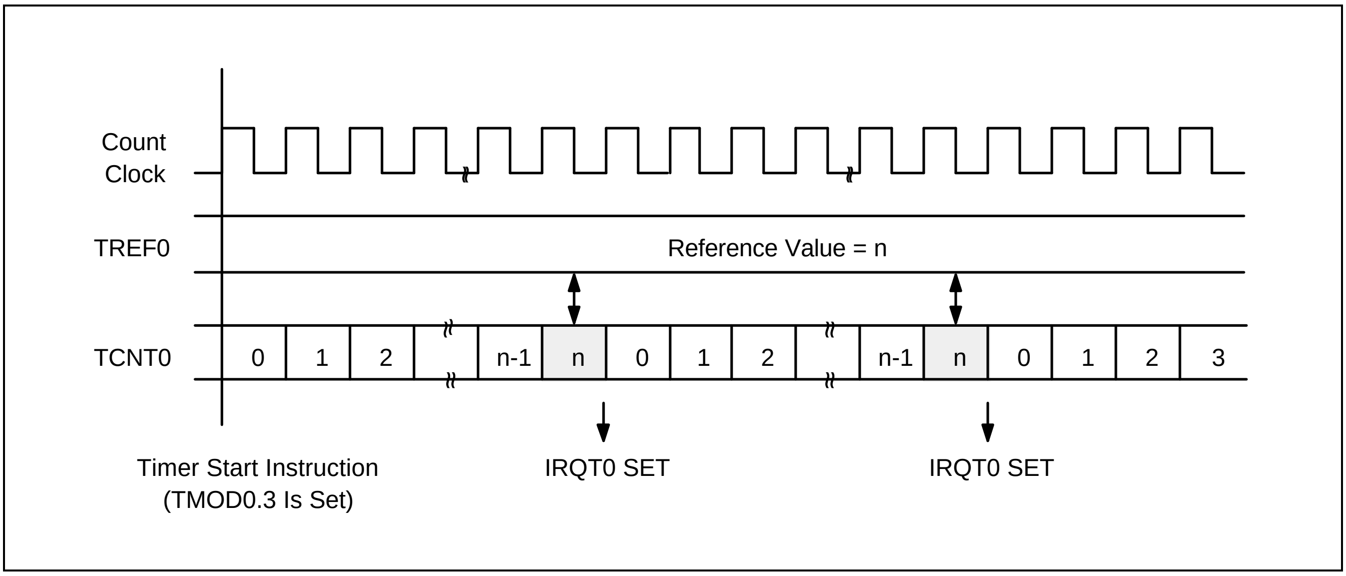


Figure 11-3. TC0 Timing Diagram

TC0 REFERENCE REGISTER (TREF0)

The TC0 reference register TREF0 is an 8-bit write-only register. It is addressable by 8-bit RAM control instructions. A chip 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.

During timer operation, the value loaded into the reference register is compared to the TCNT0 value. When TCNT0 = TREF0, an interrupt request is generated to signal the interval. 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:

$$\boxed{\begin{array}{c}\text{TC0 timer interval} = (\text{TREF0 value} + 1) \times \dfrac{1}{\text{TMOD0 frequency setting}} \\[1em] (\text{TREF0 value} \neq 0)\end{array}}$$

☞ PROGRAMMING TIP — Setting a TC0 Timer Interval

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

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

2. Calculate the TREF0 value:

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

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

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

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

```
BITS      EMB
SMB       15
LD        EA,#79H
LD        TREF0,EA
LD        EA,#0CH
LD        TMOD0,EA
```

WATCH TIMER

OVERVIEW

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

— 8-bit watch timer mode register (WMOD)

— Clock selector

— Frequency divider circuit

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

Real-Time and Watch-Time Measurement

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

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

Using a System or Subsystem Clock Source

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

$$\text{Watch timer clock (fw)} = \frac{\text{System clock (fxx)}}{128} = 32.768 \text{ kHz (fxx} = 4.19 \text{ MHz)}$$

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

Clock Source Generation for LCD Controller

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

Buzzer Output Frequency Generator

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

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

— The port 6.0 output mode flag (PM6.0) set to 'output' mode

— The output latch for I/O port 6.0 is cleared to "0"

Timing Tests in High-Speed Mode

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

Check Subsystem Clock Level Feature

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

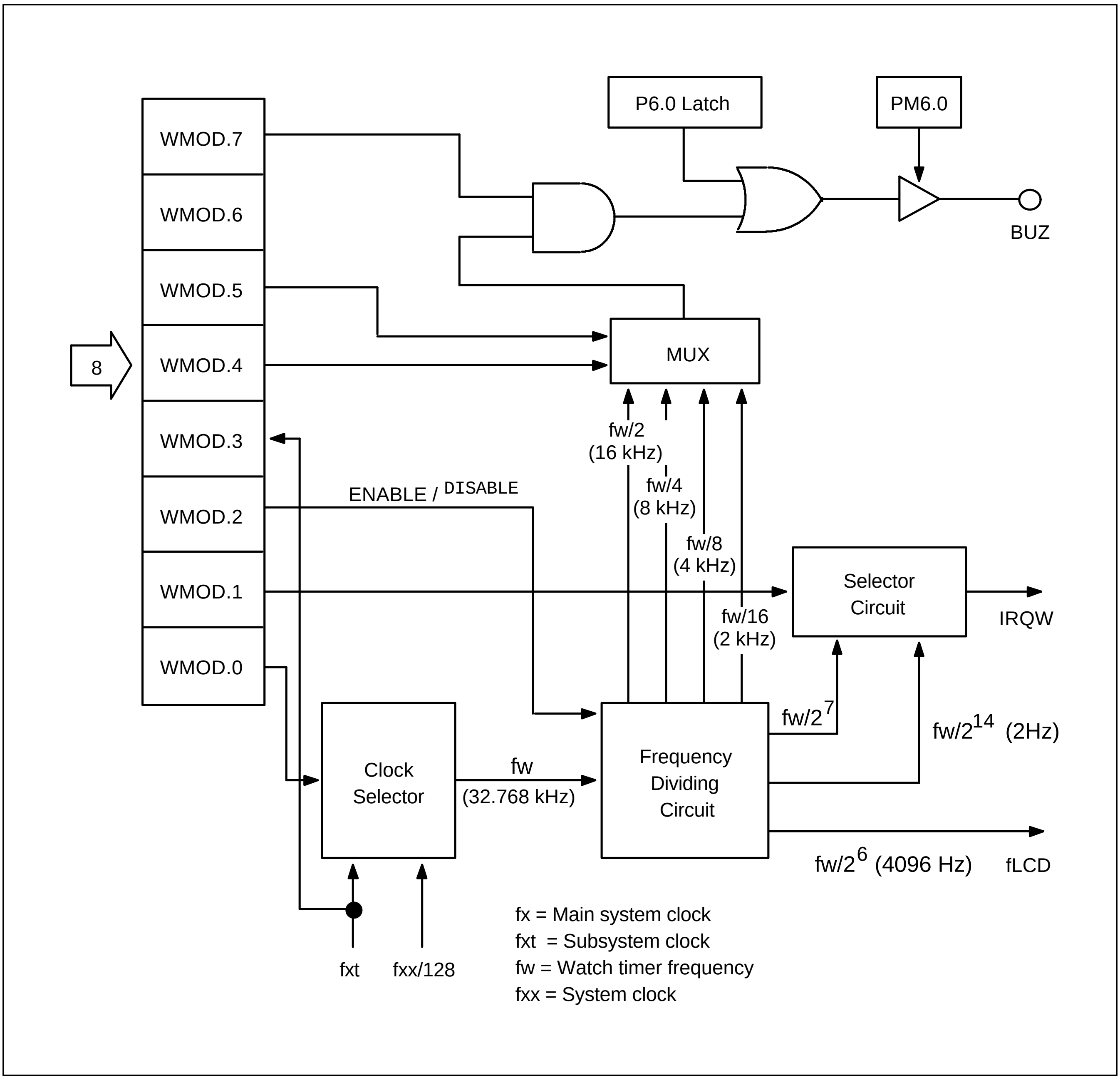


Figure 11-4. Watch Timer Circuit Diagram

WATCH TIMER MODE REGISTER (WMOD)

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

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

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

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

Table 11-8. 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		Input level to XT_{in} pin is low	F88H
	1		Input level to XT_{in} pin is high	
WMOD.2	0		Disable watch timer; clear frequency dividing circuits	
	1		Enable watch timer	
WMOD.1	0		Normal mode; sets IRQW to 0.5 seconds	
	1		High-speed mode; sets IRQW to 3.91 ms	
WMOD.0	0		Select fxx/128 as the watch timer clock (fw)	
	1		Select subsystem clock (fxt) as watch timer clock (fw)	

NOTE: System clock frequency (fxx) is assumed to be 4.19 MHz; subsystem clock (fxt) is assumed to be 32.768 kHz.

☞ PROGRAMMING TIP — Using the Watch Timer

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

```
                BITS      EMB
                SMB       15
                LD        EA,#10H
                LD        PMG1,EA        ;  P6.0 ← output mode
                BITR      P6.0
                LD        EA,#85H
                LD        WMOD,EA
                BITS      IEW
```

2. Sample real-time clock processing method:

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

NOTES

12 LCD CONTROLLER/DRIVER

OVERVIEW

The KS57C3204 microcontroller can directly drive an up to 28 SEG x 4 COM LCD panel. Its LCD block has the following components:

- LCD controller/driver
- Display RAM for storing display data
- 28 segment output pins (SEG0–SEG27)
- 4 common output pins (COM0–COM3)
- Internal resistor circuit for LCD bias

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

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

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

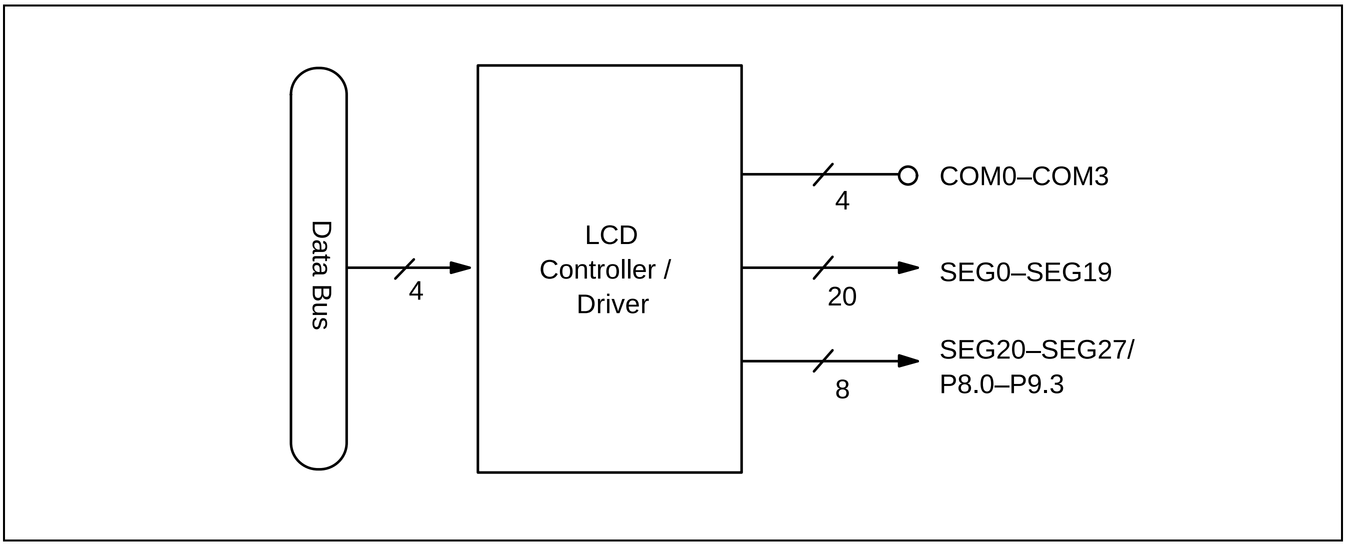


Figure 12-1. LCD Function Diagram

LCD CIRCUIT DIAGRAM

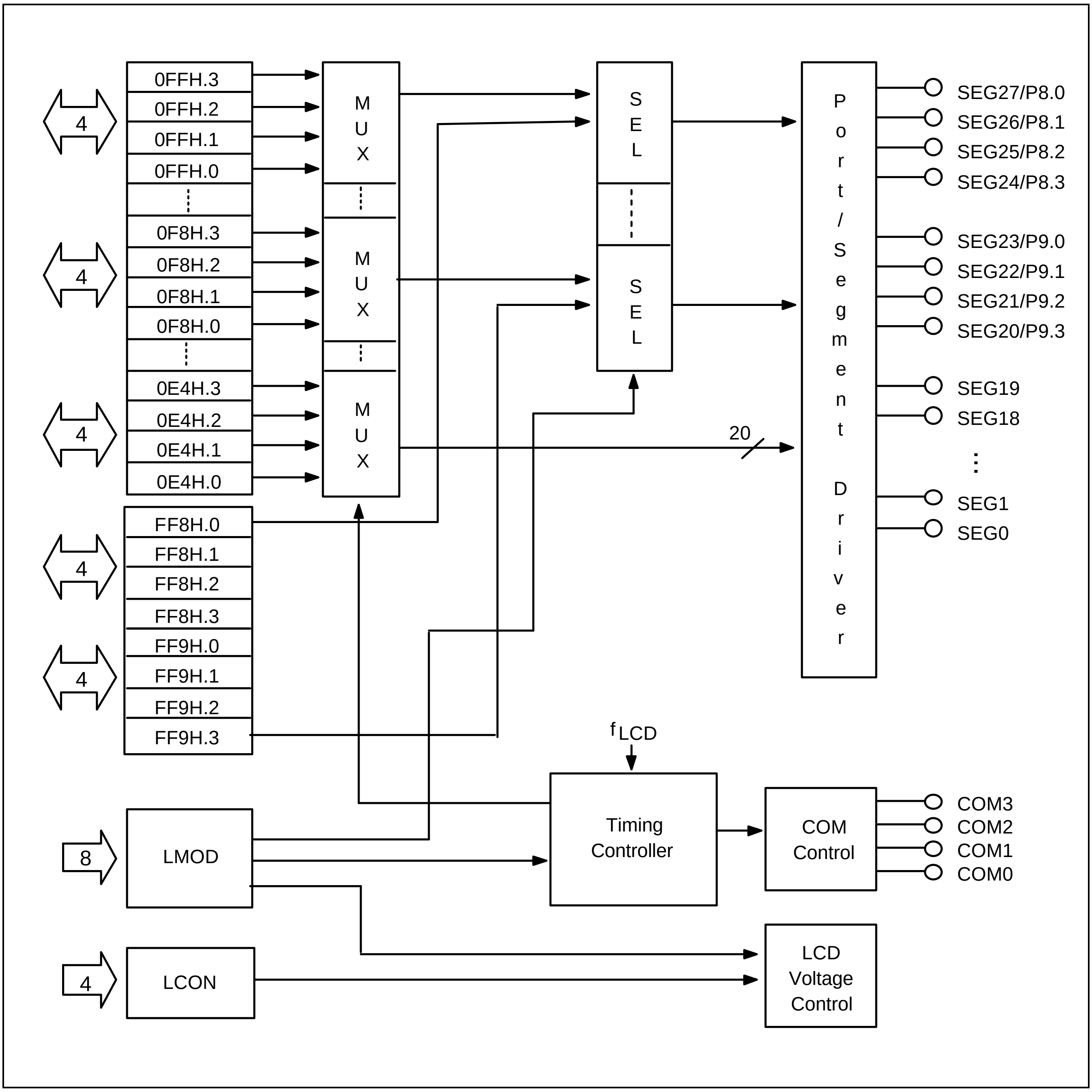


Figure 12-2. LCD Circuit Diagram

LCD RAM ADDRESS AREA

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

Display RAM data are sent out through segment pins SEG0–SEG27 using a direct memory access (DMA) method that is synchronized with the f_{LCD} signal. RAM addresses in this location that are not used for LCD display can be allocated to general-purpose use.

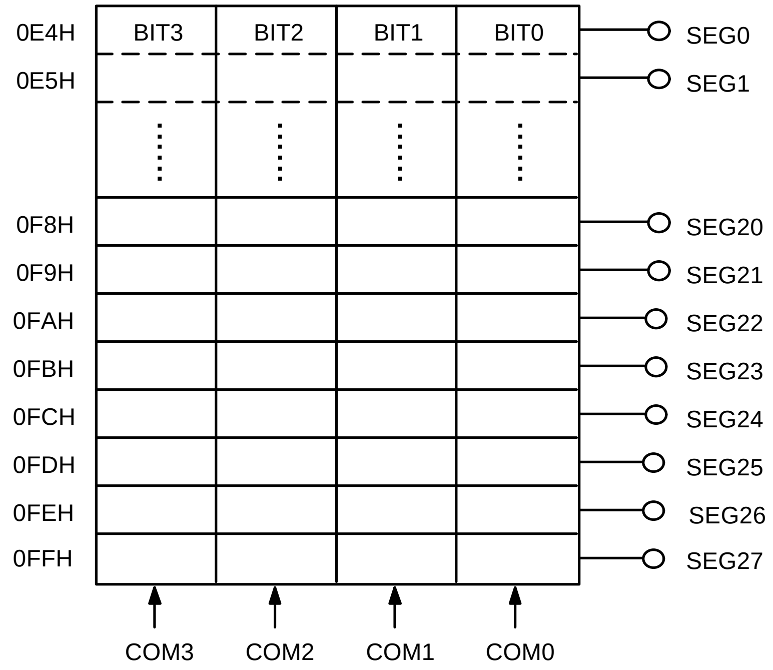


Figure 12-3. LCD Display Data RAM Organization

Table 12-1. Common Signal Pins Used per Duty Cycle

Display Mode	COM0 Pin	COM1 Pin	COM2 Pin	COM3 Pin
Static duty	Selected	N/C	N/C	N/C
1/2 duty	Selected	Selected	N/C	N/C
1/3 duty	Selected	Selected	Selected	N/C
1/4 duty	Selected	Selected	Selected	Selected

NOTE: NC = no connection is required.

LCD CONTROL REGISTER (LCON)

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

F8EH	“0”	LCON.2	LCON.1	LCON.0	LCON

The effect of the LCON.0 setting is dependent upon the current setting of bits LMOD.3.

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

LCON Bit	Setting	Description
LCON.3	0	Always logic zero.
LCON.2	0	V_{LC1} is connected to V_{LC2} through a dividing resistor; activate dividing resistor
	1	V_{LC1} is shorted to V_{LC2}; inactivate dividing resistor
LCON.1	0	Bias is connected to V_{LC0} through a dividing resistor; activate dividing resistor
	1	Bias is shorted to V_{LC0}; inactivate dividing resistor
LCON.0	0	LCD output low, display off; cut off current to dividing resistor
	1	If LMOD.3 = “0”: LCD display off. If LMOD.3 = “1”: COM and SEG output in display mode; LCD display on.

Table 12-3. LCON.0 and LMOD.3 Bit Settings

LCON.0	LMOD.3	COM0–COM2	SEG0–SEG27	Results
0	x	Output low; LCD display off	Output low; LCD display off	LCD display off. Cut off current to dividing resistors
1	0	LCD display off	LCD display off	LCD display off
	1	COM output corresponds to display mode	SEG output corresponds to display mode	LCD display on

LCD MODE REGISTER (LMOD)

The LCD mode control register LMOD is used to control display mode; LCD clock, segment or port output, and display on/off. LMOD can be manipulated using 8-bit write instructions, bit 3 (LMOD.3) can be also written by 1-bit instructions.

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

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

The LCD display can continue to operate during idle and stop modes if a subsystem clock is used as the watch timer source. The LCD mode register, LMOD, controls the output mode of the 8 pins used for normal outputs (P8.0–P9.3). Bits LMOD.7–.6 define the segment output and normal output configuration.

Table 12-4. LCD Mode Register (LMOD) Organization

LMOD.7	LMOD.6	LCD Output Segments and 1-Bit Output Pins
0	0	Segments 27–24 and 23–20
0	1	Segments 27–24 and normal output at P9.3–P9.0
1	0	Normal output at P8.3–P8.0 and segments 23–20
1	1	Normal output only at P9.3–P9.0 and P8.3–P8.0

LMOD.5	LMOD.4	LCD Clock (LCDCK) Frequency
0	0	fw/2^9 = 64 Hz
0	1	fw/2^8 = 128 Hz
1	0	fw/2^7 = 256 Hz
1	1	fw/2^6 = 512 Hz

LMOD.3	LMOD.2	LMOD.1	LMOD.0	Duty and Bias Selection for LCD Display
0	x	x	x	LCD Display off
1	0	0	0	1/4 duty, 1/3 bias (3)
1	0	0	1	1/3 duty, 1/3 bias (3)
1	0	1	0	1/2 duty, 1/2 bias (3)
1	0	1	1	1/3 duty, 1/2 bias (3)
1	1	0	0	Static

NOTES:
1. 'x' means don't care.
2. fw = 32.768 kHz, watch timer clock.
3. Bias can be configured as setting the LCON register.

Table 12-5. LCD Clock Signal (LCDCK) and Frame Frequency

LCDCK Frequency	Static	1/2 Duty	1/3 Duty	1/4 Duty
fw/2^9 = 64 Hz	64	32	21	16
fw/2^8 = 128 Hz	128	64	43	32
fw/2^7 = 256 Hz	256	128	85	64
fw/2^6 = 512 Hz	512	256	171	128

NOTES: fw = 32.768 kHz

LCD DRIVE VOLTAGE

LCD Power Supply	Static Mode	1/2 Bias	1/3 Bias
V_{LC0}	V_{LCD}	V_{LCD}	V_{LCD}
V_{LC1}	2/3 V_{LCD}	1/2 V_{LCD}	2/3 V_{LCD}
V_{LC2}	1/3 V_{LCD}	1/2 V_{LCD}	1/3 V_{LCD}
V_{LC3}	0 V	0 V	0 V

NOTE: The LCD panel display may deteriorate if a DC voltage is applied that lies between the common and segment signal voltage. Therefore, always drive the LCD panel with AC voltage.

LCD VOLTAGE DIVIDING RESISTORS

The voltage dividing resistors for the LCD circuit are configured on-chip. Figure 12-4 shows the standard voltage dividing resistor circuits. Using these internal voltage dividing resistors, LCON.0, and LCON.1, you can drive 2.5 V, 3 V, or 5 V LCD display. To cut off the current supply to the voltage dividing resistors, clear LCON.0 when you turn the LCD display off.

COMMON (COM) SIGNALS

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

— In 1/2 duty mode, COM0–COM1 pins are selected

— In 1/3 duty mode, COM0–COM2 pins are selected

— In 1/4 duty mode, COM0–COM3 pins are selected

SEGMENT (SEG) SIGNALS

The 28 LCD segment signal pins are connected to corresponding display RAM locations at bank 0. Bits of the display RAM are synchronized with the common signal output pins.

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

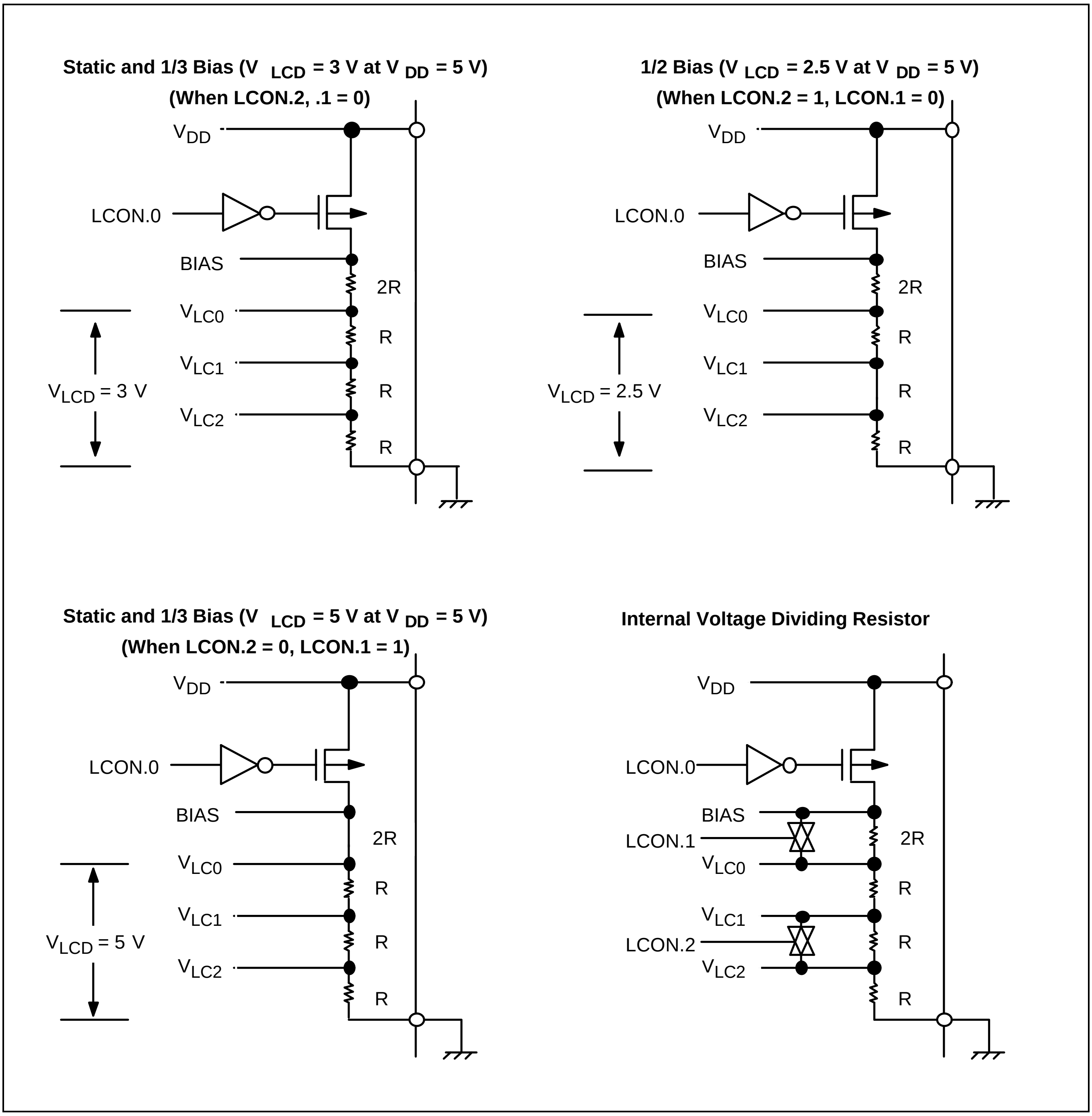

Static and 1/3 Bias (V LCD = 3 V at V DD = 5 V)
(When LCON.2, .1 = 0)
VDD
LCON.0
BIAS
2R
VLC0
R
VLC1
R
VLC2
R
VLCD = 3 V
1/2 Bias (V LCD = 2.5 V at V DD = 5 V)
(When LCON.2 = 1, LCON.1 = 0)
VDD
LCON.0
BIAS
2R
VLC0
R
VLC1
R
VLC2
R
VLCD = 2.5 V
Static and 1/3 Bias (V LCD = 5 V at V DD = 5 V)
(When LCON.2 = 0, LCON.1 = 1)
VDD
LCON.0
BIAS
2R
VLC0
R
VLC1
R
VLC2
R
VLCD = 5 V
Internal Voltage Dividing Resistor
VDD
LCON.0
BIAS
LCON.1
2R
VLC0
R
VLC1
LCON.2
R
VLC2
R

Figure 12-4. Voltage Dividing Resistor Circuit Diagrams

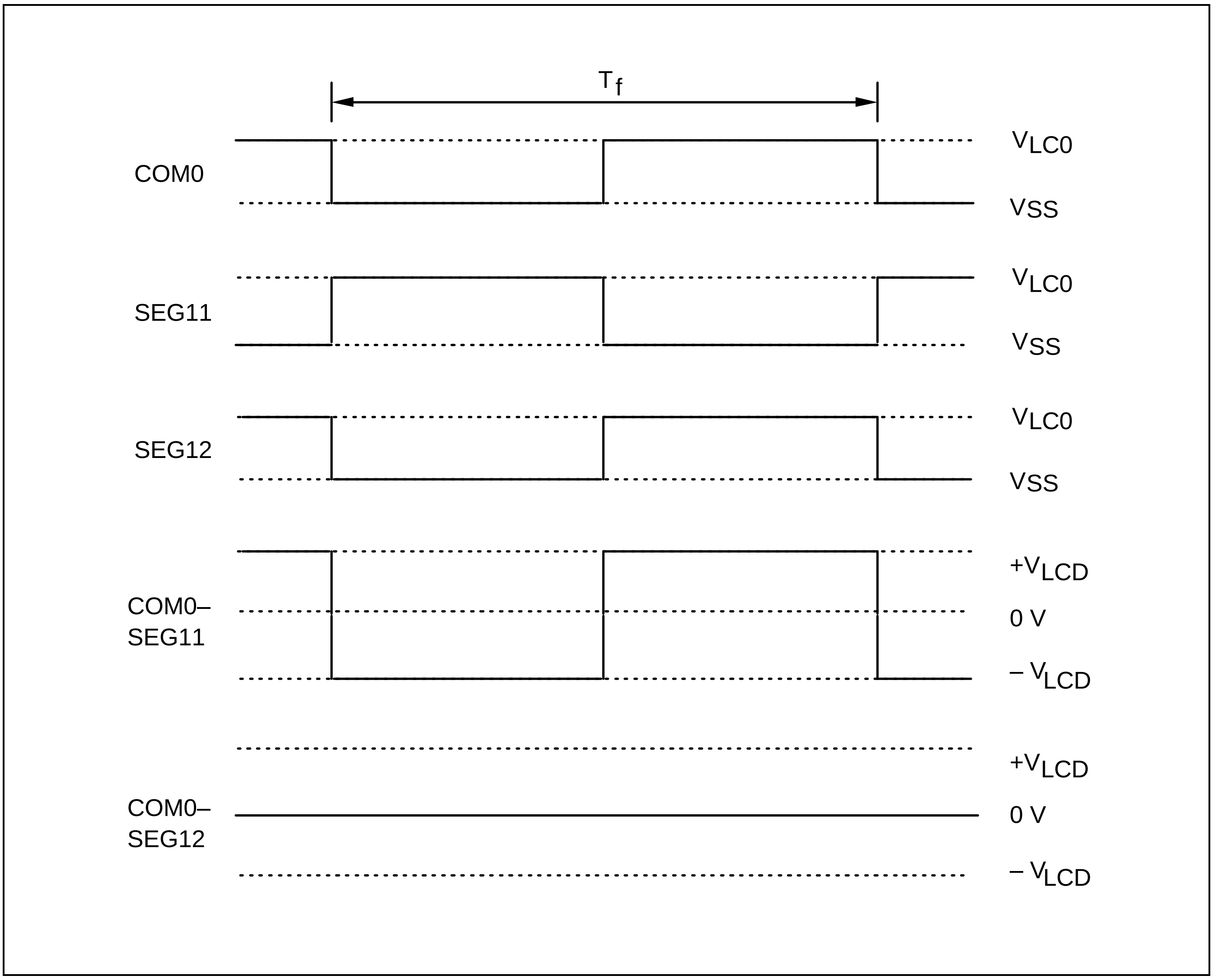
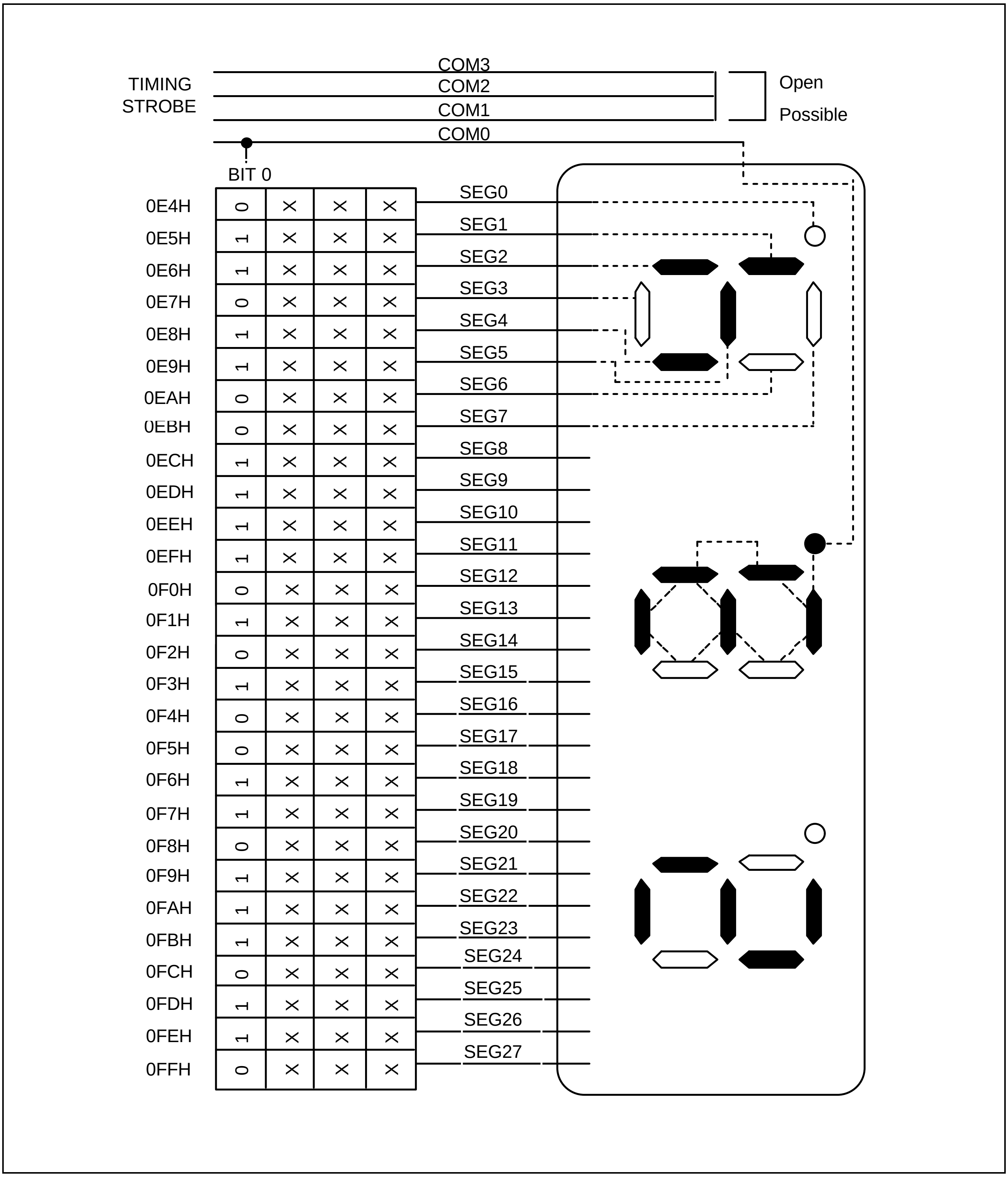


Figure 12-5. LCD Signal Waveforms in Static Mode



	BIT 0				
0E4H	0	x	x	x	SEG0
0E5H	1	x	x	x	SEG1
0E6H	1	x	x	x	SEG2
0E7H	0	x	x	x	SEG3
0E8H	1	x	x	x	SEG4
0E9H	1	x	x	x	SEG5
0EAH	0	x	x	x	SEG6
0EBH	0	x	x	x	SEG7
0ECH	1	x	x	x	SEG8
0EDH	1	x	x	x	SEG9
0EEH	1	x	x	x	SEG10
0EFH	1	x	x	x	SEG11
0F0H	0	x	x	x	SEG12
0F1H	1	x	x	x	SEG13
0F2H	0	x	x	x	SEG14
0F3H	1	x	x	x	SEG15
0F4H	0	x	x	x	SEG16
0F5H	0	x	x	x	SEG17
0F6H	1	x	x	x	SEG18
0F7H	1	x	x	x	SEG19
0F8H	0	x	x	x	SEG20
0F9H	1	x	x	x	SEG21
0FAH	1	x	x	x	SEG22
0FBH	1	x	x	x	SEG23
0FCH	0	x	x	x	SEG24
0FDH	1	x	x	x	SEG25
0FEH	1	x	x	x	SEG26
0FFH	0	x	x	x	SEG27

Figure 12-6. LCD Connection Example in Static Mode

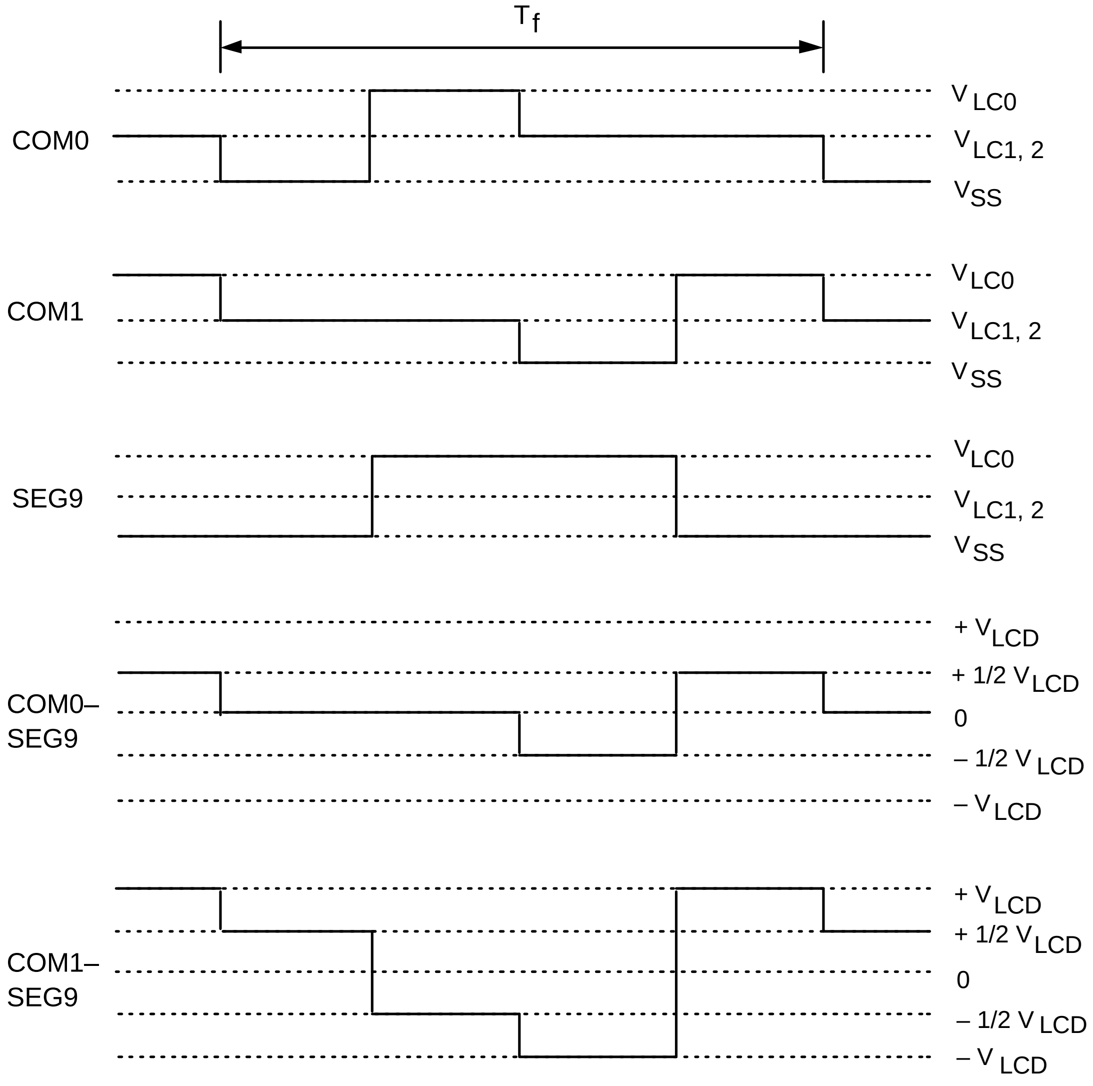
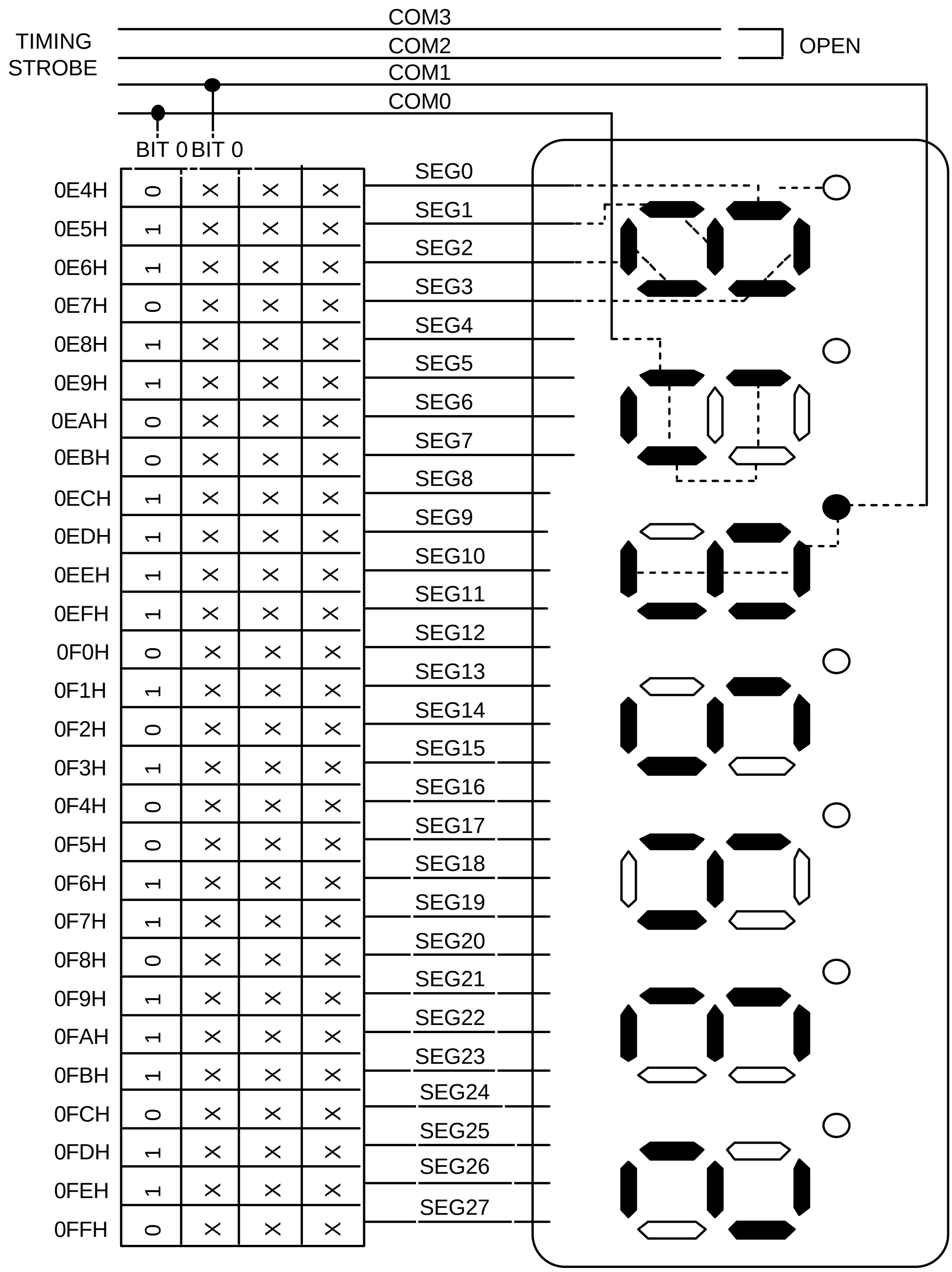


Figure 12-7. LCD Signal Waveforms at 1/2 Duty, 1/2 Bias



0E4H	0	×	×	×
0E5H	1	×	×	×
0E6H	1	×	×	×
0E7H	0	×	×	×
0E8H	1	×	×	×
0E9H	1	×	×	×
0EAH	0	×	×	×
0EBH	0	×	×	×
0ECH	1	×	×	×
0EDH	1	×	×	×
0EEH	1	×	×	×
0EFH	1	×	×	×
0F0H	0	×	×	×
0F1H	1	×	×	×
0F2H	0	×	×	×
0F3H	1	×	×	×
0F4H	0	×	×	×
0F5H	0	×	×	×
0F6H	1	×	×	×
0F7H	1	×	×	×
0F8H	0	×	×	×
0F9H	1	×	×	×
0FAH	1	×	×	×
0FBH	1	×	×	×
0FCH	0	×	×	×
0FDH	1	×	×	×
0FEH	1	×	×	×
0FFH	0	×	×	×

Figure 12-8. LCD Connection Example at 1/2 Duty, 1/2 Bias

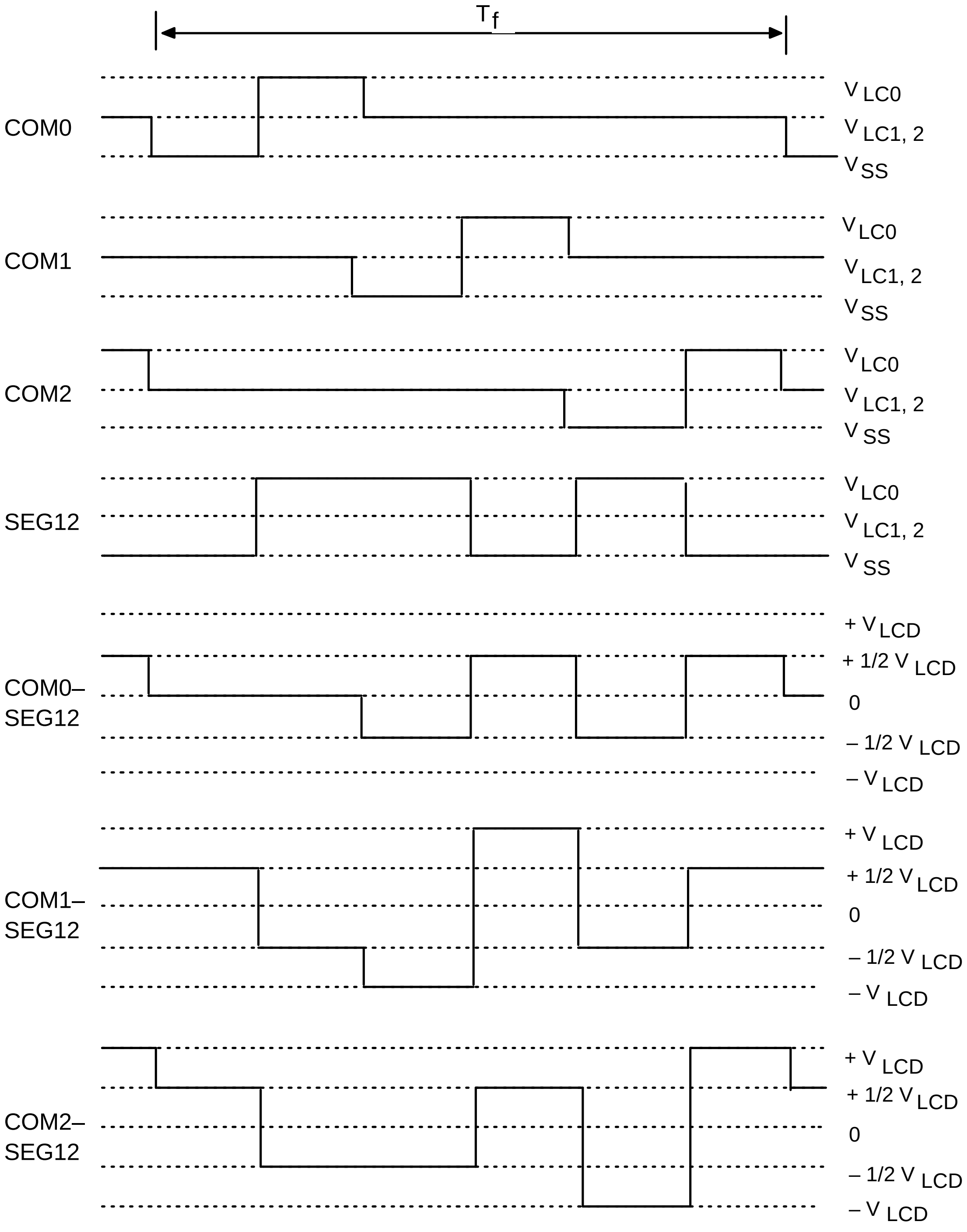


Figure 12-9. LCD Signal Waveforms at 1/3 Duty, 1/2 Bias

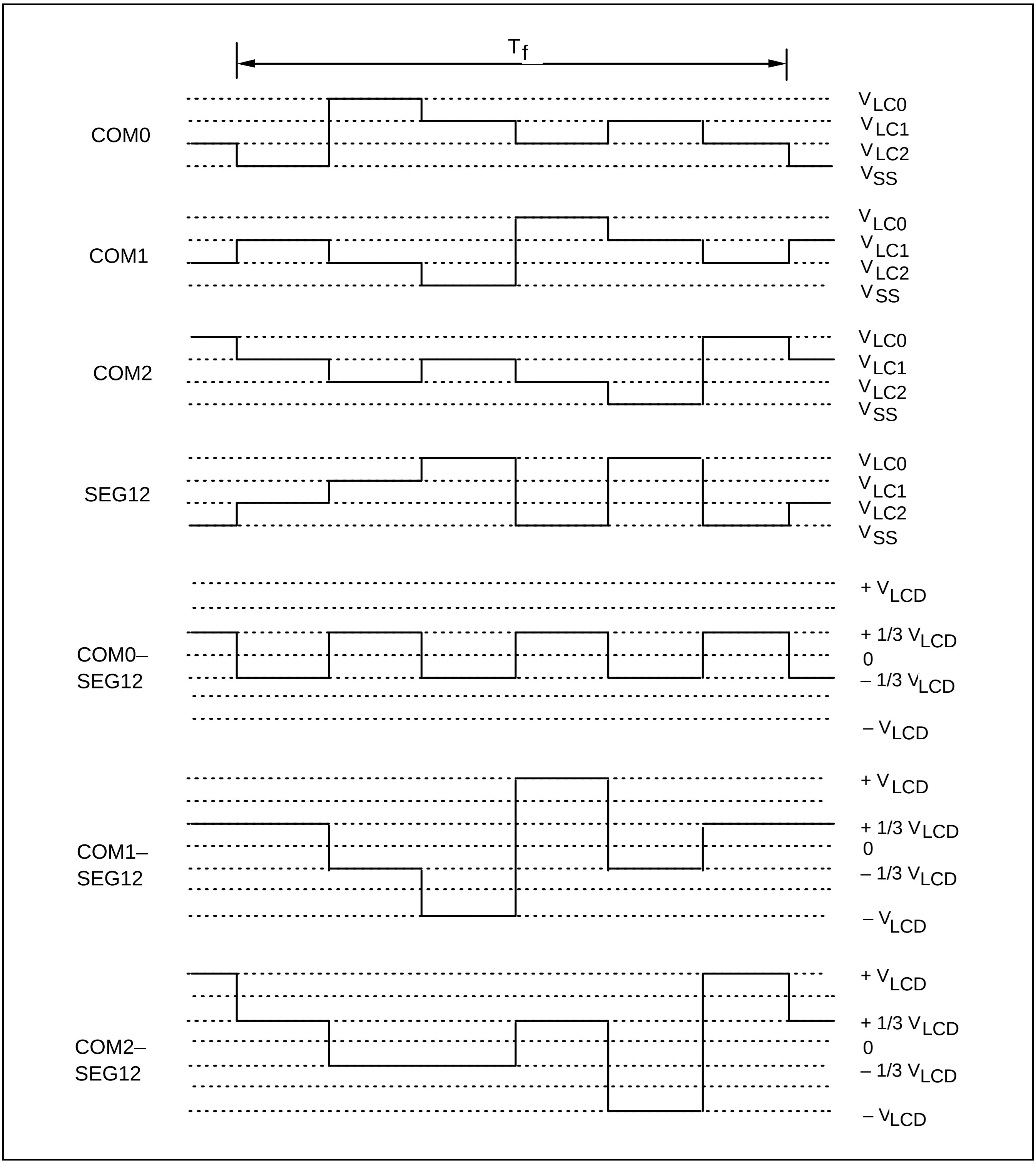
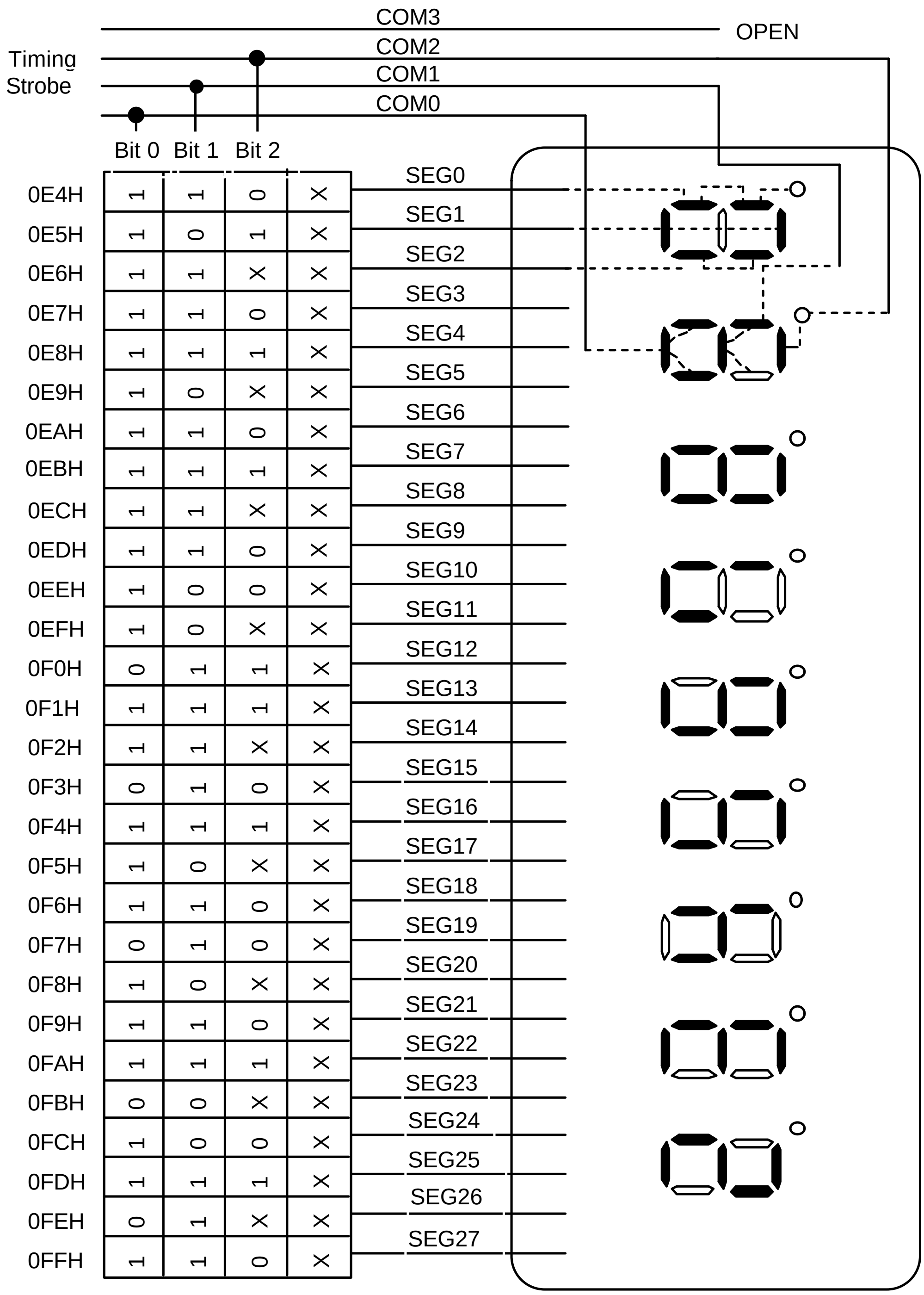


Figure 12-10. LCD Signal Waveforms at 1/3 Duty, 1/3 Bias



	Bit 0	Bit 1	Bit 2	
0E4H	1	1	0	×
0E5H	1	0	1	×
0E6H	1	1	×	×
0E7H	1	1	0	×
0E8H	1	1	1	×
0E9H	1	0	×	×
0EAH	1	1	0	×
0EBH	1	1	1	×
0ECH	1	1	×	×
0EDH	1	1	0	×
0EEH	1	0	0	×
0EFH	1	0	×	×
0F0H	0	1	1	×
0F1H	1	1	1	×
0F2H	1	1	×	×
0F3H	0	1	0	×
0F4H	1	1	1	×
0F5H	1	0	×	×
0F6H	1	1	0	×
0F7H	0	1	0	×
0F8H	1	0	×	×
0F9H	1	1	0	×
0FAH	1	1	1	×
0FBH	0	0	×	×
0FCH	1	0	0	×
0FDH	1	1	1	×
0FEH	0	1	×	×
0FFH	1	1	0	×

Figure 12-11. LCD Connection Example at 1/3 Duty, 1/3 Bias

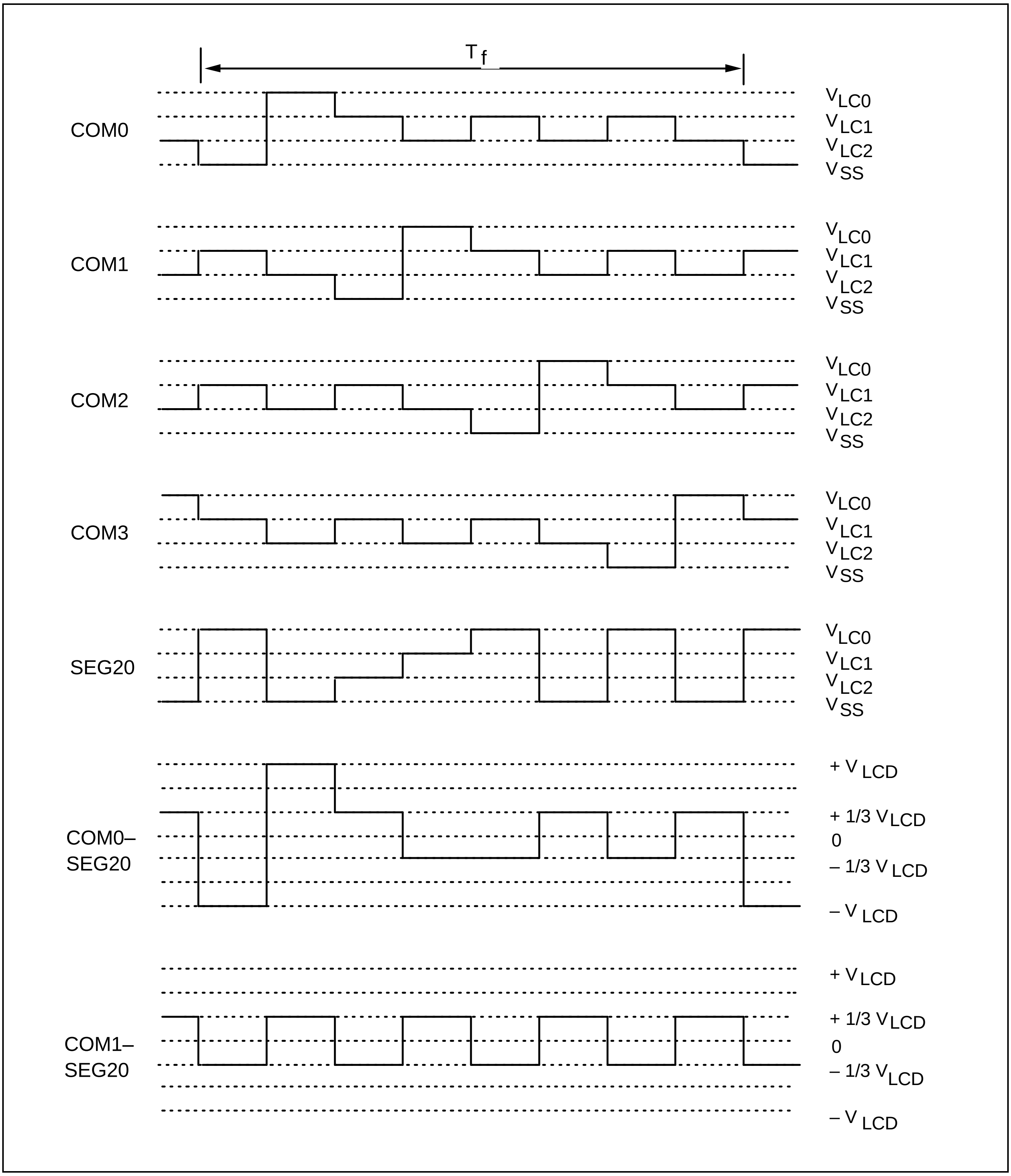
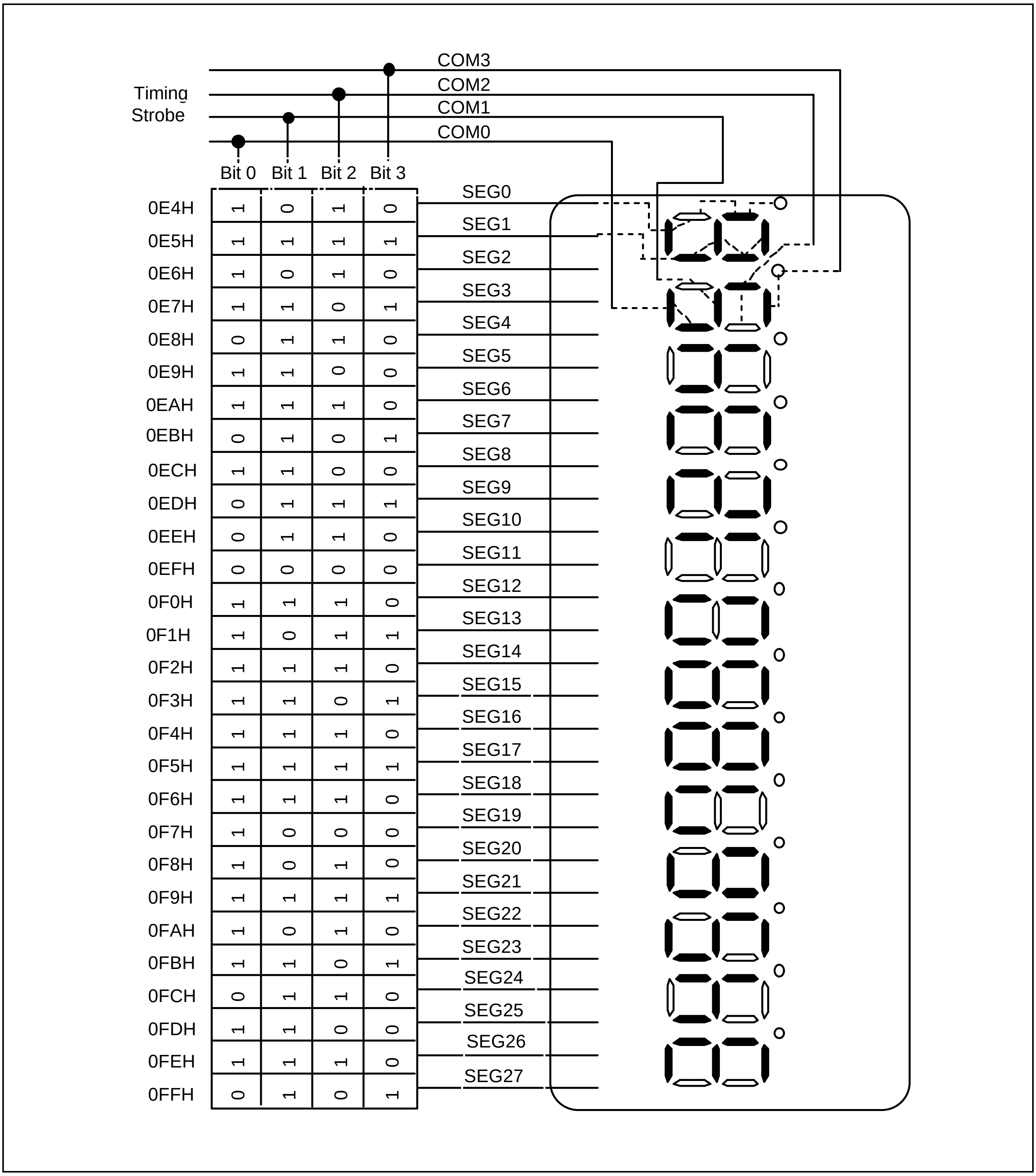


Figure 12-12. LCD Signal Waveforms at 1/4 Duty, 1/3 Bias



	Bit 0	Bit 1	Bit 2	Bit 3
0E4H	1	0	1	0
0E5H	1	1	1	1
0E6H	1	0	1	0
0E7H	1	1	0	1
0E8H	0	1	1	0
0E9H	1	1	0	0
0EAH	1	1	1	0
0EBH	0	1	0	1
0ECH	1	1	0	0
0EDH	0	1	1	1
0EEH	0	1	1	0
0EFH	0	0	0	0
0F0H	1	1	1	0
0F1H	1	0	1	1
0F2H	1	1	1	0
0F3H	1	1	0	1
0F4H	1	1	1	0
0F5H	1	1	1	1
0F6H	1	1	1	0
0F7H	1	0	0	0
0F8H	1	0	1	0
0F9H	1	1	1	1
0FAH	1	0	1	0
0FBH	1	1	0	1
0FCH	0	1	1	0
0FDH	1	1	0	0
0FEH	1	1	1	0
0FFH	0	1	0	1

Figure 12-13. LCD Connection Example at 1/4 Duty, 1/3 Bias

13 ANALOG-TO-DIGITAL CONVERTER

OVERVIEW

The 8-bit A/D converter (ADC) module uses a successive approximation logic to convert analog levels entering at one of the four input channels to equivalent 8-bit digital values. The analog input level must lie between the V_{DD} and the V_{SS} values. The A/D converter has the following components:

- Analog comparator with successive approximation logic
- D/A converter logic (resistor string type)
- ADC and port control register (APCON)
- ADC control register (AFLAG)
- ADC mode register (ADMOD)
- Four multiplexed analog data input pins (ADC0–ADC3)
- 8-bit A/D conversion data output register (ADATA)

To operate the A/D converter, P3 must be configured to ADC mode as using APCON register and one of the 4-channel is selected by writing the appropriate value to the A/D mode register, ADMOD, and the conversion start bit, AFLAG.3 must be set to ″1″. Conversion speed is determined by the system clock (f_x or f_{xt}).

When the A/D operation is complete, the EOC flag must be tested in order to verify that the conversion was successful. When the EOC value is "1", the converted digital values stored in the data register ADATA can be read.

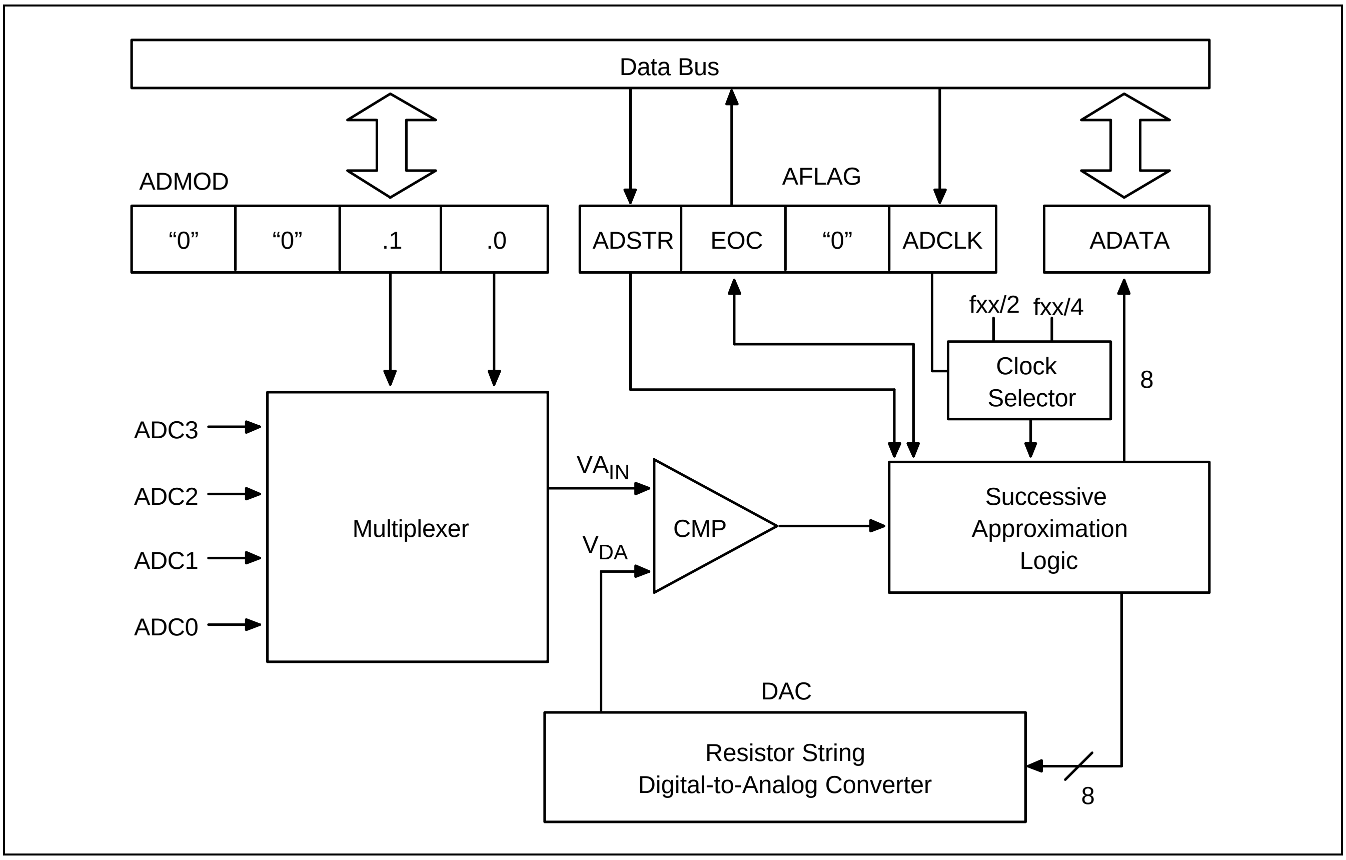


Figure 13-1. A/D Converter Circuit Diagram

Table 13-1. A/D Converter Component Overview

ADC Function	Mnemonic	Description
Digital-to-analog converter	DAC	Uses successive approximation logic to convert digital input into the reference analog voltage, V_{DA}. These V_{DA} values are input to the comparator and then compared to the multiplexed external analog source voltage, $VA_{IN.}$
Comparator	CMP	Compares the applied external analog input voltage, VA_{IN}, to the analog reference voltage (V_{DA}) that is generated by the DAC and writes the corresponding digital value to the ADATA register.
Digital data register	ADATA	Stores digital values as analog-to-digital conversion is completed.
ADC mode register	ADMOD	Used to select one of four analog channels as the input source for the analog data to be converted.
ADC control register	AFLAG	Contains the control flags used to start A/D converter operation and to monitor operational status.
Successive approximation logic	–	Control blocks in the A/D converter contain the successive approximation logic required to generate the analog reference voltage.

ADC DATA REGISTER (ADATA)

The A/D converter data register, ADATA, is an 8-bit register in which digital data values are stored as an A/D conversion operation is completed. Digital values stored in ADATA are retained until another conversion operation is initiated. ADATA is addressable by 8-bit read instructions only.

FD8H	Bit 3	Bit 2	Bit 1	Bit 0
FD9H	Bit 7	Bit 6	Bit 5	Bit 4

ADC MODE REGISTER (ADMOD)

The analog-to-digital converter mode register ADMOD is a 4-bit register that is used to select one of four analog channels as the analog data input source. ADMOD is addressable by 1-bit or 4-bit read or write instructions.

FDAH	"0"	"0"	ADMOD.1	ADMOD.0

Input channels ADC0–ADC3 (corresponding to input port , P3.0–P3.3) may be used either for analog input to the A/D converter, or as normal input ports. Since only one of the four pins can be selected at one time as external source of analog data, the three remaining input pins are always available for normal inputs.

Table 13-2. A/D Converter Mode Register Settings

0	**0**	**ADMOD.1**	**ADMOD.0**	**Effect of ADMOD Bit Setting**
		0	0	Select input channel AD0
		0	1	Select input channel AD1
		1	0	Select input channel AD2
		1	1	Select input channel AD3

ADC AND PORT CONTROL REGISTER (APCON)

FAEH	**.3**	**.2**	**.1**	**.0**	**Effect of Bit Settings**
	0	0	0	0	Set P3 to connect the normal input
	Other settings				Each bit corresponds with P3.0, P3.1, P3.2, and P3.3 respectively. If the specific bits are set to logic "1", the corresponding pins are connected to ADC block, but disconnected from the normal input and automatically the pull-up registers off.

NOTE: All bits are cleared to "0" after a chip reset.

ADC CONTROL REGISTER (AFLAG)

The A/D converter control register, AFLAG, is a 4-bit register that contains the control flags used to start the A/D converter and to monitor its operational status.

FDBH	ADSTR	EOC	"0"	ADCLK

A conversion is started by setting ADSTR in the AFLAG register. ADSTR is write-only and is 1-bit and 4-bit addressable. The EOC bit (End Of Conversion) is a flag that can be read to determine the current status of an A/D conversion operation. When a conversion is completed, this bit is set so that an A/D conversion result is ready to be read. EOC is cleared by ADSTR setting. While this flag is set, the ADC cannot start a new conversion. EOC is 1-bit or 4-bit read-only addressable.

Table 13-2. A/D Converter Control Flag Settings

ADSTR	EOC	0	ADCLK	Effect of AFLAG Bit Setting
1				Enable A/D converter (when the ADSTR bit is set to "1", the A/D converter starts operating and the ADSTR bit is cleared automatically)
	0			A/D conversion is not completed (the start of a new conversion is blocked)
	1			A/D conversion is completed.
			0	Select fxx/2 clock for conversion
			1	Select fxx/4 clock for conversion

DIGITAL-TO ANALOG CONVERTER (DAC) BLOCK

The 8-bit digital-to analog converter (DAC) generates analog voltage reference values for the comparator. The DAC is a 256-step resistor string type digital-to-analog converter that uses successive approximation logic to convert digital input into the reference analog voltage, VDA.

VDA values are input from the DAC to the comparator where they are compared to the multiplexed external analog source voltage, VA_{IN}. Since the DAC has 8-bit resolution, it generates the 256-step analog reference voltage as follows:

$$V_{DA} = V_{REF}\ (\frac{n}{256} \pm \frac{1}{512})(1/2 \text{ LSB compensation}),\ V_{REF} = V_{DD}$$

(n = 0–256, as determined by successive approximation logic)

CONVERSION TIMING

The A/D conversion process requires 8-clock to convert each bit. Therefore a total of 34 clocks are required to complete an 8-bit conversion. With a system clock frequency (fxx). 4MHz and setting ADCLK = 0, the conversion time can be calculated as follows:

Start 1 clock + (4 clock/bit × 8 bits) + EOC 1 clock = 34 clocks, 34 × 2/4 MHz = 17 µs

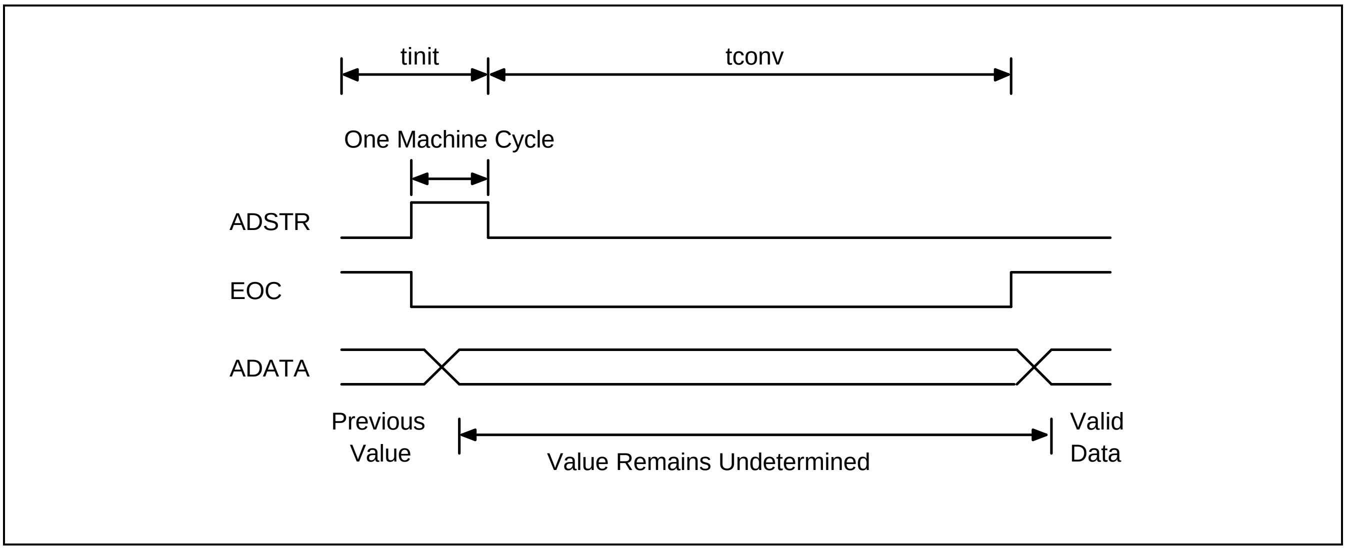


Figure 13-2. A/D Converter Timing Diagram

ADC PROCEDURE DESCRIPTION

Use these steps as a general guideline for writing A/D converter programs:

1. Select one of the conversion clocks, fxx/2 or fxx/4.
2. Configure port to ADC input mode as using APCON register.
3. Select one of the four analog channels, AD0–AD3, as the analog input source. To do this, write the appropriate value to the ADMOD register, bits ADMOD.1–ADMOD.0.

4 Start the A/D converter by setting the ADSTR flag of the AFLAG register to logic one.

5 When the converter starts, the EOC (End Of Conversion) flag in the AFLAG register is automatically set to logic one, and the ADSTR flag is cleared to logic zero.

6 The analog-to-digital conversion speed is determined by the oscillator frequency as follows:

$$\text{tconv} = 34 \times \text{conversion clock (fxx/2 or fxx/4)}$$

For example, with a 4.19 MHz oscillator clock and fxx/4, the t_{conv} value is 32.4 µs. The 'tinit' value is determined by the instruction type and the speed of the CPU clock.

7. When conversion has been completed, the EOC flag is set automatically so that a check can be made to verify that the conversion was successful.
8. Converted digital values that have been stored in the 8-bit ADATA register can now be read. Conversion values are retained until the next A/D conversion operation starts.

☞ PROGRAMMING TIP — Configuring A/D Converter Input Pins

In this A/D converter program sample, the ADC0, ADC1 and ADC2 pins are used as A/D input pins and the P3.3/ADC3 is used as normal input pin:

```
         BITR     EMB
         BITS     ADCLK              ;  Selects fxx/4 clock for conversion
         LD       A,#7H              ;  Setting ADC0, ADC1 and ADC2 as ADC input
         LD       APCON,A            ;  and P3.3 as normal input
         LD       A,#0H
         LD       ADMOD,A            ;  ADC0 pin select for A/D conversion
         BITS     ADSTR              ;  A/D conversion start
AD0CK    BTST     EOC                ;  A/D conversion end check
         JR       AD0CK              ;  A/D conversion not completed
         LD       EA,ADATA           ;  A/D conversion end
         LD       ADC0BUF,EA         ;  ADC0BUF ← ADC0 conversion data
         LD       A,#1H
         LD       ADMOD,A            ;  ADC1 pin select for A/D conversion
         BITS     ADSTR              ;  A/D conversion start
AD1CK    BTST     EOC                ;  A/D conversion end check
         JR       AD1CK              ;  A/D conversion not completed
         LD       EA,ADATA           ;  A/D conversion end
         LD       ADC1BUF,EA         ;  ADC1BUF ← ADC1 conversion data
         LD       A,#2H
         LD       ADMOD,A            ;  ADC2 pin select for A/D conversion
         BITS     ADSTR              ;  A/D conversion start
AD2CK    BTST     EOC                ;  A/D conversion end check
         JR       AD2CK              ;  A/D conversion not completed
         LD       EA,ADATA           ;  A/D conversion end
         LD       ADC2BUF,EA         ;  ADC2BUF ← ADC2 conversion data
```

14 FREQUENCY COUNTER

OVERVIEW

The microcontroller has a Frequency Counter (FC) to count the analog frequencies, FM, AM, or digital pulse. The block consists of the following components:

- Select of FMF or AMF
- 1/16 divider (for FMF)
- 24-bit counter

P2.2/FMF and P2.3/AMF signals also can be used to normal input port and count external events. When P2.2 is used to count analog FM frequency (FCMOD.0=1), input signal is connected through the block dividing by 16, but when digital pulse (FCMOD.0=0), input signal is connected directly, non- divided by 16.

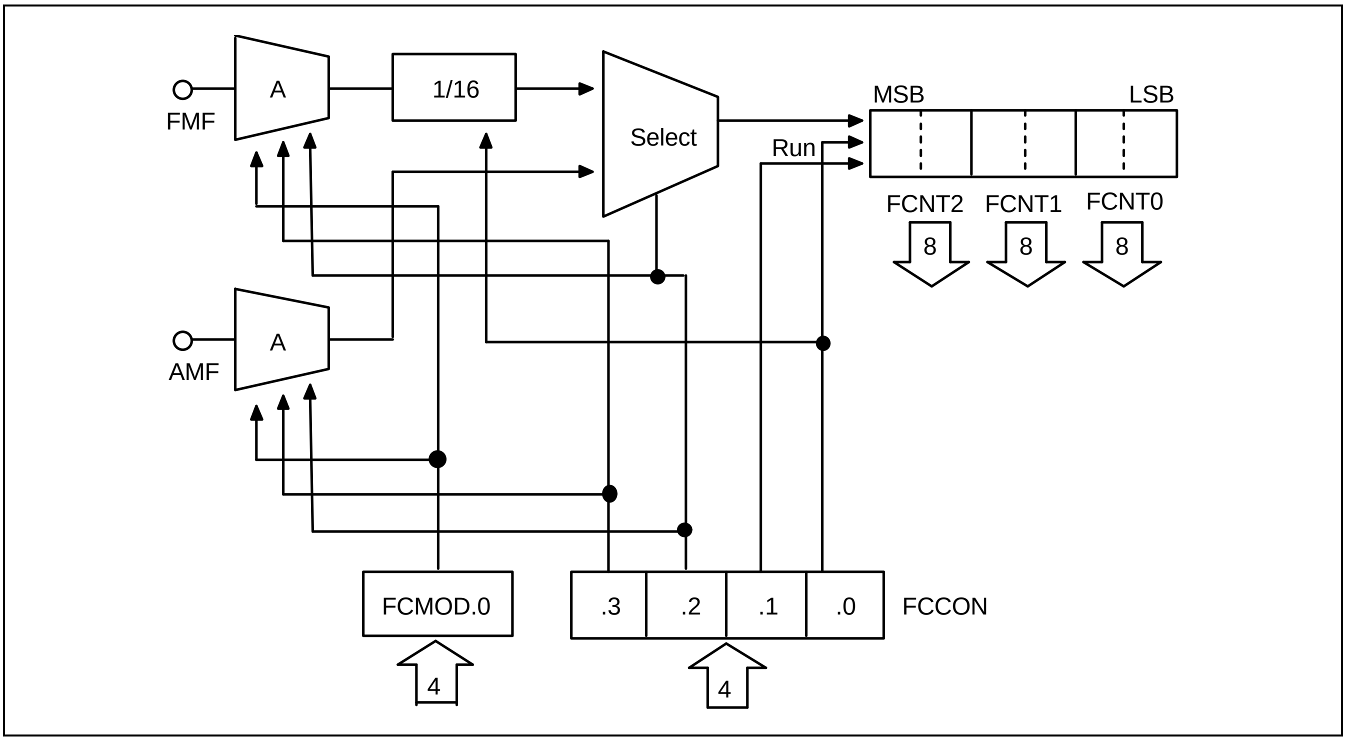


Figure 14-1. Frequency Counter Circuit Diagram

FC CONTROL REGISTER (FCCON)

FCCON is 4-bit register which can be used to enable/disable FC, select a signal of FMF or AMF and clear FCNT values and 1/16-divider. FCCON is write-only, 1-bit and 4-bit addressable.

FA7H	FCCON.3	FCCON.2	FCCON.1	FCCON.0

Table 14-1. Frequency Counter Control Register (FCCON) Bit Settings

Bit Name	Setting	Function
FCCON.3–2	–	Look at Table 14-2
FCCON.1	0	Stop increasing a counter value
	1	Start increasing a counter value
FCCON.0	1	Clear frequency values (FCNT0-2) and 1/16-divider. (This bit is automatically cleared at the next CPU cycle)

FC MODE REGISTER (FCMOD)

FCMOD is a 4-bit register which can be used to determine whether the P2.2 and P2.3 are of purpose with a frequency counter (analog input, 300mV$_{P-P}$) or general input port.

FA6H	"0"	"0"	"0"	FCMOD.0

Table 14-2. Frequency Counter Mode Register (FCMOD) Bit Settings

FCMOD.0	FCCON.3–2		Function
0	–	–	P2.2 and P2.3 are used as a general input port; also used to count DC pulse signal.
	x	0	Select P2.3 signal.
	x	1	Select P2.2 signal.
1	–	–	P2.2 or P2.3 are used to count AC frequency signal.
	0	0	Select AMF signal; disable FMF pin (feedback resistor is off; pull-down through resistor)
	0	1	Select only FMF, 1/16-divided; disable AMF pin (feedback resistor is off; pull-down through resistor)
	1	x	Disable all inputs and frequency counter (feedback resistors are off, pull-down through resistors)

NOTE: 'x' means 'don't care'.

Table 14-3. Frequency Counter Control Register (FCCON) Bit Settings in Normal Operating Mode

FCMOD.0	FCCON.3	FCCON.2	FMF's Feedback Resistor	AMF's Feedback Resistor	FMF's Pull-down	AMF's Pull-down
0	x	x	Off	Off	Off	Off
1	0	0	Off	On	On	Off
	0	1	On	Off	Off	On
	1	x	Off	Off	On	On

Table 14-4. Frequency Counter Control Register (FCCON) Bit Settings in Power-Down Mode

FCMOD.0	FCCON.3	FCCON.2	FMF's Feedback Resistor	AMF's Feedback Resistor	FMF's Pull-down	AMF's Pull-down
0	x	x	Off	Off	Off	Off
1	x	x	Off	Off	On	On

NOTE: 'x' means 'don't care'.

INPUT PIN CONFIGURATION FOR AN AC FREQUENCY COUNT

Because the FMF and AMF pins have built-in AC amplifiers, DC component of the input signal must be stripped off by the external capacitor.

When the FMF or FMF pin is selected for AC frequency counter function, the voltage level of the corresponding pin is increased to approximately 1/2 V_{DD} after a sufficiently long time. If the pin voltage does not increase, the AC amplifier exceeds its operating range and maybe cause an FC malfunction. To prevent this from occuring, you should program a sufficiently long time delay interval before starting the count operation.

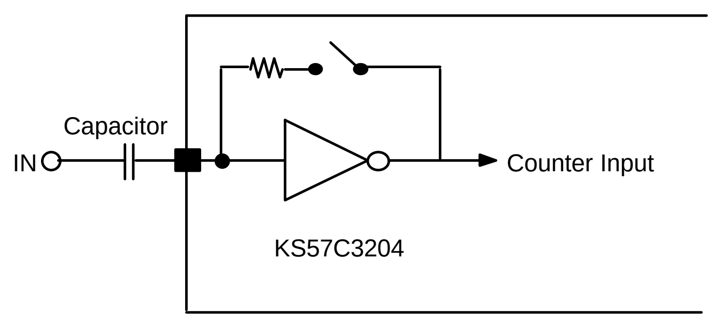


Figure 14-2. FMF and AMF Pin Configuration

☞ PROGRAMMING TIP — Using the Frequency Counter (FC)

In this Frequency counter program sample, the FMF pin are used for counting FM frequency, but the AMF pin is disabled:

```
RESET
         BITR      EMB
          LD       A,        #1H
          LD       FCMOD,  A               ;  Setting FMF and AMF as an analog frequency input
         LD        A,        #3H
         LD        BMOD,    A              ;  1.95ms
                   •
                   •
         LD        A,        #5H
         LD        FCCON,  A               ;  Enable FMF pin to count and clear counter values
         CALL      DELAY_20MS              ;  For stabilization
                   •
                   •
         BITS      BMOD.3                  ;  Start Basic timer
         BITR      FC_ENDFG                ;  Clear checking flag
         BTST      FC_ENDFG                ;  Whether FC time is over
         JR        $-2
         BITR      FC_ENDFG                ;  Clear checking flag
                   •

                   •
INTB
         BITR      EMB
                   •

                   •
         INCS      REG_625MS
         IRET
         INCS      REF_625MS+1
         IRET
         BITR      FCCON.1                 ;  Stop FC operation
         LD        EA,    FCNT0
         LD        REG_FCNT0, EA           ;  Load REG_FCNT0 ← FCNT0
         LD        EA,    FCNT1
         LD        REG_FCNT1, EA           ;  Load REG_FCNT1 ← FCNT1
         LD        EA,    FCNT2
         LD        REG_FCNT2, EA           ;  Load REG_FCNT2 ← FCNT2
         BITS      FCCON.0                 ;  Clear FC values
         BITS      FCCON.1                 ;  Start FC operation
         BITS      FC_ENDFG                ;  FC time is over
         LD        EA,  #CONST_625MS       ;  EA  ←  (100H – 32)
         LD        REG_625MS, EA           ;
         IRET
```

15 ELECTRICAL DATA

OVERVIEW

In this section, information on KS57C3204 electrical characteristics is presented as tables and graphics. The information is arranged in the following order:

Standard Electrical Characteristics

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

Miscellaneous Timing Waveforms

- A.C timing measurement point
- Clock timing measurement at X_{IN}
- Clock timing measurement at XT_{IN}
- Input timing for RESET
- Input timing for external interrupts

Stop Mode Characteristics and Timing Waveforms

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

Table 15-1. Absolute Maximum Ratings

(T_A = 25 °C)

Parameter	Symbol	Conditions	Rating	Units
Supply Voltage	V_{DD}	–	– 0.3 to + 6.5	V
Input Voltage	V_{IN}	All I/O ports	– 0.3 to V_{DD} + 0.3	
Output Voltage	V_O	–	– 0.3 to V_{DD} + 0.3	
Output Current High	I_{OH}	One I/O port active	– 15	mA
		All I/O ports active	– 30	
Output Current Low	I_{OL}	One I/O port active	+ 30 (Peak value)	
			+ 15 (note)	
		Total value for ports 1, 4, 5 and 6	+ 100 (Peak value) + 60 (note)	
Operating Temperature	T_A	–	– 40 to + 85	°C
Storage Temperature	T_{stg}	–	– 65 to + 150	

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

Table 15-2. D.C. Electrical Characteristics

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

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Input high voltage	V_{IH1}	All input pins except those specified below	0.7 V_{DD}	–	V_{DD}	V
	V_{IH2}	P1, P3, RESET, P2.0–1 and P6.1–3	0.8 V_{DD}		V_{DD}	
	V_{IH3}	X_{IN}, X_{OUT}, XT_{IN}, and XT_{OUT}	V_{DD} – 0.1		V_{DD}	
Input low voltage	V_{IL1}	All input pins except those specified below	–	–	0.3 V_{DD}	V
	V_{IL2}	P1, P3, RESET, P2.0–1 and P6.1–3			0.2 V_{DD}	
	V_{IL3}	X_{IN}, X_{OUT}, XT_{IN}, and XT_{OUT}			0.1	
Output high voltage	V_{OH1}	V_{DD} = 4.5 V to 5.5 V I_{OH} = – 1 mA Ports 1, 4, 5, and 6	V_{DD} – 1.0	–	–	V
	V_{OH2}	V_{DD} = 4.5 V to 5.5 V I_{OH} = –100 µA Port 8 and 9	V_{DD} – 2.0			

Table 15-2. D.C. Electrical Characteristics (Continued)

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

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Output low voltage	V_{OL1}	V_{DD} = 4.5 V to 5.5 V I_{OL} = 15 mA, Ports 1, 4, 5, and 6	–	0.4	2	V
	V_{OL2}	V_{DD} = 4.5 V to 5.5 V I_{OL} = 100 µA ; Ports 8and 9	–	–	1	
Input high leakage current (note)	I_{LIH1}	V_{IN} = V_{DD} All input pins	–	–	3	µA
Input low leakage current (note)	I_{LIL1}	V_{IN} = 0 V All input pins	–	–	– 3	
Output high leakage current (note)	I_{LOH1}	V_{OUT} = V_{DD} All output pins	–	–	3	
Output low leakage current (note)	I_{LOL}	V_{OUT} = 0 V All output pins			– 3	
Pull-up resistor	R_{L1}	V_{IN} = 0 V; V_{DD} = 5 V Ports 1, 2, 3, 4, 5, and 6	20	40	80	KΩ
		V_{DD} = 3 V	30	95	200	
	R_{L2}	V_{IN} = 0 V; V_{DD} = 5 V RESET	100	230	400	
		V_{DD} = 3 V	200	480	800	

NOTE: Except for X_{IN}, X_{OUT}, XT_{IN}, and XT_{OUT}

Table 15-2. D.C. Electrical Characteristics (Continued)

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

Parameter	Symbol	Conditions	Min	Typ	Max	Units
LCD voltage dividing resistor	R_{LCD}	TA = 25 °C	60	84	130	KΩ
COM output impedance	R_{COM}	V_{DD} = 5 V	–	3	6	
		V_{DD} = 3 V		5	15	
SEG output impedance	R_{SEG}	V_{DD} = 5 V	–	3	6	
		V_{DD} = 3 V		5	15	
COM output voltage deviation	VDC	VDD = 5 V (VLC0-COMi) Io = ± 15uA (I = 0–3)	–	± 45	± 90	mV
SEG output voltage deviation	VDS	V_{DD} = 5 V (VLC0-SEGi) Io = ± 15uA (I = 0–27)	–	± 45	± 90	
Oscillator feedback resistor	R_{OSC1}	V_{DD} = 5.0 V; T_A = 25; X_{IN} = V_{DD}, X_{OUT} = 0 V	300	600	1500	KΩ
	R_{OSC2}	V_{DD} = 5.0 V; T_A = 25; XT_{IN} = V_{DD}, XT_{OUT} = 0 V	1230	2630	4000	

Table 15-2. D.C. Electrical Characteristics (Concluded)

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

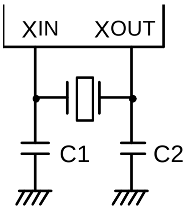
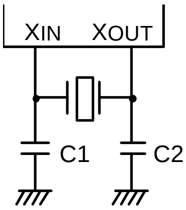
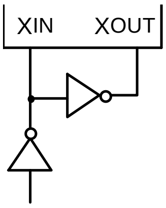
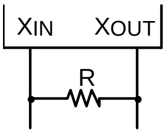
Parameter	Symbol	Conditions		Min	Typ	Max	Units
Supply Current[1]	I_{DD1}	Main operating: FC enable PCON = 0011B, SCMOD = 0000B Crystal oscillator C1 = C2 = 22 pF V_{DD} = 5 V ± 10%	4.19 MHz	–	5.2	10	mA
	I_{DD2} [2]	Main operating: PCON = 0011B, SCMOD = 0000B Crystal oscillator C1 = C2 = 22 pF V_{DD} = 5 V ± 10%	6.0 MHz	–	3.5	8	
			4.19 MHz		2.5	5.5	
		V_{DD} = 3 V ± 10%	6.0 MHz		1.6	4	
			4.19 MHz		1.2	3	
	I_{DD3} [2]	Main idle mode[3]: PCON = 0111B, SCMOD =0000B Crystal oscillator C1 = C2 = 22 pF V_{DD} = 5 V ± 10%	6.0 MHz	–	1.0	2.5	
			4.19 MHz		0.9	2.0	
		V_{DD} = 3 V ± 10%	6.0 MHz		0.5	1.0	
			4.19 MHz		0.4	0.8	
	I_{DD4}[2]	Sub operating mode: PCON = 0011B, SCMOD = 1001B V_{DD} = 3 V ± 10% 32 kHz crystal oscillator		–	15	30	uA
	I_{DD5} [2]	Sub idle mode: PCON = 0111B, SCMOD = 1001B V_{DD} = 3 V ± 10% 32 kHz crystal oscillator		–	6	15	
	IDD6[2]	Stop mode: CPU = fxt/4, SCMOD = 1101B V_{DD} = 5 V ± 10%		–	0.5	3	
	I_{DD7}[2]	Stop mode: CPU = fx/4, SCMOD = 0100B V_{DD} = 5 V ± 10%		–			

NOTES:

1. Supply current does not include current drawn through internal pull-up resistors and LCD voltage dividing resistors.
2. AMF or FMF is a normal input mode.
3. Data includes the power consumption for sub-system clock oscillation.

Table 15-3. Main System Clock Oscillator Characteristics

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

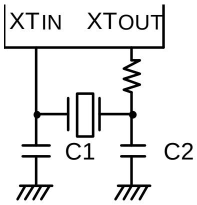
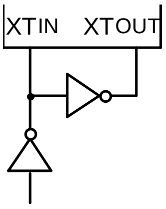
Oscillator	Clock Configuration	Parameter	Test Condition	Min	Typ	Max	Units
Ceramic Oscillator		Oscillation frequency [1]	–	0.4	–	6.0	MHz
		Stabilization time [2]	Stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	4	ms
Crystal Oscillator		Oscillation frequency [1]	–	0.4	–	6.0	MHz
		Stabilization time [2]	V_{DD} = 2.7 V to 5.5 V	–	–	10	ms
			V_{DD} = 1.8 V to 2.7 V	–	–	30	
External Clock		X_{IN} input frequency [1]	–	0.4	–	6.0	MHz
		X_{IN} input high and low level width (t_{XH}, t_{XL})	–	83.3	–	–	ns
RC Oscillator		Frequency [1]	V_{DD} = 5 V R = 15 KΩ, V_{DD} = 5 V R = 25 KΩ, V_{DD} = 3 V	0.4	– 2.0 1.0	2.5	MHz

NOTES:

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

Table 15-4. Subsystem Clock Oscillator Characteristics

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

Oscillator	Clock Configuration	Parameter	Test Condition	Min	Typ	Max	Units
Crystal Oscillator		Oscillation frequency (1)	–	32	32.768	35	kHz
		Stabilization time (2)	V_{DD} = 2.7 V to 5.5 V	–	1.0	2	s
			V_{DD} = 1.8 V to 2.7 V	–	–	10	
External Clock		XT_{IN} input frequency (1)	–	32	–	100	kHz
		XT_{IN} input high and low level width (t_{XTL}, t_{XTH})	–	5	–	15	µs

NOTES:

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

Table 15-5. Input/Output Capacitance

(T_A = 25 °C, V_{DD} = 0 V)

Parameter	Symbol	Condition	Min	Typ	Max	Units
Input capacitance	C_{IN}	f_{CLK} = 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 15-6. A.C. Electrical Characteristics

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

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Instruction cycle time (1)	t_{CY}	V_{DD} = 2.7 V to 5.5 V	0.67	–	64	µs
		V_{DD} = 1.8 V to 5.5 V	1.3		64	
Interrupt input high, low width	t_{INTH}, t_{INTL}	INT0	(2)	–	–	µs
		INT1, INT2, INT4, KS0–KS2	10			
RESET Input Low Width	t_{RSL}	Input	10	–	–	µs

NOTES:

1. Unless otherwise specified, Instruction Cycle Time condition values assume a main system clock/4 (fx/4) source.
2. Minimum value for INT0 is based on a clock of $2t_{CY}$ or 128/fxx as assigned by the IMOD0 register setting.

Table 15-6. A.C. Electrical Characteristics (continued)

(T_A = – 10 °C to + 70 °C, V_{DD} = 3.5 V to 5.5 V)

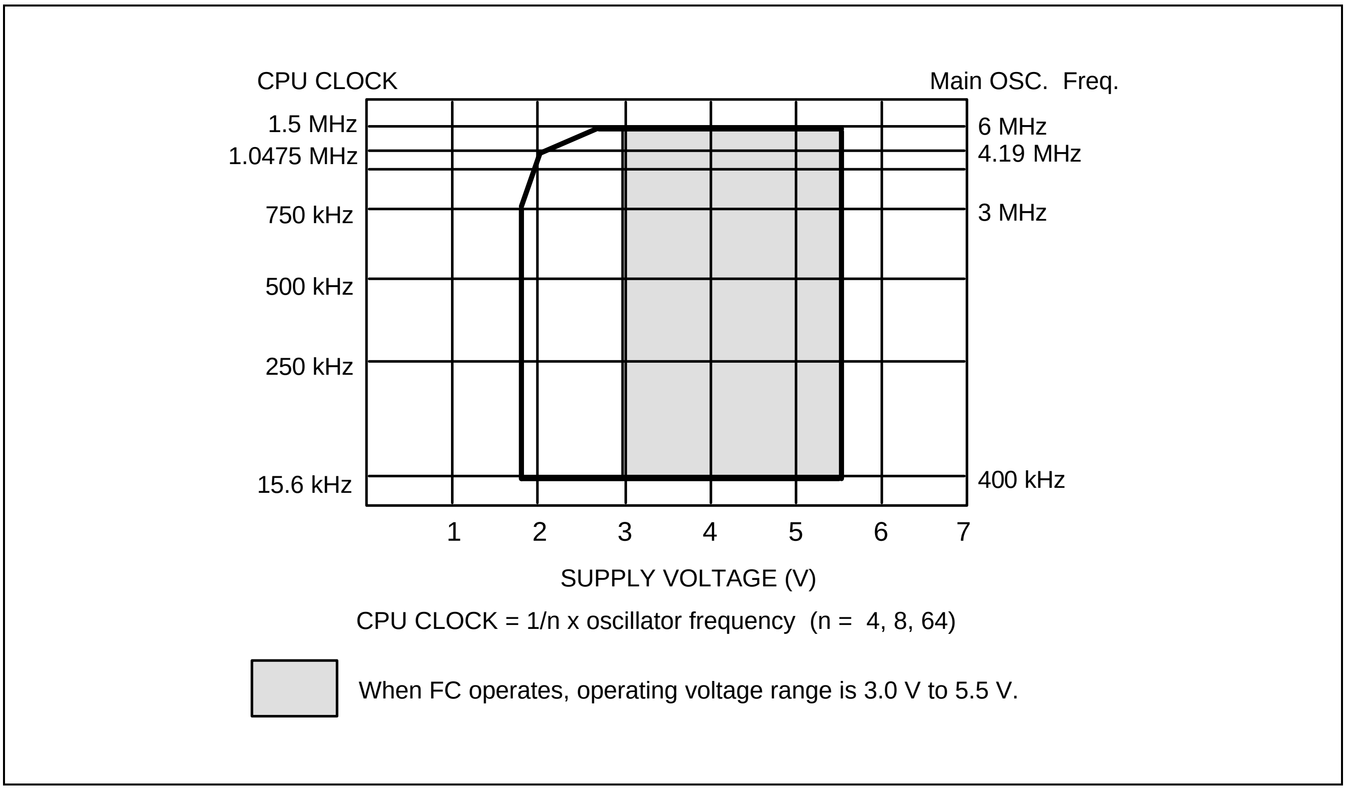
Parameter	Symbol	Conditions	Min	Typ	Max	Units
A/D converting Resolution	–	–	8	8	8	bits
Absolute accuracy	–	–	–	–	± 2	LSB
AD conversion time	t_{CON}	–	17	34/fxx (note)	–	µs
Analog input voltage	V_{IAN}	–	V_{SS}	–	V_{DD}	V
Analog input impedance	R_{AN}	–	2	1000	–	MΩ

NOTE: fxx stands for the system clock (fx or fxt).

Table 15-6. A.C. Electrical Characteristics (continued)

(T_A = – 40 °C to + 85 °C, V_{DD} = 3.0 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Input voltage (peak to peak)	V_{IN}	AMF/FMF mode, sine wave input	0.3	–	V_{DD}	V
Frequency	f_{AMF}	AMF mode, sine wave input; V_{IN} = 300mV$_{P-P}$	0.5	–	10	MHz
	f_{FMF}	FMF mode, sine wave input; V_{IN} = 300mV$_{P-P}$	30		150	



CPU CLOCK = 1/n x oscillator frequency (n = 4, 8, 64)

▨ When FC operates, operating voltage range is 3.0 V to 5.5 V.

Figure 15-1. Standard Operating Voltage Range

Table 15-7. RAM Data Retention Supply Voltage in Stop Mode

(T_A = – 40 °C to + 85 °C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data retention supply voltage	V_{DDDR}	Normal operation	1.8	–	5.5	V
Data retention supply current	I_{DDDR}	V_{DDDR} = 1.8 V	–	0.1	1	µA

TIMING WAVEFORMS

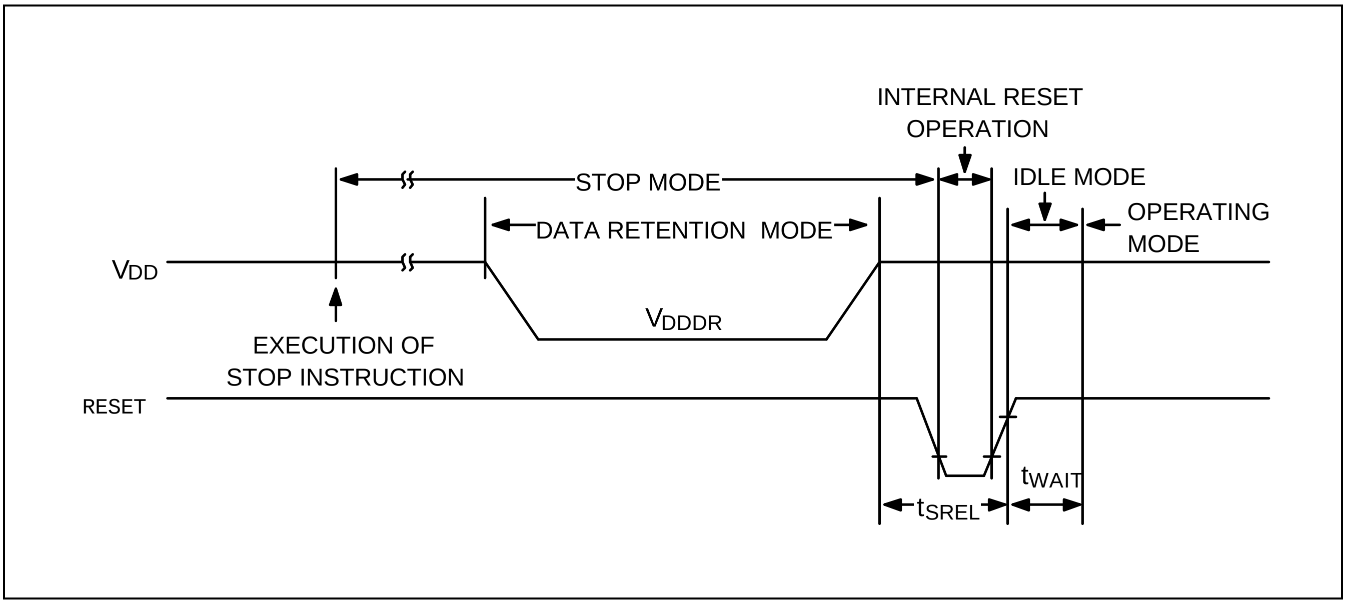


Figure 15-2. Stop Mode Release Timing When Initiated by RESET

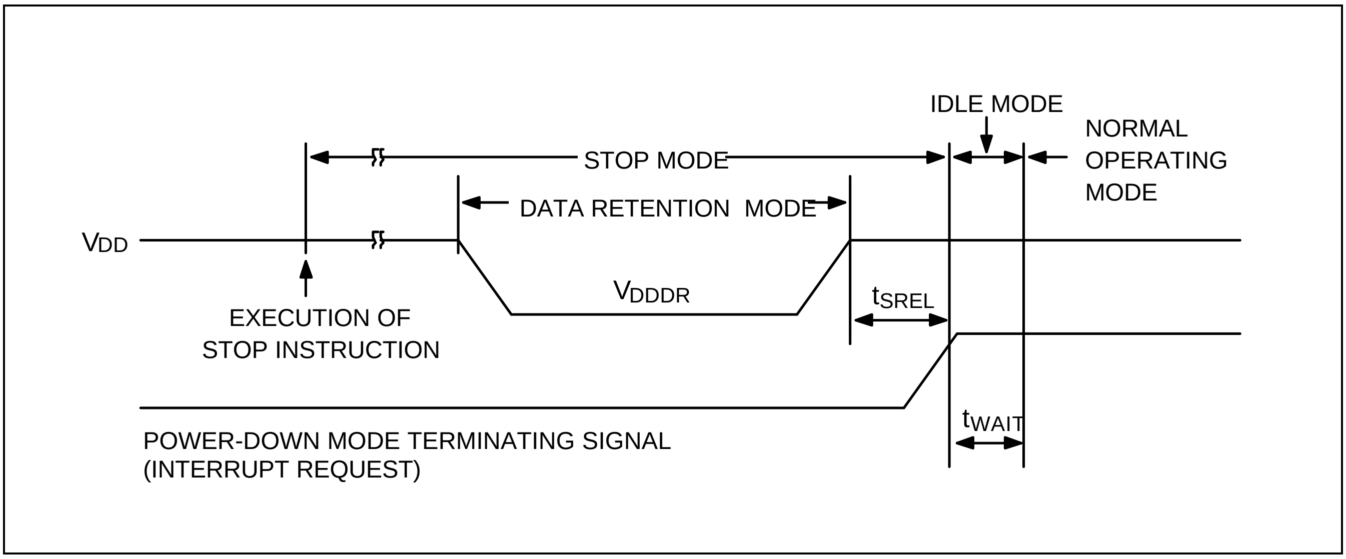


Figure 15-3. Stop Mode Release Timing When Initiated by an Interrupt Request

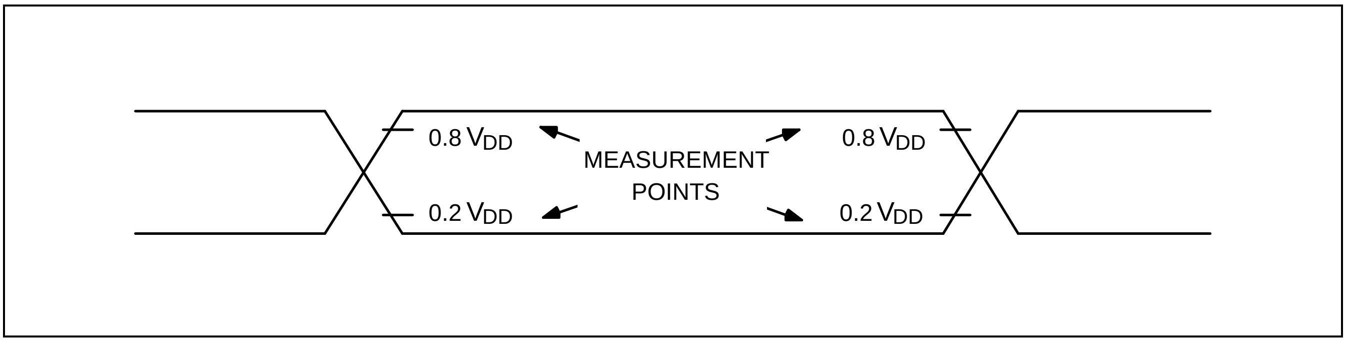


Figure 15-4. A.C. Timing Measurement Points (Except for X_{in} and XT_{in})

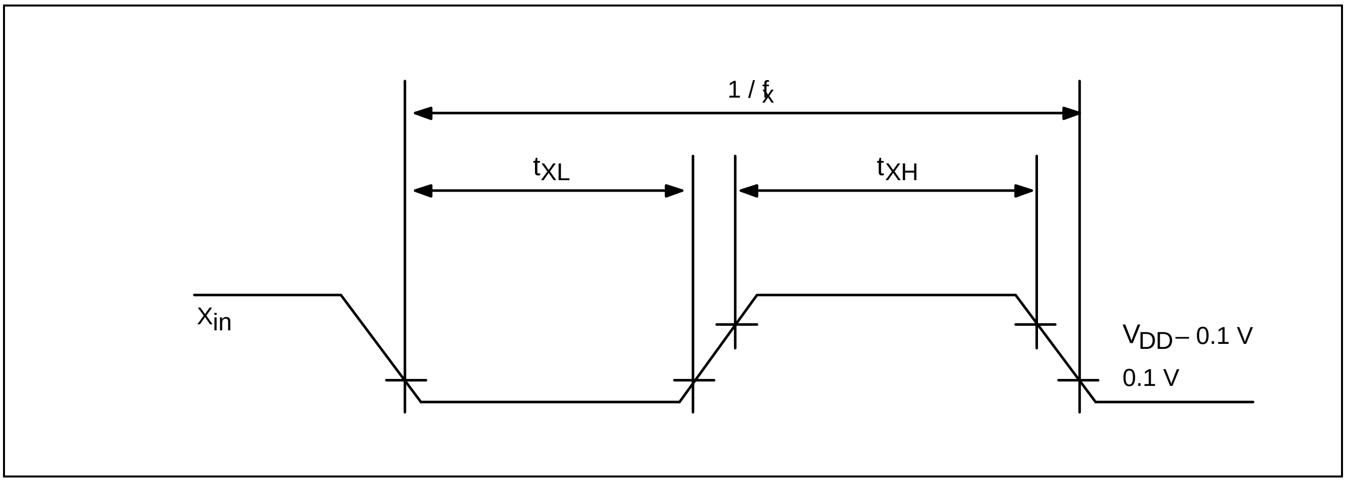


Figure 15-5. Clock Timing Measurement at X_{in}

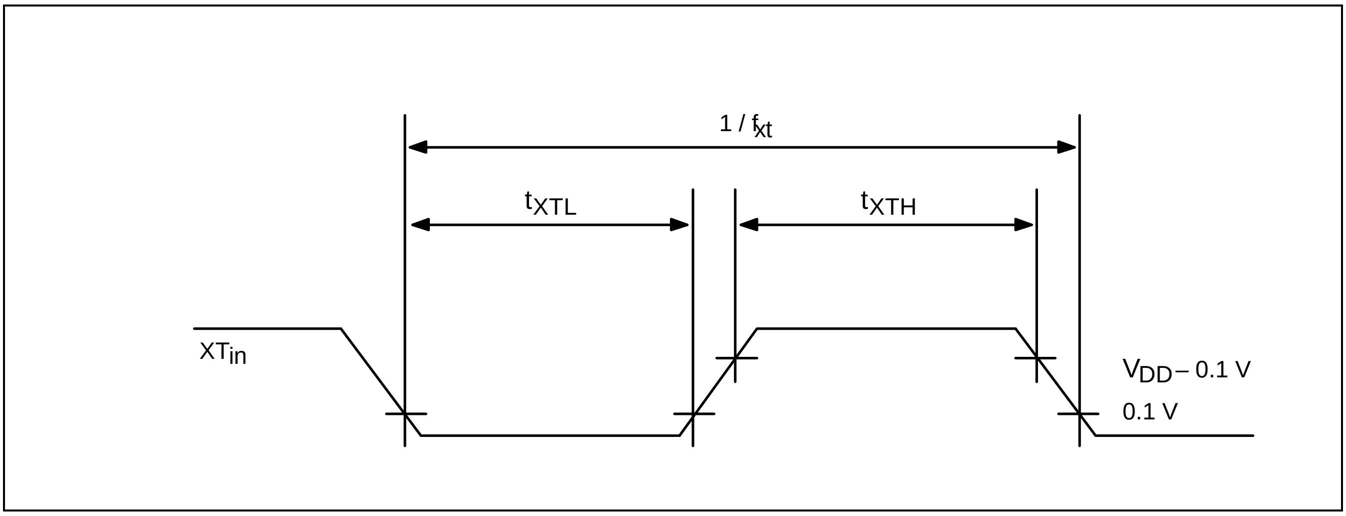


Figure 15-6. Clock Timing Measurement at XT_{in}

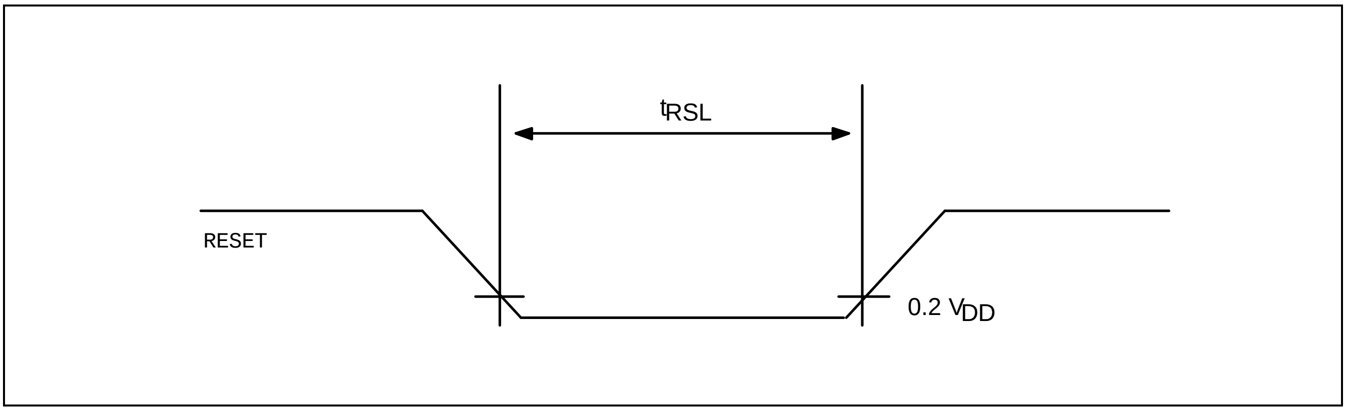


Figure 15-7. Input Timing for RESET Signal

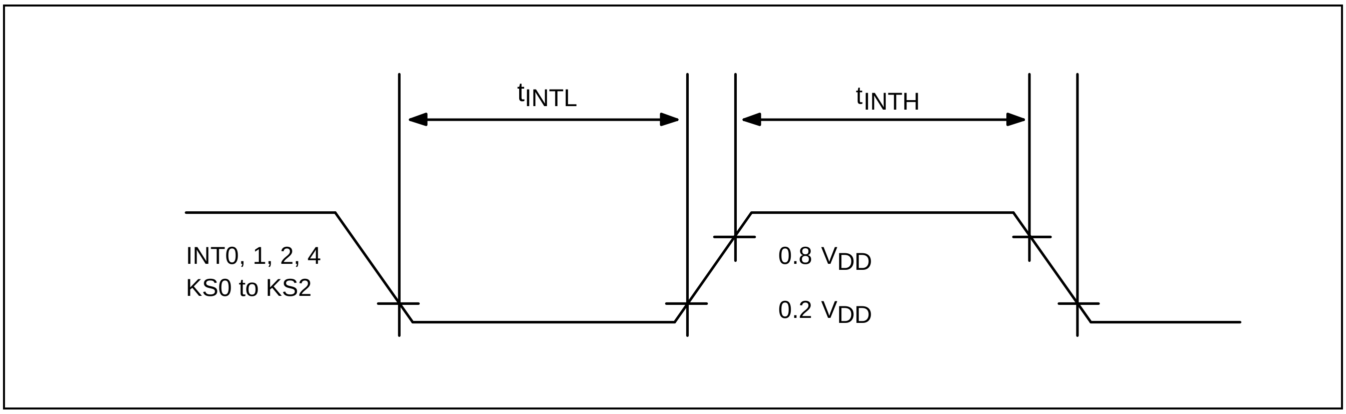
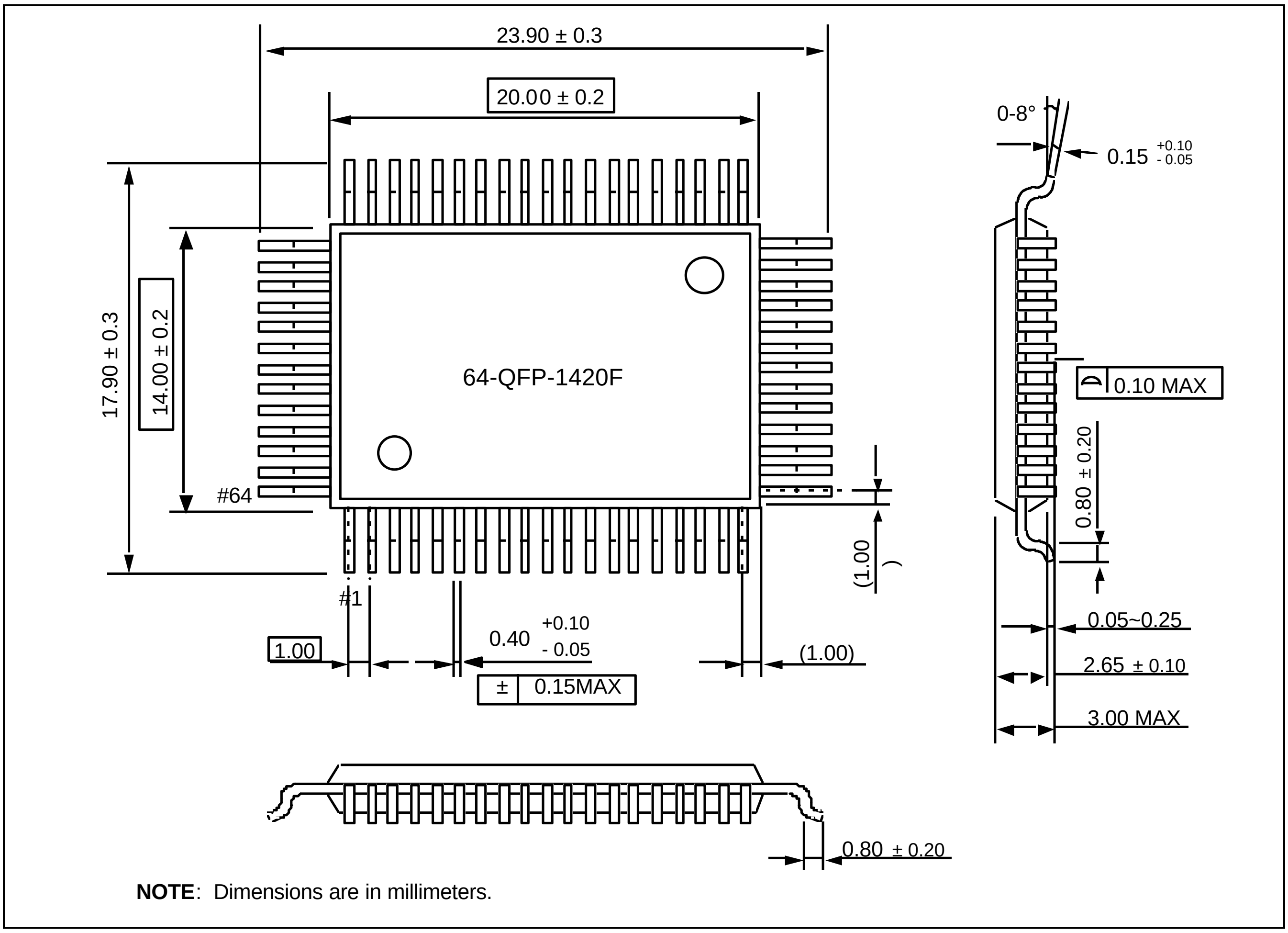


Figure 15-8. Input Timing for External Interrupts and Quasi-Interrupts

16 MECHANICAL DATA

OVERVIEW

The KS57C3204 microcontroller is available in a 64-pin QFP package (Samsung: 64-QFP-1420F). Package dimensions are shown in Figure 16-1.



NOTE: Dimensions are in millimeters.

Figure 16-1. 64-QFP-1420F Package Dimensions

NOTES

17 KS57P3204 OTP

OVERVIEW

The KS57P3204 single-chip CMOS microcontroller is the OTP (One Time Programmable) version of the KS57C3204microcontroller. It has an on-chip EPROM instead of masked ROM. The EPROM is accessed by a serial data format.

The KS57P3204 is fully compatible with the KS57C3204, both in function and in pin configuration. Because of its simple programming requirements, the KS57P3204 is ideal for use as an evaluation chip for the KS57C3204.

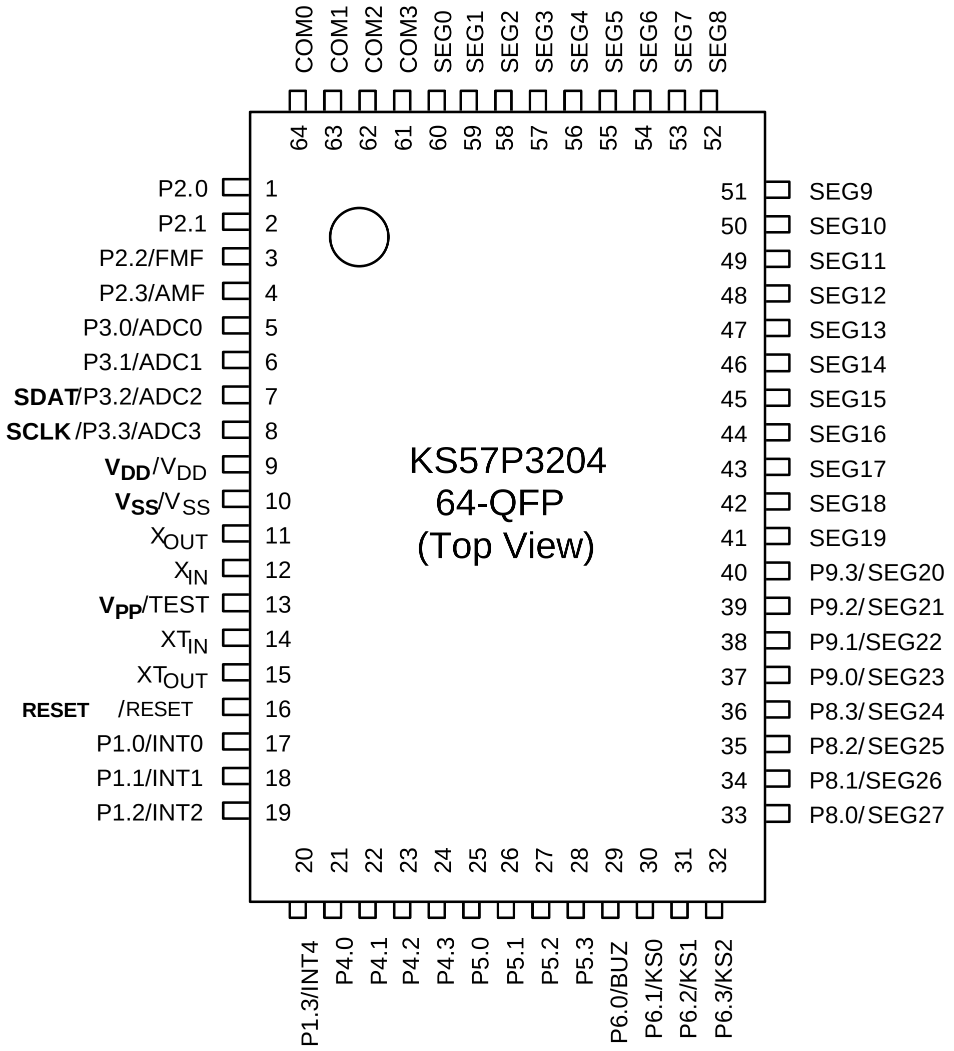


Figure 17-1. KS57P3204 Pin Assignments (64-QFP)

Table 17-1. Pin Descriptions Used to Read/Write the EPROM

Main Chip	During Programming			
Pin Name	Pin Name	Pin No.	I/O	Function
P3.2	SDAT	7	I/O	Serial data pin. Output port when reading and input port when writing. Can be assigned as a Input or push-pull output port.
P3.3	SCLK	8	I/O	Serial clock pin. Input only pin.
TEST	V_{PP} (TEST)	13	I	Power supply pin for EPROM cell writing (indicates that OTP enters into the writing mode). When 12.5 V is applied, OTP is in writing mode and when 5 V is applied, OTP is in reading mode.
RESET	RESET	16	I	Chip initialization
V_{DD} / V_{SS}	V_{DD} / V_{SS}	9/10	I	Logic power supply pin. V_{DD} should be tied to +5 V during programming.

Table 17-2. Comparison of KS57P3204 and KS57C3204 Features

Characteristic	KS57P3204	KS57C3204
Program Memory	4K bytes EPROM	4K bytes mask ROM
Operating Voltage (V_{DD})	2.0 V to 5.5 V at 4.19 MHz 1.8 V to 5.5 V at 3 MHz	2.0 V to 5.5 V at 4.19 MHz 1.8 V to 5.5 V at 3 MHz
OTP Programming Mode	V_{DD} = 5 V, V_{PP} (TEST) = 12.5 V	–
Pin Configuration	64 QFP	64 QFP
EPROM Programmability	User Program 1 time	Programmed at the factory

OPERATING MODE CHARACTERISTICS

When 12.5 V is supplied to the Vpp (TEST) pin of the KS57P3204, the EPROM programming mode is entered. The operating mode (read, write, or read protection) is selected according to the input signals to the pins listed in Table 17-3 below.

Table 17-3. Operating Mode Selection Criteria

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

NOTE: "0" means low level; "1" means high level.

Table 17-4. D.C. Electrical Characteristics

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

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Input high voltage	V_{IH1}	All input pins except those specified below	0.7 V_{DD}	–	V_{DD}	V
	V_{IH2}	P1, P3, RESET, P2.0–1 and P6.1–3	0.8 V_{DD}		V_{DD}	
	V_{IH3}	X_{IN}, X_{OUT}, XT_{IN}, and XT_{OUT}	V_{DD} – 0.1		V_{DD}	
Input low voltage	V_{IL1}	All input pins except those specified below	–	–	0.3 V_{DD}	V
	V_{IL2}	P1, P3, RESET, P2.0–1 and P6.1–3			0.2 V_{DD}	
	V_{IL3}	X_{IN}, X_{OUT}, XT_{IN}, and XT_{OUT}			0.1	
Output high voltage	V_{OH1}	V_{DD} = 4.5 V to 5.5 V I_{OH} = – 1 mA Ports 1, 4, 5, and 6	V_{DD} – 1.0	–	–	V
	V_{OH2}	V_{DD} = 4.5 V to 5.5 V I_{OH} = –100 µA Port 8 and 9	V_{DD} – 2.0			

Table 17-4. D.C. Electrical Characteristics (Continued)

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

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Output low voltage	V_{OL1}	V_{DD} = 4.5 V to 5.5 V I_{OL} = 15 mA, Ports 1, 4, 5, and 6	–	0.4	2	V
	V_{OL2}	V_{DD} = 4.5 V to 5.5 V I_{OL} = 100 µA ; Ports 8 and 9	–	–	1	
Input high leakage current(note)	I_{LIH1}	V_{IN} = V_{DD} All input pins	–	–	3	µA
Input low leakage current(note)	I_{LIL1}	V_{IN} = 0 V All input pins	–	–	– 3	
Output high leakage current(note)	I_{LOH1}	V_{OUT} = V_{DD} All output pins	–	–	3	
Output low leakage current (note)	I_{LOL}	V_{OUT} = 0 V All output pins	–	–	– 3	
Pull-up resistor	R_{L1}	V_{IN} = 0 V; V_{DD} = 5 V Ports 1, 2, 3, 4, 5, and 6	20	40	80	KΩ
		V_{DD} = 3 V	30	95	200	
	R_{L2}	V_{IN} = 0 V; V_{DD} = 5 V RESET	100	230	400	
		V_{DD} = 3 V	200	480	800	

NOTE: Except for X_{IN}, X_{OUT}, XT_{IN}, and XT_{OUT}

Table 17-4. D.C. Electrical Characteristics (Continued)

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

Parameter	Symbol	Conditions	Min	Typ	Max	Units
LCD voltage dividing resistor	R_{LCD}	T_A = 25 °C	60	84	130	KΩ
COM output impedance	R_{COM}	V_{DD} = 5 V	–	3	6	
		V_{DD} = 3 V		5	15	
SEG output impedance	R_{SEG}	V_{DD} = 5 V	–	3	6	
		V_{DD} = 3 V		5	15	
COM output voltage deviation	V_{DC}	V_{DD} = 5 V (V_{LC0}-COMi) Io = ± 15uA (I = 0–3)	–	± 45	± 90	mV
SEG output voltage deviation	V_{DS}	V_{DD} = 5 V (V_{LC0}-SEGi) Io = ± 15uA (I = 0–27)	–	± 45	± 90	
Oscillator feedback resistor	R_{OSC1}	V_{DD} = 5.0 V; T_A = 25; X_{IN} = V_{DD}, X_{OUT} = 0 V	300	600	1500	KΩ
	R_{OSC2}	V_{DD} = 5.0 V; T_A = 25; XT_{IN} = V_{DD}, XT_{OUT} = 0 V	1230	2630	4000	

Table 17-4. D.C. Electrical Characteristics (Concluded)

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

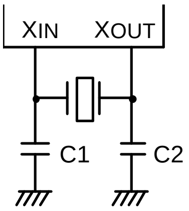
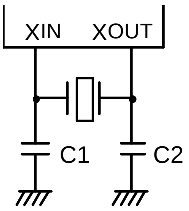
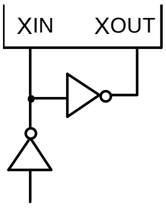
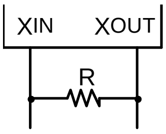
Parameter	Symbol	Conditions		Min	Typ	Max	Units
Supply Current[1]	I_{DD1}	Main operating: FC enable PCON = 0011B, SCMOD = 0000B Crystal oscillator C1 = C2 = 22 pF V_{DD} = 5 V ± 10%	4.19 MHz	–	5.2	10	mA
	I_{DD2} [2]	Main operating: PCON = 0011B, SCMOD = 0000B Crystal oscillator C1 = C2 = 22 pF V_{DD} = 5 V ± 10%	6.0 MHz	–	3.5	8	
			4.19 MHz		2.5	5.5	
		V_{DD} = 3 V ± 10%	6.0 MHz		1.6	4	
			4.19 MHz		1.2	3	
	I_{DD3} [2]	Main idle mode[3]: PCON = 0111B, SCMOD =0000B Crystal oscillator C1 = C2 = 22 pF V_{DD} = 5 V ± 10%	6.0 MHz	–	1.0	2.5	
			4.19 MHz		0.9	2.0	
		V_{DD} = 3 V ± 10%	6.0 MHz		0.5	1.0	
			4.19 MHz		0.4	0.8	
	I_{DD4}[2]	Sub operating mode: PCON = 0011B, SCMOD = 1001B V_{DD} = 3 V ± 10% 32 kHz crystal oscillator		–	15	30	uA
	I_{DD5} [2]	Sub idle mode: PCON = 0111B, SCMOD = 1001B V_{DD} = 3 V ± 10% 32 kHz crystal oscillator		–	6	15	
	IDD6[2]	Stop mode: CPU = fxt/4, SCMOD = 1101B V_{DD} = 5 V ± 10%		–	0.5	3	
	I_{DD7}[2]	Stop mode: CPU = fx/4, SCMOD = 0100B V_{DD} = 5 V ± 10%		–			

NOTES:

1. Supply current does not include current drawn through internal pull-up resistors and LCD voltage dividing resistors.
2. AMF or FMF is a normal input mode.
3. Data includes the power consumption for sub-system clock oscillation.

Table 17-5. Main System Clock Oscillator Characteristics

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

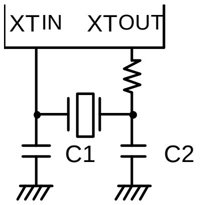
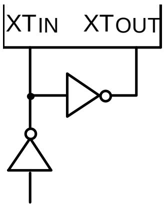
Oscillator	Clock Configuration	Parameter	Test Condition	Min	Typ	Max	Units
Ceramic Oscillator		Oscillation frequency [1]	–	0.4	–	6.0	MHz
		Stabilization time [2]	Stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	4	ms
Crystal Oscillator		Oscillation frequency [1]	–	0.4	–	6.0	MHz
		Stabilization time [2]	V_{DD} = 2.7 V to 5.5 V	–	–	10	ms
			V_{DD} = 1.8 V to 2.7 V	–	–	30	
External Clock		X_{IN} input frequency [1]	–	0.4	–	6.0	MHz
		X_{IN} input high and low level width (t_{XH}, t_{XL})	–	83.3	–	–	ns
RC Oscillator		Frequency [1]	V_{DD} = 5 V R = 15 KΩ, V_{DD} = 5 V R = 25 KΩ, V_{DD} = 3 V	0.4	– 2.0 1.0	2.5	MHz

NOTES:

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

Table 17-6. Subsystem Clock Oscillator Characteristics

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

Oscillator	Clock Configuration	Parameter	Test Condition	Min	Typ	Max	Units
Crystal Oscillator		Oscillation frequency (1)	–	32	32.768	35	kHz
		Stabilization time (2)	V_{DD} = 2.7 V to 5.5 V	–	1.0	2	s
			V_{DD} = 1.8 V to 2.7 V	–	–	10	
External Clock		XT_{IN} input frequency (1)	–	32	–	100	kHz
		XT_{IN} input high and low level width (t_{XTL}, t_{XTH})	–	5	–	15	µs

NOTES:

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

Table 17-7. Input/Output Capacitance

(T_A = 25 °C, V_{DD} = 0 V)

Parameter	Symbol	Condition	Min	Typ	Max	Units
Input capacitance	C_{IN}	f_{CLK} = 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 17-8. A.C. Electrical Characteristics

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

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Instruction cycle time (1)	t_{CY}	V_{DD} = 2.7 V to 5.5 V	0.67	–	64	µs
		V_{DD} = 1.8 V to 5.5 V	1.3		64	
Interrupt input high, low width	t_{INTH}, t_{INTL}	INT0	(2)	–	–	µs
		INT1, INT2, INT4, KS0–KS2	10			
RESET Input Low Width	t_{RSL}	Input	10	–	–	µs

NOTES:

1. Unless otherwise specified, Instruction Cycle Time condition values assume a main system clock/4 (fx/4) source.
2. Minimum value for INT0 is based on a clock of $2t_{CY}$ or 128/fxx as assigned by the IMOD0 register setting.

Table 17-8. A.C. Electrical Characteristics (continued)

(T_A = – 10 °C to + 70 °C, V_{DD} = 3.5 V to 5.5 V)

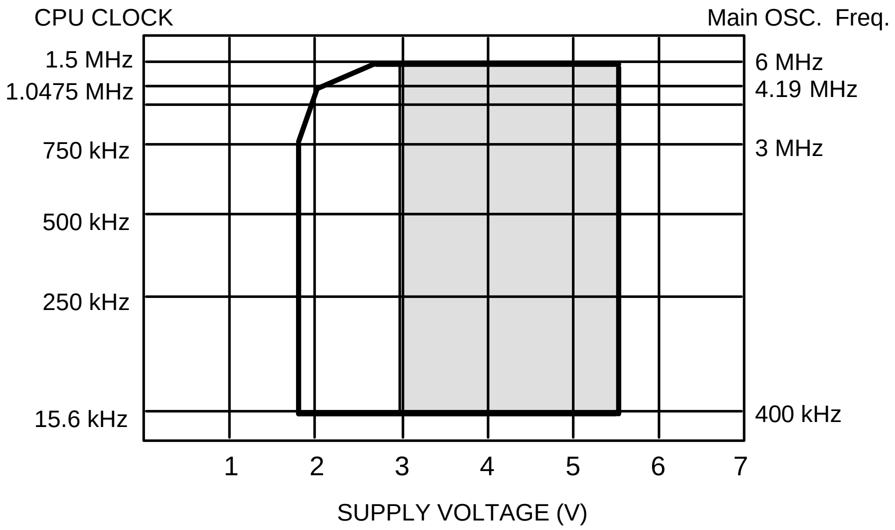
Parameter	Symbol	Conditions	Min	Typ	Max	Units
A/D converting Resolution	–	–	8	8	8	bits
Absolute accuracy	–	–	–	–	± 2	LSB
AD conversion time	t_{CON}	–	17	34/fxx (note)	–	µs
Analog input voltage	V_{IAN}	–	V_{SS}	–	V_{DD}	V
Analog input impedance	R_{AN}	–	2	1000	–	MΩ

NOTE: fxx stands for the system clock (fx or fxt).

Table 17-8. A.C. Electrical Characteristics (continued)

(T_A = – 40 °C to + 85 °C, V_{DD} = 3.0 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Input voltage (peak to peak)	V_{IN}	AMF/FMF mode, sine wave input	0.3	–	V_{DD}	V
Frequency	f_{AMF}	AMF mode, sine wave input; V_{IN} = 300mV$_{P-P}$	0.5	–	10	MHz
	f_{FMF}	FMF mode, sine wave input; V_{IN} = 300mV$_{P-P}$	30		150	



CPU CLOCK = 1/n x oscillator frequency (n = 4, 8, 64)

▨ When FC operates, operating voltage range is 3.0 V to 5.5 V.

Figure 17-2. Standard Operating Voltage Range

Table 17-9. RAM Data Retention Supply Voltage in Stop Mode

(T_A = – 40 °C to + 85 °C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data retention supply voltage	V_{DDDR}	Normal operation	1.8	–	5.5	V
Data retention supply current	I_{DDDR}	V_{DDDR} = 1.8 V	–	0.1	1	µA

TIMING WAVEFORMS

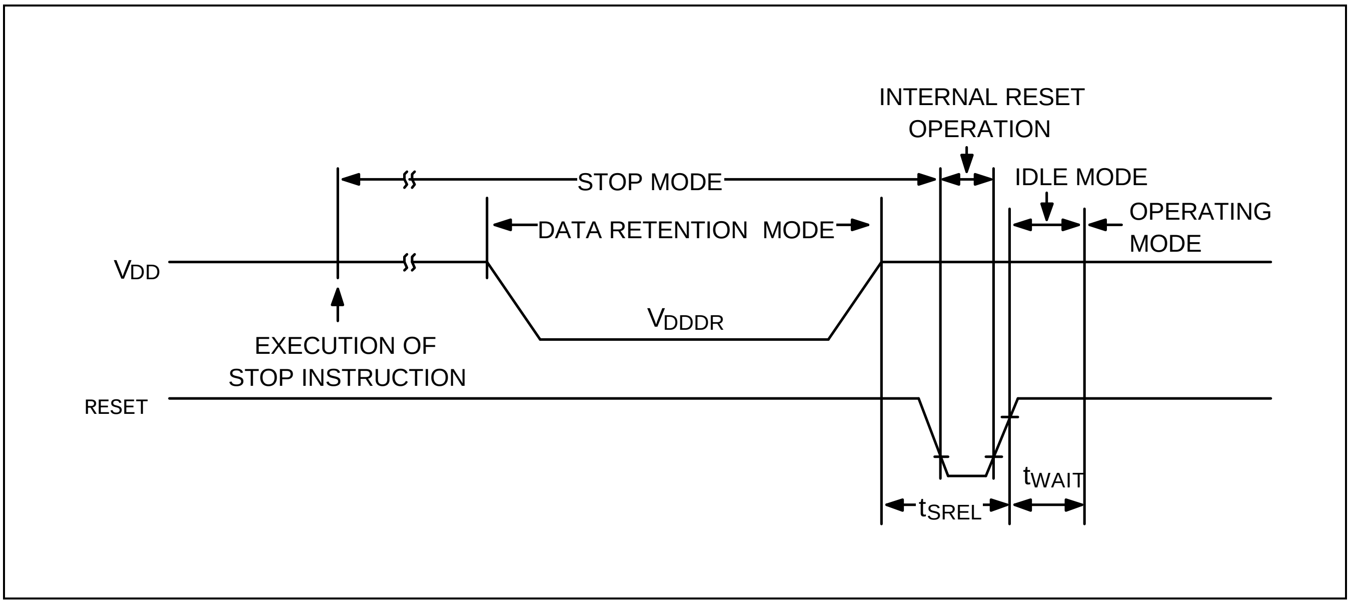


Figure 17-3. Stop Mode Release Timing When Initiated by RESET

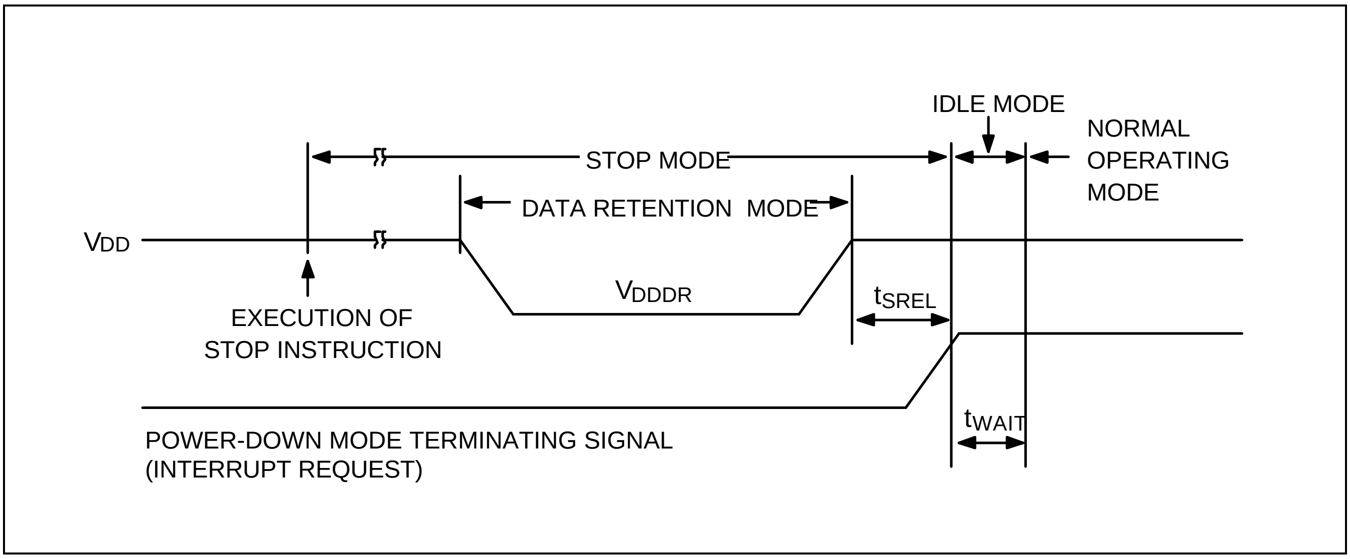


Figure 17-4. Stop Mode Release Timing When Initiated by an Interrupt Request

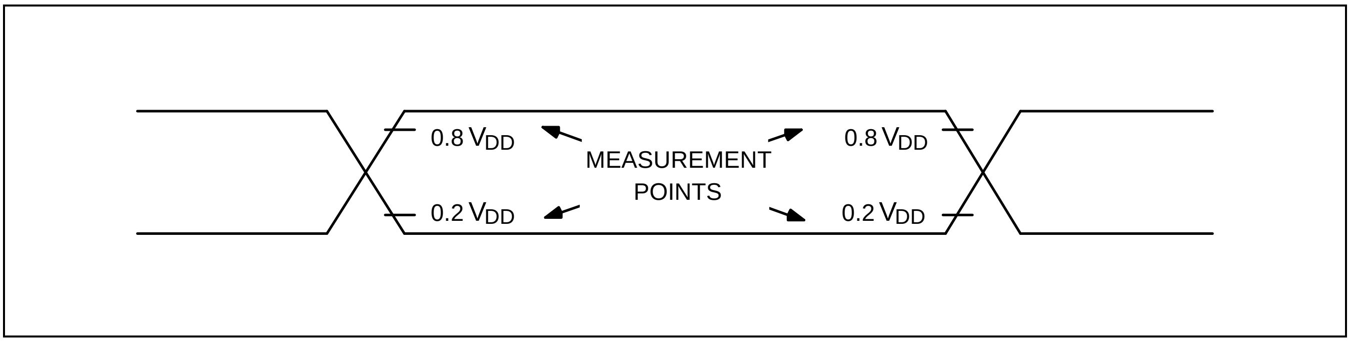


Figure 17-5. A.C. Timing Measurement Points (Except for X_{in} and XT_{in})

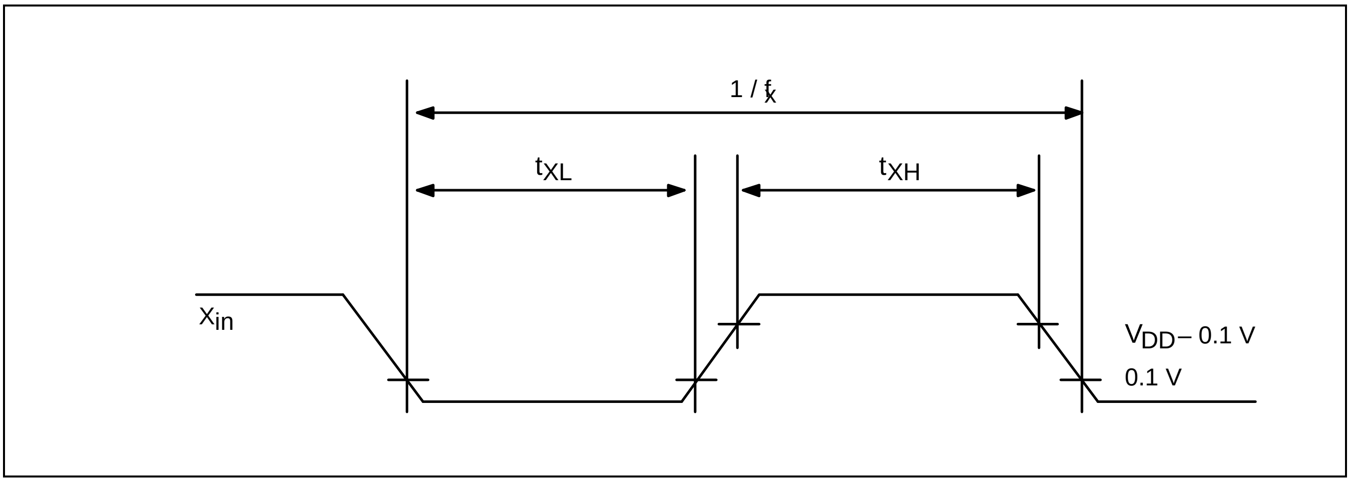


Figure 17-6. Clock Timing Measurement at X_{in}

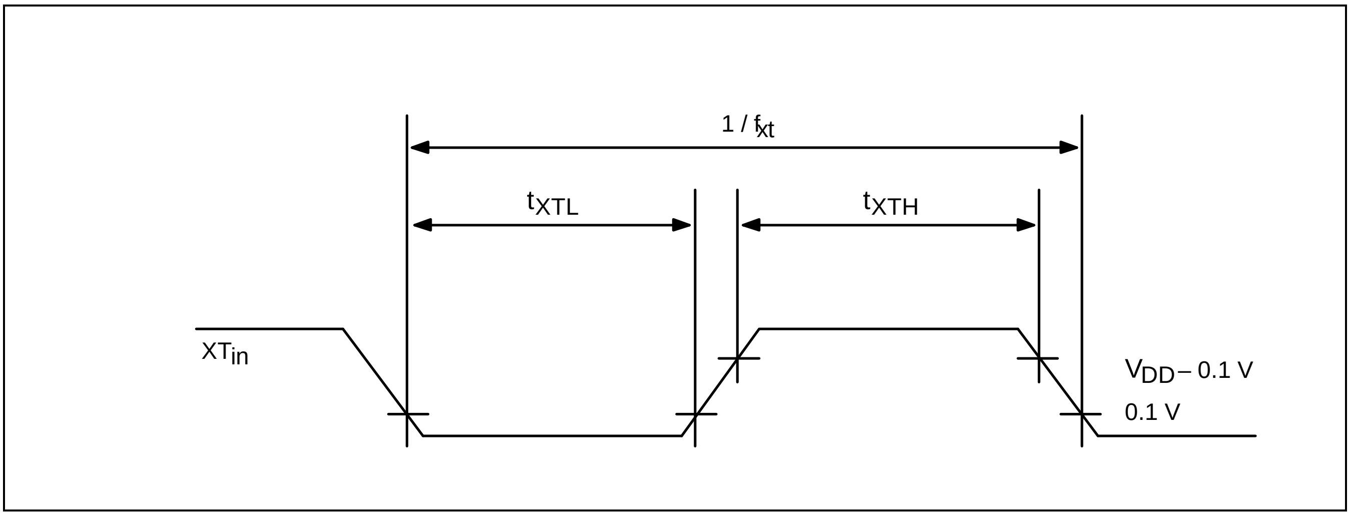


Figure 17-7. Clock Timing Measurement at XT_{in}

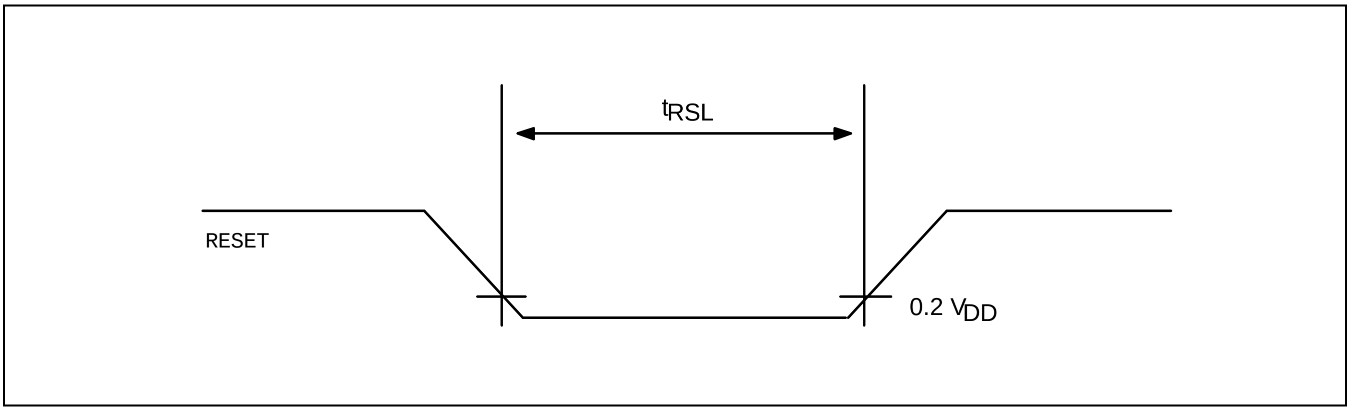


Figure 17-8. Input Timing for RESET Signal

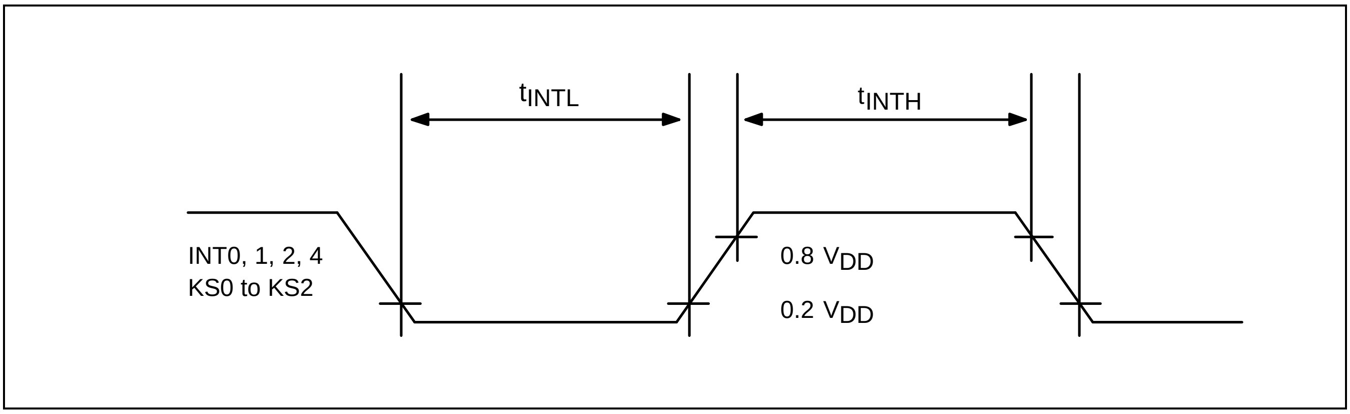


Figure 17-9. Input Timing for External Interrupts and Quasi-Interrupts

18 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 KS57C3204 microcontroller and OTP programmer (Gang) are now available.

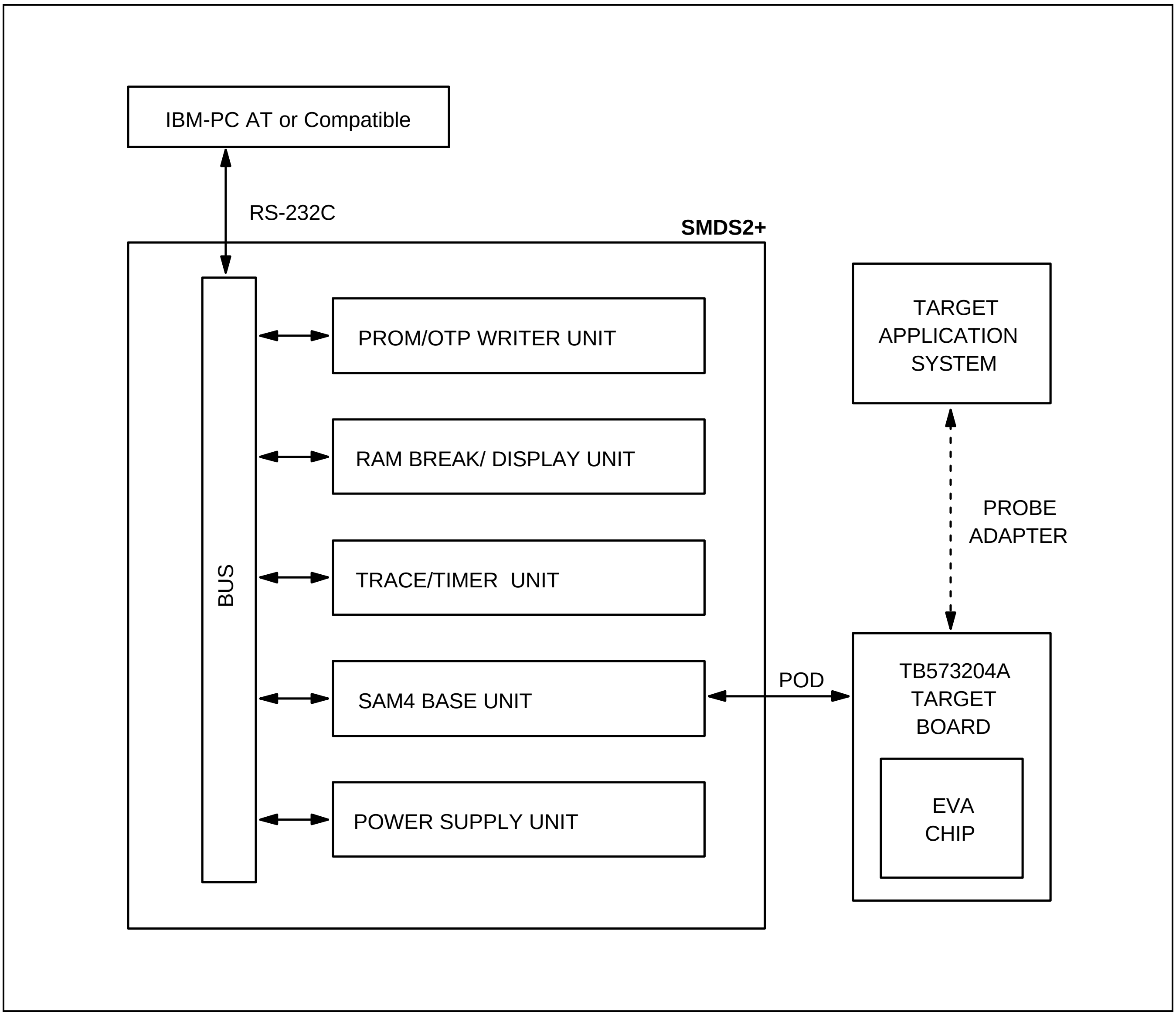


Figure 18-1. SMDS Product Configuration (SMDS2+)

TB573204A TARGET BOARD

The TB573204A target board is used for the KS57C3204/P3204 microcontroller. It is supported by the SMDS2+ development system.

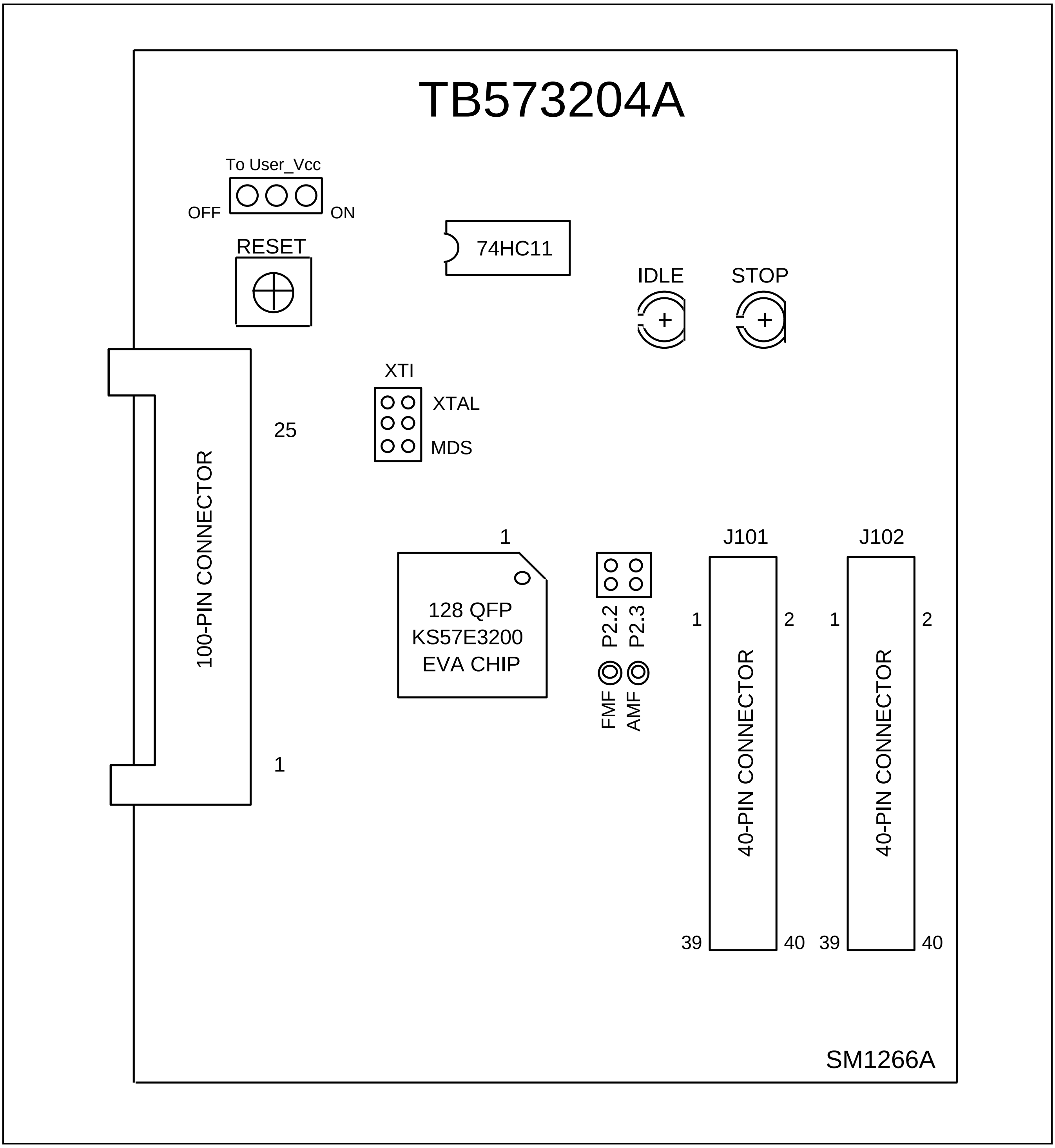


Figure 18-2. TB573204A Target Board Configuration

Table 18-1. Power Selection Settings for TB573204A

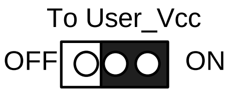
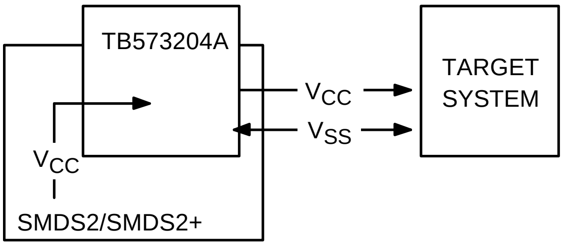
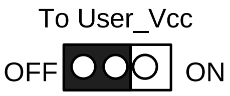
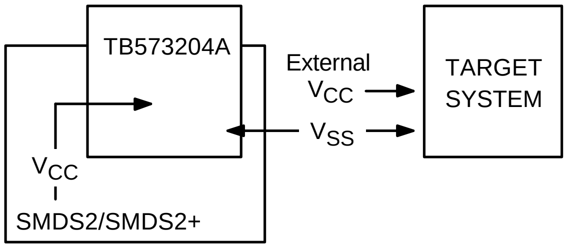
'To User_Vcc' Settings	Operating Mode	Comments
	 TB573204A V_{CC} V_{SS} TARGET SYSTEM V_{CC} SMDS2/SMDS2+	The SMDS2/SMDS2+ supplies V_{CC} to the target board (evaluation chip) and the target system.
	 TB573204A External V_{CC} V_{SS} TARGET SYSTEM V_{CC} SMDS2/SMDS2+	The SMDS2/SMDS2+ supplies V_{CC} only to the target board (evaluation chip). The target system must have its own power supply.

Table 18-2. Pin Selection Settings for TB573204A

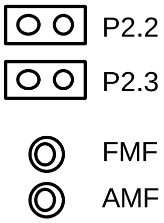
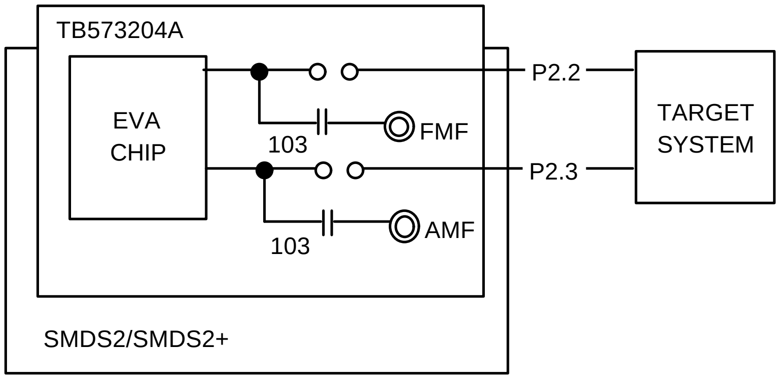
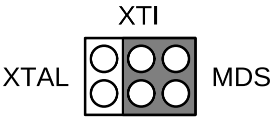
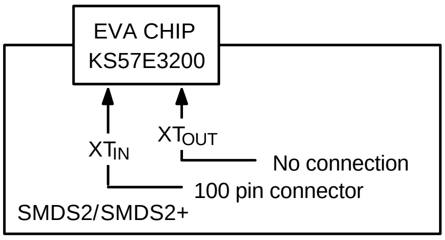
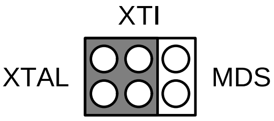
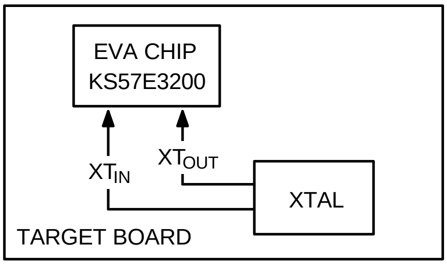
Connection P2.2 and P2.3 to Target System	Operating Mode
	 TB573204A EVA CHIP 103 FMF 103 AMF P2.2 P2.3 TARGET SYSTEM SMDS2/SMDS2+

Table 18-3. Sub-clock Selection Settings for TB573204A

Sub Clock Setting	Operating Mode	Comments
	 EVA CHIP KS57E3200 XTIN XTOUT No connection 100 pin connector SMDS2/SMDS2+	Set the XTI switch to “MDS” when the target board is connected to the SMDS2/SMDS2+.
	 EVA CHIP KS57E3200 XTIN XTOUT XTAL TARGET BOARD	Set the XTI switch to “XTAL” when the target board is used as a standalone unit, and is not connected to the SMDS2/SMDS2+.

IDLE LED

This LED is ON when the evaluation chip (KS57E3200) is in idle mode.

STOP LED

This LED is ON when the evaluation chip (KS57E3200) is in stop mode.

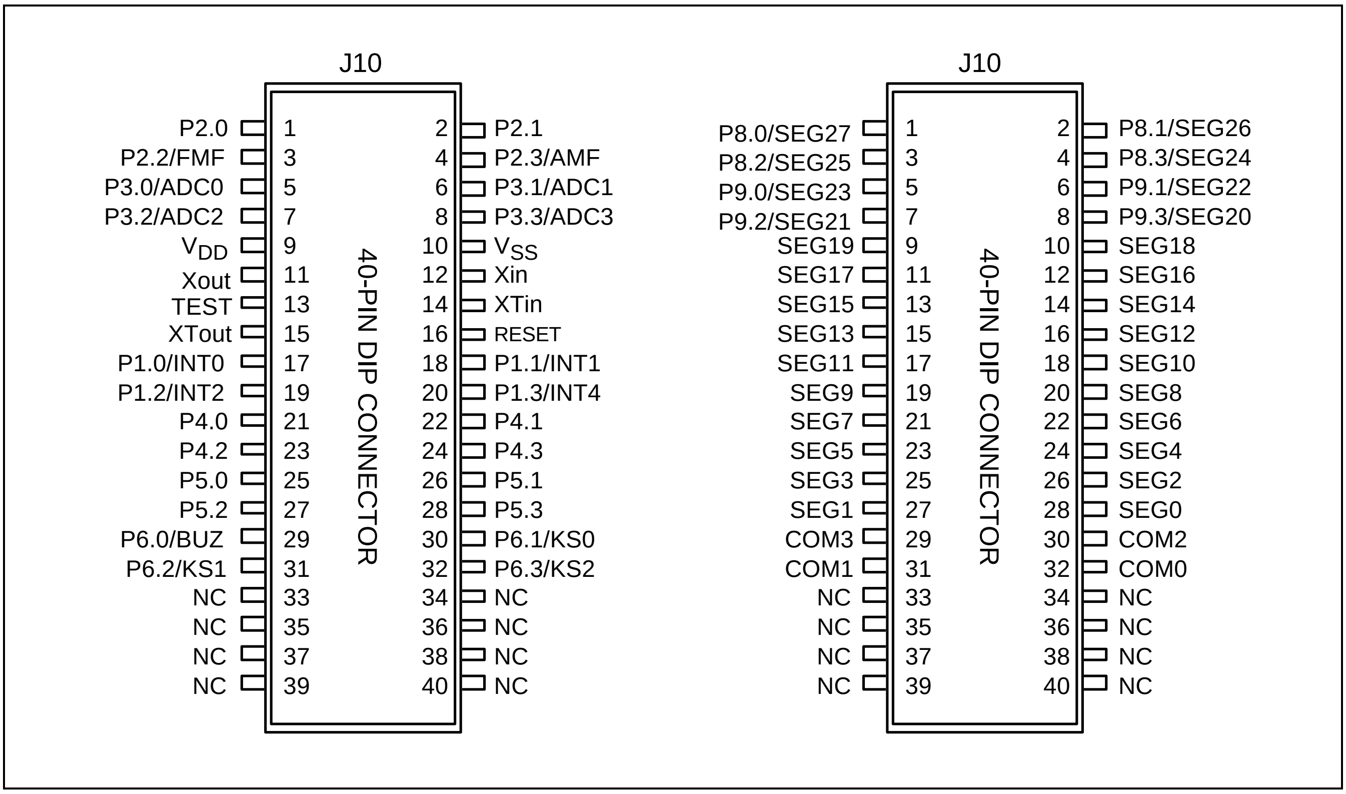


Figure 18-3. 40-Pin Connectors for TB573204A

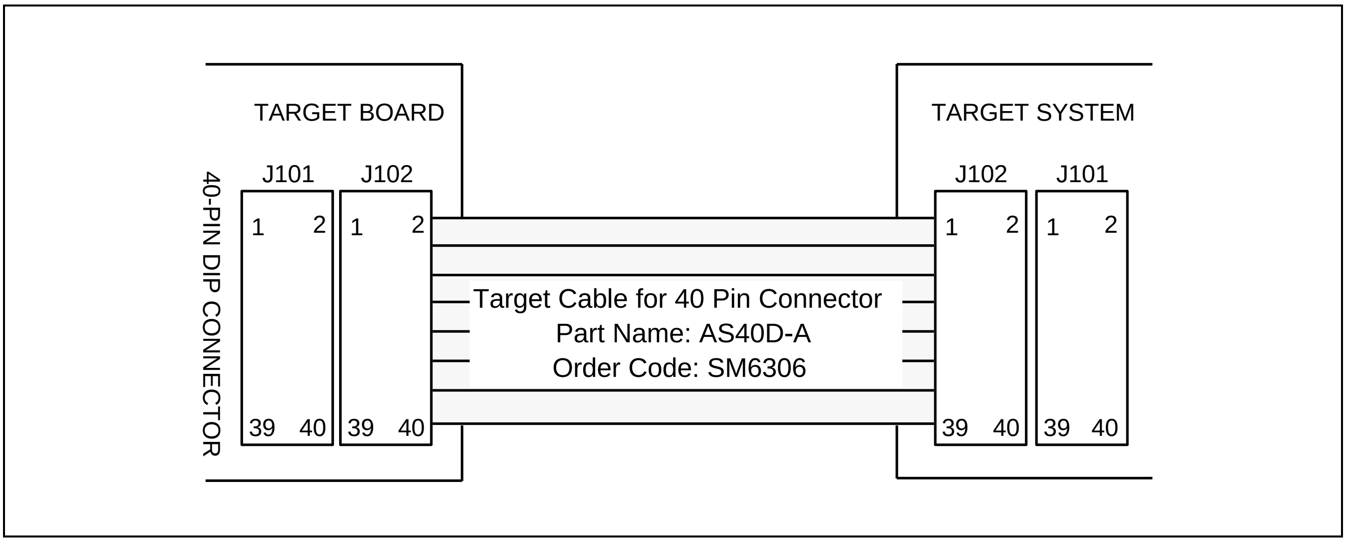


Figure 18-4. TB573204A Adapter Cable for 64-QFP Package (KS57C3204)