1 PRODUCT OVERVIEW

OVERVIEW

The KS57C21408/C21418/P21408 is a SAM47 core-based 4-bit CMOS single-chip microcontroller. It has a timer/counter and LCD drivers.

The KS57P21408 is especially suited for use in data bank, telephone and LCD general purpose.

It is built around the SAM47 core CPU and contains ROM, RAM, 39 I/O lines, programmable timer/counter, buzzer output, enough LCD dot matrix, and segment drive pins.

The KS57C21408/C21418/P21408 can be used for dedicated control functions in a variety of applications, and is especially designed for multi data bank, telephone and LCD game.

OTP

The KS57C21408/C21418 microcontroller is also available in OTP (One Time Programmable) version, KS57P21408. KS57P21408 microcontroller has an on-chip 8 K-byte one-time-programable EPROM instead of masked ROM. The KS57P21408 is comparable to KS57C21408/C21418, both in function and in pin configuration.

FEATURES SUMMARY

Memory

- 8192 × 8 bit program memory
- 5120 × 4 bit data memory in KS57C21408
- 2560 x 4 bit data memory in KS57C21418
- 108 x 5 bit display memory

39 I/O Pins

- Input: 6 pins
- I/O: 17 pins
- Output: maximum 16 pins for 1-bit level output (sharing with segment driver outputs)

8-Bit Basic Timer

- Four internal timer functions

8-Bit Timer/Counter 0

- Programmable 8-bit timer
- External event counter
- Arbitrary clock frequency output
- External clock signal divider

Watch Timer

- Time interval generation: 0,5ms, 3,9ms at 32768Hz
- 4 frequency (2/4/8/16 kHz) outputs to BUZ pin

Interrupts

- Three external vectored interrupts: INT0, INT1, INTP0
- Two internal vectored interrupts: INTB, INTT0
- Two quasi-interrupts: INTW, INT2

Memory Mapped I/O Structure

LCD Display

- 12 characters dot matrix display (5 x 7)
- 12 digit display (8 segments)
- 60 segments and 9 common pins

Power-Down Modes

- Idle mode (only CPU clock stops)
- Stop mode (Main-System clock and CPU clock stops)

Oscillation Sources

- Crystal, ceramic, or External RC for system clock
- Main-system clock frequency: 0.4 MHz - 6MHz
- Sub-system clock frequency: 32,768kHz
- CPU clock divider circuit (by 4,8, or 64)

Instruction Execution Times

- 0.67, 1.33, 10.7 µs at 6MHz
- 0.95, 1.91, 15.3 µs at 4.19 MHz
- 122 µs at 32.768 kHz

Operating Temperature

- -45 °C to 85 °C

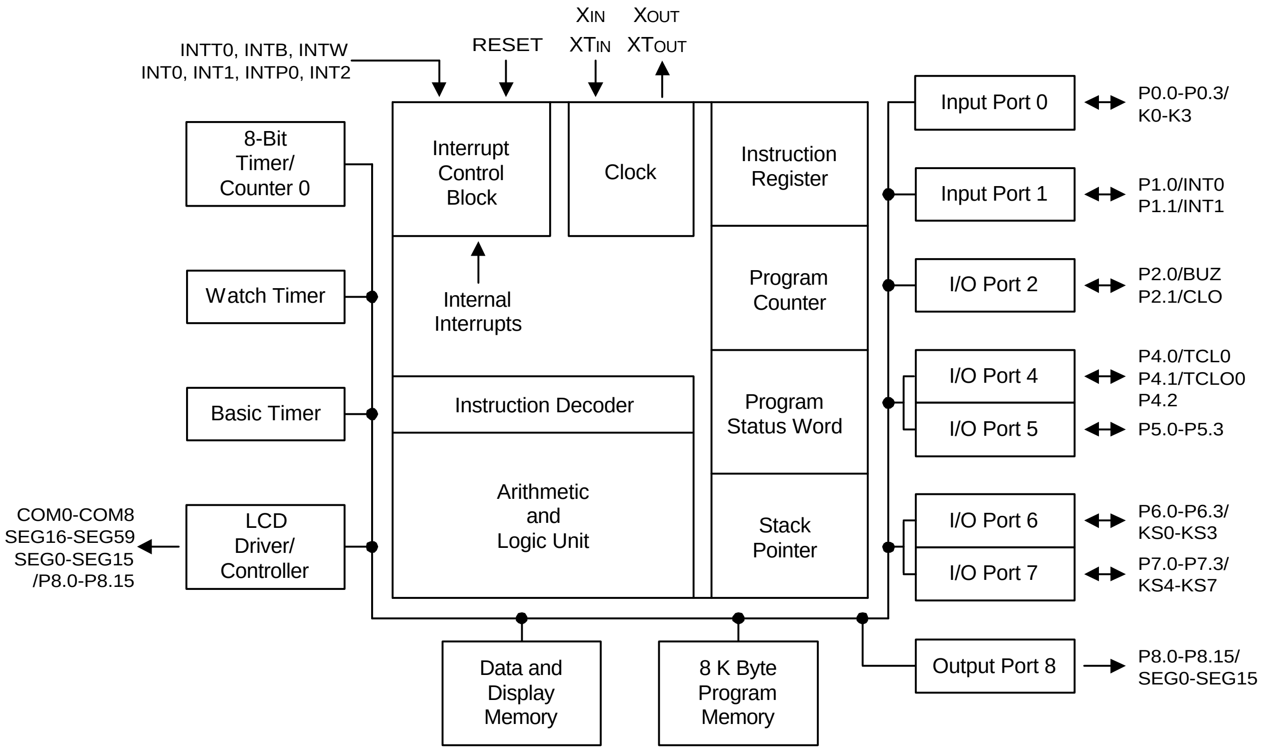
Operating Voltage Range

- 1.8 V to 5.5 V

Package Type

- 100-pin QFP Package

BLOCK DIAGRAM



NOTE:	Data memory:	5120 x 4 bits in KS57C21408
		2560 x 4 bits in KS57C21418
	Display memory:	108 x 5 bits

Figure 1-1. KS57C21408/C21418/P21408 Specified Block Diagram

PIN ASSIGNMENTS

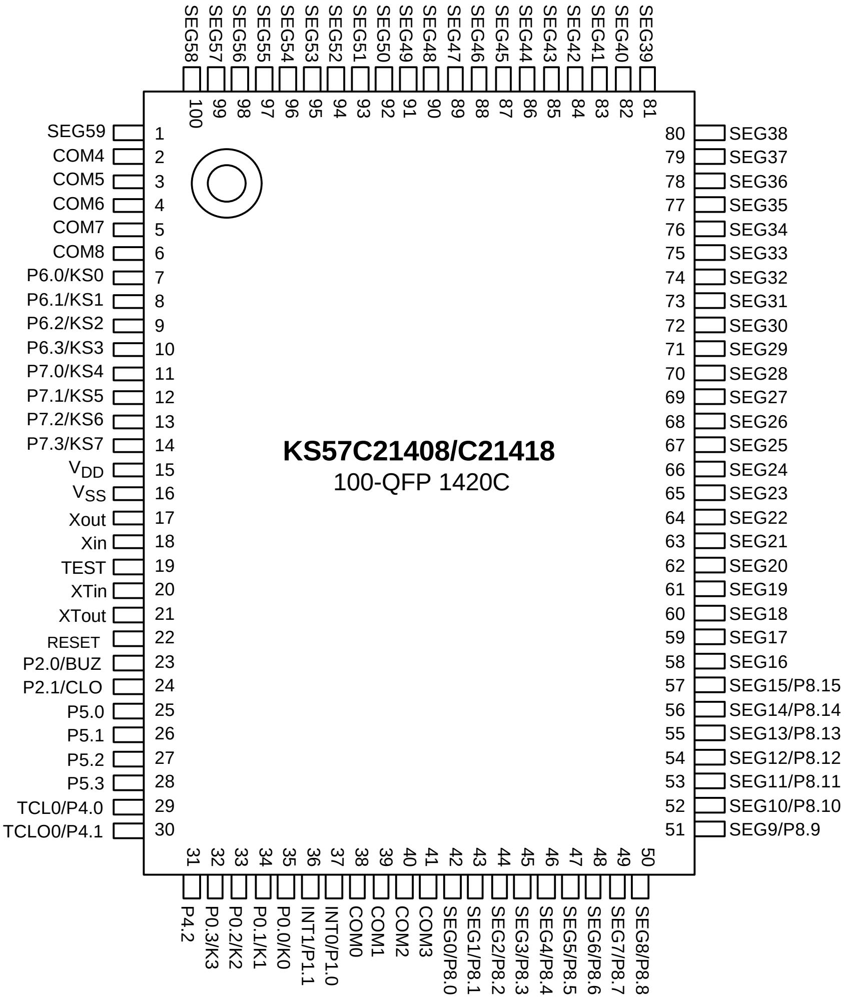


Figure 1-2. KS57C21408/C21418 Pin Assignment Diagram

PIN DESCRIPTIONS

Table 1-1. Pin Descriptions

Pin Name	Pin Type	Description	Circuit Type	Pin Number	Share Pin
P0.0 - P0.3	I	4-bit input port. 1 and 4-bit read, and test are possible. Pull-up registers.	A-1	35-32	K0-K3
P1.0 P1.1	I	2-bit Input port. 1 and 4-bit read, and test are possible, 2-bit pull-up resistors are assignable by software.	A-3	37 36	INT0 INT1
P2.0 P2.1	I/O	2-bit I/O port. 1 and 4-bit read/write, and test are possible. Each individual pin can be specified as input or output. 2-bit pull-up resistors are assignable by software. Pull-up resistors are automatically disabled for output pins.	D	23 24	BUZ CLO
P4.0 P4.1 P4.2 P5.0 - P5.3	I/O	4-bit I/O port. 1, 4, and 8-bit read/write, and test are possible. 4-pin unit can be specified as input or output. 4-bit pull-up resistors are assignable by software. Pull-up resistors are automatically disabled for output pins. Individual pins are software configurable as open-drain or push-pull output.	E E-1 E-1 E-1	29 30 31 25-28	TCL0 TCLO0
P6.0 - P6.3	I/O	4-bit I/O port. 1, 4,and 8-bit read/write, and test are possible. Each individual pin can be specified as input or output. 4-bit pull-up resistors are assignable by software. Pull-up resistors are automatically disabled for output pins.	D-1	7-10	KS0 - KS3
P7.0 - P7.3		4-bit I/O port. 1, 4, and 8-bit read/write, and test are possible. 4-pin unit can be specified as input or output. 4-bit pull-up resistors are assignable by software. Pull-up resistors are automatically disabled for output pins.		11-14	KS4 - KS7
P8.0 - P8.15	O	4-bit controllable output. (Dual function as segment output pins)	H-9	42-57	SEG0 - SEG15
SEG16-SEG59		LCD segment display signal output.	H-10	58-100 ,1	-
SEG0 - SEG15		LCD segment display signal output.	H-9	42-57	P8.0 - P8.15
COM0 - COM8		LCD common signal output.	H-11	38-41 2-6	-
INT0 - INT1	I	External interrupts. The triggering edge for INT0, and INT1 is selectable		37-36	P1.0 -P1.1
KS0 - KS7	I/O	Quasi-interrupt input for falling edge detection.		7-14	P6.0 - P7.3
K0 - K3	I	Vector interrupt input K0 - K3: falling edge detection		35-32	P0.0 - P0.3

Table 1-1. Pin Descriptions (Continued)

Pin Name	Pin Type	Description	Circuit Type	Pin Num.	Share Pin
BUZ	I/O	2,4,8 kHz or 16kHz frequency output for buzzer signal.	-	23	P2.0
CLO		Clock output	-	24	P2.1
X_{in}, X_{out}	-	Crystal, ceramic or RC oscillator pins for main system clock.	-	18, 17	-
XT_{in}, XT_{out}	-	Crystal oscillator pins for sub-system clock.	-	20, 21	-
TCL0	I/O	External clock input for Timer/Counter 0	-	29	P4.0
TCLO0	I/O	Timer/Counter 0 clock output	-	30	P4.1
RESET	I	Reset input (active low).	B	22	-
V_{DD}	-	Power supply.	-	15	-
V_{SS}	-	Ground.	-	16	-
TEST	I	Test input: it must be connected to V_{SS}	-	19	-

PIN CIRCUIT DIAGRAMS

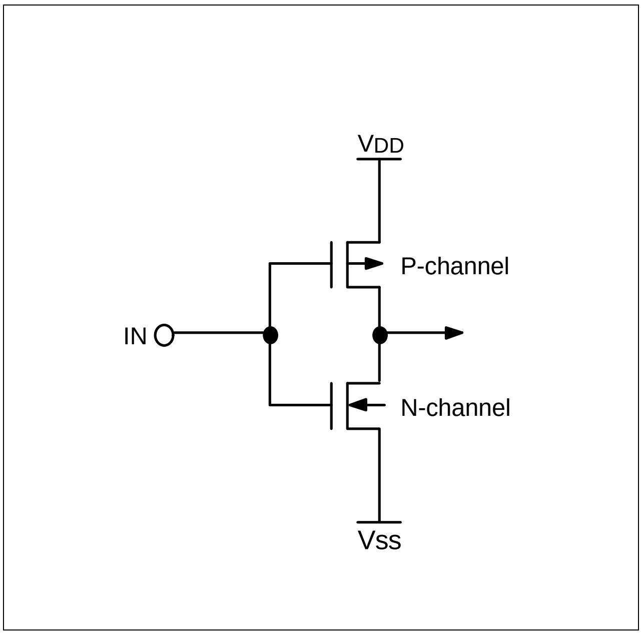


Figure 1-3. Pin Circuit Type A

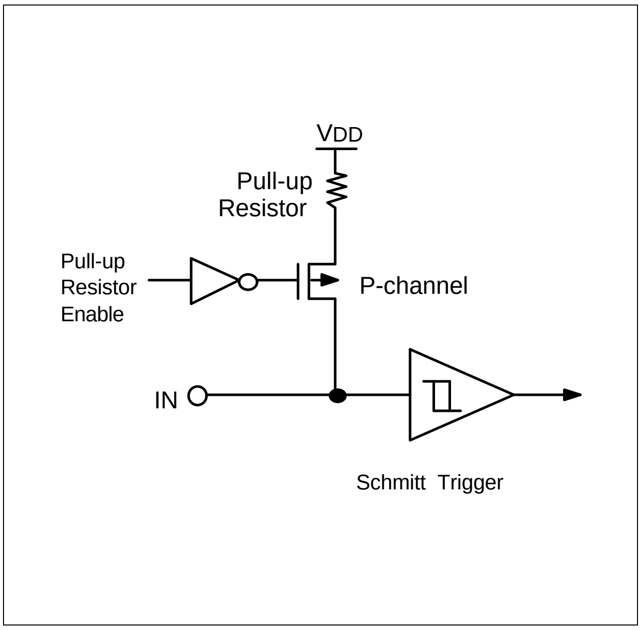


Figure 1-5. Pin Circuit Type A-3

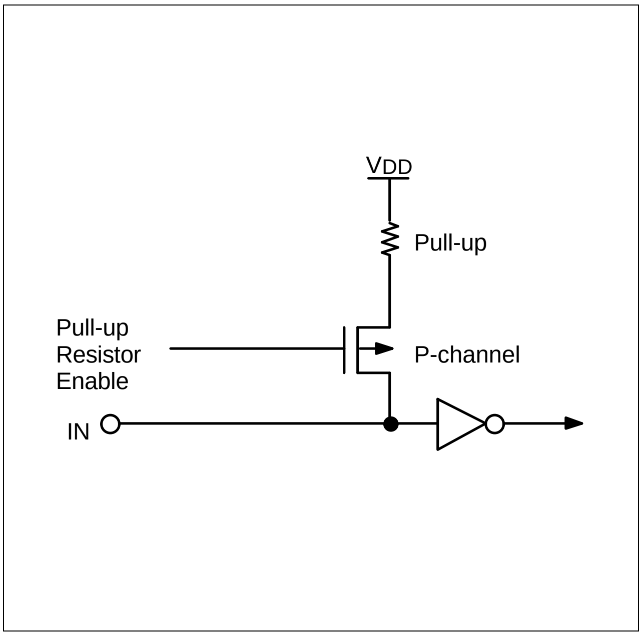


Figure 1-4. Pin Circuit Type A-1

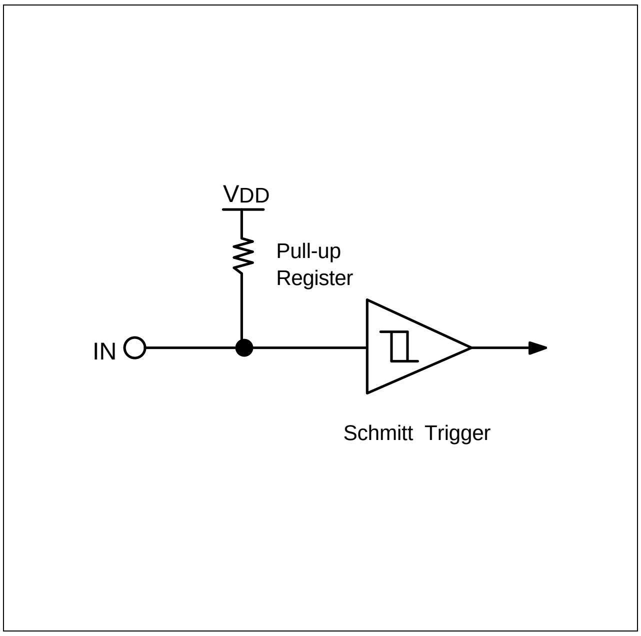


Figure 1-6. Pin Circuit Type B

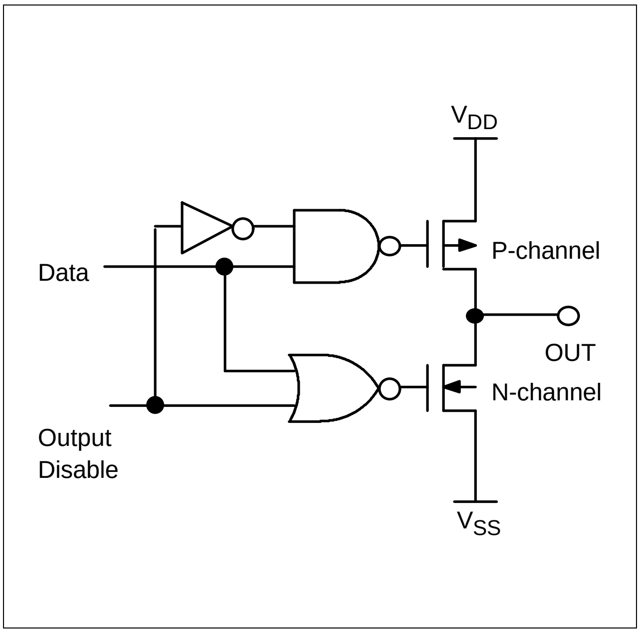


Figure 1-7. Pin Circuit Type C

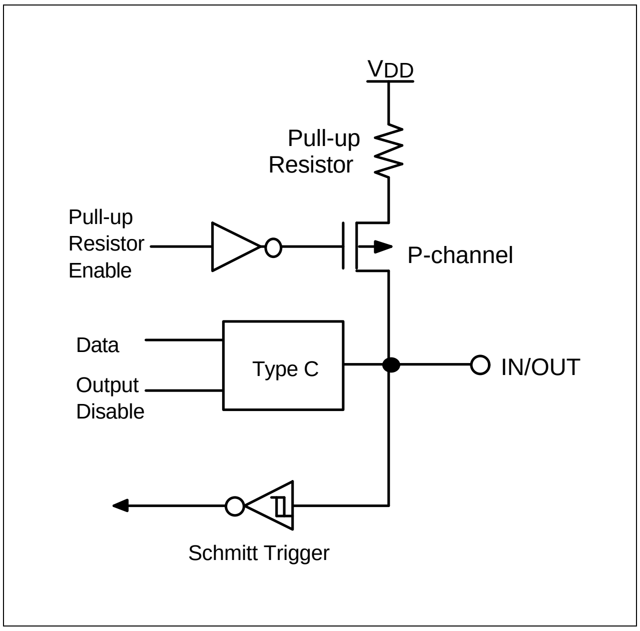


Figure 1-9. Pin Circuit Type D-1

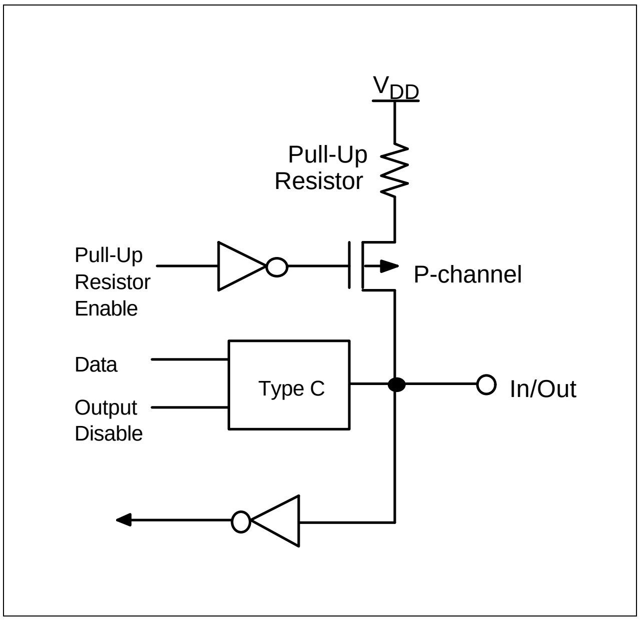


Figure 1-8. Pin Circuit Type D

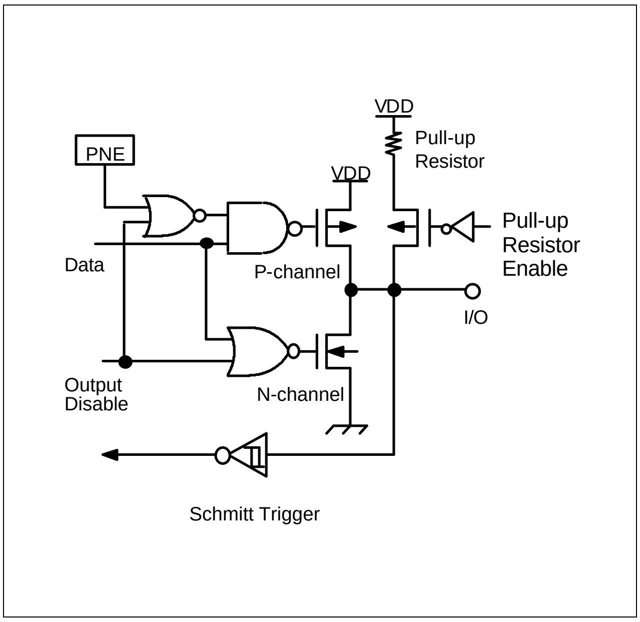


Figure 1-10. Pin Circuit Type E

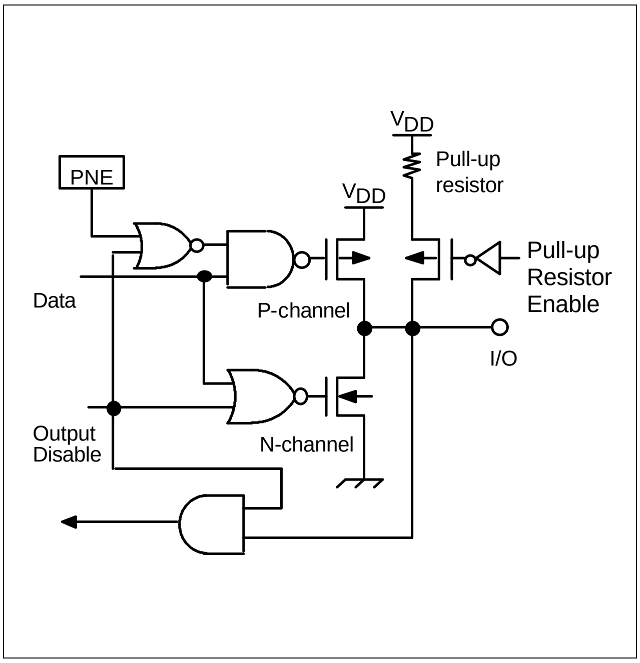


Figure 1-11. Pin Circuit Type E-1

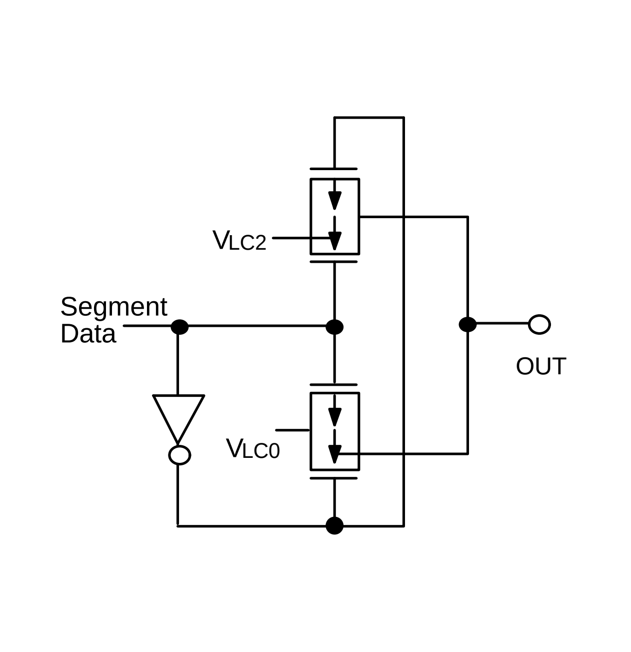


Figure 1-13. Pin Circuit Type H-10

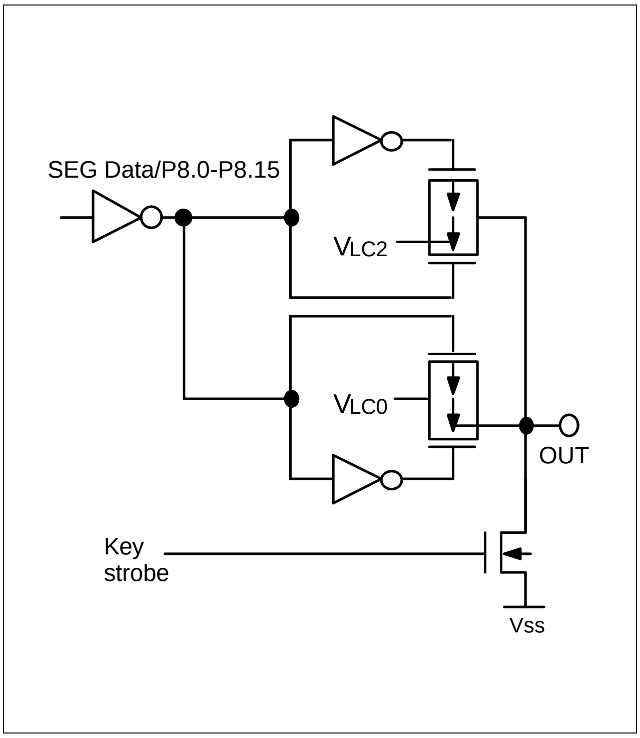


Figure 1-12. Pin Circuit Type H-9

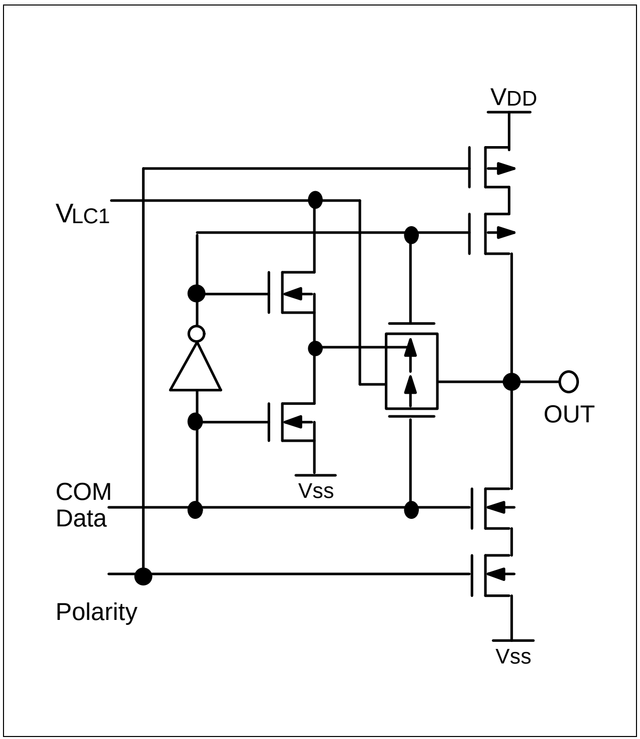


Figure 1-14. Pin Circuit Type H-11

NOTES

2 ADDRESS SPACES

PROGRAM MEMORY (ROM)

OVERVIEW

ROM maps for KS57C21408/C21418 devices are mask programmable at the factory. In its standard configuration, the device's 8,192 bytes 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

— 8,064-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 8,064-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 one 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-1FFFH	8,064

GENERAL-PURPOSE MEMORY AREAS

The 20-byte area at ROM locations 000CH-001FH and the 8,064-byte area at ROM locations 0080H-1FFFH 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. 16-byte vector addresses are organized as follows:

EMB	ERB	0	PC12	PC11	PC10	PC9	PC8
PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

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

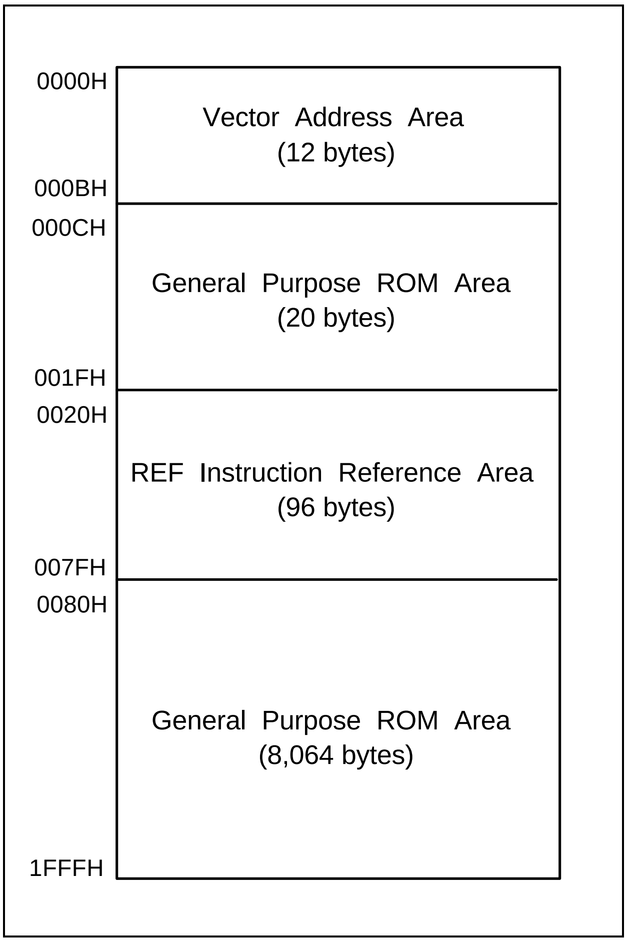


Figure 2-1. ROM Address Structure

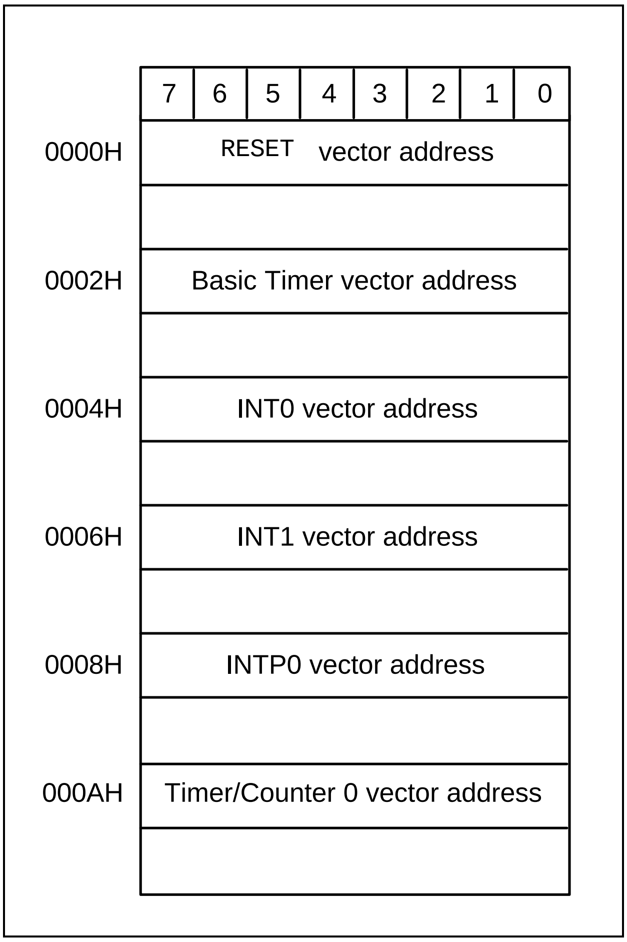


Figure 2-2. Vector Address Map

☞ PROGRAMMING TIP — Defining Vectored Interrupts

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

1. When all vector interrupts are used:

```
          ORG       0000H
;
          VENT0     1,0,RESET          ;  EMB ← 1, ERB ← 0; Jump to RESET address by RESET
          VENT1     0,0,INTB           ;  EMB ← 0, ERB ← 0; Jump to INTB address by INTB
          VENT2     0,0,INT0           ;  EMB ← 0, ERB ← 0; Jump to INT0 address by INT0
          VENT3     0,0,INT1           ;  EMB ← 0, ERB ← 0; Jump to INT1 address by INT1
          VENT4     0,0,INTP0          ;  EMB ← 0, ERB ← 0; Jump to INTP0 address by INTP0
          VENT5     0,0,INTT0          ;  EMB ← 0, ERB ← 0; Jump to INTT0 address by INTT0
```

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

```
          ORG       0000H
;
          VENT0     1,0,RESET          ;  EMB ← 1, ERB ← 0; Jump to RESET address by RESET
          VENT1     0,0,INTB           ;  EMB ← 0, ERB ← 0; Jump to INTB address by INTB
          ORG       0006H              ;  INT0 interrupt not used
          VENT3     0,0,INT1           ;  EMB ← 0, ERB ← 0; Jump to INT1 address by INT1
          VENT4     0,0,INTP0          ;  EMB ← 0, ERB ← 0; Jump to INTP0 address by INTP0
```

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 in Example 2, a CPU malfunction will occur:

```
          ORG       0000H
;
          VENT0     1,0,RESET          ;  EMB ← 1, ERB ← 0; Jump to RESET address by RESET
          VENT1     0,0,INTB           ;  EMB ← 0, ERB ← 0; Jump to INTB address by INTB
          VENT3     0,0,INT1           ;  EMB ← 0, ERB ← 0; Jump to INT0 address by INT0
          VENT4     0,0,INTP0          ;  EMB ← 0, ERB ← 0; Jump to INTP0 address by INT1
          VENT5     0,0,INTT0          ;  EMB ← 0, ERB ← 0; Jump to INTT0 address by INTP0
```

In this example, when an INTP0 interrupt is generated, the corresponding vector area is not VENT4 INTP0, but VENT5 INTT0. This causes an INTP0 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 1-byte instructions, one 2-byte instruction, or one 3-byte instruction such as a JP (jump) or CALL. The starting address of the instruction you are referencing must always be an even number. To reference a JP or CALL instruction, it must be written to the reference area in a two-byte format: for JP, this format is TJP; for CALL, it is TCALL. In summary, there are three ways to the REF instruction:

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

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

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

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

☞ PROGRAMMING TIP — Using the REF Look-Up Table

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

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

          •
          •
```

DATA MEMORY (RAM)

OVERVIEW

In its standard configuration, the data memories have four areas:

— 32 × 4-bit working register area

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

— 20 pages with 256 x 4-bit in bank1 of KS57C21408

- 19 pages for general purpose area (00H-12H page)
- 1 page for LCD Display data memory (13H page)

— 10 page with 256 x 4-bit in bank 1 of KS57C21418

- 9 pages for general purpose area (00H-08H page)
- 1 page for LCD Display data memory (13H page)

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

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

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

Bank 1 Page Selection Register (PASR)

PASR is a 5-bit write -only register for selecting the page of bank1 ,and is mapped to the RAM address FA0H. It should be written by a 8-bit RAM control instruction only and the MSB 3 bits should be "0". PASR retains the previous value as long as change is not required, and the reset value is 0. Therefore, when it returns to the Bank 1 from other bank (Bank 0 or Bank 15) without changing the contents of PASR, the previously specified Bank 1 page is selected . The PASR must not be changed in the interrupt service routine because it's value cannot be recovered as the original value when the routine is finished.

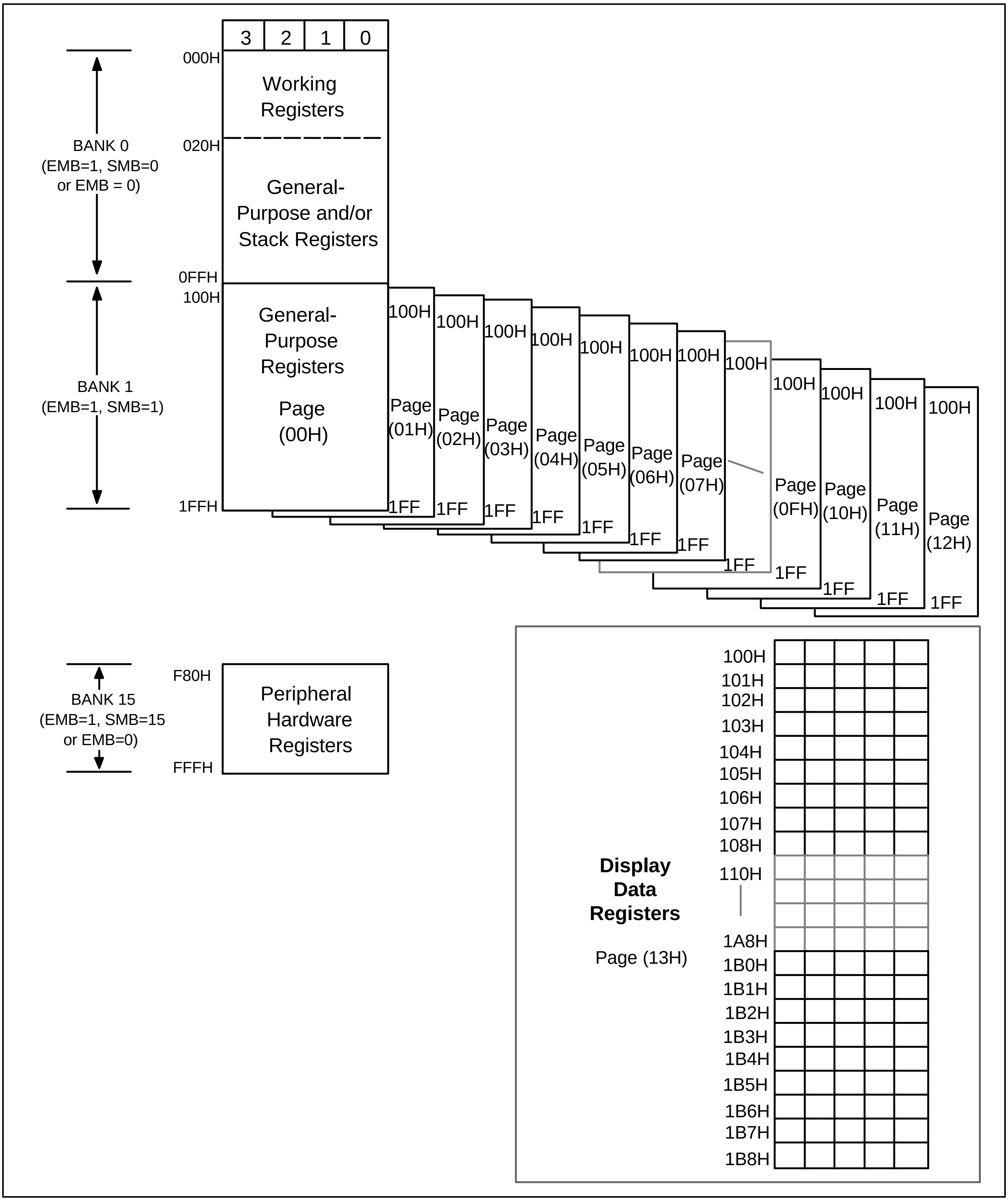


Figure 2-3. KS57C21408 Data Memory (RAM) Map

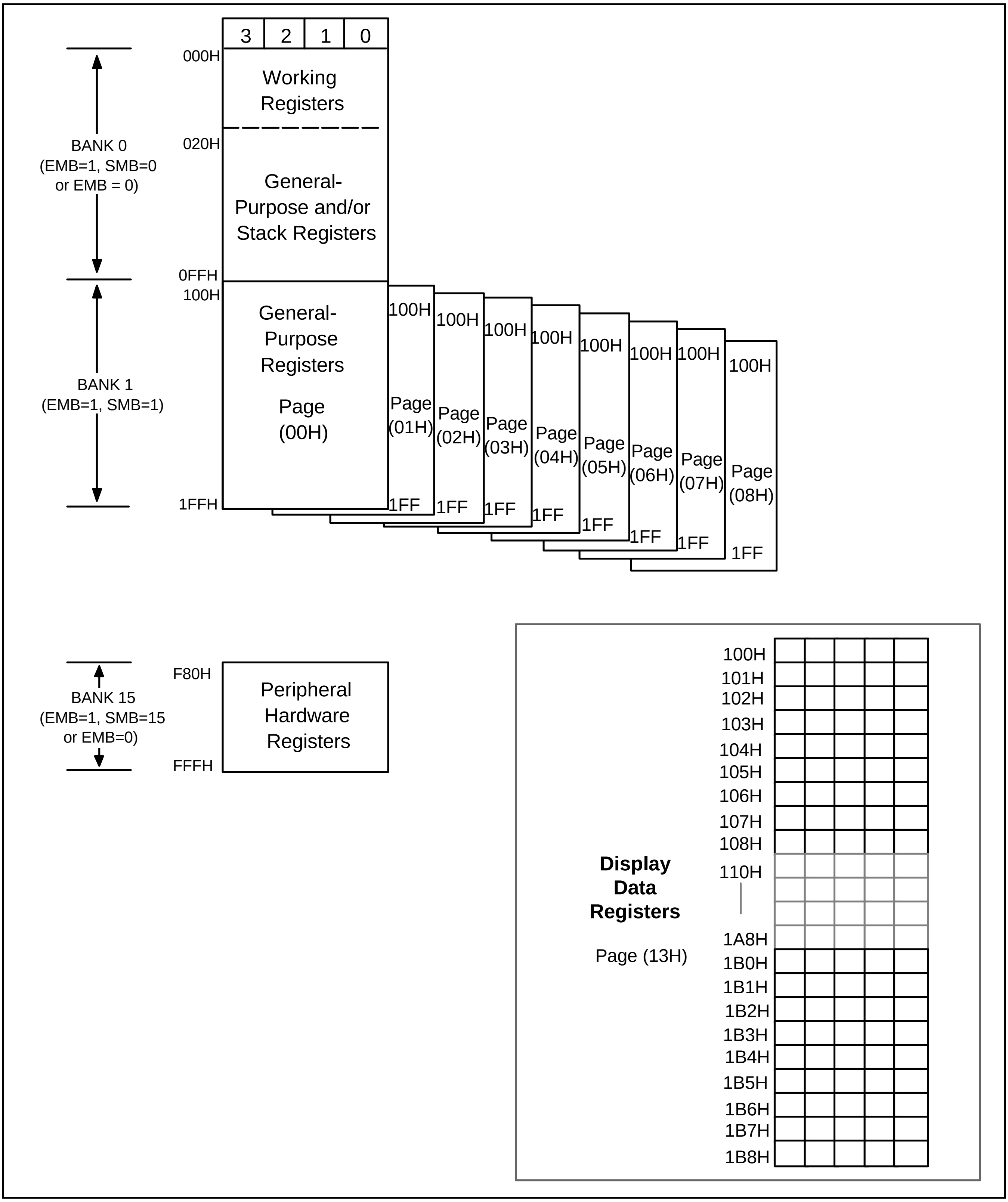


Figure 2-4. KS57C21418 Data Memory (RAM) Map

Memory Banks 0, 1, and 15

Bank 0 (000H-0FFH) The lowest 32 nibbles of bank 0 (000H-01FH) are used as working registers; the next 224 nibbles (020H-0FFH) can be used both as stack area and as general-purpose data memory. Use the stack area for implementing subroutine calls and returns, and for interrupt processing.

Bank 1 (100H-1FFH) Bank 1 has the data memory of 20 pages in KS57C21408, 10 pages in KS57C21418, the 00H-12H pages in KS57C21408, 00H-08H pages in KS57C21418 for general purpose data memory are comprised of 256 x 4-bits, and the 13th page in KS57C21408, and KS57C21418 for LCD display data memory consists of 108 x 5-bits. The KS57C21408, and KS57C21418 use specially a Bank 1 page selection register (PASR) for selecting one of these 20 pages in KS57C21408/10 pages in KS57C21418.

Bank 15 (F80H-FFFH) The microcontroller uses bank 15 for memory-mapped peripheral I/O. Fixed RAM locations for each peripheral hardware address are mapped into this area.

Data Memory Addressing Modes

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

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

Working Registers

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

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

LCD Data Register Area

Bit values for LCD segment data are stored in data memory bank 1 (13H page). Register locations in this area that are not used to store LCD data can be assigned to general-purpose use.

Table 2-2. Data Memory Organization and Addressing

Addresses	Register Areas	Bank	EMB Value	SMB Value
000H-01FH	Working registers	0	0, 1	0
020H-0FFH	Stack and general-purpose registers			
100H11FFH	General-purpose registers (KS57C21408: 00H-12H pages) General-purpose registers (KS57C21418: 00H-08H pages) LCD display data memory (the 13th page)	1	1	1
F80H-FFFH	I/O-mapped hardware registers	15	0, 1	15

NOTE: LCD data register is 13H page in data memory Bank 1 for the KS57C21408, and KS57C21418.

☞ PROGRAMMING TIP — Clearing Data Memory Bank 0 ,and the page 0 in Bank 1

Clear bank 0 of the data memory area, and the page 0 of the data memory area in Bank 1

```
          SMB     15
          LD      EA, #00H
          LD      PASR, EA

RAMCLR    SMB     1                   ;   page 0 in Bank 1 clear
          LD      HL,#00H
          LD      A,#0H
RMCL1     LD      @HL,A
          INCS    HL
          JR      RMCL1
;
          SMB     0                   ;   Bank 0 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 unit, 4-bit unit, or using paired registers, as 8-bit unit.

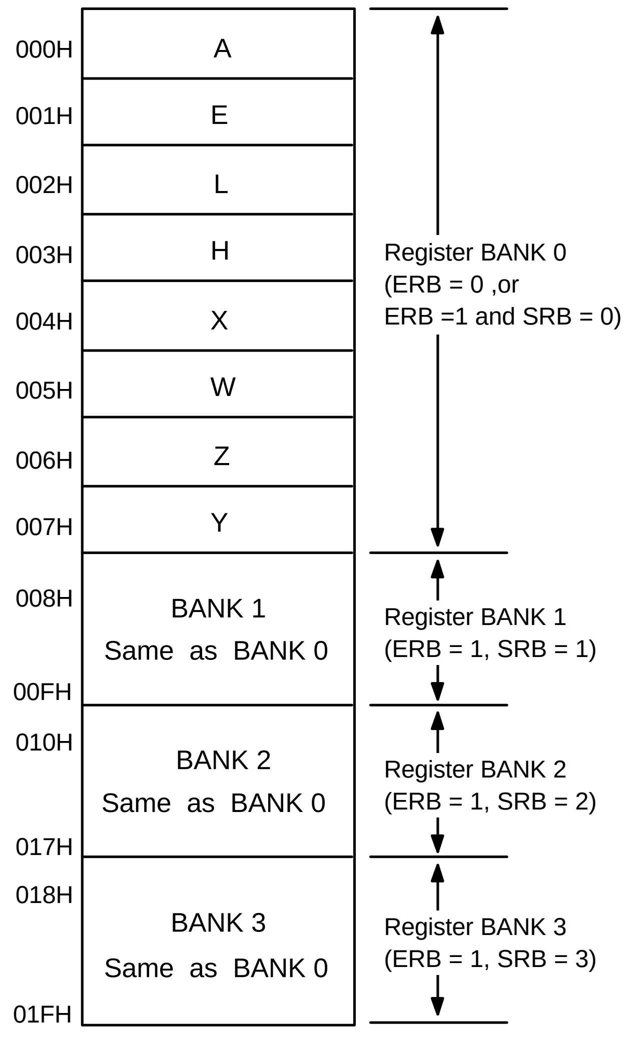


Figure 2-5. Working Register Map

Working Register Banks

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

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

Table 2-3. Working Register Organization and Addressing

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

NOTE: 'x' means don't care.

Paired Working Registers

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

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

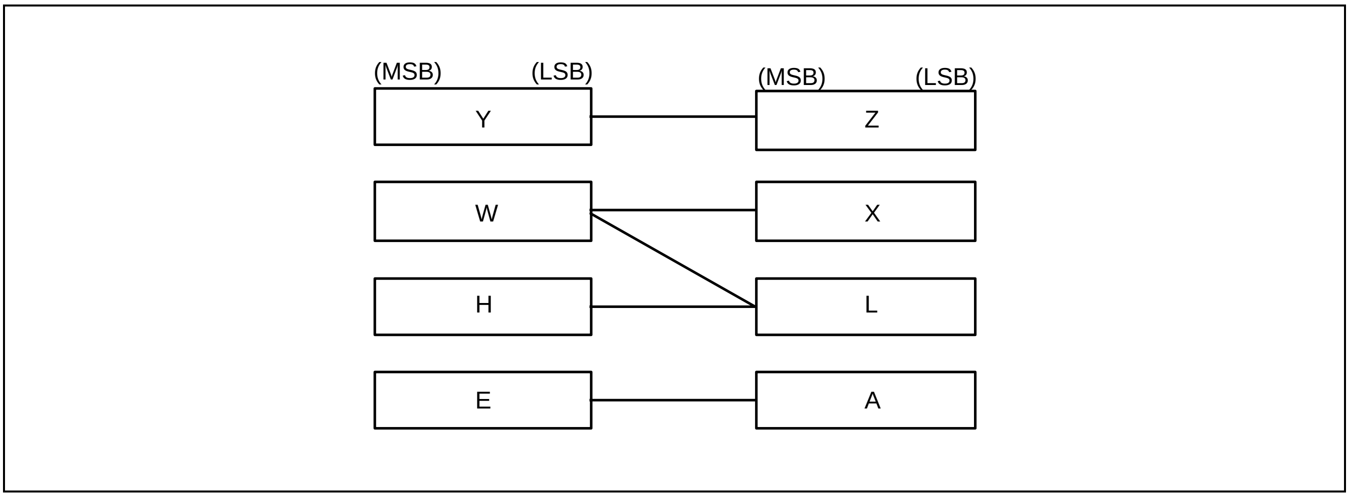


Figure 2-6. Register Pair Configuration

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.

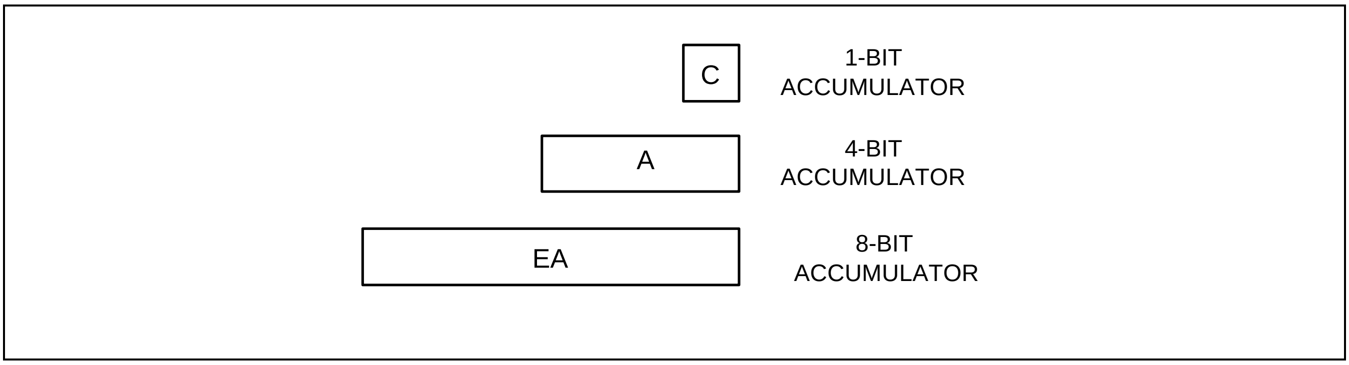


Figure 2-7. 1-Bit, 4-Bit, and 8-Bit Accumulator

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 been executed successfully, you can restore the register contents from the stack to working memory by using the POP instruction.

☞ PROGRAMMING TIP — Selecting the Working Register Area

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

1. When ERB = "0":

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

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

2. When ERB = "1":

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

STACK OPERATIONS

STACK POINTER (SP)

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

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

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

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

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

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

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

NOTE

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

☞ PROGRAMMING TIP — Initializing the Stack Pointer

To initialize the stack pointer (SP):

1. When EMB = "1":

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

2. When EMB = "0":

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

PUSH OPERATIONS

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

PUSH Instructions

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

CALL Instructions

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

Interrupt Routines

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

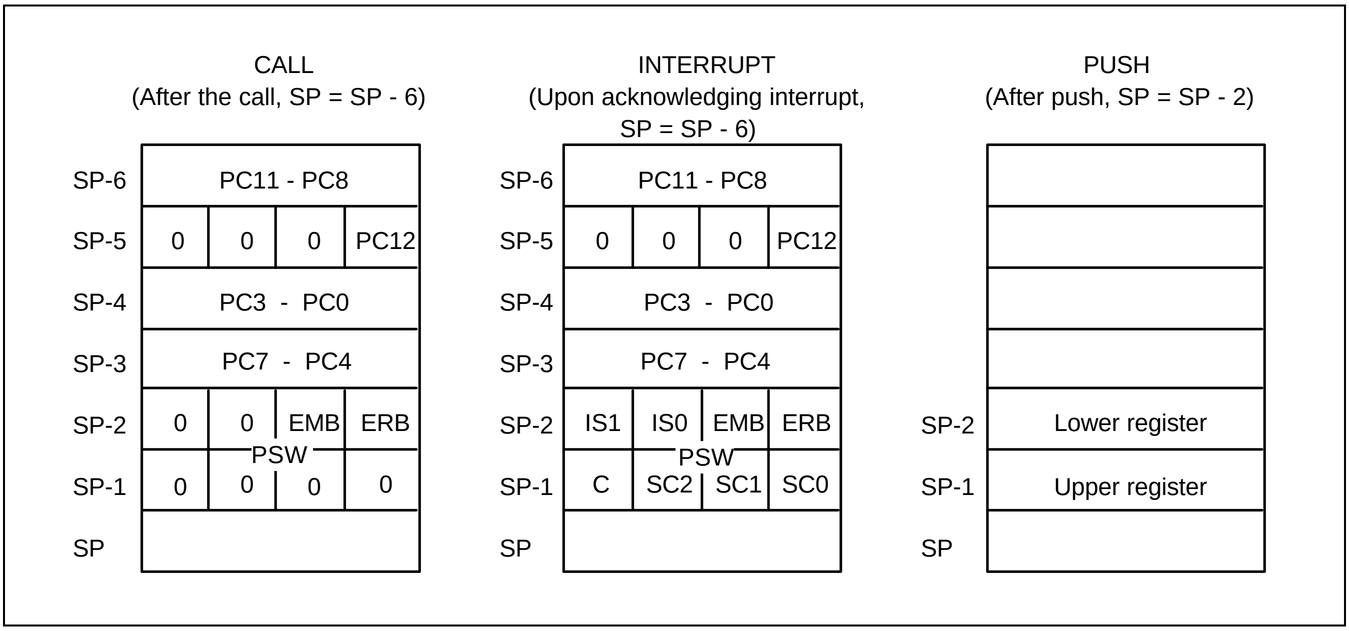


Figure 2-8. Push-Type Stack Operations

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 been 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 been 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 been executed, the SP is incremented *by six* and points to the next free stack location.

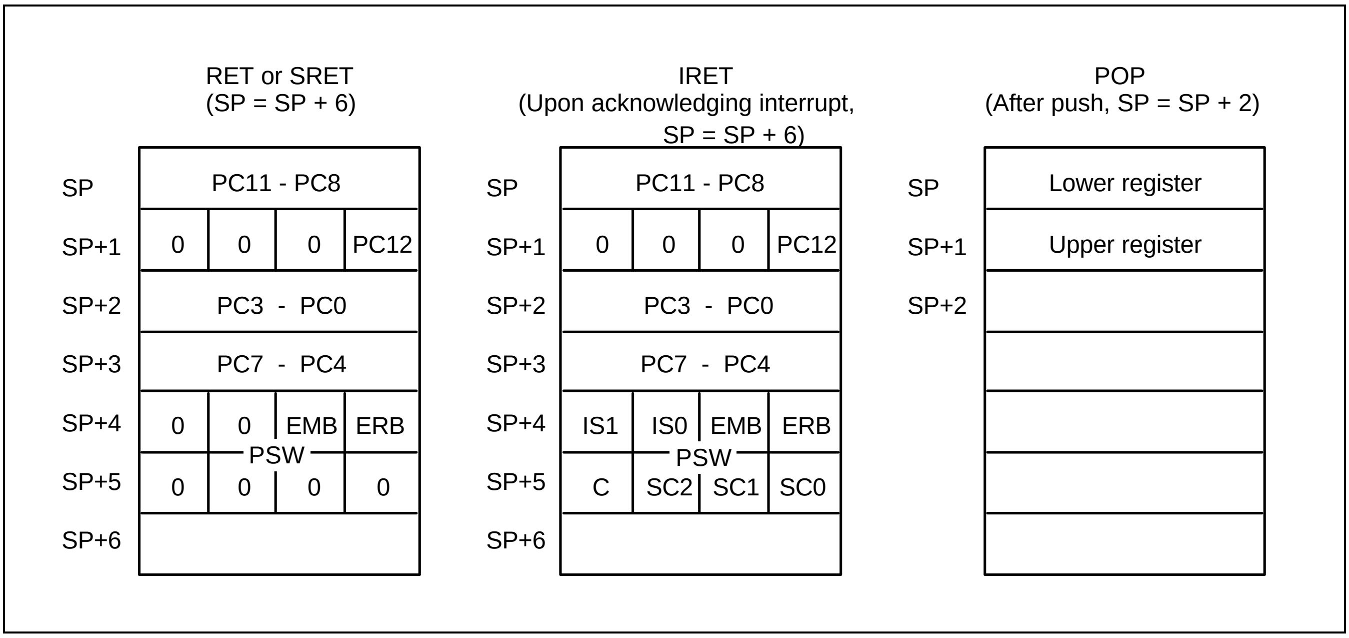


Figure 2-9. Pop-Type Stack Operations

BIT SEQUENTIAL CARRIER (BSC)

The BSC can be manipulated using 1-, 4-, and 8-bit RAM control instructions. 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 decrementing 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 P2.0 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     P2.0,C          ;  P2.0 ← C
        INCS    L
        JR      AGN
        RET
```

PROGRAM COUNTER (PC)

A 13-bit program counter (PC) stores addresses for instruction fetches during program execution. Whenever a reset operation or an interrupt occurs, bits PC12 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:

FB0H	IS1	IS0	EMB	ERB
FB1H	C	SC2	SC1	SC0

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

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

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

Table 2-5. Program Status Word Bit Descriptions

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

INTERRUPT STATUS FLAGS (IS0, IS1)

PSW bits IS0 and IS1 contain the current interrupt execution status values. 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.

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.

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

☞ 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
        •
        •
        •
        IRET
```

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.

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

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

EMB flag settings for memory bank selection:

1. When EMB = "0":

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

2. When EMB = "1":

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

ERB FLAG (ERB)

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

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

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

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

ERB flag settings for register bank selection:

1. When ERB = "0":

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

2. When ERB = "1":

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

SKIP CONDITION FLAGS (SC2, SC1, SC0)

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

CARRY FLAG (C)

The carry flag is used to save the result of an overflow or borrow when executing arithmetic instructions involving a carry (ADC, SBC). The carry flag can also be used as a 1-bit accumulator for performing Boolean operations involving bit-addressed data memory.
If an overflow or borrow condition occurs when executing arithmetic instructions with carry (ADC, SBC), the carry flag is set to "1". Otherwise, its value is "0". When a RESET occurs, the current value of the carry flag is retained during power-down mode, but when normal operating mode resumes, its value is undefined.
The carry flag can be directly manipulated by predefined set of 1-bit read/write instructions, independent of other bits in the PSW. Only the ADC and SBC instructions, and the instructions listed in Table 2-7, 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 P2.0 and output the result to P5.0:

```
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,P2.0               ;  C ← C AND P2.0
LDB       P5.0,C               ;  Output result from carry flag to P5.0
```

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

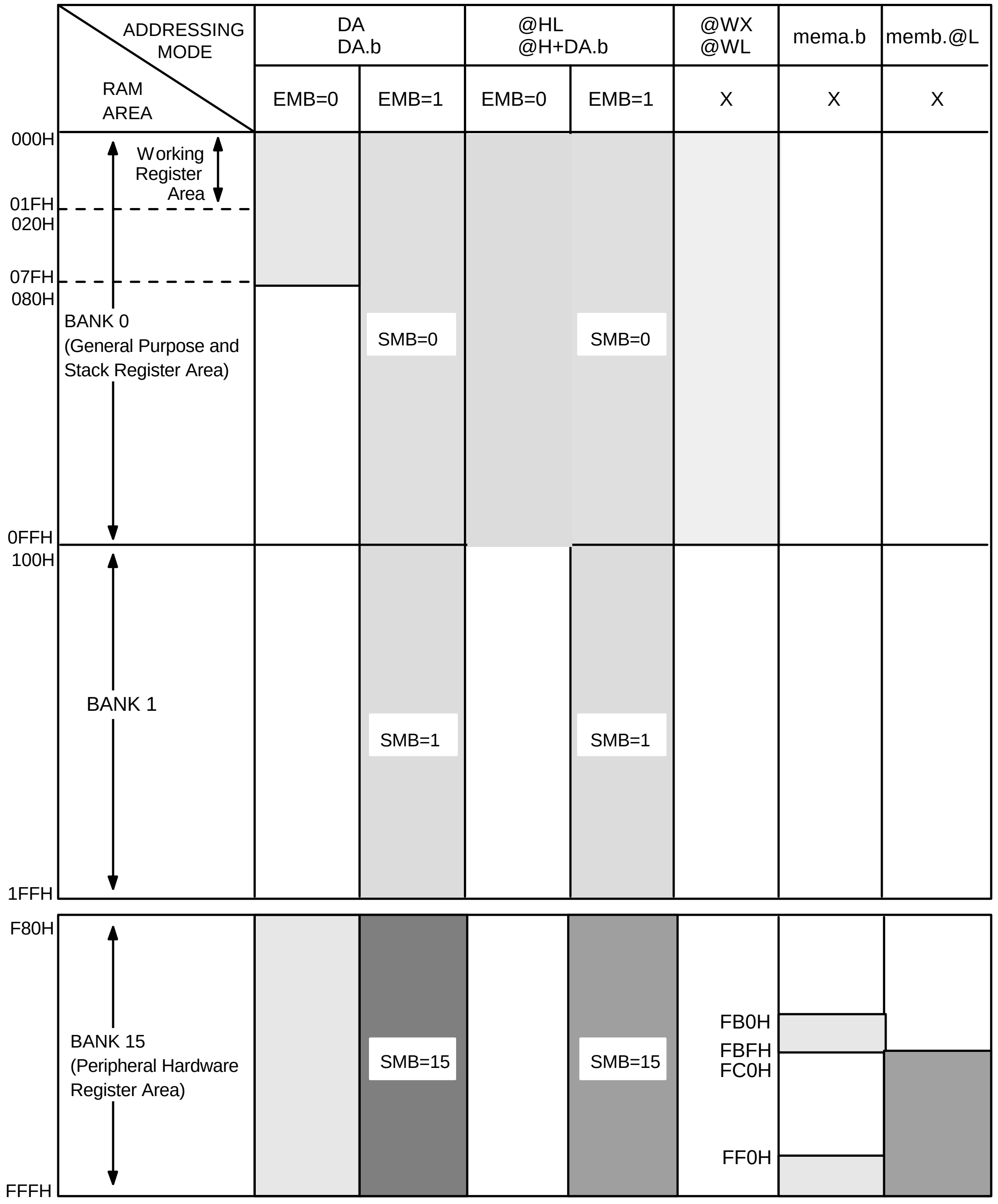
Here are a few guidelines to keep in mind regarding data memory addressing:

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

— Display RAM locations in bank 1 are 8-bit addressable only.

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

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



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

Figure 3-1. RAM Address Structure

EMB AND ERB INITIALIZATION VALUES

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

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

☞ PROGRAMMING TIP — Initializing the EMB and ERB Flags

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

```
ORG         0000H            ;  ROM address assignment
VENT0       1,0, RESET       ;  EMB ← 1, ERB ← 0, 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
VENT4       0,1,INTP0        ;  EMB ← 0, ERB ← 1, branch INTP0
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, 1, or 15) using 1-, 4-, or 8-bit instructions. You can use both direct and indirect addressing modes. The addressable RAM areas when EMB = "1" are as follows:

If SMB = 0,	000H-0FFH
If SMB = 1,	100H-1FFH
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 LD EA,SP
FB0H-FBFH FF0H-FFFH	1-bit direct addressing	PSW, SCMOD, IEx, IRQx, I/O	BITS EMB BITR IE2
FC0H-FFFH	1-bit indirect addressing using the L register	I/O	BTST F3H.@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.

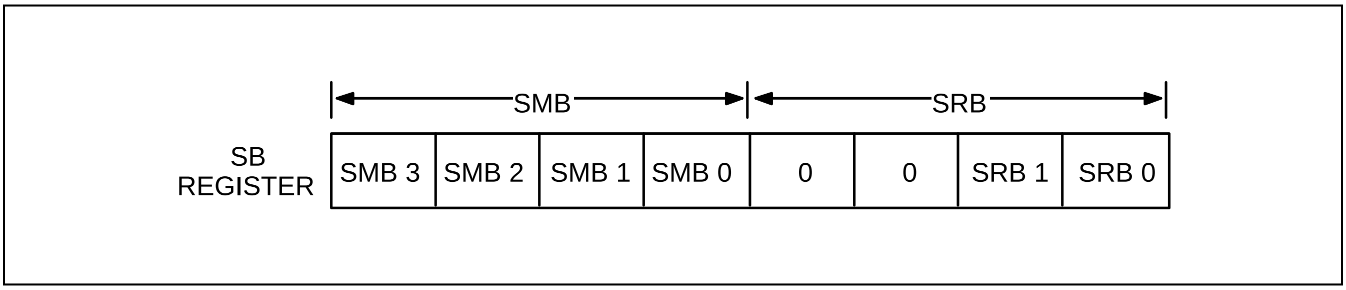


Figure 3-2. SMB and SRB Values in the SB Register

Select Register Bank (SRB) Instruction

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

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

Select Memory Bank (SMB) Instruction

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

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

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

DIRECT AND INDIRECT ADDRESSING

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

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

1-BIT ADDRESSING

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

Operand Notation	Addressing Mode Description	EMB Flag Setting	Addressable Area	Memory Bank	Hardware I/O Mapping
DA.b	Direct: bit is indicated by the RAM address (DA), memory bank selection, and specified bit number (b).	0	000H-07FH	Bank 0	–
			F80H-FFFH	Bank 15	All 1-bit addressable peripherals (SMB = 15)
		1	000H-FFFH	SMB = 0, 1, 15	
mema.b	Direct: bit is indicated by addressable area (mema) and bit number (b).	x	FB0H-FBFH FF0H-FFFH	Bank 15	IS0, IS1, EMB, ERB, IEx, IRQx, Pn.m
memb.@L	Indirect: address is indicated by the upper 6 bits of RAM area (memb) and the upper 2 bits of register L, and bit is indicated by the lower 2 bits or register L.	x	FC0H-FFFH	Bank 15	Pn.m
@H + DA.b	Indirect: bit is indicated by the lower four bits of the address (DA), memory bank selection, and the H register identifier.	0	000H-0FFH	Bank 0	–
		1	000H-FFFH	SMB = 0, 1, 15	All 1-bit addressable peripherals (SMB = 15)

NOTE: 'x' means don't care.

☞ PROGRAMMING TIP — 1-Bit Addressing Modes

1-Bit Direct Addressing

1. If EMB = "0":

```
AFLAG   EQU     34H.3
BFLAG   EQU     85H.3
CFLAG   EQU     0BAH.0
        SMB     0
        BITS    AFLAG               ;  34H.3 ← 1
        BITS    BFLAG               ;  F85H.3 ← 1
        BTST    CFLAG               ;  If FBAH.0 = 1, skip
        BITS    BFLAG               ;  Else if, FBAH.0 = 0, F85H.3 ← 1
        BITS    P2.0                ;  FF2H.0 (P2.0) ← 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    P2.0                ;  FF2H.0 (P2.0) ← 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, 1, 15	
@HL	Indirect: 4-bit address indicated by the memory bank selection and register HL	0	000H-0FFH	Bank 0	–
		1	000H-FFFH	SMB = 0, 1, 15	All 4-bit addressable peripherals (SMB = 15)
@WX	Indirect: 4-bit address indicated by register WX	x	000H-0FFH	Bank 0	–
@WL	Indirect: 4-bit address indicated by register WL	x	000H-0FFH	Bank 0	

NOTE: 'x' means don't care.

☞ PROGRAMMING TIP — 4-Bit Addressing Modes

4-Bit Direct Addressing

1. If EMB = "0":

```
ADATA     EQU       46H
BDATA     EQU       85H
          SMB       15                  ; Non-essential instruction, since EMB = "0"
          LD        A,P4                ; A ← (P4)
          SMB       0                   ; Non-essential instruction, since EMB = "0"
          LD        ADATA,A             ; (046H) ← A
          LD        BDATA,A             ; (F85H (BMOD)) ← A
```

2. If EMB = "1":

```
ADATA     EQU       46H
BDATA     EQU       85H
          SMB       15
          LD        A,P4                ; A ← (P4)
          SMB       0
          LD        ADATA,A             ; (046H) ← A
          LD        BDATA,A             ; (085H) ← A
```

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

4-Bit Indirect Addressing (Example 1)

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

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

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

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

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

4-Bit Indirect Addressing (Example 2)

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

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

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

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

8-BIT ADDRESSING

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

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

☞ PROGRAMMING TIP — 8-Bit Addressing Modes

8-Bit Direct Addressing

1. If EMB = "0":

```
ADATA   EQU     46H
BDATA   EQU     8CH
        SMB     15              ; Non-essential instruction, since EMB = "0"
        LD      EA,P4           ; E ← (P5), A ← (P4)
        SMB     0
        LD      ADATA,EA        ; (046H) ← A, (047H) ← E
        LD      BDATA,EA        ; (F8CH) ← A, (F8DH) ← E
```

2. If EMB = "1":

```
ADATA   EQU     46H
BDATA   EQU     8CH
        SMB     15
        LD      EA,P4           ; E ← (P5), A ← (P4)
        SMB     0
        LD      ADATA,EA        ; (046H) ← A, (047H) ← E
        LD      BDATA,EA        ; (08CH) ← A, (08DH) ← E
```

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

8-Bit Indirect Addressing

1. If EMB = "0":

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

2. If EMB = "1":

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

4 MEMORY MAP

OVERVIEW

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

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

I/O MAP FOR HARDWARE REGISTERS

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

— Register address

— Register name (mnemonic for program addressing)

— Bit values (both addressable and non-manipulable)

— Read-only, write-only, or read and write addressability

— 1-bit, 4-bit, or 8-bit data manipulation characteristics

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

Addressing	Symbol	Description	Affected Memory mapped I/O
1-bit direct addressing	DA.b	The bit indicated by memory bank, DA and bit. (EMB=0, or EMB=1 and SMB 15)	All peripheral hardware that can be manipulated in 1 bit.
4-bit direct addressing	DA	The address indicated by memory bank and DA. (EMB=0, or EMB=1 and SMB 15)	All peripheral hardware that can be manipulated in 4 bits.
8-bit direct addressing	DA	The address (DA specifies an even address) indicated by memory bank and DA. (EMB=0, or EMB=1 and SMB 15)	All peripheral hardware that can be manipulated in 8 bits.
4-bit indirect addressing	@HL	The address indicated by memory bank and HL register. (EMB=1 and SMB 15)	All peripheral hardware that can be manipulated in 4 bits.
8-bit indirect addressing	@HL	The address indicated by memory bank and HL (the contents of the L register are even). (EMB=1 and SMB 15)	All peripheral hardware that can be manipulated in 8 bit.
1-bit manipulating addressing	mema.b	The bit indicated by mema and bit. (regardless of the status of EMB and SMB)	IS0, IS1, EMB, ERB, IEx, IRQx, Pn.m
	memb.@L	The bit indicated by the lower 2 bits of the L register of the address indicated by the upper 10 bits of memb and the upper 2 bits of the L reigster. (regardless of the status of EMB and SMB)	Pn.m
	@H+DA.b	The bit of the address indicated by memory bank, H register and the lower 4 bits of DA. (EMB=1 and SMB=15)	All peripheral hardware that can be manipulated in 1 bit.

Table 4-2. I/O Map for Memory Bank 15

Memory Bank 15							Addressing Mode		
Address	**Register**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**R/W**	**1-Bit**	**4-Bit**	**8-Bit**
F80H	SP	.3	.2	.1	"0"	R/W	No	No	Yes
F81H		.7	.6	.5	.4				
Locations F82H-F84H are not mapped.									
F85H	BMOD	.3	.2	.1	.0	W	.3	Yes	No
F86H	BCNT					R	No	No	Yes
F87H									
F88H	WMOD	.3	.2	.1	.0	W	.3 (1)	No	Yes
F89H		.7	"0"	.5	.4				
F8CH	LMOD0	.3	.2	.1	.0	W	No	Yes	No
F8DH	LMOD1	.3	.2	.1	.0	W	No	Yes	No
F90H	TMOD0	.3	.2	"0"	"0"	W	.3	No	Yes
F91H		"0"	.6	.5	.4				
F92H		"0"	TOE0	"U"	"0"	R/W	Yes	Yes	No
Location F93H is not mapped.									
F94H	TCNT0					R	No	No	Yes
F95H									
F96H	TREF0					W	No	No	Yes
F97H									
FA0H	PASR	.3	.2	.1	.0	W	No	No	Yes
FA1H		"0"	"0"	"0"	.4				
FA2H	KSR0	.3	.2	.1	.0	W	No	Yes	No
FA3H	KSR1	.3	.2	.1	.0				
FA4H	KSR2	.3	.2	.1	.0				
FA5H	KSR3	.3	.2	.1	.0				
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		No	Yes	No
FB4H	IMOD0	"0"	"0"	.1	.0		No	Yes	No
FB5H	IMOD1	"0"	"0"	.1	.0		No	Yes	No

Table 4-2. 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**
FB6H	IMOD2	"0"	"0"	.1	.0		No	Yes	No
FB7H	SCMOD	.3	"0"	"0"	.0	W	Yes	No	No
FB8H		"U"	"U"	IEB	IRQB	R/W	Yes	No	No
FBAH		"U"	"U"	IEW	IRQW	R/W	Yes	No	No
FBCH		"U"	"U"	IET0	IRQT0	R/W	Yes	No	No
FBDH		"U"	"U"	IEP0	IRQP0	R/W	Yes	No	No
FBEH		IE1	IRQ1	IE0	IRQ0	R/W	Yes	Yes	No
FBFH		"U"	"U"	IE2	IRQ2	R/W	Yes	No	No
FC0H	BSC0					R/W	Yes	Yes	Yes
FC1H	BSC1								
FC2H	BSC2								
FC3H	BSC3								
FD0H	CLMOD	.3	"0"	.1	.0	W	No	Yes	No
FDAH	PNE1[3]	"0"	.2	.1	.0	W	No	No	Yes
FDBH		.7	.6	.5	.4				
FDCH	PUMOD0[4]	"0"	PUR2	PUR1	"0"	W	No	No	Yes
FDDH		PUR7	PUR6	PUR5	PUR4				
FE8H	PMG1	PM2.1	PM2.0	"0"	"0"	W	No	No	Yes
FE9H		PM6.3	PM6.2	PM6.1	PM6.0				
FEAH	PMG2	PM7	"0"	PM5	PM4	W	No	Yes	No
FF0H	Port0 (P0)	.3	.2	.1	.0	R	Yes	Yes	No
FF1H	Port1 (P1)	"0"	"0"	.1	.0	R	Yes	Yes	No
FF2H	Port2 (P2)	"0"	"0"	.1	.0	R/W	Yes	Yes	No
FF4H	Port4 (P4)	"0"	.2	.1	.0	R/W	Yes	Yes	Yes
FF5H	Port5 (P5)	.3/.7	.2/.6	.1/.5	.0/.4				
FF6H	Port6 (P6)	.3	.2	.1	.0	R/W	Yes	Yes	Yes
FF7H	Port7 (P7)	.3/.7	.2/.6	.1/.5	.0/.4				

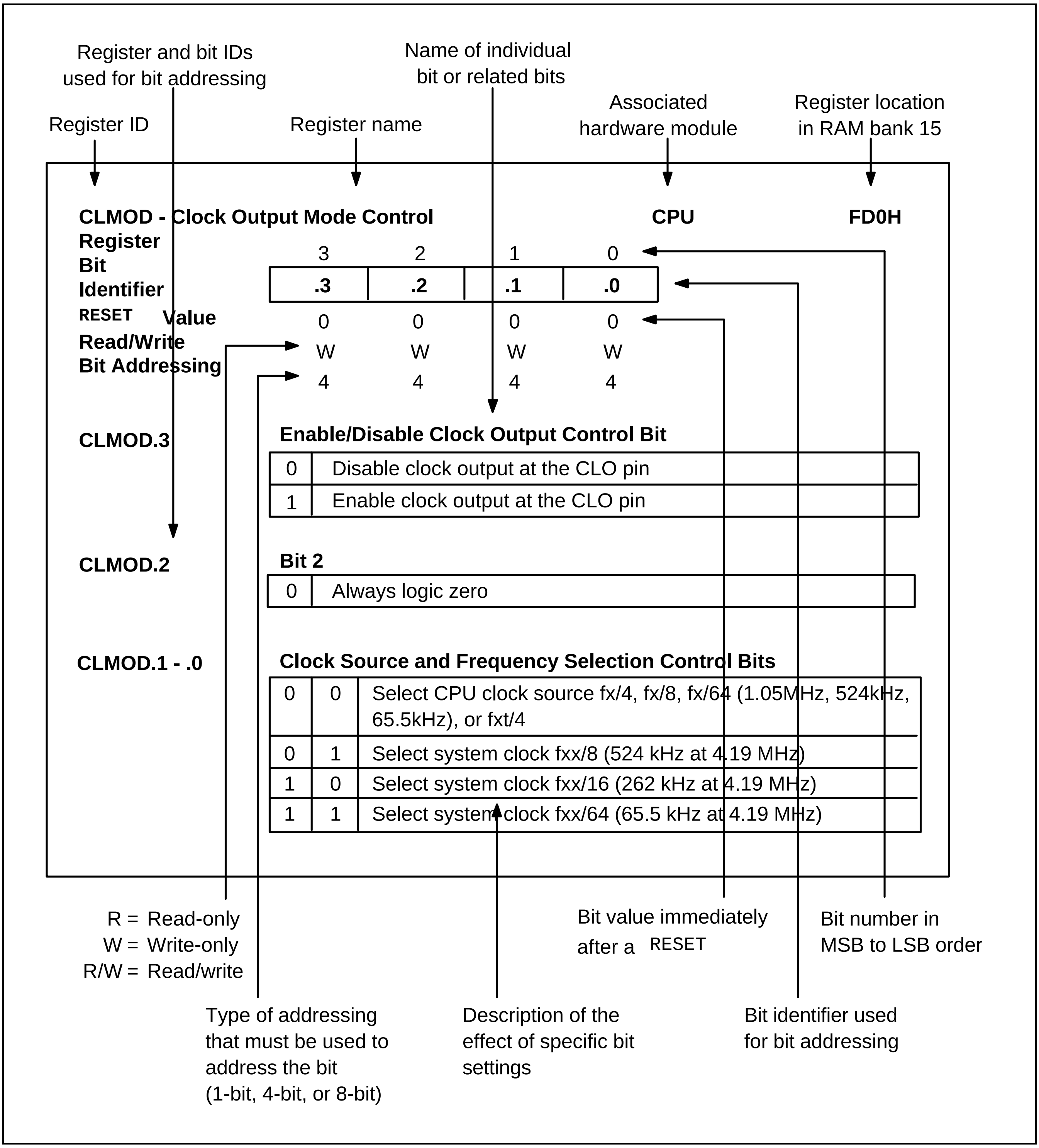
NOTES:

1. Bit 3 in the WMOD register is read only.
2. The carry flag can be read or written by specific bit manipulation instructions only.
3. The PNE1 register is used to select the output types of ports 4,5 (zero: push-pull output type, one: open-drain output type). The reset value of the PNE1 register is "00H".
4. The PUMOD0 register is used to enable/disable the internal pull-up resistors of port 1, 2, 4, 5, 6 and 7 (zero: disable, one: enable). The reset value of the PUMOD0 register is "00H".
5. The "U" means that bit is undefined.

REGISTER DESCRIPTIONS

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

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


Register and bit IDs used for bit addressing
Name of individual bit or related bits
Register ID
Register name
Associated hardware module
Register location in RAM bank 15
R = Read-only
W = Write-only
R/W = Read/write
Bit value immediately after 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**

	3	2	1	0
Bit 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 at the CLO pin
1	Enable clock output at the CLO pin

CLMOD.2 **Bit 2**

0	Always logic zero

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

0	0	Select CPU clock source fx/4, fx/8, fx/64 (1.05MHz, 524kHz, 65.5kHz), or fxt/4
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

BMOD — Basic Timer Mode Register BT F85H

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

BMOD.3 **Basic Timer Restart Bit**

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

BMOD.2 - .0 **Input Clock Frequency and Interrupt Interval Time**

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

NOTES:

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

CLMOD — Clock output Mode Register CPU FD0H

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

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

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

CLMOD.2 **Bit 2**

0	Always logic zero

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

0	0	Select CPU clock source fx/4, fx/8, or fx/64 (0.05MHz, 524 kHz, or 65.5 kHz) or fxt/4
0	1	Select system clock fxx/8 (524 kHz)
1	0	Select system clock fxx/16 (262 kHz)
1	1	Select system clock fxx/64 (65.5 kHz)

NOTE: 'fxx' is the system clock, given a clock frequency of 4.19 MHz.

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

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

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

IE1 **INT1 Interrupt Enable Flag**

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

IRQ1 **INT1 Interrupt Request Flag**

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

IE0 **INT0 Interrupt Enable Flag**

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

IRQ0 **INT0 Interrupt Request Flag**

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

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

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

.3 - .2 **Bits 3-2**

U	This bit is undefined.

IE2 **INT2 Interrupt Enable Flag**

0	Disable INT2 interrupt requests at the KS0-KS7 pins
1	Enable INT2 interrupt requests at the KS0-KS7 pins

IRQ2 **INT2 Interrupt Request Flag**

-	Generate INT2 quasi-interrupt (This bit is set and is not cleared automatically by hardware when a falling edge is detected at one of the KS0-KS7 pins. Since INT2 is a quasi-interrupt, IRQ2 flag must be cleared by software.)

NOTE: The "U" means a undefined register bit

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

Bit	3	2	1	0
Identifier	**"U"**	**"U"**	**IEB**	**IRQB**
RESET Value	U	U	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1	1	1	1

.3 - .2 **Bits 3-2**

U	This bit is undefined.

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.)

NOTE: The "U" means a undefined register bit.

IEP0, IRQP0 — INTP0 Interrupt Enable/Request Flags CPU FBDH

Bit	3	2	1	0
Identifier	**"U"**	**"U"**	**IEP0**	**IRQP0**
RESET Value	U	U	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1	1	1	1

.3 - .2 **Bits 3-2**

U	This bit is undefined.

IEP0 **INTS Interrupt Enable Flag**

0	Disable INTP0 interrupt requests
1	Enable INTP0 interrupt requests

IRQP0 **INTS Interrupt Request Flag**

-	Generate INTP0 interrupt (This bit is set and cleared automatically by hardware when falling edge is detected at K0-K3 pin.)

NOTE: The "U" means a undefined register bit.

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

Bit	3	2	1	0
Identifier	**"U"**	**"U"**	**IET0**	**IRPT0**
RESET Value	U	U	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1	1	1	1

.3 - .2 **Bits 3-2**

U	This bit is undefined.

IET **INTT0 Interrupt Enable Flag**

0	Disable INTT interrupt requests
1	Enable INTT interrupt requests

IRQT **INTT0 Interrupt Request Flag**

-	Generate INTT interrupt (This bit is set and cleared automatically by hardware when contents of TCNT and TREF registers match.)

NOTE: The "U" means a undefined register bit.

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

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

.3 - .2 **Bits 3-2**

U	This bit is undefined.

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 3.19 milliseconds.)

NOTES:

1. Since INTW is a quasi-interrupt, the IRQW flag must be cleared by software.
2. The "U" means a undefined register bit.

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

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

IMOD0.3 - .2 **Bits 3-2**

0	Always logic zero

IMOD0.1 - .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 CPU FB5H

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

IMOD1.3 - .2

Bits 3-2

0	Always logic zero

IMOD1.1 - .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 (IRQ1) cannot be set to logic one

IMOD2 — External Interrupt 2 (INT2) Mode Register CPU FB6H

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

IMOD2.3 - .2

Bits 3-2

0	Always logic zero

IMOD2.1 - .0

External Interrupt 2 Edge Detection Selection Bit

0	0	Not available
0	1	Interrupt request at KS4-KS7 triggered by falling edge
1	0	Interrupt request at KS2-KS7 triggered by falling edge
1	1	Interrupt request at KS0-KS7 triggered by falling edge

IPR —Interrupt Priority Register CPU FB2H

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

IME **Interrupt Master Enable Bit**

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

IPR.2 - .0 **Interrupt Priority Register Setting**

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 interrupt at high priority
0	1	0	Process INT0 interrupt at highest priority
0	1	1	Process INT1 interrupt at highest priority
1	0	0	Process INTP0 interrupt at highest priority
1	0	1	Process INTT0 interrupt at highest priority
1	1	0	N/A
1	1	1	N/A

NOTE: The “N/A” means a not available.

LMOD0 — LCD Mode Register 0 LCD F8CH

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

LMOD0.3 - .0 **LCD Display Control Bits**

0	1	0	0	Normal LCD display on
0	1	0	1	Dimming mode (key strobe status)
1	0	0	1	LCD display off, key check signal output, key check state

NOTE: If one of 16 bits (KSR0.0-KSR3.3) is set to logic "0", the corresponding segment pin becomes the low level, when the LCD Display is off by LMOD0 ← 9H.

LMOD1 — LCD Mode Register 1 LCD F8DH

	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

LMOD1.3 - .2 **LCD Clock Frequency Selection Bits (fx=4.19 MHz)** [(1)]

0	0	tF = 75.85 Hz
0	1	tF= 151.7 Hz
1	0	tF= 303.4 Hz
1	1	tF= 606.8 Hz

LMOD1.1 **LCD Display Output Drive Control Bit**

0	Select normal LCD bias resistor.
1	Choose the small LCD bias resistors. In this case, you can make LCD output increased but it needs more current consumption.

LMOD1.0 **Key Strobe Signal Output and Port0 Pull-up Control Bit**

0	Key strobe signal output
1	[(2)]Disable key strobe signal output, port0 pull-up resistors enable.

NOTES:

1. LCD clock can be selected only when main clock(fx) is used as clock source of watch timer. When sub clock(fxt) is used as clock source of watch timer, LCD frame frequency(tF) is always 75.85Hz.
2. In this case, there is no key strobe signal output and a port0 pull-up resistor is always enabled. The port0 pins change to external interrupt pins which detect a falling edge.

PASR — Page Selection Register MEMORY FA0H

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

PASR.7 - .5 **Bits 7 - 4**

0	Always logic zero

PASR.4 - .0 **Page Selection Register in the Bank1**

0	0	0	0	0	00H page in the Bank1
0	0	0	0	1	01H page in the Bank1
		• • • •			•••••••••••
1	0	0	1	1	13H page for LCD display register in the Bank1

NOTES:
1. The 00H-13H pages can be used in the KS57C21408.
2. The 00H-08H, and 13H can be used in the KS57C21418.

PCON — Power Control Register CPU FB3H

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

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

0	0	Normal CPU operating mode
0	1	Idle power-down mode
1	0	Stop power-down mode (only main system clock stops)
1	1	Not available

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

0	0	If SCMOD.0 = "0", fx/64; if SCMOD.0 = "1", Not available
1	0	If SCMOD.0 = "0", fx/8; if SCMOD.0 = "1", Not available
1	1	If SCMOD.0 = "0", fx/4; if SCMOD.0 = "1", fxt/4

NOTE: 'fx' is the Main-system clock; 'fxt' is the Sub-system clock.

PMG1 — Port I/O Mode Register 1 (Group 1: Port 2,6) I/O FE9H,FE8H

Bit	7	6	5	4	3	2	1	0
Identifier	**PM6.3**	**PM6.2**	**PM6.1**	**PM6.0**	**PM2.1**	**PM2.0**	**"0"**	**"0"**
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PM6.3 **P6.3 I/O Mode Selection flag**

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

PM6.2 **P6.2 I/O Mode Selection Flag**

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

PM6.1 **P6.1 I/O Mode Selection Flag**

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

PM6.0 **P6.0 I/O Mode Selection Flag**

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

PM2.1 **P2.1 I/O Mode Selection Flag**

0	Set P2.1 to input mode
1	Set P2.1 to output mode

PM2.0 **P2.0 I/O Mode Selection Flag**

0	Set P2.0 to input mode
1	Set P2.0 to output mode

.1 - .0 **Bits 1 - 0**

0	Always logic zero

PMG2 — Port I/O Mode Register 2 (Group 2: Port 4,5,7) I/O FEAH

Bit	3	2	1	0
Identifier	PM7	"0"	PM5	PM4
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	8	8	8	8

PM7 **P7 I/O Mode Selection Flag**

0	Set P7 to input mode
1	Set P7 to output mode

.2 **Bit 2**

0	Always logic zero

PM5 **P5 I/O Mode Selection Flag**

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

PM4 **P4 I/O Mode Selection Flag**

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

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

Bit	7	6	5	4	3	2	1	0
Identifier	PNE5.3	PNE5.2	PNE5.1	PNE5.0	"0"	PNE4.2	PNE4.1	PNE4.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PNE5.3 **P 5.3 N-Channel Output Configuration bit**

0	Configure P5.3 as a push-pull
1	Configure P5.3 as a N-channel open-drain

PNE5.2 **P 5.2 Output Configuration**

0	Configure P5.2 as a push-pull
1	Configure P5.2 as a N-channel open-drain

PNE5.1 **P 5.1 Output Configuration**

0	Configure P5.1 as a push-pull
1	Configure P5.1 as a N-channel open-drain

PNE5.0 **P 5.0 Output Configuration**

0	Configure P5.0 as a push-pull
1	Configure P5.0 as a N-channel open-drain

.3 **Bit 3**

0	Always logic zero

PNE4.2 **P 4.2 Output Configuration**

0	Configure P4.2 as a push-pull
1	Configure P4.2 as a N-channel open-drain

PNE4.1 **P4.1 Output Configuration**

0	Configure P4.1 as a push-pull
1	Configure P4.1 as a N-channel open-drain

PNE4.0 **Port 4.0 Output Configuration**

0	Configure P4.0 as a push-pull
1	Configure P4.0 as a N-channel open-drain

PSW — Program Status Word CPU FB1H,FB0H

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

C **Carry Flag**

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

SC2 - SC0 **Skip Condition Flags**

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

IS1, IS0 **Interrupt Status Flags**

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

EMB **Enable Data Memory Bank Flag**

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

ERB **Enable Register Bank Flag**

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

NOTES:

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

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

Bit	7	6	5	4	3	2	1	0
Identifier	PUR7	PUR6	PUR5	PUR4	"0"	PUR2	PUR1	"0"
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PUR7 **Connect/Disconnect Port 7 Pull-Up Resistor Control Bit**

0	Disconnect port 7 pull-up resistor
1	Connect port 7 pull-up resistor

PUR6 **Connect/Disconnect Port 6 Pull-Up Resistor Control Bit**

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

PUR5 **Connect/Disconnect Port 5 Pull-Up Resistor Control Bit**

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

PUR4 **Connect/Disconnect Port 4 Pull-Up Resistor Control Bit**

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

.3 **Bit 3**

0	Always logic zero

PUR2 **Connect/Disconnect Port 1 Pull-Up Resistor Control Bit**

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

PUR1 **Connect/Disconnect Port 2 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

SCMOD — System Clock Mode Control Register CPU FB7H

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

SCMOD.3 **Bit 3**

0	Enable Main-system clock (fx)
1	Disable Main-system clock (fx)

SCMOD.2-.1 **Bits 2-1**

0	Always logic zero

SCMOD.0 **Bit 0**

0	Select Main-system clock (fx)
1	Select Sub-system clock (fxt)

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/Counter 0 Mode Register T/C0 F90H,F91H

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

TMOD0.7 **Bit 7**

0	Always logic zero

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

0	0	0	External clock input at TCL0 pin on rising edge
0	0	1	External clock input at TCL0 pin on falling edge
1	0	0	fxx /2^{10} (4.09 kHz)
1	0	1	fxx/2^{6} (65.5 kHz)
1	1	0	fxx/2^{4} (262kHz)
1	1	1	fxx (4.19 MHz)

NOTE: "fxx" is selected system clock of 4.19MHz

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

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

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

0	Disable timer/counter0 ; retain TCNT0 contents
1	Enable timer/counter0

TMOD0.1 **Bit 1**

0	Always logic zero

TMOD0.0 **Bit 0**

0	Always logic zero

TOE0 — Timer/Output Enable Flag Register T/C0 F92H

Bit	3	2	1	0
Identifier	**"0"**	**TOE0**	**"U"**	**"0"**
RESET Value	0	0	U	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

.3 **Bit 3**

0	Always logic zero

TOE0 **Clear Counter And Resume Counting Control Bit**

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

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

U	This bit is undefined

NOTE: The "U" means that the bit is undefined.

.0 **Bit 0**

0	Always logic zero

WMOD — Watch/Timer Mode Register WT F88H, F89H

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

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

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

WMOD.6 **Bit 6**

0	Always logic zero

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

0	0	2 kHz buzzer (BUZ) 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 **XT_{in} Input Level Control Bit**

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

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

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

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

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

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

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

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

NOTES

6 OSCILLATOR CIRCUITS

OVERVIEW

The KS57C21408/C21418/P21408 microcontroller has two oscillator circuits: a main-system clock circuit, and a sub-system clock circuit. The CPU and peripheral hardware operate on the system clock frequency supplied through these circuits. Specifically, a clock pulse is required by the following peripheral modules:

— LCD controller

— Basic timer

— Timer/counter 0

— Watch timer

— Clock output circuit

CPU Clock Notation

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

fx	Main-system clock
fxt	Sub-system clock
fxx	Selected system clock

Clock Control Registers

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

The power control register, 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 *sub-system clock (fxt)* as the CPU clock and to start (or stop) main-system clock oscillation. The resulting clock source, either main-system clock or sub-system clock, is referred to as the *selected system clock (fxx)*.

The main-system clock is selected and oscillation started when all SCMOD bits are cleared to logic zero. By setting SCMOD.3 and SCMOD.0 to different values, you can select a sub-system clock source and start or stop main-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 sub-system clock frequencies can only be divided by 4. By manipulating PCON bits 1 and 0, you select one of the following frequencies as the CPU Clock.

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

Using a Sub-system Clock

If a sub-system clock is being used as the selected system clock, the idle power-down mode can be initiated by executing an IDLE instruction. Since the sub-system clock source cannot be stopped internally, you cannot, however, use a STOP instruction to enable the stop power-down mode.

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

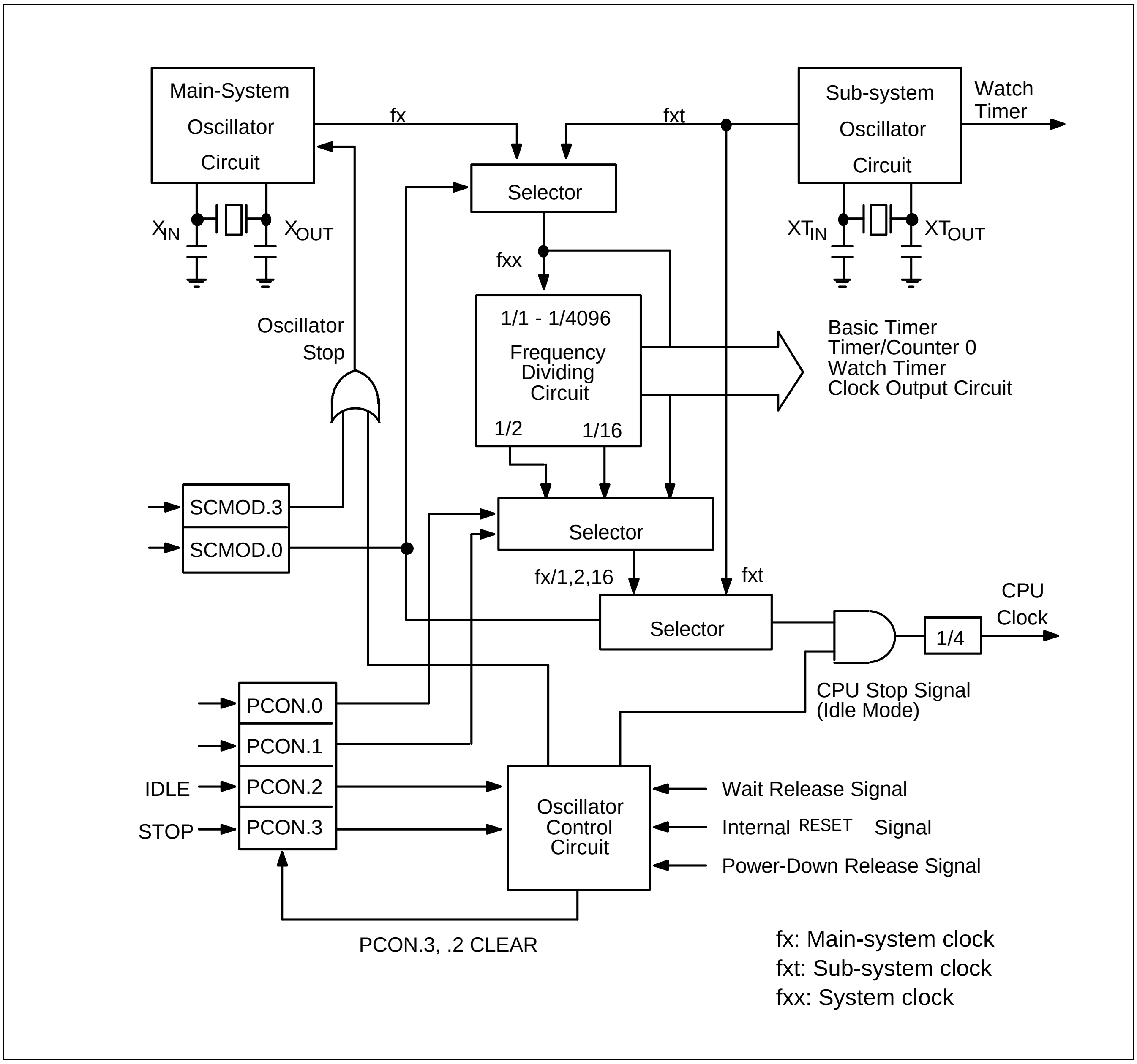


Figure 6-1. Clock Circuit Diagram

MAIN-SYSTEM OSCILLATOR CIRCUITS

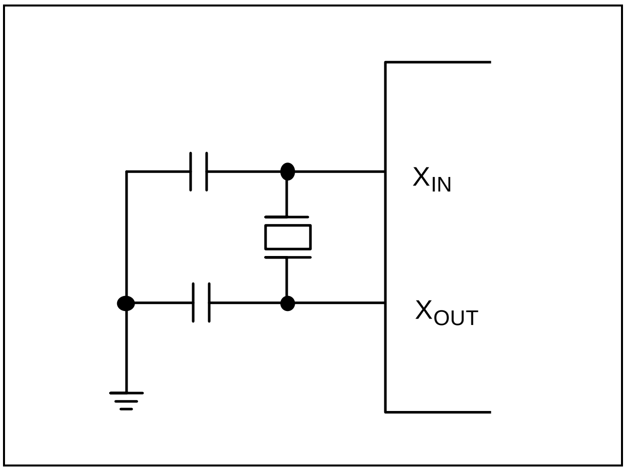


Figure 6-2. Crystal/Ceramic Oscillator

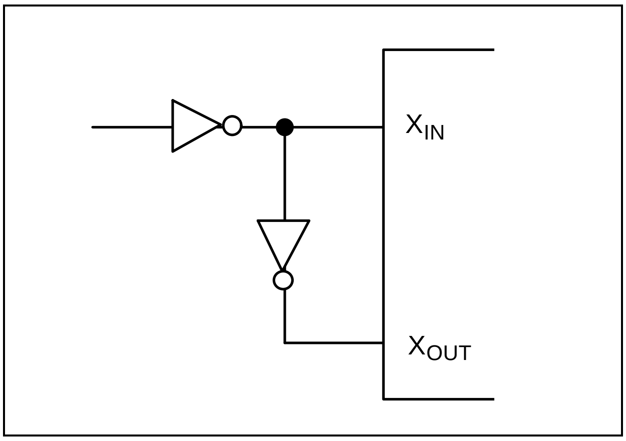


Figure 6-3. External Oscillator

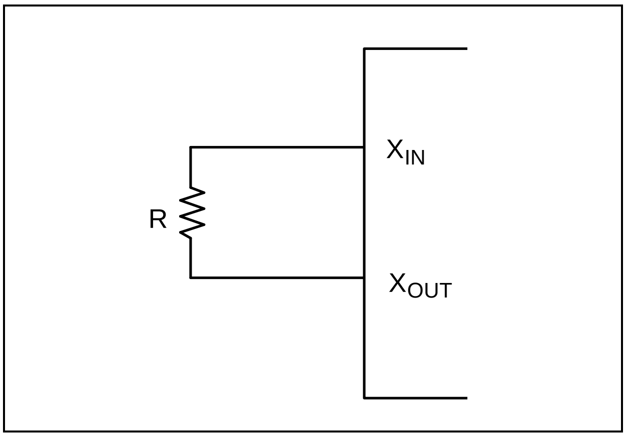


Figure 6-4. RC Oscillator

SUB-SYSTEM OSCILLATOR CIRCUITS

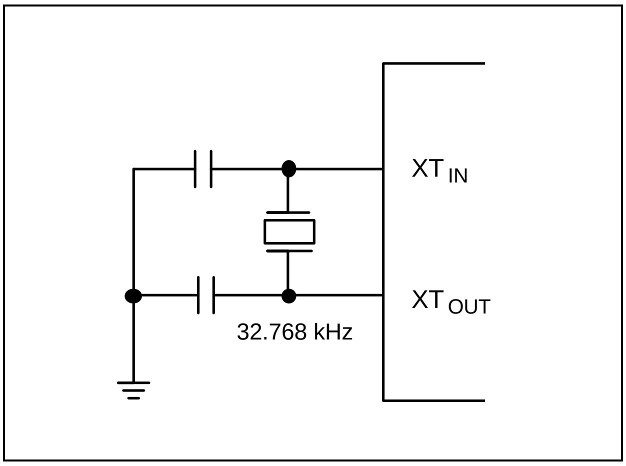


Figure 6-5. Crystal/Ceramic Oscillator

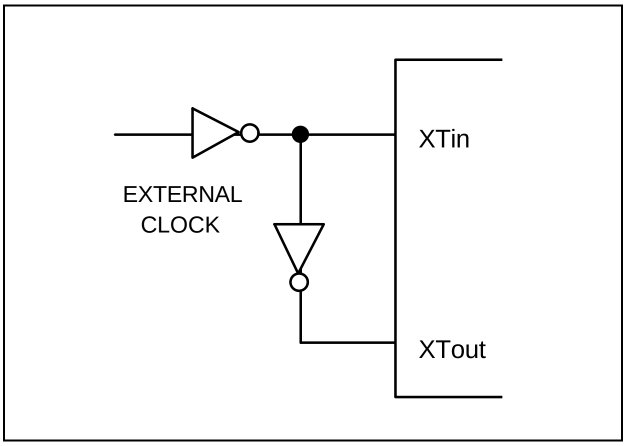


Figure 6-6. External Oscillator

POWER CONTROL REGISTER (PCON)

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

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

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

— Main-system clock (fx) or sub-system clock (fxt);

— Divided fx/4, 8, 64 or fxt/4 clock frequency.

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

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

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

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

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

☞ PROGRAMMING TIP — Setting the CPU Clock

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

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

INSTRUCTION CYCLE TIMES

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

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

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

SYSTEM CLOCK MODE REGISTER (SCMOD)

The system clock mode register, SCMOD, is a 4-bit register that is used to select the CPU clock and to control main-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.
When the clock source is sub-system clock, main-system clock oscillation is stopped by setting SCMOD.3, but sub-system clock cannot be stopped. SCMOD.0 and SCMOD.3 cannot be simultaneously modified.

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.

Only its least significant and most significant bits can be manipulated by 1-bit write instructions (In other words, SCMOD.0 and SCMOD.3 cannot be modified simultaneously by a 4-bit write). Bits 2 and 1 are always logic zero.

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

A sub-system 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 sub-system 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 it must not be used. To operate safely, main oscillation clock should be stopped 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.0**	**CPU Clock Source**	**fx Oscillation**
0	0	fx	On
0	1	fxt	On
1	1	fxt	Off

NOTE: fx is main-system clock and fxt is sub-system clock.

Table 6-4. Main Oscillation Stop Mode

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

NOTES:

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

SWITCHING THE CPU CLOCK

Together, bit settings in the power control register, PCON, and the system clock mode register, SCMOD, determine whether a main-system or a sub-system 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 sub-system clocks and to modify operating frequencies.

SCMOD.3 and SCMOD.0 select the main-system clock (fx) or a sub-system clock (fxt) and start or stop main-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 sub-system 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 sub-system clock to the main-system clock: First, clear SCMOD.3 to "0" to enable main-system clock oscillation. Then, after a certain number of machine cycles has elapsed, select the main-system clock by clearing all SCMOD values to logic zero.

Following a RESET, CPU operation starts with the lowest main-system clock frequency of 15.3 µsec at 4.19 MHz after the standard oscillation stabilization interval of 31.3 ms has elapsed. Table 6-3 details the number of machine cycles that must elapse before a CPU clock switch modification goes into effect.

Table 6-5. 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	N/A		1 MACHINE CYCLE		1 MACHINE CYCLE		N/A
	PCON.0 = 0							
	PCON.1 = 1	8 MACHINE CYCLES		N/A		1 MACHINE CYCLES		N/A
	PCON.0 = 0							
	PCON.1 = 1	16 MACHINE CYCLES		1 MACHINE CYCLES		N/A		fx / 4fxt
	PCON.0 = 1							
SCMOD.0 = 1		N/A		N/A		1MACHINE CYCLES*		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 Xin input is connected internally to V_{SS} to avoid current leakage due to the crystal oscillator in stop mode, do not set SCMOD.3 to "1" or STOP instruction when an external clock is used as the main-system clock.
3. When the system clock is switched to the sub-system clock, it is necessary to disable any interrupts which may occur during the time intervals shown in Table 6-3.
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.
6. Main-osc and sub-osc are on, refer to programming tip of P6-9

☞ PROGRAMMING TIP — Switching Between Main-system and Sub-system Clock

1. Switch from the main-system clock to the sub-system clock:

```
MA2SUB      BITS        SCMOD.0         ;   Switches to sub-system 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 sub-system clock to the main-system clock:

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

CLOCK OUTPUT MODE REGISTER (CLMOD)

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

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

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

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

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

CLMOD Bit Settings		Resulting Clock Output	
CLMOD.1	**CLMOD.0**	**Clock Source**	**Frequency**
0	0	CPU clock (fx/4, fx/8, fx/64, fxt/4)	1.05MHz,524kHz,65.5kHz or 8.19kHz
0	1	fxx/8	523.8 kHz
1	0	fxx/16	261.9 kHz
1	1	fxx/64	65.5 kHz

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

NOTES:
1. fx : Main-system clock
2. fxt: Sub-system clock
3. Frequencies assume that fxx, fx = 4,19MHz, and fxt = 32.768kHz

CLOCK OUTPUT CIRCUIT

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

— 4-bit clock output mode register (CLMOD)

— Clock selector

— Port mode flag

— CLO output pin

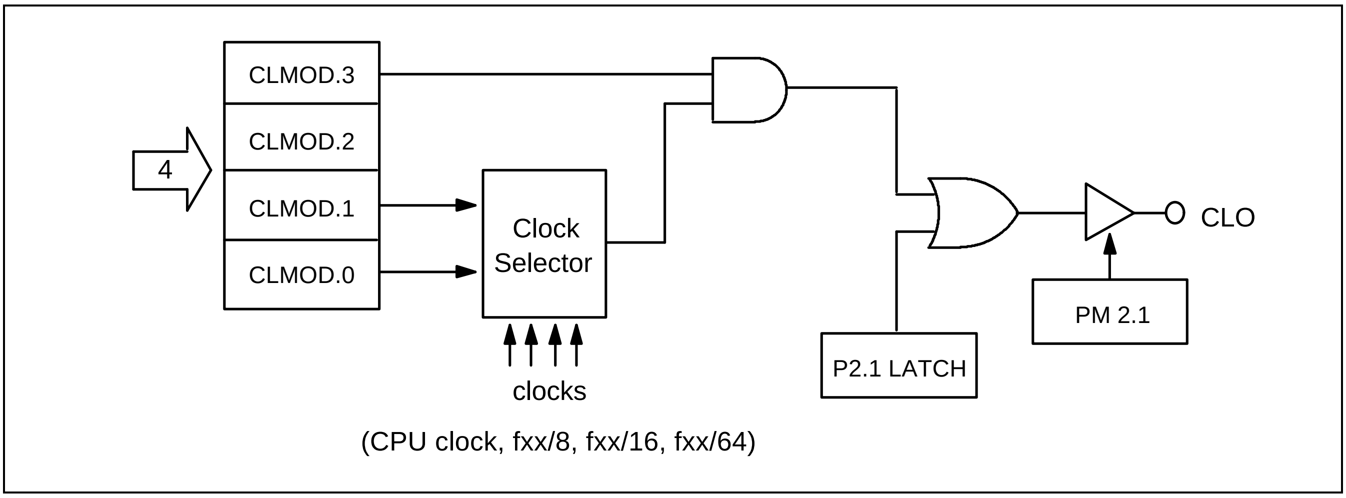


Figure 6-7. CLO Output Pin Circuit Diagram

CLOCK OUTPUT PROCEDURE

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

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

☞ PROGRAMMING TIP — CPU Clock Output to the CLO Pin

To output the CPU clock to the CLO pin:

```
        BITS      EMB
        SMB       15
        LD        EA,#08H
        LD        PMG1,EA              ;  P2.1 ← Output mode
        BITR      P2.1                 ;  Clear the CLO pin output latch
        LD        A,#9H
        LD        CLMOD,A
```

NOTES

7 INTERRUPTS

OVERVIEW

The KS57C21408/C21418/P21408's interrupt control circuit has five functional components:

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

Three kinds of interrupts are supported:

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

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

Interrupt Type	Interrupt Name	Corresponding Port Pins
External interrupts	INT0, INT1, INTP0	P1.0, P1.1, P0(K0-K3)
Internal interrupts	INTB, INTT0	Not applicable
Quasi-interrupts	INT2	P6,P7(KS0–KS7)
	INTW	Not applicable

VECTORED INTERRUPTS

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

— Interrupt enable flag (IEx)

— Interrupt master enable flag (IME)

— Interrupt request flag (IRQx)

— Interrupt status flags (IS0, IS1)

— Interrupt priority register (IPR)

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

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

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

Software-Generated Interrupts

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

Multiple 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 by using the POP instruction.

Power-Down Mode Release

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

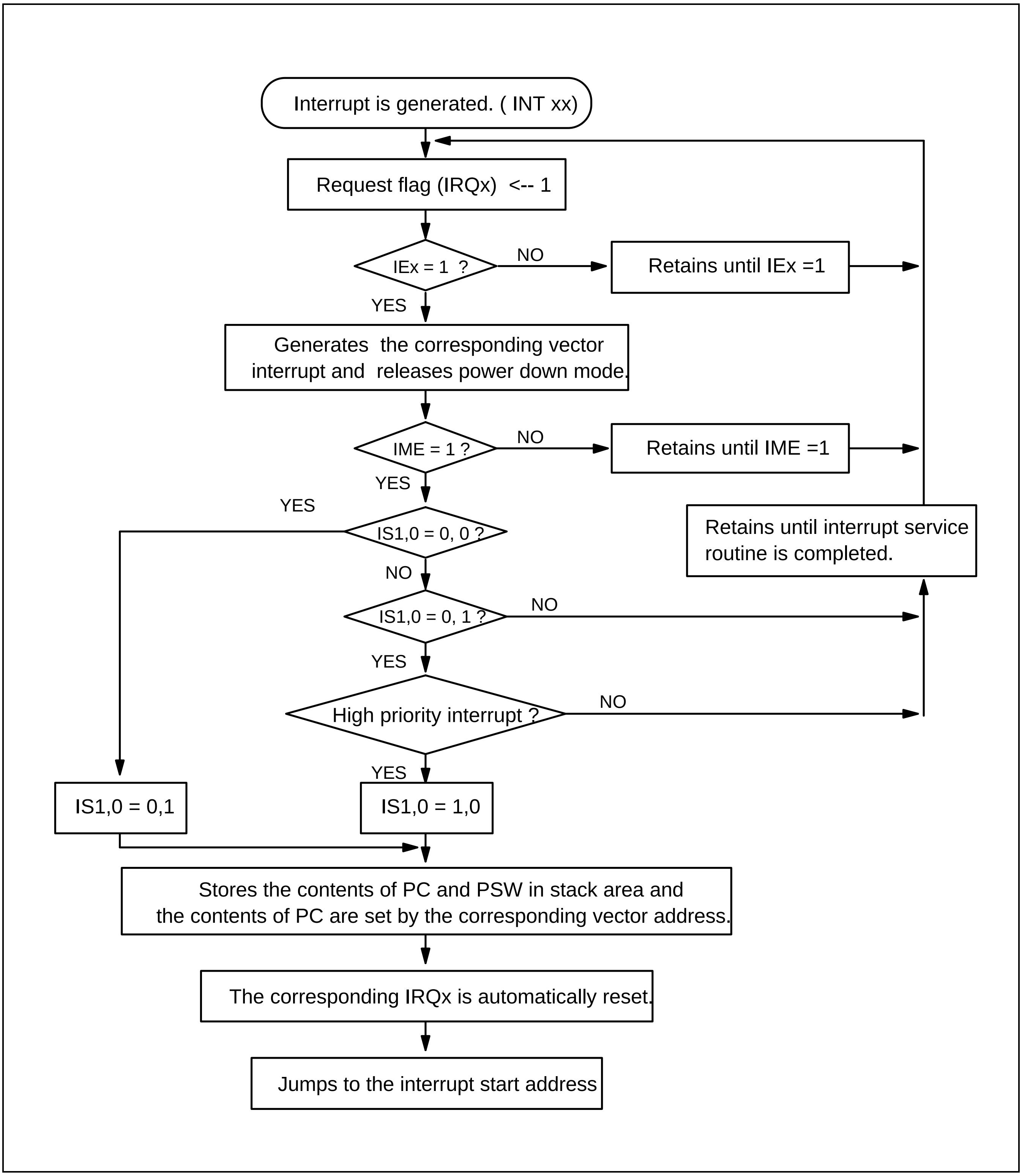

Interrupt is generated. (INT xx)
Request flag (IRQx) <-- 1
IEx = 1 ?
NO
Retains until IEx =1
YES
Generates the corresponding vector interrupt and releases power down mode.
IME = 1 ?
NO
Retains until IME =1
YES
YES
IS1,0 = 0, 0 ?
Retains until interrupt service routine is completed.
NO
IS1,0 = 0, 1 ?
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 and the contents of PC are set by the corresponding vector address.
The corresponding IRQx is automatically reset.
Jumps to the interrupt start address

Figure 7-1. Interrupt Execution Flowchart

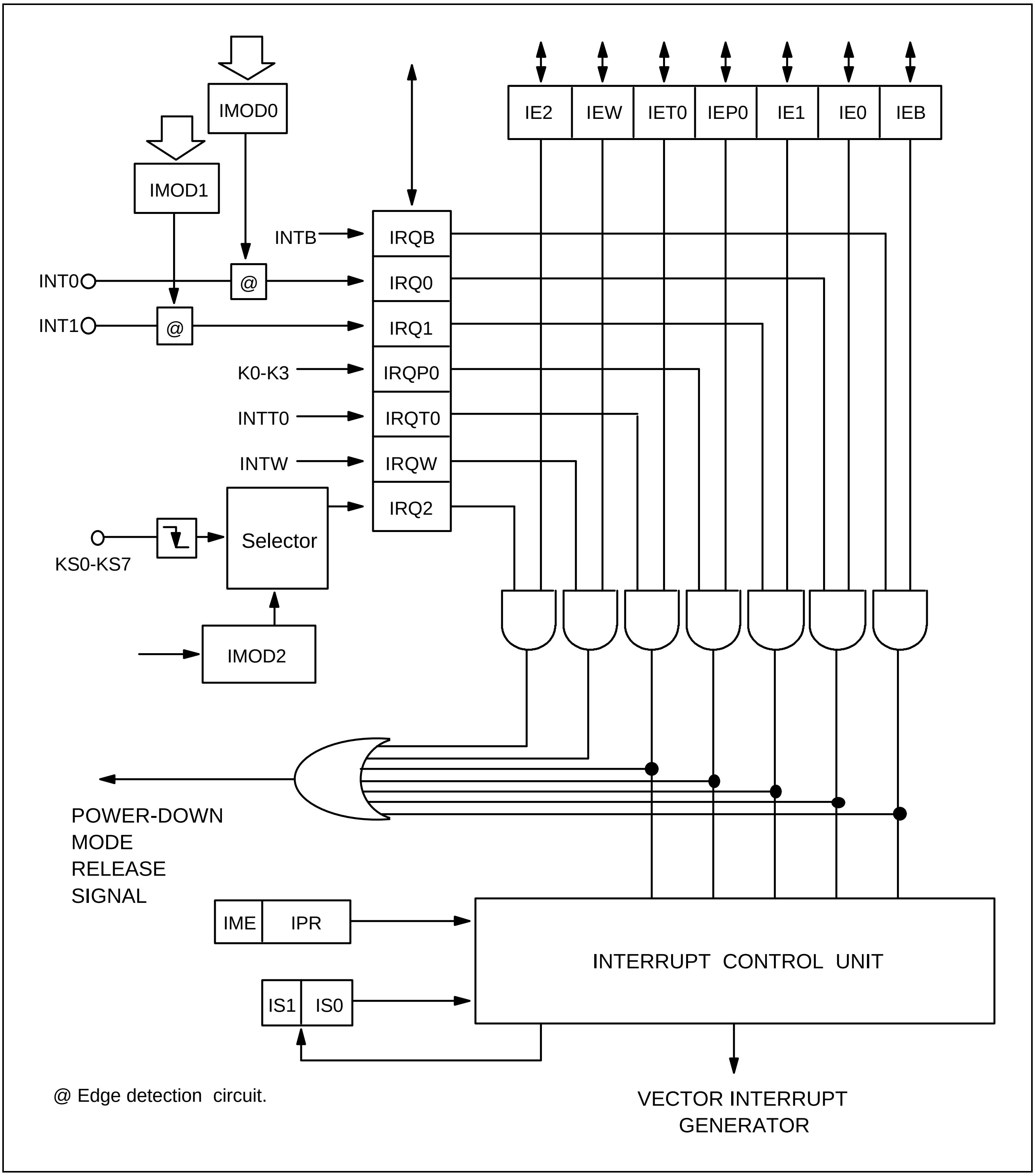


Figure 7-2. Interrupt Control Circuit Diagram

MULTIPLE INTERRUPTS

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

Two-Level Interrupt Handling

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

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

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

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

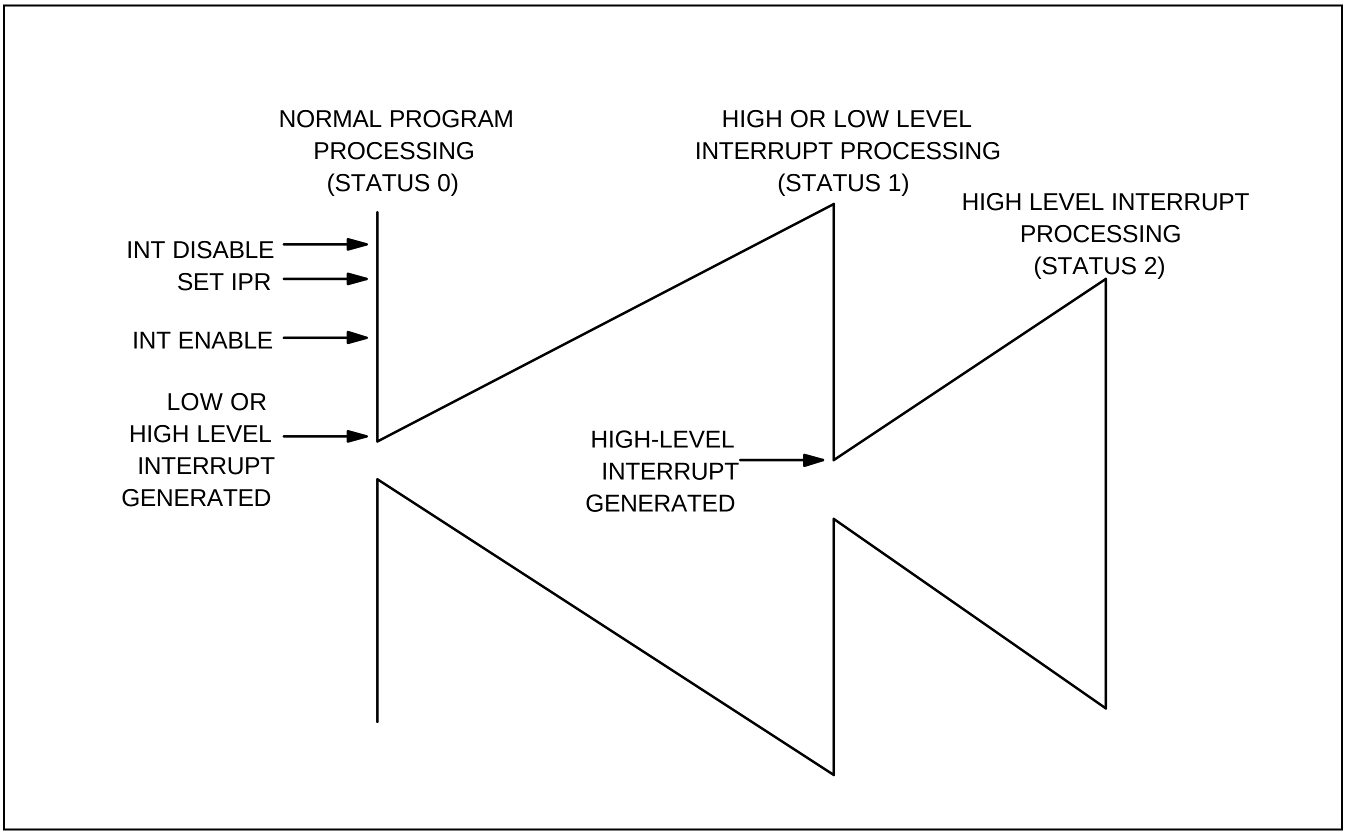


Figure 7-3. Two-Level Interrupt Handling

Multi-Level Interrupt Handling

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

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

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

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

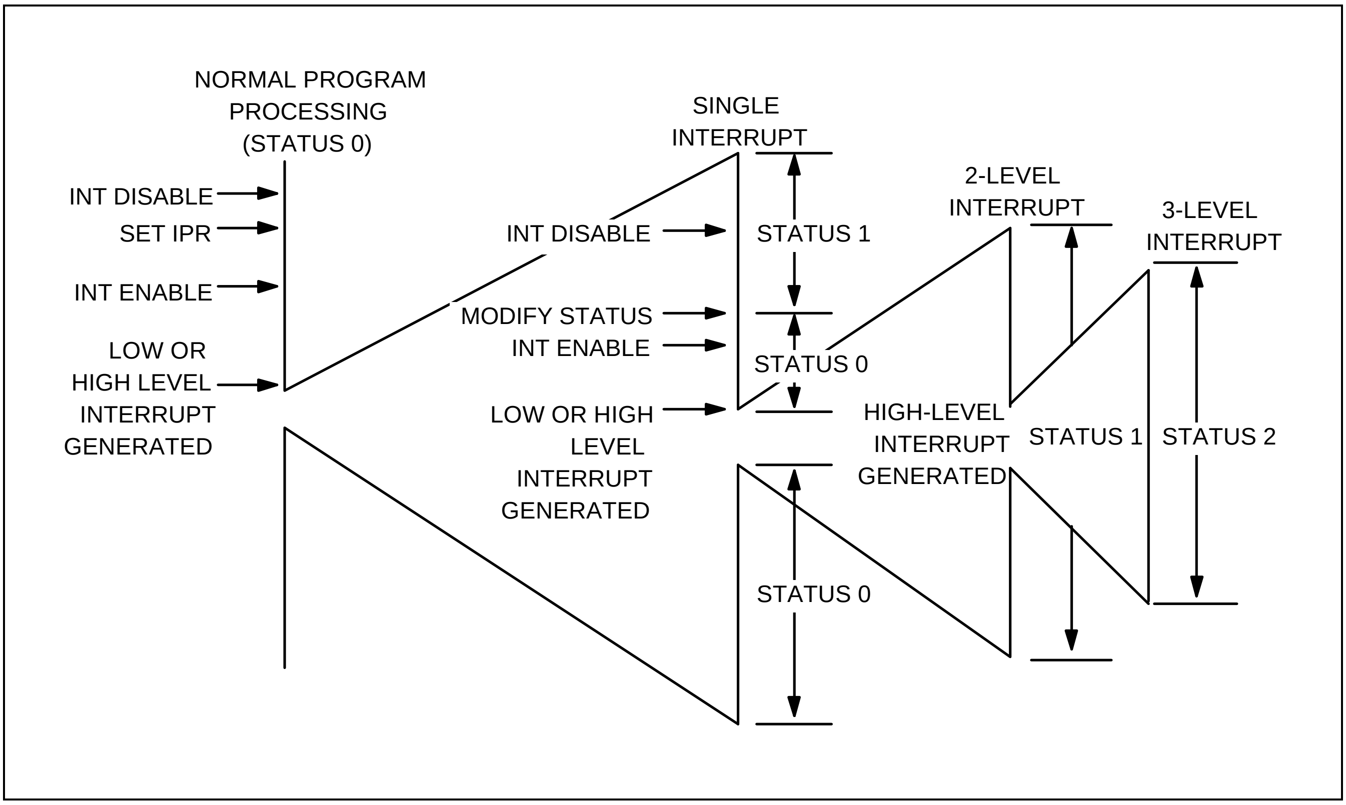


Figure 7-4. Multi-Level Interrupt Handling

INTERRUPT PRIORITY REGISTER (IPR)

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

FB2H	IME	IPR.2	IPR.1	IPR.0

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

Table 7-3. Standard Interrupt Priorities

Interrupt	Default Priority
INTB	1
INT0	2
INT1	3
INTP0	4
INTT0	5

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

Table 7-4. Interrupt Priority Register Settings

IPR.2	IPR.1	IPR.0	Result of IPR Bit Setting
0	0	0	Normal interrupt handling according to default priority settings.
0	0	1	Process INTB 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	Process INTP0 interrupt at highest priority
1	0	1	Process INTT0 interrupt at highest priority
1	1	0	N/A
1	1	1	N/A

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, IMOD1)

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

— 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.

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

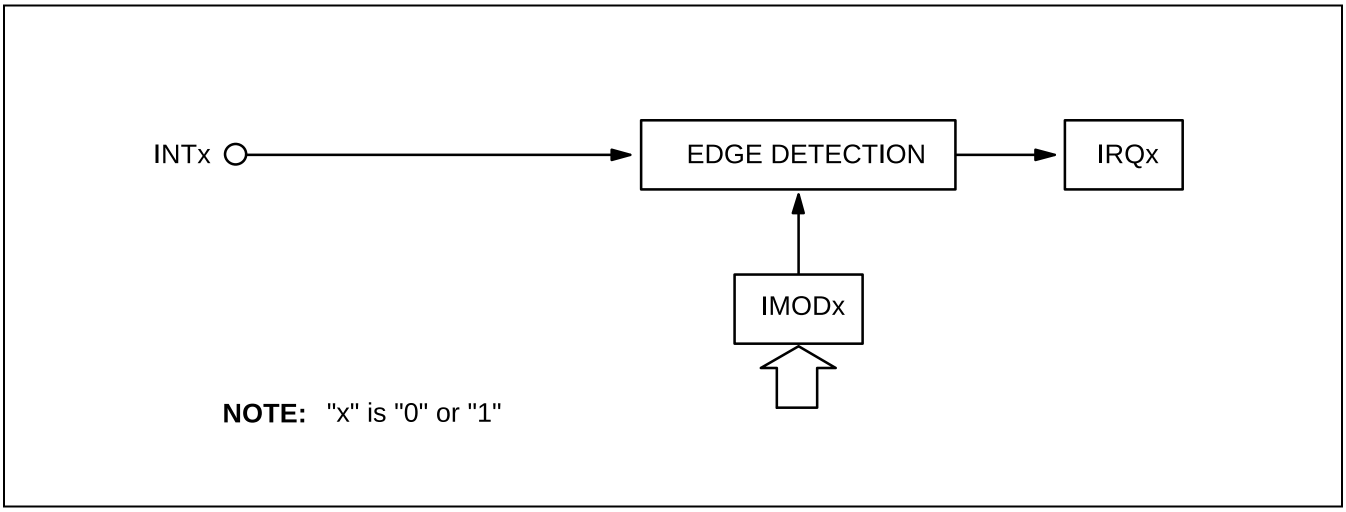
Table 7-5. IMOD0 and IMOD1 Register Organization

IMODx	**"0"**	**"0"**	**IMODx.1**	**IMODx.0**	**Effect of IMOD Settings**
			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"

NOTE: "x" means "0" or "1"

EXTERNAL INTERRUPT 0 AND 1 MODE REGISTERS (Continued)

– You can use INT0/INT1 to release power-down mode



NOTE: "x" is "0" or "1"

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

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

1.Disable all interrupts with a DI instruction.

2.Modify the IMOD0 or IMOD1 register.

3.Clear all relevant interrupt request flags.

4.Enable the interrupt by setting the appropriate IEx flag.

5.Enable all interrupts with an EI instruction.

NOTE: INT0 and INT1 are same in the function .

EXTERNAL INTERRUPT 2 MODE REGISTER (IMOD2)

To generate a key interrupt on a falling edge at KS0-KS7, all KS0-KS7 pins must be configured to input mode. IMOD2 is write-only register that can be written by 4-bit RAM control instruction only. It is mapped to the RAM address FB6H and the reset value of IMOD2 is 0.

FB6H	"0"	"0"	IMOD2.1	IMOD2.0

When a falling edge in any one of KS0–KS7 pins is detected, IRQ2 is set and the release signal of power down mode is generated. INT2, however, does not generate a vector interrupt. Among the pins which were selected as key interrupt, one or more pins which are in input low or output low don't execute a key interrupt function.

Table 7-6. IMOD2 Register Bit Settings

IMOD2	0	0	IMOD2.1	IMOD2.0	Effect of IMOD2 Settings
			0	0	Not available
			0	1	Select falling edge of KS4–KS7
			1	0	Select falling edge of KS2–KS7
			1	1	Select falling edge of KS0–KS7

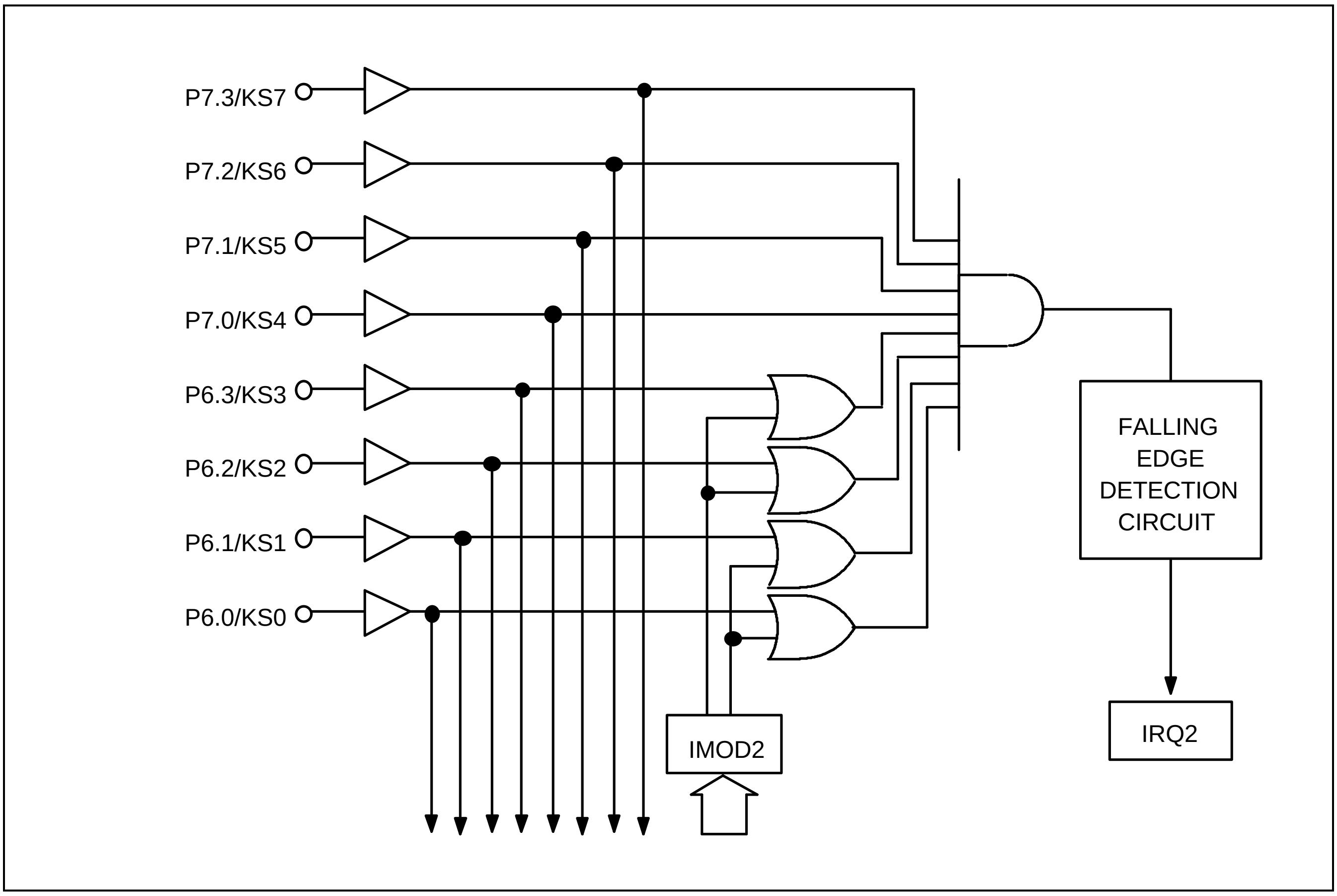


Figure 7-6. Circuit Diagram for INT2

☞ PROGRAMMING TIP — Using INT2 as a Key Input Interrupt

When the INT2 interrupt is used as a key entry interrupt, the selected key interrupt source pin must be set to input:

1.When KS0–KS3 are selected (four pins):

```
        BITS     EMB
        SMB      15
        LD       A,#3H
        LD       IMOD2,A        ;  (IMOD2) ← #3H, KS0-KS3 falling edge select
        LD       EA,#00H
        LD       PMG1,EA        ;  P6 ← input mode
        LD       A,#4H
        LD       PUMOD0,A       ;  Enable P6 pull-up resistors
```

INTERRUPT FLAGS

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

Interrupt Master Enable Flag (IME)

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

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

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

Interrupt Enable Flags (IEx)

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

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

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

Address	Bit 3	Bit 2	Bit 1	Bit 0
FB8H	"U"	"U"	IEB	IRQB
FBAH	"U"	"U"	IEW	IRQW
FBCH	"U"	"U"	IET0	IRQT0
FBDH	"U"	"U"	IEP0	IRQP0
FBEH	IE1	IRQ1	IE0	IRQ0
FBFH	"U"	"U"	IE2	IRQ2

NOTES:

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

Interrupt Request Flags (IRQx)

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

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

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

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
INT0	E	Rising or falling edge detected at INT0 pin	2	IRQ0
INT1	E	Rising or falling edge detected at INT1 pin	3	IRQ1
INTP0	E	Falling edge detected at K0–K3 (P0.0 – P0.3)	4	IRQP0
INTT0	I	Signals for TCNT0 and TREF0 registers match	5	IRQT0
INT2 (NOTE)	E	Falling edge is detected at any of the KS0–KS7 pins	–	IRQ2
INTW	I	Time interval of 0.5 s or 3.19 ms	–	IRQW

NOTE: Refer to page 7-10, 7-11.

NOTES

8 POWER-DOWN

OVERVIEW

The KS57C21408/C21418 microcontroller has two power-down modes to reduce power consumption: idle and stop. Idle mode is initiated by the IDLE instruction and stop mode by the instruction STOP. (Several NOP instructions must always follow an IDLE or STOP instruction in a program.) In idle mode, the CPU clock stops while peripherals and the oscillation source continue to operate normally.

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

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

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

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

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

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

NOTE

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

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

Operation	Stop Mode	Idle Mode
Instruction	STOP	IDLE
System clock status	STOP mode can be used only if the main-system clock is selected as system clock (CPU clock)	IDLE mode can be used if the main-system clock or sub-system clock is selected as system clock (CPU clock)
Clock oscillator	Main-system clock oscillation stops	Only CPU clock oscillation stops (main and sub-system clock oscillation continues)
Basic timer	Basic timer stops	Basic timer operates (with IRQB set at each reference interval)
Timer/counter 0	Operates only if TCL0 is selected as the counter clock	Timer/counter 0 operates
Watch timer	Operates only if sub-system clock (fxt) is selected as the counter clock	Watch timer operates
LCD controller	Operates only if a sub-system clock is selected as LCDCK	LCD controller operates
External interrupts	INT0,INT1,INT2 and INTP0 are acknowledged.	INT0,INT1,INT2 and INTP0 are acknowledged.
CPU	All CPU operations are disabled	All CPU operations are disabled
Mode release signal	Interrupt request signals are enable by an interrupt enable flag or by RESET input.	Interrupt request signals are enable by an interrupt enable flag or by RESET input.

Table 8-2. System Operating Mode Comparison

Mode	Condition	STOP/IDLE Mode Start Method	Current Consumption
Main operating mode	Main oscillator runs. Sub oscillator runs System clock is the main oscillation clock.	–	A
Main Idle mode	Main oscillator runs. Sub oscillator runs 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

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

IDLE MODE TIMING DIAGRAMS

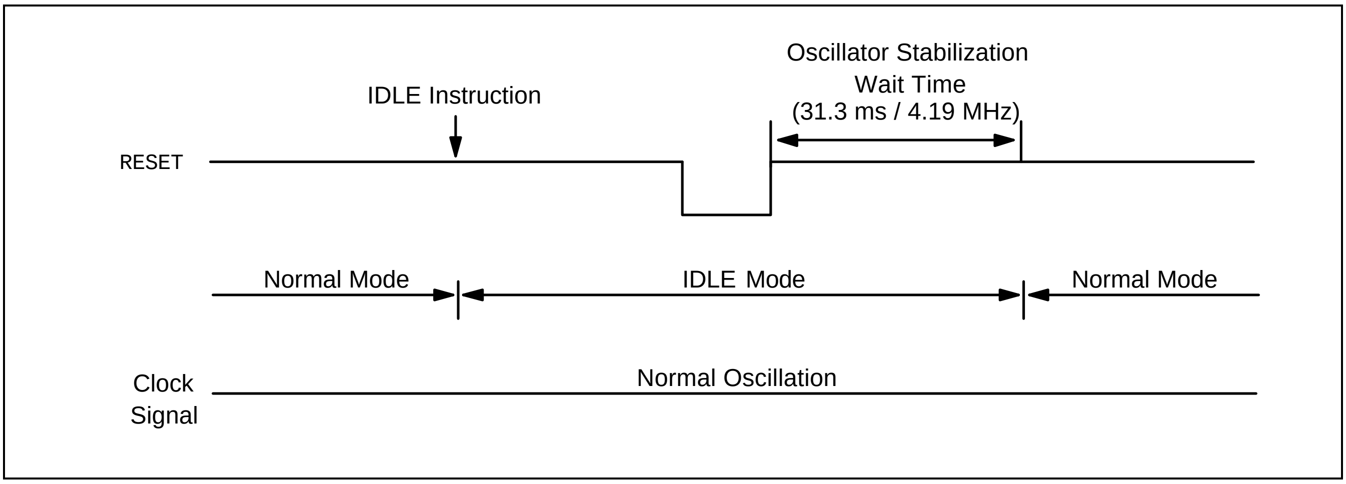


Figure 8-1. Timing When Idle Mode is Released by RESET

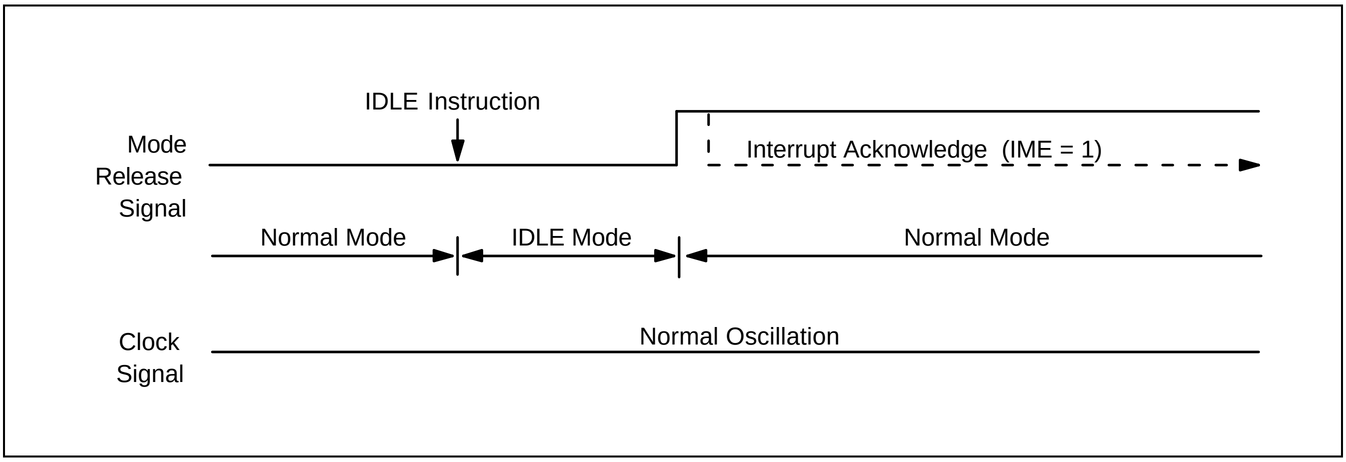


Figure 8-2. Timing When Idle Mode is Released by an Interrupt

STOP MODE TIMING DIAGRAMS

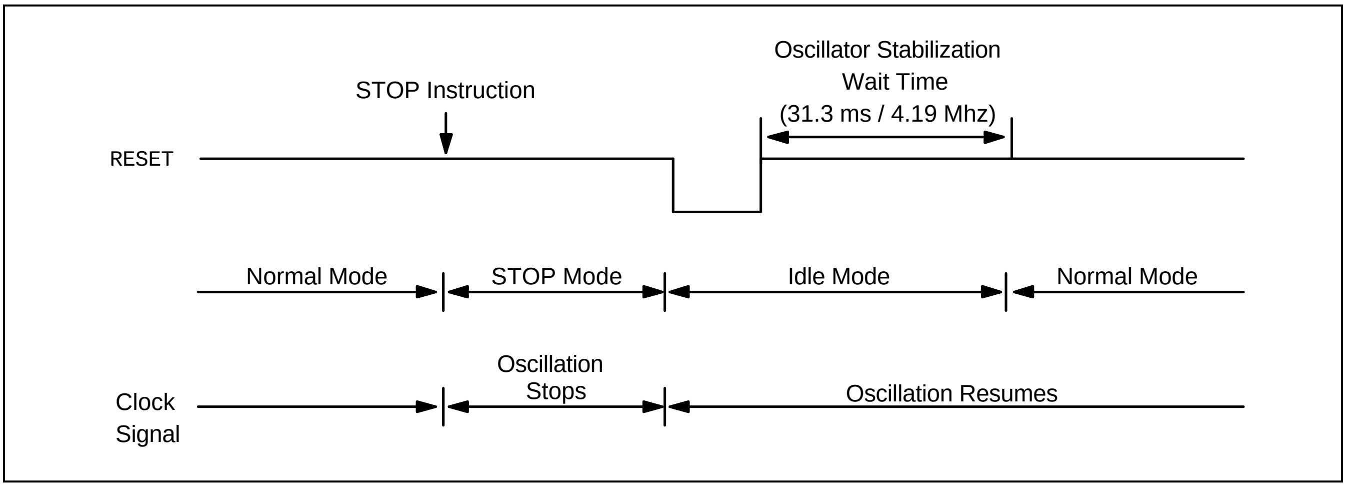


Figure 8-3. Timing When Stop Mode is Released by RESET

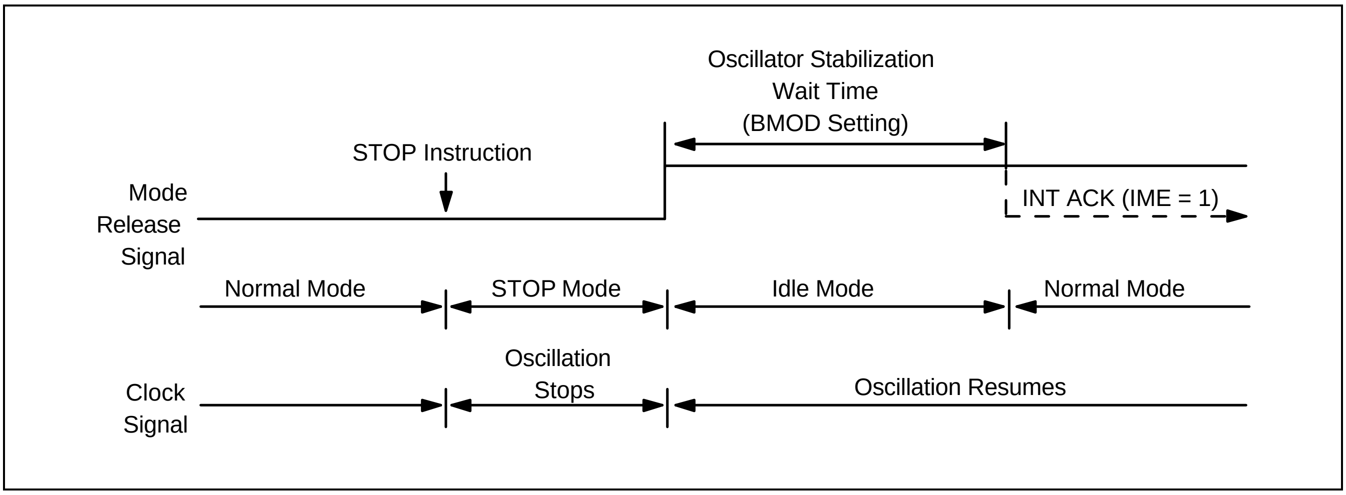


Figure 8-4. Timing When Stop Mode is Released by an Interrupt

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

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

```
KEYCLK    DI
          CALL      MA2SUB              ;  Main-system  clock →  sub-system clock switch subroutine
          SMB       15
          LD        EA,#00H
          LD        P4,EA               ;  All key strobe outputs to low level
          LD        A,#3H
          LD        IMOD2,A             ;  Select KS0-KS7 enable
          SMB       0
          BITR      IRQW
          BITR      IRQ2
          BITS      IEW
          BITS      IE2

CLKS1     CALL      WATDIS              ;  Execute clock and display changing subroutine
          BTSTZ     IRQ2
          JR        CIDLE
          CALL      SUB2MA              ;  Sub-system clock  →  main-system clock switch subroutine
          EI
          RET

CIDLE     IDLE                          ;  Engage idle mode
          NOP
          NOP
          NOP
          JPS       CLKS1
```

RECOMMENDED CONNECTIONS FOR UNUSED PINS

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

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

Pin/Share Pin Names	Recommended Connection
P0.0 P0.1 P0.2 P0.3	Input mode: Connect to V_{DD}
P1.0/ INT0 – P1.1/INT1	Connect to V_{DD}
P2.0 / BUZ P2.1 / CLO P4.0 /TCL0,P4.1/TCLO0, P4.2 P5.0 - P5.3 P6.0 / KS0 - P6.3 / KS3 P7.0 / KS4 - P7.3 / KS7	Input mode: Connect to V_{DD} Output mode: No connection
SEG0-SEG59 COM0-COM8	No connection
TEST	Connect to V_{SS}

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

— Carry flag

— Data memory values

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

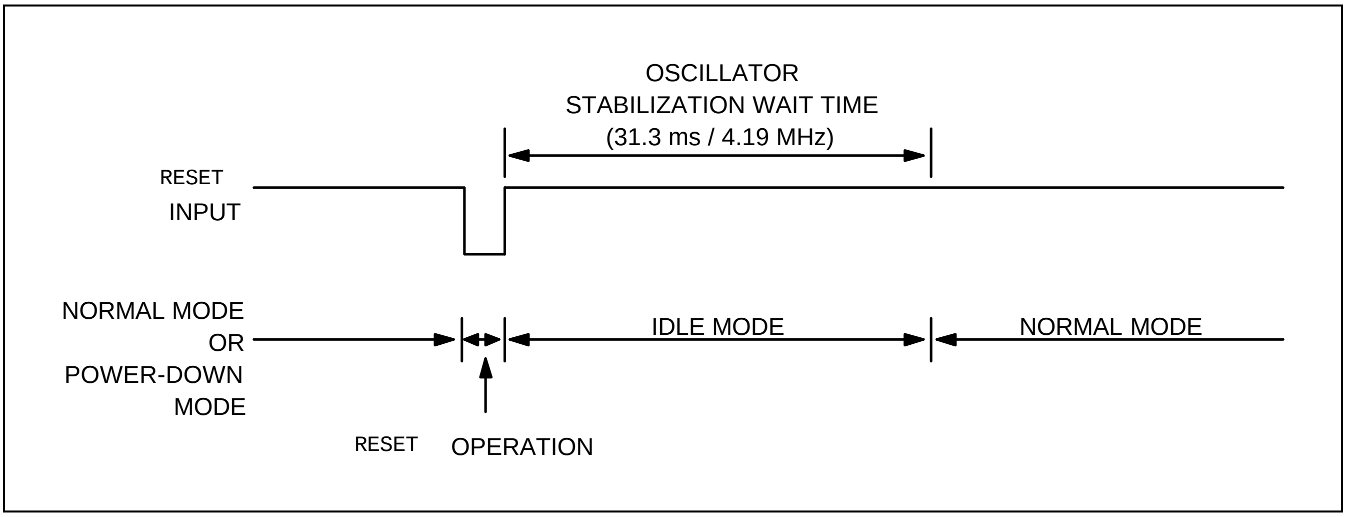


Figure 9-1. Timing for Oscillation Stabilization after RESET

HARDWARE REGISTER VALUES AFTER RESET

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

Table 9-1. Hardware Register Values After RESET

Hardware Component or Subcomponent	If RESET Occurs During Power down Mode	If RESET Occurs during normal operating
Program counter (PC)	Lower six bits of address 0000H are transferred to PC13–8, and the contents of 0001H to PC7–0.	Lower six bits of address 0000H are transferred to PC13–8, and the contents of 0001H to PC7–0.
Program Status Word (PSW):		
Carry flag (C)	Retained	Undefined
Skip flag (SC0-SC2)	0	0
Interrupt status flags (IS0, IS1)	0	0
Bank enable flags (EMB, ERB)	Bit 6 of address 0000H in program memory is transferred to the ERB flag, and bit 7 of the address to the EMB flag.	Bit 6 of address 0000H in program memory is transferred to the ERB flag, and bit 7 of the address to the EMB flag.
Stack pointer (SP)	Undefined	Undefined
Data Memory (RAM):		
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
Bank1 page select register (PASR)	0	0
Key scan register (KSR0-KSR3)	0	0
Clocks:		
Power control register (PCON)	0	0
Clock output mode register (CLMOD)	0	0
System clock mode register (SCMOD)	0	0
Interrupts:		
Interrupt request flags (IRQx)	0	0
Interrupt enable flags (IEx)	0	0
Interrupt priority flag (IPR)	0	0
Interrupt master enable flag (IME)	0	0
INT0 mode register (IMOD0)	0	0
INT1 mode register (IMOD1)	0	0
INT2 mode register (IMOD2)	0	0

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

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

Hardware Component or Subcomponent	If RESET Occurs During Power down Mode	If RESET Occurs during normal operation
I/O Ports:		
Output buffers	Off	Off
Output latches	0	0
Port mode flags (PM)	0	0
Pull-up resistor mode reg (PUMOD0)	0	0
Basic Timer:		
Count register (BCNT)	Undefined	Undefined
Mode register (BMOD)	0	0
Timer/Counter 0:		
Count register (TCNT0)	0	0
Reference register (TREF0)	FFH	FFH
Mode register (TMOD0)	0	0
Output enable flag (TOE0)	0	0
Watch Timer:		
Watch timer mode register (WMOD)	0	0
LCD Driver/Controller:		
LCD mode register (LMOD0/1)	0	0
Display data memory	Values retained	Undefined
Output buffers	Off	Off

NOTES

10 I/O PORTS

OVERVIEW

The KS57C21408/C21418/P21408 has 39 I/O Lines. There are total of 6 input pins, 16 output pins, 17 configurable I/O pins, for a total number of 39 pins.

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. PM flags are stored in one 8-bit, 4-bit register and are addressable by 8-bit, 4-bit write instructions respectively.

Output Ports 8

Output ports 8 consists of 16 pins that can be used either for LCD segment data output or for normal 1-bit output. that is used for key strobe signal output. refer to LCD part for detailed explanation.

Pull-Up Resistor Mode Register (PUMOD0)

The pull-up resistor mode register (PUMOD0) is 8-bit register used to assign internal pull-up resistors by software to specific I/O ports.

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

PUMOD0 is addressable by 8-bit write instructions only. RESET clears PUMOD0 register values to logic zero, automatically disconnecting all software-assignable port pull-up resistors.

N-channel Open-drain Mode Register (PNE1)

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

Table 10-1. I/O Port Overview

Port	I/O	Pins	Pin Names	Address	Function Description
0	I	4	P0.0-P0.3 (K0 - K3)	FF0H	4-bit input port only 1-bit and 4-bit read and test are possible. Interrupt generation at P0 falling edge.
1	I	2	P1.0-P1.1	FF1H	2-bit input port. 1-bit and 4-bit read and test are possible. 2-bit pull-up resistors are assignable by software.

Table 10-1. I/O Port Overview (Continued)

Port	I/O	Pins	Pin Names	Address	Function Description
2	I/O	2	P2.0-P2.1	FF2H	2-bit I/O ports. 1-bit and 4-bit read/wrtie, and test are possible. 2 bits pull-up resistors are assignable by software.
4, 5	I/O	7	P4.0-P4.2 P5.0-P5.3	FF4H FF5H	4-bit I/O port. 1, 4, and 8-bit read/write, and test are possible. 4-bit unit pins are software configurable as input or output. Individual pins are software configurable as open-drain or push-pull output. 4-bit pull-up resistors are assignable by software and pull-up resistors are automatically disabled for output pins.
6	I/O	4	P6.0-P6.3	FF6H	4-bit I/O port. Each individual pin can be specified as input or output. 1-bit and 4-bits read/write and test are possible; 4-bits pull-up resistors are assignable by software.
7	I/O	4	P7.0-P7.3	FF7H	4-bit I/O Port. 1-bit and 4-bit read/write and test are possible. 4-bit pull-up resistors are assignable by software. Port6 and Port7 can be paired to enable 8-bits 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 P0.1 LDB C,P1.3 LD A,P6 LD EA,P4	Input or test data at each pin	Input or test data at output latch
1-bit output	BITR P2.0	Output latch contents undefined	Output pin status is modified
4-bit output 8-bit output	LD P5,A LD P6,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. PM flags are stored in one 8-bit, 4-bit registers and are addressable by 8-bit, 4-bit write instructions refectively.

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

When a PM flag is "0", the port is set to input mode; when it is "1", the port is enabled for output. RESET clears all port mode flags to 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	Bit 2	Bit 1	Bit 0
PMG1	FE8H	PM2.1	PM2.0	"0"	"0"
	FE9H	PM6.3	PM6.2	PM6.1	PM6.0
PMG2	FEAH	PM7	"0"	PM5	PM4

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

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

Configure P6 as an output port:

```
        BITS      EMB
        SMB       15
        LD        EA,#0F0H
        LD        PMG1,EA             ;  P6 ← Output
```

PULL-UP RESISTOR MODE REGISTER (PUMOD0)

The pull-up resistor mode register (PUMOD0) is an 8-bit register used to assign internal pull-up resistors by software to specific I/O ports.

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

RESET clears PUMOD0 register values to logic zero, automatically disconnecting all software-assignable port pull-up resistors.

Table 10-4. Pull-Up Resistor Mode Register (PUMOD0) Organization

Bit Name	PUMOD0 function	
PUMOD0.7	0	Disconnect port7 pull-up resistor
	1	Connect port7 pull-up resistor
PUMOD0.6	0	Disconnect port6 pull-up resistor
	1	Connect port6 pull-up resistor
PUMOD0.5	0	Disconnect port5 pull-up resistor
	1	Connect port5 pull-up resistor
PUMOD0.4	0	Disconnect port4 pull-up resistor
	1	Connect port4 pull-up resistor
PUMOD0.3	0	Always logic zero
PUMOD0.2	0	Disconnect port2 pull-up resistor
	1	Connect port2 pull-up resistor
PUMOD0.1	0	Disconnect port1 pull-up resistor
	1	Connect port1 pull-up resistor
PUMOD0.0	0	Always logic zero

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

P6 enable pull-up resistors.

```
BITS    EMB
SMB     15
LD      EA,#40H
LD      PUMOD0,EA        ;   P6 pull-up resistor enable
```

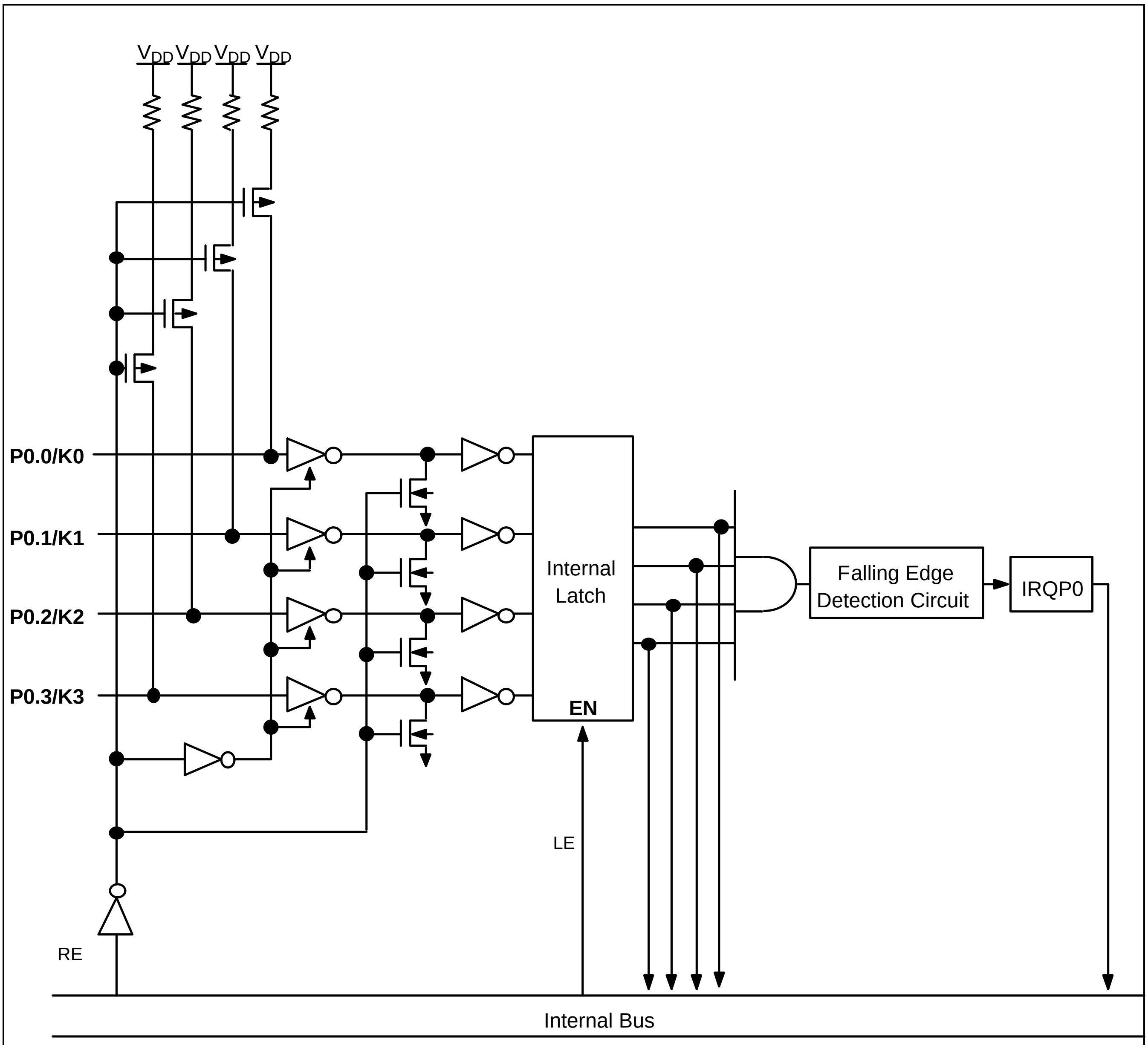
N-channel Open-drain Mode Register (PNE1)

PNE1

Address	Bit3	Bit2	Bit1	Bit0
FDAH	"0"	PNE4.2	PNE4.1	PNE4.0
FDBH	PNE5.3	PNE5.2	PNE5.1	PNE5.0

The n-channel open-drain mode register, PNE1, is used to configure port4 and 5 to n-channel open-drain outputs or as push-pull outputs. When a bit in the PNE1 register is set to one, the corresponding output pin is configured to n-channel open-drain; when set to ”0", the output pin is configured to push-pull.
The PNE register consists of an 8-bit, as shown above. PNE1 can be addressed by 8-bit write instructions only.

PORT 0 CIRCUIT DIAGRAM

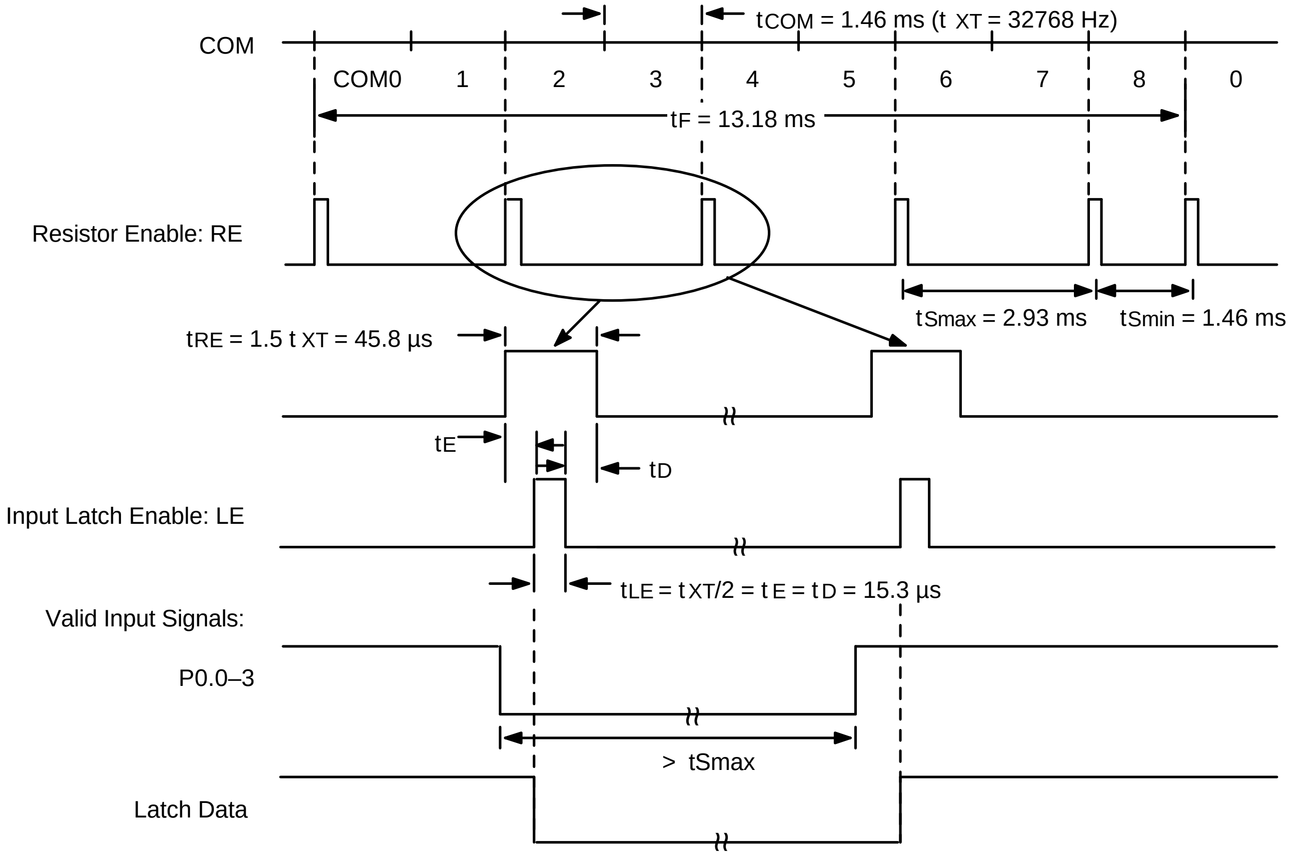


RE: Resistor Enable LE: Latch

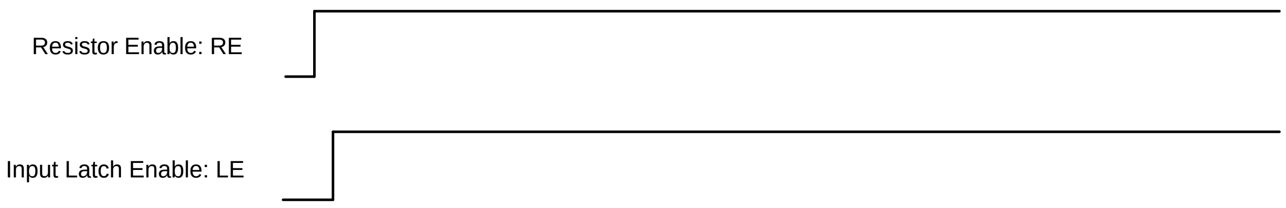
NOTE : The pull-up resistors enable signal(RE) is automatically generated and synchronized to LCD common signals.

Figure 10-1. Port 0 Circuit Diagram

1. **When LMOD0 is 4 or 5 and LMOD1.0 are set to “0” (normal display signal output), RE and LE signals are generated as below:**



2. When LMOD.3 or LMOD1.0 is set to 1, (key check status), RE and LE signals keep high state as below:



NOTE: The pull-up resistors of P0.0-P0.3 are always enabled.
The input signals (P0.0-P0.3) cannot be latched but connected to internal bus directly.

Figure 10-2. Port 0 Signal Waveform

PORT 1 CIRCUIT DIAGRAM

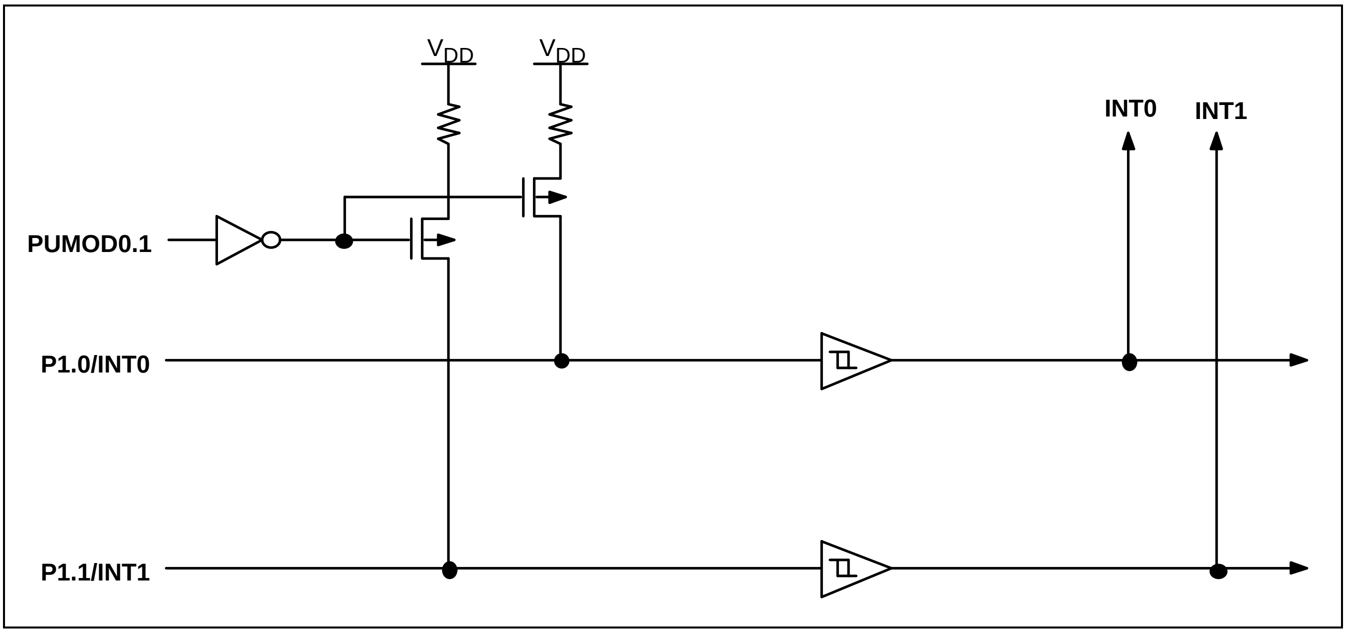
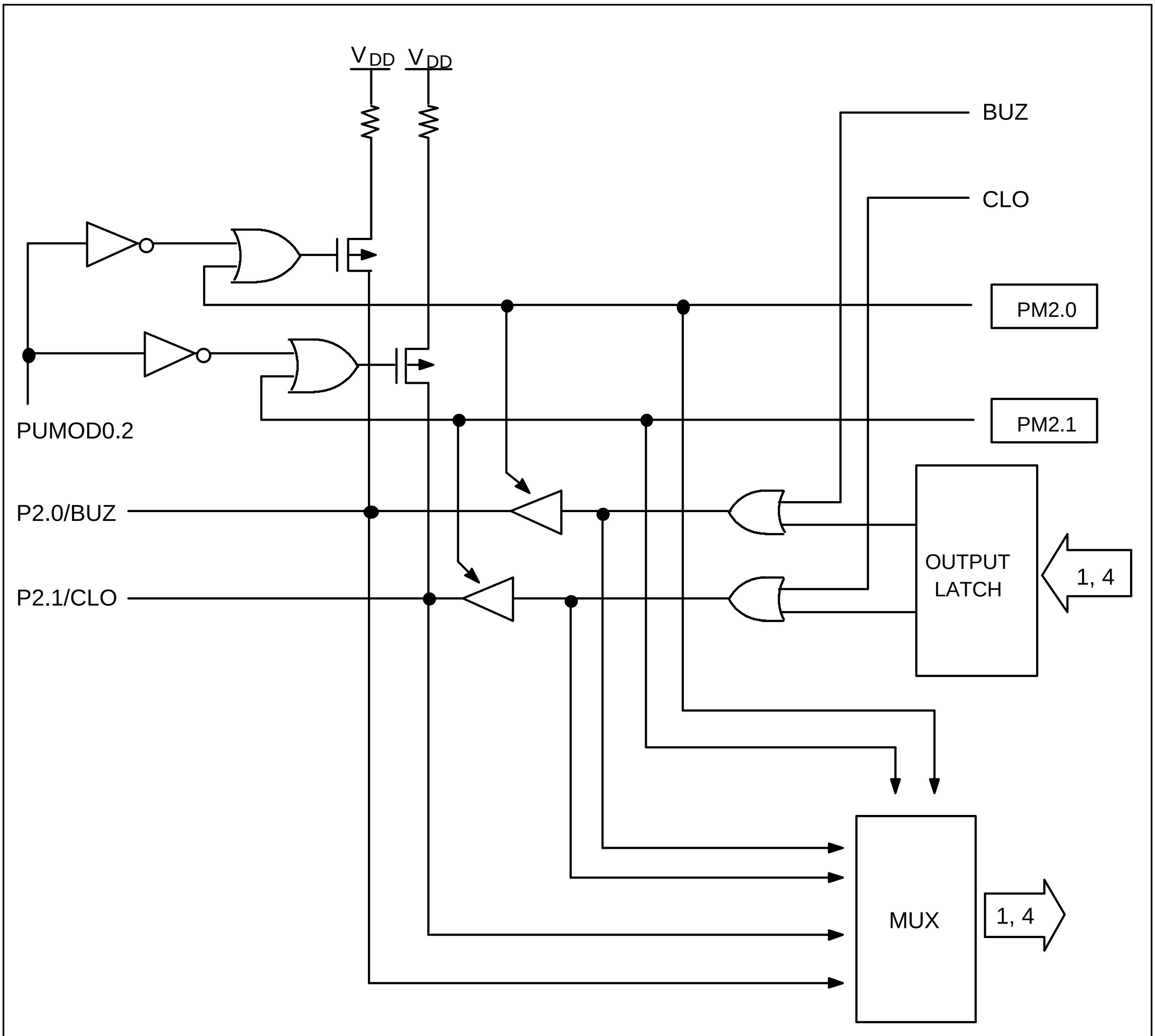


Figure 10-3. Port 1 Circuit Diagram

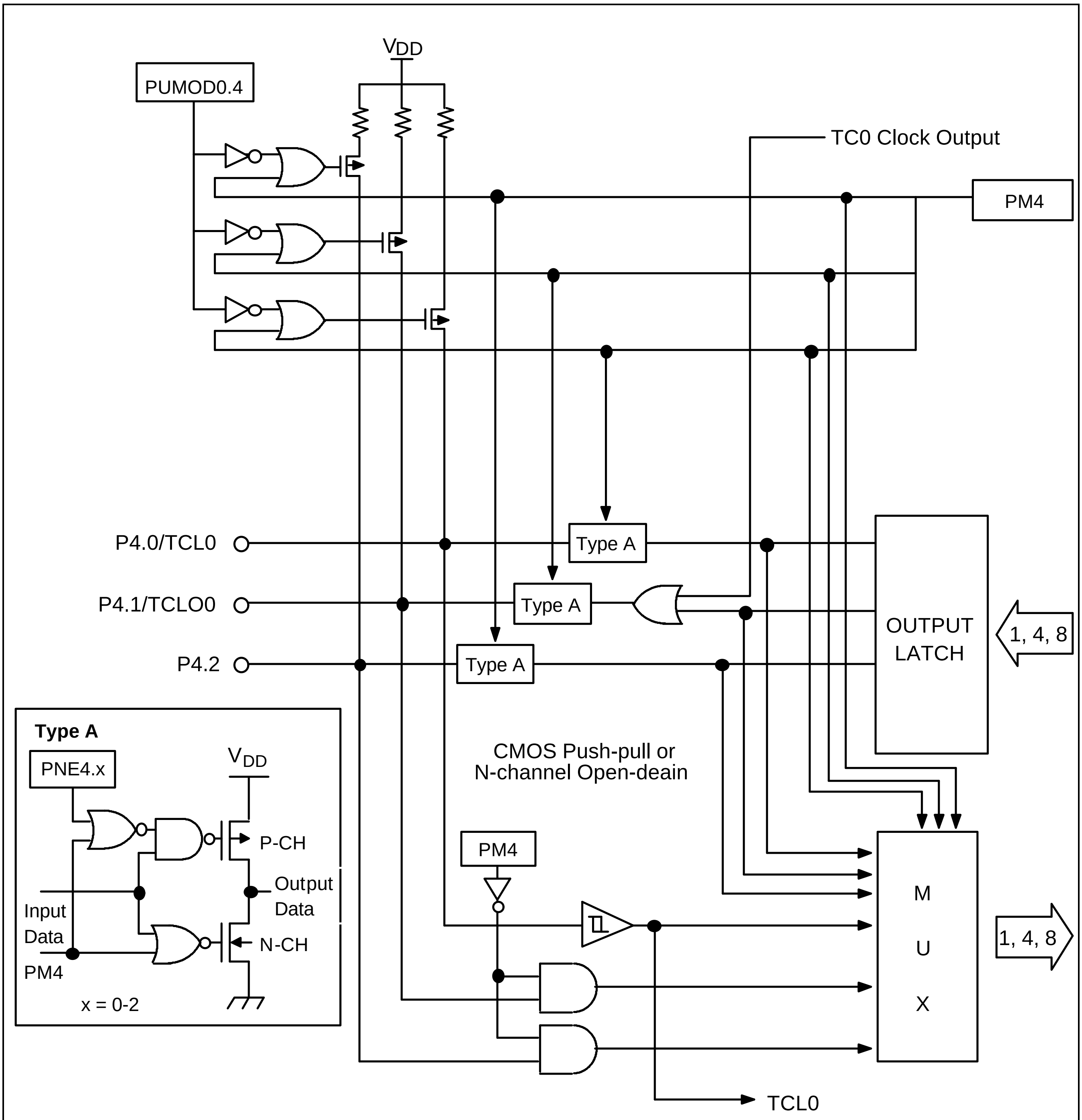
PORT 2 CIRCUIT DIAGRAM



NOTE: When a port pin acts as an output, its pull-up resistor is automatically disabled even though the port's pull-up resistor is enabled by bit settings to the pull-up resistor mode register (PUMOD0).

Figure 10-4. Port 2 Circuit diagram

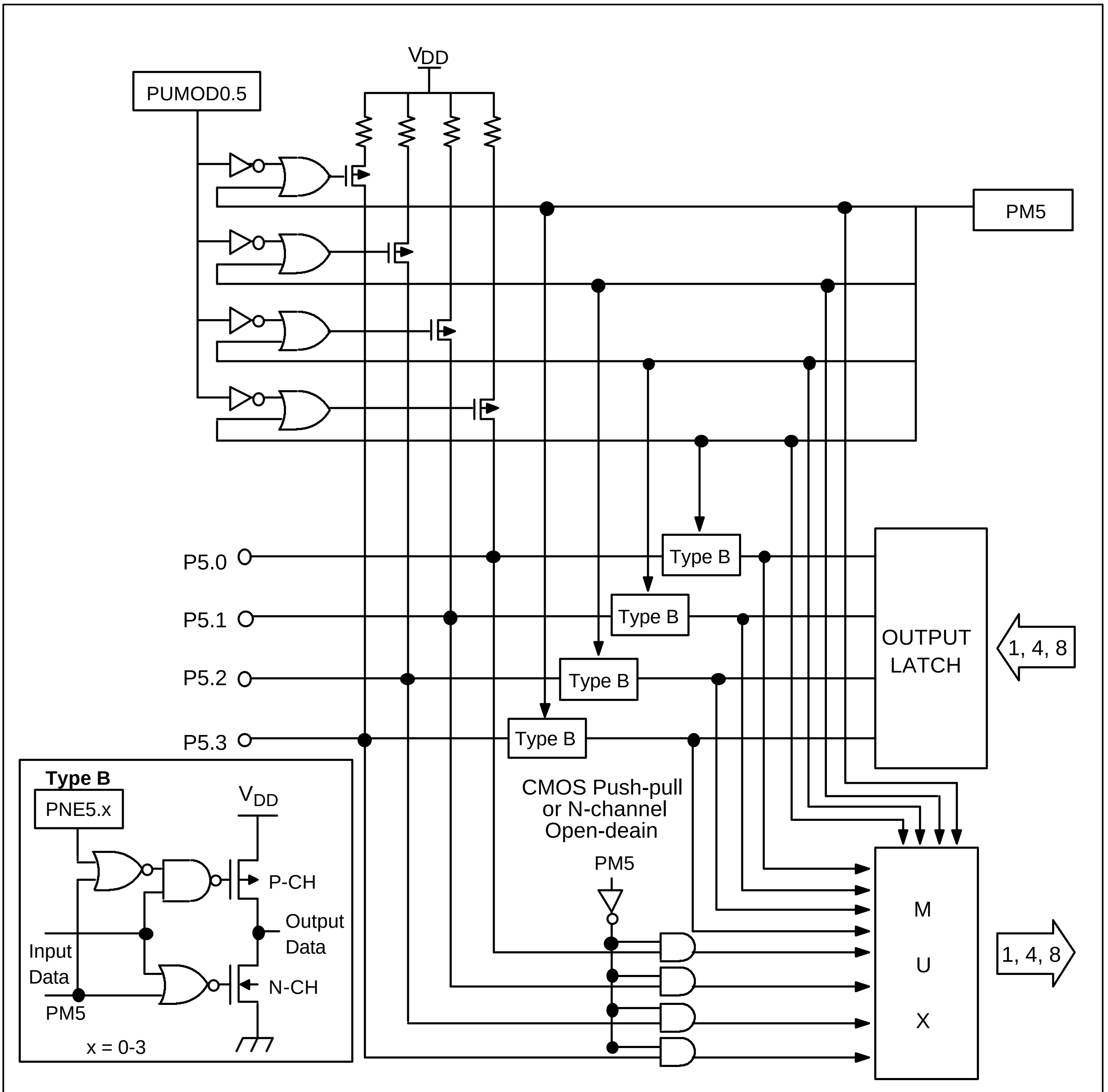
PORT 4 CIRCUIT DIAGRAM



NOTE: When a port pin serves as an output, its pull-up resistor is automatically disabled, even though the port's pull-up resistor is enabled by bit settings to the pull-up resistor mode register (PUMOD0).

Figure 10-5. Port 4 Circuit Diagram

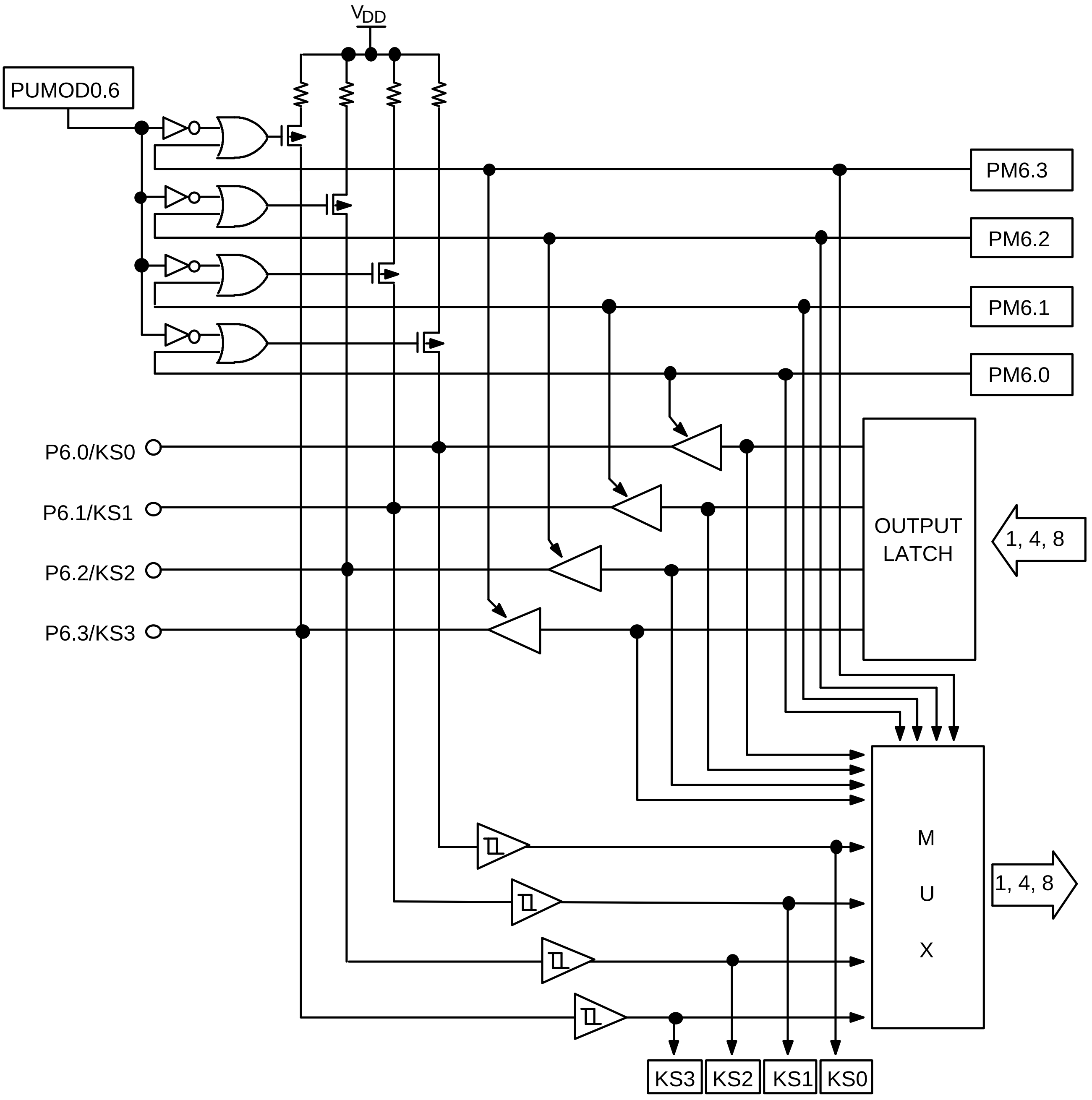
PORT 5 CIRCUIT DIAGRAM



NOTE: When a port pin serves as an output, its pull-up resistor is automatically disabled, even the though the port's pull-up resistor is enabled by bit settings to the pull-up resistor mode register (PUMOD0).

Figure 10-6. Port 5 Circuit Diagram

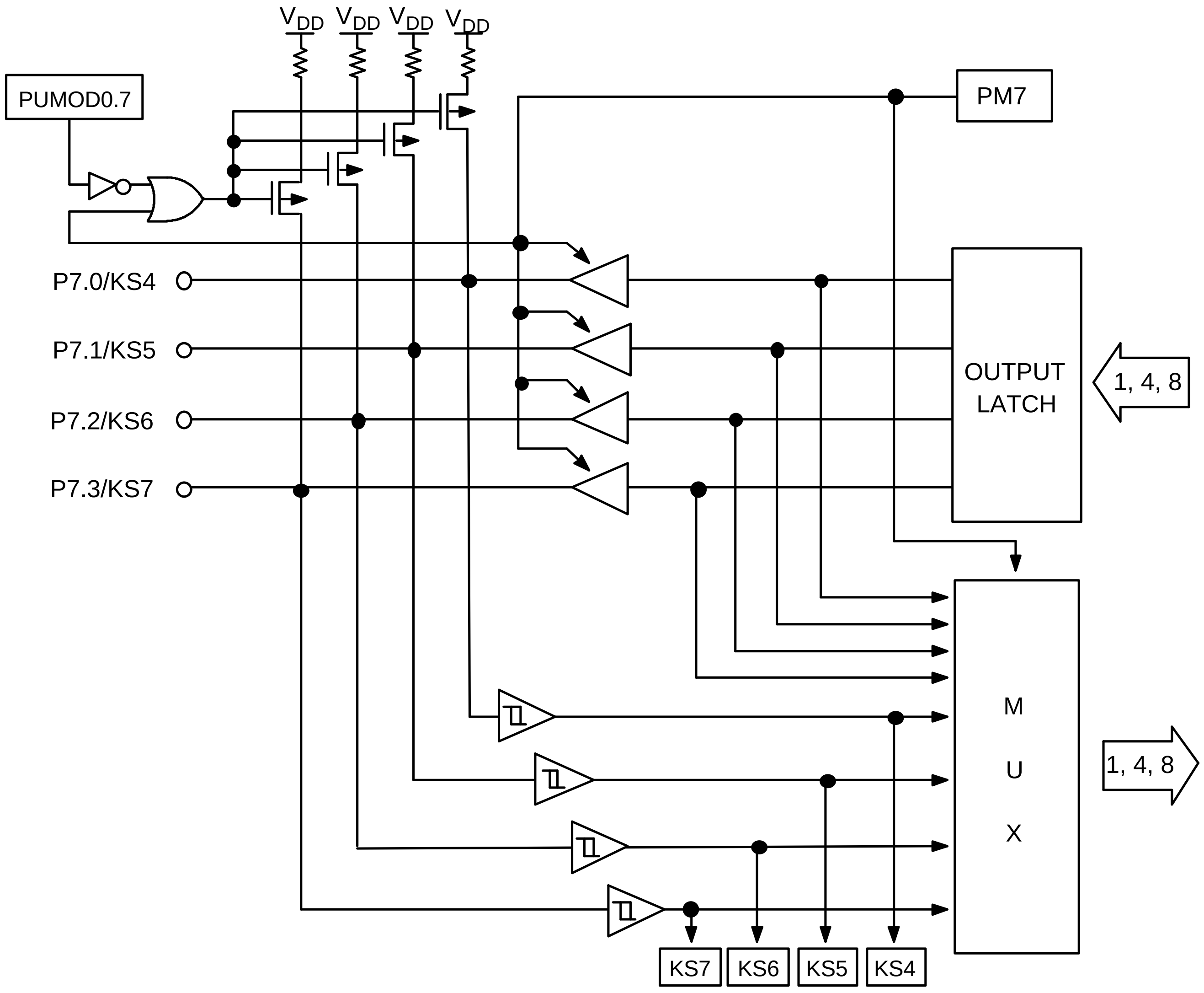
PORT 6 CIRCUIT DIAGRAM



NOTE: When a port pin acts as an output, its pull-up resistor is automatically disabled, even though the port's pull-up resistor is enabled by bit settings to the pull-up resistor mode register (PUMOD0).

Figure 10-7. Port 6 Circuit Diagram

PORT 7 CIRCUIT DIAGRAM



NOTE: When a port pin acts as an output, its pull-up resistor is automatically disabled, even though the port's pull-up resistor is enabled by bit settings to the pull-up resistor mode register (PUMOD0).

Figure 10-8. Port 7 Circuit Diagram

11 TIMERS and TIMER/COUNTER 0

OVERVIEW

The KS57C21408/C21418/P21418 microcontroller has two timers, and one timer/counter 0 modules:

— 8-bit basic timer (BT)

— 8-bit timer/counter 0 (TC0)

— Watch timer (WT)

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

The 8-bit timer/counter 0 (TC0) is a programmable timer/counter that is used primarily for event counting and for clock frequency modification and output.

The watch timer (WT) module consists of an 8-bit watch timer mode register, a clock selector, and a frequency divider circuit. Watch timer functions include real-time and watch-time measurement, 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 three functional components:

— Clock selector logic

— 4-bit mode register (BMOD)

— 8-bit counter register (BCNT)

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

Interval Timer Function

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

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

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

Oscillation Stabilization Interval Control

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

Table 11-1. Basic Timer Register Overview

Register Name	Type	Description	Size	RAM Address	Addressing Mode	Reset Value
BMOD	Control	Controls the clock frequency (mode) of the basic timer; also, the oscillation stabilization interval after power-down mode release or RESET	4-bit	F85H	4-bit write-only; BMOD.3: 1-bit writeable	"0"
BCNT	Counter	Counts clock pulses matching the BMOD frequency setting	8-bit	F86H–F87H	8-bit read-only	U *

* 'U' means the value is undetermined after a RESET.

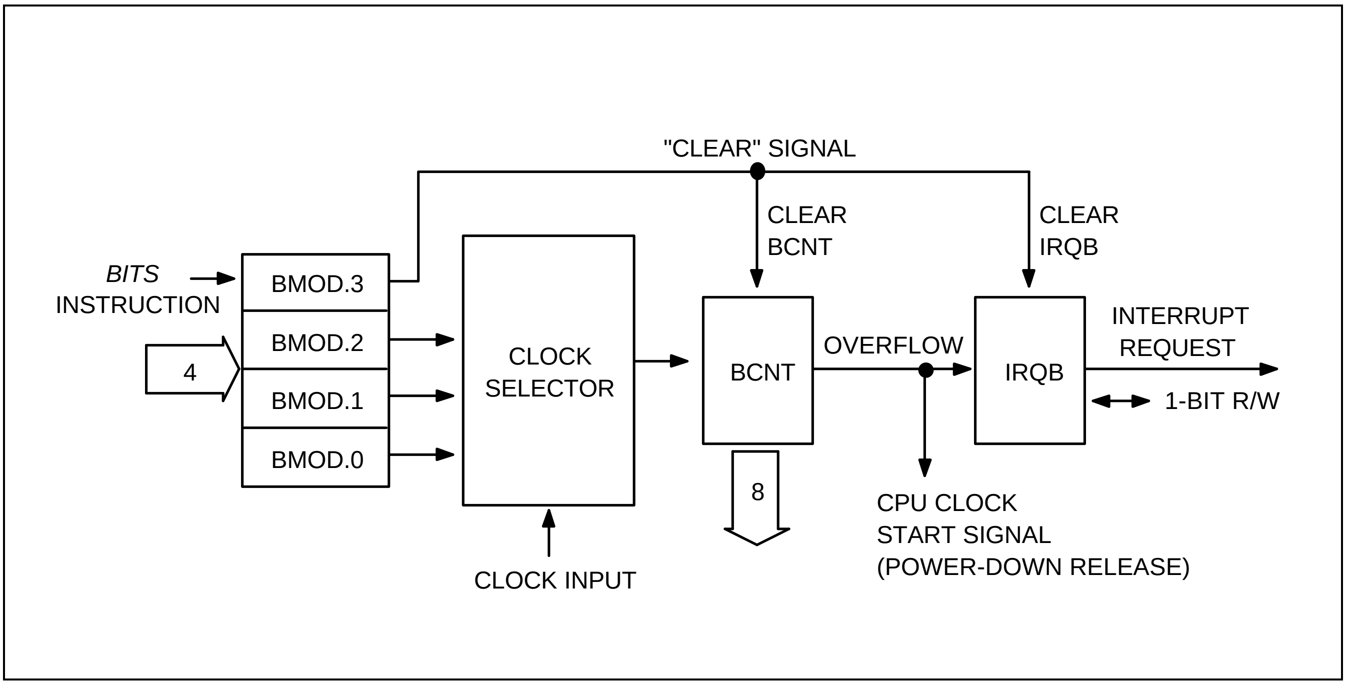


Figure 11-1. Basic Timer Circuit Diagram

BASIC TIMER MODE REGISTER (BMOD)

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

— Restart the basic timer;

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

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

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

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

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

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

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

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

NOTES:

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

BASIC TIMER COUNTER (BCNT)

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

When BCNT has 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 incoming clock signals.

NOTE

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

BASIC TIMER OPERATION SEQUENCE

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

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

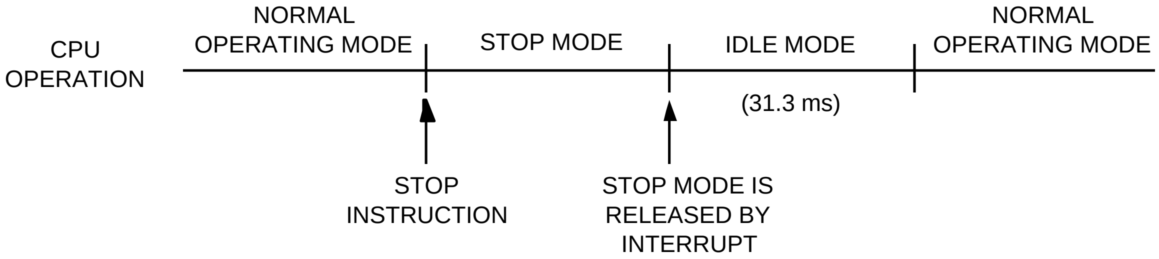
☞ PROGRAMMING TIP — Using the Basic Timer

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

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

2. When stop mode is released by an interrupt, set the oscillation stabilization interval to 31.3ms (at 4.19MHz):

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



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

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

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

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

8-BIT TIMER/COUNTER 0 (TC0)

OVERVIEW

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

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

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

TC0 FUNCTION SUMMARY

8-bit programmable timer	Generates interrupts at specific time intervals based on the selected clock frequency.
External event counter	Counts various system "events" based on edge detection of external clock signals at the TC0 input pin, TCL0. To start the event counting operation, TMOD0.2 is set to "1" and TMOD0.6 is cleared to "0".
Arbitrary frequency output	Outputs selectable clock frequencies to the TC0 output pin, TCLO0.
External signal divider	Divides the frequency of an incoming external clock signal according to a modifiable reference value (TREF0), and outputs the modified frequency to the TCLO0 pin.

TC0 COMPONENT SUMMARY

Mode register (TMOD0) Activates the timer/counter and selects the internal clock frequency or the external clock source at the TCL0 pin.

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

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

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

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

Output enable flag (TOE0) Must be set to logic one before the contents of the TOL0 latch can be output to TCLO0.

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

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

Table 11-3. TC0 Register Overview

Register Name	Type	Description	Size	RAM Address	Addressing Mode	Reset Value
TMOD0	Control	Controls TC0 enable/disable (bit 2); clears and resumes counting operation (bit 3); sets input clock and clock frequency (bits 6-4)	8-bit	F90H-F91H	8-bit write-only; (TMOD0.3 is also 1-bit writeable)	"0"
TCNT0	Counter	Counts clock pulses matching the TMOD0 frequency setting	8-bit	F94H-F95H	8-bit read-only	"0"
TREF0	Reference	Stores reference value for the timer/counter 0 interval setting	8-bit	F96H-F97H	8-bit write-only	FFH
TOE0	Flag	Controls timer/counter 0 output to the TCLO0 pin	1-bit	F92H.2	1-bit and 4-bit read/write	"0"

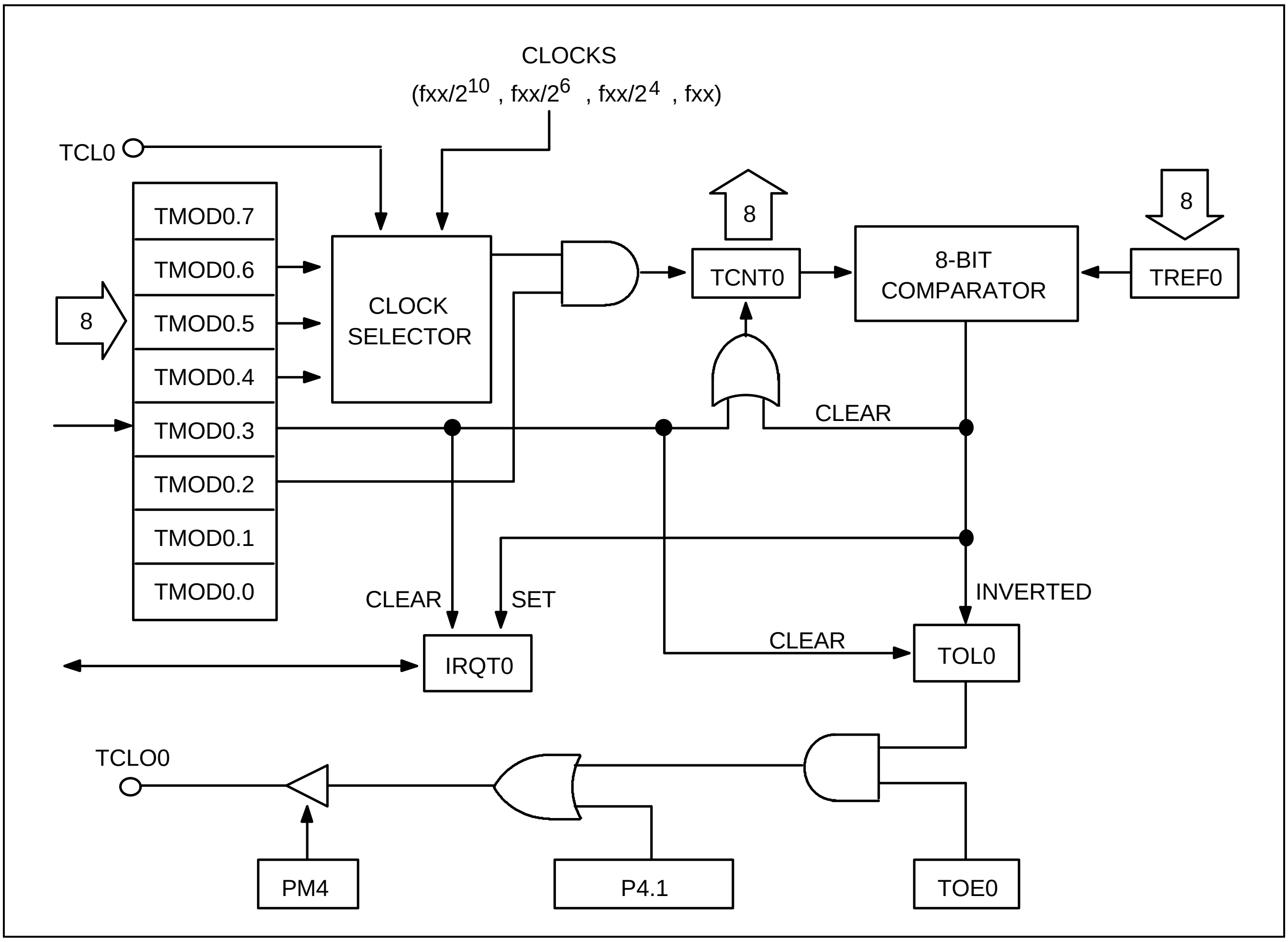


Figure 11-2. TC0 Circuit Diagram

TC0 ENABLE/DISABLE PROCEDURE

Enable Timer/Counter 0

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

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

Disable Timer/Counter 0

- Set TMOD0.2 to logic zero

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

TC0 PROGRAMMABLE TIMER/COUNTER FUNCTION

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

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

TC0 OPERATION SEQUENCE

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

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

TC0 EVENT COUNTER FUNCTION

Timer/counter 0 can monitor or detect system 'events' by using the external clock input at the TCL0 pin (I/O port 4.0) as the counter source. The TC0 mode register selects rising or falling edge detection for incoming clock signals. The counter register TCNT0 is incremented each time the selected state transition of the external clock signal occurs.

With the exception of the different TMOD0.4-TMOD0.6 settings, the operation sequence for TC0's event counter function is identical to its programmable timer/counter function. To activate the TC0 event counter function,

— Set TMOD0.2 to "1" to enable TC0;

— Clear TMOD0.6 to "0" to select the external clock source at the TCL0 pin;

— Select TCL0 edge detection for rising or falling signal edges by loading the appropriate values to TMOD0.5 and TMOD0.4.

— P4.0 must be set to input mode.

Table 11-4. TMOD0 Settings for TCL0 Edge Detection

TMOD0.5	TMOD0.4	TCL0 Edge Detection
0	0	Rising edges
0	1	Falling edges

NOTE: If you set P4.0 to a open-drain, you can use P4.0 as TCLO pin for external TCO clock, even if P4.0 is set to output mode.

TCO CLOCK FREQUENCY OUTPUT

Using timer/counter 0, a modifiable clock frequency can be output to the TC0 clock output pin, TCLO0. To select the clock frequency, load the appropriate values to the TC0 mode register, TMOD0. The clock interval is selected by loading the desired reference value into the reference register TREF0. To enable the output to the TCLO0 pin at I/O port 4.1, the following conditions must be met:

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

— I/O mode flag for P4.1 (PM4) must be set to output mode ("1")

— Output latch value for P4.1 must be set to "0"

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

1. Load a reference value to TREF0.
2. Set the internal clock frequency in TMOD0.
3. Initiate TC0 clock output to TCLO0 (TMOD0.2 = "1").
4. Set port 4 mode flag (PM4) to "1".
5. Set P4.1 output latch to "0".
6. Set TOE0 flag to "1".

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

☞ PROGRAMMING TIP — TC0 Signal Output to the TCLO0 Pin

Output a 30 ms pulse width signal to the TCLO0 pin (at 4.19 MHz):

```
BITS      EMB
SMB       15
LD        EA,#79H
LD        TREF0,EA
LD        EA,#4CH
LD        TMOD0,EA
LD        A,#1H
LD        PMG2,A                  ;  P4.1← output mode
BITR      P4.1                    ;  P4.1clear
BITS      TOE0
```

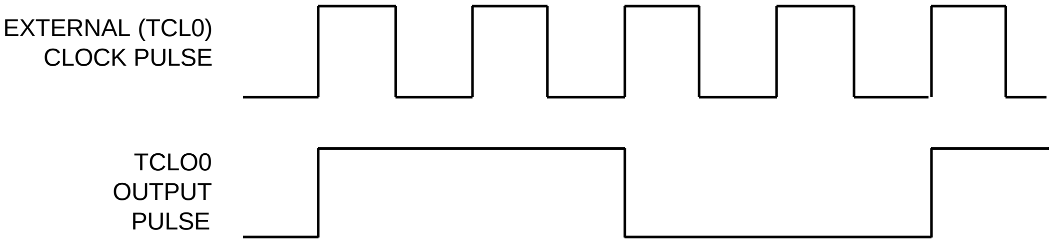
TC0 EXTERNAL INPUT SIGNAL DIVIDER

By selecting an external clock source and loading a reference value into the TC0 reference register, TREF0, you can divide the incoming clock signal by the TREF0 value and then output this modified clock frequency to the TCLO0 pin. The sequence of operations used to divide external clock input can be summarized as follows:

1. Load a signal divider value to the TREF0 register
2. Clear TMOD0.6 to "0" to enable external clock input at the TCL0 pin
3. Set TMOD0.5 and TMOD0.4 to desired TCL0 signal edge detection
4. Set port 4 mode flag (PM4) to output ("1")
5. Set P4.1 output latch to "0"
6. Set TOE0 flag to "1" to enable output of the divided frequency to the TCLO0 pin

PROGRAMMING TIP — External TCL0 Clock Output to the TCLO0 Pin

Output external TCL0 clock pulse to the TCLO0 pin (divided by four):



```
BITS    EMB
SMB     15
LD      EA,#01H
LD      TREF0,EA
LD      EA,#0CH
LD      TMOD0,EA
LD      A,#1H
LD      PMG2,A          ;  P4.1 ← output mode
BITR    P4.1            ;  P4.1 clear
BITS    TOE0
```

NOTE: The Port 4.0 must be a open-drain pin for external TC0 clock input to the TCL0 pin, when the Port 4 is set to output mode.

TC0 MODE REGISTER (TMOD0)

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

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

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

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

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

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

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

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

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

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

NOTE: 'fxx' = selected system clock of 4.19 MHz.

☞ PROGRAMMING TIP — Restarting TC0 Counting Operation

1. Set TC0 timer interval to 4.09 kHz:

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

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

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

TC0 COUNTER REGISTER (TCNT0)

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

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

Each time TCNT0 is incremented, the new value is compared to the reference value stored in the TC0 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/counter interval has elapsed.

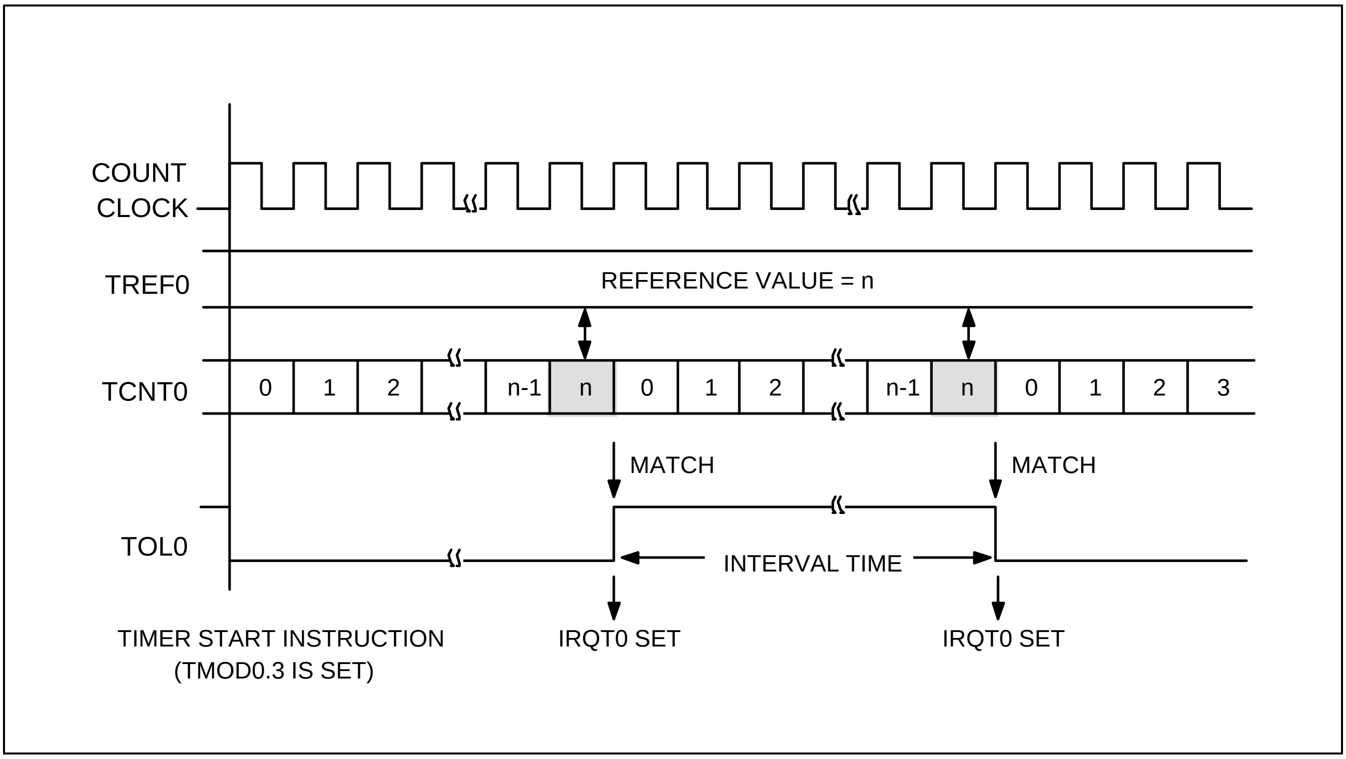


Figure 11-3. TC0 Timing Diagram

TC0 REFERENCE REGISTER (TREF0)

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

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

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

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

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

TC0 OUTPUT ENABLE FLAG (TOE0)

The 1-bit timer/counter 0 output enable flag TOE0 controls output from timer/counter 0 to the TCLO0 pin. TOE0 is addressable by 1-bit and 4-bit read/write instruction.

	(MSB)			(LSB)
F92H	"0"	**TOE0**	"U"	"0"

NOTE: The "U" means a undefined register bit.

When you set the TOE0 flag to "1", the contents of TOL0 can be output to the TCLO0 pin. Whenever a RESET occurs, TOE0 is automatically set to logic zero, disabling all TC0 output. Even when the TOE0 flag is disabled, timer/counter 0 can continue to output an internally generated clock frequency, via TOL0.

TC0 OUTPUT LATCH (TOL0)

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

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

☞ PROGRAMMING TIP — Setting a TC0 Timer Interval

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

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

2. Calculate the TREF0 value:

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

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

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

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

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

WATCH TIMER

OVERVIEW

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

- 8-bit watch timer mode register (WMOD)
- Clock selector
- Frequency divider circuit

Watch timer functions include real-time and watch-time measurement and interval timing for the main and 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 Main System or Subsystem Clock Source

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

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

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.

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 at selected clock for watch timer. To select the desired BUZ frequency , load the appropriate value to the WMOD register. This output can then be used to actuate an external buzzer sound. To generate a BUZ signal, three conditions must be met:

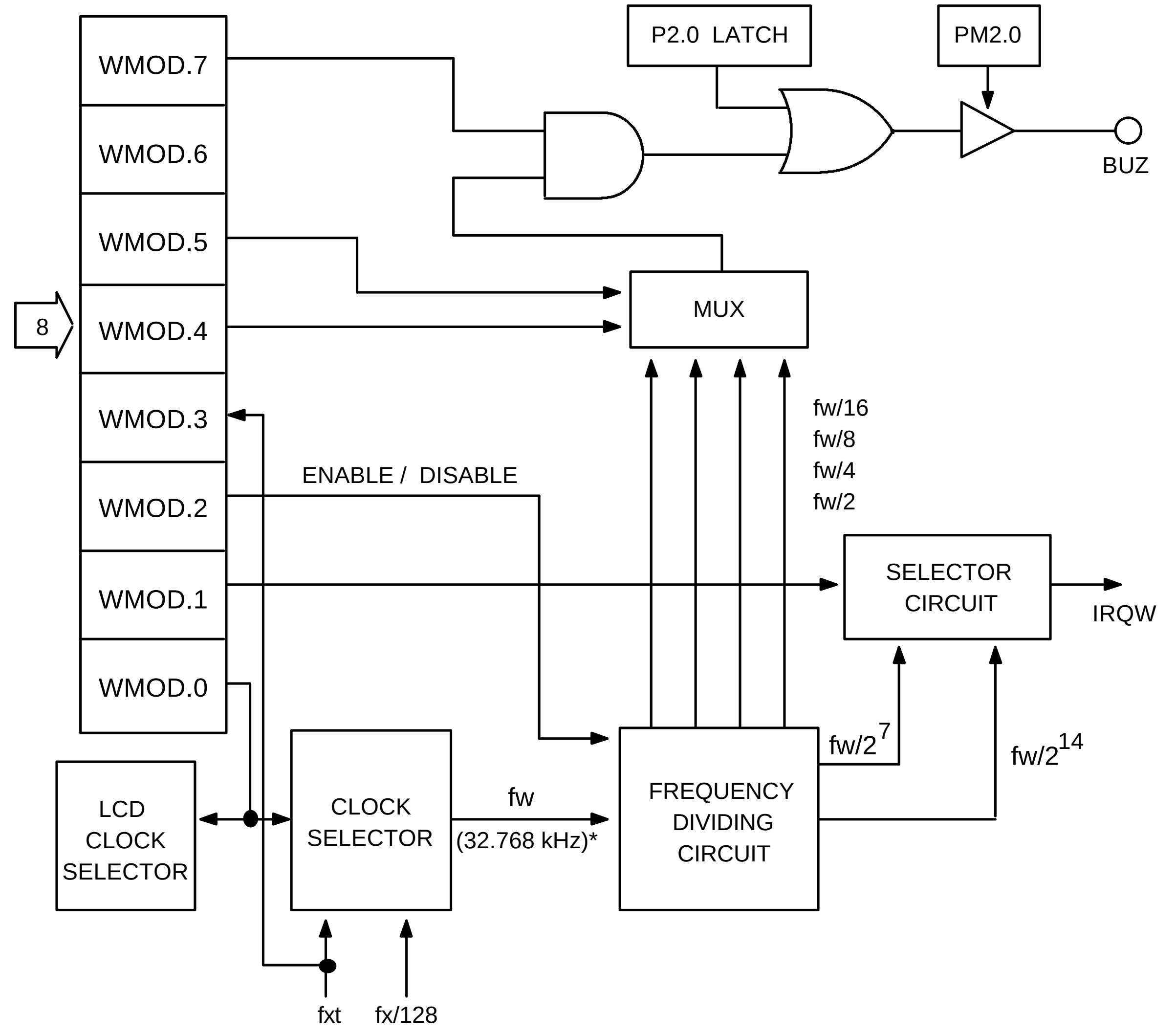
- The WMOD.7 register bit is set to "1"
- The output latch for I/O port 2.0 is cleared to "0"
- The port 2.0 output mode flag (PM2.0) set to 'output' mode

Timing Tests in High-Speed Mode

By setting WMOD.1 to "1", the watch timer will operate in high-speed mode, generating an interrupt every 3.91 ms at oscillation clock of 4.19 MHz. 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.



fx = MAN SYSTEM CLOCK
fxt = SUB SYSTEM CLOCK
fw = WATCH TIMER FREQUENCY
* fw = 32.768 kHz, when fx is 4.19 MHz.

Figure 11-4. Watch Timer Circuit Diagram

WATCH TIMER MODE REGISTER (WMOD)

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

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

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

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

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

Bit Name	Values		Function	Address
WMOD.7	0		Disable buzzer (BUZ) signal output at the BUZ pin	F89H
	1		Enable buzzer (BUZ) signal output at the BUZ pin	
WMOD.6	0		Always logic zero	
WMOD.5 – .4	0	0	2 kHz buzzer (BUZ) 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 second	
	1		High-speed mode; sets IRQW to 3.91 ms	
WMOD.0	0		Select (fx/128) as the watch timer clock (fw) Select a LCD clock source as main system clock	
	1		Select subsystem clock as watch timer clock (fw) Select a LCD clock source as sub system clock	

NOTE: Main system clock frequency (fx) is assumed to be 4.19 MHz; subsystem clock (fxx) 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,#04H
            LD        PMG1,EA                 ;  P2.0 ←  output mode
            BITR      P2.0
            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
```

12 LCD CONTROLLER/DRIVER

OVERVIEW

The KS57C21408/C21418/P21408 microcontroller can directly drive an up-to-12-characters (5x7 dots) LCD panel. Its LCD block has the following components:

- LCD controller/driver
- Display RAM (100H–1B8H) for storing display data (13H page)
- 60 segment output pins (SEG0–SEG59)
- 9 common output pins (COM0–COM8)

The frame frequency, LCD divide resistors, key strobe signal output key, check signal output, and normal LCD display are determined by bit settings in the LCD mode register, LMOD0 and LMOD1.

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

LCD CIRCUIT DIAGRAM

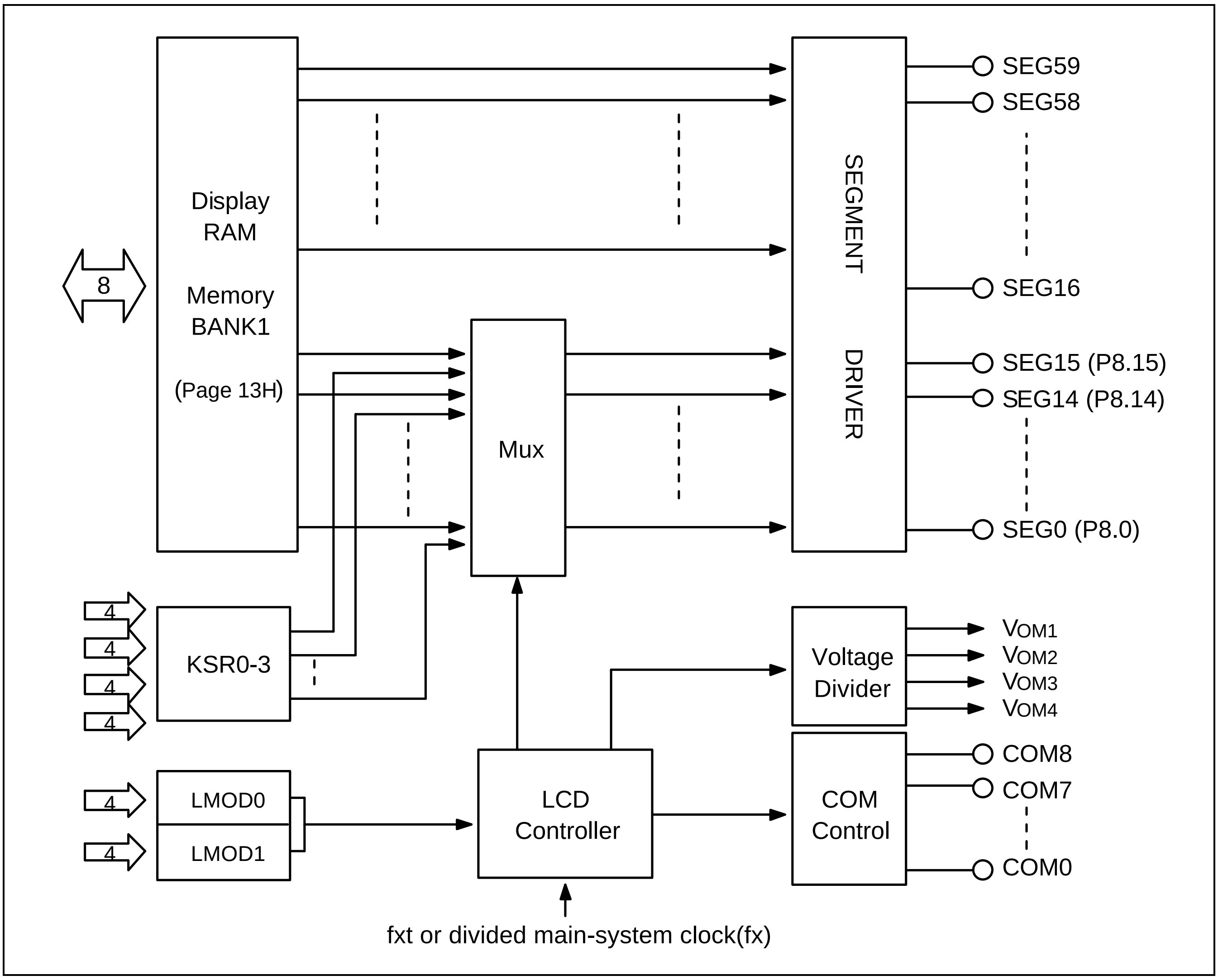


Figure 12-1. LCD Circuit Diagram

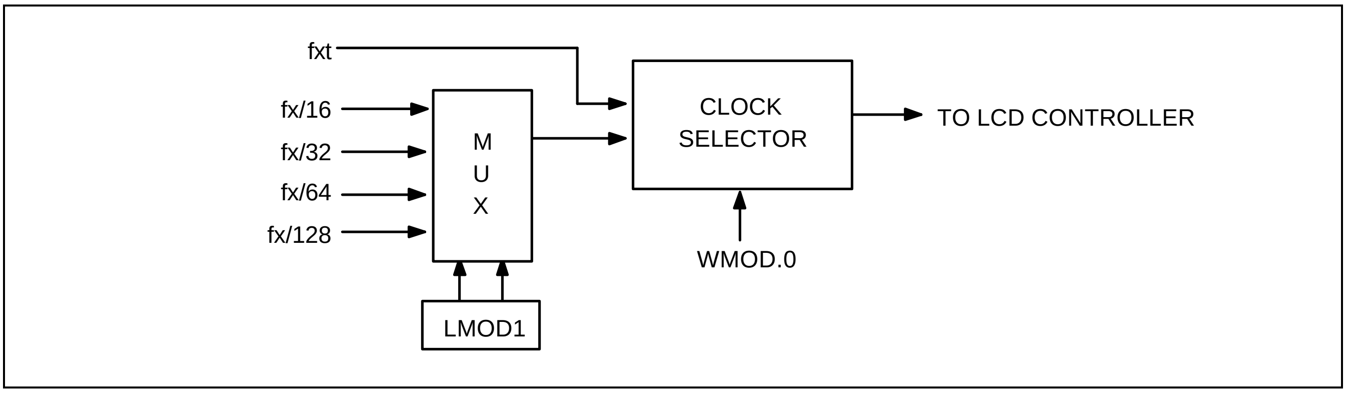


Figure 12-2. LCD Clock Circuit Diagram

LCD RAM ADDRESS AREA

RAM addresses 100H–1B8H of bank1 page 13H are used as LCD data memory. These locations can be addressed by 8-bit instructions only. However, the upper 3 bits of each address must be written to zero.
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-SEG59 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.

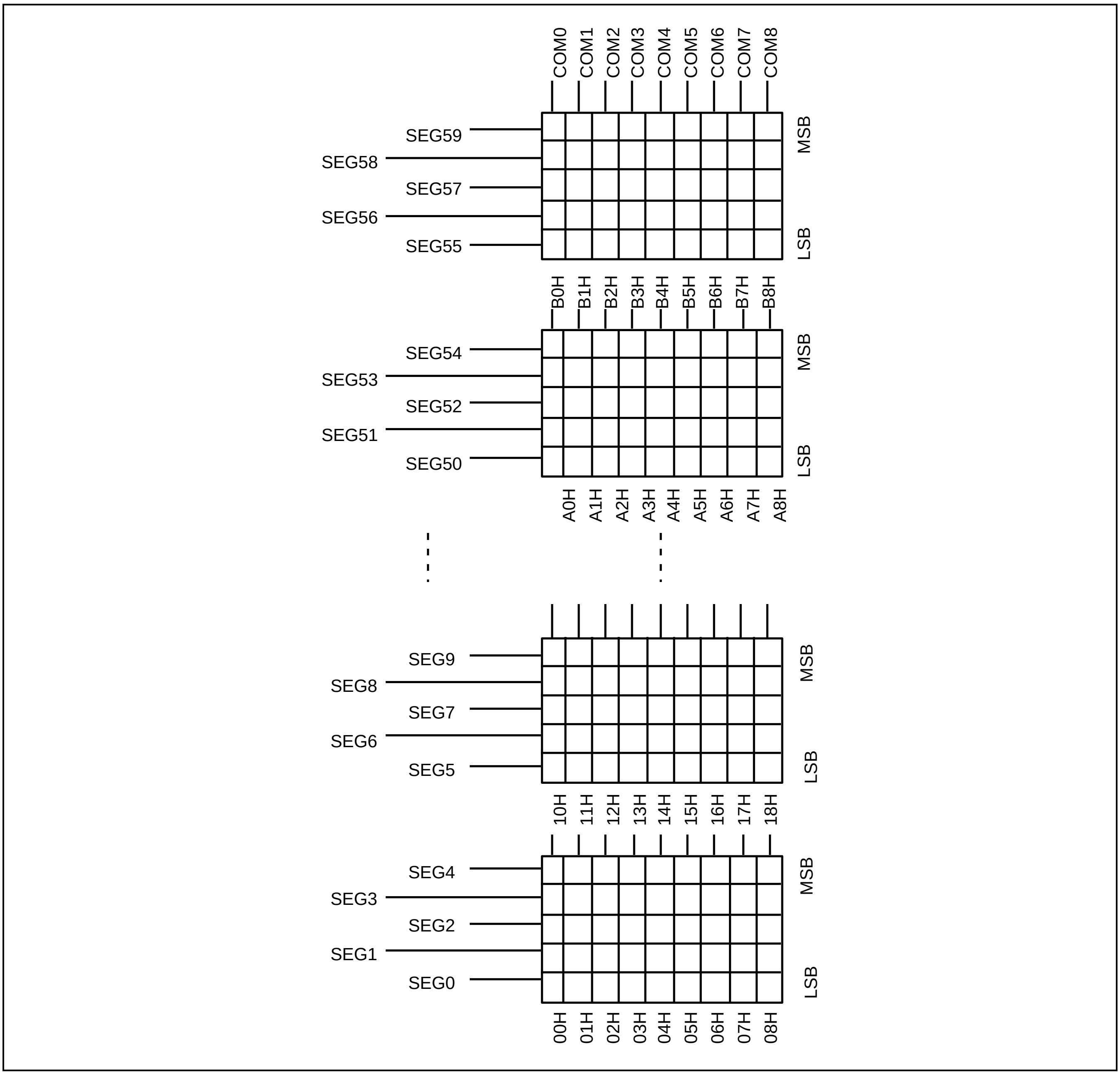


Figure 12-3. Display RAM Organization

LCD MODE REGISTER (LMOD)

The LCD mode register LMOD can be manipulated using 4-bit write instructions.

F8CH	LMOD0.3	LMOD0.2	LMOD0.1	LMOD0.0
F8DH	LMOD1.3	LMOD1.2	LMOD1.1	LMOD1.0

LMOD controls the following LCD functions:

— LCD display on/off LMOD0.2 and LMOD0.1

— Key check signal output with LCD display off (LMOD0.3)

— Dimming mode (LMOD0.0)

— LCD dividing resistors selection (LMOD1.1)

— Key strobe signal disable (LMOD1.0)

— LCD frame frequency selection for main system clock (LMOD1.2/3)

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

Table 12-1. LCD Mode Control Register (LMOD) Organization

LMOD0.3	LMOD0.2	LMOD0.1	LMOD0.0	Duty and Bias Selection for LCD Display
0	1	0	0	LCD display on
0	1	0	1	Dimming mode (key strobe state)
1	0	0	1	Key check signal output with LCD display off

NOTE: If one of 16 bits (KSR0.0 - KSR3.3) is set to logic "1", the corresponding segment pin becomes the low level, when the LCD Display is off by LMOD0 ← 9H

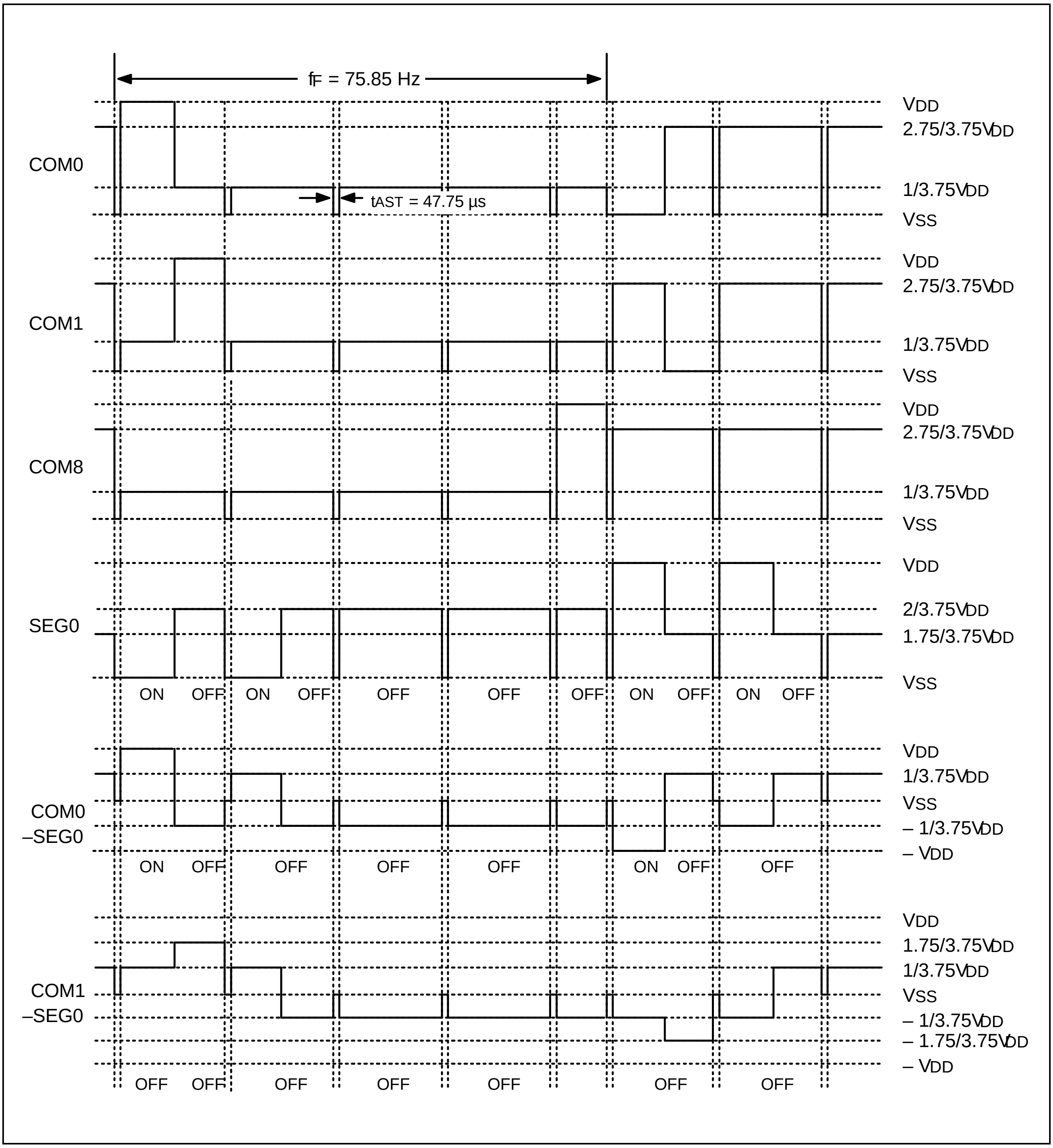
LMOD1.0	0	Key strobe signal output
	1	Disable key strobe signal output, port0 pull-up registers enable

LMOD1.1	0	Normal LCD dividing resistor
	1	Diminish a LCD dividing resistor to strengthen LCD drive

LMOD1.3	LMOD1.2	LCD clock frequency at fx = 4.19MHz
0	0	fT = 75.85 Hz at 4.19 MHz Main oscillator
0	1	fT = 151.7 Hz at 4.19 MHz Main oscillator
1	0	fT = 303.4 Hz at 4.19 MHz Main oscillator
1	1	fT = 606.8 Hz at 4.19 MHz Main oscillator

NOTES:

1. If sub-clock is selected for watch timer source , LCD frame frequency is always 75.85 Hz. Only when main clock is selected for LCD clock source, LMOD1.2 and LMOD1.3 are applied. For more information, refer to "LMOD1" in chapter4 MEMORY MAP.
2. 1/3.75 Bias, 1/9 duty



The upper figure is COM and SEG signal waveform when LMOD1.0-LMOD1.3 is "0".

Figure 12-4. COM and SEG Signal Waveform

LMOD0	LMOD0.3	LMOD0.2	LMOD0.1	LMOD0.0	F8CH

LMOD0.0	0	Normal display
	1	Key strobe state

LMOD0.2	LMOD0.1	Function
0	0	Display OFF
1	0	Display ON

LMOD0.3	SEG0-SEG15 pin
0	Normal display signal output
1	Key check signal output

When the LCD display is turned on, and if two keys are pressed simultaneously, the display is dimmed by a segment pin short. Therefore, use LMOD0.0 to protect against display dimming.
When the LCD is turned off in STOP mode, LMOD0 should be set to 1001B to minimize current.

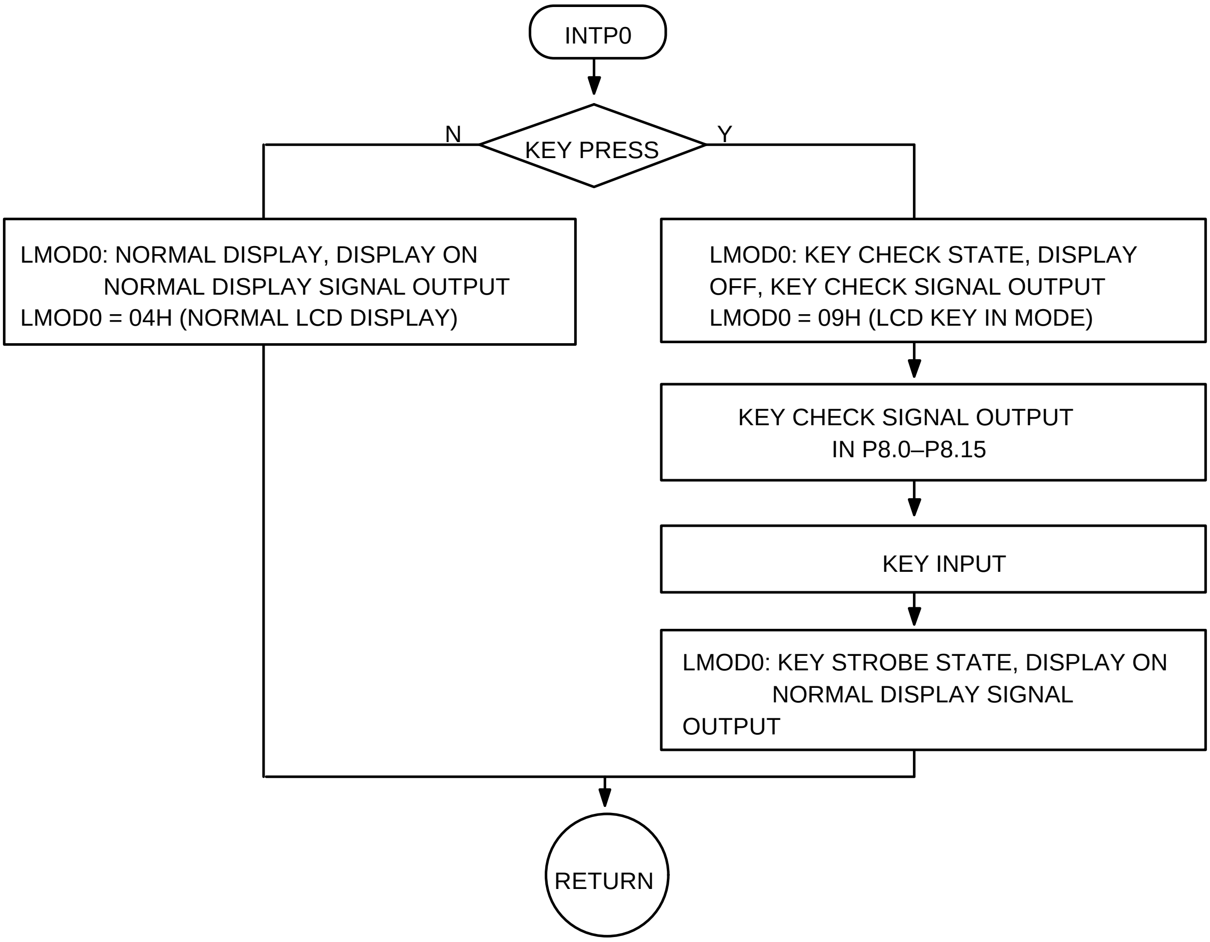

INTP0
KEY PRESS
N
Y
LMOD0: NORMAL DISPLAY, DISPLAY ON
NORMAL DISPLAY SIGNAL OUTPUT
LMOD0 = 04H (NORMAL LCD DISPLAY)
LMOD0: KEY CHECK STATE, DISPLAY
OFF, KEY CHECK SIGNAL OUTPUT
LMOD0 = 09H (LCD KEY IN MODE)
KEY CHECK SIGNAL OUTPUT
IN P8.0–P8.15
KEY INPUT
LMOD0: KEY STROBE STATE, DISPLAY ON
NORMAL DISPLAY SIGNAL
OUTPUT
RETURN

Figure 12-5. LMOD0 Organization

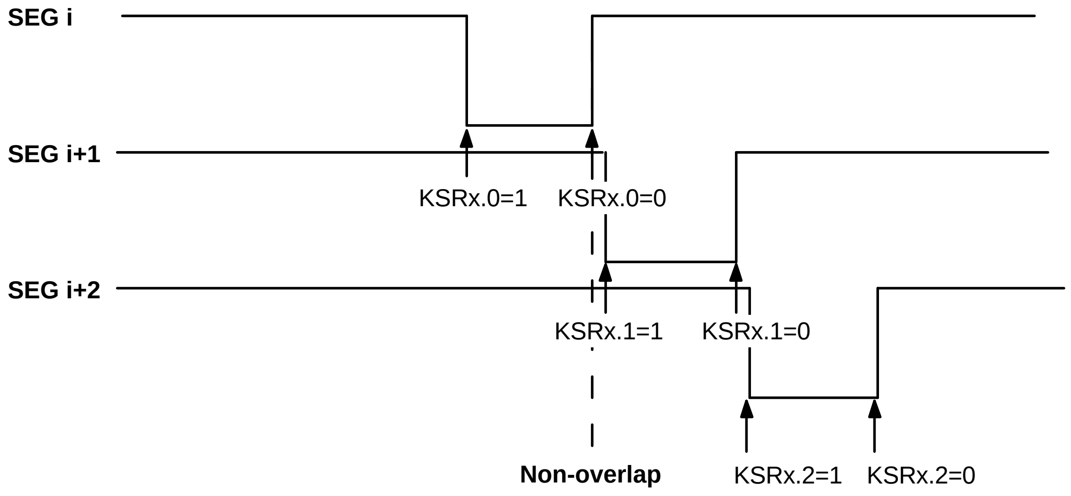
KEY SCAN REGISTER (KSR)

The 16 output pins (P8.0–P8.15) of 60 segments can be used for key check signal output. KSR0–KSR3 are mapped to the RAM address FA2H–FA5H, and the reset value is "0". KSR is the write-only register that can be manipulated by 4-bits RAM write instruction only.

Table 12-2. KSR Organization

KSR0	KSR0.3	KSR0.2	KSR0.1	KSR0.0	FA2H
KSR1	KSR1.3	KSR1.2	KSR1.1	KSR1.0	FA3H
KSR2	KSR2.3	KSR2.2	KSR2.1	KSR2.0	FA4H
KSR3	KSR3.3	KSR3.2	KSR3.1	KSR3.0	FA5H

When LMOD0.3 = 1, the values of KSR0–KSR3 are output to segment pins for key check. At this time, only one of 16 bits (KSR0.0-KSR3.3) must be set to logic "1", and the contents of KSR must be changed 16 times one by one for 16 key check by software. When a bit value of KSR is "1", the corresponding segment pin becomes the low level. Figure 12-6 shows its segment pin output.



NOTE: x means 0, 1, 2 and 3.

Figure 12-6. Segment Pin Output Signal When LMOD0.3 = 1

NOTES

13 ELECTRICAL DATA

OVERVIEW

In this section, information on KS57C21408/C21418/P21408 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
- Sub-system 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}
- TCL0 timing
- 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 13-1. 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_I	Ports 0, 1, 2, 4, 5, 6, 7		– 0.3 to V_{DD} + 0.3	V
Output Voltage	V_O	–		– 0.3 to V_{DD} + 0.3	V
High Level Output current	I_{OH}	One pin		– 15	mA
		All output pins		– 30	mA
Low Level Output Current	I_{OL}	One pin	Peak value	30	mA
			RMS value (note)	15	mA
		All pins	Peak value	100	mA
			RMS value (note)	60	mA
Operating Temperature	T_A	–		– 40 to + 85	°C
Storage Temperature	T_{STG}	–		– 65 to + 150	°C

NOTE : RMS value = Peak Value × $\sqrt{\text{Duty}}$.

Table 13-2. D.C 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	VIH1	Pins except below	0.7 V_{DD}	–	V_{DD}	V
	VIH2	Port0, 1, 6, 7, P4.0, RESET	0.8 V_{DD}	–	V_{DD}	
	VIH3	X_{IN}, X_{OUT} and XT_{IN}	V_{DD} – 0.1	–	V_{DD}	
Input Low Voltage	VIL1	All input pins except below	–	–	0.3 V_{DD}	
	VIL2	Port0, 1, 6, 7, P4.0, RESET		–	0.2 V_{DD}	
	VIL3	X_{IN},X_{OUT} and XT_{IN}		–	0.1	
Output High Voltage	VOH1	V_{DD} = 4.5 V to 5.5 V Port2, 4, 5, 6, 7 I_{OH} = – 1mA	V_{DD} – 1.0	–	–	

Table 13-2. D.C Characteristics(continued)

(T_A = – 40 °C to + 85C, 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 Port2, 4, 5, 6, 7 I_{OL}= 15mA	–	–	2	
		V_{DD} = 1.8 V to 5.5 V I_{OL}=1.6mA	–	–	0.4	
Input High Leakage Current	I_{LIH1}	Vin = V_{DD} All input pins except below	–	–	3	µA
	I_{LIH2}	Vin = V_{DD} X_{IN}, X_{OUT}, XT_{IN}			20	
Input Low Leakage Current	I_{LIL1}	V_{IN} = 0 V All input pins except X_{IN}, X_{OUT}, XT_{IN} and RESET	–	–	– 3	
	I_{LIL2}	V_{IN} = 0 V X_{IN}, X_{OUT}, XT_{IN}	–	–	– 20	
Output High Leakage Current	I_{LOH1}	V_O = V_{DD} Port2, 4, 5, 6, 7	–	–	3	
Output Low Leakage Current	I_{LOL1}	V_O = 0 V Port2, 4, 5, 6, 7	–	–	– 3	
Pull-up Resistor	RL1	V_{DD} = 5 V, V_{IN} = 0 V All pins except RESET	25	50	100	KΩ
		V_{DD} = 3 V	50	100	200	
	RL2	V_{DD} = 5 V, V_{IN} = 0 V RESET	100	250	400	
		V_{DD} = 3 V	200	500	800	
Medium Output Voltage(1)	V_{OM1}	COM0-COM8	VM1 – 0.2	VM1	VM1 + 0.2	V
	V_{OM2}	COM0-COM8	VM2 – 0.2	VM2	VM2 + 0.2	
	V_{OM3}	SEG0-CSEG59	VM3 – 0.2	VM3	VM3 + 0.2	
	V_{OM4}	SEG0-CSEG59	VM4 – 0.2	VM4	VM4 + 0.2	
High Output Impedance	ROH1	V_O = V_{DD}–0.5V SEG0-SEG59	–	–	90	KΩ
	ROH2	COM0-COM8	–	–	25	
Low Output Resistor	ROL1	VO = 0.5V SEG0-SEG59	–	–	90	kΩ
	ROL2	SEG0-SEG15 (key strobe)	–	–	2	
	ROL3	COM0-COM8	–	–	25	

Table 13-2. D.C Characteristics (continued)

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

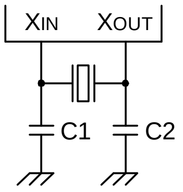
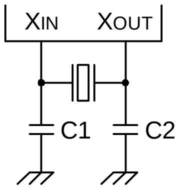
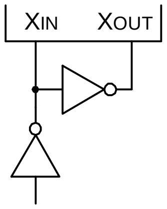
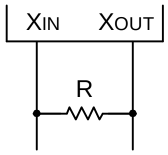
Parameter	Symbol	Conditions		Min.	Typ.	Max.	Units
Supply Current [2] [3]	IDD1	Run mode : V_{DD} = 5 V ± 10%	6MHz	–	5.1	8	mA
		Crystal oscillator C1 = C2 = 22pF	4.19MHz		3.8	6	
		V_{DD} = 3 V ± 10%	6MHz		2.5	4	
			4.19MHz		1.8	3	
	IDD2	Idle mode : V_{DD} = 5 V ± 10%	6MHz		1.3	2.5	
		Crystal oscillator C1 = C2 = 22pF	4.19MHz		1.1	1.8	
		V_{DD} = 3 V ± 10%	6MHz		0.5	1.5	
			4.19MHz		0.4	1.0	
	IDD3	Run mode: V_{DD} = 3 V ± 10% 32kHz crystal oscillator		–	30	45	µA
	IDD4	Idle mode: V_{DD} = 3 V ± 10% 32kHz crystal oscillator	LCD ON [4]	–	17	30	
		V_{DD} = 3 V ± 10% 32kHz crystal oscillator	LCD OFF		6	15	
	IDD5	Stop mode; V_{DD} = 5 V ± 10%, XT_{IN} = 0V		–	2.4	5	
		Stop mode; V_{DD} = 3 V ± 10%, XT_{IN} = 0V			0.6	3	

NOTES:

1. VM1=2.75/3.75 V_{DD}, VM2=1/3.75 V_{DD}, VM3=2/3.75 V_{DD}, VM4=1.75/3.75 V_{DD}
2. Supply current does not include current drawn through internal pull-down resistor and LCD driving resistors.
3. For D.C. electrical voltages, PCON register must be set to 0011B.
4. The mode of I_{DD4} (LCD ON) is normal.

Table 13-3. 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(fx) (1)	–	0.4	–	6.0	MHz
		Stabilization time (2)	After V_{DD} reaches the minimum level of its variable range	–	–	4	ms
Crystal Oscillator		Oscillation frequency(fx) (1)	–	0.4	–	6	MHz
		Stabilization time (2)	V_{DD} = 4.5 V to 5.5 V	–	–	10	ms
			V_{DD} = 1.8 V to 5.5 V	–	–	60	
External Clock		X_{in} input frequency(fx) (1)	–	0.4	–	6	MHz
		X_{in} input high and low level width (t_{XH}, t_{XL})	–	83.3	–	1250	ns
RC Oscillator		Frequency	V_{DD} = 5 V	–	2	–	MHz
			V_{DD} = 3 V	–	1	–	

NOTES:

1. Oscillation frequency and input frequency data are for oscillator characteristics only.
2. Stabilization time is the interval required for oscillator stabilization after a power-on or release of STOP mode.

Table 13-4. Recommended Oscillator Constants

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

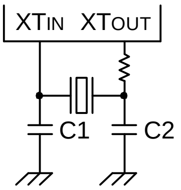
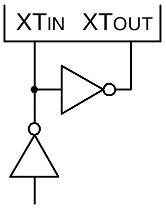
Manufacturer	Series Number (1)	Frequency Range	Load Cap (pF)		Oscillator Voltage Range (V)		Remarks
			C1	C2	MIN	MAX	
TDK	FCR ðÿM5	3.58 MHz–6.0 MHz	33	33	2.0	5.5	Leaded Type
	FCR ðÿMC5	3.58 MHz–6.0 MHz	(2)	(2)	2.0	5.5	On-chip C Leaded Type
	CCR ðÿMC3	3.58 MHz–6.0 MHz	(3)	(3)	2.0	5.5	On-chip C SMD Type

NOTES:

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

Table 13-5. Subsystem Clock Oscillator Characteristics

(T_A = – 40 °C + 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} = 4.5 V to 5.5 V	–	1.0	2	ms
			V_{DD} = 1.8 V to 5.5 V	–	–	10	
External Clock		XT_{in} input frequency [1]	–	32	–	100	kHz
		XT_{in} input high and low level width (t_{XTH}, t_{XTL})	–	5	–	15	µs

NOTES:

1. Oscillation frequency and input frequency data are for oscillator characteristics only.
2. Stabilization time is the interval required for oscillating stabilization after a power-on or release of STOP mode.

Table 13-6. Input/Output Capacitance

(T_A = 25 °C, V_{DD} = 0 V)

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

Table 13-7. 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 (NOTE)	t_{CY}	V_{DD} = 2.7 V to 5.5 V	0.67	–	64	µs
		V_{DD} = 1.8 V to 5.5 V	1.33			
		With sub-system clock (fxt)	114	122	1952	
TCL0 Input Frequency	f_{TI}	V_{DD} = 2.7 V to 5.5 V	0	–	1.5	MHz
		V_{DD} = 1.8 V to 5.5 V			1	kHz
TCL0 Input High, Low Width	t_{TIH} t_{TIL}	V_{DD} = 2.7 V to 5.5 V	0.48	–	–	µs
		V_{DD} = 1.8 V to 5.5 V	1.8			
External Interrupt Input High, Low Width	t_{INTH}, t_{INTL}	INT0, INT1, KS0 - KS7	10	–	–	µs
		KS0 - KS3	10			
RESET Low Level Width	t_{RSL}	–	10	–	–	µs

NOTE: Unless otherwise specified, the values of instruction cycle time condition assume a main-system clock (fx) source.

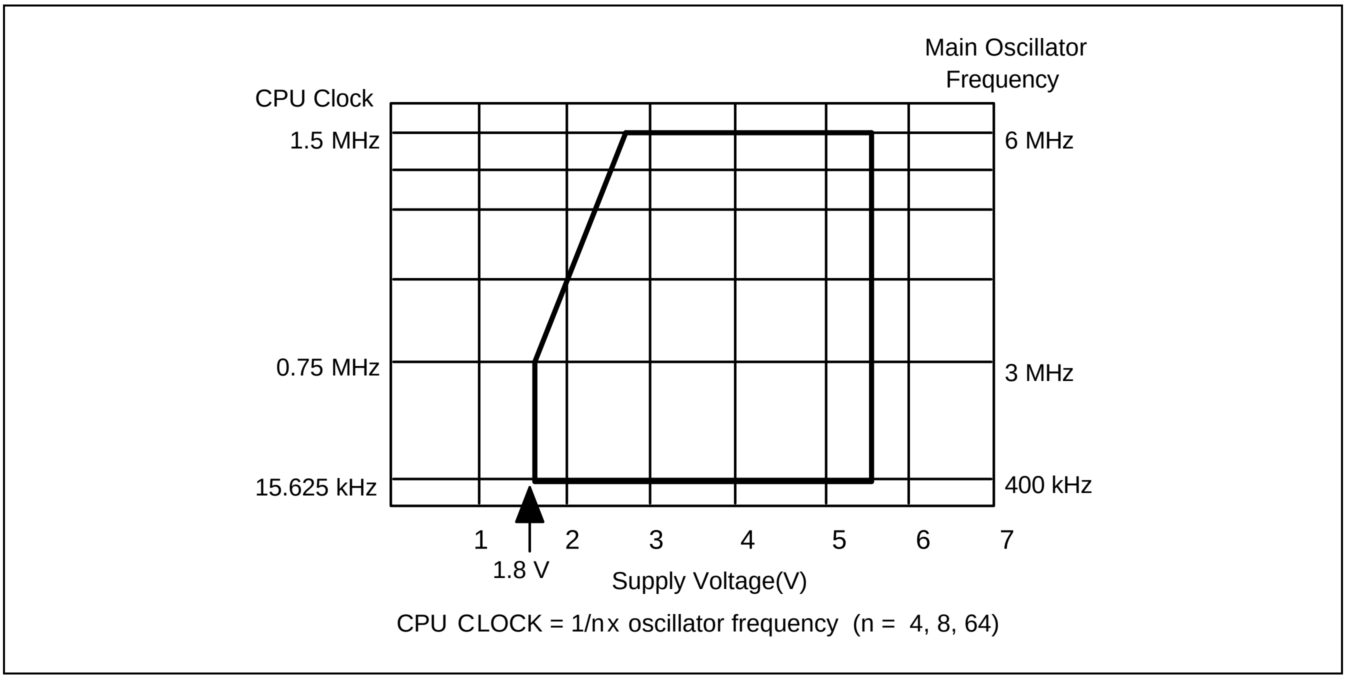


Figure 13-1. Standard Operating Voltage Range

Table 13-8. 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}	–	1.8	–	5.5	V
Data retention supply current	I_{DDDR}	V_{DDDR} = 1.8 V	–	0.1	10	μA
Release signal set time	t_{SREL}	–	0	–	–	μs
Oscillator stabilization wait time (1)	t_{WAIT}	Released by RESET	–	2^{17} / fx	–	ms
		Released by interrupt	–	(2)	–	

NOTES:
1. During oscillator stabilization time, all CPU operations are stopped to avoid unstable operation upon oscillation start.
2. The basic timer mode register (BMOD) interval timer delays execution of CPU instructions during the wait time.

TIMING WAVEFORMS

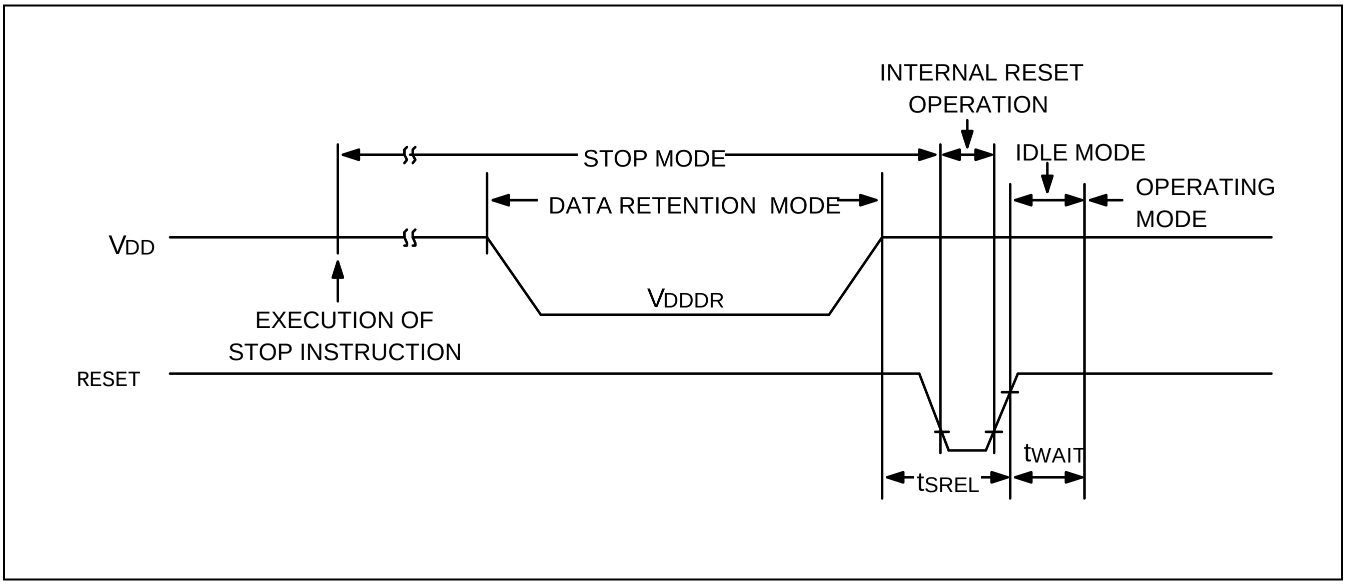


Figure 13-2. Stop Mode Release Timing When Initiated By RESET

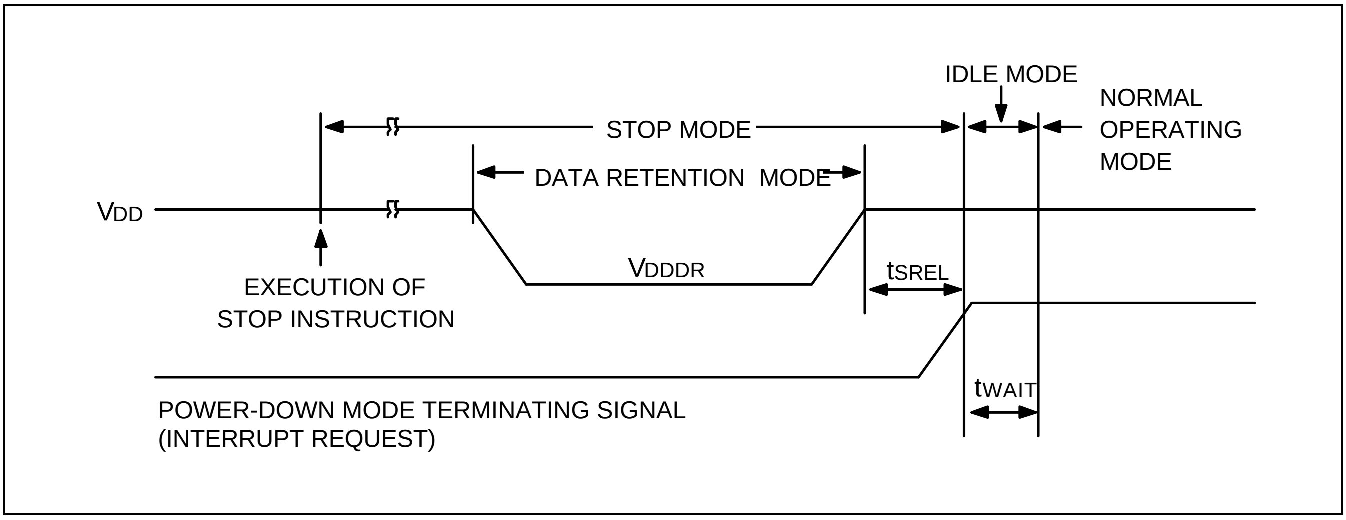


Figure 13-3. Stop Mode Release Timing When Initiated By Interrupt Request

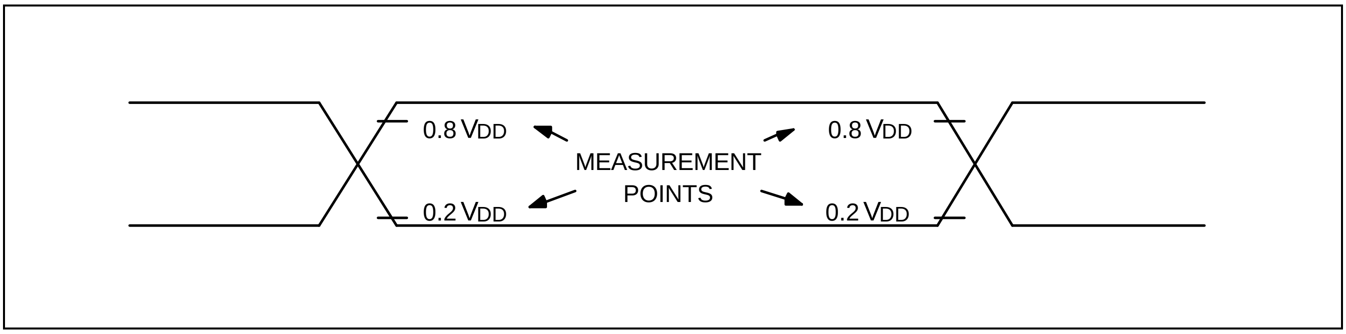


Figure 13-4. A.C. Timing Measurement Points (Except for X_{in} and XT_{in})

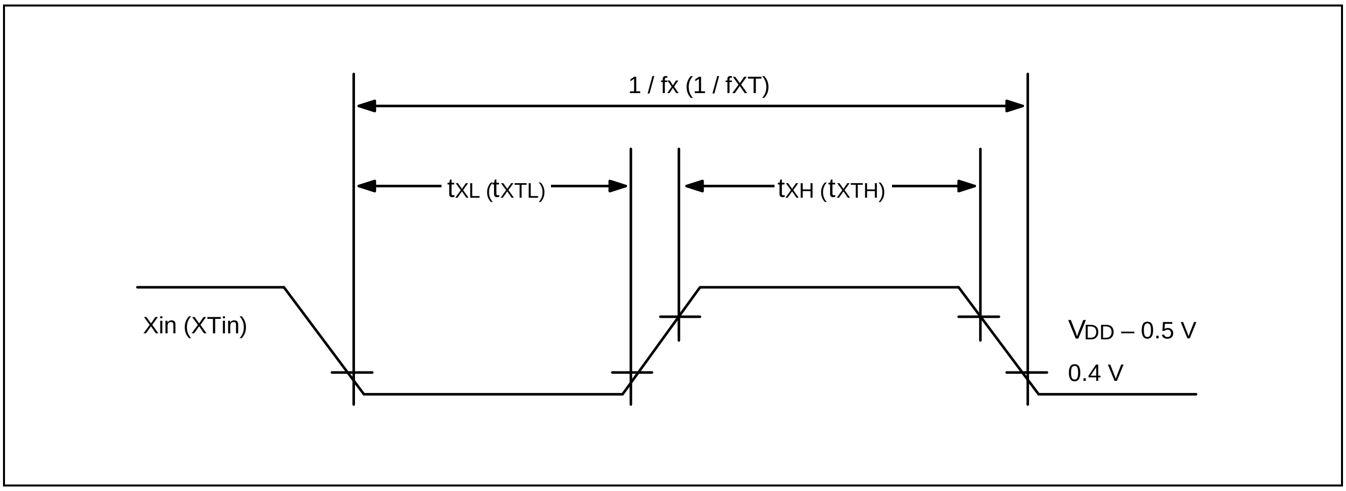


Figure 13-5. Clock Timing Measurement at X_{in} and XT_{in}

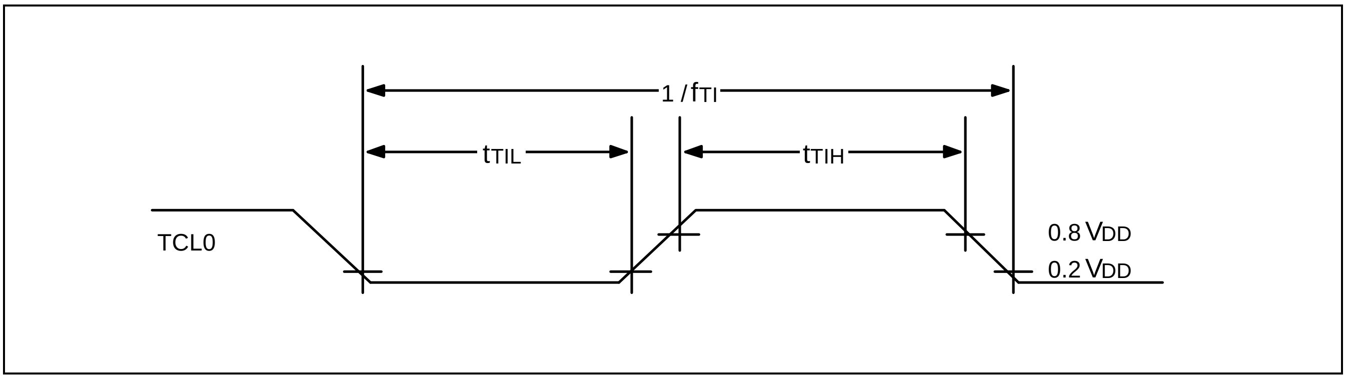


Figure 13-6. TCL0 Timing

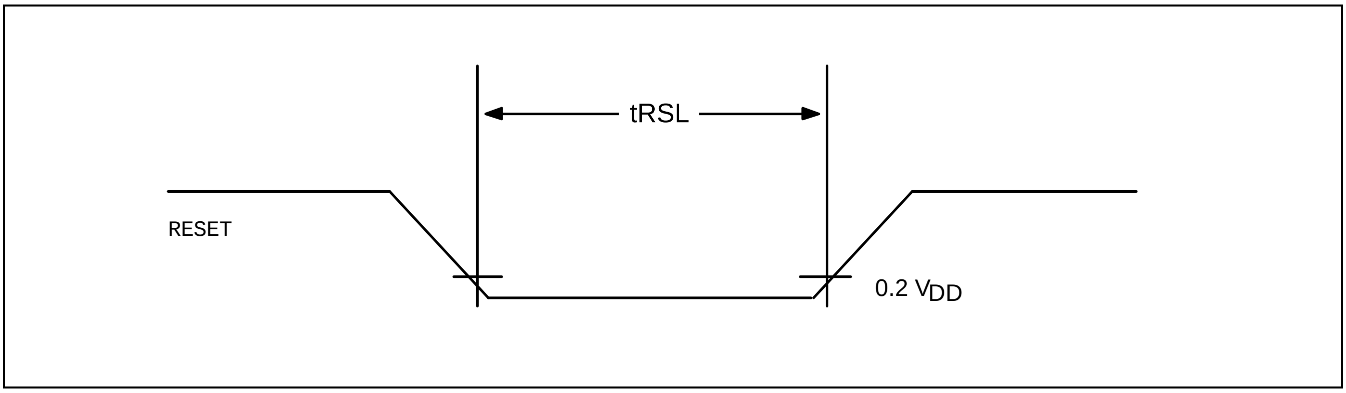


Figure 13-7. Input Timing for RESET Signal

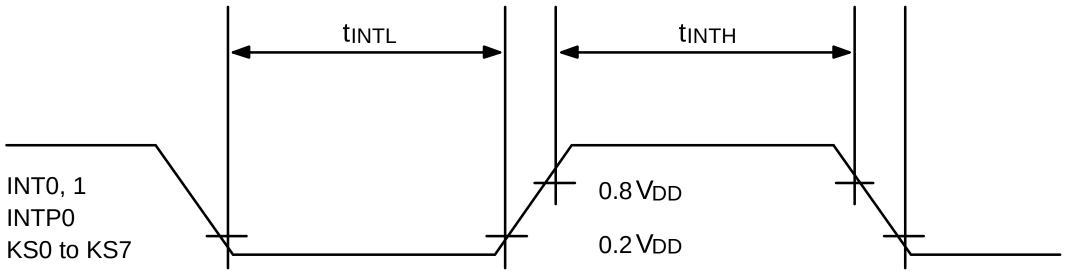


Figure 13-8. Input Timing for External Interrupts and Quasi-Interrupts

14 MECHANICAL DATA

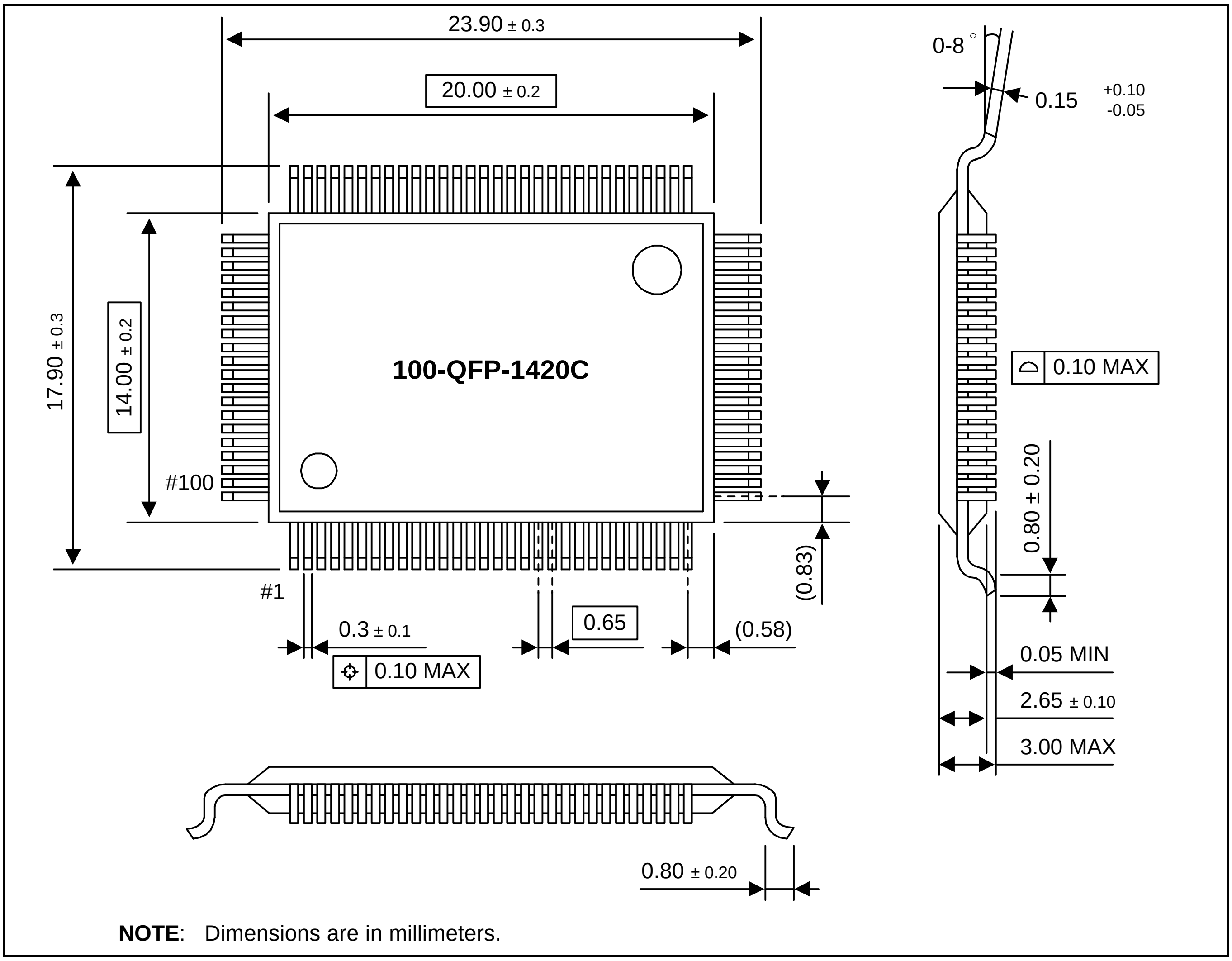
OVERVIEW

This section contains the following information about the device package:

— Package dimensions in millimeters

— Pad diagram

— Pad/pin coordinate data table



NOTE: Dimensions are in millimeters.

Figure 14-1. 100-QFP-1420 Package Dimensions

NOTES

15 KS57P21408 OTP

OVERVIEW

The KS57P21408 single-chip CMOS microcontroller is the OTP (One Time Programmable) version of the KS57C21408/C21418 microcontroller. It has an on-chip OTP ROM instead of masked ROM. The EPROM is accessed by serial data format.

The KS57P21408 is fully compatible with the KS57C21408/C21418, both in function and in pin configuration. Because of its simple programming requirements, the KS57P21408 is ideal for use as an evaluation chip for the KS57C21408/C21418.

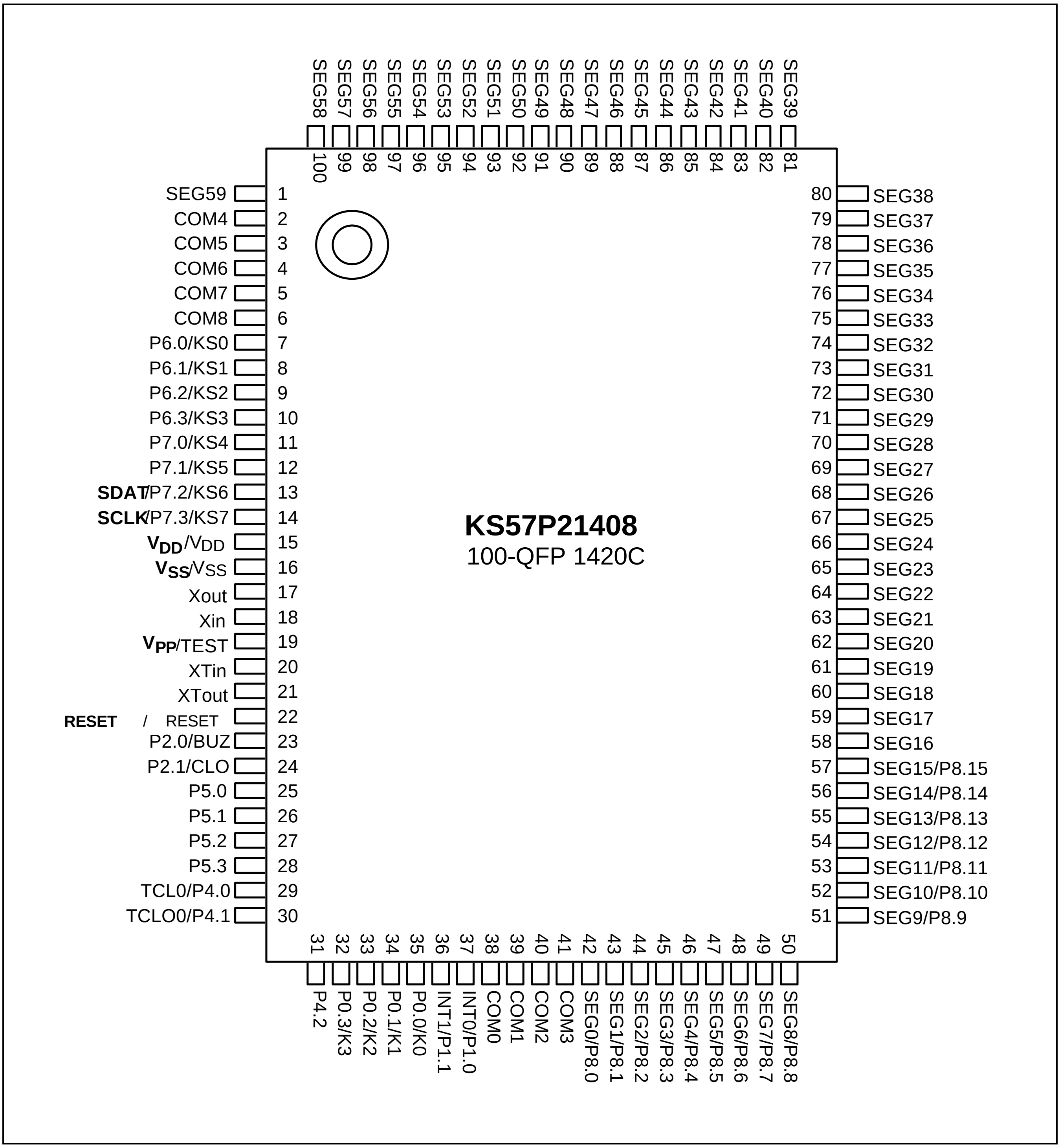


Figure 15-1. KS57P21408 Pin Assignments (100-QFP Package)

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

Main Chip	During Programming			
Pin Name	Pin Name	Pin No.	I/O	Function
P3.1	SDAT	13	I/O	Serial data pin. Output port when reading and input port when writing. Can be assigned as a Input / push-pull output port.
P3.0	SCLK	14	I/O	Serial clock pin. Input only pin.
TEST	V_{PP}(TEST)	19	I	Power supply pin for EPROM cell writing (indicates that OTP enters into the writing mode). When 12.5 V is applied, OTP is in writing mode and when 5 V is applied, OTP is in reading mode. (Option)
RESET	RESET	22	I	Chip initialization
V_{DD} / V_{SS}	V_{DD} / V_{SS}	15/16	I	Logic power supply pin. VDD should be tied to +5 V during programming.

Table 15-2. Comparison of KS57P21408 and KS57C21408/C21418 Features

Characteristic	KS57P21408	KS57C21408
Program Memory	8 Kbyte EPROM	8 Kbyte mask ROM
Operating Voltage (V_{DD})	1.8 V to 5.5 V	1.8 V to 5.5V
OTP Programming Mode	V_{DD} = 5 V, V_{PP}(TEST)=12.5V	
Pin Configuration	100 QFP	100 QFP
EPROM Programmability	User Program 1 time	Programmed at the factory

OPERATING MODE CHARACTERISTICS

When 12.5 V is supplied to the V_{PP}(TEST) pin of the KS57P21408, the EPROM programming mode is entered. The operating mode (read, write, or read protection) is selected according to the input signals to the pins listed in Table 15-3 below.

Table 15-3. Operating Mode Selection Criteria

V_{DD}	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 15-4. D.C Characteristics

(T_A = –40 °C to +85C, VDD = 1.8 V to 5.5V)

Parameter	Symbol	Conditions		Min.	Typ.	Max.	Units
Supply Current [2][3]	I_{DD1}	Run mode : V_{DD}=5V±10%	6MHz	–	5.1	8	mA
		Crystal oscillator C1=C2=22pF	4.19MHz		3.8	6	
		V_{DD}=3V±10%	6MHz		2.5	4	
			4.19MHz		1.8	3	
	I_{DD2}	Idle mode : V_{DD}=5V±10%	6MHz		1.3	2.5	
		Crystal oscillator C1=C2=22pF	4.19MHz		1.1	1.8	
		V_{DD}=3V±10%	6MHz		0.5	1.5	
			4.19MHz		0.4	1.0	
	I_{DD3}	Run mode : V_{DD}=3V±10% 32kHz crystal oscillator		–	30	45	µA
	I_{DD4}	Idle mode : V_{DD}=3V±10% 32kHz crystal oscillator	LCD ON[4]	–	17	30	
		V_{DD}=3V±10% 32kHz crystal oscillator	LCD OFF		6	15	
	I_{DD5}	Stop mode; V_{DD}=5V±10%		–	2.4	5	
		Stop mode; V_{DD}=3V±10%			0.6	3	

NOTES:

1. VM1=2.75/3.75 VDD, VM2=1/3.75 VDD, VM3=2/3.75 VDD, VM4=1.75/3.75 VDD

2. Supply current does not include current drawn through internal pull-down resistor and LCD driving resistors.

3. For D.C. electrical voltages, PCON register must be set to 0011B.

5. The mode of I_{DD4} (LCD ON) is normal.

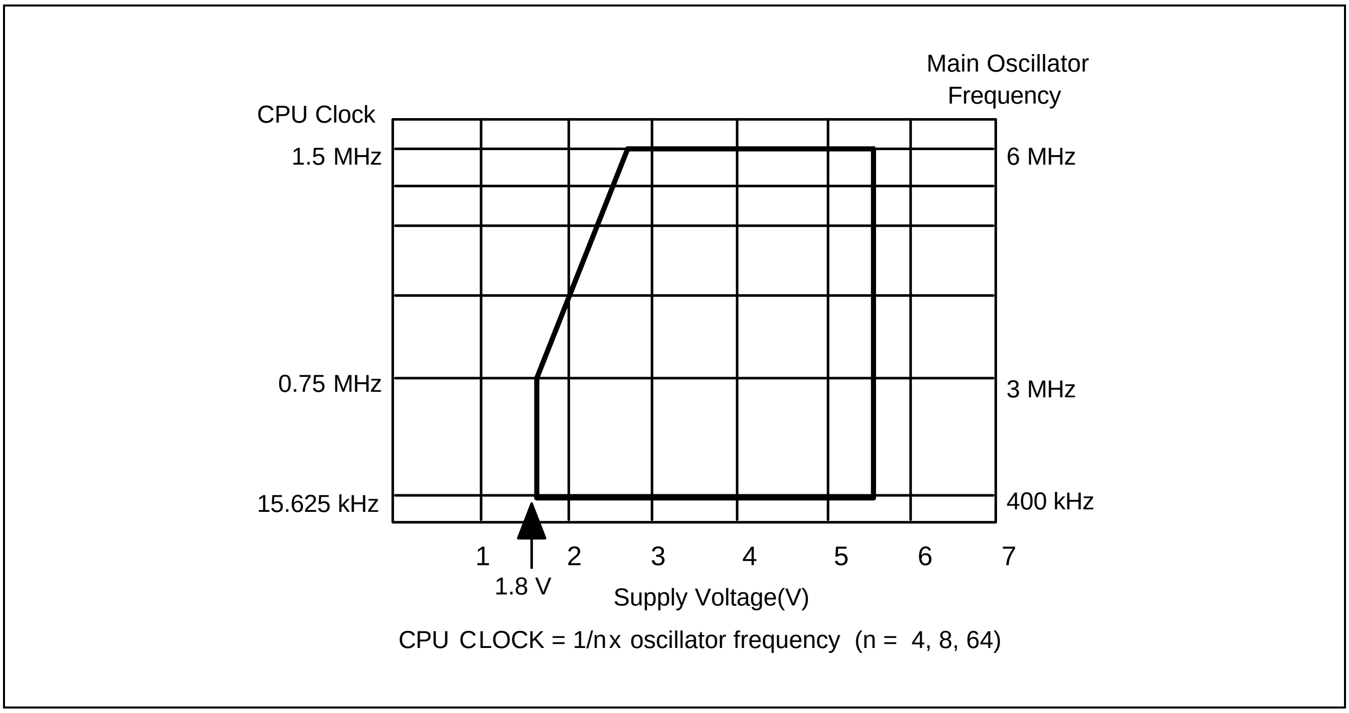


Figure 15-2. Standard Operating Voltage Range

NOTES

16 DEVELOPMENT TOOLS

OVERVIEW

Samsung provides a powerful and easy-to-use development support system in turnkey form. The development support system is configured with a host system, debugging tools, and support software. For the host system, any standard computer that operates with MS-DOS as its operating system can be used. One type of debugging tool including hardware and software is provided: the sophisticated and powerful in-circuit emulator, SMDS2+, for KS57, KS86, KS88 families of microcontrollers. The SMDS2+ is a new and improved version of SMDS2. Samsung also offers support software that includes debugger, assembler, and a program for setting options.

SHINE

Samsung Host Interface for In-Circuit Emulator, SHINE, is a multi-window based debugger for SMDS2+. SHINE provides pull-down and pop-up menus, mouse support, function/hot keys, and context-sensitive hyper-linked help. It has an advanced, multiple-windowed user interface that emphasizes ease of use. Each window can be sized, moved, scrolled, highlighted, added, or removed completely.

SAMA ASSEMBLER

The Samsung Arrangeable Microcontroller (SAM) Assembler, SAMA, is a universal assembler, and generates object code in standard hexadecimal format. Assembled program code includes the object code that is used for ROM data and required SMDS program control data. To assemble programs, SAMA requires a source file and an auxiliary definition (DEF) file with device specific information.

SASM57

The SASM57 is an relocatable assembler for Samsung's KS57-series microcontrollers. The SASM57 takes a source file containing assembly language statements and translates into a corresponding source code, object code and comments. The SASM57 supports macros and conditional assembly. It runs on the MS-DOS operating system. It produces the relocatable object code only, so the user should link object file. Object files can be linked with other object files and loaded into memory.

HEX2ROM

HEX2ROM file generates ROM code from HEX file which has been produced by assembler. ROM code must be needed to fabricate a microcontroller which has a mask ROM. When generating the ROM code (.OBJ file) by HEX2ROM, the value 'FF' is filled into the unused ROM area upto the maximum ROM size of the target device automatically.

TARGET BOARDS

Target boards are available for all KS57-series microcontrollers. All required target system cables and adapters are included with the device-specific target board.

OTPs

One time programmable microcontroller (OTP) for the KS57C21408/C21418 microcontroller and OTP programmer (Gang) are now available.

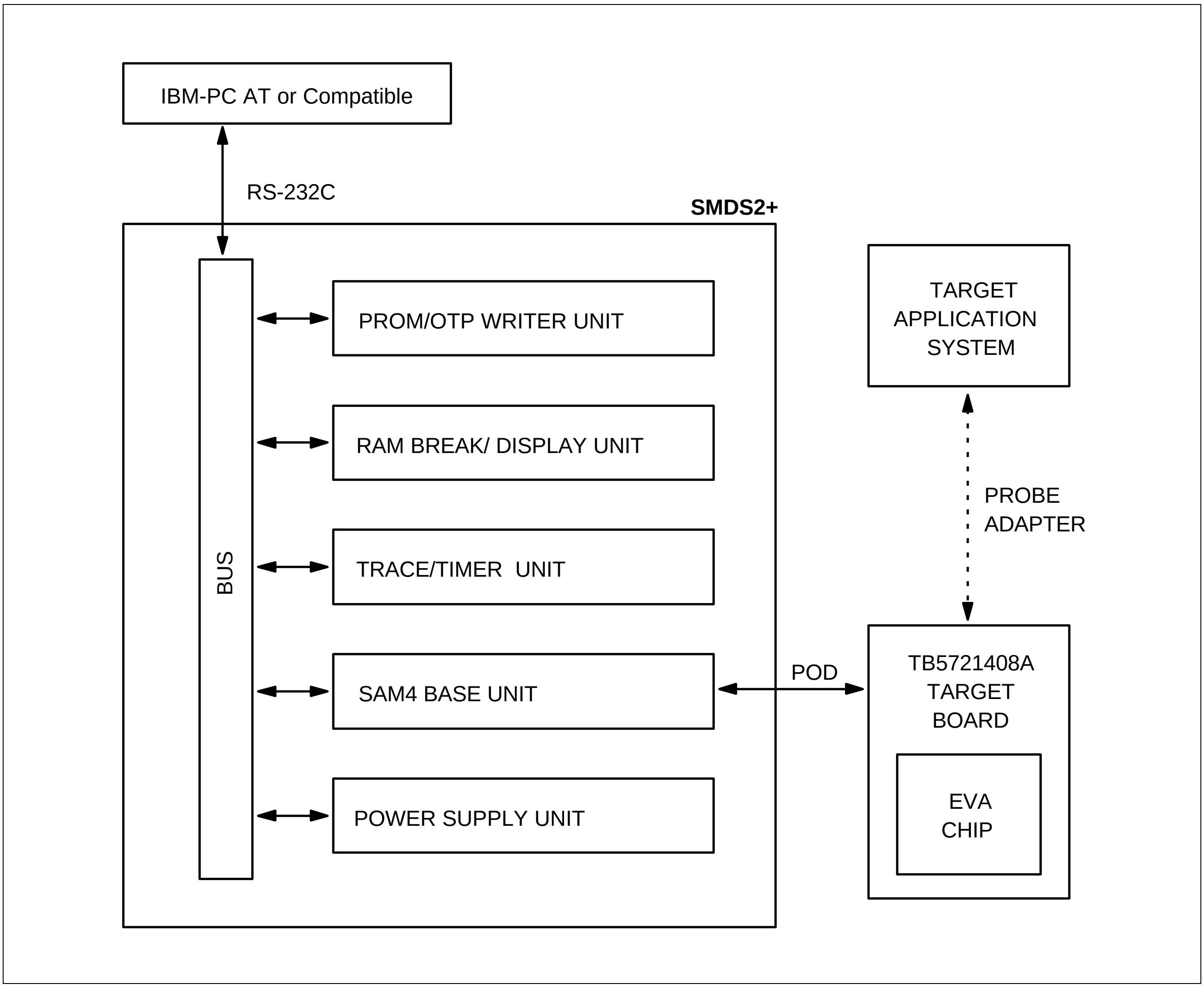


Figure 16-1. SMDS Product Configuration (SMDS2+)

TB5721408A TARGET BOARD

The TB5721408A target board is used for the KS57C21408/C21418/P21408 microcontroller. It is supported by the SMDS2+ development system.

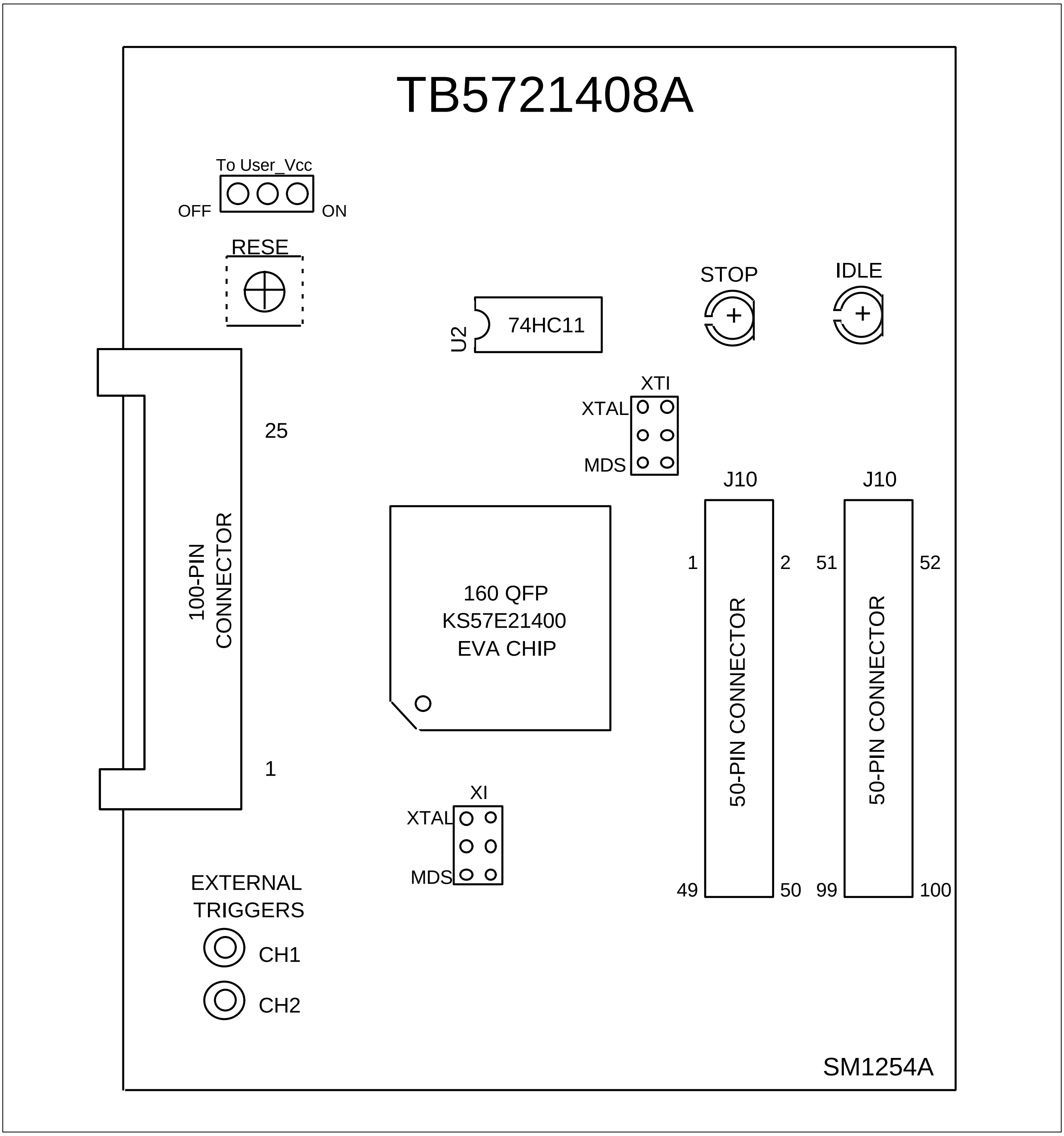


Figure 16-2. TB5721408A Target Board Configuration

Table 16-1. Power Selection Settings for TB5721408A

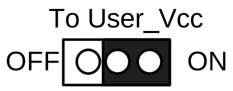
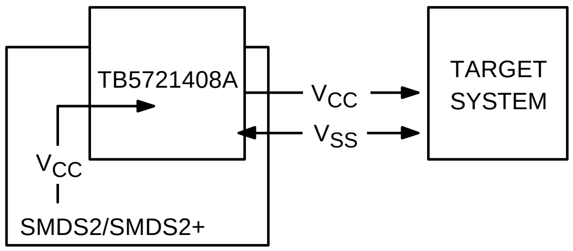
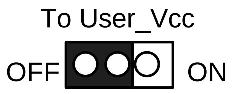
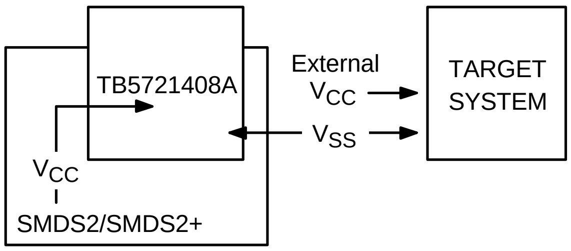
'To User_Vcc' Settings	Operating Mode	Comments
	 TB5721408A V_{CC} V_{SS} TARGET SYSTEM V_{CC} SMDS2/SMDS2+	The SMDS2/SMDS2+ supplies V_{CC} to the target board (evaluation chip) and the target system.
	 TB5721408A External V_{CC} V_{SS} TARGET SYSTEM V_{CC} SMDS2/SMDS2+	The SMDS2/SMDS2+ supplies V_{CC} only to the target board (evaluation chip). The target system must have its own power supply.

Table 16-2. Main-clock Selection Settings for TB5721408A

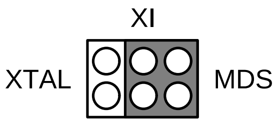
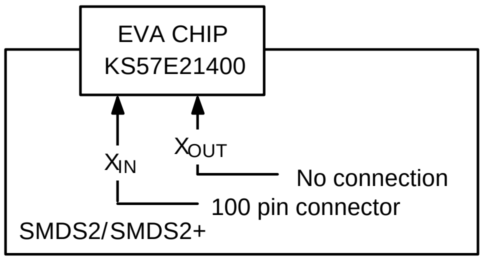
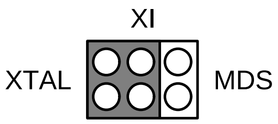
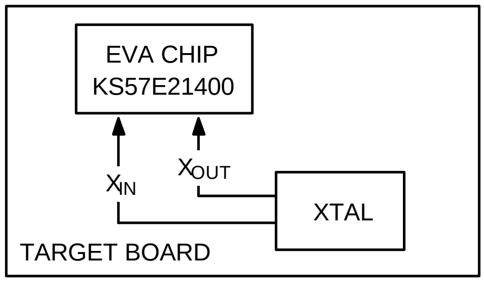
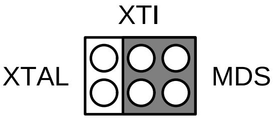
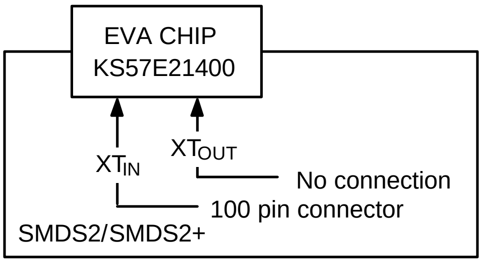
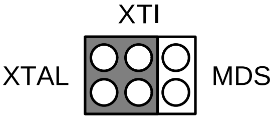
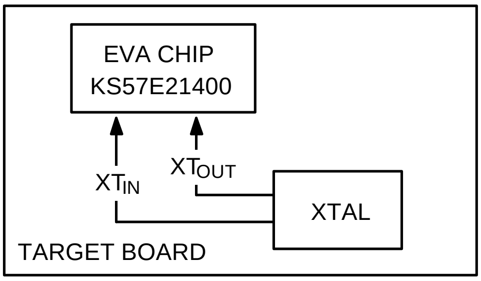
Sub Clock Setting	Operating Mode	Comments
	 EVA CHIP KS57E21400 X_{IN} X_{OUT} No connection 100 pin connector SMDS2/SMDS2+	Set the XI switch to "MDS" when the target board is connected to the SMDS2/SMDS2+.
	 EVA CHIP KS57E21400 X_{IN} X_{OUT} XTAL TARGET BOARD	Set the XI switch to "XTAL" when the target board is used as a standalone unit, and is not connected to the SMDS2/SMDS2+.

Table 16-3. Sub-clock Selection Settings for TB5721408A

Sub Clock Setting	Operating Mode	Comments
	 EVA CHIP KS57E21400 XT$_{IN}$ XT$_{OUT}$ 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 KS57E21400 XT$_{IN}$ XT$_{OUT}$ 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 (KS57E21400) is in idle mode.

STOP LED

This LED is ON when the evaluation chip (KS57E21400) is in stop mode.

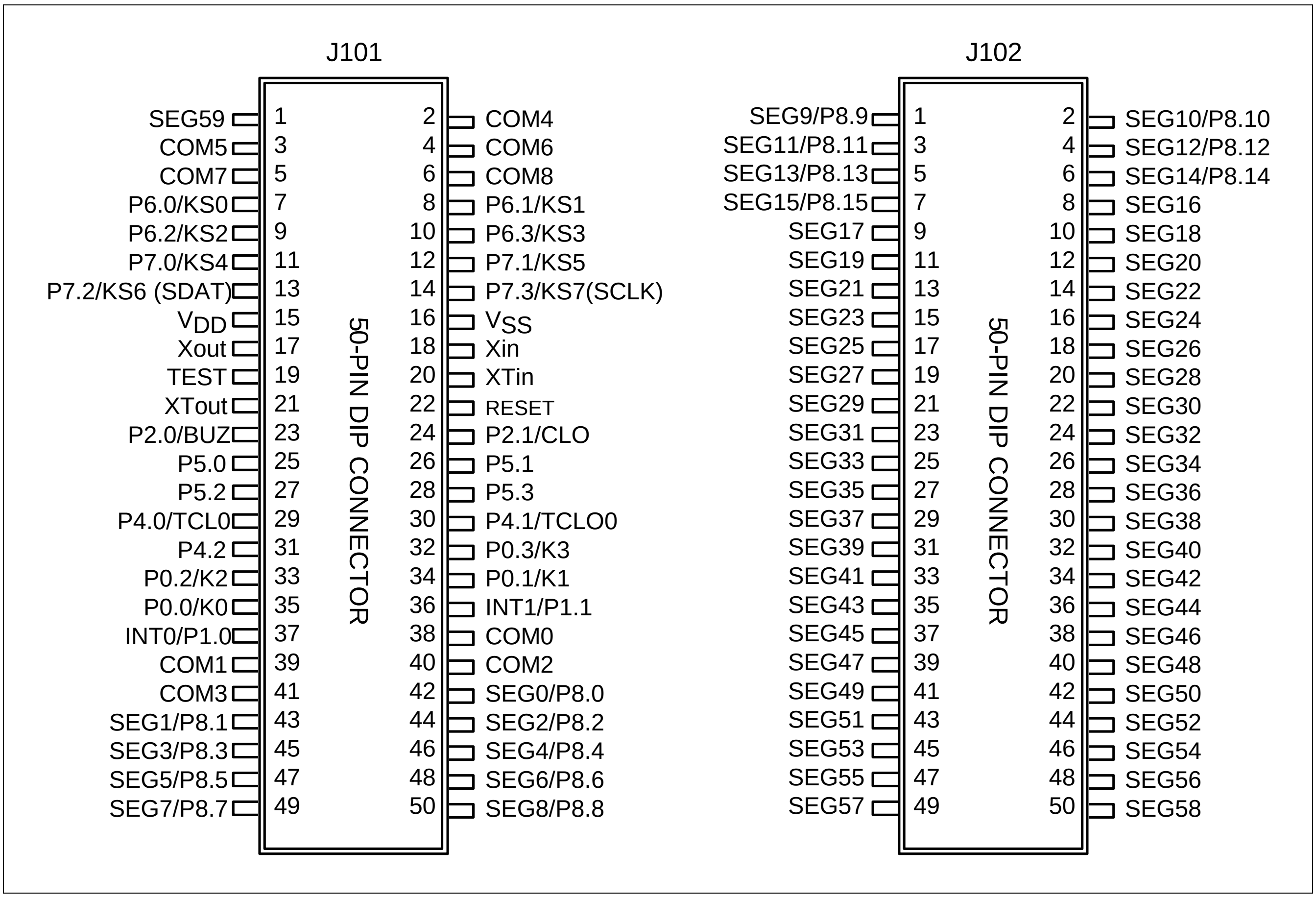


Figure 16-3. 50-Pin Connectors for TB5721408A

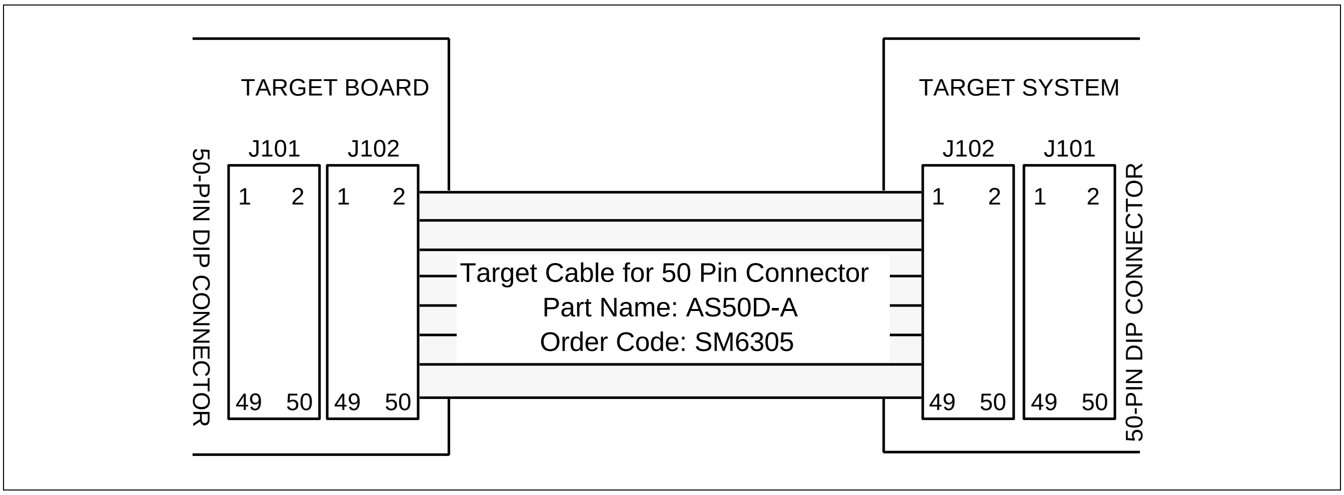


Figure 16-4. TB5721408A Adapter Cable for 100 QFP Package (KS57C21408/P21408)