

1

PRODUCT OVERVIEW

SAM8 PRODUCT FAMILY

Samsung's SAM87 family of 8-bit single-chip CMOS microcontrollers offers a fast and efficient CPU, a wide range of integrated peripherals, and various mask-programmable ROM sizes. Important CPU features include:

- Efficient register-oriented architecture
- Selectable CPU clock sources
- Idle and Stop power-down mode release by interrupt
- Built-in basic timer with watchdog function

A sophisticated interrupt structure recognizes up to eight interrupt levels. Each level can have one or more interrupt sources and vectors. Fast interrupt processing (within a minimum of six CPU clocks) can be assigned to specific interrupt levels.

KS88C0716/P0716 MICROCONTROLLERS

KS88C0716/P0716 single-chip 8-bit microcontrollers are based on the powerful SAM87 CPU architecture. The internal register file is logically expanded to increase the on-chip register space. The KS88C0716 has 16-Kbyte mask-programmable ROM. The KS88P0716 has 16-Kbyte one-time-programmable EPROM.

Following Samsung's modular design approach, the following peripherals are integrated with the SAM87 core:

- Seven programmable I/O ports (total 56 pins)
- One 8-bit basic timer for oscillation stabilization and watchdog functions
- One synchronous operating mode and three full-duplex asynchronous UART modes
- Two 8-bit timers with interval timer and PWM modes
- Two 16-bit general-purpose timer/counters

OTP

The KS88C0716 microcontroller is also available in OTP (One Time Programmable) version, KS88P0716. KS88P0716 microcontroller has an on-chip 16-Kbyte one-time-programmable EPROM instead of masked ROM. The KS88P0716 is comparable to KS88C0716, both in function and in pin configuration.

FEATURES

CPU

- SAM87 CPU core

Memory

- 272-byte general purpose register area
- 16-Kbyte internal program memory
- ROM-less operating mode

External Interface

- 64-Kbyte external data memory area
- 64-Kbyte external program memory area (ROM-less mode)

Instruction Set

- 78 instructions
- IDLE and STOP instructions for power-down mode

Instruction Execution Time

- 500 ns at 12 MHz f_{CPU} (Min.)

Interrupts

- 17 interrupt sources
- 17 interrupt vectors
- Eight interrupt levels
- Fast interrupt processing

General I/O

- Four nibble-programmable ports
- One bit-programmable port
- Two bit-programmable ports for external interrupts

Timers

- Two 8-bit timers with interval timer and PWM modes

Timer/Counters

- Two 16-bit general-purpose timer/counters

Basic Timer

- One 8-bit basic timer (BT) for oscillation stabilization control and watch dog timer function.

Serial Port

- One synchronous operating mode and three full-duplex asynchronous UART modes

Operating Temperature Range

- -40°C to $+85^{\circ}\text{C}$

Operating Voltage Range

- 2.7 V to 5.5 V

Package Types

- 64-pin SDIP, 64-pin QFP

Table 1-1. Comparison Table

Feature	KS88C0116	KS88C0716
Core	SAM8	SAM87
ROM	16 K bytes	Same
RAM	272 bytes	Same
I/O	54	56 (add two pins)
Port 6	Open drain (9 V drive)	Normal C-MOS output
I/O option	None	Same
Timer	8-bit back-up timer	None
	Timer A, B — 8-bit — Interval/PWM mode — Timer A match interrupt	Same (some differ in interval mode, see manual)
	Timer C, D — Gate function — Timer/counter	Same
Watchdog timer	None	Watchdog timer (with BT)
SIO	UART — 8-bit/9-bit UART — SIO	Same
Interrupt	External × 12 — P2.4–P2.7, P4.0–P4.7	Same
	Internal × 6 — Timer A, C, D, SI, SO, Back-up	Internal × 5 — Timer A, C, D, SI, SO
Power down	Stop/idle	Same
Oscillator	Crystal, ceramic	Same
CPU clock divider	1/2	1/1, 1/2, 1/8, 1/16
Execution time (Min.)	0.6 μ s at 20 MHz ($f_{CPU} = 10$ MHz)	0.5 μ s at 12 MHz ($f_{CPU} = 12$ MHz)
Operating frequency	Max. 20 MHz ($f_{CPU} = 10$ MHz)	Max. 12 MHz (at 4.5 V) ⁽²⁾ Max. 4 MHz (at 2.7 V)
Operating voltage	4.5–5.5 V	2.7–5.5 V at 4 MHz 4.5–5.5 V at 12 MHz
OTP/MTP	MTP	OTP
Pin assignment	—	Different
Package	64SDIP/64QFP	Same
Start address	0020h	0100h
P5CON, P6CON	BANK0	BANK1
Interrupt pending bit clear	Write "1"	Write "0"

NOTES:

1. The KS88C0716 can replace the KS88C0116. Their functions are mostly the same, but there are some differences. Table 1-1 shows the comparison of KS88C0716 and KS88C0116.
2. Operating frequency is maximum CPU clock; the maximum oscillation frequency is 22.1184 MHz.

BLOCK DIAGRAM

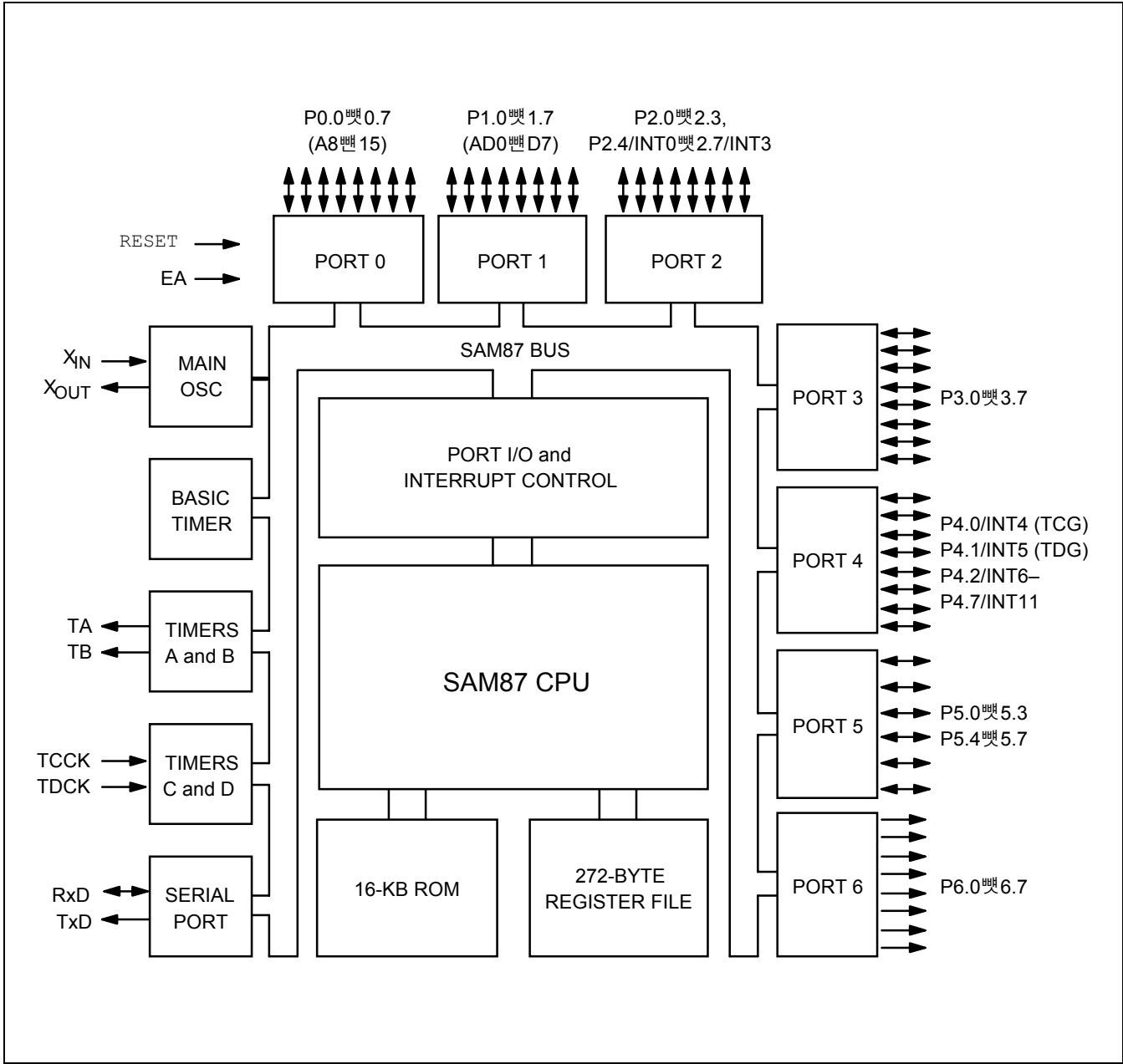


Figure 1-1. KS88C0716 Block Diagram

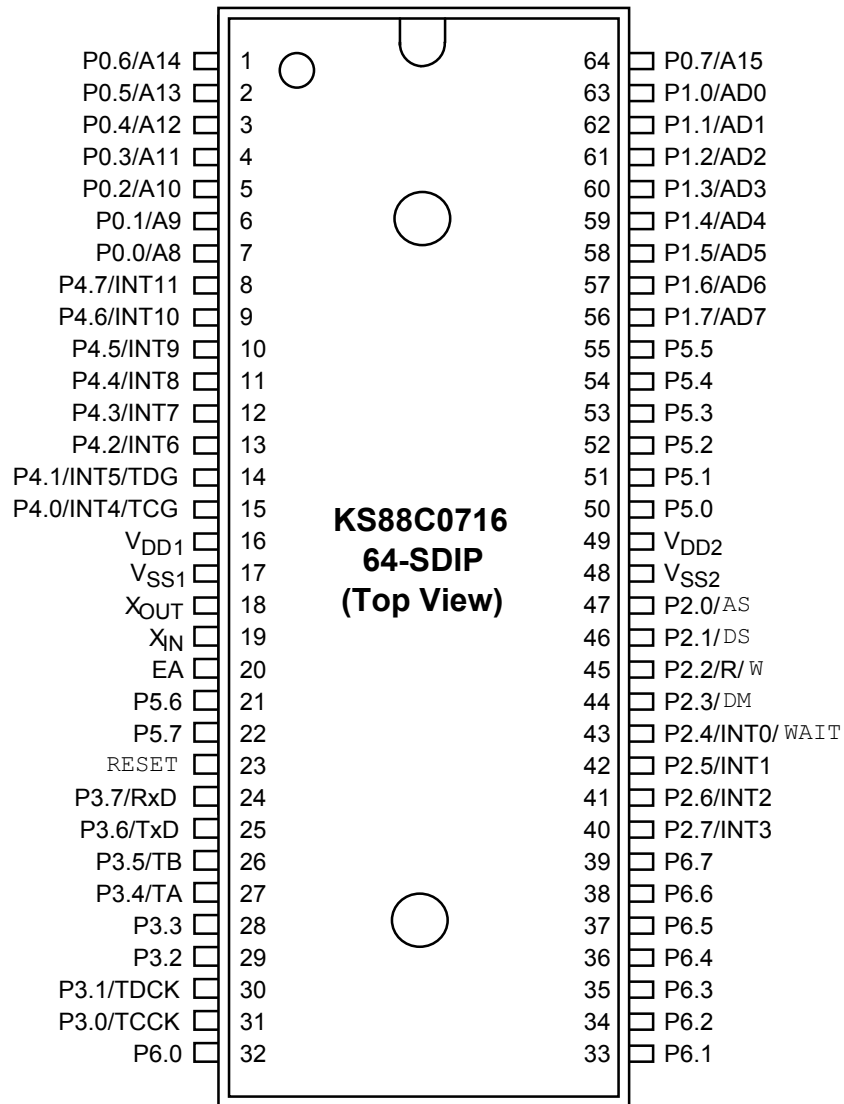


Figure 1-2. KS88C0716 Pin Assignments (64-SDIP)

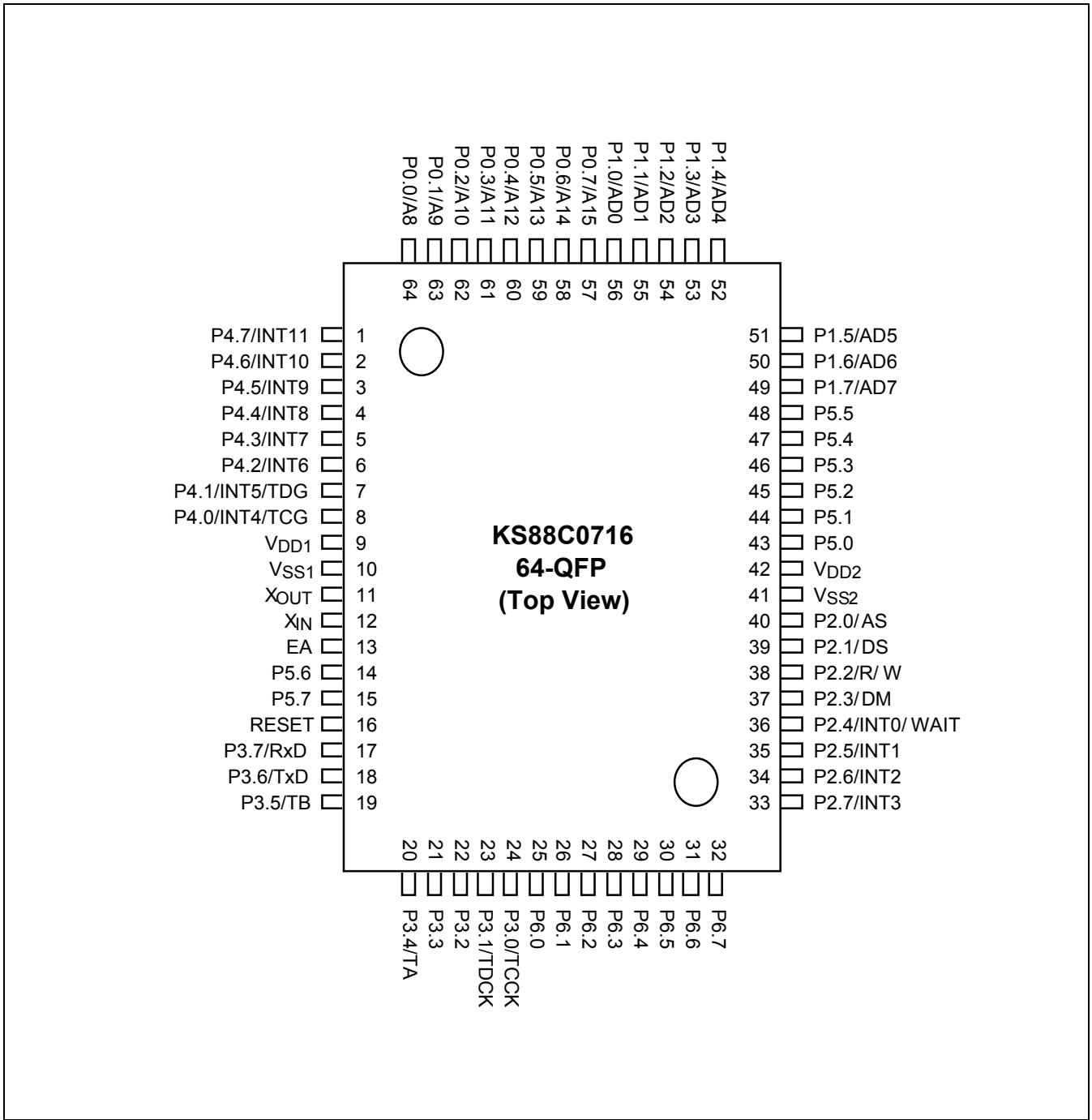


Figure 1-3. KS88C0716 Pin Assignments (64-QFP)

Table 1-2. KS88C0716 Pin Descriptions (64-SDIP)

Pin Name	Pin Type	Pin Description	Circuit Number	SDIP Pin Number	Share Pins
P0.0–P0.7	I/O	I/O port with nibble-programmable pins; Input or push-pull, open-drain output and software assignable pull-ups; also configurable as external interface address lines A8–A15.	E	1–7, 64	A8–A15
P1.0–P1.7	I/O	Same general characteristics as port 0; also configurable as external interface address/data lines AD0–AD7.	E	56–63	AD0–AD7
P2.0–P2.3 P2.4–P2.7	I/O	I/O port with bit-programmable pins; Input or push-pull output. Lower nibble pins 0–3 are configurable for external interface signals; upper nibble pins 4–7 are bit-programmable for external interrupts INT0–INT3. P2.4 can also be used for external WAIT input.	D-1 (lower nibble); D-1 (upper nibble; with noise filter)	40–47	AS, DS, DM, R/W INT0–INT3, WAIT
P3.0–P3.7	I/O	I/O port with bit-programmable pins; Input or push-pull output. Alternate functions include software-selectable UART transmit and receive on pins 3.7 and 3.6, timer B and timer A outputs at pins 3.5 and 3.4, and timer D and C clock inputs at pins 3.1 and 3.0.	D-1	24–31	TCCK, TDCK, TA, TB, TxD, RxD
P4.0–P4.7	I/O	I/O port with bit-programmable pins; Input or push-pull output; software-assignable pull-ups. Alternate functions include external interrupt inputs INT4–INT11 (with interrupt enable and pending control) and timer C and D gate input at P4.0 and P4.1.	D (with noise filter)	8–15	INT4–INT11, TCG, TDG
P5.0–P5.7	I/O	I/O port with nibble-programmable pins; Input or push-pull, open-drain output; software-assignable pull-ups.	E	21, 22, 50–55	–
P6.0–P6.7	O	Output port with nibble-programmable pins; push-pull, open-drain output; software-assignable pull-ups.	E-8	32–39	–
RxD	I/O	Bi-directional serial data input pin	–	24	P3.7
TxD	I/O	Serial data output pin	–	25	P3.6
TA, TB	I/O	Timer A and B output pins	4	27, 26	P3.4, P3.5
TCCK, TDCK	I/O	Timer C and D external clock input pins	D-1	30, 31	P3.0, P3.1
INT0–INT3	I/O	External interrupts. I/O pin 2.4 (share pin with INT0) is also configurable as a WAIT signal input pin for the external interface.	D-1 (with noise filter)	40–43	P2.4–P2.7

Table 1-2. KS88C0716 Pin Descriptions (Continued)

Pin Name	Pin Type	Pin Description	Circuit Number	SDIP Pin Number	Share Pins
INT4–INT11	I/O	Bit-programmable external interrupt input pins with interrupt pending and enable /disable control	D (with noise filter)	8–15	P4.0–P4.7
X_{IN} , X_{OUT}	–	System clock input and output pins	–	18, 19	–
RESET	I	System reset pin (internal pull-up: 280 K Ω)	B	23	–
EA	I	External access (EA) pin with three modes: 0 V: Normal operation (internal ROM) 5 V: ROM-less operation (external interface)	–	20	–
V_{DD2} , V_{SS2}	–	Power input pins for port output (external)	–	49, 48	–
V_{DD1} , V_{SS1}	–	Power input pins for CPU (internal)	–	16, 17	–

PIN CIRCUIT

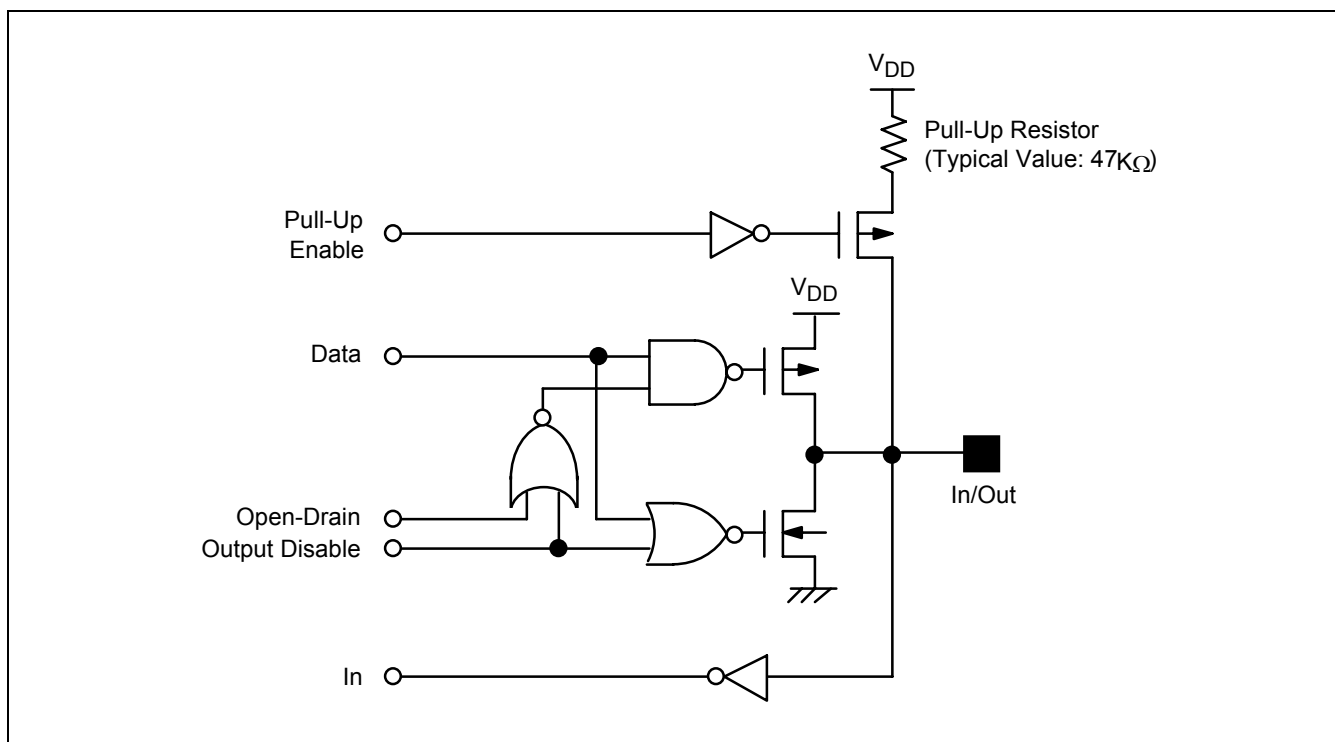


Figure 1-4. Pin Circuit Type E (Ports 0, 1, 5)

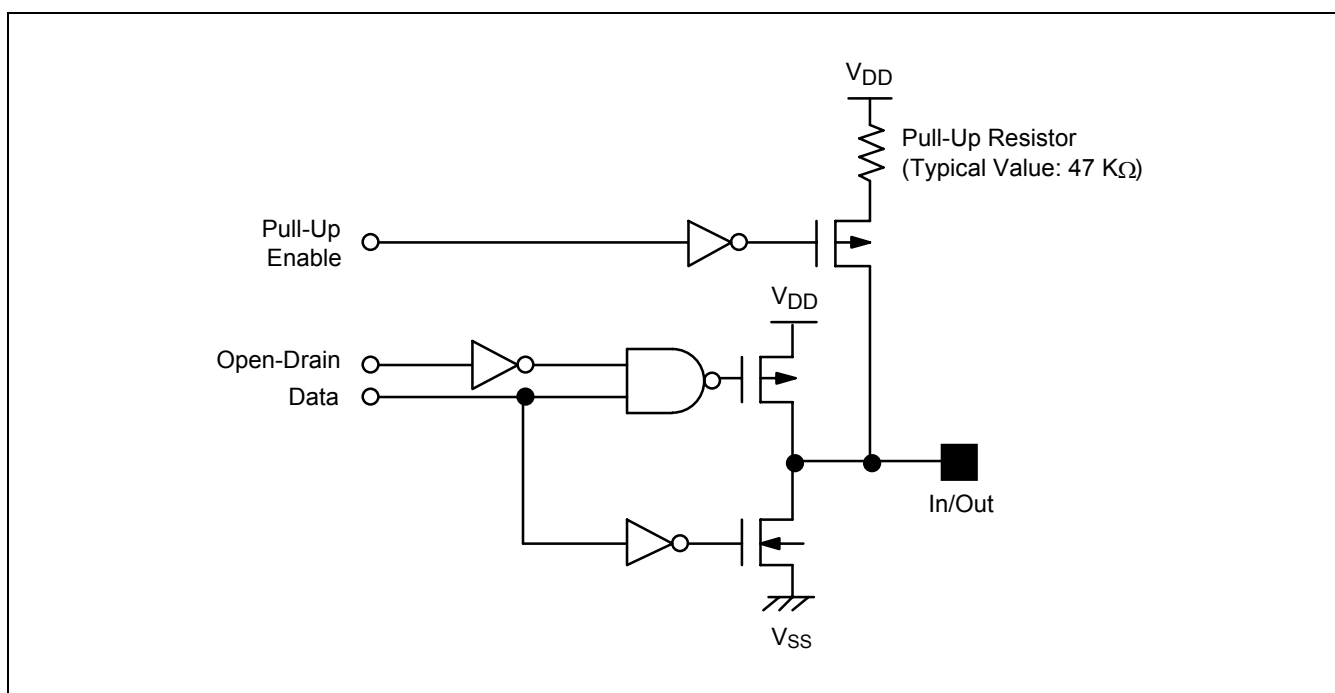


Figure 1-5. Pin Circuit Type E-8 (Ports 6)

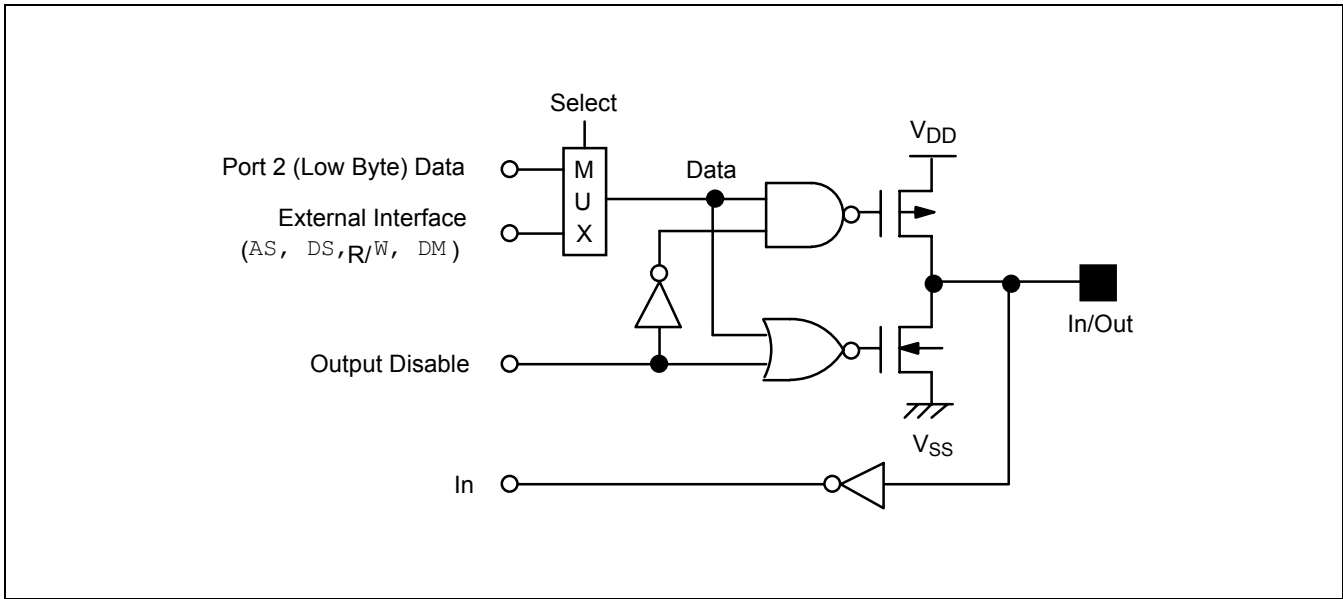


Figure 1-6. Pin Circuit Type D-1 (P2.0-P2.3)

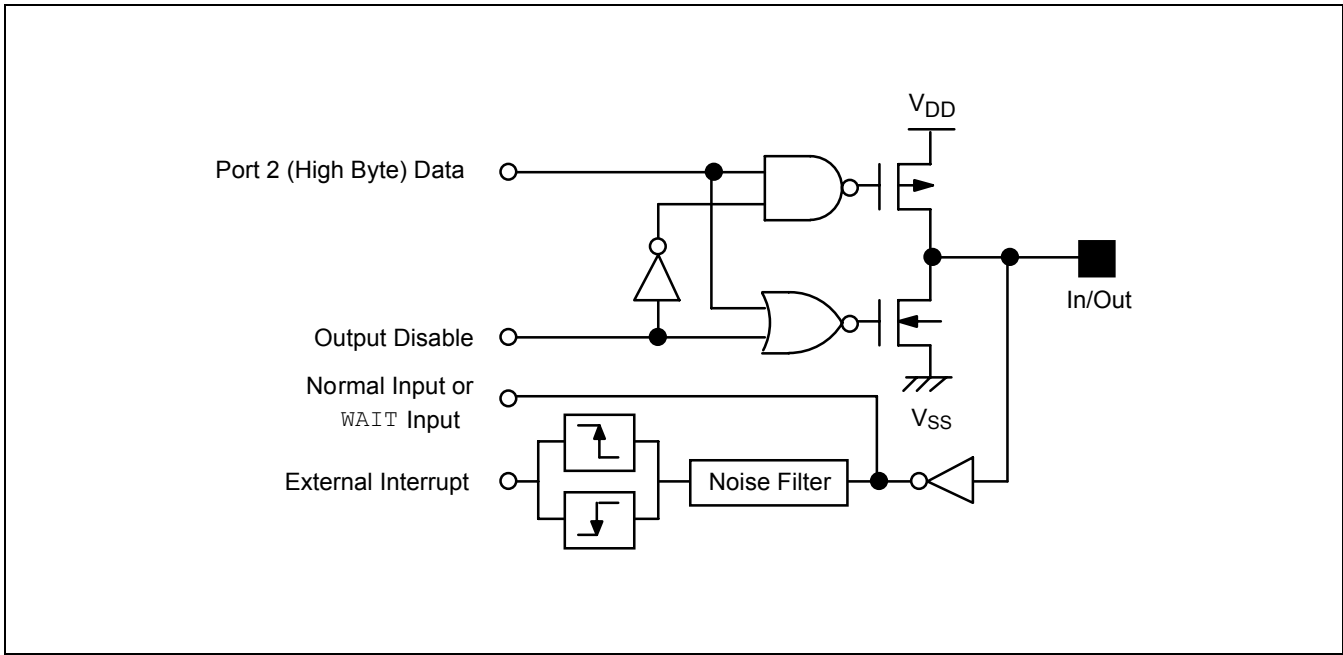


Figure 1-7. Pin Circuit Type D-1 (P2.4-P2.7)

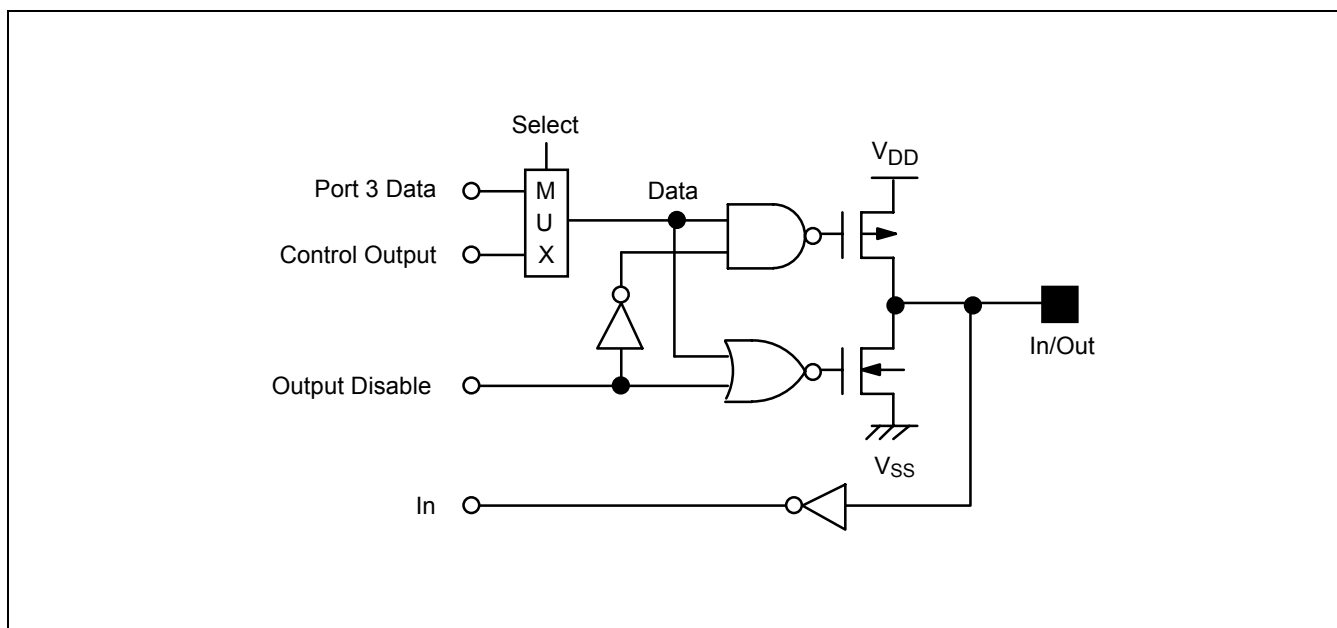


Figure 1-8. Pin Circuit Type D-1 (Port 3)

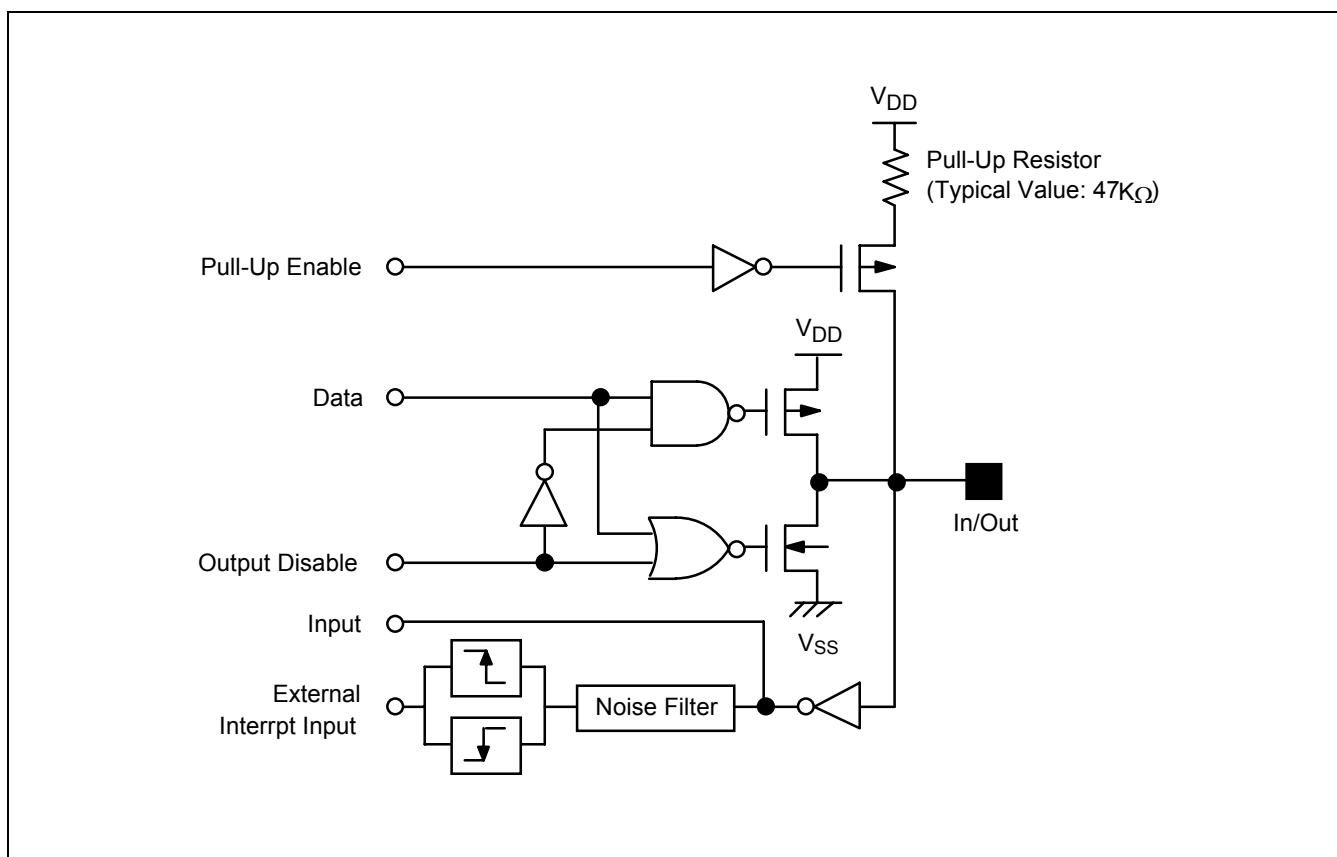


Figure 1-9. Pin Circuit Type D (Port 4)

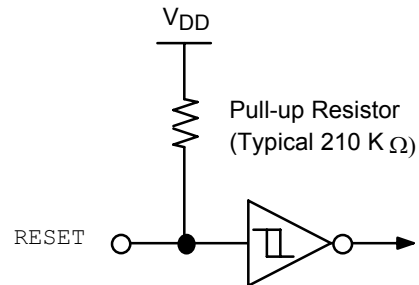


Figure 1-10. Pin Circuit Type B (RESET)

2 ADDRESS SPACES

OVERVIEW

KS88C0716 microcontroller has three kinds of address space:

- Program memory (internal and/or external)
- Internal register file
- External data memory

A 16-bit address bus and an 8-bit data bus support program memory and data memory operations. A separate 8-bit address bus and an 8-bit data bus carry addresses and data between the CPU and the register file. The SAM87 bus architecture therefore supports up to 64 K bytes of program memory. KS88C0716 has 16 K bytes of mask-programmable program memory on-chip.

There are two ROM configuration options: normal internal ROM mode and ROM-less mode.

KS88C0716 microcontroller has 272 general-purpose registers in its internal register file. Forty-nine bytes of the register file are mapped for system and peripheral control functions.

PROGRAM MEMORY (ROM)

Normal Operating Mode (Internal ROM)

KS88C0716 has 16 K bytes (locations 0H–3FFFH) of internal mask-programmable program memory. For normal internal ROM operation, the EA pin should be connected to V_{SS}.

The first 256 bytes of the ROM (0H–FFH) are reserved as an interrupt vector area. Unused locations in this address range can be used as normal program memory.

The reset address in the ROM is 0100H.

ROM-Less Operating Mode (External ROM)

For applications that require more than 16 K bytes of program memory, ROM-less operating mode can be used to configure an external area of up to 64-Kbyte. Access to the internal 16-Kbyte program memory area is disabled in ROM-less mode.

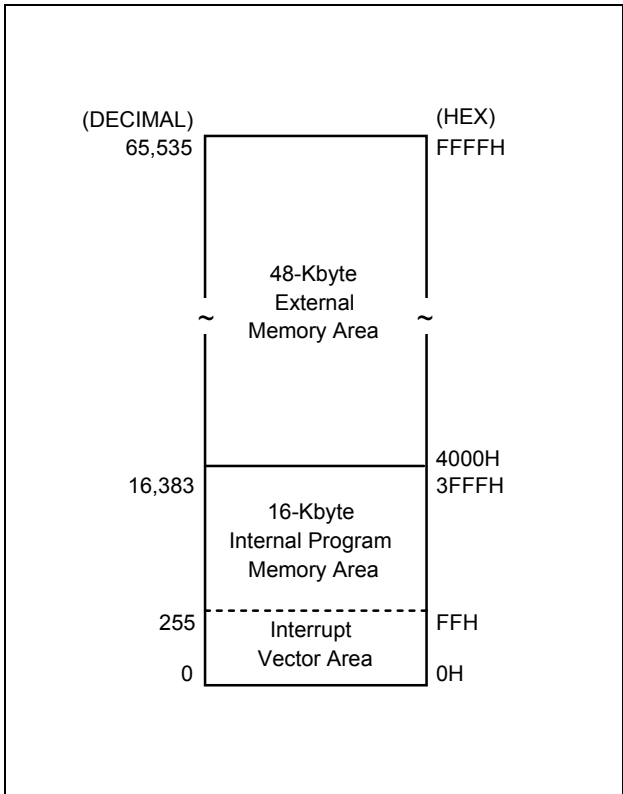
In normal operating mode, it is also possible to access external program memory of up to 48-Kbyte through the external memory interface. The 16-Kbyte on-chip ROM is accessed when program memory locations 0H–3FFFH are addressed at 0H–3FFFH and the external interface is used whenever locations are addressed 4000H–FFFFH. This configuration however, may not, be practical or cost-effective.

Mode selection (internal ROM or ROM-less) depends on the voltage applied to the EA pin during a reset operation:

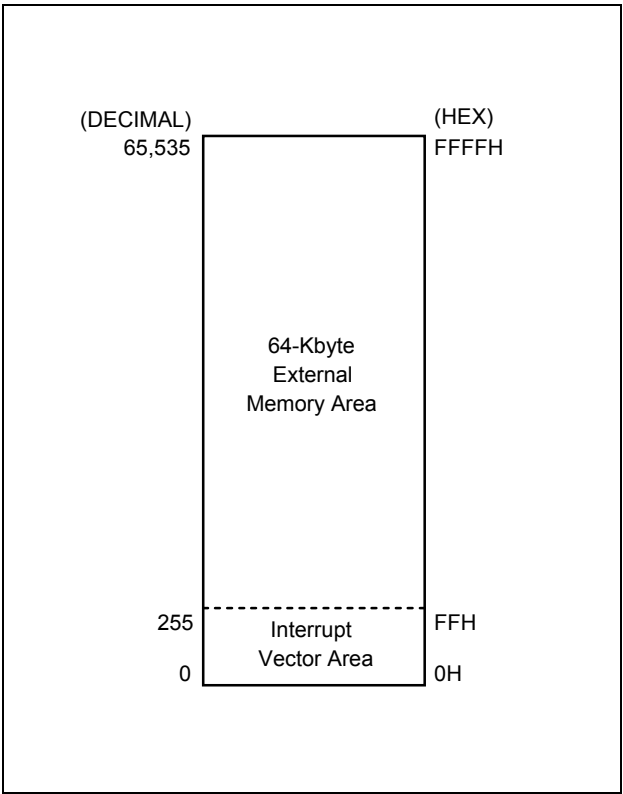
- When 0 V is applied to the EA pin, the KS88C0716's internal ROM is configured normally and the 16-Kbyte space (0H–3FFFH) is addressed.
- When 5 V is applied to the EA pin, KS88C0716 operates in ROM-less mode. External memory locations 0000H–FFFFH are accessed over the 16-bit address/data bus.

When 5 V is applied to the EA pin during a power-on reset, the external peripheral interface is automatically configured as follows:

- Port 0 and port 1 control registers are cleared to their initial value (00H), but the corresponding address and data lines are configured.
- The lower-nibble pins of port 2 (P2CONL) are set to '11B', configuring the interface signals (DM, R/W, DS, and AS) at P2.0–P2.3.



**Figure 2-1. Program Memory Map
in Normal Operating Mode (EA = "0")**



**Figure 2-2. Program Memory Map
in ROM-less Operating Mode (EA = "1")**

REGISTER ARCHITECTURE (RAM)

INTERNAL REGISTER FILE

The 256-byte register file is logically extended by duplicating its upper 64-byte area. This creates two 64-byte address space, set1 and set2. The duplication divides the upper 32 bytes of set1 into two parts (bank0 and bank1).

This register file extension is supported by specific addressing mode restrictions. The total addressable internal register space is thereby expanded from 256 bytes to 352 bytes.

KS88C0716 can access 321 8-bit registers in the 352-byte space. Of the 272 general-purpose registers, 208 can be used as accumulators, address pointers, index registers, data registers, or stack registers.

Table 2-1. Register Type Summary

Register Type	Number of Bytes
General-purpose registers (Including the 16-byte common working registers)	272
CPU and system control register (Including system and peripheral control registers)	47 (bank0) 2 (bank1)
Total address bytes	321

REGISTER PAGE POINTER (PP)

For many SAM87 microcontrollers, the addressable area of the internal register file can further be expanded by a register page implementation. This register file expansion, however, is not used for KS88C0716.

Page addressing is normally controlled by the register page pointer (PP, DFH). As the page pointer is not used in the KS88C0716 implementation, it always points to page 0.

REGISTER SET 1

The term *set 1* refers to the upper 64 bytes of the internal register file (C0H–FFH). The upper 32-byte (E0H–FFH) is divided into two 32-byte register banks (bank0 and bank1). Of these 96 bytes, the upper 80 bytes [(D0H–FFH) + (E0–FFH)] contain system and peripheral control registers. Of these 80 bytes, KS88C0716 use 49 byte registers.

The lower 16 bytes (C0H–CFH) of set 1 are used for working register addressing. This 16-byte area is called the *working register common area* because registers in these locations can serve as buffers for data transfers between register locations.

The register pointers RP0 and RP1 always point to the working register common area after a reset operation. You can access set 1 registers at all times using the register addressing mode. You can access the 16-byte working register common area using working register addressing only.

Section 2 of this document explains the difference between register addressing mode and working register addressing.

REGISTER SET 2

The same 64-byte physical space that is used for set 1 register locations C0H–FFH is logically duplicated to add 64 bytes of register space. This expanded area of the register file is called *set 2*.

The logical division of set 1 and set 2 is maintained by means of addressing mode restrictions: Set 1 can be accessed using Register addressing mode only; set 2 can only be accessed indirectly using Register Indirect or Indexed addressing mode.

192-BYTE PRIME REGISTER SPACE

The lower 192 bytes of the register file (00H - BFH) is called the *prime register space*. You can access registers in these locations at any time using any addressing mode. Registers in the prime register space are immediately addressable after a reset.

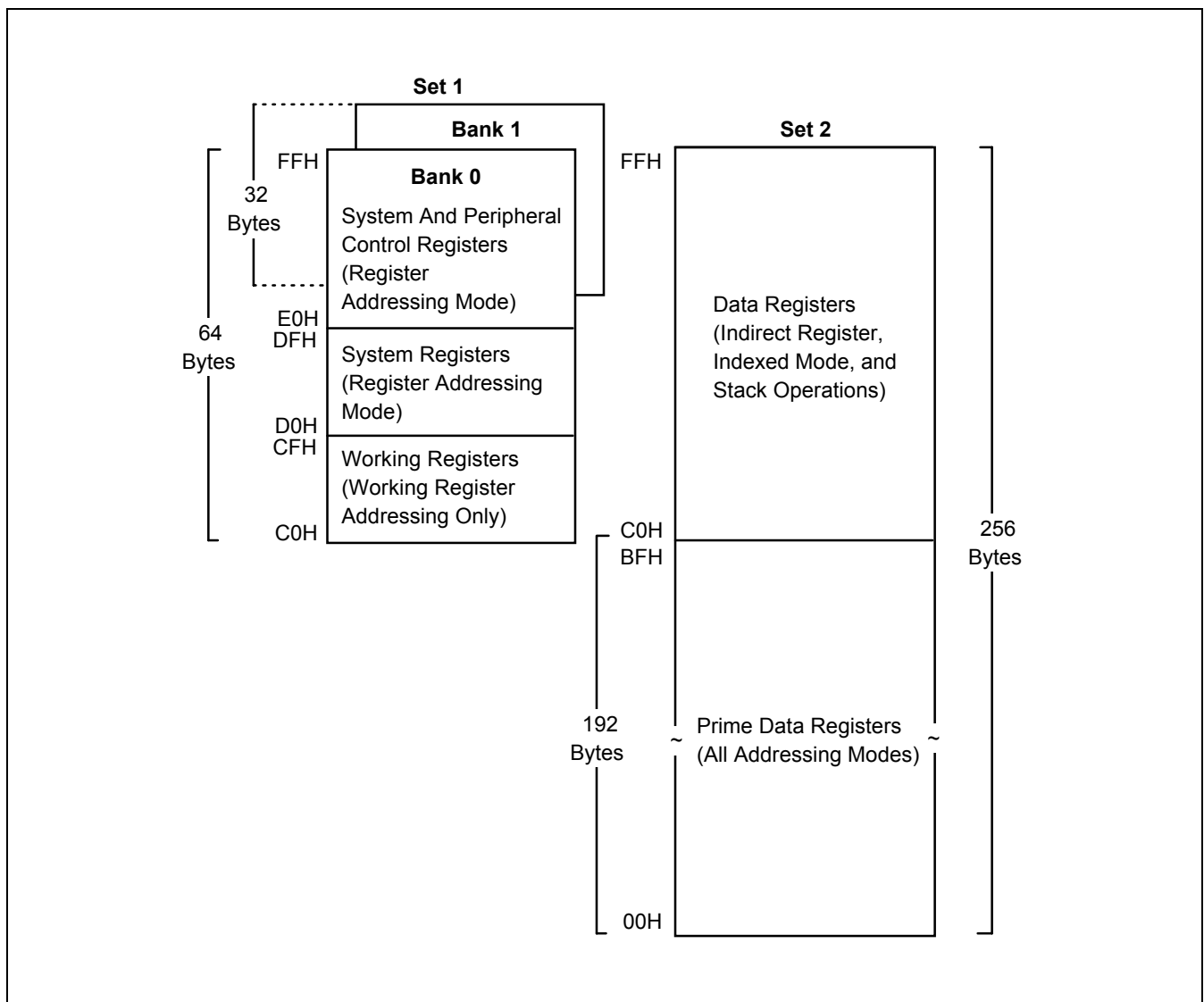


Figure 2-3. KS88C0716 Register File

PRIME REGISTER SPACE

The lower 192 bytes of the 256-byte physical internal register file (00H - BFH) is called the *prime register space* or, more simply, the *prime area*. You can access registers in this address range as page 0, and using any of the seven explicit addressing modes (see Section 3, Addressing Modes). All registers in the prime area are addressable immediately following a reset.

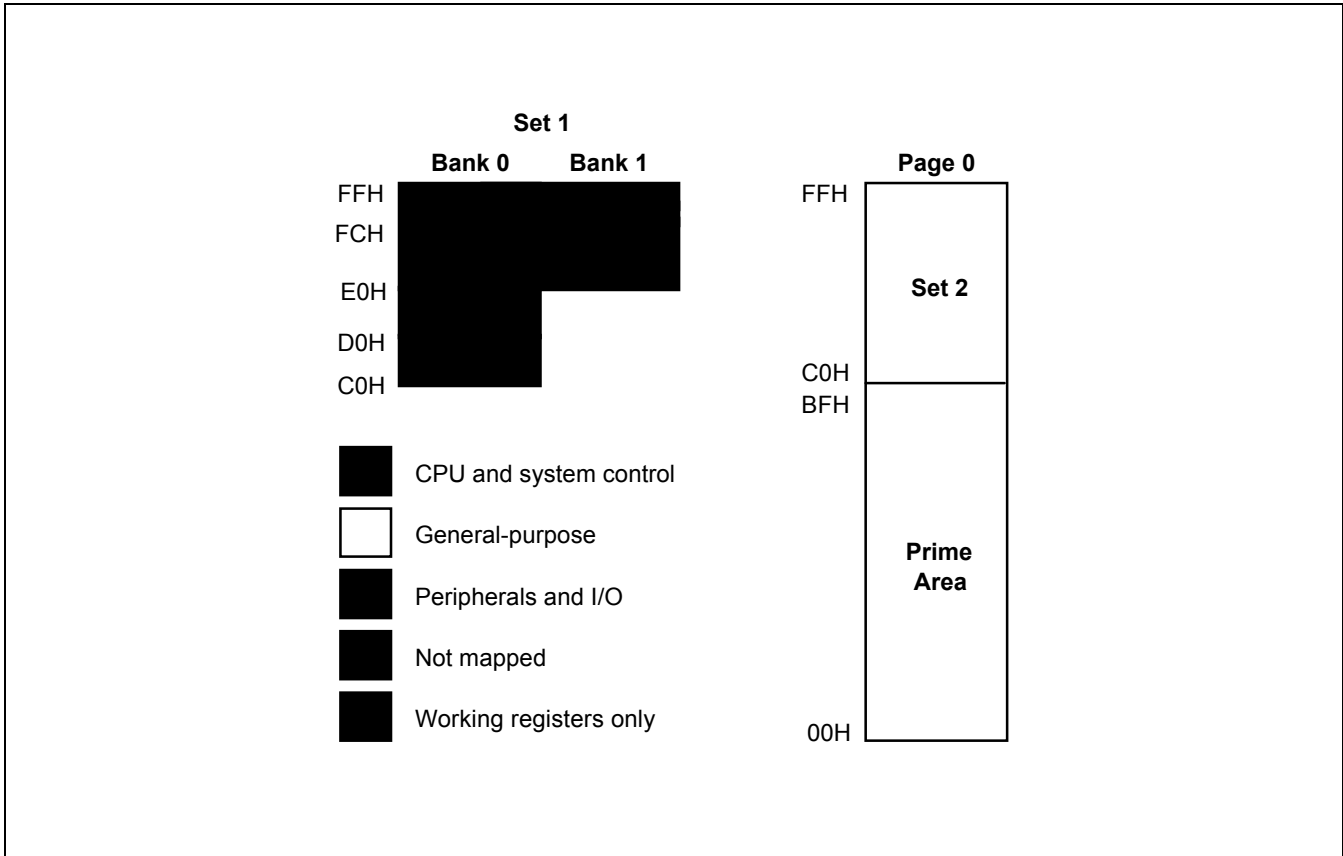


Figure 2-4. Set 1, Set 2, and Prime Area Register Map

WORKING REGISTERS

Instructions can access specific 8-bit registers or 16-bit register pairs using either 4-bit or 8-bit address fields. When 4-bit working register addressing is used, the 256-byte register file can be viewed by the programmer as 32 8-byte register groups or "slices". Each slice consists of eight 8-bit registers.

Using the two 8-bit register pointers, RP1 and RP0, two working register slices can be selected at any time to form a 16-byte working register block. Using the register pointers, you can move this 16-byte register block to anywhere in the addressable register file, except for the set 2 area.

The terms *slice* and *block* are used in this manual to help you visualize the size and relative locations of selected working register spaces:

- One working register *slice* is 8 bytes (eight 8-bit working registers; R0–R7 or R8–R15)
- One working register *block* is 16 bytes (sixteen 8-bit working registers; R0–R15)

All the registers in an 8-byte working register slice have the same binary value for their five most significant address bits. This makes it possible for each register pointer to point to one of the 24 slices in the register file. The base addresses for the two selected 8-byte register slices are contained in register pointers RP0 and RP1.

After a reset, RP0 and RP1 always point to the 16-byte common area in set 1 (C0H–CFH).

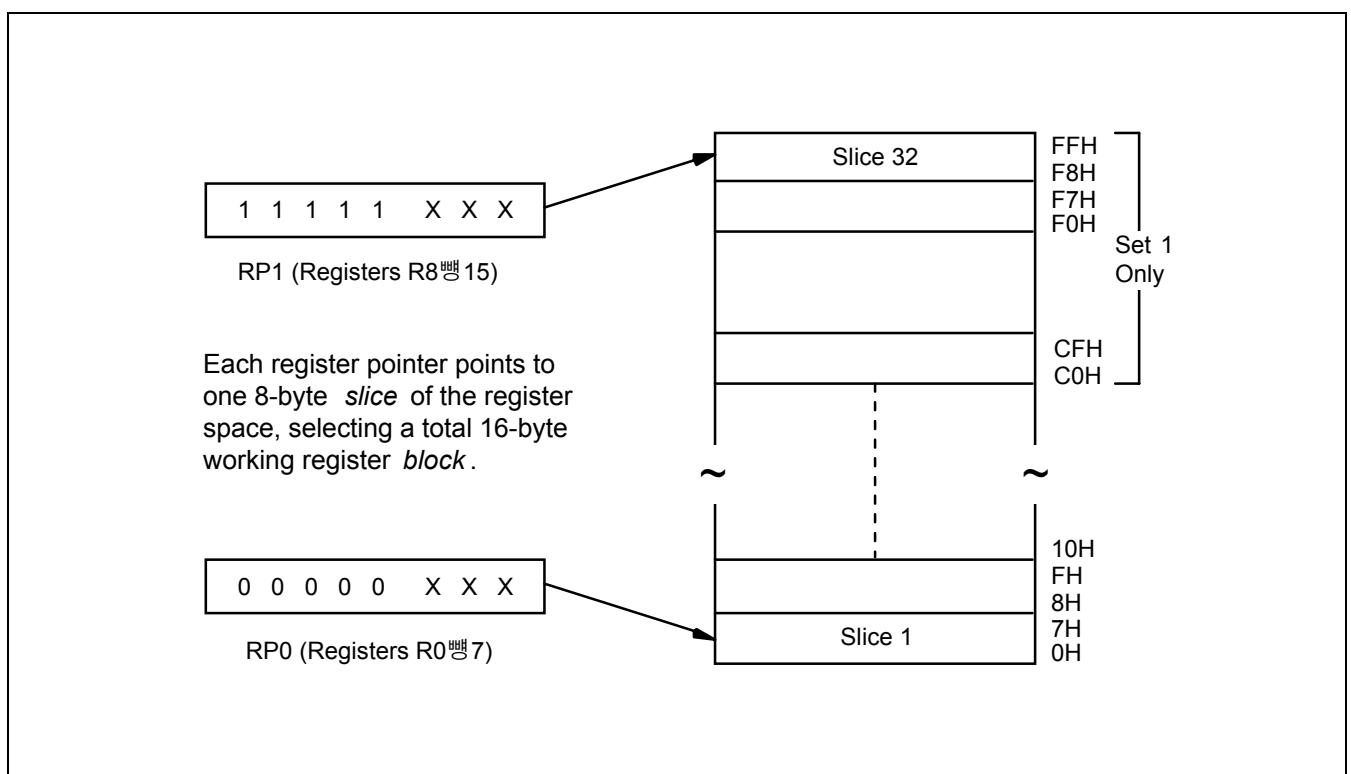


Figure 2-5. 8-Byte Working Register Areas (Slices)

USING THE REGISTER POINTERS

Register pointers RP0 and RP1, mapped to addresses D6H and D7H in set 1, are used to select two movable 8-byte working register slices in the register file. After a reset, they point to the working register common area: RP0 points to addresses C0H–C7H, and RP1 points to addresses C8H–CFH.

To change a register pointer value, you load a new value to RP0 and/or RP1 using an SRP or LD instruction (see Figures 2-6 and 2-7).

With working register addressing, you can only access those two 8-bit slices of the register file that are currently pointed to by RP0 and RP1. You cannot, however, use the register pointers to select a working register area in set 2, C0H–FFH, because these locations can be accessed only using the Indirect Register or Indexed addressing modes.

The selected 16-byte working register block usually consists of two contiguous 8-byte slices. As a general programming guideline, we recommend that RP0 point to the "lower" slice and RP1 point to the "upper" slice (see Figure 2-6). In some cases, it may be necessary to define working register areas in different (non-contiguous) areas of the register file. In Figure 2-7, RP0 points to the "upper" slice and RP1 to the "lower" slice.

Because a register pointer can point to either of the two 8-byte slices in the working register block, you can define the working register area very flexibly to support program requirements.

PROGRAMMING TIP — Setting the Register Pointers

SRP	#70H	; RP0 ← 70H, RP1 ← 78H
SRP1	#48H	; RP0 ← no change, RP1 ← 48H,
SRP0	#0A0H	; RP0 ← A0H, RP1 ← no change
CLR	RP0	; RP0 ← 00H, RP1 ← no change
LD	RP1,#0F8H	; RP0 ← no change, RP1 ← 0F8H

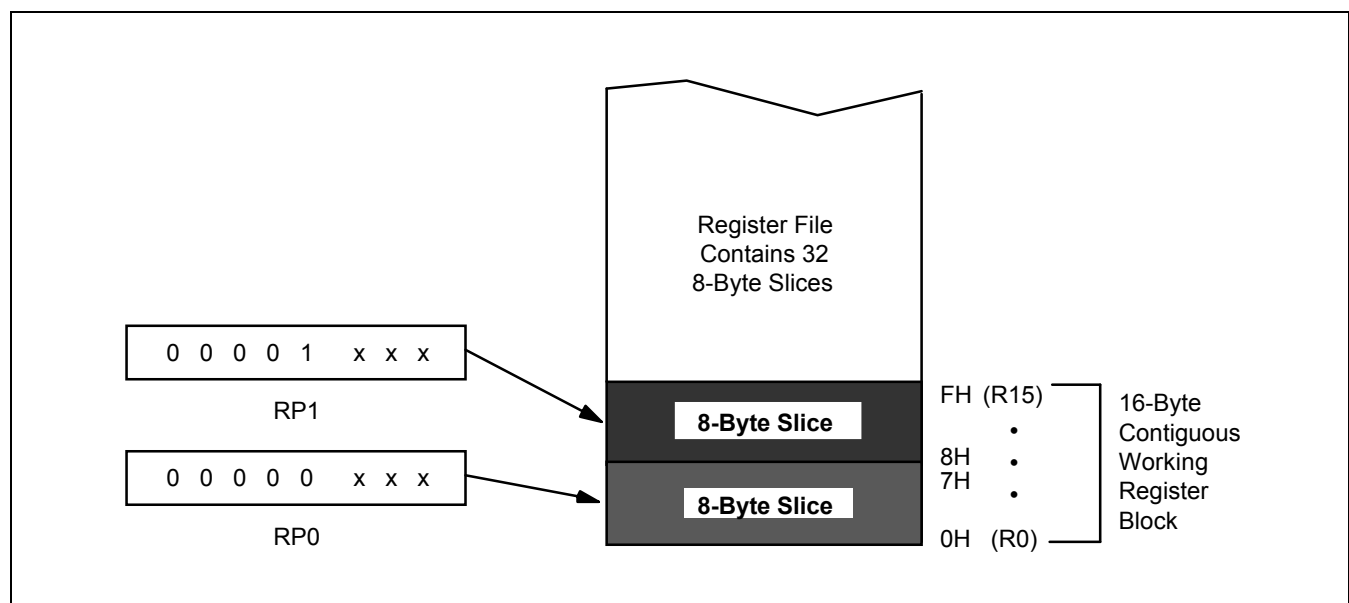


Figure 2-6. Contiguous 16-Byte Working Register Block

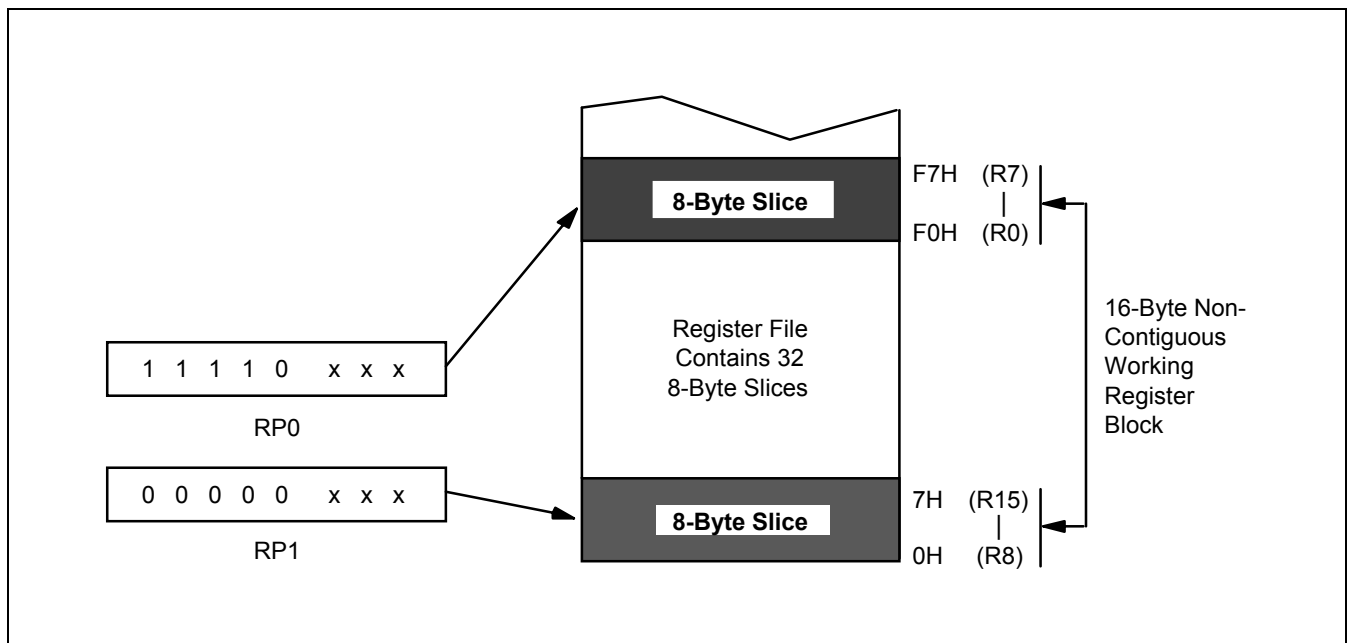


Figure 2-7. Non-Contiguous 16-Byte Working Register Block

PROGRAMMING TIP — Using the RPs to Calculate the Sum of a Series of Registers

Calculate the sum of registers 80H–85H using the register pointer and working register addressing. The register addresses 80H through 85H contain the values 10H, 11H, 12H, 13H, 14H, and 15 H, respectively:

SRP0	#80H	; RP0 ← 80H
ADD	R0,R1	; R0 ← R0 + R1
ADC	R0,R2	; R0 ← R0 + R2 + C
ADC	R0,R3	; R0 ← R0 + R3 + C
ADC	R0,R4	; R0 ← R0 + R4 + C
ADC	R0,R5	; R0 ← R0 + R5 + C

The sum of these six registers, 6FH, is located in the register R0 (80H). The instruction string used in this example takes 12 bytes of instruction code and its execution time is 36 cycles. If the register pointer is not used to calculate the sum of these registers, the following instruction sequence would have to be used:

ADD	80H,81H	; 80H ← (80H) + (81H)
ADC	80H,82H	; 80H ← (80H) + (82H) + C
ADC	80H,83H	; 80H ← (80H) + (83H) + C
ADC	80H,84H	; 80H ← (80H) + (84H) + C
ADC	80H,85H	; 80H ← (80H) + (85H) + C

Now, the sum of the six registers is also located in register 80H. However, this instruction string takes 15 bytes of instruction code instead of 12 bytes, and its execution time is 50 cycles instead of 36 cycles.

REGISTER ADDRESSING

The SAM8 register architecture provides an efficient method of working register addressing that takes full advantage of shorter instruction formats to reduce execution time.

With Register (R) addressing mode, in which the operand value is the content of a specific register or register pair, you can access all locations in the register file except for set 2. With working register addressing, you use a register pointer to specify an 8-byte working register space in the register file and an 8-bit register within that space.

Registers are addressed either as a single 8-bit register or as a paired 16-bit register space. In a 16-bit register pair, the address of the first 8-bit register is always an even number and the address of the next register is always an odd number. The most significant byte of the 16-bit data is always stored in the even-numbered register; the least significant byte is always stored in the next (+ 1) odd-numbered register.

Working register addressing differs from Register addressing because it uses a register pointer to specify an 8-byte working register space in the register file and an 8-bit register within that space (see Figure 2-8).

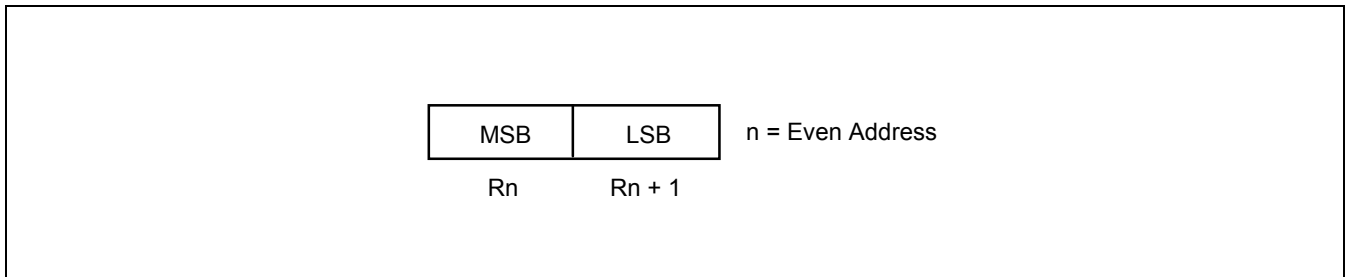


Figure 2-8. 16-Bit Register Pair

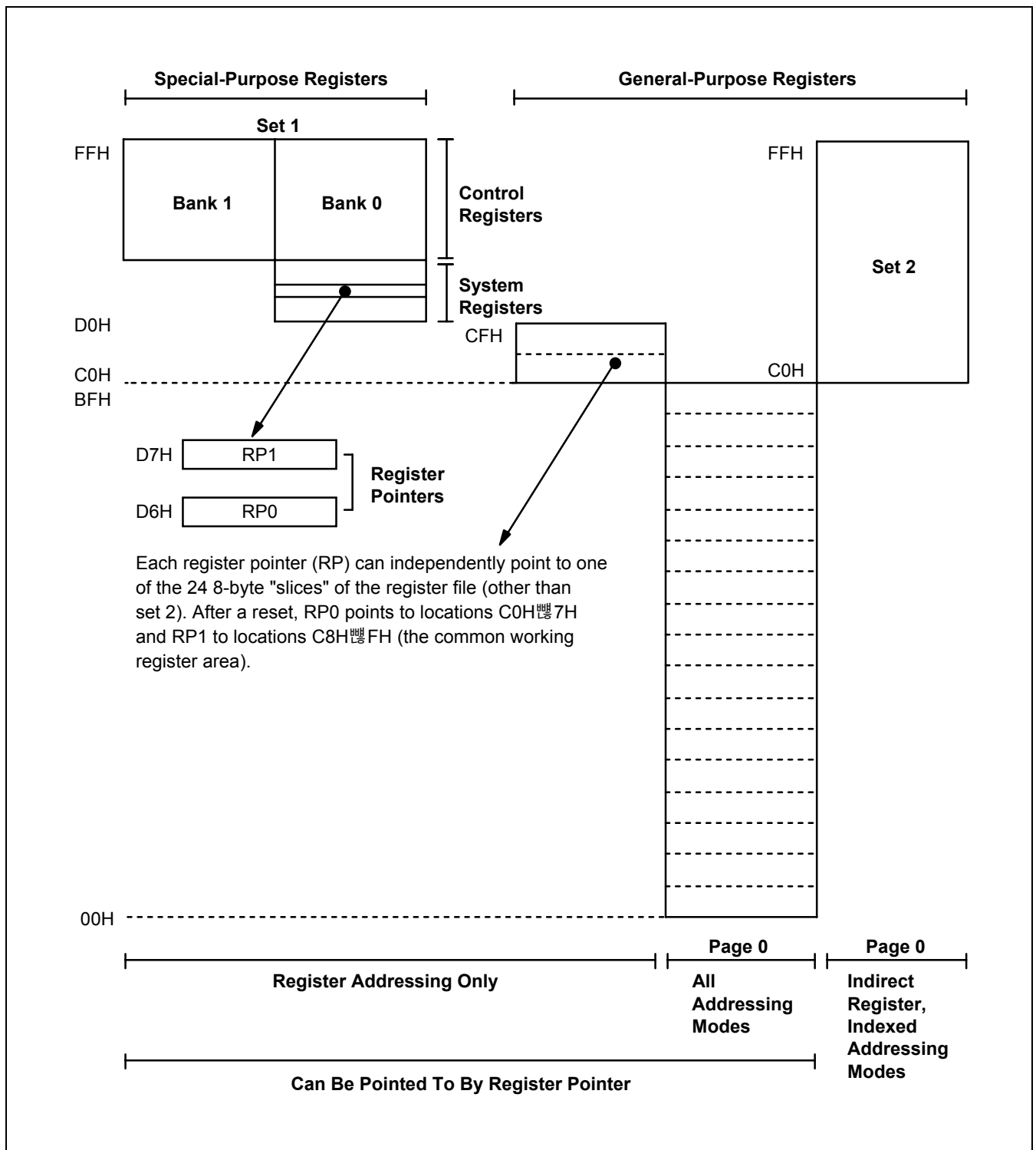


Figure 2-9. Register File Addressing

COMMON WORKING REGISTER AREA (C0H–CFH)

After a reset, register pointers RP0 and RP1 automatically select two 8-byte register slices in set 1, locations C0H–CFH, as the active 16-byte working register block:

RP0 → C0H–C7H

RP1 → C8H–CFH

This 16-byte address range is called *common working register area*. That is, locations in this area can be used as working registers by operations that address any location on any page in the register file. Typically, these working registers serve as temporary buffers for data operations between different pages. However, because KS88C0716 uses only page 0, you can use the common working register area for any internal data operation.

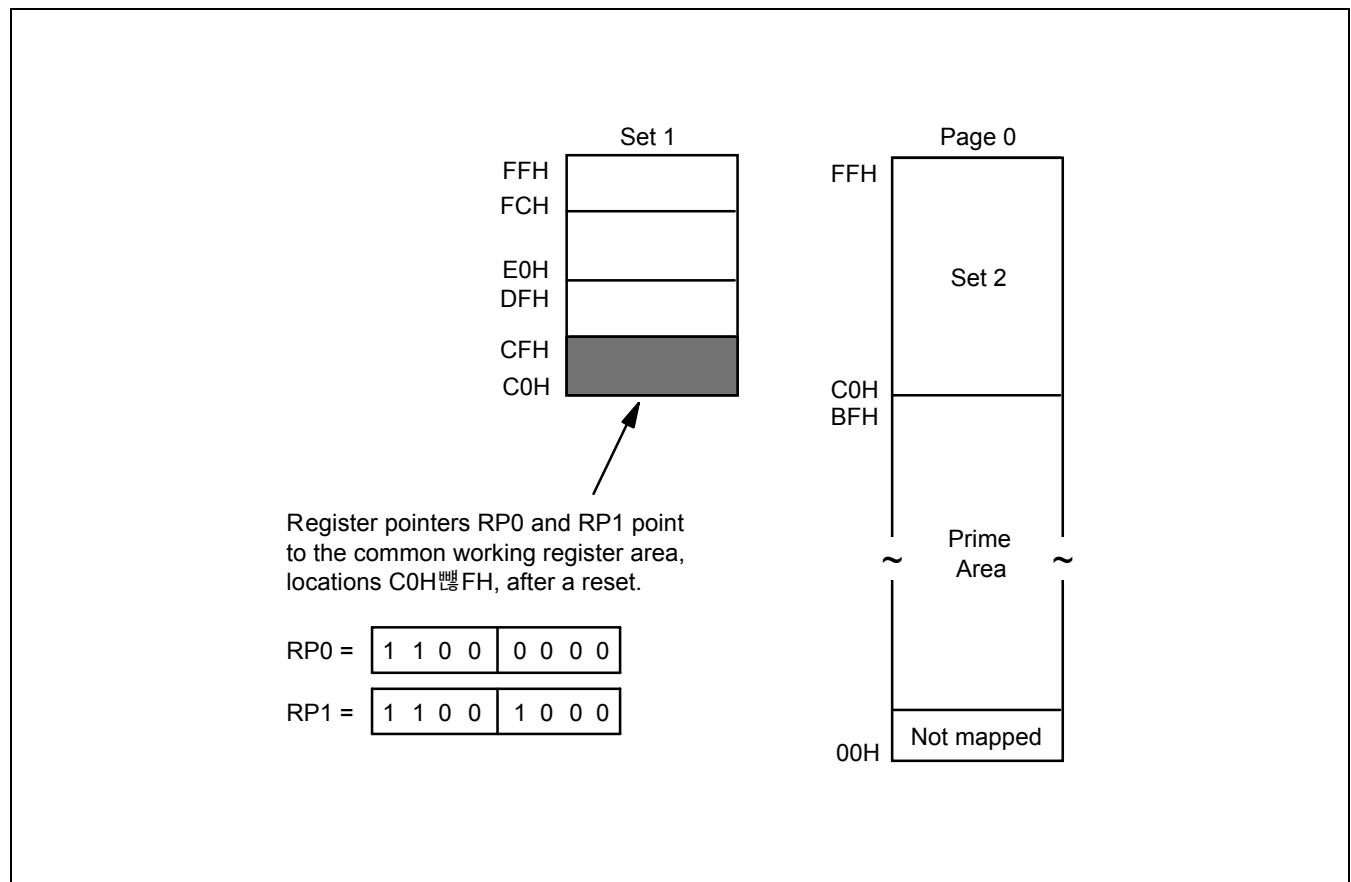


Figure 2-10. Common Working Register Area

PROGRAMMING TIP — Addressing the Common Working Register Area

As the following examples show, you should access working registers in the common area, locations C0H–CFH, using working register addressing mode only.

Examples

1. LD 0C2H,40H ; Invalid addressing mode!
 Use working register addressing instead:
 SRP #0C0H
 LD R2,40H ; R2 (C2H) ← the value in location 40H
2. ADD 0C3H,#45H ; Invalid addressing mode!
 Use working register addressing instead:
 SRP #0C0H
 ADD R3,#45H ; R3 (C3H) ← R3 + 45H

4-BIT WORKING REGISTER ADDRESSING

Each register pointer defines a movable 8-byte slice of working register space. The address information stored in a register pointer serves as an addressing "window" that makes it possible for instructions to access working registers very efficiently using short 4-bit addresses. When an instruction addresses a location in the selected working register area, the address bits are concatenated in the following way to form a complete 8-bit address:

- The high-order bit of the 4-bit address selects one of the register pointers ("0" selects RP0; "1" selects RP1);
- The five high-order bits in the register pointer select an 8-byte slice of the register space;
- The three low-order bits of the 4-bit address select one of the eight registers in the slice.

As shown in Figure 2-11, the result of this operation is that the five high-order bits from the register pointer are concatenated with the three low-order bits from the instruction address to form the complete address. As long as the address stored in the register pointer remains unchanged, the three bits from the address will always point to an address in the same 8-byte register slice.

Figure 2-12 shows a typical example of 4-bit working register addressing: The high-order bit of the instruction 'INC R6' is "0", which selects RP0. The five high-order bits stored in RP0 (01110B) are concatenated with the three low-order bits of the instruction's 4-bit address (110B) to produce the register address 76H (01110110B).

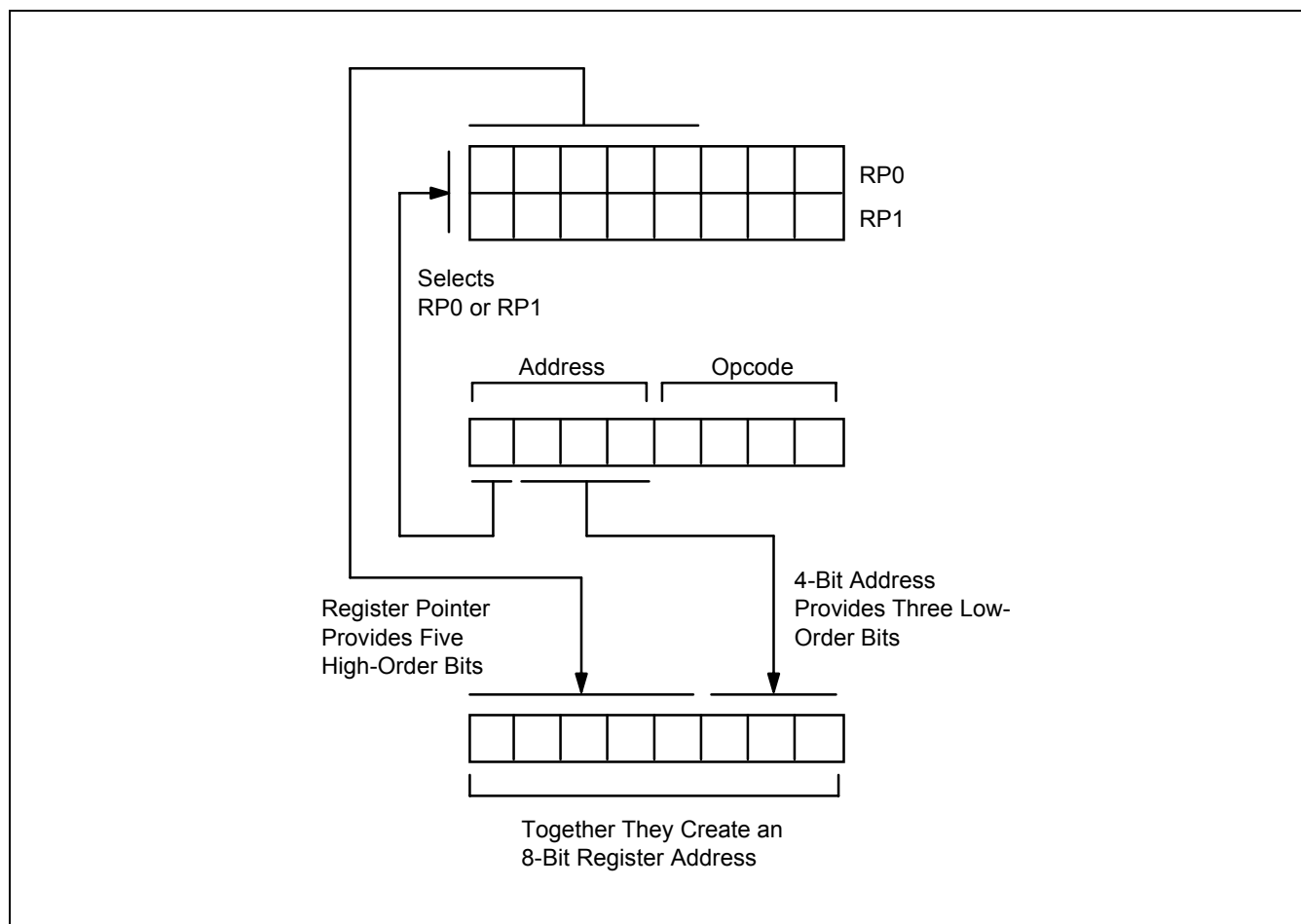


Figure 2-11. 4-Bit Working Register Addressing

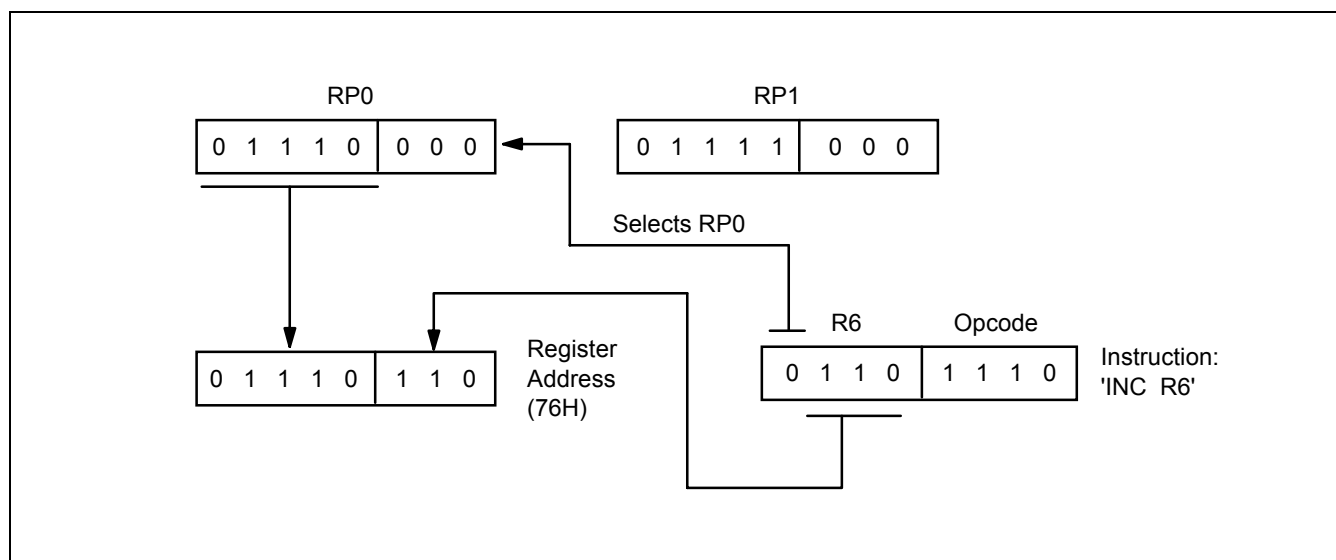


Figure 2-12. 4-Bit Working Register Addressing Example

8-BIT WORKING REGISTER ADDRESSING

You can also use 8-bit working register addressing to access registers in a selected working register area. To initiate 8-bit working register addressing, the upper four bits of the instruction address must contain the value 1100B. This 4-bit value (1100B) indicates that the remaining four bits have the same effect as 4-bit working register addressing.

As shown in Figure 2-13, the lower nibble of the 8-bit address is concatenated in much the same way as for 4-bit addressing: Bit 3 selects either RP0 or RP1, which then supplies the five high-order bits of the final address; the three low-order bits of the complete address are provided by the original instruction.

Figure 2-14 shows an example of 8-bit working register addressing: The four high-order bits of the instruction address (1100B) specify 8-bit working register addressing. Bit 4 ("1") selects RP1 and the five high-order bits in RP1 (10101B) become the five high-order bits of the register address. The three low-order bits of the register address (011) are provided by the three low-order bits of the 8-bit instruction address. The five address bits from RP1 and the three address bits from the instruction are concatenated to form the complete register address, 0ABH (10101011B).

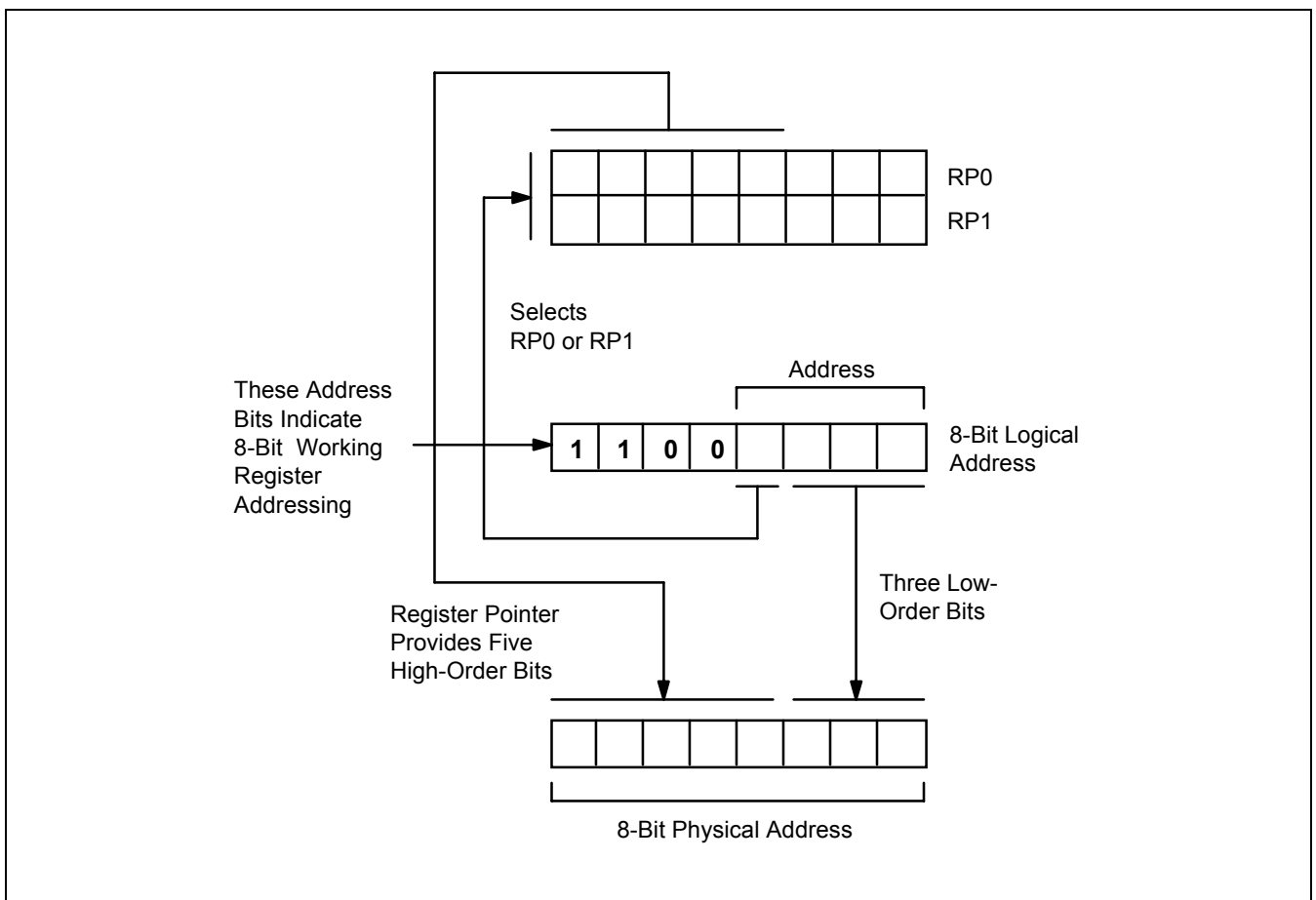
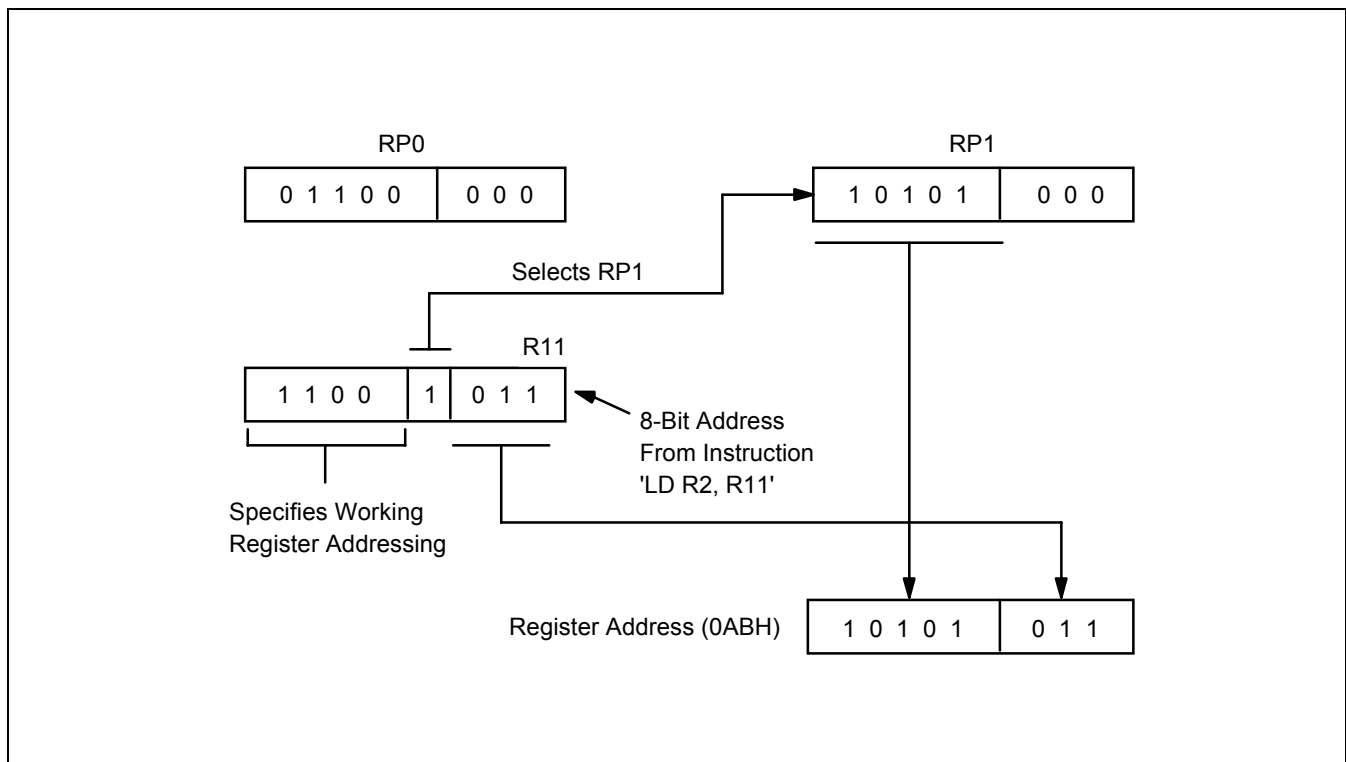


Figure 2-13. 8-Bit Working Register Addressing

**Figure 2-14. 8-Bit Working Register Addressing Example**

SYSTEM AND USER STACKS

KS88-series microcontrollers use the system stack for implementing subroutine calls and returns and for dynamic data storage. The PUSH and POP instructions support system stack operations.

Stack operations in the internal register file and in external data memory are supported by hardware. Bit 1 in the external memory timing register EMT selects an internal or external stack area.

The 16-bit stack pointer register (SPH, SPL) is used for external stack access. An 8-bit stack pointer (SPL) is sufficient for internal stack addressing.

Stack Operations

Return addresses for procedure calls, interrupts, and data are stored on the stack. The contents of the PC are saved to stack by a CALL instruction and restored by the RET instruction. When an interrupt occurs, the contents of the PC and the FLAGS register are pushed to the stack. The IRET instruction then pops these values back to their original locations. The stack address is always decremented *before* a push operation and incremented *after* a pop operation. The stack pointer (SP) always points to the stack frame stored on the top of the stack, as shown in Figure 2-15.

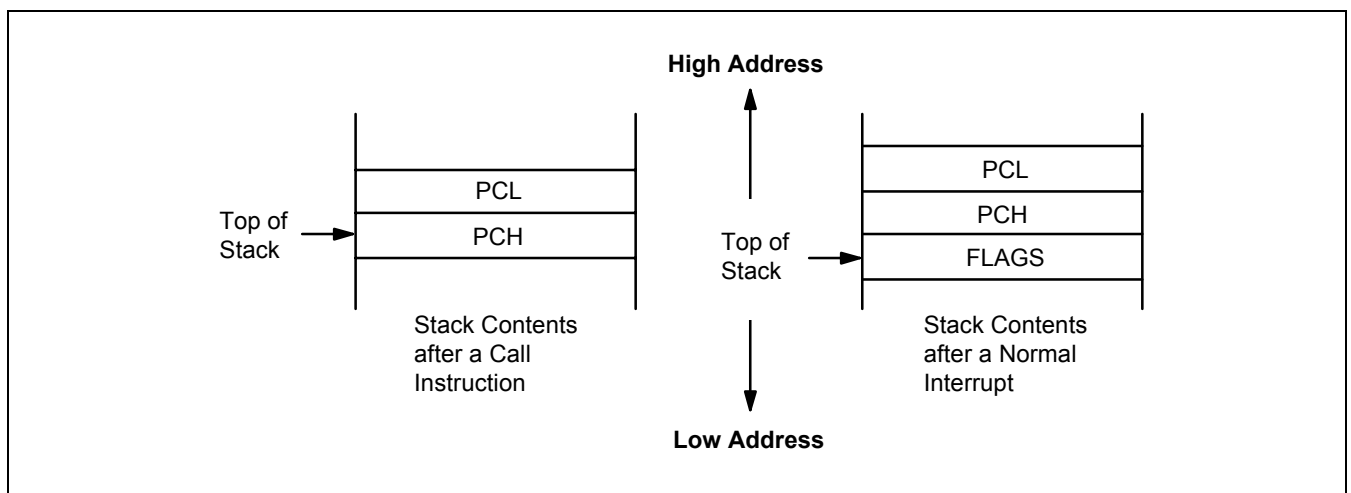


Figure 2-15. Stack Operations

User-Defined Stacks

You can freely define stacks in the internal register file as data storage locations. The instructions PUSHUI, PUSHUD, POPUI, and POPUD support user-defined stack operations.

Stack Pointers (SPL, SPH)

Register locations D8H and D9H contain the 16-bit stack pointer (SP) that is used for system stack operations. The most significant byte of the SP address, SP15–SP8, is stored in the SPH register (D8H); the least significant byte, SP7–SP0, is stored in the SPL register (D9H). After a reset, the SP value is undetermined.

Because only internal memory space is implemented in KS88C0716/P0716, the SPL must be initialized to an 8-bit value in the range 00H–FFH; the SPH register is not needed but can be used as a general-purpose register, if necessary.

When the SPL register contains the stack pointer value only (that is, when it points to a system stack in the register file), you can use the SPH register as a general-purpose data register. However, if an overflow or underflow condition occurs as the result of incrementing or decrementing the stack address in the SPL register during normal stack operations, the value in the SPL register will overflow (or underflow) to the SPH register, overwriting any other data that is currently stored there. To avoid overwriting data in the SPH register, you can initialize the SPL value to 'FFH' instead of '00H'.

PROGRAMMING TIP — Standard Stack Operations Using PUSH and POP

The following example shows you how to perform stack operations in the internal register file using PUSH and POP instructions:

```

LD      SPL,#0FFH      ; SPL ← FFH
                        ; (Normally, the SPL is set to 0FFH by the initialization
                        ; routine)
.
.
.
PUSH    PP              ; Stack address 0FEH ← PP
PUSH    RP0             ; Stack address 0FDH ← RP0
PUSH    RP1             ; Stack address 0FCH ← RP1
PUSH    R3              ; Stack address 0FBH ← R3
.
.
.
POP     R3              ; R3 ← Stack address 0FBH
POP     RP1             ; RP1 ← Stack address 0FCH
POP     RP0             ; RP0 ← Stack address 0FDH
POP     PP              ; PP ← Stack address 0FEH

```

3

ADDRESSING MODES

OVERVIEW

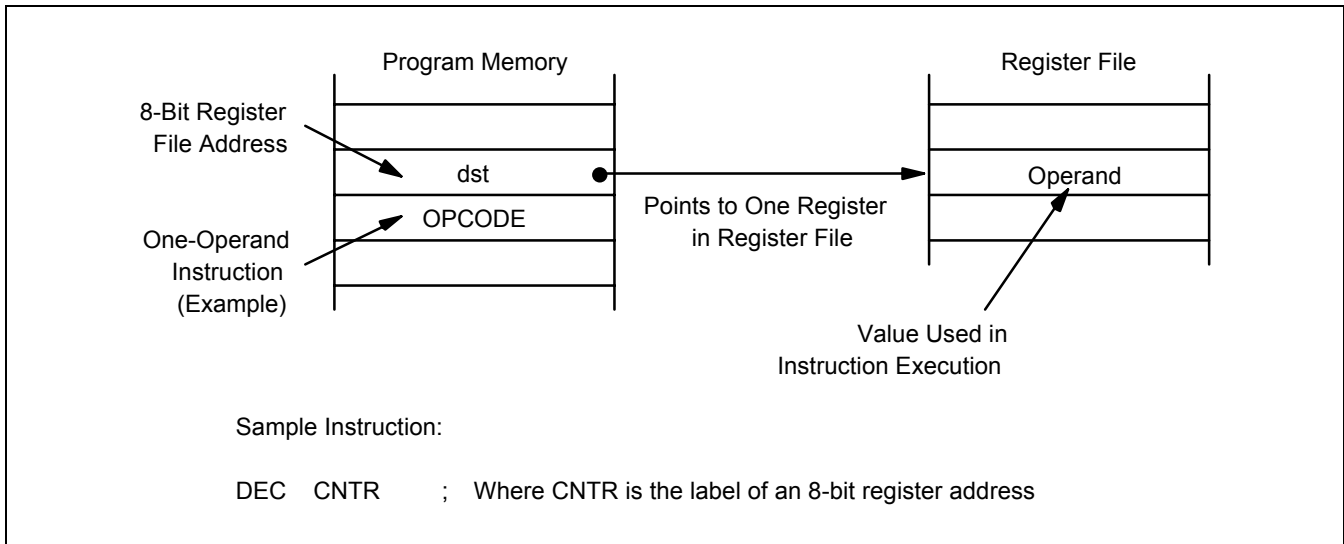
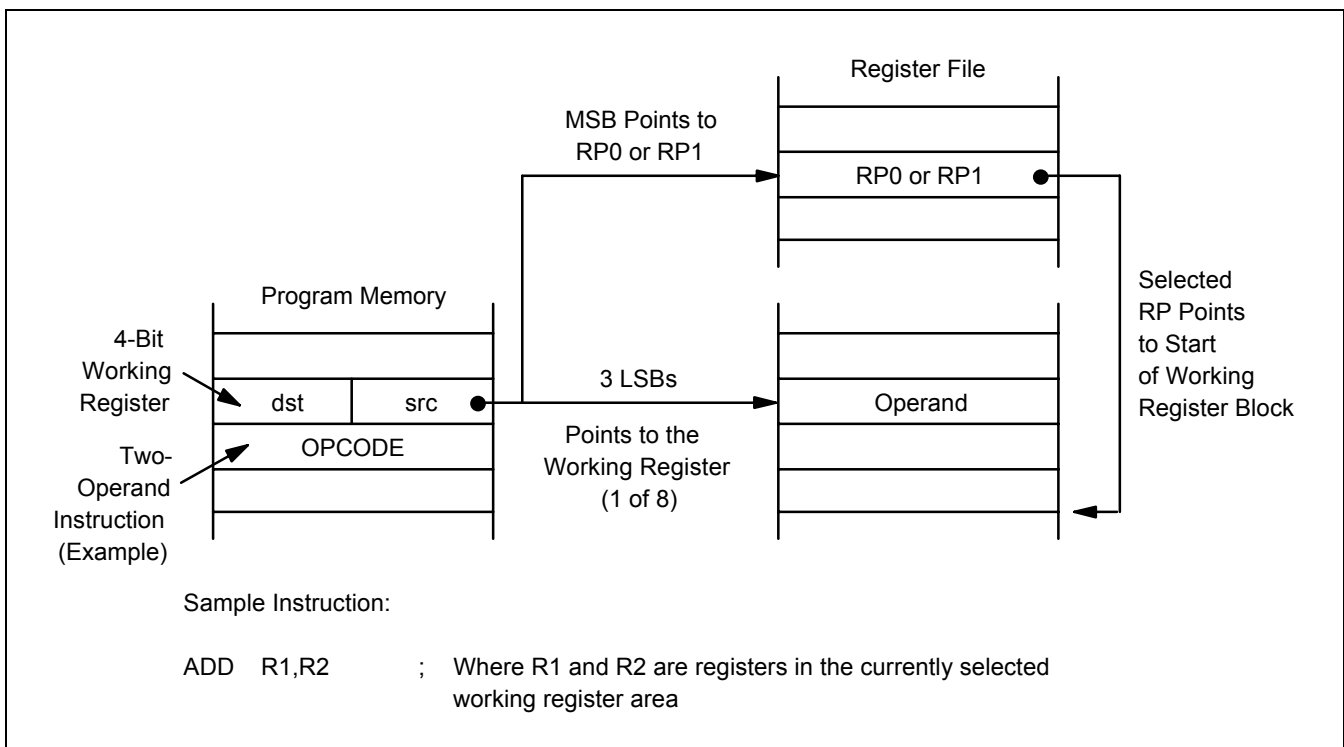
The program counter is used to fetch instructions that are stored in program memory for execution. Instructions indicate the operation to be performed and the data to be operated on. *Addressing mode* is the method used to determine the location of the data operand. The operands specified in SAM8 instructions may be condition codes, immediate data, or a location in the register file, program memory, or data memory.

The SAM8 instruction set supports seven explicit addressing modes. Not all of these addressing modes are available for each instruction:

- Register (R)
- Indirect Register (IR)
- Indexed (X)
- Direct Address (DA)
- Indirect Address (IA)
- Relative Address (RA)
- Immediate (IM)

REGISTER ADDRESSING MODE (R)

In Register addressing mode, the operand is the content of a specified register or register pair (see Figure 3-1). Working register addressing differs from Register addressing because it uses a register pointer to specify an 8-byte working register space in the register file and an 8-bit register within that space (see Figure 3-2).

**Figure 3-1. Register Addressing****Figure 3-2. Working Register Addressing**

INDIRECT REGISTER ADDRESSING MODE (IR)

In Indirect Register (IR) addressing mode, the content of the specified register or register pair is the address of the operand. Depending on the instruction used, the actual address may point to a register in the register file, to program memory (ROM), or to an external memory space, if implemented (see Figures 3-3 through 3-6).

You can use any 8-bit register to indirectly address another register. Any 16-bit register pair can be used to indirectly address another memory location. Remember, however, that locations C0H–FFH in set 1 cannot be accessed using Indirect Register addressing mode.

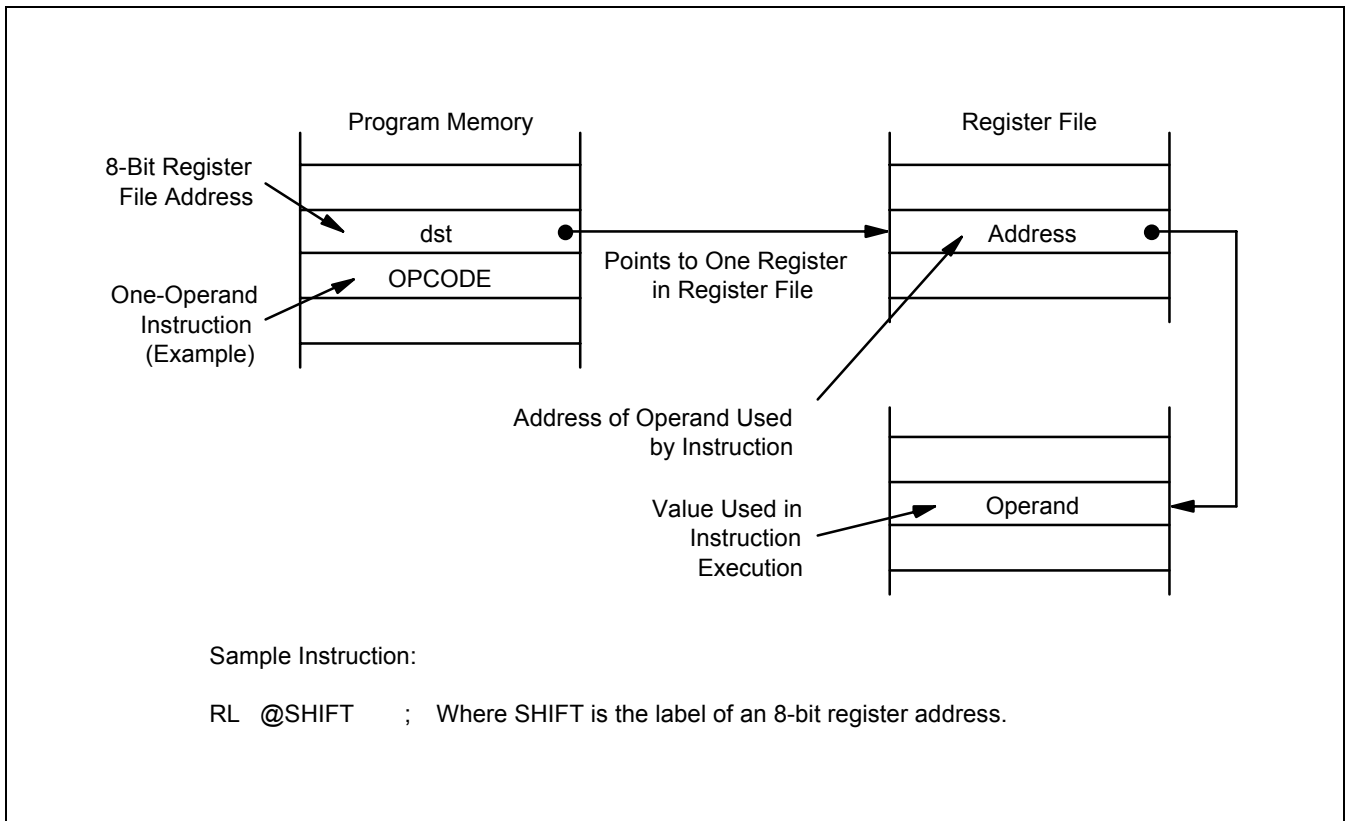


Figure 3-3. Indirect Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Continued)

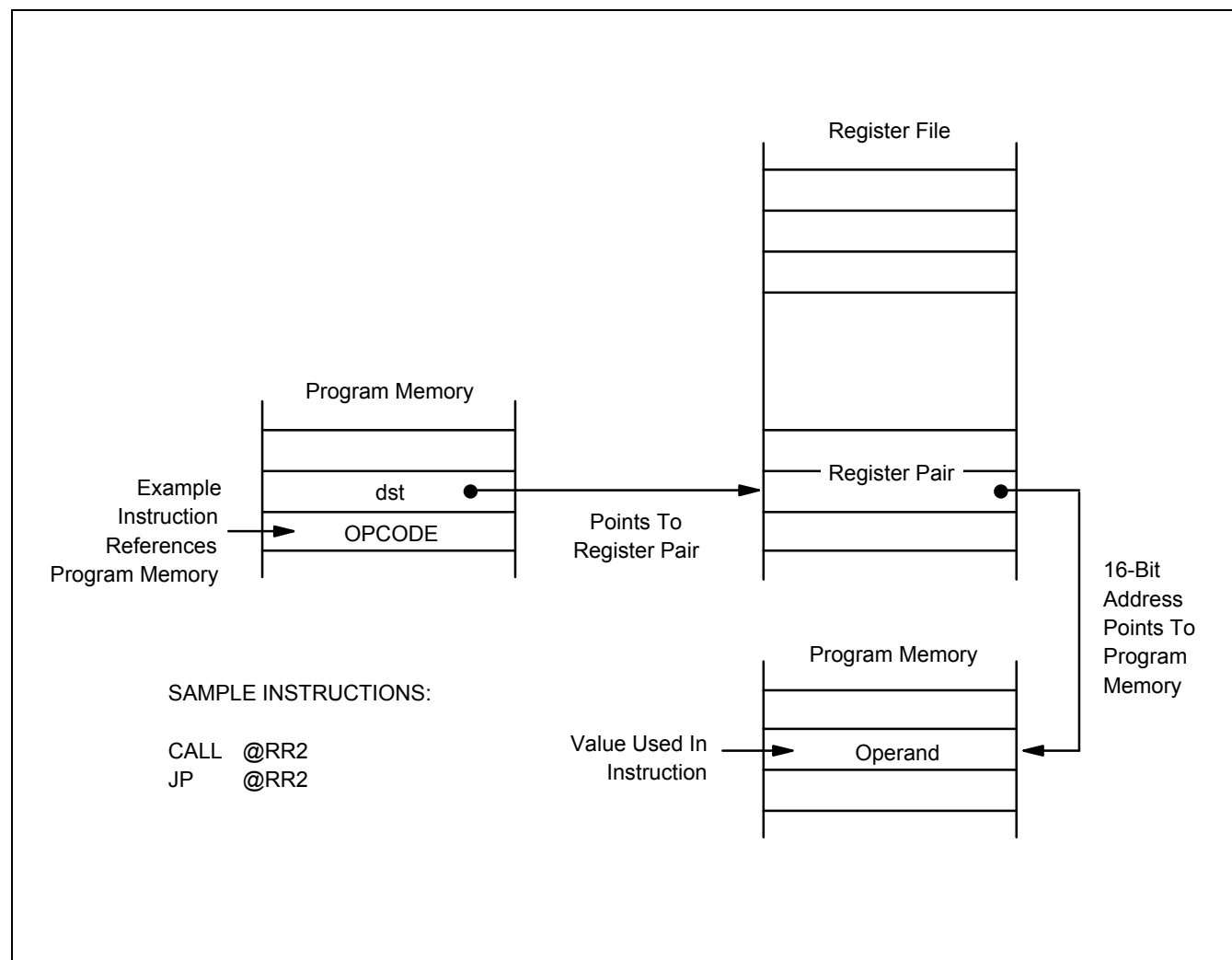


Figure 3-4. Indirect Register Addressing to Program Memory

INDIRECT REGISTER ADDRESSING MODE (Continued)

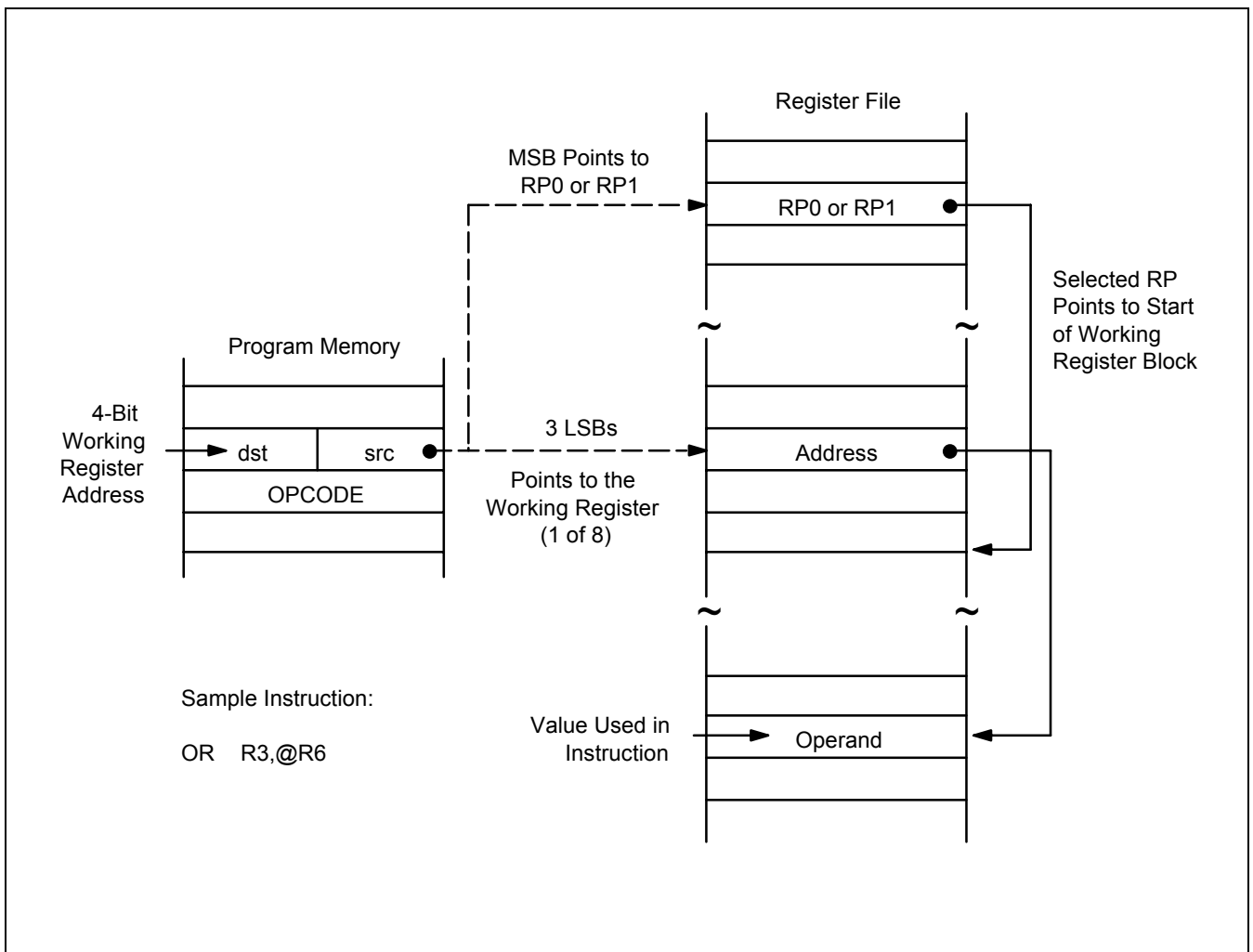


Figure 3-5. Indirect Working Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Concluded)

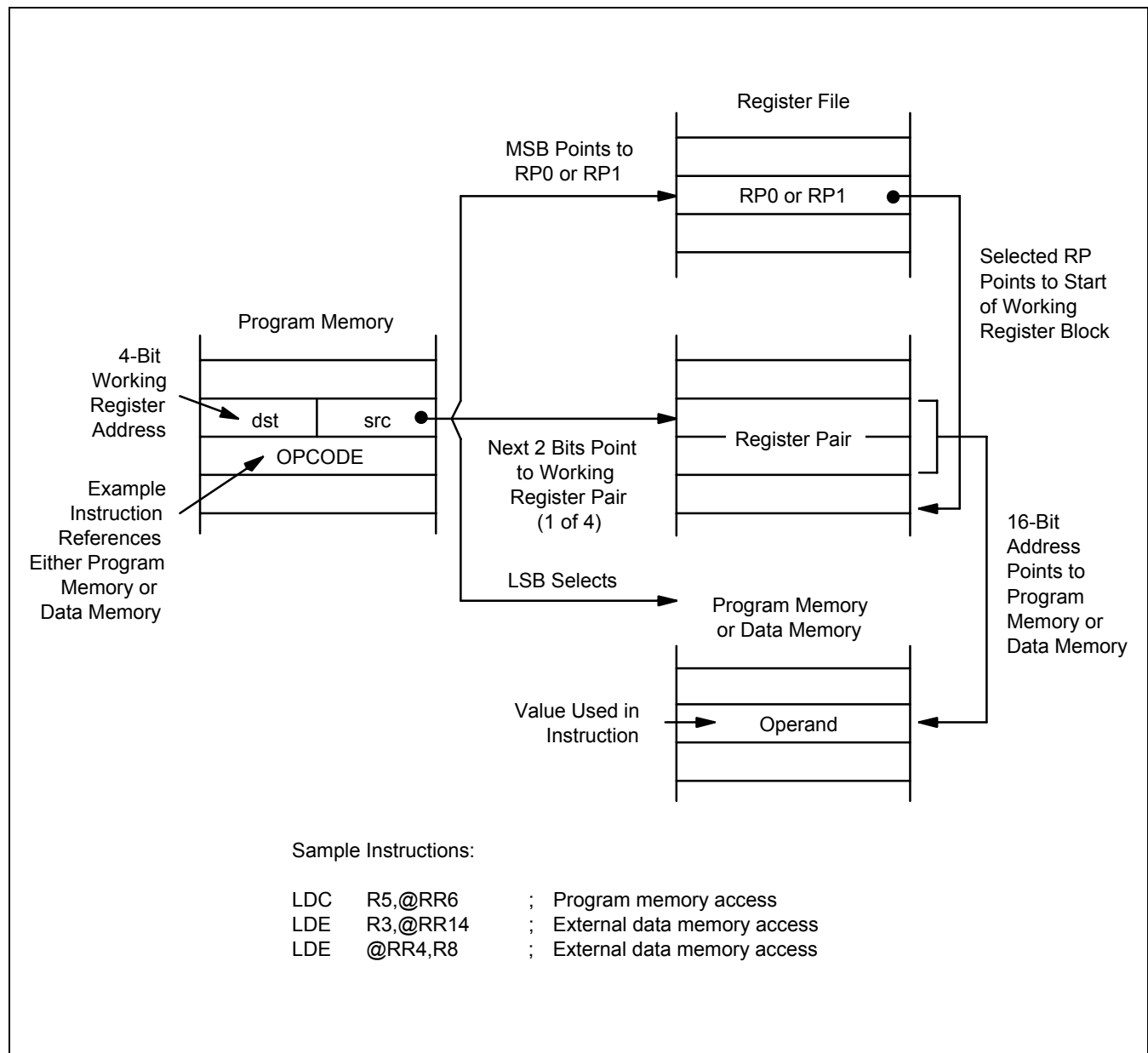


Figure 3-6. Indirect Working Register Addressing to Program or Data Memory

INDEXED ADDRESSING MODE (X)

Indexed (X) addressing mode adds an offset value to a base address during an instruction execution in order to calculate the effective operand address (see Figure 3-7). You can use Indexed addressing mode to access locations in the internal register file or in external memory (if implemented). You cannot, however, access locations C0H–FFH in set 1 using Indexed addressing.

In short offset Indexed addressing mode, the 8-bit displacement is treated as a signed integer in the range –128 to +127. This applies to external memory accesses only (see Figure 3-8).

For register file addressing, an 8-bit base address provided by the instruction is added to an 8-bit offset contained in a working register. For external memory accesses, the base address is stored in the working register pair designated in the instruction. The 8-bit or 16-bit offset given in the instruction is then added to the base address (see Figure 3-9).

The only instruction that supports Indexed addressing mode for the internal register file is the Load instruction (LD). The LDC and LDE instructions support Indexed addressing mode for internal program memory and for external data memory (if implemented).

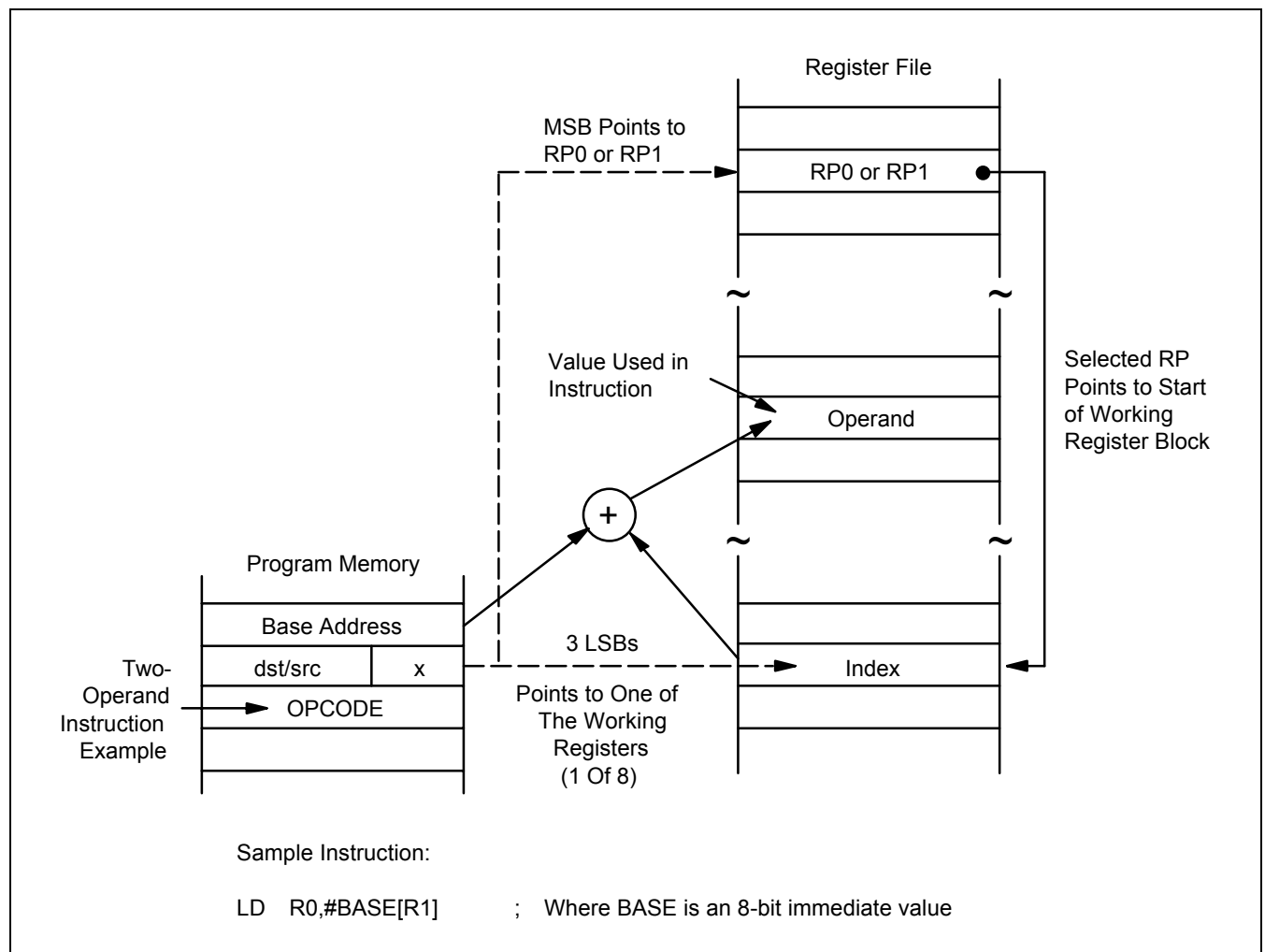


Figure 3-7. Indexed Addressing to Register File

INDEXED ADDRESSING MODE (Continued)

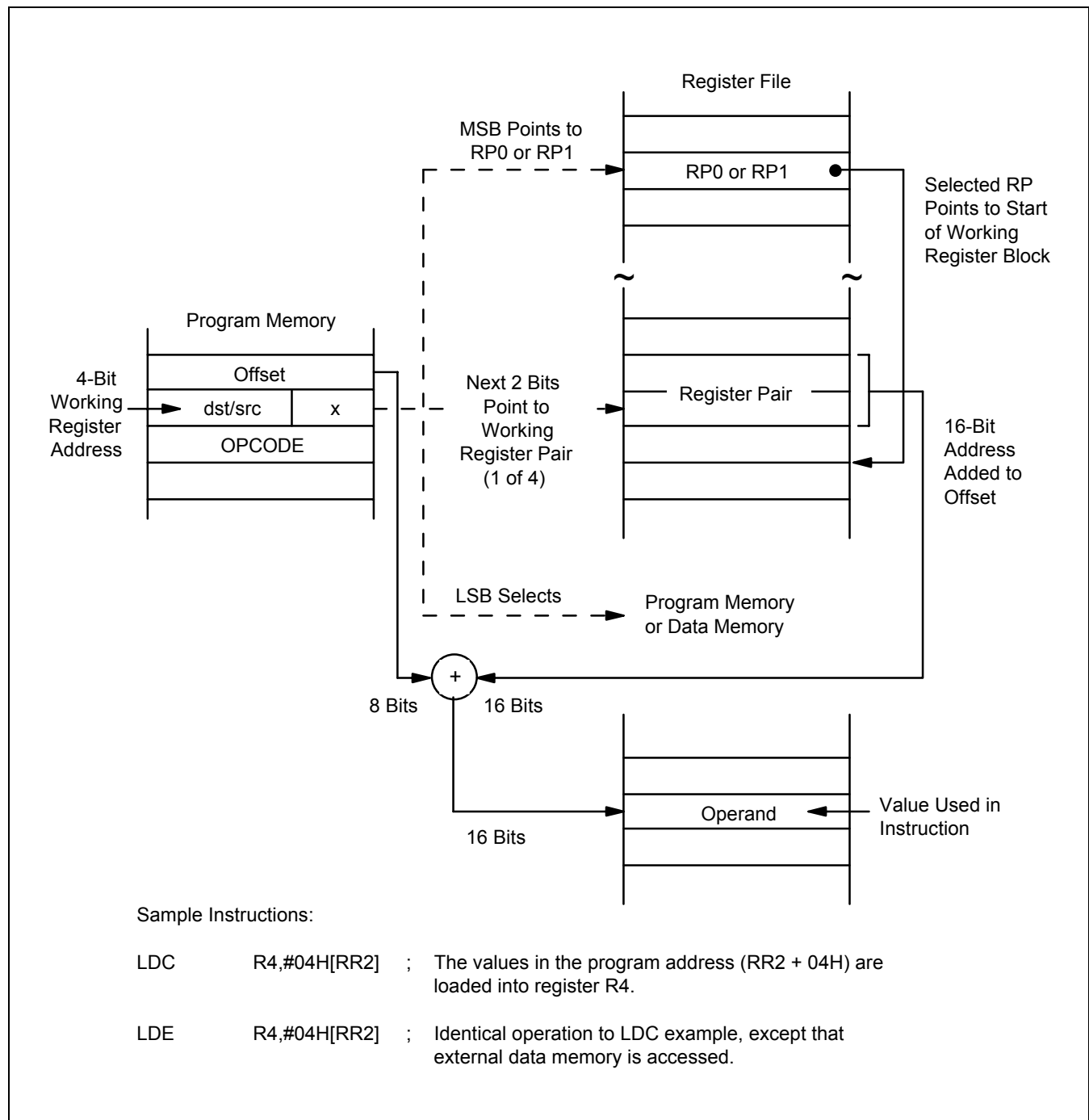


Figure 3-8. Indexed Addressing to Program or Data Memory with Short Offset

INDEXED ADDRESSING MODE (Concluded)

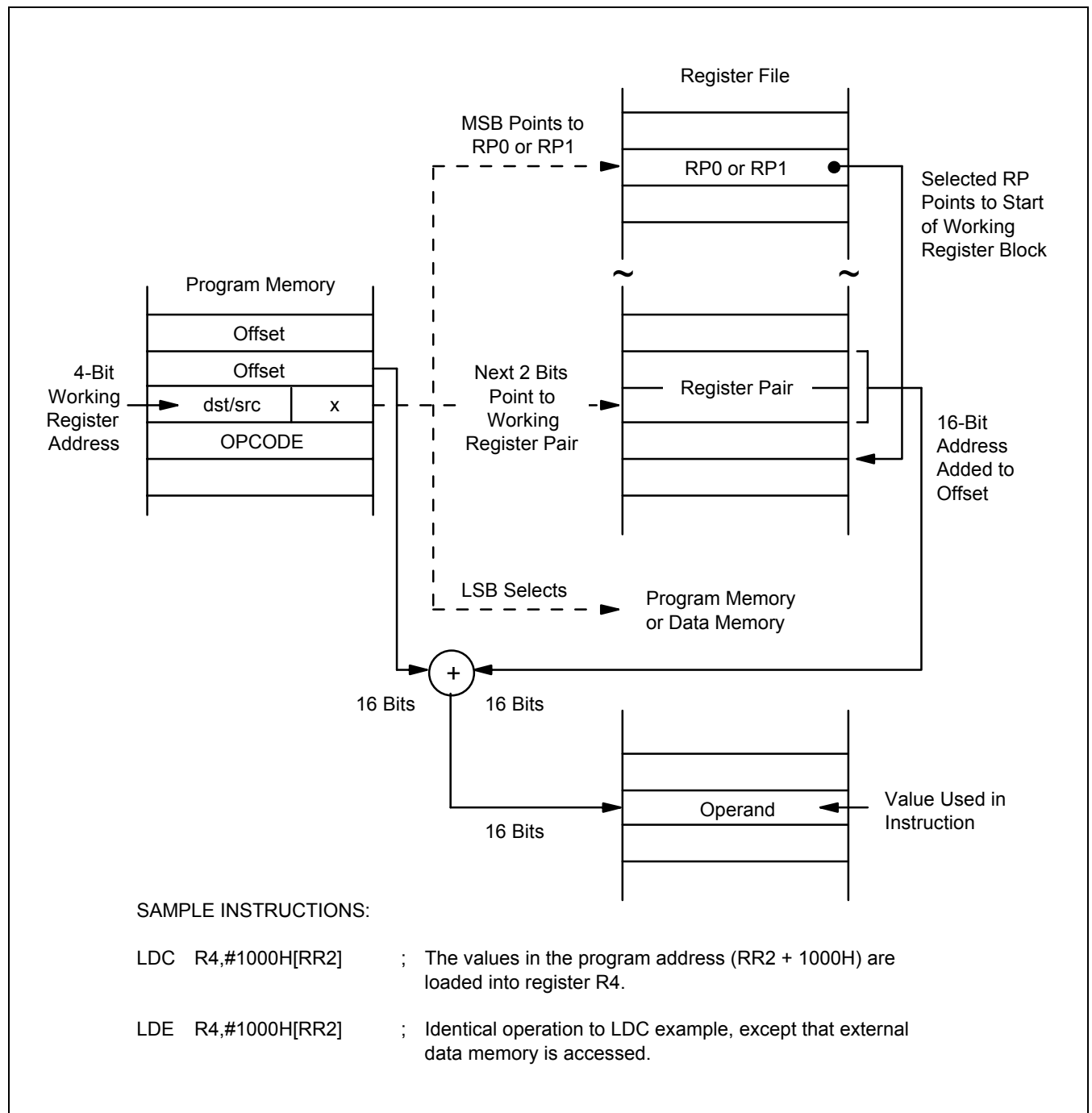
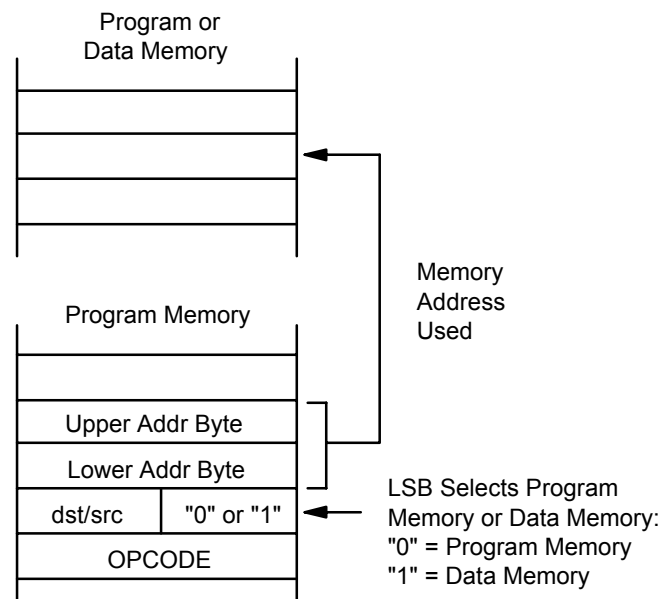


Figure 3-9. Indexed Addressing to Program or Data Memory

DIRECT ADDRESS MODE (DA)

In Direct Address (DA) mode, the instruction provides the operand's 16-bit memory address. Jump (JP) and Call (CALL) instructions use this addressing mode to specify the 16-bit destination address that is loaded into the PC whenever a JP or CALL instruction is executed.

The LDC and LDE instructions can use Direct Address mode to specify the source or destination address for Load operations to program memory (LDC) or to external data memory (LDE), if implemented.

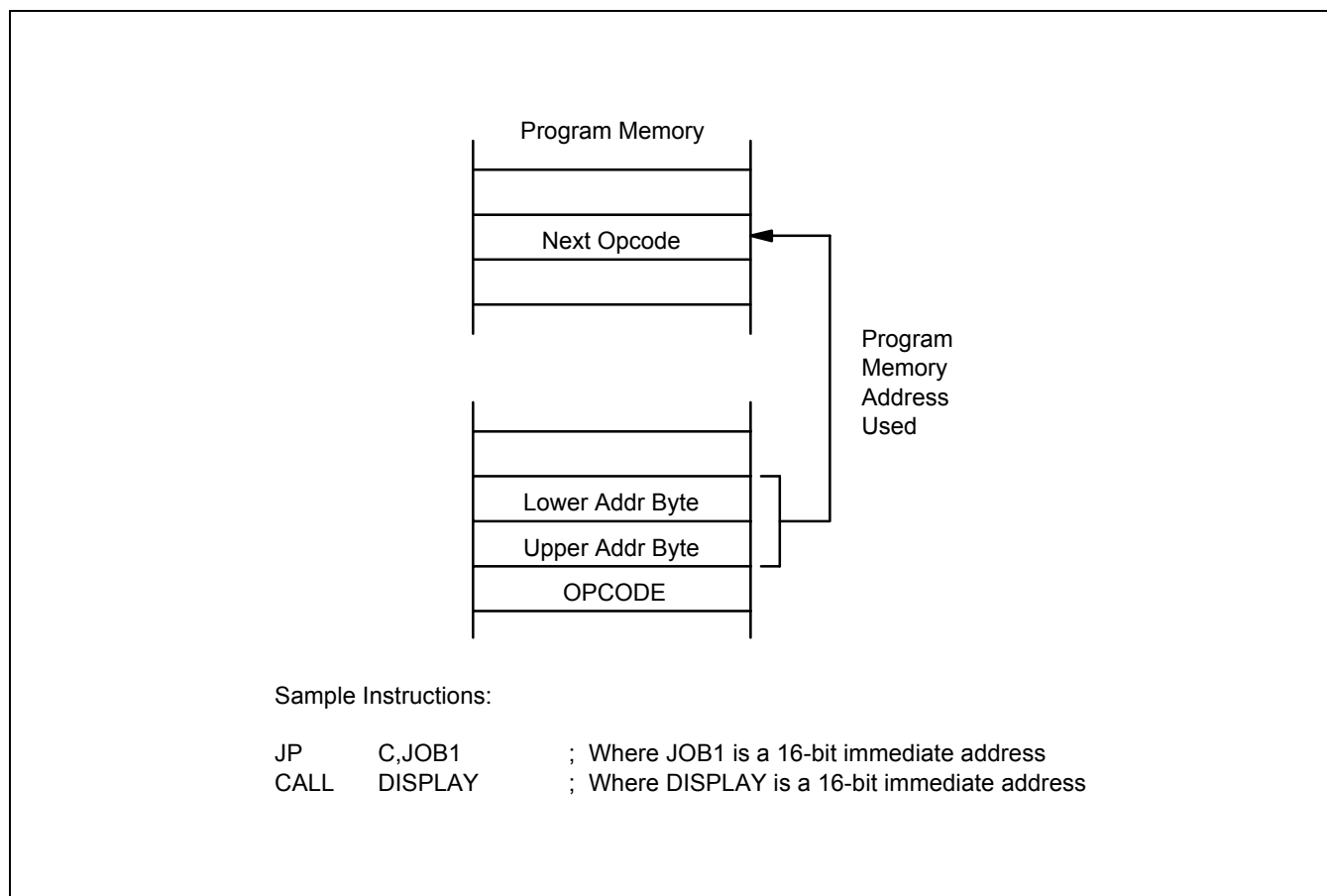


Sample Instructions:

LDC R5,1234H ; The values in the program address (1234H) are loaded into register R5.

LDE R5,1234H ; Identical operation to LDC example, except that external data memory is accessed.

Figure 3-10. Direct Addressing for Load Instructions

DIRECT ADDRESS MODE (Continued)**Figure 3-11. Direct Addressing for Call and Jump Instructions**

INDIRECT ADDRESS MODE (IA)

In Indirect Address (IA) mode, the instruction specifies an address located in the lowest 256 bytes of the program memory. The selected pair of memory locations contains the actual address of the next instruction to be executed. Only the CALL instruction can use the Indirect Address mode.

Because the Indirect Address mode assumes that the operand is located in the lowest 256 bytes of program memory, only an 8-bit address is supplied in the instruction; the upper bytes of the destination address are assumed to be all zeros.

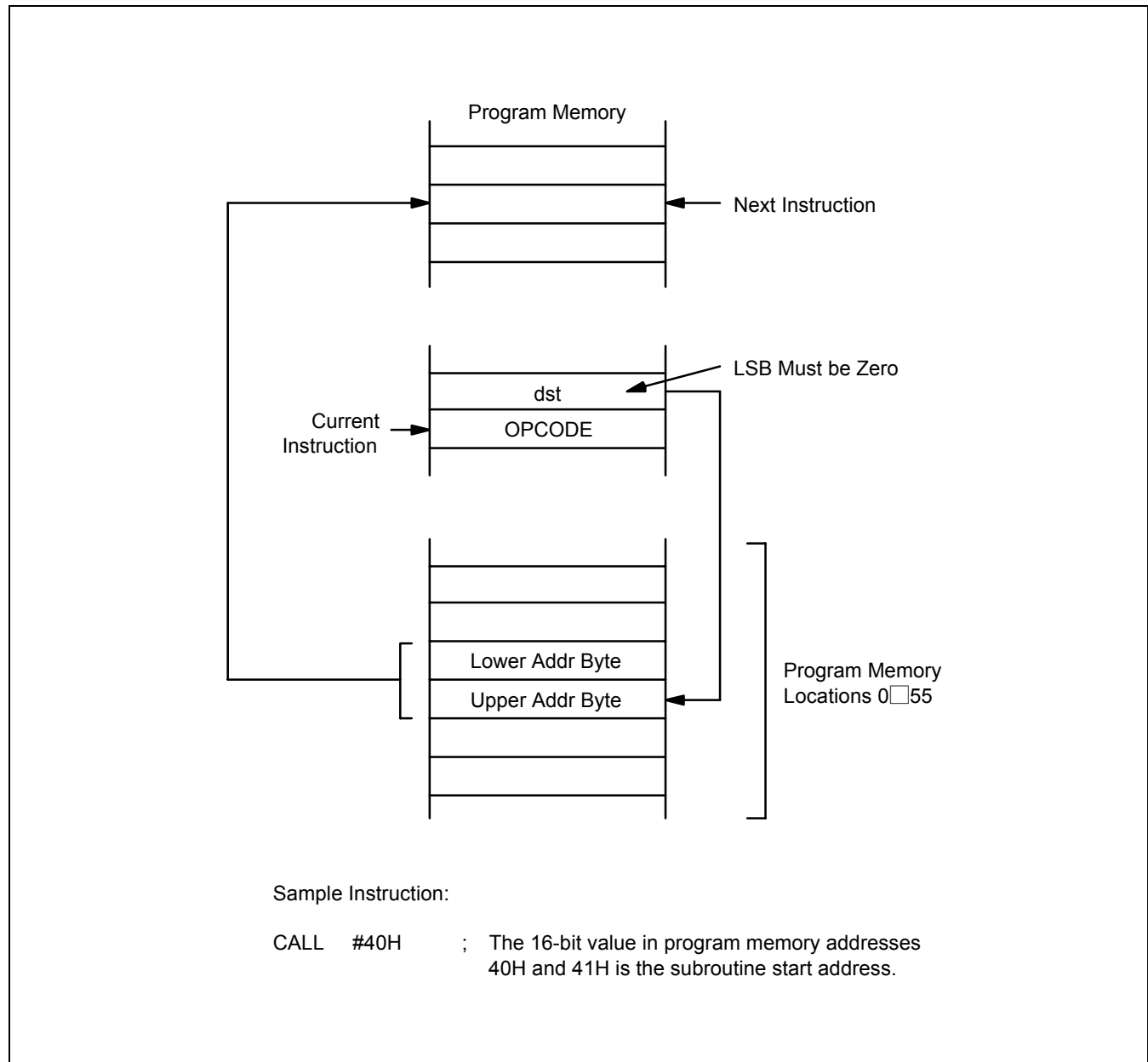


Figure 3-12. Indirect Addressing

RELATIVE ADDRESS MODE (RA)

In Relative Address (RA) mode, a two's-complement signed displacement between -128 and $+127$ is specified in the instruction. The displacement value is then added to the current PC value. The result is the address of the next instruction to be executed. Before this addition occurs, the PC contains the address of the instruction immediately following the current instruction.

Several program control instructions use the Relative Address mode to perform conditional jumps. The instructions that support RA addressing are BTJRF, BTJRT, DJNZ, CPIJE, CPIJNE, and JR.

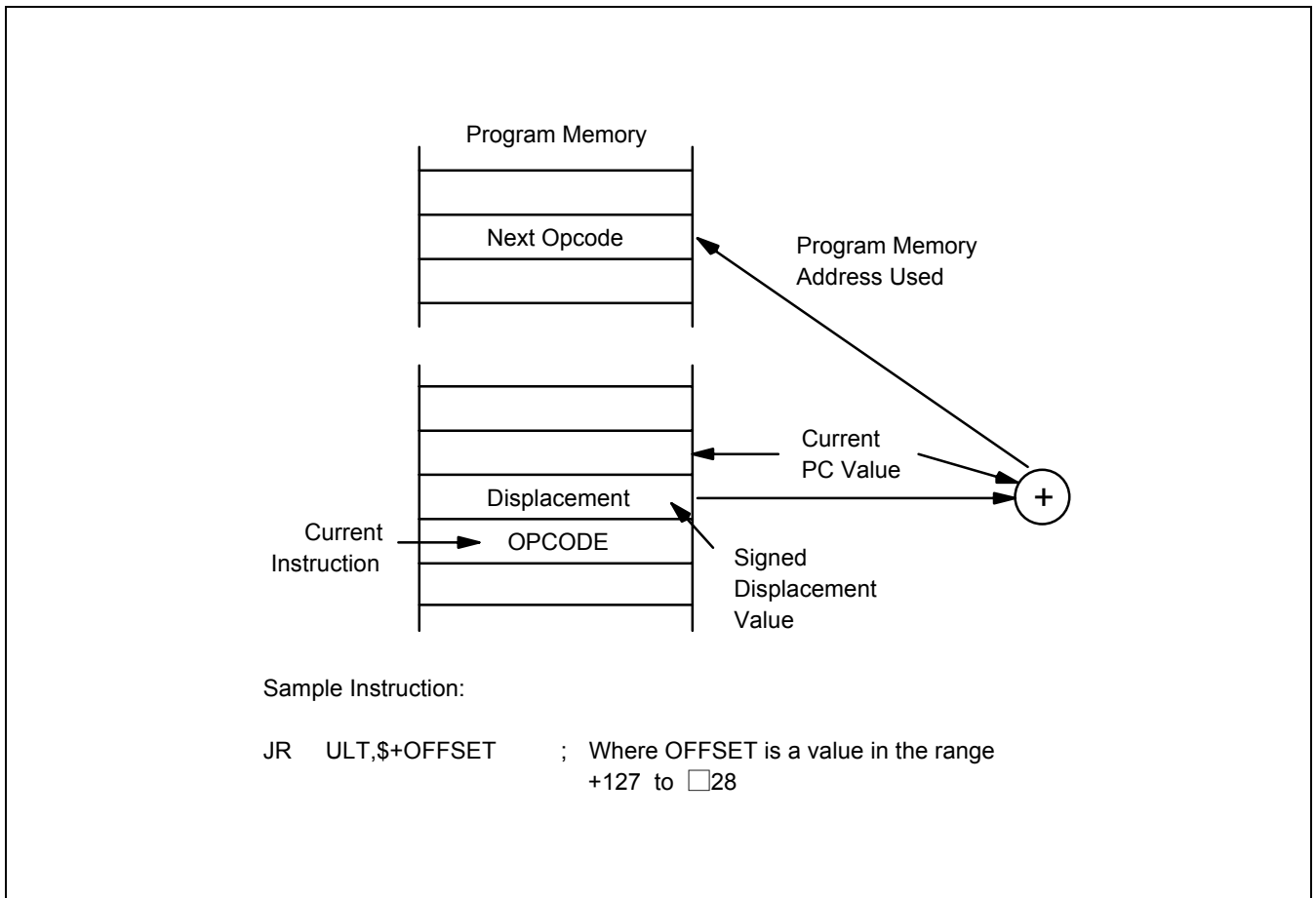


Figure 3-13. Relative Addressing

IMMEDIATE MODE (IM)

In Immediate (IM) mode, the operand value used in the instruction is the value supplied in the operand field itself. The operand may be one byte or one word in length, depending on the instruction used. Immediate addressing mode is useful for loading constant values into registers.

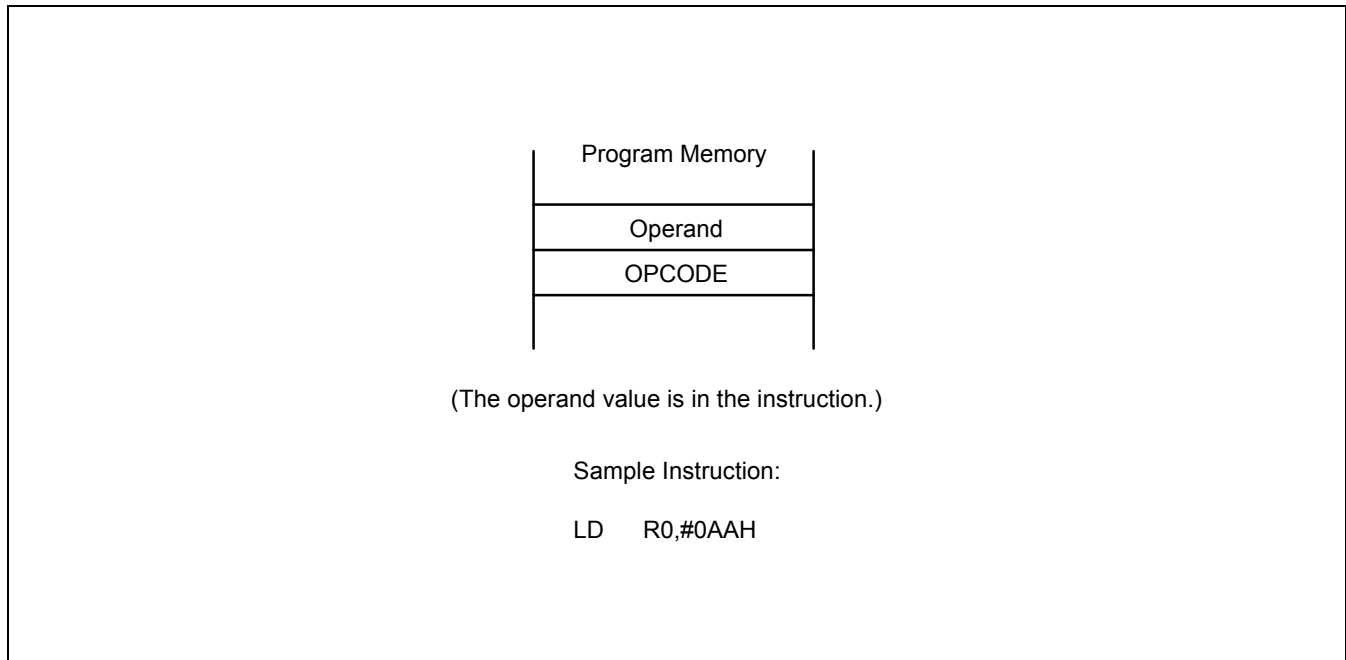


Figure 3-14. Immediate Addressing

4

CONTROL REGISTERS

OVERVIEW

In this section, detailed descriptions of the KS88C0716/P0716 control registers are presented in an easy-to-read format. You can use this section as a quick-reference source when writing application programs.

The locations and read/write characteristics of all mapped registers in the KS88C0716/P0716 register files are presented in Tables 4-1, 4-2, and 4-3. The hardware reset values for these registers are described in Section 8, “RESET and Power-Down.”

Figure 4-1 illustrates the important features of the standard register description format.

Control register descriptions are arranged in alphabetical order according to register mnemonic. More detailed information about control registers is presented in the context of the specific peripheral hardware descriptions in Part II of this manual.

Table 4-1. Set 1 Registers

Register Name	Mnemonic	Decimal	Hex	R/W
Timer C Counter Register(High Byte)	TCH	208	D0H	R/W
Timer C Counter Register(Low Byte)	TCL	209	D1H	R/W
Location D2H is not mapped.				
Basic Timer Control Register (watch dog)	BTCON	211	D3H	R/W
Clock Control Register	CLKCON	212	D4H	R/W
System Flags Register	FLAGS	213	D5H	R/W
Register Pointer 0	RP0	214	D6H	R/W
Register Pointer 1	RP1	215	D7H	R/W
Stack Pointer (High Byte)	SPH	216	D8H	R/W
Stack Pointer (Low Byte)	SPL	217	D9H	R/W
Instruction Pointer (High Byte)	IPH	218	DAH	R/W
Instruction Pointer (Low Byte)	IPL	219	DBH	R/W
Interrupt Request Register	IRQ	220	DCH	R (note)
Interrupt Mask Register	IMR	221	DDH	R/W
System Mode Register	SYM	222	DEH	R/W
Register Page Pointer	PP	223	DFH	R/W

NOTE: You cannot use a read-only register (IRQ) as a destination field for the instructions OR, AND, LD, or LDB.

Table 4-2. Set 1, Bank 0 Registers

Register Name	Mnemonic	Decimal	Hex	R/W
Port 0 Data Register	P0	224	E0H	R/W
Port 1 Data Register	P1	225	E1H	R/W
Port 2 Data Register	P2	226	E2H	R/W
Port 3 Data Register	P3	227	E3H	R/W
Port 4 Data Register	P4	228	E4H	R/W
Port 5 Data Register	P5	229	E5H	R/W
Port 6 Data Register	P6	230	E6H	R/W
Port4 Interrupt Pending Register	P4PND	231	E7H	R/W
Port 4 Interrupt Enable Register	P4INT	232	E8H	R/W
Serial Port Shift Register	SIO	233	E9H	R/W
Serial Port Control Register	SIOCON	234	EAH	R/W
Serial Port Interrupt Pending Register	SIOPND	235	EBH	R/W

Table 4-2. Set 1, Bank 0 Registers (Continued)

Register Name	Mnemonic	Decimal	Hex	R/W
Timer A Data Register	TADATA	236	ECH	R/W
Timer B Data Register	TBDATA	237	EDH	R/W
Timer Module 0 Control Register	T0CON	238	EEH	R/W
Timer B Interrupt Register	TBINT	239	EFH	W
Port 0 Control Register	P0CON	240	F0H	R/W
Port 1 Control Register	P1CON	241	F1H	R/W
Port 2 Control Register (High Byte)	P2CONH	242	F2H	R/W
Port 2 Control Register (Low Byte)	P2CONL	243	F3H	R/W
Port 3 Control Register (High Byte)	P3CONH	244	F4H	R/W
Port 3 Control Register (Low Byte)	P3CONL	245	F5H	R/W
Port 4 Control Register (High Byte)	P4CONH	246	F6H	R/W
Port 4 Control Register (Low Byte)	P4CONL	247	F7H	R/W
Timer D Counter Register(High Byte)	TDH	248	F8H	R/W
Timer D Counter Register(Low Byte)	TDL	249	F9H	R/W
Timer Module 1 Control Register	T1CON	250	FAH	R/W
Timer Module 1 Mode Register	T1MOD	251	FBH	R/W
Location FCH is reserved for factory test.				
Basic Timer counter	BTCNT	253	FDH	R (note)
External Memory Timing Register	EMT	254	FEH	R/W
Interrupt Priority Register	IPR	255	FFH	R/W

NOTE: You cannot use a read-only register (BTCNT) as a destination field for the instructions OR, AND, LD, or LDB.

Table 4-3. Set 1, Bank 1 Registers

Register Name	Mnemonic	Decimal	Hex	R/W
Port 5 Control Register	P5CON	248	F8H	R/W
Port 6 Control Register	P6CON	249	F9H	R/W

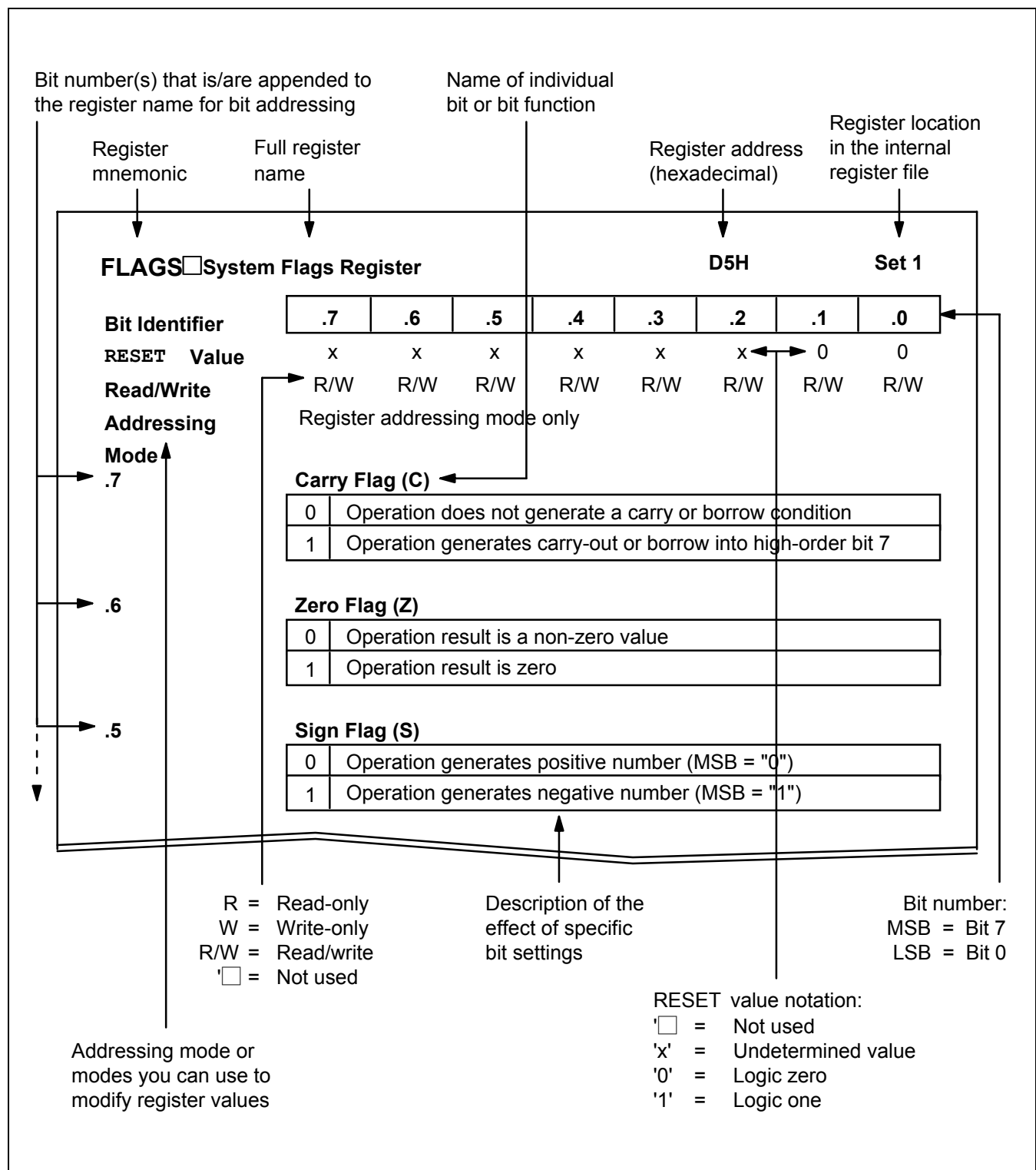


Figure 4-1. Register Description Format

BTCON — Basic Timer Control Register**D3H****Set 1**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 – .4**Watchdog Timer Function Disable Code (for Reset)**

1	0	1	0	Disable watchdog timer function
Any other value				Enable watchdog timer function

.3 and .2**Basic Timer Input Clock Selection Bits**

0	0	f _{OSC} /4096
0	1	f _{OSC} /1024
1	0	f _{OSC} /128
1	1	Invalid setting; not used for KS88C0716/P0716

.1**Basic Timer Counter Clear Bit ⁽¹⁾**

0	No effect
1	Clear the basic timer counter value

.0**Clock Frequency Divider Clear Bit for Basic Timer and Timer 0 ⁽²⁾**

0	No effect
1	Clear basic timer and timer 0 frequency dividers

NOTES:

- When you write a "1" to BTCON.1, the basic timer counter value is cleared to '00H'. Immediately following the write operation, the BTCON.1 value is automatically cleared to "0".
- When you write a "1" to BTCON.0, the corresponding frequency divider is cleared to '00H'. Immediately following the write operation, the BTCON.0 value is automatically cleared to "0".

CLKCON — System Clock Control Register**D4H****Set 1**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7	Oscillator IRQ Wake-up Function Enable Bit							
0	Enable IRQ for main system oscillator wake-up in power-down mode							
1	Disable IRQ for main system oscillator wake-up in power-down mode							

.6 and .5	Not used for KS88C0716/P0716							
------------------	------------------------------	--	--	--	--	--	--	--

.4 and .3

CPU Clock (System Clock) Selection Bits ⁽¹⁾		
0	0	Divide by 16 ($f_{OSC}/16$)
0	1	Divide by 8 ($f_{OSC}/8$)
1	0	Divide by 2 ($f_{OSC}/2$)
1	1	Non-divided clock (f_{OSC}) ⁽²⁾

.2 – .0

Subsystem Clock Selection Bits ⁽³⁾			
1	0	1	Invalid setting for KS88C0716/P0716
Other value			Select main system clock (MCLK)

NOTES:

1. After a reset, the slowest clock (divided by 16) is selected as the system clock. To select faster clock speeds, load the appropriate values to CLKCON.3 and CLKCON.4.
2. If the oscillator frequency is higher than 12 MHz, this selection is invalid.
3. These selection bits are required only for systems that have a main clock and a subsystem clock. KS88C0716/P0716 use only the main oscillator clock circuit. For this reason, the setting '101B' is invalid.

EMT — External Memory Timing register

FEH

Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	—	—	—	—	—	0	—
Read/Write	R/W	—	—	—	—	—	R/W	—
Addressing Mode	Register addressing mode only							

.7

External WAIT Input Function Enable Bit	
0	Disable WAIT input function for external device
1	Enable WAIT input function for external device

.6 – .2

Not used for KS88C0716/P0716	
------------------------------	--

.1

Stack Area Selection Bit	
0	Select internal register file area
1	Select external data memory area

.0

Not used for KS88C0716/P0716	
------------------------------	--

FLAGS — System Flags Register**D5H****Set 1**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7	Carry Flag (C)	0	Operation does not generate a carry or borrow condition
		1	Operation generates a carry-out or borrow into high-order bit 7
.6	Zero Flag (Z)	0	Operation result is a non-zero value
		1	Operation result is zero
.5	Sign Flag (S)	0	Operation generates a positive number (MSB = "0")
		1	Operation generates a negative number (MSB = "1")
.4	Overflow Flag (V)	0	Operation result is $\leq +127$ or ≥ -128
		1	Operation result is $> +127$ or < -128
.3	Decimal Adjust Flag (D)	0	Add operation completed
		1	Subtraction operation completed
.2	Half-Carry Flag (H)	0	No carry-out of bit 3 or no borrow into bit 3 by addition or subtraction
		1	Addition generated carry-out of bit 3 or subtraction generated borrow into bit 3
.1	Fast Interrupt Status Flag (FIS)	0	Cleared automatically during an interrupt return (IRET)
		1	Automatically set to logic one during a fast interrupt service routine
.0	Bank Address Selection Flag (BA)	0	Bank 0 is selected (by executing the instruction SB0)
		1	Bank 1 is selected (by executing the instruction SB1)

IMR — Interrupt Mask Register**DDH****Set 1**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	—	—	X	X	X	X	X	X
Read/Write	—	—	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 Interrupt Level 7 (IRQ7) Enable Bit; P4.4–P4.7 External Interrupt (INT8–INT11)

0	Disable IRQ7 interrupt
1	Enable IRQ7 interrupt

.6 Interrupt Level 6 (IRQ6) Enable Bit; P4.2 or P4.3 External Interrupt (INT6, INT7)

0	Disable IRQ6 interrupt
1	Enable IRQ6 interrupt

.5 Interrupt Level 5 (IRQ5) Enable Bit; P4.1 External Interrupt (INT5)

0	Disable IRQ5 interrupt
1	Enable IRQ5 interrupt

.4 Interrupt Level 4 (IRQ4) Enable Bit; P4.0 External Interrupt (INT4)

0	Disable IRQ4 interrupt
1	Enable IRQ4 interrupt

.3 Interrupt Level 3 (IRQ3) Enable Bit; P2.4–P2.7 External Interrupt (INT0–INT3)

0	Disable IRQ3 interrupt
1	Enable IRQ3 interrupt

.2 Interrupt Level 2 (IRQ2) Enable Bit; Serial Receive or Transmit Interrupt

0	Disable IRQ2 interrupt
1	Enable IRQ2 interrupt

.1 Interrupt Level 1 (IRQ1) Enable Bit; Timer C or Timer D Overflow Interrupt

0	Disable IRQ1 interrupt
1	Enable IRQ1 interrupt

.0 Interrupt Level 0 (IRQ0) Enable Bit; Timer A Match Interrupt

0	Disable IRQ0 interrupt
1	Enable IRQ0 interrupt

IPH — Instruction Pointer (High Byte)**DAH****Set 1**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 – .0**Instruction Pointer Address (High Byte)**

The high-byte instruction pointer value is the upper eight bits of the 16-bit instruction pointer address (IP15–IP8). The lower byte of the IP address is located in the IPL register (DBH).

IPL — Instruction Pointer (Low Byte)**DBH****Set 1**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 – .0**Instruction Pointer Address (Low Byte)**

The low-byte instruction pointer value is the lower eight bits of the 16-bit instruction pointer address (IP7–IP0). The upper byte of the IP address is located in the IPH register (DAH).

IPR — Interrupt Priority Register**FFH****Set 1, Bank 0**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	–	–	x	x	x	x	x
Read/Write	R/W	–	–	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7, .4, and .1**Priority Control Bits for Interrupt Groups A, B, and C ⁽¹⁾**

0	0	0	Not used
0	0	1	B > C > A
0	1	0	A > B > C
0	1	1	B > A > C
1	0	0	C > A > B
1	0	1	C > B > A
1	1	0	A > C > B
1	1	1	Not used

.6**Interrupt Subgroup C Priority Control Bit**

0	IRQ6 > IRQ7
1	IRQ7 > IRQ6

.5**Interrupt Group C Priority Control Bit**

0	IRQ5 > (IRQ6, IRQ7)
1	(IRQ6, IRQ7) > IRQ5

.3**Interrupt Subgroup B Priority Control Bit**

0	IRQ3 > IRQ4
1	IRQ4 > IRQ3

.2**Interrupt Group B Priority Control Bit**

0	IRQ2 > (IRQ3, IRQ4)
1	(IRQ3, IRQ4) > IRQ2

.0**Interrupt Group A Priority Control Bit**

0	IRQ0 > IRQ1
1	IRQ1 > IRQ0

NOTE: Interrupt group A is IRQ0 and IRQ1; interrupt group B is IRQ2, IRQ3, and IRQ4; interrupt group C is IRQ5, IRQ6 and IRQ 7.

IRQ — Interrupt Request Register**DCH****Set 1**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	—	—	0	0	0	0	0	0
Read/Write	—	—	R	R	R	R	R	R
Addressing Mode	Register addressing mode only							

.7 Level 7 (IRQ7) Request Pending Bit; P4.4–P4.7 External Interrupt (INT8–INT11)

0	No IRQ7 interrupt pending
1	IRQ7 interrupt is pending

.6 Level 6 (IRQ6) Request Pending Bit; P4.2 or P4.3 External Interrupt (INT6, INT7)

0	No IRQ6 interrupt pending
1	IRQ6 interrupt is pending

.5 Level 5 (IRQ5) Request Pending Bit; P4.1 External Interrupt (INT5)

0	No IRQ5 interrupt pending
1	IRQ5 interrupt is pending

.4 Level 4 (IRQ4) Request Pending Bit; P4.0 External Interrupt (INT4)

0	No IRQ4 interrupt pending
1	IRQ4 interrupt is pending

.3 Level 3 (IRQ3) Request Pending Bit; P2.4–P2.7 External Interrupt (INT0–INT3)

0	No IRQ3 interrupt pending
1	IRQ3 interrupt is pending

.2 Level 2 (IRQ2) Request Pending Bit; Serial Receive or Transmit Interrupt

0	No IRQ2 interrupt pending
1	IRQ2 interrupt is pending

.1 Level 1 (IRQ1) Request Pending Bit; Timer C or Timer D Overflow Interrupt

0	No IRQ1 interrupt pending
1	IRQ1 interrupt is pending

.0 Level 0 (IRQ0) Request Pending Bit; Timer A Match Interrupt

0	No IRQ0 interrupt pending
1	IRQ0 interrupt is pending

P0CON — Port 0 Control Register**F0H****Set 1, Bank 0**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.4**P0.7–P0.4 Mode Selection Bits**

0	0	x	0	Input
0	1	x	0	Input, pull-up
0	0	0	1	Output, push-pull
0	0	1	1	Output, open drain
0	1	1	1	Output, open drain, pull-up
1	x	x	x	External interface (A15–A12)

.3–.0**P0.3–P0.0 Mode Selection Bits**

0	0	x	0	Input
0	1	x	0	Input, pull-up
0	0	0	1	Output, push-pull
0	0	1	1	Output, open drain
0	1	1	1	Output, open drain, pull-up
1	x	x	x	External interface (A11–A8)

P1CON — Port 1 Control Register**F1H****Set 1, Bank 0****Bit Identifier**

.7	.6	.5	.4	.3	.2	.1	.0
----	----	----	----	----	----	----	----

RESET Value

0 0 0 0 0 0 0 0

Read/Write

R/W R/W R/W R/W R/W R/W R/W R/W

Addressing Mode

Register addressing mode only

.7–.4**P1.7–P1.4 Mode Selection Bits**

0	0	x	0	Input
0	1	x	0	Input, pull-up
0	0	0	1	Output, push-pull
0	0	1	1	Output, open drain
0	1	1	1	Output, open drain, pull-up
1	x	x	x	External interface (AD7–AD4)

.3–.0**P1.3–P1.0 Mode Selection Bits**

0	0	x	0	Input
0	1	x	0	Input, pull-up
0	0	0	1	Output, push-pull
0	0	1	1	Output, open drain
0	1	1	1	Output, open drain, pull-up
1	x	x	x	External interface (AD3–AD0)

P2CONH — Port 2 Control Register (High Byte)**F2H****Set 1, Bank 0**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6**P2.7 Mode Selection Bits**

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

.5 and .4**P2.6 / Mode Selection Bits**

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

.3 and .2**P2.5 / Mode Selection Bits**

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

.1 and .0**P2.4 / Mode Selection Bits**

0	0	Input (WAIT input enabled when set bit7 of EMT register to 1)
0	1	Input, falling edge interrupt (WAIT input enabled when set bit7 of EMT register to 1)
1	0	Output, push-pull
1	1	Input, rising edge interrupt (WAIT input enabled when set bit7 of EMT register to 1)

P2CONL — Port 2 Control Register (Low Byte)**F3H****Set 1, Bank 0****Bit Identifier**

.7	.6	.5	.4	.3	.2	.1	.0
----	----	----	----	----	----	----	----

RESET Value

0 0 0 0 0 0 0 0

Read/Write

R/W R/W R/W R/W R/W R/W R/W R/W

Addressing Mode

Register addressing mode only

.7 and .6**P2.3 Mode Selection Bits**

0	x	Input
1	0	Output, push-pull
1	1	External interface DM

.5 and .4**P2.2 Mode Selection Bits**

0	x	Input
1	0	Output, push-pull
1	1	External interface R/W

.3 and .2**P2.1 Mode Selection Bits**

0	x	Input
1	0	Output, push-pull
1	1	External interface DS

.1 and .0**P2.0 Mode Selection Bits**

0	x	Input
1	0	Output, push-pull
1	1	External interface AS

P3CONH — Port 3 Control Register (High Byte)**F4H****Set 1, Bank 0**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6**P3.7 Mode Selection Bits**

0	x	Input, Enable serial data receive input at the RxD pin
1	0	Output, push-pull
1	1	RxD

.5 and .4**P3.6 Mode Selection Bits**

0	x	Input
1	0	Output, push-pull
1	1	TxD

.3 and .2**P3.5 Mode Selection Bits**

0	x	Input
1	0	Output, push-pull
1	1	TB

.1 and .0**P3.4 Mode Selection Bits**

0	x	Input
1	0	Output, push-pull
1	1	TA

P3CONL — Port 3 Control Register (Low Byte)**F5H****Set 1, Bank 0**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6**P3.3 Mode Selection Bits**

0	x	Input
1	0	Output, push-pull
1	1	Invalid setting; Do not use

.5 and .4**P3.2 Mode Selection Bits**

0	x	Input
1	0	Output, push-pull
1	1	Invalid setting; Do not use

.3 and .2**P3.1 Mode Selection Bits**

0	x	Input, TDCK
1	0	Output, push-pull
1	1	Invalid setting; Do not use

.1 and .0**P3.0 Mode Selection Bits**

0	x	Input, TCCK
1	0	Output, push-pull
1	1	Invalid setting; Do not use

P4CONH — Port 4 Control Register (High Byte)**F6H****Set 1, Bank 0**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6**P4.7 Mode Selection Bits**

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

.5 and .4**P4.6/Mode Selection Bits**

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

.3 and .2**P4.5/Mode Selection Bits**

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

.1 and .0**P4.4/Mode Selection Bits**

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

P4CONL — Port 4 Control Register (Low Byte)**F7H****Set 1, Bank 0**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6**P4.3 Mode Selection Bits**

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

.5 and .4**P4.2 Mode Selection Bits**

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

.3 and .2**P4.1 Mode Selection Bits**

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

.1 and .0**P4.0 Mode Selection Bits**

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

P4INT — Port 4 Interrupt enable Register**F8H****Set 1, Bank 0**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7**P4.7/INT 11 Mode Selection Bits**

0	Disable interrupt
1	Enable interrupt

.6**P4.6/INT 10 Mode Selection Bits**

0	Disable interrupt
1	Enable interrupt

.5**P4.5/INT 9 Mode Selection Bits**

0	Disable interrupt
1	Enable interrupt

.4**P4.4/INT 8 Mode Selection Bits**

0	Disable interrupt
1	Enable interrupt

.3**P4.3/INT 7 Mode Selection Bits**

0	Disable interrupt
1	Enable interrupt

.2**P4.2/INT 6 Mode Selection Bits**

0	Disable interrupt
1	Enable interrupt

.1**P4.1/INT 5 Mode Selection Bits**

0	Disable interrupt
1	Enable interrupt

.0**P4.0/INT 4 Mode Selection Bits**

0	Disable interrupt
1	Enable interrupt

P4PND — Port 4 Interrupt Pending Register**E7H****Set 1, Bank 0**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7

P4.7/INT 11 Mode Selection Bits	
0	Interrupt request is not pending, When write 0 pending bit is cleared
1	Interrupt request is pending

.6

P4.6/INT 10 Mode Selection Bits	
0	Interrupt request is not pending, When write 0 pending bit is cleared
1	Interrupt request is pending

.5

P4.5/INT 9 Mode Selection Bits	
0	Interrupt request is not pending, When write 0 pending bit is cleared
1	Interrupt request is pending

.4

P4.4/INT 8 Mode Selection Bits	
0	Interrupt request is not pending, When write 0 pending bit is cleared
1	Interrupt request is pending

.3

P4.3/INT 7 Mode Selection Bits	
0	Interrupt request is not pending, When write 0 pending bit is cleared
1	Interrupt request is pending

.2

P4.2/INT 6 Mode Selection Bits	
0	Interrupt request is not pending, When write 0 pending bit is cleared
1	Interrupt request is pending

.1

P4.1/INT 5 Mode Selection Bits	
0	Interrupt request is not pending, When write 0 pending bit is cleared
1	Interrupt request is pending

.0

P4.0/INT 4 Mode Selection Bits	
0	Interrupt request is not pending, When write 0 pending bit is cleared
1	Interrupt request is pending

P5CON — Port 5 Control Register

F8HSet 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.4

P5.7–P5.4 Mode Selection Bits

x	0	x	0	Input
x	1	x	0	Input, pull-up
x	0	0	1	Output, push-pull
x	0	1	1	Output, open drain
x	1	1	1	Output, open drain, pull-up

.3–.0

P5.3–P5.0 Mode Selection Bits

x	0	x	0	Input
x	1	x	0	Input, pull-up
x	0	0	1	Output, push-pull
x	0	1	1	Output, open drain
x	1	1	1	Output, open drain, pull-up

P6CON — Port 6 Control Register

F9HSet 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.4

P6.7–P6.4 Mode Selection Bits

x	x	0	0	Output, push-pull
x	x	0	1	Output, open drain
x	x	1	0	–
x	x	1	1	Output, open drain, pull-up

.3–.0

P6.3–P6.0 Mode Selection Bits

x	x	0	0	Output, push-pull
x	x	0	1	Output, open drain
x	x	1	0	–
x	x	1	1	Output, open drain, pull-up

PP — Register Page Pointer**DFH****Set 1**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.4**Destination Register Page Selection Bits**

0	0	0	0	Destination: page 0
0	0	0	1	Not used for KS88C0716/P0716
...				"
1	1	1	1	Not used for KS88C0716/P0716

.3–.0**Source Register Page Selection Bits**

0	0	0	0	Source: page 0
0	0	0	1	Not used for KS88C0716/P0716
...				"
1	1	1	1	Not used for KS88C0716/P0716

NOTE: Because only page 0 is implemented in the KS88C0716/P0716 microcontrollers, the register page pointer always points to page 0 as the source and destination address.

RP0 — Register Pointer 0**D6H****Set 1**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	1	1	0	0	0	—	—	—
Read/Write	R/W	R/W	R/W	R/W	R/W	—	—	—
Addressing Mode	Register addressing only							

.7–.3**Register Pointer 0 Address Value**

Register pointer 0 can independently point to one of the 18 8-byte working register slices in the register file. Using the register pointers RP0 and RP1, you can select two 8-byte register slices at one time as active working register space. After a reset, RP0 points to address C0H in register set 1, selecting the 8-byte working register slice C0H–C7H.

.2–.0

Not used for KS88C0716/P0716

RP1 — Register Pointer 1**D7H****Set 1**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	1	1	0	0	1	—	—	—
Read/Write	R/W	R/W	R/W	R/W	R/W	—	—	—
Addressing Mode	Register addressing only							

.7–.3**Register Pointer 1 Address Value**

Register pointer 1 can independently point to one of the 18 8-byte working register slices in the register file. Using the register pointers RP0 and RP1, you can select two 8-byte register slices at one time as active working register space. After a reset, RP1 points to address C8H in register set 1, selecting the 8-byte working register slice C8H–CFH.

.2–.0

Not used for KS88C0716/P0716

SIOCON — UART Control Register**EAH****Set 1**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6**Mode and Baud Rate Selection Bits**

0	0	SIO mode 0 (shift register, baud rate = CPU clock /6)
0	1	UART mode 1 (8-bit UART, variable baud rate)
1	0	UART mode 2 (9-bit UART, baud rate = CPU clock /16 or /32)
1	1	UART mode 3 (9-bit UART, variable baud rate)

.5**Multiprocessor Communication Enable Bit**

0	Disable multiprocessor communication feature
1	Enable the multiprocessor communication feature (for modes 2 and 3 only). In mode 2 and 3, if SIOCON.5 = "1", the receive interrupt will <i>not</i> be activated when the 9th data bit is "0". In UART mode 1, if SIOCON.5 = "1", the receive interrupt is enabled when a valid stop bit is received regardless of the 9th data bit. In SIO mode 0, SIOCON.5 should always be "0".

.4**Serial Data Receive Enable Bit**

0	Disable serial data receive function
1	Enable serial data receive function

.3**Value of the 9th Bit To Be Transmitted in UART Mode 2 or 3**

0	Transmit a "0" as the 9th data bit (valid for UART modes 2 or 3 only)
1	Transmit a "1" as the 9th data bit (valid for UART modes 2 or 3 only)

.2**Value of the 9th Bit That Was Received in UART Mode 2 or 3**

In SIO modes 2 and 3, SIOCON.2 is the 9th data bit that was received (including the start bit). In mode 1, if SIOCON.5 = "0", SIOCON.2 is the Stop bit. In mode 0, SIOCON.2 is not used because a Start bit is not required.	
--	--

.1**UART Receive Interrupt Enable Bit**

0	Disable UART receive interrupt
1	Enable UART receive interrupt

.0**UART Transmit Interrupt Enable Bit**

0	Disable UART transmit interrupt
1	Enable UART transmit interrupt

SIOPND — UART Interrupt Pending Register**EBH****Set 1**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	—	—	—	—	—	—	0	0
Read/Write	—	—	—	—	—	—	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 – .2

Not used for KS88C4404

.1**Serial Data Receive Interrupt Pending Flag**

0	Serial data receive interrupt is not pending, when write 0 pending bit is cleared
1	Serial data receive interrupt is pending

.0**Serial data Transmit Interrupt Pending Flag**

0	Serial data transmit interrupt is not pending, when write 0 pending bit is cleared
1	Serial data transmit interrupt is pending

NOTE: In order to avoid programming errors, we recommend that you use Load instructions only (except for LDB) to modify SIOPND register values.

SPH — Stack Pointer (High Byte)**D8H****Set 1**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.0**Stack Pointer Address (High Byte)**

The high-byte stack pointer value is the upper eight bits of the 16-bit stack pointer address (SP15–SP8). The lower byte of the stack pointer value is located in register SPL (D9H). The SP value is undefined following a reset.

SPL — Stack Pointer (Low Byte)**D9H****Set 1**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.0**Stack Pointer Address (Low Byte)**

The low-byte stack pointer value is the lower eight bits of the 16-bit stack pointer address (SP7–SP0). The upper byte of the stack pointer value is located in register SPH (D8H). The SP value is undefined following a reset.

SYM — System Mode Register**DEH****Set 1**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	–	–	x	x	x	0	0
Read/Write	R/W	–	–	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 Tri-State External Interface Control Bit

0	Normal operation (disable tri-state operation)
1	Set external interface lines to high impedance (enable tri-state operation)

.6 and .5 Not used for KS88C0716/P0716**.4–.2 Fast Interrupt Level Selection Bits ⁽¹⁾**

0	0	0	IRQ0
0	0	1	IRQ1
0	1	0	IRQ2
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

.1 Fast Interrupt Enable Bit ⁽²⁾

0	Disable fast interrupt processing
1	Enable fast interrupt processing

.0 Global Interrupt Enable Bit ⁽³⁾

0	Disable global interrupt processing
1	Enable global interrupt processing

NOTES:

1. You can select only one interrupt level at a time for fast interrupt processing. IRQ3 cannot be selected for fast interrupt processing.
2. Setting SYM.1 to "1" enables fast interrupt processing for the interrupt level currently selected by SYM.2–SYM.4.
3. Following a reset, you must enable global interrupt processing by executing an EI instruction (not by writing a "1" to SYM.0).

T0CON — Timer 0 Control Register**EEH****Set 1, Bank 0**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .4**4 Bit Prescaler**

0	0	0	0	No division
0	0	0	1	Divided by 2
•	•	•	•	Divided by 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
1	1	1	1	Divided by 16

.3**Timer A and B Clock Selection Bit**

0	F _{OSC} /1024
1	CPU clock

.2**Timer A Interrupt Enable Bit**

0	Disable Timer A interrupt
1	Enable Timer A interrupt

.1**Timer A Interrupt Pending Bit**

0	Timer A Interrupt is not pending, When write 0 pending bit is cleared
1	Timer A Interrupt is pending

.0**Timer A PWM and Interval mode Selection Bit**

0	Select interval mode
1	Select PWM mode

T1CON — Timer 1 Control Register**FAH****Set 1, Bank 0**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	–	0	0	0	0	0	0
Read/Write	R/W	–	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 Baud Rate Selection Bit

0	Normal Baud Rate
1	Double Baud Rate

.6 Not used for KS88C0716/P0716**.5 Timer D Interrupt Pending Bit**

0	Timer D Interrupt is not pending, when write 0 pending bit is cleared
1	Timer D Interrupt is pending, when write 0 pending bit is cleared

.4 Timer C Interrupt Pending Bit

0	Timer C interrupt is not pending
1	Timer C interrupt is pending

.3 Timer D Interrupt Enable Bit

0	Timer D interrupt disable
1	Timer D interrupt enable

.2 Timer C Interrupt Enable Bit

0	Timer C interrupt disable
1	Timer C interrupt enable

.1 Timer D RUN Bit

0	Timer D counter stop
1	Timer D counter run

.0 Timer C RUN Bit

0	Timer C counter stop
1	Timer C counter run

T1MOD — Timer 1 Mode Register**FBH****Set 1, Bank 0**

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 **Timer D Gate Enable Bit (Same as Timer C Gate Enable Bit)**

.6 **Timer D Clock Input Selection Bit (Same as Timer C Clock Input Selection Bit)**

.5 and .4 **Timer D Mode Selection Bit (Same as Timer C Mode Selection Bit)**

.3 **Timer C Gate Enable Bit**

0	Timer C gate disable
1	Timer C gate enable

.2 **Timer C Clock Input Selection Bit**

0	CPU/6
1	External clock input(Max input frequency is CPU/12)

.1 and .0 **Timer C Mode Selection Bit**

0	0	13-bit timer/counter mode
0	1	16-bit timer/counter mode
1	0	8-bit auto reload timer/counter mode
1	1	Timer C: TCL: Controlled by timer C control bits and make Timer C INT TCH: Controlled by timer D control bits and make Timer D INT Timer D: Timer/counter is disable

NOTES

5

INTERRUPT STRUCTURE

OVERVIEW

The SAM8 interrupt structure has three basic components: levels, vectors, and sources. The CPU recognizes eight interrupt levels and supports up to 128 interrupt vectors. When a specific interrupt level has more than one vector address, the vector priorities are established in hardware. Each vector can have one or more sources.

Levels

Interrupt levels are the main unit for interrupt priority assignment and recognition. All peripherals and I/O blocks can issue interrupt requests. In other words, peripheral and I/O operations are interrupt-driven. There are eight interrupt levels: IRQ0–Q7, also called level 0–level 7. Each interrupt level directly corresponds to an interrupt request number (IRQn). The total number of interrupt levels used in the interrupt structure varies from device to device. The KS88C0716/P0716 interrupt structure recognizes eight interrupt levels, IRQ0–IRQ7.

The interrupt level numbers 0 through 7 do not necessarily indicate the relative priority of the levels. They are simply identifiers for the interrupt levels that are recognized by the CPU. The relative priority of different interrupt levels is determined by settings in the interrupt priority register, IPR. Interrupt group and subgroup logic controlled by IPR settings lets you define more complex priority relationships between different levels.

Vectors

Each interrupt level can have one or more interrupt vectors, or it may have no vector address assigned at all. The maximum number of vectors that can be supported for a given level is 128. (The actual number of vectors used for KS88-series devices will always be much smaller.) If an interrupt level has more than one vector address, the vector priorities are set in hardware. KS88C0716/P0716 have seventeen vectors — one corresponding to each of the seventeen possible interrupt sources.

Sources

A source is any peripheral that generates an interrupt. A source can be an external pin or a counter overflow, for example. Each vector can have several interrupt sources. In the KS88C0716/P0716 interrupt structure, each source has its own vector address.

When a service routine starts, the respective pending bit is either cleared automatically by hardware or is must be cleared "manually" by program software. The characteristics of the source's pending mechanism determine which method is used to clear its respective pending bit.

INTERRUPT TYPES

The three components of the SAM8 interrupt structure described above (levels, vectors, and sources) are combined to determine the interrupt structure of an individual device and to make full use of its available interrupt logic. There are three possible combinations of interrupt structure components, called interrupt types 1, 2, and 3. The types differ in the number of vectors and interrupt sources assigned to each level (see Figure 5-1):

- Type 1: One level (IRQn) + one vector (V₁) + one source (S₁)
- Type 2: One level (IRQn) + one vector (V₁) + multiple sources (S₁ – S_n)
- Type 3: One level (IRQn) + multiple vectors (V₁ – V_n) + multiple sources (S₁ – S_n, S_{n+1} – S_{n+m})

In the KS88C0716/P0716 microcontrollers, only interrupt types 1 and 3 are implemented.

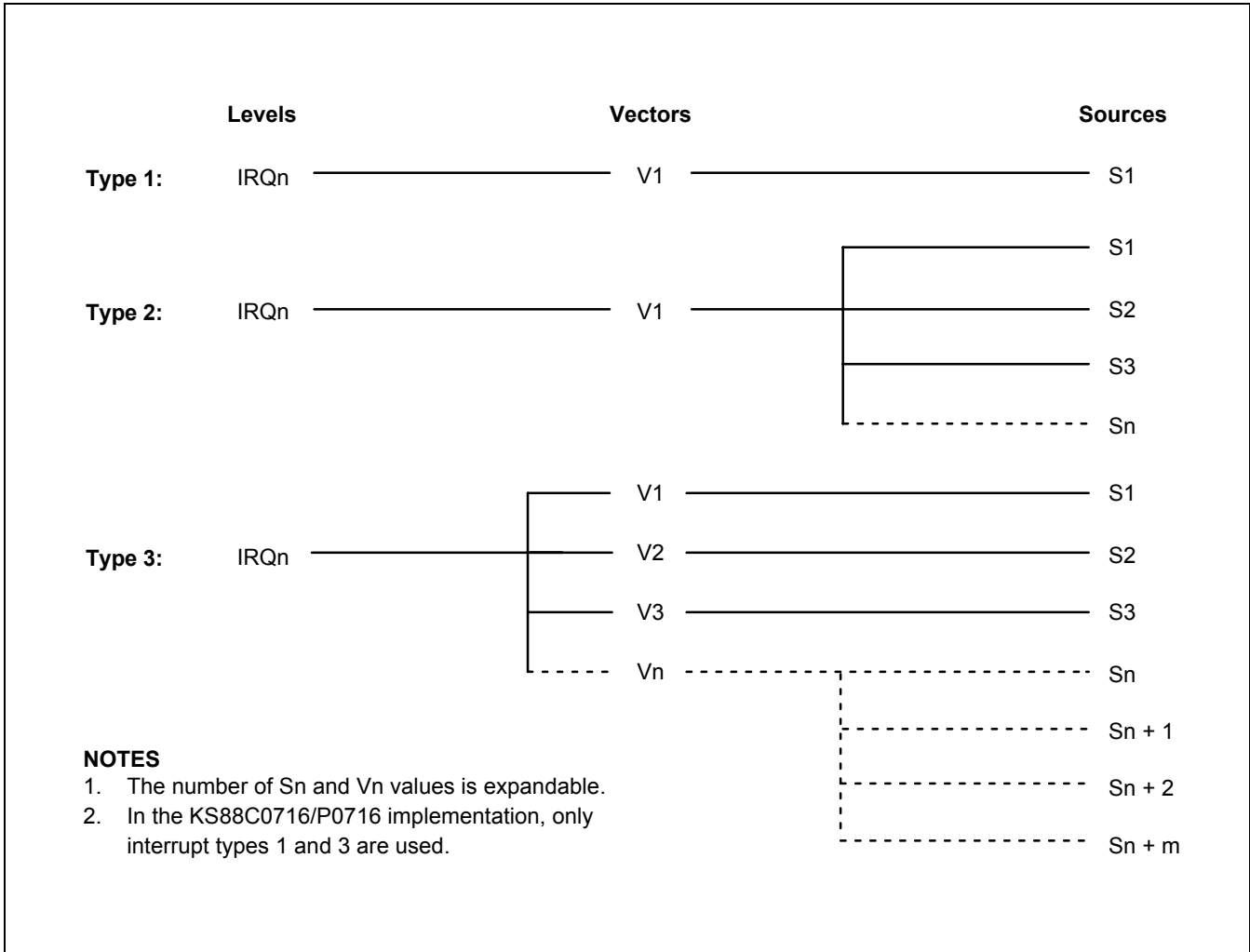


Figure 5-1. KS88-Series Interrupt Types

KS88C0716/P0716 INTERRUPT STRUCTURE

The KS88C0716/P0716 microcontroller supports seventeen interrupt sources. Each interrupt source has a corresponding interrupt vector address. Eight interrupt levels are used in the device-specific interrupt structure, which is shown in Figure 5-2.

When multiple interrupt levels are active, the interrupt priority register (IPR) determines the order in which contending interrupts are to be serviced. If multiple interrupts occur within the same interrupt level, the interrupt with the lowest vector address is usually processed first.

When the CPU grants an interrupt request, interrupt processing starts: All other interrupts are disabled and the program counter value and status flags are pushed to stack. The starting address of the service routine is fetched from the appropriate vector address (plus the next 8-bit value to concatenate the full 16-bit address) and the service routine is executed.

Level		Vector		Source	Reset (Clear)
IRQ0	—————	BEH	—————	Timer A match	S/W
IRQ1	┌———	F4H	—————	Timer C over flow	S/W
		F6H	—————	Timer D over flow	S/W
IRQ2	┌———	F0H	—————	Serial receive interrupt	S/W
		F2H	—————	Serial transmit interrupt	S/W
IRQ3	┌———	E8H	—————	P2.4 external interrupt	H/W
		EAH	—————	P2.5 external interrupt	H/W
		ECH	—————	P2.6 external interrupt	H/W
		EEH	—————	P2.7 external interrupt	H/W
IRQ4	—————	D8H	—————	P4.0 external interrupt	S/W
IRQ5	—————	DAH	—————	P4.1 external interrupt	S/W
IRQ6	┌———	DCH	—————	P4.2 external interrupt	S/W
		DEH	—————	P4.3 external interrupt	S/W
IRQ7	┌———	E0H	—————	P4.4 external interrupt	S/W
		E2H	—————	P4.5 external interrupt	S/W
		E4H	—————	P4.6 external interrupt	S/W
		E6H	—————	P4.7 external interrupt	S/W

NOTES:

1. Within a given interrupt level, the low vector address has high priority. For example, E0H has higher priority than E2H within the level IRQ7. The priorities within each level are set at the factory.
2. External interrupts are triggered by a rising or falling edge, depending on the corresponding control register setting.
3. Because port2 has no interrupt pending register, interrupt pending flags are cleared by firmware.

Figure 5-2. KS88C0716 Interrupt Structure

INTERRUPT VECTOR ADDRESSES

All interrupt vector addresses for the KS88C0716/P0716 interrupt structure are stored in the vector address area of the ROM, 00H–FFH, (see Figure 5-3). You can allocate unused locations in the vector address area as normal program memory. If you do so, please be careful not to overwrite any of the stored vector addresses. (Table 5-1 lists all vector addresses.)

The program reset address in the ROM is 0100H.

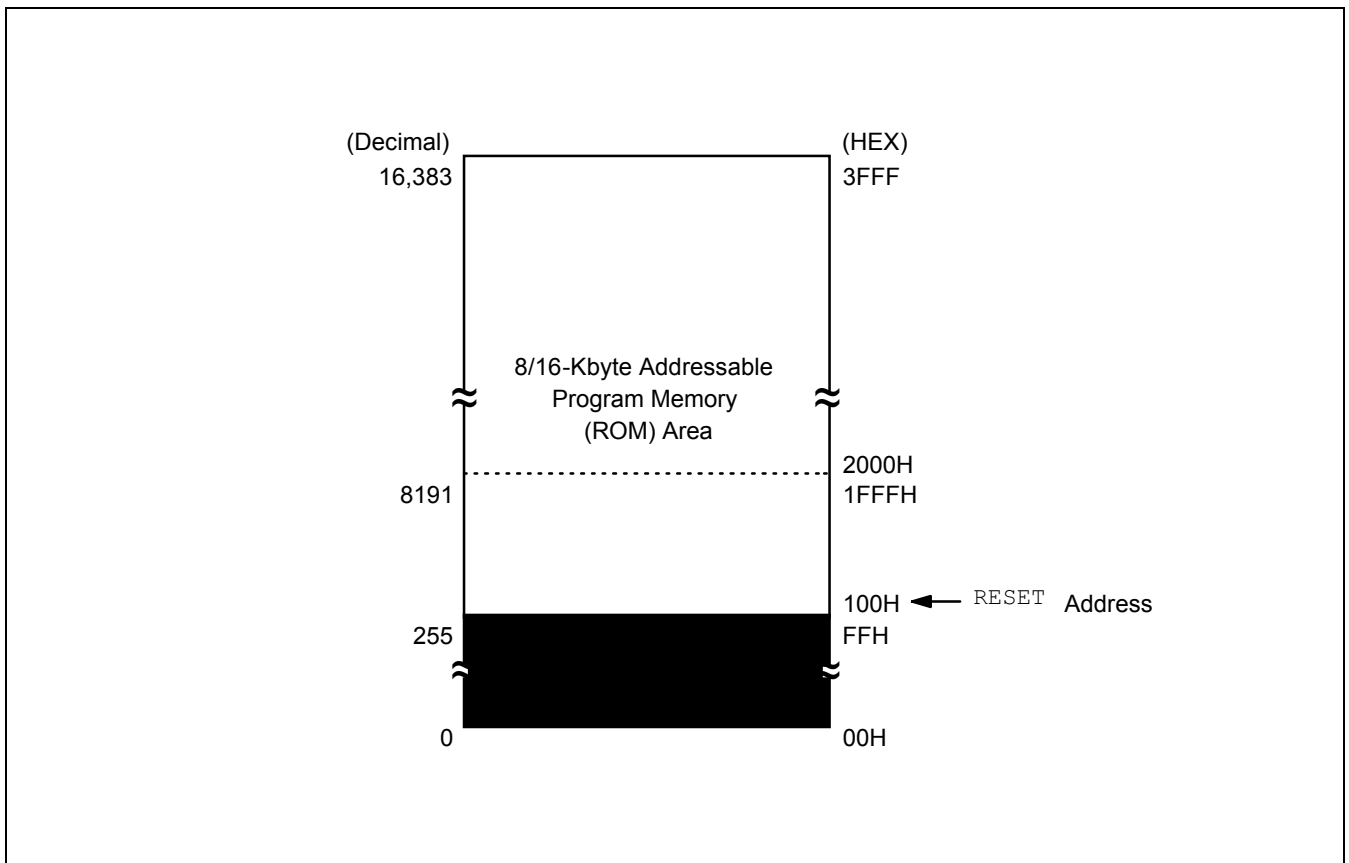


Figure 5-3. ROM Vector Address Area

Table 5-1. KS88C0716 Interrupt Vectors

Vector Address		Interrupt Source	Request		Reset/Clear	
Decimal Value	Hex Value		Interrupt Level	Priority in Level	H/W	S/W
246	F6H	Timer D overflow	IRQ1	1		√
244	F4H	Timer C overflow		0		
242	F2H	Serial transmit interrupt (TI)	IRQ2	1		√
240	F0H	Serial receive interrupt (RI)		0		
238	EEH	P2.7 external interrupt	IRQ3	3	√	
236	ECH	P2.6 external interrupt		2		
234	EAH	P2.5 external interrupt		1		
232	E8H	P2.4 external interrupt		0		
230	E6H	P4.7 external interrupt	IRQ7	3		√
228	E4H	P4.6 external interrupt		2		
226	E2H	P4.5 external interrupt		1		
224	E0H	P4.4 external interrupt		0		
222	DEH	P4.3 external interrupt	IRQ6	1		√
220	DCH	P4.2 external interrupt		0		
218	DAH	P4.1 external interrupt	IRQ5	—		√
216	D8H	P4.0 external interrupt	IRQ4	—		√
190	BEH	Timer A match	IRQ0	—		√

NOTES:

1. Interrupt priorities are identified in inverse order: '0' is highest priority, '1' is the next highest, and so on.
2. If two or more interrupts within the same level contend, the interrupt with the lowest vector address has priority over one with a higher vector address. These priorities within levels are preset at the factory. For example, in interrupt level IRQ2 the highest-priority interrupt vector is the SIO receive interrupt, F0H; the lowest-priority interrupt within the level is the SIO transmit, interrupt vector F2H. (F2H is a higher address than F0H and therefore has lower priority.)

ENABLE/DISABLE INTERRUPT INSTRUCTIONS (EI, DI)

Executing the Enable Interrupts (EI) instruction enables the interrupt structure. All interrupts are then serviced as they occur, and according to the established priorities.

NOTE

The system initialization routine that is executed following a reset must always contain an EI instruction (assuming one or more interrupts are used in the application).

During normal operation, you can execute the DI (Disable Interrupt) instruction at any time to globally disable interrupt processing. The EI and DI instructions change the value of bit 0 in the SYM register. Although you can manipulate SYM.0 directly to enable or disable interrupts, we recommend that you use the EI and DI instructions instead.

SYSTEM-LEVEL INTERRUPT CONTROL REGISTERS

In addition to the control registers for specific interrupt sources, four system-level registers control interrupt processing:

- The interrupt mask register, IMR, enables (un-masks) or disables (masks) interrupt levels.
- The interrupt priority register, IPR, controls the relative priorities of interrupt levels.
- The interrupt request register, IRQ, contains interrupt pending flags for each interrupt level (as opposed to each interrupt source).
- The system mode register, SYM, enables or disables global interrupt processing. (SYM settings also enable fast interrupts and control the activity of external interface, if implemented.)

Table 5-2. Interrupt Control Register Overview

Control Register	ID	R/W	Function Description
Interrupt mask register	IMR	R/W	Bit settings in the IMR register enable or disable interrupt processing for each of the eight interrupt levels, IRQ0–IRQ7.
Interrupt priority register	IPR	R/W	Controls the relative processing priorities of the interrupt levels. The eight levels of the KS88C0716/P0716 are organized into three groups: A, B, and C. Group A is IRQ0 and IRQ1, group B is IRQ2–IRQ4, and group C is IRQ5–IRQ7.
Interrupt request register	IRQ	R	This register contains a request pending bit for each of the eight interrupt levels, IRQ0–IRQ7.
System mode register	SYM	R/W	Dynamic global interrupt processing enable and disable, fast interrupt processing.

INTERRUPT PROCESSING CONTROL POINTS

Interrupt processing can therefore be controlled in two ways: globally or by specific interrupt level and source. The system-level control points in the interrupt structure are, therefore:

- Global interrupt enable and disable (by EI and DI instructions or by direct manipulation of SYM.0)
- Interrupt level enable/disable settings (IMR register)
- Interrupt level priority settings (IPR register)
- Interrupt source enable/disable settings in the corresponding peripheral control registers

NOTE

When writing the part of your application program that handles interrupt processing, be sure to include the necessary register file address (register pointer) information.

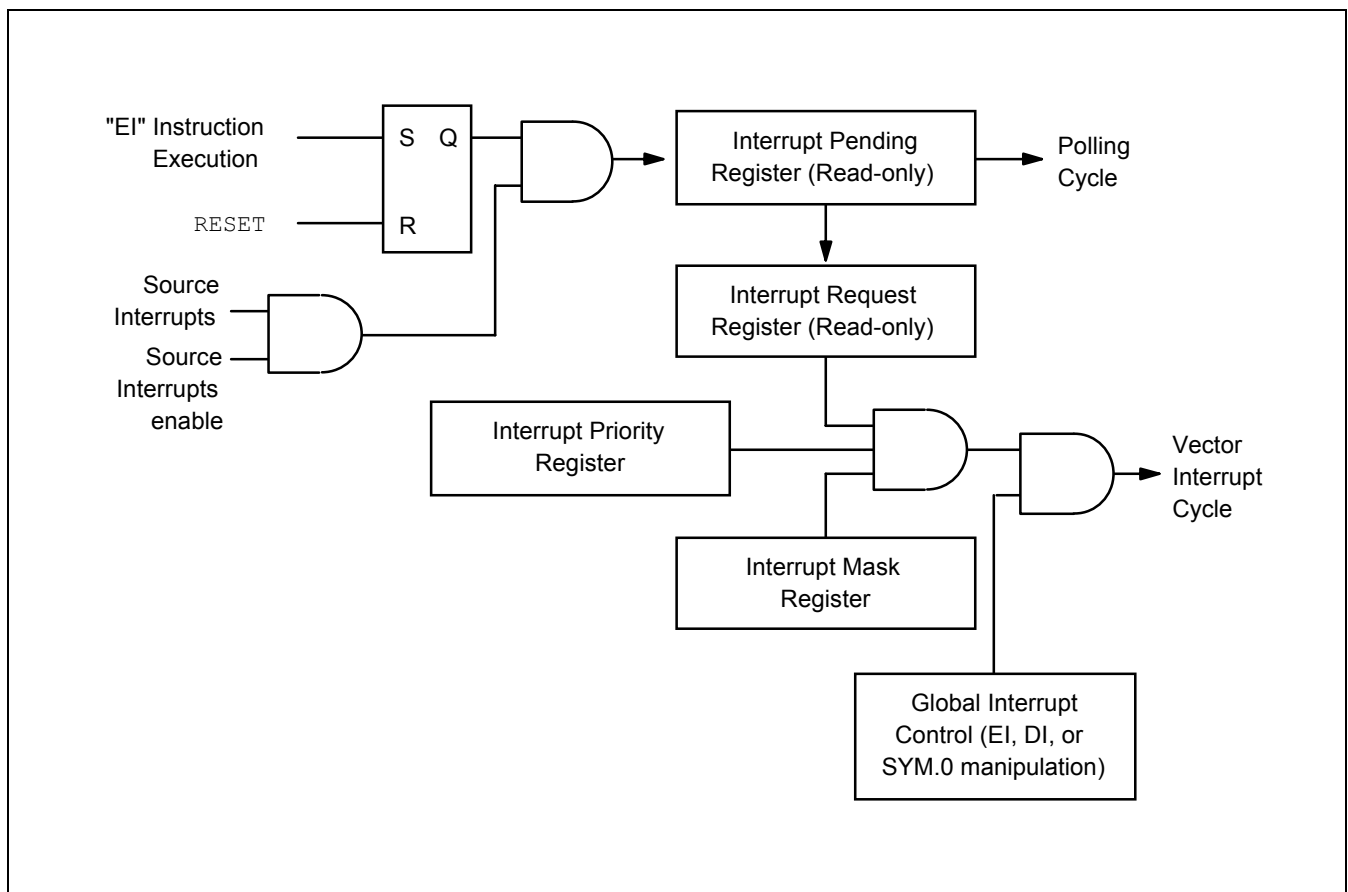


Figure 5-4. Interrupt Function Diagram

SYSTEM MODE REGISTER (SYM)

The system mode register, SYM (set 1, DEH), is used to globally enable and disable interrupt processing and to control fast interrupt processing. Figure 5-5 shows the effect of the various control settings.

A reset clears SYM.7, SYM.1, and SYM.0 to "0" and the other SYM bit values (for fast interrupt level selection) are undetermined.

The instructions EI and DI enable and disable global interrupt processing, respectively, by modifying the bit 0 value of the SYM register. An Enable Interrupt (EI) instruction must be included in the initialization routine, which follows a reset operation, in order to enable interrupt processing. Although you can manipulate SYM.0 directly to enable and disable interrupts during normal operation, we recommend using the EI and DI instructions for this purpose.

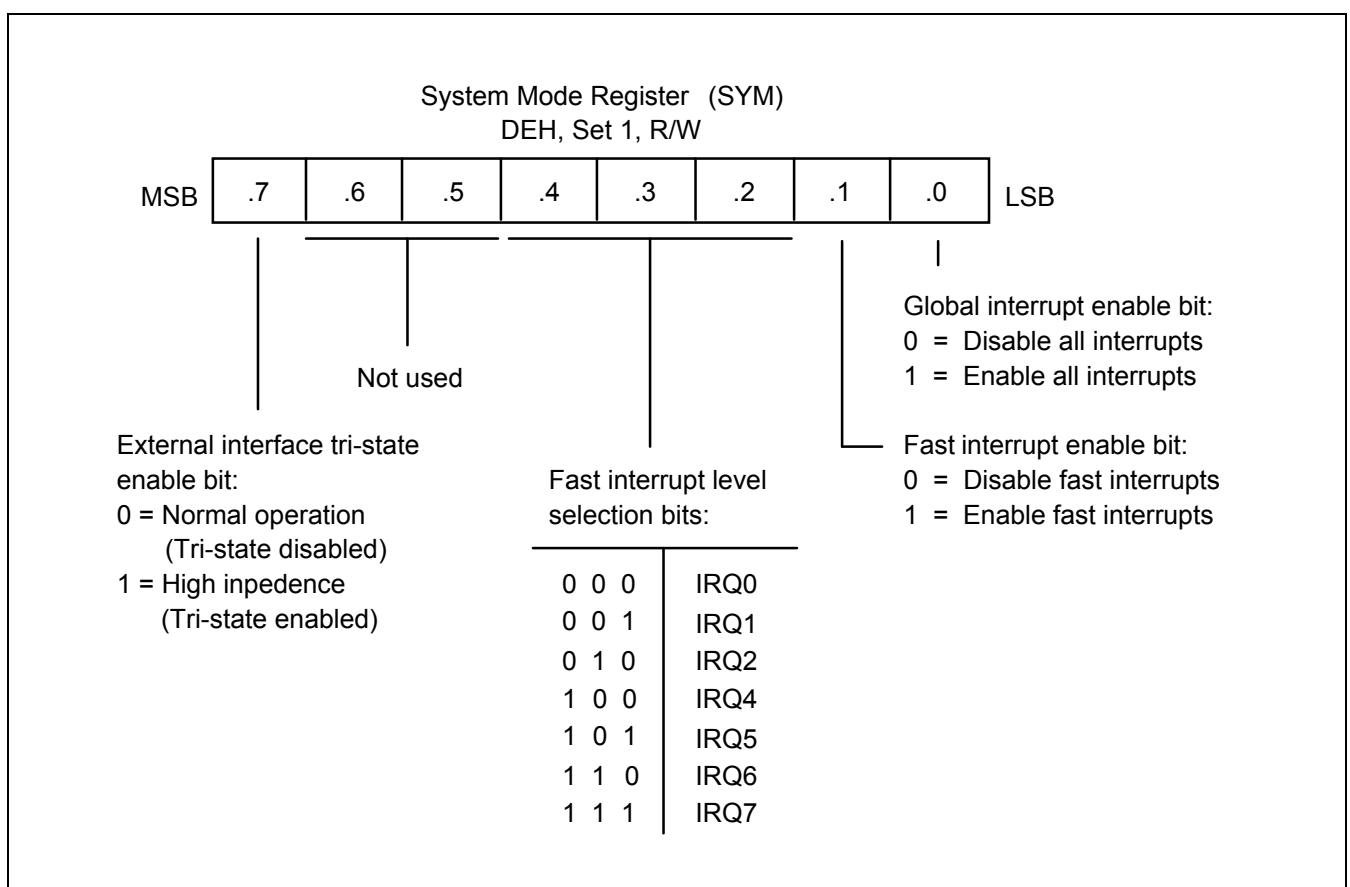


Figure 5-5. System Mode Register (SYM)

INTERRUPT MASK REGISTER (IMR)

The interrupt mask register, IMR (set 1, DDH) is used to enable or disable interrupt processing for individual interrupt levels. After a reset, all IMR bit values are undetermined and must therefore be written to their required settings by the initialization routine.

Each IMR bit corresponds to a specific interrupt level: bit 1 to IRQ1, bit 2 to IRQ2, and so on. When the IMR bit of an interrupt level is cleared to "0", interrupt processing for that level is disabled (masked). When you set a level's IMR bit to "1", interrupt processing for the level is enabled (not masked).

The IMR register is mapped to register location DDH in set 1. Bit values can be read and written by instructions using the Register addressing mode.

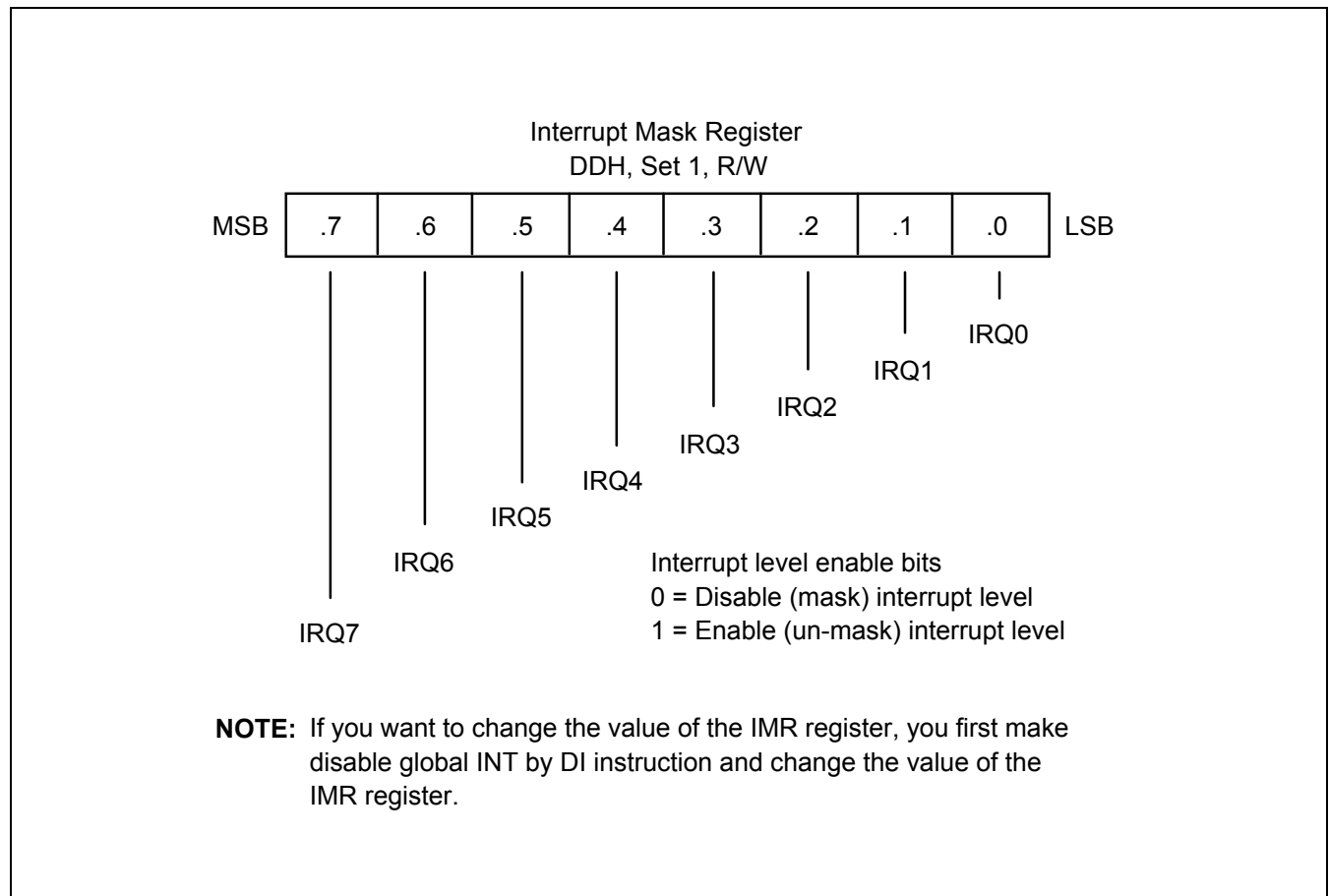


Figure 5-6. Interrupt Mask Register (IMR)

INTERRUPT PRIORITY REGISTER (IPR)

The interrupt priority register, IPR (set 1, bank 0, FFH), is used to set the relative priorities of the interrupt levels used in the microcontroller's interrupt structure. After a reset, all IPR bit values are undetermined and must therefore be written to their required settings by the initialization routine.

When more than one interrupt source is active, the source with the highest priority level is serviced first. If both sources belong to the same interrupt level, the source with the lowest vector address usually has priority. (This priority is fixed in hardware.)

To support programming of the relative interrupt level priorities, they are organized into groups and subgroups by the interrupt logic. Please note that these groups (and subgroups) are used only by IPR logic for the IPR register priority definitions (see Figure 5-7):

- Group A IRQ0, IRQ1
- Group B IRQ2, IRQ3, IRQ4
- Group C IRQ5, IRQ6, IRQ7

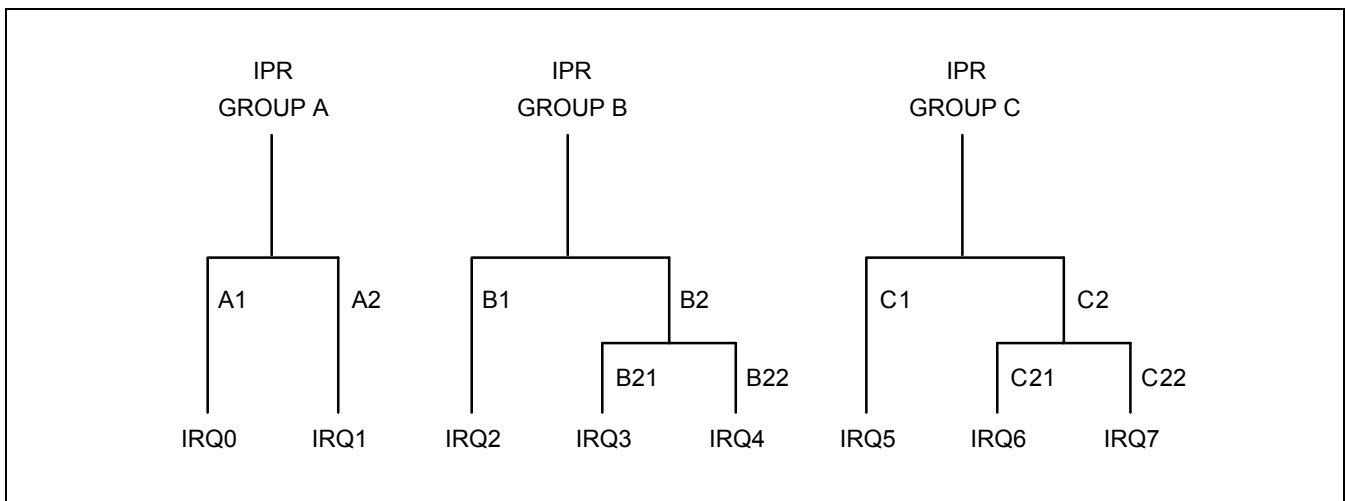


Figure 5-7. Interrupt Request Priority Groups

As you can see in Figure 5-8, IPR.7, IPR.4, and IPR.1 control the relative priority of interrupt groups A, B, and C. For example, the setting '001B' for these bits would select the group relationship B > C > A; the setting '101B' would select the relationship C > B > A.

The functions of the other IPR bit settings are as follows:

- Interrupt group C has a subgroup to provide an additional priority relationship between for interrupt levels 5, 6, and 7. IPR.6 defines the possible subgroup C relationships.
- IPR.5 controls the relative priorities of group C interrupts.
- Interrupt group B has a subgroup to provide an additional priority relationship between for interrupt levels 2, 3, and 4. IPR.3 defines the possible subgroup B relationships.
- IPR.2 controls the relative priorities of group B interrupts.
- IPR.0 controls the relative priority setting of IRQ0 and IRQ1 interrupts.

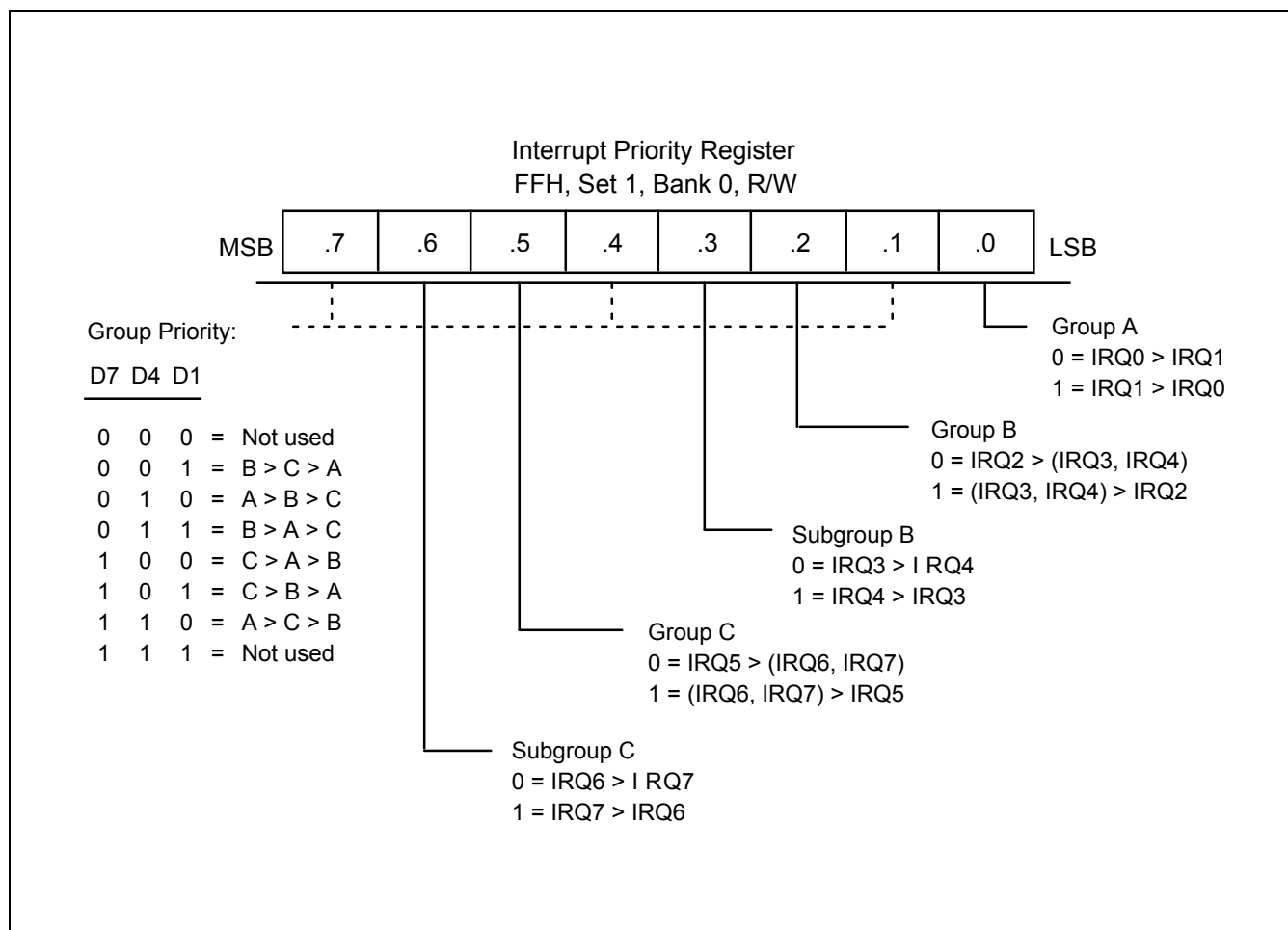


Figure 5-8. Interrupt Priority Register (IPR)

INTERRUPT REQUEST REGISTER (IRQ)

You can poll bit values in the interrupt request register, IRQ (set 1, DCH), to monitor interrupt request status for all levels in the microcontroller' interrupt structure. Each bit corresponds to the interrupt level of the same number: bit 0 to IRQ0, bit 1 to IRQ1, and so on. A "0" indicates that no interrupt request is currently being issued for that level; a "1" indicates that an interrupt request has been generated for that level.

IRQ bit values are read-only addressable using Register addressing mode. You can read (test) the contents of the IRQ register at any time using bit or byte addressing to determine the current interrupt request status of specific interrupt levels. After a reset, all IRQ status bits are cleared to "0". You can poll IRQ register values even if a DI instruction has been executed (that is, if global interrupt processing is disabled). If an interrupt occurs while the interrupt structure is disabled, the CPU will not service it. You can, however, still detect the interrupt request by polling the IRQ register. In this way, you can determine which events occurred while the interrupt structure was globally disabled.

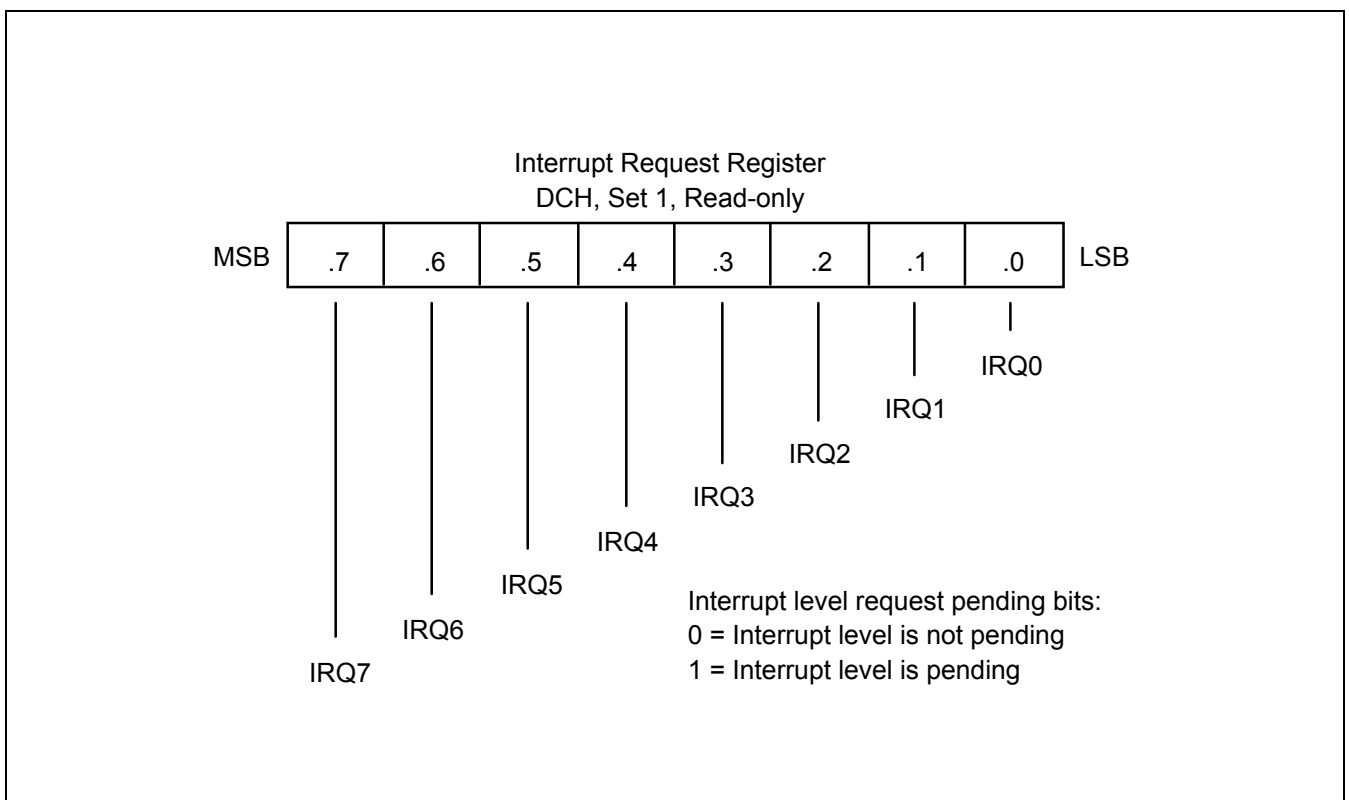


Figure 5-9. Interrupt Request Register (IRQ)

INTERRUPT PENDING FUNCTION TYPES

Overview

There are two types of interrupt pending bits: One type is automatically cleared by hardware after the interrupt service routine is acknowledged and executed; the other type must be cleared by the application program's interrupt service routine.

Pending Bits Cleared Automatically by Hardware

For interrupt pending bits that are cleared automatically by hardware, interrupt logic sets the corresponding pending bit to "1" when a request occurs. It then issues an IRQ pulse to inform the CPU that an interrupt is waiting to be serviced. The CPU acknowledges the interrupt source, executes the service routine, and clears the pending bit to "0". This type of pending bit is not mapped and cannot, therefore, be read or written by application software.

In the KS88C0716/P0716 interrupt structure, External interrupt INT0–INT3 interrupts (IRQ3) belong to this category of interrupts whose pending conditions are cleared automatically by hardware.

Pending Bits Cleared by the Service Routine

The second type of pending bit must be cleared by program software. The service routine must clear the appropriate pending bit before a return-from-interrupt subroutine (IRET) occurs. To do this, a "0" must be written to the corresponding pending bit location in the source or control register.

In the KS88C0716/P0716 interrupt structure, pending conditions for all interrupt sources *except* IRQ3 interrupt must be cleared by the program software's interrupt service routine.

INTERRUPT SOURCE POLLING SEQUENCE

The interrupt request polling and servicing sequence is as follows:

1. A source generates an interrupt request by setting the interrupt request bit to "1".
2. The CPU polling procedure identifies a pending condition for that source.
3. The CPU checks the source's interrupt level.
4. The CPU generates an interrupt acknowledge signal.
5. Interrupt logic determines the interrupt's vector address.
6. The service routine starts and the source's pending bit is cleared to "0" (by hardware or by software).
7. The CPU continues polling for interrupt requests.

INTERRUPT SERVICE ROUTINES

Before an interrupt request can be serviced, the following conditions must be met:

- Interrupt processing must be globally enabled (EI, SYM.0 = "1")
- The interrupt level must be enabled (IMR register)
- The interrupt level must have the highest priority if more than one level is currently requesting service
- The interrupt must be enabled at the interrupt's source (peripheral control register)

If all of the above conditions are met, the interrupt request is acknowledged at the end of the instruction cycle. The CPU then initiates an interrupt machine cycle that completes the following processing sequence:

1. Reset (clear to "0") the interrupt enable bit in the SYM register (SYM.0) to disable all subsequent interrupts.
2. Save the program counter (PC) and status flags to the system stack.
3. Branch to the interrupt vector to fetch the address of the service routine'.
4. Pass control to the interrupt service routine.

When the interrupt service routine is completed, the CPU issues an Interrupt Return (IRET). The IRET restores the PC and status flags and sets SYM.0 to "1", allowing the CPU to process the next interrupt request.

GENERATING INTERRUPT VECTOR ADDRESSES

The interrupt vector area in the ROM (00H–FFH) contains the addresses of interrupt service routines that correspond to each level in the interrupt structure. Vectored interrupt processing follows this sequence:

1. Push the program counter's low-byte value to the stack.
2. Push the program counter's high-byte value to the stack.
3. Push the FLAG register values to the stack.
4. Fetch the service routine's high-byte address from the vector location.
5. Fetch the service routine's low-byte address from the vector location.
6. Branch to the service routine specified by the concatenated 16-bit vector address.

NOTE

A 16-bit vector address always begins at an even-numbered ROM address within the range 00H - FFH.

NESTING OF VECTORED INTERRUPTS

It is possible to nest a higher-priority interrupt request while a lower-priority request is being serviced. To do this, you must follow these steps:

1. Push the current 8-bit interrupt mask register (IMR) value to the stack (PUSH IMR).
2. Load the IMR register with a new mask value that enables only the higher priority interrupt.
3. Execute an EI instruction to enable interrupt processing (a higher priority interrupt will be processed if it occurs).
4. When the lower-priority interrupt service routine ends, execute DI, and restore the IMR to its original value by returning the previous mask value from the stack (POP IMR).
5. Execute an IRET.

Depending on the application, you may be able to simplify the above procedure to some extent.

INSTRUCTION POINTER (IP)

The instruction pointer (IP) is used by all KS88-series microcontrollers to control the optional high-speed interrupt processing feature called *fast interrupts*. The IP consists of register pair DAH and DBH. The IP register names are IPH (high byte, IP15–IP8) and IPL (low byte, IP7–IP0).

FAST INTERRUPT PROCESSING

The feature called *fast interrupt processing* lets you specify that an interrupt within a given level be completed in approximately six clock cycles instead of the usual 22 clock cycles. SYM.4 - SYM.2 are used to select a specific interrupt level for fast processing and SYM.1 enables or disables fast interrupt processing.

Two other system registers support fast interrupt processing:

- The instruction pointer (IP) contains the starting address of the service routine (and is later used to swap the program counter values), and
- When a fast interrupt occurs, the contents of the FLAGS register is stored in an unmapped, dedicated register called FLAGS' (FLAGS prime).

NOTE

1. For the KS88C0716/P0716 microcontrollers, the service routine for any of the seven interrupt levels (except IRQ3) can be selected for fast interrupt processing.
2. If you want to use a fast interrupt in multi source interrupt vector, the fast interrupt may not be processed when you use two sources as interrupt vector in normal mode. But it is possible when you use only one source as interrupt vector.

Procedure for Initiating Fast Interrupts

To initiate fast interrupt processing, follow these steps:

1. Load the start address of the service routine into the instruction pointer (IP).
2. Load the interrupt level number (IRQn) into the fast interrupt selection field (SYM.4–SYM.2)
3. Write a "1" to the fast interrupt enable bit in the SYM register.

Fast Interrupt Service Routine

When an interrupt occurs in the level selected for fast interrupt processing, the following events occur:

1. The contents of the instruction pointer and the PC are swapped.
2. The FLAG register values are written to the FLAGS' ("FLAGS prime") register.
3. The fast interrupt status bit in the FLAGS register is set.
4. The interrupt is serviced.
5. Assuming that the fast interrupt status bit is set, when the fast interrupt service routine ends, the instruction pointer and PC values are swapped back.
6. The content of FLAGS' ("FLAGS prime") is copied automatically back to the FLAGS register.
7. The fast interrupt status bit in FLAGS is cleared automatically.

Relationship to Interrupt Pending Bit Types

As described previously, there are two types of interrupt pending bits: One type is automatically cleared by hardware after the interrupt service routine is acknowledged and executed, and the other type must be cleared by the application program's interrupt service routine. You can select fast interrupt processing for interrupts with either type of pending condition clear function — by hardware or by software.

Programming Guidelines

Remember that the only way to enable/disable a fast interrupt is to set/clear the fast interrupt enable bit in the SYM register, SYM.1. Executing an EI or DI instruction globally enables or disables all interrupt processing, including fast interrupts.

NOTE

If you use fast interrupts, remember to load the IP with a new start address when the fast interrupt service routine ends.

PROGRAMMING TIP — Setting Up the KS88C0716 Interrupt Control Structure

This example shows how to enable interrupts for select interrupt sources, disable interrupt for other sources, and to set interrupt priorities for the KS88C0716. The program does the following:

- Enable interrupts for port 4 pins P4.4–P4.7, timer A, and timer C
- Disable timer D, serial receive and transmit, backup timer, and P4.0–P4.3 interrupts
- Set interrupt priorities as P4.4–P4.7 > timer A > timer C

```

•
•
•
DI                      ; Disable interrupts
LD      IMR,#85H         ; IRQ0, IRQ2, IRQ7 are selected
LD      IPR,#80H         ; IRQ7 > IRQ0 > IRQ2
LD      TADATA,#0FH      ;
LD      T0CON,#84H       ; Timer A interrupt enable
LD      TCH,#0H          ;
LD      T1CON,#05H       ; Normal baud rate, timer C interrupt enable
LD      T1MOD,#31H       ; Timer D disable, 16-bit timer mode for timer C
LD      P4CONH,#55H      ; Input, rising edge interrupt selection
LD      P4PND,#0H        ; Reset all port 4 pending registers
LD      P4INT,#0F0H      ; Enable interrupts for port 4 pins, P4.4–P4.7
•
•
•
EI                      ; Enable interrupts
    
```

Assuming interrupt sources and priorities have been set by the above instruction sequence, you could then select interrupt level 0, 2, or 7 for fast interrupt processing. The following instructions enable fast interrupt processing for IRQ7:

```

DI                      ; Disable interrupts
LDW     IPH,#3000H       ; Load the service routine address for IRQ7
LD      SYM,#1EH         ; Enable fast interrupt processing
EI                      ; Enable interrupts
    
```

7

CLOCK CIRCUIT

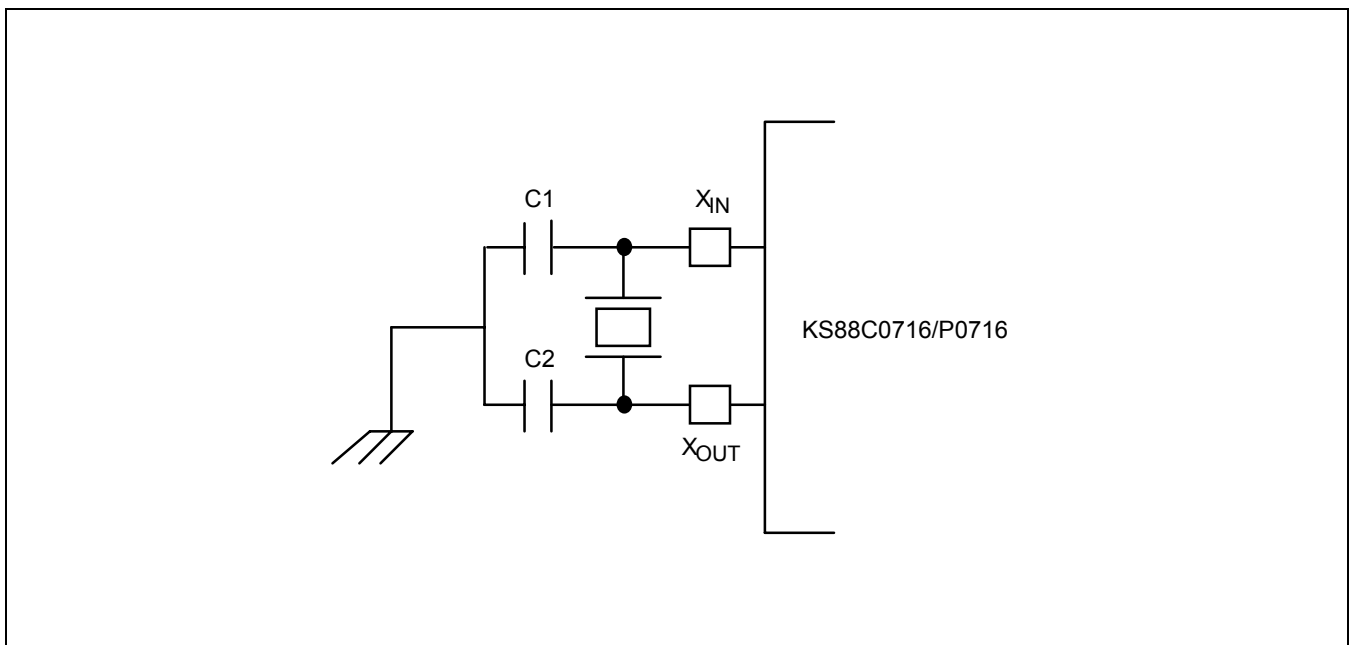
OVERVIEW

The clock frequency generated for the KS88C0716/P0716 by an external crystal can range from 1 MHz to 22.1184 MHz. The maximum CPU clock frequency is 12 MHz. The X_{IN} and X_{OUT} pins connect the external oscillator or clock source to the on-chip clock circuit.

SYSTEM CLOCK CIRCUIT

The system clock circuit has the following components:

- External crystal or ceramic resonator oscillation source (or an external clock source)
- Oscillator stop and wake-up functions
- Programmable frequency divider for the CPU clock (f_{OSC} divided by 1, 2, 8, or 16)
- System clock control register, CLKCON



**Figure 7-1. Main Oscillator Circuit
(External Crystal or Ceramic Resonator)**

CLOCK STATUS DURING POWER-DOWN MODES

The two power-down modes, Stop mode and Idle mode, affect the system clock as follows:

- In Stop mode, the main oscillator is halted. Stop mode is released, and the oscillator started, by a reset operation or an external interrupt (with RC delay noise filter).
- In Idle mode, the internal clock signal is gated to the CPU, but not to interrupt structure, timers and timer/counters. Idle mode is released by a reset or by an external or internal interrupt.

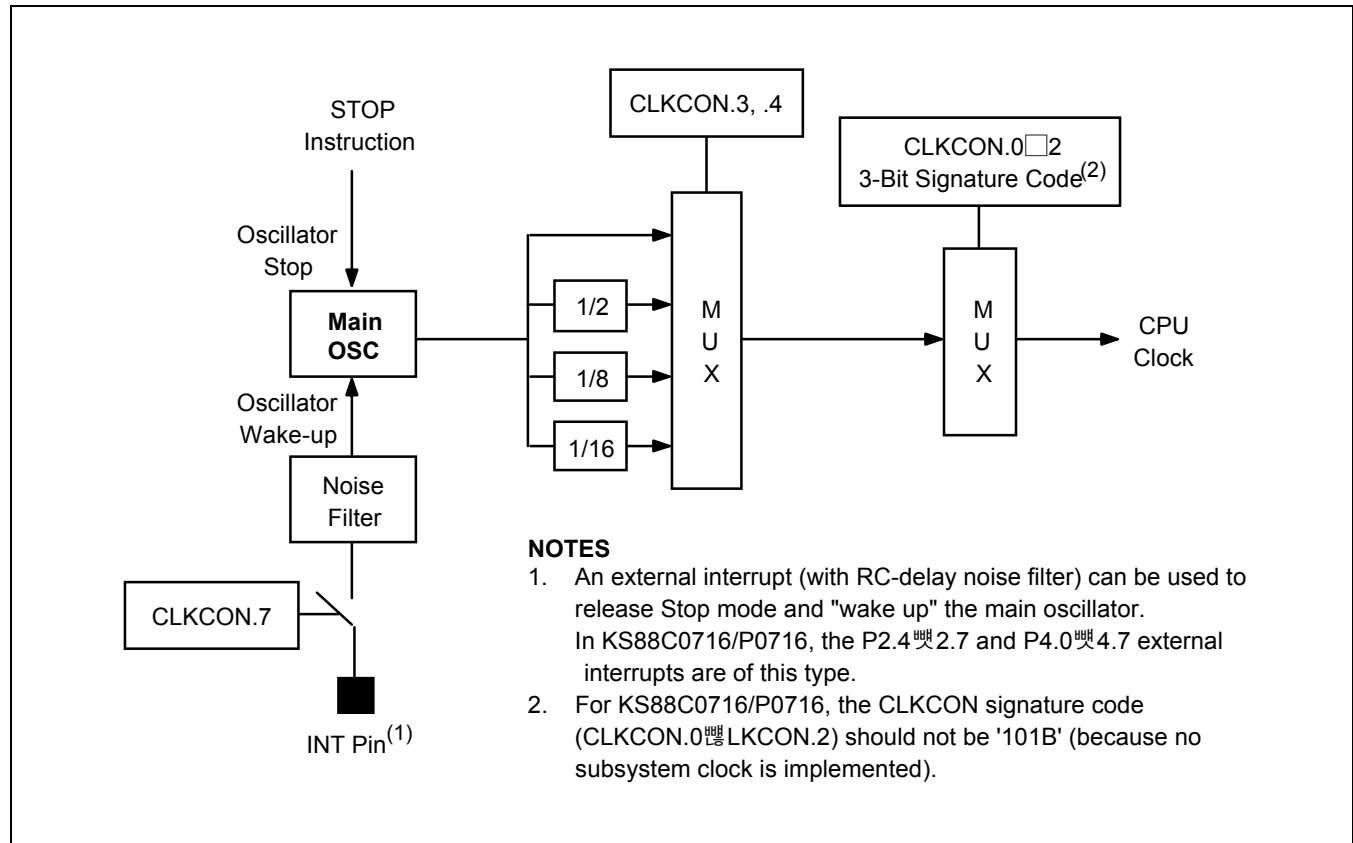


Figure 7-2. System Clock Circuit Diagram

SYSTEM CLOCK CONTROL REGISTER (CLKCON)

The system clock control register, CLKCON, is located in the bank 0 of set 1, address D4H. It is read/write addressable and has the following functions:

- Oscillator IRQ wake-up function enable/disable
- Main oscillator stop control
- Oscillator frequency divide-by value
- System clock signal selection

The CLKCON register controls whether or not an external interrupt can be used to trigger a power down mode release. (This is called the "IRQ wake-up" function.) The IRQ wake-up enable bit is CLKCON.7.

After a reset, the external interrupt oscillator wake-up function is enabled, the main oscillator is activated, and the $f_{OSC}/16$ (the slowest clock speed) is selected as the CPU clock. If necessary, you can then increase the CPU clock speed to f_{OSC} , $f_{OSC}/2$, or $f_{OSC}/8$.

NOTE

If you are using an oscillator with a frequency higher than 12 MHz, you cannot use a non-divided clock (f_{OSC}) as the CPU clock (see Figure 7-3).

For the KS88C0716/P0716 microcontrollers, the CLKCON.2–CLKCON.0 system clock signature code must be any value *other than* '101B'. (The '101B' setting is invalid because a subsystem clock is not implemented.) The reset value for the clock signature code is '000B' and should remain so during normal operation.

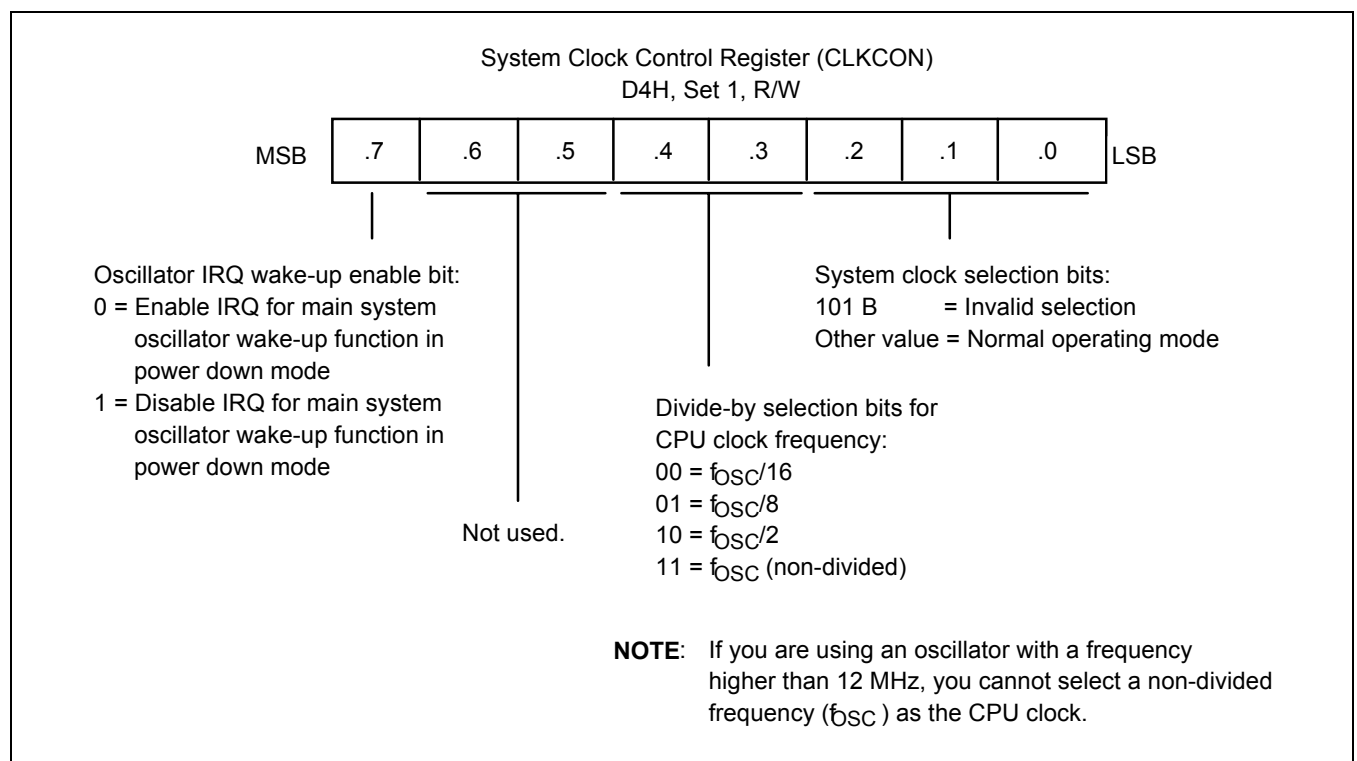


Figure 7-3. System Clock Control Register (CLKCON)

NOTES

8

RESET and POWER-DOWN

SYSTEM RESET

OVERVIEW

During a power-on reset, the voltage at V_{DD} goes to High level and the RESET pin is forced to Low level. The RESET signal is input through a Schmitt trigger circuit where it is then synchronized with the CPU clock. This procedure brings KS88C0716/P0716 into a known operating status.

To allow time for internal CPU clock oscillation to stabilize, the RESET pin must be held to Low level for a minimum time interval after the power supply comes within tolerance. The minimum required oscillation stabilization time for a reset operation is 1 millisecond.

Whenever a reset occurs during normal operation (that is, when both V_{DD} and RESET are High level), the RESET pin is forced Low and the reset operation starts. All system and peripheral control registers are then reset to their default hardware values (see Tables 8-1, 8-2, and 8-3).

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

- All interrupts are disabled.
- The watchdog function (basic timer) is enabled.
- Ports 0-5 are set to input mode and port6 is set output mode.
- Peripheral control and data registers are disabled and reset to their default hardware values.
- The program counter (PC) is loaded with the program reset address in the ROM, 0100H.
- When the programmed oscillation stabilization time interval has elapsed, the instruction stored in ROM location 0100H (and 0101H) is fetched and executed.

NORMAL MODE RESET OPERATION

In normal (masked ROM) mode, the EA pin is tied to V_{SS} . A reset enables access to the 16-Kbyte on-chip ROM. (The external interface is not automatically configured).

ROM-LESS MODE RESET OPERATION

To configure KS88C0716/P0716 as a ROM-less device, you must apply a constant 5 V current to the EA pin. Assuming the EA pin is held to high level (5 V) when a reset occurs, ROM-less mode is entered and the external interface is configured automatically.

NOTE

To program the duration of the oscillation stabilization interval, you make the appropriate settings to the basic timer control register, BTCON, *before* entering Stop mode. Also, if you do not want to use the basic timer watchdog function (which causes a system reset if a basic timer counter overflow occurs), you can disable it by writing '1010B' to the upper nibble of BTCON.

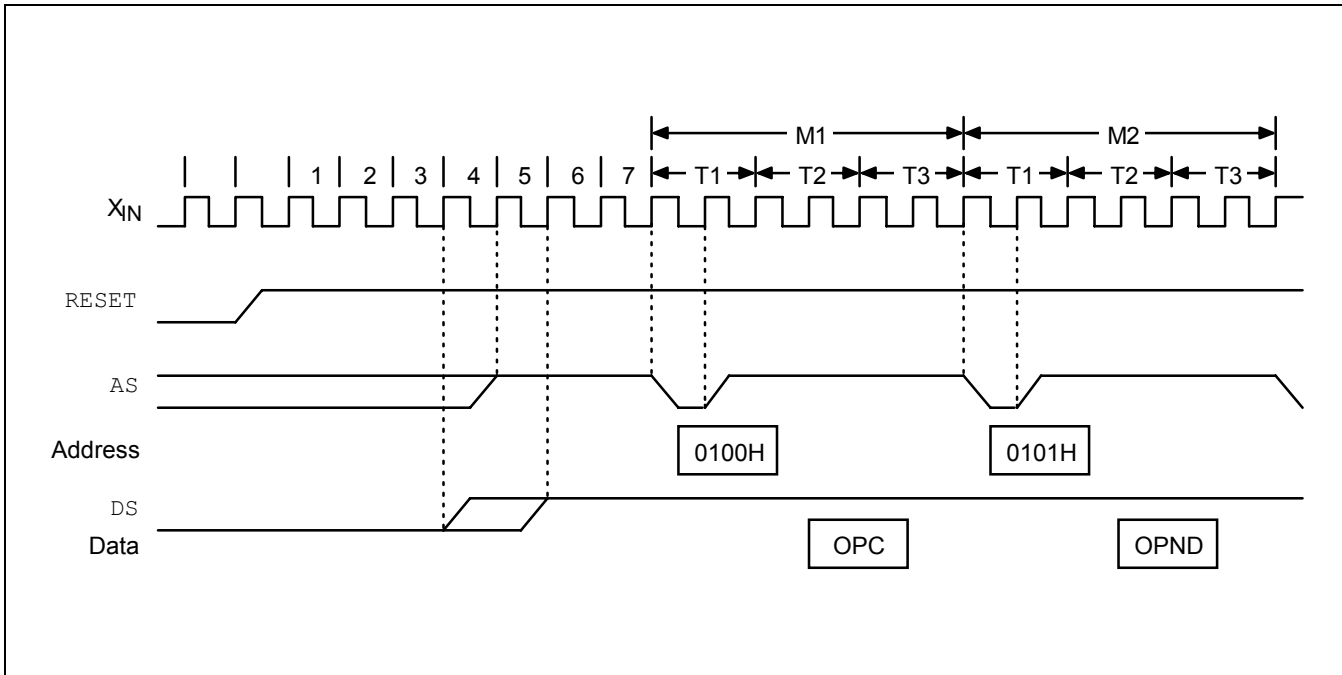


Figure 8-1. RESET Timing for External Interface (Masked ROM)

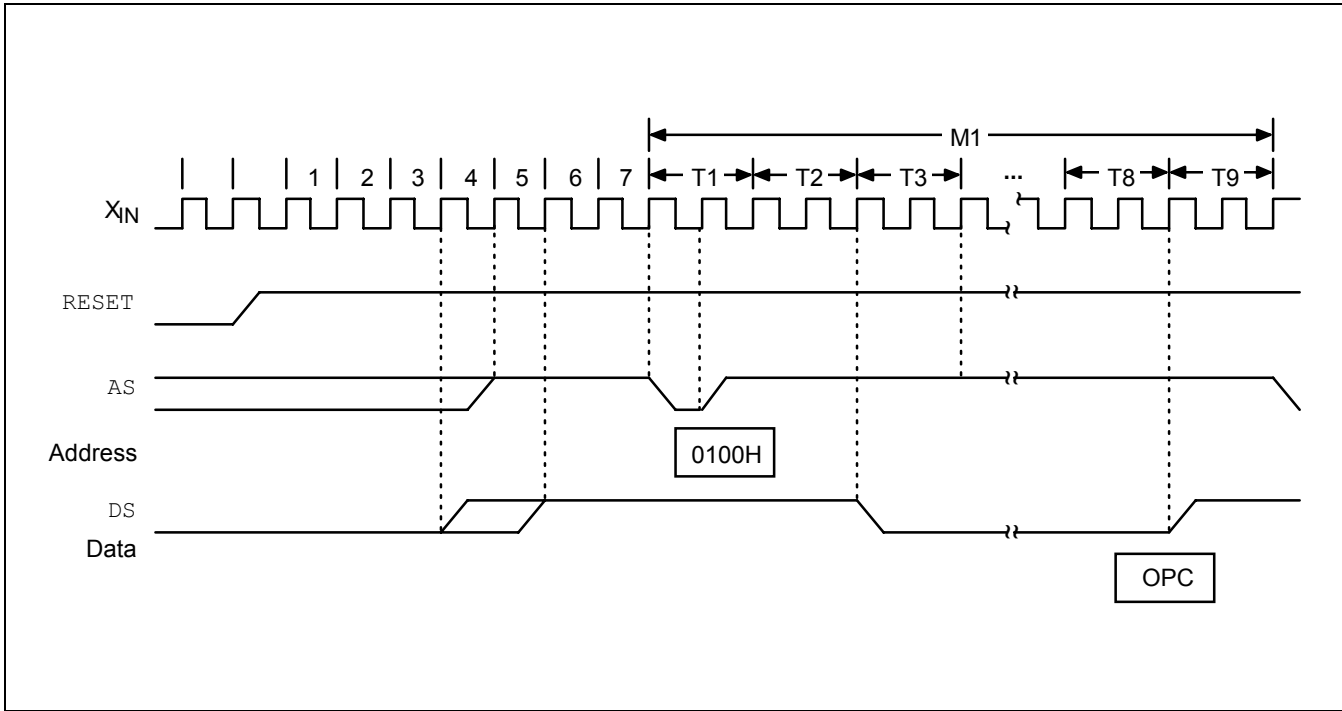


Figure 8-2. RESET Timing for External Interface (ROM-less Mode)

HARDWARE RESET VALUES

Tables 8-1, 8-2, and 8-3 list the reset values for CPU and system registers, peripheral control registers, and peripheral data registers following a reset operation. The following notation is used to represent reset values:

- A "1" or a "0" shows the reset bit value as logic one or logic zero, respectively.
- An 'x' means that the bit value is undefined after a reset.
- A dash (–) means that the bit is either not used or not mapped.

Table 8-1. KS88C0716/P0716 Set1 Register and Values after RESET (Masked ROM Mode)

Register Name	Mnemonic	Address		Bit Values after RESET (EA Pin is Low)							
		Dec	Hex	7	6	5	4	3	2	1	0
Timer C Counter Register (High Byte)	TCH	208	D0H	0	0	0	0	0	0	0	0
Timer C Counter Register (Low Byte)	TCL	209	D1H	0	0	0	0	0	0	0	0
Location D2H is not mapped.											
Basic Timer Control Register (watch dog)	BTCON	211	D3H	0	0	0	0	0	0	0	0
Clock Control Register	CLKCON	212	D4H	0	0	0	0	0	0	0	0
System Flags Register	FLAGS	213	D5H	x	x	x	x	x	x	0	0
Register Pointer 0	RP0	214	D6H	1	1	0	0	0	–	–	–
Register Pointer 1	RP1	215	D7H	1	1	0	0	1	–	–	–
Stack Pointer (High Byte)	SPH	216	D8H	x	x	x	x	x	x	x	x
Stack Pointer (Low Byte)	SPL	217	D9H	x	x	x	x	x	x	x	x
Instruction Pointer (High Byte)	IPH	218	DAH	x	x	x	x	x	x	x	x
Instruction Pointer (Low Byte)	IPL	219	DBH	x	x	x	x	x	x	x	x
Interrupt Request Register	IRQ	220	DCH	0	0	0	0	0	0	0	0
Interrupt Mask Register	IMR	221	DDH	x	x	x	x	x	x	x	x
System Mode Register	SYM	222	DEH	0	–	–	x	x	x	0	0
Register Page Pointer	PP	223	DFH	0	0	0	0	0	0	0	0

Table 8-2. KS88C0716/P0716 Set1, Bank0 Register Values after RESET (Masked ROM Mode)

Register Name	Mnemonic	Address		Bit Values after RESET (EA Pin is Low)							
		Dec	Hex	7	6	5	4	3	2	1	0
Port 0 Data Register	P0	224	E0H	0	0	0	0	0	0	0	0
Port 1 Data Register	P1	225	E1H	0	0	0	0	0	0	0	0
Port 2 Data Register	P2	226	E2H	0	0	0	0	0	0	0	0
Port 3 Data Register	P3	227	E3H	0	0	0	0	0	0	0	0
Port 4 Data Register	P4	228	E4H	0	0	0	0	0	0	0	0
Port 5 Data Register	P5	229	E5H	0	0	0	0	0	0	0	0
Port 6 Data Register	P6	230	E6H	1	1	1	1	1	1	1	1
Port4 Interrupt Pending Register	P4PND	231	E7H	0	0	0	0	0	0	0	0
Port 4 Interrupt Enable Register	P4INT	232	E8H	0	0	0	0	0	0	0	0
Serial Port Shift Register	SIO	233	E9H	x	x	x	x	x	x	x	x
Serial Port Control Register	SIOCON	234	EAH	0	0	0	0	0	0	0	0
Serial Port Interrupt Pending Register	SIOPND	235	EBH	–	–	–	–	–	–	0	0
Timer A Data Register	TADATA	236	ECH	0	0	0	0	0	0	0	0
Timer B Data Register	TBDATA	237	EDH	0	0	0	0	0	0	0	0
Timer Module 0 Control Register	T0CON	238	EEH	0	0	0	0	0	0	0	0
Timer B Interrupt Register	TBINT	239	EFH	–	–	–	–	–	–	–	0
Port 0 Control Register	P0CON	240	F0H	0	0	0	0	0	0	0	0
Port 1 Control Register	P1CON	241	F1H	0	0	0	0	0	0	0	0
Port 2 Control Register (High Byte)	P2CONH	242	F2H	0	0	0	0	0	0	0	0
Port 2 Control Register (Low Byte)	P2CONL	243	F3H	0	0	0	0	0	0	0	0
Port 3 Control Register (High Byte)	P3CONH	244	F4H	0	0	0	0	0	0	0	0
Port 3 Control Register (Low Byte)	P3CONL	245	F5H	0	0	0	0	0	0	0	0
Port 4 Control Register (High Byte)	P4CONH	246	F6H	0	0	0	0	0	0	0	0
Port 4 Control Register (Low Byte)	P4CONL	247	F7H	0	0	0	0	0	0	0	0
Timer D Counter Register (High Byte)	TDH	248	F8H	0	0	0	0	0	0	0	0
Timer D Counter Register (Low Byte)	TDL	249	F9H	0	0	0	0	0	0	0	0
Timer Module 1 Control Register	T1CON	250	FAH	0	–	0	0	0	0	0	0
Timer Module 1 Mode Register	T1MOD	251	FBH	0	0	0	0	0	0	0	0
Basic Timer counter	BTCNT	253	FDH	0	0	0	0	0	0	0	0
External Memory Timing Register	EMT	254	FEH	0	–	–	–	–	–	0	–
Interrupt Priority Register	IPR	255	FFH	x	x	x	x	x	x	x	x

Table 8-3. KS88C0716/P0716 Set1, Bank1 Register Values after RESET (Masked ROM Mode)

Register Name	Mnemonic	Address		Bit Values after RESET (EA Pin is High)							
		Dec	Hex	7	6	5	4	3	2	1	0
Port 5 Control Register	P5CON	248	F8H	0	0	0	0	0	0	0	0
Port 6 Control Register	P6CON	249	F9H	0	0	0	0	0	0	0	0

Table 8-4. External Interface Control Register Values after RESET (ROM-Less Mode)

Register Name	Mnemonic	Address		Bit Values after RESET (EA Pin is High)							
		Dec	Hex	7	6	5	4	3	2	1	0
Port 2 Control Register (Low Byte)	P2CONL	243	F3H	1	1	1	1	1	1	1	1

NOTE: The RESET values for P2CONL are valid only for the KS88P0716 ROM-less operating mode.

POWER-DOWN MODES

STOP MODE

Stop mode is invoked by the instruction STOP (opcode 7FH). In Stop mode, the operation of the CPU and all peripherals is halted. That is, the on-chip main oscillator stops and the supply current is reduced to less than 0.1 μ A. All system functions stop when the clock "freezes," but data stored in the internal register file is retained. Stop mode can be released in one of two ways: by a reset or by an external interrupt (with RC delay).

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.

Using RESET to Release Stop Mode

Stop mode is released when the RESET signal is released and returns to high level: all system and peripheral control registers are reset to their default hardware values and the contents of all data registers are retained. A reset operation automatically selects a slow clock (1/16) because CLKCON.3 and CLKCON.4 are cleared to '00B'. After the programmed oscillation stabilization interval has elapsed, the CPU starts the system initialization routine by fetching the program instruction stored in ROM location 0100H (and 0101H).

Using an External Interrupt to Release Stop Mode

Only external interrupts with an RC-delay noise filter circuit can be used to release Stop mode. Which interrupt you can use to release Stop mode in a given situation depends on the microcontroller's current internal operating mode. The external interrupts in the KS88C0716/P0716 interrupt structure that can be used to release Stop mode are:

- External interrupts P2.4–P2.7 (INT0–INT3), and P4.0–P4.7 (INT4–INT11)

Please note the following conditions for Stop mode release:

- If you release Stop mode using an external interrupt, the current values in system and peripheral control registers are unchanged.
- If you use an external interrupt for Stop mode release, you can also program the duration of the oscillation stabilization interval. To do this, you must make the appropriate control and clock settings *before* entering Stop mode.
- When the Stop mode is released by external interrupt, the CLKCON.4 and CLKCON.3 bit-pair setting remains unchanged and the currently selected clock value is used.
- The external interrupt is serviced when the Stop mode release occurs. Following the IRET from the service routine, the instruction immediately following the one that initiated Stop mode is executed.

IDLE MODE

Idle mode is invoked by the instruction IDLE (opcode 6FH). In Idle mode, CPU operations are halted while some peripherals remain active. During Idle mode, the internal clock signal is gated away from the CPU, but all peripherals timers remain active. Port pins retain the mode (input or output) they had at the time Idle mode was entered.

There are two ways to release Idle mode:

1. Execute a reset. All system and peripheral control registers are reset to their default values and the contents of all data registers are retained. The reset automatically selects a *slow clock (1/16)* because CLKCON.4 and CLKCON.3 are cleared to '00B'. If interrupts are masked, a reset is the only way to release Idle mode.
2. Activate any enabled interrupt, causing Idle mode to be released. When you use an interrupt to release Idle mode, the CLKCON.4 and CLKCON.3 register values remain unchanged, and the *currently selected clock* value is used. The interrupt is then serviced. When the return-from-interrupt (IRET) occurs, the instruction immediately following the one that initiated Idle mode is executed.

PROGRAMMING TIP — Sample KS88C0716 Initialization Routine

The following sample program suggests initialization settings for the KS88C0716 address space, interrupt vectors, and peripheral functions:

```
;      << Chip Directive >>

      CHIP      C: \ DEF \ 88C0716.DEF

;      << User Equation Definition >>

      INCLUDE   C: \ EQU.TBL
;      Reference label and vector area: 000H–00FFH
      ORG       0000H

;      << Reset Vector >>

      ORG       0100H
      JP        t,INITIAL
;      0023H–00BBH: Reserved

;      << Interrupt Vector Addresses >>

      ORG       00BEH
      VECTOR    TA_int           ; IRQ0
;      00C0H–00D7H: Reserved

      ORG       00D8H
      VECTOR    EXT40_int        ; IRQ4
      VECTOR    EXT41_int        ; IRQ5
      VECTOR    EXT42_int        ; IRQ6
      VECTOR    EXT43_int        ; IRQ6
      VECTOR    EXT44_int        ; IRQ7
      VECTOR    EXT45_int        ; IRQ7
      VECTOR    EXT46_int        ; IRQ7
      VECTOR    EXT47_int        ; IRQ7

;      ORG       00E8H
      VECTOR    EXT24_int        ; IRQ3
      VECTOR    EXT25_int        ; IRQ3
      VECTOR    EXT26_int        ; IRQ3
      VECTOR    EXT27_int        ; IRQ3
      VECTOR    SIOINT_R         ; IRQ2
      VECTOR    SIOINT_T         ; IRQ2
      VECTOR    TC_int           ; IRQ1
      VECTOR    TD_int           ; IRQ1
;      00F8H–00FDH: Reserved
```

PROGRAMMING TIP — Sample KS88C0716 Initialization Routine (Continued)

```

;          << System and Peripheral Initialization >>

INITIAL:   ORG          0200H
           DI

;          <System register setting>

           LD          SYM,#00000000B      ; Fast, global interrupt disable
           LD          EMT,#00000000B      ; 'No wait' and internal stack area select
           LD          SPH,#00H            ; Stack pointer (high byte) to zero
           LD          SPL,#00H            ; Stack pointer (low byte) to zero
           LD          CLKCON,#10H         ; fOSC/2 is selected for CPU clock

;          <Interrupt settings>

           LD          IPR,#16H            ; Interrupt priorities
                                           ; IRQ3 > 4 > 2 > 0 > 1 > 5 > 6 > 7
           LD          IMR,#00001101B      ; IRQ levels 0, 2, and 3 enable
                                           ; Level 0 = timer A interrupt
                                           ; Level 2 = SIO, TC, and TD interrupts
                                           ; Level 3 = External interrupts P2.4–P2.7

INI_PERI_SET:

;          <Port 0 setting>

           LD          P0CON,#11H          ; Output, push-pull

;          <Port 1 setting>

           LD          P1CON,#11H          ; Output, push-pull

;          <Port 2 setting>

           LD          P2CONL,#01100101B   ; P2.0 input mode
                                           ; P2.1 input mode
                                           ; P2.2 output mode, push-pull
                                           ; P2.3 input mode
           LD          P2CONH,#01101001B   ; P2.4 input, falling edge interrupt,
                                           ; This pin used for EXT_int
                                           ; P2.5, P2.6 output mode
                                           ; P2.7 input, falling edge interrupt,
                                           ; This pin used for IR_int

```

PROGRAMMING TIP — Sample KS88C0716 Initialization Routine (Continued)

```

;      <Port 3 setting>

LD      P3CONL,#01010101B    ; P3.0–P3.5 input mode
LD      P3CONH,#00110000B    ; RxD (P3.7) input is selected
                                   ; TxD (P3.6) output is selected

;      <Port 4 setting>

LD      P4CONL,#00H          ; P4.0–P4.3 input mode
LD      P4CONH,#0FFH         ; P4.4–P4.7 output, push-pull
LD      P4INT,#00H           ; All P4 interrupts disabled

;      <Port 5, 6 setting>

SB1
LD      P5CON,#11H           ; Output, push-pull
LD      P6CON,#11H           ; Output, open-drain
SB0

;      <Open-drain type>

;      <Timer A>

LD      TADATA,#8H           ; CPU clock divided by 9
LD      T0CON,#00000100B     ; If CPU clock = 11.0592 MHz/2
                                   ; CPU clock/1024/9 → 1.66 ms
                                   ; Interval mode
                                   ; Interrupt enable

;      <Timer B>                ; Disabled

;      <Timer C>                ; 16-bit free running timer, no interrupt

;      <Timer D>                ; SIO baud rate generator
                                   ; 9600 BPS if CPU clock = 11.0592 MHz/2
LD      TDL,#0FDH            ; Start value
LD      TDH,#0FDH            ; Auto-reload value
                                   ; FD = 9600 bps
                                   ; FA = 4800 bps
LD      T1CON,#00000011B     ; Normal baud rate
                                   ; Timer C, D pending bit clear
                                   ; Timer C, D interrupt disable
                                   ; Timer C, D run enable
LD      T1MOD,#00100001B     ; Timer C, D gate function disable
                                   ; Timer C, D clock = CPU clock/6
                                   ; Timer C → 16-bit timer mode
                                   ; Timer D → Auto-reload mode

```

PROGRAMMING TIP — Sample KS88C0716 Initialization Routine (Continued)

```

;      <SIO(UART)>
LD      SIOCON,#01010010B      ; Mode 1, 8-bit UART, variable baud rate
;                                ; Multi-bit clear
;                                ; Receive enable
;                                ; Tx 9th bit = "0"
;                                ; Rx 9th bit = "0"
;                                ; Receive interrupt enable
;                                ; Transmit interrupt disable
;                                ; Pending bit clear

LD      SIOPND,#00H

;      << Register Initialization >>

SRP      #0C0H

;      <Clear all data registers 00H–0FFH>

RAMCLR:  LD      R0,#0FFH
          CLR     @R0
          DJNZ    R0,RAMCLR

;      <Initialize other registers>

          .
          .
          .
EI                                ; Must be executed in this position
                                ; before external interrupt is executed

;      << Main Loop >>

MAIN:    NOP                                ; Start main loop
LD      BTCON,#03H                      ; Enable watchdog timer, clear BTCNT, and
;                                ; basic timer clock input divider.
;                                ; Basic timer overflow time = 94.81 ms
;                                ; (256 × 4096/11.0592M)

          .
          .
          .
CALL     KEY_SCAN
          .
          .
          .
CALL     LED_DISPLAY
          .
          .
          .
CALL     JOB
          .
          .
          .
JR      t,MAIN

```

 PROGRAMMING TIP — Sample KS88C0716 Initialization Routine (Continued)

```

;          <Subroutine 1>
KEY_SCAN:
    NOP
    .
    .
    .
    RET

;          <Subroutine 2>
LED_DISPLAY:
    NOP
    .
    .
    .
    RET

;          <Subroutine 3>
JOB:      NOP
    .
    .
    .
    RET

;          << Interrupt Service Routine >>
TA_int:   PUSH      RP0
          PUSH      RP1
          SRP        #TA_REG          ; Example: TA_REG = 00H
          .
          .
          .
          AND        T0CON,#11111101B ; Clear pending bit
          POP        RP1
          POP        RP0
          IRET

```

PROGRAMMING TIP — Sample KS88C0716 Initialization Routine (Concluded)

```

;          << Other Interrupt Vectors >>

EXT40_int:                ; IRQ4
EXT41_int:                ; IRQ5
EXT42_int:                ; IRQ6
EXT43_int:                ; IRQ6
EXT44_int:                ; IRQ7
EXT45_int:                ; IRQ7
EXT46_int:                ; IRQ7
EXT47_int:  AND            P4PND,#00H    ; IRQ7
            IRET

SIOint_R:                ; IRQ2
SIOint_T:  LD              SIOPND,#00H   ; IRQ2
            IRET

TC_int:                ; IRQ1
TD_int:  LD                T1CON,#00000011B ; IRQ1
            IRET

EXT24_int:                ; IRQ3
EXT25_int:                ; IRQ3
EXT26_int:                ; IRQ3
EXT27_int:                ; IRQ3
            IRET

            END

```

NOTES

9

I/O PORTS

OVERVIEW

The KS88C0716/P0716 microcontrollers have seven I/O ports with a total of 56 pins. Each port can be flexibly configured to meet application design requirements. The CPU accesses ports by directly writing or reading port registers. No special I/O instructions are required. Table 1-2 gives you an overview of port functions:

KS88C0716 has 48 pins that are dedicated to I/O and 8 pins that are used for open-drain output. Each port can be flexibly configured to meet application design requirements.

With the exception of port 6, which is used for output mode, all ports can be configured to input or output mode. Ports 0, 1, and 2 can alternatively be configured as external interface lines.

The CPU accesses ports by directly writing or reading port register addresses. No special I/O instructions are required.

PORT DATA REGISTERS

Data registers for ports 0–6 have the structure shown in Figure 9-1:

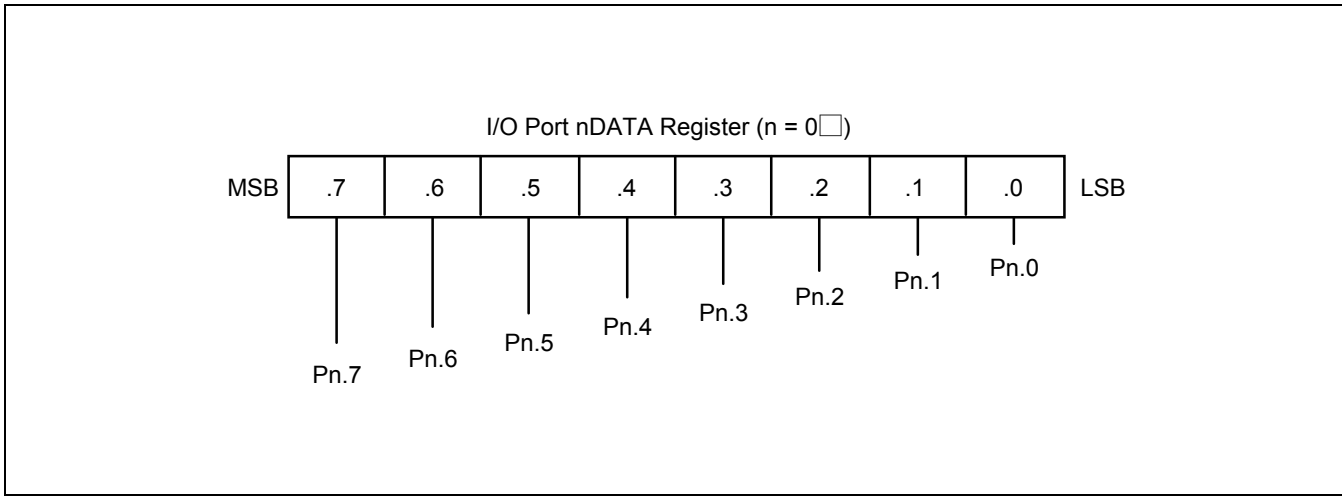
PORT 0

Port 0 is accessed directly by writing or reading the port 0 data register, P0 (E0H, set 1).

In normal operating mode, a reset clears the port 0 control register, P0CON, to '00H', configuring all port 0 pins as inputs.

In ROM-less operating mode($EA=V_{DD}$), a reset *automatically* configures port 0 (P0.0–P0.7) as address lines A8–A15 of the external peripheral interface.

You cannot access the P0 data register the external interface is configured: Write operations have no effect, and reads only return the state of the external pin.



PORT 1

Port 1 is accessed directly by writing or reading the port 1 data register, P1 (E1H) in set 1. You cannot access the P1 register when port 1 is configured for the external interface. (Write operations have no effect and reads simply return the state of the external pin.)

In normal operating mode (when the EA pin is low level) a reset clears all P1CON values to zero (00H). In ROM-less operating mode (when the EA pin is high level), a reset automatically configures port 1 as address/data lines AD0–AD7. Physically, P1CON is reset to its normal initialization value, 00H. Logically, however, the reset operation sets port 1 to external interface mode.

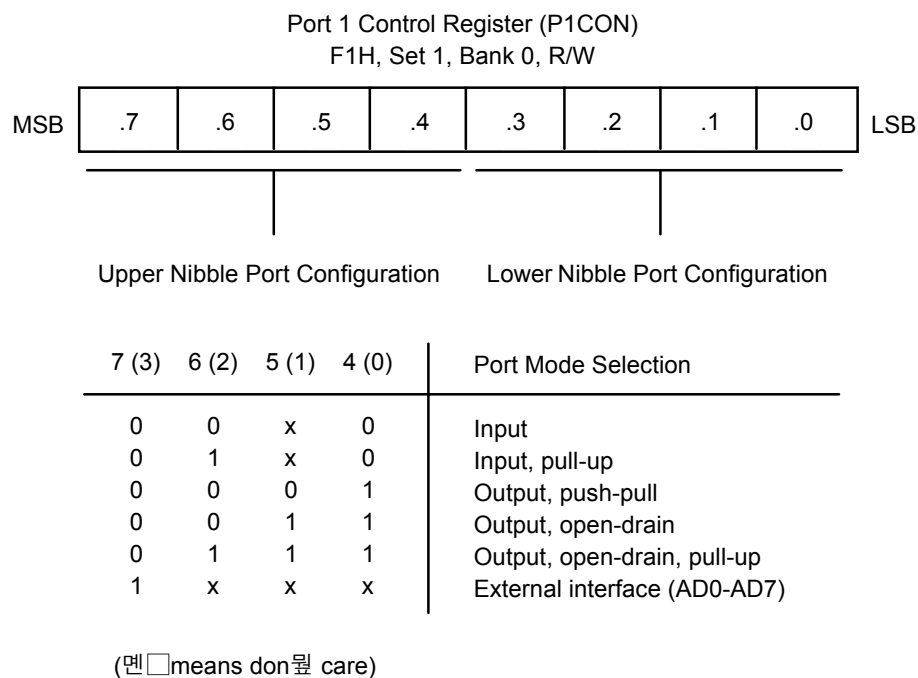


Figure 9-3. Port 1 Control Register (P1CON)

PORT 2

Port 2 is an 8-bit I/O port with individually configurable pins. It is accessed directly by writing or reading the port 2 data register, P2 (E2H, set 1). You can use port 2 for general I/O, or for the following alternative functions:

- Low-byte pins (P2.0–P2.3) can be configured as multiplexed lines for external interface bus control signals DM, R/W, DS, and AS.
- High-byte pins (P2.4–P2.7) can be configured as internal interrupt inputs with edge detection. P2.4 can also serve as the input for an externally generated WAIT signal.

Port 2 Operation in Normal and ROM-Less Modes

In normal mode $EA=0V$, a reset clears P2CONL and P2CONH register settings to '00H'. This configures all port 2 pins to input mode. In ROM-less operating mode $EA = V_{DD}$, a reset operation sets the P2CONL and P2CONH registers as follows:

- P2CONH (P2.4–P2.7) is cleared to '00H', configuring the high-byte pins to input mode.
- P2CONL (P2.0–P2.3) is *automatically* set to 'FFH', configuring the low-byte pins as signal lines for the external interface.

If you are using ROM-less mode, control of the P2CONL register is automatically passed to the external interface control logic after the reset operation.

NOTE

Any modification of P2CONL register values during ROM-less operation $EA=V_{DD}$ may disable the external interface.

Port 2 High-Byte Control Register (P2CONH)

In addition to its main port 2 configuration functions, you can use bit-pair 1/0 in the P2CONH register to configure the WAIT signal input for the external memory interface at P2.4. The wait function lets an external device "stretch" the data strobe (DS) for external peripheral accesses to compensate for timing differences.

To configure P2.4 as an input for the WAIT signal, you must first clear P2CONH bits 1 and 0. Then, to enable the wait function, set bit 7 in the external memory timing register, EMT, to "1".

Port 2 does not have a separate interrupt enable control register. Instead, when you configure a port 2 pin as an external interrupt input, the corresponding interrupt is enabled automatically.

Each external interrupt bit has an edge-triggered "interrupt pending" flip-flop bit for detecting external interrupt requests. These pending bits are cleared automatically by hardware after the CPU acknowledges the interrupt.

Port 2 Low-Byte Control Register (P2CONL)

The low-byte port 2 pins P2.0–P2.3 can be individually configured as inputs or as push-pull outputs. You can alternately configure these pins as outputs for the bus control signals of the external peripheral interface.

To enable the special control signal function for a port 2 pin, you set the appropriate bit-pair values in P2CONL to '11B'.

When you enable the external interface's tri-state function by setting SYM.7 to "1", the signals at the P2.0–P2.3 pins are automatically set to high impedance (that is, the signal levels "float").

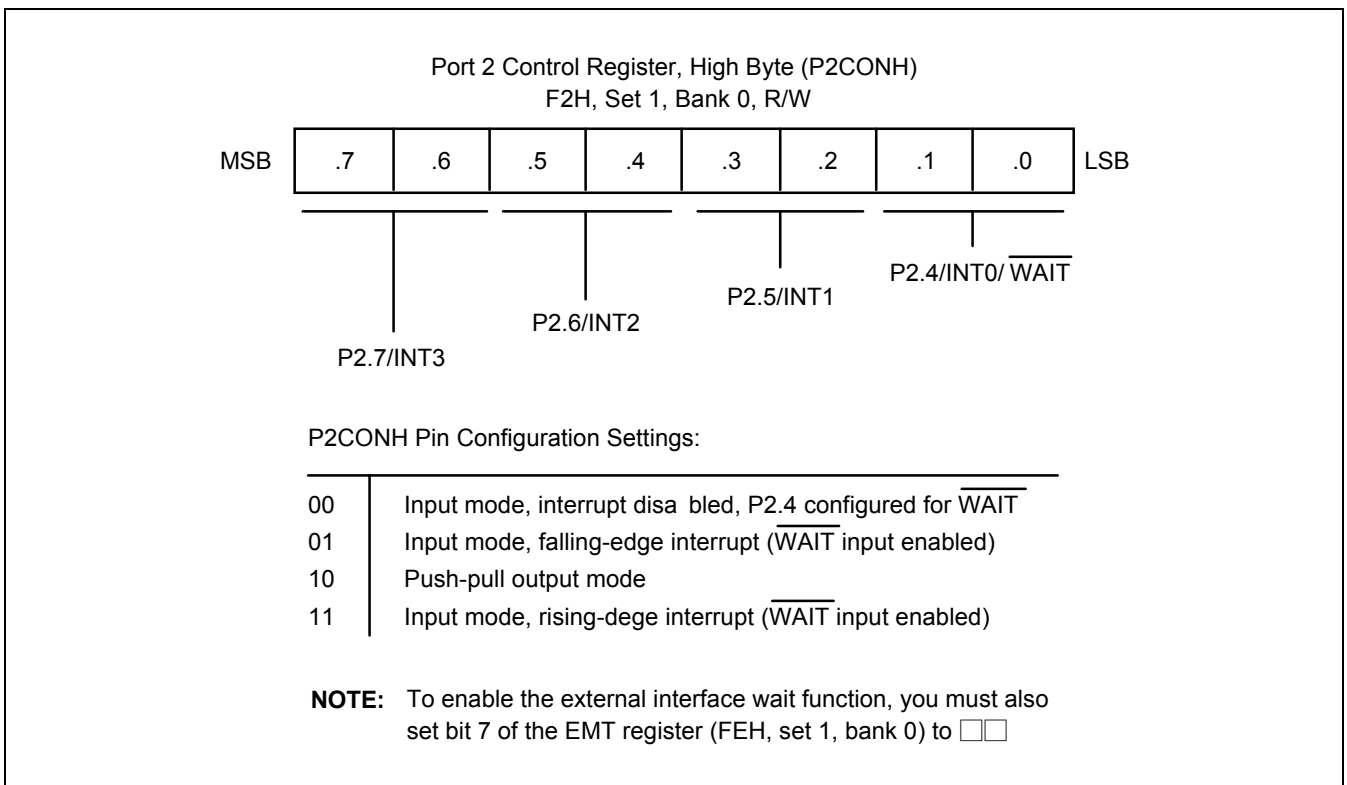


Figure 9-4. Port 2 High-Byte Control Register (P2CONH)

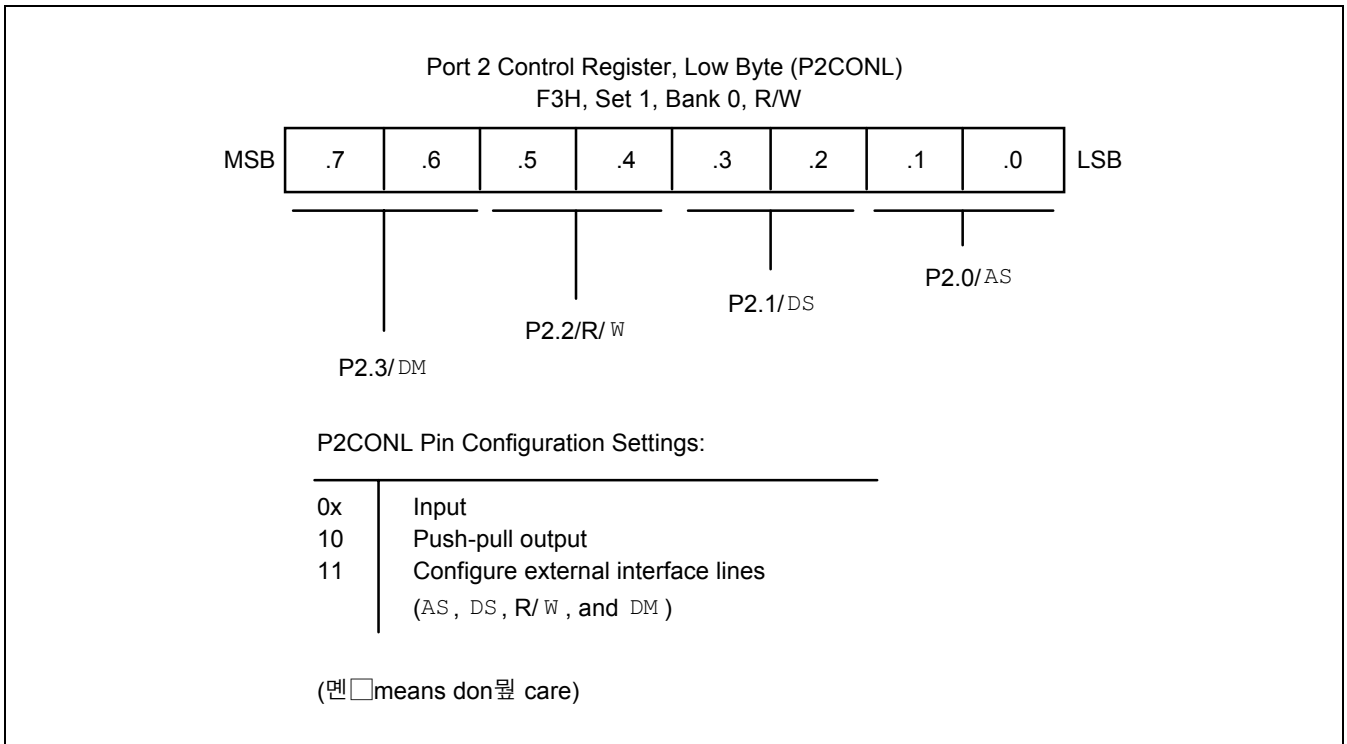


Figure 9-5. Port 2 Low-Byte Control Register (P2CONL)

PORT 3

Port 3 is an 8-bit I/O port with individually configurable pins. Port 3 is accessed directly by writing or reading the port 3 data register, P3 (E3H, set 1).

You can use port 3 pins for general I/O, or you can configure the following alternative functions:

- In input mode, pins P3.0 and P3.1 can be used as clock inputs to timer/counters C and D. The corresponding share pin names are TCCK and TDCK, respectively.
- In output mode, pins P3.4 and P3.5 can be used as timer A and B outputs, respectively. The corresponding share pin names are TA and TB.
- When configured to output mode, pin P3.6/TxD can be used for the serial data transmit interrupt (TI).
- When configured to input mode, pin P3.7/RxD can be used for the data receive interrupt (RI).

Any alternative peripheral I/O function that is programmed using the port 3 control registers must also be enabled in the associated peripheral module.

Please note that whenever you use port 3 pins for functions other than general I/O, you must still program the port 3 control registers to specify which bits are inputs and which are outputs.

Port 3 Control Registers

Port 3 has two 8-bit control registers: P3CONH for the high-byte pins, P3.4–P3.7, and P3CONL for the low-byte pins, P3.0–P3.3. A reset operation clears the P3CONH and P3CONL registers to '00H'.

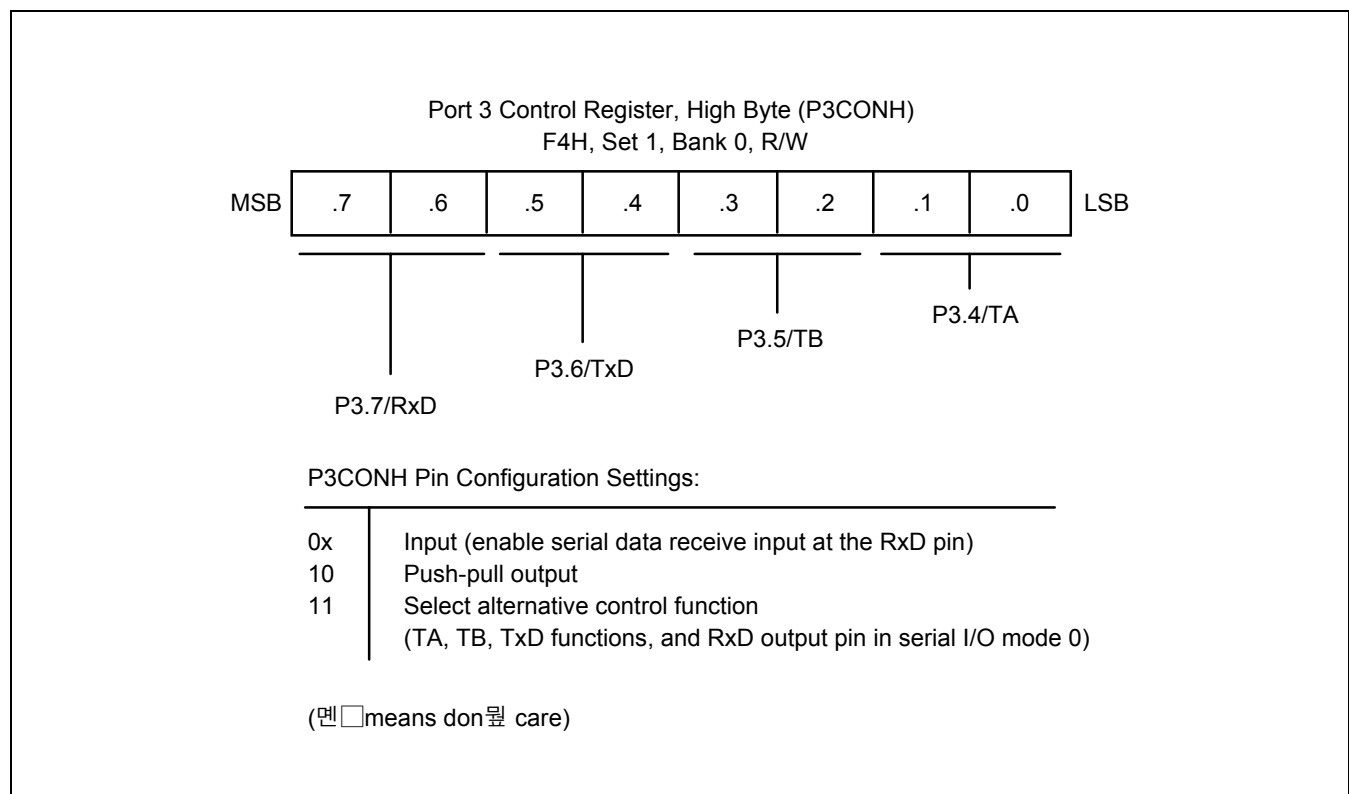


Figure 9-6. Port 3 High-Byte Control Register (P3CONH)

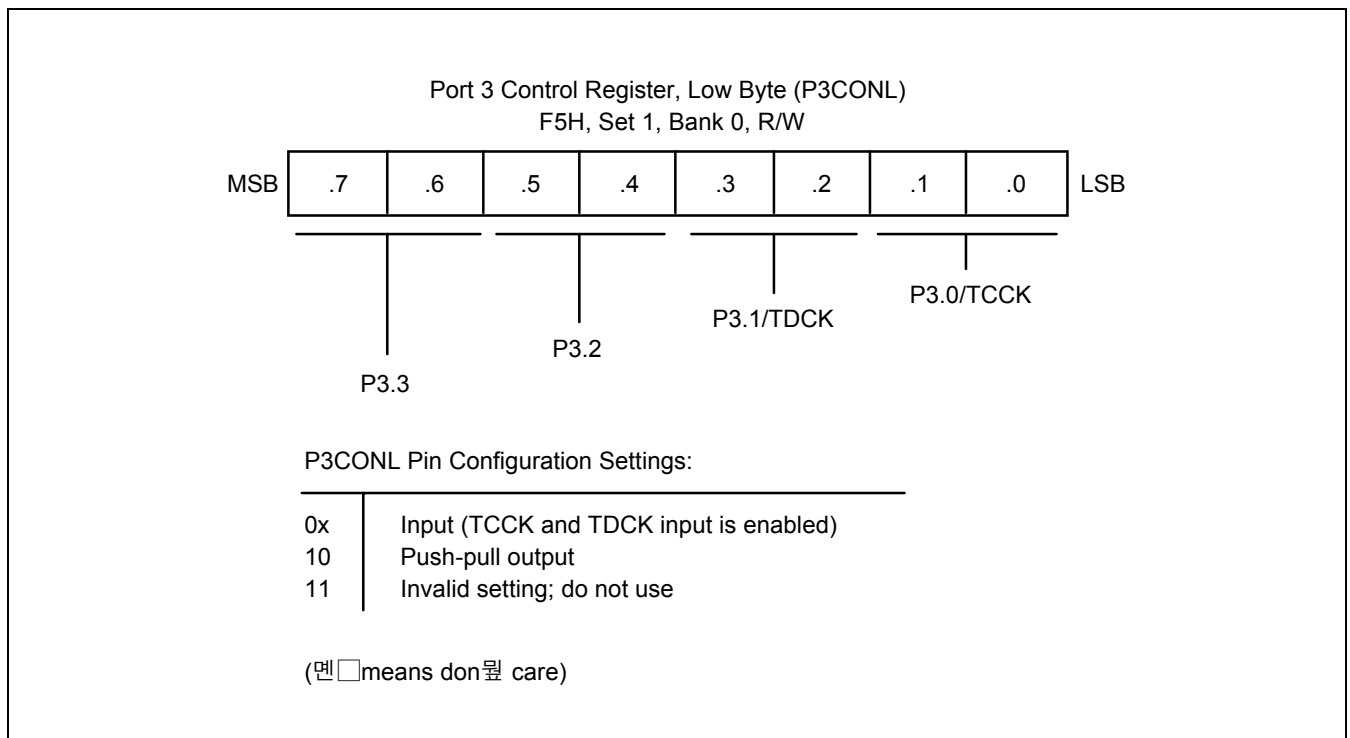


Figure 9-7. Port 3 Low-Byte Control Register (P3CONL)

PORT 4

Port 4 can serve as a general-purpose 8-bit I/O port or its pins can be configured individually as external interrupt inputs. In addition, pins P4.0 and P4.1 can alternatively be used as gate inputs for timers C and D. All inputs are Schmitt-triggered. Port 4 is accessed directly by writing or reading the port 4 data register, P4 (E4H, set 1).

Port 4 Control Registers (P4CONL, P4CONH)

The direction of each port pin is configured by bit-pair settings in two control registers: P4CONL (low byte, F7H) and P4CONH (high byte, F6H).

When output mode is selected, a push-pull circuit is automatically configured. In input mode, three selections are available: rising-edge interrupts, falling-edge interrupts, and falling-edge interrupts with pull-up resistor assigned.

Port 4 Interrupt Enable and Pending Registers (P4INT, P4PND)

To process external interrupts at the port 4 pins, two additional control registers are provided: the port 4 interrupt enable register P4INT (E8H) and the port 4 interrupt pending register P4PND (E7H).

The port 4 interrupt pending register P4PND lets you check for interrupt pending conditions and clear the pending condition when the interrupt request has been serviced. Incoming interrupt requests are detected by polling the P4PND bit values.

When the interrupt enable bit of any port 4 pin is "1", a rising or falling signal edge at that pin will generate an interrupt request. The corresponding P4PND bit is then automatically set to "1" and the IRQ level goes low to signal the CPU that an interrupt request is waiting.

When the CPU acknowledges the interrupt request, application software must clear the pending condition by writing a "0" to the corresponding P4PND bit.

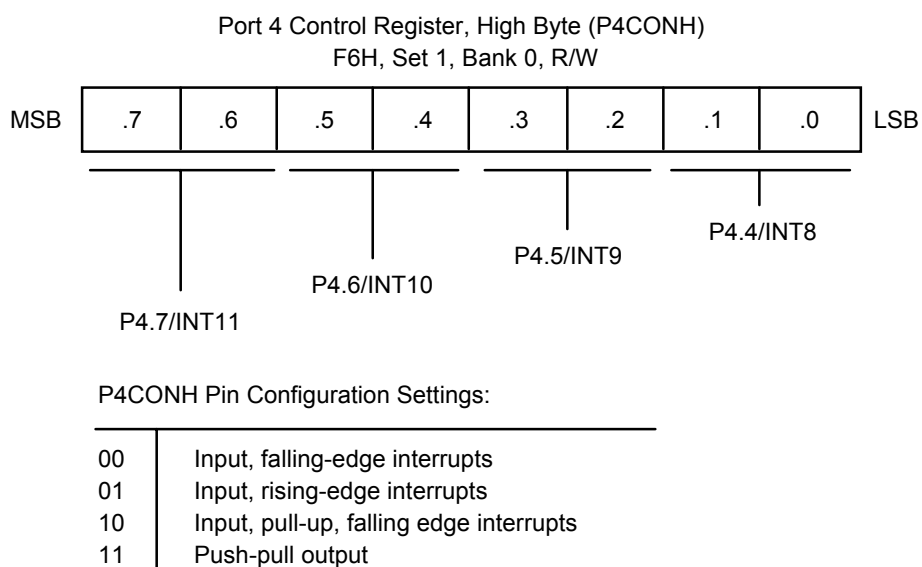


Figure 9-8. Port 4 High-Byte Control Register (P4CONH)

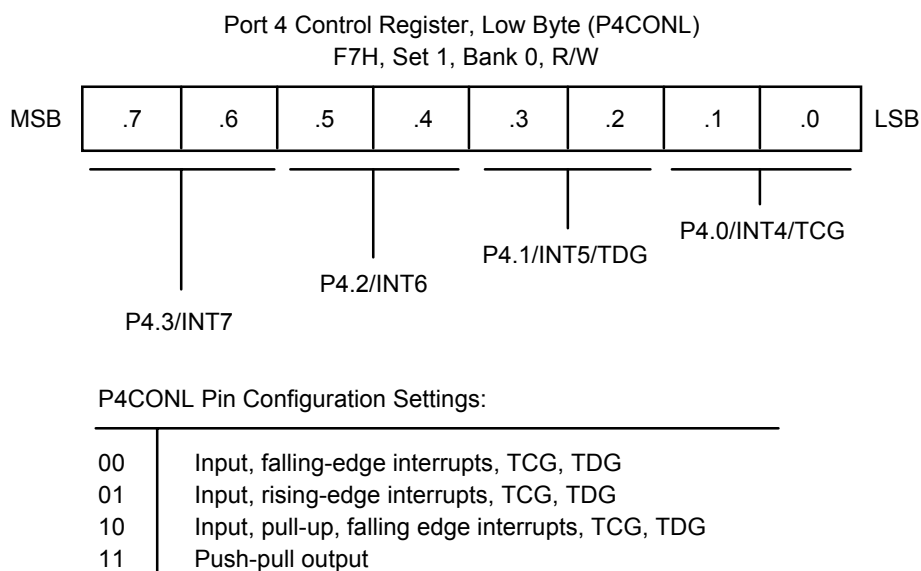


Figure 9-9. Port 4 Low-Byte Control Register (P4CONL)

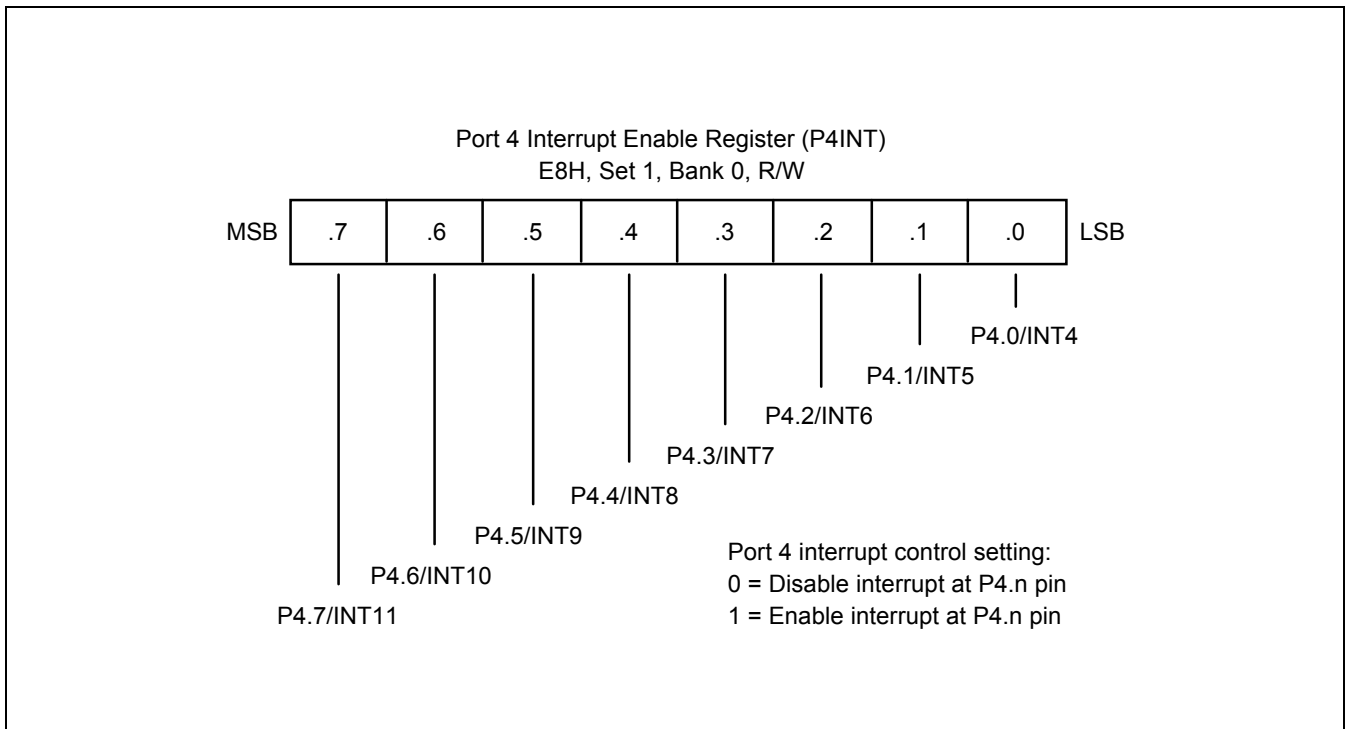


Figure 9-10. Port 4 Interrupt Enable Register(P4INT)

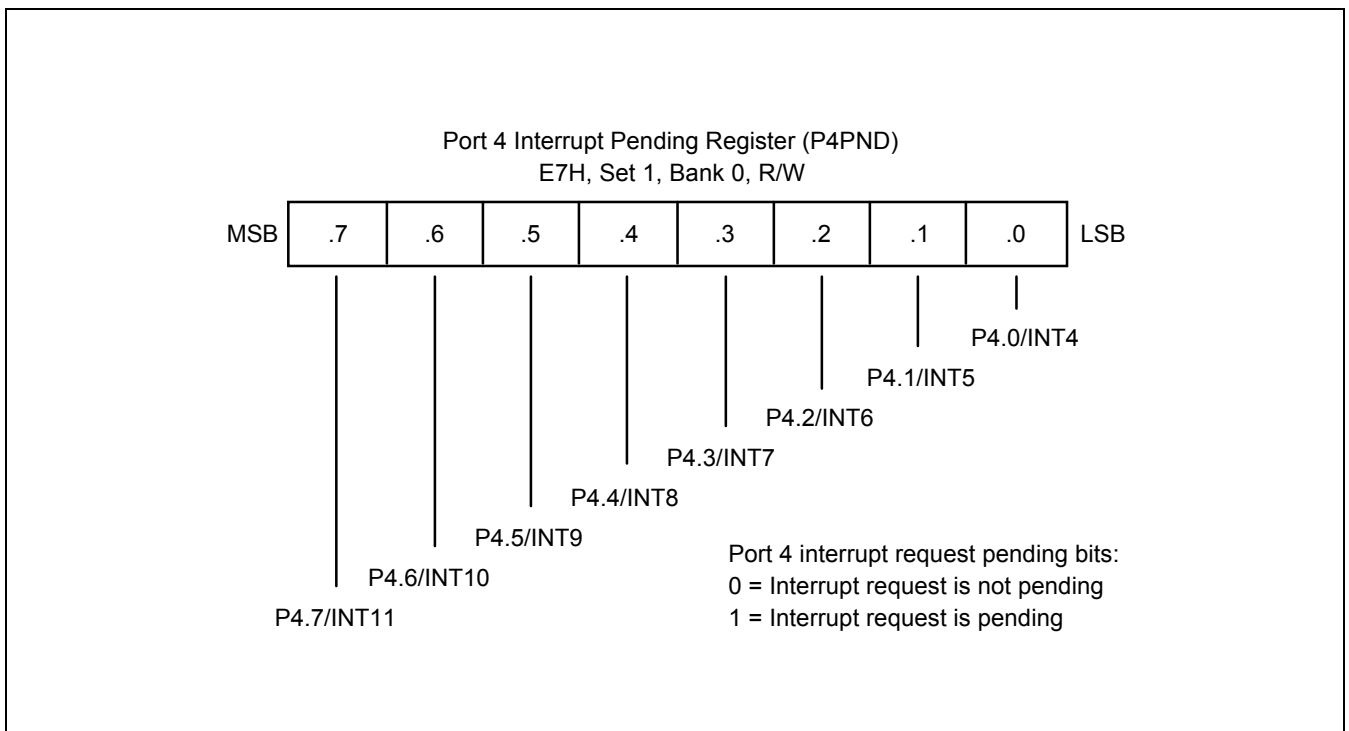


Figure 9-11. Port 4 Interrupt Pending Register (P4PND)

PORT 5

Port 5 is a general-purpose 8-bit I/O port with nibble-programmable pins. Port 5 is accessed directly by writing or reading the port 5 data register, P5 (E5H, set 1, bank 1).

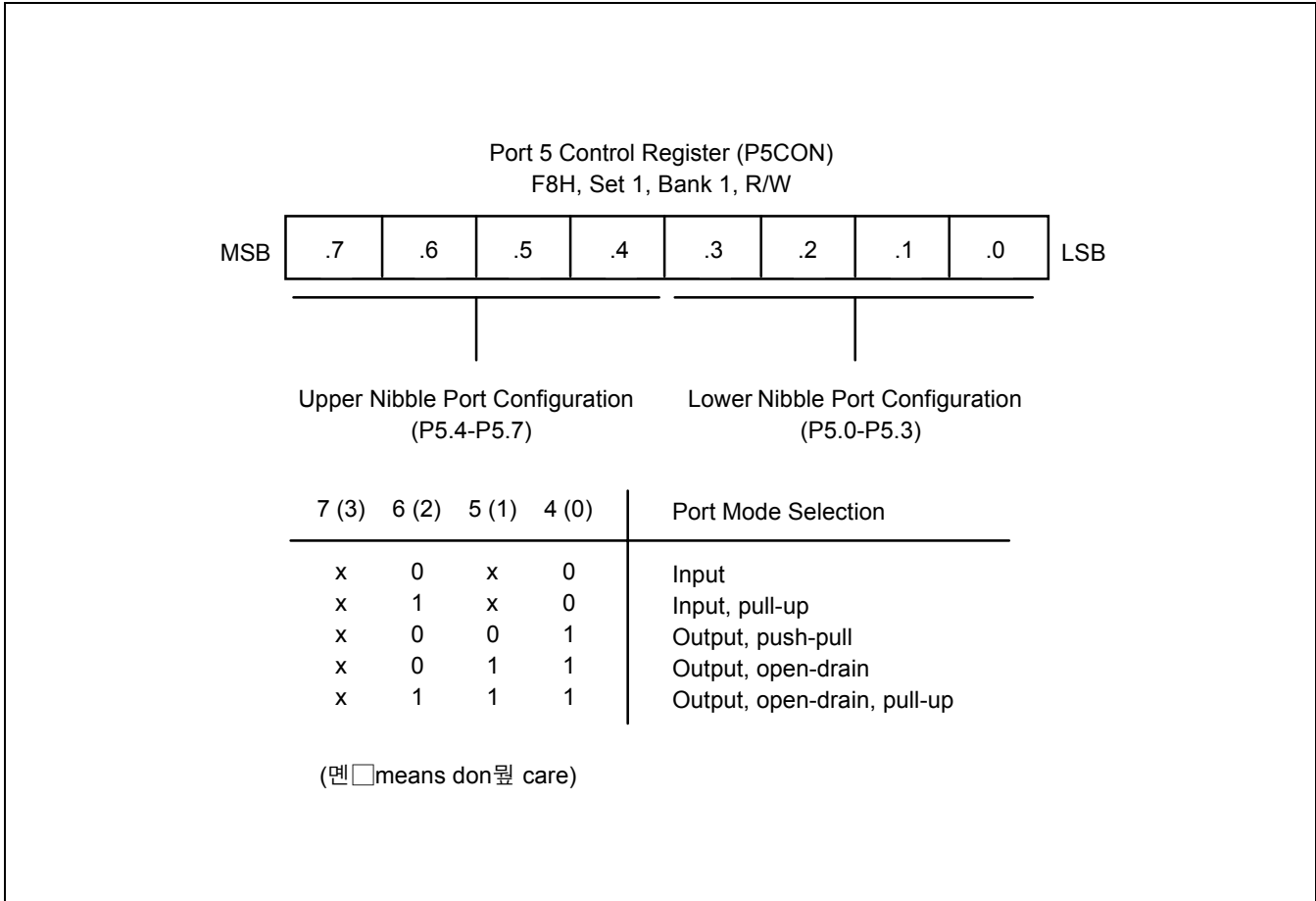


Figure 9-12. Port 5 Control Register (P5CON)

PORT 6

Port 6 is a general-purpose 8-bit output only port with nibble-programmable pins. Port 6 is accessed directly by writing or reading the port 6 data register, P6 (E6H, set 1, bank 1).

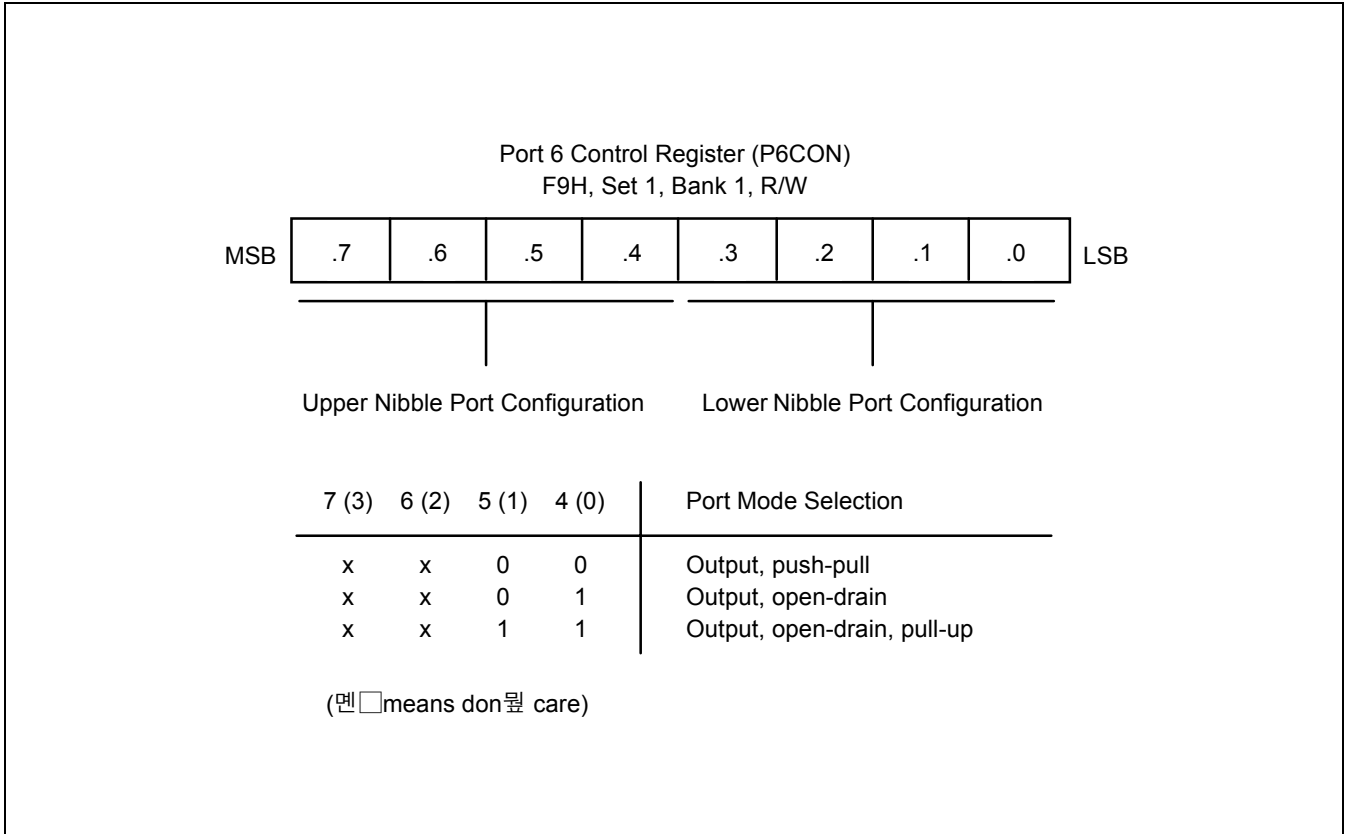


Figure 9-13. Port 6 Control Register (P6CON)

PROGRAMMING TIP — Configuring KS88C0716 Port Pins to Specification

This example shows how to configure KS88C0716 ports to sample specifications. Ports 0 through 6 to be configured as follows:

- Set P0.0–P0.3 to open-drain output mode
- Set P0.4–P0.7 to input mode with pull-up
- Set P1.0–P1.7 to push-pull output mode
- Set P2.0–P2.6 to input mode
- Set P2.7 to input mode with rising-edge interrupts
- Set P3.0 to input mode and configure timer C clock input
- Set P3.1–P3.7 to push-pull output mode
- Set P4.0–P4.4 to push-pull output mode
- Set P4.5 to input mode with rising-edge interrupts
- Set P4.6 to input mode with falling-edge interrupts
- Set P4.7 to input mode, falling-edge interrupts, push-pull
- Set P5.0–P5.7 to open-drain output mode with pull-up
- Set P6.0–P6.7 to open-drain output mode with pull-up

```

      .
      .
      .
LD      P0CON,#43H      ; P0.0–P0.3 ← open-drain output
                        ; P0.4–P0.7 ← input, pull-up
LD      P1CON,#11H      ; P1.0–P1.7 ← push-pull output
LD      P2CONH,#0C0H    ; P2.7 ← input, rising-edge interrupts
LD      P2CONL,#00H     ; P2.0–P2.6 ← input mode
LD      P3CONH,#0AAH    ; P3.1–P3.7 ← push-pull output
LD      P3CONL,#0A8H    ; P3.0 ← input, timer C clock input
LD      P4CONH,#087H    ; P4.7 ← input, falling-edge interrupts, pull-up
LD      P4CONL,#0FFH    ; P4.6 ← input, falling-edge interrupts
                        ; P4.5 ← input, rising-edge interrupts
                        ; P4.0–P4.4 ← push-pull output

SB1
LD      P5CON,#77H      ; P5.0–P5.7 ← open-drain output, pull-up
LD      P6CON,#33H      ; P6.0–P6.7 ← open-drain output, pull-up
SB0
LD      P4PND,#00H      ; Reset all pending register bits for port 4 interrupts
LD      P4INT,#0E0H     ; Enable port 4 interrupts (P4.7–P4.5)
                        ; Disable port 4 interrupts (P4.4–P4.0)
      .
      .
      .

```

10

BASIC TIMER and TIMER MODULE 0

MODULE OVERVIEW

KS88C0716/P0716 have three default timers: an 8-bit basic timer and two 8-bit general-purpose timer/counters. The 8-bit timer/counters is called timer module 0.

BASIC TIMER (BT)

You can use the basic timer (BT) in two different ways:

- As a watchdog timer to provide an automatic reset mechanism in the event of a system malfunction, or
- To signal the end of the required oscillation stabilization interval after a reset or a Stop mode release.

The functional components of the basic timer block are:

- Clock frequency divider (f_{OSC} divided by 4096, 1024, or 128) with multiplexer
- 8-bit basic timer counter, BTCNT (set 1, bank 0, FDH, read-only)
- Basic timer control register, BTCON (set 1, D3H, read/write)

BASIC TIMER CONTROL REGISTER (BTCON)

The basic timer control register, BTCON, is used to select the input clock frequency, to clear the basic timer counter and frequency dividers, and to enable or disable the watchdog timer function. It is located in set 1, address D3H, and is read/write addressable using register addressing mode.

A reset clears BTCON to '00H'. This enables the watchdog function and selects a basic timer clock frequency of $f_{OSC}/4096$. To disable the watchdog function, you must write the signature code '1010B' to the basic timer register control bits BTCON.7–BTCON.4.

The 8-bit basic timer counter, BTCNT (set 1, bank 0, FDH), can be cleared at any time during normal operation by writing a "1" to BTCON.1. To clear the frequency dividers, you write a "1" to BTCON.0.

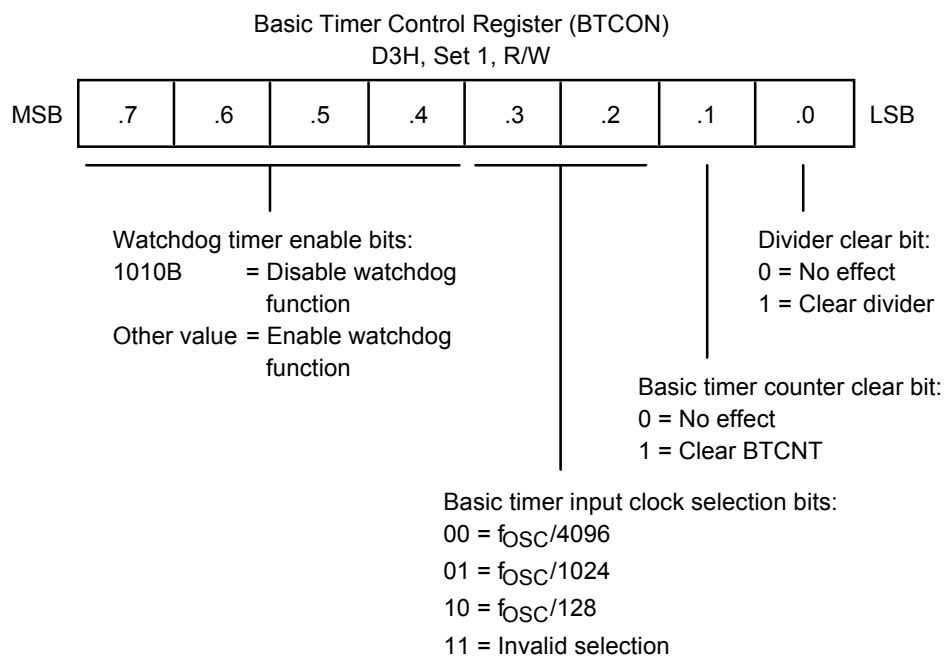


Figure 10-1. Basic Timer Control Register (BTCON)

BASIC TIMER FUNCTION DESCRIPTION

Watchdog Timer Function

You can program the basic timer overflow signal (BTOVF) to generate a reset by setting BTCON.7–BTCON.4 to any value other than '1010B'. (The '1010B' value disables the watchdog function.) A reset clears BTCON to '00H', automatically enabling the watchdog timer function. A reset also selects the CPU clock (as determined by the current CLKCON register setting), divided by 4096, as the BT clock.

A reset whenever a basic timer counter overflow occurs. During normal operation, the application program must prevent the overflow, and the accompanying reset operation, from occurring. To do this, the BTCNT value must be cleared (by writing a "1" to BTCON.1) at regular intervals.

If a system malfunction occurs due to circuit noise or some other error condition, the BT counter clear operation will not be executed and a basic timer overflow will occur, initiating a reset. In other words, during normal operation, the basic timer overflow loop (a bit 7 overflow of the 8-bit basic timer counter, BTCNT) is always broken by a BTCNT clear instruction. If a malfunction does occur, a reset is triggered automatically.

Oscillation Stabilization Interval Timer Function

You can also use the basic timer to program a specific oscillation stabilization interval following a reset or when Stop mode has been released by an external interrupt.

In Stop mode, whenever a reset or an external interrupt occurs, the oscillator starts. The BTCNT value then starts increasing at the rate of $f_{OSC}/4096$ (for reset), or at the rate of the preset clock source (for an external interrupt). When BTCNT.4 overflows, a signal is generated to indicate that the stabilization interval has elapsed and to gate the clock signal off to the CPU so that it can resume normal operation.

In summary, the following events occur when stop mode is released:

1. During stop mode, a power-on reset or an external interrupt occurs to trigger the Stop mode release and oscillation starts.
2. If a power-on reset occurred, the basic timer counter will increase at the rate of $f_{OSC}/4096$. If an external interrupt is used to release stop mode, the BTCNT value increases at the rate of the preset clock source.
3. Clock oscillation stabilization interval begins and continues until bit 4 of the basic timer counter overflows.
4. When a BTCNT.4 overflow occurs, normal CPU operation resumes.

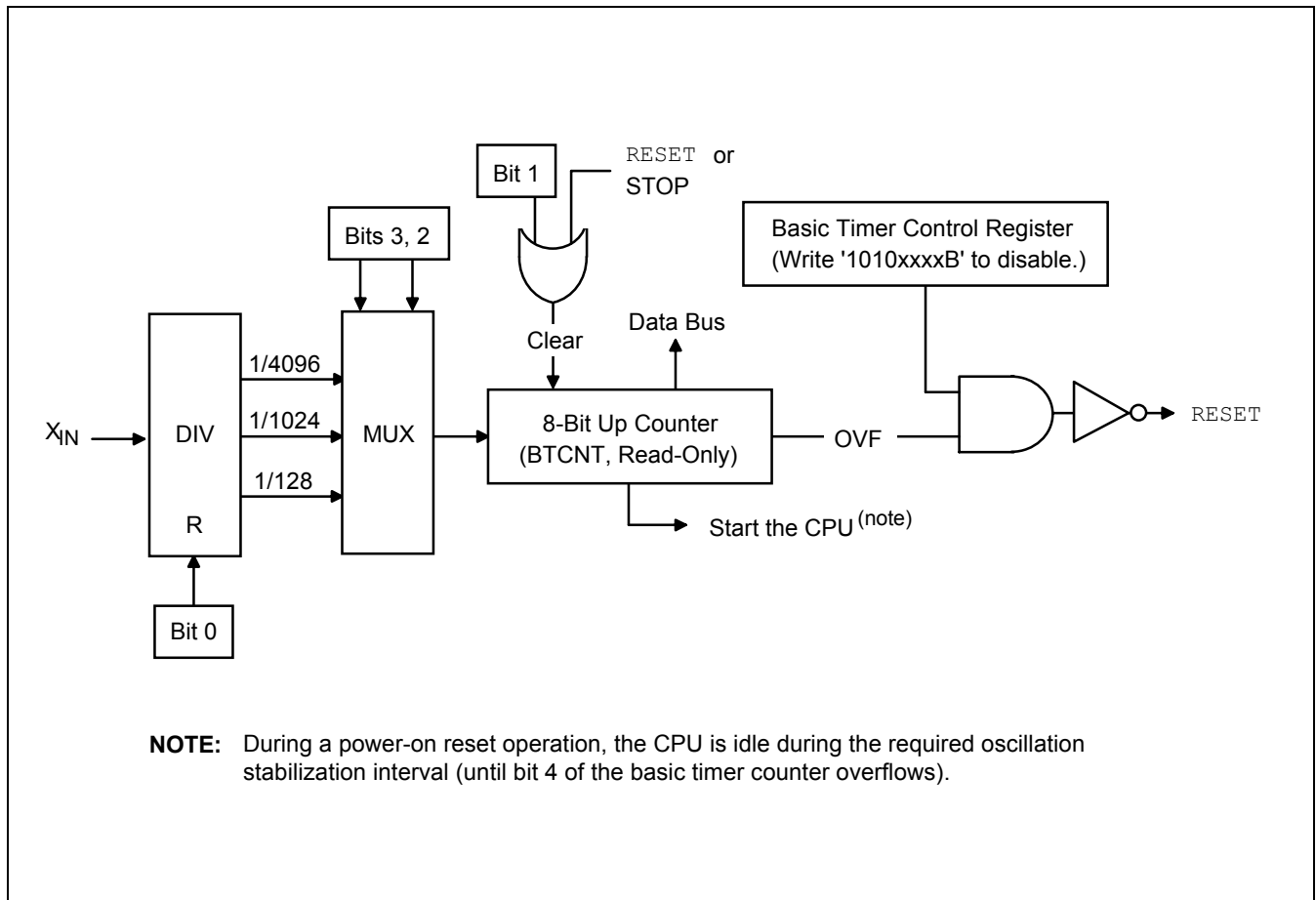


Figure 10-2. Basic Timer and Timer 0 Block Diagram

TIMER MODULE 0

OVERVIEW

The KS88C0716 timer module 0 (T0) has two 8-bit timer/counters called timer A and timer B. Each timer has an 8-bit counter register, an 8-bit data register, and an 8-bit comparator, and both are driven by the same clock input.

Timers A and B run continuously. The corresponding counters cannot be reset or accessed because their addresses are not mapped. Two control registers, T0CON and TBINT, are used to program timer module 0.

Timer A and B Operating Modes

Timers A and B can operate in interval mode or pulse width modulation (PWM) mode. The LSB of the T0CON register controls timer A operating mode; the LSB of the TBINT register controls timer B operation.

Timer Clock Input

Timers A and B have the same clock input source. You can select the non-scaled CPU clock or a divided-by-1024 basic timer divider output as the timer module 0 clock.

When the TCS bit (bit 3) in the T0CON register is "0", the system clock divided by 1024 (decimal) is selected. Setting the TCS bit to "1" configures the non-scaled CPU clock pulse.

The selected timer clock is divided by a 4-bit prescaler that is located in bits 7–4 of the T0CON register (TPS3–TPS0).

Timer A Interrupts

In interval mode, both timers generate a match signal when the count value the reference data value in the TADATA or TBDATA register is the same. When a match condition is detected, the count register is cleared, an interrupt request is generated, and counting resumes from zero.

The timer A interrupt is enabled by the ETAI bit in the T0CON register. The timer A interrupt pending bit (T0CON.1) must be cleared by software after the CPU has acknowledged a timer A interrupt.

Timer A Pending Bit (TAIP)

Bit 1 (TAIP) in the timer module 0 control register T0CON is the interrupt pending bit for timer A.

When a read operation shows TAIP = "0", no interrupt is pending; when TAIP = "1", a timer A interrupt is pending. To reset (clear) the TAIP bit, you must write a "0" to the TAIP bit.

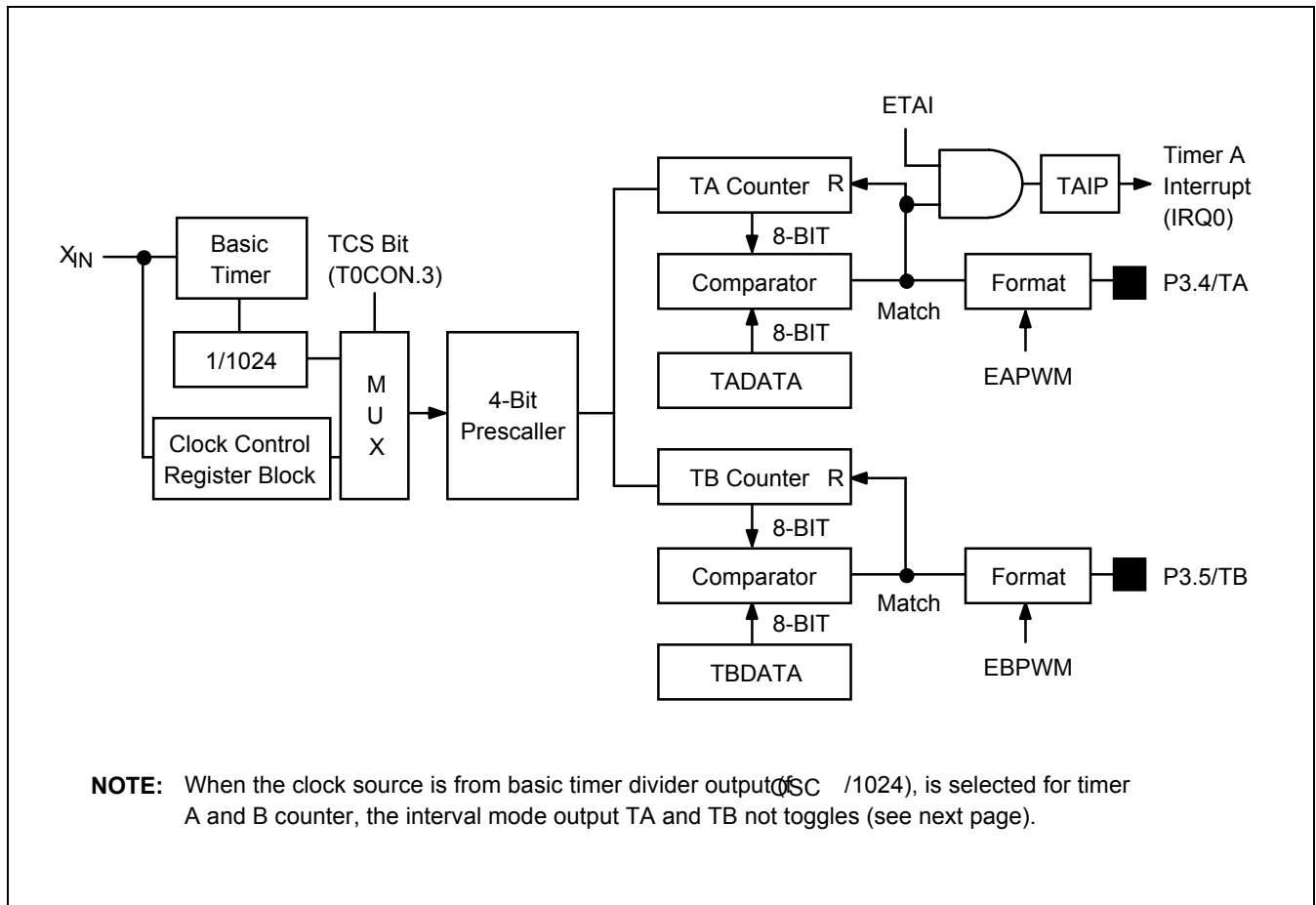


Figure 10-3. Timer Module 0 Function Block Diagram

TIMER MODULE 0 FUNCTION DESCRIPTION

Timer A and timer B operate in either interval mode or pulse width modulation mode, as selected by the EAPWM bit (T0CON.0) and the EBPWM (TBINT.0).

Pulse Width Modulation Mode

In PWM mode, the timer A data register is used to modulate the pulse width of the TA output pin. The TA pin toggles at a frequency equal to the selected timer A input clock, divided by 256 (that is, by the prescaler output). It has a duty cycle ranging from 0% to 99.6%, depending on the value in the TADATA register. The contents of the TADATA register are compared to the 8-bit TA count value. The TA pin value toggles to "1" whenever the TADATA value is greater than the TA value; otherwise, it is "0".

For example, assume that the timer A input clock frequency is 4 MHz. The TA pin toggles high every 64 μ s (4 MHz divided by 256). If the timer A data register has a value of 80H (128 decimal), the level at the TA pin goes high for 128 cycles (32 μ s), and then low for 128 cycles. This results in a 50% duty cycle (128/256).

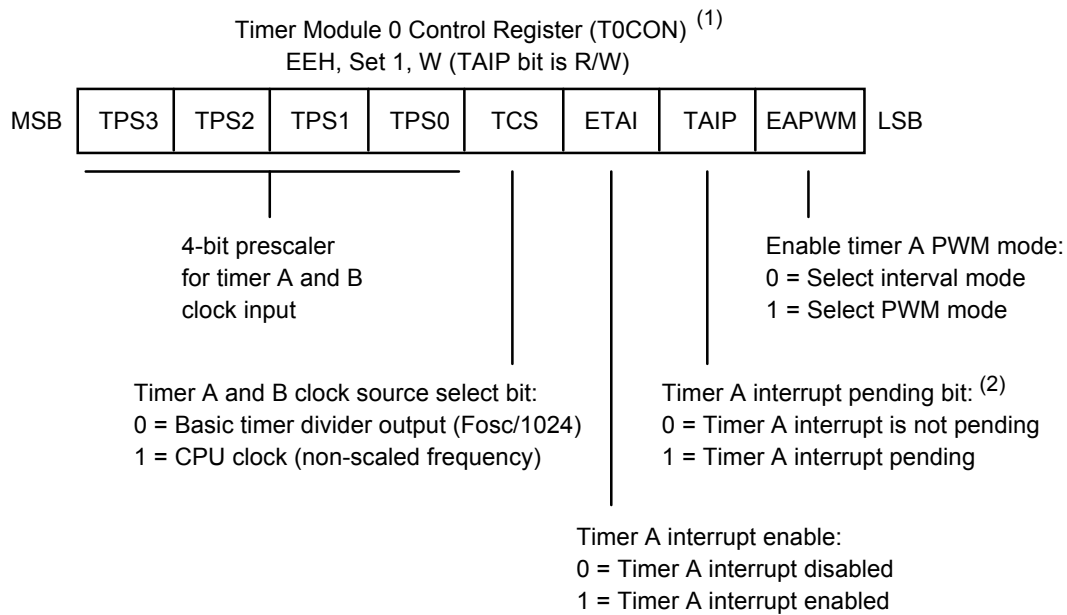
A TADATA value of '00H' (its reset value), produces a constant "0" level at the TA pin. If the TADATA value is 'FFH', a 99.6% duty cycle (255/256) will result.

Interval Mode

In interval mode, the TA and TB pin toggles low during the 1/2 period of a CPU clock cycle whenever a match occurs between the timer counter value and the reference data register value. The TA and TB pin remains at high level at all other times. The timer is reset each time a match occurs.

In interval mode, the pulse width is fixed, as determined by the frequency of the timer clock and the reference data register value.

When the basic timer divider output ($f_{OSC}/1024$) is selected for the timer A and B counter, the interval mode output TA and TB do not toggles. But when the CPU clock is selected, the TA and TB toggles. And in any case, it is capable to generate timer A match interrupt.

**NOTE:**

1. To avoid possible programming errors, we recommend using load instructions (with the exception of LDB) to manipulate T0CON register values.
2. The TAIP bit (bit 1) is the only readable bit in the T0CON register. A ☐ is written to TAIP to reset (clear) the timer A interrupt pending bit.
3. When the basic timer divider output ($f_{OSC}/1024$) is selected for timer A counter clock source, the interval mode output is disabled.

Figure 10-4. Timer Module 0 Control Register (T0CON)

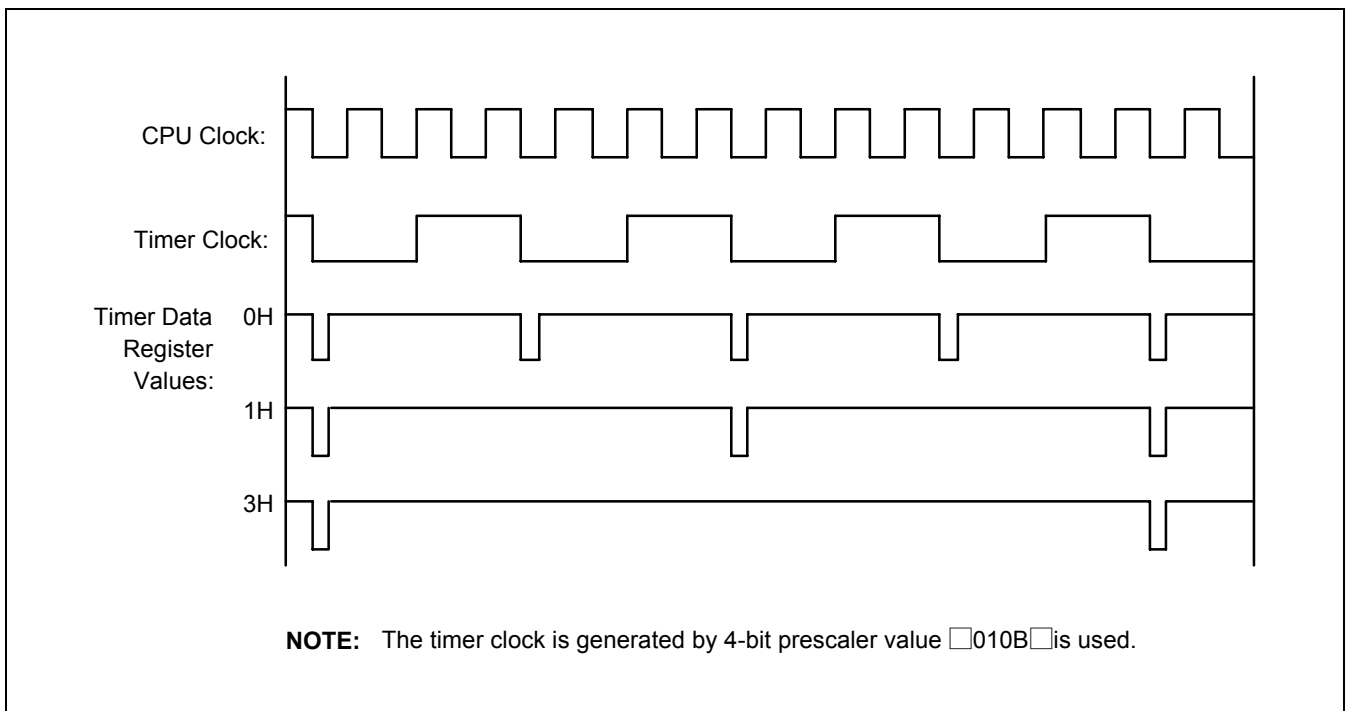


Figure 10-5. Timer A and B Waveforms in Interval Mode

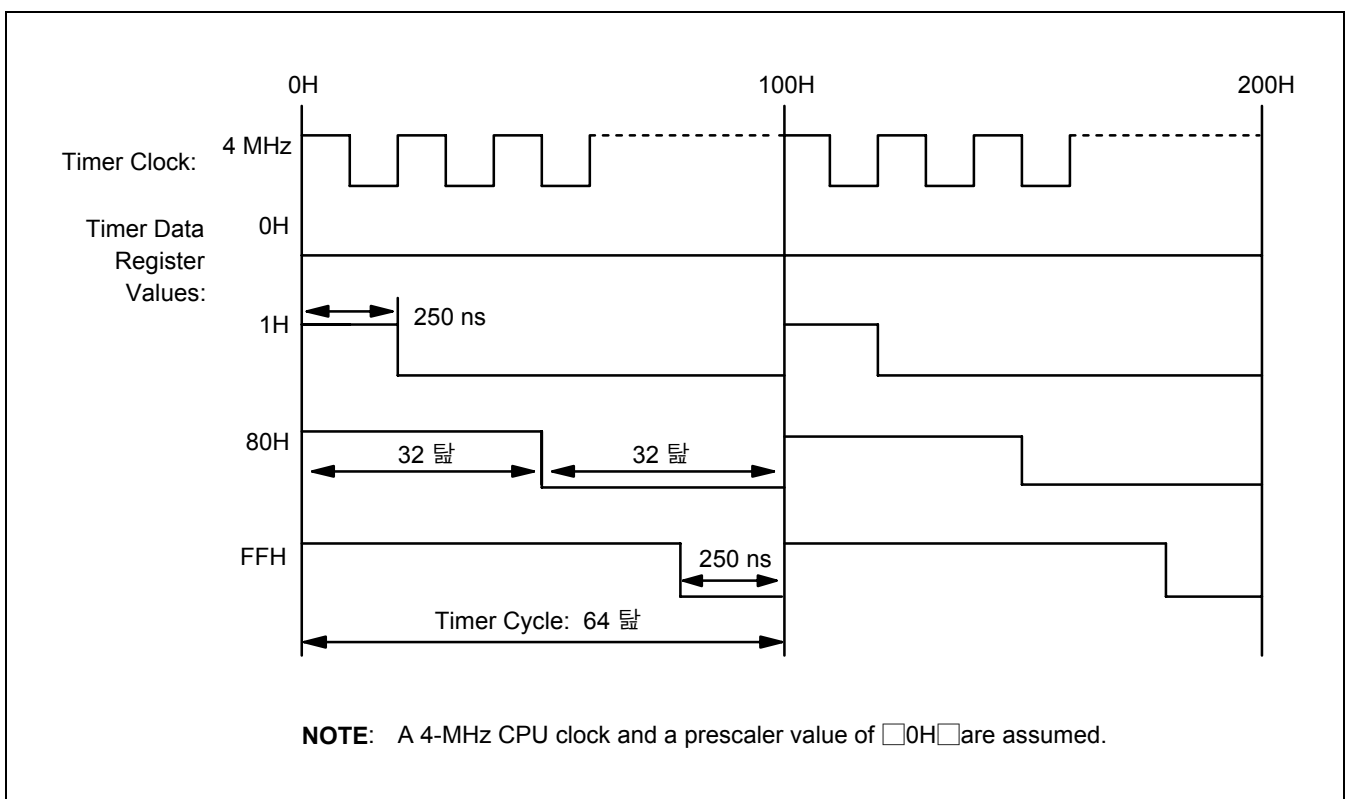


Figure 10-6. Timer A and B Waveforms in Pulse Width Modulation Mode

TIMER B INTERVAL REGISTER (TBINT)

— Select the timer B operating mode (interval timer or PWM)

The least significant bit of the TBINT register is called the EBPWM bit. When the EBPWM bit is "0", timer B operates in interval timer mode; when it is "1", timer B operates in PWM mode.

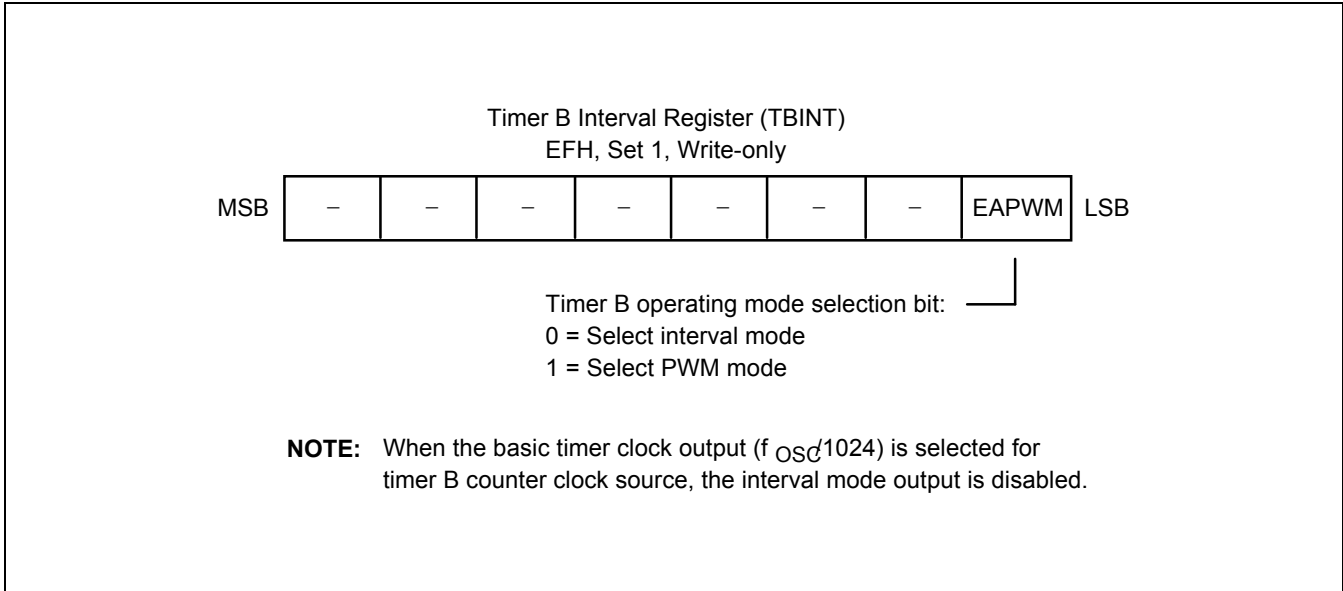


Figure 10-7. Timer B Interval Register (TBINT)

PROGRAMMING TIP — Configuring the Basic Timer

This example shows how to configure the basic timer to sample specifications:

```

RESET      ORG          0100H
            DI           ; Disable all interrupts
            SB0          ; Select bank 0
            LD           BTCON,#0A0H ; Disable the watchdog timer
            LD           CLKCON,#98H ; Non-divided clock
            CLR          SYM        ; Disable global and fast interrupts
            CLR          SPL        ; Stack pointer low byte ← "0"
            ;             ; Stack area starts at 0FFH
            .
            .
            .
            SRP          #0C0H     ; Set register pointer ← 0C0H
            EI           ; Enable interrupts
            .
            .
            .
MAIN        LD           BTCON,#52H ; Enable the watchdog timer
            ;             ; Basic timer clock: fOSC/4096
            ;             ; Clear basic timer counter

            NOP
            NOP
            .
            .
            .
            JP          T,MAIN
            .
            .
            .

```

PROGRAMMING TIP — Configuring Timer A and Timer B

This example sets timer A to normal interval mode, disables timer B, sets the oscillation frequency of the timer clock, and determines the execution sequence after a timer A interrupt occurs. The program parameters are:

- Timer A is used in interval mode; the timer interval is set to approximately 2 milliseconds
- Timer B is disabled
- Oscillation frequency is 12 MHz (CPU clock = 6 MHz)
- $90H \leftarrow 90H + 91H + 92H + 93H + 94H$ is executed after a timer A interrupt

	ORG	0100H	; Reset address
	JP	T,START	
	ORG	00BEH	; Timer A interrupt vector
	VECTOR	TAINT	
	.		
	.		
START	ORG	0200H	
	DI		
	.		
	.		
	.		
	LD	T0CON,#5CH	; PS $\leftarrow$ 5 (for divide-by-6) ; CPU clock is selected for timer 0 (A and B) ; Enable timer A interrupt, reset timer A pending register ; Select interval mode for timer A
	LD	TADATA,#01H	; TADATA $\leftarrow$ 01 (divided by two) ; $6\text{ MHz}/(6 \times 2) = 0.5\text{ MHz} (= 2\text{ }\mu\text{s})$
	EI		; Enable interrupts
	.		
	.		
	.		
TAINT	PUSH	RP0	; Save RP0 to stack
	SRP0	#90H	; $RP0 \leftarrow 90H$
	ADD	R0,R1	; $R0 \leftarrow R0 + R1$
	ADC	R0,R2	; $R0 \leftarrow R0 + R2$
	ADC	R0,R3	; $R0 \leftarrow R0 + R3$
	ADC	R0,R4	; $R0 \leftarrow R0 + R4$
			; $(R0 \leftarrow R0 + R1 + R2 + R3 + R4)$
	LD	T0CON,#54H	; Reset timer A pending register
	POP	RP0	; Restore register pointer 0 value
	IRET		; Return from interrupt service routine
	.		
	.		
	.		

11

TIMER MODULE 1

OVERVIEW

The KS88C0716 timer module 1 (T1) consists of two timer/counters, called timers C and D. Each can be used as an interval timer or as an event counter. The functional components of the T1 module are:

- Timer module 1 control register (T1CON)
- Timer module 1 mode register (T1MOD)
- Timer C, D high-byte count registers (TCH, TDH)
- Timer C, D low-byte count registers (TCL, TDL)
- Timer C gate pin (P4.0/INT4 (TCG), pin 15)
- Timer D gate pin (P4.1/INT5 (TDG), pin 14)
- Timer C external clock input pin (TCCK, pin 31)
- Timer D external clock input pin (TDCK, pin 30)

Timer module 1 can operate in four different modes:

- Mode 0 13-bit timer/counter
- Mode 1 16-bit timer/counter
- Mode 2 8-bit auto-reload timer/counter
- Mode 3 Two 8-bit timer/counters

When used as an interval timer, the corresponding count register is incremented based on the internal timer clock rate. You can select an external clock source, or a divided-by-6 CPU clock as the timer clock.

When used as an event counter, the timer's count register is incremented in response to 1-to-0 transitions at its corresponding external input pin (TCCK for timer C or TDCK for timer D). The external input is sampled at every fifth CPU clock pulse.

When a high-level sample is immediately followed by a low-level sample (that is, when a 1-to-0 transition occurs), the count register is incremented by one. (The new count value is written to the count register five CPU clocks *after* the one in which the 1-to-0 transition was detected.) It therefore takes two complete sampling cycles (12 CPU clocks) to recognize a 1-to-0 transition.

There are no restrictions on the duty cycle of the external signal input. To ensure that a given level is sampled at least once before it changes, it should be held for at least six CPU clocks.

TIMER MODULE 1 MODE REGISTER (T1MOD)

The timer 1 mode register T1MOD (FBH) contains control settings for the following timer C and D functions:

- Clock source selection for timer operation
- Gate function enable and disable
- Operating mode selection (four available modes)

The lower nibble bits (0–3) control timer C and the upper nibble bits (4–7) control timer D. After a reset, the T1MOD register is cleared to '00H'. This selects the CPU clock (divided by 6) as the clock source, disables the gate functions, and configures timers C and D as a 13-bit timer/counter.

Clock Source Selection Options

Timer C and timer D each has two clock source selection options:

- 1) the internal CPU clock pulse, divided by six, or 2) an external clock source.

If you select an external clock source, you must also configure the corresponding input pin: for timer C, the clock input pin is TCKK (P3.0, pin 31); for timer D, the external clock input pin is TDCK (P3.1, pin 30). The port 3 low-byte control register P3CONL configures the clock input pins.

Gate Function Enable

Timers C and D each has a gate function enable bit in the T1MOD register: bit 3 (GCE) is for timer C and bit 7 (GDE) for timer D. Reset clears the GxE bits to "0", turning the gate function off. You can then enable the timers in one of the four available operating modes, independently of the gate function.

Timer Module 1 Operating Modes

Two bit-pairs in T1MOD (bits 0/1 for timer C and bits 4/5 for timer D) are used to select one of four timer module 1 operating modes.

Modes 0, 1, and 2 are functionally identical for both timers; mode 3 operates differently for timer C and timer D.

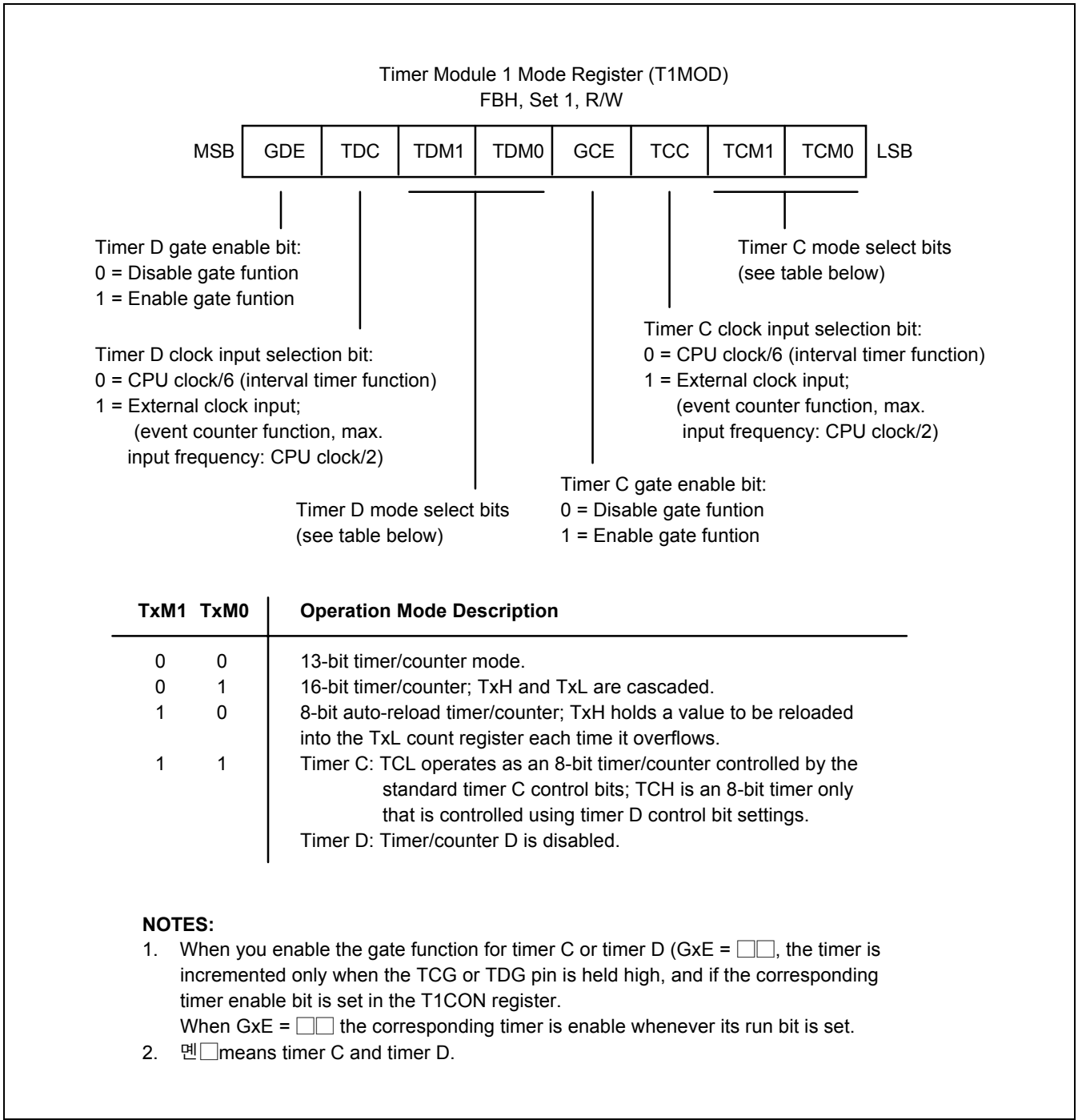


Figure 11-1. Timer Module 1 Mode Register (T1MOD)

TIMER MODULE 1 CONTROL REGISTER (T1CON)

The timer 1 control register T1CON (FAH) controls timer C and timer D operation. A reset clears T1CON to '00H'. This disables timers C and D, and stops all interrupt processing for timer module 1. It also selects the normal baud rate setting for the SIO baud rate generator function.

Timer Run Control Bits

T1CON bits 0 and 1 (TCE and TDE, respectively) are the timer C and D run control bits. You can start or stop the timers independently of each other.

Interrupt Control Function

External interrupts can be gated to timer 1 through the TCG and TDG pins. The gate input source must be enabled externally.

To configure the gate input pins, make the appropriate settings to the low-byte port 4 control register (P4CONL). When the gate function starts, the INT4 external interrupt is gated to timer C and INT5 is gated to timer D.

The timer interrupt enable bits that you must set for event counting are bit 2 (TCIE) for timer C and bit 3 (TDIE) for timer D. Each timer has an interrupt pending bit that serves as a flag for gated external interrupts. These flags can be polled by software.

When a timer's interrupt enable bit is "1", an interrupt request branches to the timer's vector address whenever a pending condition is detected. However, the branch occurs only after the current instruction has executed, and if no other interrupts with a higher priority are being serviced.

The branch operation does not automatically clear the pending flag; it must be cleared by software by writing a "0" to the appropriate pending bit in the T1CON register.

Baud Rate Generator Function

You can also use timer module 1 as a baud rate generator for the serial port (UART) module. Bit 7 of the T1CON register (BSEL) is the baud rate select bit. The "0" setting selects a normal baud rate based on the timer module 1 clock source (either CPU clock divided by six, or an external clock).

By setting the BSEL bit to "1", you can double the frequency of the normal baud rate selection. If you select the double baud rate, you must set the serial port to operate either in mode 1, 2, or 3. (Mode 0 operation is not allowed.)

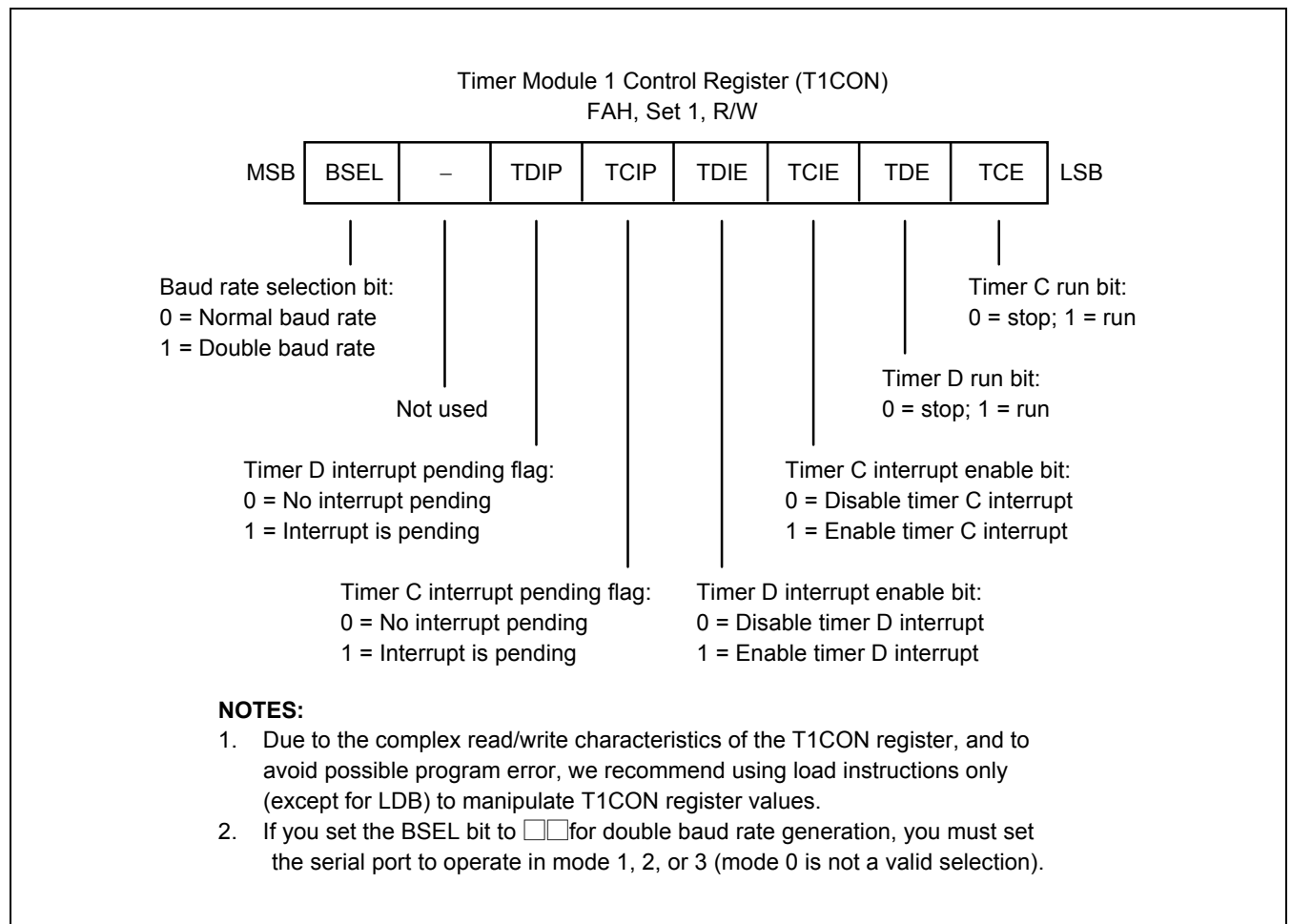


Figure 11-2. Timer Module 1 Control Register (T1CON)

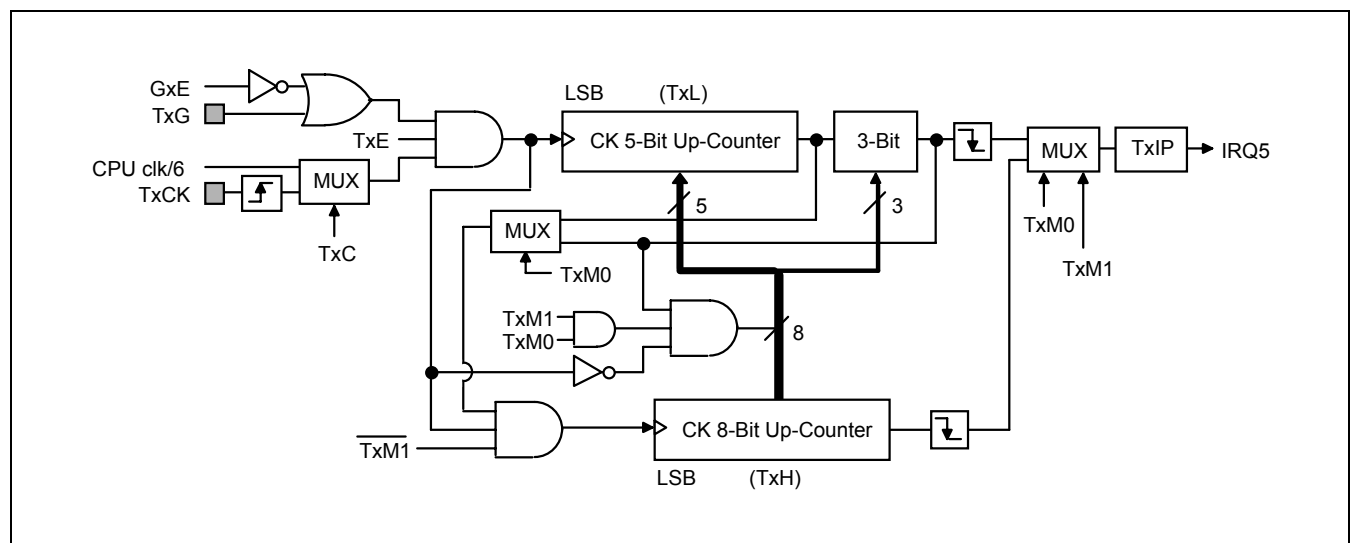


Figure 11-3. Timer Module 1 Block Diagram (In mode 0, 1 and 2)

TIMER MODULE 1 GATE FUNCTION DESCRIPTION

Two port 4 pins (P4.0 and P4.1) can be configured as interrupt input gate pins for timers C and D. These pins also serve as input pins for external interrupts INT4 and INT5, respectively.

You can use the P4.0/INT4 pin for gate input to timer C gate (TCG) and the P4.1/INT5 pin for gate input to timer D (TDG). To enable the gate function, two conditions must be met:

- The corresponding timer run bit in the T1CON register must be set to "1".
- The corresponding external interrupt pin must be configured to input mode (high level, "1").

The gate function lets the timer measure the duration of pulses applied to the external interrupt pin relative to the timer's counter value (see Figure 11-4).

If the timer's interrupt function is enabled, each gate pulse will trigger an interrupt (INT4 for timer C and INT5 for timer D). The interrupt service routine can then read the counter value that accumulated during the time the signal at the TCG or TDG pin was at high level. In this way, timer module 1 can operate as an event counter for an external device.

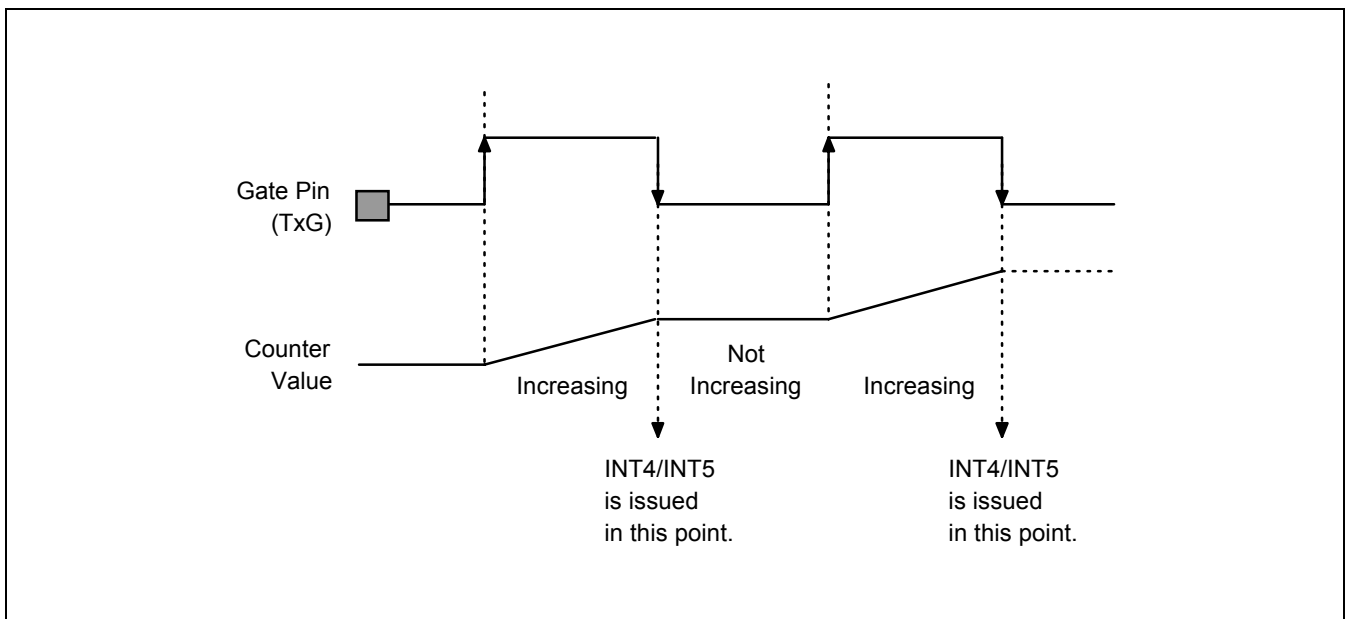


Figure 11-4. Count Value Incrementing; Gate Function Enabled

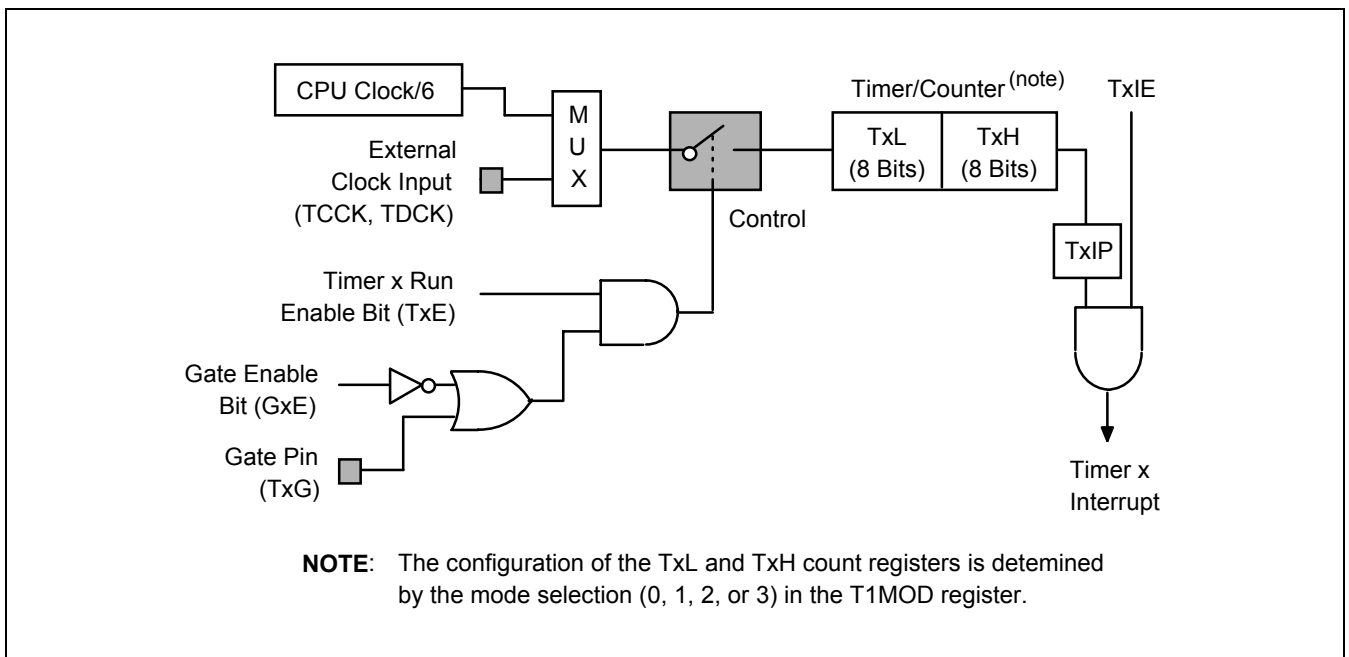


Figure 11-5. Timer Module 1 Gate Function Block Diagram

Timer Module 1, Mode 0 Operation

Mode 0 operation is functionally identical for timers C and D. In mode 0, the corresponding count registers (TxH, TxL) are configured as a 13-bit timer/counter. Mode 0 is automatically selected by a reset.

Eight bits of the high-byte count register (TxH) are used as the 8-bit counter; the five lower bits of the low-byte count register (TxL) are used as the 5-bit counter. The value of the upper three bits of TxL are undetermined and should be ignored. Both registers are read and write programmable.

The value of the 8-bit counter (TxH) is undetermined after a reset. Simply enabling a timer by setting its run bit to "1" does not automatically clear the TxH count value. It must be cleared by software.

When the TxH count overflows, the timer's interrupt pending flag TxIP (T1CON bits 4 and 5) is automatically set to "1". The interrupt pending flag is polled to generate the timer C or timer D interrupt.

Timer Module 1, Mode 1 Operation

Mode 1 operates in the same way as mode 0, except that the 8-bit timer C and D counters (TxH and TxL) function together as a 16-bit event counter.

Timer Module 1, Mode 2 Operation

In mode 2, the high-byte and low-byte count registers TxL and TxH function together as a single 8-bit counter. The low-byte register (TxL) is used as the 8-bit counter. When TxL overflows, it is automatically reloaded with an 8-bit value stored in the high-byte count register, TxH.

When the TxL counter overflow occurs, the corresponding interrupt pending flag (TxIP) in the T1CON register is automatically set to "1". The counter is then reloaded with the value stored in TxH, and counting resumes. (You must pre-load the TxH reload value by software.) The reload value is unchanged by the reload operation.

Assuming that the appropriate interrupt enable bit (TxIE) in the T1CON register is set, the timer's interrupt pending flag can then be polled to generate the timer C or timer D interrupt request.

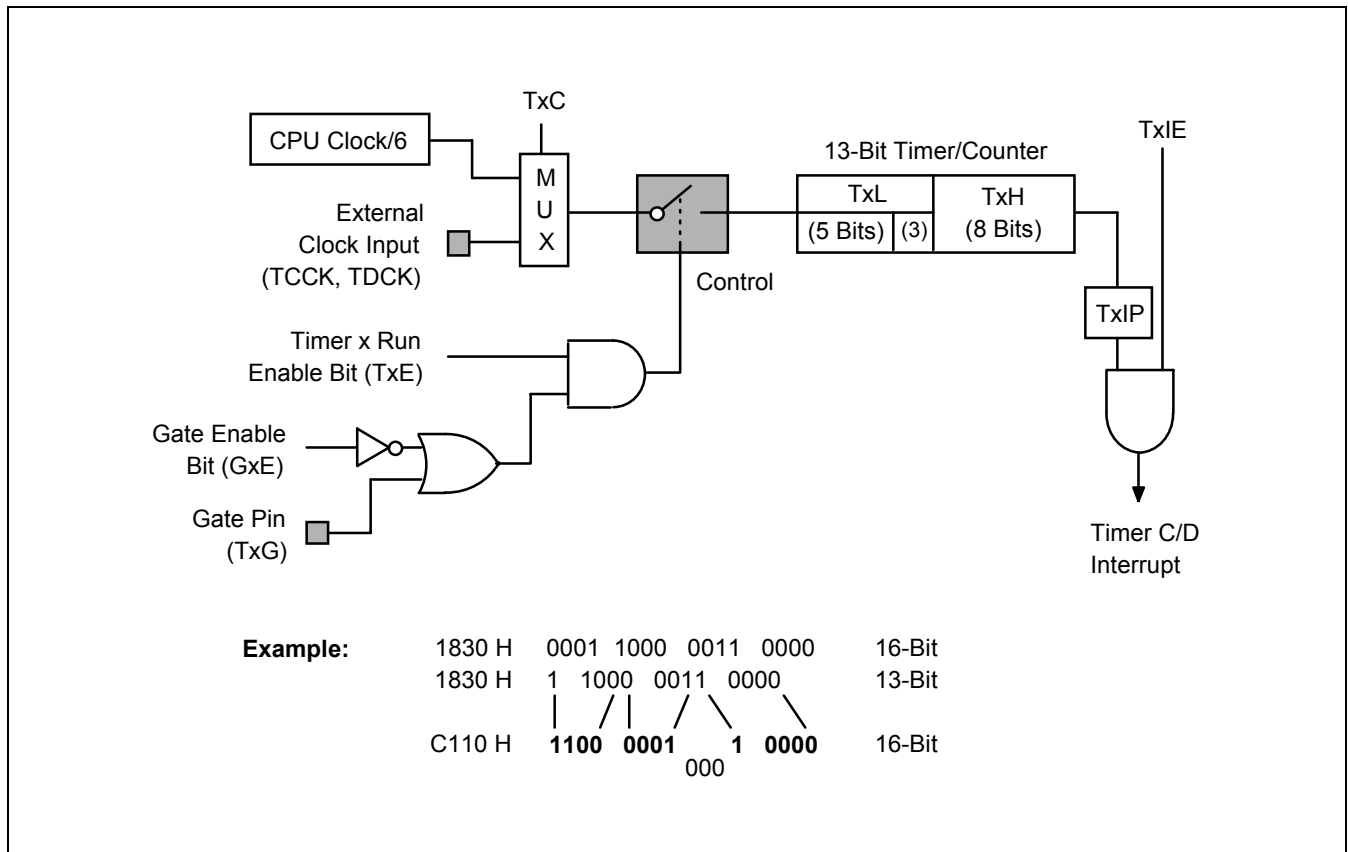


Figure 11-6. Timer Module 1 Mode 0 Function Diagram

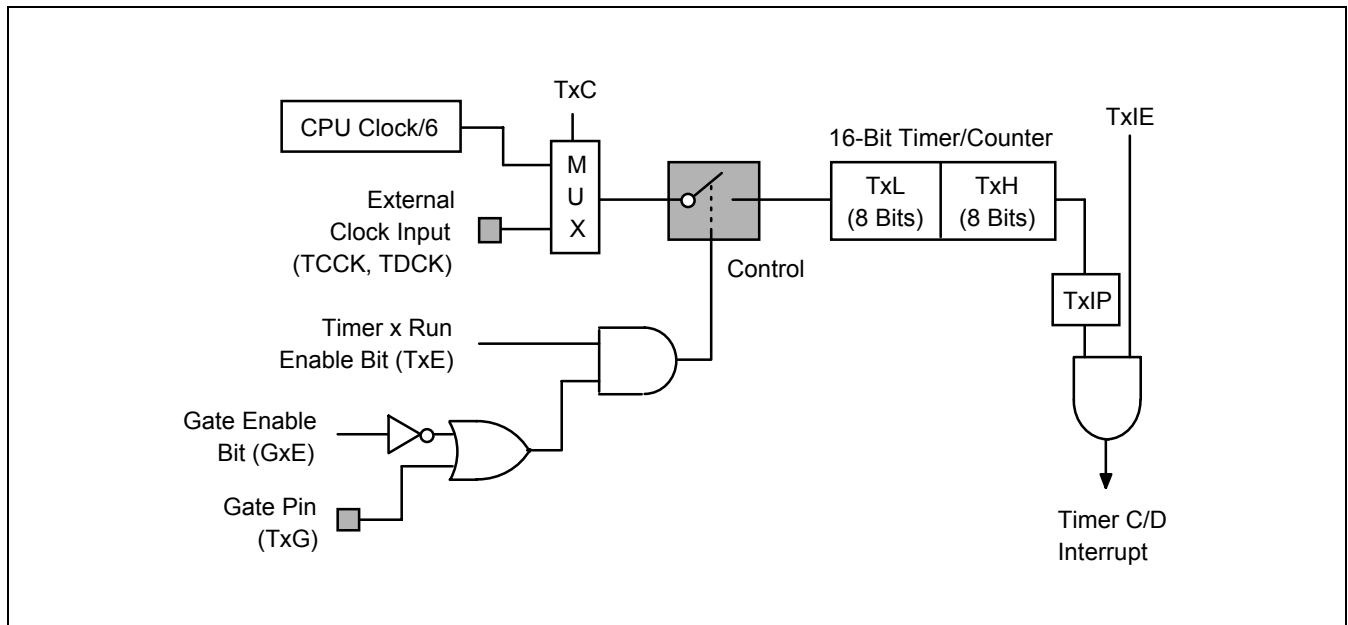


Figure 11-7. Timer Module 1 Mode 1 Function Diagram

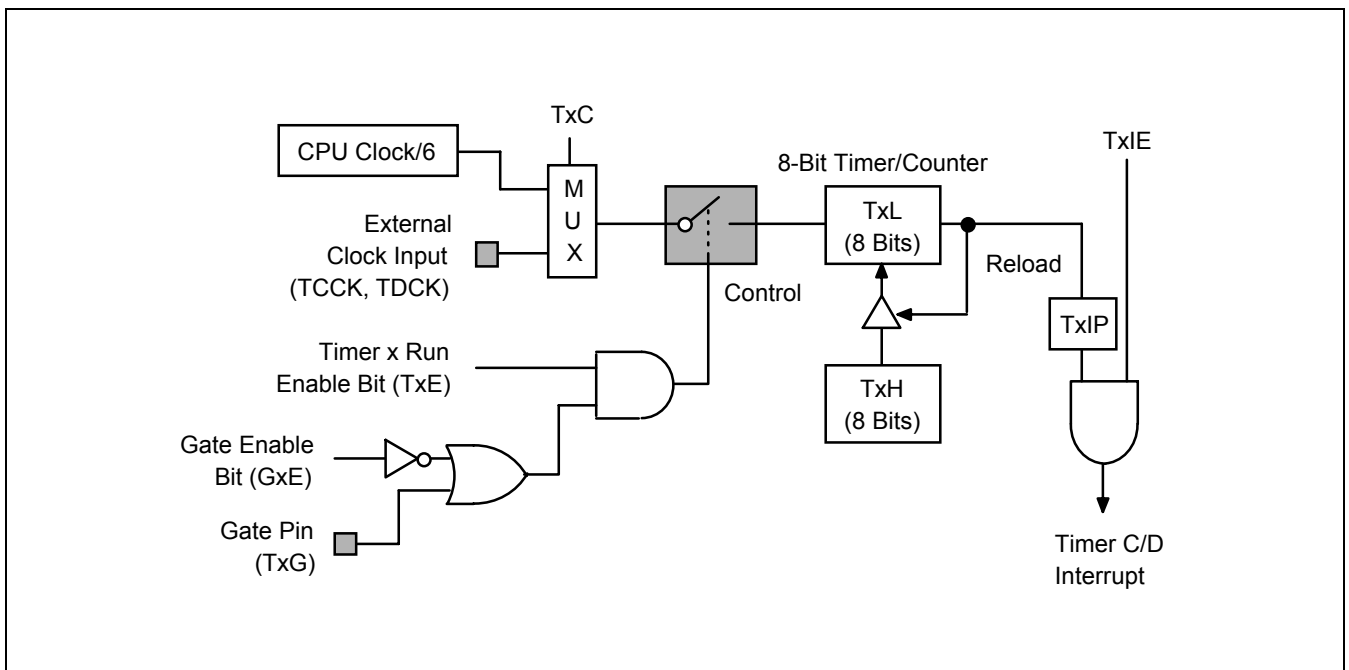


Figure 11-8. Timer Module 1 Mode 2 Function Diagram

TIMER MODULE 1, MODE 3 OPERATION

Unlike modes 0, 1, and 2, in mode 3, timer C functions as two separate 8-bit counters while timer D does not operate.

To select mode 3, the TCM1/TCM0 bit-pairs in the T1MOD register are set to '11B.' This setting establishes the timer C count registers, TCL and TCH, as two separate counters with one difference: TCL is an 8-bit timer/counter and TCH is an 8-bit timer only (that is, it has no external input). To program TCL for mode 3 operation, you must also write the appropriate values to the timer C mode control bits in the T1MOD register.

With TCH locked into a timer function of counting internal clocks, it assumes control of the timer D interrupt control bits. A "timer D interrupt" is thereby generated by the timer C counter.

When timer C is set to operate in mode 3, timer D can be turned on and off by switching it in and out of its own mode 3 operating status, or it can be used by the serial I/O port as a baud rate generator.

Timer D can be used in mode 0, 1 and 2 during the timer C is in mode 3 as any application that does not require it to generate an interrupt.

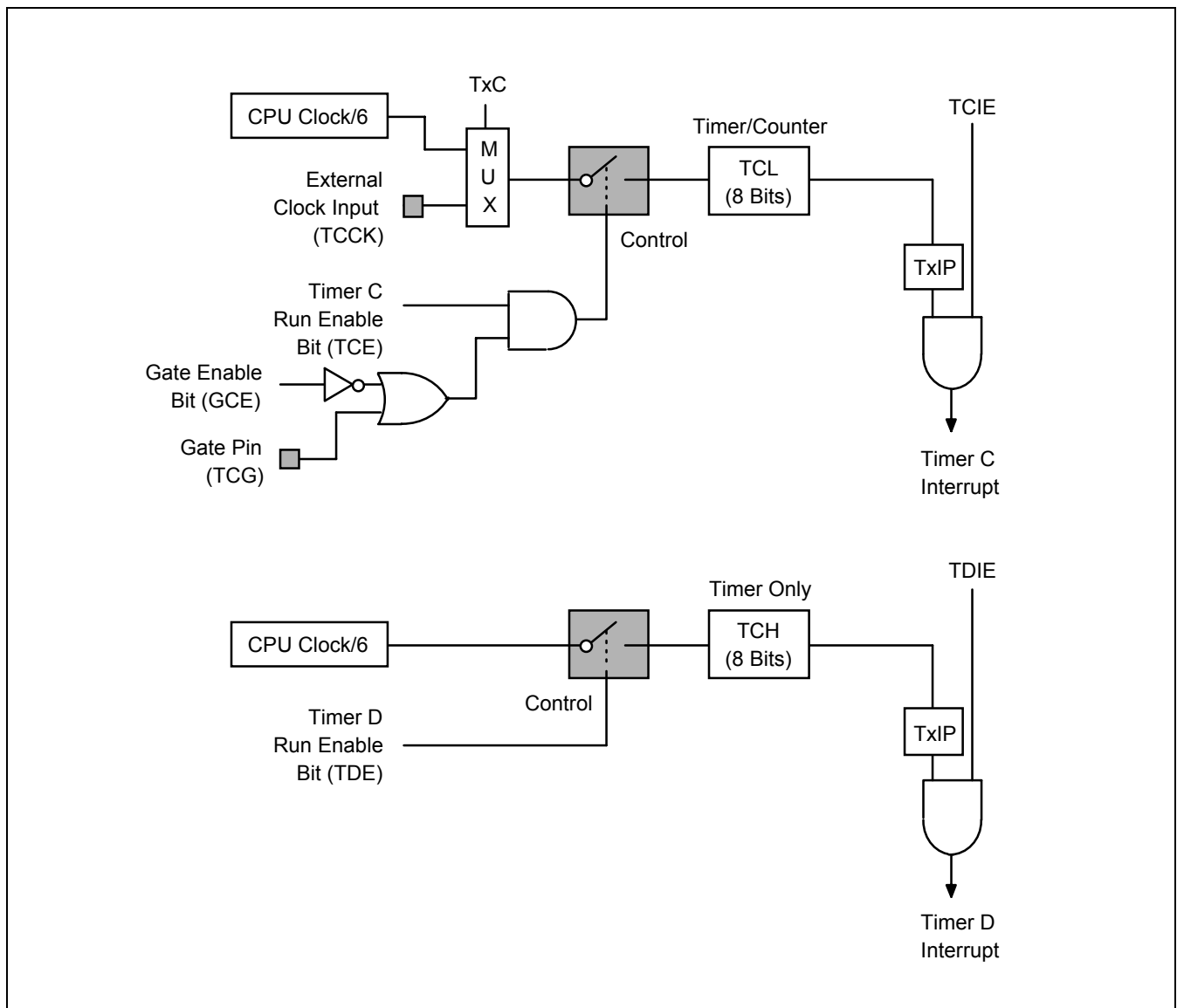


Figure 11-9. Timer Module 1 Mode 3 Function Diagram

 PROGRAMMING TIP — Timer Module 1, Operating Mode 0

This example shows how to program timer module 1 (timers C and D) to operate in 13-bit timer/counter mode (that is, in mode 0). The program parameters are:

- Only timer C is used for this example; timer D is disabled
- CPU clock frequency = 6 MHz
- Timer C input clock = 1 MHz
- Timer C interrupts occur in 2-ms intervals
- Each timer C interrupt toggles the P0.0 pin

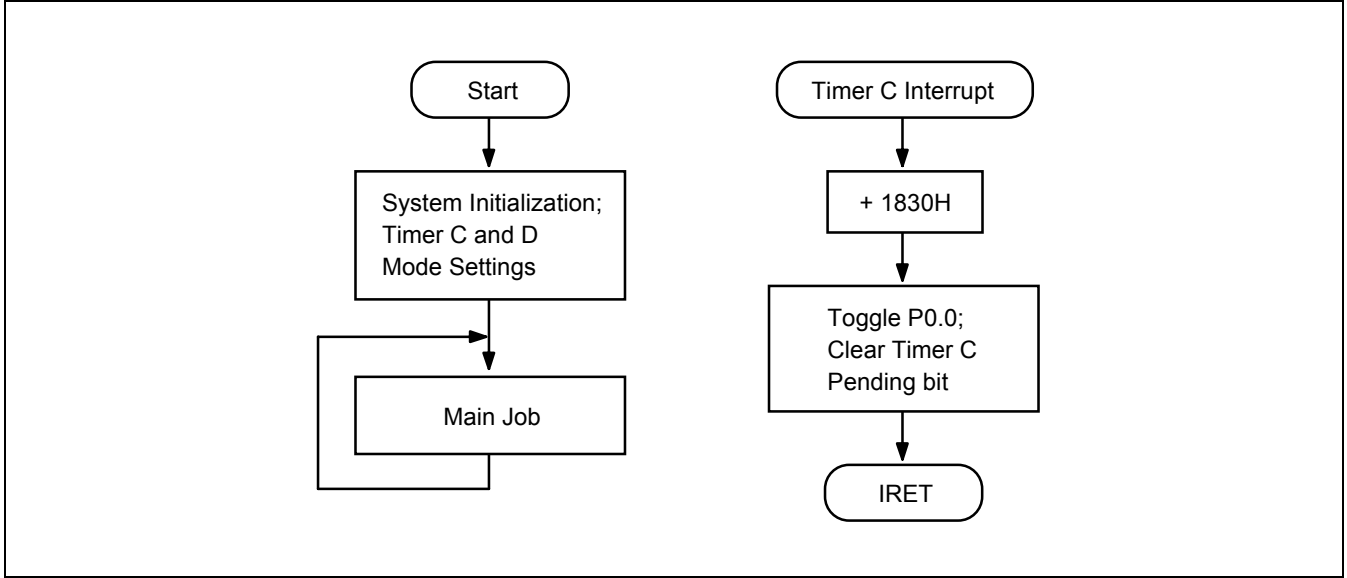


Figure 11-10. Timer Module 1 Mode 0 Programming Tip Flowchart

START	DI	; Disable interrupts
	.	
	.	; System initialization settings
	.	
	LD	P0CON,#11H ; Port 0 output push-pull mode select
	LD	TCH,#0C1H ; 07D0H counter value is 2 ms
		; 2000H – 07D0H = 1830H
	LD	TCL,#10H ; (TCH,TCL) ← 1830H
		; The low 5 bits of 1830H are '10H'
		; Bits 5–12 of 1830H are 'C1H'
	LD	T1MOD,#30H ; Timer D disable, timer C 13-bit timer mode select,
		; Timer clock = CPU clock /6
	LD	T1CON,#05H ; Timer C interrupt enable; timer C run enable
	EI	; Enable interrupts
	.	
	.	
	.	

PROGRAMMING TIP — Timer Module 1, Operating Mode 0 (Continued)

```

MAIN      NOP
          .
          .
          .
          CALL      JOB                ; Run other job
          .
          .
          .
          JP        T,MAIN

```

Then, the timer C overflow interrupt service routine (TC_INT):

```

TC_INT:
      ADD      TCH,#0C1H                ; TC ← TC + 1830H
      ADD      TCL,#10H                 ; TCL is just a 5-bit counter
      ADC      TCH,#00H                 ;
      XOR      P0,#01H                 ; Toggle the P0.0 pin
      LD       T1CON,#05H              ; Clear the timer C pending bit
      IRET                               ; Return from interrupt

```

PROGRAMMING TIP — Timer Module 1, Operating Mode 1

This example shows how to program timer module 1 (timers C and D) to operate in 16-bit timer/counter mode (that is, in mode 1). The program parameters are:

- CPU clock frequency = 6 MHz
- Clock input pulse at the TCCK pin is an unknown frequency
- Clock input pulse at the timer C gate pin (TCG) is equal to 62.5 Hz (50% duty)
- Interrupt INT4 occurs with each falling edge at the TCG pin

Program Function Description

Timer C operates as a frequency counter. An unknown frequency is being input through the timer C clock input pin (TCCK, pin 31). Suppose, however, that the frequency range at the TCCK pin is less than 500 kHz. Using the reference pulse at the timer C gate input pin (TCG, pin 15), you can count the unknown clock pulses at the TCCK pin during an 8-ms interval ($1/62.5 \text{ Hz} \div 2$). The timer C count value is saved into B_TC0 and B_TC1 in hexadecimal format. The values in B_TC0 and B_TC1 are the actual frequency value, in kHz.

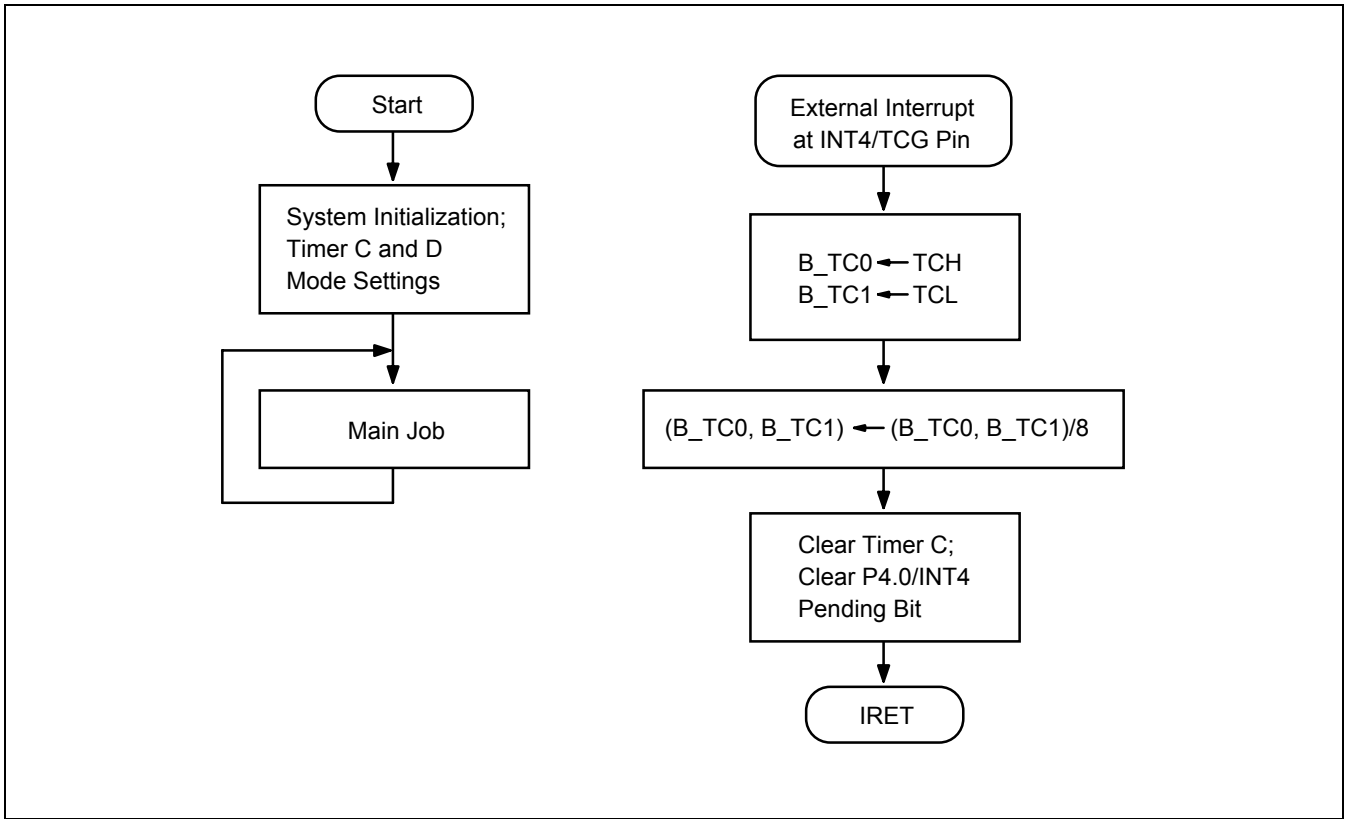


Figure 11-11. Timer Module 1 Mode 1 Programming Tip Flowchart

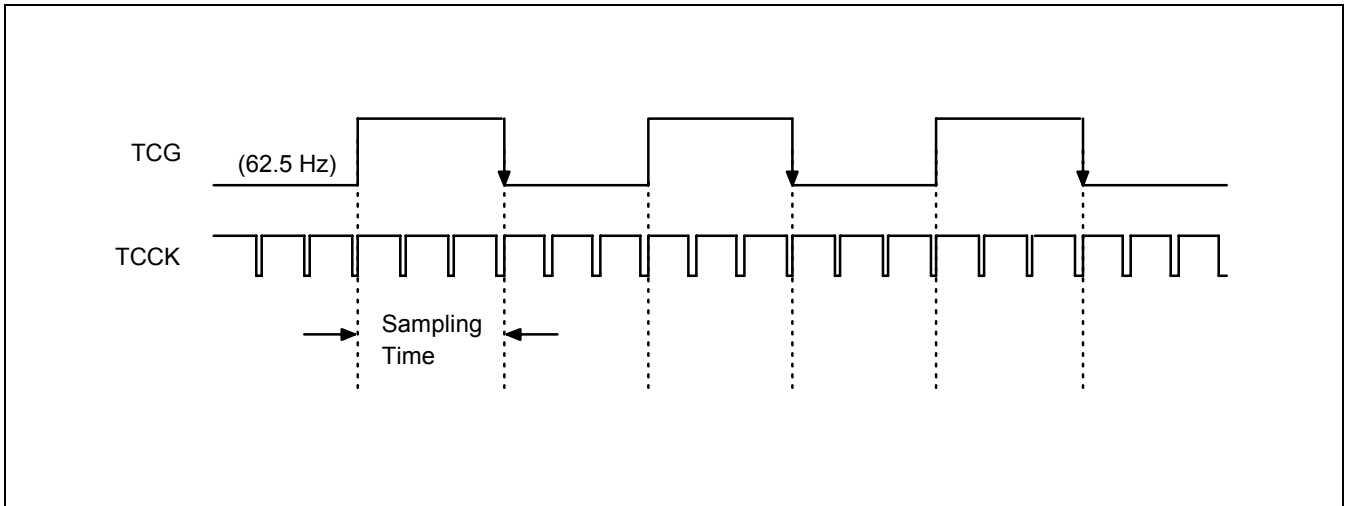


Figure 11-12. Timer Module 1 Mode 1 Timing Diagram

PROGRAMMING TIP — Timer Module 1, Operating Mode 1 (Continued)

```

B_TC0    EQU    00H                ; Timer C data buffer register
B_TC1    EQU    01H
START     DI                    ; Disable interrupts
        .
        .                        ; System initialization settings
        .
        LD      P3CONL,#00H        ; TCCK input mode select
        LD      P4CONL,#00H        ; TCG (INT4) input falling edge select
        LD      P4INT,#01H        ; P4.0 /TCG interrupt enable
        LD      P4PND,#0H         ; Clear pending bit
        LDW     TCH,#0000H         ; Initialize timer C counter
        LD      T1MOD,#3DH         ; Disable timer D, enable TCG input,
                                   ; timer clock = external clock input select,
                                   ; 16-bit counter mode select
        LD      T1CON,#01H         ; Disable timer C and D interrupt,
                                   ; timer C run enable
        EI                    ; Enable interrupts
        .
        .
        .
MAIN      NOP
        .
        .
        CALL    JOB                ; Run other job
        .
        .
        .
        JP      T,MAIN
        .
        .
        .

```

The external interrupt occurs at P4.0 /TCG in 16-ms intervals (falling edges):

```

EXTINT_TCG:
        LDW     B_TC0,TCH          ; Count value detect (two bytes)
        RCF
        RRC     B_TC0
        RRC     B_TC1              ; (B_TC0, B_TC1) ← (B_TC0, B_TC1) / 2
        RCF
        RRC     B_TC0              ; (B_TC0, B_TC1) ← (B_TC0, B_TC1) / 2
        RRC     B_TC1
        RCF
        RRC     B_TC0              ; (B_TC0, B_TC1) ← (B_TC0, B_TC1) / 2
        RRC     B_TC1              ; The value in B_TC0, B_TC1 indicates the clock
                                   ; frequency at the TCCK pin (in kHz)
        LDW     TCH,#0000H
        LD      P4PND,#0H         ; Clear the P4.0 (INT4) pending bit
        IRET

```

PROGRAMMING TIP — Timer Module 1, Operating Mode 2

This example shows how to program timer module 1 (timers C and D) to operate in auto-reload timer/counter mode (that is, in mode 2). The program parameters are:

- CPU clock frequency = 6 MHz
- Clock input pulse at the TCCK pin is 60 Hz (square wave)
- Timer C operates in auto-reload mode using external clock input
- Timer C interrupt occurs in 0.5-second intervals
- The level at P3.2 (pin 29) toggles whenever an interrupt occurs

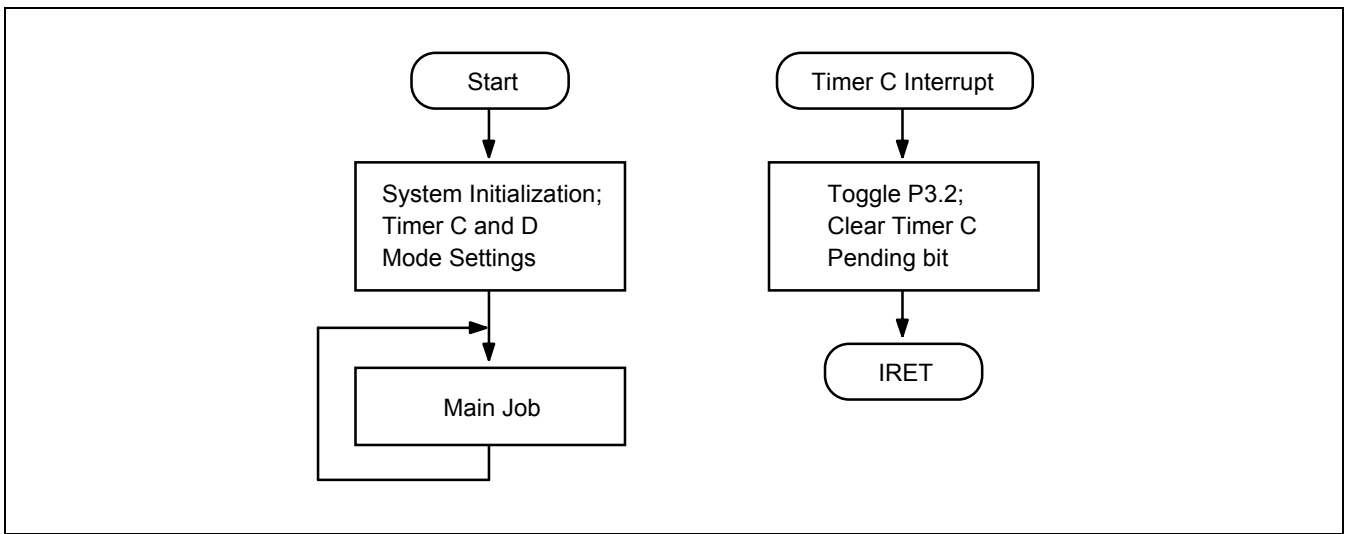


Figure 11-13. Timer Module 1 Mode 2 Programming Tip Flowchart

START	DI	; Disable interrupts
	.	
	.	; System initialization settings
	.	
LD	P3CONL,#0A0H	; P3.2, P3.3 set to output, push-pull mode
		; P3.0 (TCCK), P3.1 (TDCK) clock input select
LD	TCL,#(0FFH-1EH)	; Initial value for timer C low byte
LD	TCH,#(0FFH-1EH)	; 0.5 s = 1/60 × 30 (decimal), 30 (decimal) = 1EH
		; 1EH is the automatically reload value
LD	T1MOD,#36H	; Disable timer D
		; Select timer C auto-reload operating mode
		; Select external clock source for timer C
		; Disable timer C gate function
LD	T1CON,#05H	; Enable timer C interrupt
		; Timer C run enable
EI		; Enable interrupts
	.	
	.	
	.	

PROGRAMMING TIP — Timer Module 1, Operating Mode 2 (Continued)

```

MAIN      NOP
          .
          .
          .
          CALL      JOB                ; Run other job
          .
          .
          .
          JP        T,MAIN
          .
          .
          .

TMRC_INT:

          XOR        P3,#04H           ; P3.2 toggle
          LD         T1CON,#05H        ; Clear timer C pending bit
          IRET

```

PROGRAMMING TIP — Timer Module 1, Operating Mode 3

This example shows how to program timer module 1 to operate as two 8-bit timer/counters (that is, in mode 3). The program parameters are:

- Main oscillator frequency = 11.0592 MHz
- CLKCON = 10H (CPU clock = 5.5296 MHz)
- Clock input pulse at the TCCK pin is 60 Hz (square wave)
- Timer C operates in mode 3 (as two 8-bit timer/counters)
- Timer D operates in auto-reload mode with no interrupt function

Program Function Description

Timer C's low-byte count register (TCL) counts interrupts that occur in 0.5-second intervals. These interrupts are generated by external 60-Hz clock input through the TCCK pin.

The high-byte count register (TCH) counts interrupts generated at 278- μ s intervals. The timer D interrupt structure is used to the TCH count register overflow at a frequency of CPU clock divided by six. Timer D serves as a baud rate generator for the UART module.

The baud rate is 4.8 kHz. Timer D does not generate an interrupt. The state of the P0.0 and P0.1 pins toggles each time an interrupt service routine is start.

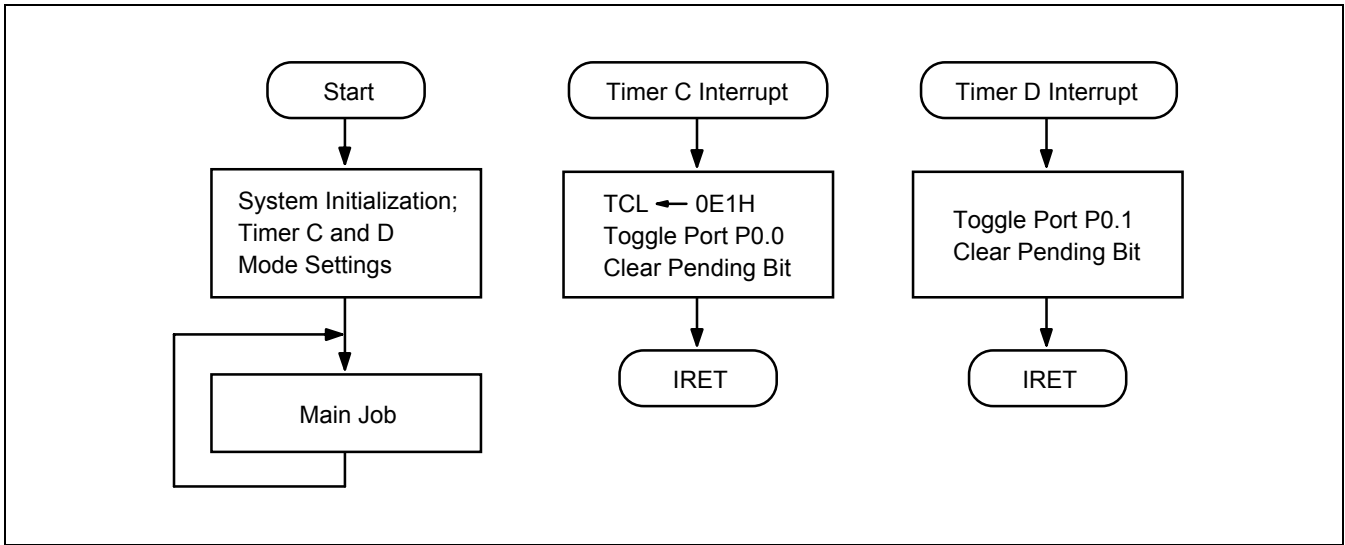


Figure 11-14. Timer Module 1 Mode 3 Programming Tip Flowchart

START	DI	; Disable interrupts
	.	
	.	; System initialization settings
	.	
	LD	P0CON,#11H ; Select port 0 output mode
	LD	P3CONL,#00H ; Select TCCK pin input mode
	LD	TCL,#0E1H ; External 60 Hz clock gives a 0.5-second interval value
	LD	TCH,#00H ; CPU clock/6 gives a 278-μs value
	LD	TDH,0FAH ; Baud rate value of 4.8 kHz with divided-by-6 CPU clock
	LD	TDL,#0FAH ;
	LD	T1MOD,#27H ; Timer D auto-reload mode (CPU clock/6)
		; Disable gate function
		; Select timer C two 8-bit timer/counter mode
		; Low-byte clock source = 60 Hz external
		; High-byte clock source = CPU clock/6
		; Disable gate function
	LD	T1CON,#0FH ; Enable timer C and D interrupts
		; Enable baud rate generator function
		; Timer C and D run enable
	EI	; Enable interrupts
	.	
	.	
MAIN	NOP	
	.	
	.	
	CALL	JOB ; Run other job
	.	
	.	
	JP	T,MAIN

 PROGRAMMING TIP — Timer Module 1, Operating Mode 3 (Continued)

The timer C interrupt is generated by a timer C low-byte counter (TCL) overflow (0.5-second intervals):

```
TMRC_INT:
    ADD    TCL,#0E1H          ; TCL ← TCL + 0E1H (0.5 seconds)
    XOR    P0,#01H           ; P0.0 toggle
    LD     T1CON,#0FH        ; Clear pending bit (bit 4)
    IRET
```

The timer D interrupt is generated by a timer C high-byte counter (TCH) overflow. The full count range of 00H–0FFH occurs once every 278 μ s, resulting in a 278- μ s interrupt interval:

```
TMRD_INT:
    XOR    P0,#02H          ; P0.1 toggle
    LD     T1CON,#0FH        ; Clear pending bit (bit 5)
    IRET
```

12

SERIAL PORT (UART)

OVERVIEW

KS88C0716 has a full-duplex serial port with programmable operating modes: There is one synchronous mode and three UART (Universal Asynchronous Receiver/Transmitter) modes:

- Serial I/O with baud rate of CPU clock/6
- 8-bit UART mode; variable baud rate
- 9-bit UART mode; CPU clock/32 or /16
- 9-bit UART mode, variable baud rate

The serial port receive and transmit buffers are accessed through the shift register, SIO (E9H). Writing to the shift register loads the transmit buffer; reading the shift register accesses a physically separate receive buffer.

The serial port is receive-buffered. Using the receive data buffer, reception of the next byte can begin before the previously received byte has been read from the receive register. If the first byte has not been read by the time the next byte has been completely received, one of the bytes is lost.

In all modes, transmission starts when any instruction uses the SIO register as its destination address.

In mode 0, serial data reception starts when the receive interrupt pending bit (RIP) in the SIOPND register is "0" and the receive enable bit (RE, SIOCON.4) is "1". In modes 1, 2, and 3, reception starts when the incoming start bit ("0") is received and the receive enable (RE) bit is set to "1".

SERIAL PORT CONTROL REGISTER (SIOCON)

The control register for the serial port is SIOCON (EAH). It has the following control functions:

- Operating mode selection
- 9th data bit location settings for transmit and receive operations (TB8, RB8)
- Multiprocessor communication and interrupt control

SERIAL PORT INTERRUPT PENDING REGISTER (SIOPND)

The serial I/O interrupt pending register SIOPND (EBH) contains the serial data transmit interrupt pending bit (TIP) and receive interrupt pending bit (RIP) in register positions SIOPND.0 and SIOPND.1, respectively.

During a serial receive operation, the receive interrupt pending flag RIP is set in mode 0 at the end of the 8th bit shift time. In the other modes, it is set halfway through the stop bit's shift time.

When the RIP flag is automatically set to "1", indicating that a receive interrupt is pending, it must then be cleared by software.

The transmit interrupt pending flag TIP is set at the end of the 8th shift bit time in mode 0, or at the beginning of the stop bit in the other modes, during any serial transmit operation.

When the TIP flag is set to "1", indicating that a transmit interrupt is pending, it must be cleared by software.

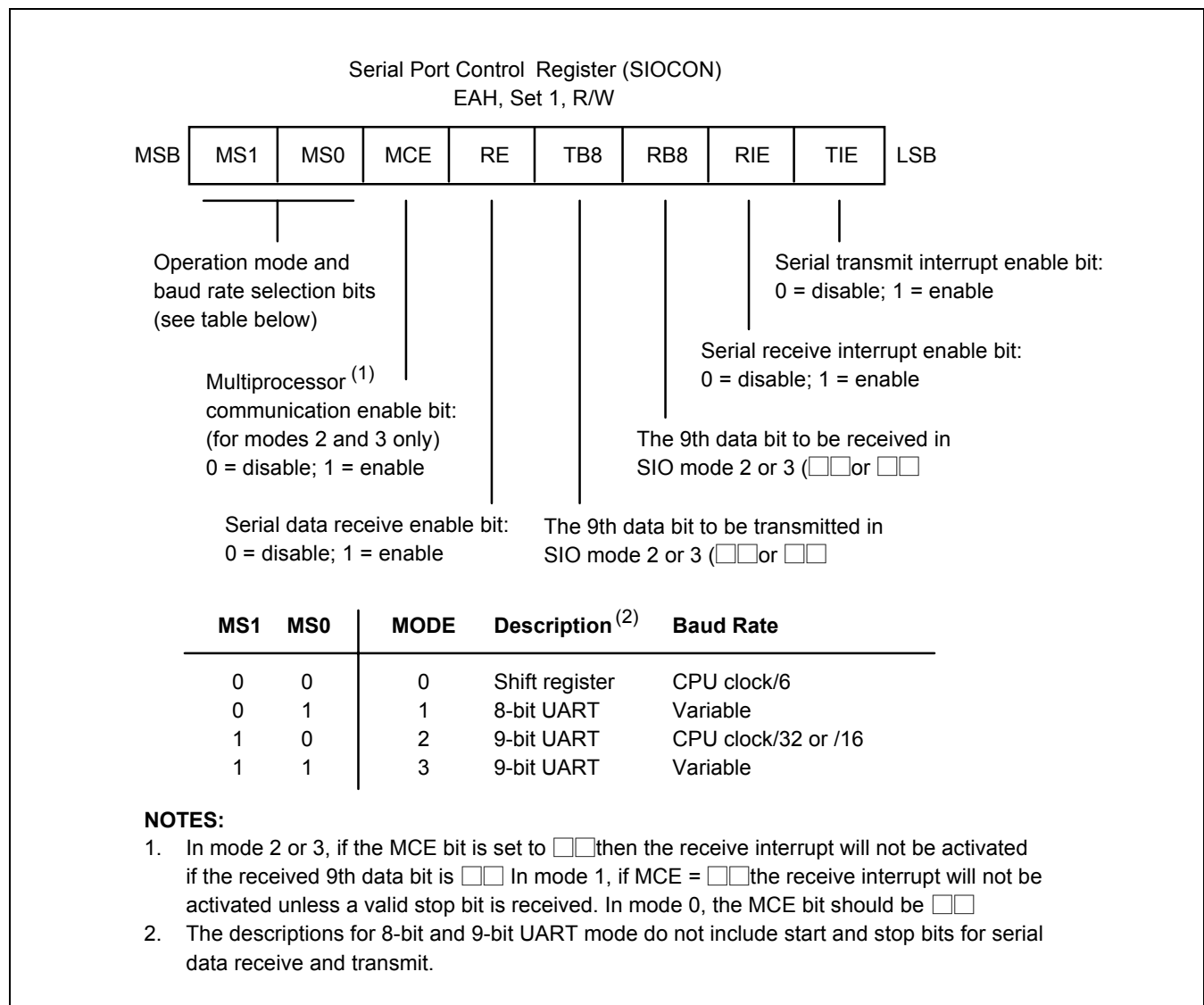
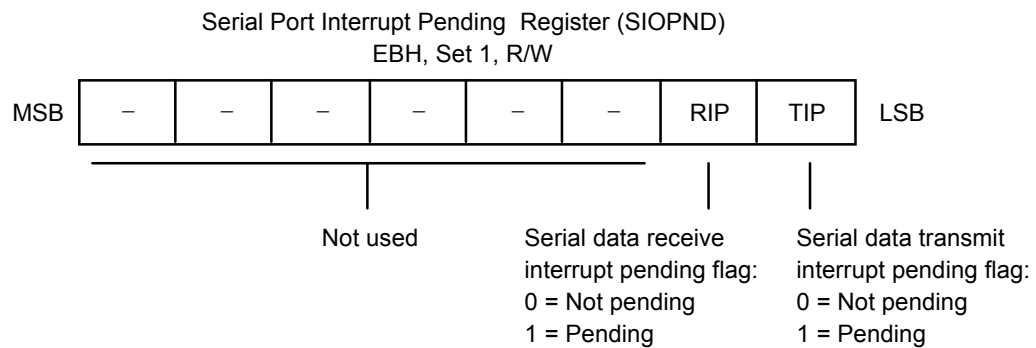


Figure 12-1. Serial Port Control Register (SIOCON)

**NOTES:**

1. In order to clear a data transmit or receive interrupt pending flag, you must write a ☐☐ to the appropriate pending bit.
To avoid possible program errors, we recommend using load.
2. Instructions only (except for LDB) to manipulate the SIOPND register.

Figure 12-2. Serial Port Interrupt Pending Register (SIOPND)

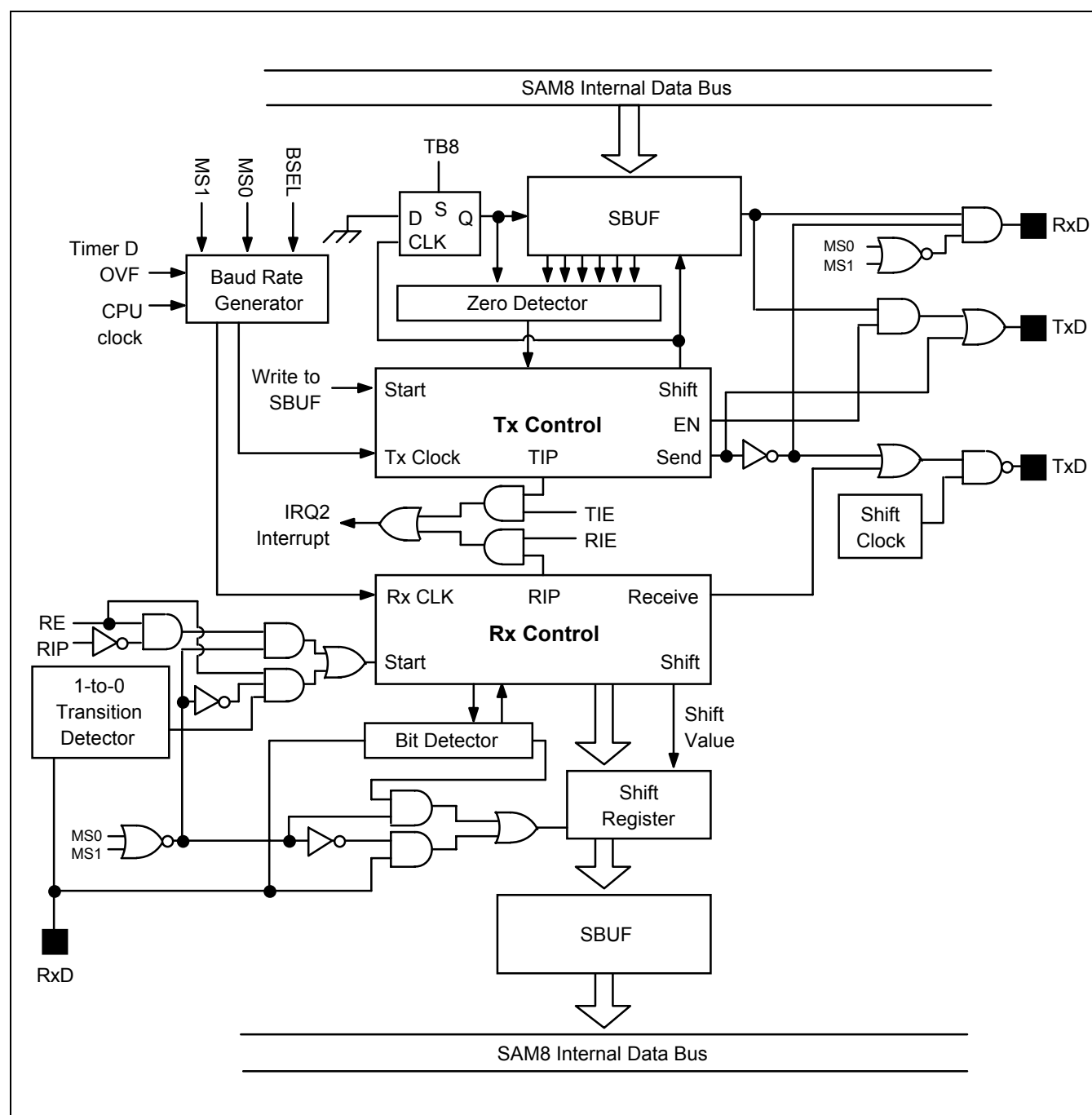


Figure 12-3. Serial Port (UART) Function Diagram

SERIAL PORT MODE 0 FUNCTION DESCRIPTION

In mode 0, serial data is input and output through the RxD pin and the TxD pin outputs the shift clock.

Data bits are transmitted or received in eight-bit units only; the LSB of the 8-bit value is transmitted (or received) first.

The baud rate for mode 0 is equal to the CPU clock frequency divided by six.

Mode 0 Transmit Procedure

1. Select mode 0 by setting SIOCON bits 6 and 7 to '00B'.
2. Write transmission data to the shift register SIO (E9H) to start the transmit operation.

Mode 0 Receive Procedure

1. Select mode 0 (shift register; CPU clock/6) by setting SIOCON bits 6 and 7 to '00B'.
2. Clear the receive interrupt pending bit (RIP, SIOPND.1) by loading a "0".
3. Set the serial data receive enable bit (RE, SIOCON.4) to "1".
4. The shift clock will now be output to the TxD pin (P3.6, pin 25) and will read the data at the RxD pin (P3.7, pin 24). Interrupts occur if the RIE bit in the SIOCON register is "1".

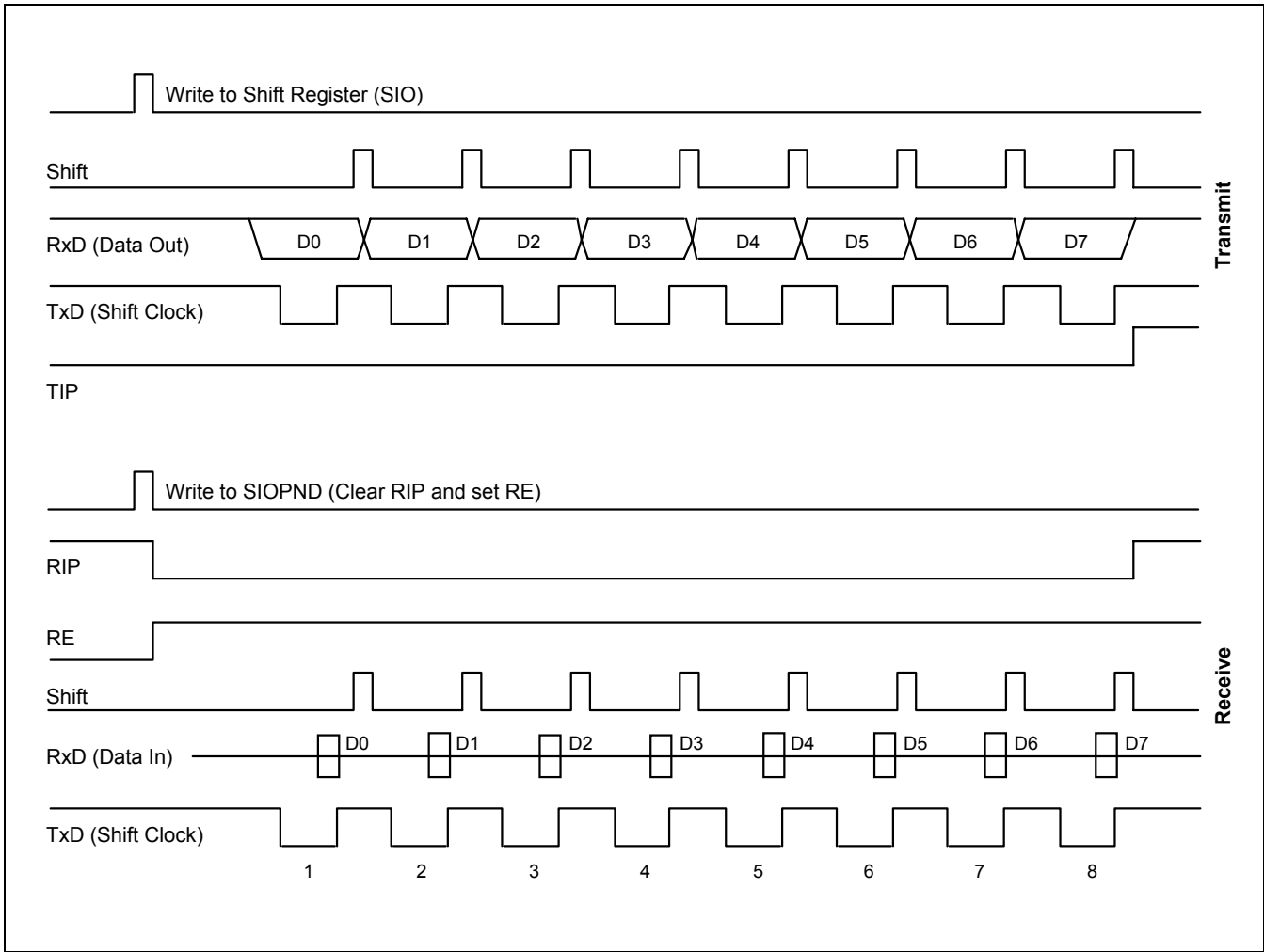


Figure 12-4. Timing Diagram for Serial Port Mode 0 Operation

SERIAL PORT MODE 1 FUNCTION DESCRIPTION

In mode 1, ten bits are transmitted (through the TxD pin) or received (through the RxD pin). Each data frame has three components:

- Start bit ("0")
- 8 data bits (LSB first)
- Stop bit ("1")

When receiving, the stop bit is written to the RB8 bit in the SIOCON register. The baud rate for mode 1 is variable.

Mode 1 Transmit Procedure

1. Select the baud rate generated by timer/counter D using the timer module 1 control register T1CON. The baud select bit (BSEL) lets you select normal or double baud rate generation for the UART module.
2. Select mode 1 (8-bit UART) by setting SIOCON bits 7 and 6 to '01B'.
3. Write transmission data to the shift register SIO (E9H). The start and stop bits are generated automatically by hardware.

Mode 1 Receive Procedure

1. Select the baud rate to be generated by timer/counter D.
2. Select mode 1 and set the RE (Receive Enable) bit in the SIOCON register to "1".
3. The start bit low ("0") condition at the RxD pin will cause the UART module to start the serial data receive operation.

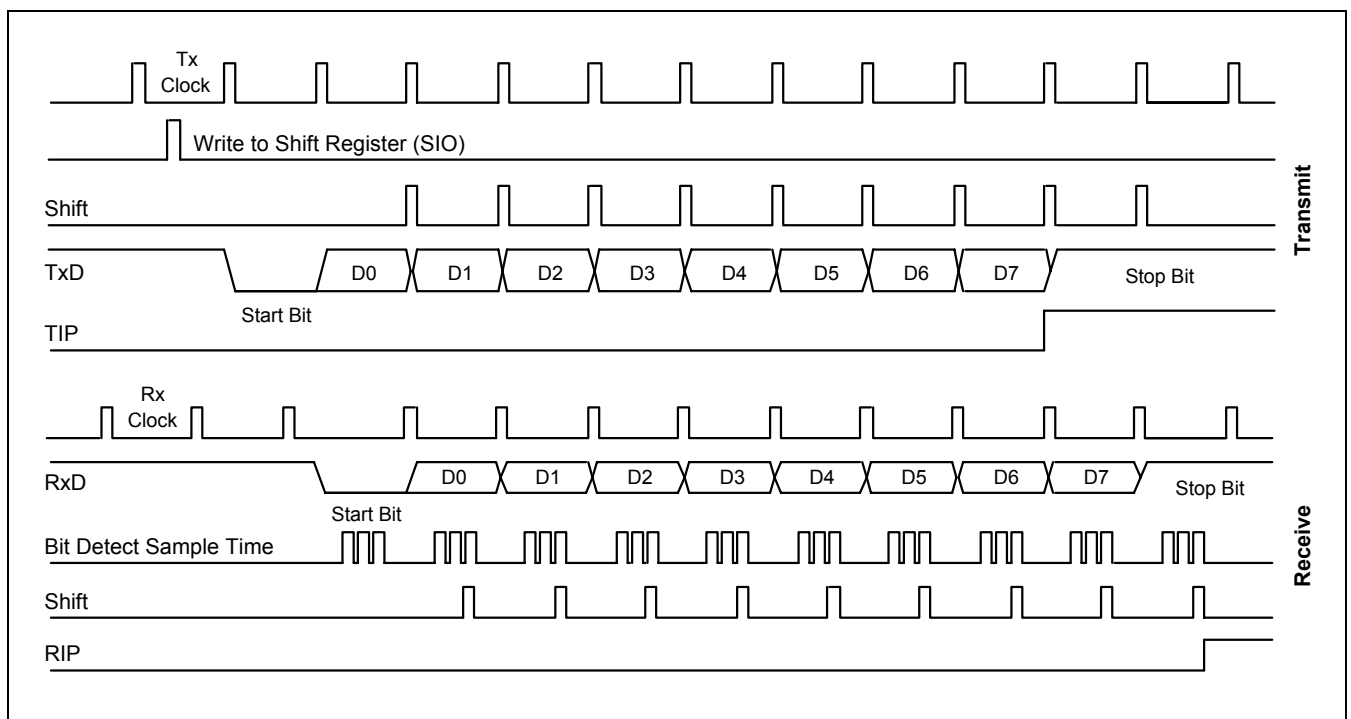


Figure 12-5. Timing Diagram for Serial Port Mode 1 Operation

SERIAL PORT MODE 2 FUNCTION DESCRIPTION

In mode 2, 11 bits are transmitted (through the TxD pin) or received (through the RxD pin). Each data frame has four components:

- Start bit ("0")
- 8 data bits (LSB first)
- Programmable 9th data bit
- Stop bit ("1")

The 9th data bit to be transmitted can be assigned a value of "0" or "1" by writing the TB8 bit (SIOCON.3). When receiving, the 9th data bit that is received is written to the RB8 bit (SIOCON.2), while the stop bit is ignored. The baud rate for mode 2 is programmable to either 1/16 or 1/32 of CPU clock frequency.

Mode 2 Transmit Procedure

1. Select mode 2 (9-bit UART) by setting SIOCON bits 6 and 7 to '10B'. Also, select the 9th data bit to be transmitted by writing TB8 to "0" or "1".
2. Select the baud rate by setting the BSEL bit in the T1CON register to "0" for normal baud or to "1" for double baud rate generation.
3. Write transmission data to the shift register, SIO (E9H), to start the transmit operation.

Mode 2 Receive Procedure

1. Select the baud rate by setting or clearing the BSEL bit in the T1CON register.
2. Select mode 2 and set the receive enable bit (RE) in the SIOCON register to "1".
3. The receive operation starts when the signal at the RxD pin goes to low level.

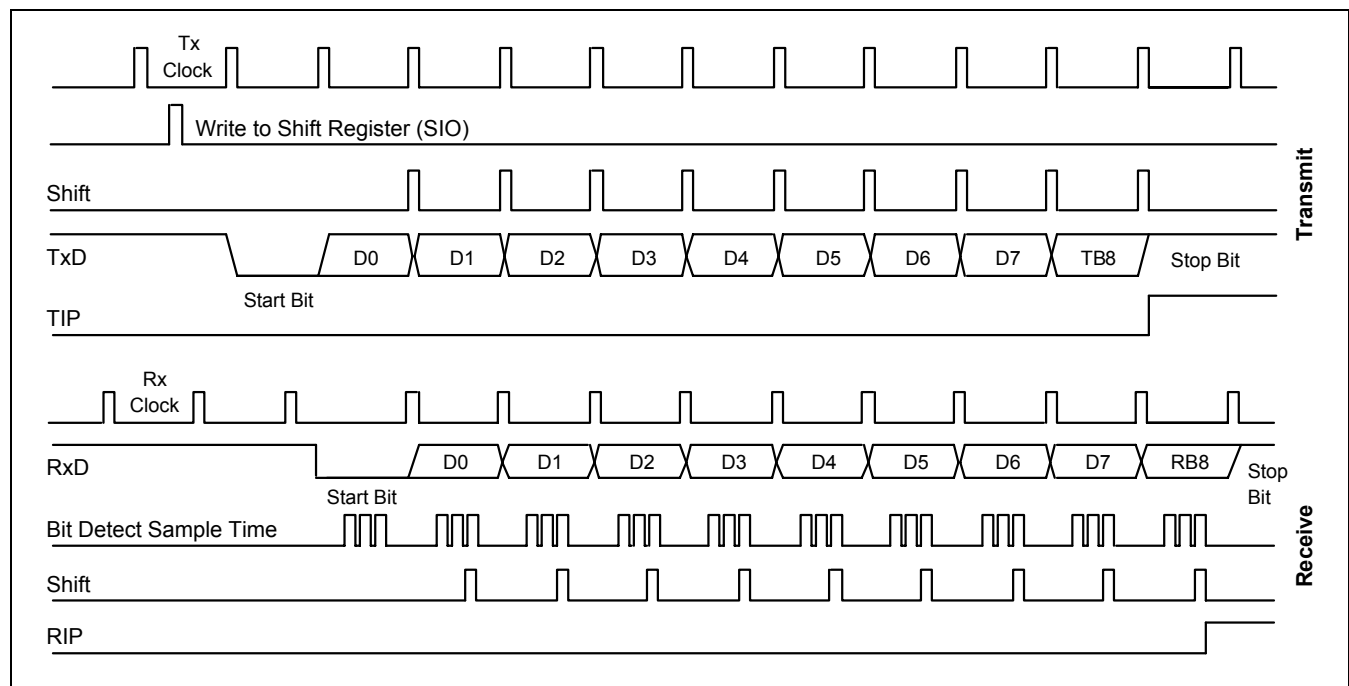


Figure 12-6. Timing Diagram for Serial Port Mode 2 Operation

SERIAL PORT MODE 3 FUNCTION DESCRIPTION

In mode 3, eleven bits are transmitted (through the TxD pin) or received (through the RxD pin). Mode 3 is identical to mode 2 except for baud rate, which is variable. Each data frame has four components:

- Start bit ("0")
- 8 data bits (LSB first)
- Programmable 9th data bit
- Stop bit ("1")

Mode 3 Transmit Procedure

1. Select the baud rate by setting the BSEL bit in the T1CON register to "0" for normal baud or to "1" for double baud rate generation. Then, enable timer D by setting the TDE bit in the T1CON register.
2. Select mode 3 operation (9-bit UART) by setting SIOCON bits 6 and 7 to '11B'. Also, select the 9th data bit to be transmitted by writing SIOCON.3 (TB8) to "0" or "1".
3. Write transmission data to the shift register, SIO (E9H), to start the transmit operation.

Mode 3 Receive Procedure

1. Select the baud rate to be generated by timer/counter D by setting or clearing the BSEL bit in the T1CON register.
2. Select mode 3 and set the RE (Receive Enable) bit in the SIOCON register to "1".
3. The receive operation will be started when the signal at the RxD pin goes to low level.

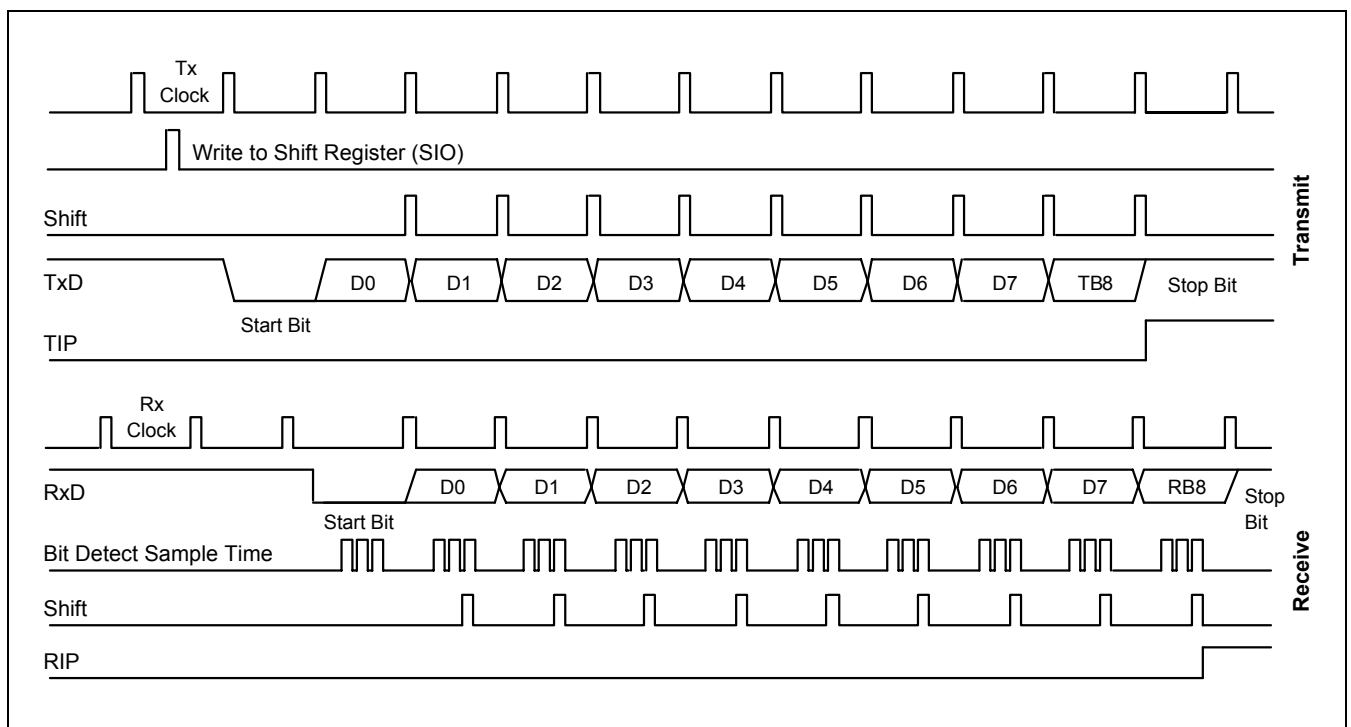


Figure 12-7. Timing Diagram for Serial Port Mode 3 Operation

BAUD RATE CALCULATIONS

Mode 0 Baud Rate Calculation

The baud rate in mode 0 is fixed at the CPU clock frequency divided by 6:

$$\text{Baud rate} = \frac{\text{CPU clock}}{6}$$

Mode 2 Baud Rate Calculation

The mode 2 baud rate depends on the value of the double baud rate select bit, BSEL (T1CON.7). If BSEL is "0" (its default value after a reset), the mode 2 baud rate is 1/32 of CPU clock frequency. If BSEL is "1", the baud rate is 1/16 of CPU clock frequency.

$$\text{Baud rate} = \frac{\text{CPU clock} \times 2^{\text{BSEL}}}{2 \times 16}$$

Modes 1 and 3 Baud Rate Calculation

When timer/counter D is used as the baud rate generator for modes 1 and 3, the baud rate is determined by the timer/counter D overflow rate and the value of the BSEL bit (T1CON.7), as follows.

$$\text{Baud rate} = \frac{\text{Timer D overflow rate} \times 2^{\text{BSEL}}}{32}$$

The timer D interrupt enable bit (TDIE, T1CON.3) should be disabled for baud generator applications. The timer itself can be configured for either "timer" or "counter" operation and any one of its three operating modes may be selected.

In most applications, timer D is configured to "timer" operation in 8-bit auto-reload mode (high nibble of T1MOD = 0010B), where baud rate is calculated by the following formula:

$$\text{Baud rate} = \frac{\text{CPU clock} \times 2^{\text{BSEL}}}{6 \times 32 \times (256 - \text{TDH})}$$

To achieve very low baud rates using timer D:

- Leave the timer D interrupt enabled,
- Configure the timer D as a 16-bit timer (set the high nibble of T1MOD to '0001B'), and
- Use the timer D interrupt to initiate a 16-bit software reload.

Table 12-1. Commonly Used Baud Rates Generated by Timer D

Baud Rate	CPU Clock	BSEL Bit	Timer D Values		
			TDC Bit	Mode	Reload Value
Mode 0; 1 MHz	6 MHz	x	x	x	x
Mode 2; 187.5 kHz	6 MHz	0	x	x	x
Mode 2; 375 kHz	6 MHz	1	x	x	x
Modes 1 and 3:					
62.5 kHz	6 MHz	1	0	2	FFH
19.2 kHz	11.059 MHz/2	1	0	2	FDH
9.6 kHz	11.059 MHz/2	0	0	2	FDH
4.8 kHz	11.059 MHz/2	0	0	2	FAH
2.4 kHz	11.059 MHz/2	0	0	2	F4H
1.2 kHz	11.059 MHz/2	0	0	2	E8H
137.5 Hz	11.986 MHz/2	0	0	2	1DH
110 Hz	3 MHz	0	0	2	72H
110 Hz	6 MHz	0	0	1	FEE3H

SERIAL COMMUNICATION FOR MULTIPROCESSOR CONFIGURATIONS

The KS88-series multiprocessor communication features lets a "master" KS88C0716/P0716 send a multiple-frame serial message to a "slave" device in a multi-KS88C0716 configuration. It does this without interrupting other slave devices that may be on the same serial line.

This feature can be used only in UART modes 2 or 3. In these modes 2 and 3, 9 data bits are received. The 9th bit value is written to RB8 (SIOCON.2). The data receive operation is concluded with a stop bit. You can program this function so that when the stop bit is received, the serial interrupt will be generated only if RB8 = "1".

To enable this feature, you set the MCE bit in the SIOCON register. When the MCE bit is "1", serial data frames that are received with the 9th bit = "0" do not generate an interrupt. In this case, the 9th bit simply separates the address from the serial data.

Sample Protocol for Master/Slave Interaction

When the master device wants to transmit a block of data to one of several slaves on a serial line, it first sends out an address byte to identify the target slave. Note that in this case, an address byte differs from a data byte: In an address byte, the 9th bit is "1" and in a data byte, it is "0".

The address byte interrupts all slaves so that each slave can examine the received byte and see if it is being addressed. The addressed slave then clears its MCE bit and prepares to receive incoming data bytes.

The MCE bits of slaves that were not addressed remain set, and they continue operating normally while ignoring the incoming data bytes.

While the MCE bit setting has no effect in mode 0, it can be used in mode 1 to check the validity of the stop bit. For mode 1 reception, if MCE is "1", the receive interrupt will be issue unless a valid stop bit is received.

Setup Procedure for Multiprocessor Communications

Follow these steps to configure multiprocessor communications:

1. Set all KS88C0716/P0716 devices (masters and slaves) to SIO mode 2 or 3.
2. Write the MCE bit of all the slave devices to "1".
3. The master device's transmission protocol is:
 - First byte: the address identifying the target slave device (9th bit = "1")
 - Next bytes: data (9th bit = "0")
4. When the target slave receives the first byte, all of the slaves are interrupted because the 9th data bit is "1". The targeted slave compares the address byte to its own address and then clears its MCE bit in order to receive incoming data. The other slaves continue operating normally.

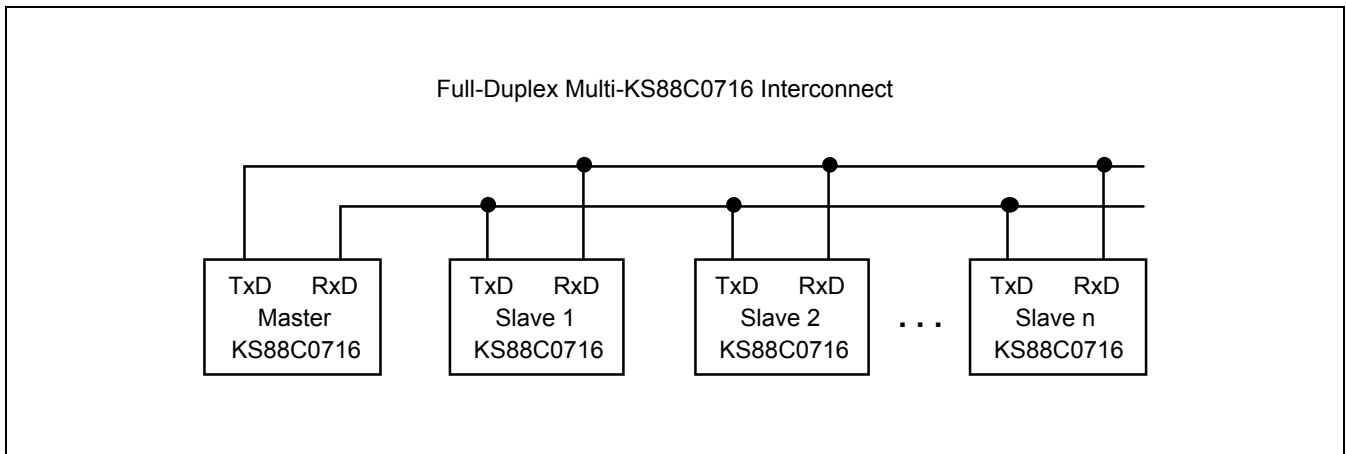


Figure 12-8. Connection Example for Multiprocessor Serial Data Communications

PROGRAMMING TIP — Programming the Serial Port for Mode 0 Operation

This example shows how to program the KS88C0716/P0716 serial port to operate in synchronous mode (mode 0). The program parameters are:

- CPU clock frequency is 6 MHz
- Device A is enabled at P0.0 (P0.0 is set to high level)
- Device B is enabled at P0.1 (P0.1 is set to high level)

Program Function Description

One-byte data is transmitted to device A and one-byte data is received from device B. The baud rate of CPU clock /6 (1 MHz) is selected. In other words, bit 7 (BSEL) in the timer module 1 control register T1CON is "0". Assume the following conditions:

- Transmit data is in the register labeled 'TRANS'.
- Data that is received from a device is loaded to register 'RECEIVE'.
- The subroutine SIO_T_R is called in 50-ms intervals.

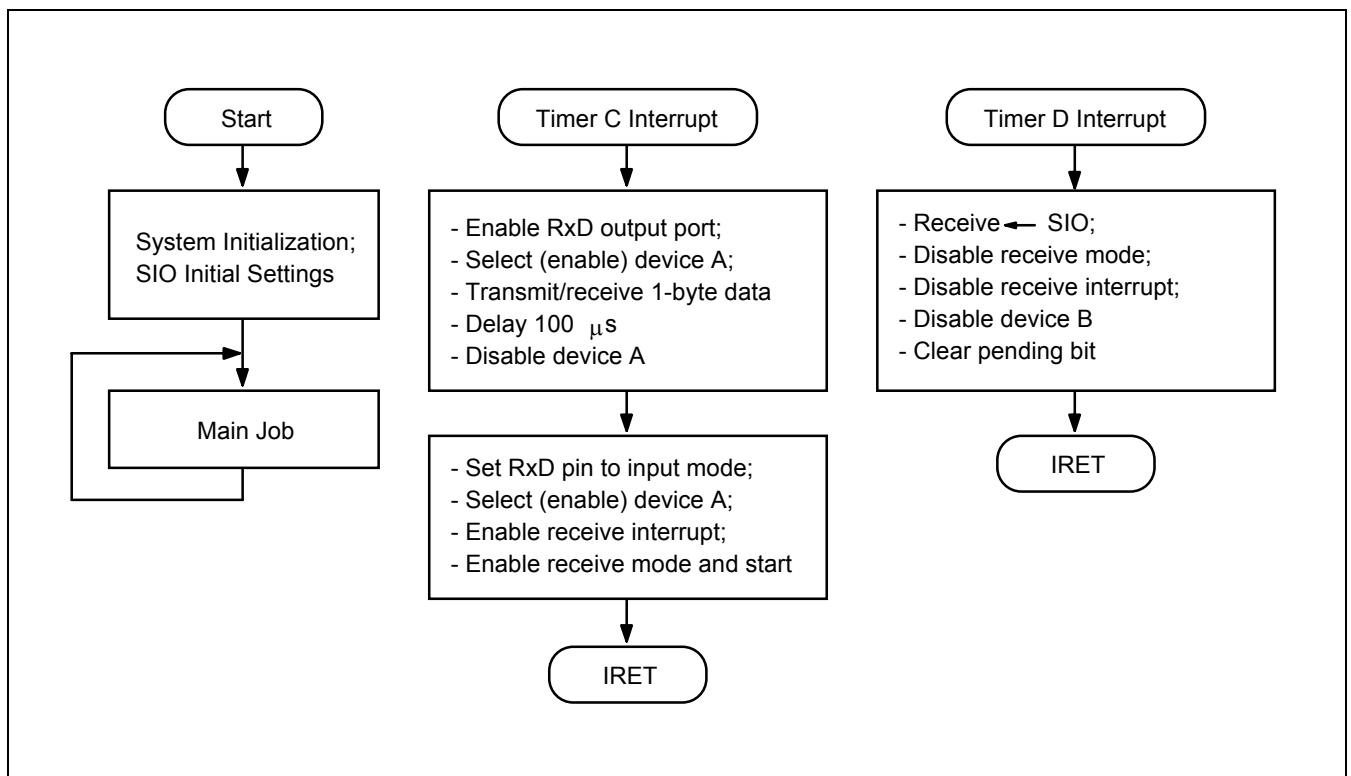


Figure 12-9. Flowchart for Serial Port Programming Tip (Mode 0)

 PROGRAMMING TIP — Programming the Serial Port for Mode 0 Operation (Continued)

```

START      DI                      ; Disable interrupts
           .                      ; System initialization settings
           .
           .
           LD      P0CON,#11H      ; Set P0 to push-pull output mode
           CLR     P0              ; Devices A and B are not selected
           LD      P3CONH,#0F0H    ; TxD /P3.6, RxD /P3.7 set to output mode
           LD      SIOCON,#02H     ; Mode 0 select; enable only the receive interrupt
                                   ; Receive disable
           AND     SIOPND,#0FDH    ; Clear pending bit
           EI                      ; Enable interrupts
           .
           .
           .
MAIN        NOP
           .
           .
           .
           CALL    JOB             ; Run other job
           .
           .
           .
           CALL    SIO_T_R         ; Run SIO subroutine
           .
           .
           .
           JP      T,MAIN
           .
           .
           .
TRANS      EQU     50H             ; Transmit data buffer
RECEIVE    EQU     51H            ; Receive data buffer
;
;
SIO_T_R    LD      P3CONH,#0F0H    ; TxD, RxD output mode select
;
           OR      P0,#01H         ; Device A is enabled (selected)
           NOP
           LD      SIO,TRANS       ; TRANS data 1-byte output
           CALL    WAIT100μs      ;
           AND     P0,#0FEH        ; Disable device A
           LD      P3CONH,#30H     ; RxD input, TxD output mode select
           NOP
           OR      P0,#02H         ; Device B is enabled (selected)
           OR      SIOCON,#10H     ; Enable receive mode and start receive operation
           RET
;

```

PROGRAMMING TIP — Programming the Serial Port for Mode 0 Operation (Concluded)

```

WAIT100 μs  PUSH      R3                ; 100-μs delay routine
              LD        R3,#57H          ;
LOOP        DJNZ      R3,LOOP
              POP        R3
              RET
;
;SIO receive interrupt service routine:
;
;
SIOINT_R    AND        P0,#0FDH          ; Disable device B
              AND        SIOCON,#0EFH    ; Receive disable
              LD         RECEIVE,SIO      ; Data restore
              LD         SIOPND,#0H      ; Clear pending bit
              IRET

```

PROGRAMMING TIP — Programming the Serial Port for Mode 3 Operation

This example shows how to program the KS88C0716/P0716 serial port to operate in 11-bit asynchronous transmit /receive mode (mode 3). Assume the following parameters:

- Main oscillator frequency = 11.0592 MHz
- Select divided by two for CPU clock (CPU clock = 11.0592/2 MHz)
- No multiprocessor communication is required
- Baud rate is 9600 BPS (see formula below)
- SIO mode 3 (11-bit, asynchronous type, is used)
- Timer/counter D overflow output is used as the SIO shift clock
- Timer C is not used in this sample program

Program Function Description

The 9600 BPS baud rate is calculated according to the following formula (CPU clock = 11.059/2 MHz, T1CON.7 = "0", and TDH = 0FDH):

$$9600 \text{ Baud} = \frac{\text{CPU clock} \propto 2^{\text{BSEL}}}{6 \propto 32 \propto (256 - \text{TDH})}$$

The subroutine SIO_T_R transmits one byte at a time. Received data is added to SR_SUM when the receive interrupt occurs.

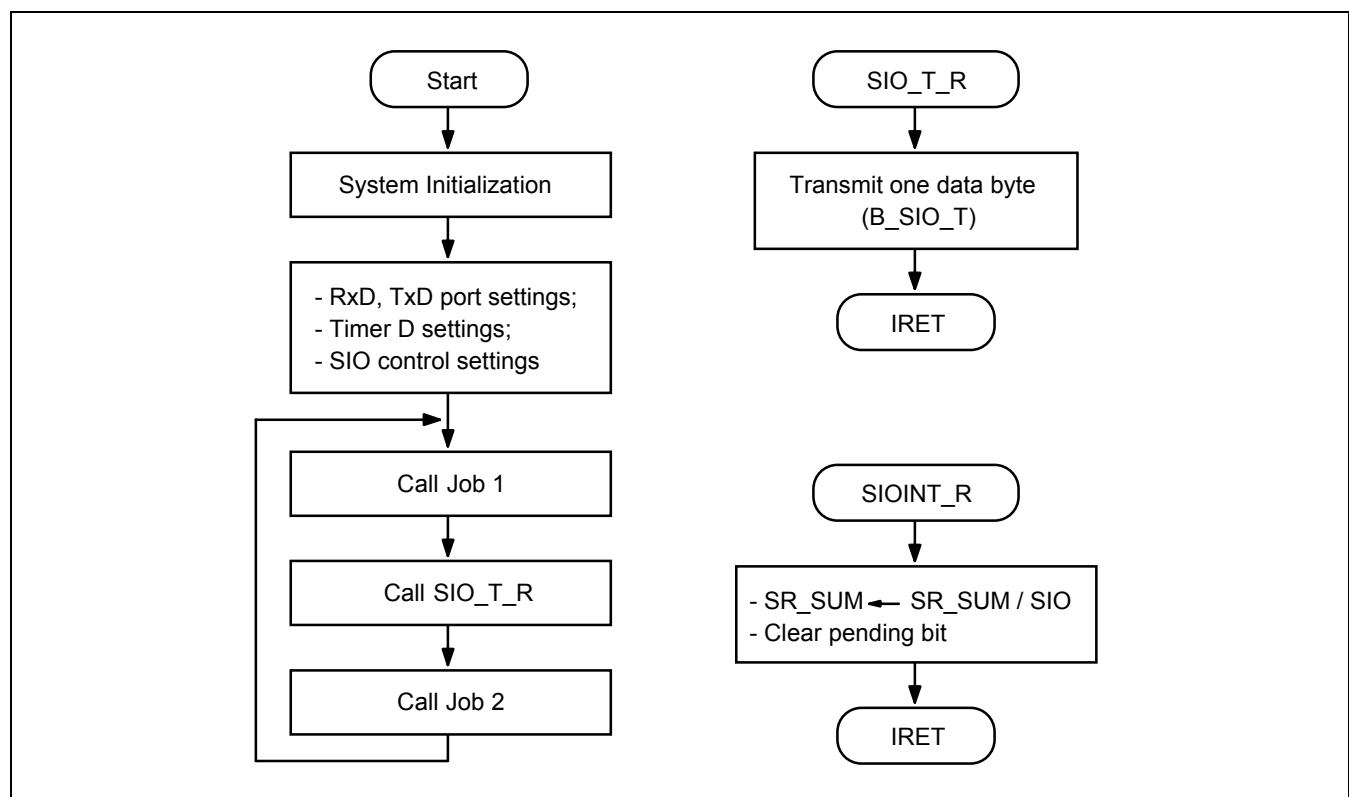


Figure 12-10. Flowchart for Serial Port Programming Tip (Mode 3)

PROGRAMMING TIP — Programming the Serial Port for Mode 3 Operation (Continued)

```

B_SIO_T    EQU    40H           ; SIO transmit buffer register
SR_SUM     EQU    41H           ; Total value of received data
.
.
.
START      DI                     ; Disable interrupts
.
.                     ; System initialization settings
.
LD          P3CONH,#3CH         ; Rx/D input, Tx/D output mode select
LD          TDL,#0FDH           ;
LD          TDH,#0FDH           ; Load the auto-reload value
LD          T1MOD,#20H          ; Disable the timer counter D gate function
.                     ; Select CPU clock /6
.                     ; Select auto-reload operating mode (mode 3)
.                     ; (Timer C is not used in this program)
LD          T1CON,#02H          ; Select normal baud rate
.                     ; Disable timer C and D interrupt
LD          SIOCON,#0D2H        ; Timer D run enable
.                     ; SIO mode 3, multiprocessing bit is low
.                     ; Receive enable; 9th transmit bit is low
.                     ; Rx/D interrupt is enabled
LD          SIOPND,#0H          ; Tx/D interrupt is disabled
.                     ; Clear the SIO pending bits
;
EI                     ; Enable interrupts
.
.
.
MAIN        NOP
.
.
CALL        JOB1              ; Run job 1
.
.
CALL        SIO_T_R            ; Run SIO transmit subroutine
CALL        JOB2              ; Run job 2
.
.
JP          T,MAIN
.
.
.

```

**PROGRAMMING TIP — Programming the Serial Port for Mode 3 Operation (Concluded)**

```
SIO_T_R    LD      SIO,B_SIO_T      ; Transmit 1-byte data
           RET
;
;       SIO receive interrupt service routine
;
SIOINT_R    ADD     SR_SUM,SIO      ; Add receive data to SR_SUM
           LD      SIOPND,#0H      ; Clear receive interrupt pending bit
           IRET
```

13

EXTERNAL INTERFACE

OVERVIEW

The KS88 architecture supports accesses to memory and other peripheral devices over an external interface. Both program and data memory areas can be accessed over the 16-bit address and data bus.

Instruction code can be fetched, or data read, from external program memory. If external program memory is implemented in a RAM-type device, you can write program code or data to this memory space.

The KS88C0716/P0716 has 64 pins, 56 of which are used for I/O. Of these 56 pins, up to 21 can alternately be configured as external interface lines. Because the address and data bus carries 16-bit memory addresses, up to 64 K bytes of memory space can be addressed. 16 K bytes of program memory are available in the on-chip ROM.

The remaining 48 K bytes of program memory can be implemented externally (or the entire 64-Kbyte program memory can be configured externally using the ROM-less operating mode).

A 64-Kbyte data memory area can also be implemented externally using the external interface. The data memory (DM) signal line is used to keep external data memory and external program memory accesses separate on the 16-bit address/data bus.

DM output remains high level whenever instructions are being fetched or when the external program memory is being accessed. DM output goes low whenever an external data memory location is addressed.

To initialize the external interface, you ports 0, 1, and 2 must be properly configured: Port 1 pins are configured as address/data bus lines AD0–AD7. These lines are multiplexed to provide address lines A0–A7 and data lines D0–D7. Port 0 pins can be configured on a nibble basis, as needed, to provide up to eight more address lines (A8–A15).

Address and data traffic on the 16-bit bus is controlled by the address strobe (AS), data strobe (DS) and read/write signal (R/W). These control lines are configured at port 2 pins P2.0–P2.2 by bit settings in the low-byte port 2 control register, P2CONL.

The external interface also has a tri-state function that is useful for multiprocessor applications.

The two system registers are also used to program the external interface: the system mode register (SYM) and the external memory timing register (EMT).

CONFIGURATION OPTIONS FOR EXTERNAL PROGRAM MEMORY

Program memory (ROM) stores program code and table data. Instructions can be fetched, or data read, from ROM locations. The KS88C0716/P0716 has 16 K bytes internal mask-programmable ROM (locations 0H–3FFFH).

You will recall that the SAM8 dual bus architecture can address up to 64 K bytes of program memory area. Using the external interface, it is possible to configure additional program memory space externally for applications that require more than the 16-Kbyte internal masked ROM. There are two ways to configure external program memory:

Option 1: Configure the remaining 48 K bytes (locations 4000H–FFFFH) externally.

Option 2: Using the ROM-less mode option, configure the entire 64-Kbyte area (0000H–FFFFH) externally.

OPTION 1: CONFIGURING THE REMAINING 48-KBYTES OF PROGRAM MEMORY EXTERNALLY

In order to access the external 48-Kbyte program memory area, the internal 16-Kbyte ROM must contain the initialization routine for configuring the external interface. The routine must also jump out of the 16-Kbyte space into the external program memory address range (4000H–FFFFH).

For this reason, the internal 16-Kbyte ROM area must always be mask-programmed.

Option 2: Using ROM-less Mode to Configure 64-Kbytes of Program Memory Externally

Since the cost of configuring 48-Kbytes or 64-Kbytes of external ROM is generally the same, option 2 will usually be chosen if external program memory is required.

To configure the entire 64-Kbyte ROM address range externally, you must configure the KS88C0716/P0716 to operate in ROM-less mode. This is done by applying 5 V to the EA pin (pin 13). You may recall that the KS88C0716/P0716 operates in normal (16-Kbyte internal ROM) mode when 0 V is applied to the EA pin.

In ROM-less mode, access to the internal ROM is disabled. A reset automatically configures the external interface lines at ports 0, 1, and 2. Please note that the 5 V must be applied to the EA pin prior to RESET and must remain at the 5-volt level during normal operation. You should not change the default settings in the port 0, 1, and 2 control registers during normal operation. Otherwise the external interface may be disabled.

If you plan to implement Option 2, the KS88C0716's internal 16-Kbyte ROM does not need to be mask-programmed.

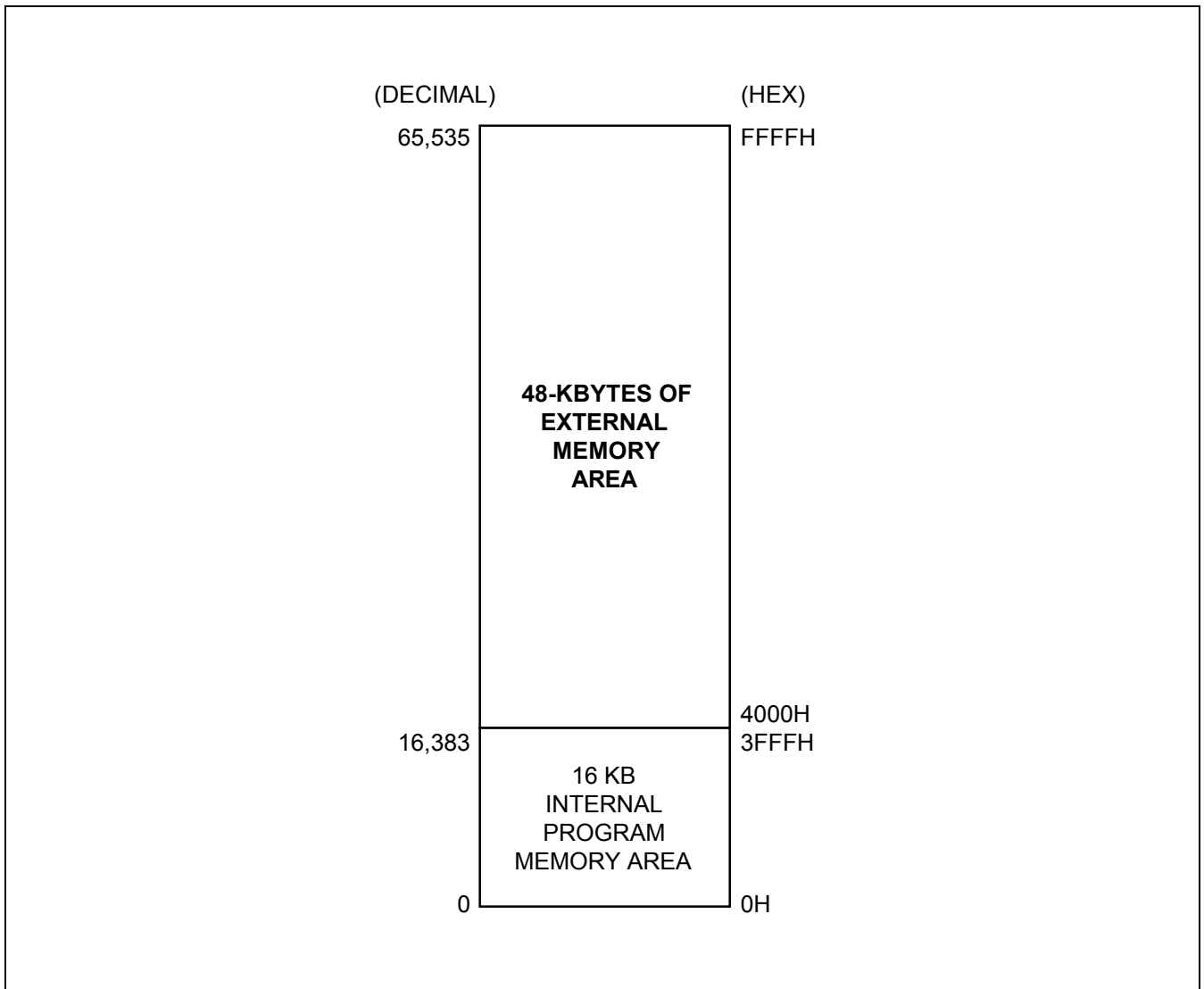


Figure 13-1. Program Memory (Normal Operating Mode)

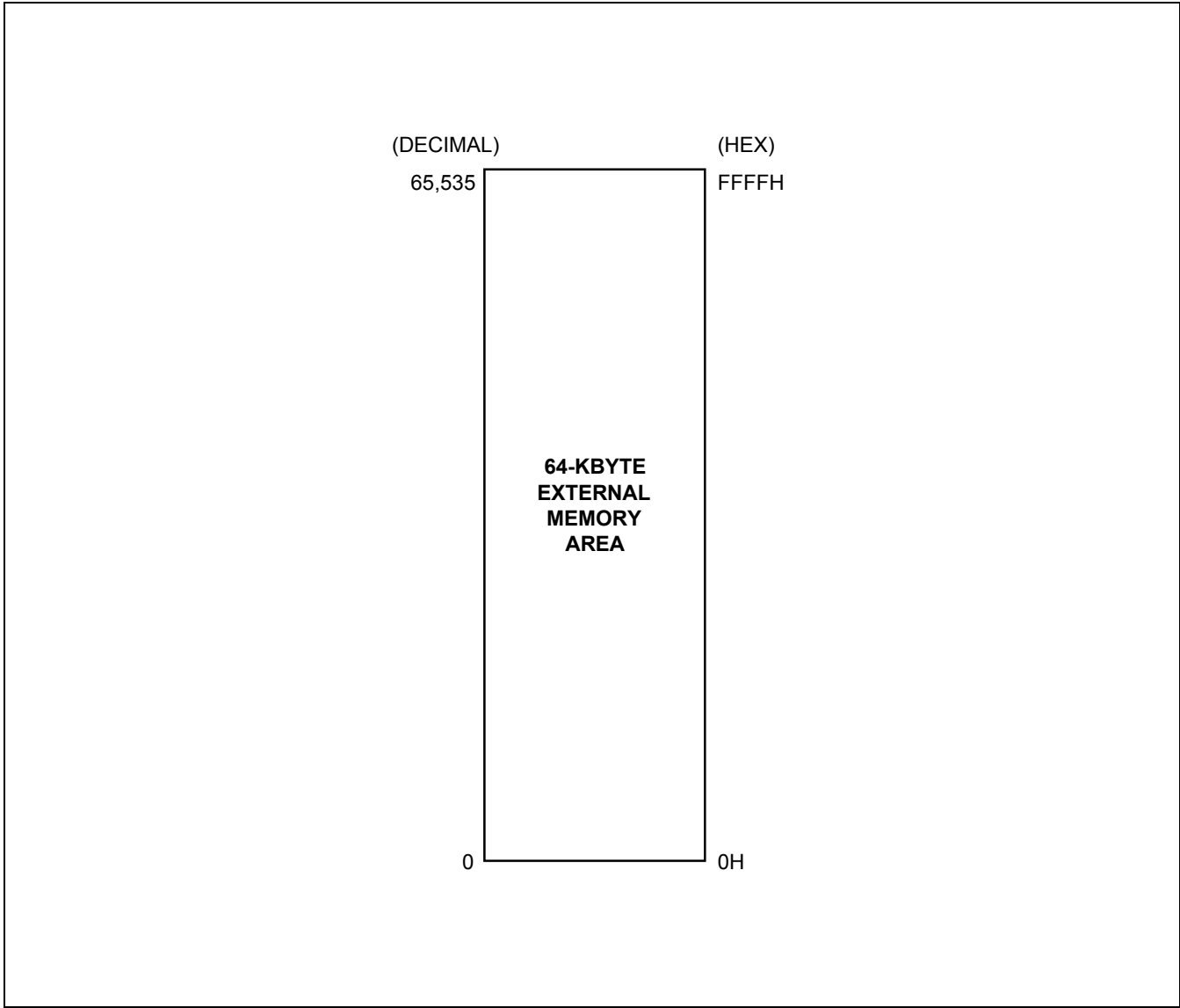


Figure 13-2. Program Memory (ROM-less Operating Mode)

SYSTEM MODE REGISTER (SYM)

The system mode register SYM controls interrupt processing and also contains the enable bit for the tri-state external memory interface (SYM 7). When the tri-state bit is enabled, the lines of the external memory interface are set to high impedance (that is, the signals "float") for use in multiprocessor applications that require a shared external bus.

EXTERNAL MEMORY TIMING REGISTER (EMT)

The external memory timing register EMT is used to control bus operations for the external interface. A reset clears the WAIT enable and stack area selection bits to "0". This disables the wait function and selects the internal register file as the system stack area. A reset also sets the program memory and data memory wait function to three wait cycles. This causes program execution time to be slowed down by a factor of two.

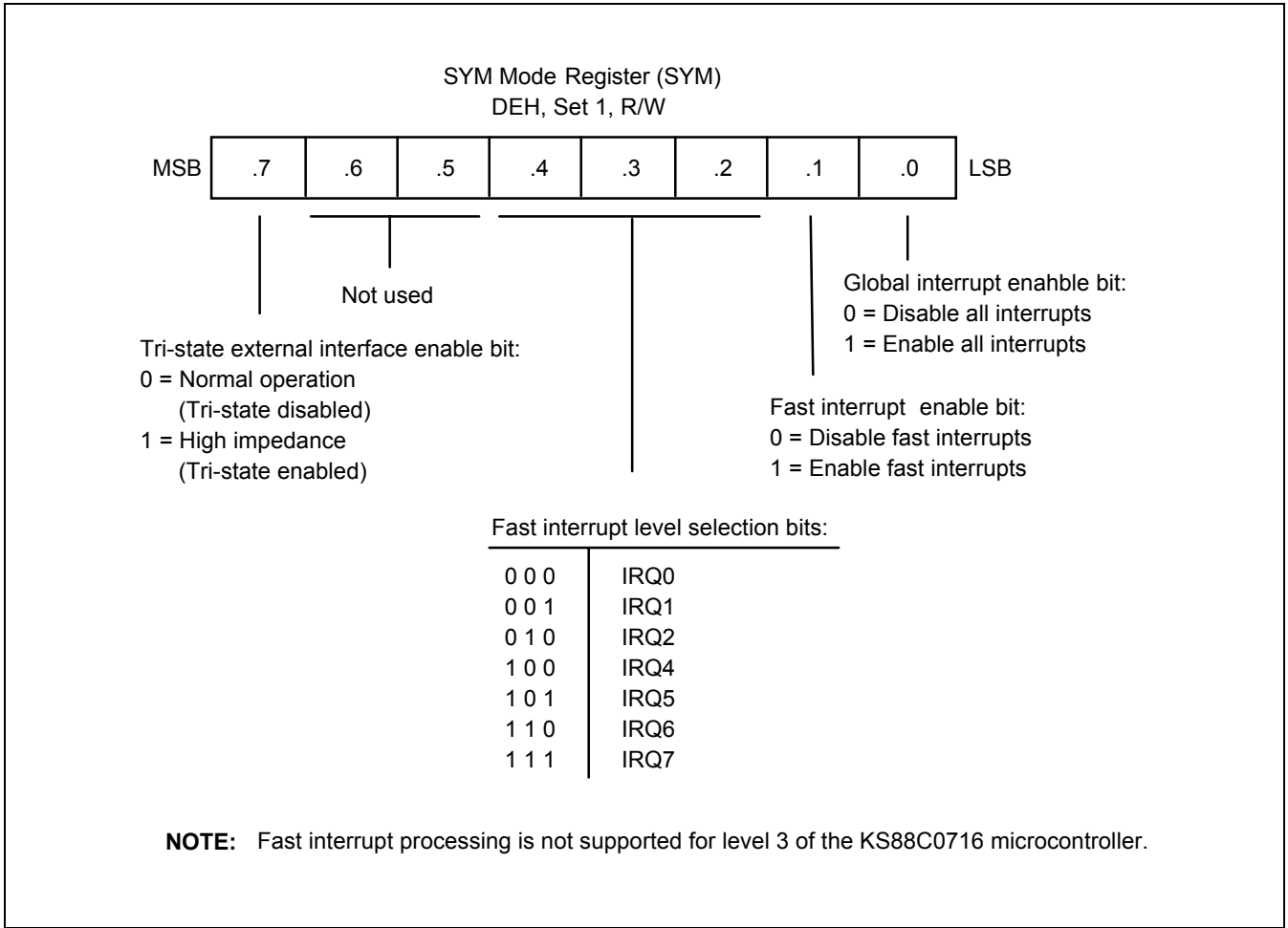


Figure 13-3. System Mode Register (SYM)

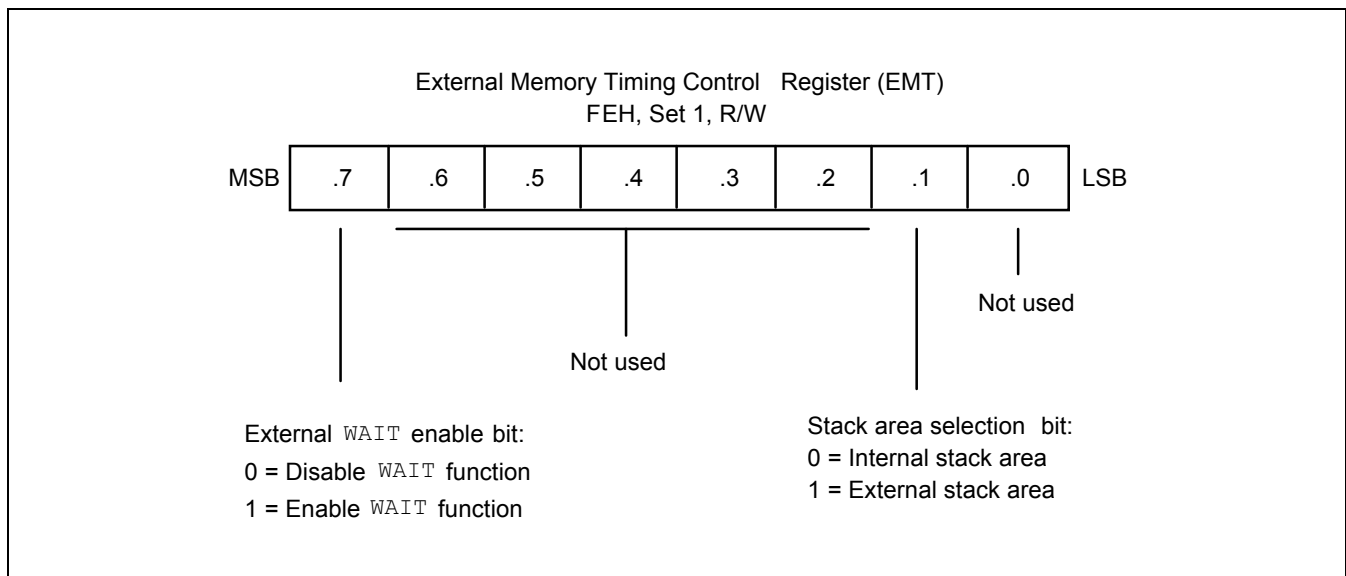


Figure 13-4. External Memory Timing Control Register (EMT)

HOW TO CONFIGURE THE EXTERNAL INTERFACE

Before you can access external memory space over the interface, ports 0 and 1 must be properly configured. In normal operating mode, this is done by the initialization routine. In ROM-less mode, ports 0 and 1 are configured automatically.

The external bus configuration uses port 1 for address/data lines AD0–AD7. The eight low-order address bits (A0–A7) are multiplexed with the eight low-order data bits (D0–D7).

Up to eight additional address lines can be configured at port 0: P0.0 corresponds to A8, P0.1 to A9, and so on through P0.7 (A15).

You can define port 0 pins as memory address lines or as standard I/O lines on a high-nibble and low-nibble basis. This is done using the port 0 control register P0CON (F0H).

If you configure port 0 as address lines A8–A15, you can no longer use the pins for general I/O. Read operations return valid port data only from those pins configured for general I/O.

The high and low nibble of port 1 can also be configured as an external memory interface by setting bit values in the port 1 control register P1CON.

You can define port 1 pins either as address and data lines or as general-purpose I/O lines. If you configure a port 1 line for the external interface, you can not use it alternately for general I/O.

CONFIGURING SEPARATE EXTERNAL PROGRAM AND DATA MEMORY AREAS

You can address external program and data memory locations as a single combined space or as two separate spaces. If program and data memory spaces are implemented separately, this separation is maintained logically using the data memory select signal (DM).

Bits 6 and 7 in the low-byte port 2 control register, P2CONL, control the DM output. To enable DM output, both bits must be set to "1". The DM pin's state goes active low to select data memory whenever one of the following instructions is executed:

- LDE (Load external data memory)
- LDED (Load external data memory and decrement)
- LDEI (Load external data memory and increment)
- LDEPD (Load external data memory with pre-decrement)
- LDEPI (Load external data memory with pre-increment)

If you set the stack area selection bit in the EMT register (EMT.1) to "1", the system stack area is configured externally. In this case, the DM signal will go active low whenever a CALL, POP, PUSH, RET, or IRET instruction is executed.

USING AN EXTERNAL SYSTEM STACK

The KS88 architecture supports stack operations in either the internal register file or in externally configured data memory. The PUSH and POP instructions support external system stack operations.

To select the external stack area option, you must set bit 1 in the external memory timing register (EMT, FEH) to "1".

NOTE

The instruction you use to modify the stack selection bit in the EMT register should not be immediately followed by an instruction that uses the stack. This could cause a program error. Also, remember to disable interrupts by executing a DI instruction before you modify the stack selection bit.

A 16-bit stack pointer value (SPH and SPL) is required for external stack operations. After a reset, the SP values are undetermined.

Return addresses for procedure calls and interrupts, as well as dynamically generated data are stored on an externally-defined stack. The contents of the PC are saved on the external stack during a CALL instruction and restored by a RET instruction. When an interrupt occurs, the contents of the PC and the

FLAGS register are saved to the external stack. These values are then restored by an IRET instruction.

Table 13-1. External Interface Control Register Values After a RESET (Normal Mode, EA=Vss)

Register Name	Mnemonic	Address		Bit Values After RESET (EA Pin is Low)								
		Dec	Hex	7	6	5	4	3	2	1	0	
System Mode Register	SYM	222	DEH	0	–	–	x	x	x	0	0	
Port 0 Control Register	P0CON	240	F0H	0	0	0	0	0	0	0	0	
Port 1 Control Register	P1CON	241	F1H	0	0	0	0	0	0	0	0	
Port 2 Control Register (Low Byte)	P2CONL	243	F3H	0	0	0	0	0	0	0	0	
External Memory Timing Register	EMT	254	FEH	0	1	1	1	1	1	0	0	

NOTE: A dash (–) indicates that the bit is not used or not mapped; an 'x' means that the value is undefined after a RESET.

Table 13-2. External Interface Control Register Values After a RESET (ROM-less Mode, EA=VDD)

Register Name	Mnemonic	Address		Bit Values After RESET (EA Pin is High)							
		Dec	Hex	7	6	5	4	3	2	1	0
Port 2 Control Register (Low Byte)	P2CONL	243	F3H	1	1	1	1	1	1	1	1

NOTE: In ROM-less operating mode, a reset initializes all external interface control registers to their normal reset values, with the exception of P2CONL. All P2CONL bits are set to logic one (FFH). This automatically configures port 2 pins P2.0–P2.3 as bus control signal lines for the external interface.

EXTERNAL BUS OPERATIONS

The number of machine cycles that are required for external memory operations may vary from 6 to 12 external clock cycles, depending on the type of operation being performed.

The notation used to describe basic timing periods in Figures 68–73 are machine cycles (Mn), timing states (Tn), and clock periods. All timing references are made with respect to the address strobe (AS) and the data strobe (DS). The clock waveform is shown for clarification only and does not have a specific timing relationship to the other signals.

Controlling External Bus Operations

Whenever the KS88C0716/P0716 external peripheral interface is active, the addresses of all internal program memory references (and the AS signal) will also appear on the external bus. This should have no effect on the external system, however, because the DS signal is always high. (DS goes low only during external memory references.)

Shared Bus Feature

The AS, DS, DM, and R/W signals, port 1, and port 0 pins (if configured as additional address lines), can be set to high impedance to enable the KS88C0716/P0716 to share common resources with other bus masters. This feature is often required for multiprocessor or related applications that require two or more devices to share the same external bus.

The tri-state memory interface enable bit in the system mode register (SYM.7) controls this function. When SYM.7 = "1", the tri-state function is enabled, all external interface lines are set to high impedance, and the external bus is put under software control.

EXTENDED BUS TIMING FEATURES

The KS88C0716/P0716 accommodates slow external memory access and cycle times using the following methods:

- Externally-issued wait signal

These methods give you flexibility in choosing the type of external memory devices to be used for your application.

Externally-Issued Wait Feature

You can use an external device to influence timing of KS88C0716/P0716 external memory accesses. You do this by stretching the data strobe (DS) by one additional cycle for an indefinite period of time. To initiate this additional "stretch" cycle, an external device must issue a WAIT signal to the KS88C0716/P0716. The P2.4 pin accepts the WAIT input from the external source. WAIT input is sampled on each internal clock.

The WAIT input can be used in conjunction with automatic wait cycles, slow memory timing, or both. The minimum data strobe period will be determined first of all by how you apply these other features.

The WAIT signal stretches the data strobe by one additional internal clock cycle for as long as the external device continues to hold the P2.4 pin to low level.

ADDRESS AND DATA STROBES

Address Strobe (AS)

All external memory transactions start when the CPU drives the address strobe (AS) active low and then sends it high. The rising edge of the address strobe validates the read/write signal (R/W), the data memory signal (DM), and the addresses that are output at ports 0 and 1.

Addresses output at port 1 remain valid only during the T1 phase of a machine cycle and typically need to be latched using AS. Port 0 address outputs remain stable through the entire machine cycle (T1–T3).

You can also configure an address strobe (AS) for P2.0 by setting the P2CONL control register bits 1 and 0 to '11B'.

Data Strobe (DS)

The KS88C0716/P0716 uses the data strobe (DS) to time the actual transfer of data over the external memory interface.

For a write operation (R/W = "0"), a DS (low-level) signal indicates that valid data is on the port 1 AD0–AD7 lines. For a read operation (when R/W = "1") the address/data bus is placed in a high impedance state before the data strobe is sent active low so that the addressed device can put its data on the bus. The CPU then samples this data before sending the data strobe high.

You can also configure a data strobe (DS) for P2.1 by setting the P2CONL control register bits 3 and 2 to '11B'.

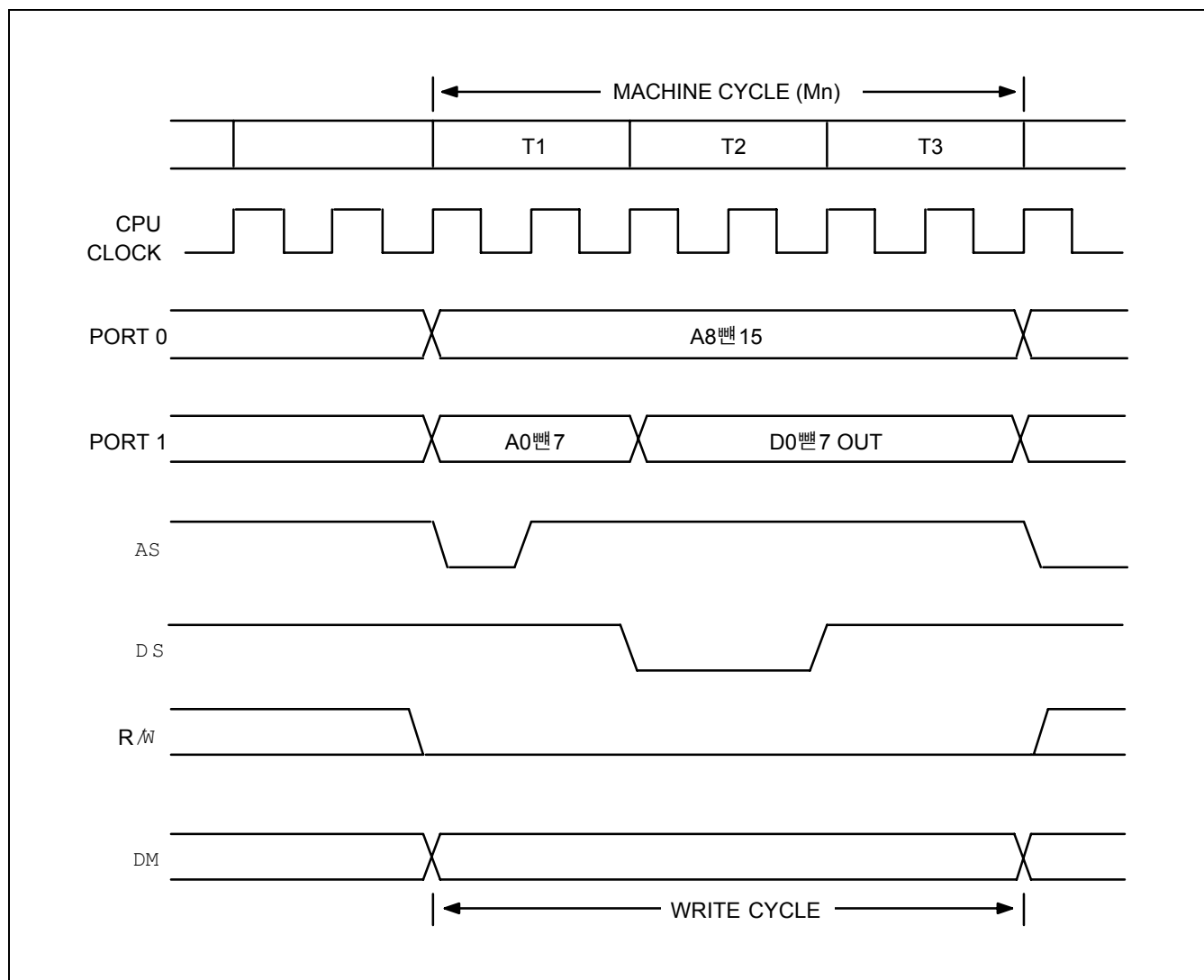


Figure 13-5. KS88C0716 External Bus Write Cycle Timing Diagram (No wait)

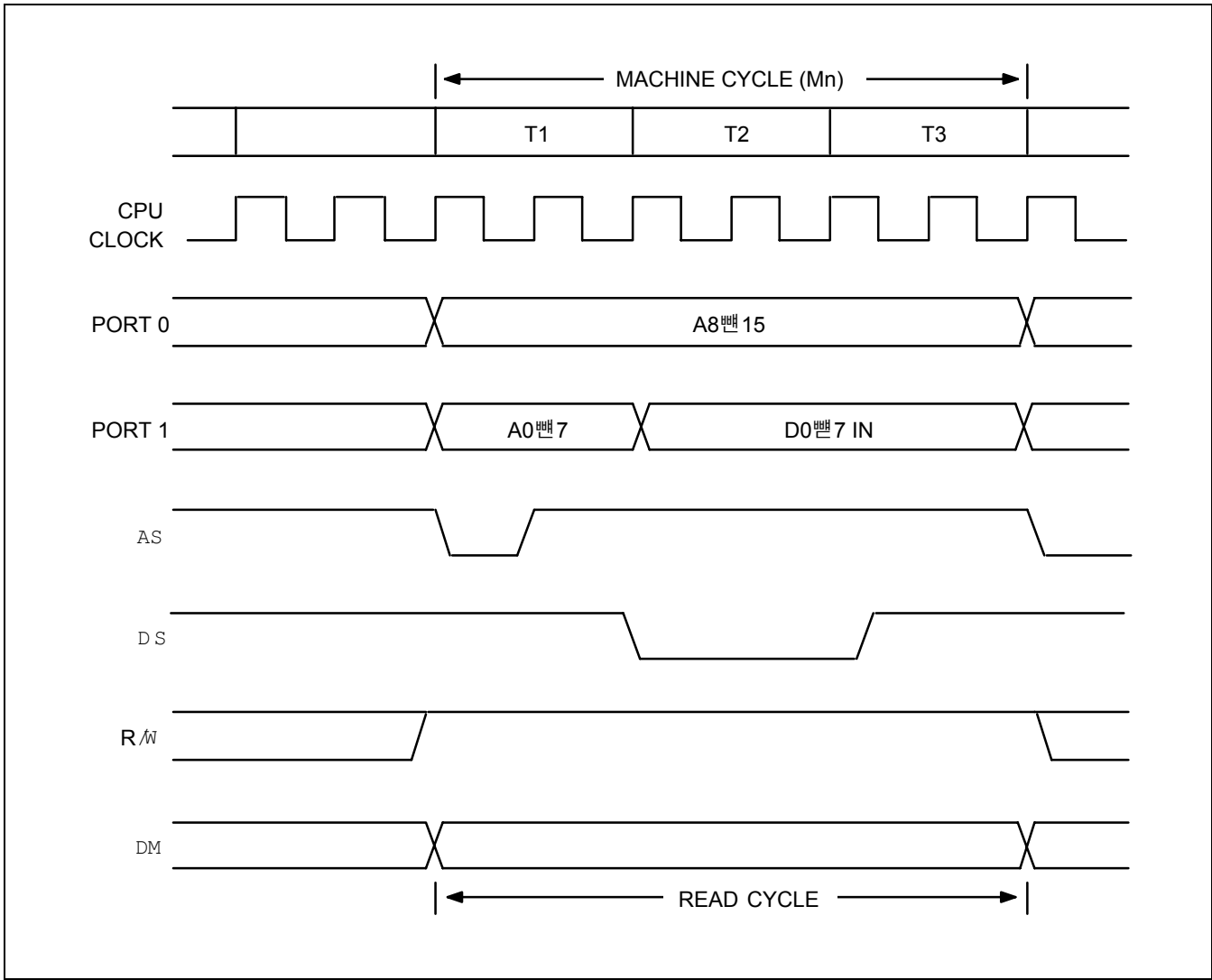


Figure 13-6. KS88C0716 External Bus Read Cycle Timing Diagram (No wait)

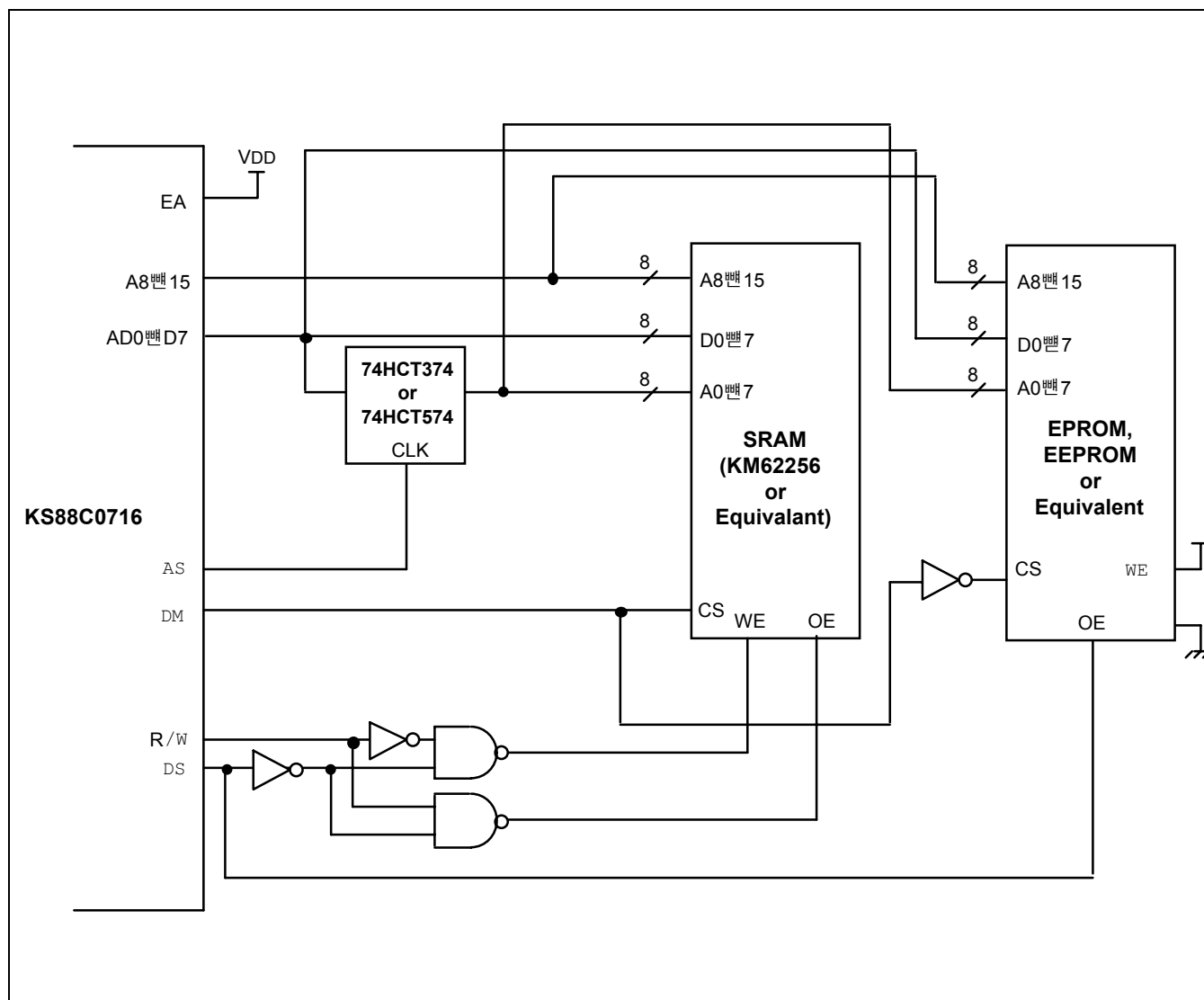


Figure 13-7. External Interface Function Diagram (with SRAM and EPROM or EEPROM)

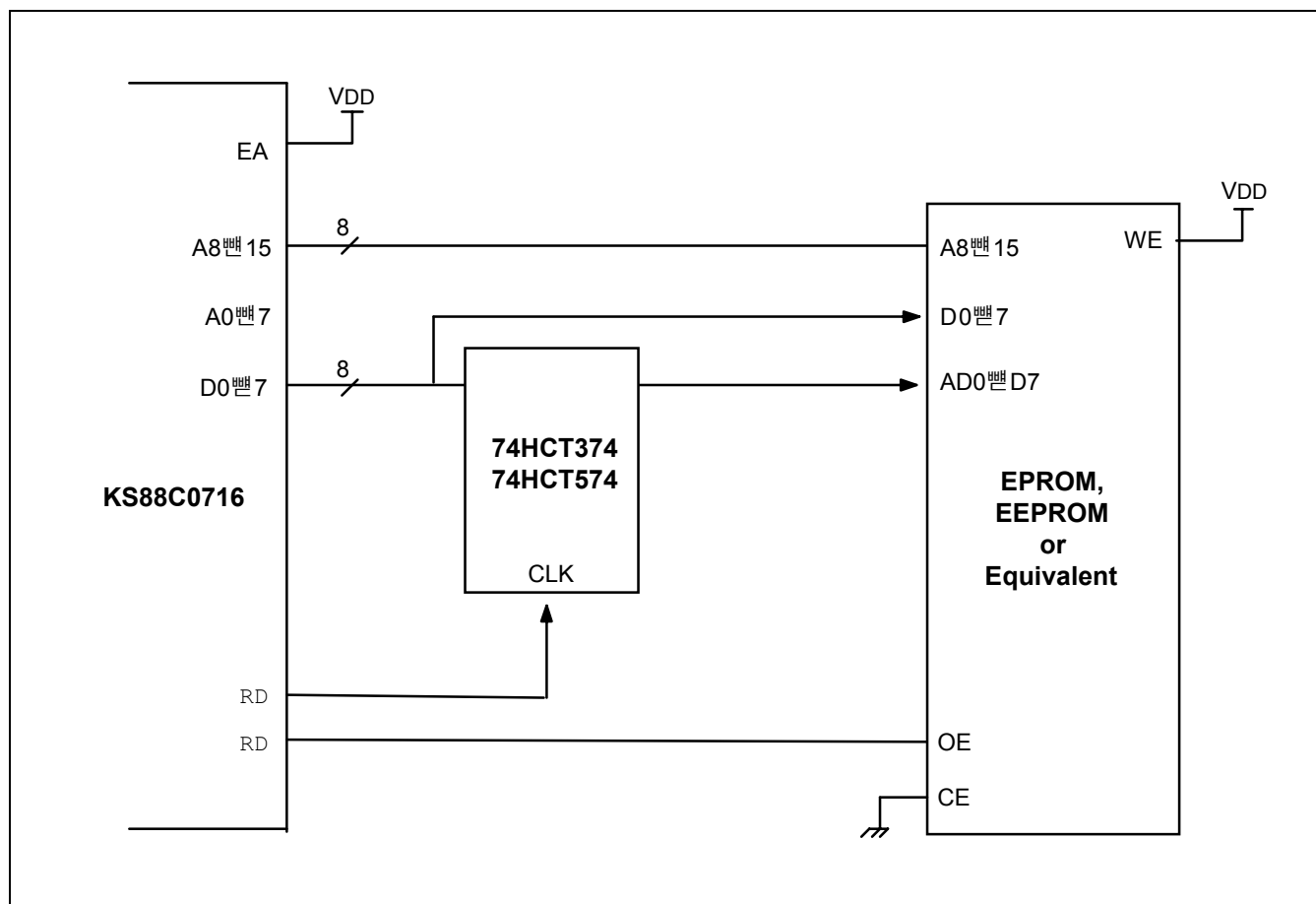


Figure 13-8. External Interface Function Diagram (External ROM Only)

14

ELECTRICAL DATA

OVERVIEW

In this section, KS88C0716 electrical characteristics are presented in tables and graphs. The information is arranged in the following order:

- Absolute maximum ratings
- D.C. electrical characteristics
- I/O capacitance
- A.C. electrical characteristics
- Oscillation characteristics
- Oscillation stabilization time

Table 14-1. Absolute Maximum Ratings

(T_A = 25°C)

Parameter	Symbol	Conditions	Rating	Unit
Supply voltage	V _{DD}		– 0.3 to + 6.5	V
Input voltage	V _I	All ports (in input mode)	– 0.3 to V _{DD} + 0.3	
Output voltage	V _O	All ports (in output mode)	– 0.3 to V _{DD} + 0.3	V
Output current high	I _{OH}	One I/O pin active	– 10	mA
		All I/O pins active	– 60	
Output current low	I _{OL}	One I/O pin active	+ 30	mA
		Total pin current for ports 0–4	+ 100	
		Total pin current for ports 5 and 6	+ 100	
Operating temperature	T _A		– 40 to + 85	°C
Storage temperature	T _{STG}		– 65 to + 150	°C

Table 14-2. D.C. Electrical Characteristics

(T_A = –40°C to +85°C, V_{DD} = 2.7 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input high voltage	V _{IH1}	All input pins except V _{IH2}	0.8 V _{DD}	–	V _{DD}	V
	V _{IH2}	X _{IN}	V _{DD} – 0.5			
Input low voltage	V _{IL1}	All input pins except V _{IL2}	–	–	0.2 V _{DD}	V
	V _{IL2}	X _{IN}			0.4	
Output high voltage	V _{OH1}	V _{DD} = 4.5 V to 5.5 V I _{OH} = –4 mA Port 5, 6	V _{DD} – 1.0	–	–	V
	V _{OH2}	V _{DD} = 4.5 V to 5.5 V I _{OH} = –1 mA All output pins except port 5, 6				
Output low voltage	V _{OL1}	V _{DD} = 4.5 V to 5.5 V I _{OL} = 15 mA Ports 5 and 6	–	–	1.0	V
	V _{OL2}	I _{OL} = 2 mA Ports 0–4			0.4	
Input high leakage current	I _{LIH1}	V _{IN} = V _{DD} All input pins except X _{IN} , X _{OUT}	–	–	3	μA
	I _{LIH2}	V _{IN} = V _{DD} , X _{IN} , X _{OUT}			20	
Input low leakage current	I _{LIL1}	V _{IN} = 0 V All input pins except X _{IN} , X _{OUT}	–	–	–3	μA
	I _{LIL2}	V _{IN} = 0 V, X _{IN} , X _{OUT}			–20	
Output high leakage current	I _{LOH}	V _{OUT} = V _{DD} All output pins	–	–	5	μA
Output low leakage current	I _{LOL}	V _{OUT} = 0 V	–	–	–5	μA
Pull-up resistor	R _{L1}	V _{IN} = 0 V; V _{DD} = 5 V Ports 0, 1, 4, 5 and 6	30	47	70	KΩ
	R _{L2}	V _{IN} = 0 V; V _{DD} = 5 V RESET only	110	210	310	

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

(T_A = –40°C to +85°C, V_{DD} = 2.7 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Supply current ⁽¹⁾	I _{DD1} ⁽²⁾	V _{DD} = 5 V ± 10% 12-MHz oscillation	–	12	25	mA
		4-MHz oscillation		4.5	10	
		V _{DD} = 3 V ± 10% 12-MHz oscillation		6	15	
		4-MHz oscillation		2.5	7	
	I _{DD2} ⁽²⁾	Idle mode; V _{DD} = 5 V ± 10% 12-MHz oscillation		3	10	
		4-MHz oscillation		1.5	4	
		Idle mode; V _{DD} = 3 V ± 10% 12-MHz oscillation		1.2	3	
		4-MHz oscillation		0.6	1.5	
	I _{DD3}	Stop mode: V _{DD} = 5 V ± 10%		0.1	3	μA

NOTES:

- Supply current does not include current drawn through internal pull-up resistors or external output current loads.
- At supply current, the CPU clock frequency is same with oscillation frequency (CPU use non divided clock).

Table 14-3. 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}	Stop mode	2	–	6	V
Data retention supply current	I _{DDDR}	Stop mode, V _{DDDR} = 2.0 V	–	–	3	μA

NOTES:

- During the oscillator stabilization wait time (t_{WAIT}), all CPU operations must be stopped.
- Supply current does not include drawn through internal pull-up resistors and external output current loads.

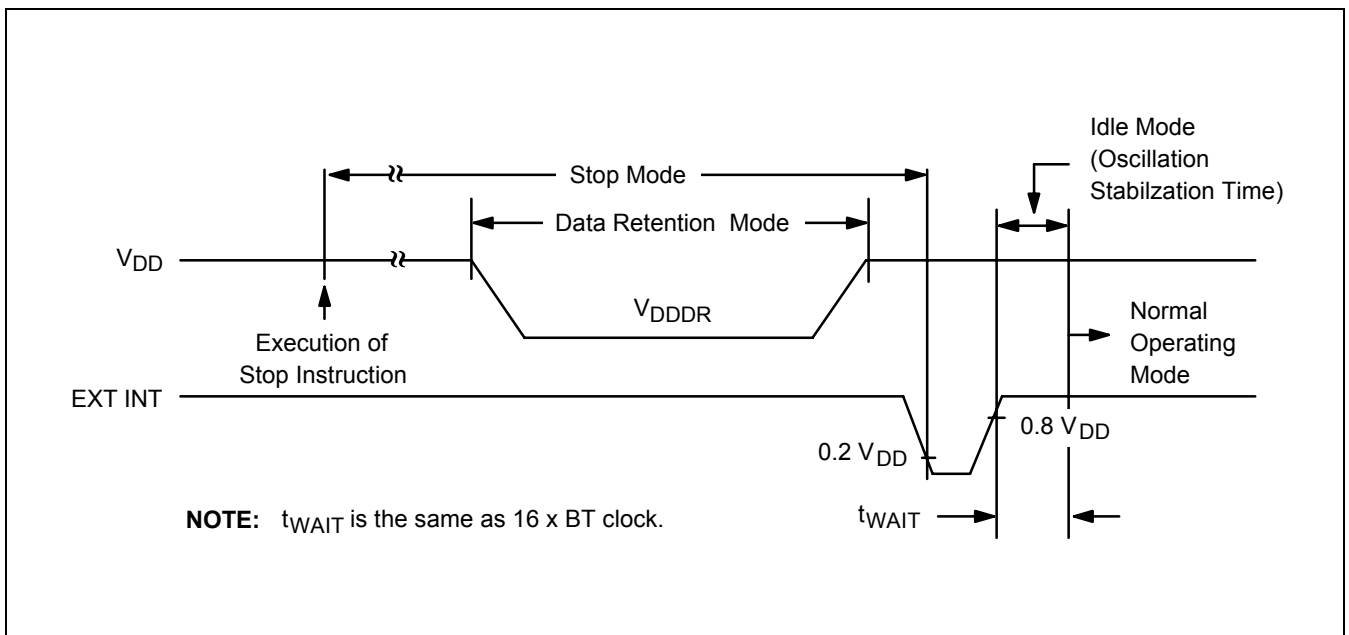


Figure 14-1. Stop Mode Release Timing When Initiated by an External Interrupt

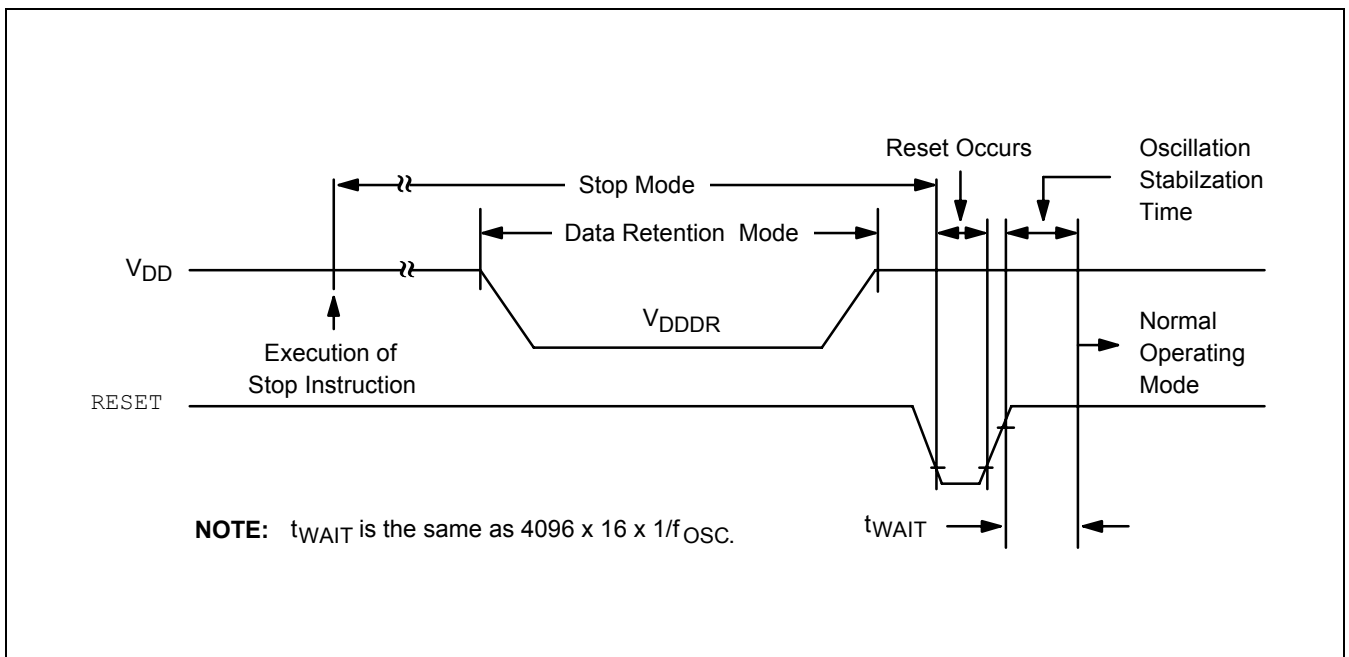


Figure 14-2. Stop Mode Release Timing When Initiated by a Reset

Table 14-4. Input/Output Capacitance

(T_A = -40°C to +85°C, V_{DD} = 0 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input capacitance	C _{IN}	f = 1 MHz; unmeasured pins are connected to V _{SS}	-	-	10	pF
Output capacitance	C _{OUT}					
I/O capacitance	C _{IO}					

Table 14-5. A.C. Electrical Characteristics

(T_A = -40°C to +85°C, V_{DD} = 2.7 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Interrupt input high, low width	t _{INTH} , t _{INTL}	P2.4-P2.7	100	-	-	ns
		P4.0-P4.7	100			
RESET input low width	t _{RSL}	Input	10	-	-	μs

NOTE: User must keep the larger value with the min value.

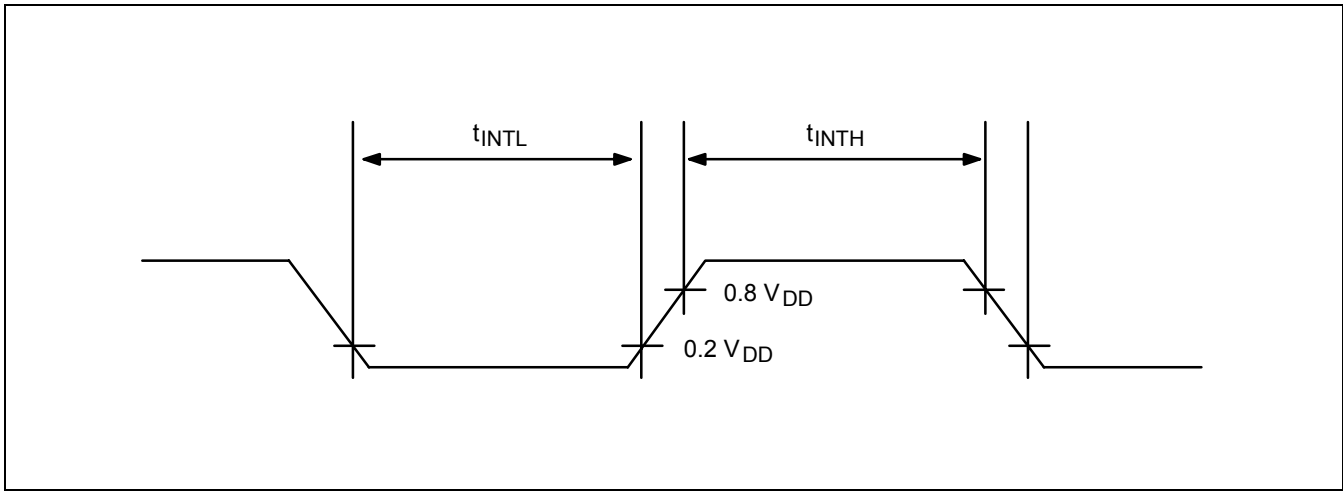


Figure 14-3. Input Timing for External Interrupts (Port 2 and 4)

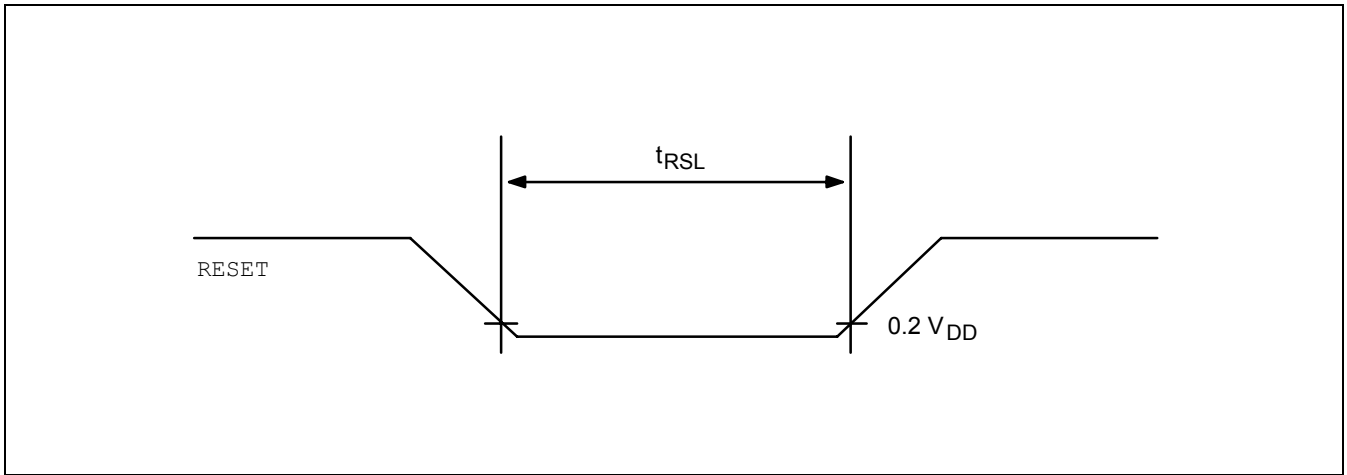


Figure 14-4. Input Timing for RESET

Table 14-6. Oscillation Characteristics

($T_A = -20^{\circ}C + 85^{\circ}C$, $V_{DD} = 4.5\text{ V to } 5.5\text{ V}$)

Oscillator	Clock Circuit	Test Condition	Min	Typ	Max	Unit
Crystal		Oscillation frequency	1	—	22.1184	MHz
Ceramic		Oscillation frequency	1	—	22.1184	MHz
External clock		X_{IN} input frequency	1	—	22.1184	MHz

Table 14-7. Main Oscillator Clock Stabilization Time (t_{ST1})

(T_A = - 20°C + 85°C, V_{DD} = 4.5 V to 5.5 V)

Oscillator	Test Condition	Min	Typ	Max	Unit
Crystal	V _{DD} = 4.5 V to 5.5 V	–	–	20	ms
Ceramic	Stabilization occurs when V _{DD} is equal to the minimum oscillator voltage range.	–	–	10	ms

NOTE: Oscillation stabilization time (t_{ST1}) is the time required for the CPU clock to return to its normal oscillation frequency after a power-on occurs, or when Stop mode is released by a RESET signal.

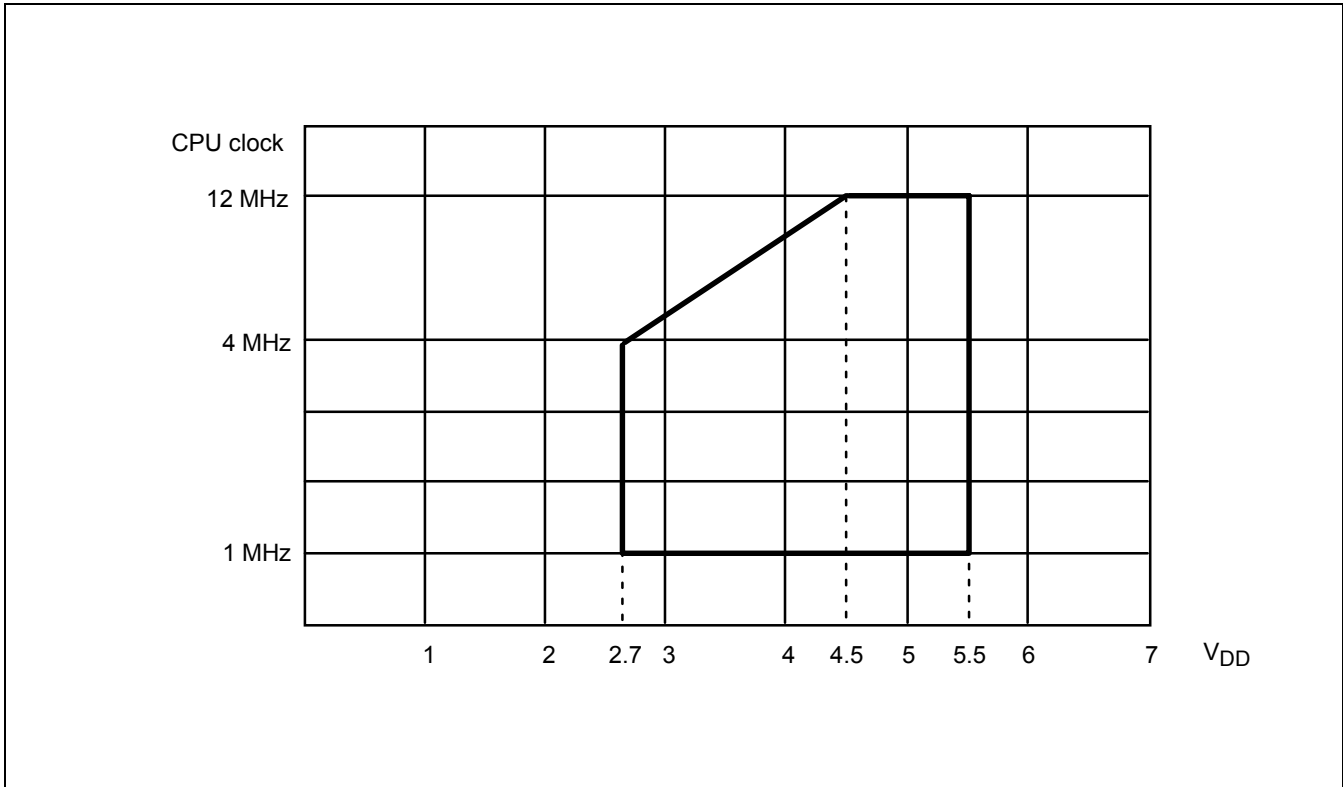


Figure 14-5. Frequency vs. Voltage

15

MECHANICAL DATA

OVERVIEW

The KS88C0716 microcontroller is available in a 64-pin SDIP package (64-SDIP-750) and a 64-pin QFP package (64-QFP-1420F).

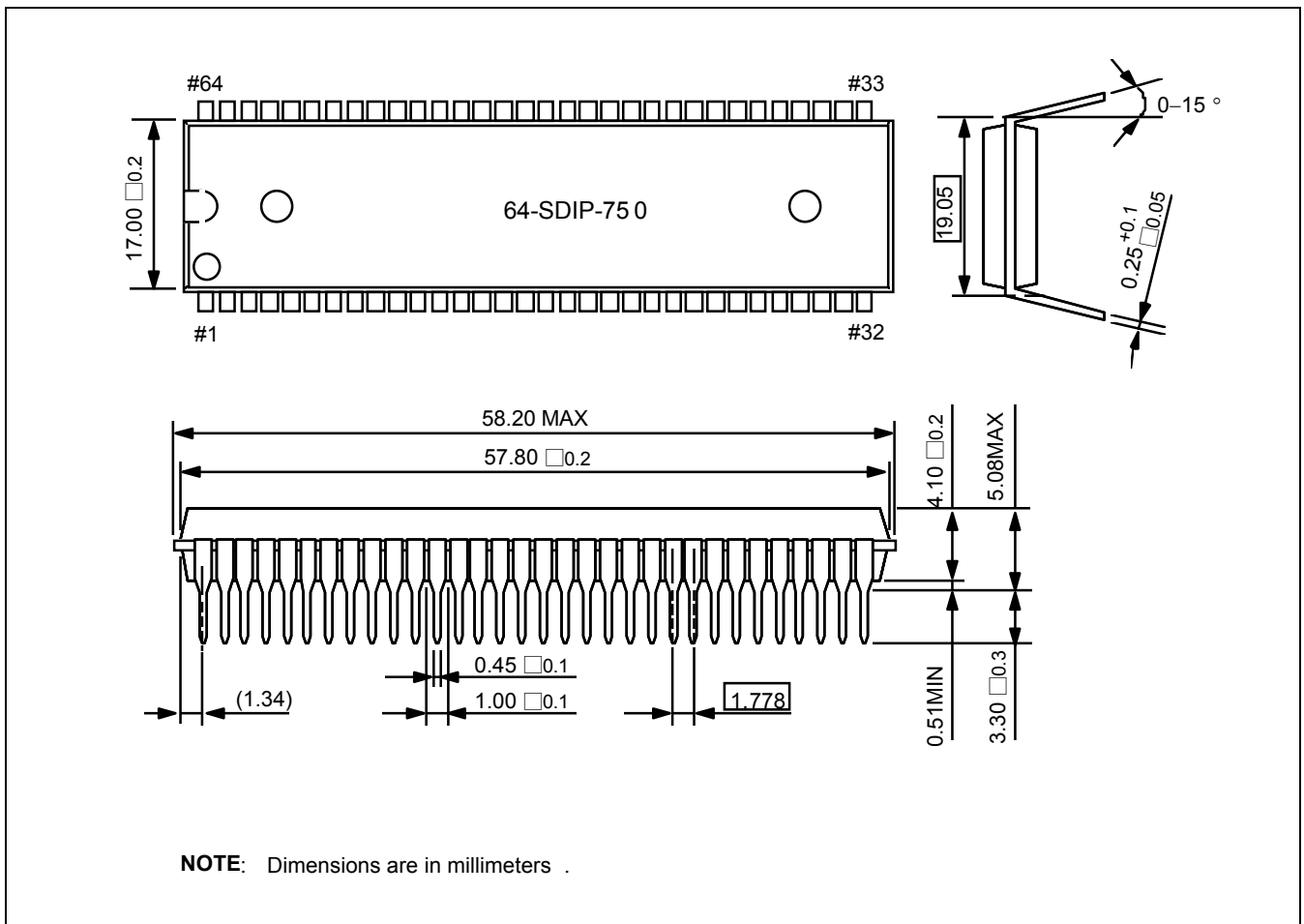
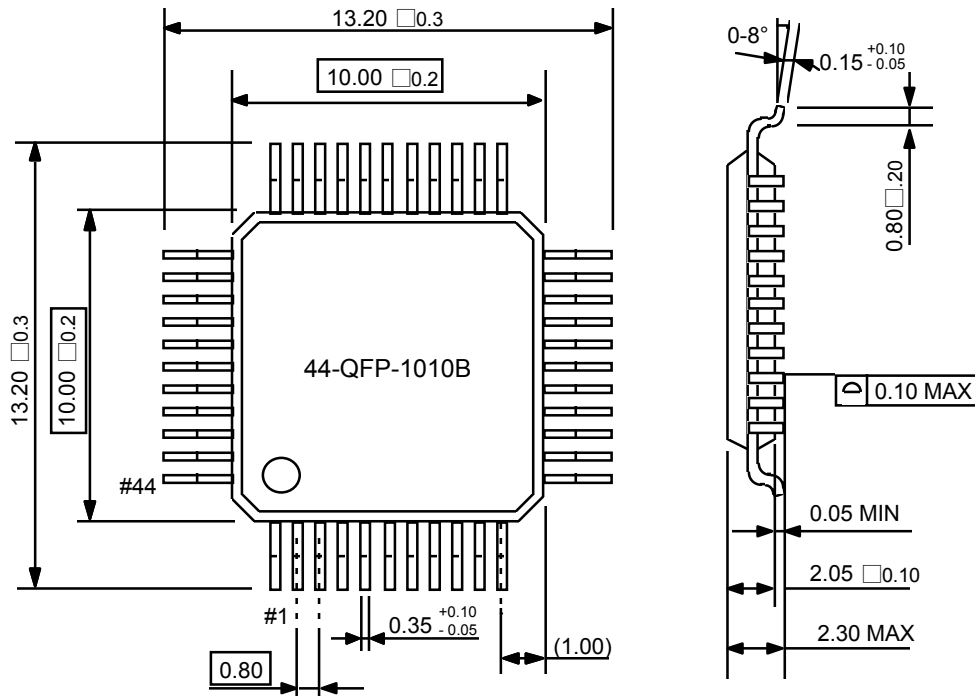


Figure 15-1. 64-SDIP-750 Package Dimensions



NOTE: Dimensions are in millimeters.

Figure 15-2. 64-QFP-1420F Package Dimensions

16

KS88P0716 OTP

OVERVIEW

The KS88P0716 single-chip CMOS microcontroller is the OTP (One Time Programmable) version of the KS88C0716 microcontrollers. It has an on-chip EPROM instead of masked ROM. The EPROM is accessed by serial data format.

KS88P0716 is fully compatible with KS88C0716, both in function and in pin configuration. As it has simple programming requirements, KS88P0716 is ideal for use as an evaluation chip for the KS88C0716.

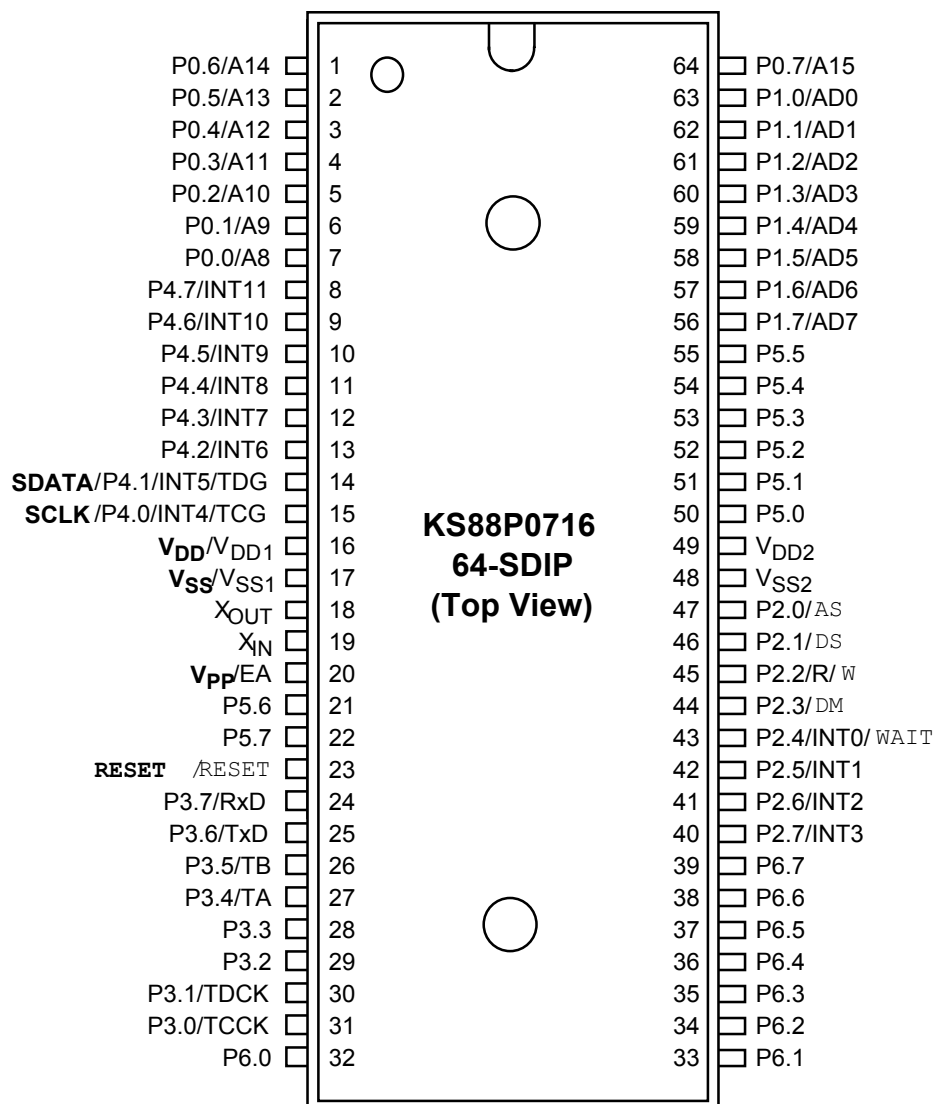


Figure 16-1. KS88P0716 Pin Assignments (64-SDIP Package)

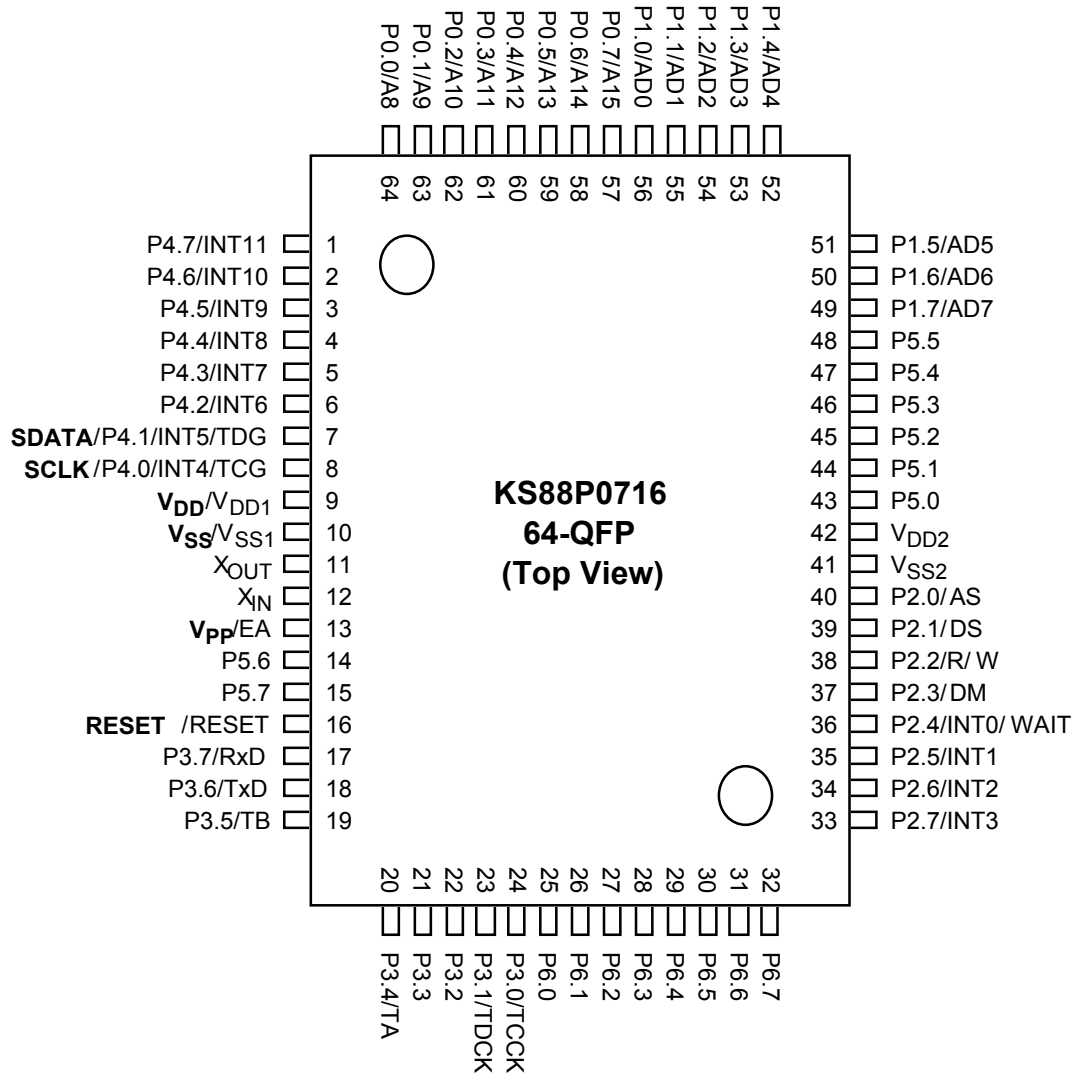


Figure 16-2. KS88P0716 Pin Assignments (64-QFP Package)

Table 16-1. Descriptions of Pins Used to Read/Write the EPROM

Main Chip Pin Name	During Programming			
	Pin Name	Pin No.	I/O	Function
P4.1	SDAT	14 (7)	I/O	Serial Data Pin (Output when reading, Input when writing) Input and Push-pull Output Port can be assigned.
P4.0	SCLK	15 (8)	I	Serial Clock Pin (Input Only Pin)
EA	V _{PP}	20 (13)	I	EPROM Cell Writing Power Supply Pin (Indicates OTP Mode Entering) When writing 12.5V is applied and when reading 5 V is applied (Option).
RESET	RESET	23 (9)	I	Chip Initialization
V _{DD1} /V _{SS1}	V _{DD} /V _{SS}	16/17 (9/10)	I	Logic Power Supply Pin. V _{DD} should be tied to 5V during programming.

NOTE: Parentheses indicate 64-QFP pin number.

Table 16-2. Comparison of KS88P0716 and KS88C0716 Features

Characteristic	KS88P0716	KS88C0716
Program Memory	16 K byte EPROM	16 K bytes mask ROM
Operating Voltage (V _{DD})	2.7 V to 5.5 V	2.7 V to 5.5V
OTP Programming Mode	V _{DD} = 5 V, V _{PP} (TEST) = 12.5V	
Pin Configuration	64 SDIP, 64 QFP	64 SDIP, 64 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 KS88P0716, 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 16-3. Operating Mode Selection Criteria

V _{DD}	V _{PP} (TEST)	REG/ MEM	ADDRESS (A15-A0)	R/W	MODE
5 V	5 V	0	0000H	1	EPROM read
	12.5 V	0	0000H	0	EPROM program
	12.5 V	0	0000H	1	EPROM verify
	12.5 V	1	0E3FH	0	EPROM read protection

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

D.C. ELECTRICAL CHARACTERISTICS

Table 16-4. D.C. Electrical Characteristics

($T_A = -40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$, $V_{DD} = 2.7\text{ V}$ to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input High Voltage	V_{IH1}	All input pins except V_{IH2}	$0.8 V_{DD}$		V_{DD}	V
	V_{IH2}	X_{IN}	$V_{DD} - 0.5$			
Input Low Voltage	V_{IL1}	All input pins except V_{IL2}			$0.2 V_{DD}$	V
	V_{IL2}	X_{IN}			0.4	
Output High Voltage	V_{OH1}	$V_{DD} = 4.5\text{ V}$ to 5.5 V $I_{OH} = -4\text{ mA}$ Port 5, 6	$V_{DD} - 1.0$			V
	V_{OH2}	$V_{DD} = 4.5\text{ V}$ to 5.5 V $I_{OH} = -1\text{ mA}$ All output pins except port 5, 6	$V_{DD} - 1.0$			
Output Low Voltage	V_{OL1}	$V_{DD} = 4.5\text{ V}$ to 5.5 V $I_{OL} = 15\text{ mA}$ Ports 5 and 6			1.0	V
	V_{OL2}	$I_{OL} = 2\text{ mA}$ Ports 0 - 4			0.4	

Table 16-4. D.C. Electrical Characteristics (Continued)(T_A = –40°C to +85°C, V_{DD} = 2.7 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input High Leakage Current	I _{LIH1}	V _{IN} = V _{DD} All input pins except X _{IN} , X _{OUT}	–	–	3	uA
	I _{LIH2}	V _{IN} = V _{DD} , X _{IN} , X _{OUT}			20	
Input Low Leakage Current	I _{LIL1}	V _{IN} = 0 V All input pins except X _{IN} , X _{OUT}	–	–	– 3	uA
	I _{LIL2}	V _{IN} = 0 V, X _{IN} , X _{OUT}			– 20	
Output High Leakage Current	I _{LOH}	V _{OUT} = V _{DD} All output pins	–	–	5	uA
Output Low Leakage Current	I _{LOL}	V _{OUT} = 0 V	–	–	– 5	uA
Pull-Up Resistor	R _{L1}	V _{IN} = 0 V; V _{DD} = 5 V Ports 0, 1, 4, 5 and 6	30	47	70	KΩ
	R _{L2}	V _{IN} = 0 V; V _{DD} = 5 V RESET only	110	210	310	
Supply Current ⁽¹⁾	I _{DD1} ⁽²⁾	V _{DD} = 5 V ± 10% 12-MHz oscillation	–	12	25	mA
		4-MHz oscillation		4.5	10	
		V _{DD} = 3 V ± 10% 12-MHz oscillation		6	15	
		4-MHz oscillation		2.5	7	
	I _{DD2} ⁽²⁾	Idle mode; V _{DD} = 5 V ± 10% 12-MHz oscillation		2.5	6	
		4-MHz oscillation		1.5	4	
		Idle mode; V _{DD} = 3 V ± 10% 12-MHz oscillation		1.2	3	
		4-MHz oscillation		0.6	1.5	
	I _{DD3}	Stop mode: V _{DD} = 5 V ± 10%		0.1	3	uA

NOTES:

1. Supply current does not include current drawn through internal pull-up resistors or external output current loads.
2. At supply current, the CPU clock frequency is the same as oscillation frequency (CPU use non divided clock).

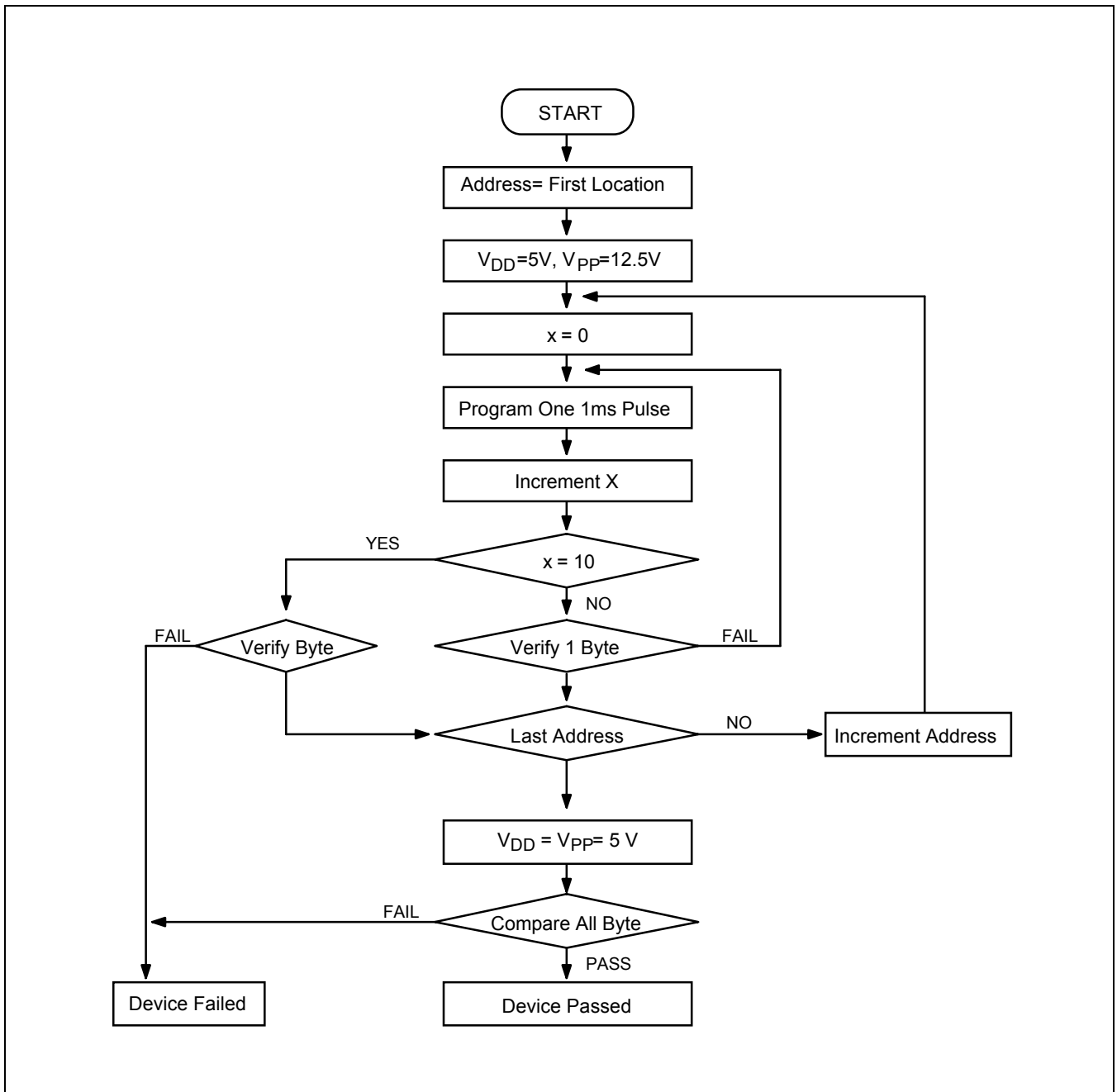


Figure 16-3. OTP Programming Algorithm

NOTES

17

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+, and KS57, KS86, KS88 families of microcontrollers. SMDS2+ is a new and improved version of SMDS2. Samsung also offers supporting software that includes a debugger, an 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-to-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, that 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.

SASM88

SASM88 is an relocatable assembler for Samsung's KS88-series microcontrollers. SASM88 takes a source file containing assembly language statements and translates them into a corresponding source code, object code and comments. The SASM88 supports macros and conditional assembly. It runs on the MS-DOS operating system. It produces relocatable object code only, so the user should link object file. Object files can be linked with other object files and can be loaded into memory.

HEX2ROM

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

TARGET BOARDS

Target boards are available for all KS88-series microcontrollers. All the required target system cables and adapters are included with the device-specific target board.

OTPs

One time programmable microcontroller (OTP) for the KS88C0716 microcontroller and OTP programmer (Gang) are now available.

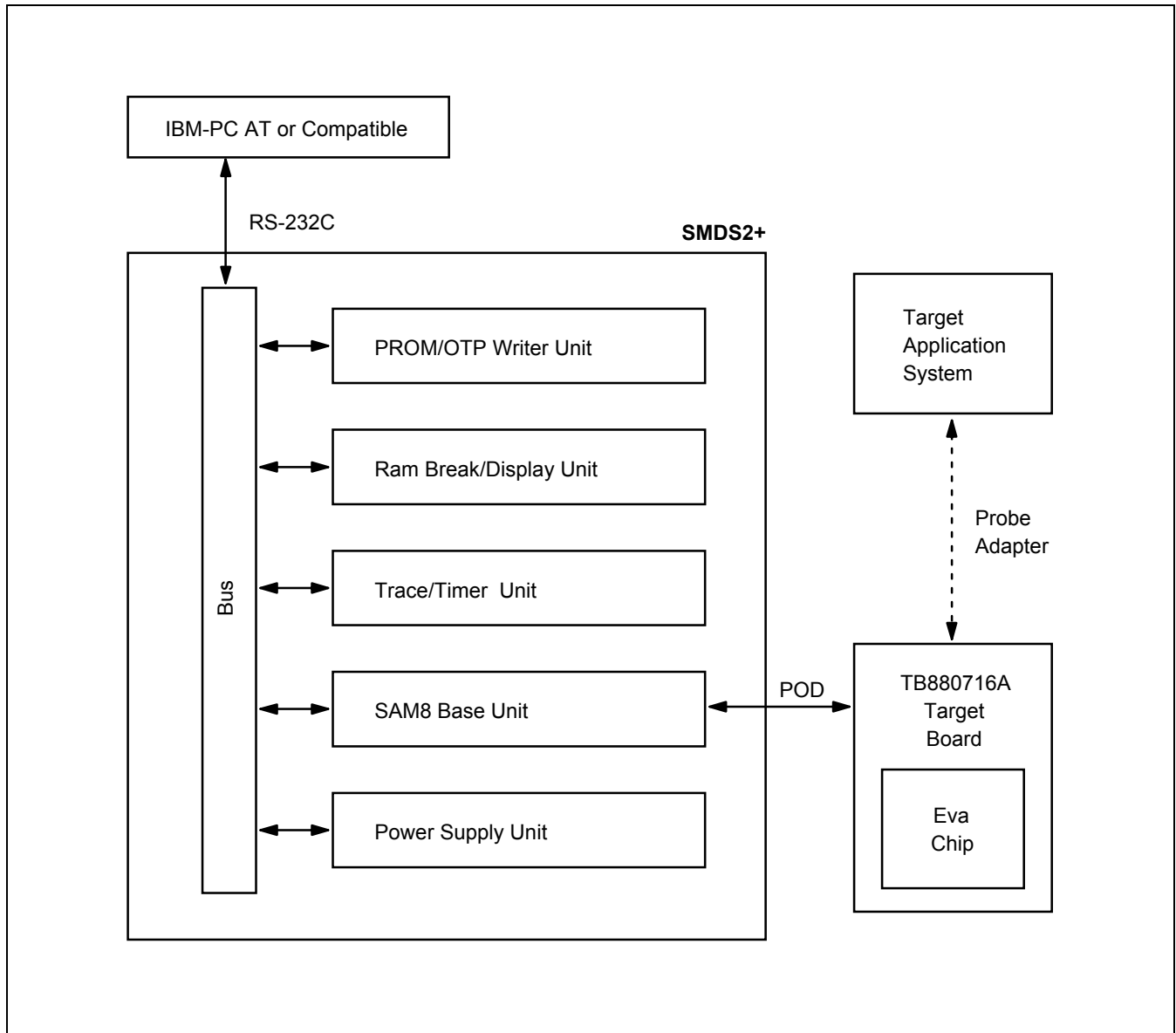


Figure 17-1. SMDS Product Configuration (SMDS2+)

TB880716A TARGET BOARD

The TB880716A target board is used for the KS88C0716/P0716 microcontroller. It is supported by the SMDS2+ development system.

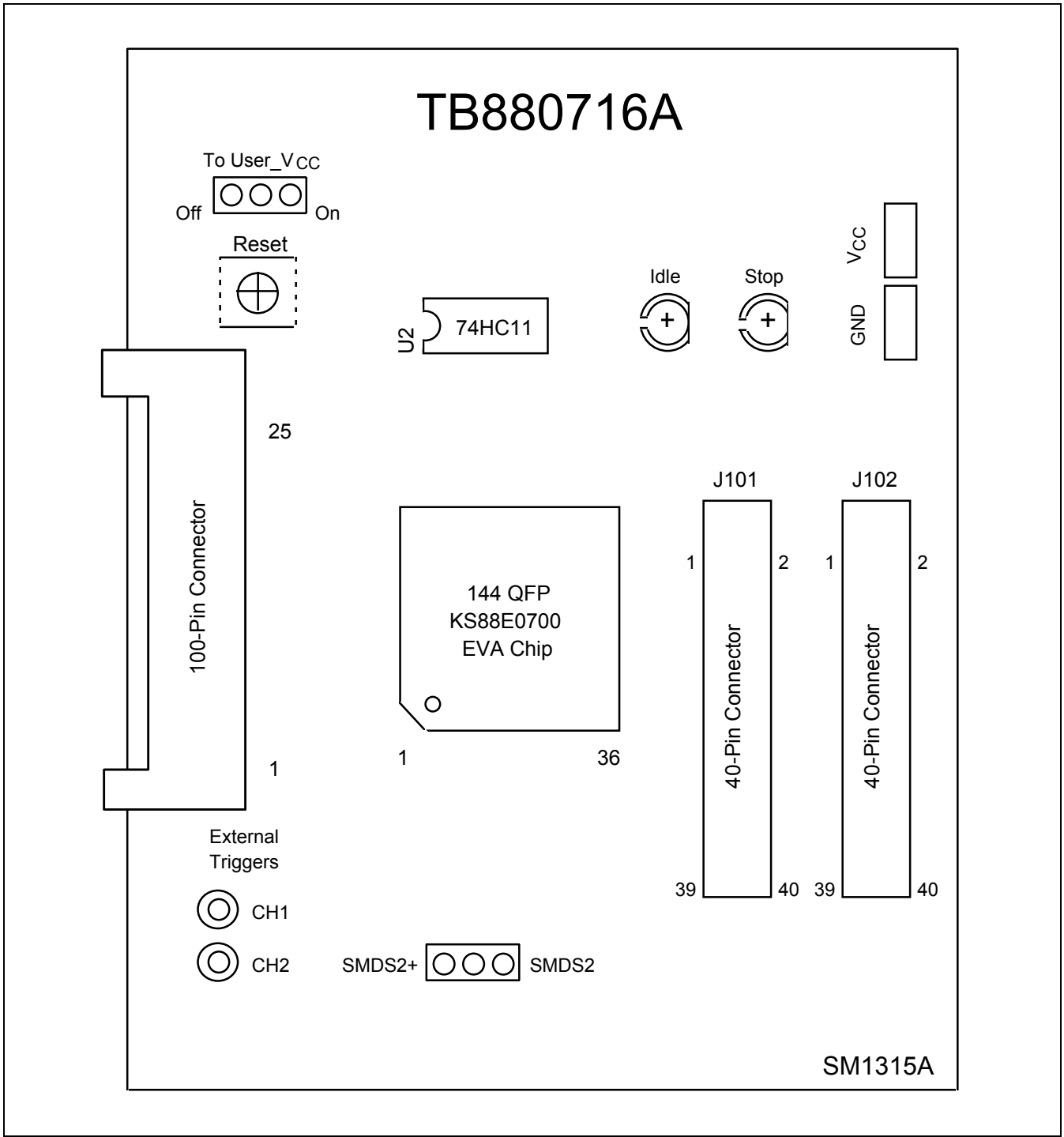
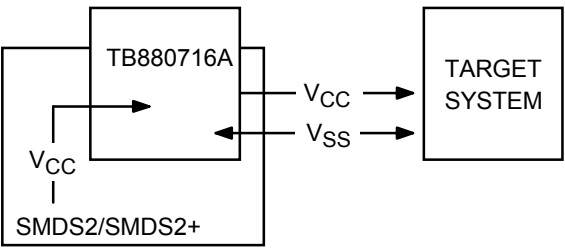
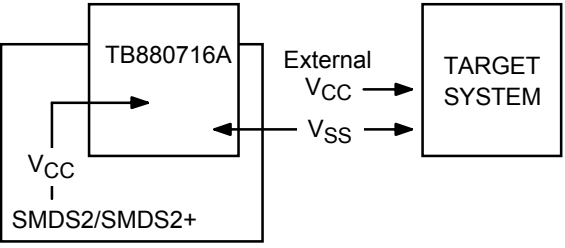


Figure 17-2. TB880716A Target Board Configuration

Table 17-1. Power Selection Settings for TB880716A

'To User_V _{CC} ' Settings	Operating Mode	Comments
To User_V_{CC} Off  On		SMDS2/SMDS2+ supply V _{CC} to the target board (evaluation chip) and the target system.
To User_V_{CC} Off  On		The SMDS2/SMDS2+ supply V _{CC} only to the target board (evaluation chip). The target system must have its own power supply.

SMDS2+ Selection (SAM8)

In order to write data into program memory that is available in SMDS2+, the target board should be selected for SMDS2+ through a switch as follows. Otherwise, the program memory writing function is not available.

Table 17-2. The SMDS2+ Tool Selection Setting

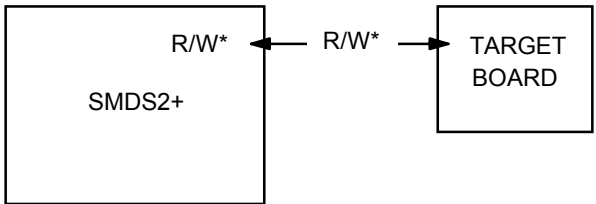
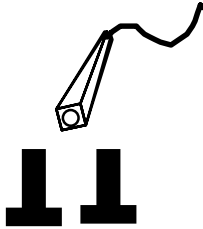
'SW1' Setting	Operating Mode
SMDS2  SMDS2+	

Table 17-3. Using Single Header Pins as the Input Path for External Trigger Sources

Target Board Part	Comments
External Triggers ○ CH1 ○ CH2	 Connector from external trigger sources of the application system You can connect an external trigger source to one of the two external trigger channels (CH1 or CH2) for the SMDS2+ breakpoint and trace functions.

IDLE LED

This LED is ON when the evaluation chip (KS88E0700) is in idle mode.

STOP LED

This LED is ON when the evaluation chip (KS88E0700) is in stop mode.

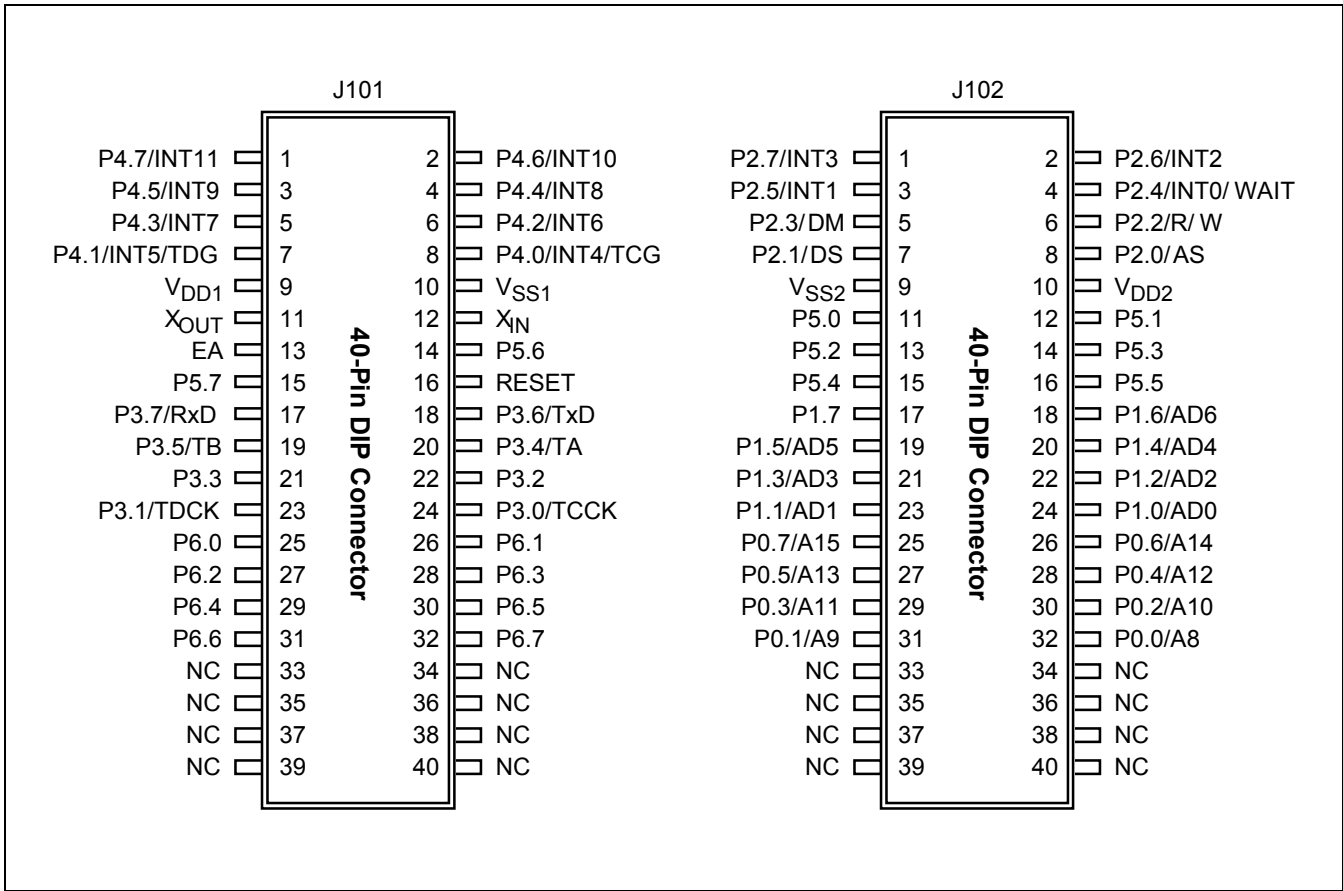


Figure 17-3. 40-Pin Connector for TB880716A

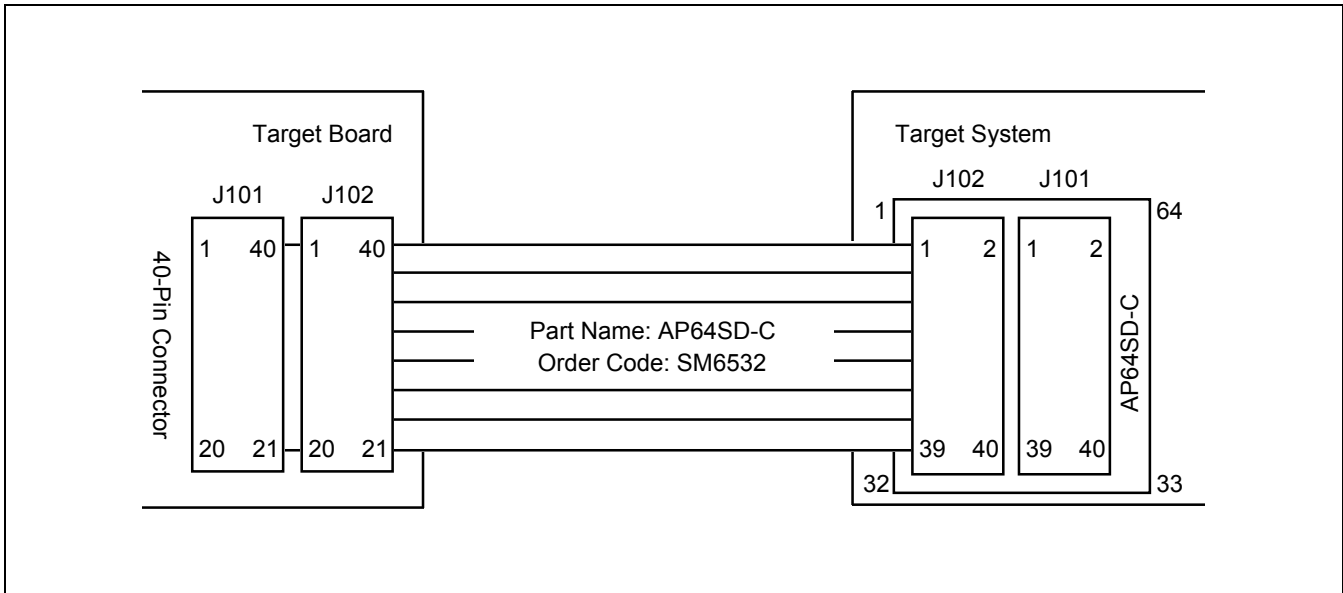


Figure 17-4. TB880716A Adapter Cable for 64-SDIP Package.