1 PRODUCT OVERVIEW

SAM87RC PRODUCT FAMILY

Samsung's new SAM87RC 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.

Timer/counters with selectable operating modes are included to support real-time operations. Many SAM87RC microcontrollers have an external interface that provides access to external memory and other peripheral devices.

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 six CPU clocks) can be assigned to one interrupt level at a time.

KS88C4708/C4716 MICROCONTROLLER

The KS88C4708/C4716 single-chip 8-bit microcontroller is designed for useful 10-bit resolution A/D converter, UART, PWM application field. Its powerful SAM87RC CPU architecture includes. The internal register file is logically expanded to increase the on-chip register space.

The KS88C4708/C4716 has 8/16 K bytes of on-chip program ROM. Following Samsung's modular design approach, the following peripherals are integrated with the SAM87RC core:

— Large number of programmable I/O ports (42 SDIP: 34 pins, 44 QFP: 36 pins)
— One asynchronous UART module
— Analog-to-digital converter with eight input channels and 10-bit resolution
— One 8-bit basic timer for watchdog function
— One 8-bit timer/counter with three operating modes (Timer 0)
— One general-purpose 16-bit timer/counters with three operating modes (Timer 1)

The KS88C4708/C4716 is a versatile general-purpose microcontroller that is ideal for use in a wide range of electronics applications requiring complex timer/counter, PWM, capture, and UART.
It is available in a 42-pin SDIP or 44-pin QFP package.

OTP

The KS88P4716 is an OTP (One Time Programmable) version of the KS88C4708/C4716 microcontroller. The KS88P4716 microcontroller has an on-chip 16-Kbyte one-time-programmable EPROM instead of a masked ROM. The KS88P4716 is comparable to the KS88C4708/C4716, both in function in D.C. electrical characteristics and in pin configuration.

FEATURES

CPU

- SAM87RC CPU core

Memory

- 272-byte general purpose register area
- 8/16-Kbyte internal program memory

Instruction Set

- 79 instructions
- IDLE and STOP instructions added for power-down modes

Instruction Execution Time

- 333 ns at 12 MHz f_{OSC} (minimum)

Interrupts

- 14 interrupt sources and 14 vectors
- Eight interrupt levels
- Fast interrupt processing

General I/O

- Five I/O ports (total 36 pins)
- Four bit-programmable ports
- Two n-channel open-drain output port

Timer/Counters

- One 8-bit basic timer for watchdog function
- One 8-bit timer/counter with three operating modes (timer 0)
- One 16-bit general-purpose timer/counters with three operation modes (timer 1)

UART

- One UART module
- Full duplex serial I/O interface with three UART modes

A/D Converter

- Eight analog input pins
- 10-bit conversion resolution
- 20 µs conversion time (10 MHz CPU clock)

Buzzer Frequency Output

- 200 Hz to 20 kHz signal can be generated

Oscillator Frequency

- 1 MHz to 12 MHz external crystal oscillator
- Maximum 12 MHz CPU clock

Operating Temperature Range

- – 40°C to + 85°C

Operating Voltage Range

- 1.8 V to 5.5 V

Package Types

- 42-pin SDIP, 44-pin QFP

BLOCK DIAGRAM

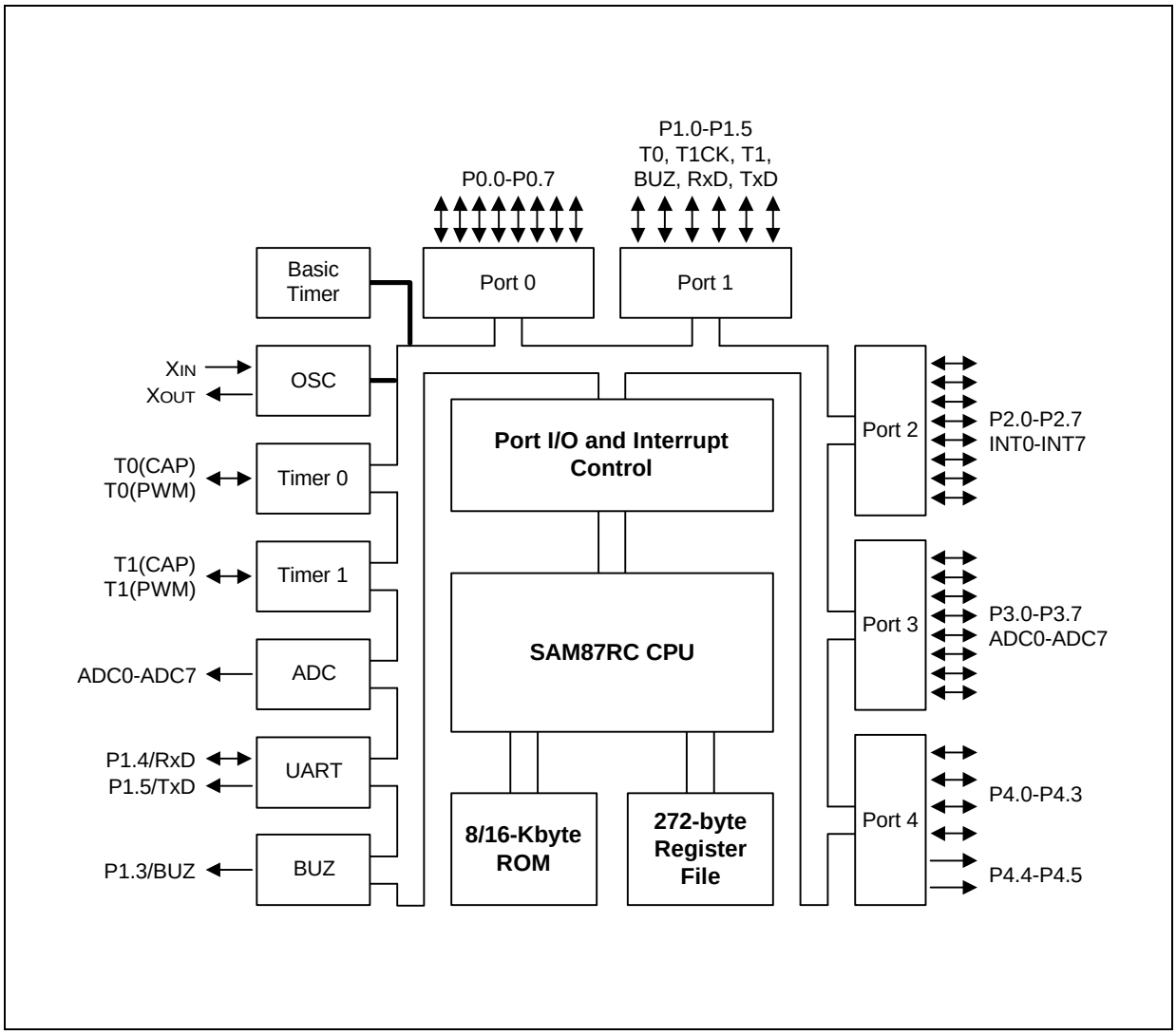


Figure 1-1. Block Diagram

PIN ASSIGNMENTS

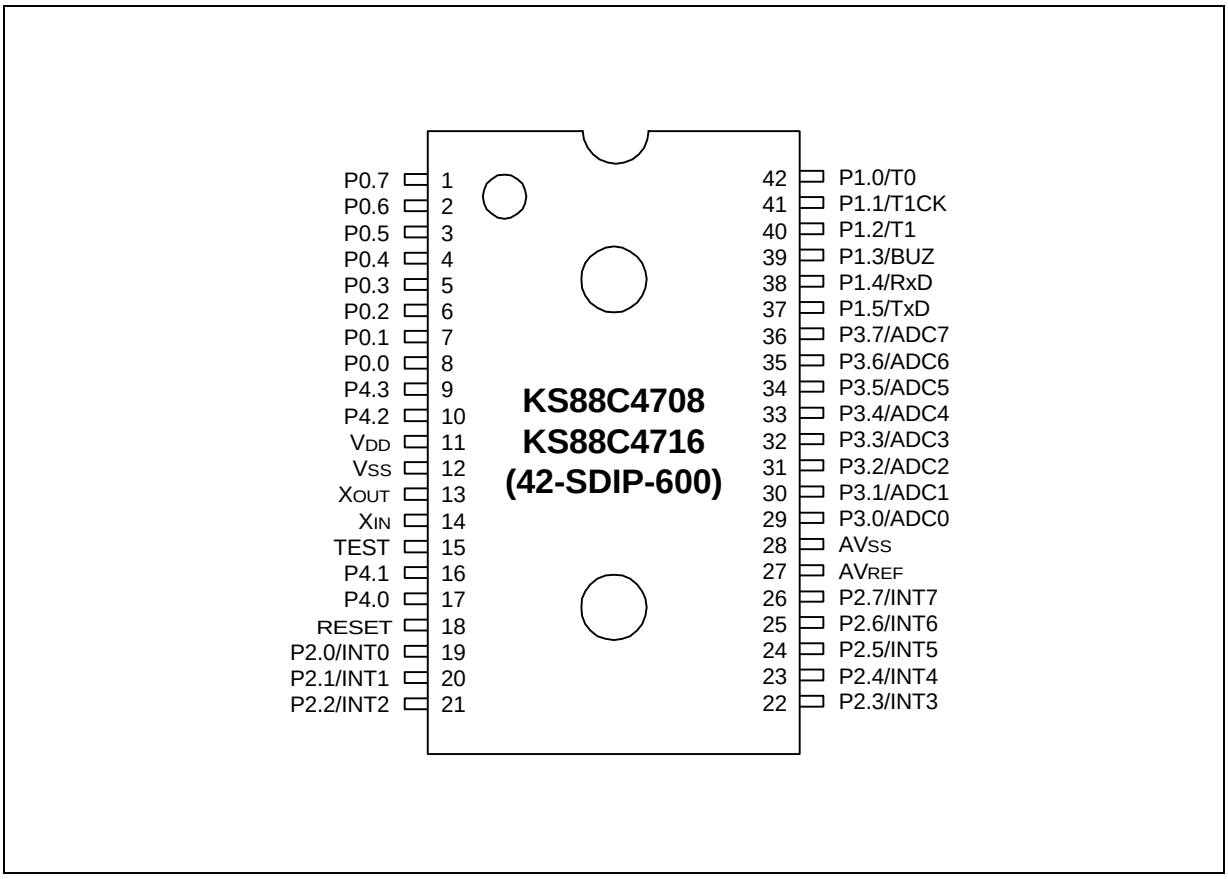


Figure 1-2. Pin Assignment Diagram (42-Pin SDIP Package)

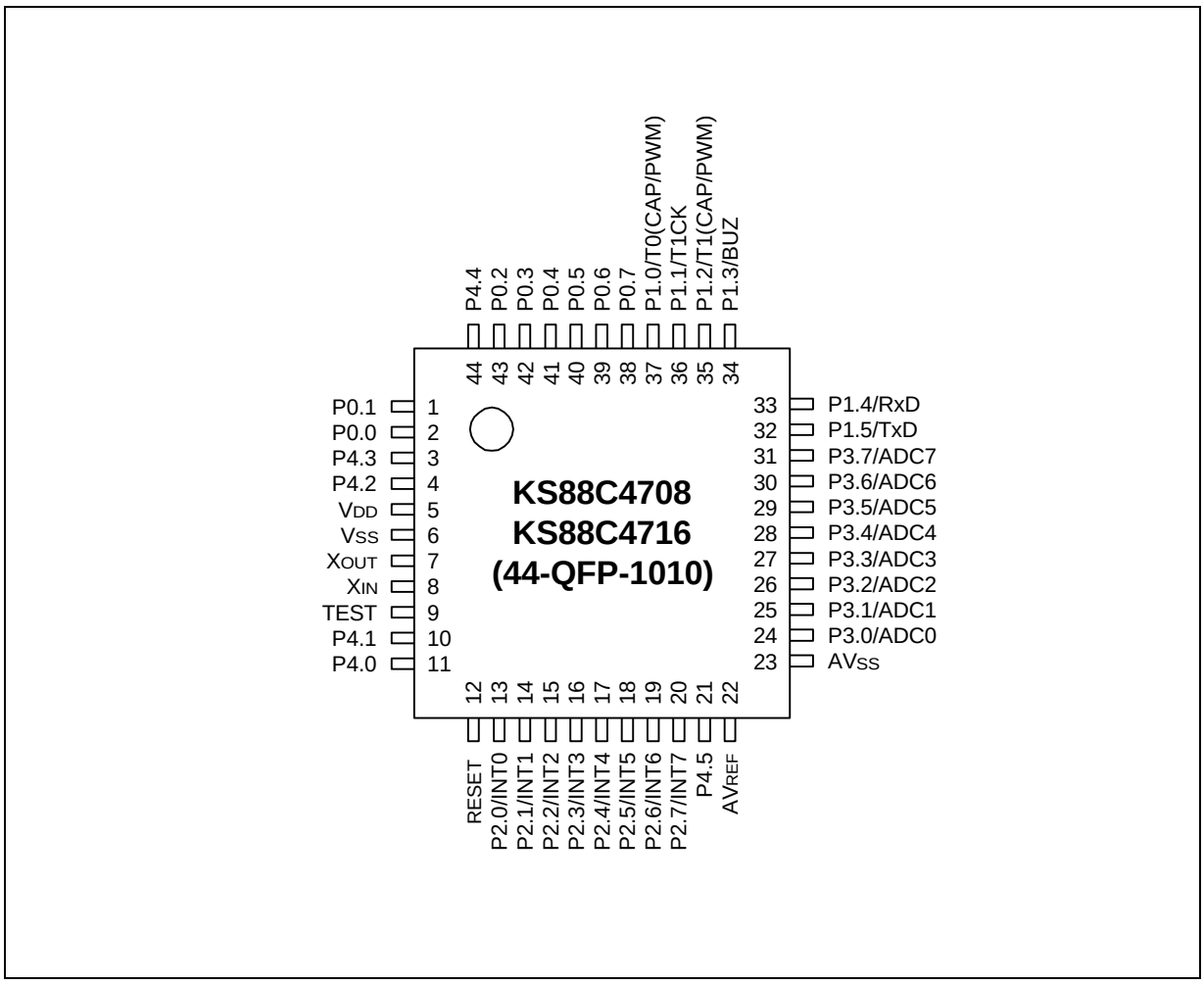


Figure 1-3. Pin Assignment Diagram (44-Pin QFP Package)

Table 1-1. KS88C4708/C4716 Pin Descriptions

Pin Name	Pin Type	Pin Description	Circuit Number	Pin Number	Share Pins
P0.0–P0.7	I/O	Nibble-programmable I/O port for Schmitt trigger input or push-pull, open-drain output. Pull-up resistors are assignable by software.	E	8-1 (2-1, 43-38)	–
P1.0–P1.5	I/O	Bit-programmable I/O port for Schmitt trigger input or push-pull output. Pull-up resistors are assignable by software. Port 1 pin can also by used as alternative function (T0, T1CK, T1, BUZ, RxD, TxD)	D	42-37 (37-32)	T0, T1CK, T1, BUZ, RxD, TxD
P2.0–P2.7	I/O	Bit-programmable I/O port for Schmitt trigger input or push-pull output. Pull-up resistors are assignable by software. Port 2 pins can also be used as external interrupt.	D	19-26 (13-20)	INT0-INT7
P3.0–P3.7	I/O	Bit-programmable I/O port for Schmitt trigger input or push-pull output. Pull-up resistors are assignable by software. Port 3 pins can also be used as A/D converter by software.	F	29-36 (24-31)	ADC0-ADC7
P4.0–P4.3	I/O	Bit-programmable I/O port for Schmitt trigger input or push-pull, open-drain output. Pull-up resistors are assingable by software.	E	17-16, 10-9 (11-10, 4-3)	–
P4.4–P4.5	O	Push-pull output only	C	(44, 21)	–
X_{IN}, X_{OUT}	–	Crystal or ceramic oscillator signal for system clock.	–	14, 13 (8, 7)	–
RESET	I	System reset signal input pin.	B	18 (12)	–
TEST	I	Test signal input pin (for factory use only; muse be connected to V_{SS})	–	15 (9)	–
AV_{REF}, AV_{SS}	–	A/D converter reference voltage input and ground	–	27, 28 (22, 23)	–
V_{DD}, V_{SS}	–	Voltage input pin and ground	–	11, 12 (5, 6)	–
T0	I/O	Timer 0 capture input or PWM output pin	D	42 (37)	P1.0
T1CK	I	Timer 1 external clock input pin	D	41 (36)	P1.1
T1	I/O	Timer 1 capture input or PWM output pin	D	40 (35)	P1.2
BUZ	O	200 Hz-20 KHz frequency output for buzzer sound	D	39 (34)	P1.3
RxD	I/O	UART receive and transmit input or output	D	38 (33)	P1.4
TxD	O	UART transmit output	D	37 (32)	P1.5
INT0-INT7	I	External interrupt input	E	19-26 (13-20)	P2.0-P2.7
ADC0-ADC7	I	A/D converter input	F	29-36 (24-31)	P3.0-P3.7

NOTE: Pin numbers shown in parentheses "( )" are for the 44-pin QFP package.

PIN CIRCUIT DIAGRAMS

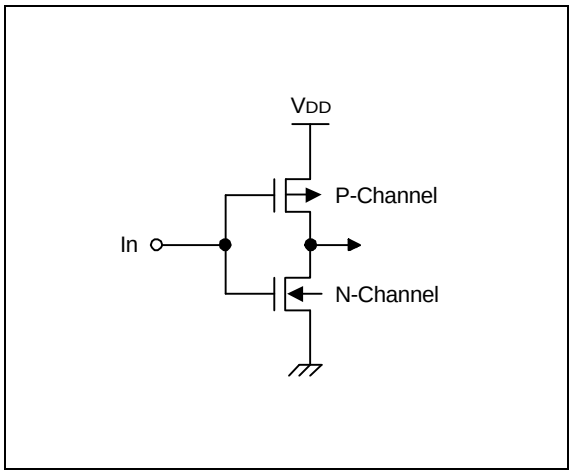


Figure 1-4. Pin Circuit Type A

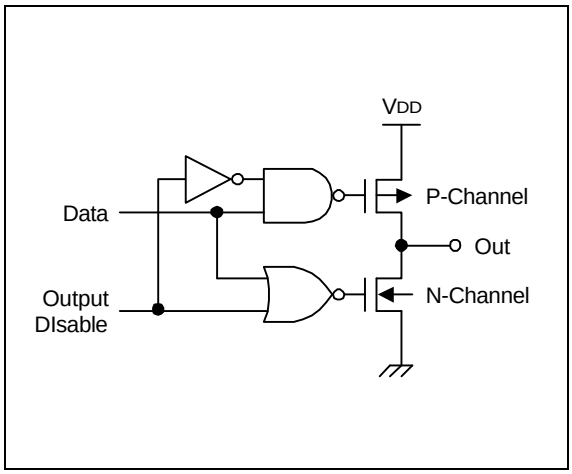


Figure 1-6. Pin Circuit Type C

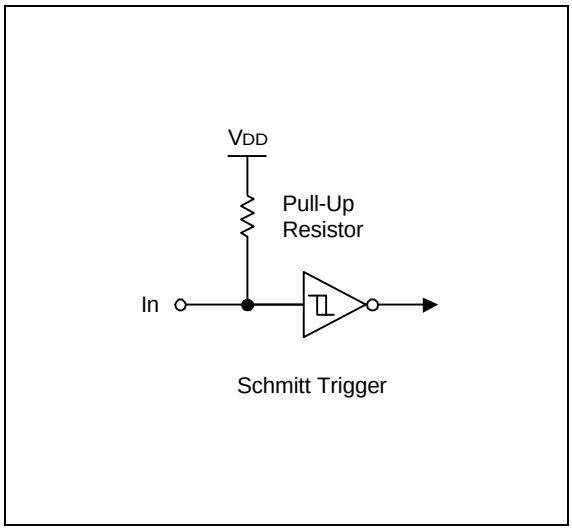


Figure 1-5. Pin Circuit Type B

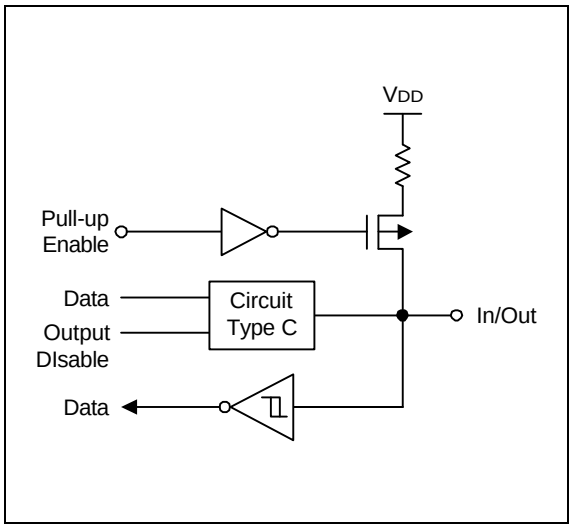


Figure 1-7. Pin Circuit Type D

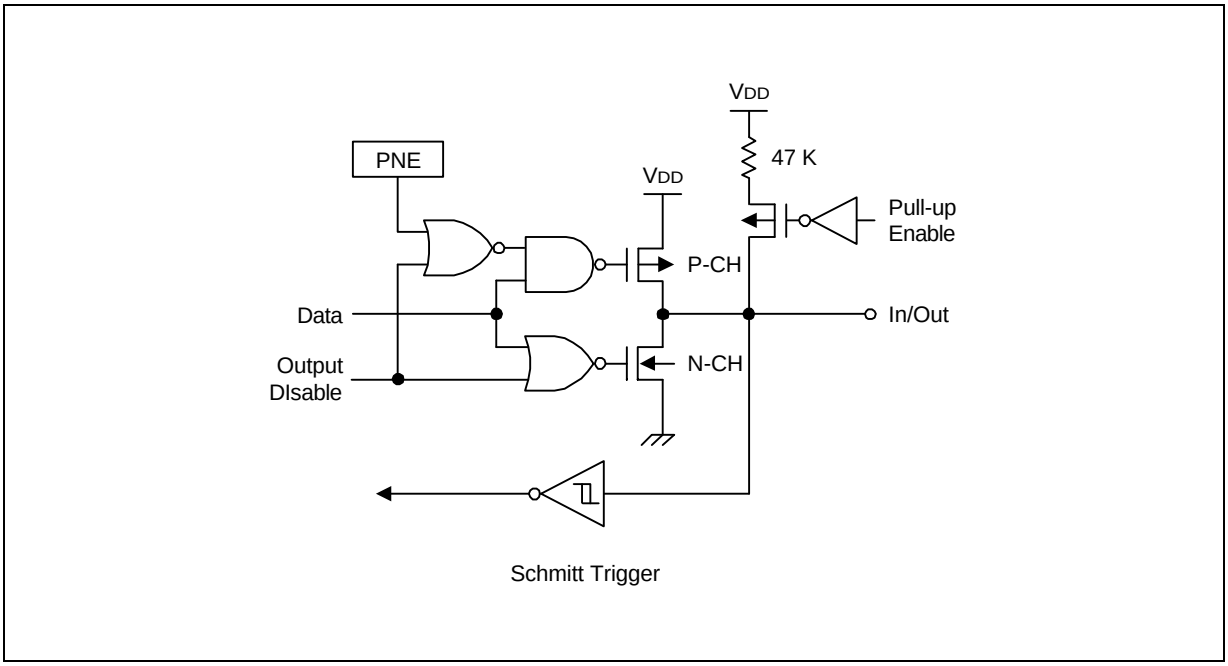


Figure 1-8. Pin Circuit Type E

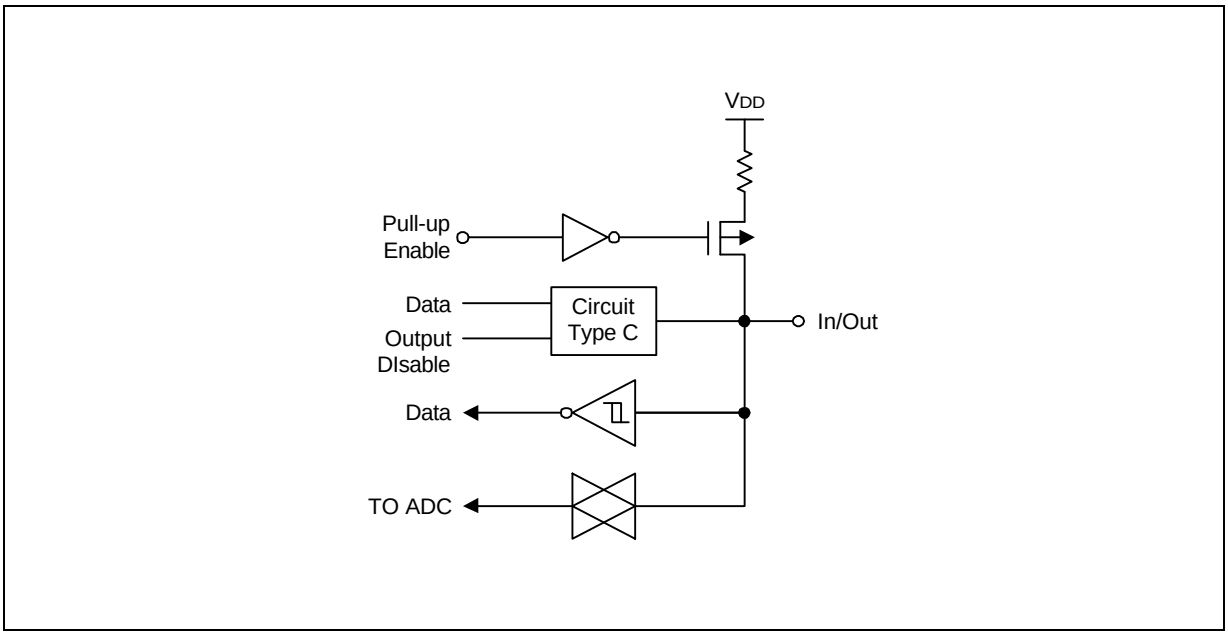


Figure 1-9. Pin Circuit Type F

2 ADDRESS SPACES

OVERVIEW

The KS88C4708/C4716 microcontroller has two types of address space:

— Internal program memory (ROM)

— Internal register file (RAM)

A 16-bit address bus supports program memory operations. A separate 8-bit register bus carries addresses and data between the CPU and the register file.

The KS88C4708/C4716/P4716 has a programmable internal 16-Kbyte ROM and 272-byte software-programmable RAM. An external memory interface is not implemented.

The 256-byte physical register space is expanded into an addressable area of 320 bytes by the use of addressing modes.

There are 317 mapped registers in the internal register file. Of these, 272 are for general-purpose use. (This number includes a 16-byte working register common area that is used as a “scratch area” for data operations, a 192-byte prime register area, and a 64-byte area (Set 2) that is also used for stack operations). Nineteen 8-bit registers are used for CPU and system control and 27 registers are mapped peripheral control and data registers. Three register locations are not mapped.

PROGRAM MEMORY (ROM AND RAM)

Programmable memory (ROM and RAM) stores program code or table data. The KS88C4708/C4716/P4716 has 16 K bytes of programmable internal memory. The programmable memory address range is therefore 0H-3FFFH for ROM.

The first 256 bytes of the ROM (0H–0FFH) are reserved for interrupt vector addresses. Unused locations in this address range can be used as normal program memory. If you do use the vector address area to store program code, be careful to avoid overwriting vector addresses stored in these locations.

The ROM address at which program execution starts after a reset is 0100H.

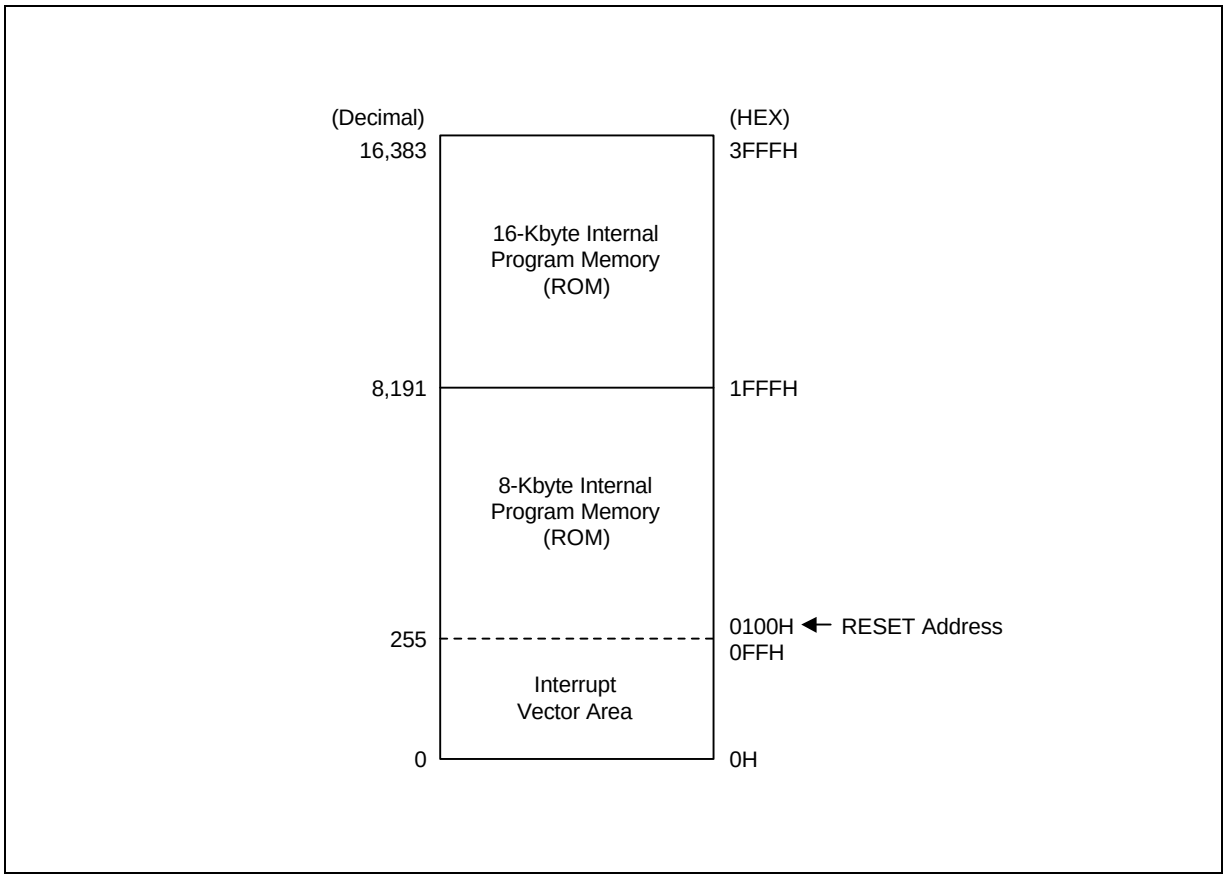


Figure 2-1. Program Memory Address Space

REGISTER ARCHITECTURE

The KS88C4708/C4716 register file has 319 registers. To increase the size of the internal register file, the upper 64-byte area of the register file is expanded two 64-byte areas, *set 1* and *set 2*. The remaining 192-byte area of the physical register file contains freely-addressable, general-purpose registers called *prime registers.*

The extension of register space into separately addressable sets is supported internally by addressing mode restrictions.

Specific register types and the area (in bytes) they occupy in the KS88C4708/C4716 internal register space are summarized in Table 2-1.

Table 2-1. KS88C4708/C4716 Register Type Summary

Register Type	Number of Bytes
General-purpose registers (including the 16-byte common working register area, the 192-byte prime register area, and the 64-byte set 2 area)	272
CPU and system control registers	20
Mapped clock, peripheral, and I/O control and data registers	27
Total Addressable Bytes	319

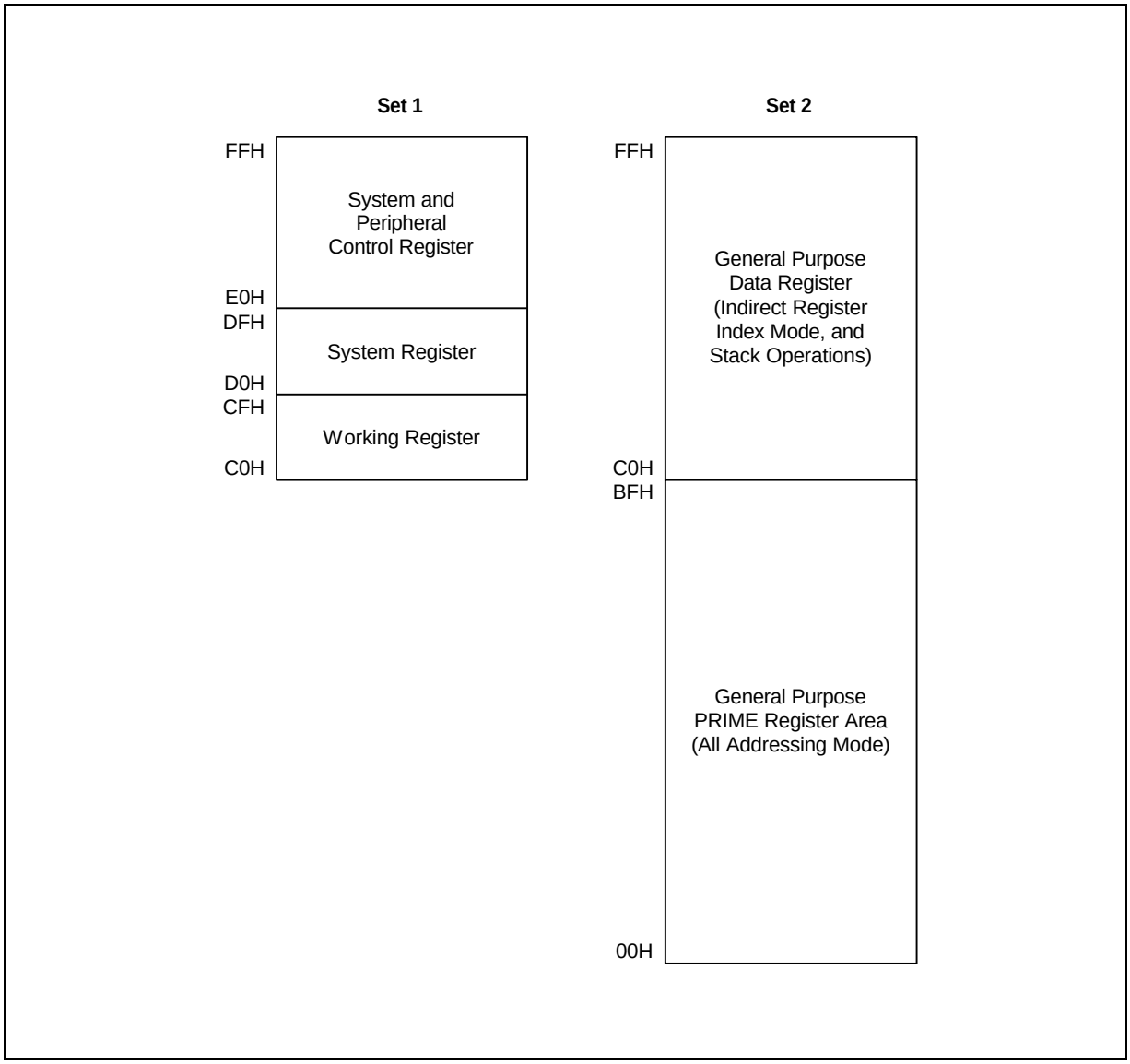


Figure 2-2. Internal Register File Organization

REGISTER PAGE POINTER (PP)

The KS88-series architecture supports the logical expansion of the physical 256-byte internal register file (using an 8-bit data bus) into as many as 15 separately addressable register pages. Page addressing is controlled by the register page pointer (PP, DFH). In the KS88C4708/C4716 microcontroller, a paged register file expansion is not implemented and the register page pointer settings therefore always point to “page 0.”

Following a reset, the page pointer's source value (lower nibble) and destination value (upper nibble) are always '0000', automatically selecting page 0 as the source and destination page for register addressing. These page pointer (PP) register settings, as shown in Figure 2-3, should not be modified during normal operation.

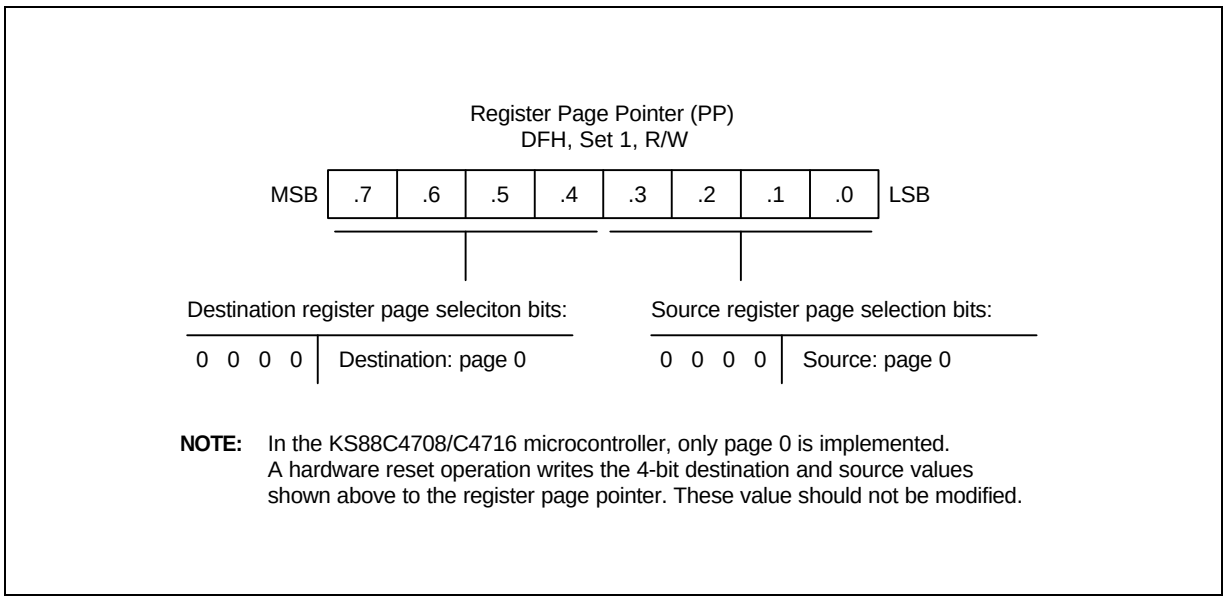


Figure 2-3. Register Page Pointer (PP)

REGISTER SET 1

The term *set 1* refers to the upper 64 bytes of the register file, locations C0H–FFH.

In some KS88-series microcontrollers, the upper 32-byte area of this 64-byte space (E0H–FFH) is divided into two 32-byte register banks, *bank 0* and *bank 1*. The set register bank instructions SB0 or SB1 are used to address one bank or the other. In the KS88C4708/C4716 microcontroller, bank 1 is not implemented. A hardware reset operation therefore always selects bank 0 addressing, and the SB0 and SB1 instructions are not necessary.

The upper 32-byte area of set 1 (FFH–E0H) contains 31 mapped system and peripheral control registers. The lower 32-byte area contains 16 system registers (DFH–D0H) and a 16-byte common working register area (CFH–C0H). You can use the common working register area as a “scratch” area for data operations being performed in other areas of the register file.

Registers in set 1 locations are directly accessible at all times using the Register addressing mode. The 16-byte working register area can only be accessed using working register addressing. (For more information about working register addressing, please refer to Section 3, “Addressing Modes,” .)

REGISTER SET 2

The same 64-byte physical space that is used for set 1 locations C0H–FFH is logically duplicated to add another 64 bytes of register space. This expanded area of the register file is called *set 2*. All set 2 locations (C0H–FFH) are addressed as part of page 0 in the KS88C4708/C4716 register space.

The logical division of set 1 and set 2 is maintained by means of addressing mode restrictions: You can use only Register addressing mode to access set 1 locations; to access registers in set 2, you must use Register Indirect addressing mode or Indexed addressing mode.

The set 2 register area is commonly used for stack operations.

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 using any addressing mode. (In other words, there is no addressing mode restriction for these registers, as is the case for set 1 and set 2 registers.) All registers in prime area locations are addressable immediately following a reset.

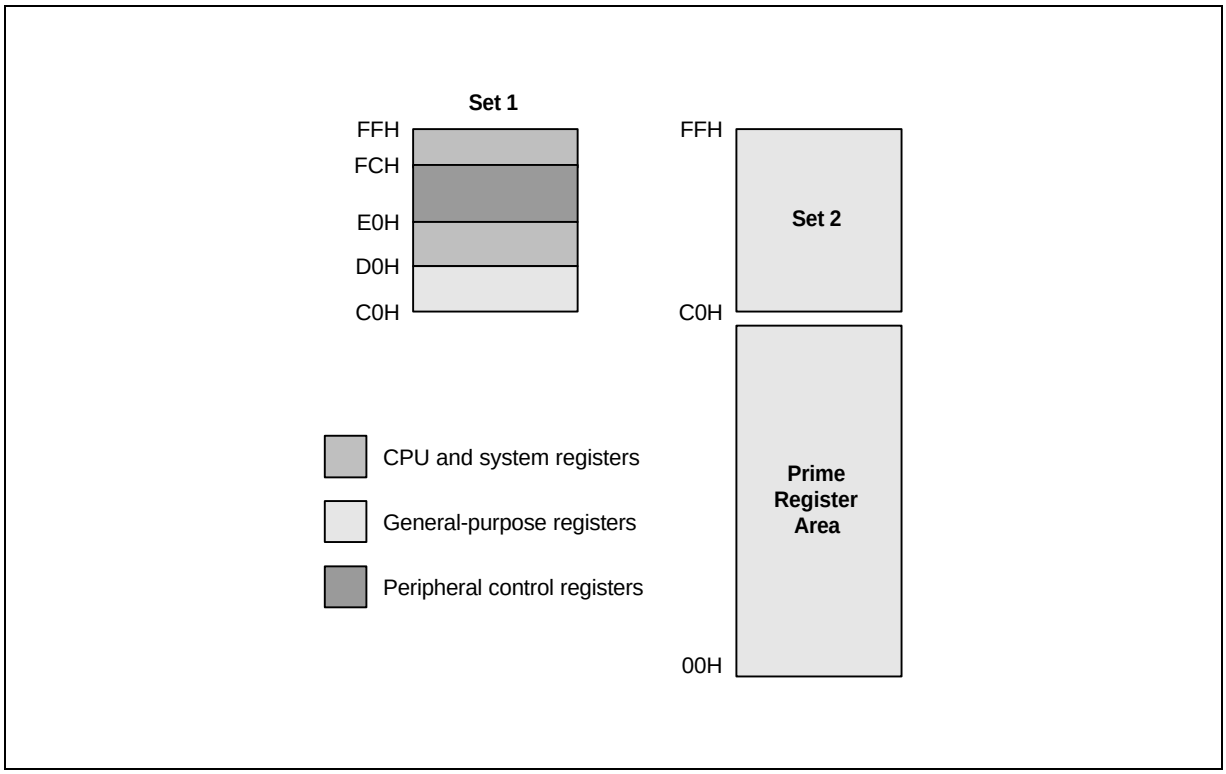


Figure 2-4. Set 1, Set 2, and Prime Area Register Map

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 seen by the programmer as consisting of 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 one time to form a 16-byte working register block. Using the register pointers, you can move this 16-byte register block 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 of 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).

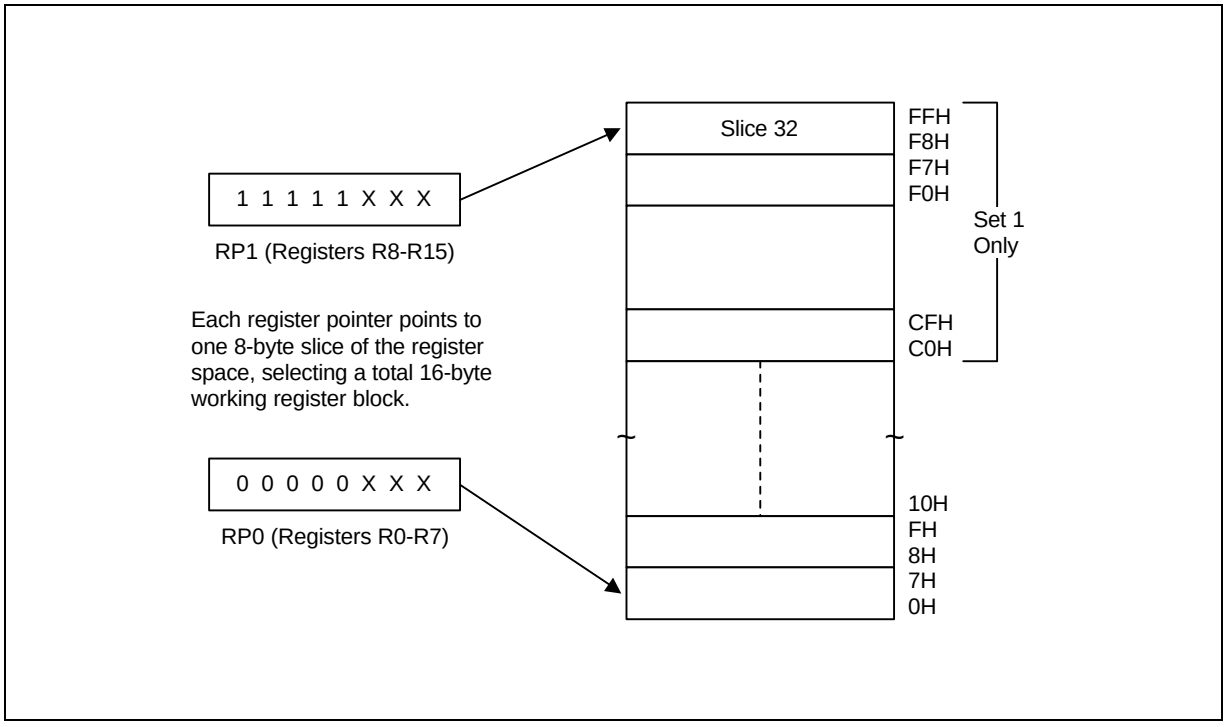


Figure 2-5. 8-Byte Working Register Areas (Slices)

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 space 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 the 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
```

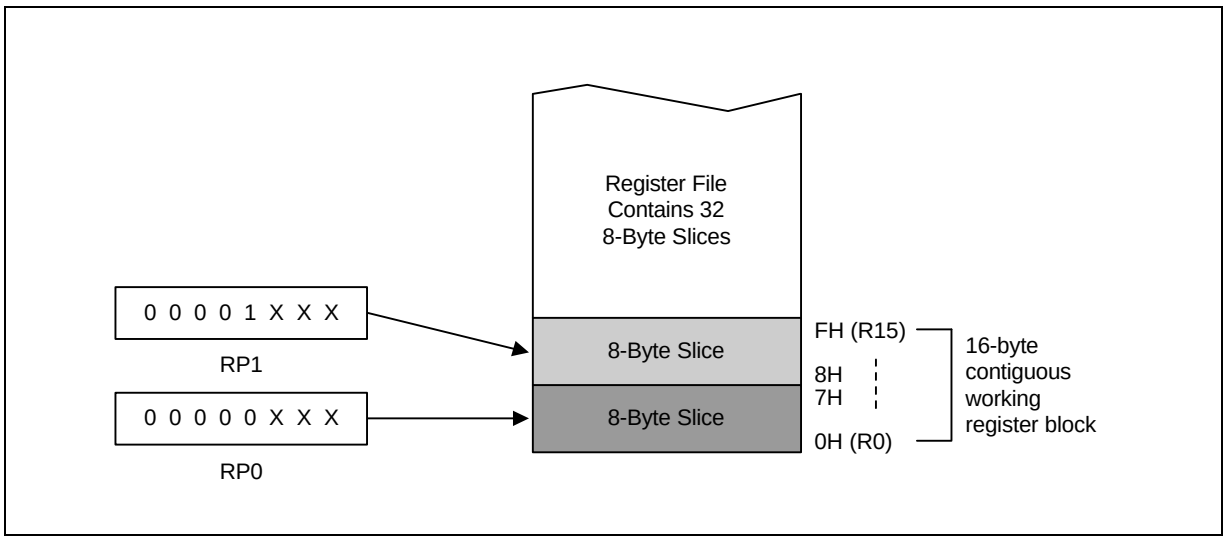


Figure 2-6. Contiguous 16-Byte Working Register Block

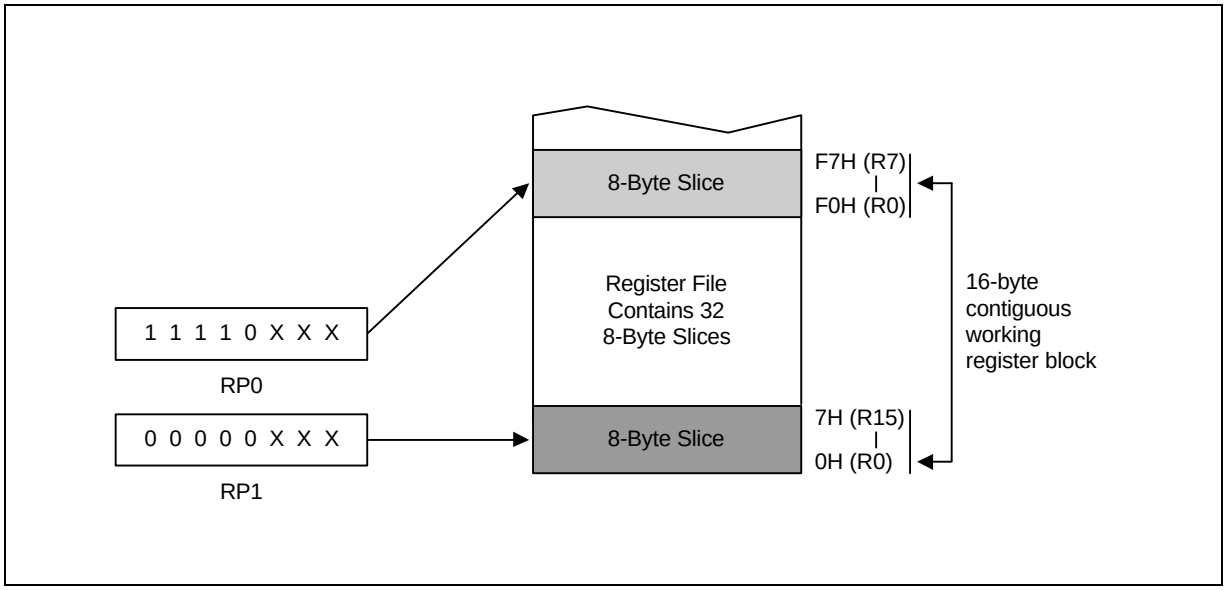


Figure 2-7. Non-Contiguous 16-Byte Working Register Block

☞ 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. The register addresses 80H through 85H contains 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 KS88-series 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 identify a specific 8-byte working register space in the internal register file and a specific 8-bit register within that space.

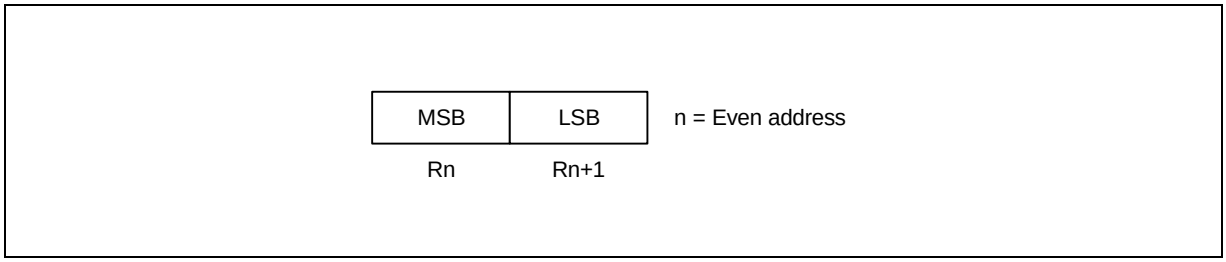


Figure 2-8. 16-Bit Register Pair

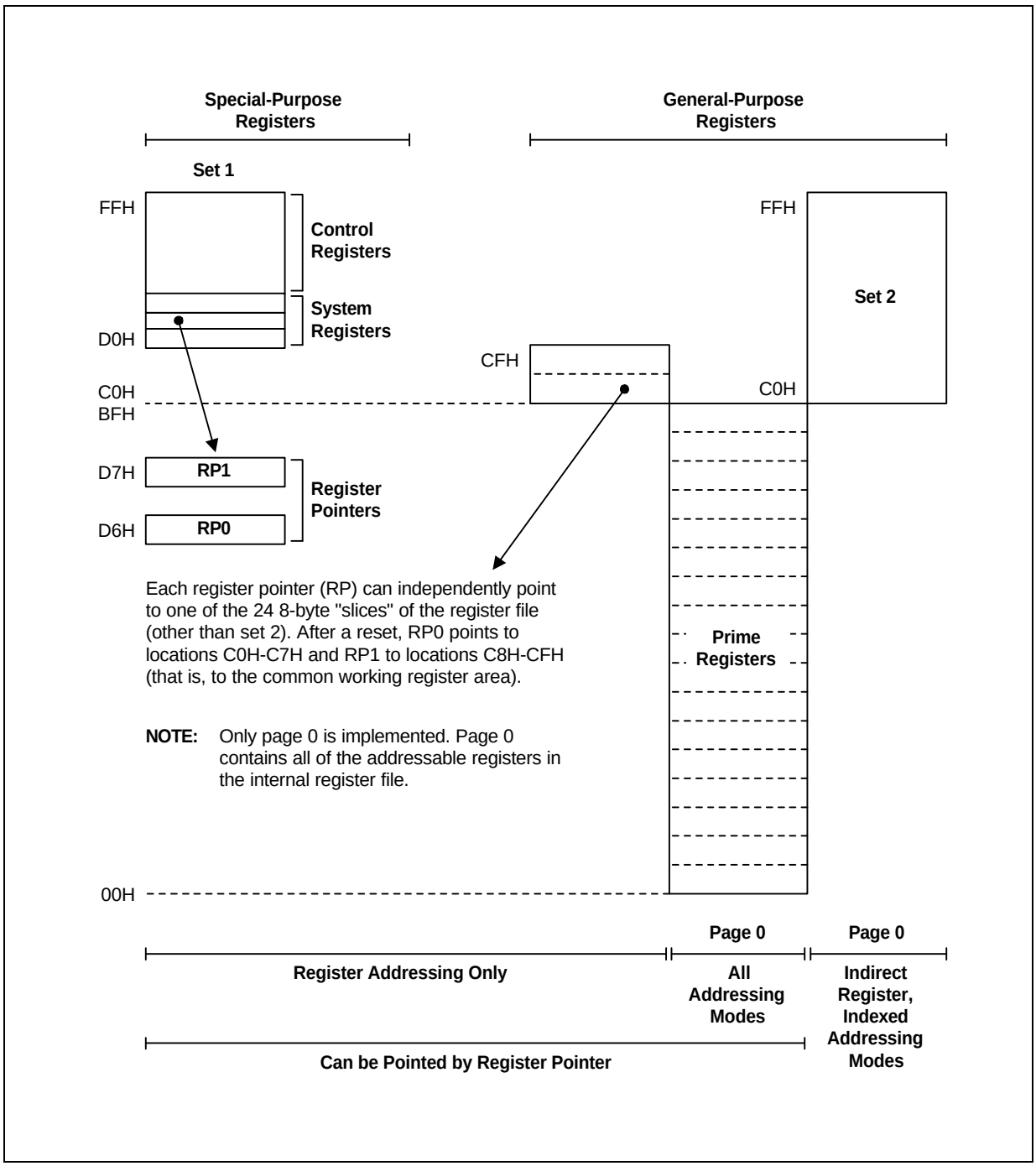


Figure 2-9. Register File Addressing

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 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.

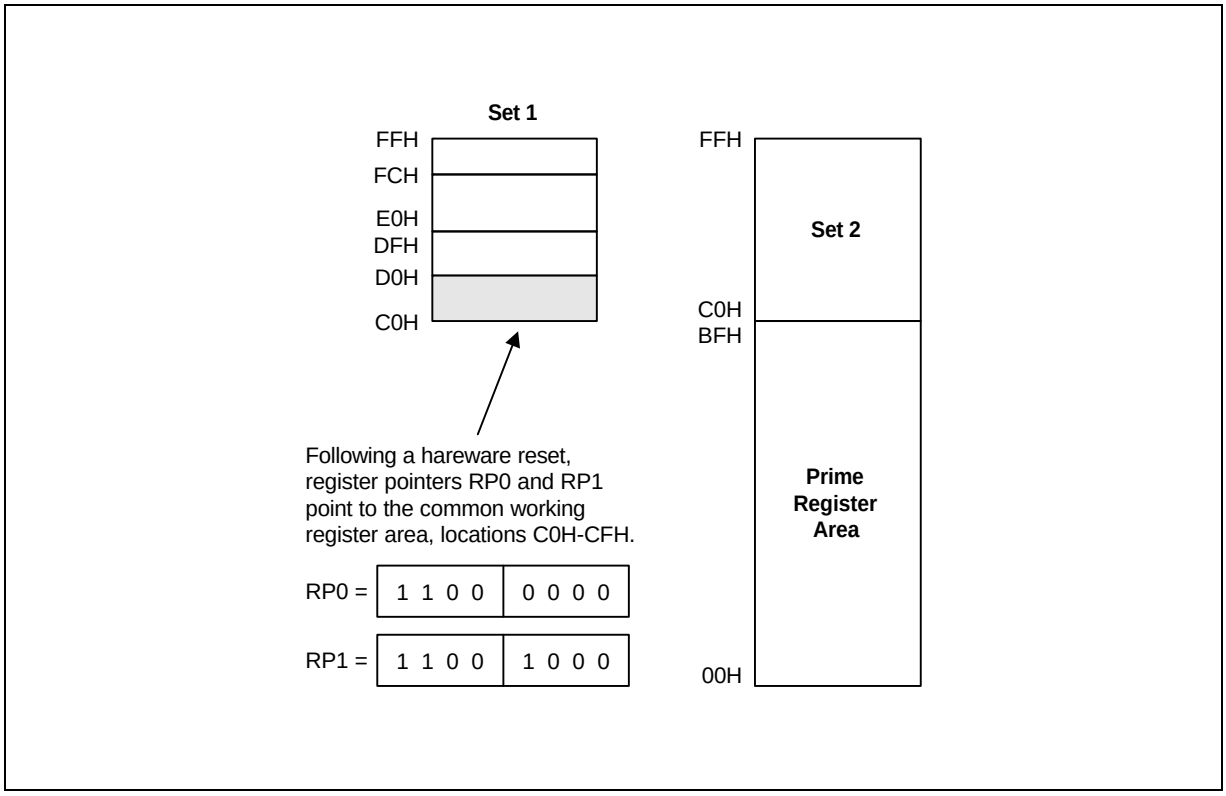


Figure 2-10. Common Working Register Area

☞ 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).

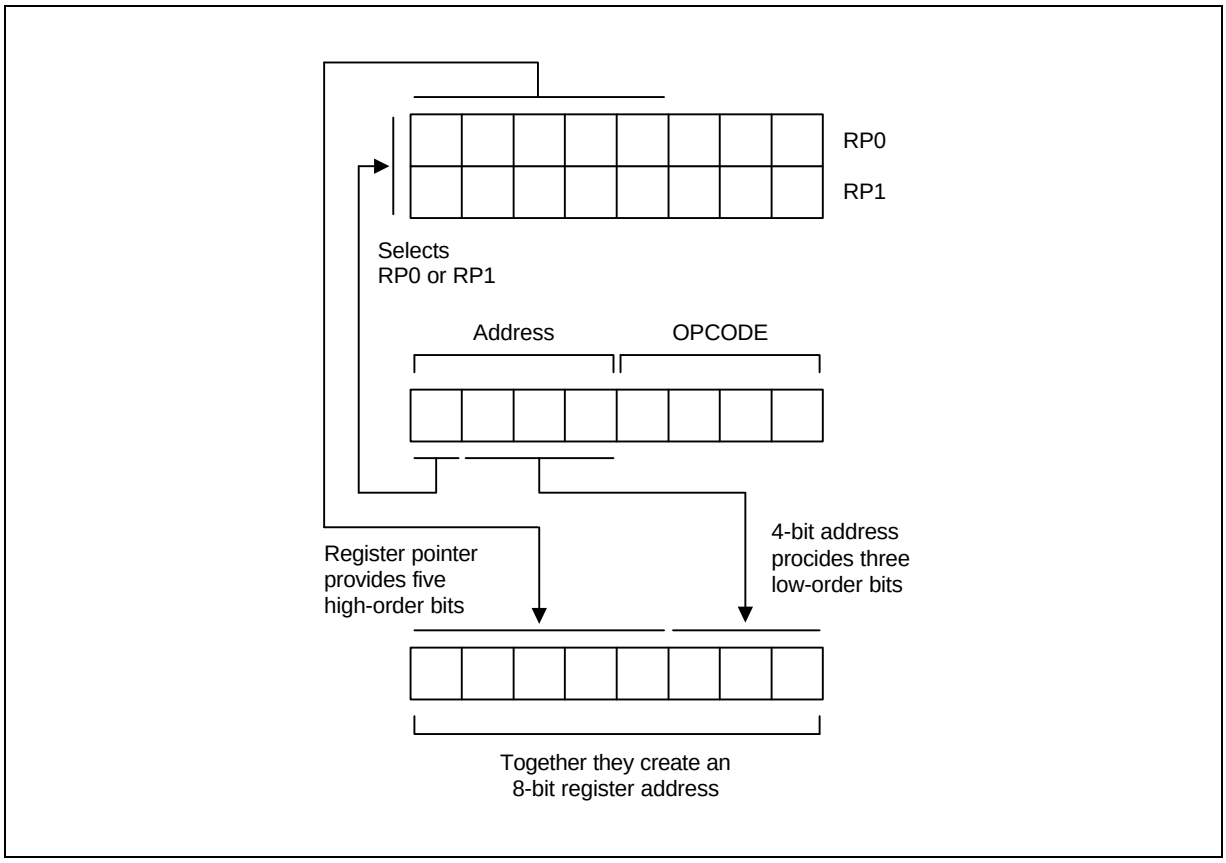


Figure 2-11. 4-Bit Working Register Addressing

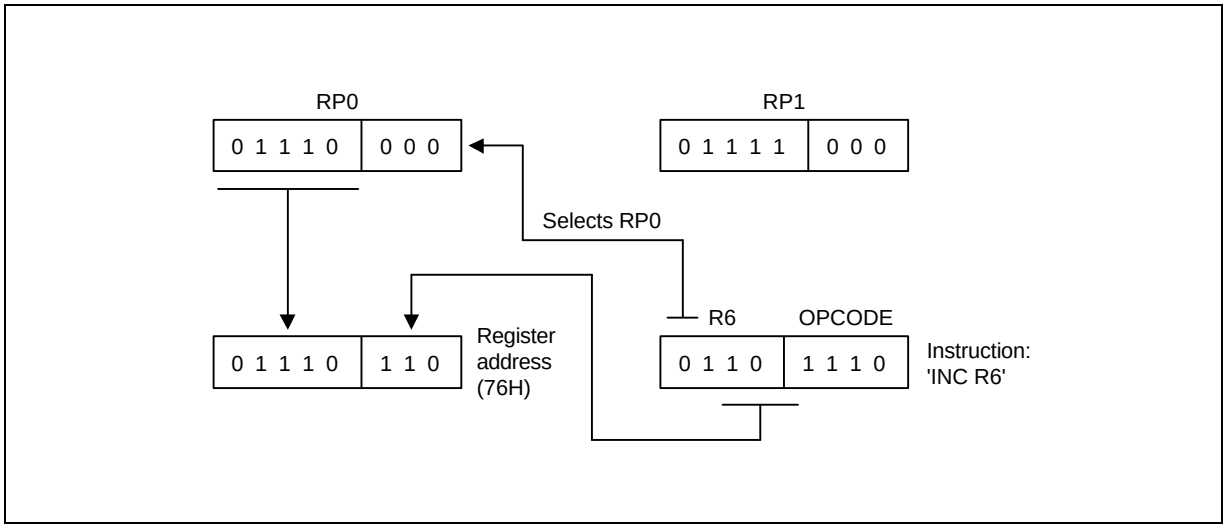


Figure 2-12. 4-Bit Working Register Addressing Example

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).

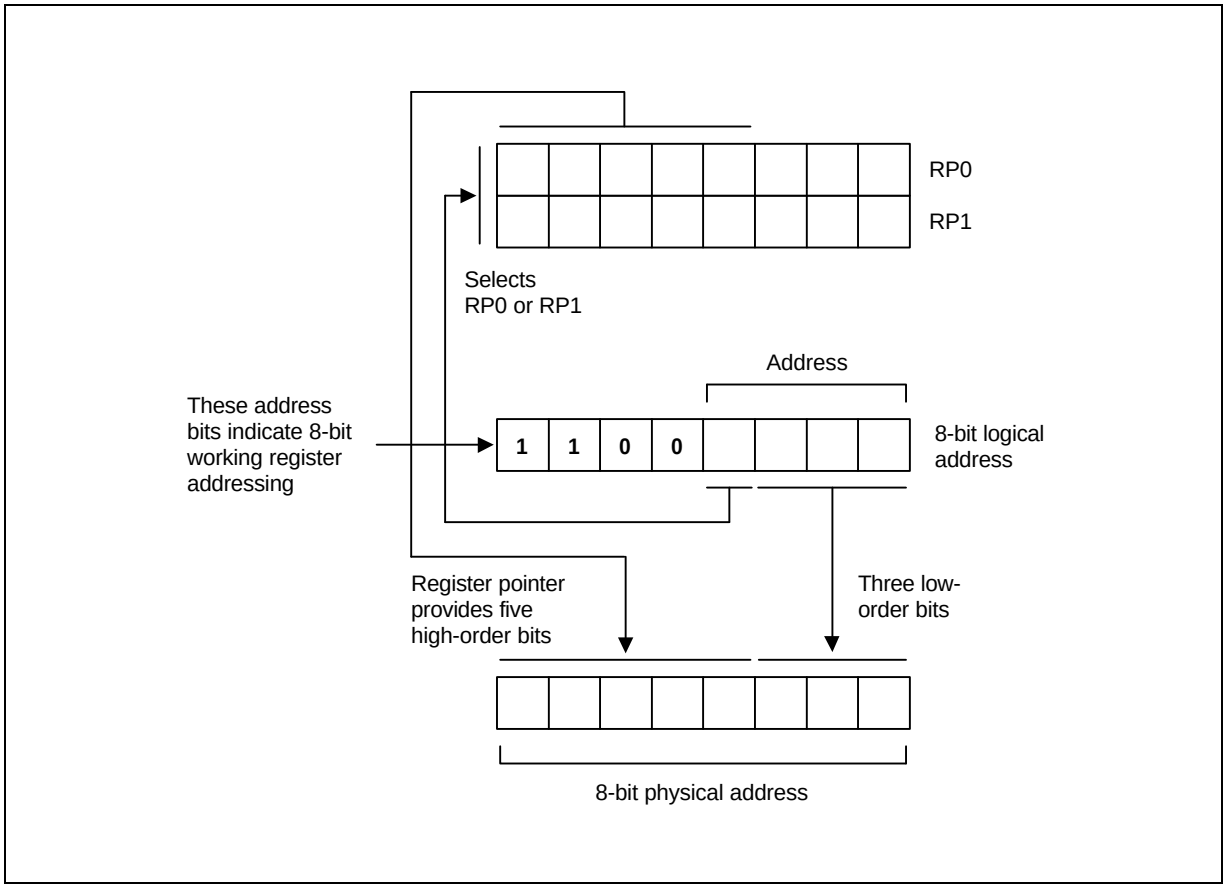


Figure 2-13. 8-Bit Working Register Addressing

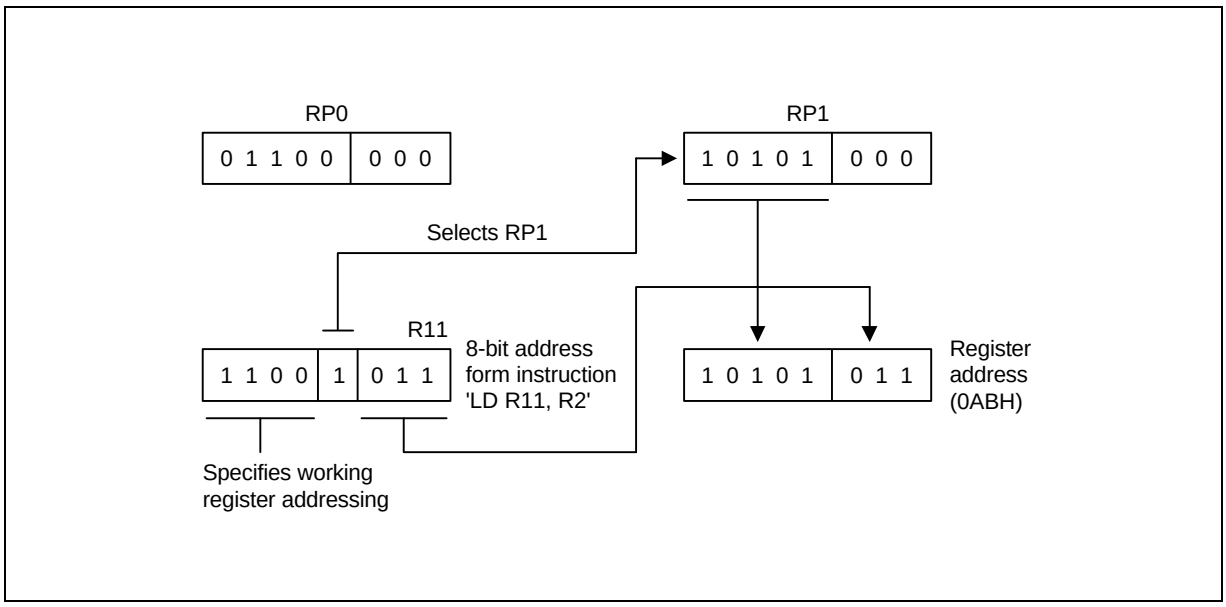


Figure 2-14. 8-Bit Working Register Addressing Example

SYSTEM AND USER STACKS

KS88-series microcontrollers use the system stack for subroutine calls and returns and to store data. The PUSH and POP instructions are used to control system stack operations. The KS88C4708/C4716 architecture supports stack operations in the internal register file.

Stack Operations

Return addresses for procedure calls and 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 value is always decreased by one *before* a push operation and increased by one *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.

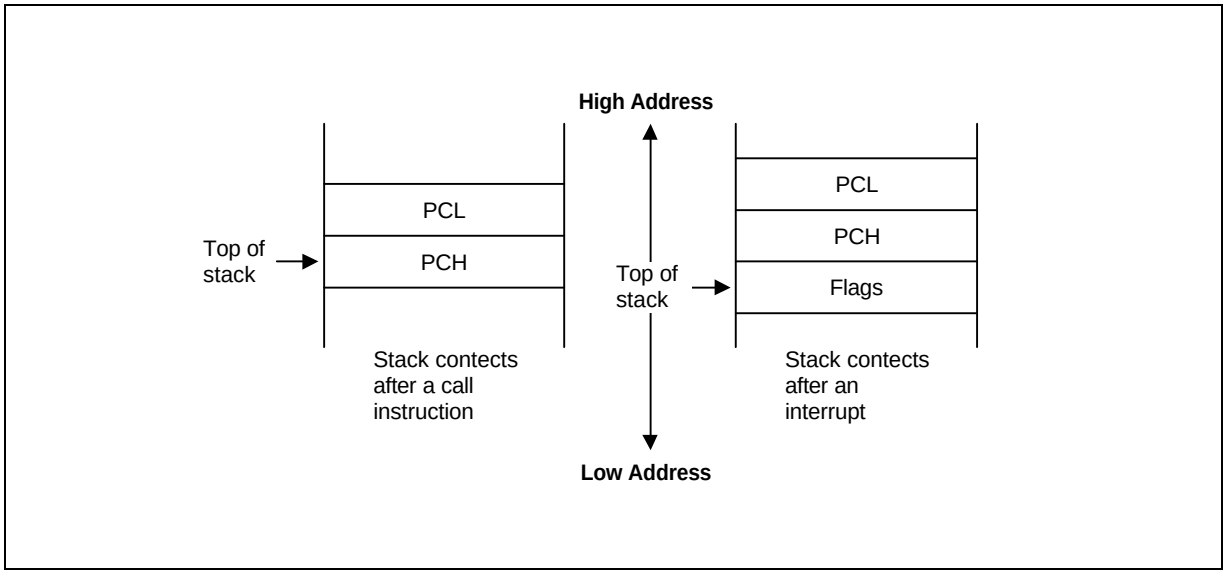


Figure 2-15. Stack Operations

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)

Register location D9H contain the 8-bit stack pointer (SPL) that is used for system stack operations. After a reset, the SPL value is undetermined. Because only internal memory 256-byte is implemented in KS88C4708/C4716, the SPL must be initialized to an 8-bit value in the range 00-FFH.

☞ 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

Instructions that are stored in program memory are fetched for execution using the program counter. 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 SAM87RC instructions may be condition codes, immediate data, or a location in the register file, program memory, or data memory.

The SAM87RC instruction set supports seven explicit addressing modes. Not all of these addressing modes are available for each instruction. The seven addressing modes and their symbols are:

— 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 (R), the operand value is the content of a specified register or register pair (see Figure 3-1).

Working register addressing differs from Register addressing in that 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).

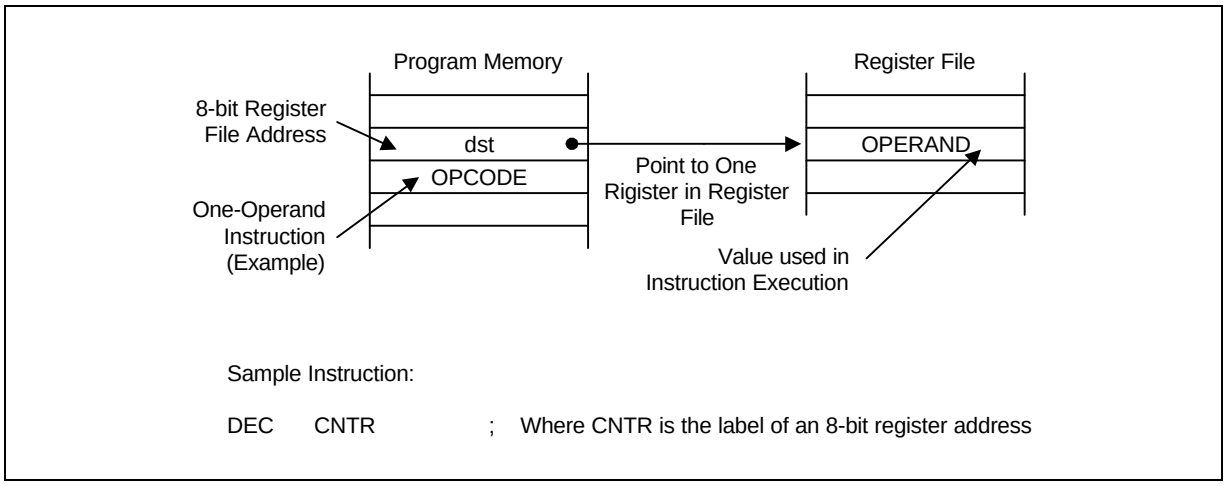


Figure 3-1. Register Addressing

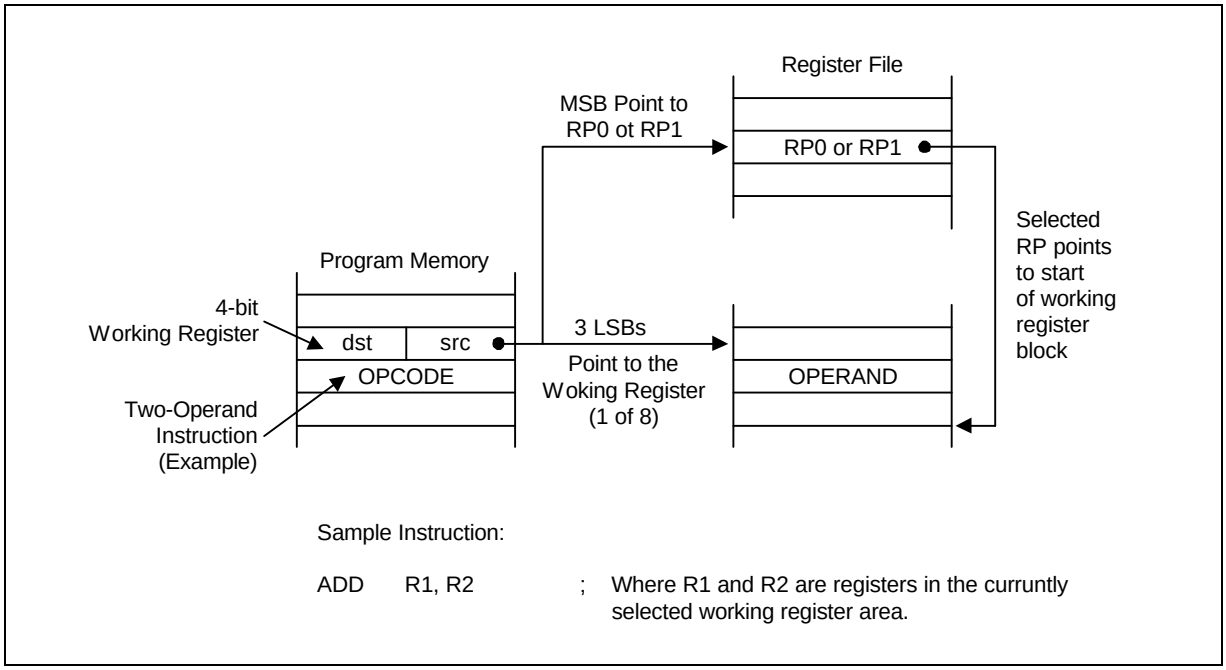


Figure 3-2. Working Register Addressing

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 (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. Please note, however, that you cannot access locations C0H–FFH in set 1 using the Indirect Register addressing mode.

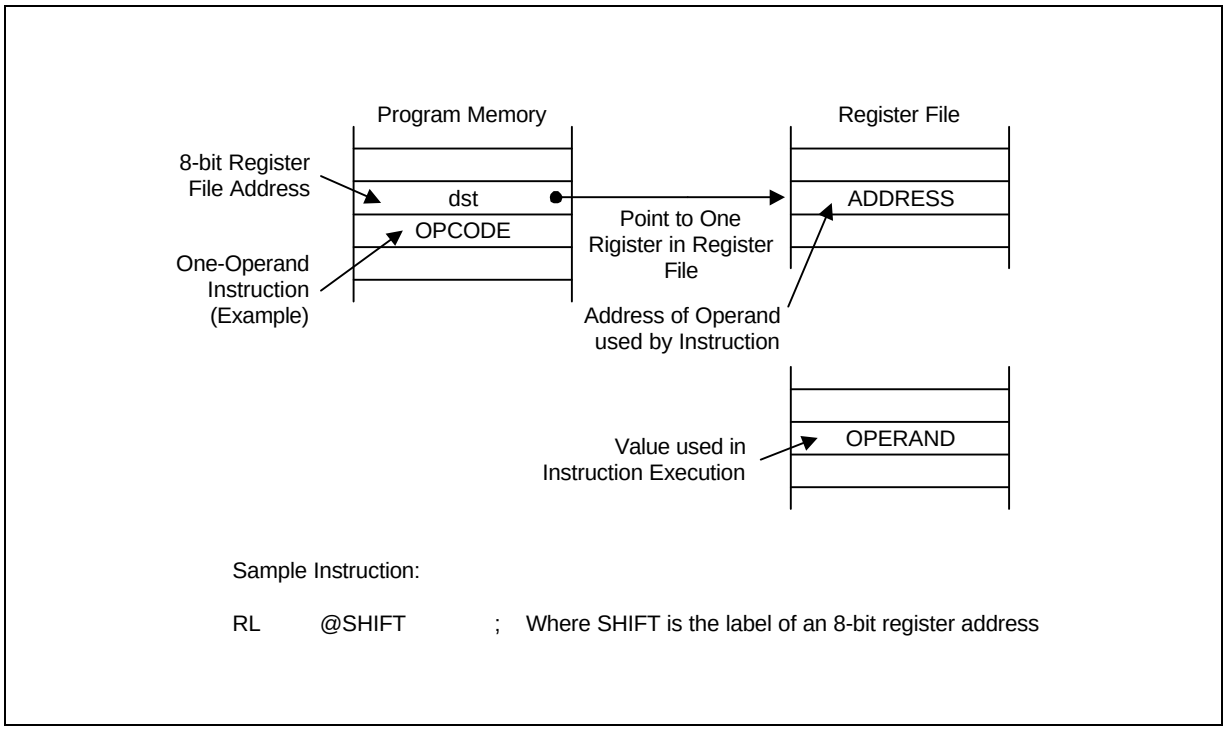


Figure 3-3. Indirect Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Continued)

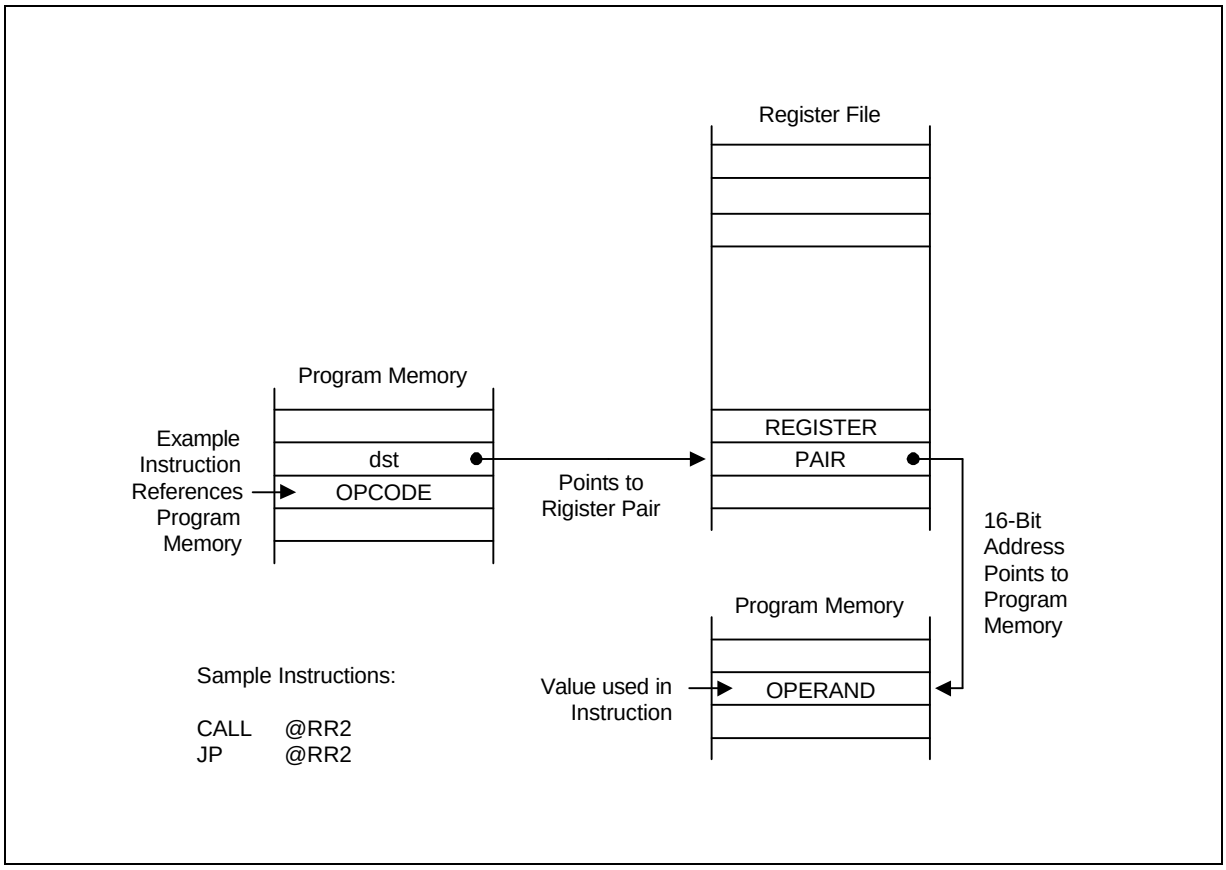


Figure 3-4. Indirect Register Addressing to Program Memory

INDIRECT REGISTER ADDRESSING MODE (Continued)

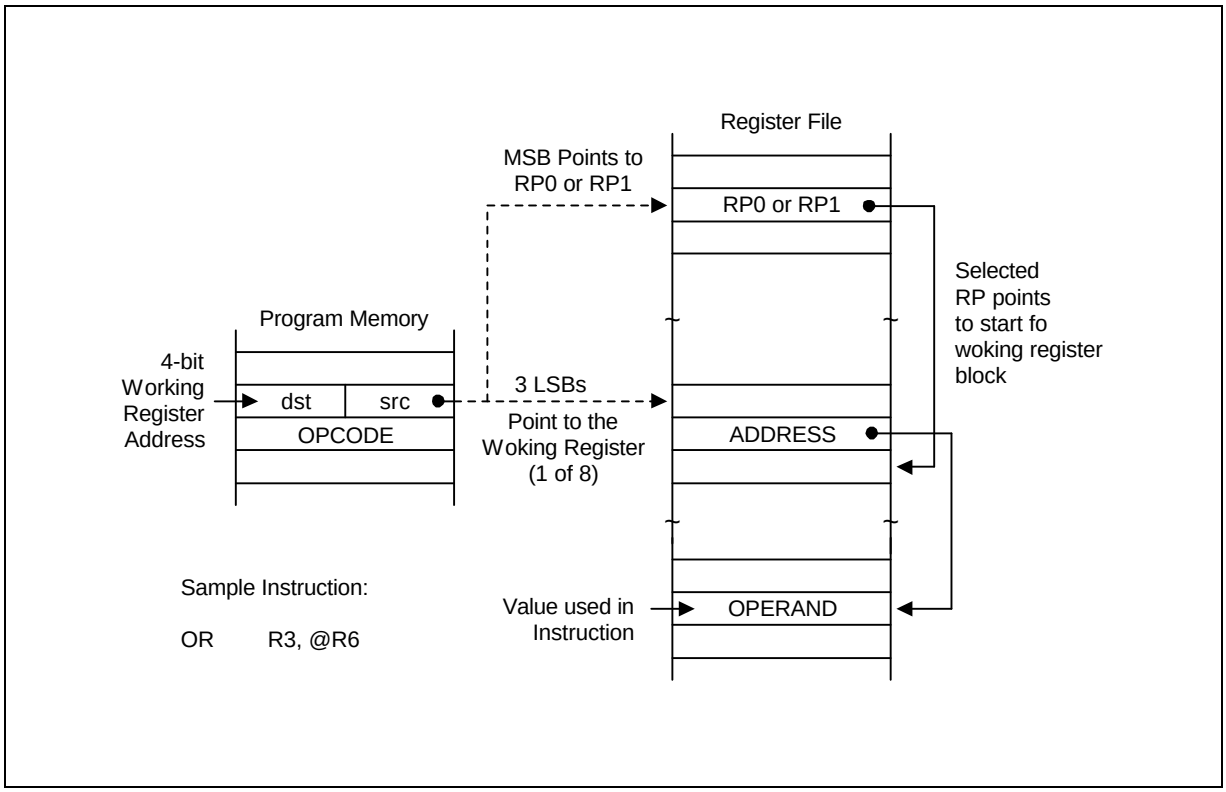


Figure 3-5. Indirect Working Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Concluded)

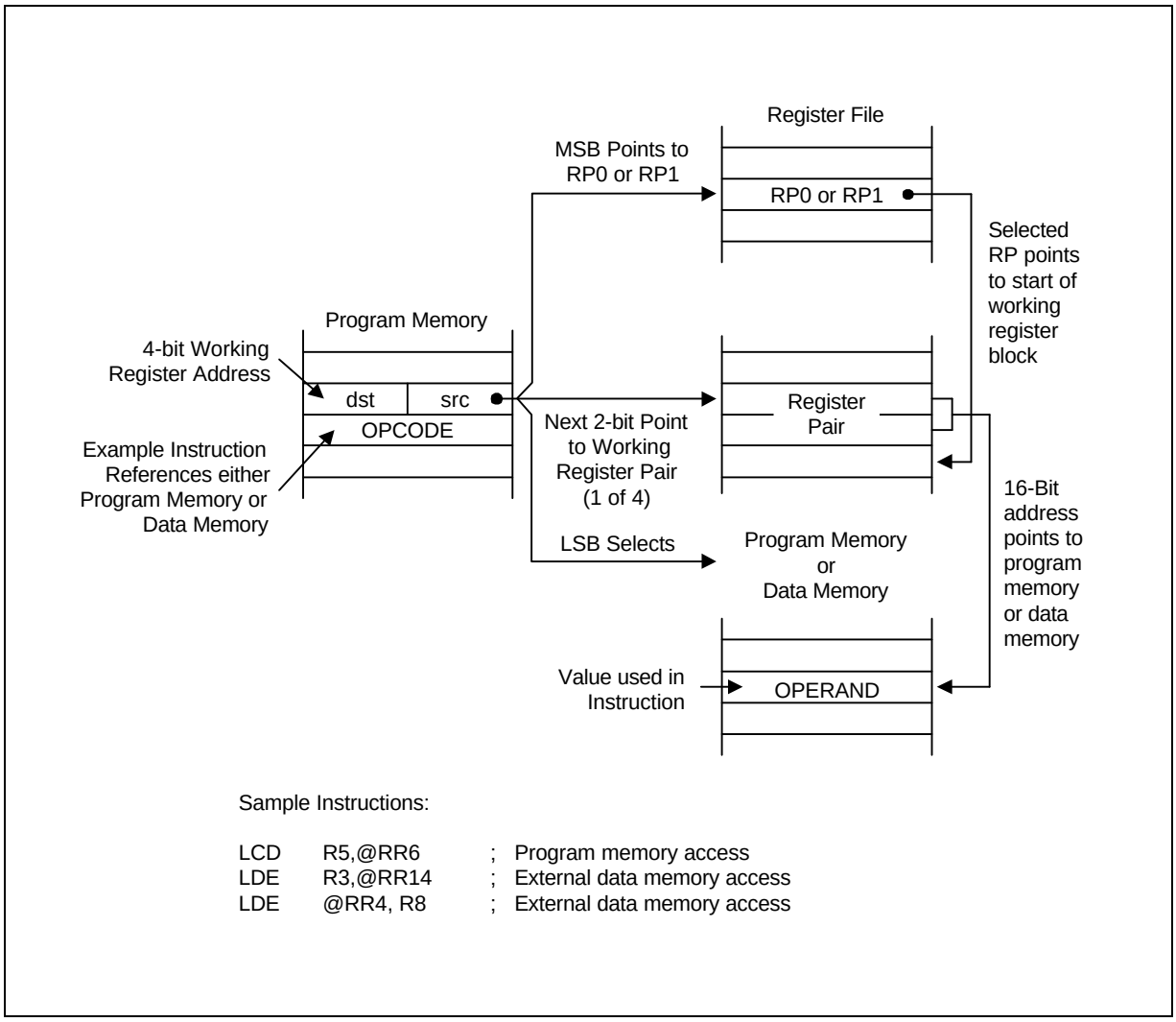


Figure 3-6. Indirect Working Register Addressing to Program or Data Memory

INDEXED ADDRESSING MODE (X)

Indexed (X) addressing mode adds an offset value to a base address during 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. Please note, however, that you cannot access locations C0H–FFH in set 1 using Indexed addressing mode.

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 that 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, when implemented.

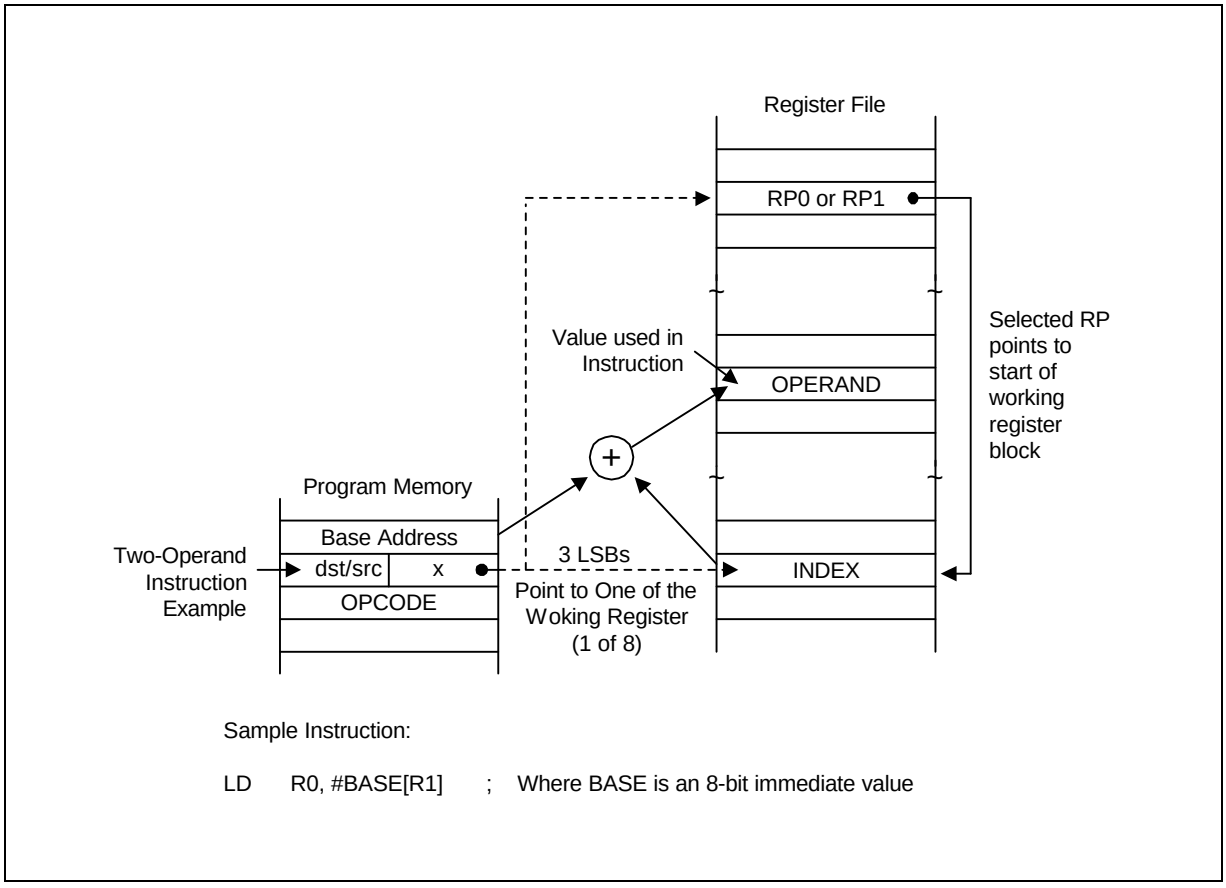


Figure 3-7. Indexed Addressing to Register File

INDEXED ADDRESSING MODE (Continued)

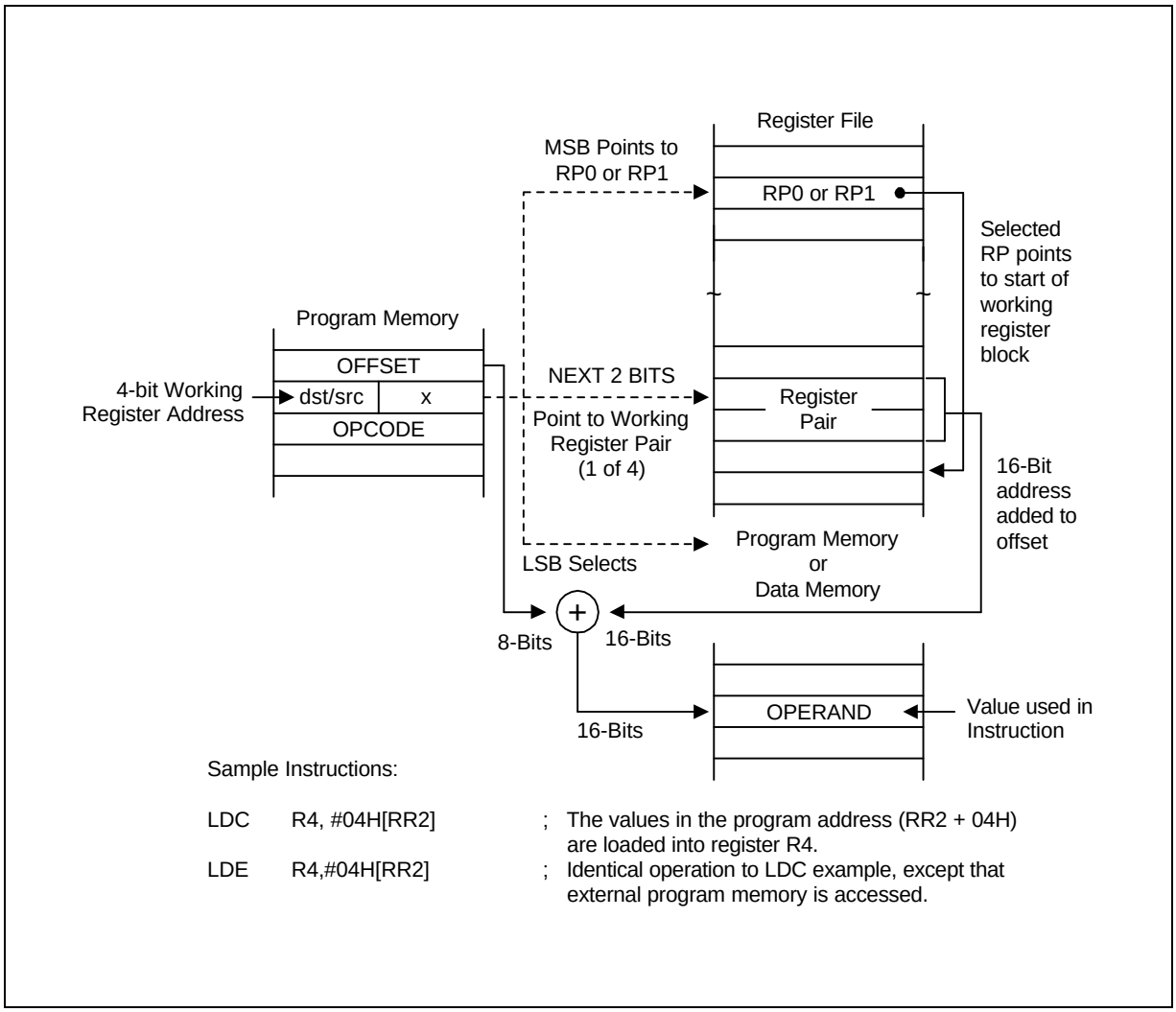


Figure 3-8. Indexed Addressing to Program or Data Memory with Short Offset

INDEXED ADDRESSING MODE (Concluded)

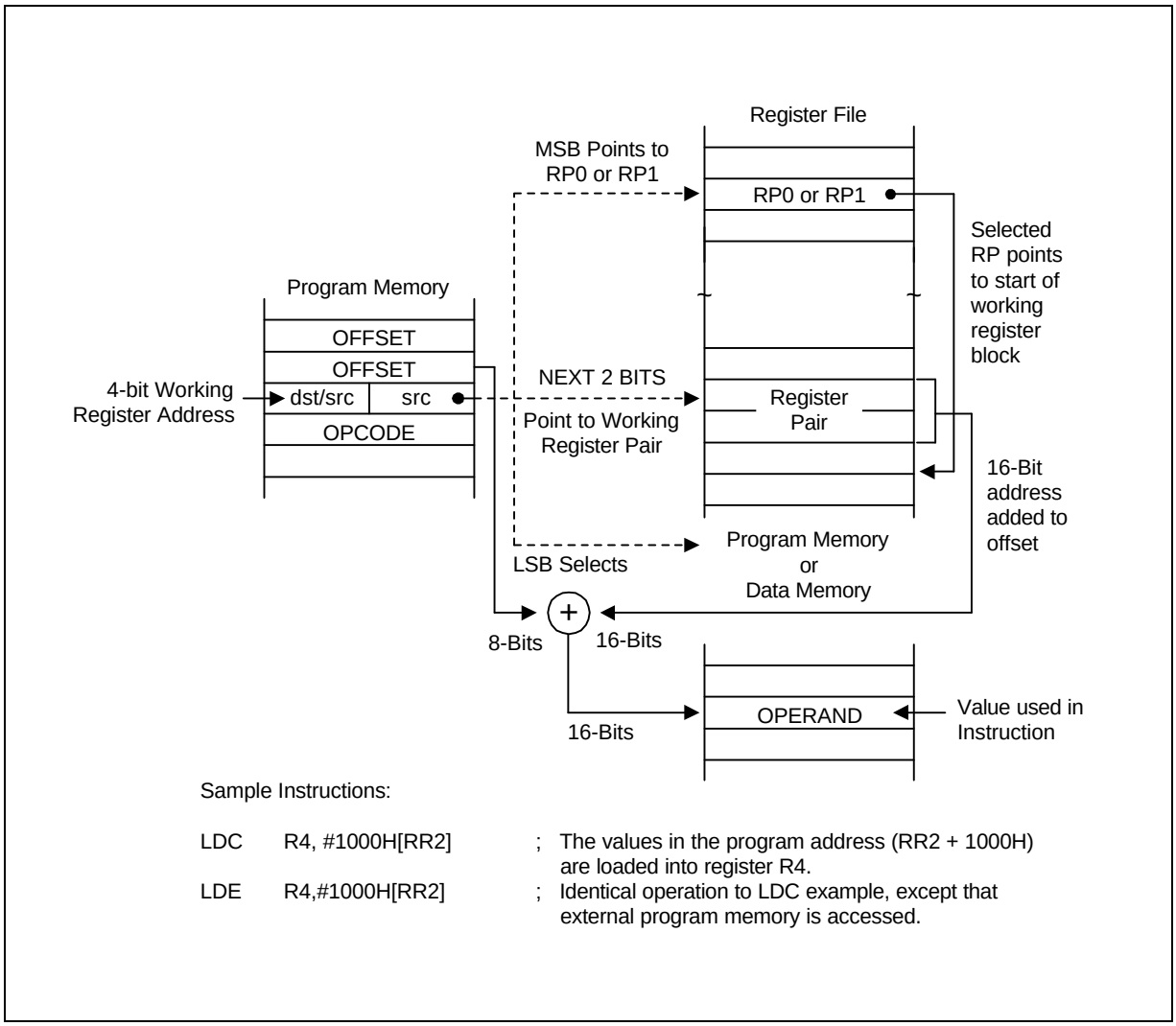


Figure 3-9. Indexed Addressing to Program or Data Memory

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.

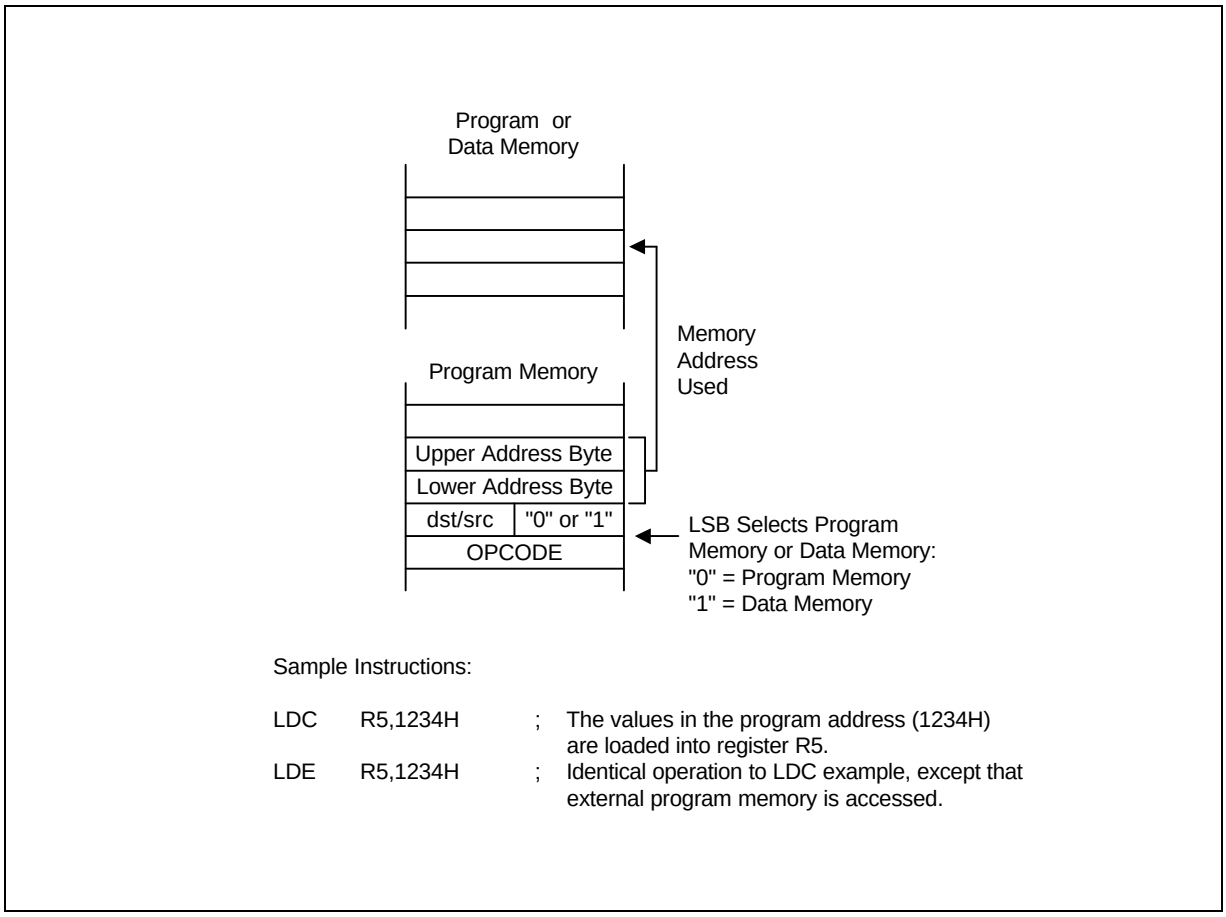


Figure 3-10. Direct Addressing for Load Instructions

DIRECT ADDRESS MODE (Continued)

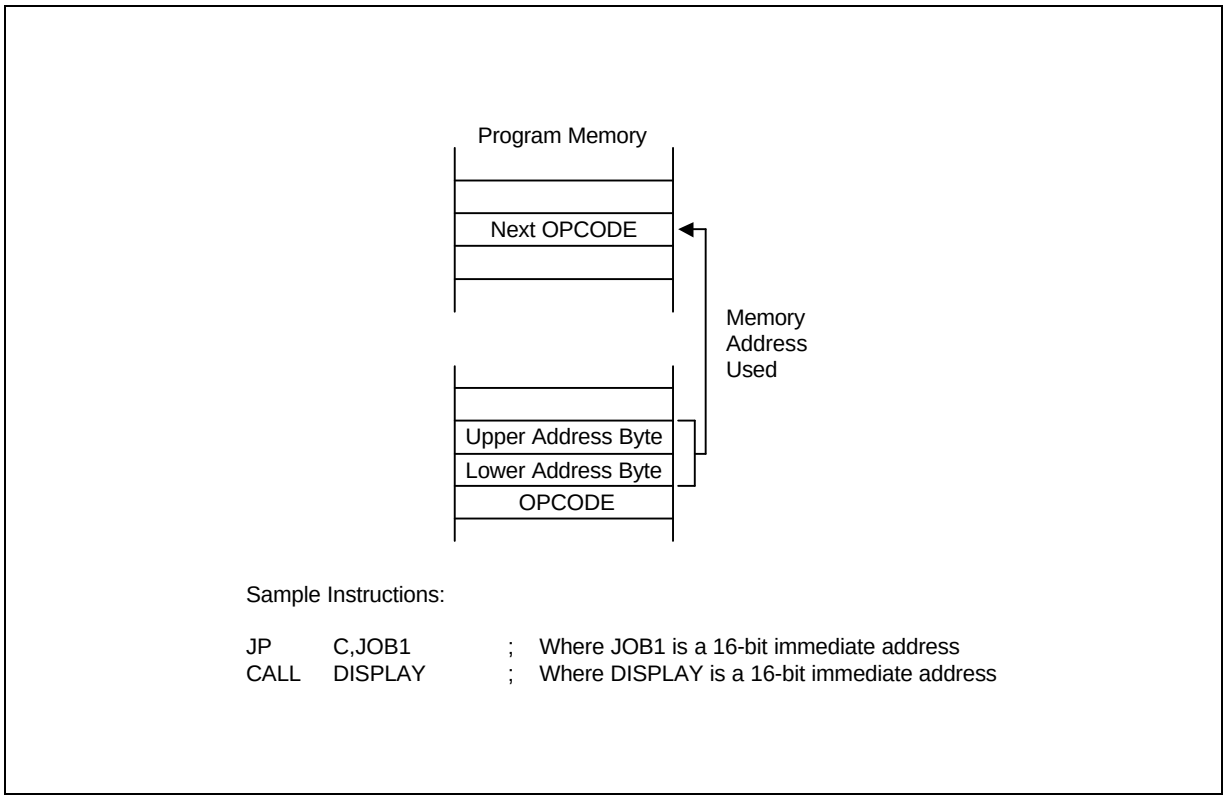


Figure 3-11. Direct Addressing for Call and Jump Instructions

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.

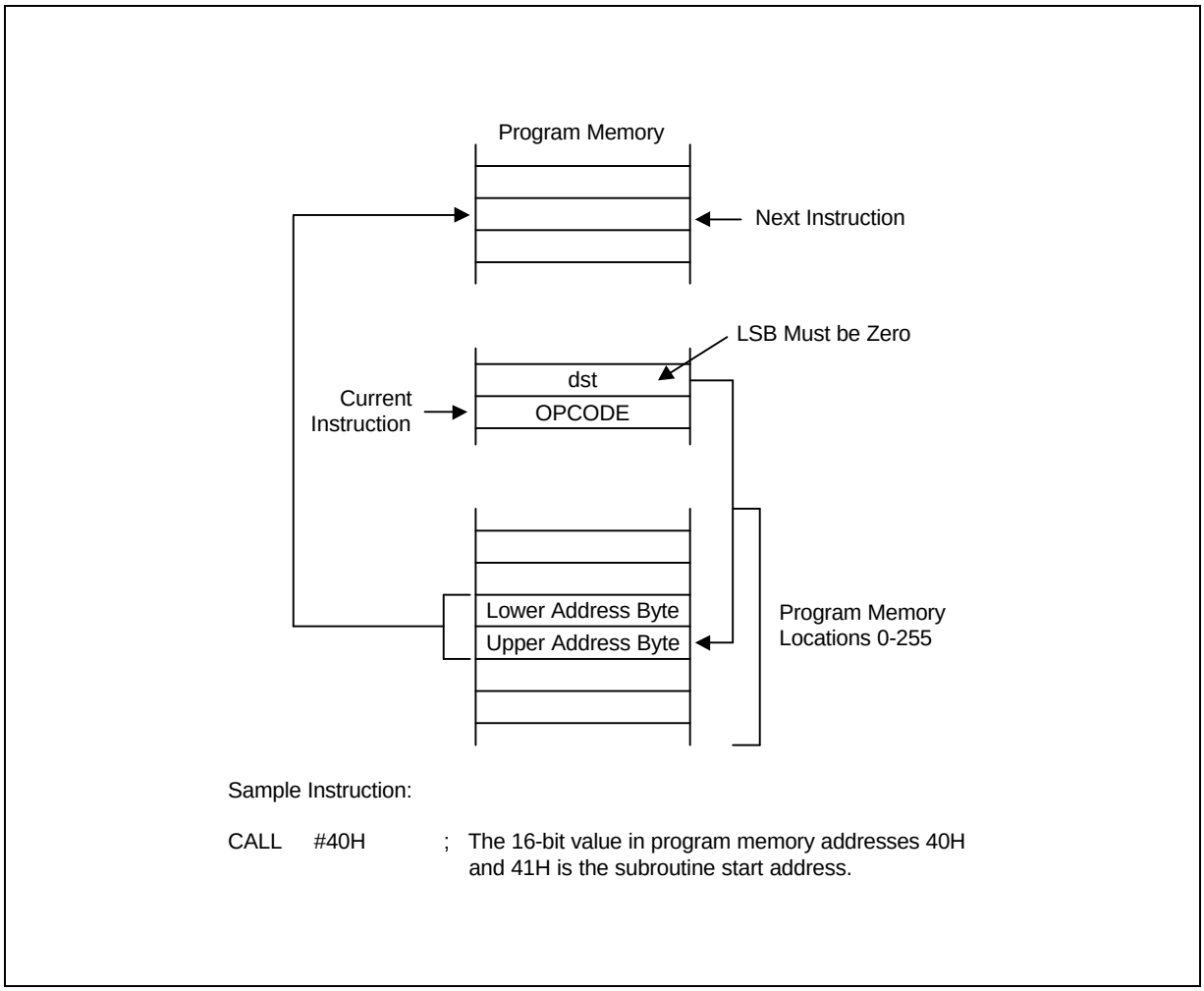


Figure 3-12. Indirect Addressing

RELATIVE ADDRESS MODE (RA)

In Relative Address (RA) mode, a twos-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.

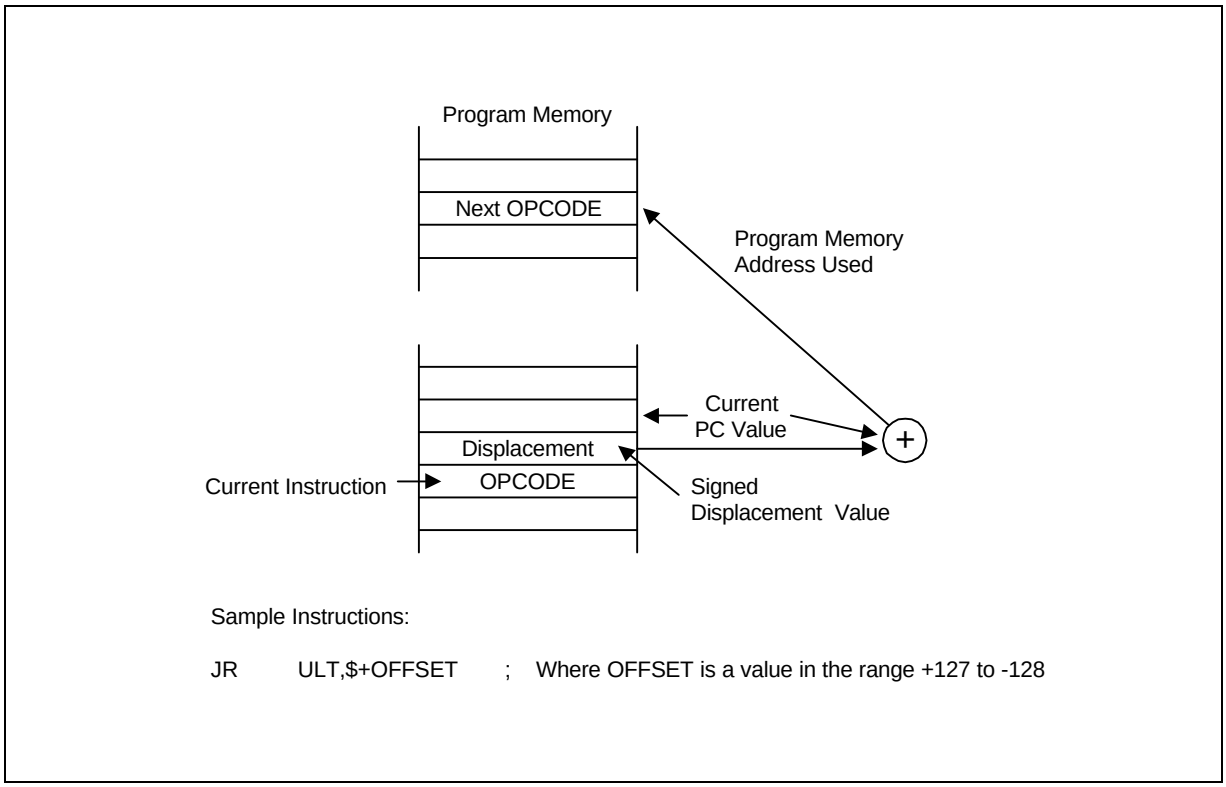


Figure 3-13. Relative Addressing

IMMEDIATE MODE (IM)

In Immediate (IM) addressing 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.

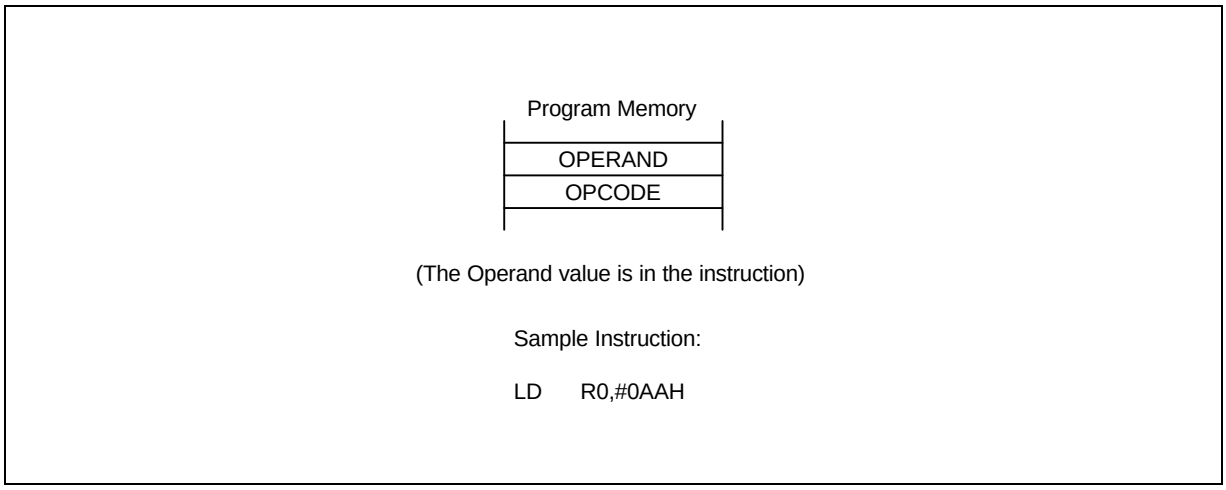


Figure 3-14. Immediate Addressing

4 CONTROL REGISTERS

OVERVIEW

In this Chapter, detailed descriptions of the KS88C4708/C4716 control registers are presented in an easy-to-read format. These descriptions will help familiarize you with the mapped locations in the register file. You can also use them as a quick-reference source when writing application programs.

System and peripheral registers are summarized in Tables 4-1, 4-2, and 4-3. 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 information about control registers is presented in the context of the various peripheral hardware descriptions in Part II of this manual.

Table 4-1. Mapped Registers

Register Name	Mnemonic	Decimal	Hex	R/W
Timer 0 counter	T0CNT	208	D0H	R
Timer 0 data register	T0DATA	209	D1H	R/W
Timer 0 control register	T0CON	210	D2H	R/W
Basic timer control register	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
Interrupt mask register	IMR	221	DDH	R/W
System mode register	SYM	222	DEH	R/W
Register page pointer	PP	223	DFH	R/W

Table 4-1. Mapped Registers (Continued)

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 0 control register	P0CON	229	E5H	R/W
Port 1 control register (high byte)	P1CONH	230	E6H	R/W
Port 1 control register (low byte)	P1CONL	231	E7H	R/W
Port 2 control register (high byte)	P2CONH	232	E8H	R/W
Port 2 control register (low byte)	P2CONL	233	E9H	R/W
Port 3 control register (high byte)	P3CONH	234	EAH	R/W
Port 3 control register (low byte)	P3CONL	235	EBH	R/W
Port 4 control register	P4CON	236	ECH	R/W
Port 2 pull-up enable register	P2PUR	237	EDH	R/W
UART control register	UARTCON	238	EEH	R/W
UART interrupt pending register	UARTPND	239	EFH	R/W
UART data register	UDATA	240	F0H	R/W
UART baud register	BRDATA	241	F1H	R/W
Timer 1 data register (high byte)	T1DATAH	242	F2H	R/W
Timer 1 data register (low byte)	T1DATAL	243	F3H	R/W
Timer 1 counter (high byte)	T1CNTH	244	F4H	R
Timer 1 counter (low byte)	T1CNTL	245	F5H	R
Timer 1 control register	T1CON	246	F6H	R/W
A/D control register	ADCON	247	F7H	R/W
A/D converter data register (high byte)	ADDATAH	248	F8H	R
A/D converter data register (low byte)	ADDATAL	249	F9H	R
8-bit prescaler for buzzer output	BUZPS	250	FAH	R/W
Locations FBH is not mapped.				
Reserved (factory test use only)	FTEST	252	FCH	R/W
Basic timer counter register	BTCNT	253	FDH	R
Reserved	EMT	254	FEH	R/W
Interrupt priority register	IPR	255	FFH	R/W

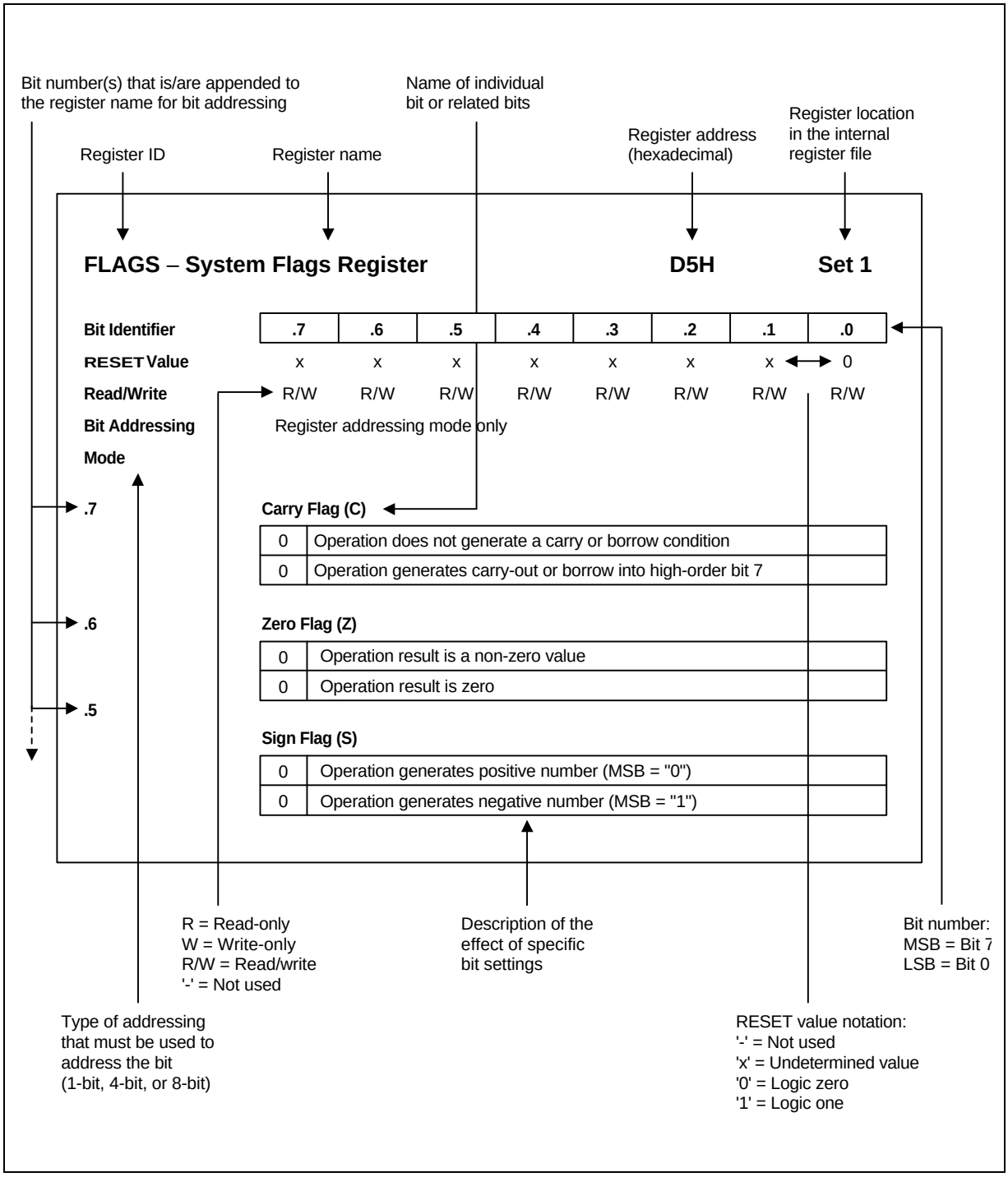


Figure 4-1. Register Description Format

ADCON — A/D Converter Control Register F7H Set 1

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 Not used for the KS88C4708/C4716

.6–.4 Analog Input Pin Selection Bits

0	0	0	ADC0 (P3.0)
0	0	1	ADC1 (P3.1)
0	1	0	ADC2 (P3.2)
0	1	1	ADC3 (P3.3)
1	0	0	ADC4 (P3.4)
1	0	1	ADC5 (P3.5)
1	1	0	ADC6 (P3.6)
1	1	1	ADC7 (P3.7)

.3 End-of-Conversion Status Bit

0	A/D conversion not complete (when read)
1	A/D conversion is complete (when read)

.2–.1 Conversion Speed Selection Bits

0	0	$f_{OSC}/16$ ($f_{OSC} \leq 12$ MHz)
0	1	$f_{OSC}/8$ ($f_{OSC} \leq 12$ MHz)
1	0	$f_{OSC}/4$ ($f_{OSC} \leq 10$ MHz)
1	1	f_{OSC} ($f_{OSC} \leq 2.5$ MHz)

.0 Conversion Start Bit

0	No meaning
1	Starting

NOTE: To configure an A/D converter input channel, you must also make the proper setting in the port 3 control register, P3CONL (ADC0–ADC3) or P3CONH (ADC4–ADC7). Only one input can be configured at a time.

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 Enable Bits

1	0	1	0	Disable watchdog timer function
Others				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

.1 Basic Timer Counter Clear Bit (note)

0	No effect
1	Clear basic timer counter value

.0 Basic Timer and Timer 0 Divider Clear Bit (note)

0	No effect
1	Clear both dividers

NOTE: When you write a "1" to BTCON.0 (or BTCON.1), the basic timer divider (or basic timer counter) is cleared, the bit is then cleared automatically to “0”.

BUZPS — 6-Bit Prescaler for Buzzer Output FAH 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 Buzzer Output Enable Bit

0	Disable buzzer output (buzzer off)
1	Enable buzzer output (buzzer on)

.6 Buzzer Clock Selection Bit

0	Divided by 256 (fx/256)
1	Divided by 64 (fx/64)

.5–.0 6-Bit Prescaler

0	0	0	0	0	0	Divided by 2 [fx/(256 or 64)]
0	0	0	0	0	1	Divided by 4 [fx/(256 or 64)]
0	0	0	0	1	0	Divided by 6 [fx/(256 or 64)]
0	0	0	0	1	1	Divided by 8 [fx/(256 or 64)]
• • • •						Divided by 2 × (n + 1) [fx/(256 or 64)]
1	1	1	1	1	1	Divided by 128 [fx/(256 or 64)]

NOTE: When P1.3/BUZ is used as buzzer output pin, the initial value is 0. When the bit 7 of BUZPS is set to 0, the output of P1.3/BUZ is now.

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
Read/Write	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 function
1	Disable IRQ for main system oscillator wake-up function

.6 and .5 Not used for the KS88C4708/C4716

.4 and .3 CPU Clock (System Clock) Selection Bits (1)

0	0	$f_{OSC}/16$
0	1	$f_{OSC}/8$
1	0	$f_{OSC}/4$
1	1	f_{OSC} (non-divided)

.2–.0 Not used for the KS88C4708/C4716

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. f_{OSC} means oscillator frequency.

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 to "0" during an interrupt return (IRET)
1	Set automatically to "1" during a fast interrupt service routine

.0 Bank Address Selection Flag (BA)

0	Bank 0 is selected
1	Bank 1 is selected

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	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 Interrupt Level 7 (IRQ7) Enable Bit; P2.4-P2.7 External Interrupts

0	Disable IRQ7 interrupts
1	Enable IRQ7 interrupts

.6 Interrupt Level 6 (IRQ6) Enable Bit; P2.3 External Interrupts

0	Disable IRQ6 interrupts
1	Enable IRQ6 interrupts

.5 Interrupt Level 5 (IRQ5) Enable Bit; P2.2 External Interrupts

0	Disable IRQ5 interrupts
1	Enable IRQ5 interrupts

.4 Interrupt Level 4 (IRQ4) Enable Bit; P2.1 External Interrupts

0	Disable IRQ4 interrupts
1	Enable IRQ4 interrupts

.3 Interrupt Level 3 (IRQ3) Enable Bit; P2.0 External Interrupts

0	Disable IRQ3 interrupts
1	Enable IRQ3 interrupts

.2 Interrupt Level 2 (IRQ2) Enable Bit; UART External Interrupts

0	Disable IRQ2 interrupts
1	Enable IRQ2 interrupts

.1 Interrupt Level 1 (IRQ1) Enable Bit; Timer 1 Interrupts

0	Disable IRQ1 interrupts
1	Enable IRQ1 interrupts

.0 Interrupt Level 0 (IRQ0) Enable Bit; Timer 0 Interrupts

0	Disable IRQ0 interrupts
1	Enable IRQ0 interrupts

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

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

0	0	0	Group priority undefined
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	Group priority undefined

.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

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	0	0
Read/Write	R	R	R	R	R	R	R	R
Addressing Mode	Register addressing mode only							

.7 Interrupt Level 7 (IRQ7) Request Pending Bit; P2.4-P2.7 External Interrupts

0	No IRQ7 interrupt pending
1	IRQ7 interrupt is pending

.6 Interrupt Level 6 (IRQ6) Request Pending Bit; P2.3 External Interrupts

0	No IRQ6 interrupt pending
1	IRQ6 interrupt is pending

.5 Interrupt Level 5 (IRQ5) Request Pending Bit; P2.2 External Interrupts

0	No IRQ5 interrupt pending
1	IRQ5 interrupt is pending

.4 Interrupt Level 4 (IRQ4) Request Pending Bit; P2.1 External Interrupts

0	No IRQ4 interrupt pending
1	IRQ4 interrupt is pending

.3 Interrupt Level 3 (IRQ3) Request Pending Bit; P2.0 External Interrupts

0	No IRQ3 interrupt pending
1	IRQ3 interrupt is pending

.2 Interrupt Level 2 (IRQ2) Request Pending Bit; UART Interrupts

0	No IRQ2 interrupt pending
1	IRQ2 interrupt is pending

.1 Interrupt Level 1 (IRQ1) Request Pending Bit; Timer 1 Interrupts

0	No IRQ1 interrupt pending
1	IRQ1 interrupt is pending

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

0	No IRQ0 interrupt pending
1	IRQ0 interrupt is pending

P0CON — Port 0 Control Register E5H 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 **P0.4-P0.7 Configuration Bits**

0	0	x	0	Input mode
0	1	x	0	Input mode, pull-up enable
0	0	0	1	Push-pull output
0	0	0	1	Open-drain output
0	1	1	1	Open-drain output, pull-up enable

.3 – .0 **Port 0, P0.3-P0.0 Configuration Bits**

0	0	x	0	Input mode
0	1	x	0	Input mode, pull-up enable
0	0	0	1	Push-pull output
0	0	0	1	Open-drain output
0	1	1	1	Open-drain output, pull-up enable

P1CONH — Port 1 Control Register (High Byte) E6H Set 1

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

.7 – .4

Not used for KS88C4708/C4716

.3 and .2 **P1.5/TxD Configuration Bits**

0	0	Schmitt trigger input
0	1	Schmitt trigger input; pull-up resistor enable
1	0	Push-pull output
1	1	Alternative function (TxD output)

.1 and .0 **P1.4/RxD Configuration Bits**

0	0	Schmitt trigger input (RxD input)
0	1	Schmitt trigger input; pull-up resistor enable
1	0	Push-pull output
1	1	Alternative function (RxD output)

P1CONL — Port 1 Control Register (Low Byte) E7H 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 **P1.3/BUZ Configuration Bits**

0	0	Schmitt trigger input
0	1	Schmitt trigger input; pull-up resistor enable
1	0	Push-pull output
1	1	Alternative function (BUZ output)

.5 and .4 **P1.2/T1 Configuration Bits**

0	0	Schmitt trigger input (or T1 capture input)
0	1	Schmitt trigger input; pull-up resistor enable (or T1 capture input)
1	0	Push-pull output
1	1	Alternative function (T1 output: match or PWM)

.3 and .2 **P1.1/T1CK Configuration Bits**

0	0	Schmitt trigger
0	1	Schmitt trigger input; pull-up resistor enable
1	0	Push-pull output
1	1	Alternative function (T1CK input)

.1 and .0 **P1.0/T0 Configuration Bits**

0	0	Schmitt trigger (or T0 capture input)
0	1	Schmitt trigger input; pull-up resistor enable (or T0 capture input)
1	0	Push-pull output
1	1	Alternative function (T0 output: match or PWM)

P2CONH — Port 2 Control Register (High Byte) E8H 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 **P2.7/INT7 Configuration Bits**

0	0	Input mode; INT7 interrupt disabled
0	1	Input mode; interrupt on falling edges
1	0	Push-pull output
1	1	Input mode; interrupt on rising edges

.5 and .4 **P2.6/INT6 Configuration Bits**

0	0	Input mode; INT6 interrupt disabled
0	1	Input mode; interrupt on falling edges
1	0	Push-pull output
1	1	Input mode; interrupt on rising edges

.3 and .2 **P2.5/INT5 Configuration Bits**

0	0	Input mode; INT5 interrupt disabled
0	1	Input mode; interrupt on falling edges
1	0	Push-pull output
1	1	Input mode; interrupt on rising edges

.1 and .0 **P2.4/INT4 Configuration Bits**

0	0	Input mode; INT4 interrupt disabled
0	1	Input mode; interrupt on falling edges
1	0	Push-pull output
1	1	Input mode; interrupt on rising edges

`P2CONL — Port 2 Control Register (Low Byte) E9H 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 **P2.3/INT3 Configuration Bits**

0	0	Input mode; INT3 interrupt disabled
0	1	Input mode; interrupt on falling edges
1	0	Push-pull output
1	1	Input mode; interrupt on rising edges

.5 and .4 **P2.2/INT2 Configuration Bits**

0	0	Input mode; INT2 interrupt disabled
0	1	Input mode; interrupt on falling edges
1	0	Push-pull output
1	1	Input mode; interrupt on rising edges

.3 and .2 **P2.1/INT1 Configuration Bits**

0	0	Input mode; INT1 interrupt disabled
0	1	Input mode; interrupt on falling edges
1	0	Push-pull output
1	1	Input mode; interrupt on rising edges

.1 and .0 **P2.0/INT0 Configuration Bits**

0	0	Input mode; INT0 interrupt disabled
0	1	Input mode; interrupt on falling edges
1	0	Push-pull output
1	1	Input mode; interrupt on rising edges

P2PUR — Port 2 Pull-up Resistor Enable Register EDH 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 P2.7/INT7 Pull-up Resistor Enable Bit

0	Disable pull-up
1	Enable pull-up

.6 P2.6/INT6 Pull-up Resistor Enable Bit

0	Disable pull-up
1	Enable pull-up

.5 P2.5/INT5 Pull-up Resistor Enable Bit

0	Disable pull-up
1	Enable pull-up

.4 P2.4/INT4 Pull-up Resistor Enable Bit

0	Disable pull-up
1	Enable pull-up

.3 P2.3/INT3 Pull-up Resistor Enable Bit

0	Disable pull-up
1	Enable pull-up

.2 P2.2/INT2 Pull-up Resistor Enable Bit

0	Disable pull-up
1	Enable pull-up

.1 P2.1/INT1 Pull-up Resistor Enable Bit

0	Disable pull-up
1	Enable pull-up

.0 P2.0/INT0 Pull-up Resistor Enable Bit

0	Disable pull-up
1	Enable pull-up

P3CONH — Port 3 Control Register (High Byte) 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 **P3.7/ADC7 Configuration Bits**

0	0	Input mode; A/D converter off
0	1	Input mode; pull-up resistor enabled; A/D converter off
1	0	Push-pull output
1	1	A/D converter input (ADC7); normal input off

.5 and .4 **P3.6/ADC6 Configuration Bits**

0	0	Input mode; A/D converter off
0	1	Input mode; pull-up resistor enabled; A/D converter off
1	0	Push-pull output
1	1	A/D converter input (ADC6); normal input off

.3 and .2 **P3.5/ADC5 Configuration Bits**

0	0	Input mode; A/D converter off
0	1	Input mode; pull-up resistor enabled; A/D converter off
1	0	Push-pull output
1	1	A/D converter input (ADC5); normal input off

.1 and .0 **P3.4/ADC4 Configuration Bits**

0	0	Input mode; A/D converter off
0	1	Input mode; pull-up resistor enabled; A/D converter off
1	0	Push-pull output
1	1	A/D converter input (ADC4); normal input off

P3CONL — Port 3 Control Register (Low Byte) EBH 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 **P3.3/ADC3 Configuration Bits**

0	0	Input mode; A/D converter off
0	1	Input mode; pull-up resistor enabled; A/D converter off
1	0	Push-pull output
1	1	A/D converter input (ADC3); normal input off

.5 and .4 **P3.2/ADC2 Configuration Bits**

0	0	Input mode; A/D converter off
0	1	Input mode; pull-up resistor enabled; A/D converter off
1	0	Push-pull output
1	1	A/D converter input (ADC2); normal input off

.3 and .2 **P3.1/ADC1 Configuration Bits**

0	0	Input mode; A/D converter off
0	1	Input mode; pull-up resistor enabled; A/D converter off
1	0	Push-pull output
1	1	A/D converter input (ADC1); normal input off

.1 and .0 **P3.0/ADC0 Configuration Bits**

0	0	Input mode; A/D converter off
0	1	Input mode; pull-up resistor enabled; A/D converter off
1	0	Push-pull output
1	1	A/D converter input (ADC0); normal input off

P4CON — Port 4 Control Register ECH 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 **P4.3 Configuration Bits**

0	0	Input mode
0	1	Input mode, pull-up enable
1	0	Push-pull output
1	1	Open-drain output

.5 and .4 **P4.2 Configuration Bits**

0	0	Input mode
0	1	Input mode, pull-up enable
1	0	Push-pull output
1	1	Open-drain output

.3 and .2 **P4.1 Configuration Bits**

0	0	Input mode
0	1	Input mode, pull-up enable
1	0	Push-pull output
1	1	Open-drain output

.1 and .0 **P4.0 Configuration Bits**

0	0	Input mode
0	1	Input mode, pull-up enable
1	0	Push-pull output
1	1	Open-drain output

NOTE:

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 (note)

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

0	0	0	0	Source: page 0 (note)

NOTE: In the KS88C4708/C4716 microcontroller, a paged expansion of the internal register file is not implemented. For this reason, only page 0 settings are valid. Register page pointer values for the source and destination register page are automatically set to '0000B' following a hardware reset. These values should not be changed during normal operation.

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 24 8-byte working register areas 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 the KS88C4708/C4716

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 24 8-byte working register areas 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 the KS88C4708/C4716

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.

NOTE: If you only use the internal register file as stack area, SPH can serve as a general-purpose register. To avoid possible overflows or underflows of the SPL register by operations that increment or decrement the stack, we recommend that you initialize SPL with the value "FFH" instead of "00H". If you use external memory as stack area, the stack pointer requires a full 16-bit address.

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	0	0	0	0
Read/Write	–	–	–	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 – .5

Not used for the KS88C4708/C4716

.4 and .2

Fast Interrupt Level Selection Bits (1)

0	0	0	IRQ0
0	0	1	IRQ1
0	1	0	IRQ2
0	1	1	IRQ3
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 all interrupts
1	Enable all interrupts

NOTES:

1. Only one interrupt level can be selected at a time 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 D2H 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 T0 Input Clock Selection Bits

0	0	$f_{OSC}/4096$
0	1	$f_{OSC}/256$
1	0	$f_{OSC}/8$
1	1	$f_{OSC}/1$

.5 and .4 T0 Operating Mode Selection Bits

0	0	Interval timer mode
0	1	Capture mode (capture on rising edge, counter running, OVF)
1	0	Capture mode (capture on falling edge, counter running, OVF)
1	1	PWM mode (OVF interrupt can occur)

.3 T0 Counter Clear Bit

0	No effect
1	Clear the timer 0 counter (when write)

.2 T0 Overflow Interrupt Enable Bit

0	Disable overflow interrupt
1	Enable overflow interrupt

.1 T0 Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 T0 Interrupt Pending Bit (Capture or Match Interrupt)

0	No interrupt pending
0	*Clear this pending bit (write)*
1	Interrupt is pending

T1CON — Timer Module 1 Control Register F6H 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 only							

.7 – .5 T1 Input Clock Selection Bits:

0	0	0	$f_{OSC}/1024$
0	0	1	$f_{OSC}/256$
0	1	0	$f_{OSC}/64$
0	1	1	$f_{OSC}/8$
1	0	0	$f_{OSC}/1$
1	0	1	External clock (T1CK) falling
1	1	0	External clock (T1CK) rising
1	1	1	Counter stop

.4 – .3 T1 Operating Mode Selection Bits:

0	0	Interval mode
0	1	Capture mode (capture on rising counter running, OVF can occur)
1	0	Capture mode (capture on edge, counter running, OVF can occur)
1	1	PWM mode (OVF interrupt can occur)

.2 T1 Counter Clear Bit:

0	No effect
0	Clear the timer 1 (when write)

.1 T1 Match/Capture Interrupt

0	Disable interrupt
1	Enable interrupt

.0 T1 Overflow Interrupt Enable

0	Disable overflow interrupt
1	Enable overflow interrupt

UARTCON — UART Control Register EEH 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 – .6

0	0	Mode 0: shift register (f_{OSC}/(16 × (BRDATA + 1))
0	1	Mode 1: 8-Bit UART (f_{OSC}/(16 × (BRDATA + 1))
1	0	Mode 2: 9-Bit UART f_{OSC}/16
1	1	Mode 3: 9-Bit UART (f_{OSC}/(16 × (BRDATA + 1))

.5 Multiprocessor Communication Enable Bit (for modes 2 and 3 only)

0	Disable
1	Enable

.4 Serial Data Receive Enable Bit

0	Disable
1	Enable

.3 Location of the 9th data bit to be transmitted in UART mode 2 or 3 (“0” or “1”)

.2 Location of the 9th data bit to be transmitted in UART mode 2 or 3 (“0” or “1”)

.1 Receive Interrupt Enable Bit

0	Disable Rx interrupt
1	Enable Rx interrupt

.0 Transmit Interrupt Enable Bit

0	Disable Tx interrupt
1	Enable Tx interrupt

NOTES:

1. In mode 2 or 3, if the MCE bit is set to "1" then the receive interrupt will not be activated if the received 9th data bit "0".
   In mode 1, if MCE = "1" the receive interrupt will not be activated if a valid stop bit was not received. In mode 0, the MCE bit should be “0”.
2. The descriptions for 8-bit and 9-bit UART mode do not include start and stop bits for serial data receive and transmit.

UARTPND — UART Interrupt Pending Register EFH 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 the KS88C4708/C4716

.1 Rx Interrupt Pending Bit

0	No interrupt pending
0	*Clear this pending bit (when write)*
1	Interrupt is pending

.0 Tx Interrupt Pending Bit

0	No interrupt pending
0	*Clear this pending bit (when write)*
1	Interrupt is pending

NOTES:

1. To clear an interrupt pending bit, you must write a “0” to the corresponding UARTPND bit location.
2. To avoid programming errors, we recommend using load instruction when manipulating UARTPND values.

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–IRQ7, 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 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.

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.

When a service routine starts, the respective pending bit is either cleared automatically by hardware or "manually" by the 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 (V_1) + one source (S_1)

Type 2: One level (IRQn) + one vector (V_1) + multiple sources ($S_1 – S_n$)

Type 3: One level (IRQn) + multiple vectors ($V_1 – V_n$) + multiple sources ($S_1 – S_n$, $S_{n+1} – S_{n+m}$)

In the KS88C4708/C4716 microcontroller, only interrupt types 1 and 3 are implemented.

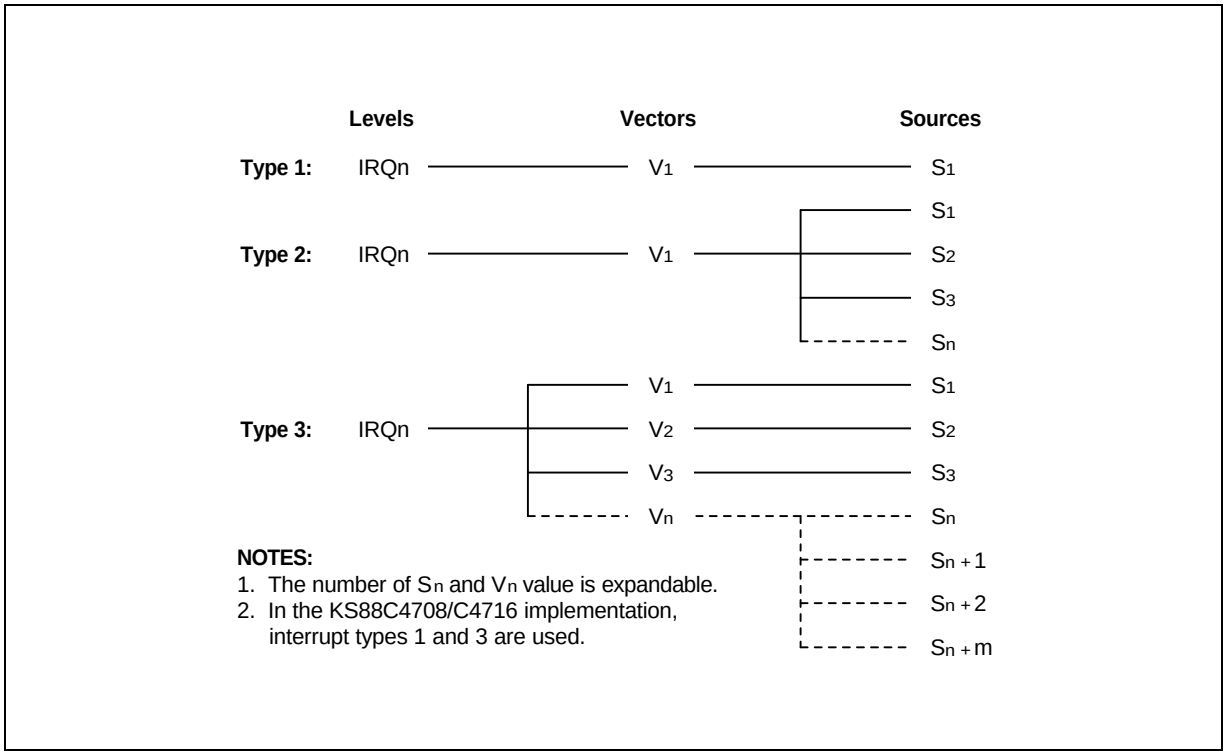


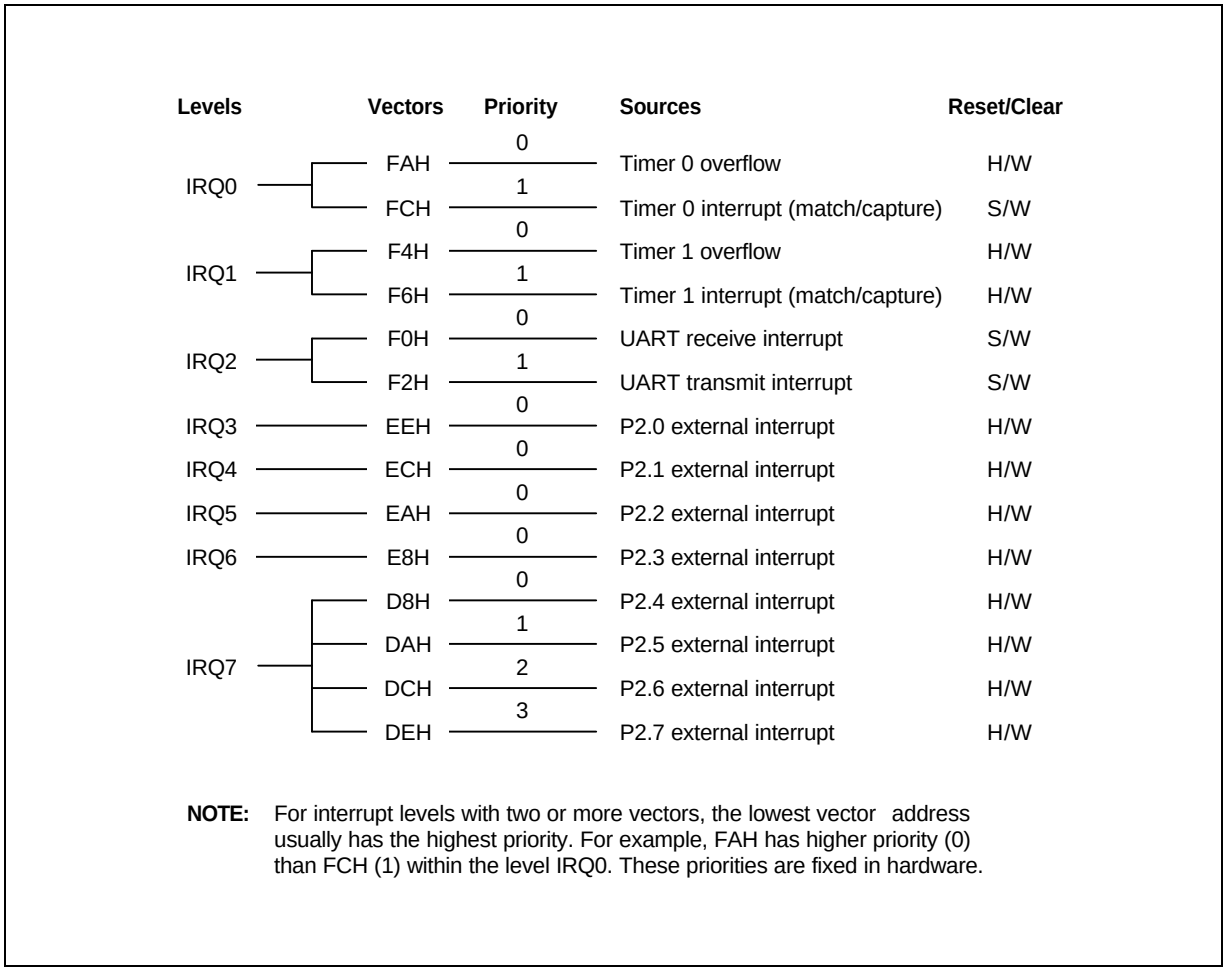
Figure 5-1. KS88-Series Interrupt Types

KS88C4708/C4716 INTERRUPT STRUCTURE

The KS88C4708/C4716 microcontroller supports up to 14 interrupt sources. Each interrupt source has as corresponding interrupt vector address. All eight levels are used in the interrupt structure: The device-specific interrupt structure 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. (The relative priorities of multiple interrupts within a single level are set in hardware.)

When an interrupt request is granted, interrupt processing starts: Subsequent 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.



NOTE: For interrupt levels with two or more vectors, the lowest vector address usually has the highest priority. For example, FAH has higher priority (0) than FCH (1) within the level IRQ0. These priorities are fixed in hardware.

Figure 5-2. KS88C4708/C4716 Interrupt Structure

INTERRUPT VECTOR ADDRESSES

All interrupt vector addresses for the KS88C4708/C4716 interrupt structure are stored in the vector address area of the ROM (0H–FFH). Unused locations in this range can be used as normal program memory. If you do so, you must be careful not to overwrite any of the vector addresses. The program reset address is 0100H.

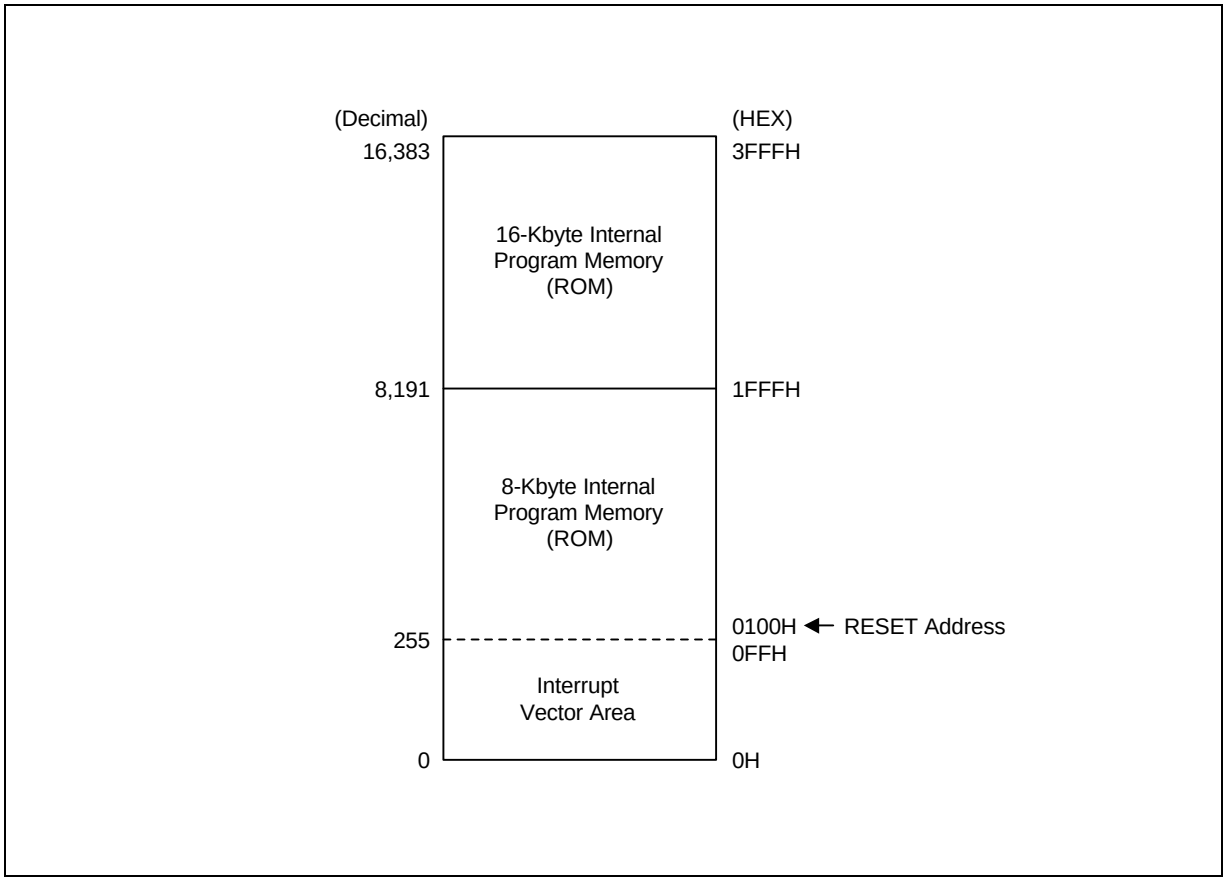


Figure 5-3. ROM Vector Address Area

Table 5-1. KS88C4708/C4716 Interrupt Vectors

Vector Address		Interrupt Source	Request		Reset/Clear	
Decimal Value	Hex Value		Interrupt Level	Priority in Level	H/W	S/W
250	FAH	Timer 0 overflow	IRQ0	0	√	
252	FCH	Timer 0 match/capture		1		√
244	F4H	Timer 1 overflow	IRQ1	0	√	
246	F6H	Timer 1 match/capture		1	√	
196	F0H	UART receive interrupt	IRQ2	0		√
198	F2H	UART transmit interrupt		1		√
224	EEH	P2.0 external interrupt	IRQ3	0	√	
226	ECH	P2.1 external interrupt	IRQ4	0	√	
286	EAH	P2.2 external interrupt	IRQ5	0	√	
230	E8H	P2.3 external interrupt	IRQ6	0	√	
232	D8H	P2.4 external interrupt	IRQ7	0	√	
234	DAH	P2.5 external interrupt		1	√	
236	DCH	P2.6 external interrupt		2	√	
238	DEH	P2.7 external interrupt		3	√	

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 usually has priority over one with a higher vector address. The priorities within a given level are set in hardware.

ENABLE/DISABLE INTERRUPT INSTRUCTIONS (EI, DI)

The Enable Interrupts (EI) instruction globally enables the interrupt structure. All interrupts are serviced as they occur, and according to established priorities. The system initialization routine that is executed following a reset must always contain an EI instruction.

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 control registers control interrupt processing:

— An interrupt level is enabled or disabled (masked) by bit settings in the interrupt mask register (IMR).

— Relative priorities of interrupt levels are controlled by the interrupt priority register (IPR).

— The interrupt request register (IRQ) contains interrupt pending flags for each level.

— The system mode register (SYM) dynamically enables or disables global interrupt processing. SYM settings also enable fast interrupts and control the external interface, if implemented.

Table 5-2. Interrupt Control Register Overview

Control Register	ID	R/W	Function Description
System mode register	SYM	R/W	Dynamic global interrupt processing enable and disable, fast interrupt processing, and external interface control.
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 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.

EXTERNAL INTERRUPT CONTROL REGISTERS

Port 2 Interrupt Control

You use the P2CONH, and P2CONL control register to enable and disable external interrupts INT0–INT7, respectively. P2CONH, and P2CONL also lets you select rising or falling signal edges to trigger interrupt requests. Pending conditions for the port 2 external interrupts are automatically cleared by hardware.

Because port 2 does not have a separate interrupt enable register, you must manipulate the bit-pair settings for each pin by software in order to enable or disable the interrupt for that pin.

INTERRUPT PROCESSING CONTROL POINTS

Interrupt processing can therefore be controlled in two ways: either 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 and disable settings (IMR register)

— Interrupt level priority settings (IPR register)

— Interrupt enable and disable settings in the corresponding peripheral control registers

NOTE

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

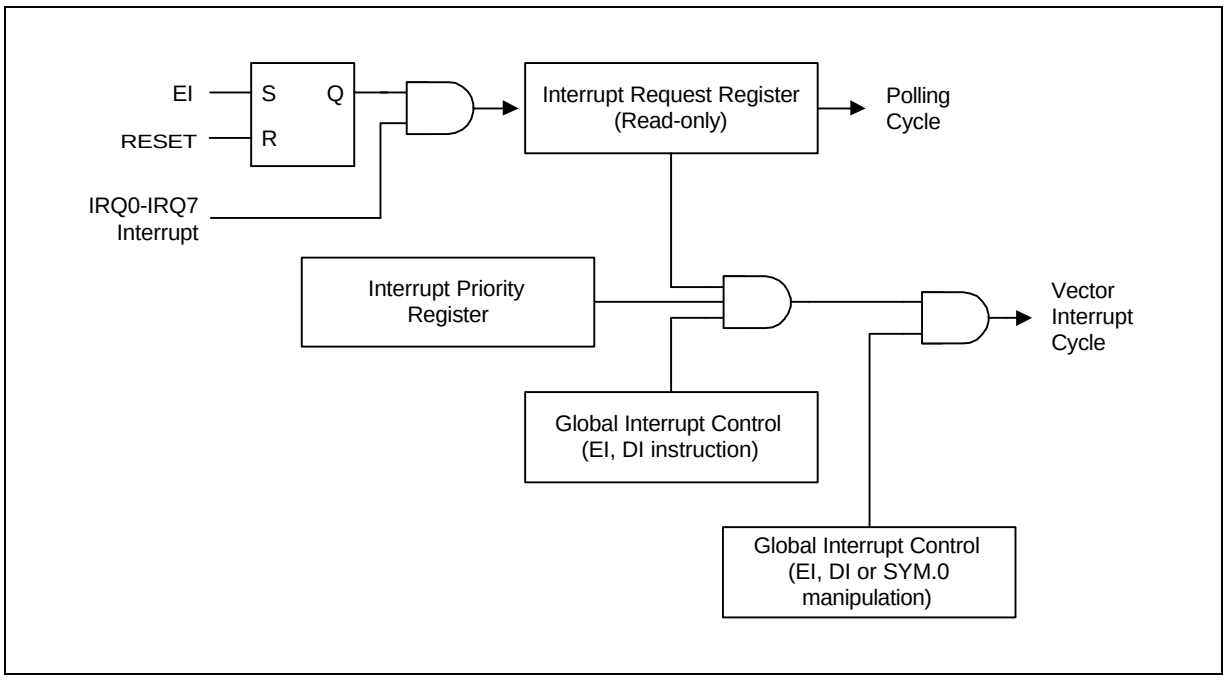


Figure 5-4. Interrupt Function Diagram

SYSTEM MODE REGISTER (SYM)

The system mode register, SYM (DEH, set 1), is used to enable and disable interrupt processing and to control fast interrupt processing.

SYM.0 is the enable and disable bit for global interrupt processing. SYM.1–SYM.4 control fast interrupt processing: SYM.1 is the enable bit; SYM.2–SYM.4 are the fast interrupt level selection bits. A reset clears SYM.0, and SYM.1 while the other bit values 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.

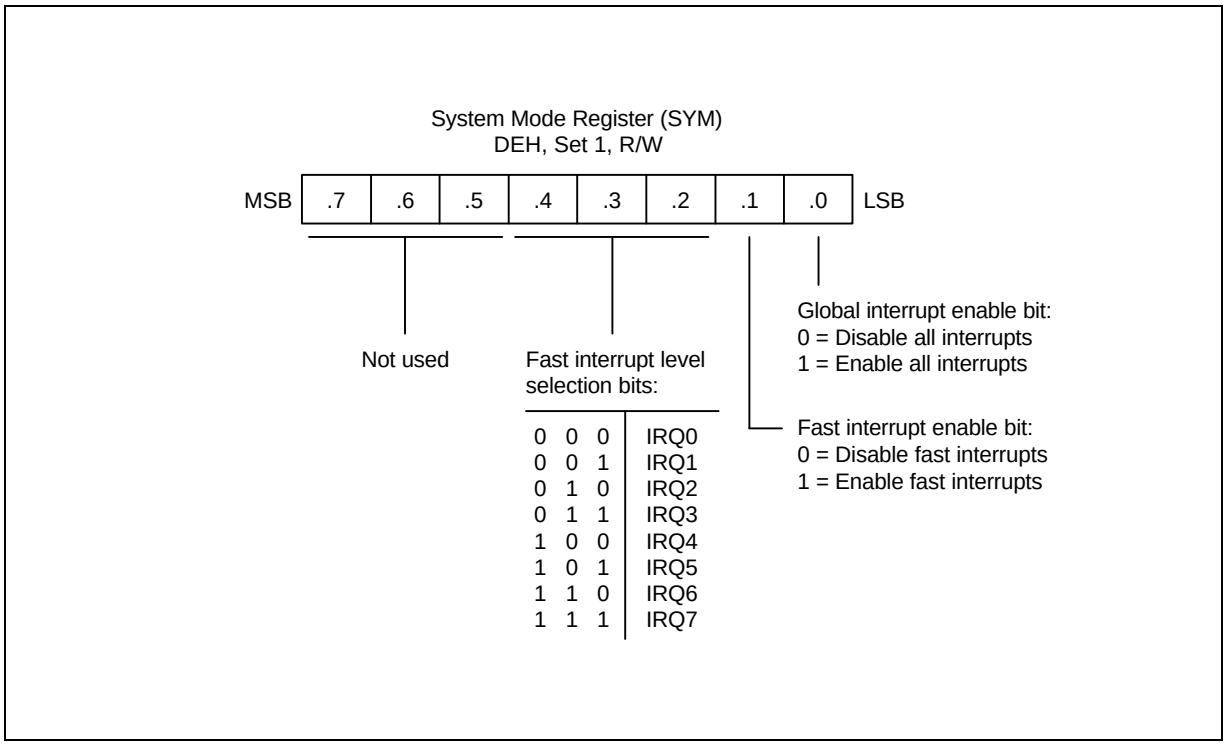


Figure 5-5. System Mode Register (SYM)

INTERRUPT MASK REGISTER (IMR)

The interrupt mask register (IMR) is used to enable or disable interrupt processing for each of the eight interrupt levels used in the KS88C4708/C4716 interrupt structure, IRQ0–IRQ7. After a reset, IMR register values are undetermined and must therefore be set 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.

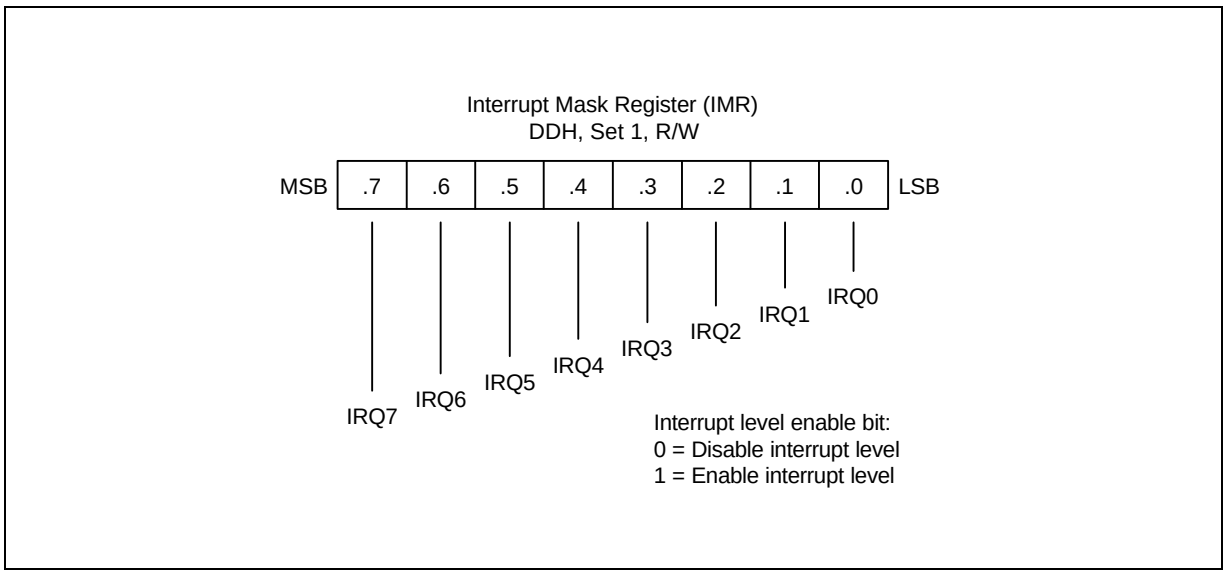


Figure 5-6. Interrupt Mask Register (IMR)

INTERRUPT PRIORITY REGISTER (IPR)

The interrupt priority register, IPR, is used to set the relative priorities of the eight interrupt levels used in the KS88C4708/C4716 interrupt structure. The IPR register is mapped to register location FFH in set 1, bank 0. After a reset, the IPR register values are undetermined and must therefore be set 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.)

In order to define the relative priorities of interrupt levels, they are organized into groups and subgroups by the interrupt logic. Three interrupt groups are defined for the IPR logic (see Figure 5-7). These groups and subgroups are used only for IPR register priority definitions:

Group A IRQ0, IRQ1

Group B IRQ2, IRQ3, IRQ4

Group C IRQ5, IRQ6, IRQ7

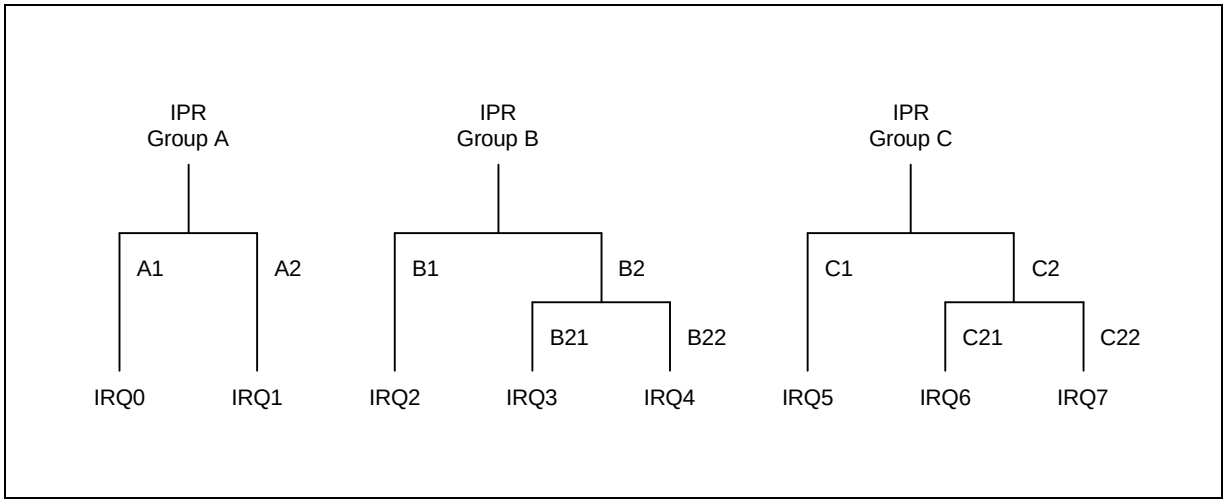


Figure 5-7. Interrupt Request Priority Groups

Bits 7, 4, and 1 of the IPR register control the relative priority of interrupt groups A, B, and C. For example, the setting "001B" would select the group relationship B > C > A, and "101B" would select C > B > A. The functions of the other IPR bit settings are as follows:

— IPR.0 controls the relative priority setting of IRQ0 and IRQ1 interrupts.

— IPR.2 controls interrupt group B.

— 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.5 controls the relative priorities of group C interrupts.

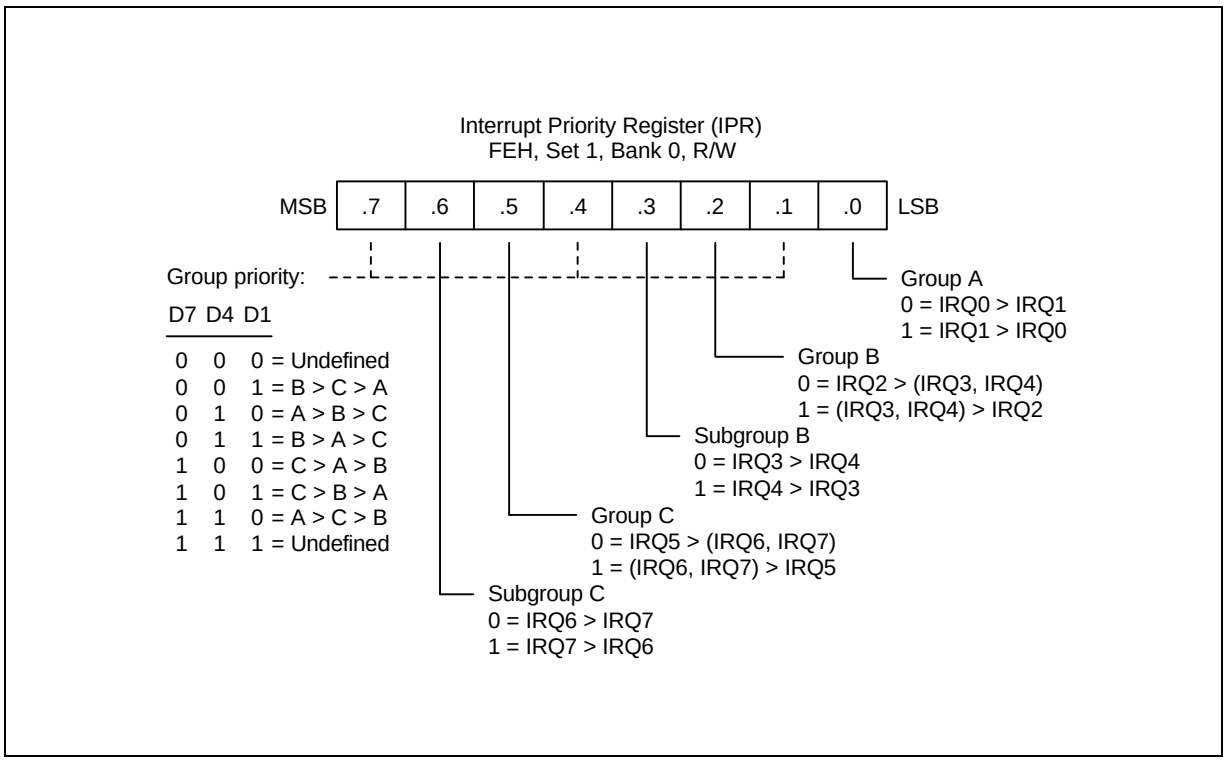


Figure 5-8. Interrupt Priority Register (IPR)

INTERRUPT REQUEST REGISTER (IRQ)

Bit values in the interrupt request register, IRQ, are polled to determine interrupt request status for the eight interrupt levels in the KS88C4708/C4716 interrupt structure, IRQ0–IRQ7. 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 issued for that level.

The IRQ register is mapped to register location DCH in set 1. 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, the IRQ register is cleared to "00H".

IRQ register values can be polled even if a DI instruction has been executed. If an interrupt occurs while the interrupt structure is disabled, it will not be serviced. But the interrupt request can still be detected by polling IRQ values. This can be useful in order to determine which events occurred while the interrupt structure was disabled.

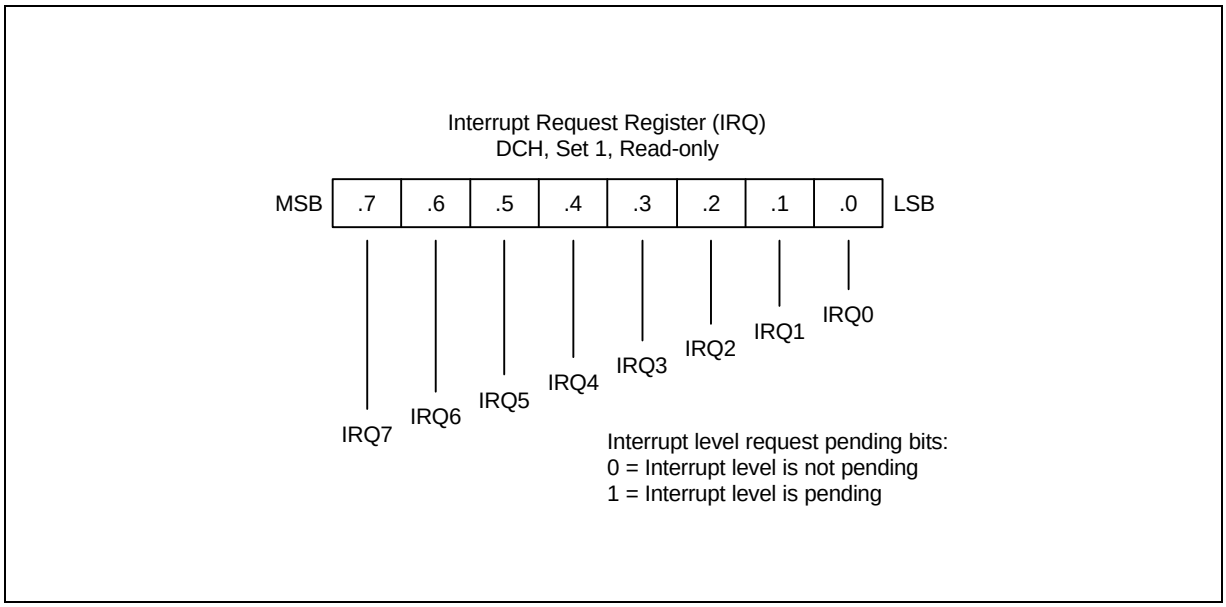


Figure 5-9. Interrupt Request Register (IRQ)

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.

Each interrupt level has a corresponding interrupt request bit in the IRQ register that the CPU polls for interrupt requests.

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 tell the CPU that an interrupt is waiting to be serviced. The CPU acknowledges the interrupt source (IACK), 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 software.

In the KS88C4708/C4716 interrupt structure, the timer 0 overflow interrupt, timer 1 interrupt, and the P2.0–P2.7 external interrupts belong to this category of interrupts whose pending conditions are cleared automatically by hardware. All other interrupt pending conditions must be cleared by software.

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 pending bit location in the corresponding mode or control register.

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 flag is cleared to "0" (either 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 enabled (EI, SYM.0 = "1")
— Interrupt level must be enabled (IMR register)
— Interrupt level must have the highest priority if more than one level is currently requesting service
— 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 and status flags to stack.
3. Branch to the interrupt vector to fetch the service routine's address.
4. Pass control to the interrupt service routine.

When the interrupt service routine is completed, an Interrupt Return instruction (IRET) occurs. 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 contains the addresses of the interrupt service routine that corresponds to each level in the interrupt structure. Vectored interrupt processing follows this sequence:

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

NOTE

A 16-bit vector address always begins at an even-numbered ROM location from 00H–FFH.

NESTING OF VECTORED INTERRUPTS

You can 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 to enable the higher priority interrupt only.
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, restore the IMR to its original value by returning the previous mask from the stack (POP IMR).
5. Execute an IRET.

Depending on the application, you may be able to simplify this 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 designated interrupts be completed in approximately six clock cycles instead of the usual 16 clock cycles. Bit 1 of the system mode register, SYM.1, enables fast interrupt processing while SYM.2–SYM.4 are used to select a specific level for fast processing.

Two other system registers support fast interrupts:

— The instruction pointer (IP) holds 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).

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.
2. Load the level number into the fast interrupt select field.
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 dedicated FLAGS' 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 into the FLAGS register.
7. The fast interrupt status bit in FLAGS is cleared automatically.

Interrupt Pending Bit Types

As described in a previous subsection, 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. Fast interrupt processing can be used for interrupts with either type of pending clear function (cleared by hardware or cleared by software).

Programming Guidelines

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

Also, if you use fast interrupts, remember to load the IP with a new start address when the fast interrupt service routine ends. (For an example, please refer to the programming tip on the next page.)

☞ PROGRAMMING TIP — Programming Level IRQ0 as a Fast Interrupt

This example shows you how to program fast interrupt processing for a select interrupt level — in this case, for the timer 0 capture interrupt (level IRQ0, vector FCH):

```
                •
                •
                •
                LD      BTCON,#10100011B    ;  Watchdog disable
                LD      T0CON,#52H          ;  Disable T0OVF interrupt
                                            ;  Enable T0 interrupt
                                            ;  Capture mode; trigger on rising signal edges
                                            ;  Select fOSC/256 as the T0 clock source
                LD      P1CONL,#00H         ;  Set P1.0 to capture input mode
                LDW     IPH,#T0_INT         ;  IPH ← high byte of interrupt service routine
                                            ;  IPL ← low byte of interrupt service routine
                LD      SYM,#02H            ;  Enable fast interrupt processing
                                            ;  Select IRQ0 for fast interrupt service
                EI                          ;  Enable interrupts
                •
                •
                •

FAST_RET:       IRET                        ;  IP ← Address of T0_INT (again)
T0_INT:
                •
                •
                •
                (Fast service routine executes)
                •
                •
                •
                LD      T0CON,#52H          ;  Clear T0INT interrupt pending bit
                JP      T,FAST_RET
```

NOTES

7 CLOCK CIRCUIT

OVERVIEW

The clock frequency generated for the KS88C4708/C4716 by an external crystal can range from 1 MHz to 12 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 external clock)
— Oscillator stop and wake-up functions
— Programmable frequency divider for the CPU clock (f_{OSC} divided by 1, 2, 8, or 16)
— Clock circuit control register, CLKCON

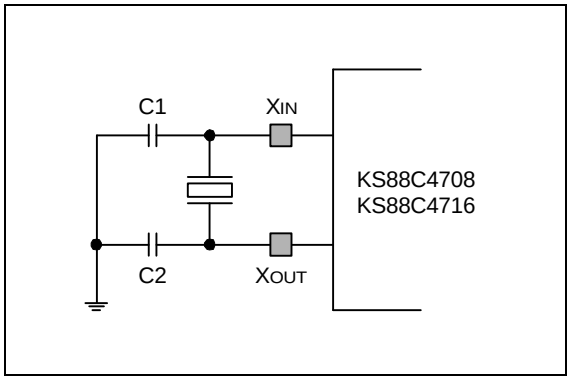


Figure 7-1. Main Oscillator Circuit (External Crystal or Ceramic Resonator)

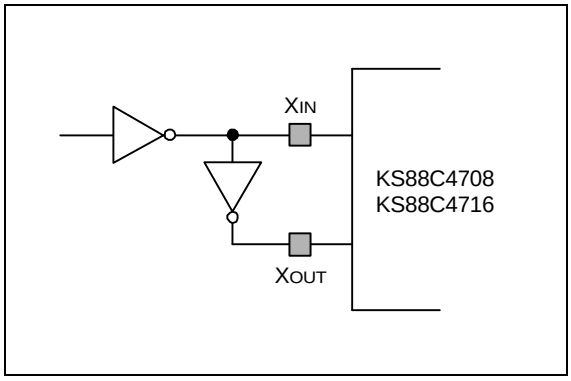


Figure 7-2. Main Oscillator Circuit (External Clock Source)

CLOCK STATUS DURING POWER-DOWN MODES

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

— In Stop mode, the main oscillator is halted. Stop mode is released, and the oscillator started, by a reset operation or by an external interrupt with RC-delay noise filter (for KS88C4708/C4716, INT2–INT7).
— In Idle mode, the internal clock signal is gated off to the CPU, but not to the interrupts, timers, and serial I/O functions. Idle mode is released by a reset or by an interrupt (external or internally-generated).

SYSTEM CLOCK CONTROL REGISTER (CLKCON)

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

— Oscillator IRQ wake-up function enable/disable (CLKCON.7)

— Main oscillator stop control (CLKCON.6 and CLKCON.5)

— Oscillator frequency divide-by value: non-divided, 2, 8, or 16 (CLKCON.4 and CLKCON.3)

— System clock signal selection (CLKCON.0–CLKCON.2)

The CLKCON register controls whether or not an external interrupt can be used to trigger a Stop 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} or $f_{OSC}/2$.

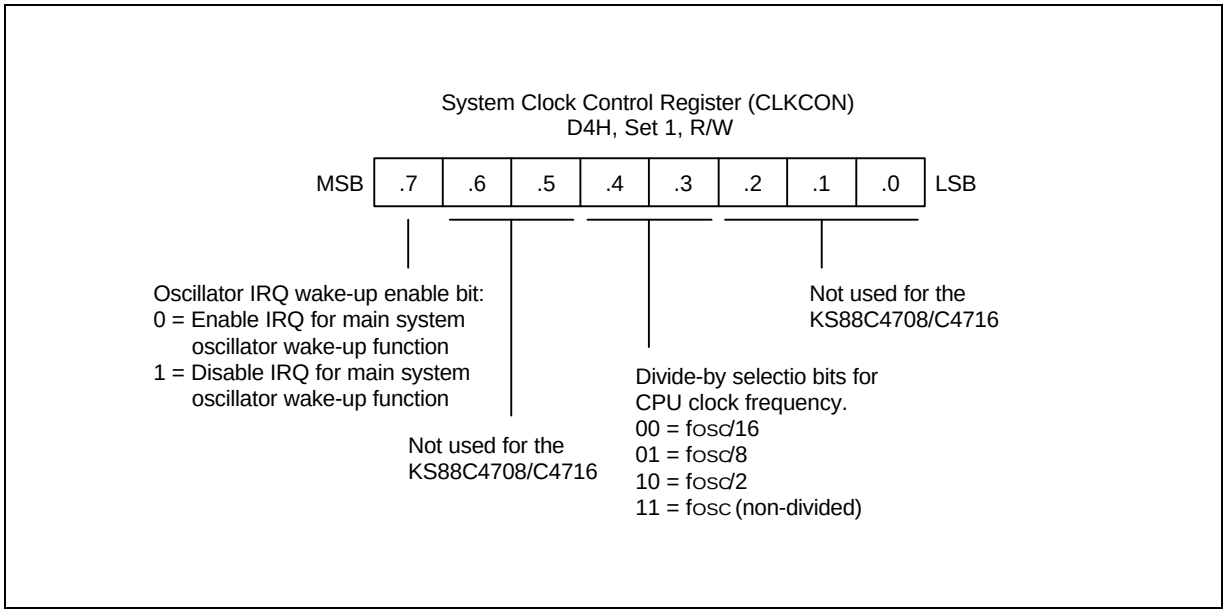


Figure 7-3. System Clock Control Register (CLKCON)

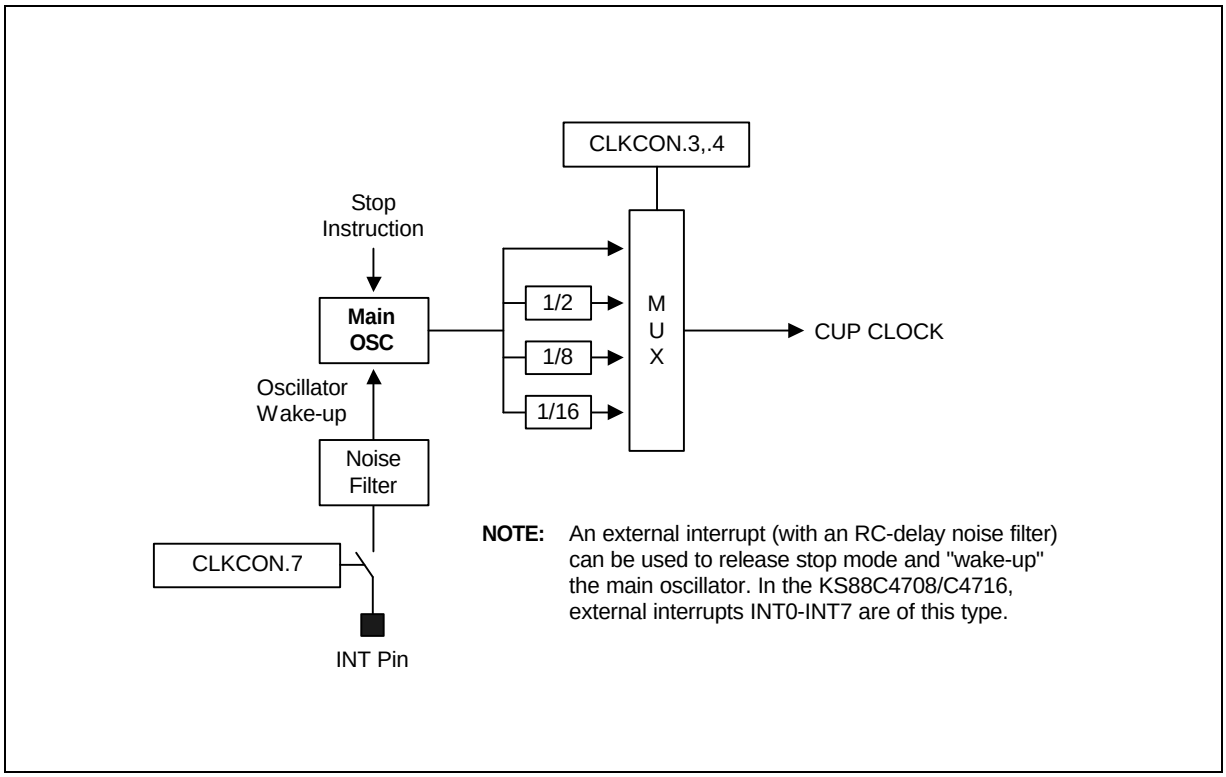


Figure 7-4. System Clock Circuit Diagram

NOTES

8 RESET AND POWER-DOWN

RESET

During a power-on reset, the voltage at V_{DD} is 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 brings the KS88C4708/C4716 into a known operating status.

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

When a reset occurs during the normal operation (with both V_{DD} and RESET at High level), the signal at the RESET pin is forced Low and the reset operation starts. All system and peripheral control registers are then set to their default hardware reset values (see Table 8-1).

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

— All interrupts are disabled.

— The watchdog function (basic timer) is enabled.

— Ports 0–4 are set to schmitt trigger input mode (except P4.4-P4.5) and all pull-up resistors are disabled

— Peripheral control and data registers are disabled and reset to their initial control values.

— The program counter is loaded with the ROM's reset address, 0100H.

— When the programmed oscillation stabilization time interval has elapsed, the address stored in ROM location 0100H (and 0101H) is fetched and executed.

NOTE

To program the duration of the oscillation stabilization interval, you must make the appropriate settings to the basic timer control register, BTCON, before entering Stop mode. Also, you 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.

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 5 μA. All system functions are halted 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 signal or by an external interrupt.

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 then reset to their default 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 oscillation stabilization interval has elapsed, the CPU executes the system initialization routine by fetching the 16-bit address stored in ROM locations 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. (Clock-related external interrupts cannot be used.) External interrupts INT0–INT7 in the KS88C4708/C4716 interrupt structure meet this criteria.

Note that when Stop mode is released by an external interrupt, the current values in system and peripheral control registers are not changed. When you use an interrupt to release Stop mode, the CLKCON.3 and CLKCON.4 register values remain unchanged, and the currently selected clock value is used. 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.

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 select peripherals remain active. During Idle mode, the internal clock signal is gated off to the CPU, but not to interrupt logic and timer/counters. 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 (f_{OSC}/16) because CLKCON.3 and CLKCON.4 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.3 and CLKCON.4 register values remain unchanged, and the currently selected clock value is used. The interrupt is then serviced. Following the IRET from the service routine, the instruction immediately following the one that initiated Idle mode is executed.

NOTE

Only external interrupts that are not clock-related can be used to release Stop mode. To release Idle mode, however, any type of interrupt (that is, internal or external) can be used.

DEFAULT HARDWARE RESET VALUES

Tables 8-1 through 8-2 list the default reset values for CPU and system registers, peripheral control registers, and peripheral data registers following a reset operation in normal operating mode. The following notation is used in these tables to represent specific 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 following a reset.

— A dash ("–") means that the bit is either not used or not mapped.

Table 8-1. Set 1 Register Values after Reset

Register Name	Mnemonic	Address		Bit Values after RESET							
		Dec	Hex	7	6	5	4	3	2	1	0
Timer 0 counter	T0CNT	208	D0H	0	0	0	0	0	0	0	0
Timer 0 data register	T0DATA	209	D1H	1	1	1	1	1	1	1	1
Timer 0 control register	T0CON	210	D2H	0	0	0	0	0	0	0	0
Basic timer control register	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

NOTES:

1. The timer 0 counter, T0CNT, and the interrupt request register, IRQ, are read-only. All other registers in set 1 are read/write addressable.
2. Shaded table cells indicate interrupt pending bits. To clear an interrupt pending condition, you must write a "0" to the appropriate bit location.
3. Although a timer 0 interrupt (match/capture) pending condition must be cleared by software (writing T0CON.0 to "0"), the timer 0 overflow interrupt and the basic timer interrupt pending conditions are cleared automatically by hardware.

Table 8-2. Set 1 Register Values after Reset

Register Name	Mnemonic	Address		Bit Values after RESET							
		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
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
Port 0 control register	P0CON	229	E5H	0	0	0	0	0	0	0	0
Port 1 control register high	P1CONH	230	E6H	–	–	–	–	0	0	0	0
Port 1 control register low	P1CONL	231	E7H	0	0	0	0	0	0	0	0
Port 2 control register high	P2CONH	232	E8H	0	0	0	0	0	0	0	0
Port 2 control register low	P2CONL	233	E9H	0	0	0	0	0	0	0	0
Port 3 control register high	P3CONH	234	EAH	0	0	0	0	0	0	0	0
Port 3 control register low	P3CONL	235	EBH	0	0	0	0	0	0	0	0
Port 4 control register	P4CON	236	ECH	0	0	0	0	0	0	0	0
Port 2 pull-up enable register	P2PUR	237	EDH	0	0	0	0	0	0	0	0
UART control register	UARTCON	238	EEH	0	0	0	0	0	0	0	0
UART interrupt pending register	UARTPND	239	EFH	–	–	–	–	–	–	0	0
UART data register	UDATA	240	F0H	0	0	0	0	0	0	0	0
UART baud rate data register	BRDATA	241	F1H	0	0	0	0	0	0	0	0
Timer 1 data register high	T1DATAH	242	F2H	1	1	1	1	1	1	1	1
Timer 1 data register low	T1DATAL	243	F3H	1	1	1	1	1	1	1	1
Timer 1 counter high	T1CNTH	244	F4H	0	0	0	0	0	0	0	0
Timer 1 counter low	T1CNTL	245	F5H	0	0	0	0	0	0	0	0
Timer 1 control register	T1CON	246	F6H	0	0	0	0	0	0	0	0
A/D control register	ADCON	247	F7H	–	0	0	0	0	0	0	0
A/D converter data register (high)	ADDATAH	248	F8H	x	x	x	x	x	x	x	x
A/D converter data register (low)	ADDATAL	249	F9H	0	0	0	0	0	0	x	x
8-bit prescaler for buzzer output	BUZPS	250	FAH	0	0	0	0	0	0	0	0
Location FBH is not mapped.											
Reserved (factory test use only)	FTEST	252	FCH								
Basic timer counter register	BTCNT	253	FDH	0	0	0	0	0	0	0	0
Reserved	EMT	254	FEH								
Interrupt priority register	IPR	255	FFH	x	x	x	x	x	x	x	x

☞ PROGRAMMING TIP — Sample KS88C4708/C4716 Initialzation Routine

The following sampl program suggests how to program the initial program settings for KS88C4608/C4716 address space, interrupt and peripheral function. Program comment guide you through the nessary steps.

```
;-----<< Interrupt Vector Address >>

        ORG     0000H

        VECTOR  0D8H,P24_INT        ; IRQ7
        VECTOR  0DAH,P25_INT        ; IRQ7
        VECTOR  0DCH,P26_INT        ; IRQ7
        VECTOR  0DEH,P27_INT        ; IRQ7

        VECTOR  0E8H,P23_INT        ; IRQ6

        VECTOR  0EAH,P22_INT        ; IRQ5

        VECTOR  0ECH,P21_INT        ; IRQ4

        VECTOR  0EEH,P20_INT        ; IRQ3

        VECTOR  0F0H,UARTRX_INT     ; IRQ2
        VECTOR  0F2H,UARTTX_INT     ; IRQ2

        VECTOR  0F4H,T1OVF_INT      ; IRQ1
        VECTOR  0F6H,T1MC_INT       ; IRQ1

        VECTOR  0FAH,T0OVF_INT      ; IRQ0
        VECTOR  0FCH,T0MC_INT       ; IRQ0
```

☞ PROGRAMMING TIP — Sample KS88C4708/C4716 Initialzation Routine (Continued)

```
;-----<< Initialize System and Peripherals >>

                ORG     0100H

RESET:          DI
                LD      SYM,#00H            ;  Global/Fast interrupt disable
                LD      EMT,#00H            ;  Internal stack/No wait
                LD      IMR,#10000001B      ;  IRQ0/IRQ7 int enable
                LD      SPL,#0FFH           ;  Set stack pointer
                LD      BTCON,#10100011B    ;  Watchdog disable
                LD      CLKCON,#00011000B   ;  Select non-divided oscillator frequency as CPU clock

INIT_PORT:      LD      P0CON,#31H          ;  P0.4-P0.7: Open-drain output mode
                                            ;  P0.0-P0.3: Push-pull output mode
                LD      P1CONH,#0FH         ;  P1.4-P1.5: UART Rx,Tx mode
                LD      P1CONL,#00000101B   ;  P1.2-P1.3: Input mode
                                            ;  P1.0-P1.1: Input mode with pull-up resistor
                LD      P2CONH,#01011010B   ;  P2.6-P2.7: falling edge interrupt mode
                                            ;  P2.4-P2.5: Push-pull output mode
                LD      P2CONL,#0AAH        ;  P2.0-P2.3: Push-pull output mode
                LD      P2PUR,#11000000B    ;  P2.6-P2.7: pull-up enable
                LD      P3CONH,#0AAH        ;  P3.4-P3.7: Push-pull output mode
                LD      P3CONL,#0AAH        ;  P3.0-P3.3: Push-pull output mode
                LD      P4CON,#55H          ;  P4.0-P4.3: Input mode with pull-up resistor
                                            ;  P4.4-P4.5: Output only port

;-----<< Timer 0 Settings >>

                LD      T0CON,#01000010B    ;  Interval mode
                LD      T0DATA,#61          ;

;-----<< Initialize Data Registers >>

                LD      PP,#00H             ;
                SRP     #0C0H               ;  Set register pointer

                LD      R0,#00H             ;  RAM clear area setting
RAMCLR_P0:      CLR     @R0                 ;  << page 0 clear >>
                DJNZ    R0,RAMCLR_P0        ;

;-----<< Initialize Other Registers >>

                •
                •
                •
                EI                          ;  Enable interrupt
```

☞ PROGRAMMING TIP — Sample KS88C4708/C4716 Initialzation Routine (Continued)

```
;-----<< Main Loop >>

MAIN:       NOP                          ;  Start main loop
            LD        BTCON,#11110010B   ;  Watch-dog enable
                                         ;  Basic timer clock : Fosc/4096
                                         ;  Clear basic timer counter before overflow
            •
            •
            •
            CALL      KEY_SCAN           ;  Sub-block module
            •
            •
            •
            CALL      LED_DISPLAY        ;  Sub-block module
            •
            •
            •
            CALL      JOB                ;  Sub-block module
            •
            •
            •
            JR        t,MAIN             ;  To loop main routine

;-----< Subroutines >

KEY_SCAN:   NOP                          ;
            •
            •
            •
            RET                          ;

LED_DISPLAY: NOP                         ;
            •
            •
            •
            RET                          ;

JOB:        NOP                          ;
            •
            •
            •
            RET                          ;
```

☞ PROGRAMMING TIP — Sample KS88C4632 Initialzation Routine (Continued)

```
;-----<< Interrupt Service Routines >>

T0MC_INT:   PUSH    RP0                 ;  IRQ0 (S/W pending)
            SRP0    #90H                ;
            •
            •
            •
            LD      T0CON,#01000010B    ;  Clear timer 0 pending bit
            POP     RP0                 ;
            IRET                        ;

;-----------------------------------------------;

T0OVF_INT:                              ;  IRQ0
            IRET                        ;  Interrupt return

;-----------------------------------------------;

T1OVF_INT:                              ;  IRQ1
            IRET                        ;  Interrupt return

;-----------------------------------------------;

T1MC_INT:                               ;  IRQ1
            IRET                        ;  Interrupt return

;-----------------------------------------------;

UARTRX_INT:                             ;  IRQ2 (S/W pending)
            LD      UARTPND,#00000001B  ;  Rx intrrrupt pending bit clear
            IRET                        ;  Interrupt return

;-----------------------------------------------;

UARTTX_INT:                             ;  IRQ2 (S/W pending)
            LD      UARTPND,#00000010B  ;  Tx intrrrupt pending bit clear
            IRET                        ;  Interrupt return
```

☞ PROGRAMMING TIP — Sample KS88C4708/C4716 Initialzation Routine (Continued)

```
;-----------------------------------------------;

P20_INT:                                ;  IRQ3
                IRET                    ;  Interrupt return

;-----------------------------------------------;

P21_INT:                                ;  IRQ4
                IRET                    ;  Interrupt return

;-----------------------------------------------;

P22_INT:                                ;  IRQ5
                IRET                    ;  Interrupt return

;-----------------------------------------------;

P23_INT:                                ;  IRQ6
                IRET                    ;  Interrupt return

;-----------------------------------------------;

P24_INT:                                ;  IRQ7
                IRET                    ;  Interrupt return
P25_INT:                                ;  IRQ7
                IRET                    ;  Interrupt return
P26_INT:                                ;  IRQ7
                IRET                    ;  Interrupt return
P27_INT:                                ;  IRQ7
                IRET

;-----------------------------------------------;

                END
```

NOTES

9 I/O PORTS

OVERVIEW

The KS88C4708/C4716 has 34 I/O pins (port 0–4), and 2 output only pins (port 4). All I/O pins have share functions that can be flexibly configured to meet application design requirements. The CPU accesses ports directly by writing or reading port register addresses.

Ports 0–4 have byte control registers. Bit settings in these registers configure port pins individually.

In addition to their alternative share functions, pins for ports 0–4 can be individually set to schmitt trigger input mode or push-pull output mode, except (P4.4-P4.5).

Pull-up resistors can be configured to port 0, port 1, port 3, and port 4 pins using regular port control register settings. Separate pull-up resistor enable registers are provided for port 2.

Port 2 has separate interrupt enable and interrupt edge control registers for external interrupts INT0–INT7. Interrupt pending conditions are cleared automatically by hardware.

PORT DATA REGISTERS

Data registers for ports 0–4 have the structure shown in Figure 9-1. Table 9-2 gives you an overview of the port data register names, locations, and addressing characteristics:

Table 9-1. Port Data Register Summary

Register Name	Mnemonic	Decimal	Hex	Location	R/W
Port 0 data register	P0	224	E0H	Set 1	R/W
Port 1 data register	P1	225	E1H	Set 1	R/W
Port 2 data register	P2	226	E2H	Set 1	R/W
Port 3 data register	P3	227	E3H	Set 1	R/W
Port 4 data register	P4	228	E4H	Set 1	R/W

NOTE: A reset operation clears the P0–P4 data register to "00H".

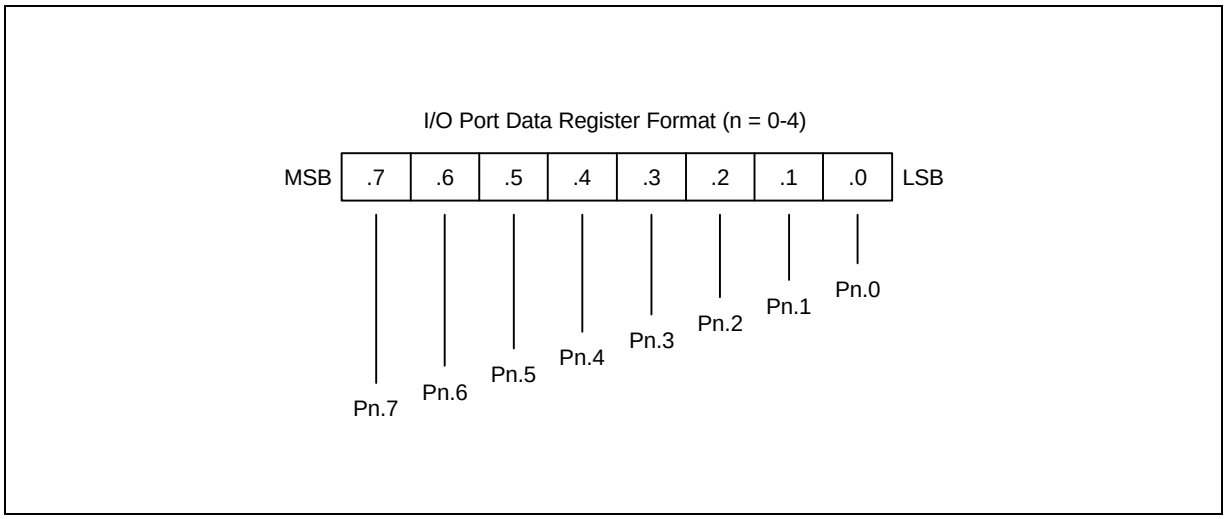


Figure 9-1. Port Data Register Format

PORT 0

Port 0 is a bit-programmable general I/O port. Using P0CON control register settings, you can configure individual nibbles to input mode, pull-up enable, push-pull output, open-drain output, open-drain, pull-up.

In normal operating mode, a reset clears the port 0 control registers, P0CON, to "00H", configuring P0.0–P0.7 as schmitt trigger inputs.

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

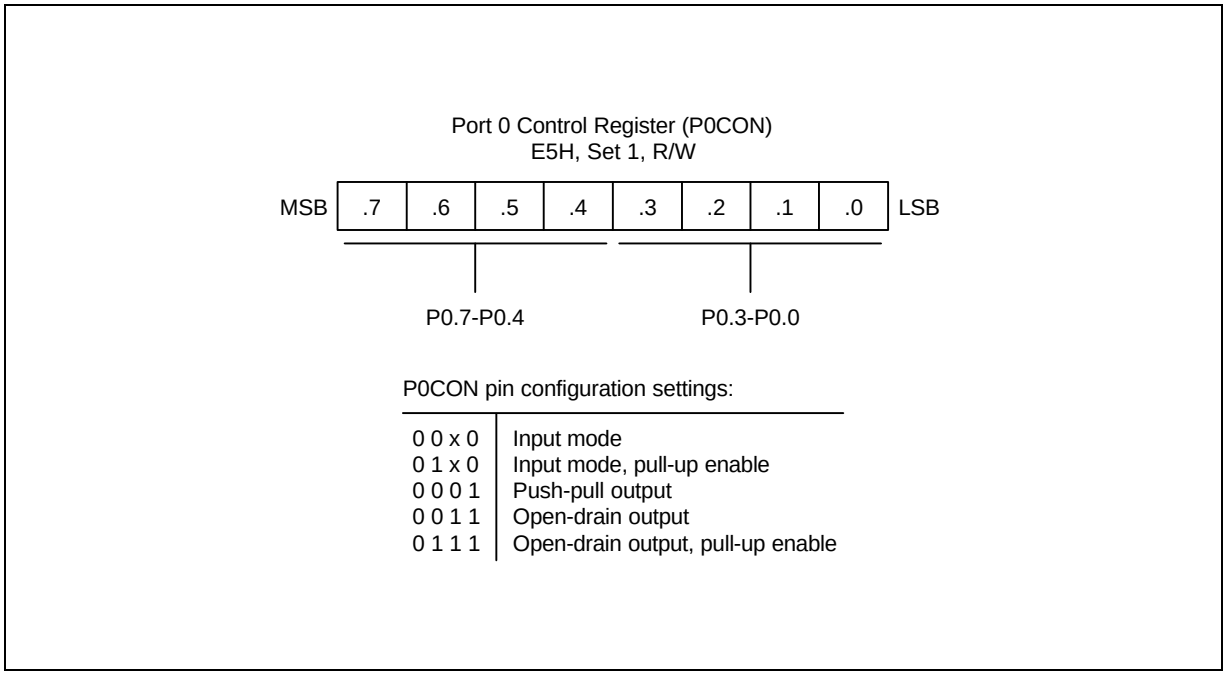


Figure 9-2. Port 0 Control Register (P0CON)

PORT 1

Port 1 is a bit-programmable general I/O port. Using P1CONH and P1CONL control register settings, you can configure individual pins to schmitt trigger input mode, alternative output mode, or push-pull output mode.

In normal operating mode, a reset clears the port 1 control registers, P1CONH and P1CONL, to "00H", configuring P1.0–P1.5 as schmitt trigger inputs.

Port 1 is accessed directly by writing or reading the port 1 data register, P1 (E1H, set 1).

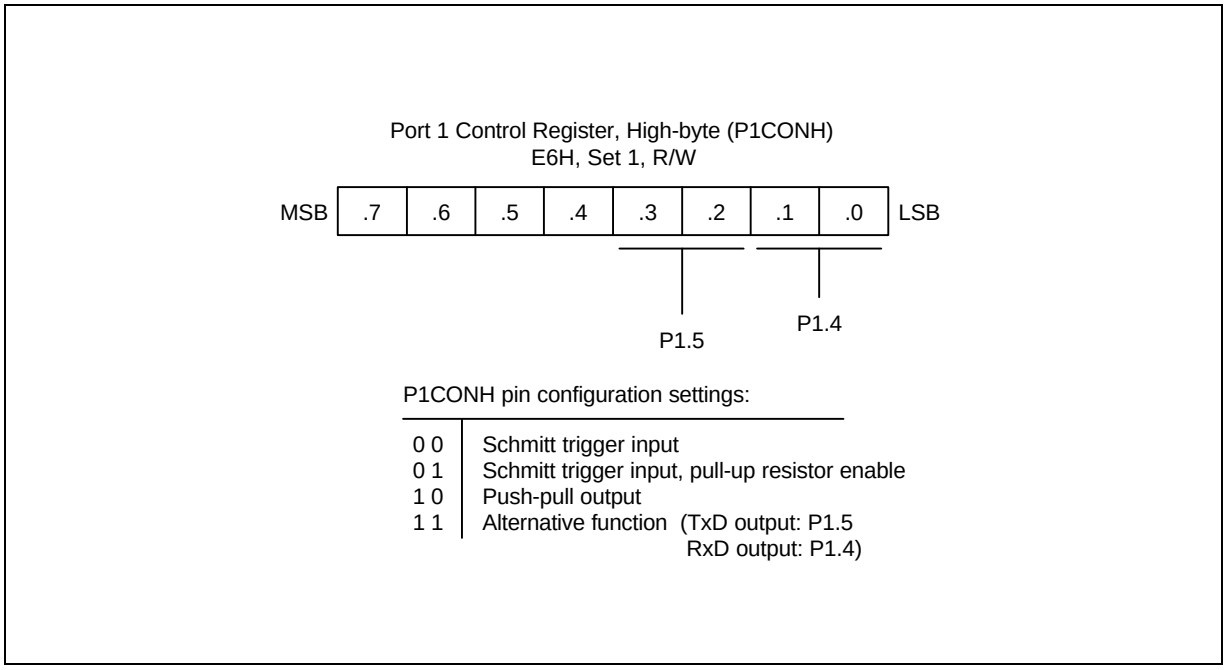
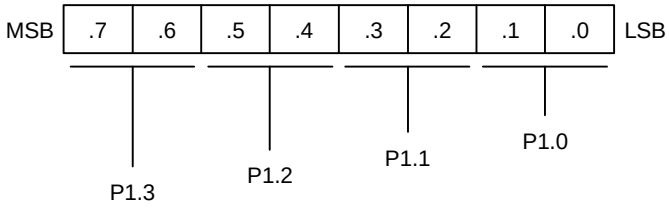


Figure 9-3. Port 1 High-Byte Control Register (P1CONH)

Port 1 Control Register, Low-byte (P1CONL)
E7H, Set 1, R/W



P1CONL pin configuration settings for 1.3

0 0	Schmitt trigger input
0 1	Schmitt trigger input, pull-up resistor enable
1 0	Push-pull output
1 1	Alternative function (BUZ output)

P1CONL pin configuration settings for 1.2

0 0	Schmitt trigger input (or T1 capture input)
0 1	Schmitt trigger input, pull-up resistor enable
1 0	Push-pull output
1 1	Alternative function (T1 output: match or PWM)

P1CONL pin configuration settings for 1.1

0 0	Schmitt trigger
0 1	Schmitt trigger input, pull-up resistor enable
1 0	Push-pull output
1 1	Alternative function (T1CK input)

P1CONL pin configuration settings for 1.0

0 0	Schmitt trigger (or T0 capture input)
0 1	Schmitt trigger input, pull-up resistor enable
1 0	Push-pull output
1 1	Alternative function (T0 output: match or PWM)

Figure 9-4. Port 1 Low-Byte Control Register (P1CONL)

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:

— External interrupt inputs for INT0-INT7

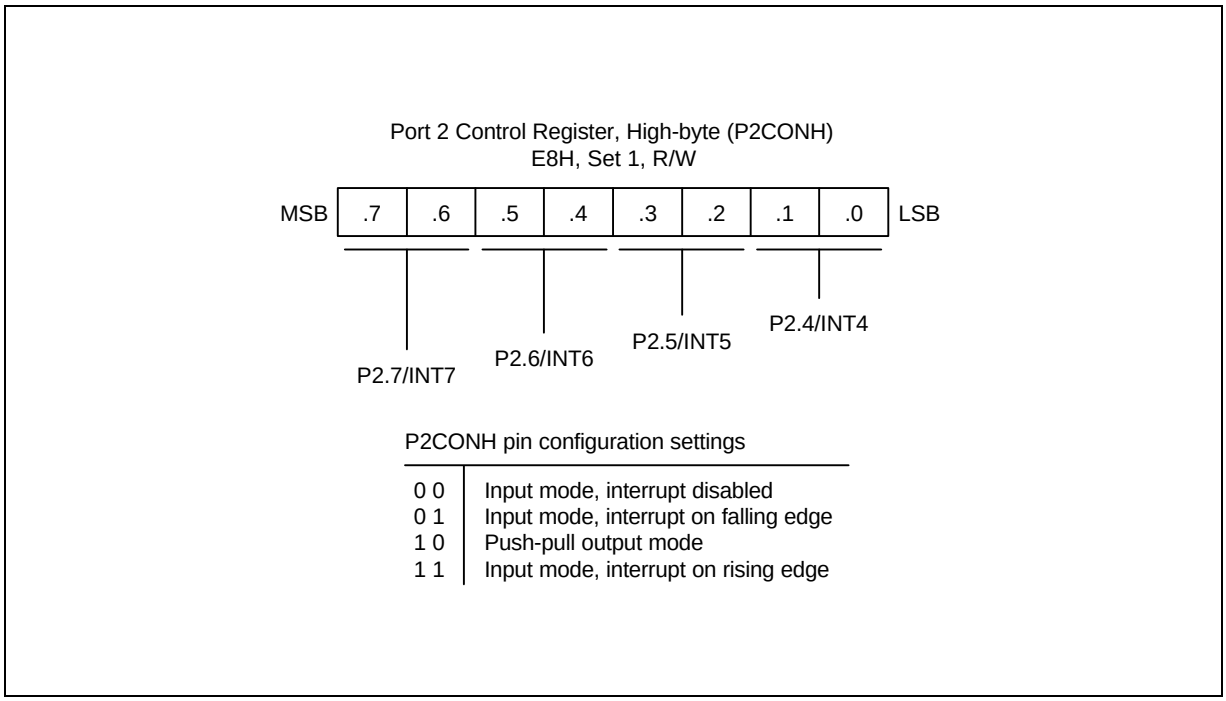


Figure 9-5. Port 2 High-Byte Control Register (P2CONH)

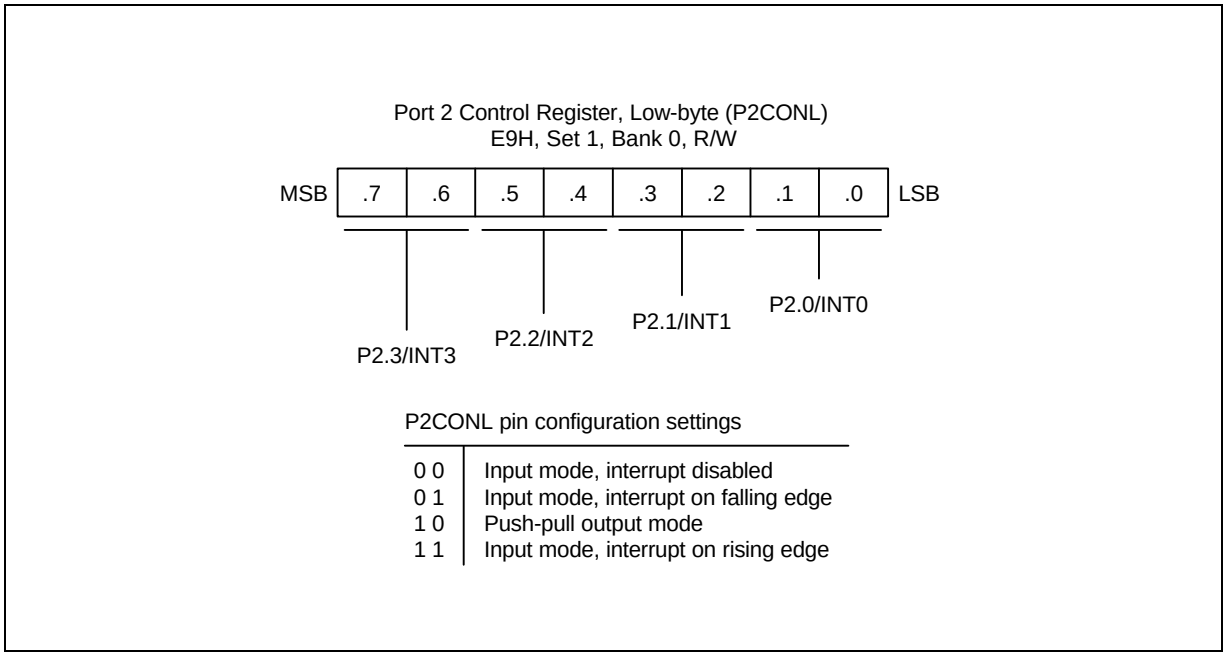


Figure 9-6. Port 2 Low-Byte Control Register (P2CONL)

Port 2 Pull-up Resistor Enable Register (P2PUR)

Using the port 2 pull-up resistor enable register, P2PUR (EDH, set 1), you can configure pull-up resistors to individual port 2 pins.

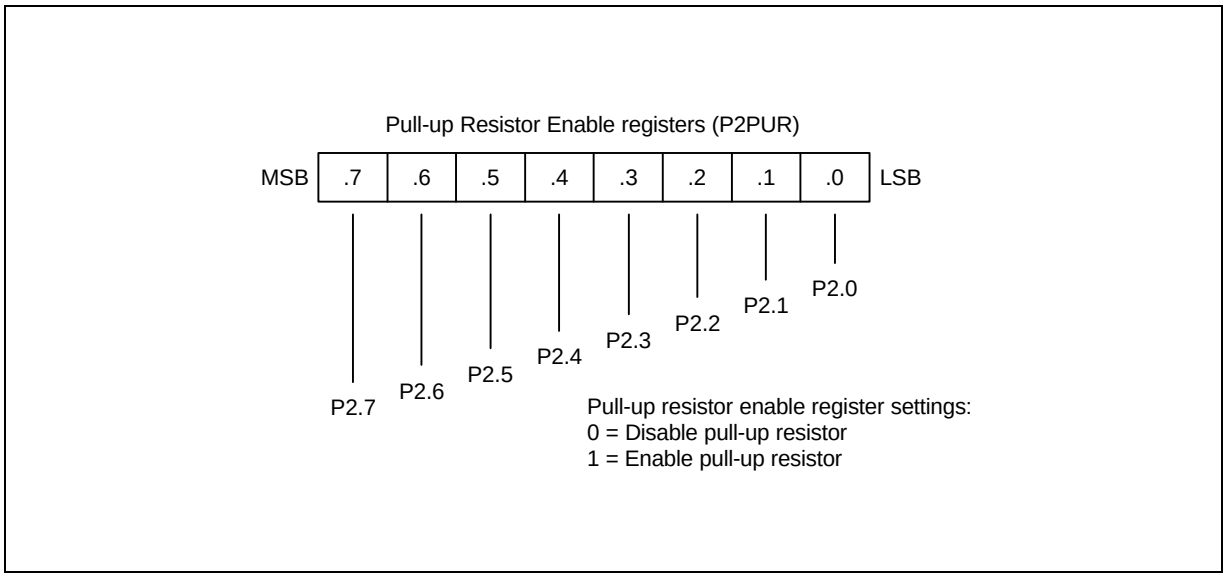


Figure 9-7. Pull-up Resistor Enable Registers (Ports 2)

PORT 3

Port 3 is an 8-bit I/O port with individually configurable pins. Port 3 pins are accessed directly by writing or reading the port 3 data register, P3 at location E3H in set 1. P3.0–P3.7 can serve as schmitt trigger inputs (with or without pull-ups), as push-pull outputs, or A/D converter input.

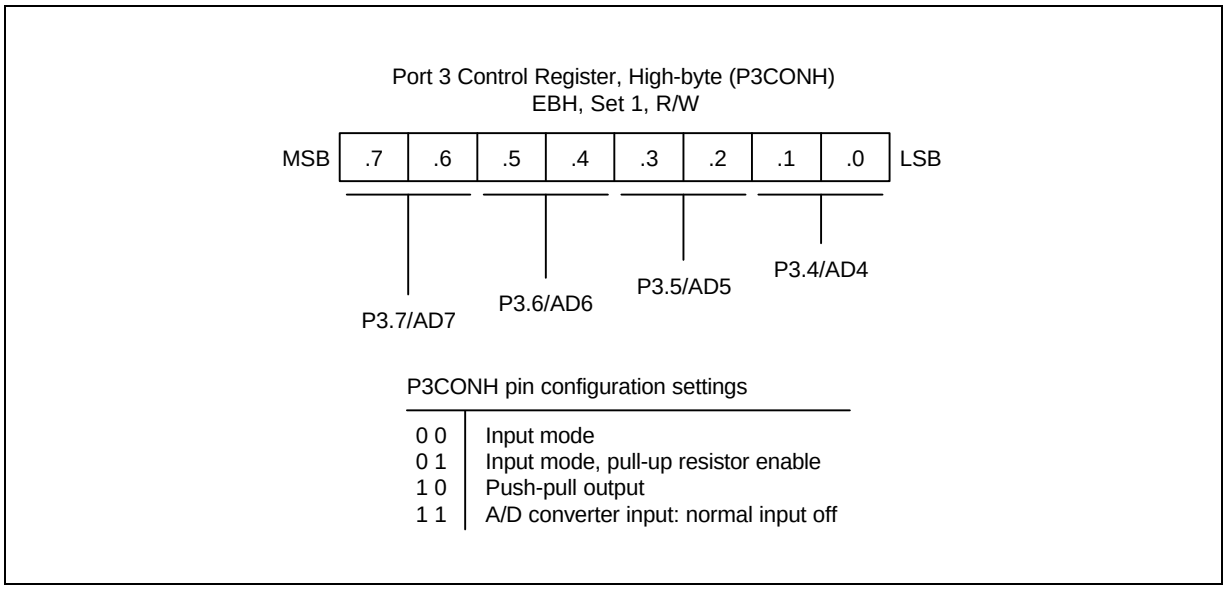


Figure 9-8. Port 3 High-Byte Control Register (P3CONH)

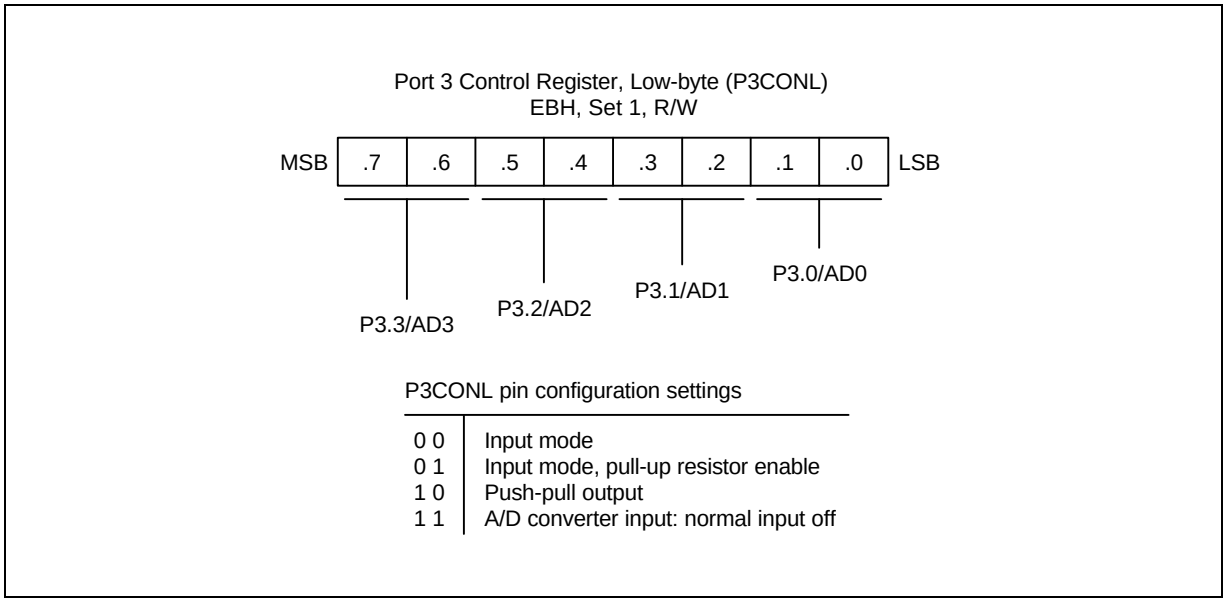


Figure 9-9. Port 3 Low-Byte Control Register (P3CONL)

PORT 4

Port 4 has 4 I/O pins and 2 output only pins.

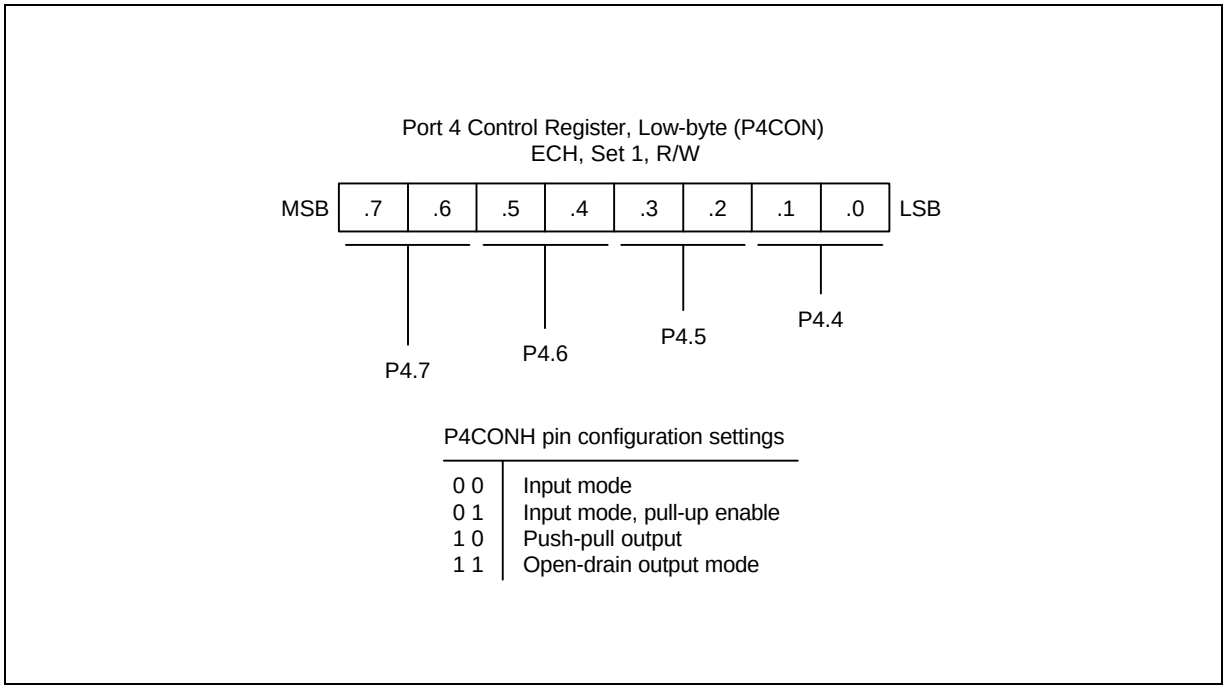


Figure 9-10. Port 4 High-Byte Control Register (P4CON)

NOTES

10 BASIC TIMER and TIMER 0

MODULE OVERVIEW

The KS88C4708/C4716 has two default timers: an 8-bit *basic timer* and one 8-bit general-purpose timer/counter. The 8-bit timer/counter is called *timer 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
— 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 (FDH, set 1, read-only)
— Basic timer control register, BTCON (D3H, set 1, read/write)

Timer 0

Timer 0 has three operating modes, one of which you select by the appropriate T0CON setting:

— Interval timer mode
— Capture mode with a rising or falling edge trigger at the T0 pin
— PWM mode

Timer 0 has the following functional components:

— Clock frequency divider (f_{OSC} divided by 4096, 256, 8, or 1) with multiplexer
— 8-bit counter (T0CNT), 8-bit comparator, and 8-bit reference data register (T0DATA)
— I/O pin (P1.0/T0) for timer 0 capture input or match output
— Timer 0 overflow interrupt (T0OVF) and match/capture interrupt (T0INT) generation
— Timer 0 control register, T0CON

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, can be cleared at any time during the normal operation by writing a "1" to BTCON.1. To clear the frequency dividers for both the basic timer input clock and the timer 0 clock, you write a "1" to BTCON.0.

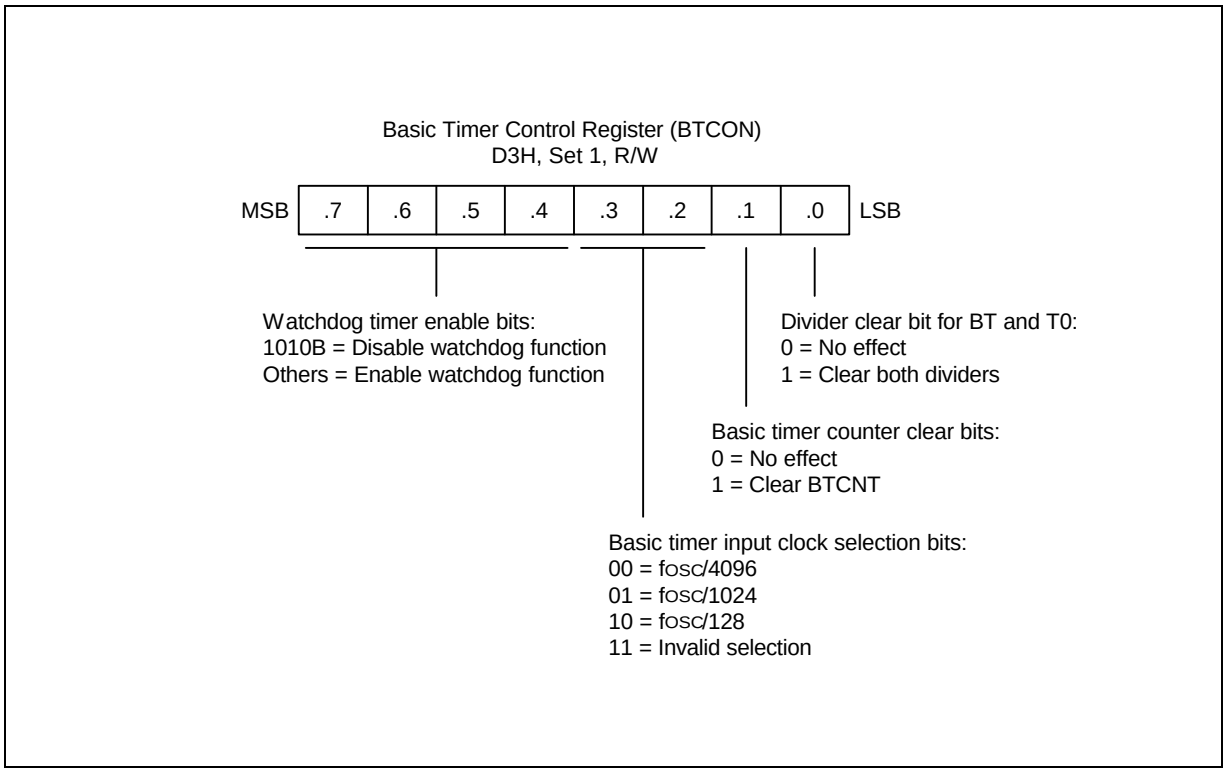


Figure 10-1. Basic Timer Control Register (BTCON)

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 Basic timer clock.

A reset whenever a basic timer counter overflow occurs. During the 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 Basic timer counter clear operation will not be executed and a basic timer overflow will occur, initiating a reset. In other words, during the 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.

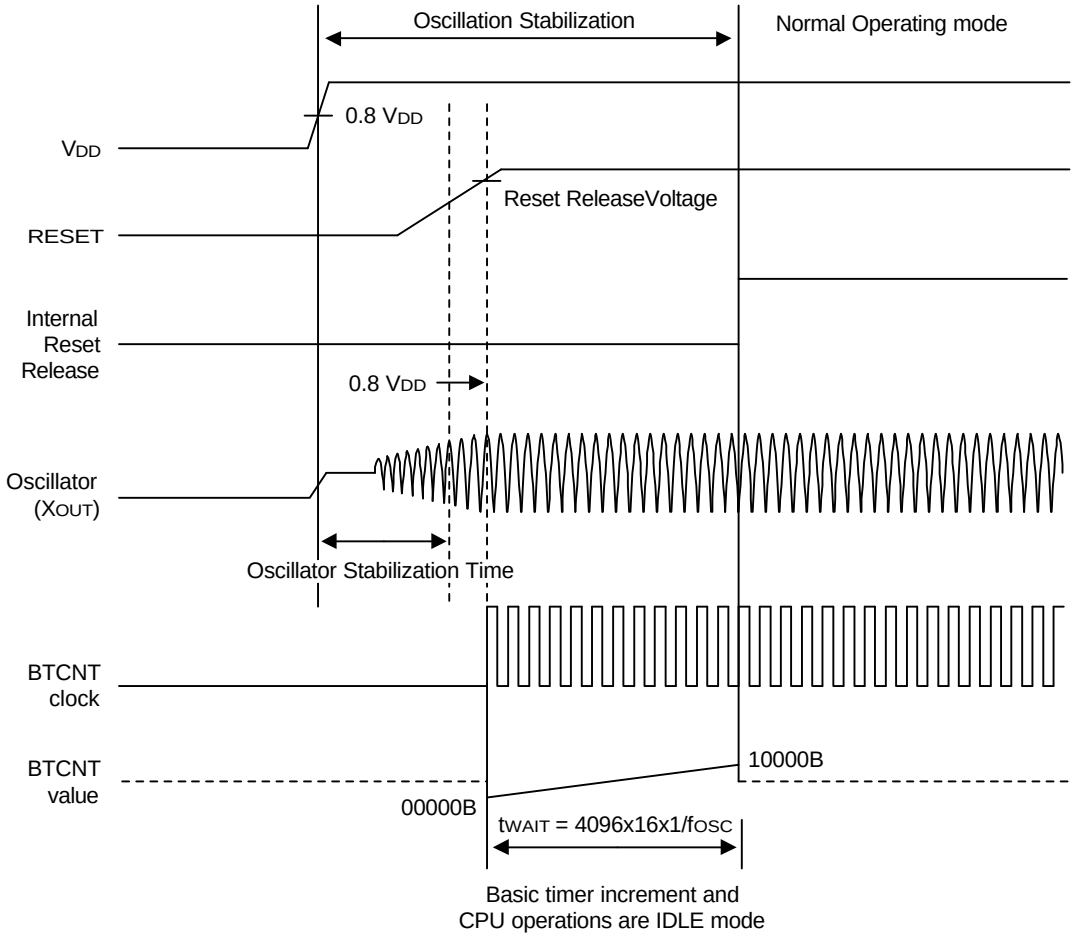
Oscillation Stabilization Interval Timer Function

You can also use the basic timer to program a specific oscillation stabilization interval after 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 is set, 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.

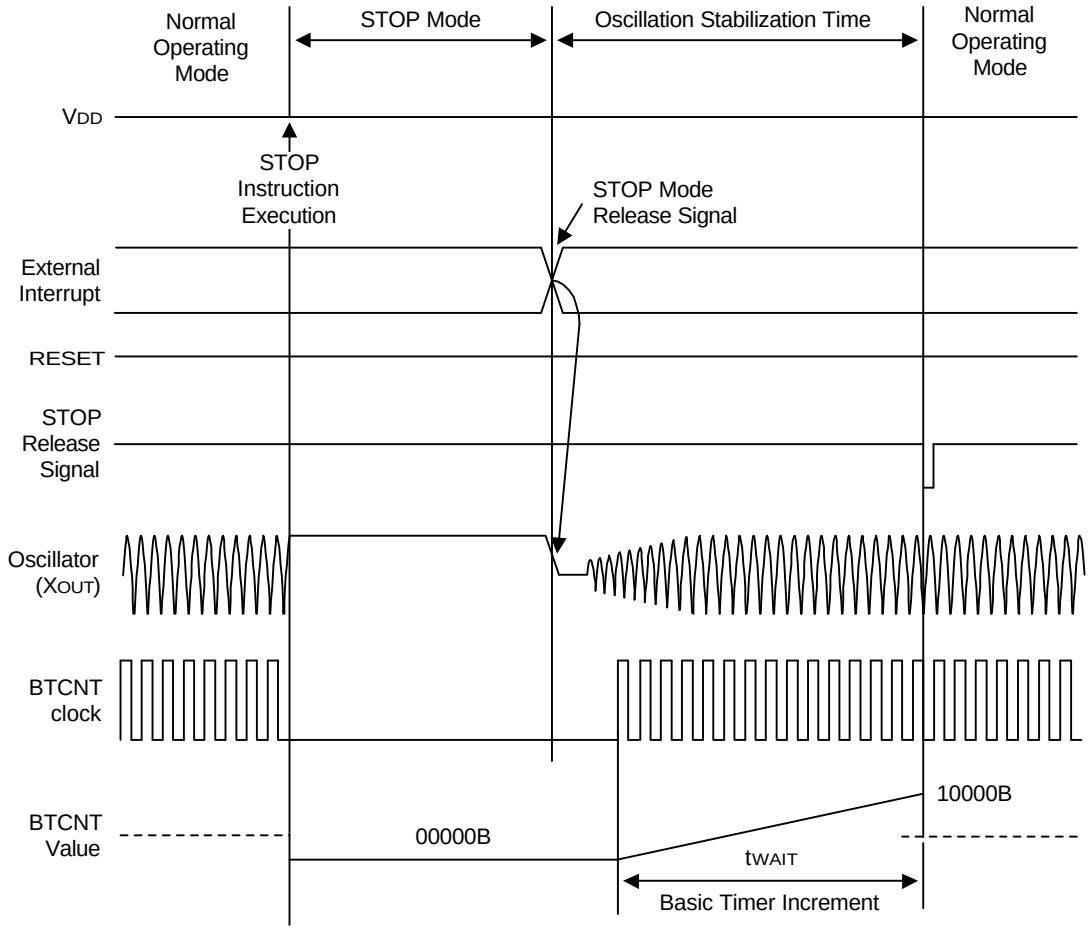
In summary, the following events occur when Stop mode is released:

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 is set.
4. When a BTCNT.4 is set, normal CPU operation resumes.
5. Figure 10-2 and 10-3 shows the oscillation stabilization time on RESET and STOP mode release



NOTE: During of the oscillator stabilization wait time, t WAIT, when it is released by a Power-on-reset is 1024x16/fosc.

Figure 10-2. Oscillation Stabilization Time on RESET



NOTE: Duration of the oscillator stabilzation wait time, tWAIT, it is released by an interrupt is determined by the setting in basic timer control register, BTCON.

BTCON.3	BTCON.2	tWAIT	tWAIT (When f OSC is 10 MHz)
0	0	4096 x 16/fosc	6.55 ms
0	1	1024 x 16/fosc	1.64 ms
1	0	128 x 16/fosc	0.2 ms
1	0	Invalid setting	—

Figure 10-3. Oscillation Stabilization Time on STOP Mode Release

TIMER 0 CONTROL REGISTER (T0CON)

You use the timer 0 control register, T0CON, to

— Select the timer 0 operating mode (interval timer, capture mode, or PWM mode)
— Select the timer 0 input clock frequency
— Clear the timer 0 counter, T0CNT
— Enable the timer 0 overflow interrupt and timer 0 match/capture interrupts
— Clear timer 0 match/capture interrupt pending conditions

T0CON is located in set 1 at address D2H, and is read/write addressable using Register addressing mode.

A reset clears T0CON to "00H". This sets timer 0 to normal interval timer mode, selects an input clock frequency of f_{OSC}/4096, and disables the timer 0 overflow interrupt and match/capture interrupts. You can clear the timer 0 counter at any time during the normal operation by writing a "1" to T0CON.3.

The timer 0 overflow interrupt (T0OVF) is interrupt level IRQ0 and has the vector address FAH. When a timer 0 overflow interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware.

To enable the timer 0 match/capture interrupt (IRQ0, vector FCH), you must write T0CON.1 to "1". To detect an interrupt pending condition, the application program polls T0CON.0. When a "1" is detected, a timer 0 match/capture interrupt is pending. When the interrupt request has been serviced, the pending condition must be cleared by software by writing a "0" to the timer 0 interrupt pending bit, T0CON.0.

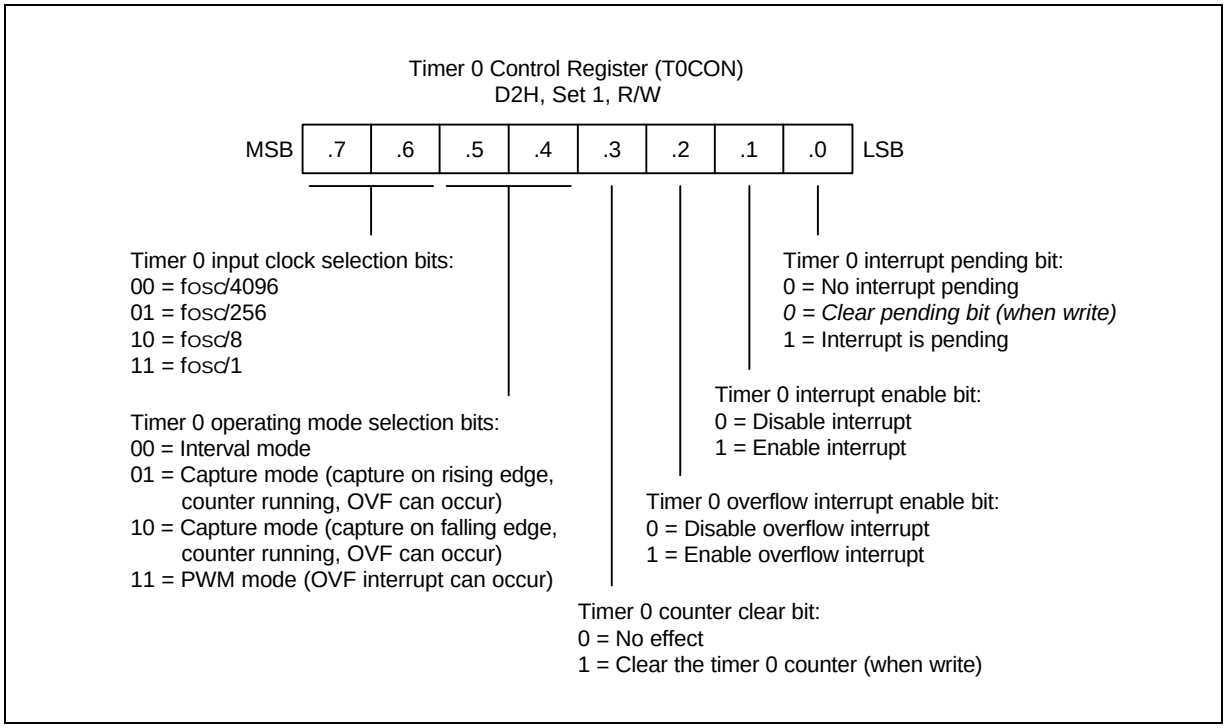


Figure 10-4. Timer 0 Control Register (T0CON)

TIMER 0 FUNCTION DESCRIPTION

Timer 0 Interrupts (IRQ0, Vectors FAH and FCH)

The timer 0 module can generate two interrupts: the timer 0 overflow interrupt (T0OVF), and the timer 0 match/ capture interrupt (T0INT). T0OVF is interrupt level IRQ0, vector FAH. T0INT also belongs to interrupt level IRQ0, but it has the vector address FCH.

A timer 0 overflow interrupt pending condition is automatically cleared by hardware when it has been serviced. The T0INT pending condition must be cleared by software, however, by writing a "0" to the T0CON.0 pending bit.

Interval Timer Mode

In interval timer mode, a match signal generates a timer 0 match interrupt (T0INT, vector FCH) and clears the counter. If you write the value "FFH" to the timer 0 reference data register, T0DATA, the counter will increment until an overflow occurs. (This condition is the same as normal counter operation.)

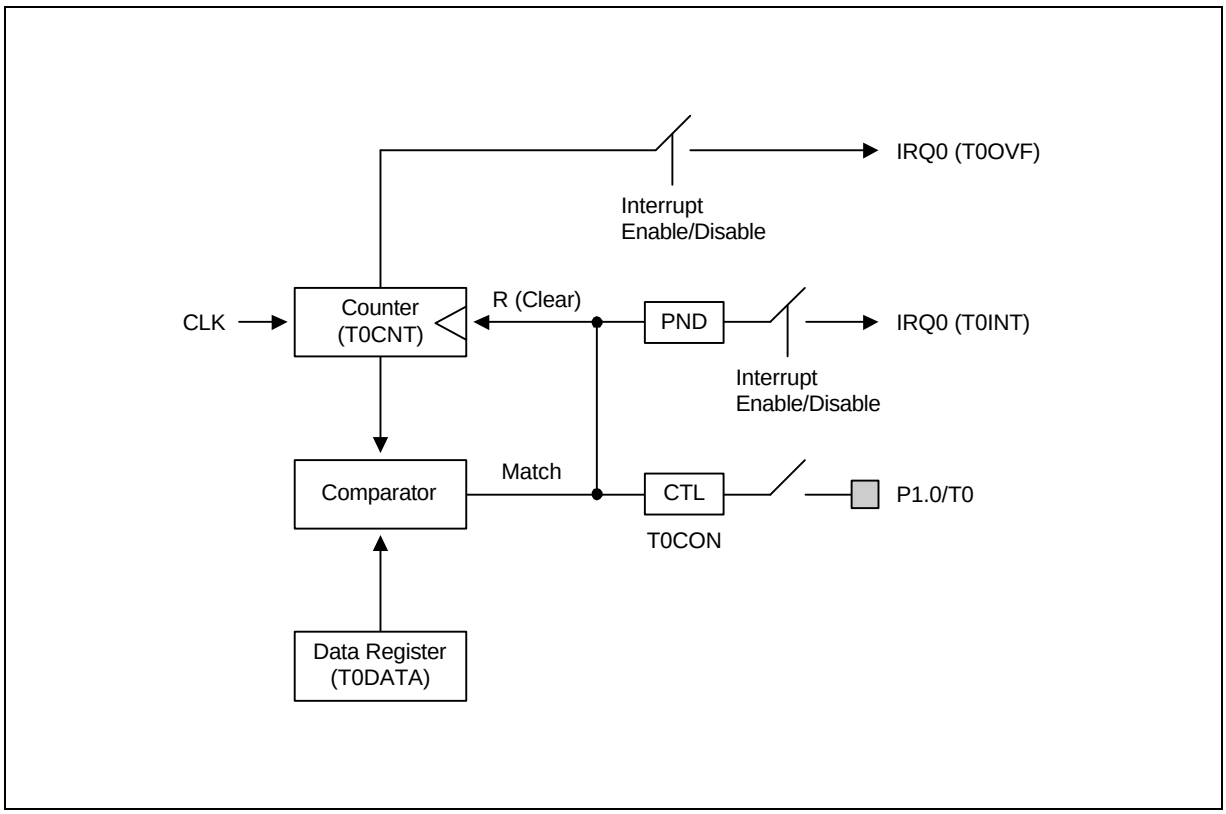


Figure 10-5. Simplified Timer 0 Function Diagram: Interval Timer Mode

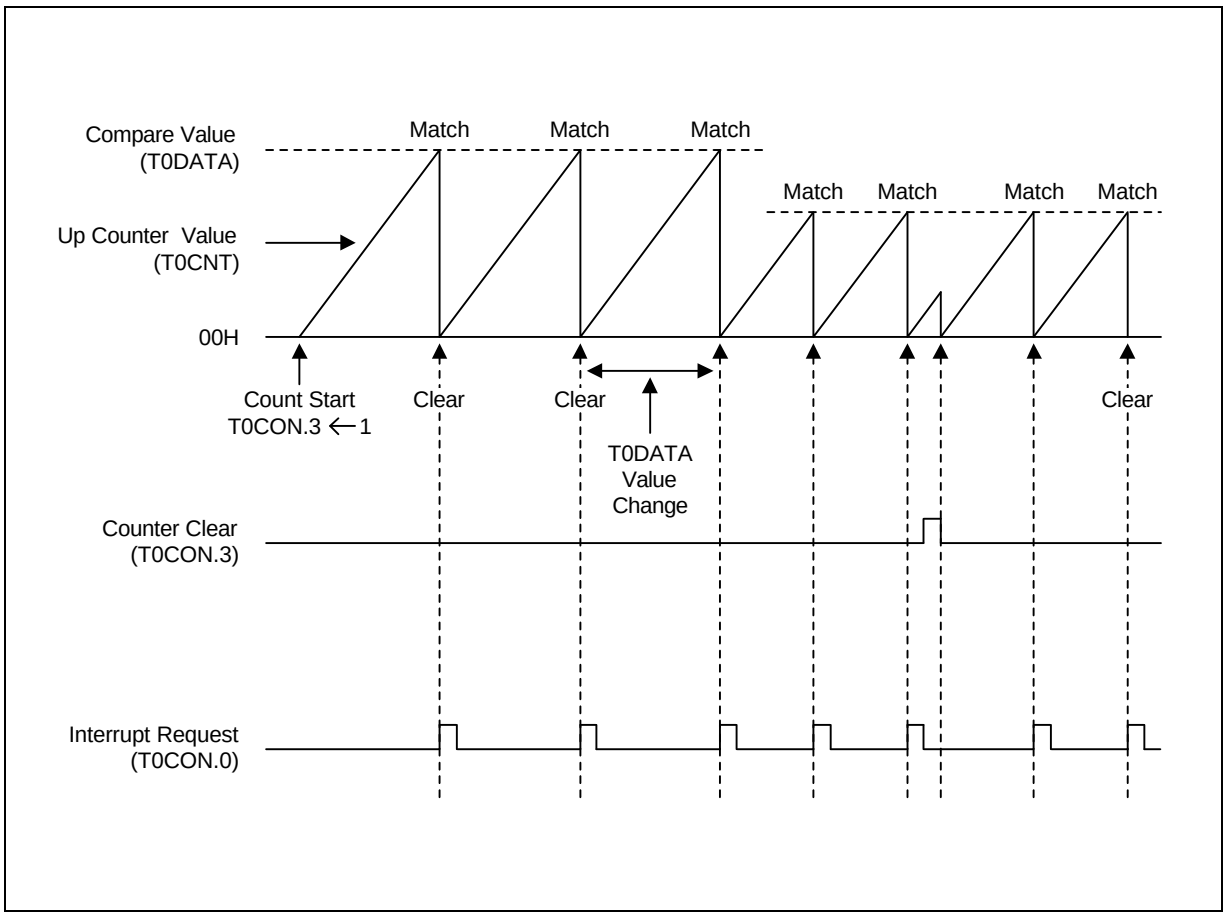


Figure 10-6. Timer 0 Timing Diagram

Pulse Width Modulation Mode

The PWM cycle width (time) is determined the timer 0 input clock. One cycle is equal to $t_{CLK} \times 2^8$ (for the 8-bit counter). The timer 0 data register value determines the pulse modulation width. (The minimum value is Low level and the maximum value is High level.) A match signal generates the timer 0 interrupt (T0INT, vector FCH), but it does not clear the timer 0 counter value.

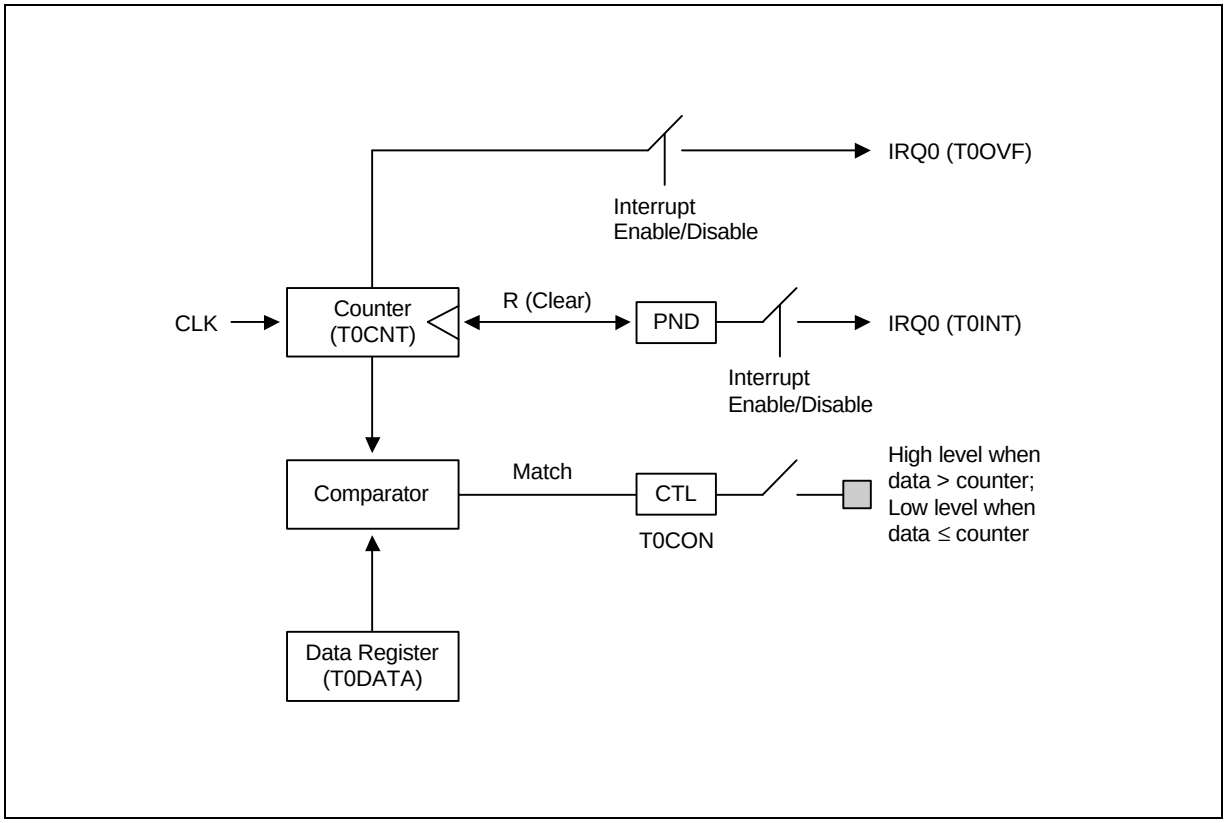


Figure 10-7. Simplified Timer 0 Function Diagram: PWM Mode

PWM FUNCTION DESCRIPTION

The 8-bit counter counts modulus 256, that is, from 0–255, inclusive. The value of the 8-bit counter is compared to the contents of the reference registers, T0DATA. When the reference register value equals the counter value, the PWM output goes low. When the counter reaches zero, the PWM output is forced high. The low-to-high ratio (duty) of the PWM output is T0DATA/256.

All PWM outputs remain inactive during the first 256 input clock signals. Then, when the counter value changes from FFH back to 00H, the PWM outputs are forced to high level. The pulse width ratio (duty cycle) is defined by the contents of the reference register and is programmed in increments of 1:256. The 8-bit PWM data register T0DATA is read and written using 8-bit register addressing mode.

PWM output can be held at low level by continuously loading the reference register with 00H. By continuously loading the reference register with FFH, you can hold the PWM output to high level, except for the last pulse of the clock source, which sends the output low (see Figure 10-8).

Table 10-1. PWM Reference Register Duty Values

Reference Register Value (T0DATA)	Duty
0000 0000	0/256 (0 %)
0000 0001	1/256 (0.39 %)
0000 0010	2/256 (0.78%)
•	•
•	•
1000 0000	128/256 (50 %)
1000 0001	129/256 (50.4 %)
•	•
•	•
1111 1110	254/256 (99.2 %)
1111 1111	255/256 (99.6 %)

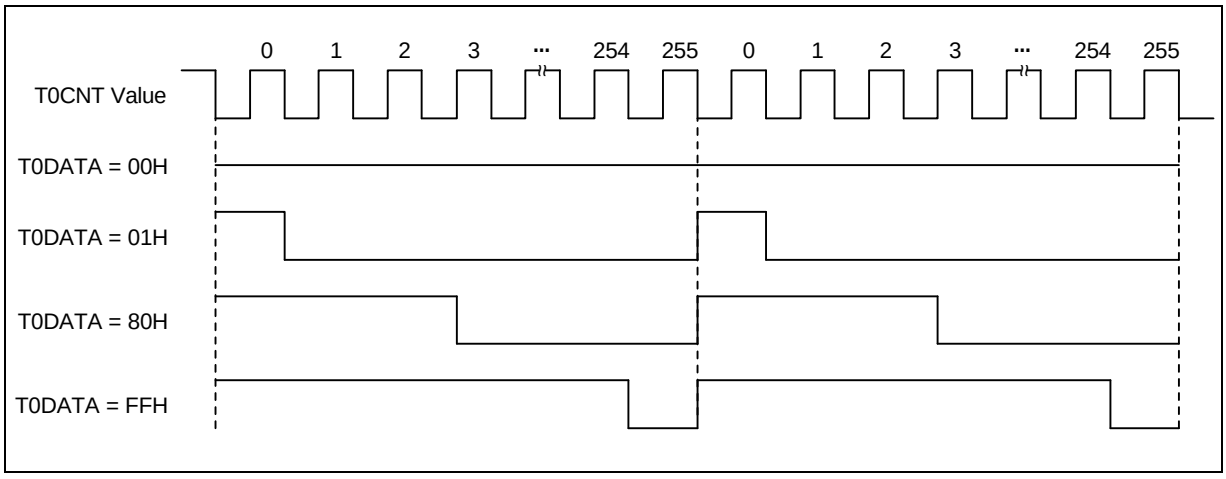


Figure 10-8. PWM Output Waveforms

Capture Mode

In capture mode, the timer 0 counter increases at a rate determined by the timer clock. The trigger signal (a rising or falling edge) for the capture operation occurs at the T0 pin. The interrupt service routine stores the captured 8-bit timer 0 counter value in the timer 0 data register whenever the capture signal is detected at the T0 pin.

By reading the counter value at programmed intervals, the routine can compare the differences between successive read values and calculate elapsed time interval. For example, if 80H is read and then 90H is read, this gives a difference of 10H. Depending on the clock speed, you can convert this hexadecimal value into the equivalent time interval (in this case, it is approximately 10 milliseconds).

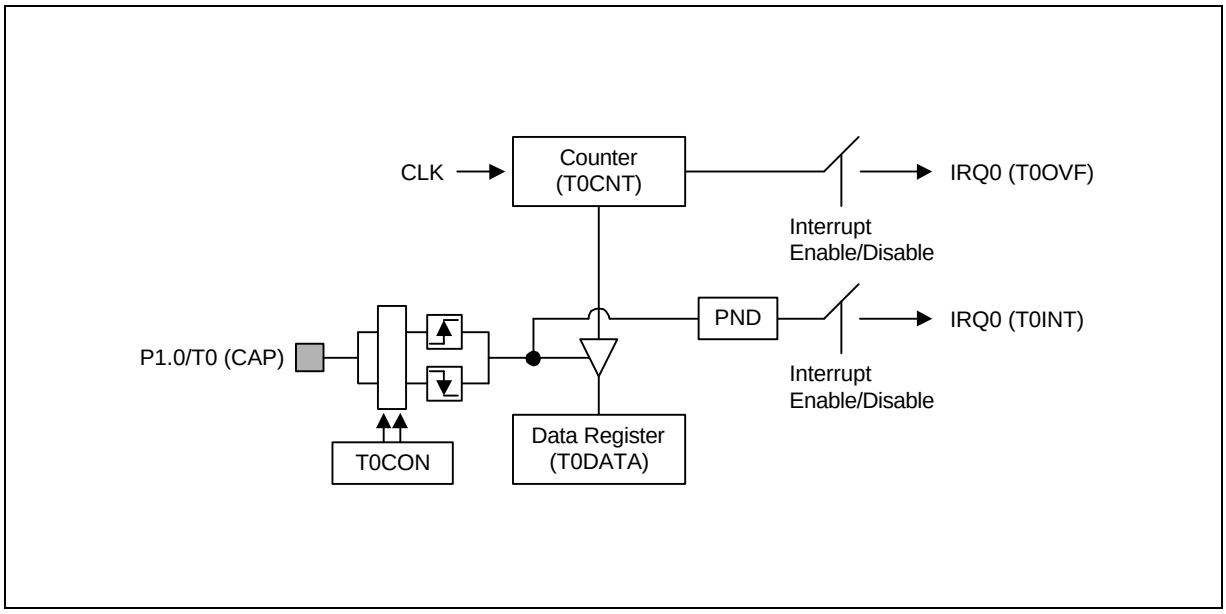
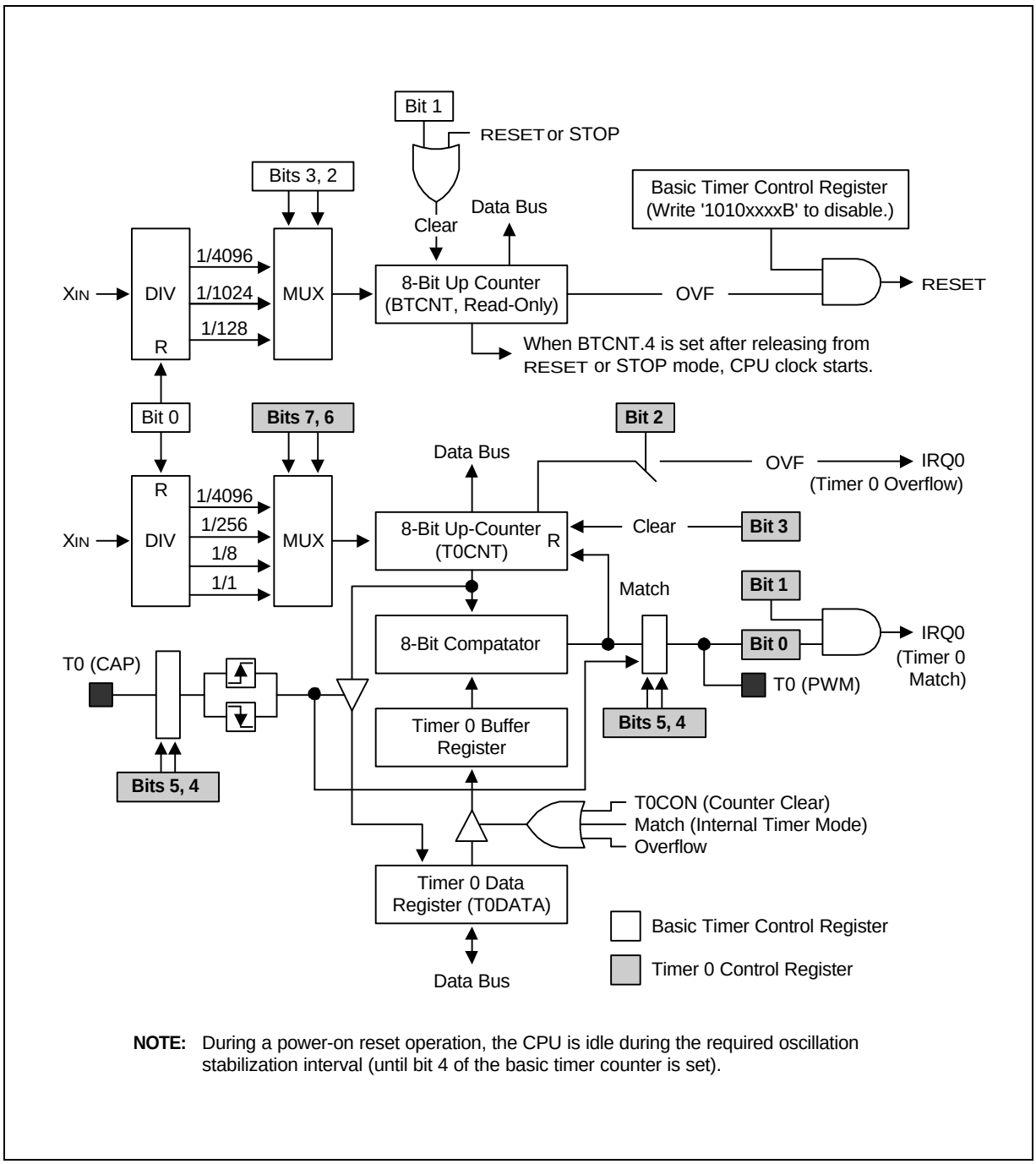


Figure 10-9. Simplified Timer 0 Function Diagram: Capture Mode



NOTE: During a power-on reset operation, the CPU is idle during the required oscillation stabilization interval (until bit 4 of the basic timer counter is set).

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

☞ PROGRAMMING TIP — Basic Timer

```
                ORG     0100H                   ;  Reset start address

RESET:
                •
                •
                •
                LD      BTCON,#10100011B        ;  Watchdog disable
                •
                •
                •

MAIN:
                LD      BTCON,#11110010B        ;  Watchdog enable
                                                ;  Basic timer clock: fOSC/4096
                                                ;  Clear basic timer counter before overflow
                •
                •
                •
                JR      t,MAIN
                •
                •
                •
```

☞ PROGRAMMING TIP — Timer 0

```
                ORG     0000H

                VECTOR  0FAH,T0OF_INT   ;  Timer 0 overflow interrupt
                VECTOR  0FCH,T0MC_INT   ;  Timer 0 interrupt (match/capture)

                ORG     0100H           ;  Reset start address

RESET:
                LD      SYM,#00H        ;  Global/Fast interrupt disable

                LD      IMR,#00000001B  ;  IRQ0 (Timer 0) interrupt enable

                LD      BTCON,#10100011B ;  Watchdog disable

                LD      CLKCON,#00011000B ;  Select non-divided oscillator frequency as CPU clock
                •
                •
                LD      T0CON,#01000010B ;  Input clock: fOSC/256
                                         ;  Interval mode
                                         ;  Disable timer 0 overflow interrupt
                                         ;  Enable timer 0 interrupt
                LD      T0DATA,#3DH      ;  Set timer interval to 1.4 ms
                                         ;  3DH/(11.0592MHz/256) = 0.7KHz (1.4ms)
                •
                •
                EI                       ;  Enable interrupt
MAIN:
                •
                •
                CALL    JOB              ;  Sub-block module
                •
                •
                JR      t,MAIN           ;  To loop main routine

T0MC_INT:
                PUSH    RP0              ;  Save RP0 to SATCK
                SRP     #60H             ;  RP0 ← 60H
                •
                •
                LD      T0CON,#01000010B ;  Clear timer 0 pending bit
                POP     RP0              ;  Restore RP0
                IRET                     ;  Return from interrupt service routine

T0OF_INT:
                IRET
```

11 TIMER 1

OVERVIEW

The 16-bit *timer1* is an 16-bit general-purpose timer/counter. Timer 1 has three operating modes, one of which you select using the appropriate T1CON setting:

— Interval timer mode(Toggle output at T1OUT pin)

— Capture input mode with a rising or falling edge trigger at the T1CAP pin

— PWM mode (T1PWM)

Timer 1 has the following functional components:

— Clock frequency divider (f_{OSC} divided by 1024,256, 64, 8 or 1) with multiplexer

— External clock input pin (T1CK)

— 16-bit counter (T1CNTH/L), 16-bit comparator, and 16-bit reference data register (T1DATAH/L)

— I/O pins for capture input (T1CAP), or PWM or match output(T1PWM, T1OUT)

— Timer 1 overflow interrupt (IRQ1, vector F4H) and match/capture interrupt (IRQ1, vector F6H) generation

— Timer 1 control register, T1CON (set 1, F6H, read/write)

FUNCTION DESCRIPTION

Timer 1 Interrupts (IRQ1, Vectors F4H and F6H)

The timer 1 module can generate two interrupts: the timer 1 overflow interrupt (T1OVF), and the timer 1 match/ capture interrupt (T1INT). T1OVF is interrupt level IRQ1, vector F4H. T1INT also belongs to interrupt level IRQ1, but is assigned the separate vector address, F6H. A timer 1 overflow interrupt pending condition is automatically cleared by hardware when it has been serviced. A timer 1 match/capture interrupt, T1INT pending condition is also cleared by hardware when it has been serviced.

Interval Timer Function

The timer 1 module can generate an interrupt: the timer 1 match interrupt (T1INT). T1INT belongs to interrupt level IRQ1, and is assigned the separate vector address, F6H. When a timer 1 measure interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware. In interval timer mode, a match signal is generated and T1OUT is toggled when the counter value is identical to the value written to the T1 reference data register, T1DATAH/L. The match signal generates a timer 1 match interrupt (T1INT, vector F6H) and clears the counter. If, for example, you write the value 0010H to T1DATAH/L and 06H to T1CON, the counter will increment until it reaches 0010H. At this point, the T1 interrupt request is generated, the counter value is reset, and counting resumes.

Pulse Width Modulation Mode

Pulse width modulation (PWM) mode lets you program the width (duration) of the pulse that is output at the T1PWM pin. As in interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer 1 data register. In PWM mode, however, the match signal does not clear the counter. Instead, it runs continuously, overflowing at 03FFH, and then continues incrementing from 0000H.

Although you can use the match signal to generate a timer 1 match interrupt, and also you can make timer 1 overflow interrupt, those interrupt are not typically used in PWM-type applications. Instead, the pulse at the T1PWM pin is held to Low level as long as the reference data value is *less than or equal to* (≤) the counter value and then the pulse is held to High level for as long as the data value is *greater than* (>) the counter value. One pulse WIDTH is equal to $t_{CLK} \infty 1024$.

Capture Mode

In capture mode, a signal edge that is detected at the T1CAP pin opens a gate and loads the current counter value into the T1 data register. You can select rising or falling edges to trigger this operation.

Timer 1 also gives you capture input source: the signal edge at the T1CAP pin. You select the capture input by setting the value of the timer 1 capture input selection bit in the port 1 control register, P1CONL, (set 1, E7H).

Both kinds of timer 1 interrupts can be used in capture mode: the timer 1 overflow interrupt is generated whenever a counter overflow occurs; the timer 1 match/capture interrupt is generated whenever the counter value is loaded into the T1 data register.

By reading the captured data value in T1DATAH/L, and assuming a specific value for the timer 1 clock frequency, you can calculate the pulse width (duration) of the signal that is being input at the T1CAP pin.

TIMER 1 CONTROL REGISTER (T1CON)

You use the timer 1 control register, T1CON, to

— Select the timer 1 operating mode (interval timer, capture mode, or PWM mode)
— Select the timer 1 input clock frequency
— Clear the timer 1 counter, T1CNTH/L
— Enable the timer 1 overflow interrupt or timer 1 match/capture interrupt
— Clear timer 1 match/capture interrupt pending conditions

TIMER 1 CONTROL REGISTER (T1CON) (Continued)

T1CON is located in set 1, at address F4H, and is read/write addressable using Register addressing mode.

A reset clears T1CON to '00H'. This sets timer 1 to normal interval timer mode, selects an input clock frequency of f_{OSC}/1024, and disables all timer 1 interrupts. To disable the counter operation, please set T1CON.7 -.5 to 111B. You can clear the timer 1 counter at any time during normal operation by writing a "1" to T1CON.3.

The timer 1 overflow interrupt (T1OVF) is interrupt level IRQ1 and has the vector address F4H. When a timer 1 overflow interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware.

To enable the timer 1 match/capture interrupt (IRQ1, vector F6H), you must write T1CON.1 to "1". To generate the exact time interval, you should write T1CON.2 which cleared counter and interrupt pending bit.

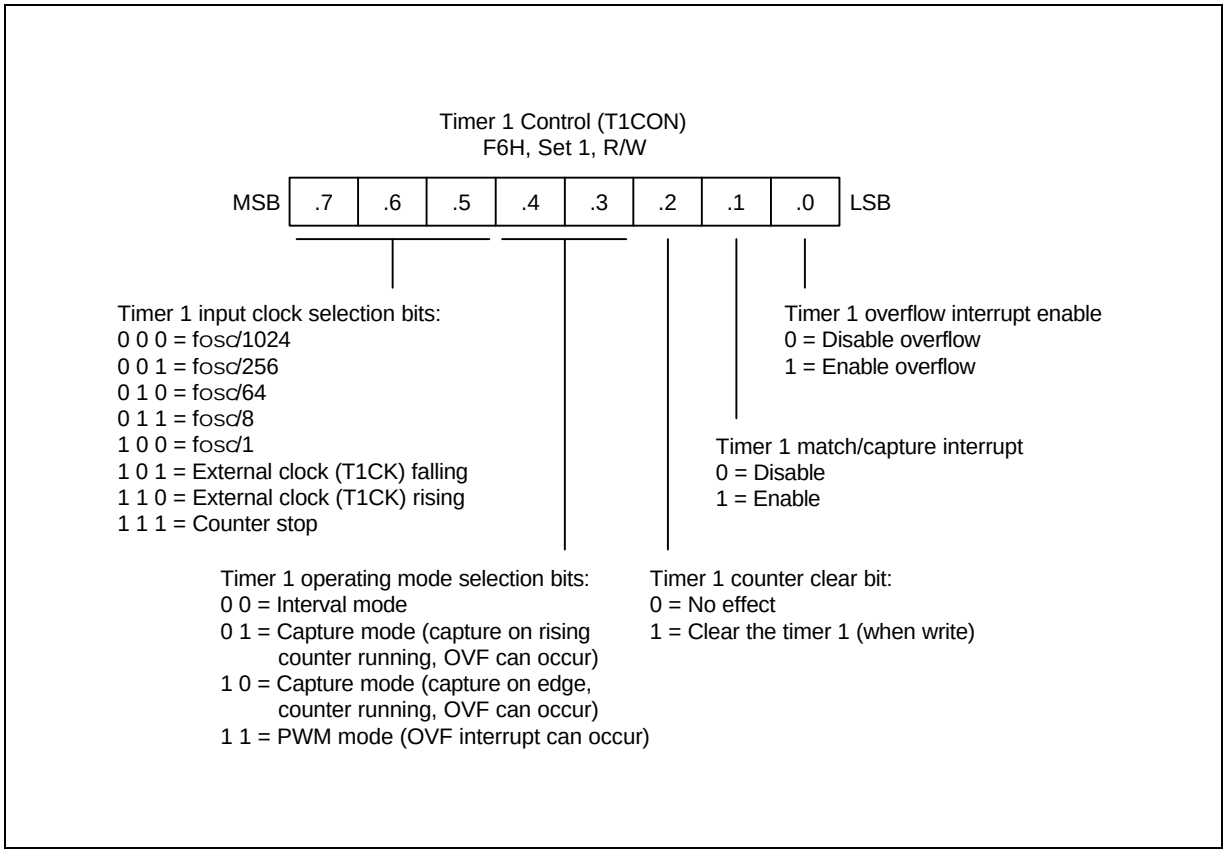


Figure 11-1. Timer 1 Control Register (T1CON)

BLOCK DIARAM

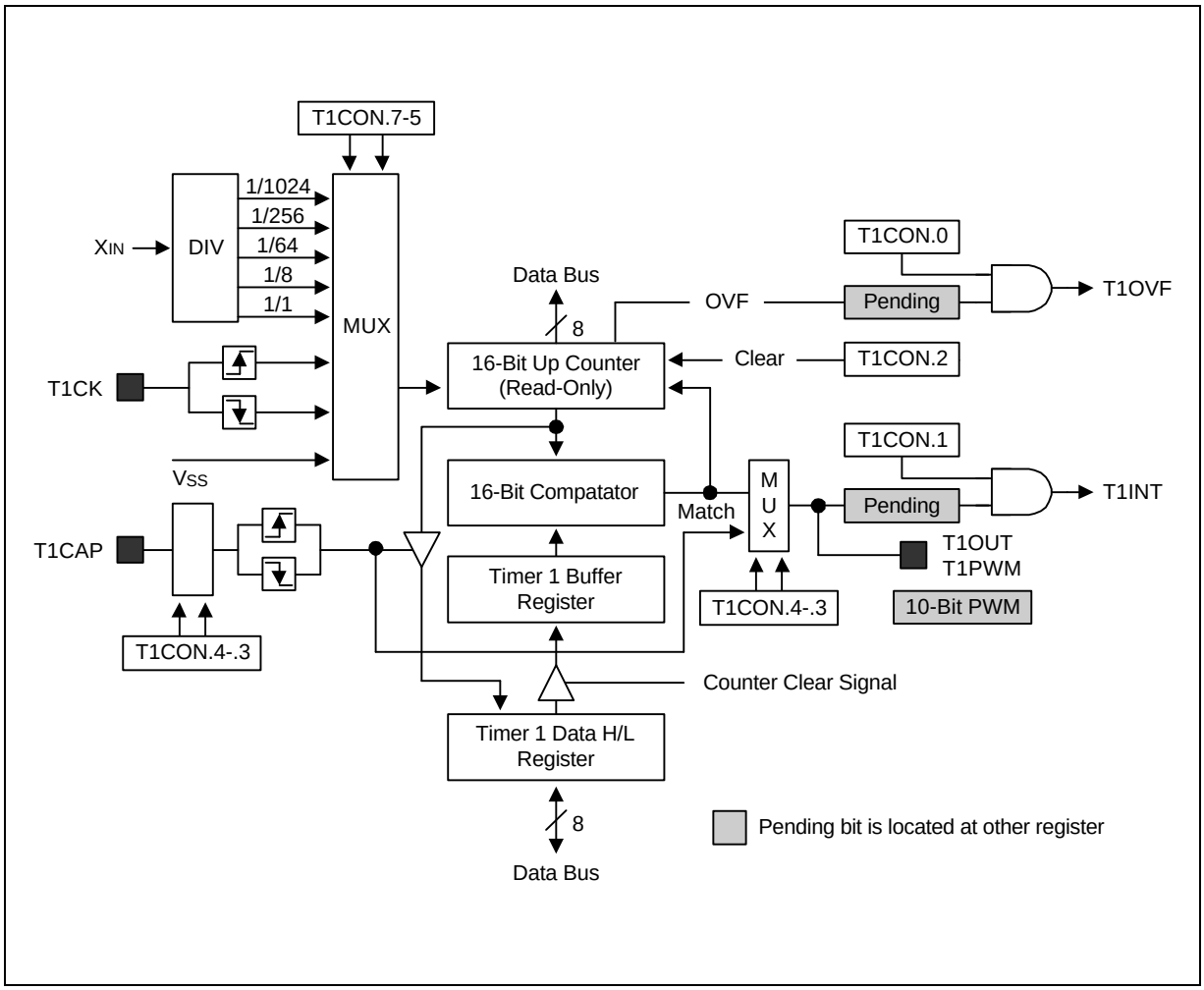


Figure 11-2. Timer 1 Functional Block Diagram

TIMER 1 FUNCTION DESCRIPTION

Timer 1 Interrupts (IRQ1, Vectors F4H and F6H)

The timer 1 module can generate two interrupts: the timer 1 overflow interrupt (T1OVF), and the timer 1 match/ capture interrupt (T1INT). T1OVF is interrupt level IRQ1, vector FAH. T1INT also belongs to interrupt level IRQ1, but it has the vector address F6H.

A timer 1 overflow interrupt pending condition is automatically cleared by hardware when it has been serviced.

Interval Timer Mode

In interval timer mode, a match signal generates a timer 1 match interrupt (T1INT, vector F6H) and clears the counter. If you write the value "FFFFH" to the timer 1 reference data register, T1DATA, the counter will increment until an overflow occurs. (This condition is the same as normal counter operation.)

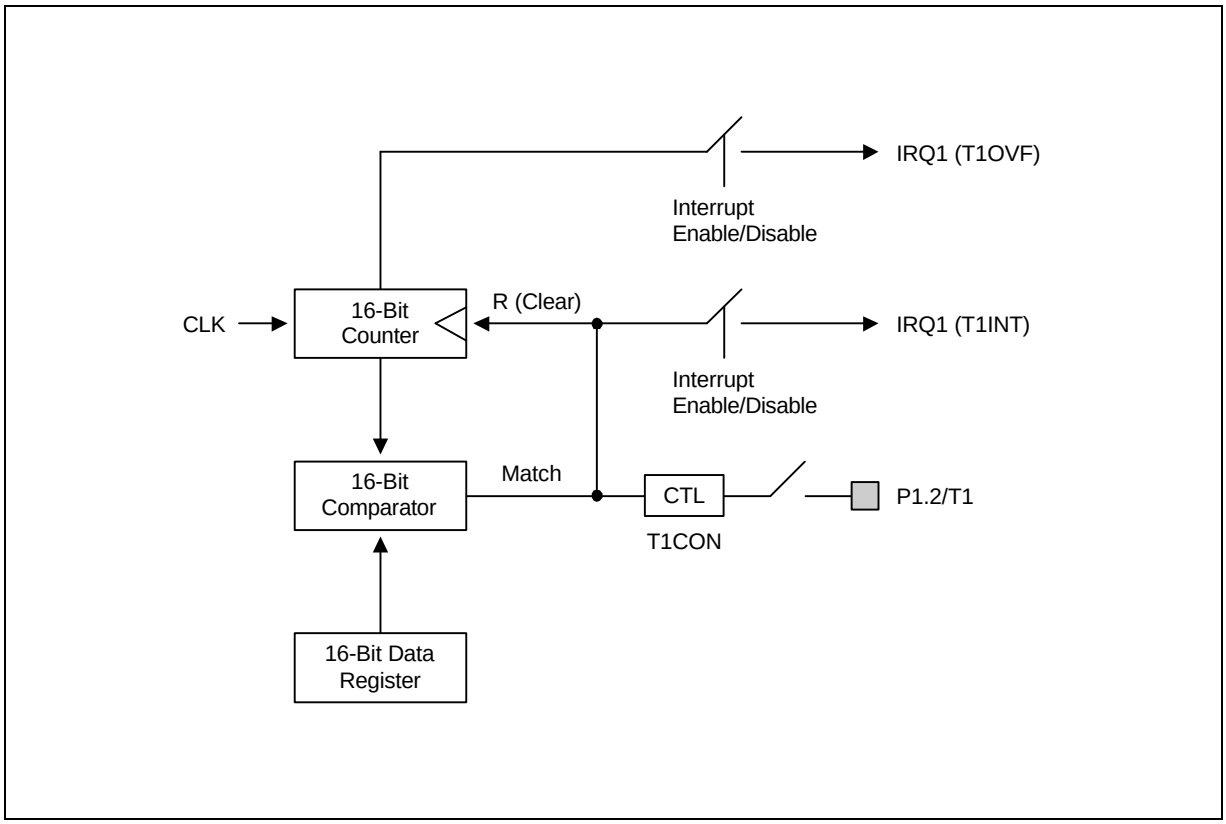


Figure 11-3. Simplified Timer 0 Function Diagram: Interval Timer Mode

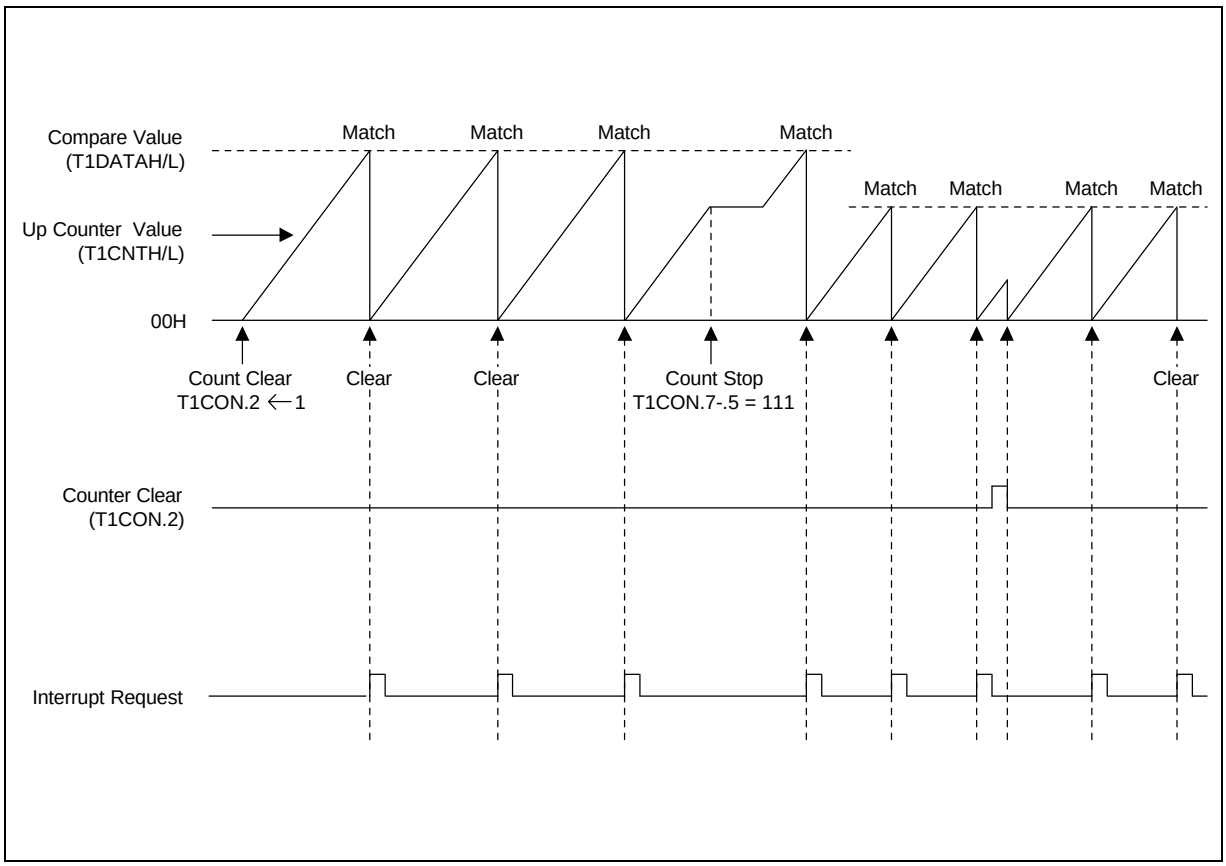


Figure 11-4. Timer 1 Timing Diagram (Internal Timer Mode)

Pulse Width Modulation Mode

The PWM cycle width (time) is determined the timer 1 input clock. One cycle is equal to $t_{CLK} \times 2^{10}$ (for the 10-bit counter). The timer 1 data register value determines the pulse modulation width. (The minimum value is Low level and the maximum value is High level.)

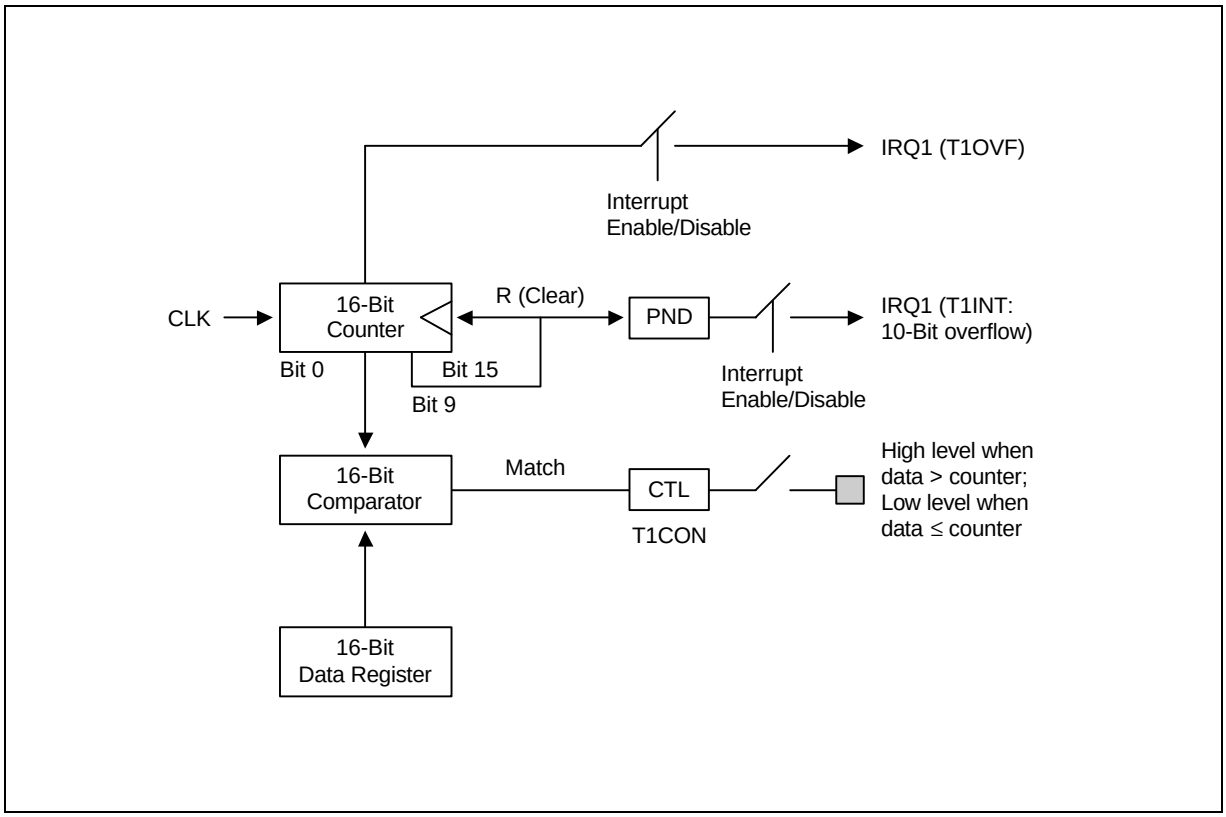


Figure 11-5. Simplified Timer 1 Function Diagram: PWM Mode

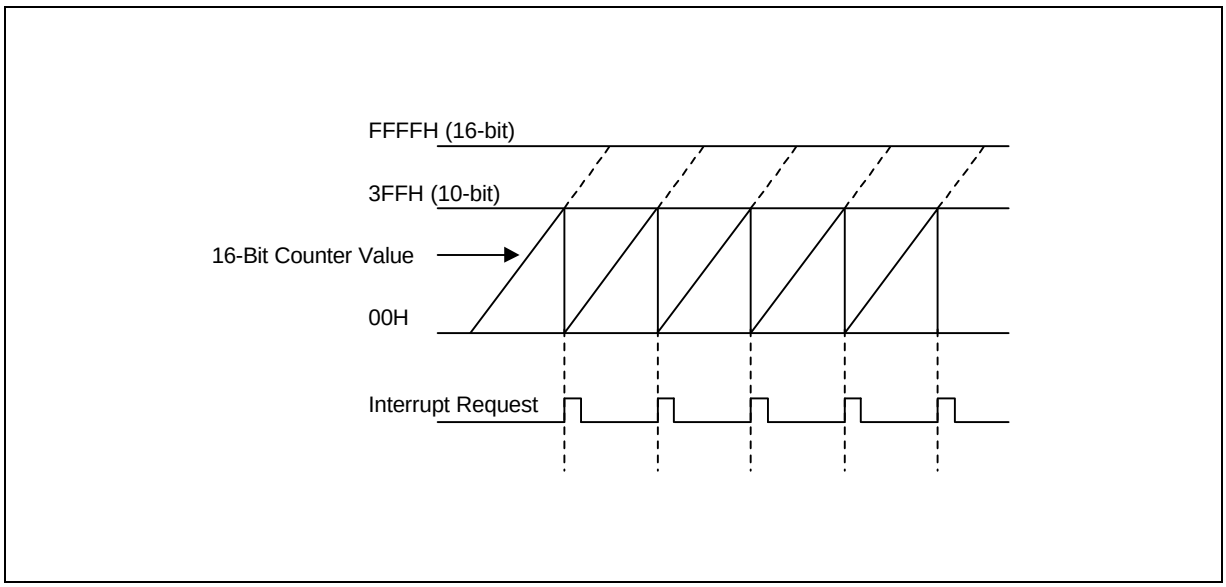


Figure 11-6. Timer 1 Overflow Interrupt in 10-Bit PWM Mode

PWM FUNCTION DESCRIPTION

The 16-bit counter counts modulus 1024, that is, from 0–1023, inclusive. The value of the 16-bit counter is compared to the contents of the reference registers, T1DATAH. When the reference register value equals the counter value, the PWM output goes low. When the counter reaches zero, the PWM output is forced high. The low-to-high ratio (duty) of the PWM output is (T1DATAH/L)/1024.

All PWM outputs remain inactive during the first input clock signals. Then, when the counter value changes from 3FFH back to 00H, the PWM outputs are forced to high level. The pulse width ratio (duty cycle) is defined by the contents of the reference register and is programmed in increments of 1:1024. The 10-bit PWM data register T1DATAH/L is read and written using register addressing mode.

PWM output can be held at low level by continuously loading the reference register with 00H. By continuously loading the reference register with 3FFH, you can hold the PWM output to high level, except for the last pulse of the clock source, which sends the output low (see Figure 11-7).

Table 11-1. PWM Reference Register Duty Values

Reference Register Value (T1DATA)	Duty
00 0000 0000	0/1024 (0 %)
00 0000 0001	1/1024 (0.097 %)
00 0000 0010	2/1024 (0.195%)
•	•
•	•
10 0000 0000	512/1024 (50 %)
10 0000 0001	513/1024 (50.097 %)
•	•
•	•
11 1111 1110	1022/1024 (99.8 %)
11 1111 1111	1023/1024 (99.9 %)

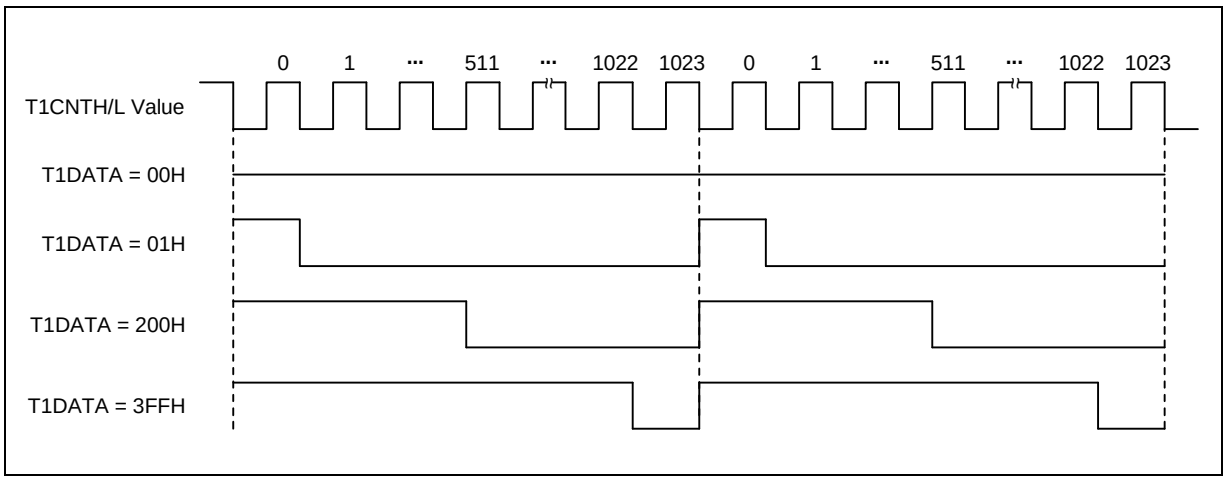


Figure 11-7. PWM Output Waveforms

Capture Mode

In capture mode, the timer 1 counter increases at a rate determined by the timer clock. The trigger signal (a rising or falling edge) for the capture operation occurs at the T1 pin. The interrupt service routine stores the captured 16-bit timer 1 counter value in the timer 1 data register whenever the capture signal is detected at the T1 pin.

By reading the counter value at programmed intervals, the routine can compare the differences between successive read values and calculate elapsed time interval. For example, if 1080H is read and then 20A0H is read, this gives a difference of 1020H. Depending on the clock speed, you can convert this hexadecimal value into the equivalent time interval.

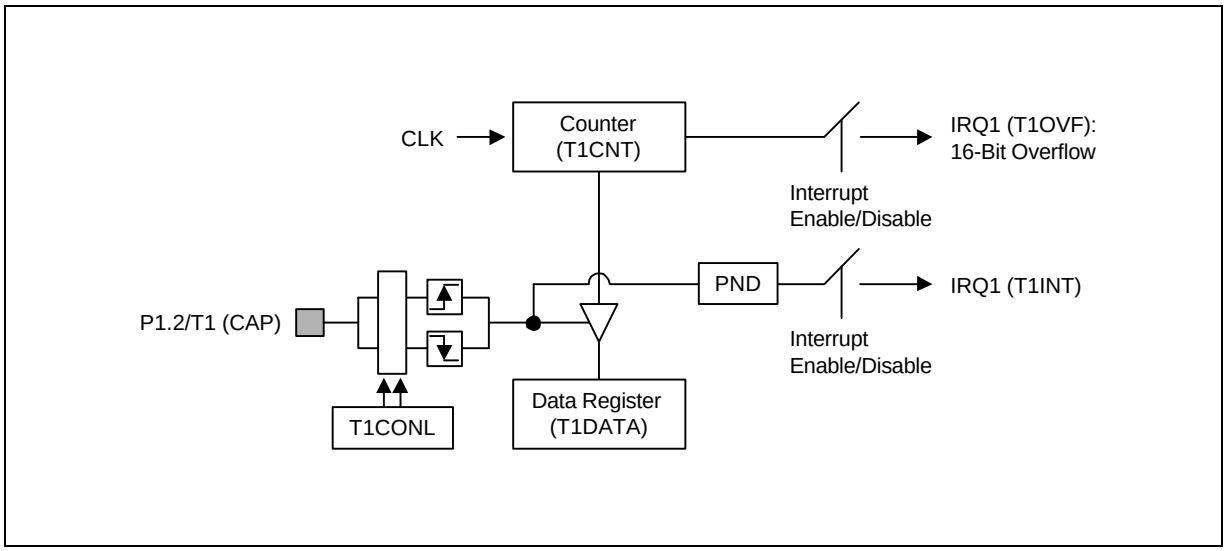


Figure 11-8. Simplified Timer 0 Function Diagram: Capture Mode

12 UART

OVERVIEW

The UART block 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 $f_{OSC}/(16 \times (BRDATA+1))$

— 8-bit UART mode; variable baud rate

— 9-bit UART mode; $f_{OSC}/16$

— 9-bit UART mode, variable baud rate

UART receive and transmit buffers are both accessed via the data register, UDATA (F0H). Writing to the UART data register loads the transmit buffer; reading the UART data register accesses a physically separate receive buffer.

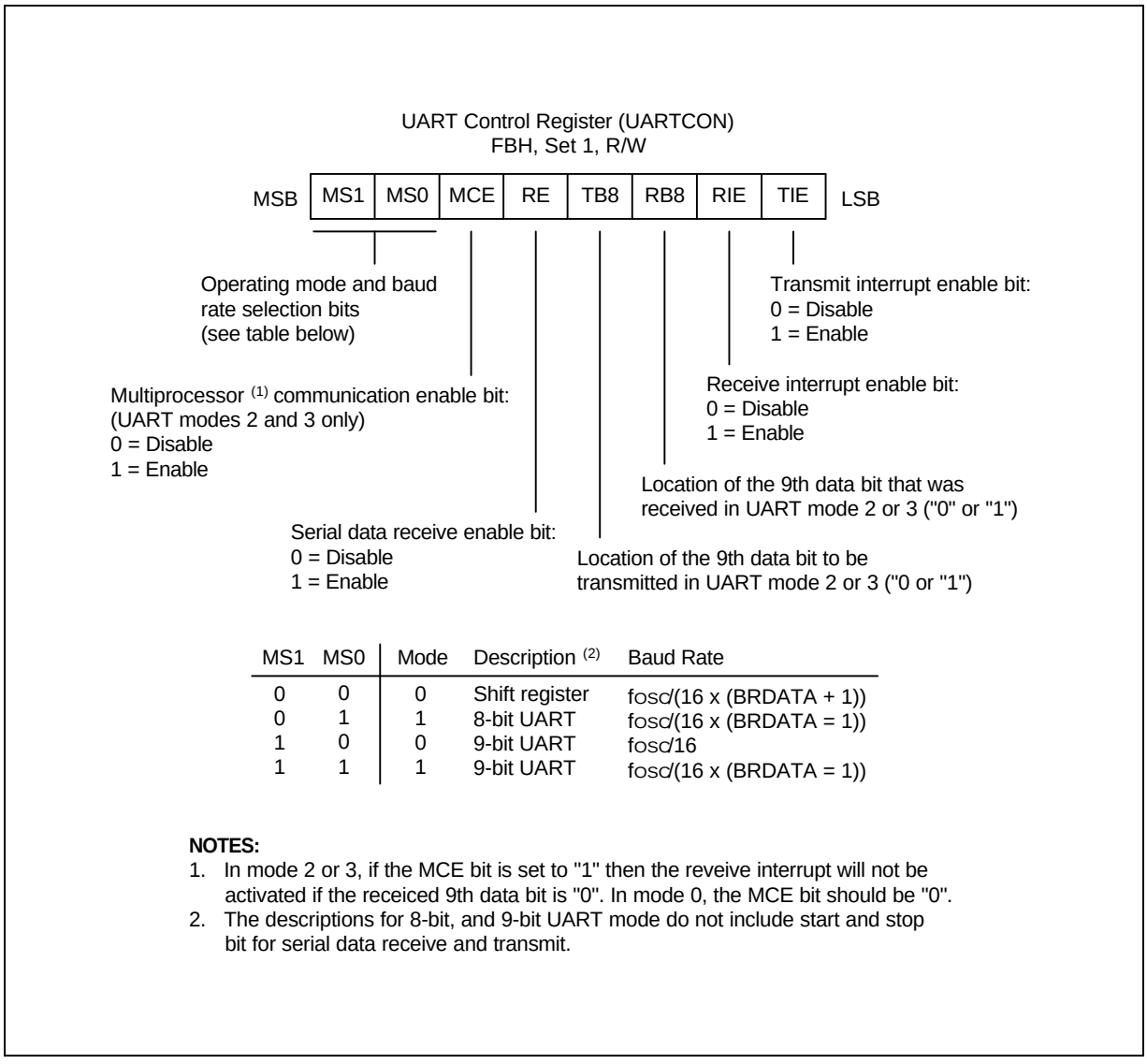
The UART block is receive-buffered. Using a receive data buffer, reception of the next byte can commence before the previously received byte has been read from the receive register. However, if the first byte has not been read by the time the next byte has been completely received, one of the bytes will be lost.

In all operating modes, transmission is initiated when any instruction addresses the UART data register UDATA (F0H) as its destination register. In mode 0, reception of serial data is initiated when the receive interrupt pending bit (RIP) in the UARTPND register is cleared to "0" and the receive enable bit (RE, UARTCON.4) is set to "1". In modes 1, 2, and 3, reception is initiated when the incoming start bit ("0") is received and the receive enable (RE) bit is "1".

UART CONTROL REGISTER (UARTCON)

The control register for the UART is called UARTCON (EEH). It has the following control functions:

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



MS1	MS0	Mode	Description (2)	Baud Rate
0	0	0	Shift register	fosc/(16 x (BRDATA + 1))
0	1	1	8-bit UART	fosc/(16 x (BRDATA = 1))
1	0	0	9-bit UART	fosc/16
1	1	1	9-bit UART	fosc/(16 x (BRDATA = 1))

NOTES:
1. In mode 2 or 3, if the MCE bit is set to "1" then the reveive interrupt will not be activated if the receiced 9th data bit is "0". In mode 0, the MCE bit should be "0".
2. The descriptions for 8-bit, and 9-bit UART mode do not include start and stop bit for serial data receive and transmit.

Figure 12-1. UART Control Register (UARTCON)

UART INTERRUPT PENDING REGISTER (UARTPND)

The UART interrupt pending register UARTPND (EFH, set 1) contains the UART data transmit interrupt pending bit (TIP) and the receive interrupt pending bit (RIP) in register positions UARTPND.0 and UARTPND.1, respectively.

In mode 0, the receive interrupt pending flag RIP is set to "1" when the 8th receive data bit has been shifted. In mode 1, 2, and 3, the RIP bit is set to "1" at the halfway point of the stop bit's shift time. When the CPU has acknowledged the receive interrupt pending condition, the RIP flag must then be cleared by software.

In mode 0, the transmit interrupt pending flag TIP is set when the 8th transmit data bit has been shifted. In mode 1, 2, or 3, the TIP bit is set at the start of the stop bit. When the CPU has acknowledged the transmit interrupt pending condition, the TIP flag must then be cleared by software.

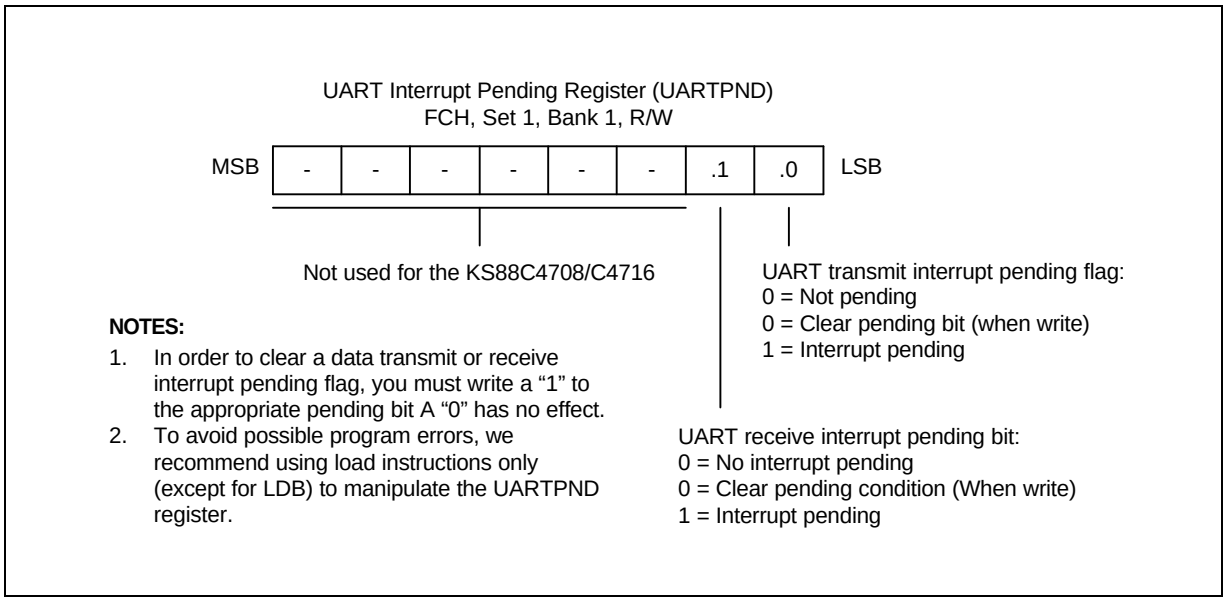


Figure 12-2. UART Interrupt Pending Register (UARTPND)

SERIAL COMMUNICATION FOR MULTIPROCESSOR CONFIGURATIONS

The multiprocessor communication feature allows a "master" to send a multiple-frame serial message to one "slave" device in a multi-MCU configuration. It does this without interrupting other slaves that may be on the same serial line.

This feature can be used only in UART modes 2 or 3. It is most commonly used with mode 2

In modes 2 and 3, nine data bits are received. The 9th bit value is written to RB8 (UARTCON.2). Then comes a stop bit. You can program this function so that when the stop bit is received, the serial port interrupt will be activated only if RB8 = "1". To enable this feature, you must set the multiprocessor communication enable bit in the UARTCON register (MCE, UARTCON.5). If the MCE bit is set to "1", serial data frames received in which the 9th bit is "0" do not generate an interrupt, but simply separate the address from the serial data

UART BAUD RATE DATA REGISTER (BRDATA)

The value stored in the UART baud rate register, BRDATA, lets you determine the UART clock rate (baud rate).

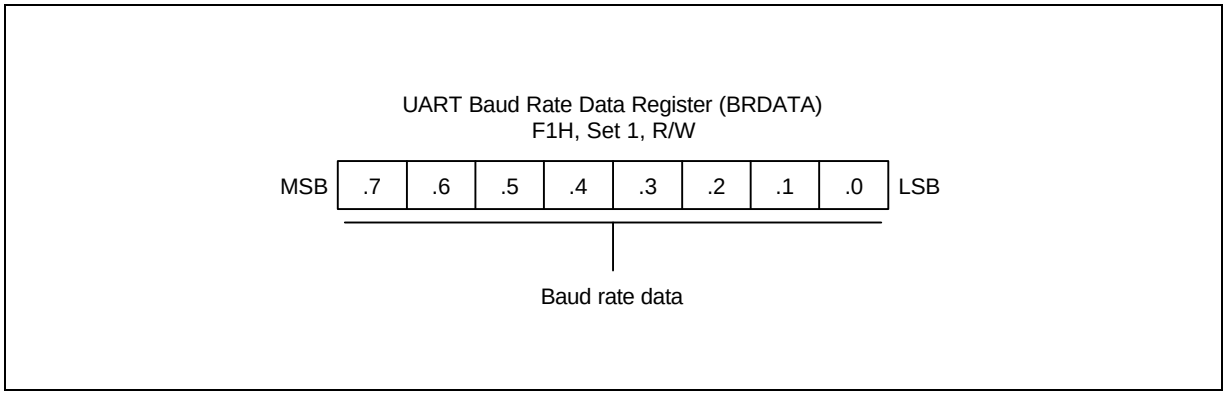


Figure 12-3. UART Baud Rate Data Register (BRDATA)

UART DATA REGISTER (UDATA)

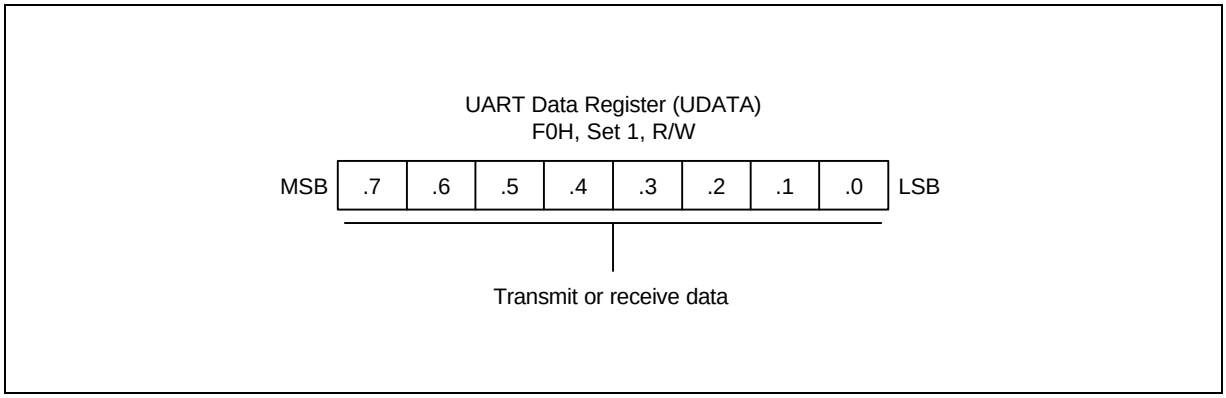


Figure 12-4. UART Data Register (UDATA)

BLOCK DIAGRAM

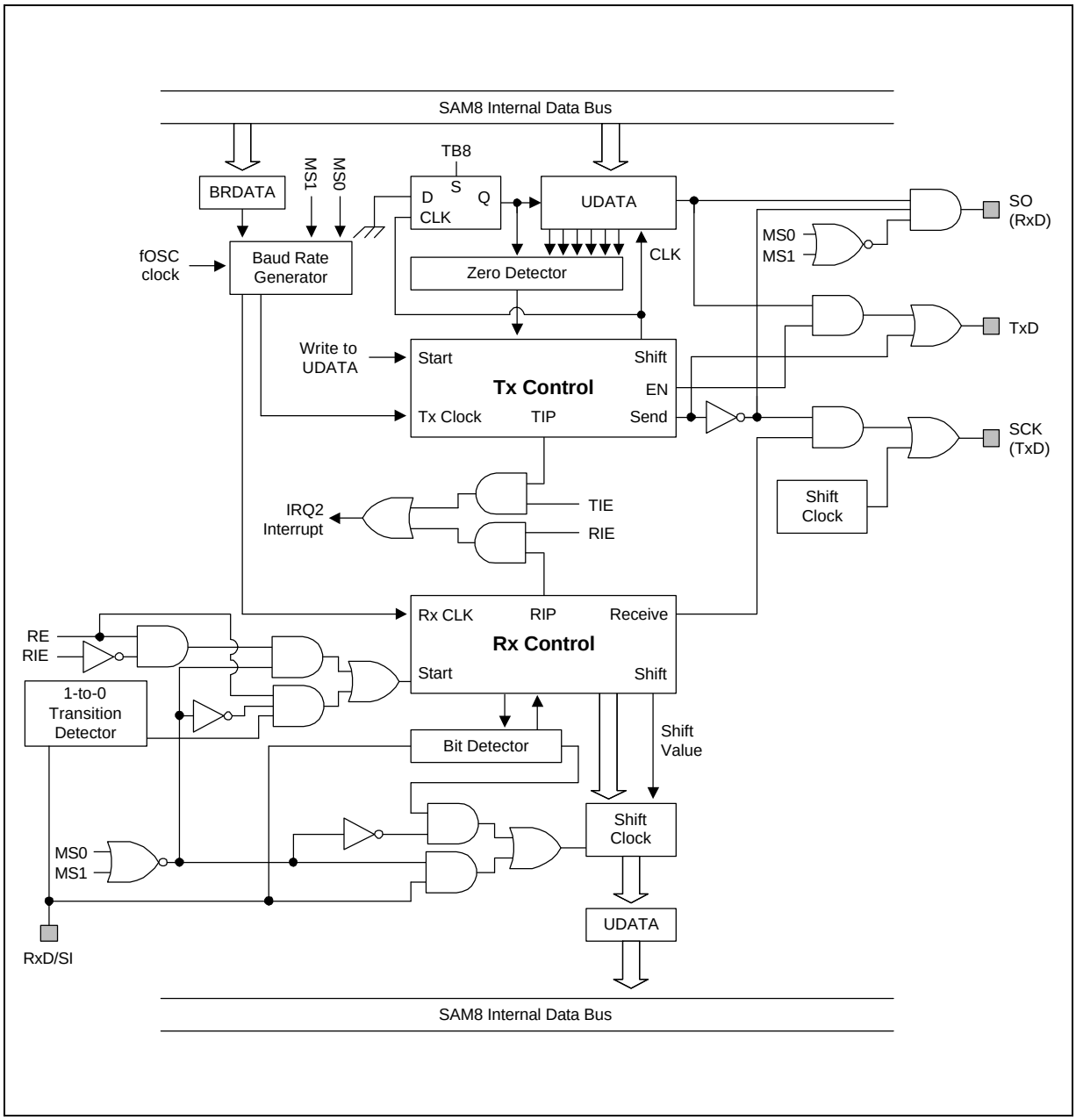


Figure 12-5. UART Functional Block Diagram

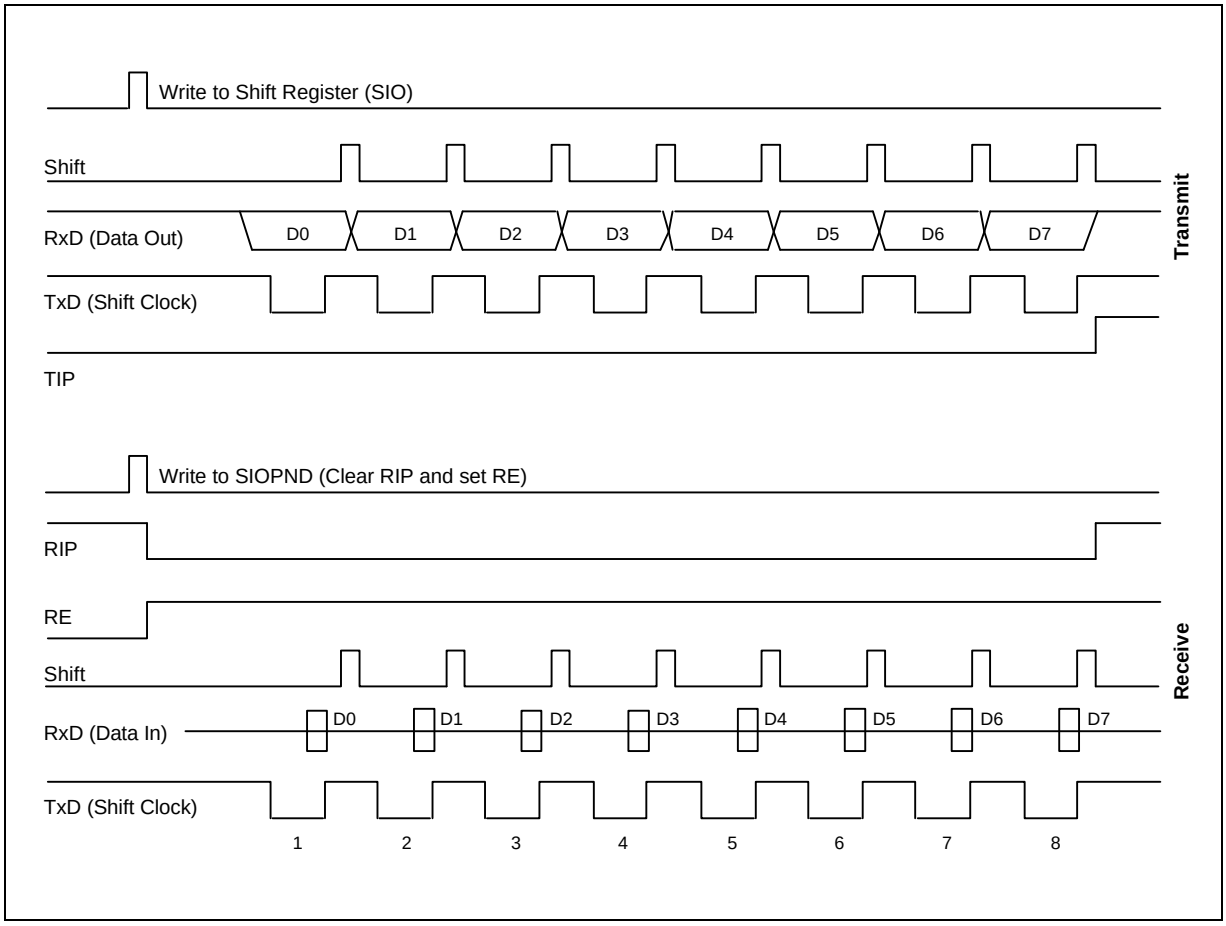


Figure 12-6. Timing Diagram for Serial Port Mode 0 Operation

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 UARTCON 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 UARTCON bits 7 and 6 to '01B'.
3. Write transmission data to the shift register UDATA (F0H, set 1). 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 UARTCON 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.

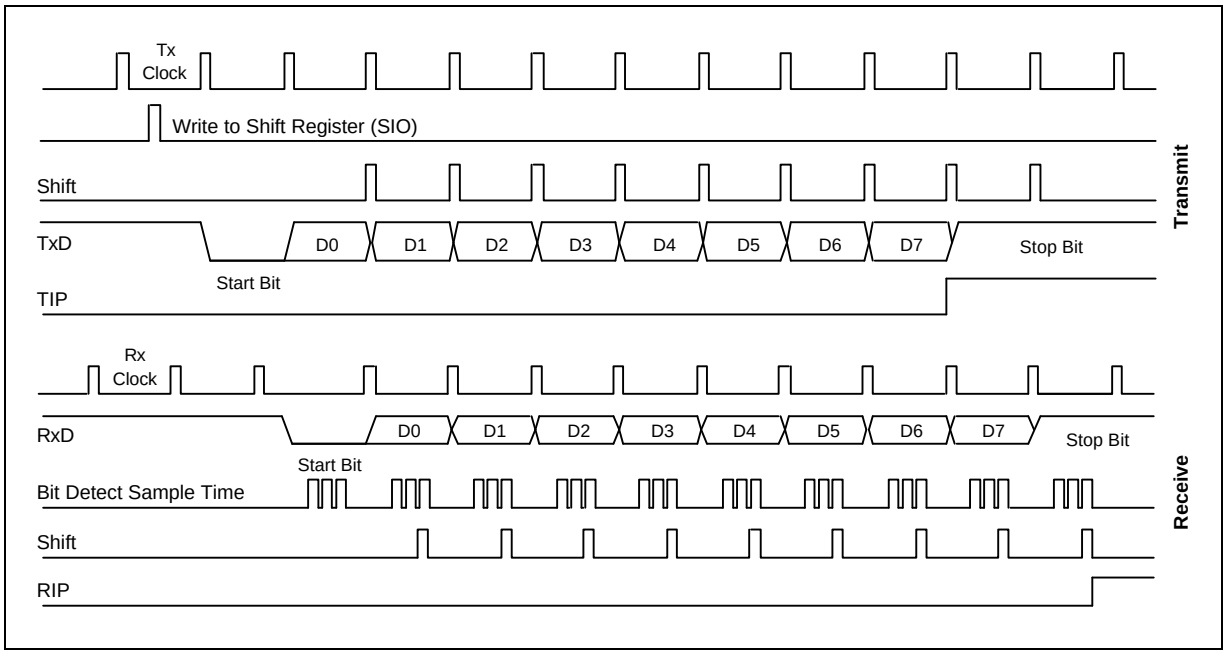


Figure 12-7. Timing Diagram for Serial Port Mode 1 Operation

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 (UARTCON.3). When receiving, the 9th data bit that is received is written to the RB8 bit (UARTCON.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 UARTCON 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, UDATA (F0H, set 1), 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 UARTCON register to "1".
3. The receive operation starts when the signal at the RxD pin goes to low level.

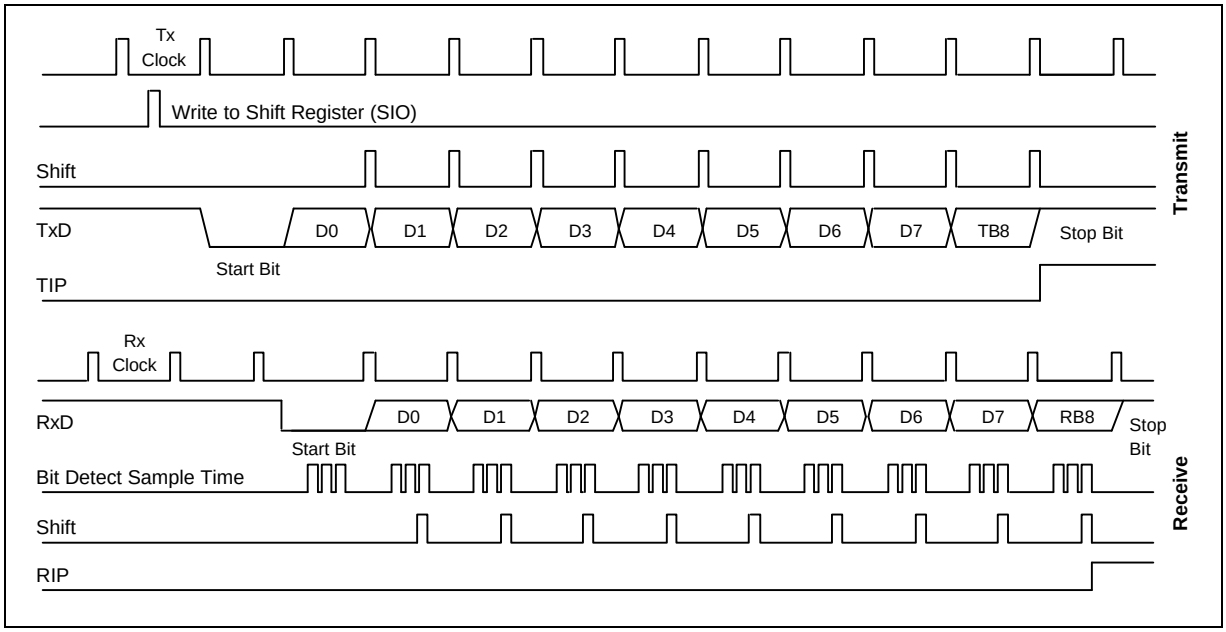


Figure 12-8. Timing Diagram for Serial Port Mode 2 Operation

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 UARTCON bits 6 and 7 to '11B'. Also, select the 9th data bit to be transmitted by writing UARTCON.3 (TB8) to "0" or "1".
3. Write transmission data to the shift register, UDATA (F0H, set 1), 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 UARTCON register to "1".
3. The receive operation will be started when the signal at the RxD pin goes to low level.

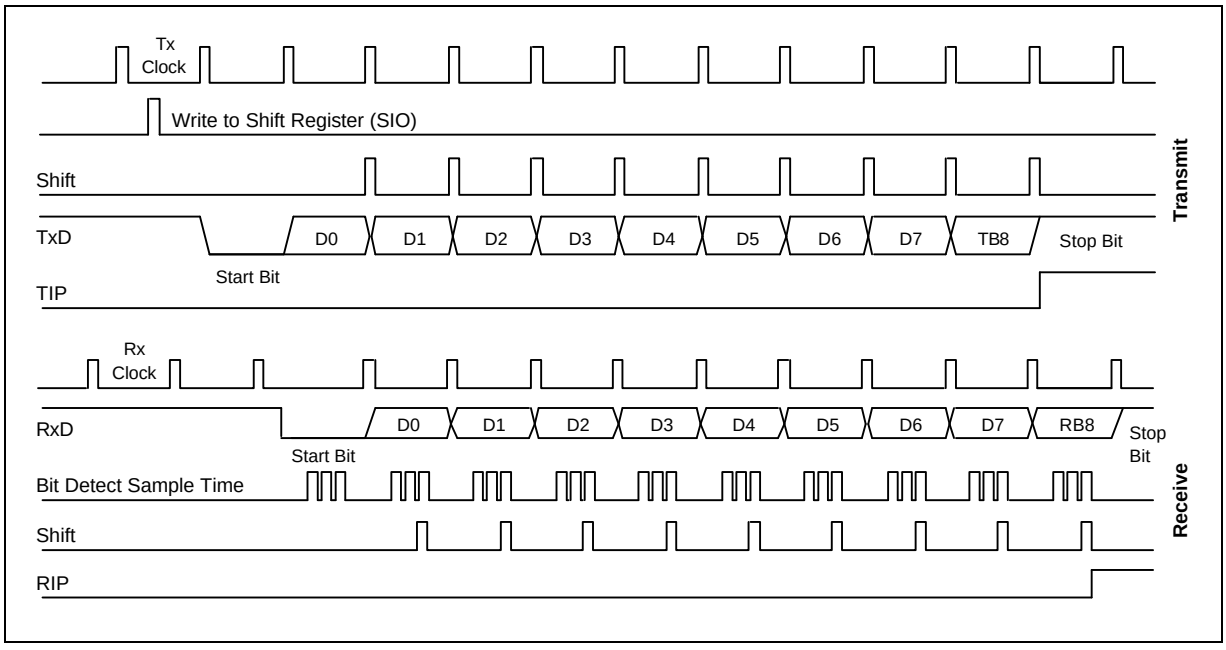


Figure 12-9. Timing Diagram for Serial Port Mode 3 Operation

BAUD RATE CALCULATIONS

Mode 0 Baud Rate Calculation

In mode 0, the baud rate is determined by the UART baud rate data register, BRDATA.

Mode 0 baud rate = $f_{OSC}/(16 \times (BRDATA + 1))$

Mode 2 Baud Rate Calculation

The baud rate in mode 2 is fixed at the f_{OSC} clock frequency divided by 16:

Mode 2 baud rate = $f_{OSC}/16$

Modes 1 and 3 Baud Rate Calculation

In modes 1 and 3, the baud rate is determined by the UART baud rate data register, BRDATA.

Mode 1 and 3 baud rate = $f_{OSC}/(16 \times (BRDATA + 1))$

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

Mode	Baud Rate	Oscillation Clock	BRDATA	
			Decimal	Hexdecimal
Mode 2	0.5 MHz	8 MHz	x	x
Mode 0	230,400 Hz	11.0592 MHz	02	02H
Mode 1	115,200 Hz	11.0592 MHz	05	05H
Mode 3	57,600 Hz	11.0592 MHz	11	0BH
	38,400 Hz	11.0592 MHz	17	11H
	19,200 Hz	11.0592 MHz	35	23H
	9,600 Hz	11.0592 MHz	71	47H
	4,800 Hz	11.0592 MHz	143	8FH
	62,500 Hz	10 MHz	09	09H
	9,615 Hz	10 MHz	64	40H
	38,461 Hz	8 MHz	12	0CH
	12,500 Hz	8 MHz	39	27H
	19,230 Hz	4 MHz	12	0CH
	9,615 Hz	4 MHz	25	19H

SERIAL COMMUNICATION FOR MULTIPROCESSOR CONFIGURATIONS

The KS88-series multiprocessor communication features lets a "master" KS88C4708/C4716 send a multiple-frame serial message to a "slave" device in a multi-KS88C4708/C4716 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 (UARTCON.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 UARTCON 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 KS88C4708/C4716 devices (masters and slaves) to UART 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.

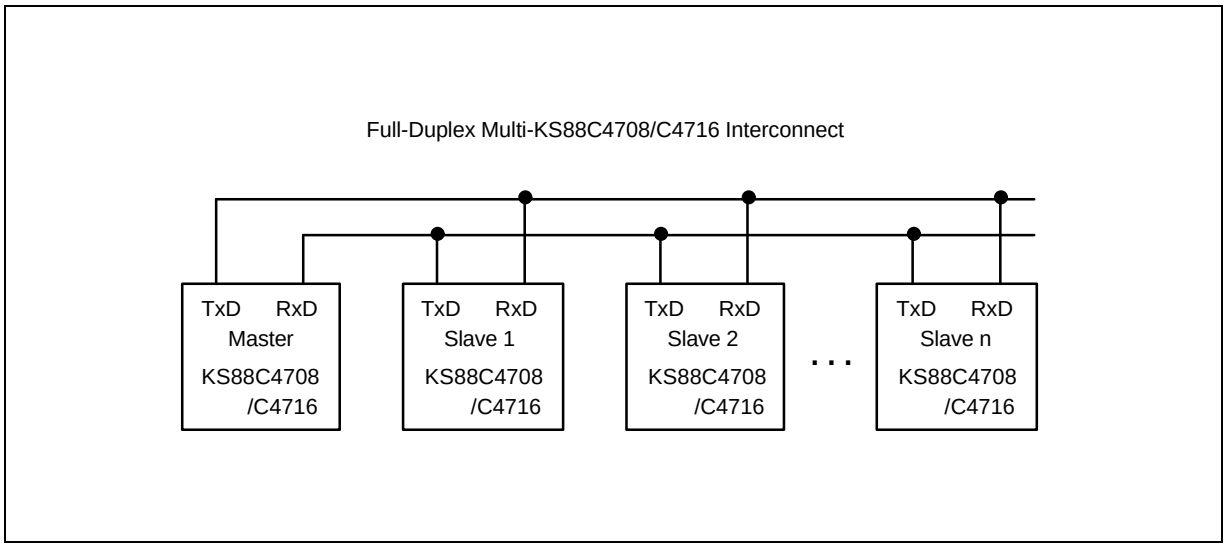


Figure 12-10. Connection Example for Multiprocessor Serial Data Communications

13 A/D CONVERTER

OVERVIEW

The 10-bit A/D converter (ADC) module uses successive approximation logic to convert analog levels entering at one of the eight input channels to equivalent 10-bit digital values. The analog input level must lie between the AV_{REF} and AV_{SS} values. The A/D converter has the following components:

— Analog comparator with successive approximation logic
— D/A converter logic (resistor string type)
— ADC control register (ADCON)
— Eight multiplexed analog data input pins (ADC0–ADC7)
— 10-bit A/D conversion data output register (ADDATAH/L)
— 8-bit digital input port (Alternately, I/O port)
— AV_{REF} and AV_{SS} pins

FUNCTION DESCRIPTION

To initiate an analog-to-digital conversion procedure, you write the channel selection data in the A/D converter control register ADCON to select one of the eight analog input pins (ADCn, n = 0-7) and set the conversion start or enable bit, ADCON.0. The read-write ADCON register is located in set 1, at address F7H.

During a normal conversion, A/DC logic initially sets the successive approximation register to 200H (the approximate half-way point of an 10-bit register). This register is then updated automatically during each conversion step. The successive approximation block performs 10-bit conversions for one input channel at a time. You can dynamically select different channels by manipulating the channel selection bit value (ADCON.6 - 4) in the ADCON register. To start the A/D conversion, you should set a the enable bit, ADCON.0. When a conversion is completed, ADCON.3, the end-of-conversion(EOC) bit is automatically set to 1 and the result is dumped into the ADDATA register where it can be read. The A/D converter then enters an idle state. Remember to read the contents of ADDATA before another conversion starts. Otherwise, the previous result will be overwritten by the next conversion result.

NOTE

Because the A/D converter has no sample-and-hold circuitry, it is very important that fluctuation in the analog level at the ADC0-ADC7 input pins during a conversion procedure be kept to an absolute minimum. Any change in the input level, perhaps due to noise, will invalidate the result. If the chip enters to STOP or IDLE mode in conversion process, there will be a leakage current path in A/D block. You must use STOP or IDLE mode after A/DC operation is finished.

CONVERSION TIMING

The A/D conversion process requires 4 steps (4 clock edges) to convert each bit and 10 clocks to set-up AD conversion. Therefore, total of 50 clocks are required to complete an 10-bit conversion: With an 1 MHz CPU clock frequency, one clock cycle is 1 us. If each bit conversion requires 4 clocks, the conversion rate is calculated as follows:

4 clocks/bit x 10 bits + set-up time = 50 clocks, 50 (clock × 1μs = 50 us at 1 MHz)

A/D CONVERTER CONTROL REGISTER (ADCON)

The A/D converter control register, ADCON, is located at address F7H in set 1. It has three functions:

— Analog input pin selection (bits 4, 5, and 6)

— End-of-conversion status detection (bit 3)

— A/D operation start or enable (bit 0)

After a reset, the ADC0 pin is automatically selected as the analog data input pin, and the start bit is turned off. You can select only one analog input channel at a time. Other analog input pins (ADC0-ADC7) can be selected dynamically by manipulating the ADCON.4-6 bits.

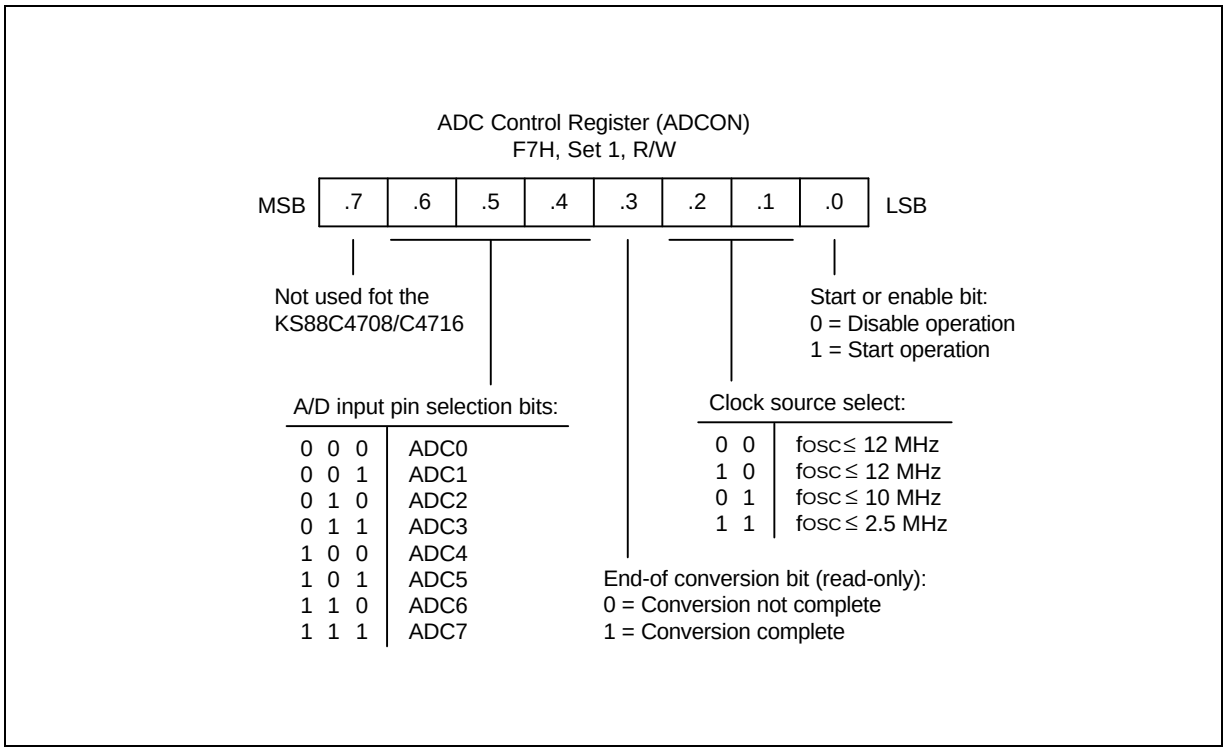


Figure 13-1. A/D Converter Control Register (ADCON)

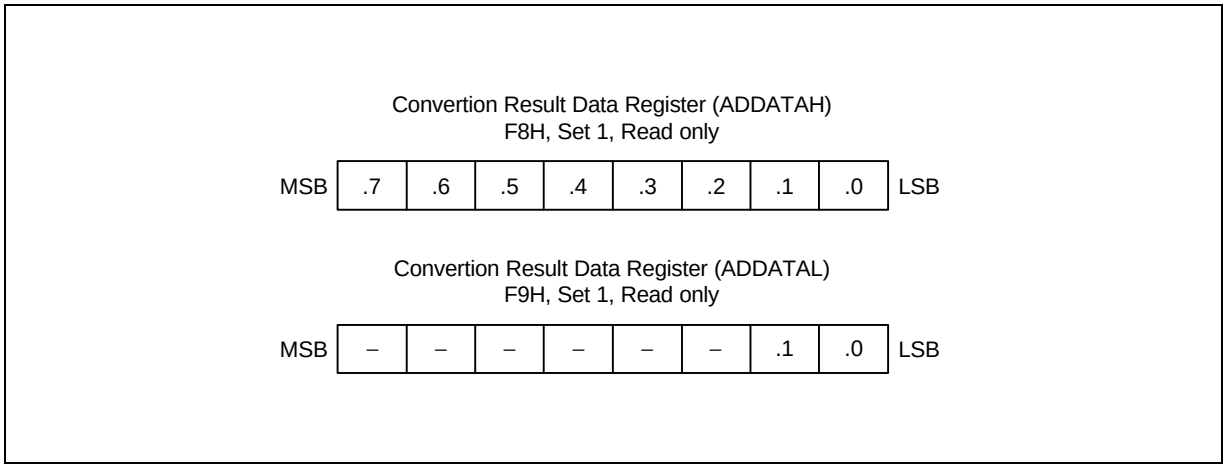


Figure 13-2. A/D Converter Data Register (ADDATAH/L)

INTERNAL REFERENCE VOLTAGE LEVELS

In the ADC function block, the analog input voltage level is compared to the reference voltage. The analog input level must remain within the range AV_{SS} to AV_{REF} (usually, $AV_{REF} = V_{DD}$).

Different reference voltage levels are generated internally along the resistor tree during the analog conversion process for each conversion step. The reference voltage level for the first conversion bit is always 1/2 AV_{REF}.

BLOCK DIAGRAM

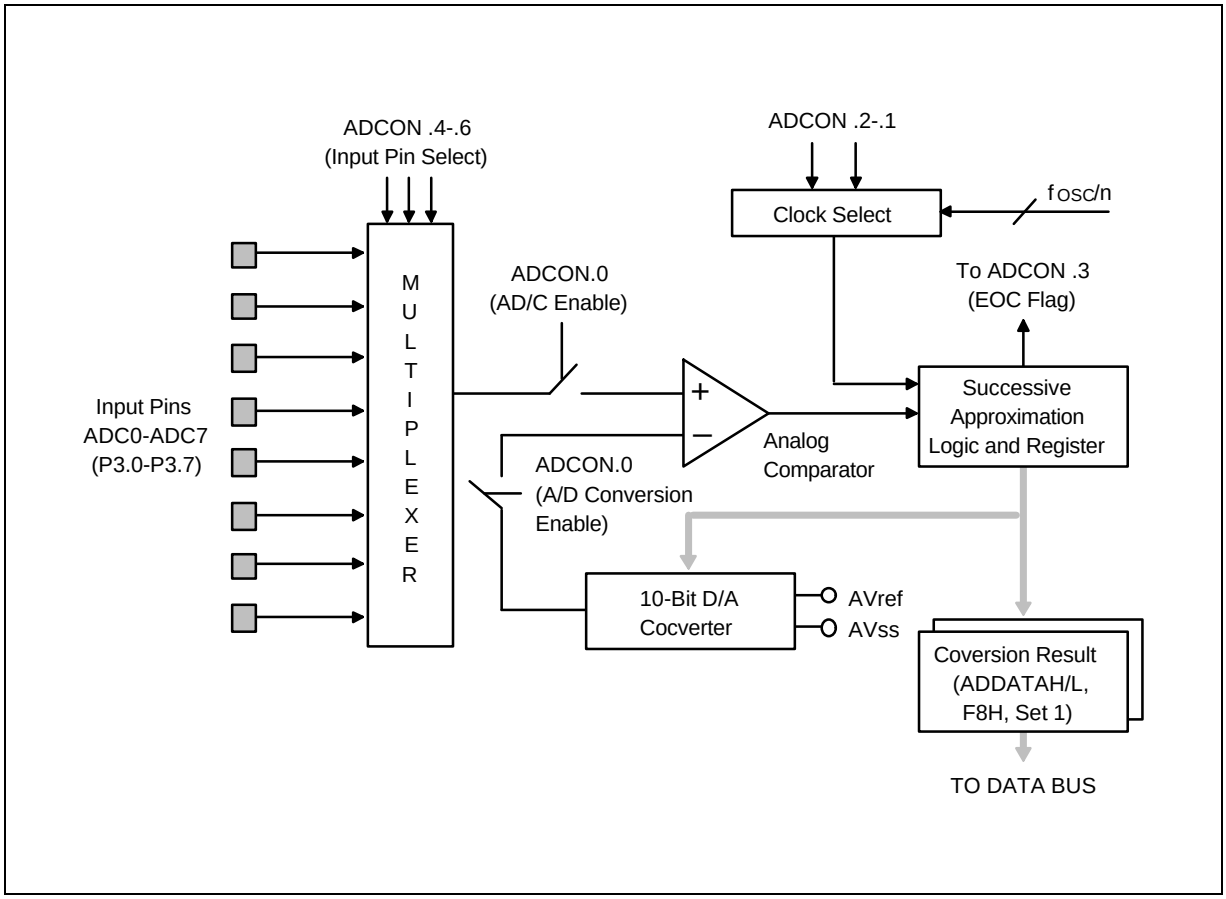


Figure 13-3. A/D Converter Functional Block Diagram

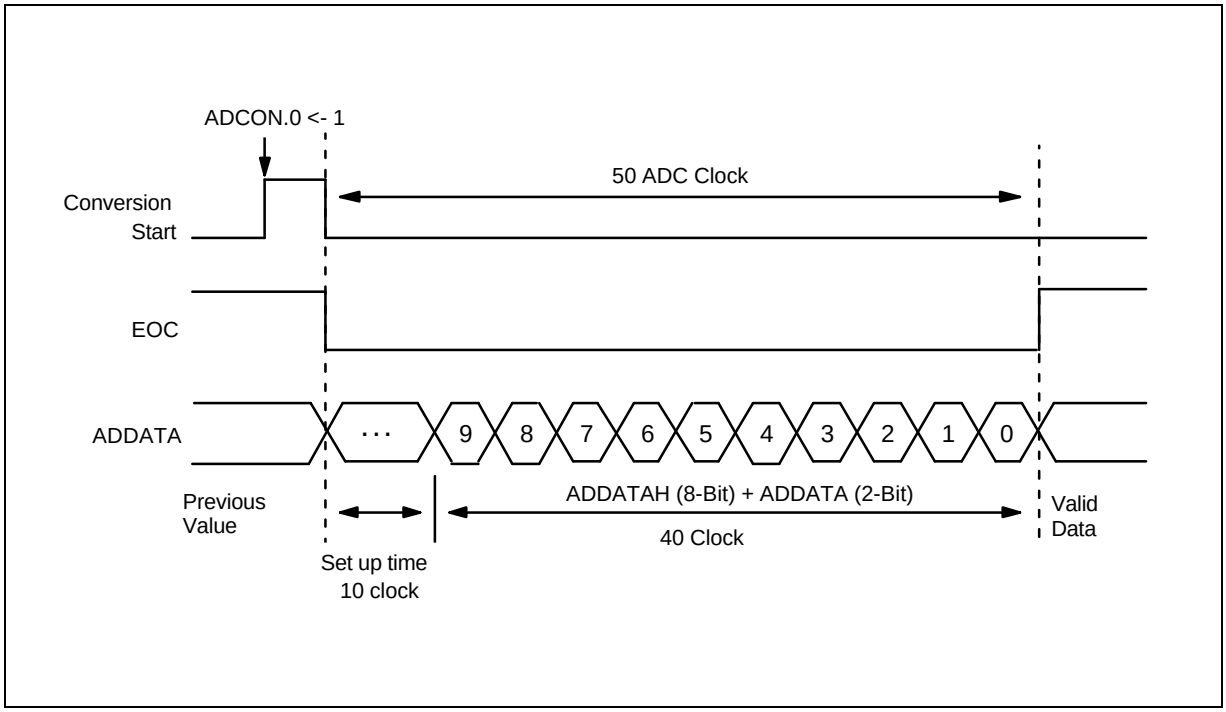


Figure 13-4. A/D Converter Timing Diagram

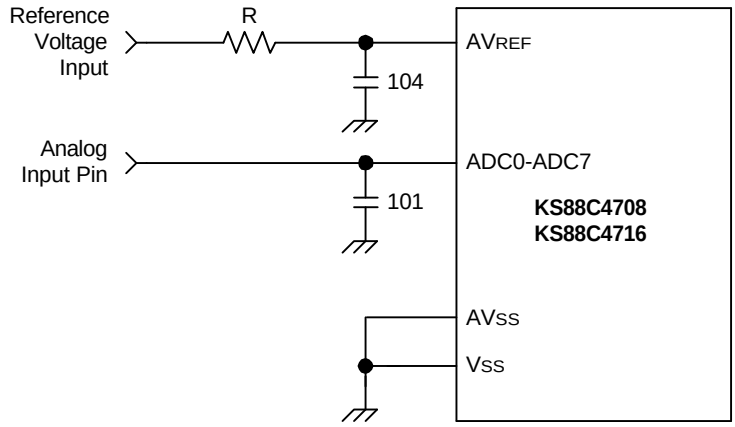
CONVERSION TIMING

The A/D conversion process requires 4 steps (4 clock edges) to convert each bit and 10 clocks to step-up A/D conversion. Therefore, total of 50 clocks are required to complete an 10-bit conversion: With an 10 MHz CPU clock frequency, one clock cycle is 400 ns (4/fosc). If each bit conversion requires 4 clocks, the conversion rate is calculated as follows:

4 clocks/bit x 10-bits + step-up time (10 clock) = 50 clocks
50 clock x 400 ns = 20 μs at 10 MHz, 1 clock time = $4/f_{OSC}$

INTERNAL A/D CONVERSION PROCEDURE

1. Analog input must remain between the voltage range of AV_{SS} and AV_{REF}.
2. Configure the analog input pins to input mode by making the appropriate settings in P5CONH and P5CONL registers.
3. Before the conversion operation starts, you must first select one of the eight input pins (ADC0–ADC7) by writing the appropriate value to the ADCON register.
4. When conversion has been completed, (50 clocks have elapsed), the EOC flag is set to “1”, so that a check can be made to verify that the conversion was successful.
5. The converted digital value is loaded to the output register, ADDATAH (8-bit) and ADDATAL (2-bit), than the ADC module enters an idle state.
6. The digital conversion result can now be read from the ADDATAH and ADDATAL register.



NOTE: The symbol "R" signifies an offset resistor with a value of from 50 to 100 Ω.

Figure 13-5. Recommended A/D Converter Circuit for Highest Absolute Accuracy

☞ PROGRAMMING TIP — Configuring A/D Converter

```
                •
                •
                •
                LD      P3CONL,#00001111B   ;  P3.1-0 A/D Input MODE
                LD      P3CONH,#11110100B   ;  P3.6-7 A/D Input MODE
                •
                •
                •

                LD      ADCON,#00000101B    ;  Channel AD0: P3.0/Conversion start
                                            ;  Conversion clock source = fOSC/4
AD0_CHK:        TM      ADCON,#00001000B    ;  A/D conversion end ? → EOC check
                JR      Z,AD0_CHK           ;  No

                LD      AD0BUFH,ADDATAH     ;  8-bit Conversion data
                LD      AD0BUFL,ADDATAL     ;  2-bit Conversion data
                •
                •
                •

                LD      ADCON,#01100101B    ;  Channel AD6: P3.6/Conversion start
                                            ;  Conversion clock source = fOSC/4
AD6_CHK:        TM      ADCON,#00001000B    ;  A/D conversion end ? → EOC check
                JR      Z,AD6_CHK           ;  No

                LD      AD6BUFH,ADDATAH     ;  8-bit Conversion data
                LD      AD6BUFL,ADDATAL     ;  2-bit Conversion data
                •
                •
                •
```

NOTES

14 ELECTRICAL DATA

OVERVIEW

In this chapter, KS88C4708/C4716 electrical characteristics are presented in tables and graphs. The information is arranged in the following order:

— Absolute maximum ratings
— Input/output capacitance
— D.C. electrical characteristics
— A.C. electrical characteristics
— Oscillation characteristics
— Oscillation stabilization time
— Data retention supply voltage in stop mode
— UART timing characteristics in mode 0
— A/D converter electrical characteristics

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	– 0.3 to V_{DD} + 10	V
Output Voltage	V_O	All output ports	– 0.3 to V_{DD} + 0.3	V
Output Current High	I_{OH}	One I/O pin active	– 18	mA
		All I/O pins active	– 60	
Output Current Low	I_{OL}	One I/O pin active	+ 30	mA
		Total pin current for ports 1, 2, and 3	+ 100	
		Total pin current for ports 0 and 4	+ 200	
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} = 1.8 V to 5.5 V)

Parameter	Symbol	Test Conditions		Min	Typ	Max	Unit
Input High Voltage	V_{IH1}	Ports 0, 1, 2, 3 ,4 and RESET	V_{DD} = 2.7 to 5.5 V	0.8 V_{DD}	–	V_{DD}	V
	V_{IH3}	X_{IN}, and X_{OUT}		V_{DD}–0.1			
Input Low Voltage	V_{IL1}	Ports 0, 1, 2, 3, 4 and RESET	V_{DD} = 2.7 to 5.5 V	–	–	0.2 V_{DD}	V
	V_{IL3}	X_{IN} and X_{OUT}				0.1	
Output High Voltage	V_{OH}	I_{OH} = – 1 mA Ports 0, 1, 2, 3, 4	V_{DD} = 4.5 to 5.5 V	V_{DD}–1.0	–	–	
Output Low Voltage	V_{OL1}	I_{OL} = 15 mA Port 0, and 4	V_{DD} = 4.5 to 5.5 V	–	0.4	2.0	V
	V_{OL2}	I_{OL} = 4 mA Ports 1, 2, and 3	V_{DD} = 4.5 to 5.5 V		0.4	2.0	V
Input High Leakage Current	I_{LIH1}	All input pins except I_{LIH2} and RESET	V_{IN} = V_{DD}	–	–	1	uA
	I_{LIH2}	X_{IN}, and X_{OUT}	V_{IN} = V_{DD}			20	
Input Low Leakage Current	I_{LIL1}	All input pins except I_{LIL2}	V_{IN} = 0 V	–	–	– 1	uA
	I_{LIL2}	X_{IN}, and X_{OUT}	V_{IN} = 0 V			– 20	
Output High Leakage Current	I_{LOH}	All output pins	V_{OUT} = V_{DD}	–	–	2	uA
Output Low Leakage Current	I_{LOL}	All output pins	V_{OUT} = 0 V	–	–	– 2	uA
Pull-up Resistor	R_{P1}	V_{IN} = 0 V, Ports 0-4	V_{DD} = 5 V	30	47	70	KΩ
	R_{P1}	RESET	V_{DD} = 5 V	100	200	350	
Supply Current	I_{DD1}	RUM mode 12 MHz CPU clock	V_{DD} = 4.5 to 5.5 V	–	10	20	mA
		3 MHz CPU clock	V_{DD} = 1.8 to 2.2 V		1.1	3	
	I_{DD2}	Idle mode 12 MHz CPU clock	V_{DD} = 4.5 to 5.5 V	–	4	8	
		3 MHz CPU clock	V_{DD} = 1.8 to 2.2 V		0.6	1.5	
	I_{DD3}	Stop mode	V_{DD} = 4.5 to 5.5 V	–	0.1	5	uA
			V_{DD} = 1.8 to 2.2 V		0.1	3	

Table 14-3. A.C. Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 4.5 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Interrupt Input High, Low Width	t_{INTH}, t_{INTL}	Ports 2 V_{DD} = 5 V ± 10 %	–	200	–	ns
RESET Input Low Width	t_{RSL}	Input V_{DD} = 5 V ± 10 %	–	1	–	µs

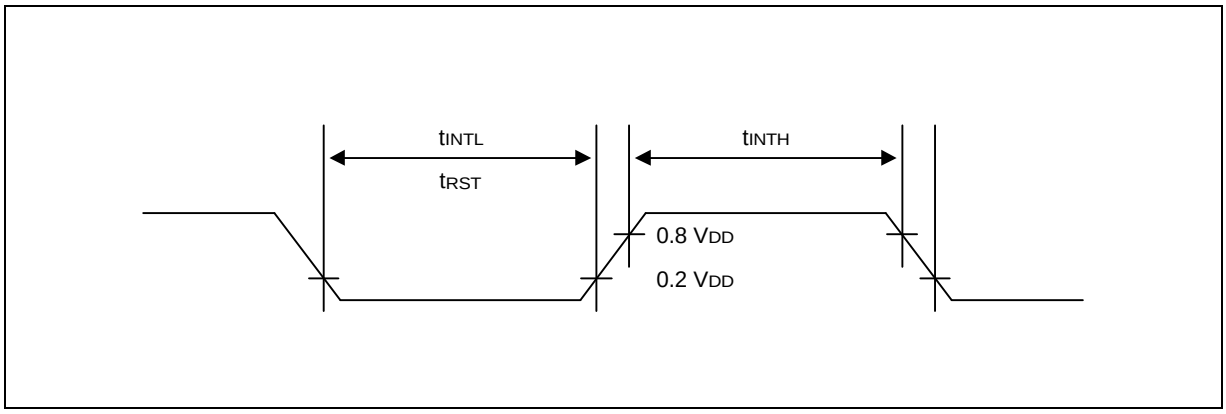
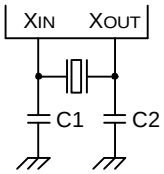
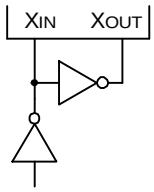


Figure 14-1. Input Timing Measurement Points

Table 14-4. Oscillation Characteristics

(T_A = – 40 °C + 85 °C)

Oscillator	Clock Circuit	Test Condition	Min	Typ	Max	Unit
Main Crystal or Ceramic		V_{DD} = 4.5 V to 5.5 V	1	–	12	MHz
		V_{DD} = 2.7 V to 4.5 V			8	
		V_{DD} = 1.8 V to 2.7 V			3	
External Clock (Main System)		V_{DD} = 4.5 V to 5.5 V	1	–	12	MHz
		V_{DD} = 2.7 V to 4.5 V			8	
		V_{DD} = 1.8 V to 2.7 V			3	

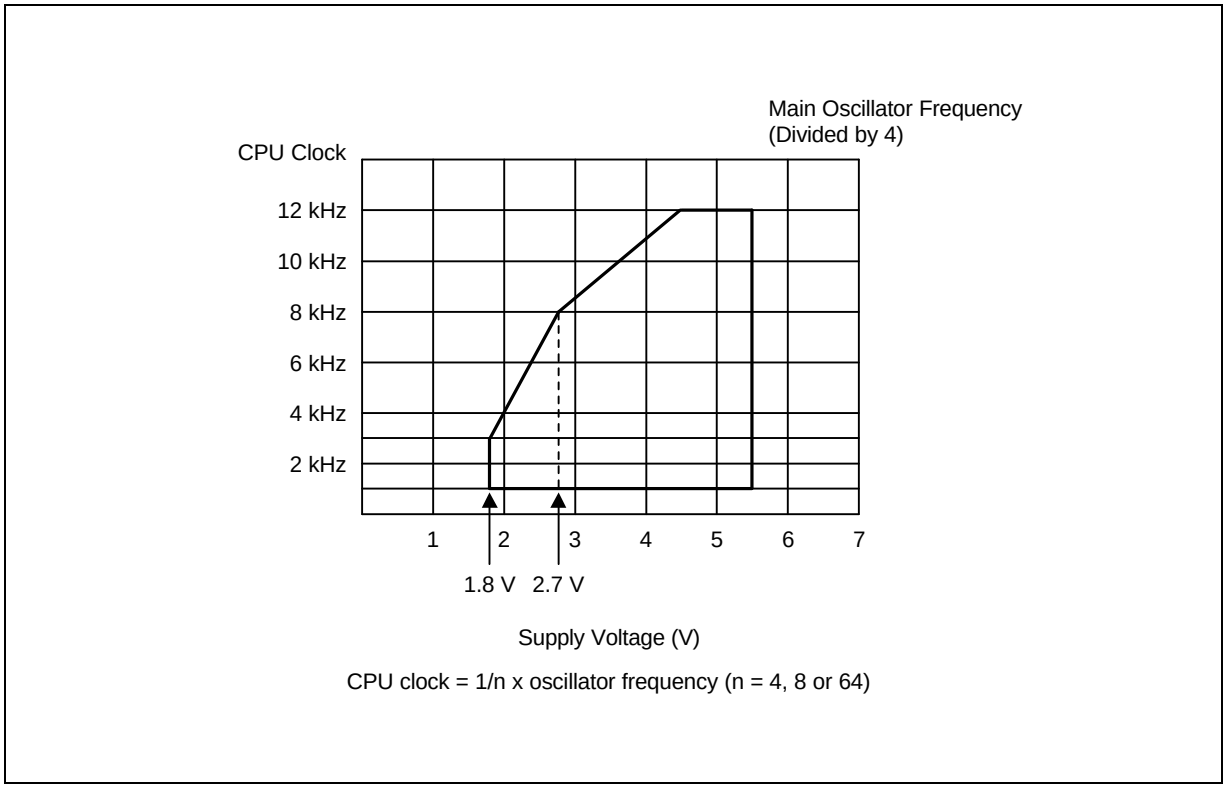


Figure 14-2. Operating Voltage Range

Table 14-5. Oscillation Stabilization Time

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

Oscillator	Test Condition	Min	Typ	Max	Unit
Main Crystal	f_{OSC} > 1.0 kHz; Oscillation stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	20	ms
Main Ceramic		–	–	10	ms
External Clock (Main System)	X_{IN} input High and Low width (t_{XH}, t_{XL})	25	–	500	ns
Oscillator Stabilization Wait Time	t_{WAIT} when released by a reset (1)	–	$2^{16}/f_{OSC}$	–	ms
	t_{WAIT} when released by an interrupt (2)	–	–	–	ms

NOTES:
1. f_{OSC} is the oscillator frequency.
2. The duration of the oscillator stabilization wait time, t_{WAIT}, when it is released by an interrupt is determined by the settings in the basic timer control register, BTCON.

Table 14-6. UART Timing Characteristics in Mode 0 (10 MHz)

(T_A = – 40°C to + 85°C, V_{DD} = 1.8 V to 5.5 V, Load capacitance = 80 pF)

Parameter	Symbol	Min	Typ	Max	Unit
Serial port clock cycle time	t_{SCK}	500	$t_{CPU} \times 6$	700	ns
Output data setup to clock rising edge	t_{S1}	300	$t_{CPU} \times 5$	–	
Clock rising edge to input data valid	t_{S2}	–	–	300	
Output data hold after clock rising edge	t_{H1}	t_{CPU} – 50	t_{CPU}	–	
Input data hold after clock rising edge	t_{H2}	0	–	–	
Serial port clock High, Low level width	t_{HIGH}, t_{LOW}	200	$t_{CPU} \times 3$	400	

NOTES:
1. All timings are in nanoseconds (ns) and assume a 10-MHz CPU clock frequency.
2. The unit t_{CPU} means one CPU clock period.

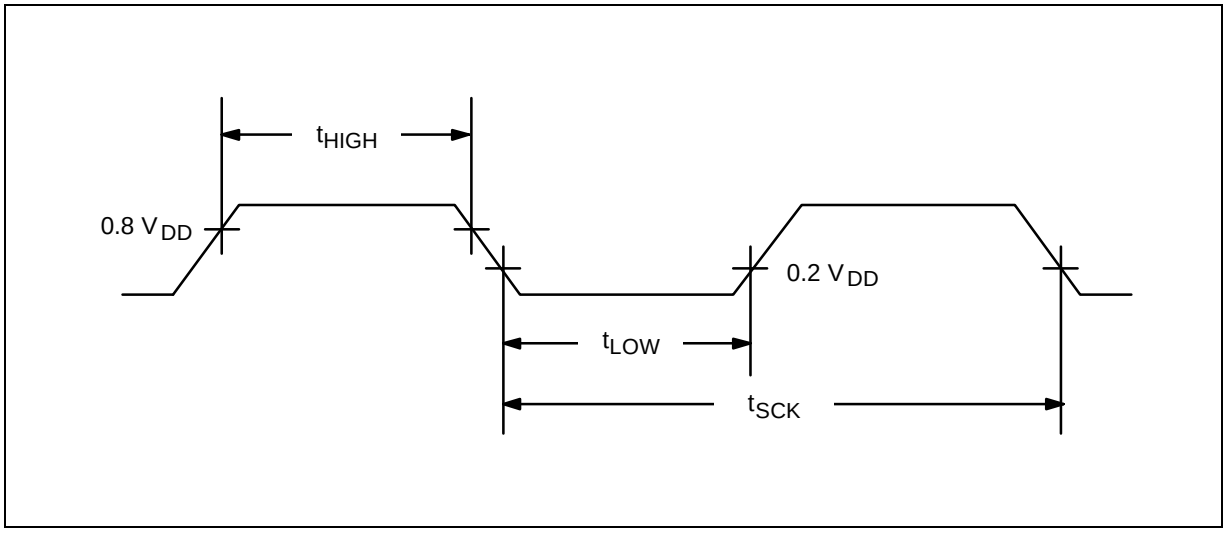
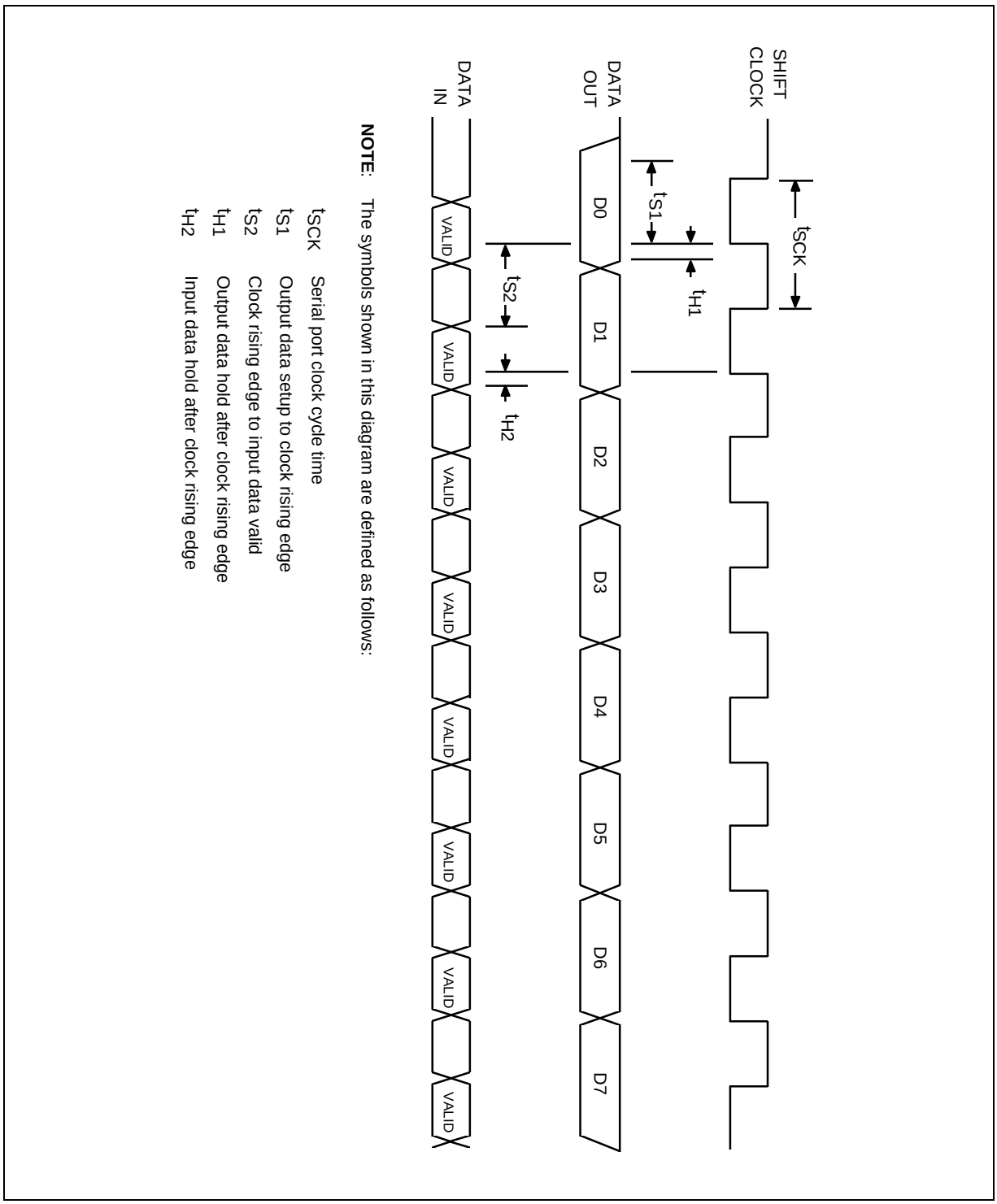


Figure 14-3. Waveform for UART Timing Characteristics



NOTE: The symbols shown in this diagram are defined as follows:

t_{SCK}	Serial port clock cycle time
t_{S1}	Output data setup to clock rising edge
t_{S2}	Clock rising edge to input data valid
t_{H1}	Output data hold after clock rising edge
t_{H2}	Input data hold after clock rising edge

Figure 14-4. A.C. Timing Waveform for the UART Module

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

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

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data Retention Supply Voltage	V_{DDDR}	Stop mode	1.8	–	5.5	V
Data Retention Supply Current	I_{DDDR}	Stop mode, V_{DDDR} = 1.8 V	–	0.1	5	µA

NOTE: Supply current does not include current drawn through internal pull-up resistors or external output current loads.

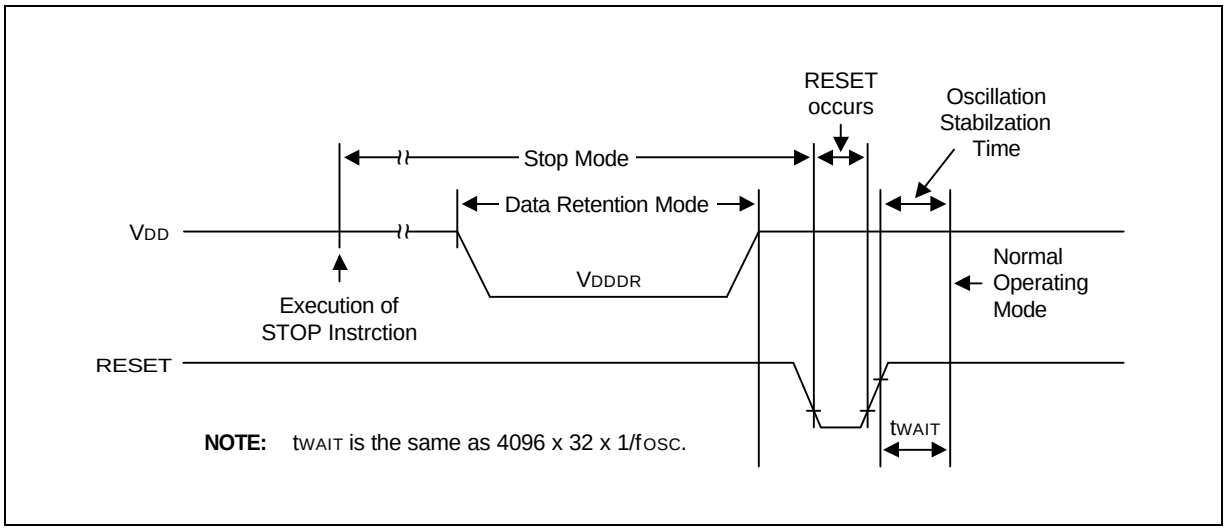


Figure 14-5. Stop Mode Release Timing When Initiated by a Reset

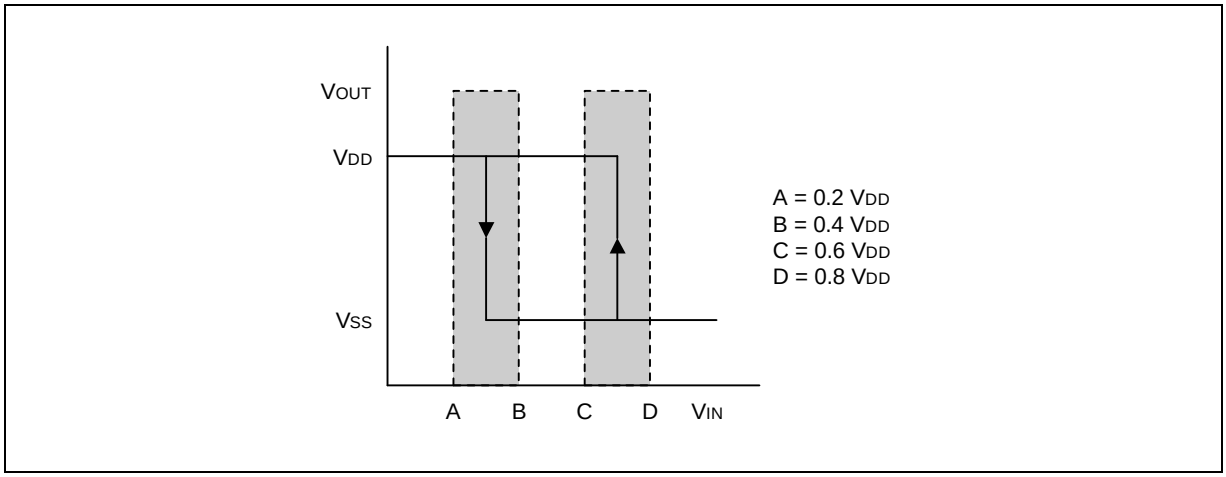


Figure 14-6. Schmitt Trigger Input Characteristics

Table 14-8. A/D Converter Electrical Characteristics

(T_A = – 40°C to + 85°C, V_{DD} = 2.7 V to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Unit
Total accuracy	–	V_{DD} = 5.12 V	–	–	± 3	LSB
Integral linearity error	ILE	CPU clock = 8 MHz AV_{REF} = 5.12 V		–	± 2	
Differential linearity error	DLE	AV_{SS} = 0 V		–	± 1	
Offset error of top	EOT			± 1	± 3	
Offset error of bottom	EOB			± 1	± 2	
Conversion time (1)	t_{CON}	f_{OSC} = 10 MHz (3)	20	–	–	µs
Analog input voltage	V_{IAN}	–	AV_{SS}	–	AV_{REF}	V
Analog input impedance	R_{AN}	–	2	–	–	MΩ
ADC reference voltage	AV_{REF}	–	2.5	–	V_{DD}	V
ADC reference ground	AV_{SS}	–	V_{SS}	–	V_{SS} + 0.3	V
Analog input current	I_{ADIN}	AV_{CC} = V_{CC} = 5 V	–	–	10	µA
Analog block current (2)	I_{ADC}	AV_{CC} = V_{CC} = 5 V	–	1	3	mA
		AV_{CC} = V_{CC} = 3 V		0.5	1.5	
		AV_{CC} = V_{CC} = 5 V power down mode		100	500	nA

NOTES:

1. "Conversion time" is the time required from the moment a conversion operation starts until it ends.
2. I_{ADC} is operating current during A/D conversion.
3. f_{OSC} is the main oscillator clock.

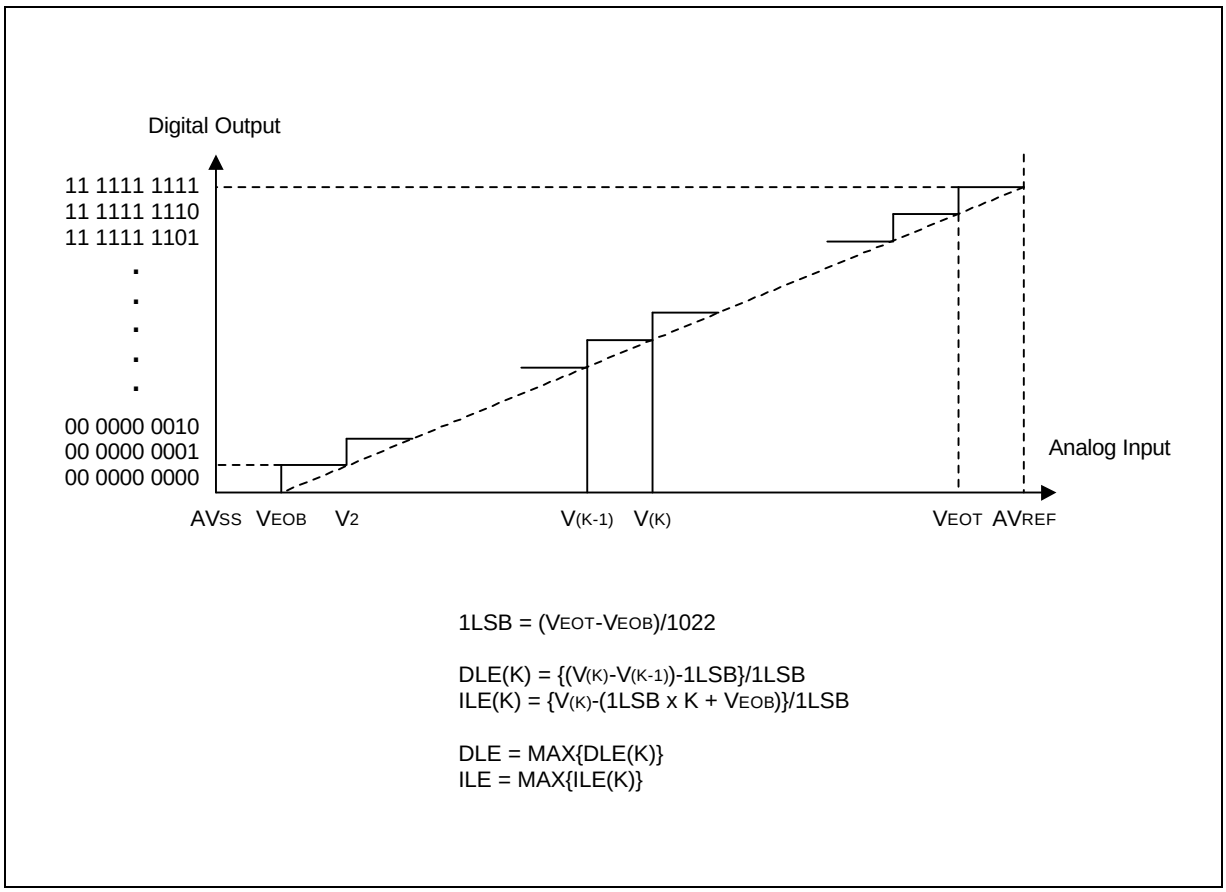


Figure 14-7. Definition of DLE and ILE

NOTES

15 MECHANICAL DATA

OVERVIEW

This section contains the following information about the device package:

— Package dimensions in millimeters

— Pad diagram

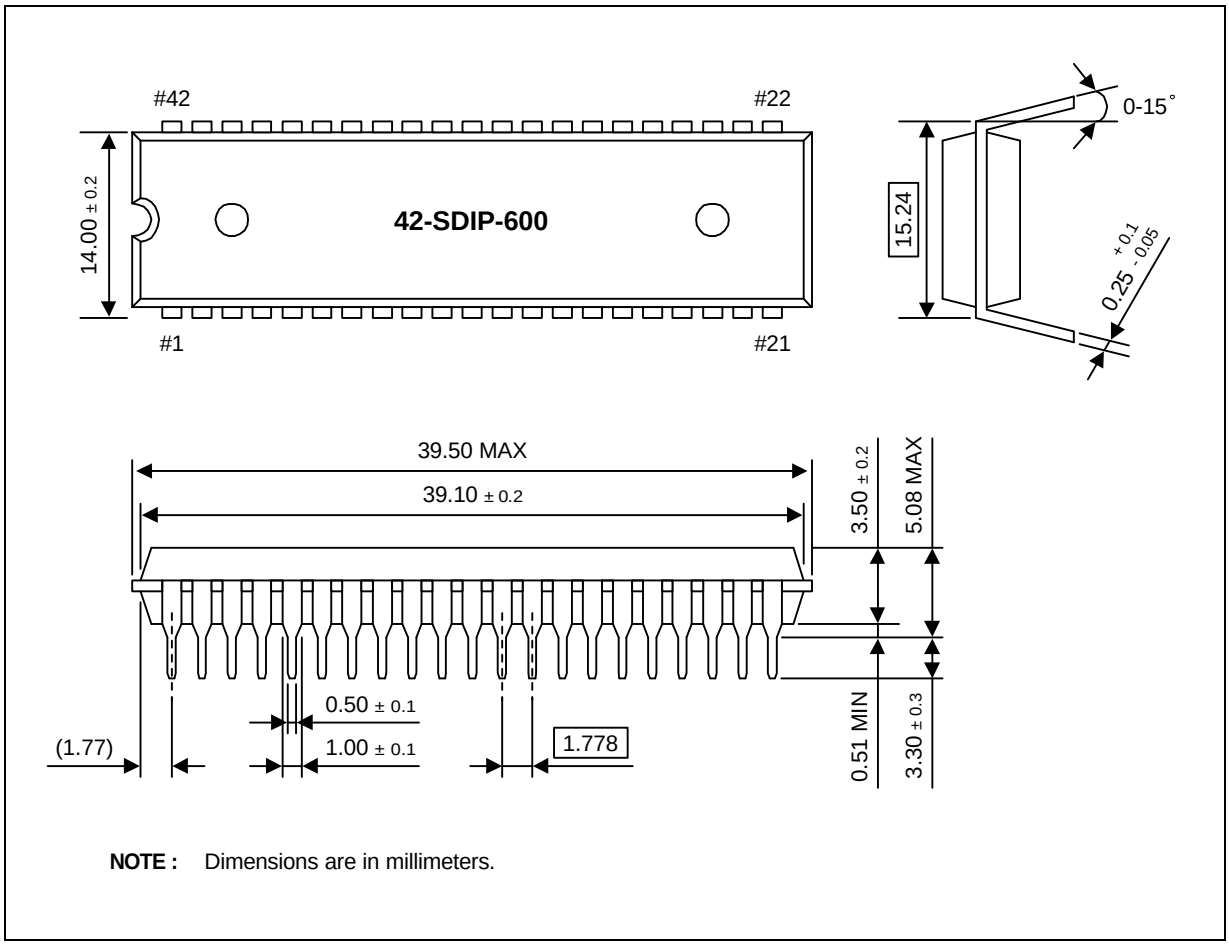
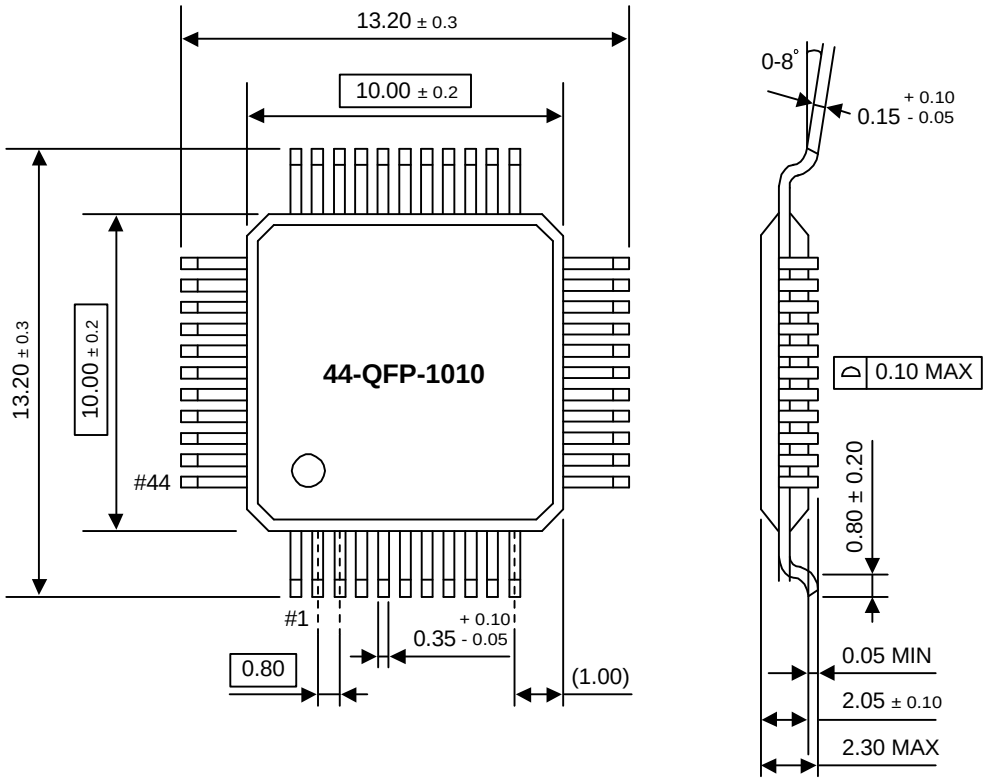


Figure 15-1. 42-SDIP-600 Package Dimensions



NOTE : Dimensions are in millimeters.

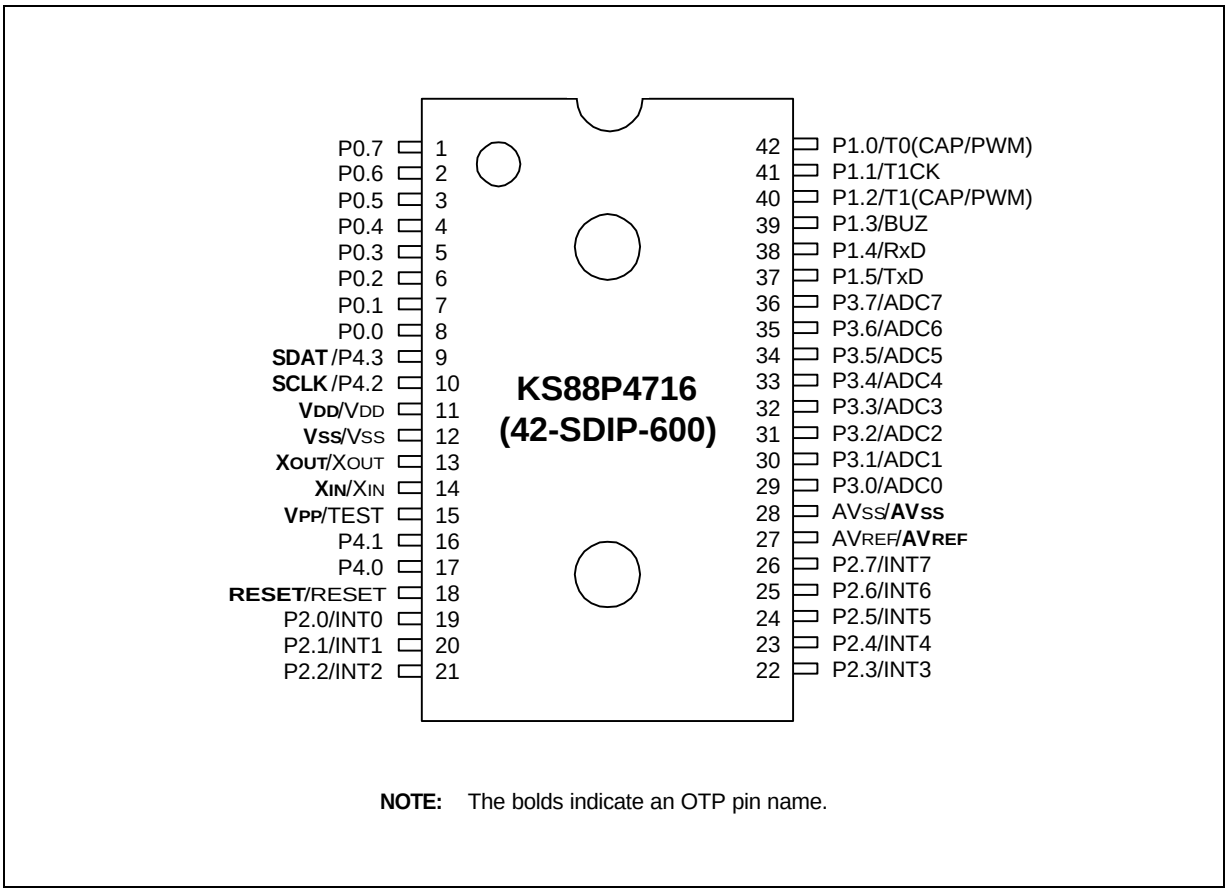
Figure 15-2. 44-QFP-1010 Package Dimensions

16 KS88P4716 OTP

OVERVIEW

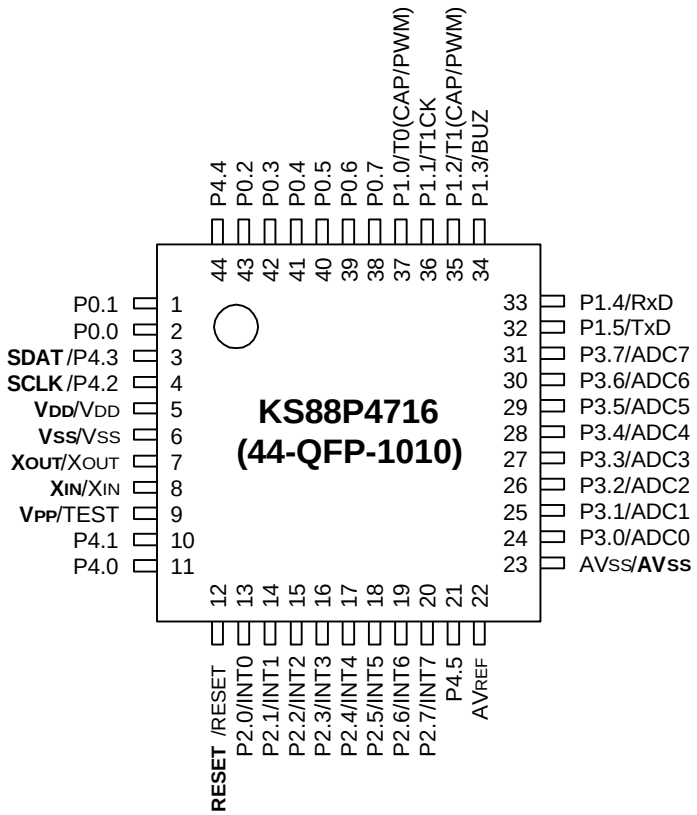
The KS88P4716 single-chip CMOS microcontroller is the OTP (One Time Programmable) version of the KS88C4708/C4716 microcontroller. It has an on-chip OTP ROM instead of a masked ROM. The EPROM is accessed by serial data format.

The KS88P4716 is fully compatible with the KS88C4708/C4716, both in function in D.C. electrical characteristics and in pin configuration. Because of its simple programming requirements, the KS88P4716 is ideal as an evaluation chip for the KS88C4708/C4716.



NOTE: The bolds indicate an OTP pin name.

Figure 16-1. KS88P4716 Pin Assignments (42-SDIP Package)



NOTE: The bolds indicate an OTP pin name.

Figure 16-2. KS88P4716 Pin Assignments (44-QFP Package)

Table 16-1. Descriptions of Pins Used to Read/Write the EPROM

Main Chip	During Programming			
Pin Name	Pin Name	Pin No.	I/O	Function
P4.3	SDAT	9(3)	I/O	Serial data pin. Output port when reading and input port when writing. Can be assigned as a Input/push-pull output port.
P4.2	SCLK	10(4)	I	Serial clock pin. Input only pin.
TEST	V_{PP}	14(16)	I	Power supply pin for EPROM cell writing (indicates that OTP enters into the writing mode). When 12.5 V is applied, OTP is in writing mode and when 5 V is aplied, OTP is in reading mode. (Option)
RESET	RESET	18(12)	I	Chip Initialization
V_{DD}/V_{SS}	V_{DD}/V_{SS}	11(5)/12(6)	–	Logic power supply pin. V_{DD} should be tied to +5 V during programming.

NOTE: ( ) means 44 QFP package.

Table 16-2. Comparison of KS88P4716 and KS88C4708/C4716 Features

Characteristic	KS88P4716	KS88C4708/C4716
Program Memory	16-Kbyte EPROM	8/16-Kbyte mask ROM
Operating Voltage (V_{DD})	1.8 V to 5.5 V	1.8 V to 5.5 V
OTP Programming Mode	V_{DD} = 5 V, V_{PP} (EA) = 12.5 V	
Pin Configuration	42 SDIP/44 QFP	42 SDIP/44 QFP
EPROM Programmability	User Program 1 time	Programmed at the factory

OPERATING MODE CHARACTERISTICS

When 12.5 V is supplied to the V_{PP} (TEST) pin of the KS88P4716, 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 16-3 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.

NOTES

17 DEVELOPMENT TOOLS

OVERVIEW

Samsung provides a powerful and easy-to-use development support system on a turnkey basis. The development support system is composed of a host system, debugging tools, and supporting software. For a host system, any standard computer that employs MS-DOS as its operating system can be used. A sophisticated debugging tool is provided both in hardware and software: the powerful in-circuit emulator, SMDS2+, for the KS57, KS86, and KS88 microcontroller families. SMDS2+ is a newly improved version of SMDS2. Samsung also offers supporting software that includes, 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 of use. Each window can be easily sized, moved, scrolled, highlighted, added, or removed.

SAMA ASSEMBLER

The Samsung Arrangeable Microcontroller (SAM) Assembler, SAMA, is a universal assembler, and generating an object code in the standard hexadecimal format. Assembled program codes include the object code 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

The SASM88 is an relocatable assembler for Samsung's KS88-series microcontrollers. The SASM88 takes a source file containing assembly language statements and translates them into a corresponding source code, an object code and comments. The SASM88 supports macros and conditional assembly. It runs on the MS-DOS operating system. As it produces relocatable object codes only, the user should link object files. Object files can be linked with other object files and loaded into memory.

HEX2ROM

HEX2ROM file generates a ROM code from a HEX file which is produced by the assembler. A ROM code is needed to fabricate a microcontroller which has a mask ROM. When generating a 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 the KS88-series microcontrollers. All the required target system cables and adapters are included on the device-specific target board.

OTPS

One time programmable microcontrollers (OTP) for the KS88C4708/C4716 and OTP programmer (Gang) are now available.

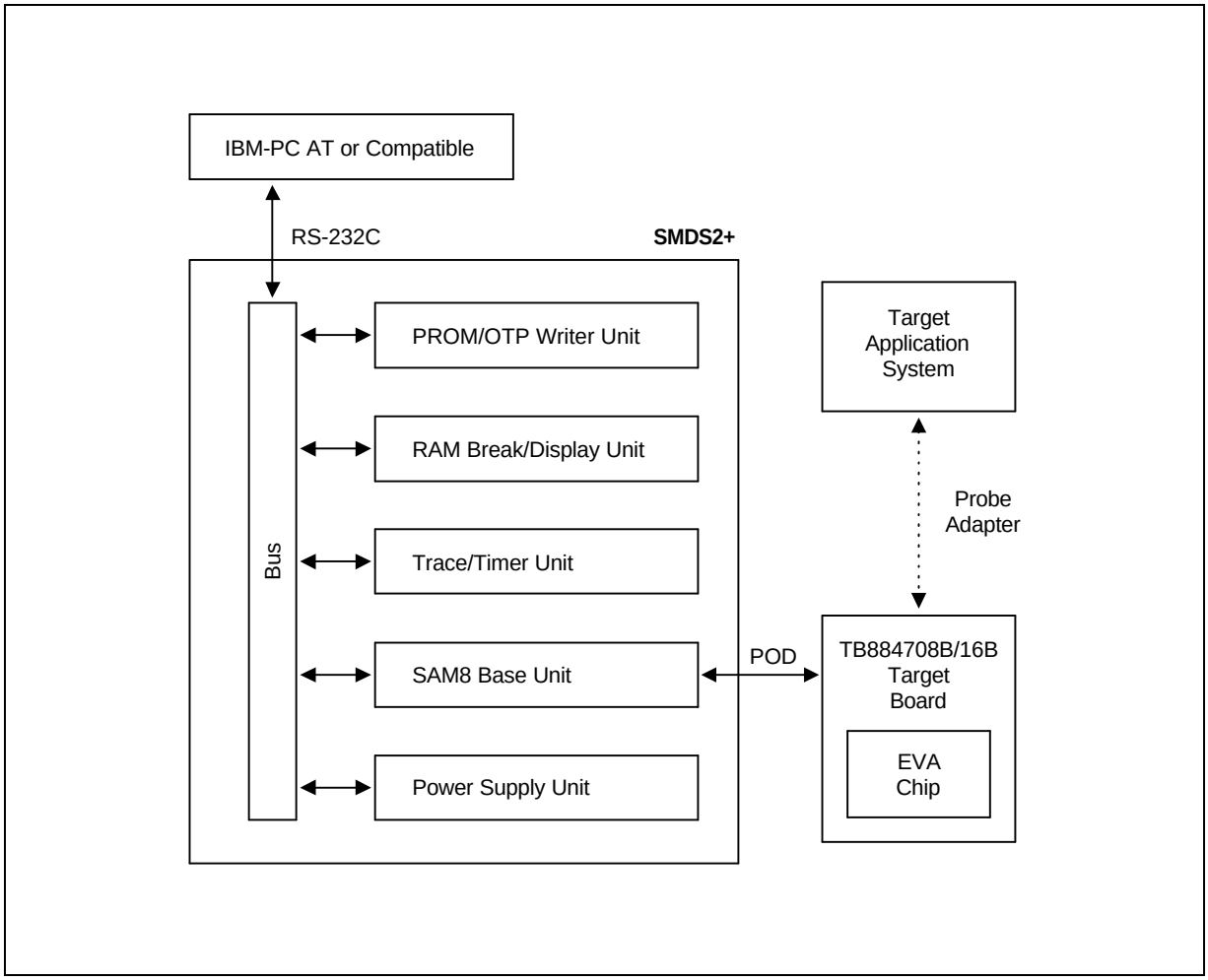


Figure 17-1. SMDS Product Configuration (SMDS2+)

TB884708B/16B TARGET BOARD

The TB884708B/16B target board is used for the KS88C4708/C4716 and the KS88P4716 microcontroller. It is supported by the SMDS2+ development system.

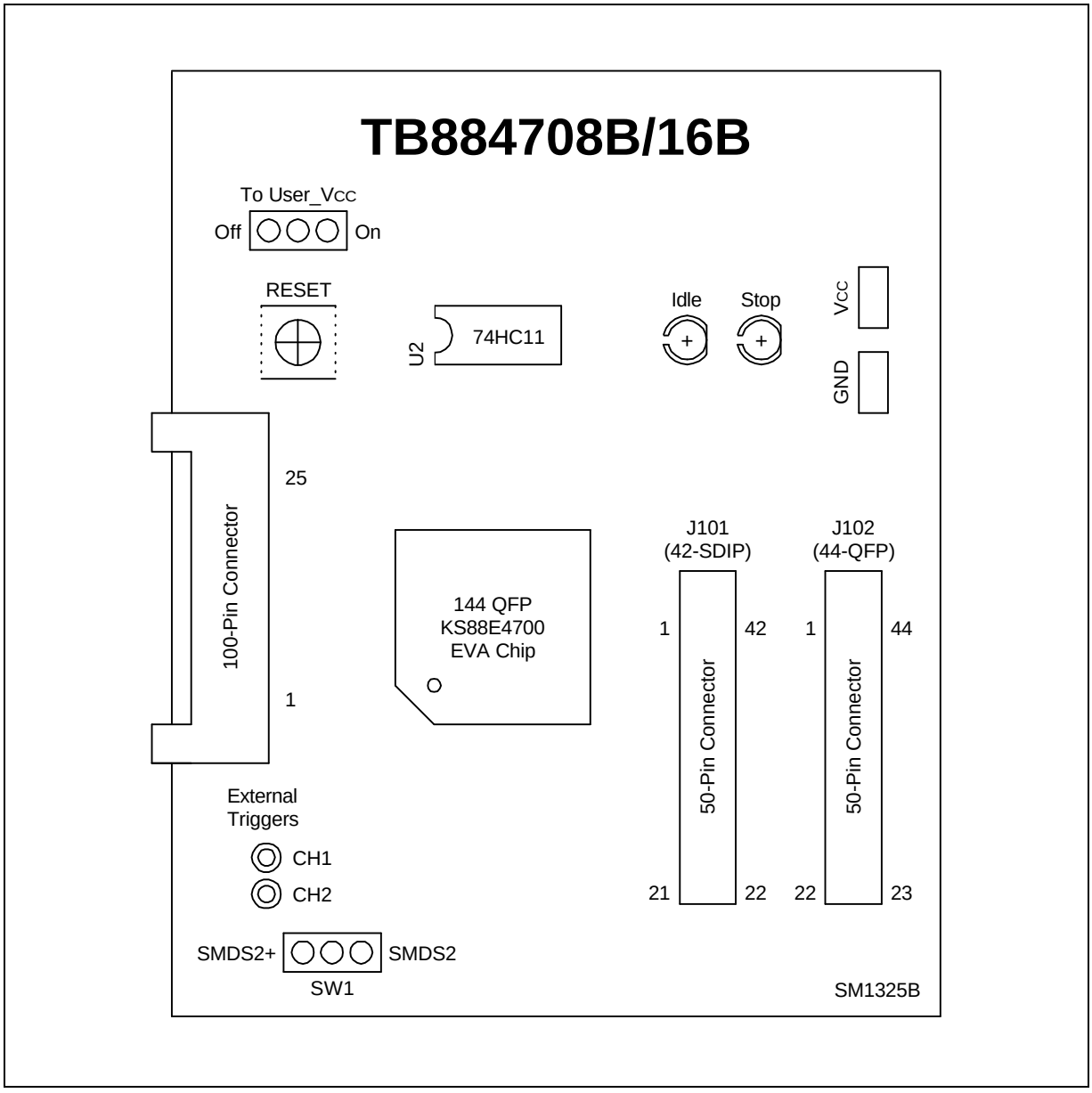


Figure 17-2. TB884708A/16A Target Board Configuration

Table 17-1. Power Selection Settings for TB884708B/16B

'To User_Vcc' Settings	Operating Mode	Comments
To User_Vcc Off On	TB884708B/16B Vcc Vss Target System Vcc SMDS2/SMDS2+	SMDS2/SMDS2+ supplies V_{CC} to the target board (evaluation chip) and the target system.
To User_Vcc Off On	TB884708B/16B External Vcc Vss Target System Vcc SMDS2+	SMDS2/SMDS2+ supplies V_{CC} only to the target board (evaluation chip). The target system must have a power supply of its own.

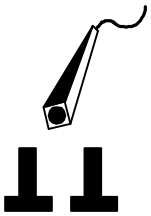
SMDS2+ Selection (SAM8)

In order to write data into program memory 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+	R/W* R/W* Target Board 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 (KS88E4700) is in idle mode.

STOP LED

This LED is ON when the evaluation chip (KS88E4700) is in stop mode.

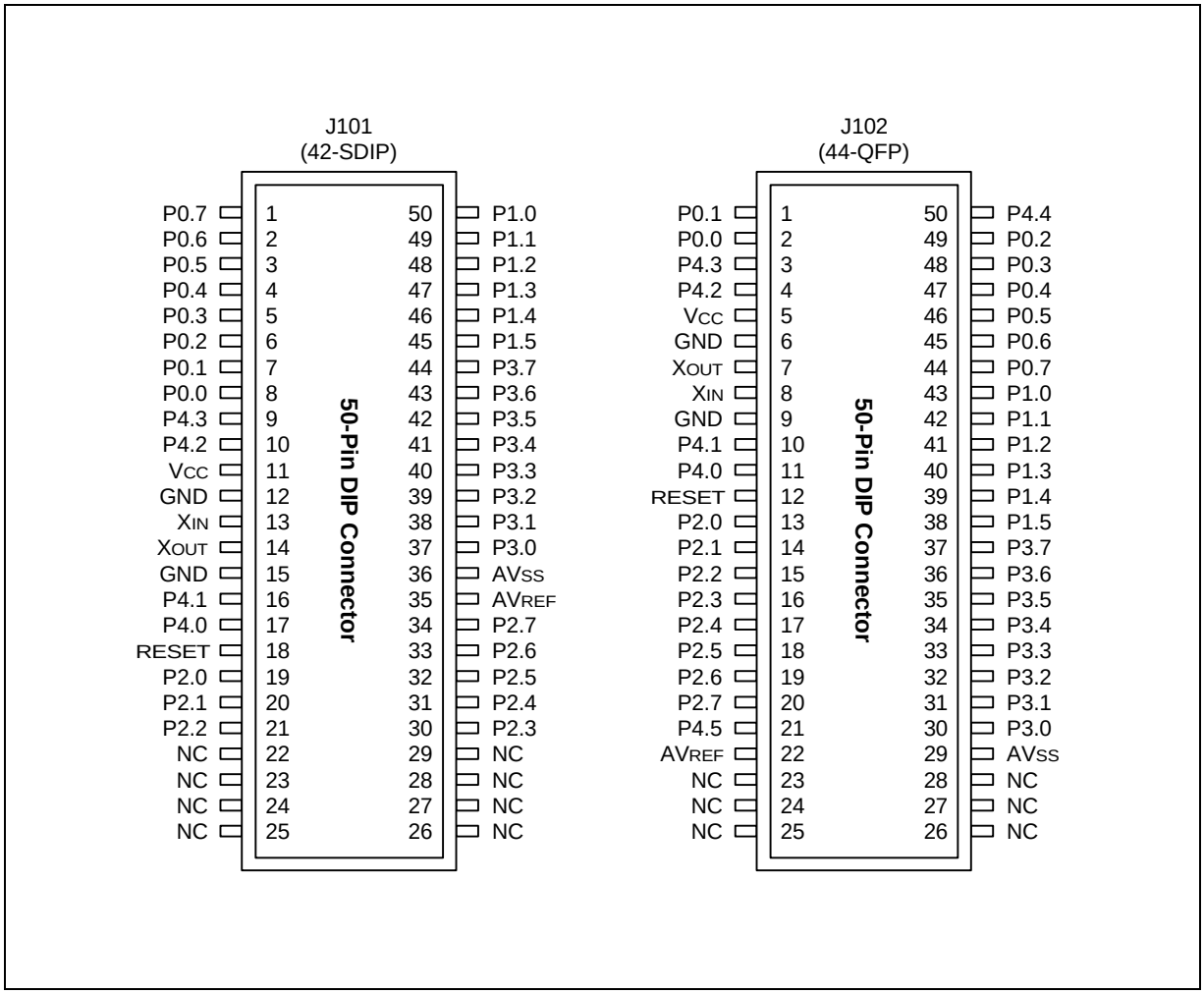


Figure 17-3. 40-Pin Connector for TB884708A/16A

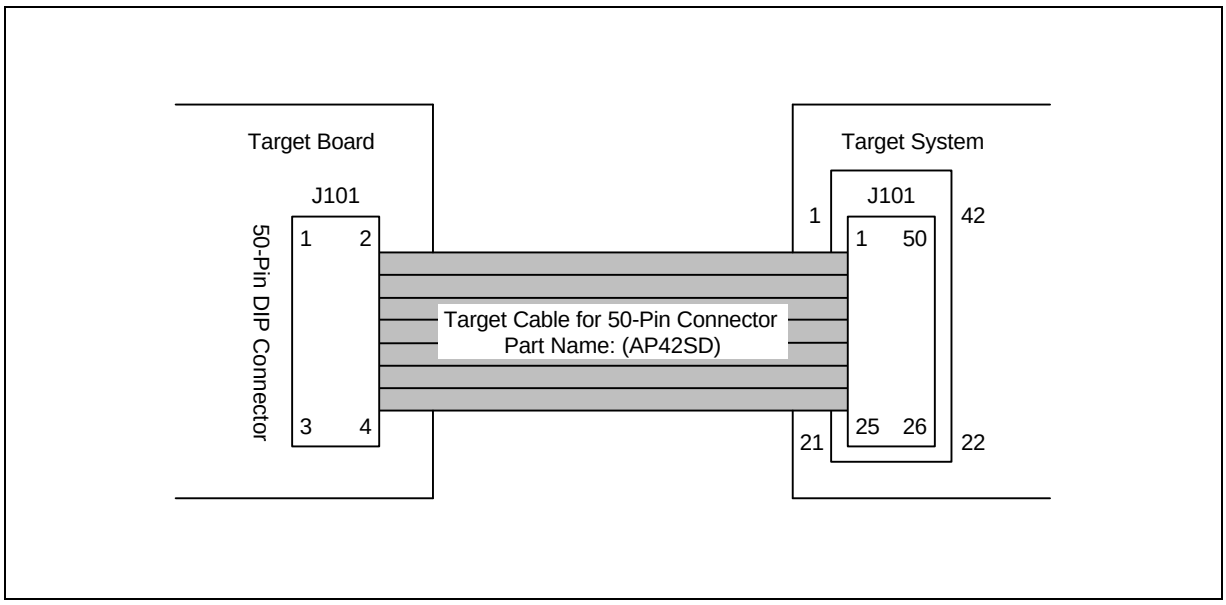


Figure 17-4. TB884708A/16A Adapter Cable for 64-SDIP Package

NOTES