1 PRODUCT OVERVIEW

OVERVIEW

Samsung's KS88 series of 8-bit single-chip CMOS microcontrollers offers a fast and efficient CPU, a wide range of integrated peripherals, and various mask-programmable ROM sizes. Important CPU features include:

— Efficient register-oriented architecture

— Selectable CPU clock sources

— Idle and Stop power-down mode release by interrupt

— Built-in basic timer with watchdog function

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

KS88C01016/C01008/C01004/C01116/C01108/C01104 MICROCONTROLLER

The KS88C01016/C01008/C01004/C01116/C01108/C01104 single-chip CMOS microcontroller is fabricated using a highly advanced CMOS process and is based on Samsung's newest CPU architecture.

The KS88C01016/C01008/C01004/C01116/C01108/C01104 is the microcontroller which has mask-programmable ROM.

The KS88P01016/P01008/P01004/P01116/P01108/P01104 is the microcontroller which has one-time-programmable EPROM.

Using a proven modular design approach, Samsung engineers developed the KS88C01016/C01008/C01004/C01116/C01108/C01104 by integrating the following peripheral modules with the powerful SAM87 RC core:

— Three programmable I/O ports, including two 8-bit ports and one 3-bit port, for a total of 19 pins.

— Internal LVD circuit and eight bit-programmable pins for external interrupts.

— One 8-bit basic timer for oscillation stabilization and watchdog functions (system reset).

— One 8-bit timer/counter and one 16-bit timer/counter with selectable operating modes.

— One 8-bit counter with auto-reload function and one-shot or repeat control.

The KS88C01016/C01008/C01004/C01116/C01108/C01104 is a versatile general-purpose microcontroller which is especially suitable for use as remote transmitter controller. It is currently available in a 24-pin SOP and SDIP package

FEATURES

CPU

- SAM87RC CPU core

Memory

- Program memory (ROM)
  - KS88C01016/C01116: 15,872-byte (0000H-3E00H)
  - KS88C01008/C01108: 8-Kbyte (0000H-1FFFH)
  - KS88C01004/C01104: 4-Kbyte (0000H-0FFFH)
- Data memory: 256-byte RAM

Instruction Set

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

Instruction Execution Time

- 500 ns at 8-MHz f_{OSC} (minimum)

Interrupts

- 13 interrupt sources with 10 vector.
- 5 level, 10 vector interrupt structure

I/O Ports

- Two 8-bit I/O ports (P0-P1) and one 3-bit port (P2) for a total of 19 bit-programmable pins
- Eight input pins for external interrupts

Carrier Frequency Generator

- One 8-bit counter with auto-reload function and one-shot or repeat control (Counter A)

Back-up mode

- When V_{DD} is lower than V_{LVD}, the chip enters Back-up mode to block oscillation and reduce the current consumption.

Timers and Timer/Counters

- One programmable 8-bit basic timer (BT) for oscillation stabilization control or watchdog timer function
- One 8-bit timer/counter (Timer 0) with two operating modes; Interval mode and PWM mode.
- One 16-bit timer/counter with one operating modes; Interval mode

Low Voltage Detect Circuit

- Low voltage detect for reset or Back-up mode.
- Low level detect voltage
  – **KS88C01016/C01008/C01004** :
  2.20V (Typ) ± 200 mV
  – **KS88C01116/C01108/C01104**:
  1.90V (Typ) ± 200 mV

Auto Reset Function

- Reset occurs when stop mode is released by P0.
- When a falling edge is detected at Port 0 during Stop mode, system reset occurs.

Operating Temperature Range

- –40°C to + 85°C

Operating Voltage Range

- 1.7 V to 3.6 V at 4 MHz f_{OSC}
- 2.0 V to 3.6 V at 8 MHz f_{OSC}

Package Type

- 24-pin SOP/SDIP

BLOCK DIAGRAM

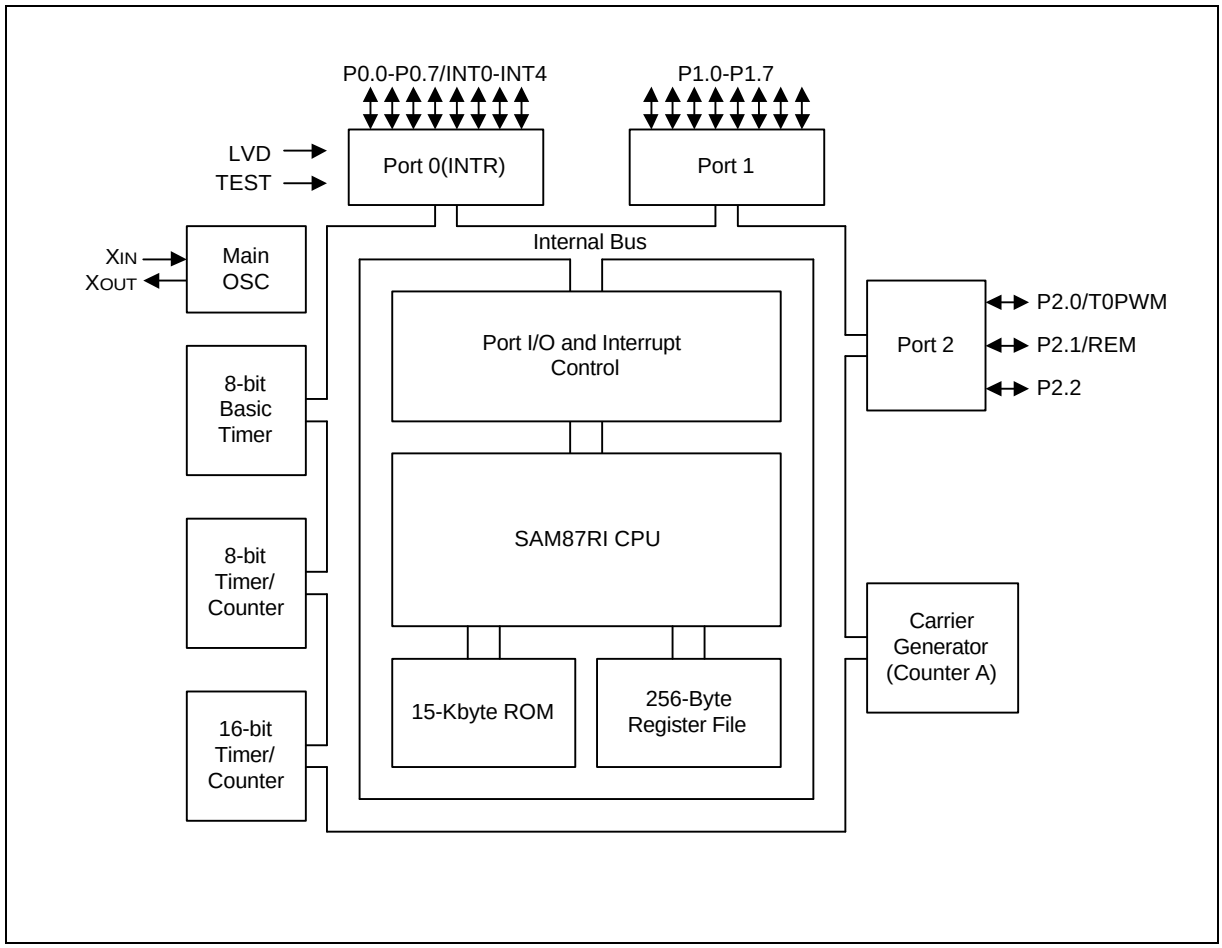


Figure 1-1. Block Diagram

PIN ASSIGNMENTS

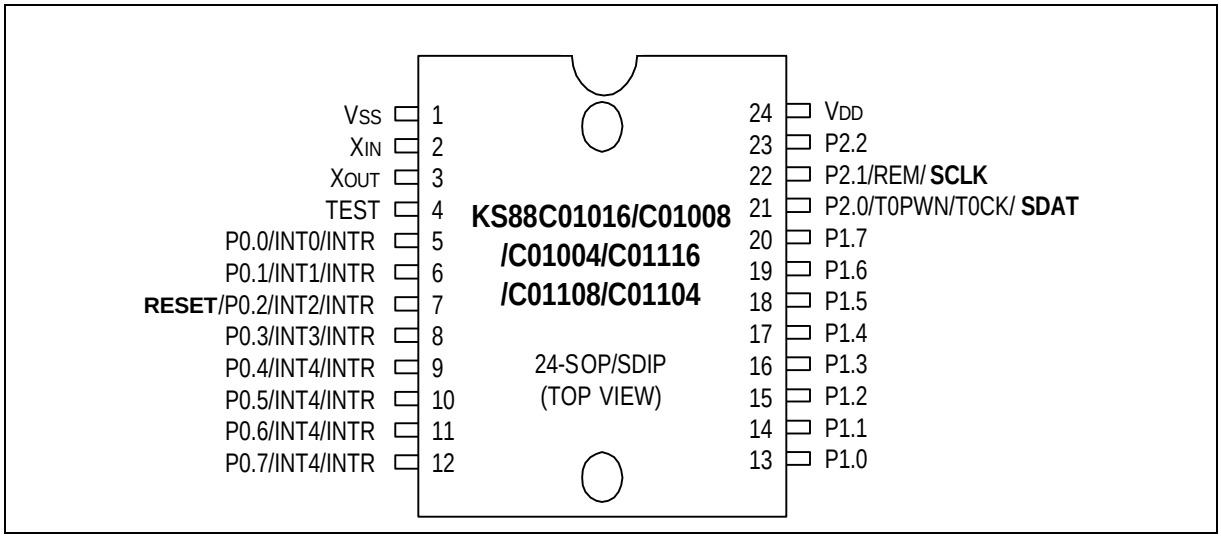


Figure 1-2. Pin Assignment Diagram (24-Pin SOP/SDIP Package)

PIN DESCRIPTIONS

Table 1-1. Pin Descriptions

Pin Names	Pin Type	Pin Description	Circuit Type	24- Pin Number	Shared Functions
P0.0-P0.7	I/O	I/O port with bit-programmable pins. Configurable to input or push-pull output mode. Pull-up resistors are assignable by software. Pins can be assigned individually as external interrupt inputs with noise filters, interrupt enable/ disable, and interrupt pending control. Interrupt with Reset(INTR) is assigned to Port 0.	1	5-12	INT0 – INT4/INTR
P1.0-P1.7	I/O	I/O port with bit-programmable pins. Configurable to input mode or output mode. Pin circuits are either push-pull or n-channel open-drain type. Pull-up resistors are assignable by software.	2	13-20	
P2.0 P2.1 P2.2	I/O	3-bit I/O port with bit-programmable pins. Configurable to input mode, push-pull output mode, or n-channel open-drain output mode. Input mode with pull-up resistors are assignable by software. The two pins of port2 have high current drive capability.	3 4 5	21-23	REM/T0CK
X_{IN}, X_{OUT}	–	System clock input and output pins	–	2,3	–
TEST	I	Test signal input pin (for factory use only; must be connected to V_{SS}).	–	4	–
V_{DD}	–	Power supply input pin	–	24	–
V_{SS}	–	Ground pin	–	1	–

PIN CIRCUITS

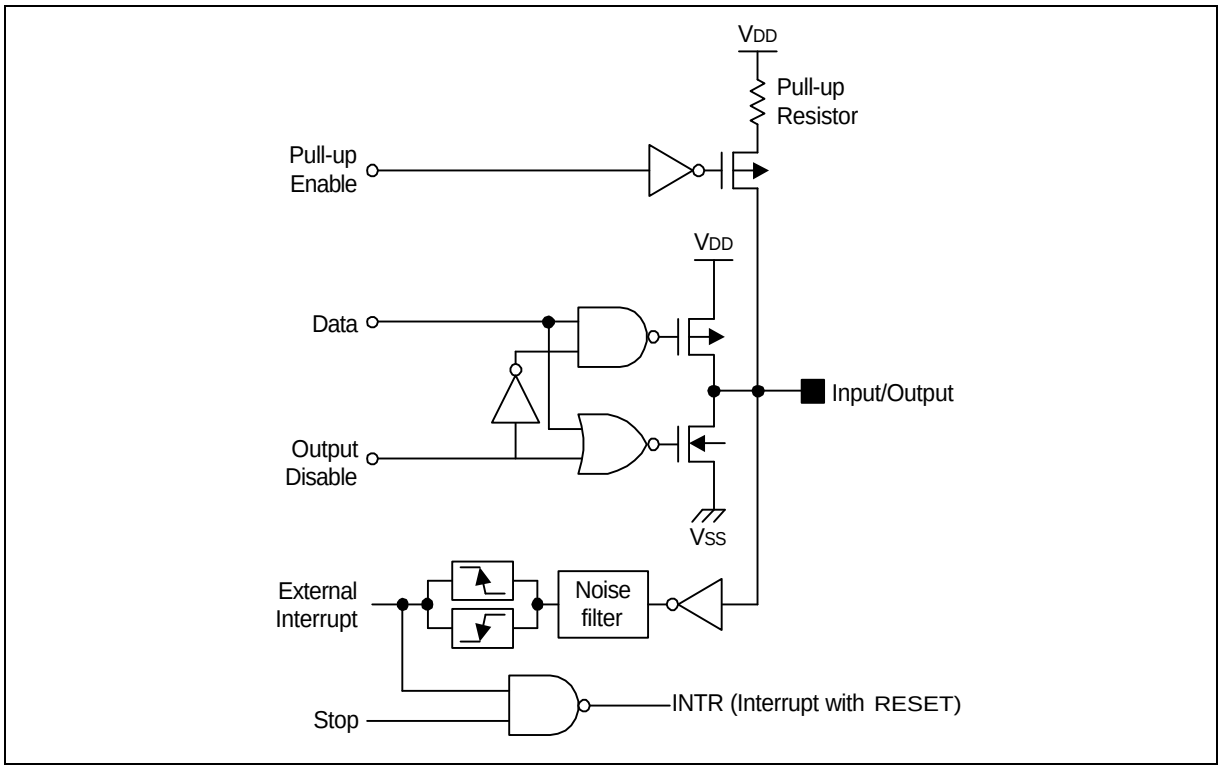


Figure 1-3. Pin Circuit Type 1 (Port 0)

NOTE

Interrupt with reset (INTR) is assigned to port 0 of KS88C01016/C01008/C01004/C01116/C01108 /C01104. It is designed to release stop status with reset. When the falling/rising edge is detected at any pin of Port 0 during stop status, non vectored interrupt INTR signal occurs, after then system reset occurs automatically. It is designed for a application which are using “stop mode” like remote controller. If stop mode is not used, INTR do not operates and it can be discarded.

PIN CIRCUITS (Continued)

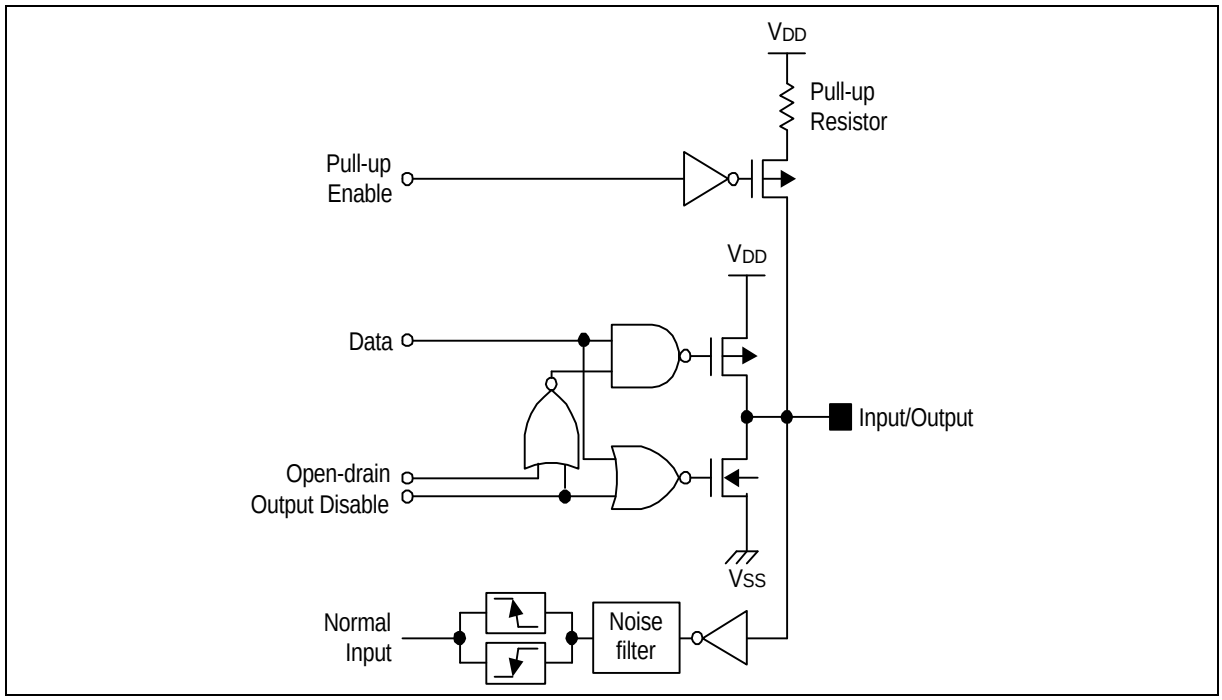


Figure 1-4. Pin Circuit Type 2 (Port 1)

PIN CIRCUITS (Continued)

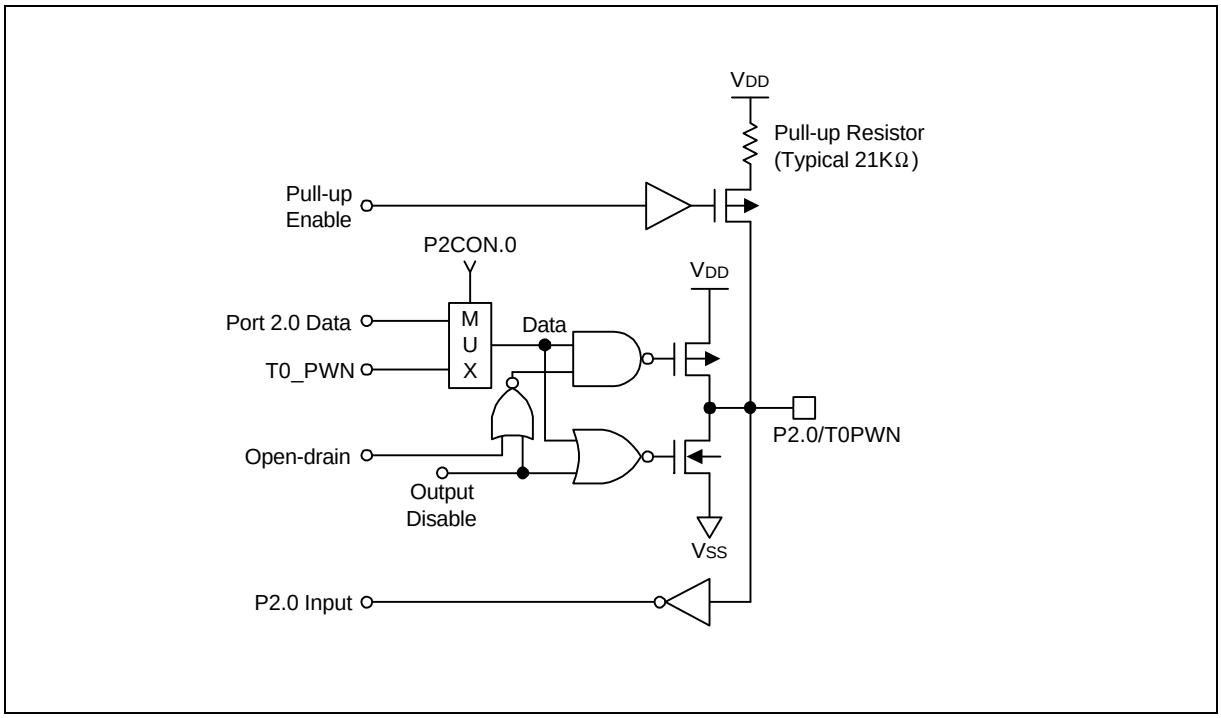


Figure 1-5. Pin Circuit Type 3 (P2.0)

PIN CIRCUITS (Continued)

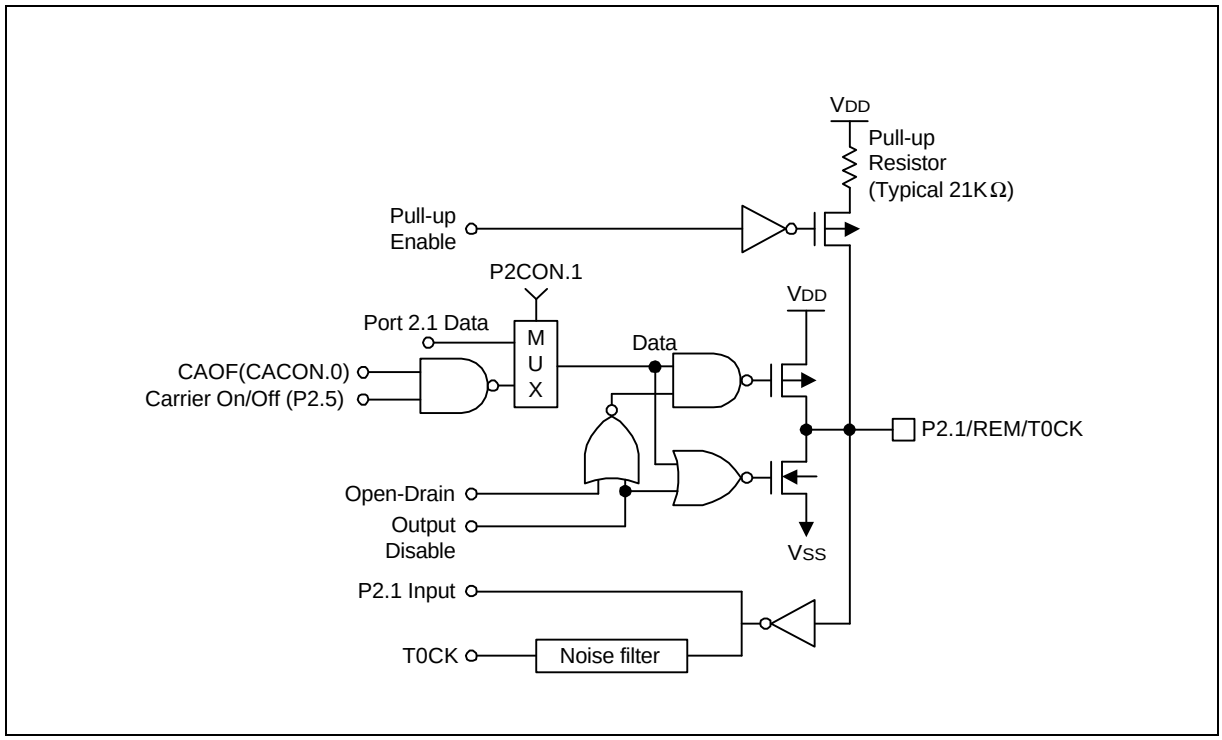


Figure 1-6. Pin Circuit Type 4 (P2.1)

PIN CIRCUITS (Continued)

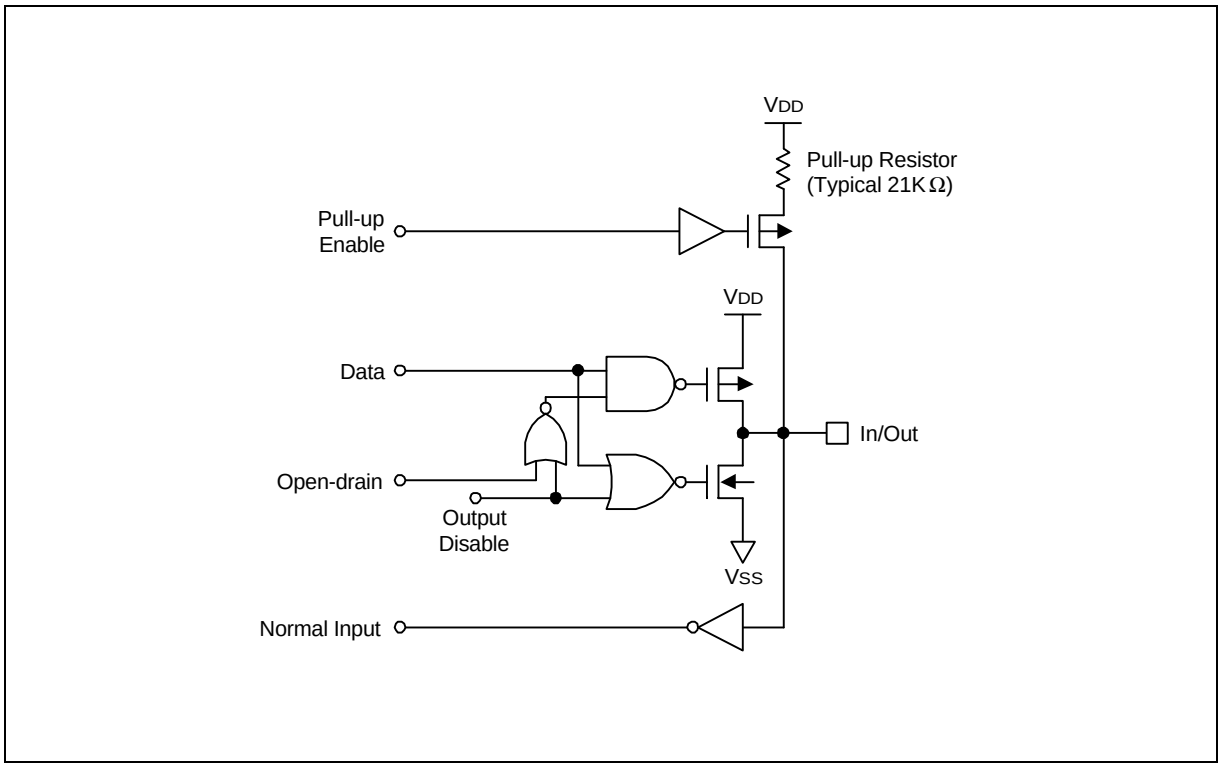


Figure 1-7. Pin Circuit Type 5 (P2.2)

2 ADDRESS SPACES

OVERVIEW

The KS88C01016/C01008/C01004/C01116/C01108/C01104 microcontroller has two types of address space:

— Internal program memory (ROM)

— Internal register file

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 KS88C01016/C01116 has an internal 15,872 byte programmable ROM, the KS88C01008/C01108 has an internal 8-Kbyte programmable ROM, and the KS88C01004/C01104 has an internal 4-Kbyte programmable ROM. An external memory interface is not implemented. The 256-byte physical RAM space is expanded into an addressable area of 320 bytes by the use of addressing modes.

There are 312 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 256 prime register area that is used for general purpose and stack operation). Eighteen 8-bit registers are used for CPU and system control and 22 registers are mapped peripheral control and data registers.

PROGRAM MEMORY (ROM)

Program memory stores program code or table data. The KS88C01016/C01116 has 15, 872 bytes of internal programmable program memory, and the program memory address range is therefore 0000H-3E00H of ROM . The KS88C01008/C01108 has 8-Kbyte(0000H-1FFFH) of internal programmable program memory and the KS88C01004/C01104 has 4-Kbyte(0000H-0FFFH) of internal programmable program memory (see Figure 2-1).

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.

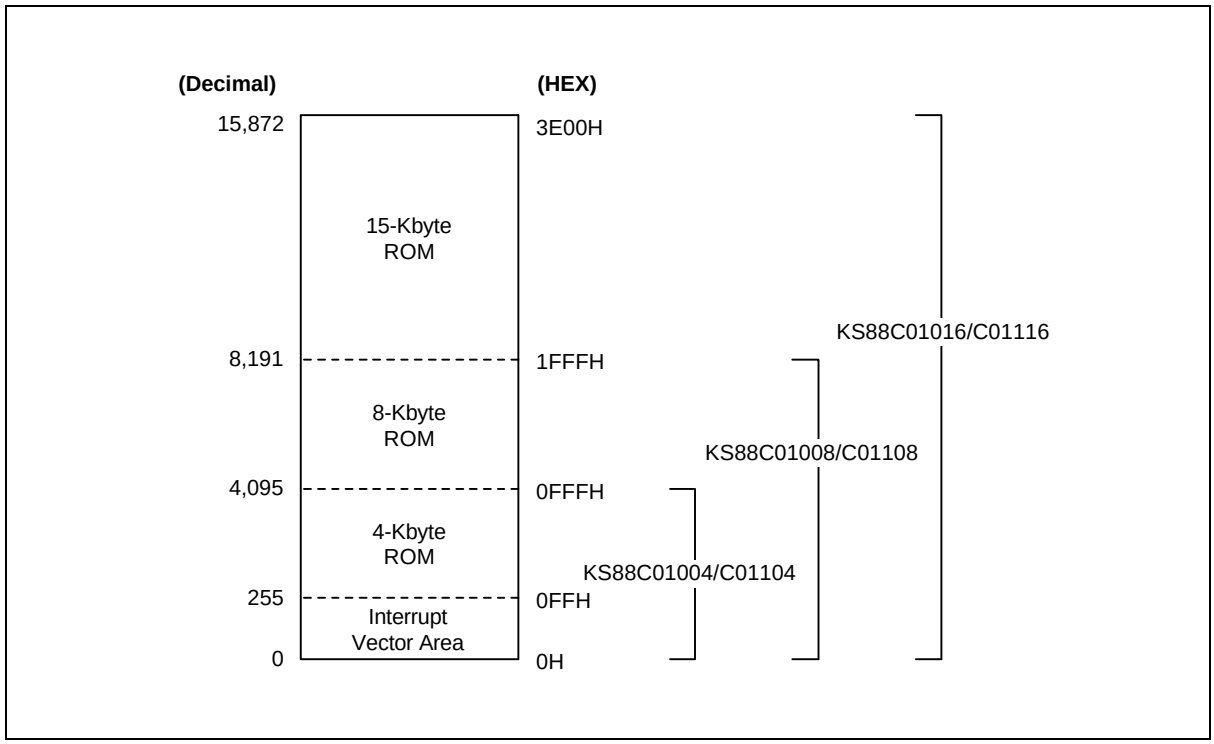


Figure 2-1. Program Memory Address Space

REGISTER ARCHITECTURE

The KS88C01016/C01008/C01004/C01116/C01108/C01104 register file has 312 registers. The upper 64 bytes register files are addressed as system control register and working register. The lower 192-byte area of the physical register file(00H - BFH) contains freely-addressable, general-purpose registers called *prime registers. It can be also used for stack operation.*

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 KS88C01016/C01008/C01004/C01116/C01108/C01104 internal register space are summarized in Table 2-1.

Table 2-1. KS88C01016/C01008/C01004/C01116/C01108/C01104 Register Type Summary

Register Type	Number of Bytes
General-purpose registers (including the 16-byte common working register area, the 256-byte prime register area.)	272
CPU and system control registers	18
Mapped clock, peripheral, and I/O control and data registers	22
Total Addressable Bytes	**312**

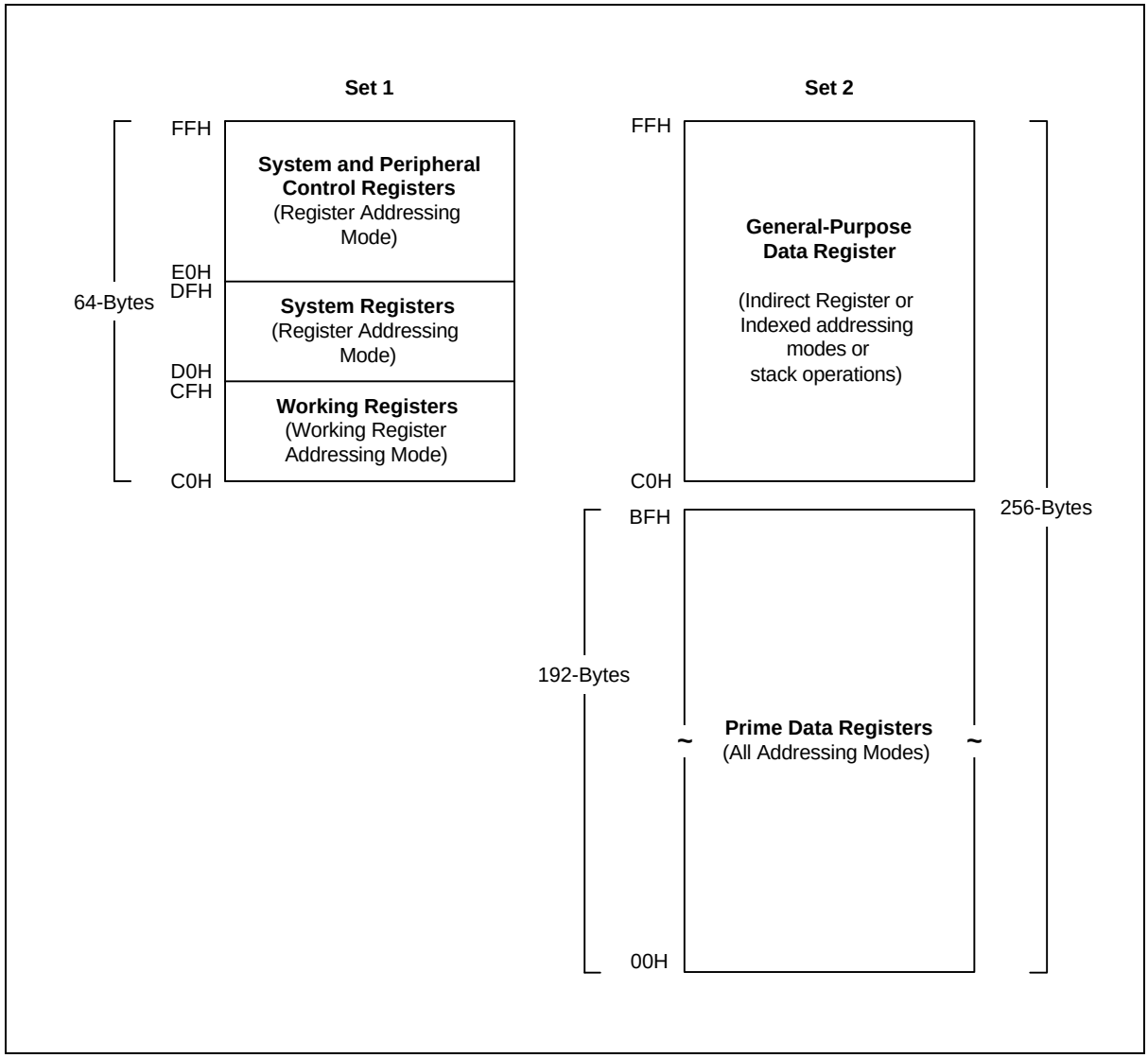


Figure 2-2. Internal Register File Organization

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 KS88C01016/C01008/C01004/C01116/C01108/C01104 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.

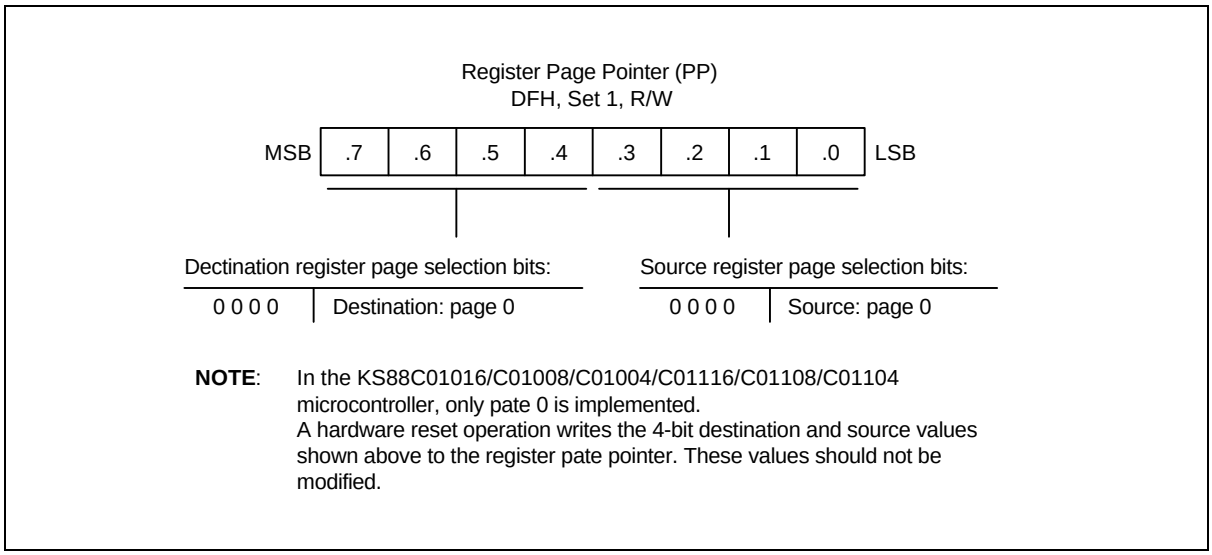


Figure 2-3. Register Page Pointer (PP)

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 KS88C01016/C01008/C01004/C01116/C01108/C01104 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 26 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 KS88C01016/C01008/C01004/C01116/C01108/C01104 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.

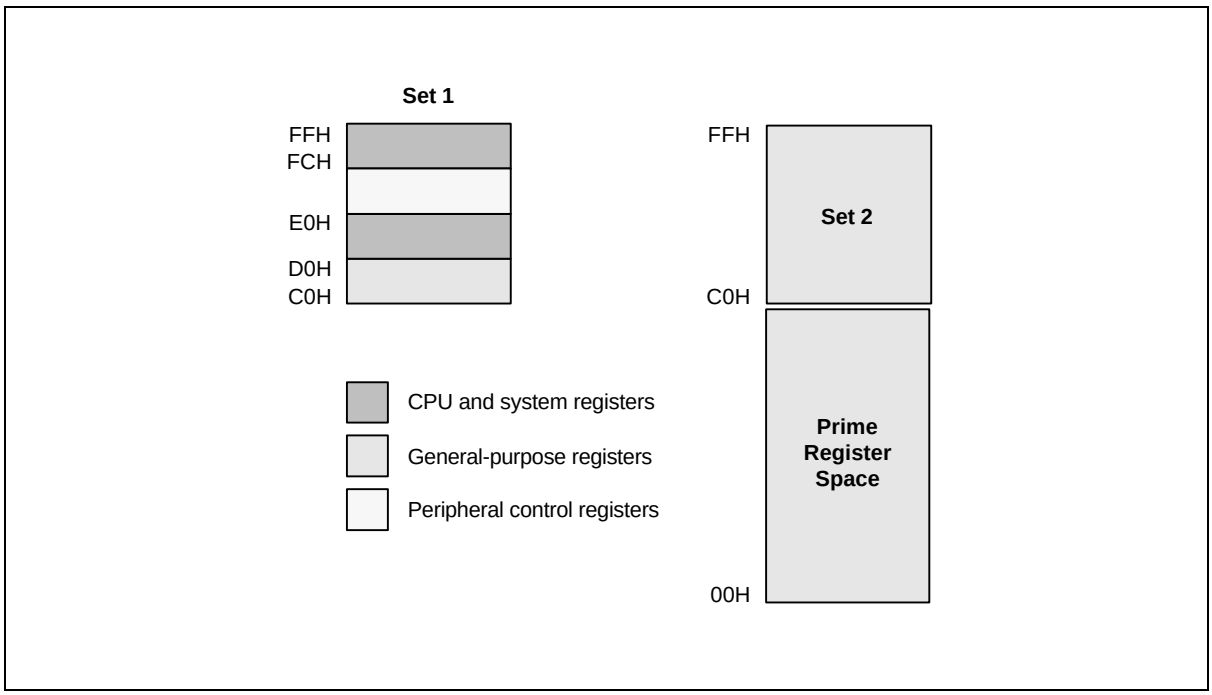


Figure 2-4. Set 1, Set 2, and Prime Area Register Map

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).

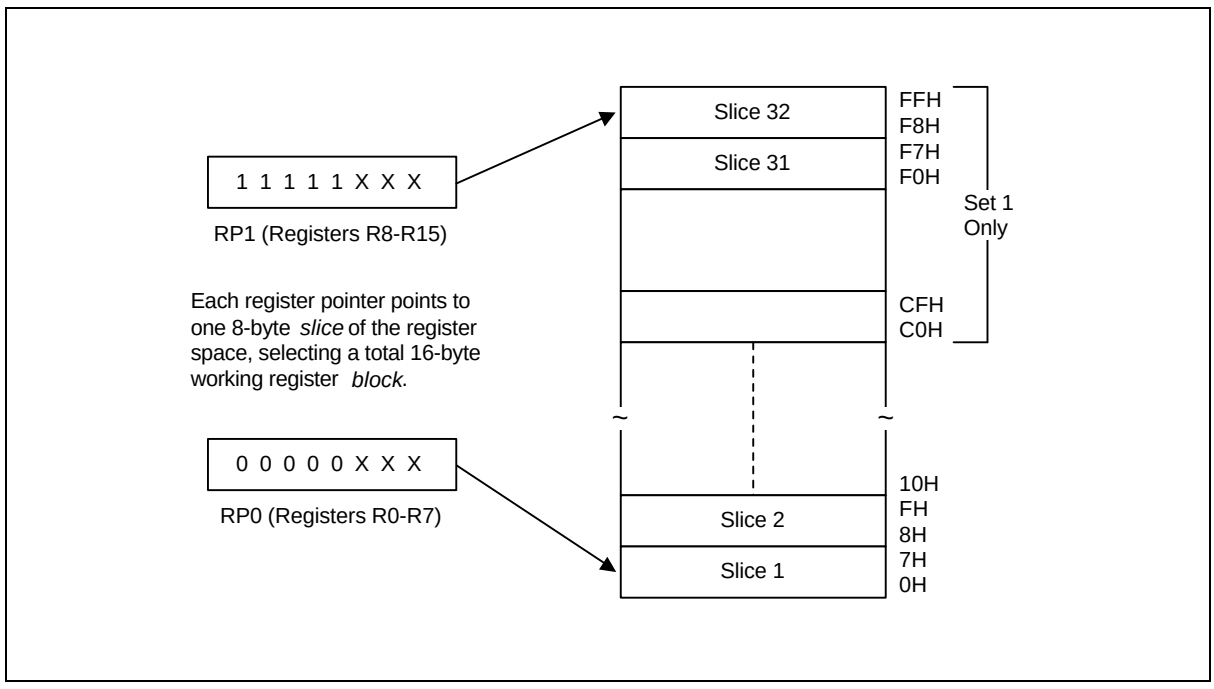


Figure 2-5. 8-Byte Working Register Areas (Slices)

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
```

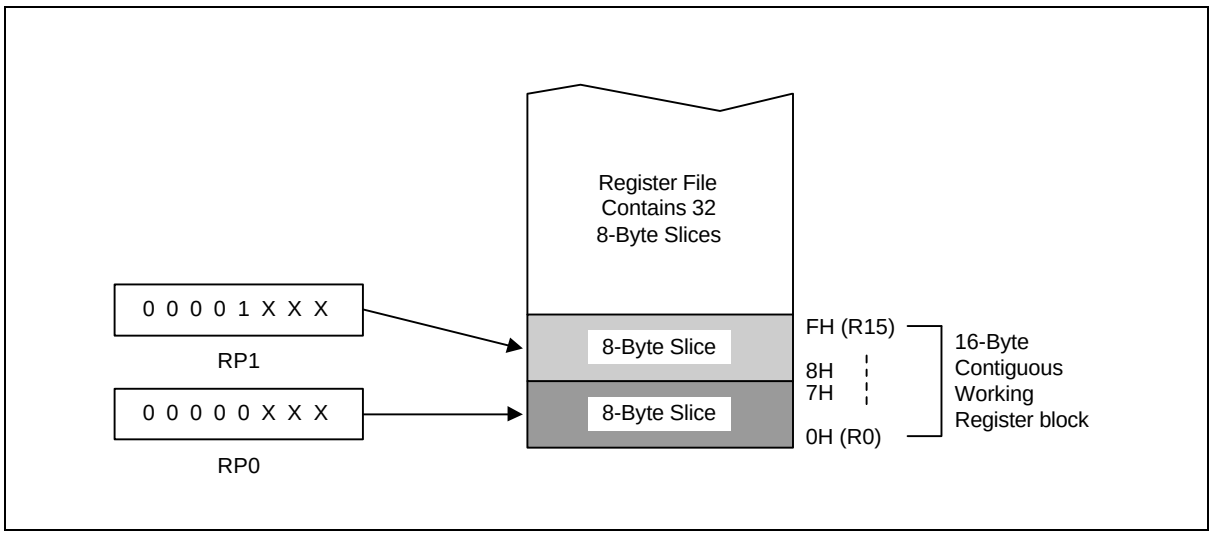


Figure 2-6. Contiguous 16-Byte Working Register Block

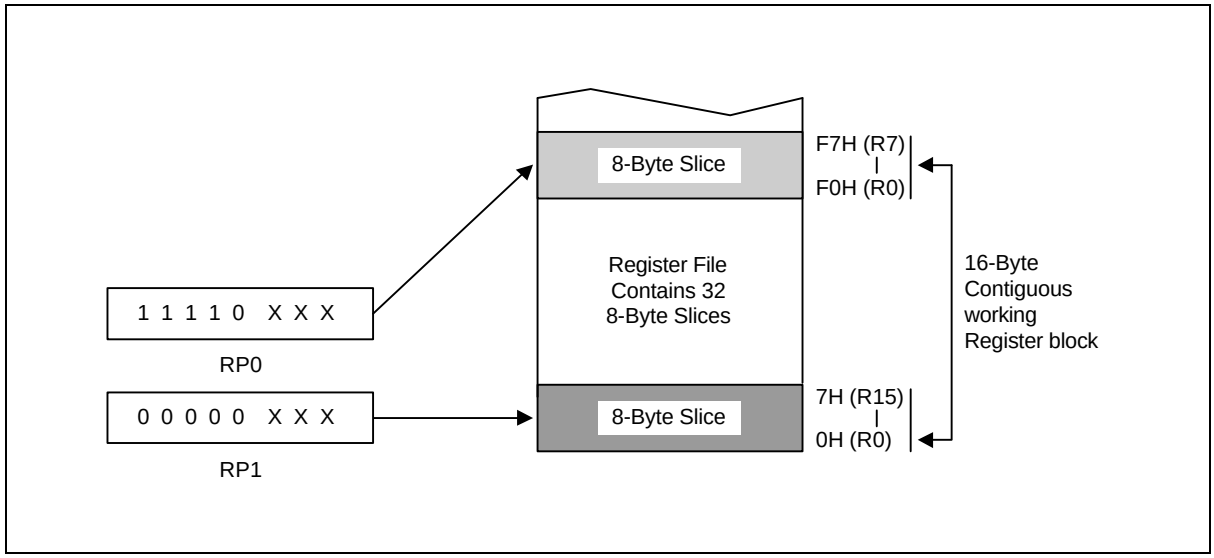


Figure 2-7. Non-Contiguous 16-Byte Working Register Block

☞ 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.

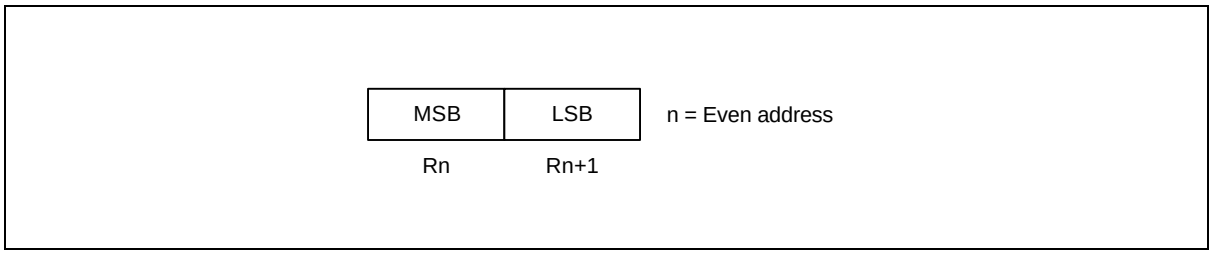


Figure 2-8. 16-Bit Register Pair

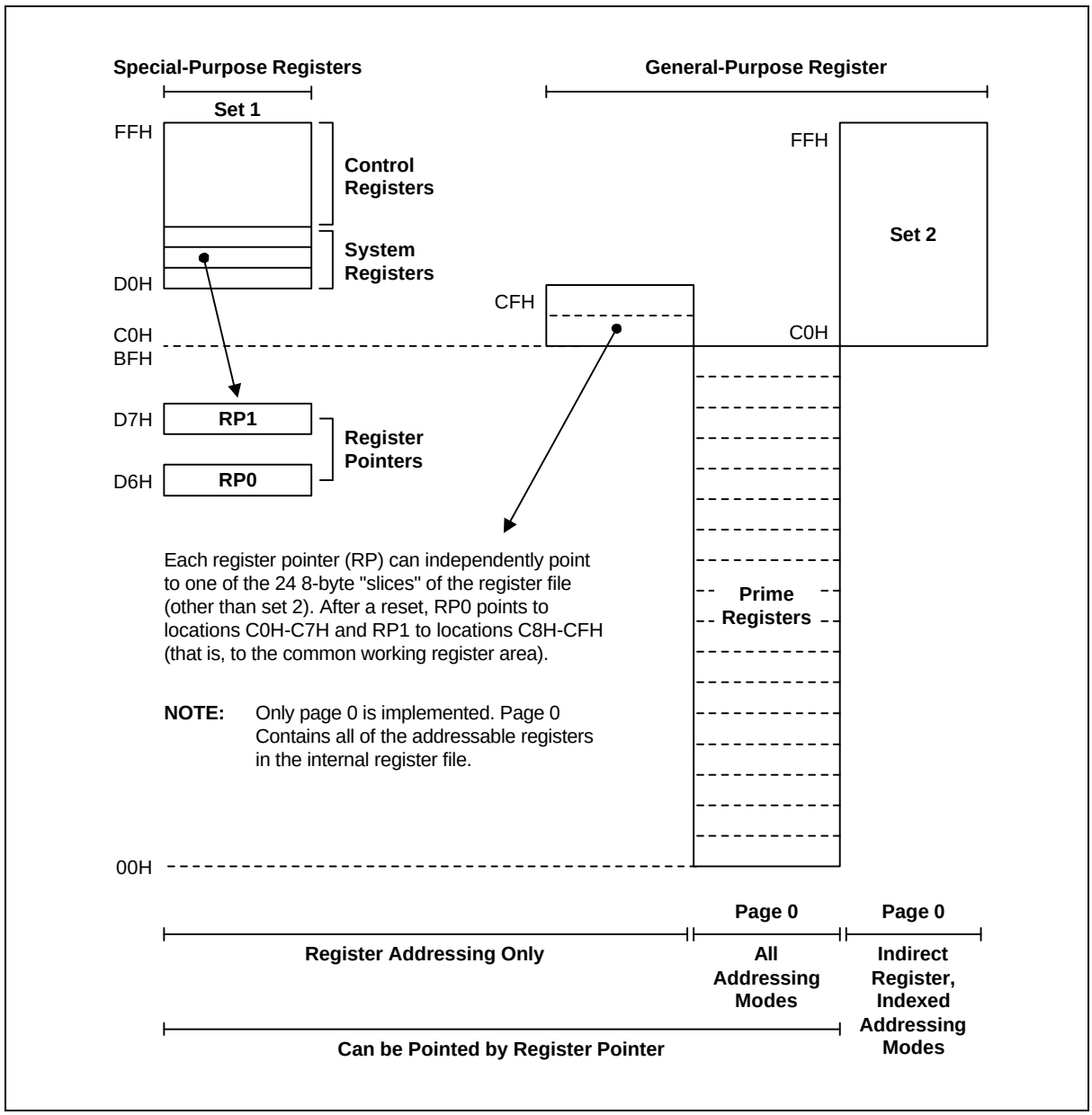


Figure 2-9. Register File Addressing

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.

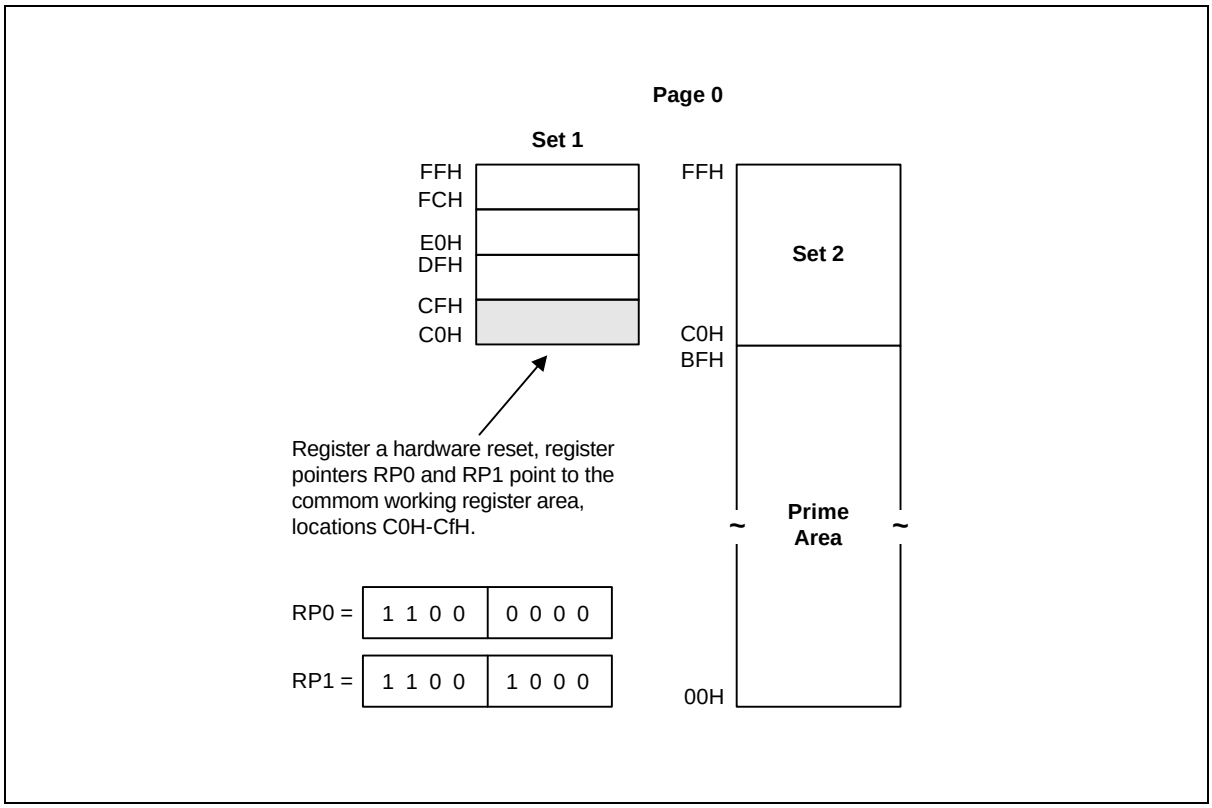


Figure 2-10. Common Working Register Area

☞ 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.

Example 1:

```
LD      0C2H,40H        ;  Invalid addressing mode!
```

Use working register addressing instead:

```
SRP     #0C0H
LD      R2,40H          ;  R2 (C2H) ← the value in location 40H
```

Example 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).

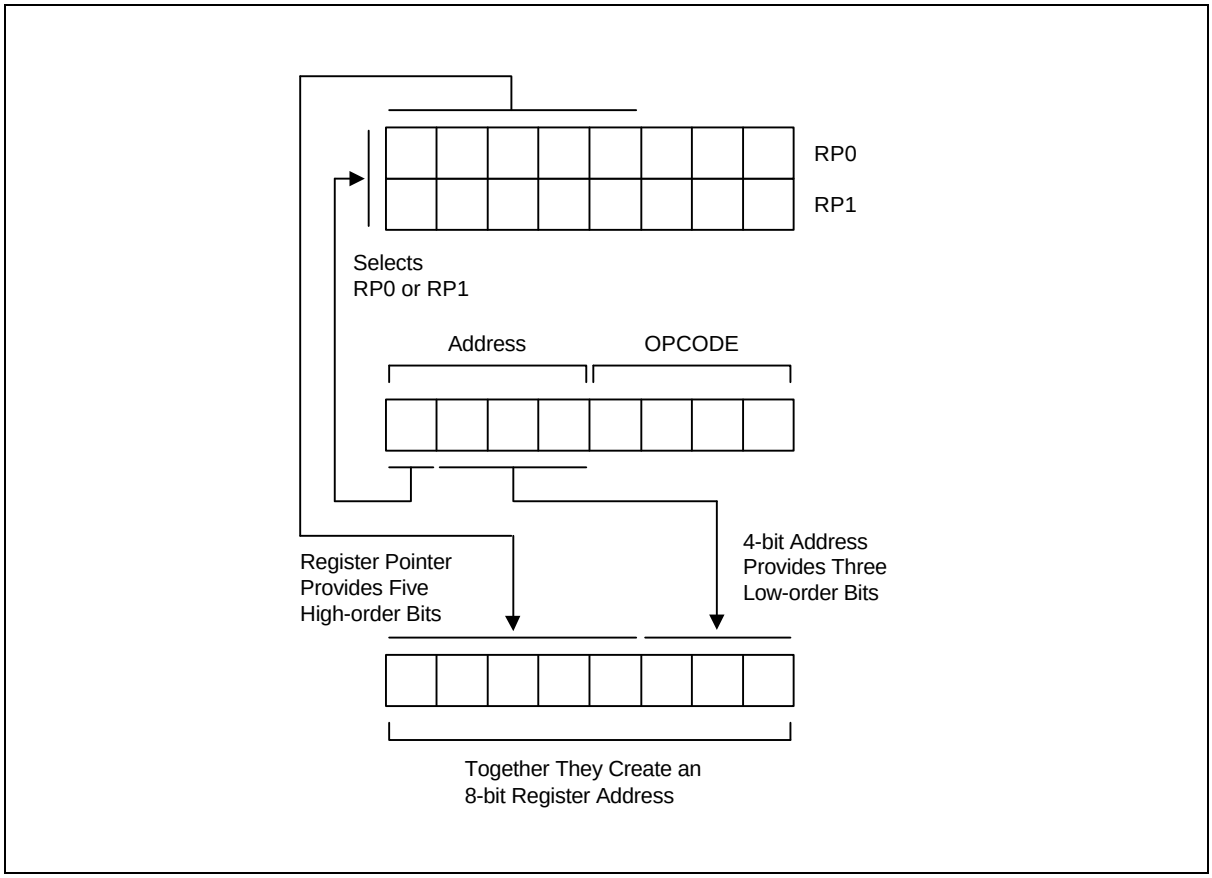


Figure 2-11. 4-Bit Working Register Addressing

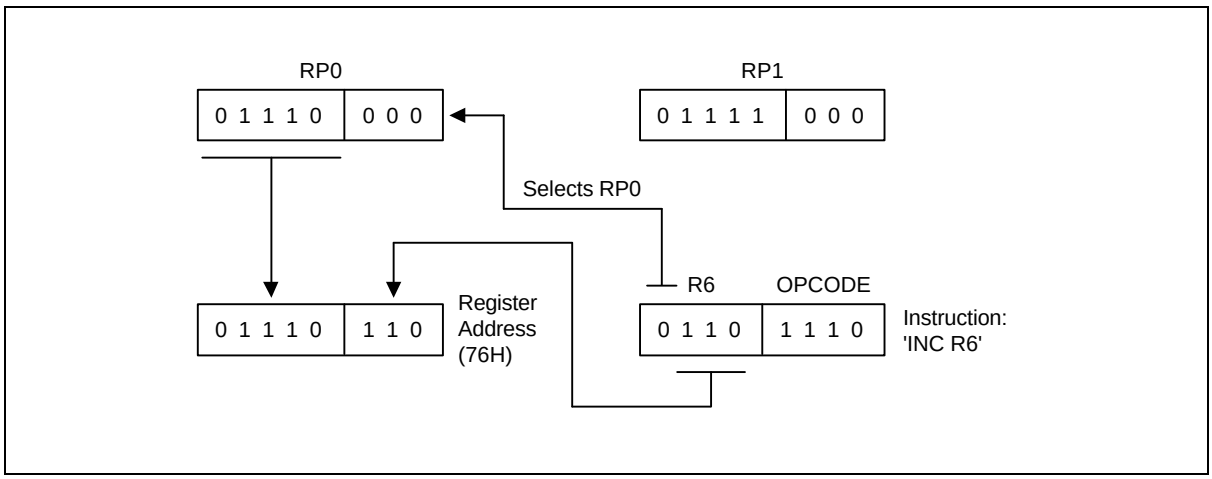


Figure 2-12. 4-Bit Working Register Addressing Example

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).

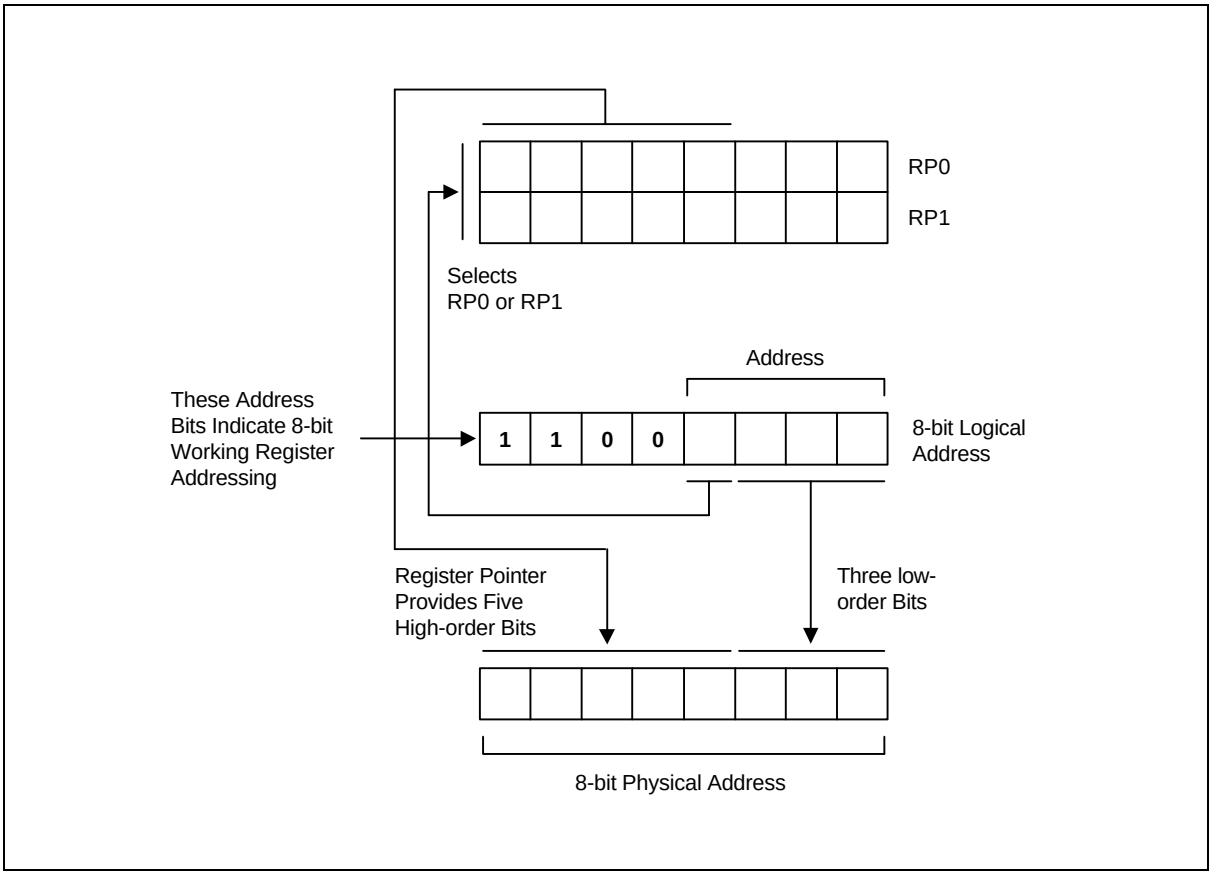


Figure 2-13. 8-Bit Working Register Addressing

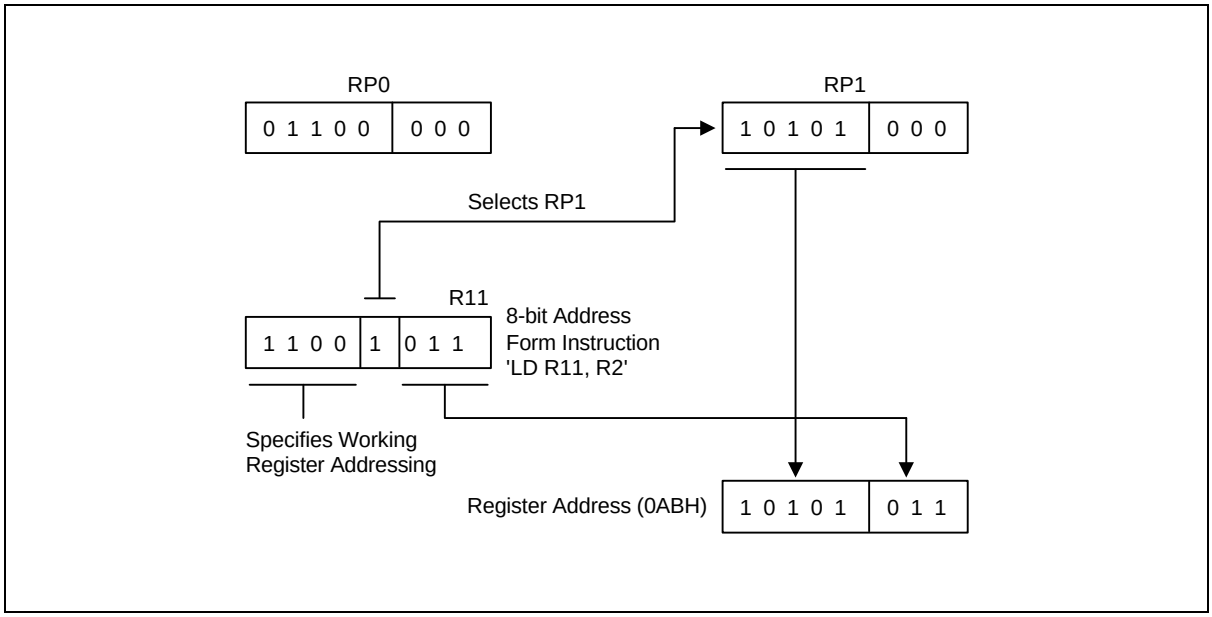


Figure 2-14. 8-Bit Working Register Addressing Example

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 KS88C01016/C01008/C01004/C01116/C01108/C01104 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.

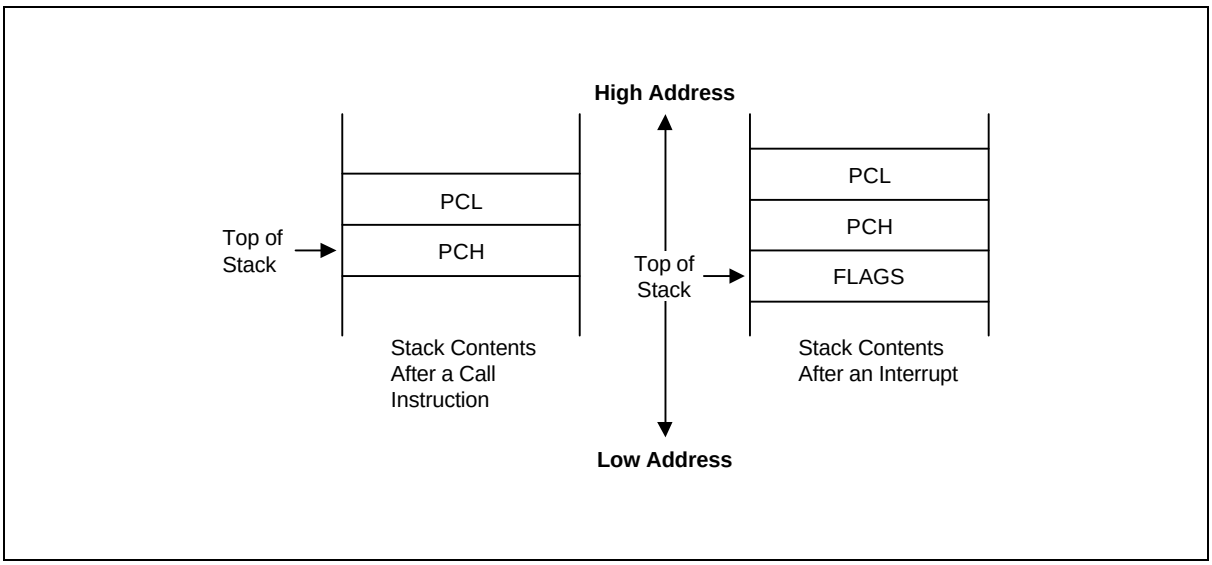


Figure 2-15. Stack Operations

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 KS88C01016/C01008/C01004/C01116/C01108/C01104, 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

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

The KS88-series instruction set supports seven explicit addressing modes. Not all of these addressing modes are available for each instruction:

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

REGISTER ADDRESSING MODE (R)

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

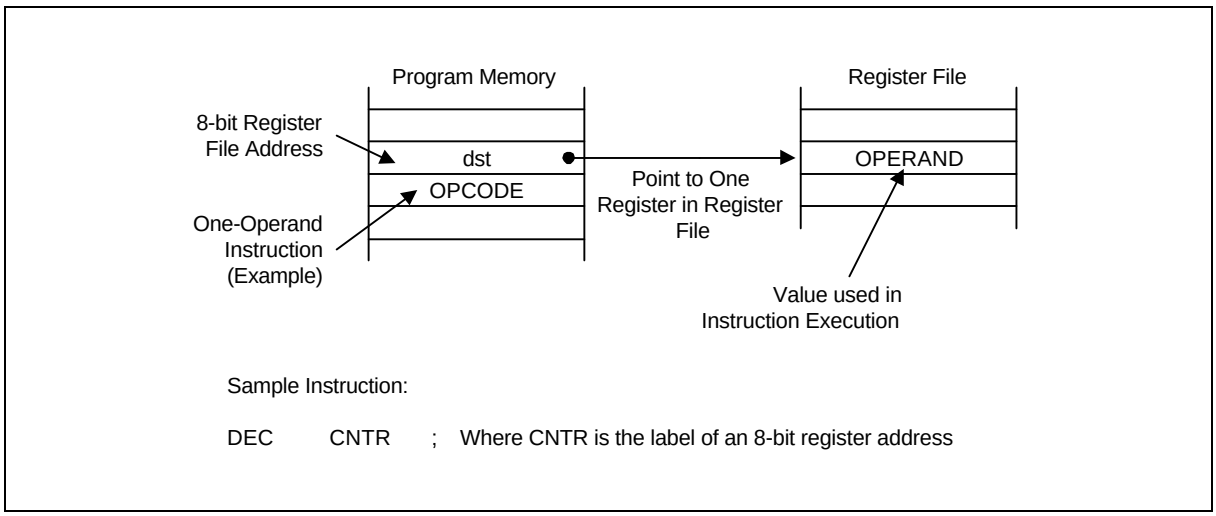


Figure 3-1. Register Addressing

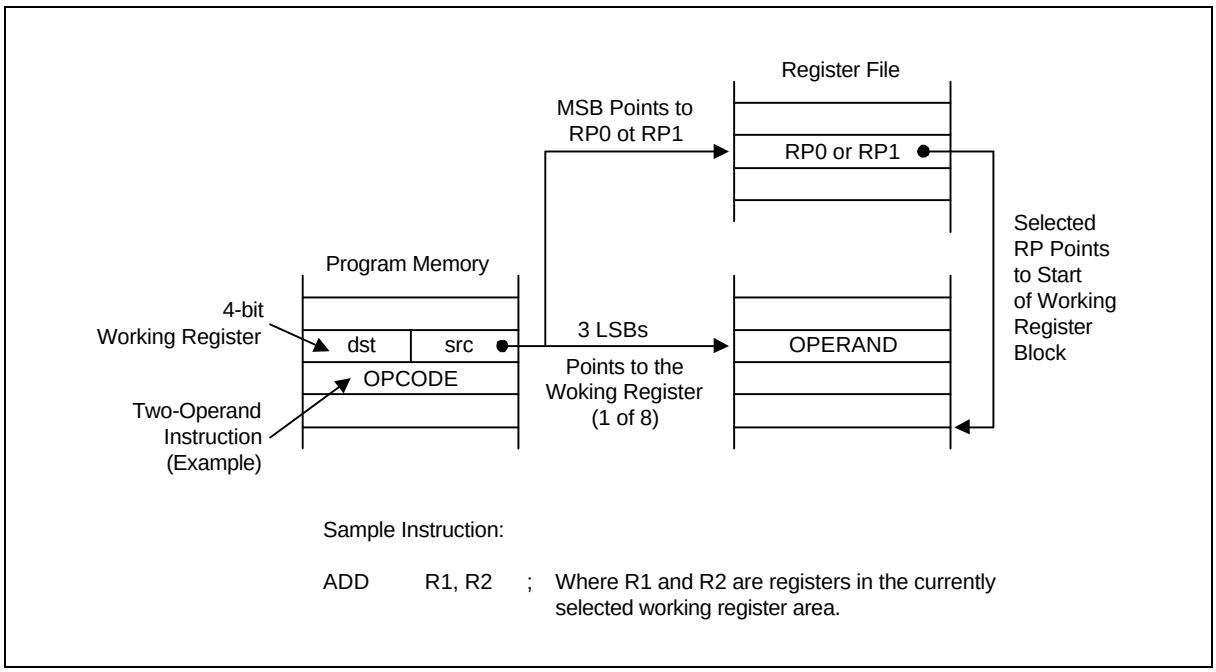


Figure 3-2. Working Register Addressing

INDIRECT REGISTER ADDRESSING MODE (IR)

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

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

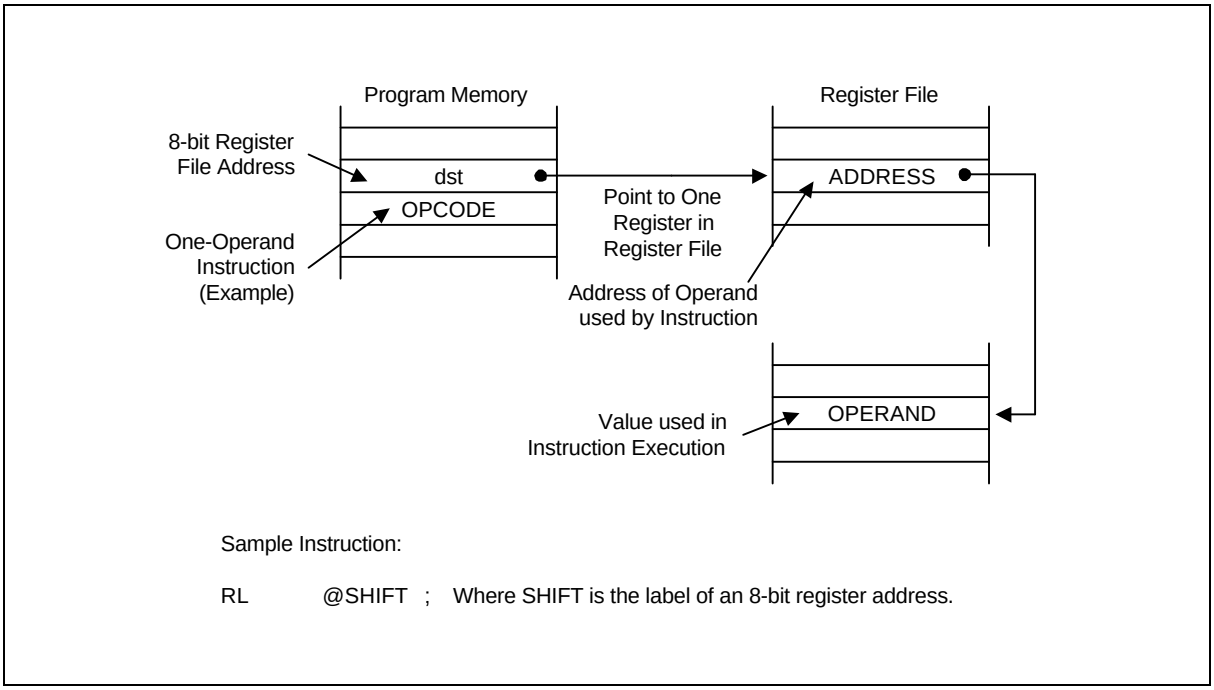


Figure 3-3. Indirect Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Continued)

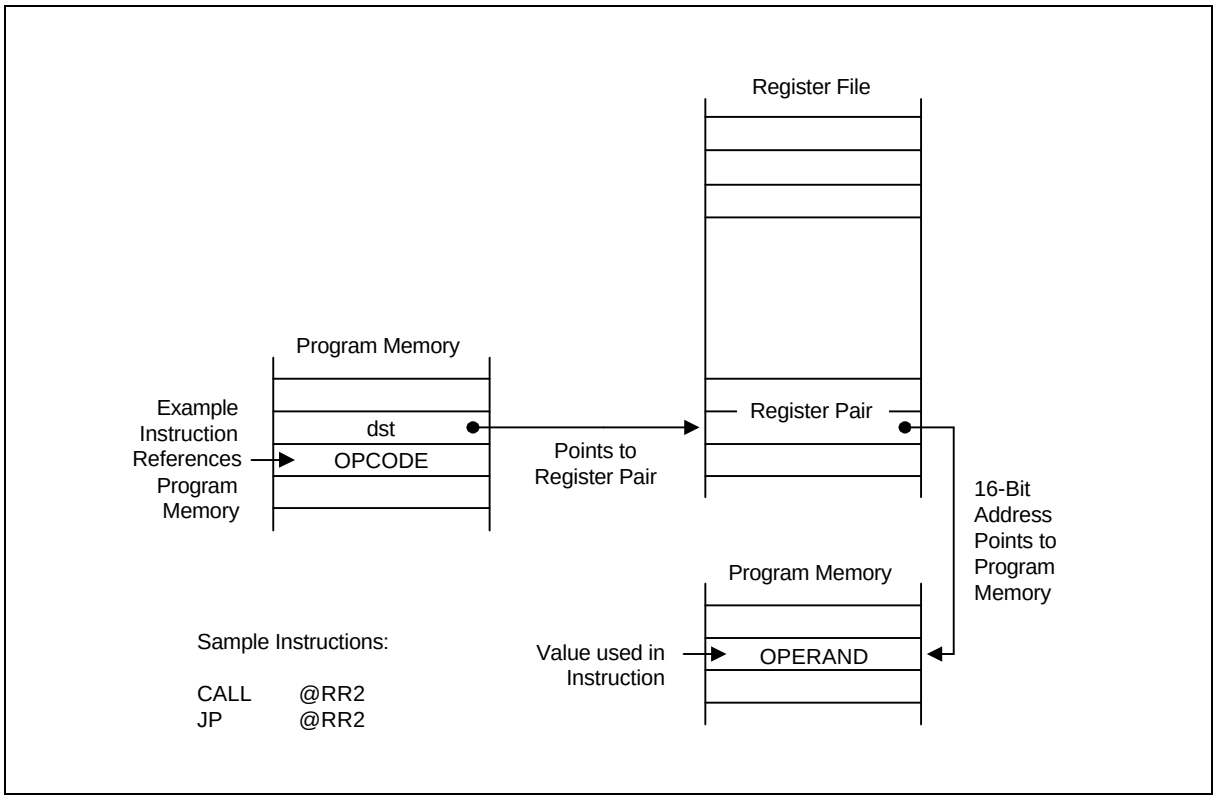


Figure 3-4. Indirect Register Addressing to Program Memory

INDIRECT REGISTER ADDRESSING MODE (Continued)

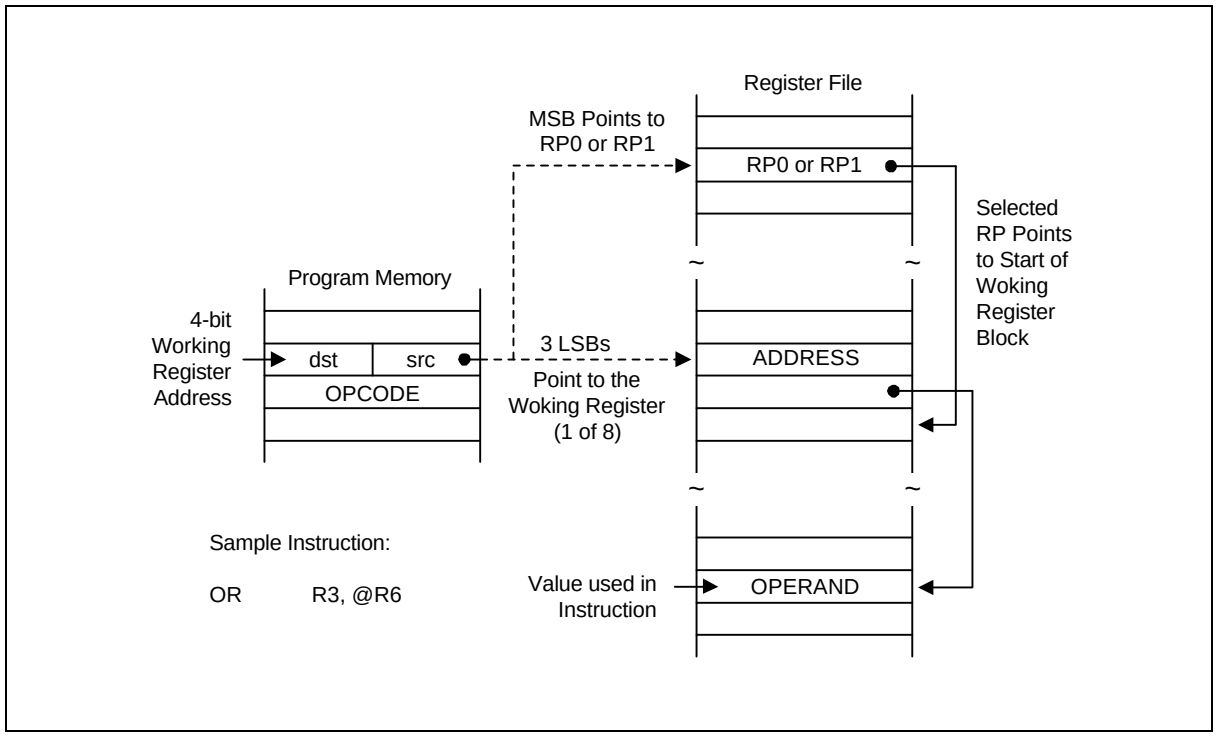
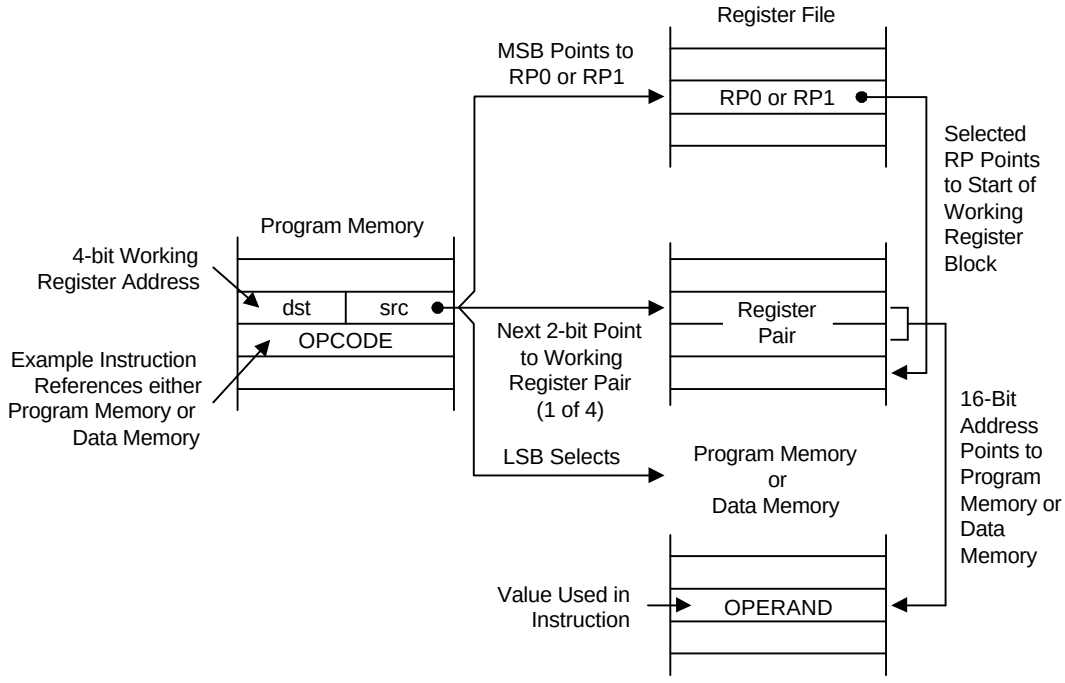


Figure 3-5. Indirect Working Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Continued)



Sample Instructions:

```
LDC     R5,@RR6       ;  Program memory access
LDE     R3,@RR14      ;  External data memory access
LDE     @RR4, R8      ;  External data memory access
```

NOTE: LDE command is not available, because an external interface is not implemented for the KS88C01016/C01008/C01004/C01116/C01108/C01104.

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 (if implemented). You cannot, however, access locations C0H–FFH in set 1 using Indexed addressing.

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

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

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

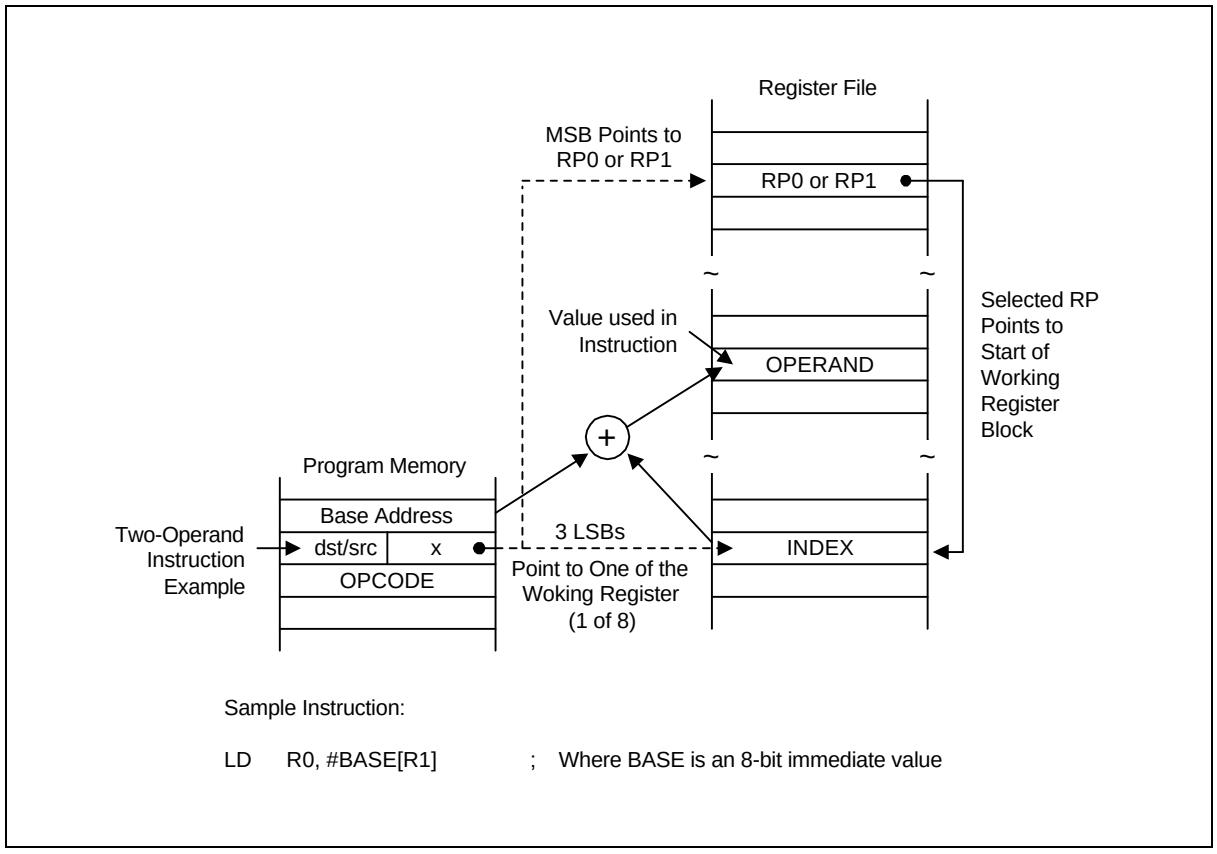
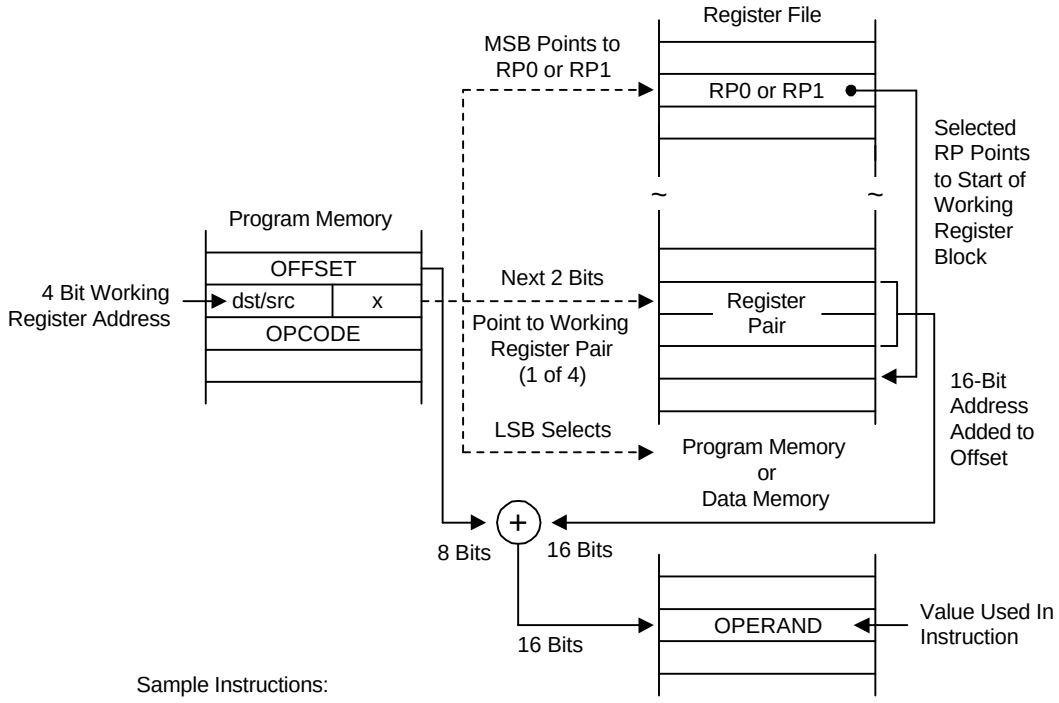


Figure 3-7. Indexed Addressing to Register File

INDEXED ADDRESSING MODE (Continued)



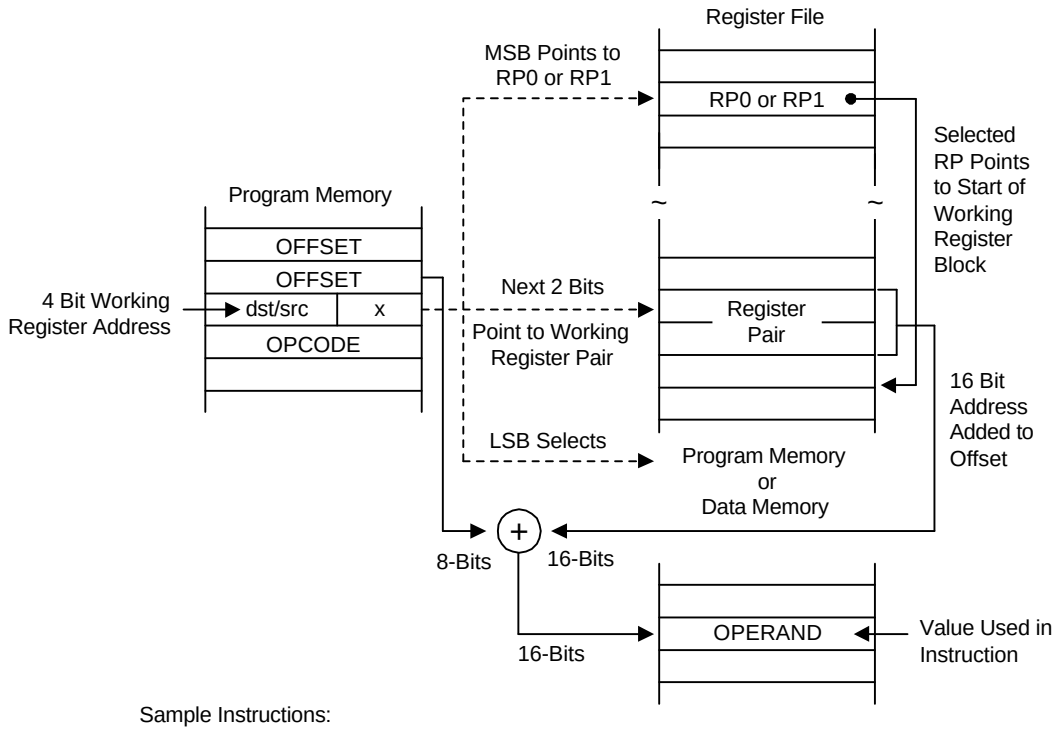
Sample Instructions:

```
LDC     R4, #04H[RR2]    ;  The values in the program address (RR2 + 04H) are
                            loaded into register R4.
LDE     R4,#04H[RR2]     ;  Identical operation to LDC example, except that
                            external program memory is accessed.
```

NOTE: LDE command is not available, because an external interface is not implemented for the KS88C01016/C01008/C01004/C01116/C01108/C01104.

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

INDEXED ADDRESSING MODE (Continued)



Sample Instructions:

```
LDC     R4, #1000H[RR2]   ;  The values in the program address (RR2 + 1000H)
                             are loaded into register R4.
LDE     R4, #1000H[RR2]   ;  Identical operation to LDC example, except that
                             external program memory is accessed.
```

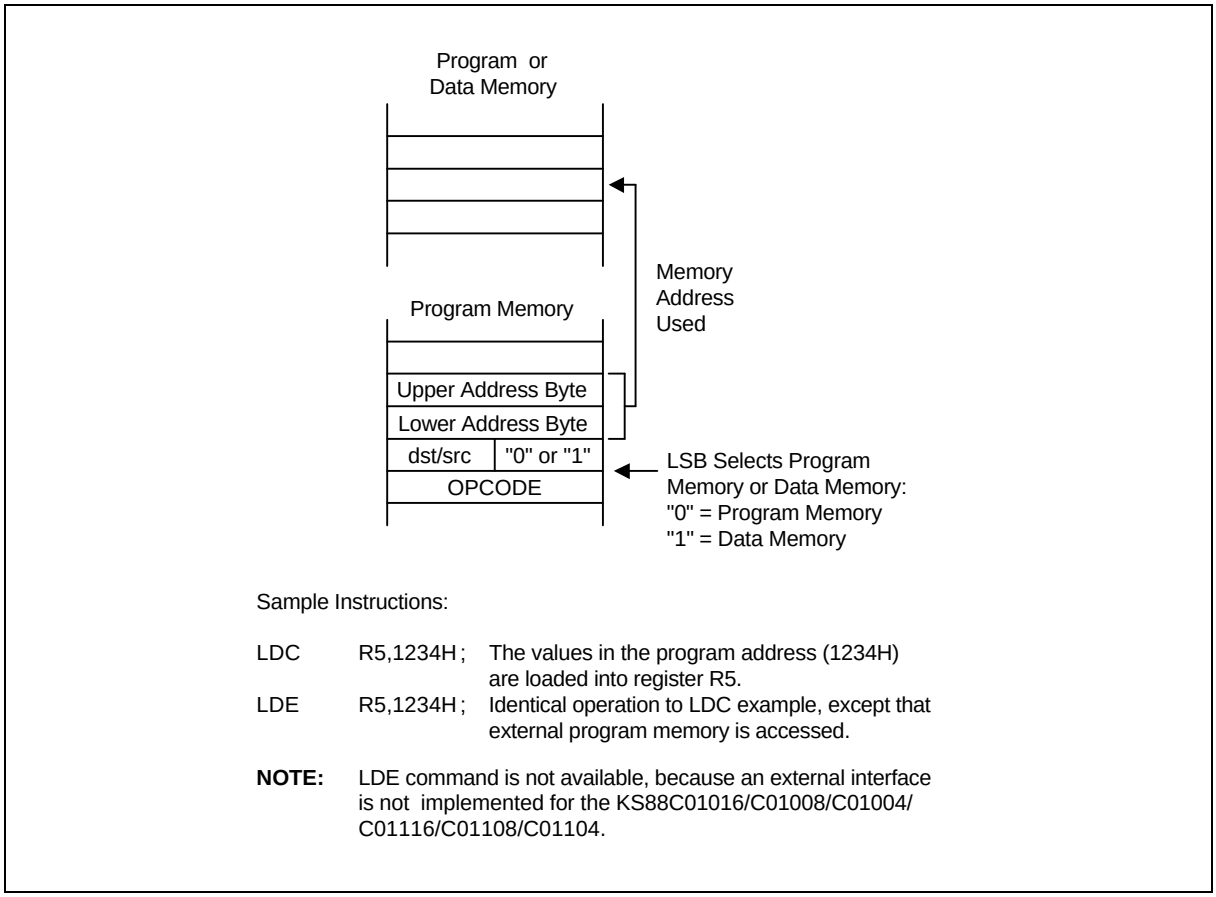
NOTE: LDE command is not available, because an external interface is not implemented for the KS88C01016/C01008/C01004/C0116/C01108/C01104.

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

DIRECT ADDRESS MODE (DA)

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

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



Sample Instructions:

LDC	R5,1234H ;	The values in the program address (1234H) are loaded into register R5.
LDE	R5,1234H ;	Identical operation to LDC example, except that external program memory is accessed.

NOTE: LDE command is not available, because an external interface is not implemented for the KS88C01016/C01008/C01004/ C01116/C01108/C01104.

Figure 3-10. Direct Addressing for Load Instructions

DIRECT ADDRESS MODE (Continued)

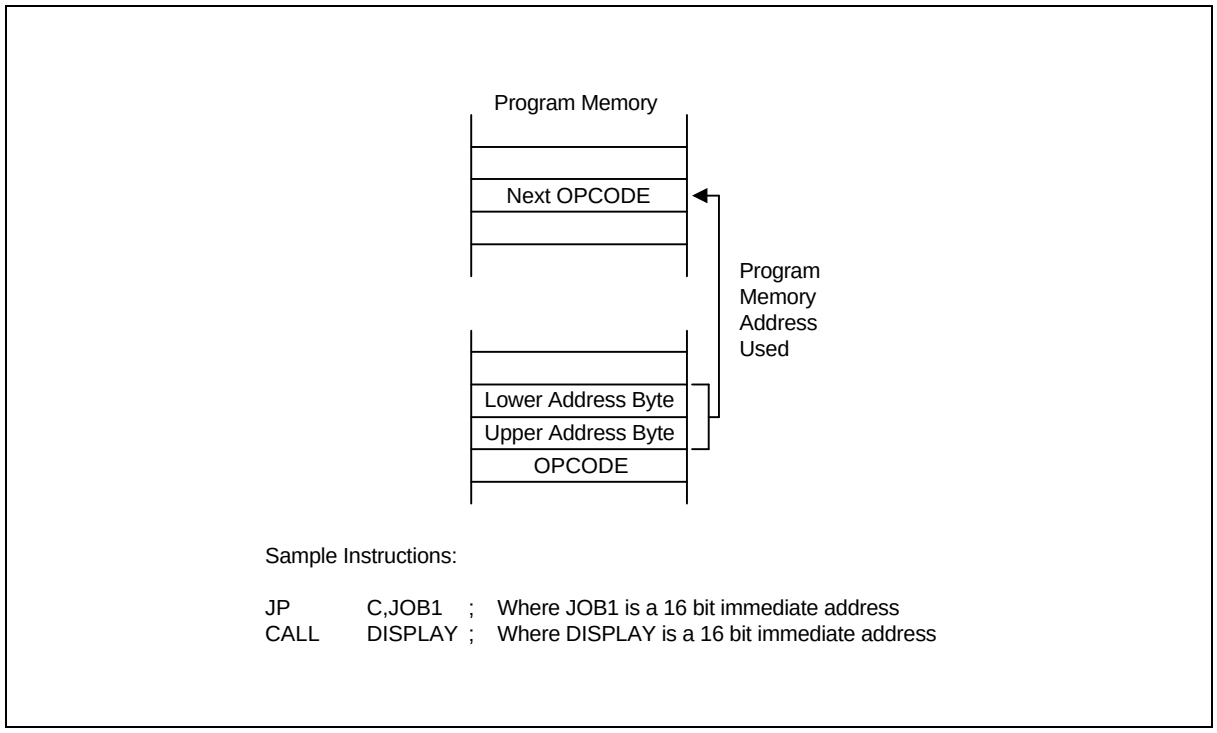


Figure 3-11. Direct Addressing for Call and Jump Instructions

INDIRECT ADDRESS MODE (IA)

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

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

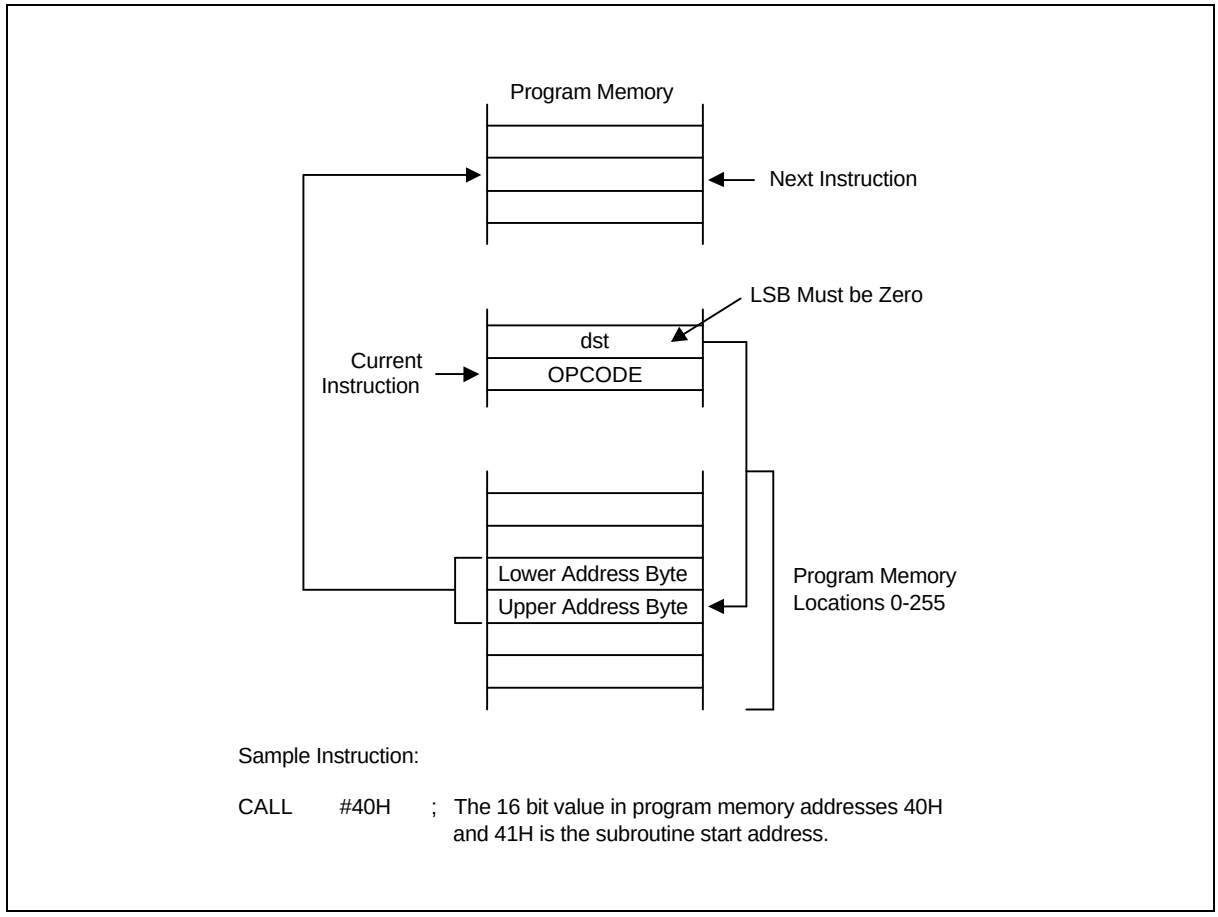


Figure 3-12. Indirect Addressing

RELATIVE ADDRESS MODE (RA)

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

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

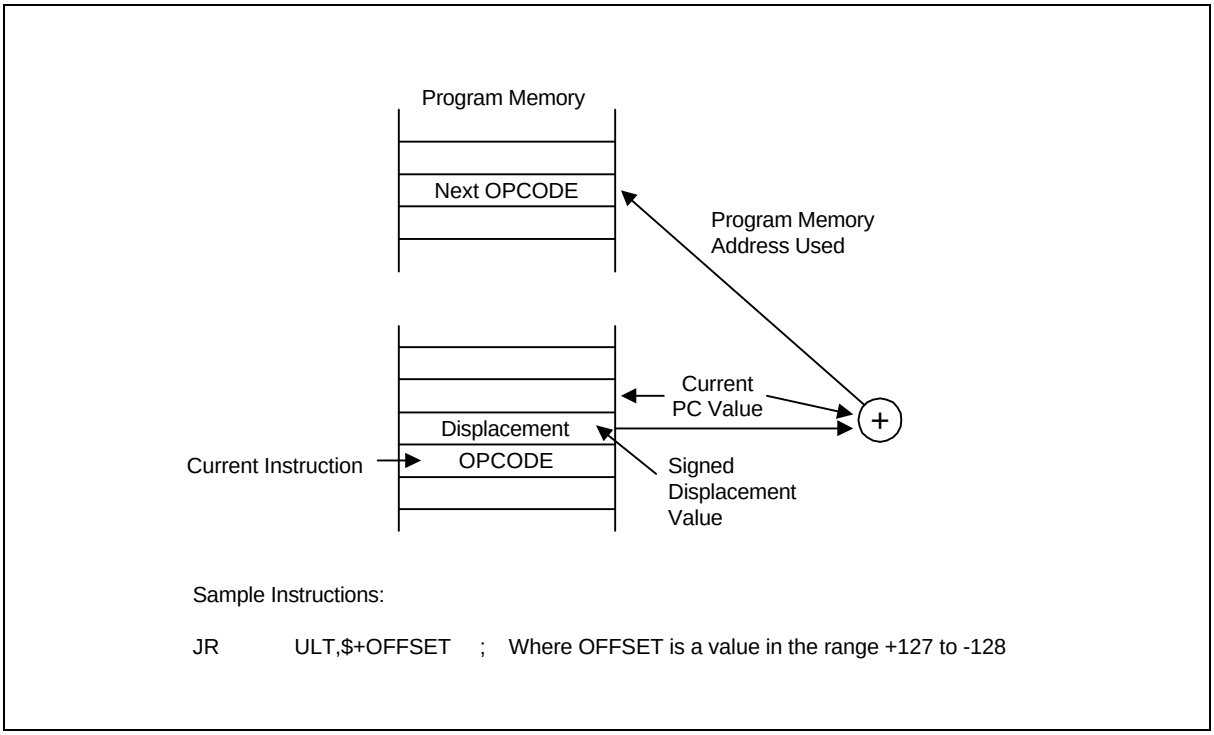


Figure 3-13. Relative Addressing

IMMEDIATE MODE (IM)

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

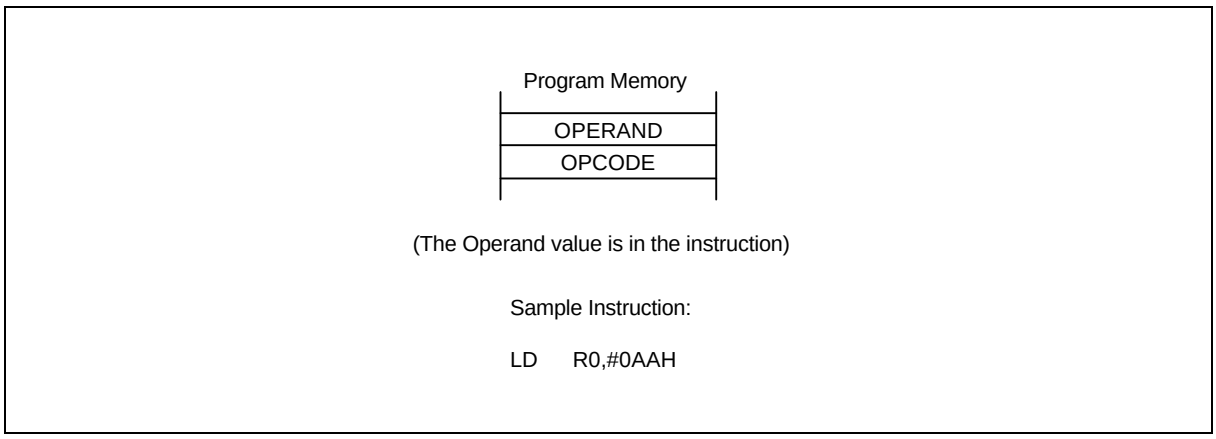


Figure 3-14. Immediate Addressing

4 CONTROL REGISTERS

OVERVIEW

In this section, detailed descriptions of the KS88C01016/C01008/C01004/C01116/C01108/C01104 control registers are presented in an easy-to-read format. You can use this section as a quick-reference source when writing application programs. Figure 4-1 illustrates the important features of the standard register description format.

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

Data and counter registers are not described in detail in this reference section. More information about all of the registers used by a specific peripheral is presented in the corresponding peripheral descriptions in Part II of this manual.

Table 4-1. Mapped Registers (Set1)

Register Name	Mnemonic	Decimal	Hex	R/W
Timer 0 counter	T0CNT	208	D0H	R(note)
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
Locations D8H is not mapped.				
Stack pointer (low byte)	SPL	217	D9H	R/W
Instruction pointer (high byte)	IPH	218	DAH	R/W
Instruction pointer (low byte)	IPL	219	DBH	R/W
Interrupt request register	IRQ	220	DCH	R (note)
Interrupt mask register	IMR	221	DDH	R/W
System mode register	SYM	222	DEH	R/W
Register page pointer	PP	223	DFH	R/W
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
Location E3H - E6 is not mapped.				
Port 0 pull-up resistor enable register	P0PUR	231	E7H	R/W
Port 0 control register (high byte)	P0CONH	232	E8H	R/W
Port 0 control register (low byte)	P0CONL	233	E9H	R/W
Port 1 control register (high byte)	P1CONH	234	EAH	R/W
Port 1 control register (low byte)	P1CONL	235	EBH	R/W
Port 1 pull-up resistor enable register	P1PUR	236	ECH	R/W
Location EDH - EF is not mapped.				
Port 2 control register	P2CON	239	F0H	R/W
Port 0 interrupt enable register	P0INT	241	F1H	R/W
Port 0 interrupt pending register	P0PND	242	F2H	R/W
Counter A control register	CACON	243	F3H	R/W
Counter A data register (high byte)	CADATAH	244	F4H	R/W
Counter A data register (low byte)	CADATAL	245	F5H	R/W
Timer 1 counter register (high byte)	T1CNTH	246	F6H	R (note)
Timer 1 counter register (low byte)	T1CNTL	247	F7H	R (note)
Timer 1 data register (high byte)	T1DATAH	248	F8H	R/W

Table 4-1. Mapped Registers (Continued)

Register Name	Mnemonic	Decimal	Hex	R/W
Timer 1 data register (low byte)	T1DATAL	249	F9H	R/W
Timer 1 control register	T1CON	250	FAH	R/W
STOP Control register	STOPCON	251	FBH	W
Locations FCH is not mapped.				
Basic timer counter	BTCNT	253	FDH	R (note)
External memory timing register	EMT	254	FEH	R/W
Interrupt priority register	IPR	255	FFH	R/W

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

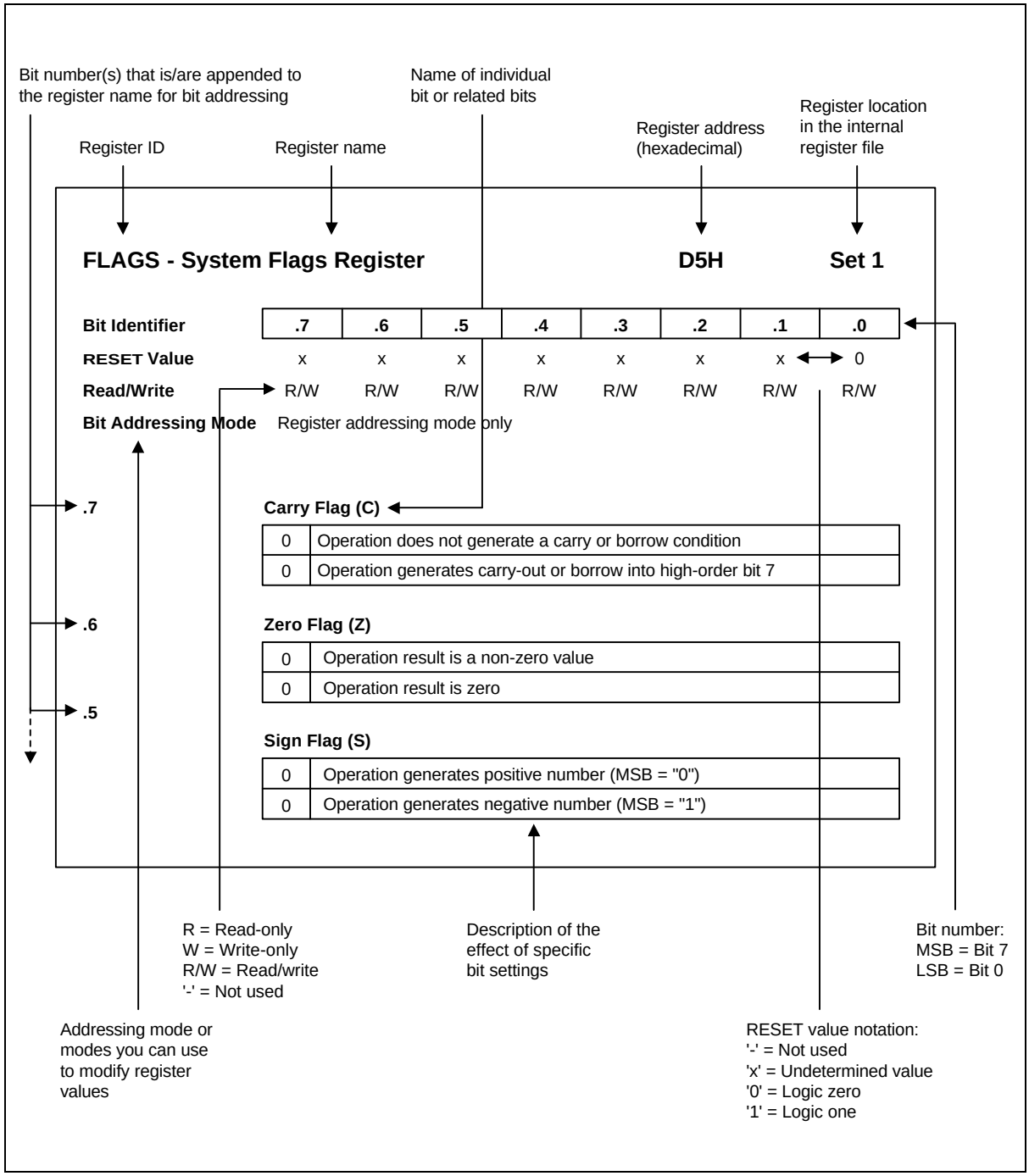


Figure 4-1. Register Description Format

BTCON — Basic Timer Control Register D3H SET1

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
Bit Addressing	Register addressing mode only							

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

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

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

0	0	$f_{OSC}/4096$
0	1	$f_{OSC}/1024$
1	0	$f_{OSC}/128$
1	1	Invalid setting; not used for KS88C01016/C01008/C01004/C01116/C01108/C01104

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

0	No effect
1	Clear the basic timer counter value

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

0	No effect
1	Clear both clock frequency dividers

NOTES:

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

CACON — Counter A Control Register F3H SET1

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 Counter A Input Clock Selection Bits

0	0	f_{OSC}
0	1	$f_{OSC}/2$
1	0	$f_{OSC}/4$
1	1	$f_{OSC}/8$

.5 - .4 Counter A Interrupt Timing Selection Bits

0	0	Elapsed time for Low data value
0	1	Elapsed time for High data value
1	0	Elapsed time for combined Low and High data values
1	1	Invalid setting; not used for KS88C01016/C01008/C01004/C01116/C01108/C01104

.3 Counter A Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.2 Counter A Start Bit

0	Stop counter A
1	Start counter A

.1 Counter A Mode Selection Bit

0	One-shot mode
1	Repeating mode

.0 Counter A Output Flip-Flop Control Bit

0	Flip-flop Low level (T-FF = Low)
1	Flip-flop High level (T-FF = High)

CLKCON — System Clock Control Register D4H SET1

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 Oscillator IRQ Wake-up Function Enable Bit

Not used for KS88C01016/C01008/C01004/C01116/C01108/C01104 .

.6 - .5 Main Oscillator Stop Control Bits

Not used for KS88C01016/C01008/C01004/C01116/C01108/C01104 .

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

0	0	$f_{OSC}/16$
0	1	$f_{OSC}/8$
1	0	$f_{OSC}/2$
1	1	f_{OSC} (non-divided)

.2 - 0 Subsystem Clock Selection Bit (2)

1	0	1	Invalid setting for KS88C01016/C01008/C01004/C01116/C01108/C01104
Other value			Select main system clock (MCLK)

NOTES:

1. After a reset, the slowest clock (divided by 16) is selected as the system clock. To select faster clock speeds, load the appropriate values to CLKCON.3 and CLKCON.4.
2. These selection bits are required only for systems that have a main clock and a subsystem clock. The KS88C01016/C01008/C01004/C01116/C01108/C01104 uses only the main oscillator clock circuit. For this reason, the setting '101B' is invalid.

EMT — External Memory Timing Register (note) FEH SET1

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

.7 External WAIT Input Function Enable Bit

0	Disable WAIT input function for external device
1	Enable WAIT input function for external device

.6 Slow Memory Timing Enable Bit

0	Disable WAIT input function for external device
1	Enable WAIT input function for external device

.5 - .4 Program Memory Automatic Wait Control Bits

0	0	No wait
0	1	Wait one cycle
1	0	Wait two cycles
1	1	Wait three cycles

.3 - .2 Data Memory Automatic Wait Control Bits

0	0	No wait
0	1	Wait one cycle
1	0	Wait two cycles
1	1	Wait three cycles

.1 Stack Area Selection Bit

0	Select internal register file area
1	Select external data memory area

.0 Not used for KS88C01016/C01008/C01004/C01116/C01108/C01104

NOTE: The EMT register is not used for KS88C01016/C01008/C01004/C01116/C01108/C01104 , because an external peripheral interface is not implemented in the KS88C01016/C01008/C01004/C01116/C01108/C01104 . The program initialization routine should clear the EMT register to '00H' following a reset. Modification of EMT values during normal operation may cause a system malfunction.

FLAGS — System Flags Register D5H

SET1

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	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 ≤ +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	Interrupt return (IRET) in progress (when read)
1	Fast interrupt service routine in progress (when read)

.0 Bank Address Selection Flag (BA)

0	Bank 0 is selected (normal setting for KS88C01016/C01008/C01004/ C01116/C01108/C01104)
1	Invalid selection (bank 1 is not implemented)

IMR — Interrupt Mask Register DDH SET1

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; External Interrupts P0.7–P0.4

0	Disable (mask)
1	Enable (un-mask)

.6 Interrupt Level 6 (IRQ6) Enable Bit; External Interrupts P0.3–P0.0

0	Disable (mask)
1	Enable (un-mask)

.5 Not used for KS88C01016/C01008/C01004/C01116/C01108/C01104 .

.4 Interrupt Level 4 (IRQ4) Enable Bit; Counter A Interrupt

0	Disable (mask)
1	Enable (un-mask)

.3 - .2 Not used for KS88C01016/C01008/C01004/C01116/C01108/C01104 .

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

0	Disable (mask)
1	Enable (un-mask)

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

0	Disable (mask)
1	Enable (un-mask)

NOTES:

1. When an interrupt level is masked, any interrupt requests that may be issued are not recognized by the CPU.
2. Interrupt levels IRQ2,IRQ3 and IRQ5 are not used in the KS88C01016/C01008/C01004/C01116/C01108/C01104 interrupt structure.

IPH — Instruction Pointer (High Byte) DAH SET1

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 SET1

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 SET1

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
Bit Addressing	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, .3

Not used for KS88C01016/C01008/C01004/C01116/C01108/C01104

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

0	IRQ4
1	IRQ4

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

0	IRQ0 > IRQ1
1	IRQ1 > IRQ0

NOTE: The KS88C01016/C01008/C01004/C01116/C01108/C01104 interrupt structure uses only five levels: IRQ0, IRQ1, IRQ4, IRQ6–IRQ7. Because IRQ2, IRQ3, IRQ5 are not recognized, the interrupt subgroup B and group C settings (IPR.2,.3 and IPR.5) are not evaluated.

IRQ — Interrupt Request Register DCH SET1

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 **Level 7 (IRQ7) Request Pending Bit; External Interrupts P0.7–P0.4**

0	Not pending
1	Pending

.6 **Level 6 (IRQ6) Request Pending Bit; External Interrupts P0.3–P0.0**

0	Not pending
1	Pending

.5 Not used for KS88C01016/C01008/C01004/C01116/C01108/C01104 .

.4 **Level 4 (IRQ4) Request Pending Bit; Counter A Interrupt**

0	Not pending
1	Pending

.3 - .2 Not used for KS88C01016/C01008/C01004/C01116/C01108/C01104 .

.1 **Level 1 (IRQ1) Request Pending Bit; Timer 1 Match or Overflow**

0	Not pending
1	Pending

.0 **Level 0 (IRQ0) Request Pending Bit; Timer 0 Match or Overflow**

0	Not pending
1	Pending

NOTE: Interrupt level IRQ2, IRQ3 and IRQ5 is not used in the KS88C01016/C01008/C01004/C01116/C01108/C01104 interrupt structure.

P0CONH — Port 0 Control Register (High Byte) E8H SET1

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 P0.7/INT4 Mode Selection Bits

0	0	C-MOS input mode; interrupt on falling edges
0	1	C-MOS input mode; interrupt on rising and falling edges
1	0	Push-pull output mode
1	1	C-MOS input mode; interrupt on rising edges

.5 - .4 P0.6/INT4 Mode Selection Bits

0	0	C-MOS input mode; interrupt on falling edges
0	1	C-MOS input mode; interrupt on rising and falling edges
1	0	Push-pull output mode
1	1	C-MOS input mode; interrupt on rising edges

.3 - .2 P0.5/INT4 Mode Selection Bits

0	0	C-MOS input mode; interrupt on falling edges
0	1	C-MOS input mode; interrupt on rising and falling edges
1	0	Push-pull output mode
1	1	C-MOS input mode; interrupt on rising edges

.1 - .0 P0.4/INT4 Mode Selection Bits

0	0	C-MOS input mode; interrupt on falling edges
0	1	C-MOS input mode; interrupt on rising and falling edges
1	0	Push-pull output mode
1	1	C-MOS input mode; interrupt on rising edges

NOTES:

1. The INT4 external interrupts at the P0.7–P0.4 pins share the same interrupt level (IRQ7) and interrupt vector address (E8H).
2. You can assign pull-up resistors to individual port 0 pins by making the appropriate settings to the P0PUR register.

P0CONL — Port 0 Control Register (Low Byte) E9H SET1

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 P0.3/INT3 Mode Selection Bits

0	0	C-MOS input mode; interrupt on falling edges
0	1	C-MOS input mode; interrupt on rising and falling edges
1	0	Push-pull output mode
1	1	C-MOS input mode; interrupt on rising edges

.5 - .4 P0.2/INT2 Mode Selection Bits

0	0	C-MOS input mode; interrupt on falling edges
0	1	C-MOS input mode; interrupt on rising and falling edges
1	0	Push-pull output mode
1	1	C-MOS input mode; interrupt on rising edges

.3 - .2 P0.1/INT1 Mode Selection Bits

0	0	C-MOS input mode; interrupt on falling edges
0	1	C-MOS input mode; interrupt on rising and falling edges
1	0	Push-pull output mode
1	1	C-MOS input mode; interrupt on rising edges

.1 - .0 P0.0/INT0 Mode Selection Bits

0	0	C-MOS input mode; interrupt on falling edges
0	1	C-MOS input mode; interrupt on rising and falling edges
1	0	Push-pull output mode
1	1	C-MOS input mode; interrupt on rising edges

NOTES:

1. The INT3–INT0 external interrupts at P0.3–P0.0 are interrupt level IRQ6. Each interrupt has a separate vector address.
2. You can assign pull-up resistors to individual port 0 pins by making the appropriate settings to the P0PUR register.

P0INT — Port 0 External Interrupt Enable Register F1H SET1

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 P0.7 External Interrupt (INT4) Enable Bit

0	Disable interrupt
1	Enable interrupt

.6 P0.6 External Interrupt (INT4) Enable Bit

0	Disable interrupt
1	Enable interrupt

.5 P0.5 External Interrupt (INT4) Enable Bit

0	Disable interrupt
1	Enable interrupt

.4 P0.4 External Interrupt (INT4) Enable Bit

0	Disable interrupt
1	Enable interrupt

.3 P0.3 External Interrupt (INT3) Enable Bit

0	Disable interrupt
1	Enable interrupt

.2 P0.2 External Interrupt (INT2) Enable Bit

0	Disable interrupt
1	Enable interrupt

.1 P0.1 External Interrupt (INT1) Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 P0.0 External Interrupt (INT0) Enable Bit

0	Disable interrupt
1	Enable interrupt

P0PND — Port 0 External Interrupt Pending Register F2H SET1

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 P0.7 External Interrupt (INT4) Pending Flag (note)

0	No P0.7 external interrupt pending (when read)
1	P0.7 external interrupt is pending (when read)

.6 P0.6 External Interrupt (INT4) Pending Flag

0	No P0.6 external interrupt pending (when read)
1	P0.6 external interrupt is pending (when read)

.5 P0.5 External Interrupt (INT4) Pending Flag

0	No P0.5 external interrupt pending (when read)
1	P0.5 external interrupt is pending (when read)

.4 P0.4 External Interrupt (INT4) Pending Flag

0	No P0.4 external interrupt pending (when read)
1	P0.4 external interrupt is pending (when read)

.3 P0.3 External Interrupt (INT3) Pending Flag

0	No P0.3 external interrupt pending (when read)
1	P0.3 external interrupt is pending (when read)

.2 P0.2 External Interrupt (INT2) Pending Flag

0	No P0.2 external interrupt pending (when read)
1	P0.2 external interrupt is pending (when read)

.1 P0.1 External Interrupt (INT1) Pending Flag

0	No P0.1 external interrupt pending (when read)
1	P0.1 external interrupt is pending (when read)

.0 P0.0 External Interrupt (INT0) Pending Flag

0	No P0.0 external interrupt pending (when read)
1	P0.0 external interrupt is pending (when read)

NOTE: To clear an interrupt pending condition, write a "0" to the appropriate pending flag. Writing a "1" to an interrupt pending flag (POND.0–7) has no effect.

P0PUR — Port 0 Pull-Up Resistor Enable Register E7H SET1

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 P0.7 Pull-up Resistor Enable Bit

0	Disable pull-up resistor
1	Enable pull-up resistor

.6 P0.6 Pull-up Resistor Enable Bit

0	Disable pull-up resistor
1	Enable pull-up resistor

.5 P0.5 Pull-up Resistor Enable Bit

0	Disable pull-up resistor
1	Enable pull-up resistor

.4 P0.4 Pull-up Resistor Enable Bit

0	Disable pull-up resistor
1	Enable pull-up resistor

.3 P0.3 Pull-up Resistor Enable Bit

0	Disable pull-up resistor
1	Enable pull-up resistor

.2 P0.2 Pull-up Resistor Enable Bit

0	Disable pull-up resistor
1	Enable pull-up resistor

.1 P0.1 Pull-up Resistor Enable Bit

0	Disable pull-up resistor
1	Enable pull-up resistor

.0 P0.0 Pull-up Resistor Enable Bit

0	Disable pull-up resistor
1	Enable pull-up resistor

P1CONH — Port 1 Control Register (High Byte) EAH SET1

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 **P1.7 Mode Selection Bits**

0	0	C-MOS input mode
0	1	Open-drain output mode
1	0	Push-pull output mode
1	1	Invalid setting

.5 - .4 **P1.6 Mode Selection Bits**

0	0	C-MOS input mode
0	1	Open-drain output mode
1	0	Push-pull output mode
1	1	Invalid setting

.3 - .2 **P1.5 Mode Selection Bits**

0	0	C-MOS input mode
0	1	Open-drain output mode
1	0	Push-pull output mode
1	1	Invalid setting

.1 - .0 **P1.4 Mode Selection Bits**

0	0	C-MOS input mode
0	1	Open-drain output mode
1	0	Push-pull output mode
1	1	Invalid setting

P1CONL — Port 1 Control Register (Low Byte) EBH SET1

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 **P1.3 Mode Selection Bits**

0	0	C-MOS input mode
0	1	Open-drain output mode
1	0	Push-pull output mode
1	1	Invalid setting

.5 - .4 **P1.2 Mode Selection Bits**

0	0	C-MOS input mode
0	1	Open-drain output mode
1	0	Push-pull output mode
1	1	Invalid setting

.3 - .2 **P1.1 Mode Selection Bits**

0	0	C-MOS input mode
0	1	Open-drain output mode
1	0	Push-pull output mode
1	1	Invalid setting

.1 - .0 **P1.0 Mode Selection Bits**

0	0	C-MOS input mode
0	1	Open-drain output mode
1	0	Push-pull output mode
1	1	Invalid setting

P1PUR — Port 0 Pull-up Resistor Enable Register ECH SET1

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 P1.7 Pull-up Resistor Enable Bit

0	Disable pull-up resistor
1	Enable pull-up resistor

.6 P1.6 Pull-up Resistor Enable Bit

0	Disable pull-up resistor
1	Enable pull-up resistor

.5 P1.5 Pull-up Resistor Enable Bit

0	Disable pull-up resistor
1	Enable pull-up resistor

.4 P1.4 Pull-up Resistor Enable Bit

0	Disable pull-up resistor
1	Enable pull-up resistor

.3 P1.3 Pull-up Resistor Enable Bit

0	Disable pull-up resistor
1	Enable pull-up resistor

.2 P1.2 Pull-up Resistor Enable Bit

0	Disable pull-up resistor
1	Enable pull-up resistor

.1 P1.1 Pull-up Resistor Enable Bit

0	Disable pull-up resistor
1	Enable pull-up resistor

.0 P1.0 Pull-up Resistor Enable Bit

0	Disable pull-up resistor

1	Enable pull-up resistor

P2CON — Port 2 Control Register F0H SET1

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 P2.2 Mode Selection Bits

0	0	C-MOS input mode
0	1	Open-drain output mode
1	0	Push-pull output mode
1	1	C-MOS input with pull up mode

.5 - .4 P2.1 Mode Selection Bits

0	0	C-MOS input mode
0	1	Open-drain output mode
1	0	Push-pull output mode
1	1	C-MOS input with pull up mode

.3 - .2 P2.0 Mode Selection Bits

0	0	C-MOS input mode
0	1	Open-drain output mode
1	0	Push-pull output mode
1	1	C-MOS input with pull up mode

.1 P2.1 Alternative Function Selection Bits

0	Normal I/O function
0	REM/T0CK function

.0 P2.0 Alternative Function Selection Bits

0	Normal I/O function
0	T0PWN function

PP — Register Page Pointer DFH SET1

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 Bits

0	0	0	0	Source: page 0 (note)

NOTE: In the KS88C01016/C01008/C01004/C01116/C01108/C01104 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 SET1

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 mode only							

.7 - .3 Destination Register Page Selection Bits

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 KS88C01016/C01008/C01004/C01116/C01108/C01104

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 mode 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 KS88C01016/C01008/C01004/C01116/C01108/C01104

SPL — Stack Pointer (Low Byte) D9H SET1

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 SP value is undefined following a reset.

STOPCON — Stop Control Register FBH SET1

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

.7 - .0 **Stop Control Register enable bits**

1	0	1	0	0	1	0	1	Enable STOPCON

NOTES:
1. To get into STOP mode, stop control register must be enabled just before STOP instruction.
2. When STOP mode is released, stop control register (STOPCON) value is cleared automatically.
3. It is prohibited to write another value into STOPCON.

SYM — System Mode Register DEH SET1

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

.7 Tri-State External Interface Control Bit (1)

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

.6 - .5

Not used for KS88C01016/C01008/C01004/C01116/C01108/C01104

.4 - .2 Fast Interrupt Level Selection Bits (2)

0	0	0	IRQ0
0	0	1	IRQ1
0	1	0	Not used for
0	1	1	KS88C01016/C01008/C01004/C01116/C01108/C01104
1	0	0	
1	0	1	
1	1	0	IRQ6
1	1	1	IRQ7

.1 Fast Interrupt Enable Bit (3)

0	Disable fast interrupt processing
1	Enable fast interrupt processing

.0 Global Interrupt Enable Bit (4)

0	Disable global interrupt processing
1	Enable global interrupt processing

NOTES:

1. Because an external interface is not implemented for the KS88C01016/C01008/C01004/C01116/C01108/C01104 , SYM.7 must always be "0".
2. You can select only one interrupt level at a time for fast interrupt processing.
3. Setting SYM.1 to "1" enables fast interrupt processing for the interrupt level currently selected by SYM.2–SYM.4.
4. 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 SET1

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 Timer 0 Input Clock Selection Bits

0	0	$f_{OSC}/4096$
0	1	$f_{OSC}/256$
1	0	$f_{OSC}/8$
1	1	External clock input (at the T0CK pin, P2.1)

.5 - .4 Timer 0 Operating Mode Selection Bits

0	0	Interval timer mode (counter cleared by match signal)
0	1	Overflow mode(OVF interrupt can occur)
1	0	Overflow mode(OVF interrupt can occur)
1	1	PWM mode (OVF interrupt can occur)

.3 Timer 0 Counter Clear Bit

0	No effect (when write)
1	*Clear T0 counter, T0CNT (when write)*

.2 Timer 0 Overflow Interrupt Enable Bit (note)

0	Disable T0 overflow interrupt
1	Enable T0 overflow interrupt

.1 Timer 0 Match Interrupt Enable Bit

0	Disable T0 match interrupt
1	Enable T0 match interrupt

.0 Timer 0 Match Interrupt Pending Flag

0	No T0 match interrupt pending (when read)
0	*Clear T0 match interrupt pending condition (when write)*
1	T0 match interrupt is pending (when read)
1	*No effect (when write)*

NOTE: A timer 0 overflow interrupt pending condition is automatically cleared by hardware. However, the timer 0

match/ capture interrupt, IRQ0, vector FCH, must be cleared by the interrupt service routine.

T1CON — Timer 1 Control Register FAH SET1

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 Timer 1 Input Clock Selection Bits

0	0	$f_{OSC}/4$
0	1	$f_{OSC}/8$
1	0	$f_{OSC}/16$
1	1	Internal clock (counter A flip-flop, T-FF)

.5 - .4 Timer 1 Operating Mode Selection Bits

0	0	Interval timer mode (counter cleared by match signal)
0	1	Overflow mode(OVF interrupt can occur)
1	0	Overflow mode(OVF interrupt can occur)
1	1	Overflow mode(OVF interrupt can occur)

.3 Timer 1 Counter Clear Bit

0	No effect (when write)
1	*Clear T1 counter, T1CNT (when write)*

.2 Timer 1 Overflow Interrupt Enable Bit (note)

0	Disable T1 overflow interrupt
1	Enable T1 overflow interrupt

.1 Timer 1 Match/Capture Interrupt Enable Bit

0	Disable T1 match interrupt
1	Enable T1 match interrupt

.0 Timer 1 Match/Capture Interrupt Pending Flag

0	No T1 match interrupt pending (when read)
0	*Clear T1 match interrupt pending condition (when write)*
1	T1 match interrupt is pending (when read)
1	*No effect (when write)*

NOTE: A timer 1 overflow interrupt pending condition is automatically cleared by hardware. However, the timer 1 match/ capture interrupt, IRQ1, vector F6H, must be cleared by the interrupt service routine.

5 INTERRUPT STRUCTURE

OVERVIEW

The KS88-series interrupt structure has three basic components: levels, vectors, and sources. The SAM87 CPU recognizes up to 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. A vector address can be assigned to 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 possible 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 KS88C01016/C01008/C01004/C01116/C01108/C01104 interrupt structure recognizes five interrupt levels.

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 is always much smaller.) If an interrupt level has more than one vector address, the vector priorities are set in hardware. The KS88C01016/C01008/C01004/C01116/C01108/C01104 uses ten vectors. One vector address is shared by four interrupt sources.

Sources

A source is any peripheral that generates an interrupt. A source can be an external pin or a counter overflow, for example. Each vector can have several interrupt sources. In the KS88C01016/C01008/C01004/C01116/ C01108/C01104 interrupt structure, there are thirteen possible interrupt sources.

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

INTERRUPT TYPES

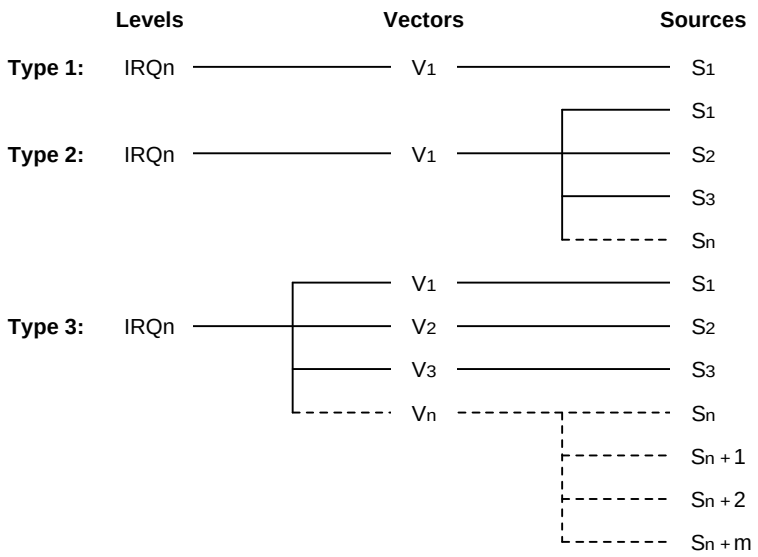
The three components of the KS88 interrupt structure described above — levels, vectors, and sources — are combined to determine the interrupt structure of an individual device and to make full use of its available interrupt logic. There are three possible combinations of interrupt structure components, 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 KS88C01016/C01008/C01004/C01116/C01108/C01104 microcontroller, all three interrupt types are implemented.



NOTES:
1. The number of S n and Vn value is expandable.
2. In the KS88C01016/C01008/C01004/C01116/C01104/C01108 implementation, interrupt types 1, 2, and 3 is used.

Figure 5-1. KS88-Series Interrupt Types

KS88C01016/C01008/C01004/C01116/C01108/C01104 INTERRUPT STRUCTURE

The KS88P01016/C01008/C01004/C01116/C01108/C01104 microcontroller supports two kinds interrupt structure

— Vectored Interrupt

— Non vectored interrupt (Reset interrupt): INTR

The KS88C01016/C01008/C01004/C01116/C01108/C01104 microcontroller supports thirteen interrupt sources. Nine of the interrupt sources have a corresponding interrupt vector address; the remaining four interrupt sources share the same vector address. Five interrupt levels are recognized by the CPU in this device-specific interrupt structure, as 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 fixed in hardware.)

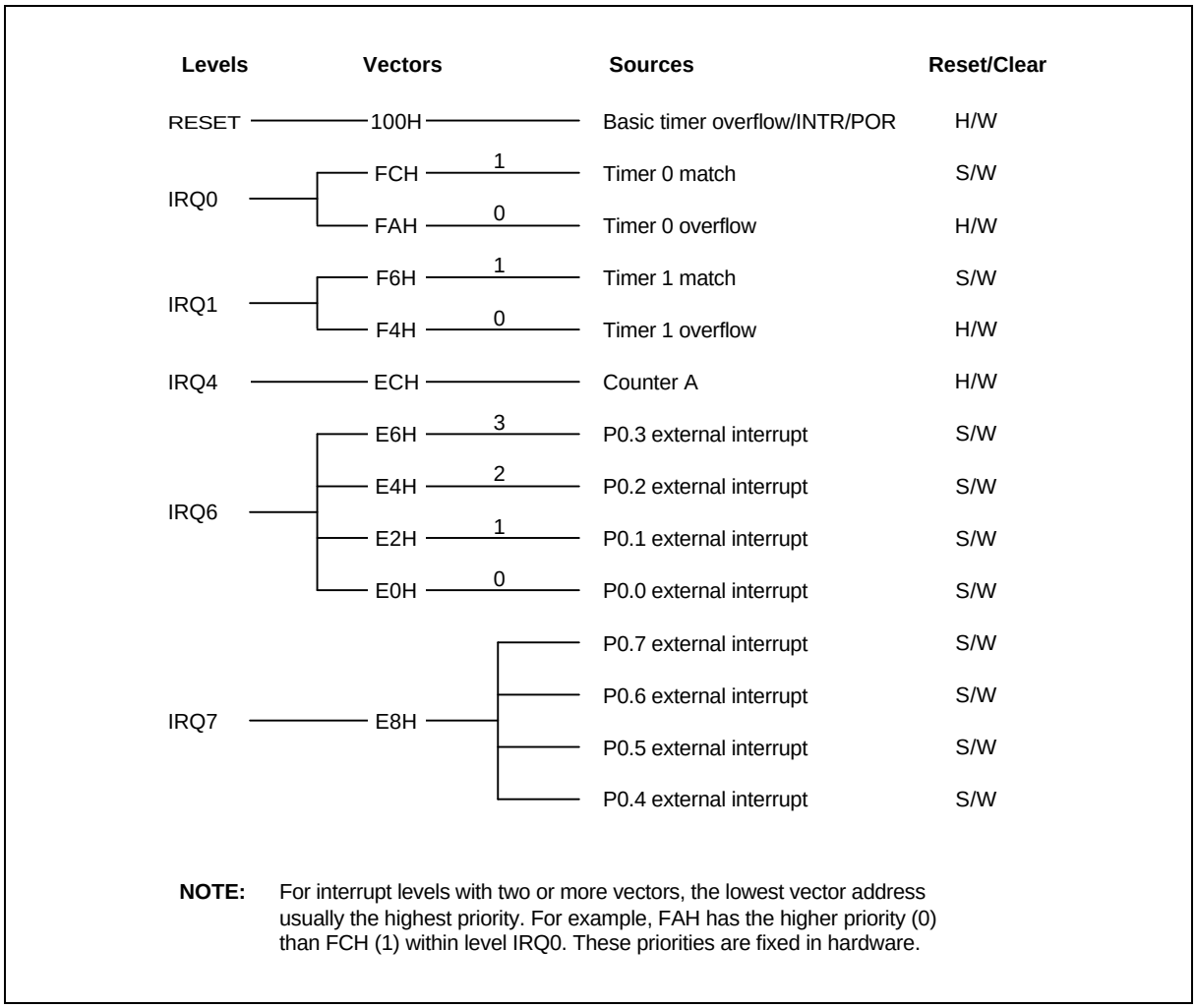
When the CPU grants an interrupt request, interrupt processing starts: All other interrupts are disabled and the program counter value and state flags are pushed to stack. The starting address of the service routine is fetched from the appropriate vector address (plus the next 8-bit value to concatenate the full 16-bit address) and the service routine is executed. The KS88C01016/C01008/C01004/C01116/C01108/C01104 microcontroller supports non vectored interrupt - Interrupt with Reset(INTR) - to occur interrupt with system reset. The Interrupt with Reset(INTR) has nothing to do with interrupt levels, vectors and the registers that are related to interrupt setting. *It occurs only according to the "P0" during " STOP " regardless any other things.* Namely, only when a falling/rising edge occurs at any pin of Port 0 during STOP status, this INTR and a system reset occurs even though SYM.0 is "0"(Disable interrupt). But it dose not occurs while the oscillation - "IDLE" or "OPERATING" status- even though a falling/rising edge occurs at port 0.

Following is the sequence that occurs Interrupt with Reset(INTR).

1. The oscillation status is "freeze" : STOP mode
2. A falling/rising edge is detected to any pin of Port 0.
3. INTR occurs and it makes system reset.
4. STOP mode is released by this system reset.

NOTE

Because H/W reset occurs whenever INTR occurs. A user should aware of the each ports, system register, control register etc."



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

Figure 5-2. KS88C01016/C01008/C01004/C01116/C01108/C01104 Interrupt Structure

INTERRUPT VECTOR ADDRESSES

All interrupt vector addresses for the KS88C01016/C01008/C01004/C01116/C01108/C01104 interrupt structure are stored in the vector address area of the internal program memory ROM, 00H-FFH. You can allocate unused locations in the vector address area as normal program memory. If you do so, please be careful not to overwrite any of the stored vector addresses. (Table 5-2 lists all vector addresses.)

The program reset address in the ROM is 0100H.

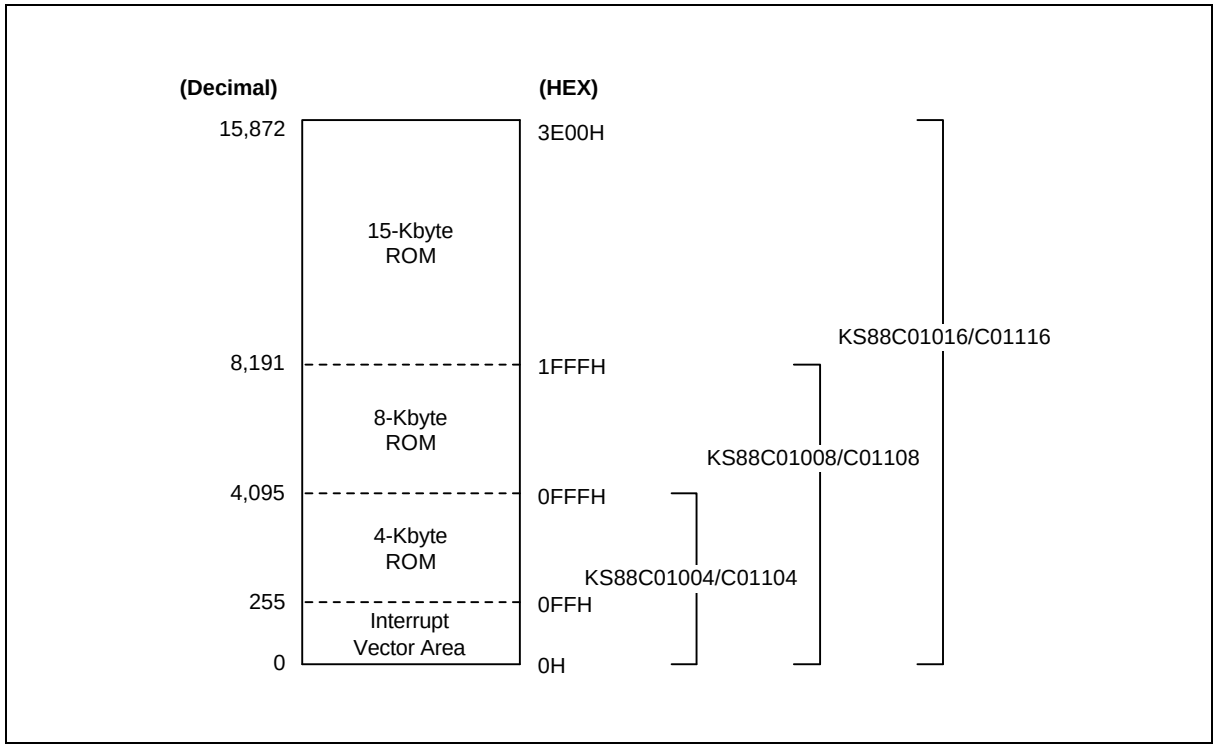


Figure 5-3. ROM Vector Address Area

Table 5-1. KS88C01016/C01008/C01004/C01116/C01108/C01104 Interrupt Vectors

Vector Address		Interrupt Source	Request		Reset/Clear	
Decimal Value	Hex Value		Interrupt Level	Priority in Level	H/W	S/W
254	**100H**	Basic timer overflow	RESET	-	√	
252	FCH	Timer 0 (match)	IRQ0	1		√
250	FAH	Timer 0 overflow		0	√	
246	F6H	Timer 1 (match)	IRQ1	1		√
244	F4H	Timer 1 overflow		0	√	
236	ECH	Counter A	IRQ4	–	√	
232	E8H	P0.7 external interrupt	IRQ7	–		√
232	E8H	P0.6 external interrupt		–		√
232	E8H	P0.5 external interrupt		–		√
232	E8H	P0.4 external interrupt		–		√
230	E6H	P0.3 external interrupt	IRQ6	3		√
228	E4H	P0.2 external interrupt		2		√
226	E2H	P0.1 external interrupt		1		√
224	E0H	P0.0 external interrupt		0		√

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 fixed in hardware.

ENABLE/DISABLE INTERRUPT INSTRUCTIONS (EI, DI)

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

NOTE

The system initialization routine that is executed following a reset must always contain an EI instruction to globally enable the interrupt structure.

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

SYSTEM-LEVEL INTERRUPT CONTROL REGISTERS

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

— The interrupt mask register, IMR, enables (un-masks) or disables (masks) interrupt levels.

— The interrupt priority register, IPR, controls the relative priorities of interrupt levels.

— The interrupt request register, IRQ, contains interrupt pending flags for each interrupt level (as opposed to each interrupt source).

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

Table 5-2. Interrupt Control Register Overview

Control Register	ID	R/W	Function Description
Interrupt mask register	IMR	R/W	Bit settings in the IMR register enable or disable interrupt processing for each of the five interrupt levels: IRQ0, IRQ1, IRQ4, and IRQ6–IRQ7.
Interrupt priority register	IPR	R/W	Controls the relative processing priorities of the interrupt levels. The five levels of the KS88C01016/C01008/C01004/ C01116/C01108/C01104 are organized into three groups: A, B, and C. Group A is IRQ0 and IRQ1, group B is IRQ4, and group C is IRQ6, and IRQ7.
Interrupt request register	IRQ	R	This register contains a request pending bit for each interrupt level.
System mode register	SYM	R/W	Dynamic global interrupt processing enable/ disable, fast interrupt processing, and external interface control (An external memory interface is not implemented in the KS88C01016/C01008/C01004/C01116/C01108/C01104 microcontroller).

INTERRUPT PROCESSING CONTROL POINTS

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

— Global interrupt enable and disable (by EI and DI instructions or by direct manipulation of SYM.0)

— Interrupt level enable/disable settings (IMR register)

— Interrupt level priority settings (IPR register)

— Interrupt source enable/disable settings in the corresponding peripheral control registers

NOTE

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

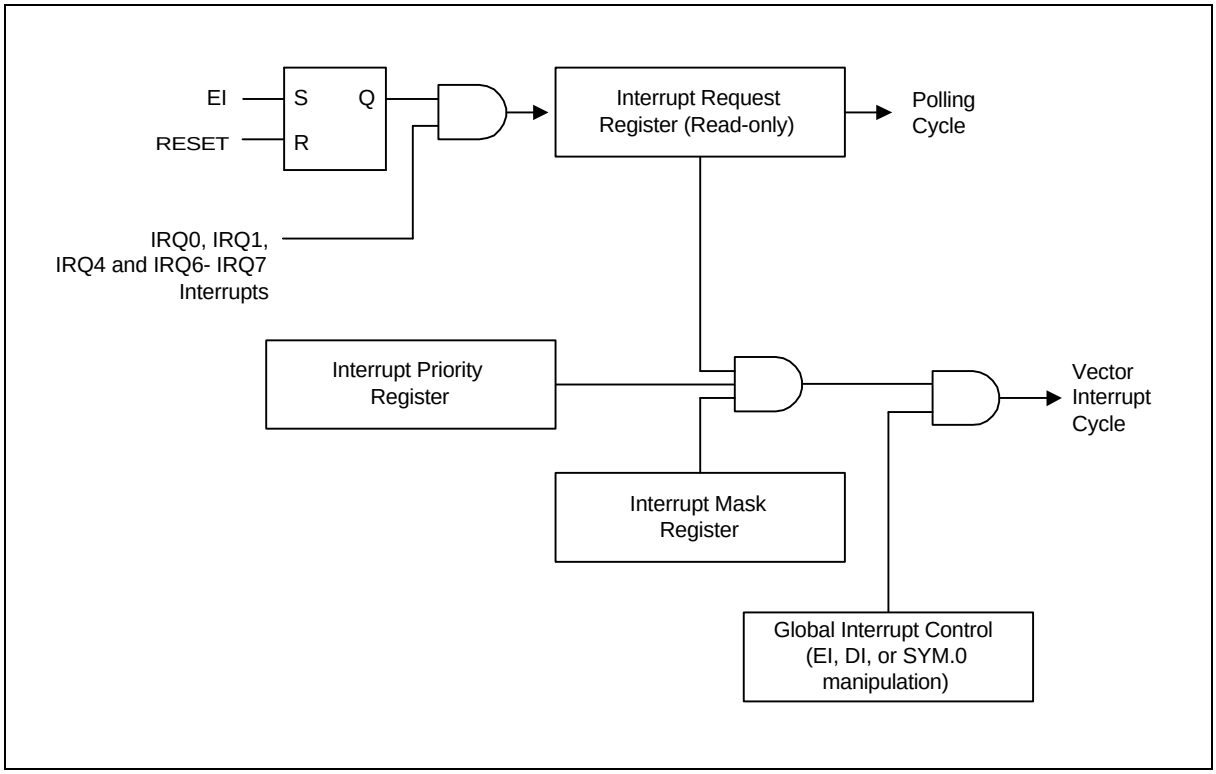


Figure 5-4. Interrupt Function Diagram

PERIPHERAL INTERRUPT CONTROL REGISTERS

For each interrupt source there is one or more corresponding peripheral control registers that let you control the interrupt generated by that peripheral (see Table 5-3).

Table 5-3. Interrupt Source Control and Data Registers

Interrupt Source	Interrupt Level	Register(s)	Location(s) in Set 1
Timer 0 match or timer 0 overflow	IRQ0	T0CON (note) T0DATA	D2H D1H
Timer 1 match or timer 1 overflow	IRQ1	T1CON (note) T1DATAH, T1DATAL	FAH F8H, F9H
Counter A	IRQ4	CACON CADATAH, CADATAL	F3H F4H, F5H
P0.7 external interrupt P0.6 external interrupt P0.5 external interrupt P0.4 external interrupt	IRQ7	P0CONH P0INT P0PND	E8H F1H F2H
P0.3 external interrupt P0.2 external interrupt P0.1 external interrupt P0.0 external interrupt	IRQ6	P0CONL P0INT P0PND	E9H F1H F2H

NOTE: Because the timer 0 and timer 1 overflow interrupts are cleared by hardware, the T0CON and T1CON registers control only the enable/disable functions. The T0CON and T1CON registers contain enable/disable and pending bits for the timer 0 and timer 1 match interrupts, respectively.

SYSTEM MODE REGISTER (SYM)

The system mode register, SYM (set 1, DEH), is used to globally enable and disable interrupt processing and to control fast interrupt processing (see Figure 5-5).

A reset clears SYM.7, SYM.1, and SYM.0 to "0". The 3-bit value for fast interrupt level selection, SYM.4–SYM.2, is undetermined.

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

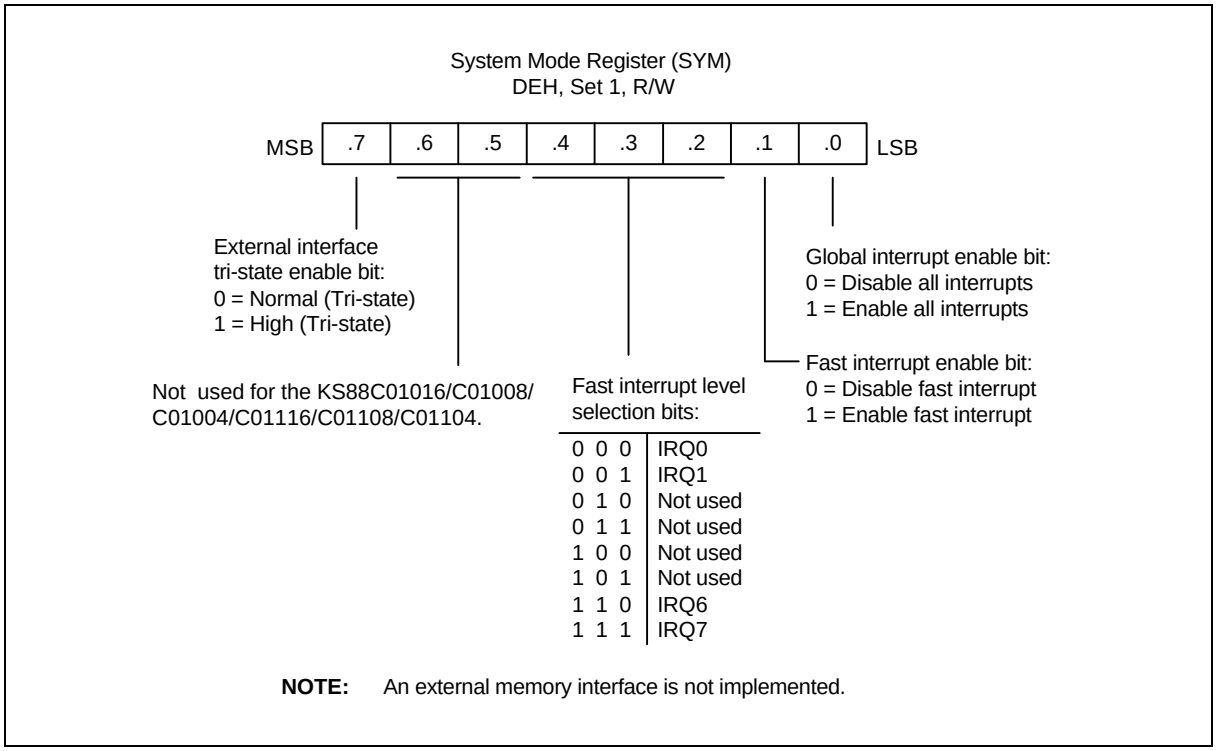


Figure 5-5. System Mode Register (SYM)

INTERRUPT MASK REGISTER (IMR)

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

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

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

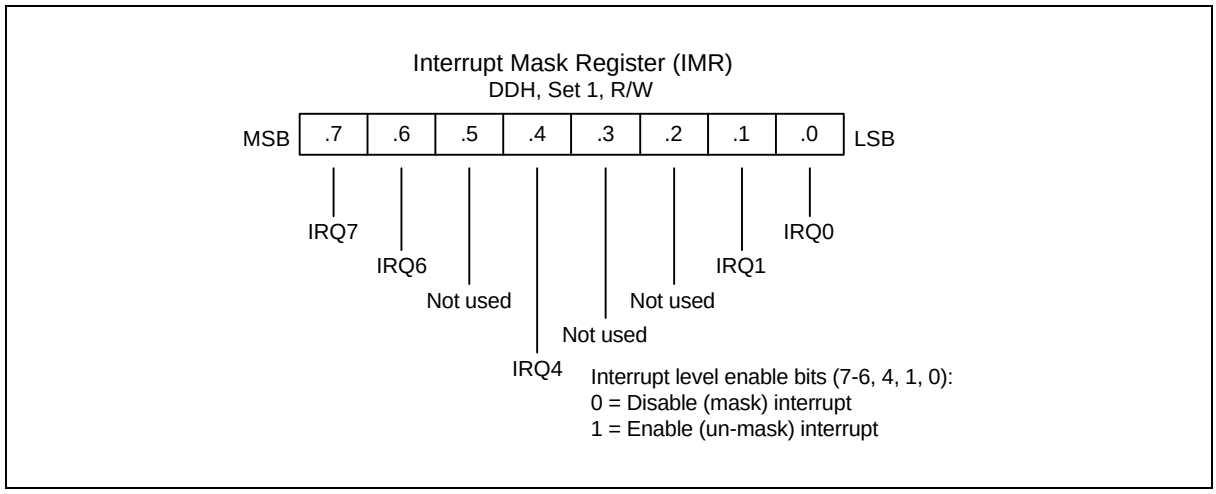


Figure 5-6. Interrupt Mask Register (IMR)

INTERRUPT PRIORITY REGISTER (IPR)

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

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

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

Group A	IRQ0, IRQ1
Group B	IRQ4
Group C	IRQ6, IRQ7

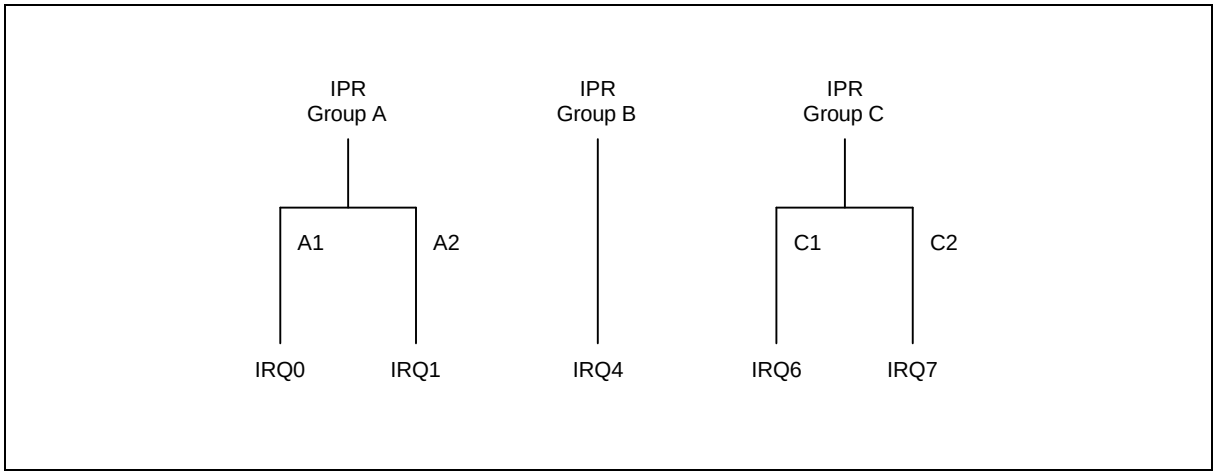


Figure 5-7. Interrupt Request Priority Groups

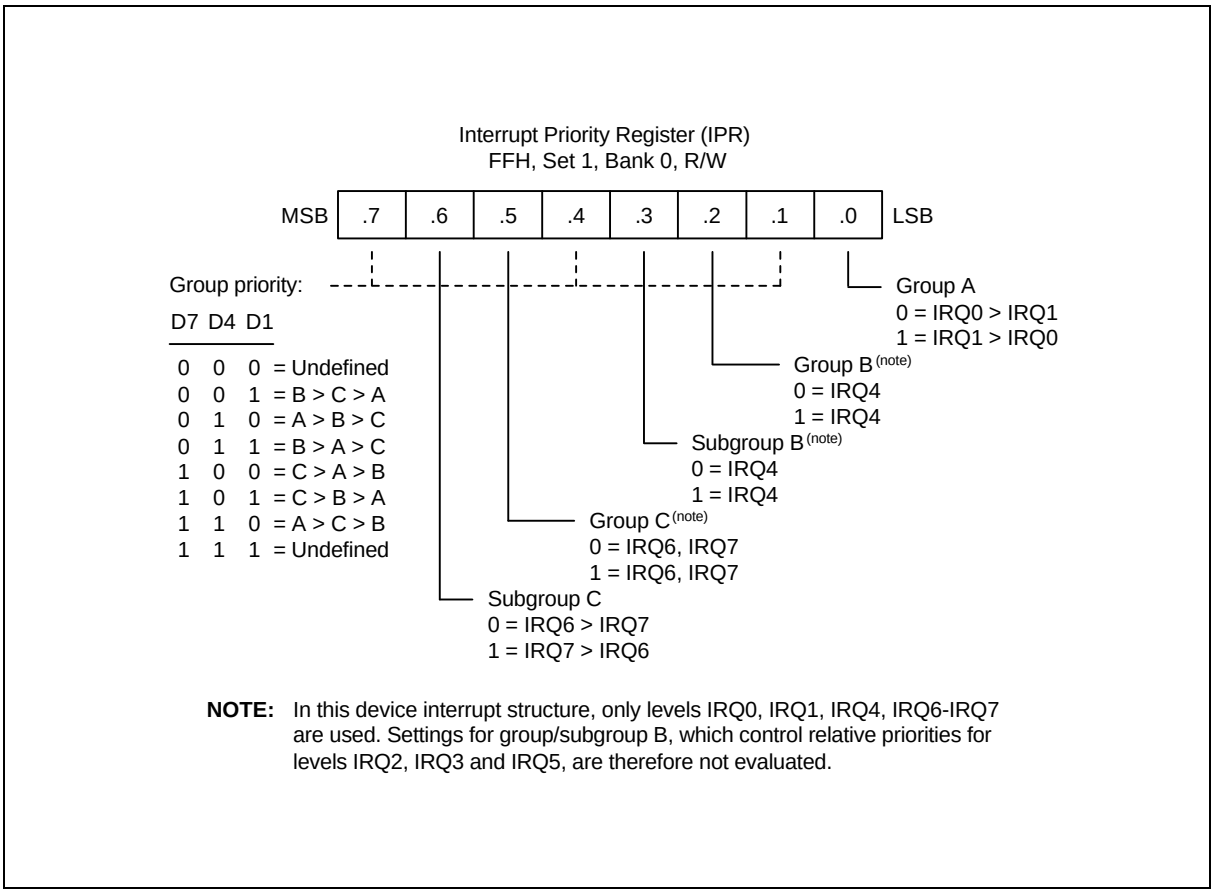
As you can see in Figure 5-8, IPR.7, IPR.4, and IPR.1 control the relative priority of interrupt groups A, B, and C. For example, the setting '001B' for these bits would select the group relationship B > C > A; the setting '101B' would select the relationship C > B > A.

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

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

— Interrupt group B has a subgroup to provide an additional priority relationship between for interrupt levels 2, 3, and 4. IPR.3 defines the possible subgroup B relationships. IPR.2 controls interrupt group B. In the KS88C01016/C01008/C01004/C01116/C01108/C01104 implementation, interrupt levels 2 and 3 are not used. Therefore, IPR.2 and IPR.3 settings are not evaluated, as IRQ4 is the only remaining level in the group.

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



NOTE: In this device interrupt structure, only levels IRQ0, IRQ1, IRQ4, IRQ6-IRQ7 are used. Settings for group/subgroup B, which control relative priorities for levels IRQ2, IRQ3 and IRQ5, are therefore not evaluated.

Figure 5-8. Interrupt Priority Register (IPR)

INTERRUPT REQUEST REGISTER (IRQ)

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

IRQ bit values are read-only addressable using Register addressing mode. You can read (test) the contents of the IRQ register at any time using bit or byte addressing to determine the current interrupt request status of specific interrupt levels. After a reset, all IRQ status bits are cleared to "0".

You can poll IRQ register values even if a DI instruction has been executed (that is, if global interrupt processing is disabled). If an interrupt occurs while the interrupt structure is disabled, the CPU will not service it. You can, however, still detect the interrupt request by polling the IRQ register. In this way, you can determine which events occurred while the interrupt structure was globally disabled.

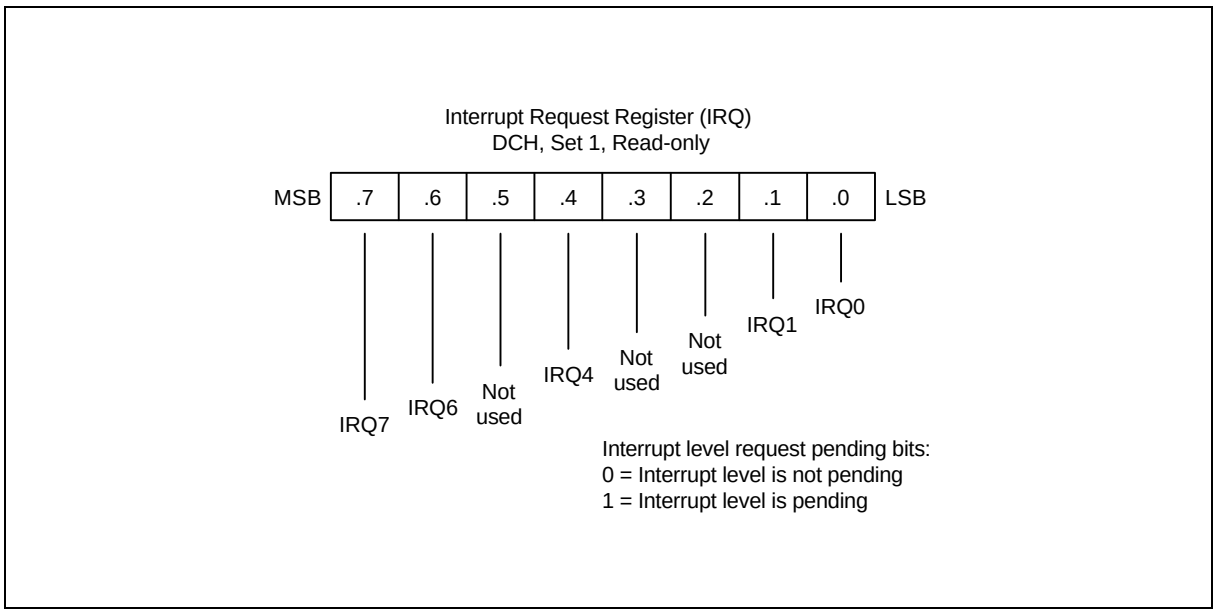


Figure 5-9. Interrupt Request Register (IRQ)

INTERRUPT PENDING FUNCTION TYPES

Overview

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

Pending Bits Cleared Automatically by Hardware

For interrupt pending bits that are cleared automatically by hardware, interrupt logic sets the corresponding pending bit to "1" when a request occurs. It then issues an IRQ pulse to inform the CPU that an interrupt is waiting to be serviced. The CPU acknowledges the interrupt source by sending an 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 application software.

In the KS88C01016/C01008/C01004/C01116/C01108/C01104 interrupt structure, the timer 0 and timer 1 overflow interrupts (IRQ0 and IRQ1), and the counter A interrupt (IRQ4) belong to this category of interrupts whose pending condition is cleared automatically by hardware.

Pending Bits Cleared by the Service Routine

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

In the KS88C01016/C01008/C01004/C01116/C01108/C01104 interrupt structure, pending conditions for all interrupt sources *except* the timer 0 and timer 1 overflow interrupts and the counter A borrow interrupt, must be cleared by the interrupt service routine.

INTERRUPT SOURCE POLLING SEQUENCE

The interrupt request polling and servicing sequence is as follows:

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

INTERRUPT SERVICE ROUTINES

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

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

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

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

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

GENERATING INTERRUPT VECTOR ADDRESSES

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

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

NOTE

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

NESTING OF VECTORED INTERRUPTS

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

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

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

INSTRUCTION POINTER (IP)

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

FAST INTERRUPT PROCESSING

The feature called *fast interrupt processing* lets you specify that an interrupt within a given level be completed in approximately six clock cycles instead of the usual 16 clock cycles. To select a specific interrupt level for fast interrupt processing, you write the appropriate 3-bit value to SYM.4–SYM.2. Then, to enable fast interrupt processing for the selected level, you set SYM.1 to "1".

FAST INTERRUPT PROCESSING (Continued)

Two other system registers support fast interrupt processing:

— The instruction pointer (IP) contains the starting address of the service routine (and is later used to swap the program counter values), and

— When a fast interrupt occurs, the contents of the FLAGS register is stored in an unmapped, dedicated register called FLAGS' ("FLAGS prime").

NOTE

For the KS88C01016/C01008/C01004/C01116/C01108/C01104 microcontroller, the service routine for any one of the five interrupt levels: IRQ0, IRQ1, IRQ4 or IRQ6–IRQ7, can be selected for fast interrupt processing.

Procedure for Initiating Fast Interrupts

To initiate fast interrupt processing, follow these steps:

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

Fast Interrupt Service Routine

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

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

Relationship to Interrupt Pending Bit Types

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

Programming Guidelines

Remember that the only way to enable/disable a fast interrupt is to set/clear the fast interrupt enable bit in the SYM register, SYM.1. Executing an EI or DI instruction globally enables or disables all interrupt processing, including fast interrupts. If you use fast interrupts, remember to load the IP with a new start address when the fast interrupt service routine ends.

7 CLOCK CIRCUITS

OVERVIEW

The clock frequency generated for the KS88C01016/C01008/C01004/C01116/C01108/C01104 by an external crystal, or supplied by an external clock source, can range from 1MHz to 8 MHz. The maximum CPU clock frequency, as determined by CLKCON register settings, is 8 MHz. The X_{IN} and X_{OUT} pins connect the external oscillator or clock source to the on-chip
clock circuit.

SYSTEM CLOCK CIRCUIT

The system clock circuit has the following components:

— External crystal or ceramic resonator oscillation source (or an external clock)
— 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

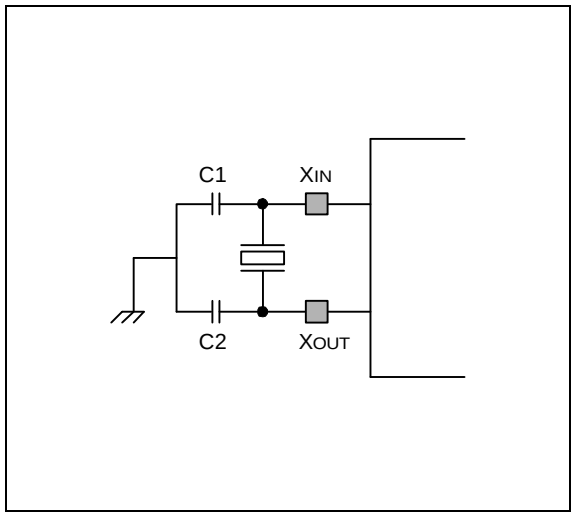


Figure 7-1. Main Oscillator Circuit (External Crystal or Ceramic Resonator)

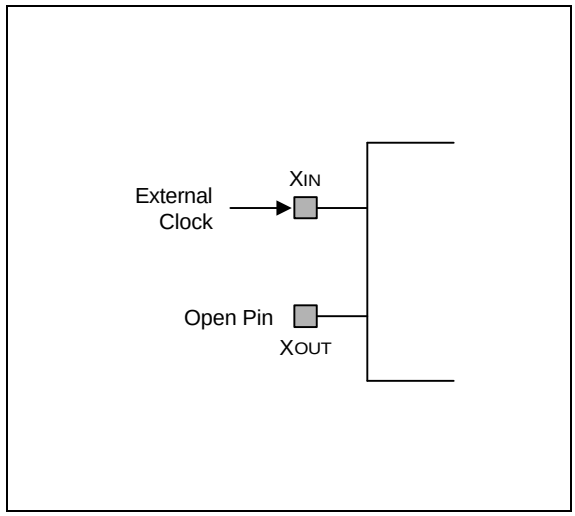


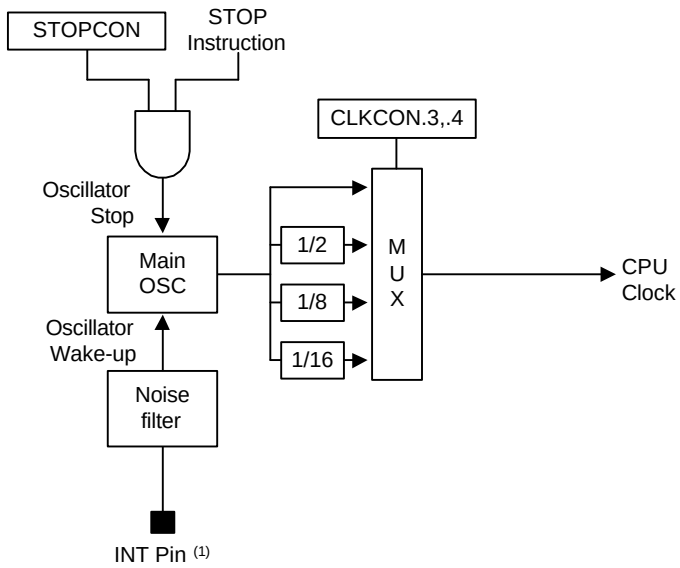
Figure 7-2. External Clock Circuit

CLOCK STATUS DURING POWER-DOWN MODES

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

— In Stop mode, the main oscillator is halted. Stop mode is released, and the oscillator started, by Power On Reset operation or by a non-vectored interrupt - interrupt with reset (INTR). To enter the Stop mode, STOPCON (STOP Control register) has to be loaded with value, #0A5H before STOP instruction execution. After recovering from the Stop mode by reset or interrupt, STOPCON register is automatically cleared.

— In Idle mode, the internal clock signal is gated away from the CPU, but continues to be supplied to the interrupt structure, timer 0, and counter A. Idle mode is released by a reset or by an interrupt (external or internally generated).



NOTES:

1. An external interrupt with an RC-delay noise filter (for KS88C01016/ C01008/C01004/C01116/C01108/C01104, INT0-4) is fiexed to release Stop mode and "wake up" the main oscillator.
2. Because the KS88C01016/C01008/C01004/C01116/C01108/C01104 has no subsystem clock, the 3-bit CLKCON signature code (CLKCON.2-CLKCON.0) is no meaning.

Figure 7-3. System Clock Circuit Diagram

SYSTEM CLOCK CONTROL REGISTER (CLKCON)

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

— Oscillator frequency divide-by value

CLKCON register settings control 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. In KS88C01016/C01008/C01004/C01116/C01108/C01104 , this bit is not valid any more. Actually bit 7, 6, 5, 2, 1, and 0 are no meaning in KS88C01016/C01008/C01004/C01116/C01108/C01104 .

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

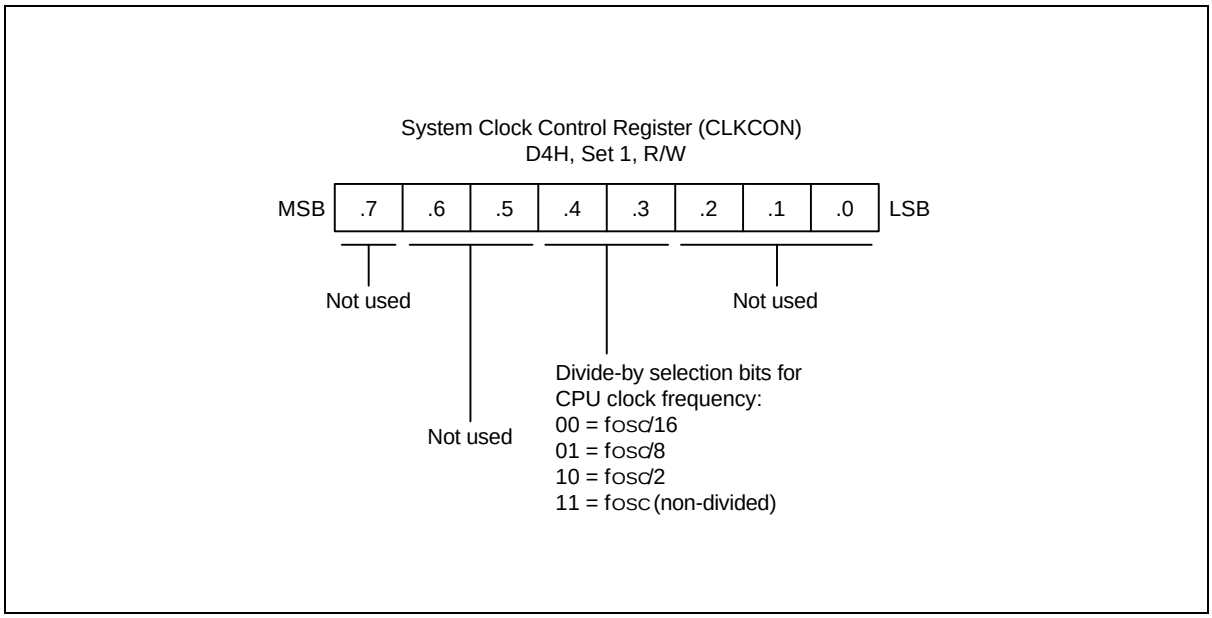


Figure 7-4. System Clock Control Register (CLKCON)

8 RESET and POWER-DOWN

SYSTEM RESET

KS88C01016/C01008/C01004/C01116/C01108/C01104 has four different system reset sources as followings:

— Low Voltage Detect (LVD)
— Internal POR circuit
— INTR (Interrupt with RESET)
— Basic Timer (Watchdog timer)

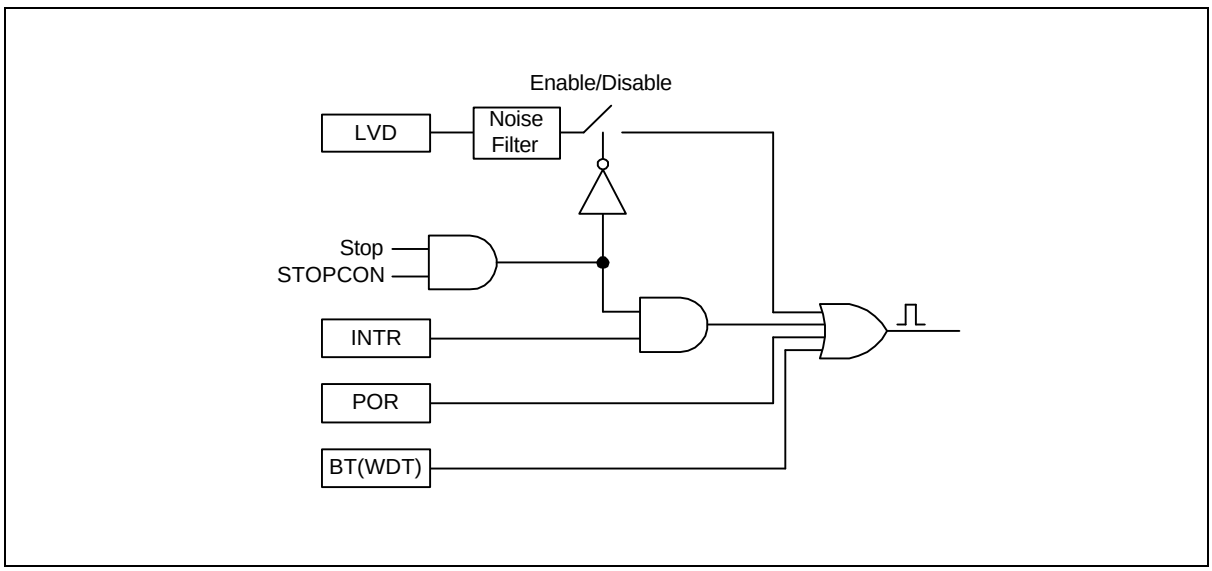


Figure 8-1. Reset Block Diagram

LVD RESET

The Low Voltage detect circuit is built on the KS88C01016/C01008/C01004/C01116/C01108/C01104 product for system reset not in stop mode. When the operating status is not stop mode it detects a slope of V_{DD} by comparing the voltage at V_{DD} with V_{LVD} (Low level Detect Voltage). The reset pulse is generated by the rising slope of V_{DD}. While the voltage at V_{DD} is rising up and passing V_{LVD}, the reset pulse is occurred at the moment "V_{DD} >= V_{LVD} ". This function is disabled when the operating state is "stop mode" to reduce the current consumption under 1uA instead of 6uA.

INTERRUPT WITH RESET(INTR)

A non vectored interrupt called Interrupt with reset (INTR) is built in KS88C01016/C01008/C01004/C01116/ C01108/C01104 to release stop status with system reset. When a falling/rising edge occurs at Port 0 during stop mode, INTR signal is generated and it makes the system reset pulse. An INTR signal is generated relating to interaction between Port 0 and operating status. It is enabled by STOP status and occurs by falling/rising edge at port0. So only when the chip status is "STOP", it is available. If the operating status is not stop status INTR does not occurs.

NOTE

This INTR is supplementary function to make system reset for an application which is using " stop mode" like remote controller. If an application which is not using "stop mode" , INTR function can be discarded.

WATCHDOG TIMER RESET

The KS88C01016/C01008/C01004/C01116/C01108/C01104 build a watch-dog timer that can recover to normal operation from abnormal function. Watchdog timer generates a system reset signal if not clearing a BT-Basic Counter within a specific time by program. System reset can return to the proper operation of chip.

POWER-ON RESET(POR)

The power-on reset circuit is built on the KS88C01016/C01008/C01004/C01116/C01108/C01104 product. During a power-on reset, the voltage at V_{DD} goes to High level and the Schmitt trigger input of POR circuit is forced to Low level and then to High level. The power-on reset circuit makes a reset signal whenever the power supply voltage is powering-up and the Schmitt trigger input senses the Low level. This on-chip POR circuit consists of an internal resistor, an internal capacitor, and a Schmitt trigger input transistor.

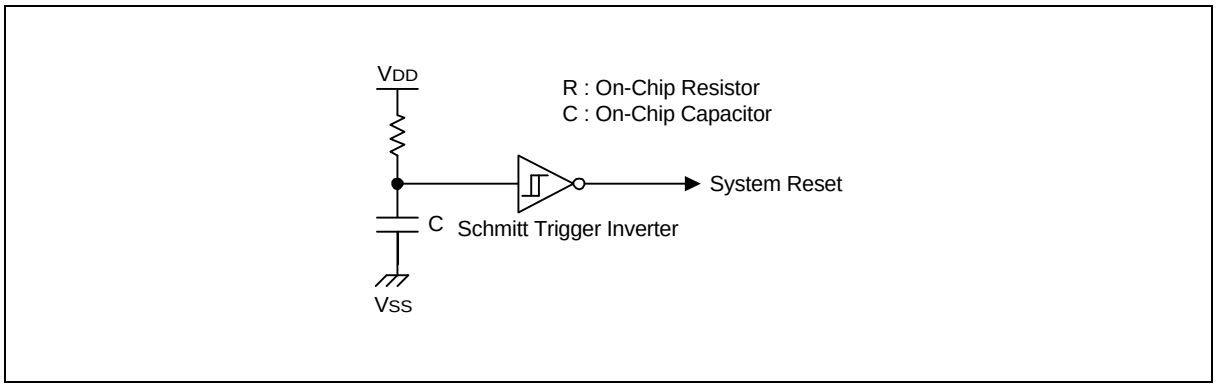


Figure 8-2. Power-on Reset Circuit

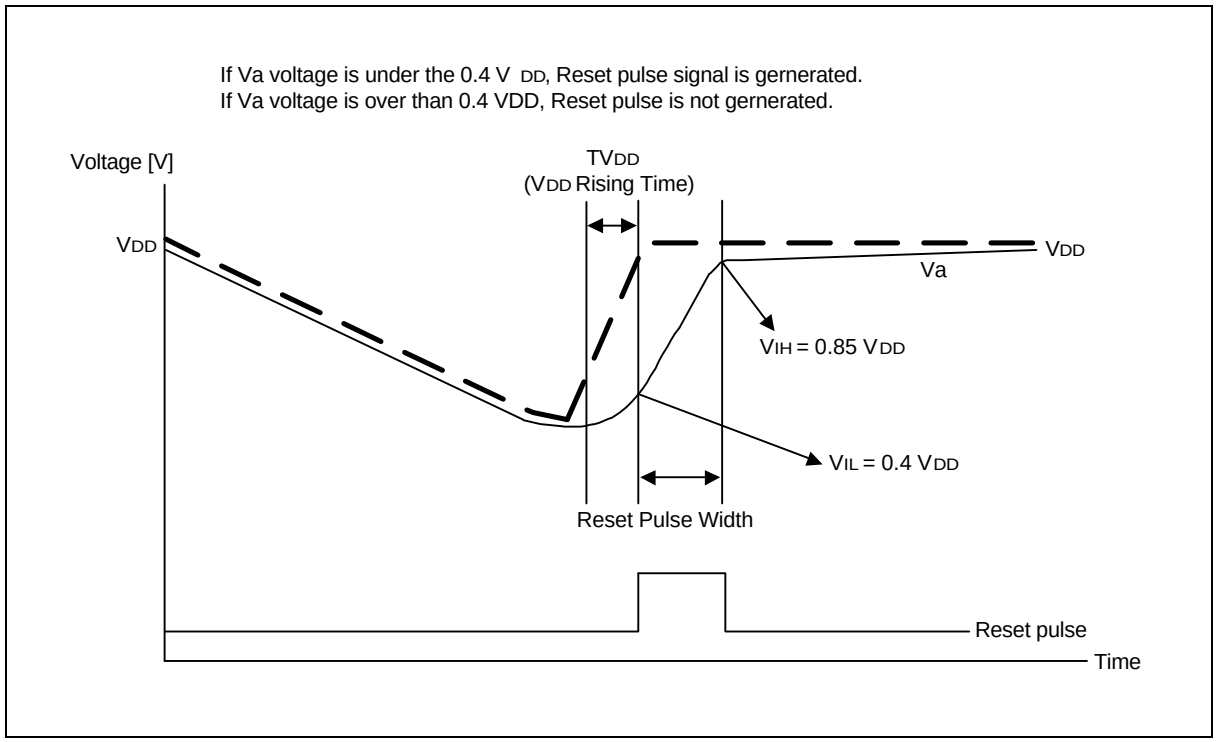


Figure 8-3. Timing Diagram for Power-on Reset Circuit

SYSTEM RESET OPERATION

System reset starts the oscillation circuit, synchronize chip operation with CPU clock, and initialize the internal CPU and peripheral modules. This procedure brings the KS88C01016/C01008/C01004/C01116/C01108/C01104 into a known operating status. To allow time for internal CPU clock oscillation to stabilize, the reset pulse generator must be held to active level for a minimum time interval after the power supply comes within tolerance. The minimum required reset operation for a oscillation stabilization time is 16 oscillation clocks. All system and peripheral control registers are then reset to their default hardware values (see Tables 5-1).

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

— All interrupts are disabled.

— The watchdog function (basic timer) is enabled.

— Ports 0, 1 and 2 are set to input mode and all pull-up resistors are disabled for the I/O port pin circuits.

— Peripheral control and data register settings are disabled and reset to their default hardware values (see Table 5-1).

— The program counter (PC) is loaded with the program reset address in the ROM, 0100H.

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

HARDWARE RESET VALUES

Tables 5-1 list the reset values for CPU and system registers, peripheral control registers, and peripheral data registers following a reset operation. The following notation is used to represent reset values:

— A "1" or a "0" shows the reset bit value as logic one or logic zero, respectively.

— An 'x' means that the bit value is undefined after a reset.

— A dash ('-') means that the bit is either not used or not mapped (but a 0 is read from the bit position)

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 (read-only)	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	×	×	×	×	×	×	0	0
Register pointer 0	RP0	214	D6H	1	1	0	0	0	–	–	–
Register pointer 1	RP1	215	D7H	1	1	0	0	1	–	–	–
Location D8H (SPH) is not mapped.											
Stack pointer (low byte)	SPL	217	D9H	×	×	×	×	×	×	×	×
Instruction pointer (high byte)	IPH	218	DAH	×	×	×	×	×	×	×	×
Instruction pointer (low byte)	IPL	219	DBH	×	×	×	×	×	×	×	×
Interrupt request register (read-only)	IRQ	220	DCH	0	0	0	0	0	0	0	0
Interrupt mask register	IMR	221	DDH	×	×	×	×	×	×	×	×
System mode register	SYM	222	DEH	0	–	–	×	×	×	0	0
Register page pointer	PP	223	DFH	0	0	0	0	0	0	0	0
Port 0 data register	P0	224	E0H	0	0	0	0	0	0	0	0
Port 1 data register	P1	225	E1H	0	0	0	0	0	0	0	0
Port 2 data register	P2	226	E2H	0	0	0	0	0	0	0	0
Location E3H–E6H is not mapped.											
Port 0 pull-up enable register	P0PUR	231	E7H	0	0	0	0	0	0	0	0
Port 0 control register (high byte)	P0CONH	232	E8H	0	0	0	0	0	0	0	0
Port 0 control register (low byte)	P0CONL	233	E9H	0	0	0	0	0	0	0	0

Table 8-1. Set 1 Register Values After Reset (Continued)

Register Name	Mnemonic	Address		Bit Values After Reset							
		Dec	Hex	7	6	5	4	3	2	1	0
Port 1 control register (high byte)	P1CONH	234	EAH	0	0	0	0	0	0	0	0
Port 1 control register (low byte)	P1CONL	235	EBH	0	0	0	0	0	0	0	0
Port 1 pull-up enable register	P1PUR	236	ECH	0	0	0	0	0	0	0	0
Location EDH–EFH is not mapped.											
Port 2 control register	P2CON	240	F0H	–	–	0	0	0	0	0	0
Port 0 interrupt enable register	P0INT	241	F1H	0	0	0	0	0	0	0	0
Port 0 interrupt pending register	P0PND	242	F2H	0	0	0	0	0	0	0	0
Counter A control register	CACON	243	F3H	0	0	0	0	0	0	0	0
Counter A data register (high byte)	CADATAH	244	F4H	1	1	1	1	1	1	1	1
Counter A data register (low byte)	CADATAL	245	F5H	1	1	1	1	1	1	1	1
Timer 1 counter register (high byte)	T1CNTH	246	F6H	0	0	0	0	0	0	0	0
Timer 1 counter register (low byte)	T1CNTL	247	F7H	0	0	0	0	0	0	0	0
Timer 1 data register (high byte)	T1DATAH	248	F8H	1	1	1	1	1	1	1	1
Timer 1 data register (low byte)	T1DATAL	249	F9H	1	1	1	1	1	1	1	1
Timer 1 control register	T1CON	250	FAH	0	0	0	0	0	0	0	0
Stop control register	STOPCON	251	FBH	0	0	0	0	0	0	0	0
Locations FCH is not mapped.											
Basic timer counter	BTCNT	253	FDH	×	×	×	×	×	×	×	×
External memory timing register	EMT	254	FEH	0	1	1	1	1	1	0	–
Interrupt priority register	IPR	255	FFH	×	×	×	×	×	×	×	×

NOTES:

1. Although the SYM register is not used for the KS88C01016/C01008/C01004/C01116/C01108/C01104 , SYM.5 should always be "0". If you accidentally write a 1 to this bit during normal operation, a system malfunction may occur.
2. Except for T0CNT, IRQ, T1CNTH, T1CNTL, and BTCNT, which are read-only, all registers in set 1 are read/write addressable.
3. You cannot use a read-only register as a destination field for the instructions OR, AND, LD, and LDB.
4. Interrupt pending flags are noted by shaded table cells.

POWER-DOWN MODES

STOP MODE

Stop mode is invoked by stop control register (STOPCON) setting and the instruction STOP. 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 3uA at 5.5V. All system functions stop when the clock "freezes,"

Stop mode can be released of two ways : by an INTR (Interrupt with Reset) or by a POR (Power On Reset).

USING POR TO RELEASE STOP MODE

Stop mode is released when the reset signal goes active by power on reset (POR): all system and peripheral control registers are reset to their default hardware values and the contents of all data registers are unknown states. When the oscillation stabilization interval has elapsed, the CPU starts the system initialization routine by fetching the program instruction stored in ROM location 0100H.

USING AN INTR TO RELEASE STOP MODE

Stop mode is released when INTR (Interrupt with Reset) occurs. INTR occurs when falling/rising edge is detected at P0 during stop mode and it make system reset.

NOTE

1. Do not use stop mode if you are using an external clock source because X_{IN} input must be cleared internally to V_{SS} to reduce current leakage.
2. STOP mode always be released by the system reset (INTR or POR) so the system register value and control register value are initialized as reset value. And when the reset occurs from INTR, the prime register value will be retained but it will be unknown states if it occurs from POR. So an application which is using stop mode should be added specific S/W which divide the system reset into STOP mode releasing or power on reset. Following Programming Tip can be useful for more understanding.

☞ PROGRAMMING TIP — To Divide STOP Mode Releasing and POR.

This example shows how to enter the stop mode and how to know it is stop mode releasing or power on RESET.

```
                ORG     0100H           ; Reset address
START           DI                      ;
                LD      BTCON,#03h      ; enable basic timer counter.
                LD      SPL,#0FFH       ; Initialize the system register
                CLR     SYM
                CLR     PP
                CLR     EMT
                CLR     IPR
                •
                •
                LD      P0CONH,#00H     ; Initialize the control register
                LD      P0CONL,#00H
                LD      P0PUR,#0FFH
                •
                •
                •

CHECK_RAM:                              ; Check the RAM data whether it is stop mode releasing
                                        ; or Power On RESET
                LD      R0,#0BFH        ; If Power On Reset , go to POR_RESET

CHK_R           CP      R0,@R0          ;
                JR      NE,POR_RESET
                DEC     R0
                CP      R0,#0B0H
                JR      UGE, CHK_R

STOP_RESET:                             ; STOP mode releasing.
                JR      MAIN            ;

POR_RESET LD            R0,#0FFH        ;Power On Reset
                                        ;CHECK RAM data are failed so clear all RAM data.

RAM_CLR         CLR     @R0
                DJNJ    R0,RAMCLR
                LD      R0,#0BFH        ;Initialize the CHECK RAM data as default value .
```

☞ **PROGRAMMING TIP — To Divide STOP Mode Releasing and POR. (Continued)**

```
CHK_W     LD      @R0,R0
          DEC     R0
          CP      R0,#0B0H
          JR      UGE,CHK_W

MAIN:     CP      P0,#0FFH          ;
          JR      EQ,ENT_STOP
          •
          •
          •
          JP      T,MAIN

ENT_STOP  LD      STOPCON,#0A5H     ;Enter the STOP mode.
          STOP
          NOP
          NOP
          JP      RESET
          •
          •
          •
```

IDLE MODE

Idle mode is invoked by the instruction IDLE (OPCODE 6FH). In Idle mode, CPU operations are halted while some peripherals remain active. During idle mode, the internal clock signal is gated away from the CPU and from all but the following peripherals, which remain active:

— Interrupt logic
— Timer 0
— Timer 1
— Counter A

I/O port pins retain the mode (input or output) they had at the time Idle mode was entered.

Idle Mode Release

You can release Idle mode in one of two ways:

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 the *slowest clock* because of the hardware reset value for the CLKCON register. If all external interrupts are masked in the IMR register, a reset is the only way you can release Idle mode.
2. Activate any enabled interrupt ; internal or external. When you use an interrupt to release Idle mode, the 2-bit CLKCON.4/CLKCON.3 value remains unchanged, and the *currently selected clock* value is used. The interrupt is then serviced. When the return-from-interrupt condition (IRET) occurs, the instruction immediately following the one which initiated Idle mode is executed.

NOTE

Only external interrupts with an RC delay built in to the pin circuit can be used to release Stop mode without reset. To release Idle mode, you can use either an external interrupt or an internally-generated interrupt.

SUMMARY TABLE OF STOP MODE, AND IDLE MODE

Table 8-2. Summary of Each Mode

Item/Mode	IDLE	STOP
Approach Condition	V_{DD} is higher than V_{LVD} ($V_{LVD} < V_{DD}$). IDLE (instruction).	V_{DD} is higher than V_{LVD} ($V_{LVD} < V_{DD}$). STOPCON ≤ A5H STOP instruction
Release Source	Interrupt T0/T1 interrupt Counter A interrupt Ext. interrupt (Port0) RESET POR LVD WDT	RESET INTR POR

9 I/O PORTS

OVERVIEW

The KS88C01016/C01018/C01004/C01116/C01108/C01104 microcontroller has three bit-programmable I/O ports, P0-P2. Two ports, P0-P1, are 8-bit ports and P2 is a 3-bit port. This gives a total of 19 I/O pins in the KS88C01016/C01008/C01004/C01116/C01108/C01104 "s 24-pin package. Each port is bit-programmable and can be flexibly configured to meet application design requirements. The CPU accesses ports by directly writing or reading port registers. No special I/O instructions are required.

For IR universal remote controller applications, ports 0, and1 are usually configured to the keyboard matrix and port 2 is used to transmit the remote controller carrier signal or to indicate operating status by turning on a LED.

Table 9-1 gives you a general overview of KS88C01016/C01008/C01004/C01116/C01108/C01104 I/O port functions.

Table 9-1. KS88C01016/C01008/C01004/C01116/C01108/C01104 Port Configuration Overview

Port	Configuration Options
0	8-bit general-purpose I/O port; Input or push-pull output; external interrupt input on falling edges, rising edges, or both edges; all P0 pin circuits have noise filters and interrupt enable/disable (P0INT) and pending control (P0PND); Pull-up resistors can be assigned to individual P0 pins using P0PUR register settings. Specially Interrupt with Reset(INTR) is assigned to release stop mode with system reset.
1	8-bit general-purpose I/O port; Input, open-drain output, or push-pull output. Pull-up resistors can be assigned to individual P1 pins using P1PUR register settings.
2	3-bit I/O port; input mode with or without pull-up, push-pull or open-drain output mode. REM and T0PWM can be assigned. Port 2 pins have high current drive capability to support LED applications. The port 2 data register contains three status bits: three for P2.0, P2.1 and P2.2 and one for remote controller carrier signal on/off status.

PORT DATA REGISTERS

Table 9-2 gives you an overview of the register locations of all three KS88C01016/C01008/C01004/C01116 /C01108/C01104 I/O port data registers. Data registers for ports 0, and 1 have the general format.

NOTE

The data register for port 2, P2, contains three bits for P2.0, P2.1 and P2.2, and an additional status bit for carrier signal on/off.

Table 9-2. Port Data Register Summary

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

Because port2 is a 3-bit I/O port, the port2 data register only contains values for P2.0,P2.1 and P2.2. The P2 register also contains values for P2.0,P2.1 and P2.2. The P2 register also contains a special carrier on/off bit(P2.5). See the port 2 description for details. All other KS88C01016/C01008/C01004/C01116/C01108/C01104 I/O ports are 8-bit.

PULL-UP RESISTOR ENABLE REGISTERS

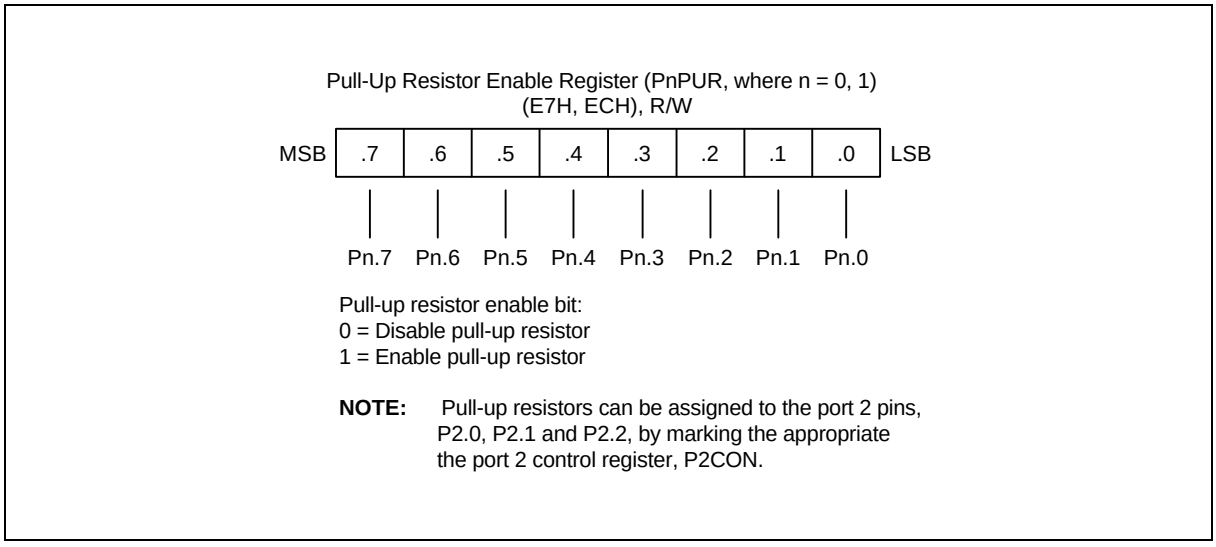


Figure 9-1. KS88C01016/C01008/C01004/C01116/C01108/C01104 I/O Port Data Register Format

PORT 0

Port 0 is a general-purpose, 8-bit I/O port. It is bit-programmable. Port 0 pins are accessed directly by read/write operations to the port 0 data register, P0 (set 1, E0H). The P0 pin circuits support pull-up resistor assignment using P0PUR register settings and all pins have noise filters for external interrupt inputs.

Two 8-bit control registers are used to configure port 0 pins: P0CONH (set 1, E8H) for the upper nibble pins, P0.7–P0.4, and P0CONL (set 1, E9H) for lower nibble pins, P0.3–P0.0. Each control register byte contains four bit-pairs and each bit-pair configures one pin (see Figures 9-2 and 9-3). A hardware reset clears all P0 control and data registers to '00H'.

A separate register, the port 0 interrupt control register, P0INT (set 1, F1H), is used to enable and disable external interrupt input. You can poll the port 0 interrupt pending register, P0PND to detect and clear pending conditions for these interrupts.

The lower-nibble pins, P0.3–P0.0, are used for INT3–INT0 input (IRQ6), respectively. The upper nibble pins, P0.7–P0.4, are all used for INT4 input (IRQ7). Interrupts that are detected at any of these four pins are processed using the same vector address (E8H).

Port 0 , P0.0 - P0.7, is assigned interrupt with reset(INTR) to release stop mode with system reset.

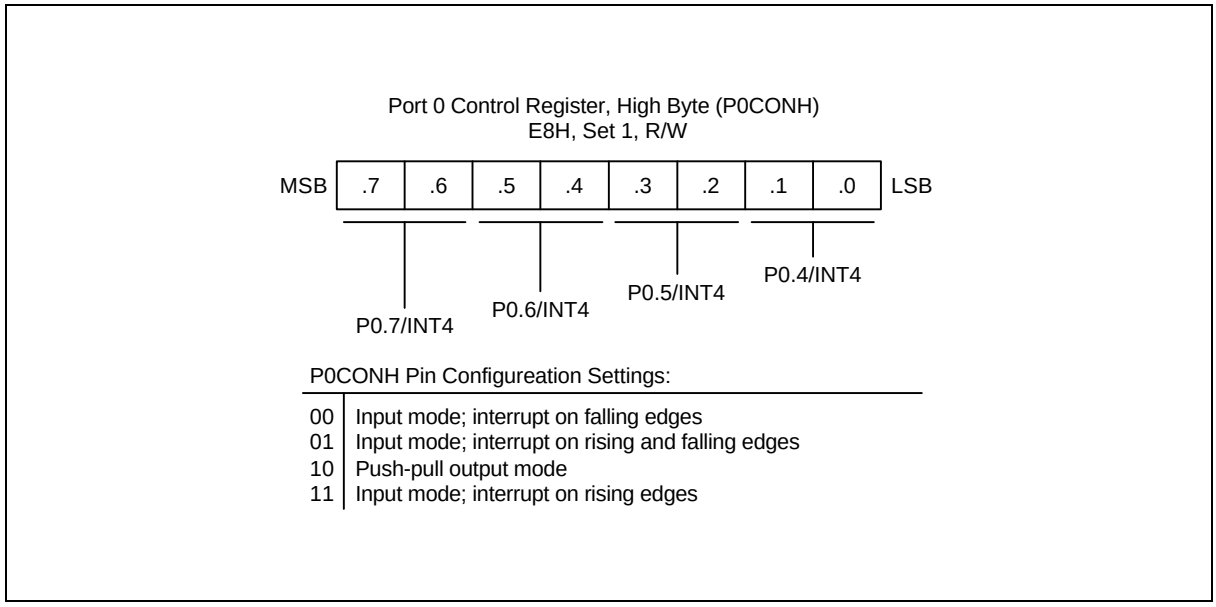


Figure 9-2. Port 0 High-Byte Control Register (P0CONH)

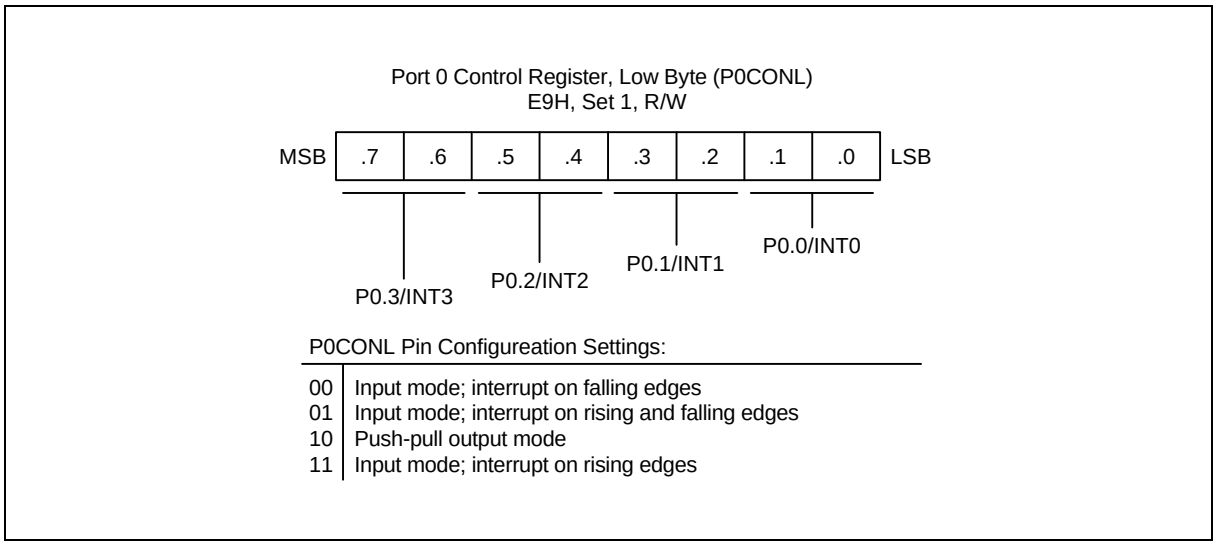


Figure 9-3. Port 0 Low-Byte Control Register (P0CONL)

PORT 0 INTERRUPT ENABLE REGISTER (P0INT)

The port 0 interrupt control register, P0INT, is used to enable and disable external interrupt input at individual P0 pins (see Figure 10-5). To enable a specific external interrupt, you set its P0INT.n bit to "1". You must also be sure to make the correct settings in the corresponding port 0 control register (P0CONH, P0CONL).

PORT 0 INTERRUPT PENDING REGISTER (P0PND)

The port 0 interrupt pending register, P0PND, contains pending bits (flags) for each port 0 interrupt (see Figure 10-6). When a P0 external interrupt is acknowledged by the CPU, the service routine must clear the pending condition by writing a "0" to the appropriate pending flag in the P0PND register (Writing a "1" to the pending bit has no effect).

NOTE

A hardware reset(INTR, POR) clears the P0INT and P0PND registers to '00H'. For this reason, the application program's initialization routine must enable the required external interrupts for port 0, and for the other I/O ports.

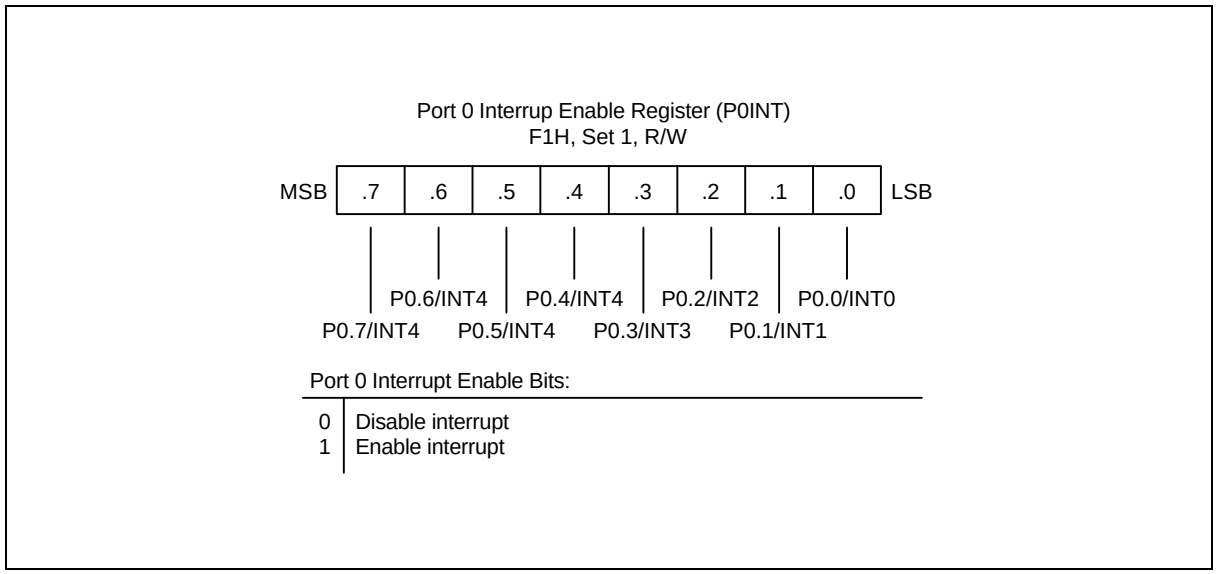


Figure 9-4. Port 0 External Interrupt Control Register (P0INT)

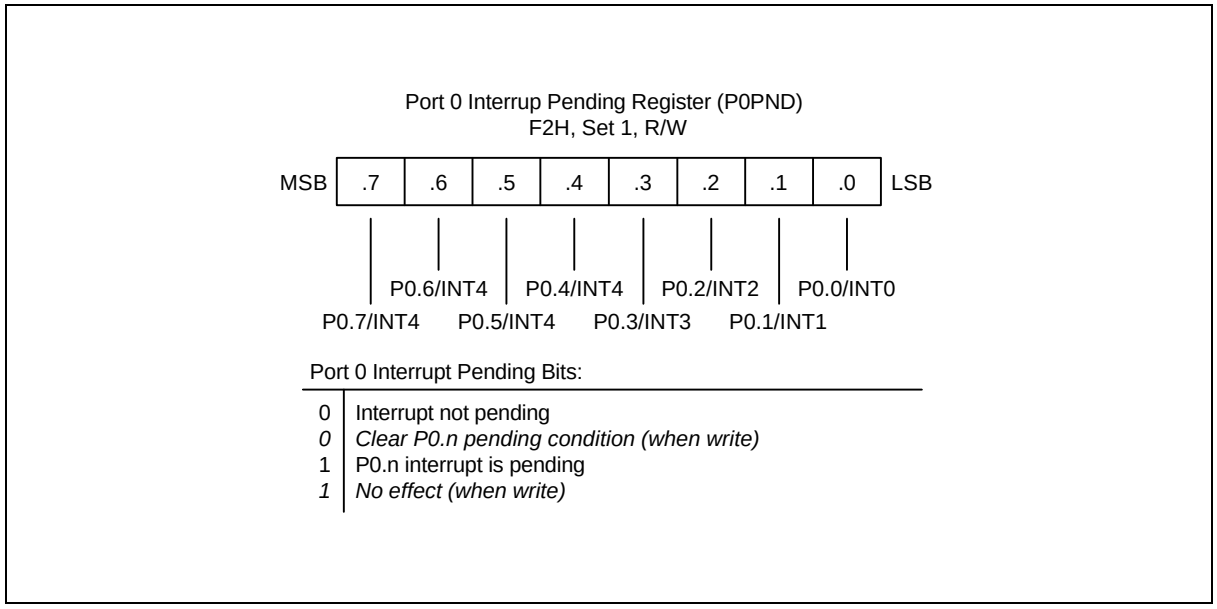


Figure 9-5. Port 0 External Interrupt Pending Register (P0PND)

PORT 1

Port 1 is a bit-programmable 8-bit I/O port. Port 1 pins are accessed directly by read/write operations to the port 1 data register, P1 (set 1, E1H).

To configure port 1, the initialization routine writes the appropriate values to the two port 1 control registers: P1CONH (set 1, EAH) for the upper nibble pins, P1.7–P1.4, and P1CONL (set 1, EBH) for the lower nibble pins, P1.3–P1.0. Each 8-bit control register contains four bit-pairs and each 2-bit value configures one port pin (see Figures 9-6 and 9-7).

Following a hardware reset, the port 1 control registers are cleared to '00H', configuring port 0 initially to Input mode.

To assign pull-up resistors to P1 pins, you make the appropriate settings to the port 1 pull-up resistor enable register, P1PUR.

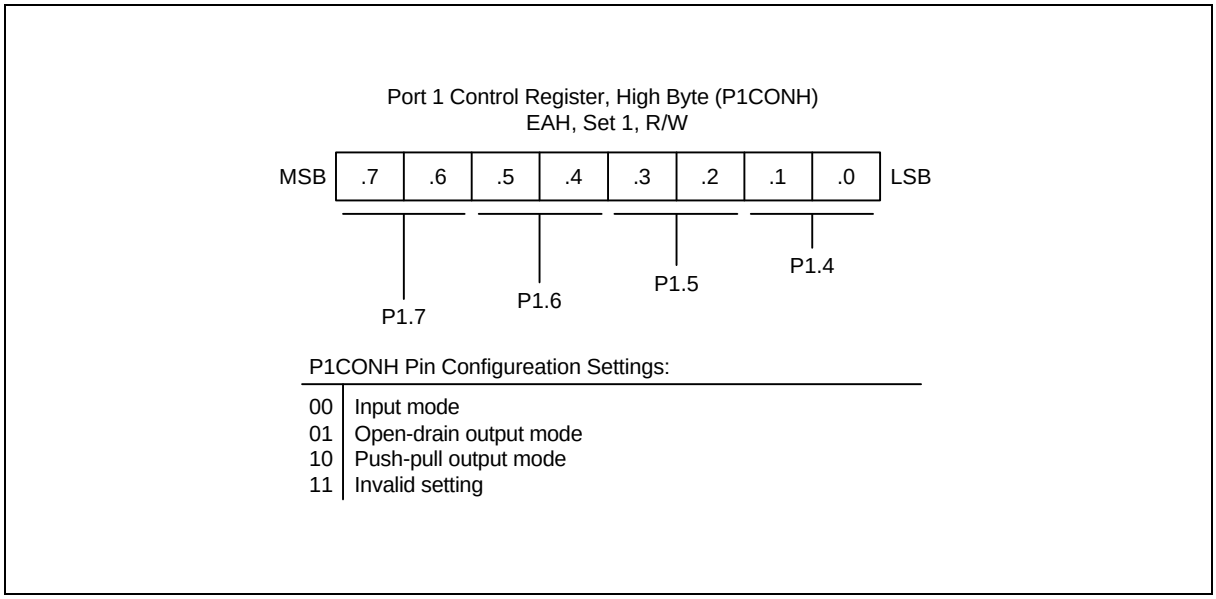


Figure 9-6. Port 1 High-Byte Control Register (P1CONH)

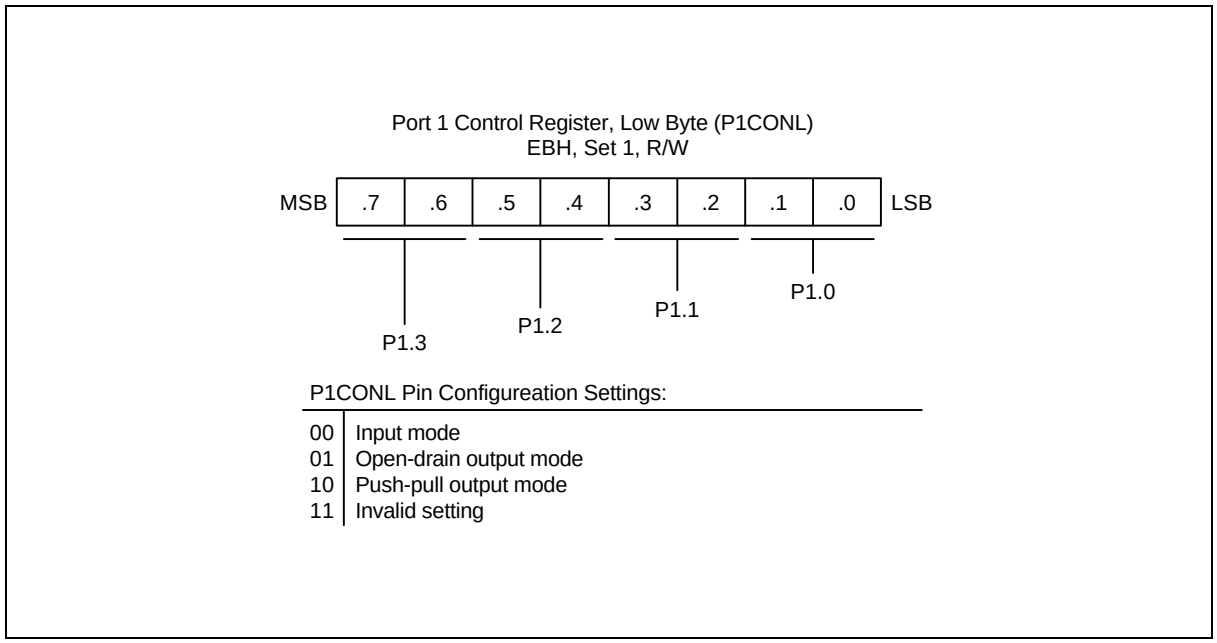


Figure 9-7. Port 1 Low-Byte Control Register (P1CONL)

PORT 2

Port 2 is a bit-programmable 3-bit I/O port. Port 2 pins are accessed directly by read/write operations to the port 2 data register, P2 (set 1, E2H). You can configure port 2 pins individually to Input mode, open-drain output mode, or push-pull output mode.

P2.0, P2.1 and P2.2 are configured by writing 6-bit data value to the port 2 control register, P2CON. You can configure these pins to support input functions (Input mode, with or without pull-up, for T0CK) or output functions (push-pull or open-drain output mode for REM and timer 0 PWM).

Port 2 pins have high current drive capability to support LED applications.

A reset operation clears P2CON to '00H', selecting Input mode as the initial port 2 function.

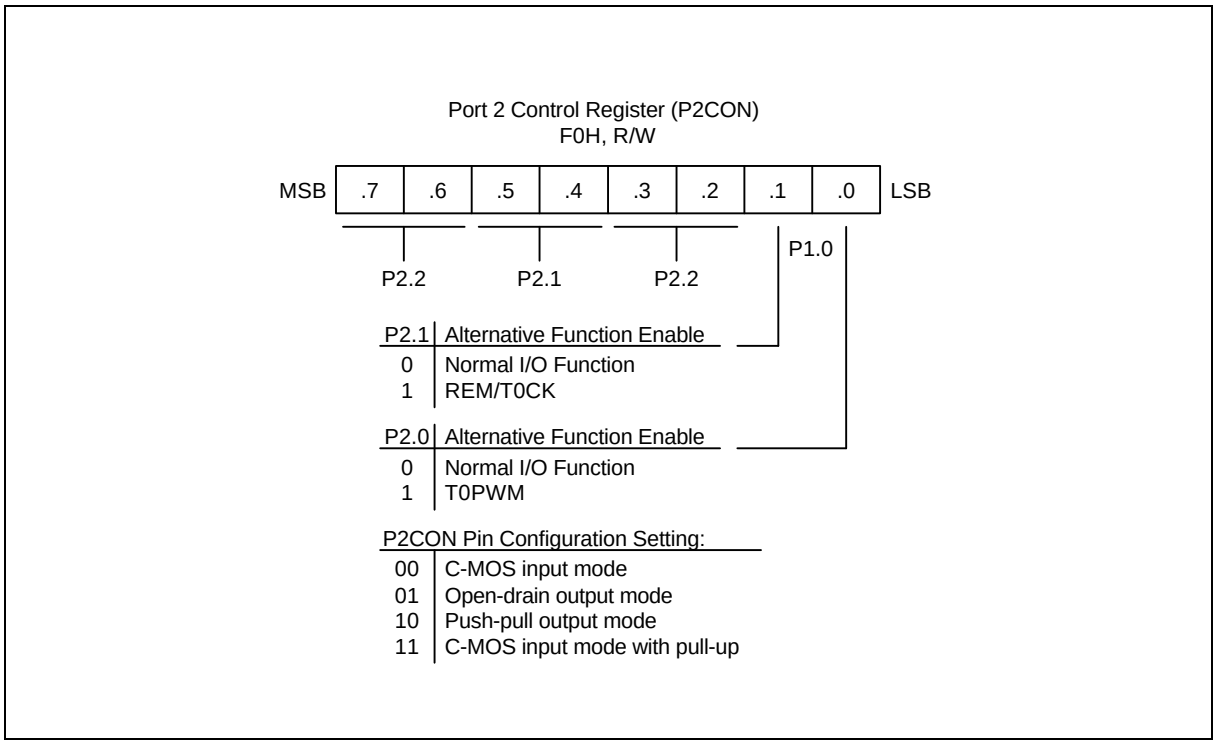


Figure 9-8. Port 2 Control Register (P2CON)

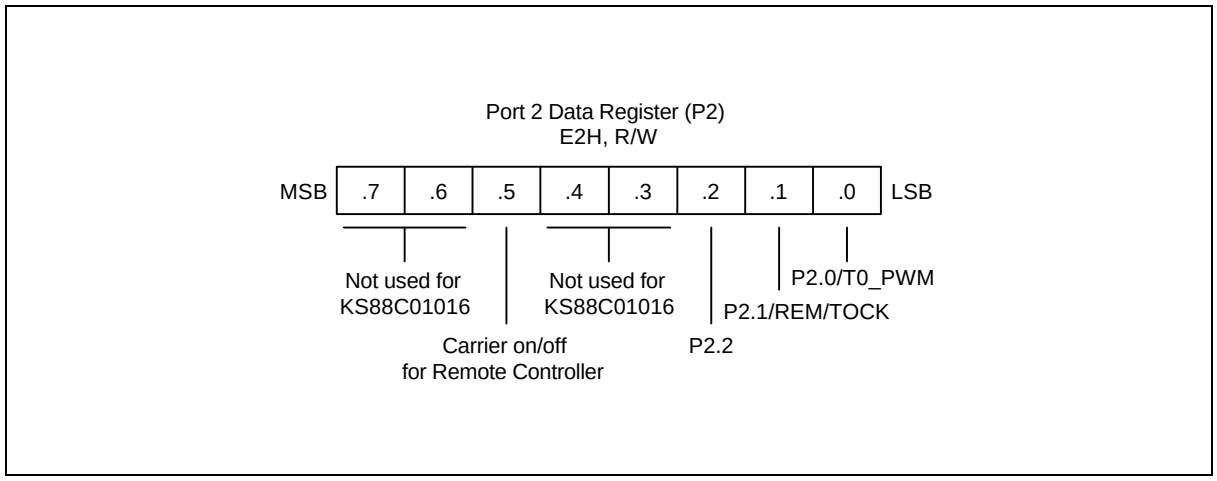


Figure 9-9. Port 2 Data Register (P2)

10 BASIC TIMER and TIMER 0

MODULE OVERVIEW

The KS88C01016/C01008/C01004/C01116/C01108/C01104 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, or

— To signal the end of the required oscillation stabilization interval after a reset or a Stop mode release.

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 system reset clears BTCON. 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. For more reliability, we recommend to use the Watch-dog timer function in remote controller and hand-held product application.

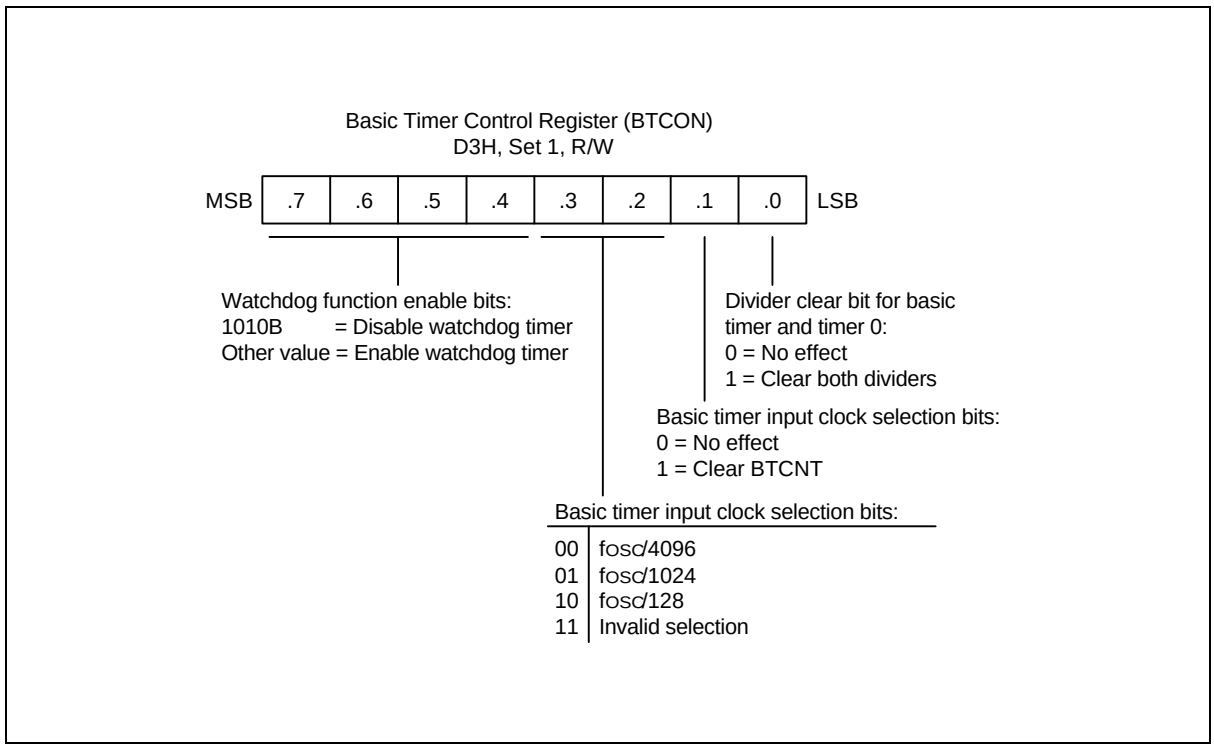


Figure 10-1. Basic Timer Control Register (BTCON)

BASIC TIMER FUNCTION DESCRIPTION

Watchdog Timer Function

You can program the basic timer overflow signal (BTOVF) to generate a reset by enabling the watchdog function. A reset clears BTCON to '00H', automatically enabling the watchdog timer function. A reset also selects the CPU clock (as determined by the current CLKCON register setting),divided by 4096, as the BT clock.

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

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

TIMER 0 CONTROL REGISTER (T0CON)

You use the timer 0 control register, T0CON, to

— Select the timer 0 operating mode (interval timer)

— Select the timer 0 input clock frequency

— Clear the timer 0 counter, T0CNT

— Enable the timer 0 overflow interrupt or timer 0 match interrupt

— Clear timer 0 match 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 all timer 0 interrupts. You can clear the timer 0 counter at any time during 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 interrupt (IRQ0, vector FCH), you must write T0CON.1 to "1". To detect a match interrupt pending condition, the application program polls T0CON.0. When a "1" is detected, a timer 0 match 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.

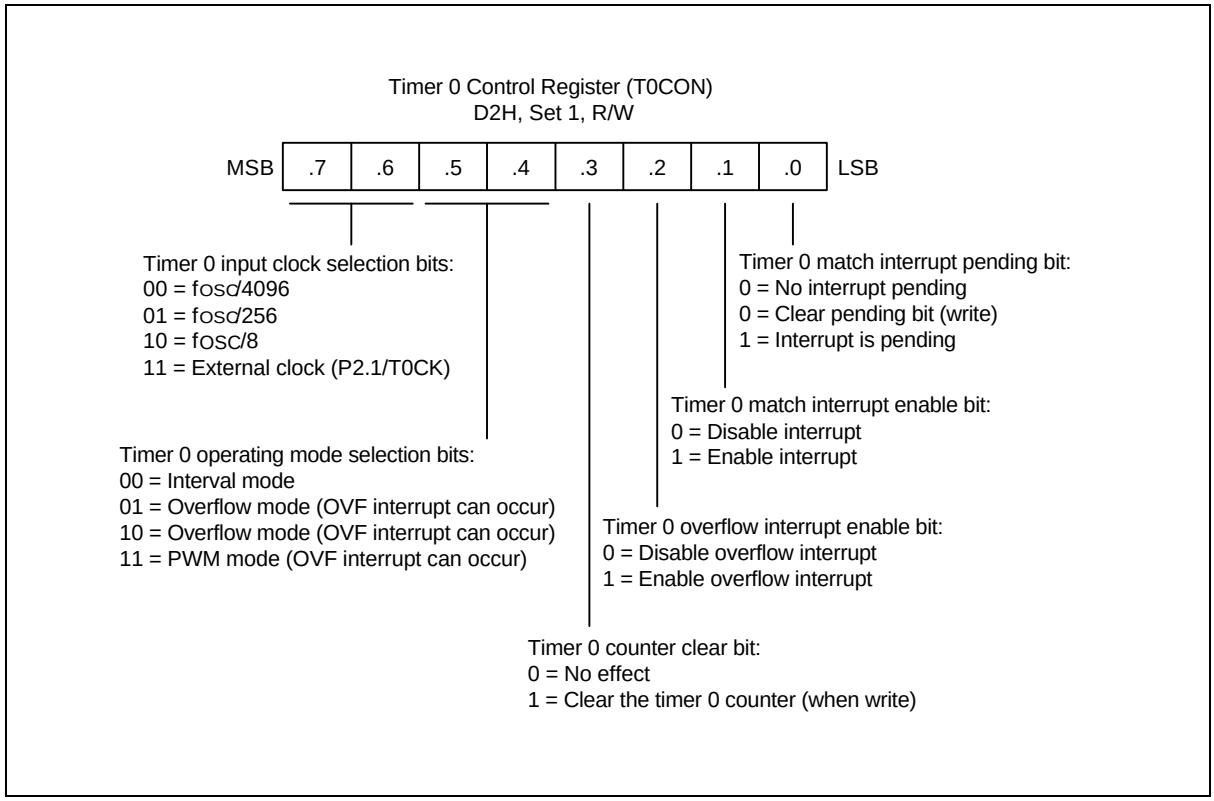


Figure 10-2. Timer 0 Control Register (T0CON)

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 interrupt (T0INT). T0OVF is interrupt level IRQ0, vector FAH. T0INT also belongs to interrupt level IRQ0, but is assigned the separate 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, however, be cleared by the application's interrupt service routine by writing a "0" to the T0CON.0 interrupt pending bit.

Interval Timer Mode

In interval timer mode, a match signal is generated when the counter value is identical to the value written to the T0 reference data register, T0DATA. The match signal generates a timer 0 match interrupt (T0INT, vector FCH) and clears the counter.

If, for example, you write the value '10H' to T0DATA and '0BH' to T0CON, the counter will increment until it reaches '10H'. At this point, the T0 interrupt request is generated, the counter value is reset, and counting resumes. With each match, the level of the signal at the timer 0 output pin is inverted (see Figure 10-3).

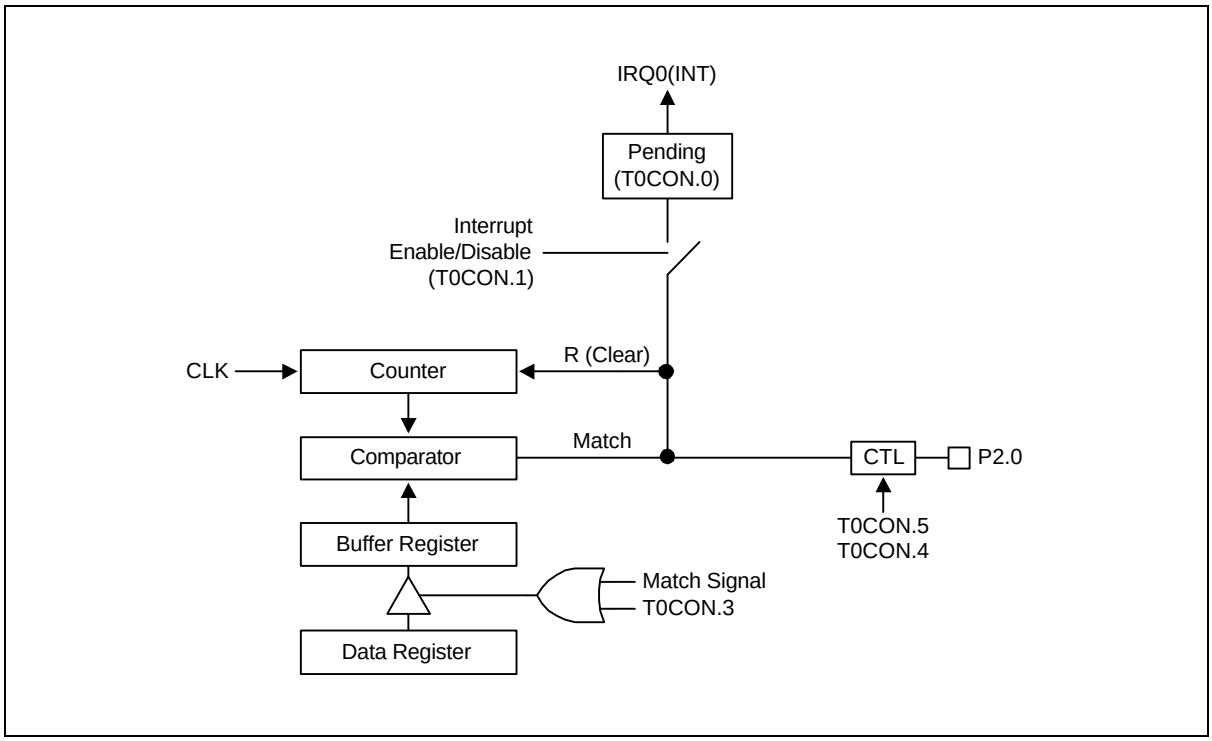


Figure 10-3. Simplified Timer 0 Function Diagram: Interval Timer Mode

Pulse Width Modulation Mode

Pulse width modulation (PWM) mode lets you program the width (duration) of the pulse that is output at the T0PWM pin. As in interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer 0 data register. In PWM mode, however, the match signal does not clear the counter. Instead, it runs continuously, overflowing at 'FFH', and then continues incrementing from '00H'.

Although you can use the match signal to generate a timer 0 overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the T0PWM 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 $t_{CLK} \times 256$ (see Figure 11-4).

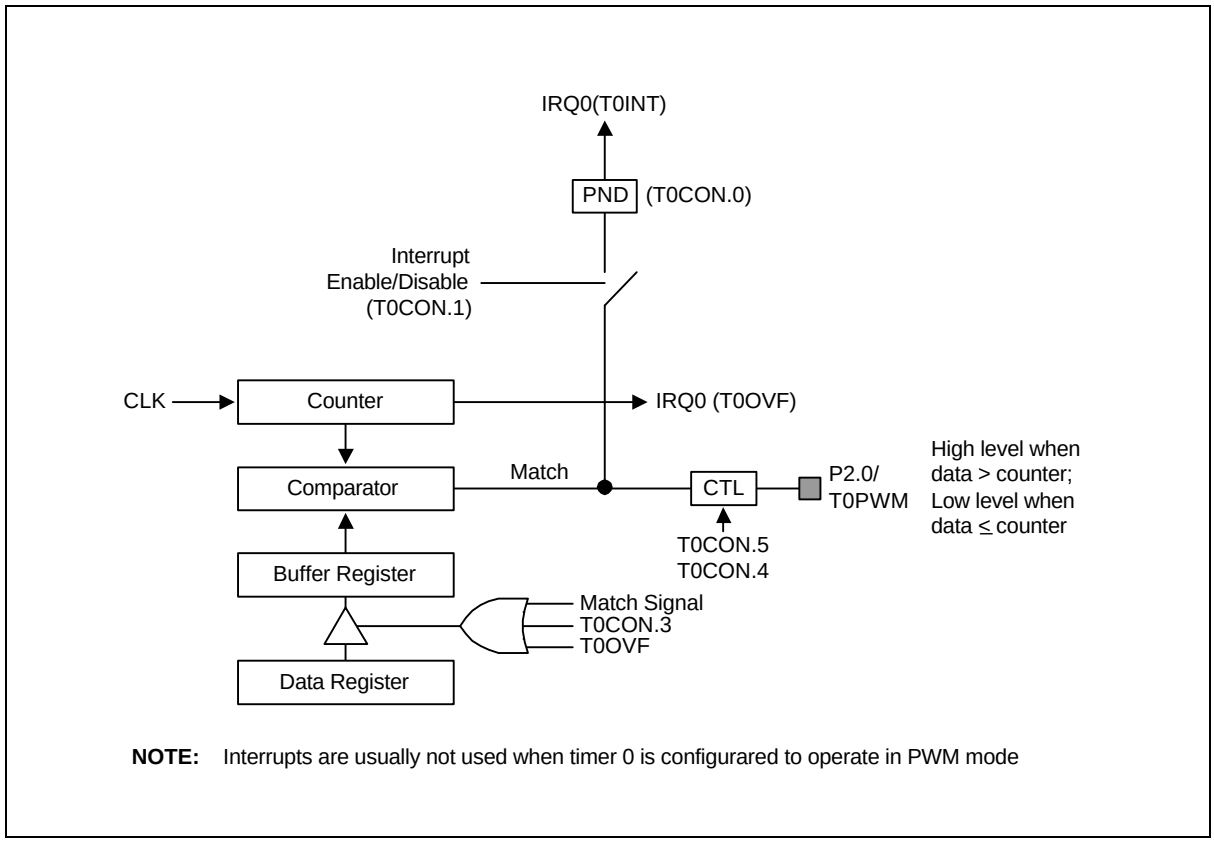
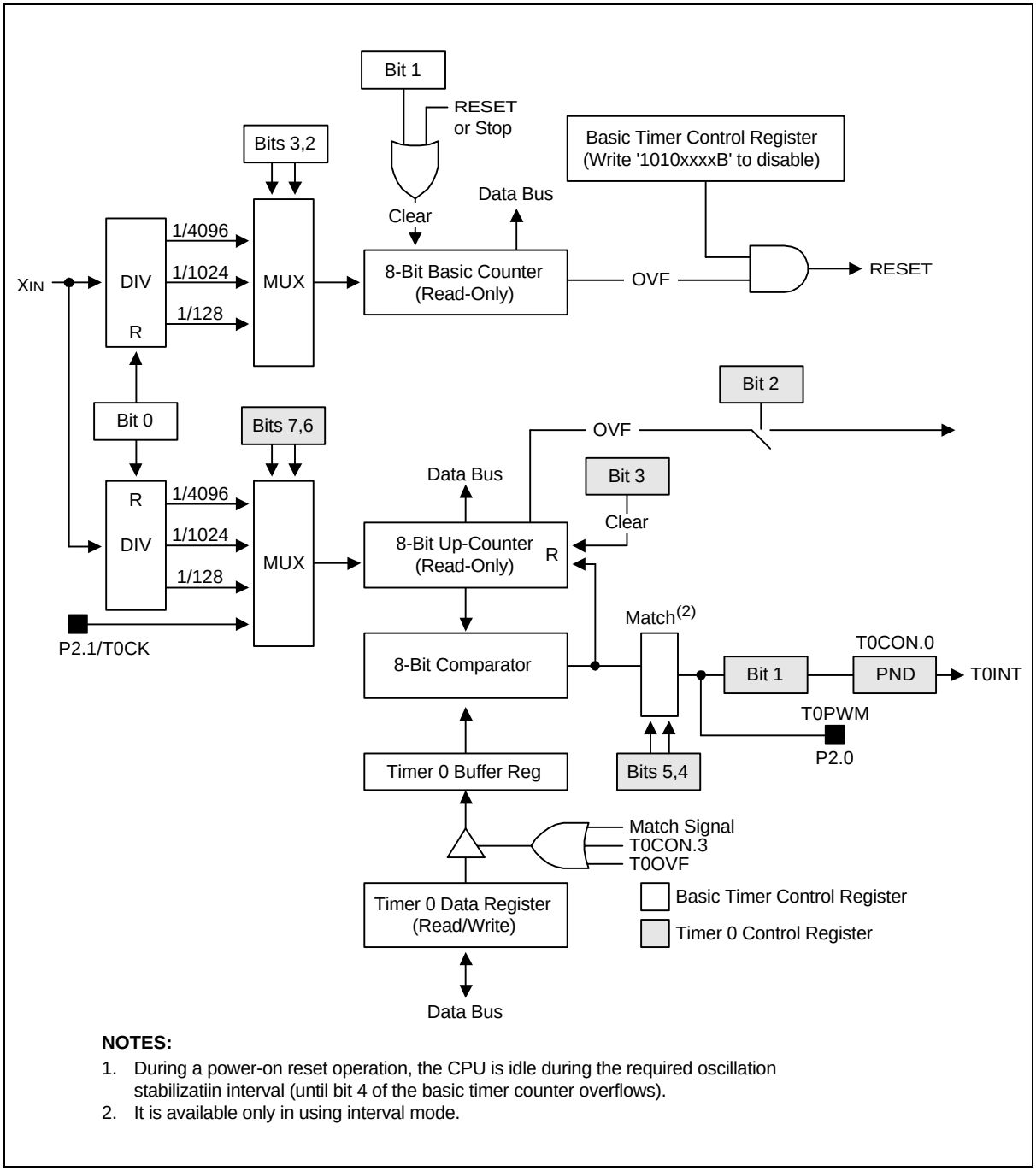


Figure 10-4. Simplified Timer 0 Function Diagram: PWM Mode



NOTES:

1. During a power-on reset operation, the CPU is idle during the required oscillation stabilizatiin interval (until bit 4 of the basic timer counter overflows).
2. It is available only in using interval mode.

Figure 10-5. Basic Timer and Timer 0 Block Diagram

☞ PROGRAMMING TIP — Configuring the Basic Timer

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

```
        ORG     0100H

RESET   DI                      ;  Disable all interrupts
        LD      BTCON,#03H      ;  Enable the watchdog timer
        LD      CLKCON,#18H     ;  Non-divided clock
        CLR     SYM             ;  Disable global and fast interrupts
        CLR     SPL             ;  Stack pointer low byte ← "0"
                                ;  Stack area starts at 0FFH
        •
        •
        •
        SRP     #0C0H           ;  Set register pointer ← 0C0H
        EI                      ;  Enable interrupts
        •
        •
        •
MAIN    LD      BTCON,#02H      ;  Enable the watchdog timer
                                ;  Basic timer clock: fOSC/4096
                                ;  Clear basic timer counter
        NOP
        NOP
        •
        •
        •
        JP      T,MAIN
        •
        •
        •
```

☞ Programming Tip — Programming Timer 0

This sample program sets timer 0 to interval timer mode, sets the frequency of the oscillator clock, and determines the execution sequence which follows a timer 0 interrupt. The program parameters are as follows:

— Timer 0 is used in interval mode; the timer interval is set to 4 milliseconds

— Oscillation frequency is 6 MHz

— General register 60H (page 0) ← 60H + 61H + 62H + 63H + 64H (page 0) is executed after a timer 0 interrupt

```
        ORG     0FAH            ;  Timer 0 overflow interrupt
        VECTOR  T0OVER
        ORG     0FCH            ;  Timer 0 match/capture interrupt
        VECTOR  T0INT
                ORG             0100H

RESET   DI                      ;  Disable all interrupts
    LD  BTCON,#0AAH             ;  Disable the watchdog timer
    LD  CLKCON,#18H             ;  Select non-divided clock
    CLR SYM         ;           Disable global and fast interrupts
    CLR SPL         ;           Stack pointer low byte ← "0"
                    ;           Stack area starts at 0FFH
    •
    •
    •
    LD  T0CON,#4BH              ;  Write '01001011B'
                    ;           Input clock is fOSC/256
                    ;           Interval timer mode
                    ;           Enable the timer 0 interrupt
                    ;           Disable the timer 0 overflow interrupt
    LD  T0DATA,#5DH             ;  Set timer interval to 4 milliseconds
                    ;           (6 MHz/256) ÷ (93 + 1)  =  0.25 KHz (4 ms)

    SRP #0C0H       ;           Set register pointer ← 0C0H
    EI              ;           Enable interrupts
    •
    •
    •
```

☞ PROGRAMMING TIP — Programming Timer 0 (Continued)

```
T0INT   PUSH    RP0                 ;  Save RP0 to stack
        SRP0    #60H                ;  RP0 ← 60H
        INC     R0                  ;  R0 ← R0 + 1
        ADD     R2,R0               ;  R2 ← R2 + R0
        ADC     R3,R2               ;  R3 ← R3 + R2 + Carry
        ADC     R4,R0               ;  R4 ← R4 + R0 + Carry

CP      R0,#32H                     ;  50 × 4 = 200 ms
        JR      ULT,NO_200MS_SET
        BITS    R1.2                ;  Bit setting (61.2H)


NO_200MS_SET:
        LD      T0CON,#42H          ;  Clear pending bit
        POP     RP0                 ;  Restore register pointer 0 value

T0OVER  IRET                        ;  Return from interrupt service routine
```

11 TIMER 1

OVERVIEW

The KS88C01016/C01008/C01004/C01116/C01108/C01104 microcontroller has a 16-bit timer/counter called timer 1 (T1). For universal remote controller applications, timer 1 can be used to generate the envelope pattern for the remote controller signal. Timer 1 has the following components:

— One control register, T1CON (set 1, FAH, R/W)

— Two 8-bit counter registers, T1CNTH and T1CNTL (set 1, F6H and F7H, read-only)

— Two 8-bit reference data registers, T1DATAH and T1DATAL (set 1, F8H and F9H, R/W)

— A 16-bit comparator

You can select one of the following clock sources as the timer 1 clock:

— Oscillator frequency (f_{OSC}) divided by 4, 8, or 16

— Internal clock input from the counter A module (counter A flip/flop output)

You can use Timer 1 in two ways:

— As a normal free run counter, generating a timer 1 overflow interrupt (IRQ1, vector F4H) at programmed time intervals.

— To generate a timer 1 match interrupt (IRQ1, vector F6H) when the 16-bit timer 1 count value matches the 16-bit value written to the reference data registers.

In the KS88C01016/C01008/C01004/C01116/C01108/C01104 interrupt structure, the timer 1 overflow interrupt has higher priority than the timer 1 match.

TIMER 1 OVERFLOW INTERRUPT

Timer 1 can be programmed to generate an overflow interrupt (IRQ1, F4H) whenever an overflow occurs in the 16-bit up counter. When you set the timer 1 overflow interrupt enable bit, T1CON.2, to “1”, the overflow interrupt is generated each time the 16-bit up counter reaches ‘FFFFH’. After the interrupt request is generated, the counter value is automatically cleared to ‘00H’ and up counting resumes. By writing a “1” to T1CON.3, you can clear/reset the 16-bit counter value at any time during program operation.

TIMER 1 MATCH INTERRUPT

Timer 1 can also be used to generate a match interrupt (IRQ1, vector F6H) whenever the 16-bit counter value matches the value that is written to the timer 1 reference data registers, T1DATAH and T1DATAL. When a match condition is detected by the 16-bit comparator, the match interrupt is generated, the counter value is cleared, and up counting resumes from ‘00H’.

In match mode, program software can poll the timer 1 match interrupt pending bit, T1CON.0, to detect when a timer 1 match interrupt pending condition exists (T1CON.0 = "1"). When the interrupt request is acknowledged by the CPU and the service routine starts, the interrupt service routine for vector F6H must clear the interrupt pending condition by writing a “0” to T1CON.0.

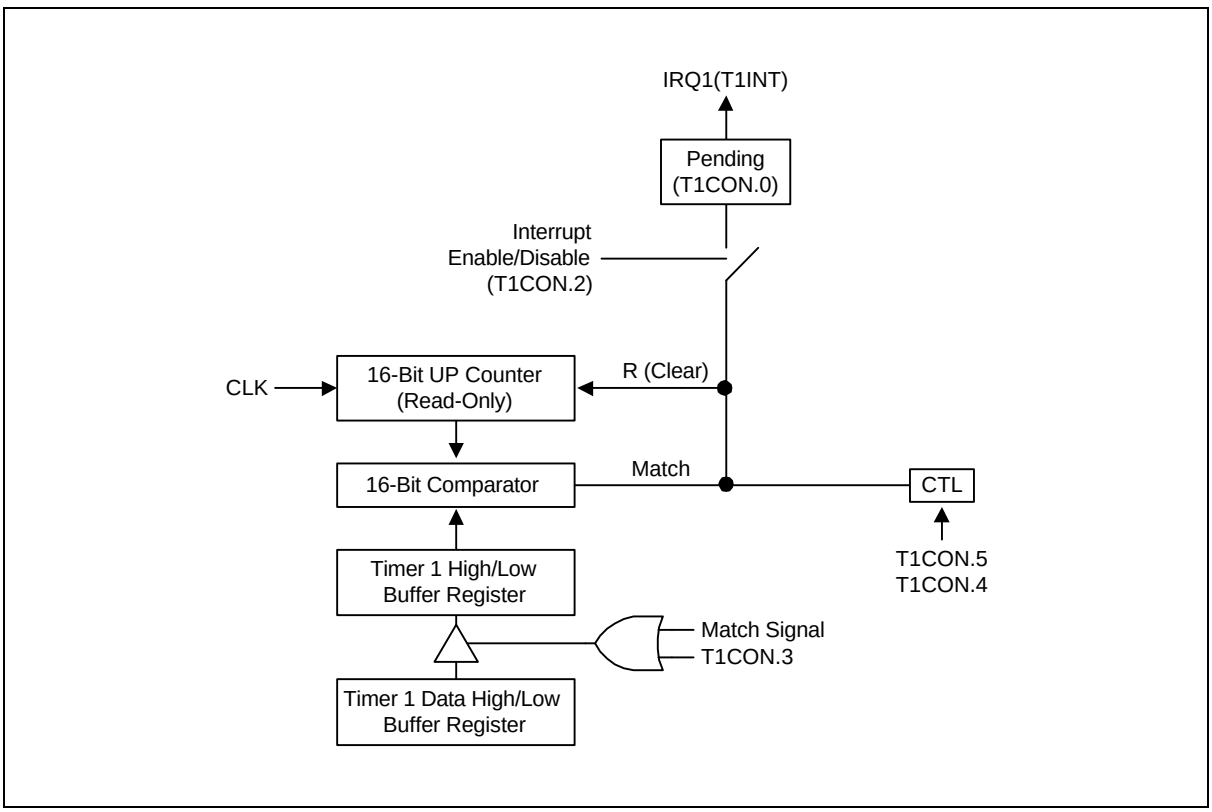
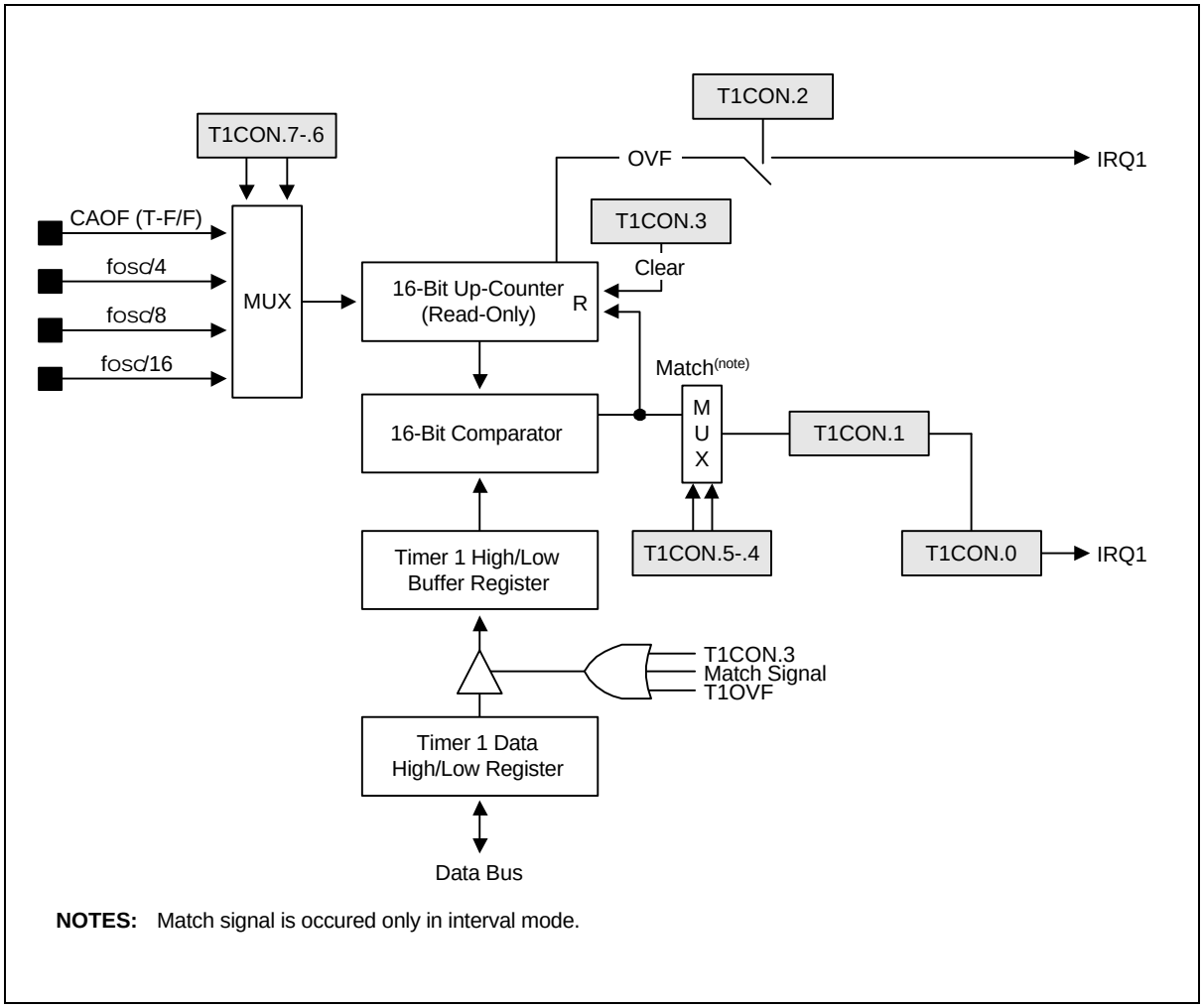


Figure 11-1. Simplified Timer 1 Function Diagram: Interval Timer Mode



NOTES: Match signal is occured only in interval mode.

Figure 11-2. Timer 1 Block Diagram

TIMER 1 CONTROL REGISTER (T1CON)

The timer 1 control register, T1CON, is located in set 1, FAH, and is read/write addressable. T1CON contains control settings for the following T1 functions:

— Timer 1 input clock selection
— Timer 1 operating mode selection
— Timer 1 16-bit down counter clear
— Timer 1 overflow interrupt enable/disable
— Timer 1 match interrupt enable/disable
— Timer 1 interrupt pending control (read for status, write to clear)

A reset operation clears T1CON to '00H', selecting f_{OSC} divided by 4 as the T1 clock, configuring timer 1 as a normal interval timer, and disabling the timer 1 interrupts.

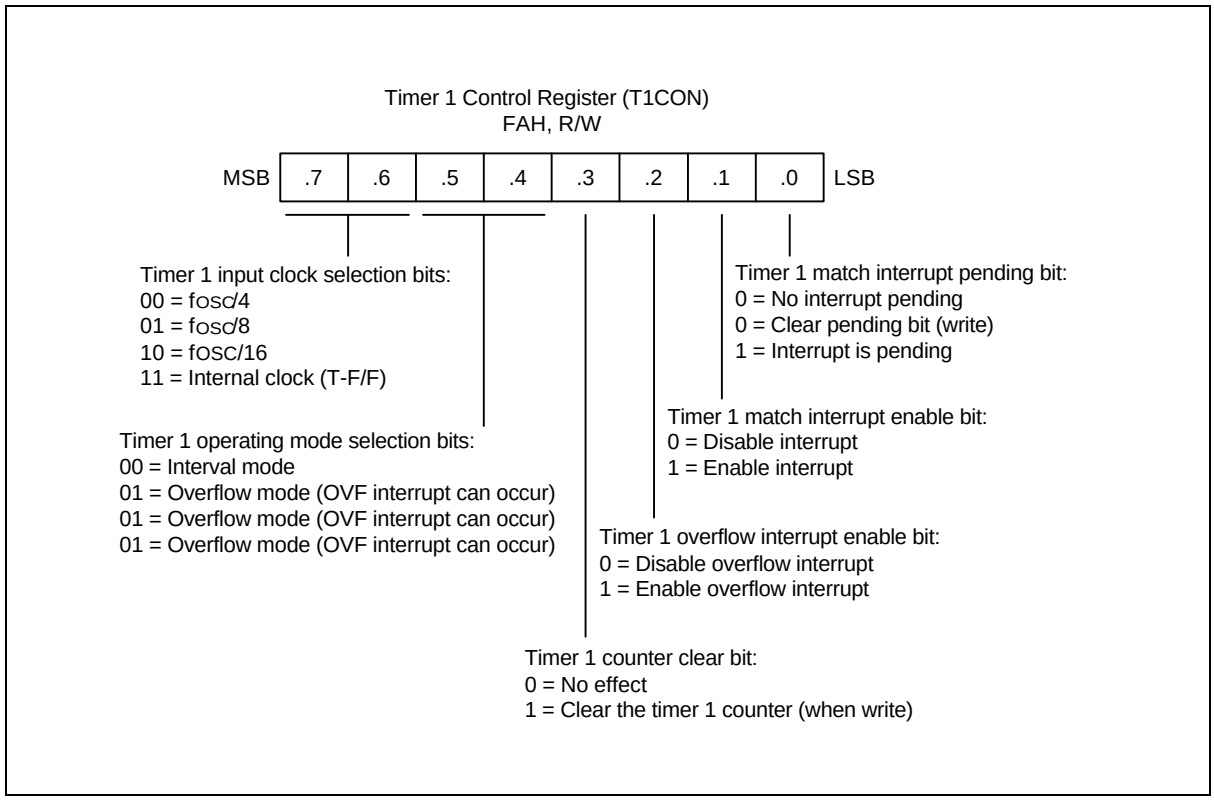


Figure 11-3. Timer 1 Control Register (T1CON)

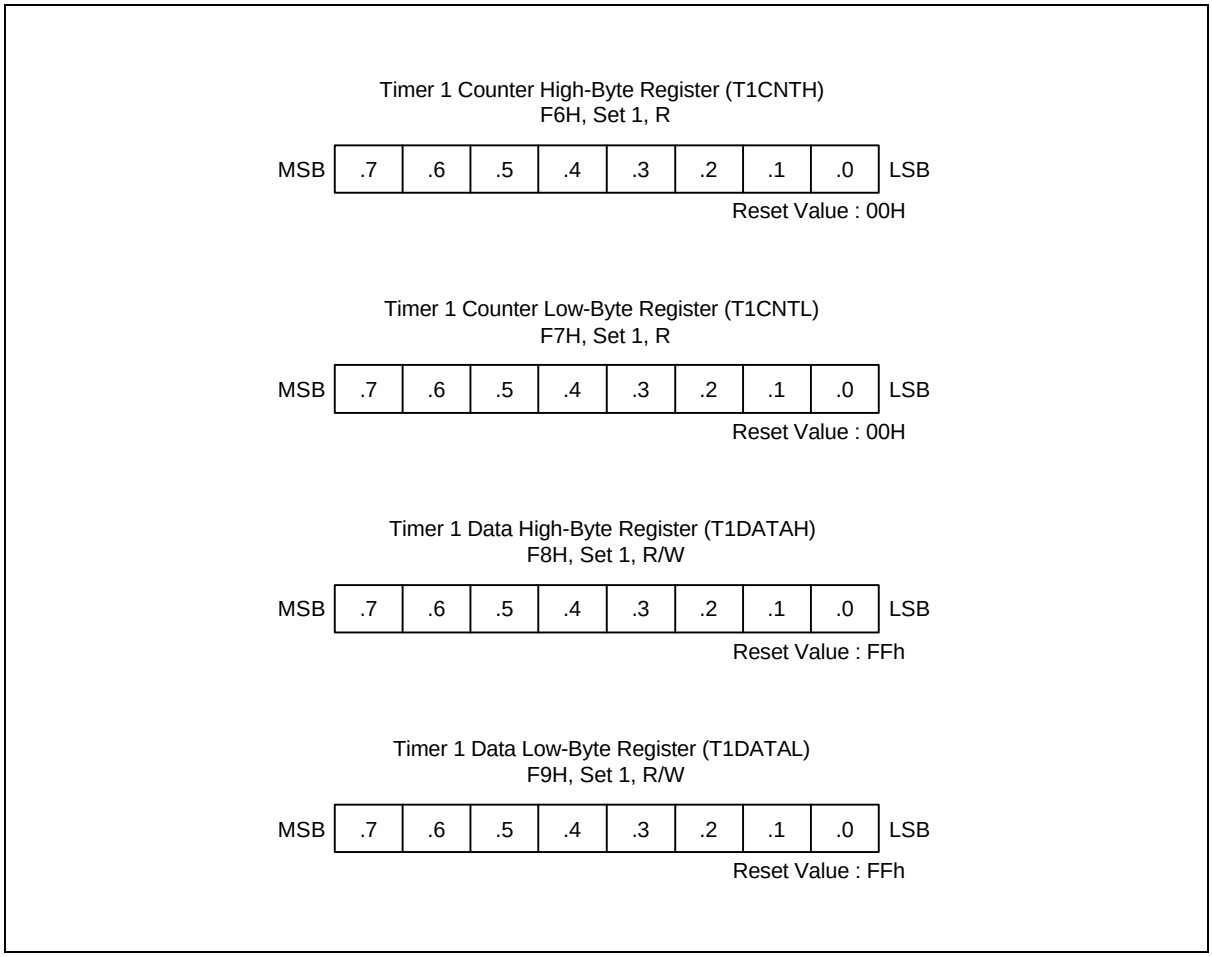


Figure 11-4. Timer 1 Registers

12 COUNTER A

OVERVIEW

The KS88C01016/C01008/C01004/C01116/C01108/C01104 microcontroller has an 8-bit counter called counter A. Counter A, which can be used to generate the carrier frequency, has the following components (see Figure 12-1):

— Counter A control register, CACON

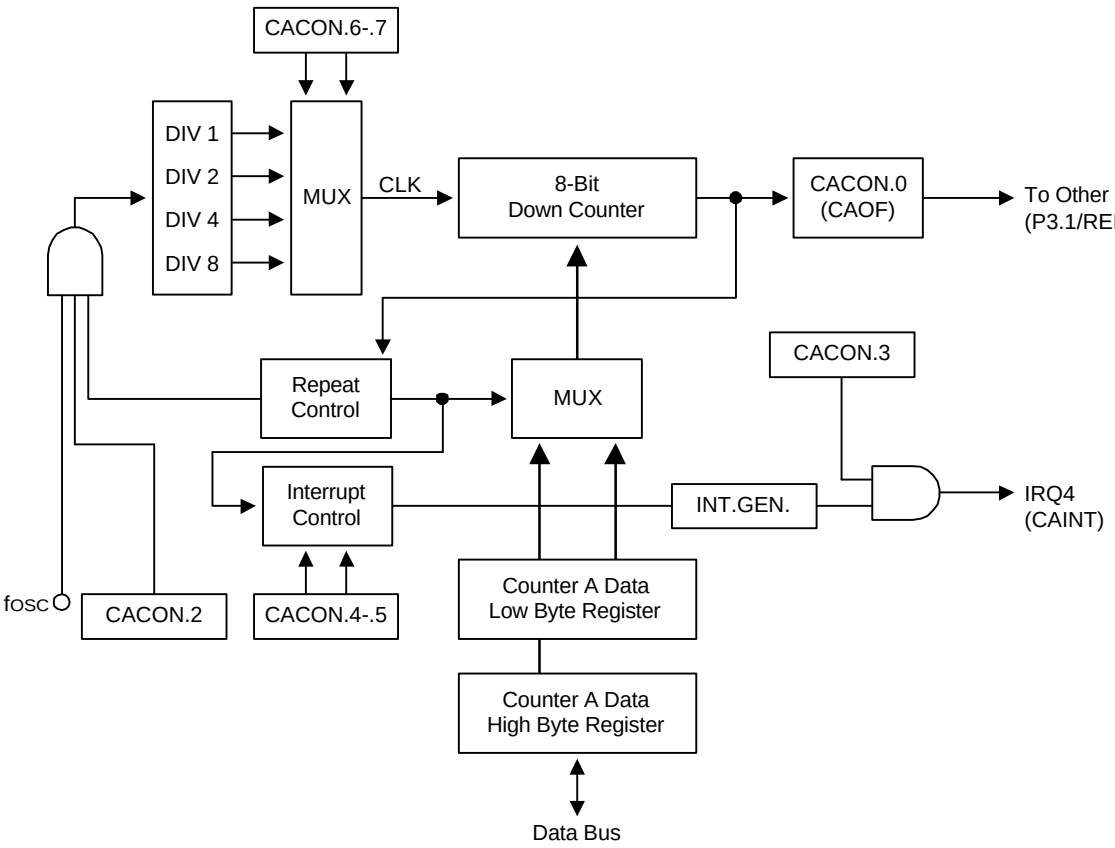
— 8-bit down counter with auto-reload function

— Two 8-bit reference data registers, CADATAH and CADATAL

Counter A has two functions:

— As a normal interval timer, generating a counter A interrupt (IRQ4, vector ECH) at programmed time intervals.

— To supply a clock source to the 16-bit timer/counter module, timer 1, for generating the timer 1 overflow interrupt.



NOTES: The value of the CADATAL register is loaded into the 8-bit counter when the operaion of the counter A Starts. If a borrow occurs in the counter, the value of the CADATAH register is loaded into the 8-bit counter. However, if the next borrow ovvurs, the value of the CADATAL register is loaded into the 8-bit counter.

Figure 12-1. Counter A Block Diagram

COUNTER A CONTROL REGISTER (CACON)

The counter A control register, CACON, is located in set 1, bank 0, F3H, and is read/write addressable. CACON contains control settings for the following functions (see Figure 12-2):

— Counter A clock source selection

— Counter A interrupt enable/disable

— Counter A interrupt pending control (read for status, write to clear)

— Counter A interrupt time selection

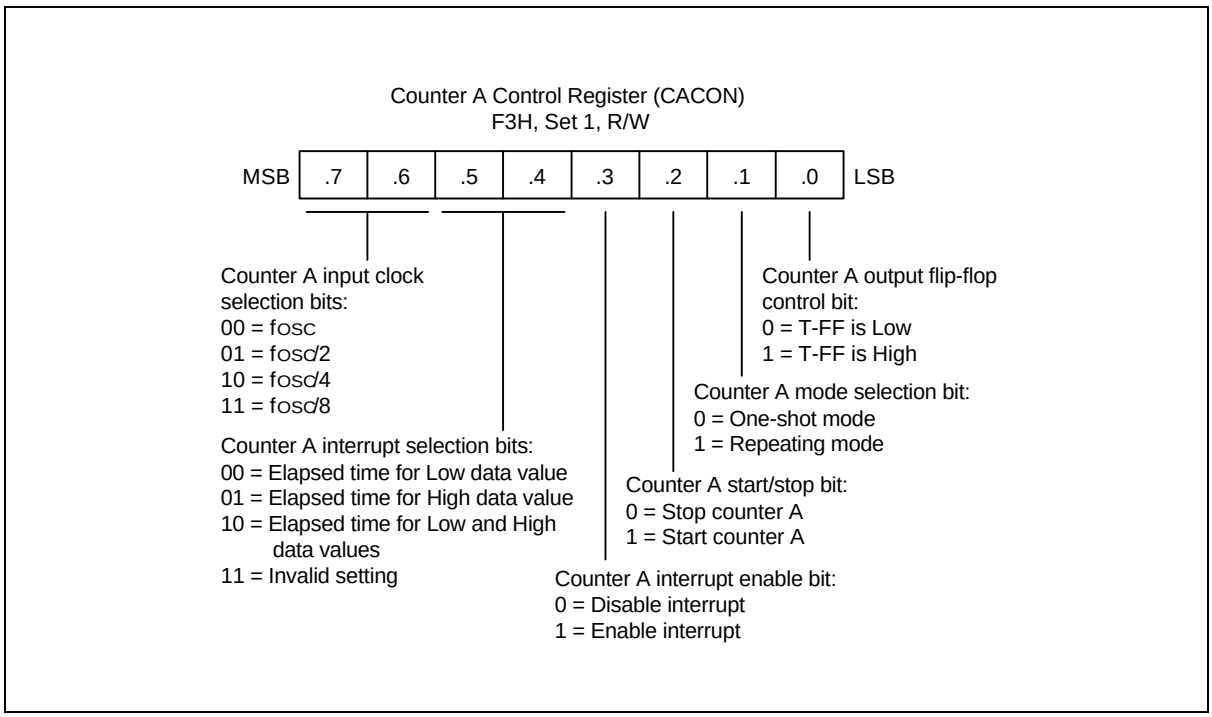


Figure 12-2. Counter A Control Register (CACON)

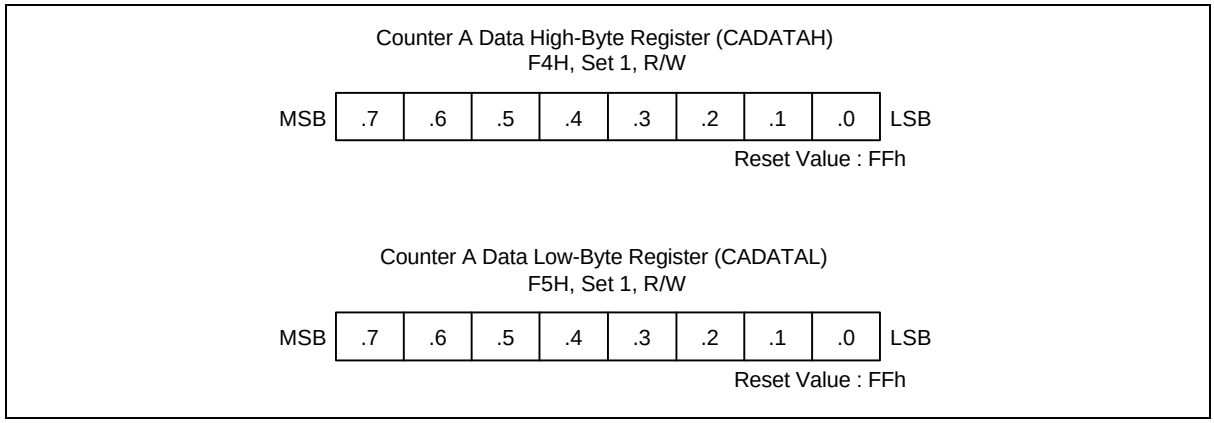
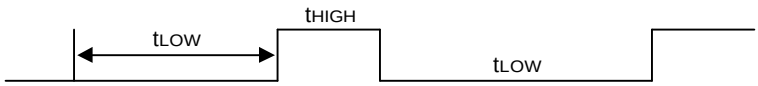


Figure 12-3. Counter A Registers

COUNTER A PULSE WIDTH CALCULATIONS



To generate the above repeated waveform consisted of low period time, t_{LOW}, and high period time, t_{HIGH}.

When CAOF = 0,
t_{LOW} = (CADATAL + 2) x 1/fxx, 0H < CADATAL < 100H, where fx = The selected clock.
t_{HIGH} = (CADATAH + 2) x 1/fxx, 0H < CADATAH < 100H, where fx = The selected clock.
When CAOF = 1,
t_{LOW} = (CADATAH + 2) x 1/fxx, 0H < CADATAH < 100H, where fx = The selected clock.
t_{HIGH} = (CADATAL + 2) x 1/fxx, 0H < CADATAL < 100H, where fx = The selected clock.

To make t_{LOW} = 24 us and t_{HIGH} = 15 us. f_{OSC} = 4 MHz, fx = 4 MHz/4 = 1 MHz

[Method 1] When CAOF = 0,

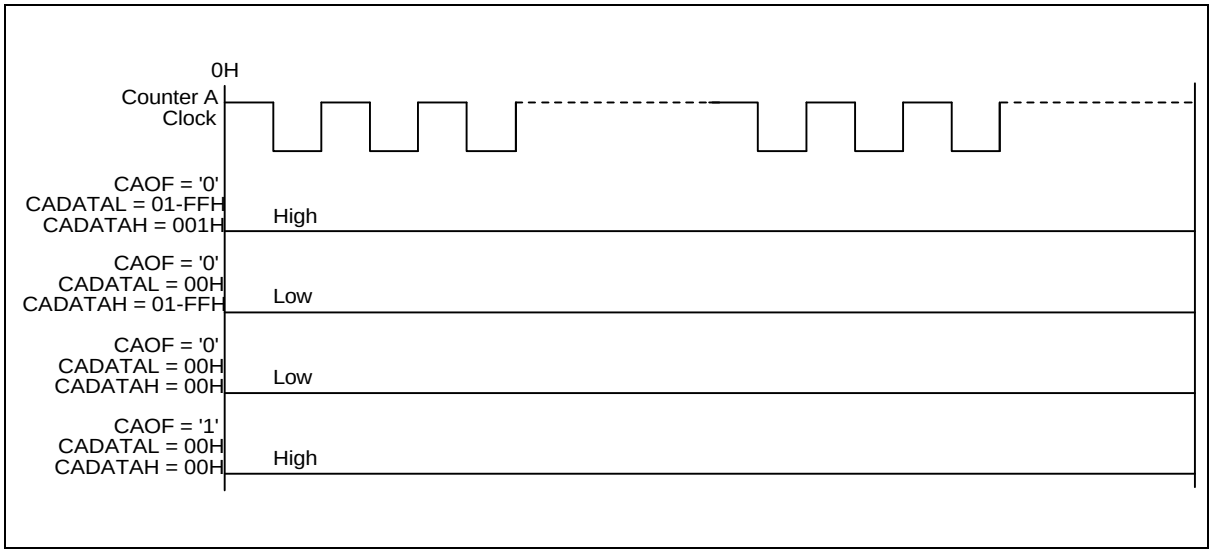
t_{LOW} = 24 us = (CADATAL + 2) /fx = (CADATAL + 2) × 1us, CADATAL = 22.

t_{HIGH} = 15 us = (CADATAH + 2) /fx = (CADATAH + 2) × 1us, CADATAH = 13.

[Method 2] When CAOF = 1,

t_{HIGH} = 15 us = (CADATAL + 2) /fx = (CADATAL + 2) × 1us, CADATAL = 13.

t_{LOW} = 24 us = (CADATAH + 2) /fx = (CADATAH + 2) × 1us, CADATAH = 22.



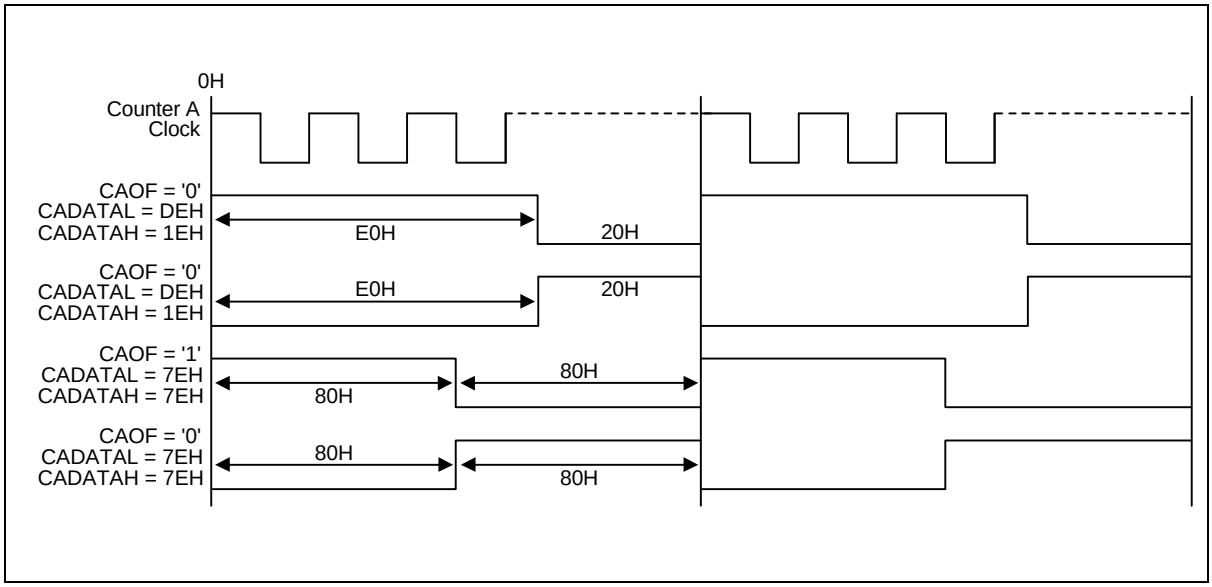
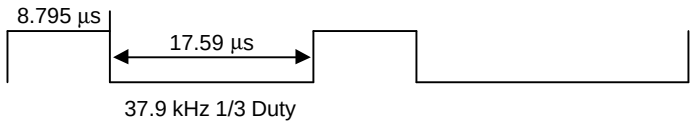


Figure 12-4. Counter A Output Flip-Flop Waveforms in Repeat Mode

☞ PROGRAMMING TIP — To Generate 38 kHz, 1/3duty Signal Through P2.1

This example sets Counter A to the repeat mode, sets the oscillation frequency as the Counter A clock source, and CADATAH and CADATAL to make a 38 kHz,1/3 Duty carrier frequency. The program parameters are:



— Counter A is used in repeat mode

— Oscillation frequency is 4 MHz (0.25 µs)

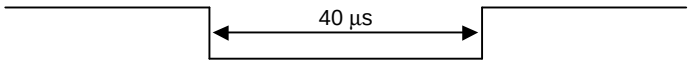
— CADATAH = 8.795 µs / 0.25 µs = 35.18, CADATAL = 17.59 µs / 0.25 µs = 70.36

— Set P2.1 C-MOS push-pull output and CAOF mode.

```
        ORG     0100H               ; Reset address
START   DI
        •
        •
        •
        LD      CADATAL,#(70-2)     ; Set 17.5 µs
        LD      CADATAH,#(35-2)     ; Set 8.75 µs
                                    ;
        LD      P2CON,#10101010B    ; Set P2 to C-MOS push-pull output.
                                    ; Set P2.1 to REM output
                                    ;
        LD      CACON,#00000110B    ; Clock Source ← fOSC
                                    ; Disable Counter A interrupt.
                                    ; Select repeat mode for Counter A.
                                    ; Start Counter A operation.
                                    ; Set Counter A Output Flip-flop(CAOF) high.
                                    ;
        LD      P2,#20H             ; Set P2.5(Carrier On/Off) to high.
                                    ; This command generates 38 kHz, 1/3duty pulse signal
                                    ;       through P2.1
                                    ;
        •
        •
        •
```

☞ PROGRAMMING TIP — To Generate a One Pulse Signal Through P2.1

This example sets Counter A to the one shot mode, sets the oscillation frequency as the Counter A clock source, and CADATAH and CADATAL to make a 40 µs width pulse. The program parameters are:



— Counter A is used in one shot mode
— Oscillation frequency is 4 MHz (1 clock = 0.25 µs)
— CADATAH = 40 µs / 0.25 µs = 160, CADATAL = 1
— Set P2.1 C-MOS push-pull output and CAOF mode.

```
        ORG     0100H               ;  Reset address
START   DI
        •
        •
        LD      CADATAH,# (160-2)   ;  Set 40 µs
        LD      CADATAL,# 1         ;  Set any value except 00H
                                    ;
        LD      P2CON,#10101010B    ;  Set P2 to C-MOS push-pull output.
                                    ;  Set P2.1 to REM output
                                    ;
        LD      CACON,#00000001B    ;  Clock Source ← fOSC
                                    ;  Disable Counter A interrupt.
                                    ;  Select one shot mode for Counter A.
                                    ;  Stop Counter A operation.
                                    ;  Set Counter A Output Flip-Flop (CAOF) high
        LD      P2,#20H             ;  Set P2.5(Carrier On/Off) to high.
        •
        •
        •

Pulse_out: LD   CACON,#00000101B    ;  Start Counter A operation
                                    ;  to make the pulse at this point.
        •                           ;  After the instruction is executed, 0.75 µs is required
        •                           ;  before the falling edge of the pulse starts.
        •
```

13 ELECTRICAL DATA

OVERVIEW

In this section, KS88C01016/C01008/C01004/C01116/C01108/C01104 electrical characteristics are presented in tables and graphs. The information is arranged in the following order:

- Absolute maximum ratings
- D.C. electrical characteristics
- Data retention supply voltage in Stop mode
- Stop mode release timing when initiated by a Reset
- I/O capacitance
- A.C. electrical characteristics
- Input timing for external interrupts (port 0)
- Oscillation characteristics
- Oscillation stabilization time

Table 13-1. 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_{IN}	–	– 0.3 to V_{DD} + 0.3	V
Output voltage	V_O	All output pins	– 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 0, 1, and 2	+ 100	
		Total pin current for port 3	+ 40	
Operating temperature	T_A	–	– 40 to + 85	°C
Storage temperature	T_{STG}	–	– 65 to + 150	°C

Table 13-2. D.C. Electrical Characteristics

(T_A = – 40°C to + 85°C, V_{DD} = 2.0 V to 3.6 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Operating Voltage	V_{DD}	f_{OSC} =8MHz (Instruction clock = 1.33 MHz)	2.0	–	3.6	V
		f_{OSC} = 4MHz (Instruction clock = 0.67 MHz)	1.7	–	3.6	
Input High voltage	V_{IH1}	All input pins except V_{IH2} and V_{IH3}	0.8 V_{DD}	–	V_{DD}	V
	V_{IH2}	X_{IN}	V_{DD} – 0.3		V_{DD}	
Input Low voltage	V_{IL1}	All input pins except V_{IL2} and V_{IL3}	0	–	0.2 V_{DD}	V
	V_{IL2}	X_{IN}			0.3	
Output High voltage	V_{OH1}	V_{DD}= 2.4 V, I_{OH} = – 6 mA Port 2.1 only, T_A = 25°C	V_{DD} – 0.7	–	–	V
	V_{OH2}	V_{DD} = 2.4 V, I_{OH} = – 2.2mA Port 2.0, 2.2, T_A = 25°C	V_{DD} - 0.7			
	V_{OH3}	V_{DD} = 2.4 V, I_{OH} = – 1 mA All output pins except Port2, T_A = 25°C	V_{DD} - 1.0	–	–	

Table 13-2. D.C. Electrical Characteristics (Continued)

(T_A = – 40°C to + 85°C, V_{DD} = 2.0 V to 3.6 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Output Low voltage	V_{OL1}	V_{DD} = 2.4 V, I_{OL} = 12 mA, port 2.1 only, T_A = 25°C	-	0.4	0.5	
	V_{OL2}	V_{DD} = 2.4 V, I_{OL} = 5 mA Port 2.0,2.2, T_A = 25°C		0.4	0.5	
	V_{OL3}	I_{OL} = 1 mA Ports 0 and 1, T_A = 25°C		0.4	1.0	
Input High leakage current	I_{LIH1}	V_{IN} = V_{DD} All input pins except X_{IN} and X_{OUT}	–	–	1	µA
	I_{LIH2}	V_{IN} = V_{DD}, X_{IN} and X_{OUT}			20	
Input Low leakage current	I_{LIL1}	V_{IN} = 0 V All input pins except X_{IN}, X_{OUT}	–	–	– 1	µA
	I_{LIL2}	V_{IN} = 0 V X_{IN} and X_{OUT}			– 20	
Output High leakage current	I_{LOH}	V_{OUT} = V_{DD} All output pins	–	–	1	µA
Output Low leakage current	I_{LOL}	V_{OUT} = 0 V All output pins	–	–	– 1	µA
Pull-up resistors	R_{L1}	V_{DD} = 2.4V, V_{IN} = 0 V; T_A = 25°C , Ports 0-2	44	55	95	KΩ
Supply current (note)	I_{DD1}	V_{DD} = 3.6 V ± 10% 8-MHz crystal	-	5	9	mA
		4-MHz crystal		2.6	5	
	I_{DD2}	Idle mode; V_{DD} = 3.6 V ± 10% 8-MHz crystal	–	1.0	2.5	
		4-MHz crystal		0.7	2.0	
	I_{DD3}	Stop mode; V_{DD} = 3.6 V	-	1	6	uA

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

Table 13-3. Characteristics of Low Voltage Detect circuit

(T_A = – 40 °C to + 85 °C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Hysteresys Voltage of LVD (Slew Rate of LVD)	ΔV	–	-	30	300	mV
Low level detect voltage (KS88C01016/C01008/C01004)	V_{LVD}	–	2.0	2.20	2.40	V
Low level detect voltage (KS88C01116/C01108/C01104)	V_{LVD}	–	1.70	1.90	2.1	V

Table 13-4. Data Retention Supply Voltage in Stop Mode

(T_A = – 40 °C to + 85 °C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data retention supply voltage	V_{DDDR}	–	1.0	–	3.6	V
Data retention supply current	I_{DDDR}	V_{DDDR} = 1.0 V Stop mode	–	–	1	µA

Table 13-5. Input/Output Capacitance

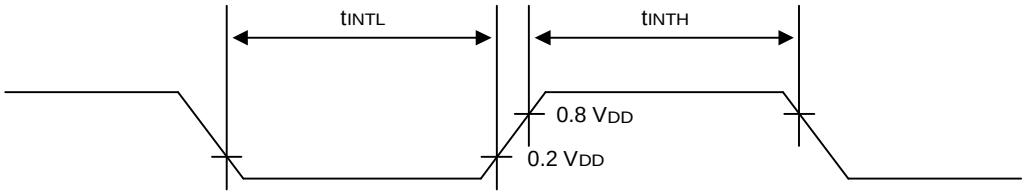
(T_A = – 40°C to + 85°C, V_{DD} = 0 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input capacitance	C_{IN}	f = 1 MHz; unmeasured pins are connected to V_{SS}	—	—	10	pF
Output capacitance	C_{OUT}					
I/O capacitance	C_{IO}					

Table 13-6. A.C. Electrical Characteristics

(T_A = – 40°C to + 85°C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Interrupt input, High, Low width	t_{INTH}, t_{INTL}	P0.0–P0.7, V_{DD} =3.6 V	200	300	—	ns



NOTE: The unit t_{CPU} means one CPU clock period.

Figure 13-1. Input Timing for External Interrupts (Port 0)

Table 13-7. Oscillation Characteristics

(T_A = – 40°C + 85°C)

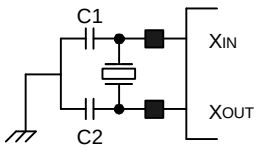
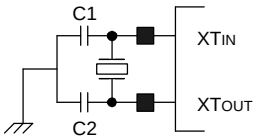
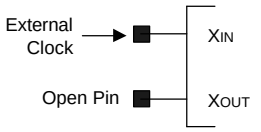
Oscillator	Clock Circuit	Conditions	Min	Typ	Max	Unit
Crystal		CPU clock oscillation frequency	1	–	8	MHz
Ceramic		CPU clock oscillation frequency	1	–	8	MHz
External clock		X_{IN} input frequency	1	–	8	MHz

Table 13-8. Oscillation Stabilization Time

(T_A = – 40°C + 85°C, V_{DD} = 3.6 V)

Oscillator	Test Condition	Min	Typ	Max	Unit
Main crystal	f_{OSC} > 400 kHz	–	–	20	ms
Main ceramic	Oscillation stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	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 oscillation stabilization time (t_{WAIT}) when it is released by an interrupt is determined by the setting in the basic timer control register, BTCON.

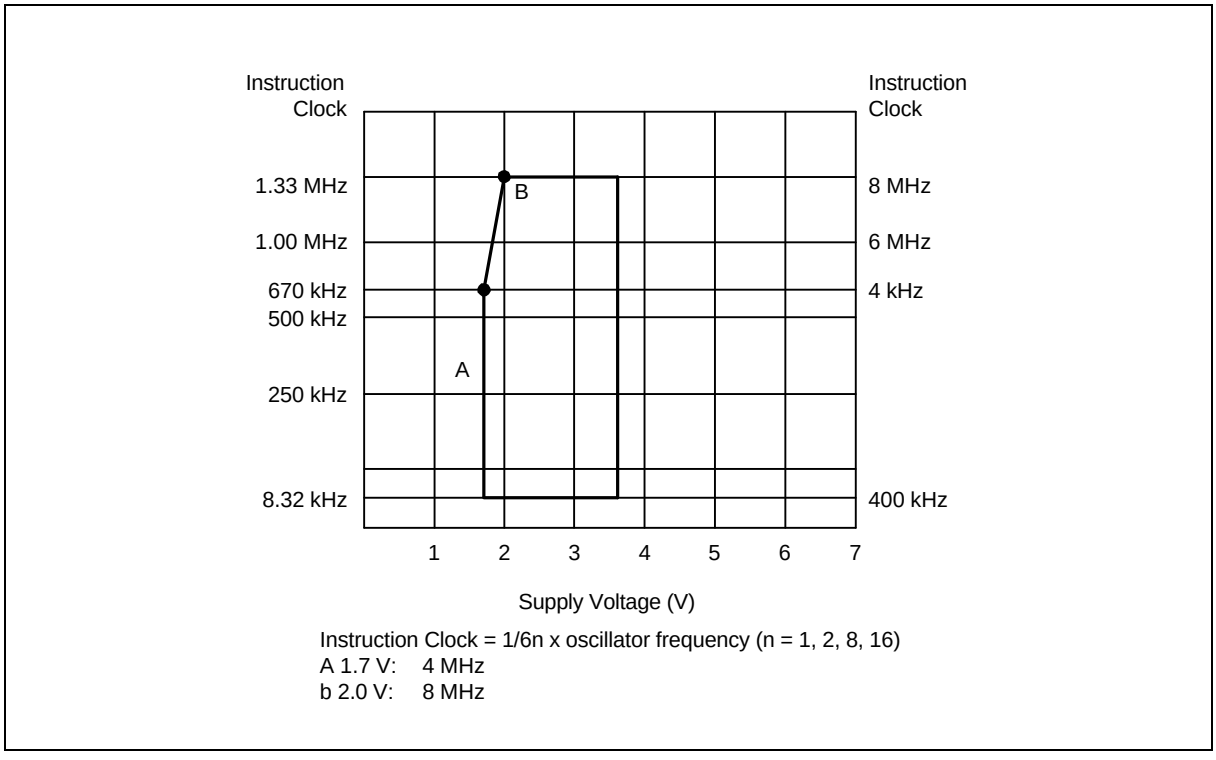


Figure 13-2. Operating Voltage Range of KS88C01016/C01008/C01004/C01116/C01108/C01104

14 MECHANICAL DATA

OVERVIEW

The KS88C01016/C01008/C01004/C01116/C01108/C01104 microcontroller is currently available in a 24-pin SOP and SDIP package.

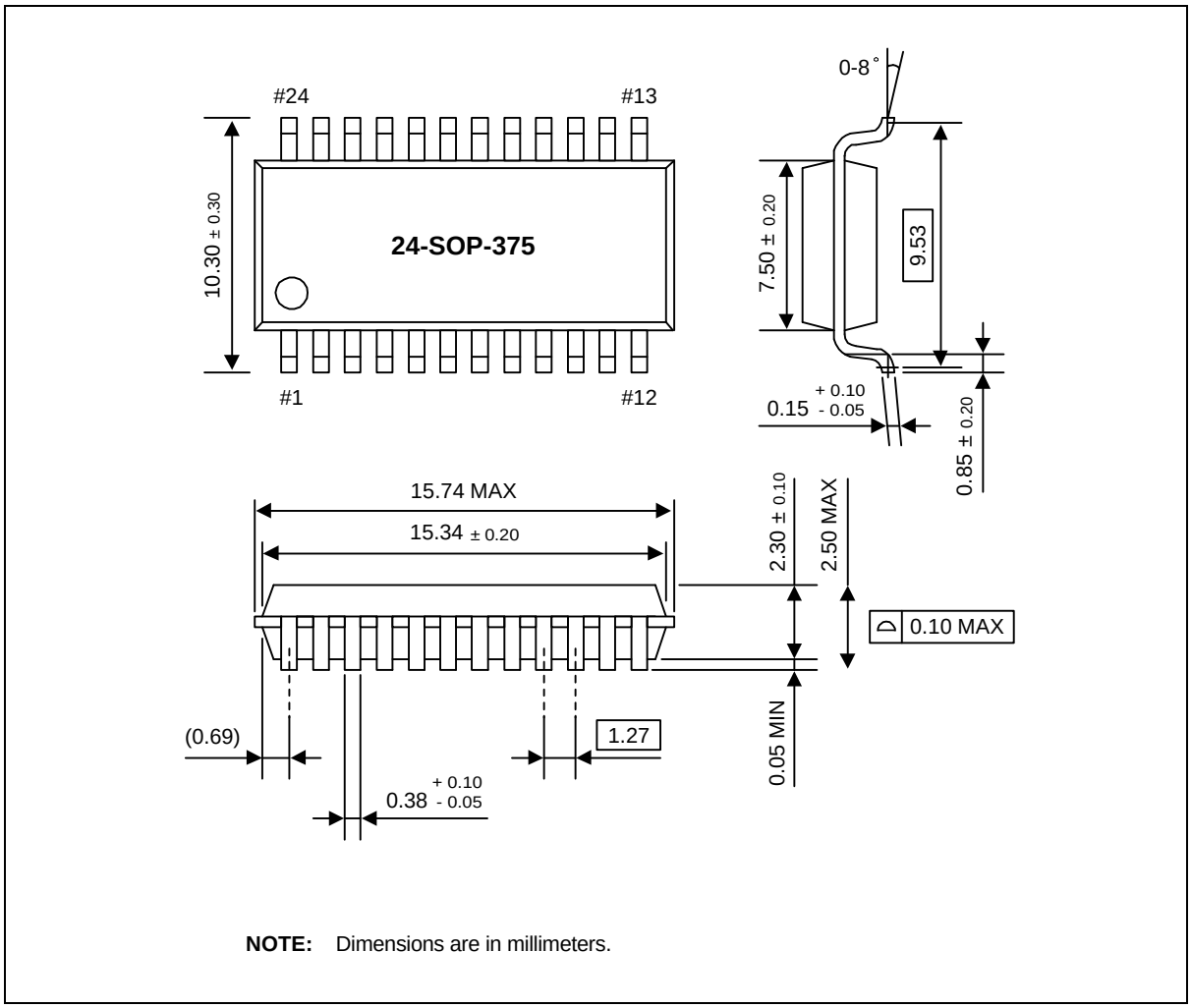


Figure 14-1. 24-Pin SOP Package Mechanical Data

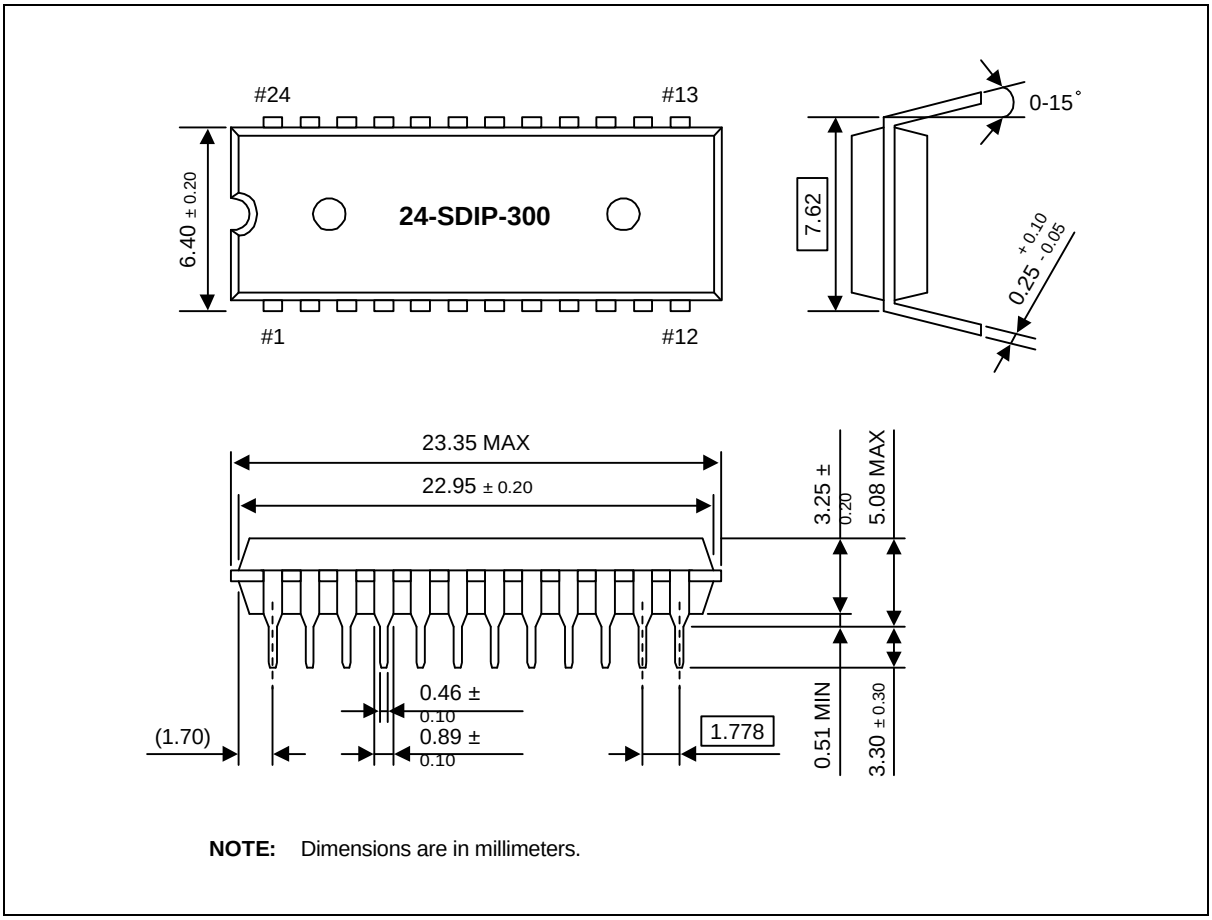


Figure 14-2. 24-Pin SDIP Package Mechanical Data