1 PRODUCT OVERVIEW

SAM87 RC PRODUCT FAMILY

Samsung's new SAM87RC family of 8-bit single-chip CMOS microcontrollers offers a fast and efficient CPU, a wide range of integrated peripherals, and various mask-programmable ROM sizes. Timer/counters with selectable operating modes are included to support real-time operations. Many SAM87RC microcontrollers have an external interface that provides access to external memory and other peripheral devices. The 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.

KS88C4504 MICROCONTROLLER

The KS88C4504 single-chip microcontroller is fabricated using a highly advanced CMOS process. Its design is based on the powerful SAM87RC CPU core. Stop and Idle power-down modes were implemented to reduce power consumption. The size of the internal register file is logically expanded, increasing the addressable on-chip register space to 1040 bytes. A flexible yet sophisticated external interface is used to access up to 64-Kbytes of program and data memory. The KS88C4504 is a versatile microcontroller that is ideal for use in a wide range of general-purpose applications such as CD-ROM/DVD-ROM drives.

Using the SAM87RC modular design approach, the following peripherals were integrated with the SAM87RC CPU core:

— Five configurable 8-bit general I/O ports
— One 2-bit general I/O ports
— Full-duplex serial data port with one synchronous operating modes
— Two 8-bit timers with interval timer
— Two 16-bit timers/counters with PWM operating modes or capture modes
— One voltage level detector pin
— Four embedded chip selection pins (CS0–CS4) or normal I/O ports
— Two programmable 8-bit PWM modules with corresponding output pins
— A/D converter with 4 selectable input pins

OTP

The KS88C4504 microcontroller is also avaiable in OTP(One Time Programmable) version, KS88P4504
The KS88P4504 microcontroller has an on-chip 4K-byte one-time-programable EPROM instead of masked ROM.
The KS88P4504 is comparable To KS88C4504, both in function and in pin configuration.

FEATURES

CPU

- SAM87RC CPU core

Memory

- 1040-byte internal register file
- 4-Kbyte internal program memory

External Interface

- 64K-byte external data memory
- 64K-byte external program memory area (ROMless)
- 60K-byte external program memory and 4K-byte internal program memory

ADC

- Can be used as a general input/output port
- 8-bit resolution four channels

SIO

- 8-bit transmit/receive mode
- 8-bit receive mode
- LSB-first or MSB-first transmission selectable
- Internal or external clock mode

8-bit Timers

- Two 8-bit timers with interval timer mode (Timer A and B)

16-bit Timer/Counters

- Two programmable 16-bit timer/counters
- Interval, or event counter mode operation
- 16-bit capture and 16-bit PWM mode
- Internal or external clock source

Basic Timer (Watchdog Timer)

- Overflow signal makes a system reset
- 8-bit timer with interval timer mode

General I/O Ports

- Five 8-bit general I/O ports (port 0, 1, 2, 3, 4)
- One 2-bit general I/O port (port 5)
- Port 2 can drive LED directly

Interrupts

- Six edge-driven external interrupts
- Two level-driven external interrupts
- Fast interrupt mode processing

PWM

- Four output channels (PWM0, PWM1, TCPWM, TDPWM)
- 8-bit resolution with a 4-bit prescaler (PWM0, PWM1)
- From 16-bit counter (Timer C/D) (TCPWM, TDPWM)

Embedded chip selection

- To reduce interface glue logic, chip selection logic is bold

Voltage level detector

- To prevent MCU from malfunctioning in an unstable power level, a voltage level detector circuit is inserted

Operating Voltage Range

- 2.7 V to 5.5 volts (@12 MHz)

Operating Temperature Range

- – 40 °C to + 85 °C

Package Types

- 80-pin QFP or TQFP

Operating frequency

- 25 MHz (4.5 V to 5.5 V)

BLOCK DIAGRAM

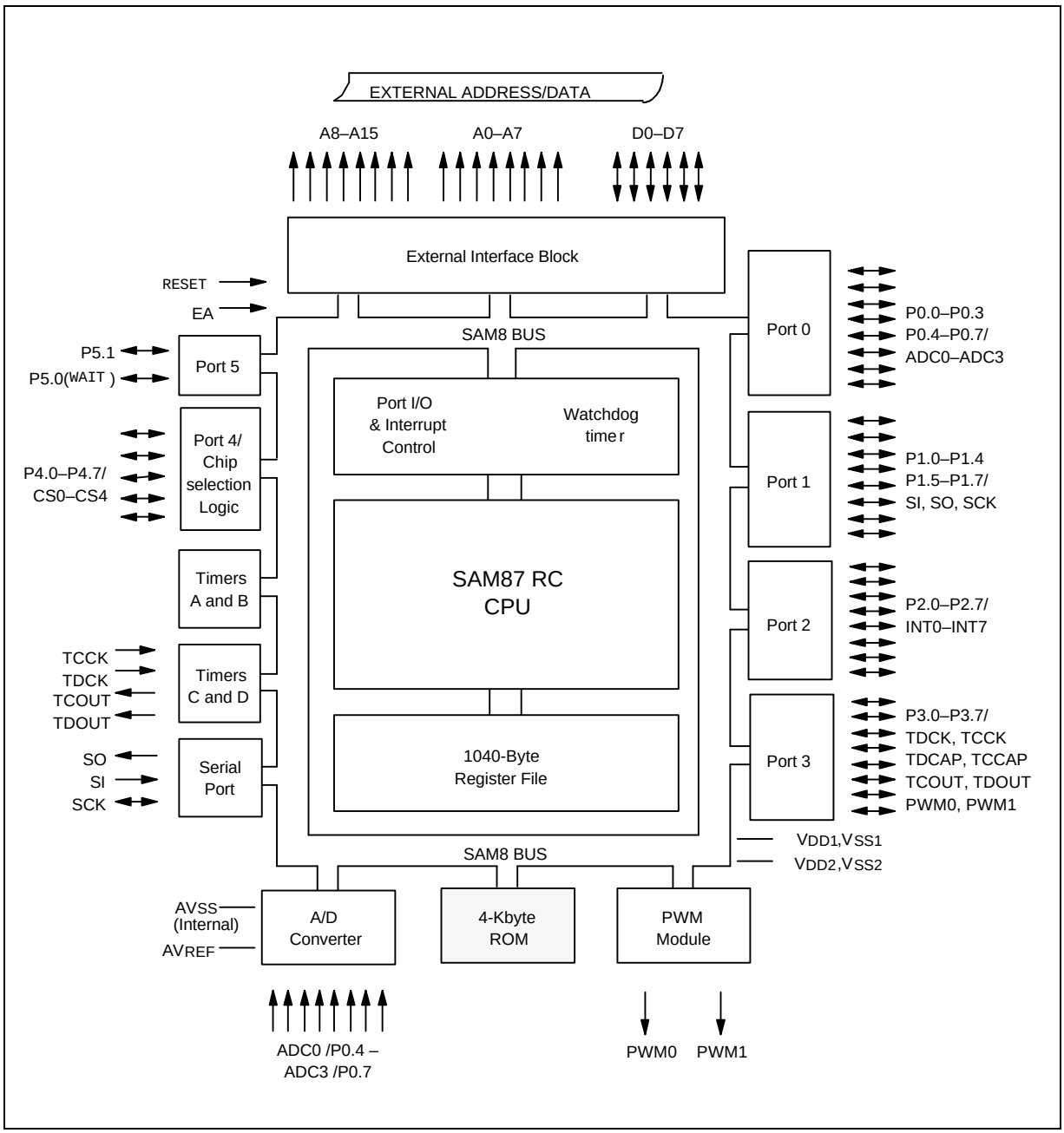


Figure 1-1. KS88C4504 Block Diagram

PIN ASSIGNMENT

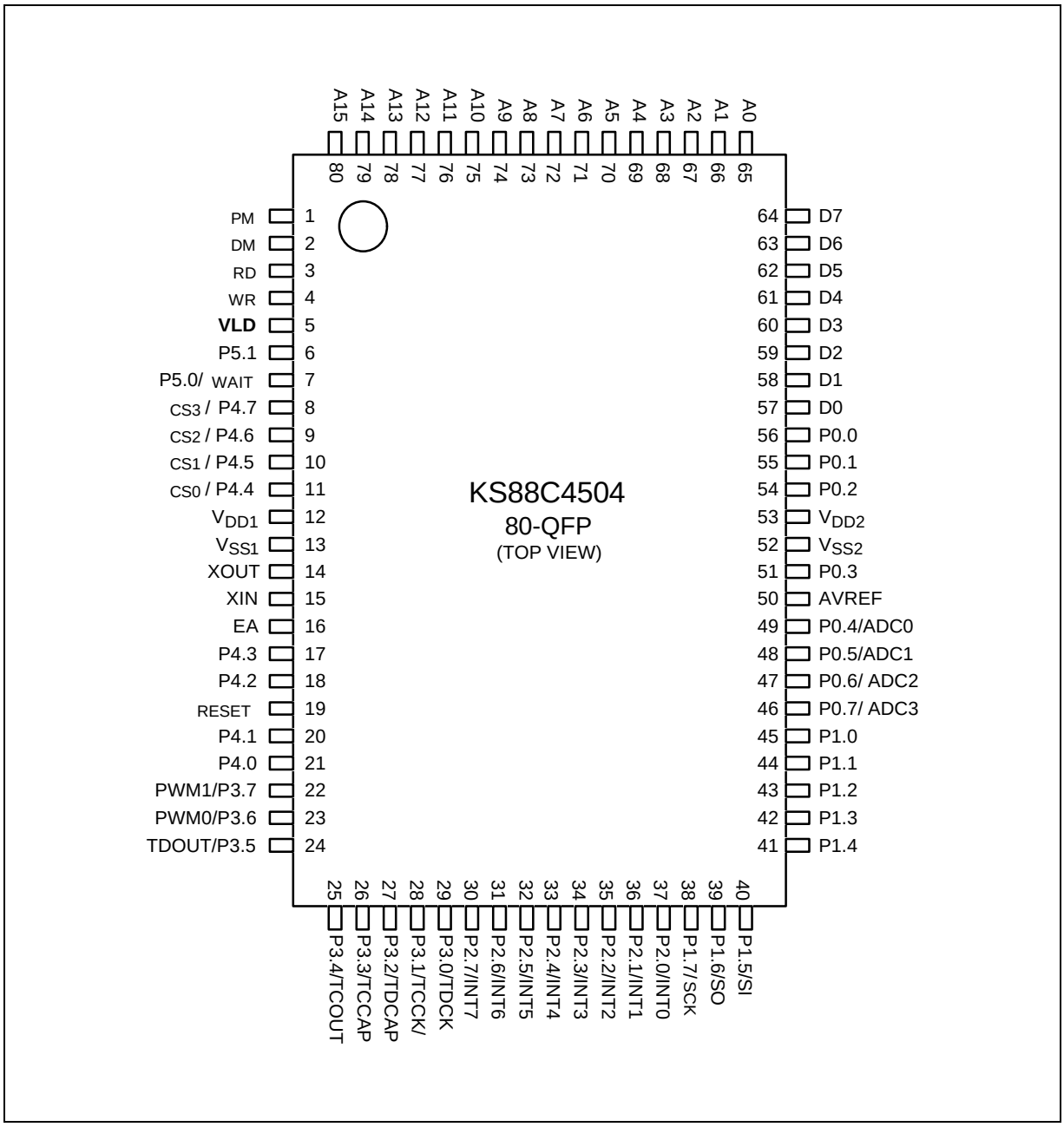


Figure 1-2. KS88C4504 Pin Assignments

PIN ASSIGNMENTS (Continued)

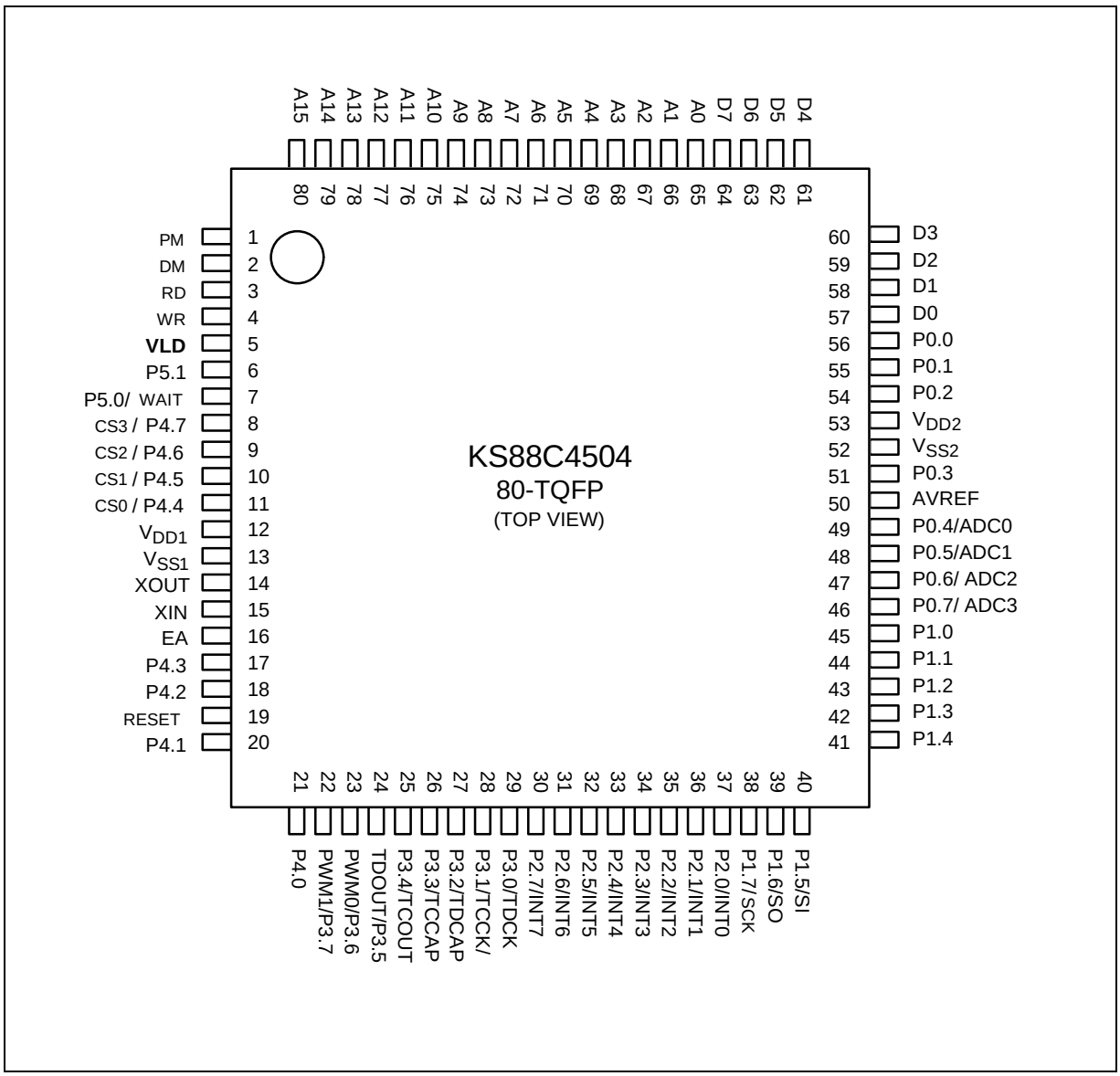


Figure 1-3. KS88C4504 Pin Assignments

PIN DESCRIPTIONS

Table 1-1. KS88C4504/P4504 Pin Descriptions

Pin Name	Pin Type	Pin Description	Circuit Type	Pin Number	Share Pins
P0.0–P0.7	I/O	Bit programmable port; input or output mode selected by software; normal input or push-pull output with software assignable pull-up (P0.0–P0.3) or pull-down (P0.4–P0.7). Alternately, P0.4–P0.7 can be use as a ADC input port with 8-bit resolution.	2, 3	56–54, 51, 49–46	ADC0–ADC3
P1.0–P1.7	I/O	Bit programmable port; input or output mode selected by software; normal input or push-pull output with software assignable pull-up. P1.5–P1.7 can be used as a synchronous SIO port P1.5/SI P1.6/SO P1.7/SCK	3	45–38	SI, SO, SCK
P2.0–P2.7	I/O	General I/O port with normal input or push-pull output with software; assignable pull-up. Bit programmable. Alternately, P2.0–P2.7 can be used as inputs for external interrupts, INT0–INT7 (with noise filter and interrupt control). INT0/INT1 is level interrupts.	4	37–30	INT0–INT7
P3.0–P3.7	I/O	General I/O port with bit programmable pins. Normal input or push-pull output with software assignable pull-up. Input or output mode is selectable by software. Respectively, each pin can serve as (with noise filters): P3.0/timer D clock input (TDCK) P3.1/timer C clock input (TCCK) P3.2/timer D capture input (TDCAP) P3.3/timer C capture input (TCCAP) P3.4/timer C out (TCOUT)/PWM out (TCPWM) P3.5/timer D out (TDOUT)/PWM out (TDPWM) P3.6/PWM0 output port P3.7/PWM1 output port	3, 5	29–22	TDCK TCCK TDCAP TCCAP TDOUT/ TDPWM TCOUT/ TCPWM PWM0 PWM1
P4.0–P4.7	I/O	General I/O port with bit programmable pins. Normal input or push-pull output with software assignable pull-up. Input or output mode is selectable by software. P4.0–P4.7 can alternately be used as inputs for embedded chip selection output. P4.4/CS0 P4.5/CS1 P4.6/CS2 P4.7/CS3	3, 5	21, 20, 18, 17, 11–8	CS0–CS3

Table 1-1. KS88C4504/P4504 Pin Descriptions (Continued)

Pin Name	Pin Type	Pin Description	Circuit Type	QFP Pin Number	Share Pins
P5.0–P5.1	I/O	General I/O port with bit programmable pins. Normal input or push-pull, output mode. Alternately It can use as external interface control signal P5.0/WAIT signal	5	7	WAIT
ADC0–ADC3	I	Analog input pins for A/D converter module. Alternatively used as general-purpose I/O	2	49–46	P0.4–P0.7
AV_{REF}	–	A/D converter reference voltage AV_{SS} is connected to ground internally		50	–
PWM0, PWM1	O	Pulse width modulation output pins	5	23,22	P3.6 P3.7
INT0–INT7	I	External interrupt input pins	4	37–30	P2.0–P2.7
TCCK, TDCK	I	External clock input for timer C and timer D	3	28,29	P3.1/P3.0
TCCAP,TDCAP	I	Timer C/ timer D capture input	3	26,27	P3.3/P3.2
WAIT	I	Input pin for the slow memory timing signal from the external interface	5	7	P5.0
RESET	I	System reset pin (pull-up resistor: 240 kΩ)	1	19	–
EA	I	5V: ROMless operating 0V: internal 4K and external 60K addressing mode	–	16	–
V_{DD1}, V_{SS1}	–	Power input pins for CPU operation (internal) and Power input for OTP writing	–	12,13	–
V_{DD2}, V_{SS2}	–	Power input pins for port output (external)	–	53,52	–
X_{IN}, X_{OUT}	–	Main oscillator pins	–	15,14	–
SI,SO,SCK	I/O	synchronous SIO communication port	3	40,39,38	P1.5/P1.6 P1.7
A0–A15	O	Address output for external device	6	65–80	–
D0–D7	I/O	Data I/O for external device	7	57–64	–
PM,DM	O	External memory selection output	–	1,2	–
RD,WR	O	Memory read/write output	–	3,4	–
CS0–CS3	O	Embedded chip selection output	5	11–8	P4.4–P4.7
TCOUT,TDOUT	O	16-bit timer PWM mode output	5	25,24	P3.4, P3.5
VLD	–	Voltage Level Detect Pin	–	5	–

NOTE: VDD1 must be connected to VDD2 in users application circuit, VSS1 & VSS2 also.

PIN CIRCUITS

Table 1-2. Pin Circuit Assignments for the KS88C4504/P4504

Circuit Number	Circuit Type	KS88C4504 Assignments
1	Input	RESET pin
2	I/O	A/D converter input pins, ADC0–ADC3, P0.4–P0.7
3	I/O	Port 0, 1, 3, 4, and 5
4	I/O	P2 (INT0–INT7)
5	I/O	P3 (TDCK, TCCK, TDCAP, TCCAP, TCOUT, TDOUT, TCPWM, TDPWM, PWM0, PWM1)
6	Output	A0–A15,PM, DM, RD, WR
7	I/O	D0–D7

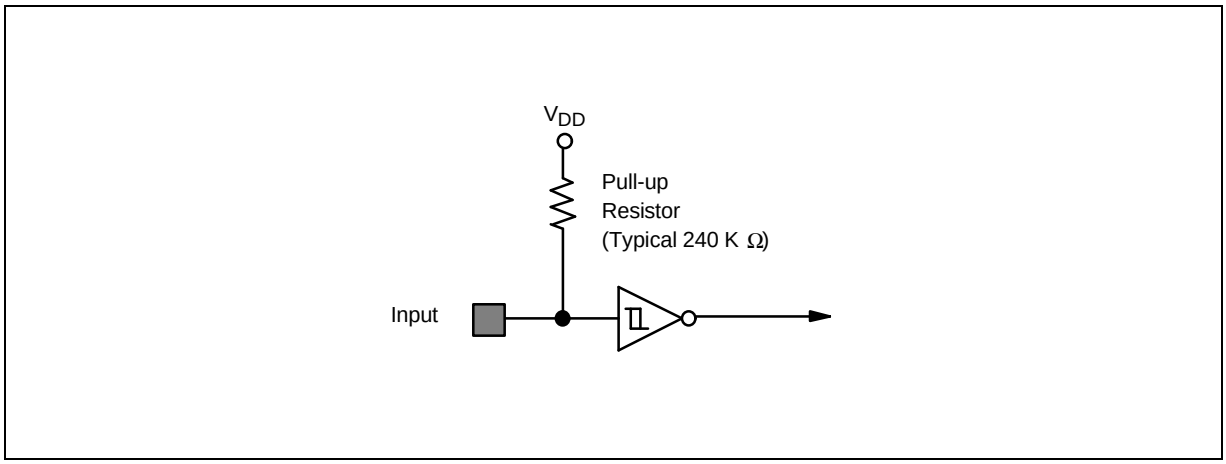


Figure 1-4. Pin Circuit Type 1 (RESET)

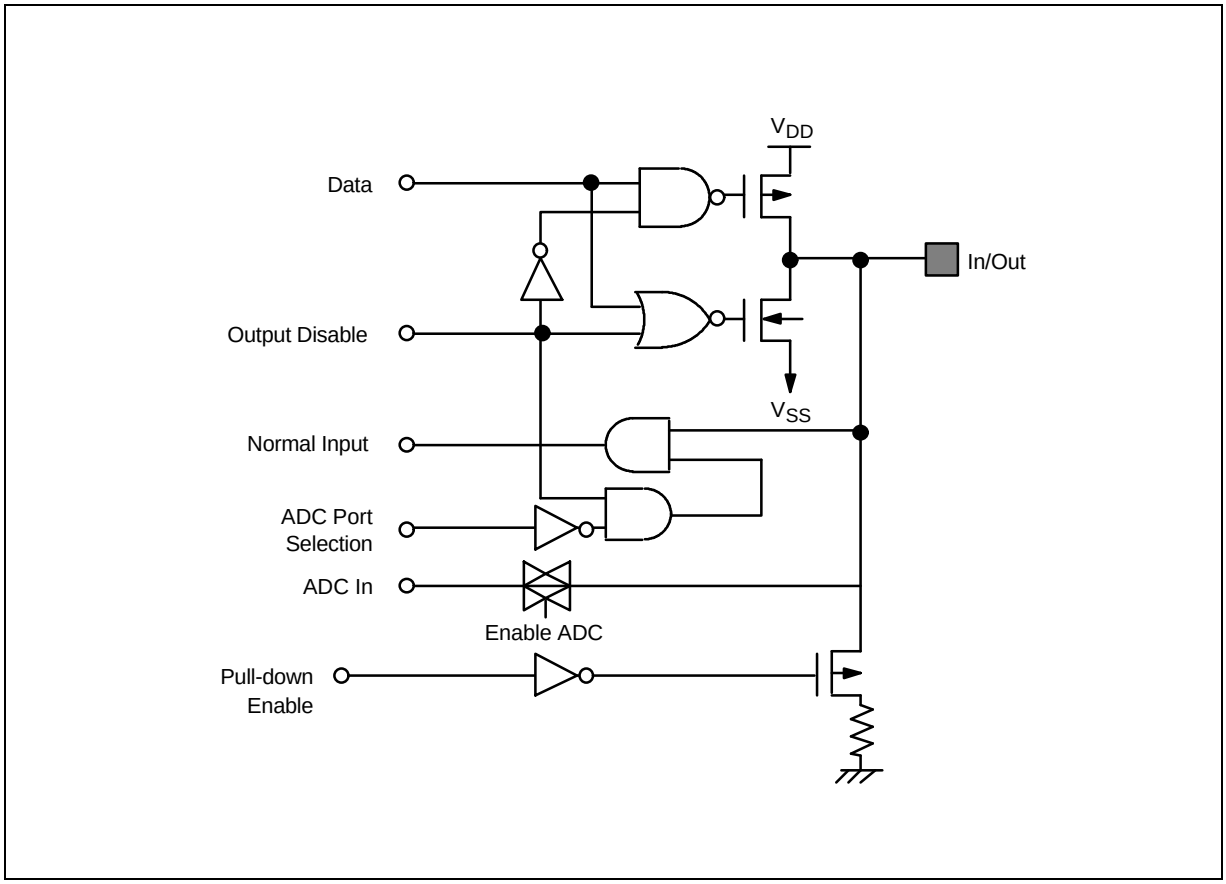


Figure 1-5. Pin Circuit Type 2 (ADC0–ADC3)

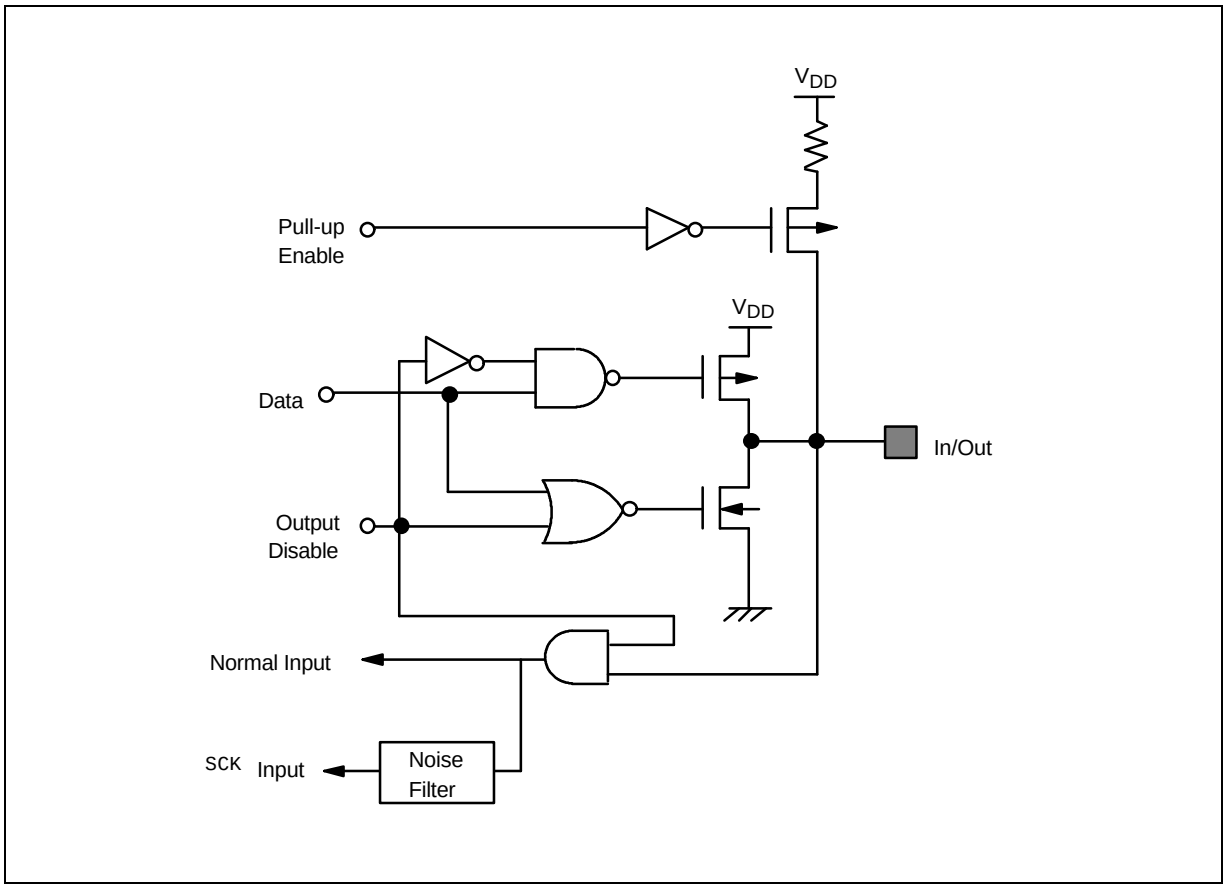


Figure 1-6. Pin Circuit Type 3

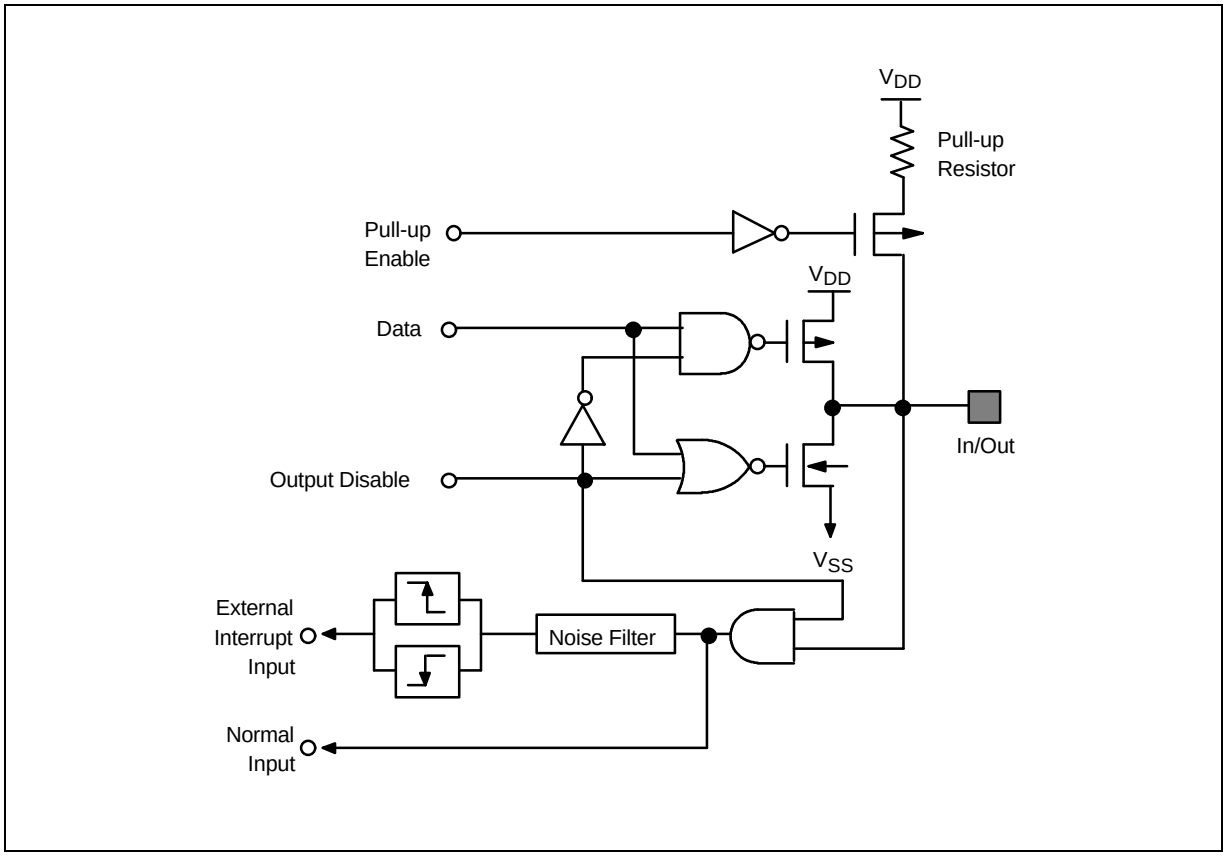


Figure 1-7. Pin Circuit Type 4

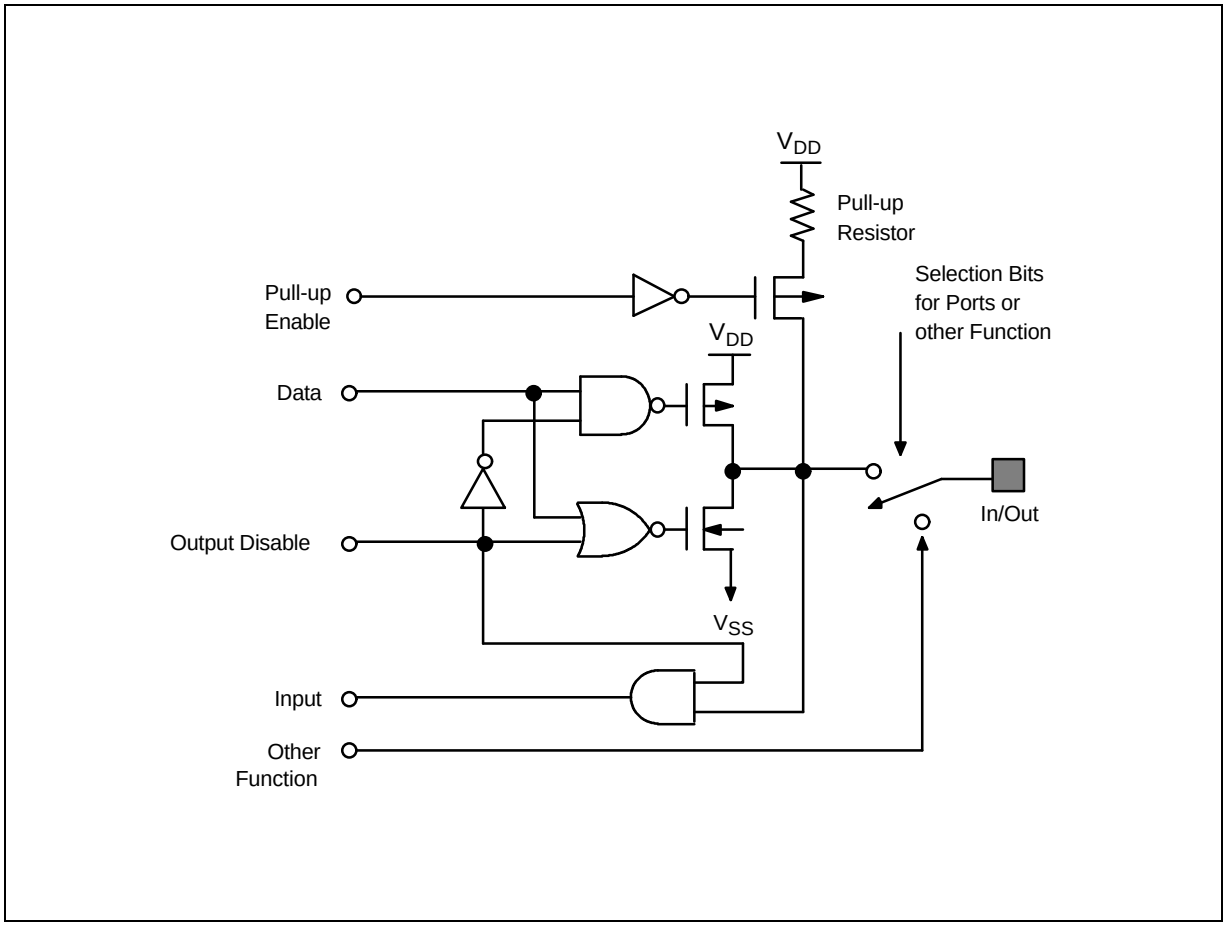


Figure 1-8. Pin Circuit Type 5

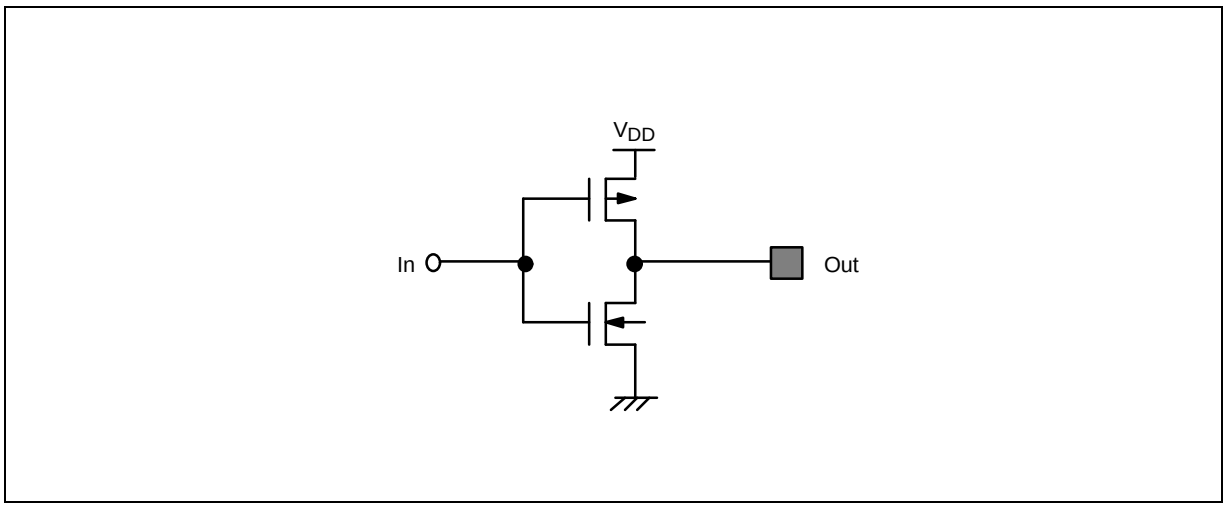


Figure 1-9. Pin Circuit Type 6

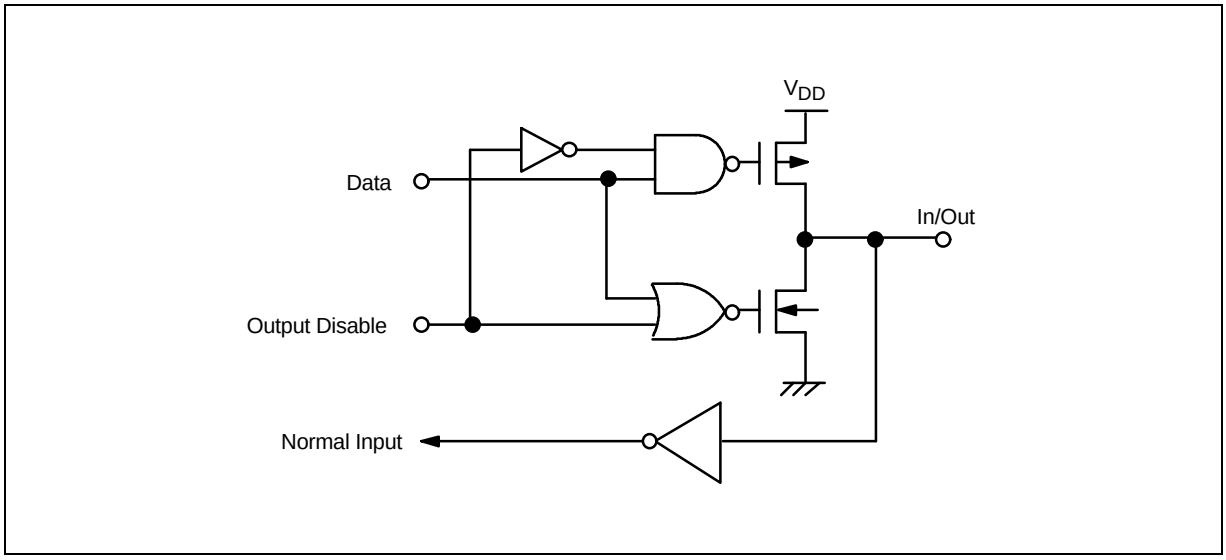


Figure 1-10. Pin Circuit Type 7

NOTES

2 ADDRESS SPACES

OVERVIEW

The KS88C4504/P4504 microcontroller has three kinds of address space:

— External program memory

— External data memory

— Internal register file

A 16-bit address bus supports both external program memory and external data memory operations. Special instructions and related internal logic determine when the 16-bit bus carries addresses for external program memory or for external data memory locations. SAM87RC bus architecture therefore supports up to 64 K bytes of program memory (ROM). Using the external interface, you can address up to 64 K bytes of program memory and 64 K bytes of data memory simultaneously. These spaces can be combined or kept separate.

The KS88C4504/P4504 microcontroller has 1107 registers in its internal register file. A separate 8-bit register bus carries addresses and data between the CPU and the internal register file. The most of these registers can serve as either a source or destination address, or as accumulators for data memory operations. Special 67 bytes of the register file are used for system and peripheral control functions.

PROGRAM MEMORY (ROM)

KS88C4504/P4504 NORMAL OPERATING MODE (MASKED ROM)

Program memory (ROM) stores program code or table data. Instructions can be fetched, or data read, from the ROM. The KS88C4504/P4504 has 4 K bytes (locations 0H–0FFFH) of internal mask-programmable program memory. If your application requires more than 4 K bytes of program memory, you can use ROM-less operating mode to configure up to 64 K bytes of external ROM.

In normal operating mode, it is also possible to access up to 60 K bytes of program memory externally over the external memory interface. The 4-Kbyte on-chip ROM is accessed when program memory locations 0H–0FFFH are addressed and the external interface is used whenever locations 1000H–FFFFH are addressed. This configuration may not, however, be practical or cost-effective.

The SAM8 interrupt structure supports up to 127 vector addresses. As shown in Figure 2-1, the first 256 bytes of the ROM (0H–FFH) are reserved for this maximum number of vectors. Unused locations in this address range can be used as normal program memory. The reset address in the ROM is 0100H.

If the vector address area is used to store normal program data, care must be taken to avoid overwriting vector addresses stored in these locations. For detailed information about interrupt vector addresses, please refer to Section 5, "Interrupt Structure."

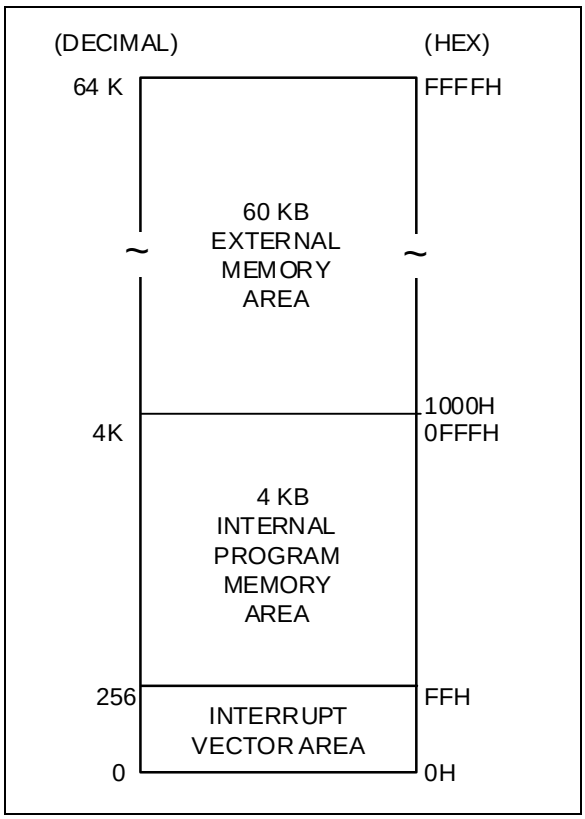


Figure 2-1. Program Memory Map in Normal Operating Mode (EA = "0")

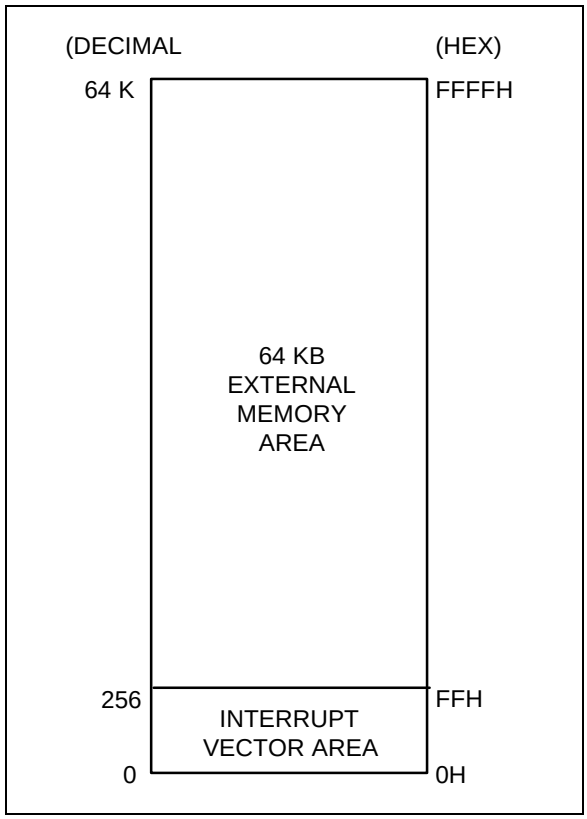


Figure 2-2. Program Memory Map in ROM-less Operating Mode (EA = "1")

KS88C4504/P4504 ROM-LESS OPERATING MODE

The KS88C4504/P4504 microcontroller can also be used as a ROM-less device with the entire program memory space configured externally. Access to the internal program memory area is disabled during ROM-less operating mode. The EA pin is used to select from three operating modes. Which mode is activated depends upon the voltage that is input at this pin:

— When 0 V is input at the EA pin, the KS88C4504/P4504's internal ROM is configured normally and the 4-Kbyte space (0H–0FFFH) is addressed. For normal operation, the EA pin should be connected to V_{SS}. (Up to 60 Kbytes of additional program memory space can be accessed externally in this mode, if required.)

— When 5 V is input at the EA pin prior to a reset, the KS88C4504/P4504 operates in ROM-less mode. Any access to a program memory location goes out over the 16-bit external address bus. Up to 64 Kbytes of external ROM can be accessed in ROM-less mode.

— When 9 V to 10 V is input at the EA pin, factory test mode is activated. (This mode is not intended for customer use.)

NOTE

When the EA pin is tied to V_{SS}, a reset always selects the normal (internal ROM) operating mode.

The external interface is not configured automatically in normal operating mode. In this case, an initialization routine must set the necessary control register values to configure the external interface. When initialization is complete, a Jump or Load instruction can address up to 60 Kbytes of external program memory (locations 1000H–FFFFH) in addition to the 4-Kbyte internal ROM.

ROM-less mode is only activated when 5 V is constantly applied at the EA pin. Please note that the 5 V input must be applied *before* power-on or reset.

Also, whichever operating mode is used, the input voltage at the EA pin (either 0 V for normal operation or 5 V for ROM-less operation) must remain constant.

REGISTER ARCHITECTURE

In the KS88C4504/P4504 implementation, the upper 64 bytes of the 256-byte physical register file is divided into two 64-byte areas, called *set 1* and *set 2*. Set 1 is further divided into two 32-byte register banks (bank 0 and bank 1) and a single 32-byte common area. In addition, the 256-byte area is logically expanded into four separately addressable register pages, *page 0–page 3.* This gives a giving a total of 1024 addressable general-purpose registers.

The 8-bit register bus can address up to 256 bytes (0H–FFH) in any one of the four pages. The register file area is, therefore, 1120 bytes, calculated as 256 bytes × 4 (pages 0–3) + 64 bytes (set 1) + 32 bytes (common area). However, because only 19 bytes are mapped in set 1, bank 1, the total number of addressable 8-bit registers is 1107. Of these 1107 registers, 13 bytes are for CPU and system control registers, 54 bytes are for peripheral control and data registers, 16 bytes are used as a shared working registers, and 1024 registers are for general-purpose use.

You can always address set 1 register locations, regardless of which of the four register pages is currently selected. Set 1 locations can, however, only be addressed using indirect addressing modes.

The extension of the physical register space into separately addressable areas (sets, banks, and pages) is supported by various addressing mode restrictions, the select bank instructions, SB0 and SB1, and the register page pointer (PP).

Specific register types and the area (in bytes) that they occupy in the register file are summarized in Table 2-1.

Table 2-1. Register Type Summary

Register Type	Number of Bytes
CPU and system control registers	13
Peripheral, I/O, and clock control/data registers	54
Reserved working register area	16
General-purpose registers	1024
Total Addressable Bytes	1107

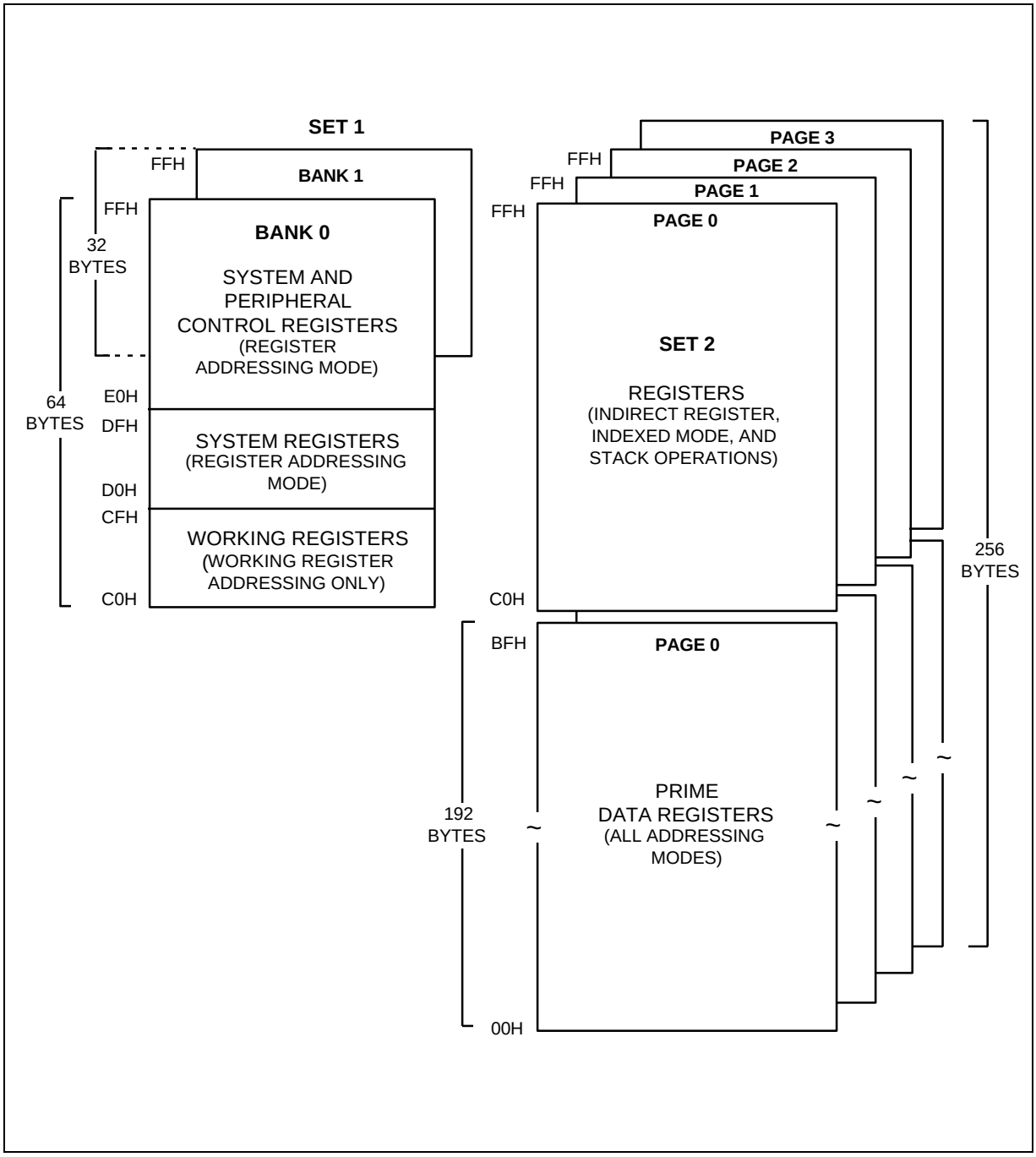


Figure 2-3. Internal Register File Organization

REGISTER PAGE POINTER (PP)

In the KS88C4504/P4504, the physical area of the internal register file is logically expanded by the additional of four register pages. Page addressing is controlled by the register page pointer (PP, DFH). See Figure 2-4.

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.

Whenever you select a different page, the current 256-byte address area (0H–FFH) is logically switched with the address range of the new page (see section 4 "PP" register for more information).

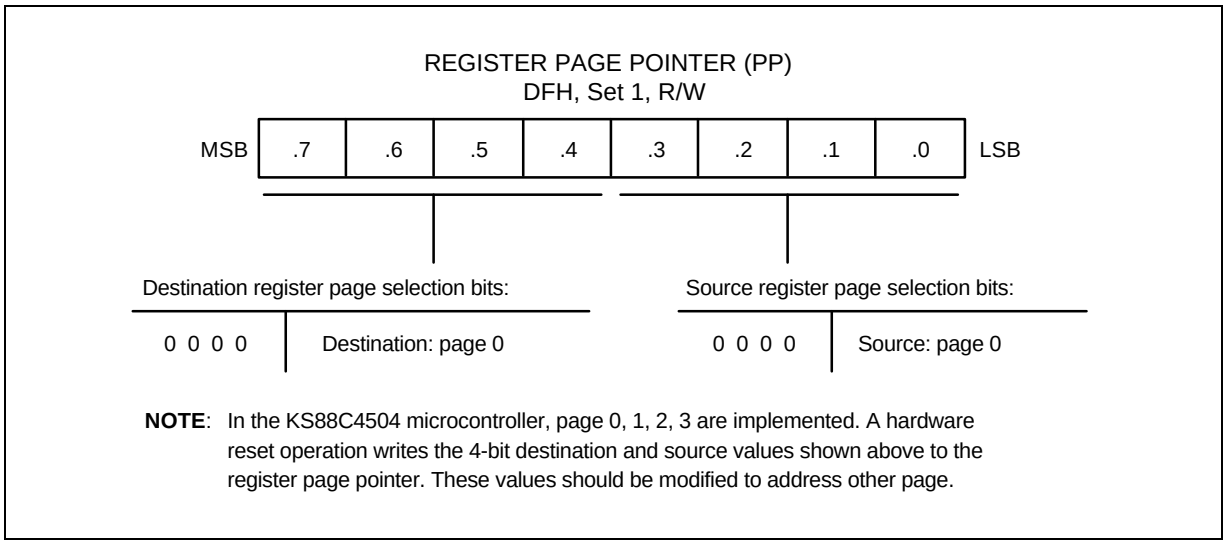


Figure 2-4. Register Page Pointer (PP)

NOTE: You should refer to page 6-39 and use DJNZ instruction properly when DJNZ instruction is used in your program.

REGISTER SET 1

The term *set 1* refers to the upper 64 bytes of the register file, locations C0H–FFH. This area can be accessed at any time, regardless of which page is currently selected.

The upper 32-byte area of this 64-byte space is divided into two 32-byte register banks, called *bank 0* and *bank 1*. You use the select register bank instructions, SB0 or SB1, to address one bank or the other. A reset operation automatically selects bank 0 addressing.

The lower 32-byte area of set 1 is not banked. This area contains 16 bytes for mapped system registers (D0H–DFH) and a 16-byte common area (C0H–CFH) for working register addressing.

Registers in set 1 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, however.

Working register addressing is a function of Register addressing mode (see Section 3, "Addressing Modes," for more information).

REGISTER SET 2

The same 64-byte physical space that is used for set 1 register locations C0H–FFH is logically duplicated to add another 64 bytes. This expanded area of the register file is called *set 2*. For the KS88C4504/P4504, the set 2 address range (C0H–FFH) is accessible on pages 0–3.

The logical division of set 1 and set 2 is maintained by means of addressing mode restrictions: While you can access set 1 using Register addressing mode only, you can only use Register Indirect addressing mode or Indexed addressing mode to access set 2.

PRIME REGISTER SPACE

The lower 192 bytes (00H–BFH) of the KS88C4504/P4504's four 256-byte register pages is called *prime register area.* Prime registers can be accessed using any of the seven addressing modes (see Section 3, "Addressing Modes").

The prime register area on page 0 is immediately addressable following a reset. In order to address prime registers on pages 1, 2, or 3, you must set the register page pointer (PP) to the appropriate source and destination values.

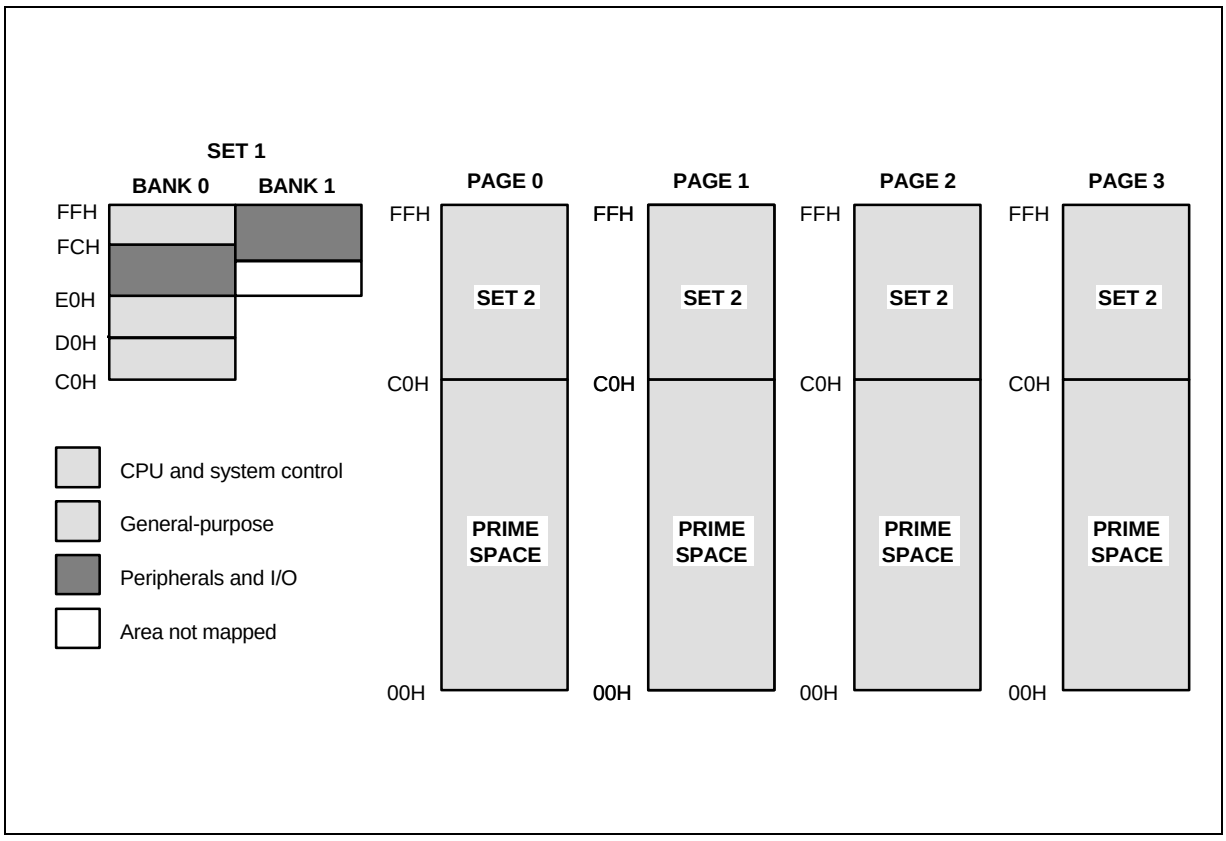


Figure 2-5. Map of Set 1, Set 2, and Prime Register Spaces

WORKING REGISTERS

Instructions can access specific 8-bit registers or 16-bit register pairs using either 4-bit or 8-bit address fields. When 4-bit working register addressing is used, the 256-byte register file can be viewed by the programmer as 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).

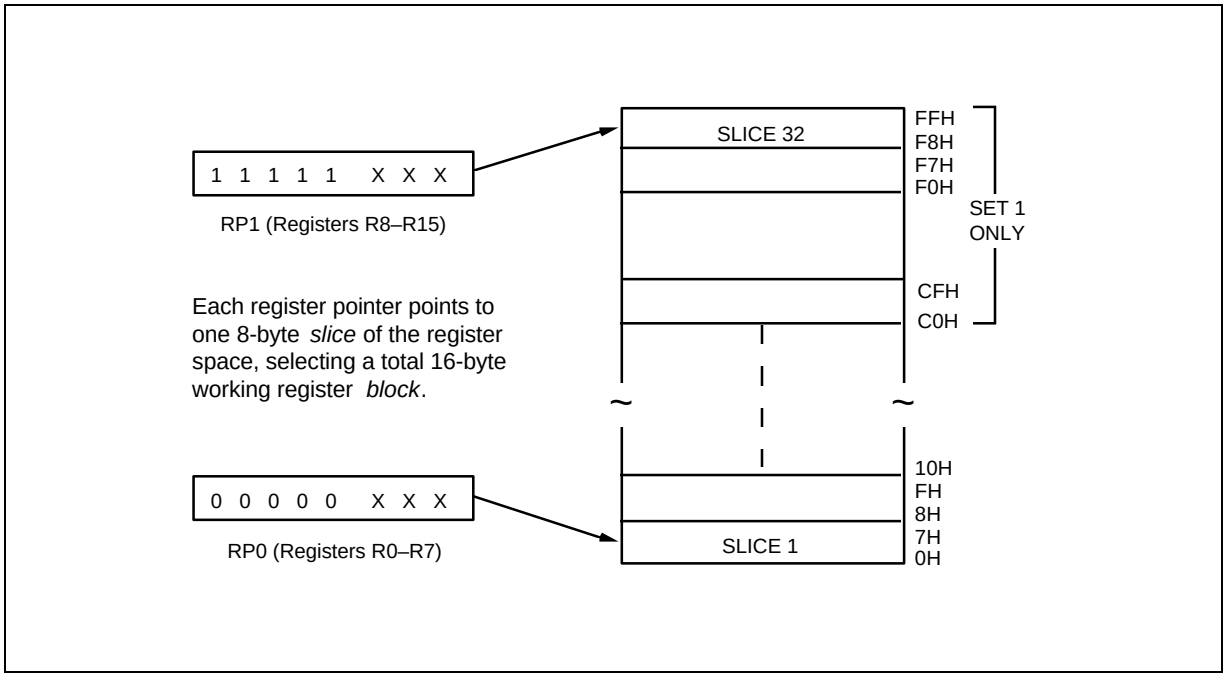


Figure 2-6. 8-Byte Working Register Areas (Slices)

USING THE REGISTER POINTERS

Register pointers RP0 and RP1 are mapped to addresses D6H and D7H in set 1. They 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 2-7 and 2-8).

With working register addressing, you can only access those locations that are pointed to by the register pointers. Please note that you cannot use the register pointers to select working register area in set 2, C0H–FFH, because these locations are accessible 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 2-7).

In some cases, you may need to define working register areas in different (non-contiguous) areas of the register file. In Figure 2-8, 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, definition of the working register area is very flexible.

☞ 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
```

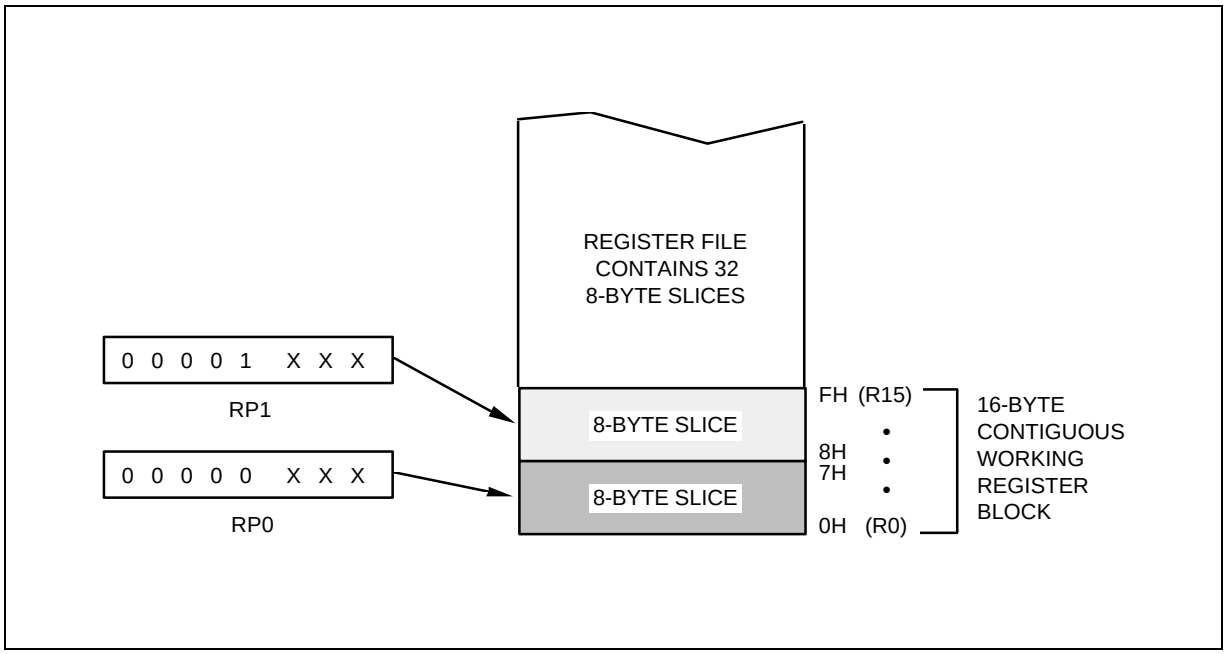


Figure 2-7. Contiguous 16-Byte Working Register Block

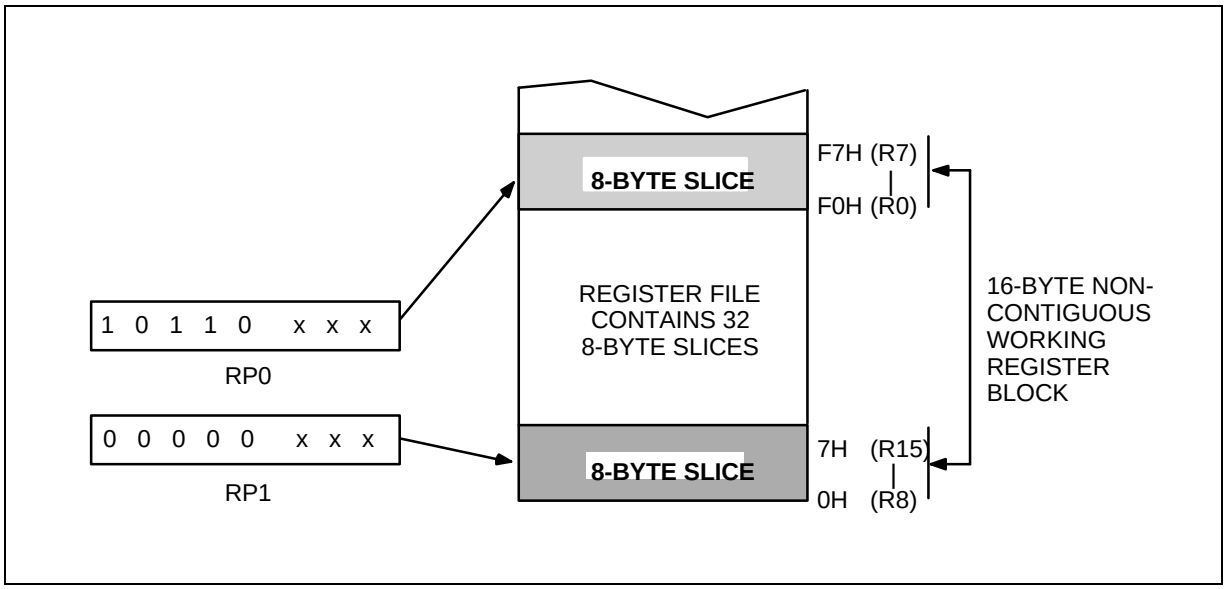


Figure 2-8. Non-Contiguous 16-Byte Working Register Block

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 24 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 30 cycles instead of 24 cycles.

REGISTER ADDRESSING

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

The Register (R) addressing mode, in which the operand value is the content of a specific register or register pair, can be used to access all locations in the register file except for set 2.

For working register addressing, the register pointers RP0 and RP1 are used to select a specific register within a selected 16-byte working register area. To increase the speed of context switches in an application program, you can use the register pointers to dynamically select different 8-byte "slices" of the register file as the program's active working register space.

Registers are addressed either as a single 8-bit register or as a paired 16-bit register. In 16-bit register pairs, the address of the first 8-bit register is always an even number and the address of the next register is 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.

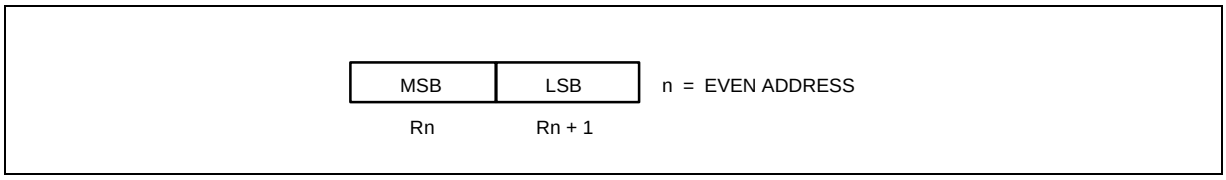


Figure 2-9. 16-Bit Register Pairs

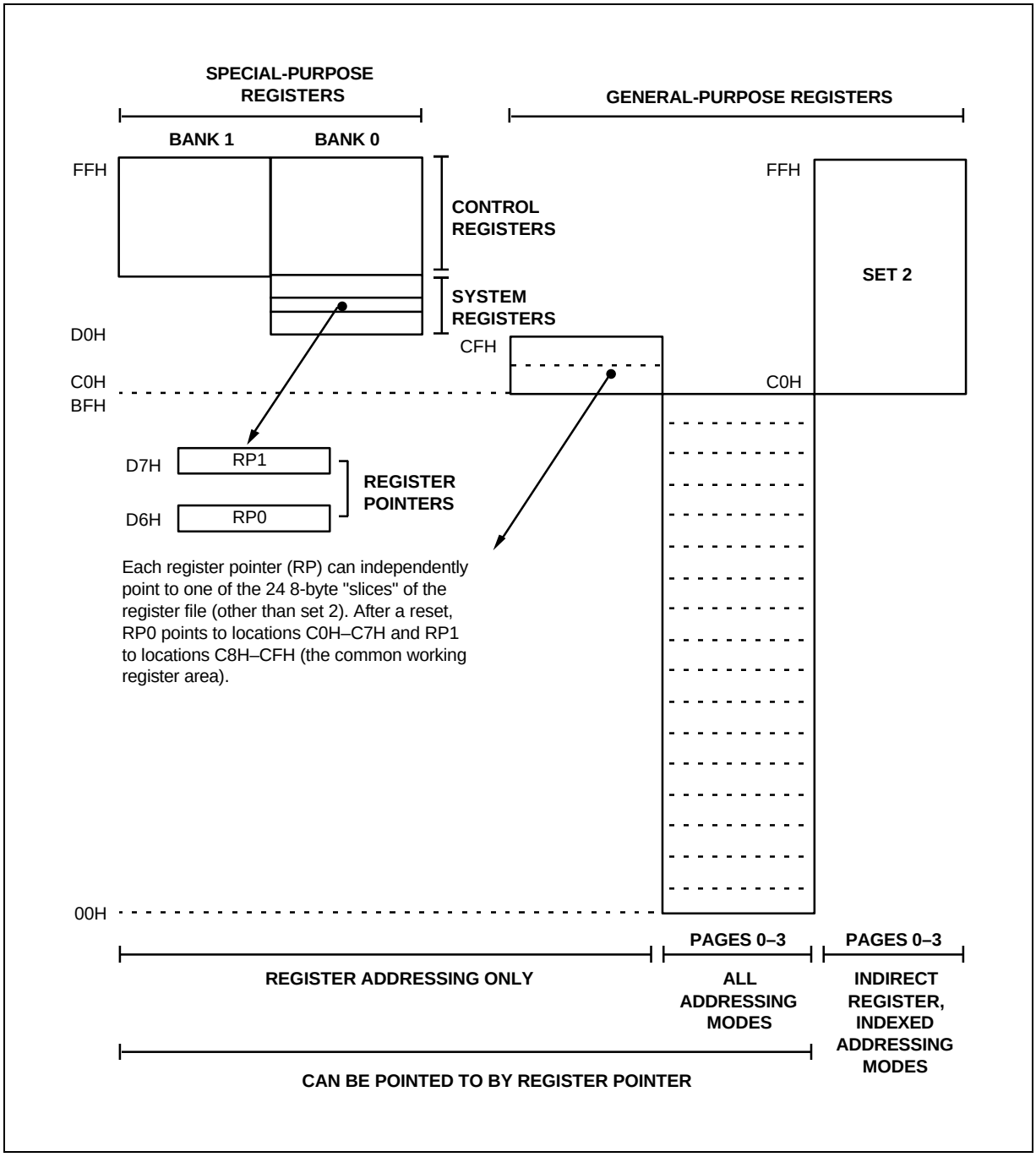


Figure 2-10. Register File Addressing

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 the *common area*. You can use common area registers as working registers for operations that address locations on different pages in the register file.

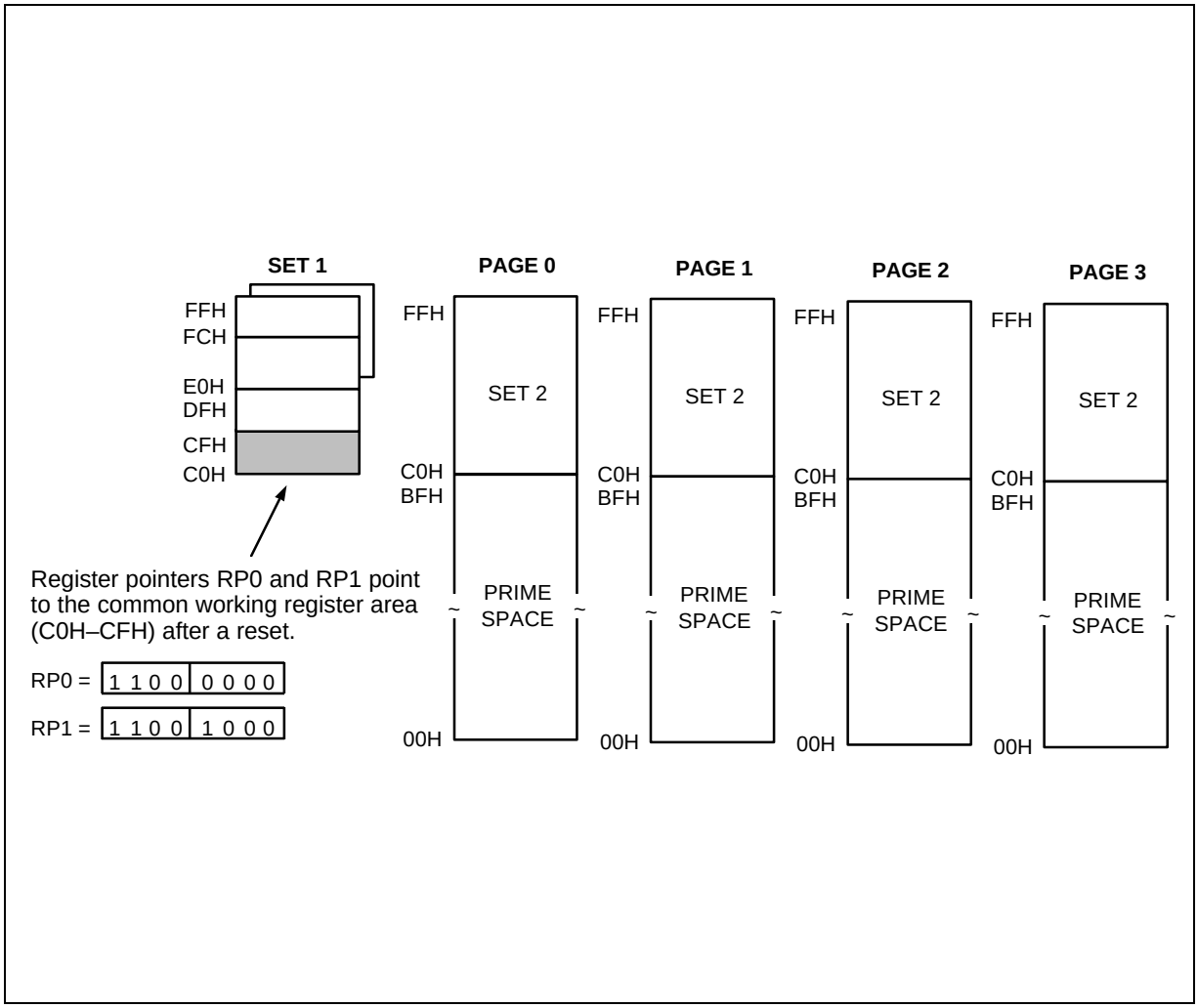


Figure 2-11. Common Working Register Area

☞ 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 enables 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 2-12, the net effect 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 2-13 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).

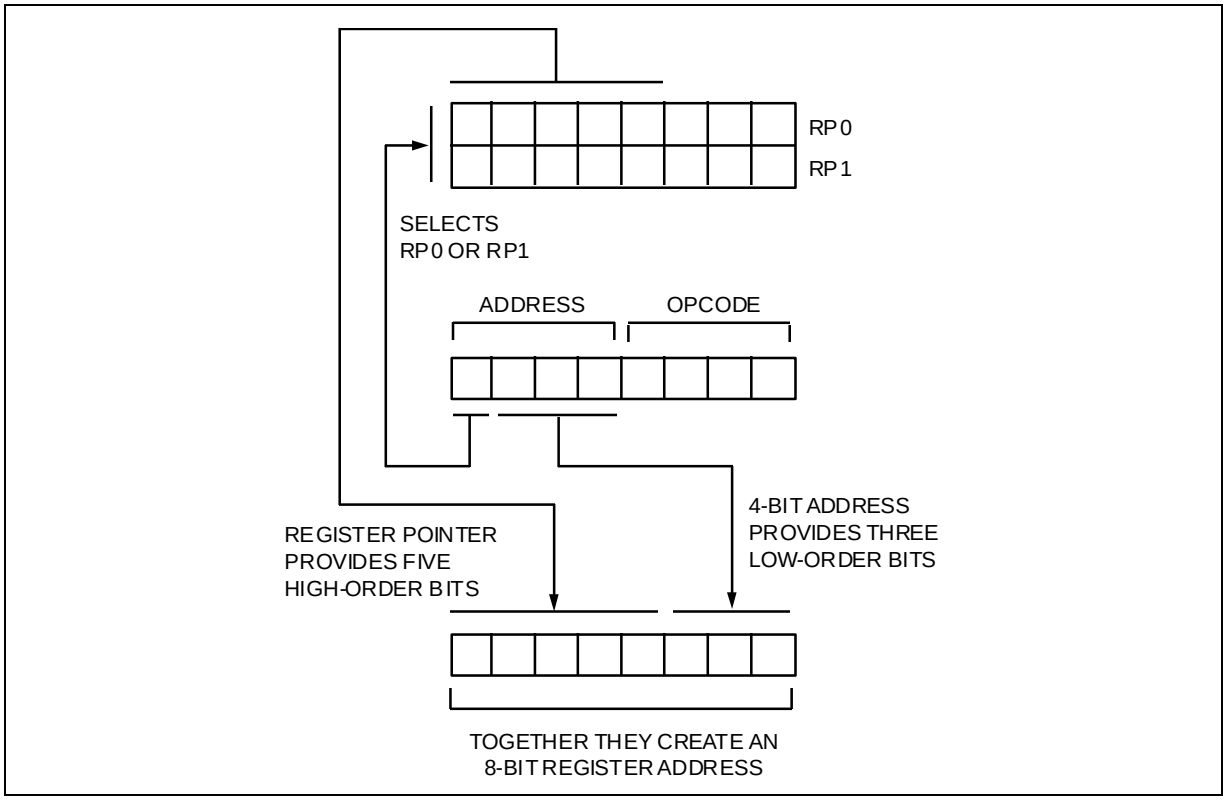


Figure 2-12. 4-Bit Working Register Addressing

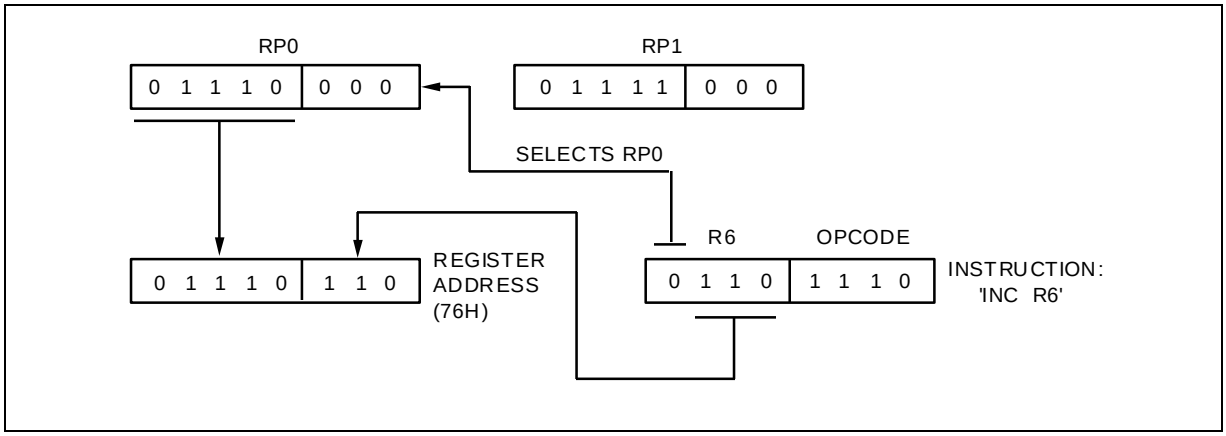


Figure 2-13. 4-Bit Working Register Addressing Example

8-BIT WORKING REGISTER ADDRESSING

You can also use 8-bit working register addressing to access registers in a selected working register area. In order 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 2-14, 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, and the three low-order bits of the complete address are provided by the original instruction.

Figure 2-15 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. The fourth bit ("1") selects RP1 and the five high-order bits in RP1 (10100B) 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. Together, the five address bits from RP1 and the three address bits from the instruction comprise the complete register address, 0ABH (10100011B).

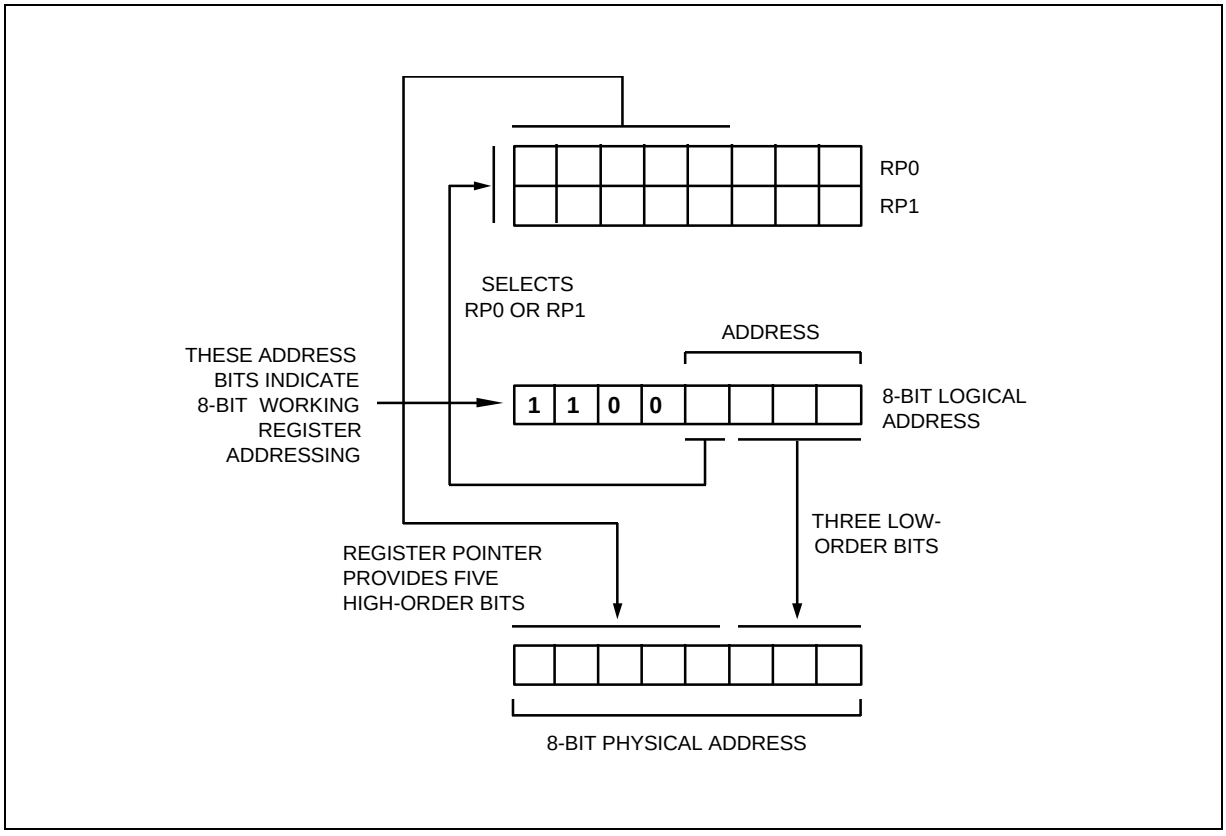


Figure 2-14. 8-Bit Working Register Addressing

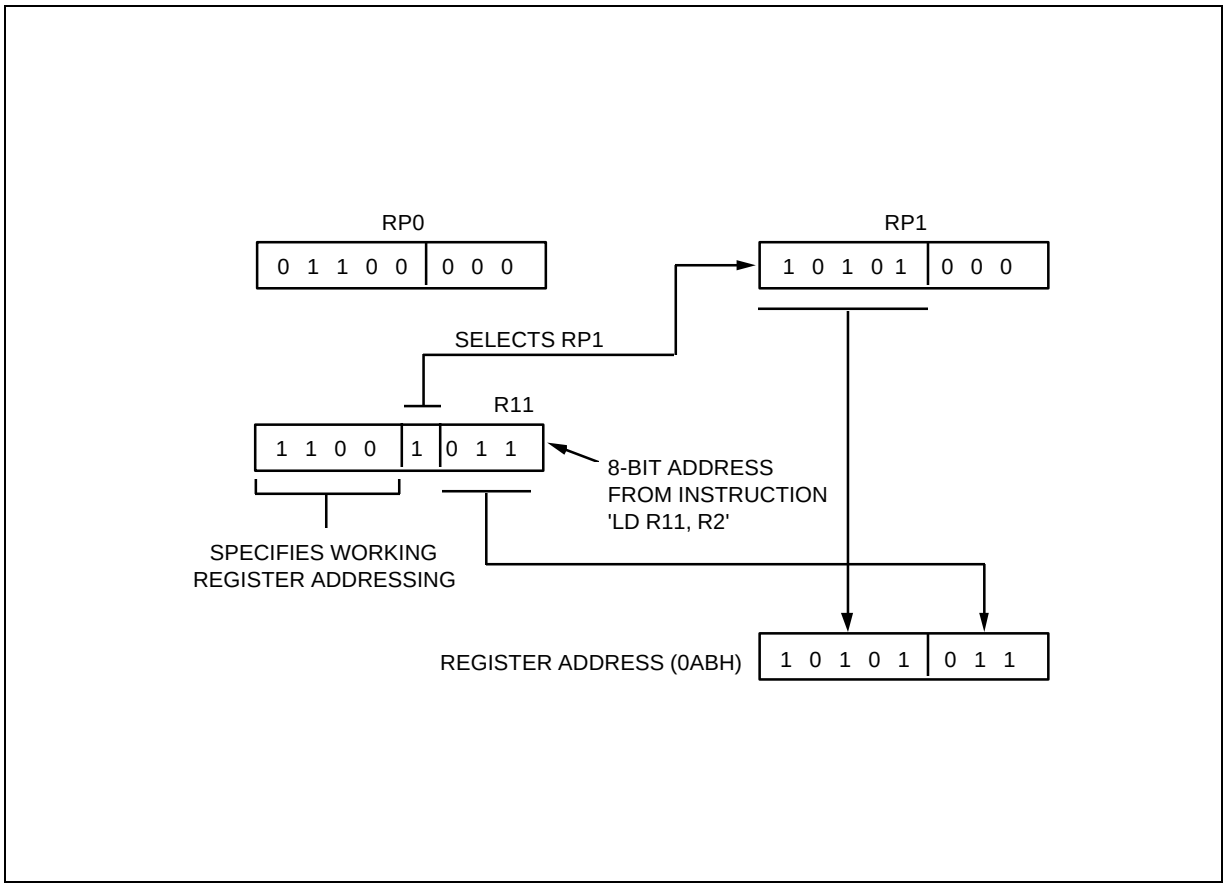


Figure 2-15. 8-Bit Working Register Addressing Example

SYSTEM AND USER STACKS

KS88-series microcontrollers can be programmed to use system stack for subroutine calls, returns, interrupts, and to store data. The PUSH and POP instructions are used to control system stack operations.

The SAM8 architecture supports stack operations in the internal register file as well as in external data memory. To select an internal or external stack area, you set bit 1 of the external memory timing register, EMT.1 to the appropriate value.

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 is always decremented *before* a push operation and incremented *after* a pop operation. The stack pointer (SP) always points to the stack frame stored on the top of the stack, as shown in Figure 2-16.

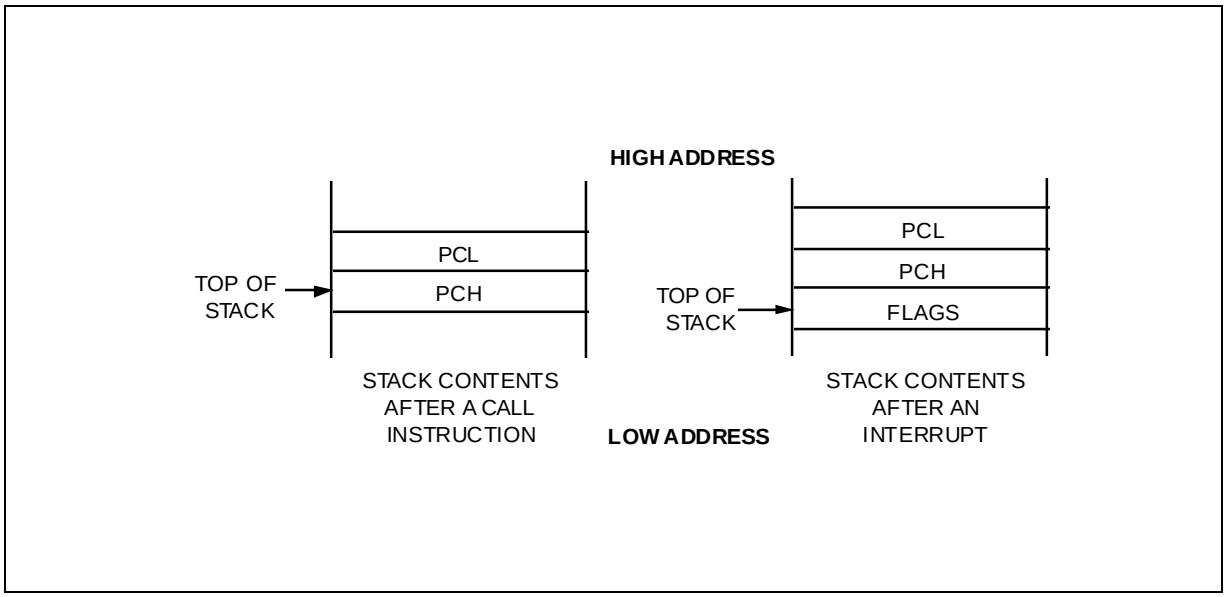


Figure 2-16. Stack Operations

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.

These instructions cannot address external memory locations. Only PUSH and POP instructions can be used for an externally defined stack.

Stack Pointers (SPL, SPH)

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

If only internal memory space is implemented, the SPL must be initialized to an 8-bit value in the range 00H–FFH; the SPH register is not needed (and can be used as a general-purpose register, if needed). If external memory is implemented, both SPL and SPH must be initialized with a full 16-bit address.

When the SPL register contains the only stack pointer value (that is, when it points to a system stack in the register file), the SPH register can be used as a general-purpose data register.

However, if an overflow or underflow condition occurs as the result of incrementing or decrementing the stack address in the SPL register during normal stack operations, the value in the SPL register will overflow (or underflow) to the SPH register, overwriting any other data that is currently stored there.

To avoid overwriting data in the SPH register, you can initialize the SPL value to FFH instead of 00H.

Stack operation page is in only *page 0*, regardless the processing page.

☞ Programming Tip — Standard Stack Operations Using PUSH and POP

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

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

3 ADDRESSING MODES

OVERVIEW

Instructions that are stored in program memory are fetched for execution using the program counter. Instructions indicate the operation to be performed and the data to be operated on.

Addressing mode is the method used to determine the location of the data operand. The operands specified in SAM8 instructions may be condition codes, immediate data, or a location in the register file, program memory, or data memory.

The SAM8 instruction set supports seven explicit addressing modes. Not all of these addressing modes are available for each instruction. The addressing modes and their symbols are as follows:

— 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).

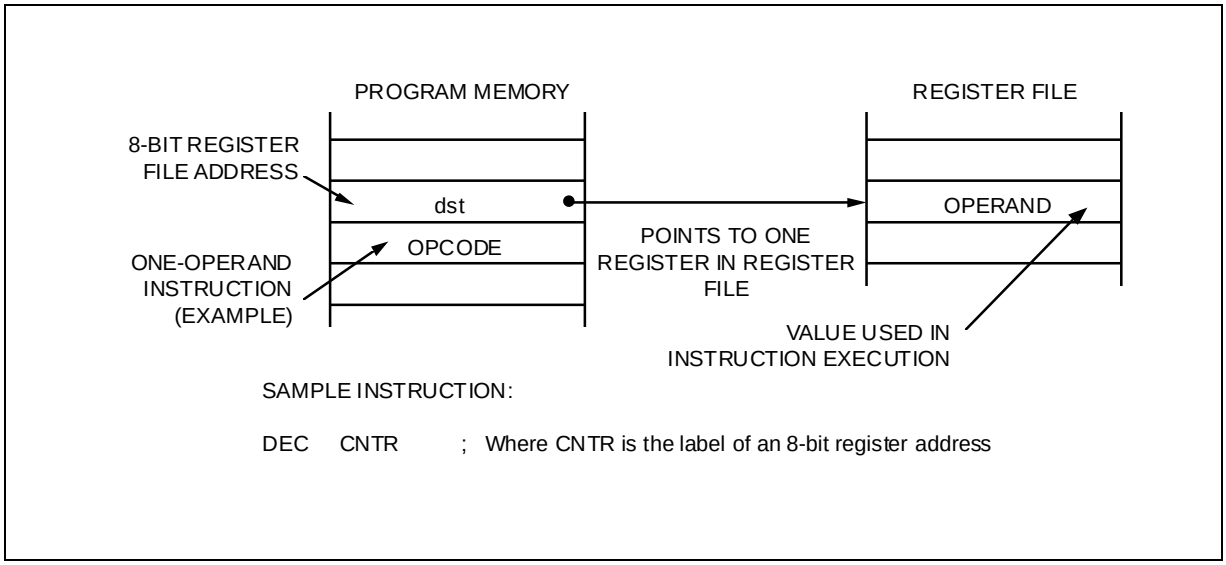


Figure 3-1. Register Addressing

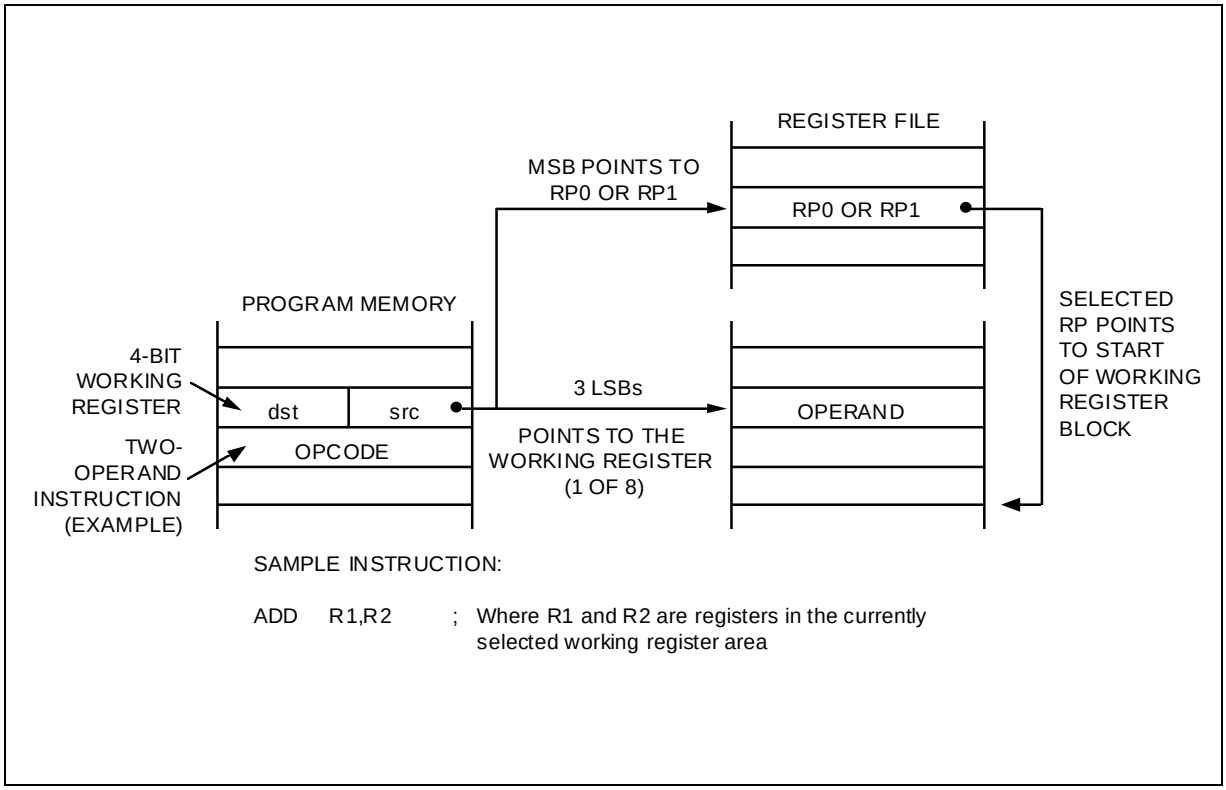


Figure 3-2. Working Register Addressing

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 (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.

You cannot, however, access locations C0H–FFH in set 1 using Indirect Register addressing mode.

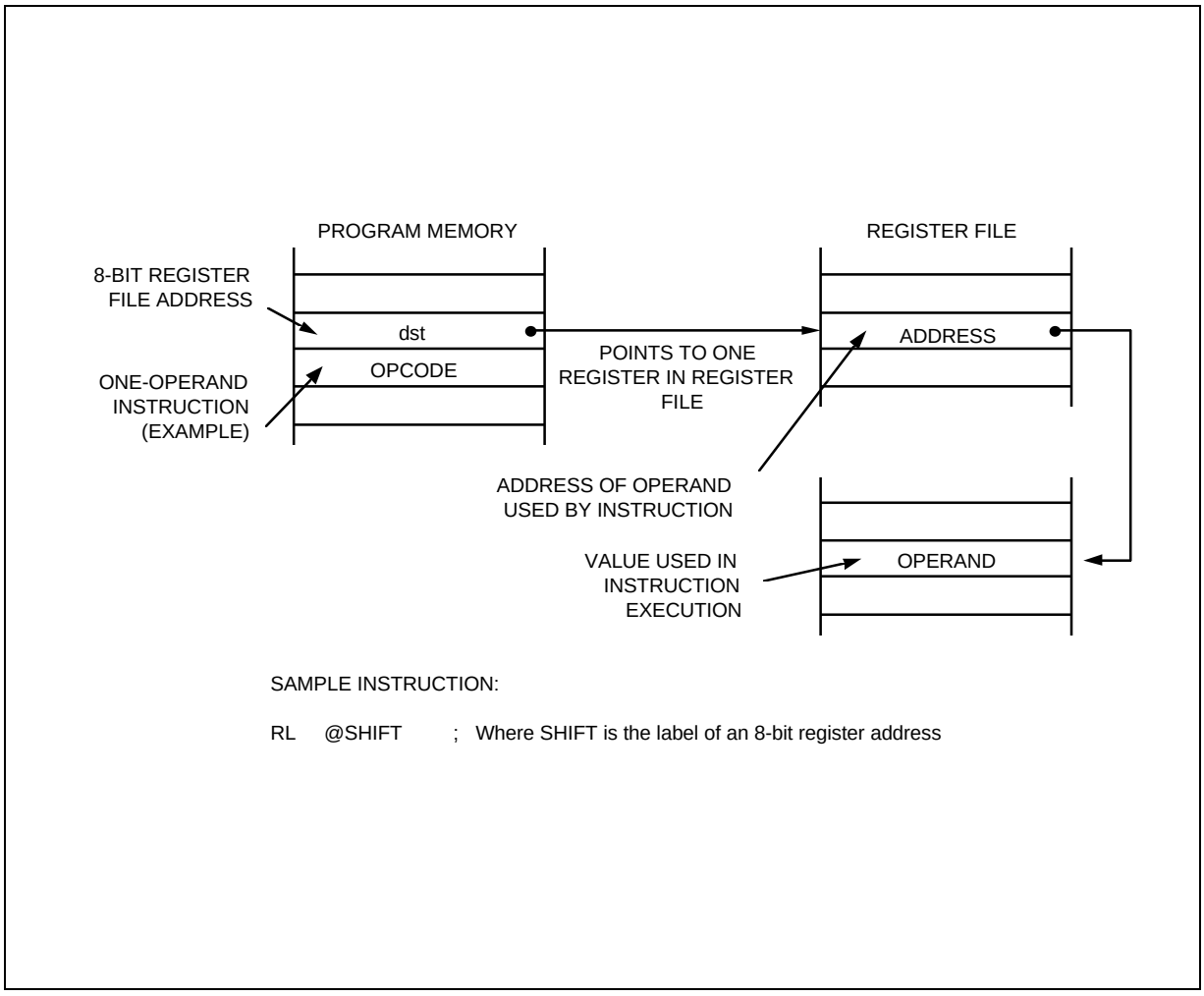


Figure 3-3. Indirect Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Continued)

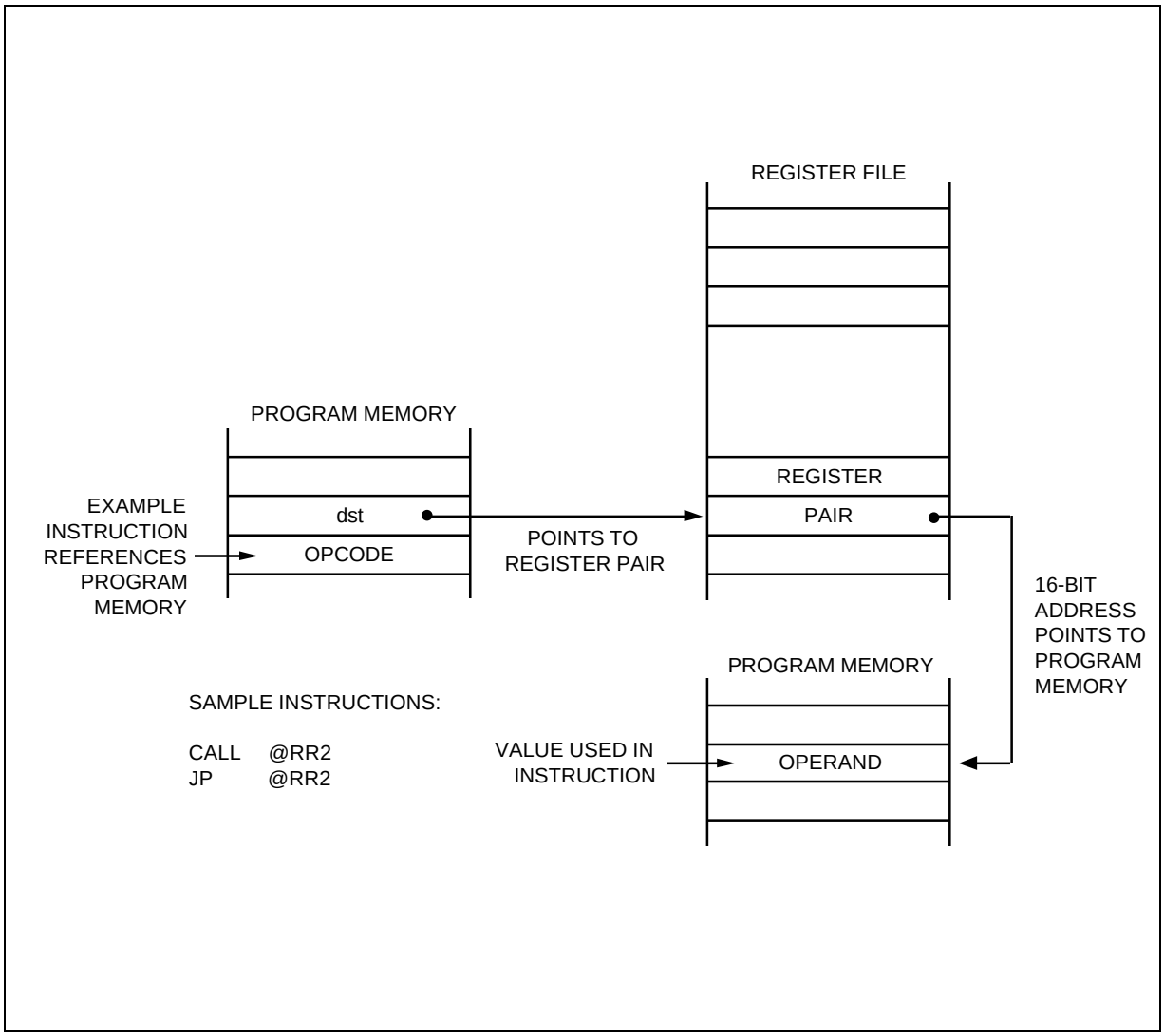


Figure 3-4. Indirect Register Addressing to Program Memory

INDIRECT REGISTER ADDRESSING MODE (Continued)

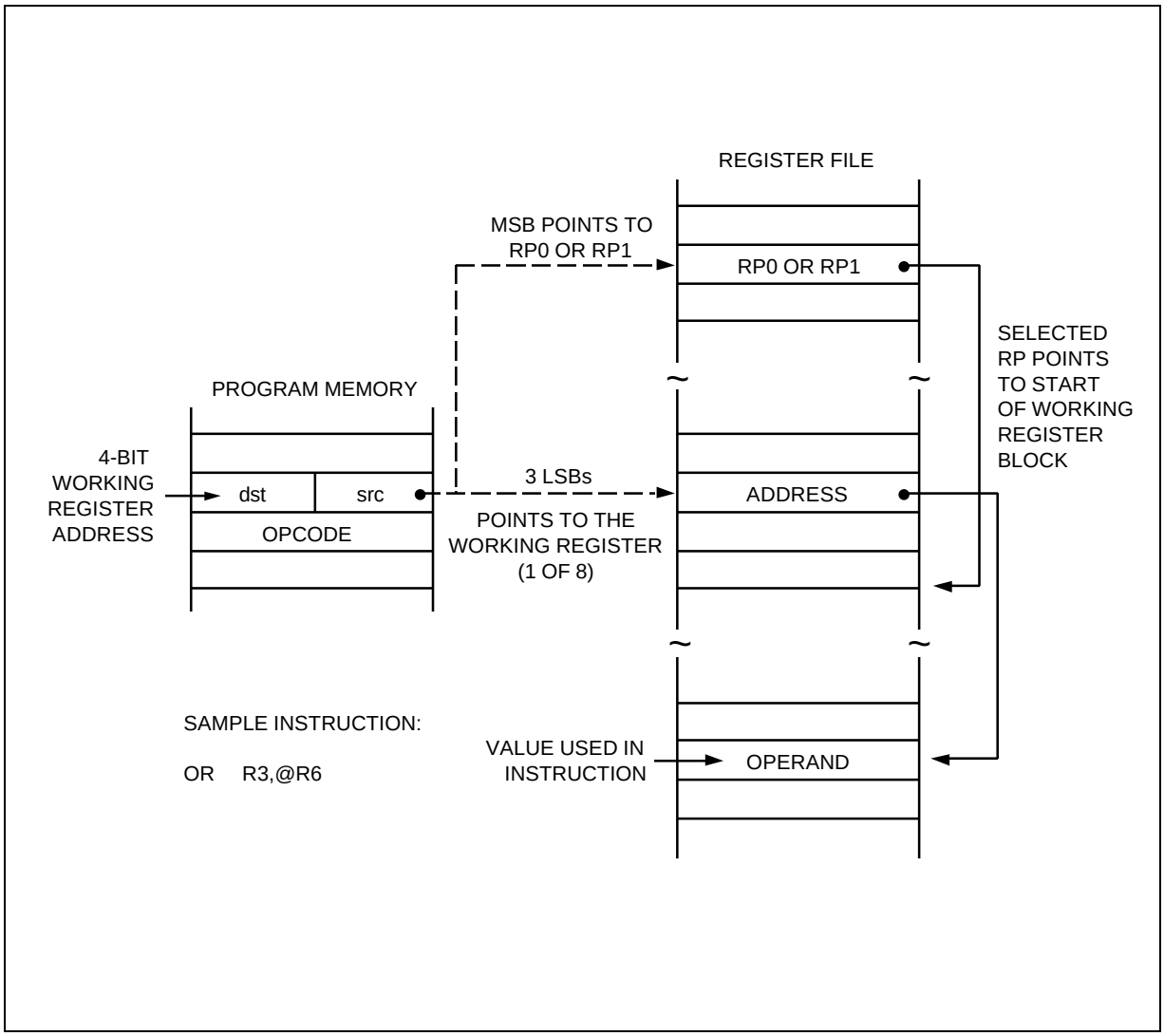


Figure 3-5. Indirect Working Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Concluded)

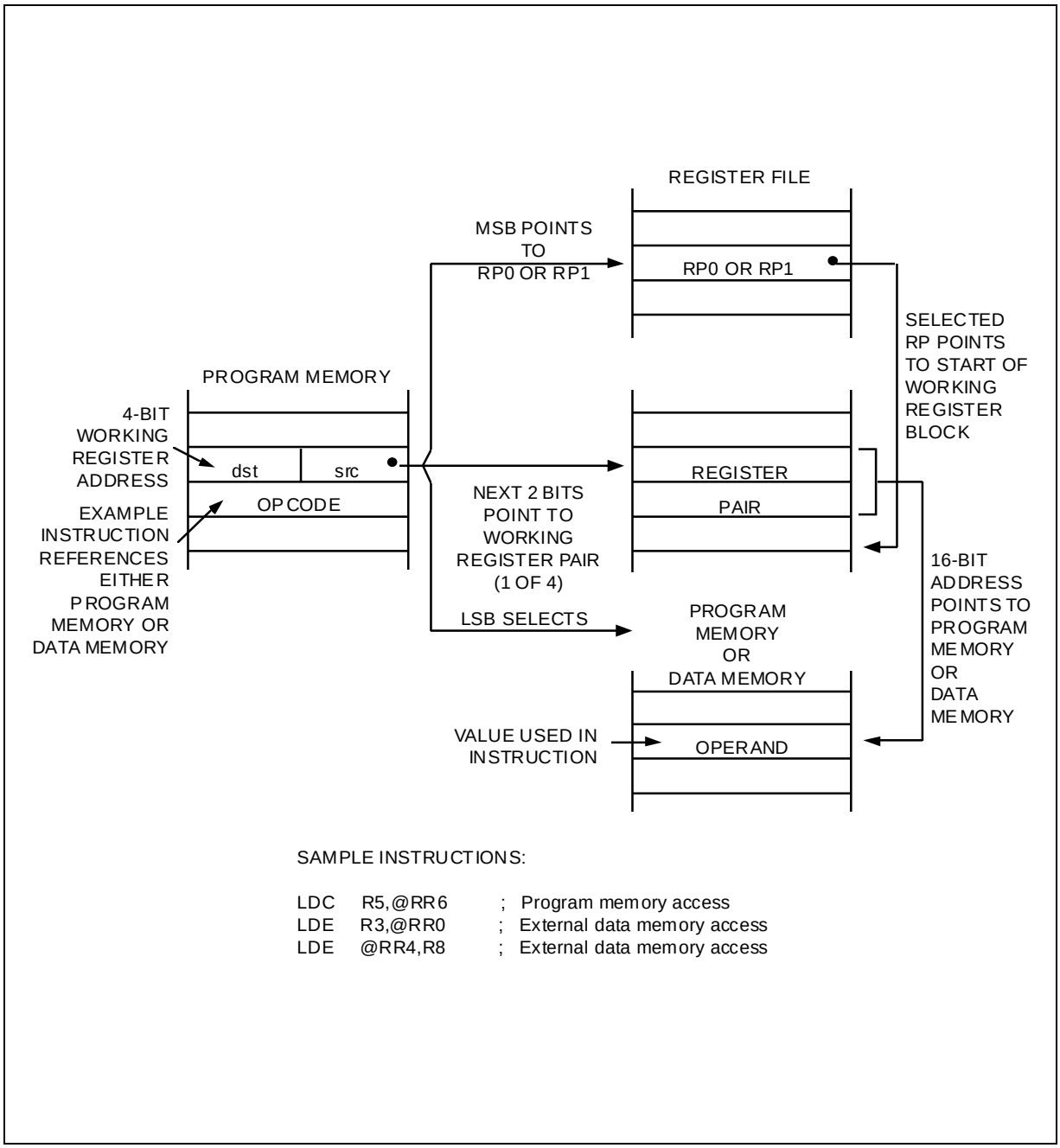


Figure 3-6. Indirect Working Register Addressing to Program or Data Memory

INDEXED ADDRESSING MODE (X)

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

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

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

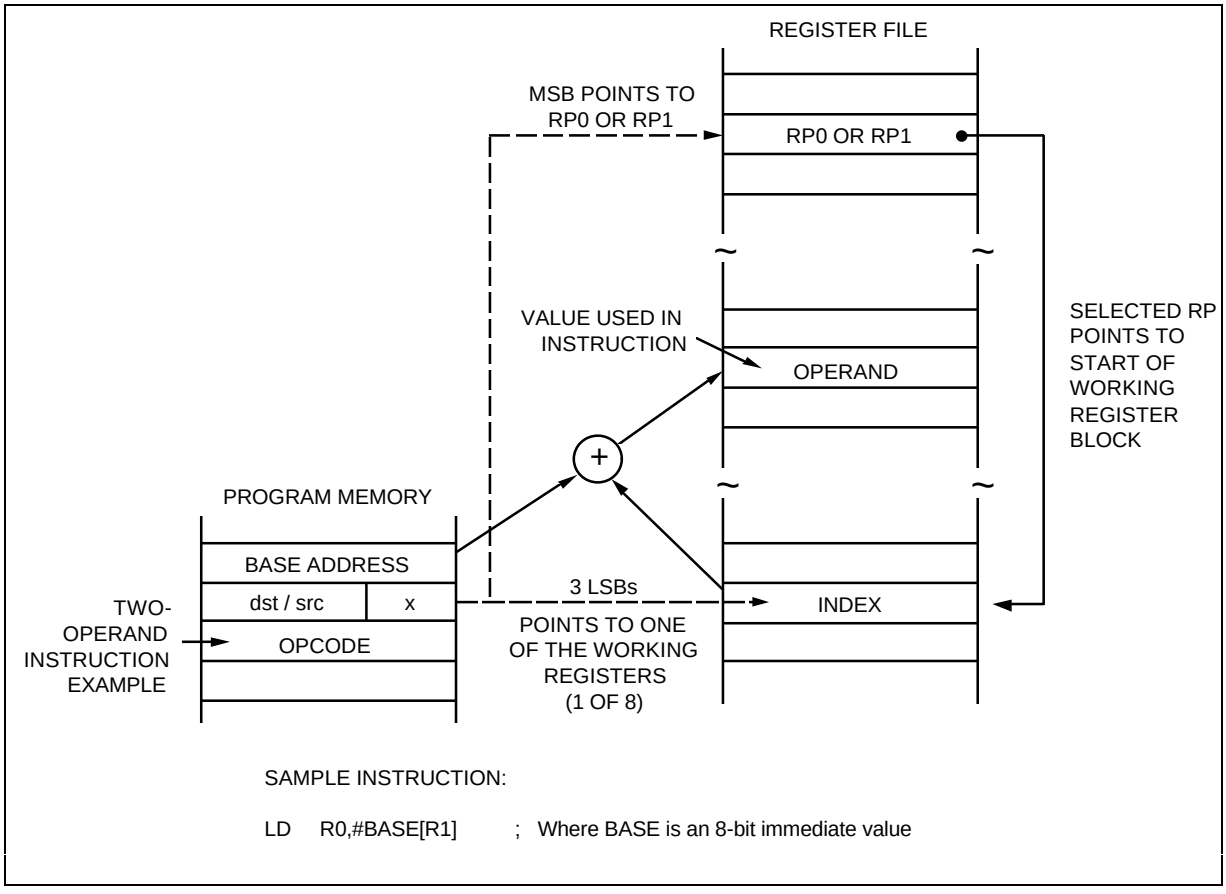
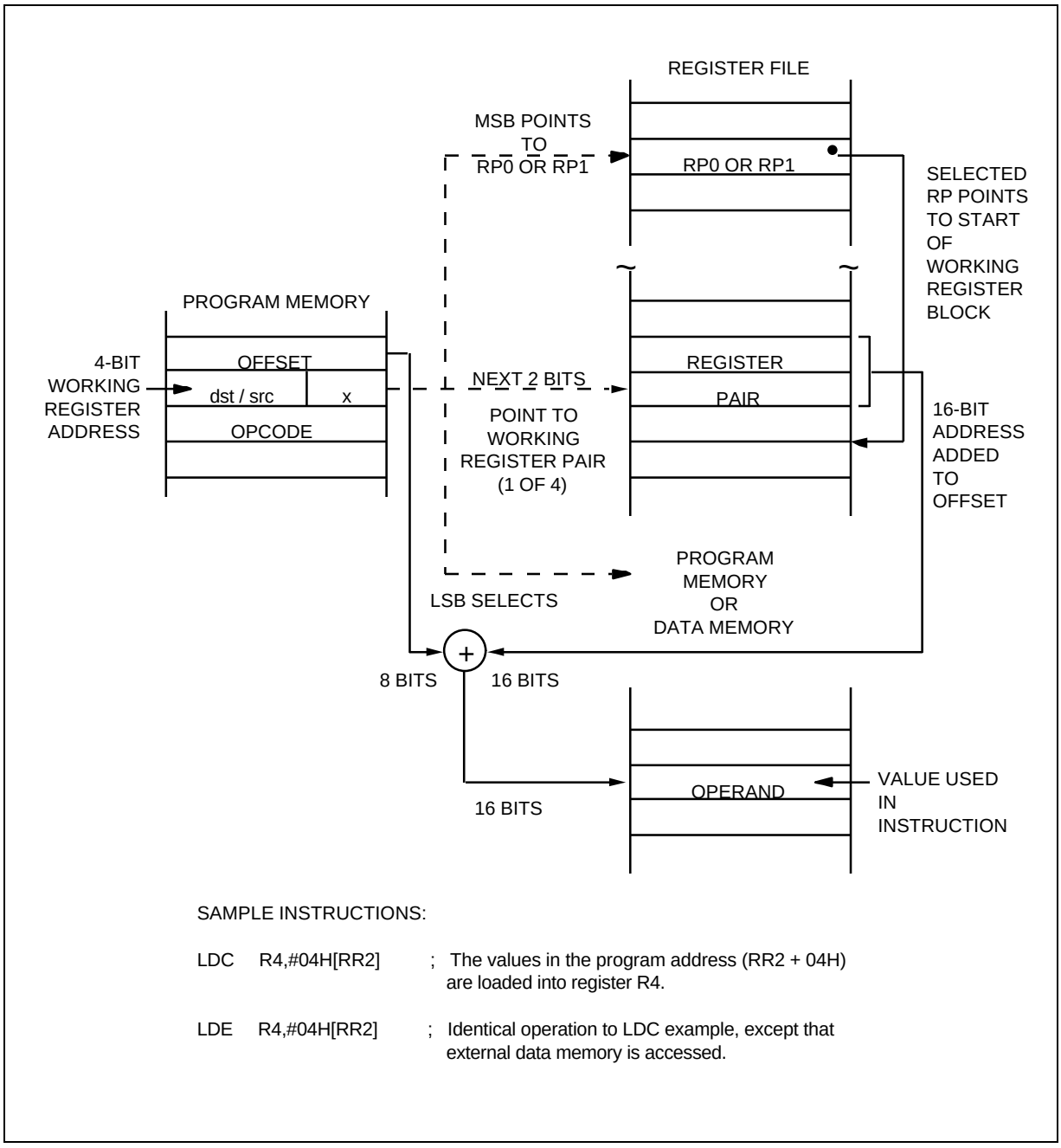


Figure 3-7. Indexed Addressing to Register File

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 data memory is accessed.
```

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

INDEXED ADDRESSING MODE (Concluded)

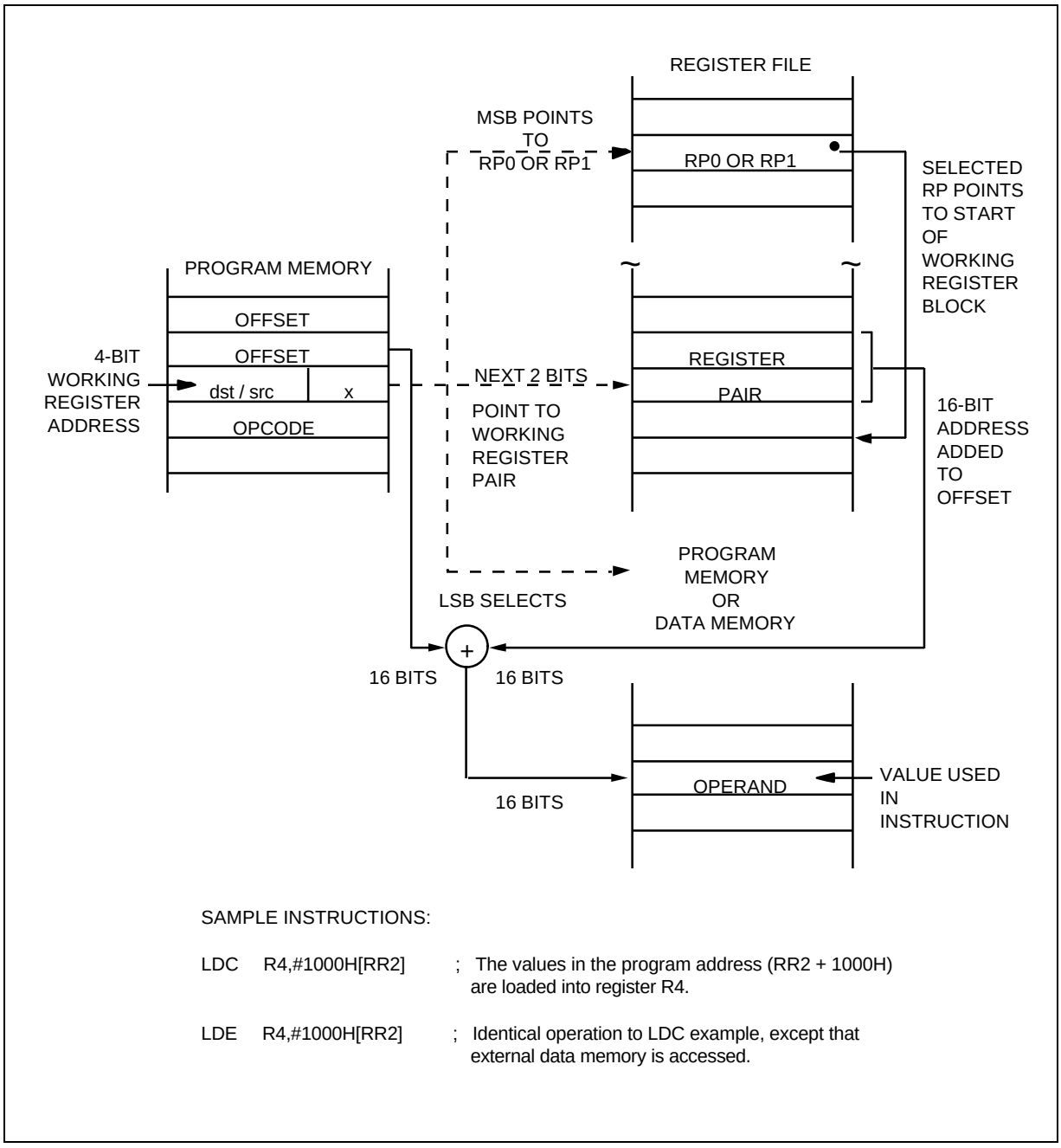


Figure 3-9. Indexed Addressing to Program or Data Memory

DIRECT ADDRESS MODE (DA)

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

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

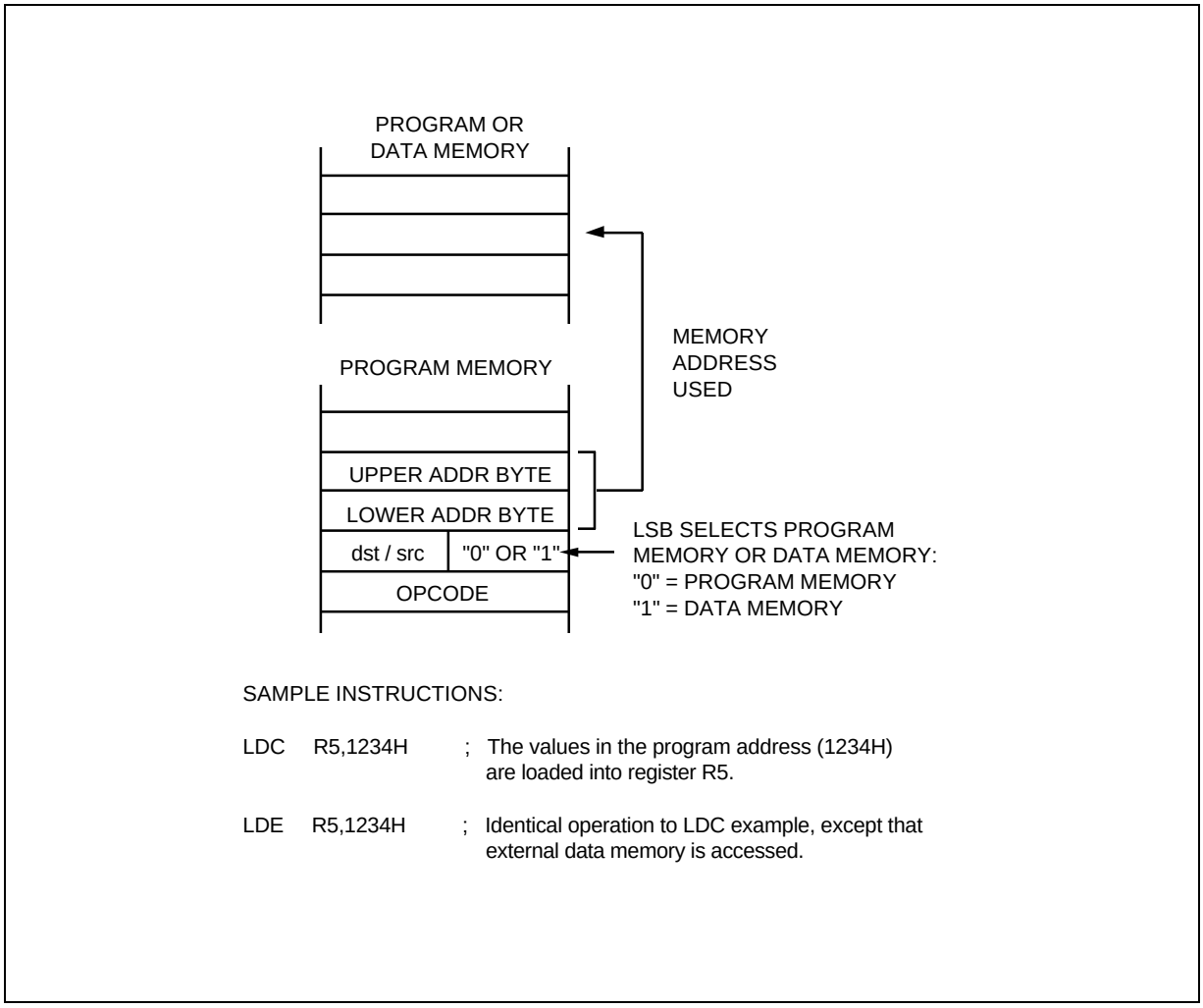
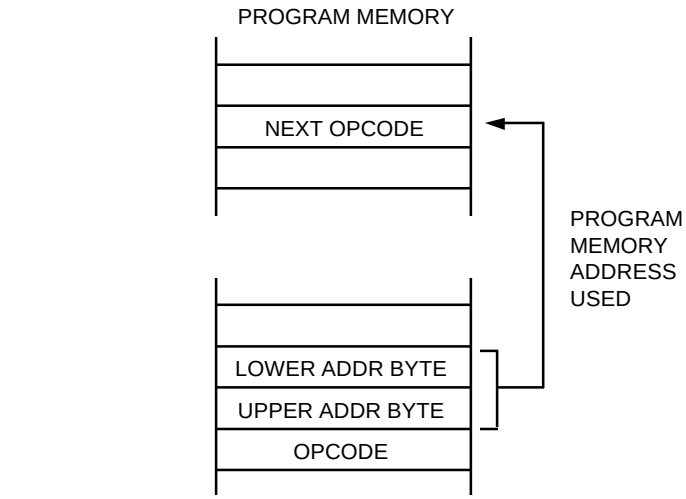


Figure 3-10. Direct Addressing for Load Instructions

DIRECT ADDRESS MODE (Continued)



SAMPLE INSTRUCTIONS:

```
JP      C,JOB1      ;  Where JOB1 is a 16-bit immediate address

CALL    DISPLAY     ;  Where DISPLAY is a 16-bit immediate address
```

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.

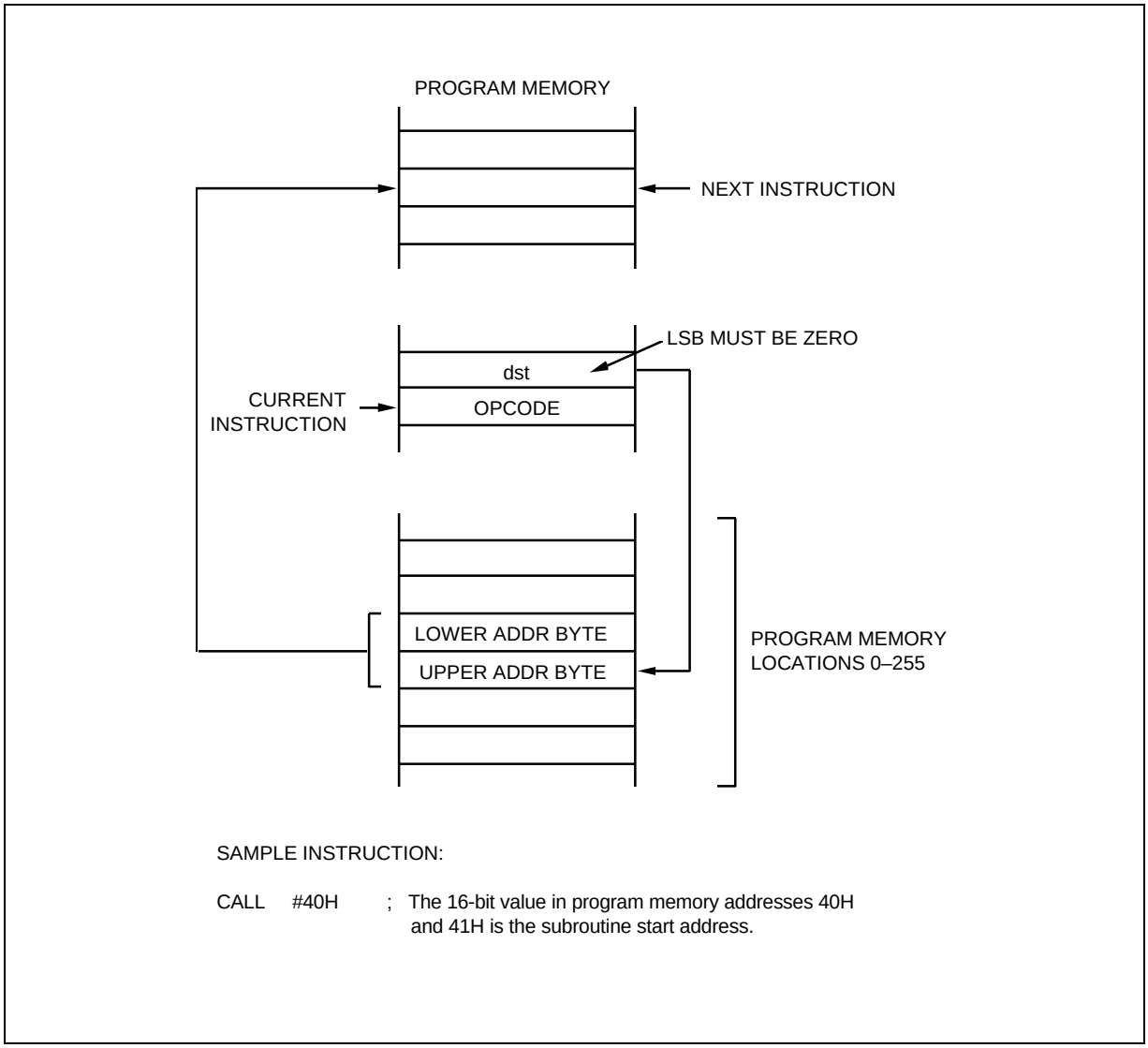


Figure 3-12. Indirect Addressing

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.

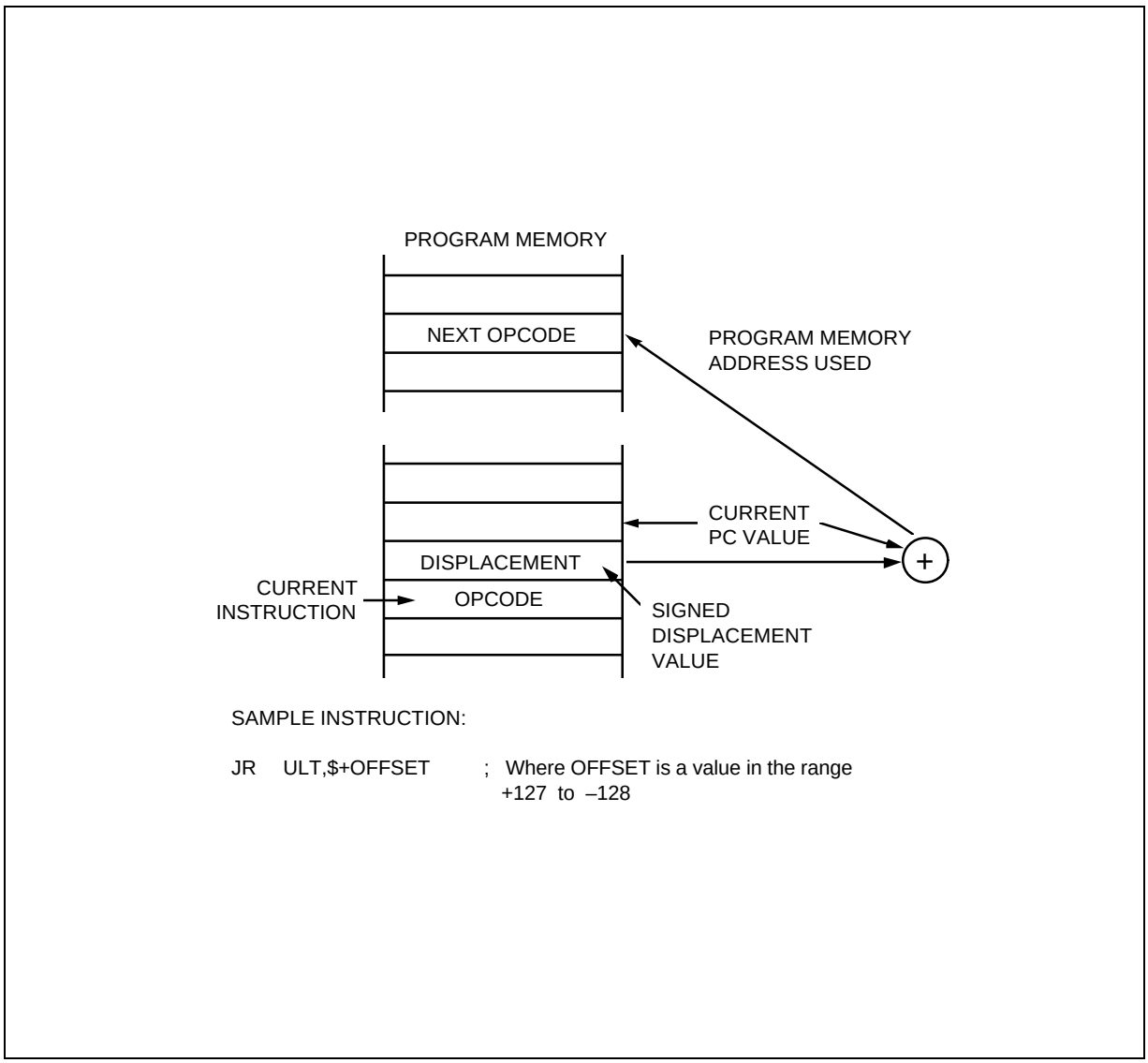


Figure 3-13. Relative Addressing

IMMEDIATE MODE (IM)

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

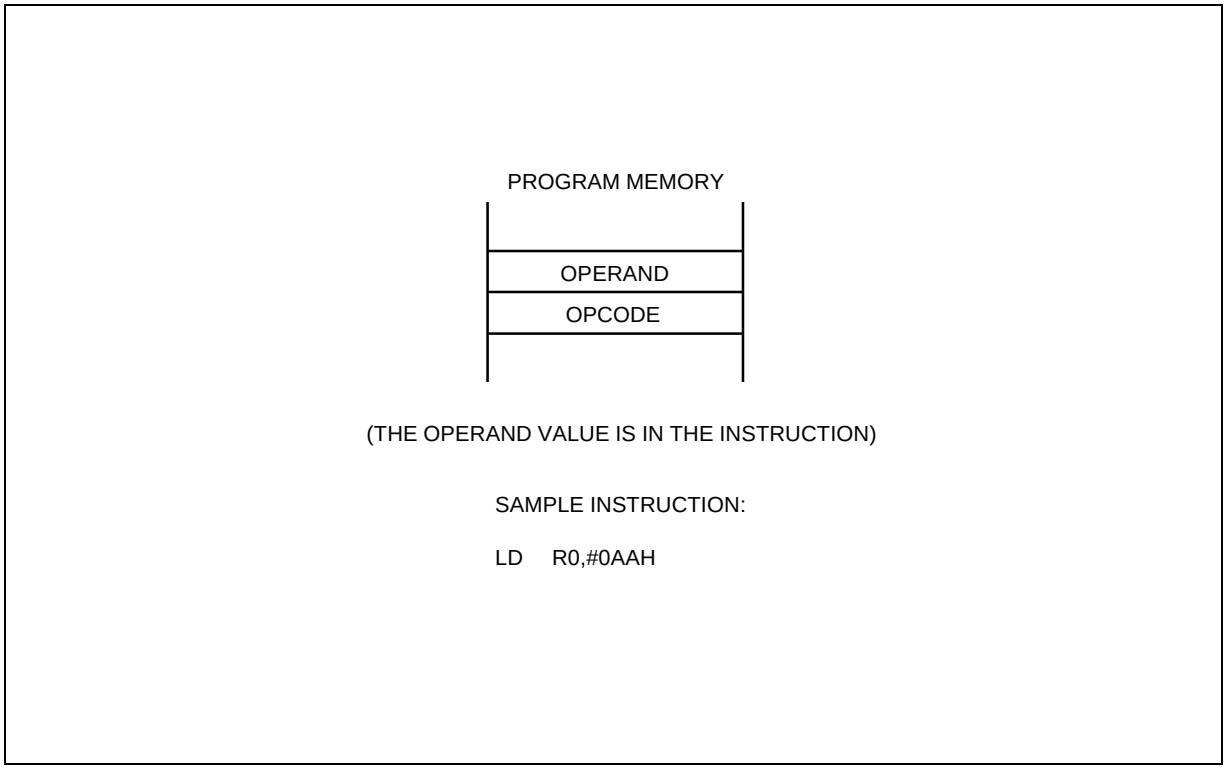


Figure 3-14. Immediate Addressing

4 CONTROL REGISTERS

OVERVIEW

In this section, detailed descriptions of the KS88C4504/P4504 control registers are presented in an easy-to-read format.

These descriptions will help familiarize you with the mapped locations in the register file. You can also use them as a quick-reference source when writing application programs.

System and peripheral registers are summarized in Tables 4-1, 4-2, and 4-3. Figure 4-1 illustrates the important features of the standard register description format.

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

Table 4-1. Set 1 Registers

Register Name	Mnemonic	Decimal	Hex	R/W
Timer A control register	TACON	208	D0H	R/W
Timer B control register	TBCON	209	D1H	R/W
Timer A data register	TADATA	210	D2H	R/W
Basic timer control register	BTCON	211	D3H	R/W
Clock Control register	CLKCON	212	D4H	R/W
System flags register	FLAGS	213	D5H	R/W
Register pointer 0	RP0	214	D6H	R/W
Register pointer 1	RP1	215	D7H	R/W
Stack pointer (high byte)	SPH	216	D8H	R/W
Stack pointer (low byte)	SPL	217	D9H	R/W
Instruction pointer (high byte)	IPH	218	DAH	R/W
Instruction pointer (low byte)	IPL	219	DBH	R/W
Interrupt request register	IRQ	220	DCH	R
Interrupt mask register	IMR	221	DDH	R/W
System mode register	SYM	222	DEH	R/W
Register page pointer	PP	223	DFH	R/W

Table 4-2. Set 1, Bank 0 Registers

Register Name	Mnemonic	Decimal	Hex	R/W
Port 0 data register	P0	224	E0H	R/W
Port 1 data register	P1	225	E1H	R/W
Port 2 data register	P2	226	E2H	R/W
Port 3 data register	P3	227	E3H	R/W
Port 4 data register	P4	228	E4H	R/W
Port 5 data register	P5	229	E5H	R/W
Timer B data register	TBDATA	230	E6H	R/W
Port 2 interrupt control register	P2INT	231	E7H	R/W
Timer C control register	TCCON	232	E8H	R/W
Timer D control register	TDCON	233	E9H	R/W
Timer C data register (High byte)	TCDATAH	234	EAH	R/W
Timer C data register (Low byte)	TCDATAL	235	EBH	R/W
Timer D data register (High byte)	TDDATAH	236	ECH	R/W
Timer D data register (Low byte)	TDDATAL	237	EDH	R/W
Timer C pending register	TCPND	238	EEH	R/W
Timer D pending register	TDPND	239	EFH	R/W

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

Register Name	Mnemonic	Decimal	Hex	R/W
Port 0 control register (High byte)	P0CONH	240	F0H	R/W
Port 0 control register (Low byte)	P0CONL	241	F1H	R/W
Port 1 control register (High byte)	P1CONH	242	F2H	R/W
Port 1 control register (Low byte)	P1CONL	243	F3H	R/W
Port 2 control register (High byte)	P2CONH	244	F4H	R/W
Port 2 control register (Low byte)	P2CONL	245	F5H	R/W
Port 3 control register (High byte)	P3CONH	246	F6H	R/W
Port 3 control register (Low byte)	P3CONL	247	F7H	R/W
Port 4 control register (High byte)	P4CONH	248	F8H	R/W
Port 4 control register (Low byte)	P4CONL	249	F9H	R/W
Port 5 control register	P5CON	250	FAH	R/W
Port 2 interrupt pending register	P2PND	251	FBH	R/W
Test mode register	TESTMOD	252	FCH	R/W
Basic timer counter register	BTCNT	253	FDH	R
External memory timing register	EMT	254	FEH	R/W
Interrupt priority register	IPR	255	FFH	R/W

Table 4-3. Set 1, Bank 1 Registers

Register Name	Mnemonic	Decimal	Hex	R/W
Locations E0H-E9H in set 1, bank 1, are not mapped.				
Timer A counter register	TACNT	234H	EAH	R
Timer B counter register	TBCNT	235H	EBH	R
Timer C counter register(High byte)	TCCNTH	236H	ECH	R
Timer C counter register(Low byte)	TCCNTL	237H	EDH	R
Timer D counter register(High byte)	TDCNTH	238H	EEH	R
Timer D counter register(Low byte)	TDCNTL	239H	EFH	R
Locations F0H in set 1, bank 1, are not mapped.				
Synchronous SIO control register	SIOCON	241	F1H	R/W
Synchronous SIO data register	SIODATA	242	F2H	R/W
Synchronous SIO clock prescaler register	SIOPS	243	F3H	R/W
A/D converter input register	ADDATA	244	F4H	R
External Bus control register	EXTBUS	245	F5H	R/W
Locations F6H in set 1, bank1, are not mapped.				
A/D converter control register	ADCON	247	F7H	R/W
Locations F8H in set 1, bank 1, are not mapped.				
PWM module control register	PWMCON	249	F9H	R/W
PWM1 data register	PWM1	250	FAH	R/W
PWM0 data register	PWM0	251	FBH	R/W
Chip selection address data register	CSDATA0	252	FCH	R/W
Chip selection address data register	CSDATA1	253	FDH	R/W
Chip selection address data register	CSDATA2	254	FEH	R/W
Chip selection address data register	CSDATA3	255	FFH	R/W

NOTE: The A/D converter end-of-conversion bit, ADCON.3, is read-only.

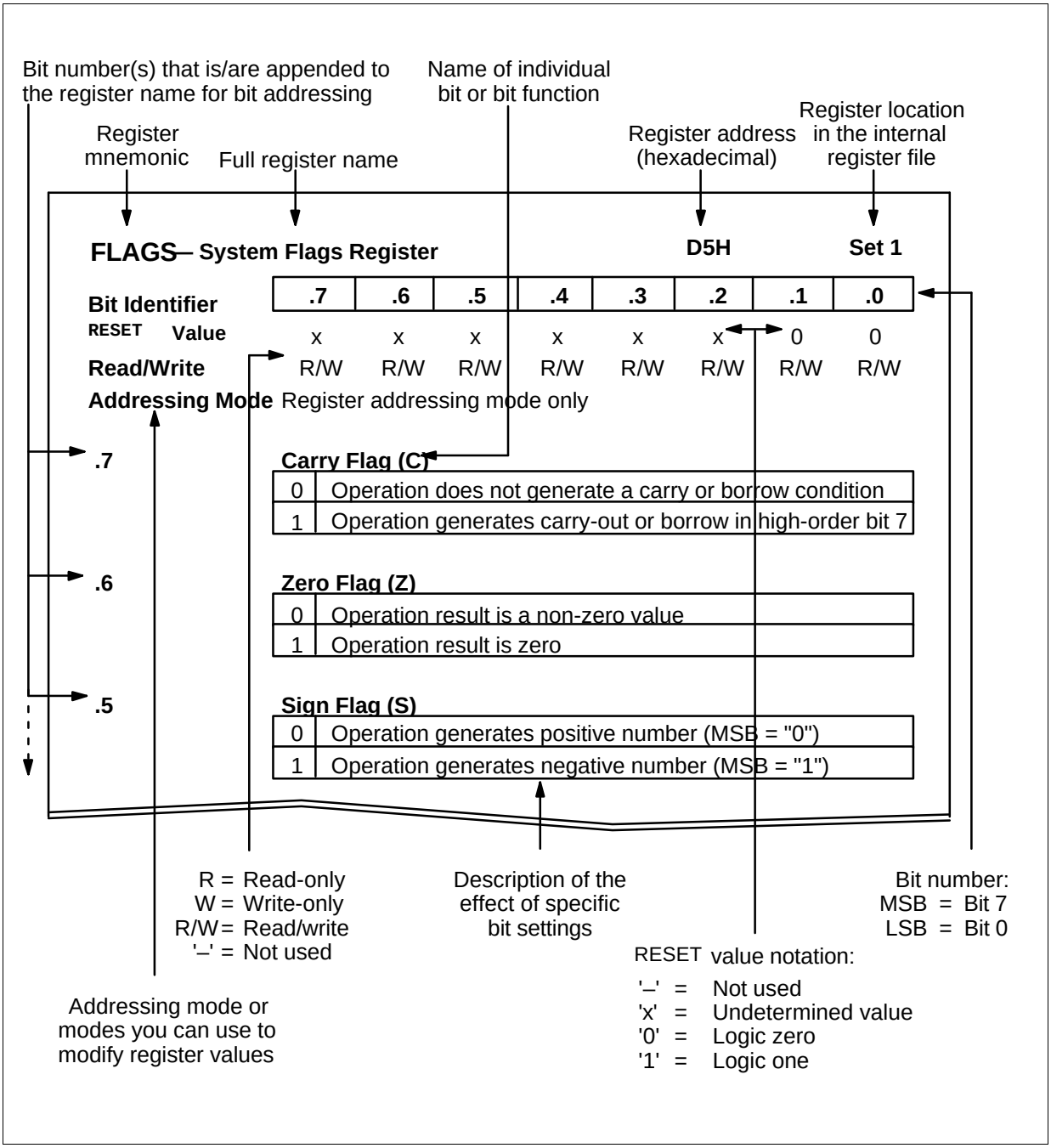


Figure 4-1. Register Description Format

ADCON — A/D Converter Control Register F7H Set 1, Bank 1

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

.7 A/D Converter Test Mode Control Bit

This bit is used for factory testing only. During normal operation, ADCON.7 should always remain cleared to "0".

.6 – .4 A/D Converter Analog Input Pin Selection Bits

0	0	0	ADC0 (P7.0)
0	0	1	ADC1 (P7.1)
0	1	0	ADC2 (P7.2)
0	1	1	ADC3 (P7.3)
1	0	0	Not used for KS88C4504/P4504
• • •			"
1	1	1	Not used for KS88C4504/P4504

.3 End-of-Conversion Bit (Read-only) (note)

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

.2 – .1 Clock Source Selection

0	0	fosc/16
0	1	fosc/8
1	0	fosc/4
1	1	fosc/1

.0 Start or Enable Bit

0	Disable operation
1	Start operation

NOTE: This bit is read-only. You can poll ADCON.3 to determine internally when an A/D conversion operation has been completed. A reset operation sets ADCON.3 to "1".

BTCON — Basic Timer Control Register D3H Set 1

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

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

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

.3 and .2 Basic Timer Input Clock Selection Bits

0	0	$f_{OSC}/4096$
0	1	$f_{OSC}/1024$
1	0	$f_{OSC}/128$
1	1	Invalid setting; not used for KS88C4504/P4504

.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 (2)

0	No effect
1	Clear basic timer

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".

CLKCON — System Clock Control Register D4H Set 1

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

.7 Oscillator IRQ Wake-up Function Enable Bit

0	Enable IRQ for main system oscillator wake-up in power-down mode
1	Disable IRQ for main system oscillator wake-up in power-down mode

.6 and .5 Main Oscillator Stop Control Bits

0	0	No effect
0	1	No effect
1	0	Stop main oscillator (2)
1	1	No effect

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

0	0	Divide by 16 (f_{OSC}/16)
0	1	Divide by 8 (f_{OSC}/8)
1	0	Divide by 2 (f_{OSC}/2)
1	1	Non-divided clock (f_{OSC})

.2 – .0 Subsystem Clock Selection Bits (3)

1	0	1	Invalid setting for KS88C4504/P4504
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. This selection is not recommended.
3. These selection bits are required only for systems that have a main clock and a subsystem clock. KS88C4504/P4504 use only the main oscillator clock circuit. For this reason, the setting '101B' is invalid.

EMT — External Memory Timing Register FEH Set 1, Bank 0

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

.7 **External WAIT Input Function Enable Bit** (Note)

0	Disable WAIT input function for external device (normal operating mode)
1	Enable WAIT input function for external device

.6–.2 Not used for KS88C4504/P4504

.1 **Stack Area Selection Bit**

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

.0 Not used for KS88C4504/P4504

NOTE: Before you enable the external interface WAIT input function, you must first configure P5.0 as the WAIT signal input pin. To do this, bits 0, 1 in the P5CON register must be set to appropriate values.

FLAGS — System Flags Register D5H Set 1

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

.7 Carry Flag (C)

0	Operation does not generate a carry or borrow condition
1	Operation generates a carry-out or borrow into high-order bit 7

.6 Zero Flag (Z)

0	Operation result is a non-zero value
1	Operation result is zero

.5 Sign Flag (S)

0	Operation generates a positive number (MSB = "0")
1	Operation generates a negative number (MSB = "1")

.4 Overflow Flag (V)

0	Operation result is $\leq$ +127 or $\geq$ –128
1	Operation result is > +127 or < –128

.3 Decimal Adjust Flag (D)

0	Add operation completed
1	Subtraction operation completed

.2 Half-Carry Flag (H)

0	No carry-out of bit 3 or no borrow into bit 3 by addition or subtraction
1	Addition generated carry-out of bit 3 or subtraction generated borrow into bit 3

.1 Fast Interrupt Status Flag (FIS)

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

.0 Bank Address Selection Flag (BA)

0	Bank 0 is selected
1	Bank 1 is selected

IMR — Interrupt Mask Register DDH Set 1

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

.7 Interrupt Level 7 (IRQ7) Enable Bit; PWM Overflow

0	Disable IRQ7 interrupts
1	Enable IRQ7 interrupts

.6 Interrupt Level 6 (IRQ6) Enable Bit; INT2–INT7

0	Disable IRQ6 interrupts
1	Enable IRQ6 interrupts

.5 Interrupt Level 5 (IRQ5) Enable Bit; INT0–INT1

0	Disable IRQ5 interrupts
1	Enable IRQ5 interrupts

.4 Interrupt Level 4 (IRQ4) Enable Bit; Timer B Match

0	Disable IRQ4 interrupts
1	Enable IRQ4 interrupts

.3 Interrupt Level 3 (IRQ3) Enable Bit; Timer A Match

0	Disable IRQ3 interrupts
1	Enable IRQ3 interrupts

.2 Interrupt Level 2 (IRQ2) Enable Bit; SIO Transmit/Receive

0	Disable IRQ2 interrupts
1	Enable IRQ2 interrupts

.1 Interrupt Level 1 (IRQ1) Enable Bit; Timer D Overflow/Capture & Match

0	Disable IRQ1 interrupts
1	Enable IRQ1 interrupts

.0 Interrupt Level 0 (IRQ0) Enable Bit; Timer C Overflow/Capture & Match

0	Disable IRQ0 interrupts
1	Enable IRQ0 interrupts

NOTE: When this register is manipulated, all interrupts must be disabled.

IPH — Instruction Pointer (High Byte) DAH Set 1

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

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

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

IPL — Instruction Pointer (Low Byte) DBH Set 1

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

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

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

IPR — Interrupt Priority Register FFH Set 1

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

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

0	0	0	Group priority undefined
0	0	1	B > C > A
0	1	0	A > B > C
0	1	1	B > A > C
1	0	0	C > A > B
1	0	1	C > B > A
1	1	0	A > C > B
1	1	1	Group priority undefined

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

0	IRQ6 > IRQ7
1	IRQ7 > IRQ6

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

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

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

0	IRQ3 > IRQ4
1	IRQ4 > IRQ3

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

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

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

0	IRQ0 > IRQ1
1	IRQ1 > IRQ0

NOTE: Interrupt group A is IRQ0 and IRQ1; interrupt group is IRQ2, IRQ3, and IRQ4; interrupt group C is IRQ5, IRQ6, and IRQ7.

IRQ — Interrupt Request Register DCH Set 1

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

.7 Interrupt Level 7 (IRQ7) Request Pending Bit; PWM

0	No IRQ7 interrupt pending
1	IRQ7 interrupt is pending

.6 Interrupt Level 6 (IRQ6) Request Pending Bit; INT2– INT7

0	No IRQ6 interrupt pending
1	IRQ6 interrupt is pending

.5 Interrupt Level 5 (IRQ5) Request Pending Bit; INT0–INT1

0	No IRQ5 interrupt pending
1	IRQ5 interrupt is pending

.4 Interrupt Level 4 (IRQ4) Request Pending Bit; Timer B Match

0	No IRQ4 interrupt pending
1	IRQ4 interrupt is pending

.3 Interrupt Level 3 (IRQ3) Request Pending Bit; Timer A Match

0	No IRQ3 interrupt pending
1	IRQ3 interrupt is pending

.2 Interrupt Level 2 (IRQ2) Request Pending Bit; SIO Transmit/Receive

0	No IRQ2 interrupt pending
1	IRQ2 interrupt is pending

.1 Interrupt Level 1 (IRQ1) Request Pending Bit; Timer D Overflow/Capture & Match

0	No IRQ1 interrupt pending
1	IRQ1 interrupt is pending

.0 Interrupt Level 0 (IRQ0) Request Pending Bit; Timer C Overflow/Capture & Match

0	No IRQ0 interrupt pending
1	IRQ0 interrupt is pending

P0CONH — Port 0 Control Register F0H Set 1, Bank 0

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

.7 – .6 **Port 0.7 (ADC3)**

0	0	Normal input mode
0	1	A/DC input mode
1	0	Normal Input, pull-down
1	1	Push-pull output mode

.5 – .4 **Port 0.6 (ADC2)**

0	0	Normal input mode
0	1	A/DC input mode
1	0	Normal input, pull-down
1	1	Push-pull output mode

.3 – .2 **Port 0.5 (ADC1)**

0	0	Normal input mode
0	1	A/DC input mode
1	0	Normal input, pull-down
1	1	Push-pull output mode

.1 – .0 **Port 0.4 (ADC0)**

0	0	Normal input mode
0	1	A/DC input mode
1	0	Normal input, pull-down
1	1	Push-pull output mode

P0CONL — Port 0 Control Register F1H Set 1, Bank 0

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

.7 – .6 **Port 0.3**

0	x	normal input mode
1	0	Normal Input, Pull-up mode
1	1	Push-pull output mode

.5 – .4 **Port 0.2**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.3 – .2 **Port 0.1**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.1 – .0 **Port 0.0**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

P1CONH — Port 1 Control Register F2H Set 1, Bank 0

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

.7 – .6 **Port 1.7 (SCK)**

0	0	Normal input mode
0	1	Alternative function select
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.5 – .4 **Port 1.6 (SO)**

0	0	Normal input mode
0	1	Alternative function select; SO enable
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.3 – .2 **Port 1.5 (SI)**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.1 – .0 **Port 1.4**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

P1CONL — Port 1 Control Register F3H Set 1, Bank 0

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

.7 – .6 **Port 1.3**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.5 – .4 **Port 1.2**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.3 – .2 **Port 1.1**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.1 – .0 **Port 1.0**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

P2CONH — Port 2 Control Register F4H Set 1, Bank 0

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

.7 – .6 **Port 2.7 (INT7)**

0	0	Normal input, falling-edge interrupt
0	1	Normal input, rising-edge interrupt
1	0	Normal input, falling-edge interrupt, pull-up mode
1	1	Push-pull output mode

.5 – .4 **Port 2.6 (INT6)**

0	0	Normal input, falling-edge interrupt
0	1	Normal input, rising-edge interrupt
1	0	Normal input, falling-edge interrupt, pull-up mode
1	1	Push-pull output mode

.3 – .2 **Port 2.5 (INT5)**

0	0	Normal input, falling-edge interrupt
0	1	Normal input, rising-edge interrupt
1	0	Normal input, falling-edge interrupt, pull-up mode
1	1	Push-pull output mode

.1 – .0 **Port 2.4 (INT4)**

0	0	Normal input, falling-edge interrupt
0	1	Normal input, rising-edge interrupt
1	0	Normal input, falling-edge interrupt, pull-up mode
1	1	Push-pull output mode

P2CONL — Port 2 Control Register F5H Set 1, Bank 0

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

.7 – .6 **Port 2.3 (INT3)**

0	0	Normal input, falling-edge interrupt
0	1	Normal input, rising-edge interrupt
1	0	Normal input, falling-edge interrupt, pull-up mode
1	1	Push-pull output mode

.5 – .4 **Port 2.2 (INT2)**

0	0	Normal input, falling-edge interrupt
0	1	Normal input, rising-edge interrupt
1	0	Normal input, falling-edge interrupt, pull-up mode
1	1	Push-pull output mode

.3 – .2 **Port 2.1 (INT1)**

0	x	Normal input, low-level interrupt mode
1	0	Normal input, low-level interrupt, pull-up mode
1	1	Push-pull output mode

.1 – .0 **Port 2.0 (INT0)**

0	x	Normal input, low-level interrupt mode
1	0	Normal input, low-level interrupt, pull-up mode
1	1	Push-pull output mode

P2INT — Port 2 Interrupt Enable Register E7H Set 1, Bank 0

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

.7 INT7 Interrupt Enable Bit

0	Disable INT7
1	Enable INT7

.6 INT6 Interrupt Enable Bit

0	Disable INT6
1	Enable INT6

.5 INT5 Interrupt Enable Bit

0	Disable INT5
1	Enable INT5

.4 INT4 Interrupt Enable Bit

0	Disable INT4
1	Enable INT4

.3 INT3 Interrupt Enable Bit

0	Disable INT3
1	Enable INT3

.2 INT2 Interrupt Enable Bit

0	Disable INT2
1	Enable INT2

.1 INT1 Interrupt Enable Bit

0	Disable INT1
1	Enable INT1

.0 INT7 Interrupt Enable Bit

0	Disable INT4
1	Enable INT4

NOTE: The IRQ5 interrupts at P2.0 and P2.1 have the same priority level and vector address in the interrupts structure. In case of connection, the P2.0 interrupt is serviced first.

P2PND — Port 2 Pending Register FBH Set 1, Bank 0

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

.7 INT7 Interrupt Enable Bit

0	INT7 is not pending
1	INT7 is pending

.6 INT6 Interrupt Enable Bit

0	INT6 is not pending
1	INT6 is pending

.5 INT5 Interrupt Enable Bit

0	INT5 is not pending
1	INT5 is pending

.4 INT4 Interrupt Enable Bit

0	INT4 is not pending
1	INT4 is pending

.3 INT3 Interrupt Enable Bit

0	INT3 is not pending
1	INT3 is pending

.2 INT2 Interrupt Enable Bit

0	INT2 is not pending
1	INT2 is pending

.1 INT1 Interrupt Enable Bit

0	INT1 is not pending
1	INT1 is pending

.0 INT0 Interrupt Enable Bit

0	INT0 is not pending
1	INT0 is pending

NOTES:
1. P2PND bits can be polled by application software to detect interrupt pending conditions.
2. To clear an interrupt pending condition, write a "0" to the P2PND register bit location; writing a "1" has no effect.
3. To avoid errors, we recommend using load instructions (except for LDB) to manipulate the P2PND register.

P3CONH — Port 3 Control Register F6H Set 1, Bank 0

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

.7 and .6 **P3.7/PWM1**

0	0	Normal input mode
0	1	Alternative function output enable (PWM1)
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.5 and .4 **P3.6/PWM0**

0	0	Normal input mode
0	1	Alternative function output enable (PWM0)
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.3 and .2 **P3.5/TDPWM/TDOUT**

0	0	Normal input mode
0	1	Alternative function output enable (TDPWM, TDOUT)
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.1 and .0 **P3.4/TCPWM/TCOUT**

0	0	Normal input mode
0	1	Alternative function output enable (TCPWM, TCOUT)
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

NOTE: To enable the alternative output function, set the appropriate bits in TCCON, TDCON, PWMCON.

P3CONL — Port 3 Control Register (Low Byte) F7H Set 1, Bank 0

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

.7 and .6 **P3.3/TCCAP**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.5 and .4 **P3.2/TDCAP**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.3 and .2 **P3.1/TCCK**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.1 and .0 **P3.0/TDCK**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

NOTE: To enable the alternative input function, configure any one of the two input mode selections.

P4CONH — Port 4 Control Register (High Byte) F8H Set 1, Bank 0

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

.7 and .6 **P4.7 / CS3**

0	0	Normal input mode
0	1	Embedded chip selection output enable
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.5 and .4 **P4.6 / CS2**

0	0	Normal input mode
0	1	Embedded chip selection output enable
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.3 and .2 **P4.5 / CS1**

0	0	Normal input mode
0	1	Embedded chip selection output enable
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.1 and .0 **P4.4 / CS0**

0	0	Normal input mode
0	1	Embedded chip selection output enable
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

P4CONL — Port 4 Control Register (Low Byte) F9H Set 1, Bank 0

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

.7 and .6 **P4.3**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.5 and .4 **P4.2**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.3 and .2 **P4.1**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

.1 and .0 **P4.0**

0	x	Normal input mode
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

P5CON — Port 5 Control Register

FAH **Set 1, Bank 0**

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

.7 – .4

Not used for the KS88C4504/P4504

.3 – .2 **P5.0 Bit Pair Configuration Settings**

0	x	Normal input; WAIT signal input enable
1	0	Normal input, pull-up; WAIT signal input enable
1	1	Push-pull output mode

.1 – .0 **P5.1 Bit Pair Configuration Settings**

0	0	Normal input mode
0	1	Invalid setting
1	0	Normal input, pull-up mode
1	1	Push-pull output mode

NOTES:

1. 'x' means don't care.
2. To enable the WAIT input function, you must also make the appropriate settings in the EMT register.

PP — Register Page Pointer DFH Set 1

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

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

0	0	0	0	Destination: page 0
0	0	0	1	Destination: page 1
0	0	1	0	Destination: page 2
0	0	1	1	Destination: page 3
0	1	0	0	Not used
• • •				"
1	1	1	1	Not used

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

0	0	0	0	Source: page 0
0	0	0	1	Source: page 1
0	0	1	0	Source: page 2
0	0	1	1	Source: page 3
0	1	0	0	Not used
• • •				"
1	1	1	1	Not used

NOTE: You should refer to page 6-39 and use DJNZ instruction properly when DJNZ instruction is used in your program.

PWMCON — PWM Control Register F9H Set 1, Bank 1

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

.7–.4 4-Bit Prescaler Value for PWM Counter Input Clock

0	0	0	0	Non-divided
0	0	0	1	Divide input clock by 2
0	0	1	0	Divide input clock by 3
0	0	1	1	Divide input clock by 4
0	1	0	0	Divide input clock by 5
•	•	•	•	Divide input clock by 6–15
1	1	1	1	Divide input clock by 16

.3 PWM Counter Enable Bit

0	Stop PWM counter operation
1	Start (start counting)

.2 PWM Clock Selection Bit

0	CPU clock
1	PWMCK (P3.1)

.1 PWM Counter Interrupt Enable Bit

0	Disable PWM OVF interrupt
1	Enable PWM OVF interrupt

.0 PWM OVF Pending Bit

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

NOTE: To avoid errors, we recommend using Load instructions (except for LDB) to modify PWMCON register values.

RP0 — Register Pointer 0 D6H Set 1

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

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

Register pointer 0 can independently point to one of the 24 8-byte working register areas in the register file. Using the register pointers RP0 and RP1, you can select two 8-byte register slices at one time as active working register space. After a reset, RP0 points to address C0H in register set 1, selecting the 8-byte working register slice C0H–C7H.

.2 – .0 Not used for KS88C4504/P4504

RP1 — Register Pointer 1 D7H Set 1

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

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

Register pointer 1 can independently point to one of the 24 8-byte working register areas in the register file. Using the register pointers RP0 and RP1, you can select two 8-byte register slices at one time as active working register space. After a reset, RP1 points to address C8H in register set 1, selecting the 8-byte working register slice C8H–CFH.

.2 – .0 Not used for KS88C4504/P4504

SIOCON — SIO Control Register F1H Set 1, Bank 1

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

.7 SIO Shift Clock Selection Bit

0	Internal clock (P.S clock)
1	External clock (SCK)

.6 Data Direction Control Bit

0	MSB first mode
1	LSB first mode

.5 SIO Mode Selection Bit

0	Receive only mode
1	Transmit/receive mode

.4 Shift Start Edge Selection Bit

0	Falling edge start
1	Rising edge start

.3 SIO Counter Clear and Shift Start Bit

0	No action
1	Clear 3-bit counter and start shifting

.2 SIO Shift Operation Enable Bit

0	Disable shifter and clock counter
1	Enable shifter and clock counter

.1 SIO Interrupt Enable Bit

0	Disable SIO interrupt
1	Enable SIO interrupt

.0 SIO Interrupt Pending Bit

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

SIOPS — SIO Prescaler Register F3H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	–	–	–	–	–	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 – .0	Baud rate = (CPU clock/2) / (SIOPS + 1)

SPH — Stack Pointer (High Byte) D8H Set 1

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

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

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

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

SPL — Stack Pointer (Low Byte) D9H Set 1

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

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

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

SYM — System Mode Register DEH Set 1

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

.7 Tri-State External Interface Control Bit

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

.6 and .5

Not used for KS88C4504/P4504

.4 – .2 Fast Interrupt Level Selection Bits

0	0	0	IRQ0 (timer C overflow/capture & match)
0	0	1	IRQ1 (timer D overflow capture & match)
0	1	0	IRQ2 (SIO transmit/receive)
0	1	1	IRQ3 (timer A match)
1	0	0	IRQ4 (timer B match)
1	0	1	IRQ5 (INT0–INT1)
1	1	0	IRQ6 (INT2–INT7)
1	1	1	IRQ7 (PWM)

.1 Fast Interrupt Enable Bit

0	Disable fast interrupt processing
1	Enable fast interrupt processing

.0 Global Interrupt Enable Bit (note)

0	Disable global interrupt processing
1	Enable global interrupt processing

NOTE: Following a reset, you enable global interrupt processing by executing an EI instruction (not by writing a "1" to SYM.0).

TACON — Timer A Control Register D0H Set 1, Bank 0

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

.7 – .6 Timer A Input Clock Selection Bits

0	0	$f_{OSC}/1024$
0	1	$f_{OSC}/256$
1	0	$f_{OSC}/64$
1	1	$f_{OSC}/8$

.5–.4 Not used for the KS88C4504/P4504

.3 Timer A Counter Clear Bit

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

.2 Timer A Count Enable Bit

0	Disable count operation
1	Enable count operation

.1 Timer A Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Timer A Interrupt Pending Bit

0	No interrupt pending
0	*Clear pending bit*
1	Interrupt is pending

TBCON — Timer B Control Register D1H Set 1, Bank 0

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

.7 – .6 Timer B Input Clock Selection Bits

0	0	fosc/1024
0	1	fosc/256
1	0	fosc/64
1	1	fosc/8

.5–.4

Not used for the KS88C4504/P4504

.3 Timer B Counter Clear Bit

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

.2 Timer B Count Enable Bit

0	Disable count operation
1	Enable count operation

.1 Timer B Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Timer B Interrupt Pending Bit

0	No interrupt pending
0	*Clear pending bit*
1	Interrupt is pending

TCCON — Timer C Control Register E8H Set 1, Bank 0

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

.7 – .5 Timer C Input Clock Selection Bits

0	0	0	fosc/1024
0	1	0	fosc/256
1	0	0	fosc/64
1	1	0	fosc/8
0	0	1	fosc/1
0	1	1	External clock (TCCK) falling edge
1	0	1	External clock (TCCK) rising edge
1	1	1	Counter stop

.4–.3 Timer C Operating Mode Selection Bit

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

.2 Timer C Counter Clear Bit

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

.1 Timer C Match/Capture Interrupt

0	Disable interrupt
1	Enable interrupt

.0 Timer C Overflow Interrupt Enable Bit

0	Disable overflow interrupt
1	Enable overflow interrupt

TDCON — Timer D Control Register E9H Set 1, Bank 0

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

.7 – .5 Timer D Input Clock Selection Bits

0	0	0	fosc/1024
0	1	0	fosc/256
1	0	0	fosc/64
1	1	0	fosc/8
0	0	1	fosc/1
0	1	1	External clock (TDCK) falling edge
1	0	1	External clock (TDCK) rising edge
1	1	1	Counter stop

.4–.3 Timer D Operating Mode Selection Bit

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

.2 Timer D Counter Clear Bit

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

.1 Timer D Match/Capture Interrupt

0	Disable interrupt
1	Enable interrupt

.0 Timer D Overflow Interrupt Enable Bit

0	Disable overflow interrupt
1	Enable overflow interrupt

TCPND — Timer C Pending Register

EEH Set 1, Bank 0

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

.7 – .2

Not used for KS88C4504/P4504

.1 Timer C Overflow Interrupt Pending Bit

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

.0 Timer C Match/Capture Interrupt Pending Bit

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

TDPND — Timer D Pending Register EFH Set 1, Bank 0

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

.7 – .2

Not used for KS88C4504/P4504

.1 Timer D Overflow Interrupt Pending Bit

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

.0 Timer D Match/Capture Interrupt Pending Bit

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

5 INTERRUPT STRUCTURE

OVERVIEW

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

Levels

Interrupt levels are the main unit for interrupt priority assignment and recognition. All peripherals and I/O blocks can issue interrupt requests. In other words, peripheral and I/O operations are interrupt-driven. There are eight interrupt levels: IRQ0–IRQ7, also called level 0-level 7. Each interrupt level directly corresponds to an interrupt request number (IRQn). The total number of interrupt levels used in the interrupt structure varies from device to device. The KS88C4504/P4504 interrupt structure recognizes eight interrupt levels, IRQ0–IRQ7.

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

Vectors

Each interrupt level can have one or more interrupt vectors, or it may have no vector address assigned at all. The maximum number of vectors that can be supported for a given level is 128. (The actual number of vectors used for KS88-series devices will always be much smaller.) If an interrupt level has more than one vector address, the vector priorities are set in hardware. KS88C4504/P4504 have seventeen vectors— corresponding to each of the seventeen possible interrupt sources.

Sources

A source is any peripheral that generates an interrupt. A source can be an external pin or a counter overflow, for example. Each vector can have several interrupt sources. In the KS88C4504/P4504 interrupt structure, each source has its own vector address.

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

INTERRUPT TYPES

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

Type 1: One level (IRQn) + one vector (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 KS88C4504/P4504 microcontrollers, only interrupt types 1 and 3 are implemented.

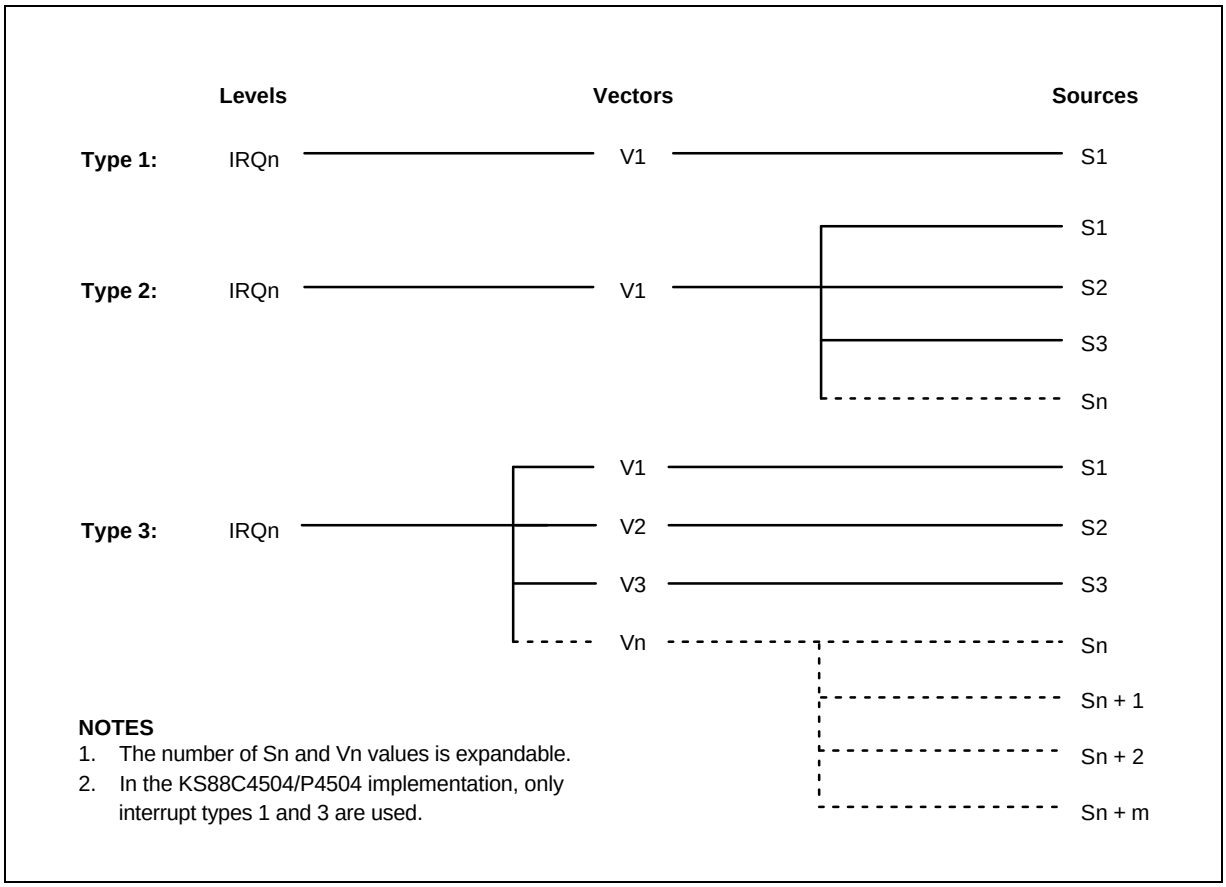


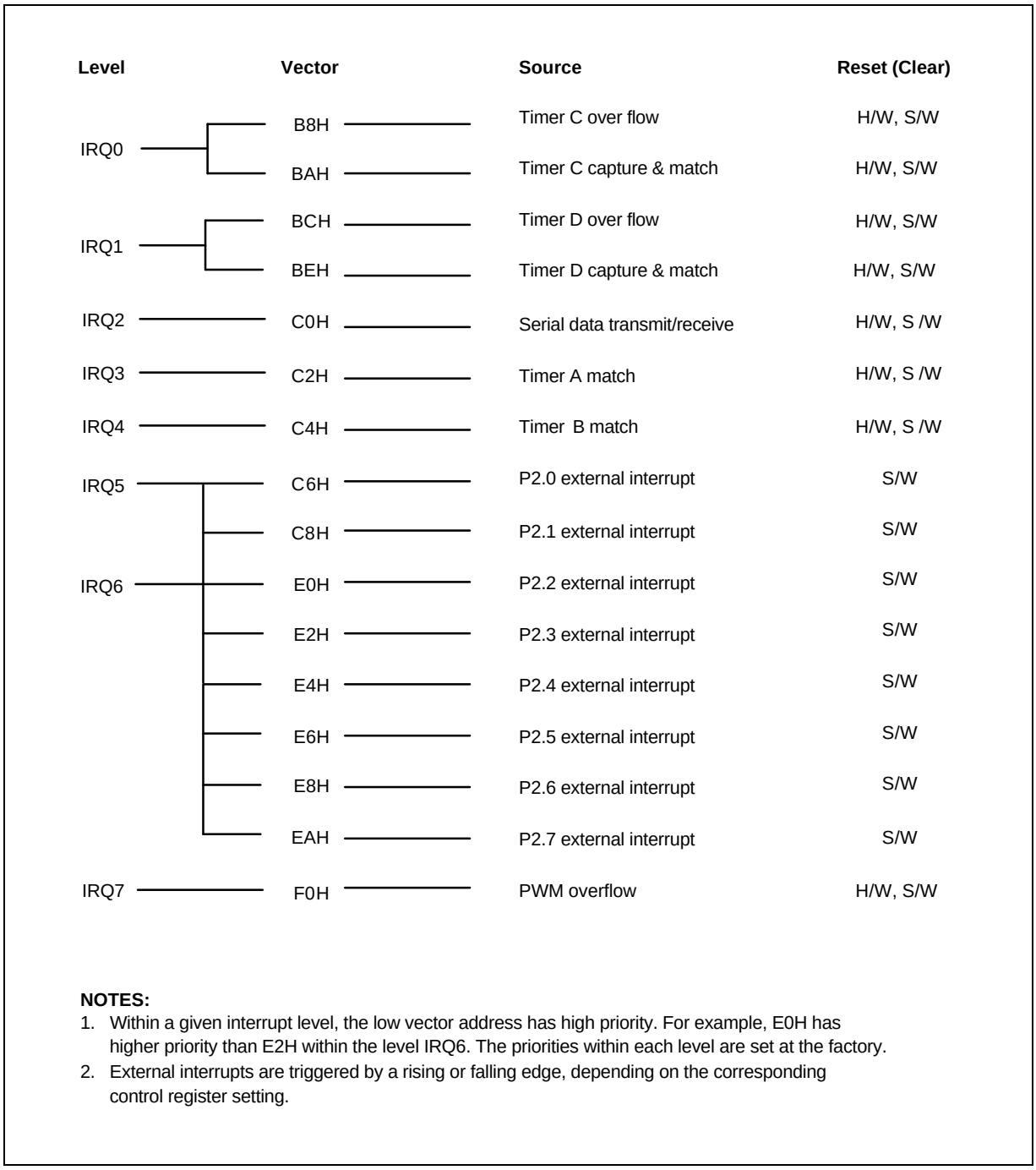
Figure 5-1. KS88-Series Interrupt Types

KS88C4504/P4504 INTERRUPT STRUCTURE

The KS88C4504/P4504 microcontroller supports sixteen interrupt sources. Each interrupt source has a corresponding interrupt vector address. Eight interrupt levels are used in the device-specific interrupt structure, which is shown in Figure 5-2.

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

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



NOTES:

1. Within a given interrupt level, the low vector address has high priority. For example, E0H has higher priority than E2H within the level IRQ6. The priorities within each level are set at the factory.
2. External interrupts are triggered by a rising or falling edge, depending on the corresponding control register setting.

Figure 5-2. KS88C4504 Interrupt Structure

INTERRUPT VECTOR ADDRESSES

Interrupt vector addresses for the KS88C4504/P4504 are stored in the first 256 bytes of the external program memory (ROM). Vectors for all interrupt levels are stored in the vector address area, 0H–FFH.

Unused locations in this range can be used as normal program memory. When writing an application program, you should be careful not to overwrite the address data stored in this area.

The program reset address in the program memory is 0100H.

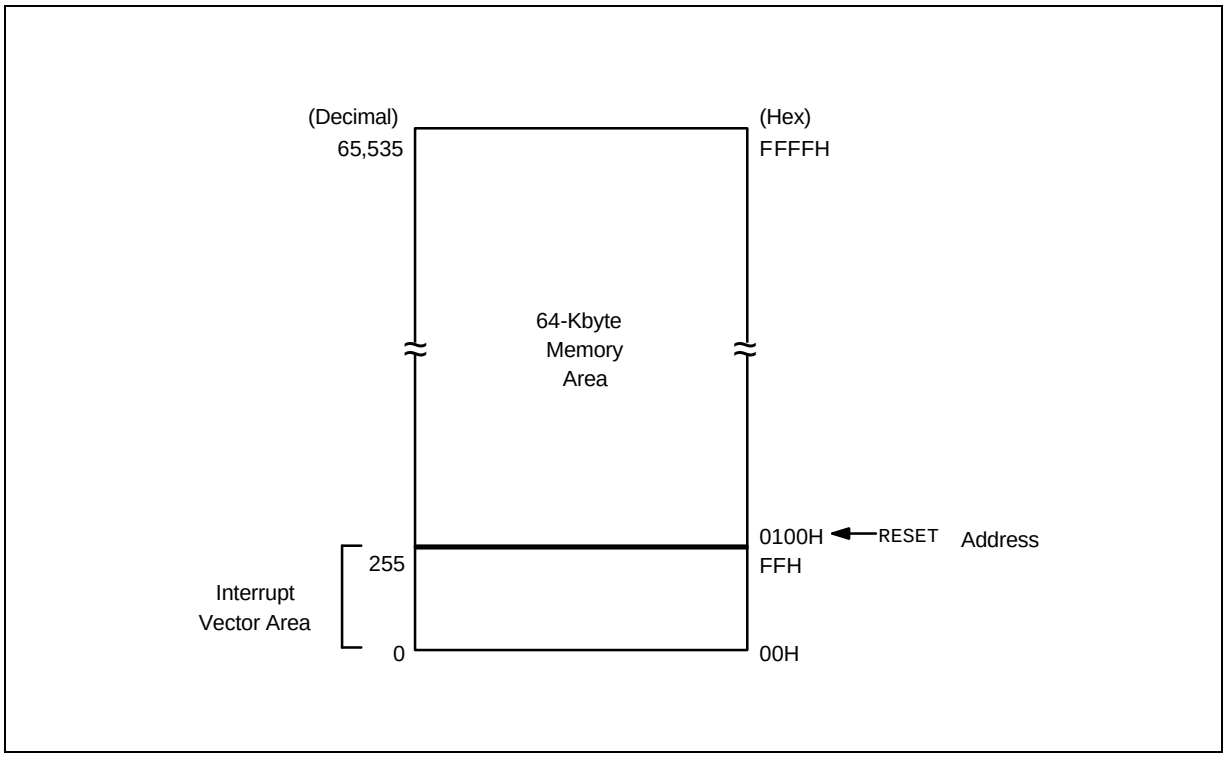


Figure 5-3. Vector Address Area in Program Memory (ROM)

Table 5-1. KS88C4504/P4504 Interrupt Vectors

Vector Address		Interrupt Source	Request		Reset/Clear	
Decimal Value	Hex Value		Interrupt Level	Priority in Level	H/W	S/W
184	B8H	Timer C overflow	IRQ0	0	√	√
186	BAH	Timer C capture & match		1		
188	BCH	Timer D overflow	IRQ1	0	√	√
190	BEH	Timer D capture & match		1		
192	C0H	Serial data transmit/receive	IRQ2	–	√	√
194	C2H	Timer A match	IRQ3	–	√	√
196	C4H	Timer B match	IRQ4	–	√	√
198	C6H	P2.0 external interrupt (level trigger)	IRQ5	0		√
200	C8H	P2.1 external interrupt (level trigger)		1		
224	E0H	P2.2 external interrupt (edge trigger)	IRQ6	0		√
226	E2H	P2.3 external interrupt (edge trigger)		1		
228	E4H	P2.4 external interrupt (edge trigger)		2		
230	E6H	P2.5 external interrupt (edge trigger)		3		
232	E8H	P2.6 external interrupt (edge trigger)		4		
234	EAH	P2.7 external interrupt (edge trigger)		5		
240	F0H	PWM overflow	IRQ7	–	√	√

NOTES:

1. Interrupt priorities are identified in inverse order: '0' is highest priority, '1' is the next highest, and so on.
2. If two or more interrupts within the same level contend, the interrupt with the lowest vector address has priority over one with a higher vector address. These priorities within levels are preset at the factory.

ENABLE/DISABLE INTERRUPT INSTRUCTIONS (EI, DI)

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

NOTE

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

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

SYSTEM-LEVEL INTERRUPT CONTROL REGISTERS

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

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

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

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

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

Table 5-2. Interrupt Control Register Overview

Control Register	ID	R/W	Function Description
Interrupt mask register	IMR	R/W	Bit settings in the IMR register enable or disable interrupt processing for each of the eight interrupt levels, IRQ0–IRQ7.
Interrupt priority register	IPR	R/W	Controls the relative processing priorities of the interrupt levels. The eight levels of the KS88C4504/P4504 are organized into three groups: A, B, and C. Group A is IRQ0 and IRQ1, group B is IRQ2–IRQ4, and group C is IRQ5–IRQ7
Interrupt request register	IRQ	R	This register contains a request pending bit for each of the eight interrupt levels, IRQ0–RQ7.
System mode register	SYM	R/W	Dynamic global interrupt processing enable and disable, fast interrupt processing.

NOTE

DI instruction must be used before changing the IMR, interrupt pending register and interrupt source control register. If IMR, interrupt pending register or source control register is controlled in EI status, program control could be in uncontrollable state.

INTERRUPT PROCESSING CONTROL POINTS

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

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

— Interrupt level enable/disable settings (IMR register)

— Interrupt level priority settings (IPR register)

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

NOTE

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

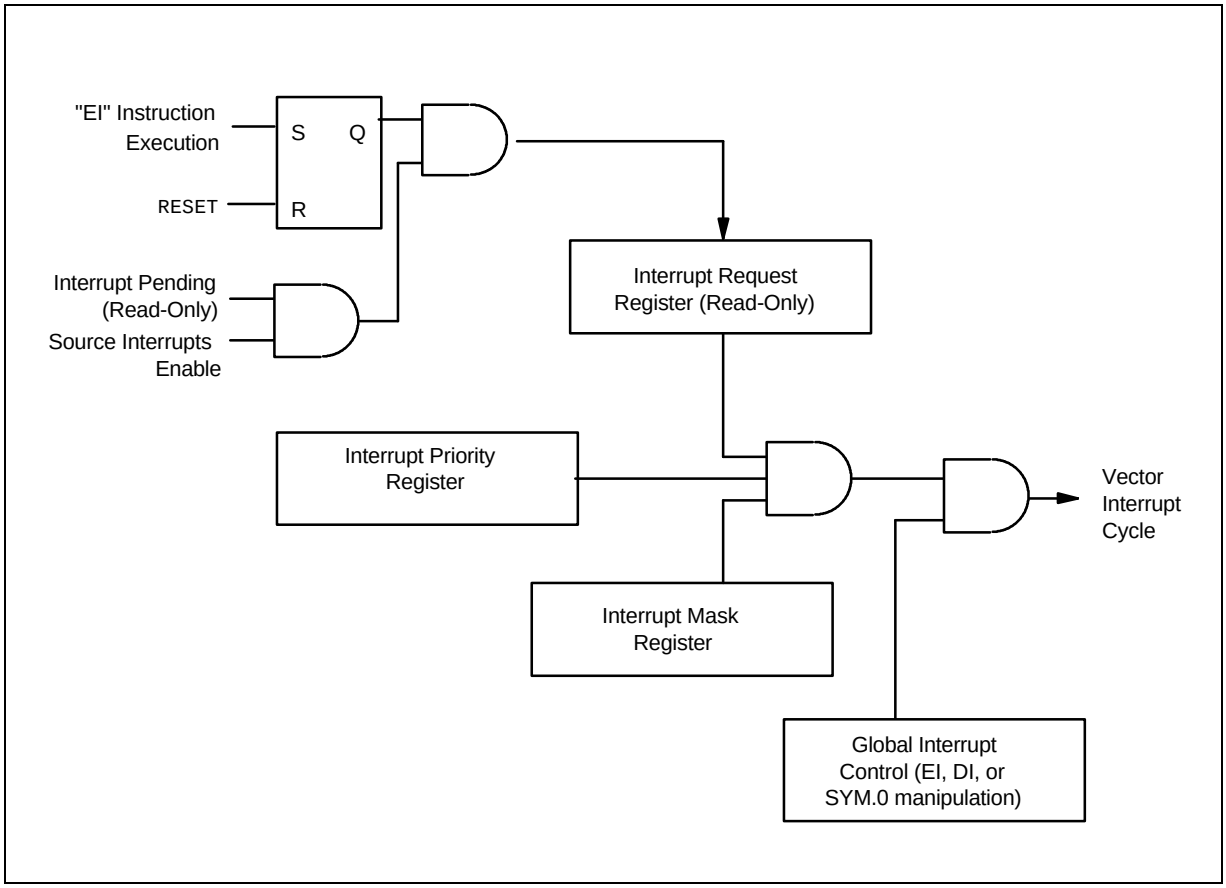


Figure 5-4. Interrupt Function Diagram

SYSTEM MODE REGISTER (SYM)

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

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

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

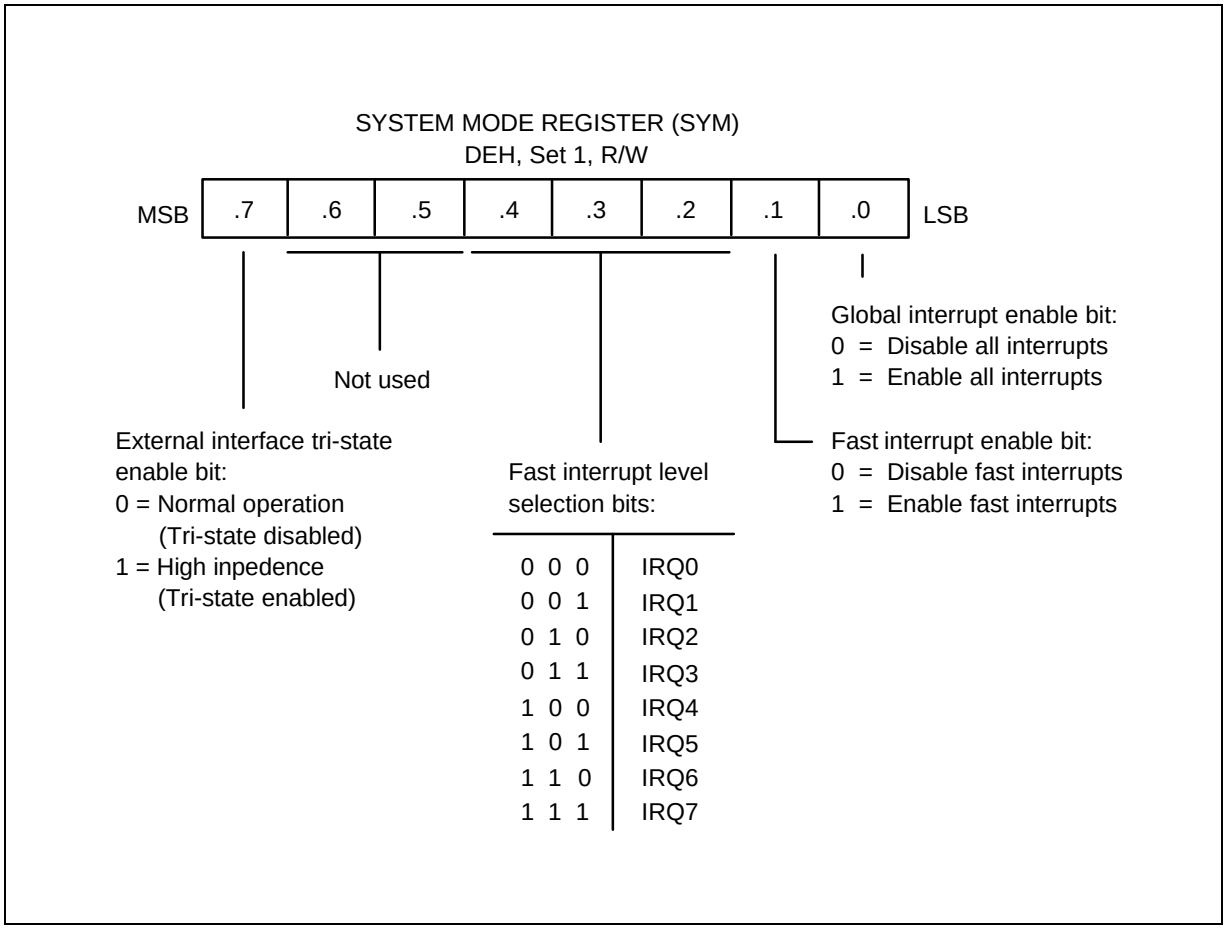


Figure 5-5. System Mode Register (SYM)

INTERRUPT MASK REGISTER (IMR)

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

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

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

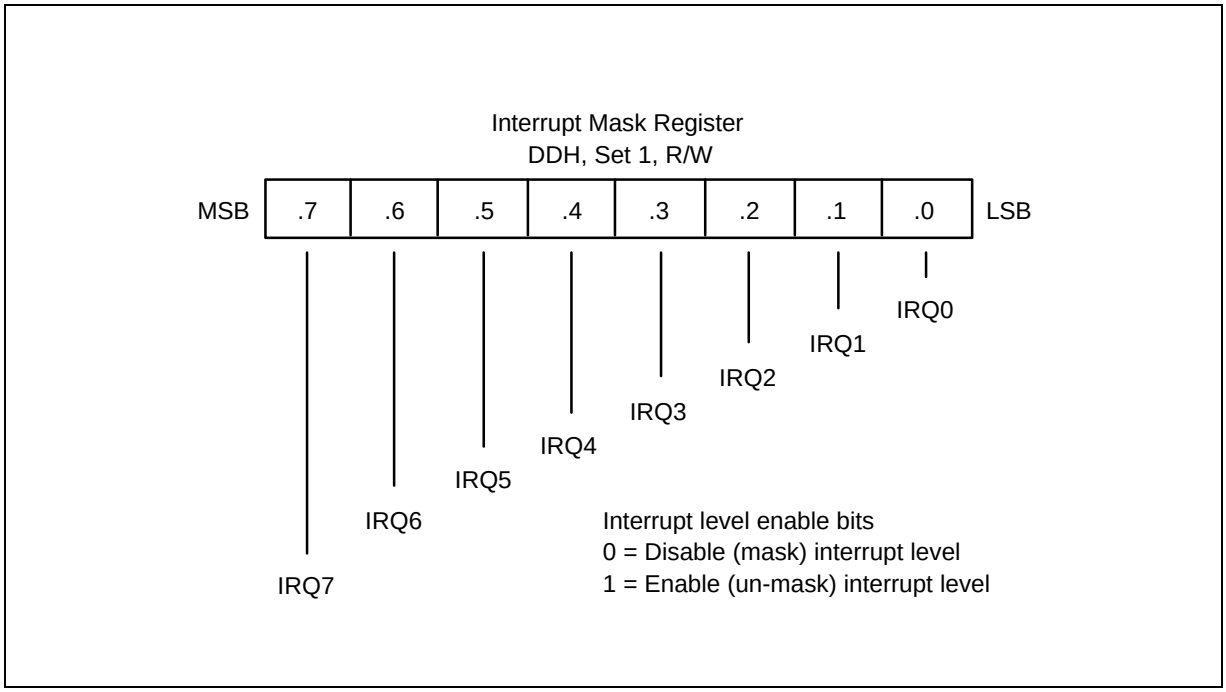


Figure 5-6. Interrupt Mask Register (IMR)

NOTE

Before IMR register is changed to any value, all interrupts must be disable. Using DI instruction is recommended.

INTERRUPT PRIORITY REGISTER (IPR)

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

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

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

Group A IRQ0, IRQ1

Group B IRQ2, IRQ3, IRQ4

Group C IRQ5, IRQ6, IRQ7

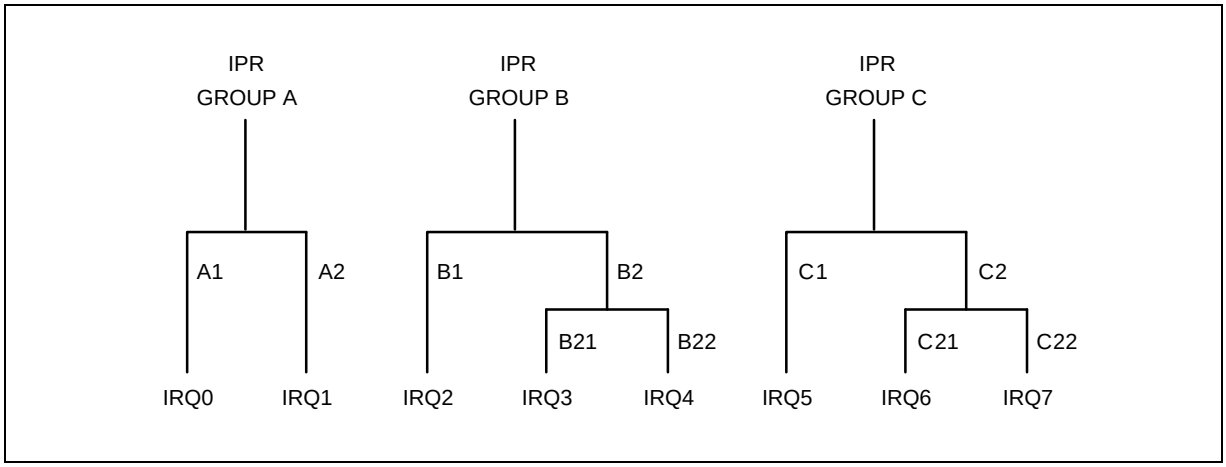


Figure 5-7. Interrupt Request Priority Groups

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

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

— Interrupt group C has a subgroup to provide an additional priority relationship between for interrupt levels 5, 6, and 7. IPR.6 defines the possible subgroup C relationships.

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

— Interrupt group B has a subgroup to provide an additional priority relationship between for interrupt levels 2, 3, and 4. IPR.3 defines the possible subgroup B relationships.

— IPR.2 controls the relative priorities of group B interrupts.

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

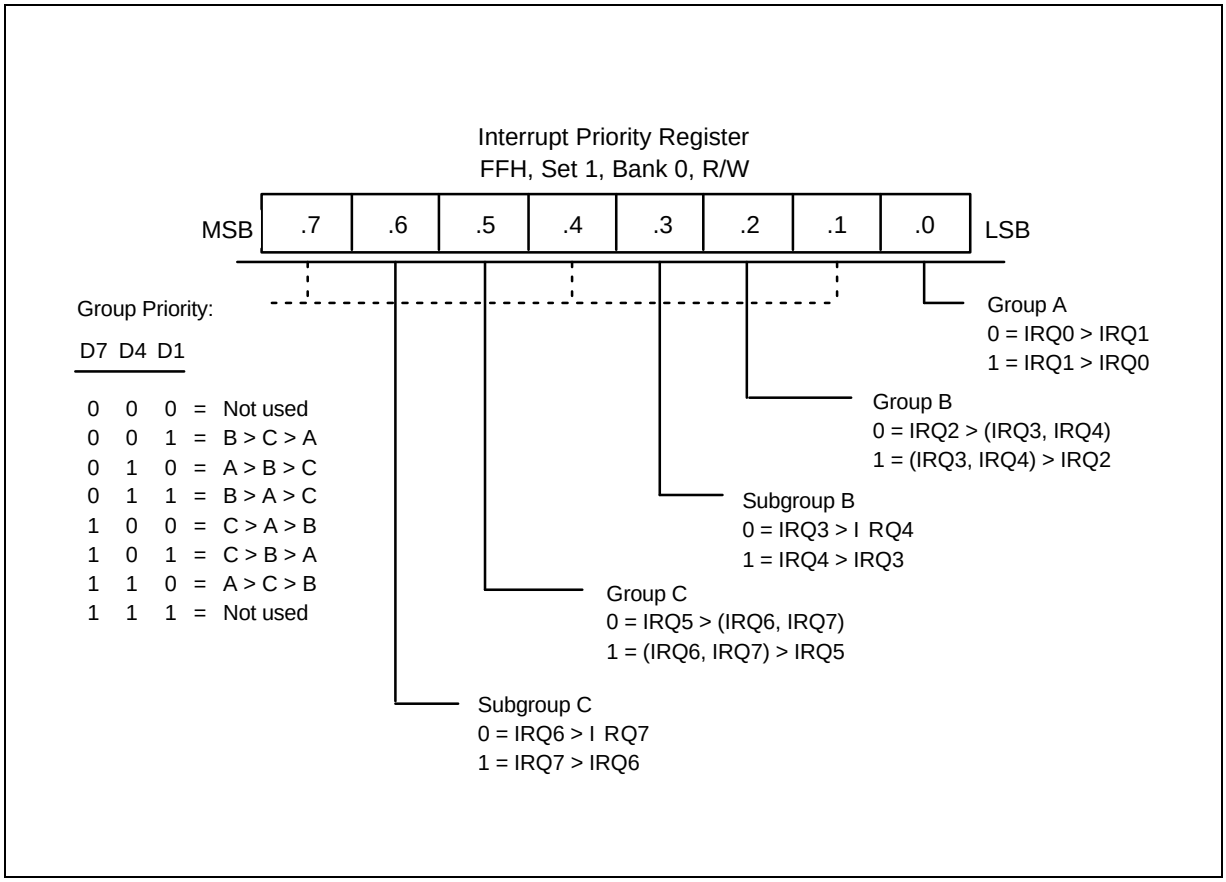


Figure 5-8. Interrupt Priority Register (IPR)

INTERRUPT REQUEST REGISTER (IRQ)

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

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

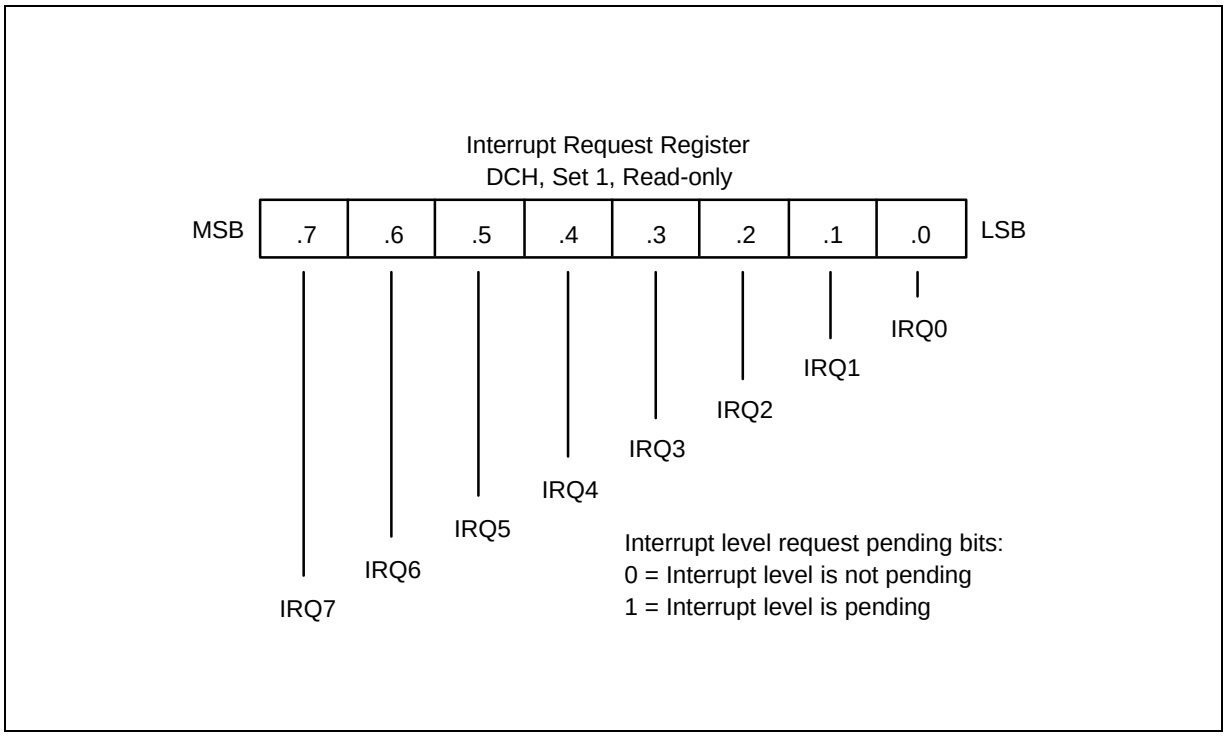


Figure 5-9. Interrupt Request Register (IRQ)

INTERRUPT PENDING FUNCTION TYPES

Overview

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

Pending Bits Cleared Automatically by Hardware

For interrupt pending bits that are cleared automatically by hardware, interrupt logic sets the corresponding pending bit to "1" when a request occurs. It then issues an IRQ pulse to inform the CPU that an interrupt is waiting to be serviced. The CPU acknowledges the interrupt source, executes the service routine, and clears the pending bit to "0". This type of pending bit is mapped and can, therefore, be read or written by application software.

Please refer to the page 5-4 (interrupt structure) to recognize which interrupts belong to this category of interrupts whose pending conditions are cleared automatically by hardware.

Pending Bits Cleared by the Service Routine

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

In the KS88C4504/P4504 interrupt structure, pending conditions for all external interrupt sources must be cleared by the program software's interrupt service routine.

INTERRUPT SOURCE POLLING SEQUENCE

The interrupt request polling and servicing sequence is as follows:

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

INTERRUPT SERVICE ROUTINES

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

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

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

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

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

GENERATING INTERRUPT VECTOR ADDRESSES

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

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

NOTE

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

NESTING OF VECTORED INTERRUPTS

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

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

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

INSTRUCTION POINTER (IP)

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

FAST INTERRUPT PROCESSING

The feature called *fast interrupt processing* lets you specify that an interrupt within a given level be completed in approximately six clock cycles instead of the usual 16 clock cycles. SYM.4–SYM.2 are used to select a specific interrupt level for fast processing and SYM.1 enables or disables fast interrupt processing.

Two other system registers support fast interrupt processing:

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

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

NOTES

1. For the KS88C4504/P4504 microcontrollers, the service routine for any of the eight interrupt levels can be selected for fast interrupt processing.

2. If you want to use a fast interrupt in multi source interrupt vector, the fast interrupt may not be processed when you use two sources as interrupt vector in normal mode. But it is possible when you use only one source as interrupt vector.

Procedure for Initiating Fast Interrupts

To initiate fast interrupt processing, follow these steps:

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

Fast Interrupt Service Routine

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

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

Relationship to Interrupt Pending Bit Types

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

Programming Guidelines

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

NOTE

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

☞ PROGRAMMING TIP — Setting Up the KS88C4504 Interrupt Control Structure

This example shows how to enable interrupts for select interrupt sources, disable interrupt for other sources, and to set interrupt priorities for the KS88C4504. The program does the following:

— Enable interrupts for timer C, serial
— Disable timer A/B/D and all external interrupts
— Set interrupt priorities as SIO > timer C > PWM

```
•
•
•
DI                          ;  Disable interrupts
LD      IMR,#85H            ;  IRQ0, IRQ2, IRQ7 are selected
LD      IPR,#12H            ;  IRQ2 > IRQ0 > IRQ7
LD      TCDATAH,#00H        ;
LD      TCDATAL,#0FFH       ;
LD      TCCON,#06H          ;  Timer C interrupt enable
LD      TCPND,#00H          ;  TC Pending clear
SB1
LD      SIOCON,#3EH         ;  SIO interrupt enable
LD      SIOPS,#29H          ;  SIO Prescaler setting
LD      PWMCON,#0FAH        ;  PWM overflow interrupt enable
•
•
•
EI                          ;  Enable interrupts
```

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

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

NOTES

7 CLOCK CIRCUIT

OVERVIEW

The clock frequency generated for the KS88C4504/P4504 by an external crystal can range from 1 MHz to 25 MHz. The maximum CPU clock frequency is 25 MHz. The X_{IN} and X_{OUT} pins connect the external oscillator or clock source to the on-chip clock circuit.

SYSTEM CLOCK CIRCUIT

The system clock circuit has the following components:

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

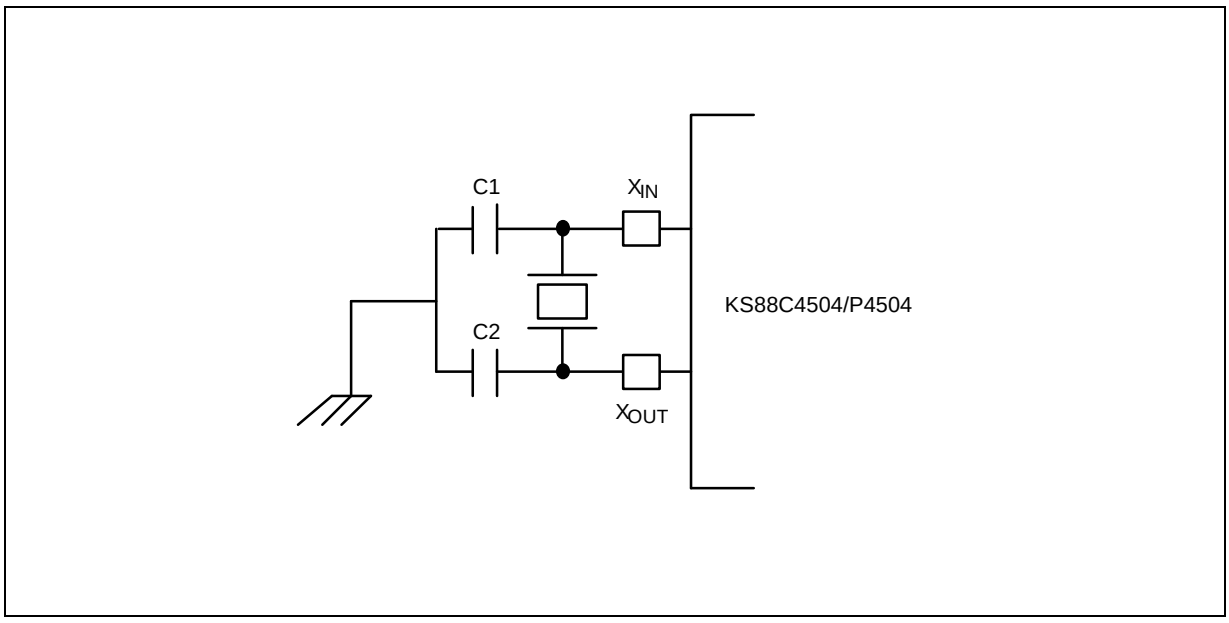


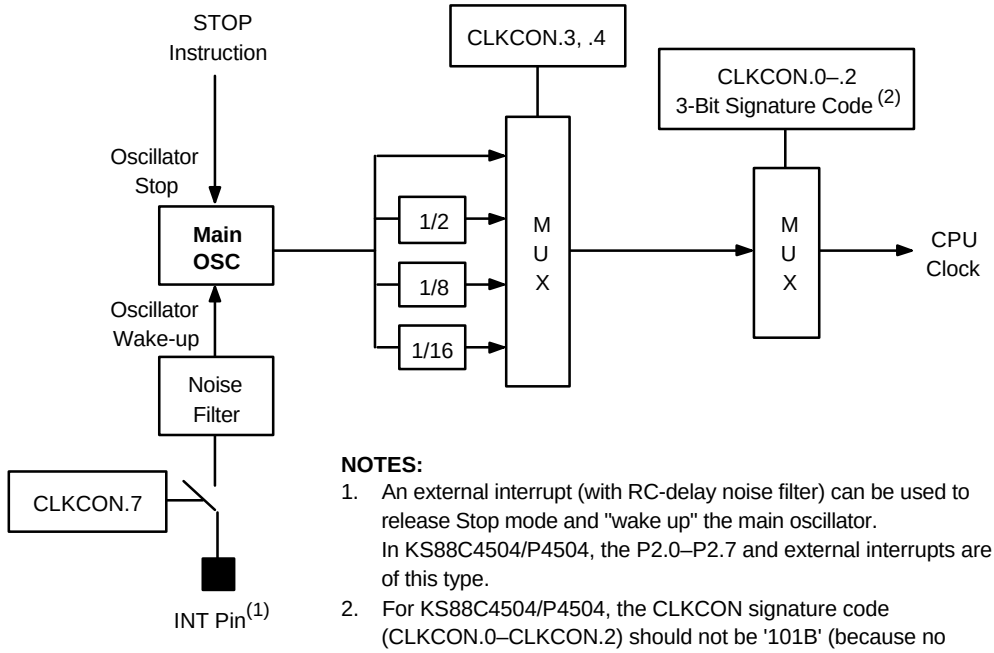
Figure 7-1. Main Oscillator Circuit (External Crystal or Ceramic Resonator)

CLOCK STATUS DURING POWER-DOWN MODES

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

— In Stop mode, the main oscillator is halted. Stop mode is released, and the oscillator started, by a reset operation or an external interrupt (with RC delay noise filter).

— In Idle mode, the internal clock signal is gated to the CPU, but not to interrupt structure, timers and timer/counters. Idle mode is released by a reset or by an external or internal interrupt.



NOTES:

1. An external interrupt (with RC-delay noise filter) can be used to release Stop mode and "wake up" the main oscillator. In KS88C4504/P4504, the P2.0–P2.7 and external interrupts are of this type.
2. For KS88C4504/P4504, the CLKCON signature code (CLKCON.0–CLKCON.2) should not be '101B' (because no subsystem clock is implemented).

Figure 7-2. System Clock Circuit Diagram

SYSTEM CLOCK CONTROL REGISTER (CLKCON)

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

— Oscillator IRQ wake-up function enable/disable

— Oscillator frequency divide-by value

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

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

For the KS88C4504/P4504 microcontrollers, the CLKCON.2–CLKCON.0 system clock signature code must be any other value than '101B'. (The '101B' setting is invalid because a subsystem clock is not implemented.) The reset value for the clock signature code is '000B' and should remain so during normal operation.

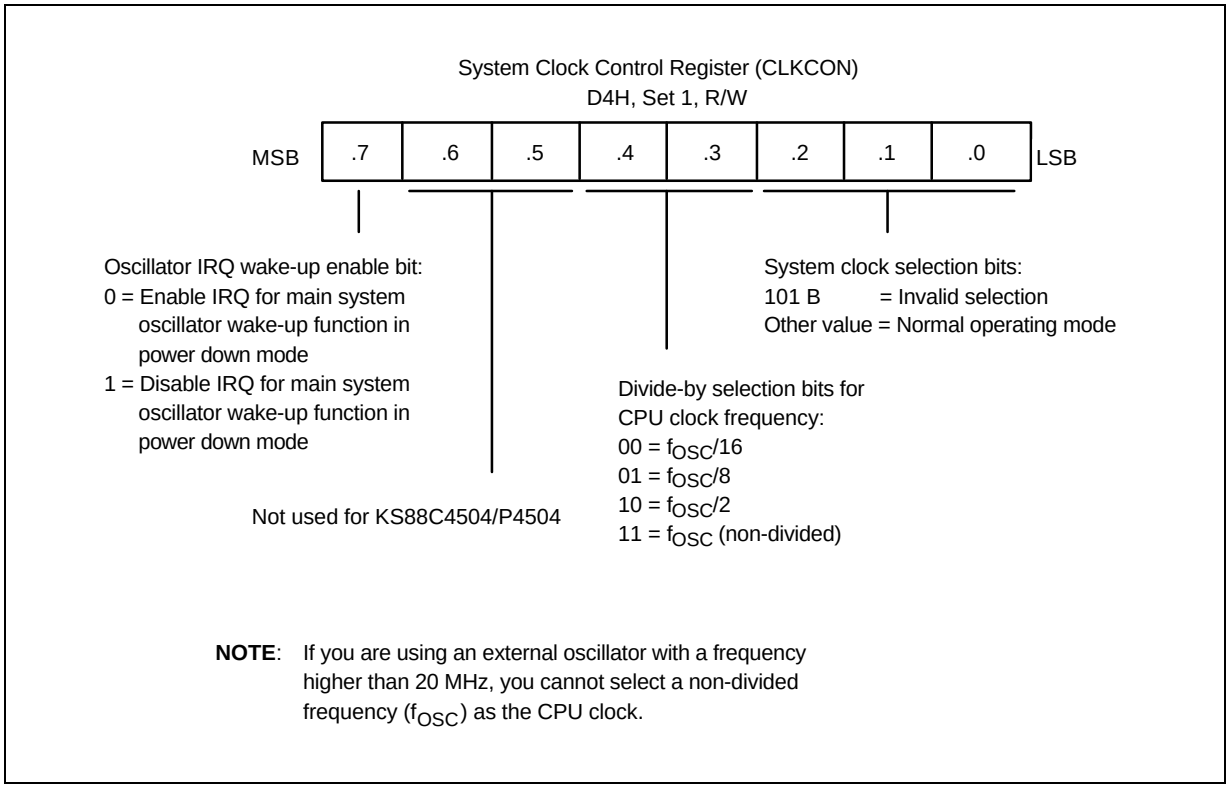


Figure 7-3. System Clock Control Register (CLKCON)

NOTES

8 RESET and POWER-DOWN

SYSTEM RESET

OVERVIEW

During a power-on reset, the voltage at V_{DD} goes to High level and the RESET pin is forced to Low level. The RESET signal is input through a Schmitt trigger circuit where it is then synchronized with the CPU clock. This procedure brings KS88C4504/P4504 into a known operating status.

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

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

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

— All interrupts are disabled.

— The watchdog function (basic timer) is enabled.

— Ports 0-5 are set to input mode

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

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

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

— EXTBUS register is set to 00H, it can affect external interface output while EA pin is low.

NORMAL MODE RESET OPERATION

In normal (masked ROM) mode, the EA pin is tied to V_{SS}. A reset enables access to the 4-Kbyte on-chip ROM. (The external interface is not automatically configured).

ROM-LESS MODE RESET OPERATION

To configure KS88C4504/P4504 as a ROM-less device, you must apply a constant 5 V current to the EA pin. Assuming the EA pin is held to high level (5 V) when a reset occurs, ROM-less mode is entered and the external interface is configured automatically.

NOTE

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

HARDWARE RESET VALUES

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

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

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

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

Table 8-1. KS88C4504/P4504 Set1 Register and Values after RESET (Masked ROM Mode)

Register Name	Mnemonic	Address		Bit Values after RESET (EA Pin is Low)							
		Dec	Hex	7	6	5	4	3	2	1	0
Timer A Control Register	TACON	208	D0H	0	0	0	0	0	0	0	0
Timer B Control Register	TBCON	209	D1H	0	0	0	0	0	0	0	0
Timer A Data Register	TADATA	210	D2H	0	0	0	0	0	0	0	0
Basic Timer Control Register	BTCON	211	D3H	0	0	0	0	0	0	0	0
Clock Control Register	CLKCON	212	D4H	0	0	0	0	0	0	0	0
System Flags Register	FLAGS	213	D5H	x	x	x	x	x	x	x	x
Register Pointer 0	RP0	214	D6H	1	1	0	0	0	–	–	–
Register Pointer 1	RP1	215	D7H	1	1	0	0	1	–	–	–
Stack Pointer (High Byte)	SPH	216	D8H	x	x	x	x	x	x	x	x
Stack Pointer (Low Byte)	SPL	217	D9H	x	x	x	x	x	x	x	x
Instruction Pointer (High Byte)	IPH	218	DAH	x	x	x	x	x	x	x	x
Instruction Pointer (Low Byte)	IPL	219	DBH	x	x	x	x	x	x	x	x
Interrupt Request Register	IRQ	220	DCH	0	0	0	0	0	0	0	0
Interrupt Mask Register	IMR	221	DDH	x	x	x	x	x	x	x	x
System Mode Register	SYM	222	DEH	0	–	–	x	x	x	0	0
Register Page Pointer	PP	223	DFH	0	0	0	0	0	0	0	0
Port 0 Data Register	P0	224	E0H	0	0	0	0	0	0	0	0
Port 1 Data Register	P1	225	E1H	0	0	0	0	0	0	0	0
Port 2 Data Register	P2	226	E2H	0	0	0	0	0	0	0	0
Port 3 Data Register	P3	227	E3H	0	0	0	0	0	0	0	0
Port 4 Data Register	P4	228	E4H	0	0	0	0	0	0	0	0
Port 5 Data Register	P5	229	E5H	0	0	0	0	0	0	0	0
Timer B Data Register	TBDATA	230	E6H	0	0	0	0	0	0	0	0

Table 8-2. KS88C4504/P4504 Set1, Bank0 Register Values after RESET (Masked ROM Mode)

Register Name	Mnemonic	Address		Bit Values after RESET (EA Pin is Low)							
		Dec	Hex	7	6	5	4	3	2	1	0
Port 2 Interrupt Control Register	P2INT	231	E7H	0	0	0	0	0	0	0	0
Timer C Control Register	TCCON	232	E8H	0	0	0	0	0	0	0	0
Timer D Control Register	TDCON	233	E9H	0	0	0	0	0	0	0	0
Timer C Data Register (High Byte)	TCDATAH	234	EAH	0	0	0	0	0	0	0	0
Timer C Data Register (Low Byte)	TCDATAL	235	EBH	0	0	0	0	0	0	0	0
Timer D Data Register (High Byte)	TDDATAH	236	ECH	0	0	0	0	0	0	0	0
Timer D Data Register (Low Byte)	TDDATAL	237	EDH	0	0	0	0	0	0	0	0
Timer C Pending Register	TCPND	238	EEH	0	0	0	0	0	0	0	0
Timer D Pending Register	TDPND	239	EFH	0	0	0	0	0	0	0	0
Port 0 Control Register (High Byte)	P0CONH	240	F0H	0	0	0	0	0	0	0	0
Port 0 Control Register (Low Byte)	P0CONL	241	F1H	0	0	0	0	0	0	0	0
Port 1 Control Register	P1CONH	242	F2H	0	0	0	0	0	0	0	0
Port 1 Pull-up Resistor Control Register	P1CONL	243	F3H	0	0	0	0	0	0	0	0
Port 2 Control Register (High Byte)	P2CONH	244	F4H	0	0	0	0	0	0	0	0
Port 2 Control Register (Low Byte)	P2CONL	245	F5H	0	0	0	0	0	0	0	0
Port 3 Control Register (High Byte)	P3CONH	246	F6H	0	0	0	0	0	0	0	0
Port 3 Control Register (Low Byte)	P3CONL	247	F7H	0	0	0	0	0	0	0	0
Port 4 Control Register (High Byte)	P4CONH	248	F8H	0	0	0	0	0	0	0	0
Port 4 Control Register (Low Byte)	P4CONL	249	F9H	0	0	0	0	0	0	0	0
Port 5 Control Register	P5CON	250	FAH	–	–	–	–	0	0	0	0
Port 2 Interrupt Pending Register	P2PND	251	FBH	0	0	0	0	0	0	0	0
Test Mode Register	TESTMOD	252	FCH	–	–	–	–	–	–	–	–
Basic Timer Data Register	BTCNT	253	FDH	0	0	0	0	0	0	0	0
External Memory Timing Register	EMT	254	FEH	0	–	–	–	–	–	0	–
Interrupt Priority Register	IPR	255	FFH	x	x	x	x	x	x	x	x

Table 8-3. KS88C4504/P4504 Set1, Bank1 Register Values after RESET (Masked ROM Mode)

Register Name	Mnemonic	Address		Bit Values after RESET (EA Pin is High)							
		Dec	Hex	7	6	5	4	3	2	1	0
Synchronous SIO Control Register	SIOCON	241	F1H	0	0	0	0	0	0	0	0
Synchronous SIO Data Register	SIODATA	242	F2H	x	x	x	x	x	x	x	x
A/D Converter Input Register	ADDATA	244	F4H	x	x	x	x	x	x	x	x
External Bus Control Register	EXTBUS	245	F5H	–	–	–	–	–	–	–	0
A/D Converter Control Register	ADCON	247	F7H	–	0	0	0	x	0	0	0
PWM Module Control Register	PWMCON	249	F9H	0	0	0	0	0	0	0	0
PWM1 Data Register	PWM1	250	FAH	0	0	0	0	0	0	0	0
PWM0 Data Register	PWM0	251	FBH	0	0	0	0	0	0	0	0

POWER-DOWN MODES

STOP MODE

Stop mode is invoked by the instruction STOP (opcode 7FH). In Stop mode, the operation of the CPU and all peripherals is halted. All system functions stop when the clock "freezes," but data stored in the internal register file is retained. Stop mode can be released in one of two ways: by a reset or by an external interrupt (with RC delay).

NOTE

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

Using RESET to Release Stop Mode

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

Using an External Interrupt to Release Stop Mode

Only external interrupts with an RC-delay noise filter circuit can be used to release Stop mode. Which interrupt you can use to release Stop mode in a given situation depends on the microcontroller's current internal operating mode. The external interrupts in the KS88C4504/P4504 interrupt structure that can be used to release Stop mode are:

— External interrupts P2.0–P2.7 (INT0–INT7)

Please note the following conditions for Stop mode release:

— If you release Stop mode using an external interrupt, the current values in system and peripheral control registers are unchanged.

— If you use an external interrupt for Stop mode release, you can also program the duration of the oscillation stabilization interval. To do this, you must make the appropriate control and clock settings *before* entering Stop mode.

— When the Stop mode is released by external interrupt, the CLKCON.4 and CLKCON.3 bit-pair setting remains unchanged and the currently selected clock value is used.

— The external interrupt is serviced when the Stop mode release occurs. Following the IRET from the service routine, the instruction immediately following the one that initiated Stop mode is executed.

IDLE MODE

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

There are two ways to release Idle mode:

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

☞ PROGRAMMING TIP — Sample KS88C4504 Initialization Routine

The following sample program suggests initialization settings for the KS88C4504 address space, interrupt vectors, and peripheral functions:

```
;               << Register file reference >>

                .INCLUDE    "C: \ SMDS2P \ INCLUDE \ REG \ 88C4504.REG"

;               << User Equation Definition >>

                .INCLUDE    "C: \ EQU.TBL"

;               << Interrupt Vector Addresses >>

                .ORG        00B8H
                .DW         TC_int              ;  IRQ0 timer C overflow interrupt
                .ORG        00BCH
                .DW         TD_int              ;  IRQ1 timer D overflow interrupt
;               00BAH,00BEH                     ;  timer C/D match & capture int. is not used.

                .ORG        00C0H
                .DW         SIOINT_T_R          ;  IRQ2
                .DW         TA_int              ;  IRQ3
                .DW         TB_int              ;  IRQ4
                .DW         EXT20_int           ;  IRQ5: level triggered ext. int.
                .DW         EXT21_int           ;  IRQ5: level triggered ext. int.
;               00CAH–00CFH: Reserved

                .ORG        00E0H
                .DW         EXT22_int           ;  IRQ6: Edge triggered ext. int.
                .DW         EXT23_int           ;  IRQ6
                .DW         EXT24_int           ;  IRQ6
                .DW         EXT25_int           ;  IRQ6
                .DW         EXT26_int           ;  IRQ6
                .DW         EXT27_int           ;  IRQ6
;               00ECH–00EFH: Reserved

                .ORG        00F0H
                .DW         PWM_int             ;  IRQ7: PWM over low int.

;               << Reset Vector >>

                .ORG        0100H
                JP          t,INITIAL

                •
                •
                •
```

☞ **PROGRAMMING TIP — Sample KS88C4504 Initialization Routine (Continued)**

```
;               << System and Peripheral Initialization >>

                .ORG        0200H
INITIAL:        DI

;               <System register setting>

                LD          SYM,#00000000B      ;  Fast, global interrupt disable
                LD          EMT,#00000000B      ;  'No wait' and internal stack area select
                LD          SPH,#00H            ;  Stack pointer (high byte) to zero
                LD          SPL,#0FFH           ;  Stack pointer (low byte) to zero
                LD          CLKCON,#10H         ;  fOSC/2 is selected for CPU clock

;               <Interrupt settings>

                LD          IPR,#16H            ;  Interrupt priorities
                                                ;  IRQ3 > 4 > 2 > 0 > 1 > 5 > 6 > 7
                LD          IMR,#00001101B      ;  IRQ levels 0, 2, and 3 enable
                                                ;  Level 0  =  timer C interrupt
                                                ;  Level 2  =  SIO interrupt
                                                ;  Level 3  =  timer A interrupt
INI_PERI_SET:

;               <Port 0 setting>

                LD          P0CONH,#0FFH        ;  Output, push-pull
                LD          P0CONL,#0FFH

;               <Port 1 setting>

                LD          P1CONH,#0FFH        ;  Output, push-pull
                LD          P1CONL,#0FFH

;               <Port 2 setting>

                LD          P2CONL,#00100000B   ;  P2.0 input mode, low level interrupt
                                                ;  P2.1 input mode, low level interrupt
                                                ;  P2.2 input mode with pull-up
                                                ;  P2.3 input mode, falling edge interrupt
                LD          P2CONH,#00111100B   ;  P2.4 input, falling edge interrupt
                                                ;  P2.5, P2.6 output mode
                                                ;  P2.7 input, falling edge interrupt
                LD          P2INT,#00H          ;  All external interrupt disable
```

☞ PROGRAMMING TIP — Sample KS88C4504 Initialization Routine (Continued)

```
;       <Port 3 setting>

        LD      P3CONL,#10101010B   ;  P3.0–P3.5 input mode with pull-up
        LD      P3CONH,#01001010B   ;  P3.6 input mode
                                    ;  P3.7 PWM1 output mode
;       <Port 4 setting>

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

;       <Port 5 setting>

        LD      P5CON,#0FH          ;  Output, push-pull

;       <Timer A>
        LD      TADATA,#8H          ;  Timer A clock source clock divided by 9
        LD      TACON,#00001110B    ;  If fOSC = 11.0592 MHz/2
                                    ;  fOSC /1024/9 → 1.66 ms
                                    ;  Interrupt enable
;       <Timer B>                   ;  Disabled

;       <Timer C>                   ;  Disabled

;       <Timer D>                   ;  Disabled

        LD      TBCON,#00H
        LD      TCCON,#00H
        LD      TDCON,#00H
```

☞ PROGRAMMING TIP — Sample KS88C4504 Initialization Routine (Continued)

```
;           <SIO setting>                       ;  Internal clock
            SB1
            LD          SIOCON,#01010010B       ;  LSB first
                                                ;  Receive only mode
                                                ;  Rising edge start
                                                ;  Disable shifter and clock counter
                                                ;  Enable SIO interrupt
            LD          SIOPS,#00H              ;  Pending bit clear
                                                ;  Baud rate = (Xin/2)/(SIOPS + 1)
;           << Register Initialization >>

            SB0

            SRP         #0C0H

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

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

;           <Initialize other registers>

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

;           << Main Loop >>

MAIN:       NOP                                 ;  Start main loop
            LD          BTCON,#03H              ;  Enable watchdog timer, clear BTCNT, and
                                                ;  Basic timer clock input divider.
                                                ;
            •
            •
            •
            CALL        KEY_SCAN
            •
            •
            •
            CALL        LED_DISPLAY
            •
            •
            •
            CALL        JOB
            •
            •
            •
            JP          t,MAIN
```

☞ PROGRAMMING TIP — Sample KS88C4504 Initialization Routine (Continued)

```
;               <Subroutine 1>

KEY_SCAN:
                NOP
                •
                •
                •
                RET

;               <Subroutine 2>

LED_DISPLAY:
                NOP
                •
                •
                •
                RET

;               <Subroutine 3>

JOB:            NOP
                •
                •
                •
                RET

;               << Interrupt Service Routine >>

TA_int:         PUSH    RP0
                PUSH    RP1
                SRP     #TA_REG                 ;  Example: TA_REG = 00H
                •
                •
                •
                AND     TACON,#11111110B        ;  Clear pending bit (omissible)
                POP     RP1
                POP     RP0
                IRET

TB_int:         AND     TBCON,#11111110B        ;  Omissible instruction
                IRET
TC_int:         AND     TCPND,#11111101B        ;  Omissible instruction
                IRET
TD_int:         AND     TDPND,#11111101B        ;  Omissible instruction
                IRET
```

☞ **PROGRAMMING TIP — Sample KS88C4504 Initialization Routine (Concluded)**

```
;               << Other Interrupt Vectors >>

EXT20_int:      LD      P2PND,#11111110B        ;       IRQ5
                •
                •
                IRET
EXT21_int:      LD      P2PND,#11111101B        ;       IRQ5
                •
                •
                IRET
EXT22_int:      LD      P2PND,#11111011B        ;       IRQ6
                •
                •
                IRET
EXT23_int:      LD      P2PND,#11110111B        ;       IRQ6
                •
                •
                IRET
EXT24_int:      LD      P2PND,#11101111B        ;       IRQ6
                •
                •
                IRET
EXT25_int:      LD      P2PND,#11011111B        ;       IRQ6
                •
                •
                IRET
EXT26_int:      LD      P2PND,#10111111B        ;       IRQ6
                •
                •
                IRET
EXT27_int:      LD      P2PND,#01111111B        ;       IRQ6
                •
                •
                IRET

SIOINT_T_R      AND     SIOCON,#11111110H       ;   IRQ2 (omissible)
                IRET

                END
```

NOTE. When clearing a interrupt pending bit by software, using ***LD*** instruction is recommended to prevent malfunction of interrupt operation.

NOTES

9 I/O PORTS

OVERVIEW

The KS88C4504/P4504 microcontrollers have 34 I/O ports. Each port can be flexibly configured to meet application design requirements. The CPU accesses ports by directly writing or reading port registers. No special I/O instructions are required. Table 1-2 gives you an overview of port functions:

KS88C4504/P4504 has 48 pins that are dedicated to I/O and 8 pins that are used for open-drain output. Each port can be flexibly configured to meet application design requirements.

All ports can be configured to input or output mode. A0–A15, D0-D7, PM, DM, RD, WR, can be configured as external interface only.

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

PORT DATA REGISTERS

Data registers for ports 0–5 have the structure shown in Figure 9-1:

PORT 0

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

A reset clears the port 0 control register, P0CONH/P0CONL, to '00H', configuring all port 0 pins as inputs.

High nibble of P0 (P0.4–P0.7) can be configured as ADC input or normal digital input.

NOTE

High nibble of P0 (P0.4–P0.7) can be connected with *pull-down* resistors by setting appropriate value to P0CONH register.

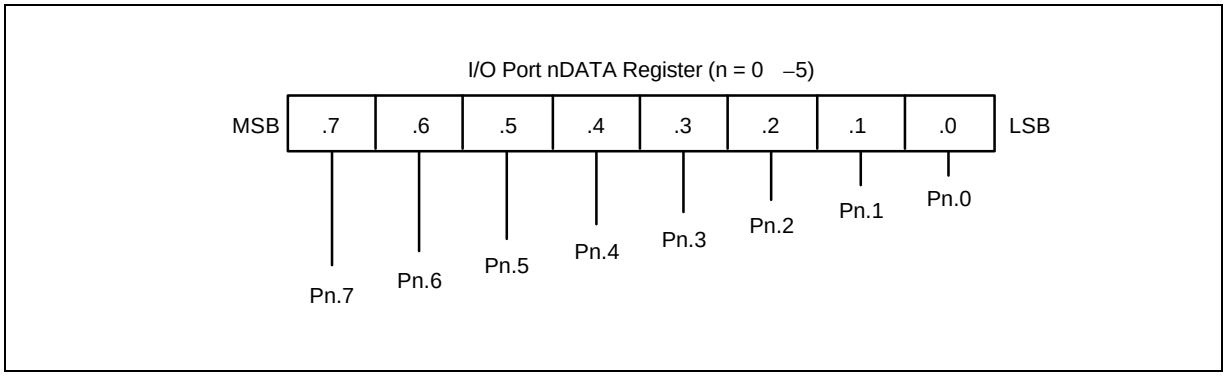


Figure 9-1. Port Data Register Structure

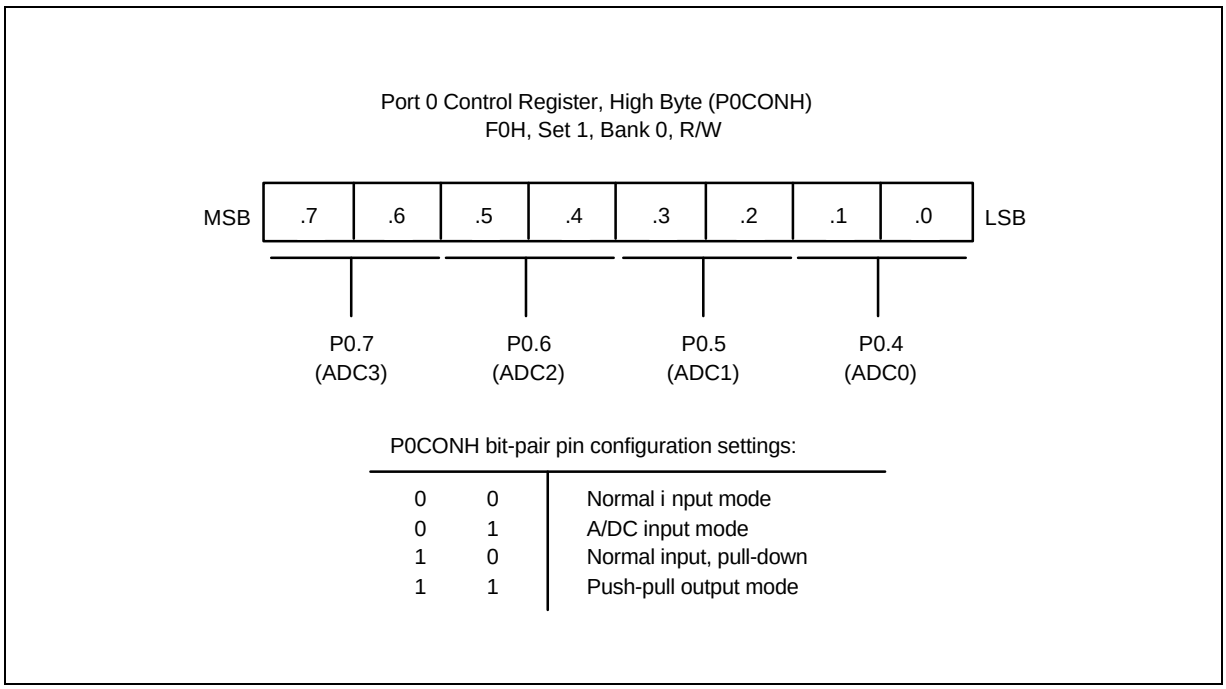


Figure 9-2. Port 0 High-byte Control Register (P0CONH)

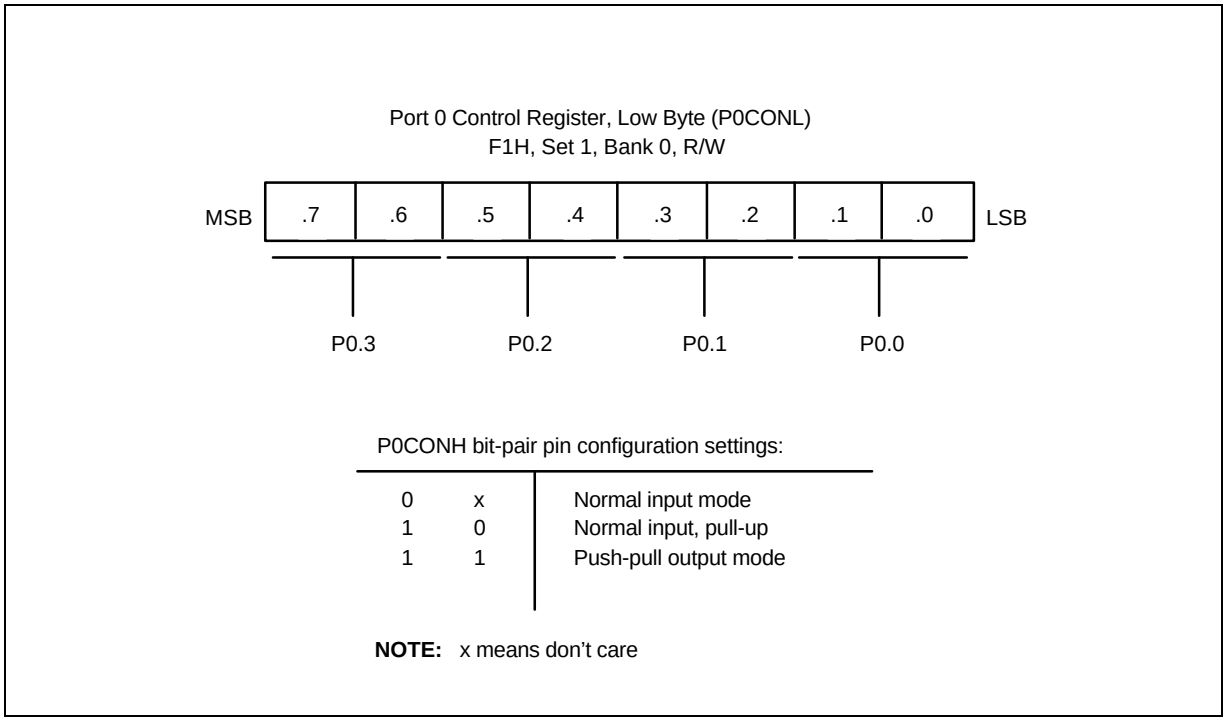


Figure 9-3. Port 0 Low-byte Control Register (P0CONL)

PORT 1

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

Port 1 is a 8-bit I/O port with individually configurable pins and you can use port 1 pins for general I/O or configure the alternative function (P1.6/P1.7). Please note that whenever you use port 1 pins for functions other than general I/O, you must still program the P1CONH register to specify which bits are inputs and which are outputs.

Port 1 Control Register, High Byte (P1CONH)
F2H, Set 1, Bank 0, R/W

P1CONH bit-pair pin configuration settings: (P1,5/P1.4)

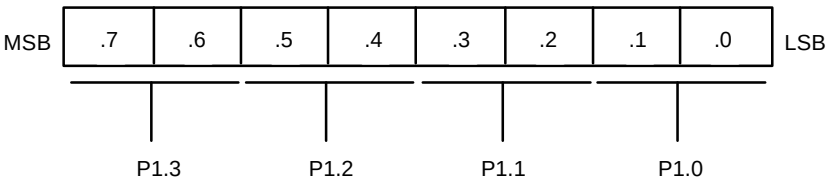
0	x	Normal i nput mode
1	0	Normal input, pull-up
1	1	Push-pull output mode

P1CONH bit-pair pin configuration settings: (P1,7/P1.6)

0	0	Normal i nput mode
0	1	Alternative function select; SO enable
1	0	Normal input, pull-up
1	1	Push-pull output mode

Figure 9-4. Port 1 High-byte Control Register (P1CONH)

Port 1 Control Register, Low Byte (P1CONL)
F3H, Set 1, Bank 0, R/W



P1CONL bit-pair pin configuration settings:

0	x	Normal i nput mode
1	0	Normal input, pull-up
1	1	Push-pull output mode

('x' means don't care)

Figure 9-5. Port 1 Low-byte Control Register (P1CONL)

PORT 2

Port 2 can serve as a general-purpose 8-bit I/O port or its pins can be configured individually as external interrupt inputs. Port 2 is accessed directly by writing or reading the port 2 data register, P2 (E2H, set 1).

Port 2 Control Registers (P2CONL, P2CONH)

The direction of each port pin is configured by bit-pair settings in two control registers: P2CONL (low byte, F5H) and P2CONH (high byte, F4H).

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

Port 2 Interrupt Enable and Pending Registers (P2INT, P2PND)

To process external interrupts at the port 2 pins, two additional control registers are provided: the port 2 interrupt enable register P2INT (E7H) and the port 2 interrupt pending register P2PND (FBH). Before changing P2INT, you must use DI instruction to prevent mal-operation.

The port 2 interrupt pending register P2PND lets you check for interrupt pending conditions and clear the pending condition when the interrupt request has been serviced. Incoming interrupt requests are detected by polling the P2PND bit values.

When the interrupt enable bit of any port 2 pin is "1", a rising or falling signal edge at that pin will generate an interrupt request. The corresponding P2PND bit is then automatically set to "1" and the IRQ level goes high to signal the CPU that an interrupt request is waiting.

When the CPU acknowledges the interrupt request, application software must clear the pending condition by writing a "0" to the corresponding P2PND bit, only when DI state is activated.

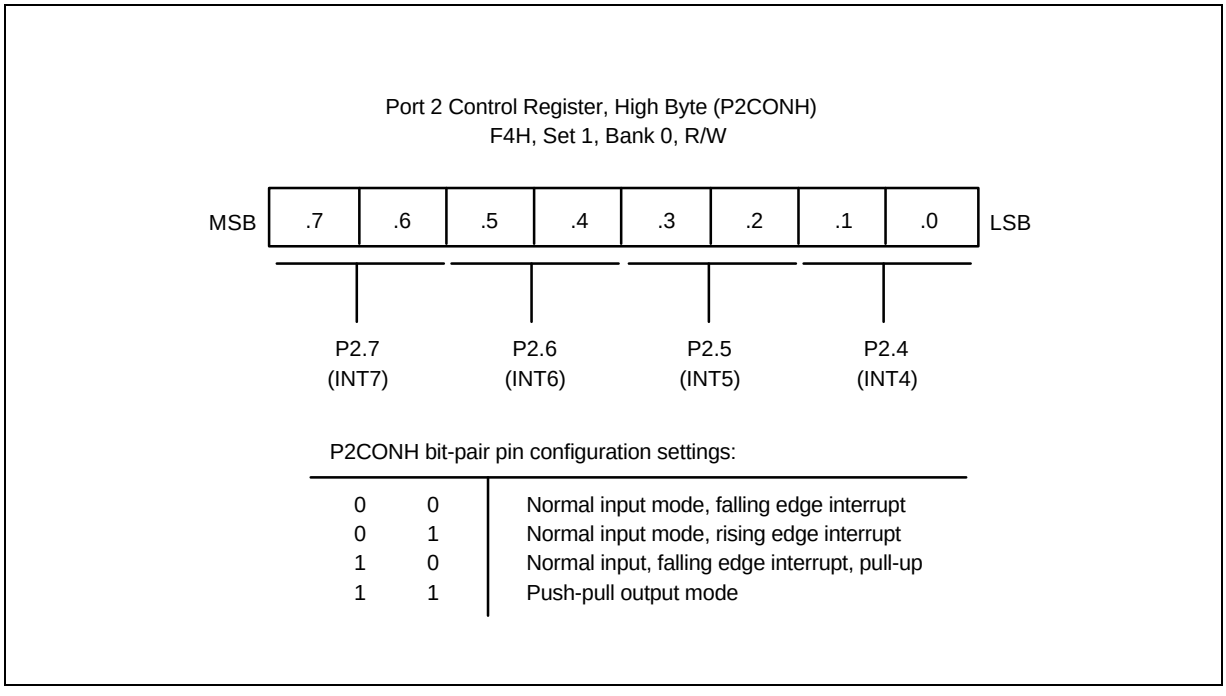


Figure 9-6. Port 2 High-Byte Control Register (P2CONH)

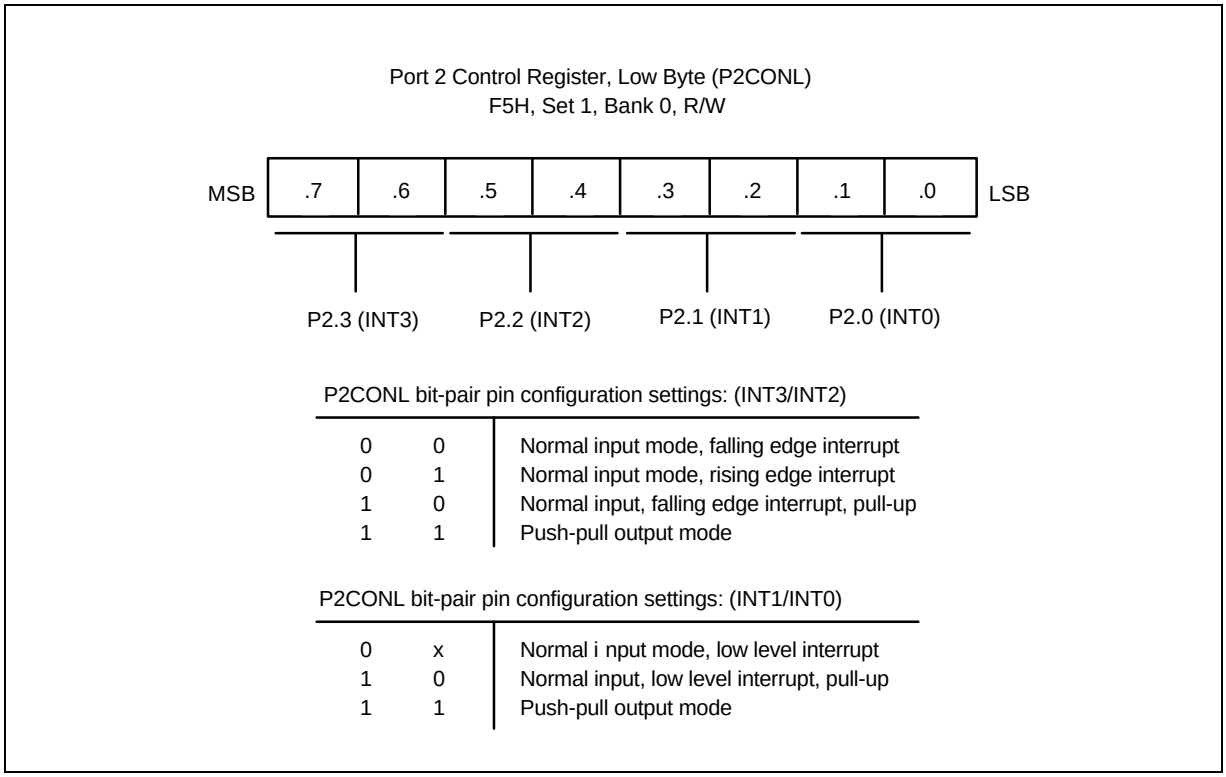


Figure 9-7. Port 2 Low-Byte Control Register (P2CONL)

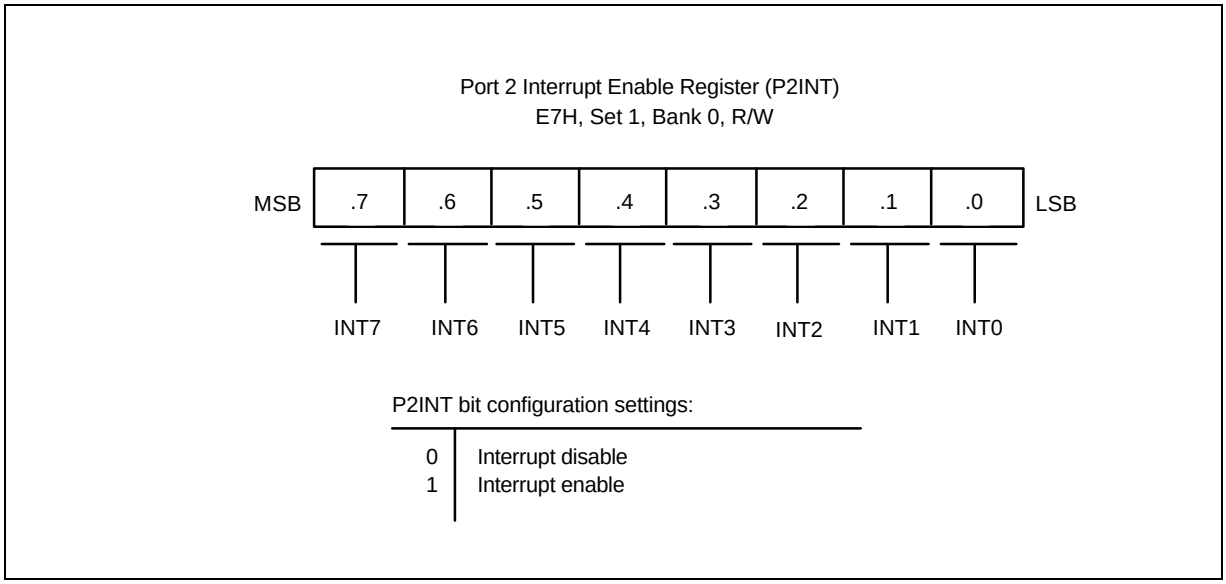


Figure 9-8. Port 2 Interrupt Control Register (P2INT)

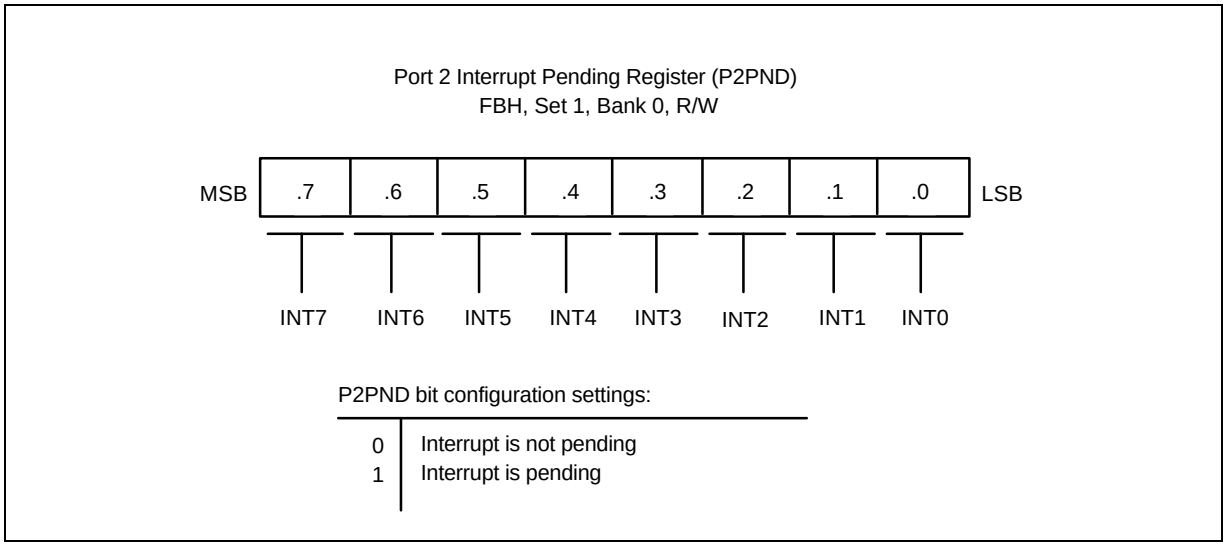


Figure 9-9. Port 2 Interrupt Pending Register (P2PND)

PORT 3

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

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

— In input mode, pins P3.1 and P3.0 can be used as clock inputs to timer/counters C and D. The corresponding share pin names are TCCK and TDCK, respectively.

— In input mode, pins P3.3 and P3.2 can be used as capture inputs to timer/counters C and D. The corresponding share pin names are TCCAP and TDCAP, respectively.

— In alternative output mode, pins P3.4 and P3.5 can be used as timer/counters C and D outputs or PWM outputs, respectively. The corresponding share pin names are TCPWM/TCOUT and TDPWM/TDOUT.

— In alternative output mode, pins P3.6 and P3.7 can be used as PWM outputs. The corresponding share pin names are PWM0 and PWM1.

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

Port 3 Control Registers

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

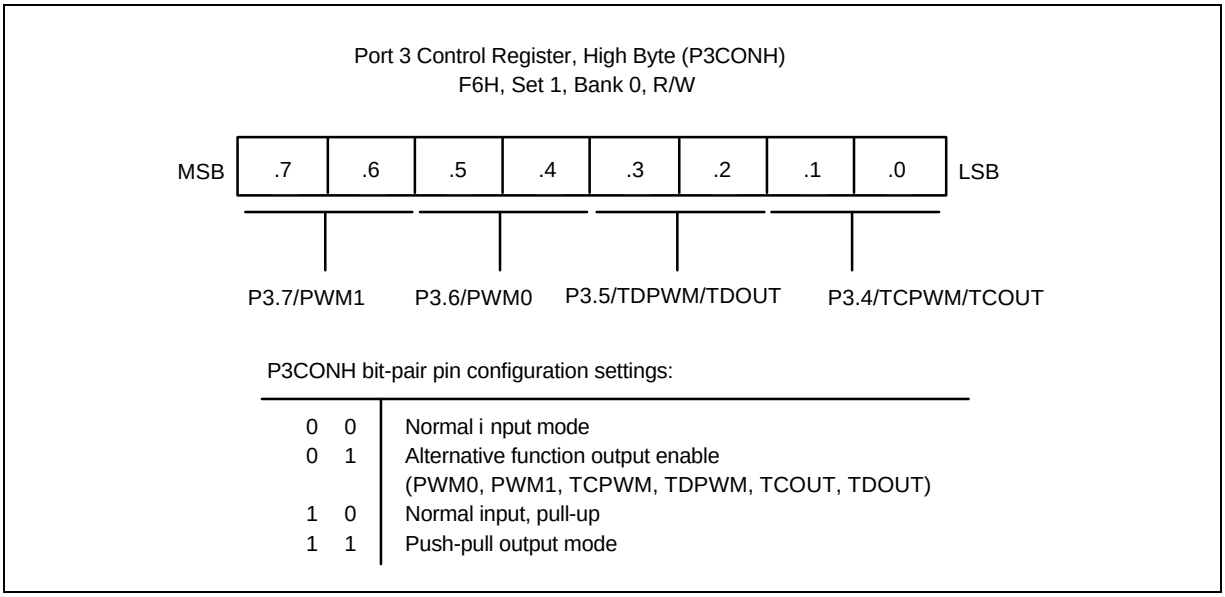


Figure 9-10. Port 3 High-Byte Control Register (P3CONH)

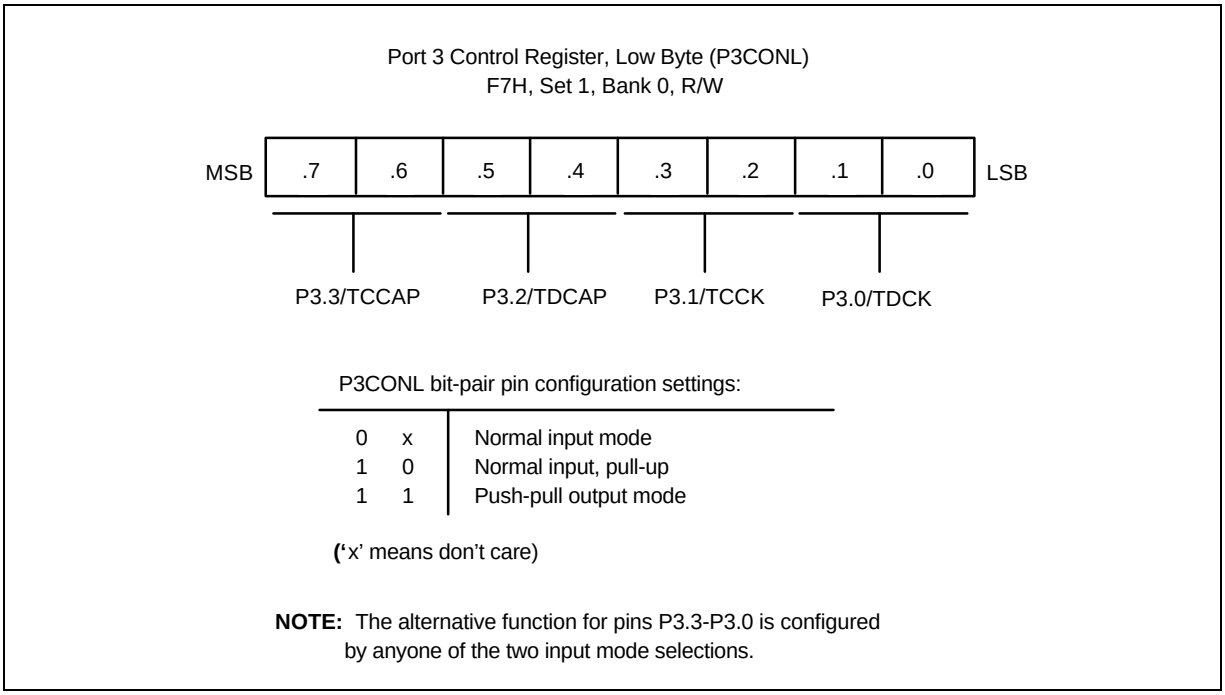


Figure 9-11. Port 3 Low-Byte Control Register (P3CONL)

PORT 4

Port 4 is an 8-bit port with individually configurable pins. Port 4 is accessed directly by writing or reading the port 3 data register, P4 (E4, set1).

You can use port 3 pins for general I/O, or configure the following alternative function. In embedded chip selection output mode, pins P4.4 (CS0)–P4.7 (CS3) can be used as chip selection outputs to get rid of external decoding gates.

When the alternative function (CSx) is used, CSDATAx (FC–FF, bank1) should be set with appropriate value.

Port 4 Control Registers

Port 4 has two 8-bit control registers: P4CONH for the high byte pins, P4.4–P4.7, and P4CONL for the low byte pins, P4.0–P4.3. A reset operation clears the P4CONH and P4CONL registers to 00H.

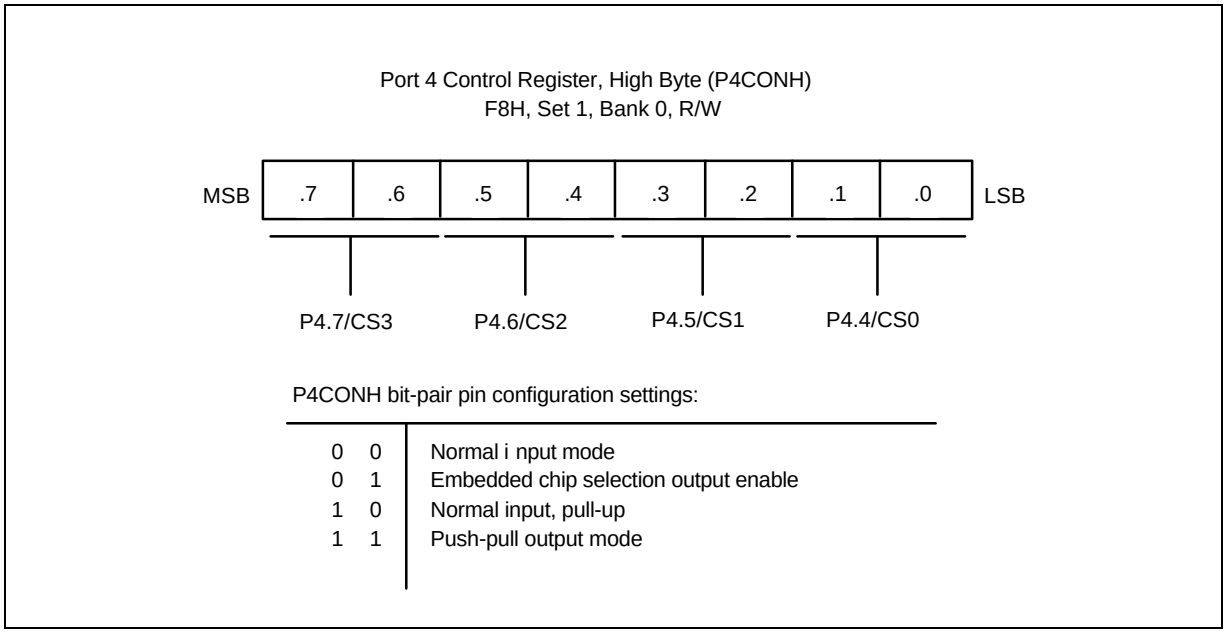


Figure 9-12. Port 4 High-Byte Control Register (P4CONH)

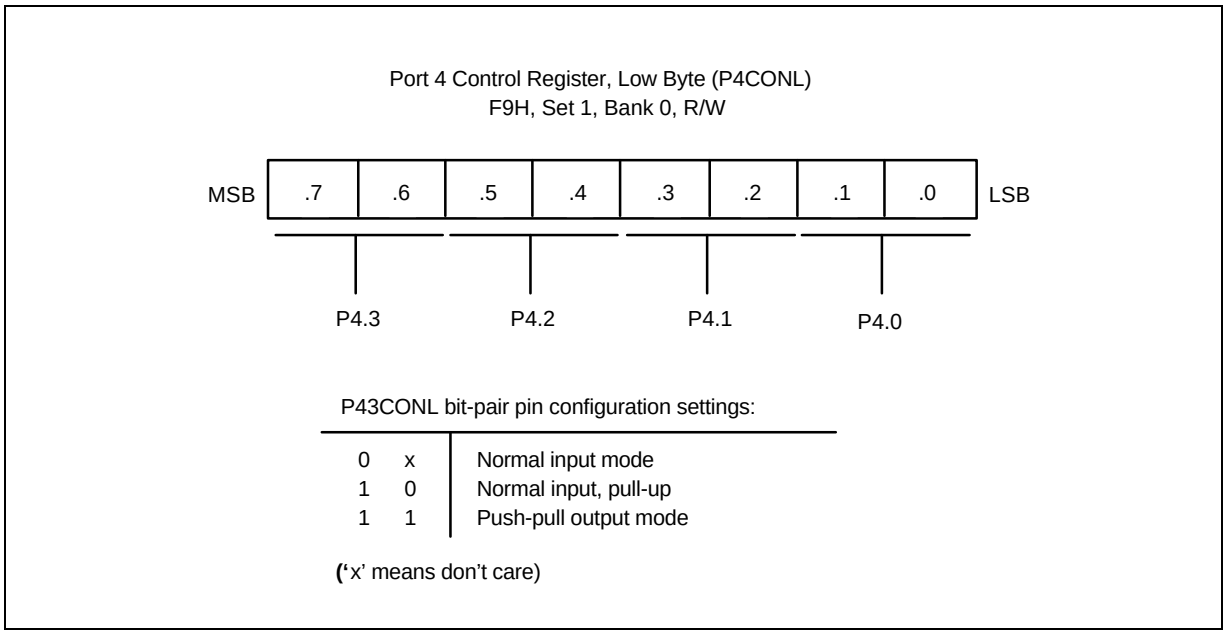


Figure 9-13. Port 4 Low-Byte Control Register (P4CONL)

PORT 5

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

You can use port 5 pins for general I/O or configure the following alternative function.

— In input mode, pin P5.0 can be used as WAIT signal input.

Port 5 control register

Just low nibble is configured because only two pins are assigned at port 5.

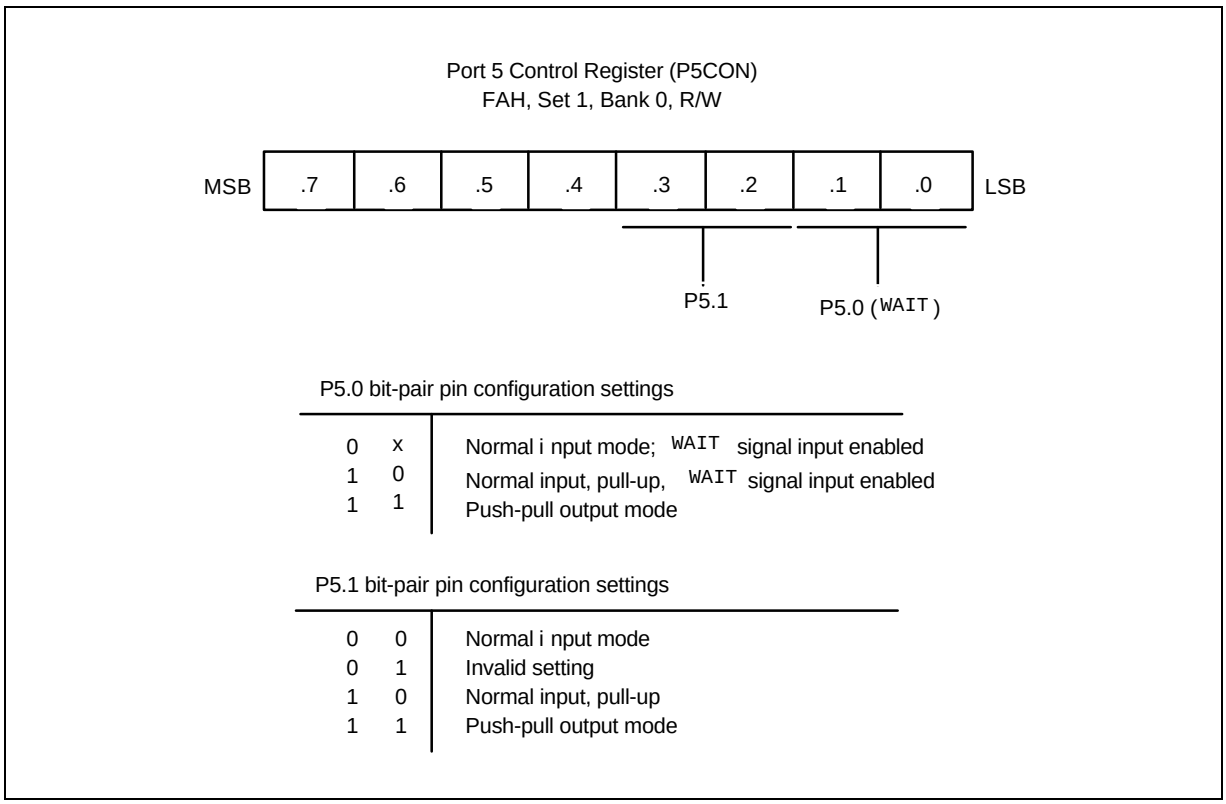


Figure 9-14. Port 5 Control Register (P5CON)

☞ PROGRAMMING TIP — Configuring KS88C4504/P4504 Port Pins to Specification

This example shows how to configure KS88C4504 /P4504 ports to sample specifications. Ports 0 through 5 to be configured as follows:

— Set P0.0–P0.3 to push-pull output mode
— Set P0.4–P0.7 to ADC input mode
— Set Port 1 to normal input mode with pull-up resistor
— Set P2.0–P2.1 to input mode with low level interrupts
— Set P2.2–P2.3 to input mode with falling-edge interrupts
— Set P2.4–P2.7 to input mode with rising-edge interrupts
— Set P3.0–P3.3 to push-pull output mode
— Set P3.4 to timer C output mode
— Set P3.5 to timer D PWM output mode
— Set P3.6–P3.7 to PWM output mode
— Set P4.0–P4.5 to normal input mode with pull-up resistors
— Set P4.6– P4.7 to embedded chip selection output mode
— Set P5.0–P5.1 to push-pull output mode

```
•
•
•
LD      P0CONL,#0FFH    ; P0.0–P0.3 ← push-pull output mode
LD      P0CONH,#55H     ; P0.4–P0.7 ← ADC input mode

LD      P1CONL,#0AAH    ; P1 ← normal input mode with pull-up resistor
LD      P1CONH,#0AAH

LD      P2CONL,#00H     ; P2.0–P2.1 ← input mode with low level interrupts
                        ; P2.2–P2.3 ← input mode with falling edge interrupts
LD      P2CONH,#55H     ; P2.4–P2.7 ← input mode with rising edge interrupts
LD      P2INT,#0FFH     ; All external interrupts enable
LD      P2PND,#00H      ; External interrupts pending clear

LD      P3CONL,#0FFH    ; P3.0–P3.3 ← push-pull output mode
LD      P3CONH,#55H     ; P3.4–P3.7 ← alternative output mode

LD      P4CONL,#0AAH    ; P4.0–P4.3 ← normal input mode with pull-up resistors
LD      P4CONH,#05AH    ; P4.4–P4.5 ← normal input mode with pull-up resistors
                        ; P4.6–P4.7 ← embedded chip selection output mode

LD      P5CON,#0FH      ; P5.0–P5.1 ← push-pull output mode

•
•
•
```

10 8-BIT TIMERS

MODULE OVERVIEW

KS88C4504/P4504 have three default timers: an 8-bit basic timer and two 8-bit general-purpose timer/counters.

BASIC TIMER (BT)

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

— As a watchdog timer to provide an automatic reset mechanism in the event of a system malfunction, or

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

The functional components of the basic timer block are:

— Clock frequency divider (f_{OSC} divided by 4096, 1024, or 128) with multiplexer

— 8-bit basic timer counter, BTCNT (set 1, bank 0, FDH, read-only)

— Basic timer control register, BTCON (set 1, D3H, read/write)

BASIC TIMER CONTROL REGISTER (BTCON)

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

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

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

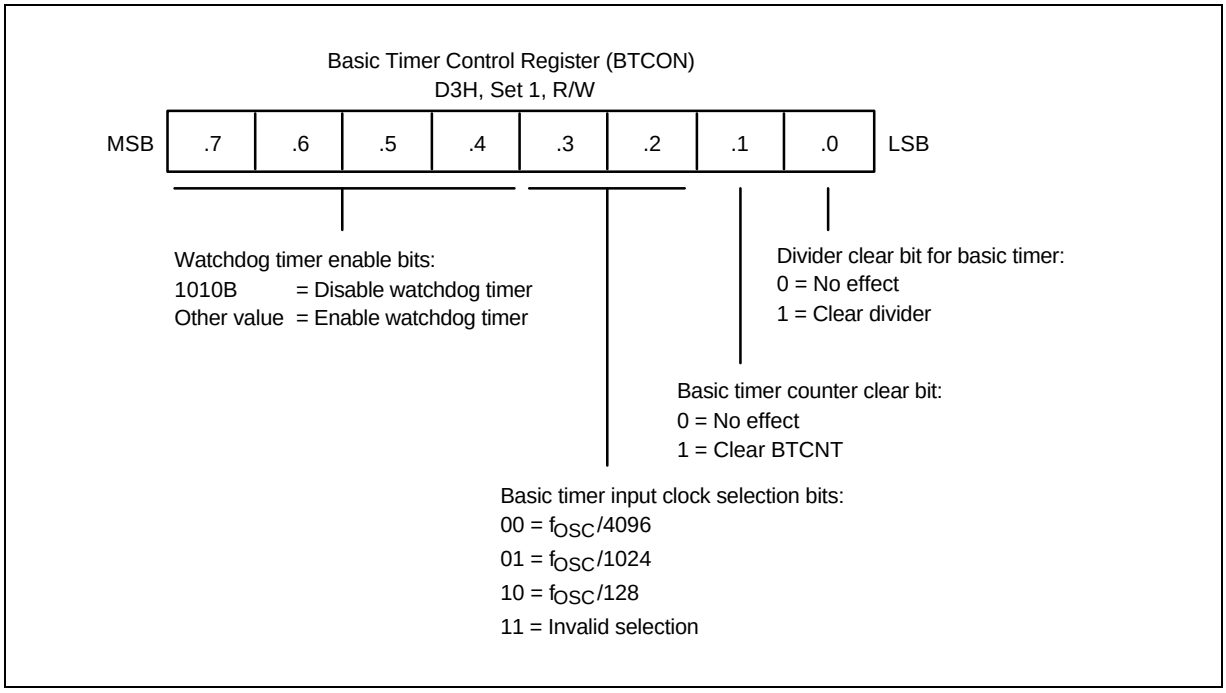


Figure 10-1. Basic Timer Control Register (BTCON)

BASIC TIMER FUNCTION DESCRIPTION

Watchdog Timer Function

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

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

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

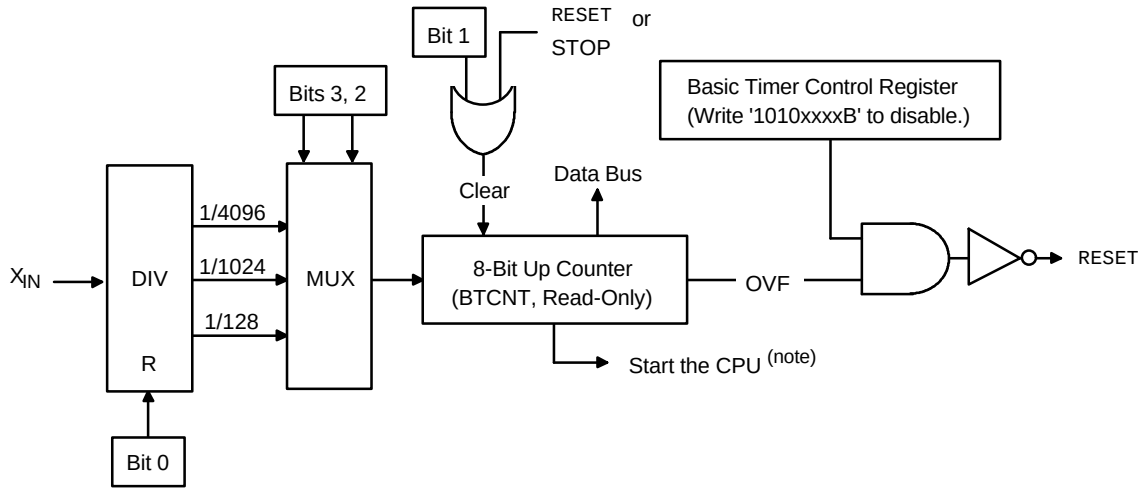
Oscillation Stabilization Interval Timer Function

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

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

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

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



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

Figure 10-2. Basic Timer Block Diagram

8-BIT TIMERS

OVERVIEW

The KS88C4504/P4504 has two 8-bit timers. The 8-bit timers A/B are 8-bit general-purpose timers. Timers A/B have the interval timer mode by using the appropriate TACON or TBCON settings.

Timers A/B have the following functional components:

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

— 8-bit counter (TACNT/TBCNT), 8-bit comparator, and 8-bit reference data register (TADATA/TBDATA)

— Timer A match interrupt (IRQ3, vector C2H) generation

— Timer A control register, TACON (set 1, D0H, read/write)

— Timer B match interrupt (IRQ4, vector C4H) generation

— Timer B control register, TBCON (set 1, D1H, read/write)

FUNCTION DESCRIPTION

Interval Timer Function

Timers A/B can generate an interrupt (TAINT/TBINT). TAINT/TBINT belong to the interrupt levels IRQ3/IRQ4 and they are assigned separate vector addresses, C2H/C4H.

The TAINT/TBINT pending conditions are automatically cleared by hardware when they have been serviced. Even though TAINT/TBINT are disabled, the application's service routine can detect a pending condition of TAINT/TBINT by software and execute the sub-routine. In this case, the TAINT pending bit must be cleared by the application sub-routine by writing a "0" to the TACON.0 pending bit.

A match signal is generated when the counter value is identical to the value written to the TA/TB reference data registers, TADATA/TBDATA. The match signal generates a timer A match interrupt and clears the counter.

For example, if you write the value 10H to TADATA and 0FH to TACON, the counter will increase until it reaches 10H. At this point, the TA interrupt request is generated, the counter value is reset, and counting resumes.

Timers A/B Control Registers (TACON/TBCON)

You use the timers A/B control register, TACON/TBCON, to

— Enable the timer A/B operation (interval timer)

— Select the timer A/B input clock frequency

— Clear the timer A/B counters, TACNT and TBCNT

— Enable the timer A/B interrupt

— Clear the timer A/B interrupt pending conditions

TACON is located in set 1, at address D0H, and is read/write addressable using Register addressing mode.

A reset clears TACON to '00H'. This sets timer A to disable interval timer mode, selects an input clock frequency of fOSC/1024, and disables a timer A interrupt. You can clear the timer A counter at any time during normal operation by writing a "1" to TACON.3.

To enable a timer A interrupt (IRQ3, vector C2H), you must write a "1" to TACON.2 and TACON.1. To generate the exact time interval, you should write a "0" to TACON.3, which clears the counter and interrupt pending bit. To detect an interrupt pending condition when TAINT is disabled, the application program polls the pending bit, TACON.0. When a "1" is detected, a timer 0 interrupt is pending. When the TAINT sub-routine is serviced, the pending condition must be cleared by software by writing a "0" to the timer A interrupt pending bit, TACON.0.

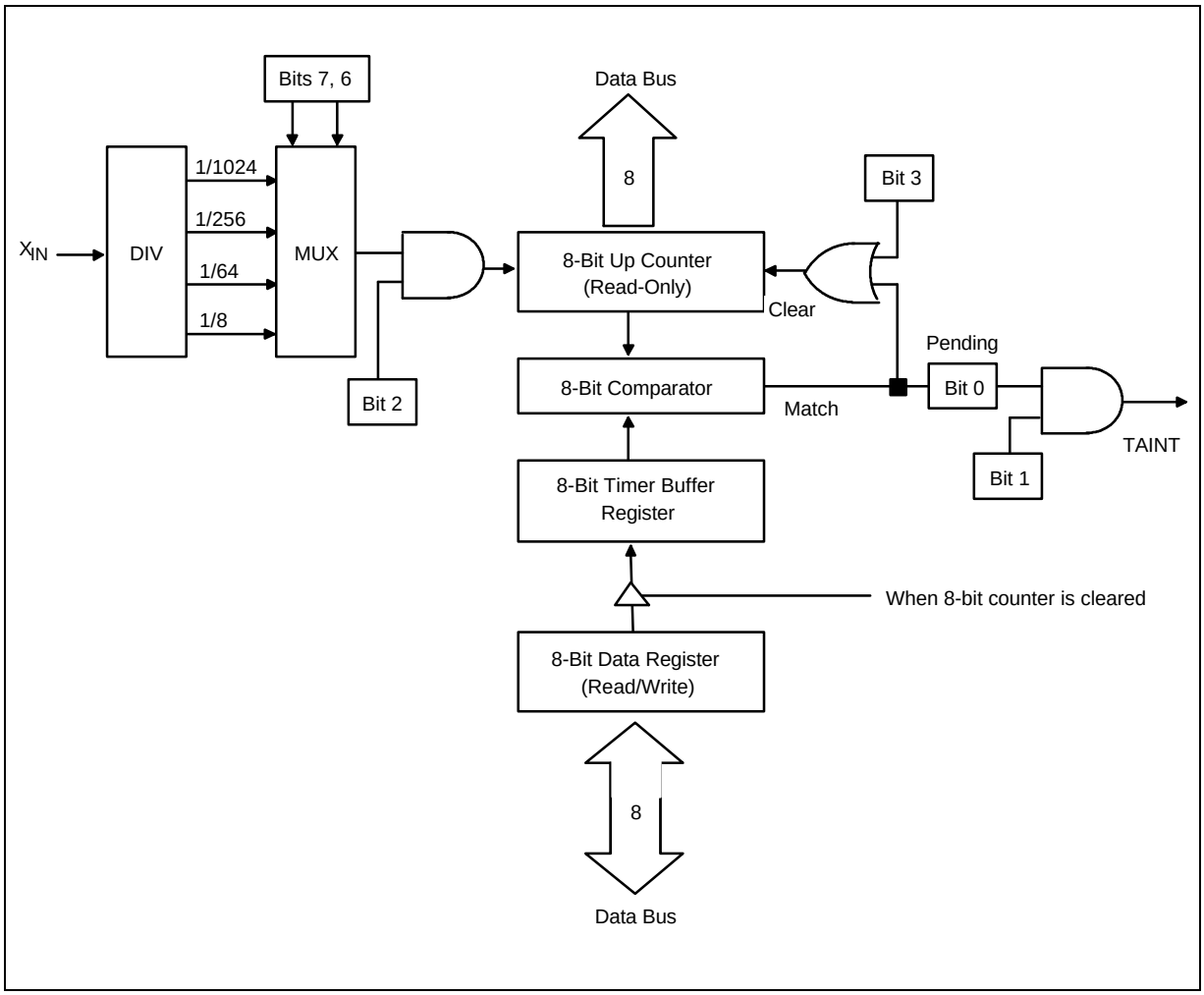


Figure 10-3. Timer A/B Function Block Diagram

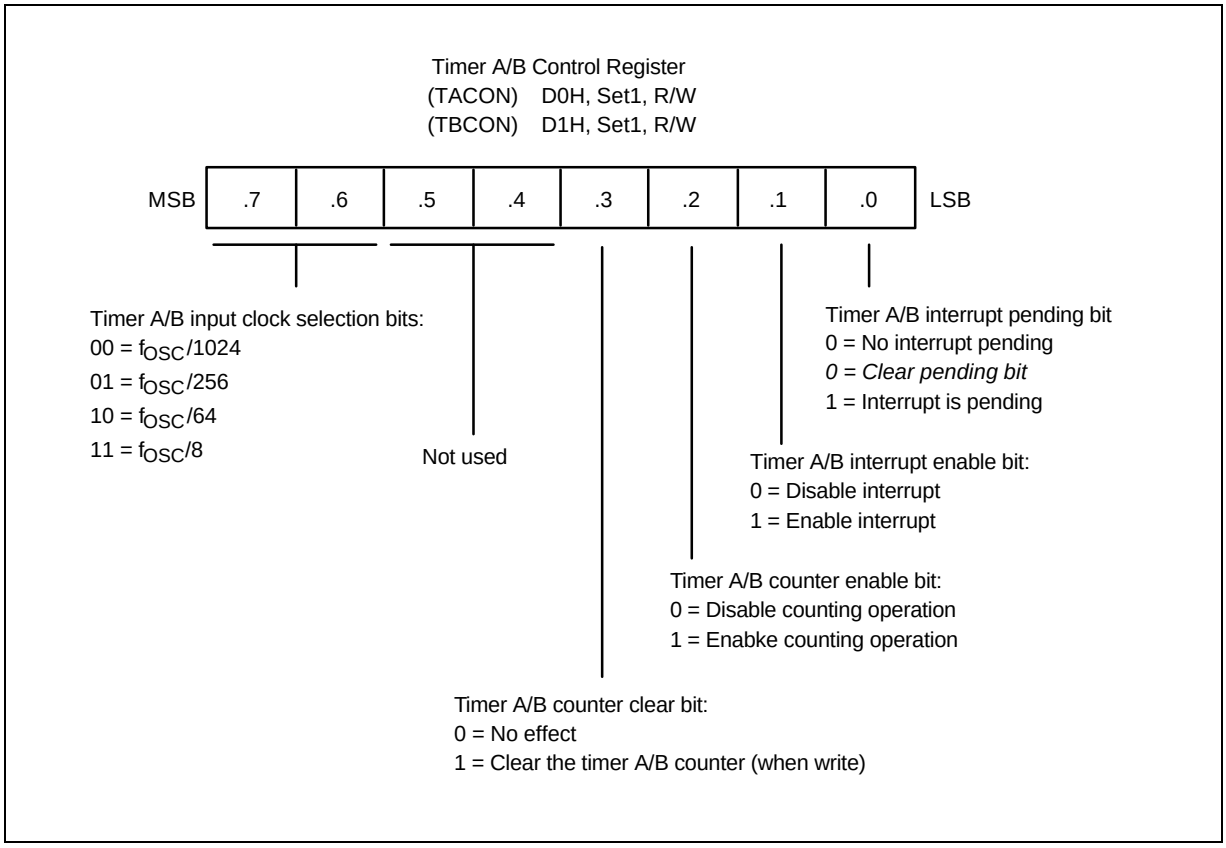


Figure 10-3. Timer A Control Register (TACON/TBCON)

☞ 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
        SB0                     ;  Select bank 0
        LD      BTCON,#0A0H     ;  Disable the watchdog timer
        LD      CLKCON,#98H     ;  Non-divided clock
        CLR     SYM             ;  Disable global and fast interrupts
        CLR     SPL             ;  Stack pointer low byte ← "0"
                                ;  Stack area starts at 0FFH
        •
        •
        •
        SRP     #0C0H           ;  Set register pointer ← 0C0H
        EI                      ;  Enable interrupts
        •
        •
        •
MAIN    LD      BTCON,#52H      ;  Enable the watchdog timer
                                ;  Basic timer clock: fOSC/4096
                                ;  Clear basic timer counter
        NOP
        NOP
        •
        •
        •
        OR      BTCON,#00000010B ;  Clear basic timer
        •
        •
        •
        JP      T,MAIN
```

☞ PROGRAMMING TIP — Configuring Timer A and Timer B

Disables timer B, sets the oscillation frequency of the timer clock, and determines the execution sequence after a timer A interrupt occurs. The program parameters are:

— The timer A interval is set to approximately 2 milliseconds

— Timer B is disabled

— Oscillation frequency is 12 MHz

— 90H ← 90H + 91H + 92H + 93H + 94H is executed after a timer A interrupt

```
                .VECTOR     0C2H,TAINT          ;  Timer A interrupt vector
;
                .ORG        0100H               ;  Reset address
                JP          T,START
                •
                •
                .ORG        0200H
START:          DI
                •
                •
                •
                LD          TACON,#00001110B    ;
                                                ;  fOSC/1024 is selected for timer A
                                                ;  Enable timer A interrupt, reset timer A pending register
                                                ;  Enable counting operation
                LD          TBCON,#00H          ;  Disable Timer B
                LD          TADATA,#17H         ;  TADATA ← 23
                                                ;  12 MHz/(1024 × 23) = 0.5 kHz (= 2 ms)
                                                ;
                EI                              ;  Enable interrupts
                •
                •
                •
TAINT:          PUSH        RP0                 ;  Save RP0 to stack
                SRP0        #90H                ;  RP0 ← 90H
                ADD         R0,R1               ;  R0 ← R0 + R1
                ADC         R0,R2               ;  R0 ← R0 + R2
                ADC         R0,R3               ;  R0 ← R0 + R3
                ADC         R0,R4               ;  R0 ← R0 + R4
                                                ;  (R0 ← R0 + R1 + R2 + R3 + R4)
                AND         TACON,#0FEH         ;  Reset timer A pending register (omissible)
                                                ;
                POP         RP0                 ;  Restore register pointer 0 value
                IRET                            ;  Return from interrupt service routine
                •
                •
                •
```

11 16-BIT TIMERS

OVERVIEW

The KS88C4504/P4504 has two 16-bit timer /counters. The 16-bit timers C/D are 16-bit general-purpose timer/counters. Timer C/D have three operating modes, one of which you select using the appropriate TCCON/TDCON settings:

— Interval timer mode(Toggle output at TCOUT/TDOUT pin)

— Capture input mode with a rising or falling edge trigger at the TCCAP/TDCAP pin

— PWM mode (TCPWM/TDPWM); PWM output shares their output port with TCOUT/TDOUT pin

Timers C/D have the following functional components:

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

— External clock input pin (TCCK/TDCK)

— 16-bit counter (TCCNTH/L,TDCNTH/L), 16-bit comparator, and 16-bit reference data register (TCDATAH/L, TDDATAH/L)

— I/O pins for capture input (TCCAP/TDCAP), or PWM or match output(TCPWM/TDPWM, TCOUT/TDOUT)

Timer C/D overflow interrupt and match/capture interrupt generation

Timer C: overflow interrupt (IRQ0,vector B8h) / match interrupt (IRQ0,vector BAh)

Timer D: overflow interrupt (IRQ1,vector BCh) / match interrupt (IRQ1,vector BEh)

— Interrupt pending register: TCPND (EEHh, Set 1, Bank 0), TDPND (EFH, Set 1, Bank 0)

— Timer C/D control register: TCCON (set 1, Bank0, E8H, read/write), TDCON (set 1, Bank0, E9H, read/write)

FUNCTION DESCRIPTION

TIMER C/D INTERRUPTS

The timer C/D module can generate two interrupts: overflow interrupts (TCOVF/TDOVF), and match/capture interrupt (TCINT/TDINT). TCOVF is in interrupt level IRQ0, vector B8H. TCINT also belongs to the interrupt level, IRQ0, but is assigned a separate vector address, BAH.
TDOVF is in interrupt level IRQ1, vector BCH. TDINT also belongs to the interrupt level, IRQ1, but is assigned the separate vector address, BEH.

A timer C/D overflow interrupt pending condition is automatically cleared by hardware when it is serviced. Timers C/D match/capture interrupt register, TCPND/TDPND are also cleared by hardware when they are serviced.

INTERVAL TIMER FUNCTION

The timer C/D module can generate an timer C/D match interrupt (TCINT/TDINT). TCINT belongs to the interrupt level, IRQ0, and is assigned the separate vector address, BAH. and TDINT belongs to the interrupt level, IRQ1, and is assigned the separate vector address, BEH.
When a timer C/D match interrupt occurs and is serviced by the CPU, the pending register is cleared automatically by hardware.

In interval timer mode, a match signal is generated and TCOUT/TDOUT is toggled when the counter value is identical to the value written to the TC/TD reference data register, TCDATAH/L and TDDATAH/L. The match signal generates a timer 1 match interrupt (TCINT, vector BAh,TDINT, vector BEh) and clears the counter.

For example, if you write the value 0010H to TCDATAH/L and 06H to TCCON, the counter will increase until it reaches 0010H. At this point, the TC interrupt request is generated, the counter value is reset, and counting resumes.

PULSE WIDTH MODULATION MODE

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

Although you can use the match interrupt , this interrupt is not typically used in PWM-type applications. Instead, the pulse at the TCPWM/TDPWM 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 tCLK x 65536 .

CAPTURE MODE

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

Timer C/D also gives you a capture input source: the signal edge at the TCCAP/TDCAP pin. You can select the capture input by setting the value of the timer C/D capture input selection bit in the port 3 control register, P3CONL.

When the corresponding bit of P3CONL is 0, the TCCAP/TDCAP input or normal input is selected.
When the corresponding bit of P3CONL is set to 1, normal output is selected. And Capture function will not work correctly. Both kinds of timer C/D interrupts can be used in capture mode: the timer C/D overflow interrupt is generated whenever a counter overflow occurs; the timer C/D match/capture interrupt is generated whenever the counter value is loaded into the TC/TD data register.

By reading the captured data value in TCDATAH/L, and assuming a specific value for the 16-timer clock frequency, you can calculate the pulse width (duration) of the signal that is being input to the TCCAP pin.

16-BIT TIMER CONTROL REGISTER (TCCON/TDCON)

You can use the timer 1 control register, TCCON/TDCON, to

— Select the timer C/D operating mode (interval timer, capture mode, or PWM mode)

— Select the timer C/D input clock frequency

— Clear the timer C/D counters, TCCNTH/L and TDCNTH/L

— Enable the timer C/D overflow interrupt or timer C/D match/capture interrupt

— Clear the timer C/D match/capture interrupt pending conditions

TCCON is located in set 1, bank 0, at address E8h, and TDCON is located in set 1 bank 0, at address E9h, are both read/write addressable using Register addressing mode.

A reset clears TCCON/TDCON to '00H'. This sets timer C/D to normal interval timer mode, selects an input clock frequency of fOSC/1024, and disables all timer C/D interrupts. To disable the counter operation, please set TxCON.7–.5 to 111B. You can clear the timer C/D counter at any time during normal operation by writing a "1" to TxCON.2.

The timer C/D overflow interrupt (TCOVF/TDOVF) is in interrupt level IRQ0/IRQ1 and has the vector address B8H/BCH. When a timer C/D overflow interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware.

To enable the timer C/D match/capture interrupt , you must write TCCON/TDCON.1 to "1". To generate the exact time interval, you should write TxCON.2 which clears counter and interrupt pending bit. To detect a match/capture or overflow interrupt pending condition when TCINT/TDINT or TCOVF/TDOVF is disabled, the application program should poll the register bit. When a "1" is detected, a timer C/D match/capture or overflow interrupt is pending. When its sub-routine is serviced, the pending condition must be cleared by software by writing a "0" to the interrupt pending bit.

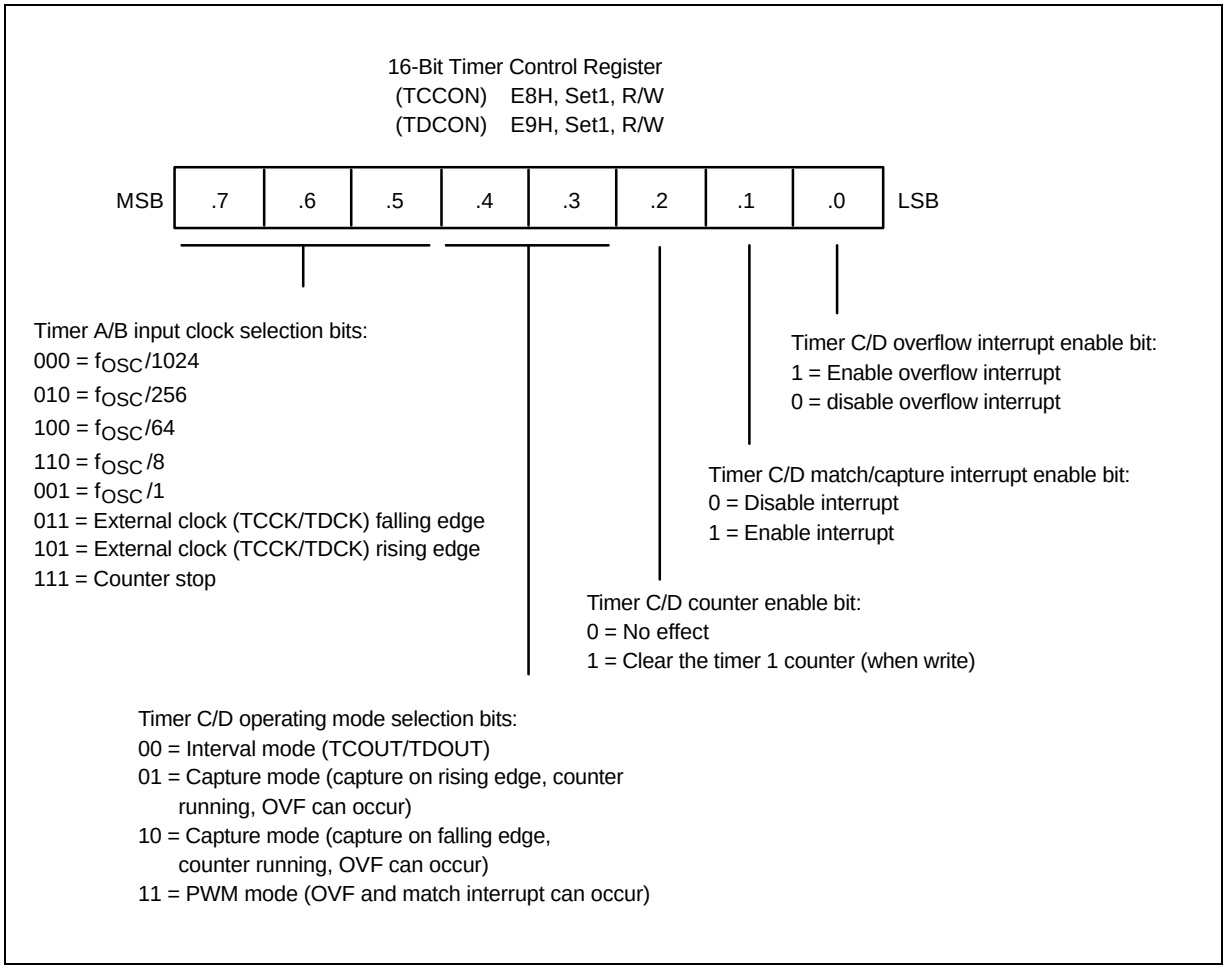


Figure 11-1. 16-BIT Timer Control Register (TCCON/TDCON)

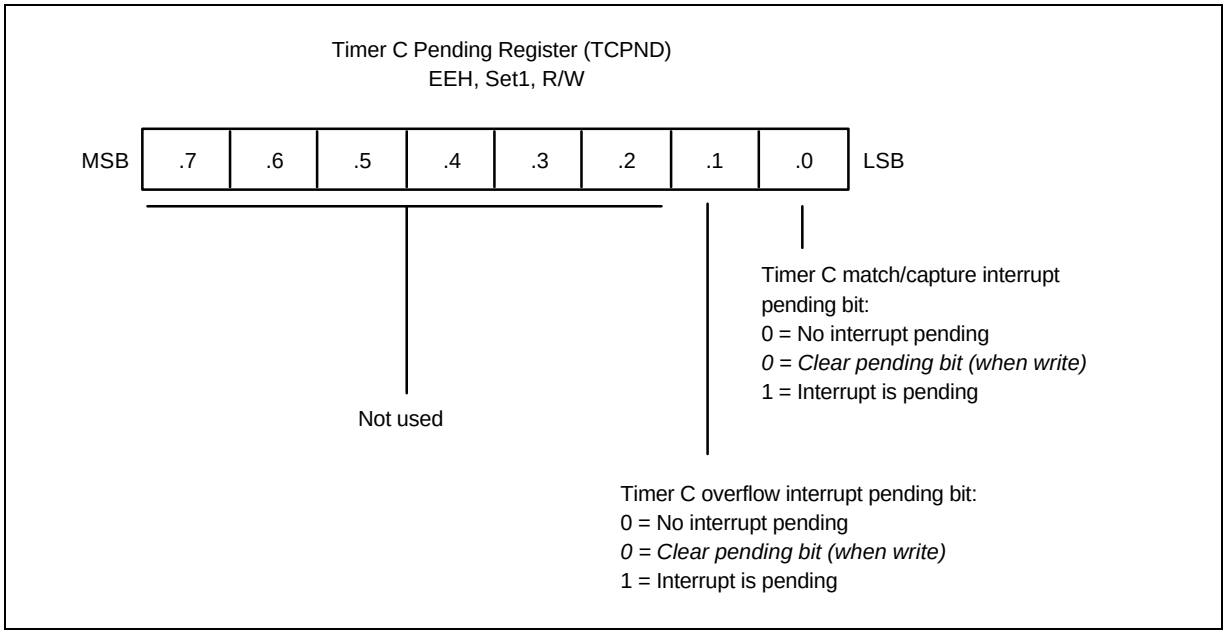


Figure 11-2. Timer C pending Register (TCPND)

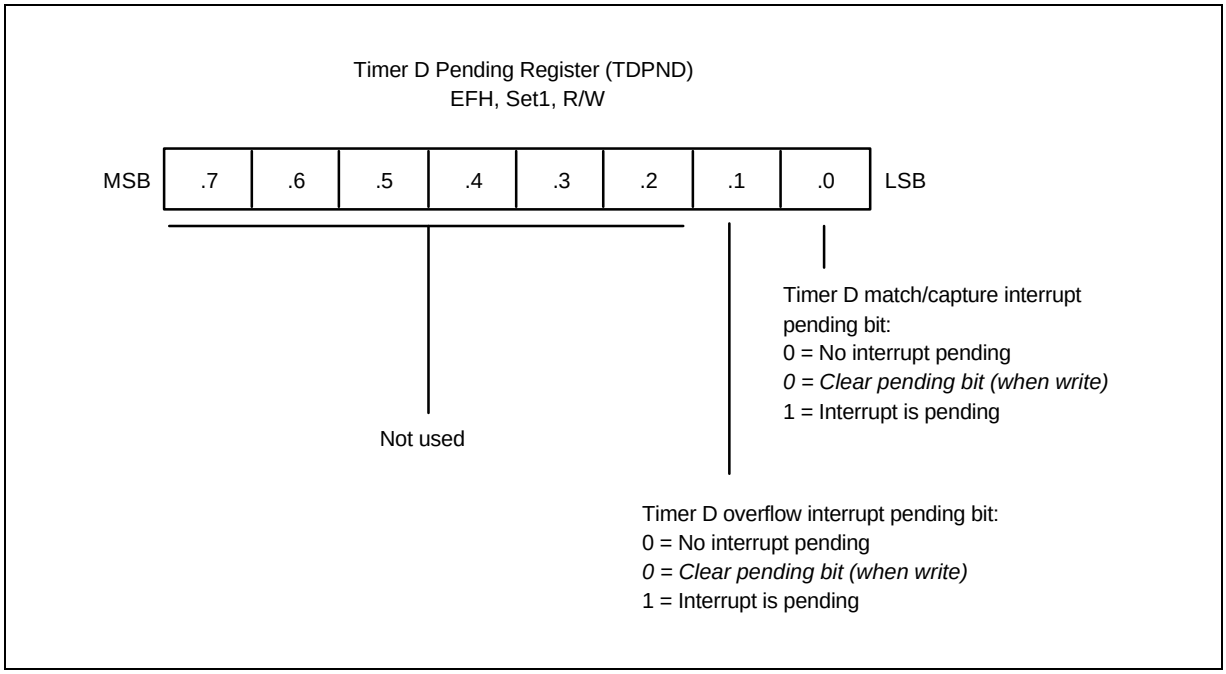


Figure 11-3. Timer D pending Register (TDPND)

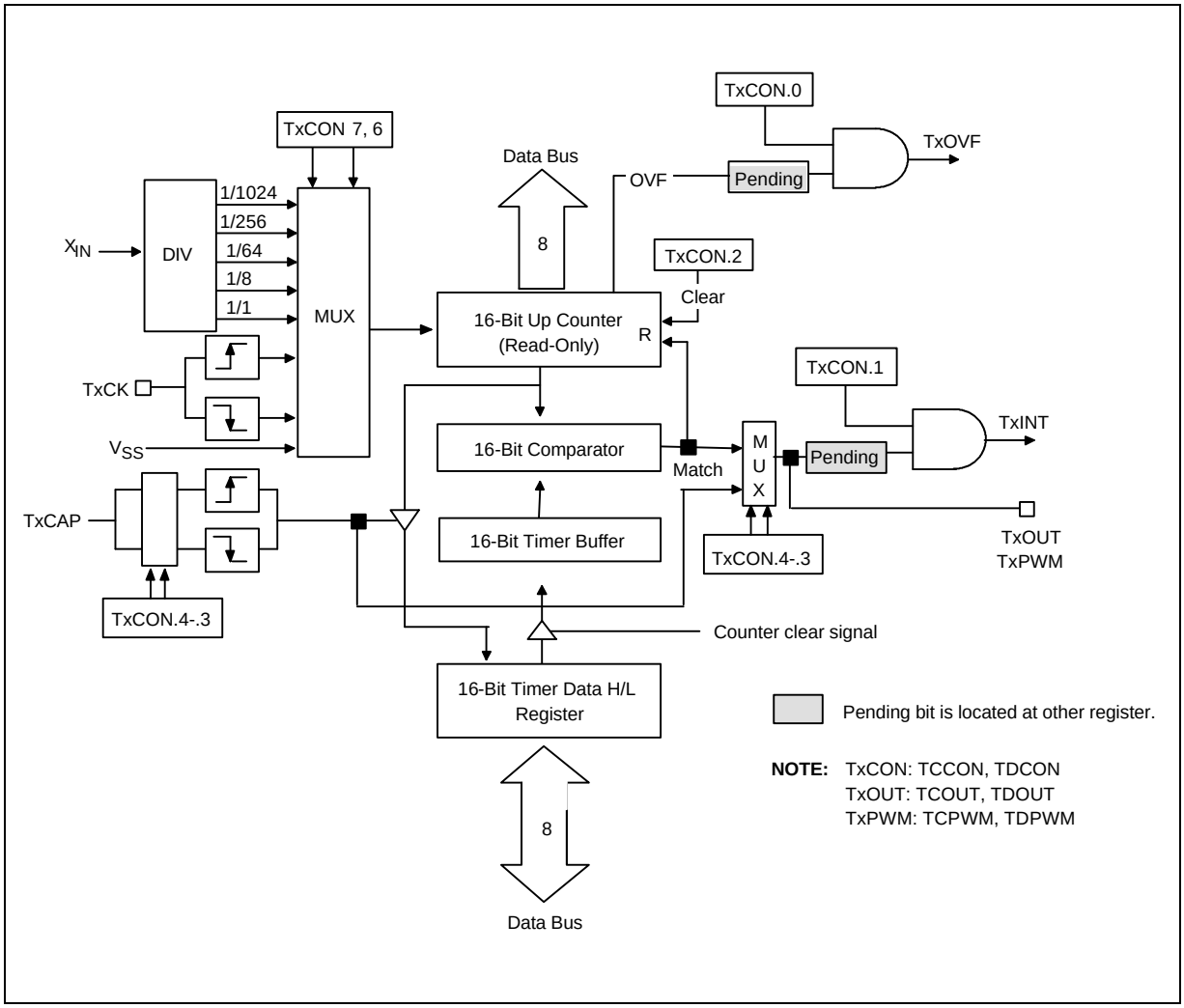


Figure 11-4. 16-Bit Timer Functional Block Diagram (Timer C/D)

☞ PROGRAMMING TIP — 16-bit Timers

The timer C is used in internal mode and the timer D in capture mode and save the counting value to capture registers.

— Timer C interval depends on the speed of external clock (TCCK, P3.1) and TCDATA.

— Timer D captures the falling edge at TDCAP input pin (P3.2).

```
            .ORG    0100H
            JP      t,START
;
            .VECTOR 0BAH,TCINT
            .VECTOR 0BEH,TDCAP
            •
            •
            •
            .ORG    0200H
START:      DI
            •
            •
            •
            LD      TCCON,#01100110B    ;  External clock selection
                                        ;  Interval mode
                                        ;  Clear count and enable interrupt
            LD      TDCON,#00010110B    ;  fOSC/1024 clock source
                                        ;  Capture mode (on falling edge)
                                        ;  Clear count and enable interrupt
            LD      TCPND,#00H          ;  TC interrupt pending clear
            LD      TDPND,#00H          ;  TD interrupt pending clear
            AND     P3CONL,#11000011B   ;  Enable TCCK input and TDCAP input
            •
            •
            •
            EI
            •
            •
            •
TCINT:
            •
            •
            •
            AND     TCPND,#02H          ;  TC interrupt pending clear
            IRET
;
TDCAP:      PUSH    RP0
            SRP0    #CAP_REG
            LD      R0,TCDATAH          ;  Save the counting value to CAP_REG (16-bit)
            LD      R1,TCDATAL
            AND     TDPND,#02H          ;  TD interrupt pending clear
            •
            •
```

•

```
POP     RP0
IRET
```

12 SERIAL PORT

OVERVIEW

Serial I/O module, SIO can interface with various types of external devices that require serial data transfer. The components of each SIO function block are:

— 8-bit control register (SIOCON)
— Clock selection logic
— 8-bit data buffer (SIODATA)
— 8-bit prescaler (SIOPS)
— 3-bit serial clock counter
— Serial data I/O pins (SI, SO)
— External clock input/out pin (SCK)

The SIO module can transmit or receive 8-bit serial data at a frequency determined by its corresponding control register settings. To ensure flexible data transmission rates, you can select an internal or external clock source.

PROGRAMMING PROCEDURE

To program the SIO modules, follow these basic steps:

1. Configure the I/O pins at port (SO, SCK, SI) by loading the appropriate value to the P1CONH register if necessary.
2. Load an 8-bit value to the SIOCON control register to properly configure the serial I/O module. In this operation, SIOCON.2 must be set to "1" to enable the data shifter.
3. For interrupt generation, set the serial I/O interrupt enable bit (SIOCON.1) to "1".
4. When you transmit data to the serial buffer, write data to SIODATA and set SIOCON.3 to 1, the shift operation starts.
5. When the shift operation (transmit/receive) is completed, the SIO pending bit (SIOCON.0) is set to "1" and an SIO interrupt request is generated.

SIO CONTROL REGISTER (SIOCON)

The control register for the serial I/O interface module, SIOCON, is located at F1H in set 1, bank 1. It has the control settings for SIO module.

— Clock source selection (internal or external) for shift clock
— Interrupt enable
— Edge selection for shift operation
— Clear 3-bit counter and start shift operation
— Shift operation (transmit) enable
— Mode selection (transmit/receive or receive-only)
— Data direction selection (MSB first or LSB first)

A reset clears the SIOCON value to '00H'. This configures the corresponding module with an internal clock source at the SCK , selects receive-only operating mode. The data shift operation and the interrupt are disabled. The data direction selected is MSB-first.

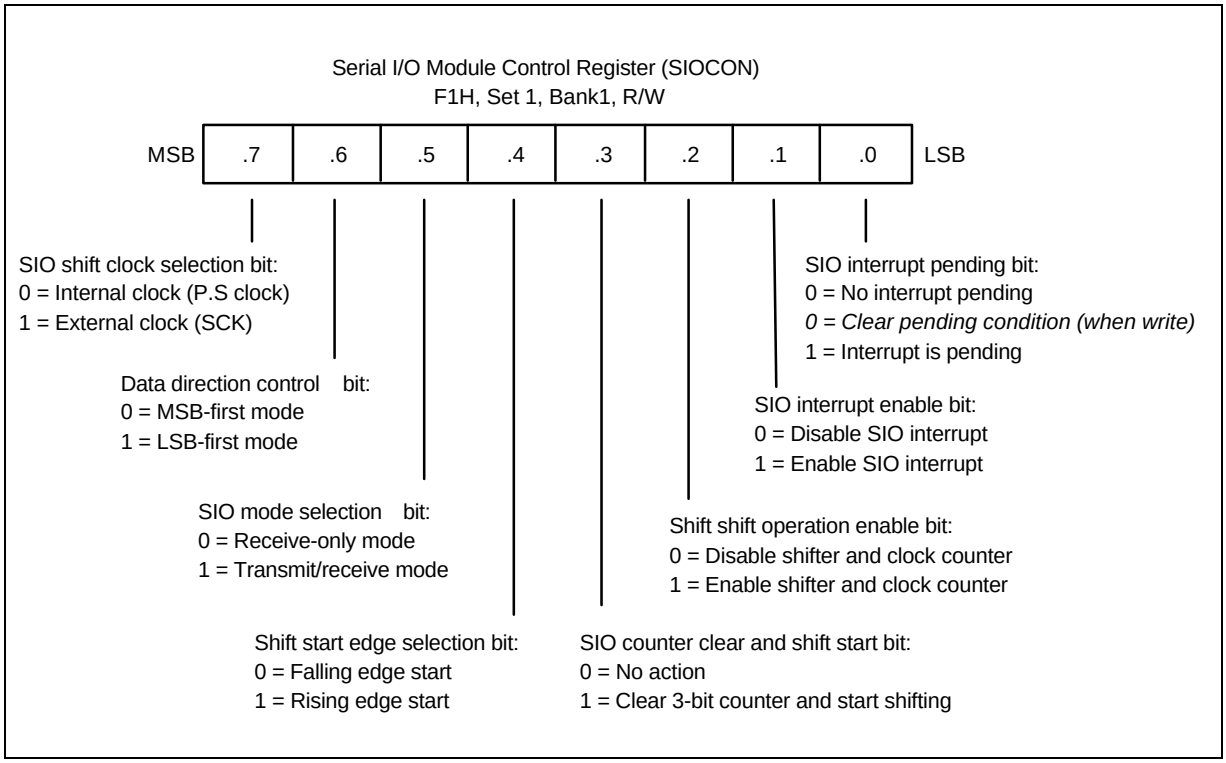


Figure 12-1. Serial I/O Module Control Registers (SIOCON)

SIO PRESCALER REGISTER (SIOPS)

The control register for the serial I/O interface module, SIOPS, is located at F3H in set 1, bank 1.

The value stored in the SIO prescaler registers, SIOPS, lets you determine the SIO clock rate (baud rate) as follows:

Baud rate = Input clock (CPU clock/2)/(Pre-scaler value + 1) or SCK input clock, where the input clock is CPU clock/2.

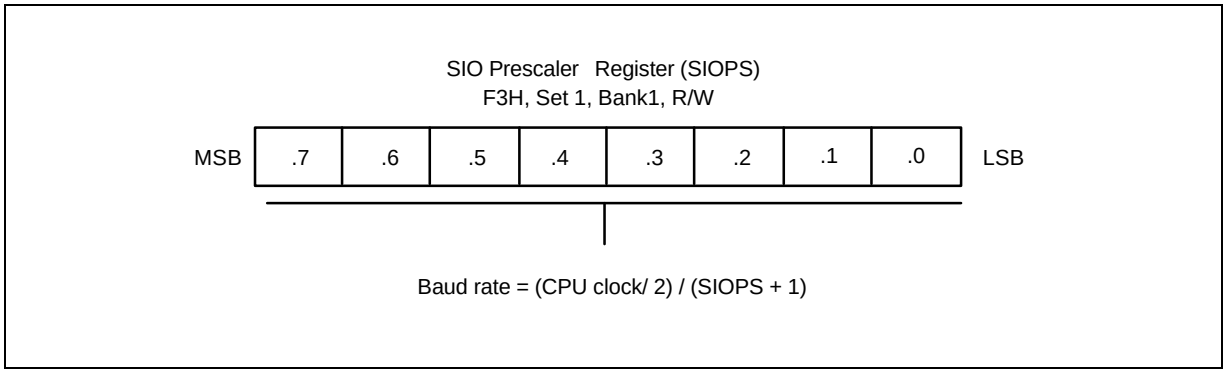
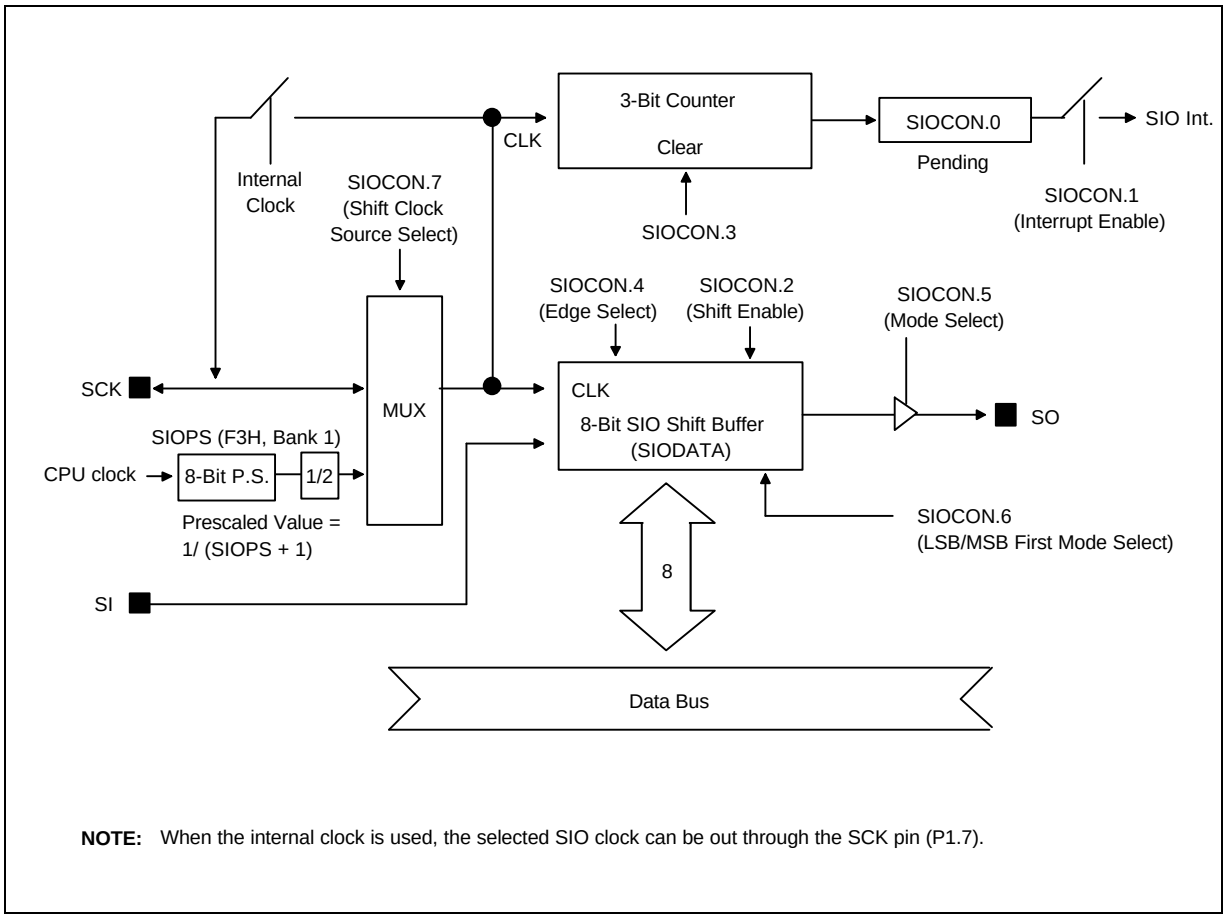


Figure 12-2. SIO Prescaler Register (SIOPS)

BLOCK DIAGRAM



NOTE: When the internal clock is used, the selected SIO clock can be out through the SCK pin (P1.7).

Figure 12-3. SIO Functional Block Diagram

SERIAL I/O TIMING DIAGRAMS

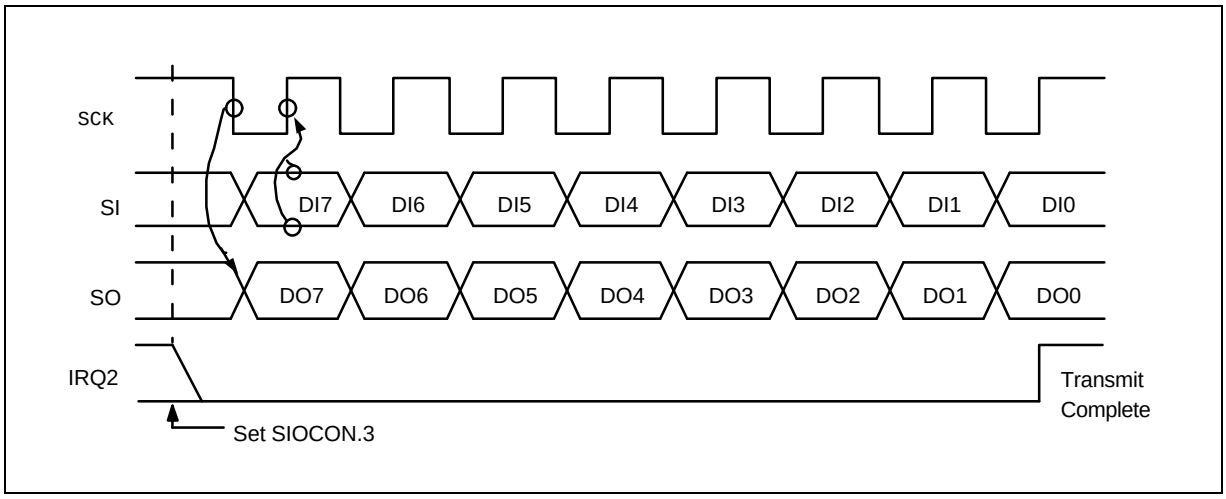


Figure 12-4. SIO Timing in Transmit/Receive Mode (Falling edge start)

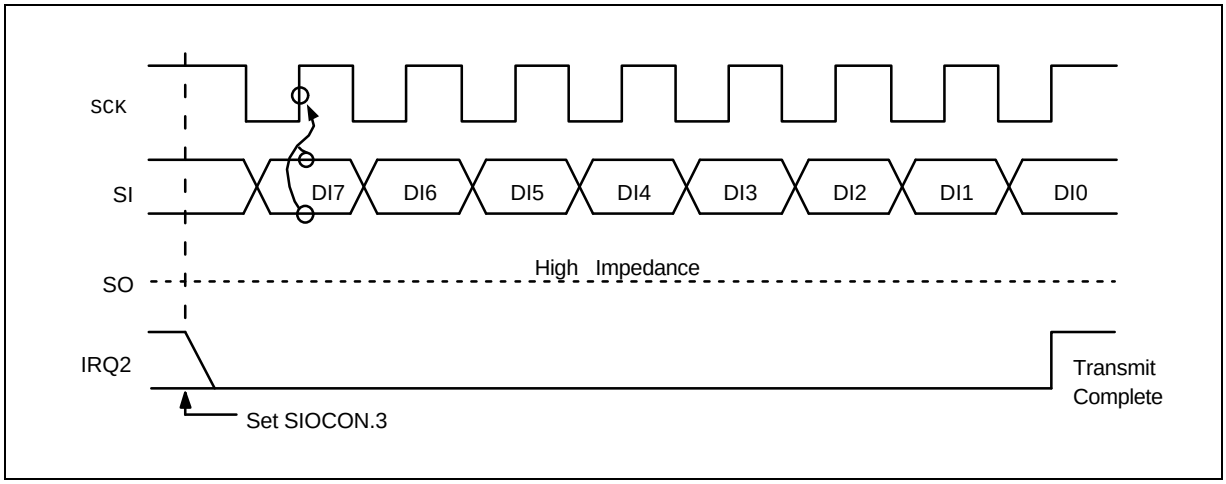


Figure 12-5. SIO Timing in Receive-Only Mode (Rising edge start)

☞ PROGRAMMING TIP — Use Internal Clock to Transfer And Receive Serial Data

1. The method that uses hardware pending check is used.

```
        •
        •
        •
        DI                          ;  P1 high nibble set to So enable, SCK output, SI input
        LD      P1CONH,#50H         ;

        SB1
        LD      SIODATA,TDATA       ;  Load data to be transferred
        LD      SIOPS,#90H          ;  Baud rate = (CPU clock/2)/(144 + 1)
        LD      SIOCON,#2EH         ;  Internal clock, MSB first, transmit/receive mode
        SB0                         ;  Falling edge start, clear 3-bit counter and start
        EI
        •
        •
        •
SIOINT  PUSH    RP0                 ;  Enable shifter and clock counter
        SRP0    #RDATA              ;  Enable SIO interrupt and clear pending
        SB1
        LD      R0,SIODATA          ;  Load received data to general register
        OR      SIOCON,#08H         ;  SIO restart
        POP     RP0
        IRET
```

☞ PROGRAMMING TIP — Use Internal Clock to Transfer And Receive Serial Data (Continued)

2. The method that uses software pending check is used.

```
                •
                •
                •
                LD      P1CONH,#50H     ; P1 high nibble set to SO enable, SCK output, SI input
                SB1
                LD      SIODATA,TDATA   ; Load data to be transferred
                LD      SIOPS,#90H      ; Baud rate = (CPU clock/2)/(144 + 1)
                LD      SIOCON,#2CH     ; Internal clock, MSB first, transmit/receive mode
                                        ; Falling edge start, enable shifter and clock counter
                                        ; Disable SIO interrupt
                EI

SIOtest:        LD      R6,SIOCON       ; To check whether transmit and receive is finished
                BTJRF   SIOtest,R6.0    ; Check pending bit
                NOP
                AND     SIOCON,#0FEH    ; Pending clear by software
                LD      RDATA,SIODATA   ; Load received data to RDATA
                •
                •
                •
                SB0
                •
                •
                •
```

NOTES

13 PWM

OVERVIEW

The KS88C4504 has two 8-bit PWM circuits. The operation of PWM circuits is controlled by a single control register, PWMCON. PWMCON contains a 4-bit prescaler adjusting the PWM frequency (cycle).

The PWM counter is an 8-bit incrementing counter. It is used by the 8-bit PWM circuits. To start the counter and enable the PWM circuits, you should set PWMCON.3 to "1". If the counter is stopped, it retains its current count value; when re-started, the counter resumes.

A 4-bit prescaler controls the clock input frequency to the PWM counter. By modifying the prescaler value, you can divide the input clock by one (non-divided), two, three, four, . . ., or sixteen. The prescaler output is the clock frequency of the PWM counter.

FUNCTION DESCRIPTION

PWM

The 8-bit PWM circuits have the following components:

— 8-bit counter with 4-bit prescaler

— 8-bit comparator (PWM0, PWM1)

— 8-bit reference data registers (PWM0, PWM1)

— PWM output pins (PWM0, PWM1)

PWM COUNTER

The PWM counter is an 8-bit incrementing counter. To determine the PWM module's operating frequency, the byte counter is compared to the PWM data register value.

PWM DATA REGISTER

PWM (duty) data registers, located in set 1, bank 1, determine the output value generated by each 8-bit PWM circuit. These PWM (duty) data registers are read/write addressable.

— 8-bit data registers PWM0 (FBH,set 1, bank1) and PWM1(FAH, set 1, bank 1)

To program the required PWM outputs, you should load the appropriate initialization values into the 8-bit data registers (PWM). To start the PWM counter, you set PWMCON.3 to "1".

A reset operation disables all PWM output. The current counter value is retained when the counter stops.

PWM CLOCK RATE

The timing characteristics of both 8-bit output channels are identical, and they are based on the CPU clock frequency. The 4-bit prescaler value in the PWMCON register determines the frequency of the counter clock: You can set PWMCON.4–PWMCON.7 to divide the CPU clock frequency by one (non-divided), two, three, etc.

Table 13-1. PWM Control and Data Registers

Register Name	Mnemonic	Address (Set 1, Bank 1)	Function
PWM data registers	PWM0, PWM1	FBH, FAH	8-bit PWM basic cycle frame value
PWM control register	PWMCON	F9H	PWM counter stop/resume, and 4-bit prescaler for CPU clock setting

PWM CONTROL REGISTER (PWMCON)

The control register for the PWM module, PWMCON, is located at register address F9H in set 1, bank 1. PWMCON is used for the 8-bit PWM modules. Bit settings in the PWMCON register control the following functions:

— PWM clock source selection
— 4-bit prescaler for scaling the PWM counter clock
— PWM counter stop/start operation
— PWM counter overflow interrupt control

A reset clears all PWMCON bits to logic zero, disabling the entire PWM module.

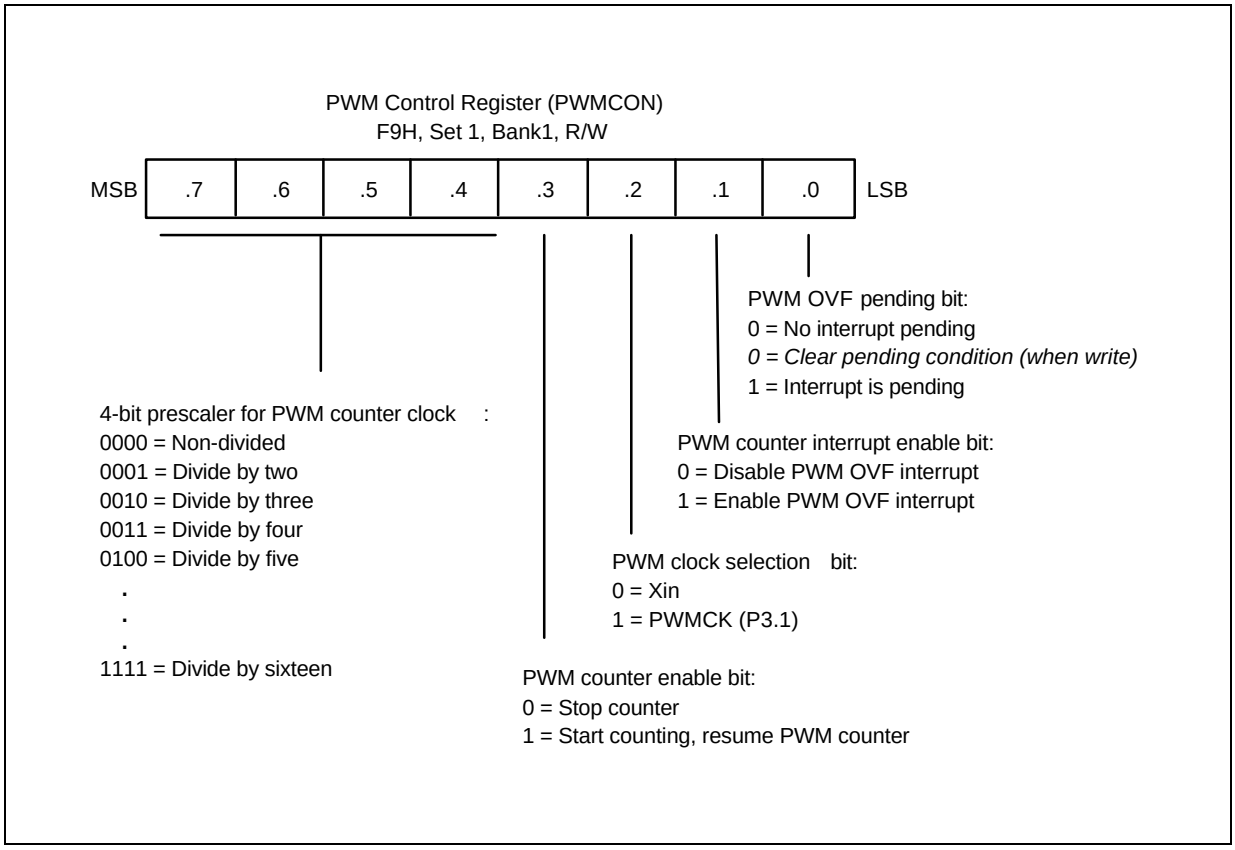


Figure 13-1. PWM Control Register (PWMCON)

BLOCK DIAGRAM

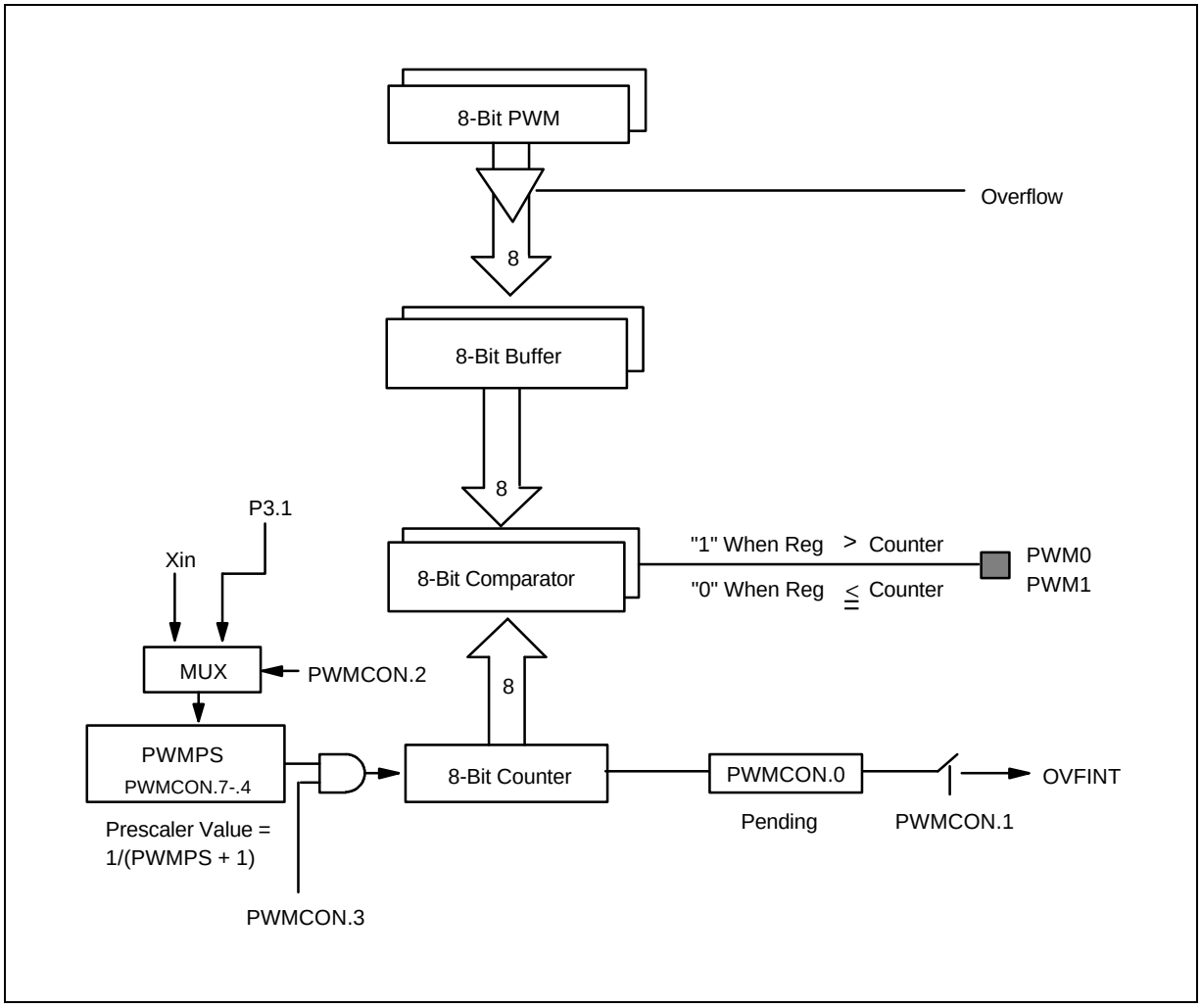


Figure 13-2. 8-Bit PWM Functional Block Diagram

PWM FUNCTION DESCRIPTION

The PWM counter is a 8-bit incrementing counter. To start the counter and enable the PWM module, you set bit 3 of the PWMCON register to "1". If the counter is stopped, it retains its current count value.

A 4-bit prescaler controls the clock input frequency to the PWM counter. Using prescaler bit settings, you can divide the input clock from one to sixteen. The prescaler output is the clock frequency of the PWM counter.

The PWM counter overflows when it reaches FFH, and then continues counting from zero. If the PWM counter overflow interrupt is enabled, an IRQ7 interrupt (vector F0H) is generated. The interrupt enable bit is bit 1 in the PWMCON register.

The PWM0 data register, called PWM0, is located in set 1, bank 1, address FBH. The PWM1 data register, PWM1, is in set 1, bank 1, at address FAH. Both data registers are read-write addressible. By loading specific values into the respective data registers, you can modulate the pulse width at the corresponding PWM output pins, PWM0 and PWM1. The two 8-bit PWM circuits function identically: Each has its own 8-bit data register and 8-bit comparator, and comparing its unique data register value to the 8-bit counter value.

The duty cycle of the PWM0 and PWM1 pin ranges from 0% to 99.6%, depending on the corresponding data register value.

To determine the PWM circuit's duty cycle, its 8-bit comparator sends the output level high ("1") when the data register value is greater than the 8-bit count value. The output level is low ("0") when the data register value is less than or equal to the 8-bit count value. Then, each PWM waveform is repeated continuously, at the same time frequency and duty cycle, until one of three events occurs:

— The counter is stopped

— The counter clock frequency is changed

— A new value is written to the PWM data register

The PWM0 data register, called PWM0, is located in set 1, bank 1, address F**B**H. The PWM1 data register, PWM1, is in set 1, bank 1, at address FAH. Both data registers are read-write addressable. By loading specific values into the respective data registers, you can modulate the pulse width at the corresponding PWM output pins, PWM0 and PWM1. The two 8-bit PWM circuits function identically: Each has its own 8-bit data register and 8-bit comparator, and comparing its unique data register value to the 8-bit counter value.

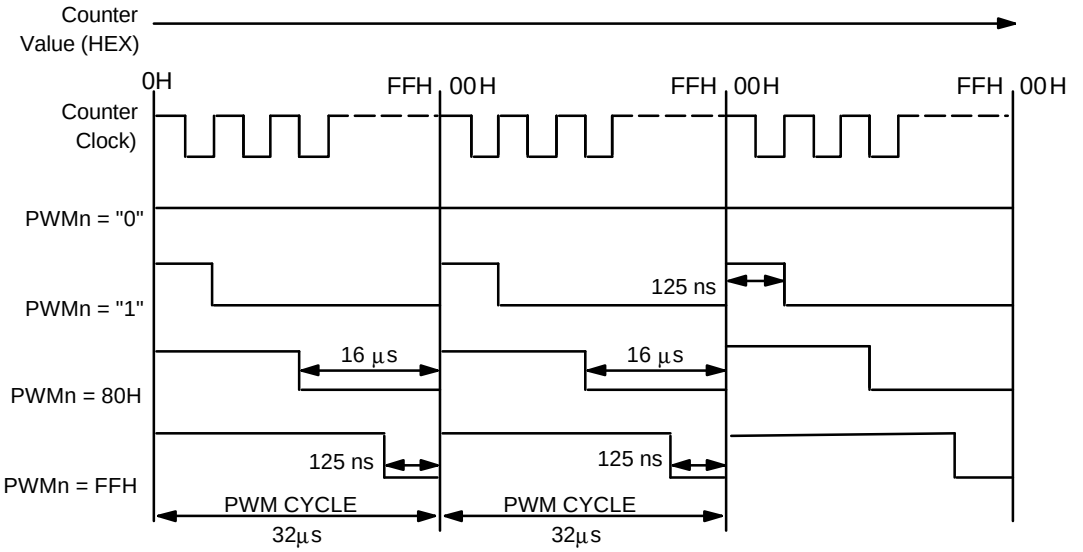
The duty cycle of the PWM0 and PWM1 pin ranges from 0% to 99.6%, depending on the corresponding data register value.

To determine the PWM circuit's duty cycle, its 8-bit comparator sends the output level high ("1") when the data register value is greater than the 8-bit count value. The output level is low ("0") when the data register value is less than or equal to the 8-bit count value. Then, each PWM waveform is repeated continuously, at the same frequency and duty cycle, until one of three events occurs:

— The counter is stopped

— The counter clock frequency is changed

— A new value is written to the PWM data register



NOTE: 1. 'n' = 0 and 1, for PWM0–PWM1.

Figure 13-3. PWM Output Waveform and Timing Diagram (PWM Clock = 8.19 MHz)

☞ PROGRAMMING TIP — Programming the PWM Module to Sample Specifications

This example shows you how to program the 8-bit pulse-width modulation (PWM) module, assuming the following parameters:

— External clock source selection at P3.1

— Enable PWM overflow interrupt

— PWM0 is toggled and PWM1 is high level output

```
        •
        •
        •
DI
AND     P3CONL,#11110011    ; Set P3.1 to normal input to supply PWM clock
SB1                         ; PWM0, PWM1 alternative function output enable
LD      PWM0,#80H           ; 50% duty cycle
LD      PWM1,#00H           ; High level output
LD      PWMCON,#3EH         ; PWM clock = ext clock/4
                            ; PWM start and counter clear
                            ; PWM OVF interrupt enable
                            ; Pending clear
EI
SB0
        •
        •
        •
```

NOTES

14 ANALOG-TO-DIGITAL CONVERTER

OVERVIEW

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

— Analog comparator with successive approximation logic

— D/A converter logic (resistor string type)

— ADC control register (ADCON)

— Four multiplexed analog data input pins (ADC0–ADC3)

— 8-bit A/D conversion data output register (ADDATA)

— AV_{REF} pin

FUNCTION DESCRIPTION

To initiate an analog-to-digital conversion procedure, you should write the channel selection data to the A/D converter control register ADCON to select one of the four analog input pins (ADCn, n = 0–3) and set the conversion start or enable bit, ADCON.0. The read-write ADCON register is located in set 1, bank 1, at address F7H.

During a normal conversion, ADC logic initially sets the successive approximation register to 80H (the approximate half-way point of an 8-bit register). This register is then updated automatically during each conversion step. The successive approximation block performs 8-bit conversions for one input channel at a time. You can dynamically select different channels by manipulating the channel selection bit value (ADCON.6–4) in the ADCON register. To start the A/D conversion, you should set a the enable bit, ADCON.0. When a conversion is completed, ADCON.3, the end-of-conversion (EOC) bit is automatically set to 1 and the result is dumped into the ADDATA register where it can be read. The A/D converter then enters an idle state. Remember to read the contents of ADDATA before another conversion starts. Otherwise, the previous result will be overwritten by the next conversion result.

NOTE

Because the A/D converter has no sample-and-hold circuitry, it is very important that fluctuation in the analog level at the ADC0–ADC3 input pins during a conversion procedure be kept to an absolute minimum. Any change in the input level, perhaps due to noise, will invalidate the result. If the chip enters to STOP or IDLE mode in conversion process, there will be a leakage current path in A/D block. You must use STOP or IDLE mode after ADC operation is finished.

CONVERSION TIMING

The A/D conversion process requires 4 clocks to convert each bit. Therefore, a total of 34 clocks are required to complete an 8-bit conversion: With an 16 MHz f_{OSC} clock frequency and with divide by 16, one clock cycle is 1 us. Each bit conversion requires 4 clocks, so the conversion rate is calculated as follows:

start 1clock + (4 clocks/bit x 8 bits) + EOC 1clock = 34 clocks, 1us x 34 = 34 us at 16 MHz

To get the correct A/D conversion data, A/D conversion time should be longer than 17us whichever OSC frequency is used.

A/D CONVERTER CONTROL REGISTER (ADCON)

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

— Analog input pin selection (bits 4, 5, and 6)
— End-of-conversion status detection (bit 3)
— A/D operation start or enable (bit 0)
— ADC clock source selection (bit 1 and bit 2)

After a reset, the ADC0 pin is automatically selected as the analog data input pin, and the start bit is turned off.

You can select only one analog input channel at a time. Other analog input pins (ADC0–ADC3) can be selected dynamically by manipulating the ADCON.4–6 bits.

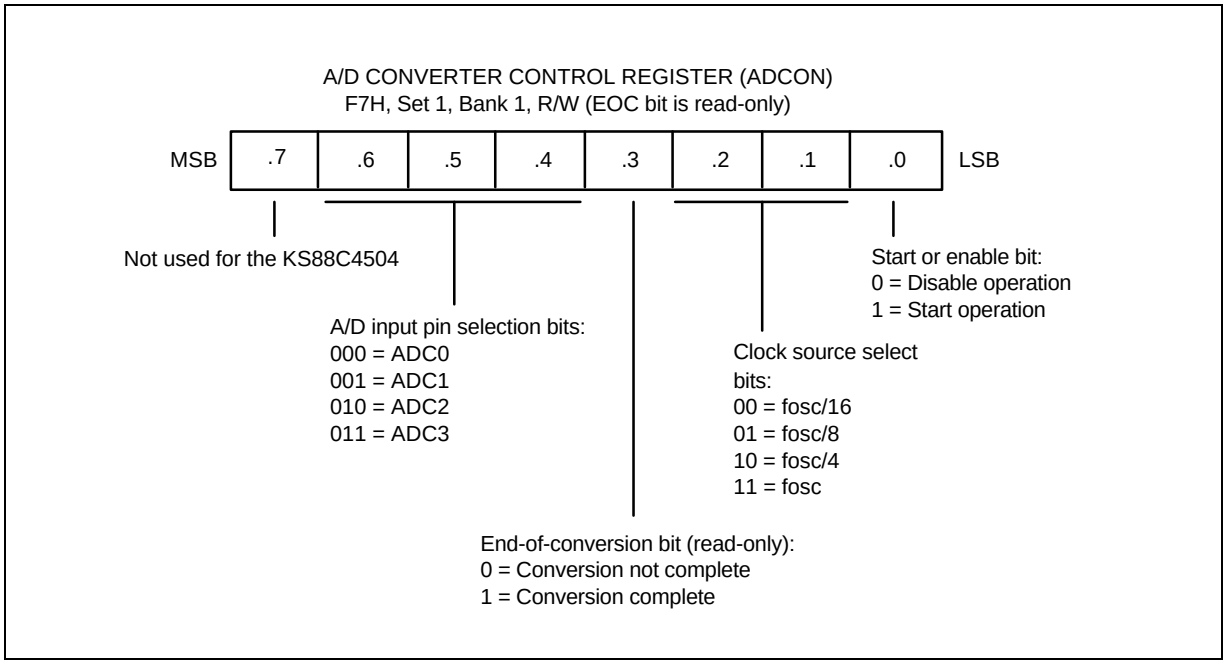


Figure 14-1. A/D Converter Control Register (ADCON)

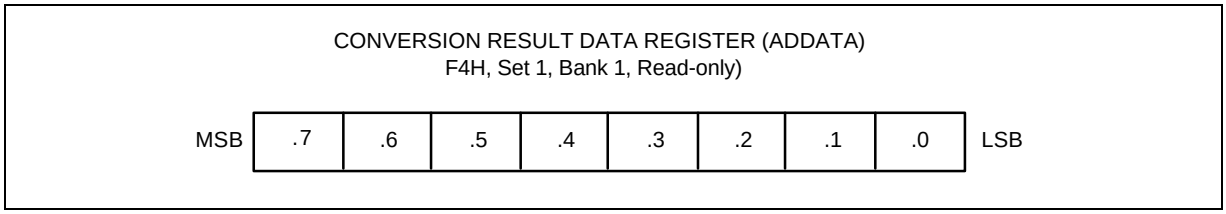


Figure 14-2. A/D Converter Data Register (ADDATA)

INTERNAL REFERENCE VOLTAGE LEVELS

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

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

BLOCK DIAGRAM

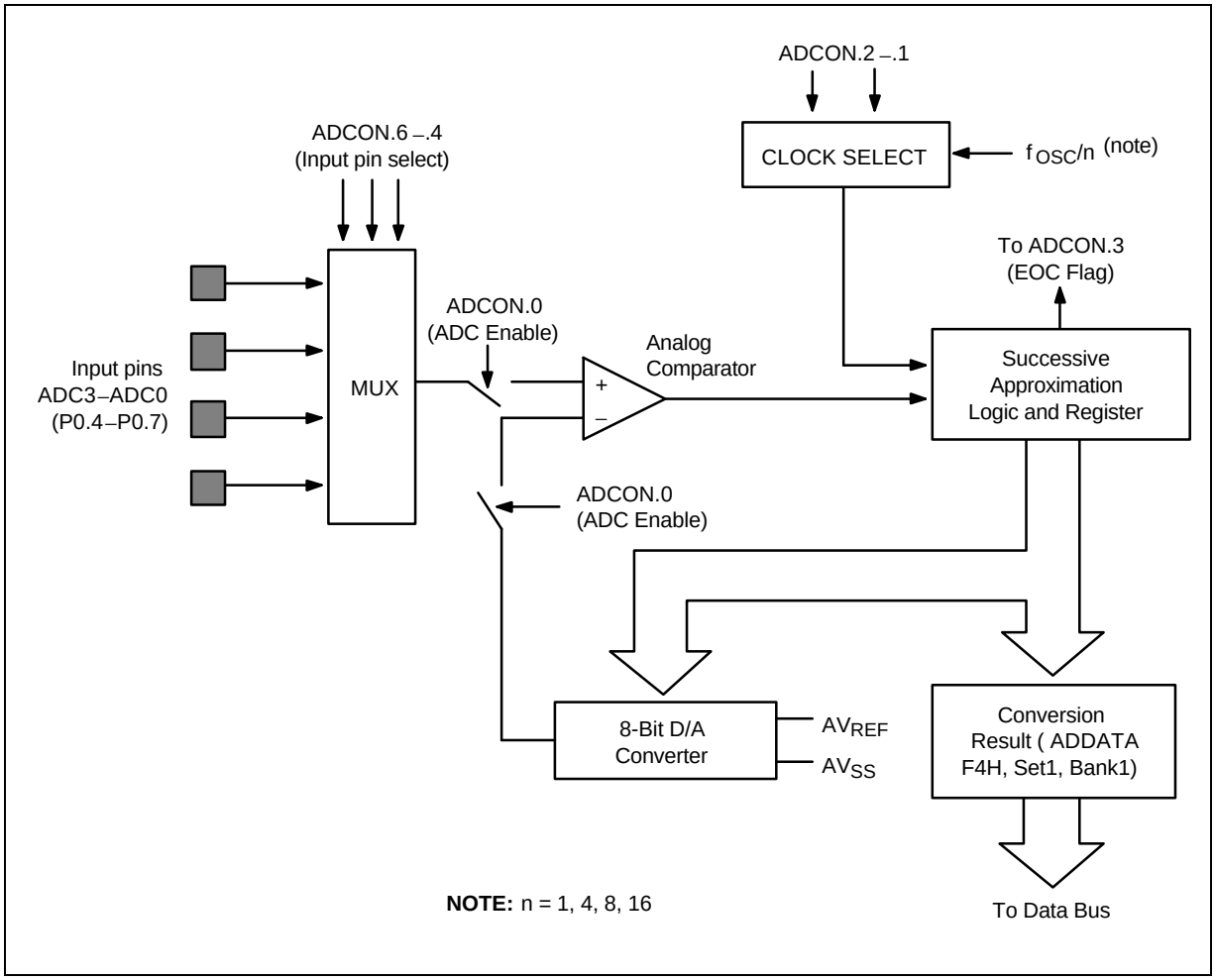


Figure 14-3. A/D Converter Functional Block Diagram

INTERNAL A/D CONVERSION PROCEDURE

1. To enable the A/D converter for incoming analog data, you select one of the four analog data input pins (ADC0–AD3) by writing the appropriate value to the ADCON register bits bit 4–bit 6.

2. Analog data can then be input within the acceptable voltage range (between AV_{SS} and AV_{REF}).

3. If input voltage > first reference voltage (1/2 AV_{REF}), the first conversion output is "1";
   If input voltage < first reference voltage (1/2 AV_{REF}), the first conversion output is "0".

4. After 34 clocks have elapsed, the converted digital values are loaded to the output buffer ADDATA (set 1, bank 1, F4H) and the ADC module goes into an idle state.

NOTE

During the 34-clock conversion time, the EOC bit value in the ADCON register is "0". When the 8-bit conversion is completed, the EOC bit value is automatically set to "1". This makes it possible to monitor the progress of A/D conversions internally by software.

5. You can now read the converted digital value from the ADDATA register.
6. To perform another conversion, execute steps 1–5 again.

☞ PROGRAMMING TIP — Sample A/D Conversion Program

This example show you how to program the A/D converter module. The program specifications are as follows:

— An A/D conversion operation occurs for each of the four analog input channels. The analog input at the selected channel will be converted to specific digital value.

— The conversion result for each input channel will be loaded to ADC_0 through ADC_3.

The program conditions are:

- ADC clock = 1 MHz (conversion time is 34 us)
- V_{DD} = 5 V, AV_{REF} = V_{DD}, AV_{SS} = GND (internally)
- ADC input = 0 V to 5 V
- Use the register ADC_REG+AD_CHNL (57H) to select a specific input channel before calling the conversion subroutine for that channel.

For a decision flow diagram of this sample program, see Figure 14-4.

☞ **PROGRAMMING TIP — Sample A/D Conversion Program (Continued)**

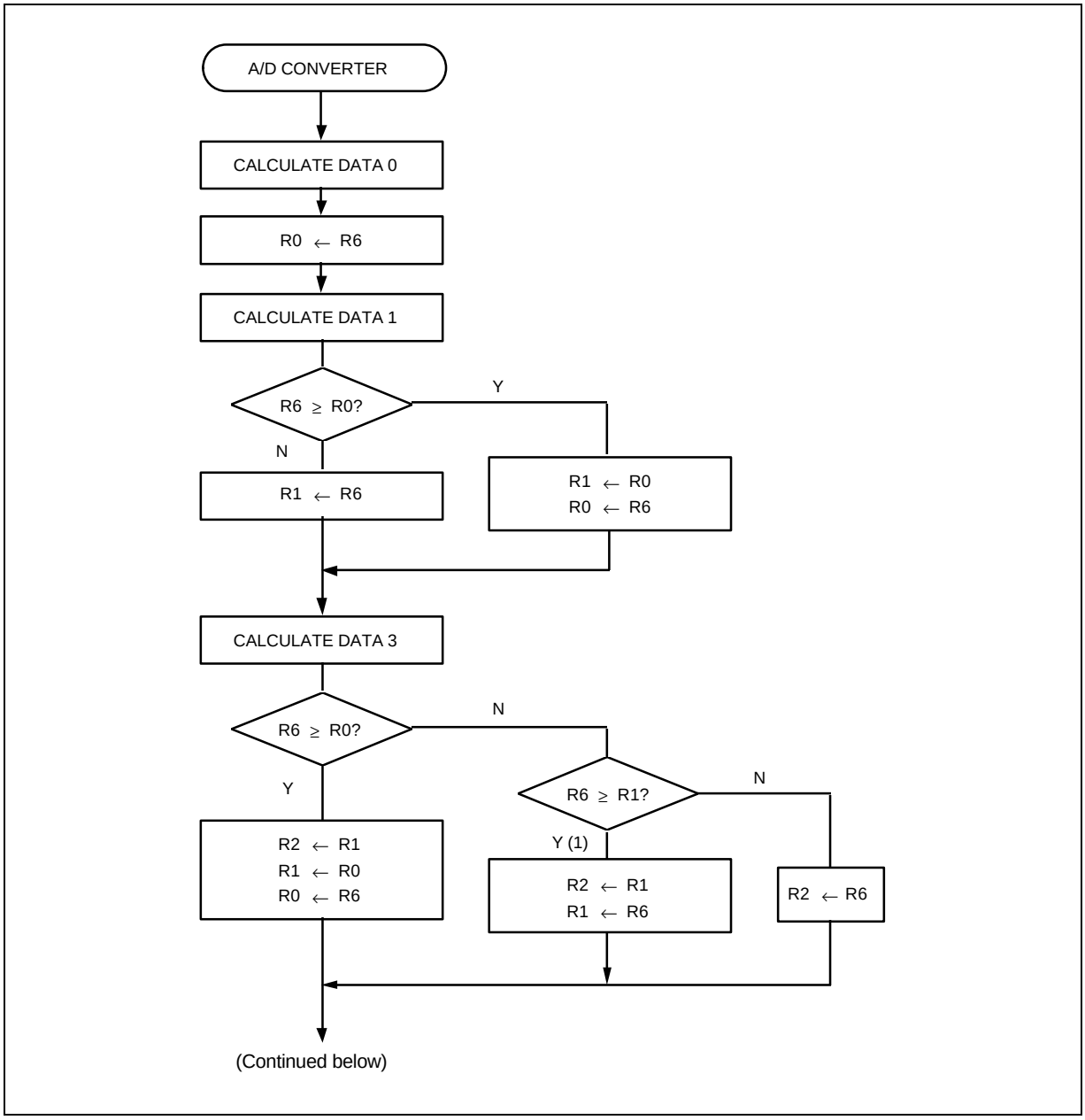


Figure 14-4. Decision Flow for A/D Converter Programming Tip

☞ **PROGRAMMING TIP — Sample A/D Conversion Program (Continued)**

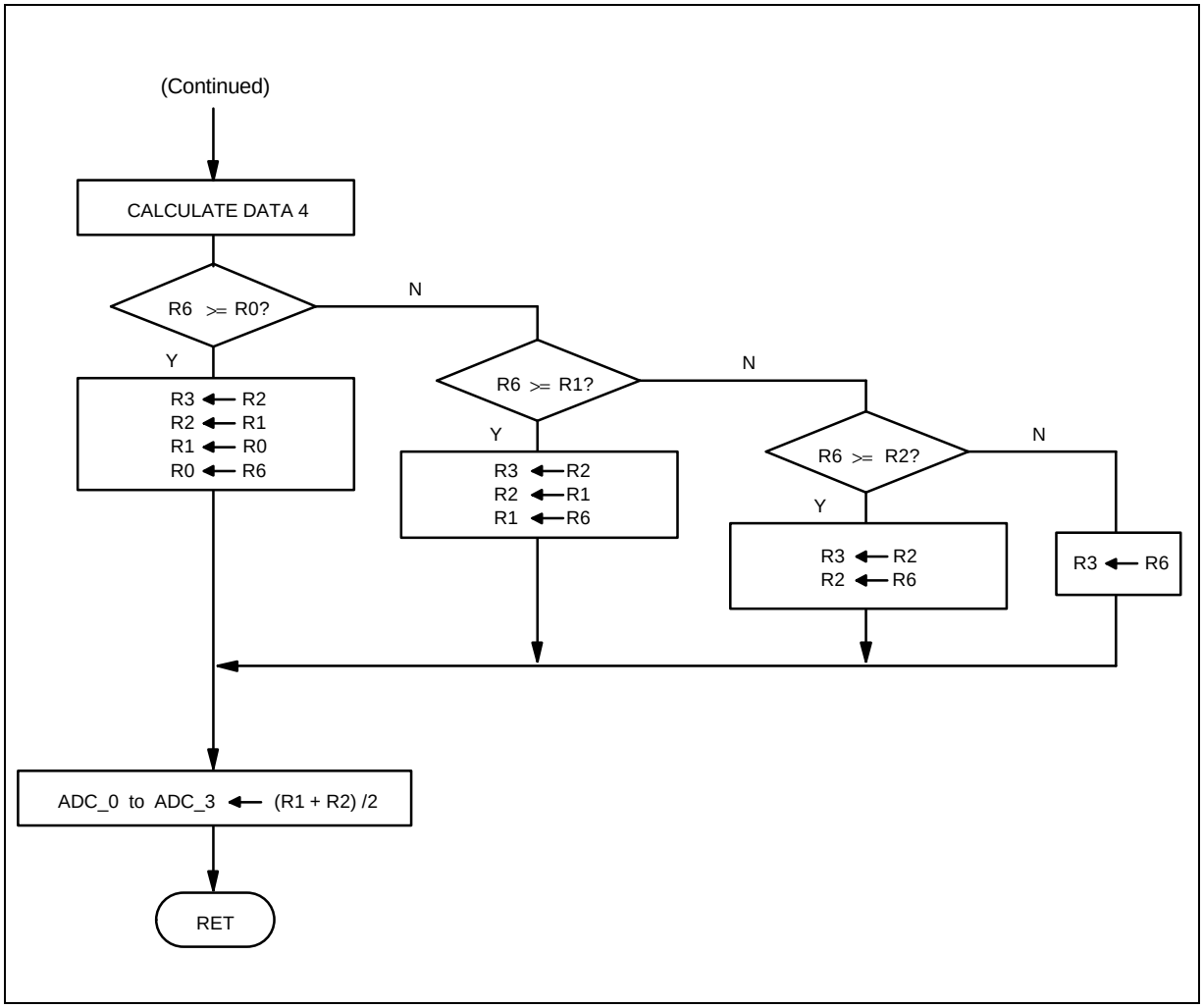


Figure 14-5. Decision Flow for A/D Converter Programming Tip (Continued)

☞ **PROGRAMMING TIP — Sample A/D Conversion Program (Continued)**

```
ADC_REG   .EQU    50H              ;  Page 0, 50H–57H is the working register area

AD_CHNL   .EQU    3                ;  Target channels are ADC0–ADC3

ADC_0     .EQU    58H              ;  A/D conversion result goes to these locations:
ADC_1     .EQU    59H
ADC_2     .EQU    5AH
ADC_3     .EQU    5BH

AD_CONVERTER:
          PUSH    PP
          PUSH    RP0
          PUSH    RP1

          CLR     PP               ;  Select page 0
          SRP     #ADC_REG
          SB1                      ;  Select bank 1

          CALL    AD_CONV          ;  Start 1st conversion
          LD      R0,R6

          CALL    AD_CONV          ;  Start 2nd conversion
          CP      R6,R0
          JR      ult,AD_00
          LD      R1,R0            ;  R6 ≥ R0
          LD      R0,R6
          JR      t,AD_10

AD_00:    LD      R1,R6            ;  R6 < R0

AD_10:    CALL    AD_CONV          ;  Start 3rd conversion
          CP      R6,R0
          JR      ult,AD_11
          LD      R2,R1            ;  R6 ≥ R0
          LD      R1,R0
          LD      R0,R6
          JR      t,AD_20
AD_11:    CP      R6,R1
          JR      ult,AD_12
          LD      R2,R1            ;  R6 ≥ R0
          LD      R1,R6
          JR      t,AD_20
AD_12:    LD      R2,R6            ;  R6 < R1
```

(Continued on the next page)

☞ **PROGRAMMING TIP — Sample A/D Conversion Program (Concluded)**

```
AD_20:    CALL    AD_CONV              ;  Start 4th conversion
          CP      R6,R0
          JR      ult,AD_21
          LD      R3,R2                ;  R6 ≥ R1
          LD      R2,R1
          LD      R1,R0
          LD      R0,R6
          JR      t,AD_30
AD_21:    CP      R6,R1
          JR      ult,AD_22
          LD      R3,R2                ;  R6 ≥ R1
          LD      R2,R1
          LD      R1,R6
          JR      t,AD_30
AD_22:    CP      R6,R2
          JR      ult,AD_23
          LD      R3,R2                ;  R6 ≥ R2
          LD      R2,R6
          JR      t,AD_30
AD_23:    LD      R3,R6                ;  R6 < R2

AD_30:    CLR     R0
          ADD     R1,R2
          ADC     R0,#0H
          DIV     RR0,#2H
          LD      R5,#AD_CHNL
          LD      #ADC.0[R5],R1        ;  Final result
          SB0
          RET                          ;  End of AD_CONVERTER routine

AD_CONV:  LD      ADCON,R7             ;  Start A/D conversion! Clock and channel is selected
                                       ;  according to R7
AD_WAIT:  LD      R6,ADCON
          BTJRF   AD_WAIT,R6.3         ;  Is the conversion finished?
          NOP                          ;  Yes.
          LD      R6,ADDATA            ;  Load converted data to output register
          RET
```

15 EXTERNAL INTERFACE

OVERVIEW

The KS88 architecture supports accesses to memory and other peripheral devices over an external interface. Both program and data memory areas can be accessed by up to the 16-bit address and 8-bit data bus.

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

The KS88C4504/P4504 has 80 pins, 42 of which are used for I/O. 28 pins are dedicated to external interface. Because the address and data bus carries 16-bit memory addresses, up to 64 Kbytes of memory space can be addressed. 4 Kbytes of program memory are available in the on-chip ROM.

The remaining 60 Kbytes of program memory can be implemented externally (or the entire 64-Kbyte program memory can be configured externally using the ROM-less operating mode).

A 64-Kbyte data memory area can also be implemented externally using the external interface. The data memory (DM) signal line is used to keep external data memory and the program memory signal line is used to keep external program memory.(PM)

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

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

Wait signal (WAIT) line is configured at port 5 pins P5.0 by bit settings in the port 5 control register, P5CON.

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

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

CONFIGURATION OPTIONS FOR EXTERNAL PROGRAM MEMORY

Program memory (ROM) stores program code and table data. Instructions can be fetched, or data read, from ROM locations. The KS88C4504 has 4 Kbytes internal mask-programmable ROM (locations 0H–0FFFH).

NOTE:

PM , DM, RD, WR signals are "HIGH" during the reset low.

You will recall that the SAM87RC dual bus architecture can address up to 64 K bytes of program memory area. Using the external interface, it is possible to configure additional program memory space externally for applications that require more than the 4-Kbyte internal masked ROM. There are two ways to configure external program memory:

Option 1: Configure the remaining 60 Kbytes (locations 1000H–FFFFH) externally.

Option 2: Using the ROM-less mode option, configure the entire 64-Kbyte area (0000H–FFFFH) externally.

Option 1: Configuring the Remaining 60 Kbytes of Program Memory Externally

In order to access the external 60-Kbyte program memory area, the internal 4-Kbyte ROM must contain the initialization routine for configuring the external interface. The routine must also jump out of the 4-Kbyte space into the external program memory address range (1000H–FFFFH). For this reason, the internal 4-Kbyte ROM area must always be mask-programmed. When 0 V is supplied to the EA pin, you can set external interface pin to the output High status by loading 00h to EXTBUS (F5h, set 1, bank 1). When 01h is loaded to EXTBUS, Address and data will appear on the external interface pin. RESET value of EXTBUS is 00H.

Option 2: Using ROM-less Mode to Configure 64 Kbytes of Program Memory Externally

Since the costs of configuring 60 Kbytes or 64 Kbytes of external ROM is generally the same, option 2 will usually be chosen if an external program memory is required.

To configure the entire 64-Kbyte ROM address range externally, you must configure the KS88C4504/P4504 to operate in ROMless mode. This is done by applying 5 V to the EA pin (pin 16). You may recall that the KS88C4504/P4504 operates in normal (4-Kbyte internal ROM) mode when 0 V is applied to the EA pin.

In ROMless mode, an access to the internal ROM is disabled. Please note that 5 V must be applied to the EA pin prior to a RESET and the voltage must remain at the 5-volt level during normal operation.

If you plan to implement Option 2, the KS88C4504's internal 4-Kbyte ROM does not need to be mask-programmed.

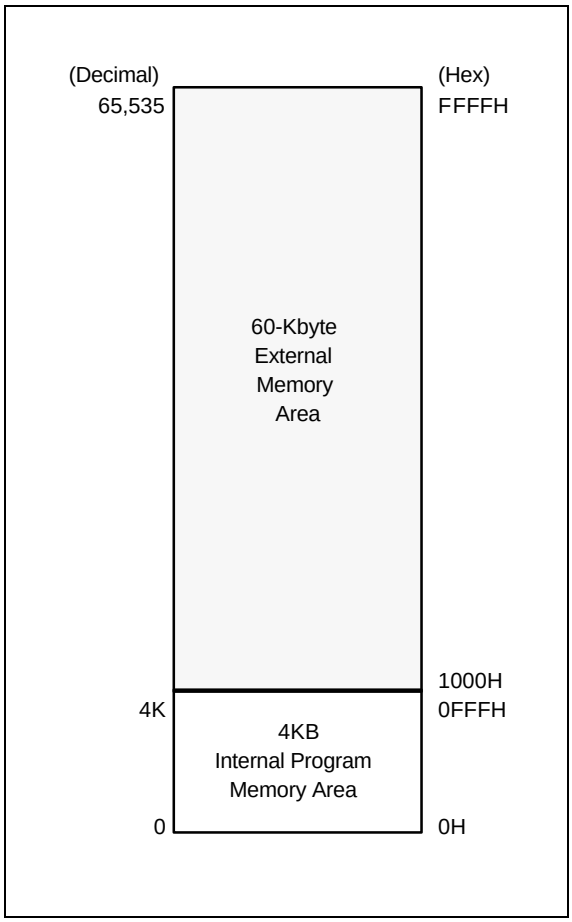


Figure 15-1. Program Memory (Normal Operating Mode)

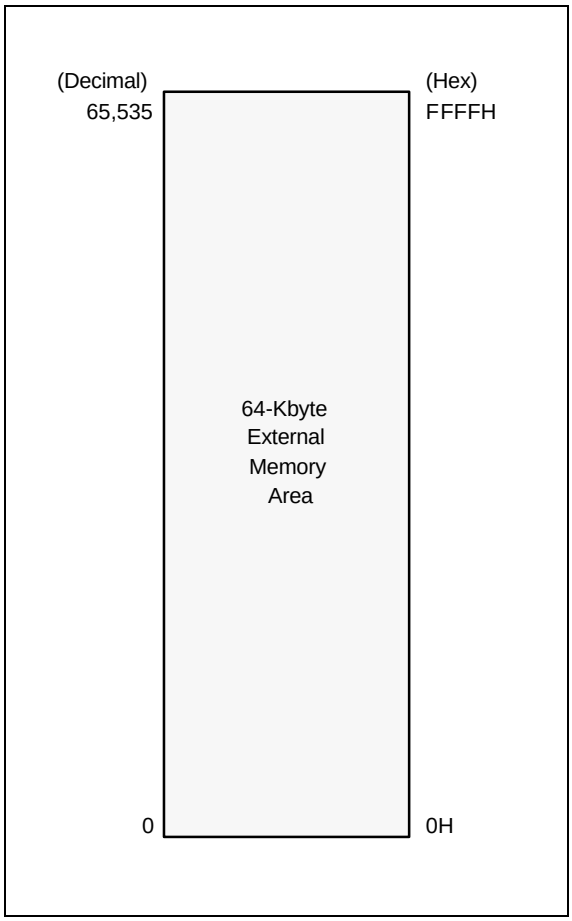


Figure 15-2. Program Memory (ROMless Operating Mode)

SYSTEM MODE REGISTER (SYM)

The system mode register, SYM, controls interrupt processing and also contains the enable bit for the tri-state external memory interface (SYM 7). When the tri-state bit is enabled, the lines of the external memory interface are set to high impedance (that is, the signals "float") for use in multiprocessor applications that require a shared external bus.

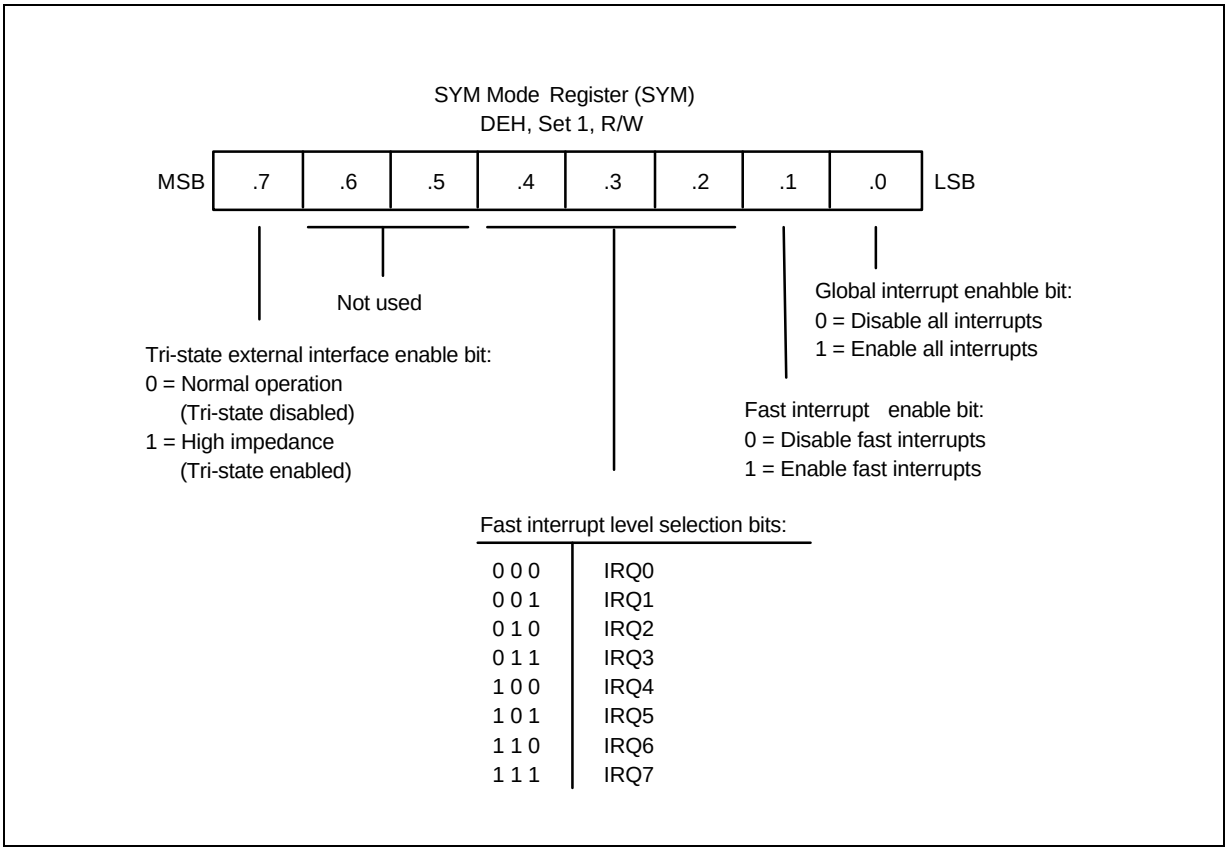


Figure 15-3. System Mode Register (SYM)

EXTERNAL MEMORY TIMING REGISTER (EMT)

The external memory timing register, EMT, is used to control bus operations for the external interface. A reset clears the WAIT enable and stack area selection bits to "0". This disables the wait function and selects the internal register file as the system stack area.

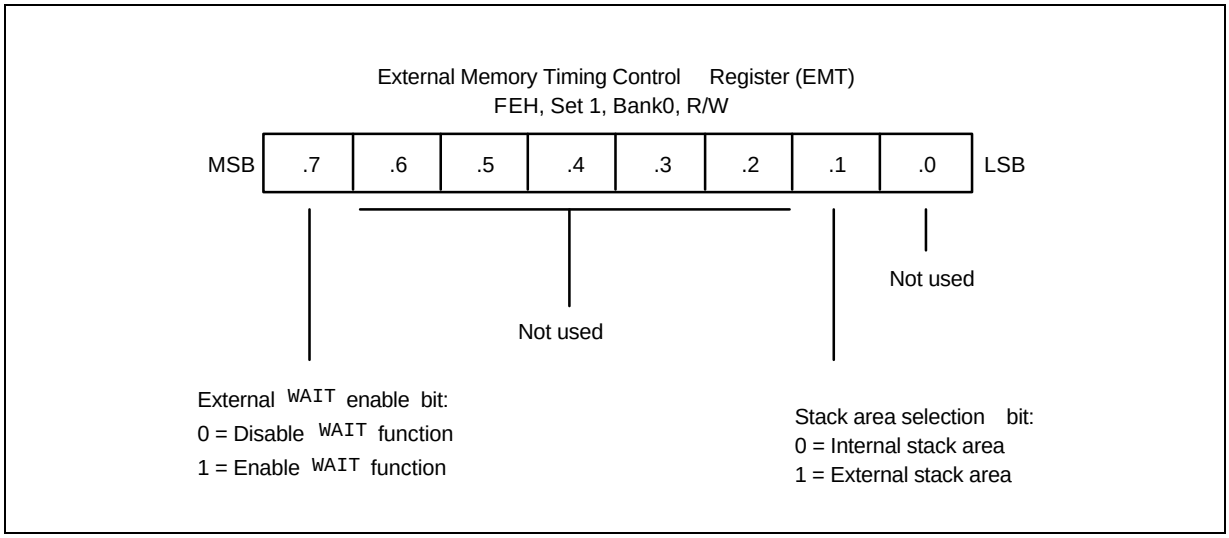


Figure 15-4. External Memory Timing Control Register (EMT)

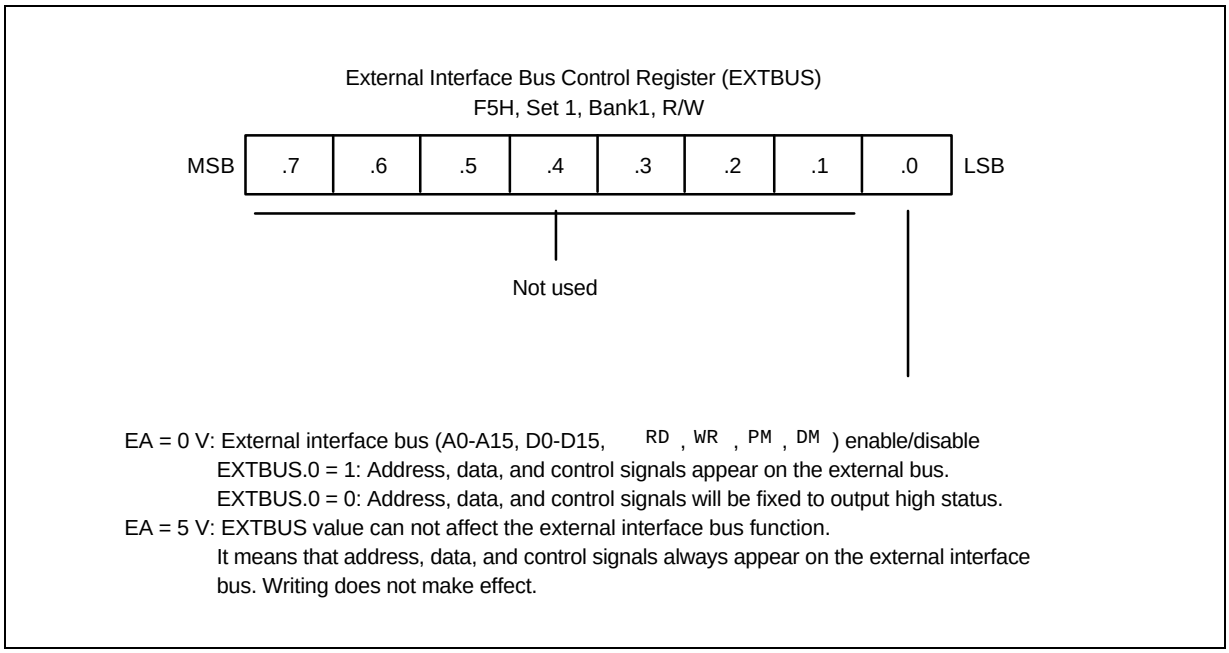


Figure 15-5. External Interface BUS Control Register (EXTBUS)

HOW TO CONFIGURE THE EXTERNAL INTERFACE

As the KS88C4504/P4504 has dedicated external interface ports, so there needs no special settings for external interface. P5 (WAIT) can be used as normal I/O or external interface. A0–A15 and PM, DM, RD, WR can be dedicated to external interface and WAIT pins enable shared with P5. If you want to use WAIT as normal port, you must set proper value to P5CON.

CONFIGURING SEPARATE EXTERNAL PROGRAM AND DATA MEMORY AREAS

You can address external program and data memory locations as a single combined space or as two separate spaces. If program and data memory spaces are implemented separately, this separation is maintained logically using the data and program memory select signal (DM and PM).

The DM pin's state goes active low to select data memory whenever one of the following instructions is executed:

— LDE (Load external data memory)
— LDED (Load external data memory and decrement)
— LDEI (Load external data memory and increment)
— LDEPD (Load external data memory with pre-decrement)
— LDEPI (Load external data memory with pre-increment)

If you set the stack area selection bit in the EMT register (EMT.1) to "1", the system stack area is configured externally. In this case, the DM signal will go active low whenever a CALL, POP, PUSH, RET, or IRET instruction is executed.

USING AN EXTERNAL SYSTEM STACK

The KS88 architecture supports stack operations in either the internal register file or in externally configured data memory. The PUSH and POP instructions support external system stack operations.

To select the external stack area option, you must set bit 1 in the external memory timing register (EMT, FEH) to "1".

NOTE

The instruction you use to modify the stack selection bit in the EMT register should not be immediately followed by an instruction that uses the stack. This could cause a program error. Also, remember to disable interrupts by executing a DI instruction before you modify the stack selection bit.

A 16-bit stack pointer value (SPH and SPL) is required for external stack operations. After a reset, the SP values are undetermined.

Return addresses for procedure calls and interrupts, as well as dynamically generated data are stored on an externally-defined stack. The contents of the PC are saved on the external stack during a CALL instruction and restored by a RET instruction. When an interrupt occurs, the contents of the PC and the FLAGS register are saved to the external stack. These values are then restored by an IRET instruction.

Table 15-1. External Interface Control Register Values After a RESET (Normal Mode, EA=Vss)

Register Name	Mnemonic	Address		Bit Values After RESET (EA Pin is Low)							
		Dec	Hex	7	6	5	4	3	2	1	0
System Mode Register	SYM	222	DEH	0	–	–	x	x	x	0	0
External interface Bus control register	EXTBUS	245	F5H	–	–	–	–	–	–	–	0
Port 5 Control Register	P5CON	250	FAH	–	–	–	–	0	0	0	0
External Memory Timing Register	EMT	254	FEH	0	–	–	–	–	–	0	–

NOTE: A dash (-)indicates that the bit is not used or not mapped; an 'x' means that the value is undefined after a RESET.

EXTERNAL BUS OPERATIONS

The number of machine cycles that are required for external memory operations is two machine cycle.

The notation used to describe basic timing periods in Figures 15.6–15.13 are machine cycles (Mn), timing states (Tn), and clock periods. The clock wave form is shown for clarification only and does not have a specific timing relationship to the other signals.

CONTROLLING EXTERNAL BUS OPERATIONS

Whenever the KS88C4504/P4504 external peripheral interface is active, the addresses of all internal program memory references will also appear on the external bus. This should have no effect on the external system, however, because the RD and WR signals are always high. (RD and WR goes low only during external memory references.)

Shared Bus Feature

The RD, WR, DM, PM signals, address, and data bus can be set to high impedance to enable the KS88C4504/P4504 to share common resources with other bus masters. This feature is often required for multiprocessor or related applications that require two or more devices that share the same external bus.

The tri-state memory interface enable bit in the system mode register (SYM.7) controls this function. When SYM.7 = "1", the tri-state function is enabled, all external interface lines are set to high impedance, and the external bus is put under software control.

EXTENDED BUS TIMING FEATURES

The KS88C4504 accommodates slow external memory access and cycle times using externally-issued wait signal method.

Externally-Issued Wait Feature

You can use an external device to influence timing of KS88C4504/P4504 external memory accesses. You do this by stretching the RD and WR by one additional cycle for an indefinite period of time. To initiate this additional "stretch" cycle, an external device must issue a WAIT signal to the KS88C4504/P4504. The P5.0 pin accepts the WAIT input from the external source. WAIT input is sampled on each internal clock.

The WAIT signal stretches the data strobe by one additional internal clock cycle for as long as the external device continues to hold the P5.0 pin to low level.

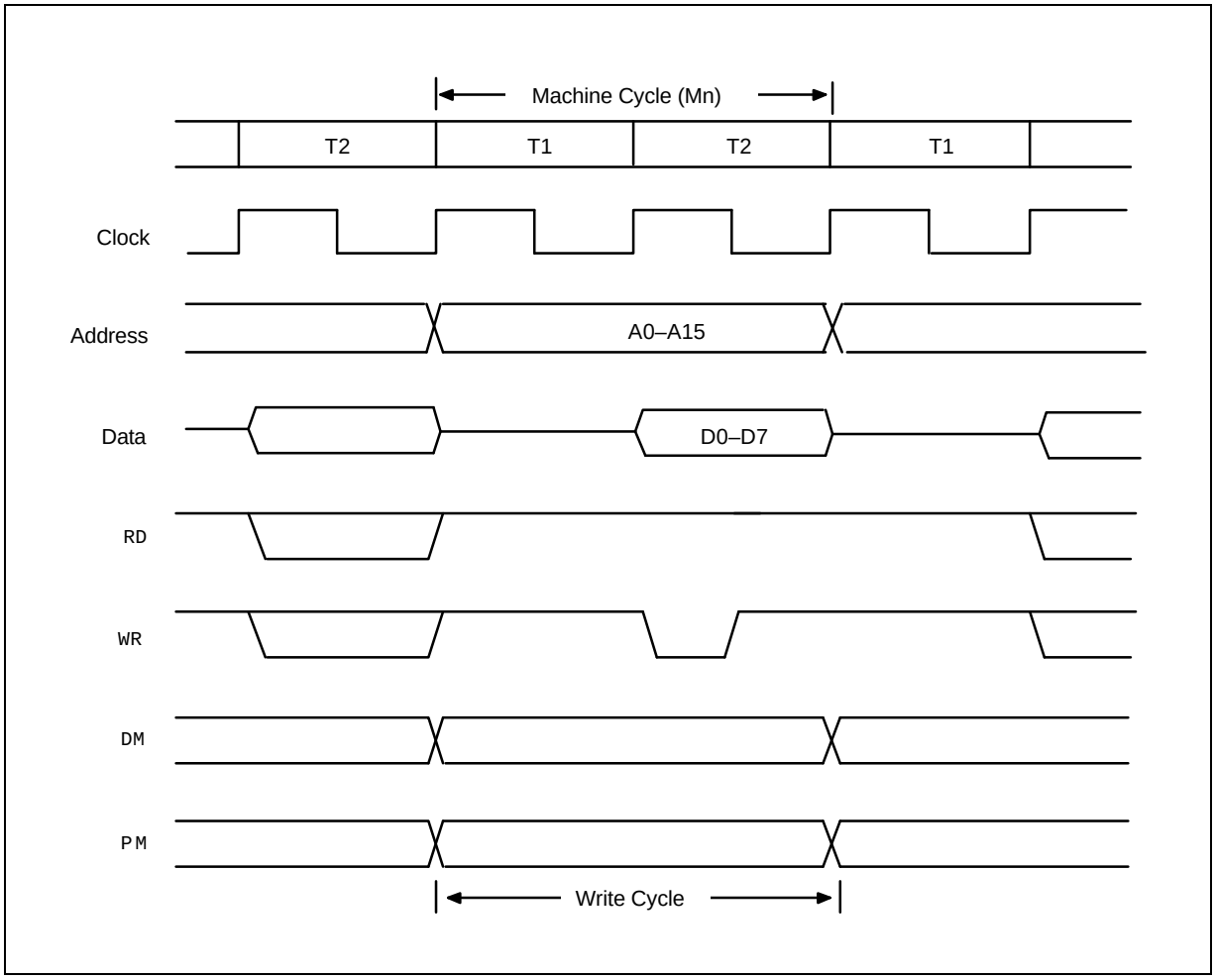


Figure 15-6. External Bus Write Cycle Timing Diagram (No Wait, Address, and Data Separated)

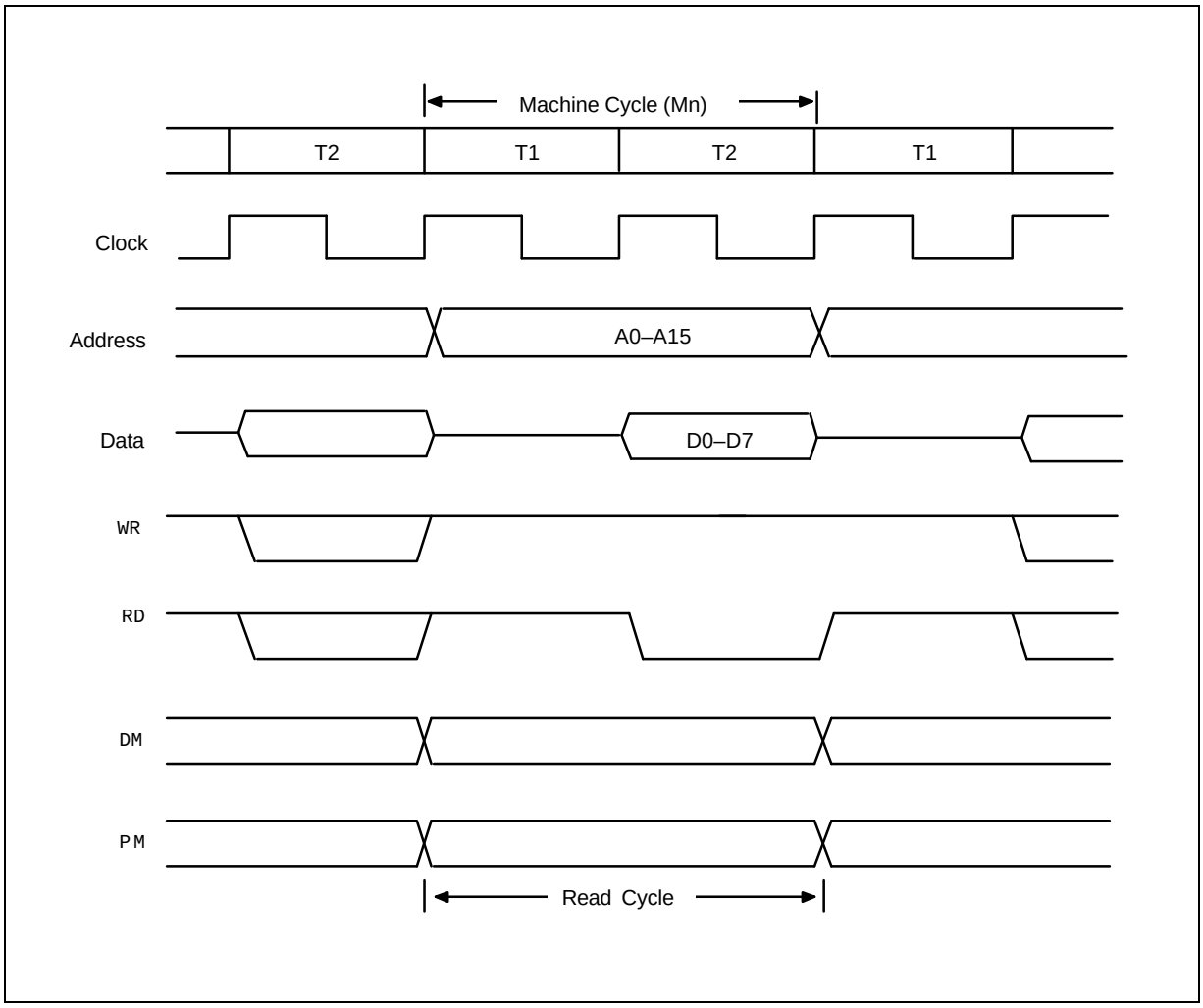


Figure 15-7. External Bus Read Cycle Timing Diagram (No Wait)

Table 15-2. KS88C4504 External Memory Interface Signal Descriptions

Signal Name	Symbol	Pin	Active Level	Description
Read	RD	3	Low	RD determines the data transfer direction for external memory operations.
Write	WR	4	Low	WR is low when writing to external program memory or data memory locations, and is high for all other operations.
Memory select	DM	2	Low	When it is low, DM selects data memory.
	PM	1	Low	When it is low, PM selects program memory.
External hardware wait signal	WAIT	7	Low	This pin is sampled at each rising edge of the CPU clock. If it is held low, it can "stretch" the data strobe indefinitely by adding one clock period to the data valid time.

NOTE: If bit 7 of the SYM register is high level, and assuming the external memory interface is configured, the RD, PM, WR, and DM signals, as well as address and data signal, will be set to high impedance state. This causes the external interface signals to 'float'.

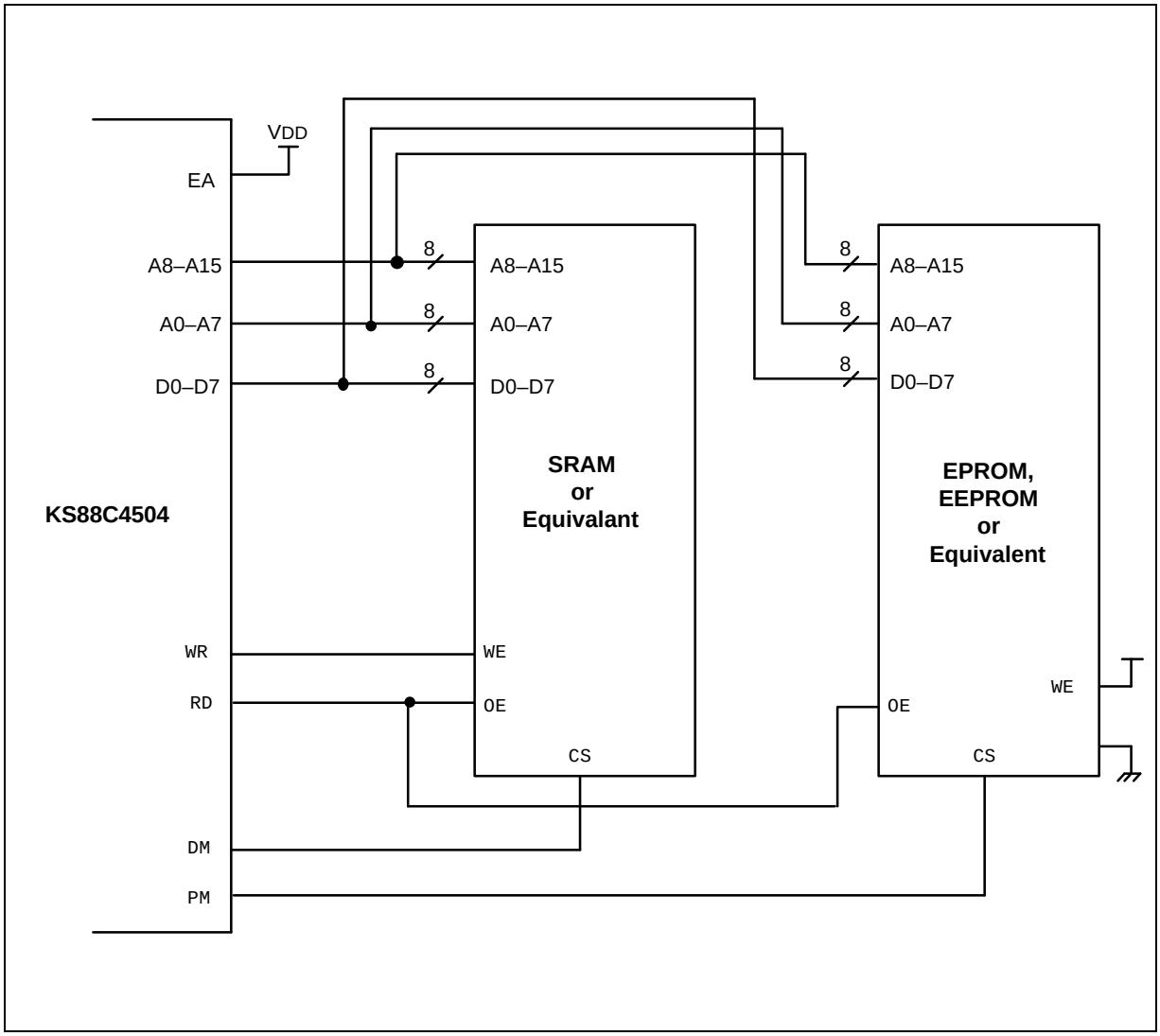


Figure 15-8. External Interface Function Diagram (with SRAM and EPROM or EEPROM)

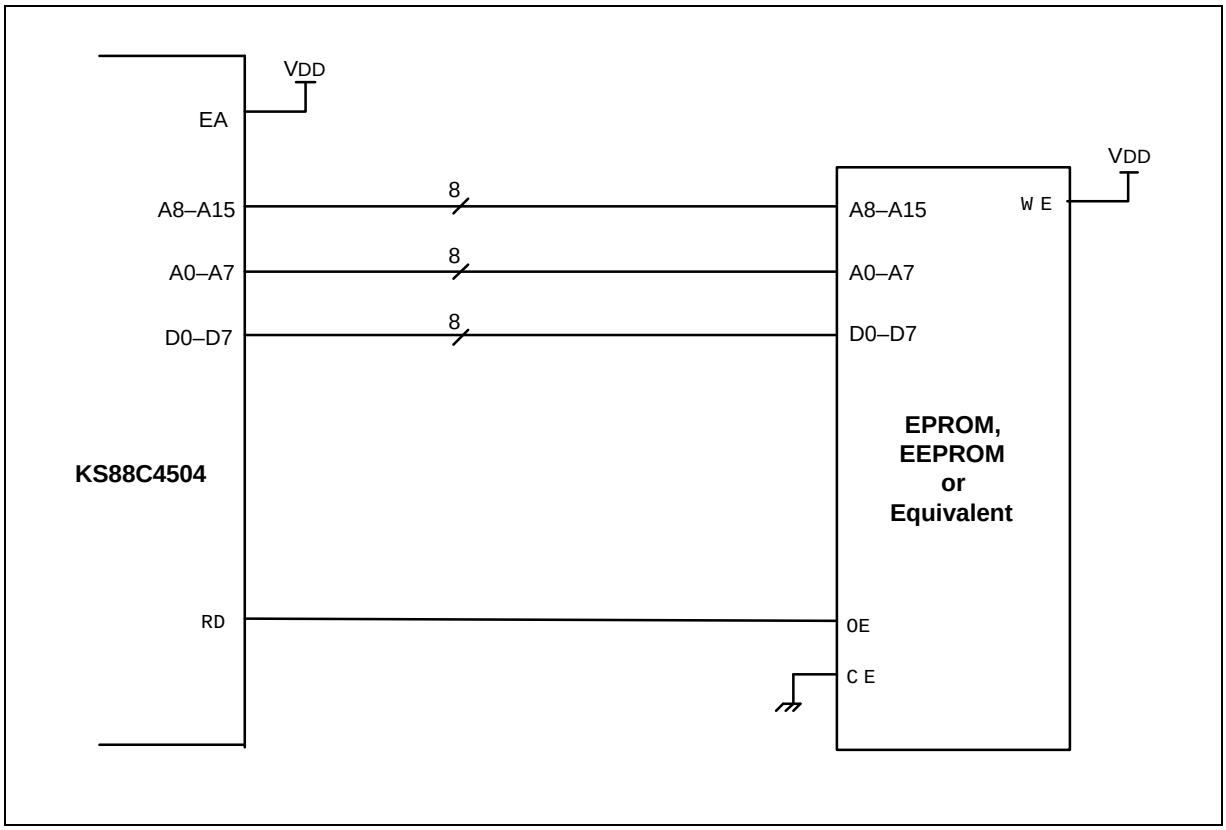


Figure 15-9. External Interface Function Diagram (External ROM Only)

SAM8 INSTRUCTION EXECUTION TIMING DIAGRAMS

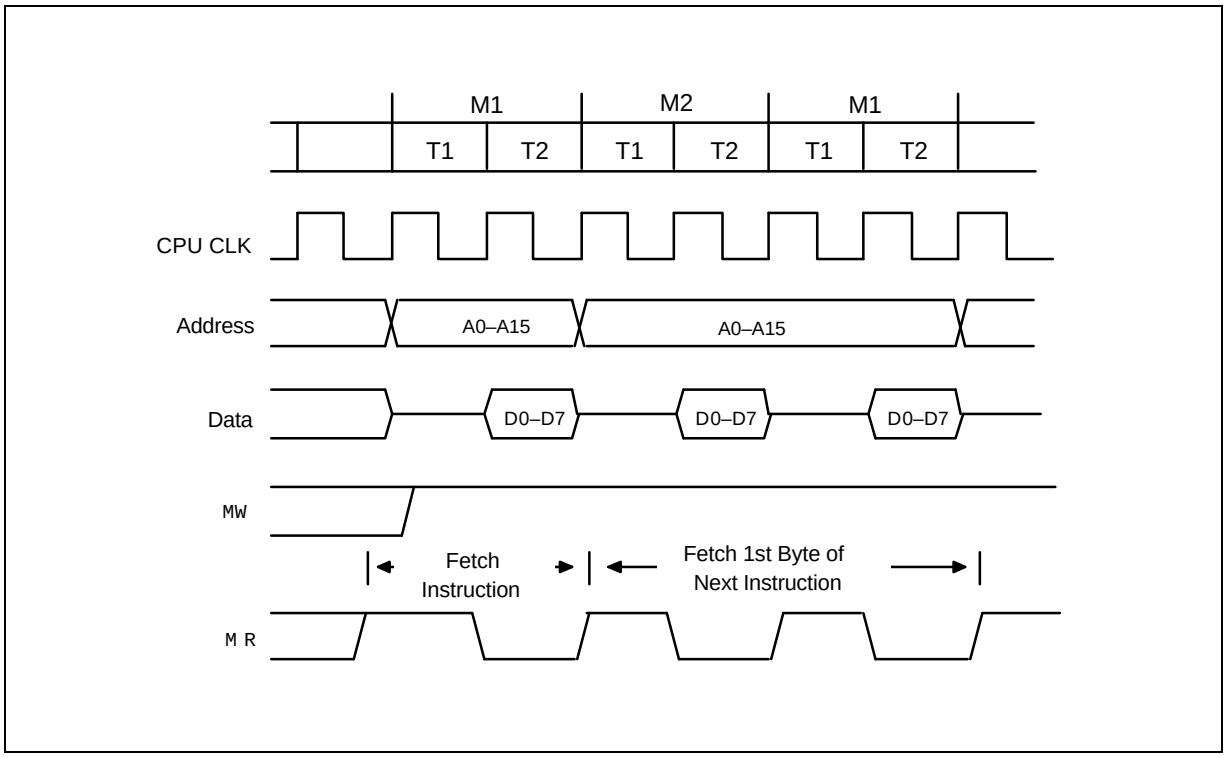


Figure 15-10. External Bus Timing Diagram for 1-Byte Fetch Instructions

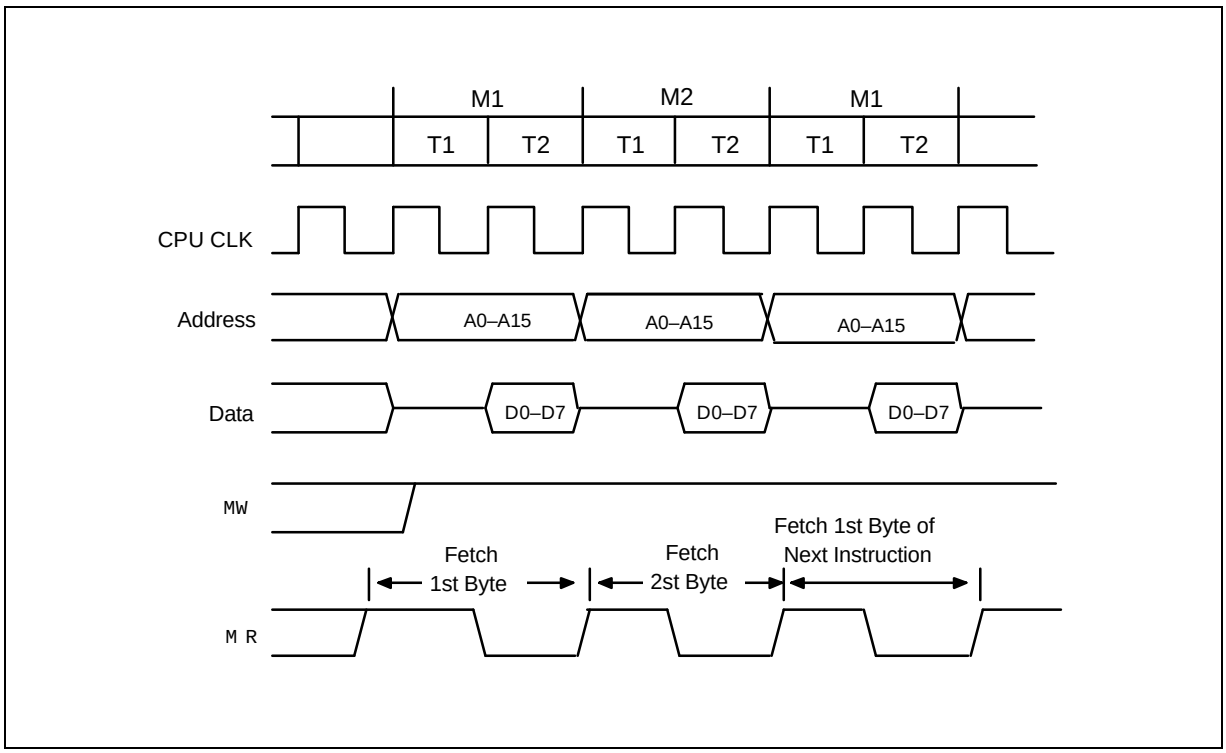


Figure 15-11. External Bus Timing Diagram for 2-Byte Fetch Instructions

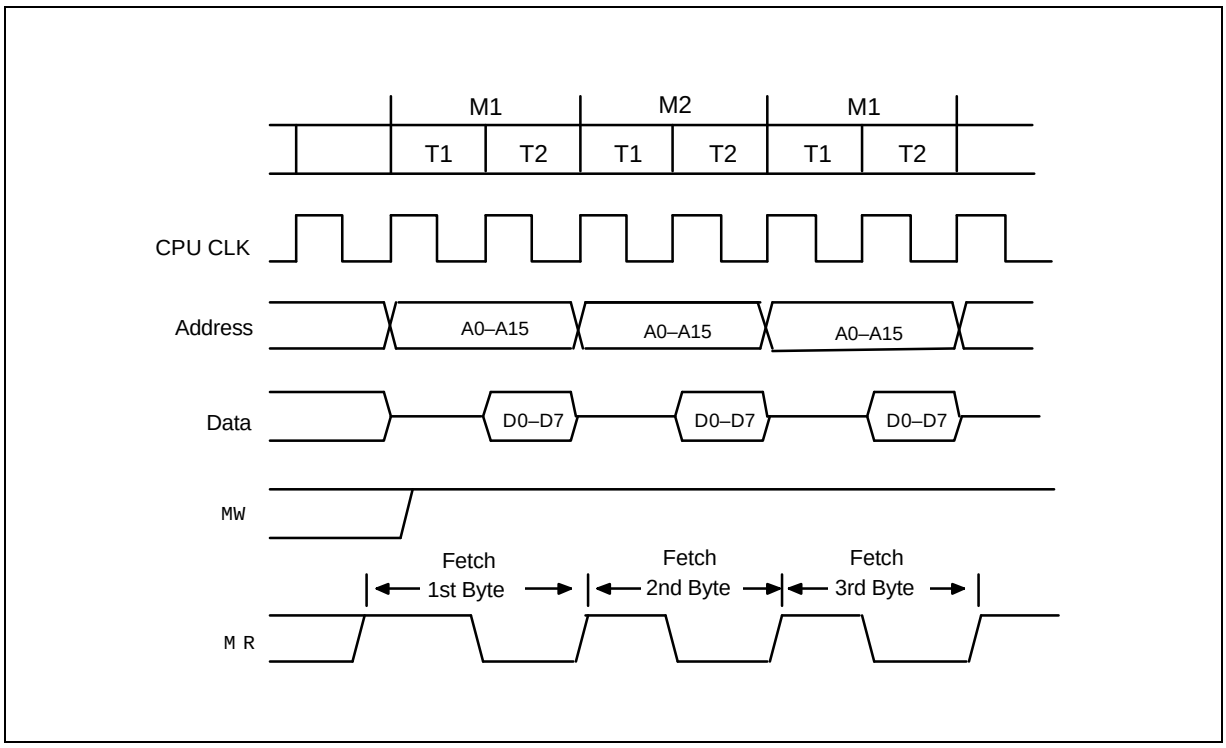


Figure 15-12. External Bus Timing Diagram for 3-Byte Fetch Instructions

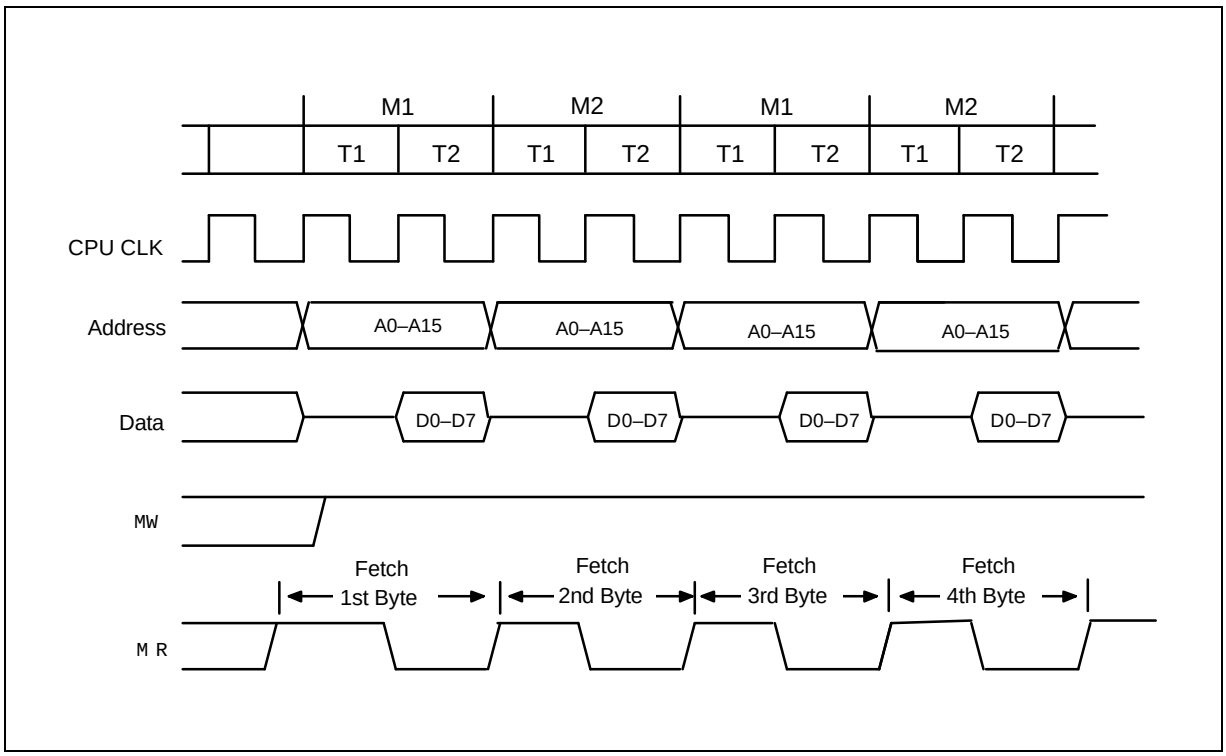


Figure 15-13. External Bus Timing Diagram for 4-Byte Fetch Instructions

NOTES

16 EMBEDDED CHIP SELECTION

OVERVIEW

The KS88C4504 has the embedded chip selection function to minimize the glue logic of address decoding. You can make chip selection signals by setting the address of chip onto a certain register (CSDATAx) and enabling this function. When the microcontroller communicates with an external chip set using the external interface command (LDE), chip selection signal is made from the instruction operand. For example,

```
LDE     8000h,#0F3h     ;  8000h is address of external chip and 0F3h is data or
                           internal address of  external chip.
```

To decode 8000h, we need an extra-decoding logic on System board.

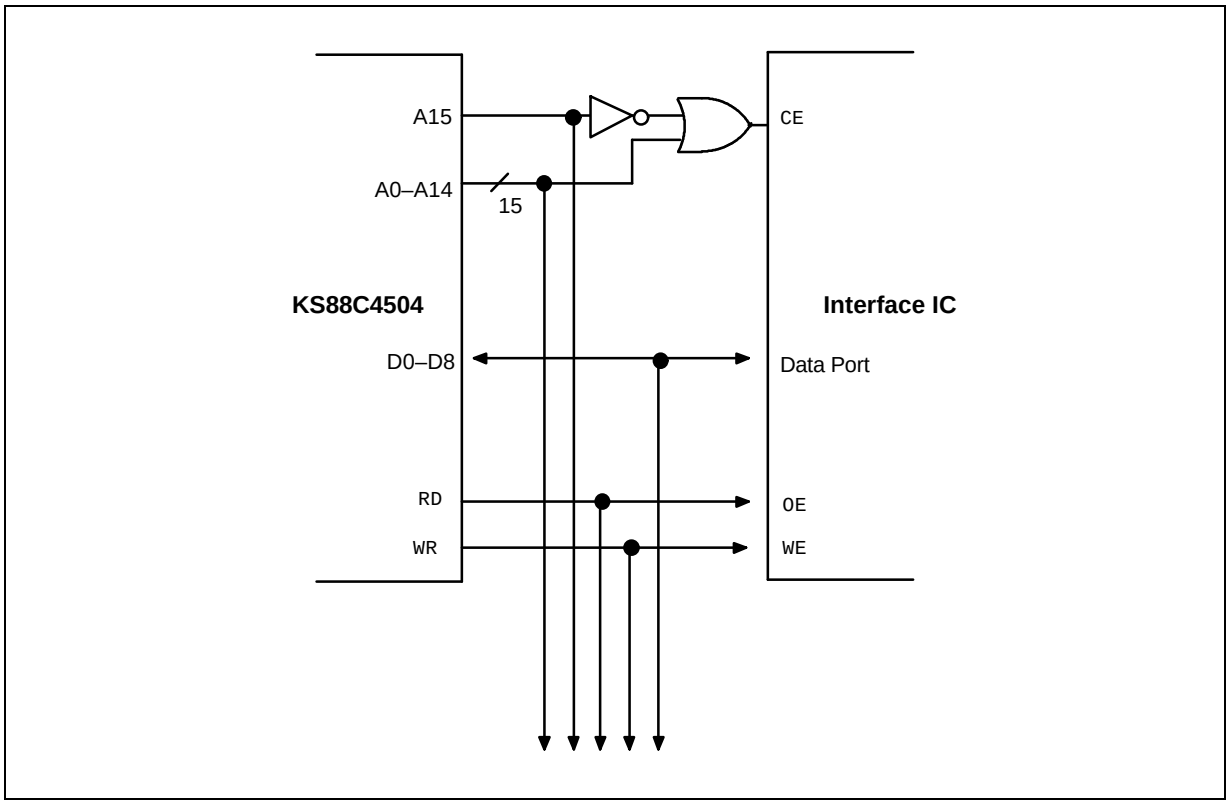


Figure 16-1. External Interface with Addressing Decoding method (old)

Using the Embedded chip selection function of the KS88C4504, you do not need an inverter.

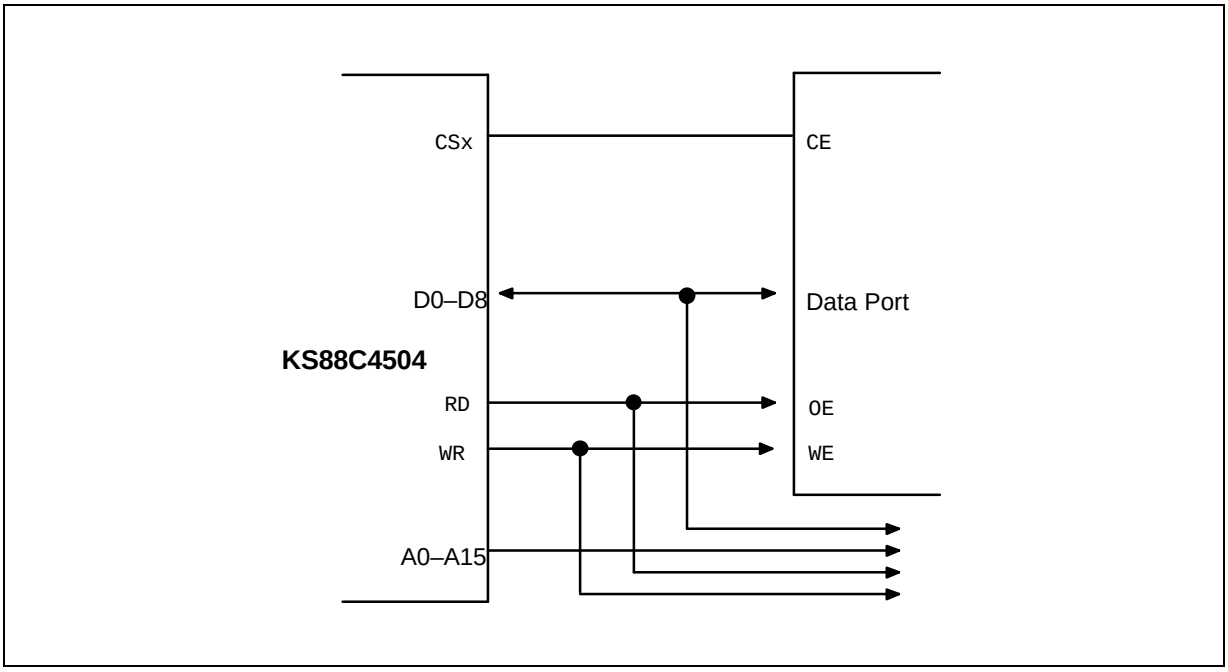


Figure 16-2. External Interface with Embedded Chip Selection Function

The Embedded chip selection block consists of a 16-bit comparator, the output of this block is controlled by P4CONH.

If there is an address identical to the one specified by CSDATAx on the address bus, the CSx pin goes low. When the address is not equal, CSx pin remains always high. (The chip selection function is not selected.)

Because the lower 8 bits of CSDATA are always zero, it is better for the lower 8-bit address of the external device to be zero.

NOTES

1. When CSx (chip selection) is active low, DM (Data memory selection) is not active.
2. *Although CS function is disable by setting P4CONH, the access to external data memory at the same address as CSDATA cannot make DM active low. Therefore you should set unused CSDATAx registers into unused data memory address.*

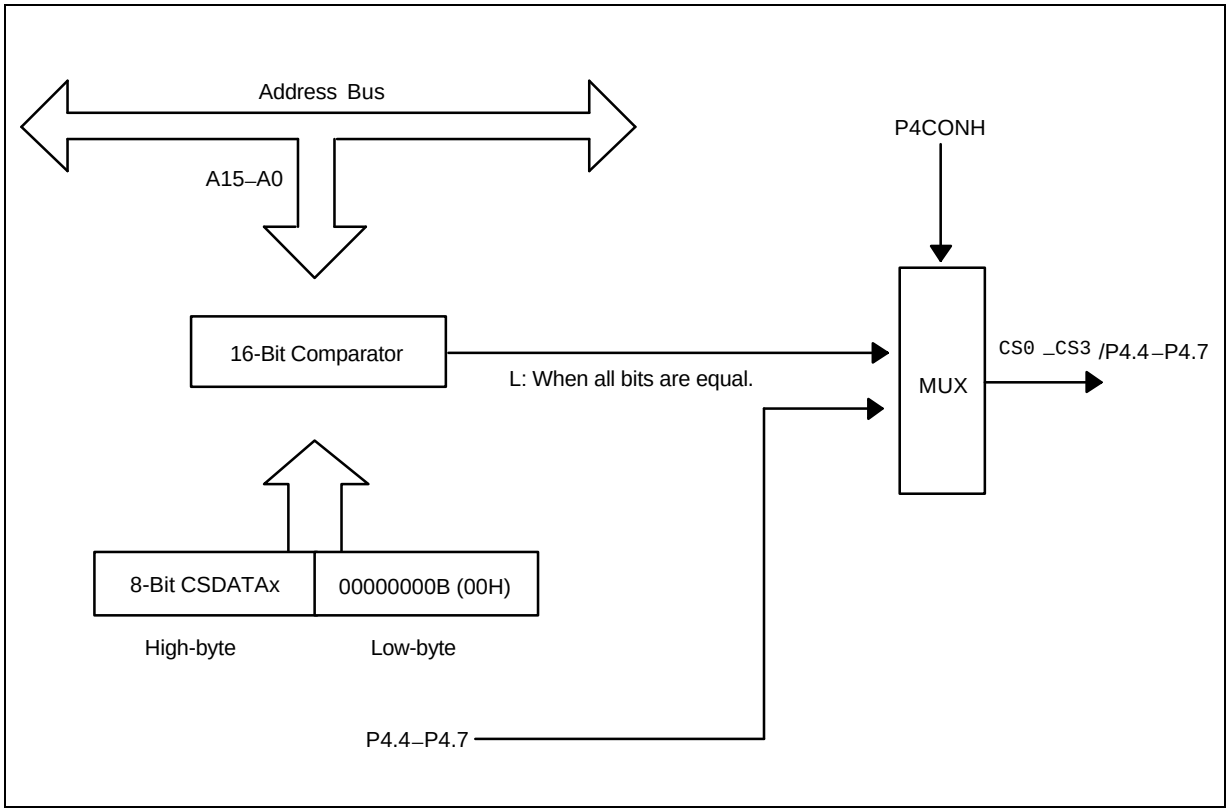
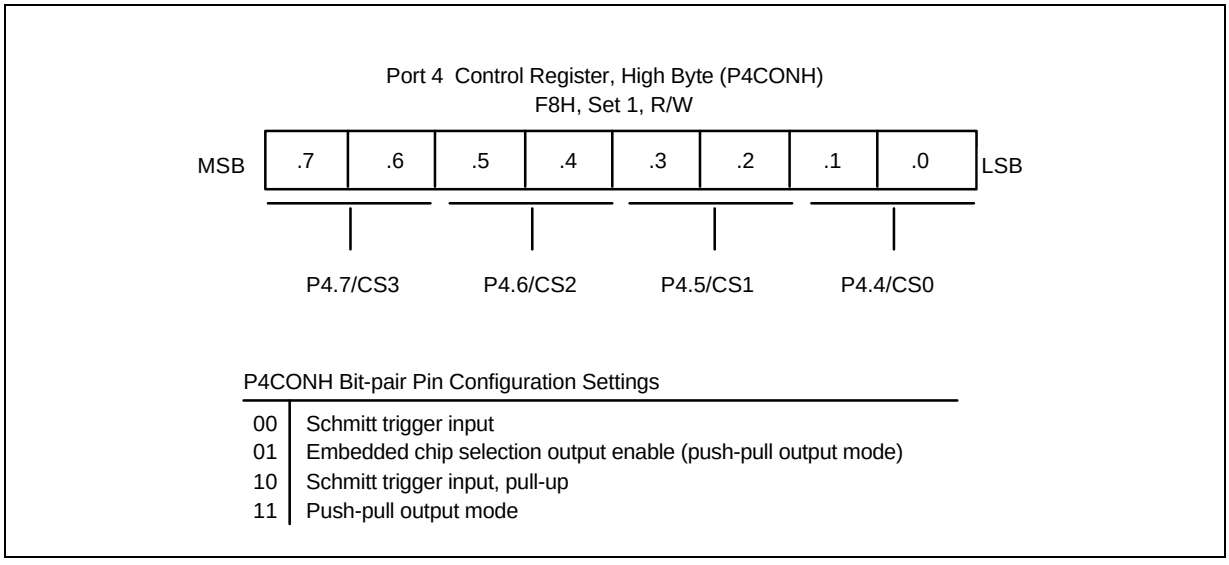


Figure 16-3. Embedded Chip Selection Functional Diagram



P4CONH Bit-pair Pin Configuration Settings

00	Schmitt trigger input
01	Embedded chip selection output enable (push-pull output mode)
10	Schmitt trigger input, pull-up
11	Push-pull output mode

Figure 16-4. Port 4 Control Register

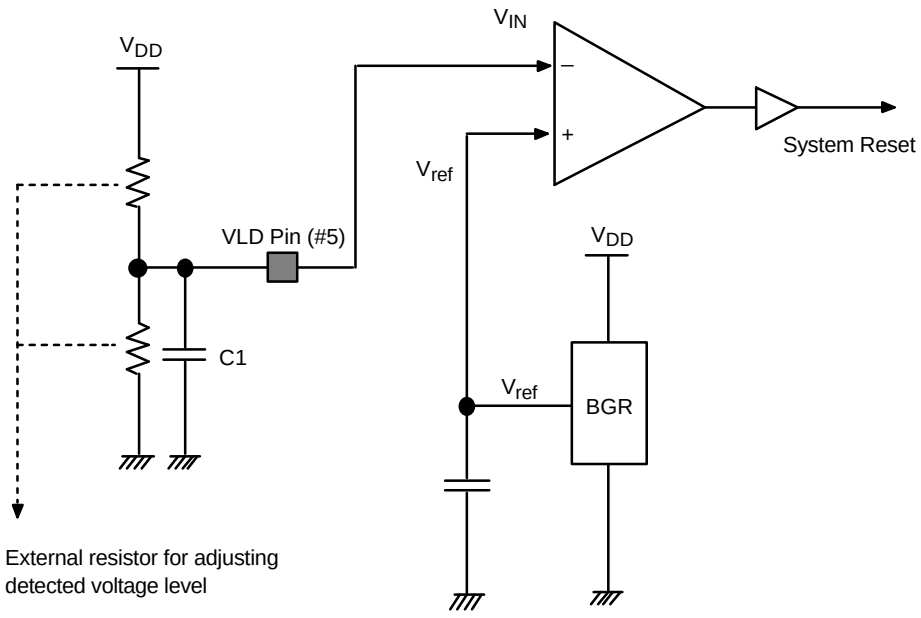
NOTES

17 VOLTAGE LEVEL DETECTOR

OVERVIEW

The KS88C4504/P4504 microcontroller has a built-in VLD (Voltage Level Detector) circuit which allows detection of power voltage drop of external input level to prevent a MCU from malfunctioning in unstable MCU power level. Detection voltage level can be adjusted by external divided resistors ratio on VLD pin (#5).

If VLD pin level is lower than the reference voltage, a system reset will be issued. If the VLD pin level is higher, the CPU will enter the normal operating mode.



NOTES:

1. The target level of voltage detection is 3.5 V at V_{DD} = 5 V.
2. VLD block is always enabled.
3. About the current consumption, refer to the electrical data.
4. C1 = 1 nF

Figure 17-1. Voltage Level Detection Circuit

NOTES

18 ELECTRICAL DATA

OVERVIEW

In this section, KS88C4504 electrical characteristics are presented in tables and graphs. The information is arranged in the following order:

— Absolute maximum ratings
— D.C. electrical characteristics
— A.C. electrical characteristics
— I/O capacitance
— Oscillation characteristics
— Oscillation stabilization time

Table 18-1. Absolute Maximum Ratings

(T_A = 25°C)

Parameter	Symbol	Conditions	Rating	Unit
Supply voltage	V_{DD}		– 0.3 to + 6.5	V
Input voltage	V_I	All ports (in input mode)	– 0.3 to V_{DD} + 0.3	
Output voltage	V_O	All ports (in output mode)	– 0.3 to V_{DD} + 0.3	V
Output current high	I_{OH}	One I/O pin active	– 18	mA
		All I/O pins active	– 60	
Output current low	I_{OL}	One I/O pin active	+ 30	mA
		Total pin current for port	+ 100	
Operating temperature	T_A		– 40 to + 85	°C
Storage temperature	T_{STG}		– 65 to + 150	°C

Table 18-2. D.C. Electrical Characteristics

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

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Operating voltage	V_{DD}	F_{OSC} = 25 MHz (instruction clock = 6.25 MHz)	4.5	–	5.5	V
		F_{OSC} = 12 MHz (instruction clock = 3 MHz)	2.7	–	5.5	
Input high voltage	V_{IH1}	All input pins except V_{IH2}	0.51 V_{DD}	–	V_{DD}	V
	V_{IH2}	X_{IN}	V_{DD} – 0.5			
Input low voltage	V_{IL1}	All input pins except V_{IL2}	–	–	0.2 V_{DD}	V
	V_{IL2}	X_{IN}			0.4	
Output high voltage	V_{OH}	V_{DD}= 5 V I_{OH} = – 1 mA	V_{DD} – 1.0	–	–	V
		I_{OH} = – 100 uA	V_{DD} – 0.5	–	–	
Output low voltage	V_{OL1}	V_{DD} = 5 V I_{OL} = 2 mA All output pins except port 2	–	–	0.4	V
	V_{OL2}	V_{DD} = 5 V I_{OL} = 15 mA, port 2	–	0.5	1.0	
Input high leakage current	I_{LIH1}	V_{IN} = V_{DD} All input pins except X_{IN}	–	–	3	µA
	I_{LIH2}	V_{IN} = V_{DD} X_{IN}			20	
Input low leakage current	I_{LIL1}	V_{IN} = 0 V All input pins except X_{IN} and RESET	–	–	– 3	
	I_{LIL2}	V_{IN} = 0 V, X_{IN}, RESET			– 20	
Output high leakage current	I_{LOH}	V_{OUT} = V_{DD} All I/O pins and output pins	–	–	5	
Output low leakage current	I_{LOL}	V_{OUT} = 0 V All I/O pins and output pins	–	– 0	– 5	
Pull-up and pull-down resistor	R_{L1}	V_{IN} = 0 V; V_{DD} = 5 V ± 10% Ports 0–5, TA = 25 °C	30	46	80	KΩ
	R_{L2}	V_{IN} = 0 V; V_{DD} = 5 V ± 10% TA = 25 °C, RESET only	120	240	320	

Table 18-2. D.C. Electrical Characteristics (Continued)

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

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Supply current (note)	I_{DD1}	V_{DD} = 5 V ± 10% 20 MHz oscillation		20	40	mA
		V_{DD} = 2.7 V 12 MHz oscillation		7	14	
	I_{DD2}	Idle mode; V_{DD} = 5 V ± 10% 20 MHz oscillation		8	16	
		Idle mode; V_{DD} = 2.7 V 12 MHz oscillation		3	6	
	I_{DD3}	Stop mode; V_{DD} = 5 V ± 10% LVD enable, T_A = 25°C		110	220	µA

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

Table 18-3. A.C. Electrical Characteristics

(T_A = -40°C to + 85°C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Interrupt input high, low width (P2.0-P2.7)	t_{INTH}, t_{INTL}	VDD=5V	180	-	-	nS
RESET input low width	t_{RSL}	VDD=5V	1000	-	-	nS

NOTES:
1. The unit t_{CPU} means one CPU clock period.
2. The oscillator frequency is the same as theCPU clock frequency.

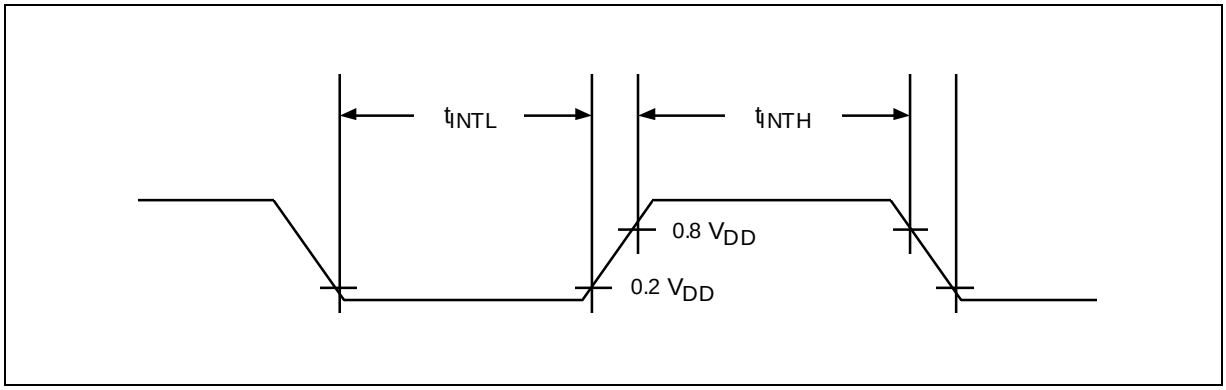


Figure 18-1. Input Timing for External Interrupts (Ports 2)

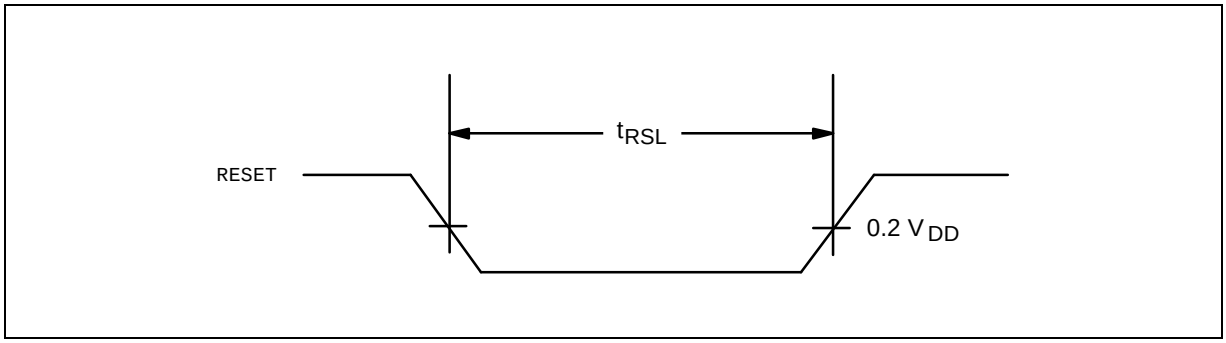


Figure 18-2. Input Timing for RESET

Table 18-4. Input/Output Capacitance

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

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

Table 18-5. 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}		2	–	5.5	V
Data retention supply current	I_{DDDR}	Stop mode, V_{DDDR} = 2.0 V	–	–	50	µA

NOTES:
1. During the oscillator stabilization wait time (t_{WAIT}), all CPU operations must be stopped.
2. Supply current does not include drawn through internal pull–up resistors and external output current loads.

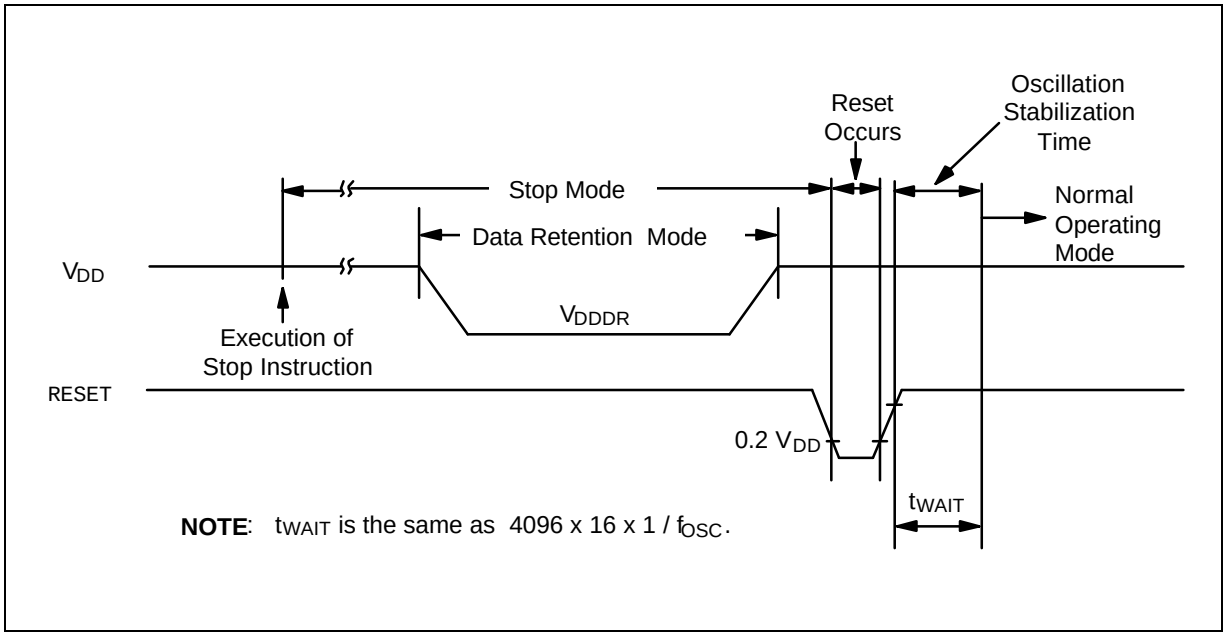


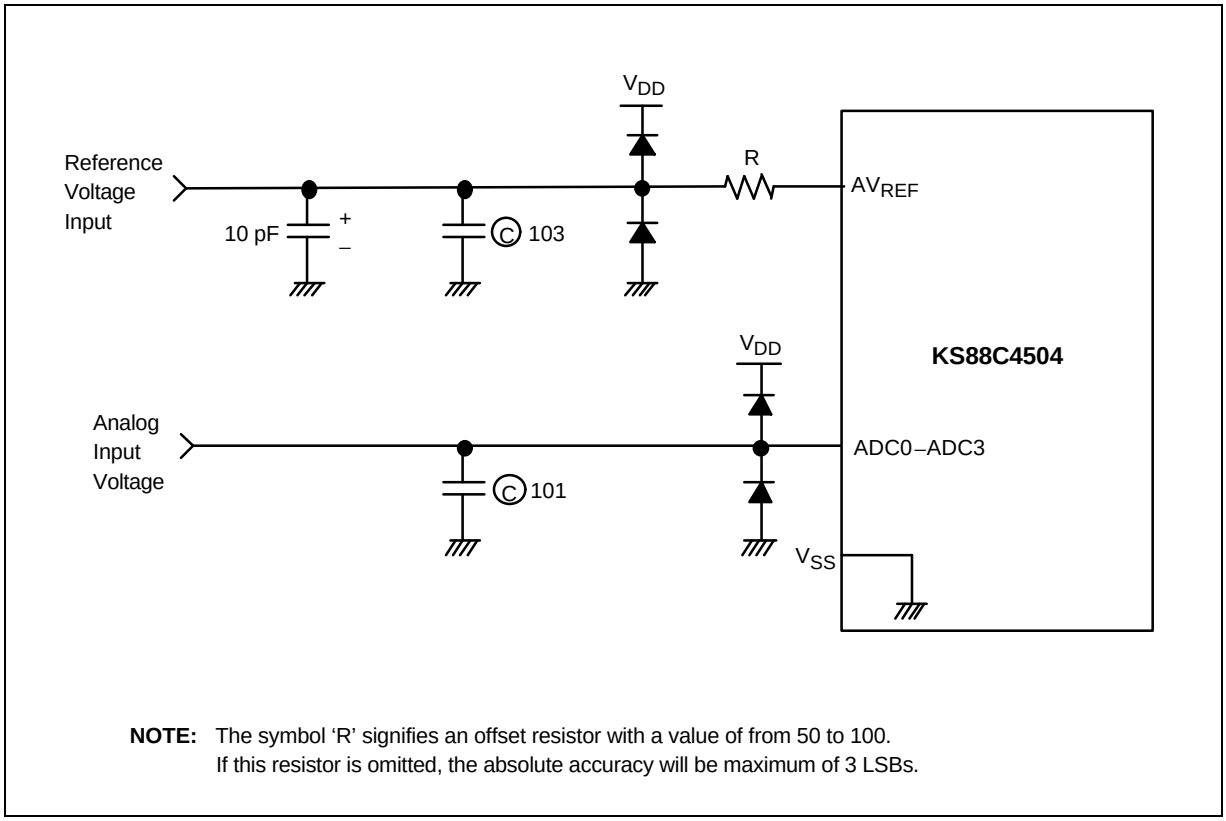
Figure 18-3. Stop Mode Release Timing Initiated by RESET

Table 18-6. A/D Converter Electrical Characteristics

(T_A = – 40°C to + 85°C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Resolution			–	8	–	bit
Total accuracy		V_{DD} = 5 V	–	–	± 2	LSB
Integral linearity error	ILE	Conversion time = 5 us		–	± 1	
Integral linearity error	DLE	AV_{REF} = 5 V		–	± 1	
Offset error of top	EOT	AV_{SS} = 0 V		± 1	± 2	
Offset error of bottom	EOB			± 0.5	± 2	
Conversion time (1)	t_{CON}		17	–	170	µs
Analog input voltage	V_{IAN}		AVss	–	AVref	V
Analog input impedance	R_{AN}	–	2	1000	–	MΩ
Analog reference voltage	AV_{REF}	–	2.5	–	V_{DD}	V
Analog ground	AV_{SS}	–	V_{SS}	–	V_{SS}+ 0.3	V
Analog input current	I_{ADIN}	AV_{REF} = V_{DD} = 5V	–	–	10	uA
Analog block current(2)	I_{ADC}	AV_{REF} = V_{DD} = 5V		1	3	mA
		AV_{REF} = V_{DD} = 3V		0.5	1.5	mA
		AV_{REF} = V_{DD} = 5V When Power Down mode		100	500	nA

NOTES:
1. 'Conversion time' is the time required from the moment a conversion operation starts until it ends.
2. I_{ADC} is an operating current during A/D conversion.



NOTE: The symbol 'R' signifies an offset resistor with a value of from 50 to 100.
If this resistor is omitted, the absolute accuracy will be maximum of 3 LSBs.

Figure 18-4. Recommended A/D Converter Circuit for Highest Absolute Accuracy

Table 18-7. Synchronous SIO Electrical Characteristics

(T_A = – 40°C to + 85°C, V_{DD} = 4.5 V to 5.5 V, V_{SS} = 0 V, fOSC=10 MHz oscillator)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
SCK cycle time	t_{CYC}	–	200	–	–	nS
Serial clock high width	t_{SCKH}	–	60	–	–	
Serial clock low width	T_{SCKL}	–	60	–	–	
Serial output data delay time	T_{OD}	–	–	–	50	
Serial input data setup time	T_{ID}	–	40	–	–	
Serial input data hold time	T_{IH}	–	100	–	–	

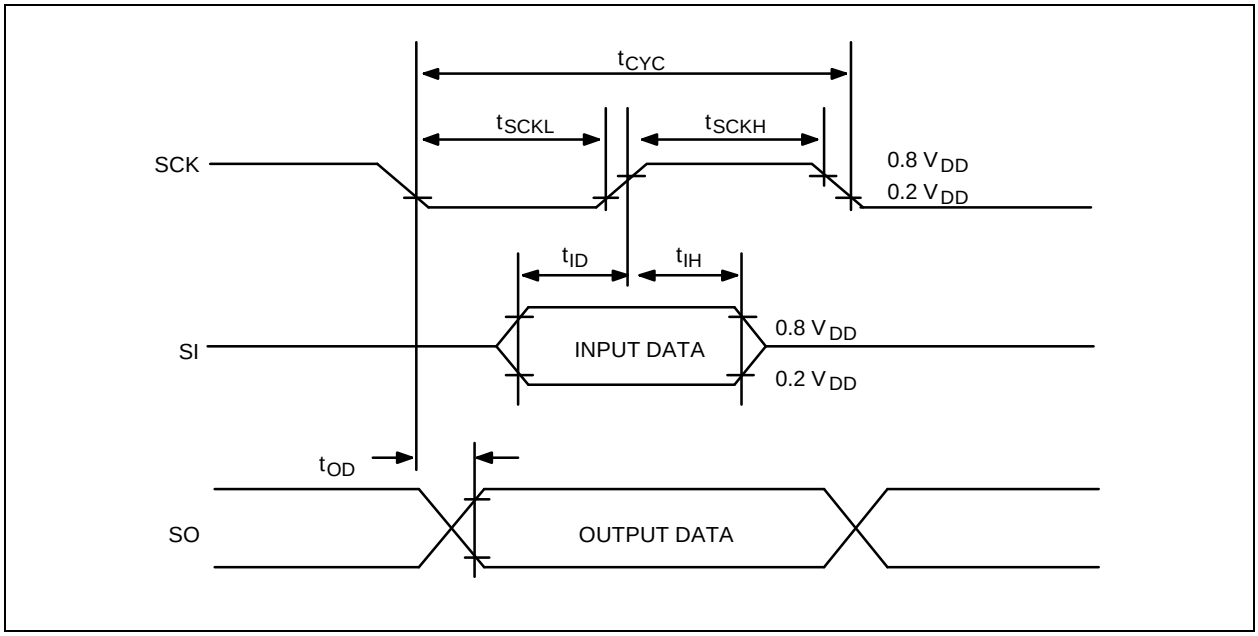


Figure 18-5. Serial Data Transfer Timing

Table 18-8. Main Oscillator Frequency (f_{OSC1})

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

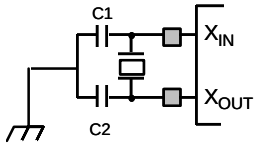
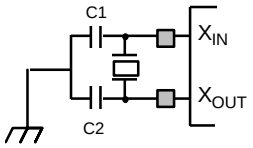
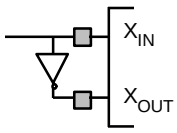
Oscillator	Clock Circuit	Test Condition	Min	Typ	Max	Unit
Crystal		CPU clock oscillation frequency	4	–	25	MHz
Ceramic		CPU clock oscillation frequency	4	–	25	MHz
External clock		X_{IN} input frequency	4	–	25	MHz

Table 18-9. Main Oscillator Clock Stabilization Time (t_{ST1})

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

Oscillator	Test Condition	Min	Typ	Max	Unit
Crystal	V_{DD} = 4.5 V to 5.5 V	–	–	10	ms
Ceramic	Stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	4	ms
External clock	X_{IN} input high and low level width (t_{XH}, t_{XL})	50	–	–	ns

NOTE: Oscillation stabilization time (tST1) is the time required for the CPU clock to return to its normal oscillation frequency after a power-on occurs, or when Stop mode is ended by a RESET signal. The RESET should therefore be held at low level until the t_{ST1} time has elapsed.

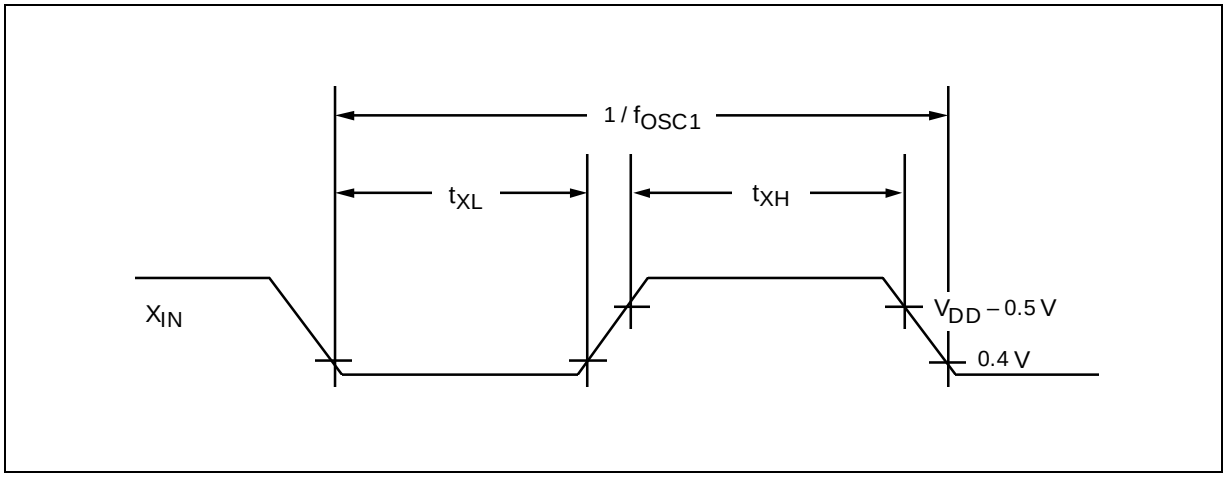
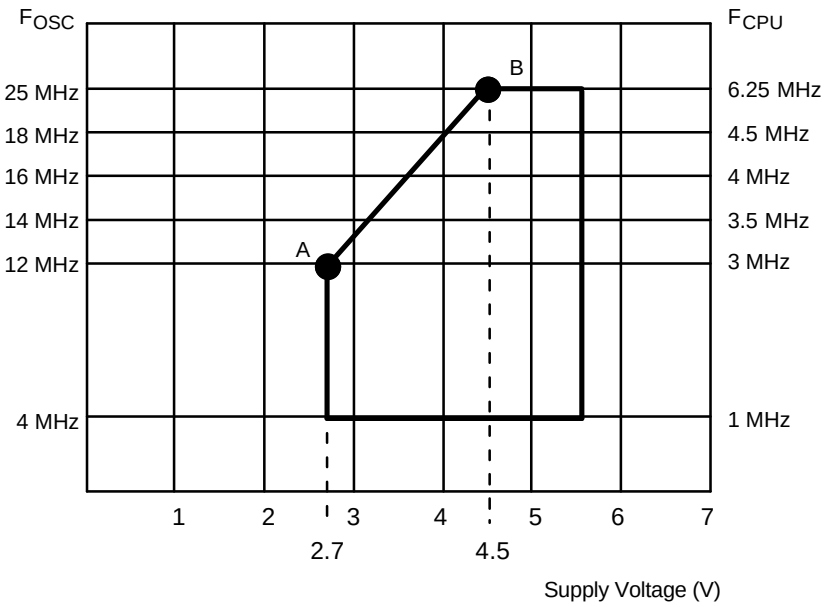


Figure 18-6. Clock Timing Measurement at X_{IN}

Table 18-11. Characteristics of Voltage Level Detect circuit

(T_A = – 40°C + 85°C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Operating Voltage of VLD	V_{DDVLD}	–	2.7	–	5.5	V
Detect Voltage	V_{VLD}	–	1.15	1.40	1.51	V
Current consumption	I_{VLD}	V_{DD} = 5.5 V	–	100	200	uA



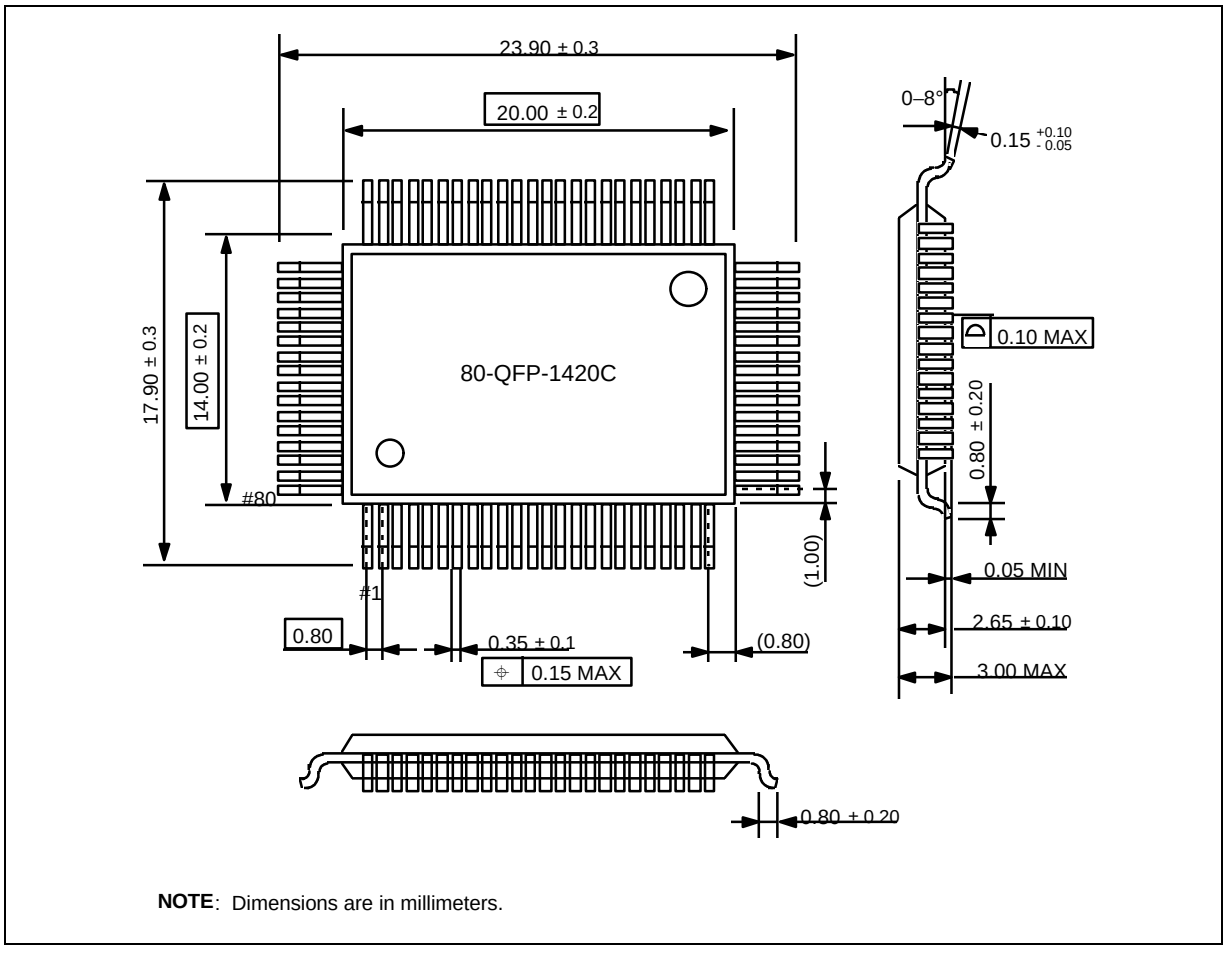
Instruction Clock = 1/4n x oscillator frequency (n = 1, 2, 8, 16)

Figure 18-7. Operating Voltage Range

19 MECHANICAL DATA

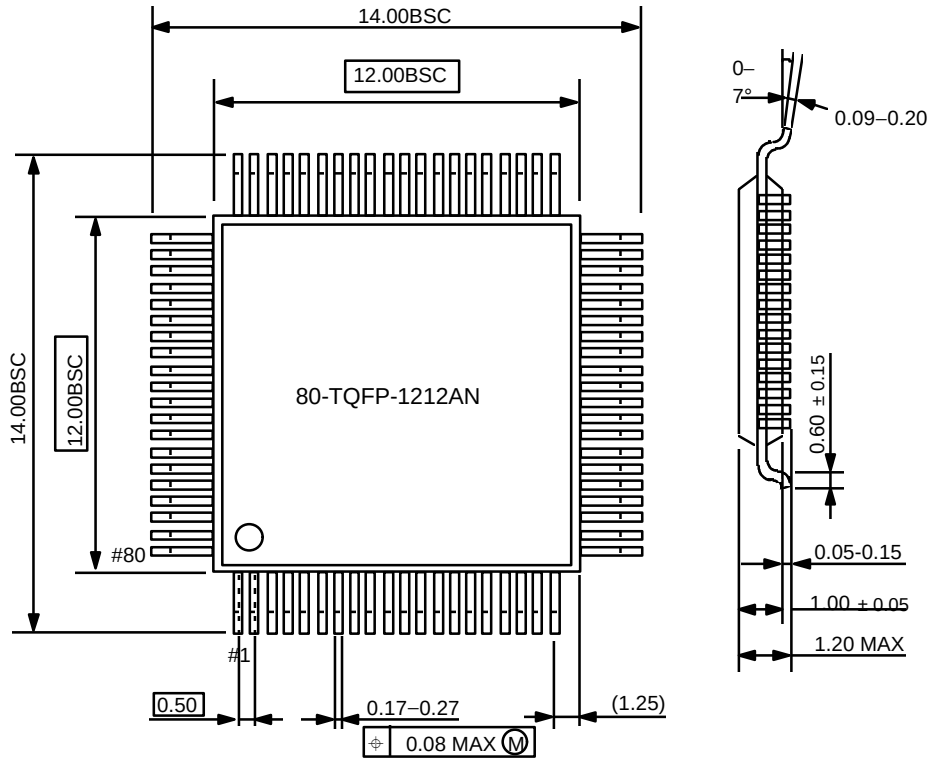
OVERVIEW

The KS88C4504 microcontroller is available in a 80-pin QFP package (80-QFP-1420C) and a 80-pin TQFP package (80-TQFP-1212AN).



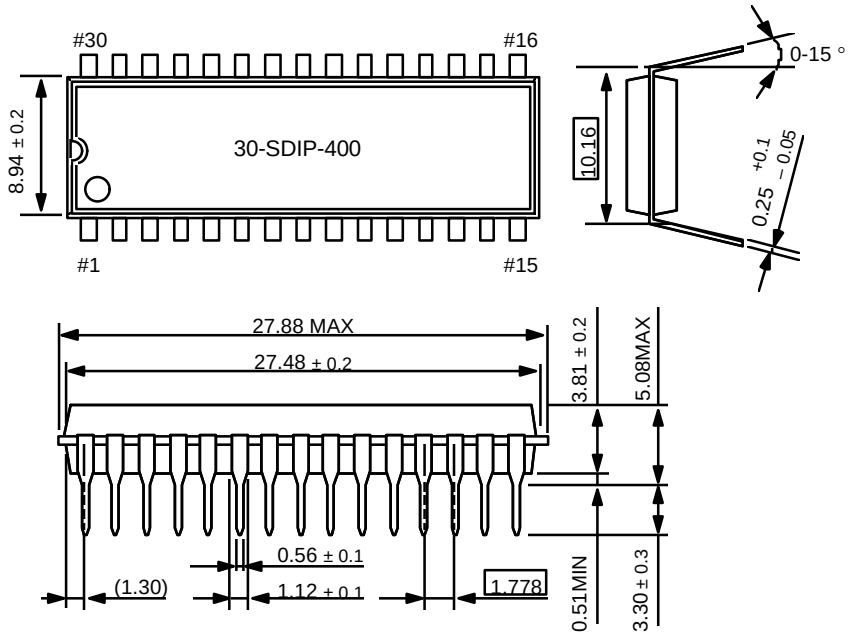
NOTE: Dimensions are in millimeters.

Figure 19-1. 80-QFP-1420C Package Dimensions



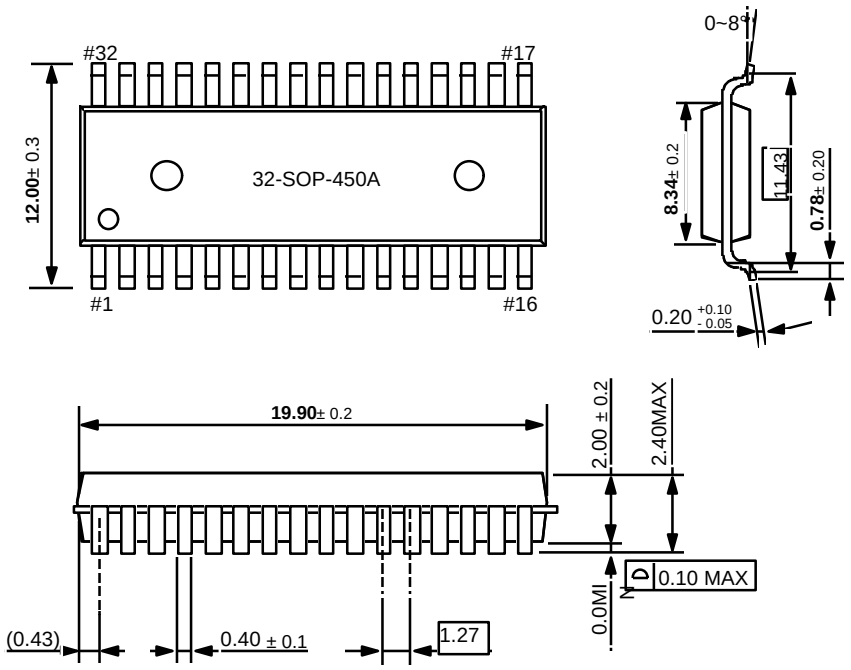
NOTE: Dimensions are in millimeters.

Figure 19-2. 80-TQFP-1212AN Package Dimensions



NOTE: Dimensions are in millimeters.

Figure 19-3. 30-SDIP-400 Package Dimensions



NOTE: Dimensions are in millimeters.

Figure 19-4. 32-SOP-450A Package Dimensions

20 KS88P4504 OTP

OVERVIEW

The KS88P4504 single-chip CMOS microcontroller is the OTP (One Time Programmable) version of the KS88C4504 microcontrollers. It has an on-chip EPROM instead of masked ROM. The EPROM is accessed by serial data format.

KS88P4504 is fully compatible with KS88C4504, both in function and in pin configuration. As it has simple programming requirements, KS88P4504 is ideal for use as an evaluation chip for the KS88C4504.

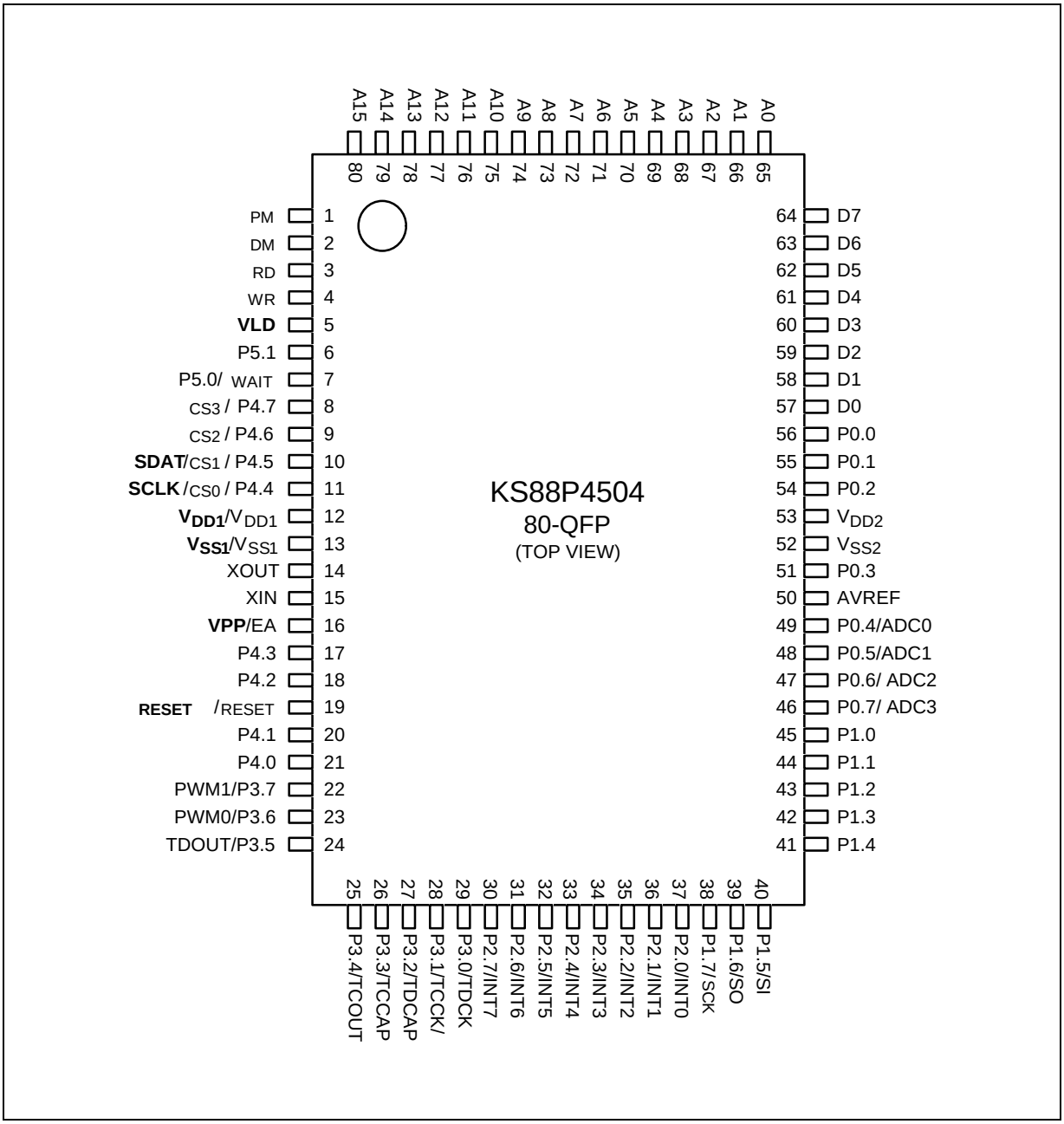


Figure 20-1. KS88P4504 Pin Assignments (80-QFP Package)

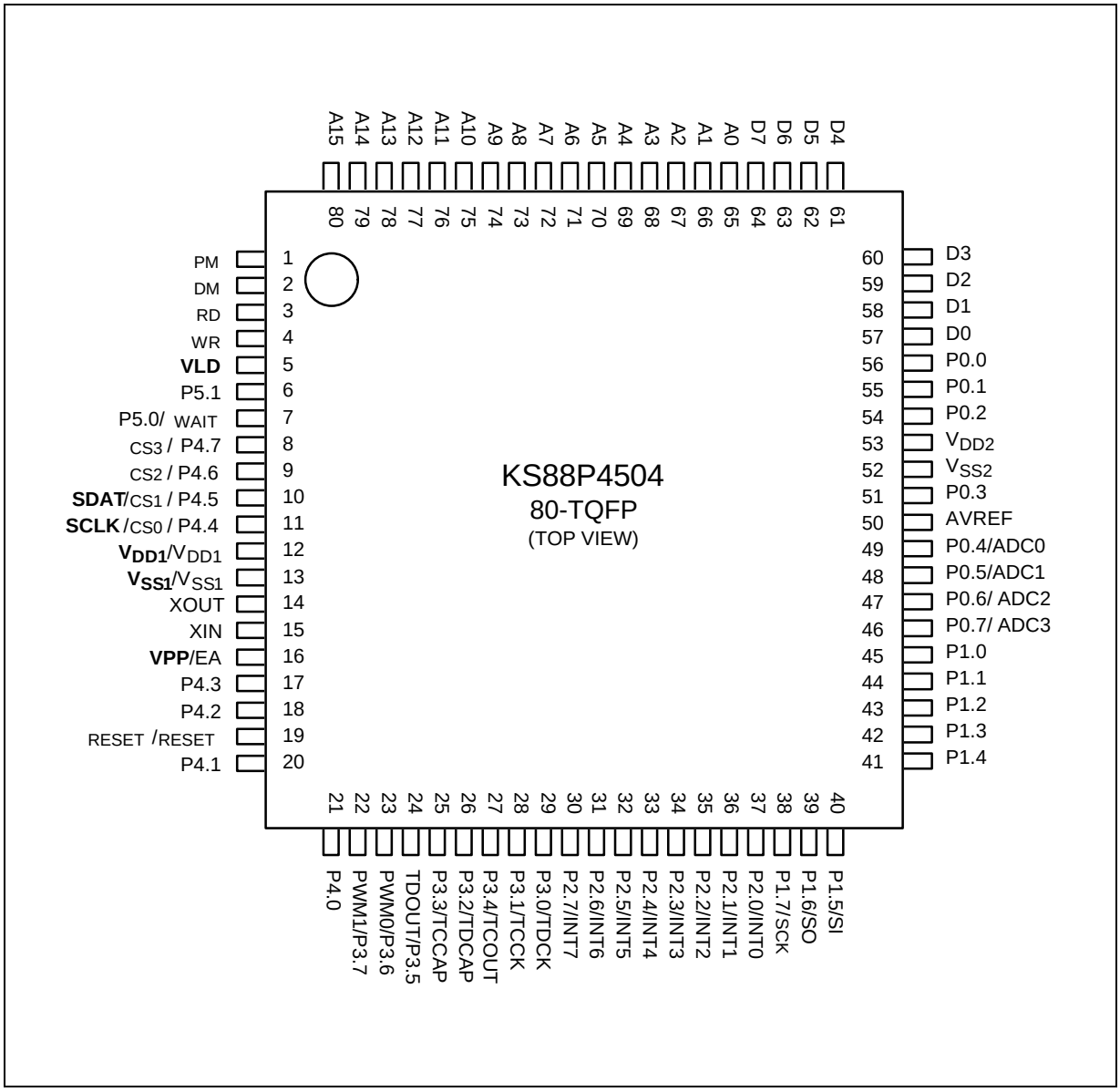


Figure 20-2. KS88P4504 Pin Assignments (80-TQFP Package)

Table 20-1. Descriptions of Pins Used to Read/Write the EPROM

Main Chip	During Programming			
Pin Name	Pin Name	Pin No.	I/O	Function
P4.5	SDAT	10	I/O	Serial Data Pin (Output when reading, Input when writing) Input and Push-pull Output Port can be assigned.
P4.4	SCLK	11	I	Serial Clock Pin (Input Only Pin)
EA	V_{PP}	16	I	EPROM Cell Writing Power Supply Pin (Indicates OTP Mode Entering) When writing 12.5V is applied and when reading 5 V is applied (Option).
RESET	RESET	19	I	Chip Initialization
V_{DD1}/V_{SS1}	V_{DD}/V_{SS}	12/13	I	Logic Power Supply Pin. V_{DD} should be tied to 5V during programming.

Table 20-2. Comparison of KS88P4504 and KS88C4504 Features

Characteristic	KS88P4504	KS88C4504
Program Memory	4 K byte EPROM	4 K bytes mask ROM
Operating Voltage (V_{DD})	2.7 V to 5.5 V	2.7 V to 5.5V
OTP Programming Mode	V_{DD} = 5 V, V_{PP} (TEST) = 12.5V	
Pin Configuration	80 QFP, 80 TQFP	80 QFP, 80 TQFP
EPROM Programmability	User Program 1 time	Programmed at the factory

OPERATING MODE CHARACTERISTICS

When 12.5 V is supplied to the V_{PP} (TEST) pin of KS88P4504, the EPROM programming mode is entered. The operating mode (read, write, or read protection) is selected according to the input signals to the pins listed in Table 20-3 below.

Table 20-3. Operating Mode Selection Criteria

V_{DD}	VPP (TEST)	REG/ MEM	ADDRESS (A15-A0)	R/W	MODE
5 V	5 V	0	0000H	1	EPROM read
	12.5 V	0	0000H	0	EPROM program
	12.5 V	0	0000H	1	EPROM verify
	12.5 V	1	0E3FH	0	EPROM read protection

NOTE: "0" means Low level; "1" means High level.

D.C. ELECTRICAL CHARACTERISTICS

Table 20-4. D.C. Electrical Characteristics

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

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Operating voltage	V_{DD}	F_{OSC} = 25 MHz (instruction clock = 6.25 MHz)	4.5	–	5.5	V
		F_{OSC} = 12 MHz (instruction clock = 3 MHz)	2.7	–	5.5	
Input high voltage	V_{IH1}	All input pins except V_{IH2}	0.51 V_{DD}	–	V_{DD}	V
	V_{IH2}	X_{IN}	V_{DD} – 0.5			
Input low voltage	V_{IL1}	All input pins except V_{IL2}	–	–	0.2 V_{DD}	V
	V_{IL2}	X_{IN}			0.4	
Output high voltage	V_{OH}	V_{DD}= 5 V I_{OH} = – 1 mA	V_{DD} – 1.0	–	–	V
		I_{OH} = – 100 uA	V_{DD} – 0.5	–	–	
Output low voltage	V_{OL1}	V_{DD} = 5 V I_{OL} = 2 mA All output pins except port 2	–	–	0.4	V
	V_{OL2}	V_{DD} = 5 V I_{OL} = 15 mA, port 2	–	0.5	1.0	
Input high leakage current	I_{LIH1}	V_{IN} = V_{DD} All input pins except X_{IN}	–	–	3	µA
	I_{LIH2}	V_{IN} = V_{DD} X_{IN}			20	
Input low leakage current	I_{LIL1}	V_{IN} = 0 V All input pins except X_{IN} and RESET	–	–	– 3	
	I_{LIL2}	V_{IN} = 0 V, X_{IN}, RESET			– 20	
Output high leakage current	I_{LOH}	V_{OUT} = V_{DD} All I/O pins and output pins	–	–	5	
Output low leakage current	I_{LOL}	V_{OUT} = 0 V All I/O pins and output pins	–	– 0	– 5	
Pull-up and pull-down resistor	R_{L1}	V_{IN} = 0 V; V_{DD} = 5 V ± 10% Ports 0–5, TA = 25 °C	30	46	80	KΩ
	R_{L2}	V_{IN} = 0 V; V_{DD} = 5 V ± 10% TA = 25 °C, RESET only	120	240	320	

Table 20-4. D.C. Electrical Characteristics (Continued)

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

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Supply current (note)	I_{DD1}	V_{DD} = 5 V ± 10% 20 MHz oscillation	–	20	40	mA
		V_{DD} = 2.7 V 12 MHz oscillation		7	14	
	I_{DD2}	Idle mode; V_{DD} = 5 V ± 10% 20 MHz oscillation		8	16	
		Idle mode; V_{DD} = 2.7 V 12 MHz oscillation		3	6	
	I_{DD3}	Stop mode; V_{DD} = 5 V ± 10%, LVD enable		110	220	µA

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

21 DEVELOPMENT TOOLS

OVERVIEW

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

SHINE

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

SAMA ASSEMBLER

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

SASM88

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

HEX2ROM

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

TARGET BOARDS

Target boards are available for all KS88-series microcontrollers. All the required target system cables and adapters are included with the device-specific target board.

OTPs

One time programmable microcontroller (OTP) for the KS88C4504 microcontroller and OTP programmer (Gang) are now available.

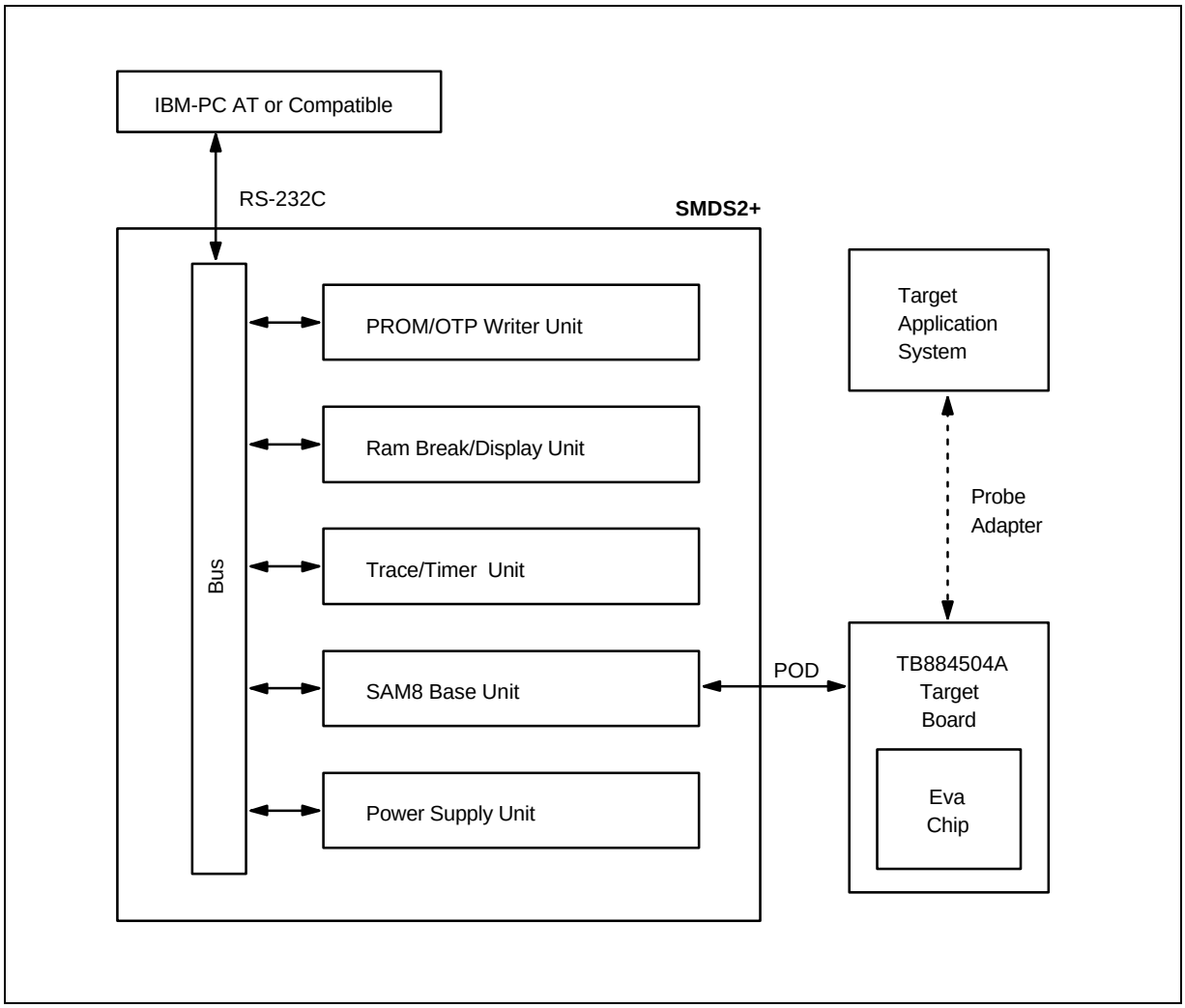


Figure 21-1. SMDS Product Configuration (SMDS2+)

TB884504A TARGET BOARD

The TB885404A target board is used for the KS88C4504/P4504 microcontroller. It is supported by the SMDS2+ development system.

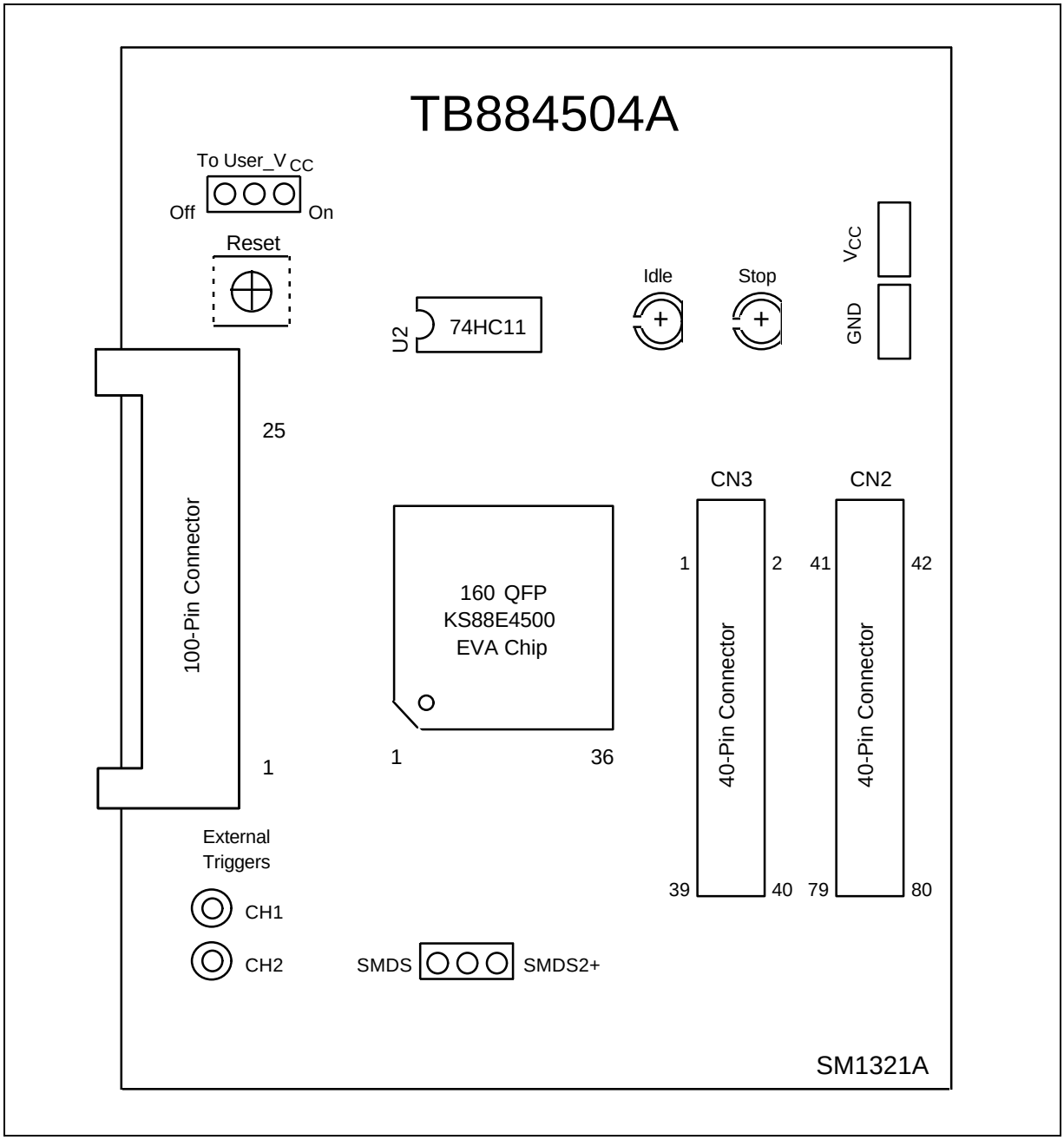
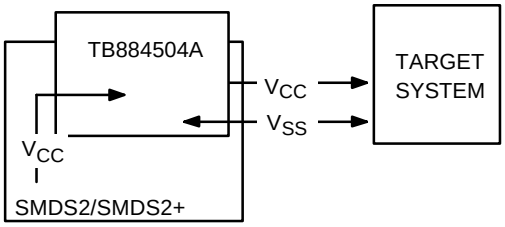
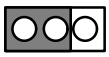
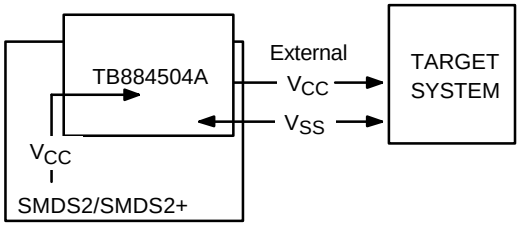


Figure 21-2. TB884504A Target Board Configuration

Table 21-1. Power Selection Settings for TB884504A

'To User_V_{CC}' Settings	Operating Mode	Comments
To User_V_{CC} Off  On		SMDS2/SMDS2+ supply V_{CC} to the target board (evaluation chip) and the target system.
To User_V_{CC} Off  On		The SMDS2/SMDS2+ supply V_{CC} only to the target board (evaluation chip). The target system must have its own power supply.

SMDS2+ Selection (SAM8)

In order to write data into program memory that is available in SMDS2+, the target board should be selected for SMDS2+ through a switch as follows. Otherwise, the program memory writing function is not available.

Table 21-2. The SMDS2+ Tool Selection Setting

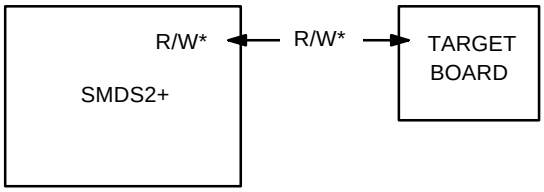
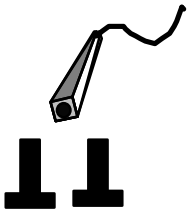
'SW1' Setting	Operating Mode
SMDS2  SMDS2+	

Table 21-3. Using Single Header Pins as the Input Path for External Trigger Sources

Target Board Part	Comments
External Triggers CH1 CH2	 Connector from external trigger sources of the application system You can connect an external trigger source to one of the two external trigger channels (CH1 or CH2) for the SMDS2+ breakpoint and trace functions.

IDLE LED

This LED is ON when the evaluation chip (KS88E4500) is in idle mode.

STOP LED

This LED is ON when the evaluation chip (KS88E4500) is in stop mode.

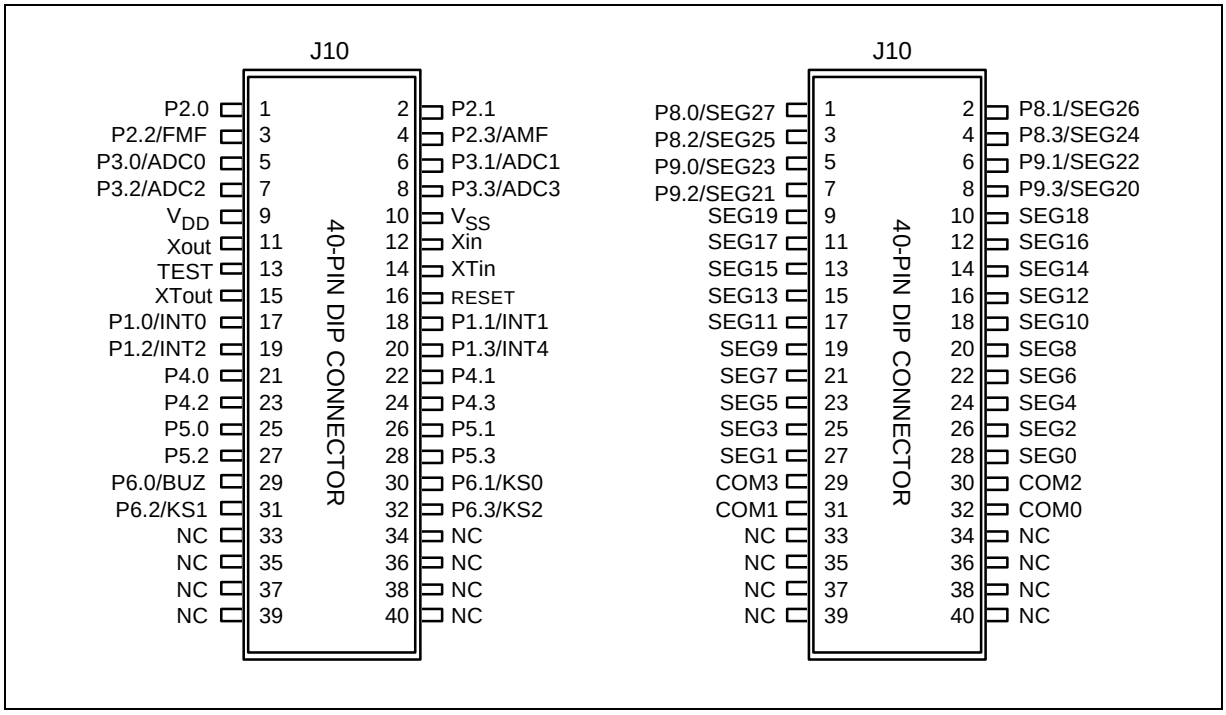


Figure 21-3. 40-Pin Connector for TB884504A

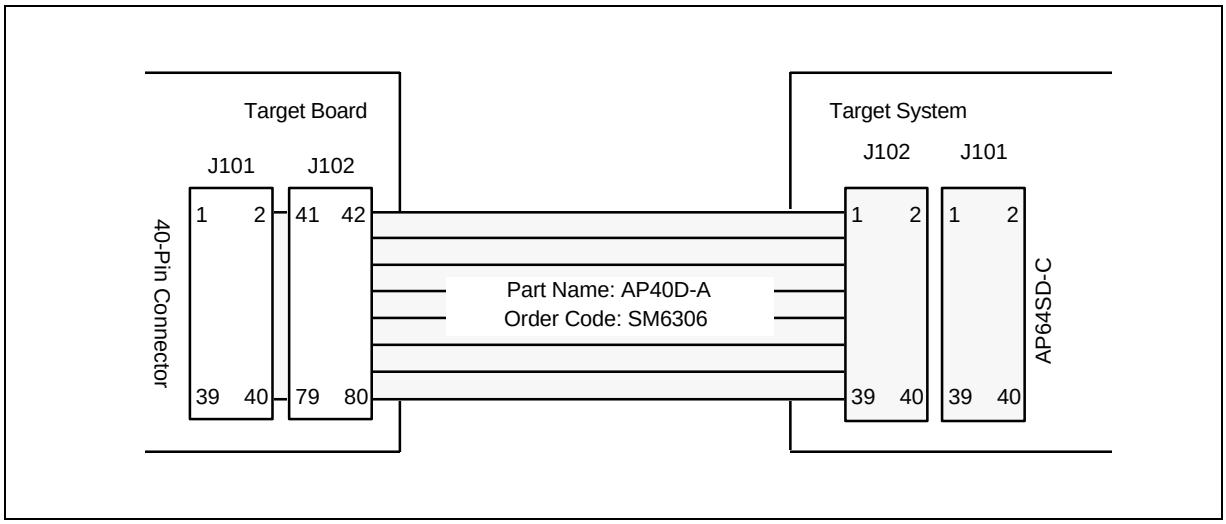


Figure 21-4. TB884504A Adapter Cable for 80-QFP Package