Product Overview

Address Spaces

Addressing Modes

Control Registers

Interrupt Structure

SAM87RI Instruction Set

1 PRODUCT OVERVIEW

SAM87RI PRODUCT FAMILY

Samsung's SAM87RI 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.

A dual address/data bus architecture and a large number of bit- or nibble-configurable I/O ports provide a flexible programming environment for applications with varied memory and I/O requirements. Timer/counters with selectable operating modes are included to support real-time operations. Many SAM87RI microcontrollers have an external interface that provides access to external memory and other peripheral devices.

KS86C6004/C6008/P6008 MICROCONTROLLER

The KS86C6004/C6008/P6008 single-chip 8-bit microcontroller is fabricated using an advanced CMOS process. It is built around the powerful SAM87RI CPU core.

Stop and Idle power-down modes were implemented to reduce power consumption. To increase on-chip register space, the size of the internal register file was logically expanded. The KS86C6004 has 4 K bytes of program memory on-chip and KS86C6008 has 8 K bytes.

Using the SAM87RI design approach, the following peripherals were integrated with the SAM87RI core:

— Five configurable I/O ports (32 pins)

— 12 bit-programmable pins for external interrupts

— 8-bit timer/counter with three operating modes

— Low speed USB function

The KS86C6004/C6008/P6008 is a versatile microcontroller that can be used in a wide range of low speed USB support general purpose applications. It is especially suitable for use as a keyboard controller and is available in a 42-pin SDIP and a 44-pin QFP package.

OTP

The KS86C6004/C6008 microcontroller is also available in OTP (One Time Programmable) version, KS86P6008. KS86P6008 microcontroller has an on-chip 8-Kbyte one-time-programmable EPROM instead of masked ROM. The KS86P6008 is comparable to KS86C6004/C6008, both in function and in pin configuration.

FEATURES

CPU

- SAM87RI CPU core

Memory

- 8-Kbyte internal program memory (ROM)
- 208-byte RAM

Instruction Set

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

Instruction Execution Time

- 1.0 µs at 6 MHz f_{OSC}

Interrupts

- 25 interrupt sources with one vector, each source has its pending bit
- One level, one vector interrupt structure

Oscillation Circuit

- 6 MHz crystal/ceramic oscillator
- External clock source (6 MHz)

General I/O

- Bit programmable five I/O ports (32 pins total)

Timer/Counter

- One 8-bit basic timer for watchdog function and programmable oscillation stabilization interval generation function
- One 8-bit timer/counter with Compare/Overflow

USB Serial Bus

- Compatible to USB low speed (1.5 Mbps) device 1.0 specification.
- Serial bus interface engine (SIE)
  - Packet decoding/generation
  - CRC generation and checking
  - NRZI encoding/decoding and bit-stuffing
- 8 bytes each receive/transmit USB buffer

Operating Temperature Range

- – 40°C to + 85°C

Operating Voltage Range

- 4.5 V to 5.5 V

Package Types

- 42-pin SDIP
- 44-pin QFP

BLOCK DIAGRAM

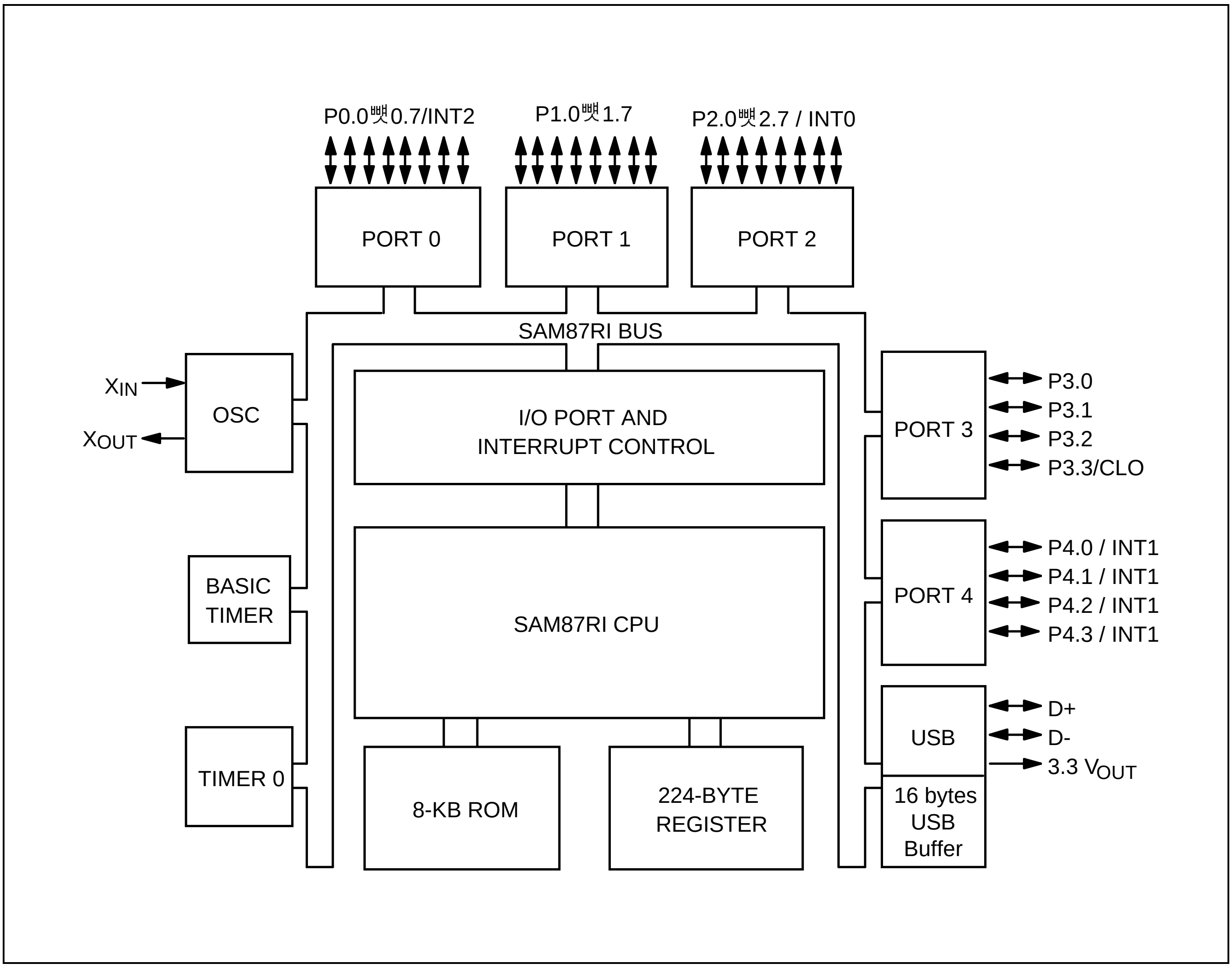


Figure 1-1. Block Diagram

PIN ASSIGNMENTS

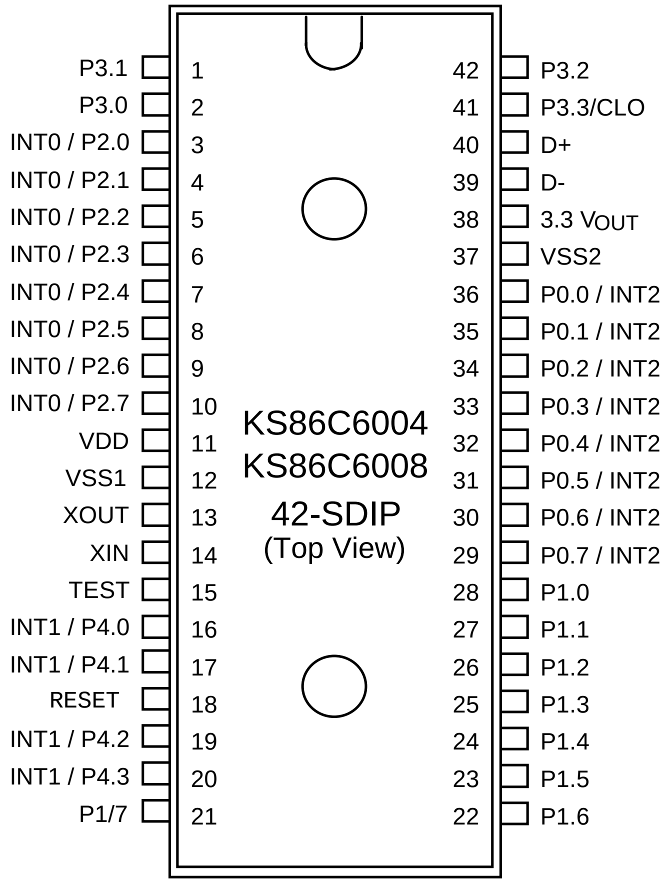


Figure 1-2. Pin Assignment Diagram (42-Pin SDIP Package)

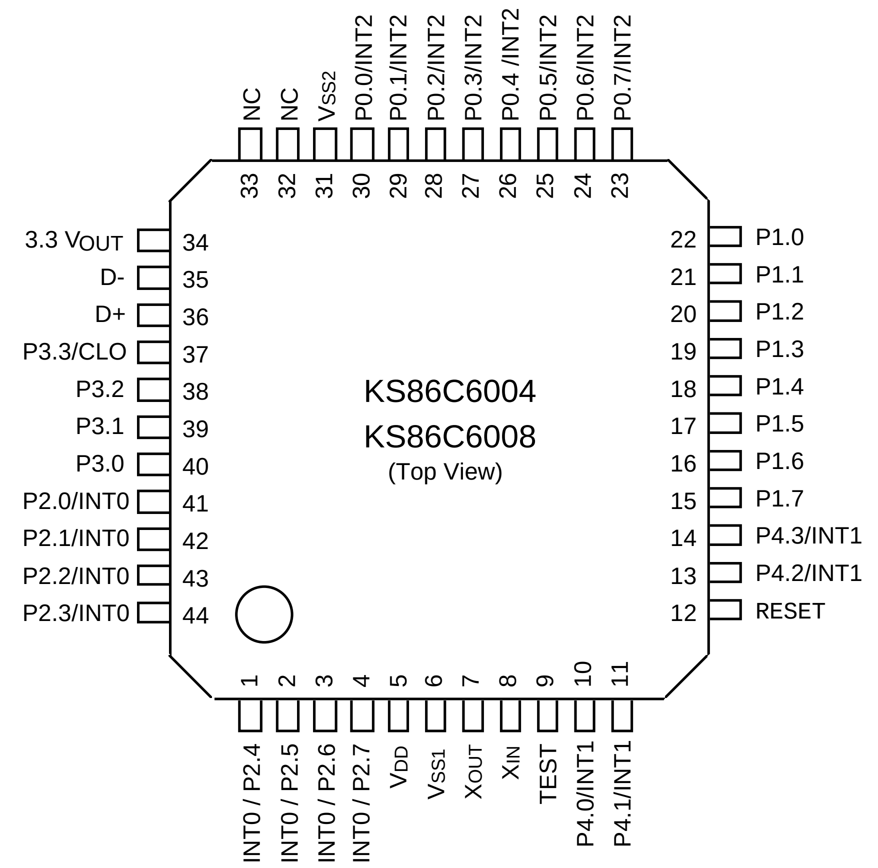


Figure 1-3. Pin Assignment Diagram (44-Pin QFP Package)

PIN DESCRIPTIONS

Table 1-1. KS86C6004/C6008/P6008 Pin Descriptions

Pin Names	Pin Type	Pin Description	Circuit Number	Pin Numbers	Share Pins
P0.0-P0.7	I/O	Bit-programmable I/O port for Schmitt trigger input or open-drain output. Port0 can be individually configured as external interrupt inputs. Pull-up resistors are assignable by software.	B	36–29 (30–23)	INT2
P1.0-P1.7	I/O	Bit-programmable I/O port for Schmitt trigger input or open-drain output. Pull-up resistors are assignable by software.	B	28–21 (22–15)	–
P2.0-P2.7	I/O	Bit-programmable I/O port for Schmitt trigger input or open-drain output. Port2 can be individually configured as external interrupt inputs. Pull-up resistors are assignable by software.	B	3–10 (41–44, 1–4)	INT0
P3.0-P3.3	I/O	Bit-programmable I/O port for Schmitt trigger input, open-drain or push-pull output. P3.3 can be used to system clock output(CLO) pin.	C	2, 1, 42, 41 (40–37)	P3.3/CLO
P4.0-P4.3	I/O	Bit-programmable I/O port for Schmitt trigger input or open-drain output or push-pull output. Port4 can be individually configured as external interrupt inputs. In output mode, pull-up resistors are assignable by software. But in input mode, pull-up resistors are fixed.	D	16, 17, 19, 20 (10, 11, 13, 14)	INT1
D+/D-	I/O	Only be used USB tranceive/receive port; D+/D-.	–	40–39 (36-35)	–
3.3 V_{OUT}	–	3.3 V output from internal voltage regulator	–	38 (34)	–
X_{IN}, X_{OUT}	–	System clock input and output pin (crystal/ceramic oscillator, or external clock source)	–	14, 13 (8, 7)	–
INT0 INT1 INT2	I	External interrupt for bit-programmable port0, port2 and port4 pins when set to input mode.	–	3–10, 16,17, 19, 20, 29–36 (30–23, 41–44, 1–4, 10, 11, 13, 14)	PORT2/ PORT4/ PORT0
RESET	I	RESET signal input pin. Schmitt trigger input with internal pull-up resistor.	A	18 (12)	–
TEST	I	Test signal input pin (for factory use only; must be connected to V_{SS})	–	15 (9)	–
V_{DD}	–	Power input pin	–	11 (5)	–
V_{SS1},V_{SS2}	–	Vss1 is a ground power for CPU core. Vss2 is a ground power for I/O and OSC block.	–	12, 37 (6, 31)	–
NC	–	No connection	–	– (32, 33)	–

NOTE: Pin numbers shown in parenthesis '( )' are for the 44-QFP package; others are for the 42-SDIP package.

PIN CIRCUITS

Table 1-2. Pin Circuit Assignments for the KS86C6004/C6008/P6008

Circuit Number	Circuit Type	KS86C6004/C6008/P6008 Assignments
A	I	RESET signal input
B	I/O	Ports 0, 1, and 2
C	I/O	Port 3
D	I/O	Port 4

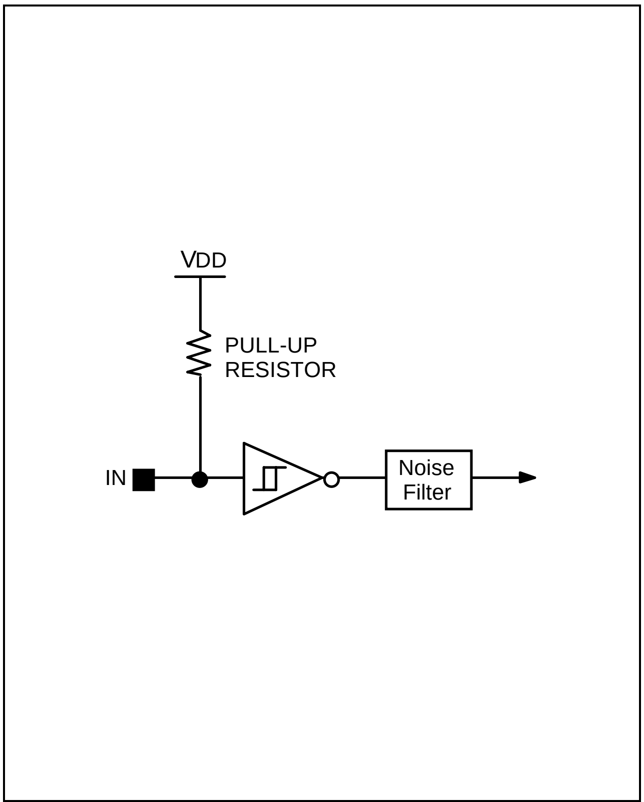


Figure 1-4. Pin Circuit Type A (RESET)

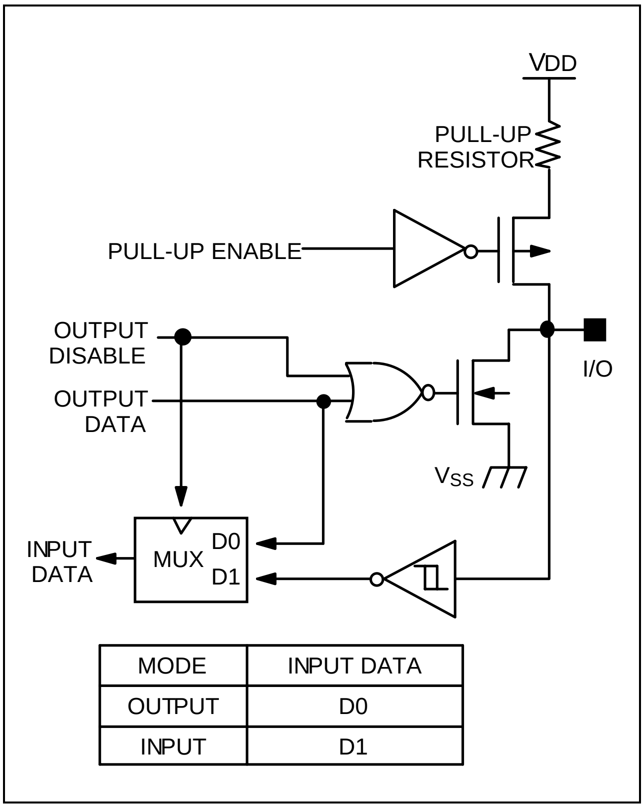
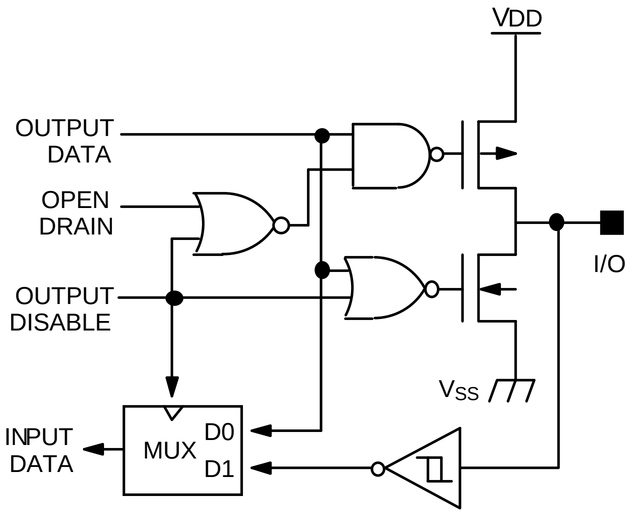
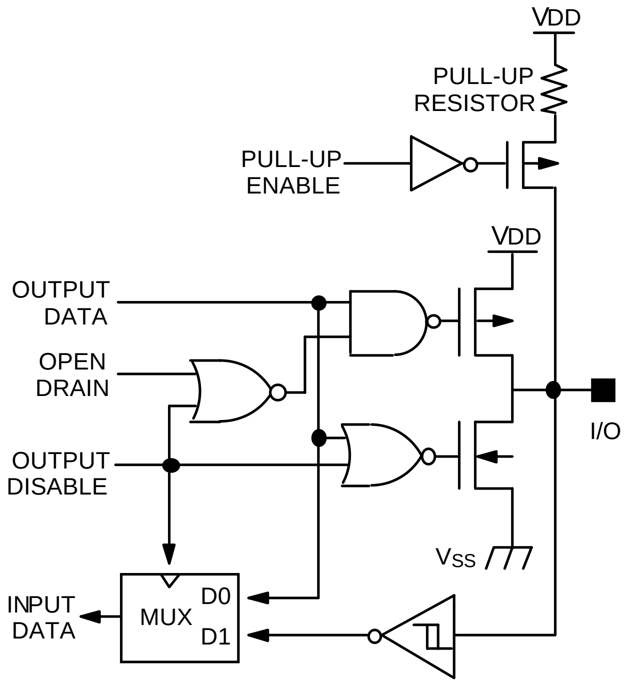


Figure 1-5. Pin Circuit Type B (Ports 0, 1 and 2)



MODE	INPUT DATA
OUTPUT	D0
INPUT	D1

Figure 1-6. Pin Circuit Type C (Port 3)



MODE	INPUT DATA
OUTPUT	D0
INPUT	D1

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

APPLICATION CIRCUIT

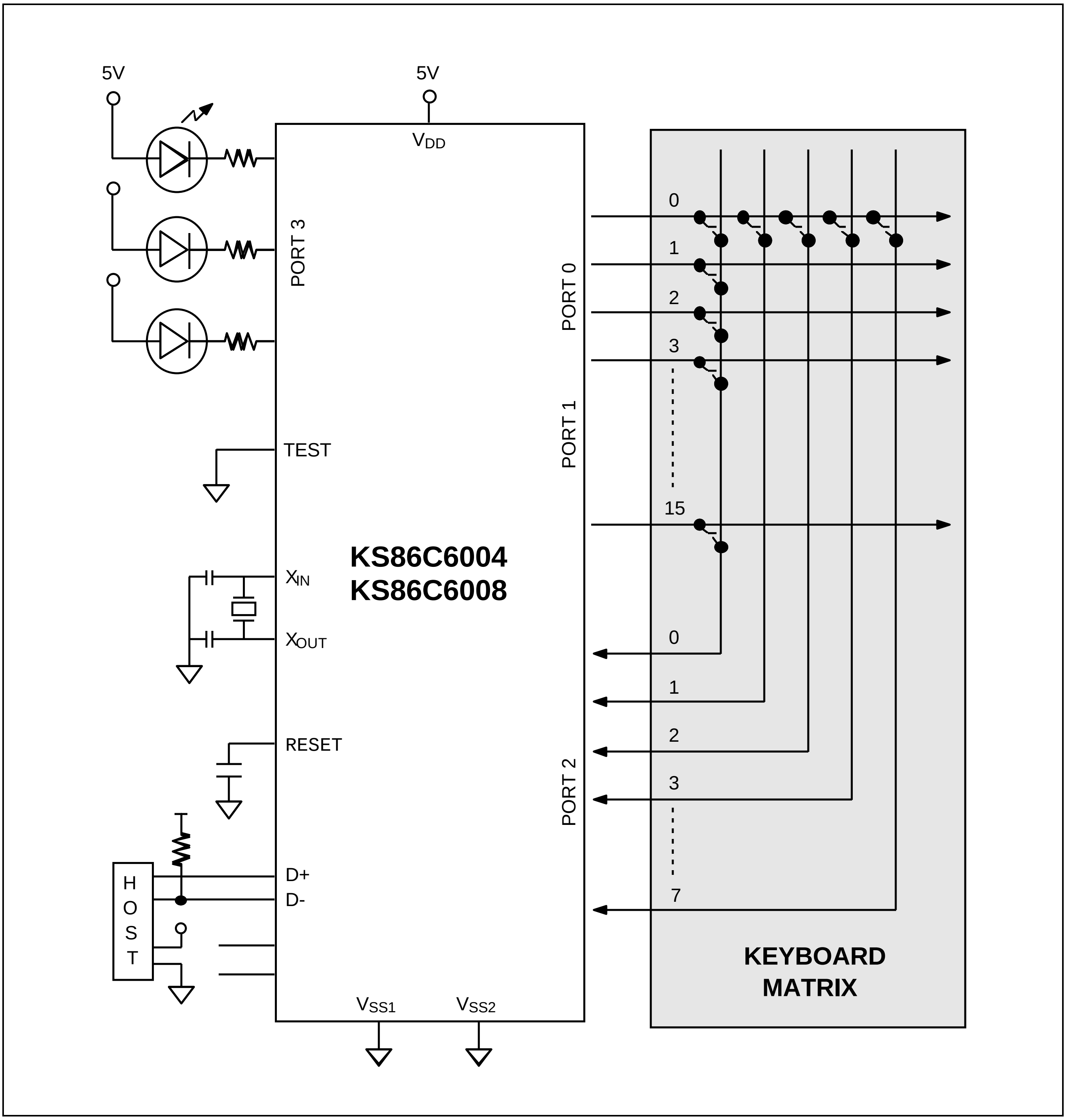


Figure 1-8. Keyboard Application Circuit Diagram

2 ADDRESS SPACES

OVERVIEW

The KS86C6004/C6008/P6008 microcontroller has two kinds of address space:

— Program memory (ROM), internal
— Internal register file

A 13-bit address bus supports both program memory. A separate 8-bit register bus carries addresses and data between the CPU and the internal register file.

The KS86C6004 has 4 K bytes of mask-programmable program memory on-chip and KS86C6008 has 8 K bytes. There is one program memory configuration option:

— Internal ROM mode, in which only the 8-Kbyte internal program memory is used.

The KS86C6004/C6008/P6008 microcontroller has 208 general-purpose registers in its internal register file. Twenty-seven bytes in the register file are mapped for system and peripheral control functions.

PROGRAM MEMORY (ROM)

Normal Operating Mode (Internal ROM)

The KS86C6004 has 4 K bytes (locations 0H–0FFFH) of internal mask-programmable program memory. The KS86C6008/P6008 has 8 K bytes (locations 0H–1FFFH) of internal mask-programmable program memory.

The first 2 bytes of the ROM (0000H–0001H) are an interrupt vector address.

The program reset address in the ROM is 0100H.

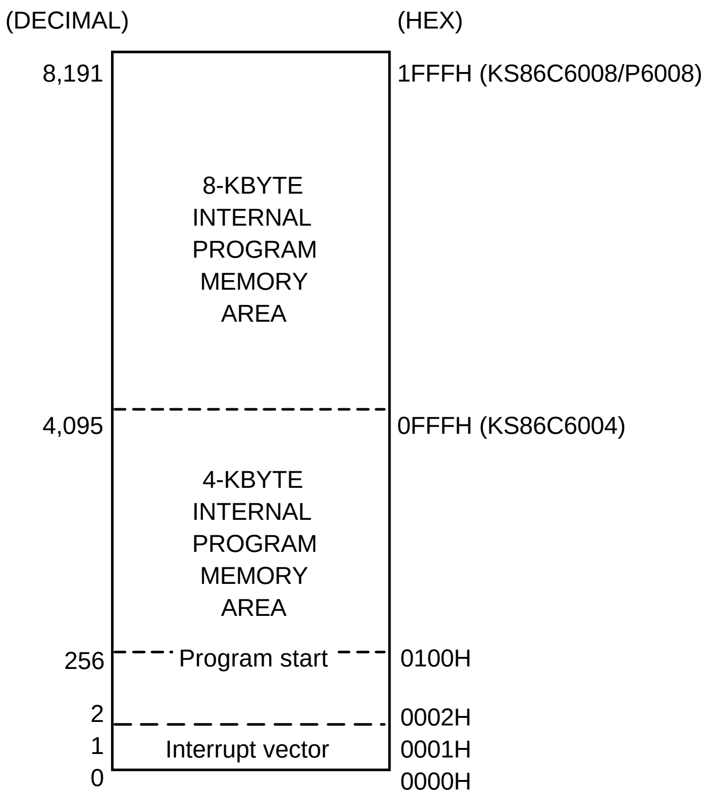


Figure 2-1. Program Memory Address Space

REGISTER ARCHITECTURE

The upper 64 bytes of the KS86C6004/C6008/P6008's internal register file are addressed as working registers, system control registers and peripheral control registers. The lower 192 bytes of internal register file (00H–BFH) is called the *general purpose register space*. The total addressable register space is thereby 256 bytes. 233 registers in this space can be accessed.; 208 are available for general-purpose use.

For many SAM87RI microcontrollers, the addressable area of the internal register file is further expanded by the additional of one or more register pages at general purpose register space (00H–BFH). This register file expansion is not implemented in the KS86C6004/C6008/P6008, however. Page addressing is controlled by the System Mode Register (SYM.1–SYM.0).

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

Table 2-1. Register Type Summary

Register Type	Number of Bytes
CPU and system control registers	11
Peripheral, I/O, and clock control and data registers	26
General-purpose registers (including the 16-bit common working register area)	208
Total Addressable Bytes	**245**

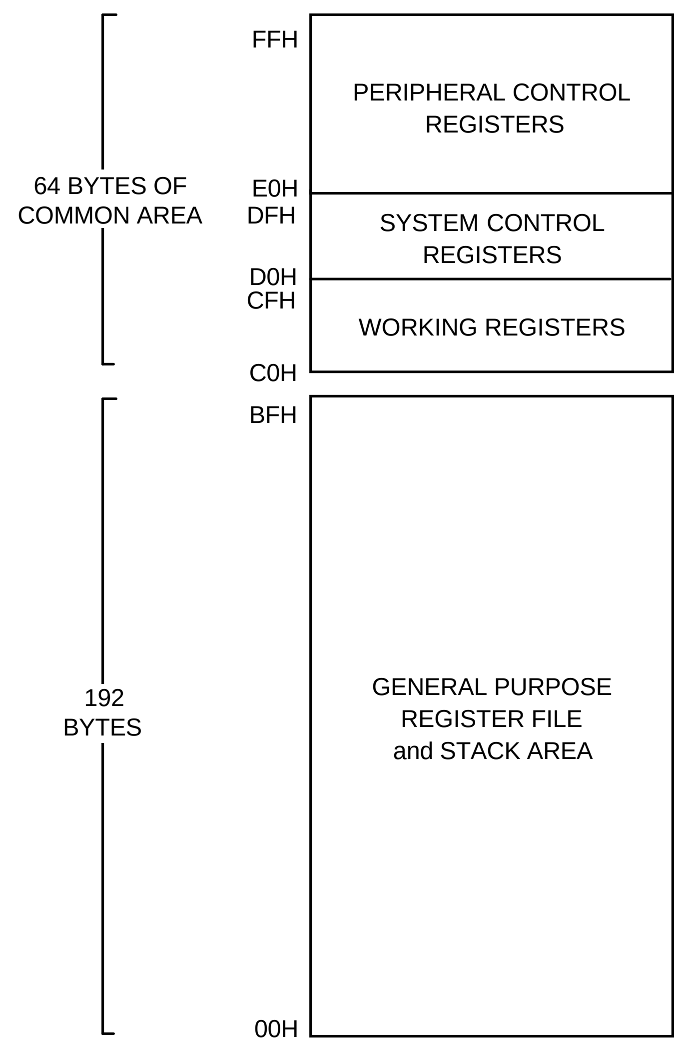


Figure 2-2. Internal Register File Organization

COMMON WORKING REGISTER AREA (C0H–CFH)

The SAM87RI register architecture provides an efficient method of working register addressing that takes full advantage of shorter instruction formats to reduce execution time.

This 16-byte address range is called *common area*. That is, locations in this area can be used as working registers by operations that address any location on any page in the register file. Typically, these working registers serve as temporary buffers for data operations between different pages. However, because the KS86C6004/C6008/P6008 uses only page 0, you can use the common area for any internal data operation.

The Register (R) addressing mode can be used to access this area

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.

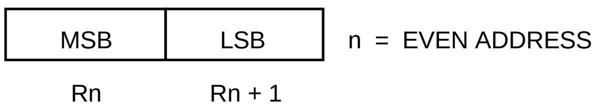


Figure 2-3. 16-Bit Register Pairs

☞ PROGRAMMING TIP — Addressing the Common Working Register Area

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

Examples:

1. LD 0C2H,40H ; Invalid addressing mode!

   Use working register addressing instead:

   LD R2,40H ; R2 (C2H) ← the value in location 40H

2. ADD 0C3H,#45H ; Invalid addressing mode!

   Use working register addressing instead:

   ADD R3,#45H ; R3 (C3H) ← R3 + 45H

SYSTEM STACK

KS86-series microcontrollers use the system stack for subroutine calls and returns and to store data. The PUSH and POP instructions are used to control system stack operations. The KS86C6004/C6008/P6008 architecture supports stack operations in the internal register file.

Stack Operations

Return addresses for procedure calls and interrupts and data are stored on the stack. The contents of the PC are saved to stack by a CALL instruction and restored by the RET instruction. When an interrupt occurs, the contents of the PC and the FLAGS register are pushed to the stack. The IRET instruction then pops these values back to their original locations. The stack address 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 2-4.

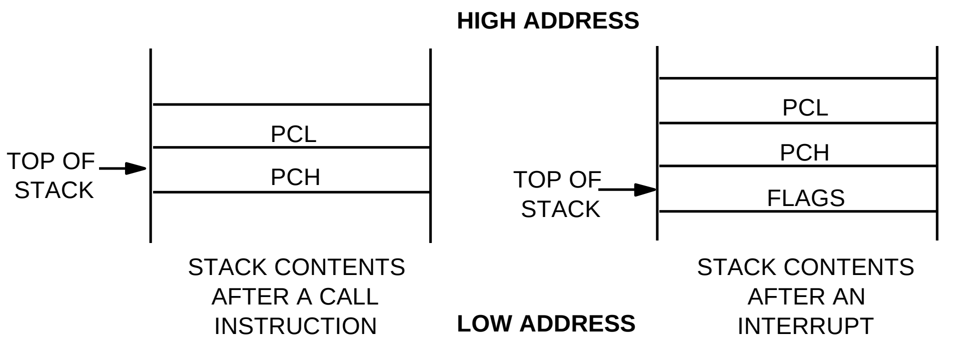


Figure 2-4. Stack Operations

Stack Pointer (SP)

Register location D9H contains the 8-bit stack pointer (SP) that is used for system stack operations. After a reset, the SP value is undetermined.

Because only internal memory space is implemented in the KS86C6004/C6008/P6008, the SP must be initialized to an 8-bit value in the range 00H–BFH.

NOTE

In case a Stack Pointer is initialized to 00H, it is decreased to FFH when stack operation starts. This means that a Stack Pointer access invalid stack area.

☞ 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      SP,#0C0H        ;  SP ← C0H (Normally, the SP is set to 0C0H by the
                        ;  initialization routine)
•
•
•
PUSH    SYM             ;  Stack address 0BFH ← SYM
PUSH    CLKCON          ;  Stack address 0BEH ← CLKCON
PUSH    20H             ;  Stack address 0BDH ← 20H
PUSH    R3              ;  Stack address 0BCH ← R3
•
•
•
POP     R3              ;  R3 ← Stack address 0BCH
POP     20H             ;  20H ← Stack address 0BDH
POP     CLKCON          ;  CLKCON ← Stack address 0BEH
POP     SYM             ;  SYM ← Stack address 0BFH
```

NOTES

3 ADDRESSING MODES

OVERVIEW

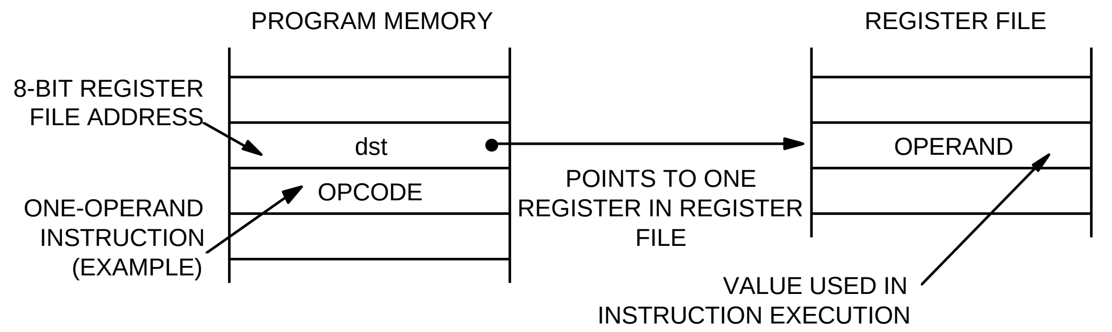
Instructions that are stored in program memory are fetched for execution using the program counter. Instructions indicate the operation to be performed and the data to be operated on. Addressing mode is the method used to determine the location of the data operand. The operands specified in SAM87RI instructions may be condition codes, immediate data, or a location in the register file, program memory, or data memory.

The SAM87RI instruction set supports six 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)
- Relative Address (RA)
- Immediate (IM)

REGISTER ADDRESSING MODE (R)

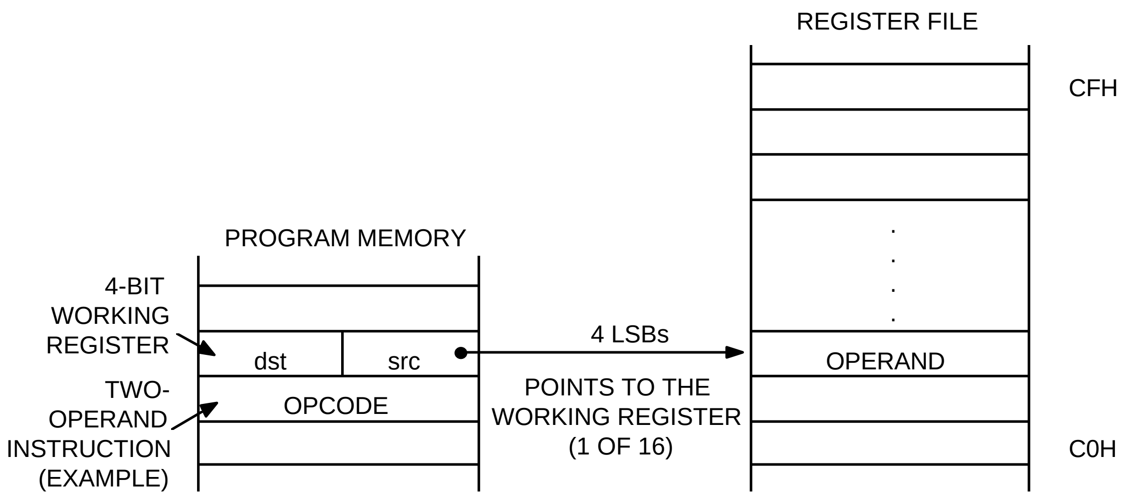
In Register addressing mode, the operand is the content of a specified register (see Figure 3-1). Working register addressing differs from Register addressing because it uses an 16-byte working register space in the register file and an 4-bit register within that space (see Figure 3-2).



SAMPLE INSTRUCTION:

DEC CNTR ; Where CNTR is the label of an 8-bit register address

Figure 3-1. Register Addressing



SAMPLE INSTRUCTION:

ADD R1,R2 ; Where R1=C1H and R2=C2H

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.

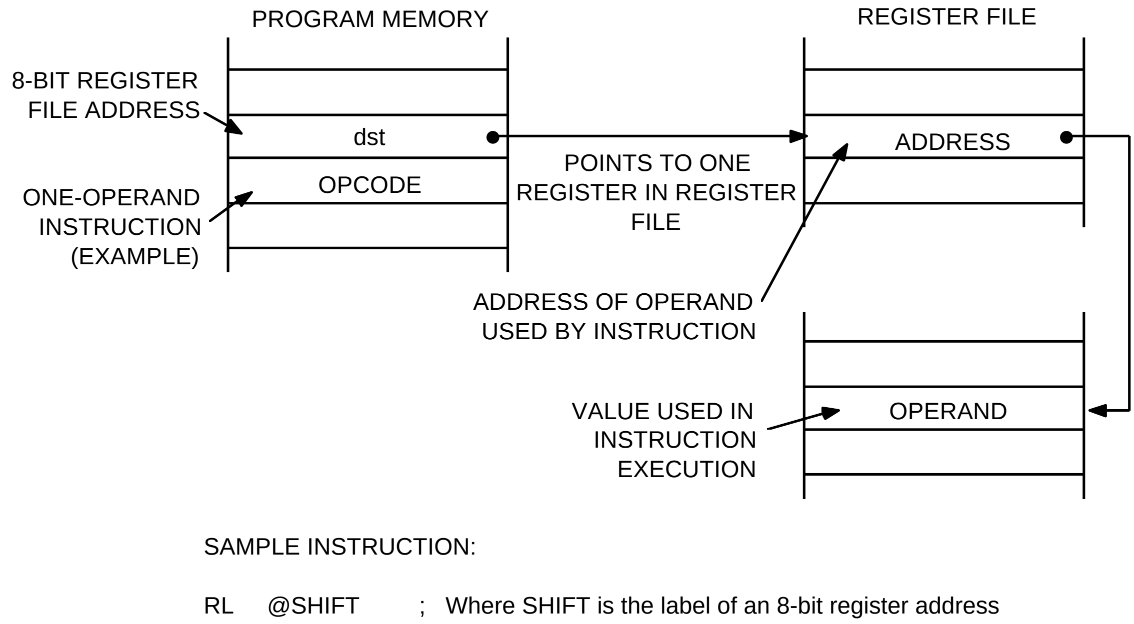


Figure 3-3. Indirect Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Continued)

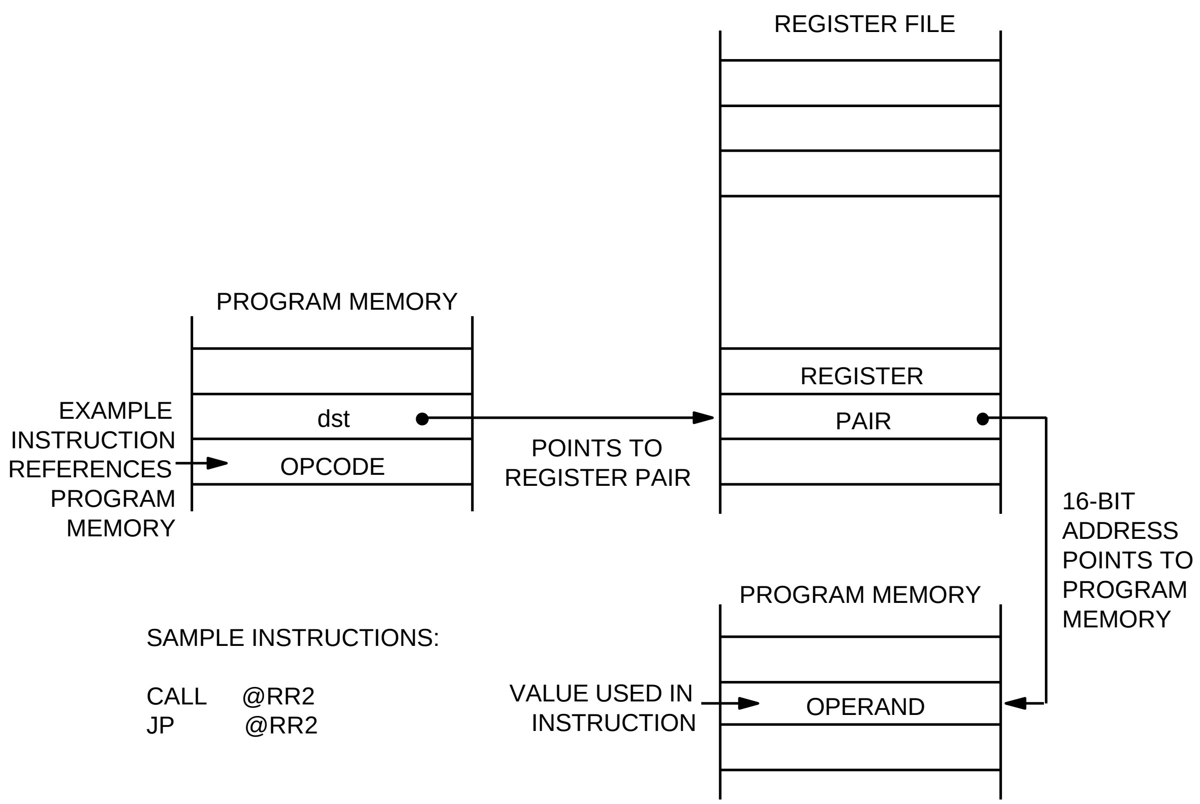


Figure 3-4. Indirect Register Addressing to Program Memory

INDIRECT REGISTER ADDRESSING MODE (Continued)

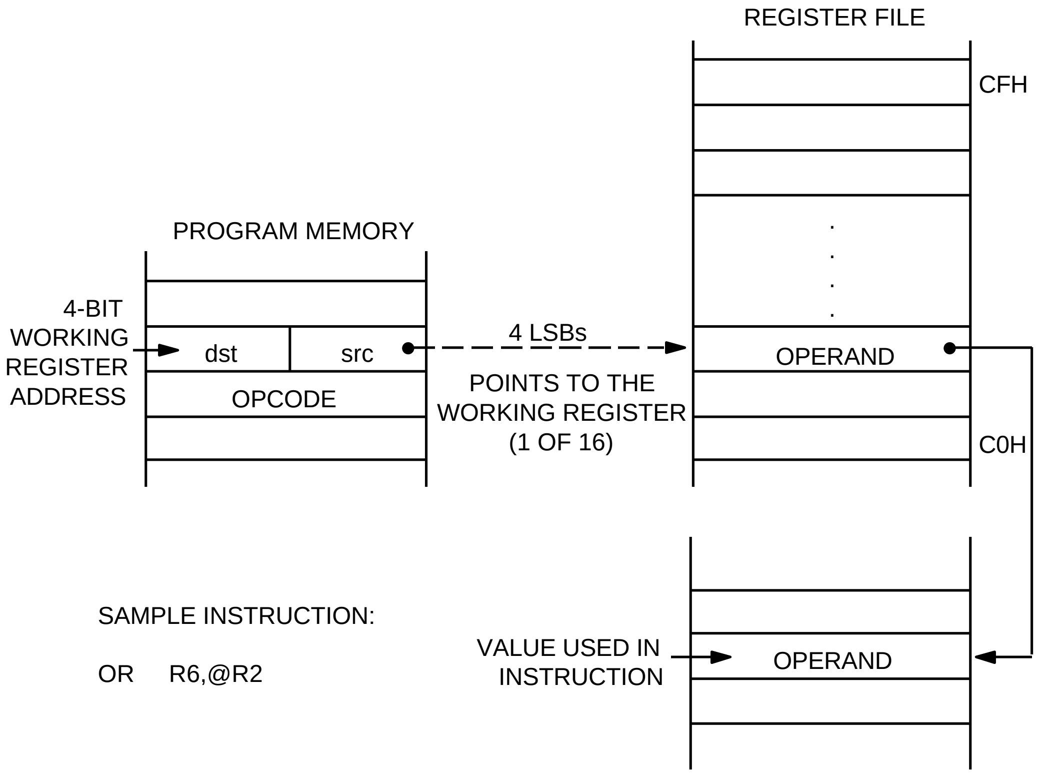
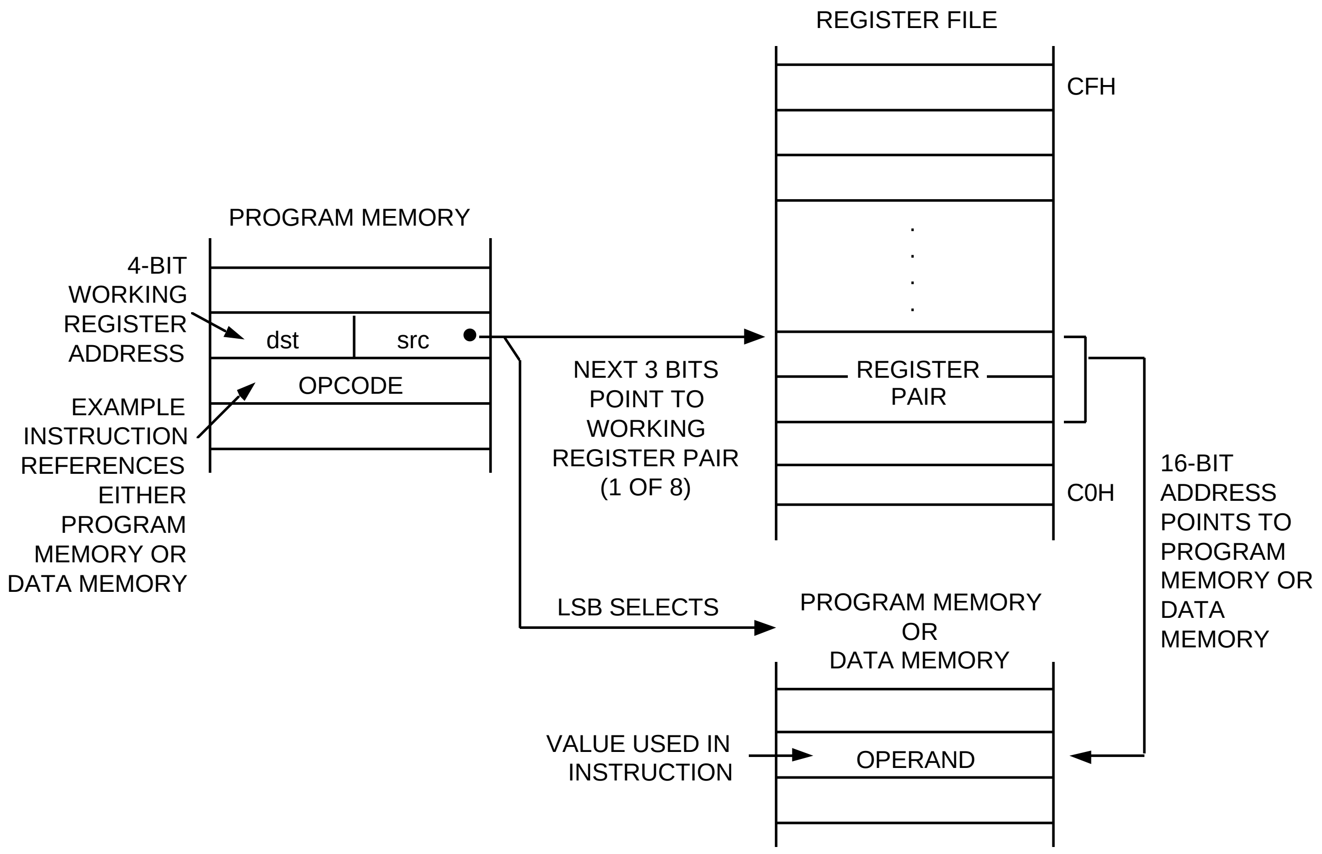


Figure 3-5. Indirect Working Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Concluded)



SAMPLE INSTRUCTIONS:

```
LDC   R5,@RR2     ;  Program memory access
LDE   R3,@RR14    ;  External data memory access
LDE   @RR4,R8     ;  External data memory access
```

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

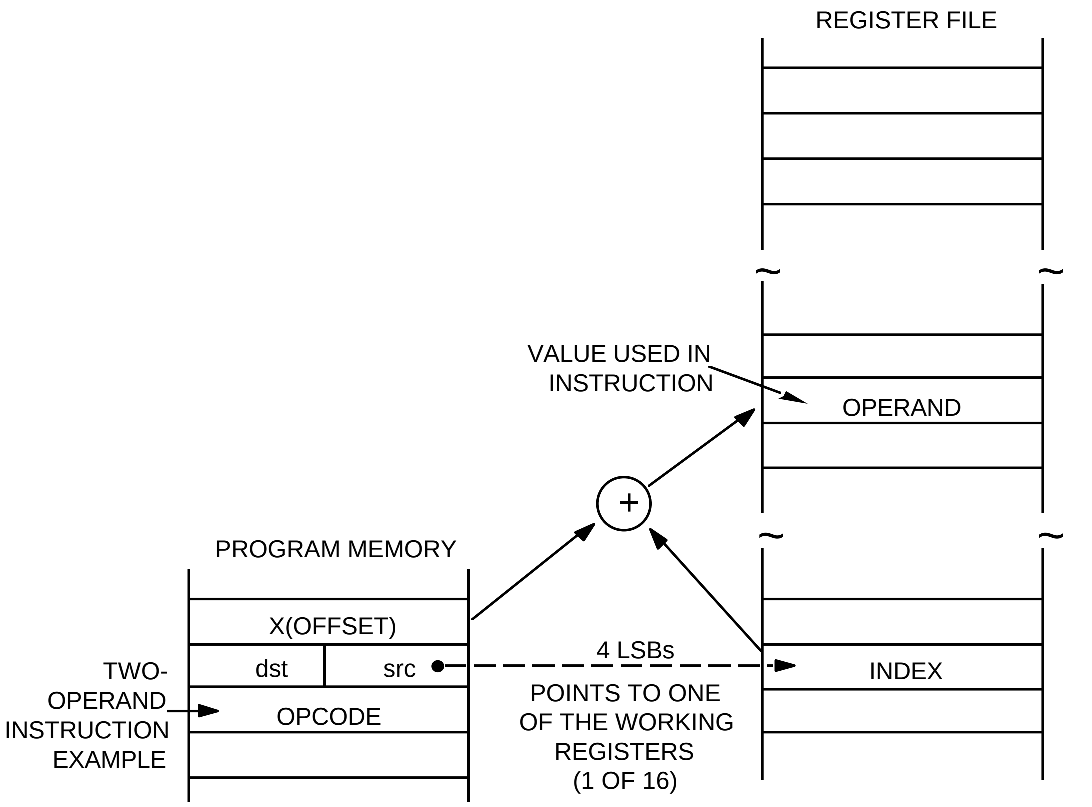
INDEXED ADDRESSING MODE (X)

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.

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, external program memory, and for external data memory, when implemented.

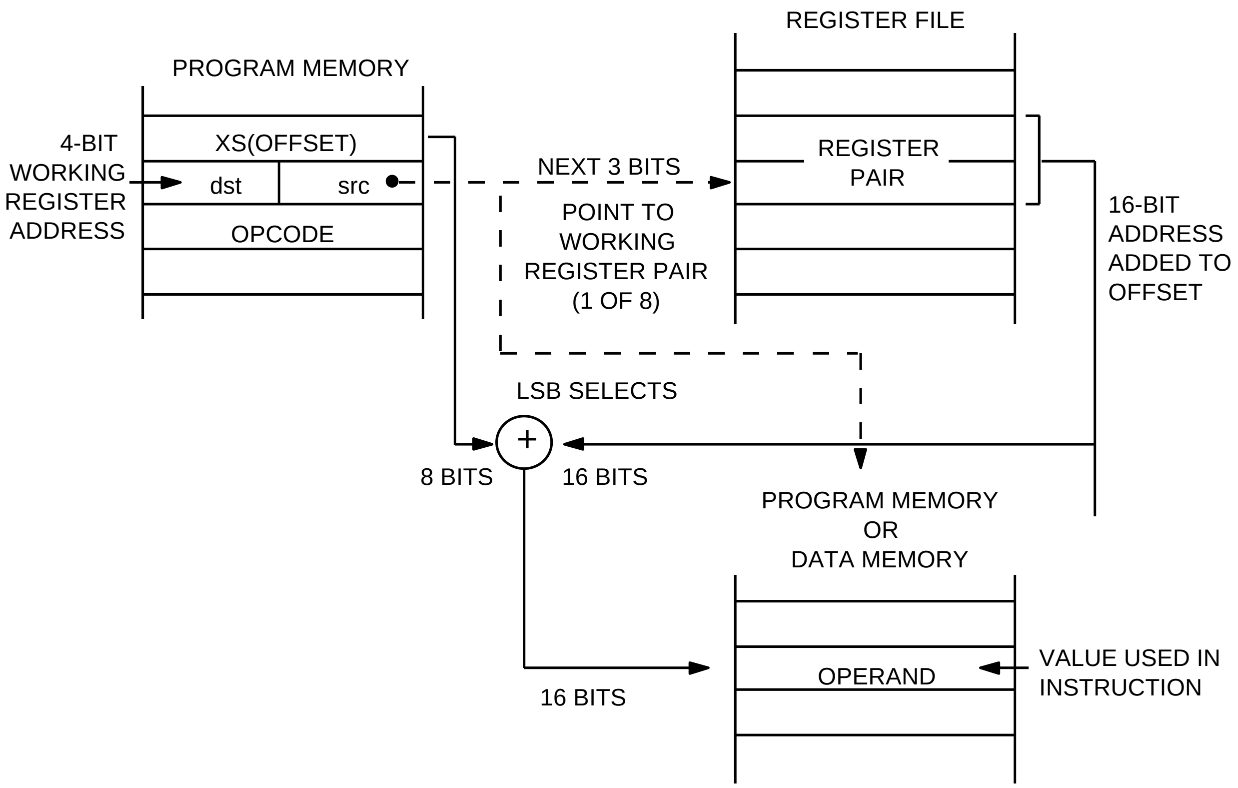


SAMPLE INSTRUCTION:

LD R0,#BASE[R1] ; Where BASE is an 8-bit immediate value

Figure 3-7. Indexed Addressing to Register File

INDEXED ADDRESSING MODE (Continued)



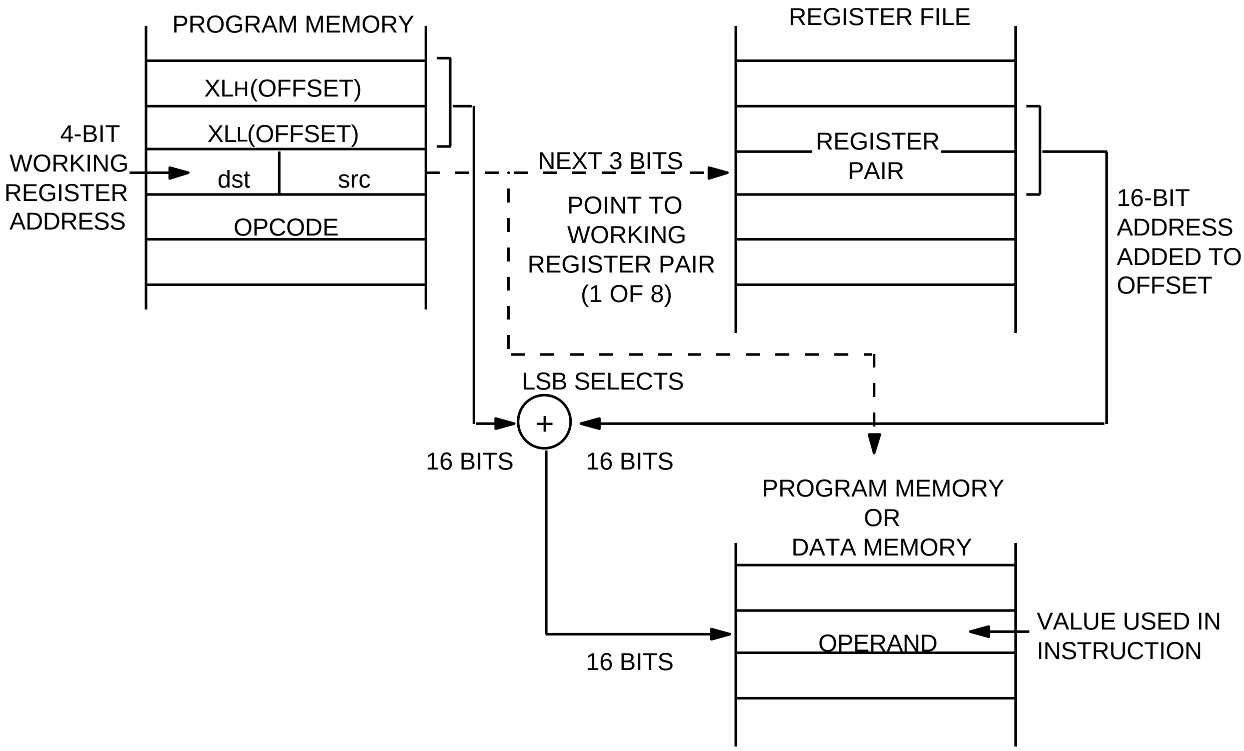
SAMPLE INSTRUCTIONS:

LDC R4,#04H[RR2] ; The values in the program address (RR2 + #04H) are loaded into register R4.

LDE R4,#04H[RR2] ; Identical operation to LDC example, except that external program memory is accessed.

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

INDEXED ADDRESSING MODE (Concluded)



SAMPLE INSTRUCTIONS:

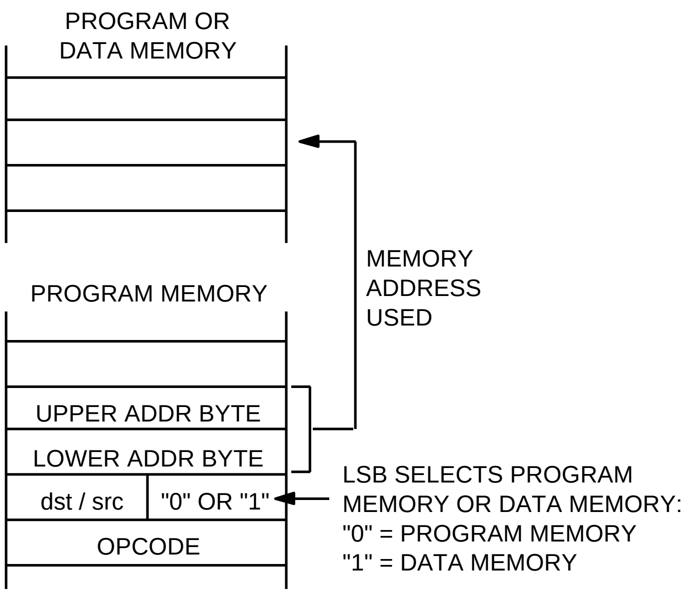
LDC	R4,#1000H[RR2]	; The values in the program address (RR2 + #1000H) are loaded into register R4.
LDE	R4,#1000H[RR2]	; Identical operation to LDC example, except that external program memory is accessed.

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

DIRECT ADDRESS MODE (DA)

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

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

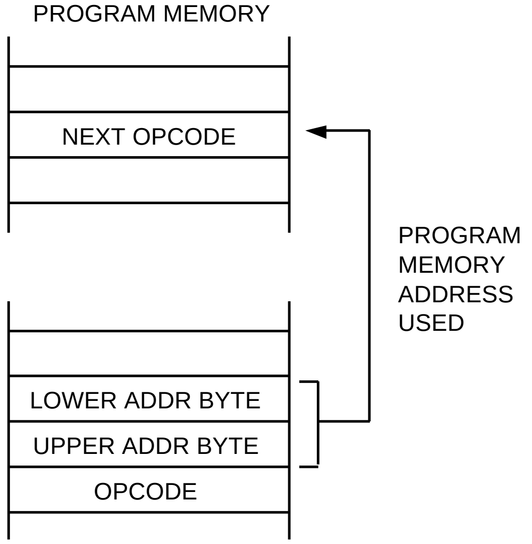


SAMPLE INSTRUCTIONS:

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

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

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.

The instructions that support RA addressing is JR.

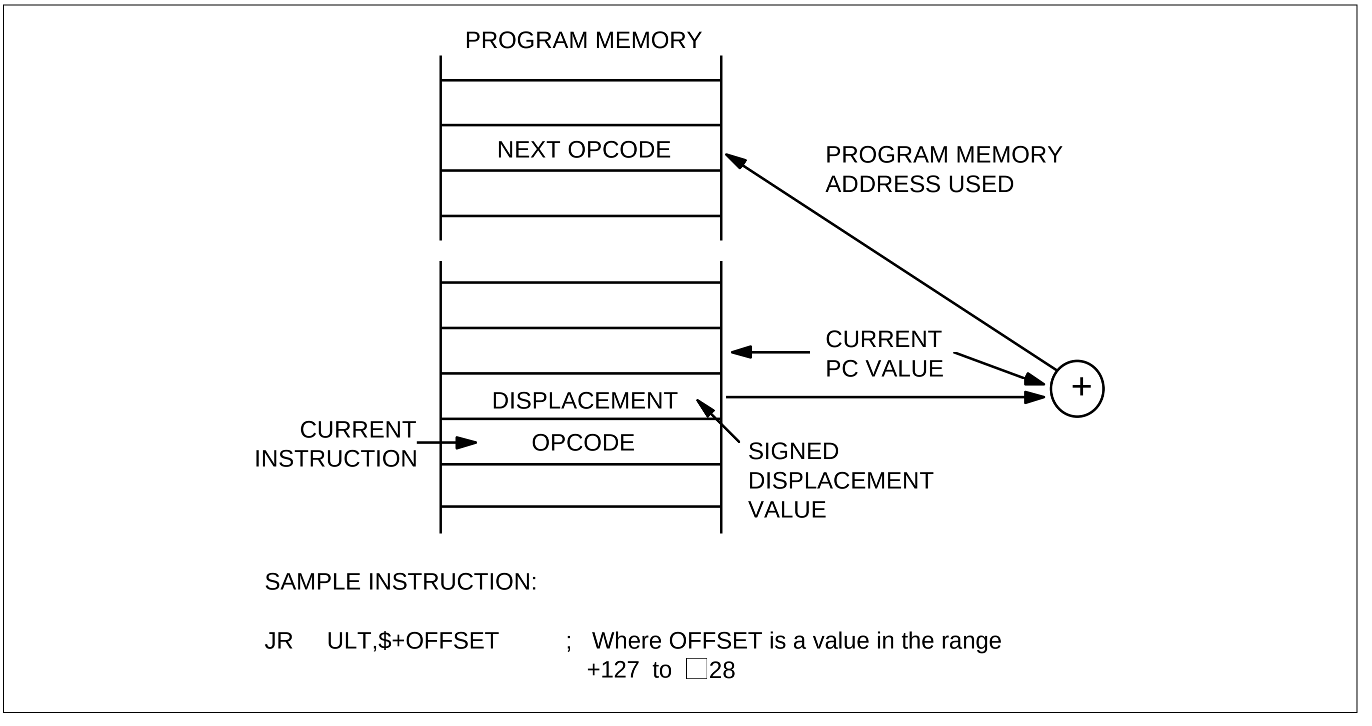


Figure 3-12. Relative Addressing

IMMEDIATE MODE (IM)

In Immediate (IM) addressing mode, the operand value used in the instruction is the value supplied in the operand field itself. Immediate addressing mode is useful for loading constant values into registers.

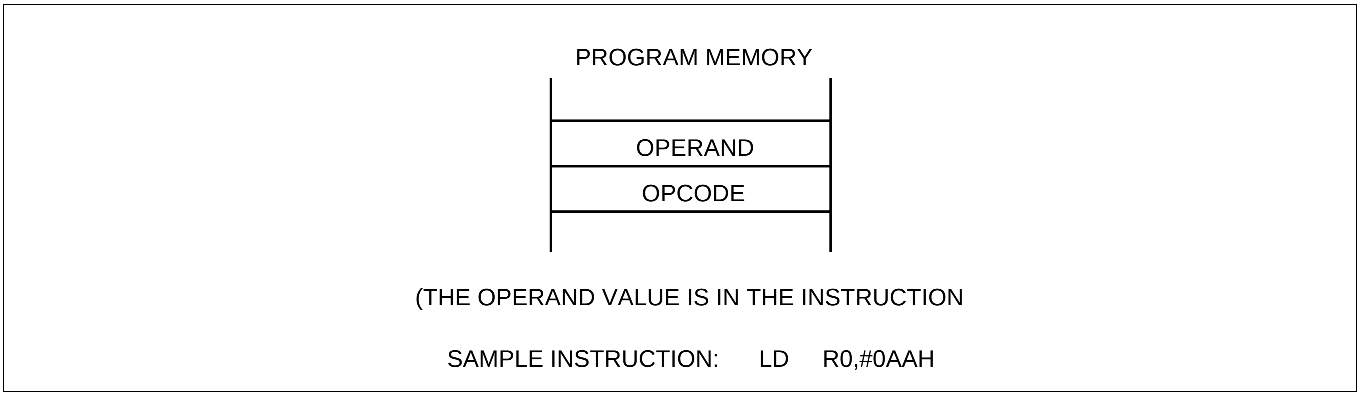


Figure 13-13. Immediate Addressing

4 CONTROL REGISTERS

In this section, detailed descriptions of the KS86C6004/C6008/P6008 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 Table 4-1. 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. System and Peripheral control Registers

Register Name	Mnemonic	Decimal	Hex	R/W
Timer 0 counter register	T0CNT	208	D0H	R
Timer 0 data register	T0DATA	209	D1H	R/W
Timer 0 control register	T0CON	210	D2H	R/W
Location D3H is not mapped.				
Clock control register	CLKCON	212	D4H	R/W
System flags register	FLAGS	213	D5H	R/W
Locations D6H–D7H are not mapped.				
Port 0 interrupt control register	P0INT	216	D8H	R/W
Stack pointer	SP	217	D9H	R/W
Port 0 interrupt pending register	P0PND	218	DAH	R/W
Location DBH is not mapped.				
Basic timer control register	BTCON	220	DCH	R/W
Basic timer counter register	BTCNT	221	DDH	R
Location DEH is not mapped.				
System mode register	SYM	223	DFH	R/W
Port 0 data register	P0	224	E0H	R/W
Port 1 data register	P1	225	E1H	R/W
Port 2 data register	P2	226	E2H	R/W
Port 3 data register	P3	227	E3H	R/W
Port 4 data register	P4	228	E4H	R/W
Port 3 control register	P3CON	229	E5H	R/W
Port 0 control register (high byte)	P0CONH	230	E6H	R/W
Port 0 control register (low byte)	P0CONL	231	E7H	R/W
Port 1 control register (high byte)	P1CONH	232	E8H	R/W
Port 1 control register (low byte)	P1CONL	233	E9H	R/W
Port 2 control register (high byte)	P2CONH	234	EAH	R/W
Port 2 control register (low byte)	P2CONL	235	EBH	R/W
Port 2 interrupt control register	P2INT	236	ECH	R/W
Port 2 interrupt pending register	P2PND	237	EDH	R/W
Port 4 control register	P4CON	238	EEH	R/W
Port 4 interrupt enable/pending register	P4INTPND	239	EFH	R/W

Table 4-1. System and Peripheral control Registers (Continued)

Register Name	Mnemonic	Decimal	Hex	R/W
USB function address register	FADDR	240	F0H	R/W
Control endpoint status register	EP0CSR	241	F1H	R/W
Interrupt endpoint status register	EP1CSR	242	F2H	R/W
Control endpoint byte count register	EP0BCNT	243	F3H	R/W
Control endpoint FIFO register	EP0FIFO	244	F4H	R/W
Interrupt endpoint FIFO register	EP1FIFO	245	F5H	R/W
USB interrupt pending register	USBPND	246	F6H	R/W
USB interrupt enable register	USBINT	247	F7H	R/W
USB power management register	PWRMGR	248	F8H	R/W
Locations F9H–FEH are not mapped.				
USB reset register	USBRST	255	FFH	R/W

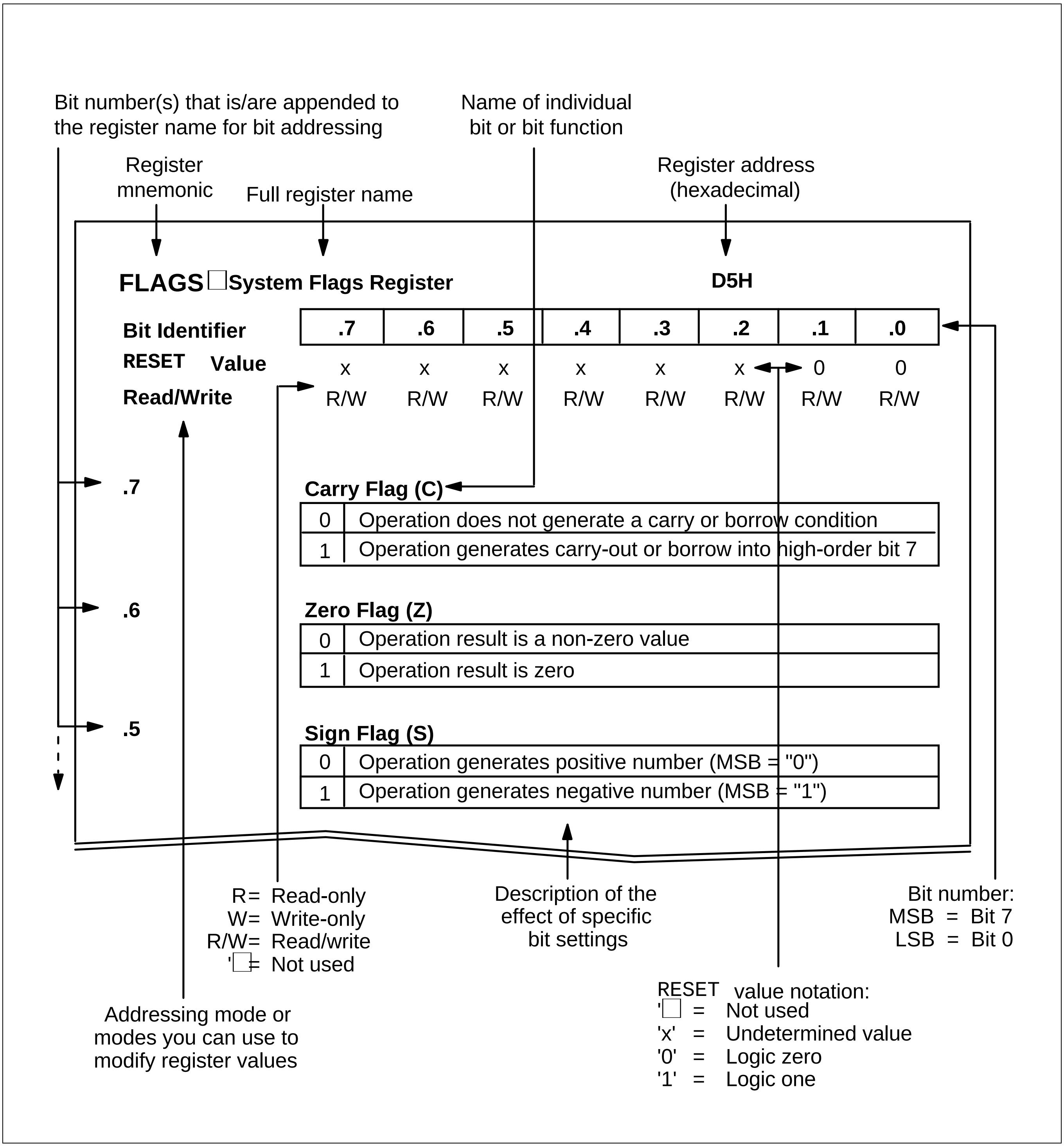


Figure 4-1. Register Description Format

BTCON — Basic Timer Control Register DCH

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

.7–.4 **Watchdog Timer Enable Bits**

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

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

0	0	$f_{OSC}/4096$
0	1	$f_{OSC}/1024$
1	0	$f_{OSC}/128$
1	1	Invalid setting

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

0	No effect
1	Clear BTCNT

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

0	No effect
1	Clear both dividers

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

CLKCON — System Clock Control Register D4H

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

.7 **Oscillator IRQ Wake-up Function Bit**

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

.6 and .5 Not used for KS86C6004/C6008/P6008

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

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

.2–.0 Not used for KS86C6004/C6008/P6008

NOTES:
1. After a reset, the slowest clock (divided by 16) is selected as the system clock. To select faster clock speeds, load the appropriate values to CLKCON.3 and CLKCON.4.
2. f_{OSC} means oscillator frequency.

EP0CSR — Control Endpoint 0 Status Register F1H

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

.7 **Setup Data End Clear Bit**

0	No effect (when write)
1	To clear SETUP_END bit

.6 **Out Packet Ready Clear Bit**

0	No effect (when write)
1	To clear OUT_PKT_RDY bit

.5 **STALL Signal Sending Bit**

0	No effect (when write)
1	To send STALL signal

.4 **Setup Transfer End Bit**

0	No effect (when write)
1	SIE sets this bit when a control transfer ends before DATA_END (bit3) is set

.3 **Setup Data End Bit**

0	No effect (when write)
1	MCU set this bit after loading or unloading the last packet data into the FIFO

.2 **STALL Signal Receive Bit**

0	MCU clear this bit to end the STALL condition
1	SIE sets this bit if a control transaction is ended due to a protocol violation

.1 **In Packet Ready Bit**

0	SIE clear this bit once the packet has been successfully sent to the host
1	MCU sets this bit after writing a packet of data into ENDPOINT0 FIFO

.0 **Out Packet Ready Bit**

0	No effect (when write)
1	SIE sets this bit once a valid token is written to the FIFO

EP1CSR — Control Endpoint 1 Status Register F2H

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

.7 **Data Toggle Sequence Clear Bit**

0	No effect (when write)
1	MCU sets this bit to clear the data toggle sequence bit. The data toggle is initialized to DATA0.

.6–.3 **Maximum Packet Size Bits**

0	No effect (when write)
1	These bits indicate the maximum packet size for IN endpoint, and needs to be updated by the MCU before it sets IN_PKT_RDY. Once set, the contents are valid till MCU re-writes them.

.2 **FIFO Flush Bit**

0	No effect (when write)
1	When MCU writes a one to this register, the FIFO is flushed, and IN_PKT_RDY cleared. The MCU should wait for IN_PKT_RDY to be cleared for the flush to take place.

.1 **Force STALL Bit**

0	No effect (when write)
1	MCU writes a 1 to this register to issue a STALL handshake to USB. MCU clears this bit, to end the STALL condition.

.0 **In Packet Ready Bit**

0	SIE clear this bit once the packet has been successfully sent to the host
1	MCU sets this bit, after writing a packet of data into ENDPOINT0 FIFO. USB clears this bit, once the packet has been successfully sent to the host. An interrupt is generated when USB clears this bit, so MCU can load the next packet.

FADDR — USB Function Address Register F0H

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

.7 Not used for KS86C6004/C6008/P6008

.6–.0 **FADDR**

This register holds the USB address assigned by the host computer. FADDR is located at address F0H and is read/write addressable.

FLAGS — System Flags Register D5H

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

.7 **Carry Flag (C)**

0	Operation does not generate a carry or borrow condition

.6 **Zero Flag (Z)**

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

.5 **Sign Flag (S)**

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

.4 **Overflow Flag (V)**

0	Operation result is ≤ +127 or ≥ –128
1	Operation result is ≥ +127 or ≤ –128

.3–0. Not used for KS86C6004/C6008/P6008

P0CONH — Port 0 Control Register (High Byte) E6H

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

.7 and .6 — **Port 0, P0.7 Configuration Bits**

0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edge external interrupt with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.5 and .4 — **Port 0, P0.6 Configuration Bits**

0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edge external interrupt with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.3 and .2 — **Port 0, P0.5 Configuration Bits**

0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edge external interrupt with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.1 and .0 — **Port 0, P0.4 Configuration Bits**

0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edge external interrupt with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

P0CONL — Port 0 Control Register (Low Byte) E7H

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

.7 and .6 **Port 0, P0.3 Configuration Bits**

0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edge external interrupt with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.5 and .4 **Port 0, P0.2 Configuration Bits**

0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edge external interrupt with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.3 and .2 **Port 0, P0.1 Configuration Bits**

0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edge external interrupt with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.1 and .0 **Port 0, P0.0 Configuration Bits**

0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edge external interrupt with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

P0INT — Port 0 Interrupt Control Register D8H

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

.7 **P0.7 Configuration Bits**

0	External interrupt disable
1	External interrupt enable

.6 **P0.6 Configuration Bits**

0	External interrupt disable
1	External interrupt enable

.5 **P0.5 Configuration Bits**

0	External interrupt disable
1	External interrupt enable

.4 **P0.4 Configuration Bits**

0	External interrupt disable
1	External interrupt enable

.3 **P0.3 Configuration Bits**

0	External interrupt disable
1	External interrupt enable

.2 **P0.2 Configuration Bits**

0	External interrupt disable
1	External interrupt enable

.1 **P0.1 Configuration Bits**

0	External interrupt disable
1	External interrupt enable

.0 **P0.0 Configuration Bits**

0	External interrupt disable
1	External interrupt enable

P0PND — Port 0 Interrupt Pending Register DAH

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

.7 **P0.7 Interrupt Pending Bit**

0	No pending (when read)/clear pending bit (when write)
1	Pending (when read)/no effect (when write)

.6 **P0.6 Interrupt Pending Bit**

0	No pending (when read)/clear pending bit (when write)
1	Pending (when read)/no effect (when write)

.5 **P0.5 Interrupt Pending Bit**

0	No pending (when read)/clear pending bit (when write)
1	Pending (when read)/no effect (when write)

.4 **P0.4 Interrupt Pending Bit**

0	No pending (when read)/clear pending bit (when write)
1	Pending (when read)/no effect (when write)

.3 **P0.3 Interrupt Pending Bit**

0	No pending (when read)/clear pending bit (when write)
1	Pending (when read)/no effect (when write)

.2 **P0.2 Interrupt Pending Bit**

0	No pending (when read)/clear pending bit (when write)
1	Pending (when read)/no effect (when write)

.1 **P0.1 Interrupt Pending Bit**

0	No pending (when read)/clear pending bit (when write)
1	Pending (when read)/no effect (when write)

.0 **P0.0 Interrupt Pending Bit**

0	No pending (when read)/clear pending bit (when write)
1	Pending (when read)/no effect (when write)

P1CONH — Port 1 Control Register (High Byte) E8H

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

.7 and .6 **Port 1, P1.7 Configuration Bits**

0	0	Schmitt trigger input
0	1	Schmitt trigger input with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.5 and .4 **Port 1, P1.6 Configuration Bits**

0	0	Schmitt trigger input
0	1	Schmitt trigger input with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.3 and .2 **Port 1, P1.5 Configuration Bits**

0	0	Schmitt trigger input
0	1	Schmitt trigger input with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.1 and .0 **Port 1, P1.4 Configuration Bits**

0	0	Schmitt trigger input
0	1	Schmitt trigger input with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

P1CONL — Port 1 Control Register (Low Byte) E9H

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

.7 and .6 **Port 1, P1.3 Configuration Bits**

0	0	Schmitt trigger input
0	1	Schmitt trigger input with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.5 and .4 **Port 1, P1.2 Configuration Bits**

0	0	Schmitt trigger input
0	1	Schmitt trigger input with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.3 and .2 **Port 1, P1.1 Configuration Bits**

0	0	Schmitt trigger input
0	1	Schmitt trigger input with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.1 and .0 **Port 1, P1.0 Configuration Bits**

0	0	Schmitt trigger input
0	1	Schmitt trigger input with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

P2CONH — Port 2 Control Register (High Byte) EAH

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

.7 and .6 **Port 2, P2.7 Configuration Bits**

0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edges external interrupt with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.5 and .4 **Port 2, P2.6 Configuration Bits**

0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edges external interrupt with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.3 and .2 **Port 2, P2.5 Configuration Bits**

0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edges external interrupt with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.1 and .0 **Port 2, P2.4 Configuration Bits**

0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edges external interrupt with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

P2CONL — Port 2 Control Register (Low Byte) EBH

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

.7 and .6 **Port 2, P2.3 Configuration Bits**

0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edges external interrupt with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.5 and .4 **Port 2, P2.2 Configuration Bits**

0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edges external interrupt with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.3 and .2 **Port 2, P2.1 Configuration Bits**

0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edges external interrupt with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

.1 and .0 **Port 2, P2.0 Configuration Bits**

0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edges external interrupt with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

P2INT — Port 2 Interrupt Enable Register ECH

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

.7 **P2.7 Interrupt Enable Bit**

0	External interrupt disable
1	External interrupt enable

.6 **P2.6 Interrupt Enable Bit**

0	External interrupt disable
1	External interrupt enable

.5 **P2.5 Interrupt Enable Bit**

0	External interrupt disable
1	External interrupt enable

.4 **P2.4 Interrupt Enable Bit**

0	External interrupt disable
1	External interrupt enable

.3 **P2.3 Interrupt Enable Bit**

0	External interrupt disable
1	External interrupt enable

.2 **P2.2 Interrupt Enable Bit**

0	External interrupt disable
1	External interrupt enable

.1 **P2.1 Interrupt Enable Bit**

0	External interrupt disable
1	External interrupt enable

.0 **P2.0 Interrupt Enable Bit**

0	External interrupt disable
1	External interrupt enable

P2PND — Port 2 Interrupt Pending Register EDH

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

.7 **P2.7 Interrupt Pending Bit**

0	No pending (when read)/clear pending bit (when write)
1	Pending (when read)/no effect (when write)

.6 **P2.6 Interrupt Pending Bit**

0	No pending (when read)/clear pending bit (when write)
1	Pending (when read)/no effect (when write)

.5 **P2.5 Interrupt Pending Bit**

0	No pending (when read)/clear pending bit (when write)
1	Pending (when read)/no effect (when write)

.4 **P2.4 Interrupt Pending Bit**

0	No pending (when read)/clear pending bit (when write)
1	Pending (when read)/no effect (when write)

.3 **P2.3 Interrupt Pending Bit**

0	No pending (when read)/clear pending bit (when write)
1	Pending (when read)/no effect (when write)

.2 **P2.2 Interrupt Pending Bit**

0	No pending (when read)/clear pending bit (when write)
1	Pending (when read)/no effect (when write)

.1 **P2.1 Interrupt Pending Bit**

0	No pending (when read)/clear pending bit (when write)
1	Pending (when read)/no effect (when write)

.0 **P2.0 Interrupt Pending Bit**

0	No pending (when read)/clear pending bit (when write)
1	Pending (when read)/no effect (when write)

NOTE: To clear a port 2 interrupt pending condition, write a "0" to the corresponding P2PND register bit location.

P3CON — Port 3 Control Register E5H

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

.7 and .6 **Port 3, P3.3 Configuration Bits**

0	0	Schmitt trigger input
0	1	System clock output(CLO) mode. CLO comes from system clock circuit.
1	0	Push-pull output
1	1	N-channel open-drain output mode

.5 and .4 **Port 3, P3.2 Configuration Bits**

0	x	Schmitt trigger input
1	0	Push-pull output
1	1	N-channel open-drain output mode

.3 and .2 **Port 3, P3.1 Configuration Bits**

0	x	Schmitt trigger input
1	0	Push-pull output
1	1	N-channel open-drain output mode

.1 and .0 **Port 3, P3.0 Configuration Bits**

0	x	Schmitt trigger input
1	0	Push-pull output
1	1	N-channel open-drain output mode

NOTE: "x" means don't care

P4CON — Port 4 Control Register EEH

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

.7 and .6 **Port 4, P4.3 Configuration Control Bits**

0	0	Schmitt trigger input, falling edge external interrupt with pull-up
0	1	N-CH open drain output mode with pull-up
1	0	N-CH open drain output mode
1	1	Output pull-pull mode

.5 and .4 **Port 4, P4.2 Configuration Control Bits**

0	0	Schmitt trigger input, falling edge external interrupt with pull-up
0	1	N-CH open drain output mode with pull-up
1	0	N-CH open drain output mode
1	1	Output pull-pull mode

.3 and .2 **Port 4, P4.1 Configuration Control Bits**

0	0	Schmitt trigger input, falling edge external interrupt with pull-up
0	1	N-CH open drain output mode with pull-up
1	0	N-CH open drain output mode
1	1	Output pull-pull mode

.1 and .0 **Port 4, P4.0 Configuration Control Bits**

0	0	Schmitt trigger input, falling edge external interrupt with pull-up
0	1	N-CH open drain output mode with pull-up
1	0	N-CH open drain output mode
1	1	Output pull-pull mode

P4INTPND — Port 4 Interrupt Enable and Pending Register EFH

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

.7 **P4.3 Interrupt Enable Bit**

0	External interrupt disable
1	External interrupt enable

.6 **P4.2 Interrupt Enable Bit**

0	External interrupt disable
1	External interrupt enable

.5 **P4.1 Interrupt Enable Bit**

0	External interrupt disable
1	External interrupt enable

.4 **P4.0 Interrupt Enable Bit**

0	External interrupt disable
1	External interrupt enable

.3 **P4.3 Interrupt Pending Bit**

0	No pending (when bit is read)/clear pending bit (when bit is write)
1	Pending (when bit is read)/no effect (when bit is write)

.2 **P4.2 Interrupt Pending Bit**

0	No pending (when bit is read)/clear pending bit (when bit is write)
1	Pending (when bit is read)/no effect (when bit is write)

.1 **P4.1 Interrupt Pending Bit**

0	No pending (when bit is read)/clear pending bit (when bit is write)
1	Pending (when bit is read)/no effect (when bit is write)

.0 **P4.0 Interrupt Pending Bit**

0	No pending (when bit is read)/clear pending bit (when bit is write)
1	Pending (when bit is read)/no effect (when bit is write)

PWRMGR — USB Power Management Register F8H

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

.7–.2 Not used for KS86C6004/C6008/P6008

.1 **RESUME Signal Sending Bit**

0	RESUME signal is ended
1	While in suspend state, if the MCU wants to initiate a resume, it writes a 1 to this register for 10ms (maximum of 15ms), and clears this register. In suspend mode if this bit is a 1, USB generates resume signaling.

.0 **SUSPEND Status Bit**

0	Cleared when MCU writes a zero to SEND_RESUME or function receives resume signal from the host while in suspend mode
1	This bit is set when SUSPEND interrupt occur

SYM — System Mode Register DFH

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	–	–	–	–	0	0	0
Read/Write	–	–	–	–	–	R/W	R/W	R/W

.7–.3

Not used for KS86C6004/C6008/P6008

.2

Global Interrupt Enable Bit (note)

0	Disable global interrupt processing
1	Enable global interrupt processing

.1 and .0

Page Selection Bits

0	0	Addressing page 0 locations for KS86C6004/C6008/P6008
Other values		Enable global interrupt processing

NOTE: SYM must be selected bit 1 and 0 into 00 for KS86C6004/C6008/P6008.

T0CON — Timer 0 Control Register D2H

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

.7 and .6 **T0 Counter Input Clock Selection Bits**

0	0	CPU clock/4096
0	1	CPU clock/256
1	0	CPU clock/8
1	1	Invalid selection

.5 and .4 **T0 Operating Mode Selection Bits**

0	0	Interval timer mode (The counter is automatically cleared whenever T0DATA value equals to T0CNT value)
0	1	Invalid selection
1	0	
1	1	Overflow mode (OVF interrupt can occur)

.3 **T0 Counter Clear Bit (T0CLR)**

0	No effect when written
1	Clear T0 counter

.2 **T0 Overflow Interrupt Enable Bit (T0OVF)**

0	Disable T0 overflow interrupt
1	Enable T0 overflow interrupt

.1 **T0 Match Interrupt Enable Bit (T0INT)**

0	Disable T0 match interrupt
1	Enable T0 match interrupt

.0 **T0 Interrupt Pending Bit (T0PND)**

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

NOTE: When you write a "1" to T0CON.3, the timer 0 counter is cleared. The bit is then cleared automatically to "0".

USBPND — USB Interrupt Pending Register F6H

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

.7–.4 Not used for KS86C6004/C6008/P6008

.3 **RESUME Interrupt Pending Bit**

0	No effect (once read, this bit is cleared automatically)
1	While in suspend mode, if resume signaling is received this bit gets set

.2 **SUSPEND Interrupt Pending Bit**

0	No effect (once read, this bit is cleared automatically)
1	This bit is set, when suspend signaling is received

.1 **ENDPOINT1 Interrupt Pending Bit**

0	No effect (once read, this bit is cleared automatically)
1	This bit is set, when endpoint1 needs to be serviced

.0 **ENDPOINT1 Interrupt Pending Bit**

0	No effect (once read, this bit is cleared automatically)
1	This bit is set, while endpoint 0 needs to serviced. It is set under the following conditions; — OUT_PKT_RDY is set — IN_PKT_STALL get cleared — SENT_STALL gets set — DATA_END gets cleared — SETUP_END gets set

USBINT — USB Interrupt Enable Register F7H

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7–.3 Not used for KS86C6004/C6008/P6008

.2 **SUSPEND/RESUME Interrupt Enable Bit**

0	Disable SUSPEND and RESEME interrupt
1	Enable SUSPEND and RESEME interrupt

.1 **ENDPOINT1 Interrupt Pending Bit**

0	Disable ENDPOINT 1 interrupt
1	Enable ENDPOINT 1 interrupt

.0 **ENDPOINT1 Interrupt Pending Bit**

0	Disable ENDPOINT 0 interrupt
1	Enable ENDPOINT 0 interrupt

USBRST — USB RESET Register FFH

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	–	–	–	–	–	–	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7–.1 Not used for KS86C6004/C6008/P6008

.0 **USB Reset Signal Receive Bit**

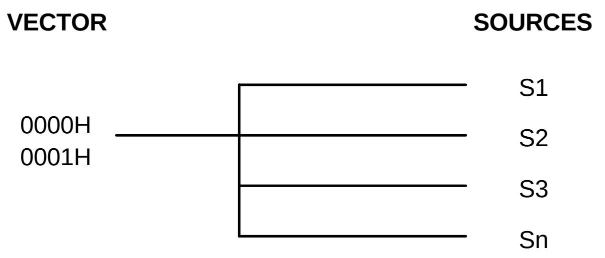
0	No effect (this is automatically cleared once read)
1	This bit is set when host send USB reset signal

NOTES

5 INTERRUPT STRUCTURE

OVERVIEW

The SAM87RI interrupt structure has two basic components: a vector, and sources. The number of interrupt sources can be serviced through a interrupt vector which is assigned in ROM address 0000H–0001H.



NOTES:
1. The SAM87RI interrupt has only one vector address (0000H-0001H)
2. The number of Sn value is expandable.

Figure 5-1. KS86-Series Interrupt Type

INTERRUPT PROCESSING CONTROL POINTS

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

— Global interrupt enable and disable (by EI and DI instructions)

— Interrupt source enable and disable settings in the corresponding peripheral control register(s)

ENABLE/DISABLE INTERRUPT INSTRUCTIONS (EI, DI)

The system mode register, SYM (DFH), is used to enable and disable interrupt processing.

SYM.2 is the enable and disable bit for global interrupt processing respectively, by modifying SYM.2. An Enable Interrupt (EI) instruction must be included in the initialization routine that follows a reset operation in order to enable interrupt processing. Although you can manipulate SYM.2 directly to enable and disable interrupts during normal operation, we recommend that you use the EI and DI instructions for this purpose.

INTERRUPT PENDING FUNCTION TYPES

When the interrupt service routine has executed, the application program's service routine must clear the appropriate pending bit before the return from interrupt subroutine (IRET) occurs.

INTERRUPT PRIORITY

Because there is not a interrupt priority register in SAM87RI, the order of service is determined by a sequence of source which is executed in interrupt service routine.

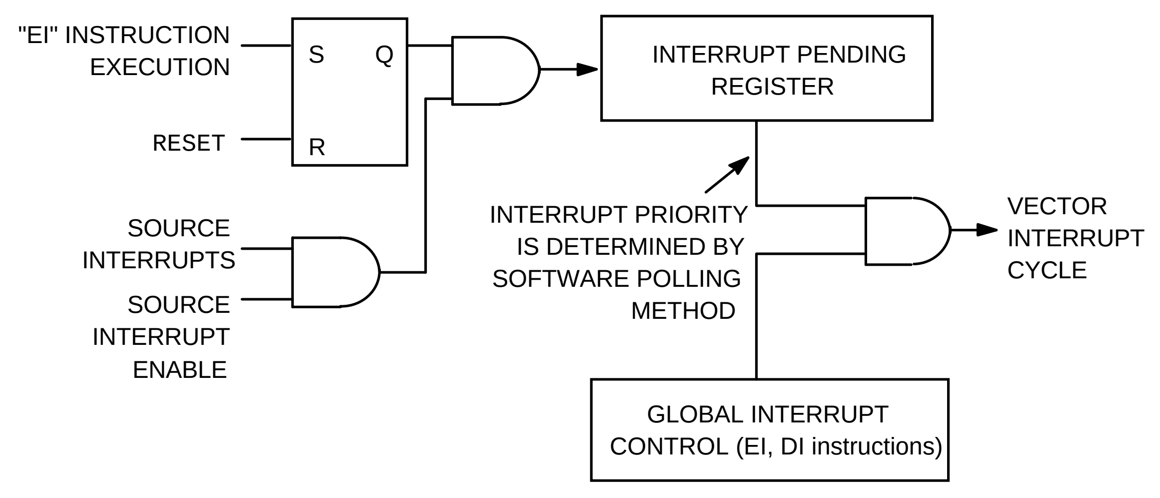


Figure 5-2. Interrupt Function Diagram

INTERRUPT SOURCE SERVICE SEQUENCE

The interrupt request polling and servicing sequence is as follows:

1. A source generates an interrupt request by setting the interrupt request pending bit to "1".
2. The CPU generates an interrupt acknowledge signal.
3. The service routine starts and the source's pending flag is cleared to "0" by software.
4. Interrupt priority must be determined by software polling method.

INTERRUPT SERVICE ROUTINES

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

— Interrupt processing must be enabled (EI, SYM.2 = "1")

— 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 global interrupt enable bit in the SYM register (DI, SYM.2 = "0") to disable all subsequent interrupts.
2. Save the program counter and status flags to stack.
3. Branch to the interrupt vector to fetch the service routine's address.
4. Pass control to the interrupt service routine.

When the interrupt service routine is completed, an Interrupt Return instruction (IRET) occurs. The IRET restores the PC and status flags and sets SYM.2 to "1"(EI), allowing the CPU to process the next interrupt request.

GENERATING INTERRUPT VECTOR ADDRESSES

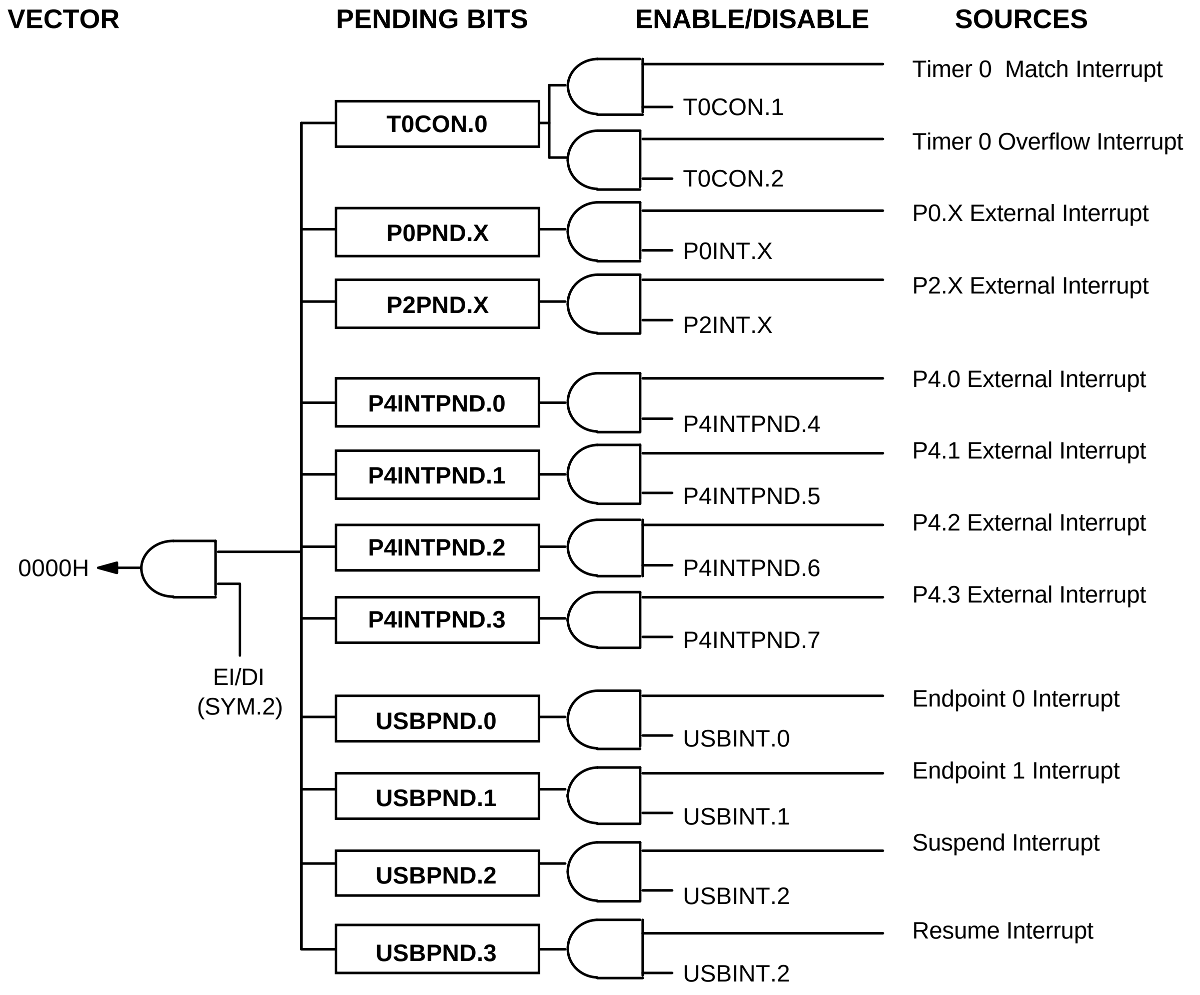
The interrupt vector area in the ROM contains the address of the interrupt service routine. Vectored interrupt processing follows this sequence:

1. Push the program counter's low-byte value to stack.
2. Push the program counter's high-byte value to stack.
3. Push the FLAGS register values to stack.
4. Fetch the service routine's high-byte address from the vector address 0000H.
5. Fetch the service routine's low-byte address from the vector address 0001H.
6. Branch to the service routine specified by the 16-bit vector address.

KS86C6004/C6008/P6008 INTERRUPT STRUCTURE

The KS86C6004/C6008/P6008 microcontroller has fourteen peripheral interrupt sources:

- Timer 0 match interrupt
- Timer 0 overflow interrupt
- Eight external interrupts for port 2, P2.0–P2.7
- Four external interrupts for port 4, P4.0–P4.3



NOTE: 백□means 0-7 bit.

Figure 5-3. KS86C6004/C6008/P6008 Interrupt Structure

6 SAM87RI Instruction Set

OVERVIEW

The SAM87RI instruction set is designed to support the large register file. It includes a full complement of 8-bit arithmetic and logic operations. There are 41 instructions. No special I/O instructions are necessary because I/O control and data registers are mapped directly into the register file. Flexible instructions for bit addressing, rotate, and shift operations complete the powerful data manipulation capabilities of the SAM87RI instruction set.

REGISTER ADDRESSING

To access an individual register, an 8-bit address in the range 0-255 or the 4-bit address of a working register is specified. Paired registers can be used to construct 13-bit program memory or data memory addresses. For detailed information about register addressing, please refer to Section 2, "Address Spaces".

ADDRESSING MODES

There are six addressing modes: Register (R), Indirect Register (IR), Indexed (X), Direct (DA), Relative (RA), and Immediate (IM). For detailed descriptions of these addressing modes, please refer to Section 3, "Addressing Modes".

Table 6-1. Instruction Group Summary

Mnemonic	Operands	Instruction
Load Instructions		
CLR	dst	Clear
LD	dst,src	Load
LDC	dst,src	Load program memory
LDCD	dst,src	Load program memory and decrement
LDED	dst,src	Load external data memory and decrement
LDCI	dst,src	Load program memory and increment
LDEI	dst,src	Load external data memory and increment
POP	dst	Pop from stack
PUSH	src	Push to stack
Arithmetic Instructions		
ADC	dst,src	Add with carry
ADD	dst,src	Add
CP	dst,src	Compare
DEC	dst	Decrement
INC	dst	Increment
SBC	dst,src	Subtract with carry
SUB	dst,src	Subtract
Logic Instructions		
AND	dst,src	Logical AND
COM	dst	Complement
OR	dst,src	Logical OR
XOR	dst,src	Logical exclusive OR

Table 6-1. Instruction Group Summary (Continued)

Mnemonic	Operands	Instruction
Program Control Instructions		
CALL	dst	Call procedure
IRET		Interrupt return
JP	cc,dst	Jump on condition code
JP	dst	Jump unconditional
JR	cc,dst	Jump relative on condition code
RET		Return
Bit Manipulation Instructions		
TCM	dst,src	Test complement under mask
TM	dst,src	Test under mask
Rotate and Shift Instructions		
RL	dst	Rotate left
RLC	dst	Rotate left through carry
RR	dst	Rotate right
RRC	dst	Rotate right through carry
SRA	dst	Shift right arithmetic
CPU Control Instructions		
CCF		Complement carry flag
DI		Disable interrupts
EI		Enable interrupts
IDLE		Enter Idle mode
NOP		No operation
RCF		Reset carry flag
SCF		Set carry flag
STOP		Enter Stop mode

FLAGS REGISTER (FLAGS)

The flags register FLAGS contains eight bits that describe the current status of CPU operations. Four of these bits, FLAGS.4–FLAGS.7, can be tested and used with conditional jump instructions.

FLAGS register can be set or reset by instructions as long as its outcome does not affect the flags, such as, Load instruction. Logical and Arithmetic instructions such as, AND, OR, XOR, ADD, and SUB can affect the Flags register. For example, the AND instruction updates the Zero, Sign and Overflow flags based on the outcome of the AND instruction. If the AND instruction uses the Flags register as the destination, then simultaneously, two write will occur to the Flags register producing an unpredictable result.

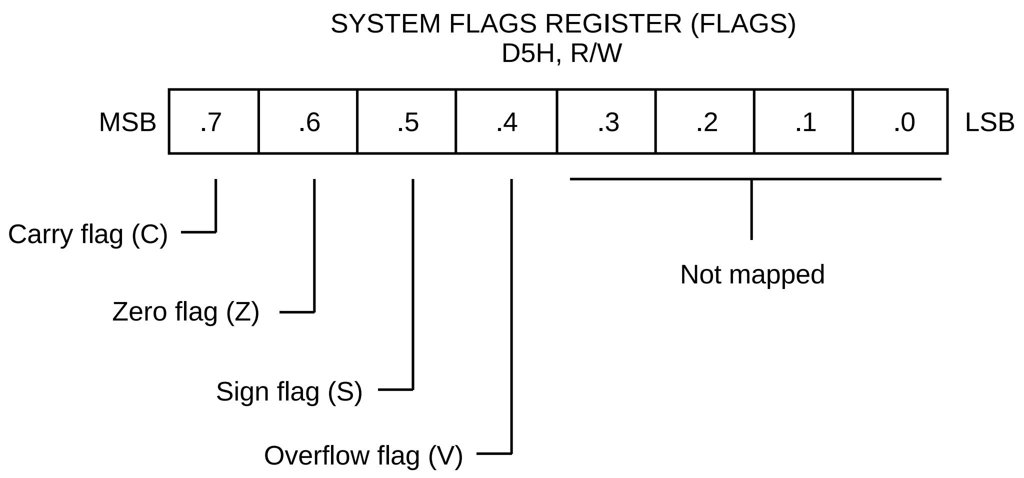


Figure 6-1. System Flags Register (FLAGS)

FLAG DESCRIPTIONS

Overflow Flag (FLAGS.4, V)

The V flag is set to "1" when the result of a two's-complement operation is greater than + 127 or less than – 128. It is also cleared to "0" following logic operations.

Sign Flag (FLAGS.5, S)

Following arithmetic, logic, rotate, or shift operations, the sign bit identifies the state of the MSB of the result. A logic zero indicates a positive number and a logic one indicates a negative number.

Zero Flag (FLAGS.6, Z)

For arithmetic and logic operations, the Z flag is set to "1" if the result of the operation is zero. For operations that test register bits, and for shift and rotate operations, the Z flag is set to "1" if the result is logic zero.

Carry Flag (FLAGS.7, C)

The C flag is set to "1" if the result from an arithmetic operation generates a carry-out from or a borrow to the bit 7 position (MSB). After rotate and shift operations, it contains the last value shifted out of the specified register. Program instructions can set, clear, or complement the carry flag.

INSTRUCTION SET NOTATION

Table 6-2. Flag Notation Conventions

Flag	Description
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
0	Cleared to logic zero
1	Set to logic one
*	Set or cleared according to operation
–	Value is unaffected
x	Value is undefined

Table 6-3. Instruction Set Symbols

Symbol	Description
dst	Destination operand
src	Source operand
@	Indirect register address prefix
PC	Program counter
FLAGS	Flags register (D5H)
#	Immediate operand or register address prefix
H	Hexadecimal number suffix
D	Decimal number suffix
B	Binary number suffix
opc	Opcode

Table 6-4. Instruction Notation Conventions

Notation	Description	Actual Operand Range
cc	Condition code	See list of condition codes in Table 6-6.
r	Working register only	Rn (n = 0–15)
rr	Working register pair	RRp (p = 0, 2, 4, ..., 14)
R	Register or working register	reg or Rn (reg = 0–255, n = 0–15)
RR	Register pair or working register pair	reg or RRp (reg = 0–254, even number only, where p = 0, 2, ..., 14)
Ir	Indirect working register only	@Rn (n = 0–15)
IR	Indirect register or indirect working register	@Rn or @reg (reg = 0–255, n = 0–15)
Irr	Indirect working register pair only	@RRp (p = 0, 2, ..., 14)
IRR	Indirect register pair or indirect working register pair	@RRp or @reg (reg = 0–254, even only, where p = 0, 2, ..., 14)
X	Indexed addressing mode	#reg[Rn] (reg = 0–255, n = 0–15)
XS	Indexed (short offset) addressing mode	#addr[RRp] (addr = range –128 to +127, where p = 0, 2, ..., 14)
XL	Indexed (long offset) addressing mode	#addr [RRp] (addr = range 0–8191, where p = 0, 2, ..., 14)
DA	Direct addressing mode	addr (addr = range 0–8191)
RA	Relative addressing mode	addr (addr = number in the range +127 to –128 that is an offset relative to the address of the next instruction)
IM	Immediate addressing mode	#data (data = 0–255)

Table 6-5. Opcode Quick Reference

OPCODE MAP									
LOWER NIBBLE (HEX)									
	—	0	1	2	3	4	5	6	7
U	0	DEC R1	DEC IR1	ADD r1,r2	ADD r1,Ir2	ADD R2,R1	ADD IR2,R1	ADD R1,IM	
P	1	RLC R1	RLC IR1	ADC r1,r2	ADC r1,Ir2	ADC R2,R1	ADC IR2,R1	ADC R1,IM	
P	2	INC R1	INC IR1	SUB r1,r2	SUB r1,Ir2	SUB R2,R1	SUB IR2,R1	SUB R1,IM	
E	3	JP IRR1		SBC r1,r2	SBC r1,Ir2	SBC R2,R1	SBC IR2,R1	SBC R1,IM	
R	4			OR r1,r2	OR r1,Ir2	OR R2,R1	OR IR2,R1	OR R1,IM	
	5	POP R1	POP IR1	AND r1,r2	AND r1,Ir2	AND R2,R1	AND IR2,R1	AND R1,IM	
N	6	COM R1	COM IR1	TCM r1,r2	TCM r1,Ir2	TCM R2,R1	TCM IR2,R1	TCM R1,IM	
I	7	PUSH R2	PUSH IR2	TM r1,r2	TM r1,Ir2	TM R2,R1	TM IR2,R1	TM R1,IM	
B	8								LD r1, x, r2
B	9	RL R1	RL IR1						LD r2, x, r1
L	A			CP r1,r2	CP r1,Ir2	CP R2,R1	CP IR2,R1	CP R1,IM	LDC r1, Irr2, xL
E	B	CLR R1	CLR IR1	XOR r1,r2	XOR r1,Ir2	XOR R2,R1	XOR IR2,R1	XOR R1,IM	LDC r2, Irr2, xL
	C	RRC R1	RRC IR1		LDC r1,Irr2				LD r1, Ir2
H	D	SRA R1	SRA IR1		LDC r2,Irr1			LD IR1,IM	LD Ir1, r2
E	E	RR R1	RR IR1	LDCD r1,Irr2	LDCI r1,Irr2	LD R2,R1	LD R2,IR1	LD R1,IM	LDC r1, Irr2, xs
X	F					CALL IRR1	LD IR2,R1	CALL DA1	LDC r2, Irr1, xs

Table 6-5. Opcode Quick Reference (Continued)

OPCODE MAP									
LOWER NIBBLE (HEX)									
	—	8	9	A	B	C	D	E	F
U	0	LD r1,R2	LD r2,R1		JR cc,RA	LD r1,IM	JP cc,DA	INC r1	
P	1	↓	↓		↓	↓	↓	↓	
P	2								
E	3								
R	4								
	5								
N	6								IDLE
I	7	↓	↓		↓	↓	↓	↓	STOP
B	8								DI
B	9								EI
L	A								RET
E	B								IRET
	C								RCF
H	D	↓	↓		↓	↓	↓	↓	SCF
E	E								CCF
X	F	LD r1,R2	LD r2,R1		JR cc,RA	LD r1,IM	JP cc,DA	INC r1	NOP

CONDITION CODES

The opcode of a conditional jump always contains a 4-bit field called the condition code (cc). This specifies under which conditions it is to execute the jump. For example, a conditional jump with the condition code for "equal" after a compare operation only jumps if the two operands are equal. Condition codes are listed in Table 6-6.

The carry (C), zero (Z), sign (S), and overflow (V) flags are used to control the operation of conditional jump instructions.

Table 6-6. Condition Codes

Binary	Mnemonic	Description	Flags Set
0000	F	Always false	–
1000	T	Always true	–
0111 *	C	Carry	C = 1
1111 *	NC	No carry	C = 0
0110 *	Z	Zero	Z = 1
1110 *	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110 *	EQ	Equal	Z = 1
1110 *	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	(Z OR (S XOR V)) = 0
0010	LE	Less than or equal	(Z OR (S XOR V)) = 1
1111 *	UGE	Unsigned greater than or equal	C = 0
0111 *	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1

NOTES:
1. Asterisks (*) indicate condition codes that are related to two different mnemonics but which test the same flag. For example, Z and EQ are both true if the zero flag (Z) is set, but after an ADD instruction, Z would probably be used; after a CP instruction, however, EQ would probably be used.
2. For operations involving unsigned numbers, the special condition codes UGE, ULT, UGT, and ULE must be used.

INSTRUCTION DESCRIPTIONS

This section contains detailed information and programming examples for each instruction in the SAM87RI instruction set. Information is arranged in a consistent format for improved readability and for fast referencing. The following information is included in each instruction description:

- Instruction name (mnemonic)
- Full instruction name
- Source/destination format of the instruction operand
- Shorthand notation of the instruction's operation
- Textual description of the instruction's effect
- Specific flag settings affected by the instruction
- Detailed description of the instruction's format, execution time, and addressing mode(s)
- Programming example(s) explaining how to use the instruction

ADC — Add With Carry

ADC dst,src

Operation: dst ← dst + src + c

The source operand, along with the setting of the carry flag, is added to the destination operand and the sum is stored in the destination. The contents of the source are unaffected. Two's-complement addition is performed. In multiple precision arithmetic, this instruction permits the carry from the addition of low-order operands to be carried into the addition of high-order operands.

Flags:

C: Set if there is a carry from the most significant bit of the result; cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurs, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise.
D: Always cleared to "0".
H: Set if there is a carry from the most significant bit of the low-order four bits of the result; cleared otherwise.

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	6	12	r	r
			13	r	Ir
opc | src | dst	3	10	14	R	R
			15	R	IR
opc | dst | src	3	10	16	R	IM

Examples: Given: R1 = 10H, R2 = 03H, C flag = "1", register 01H = 20H, register 02H = 03H, and register 03H = 0AH:

ADC	R1,R2	→	R1 = 14H, R2 = 03H
ADC	R1,@R2	→	R1 = 1BH, R2 = 03H
ADC	01H,02H	→	Register 01H = 24H, register 02H = 03H
ADC	01H,@02H	→	Register 01H = 2BH, register 02H = 03H
ADC	01H,#11H	→	Register 01H = 32H

In the first example, destination register R1 contains the value 10H, the carry flag is set to "1", and the source working register R2 contains the value 03H. The statement "ADC R1,R2" adds 03H and the carry flag value ("1") to the destination value 10H, leaving 14H in register R1.

ADD — Add

ADD dst,src

Operation: dst ← dst + src

The source operand is added to the destination operand and the sum is stored in the destination. The contents of the source are unaffected. Two's-complement addition is performed.

Flags:

C: Set if there is a carry from the most significant bit of the result; cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise.
D: Always cleared to "0".
H: Set if a carry from the low-order nibble occurred.

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	6	02	r	r
			03	r	Ir
opc | src | dst	3	10	04	R	R
			05	R	IR
opc | dst | src	3	10	06	R	IM

Examples: Given: R1 = 12H, R2 = 03H, register 01H = 21H, register 02H = 03H, register 03H = 0AH:

ADD	R1,R2	→	R1 = 15H, R2 = 03H
ADD	R1,@R2	→	R1 = 1CH, R2 = 03H
ADD	01H,02H	→	Register 01H = 24H, register 02H = 03H
ADD	01H,@02H	→	Register 01H = 2BH, register 02H = 03H
ADD	01H,#25H	→	Register 01H = 46H

In the first example, destination working register R1 contains 12H and the source working register R2 contains 03H. The statement "ADD R1,R2" adds 03H to 12H, leaving the value 15H in register R1.

AND — Logical AND

AND dst,src

Operation: dst ← dst AND src

The source operand is logically ANDed with the destination operand. The result is stored in the destination. The AND operation results in a "1" bit being stored whenever the corresponding bits in the two operands are both logic ones; otherwise a "0" bit value is stored. The contents of the source are unaffected.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always cleared to "0".
D: Unaffected.
H: Unaffected.

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	6	52	r	r
			53	r	Ir
opc | src | dst	3	10	54	R	R
			55	R	IR
opc | dst | src	3	10	56	R	IM

Examples: Given: R1 = 12H, R2 = 03H, register 01H = 21H, register 02H = 03H, register 03H = 0AH:

AND	R1,R2	→	R1 = 02H, R2 = 03H
AND	R1,@R2	→	R1 = 02H, R2 = 03H
AND	01H,02H	→	Register 01H = 01H, register 02H = 03H
AND	01H,@02H	→	Register 01H = 00H, register 02H = 03H
AND	01H,#25H	→	Register 01H = 21H

In the first example, destination working register R1 contains the value 12H and the source working register R2 contains 03H. The statement "AND R1,R2" logically ANDs the source operand 03H with the destination operand value 12H, leaving the value 02H in register R1.

CALL — Call Procedure

CALL dst

Operation:

SP	←	SP–1
@SP	←	PCL
SP	←	SP–1
@SP	←	PCH
PC	←	dst

The current contents of the program counter are pushed onto the top of the stack. The program counter value used is the address of the first instruction following the CALL instruction. The specified destination address is then loaded into the program counter and points to the first instruction of a procedure. At the end of the procedure the return instruction (RET) can be used to return to the original program flow. RET pops the top of the stack back into the program counter.

Flags: No flags are affected.

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	3	18	F6	DA
opc | dst	2	18	F4	IRR

Examples: Given: R0 = 15H, R1 = 21H, PC = 1A47H, and SP = 0B2H:

CALL 1521H → SP = 0B0H
(Memory locations 00H = 1AH, 01H = 4AH, where 4AH is the address that follows the instruction.)

CALL @RR0 → SP = 0B0H (00H = 1AH, 01H = 49H)

In the first example, if the program counter value is 1A47H and the stack pointer contains the value 0B2H, the statement "CALL 1521H" pushes the current PC value onto the top of the stack. The stack pointer now points to memory location 00H. The PC is then loaded with the value 1521H, the address of the first instruction in the program sequence to be executed.

If the contents of the program counter and stack pointer are the same as in the first example, the statement "CALL @RR0" produces the same result except that the 49H is stored in stack location 01H (because the two-byte instruction format was used). The PC is then loaded with the value 1521H, the address of the first instruction in the program sequence to be executed.

CCF — Complement Carry Flag

CCF

Operation: C ← NOT C

The carry flag (C) is complemented. If C = "1", the value of the carry flag is changed to logic zero; if C = "0", the value of the carry flag is changed to logic one.

Flags: **C:** Complemented.

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	6	EF

Example: Given: The carry flag = "0":

CCF

If the carry flag = "0", the CCF instruction complements it in the FLAGS register (0D5H), changing its value from logic zero to logic one.

CLR — Clear

CLR dst

Operation: dst ← "0"

The destination location is cleared to "0".

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	6	B0	R
			B1	IR

Examples: Given: Register 00H = 4FH, register 01H = 02H, and register 02H = 5EH:

CLR	00H	→	Register 00H = 00H
CLR	@01H	→	Register 01H = 02H, register 02H = 00H

In Register (R) addressing mode, the statement "CLR 00H" clears the destination register 00H value to 00H. In the second example, the statement "CLR @01H" uses Indirect Register (IR) addressing mode to clear the 02H register value to 00H.

SAMSUNG ELECTRONICS

COM — Complement

COM dst

Operation: dst ← NOT dst

The contents of the destination location are complemented (one's complement); all "1s" are changed to "0s", and vice-versa.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always reset to "0".
D: Unaffected.
H: Unaffected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc	dst	2	6	60	R
				61	IR

Examples: Given: R1 = 07H and register 07H = 0F1H:

COM	R1	→	R1 = 0F8H
COM	@R1	→	R1 = 07H, register 07H = 0EH

In the first example, destination working register R1 contains the value 07H (00000111B). The statement "COM R1" complements all the bits in R1: all logic ones are changed to logic zeros, and vice-versa, leaving the value 0F8H (11111000B).

In the second example, Indirect Register (IR) addressing mode is used to complement the value of destination register 07H (11110001B), leaving the new value 0EH (00001110B).

CP — Compare

CP dst,src

Operation: dst – src

The source operand is compared to (subtracted from) the destination operand, and the appropriate flags are set accordingly. The contents of both operands are unaffected by the comparison.

Flags:
C: Set if a "borrow" occurred (src > dst); cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the operands were of opposite signs and the sign of the result is of the same as the sign of the source operand; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	6	A2	r	r
			A3	r	lr
opc | src | dst	3	10	A4	R	R
			A5	R	IR
opc | dst | src	3	10	A6	R	IM

Examples:

1. Given: R1 = 02H and R2 = 03H:

```
        CP      R1,R2  →    Set the C and S flags
```

Destination working register R1 contains the value 02H and source register R2 contains the value 03H. The statement "CP R1,R2" subtracts the R2 value (source/subtrahend) from the R1 value (destination/minuend). Because a "borrow" occurs and the difference is negative, C and S are "1".

2. Given: R1 = 05H and R2 = 0AH:

```
        CP      R1,R2
        JP      UGE,SKIP
        INC     R1
SKIP    LD      R3,R1
```

In this example, destination working register R1 contains the value 05H which is less than the contents of the source working register R2 (0AH). The statement "CP R1,R2" generates C = "1" and the JP instruction does not jump to the SKIP location. After the statement "LD R3,R1" executes, the value 06H remains in working register R3.

DEC — Decrement

DEC dst

Operation: dst ← dst – 1

The contents of the destination operand are decremented by one.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, dst value is –128(80H) and result value is +127(7FH); cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	6	00	R
			01	IR

Examples: Given: R1 = 03H and register 03H = 10H:

DEC	R1	→	R1 = 02H
DEC	@R1	→	Register 03H = 0FH

In the first example, if working register R1 contains the value 03H, the statement "DEC R1" decrements the hexadecimal value by one, leaving the value 02H. In the second example, the statement "DEC @R1" decrements the value 10H contained in the destination register 03H by one, leaving the value 0FH.

DI — Disable Interrupts

DI

Operation: SYM (2) ← 0

Bit zero of the system mode register, SYM.2, is cleared to "0", globally disabling all interrupt processing. Interrupt requests will continue to set their respective interrupt pending bits, but the CPU will not service them while interrupt processing is disabled.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	6	8F

Example: Given: SYM = 04H:

DI

If the value of the SYM register is 04H, the statement "DI" leaves the new value 00H in the register and clears SYM.2 to "0", disabling interrupt processing.

EI — Enable Interrupts

EI

Operation: SYM (2) ← 1

An EI instruction sets bit 2 of the system mode register, SYM.2 to "1". This allows interrupts to be serviced as they occur. If an interrupt's pending bit was set while interrupt processing was disabled (by executing a DI instruction), it will be serviced when you execute the EI instruction.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	6	9F

Example: Given: SYM = 00H:

EI

If the SYM register contains the value 00H, that is, if interrupts are currently disabled, the statement "EI" sets the SYM register to 04H, enabling all interrupts (SYM.2 is the enable bit for global interrupt processing).

IDLE — Idle Operation

IDLE

Operation:

The IDLE instruction stops the CPU clock while allowing system clock oscillation to continue. Idle mode can be released by an interrupt request (IRQ) or an external reset operation.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	1	3	6F	–	–

Example: The instruction

IDLE

stops the CPU clock but not the system clock.

INC — Increment

INC dst

Operation: dst ← dst + 1

The contents of the destination operand are incremented by one.

Flags:
- **C:** Unaffected.
- **Z:** Set if the result is "0"; cleared otherwise.
- **S:** Set if the result is negative; cleared otherwise.
- **V:** Set if arithmetic overflow occurred, that is dst value is +127(7FH) and result is –128(80H); cleared otherwise.
- **D:** Unaffected.
- **H:** Unaffected.

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
dst | opc	1	6	rE r = 0 to F	r
opc | dst	2	6	20	R
			21	IR

Examples: Given: R0 = 1BH, register 00H = 0CH, and register 1BH = 0FH:

INC	R0	→	R0 = 1CH
INC	00H	→	Register 00H = 0DH
INC	@R0	→	R0 = 1BH, register 01H = 10H

In the first example, if destination working register R0 contains the value 1BH, the statement "INC R0" leaves the value 1CH in that same register.

The next example shows the effect an INC instruction has on register 00H, assuming that it contains the value 0CH.

In the third example, INC is used in Indirect Register (IR) addressing mode to increment the value of register 1BH from 0FH to 10H.

IRET — Interrupt Return

IRET IRET

Operation: FLAGS ← @SP
SP ← SP + 1
PC ← @SP
SP ← SP + 2
SYM(2) ← 1

This instruction is used at the end of an interrupt service routine. It restores the flag register and the program counter. It also re-enables global interrupts.

Flags: All flags are restored to their original settings (that is, the settings before the interrupt occurred).

Format:

IRET (Normal)	Bytes	Cycles	Opcode (Hex)
opc	1	16	BF

JP — Jump

JP cc,dst (Conditional)

JP dst (Unconditional)

Operation: If cc is true, PC ← dst

The conditional JUMP instruction transfers program control to the destination address if the condition specified by the condition code (cc) is true; otherwise, the instruction following the JP instruction is executed. The unconditional JP simply replaces the contents of the PC with the contents of the specified register pair. Control then passes to the statement addressed by the PC.

Flags: No flags are affected.

Format: (1)

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
(2) cc | opc | dst	3	10/12 (3)	ccD cc = 0 to F	DA
opc | dst	2	10	30	IRR

NOTES:
1. The 3-byte format is used for a conditional jump and the 2-byte format for an unconditional jump.
2. In the first byte of the three-byte instruction format (conditional jump), the condition code and the opcode are both four bits.
3. For a conditional jump, execution time is 12 cycles if the jump is taken or 10 cycles if it is not taken.

Examples: Given: The carry flag (C) = "1", register 00 = 01H, and register 01 = 20H:

```
JP    C,LABEL_W   →   LABEL_W = 1000H, PC = 1000H
JP    @00H        →   PC = 0120H
```

The first example shows a conditional JP. Assuming that the carry flag is set to "1", the statement "JP C,LABEL_W" replaces the contents of the PC with the value 1000H and transfers control to that location. Had the carry flag not been set, control would then have passed to the statement immediately following the JP instruction.

The second example shows an unconditional JP. The statement "JP @00" replaces the contents of the PC with the contents of the register pair 00H and 01H, leaving the value 0120H.

JR — Jump Relative

JR cc,dst

Operation: If cc is true, PC ← PC + dst

If the condition specified by the condition code (cc) is true, the relative address is added to the program counter and control passes to the statement whose address is now in the program counter; otherwise, the instruction following the JR instruction is executed (See list of condition codes).

The range of the relative address is +127, –128, and the original value of the program counter is taken to be the address of the first instruction byte following the JR statement.

Flags: No flags are affected.

Format:

(1)		Bytes	Cycles	Opcode (Hex)	Addr Mode dst
cc | opc	dst	2	10/12 (2)	ccB	RA
				cc = 0 to F	

NOTES:
1. In the first byte of the two-byte instruction format, the condition code and the opcode are each four bits.
2. Instruction execution time is 12 cycles if the jump is taken or 10 cycles if it is not taken.

Example: Given: The carry flag = "1" and LABEL_X = 1FF7H:

JR C,LABEL_X → PC = 1FF7H

If the carry flag is set (that is, if the condition code is true), the statement "JR C,LABEL_X" will pass control to the statement whose address is now in the PC. Otherwise, the program instruction following the JR would be executed.

SAMSUNG ELECTRONICS

LD — Load

LD dst,src

Operation: dst ← src

The contents of the source are loaded into the destination. The source's contents are unaffected.

Flags: No flags are affected.

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
dst | opc, src	2	6	rC	r	IM
		6	r8	r	R
src | opc, dst	2	6	r9	R	r
			r = 0 to F		
opc, dst | src	2	6	C7	r	Ir
		6	D7	Ir	r
opc, src, dst	3	10	E4	R	R
		10	E5	R	IR
opc, dst, src	3	10	E6	R	IM
		10	D6	IR	IM
opc, src, dst	3	10	F5	IR	R
opc, dst | src, x	3	10	87	r	x [r]
opc, src | dst, x	3	10	97	x [r]	r

LD — Load

LD (Continued)

Examples: Given: R0 = 01H, R1 = 0AH, register 00H = 01H, register 01H = 20H, register 02H = 02H, LOOP = 30H, and register 3AH = 0FFH:

LD	R0,#10H	→	R0 = 10H
LD	R0,01H	→	R0 = 20H, register 01H = 20H
LD	01H,R0	→	Register 01H = 01H, R0 = 01H
LD	R1,@R0	→	R1 = 20H, R0 = 01H
LD	@R0,R1	→	R0 = 01H, R1 = 0AH, register 01H = 0AH
LD	00H,01H	→	Register 00H = 20H, register 01H = 20H
LD	02H,@00H	→	Register 02H = 20H, register 00H = 01H
LD	00H,#0AH	→	Register 00H = 0AH
LD	@00H,#10H	→	Register 00H = 01H, register 01H = 10H
LD	@00H,02H	→	Register 00H = 01H, register 01H = 02, register 02H = 02H
LD	R0,#LOOP[R1]	→	R0 = 0FFH, R1 = 0AH
LD	#LOOP[R0],R1	→	Register 31H = 0AH, R0 = 01H, R1 = 0AH

LDC/LDE — Load Memory

LDC/LDE dst,src

Operation: dst ← src

This instruction loads a byte from program or data memory into a working register or vice-versa. The source values are unaffected. LDC refers to program memory and LDE to data memory. The assembler makes 'Irr' or 'rr' values an even number for program memory and odd an odd number for data memory.

Flags: No flags are affected.

Format:

	Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
1.	opc | dst | src	2	12	C3	r	Irr
2.	opc | src | dst	2	12	D3	Irr	r
3.	opc | dst | src | XS	3	18	E7	r	XS [rr]
4.	opc | src | dst | XS	3	18	F7	XS [rr]	r
5.	opc | dst | src | XL_L | XL_H	4	20	A7	r	XL [rr]
6.	opc | src | dst | XL_L | XL_H	4	20	B7	XL [rr]	r
7.	opc | dst | 0000 | DA_L | DA_H	4	20	A7	r	DA
8.	opc | src | 0000 | DA_L | DA_H	4	20	B7	DA	r
9.	opc | dst | 0001 | DA_L | DA_H	4	20	A7	r	DA
10.	opc | src | 0001 | DA_L | DA_H	4	20	B7	DA	r

NOTES:
1. The source (src) or working register pair [rr] for formats 5 and 6 cannot use register pair 0–1.
2. For formats 3 and 4, the destination address 'XS [rr]' and the source address 'XS [rr]' are each one byte.
3. For formats 5 and 6, the destination address 'XL [rr] and the source address 'XL [rr]' are each two bytes.
4. The DA and r source values for formats 7 and 8 are used to address program memory; the second set of values, used in formats 9 and 10, are used to address data memory.

LDC/LDE — Load Memory

LDC/LDE (Continued)

Examples: Given: R0 = 11H, R1 = 34H, R2 = 01H, R3 = 04H, R4 = 00H, R5 = 60H; Program memory locations 0061 = AAH, 0103H = 4FH, 0104H = 1A, 0105H = 6DH, and 1104H = 88H. External data memory locations 0061H = BBH, 0103H = 5FH, 0104H = 2AH, 0105H = 7DH, and 1104H = 98H:

```
LDC     R0,@RR2          ; R0 ← contents of program memory location 0104H
                         ; R0 = 1AH, R2 = 01H, R3 = 04H
LDE     R0,@RR2          ; R0 ← contents of external data memory location 0104H
                         ; R0 = 2AH, R2 = 01H, R3 = 04H
LDC *   @RR2,R0          ; 11H (contents of R0) is loaded into program memory
                         ; location 0104H (RR2),
                         ; working registers R0, R2, R3 → no change
LDE     @RR2,R0          ; 11H (contents of R0) is loaded into external data memory
                         ; location 0104H (RR2),
                         ; working registers R0, R2, R3 → no change
LDC     R0,#01H[RR4]     ; R0 ← contents of program memory location 0061H
                         ; (01H + RR4),
                         ; R0 = AAH, R2 = 00H, R3 = 60H
LDE     R0,#01H[RR4]     ; R0 ← contents of external data memory location 0061H
                         ; (01H + RR4), R0 = BBH, R4 = 00H, R5 = 60H
LDC *   #01H[RR4],R0     ; 11H (contents of R0) is loaded into program memory location
                         ; 0061H (01H + 0060H)
LDE     #01H[RR4],R0     ; 11H (contents of R0) is loaded into external data memory
                         ; location 0061H (01H + 0060H)
LDC     R0,#1000H[RR2]   ; R0 ← contents of program memory location 1104H
                         ; (1000H + 0104H), R0 = 88H, R2 = 01H, R3 = 04H
LDE     R0,#1000H[RR2]   ; R0 ← contents of external data memory location 1104H
                         ; (1000H + 0104H), R0 = 98H, R2 = 01H, R3 = 04H
LDC     R0,1104H         ; R0 ← contents of program memory location 1104H, R0 = 88H
LDE     R0,1104H         ; R0 ← contents of external data memory location 1104H,
                         ; R0 = 98H
LDC *   1105H,R0         ; 11H (contents of R0) is loaded into program memory location
                         ; 1105H, (1105H) ← 11H
LDE     1105H,R0         ; 11H (contents of R0) is loaded into external data memory
                         ; location 1105H, (1105H) ← 11H
```

* These instructions are not supported by masked ROM type devices.

LDCD/LDED — Load Memory and Decrement

LDCD/LDED dst,src

Operation: dst ← src

rr ← rr – 1

These instructions are used for user stacks or block transfers of data from program or data memory to the register file. The address of the memory location is specified by a working register pair. The contents of the source location are loaded into the destination location. The memory address is then decremented. The contents of the source are unaffected.

LDCD references program memory and LDED references external data memory. The assembler makes 'Irr' an even number for program memory and an odd number for data memory.

Flags: No flags are affected.

Format:

Format		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | src	2	16	E2	r	Irr

Examples: Given: R6 = 10H, R7 = 33H, R8 = 12H, program memory location 1033H = 0CDH, and external data memory location 1033H = 0DDH:

```
LDCD    R8,@RR6    ; 0CDH (contents of program memory location 1033H) is loaded
                   ; into R8 and RR6 is decremented by one
                   ; R8 = 0CDH, R6 = 10H, R7 = 32H (RR6 ← RR6 – 1)
LDED    R8,@RR6    ; 0DDH (contents of data memory location 1033H) is loaded
                   ; into R8 and RR6 is decremented by one (RR6 ← RR6 – 1)
                   ; R8 = 0DDH, R6 = 10H, R7 = 32H
```

LDCI/LDEI — Load Memory and Increment

LDCI/LDEI dst,src

Operation: dst ← src

rr ← rr + 1

These instructions are used for user stacks or block transfers of data from program or data memory to the register file. The address of the memory location is specified by a working register pair. The contents of the source location are loaded into the destination location. The memory address is then incremented automatically. The contents of the source are unaffected.

LDCI refers to program memory and LDEI refers to external data memory. The assembler makes 'Irr' even for program memory and odd for data memory.

Flags: No flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | src	2	16	E3	r	Irr

Examples: Given: R6 = 10H, R7 = 33H, R8 = 12H, program memory locations 1033H = 0CDH and 1034H = 0C5H; external data memory locations 1033H = 0DDH and 1034H = 0D5H:

```
LDCI   R8,@RR6    ; 0CDH (contents of program memory location 1033H) is loaded
                  ; into R8 and RR6 is incremented by one (RR6 ← RR6 + 1)
                  ; R8 = 0CDH, R6 = 10H, R7 = 34H
LDEI   R8,@RR6    ; 0DDH (contents of data memory location 1033H) is loaded
                  ; into R8 and RR6 is incremented by one (RR6 ← RR6 + 1)
                  ; R8 = 0DDH, R6 = 10H, R7 = 34H
```

NOP — No Operation

NOP

Operation: No action is performed when the CPU executes this instruction. Typically, one or more NOPs are executed in sequence in order to effect a timing delay of variable duration.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	6	FF

Example: When the instruction

NOP

is encountered in a program, no operation occurs. Instead, there is a delay in instruction execution time.

OR — Logical OR

OR dst,src

Operation: dst ← dst OR src

The source operand is logically ORed with the destination operand and the result is stored in the destination. The contents of the source are unaffected. The OR operation results in a "1" being stored whenever either of the corresponding bits in the two operands is a "1"; otherwise a "0" is stored.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always cleared to "0".
D: Unaffected.
H: Unaffected.

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	6	42	r	r
		6	43	r	lr
opc | src | dst	3	10	44	R	R
		10	45	R	IR
opc | dst | src	3	10	46	R	IM

Examples: Given: R0 = 15H, R1 = 2AH, R2 = 01H, register 00H = 08H, register 01H = 37H, and register 08H = 8AH:

OR	R0,R1	→	R0 = 3FH, R1 = 2AH
OR	R0,@R2	→	R0 = 37H, R2 = 01H, register 01H = 37H
OR	00H,01H	→	Register 00H = 3FH, register 01H = 37H
OR	01H,@00H	→	Register 00H = 08H, register 01H = 0BFH
OR	00H,#02H	→	Register 00H = 0AH

In the first example, if working register R0 contains the value 15H and register R1 the value 2AH, the statement "OR R0,R1" logical-ORs the R0 and R1 register contents and stores the result (3FH) in destination register R0.

The other examples show the use of the logical OR instruction with the various addressing modes and formats.

POP — Pop From Stack

POP dst

Operation: dst ← @SP

SP ← SP + 1

The contents of the location addressed by the stack pointer are loaded into the destination. The stack pointer is then incremented by one.

Flags: No flags affected.

Format:

opc	dst

Bytes	Cycles	Opcode (Hex)	Addr Mode dst
2	10	50	R
	10	51	IR

Examples: Given: Register 00H = 01H, register 01H = 1BH, SP (0D9H) = 0BBH, and stack register 0BBH = 55H:

POP	00H	→	Register 00H = 55H, SP = 0BCH
POP	@00H	→	Register 00H = 01H, register 01H = 55H, SP = 0BCH

In the first example, general register 00H contains the value 01H. The statement "POP 00H" loads the contents of location 0BBH (55H) into destination register 00H and then increments the stack pointer by one. Register 00H then contains the value 55H and the SP points to location 0BCH.

PUSH — Push To Stack

PUSH src

Operation: SP ← SP – 1

@SP ← src

A PUSH instruction decrements the stack pointer value and loads the contents of the source (src) into the location addressed by the decremented stack pointer. The operation then adds the new value to the top of the stack.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
opc | src	2	10	70	R
		12	71	IR

Examples: Given: Register 40H = 4FH, register 4FH = 0AAH, SP = 0C0H:

PUSH 40H → Register 40H = 4FH, stack register 0BFH = 4FH, SP = 0BFH

PUSH @40H → Register 40H = 4FH, register 4FH = 0AAH, stack register 0BFH = 0AAH, SP = 0BFH

In the first example, if the stack pointer contains the value 0C0H, and general register 40H the value 4FH, the statement "PUSH 40H" decrements the stack pointer from 0C0 to 0BFH. It then loads the contents of register 40H into location 0BFH. Register 0BFH then contains the value 4FH and SP points to location 0BFH.

RCF — Reset Carry Flag

RCF RCF

Operation: C ← 0

The carry flag is cleared to logic zero, regardless of its previous value.

Flags: **C:** Cleared to "0".

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	6	CF

Example: Given: C = "1" or "0":

The instruction RCF clears the carry flag (C) to logic zero.

RET — Return

RET

Operation: PC ← @SP

SP ← SP + 2

The RET instruction is normally used to return to the previously executing procedure at the end of a procedure entered by a CALL instruction. The contents of the location addressed by the stack pointer are popped into the program counter. The next statement that is executed is the one that is addressed by the new program counter value.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	14	AF

Example: Given: SP = 0BCH, (SP) = 101AH, and PC = 1234:

RET → PC = 101AH, SP = 0BEH

The statement "RET" pops the contents of stack pointer location 0BCH (10H) into the high byte of the program counter. The stack pointer then pops the value in location 0BDH (1AH) into the PC's low byte and the instruction at location 101AH is executed. The stack pointer now points to memory location 0BEH.

RL — Rotate Left

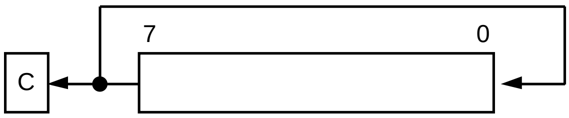
RL dst

Operation: C ← dst (7)

dst (0) ← dst (7)

dst (n + 1) ← dst (n), n = 0–6

The contents of the destination operand are rotated left one bit position. The initial value of bit 7 is moved to the bit zero (LSB) position and also replaces the carry flag.



Flags:

C: Set if the bit rotated from the most significant bit position (bit 7) was "1".
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc	dst	2	6	90	R
			6	91	IR

Examples: Given: Register 00H = 0AAH, register 01H = 02H and register 02H = 17H:

RL 00H → Register 00H = 55H, C = "1"

RL @01H → Register 01H = 02H, register 02H = 2EH, C = "0"

In the first example, if general register 00H contains the value 0AAH (10101010B), the statement "RL 00H" rotates the 0AAH value left one bit position, leaving the new value 55H (01010101B) and setting the carry and overflow flags.

RLC — Rotate Left Through Carry

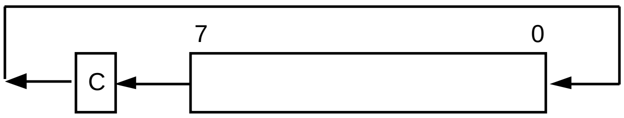
RLC dst

Operation: dst (0) ← C

C ← dst (7)

dst (n + 1) ← dst (n), n = 0–6

The contents of the destination operand with the carry flag are rotated left one bit position. The initial value of bit 7 replaces the carry flag (C); the initial value of the carry flag replaces bit zero.



Flags:
C: Set if the bit rotated from the most significant bit position (bit 7) was "1".
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	6	10	R
		6	11	IR

Examples: Given: Register 00H = 0AAH, register 01H = 02H, and register 02H = 17H, C = "0":

RLC	00H	→	Register 00H = 54H, C = "1"
RLC	@01H	→	Register 01H = 02H, register 02H = 2EH, C = "0"

In the first example, if general register 00H has the value 0AAH (10101010B), the statement "RLC 00H" rotates 0AAH one bit position to the left. The initial value of bit 7 sets the carry flag and the initial value of the C flag replaces bit zero of register 00H, leaving the value 55H (01010101B). The MSB of register 00H resets the carry flag to "1" and sets the overflow flag.

RR — Rotate Right

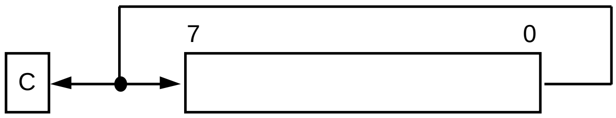
RR dst

Operation: C ← dst (0)

dst (7) ← dst (0)

dst (n) ← dst (n + 1), n = 0–6

The contents of the destination operand are rotated right one bit position. The initial value of bit zero (LSB) is moved to bit 7 (MSB) and also replaces the carry flag (C).



Flags:

C: Set if the bit rotated from the least significant bit position (bit zero) was "1".
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
opc | dst	2	6	E0	R
		6	E1	IR

Examples: Given: Register 00H = 31H, register 01H = 02H, and register 02H = 17H:

RR 00H → Register 00H = 98H, C = "1"

RR @01H → Register 01H = 02H, register 02H = 8BH, C = "1"

In the first example, if general register 00H contains the value 31H (00110001B), the statement "RR 00H" rotates this value one bit position to the right. The initial value of bit zero is moved to bit 7, leaving the new value 98H (10011000B) in the destination register. The initial bit zero also resets the C flag to "1" and the sign flag and overflow flag are also set to "1".

RRC — Rotate Right Through Carry

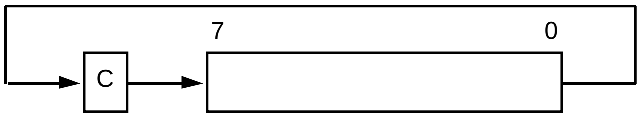
RRC dst

Operation: dst (7) ← C

C ← dst (0)

dst (n) ← dst (n + 1), n = 0–6

The contents of the destination operand and the carry flag are rotated right one bit position. The initial value of bit zero (LSB) replaces the carry flag; the initial value of the carry flag replaces bit 7 (MSB).



Flags:

C: Set if the bit rotated from the least significant bit position (bit zero) was "1".
Z: Set if the result is "0" cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	6	C0	R
		6	C1	IR

Examples: Given: Register 00H = 55H, register 01H = 02H, register 02H = 17H, and C = "0":

RRC	00H	→	Register 00H = 2AH, C = "1"
RRC	@01H	→	Register 01H = 02H, register 02H = 0BH, C = "1"

In the first example, if general register 00H contains the value 55H (01010101B), the statement "RRC 00H" rotates this value one bit position to the right. The initial value of bit zero ("1") replaces the carry flag and the initial value of the C flag ("1") replaces bit 7. This leaves the new value 2AH (00101010B) in destination register 00H. The sign flag and overflow flag are both cleared to "0".

SBC — Subtract With Carry

SBC dst,src

Operation: dst ← dst – src – c

The source operand, along with the current value of the carry flag, is subtracted from the destination operand and the result is stored in the destination. The contents of the source are unaffected. Subtraction is performed by adding the two's-complement of the source operand to the destination operand. In multiple precision arithmetic, this instruction permits the carry ("borrow") from the subtraction of the low-order operands to be subtracted from the subtraction of high-order operands.

Flags:

C: Set if a borrow occurred (src > dst); cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the operands were of opposite sign and the sign of the result is the same as the sign of the source; cleared otherwise.
D: Always set to "1".
H: Cleared if there is a carry from the most significant bit of the low-order four bits of the result; set otherwise, indicating a "borrow".

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	6	32	r	r
		6	33	r	Ir
opc | src | dst	3	10	34	R	R
		10	35	R	IR
opc | dst | src	3	10	36	R	IM

Examples: Given: R1 = 10H, R2 = 03H, C = "1", register 01H = 20H, register 02H = 03H, and register 03H = 0AH:

SBC	R1,R2	→	R1 = 0CH, R2 = 03H
SBC	R1,@R2	→	R1 = 05H, R2 = 03H, register 03H = 0AH
SBC	01H,02H	→	Register 01H = 1CH, register 02H = 03H
SBC	01H,@02H	→	Register 01H = 15H,register 02H = 03H, register 03H = 0AH
SBC	01H,#8AH	→	Register 01H = 95H; C, S, and V = "1"

In the first example, if working register R1 contains the value 10H and register R2 the value 03H, the statement "SBC R1,R2" subtracts the source value (03H) and the C flag value ("1") from the destination (10H) and then stores the result (0CH) in register R1.

SCF — Set Carry Flag

SCF

Operation: C ← 1

The carry flag (C) is set to logic one, regardless of its previous value.

Flags: **C:** Set to "1".

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	6	DF

Example: The statement

SCF

sets the carry flag to logic one.

SRA — Shift Right Arithmetic

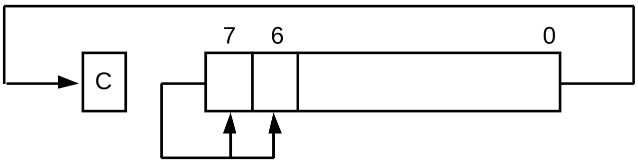
SRA dst

Operation: dst (7) ← dst (7)

C ← dst (0)

dst (n) ← dst (n + 1), n = 0–6

An arithmetic shift-right of one bit position is performed on the destination operand. Bit zero (the LSB) replaces the carry flag. The value of bit 7 (the sign bit) is unchanged and is shifted into bit position 6.



Flags:
C: Set if the bit shifted from the LSB position (bit zero) was "1".
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Always cleared to "0".
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	6	D0	R
		6	D1	IR

Examples: Given: Register 00H = 9AH, register 02H = 03H, register 03H = 0BCH, and C = "1":

SRA 00H → Register 00H = 0CD, C = "0"

SRA @02H → Register 02H = 03H, register 03H = 0DEH, C = "0"

In the first example, if general register 00H contains the value 9AH (10011010B), the statement "SRA 00H" shifts the bit values in register 00H right one bit position. Bit zero ("0") clears the C flag and bit 7 ("1") is then shifted into the bit 6 position (bit 7 remains unchanged). This leaves the value 0CDH (11001101B) in destination register 00H.

STOP — Stop Operation

STOP

Operation:

The STOP instruction stops the both the CPU clock and system clock and causes the microcontroller to enter Stop mode. During Stop mode, the contents of on-chip CPU registers, peripheral registers, and I/O port control and data registers are retained. Stop mode can be released by an external reset operation or External interrupt input. For the reset operation, the RESET pin must be held to Low level until the required oscillation stabilization interval has elapsed.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	1	3	7F	–	–

Example: The statement

STOP

halts all microcontroller operations.

SUB — Subtract

SUB dst,src

Operation: dst ← dst – src

The source operand is subtracted from the destination operand and the result is stored in the destination. The contents of the source are unaffected. Subtraction is performed by adding the two's complement of the source operand to the destination operand.

Flags:

C: Set if a "borrow" occurred; cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the operands were of opposite signs and the sign of the result is of the same as the sign of the source operand; cleared otherwise.
D: Always set to "1".
H: Cleared if there is a carry from the most significant bit of the low-order four bits of the result; set otherwise indicating a "borrow".

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	6	22	r	r
		6	23	r	Ir
opc | src | dst	3	10	24	R	R
		10	25	R	IR
opc | dst | src	3	10	26	R	IM

Examples: Given: R1 = 12H, R2 = 03H, register 01H = 21H, register 02H = 03H, register 03H = 0AH:

SUB	R1,R2	→	R1 = 0FH, R2 = 03H
SUB	R1,@R2	→	R1 = 08H, R2 = 03H
SUB	01H,02H	→	Register 01H = 1EH, register 02H = 03H
SUB	01H,@02H	→	Register 01H = 17H, register 02H = 03H
SUB	01H,#90H	→	Register 01H = 91H; C, S, and V = "1"
SUB	01H,#65H	→	Register 01H = 0BCH; C and S = "1", V = "0"

In the first example, if working register R1 contains the value 12H and if register R2 contains the value 03H, the statement "SUB R1,R2" subtracts the source value (03H) from the destination value (12H) and stores the result (0FH) in destination register R1.

TCM — Test Complement Under Mask

TCM dst,src

Operation: (NOT dst) AND src

This instruction tests selected bits in the destination operand for a logic one value. The bits to be tested are specified by setting a "1" bit in the corresponding position of the source operand (mask). The TCM statement complements the destination operand, which is then ANDed with the source mask. The zero (Z) flag can then be checked to determine the result. The destination and source operands are unaffected.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always cleared to "0".
D: Unaffected.
H: Unaffected.

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	6	62	r	r
		6	63	r	lr
opc | src | dst	3	10	64	R	R
		10	65	R	IR
opc | dst | src	3	10	66	R	IM

Examples: Given: R0 = 0C7H, R1 = 02H, R2 = 12H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

TCM	R0,R1	→	R0 = 0C7H, R1 = 02H, Z = "1"
TCM	R0,@R1	→	R0 = 0C7H, R1 = 02H, register 02H = 23H, Z = "0"
TCM	00H,01H	→	Register 00H = 2BH, register 01H = 02H, Z = "1"
TCM	00H,@01H	→	Register 00H = 2BH, register 01H = 02H, register 02H = 23H, Z = "1"
TCM	00H,#34	→	Register 00H = 2BH, Z = "0"

In the first example, if working register R0 contains the value 0C7H (11000111B) and register R1 the value 02H (00000010B), the statement "TCM R0,R1" tests bit one in the destination register for a "1" value. Because the mask value corresponds to the test bit, the Z flag is set to logic one and can be tested to determine the result of the TCM operation.

TM — Test Under Mask

TM dst,src

Operation: dst AND src

This instruction tests selected bits in the destination operand for a logic zero value. The bits to be tested are specified by setting a "1" bit in the corresponding position of the source operand (mask), which is ANDed with the destination operand. The zero (Z) flag can then be checked to determine the result. The destination and source operands are unaffected.

Flags:

C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always reset to "0".
D: Unaffected.
H: Unaffected.

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	6	72	r	r
		6	73	r	Ir
opc | src | dst	3	10	74	R	R
		10	75	R	IR
opc | dst | src	3	10	76	R	IM

Examples: Given: R0 = 0C7H, R1 = 02H, R2 = 18H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

TM	R0,R1	→	R0 = 0C7H, R1 = 02H, Z = "0"
TM	R0,@R1	→	R0 = 0C7H, R1 = 02H, register 02H = 23H, Z = "0"
TM	00H,01H	→	Register 00H = 2BH, register 01H = 02H, Z = "0"
TM	00H,@01H	→	Register 00H = 2BH, register 01H = 02H, register 02H = 23H, Z = "0"
TM	00H,#54H	→	Register 00H = 2BH, Z = "1"

In the first example, if working register R0 contains the value 0C7H (11000111B) and register R1 the value 02H (00000010B), the statement "TM R0,R1" tests bit one in the destination register for a "0" value. Because the mask value does not match the test bit, the Z flag is cleared to logic zero and can be tested to determine the result of the TM operation.

XOR — Logical Exclusive OR

XOR dst,src

Operation: dst ← dst XOR src

The source operand is logically exclusive-ORed with the destination operand and the result is stored in the destination. The exclusive-OR operation results in a "1" bit being stored whenever the corresponding bits in the operands are different; otherwise, a "0" bit is stored.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always reset to "0".
D: Unaffected.
H: Unaffected.

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	6	B2	r	r
		6	B3	r	Ir
opc | src | dst	3	10	B4	R	R
		10	B5	R	IR
opc | dst | src	3	10	B6	R	IM

Examples: Given: R0 = 0C7H, R1 = 02H, R2 = 18H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

XOR	R0,R1	→	R0 = 0C5H, R1 = 02H
XOR	R0,@R1	→	R0 = 0E4H, R1 = 02H, register 02H = 23H
XOR	00H,01H	→	Register 00H = 29H, register 01H = 02H
XOR	00H,@01H	→	Register 00H = 08H, register 01H = 02H, register 02H = 23H
XOR	00H,#54H	→	Register 00H = 7FH

In the first example, if working register R0 contains the value 0C7H and if register R1 contains the value 02H, the statement "XOR R0,R1" logically exclusive-ORs the R1 value with the R0 value and stores the result (0C5H) in the destination register R0.

Clock Circuit

RESET and Power-Down

I/O Ports

Basic Timer and Timer 0

USB Block

Universal Serial Bus

Electrical Data

Mechanical Data

KS86P6008 OTP

7 CLOCK CIRCUIT

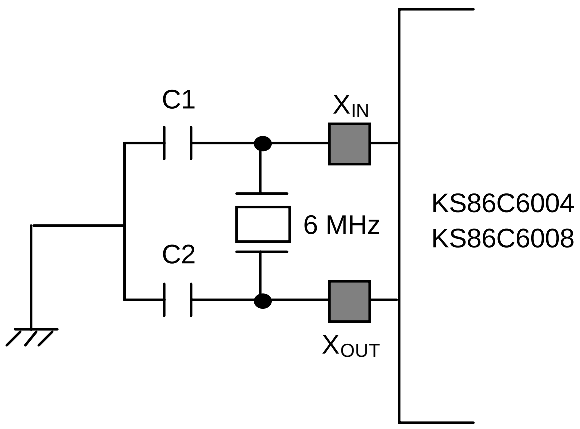


Figure 7-1. Main Oscillator Circuit (Crystal/Ceramic Oscillator)

MAIN OSCILLATOR LOGIC

To increase processing speed and to reduce clock noise, non-divided logic is implemented for the main oscillator circuit. For this reason, very high resolution waveforms (square signal edges) must be generated in order for the CPU to efficiently process logic operations.

CLOCK STATUS DURING POWER-DOWN MODES

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

— In Stop mode, the main oscillator "freezes", halting the CPU and peripherals. The contents of the register file and current system register values are retained. Stop mode is released, and the oscillator started, by a reset operation or by an external interrupt with RC-delay noise filter (for KS86C6004/C6008/P6008, INT0–INT2).

— In Idle mode, the internal clock signal is gated off to the CPU, but not to interrupt control and the timer. The current CPU status is preserved, including stack pointer, program counter, and flags. Data in the register file is retained. Idle mode is released by a reset or by an interrupt (external or internally-generated).

SYSTEM CLOCK CONTROL REGISTER (CLKCON)

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

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

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

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

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

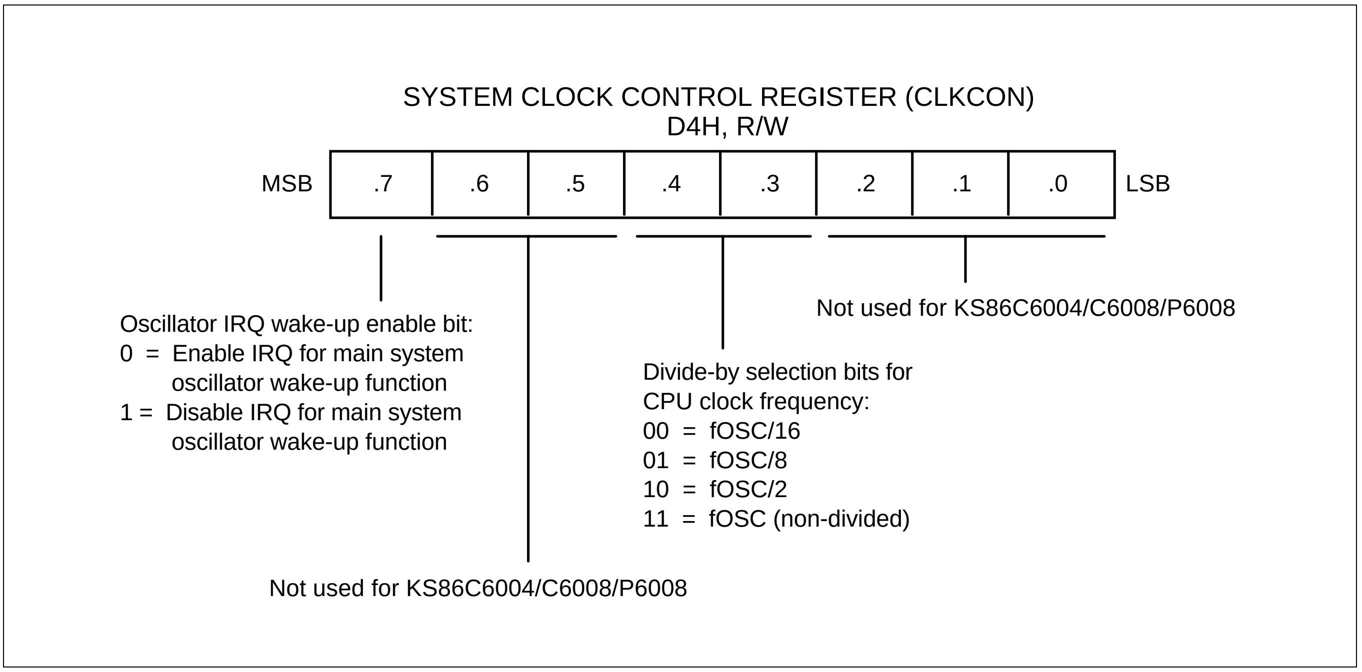


Figure 7-2. System Clock Control Register (CLKCON)

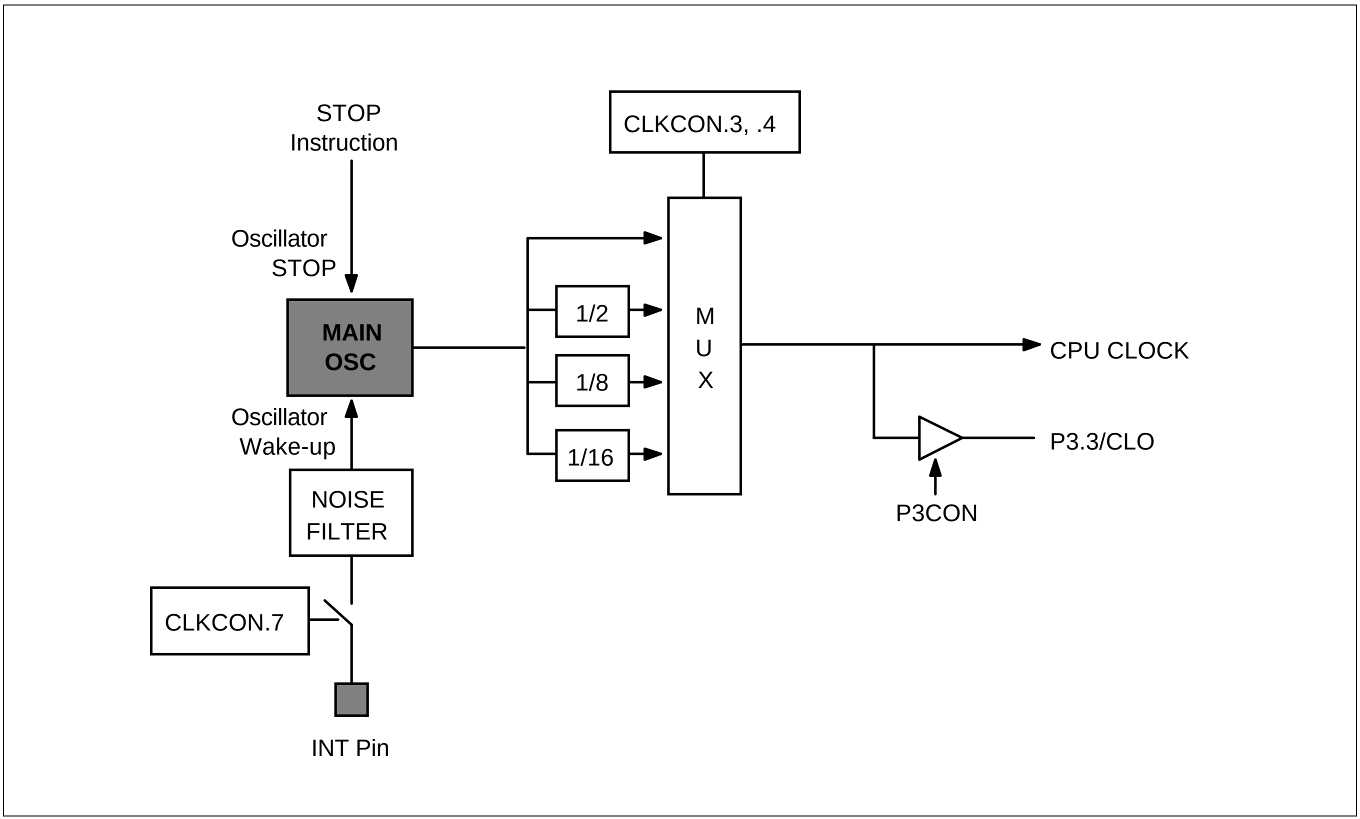


Figure 7-3. System Clock Circuit Diagram

NOTES

8 RESET and POWER-DOWN

SYSTEM RESET

OVERVIEW

During a power-on reset, the voltage at V_{DD} is High level and the RESET pin is forced to Low level. The RESET signal is input through a Schmitt trigger circuit where it is then synchronized with the CPU clock. This brings the KS86C6004/C6008/P6008 into a known operating status.

The RESET pin must be held to Low level for a minimum time interval after the power supply comes within tolerance in order to allow time for internal CPU clock oscillation to stabilize. The minimum required oscillation stabilization time for a reset is approximately 10ms ($\cong 2^{16}/f_{OSC}$, f_{OSC} = 6 MHz).

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

The following sequence of events occurs during a reset operation:

— All interrupts are disabled.

— The watchdog function (basic timer) is enabled.

— Ports 0–4 are set to Schmitt trigger input mode and all pull-up resistors are disabled.

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

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

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

NOTE

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

POWER-DOWN MODES

STOP MODE

Stop mode is invoked by the instruction STOP (opcode 7FH). In Stop mode, the operation of the CPU and all peripherals is halted. That is, the on-chip main oscillator stops and the supply current is reduced to less than 300茄A. All system functions are halted when the clock "freezes", but data stored in the internal register file is retained. Stop mode can be released in one of two ways: by a RESET signal or by an external interrupt.

Using RESET to Release Stop Mode

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

Using an External Interrupt to Release Stop Mode

Only external interrupts with an RC-delay noise filter circuit can be used to release Stop mode (Clock-related external interrupts cannot be used). External interrupts INT0–INT2 in the KS86C6004/C6008/P6008 interrupt structure meet this criteria.

Note that when Stop mode is released by an external interrupt, the current values in system and peripheral control registers are not changed. When you use an interrupt to release Stop mode, the CLKCON.3 and CLKCON.4 register values remain unchanged, and the currently selected clock value is used. If you use an external interrupt for Stop mode release, you can also program the duration of the oscillation stabilization interval. To do this, you must make the appropriate control and clock settings *before* entering Stop mode.

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

IDLE MODE

Idle mode is invoked by the instruction IDLE (opcode 6FH). In Idle mode, CPU operations are halted while select peripherals remain active. During Idle mode, the internal clock signal is gated off to the CPU, but not to interrupt logic and timer/counters. Port pins retain the mode (input or output) they had at the time Idle mode was entered.

There are two ways to release Idle mode:

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

NOTE

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

HARDWARE RESET VALUES

Tables 8-1 through 8-3 list the values for CPU and system registers, peripheral control registers and peripheral data registers following a reset operation in normal operating mode. The following notation is used in these tables to represent specific reset values:

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

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

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

Table 8-1. Register Values After a Reset

Register Name	Mnemonic	Address		Bit Values After RESET							
		Dec	Hex	7	6	5	4	3	2	1	0
General purpose registers	–	000–191	00H–BFH	x	x	x	x	x	x	x	x
Working registers	R0 – R15	192–207	C0H–CFH	x	x	x	x	x	x	x	x
Timer 0 counter	T0CNT	208	D0H	0	0	0	0	0	0	0	0
Timer 0 data register	T0DATA	209	D1H	1	1	1	1	1	1	1	1
Timer 0 control register	T0CON	210	D2H	0	0	0	0	0	0	0	0
Location D3H is not mapped.											
Clock control register	CLKCON	212	D4H	0	0	0	0	0	0	0	0
System flags register	FLAGS	213	D5H	0	0	0	0	–	–	–	–
Locations D6H – D8H are not mapped.											
Port 0 interrupt control register	P0INT	216	D8H	0	0	0	0	0	0	0	0
Stack pointer	SP	217	D9H	x	x	x	x	x	x	x	x
Port 0 interrupt pending register	P0PND	218	DAH	0	0	0	0	0	0	0	0
Location DBH is not mapped.											
Basic timer control register	BTCON	220	DCH	0	0	0	0	0	0	0	0
Basic timer counter	BTCNT	221	DDH	0	0	0	0	0	0	0	0
Location DEH is not mapped.											
System mode register	SYM	223	DFH	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

Table 8-1. Register Values After a Reset (continued)

Bank 0 Register Name	Mnemonic	Address		Bit Values After a Reset							
		Dec	Hex	7	6	5	4	3	2	1	0
Port 3 control register	P3CON	229	E5H	0	0	0	0	0	0	0	0
Port 0 control register (high byte)	P0CONH	230	E6H	0	0	0	0	0	0	0	0
Port 0 control register (low byte)	P0CONL	231	E7H	0	0	0	0	0	0	0	0
Port 1 control register (high byte)	P1CONH	232	E8H	0	0	0	0	0	0	0	0
Port 1 control register (low byte)	P1CONL	233	E9H	0	0	0	0	0	0	0	0
Port 2 control register (high byte)	P2CONH	234	EAH	0	0	0	0	0	0	0	0
Port 2 control register (low byte)	P2CONL	235	EBH	0	0	0	0	0	0	0	0
Port 2 interrupt enable register	P2INT	236	ECH	0	0	0	0	0	0	0	0
Port 2 interrupt pending register	P2PND	237	EDH	0	0	0	0	0	0	0	0
Port 4 control register	P4CON	238	EEH	0	0	0	0	0	0	0	0
Port 4 interrupt enable/pending register	P4INTPND	239	EFH	0	0	0	0	0	0	0	0
USB function address register	FADDR	240	F0H	0	0	0	0	0	0	0	0
Control endpoint status register	EP0CSR	241	F1H	0	0	0	0	0	0	0	0
Interrupt endpoint status register	EP1CSR	242	F2H	0	0	0	0	0	0	0	0
Control endpoint byte count register	EP0BCNT	243	F3H	0	0	0	0	0	0	0	0
Control endpoint FIFO register	EP0FIFO	244	F4H	x	x	x	x	x	x	x	x
Interrupt endpoint FIFO register	EP1FIFO	245	F5H	x	x	x	x	x	x	x	x
USB interrupt pending register	USBPND	246	F6H	0	0	0	0	0	0	0	0
USB interrupt enable register	USBINT	247	F7H	0	0	0	0	0	0	1	1
USB power management register	PWRMGR	248	F8H	0	0	0	0	0	0	0	0
Locations F9H–FEH are not mapped.											
USB reset register	USBRST	255	FFH	x	x	x	x	x	x	x	1

9 I/O PORTS

OVERVIEW

The KS86C6004/C6008/P6008 has five I/O ports (0–4) with a total of 32 pins. You can access these ports directly by writing or reading port data register addresses.

For keyboard applications, ports 0, 1 and 2 are usually configured as keyboard matrix input/output. Port 3 can be configured as LED drive. Port 4 is used for host communication or for controlling a mouse or other external device.

Table 9-1. KS86C6004/C6008/P6008 Port Configuration Overview

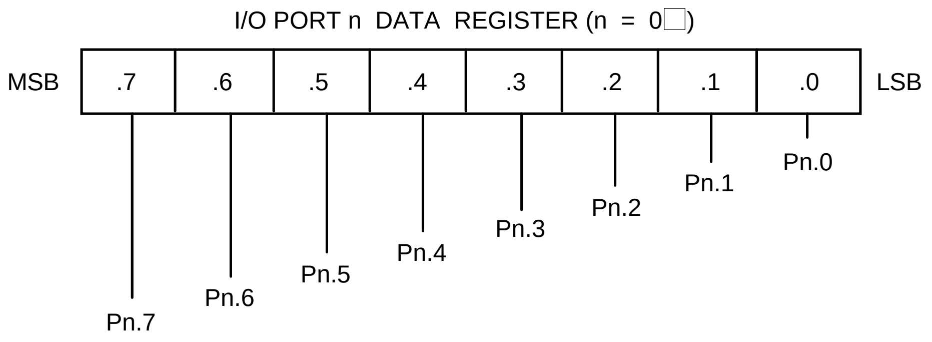
Port	Function Description	Programmability
0	Bit-programmable I/O port for Schmitt trigger input or open-drain output. Port0 can be individually configured as external interrupt inputs. Pull-up resistors are assignable by software.	Bit
1	Bit-programmable I/O port for Schmitt trigger input or open-drain output. Pull-up resistors are assignable by software.	Bit
2	Bit-programmable I/O port for Schmitt trigger input or open-drain output. Port2 can be individually configured as external interrupt inputs. Pull-up resistors are assignable by software.	Bit
3	Bit-programmable I/O port for Schmitt trigger input, open-drain or push-pull output. P3.3 can be used to system clock output (CLO) pin.	Bit
4	Bit-programmable I/O port for Schmitt trigger input or open-drain output or push-pull output. Port4 can be individually configured as external interrupt inputs. In output mode, pull-up resistors are assignable by software. But in input mode, pull-up resistors are fixed.	Bit

PORT DATA REGISTERS

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

Table 9-2. Port Data Register Summary

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



NOTE: Because only the four lower-nibble pins of port 3 and port 4 are mapped, data register bits P3.4뾎3.7 and P4.4뾎4.7 are not used.

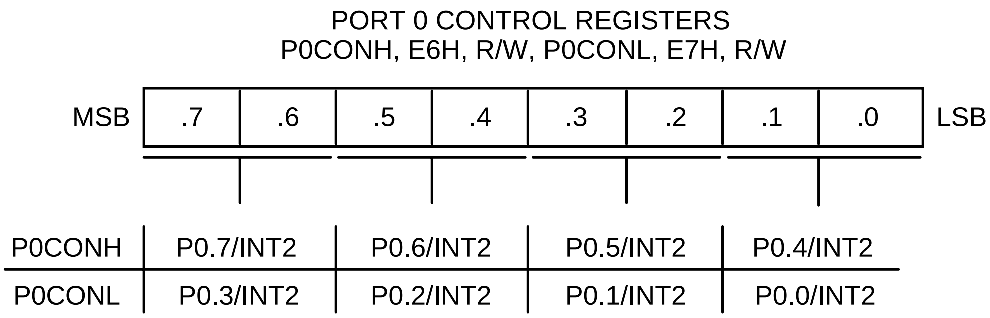
Figure 9-1. Port Data Register Format

PORT 0 AND PORT 1

Ports 0 bit-programmable, general-purpose, I/O ports. You can select Schmitt trigger input mode, N-CH open drain output mode.

You can access ports 0 and 1 directly by writing or reading the corresponding port data registers — P0 (E0H) and P1 (E1H). A reset clears the port control registers P0CONH, P0CONL, P1CONH and P1CONL to '00H', configuring all port 0 and port 1 pins as Schmitt trigger inputs.

In typical keyboard controller applications, the sixteen port 0 and port 1 pins can be used to check pressed key from keyboard matrix by generating keystrobe output signals.



7,5,3,1	6,4,2,0	Port Mode Selection
0	0	Schmitt trigger input, rising edge external interrupt mode
0	1	Schmitt trigger input, falling edge external interrupt mode with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

Figure 9-2. Port 0 Control Registers (P0CONH, P0CONL)

PORT 1 CONTROL REGISTERS
P1CONH, E8H, R/W, P1CONL, E9H, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

	.7/.6	.5/.4	.3/.2	.1/.0
P1CONH	P1.7	P1.6	P1.5	P1.4
P1CONL	P1.3	P1.2	P1.1	P1.0

7,5,3,1	6,4,2,0	Port Mode Selection
0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode with pull-up
1	0	N-CH open drain output mode
1	1	N-CH open drain output mode with pull-up

Figure 9-3. Port 1 Control Registers (P1CONH, P1CONL)

PORT 2

Port 2 is an 8-bit I/O port with individually configurable pins. It can be used for general I/O (Schmitt trigger input mode or push-pull output mode). Or, you can use port 2 pins as external interrupt (INT0) inputs. In addition, you can configure a pull-up resistor to individual pins using control register settings. All port 2 pin circuits have noise filters.

In typical keyboard controller applications, the port 2 pins are programmed to receive key input data from the keyboard matrix.

You can address port 2 bits directly by writing or reading the port 2 data register, P2 (E2H). The port 2 high-byte and low-byte control registers, P2CONH and P2CONL, are located at addresses EAH and EBH, respectively.

Two additional registers, are used for interrupt control: P2INT (ECH) and P2PND (EDH). By setting bits in the port 2 interrupt enable register P2INT, you can configure specific port 2 pins to generate interrupt requests when rising or falling signal edges are detected. The application program polls the port 2 interrupt pending register, P2PND, to detect interrupt requests. When an interrupt request is acknowledged, the corresponding pending bit must be cleared by the interrupt service routine.

In case of keyboard applications, the port 2 pins can be used to read key value from key matrix.

PORT 2 CONTROL REGISTERS
P2CONH, EAH, R/W, P2CONL, EBH, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

	.7/.6	.5/.4	.3/.2	.1/.0
P2CONH	P2.7/INT0	P2.6/INT0	P2.5/INT0	P2.4/INT0
P2CONL	P2.3/INT0	P2.2/INT0	P2.1/INT0	P2.0/INT0

7,5,3,1	6,4,2,0	Port Mode Selection
0	0	Schmitt trigger input, rising edge external interrupt
0	1	Schmitt trigger input, falling edge external interrupt with pull-u
1	0	N-CH open drain
1	1	N-CH open drain with pull-up

Figure 9-4. Port 2 Control Registers (P2CONH, P2CONL)

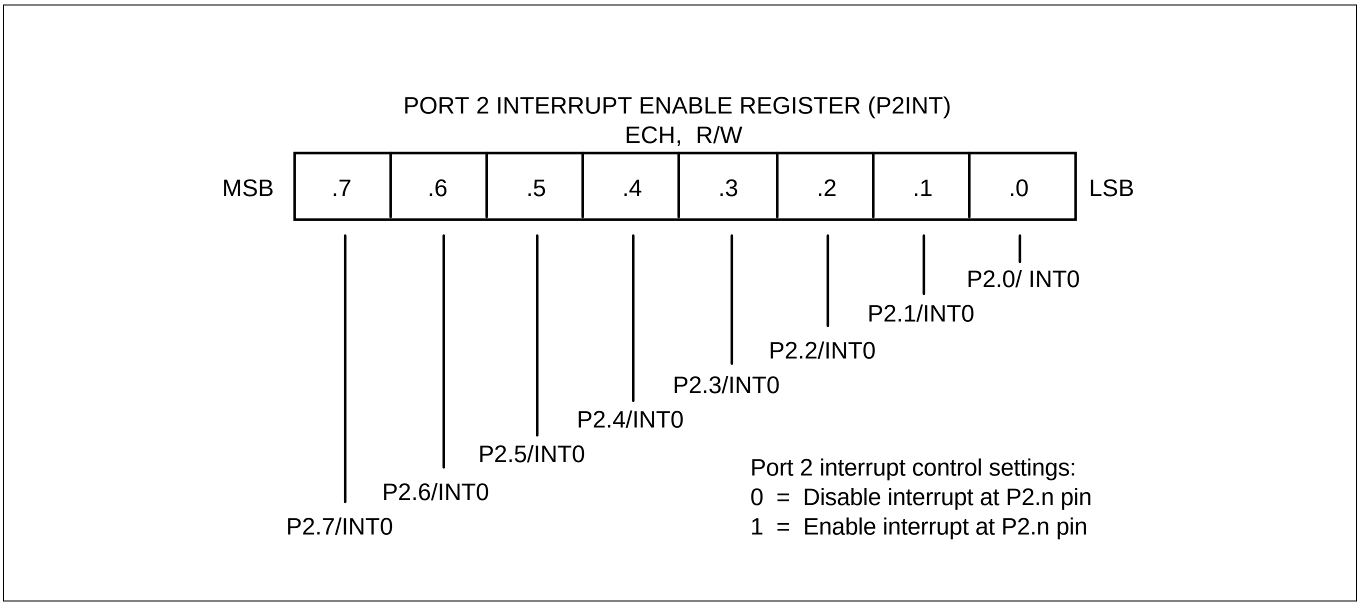


Figure 9-5. Port 2 Interrupt Enable Register (P2INT)

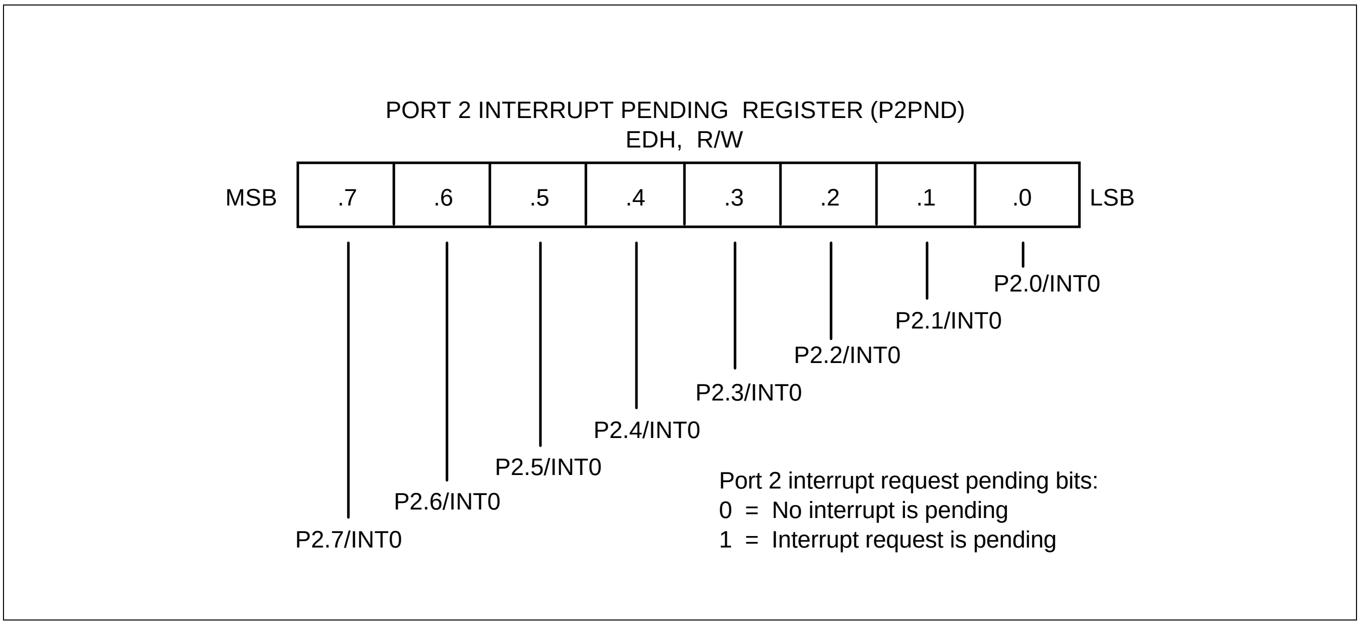
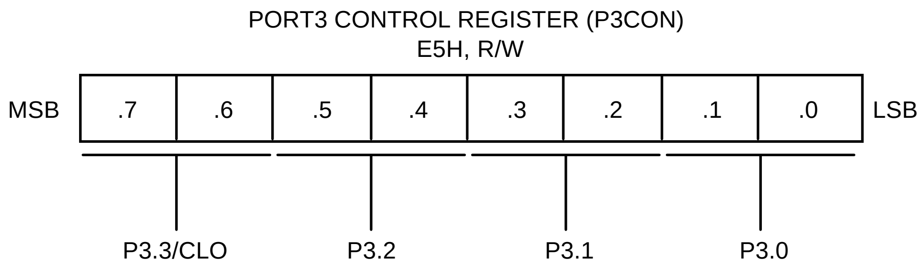


Figure 9-6. Port 2 Interrupt Pending Register (P2PND)

PORT 3

Port 3 is a 4-bit, bit-configurable, general I/O port. It is designed for high-current functions such as LED drive.

A reset configures P3.0–P3.3 to Schmitt trigger input mode. Using the P3CON register (E5H), you can alternatively configure the port 3 pins as n-channel, open-drain outputs. P3.3 can be used to system clock output (CLO) port.



7	6	Port Mode Selection (Pin 3.3)
0	0	Schmitt trigger input
0	1	System Clock output (CLO) mode. CLO comes from System clock circuit.
1	0	Push-pull output
1	1	N-CH Open drain output

5,3,1	4,2,0	Port Mode Selection (Pin 3.2-Pin 3.0)
0	x	Schmitt trigger input
1	0	Push-pull output
1	1	N-CH open drain output

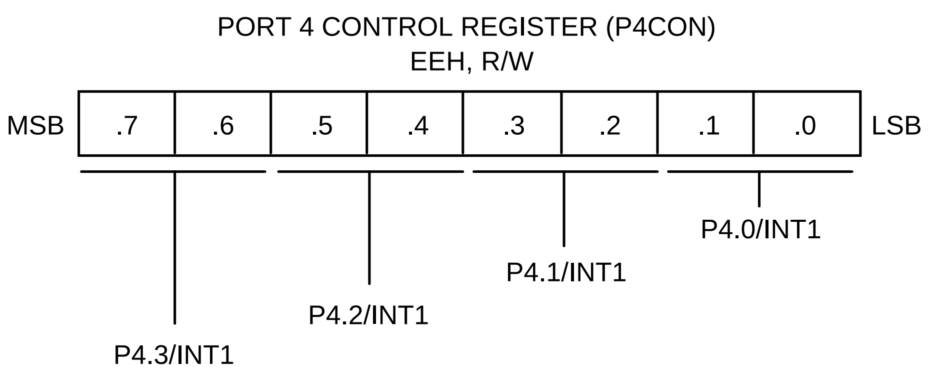
Figure 9-7. Port 3 Control Register (P3CON)

PORT 4

Port 4 is a 4-bit I/O port with individually configurable pins. It can be used for general I/O (Schmitt trigger, N-CH open drain output mode, push-pull output mode). Or, you can use port 4 pins as external interrupt (INT1) inputs. In addition, you can configure a pull-up resistor to individual pins using control register settings. All port 4 pins have noise filters.

A reset configures P4.0–P4.3 to input mode. You address port 4 directly by writing or reading the port 4 data register, P4 (E4H). The port 4 control register, P4CON, is located at EEH.

A additional registers used for interrupt control: P4INTPND (EFH). By setting bits in the port 4 interrupt enable and pending register P4INTPND.7–P4INTPND.4, you can configure specific port 4 pins to generate interrupt requests when falling signal edges are detected. The application program polls the interrupt pending register, P4INTPND.3–P4INTPND.0, to detect interrupt requests. When an interrupt request is acknowledged, the corresponding pending bit must be cleared by the interrupt service routine.

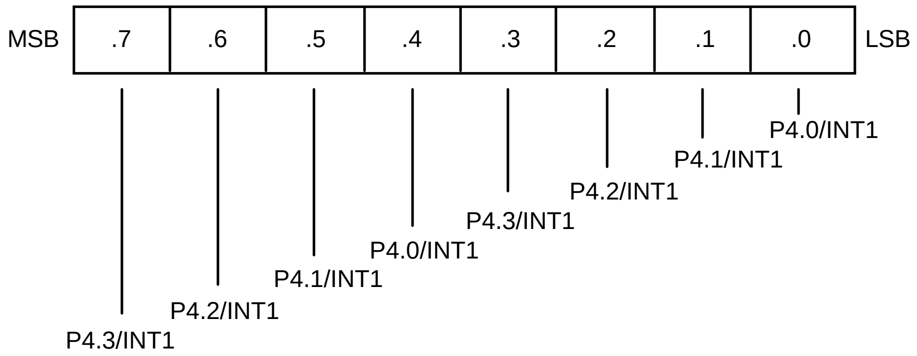


P4CON Pin Configuration Settings:

00	Schmitt trigger input, falling edge external interrupt with pull-up
01	N-CH open drain output with pull-up register
10	N-CH open drain output
11	Push-pull output

Figure 9-8. Port 4 Control Register (P4CON)

PORT 4 INTERRUPT ENABLE AND PENDING REGISTER (P4INTPND)
EFH, R/W



P4INTPND.7 - 4 : Port 4 interrupt control settings:
0 = Disable interrupt at P4.n pin
1 = Enable interrupt at P4.n pin

P4INTPND.3 - 0 : Port 4 interrupt pending bits:
0 = No interrupt request pending
1 = Interrupt request is pending

Figure 9-9. Port 4 Interrupt Enable and Pending Register (P4INTPND)

NOTES

10 BASIC TIMER and TIMER 0

MODULE OVERVIEW

The KS86C6004/C6008/P6008 has two default timers: an 8-bit *basic timer* and one 8-bit general-purpose timer/counter. The 8-bit timer/counter is called *timer 0.*

Basic Timer (BT)

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

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

The functional components of the basic timer block are:

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

Timer 0

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

- Interval timer mode
- Overflow mode

Timer 0 has the following functional components:

- Clock frequency divider (f_{OSC} divided by 4096, 256, or 8) with multiplexer
- 8-bit counter (T0CNT), 8-bit comparator, and 8-bit reference data register (T0DATA)
- Timer 0 overflow interrupt (T0OVF) and match interrupt (T0INT) generation
- Timer 0 control register, T0CON

BASIC TIMER CONTROL REGISTER (BTCON)

The basic timer control register, BTCON, is used to select the input clock frequency, to clear the basic timer counter and frequency dividers, and to enable or disable the watchdog timer function.

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

The 8-bit basic timer counter, BTCNT, can be cleared at any time during normal operation by writing a "1" to BTCON.1. To clear the frequency dividers for both the basic timer input clock and the timer 0 clock, you write a "1" to BTCON.0.

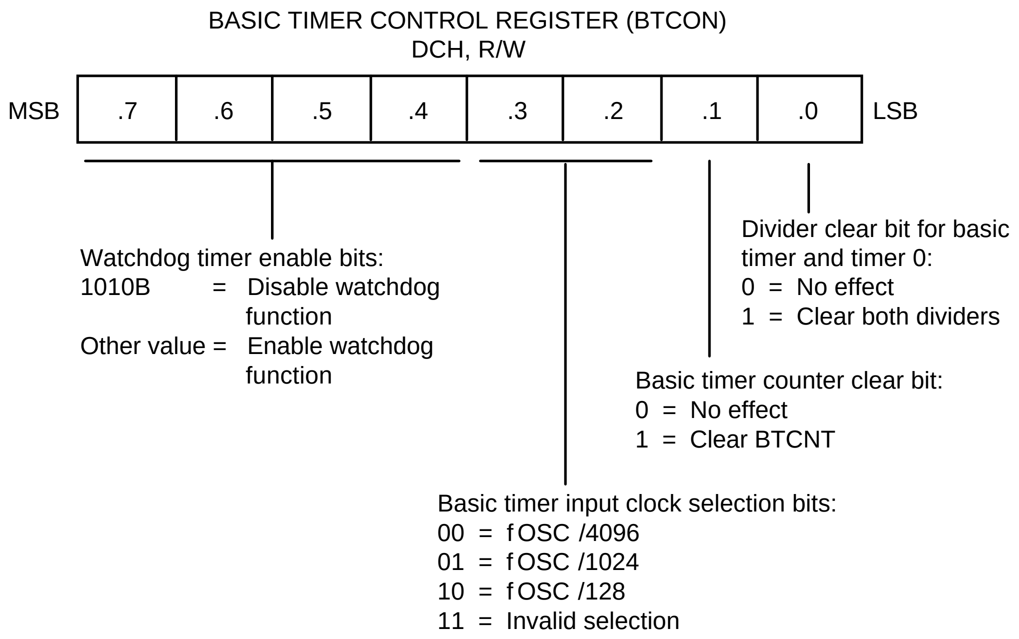


Figure 10-1. Basic Timer Control Register (BTCON)

BASIC TIMER FUNCTION DESCRIPTION

Watchdog Timer Function

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

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

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

Oscillation Stabilization Interval Timer Function

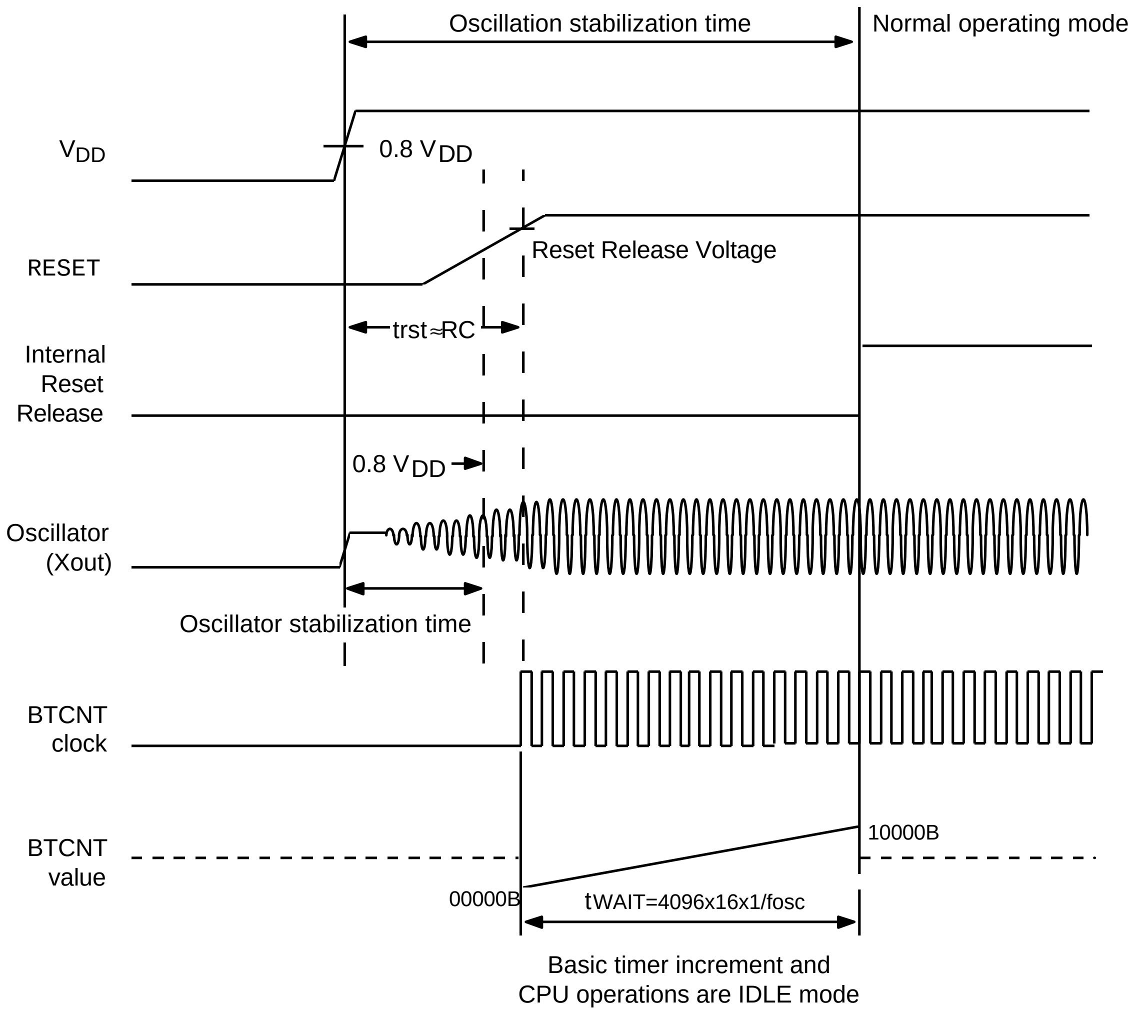
You can also use the basic timer to program a specific oscillation stabilization interval following a reset or when Stop mode has been released by an external interrupt.

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

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

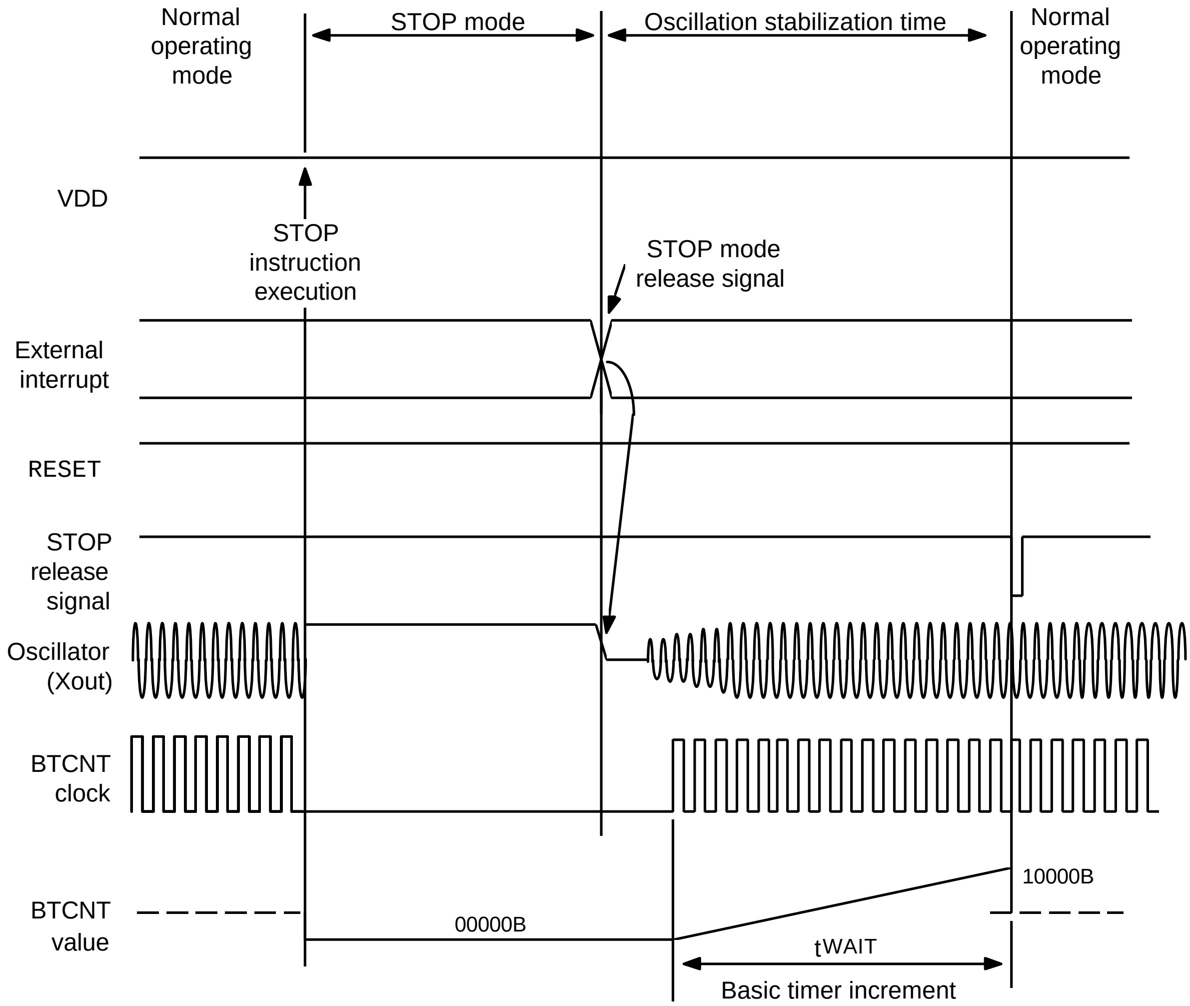
1. During Stop mode, a power-on reset or an external interrupt occurs to trigger the Stop mode release and oscillation starts.
2. If a power-on reset occurred, the basic timer counter will increase at the rate of f_{OSC} /4096. If an external interrupt is used to release Stop mode, the BTCNT value increases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

Figures 10-2 and 10-3 shows the oscillation stabilization time on RESET and STOP mode release



NOTE: Duration of the oscillator stabilization wait time, tWAIT, when it is released by a Power-on-reset is 4096x16/fosc.
trst ≈RC (R is external resister and C is on chip capacitor)

Figure 10-2. Oscillation Stabilization Time on RESET



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

BTCON.3	BTCON.2	tWAIT	tWAIT (When fosc is 6 MHz)
0	0	4096 x 16 / fosc	5.46 ms
0	1	1024 x 16 / fosc	1.365 ms
1	0	128 x 16 / fosc	0.17 ms
1	1	Invalid setting	—

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

TIMER 0 CONTROL REGISTER (T0CON)

T0CON is located at address D2H, and is read/write addressable.

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

The timer 0 overflow interrupt can be enabled by writing a "1" to T0CON.1. When a timer 0 overflow interrupt occurs and is serviced by the CPU, the pending condition must be cleared by software by writing a "0" to the timer 0 interrupt pending bit, T0CON.0.

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

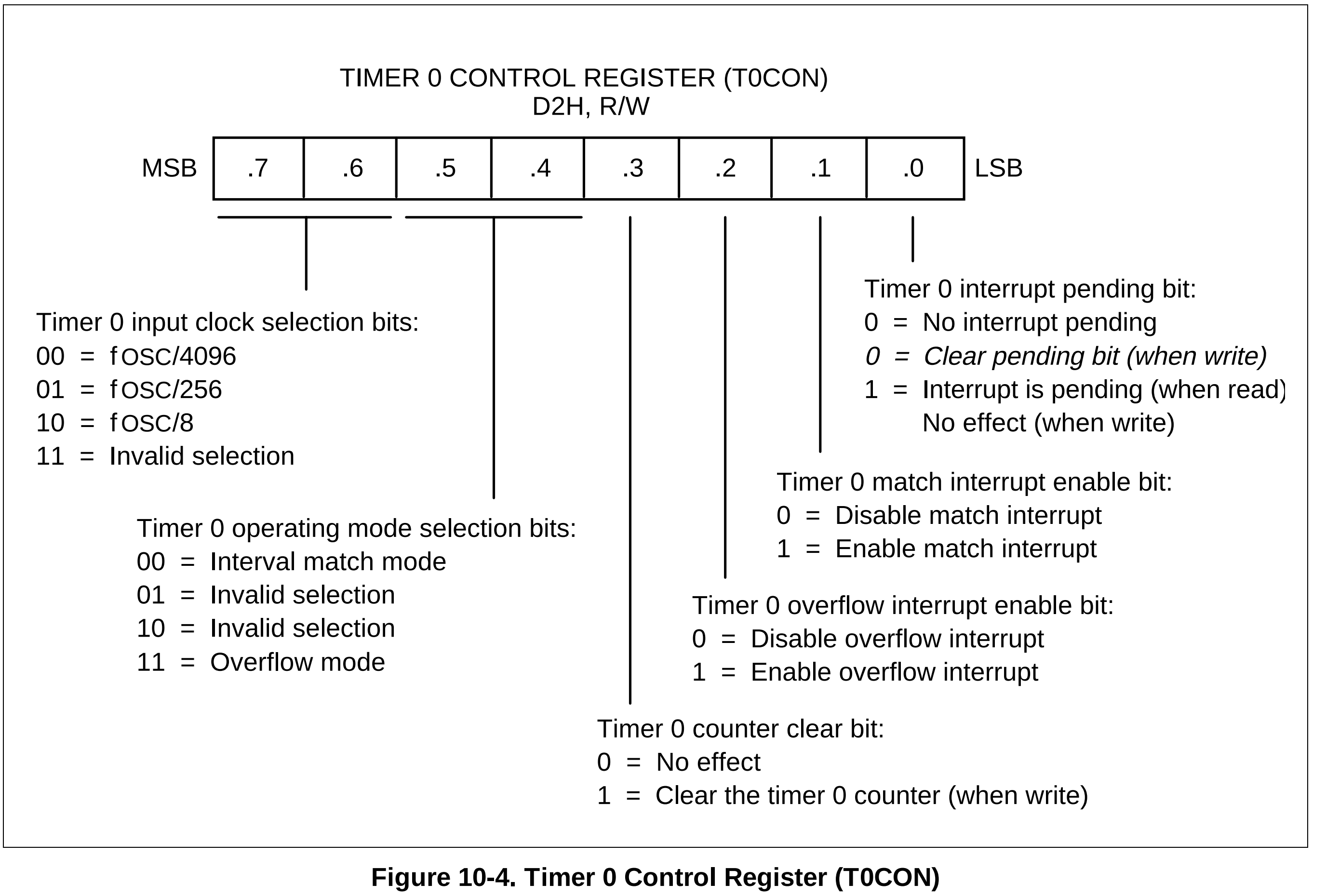


Figure 10-4. Timer 0 Control Register (T0CON)

TIMER 0 FUNCTION DESCRIPTION

Interval Match Mode

In interval match mode, a match signal is generated when the counter value is identical to the value written to the T0 reference data register, T0DATA. The match signal generates a timer 0 match interrupt and then clears the counter. If for example, you write the value '10H' to T0DATA, the counter will increment until it reaches '10H'. At this point, the T0 match interrupt is generated, the counter value is reset and counting resumes.

Overflow Mode

In overflow mode, a overflow signal is generated regardless of the value written to the T0 reference data register when the counter value is overflowed. The overflow signal generates a timer 0 overflow interrupt and then T0 counter is cleared.

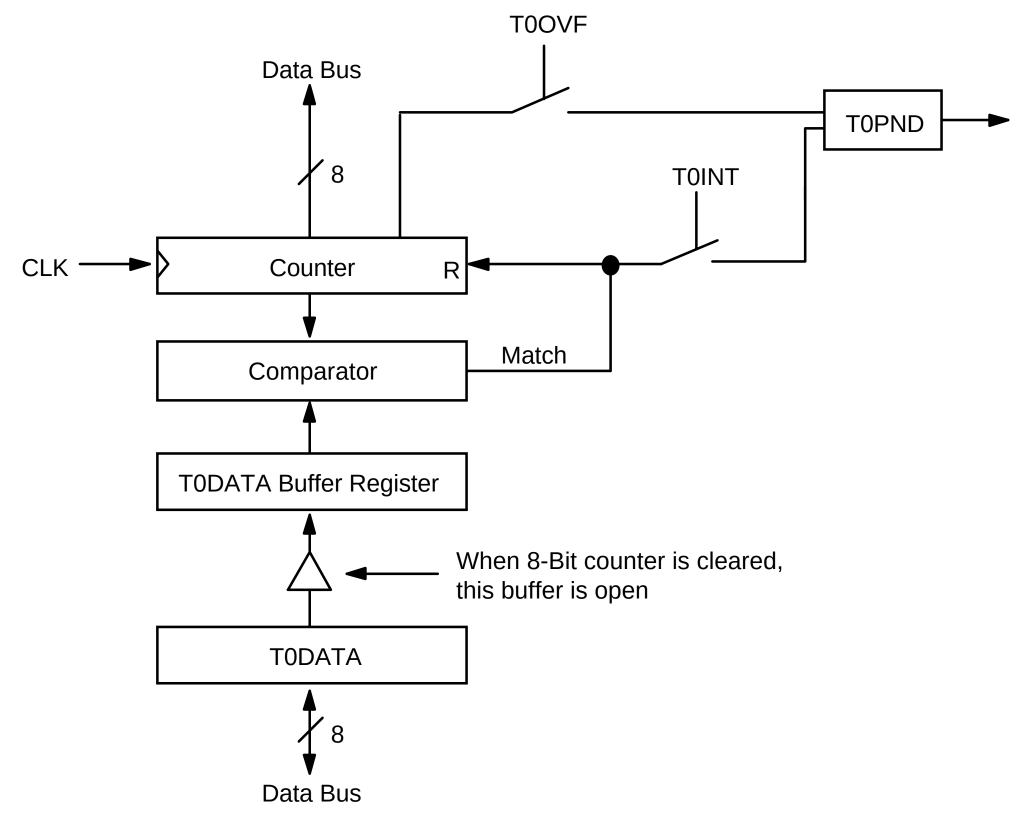


Figure 10-5. Simplified Timer 0 Function Diagram: Interval Timer Mode

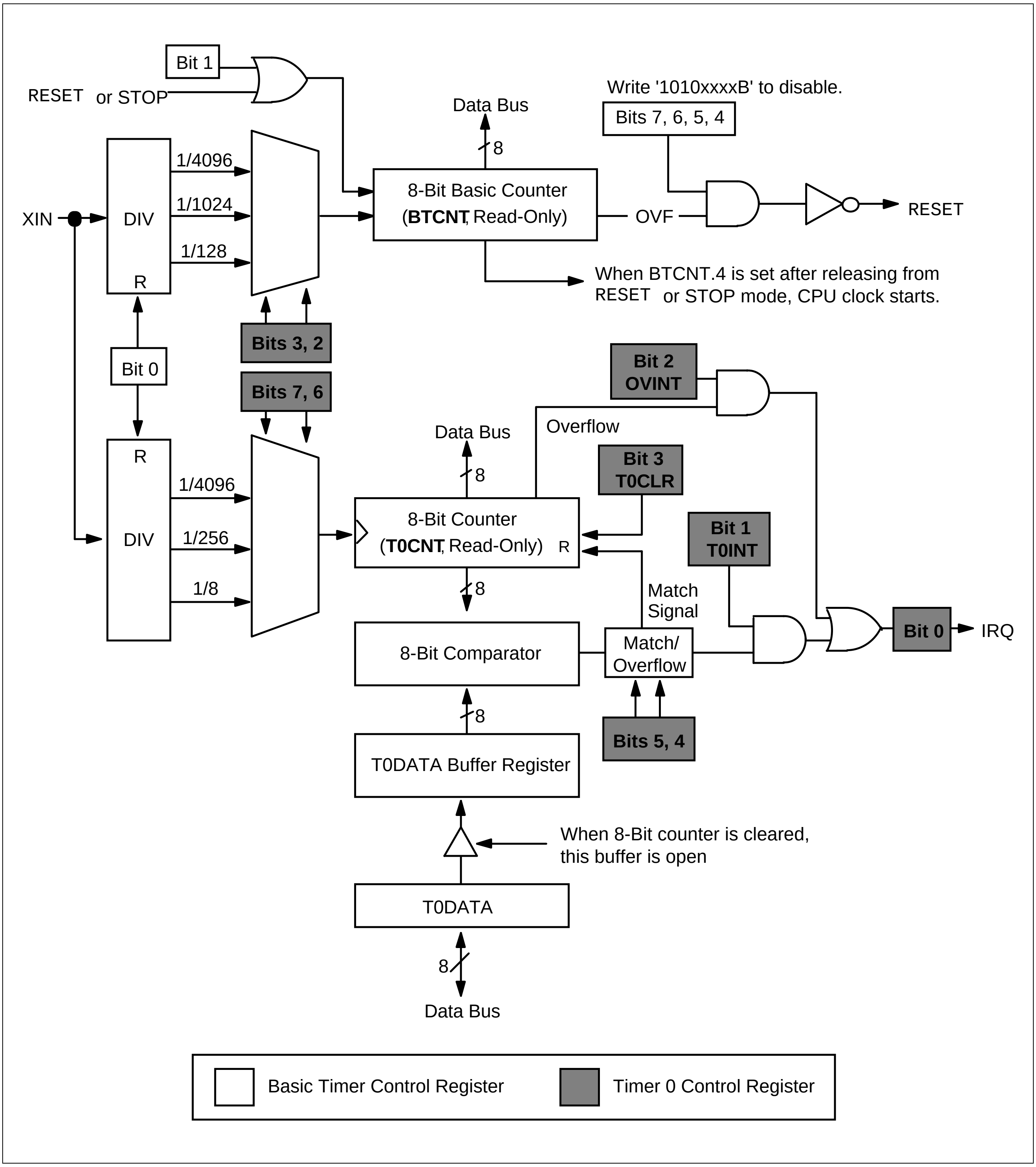


Figure 10-6. Basic Timer and Timer 0 Block Diagram

NOTES

11 UNIVERSAL SERIAL BUS

OVERVIEW

Universal Serial Bus (USB) is a communication architecture that supports data transfer between a host computer and a wide range of PC peripherals. USB is actually a cable bus in which the peripherals share its bandwidth through a host scheduled token based protocol.

The USB module in KS86C6004/C6008/P6008 is designed to serve at a low speed transfer rate (1.5 Mbs) USB device as described in the Universal Serial Bus Specification Revision 1.0. KS86C6004/C6008/P6008 can be briefly describe as a microcontroller with SAM 87RI core with an on-chip USB peripheral as can be seen in figure 11-1.

The KS86C6004/C6008/P6008 comes equipped with Serial Interface Engine (SIE), which handles the communication protocol of the USB. The KS86C6004/C6008/P6008 supports the following control logic: packet decoding/generation, CRC generation/checking, NRZI encoding/decoding, Sync detection, EOP (end of packet) detection and bit stuffing.

KS86C6004/C6008/P6008 supports two types of data transfers; control and interrupt. Two endpoints are used in this device; Endpoint 0 and Endpoint 1. Please refer to the USB specification revision 1.0 for detail description of USB.

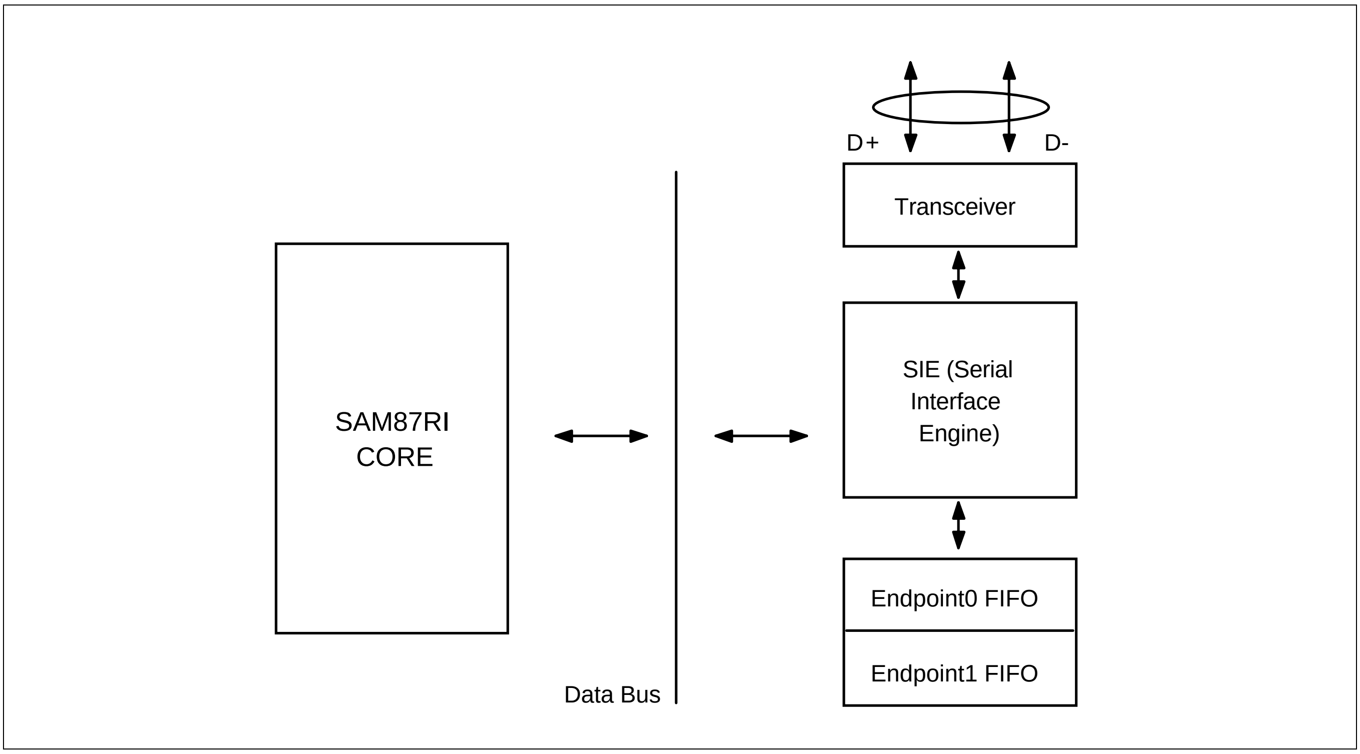


Figure 11-1. USB Peripheral Interface

Serial Bus Interface Engine (SIE)

The Serial Interface Engine interfaces to the USB serial data and handles, deserialization/serialization of data, NRZI encoding/decoding, clock extraction, CRC generation and checking, bit stuffing and other specifications pertaining to the USB protocol such as handling inter packet time out and PID decoding. .

Control Logic

The USB control logic manages data movements between the CPU and the transceiver by manipulating the transceiver and the endpoint register. This includes both transmit and receive operations on the USB. The logic contains byte count buffers for transmit operations that load the active transmit endpoint's byte count and use this to determine the number of bytes to transfer. The same buffer is used for receive transactions to count the number of bytes received and transfer that number to the receive endpoint's byte count register at the end of the transaction.

The control logic in KS86C6004/C6008/P6008, when transmitting, manages parallel to serial conversion, packet generation, CRC generation, NRZI encoding and bit stuffing.

When receiving, the control logic in KS86C6004/C6008/P6008 handles Sync detection, packet decoding, EOP (end of packet) detection, bit stuffing, NRZI decoding, CRC checking and serial to parallel conversion

Bus Protocol

All bus transactions involve the transmission of packets. KS86C6004/C6008/P6008 supports three packet types; Token, Data and Handshake. Each transaction starts when the host controller sends a Token Packet to the USB device. The Token packets are generated by the USB host and decoded by the USB device. A Token Packet includes the type description, direction of the transaction, USB device address and the endpoint number.

Data and Handshake packets are both decoded and generated by the USB device, pending on the type of transaction. In any transaction, the data is transferred from the host to a device or from a device to the host. The transaction source then sends a Data Packet or indicates that it has no data to transfer. The destination then responds with a Handshake Packet indicating whether the transfer was successful.

Data Transfer Types

USB data transfer occurs between the host software and a specific endpoint on the USB device. An endpoint supports a specific type of data transfer. The KS86C6004/C6008/P6008 supports two data transfer endpoints: control and interrupt.

Control transfer configures and assigns an address to the device when detected. Control transfer also supports status transaction, returning status information from device to host.

Interrupt transfer refers to a small, spontaneous data transfer from USB device to host.

Endpoints

Communication flows between the host software and the endpoints on the USB device. Each endpoint on a device has an identifier number. In addition to the endpoint number, each endpoint supports a specific transfer type. KS86C6004/C6008/P6008 supports two endpoints: Endpoint 0 supports control transfer, and Endpoint 1 supports interrupt transfer.

USB FUNCTION ADDRESS REGISTER (FADDR)

This register holds the USB address assigned by the host computer. FADDR is located at address F0H and is read/write addressable.

Bit7 Not used

Bit6–0 **FADDR:** MCU updates this register once it decodes a SET_ADDRESS command. MCU must write this register before it clears OUT_PKT_RDY (bit0) and sets DATA_END (bit3) in the EP0CSR register. The function controller use this register's value to decode USB Token packet address. At reset, if the device is not yet configured the value is reset to 0.

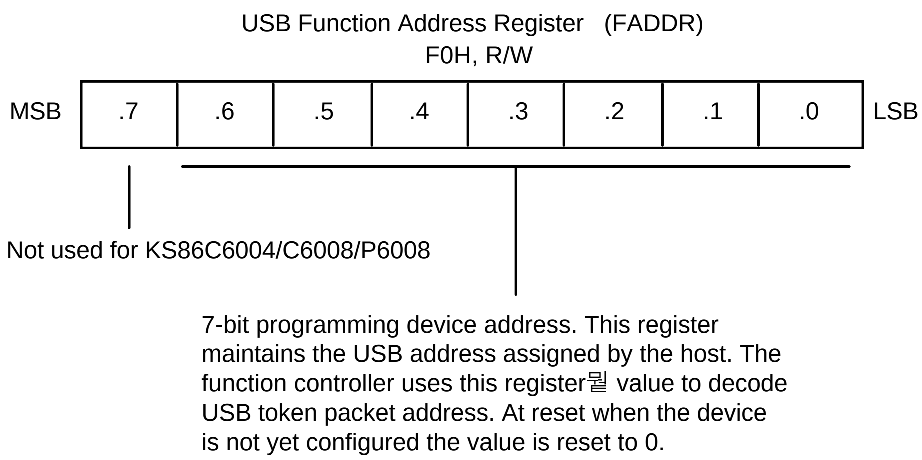


Figure 11-2. USB Function Address Register (FADDR)

CONTROL ENDPOINT STATUS REGISTER (EP0CSR)

EP0CSR register controls Endpoint 0 (Control Endpoint), and also holds status bits for Endpoint 0. EP0CSR is located at F1H and is read/write addressable.

Bit7 **CLEAR_SETUP_END:** MCU writes “1” to this bit to clear SETUP_END bit (bit4). This bit is automatically cleared after writing "1" by USB block.

Bit6 **CLEAR_OUT_PKT_RDY:** MCU writes “1” to this bit to clear OUT_PKT_RDY bit (bit0). This bit is automatically cleared after writing "1" by USB block.

Bit5 **SEND_STALL:** MCU writes “1” to this bit to send STALL signal to the Host, at the same time it clears OUT_PKT_RDY (bit0), if it decodes an invalid token. USB issues a STALL Handshake to the current control transfer. This bit gets cleared once a STALL Handshake is issued to the current control transfer.

Bit4 **SETUP_END:** MCU sets this bit, when a control transfer ends before DATA_END bit (bit3) is set. MCU clears this bit, by writing a “1” to CLEAR_SETUP_END bit (bit7). When USB sets this bit, an interrupt is generated to MCU. When such condition occurs, USB flushes the FIFO, and invalidates MCU’s access to FIFO.

Bit3 **DATA_END:** MCU sets this bit:

— After loading the last packet of data into the FIFO, and at the same time IN_PKT_RDY bit is set.
— While it clears OUT_PKT_RDY bit after unloading the last packet of data.
— For a zero length data phase, when it clears OUT_PKT_RDY bit, and sets IN_PKT_RDY bit.

Bit2 **SENT_STALL:** USB sets this bit, if a control transaction has ended due to a protocol violation. An interrupt is generated when this bit gets set. MCU clears this bit to end the STALL condition.

Bit1 **IN_PKT_RDY:** MCU sets this bit, after writing a packet of data into Endpoint 0 FIFO. USB clears this bit, once the packet has been successfully sent to the host. An interrupt is generated when USB clears this bit so that MCU can load the next packet. For a zero length data phase, MCU sets IN_PKT_RDY bit and DATA_END bit at the same time.

Bit0 **OUT_PKT_RDY:** USB sets this bit, once a valid token is written to FIFO. An interrupt is generated, when USB sets this bit. MCU clears this bit by writing "1" to CLEAR_OUT_PKT_RDY bit.

In control transfer case, where there is no data phase, MCU after unloading the setup token, sets IN_PKT_RDY, and DATA_END at the same time it clears OUT_PKT_RDY for the setup token.

When SETUP_END bit is set, OUT_PKT_RDY bit may also be set. This happens when the current transfer has ended, and a new control transfer is received before MCU can service the interrupt. In such case, MCU should first clear SETUP_END bit, and then start servicing the new control transfer.

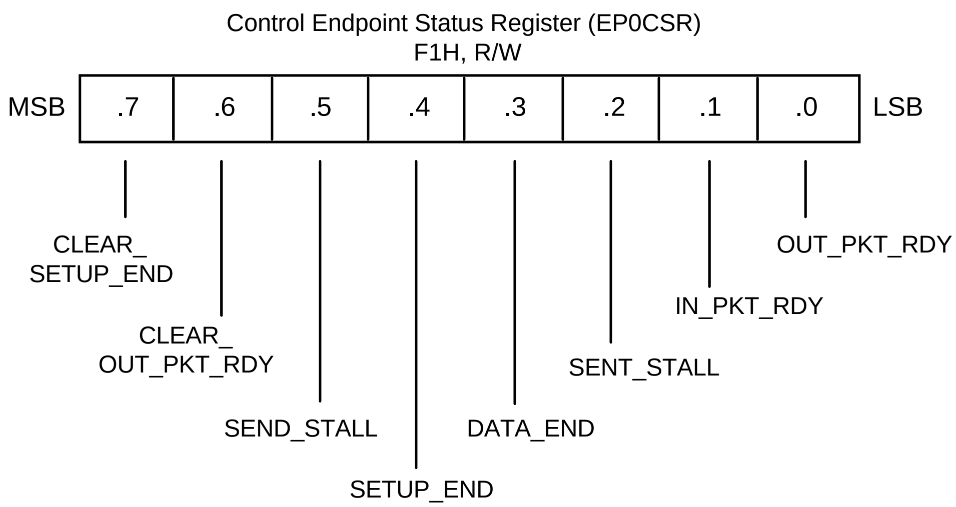


Figure 11-3. Control Endpoint Status Register (EP0CSR)

INTERRUPT ENDPOINT STATUS REGISTER (EP1CSR)

EP1CSR is the control register for Endpoint 1, Interrupt Endpoint. This register is located at address F2H and is read/write addressable.

Bit7 **CLEAR_DATA_TOGGLE:** MCU writes “1” to this bit to clear the data toggle sequence bit. When the MCU writes a 1 to this register, the data toggle bit is initialized to DATA0.

Bit6–3 **MAXP:** These bits indicate the maximum packet size for IN endpoint, and needs to be updated by MCU before it sets IN_PKT_RDY. Once set, the contents are valid till MCU re-writes them.

Bit2 **FLUSH_FIFO:** When MCU writes “1” to this register, the FIFO is flushed, and IN_PKT_RDY cleared. The MCU should wait for IN_PKT_RDY to be cleared for the flush to take place.

Bit1 **FORCE_STALL:** MCU writes “1” to this register to issue a STALL Handshake to USB. MCU clears this bit, to end the STALL condition.

Bit0 **IN_PKT_RDY:** MCU sets this bit, after writing a packet of data into Endpoint 1 FIFO. USB clears this bit, once the packet has been successfully sent to the Host. An interrupt is generated when USB clears this bit, so MCU can load the next packet.

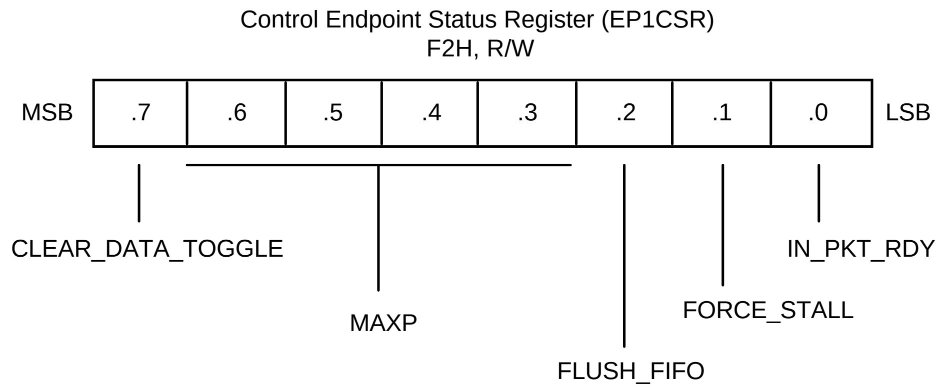


Figure 11-4. Interrupt Endpoint Status Register (EP1CSR)

CONTROL ENDPOINT BYTE COUNT REGISTER (EP0BCNT)

EP0BCNT register has the number of valid bytes in Endpoint 0 FIFO. It is located at address F3H and is read/write addressable. Once the MCU receives a OUT_PKT_RDY (Bit0 of EP0CSR) for Endpoint 0, then it can read this register to find out the number of bytes to be read from Endpoint 0 FIFO.

CONTROL ENDPOINT FIFO REGISTER (EP0FIFO)

This register is bi-directional, 8-byte depth FIFO used to transfer Control Endpoint data. EP0FIFO is located at address F4H and is read/write addressable.

Initially, the direction of the FIFO, is from the Host to the MCU. After a setup token is received for a control transfer, that is, after MCU unload the setup data packet, and clears OUT_PKT_RDY, the direction of FIFO is changed automatically by the direction bit of data packet.

INTERRUPT ENDPOINT FIFO REGISTER (EP1FIFO)

EP1FIFO is an uni-direction 8-byte depth FIFO used to transfer data from the MCU to the Host. MCU writes data to this register, and when finished set IN_PKT_RDY. This register is located at address F5H and is able to write.

USB INTERRUPT PENDING REGISTER (USBPND)

USBPND register has the interrupt bits for endpoints and power management. *This register is cleared once read by MCU.* While any one of the bits is set, an interrupt is generated. USBPND is located at address F6H and is read/write addressable.

Bit7–4 Not used

Bit3 **RESUME_PND:** While in suspend mode, if resume signaling is received this bit gets set.

Bit2 **SUSPEND_PND:** This bit is set, when suspend signaling is received.

Bit1 **ENDPT1_PND:** This bit is set, when Endpoint 1 needs to be serviced.

Bit0 **ENDPT0_PND:** This bit is set, when Endpoint 0 needs to be serviced. It is set under any one of the following conditions:

- — OUT_PKT_RDY is set.
- — IN_PKT_RDY gets cleared.
- — SENT_STALL gets set.
- — DATA_END gets cleared.
- — SETUP_END gets set.

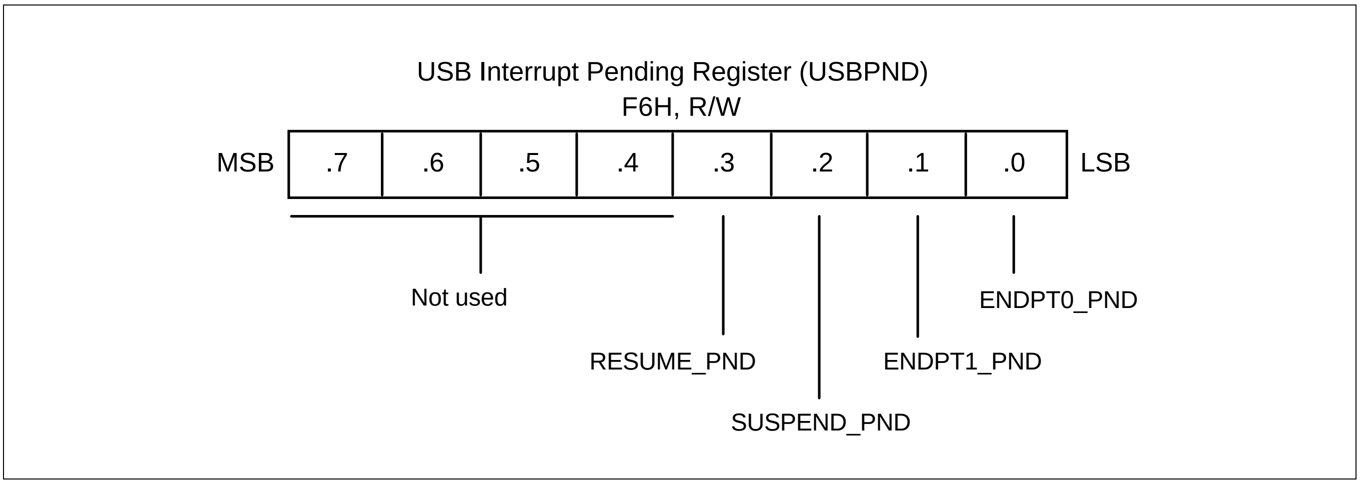


Figure 11-5. USB Interrupt Pending Register (USBPND)

USB INTERRUPT ENABLE REGISTER (USBINT)

USBINT is located at address F7H and is read/write addressable. This register serves as an interrupt mask register. If the corresponding bit = 1 then the respective interrupt is enabled.

By default, all interrupts except suspend interrupt is enabled. Interrupt enables bits for suspend and resume is combined into a single bit (bit 2).

Bit7–3 Not used

Bit2 **ENABLE_SUSPEND_RESUME_INT:**
1 Enable SUSPEND and RESUME INTERRUPT
0 Disable SUSPEND and RESUME INTERRUPT (default)

Bit1 **ENABLE_ENDPT1_INT:**
1 Enable ENDPOINT 1 INTERRUPT (default)
0 Disable ENDPOINT 1 INTERRUPT

Bit0 **ENABLE_ENDPT0_INT:**
1 Enable ENDPOINT 0 INTERRUPT (default)
0 Disable ENDPOINT 0 INTERRUPT

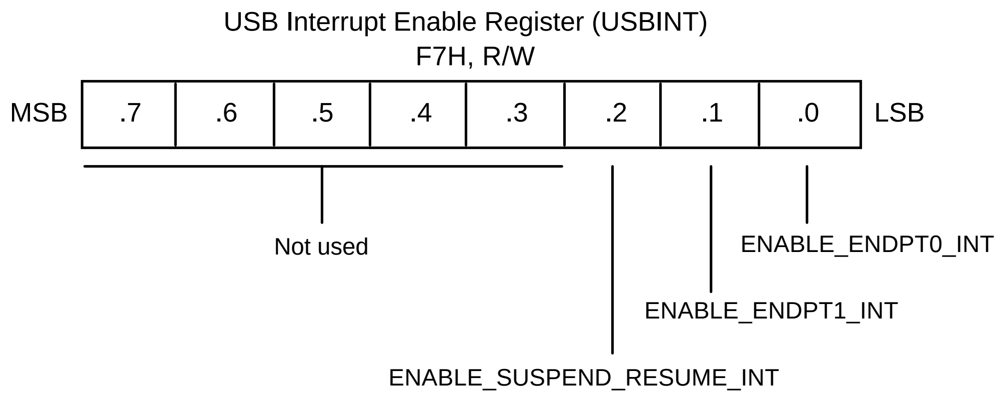


Figure 11-6. USB Interrupt Enable Register (USBINT)

USB POWER MANAGEMENT REGISTER (PWRMGR)

PWRMGR register interacts with the Host's power management system to execute system power events such as SUSPEND or RESUME. This register is located at address F8H and is read/write addressable.

Bit7–2 **RESERVED:** The value read from this bit is zero.

Bit1 **SEND_RESUME:** While in SUSPEND state, if the MCU wants to initiate RESUME, it writes "1" to this register for 10ms (maximum of 15 ms), and clears this register. In SUSPEND mode if this bit reads "1", USB generates RESUME signaling.

Bit0 **SUSPEND_STATE:** Suspend state is set when the MCU sets suspend interrupt. This bit is cleared automatically when:

— MCU writes "0" to SEND_RESUME bit to end the RESUME signaling (after SEND_RESUME is set for 10 ms).
— MCU receives RESUMES signaling from the Host while in SUSPEND mode.

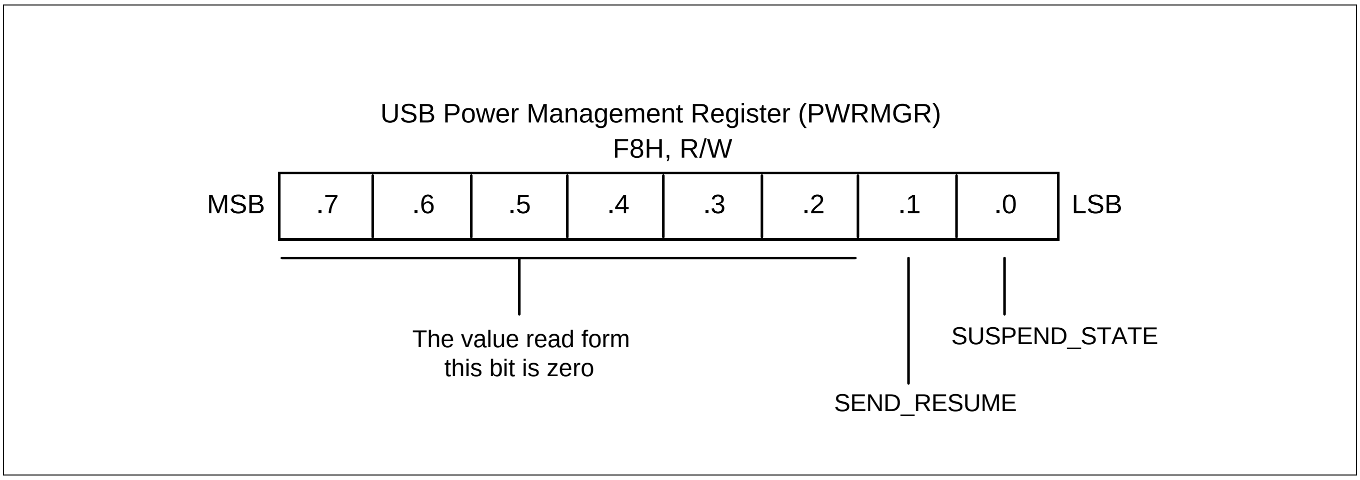


Figure 11-7. USB Power Management Register (PWRMGR)

USB RESET REGISTER (USBRST)

USBRST register receives a reset signal from the Host when there has been no activities on UBS for a certain period of time. This register is located at address FFH and is read/write addressable.

Bit7–1 Not used

Bit0 **USBRST:** This bit is set when the Host issues an USB reset signal.

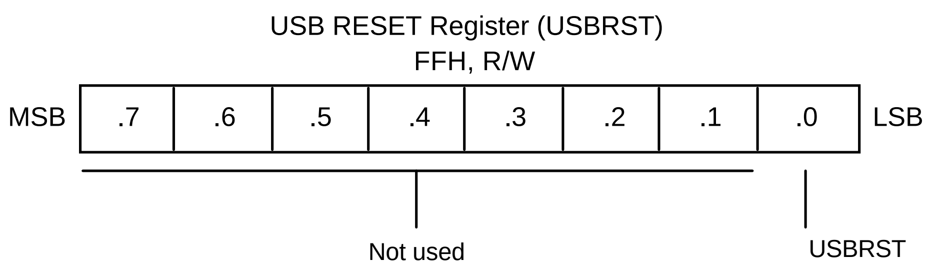


Figure 11-8. USB RESET Register (USBRST)

12 ELECTRICAL DATA

OVERVIEW

In this section, the following KS86C6004/C6008/P6008 electrical characteristics are presented in tables and graphs:

- Absolute maximum ratings
- D.C. electrical characteristics
- Input/Output capacitance
- A.C. electrical characteristics
- Input timing for external interrupt (Ports 0, 2, and 4)
- Input timing for RESET
- Oscillator characteristics
- Oscillation stabilization time
- Clock timing measurement points at X_{IN}
- Data retention supply voltage in Stop mode
- Stop mode release timing when initiated by a reset
- Stop mode release timing when initiated by an external interrupt
- Characteristic curves

Table 12-1. Absolute Maximum Ratings

(T_A = 25°C)

Parameter	Symbol	Conditions	Rating	Unit
Supply Voltage	V_{DD}	–	– 0.3 to + 6.5	V
Input Voltage	V_{IN}	All input ports	– 0.3 to V_{DD} + 0.3	V
Output Voltage	V_O	All output ports	– 0.3 to V_{DD} + 0.3	V
Output Current High	I_{OH}	One I/O pin active	– 18	mA
		All I/O pins active	– 60	
Output Current Low	I_{OL}	One I/O pin active	+ 30	mA
		Total pin current for ports 3	+ 100	
		Total pin current for ports 0, 1, 2, 4	+ 100	
Operating Temperature	T_A	–	– 40 to +85	°C
Storage Temperature	T_{STG}	–	– 65 to + 150	°C

Table 12-2. D.C. Electrical Characteristics

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

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Operating Voltage	V_{DD}	f_{OSC} = 6 MHz (instruction clock = 1 MHz)	4.5	5.0	5.5	V
Input High Voltage	V_{IH1}	All input pins except V_{IH2}	0.8 V_{DD}	–	V_{DD}	V
	V_{IH2}	X_{IN}	V_{DD} – 0.5		V_{DD}	
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}	I_{OH} = – 200 µA; All output ports except ports 0, 1 and 2	V_{DD} – 1.0	–	–	V
Output Low Voltage	V_{OL}	I_{OL} = 1 mA All output port	–	–	0.4	V
Output High Leakage Current	I_{LOH} (1)	V_{OUT} = V_{DD} All I/O pins and output pins	–	–	3	µA
Output Low Leakage Current	I_{LOL} (1)	V_{OUT} = 0 V All I/O pins and output pins	–	–	– 3	µA
Pull-up Resistors	R_{L1}	V_{IN} = 0 V Ports 0, 1, 2, 4	25	50	100	KΩ
	R_{L2}	V_{IN} = 0 V; $\overline{RESET}$ only	100	220	300	
Supply Current (2)	I_{DD1}	Normal operation mode 6 MHz CPU clock	–	5.5	12	mA
	I_{DD2}	Idle mode; 6 MHz oscillator		2.2	5	mA
	I_{DD3}	Stop mode		180	300	µA
Oscillator Feed Back Resistor	R_{OSC}	V_{DD} = 5.5 V; T_A = 25°C X_{IN} = V_{DD}; X_{OUT} = 0V	500	900	1200	KΩ

NOTES:
1. Except X_{IN} and X_{OUT}.
2. Supply current does not include current drawn through internal pull-up resistors or external output current loads.

Table 12-3. 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 12-4. A.C. Electrical Characteristics

(T_A = – 40°C to + 85°C, V_{DD} = 4.5 V to 6.0 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Interrupt Input High, Low Width	t_{INTH}, t_{INTL}	P0, P2 and P4	–	200	–	ns
RESET Input Low Width	t_{RSL}	RESET	–	1,000	–	

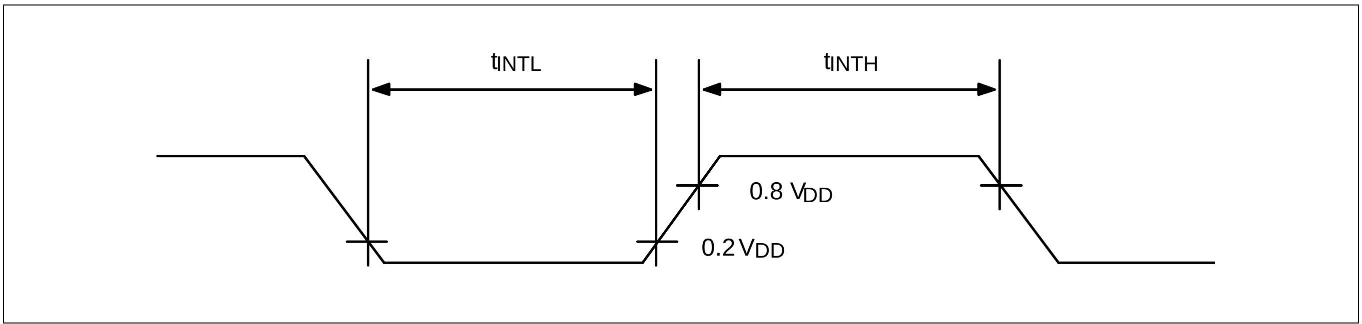


Figure 12-1. Input timing for external interrupt (Ports 0, 2, and 4)

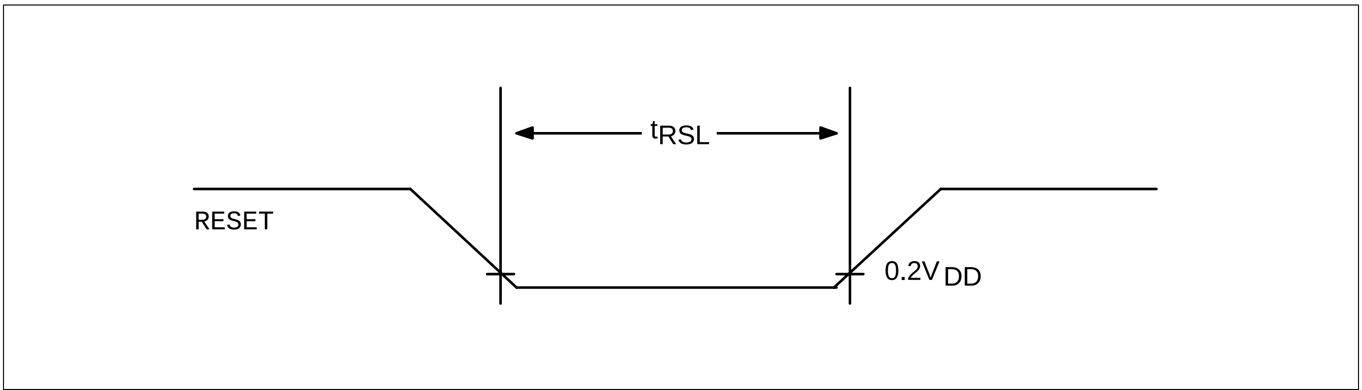


Figure 12-2. Input Timing for RESET

Table 12-5. Oscillator Characteristics

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

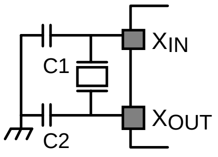
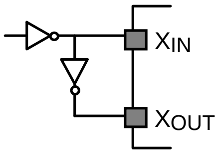
Oscillator	Clock Circuit	Test Condition	Min	Typ	Max	Unit
Main crystal Main ceramic (f_{OSC})		Oscillation frequency	–	6.0	–	MHz
External clock		Oscillation frequency	–	6.0	–	

Table 12-6. Oscillation Stabilization Time

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

Oscillator	Test Condition	Min	Typ	Max	Unit
Main Crystal	f_{OSC} = 6.0 MHz (Oscillation stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.)	–	–	10	ms
Main Ceramic					
Oscillator Stabilization Wait Time	t_{WAIT} stop mode release time by a reset	–	2^{16} / f_{OSC}	–	
	t_{WAIT} stop mode release time by an interrupt	–	(note)	–	

NOTE: The oscillator stabilization wait time, t_{WAIT}, is determined by the setting in the basic timer control register, BTCON.

Table 12-7. Data Retention Supply Voltage in Stop Mode

(T_A = – 40°C to + 85°C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data Retention Supply Voltage	V_{DDDR}	Stop mode	2.0	–	6	V
Data Retention Supply Current	I_{DDDR}	Stop mode; V_{DDDR} = 2.0 V	–	–	300	μA

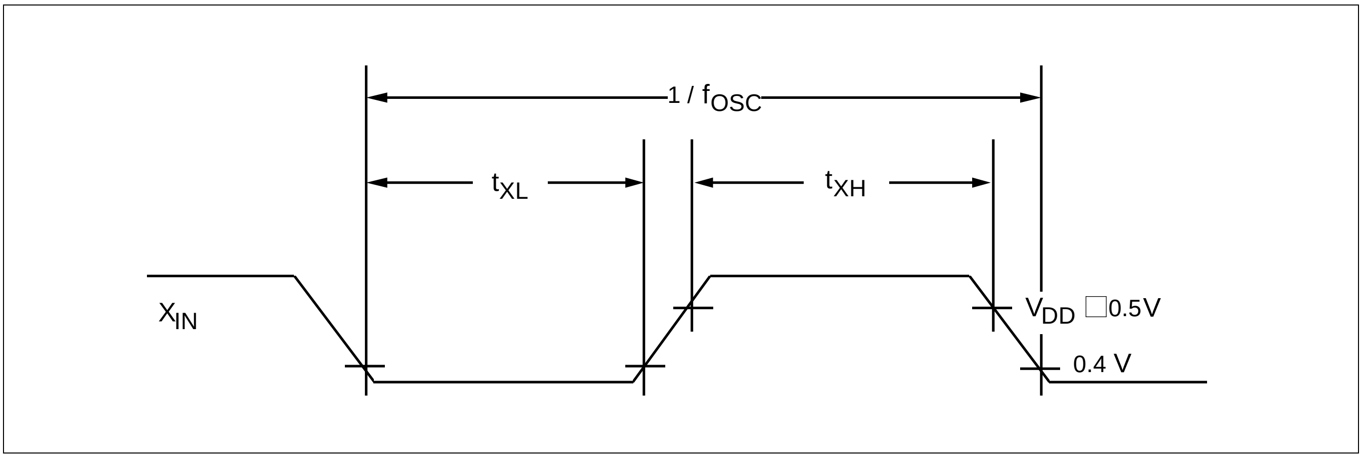


Figure 12-3. Clock Timing Measurement Points at X_{IN}

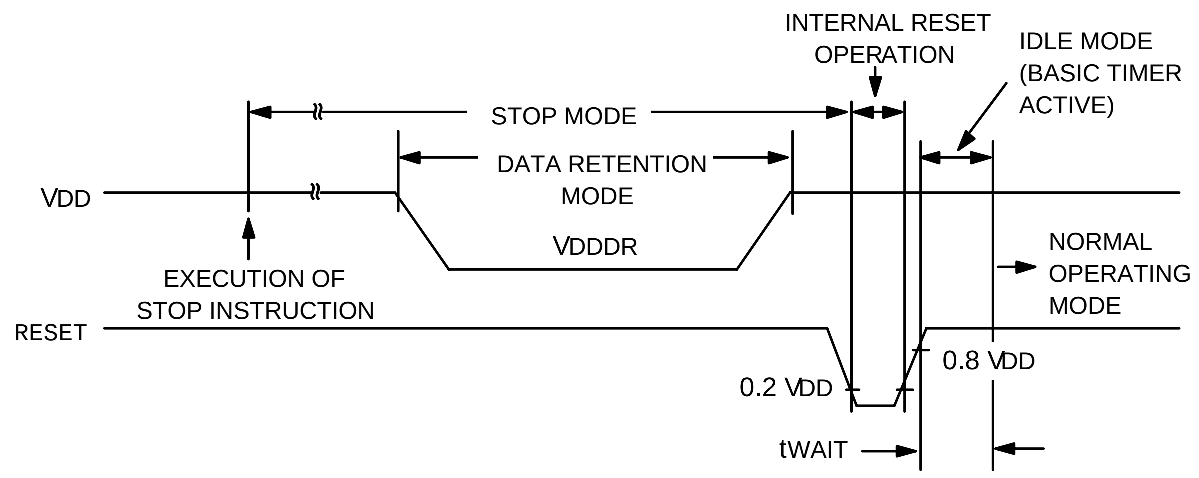


Figure 12-4. Stop Mode Release Timing When Initiated by a Reset

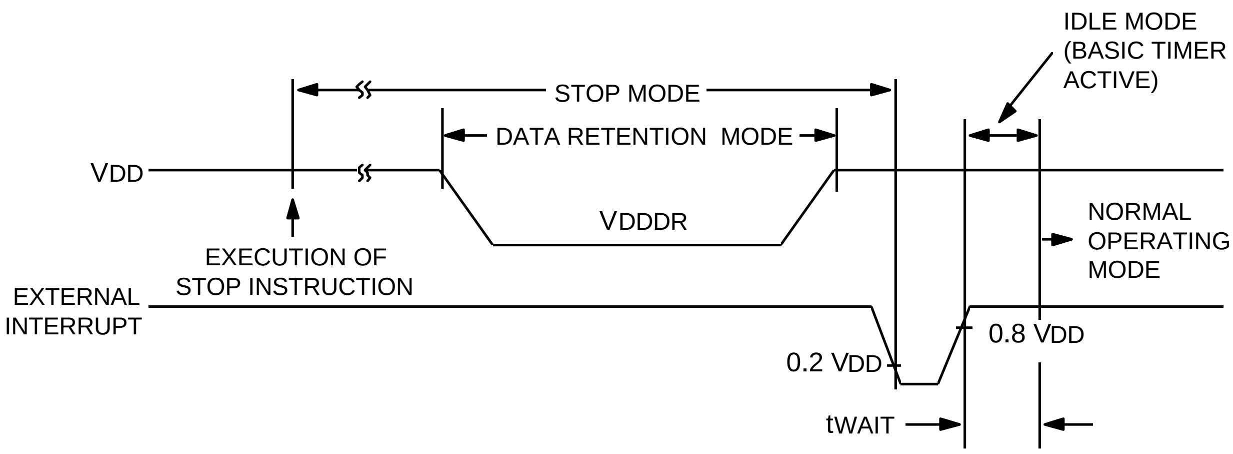


Figure 12-5. Stop Mode Release Timing When Initiated by an External Interrupt

Table 12-8. Low Speed Source Electrical Characteristics

Parameter	Symbol	Conditions (1, 2, 3)	Min	Max	Unit
Driver Characteristics:					
Transition Time:		(4, 5), Figure 7-16			
Rise Time	TR	CL = 50 pF	75	–	ns
		CL = 350 pF	–	300	
Fall Time	TF	CL = 50 pF	75	–	
		CL = 350 pF	–	300	
Rise/Fall Time Matching	TRFM	(FR/TF)	80	120	%
Output Signal Crossover Voltage	VCRS		1.3	2.0	V
Data Source Timings:					
Low Speed Data Rate	TRDATE	Ave. Bit rate (1.5 Mb/s ± 1.5 %)	1.4775	1.5225	Mbs
Source Differential Driver Jitter		(6, 7), Figure 12-6			
Host (Downstream):					
To Next Transition	TDDJ1		– 75	75	ns
For Paired Transitions	TDDJ2		– 45	45	
Function (Upstream):					
To Next Transition	TUDJ1		– 95	95	
For Paired Transitions	TUDJ2		– 150	150	
Source EOP Width	TEOPT	(7), Figure 12-7	1.25	1.5	µs
Differential to EOP Transition Skew	TDEOP	(7), Figure 12-7	– 40	100	ns
Receiver Data Jitter Tolerance		(7), Figure 12-8			
At Host (Upstream):					
To Next Transition	TUJR1		– 152	152	
For Paired Transitions	TUJR2		– 200	200	
At Function (Downstream):					
To Next Transition	TDJR1		– 75	75	
For Paired Transitions	TDJR2		– 45	45	
EOP Width at Receiver		(7), Figure 12-7			
Must Reject as EOP	TEOPR1		330	–	
Must Accept	TEOPR2		675	–	

NOTES:

1. All voltages measured from the local ground potential, unless otherwise specified.
2. All timings use a capacitive load (CL) to ground of 50 pF, unless otherwise specified.
3. Low speed timings have a 1.5 kΩ pull-up 2.8 V on the D- data line.
4. Measured from 10 % to 90 % of the data signal.
5. The rising and falling edges should be smoothly transitioning (monotonic).
6. Timing difference between the differential data signals.
7. Measured at crossover point of differential data signals.

TIMING WAVEFORMS

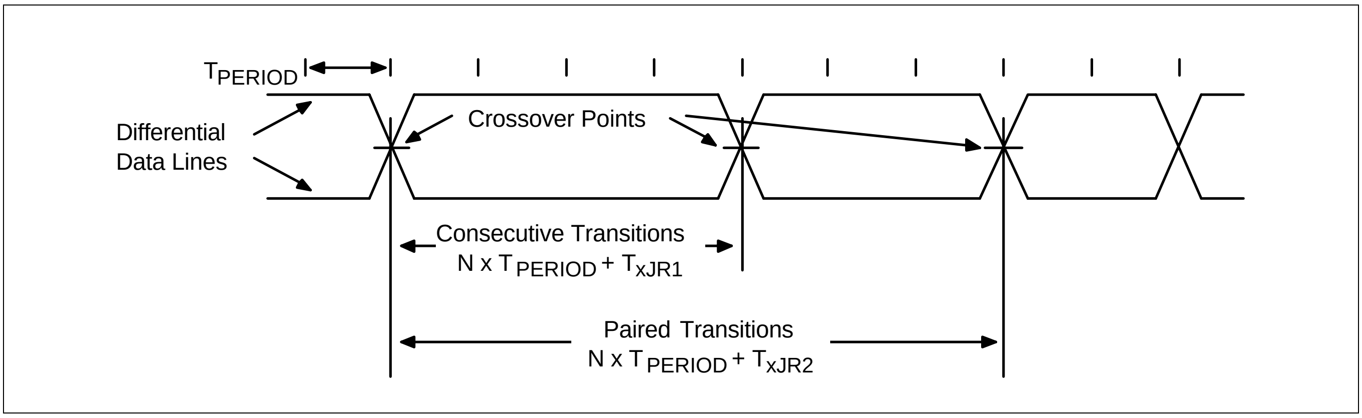


Figure 12-6. Differential Data Jitter

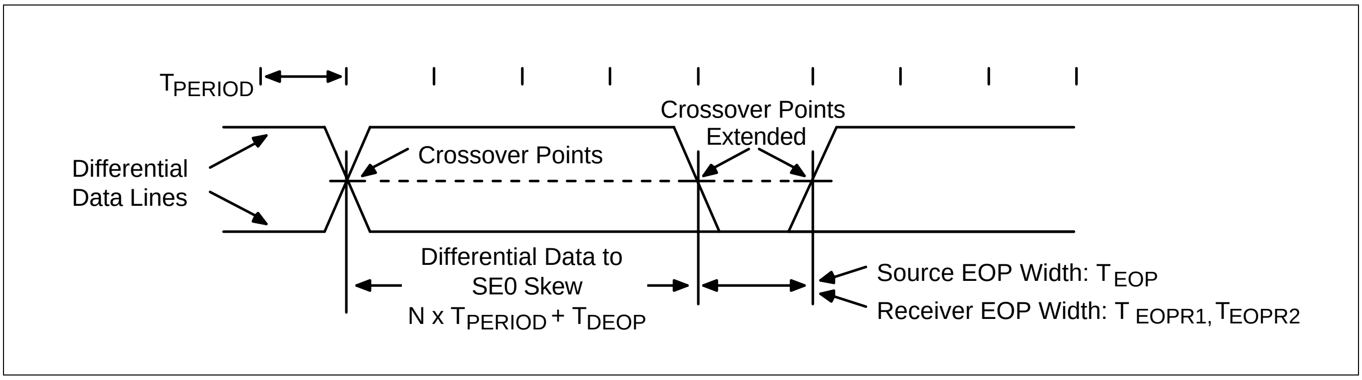


Figure 12-7. Differential to EOP Transition Skew and EOP Width

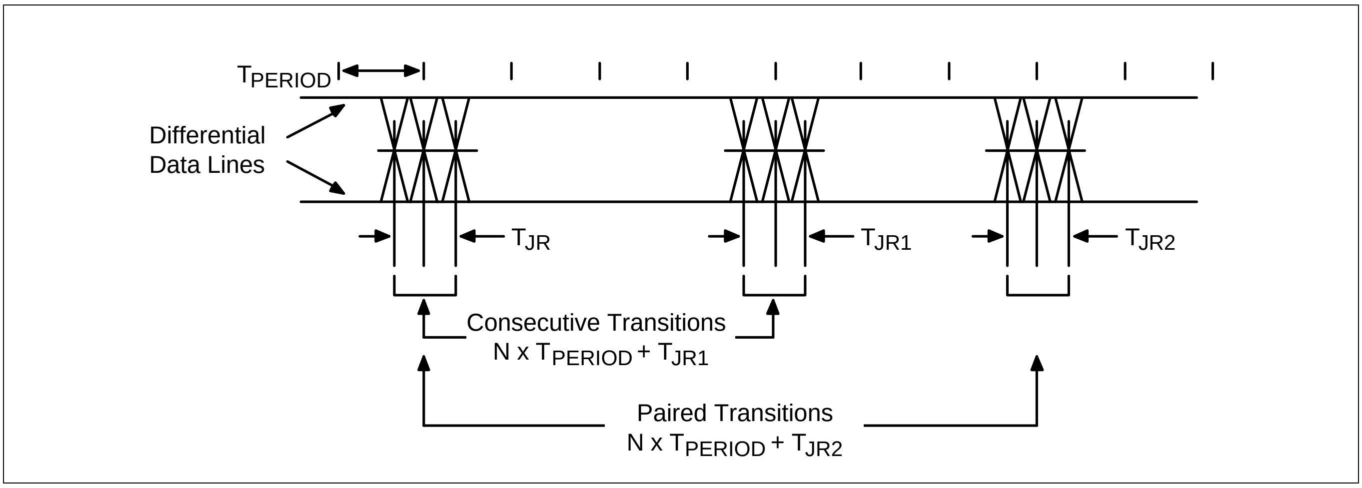


Figure 12-8. Receiver Jitter Tolerance

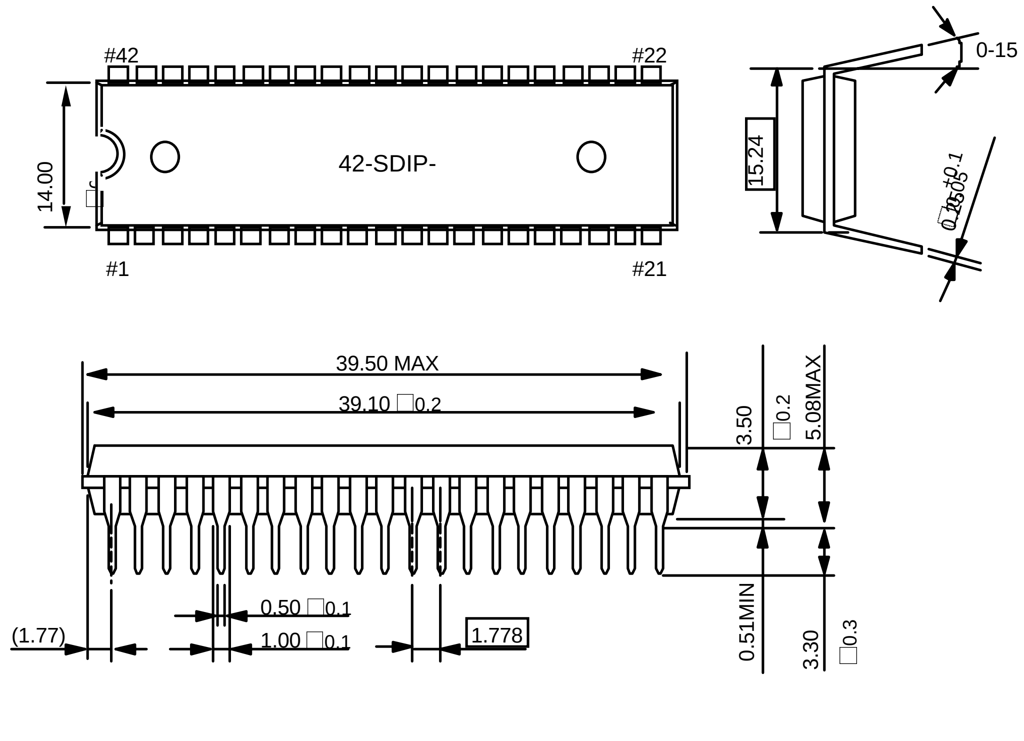
SAMSUNG ELECTRONICS

NOTES

13 MECHANICAL DATA

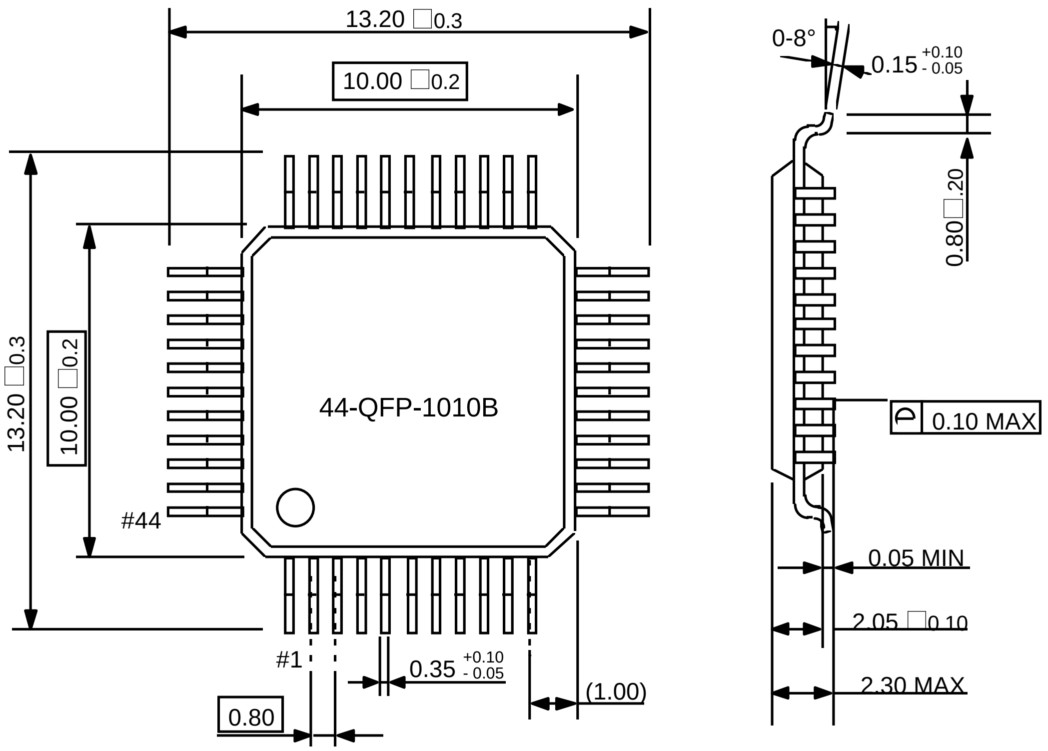
OVERVIEW

The KS86C6004/C6008/P6008 is available in a 42-pin SDIP package (Samsung: 42-SDIP-600) and a 44-pin QFP package (44-QFP-1010B). Package dimensions are shown in Figures 13-1 and 13-2.



NOTE: Dimensions are in millimeters.

Figure 13-1. 42-Pin SDIP Package Mechanical Data (42-SDIP-600)



NOTE: Dimensions are in millimeters.

Figure 13-2. 44-Pin QFP Package Mechanical Data (44-QFP-1010B)

SAMSUNG ELECTRONICS

14 KS86P6008 OTP

OVERVIEW

The KS86P6008 single-chip CMOS microcontroller is the OTP (One Time Programmable) version of the KS86C6004/C6008 microcontroller. It has an on-chip OTP ROM instead of masked ROM. The EPROM is accessed by serial data format.

The KS86P6008 is fully compatible with the KS86C6004/C6008, both in function and in pin configuration. Because of its simple programming requirements, the KS86P6008 is ideal for use as an evaluation chip for the KS86C6004/C6008.

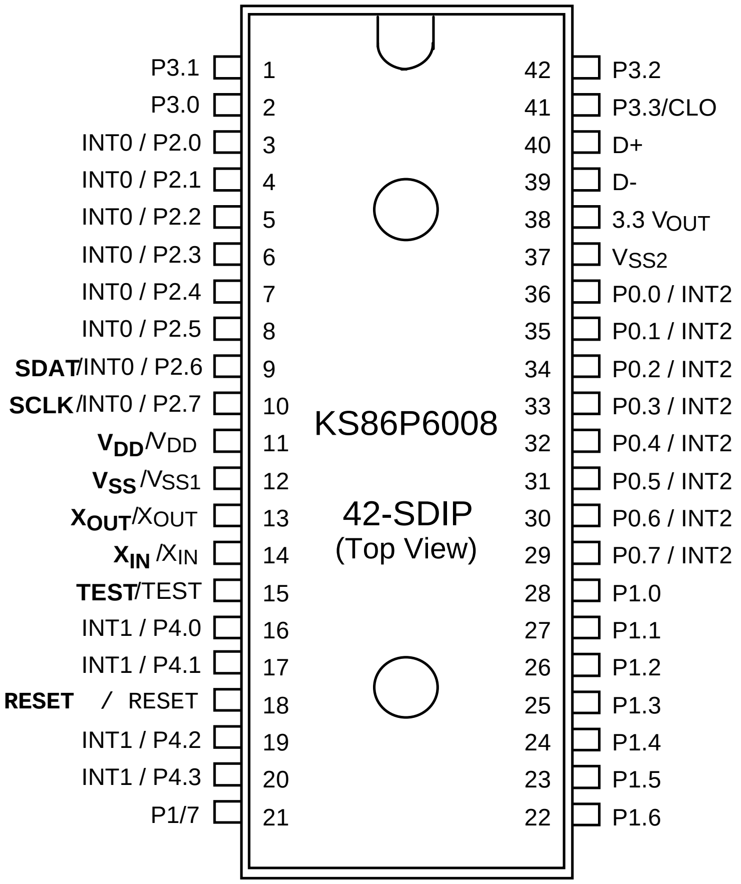


Figure 14-1. KS86P6008 Pin Assignments (42-SDIP Package)

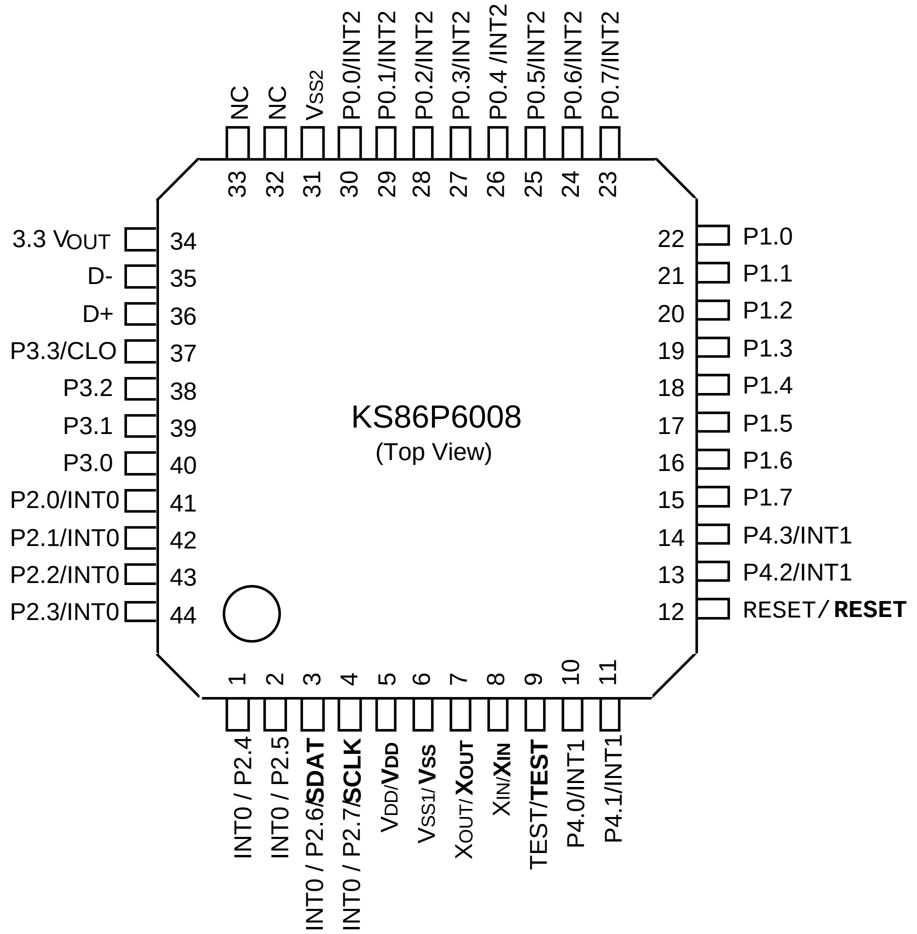


Figure 14-2. KS86P6008 Pin Assignments (44-QFP Package)

Table 14-1. Descriptions of Pins Used to Read/Write the EPROM

Main Chip	During Programming			
Pin Name	**Pin Name**	**Pin No.**	**I/O**	**Function**
P2.6	SDAT	9 (3)	I/O	Serial DATa Pin (Output when reading, Input when writing) Input and Push-pull Output Port can be assigned
P2.7	SCLK	10 (4)	I/O	Serial CLocK Pin (Input Only Pin)
TEST	TEST	15 (9)	I	0 V: OTP write and test mode 5 V: Operating mode
RESET	RESET	18 (12)	I	Chip Initialization and EPROM Cell Writing Power Supply Pin (Indicates OTP Mode Entering) When writing 12.5 V is applied and when reading.
V_{DD} / V_{SS}	V_{DD} / V_{SS}	11(5)/12(6)	–	Logic Power Supply Pin.

NOTE: ( ) means 44 QFP package.

Table 14-2. Comparison of KS86P6008 and KS86C6004/C6008 Features

Characteristic	KS86P6008	KS86C6004/C6008
Program Memory	8-Kbyte EPROM	8-Kbyte mask ROM
Operating Voltage (V_{DD})	4.5 V to 5.5 V	4.5 V to 5.5 V
OTP Programming Mode	V_{DD} = 5 V, V_{PP} (RESET) = 12.5 V	
Pin Configuration	42 SDIP/44 QFP	42 SDIP/44 QFP
EPROM Programmability	User Program 1 time	Programmed at the factory

OPERATING MODE CHARACTERISTICS

When 12.5 V is supplied to the VPP (RESET) pin of the KS86P6008, 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 14-3 below.

Table 14-3. Operating Mode Selection Criteria

V_{DD}	VPP (RESET)	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.

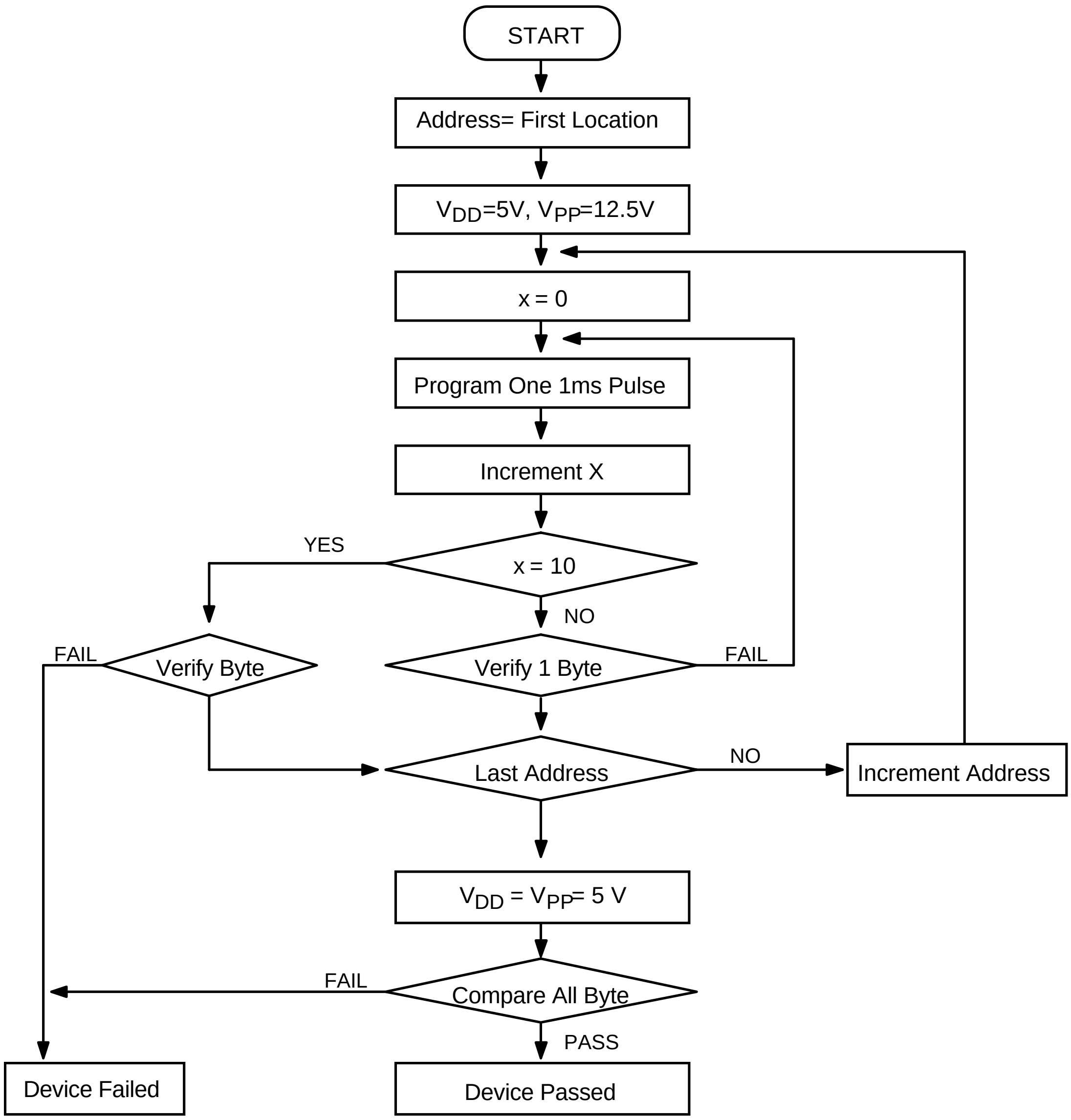


Figure 14-3. OTP Programming Algorithm

Table 14-4. D.C. Electrical Characteristics

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

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Supply Current (note)	I_{DD1}	Normal mode; 6 MHz CPU clock	–	5.5	12	mA
	I_{DD2}	Idle mode; 6 MHz CPU clock		2.2	5	
	I_{DD3}	Stop mode		180	300	µA

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

NOTES