SAMSUNG ELECTRONICS

KS88C3208/3216
8-Bit CMOS Microcontroller

Product Specification

OVERVIEW

The KS88C3208/3216 single-chip 8-bit microcontroller is fabricated using a highly advanced CMOS process. The KS88C3208/3216 is using a modular design approach and the following peripherals are intergrated with the SAM8 core to make the KS88C3208/3216 microcontroller suitable for being used in color television and other types of screen display applications.

FEATURES

CPU
- SAM8 CPU core

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

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

Instruction Execution Time
- 750 ns (minimum) with an 8-MHz CPU clock

Interrupts
- 12 interrupt sources with 12 vectors
- Seven interrupt levels
- Fast interrupt processing for select levels

General I/O
- Four I/O ports (30 pins total)
- 15 bit-programmable pins for general I/O
- Ten open-drain pins for up to 10-volt loads
- Five open-drain pins for up to 5-volt loads

8-Bit Basic Timer
- Three selectable internal clock frequencies
- Watchdog or oscillation stabilization function

Timer/Counters

One 8-bit timer/counter (T0) with three internal c locks or an external clock, and three operating modes; includes capture input module
- Two general-purpose 8-bit timer/counters with interval timer and PWM modes (timers A and B)

I^2C-Bus Interface Controller
- Single master bus controller
- Two pairs of bus interface pins

A/D Converter
- Four analog input pins; 4-bit resolution
- 7.68-μs conversion time (8-MHz CPU clock)

Pulse Width Modulation Module
- 14-bit PWM with 2-channel output (one open-drain and one push-pull)
- 8-bit PWM with 4-channel, open-drain output
- PWM counter and data capture input pin
- Frequency: 5.859 kHz to 23.437kHz with a 6 -MHz CPU clock

On-Screen Display (OSD)
- Video RAM: 240 × 11 bits
- Character generator ROM: 128 × 18 × 12 bits (128 display characters: fixed: 2, variable: 126)
- 240 display positions (10 rows × 24 columns)
- 12-dot × 18-dot character resolution
- 16 different character sizes
- Eight character colors
- Eight colors for character and frame background
- Halftone control signal output; selectable for individual characters
- Vertical direction fade-in/fade-out control
- Fringe function
- Synchronous polarity selector for H-sync and V-sync input

Oscillator Frequency
- 5-MHz to 8-MHz external crystal oscillator
- Maximum 8-MHz CPU clock

Operating Temperature Range
- – 25°C to + 85°C

Operating Voltage Range
- 4.5 V to 5.5 V

Package Type
- 42-pin SDIP

BLOCK DIAGRAM

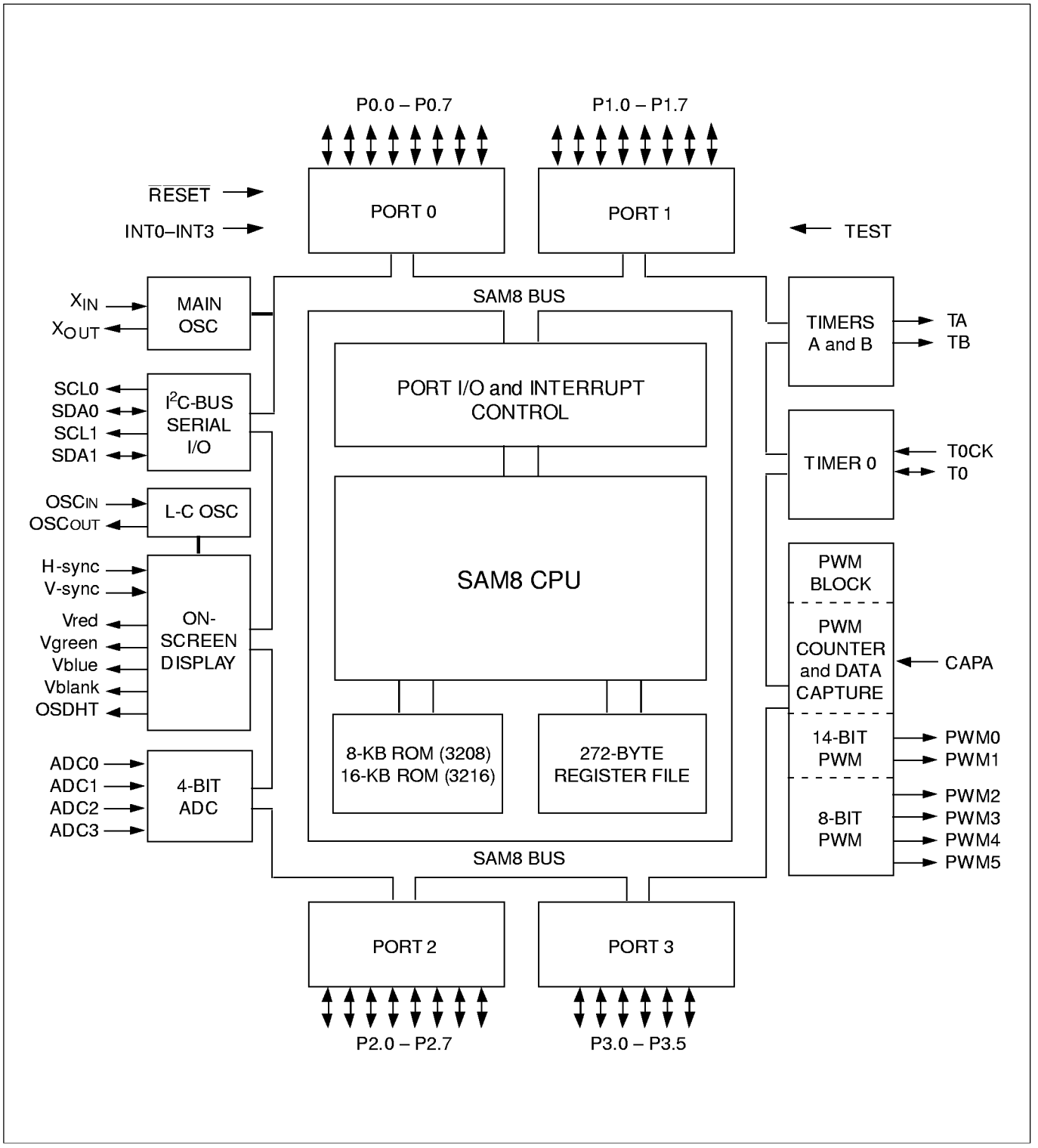


Figure 1. KS88C3208/3216 Block Diagram

PIN ASSIGNMENTS

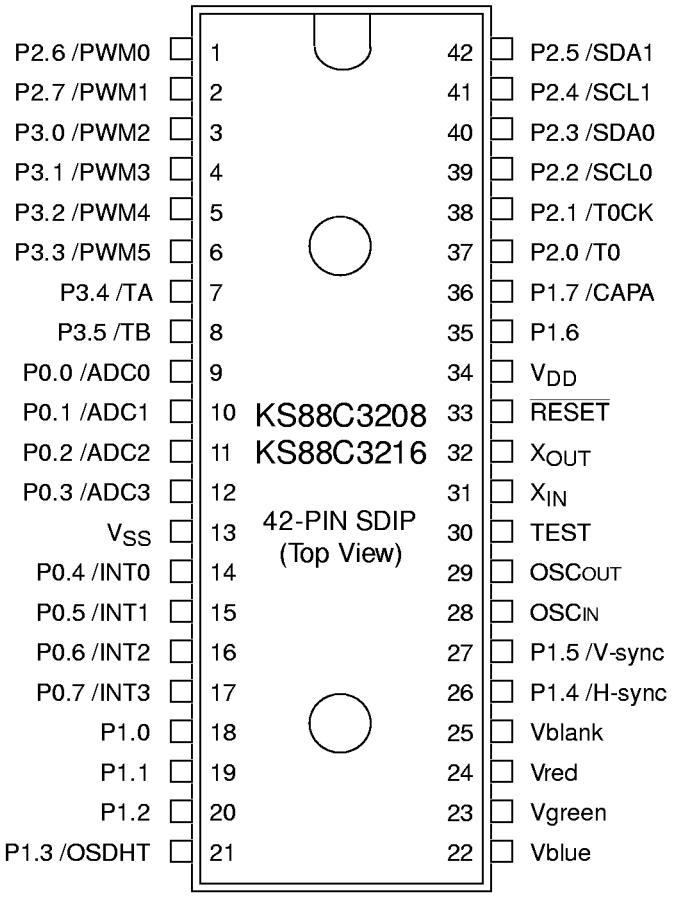


Figure 2 KS88C3208/3216 Pin Assignment Diagram

Table 1. KS88C3208/3216 Pin Descriptions

Pin Name	Pin Type	Pin Description	Circuit Type	Pin Numbers	Share Pins
P0.0–P0.3	I/O	General I/O port (4-bit), configurable for digital input or n-channel, open-drain output. P0.0–P0.3 can withstand up to 5-volt loads. Multiplexed for alternative use as A/D converter inputs ADC0–ADC3	6	9–12	ADC0–ADC3
P0.4–P0.7		General I/O port (4-bit), configurable for digital input or n-channel, open-drain output. P0.4–P0.7 can withstand up to 10-volt loads. Multiplexed for alternative use as external interrupt inputs INT0–INT3.	7	14–17	INT0–INT3
P1.0–P1.7	I/O	General I/O port (8-bit). Input or output mode is software configurable. Pull-up resistors can be assigned to individual pins. Four alternative functions are supported: P1.3: OSDHT (Halftone signal output) P1.4: H-sync (H-sync input for OSD) P1.5: V-sync (V-sync input for OSD) P1.7: CAPA (capture A input)	3	18–21, 26, 27, 35, 36	(See pin description)
P2.0–P2.6	I/O	General I/O port (7-bit). Input/output mode or n-channel, open-drain output mode is software configurable. Pins can withstand up to 5-volt loads. Pull-up resistors are assignable to individual pins. Each pin has an alternative function: P2.0: T0 (timer 0 input and output) P2.1: T0CK (timer 0 clock input) P2.2: SCL0 (I^2C-bus clock output 0) P2.3: SDA0 (I^2C-bus data I/O 0) P2.4: SCL1 (I^2C-bus clock output 1) P2.5: SDA1 (I^2C-bus data I/O 1) P2.6: PWM0 (14-bit PWM0 output)	2	37–42, 1	(See pin description)
P2.7	I/O	Same as P2.0–P2.6, except with circuit 6, which can withstand up to 10-volt loads. P2.7 is used alternatively as the PWM1 output for the 14-bit PWM module.	5	2	PWM1
P3.0–P3.5	I/O	General I/O port (6 bits), configurable for digital input or n-channel, open-drain output. Pins can withstand up to 10-volt loads. P3.0–P3.3 alternate as outputs for the 8-bit PWM circuits; P3.4 and P3.5 alternate as output pins for timers A and B.	5	3–8	PWM2–PWM5, TA, TB
PWM0	O	Output pin for 14-bit PWM0 circuit	2	1	P2.6
PWM1	O	Output pin for 14-bit PWM1 circuit	5	2	P2.7
PWM2–PWM5	O	Output pins for 8-bit PWM circuits	5	3–6	P3.0–P3.3
TA, TB	I/O	Output pins for 8-bit timer A and timer B	5	7, 8	P3.4, P3.5

Table 1 KS88C3208/3216 Pin Descriptions (Continued)

Pin Name	Pin Type	Pin Description	Circuit Type	Pin Numbers	Share Pins
ADC0–ADC3	I	Analog inputs for 4-bit A/D converter	6	9–12	P0.0–P0.3
INT0–INT3	I	External interrupt input pins	7	14–17	P0.4–P0.7
OSDHT	O	Halftone control signal output for OSD	3	21	P1.3
Vblue Vgreen Vred Vblank	O	Digital blue, green, red, and video blank signal outputs for OSD	4	22–25	–
H-sync	I	H-sync input for OSD	3	26	P1.4
V-sync		V-sync input for OSD		27	P1.5
OSC_{IN}, OSC_{OUT}	I, O	L-C oscillator pins for OSD clock frequency generation	–	28, 29	–
TEST	–	OSD Block GND PIN	–	30	–
X_{IN}, X_{OUT}	I, O	System clock pins	–	31,32	–
$\overline{RESET}$	I	System reset input pin,Factory test mode is activated when 12V is applied.	1	33	–
V_{DD}, V_{SS}	–	Power supply pins	–	34, 13	–
CAPA	I	Input for capture A module	3	36	P1.7
T0	I/O	Timer 0 input and output	2	37	P2.0
T0CK	I	Timer 0 clock input	2	38	P2.1
SCL0	O	I^2C-bus clock output pin 0	2	39	P2.2
SDA0	I/O	I^2C-bus data I/O pin 0	2	40	P2.3
SCL1	O	I^2C-bus clock output pin 1	2	41	P2.4
SDA1	I/O	I^2C-bus data I/O pin 1	2	42	P2.5

PIN CIRCUITS

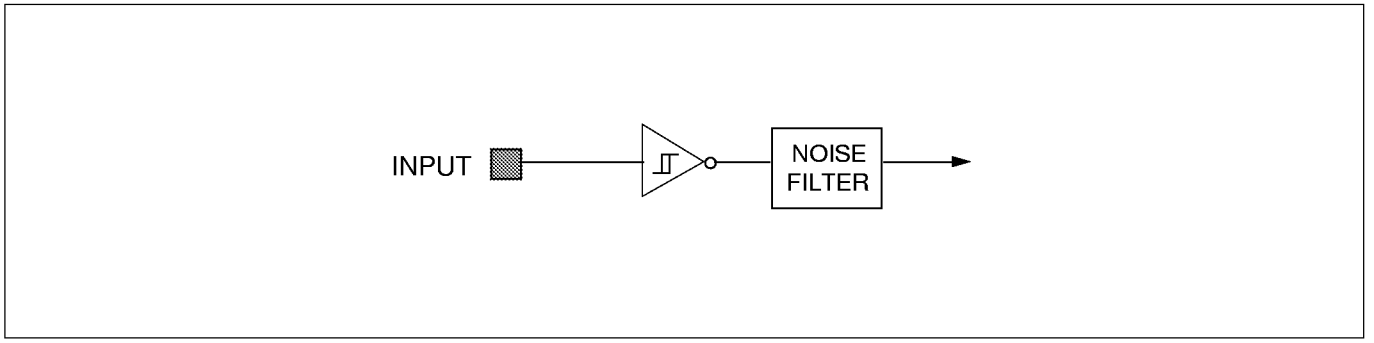


Figure 3. Pin Circuit Type 1 ($\overline{RESET}$)

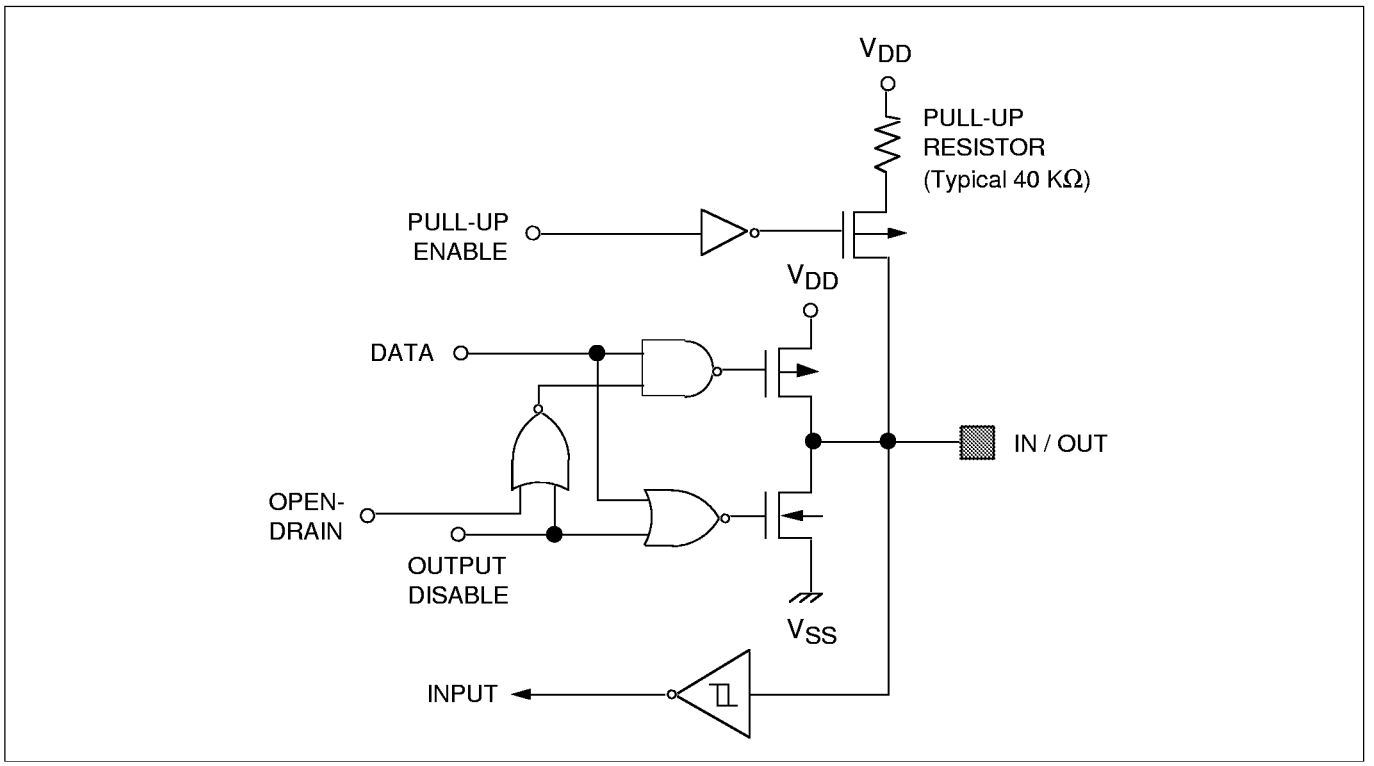


Figure 4. Pin Circuit Type 2 (P2.0–P2.6, PWM0, T0, T0CK, SCL0, SDA0, SCL1, SDA1)

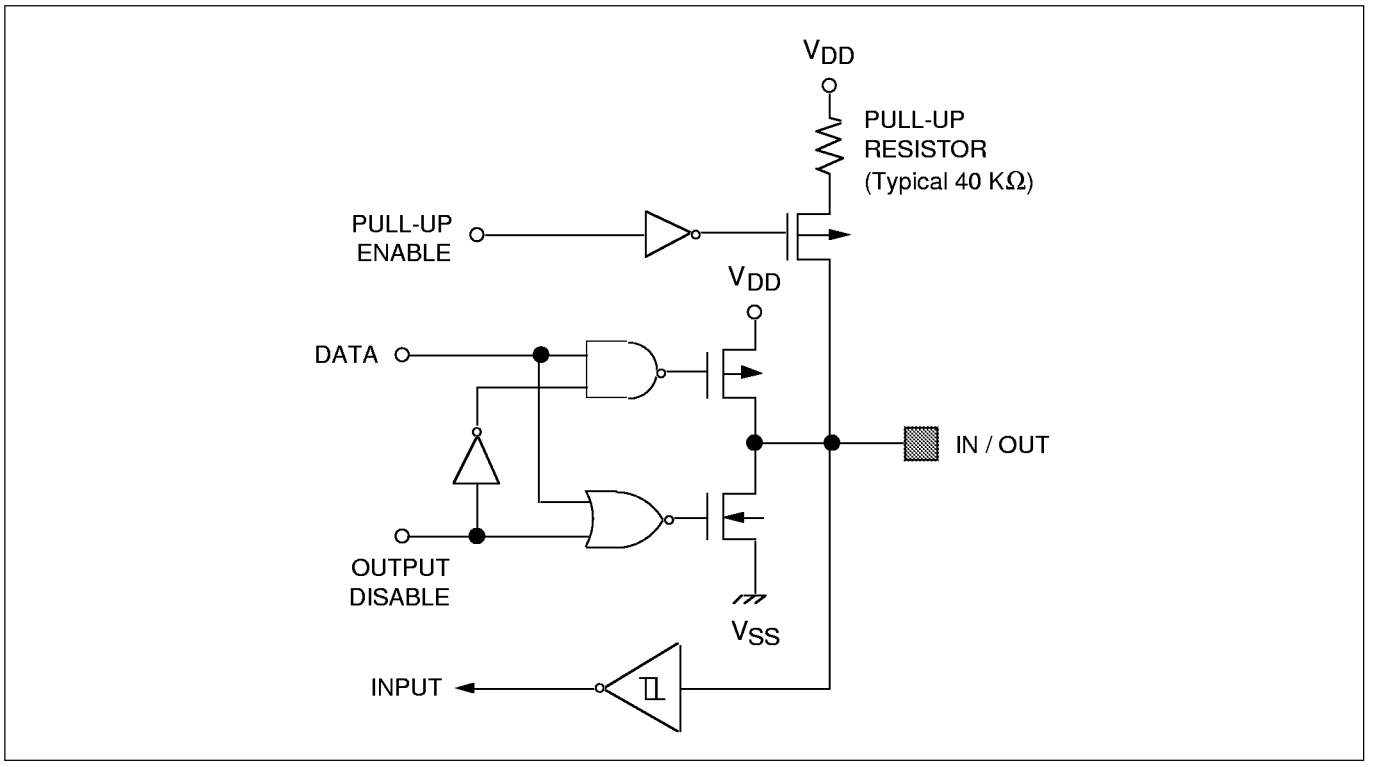


Figure 5 Pin Circuit Type 3 (P1.0–P1.7, OSDHT, H-sync, V-sync, CAPA)

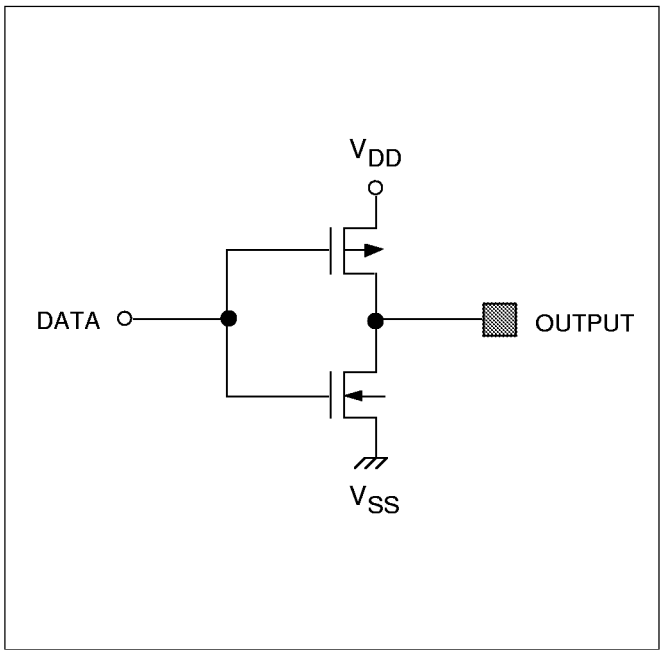


Figure 6. Pin Circuit Type 4 (Vblue, Vgreen, Vred, Vblank)

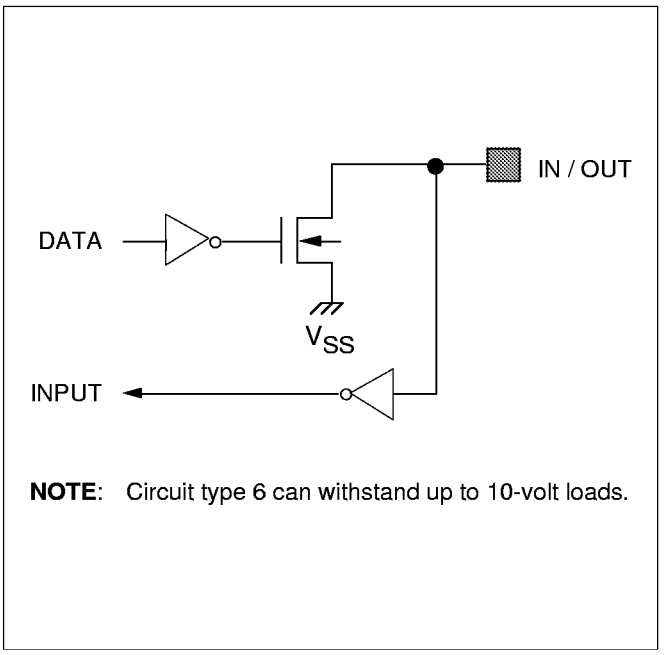


Figure 7. Pin Circuit Type 5 (P2.7, P3.0–P3.5, PWM1, PWM2–PWM5, TA, TB)

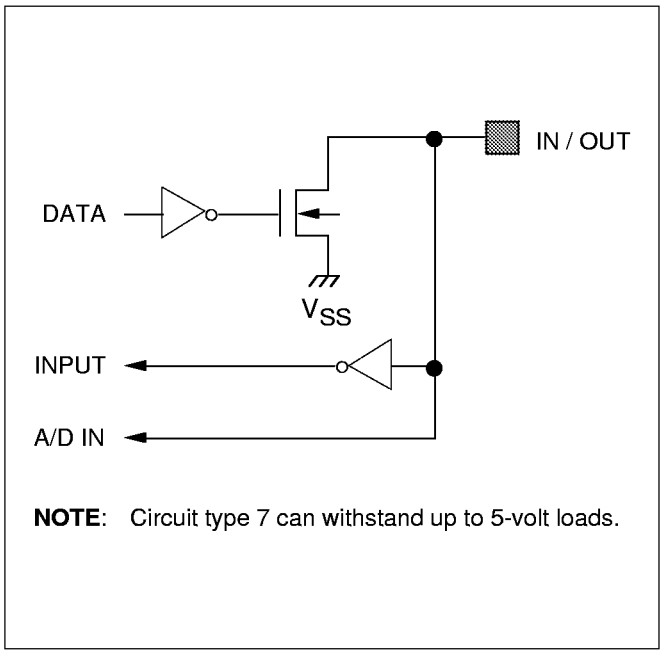


Figure 8 Pin Circuit Type 6 (P0.0–P0.3)

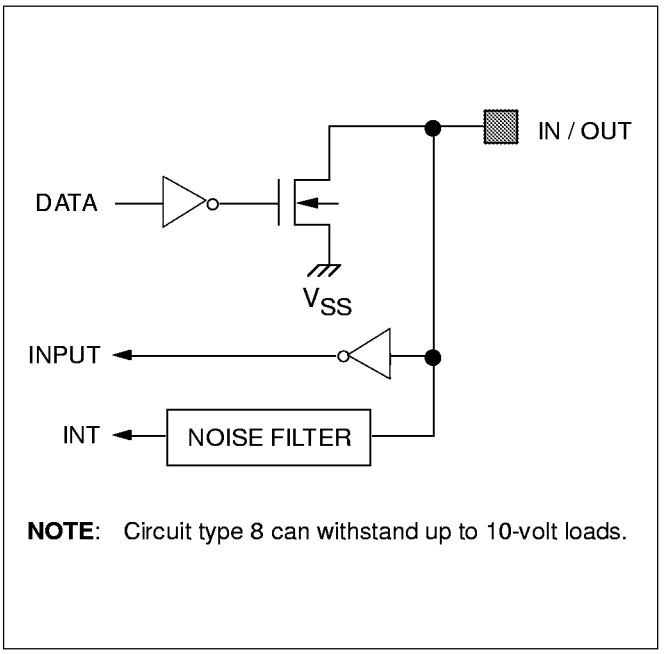


Figure 9. Pin Circuit Type 7 (P0.4–P0.7, INT0–INT3)

ADDRESS SPACES

OVERVIEW

The KS88C3208/3216 microcontroller supports two types of address space:

— Internal program memory (ROM)

— Internal register file

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

The KS88C3208 has a 8-Kbyte and KS88C3216 has a 16-Kbyte mask-programmable ROM. An external memory interface is not implemented.

There are 272 general-purpose data registers in the internal register file. These registers can serve as either a source or destination address, or as accumulators for data memory operations.

Sixteen 8-bit registers are used for CPU and system control. For peripheral functions, there are 29 control registers, 13 data registers. In addition, there is a 240-byte area for on-screen display (OSD) video RAM.

PROGRAM MEMORY (ROM)

Program memory (ROM) stores program code or table data.

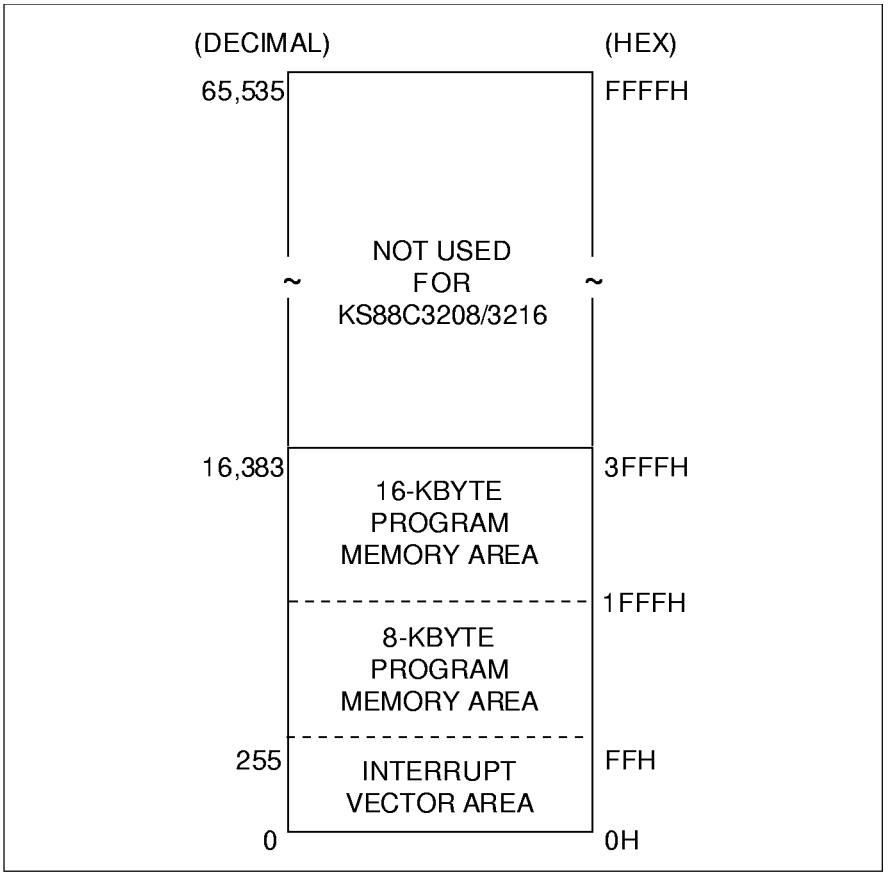


Figure 10. KS88C3208/3216 Program Memory Map

Instructions can be fetched, or data read, from the ROM. The KS88C3208 has a 8-Kbyte mask-programmable program memory (0H-1FFFH). KS88C3216 has a 16-Kbyte mask-programmable program memory (0H–3FFFH). The reset address in the ROM is 0100H.

The KS88 interrupt structure can support up to 127 vectors. (The KS88C3208/3216 uses eight vectors.) The first 256 bytes of the ROM (0H–FFH) are reserved for the maximum number of vector addresses. You can allocate unused locations as normal program memory. Be careful, however, to avoid overwriting vector addresses stored in this area.

REGISTER ARCHITECTURE

Internal Register File

The upper 64-byte area of the internal register file is logically extended to create the *set 1* and *set 2* address spaces.

The upper 32-byte area of set 1 is further divided into two register banks, called *bank 0* and *bank 1*.

In addition, the basic 256-byte register space of the KS88C3208/3216 is expanded into separately addressable 256-byte *pages*.

These extensions into separately addressable register sets, banks, and pages are realized by the combined implementation of:

— Select bank instructions (SB0 and SB1)

— Register page pointer (PP)

— Addressing mode restrictions

The total addressable register space is thereby expanded from 256 bytes to 1120 bytes. The KS88C3208/3216 can access 579 registers in this 1120-byte space.

Register Paging Concept

The addressable area of the 256-byte register file is logically extended into four 256-byte pages (pages 0–3). Only page 0, page 1, and a small section of page 2 are used, however, for the standard KS88C3208/3216 implementation.

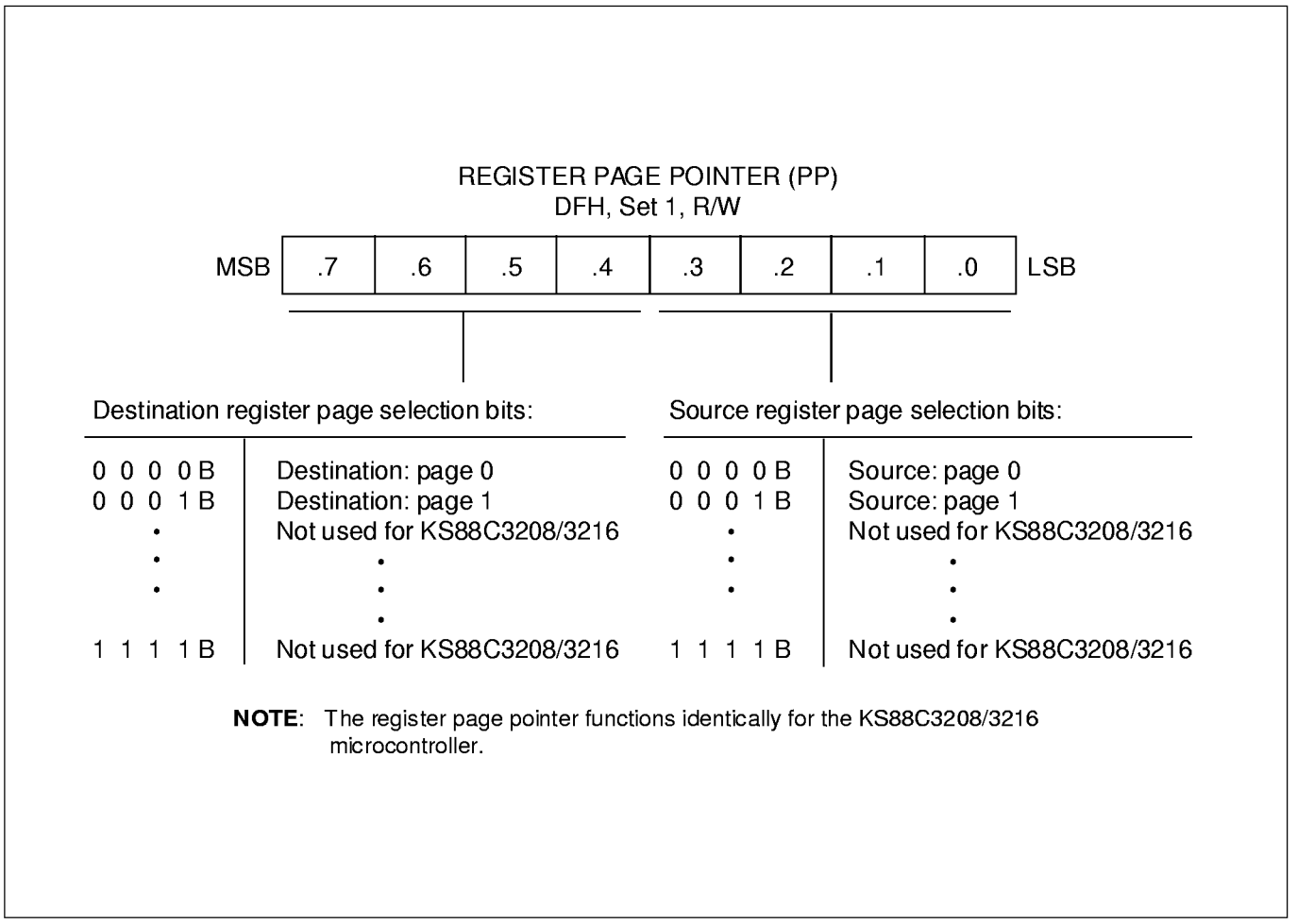
After a reset, the page pointer always points to page 0. The register page pointer and addressing mode restr ictions support register addressing for pages 0–2.

Register Page Pointer (PP)

The SAM8 architecture supports the logical expansion of the physical 256-byte register file in up to 16 separately addressable register pages. Page addressing is controlled by the register page pointer, PP, DFH). Only two pages are implemented in the KS88C3208/3216 microcontroller: Page 0 is used as general-purpose register space and page 1 contains a 240 × 11-bit area for the on-screen display (OSD) video ROM.

As shown in Figure 12, when the upper nibble of the PP register is '0000B', the selected destination address is located on page 0. When the upper nibble value is '0001B', page 1 is the selected destination. The lower nibble of the page pointer controls the source register page destination addressing: when the lower nibble is '0000B', page0 is the selected source register page; when the lower nibble is '0001B', page 1 is the source register page.

Following a reset, the page pointer's source value (lower nibble) and destination value (upper nibble) are always '0000B', automatically selecting page 0 as both source and destination. To select page 1 as the source or destination register page, you must modify the register page pointer values accordingly. Because only page 0 and page 1 are used in the KS88C3208/3216 implementation, only pointer values '0000B' and '0001B' are used.



NOTE: The register page pointer functions identically for the KS88C3208/3216 microcontroller.

Figure 11. Register Page Pointer (PP)

Register Set 1

The term *set 1* refers to the upper 64 bytes of the register file, locations C0H–FFH. This area can be accessed at any time, regardless of which page is currently selected. The upper 32-byte area of this 64-byte space is divided into two 32-byte register banks, called *bank 0* and *bank 1*. You use the select register bank instructions, SB0 or SB1, to address one bank or the other. A reset operation automatically selects bank 0 addressing.

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

Registers in set 1 are directly accessible at all times using the Register addressing mode. The 16-byte working register area can only be accessed using working register addressing, however. Working register addressing is a function of Register addressing mode (see Section 3, "Addressing Modes," for more information).

Register Set 2

The same 64-byte physical space that is used for set 1 register locations C0H–FFH is logically duplicated to add another 64 bytes. This expanded area of the register file is called *set 2*. The logical division of set 1 and set 2 is maintained by means of addressing mode restrictions: While you can access set 1 using Register addressing mode only, you can only use Register Indirect addressing mode or Indexed addressing mode to access set 2.

For the KS88C3208/3216, the set 2 address range (C0H–FFH) is accessible on page 0 and on page 1. Please note, however, that on page 1, set 2 locations F0H–FFH are not mapped.

Part of the OSD video RAM is in page 1, set 2 (C0H–EFH), and the other part (00H–BFH) is in the page 1 prime register area. To avoid programming errors, we recommend using either Register Indirect or Indexed mode to address the entire 240-byte video RAM area.

Prime Register Space

The lower 192 bytes (00H–BFH) of the KS88C3208/3216's two 256-byte register pages is called *prime register area.* Prime registers can be accessed using any of the seven addressing modes. The prime register area on page 0 is immediately addressable following a reset. In order to address registers on page 1 (in the OSD video RAM), you must first set the register page pointer (PP) to the appropriate source and destination values.

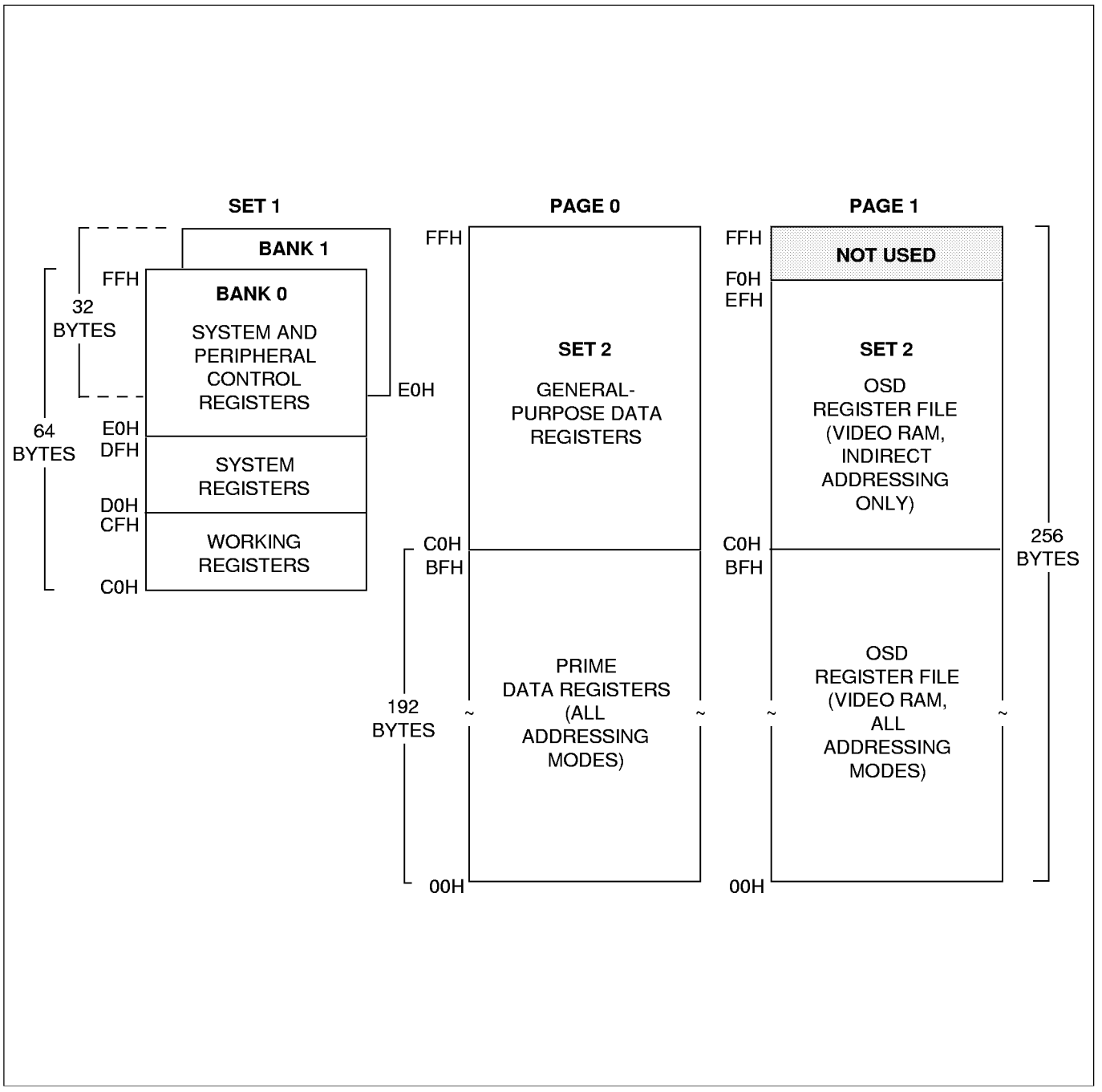


Figure 12. KS88C3208/3216 Internal Register File

CONTROL REGISTERS

Table 2. KS88C3208/3216 Set 1 Registers

Register Name	Mnemonic	Decimal	Hex	R/W
Timer 0 counter	T0CNT	208	D0H	R
Timer 0 data register	T0DATA	209	D1H	R/W
Timer 0 control register	T0CON	210	D2H	R/W
Basic timer control register	BTCON	211	D3H	R/W
Clock control register	CLKCON	212	D4H	R/W
System flags register	FLAGS	213	D5H	R/W
Register pointer 0	RP0	214	D6H	R/W
Register pointer 1	RP1	215	D7H	R/W
Stack pointer (high byte)	SPH	216	D8H	R/W
Stack pointer (low byte)	SPL	217	D9H	R/W
Instruction pointer (high byte)	IPH	218	DAH	R/W
Instruction pointer (low byte)	IPL	219	DBH	R/W
Interrupt request register	IRQ	220	DCH	R
Interrupt mask register	IMR	221	DDH	R/W
System mode register	SYM	222	DEH	R/W
Register page pointer	PP	223	DFH	R/W

Table 3. KS88C3208/3216 Set 1, Bank 0 Registers

Register Name	Mnemonic	Decimal	Hex	R/W
Port 0 data register	P0	224	E0H	R/W
Port 1 data register	P1	225	E1H	R/W
Port 2 data register	P2	226	E2H	R/W
Port 3 data register	P3	227	E3H	R/W
Port 0 control register (high byte)	P0CONH	228	E4H	R/W
Port 0 control register (low byte)	P0CONL	229	E5H	R/W
Port 1 control register (high byte)	P1CONH	230	E6H	R/W
Port 1 control register (low byte)	P1CONL	231	E7H	R/W
Port 2 control register (high byte)	P2CONH	232	E8H	R/W
Port 2 control register (low byte)	P2CONL	233	E9H	R/W
Port 3 control register (high byte)	P3CONH	234	EAH	R/W
Port 3 control register (low byte)	P3CONL	235	EBH	R/W
Port 2 pull-up resistor control register	P2PUR	236	ECH	R/W
I^2C-bus data shift register	IIC	237	EDH	R/W
I^2C-bus control register	IICCON	238	EEH	R/W
I^2C-bus prescaler	IICPS	239	EFH	R/W

Table 3. KS88C3208/3216 Set 1, Bank 0 Registers (Continued)

Register Name	Mnemonic	Decimal	Hex	R/W
Timer A data register	TADATA	240	F0H	R/W
Timer B data register	TBDATA	241	F1H	R/W
Timer A control register	TACON	242	F2H	R/W
Timer B control register	TBCON	243	F3H	R/W
PWM0 data register (main byte)	PWM0	244	F4H	R/W
PWM0 data register (extension byte)	PWM0EX	245	F5H	R/W
PWM1 data register (main byte)	PWM1	246	F6H	R/W
PWM1 data register (extension byte)	PWM1EX	247	F7H	R/W
PWM control register	PWMCON	248	F8H	R/W
Capture A data register	CAPA	249	F9H	R
A/D converter control register	ADCON	250	FAH	R/W (1)
Locations FBH and FCH in set 1, bank 0, are not mapped.				
Basic timer counter	BTCNT	253	FDH	R
External memory timing register	EMT	254	FEH	R/W
Interrupt priority register	IPR	255	FFH	R/W

NOTE: Bit 7 and the lower nibble of the ADCON register are read-only.

Table 4. KS88C3208/3216 Set 1, Bank 1 Registers

Register Name	Mnemonic	Decimal	Hex	R/W
Locations E0H–EFH in set 1, bank 1, are not mapped.				
OSD character size control register	CHACON	240	F0H	R/W
OSD fade control register	FADECON	241	F1H	R/W
OSD row position control register	ROWCON	242	F2H	R/W
OSD column position control register	CLMCON	243	F3H	R/W
OSD background color control register	COLCON	244	F4H	R/W
On-screen display control register	DSPCON	245	F5H	R/W
Halftone signal control register	HTCON	246	F6H	R/W
OSD color buffer	COLBUF	247	F7H	R/W
PWM2 data register	PWM2	248	F8H	R/W
PWM3 data register	PWM3	249	F9H	R/W
PWM4 data register	PWM4	250	FAH	R/W
PWM5 data register	PWM5	251	FBH	R/W
Locations FCH–FFH in set 1, bank 1, are not mapped.				

☞ PROGRAMMING TIP — Using Load Instructions to Access Write-Only and Read-Only Registers

You cannot use the instructions OR (Logical OR), AND (Logical AND), CP (Compare), and LDB (Load Bit) to access write-only or read-only registers. Use load (LD) instructions instead, except for LDB. Here are some examples:

Example 1:

```
        OR      T0CON,#04H      ;  Invalid use of logical OR instruction!
```

Use a LD instruction instead to manipulate the T0CON register:

```
        BITS    ST0CON.2        ;  ST0CON is a shadow register for T0CON
        LD      T0CON,ST0CON    ;  Set bit 2 in the T0CON register
```

Example 2:

```
        CP      PWMCON,#3CH     ;  Invalid use of the CP instruction!
        JP      EQ,AAA
        •
        •
        •
AAA     NOP
```

Use a shadow register instead to manipulate the PWMCON register:

```
        CP      SPWMCON,#3CH    ;  SPWMCON is a shadow register for PWMCON
        JP      EQ,AAA
        •
        •
        •
AAA     NOP
```

Example 3:

```
        LDB     IICCON.0,R0     ;  Invalid use of the LDB instruction!
```

Use a shadow register instead to load the value into the IICCON register:

```
        LDB     SIICCON.0,R0    ;  SIICCON is a shadow register for IICCON
        LD      IICCON,SSIOCON
```

INTERRUPT STRUCTURE

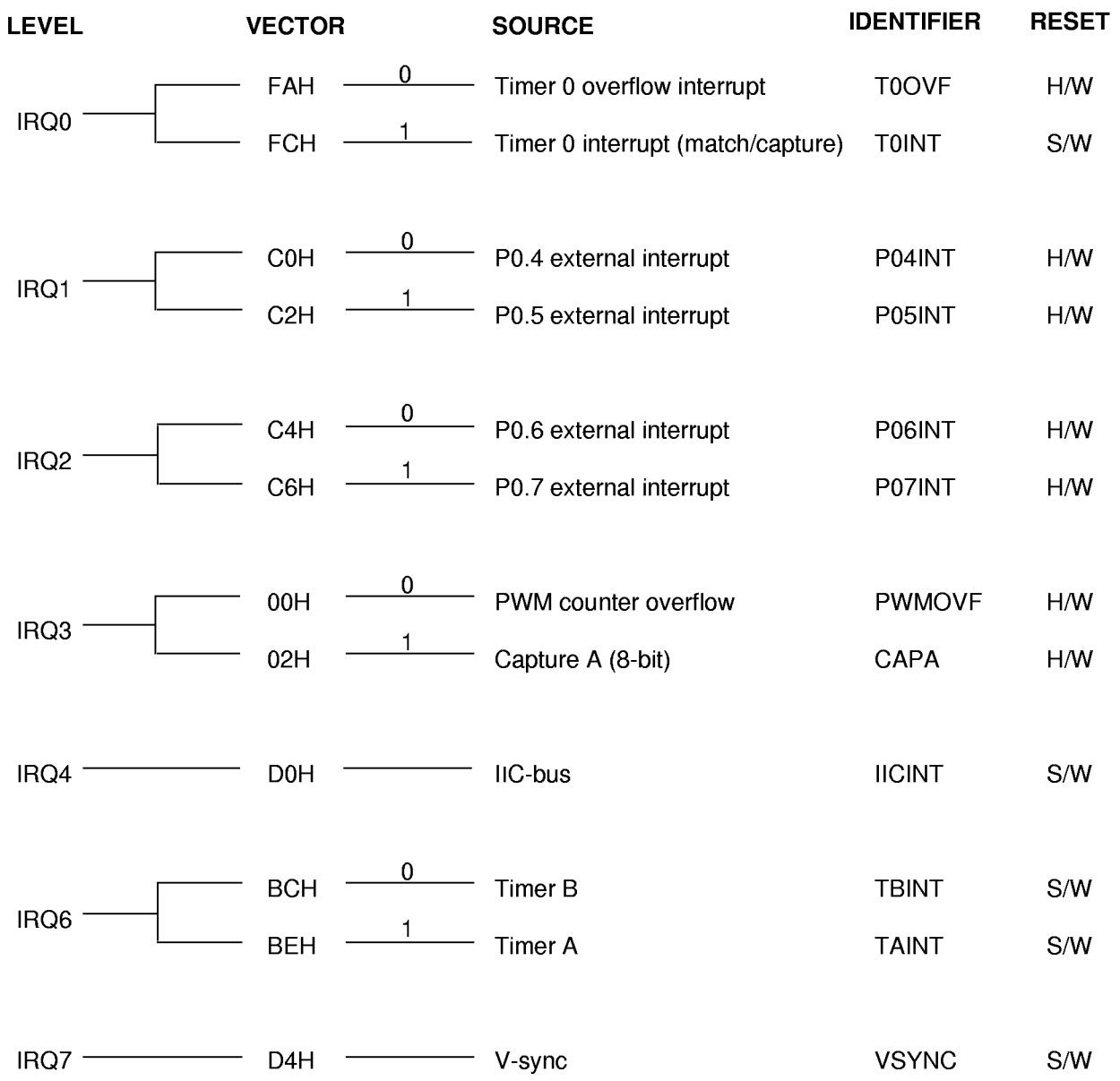
OVERVIEW

The KS88C3208/3216 microcontroller has fourteen peripheral interrupt sources. Eight vector addresses are used to support these sources. Eight interrupt levels are used in the interrupt structure (IRQ0, and IRQ7). This device-specific interrupt structure is illustrated in Figure 14.

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

When an interrupt request is granted, an interrupt machine cycle is entered. All subsequent interrupts are disabled and the program counter and status flags are saved to stack. Program control then branches to the interrupt's vector address. This memory location, together with the next memory byte, comprises the 16-bit address of the interrupt service routine for that particular interrupt request.

Interrupt sources can be external or internal. Internal sources are hardwired to a particular vector and level while external sources can be assigned to various external events. External interrupts are triggered either by rising or falling signal edges, based on the corresponding control register settings.



NOTES:

1. Interrupt level IRQ5 is not used in the KS88C3208/3216 interrupt structure.
2. For interrupt levels with two or more vectors, the lowest vector address usually has the highest priority. For example, FAH has higher priority (0) than FCH (2) within level IRQ0. These priorities (see numbers) are hardwired.
3. The interrupt names in the 'Identifier' column are used in this documentation to refer to specific interrupts, as distinguished from the interrupt source name or the pin at which an external interrupt request arrives.

Figure 13. KS88C3208/3216 Interrupt Structure

INTERRUPT VECTOR ADDRESSES

Interrupt vector addresses for the KS88C3208/3216 are stored in the first 256 bytes of the ROM. The reset address is 0100H. Vectors for all interrupt levels are stored in the vector address area (0H–FFH). Unused ROM in the range 00H–FFH can be used as program memory locations. You must be careful, however, not to overwrite interrupt vector addresses stored in this area.

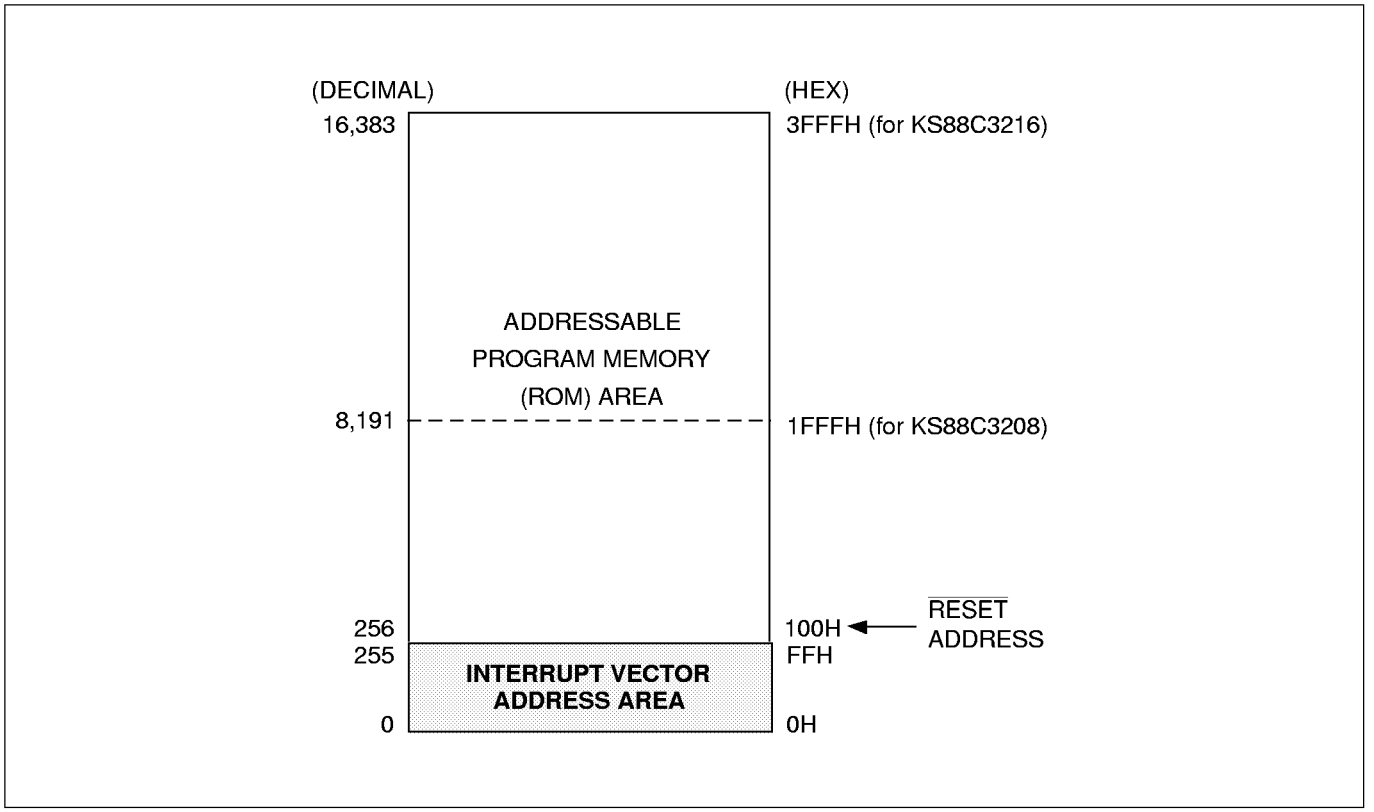


Figure 14. ROM Vector Address Area

Table 5. KS88C3208/3216 Interrupt Vectors

Vector Address		Interrupt Source	Request		Reset/Clear	
Decimal Value	Hex Value		Interrupt Level	Priority in Level	H/W	S/W
252	FCH	Timer 0 (match/cap)	IRQ0	1		√
250	FAH	Timer 0 overflow		0	√	
212	D4H	V-sync	IRQ7	–		√
208	D0H	I^2C-bus	IRQ4	–		√
198	C6H	P0.7 external interrupt	IRQ2	1	√	
196	C4H	P0.6 external interrupt		0	√	
194	C2H	P0.5 external interrupt	IRQ1	1	√	
192	C0H	P0.4 external interrupt		0	√	
190	BEH	Timer A	IRQ6	1		√
188	BCH	Timer B		0		√
2	02H	Capture A (8-bit)	IRQ3	1	√	
0	00H	PWM counter overflow		0	√	

NOTES:
1. Interrupt priorities are identified in inverse order: '0' is highest priority, '1' is the next highest, and so on.
2. If two or more interrupts within the same level contend, the interrupt with the lowest vector address usually has priority over one with a higher vector address. (The priorities within a level are hardwired) For example, in interrupt level IRQ1, the higher-priority interrupt vector is the P0.4 external interrupt, vector C0H; the lower-priority interrupt within that level is the P0.5 external interrupt, vector C2H.

ENABLE/DISABLE INTERRUPT INSTRUCTIONS (EI, DI)

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

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

SYSTEM-LEVEL INTERRUPT CONTROL REGISTERS

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

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

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

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

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

Table 6. Interrupt Control Register Overview

Control Register	ID	R/W	Function Description
System mode register	SYM	R/W	Global interrupt processing enable and disable, fast interrupt processing.
Interrupt mask register	IMR	R/W	Bit settings in the IMR register enable and disable interrupt processing for each of the seven recognized interrupt levels, IRQ0–IRQ4, IRQ6, and IRQ7.
Interrupt priority register	IPR	R/W	Controls the relative processing priorities of the interrupt levels. For the KS88C3208/3216, the seven levels are organized into three groups: A, B, and C. Group A includes IRQ0 and IRQ1, group B is IRQ2–IRQ4, and group C is IRQ6 and IRQ7.
Interrupt request register	IRQ	R	This register contains a request pending bit for each interrupt level.

INTERRUPT PROCESSING CONTROL POINTS

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

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

— Interrupt level enable and disable settings (IMR register)

— Interrupt level priority settings (IPR register)

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

NOTE

When writing an interrupt service routine, be sure that it properly manages the register pointer values (RP0 and RP1).

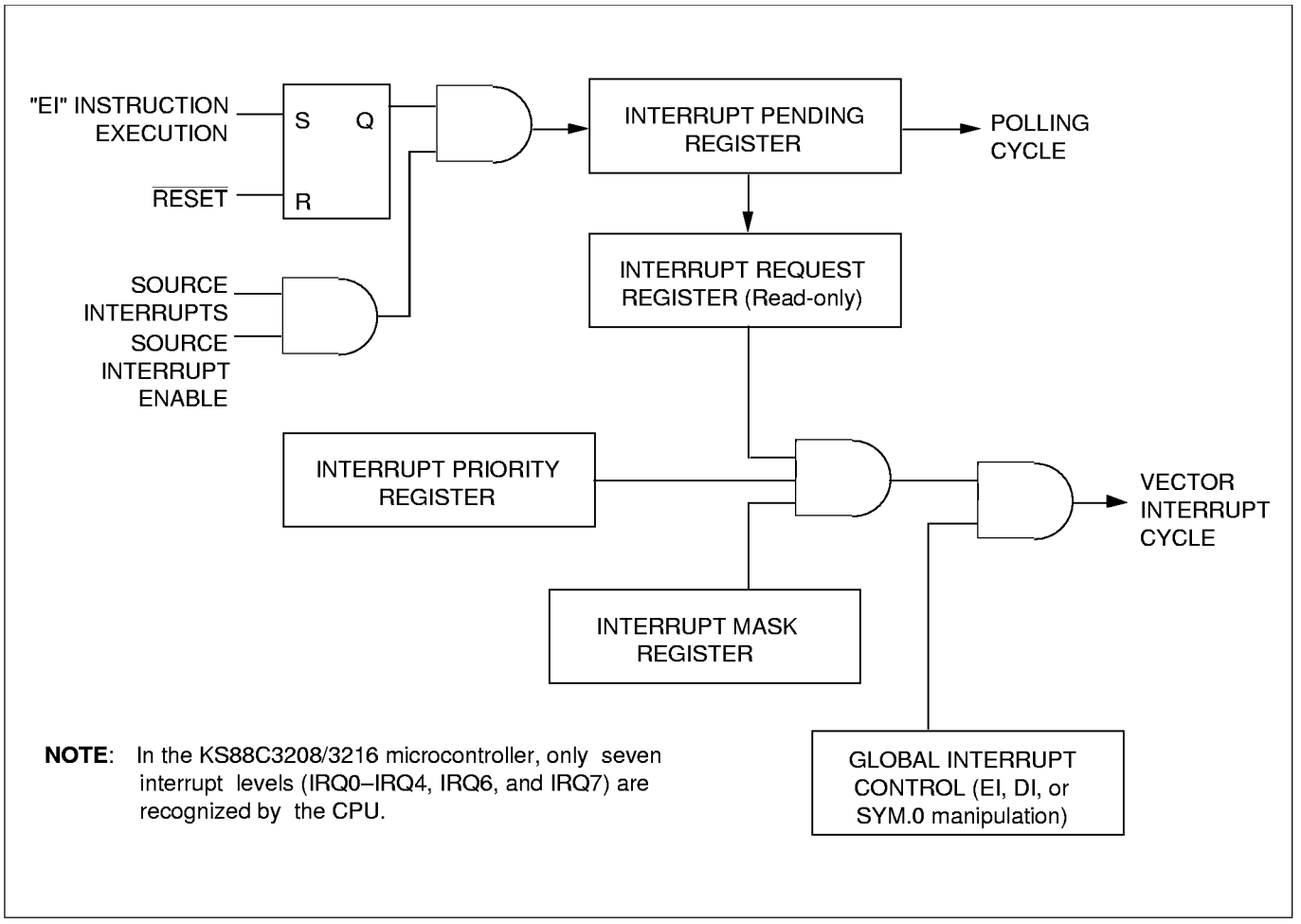


Figure 15. Interrupt Function Diagram

PERIPHERAL INTERRUPT CONTROL REGISTERS

For each interrupt source there is a corresponding peripheral control register (or registers) to control the interrupts generated by that peripheral. These registers and their locations are listed in Table 7.

Table 7. Interrupt Source Control Registers

Interrupt Source	Interrupt Level	Control Register	Register Location
Timer 0 (match/cap) Timer 0 overflow	IRQ0	T0CON	Set 1, D2H
P0.5 external interrupt P0.4 external interrupt	IRQ1	P0CONH	Set 1, bank 0, E4H
P0.7 external interface P0.6 external interface	IRQ2	P0CONH	Set 1, bank 0, E4H
Capture A (8-bit) PWM counter overflow	IRQ3	PWMCON	Set 1, bank 0, F8H
I^2C-bus	IRQ4	IICCON	Set 1, bank 0, EEH
Timer A Timer B	IRQ6	TACON TBCON	Set 1, bank 0, F2H Set 1, bank 0, F3H
V-sync	IRQ7	HTCON	Set 1, bank 1, F6H

SYSTEM MODE REGISTER (SYM)

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

SYM.0 is the enable and disable bit for global interrupt processing. SYM.1–SYM.4 control fast interrupt processing: SYM.1 is the enable bit; SYM.2–SYM.4 are the fast interrupt level selection bits. SYM.7 is the enable bit for the tri-state external memory interface (not implemented in the KS88C3208/3216). A reset clears SYM.0, SYM.1, and SYM.7 to "0"; the other bit values are undetermined.

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

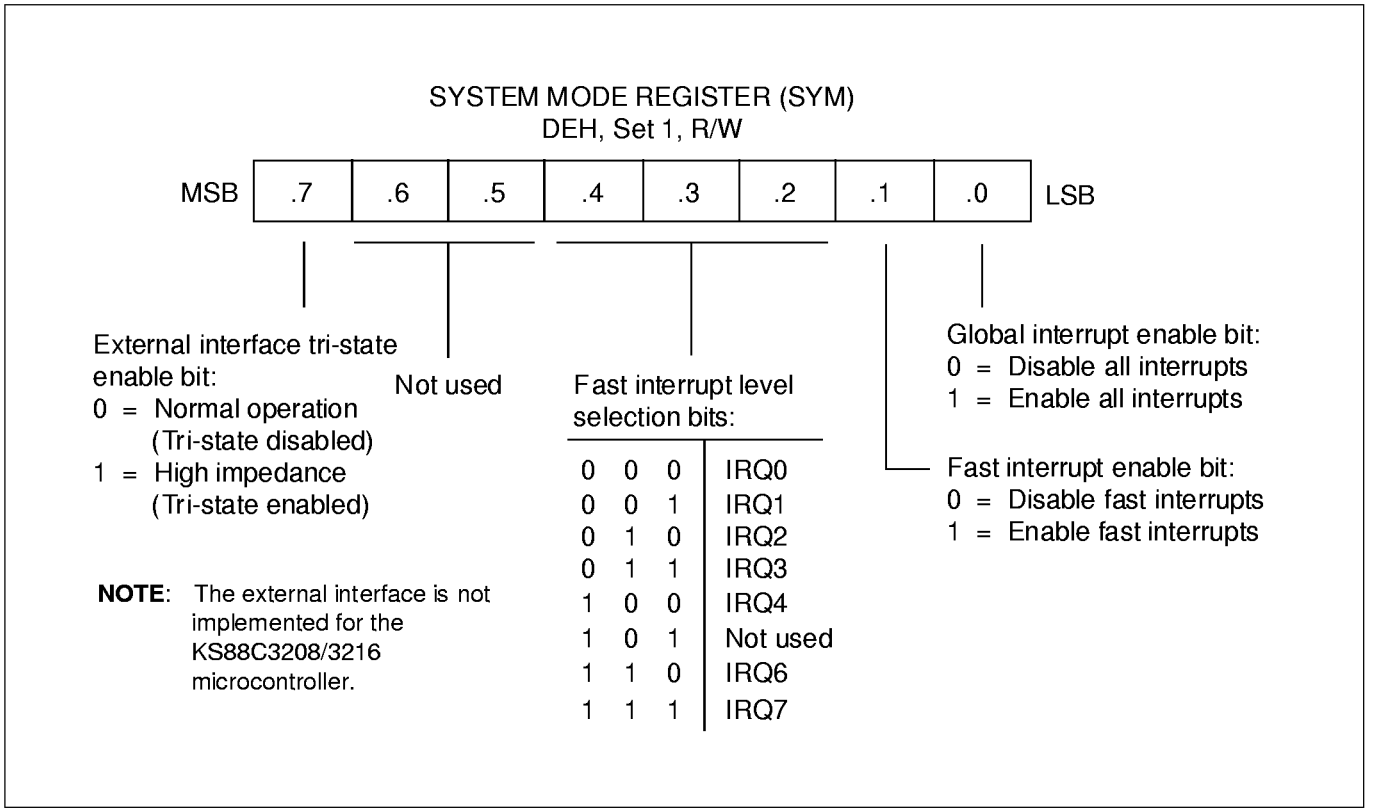


Figure 16. System Mode Register (SYM)

INTERRUPT MASK REGISTER (IMR)

The interrupt mask register (IMR) is used to enable or disable interrupt processing for each of the seven interrupt levels used in the KS88C3208/3216 interrupt structure, IRQ0–IRQ4, IRQ6, and IRQ7. After a reset, all of the IMR register values are undetermined.

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

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

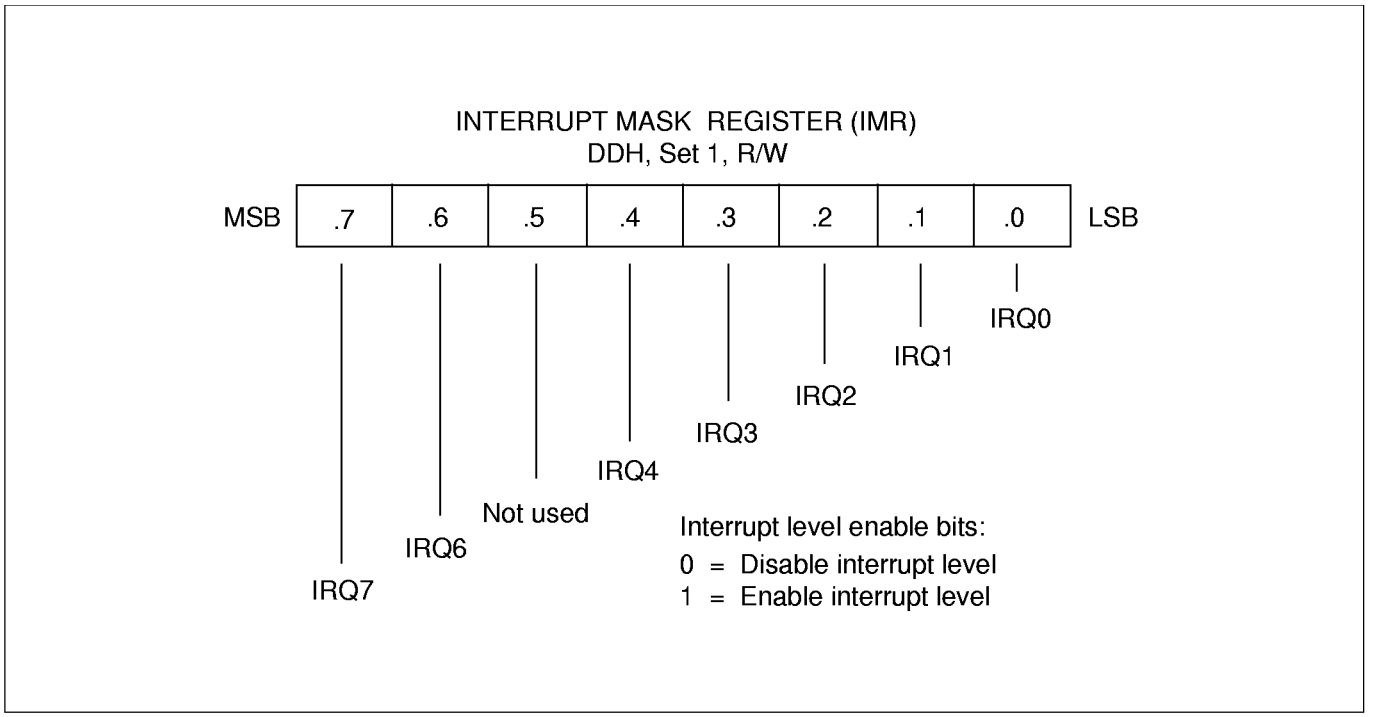


Figure 17. Interrupt Mask Register (IMR)

INTERRUPT PRIORITY REGISTER (IPR)

The interrupt priority register, IPR, is used to set the relative priorities of the seven interrupt levels used in the KS88C3208/3216 interrupt structure. The IPR register is mapped to register location FFH in set 1, bank 0. After a reset, the IPR register values are undetermined. If more than one interrupt source is active, the source with the highest priority level is serviced first. If both sources belong to the same interrupt level, the source with the lowest vector address usually has priority. (This priority is hardwired.)

In order to define the relative priorities of interrupt levels, they are organized into groups and subgroups by the interrupt logic. Three interrupt groups are defined for the IPR logic (see Figure 19). These groups and subgroups are used only for IPR register priority definitions:

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

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

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

— IPR.2 controls interrupt group B.

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

— IPR.6 controls the relative priorities of group C interrupts.

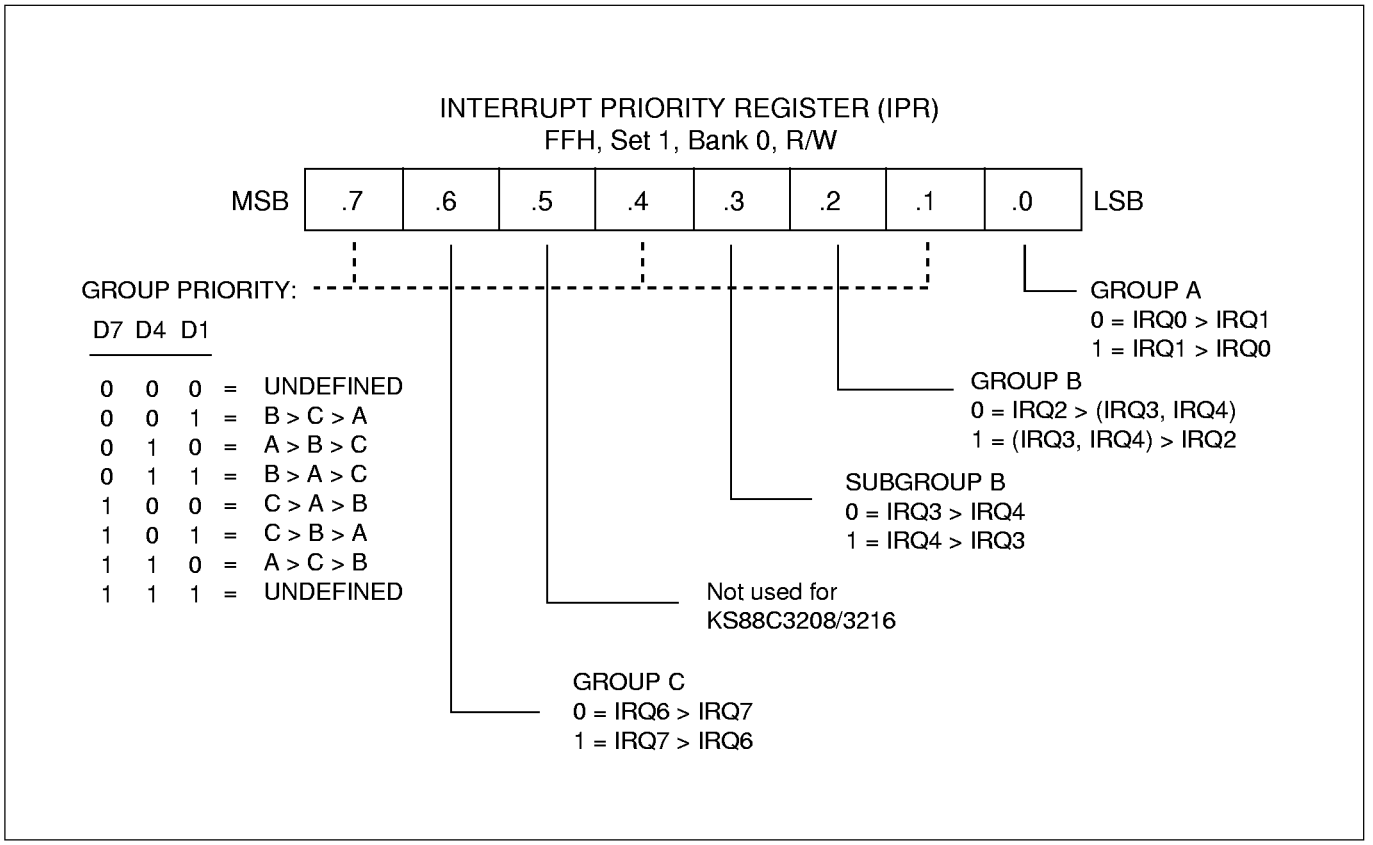


Figure 18. Interrupt Priority Register (IPR)

INTERRUPT REQUEST REGISTER (IRQ)

Bit values in the interrupt request register, IRQ, are polled to determine interrupt request status for the seven interrupt levels in the KS88C3208/3216 interrupt structure (IRQ0–IRQ4, IRQ6, and IRQ7). Each bit corresponds to the interrupt level of the same number: bit 0 to IRQ0, bit 1 to IRQ1, and so on. A "0" indicates that no interrupt is requested and a "1" indicates that an interrupt is requested for that level.

The IRQ register is mapped to register location DCH in set 1. IRQ bit values are read-only addressable using Register addressing mode. You can read (test) the contents of the IRQ register at any time using bit or byte addressing to determine the current interrupt request status of specific interrupt levels. After a reset, the IRQ register is cleared to 00H.

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

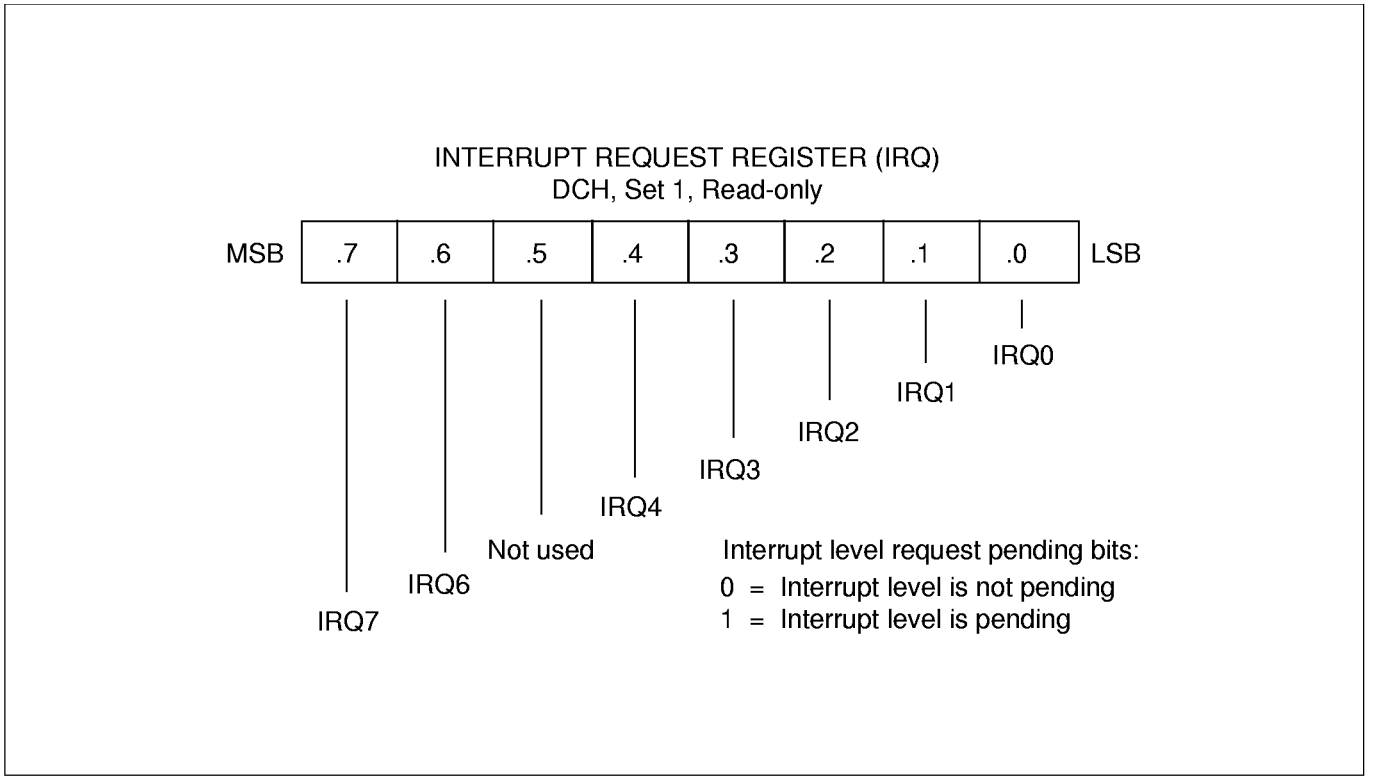


Figure 19. Interrupt Request Register (IRQ)

INTERRUPT PENDING FUNCTION TYPES

Overview

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

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

Pending Bits Cleared Automatically by Hardware

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

In the KS88C3208/3216 interrupt structure, the timer 0 overflow interrupt , the P0.4–P0.7 external interrupts, the PWM counter overflow interrupt, and the capture A interrupt belong to this category of interrupts whose pending conditions are cleared automatically by hardware.

Pending Bits Cleared by the Service Routine

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

Pending conditions for the timer 0 match/capture interrupt, the I^2C bus interrupt, the timer A and timer B interrupts, and the V-sync interrupt must be cleared by the application's service routines.

INTERRUPT SOURCE POLLING SEQUENCE

The interrupt request polling and servicing sequence is as follows:

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

INTERRUPT SERVICE ROUTINES

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

— Interrupt processing must be enabled (EI, SYM.0 = "1")

— Interrupt level must be enabled (IMR register)

— Interrupt level must have the highest priority if more than one level is currently requesting service

— Interrupt must be enabled at the interrupt's source (peripheral control register)

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

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

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

GENERATING INTERRUPT VECTOR ADDRESSES

The interrupt vector area in the ROM contains the addresses of the interrupt service routine that corresponds to each level in the interrupt structure. Vectored interrupt processing follows this sequence:

1. Push the program counter's low-byte value to stack.
2. Push the program counter's high-byte value to stack.
3. Push the FLAGS register values to stack.
4. Fetch the service routine's high-byte address from the vector address.
5. Fetch the service routine's low-byte address from the vector address.

GENERATING INTERRUPT VECTOR ADDRESSES (CONTINUED)

6. Branch to the service routine specified by the 16-bit vector address.

NOTE

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

NESTING OF VECTORED INTERRUPTS

You can nest a higher priority interrupt request while a lower priority request is being serviced. To do this, you must follow these steps:

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

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

INSTRUCTION POINTER (IP)

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

FAST INTERRUPT PROCESSING

The feature called *fast interrupt processing* lets designated interrupts be completed in approximately six clock cycles instead of the usual 22 clock cycles. Bit 1 of the system mode register, SYM.1, enables fast interrupt processing while SYM.2–SYM.4 are used to select a specific level for fast processing.

Two other system registers support fast interrupts:

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

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

NOTE

For the KS88C3208/3216 microcontroller, the service routine for any one of the seven interrupt levels (IRQ0–IRQ4, IRQ6, or IRQ7) can be designated as a fast interrupt.

Procedure for Initiating Fast Interrupts

To initiate fast interrupt processing, follow these steps:

1. Load the start address of the service routine into the instruction pointer.
2. Load the level number into the fast interrupt select field.
3. Write a "1" to the fast interrupt enable bit in the SYM register.

Fast Interrupt Service Routine

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

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

Programming Guidelines

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

Also, if you use fast interrupts, remember to load the IP with a new start address when the fast interrupt service routine ends. (Please refer to the programming tip on page 4–8 for an example.)

☞ PROGRAMMING TIP — Programming Level IRQ0 as a Fast Interrupt

This example shows you how to program fast interrupt processing for a select interrupt level — in this case, for the timer 0 (capture) interrupt, INT0:

```
        •
        •
        •
        LD      T0CON,#52H      ; Disable T0OVF interrupt
                                ; Enable T0 interrupt
                                ; Capture mode; trigger on rising signal edges
                                ; Select fOSC /256 as T0 clock source
        LD      P2CONL,#02H     ; Set P2.0 to capture input mode
        LDW     IPH,#T0_INT     ; IPH ← high byte of interrupt service routine
                                ; IPL ← low byte of interrupt service routine
        LD      SYM,#02H        ; Enable fast interrupt processing
                                ; Select IRQ0 for fast service
        EI                      ; Enable interrupts
        •
        •
        •

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

CLOCK CIRCUITS

OVERVIEW

The clock frequency generated by an external crystal or ceramic resonator may range from 0.5 MHz to 8 MHz. The maximum CPU clock frequency is 8 MHz. The X_{IN} and X_{OUT} pins connect the external oscillation source to the on-chip clock circuit.

A separate external L-C resonator circuit generates a clock pulse for the on-screen display (OSD) block.

SYSTEM CLOCK CIRCUIT

The system clock circuit has the following components:

— External crystal or ceramic oscillation source

— Oscillator stop and wake-up functions

— Programmable frequency divider for the CPU clock (f_{OSC} divided by 1, 2, 8, or 16)

— Clock circuit control register, CLKCON

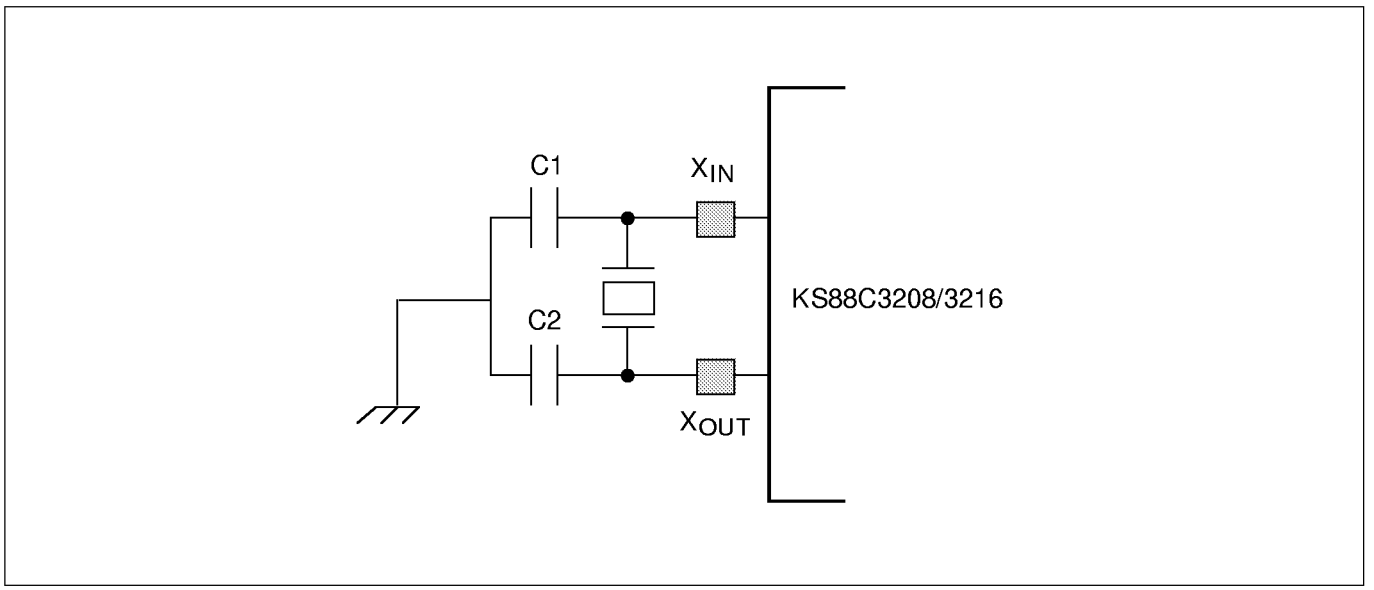


Figure 20. Main Oscillator Circuit (External Crystal or Ceramic Resonator)

CLOCK STATUS DURING POWER-DOWN MODES

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

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

— In Idle mode, the internal clock signal is gated off to the CPU and to all peripherals except for the OSD block and the A/D converter, which are inactive. Idle mode is released by a reset or by an interrupt.

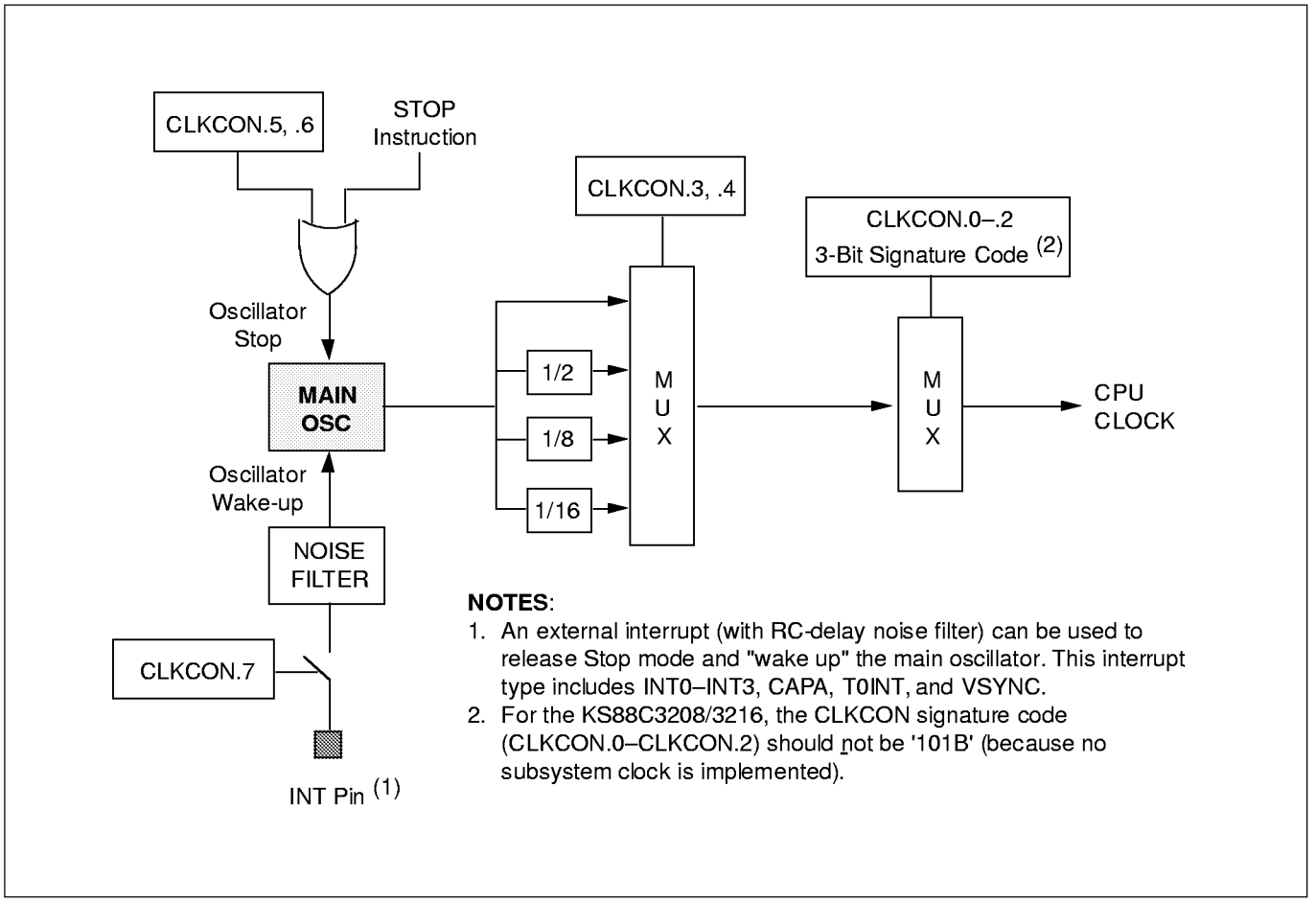


Figure 21. System Clock Circuit Diagram

SYSTEM CLOCK CONTROL REGISTER (CLKCON)

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

— Oscillator IRQ wake-up function enable/disable

— Main oscillator stop control

— Oscillator frequency divide-by value: non-divided, 2, 8, or 16

— System clock signal selection

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

After a reset, the external interrupt oscillator wake-up bit is set to "1", the main oscillator is activated, and the $f_{OSC}/16$ (the slowest clock speed) is selected as the CPU clock. If necessary, you can then increase the CPU clock speed to f_{OSC}, $f_{OSC}/2$, or $f_{OSC}/8$.

For the KS88C3208/3216, the CLKCON.0–CLKCON.2 system clock signature code should be any value *other than* '101B'. (This setting is invalid because a subsystem clock is not implemented.) The reset value for the clock signature code is '000B'.

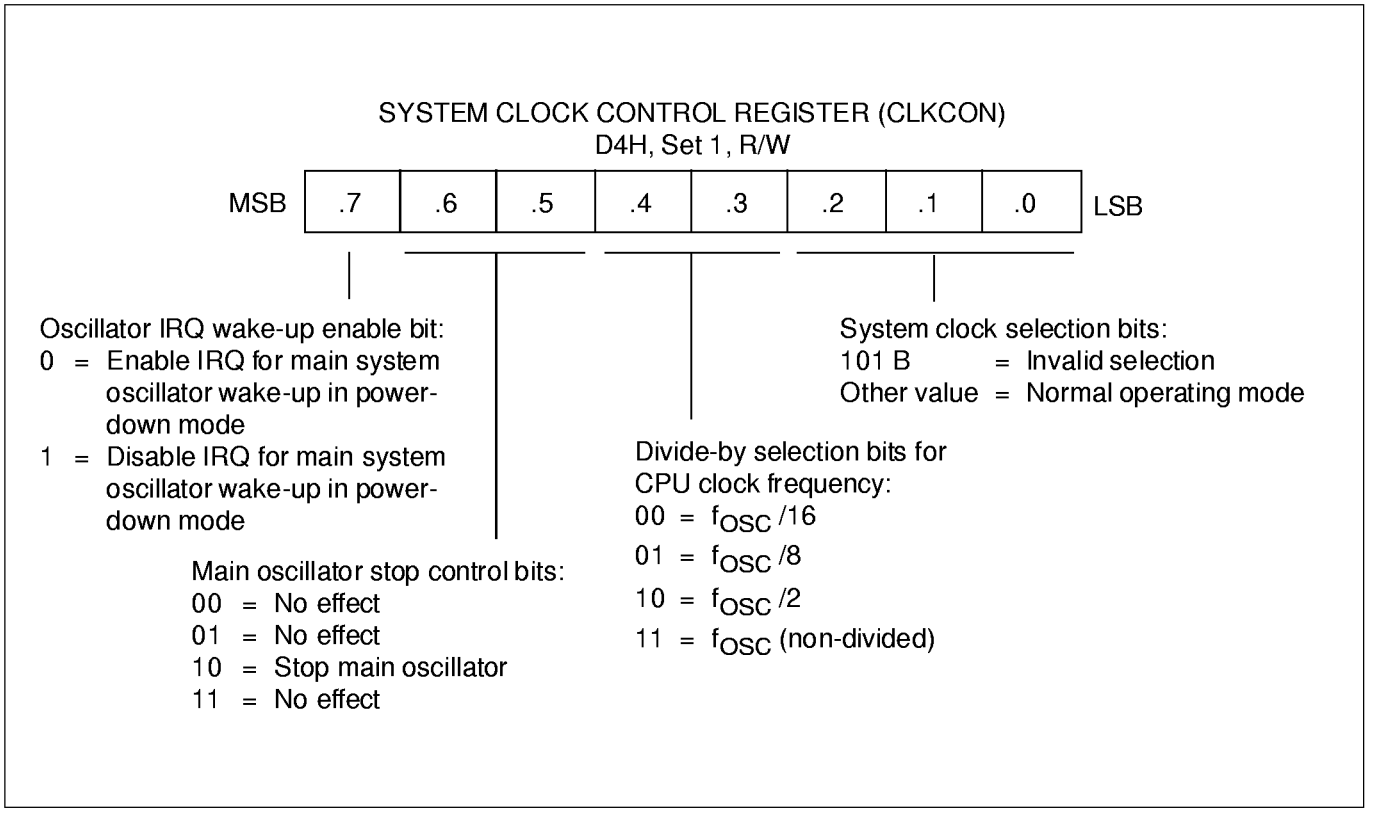


Figure 22. System Clock Control Register (CLKCON)

L-C OSCILLATOR CIRCUIT

The L-C oscillator circuit has the following components:

— External L-C oscillator with a 5–8 MHz frequency range

— Oscillator clock divider value (CHACON.4 and CHACON.5)

— OSC_{IN} and OSC_{OUT} pins

— On/off control bit (DSPCON.0)

Red-green-blue (RGB) color outputs, as well as display rates and positions, are determined by the L-C clock signal. This signal is scaled by the dot and column counter. The clock signal equals the OSD oscillator clock divided by the clock divider value. The clock divider value is determined by the horizontal character size settings in the CHACON register.

The rate at which each new display line is generated is determined strictly by the H-sync input. The rate at which each new frame (screen) is generated is determined by V-sync input. The recommended L-C clock frequency is 6.5 MHz.

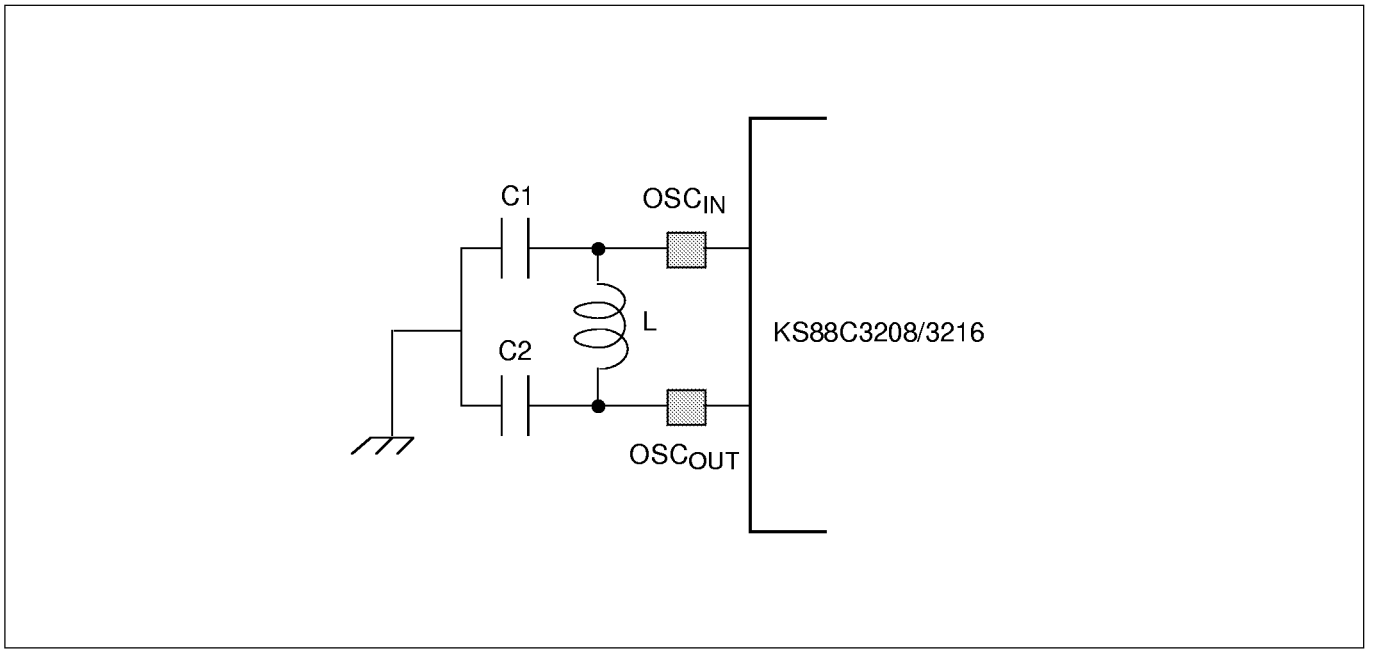


Figure 23. L-C Oscillator Circuit for OSD

SYSTEM RESET

OVERVIEW

During a power-on reset, the voltage at V_{DD} is High level and the $\overline{RESET}$ pin is forced to Low level. The $\overline{RESET}$ signal is input through a Schmitt trigger circuit where it is then synchronized with the CPU clock. This brings the KS88C3208/3216 into a known operating status.

The $\overline{RESET}$ 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 1millisecond.

When a reset occurs during normal operation (that is, when V_{DD} and $\overline{RESET}$ are High level), the $\overline{RESET}$ pin is forced Low and the reset operation starts. All system and peripheral control registers are set to their default hardware reset values (see Table 8–1). In summary, the following sequence of events occurs during a reset operation:

— All interrupts are disabled.

— The watchdog function (basic timer) is enabled.

— Ports 0, 2, and 3 are set to n-channel, open-drain output mode; port 2 pull-up resistors are disabled.

— Port 1 is set to input mode (including multiplexed inputs for H-sync, Y-sync, and capture A).

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

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

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

NOTE

You can program the duration of the oscillation stabilization interval by making the appropriate settings to the basic timer control register, BTCON, before entering Stop mode.

Also, you if you do not want to use the basic timer watchdog function (which causes a system reset if a basic timer counter overflow occurs), you can disable it by writing '1010B' to the upper nibble of BTCON.

HARDWARE RESET VALUES

Tables 8–1 through 8–4 list the reset values for CPU and system ©registers, peripheral control registers, and peripheral data registers following a reset operation. The following notation is used to represent reset values:

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

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

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

Table 8. Set 1 Register Values After a Reset

Register Name	Mnemonic	Address		Bit Values After a Reset							
		Dec	Hex	7	6	5	4	3	2	1	0
Timer 0 counter	T0CNT	208	D0H	0	0	0	0	0	0	0	0
Timer 0 data register	T0DATA	209	D1H	1	1	1	1	1	1	1	1
Timer 0 control register	T0CON	210	D2H	0	0	0	0	0	0	0	0
Basic timer control register	BTCON	211	D3H	0	0	0	0	0	0	0	0
Clock control register	CLKCON	212	D4H	0	0	0	0	0	0	0	0
System flags register	FLAGS	213	D5H	x	x	x	x	x	x	0	0
Register pointer 0	RP0	214	D6H	1	1	0	0	0	–	–	–
Register pointer 1	RP1	215	D7H	1	1	0	0	1	–	–	–
Stack pointer (high byte)	SPH	216	D8H	x	x	x	x	x	x	x	x
Stack pointer (low byte)	SPL	217	D9H	x	x	x	x	x	x	x	x
Instruction pointer (high byte)	IPH	218	DAH	x	x	x	x	x	x	x	x
Instruction pointer (low byte)	IPL	219	DBH	x	x	x	x	x	x	x	x
Interrupt request register	IRQ	220	DCH	0	0	–	0	0	0	0	0
Interrupt mask register	IMR	221	DDH	x	x	–	x	x	x	x	x
System mode register	SYM	222	DEH	0	–	0	x	x	x	0	0
Register page pointer	PP	223	DFH	0	0	0	0	0	0	0	0

NOTE: Although it is not used for the KS88C3208/3216, bit 5 of the SYM register should always be "0". If this bit is accidentally written to "1" by software, a system malfunction may occur.

Table 9. Set 1, Bank 0 Register Values After a Reset

Register Name	Mnemonic	Address		Bit Values After a Reset							
		Dec	Hex	7	6	5	4	3	2	1	0
Port 0 data register	P0	224	E0H	1	1	1	1	1	1	1	1
Port 1 data register	P1	225	E1H	0	0	0	0	0	0	0	0
Port 2 data register	P2	226	E2H	1	1	1	1	1	1	1	1
Port 3 data register	P3	227	E3H	–	–	1	1	1	1	1	1
Port 0 control register (high byte)	P0CONH	228	E4H	0	0	0	0	0	0	0	0
Port 0 control register (low byte)	P0CONL	229	E5H	0	0	0	0	0	0	0	0
Port 1 control register (high byte)	P1CONH	230	E6H	0	0	0	0	0	0	0	0
Port 1 control register (low byte)	P1CONL	231	E7H	0	0	0	0	0	0	0	0
Port 2 control register (high byte)	P2CONH	232	E8H	0	0	0	0	0	0	0	0
Port 2 control register (low byte)	P2CONL	233	E9H	0	0	0	0	0	0	0	0
Port 3 control register (high byte)	P3CONH	234	EAH	–	–	–	–	0	0	0	0
Port 3 control register (low byte)	P3CONL	235	EBH	0	0	0	0	0	0	0	0
Port 2 pull-up resistor enable register	P2PUR	236	ECH	–	0	0	0	0	0	0	0
I^2C-bus data shift register	IIC	237	EDH	x	x	x	x	x	x	x	x
I^2C-bus control register	IICCON	238	EEH	0	0	0	0	0	0	0	0
I^2C-bus prescaler	IICPS	239	EFH	0	0	0	0	0	0	0	0
Timer A data register	TADATA	240	F0H	1	1	1	1	1	1	1	1
Timer B data register	TBDATA	241	F1H	1	1	1	1	1	1	1	1
Timer A control register	TACON	242	F2H	0	0	0	0	0	0	0	0
Timer B control register	TBCON	243	F3H	0	0	0	0	0	0	0	0
PWM0 data register (main byte)	PWM0	244	F4H	1	1	1	1	1	1	1	1
PWM0 data register (extension byte)	PWM0EX	245	F5H	0	0	0	0	0	0	–	–
PWM1 data register (main byte)	PWM1	246	F6H	1	1	1	1	1	1	1	1
PWM1 data register (extension byte)	PWMEX	247	F7H	0	0	0	0	0	0	–	–
PWM control register	PWMCON	248	F8H	0	0	0	0	0	–	0	0
Capture A data register	CAPA	249	F9H	0	0	0	0	0	0	0	0
A/D converter control register	ADCON	250	FAH	0	–	0	0	0	0	0	0
Locations FBH and FCH in set 1, bank 0, are not mapped.											
Basic timer counter	BTCNT	253	FDH	0	0	0	0	0	0	0	0
External memory timing register	EMT	254	FEH	0	1	1	1	1	1	0	–
Interrupt priority register	IPR	255	FFH	x	x	–	x	x	x	x	x

Table 10. Set 1, Bank 1 Register Values After a Reset

Register Name	Mnemonic	Address		Bit Values After a Reset							
		Dec	Hex	7	6	5	4	3	2	1	0
Locations E0H–EFH in set 1, bank 1, are not mapped.											
OSD character size control register	CHACON	240	F0H	0	0	0	0	0	0	0	0
OSD fade control register	FADECON	241	F1H	0	0	0	0	0	0	0	0
OSD row position control register	ROWCON	242	F2H	0	0	0	0	0	0	0	0
OSD column position control register	CLMCON	243	F3H	0	0	0	0	0	0	0	0
OSD background color control register	COLCON	244	F4H	0	0	0	0	0	0	0	0
On-screen display control register	DSPCON	245	F5H	0	0	0	0	0	0	0	0
Halftone signal control register	HTCON	246	F6H	0	0	0	0	0	0	0	0
OSD color buffer	COLBUF	247	F7H	–	–	–	–	–	x	x	x
PWM2 data register	PWM2	248	F8H	x	x	x	x	x	x	x	x
PWM3 data register	PWM3	249	F9H	x	x	x	x	x	x	x	x
PWM4 data register	PWM4	250	FAH	x	x	x	x	x	x	x	x
PWM5 data register	PWM5	251	FBH	x	x	x	x	x	x	x	x
Locations FCH–FFH in set 1, bank 1, are not mapped.											

Table 11. Page 1 Video RAM Register Values After a Reset

Register Name	Address	Bit Values After a Reset							
		7	6	5	4	3	2	1	0
OSD video RAM	00H–EFH, page 1	x	x	x	x	x	x	x	x

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 5μA. All system functions stop when the clock "freezes," but data stored in the internal register file is retained. Stop mode can be released in one of two ways: by a RESET signal or by an external interrupt.

Using RESET to Release Stop Mode

Stop mode is released when the RESET signal goes inactive (High level). All system and peripheral control registers are 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 programmed oscillation stabilization interval has elapsed, the CPU starts the system initialization routine by fetching the address stored in ROM location 0100H.

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. The external interrupts in the KS88C3208/3216 interrupt structure that meet this requirement are INT0–INT3 (P0.4–P0.7), CAPA, T0, and V-sync. Which interrupt you can use to release Stop mode in a given situation depends on the microcontroller's current internal operating mode.

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 and to all peripherals except the OSD block and the A/D converter. 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. When the return-from-interrupt (IRET) occurs, the instruction immediately following the one that initiated Idle mode is executed.

NOTE

Only external interrupts can be used to release Stop mode. To release Idle mode, you can use either type of interrupt (internal or external).

☞ PROGRAMMING TIP — Initial Settings for Address Space, Vectors, and Peripherals

The following sample program shows you recommended initial settings for the KS88C3208/3216 address space, interrupt vectors, and peripheral functions. Program comments guide you through the required steps:

```
            •
            •
            •
OSD_REG     EQU     0C8H            ;  OSD working register area
osd_flg     equ     8
dsp_typ     equ     9
VRAMAD      EQU     0CH
WORK1       EQU     0BH             ;  General-purpose area
WORK2       EQU     0AH             ;  General-purpose area
REMOCON     EQU     3FH             ;  CAPA data save register
            •
            •
            •
            ORG     00H
            DW      PWM_OVF         ;  PWM counter overflow vector
            DW      CAPA_INT        ;  Capture A interrupt

            ORG     0BCH
            DW      TIMERB_INT      ;  Timer B interrupt
            DW      TIMERA_INT      ;  Timer A interrupt

            ORG     0C0H
            DW      P04_INT         ;  P0.4 external interrupt
            DW      P05_INT         ;  P0.5 external interrupt
            DW      P06_INT         ;  P0.6 external interrupt
            DW      P07_INT         ;  P0.7 external interrupt

            ORG     0D0H
            DW      IIC_INT         ;  IIC-bus interrupt

            ORG     0D4H
            DW      V_SYNC_INT      ;  V-sync interrupt

            ORG     0FAH
            DW      TIMER0_OVF      ;  Timer 0 overflow interrupt
            DW      TIMER0_INT      ;  Timer 0 interrupt
            ORG     0100H

START       DI                      ;  Disable all interrupts
            LD      BTCON,#0AAH     ;  Disable the watchdog timer
            LD      CLKCON,#98H     ;  Non-divided clock
            CLR     SYM             ;  Disable global and fast interrupts
            CLR     SPL             ;  Stack pointer low byte ← "0"
                                    ;  Stack area will start at 0FFH
            SB1                     ;  Select bank 1
```

(Continued on next page)

☞ **PROGRAMMING TIP — Initial Settings for Address Space, Vectors, and Peripherals (Continued)**

```
        LD      HTCON,#0AH          ; Enable V-sync interrupt
        LD      DSPCON,#0A0H        ; Disable OSD logic
        SB0                         ; Select bank 0
        LD      PWMCON,#0E9H        ; Prescaler ← 4
                                    ; Enable PWM counter
                                    ; Disable PWM overflow interrupt
                                    ; Enable capture A interrupt
        LD      IPR,#0AEH           ; Interrupt priority settings
                                    ; IRQ6 > 7 > 4 > 2 > 0 > 1 > 5
        LD      IMR,#0C8H           ; Enable level 3, 6, and 7 interrupts
        LD      P0CONH,#00H         ; Output mode
        LD      P0CONL,#0FFH        ; ADC input mode
        LD      P1CONH,#0FFH        ; Output mode
        LD      P1CONL,#00H         ; Input mode
        LD      P2CONH,#00H         ; Open-drain output mode
        LD      P2CONL,#00H         ; Open-drain output mode
        LD      P2PUR,#02H          ; Enable pull-up resistor at P2.1
        LD      P3CONH,#00H         ; Open-drain output mode
        LD      P3CONL,#00H         ; Open-drain output mode

        LD      TACON,#54H          ; Prescaler ← 6
                                    ; Clock source ← CPU clock / 1000
                                    ; Enable timer A interrupt
                                    ; Interval timer mode
        LD      TADATA,#03H         ; 4-millisecond interrupt

        EI

MAIN    NOP
        NOP
        •
        •
        •
        NOP
        JP      T,MAIN              ; Jump MAIN

CAPA_INT:                           ; CAPA interrupt service
        PUSH    PP                  ; Save page pointer to stack
        PUSH    RP0                 ; Save register pointer 0 to stack
        PUSH    RP1                 ; Save register pointer 1 to stack
        •
        •
        •
        LD      REMOCON,CAPA        ; REMOCON ← CAPA data
        POP     RP1                 ; Restore register pointer 1 value
        POP     RP0                 ; Restore register pointer 0 value
        POP     PP                  ; Restore page pointer value
        IRET                        ; Return from interrupt service routine
```

(Continued on next page)

☞ **PROGRAMMING TIP — Initial Settings for Address Space, Vectors, and Peripherals (Continued)**

```
TIMERA_INT  PUSH    PP              ;  TIMER_A interrupt service
            PUSH    RP0
            PUSH    RP1
            •
            •
            •
            LD      TACON,#54H      ;  Clear pending bit
            POP     RP1
            POP     RP0
            POP     PP
            IRET                    ;  Return from interrupt service routine

V_SYNC_INT  PUSH    PP              ;  V_SYNC interrupt service
            PUSH    RP0
            PUSH    RP1
            •
            •
            •
            LD      TACON,#54H      ;  Clear pending bit
            POP     RP1
            POP     RP0
            POP     PP
            IRET                    ;  Return from interrupt service routine

PWM_OVF:                            ;  PWM counter overflow interrupt

TIMERB_INT:                         ;  Timer B interrupt

P04_INT:                            ;  P0.4 external interrupt

P05_INT:                            ;  P0.5 external interrupt

P06_INT:                            ;  P0.6 external interrupt

P07_INT:                            ;  P0.7 external interrupt

IIC_INT:                            ;  IIC-bus interrupt

TIMER0_INT:                         ;  Timer 0 interrupt

TIMER0_OVF:                         ;  Timer 0 overflow interrupt

            IRET                    ;  Return from interrupt service routine
```

I/O PORTS

OVERVIEW

The KS88C3208/3216 microcontroller have four I/O ports with a total of 30 pins. Up to 22 pins can be configured as n-channel, open-drain outputs. Of these 22 open-drain pins, 11 pins can withstand loads of up to 10volts and 11 pins can withstand loads of up to 5 volts.

The CPU accesses ports by directly writing or reading port registers. No special I/O instructions are required. Table 9–1 gives you a summary of port functions:

Table 12. KS88C3208/3216 Port Configuration Overview

Port	Configuration Options	Programmability
0	General I/O port that is configurable for digital input or n-channel, open-drain output. The lower nibble pins can withstand up to 5 V and the upper nibble pins up to 10 V. Lower nibble pins are multiplexed for alternative use as A/D converter inputs; upper nibble pins are multiplexed for alternative use as external interrupt inputs.	Bit programmable
1	General I/O port. Input or output mode is software configurable. Pull-up resistors can be assigned to individual pins. Pins 1.3–P1.7 are multiplexed to support alternative functions.	Bit programmable
2	General I/O port. Input/output mode or n-channel, open-drain output mode is software configurable. P2.0–P2.6 can withstand up to 5 V; P2.7 can withstand 10 V. Pull-up resistors can be assigned to P2.0–P2.6 (but not to P2.7) using the P2PUR register. Each port 2 pin has an alternative function.	Bit programmable
3	General 6-bit I/O port, configurable for digital input or n-channel, open-drain output. Pins can withstand up to 10V P3.0–P3.3 alternate as outputs for the 8-bit PWM circuits; P3.4 and P3.5 alternate as output pins for timer A and timer B, respectively.	Bit programmable

PORT DATA REGISTERS

Data registers for ports 0–3 have the structure shown in Figure 25. Table 13 gives you an overview of the port data register locations:

Table 13. Port Data Register Summary

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

NOTE

After a reset , the port 1 data register is cleared to '00H'. The other data registers are, however, set to 'FFH' to close n-channel, open-drain outputs. See Section 8, "RESET and Power-Down," for details.

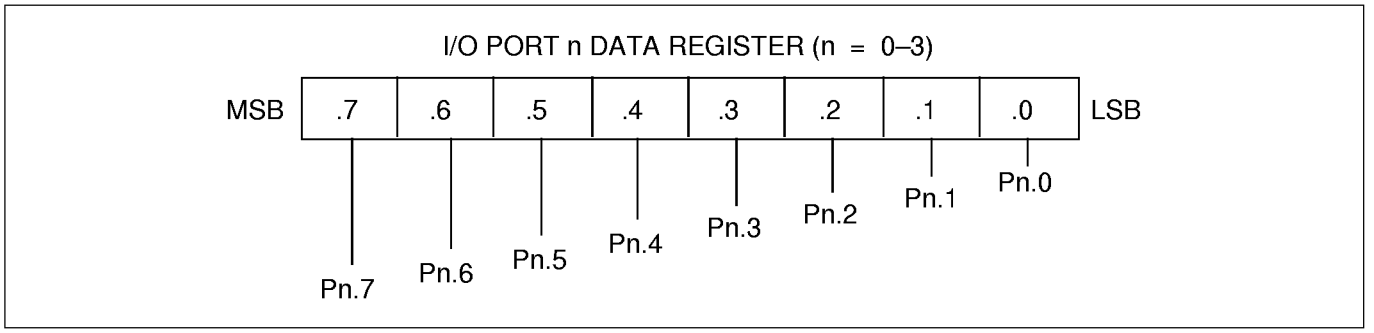


Figure 24. Port Data Register Format

PORT 0

Port 0 is a bit-programmable general I/O port. The lower byte (P0.0–P0.3) and the upper byte (P0.4–P0.7) are assigned different pin circuit types (see Section 1, "Product Overview"). Port 0 is accessed directly by writing or reading the port0 data register, P0 (E0H, set 1, bank 0).

A reset operation sets the P0 data register to 'FFH'. RESET also clears P0CONH and P0CONL to '00H', configuring the port 0 pins to n-channel, open-drain output mode. The pin circuit assigned to the lower-byte pins can withstand 5-V loads; the upper-byte pins can withstand 10-V loads.

Each pin has an individually configurable input function: P0.0–P0.3 are multiplexed to serve as inputs for the A/D converter (ADC0 through ADC3, respectively). Whenever one of the lower-byte port 3 pins is set to ADC input mode, digital input at that pin is disabled.

You can alternatively use P0.4–P0.7 for external interrupt inputs INT0–INT3, respectively. Interrupts can be enabled or disabled selectively, and they can be set to trigger the corresponding interrupt request on either rising or falling signal edges.

The port 0 pins are configured by bit-pair settings in the P0CONH and P0CONL registers. P0CONH controls I/O for the upper byte pins and the alternative external interrupt function. P0CONL controls I/O for the lower byte pins and the alternative A/D converter input function.

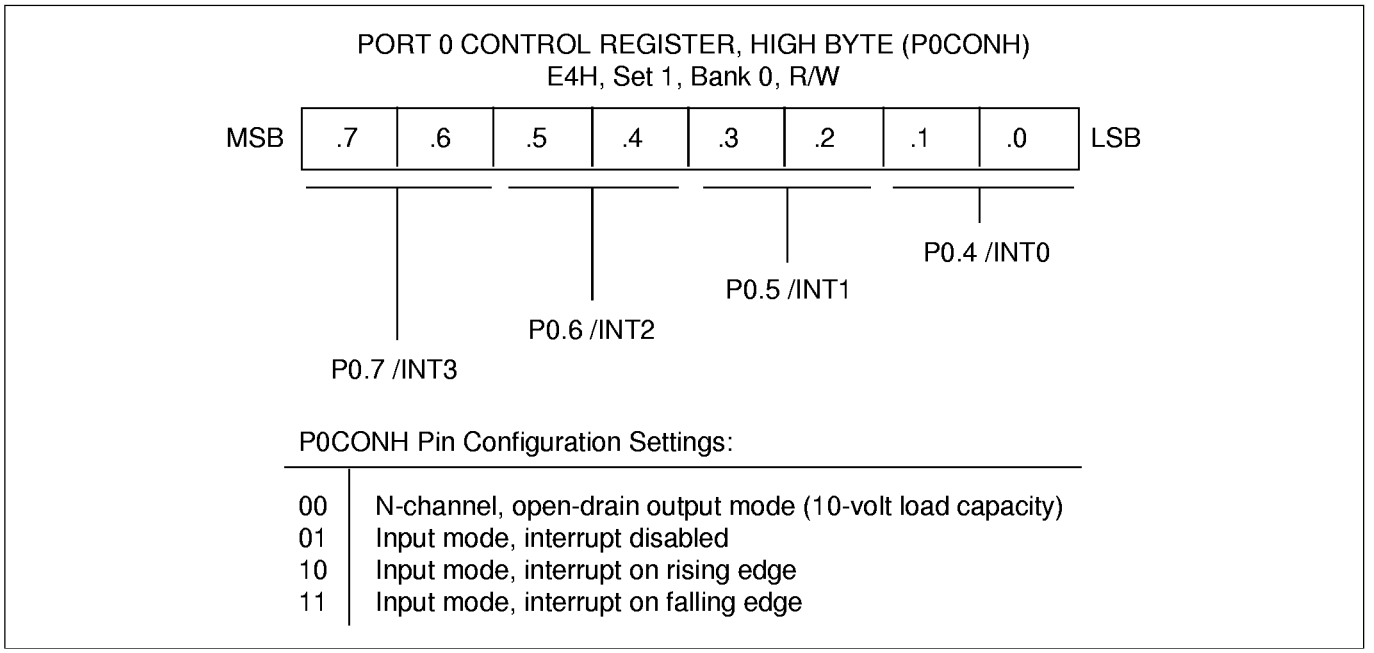


Figure 25. Port 0 High-Byte Control Register (P0CONH)

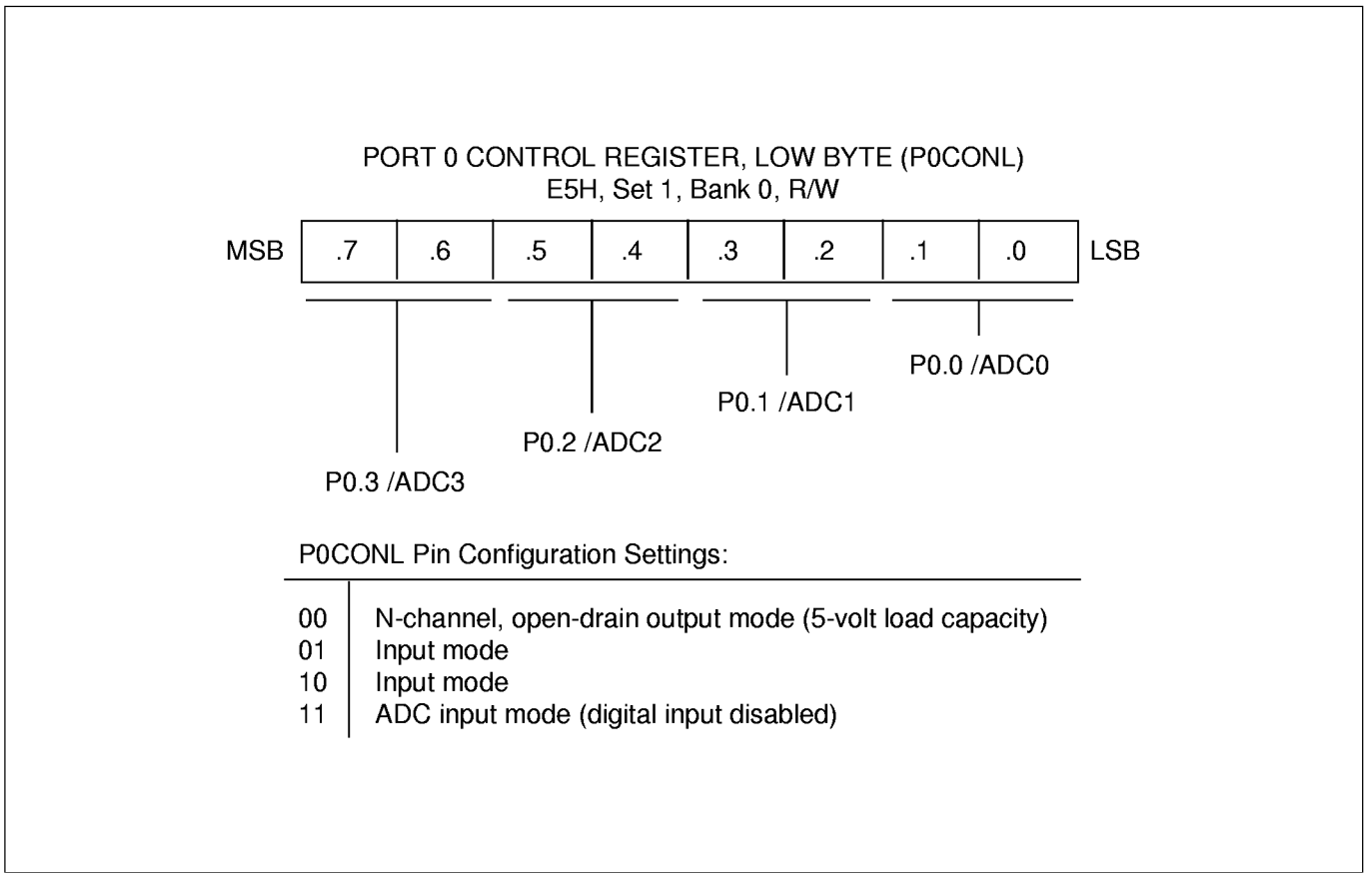


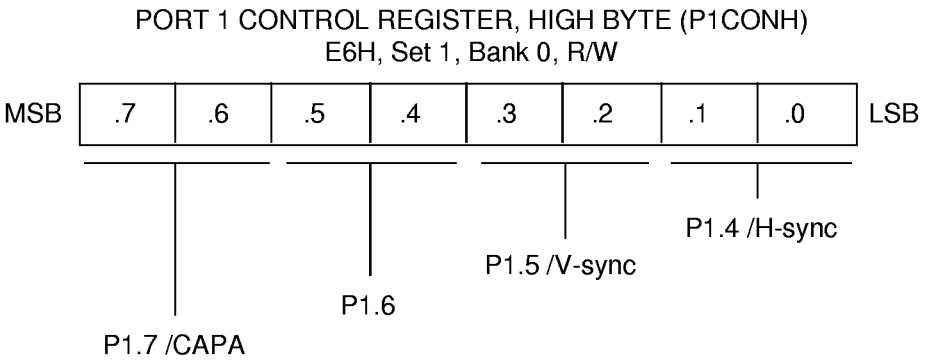
Figure 26. Port 0 Low-Byte Control Register, (P0CONL)

PORT 1

Port 1 is a bit-programmable general I/O port. Port 1 is accessed directly by writing or reading the port 1 data register, P1 (E1H, set 1, bank 0). The upper byte (P1.4–P1.7) and the lower byte (P1.0–P1.3) are controlled by the P1CONH and P1CONL registers, respectively. P1CONH is located at E6H in set 1, bank 0 and P1CONL is located at E7H in set 1, bank 0.

A reset clears each control register to '00H', configuring the port 1 pins to input mode: to multiplexed input mode if an alternative function is assigned, or otherwise to normal input mode. You can also assign pull-up resistors to individual port 1 input pins, if required. Reset also clears the P1 data register value to '00H.'

Each port 1 pin has two output modes: push-pull or normal/multiplexed output. Alternative input functions for the upper byte pins are H-sync and V-sync input for the OSD module (P1.4 and P1.5, respectively) and capture A (CAPA) input for P1.7. The single alternative output function for the lower byte is the OSD half-tone output (OSDHT) at P1.3.



P1CONH Pin Configuration Settings:

00	Multiplexed input mode for P1.7, P1.5, and P1.4; normal input mode for P1.6
01	Multiplexed input mode for P1.7, P1.5, and P1.4; normal input mode for P1.6; pull-up resistor enabled
10	Push-pull output mode
11	Output mode (push-pull circuit type)

Figure 27. Port 1 High-Byte Control Register (P1CONH)

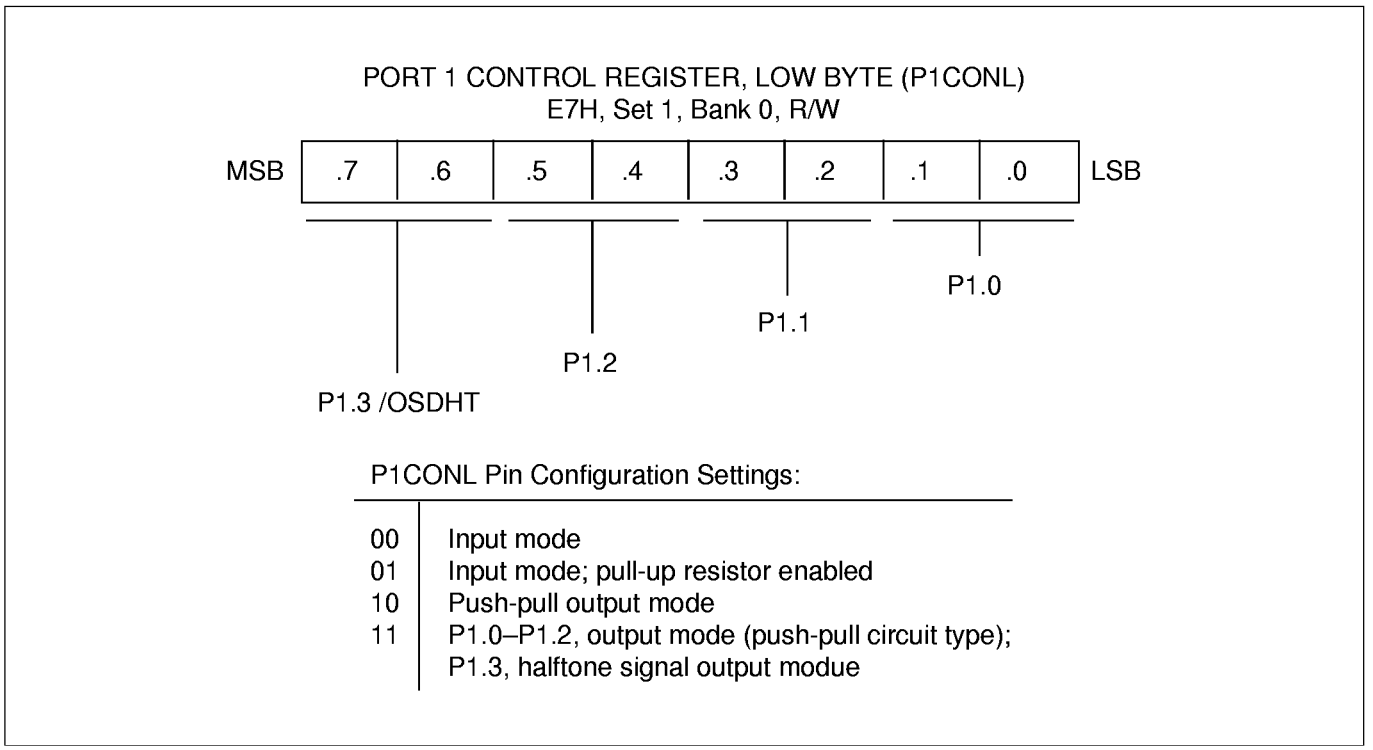


Figure 28. Port 1 Low-Byte Control Register (P1CONL)

PORT 2

Port 2 is a bit-programmable general I/O port. Port 2 is accessed directly by writing or reading the port 2 data register, P2 (E2H, set 1, bank 0). The upper byte (P2.4–P2.7) and the lower byte (P2.0–P2.3) are controlled by the P2CONH and P2CONL registers, respectively.

A reset clears the port 2 control registers to '00H', configuring the port 2 pins to n-channel, open-drain output mode. It also sets the P2 data register value to 'FFH', turning off the 'n' channel. P2.0–P2.6 are assigned the same circuit type which can withstand loads up to 5 V. A different circuit type is assigned to P2.7 which can withstand up to 10-volt loads.

You use P2CONH and P2CONL register settings to configure individual port 2 pins:

— The '01B' setting configures P2.0–P2.6 to push-pull output mode. This setting does not assign a push-pull circuit to P2.7, which remains configured to n-channel, open-drain output mode.

— The '10B' setting configures a port 2 pin to input mode. It sets the upper byte pins P2.4–P2.7 to normal input mode. The lower byte pins P2.2 and P2.3 are also set to normal input mode. P2.0 is, however, set to timer 0 capture mode (T0) and P2.1 to timer 0 clock input mode (T0CK).

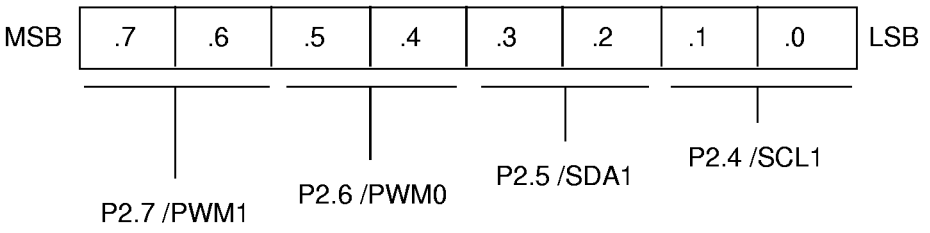
— The '11B' setting configure the alternative port 2 output function (timer 0 in interval or PWM mode, I^2C-bus, PWM0, and PWM1). An exception is P2.1, for which '11B' is an invalid setting.

NOTE

P2.2/P2.3 and P2.4/P2.5 are pin pairs for the I^2C-bus interface (SCL0/SDA0 and SCL1/SDA1, respectively). You can only select one pair at a time.

Using the pull-up resistor enable register, P2PUR, you can assign pull-ups to P2.0–P2.6 (but not to P2.7).

PORT 2 CONTROL REGISTER, HIGH BYTE (P2CONH)
E8H, Set 1, Bank 0, R/W



P2CONH Pin Configuration Settings:

00	N-channel, open-drain output mode; P2.7 with 10-volt load capacity; P2.4–P2.6 with 5-volt load capacity
01	P2.7: N-channel, open-drain output mode P2.4–P2.6: push-pull output mode
10	Input mode (normal or multiplexed)
11	P2.7: PWM1 output mode (n-channel, open-drain type) P2.6: PWM0 output mode (push-pull circuit type) P2.5: IIC-bus data I/O (SDA1) P2.4: IIC-bus clock output (SCL1)

Figure 29. Port 2 High-Byte Control Register (P2CONH)

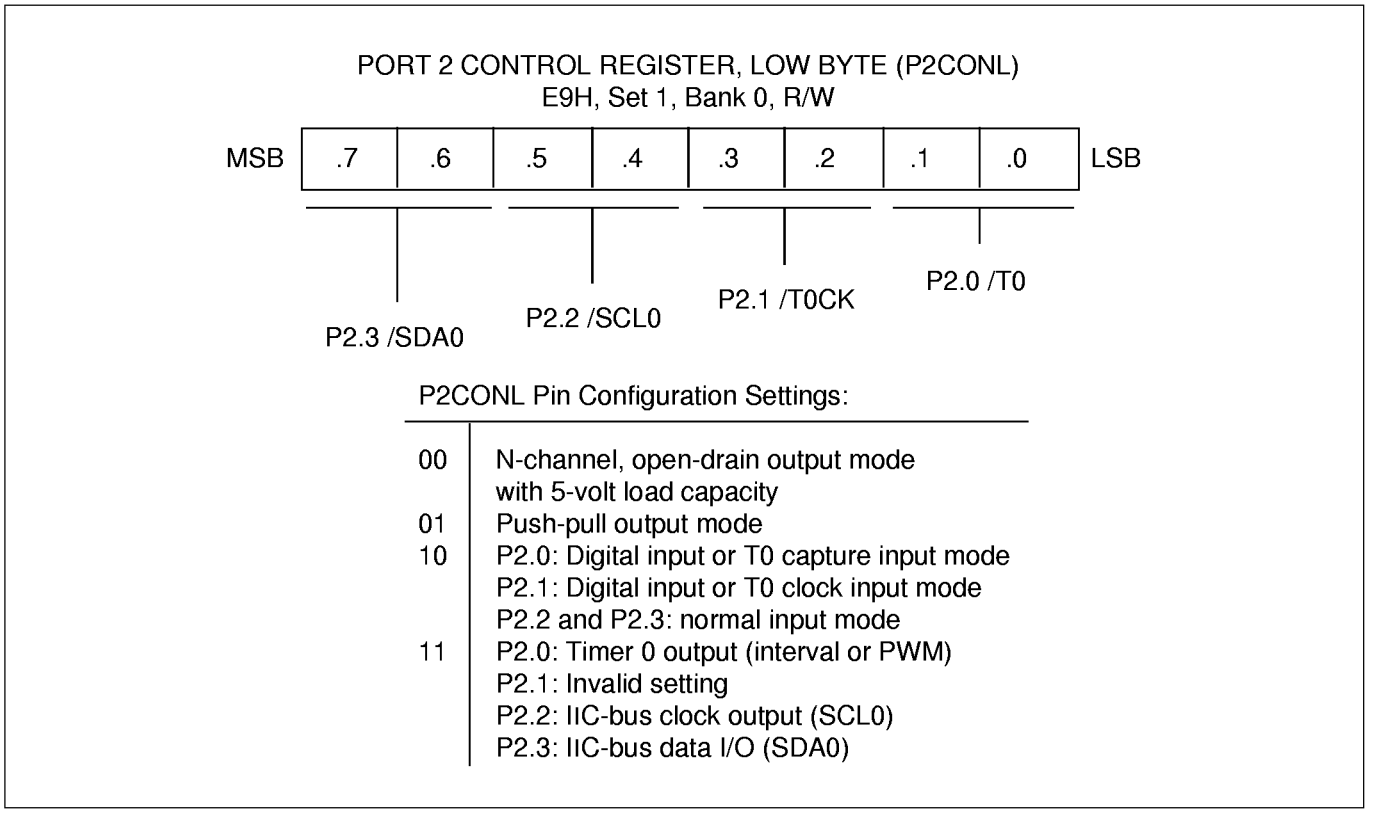


Figure 30. Port 2 Low-Byte Control Register (P2CONL)

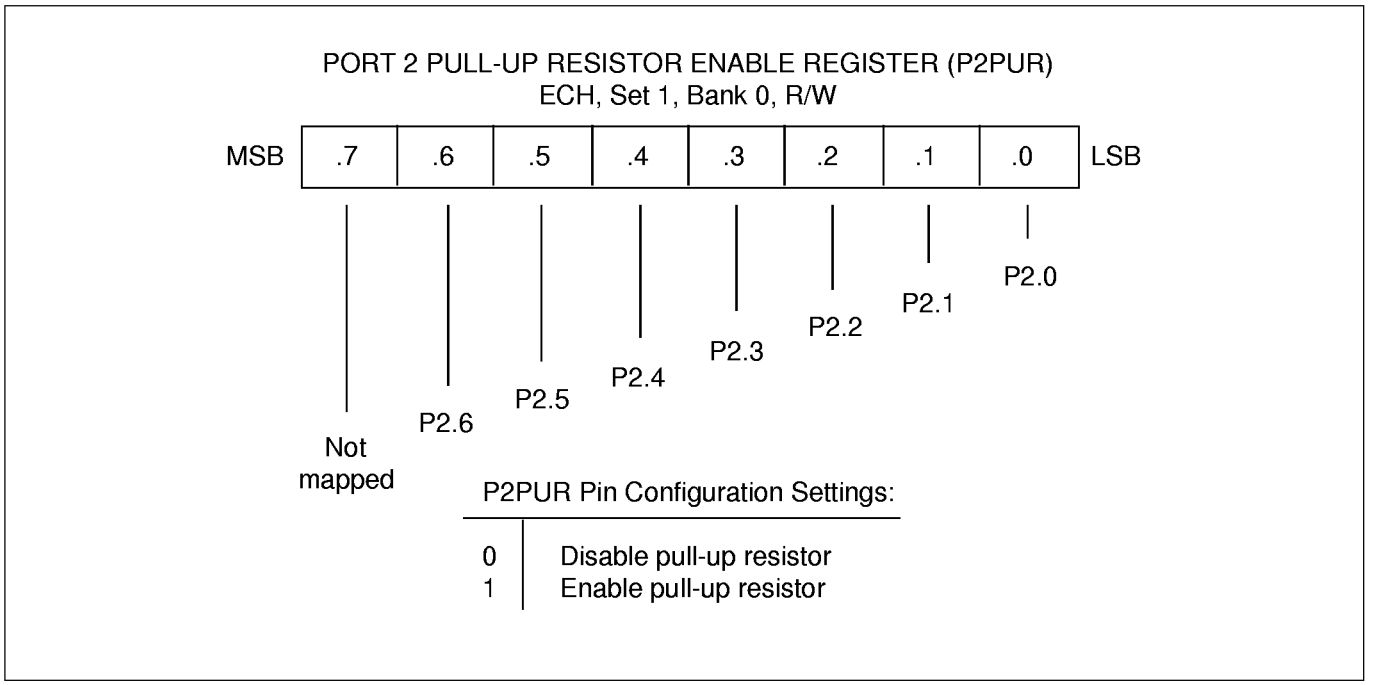


Figure 31. Port 2 Pull-Up Resistor Enable Register (P2PUR)

PORT 3

Port 3 is a bit-programmable general I/O port. Only six bits are used. Port 3 is accessed directly by writing or reading the port3 data register, P3 (E3H, set 1, bank 0).

A reset operation sets the P3 data register to 'FFH', turning off the n-channels. RESET also clears P3CONH and P3CONL to '00H', configuring the port 3 pins to n-channel, open-drain output mode. The pin circuit assigned to the upper-byte and lower-byte pins can withstand 10-volt loads.

The port 3 pins are configured by bit-pair settings in the P3CONH and P3CONL registers. P3CONH controls I/O for the two upper-byte pins, P3.4–P3.5, and the alternative TA and TB output functions. P3CONL controls I/O for the four lower-byte pins, P3.0–P3.3, and the alternative PWM2–PWM5 output functions.

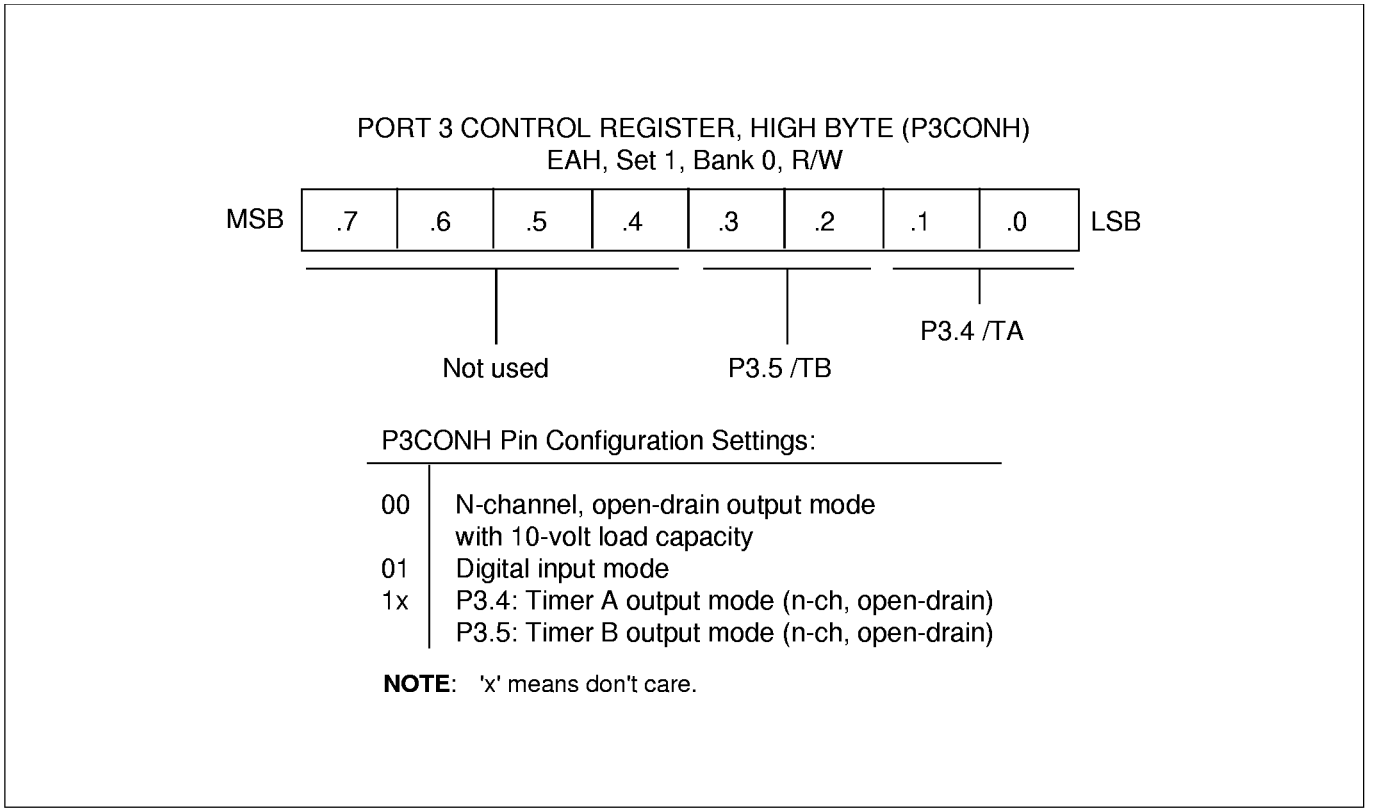
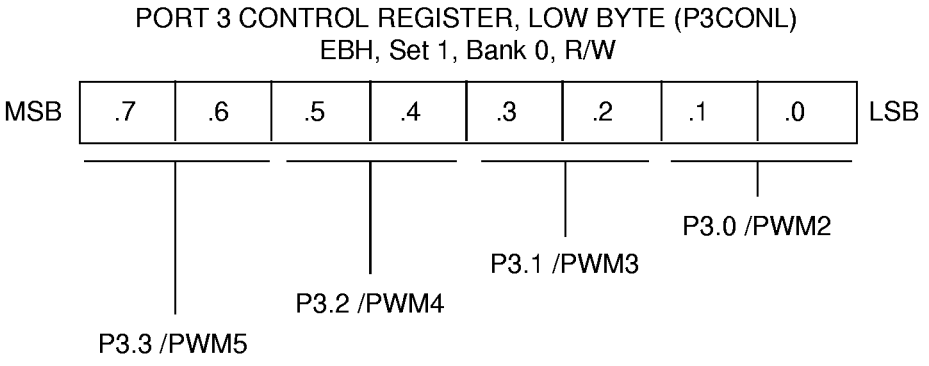


Figure 32. Port 3 High-Byte Control Register (P3CONH)



P3CONL Pin Configuration Settings:

00	N-channel, open-drain output mode with 10-volt load capacity
01	Digital input mode
1x	PWMn output mode, where n = 2–5 (n-channel, open-drain circuit type)

NOTE: 'x' means don't care.

Figure 33. Port 3 Low-Byte Control Register (P3CONL)

☞ PROGRAMMING TIP — Configuring I/O Port Pins to Specification

The following sample program shows you how to configure the KS88C3208/3216 I/O ports to specification. The following parameters are given for ports 0, 1, 2, and 3:

— Set P0.0 and P0.1 to open-drain output mode
— Set P0.2 and P0.3 to ADC input mode
— Set P0.4 and P0.5 to input mode with rising edge interrupts
— Set P0.6 and P0.7 to input mode with falling edge interrupts
— Set P1.0–P1.3 to push-pull output mode
— Set P1.4 to H-sync input mode
— Set P1.5 to V-sync input mode
— Set P1.6 to push-pull output mode
— Set P1.7 to CAPA input mode
— Set P2.0 and P2.1 to open-drain output mode
— Set P2.2–P2.5 to IIC-bus mode
— Set P2.6 to open-drain output mode
— Set P2.7 to PWM output mode
— Set P3.0–P3.5 to open-drain output mode

```
•
•
•
SB0                             ;  Select bank 0

LD      P0CONH,#0FAH            ;  P0.4, P0.5 ← Input mode; rising edge interrupts
                                ;  P0.6, P0.7 ← Input mode; falling edge interrupts
LD      P0CONL,#0F0H            ;  P0.0, P0.1 ← Open-drain output mode
                                ;  P0.2, P0.3 ← ADC input mode

LD      P1CONH,#20H             ;  P1.4 ← H-sync input mode
                                ;  P1.5 ← V-sync input mode
                                ;  P1.6 ← Push-pull output mode
                                ;  P1.7 ← CAPA input mode
LD      P1CONL,#0AAH            ;  P1.0–P1.3 ← Push-pull output mode

LD      P2CONH,#0CFH            ;  Configure P2.4 and P2.5 as IIC-bus lines
                                ;  P2.6 ← Open-drain output mode
                                ;  P2.7 ← PWM output mode
LD      P2CONL,#0F0H            ;  P2.0, P2.1 ← Open-drain output mode
                                ;  Configure P2.2 and P2.3 as IIC-bus lines

LD      P3CONH,#00H             ;  P3.4, P3.5 ← Open-drain output mode
LD      P3CONL,#00H             ;  P3.0–P3.4 ← Open-drain output mode
•
•
•
```

☞ PROGRAMMING TIP — Clearing Port 0 Interrupt Pending Bits

This sample program shows you how to clear the interrupt pending bits for port 0. The program parameters are as follows:

— Enable only interrupt level 1 (IRQ1) for P0.4–P0.7

— Set the interrupt priorities as P0.4 > P0.5 > P0.6 > P0.7

```
            ORG     0C0H
            VECTOR  EXT_INT_P04
            VECTOR  EXT_INT_P05
            VECTOR  EXT_INT_P06
            VECTOR  EXT_INT_P07
            •
            •
            •
            ORG     0100H
RESET       DI                          ;  Disable all interrupts
            SB0                         ;  Select bank 0
            LD      BTCON,#0AAH         ;  Disable the watchdog timer
            LD      CLKCON,#98H         ;  Non-divided clock
            CLR     SPL                 ;  Stack pointer low byte ← "0"
                                        ;  Stack area starts at 0FFH
            •
            •
            •
            LD      IMR,#06H            ;  Enable IRQ1 and IRQ2 interrupts
            LD      IPR,#11H            ;  IRQ1 > IRQ2
            LD      P0CONH,#0FAH        ;  P0.4, P0.5 ← Input mode; rising edge interrupts
                                        ;  P0.6, P0.7 ← Input mode; falling edge interrupts
            LD      P0CONL,#0F0H        ;  P0.0, P0.1 ← Open-drain output mode
                                        ;  P0.2, P0.3 ← ADC input mode
            •
            •
            •
            SRP     #0C0H               ;  Set register pointer to 0C0H
            EI                          ;  Enable interrupts
            •
            •
            •
MAIN        NOP
            NOP
            •
            •
            •
            JP      T,MAIN
            •
            •
            •
```

(Continued on next page)

☞ **PROGRAMMING TIP — Clearing Port 0 Interrupt Pending Bits (Continued)**

```
EXT_INT_P04:                        ; P0.4 external interrupt service
            PUSH    PP              ; Save page pointer to stack
            PUSH    RP0             ; Save register pointer 0 to stack
            PUSH    RP1             ; Save register pointer 1 to stack
            •
            •
            •
            POP     RP1             ; Restore register pointer 1 value
            POP     RP0             ; Restore register pointer 0 value
            POP     PP              ; Restore page pointer value
            IRET                    ; Return from interrupt service routine

EXT_INT_P05:                        ; P0.5 external interrupt service
            PUSH    PP
            PUSH    RP0
            PUSH    RP1
            •
            •
            •
            POP     RP1
            POP     RP0
            POP     PP
            IRET

EXT_INT_P06:                        ; P0.6 external interrupt service
            PUSH    PP
            PUSH    RP0
            PUSH    RP1
            •
            •
            •
            POP     RP1
            POP     RP0
            POP     PP
            IRET

EXT_INT_P07:                        ; P0.7 external interrupt service
            PUSH    PP
            PUSH    RP0
            PUSH    RP1
            •
            •
            •
            POP     RP1
            POP     RP0
            POP     PP
            IRET
```

BASIC TIMER and TIMER 0

MODULE OVERVIEW

The KS88C3208/3216 microcontroller have two default timers: an 8-bit *basic timer (BT)* and one 8-bit general-purpose timer/counter, called *timer 0 (T0)*.

The basic timer (BT) has two alternative functions: 1) It can be used as a watchdog timer to provide an automatic reset mechanism in the event of a system malfunction, and 2) It can be used to signal the end of the required oscillation stabilization interval after a reset or a Stop mode release. The components of the basic timer are:

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

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

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

Timer 0 (T0) has three operating modes, one of which is selected by the appropriate T0CON register setting: interval timer output mode, PWM output mode, and capture input mode. Timer 0 has the following components:

— Clock frequency divider (f_{OSC} divided by 4096, 256, or 8) with multiplexer

— External clock input pin, T0CK

— 8-bit counter (T0CNT), 8-bit comparator, and 8-bit reference data register (T0DATA)

— I/O pin (P2.0/T0) for capture input or match output

— Timer 0 overflow interrupt (T0OVF) and match/capture interrupt (T0INT) generation

— Timer 0 control register, T0CON (set 1, D2H, read/write)

BASIC TIMER CONTROL REGISTER (BTCON)

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

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

The 8-bit basic timer counter, BTCNT (set 1, bank 0, FDH), can be cleared 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 (unless timer 0 is using an external clock source), you write a "1" to BTCON.0.

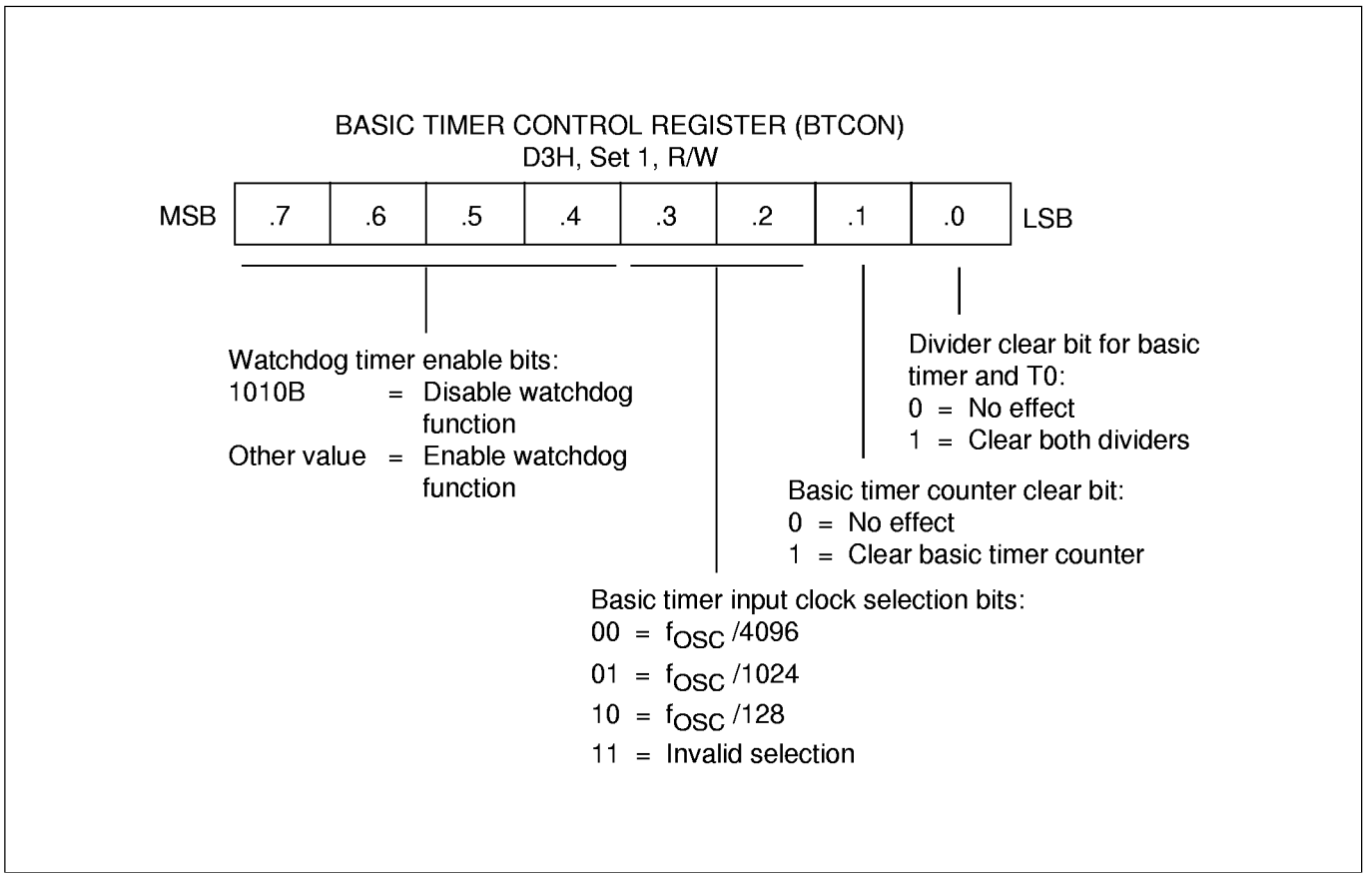


Figure 34. Basic Timer Control Register (BTCON)

BASIC TIMER FUNCTION DESCRIPTION

Watchdog Timer Function

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

With every overflow of the basic timer counter, a reset occurs. During normal operation, this overflow-generated reset should be prevented from occurring. To do this, the basic timer counter value must be cleared by software (write BTCON.1 to "1") in regular intervals.

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

An application program can use the basic timer as a watchdog timer to trigger an automatic system reset in case a malfunction occurs.

Oscillation Stabilization Interval Timer Function

The basic timer determines the oscillation stabilization interval following a reset or after the release of Stop mode by an external interrupt. Whenever a reset or an external interrupt occurs during Stop mode, the oscillator begins operating. The basic timer value then starts increasing at the rate of $f_{OSC}/4096$ (in the case of a reset), or at the rate of the preset clock source (in the case of an external interrupt).

When bit 4 of the BT counter overflows, a signal is generated to indicate that the stabilization interval has elapsed. This allows the clock signal to be gated on 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 basic timer value increased at the rate of the preset clock source.
3. Clock oscillation stabilization interval begins and continues until bit 4 of the basic timer counter overflows.
4. When a bit 4 overflow of B TCNT occurs, normal CPU operation resumes.

TIMER 0 CONTROL REGISTER (T0CON)

The timer 0 control register, T0CON, is used to select the timer 0 operating mode (interval timer, capture mode, or PWM mode) and input clock frequency, to clear the timer 0 counter, and to enable the T0 overflow interrupt and T0 match/capture interrupt. It also contains a pending bit for T0 match/capture interrupts. It is located in set 1, address D2H, and is read/write addressable using Register addressing mode.

A reset clears T0CON to '00H'. This sets timer 0 to normal interval timer mode, selects an input clock frequency of f_{OSC} /4096, and disables the T0 overflow interrupt and match/capture interrupts. The T0 counter can be cleared at any time during normal operation by writing a "1" to T0CON.3.

The T0 overflow interrupt, T0INT, is IRQ0 with vector FAH. When a T0 overflow interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware. To enable the T0 match/capture interrupt (T0INT, IRQ0, vector FCH), you must set T0CON.1 to "1". The interrupt service routine must clear the pending condition by writing a "0" to the T0 interrupt pending bit, T0CON.0.

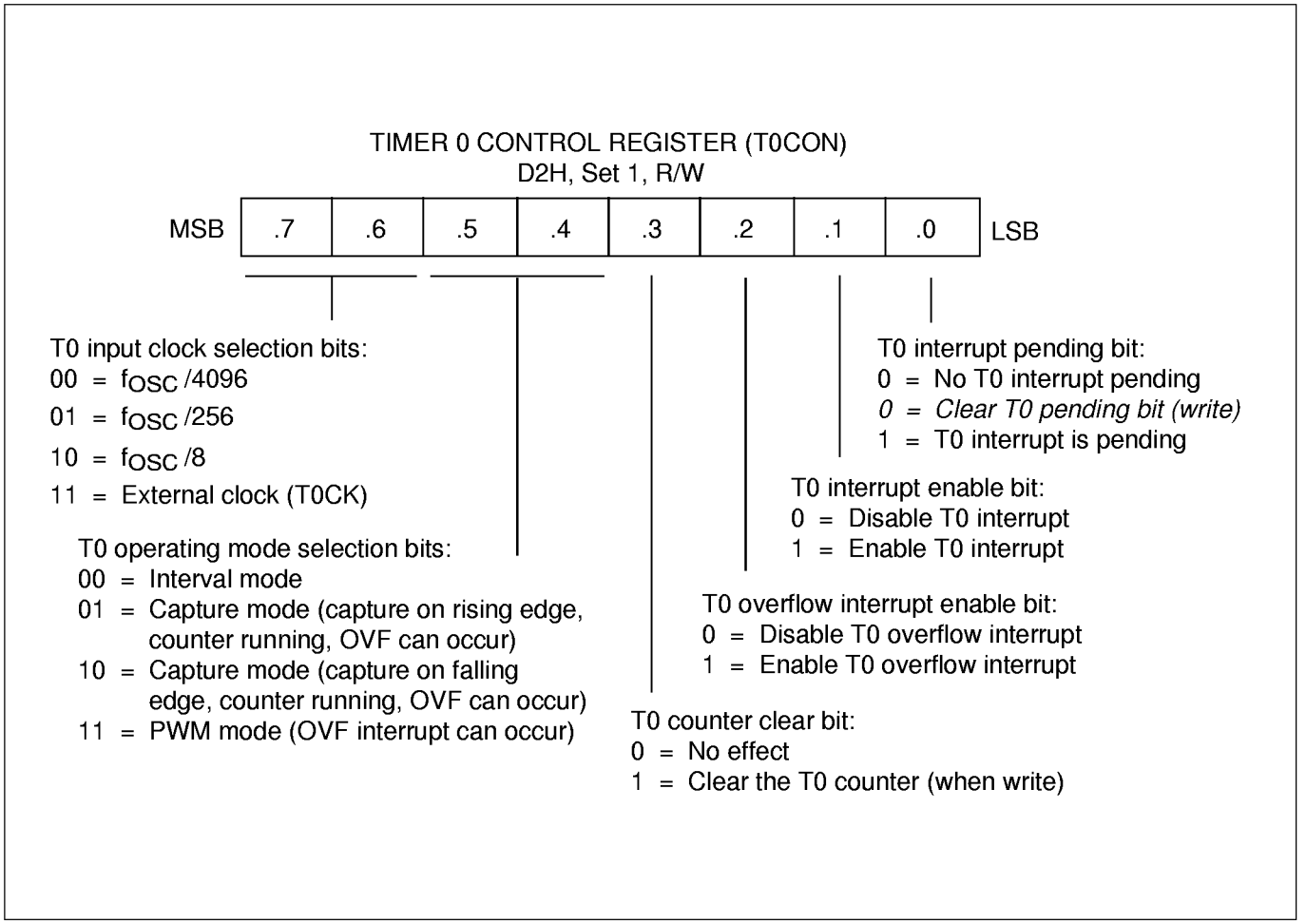


Figure 35. Timer 0 Control Register (T0CON)

TIMER 0 FUNCTION DESCRIPTION

T0 Interrupts (IRQ0, Vectors FAH and FCH)

The T0 module can generate two interrupts: the timer 0 overflow interrupt (T0OVF), and the timer 0 match/capture interrupt (T0INT). T0OVF is interrupt level IRQ0, vector FAH; T0INT is also level IRQ0, but has a different vector address: FCH. The T0OVF interrupt pending condition is automatically cleared by hardware when it has been serviced. The T0INT pending condition must be cleared by software by writing a "0" to the T0CON.0 pending bit.

Interval Timer Mode

In interval timer mode, a match signal is generated when the counter value is identical to the value written to the T0 reference data register, T0DATA. The match signal generates a T0 match interrupt (T0INT, vector FCH) 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 interrupt request is generated, the counter value is reset and counting resumes. With each match, the level of the signal at the T0 output pin is inverted. (See Figure 37.)

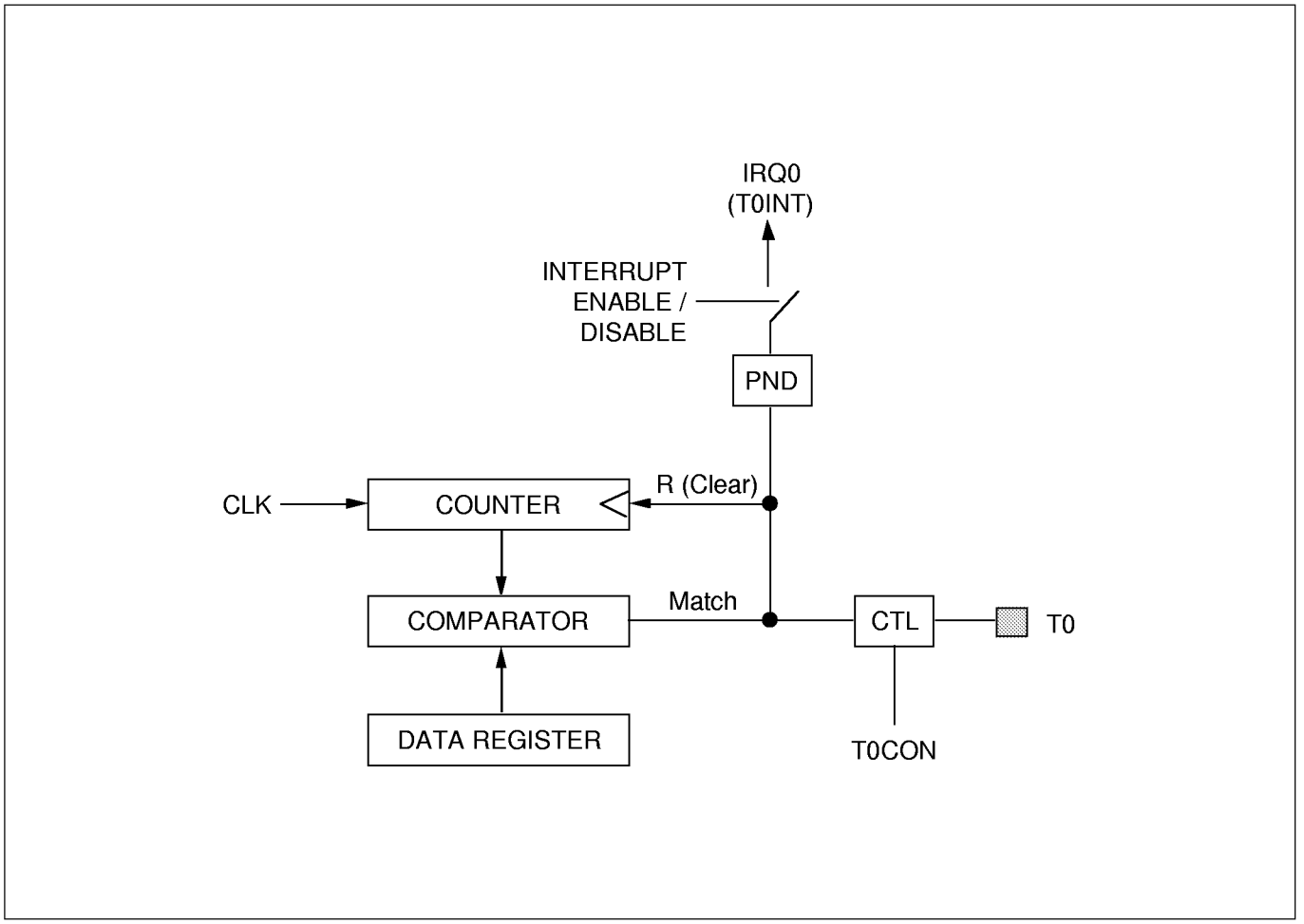
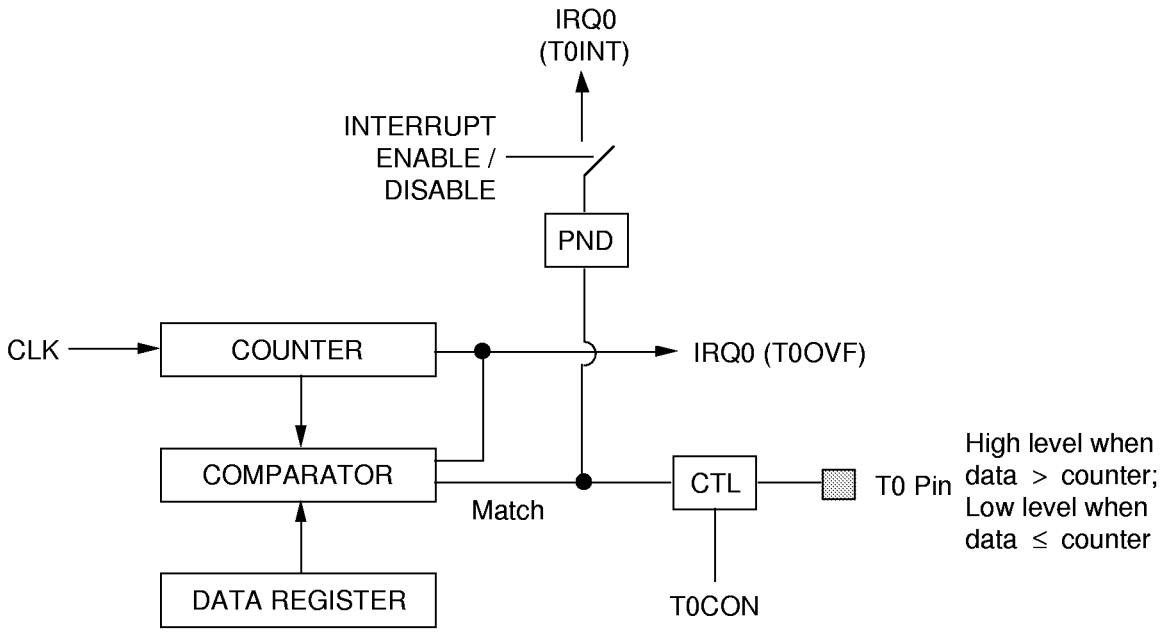


Figure 36. Timer 0 Function Diagram (Interval Timer Mode)

Pulse Width Modulation Mode

Pulse width modulation (PWM) mode lets you program the width (duration) of the pulse that is output at the T0 pin. As in interval timer mode, a match signal is generated when the counter value is identical to the value written to the T0 data register. In PWM mode, however, the match signal does not clear the counter (it runs continuously, overflowing at 'FFH', and continuing incrementing from '00H').

Although it is possible to use the match signal to generate a T0INT interrupt, an interrupt is typically not used in PWM-type applications. Instead, the pulse at the T0 pin is held to Low level as long as the reference data value is *less than or equal to* the counter value; the pulse is then held to High level for as long as the data value is *greater than* the counter value. One pulse width is equal to $t_{CLK} \times 256$. (See Figure 38).



NOTE: When the basic timer is configured to operate in PWM mode, interrupts are typically not used.

Figure 37. Timer 0 Function Diagram (PWM Mode)

Capture Mode

In capture mode, a signal edge that is detected tat the T0 pin opens a gate and loads the current counter value into the T0 data register. Rising edges or falling edges can be selected to trigger this operation. Both kinds of T0 interrupts can be used in capture mode: T0OVF is generated when a counter overflow occurs, and T0INT is generated when the counter value is loaded into the data register. By reading the captured data value in T0DATA, and assuming a specific value for t_{CLK}, you can determine the pulse width (duration) of the signal being input at the T0 pin. (See Figure 39).

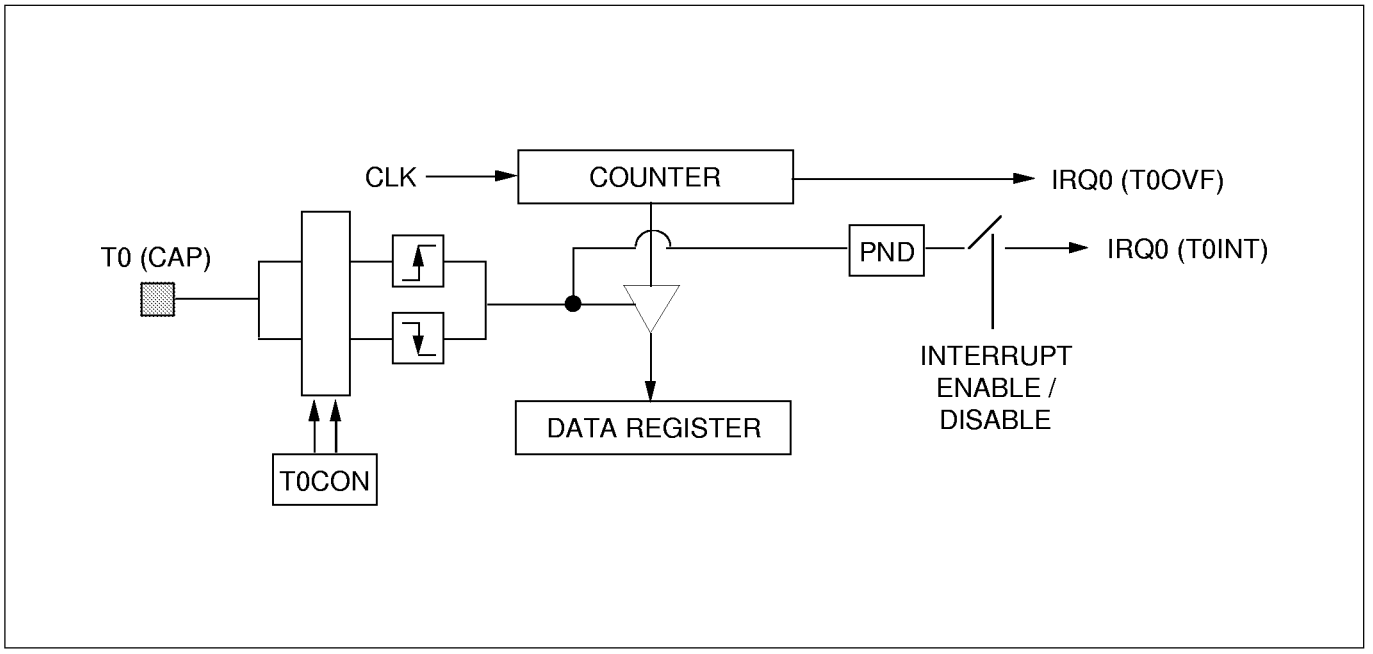
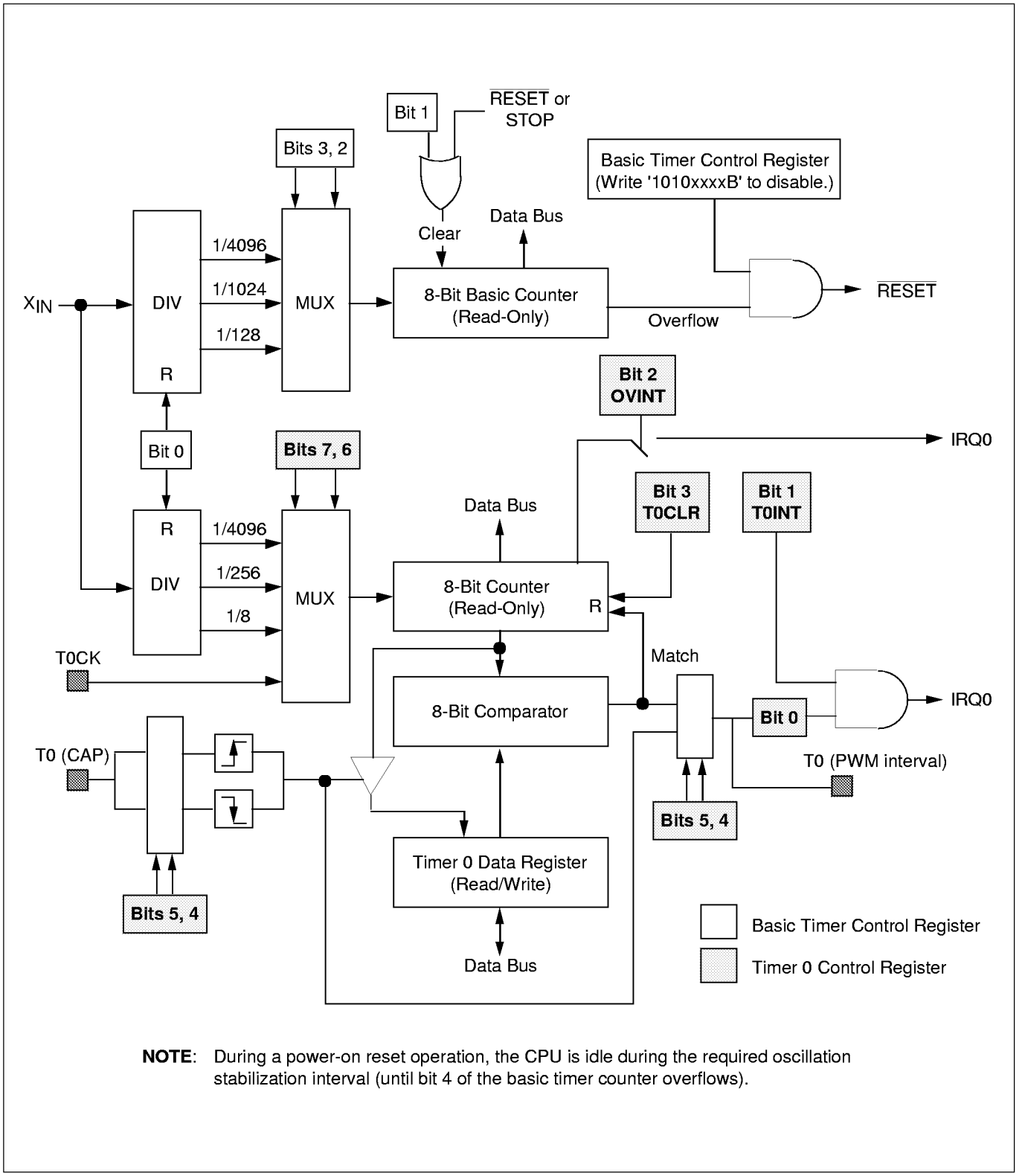


Figure 38. Timer 0 Function Diagram (Capture Mode)



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

Figure 39. Basic Timer and Timer 0 Block Diagram

☞ PROGRAMMING TIP — Configuring the Basic Timer

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

```
            ORG     0100H

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

☞ PROGRAMMING TIP — Configuring Timer 0

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

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

— Oscillation frequency is 6 MHz

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

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

RESET   DI                      ; Disable all interrupts
        SB0                     ; Select bank 0
        LD      BTCON,#0AAH     ; Disable the watchdog timer
        LD      CLKCON,#98H     ; Non-divided clock
        CLR     SYM             ; Disable global and fast interrupts
        CLR     SPL             ; Stack pointer low byte ← "0"
                                ; Stack area starts at 0FFH
        •
        •
        •
        LD      T0CON,#42H      ; 01000010B
                                ; Input clock is fOSC/256
                                ; Interval timer mode
                                ; Enable the timer 0 interrupt
                                ; Disable the timer 0 overflow interrupt
        LD      T0DATA,#5DH     ; Set timer interval to 4 milliseconds
                                ; (6 MHz/256) ÷ (93 + 1) = 0.25 KHz (4 ms)

        SRP     #0C0H           ; Set register pointer ← 0C0H
        EI                      ; Enable interrupts
        •
        •
        •
T0INT   PUSH    PP              ; Save page pointer to the stack
        PUSH    RP0             ; Save RP0 to stack
        SB0                     ; Select bank 0
        LD      PP,#00H         ; Page pointer ← 00H (select page 0)
        SRP0    #60H            ; RP0 ← 60H
        INC     R0              ; R0 ← R0 + 1
        ADD     R2,R0           ; R2 ← R2 + R0
        ADC     R3,R2           ; R3 ← R3 + R2 + Carry
        ADC     R4,R0           ; R4 ← R4 + R0 + Carry
        CP      R0,#32H         ; 50 × 4 = 200 ms
        JR      ult,NO_200MS_SET
        BITS    R1.2            ; Bit setting (61.2H)
```

(Continued on next page)

☞ **PROGRAMMING TIP — Configuring Timer 0 (Continued)**

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

T0OVER      IRET                    ;  Return from interrupt service routine
```

PWM and CAPTURE

PWM/CAPTURE MODULE

The KS88C3208/3216 microcontroller have two 14-bit PWM circuits and four 8-bit PWM circuits. The 14-bit circuits are called PWM0 and PWM1; the 8-bit circuits are PWM2–PWM5. The operation of all PWM circuits is controlled by a single control register, PWMCON. PWMCON also contains a 2-bit prescaler for adjusting the PWM frequency (cycle).

The capture function, called capture A, is integrated in this block. Using PWMCON settings, you can enable the capture A interrupt and select the desired triggering edge for data capture on the CAPA input pin.

The PWM counter is a 16-bit incrementing counter. It is used by both the 8-bit and 14-bit PWM circuits. To start the counter and enable the PWM circuits, you set PWMCON.5 to "1". If the counter is stopped, it retains its current count value; when re-started, it resumes counting from the retained count value.

A 2-bit prescaler controls the clock input frequency to the PWM counter. By modifying the prescaler value, you can divide the input clock by one (non-divided), two, three, or four. The prescaler output is the clock frequency of the PWM counter.

The PWM counter overflows when it reaches 'FFFFH', and then continues counting from zero. If the PWM counter overflow interrupt is enabled, an IRQ3 interrupt is generated. The enable bit for this interrupt is PWMCON.4.

PWM CONTROL REGISTER (PWMCON)

The control register for the PWM module, PWMCON, is located at register address F8H in set 1, bank 0. PWMCON is used for both the 8-bit and the 14-bit PWM modules. Bit settings in the PWMCON register control the following functions:

— 2-bit prescaler for scaling the PWM counter clock

— PWM counter stop/start (or resume) operation

— PWM counter overflow interrupt enable

— Capture A interrupt enable and capture A edge selection

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

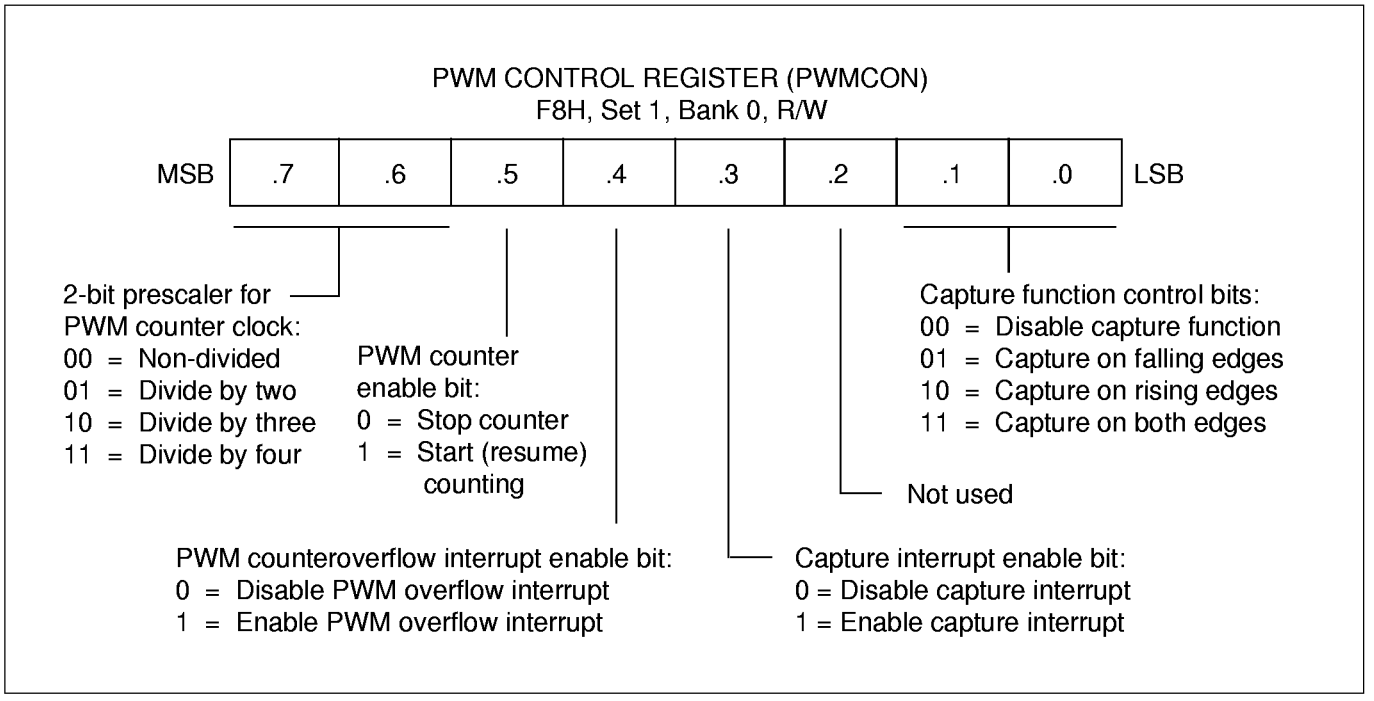


Figure 40. PWM Control Register (PWMCON)

PWM2–PWM5

The KS88C3208/3216 microcontroller have four 8-bit PWM circuits, called PWM2–PWM5. These 8-bit circuits have the following components:

- 16-bit counter with 2-bit prescaler
- 8-bit comparators
- 8-bit PWM data registers (PWM0–PWM5)
- PWM output pins (PWM2–PWM5)

The PWM2–PWM5 circuits are controlled by the PWMCON register (F8H, set 1, bank 0).

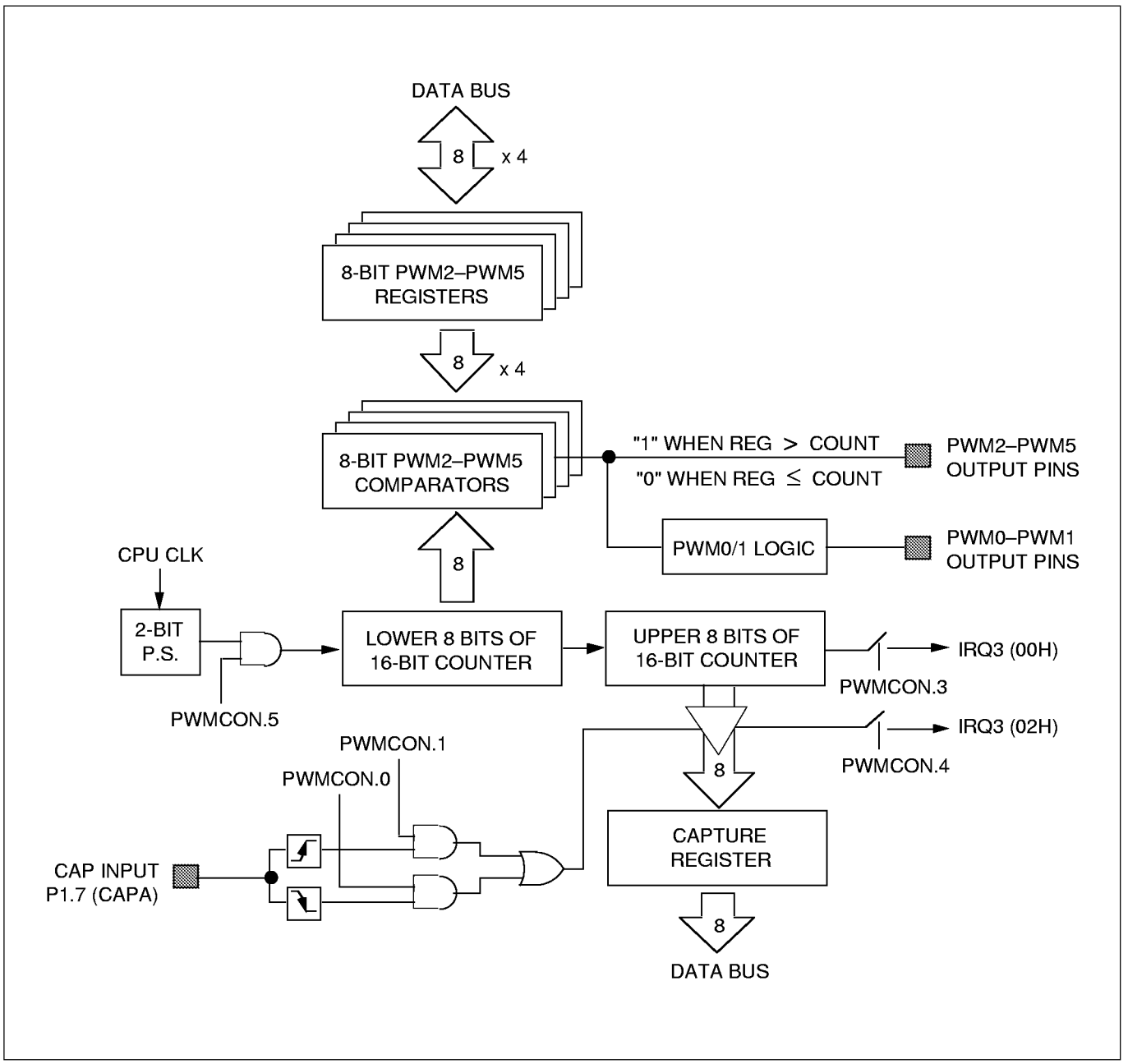


Figure 41. Block Diagram for PWM2–PWM5

PWM2–PWM5 FUNCTION DESCRIPTION

All four 8-bit PWM circuits function identically: Each has its own 8-bit data register and 8-bit comparator. Each circuit compares a unique data register value to the lower 8-bit value of the 16-bit PWM counter.

The PWM2–PWM5 data registers are located in set 1, bank 1, at locations F8H–FBH, respectively. These data registers are read/write addressable. By loading specific values into the respective data registers, you can modulate the pulse width at the corresponding PWM output pins, PWM2–PWM5.

The level at the output pins toggles High and Low at a frequency equal to the counter clock, divided by 256 (2^8). The duty cycle of the PWM0 and PWM1 pins ranges from 0% to 99.6%, based on the corresponding data register values.

To determine the PWM output duty cycle, its 8-bit comparator sends the output level High when the data register value is greater than the lower 8-bit count value. The output level is Low when the data register value is less than or equal to the lower 8-bit count value. The output level at the PWM2–PWM5 pins remains at Low level for the first 256 counter clocks. Then, each PWM waveform is repeated continuously, at the same frequency and duty cycle, until one of three events occurs:

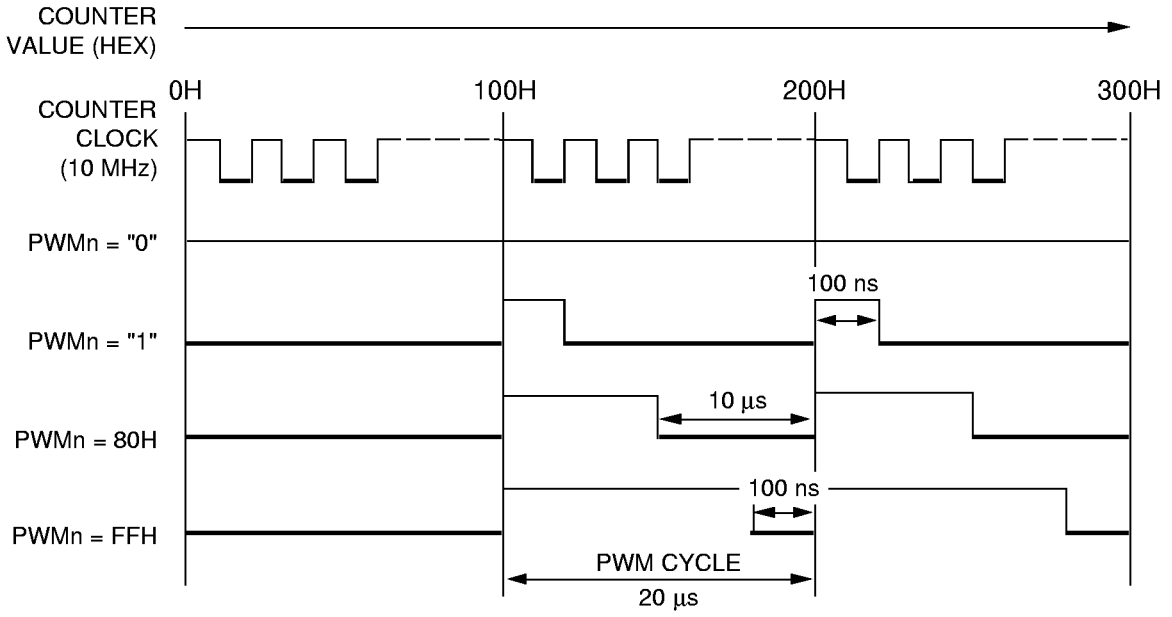
— The counter is stopped

— The counter clock frequency is changed

— A new value is written to the PWM data register

STAGGERED PWM OUTPUTS

The PWM2–PWM5 outputs are staggered in order to reduce the overall noise level on the pulse width modulation circuits. If you load the same value to the PWM2–PWM5 data registers, a match condition (data register value is equal to the lower 8-bit count value) will occur on the same clock cycle for all four 8-bit PWM circuits. The PWM3 output is delayed by one-half of a counter clock, PWM4 by one-half of a counter clock, PWM5 one-half of a counter clock, and so on for subsequent clock cycles (see Figure 44).



NOTES:
1. A counter clock value of 10 MHz is assumed for all timing values.
2. 'n' = 2–5, for PWM2–PWM5.

Figure 42. PWM Waveforms for PWM2–PWM5

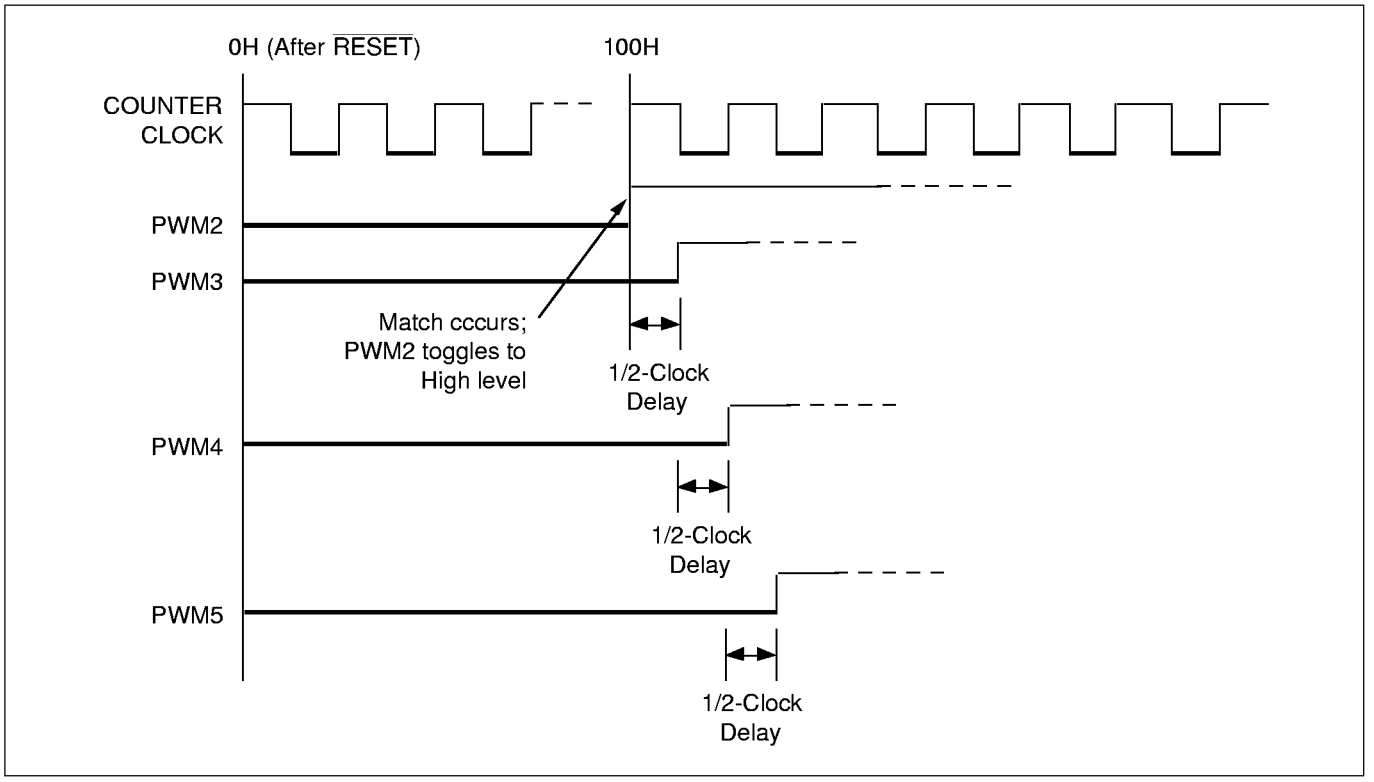


Figure 43. PWM Clock to PWM2–PWM5 Output Delays

PWM0 and PWM1

In addition to the four 8-bit PWM circuits, the KS88C3208/3216 pulse width modulation (PWM) module has two 14-bit PWM circuits, called PWM0 and PWM1. The 14-bit PWM circuits have the following components:

— 16-bit counter with 2-bit prescaler (an 8-bit counter with 6-bit extension is used for 14-bit output resolution)
— 8-bit comparator and extension cycle circuit
— 8-bit reference data registers (PWM0, PWM1)
— 6-bit extension data registers (PWM0EX, PWM1EX)
— PWM output pins (PWM0, PWM1)

The PWM0 and PWM1 circuits are enabled by the PWMCON register (F8H, set 1, bank 0).

PWM COUNTER

The PWM counter is a 16-bit incrementing counter comprised of a lower byte counter and an upper byte counter. The same 16-bit counter is used by all four PWM circuits (that is, by 8-bit *and* 14-bit circuits).

To determine the PWM module's base operating frequency, the lower byte counter is compared to the PWM data register value. In order to achieve higher resolutions, the lower six bits of the upper byte counter can be used to modulate the "stretch" cycle. To control the "stretching" of the PWM output duty cycle at specific intervals, the 6-bit extended counter value is compared with the 6-bit value (bits 2–7) that you write to the module's extension register.

PWM DATA AND EXTENSION REGISTERS

Two PWM (duty) data registers, located in set 1, bank 0, determine the output value generated by each 14-bit PWM circuit. These registers, PWM0 and PWM1 are read/write addressable..

— 8-bit data registers PWM0 (F4H) and PWM1 (F6H)
— 8-bit extension registers PWM0EX (F5H) and PWM1EX (F7H), of which only bits 2–7 are used

To program the required PWM output, you load the appropriate initialization values into the 8-bit data registers (PWM0, PWM1) and the 6-bit extension registers (PWM0EX, PWM1EX). To start the PWM counter, or to resume counting, you set PWMCON.5 to "1".

A reset operation disables all PWM output. The current counter value is retained when the counter stops. When the counter starts, counting resumes at the retained value.

PWM CLOCK RATE

The timing characteristics of both 14-bit output channels are identical, and are based on the maximum 8-MHz CPU clock frequency. The 2-bit prescaler value in the PWMCON register determines the frequency of the counter clock: You can set PWMCON.6 and PWMCON.7 to divide the CPU clock frequency by one (non-divided), two, three, or four.

Because the maximum CPU clock rate for the KS88C3208/3216 microcontroller is 8 MHz, the maximum base PWM frequency is 31.25kHz (8MHz divided by 256). This assumes a non-divided CPU clock.

Table 14. PWM0 and PWM1 Control and Data Registers

Register Name	Mnemonic	Address (Set 1, Bank 0)	Function
PWM0 data registers	PWM0	F4H	8-bit PWM0 basic cycle frame value
	PWM0EX	F5H	6-bit extension ("stretch") value
PWM1 data registers	PWM1	F6H	8-bit PWM1 basic cycle frame value
	PWM1EX	F7H	6-bit extension ("stretch") value
PWM control register	PWMCON	F8H	PWM0/1 counter stop/start (resume), and 2-bit prescaler for CPU clock; also contains capture A control settings

PWM0/PWM1 FUNCTION DESCRIPTION

The PWM output signal toggles to Low level whenever the lower 8-bit counter matches the reference value stored in the module's data register (PWM0 or PWM1). If the value in the PWM data register is not zero, an overflow of the lower counter causes the PWM output to toggle to High level. In this way, the reference value written to the data register determines the module's base duty cycle.

The value in the 6-bit extension counter (the lower six bits of the upper counter) is compared with the extension settings in the 6-bit extension data register (PWM0EX, PWM1EX). This 6-bit extension counter value (bits 2-6), together with extension logic and the PWM module's extension register, is then used to "stretch" the duty cycle of the PWM output. The "stretch" value is one extra clock period at specific intervals, or cycles (see Table15).

If, for example, the value in the extension register is '1', the 32nd cycle will be one pulse longer than the other 63 cycles. If the base duty cycle is 50%, the duty of the 32nd cycle will therefore be "stretched" to approximately 51% duty. For example, if you write 80H to the extension register, all odd-numbered pulses will be one cycle longer. If you write FCH to the extension register, all pulses will be stretched by one cycle except the 64th pulse. PWM output goes to an output buffer and then to the corresponding PWM0 or PWM1 output pin. In this way, you can obtain high output resolution at high frequencies.

Table 15. PWM Output "Stretch" Values for Extension Registers PWM0EX and PWM1EX

PWM0/1EX Bit	"Stretched" Cycle Number
7	32
6	16,48
5	8, 24, 40, 56
4	4, 12, 20, . . . , 44, 52, 60
3	2, 6, 10, 14, . . . , 50, 54, 58, 62
2	1, 3, 5, 7, 9, . . . , 55, 57, 59, 61, 63
1	Not used
0	Not used

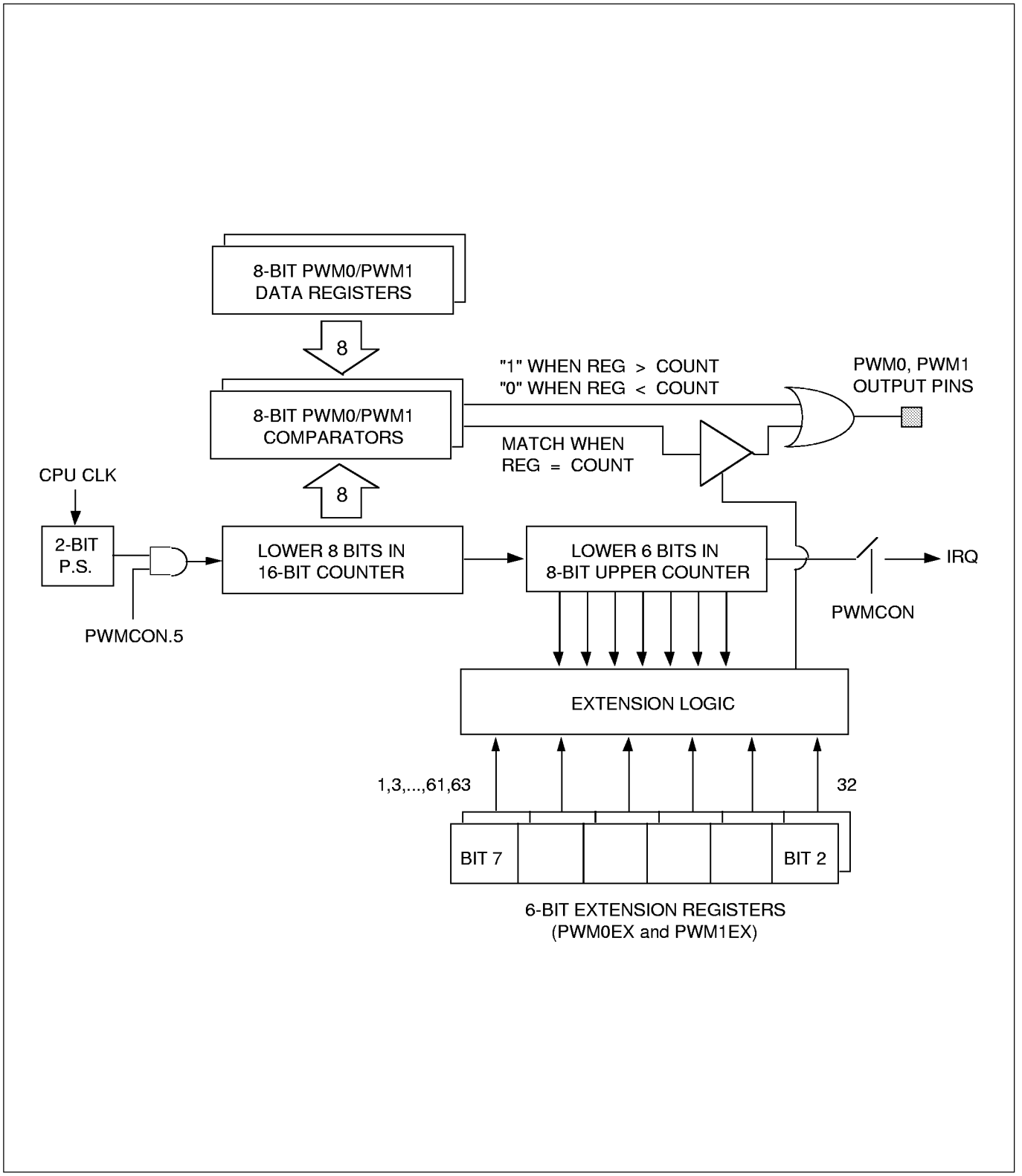


Figure 44. Block Diagram for PWM0 and PWM1

☞ PROGRAMMING TIP — Programming PWM0 to Sample Specifications

This example shows how to program the 14-bit pulse-width modulation module, PWM0. In this case, the PWM1 module is disabled. (The procedure for programming the PWM1 module is identical.) The program parameters are as follows:

— The oscillation frequency of the main crystal is 6 MHz

— PWM0 data is in working register R0

— PWM0EX (PWM0 extension value) is in working register R1, bits 2–7

The program performs the following operations:

1. Set the PWM0 frequency to 23.437 kHz
2. If R3.0 = "1", then PWM ← PWM + 12H
   (If an overflow occurs from R0, then R0 ← 0FFH and R1 ← 0FCH.)
3. If R3.0 = "0", then PWM ← PWM – 11H
   (If an underflow occurs from R0, then R0 ← 00H and R1 ← 00H.)

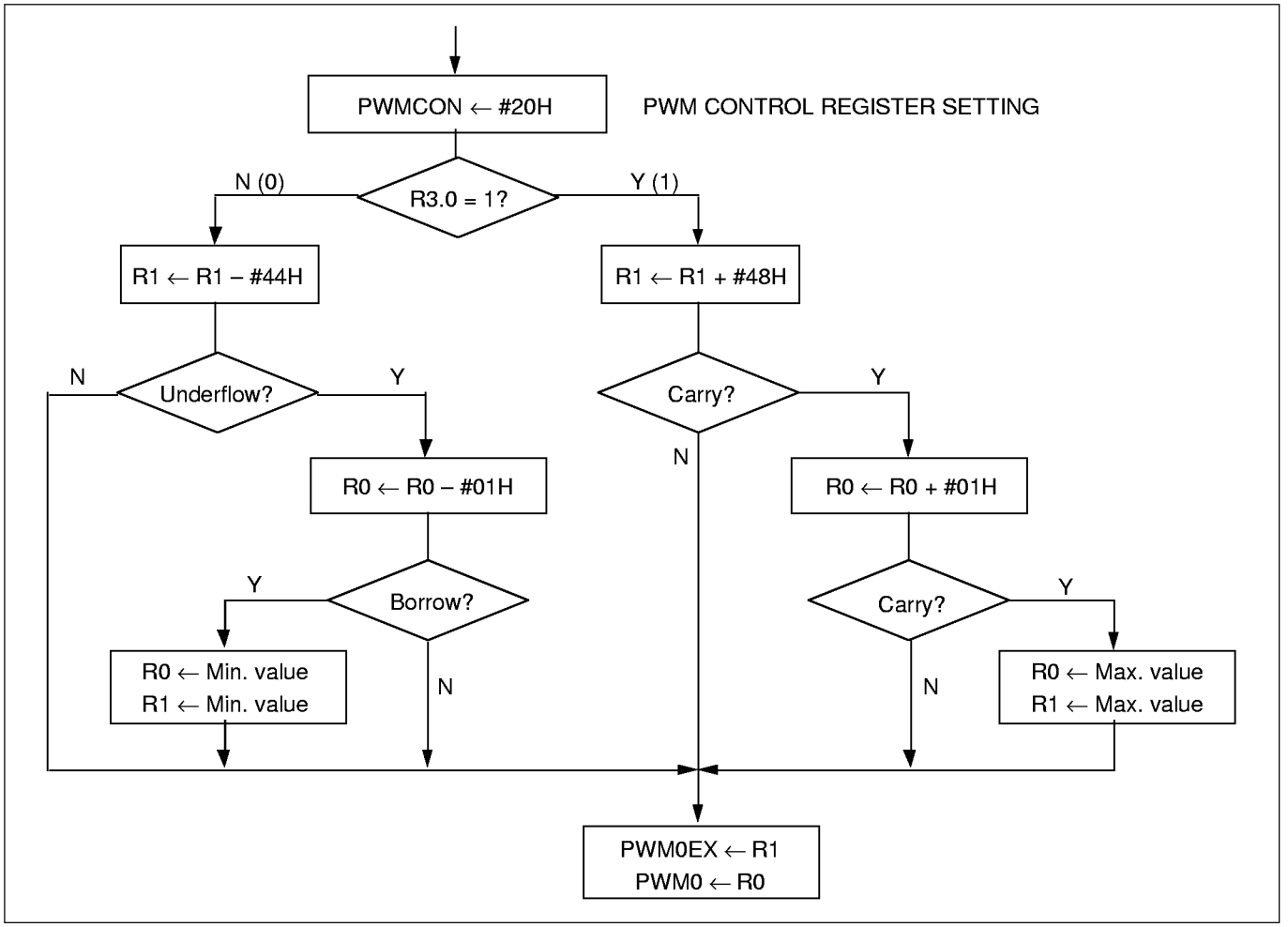


Figure 45. Decision Flowchart for PWM0 Programming Tip

☞ **PROGRAMMING TIP — Programming PWM0 to Sample Specifications (Continued)**

```
                •
                •
                •
                LD      PWMCON,#20H     ; PS ← 0 (Select 23.437-kHz PWM frequency)
                                        ; Enable the PWM counter
                •
                •
                •
                BTJRF   pwm0_dec,R3.0   ; If R3.0 = "0", then jump to pwm0_dec

pwm0_inc:
                ADD     R1,#48H         ; If R3.0 = "1", then add 48H to the PWM data
                JR      NC,pwm0_data_end ; If no carry, go to pwm0_data_end
                INC     R0              ; R0 ← R0 + 1
                JR      NZ,pwm0_data_end ; If no overflow, jump to pwm0_data_end for update
                LD      R0,#0FFH        ; If overflow, set 0FFH to R0
                LD      R1,#0FCH        ; Set 0FCH to R1
                JR      T,pwm0_data_end ; Jump to pwm0_data_end unconditionally

pwm0_dec:
                SUB     R1,#44H         ; R3.0 = "0", so subtract 44H from PWM data
                JP      NC,pwm0_data_end ; If no borrow, jump to pwm0_data_end for update
                SUB     R0,#01H         ; Decrement R0 (R0 ← R0 – 1)
                JR      NC,pwm0_data_end ; If no borrow, jump to pwm0_data_end
                CLR     R0              ; Clear data R0
                CLR     R1              ; Clear data R1

pwm0_data_end:
                LD      PWM0EX,R1       ; Load new value to PWM0EX (bits 2–7)
                LD      PWM0,R0         ; Load new value to PWM0
                •
                •
                •
```

CAPTURE UNIT

An 8-bit capture unit is integrated in the PWM module. The capture unit detects incoming signal edges and can be used to measure the pulse width of the incoming signals. PWMCON register settings control the capture unit, which has the following components:

— 8-bit capture data register (CAPA)

— Capture input pin (P1.7/CAPA)

— 8-bit capture interrupt (IRQ3, vector 02H)

The capture unit captures the upper 8-bit value of the 16-bit counter when a signal edge transition is detected at the CAPA pin. The captured value is then dumped into the capture A data register, also called CAPA, where it can be read.

Using PWMCON.0 and PWMCON.1 settings, you can set edge detection at the CAPA pin for rising edges, falling edges, or for both signal edge types.

You can also use signal edges at the CAPA pin to generate an interrupt. PWMCON.3 is the capture A interrupt enable bit.

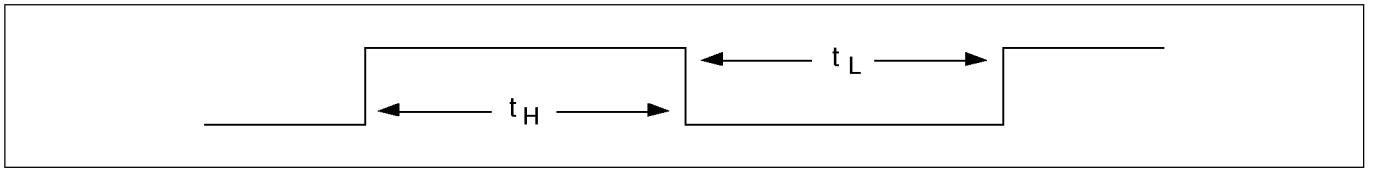
The capture interrupt is level 3 (IRQ3) and its vector address is 02H. Level IRQ3 has two interrupts: the PWM counter overflow interrupt (vector 00H) and the capture A interrupt (vector 02H). The PWM counter overflow interrupt has higher priority in the interrupt structure.

Using the capture A interrupt, you can read the contents of the CAPA data register from edge to edge and use the values to calculate the elapsed time between pulses.

☞ PROGRAMMING TIP — Programming the Capture Module to Sample Specifications

This example shows you how to program the KS88C3208/3216 capture A module. The sample parameters are as follows:

— The main oscillator frequency is 6 MHz
— Timer A interrupt occurs every 2 ms
— The following waveform is currently being input at the capture (CAP) pin:



— The following registers are assigned for program values:

```
Register 70H    LDR       ;  First captured count value
Register 71H              ;  Second captured count value
Register 72H              ;  Third capture count value
Register 73H    DWNCNT    ;  Down-counter; decremented by 1 with each timer A interrupt
Register 74H    CAPCNT    ;  Capture counter
Register 77H    FLAG      ;  Flags
```

Here is some additional information about the sample program:

1. If 4.35 ms < t_H, t_L < 4.6 ms, then set bit zero (LDR) in register 77H; otherwise clear the zero bit (LDR) in register 77H.
2. If the interval between two rising signal edges (capture trigger) is > 30 ms, disregard the capture setting.

Figures 47 and 48 show decision flowcharts for the sample program.

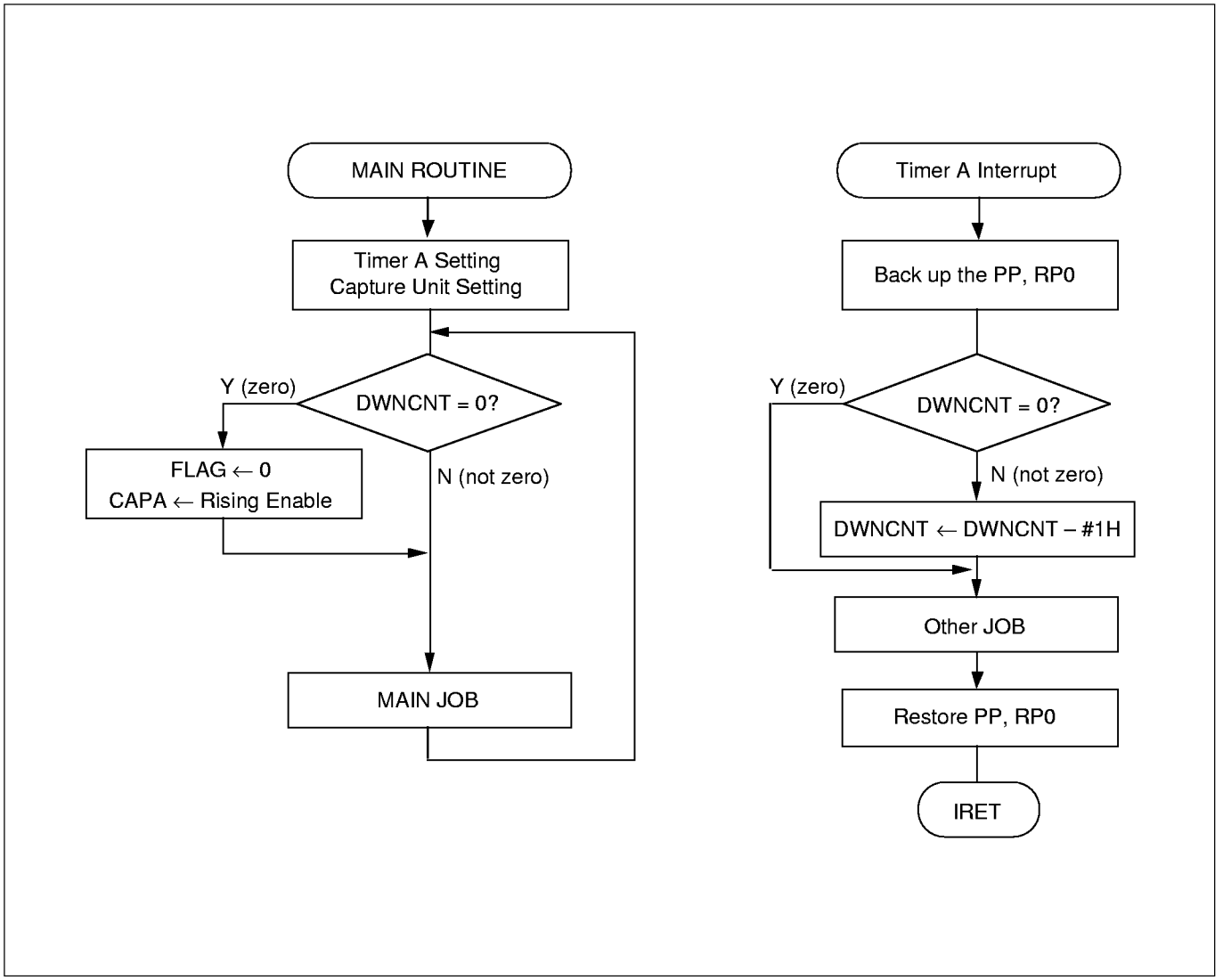


Figure 46. Decision Flowchart (Main Routine and Timer A Interrupt)

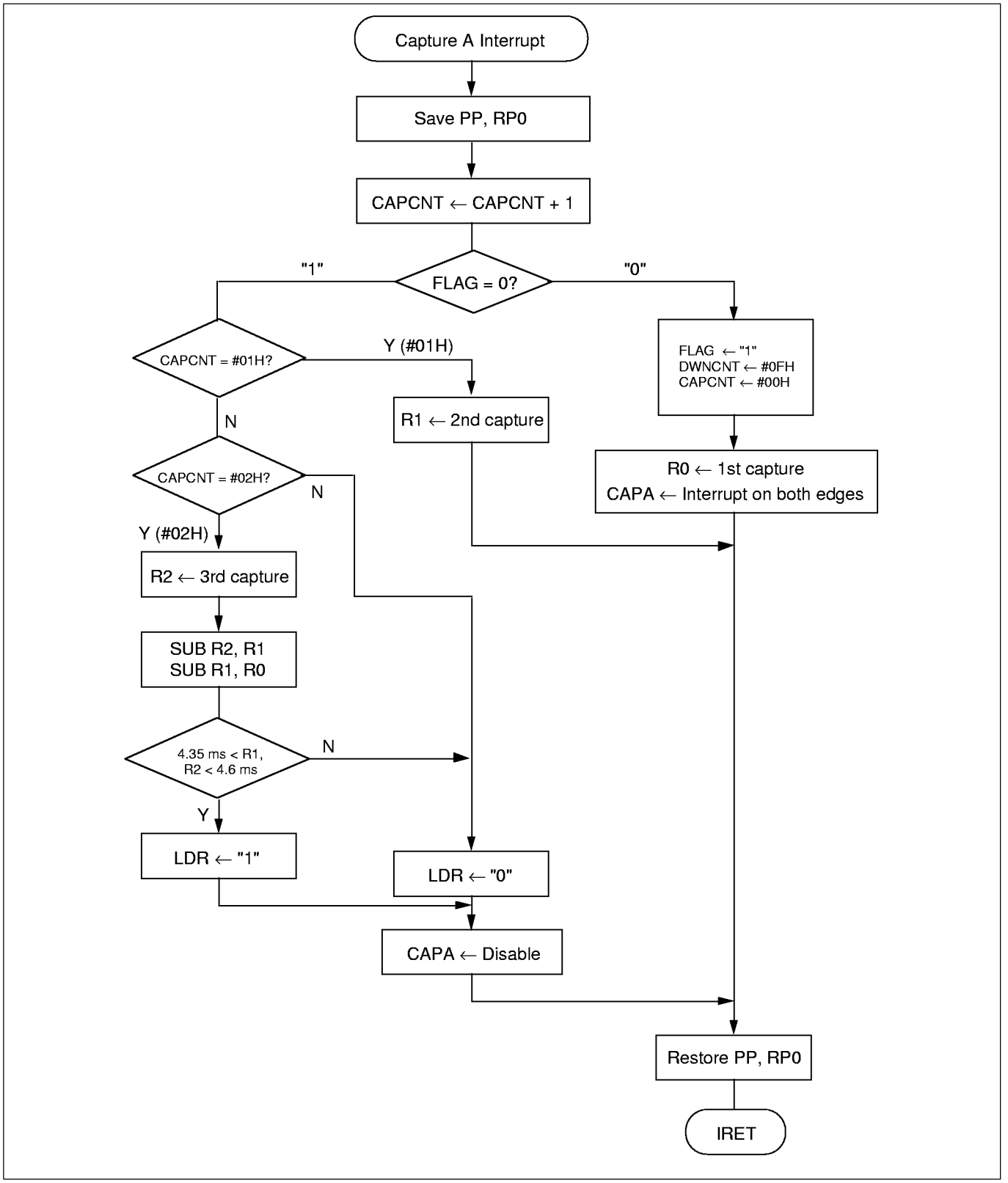


Figure 47. Decision Flowchart for Capture A Interrupt

☞ **PROGRAMMING TIP — Programming the Capture Module to Sample Specifications (Continued)**

```
            •
            •
            •
LDR         EQU     0
DWNCNT      EQU     3
CAPCNT      EQU     4
FLAG        EQU     7
            •
            •
            •
            CLR     PP                  ; Select page 0
            LD      P1CONH,#00H         ; Set P1.7 to capture input mode
            LD      TACON,#54H          ; PS ← 5, interval mode
                                        ; Enable timer A interrupt
            LD      TADATA,#01H         ; 2-ms interval (6 MHz /1000 ÷ 6 ÷ 2 = 0.5 kHz = 2 ms)
            •
            •
            •
exec_main:
            SRP0    #70H                ; RP0 ← 70H
            CP      RDWNCNT,#00H        ; Down-counter = "0"?
            JP      NE,MAIN             ; If not zero, then jump to MAIN
            BITR    R7.FLAG             ; Clear the 'FLAG'
            LD      PWMCON,#0AH         ; Enable capture A interrupt
                                        ; Trigger interrupt on rising edges
MAIN:                                   ; Other job...
            •
            •
            •
            JP      T,exec_main         ; For looping
            •
            •
            •
TAINT       PUSH    PP                  ; Save page pointer
            PUSH    RP0                 ; Save register pointer 0
            SRP0    #70H                ; RP0 ← 70H
            CP      RDWNCNT,#00H        ; R3 (down-counter) = "0"?
            JP      EQ,ta_exec          ;
            DEC     RDWNCNT             ; If not zero, then decrement R3 by 1
ta_exec:
            •
            •
            •
            POP     RP0                 ; Restore register pointer 0
            POP     PP                  ; Restore page pointer
            IRET                        ; Return from timer A interrupt service routine
```

(Continued on next page)

☞ **PROGRAMMING TIP — Programming the Capture Module to Sample Specifications (Continued)**

```
CAPINT    PUSH    PP                  ; Save the page pointer to stack
          PUSH    RP0                 ; Save register pointer 0 to stack
          SRP0    #70H                ; RP0 ← 70H
          INC     RCAPCNT             ; Increment the capture counter
          BTJRT   cap_one,R7.FLAG     ; R7.FLAG ← "1", then jump to cap_one
          BITS    R7.FLAG             ; Set R7.FLAG
          CLR     RCAPCNT             ; Clear capture counter
          LD      RDWNCNT,#0FH        ; Down-counter ← 15 (for counting 30 ms)
          LD      R0,CAPA             ; R0 ← 1st captured count value
                                      ; CAPA = 0F9H, page 0
          LD      PWMCON,#0BH         ; Enable capture interrupt
                                      ; Trigger interrupt on both rising and falling edges

cap_end   POP     RP0                 ; Restore the register pointer 0 value
          POP     PP                  ; Restore the page pointer value
          IRET                        ;

cap_one   CP      RCAPCNT,#01H        ; CAPCNT = #01H?
          JP      NE,cap_con2         ;
          LD      R1,CAPA             ; R1 ← 2nd captured count value
          JR      T,cap_end           ;

cap_con2  CP      RCAPCNT,#02H        ; CAPCNT = #02H?
          JP      EQ,cap_con3         ;

cap_con4  BITR    R7.LDR              ; Clear the LDR bit in R7
cap_con5  LD      PWMCON,#00H         ; Disable the capture module
          JR      T,cap_end           ;

cap_con3  LD      R2,CAPA             ; R2 ← 3rd capture count value
          SUB     R2,R1               ; R2 ← (3rd capture value – 2nd capture value)
          SUB     R1,R0               ; R1 ← (2nd capture value – 1st capture value)
          CP      R1,#24H             ; 24H = 4.6 ms

          JP      UGT,cap_con4        ; If High signal period > 4.6 ms, then go to cap_con4
          CP      R2,#24H             ;
          JP      UGT,cap_con4        ; If Low signal period > 4.6 ms, then go to cap_con4

          CP      R1,#22H             ; 22H = 4.35 ms

          JP      ULT,cap_con4        ; If High signal period < 4.35 ms, then go to cap_con4
          CP      R2,#22H             ;
          JP      ULT,cap_con4        ; If Low signal period < 4.35 ms, then go to cap_con4

          BITS    R7.LDR              ; Set bit 'LDR'
          JP      T,cap_con5          ; Jump to cap_con5 unconditionally
          •
          •
          •
```

ON-SCREEN DISPLAY (OSD)

OVERVIEW

The on-screen display (OSD) module displays channel number, the time, and other information on a display screen. The OSD character display module has 240 locations and supports a set of 128 characters. (Two characters are reserved: 00H for the blank function and 7FH for the test pattern.) There are eight display colors.

PATTERN GENERATION SOFTWARE

For application development using the KS88C3208/3216 microcontroller, Samsung provides OSD pattern generation software (filename **CGROM1.EXE**). You can customize standard OSD patterns contained in this file.

INTERNAL OSD CLOCK

Red-green-blue (RGB) color outputs, as well as display rates and positions, are determined by the clock signal, DOT_CLK. This signal is generated by the L-C oscillator and is scaled by the dot and column counter. DOT_CLK equals the OSD oscillator clock divided by the clock divider value. The clock divider value is set by the horizontal character size settings in the CHACON register.

The rate at which each new display line is generated is determined by H-sync input. The rate at which each new frame (screen) is generated is determined by V-sync input. The recommended L-C clock frequency is 6.5 MHz.

OSD VIDEO RAM

The OSD video RAM contains 240 word lines. Each line is 11 bits long. Of these 11 bits, seven are character display codes (bits 0–6). Bit 7 is the character halftone or character background color display control bit and bits 8–10 are used to determine the red, green, and blue components of the character color.

Table 16. OSD Function Block Summary

OSD Function Block	Function Description
Video RAM	Located in register page 1, the video RAM contains 240 "word" lines. Each line is 11bits long. Each 11-bit RAM address stores a 7-bit character code, a character halftone or character background color display control bit, and a 3-bit color code. Video RAM locations can be read or written: 00H–BFH can be accessed using any addressing mode; C0H–EFH can be accessed using Indirect Register or Indexed addressing mode only.
Character ROM	The character ROM contains an 18-dot × 12-dot matrix data for 128 characters. It is synchronized with the internal dot clock. The ROM outputs the dot matrix data for each character. The function of two characters is pre-determined: 00H is used for blank (no-display) data and 7FH is used for a factory test pattern.
Output control logic	Output control logic receives input from the CG ROM1, OSD control registers, and fade control circuits. It then decides what to display on the screen and what color the display should be. On the basis of truth table calculations, the final OSD signals (blue, green, red, and blank) are output from the OSD block at pins 22–25.

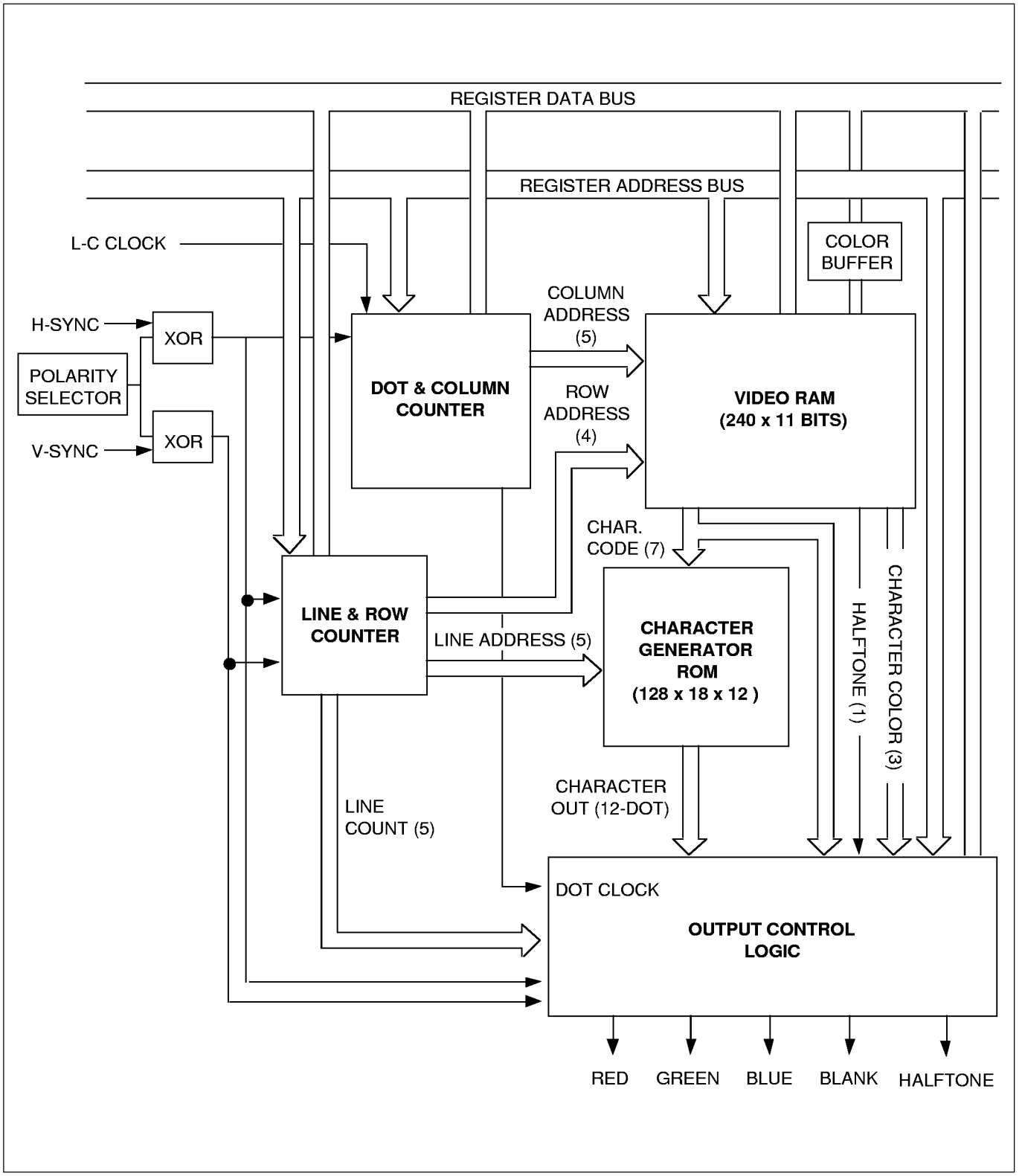
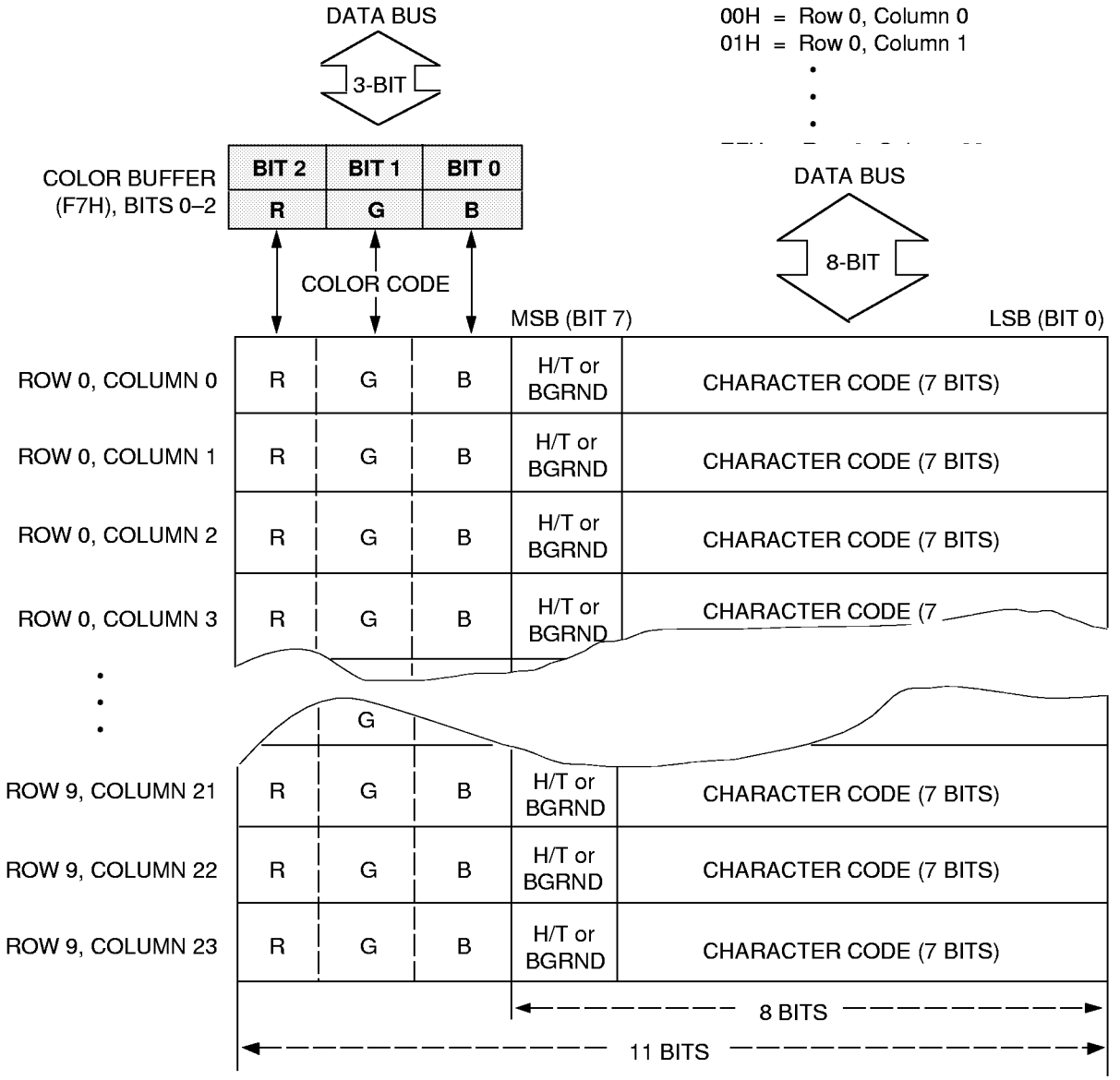


Figure 48. On-Screen Display Function Block Diagram



NOTE:
1. Bit 7 controls the character halftone or character background color display function. A "0" enables the character background color display f function and a "1" enables the character halftone display function
2. When programming the OSD, writing operations to the OSD video RAM must be synchronized to the V-sync signal when changing a display format. Otherwise, flicker will occur.
3. The contents of the color buffer COLBUF are changed each time the OSD video RAM is read.(color data bits are loaded into COLBUF with each subsequent read operation)

Figure 49. On-Screen Display Video RAM Data Organization

OSD CONTROL REGISTER OVERVIEW

Seven control registers are used to control specific functions of the on-screen display module:

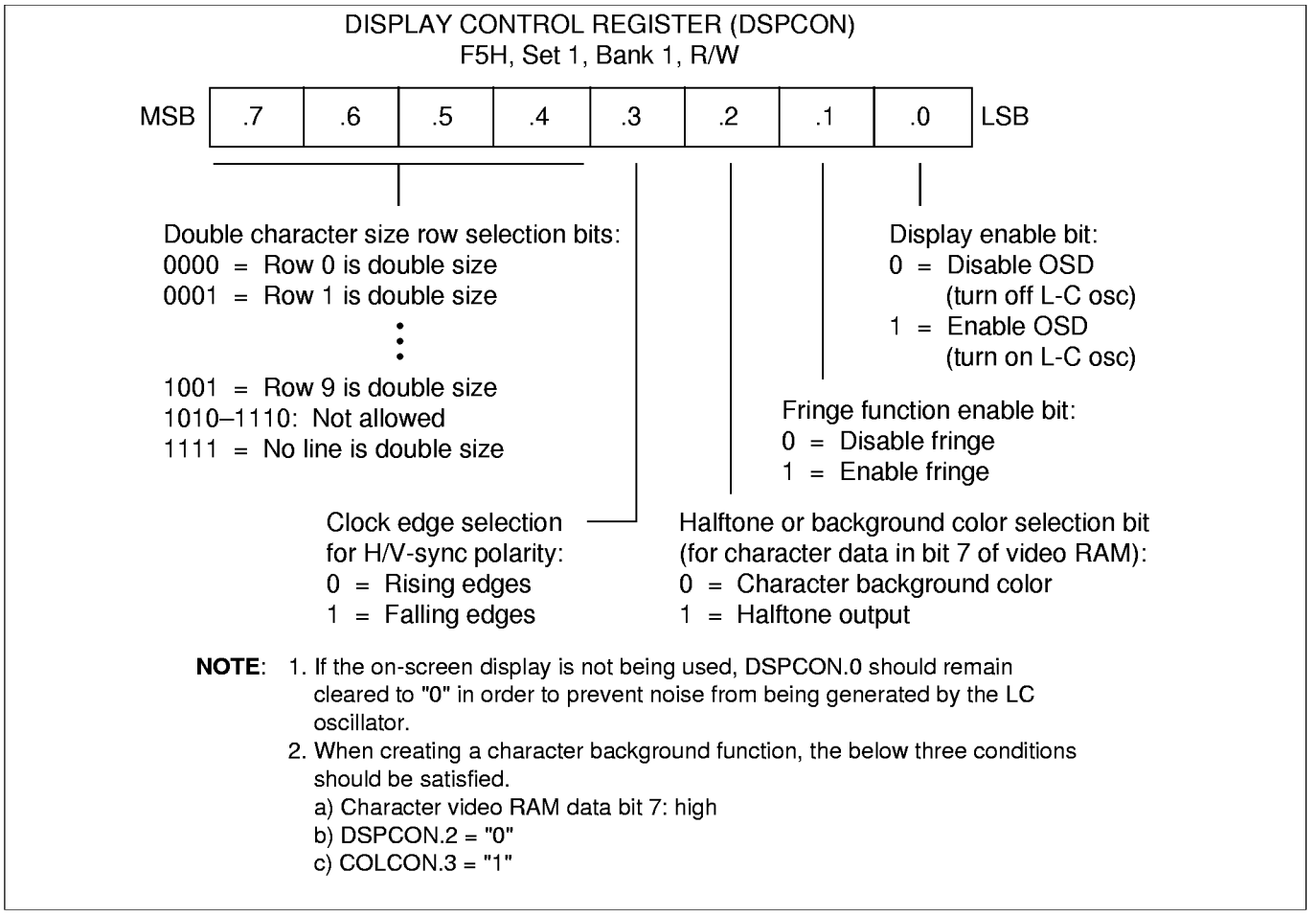
There are seven control registers for OSD functions and one color buffer register:

DSPCON	Display control register
CHACON	Character size and fade control register
FADECON	Fade control register
ROWCON	Row display position and inter-row spacing control register
CLMCON	Column display position and inter-column spacing control register
COLCON	Background color control register
HTCON	Halftone signal control register
COLBUF	Character color buffer register

These registers are described in this section within the context of the OSD hardware module description. For detailed, quick-reference descriptions of the control register bit settings, please refer to Section 4, "Control Registers."

DISPLAY CONTROL REGISTER (DSPCON)

Settings in the display control register, DSPCON (F5H, set 1, bank 1), are used to enable and disable the on-screen display, to control the fringe function, to select halftone or background color for character displays, choose the polarity for H-sync and V-sync signal synchronization, and to select double-size character display for a row:



NOTE: 1. If the on-screen display is not being used, DSPCON.0 should remain cleared to "0" in order to prevent noise from being generated by the LC oscillator.
2. When creating a character background function, the below three conditions should be satisfied.
a) Character video RAM data bit 7: high
b) DSPCON.2 = "0"
c) COLCON.3 = "1"

Figure 50. OSD Display Control Register (DSPCON)

OSD Enable/Disable

The DSPCON.0 setting enables or disables the on-screen display module. To enable the OSD (turn L-C oscillation on), set DSPCON.0 to "1"; to disable OSD (turn L-C oscillation off), clear DSPCON.0 to "0". When you do not use the display module, we recommend that you keep DSPCON.0 cleared to "0" in order to reduce possible noise generation by the L-C oscillator.

OSD Fringe Function

DSPCON.1 enables or disables the fringe area display. Fringe (black color only) is added to the bottom and right-hand sides of the displayed character (see Figure 52). The fringe gives a shadow effect to the character in order to enhance readability.

H-Sync and V-Sync Polarity Selection

DSPCON.3 selects the triggering edge of H-sync and V-sync inputs to the OSD block. Incoming sync pulses enter a polarity option circuit that is controlled by the SYNC bit. If DSPCON.3 = "0", rising edges are selected; if it is "1", falling edges are selected.

Character Halftone or Character Background Color Selection

DSPCON.2 lets you select a halftone or background color display for individual characters. (Which characters are displayed as halftones, or with character background color, depends on the bit 7 settings in the character video RAM data.) When DSPCON.2 = "0", the character background color option is selected; when DSPCON.2 = "1", the character halftone function is selected.

Double-Size Character Row Display Function

You can select any one of the ten rows (0–9) for double-size character display. To do this, you set DSPCON.4–DSPCON.7 to the value that corresponds to the row you want displayed in double-size characters.

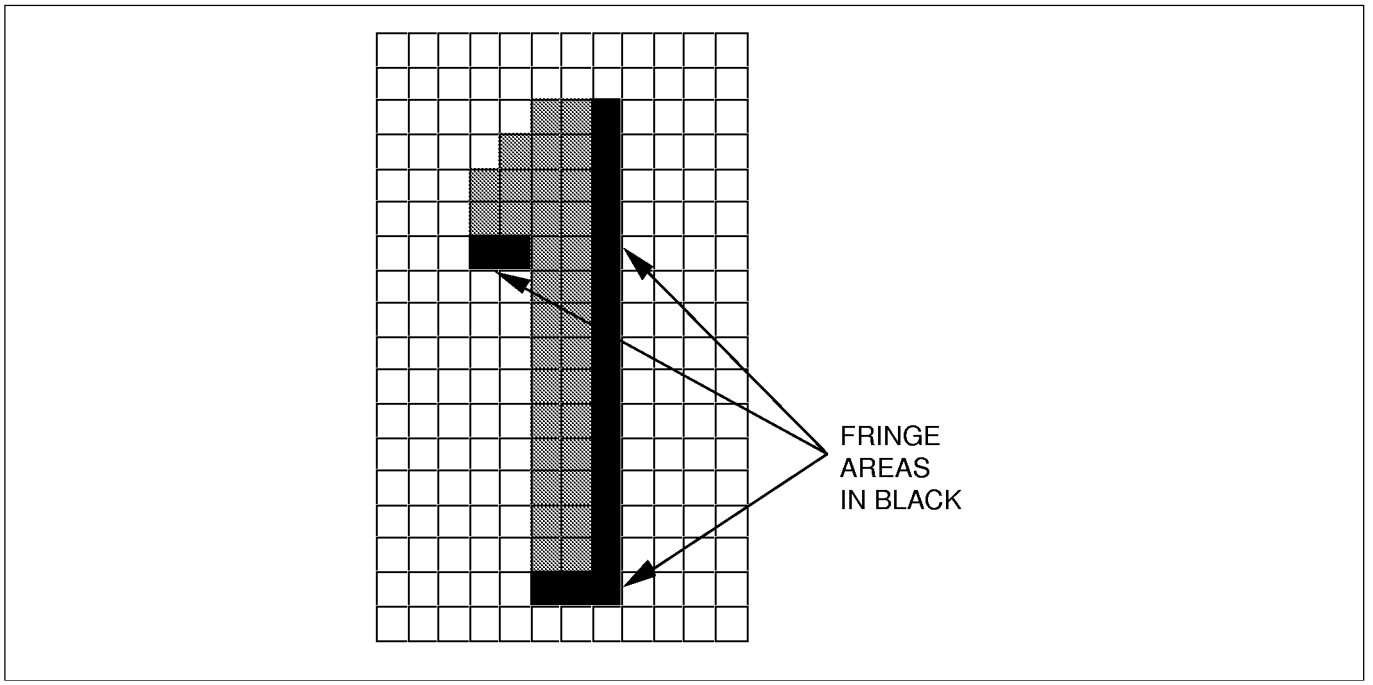


Figure 51. OSD Fringe Display Function

CHARACTER SIZE CONTROL REGISTER (CHACON)

Using the character size control register, CHACON, you can specify four different standard character sizes in both vertical and horizontal directions. You also use the CHACON register to select rows (0–9) for the character fade function (see Figure 53).

Vertical character size is defined by bits 6 and 7 of the CHACON register; horizontal direction is defined by bits 4 and 5. There are four basic character size settings: ×1, ×2, ×3, and ×4. Size '×1' is the smallest size and '×4' is the largest. For example, to display a '×1' (horizontal) by '×1' (vertical) size character, you would clear CHACON.4–CHACON.7 to "0". To display a '×4' by '×4' size character, you would set bits 4–7 to '1111B'.

You can also combine different vertical and horizontal size selections to produce flattened or elongated characters (see Figure 54).

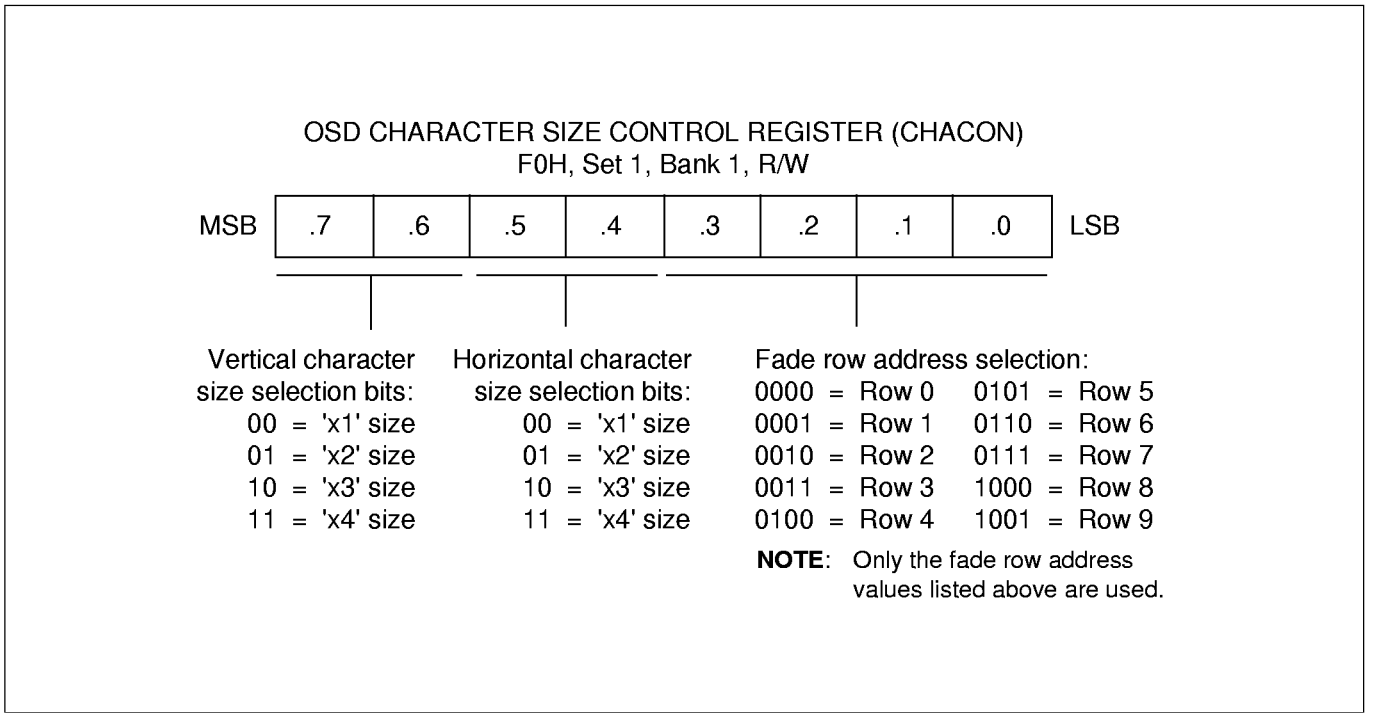


Figure 52. OSD Character Size Control Register (CHACON)

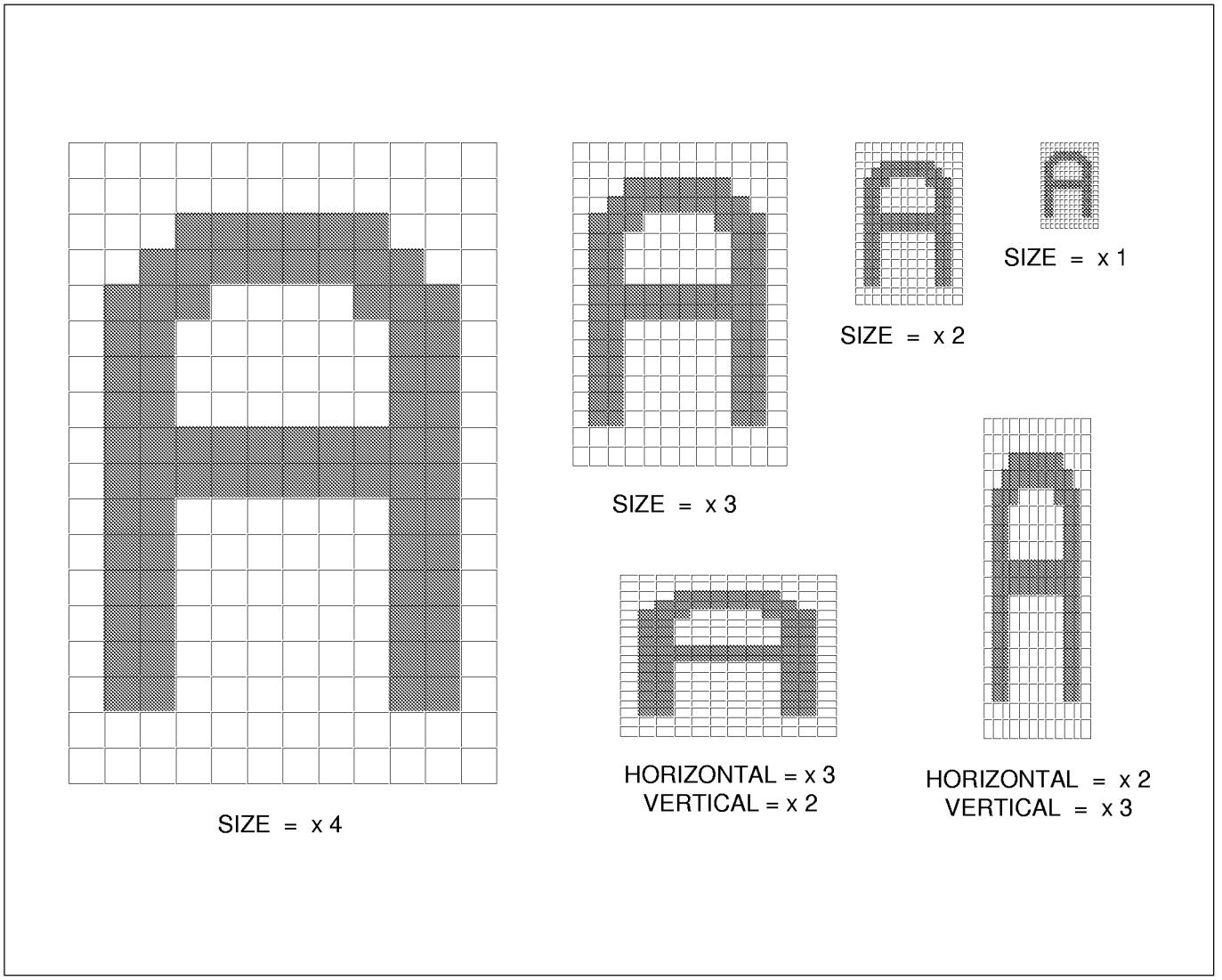


Figure 53. OSD Character Sizing Dimensions

FADE-IN AND FADE-OUT CONTROL REGISTER (FADECON)

The OSD block lets you program fade-in and fade-out displays. A *fade-in display* is one in which a character matrix is displayed incrementally until the complete character has "appeared." A *fade-out display* shows the complete character matrix first and then decrements the matrix line-by-line until the character "disappears" from the display field.

The address of the character display (and the specific line) to be faded-in or faded-out is selected by writing bit values into the CHACON and FADECON registers. Bits 3–0 in the CHACON register specify the 4-bit video RAM address of one of the ten rows (0-9) of the fade display. Bits 0–4 in the FADECON register specify the 5-bit line address (0–17) within the selected row.

Fade direction is controlled by FADECON.5. There are two choices of fade direction: before (FADECON.5 = "0") and after (FADECON.5 = "1"). When you select *fade before*, the character matrix is faded starting with line 0. When you select *fade after*, the matrix is faded starting with line 17. (The line 17 start position is only a suggestion, however, as the fade interval is assignable by software.) To enable the fade function, you set FADECON.6 to "1". (FADECON.7 is not used).

NOTE

To avoid confusion in determining fade row and line addresses in the CHACON and FADECON registers, please note that *line* is a horizontal value that encompasses the entire character display field while *row* is a horizontal value for the character display matrix.

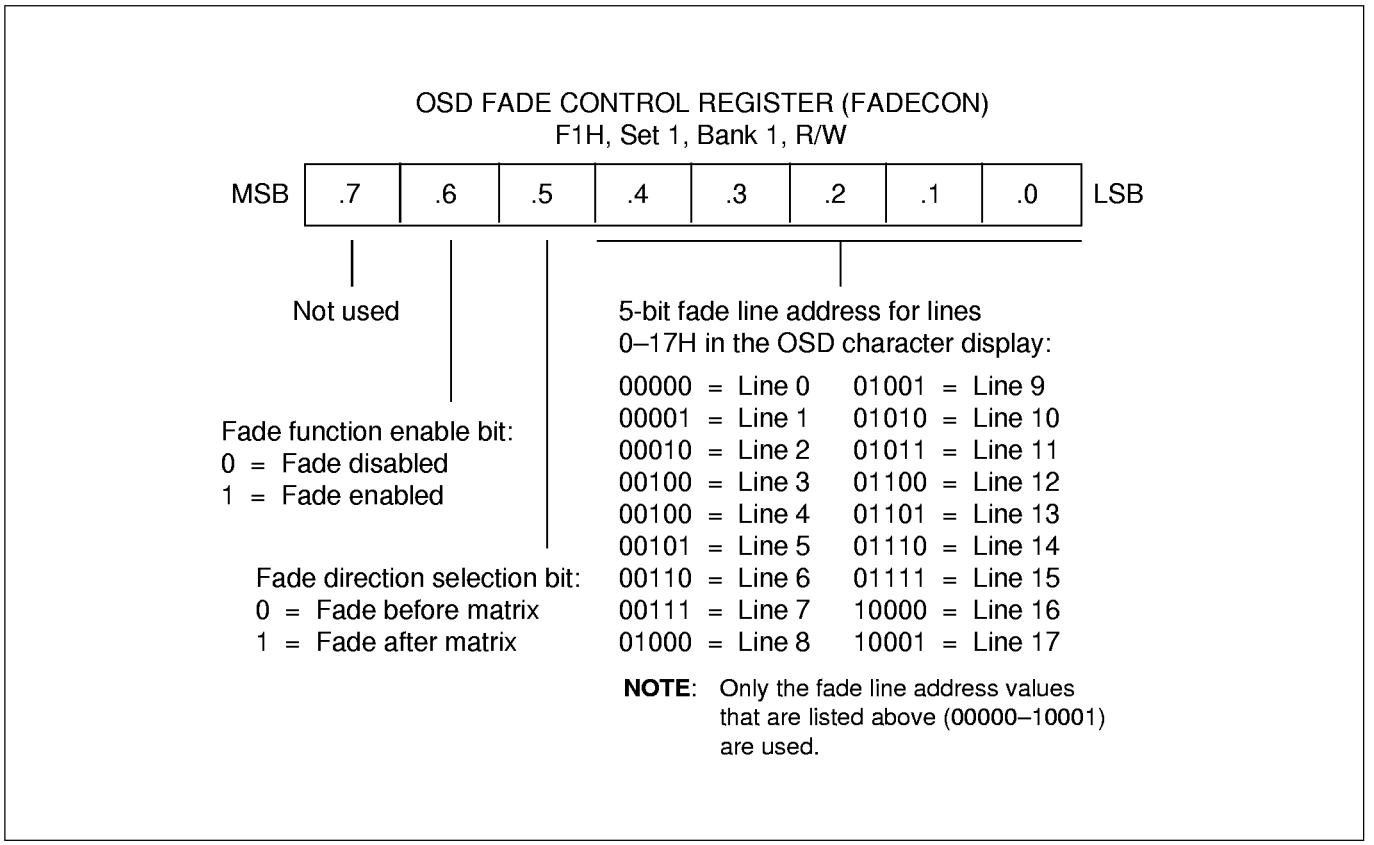


Figure 54. OSD Fade Control Register (FADECON)

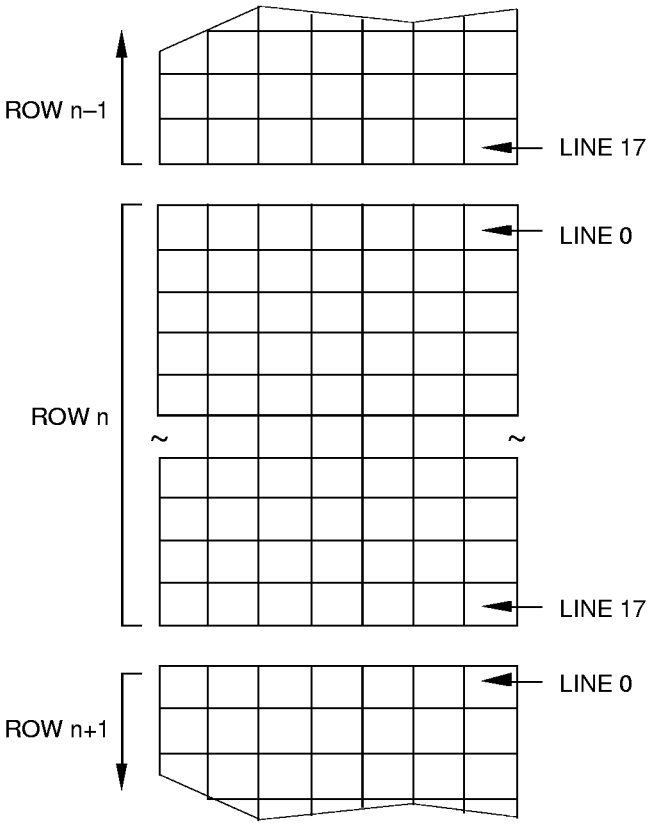


Figure 55. Line and Row Addressing Conventions

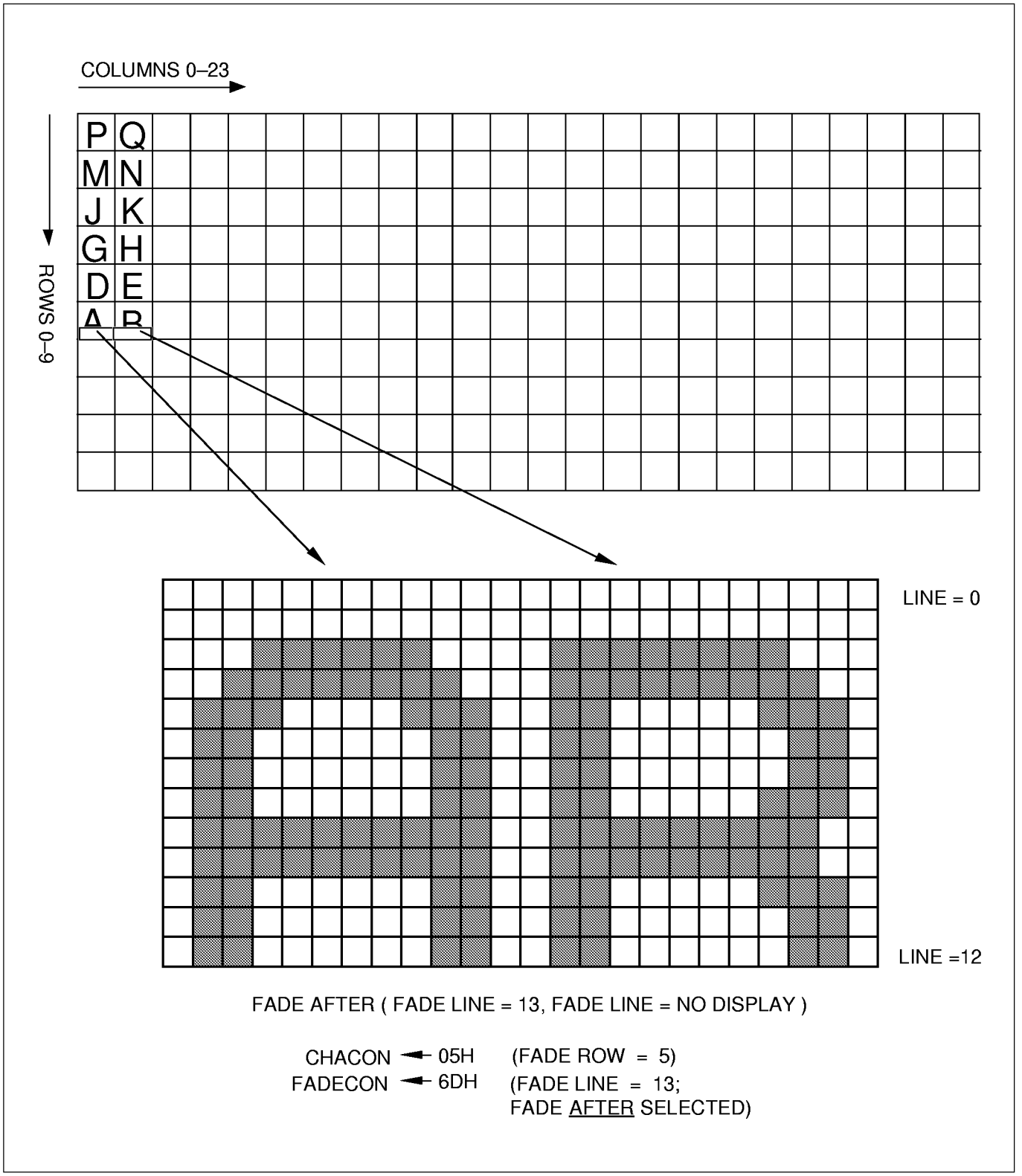


Figure 56. OSD Fade Function Example: Fade After

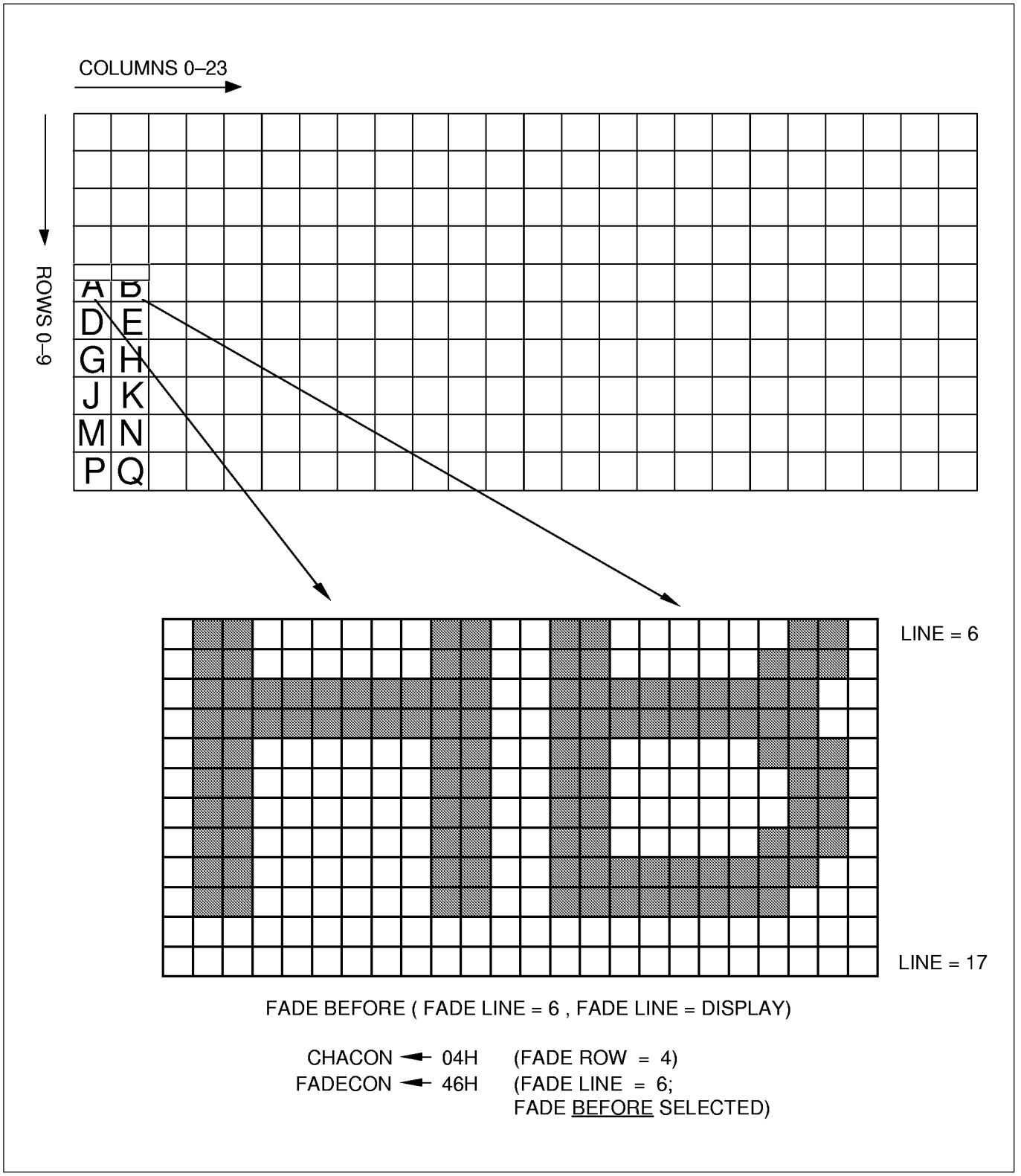


Figure 57. OSD Fade Function Example: Fade Before

DISPLAY POSITION CONTROL

The on-screen display has 240 character display positions. There are 24 horizontal columns and 10 vertical rows. Positions can be numbered sequentially from 0–239 (decimal) or from 0–EF (hexadecimal), as shown in Figures 59 and 60. To control display position, you adjust the top and left margins and the inter-column and inter-row spacing between characters on the screen.

COLUMNS 0–23 →

DECIMAL

ROWS 0–9 ↓

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71
72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	92	94	95
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167
168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199	200	210	202	203	204	205	206	207	208	209	210	211	212	213	214	215
216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239

Figure 58. 240-Byte On-Screen Character Display Map (Decimal)

COLUMNS 0–23 →

HEXADECIMAL

ROWS 0–9 ↓

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	40	41	42	43	44	45	46	47
48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	70	71	72	73	74	75	76	77
78	79	7A	7B	7C	7D	7E	7F	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7
A8	A9	AA	AB	AC	AD	AE	AF	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7
D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF

Figure 59. 240-Byte On-Screen Character Display Map (Hexadecimal)

ROW CONTROL REGISTER (ROWCON)

The row control register, ROWCON, controls the top margin and inter-row spacing. *Top margin* is the distance (in H-sync pulses) of the top row of a character display from the top edge of its display frame. *Inter-row spacing* is the distance (in H-sync pulses) between two rows of displayed characters. The inter-row spacing value you select is applied equally to all rows in the display.

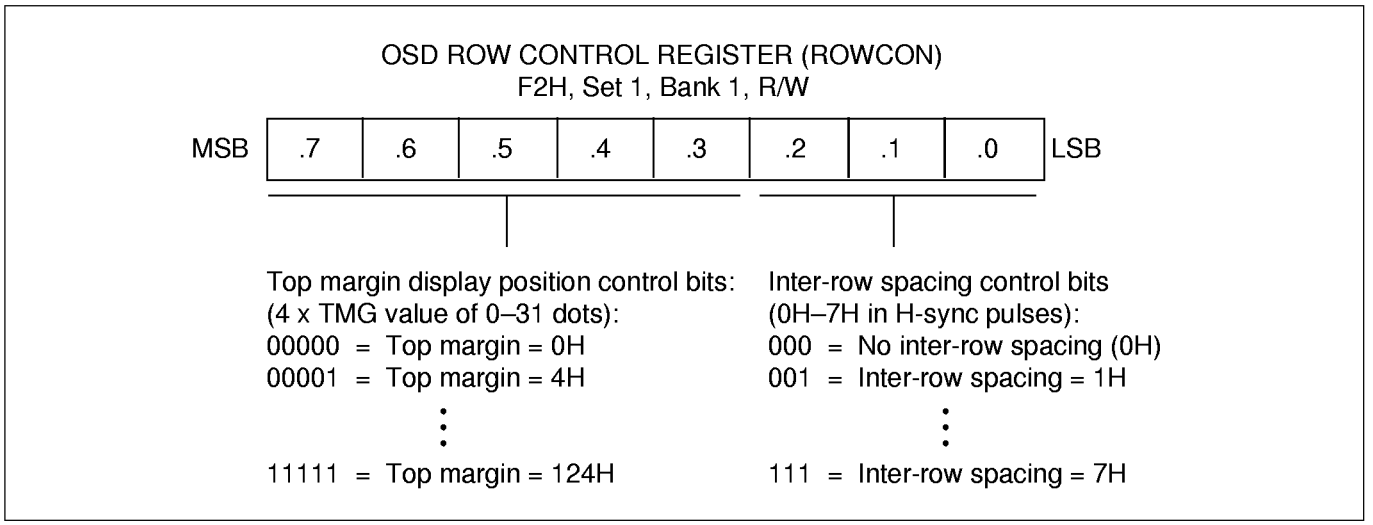


Figure 60. OSD Row Control Register (ROWCON)

COLUMN CONTROL REGISTER (CLMCON)

The column control register, CLMCON, controls the left margin and inter-column spacing. *Left margin* is the distance of the character display from the left edge of the display frame. *Inter-column spacing* is the distance (0–7 dots) of space separating the characters displayed in a row. The inter-column spacing value that you select is applied equally to all columns in the display.

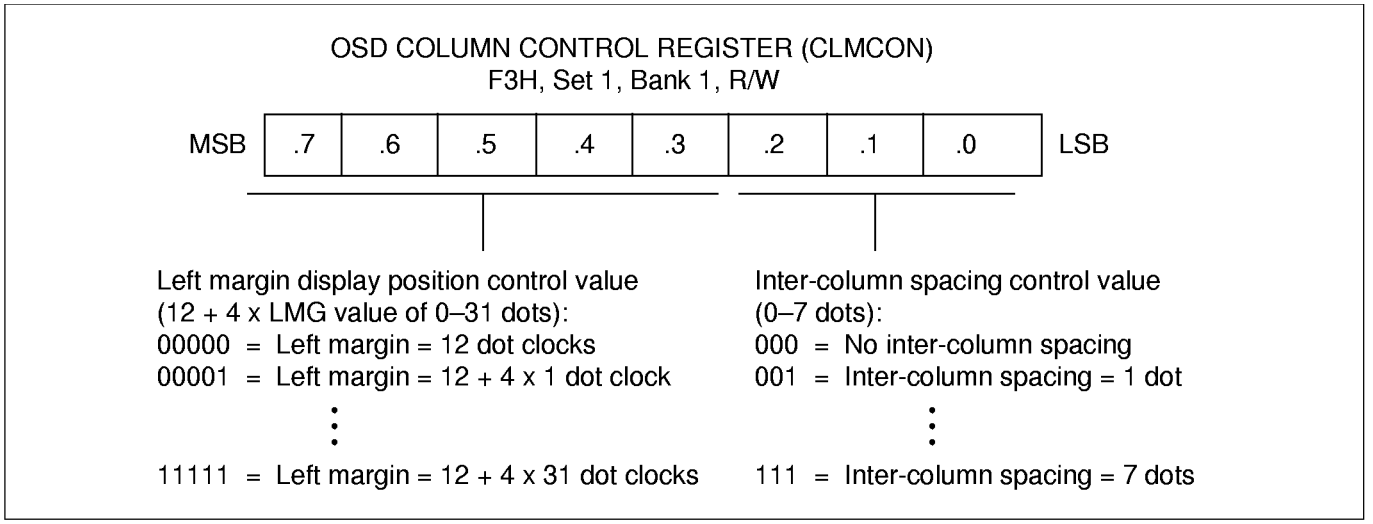


Figure 61. OSD Column Control Register (CLMCON)

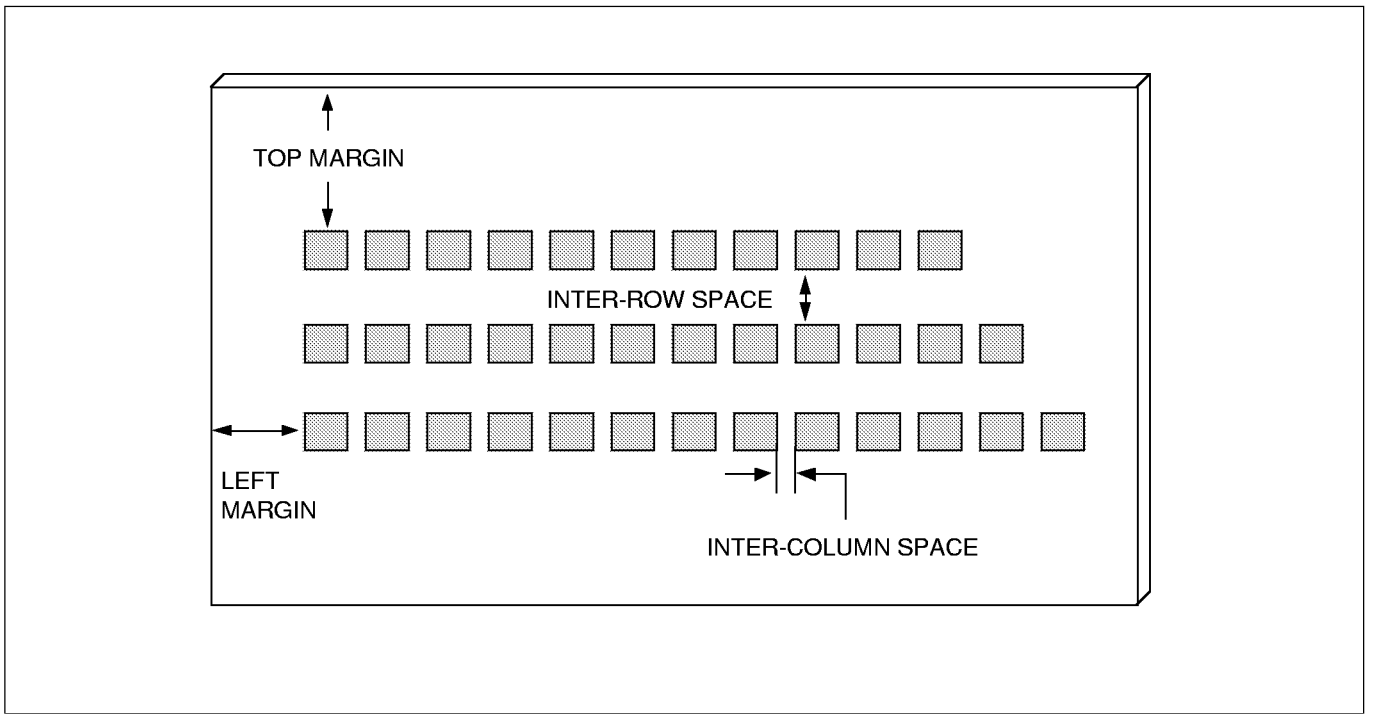


Figure 62. OSD Display Formatting and Spacing Conventions

Calculating Row and Column Spacing

Inter-row spacing and inter-column spacing are controlled by the ROWCON and CLMCON registers. You can select from zero to seven dots of spacing.

For inter-row spacing, the desired spacing value (0–7) is written to bits 0–2 of the ROWCON register. For inter-column spacing, the desired spacing value (0–7) is written to bits 0–2 of the CLMCON register.

Calculating Margin Settings

By writing a value to ROWCON.3–ROWCON.7, you set the top margin at 4 × the top margin dot value (TMG). Because TMG is a 5-bit value, you can select any dot value in the range 0–31.

By writing a value to CLMCON.3–CLMCON.7, you set the left margin at 12 + 4 × the left margin dot value (LMG). Because LMG is a 5-bit value, you can select any dot value in the range 0–31. The zero position for the left margin is always 12 dots.

— Top margin = 4 × (top margin register value) H

— Left margin = 12 + 4 × (left margin register value) dot clock

— Inter-column space = (Register value) dot clock

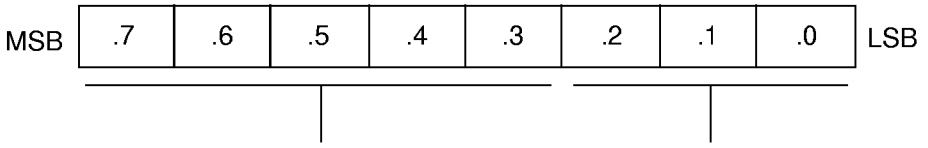
— Inter-row space = (Register value) H

CHARACTER COLOR CONTROL REGISTER (COLBUF)

The color of the character matrix display is controlled by manipulating a 3-bit value in the OSD video RAM. You can modify the *character color selection bits* only by addressing the OSD color buffer register, COLBUF (F7H, set1, bank1). The color selection bits are COLBUF.2 (red), COLBUF.1 (green), and COLBUF.0 (blue). These three bits comprise the RGB value (bits 8, 9, and 10)of the character data stored in the video RAM.

When programming the display RAM values for a character display, you must first load a 3-bit color value into the color buffer. This color setting is automatically appended to each 7-bit character code as it is written to the OSD RAM addresses. If only one COLBUF value is loaded, all characters in the screen display will, of course, be the same color. To change the display color of successive characters, modify the COLBUF value *before* you load the address data for a specific row and column into the video RAM.

OSD CHARACTER COLOR BUFFER (COLBUF)
F7H, Set 1, Bank 1, R/W



Not used

Character color selection bits
(.2 = red, .1 = green, .0 = blue):

.2	.1	.0	Color Displayed
0	0	0	Black character color
0	0	1	Blue character color
0	1	0	Green character color
0	1	1	Cyan character color
1	0	0	Red character color
1	0	1	Magenta character color
1	1	0	Yellow character color
1	1	1	White character color

Figure 63. OSD Character Color Buffer Register (COLBUF)

BACKGROUND COLOR CONTROL

The background color control register, COLCON, lets you select background colors for both the display frame and for characters:

— *Frame background* is the full-screen display field upon which the character display is imposed.

— *Character background* is a color field that surrounds the individual character. To enhance readability, the background is usually a color that contrasts or highlights the characters in a pleasing manner.

BACKGROUND COLOR CONTROL REGISTER (COLCON)

Bit 7 in the COLCON register enables the frame background color display. Bit 3 in the COLCON register enables the character background color display. Bits 6–4 color control bits control the RGB color output for the frame background, respectively, and bits 2–0 control the RGB color output for the character background.

The RGB setting for red is '100B', green is '010B', and blue is '001B'. These three primary colors can also be combined by modifying COLCON bit settings to produce other colors. You can, for example, display a yellow character background color by setting the character color selection bits to '110B'. This setting combines the red and green colors to produce yellow.

To produce a white character or frame background, you would set all of the corresponding color selection bits to "1". To produce a black background, you would set all of the the color selection bits bits to "0". When COLCON.7 or COLCON.3 are "0", the background display is disabled and no color appears.

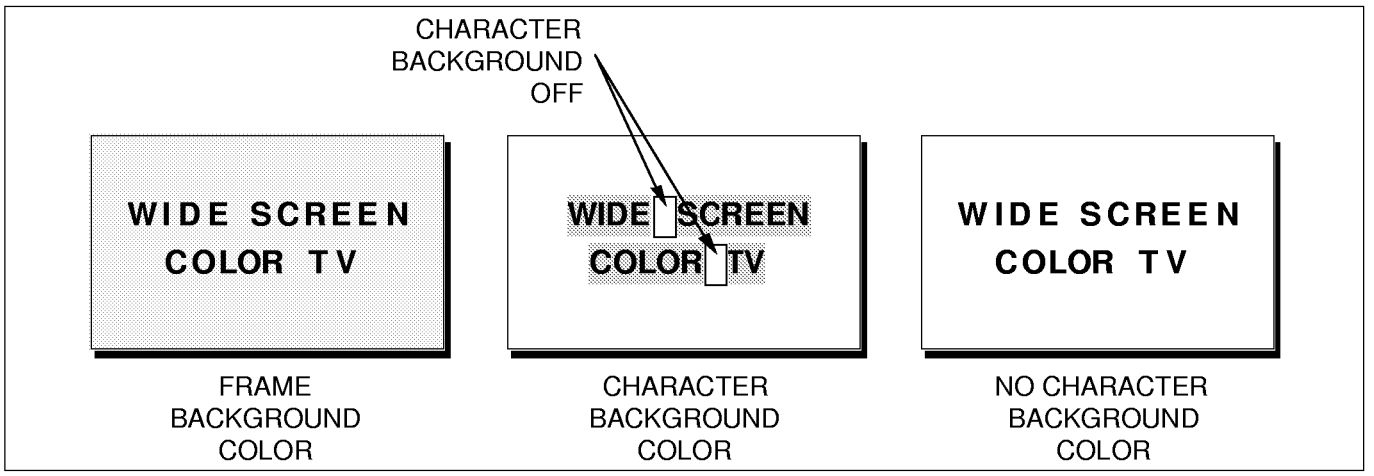


Figure 64. Background Color Display Conventions

OSD BACKGROUND COLOR CONTROL REGISTER (COLCON)
F4H, Set 1, Bank 1, R/W

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

Frame background color enable bit:
0 = Disable frame background color (no color is displayed)
1 = Enable frame background color display

Frame background color selection bits:
(.6 = red, .5 = green, .4 = blue; see table)

Character background color selection bits:
.2 = red, .1 = green, .0 = blue; see table)

Character background color enable bit:
0 = Disable character background color (no color is displayed)
1 = Enable background color display

Bit 7 / Bit 3	R	G	B	Frame and Character Background
0	x	x	x	No background display
1	0	0	0	Black background color
1	0	0	1	Blue background color
1	0	1	0	Green background color
1	0	1	1	Cyan background color
1	1	0	0	Red background color
1	1	0	1	Magenta background color
1	1	1	0	Yellow background color
1	1	1	1	White background color

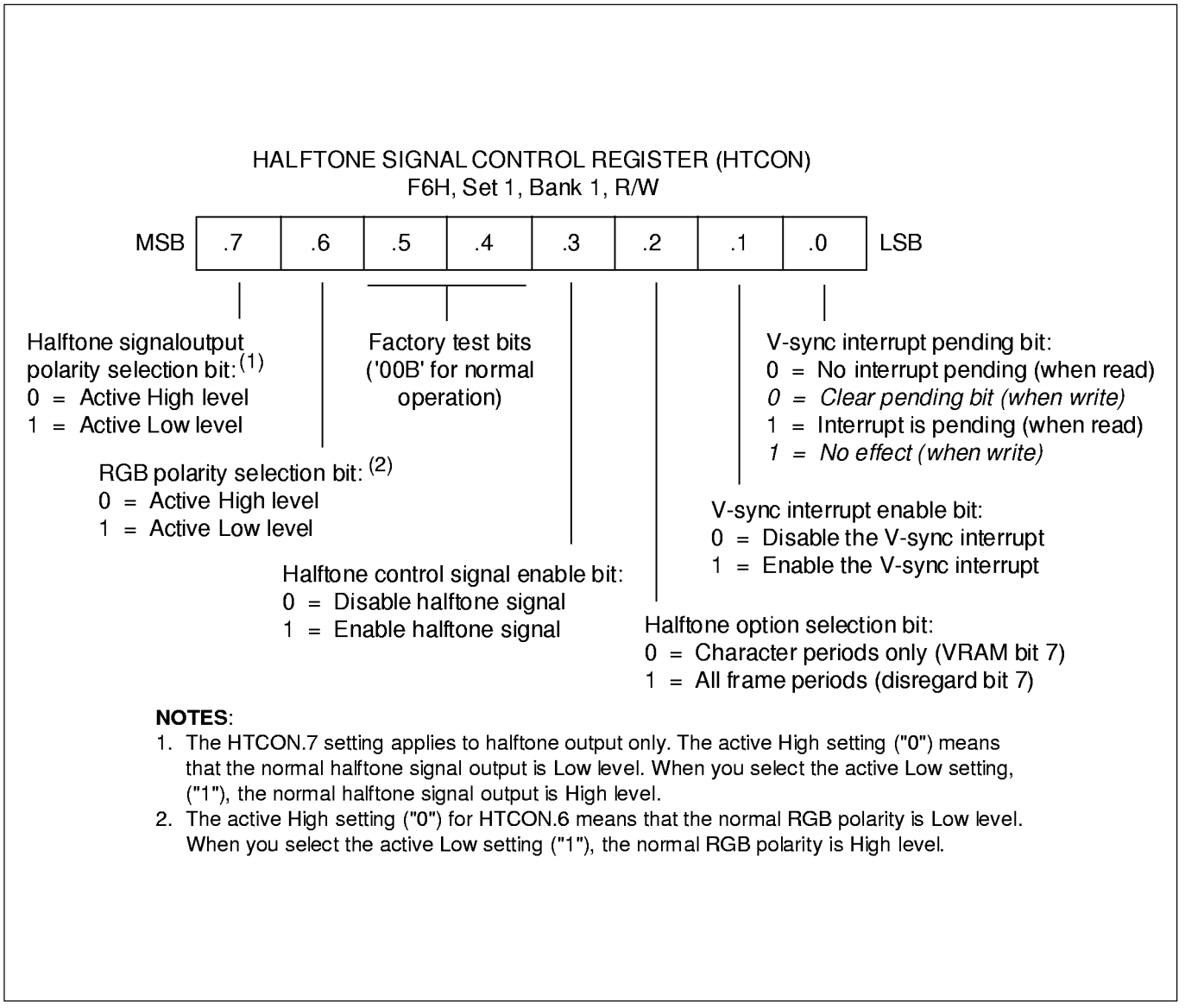
Figure 65. OSD Background Color Control Register (COLCON)

HALFTONE SIGNAL CONTROL REGISTER (HTCON)

The halftone function lets you output halftone control signals to peripherals such as a chroma-IC. You can select halftone output for character periods (as selected by bit 7 in the video RAM) or for frame periods (regardless of the bit 7 setting). The halftone signal control register, HTCON, has the following functions:

- Halftone option selection (character or frame)
- Halftone display enable/disable
- V-sync interrupt enable and pending control
- Polarity selection of RGB and halftone outputs

Bits 4 and 5 of the HTCON register are used for factory testing and should always remain cleared to '00B' during normal operation. Figure 68 shows the affect of the two halftone display options. The timing diagrams assume that the current halftone polarity selection is active High (HTCON.7 = "0"):



NOTES:
1. The HTCON.7 setting applies to halftone output only. The active High setting ("0") means that the normal halftone signal output is Low level. When you select the active Low setting, ("1"), the normal halftone signal output is High level.
2. The active High setting ("0") for HTCON.6 means that the normal RGB polarity is Low level. When you select the active Low setting ("1"), the normal RGB polarity is High level.

Figure 66. Halftone Signal Control Register (HTCON)

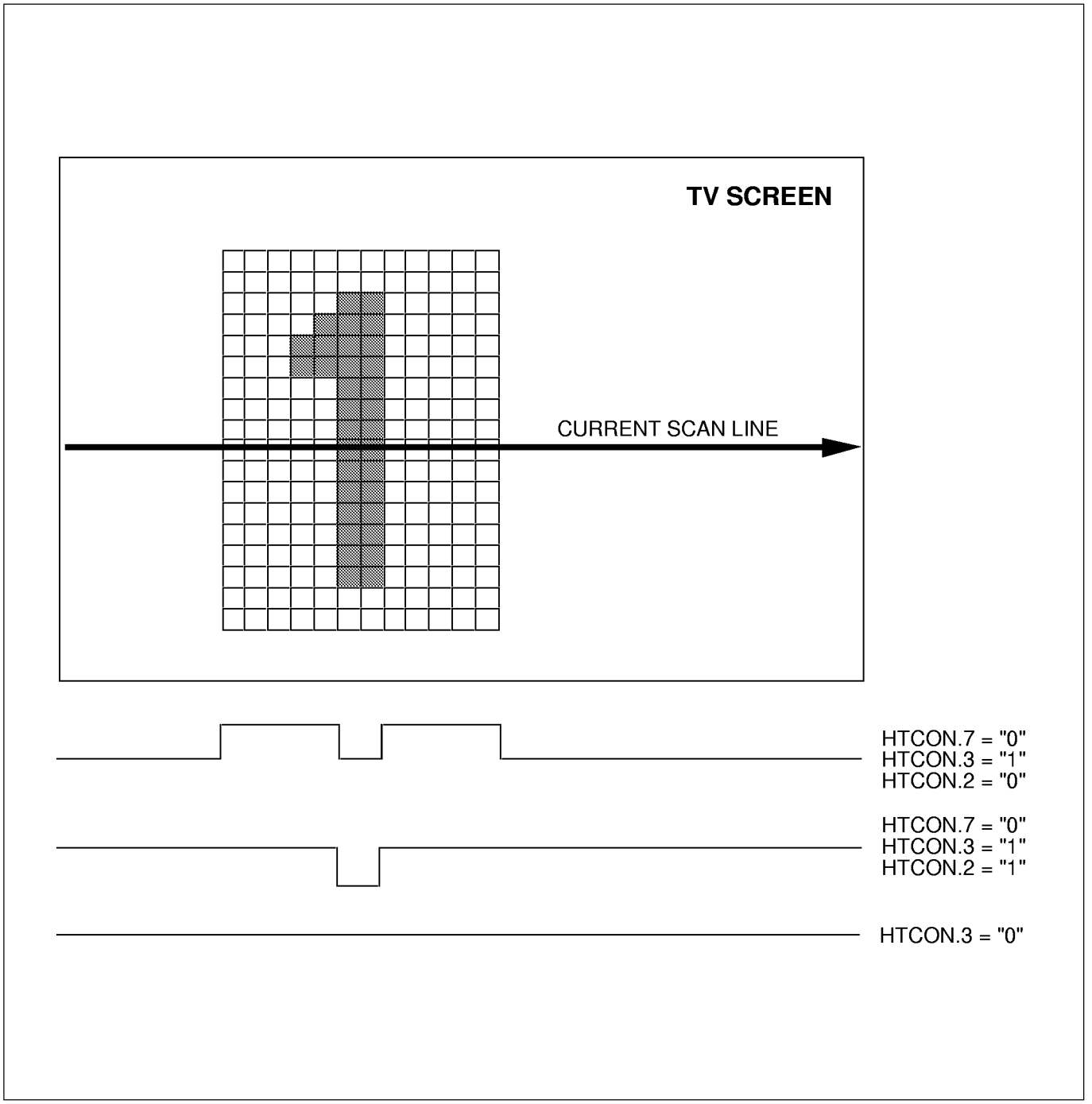


Figure 67. Halftone Control Signal Output Options

☞ PROGRAMMING TIP — Writing Character Code and Color Data to the OSD Video RAM

This example shows how to write character code and color data to the OSD video RAM. The sample program performs the following operations:

1. Write red character 'A' (code 0A, for example) to the video RAM from address 00H to 77H
2. Write green character 'B' (code 0B, for example) to the video RAM from address 78H to 0EFH.

```
          •
          •
          •
          SB1                          ;  Select bank 1
          LD      DSPCON,#0F9H         ;  OSD module on; negative sync trigger is selected
          LD      PP,#11H              ;  Select OSD video RAM page (page 1)
          SRP0    #0C0H                ;  Select common working register area
          LD      COLBUF,#04H          ;  Load color buffer (red color)
          CLR     R0                   ;  Load starting address (00H) to R0
OSDLP1    LD      @R0,#0AH             ;  Write red character A to video RAM address 00H–77H
          INC     R0                   ;          "
          CP      R0,#77H              ;          "
          JP      ULE,OSDLP1           ;          "
          LD      COLBUF,#02H          ;  Load green color code (02H) to the color buffer
OSDLP2    LD      @R0,#0BH             ;  Write green character B to RAM address 78H–0EFH
          INC     R0                   ;          "
          CP      R0,#0EFH             ;          "
          JP      ULE,OSDLP2           ;          "
          SB0                          ;  Select bank 0
          •
          •
          •
```

☞ PROGRAMMING TIP — OSD Fade Function; Line and Row Counters

This example is a continuation of the previous OSD example in which character code and color data were written to the video RAM. Assuming a timer A interrupt interval of 2 milliseconds, the sample program should meet the following specifications:

1. If bit fade (R4.0) is set, then enable the fade function.
2. Interval time between two lines = 20 ms. (The flag 'INTVAL' is set at 20-ms intervals in the timer A service routine.)
3. Fade direction is 'fade after'.

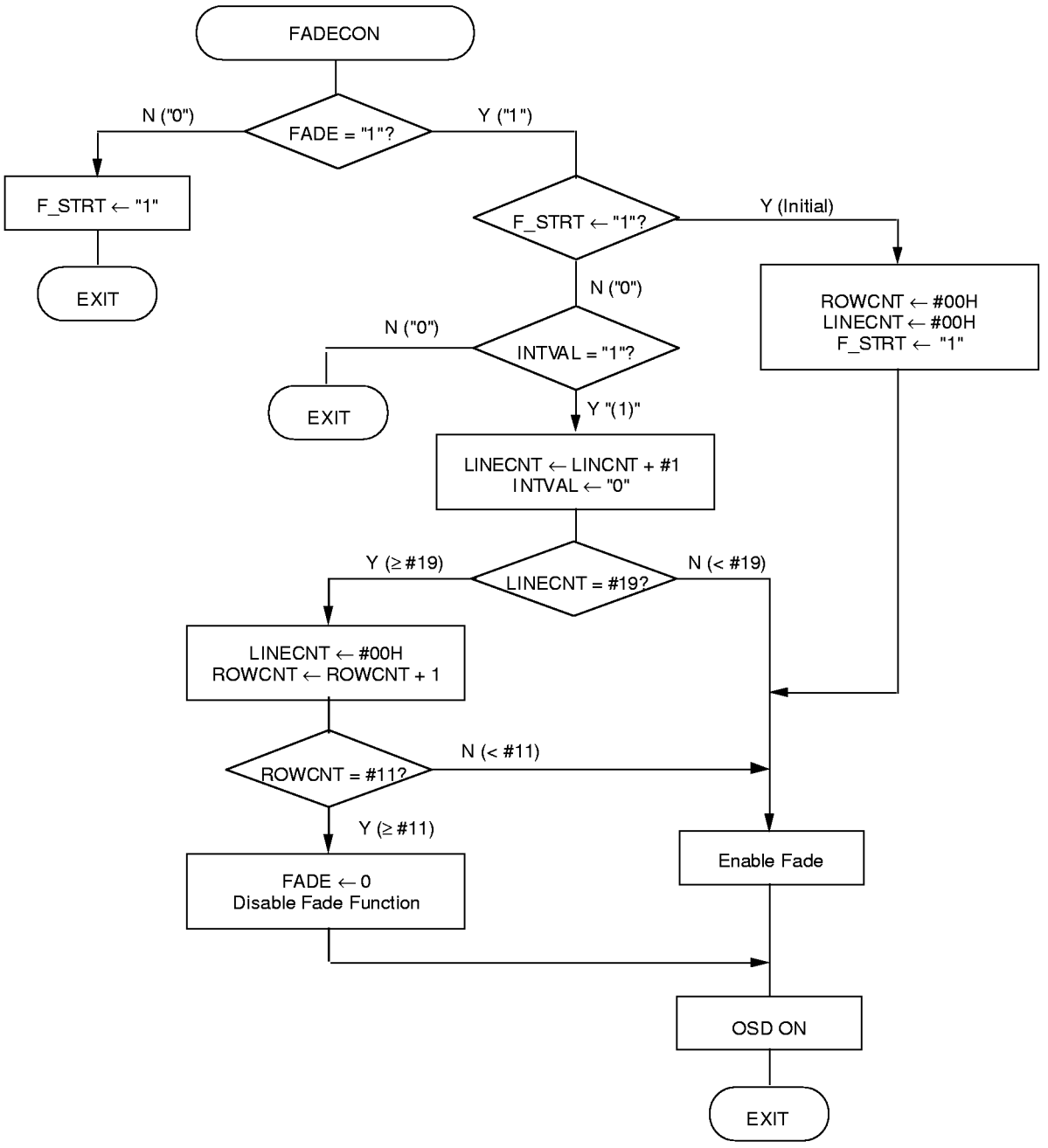


Figure 68. Decision Flowchart for Fade Function Programming Tip

☞ **PROGRAMMING TIP — OSD Fade Function; Line and Row Counters (Continued)**

```
ROWCNT   EQU     6
LINECNT  EQU     7
FADE     EQU     0
F_STRT   EQU     1
INT_CNT  EQU     5
INTVAL   EQU     2
         •
         •
         •
         SB1                         ; Select bank 1
         LD      PP,#11H             ; Select OSD video RAM page (page 1)
         SRP0    #0C0H               ; RP0 ← 0C0H (common working register area)
         BTJRF   EXIT1,R4.FADE       ; If flag FADE = "0", then jump to EXIT1
         BTJRT   FAD1,R4.F_STRT      ; If F_STRT = "1", then jump to FAD1
         BTJRF   EXIT,R4.INTVAL      ; If INTVAL = "1", then jump to EXT
         INC     RLINECNT            ; Line counter ← line counter + 1
         BITR    R4.INTVAL           ; INTVAL ← "0"
         CP      RLINECNT,#13H       ; Line counter ≥ 19?
         JP      ULT,FAD2            ; If line counter < 19, then jump to FAD2
         CLR     RLINECNT            ; Line counter ← "0"
         INC     RROWCNT             ; Row counter ← row counter + 1
         CP      RROWCNT,#0BH        ; Row counter < 11?
         JP      ULT,FAD2            ; If row ≤ 10, then jump to FAD2
         LD      R1,#0F1H            ; If row > 10, then finish the fade function
         LD      R2,@R1
         BITR    R2.6                ; Fade disable

         LD      @R1,R2

FAD3     LD      DSPCON,#0F9H        ; OSD module on
         JR      T,EXIT

FAD1     CLR     RROWCNT             ; Row counter (R6) ← 0H
         CLR     RLINECNT            ; Line counter (Rn) ← 0H
         BITS    R4.F_STRT

FAD2     LD      R2,CHACON           ; R2 ← CHACON
         AND     R2,#0F0H            ; Clear the fade row address
         OR      R2,RROWCNT          ; Load new fade row address to R2
         LD      CHACON,R2           ; CHACON ← R2
         INC     R1                  ; R1 ← 0F1H (fade line address)
         LD      R2,RLINECNT         ; R2 ← new fade line address
         OR      R2,#60H             ; Enable fade function, select fade after
         LD      FADECON,R2
         JR      T,FAD3

EXIT1    BITS    R4.F_STRT

EXIT     SB0                         ; Select bank 0
         •
         •
         •
```

(Continued on next page)

☞ PROGRAMMING TIP — OSD Fade Function; Line and Row Counters (Continued)

```
TAINT   PUSH    PP
        PUSH    RP0
        LD      PP,#11              ;  Select video RAM page (page 1)
        SRP0    #0C0H               ;  RP0 ← 0C0H
        INC     RINT_CNT            ;  Interval counter ← interval counter + 1
        CP      RINT_CNT,#0AH       ;  Interval counter ≤ 10? (Has 20 ms elapsed?)
        JP      ULE,TA1             ;  If yes, then jump to TA1
        CLR     RINT_CNT            ;  20 ms has elapsed, so clear interval counter
        BITS    R4.INTVAL           ;  INTVAL ← "1"

TA1     NOP
        •
        •
        •
        POP     RP0
        POP     PP
        IRET
```

☞ PROGRAMMING TIP — Manipulating OSD Character Colors; Halftone Function

This example is a continuation of the previous OSD examples. Following the second sample program, red character A is in video RAM address 00H–77H and green character B has been written to addresses 78H–0EFH. The program performs the following additional actions:

1. Change the color of character 'A' to white.
2. Change the color of character 'B' to its complementary color.
3. Enable the halftone function for character 'B'.

```
        •
        •
        •
        SB1                         ;  Select bank 1
        LD      PP,#11H             ;  Select video RAM page (page 1)
        SRP0    #0C0H               ;  RP0 ← 0C0H (common working register area)
        LD      COLBUF,#07H         ;  Color buffer ← white color code (07H)
        CLR     R0                  ;  R0 (video RAM address) ← 00H
OSDLP1  LD      @R0,#0AH            ;  Video RAM (00H–77H) ← white 'A'
        INC     R0                  ;          "
        CP      R0,#77H             ;          "
        JP      ULE,OSDLP1          ;          "
        LD      R2,COLBUF           ;  R2 ← color buffer (color of character in address 78H)
        COM     R2                  ;  R2 ← (not R2)
        AND     R2,#07H             ;  Mask out bit 7 through bit 3 of R2
        LD      COLBUF,R2           ;  Color buffer ← complementary color of the character
                                    ;  in address 78H
```

(Continued on next page)

☞ **PROGRAMMING TIP — Manipulating Character Colors; Halftone Function (Continued)**

```
                LD      DSPCON,#0F9H        ;  OSD module on; negative sync trigger selected
                •
                •
                •

halftone        CALL    halftone1           ;  Halftone signal control
                •
                •
                •

halftone1       PUSH    PP                  ;  Stack ← PP
                PUSH    RP0                 ;  Stack ← RP0
                PUSH    FLAGS               ;  Save flags to stack
                SB1                         ;  Select bank 1
                LD      PP,#11H             ;  Page 1 selected
                SRP0    #20H                ;  RP0 ← 20H (working register area)
                CLR     R0                  ;  R0 ← 00H

loop_halftone
                LD      HTCON,#02H          ;  Disable halftone control register
                                            ;  Enable V-sync interrupt
                LD      DSPCON,#0F9H        ;  Enable OSD; select negative sync trigger
                LD      R1,@R0              ;  Video RAM zero address
                INC     R0
                CP      R0,#0EFH            ;  Video RAM end?
                JP      UGT,end_halftone
                tm      R1,#80H             ;  Check bit 7 value
                JR      Z,loop_halftone
                LD      HTCON,#0AH          ;  Enable halftone
                                            ;  Enable V-sync interrupt
                LD      DSPCON,#0FDH        ;  Halftone output mode
                                            ;  Select negative sync trigger
                                            ;  No line is double size

                JP      t,loop_halftone

end_halftone
                POP     FLAGS               ;  Restore flag values from stack
                POP     RP0                 ;  Restore register pointer 0 value
                POP     PP                  ;  Restore page pointer
                RET                         ;  Return
                •
                •
                •
```

☞ PROGRAMMING TIP — OSD Character Size, Fringe, Background Color, and Display Position

This example is a continuation of the previous OSD examples. It performs the following additional actions:

1. Change the character size to horizontal ×3 and vertical ×2.
2. Enable the fringe function.
3. Enable character background color to the complementary color of the character code in address 0EFH of the video RAM.
4. Enable the frame background; select the color cyan.
5. Set top margin to 16H, set inter-row spacing to 1H, set left margin to 24 dots, and set inter-column spacing to three (3) dots.

```
•
•
•
SB1                          ;  Select bank 1
LD      PP,#11H              ;  Select video RAM page (page 1)
SRP0    #0C0H                ;  Select common working register area
LD      CHACON,#60H          ;  Horizontal ×3, vertical ×2 for character size
LD      FADECON,#00H         ;  Disable the fade function
LD      ROWCON,#21H          ;  Top margin ← 16H, inter-row space ← 1H
LD      CLMCON,#1BH          ;  Left margin ← 24 dots, inter-column space ← 3 dots
LD      R3,COLBUF            ;  R3 ← color of the character in address 0EFH
COM     R3                   ;  R3 ← not R3
AND     R3,#07H              ;  Mask out bit 7 through bit 3 of R3
OR      R3,#0B8H             ;  R3 ← cyan frame background color
LD      COLBUF,R3            ;  Enable character and frame background color
LD      DSPCON,#0FBH         ;  Falling edge sync trigger, fringe function on, OSD on
SB0                          ;  Select bank 0
•
•
•
```

☞ PROGRAMMING TIP — Helpful Hints About COLBUF and OSD Character Code 0

When working with the OSD module, please note the somewhat unusual characteristics of the color buffer register (COLBUF) and the OSD character code 0:

— The color buffer register, COLBUF (0F7H, set 1, bank 1) provides a somewhat unusual method for manipulating character color data.

— OSD character code 0 produces a no-display and no-background condition, regardless of the font coding used.

ANALOG-TO-DIGITAL CONVERTER

OVERVIEW

The 4-bit A/D converter (ADC) uses successive approximation logic to convert analog signals at one of the four analog input pins, ADC0–ADC3, to an equivalent 4-bit digital value. The A/D converter has the following components:

— Analog comparator

— D/A converter logic (resistor ladder type)

— ADC control and digital result register (ADCON)

— Analog input pins (ADC0–ADC3)

The KS88C3208/3216 perform 4-bit conversions for one input channel at a time. You select the channel by writing the appropriate 2-bit value to ADCON.4 and ADCON.5. This write operation starts the A/D conversion procedure.

The analog input voltage must lie within the comparator range of V_{DD} to V_{SS}. The conversion process requires six CPU clocks to convert each bit and therefore requires a total of 25 clocks to complete a 4-bit conversion. The digital result can be read from the ADCON register after an additional clock cycle has elapsed.

The digital result is dumped into the lower nibble of the ADCON register. Then, the A/D converter unit goes idle. By polling ADCON.7, the application can detect when a 4-bit conversion has been completed. The contents of ADCON.0–ADCON.3 must then be read out before another conversion starts. Otherwise, the previous result will be overwritten.

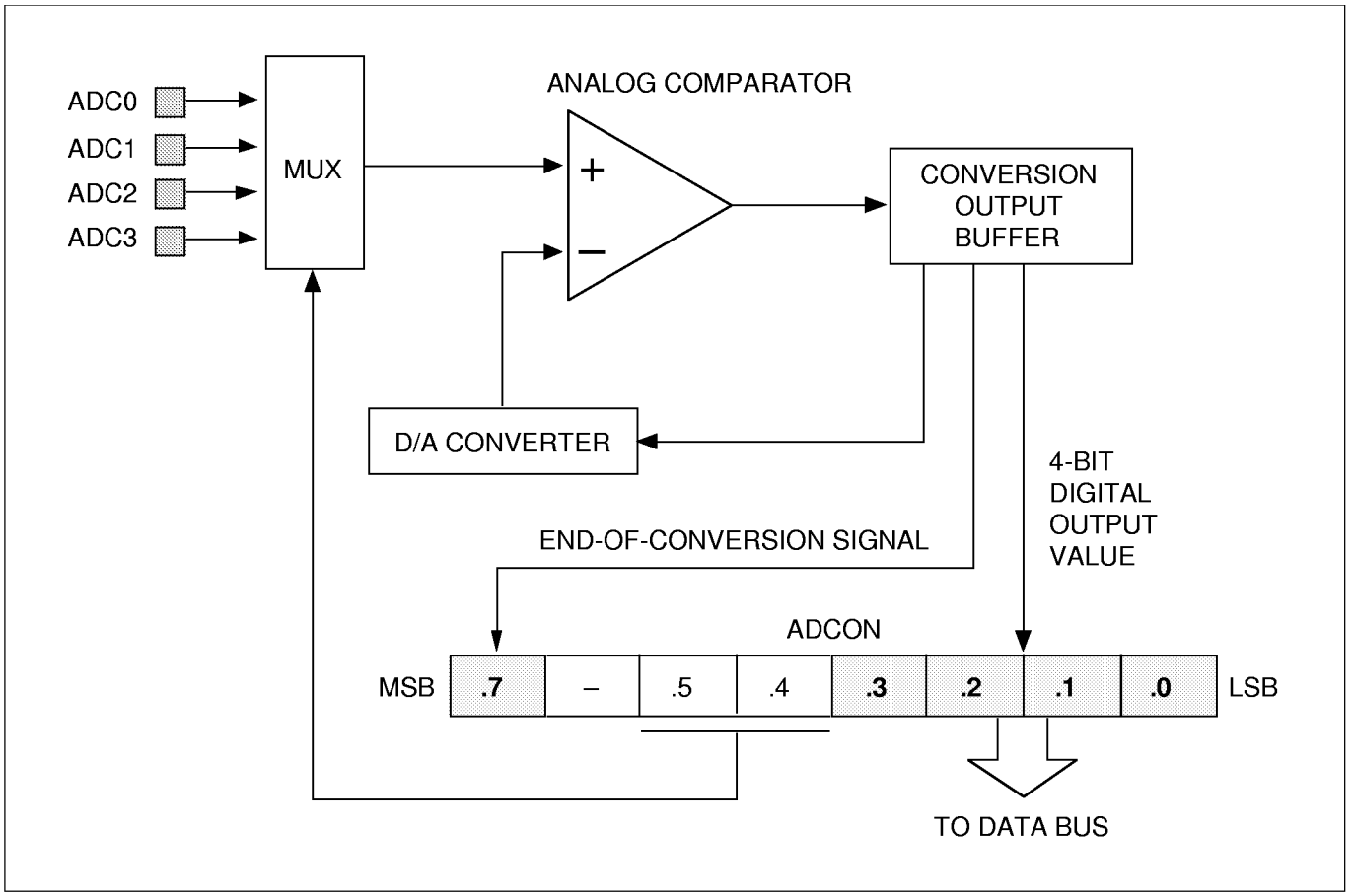


Figure 69. A/D Converter Functional Block Diagram

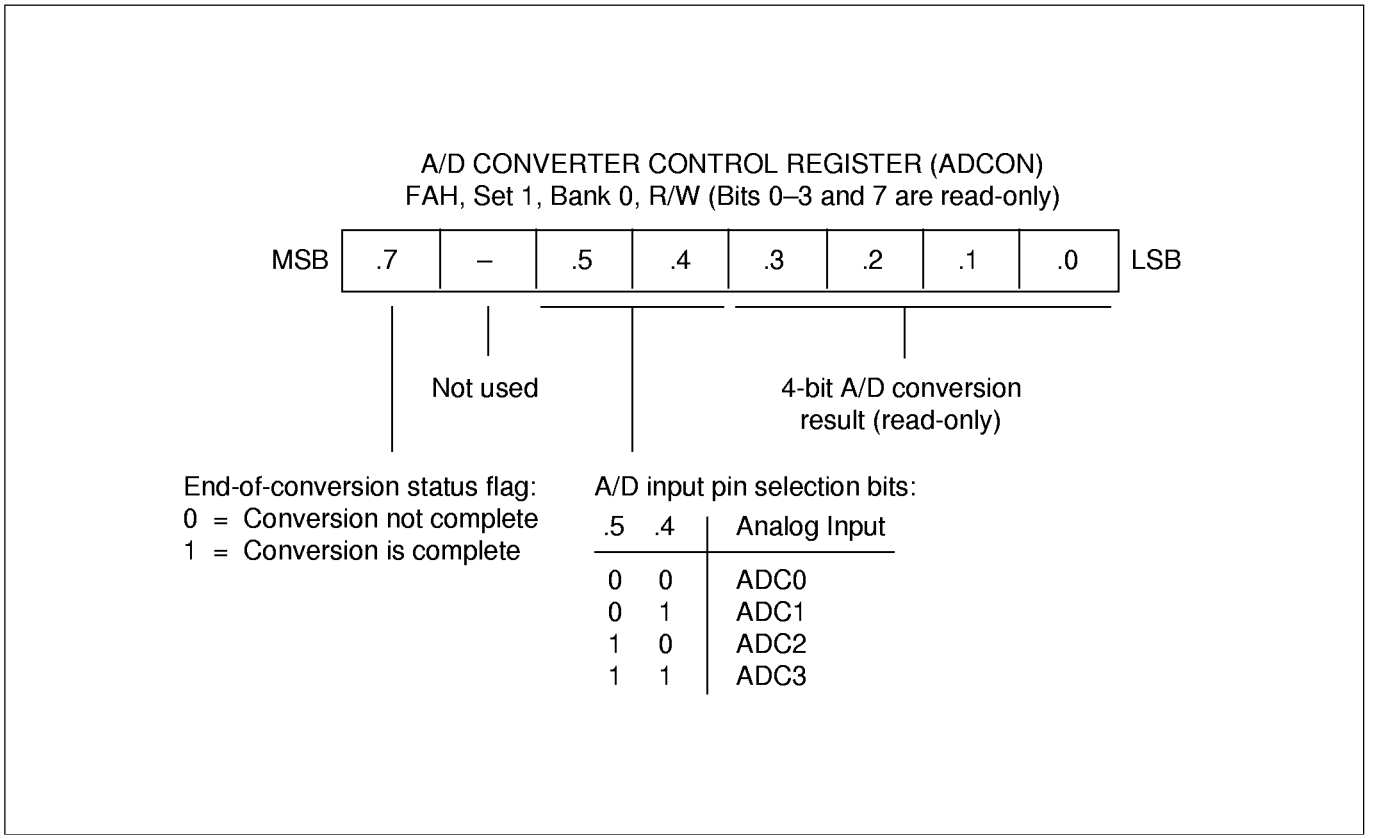


Figure 70. A/D Converter Control Register (ADCON)

INTER NAL A/D CONVERSION PROCEDURE

1. Select the analog input channel by writing the appropriate value to ADCON.4 and ADCON.5. The write operation starts the A/D converter procedure.
2. Analog data is input within the acceptable voltage range of V_{SS} to V_{DD}.
3. After 24 clocks have elapsed, the converted 4-bit digital value is loaded to the lower nibble of the ADCON register and an end-of-conversion signal sets ADCON.7 to "1". The ADC module then enters an idle state.
4. After the 25th clock cycle has elapsed (or when ADCON.7 = "1"), you can read the digital result from the lower nibble of the ADCON register.

INTERNAL REFERENCE VOLTAGE LEVELS

In the ADC block, the analog input voltage level is logically compared to a reference voltage. For the KS88C3208/3216, the analog input level must be within the range V_{SS} to V_{REF}, where $V_{REF} = V_{DD}$.

Different reference voltage levels are generated internally during the analog conversion process for each conversion step. The reference voltage level for the first bit conversion is always 1/2 V_{DD}.

☞ PROGRAMMING TIP — A/D Converter Noise Level and Sampling Frequency

This example shows how to program the A/D converter module to sample specifications. It is assumed that the noise level on the analog channel is relatively low, and that the A/D converter sampling frequency is relatively high. The program performs the following actions:

— Given a low noise level and high sampling frequency, load an A/D conversion value to the register R0 and set bit 7 in R1.

— Enable ADC0.

— Disable ADC1–ADC3.

```
ADFLAG   EQU     7
         •
         •
         •
         CLR     PP              ;  Select page 0
         •
         •
         •
         LD      ADCON,#00H      ;  Select the ADC0 pin and start conversion
         NOP                     ;  Allow sufficient wait time for the conversion;
         NOP                     ;  minimum 25 cycles are required.
         NOP
         NOP
         NOP
         NOP
         NOP
         LD      R0,ADCON        ;  R0 ← conversion value
         AND     R0,#0FH         ;  Need lower nibble only (mask upper nibble)
         BITS    R1.ADFLAG       ;  Set R1.7
         •
         •
         •
```

☞ PROGRAMMING TIP — A/D Converter Noise Level and Sampling Frequency (Continued)

In some systems, the noise level may be quite high or the A/D conversion sampling frequency may be low. In such cases, you could use the following method to filter out noise: Perform each A/D conversion three times, discard the maximum and minimum values that you obtain, and use the median value as the result. The following program code uses this method to perform the same operation as the first A/D converter sample program:

— Assume a high noise level and a low sampling frequency.

— Use median value of three conversions as the result.

— Load an A/D conversion value to the register R0 and set bit 7 in R1.

```
AD0       EQU     0
AD1       EQU     1
AD2       EQU     2
AVE       EQU     0
MIN       EQU     1
MAX       EQU     2
TEMP      EQU     3
ADFLAG    EQU     7
          •
          •
          •
          CLR     PP                  ;  Select page 0
          •
          •
          •
          LD      RTEMP,#03H          ;  R3 ← 03H
ADLOOP    LD      ADCON,#00H          ;  Select ADC0 pin and start conversion
          NOP                         ;  Wait minimum 25 cycles for conversion
          NOP                         ;
          NOP                         ;
          NOP                         ;
          NOP                         ;
          NOP                         ;
          LD      RAD0,RAD1           ;  Save conversion result to R0–R2
          LD      RAD1,RAD2           ;  (first conversion result will be in R0, the second result is
                                      ;  in R1, and the third result is in R2)
          LD      RAD2,ADCON          ;
          AND     RAD2,#0FH           ;  We only need the lower nibble
                                      ;
          DJNZ    RTEMP,ADLOOP        ;  Conversion completed three times? If no, return to loop
          CP      RMAX,RMIN           ;
          JP      UGE,JAD1            ;  If MAX ≥ MIN, jump to JAD1
          LD      RTEMP,RMIN          ;  If MAX < MIN, exchange values (MAX ↔ MIN)
          LD      RMIN,RMAX           ;          "
          LD      RMAX,RTEMP          ;          "
          •
          •
          •
```

(Continued on next page)

☞ **PROGRAMMING TIP — A/D Converter Noise Level and Sampling Frequency (Continued)**

```
JAD1    CP      RMAX,RAVE       ;
        JP      UGE,JAD2        ;  If MAX ≥ AVE, go to JAD2
        LD      RTEMP,RMAX      ;  If MAX < AVE, exchange MAX and AVE
        LD      RMAX,RAVE       ;          "
        LD      RAVE,RTEMP      ;          "

JAD2    CP      RMIN,RAVE       ;
        JP      ULE,JAD3        ;  If MIN ≤ AVE, go to JAD3
        LD      RTEMP,RMIN      ;  If MIN > AVE, exchange MIN and AVE
        LD      RMIN,RAVE       ;          "
        LD      RAVE,RTEMP      ;          "

JAD3    BITS    R1.ADFLAG       ;  Set ADFLAG
        •
        •
        •
```

I²C-BUS INTERFACE

OVERVIEW

The KS88C3208/3216 microcontroller include support for the I²C-bus interface. Two pairs of serial data (SDA) and serial clock (SCL) lines are provided to carry information between the master and peripherals that are connected to the bus.

The KS88C3208/3216 behave as a single master and can operate as a receiver or a transmitter of serial data to/from slave devices. Because the KS88-series chip is the master, it initiates data transfers and also generates the clock signal to permit the transfer. The master also terminates each transfer of data over the I²C-bus. The multi-master and arbitration functions of the standard I²C-bus are not supported in the KS88C3208/3216 implementation.

There are two pairs of serial data I/O and clock output pins: SDA0/SCL0 and SDA1/SCL1. Using the IICCON control register, you can select one pair or the other. This feature gives you additional flexibility in supporting various types of applications. SDA0 and SDA1 are bi-directional and SCL0 and SCL1 are output-only. When the bus is free, both lines are at High level. Data on the SDA0/1 line must remain stable when the clock period is High level. The state of the data line changes only when the clock signal on the SCL0/1 line is Low level.

A High-to-Low transition of SDA0/1 signals a START condition. A Low-to-High transition of SDA0/1 while SCL0/1 is High level signals a STOP condition. START and STOP conditions are always generated by the master. A 7-bit value in the first incoming byte after the START condition is met determines which slave will be selected by the master.

Every data byte put on the SDA0/1 line must be eight bits in length. The number of bytes that you can send or receive per transfer is unrestricted. Data is transferred with the most significant bit (MSB) first. Each byte must be followed by an acknowledge bit.

I²C-BUS PRESCALER (IICPS)

The programmable 8-bit prescaler for the I²C-bus shift clock, IICPS, is located at EFH in set 1, bank 0. To determine the shift clock frequency (speed), you use this formula:

Shift clock frequency = CPU clock frequency /16 ÷ (Prescaler value + 1)

The shift clock may be "stretched" if a slow slave device holds the clock for clock synchronization.

I²C-BUS SHIFT REGISTER (IIC)

The data shift register for the I²C-bus interface, called 'IIC', is located at EDH in set 1, bank 0. It is read/write addressable. When you read or write the IIC buffer, the IIC interrupt pending bit (IICCON.0) is cleared. After a reset, all shift register values are undetermined.

I²C-BUS INTERRUPT

The I²C-bus interrupt is assigned level IRQ4 in the interrupt structure. Its vector address is D0H. The enable bit for the I²C-bus interrupt is IICCON.1. Software polls the IIC interrupt pending bit IICCON.0 in order to determine when a receive or transmit procedure is in-progress or has been completed. When the CPU acknowledges the interrupt request, the interrupt service routine must also clear the pending bit, IICCON.0, to "0".

I²C-BUS CONTROL REGISTER (IICCON)

The I²C-bus control register, IICCON, is located at EEH in set 1, bank 0. Although the IICCON register is read-write addressable, two bits are read-only: bit 2 (the last-received bit flag) and bit 6 (the I²C-bus busy status flag). A reset clears IICCON to '00H'.

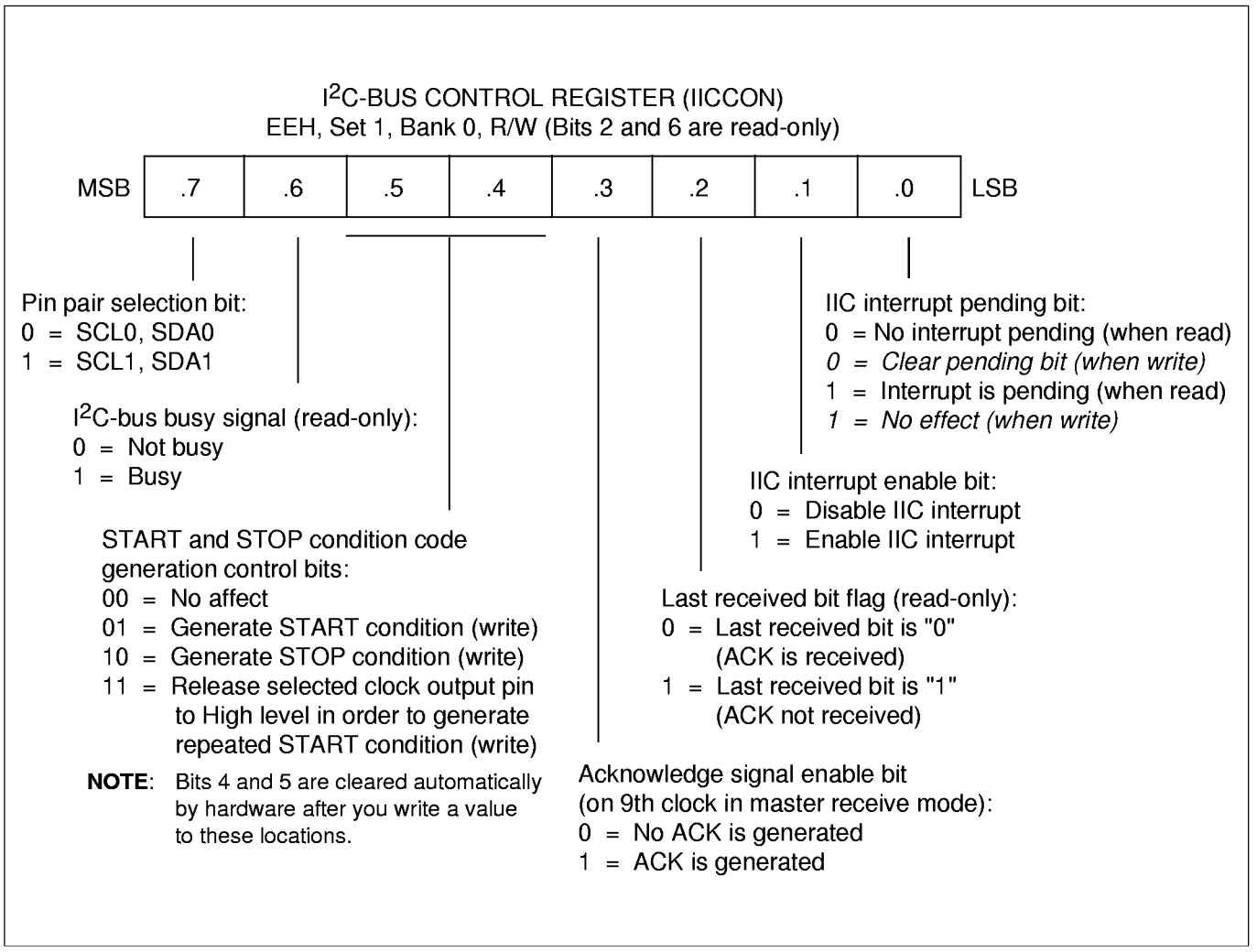


Figure 71. I2C Bus Control Register (IICCON)

☞ PROGRAMMING TIP — Programming the I²C-Bus Interface

This example shows how to program the I²C-bus module to sample specifications. The program parameters are as follows:

— Main crystal oscillation frequency is 6 MHz
— I²C clock frequency is 94 kHz (I²C specification: maximum 100 kHz)
— The following registers are assigned for the program values:

```
Register 60H    SLAVE      ;  Slave address data
Register 61H    SUBADDR    ;  Sub-addressor data
Register 62H    IICDATA    ;  I2C-bus data
Register 63H    IIC_CON    ;  I2C-bus condition code control register
```

Additional address information:

— Slave address: #0A0H (EEPROM-type)
— Sub address: #10H (data address pointer)
— IIC data: #20H (I²C data)

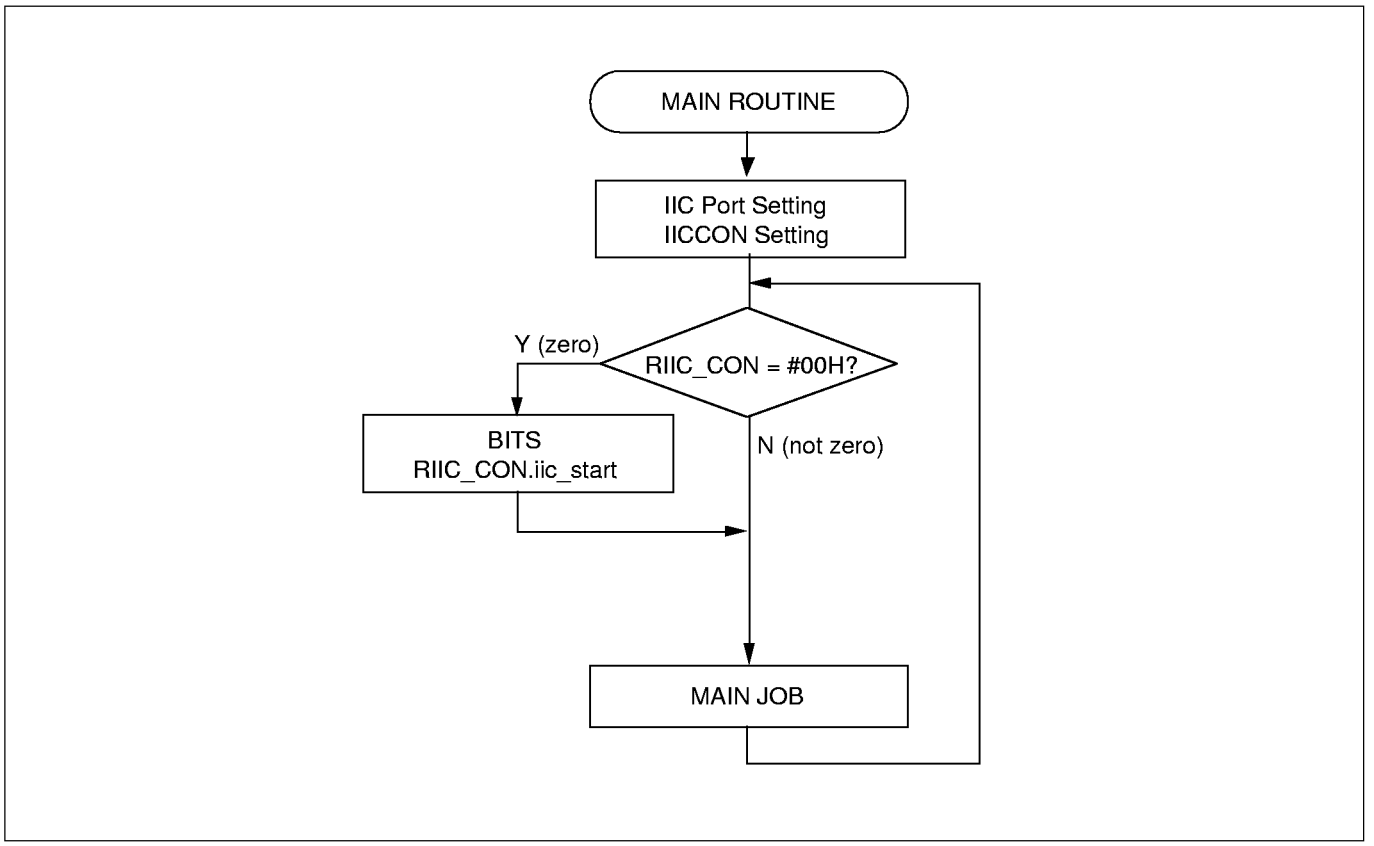


Figure 72. Decision Flowchart for I²C-Bus Programming Tip (Main Routine)

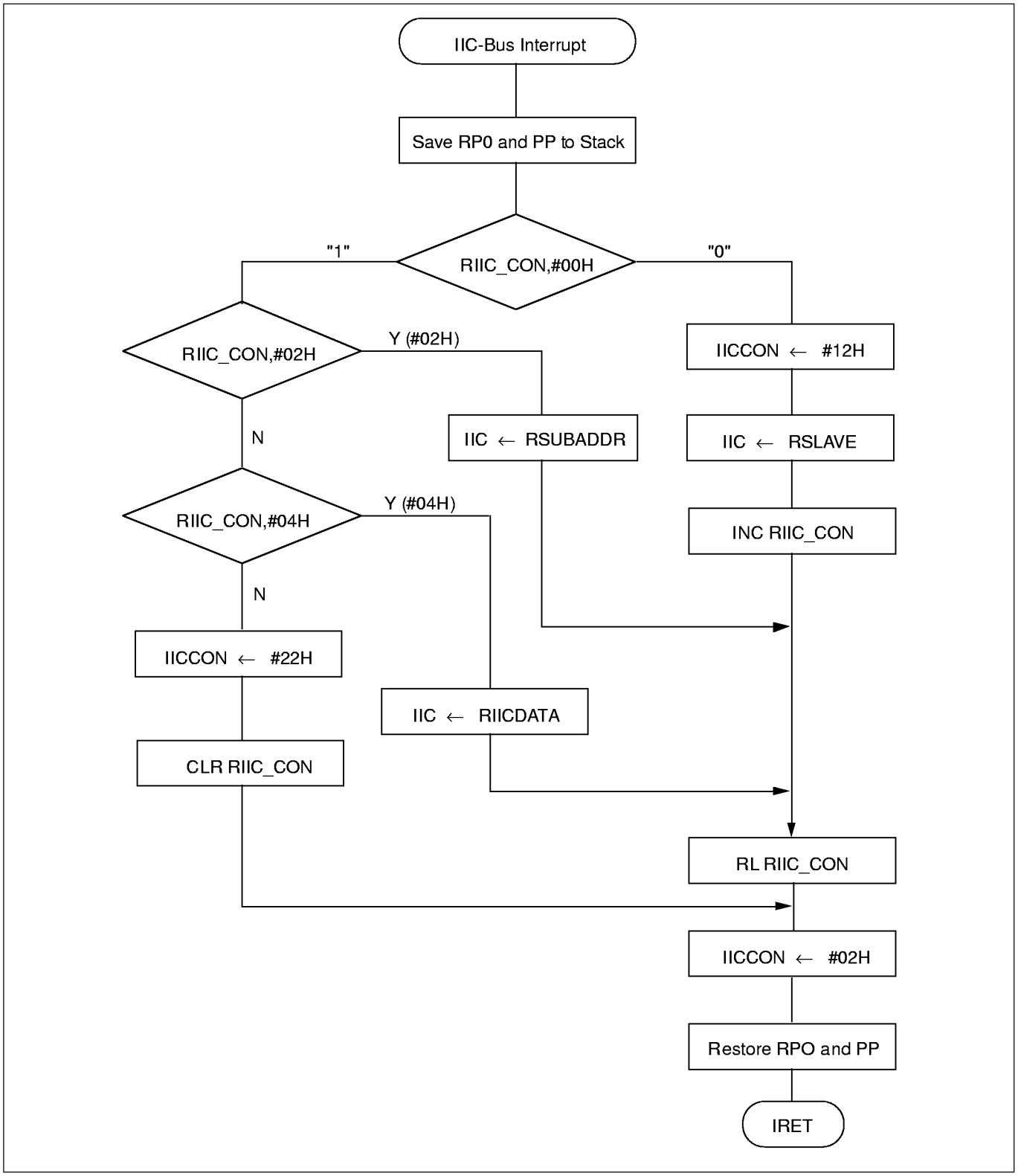


Figure 73. Decision Flowchart for I²C-Bus Programming Tip (I²C-Bus Interrupt)

☞ PROGRAMMING TIP — Programming the I²C-Bus Interface (Continued)

```
                •
                •
                •
                SLAVE       EQU    0
                SUBADDR     EQU    1
                IICDATA     EQU    2
                IIC_CON     EQU    3
                iic_start   equ    1
                •
                •
                •
                CLR         PP                      ;  Select page 0
                LD          P2CONL,#0F0H            ;  Set P2.2 and P2.3 to IIC-bus data/clock mode
                                                    ;  SCLo, SDA0 mode enable
                LD          P2CONH,#00H             ;  P2.2 and P2.3 are open-drain outputs
                                                    ;  SCL1, SDA1 mode disable
                LD          IICCON,#02H             ;  Enable the IIC-bus interrupt
exec_main:
                SRP0        #60H                    ;  RP0 ← 60H
                CP          RIIC_CON,#00H           ;  Down counter = "0"
                JP          NE,MAIN                 ;  IF not zero, then jump to MAIN
                BITS        RIIC_CON.iic_start      ;  Set "iic_start" bit
                LD          RSLAVE,#0A0H            ;  Slave address ← #0A0H
                LD          RSUBADDR,#10H           ;  Sub address ← #10H
                LD          RIICDATA,#20H           ;  Slave IIC data ← #20H
MAIN:                                               ;  Other job...
                •
                •
                •
                JP          T,exec_main             ;  For looping
                •
                •
                •
IICINT          PUSH        PP                      ;  Save the page pointer
                PUSH        RP0                     ;  Save register pointer 0
                SB0                                 ;  Select bank 0
                SRP0        #60H                    ;  RP0 ← 60H
                CP          RIIC_CON,#00H           ;  R3 (IIC condition code control register) = "#00H"?
                JP          NE,iic_control1
                LD          IICCON,#12H             ;  IIC start condition
                LD          IIC,RSLAVE              ;  Slave address
                INC         RIIC_CON
end_iicint:
                RL          RIIC_CON

end_iicint1:
                LD          IICCON,#02H             ;  IIC generate condition
                POP         RP0                     ;  Restore register pointer 0
                POP         PP                      ;  Restore page pointer
                IRET
```

(Continued on next page)

☞ PROGRAMMING TIP — Programming the I²C-Bus Interface (Continued)

```
iic_control1:
            CP      RIIC_CON,#02H
            JP      NE,iic_control2
            LD      IIC,RSUBADDR        ;  Slave sub address
            JP      T,end_iicint

iic_control2:
            CP      RIIC_CON,#04H
            JP      NE,iic_control3
            LD      IIC,RIICDATA        ;  Slave data
            JP      T,end_iicint

iic_control3:
            LD      IICCON,#22H         ;  IIC stop condition
            CLR     RIIC_CON            ;  Clear IIC_CON data
            JP      T,end_iicint1
```

ELECTRICAL DATA

Table 17. Absolute Maximum Ratings

(T_A = 25°C)

Parameter	Symbol	Conditions	Rating	Unit
Supply Voltage	V_{DD}	–	– 0.3 to + 7.0	V
Input Voltage	V_{I1}	P0.4–P0.7, P2.7, P3.0–P3.5 (open-drain)	– 0.3 to + 10	V
	V_{I2}	All port pins except V_{I1}	– 0.3 to V_{DD} + 0.3	
Output Voltage	V_O	All output pins	– 0.3 to V_{DD} + 0.3	V
Output Current High	I_{OH}	One I/O pin active	– 18	mA
		All I/O pins active	– 60	
Output Current Low	I_{OL}	One I/O pin active	+ 30	mA
		Total pin current for port 3	+ 100	
		Total pin current for ports 0, 1, and 2	+ 100	
Operating Temperature	T_A	–	– 25 to + 85	°C
Storage Temperature	T_{STG}	–	– 65 to + 150	°C

Table 18. D.C. Electrical Characteristics

(T_A = – 25°C to + 85°C, V_{DD} = 4.5 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input High Voltage	V_{IH1}	All input pins	0.8 V_{DD}	–	V_{DD}	V
Input Low Voltage	V_{IL1}	All input pins except V_{IL2}	–	–	0.2 V_{DD}	V
	V_{IL2}	X_{IN}, X_{OUT} OSC_{IN}, OSC_{OUT}			0.15 V_{DD}	
Output High Voltage	V_{OH}	I_{OH} = – 1 mA P1, P2.0–P2.6 R, G, B, Vblank	V_{DD} – 1.0	–	–	V
Output Low Voltage	V_{OL1}	I_{OL} = 4 mA P0.0–P0.3, P1.4–P1.7	–	–	0.4	V
	V_{OL2}	I_{OL} = 15 mA P0.4–P0.7, P1.0–P1.2	–	–	1.0	
	V_{OL3}	I_{OL} = 2 mA P2.0–P2.5	–	–	0.4	
	V_{OL4}	I_{OL} = 1 mA R, G, B, Vblank, P1.0, OSDHT (P1.3), P2.0–P2.7, P3.0–P3.5	–	–	0.4	V

Table 19. D.C. Electrical Characteristics (Continued)

(T_A = – 25°C to + 85°C, V_{DD} = 4.5 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input High Leakage Current	I_{LIH1}	$V_{IN} = V_{DD}$ All input pins except I_{LIH2} and I_{LIH3}	–	–	3	µA
	I_{LIH2}	$V_{IN} = V_{DD}$, X_{OUT}, OSC_{IN}, OSC_{OUT}			20	
	I_{LIH3}	$V_{IN} = V_{DD}$, X_{IN} only	2.5	19	30	
Input Low Leakage Current	I_{LIL1}	V_{IN} = 0 V All input pins except I_{LIL2}, I_{LIL3}, and RESET	–	–	– 3	µA
	I_{LIL2}	V_{IN} = 0 V, X_{OUT} OSC_{IN}, OSC_{OUT}			– 20	
	I_{LIL3}	V_{IN} = 0 V, X_{IN} only	– 2.5	– 19	– 30	
Output High Leakage Current	I_{LOH1}	$V_{OUT} = V_{DD}$ All output pins except I_{LOH2}	–	–	3	µA
	I_{LOH2}	V_{OUT} = 10 V P0.4–P0.7 and port 3			20	
Output Low Leakage Current	I_{LOL}	V_{OUT} = 0 V All output pins	–	–	– 3	µA
Pull-up Resistor	R_{L1}	V_{IN} = 0 V; V_{DD} = 4.5 V to 5.5 V P1, P2.0–P2.6	30	40	70	KΩ
Supply Current (see Note)	I_{DD1}	Normal mode; V_{DD} = 4.5 V to 5.5 V 8-MHz CPU clock	–	21	35	mA
	I_{DD2}	Idle mode; V_{DD} = 4.5 V to 5.5 V 8-MHz CPU clock		5	10	
	I_{DD3}	Stop mode; V_{DD} = 4.5 V to 5.5 V		0.5	5	µA

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

Table 20. Input/Output Capacitance

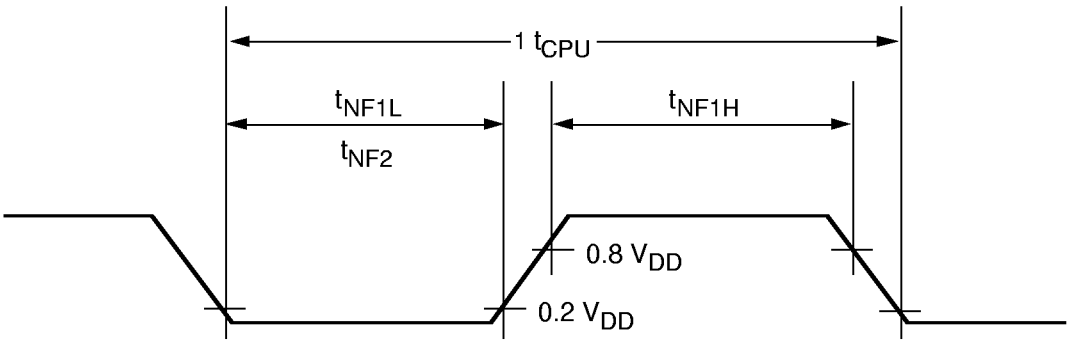
(T_A = –25°C to +85°C, V_{DD} = 0 V)

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 21. A.C. Electrical Characteristics

(T_A = –25°C to +85°C, V_{DD} = 4.5 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
V-sync Pulse Width	t_{VW}	–	4	–	–	µs
H-sync Pulse Width	t_{HW}	–	3	–	–	µs
Noise Filter	t_{NF1}	P0.4–P0.7, H-sync, V-sync, T0 (input), SDA0, SDA1, SCL0, SCL1	–	300	–	ns
	t_{NF2}	$\overline{\text{RESET}}$	–	750		
	t_{NF3}	Glitch filter (oscillator block)	–	25		
	t_{NF4}		–	100	–	µs



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

Figure 74. Input Timing Measurement Points for t_{NF1} and t_{NF2}

Table 22. Data Retention Supply Voltage in Stop Mode

(T_A = – 20 °C to + 85 °C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data Retention Supply Voltage	V_{DDDR}	Stop mode	2	–	6	V
Data Retention Supply Current	I_{DDDR}	Stop mode, V_{DDDR} = 2.0 V	–	–	5	µA

NOTES:
1. Supply current does not include current drawn through internal pull-up resistors or external output current loads.
2. During the oscillator stabilization wait time (t_{WAIT}), all CPU operations must be stopped.

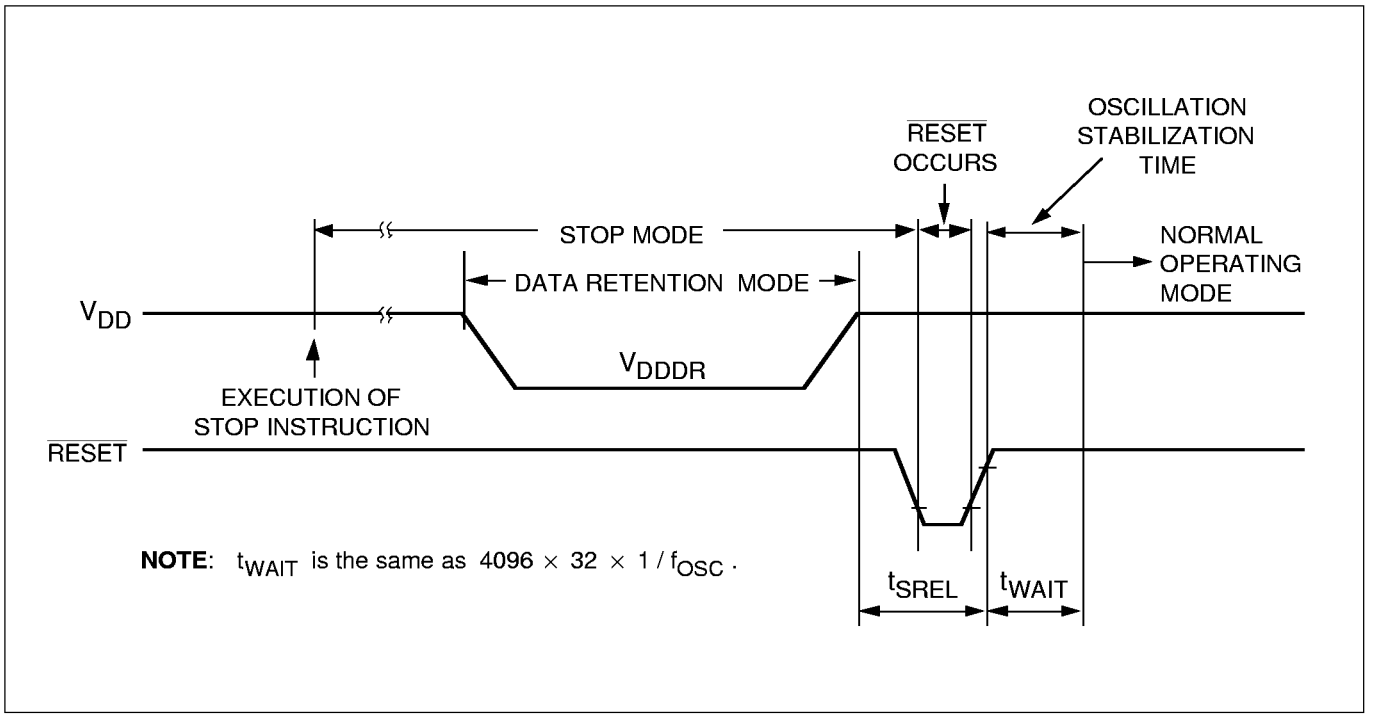
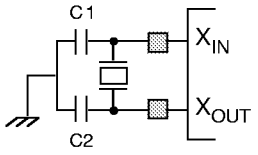
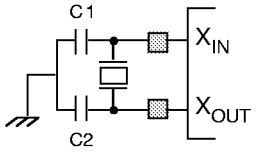
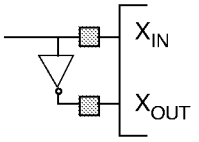
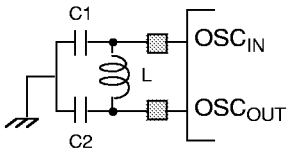


Figure 75. Stop Mode Release Timing When Initiated by a Reset

Table 23. Main Oscillator and L-C Oscillator Frequency

(T_A = – 25°C + 85°C, V_{DD} = 4.5–5.5 V)

Oscillator	Clock Circuit	Conditions	Min	Typ	Max	Unit
Crystal		OSD block active	5	6	8	MHz
		OSD block inactive	0.5	6	8	
Ceramic		OSD block active	5	6	8	MHz
		OSD block inactive	0.5	6	8	
External Clock		OSD block active	5	6	8	MHz
		OSD block inactive	0.5	6	8	
L-C Oscillator		–	5	6.5	8	MHz
CPU Clock Frequency		–	0.032	6.0	8	MHz

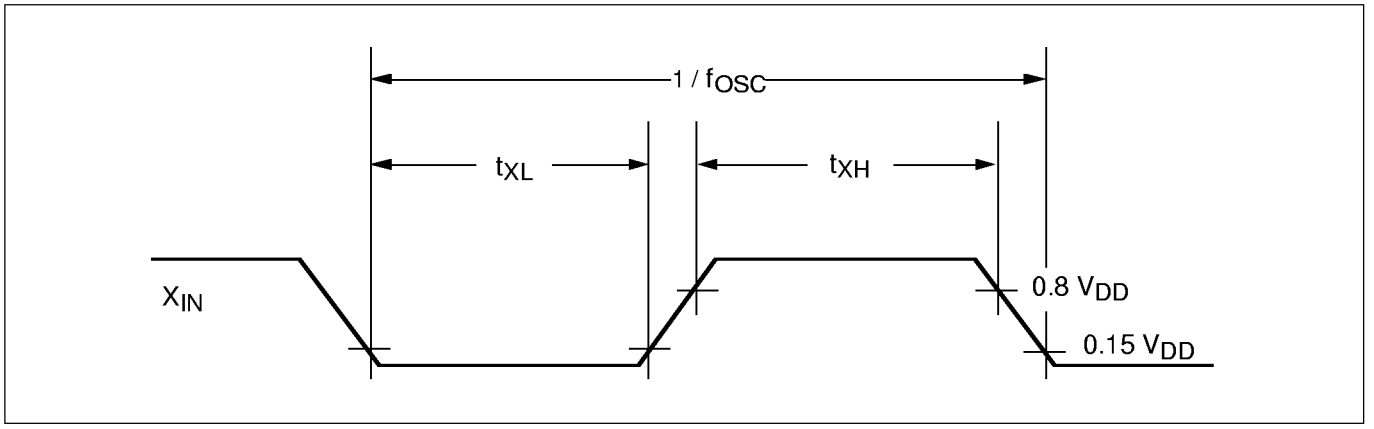


Figure 76. Clock Timing Measurement Points for X_{IN}

Table 24. Main Oscillator Clock Stabilization Time

(T_A = −25°C +85°C, V_{DD} = 4.5 V to 5.5 V)

Oscillator	Symbol	Test Condition	Min	Typ	Max	Unit
Crystal	–	V_{DD} = 4.5 V to 5.5 V	–	–	20	ms
Ceramic		(Oscillation stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.)			10	
External Clock		X_{IN} input High and Low level width (t_{XH}, t_{XL})	65	–	100	ns
Release Signal Setup Time	t_{SREL}	Normal operation	5	–	–	t_{CPU} (1)
Oscillation Stabilization Wait Time (2)	t_{WAIT}	CPU clock = 8 MHz; Stop mode released by RESET	–	16.5	–	ms
		CPU clock = 8 MHz; Stop mode released by an interrupt		(See Note 3)		

NOTES:
1. The unit t_{CPU} is one CPU clock period.
2. Oscillation stabilization time is the time required for the CPU clock to return to its normal oscillation frequency after a power-on occurs, or when Stop mode is released.
3. The oscillation stabilization interval is determined by the basic timer (BT) input clock setting.

Table 25. A/D Converter Electrical Characteristics

(T_A = −25°C to +85°C, V_{DD} = 4.5 V to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Absolute Accuracy (1)	–	CPU clock = 8 MHz	–	–	1	LSB
Conversion Time (2)	t_{CON}		$t_{CPU} \times 25$ (3)	–	–	μs
Analog Input Voltage	V_{IAN}	–	V_{SS}	–	V_{DD}	V
Analog Input Impedance	R_{AN}	–	2		–	MΩ

NOTES:
1. Excluding quantization error, absolute accuracy values are within ± 1/2 LSB.
2. 'Conversion time' is the time required from the moment a conversion operation starts until it ends.
3. The unit t_{CPU} means one CPU clock period.

CHARACTERISTIC CURVES

NOTE

The characteristic values shown in the following graphs are based on actual test measurements. They do not, however, represent guaranteed operating values.

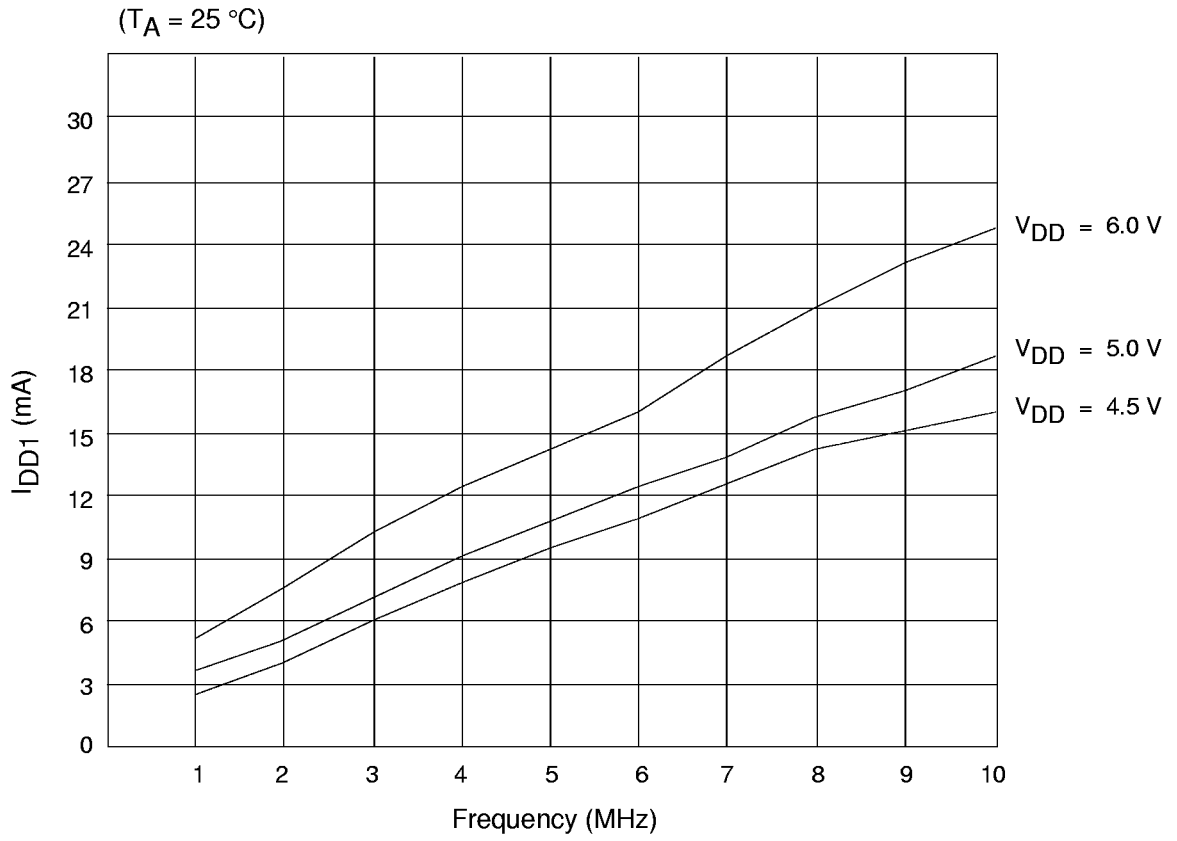


Figure 77. I_{DD1} vs. Frequency

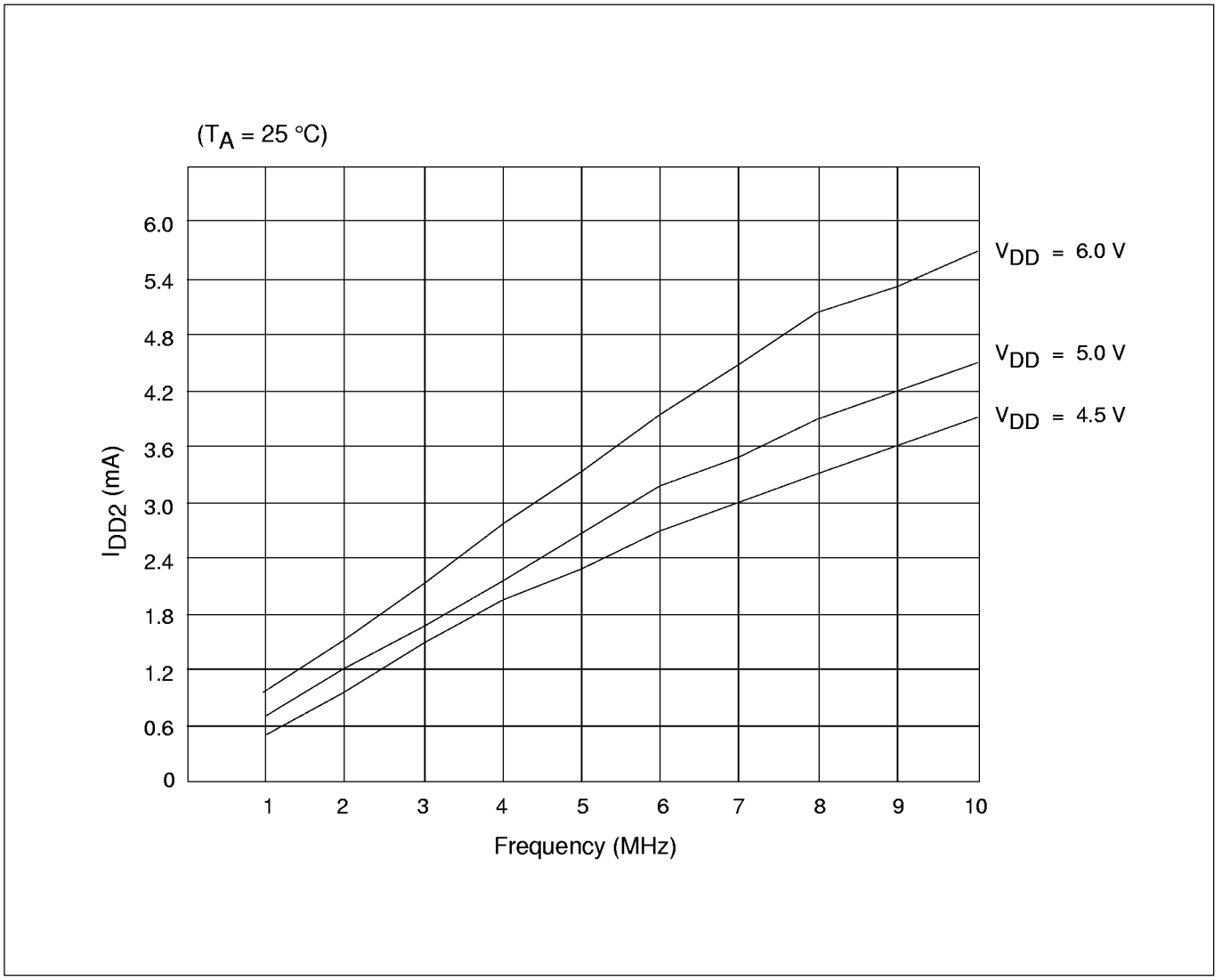


Figure 78. I_{DD2} vs. Frequency

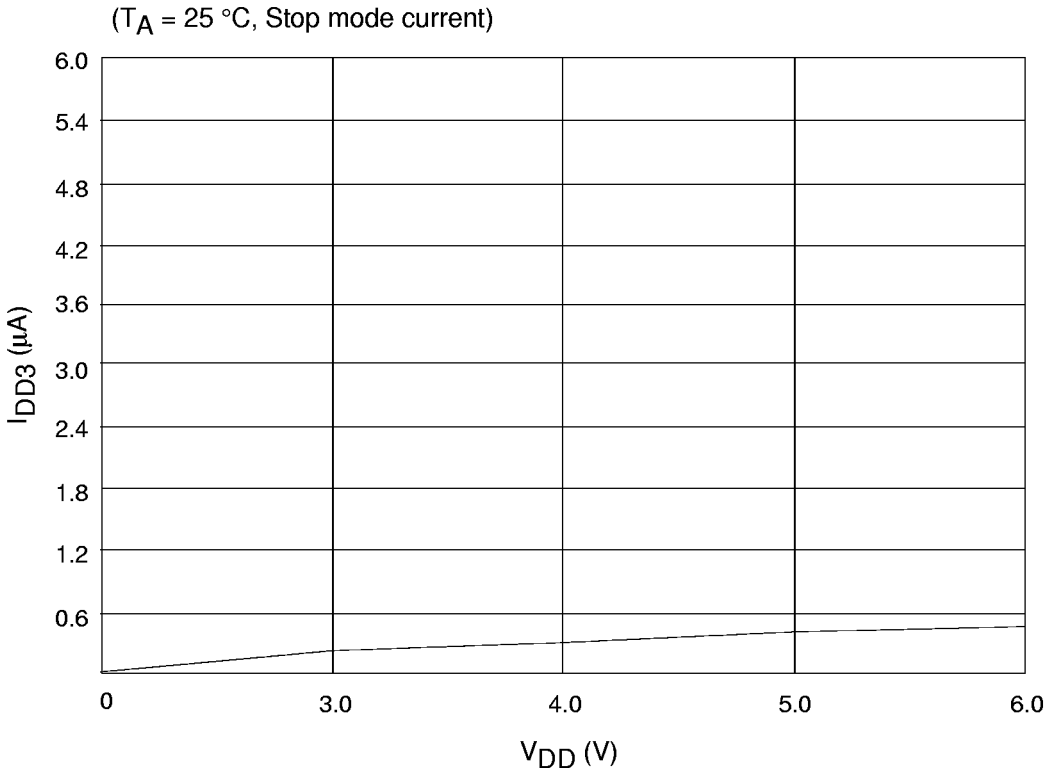
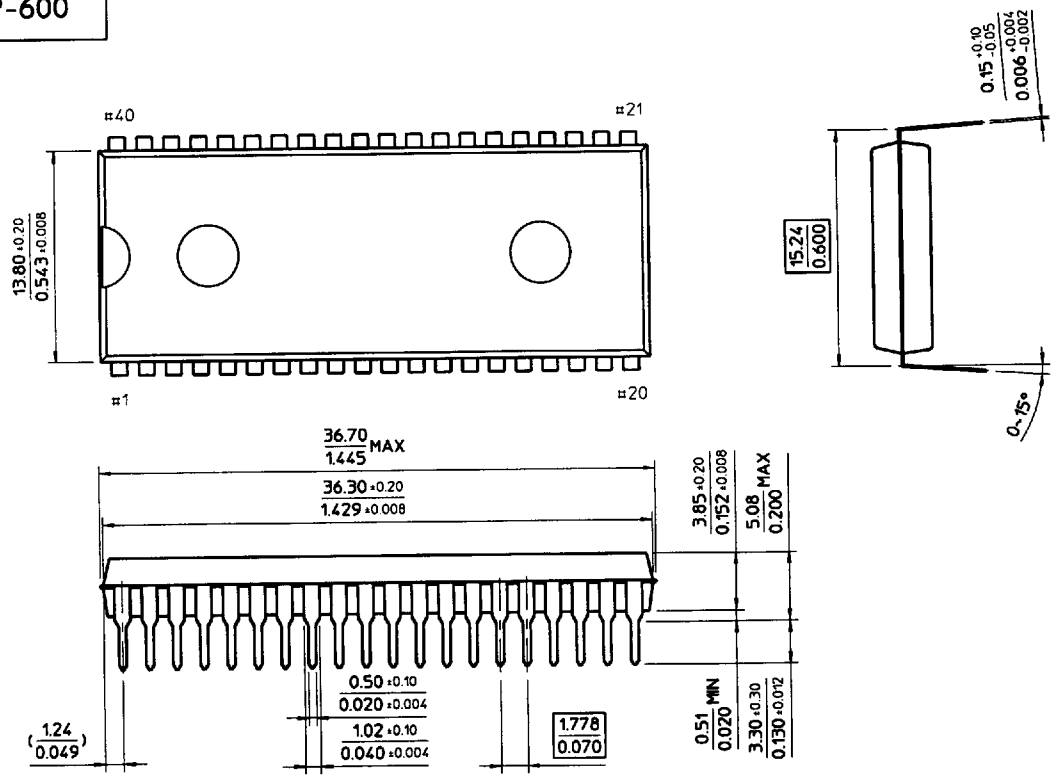
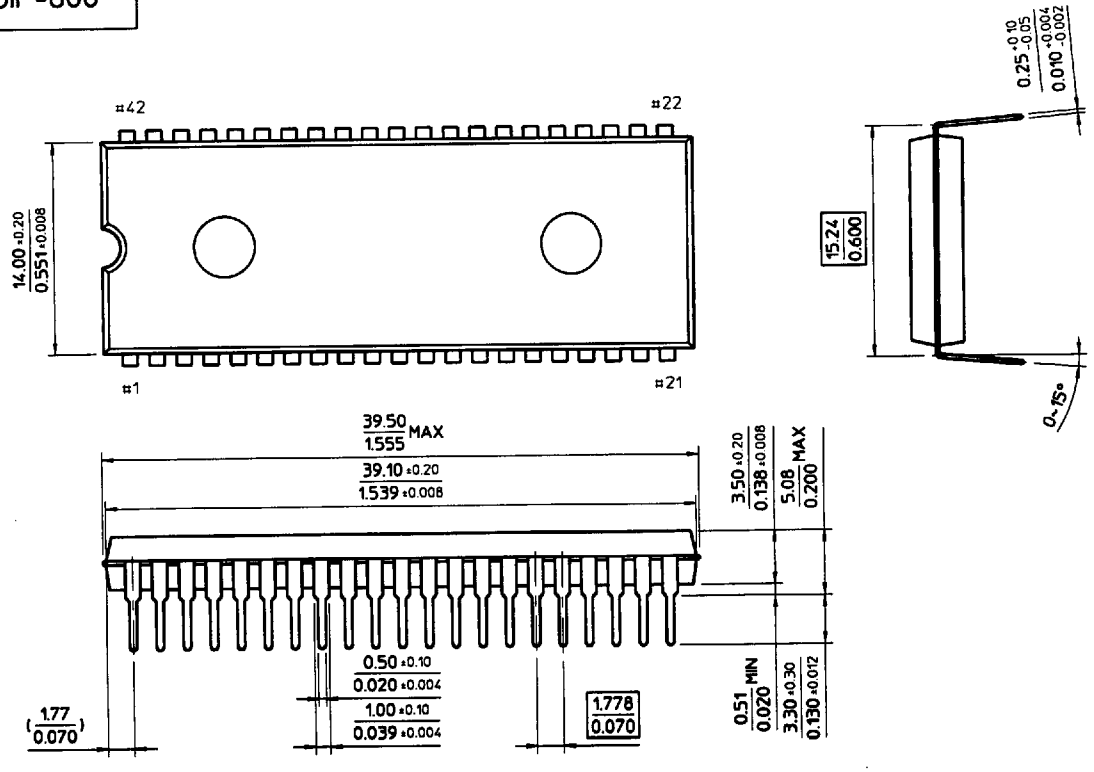


Figure 79. I_{DD3} vs. V_{DD} (Stop Mode Current)

40-SDIP-600



42-SDIP-600



SAMSUNG ELECTRONICS