

1 PRODUCT OVERVIEW

SAM87 PRODUCT FAMILY

Samsung's SAM87 family of 8-bit single-chip CMOS microcontrollers offers a fast and efficient CPU, a wide range of integrated peripherals, and various mask-programmable ROM sizes. Important CPU features include:

- Efficient register-oriented architecture
- Selectable CPU clock sources
- Release of Idle and Stop power-down modes by interrupt
- Built-in basic timer circuit with watchdog function

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

S3C8847/C8849/P8849 MICROCONTROLLERS

The S3C8847 microcontroller has a 24-Kbyte on-chip program memory and the S3C8849 has a 32-Kbyte. Both chips have a 272-byte general-purpose internal register file. The interrupt structure has nine interrupt sources with nine interrupt vectors. The CPU recognizes seven interrupt priority levels.

Using a modular design approach, the following peripherals were integrated with the SAM87 core to make the S3C8847/C8849/P8849 microcontrollers suitable for use in color television and other types of screen display applications:

- Four programmable I/O ports (26 pins total: 16 general-purpose I/O pins; 10 n-channel, open-drain output pins)
- 4-bit resolution A/D converter (4 channels)
- 14-bit PWM output (Two channels: push-pull type, open-drain type)
- Basic timer (BT) with watchdog timer function
- One 8-bit timer/counter (T0) with interval timer and PWM mode
- One 8-bit general-purpose timer/counter (TA) with prescalers
- On-screen display (OSD) with a wide range of programmable features, including halftone control signal output

The S3C8847 and the S3C8849 are available in versatile 42-pin SDIP package.

OTP

The S3C8847/C8849 microcontrollers are also available in OTP (One Time Programmable) version, named the S3P8849. The S3P8849 microcontroller has an on-chip 32-Kbyte one-time-programmable EPROM instead of a masked ROM. The KS88P8432 is comparable to the S3C8847/C8849, both in function and pin configuration.

FEATURES

CPU

- SAM87 CPU core

Memory

- 24-Kbyte (S3C8847) or 32-Kbyte (S3C8849) internal program memory
- 272-byte general-purpose register area

Instruction Set

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

Instruction Execution Time

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

Interrupts

- 9 interrupt sources with 9 vectors
- 7 interrupt levels
- Fast interrupt processing for select levels

General I/O

- Four I/O ports (26 pins total)
- Six open-drain pins for up to 6 V loads
- Four open-drain pins for up to 5 V 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 clocks or an external clock and interval timer mode or PWM mode.
- One general-purpose 8-bit timer/counters with interval timer mode (timer A)

A/D Converter

- Four analog input pins; 4-bit resolution
- 3.125 μ s conversion time (8 MHz CPU clock)

Pulse Width Modulation Module

- 14-bit PWM with two-channel output (push-pull type, open-drain type)
- 8-bit PWM with four-channel, push-pull and open-drain
- PWM counter and data capture input pin
- Frequency: 5.859 kHz to 23.437 kHz with a 6 MHz CPU clock

On-Screen Display (OSD)

- Video RAM: 252 \times 13-bits
- Character generator ROM: 384 \times 18 \times 16-bits (384 display characters; fixed; 2, variable; 382)
- 252 display positions (12 rows \times 21 columns)
- 16-dot \times 18-dot character resolution
- 16 different character sizes
- Eight character colors
- Vertical direction fade-in/fade-out control
- Eight colors for character and frame background
- Halftone control signal output; selectable for individual characters
- 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

- -20°C to $+85^{\circ}\text{C}$

Operating Voltage Range

- 4.5 V to 5.5 V

Package Type

- 42-pin SDIP

BLOCK DIAGRAM

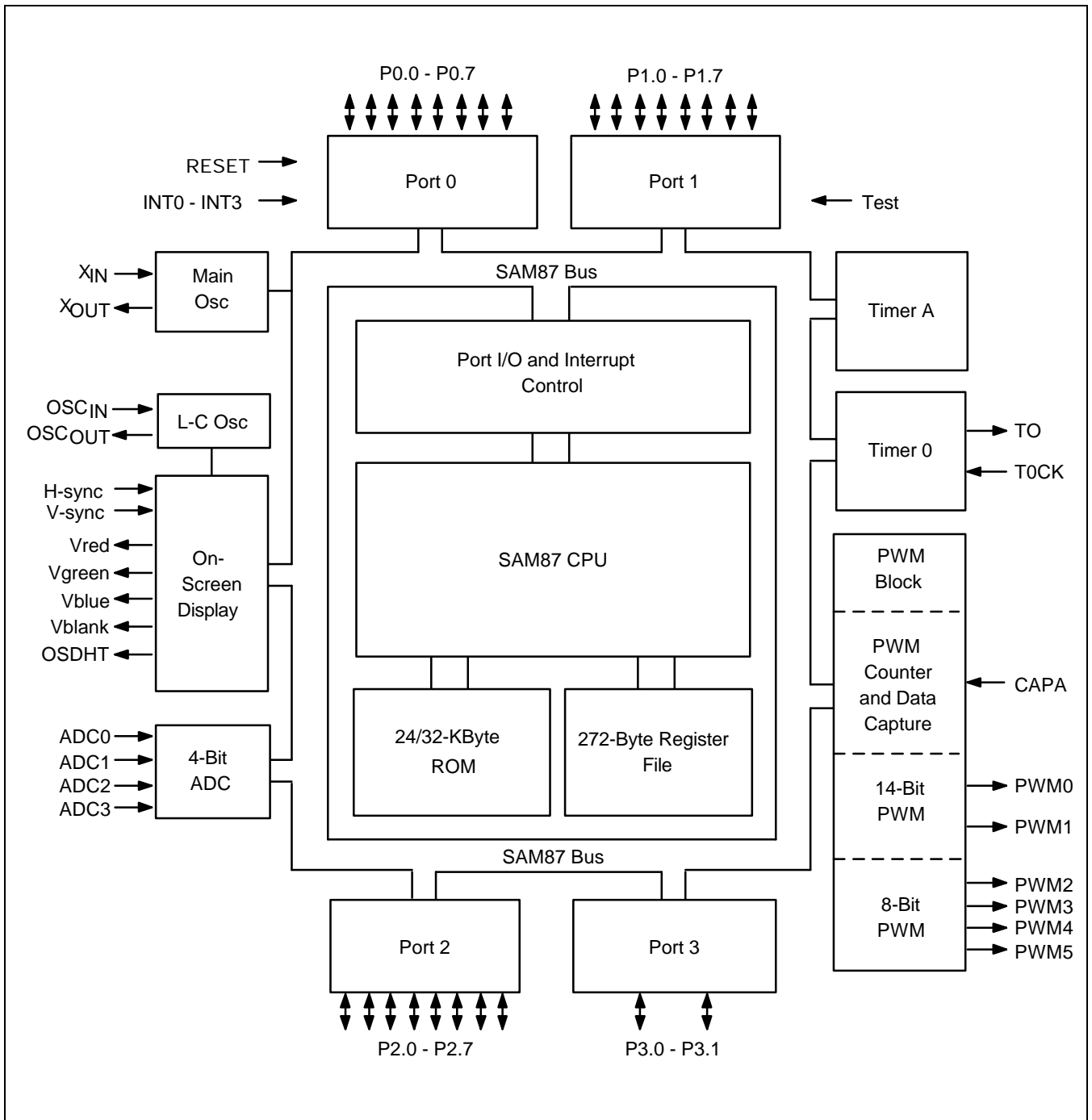


Figure 1-1. Block Diagram

PIN ASSIGNMENTS

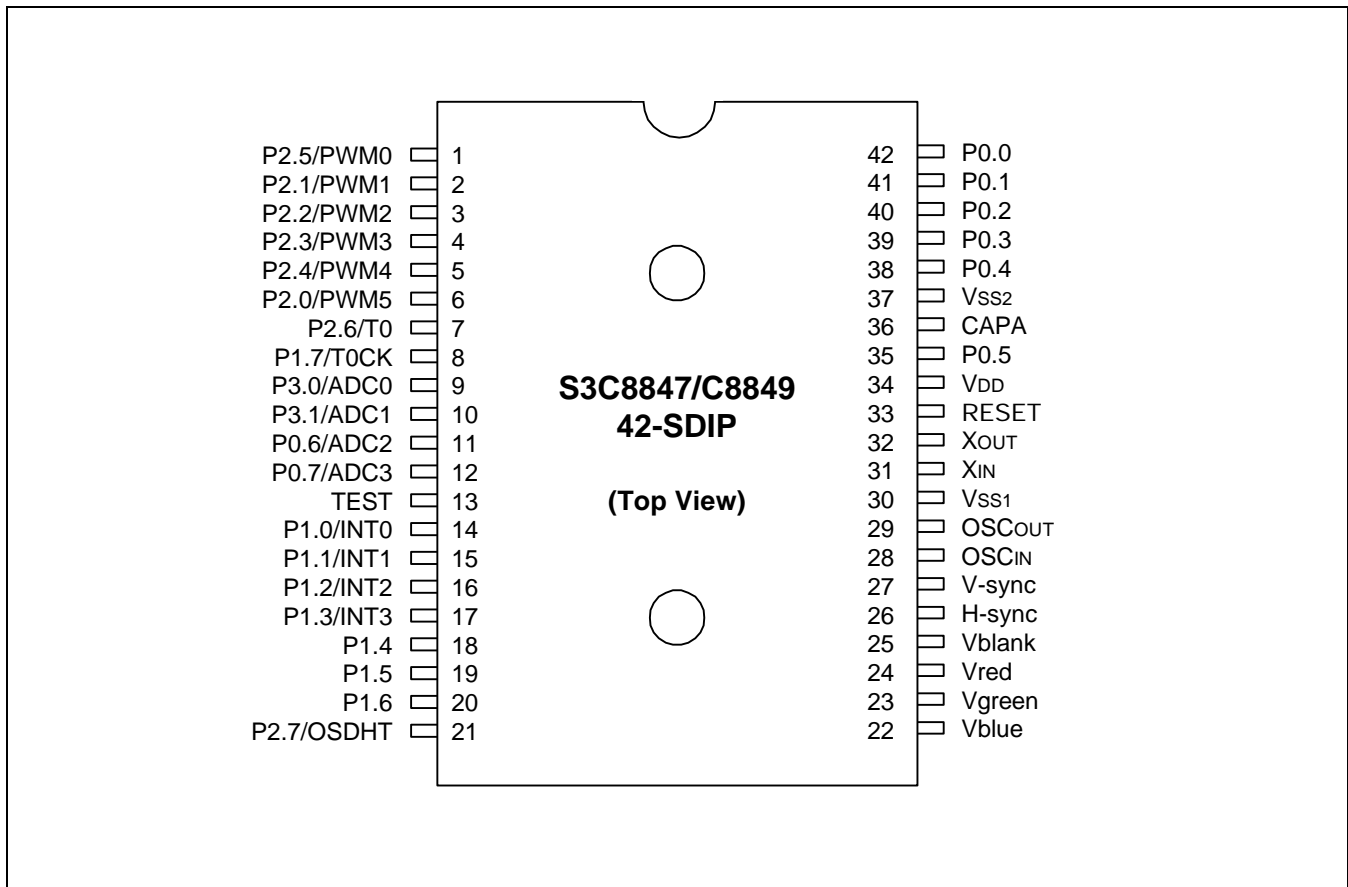


Figure 1-2. S3C8847/C8849/P8849 Pin Assignment Diagram

PIN DESCRIPTIONS

Table 1-1. S3C8847/C8849/P8849 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, push-pull output. Pins can withstand up to 5 V loads.	2	39–42	(see pin description)
P0.4–P0.5		General I/O port (2-bit), configurable for digital input or push-pull output.	3	38, 35	
P0.6–P0.7		General I/O port (2-bit), configurable for digital input or n-channel open-drain output. P0.6–P0.7 can withstand up to 5 V loads. Multiplexed for alternative use as external inputs, ADC2–ADC3.	6	11–12	ADC2–ADC3
P1.0–P1.3	I/O	General I/O port (4-bit), configurable for digital input or n-channel open-drain output. P1.0–P1.3 can withstand up to 6 V loads. Multiplexed for alternative use as external interrupt inputs, INT0–INT3.	7	14–17	INT0–INT3
P1.4–P1.5		General I/O port (2-bit), configurable for digital input or n-channel open-drain output. P1.4–P1.5 can withstand up to 6 V loads. High current port(10mA)	5	18–19	
P1.6–P1.7		General I/O port (2-bit), configurable for digital input or push-pull output. Each pin has an alternative function. P1.7: T0CK (Timer 0 clock input)	3	20, 8	T0CK
P2.0–P2.7	I/O	General I/O port (8-bit). Input/output mode or n-channel open-drain, push-pull output mode are software configurable. Pins can withstand up to 5 V loads. Each pin has an alternative function. P2.0: PWM5 (8-bit PWM output) P2.1: PWM1 (14-bit PWM output) P2.2: PWM2 (8-bit PWM output) P2.3: PWM3 (8-bit PWM output) P2.4: PWM4 (8-bit PWM output) P2.5: PWM0 (14-bit PWM output) P2.6: T0 (Timer 0 PWM and interval output) P2.7: OSDHT (Halftone signal output)	2	1–7, 21	PWM0– PWM5 T0, OSDHT
P3.0–P3.1	I/O	General I/O port (2-bit), configurable for digital input or n-channel open-drain output. P3.0–P3.1 can withstand up to 5 V loads. Multiplexed for alternative use as external inputs ADC0–ADC1.	6	9–10	ADC0–ADC1

Table 1-1. S3C8847/C8849/P8849 Pin Descriptions (Continued)

Pin Name	Pin Type	Pin Description	Circuit Type	Pin Numbers	Share Pins
PWM0–PWM1	O	Output pin for 14-bit PWM circuit	2	1, 2	P2.5, P2.1
PWM2–PWM5	O	Output pin for 8-bit PWM circuit	2	3–6	P2.2–P2.4, P2.0
ADC0–ADC3	I	Analog inputs for 4-bit A/D converter	6	9–12	P3.0–P3.1, P0.6–P0.7
INT0–INT3	I	External interrupt input pins	7	14–17	P1.0–P1.3
T0	O	Timer 0 output (interval, PWM)	2	7	P2.6
T0CK	I	Timer 0 clock input	3	8	P1.7
OSDHT	O	Halftone control signal output for OSD	2	21	P2.7
Vblue, Vgreen Vred, Vblank	O	Digital blue, green, red, and video blank signal outputs for OSD	4	22–25	–
H-sync, V-sync	I	H-sync, V-sync input for OSD	1	26, 27	–
OSC _{IN} , OSC _{OUT}	I, O	L-C oscillator pins for OSD clock frequency generation	–	28, 29	–
X _{IN} , X _{OUT}	I, O	System clock pins	–	31, 32	–
RESET	I	System reset input pin	8	33	–
TEST	–	Test Pin (must be connected to V _{SS}). Factory test mode is activated when 12V is applied.	–	13	–
V _{DD} , V _{SS1} , V _{SS2}	–	Power supply pins	–	34, 30, 37	–
CAPA	I	Input for capture A module	1	36	–

PIN CIRCUITS

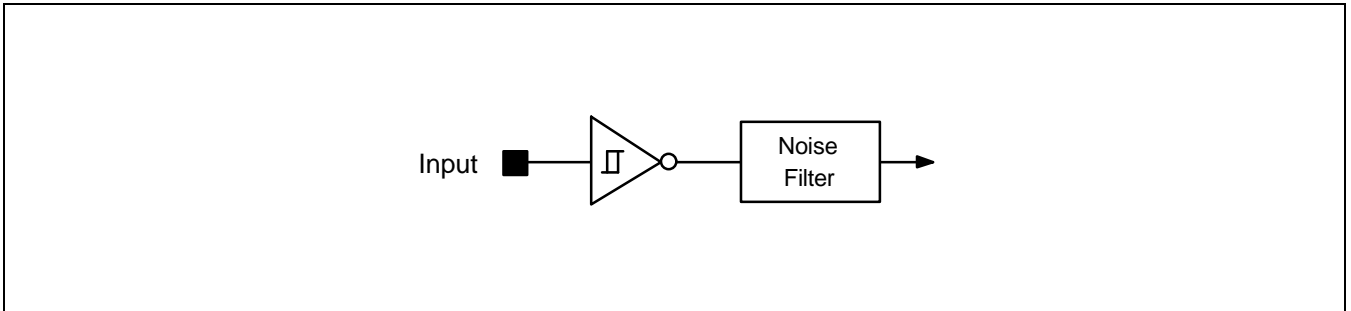


Figure 1-3. Pin Circuit Type 1 (V-Sync H-Sync, CAPA)

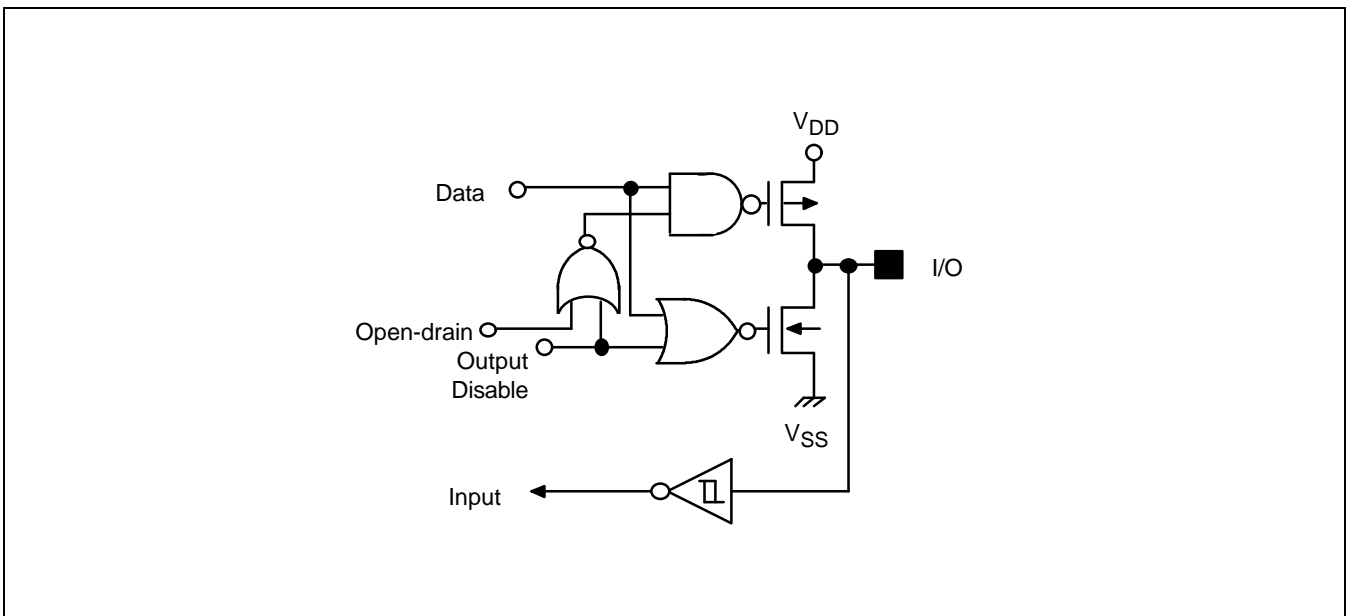


Figure 1-4. Pin Circuit Type 2 (P2.0–P2.7, P0.0–P0.3, PWM0–PWM5, T0, OSDHT)

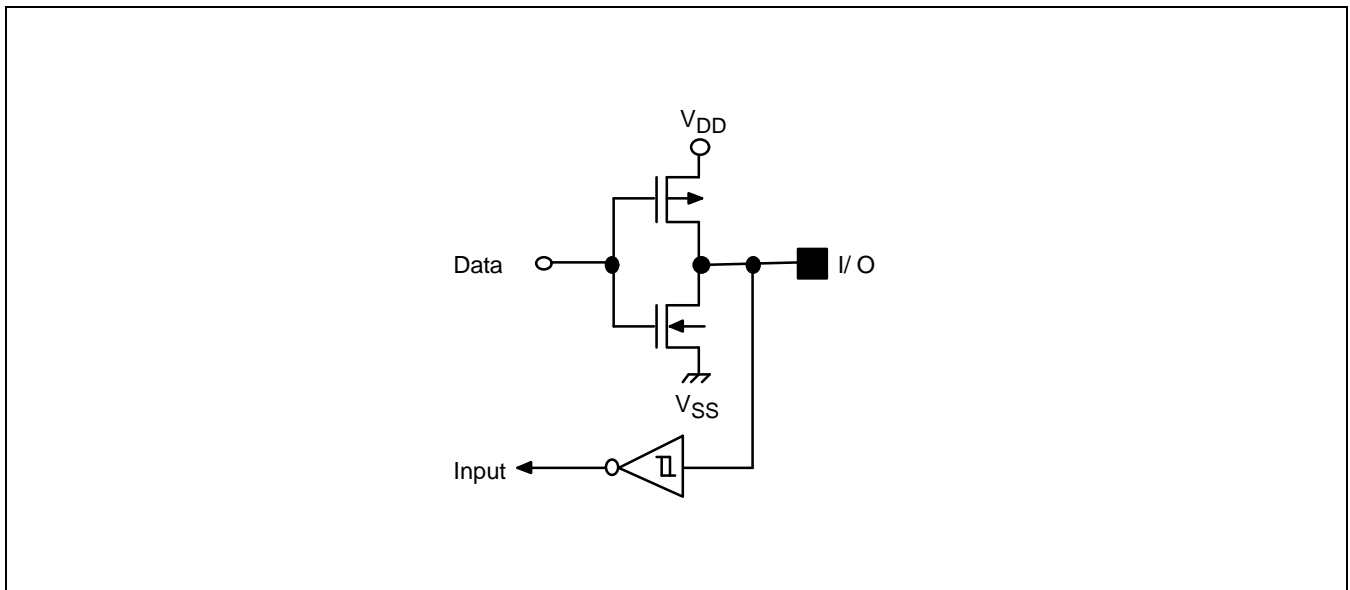


Figure 1-5. Pin Circuit Type 3 (P0.4–P0.5, P1.6–P1.7, T0CK)

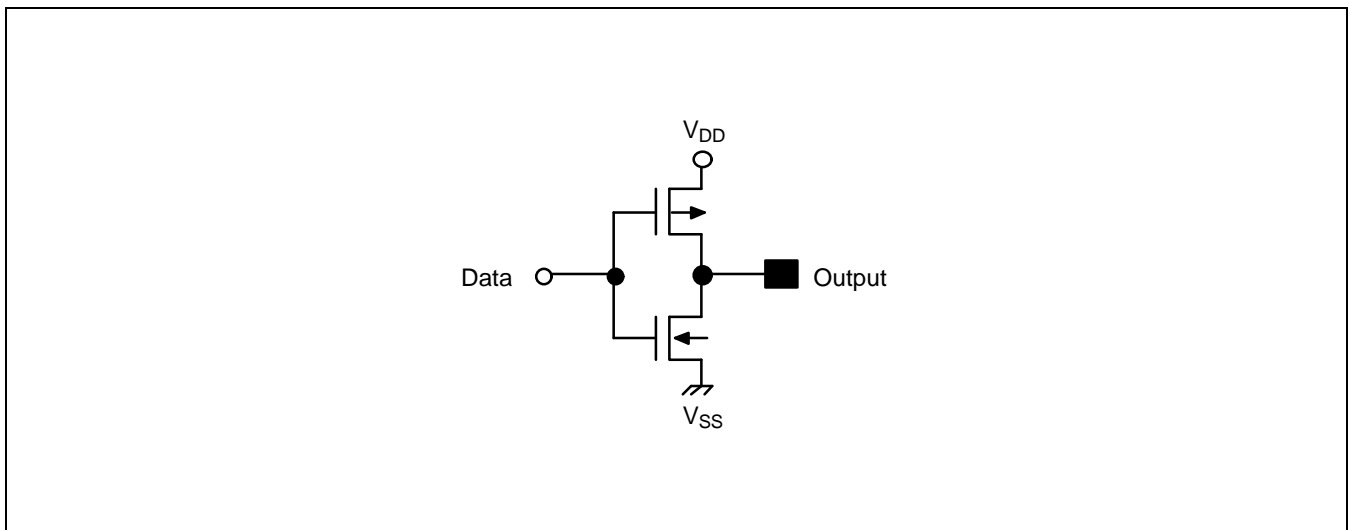


Figure 1-6. Pin Circuit Type 4 (Vblue, Vgreen, Vred, Vblank)

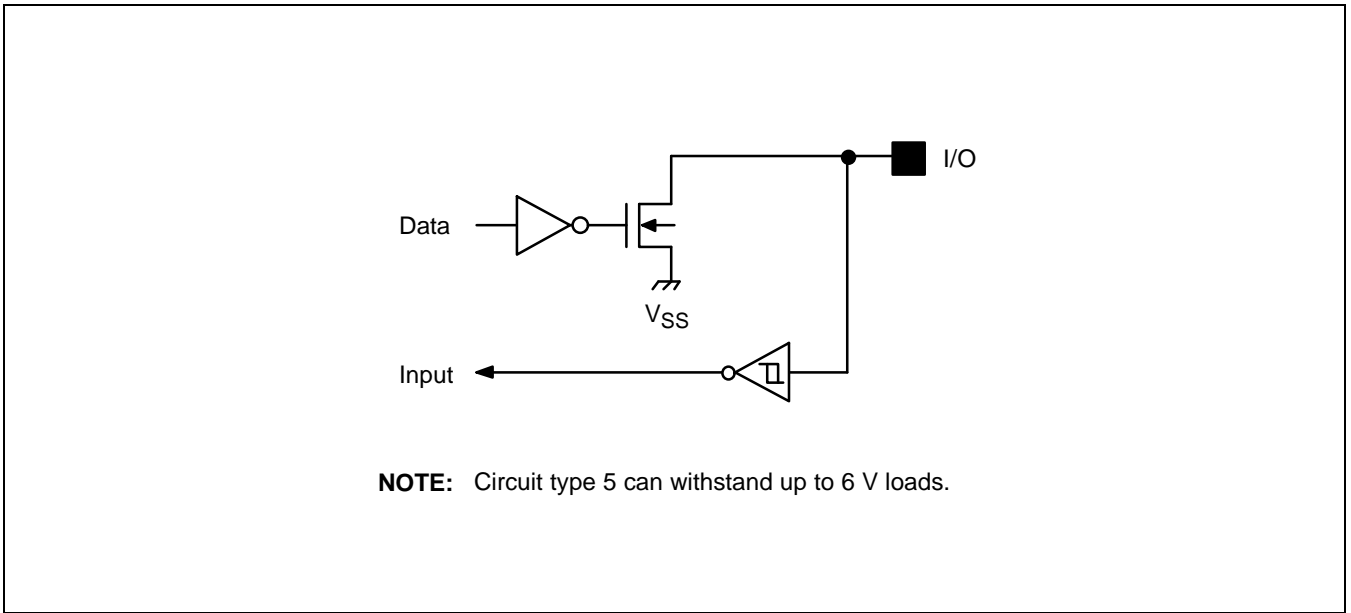


Figure 1-7. Pin Circuit Type 5 (P1.4-P1.5)

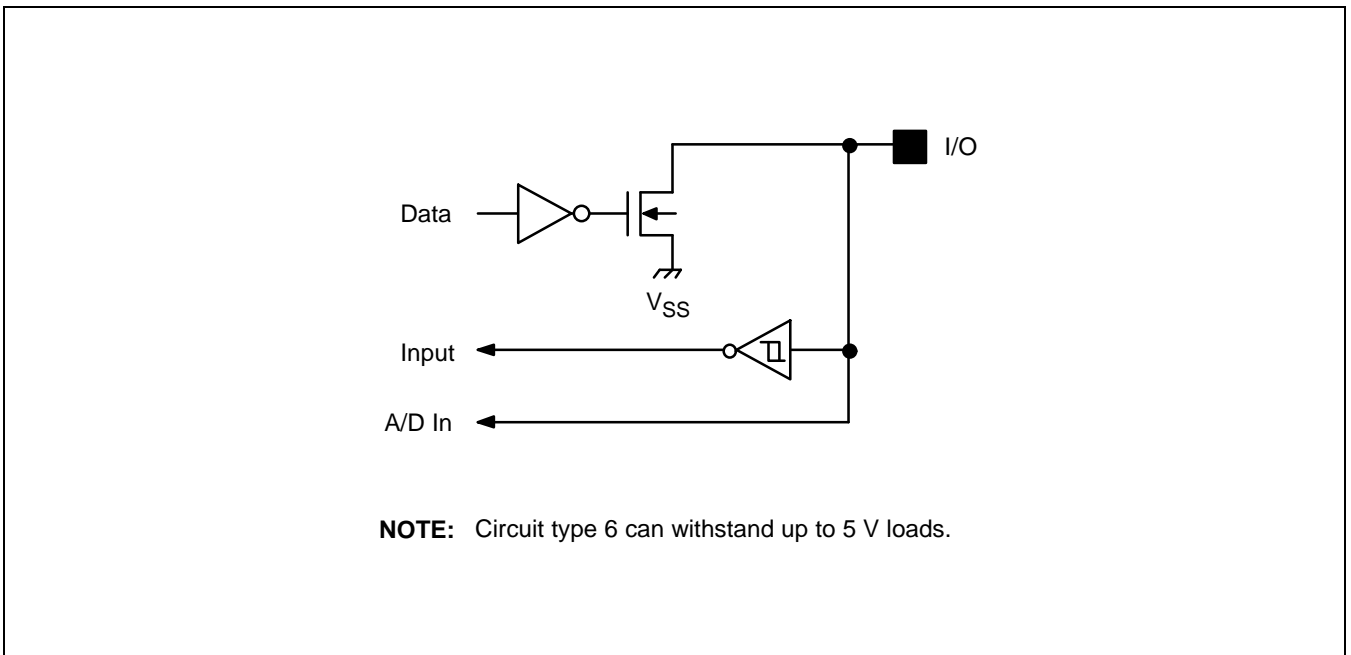


Figure 1-8. Pin Circuit Type 6 (P3.0-P3.1, P0.6-P0.7, ADC0-ADC3)

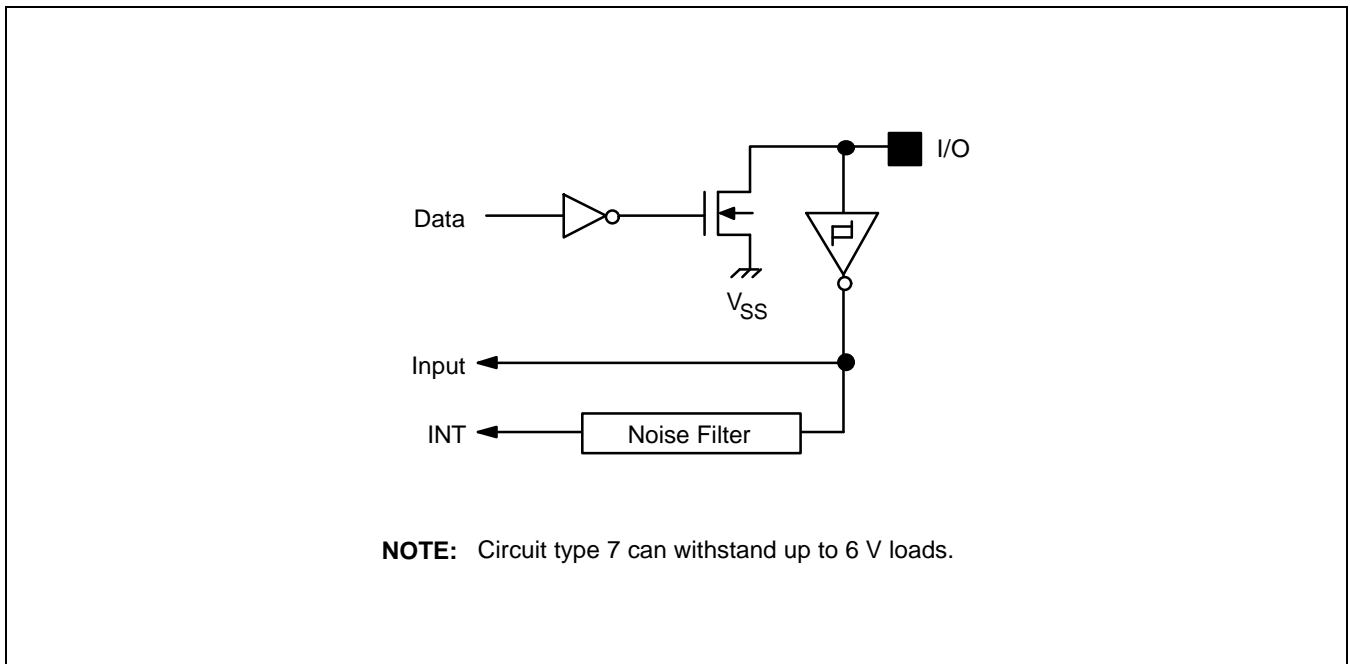


Figure 1-9. Pin Circuit Type 7 (P1.0–P1.3, INT0–INT3)

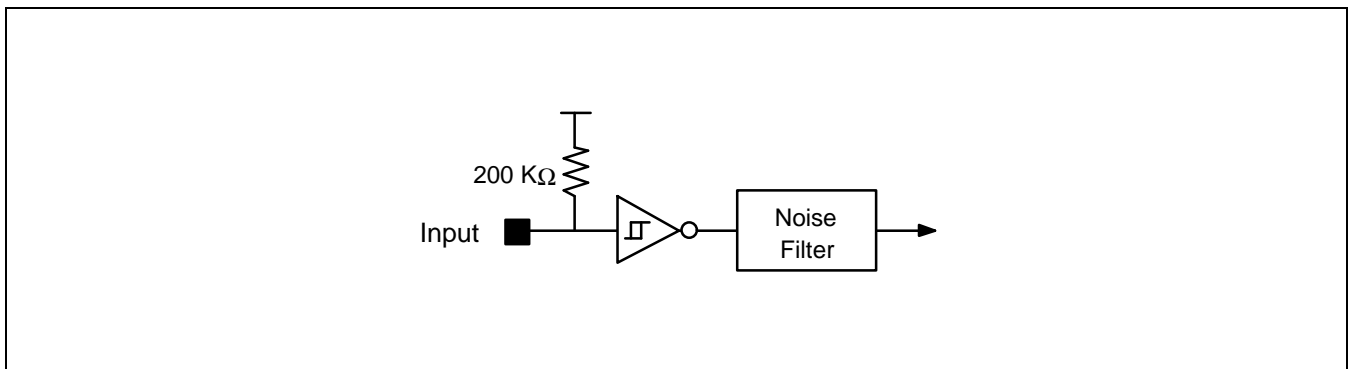


Figure 1-10. Pin Circuit Type 8 (RESET)

2 ADDRESS SPACES

OVERVIEW

The S3C8847 and S3C8849 microcontrollers have two kinds of address space:

- Internal program memory (ROM)
- Internal register file

The S3C8847 has an on-chip 24-Kbyte mask-programmable ROM; the S3C8849 has a 32-Kbyte. An external memory interface is not implemented.

There are 272 general-purpose 8-bit data registers in the register file. Thirteen 8-bit registers are used for CPU and system control. To support peripheral, I/O, and clock functions, there are 20 control registers and 9 data registers. In addition, there is a 252×13 -bit area for on-screen display (OSD) video RAM.

PROGRAM MEMORY (ROM)

Program memory (ROM) stores program codes or table data. The S3C8847 has a 24-Kbyte mask-programmable program memory (5FFFH) and the S3C8849 has a 32-Kbyte (7FFFH).

As shown in Figure 2-1, the first 256 bytes of the ROM (0H–0FFH) are reserved for interrupt vector addresses. Unused locations in this range can be used as normal program memory. If the vector address area is used to store normal program data, care must be taken to avoid overwriting vector addresses stored in these locations. The ROM address at which program execution starts after a RESET is 0100H.

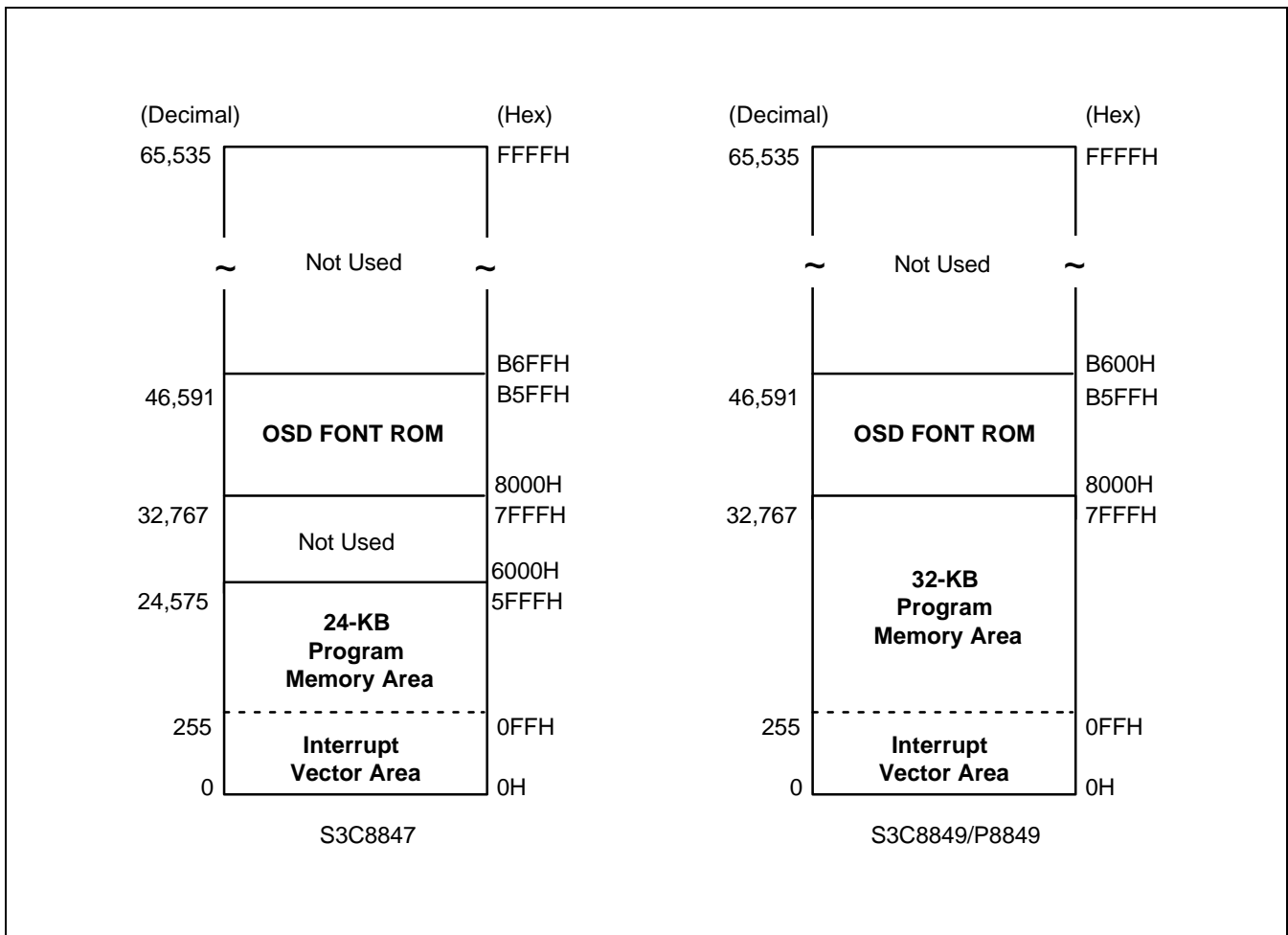


Figure 2-1. Program Memory Address Spaces

REGISTER ARCHITECTURE

The upper 64 bytes of the S3C8847/C8849/P8849 internal register file is logically expanded into two 64-byte areas, called *set 1* and *set 2*. The upper 32-byte area of set 1 is divided into two register banks, called *bank 0* and *bank 1*. In addition, two register pages are implemented, called *page 0* and *page 1*. The total addressable register space is thereby expanded from 256 bytes to 569 bytes.

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

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

Table 2-1. Register Type Summary

Register Type	Number of Bytes
General-purpose registers (including the 16-byte working register common area)	272
CPU and system control registers	16
Peripheral, I/O, and clock control/data registers	29
On-screen display (OSD) video RAM	252
Total Addressable Bytes	569

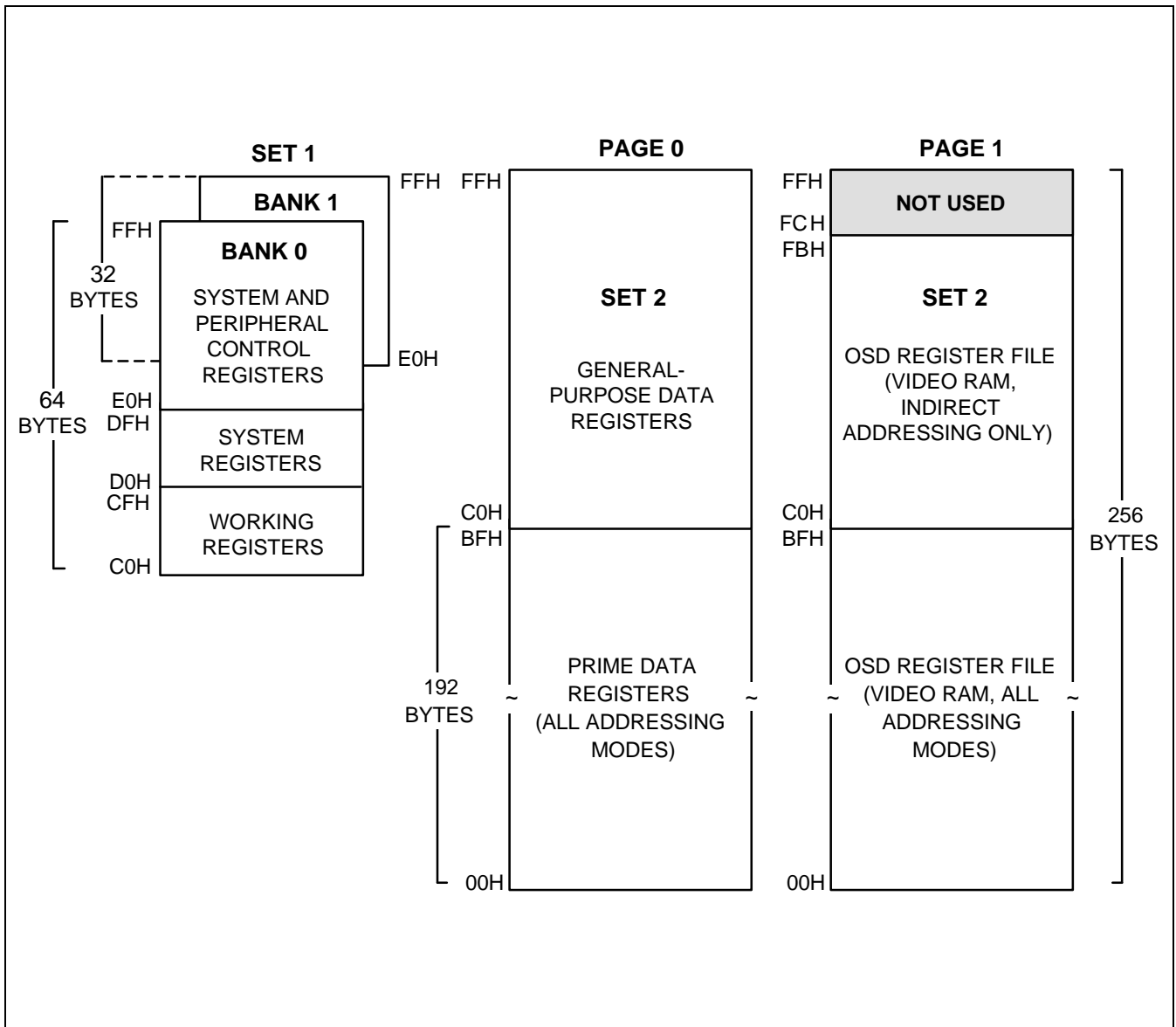


Figure 2-2. Internal Register File Organization

REGISTER PAGE POINTER (PP)

The SAM87 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 S3C8847/C8849/P8849 microcontrollers: page 0 is used as general-purpose register space and page 1 contains a 252 × 13-bit area for the on-screen display (OSD) video ROM.

As shown in Figure 2-3, 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', page 0 is the selected source register page; when the lower nibble is '0001B', page 1 is the source register page.

After a reset, the page pointer's source value (the lower nibble) and the destination value (the upper nibble) are always '0000B', automatically selecting page 0 as both the source and the 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 S3C8847/C8849/P8849 implementation, only pointer values '0000B' and '0001B' are used.

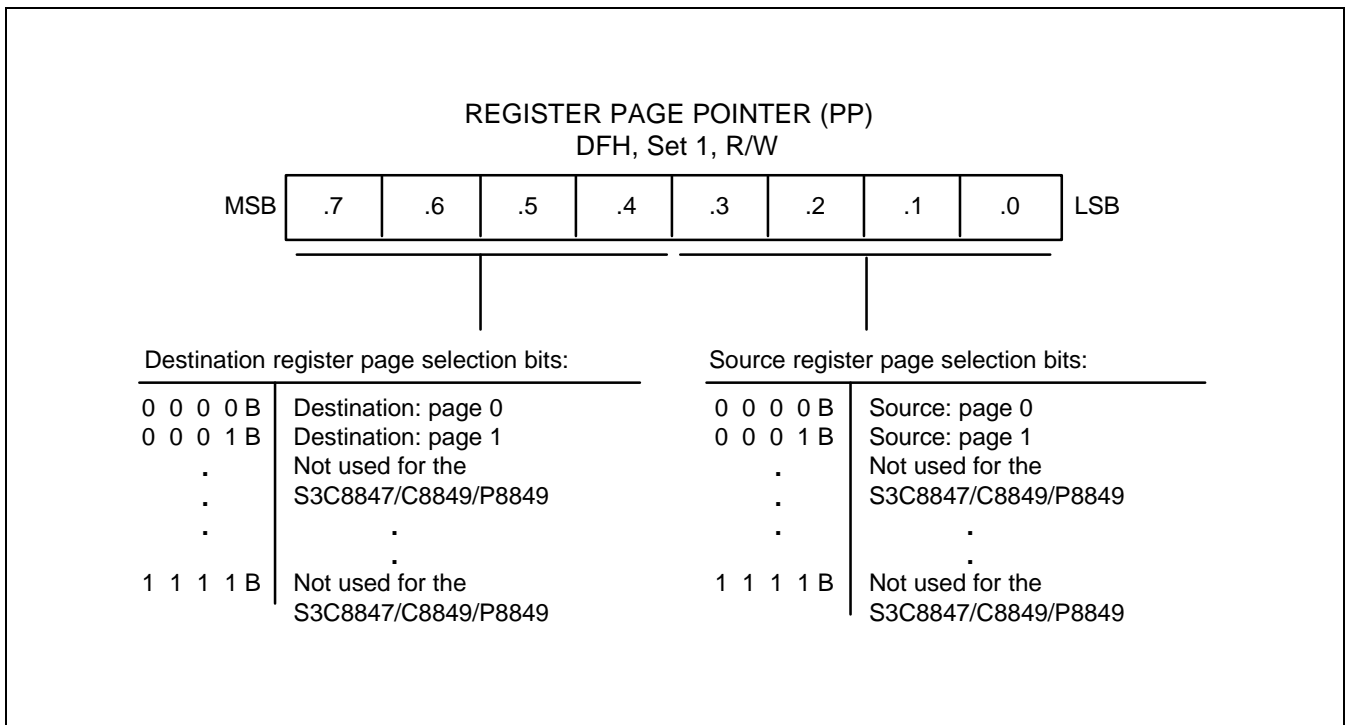


Figure 2-3. Register Page Pointer (PP)

 **PROGRAMMING TIP – Data Operations Between Register Pages**

```

LD      PP,#10H      ; Destination register page 1, source register page 0
LD      20H,45H      ; Register 20H in page 1 ← Content of the register 45H
                        ; in page 0
.
.
.
ADD     30H,40H      ; Register 30H in page 1 ← Content of 30H in page 1
                        ; plus (+) the content of 40H in page 0

```

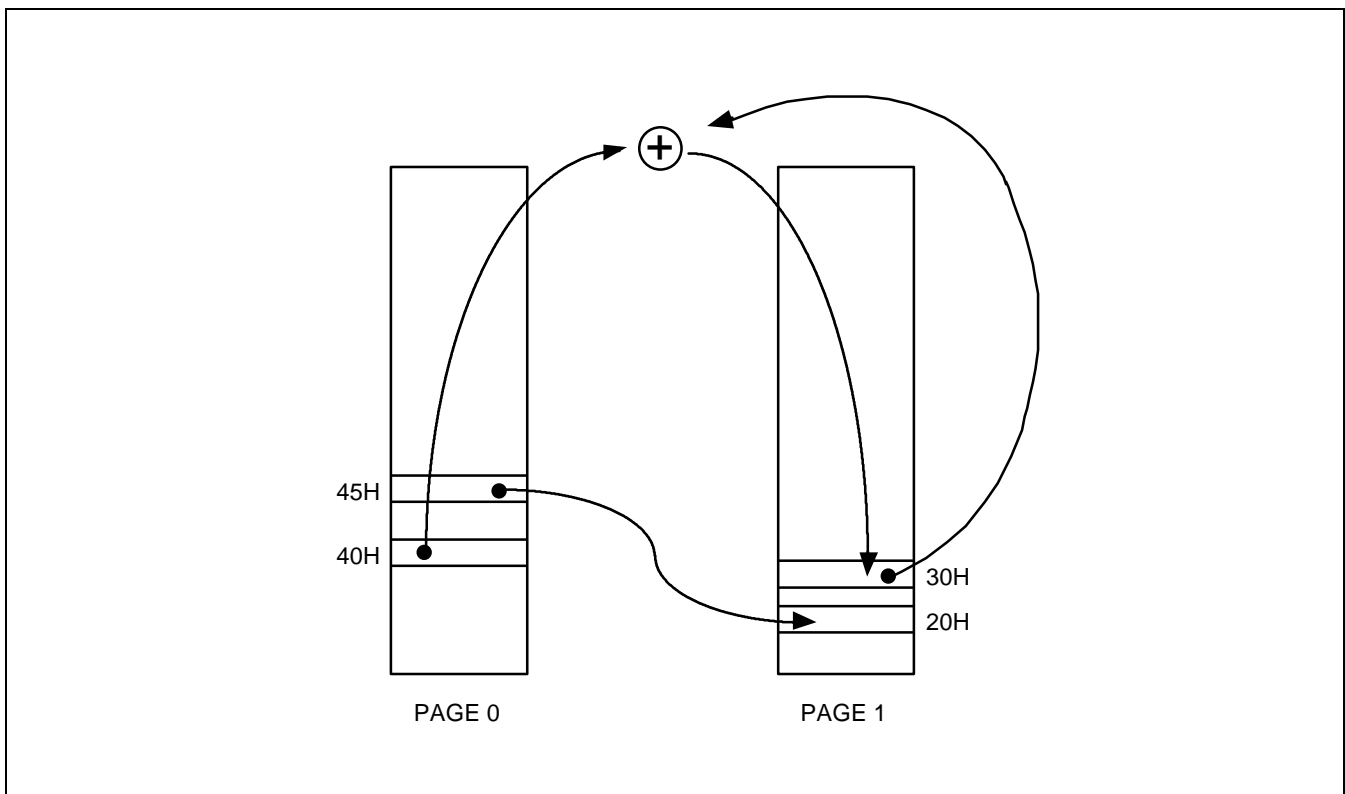


Figure 2-4. Programming Tip Example for Inter-Page Data Operations

EFFECT OF DIFFERENT INSTRUCTIONS FOR INTER-PAGE DATA OPERATIONS

The source and the destination pages for data operations between pages differ, depending on which instruction you use. The following programming tip, "Examples of Inter-Page Data Transfer Operations," provides you with a detailed list of various case.

 **PROGRAMMING TIP – Examples of Inter-Page Data Transfer Operations**

- Example 1.**
- | | | | | |
|----|-----|------------------|---|------------------------|
| a) | ADC | R1,R0 | ; | R0 – source page |
| | | | ; | R1 – destination page |
| b) | ADC | R4,@R2 | ; | R2, 40H – source page |
| | | R2 contains 40H | ; | R4 – destination page |
| c) | ADC | R0,#0AAH | ; | R0 – destination page |
| d) | ADC | 40H,42H | ; | 42H – source page |
| | | | ; | 40H – destination page |
| e) | ADC | 40H,@42H | ; | 42H, 60H – source page |
| | | 42H contains 60H | ; | 40H – destination page |
| f) | ADC | 40H,#02H | ; | 40H – destination page |

NOTE: The above examples also apply to the instructions ADD, SUB, AND, OR, and XOR.

- Example 2.**
- | | | | | |
|----|------|----------|---|------------------------|
| a) | BAND | R0,40H.7 | ; | 40H – source page |
| | | | ; | R0 – destination page |
| b) | BAND | 40H.7,R0 | ; | R0 – source page |
| | | | ; | 40H – destination page |

NOTE: The above examples also apply to the instructions BOR, BXOR, and LDB.

- Example 3.**
- | | | | | |
|----|-----|----------|---|-----------------------|
| a) | BCP | R3,44H.7 | ; | 44H – source page |
| | | | ; | R3 – destination page |

- Example 4.**
- | | | | | |
|----|------|------|---|-----------------------|
| a) | BITC | R3.7 | ; | R3 – destination page |
|----|------|------|---|-----------------------|

NOTE: The above examples also apply to the instructions BITR and BITS.

 **PROGRAMMING TIP – Examples of Inter-Page Data Transfer Operations (continued)**

Example 5. a) BTJRF SKIP,R6.7 ; R6 – source page

NOTE: The above example also applies to the instructions BTJRT.

Example 6. a) CALL @60H ; 60H, 61H – source page

Example 7. a) CLR 30H ; 30H – destination page

b) CLR @44H ; 44H – source page
44H contains 40H ; 40H – destination page

NOTE: The above examples also apply to the instructions RL, RLC, and SRA.

Example 8. a) COM 03H ; 03H – destination page

b) COM @44H ; 44H – source page
44H contains 40H ; 40H – destination page

NOTE: The above examples also apply to the instructions DEC, INC, RR, and RRC.

Example 9. a) CP R1,R0 ; R0 – source page
; R1 – destination page

b) CP R2,@R4 ; R4, 40H – source page
R4 contains 40H ; R2 – destination page

c) CP 40H,42H ; 42H – source page
; 40H – destination page

d) CP 40H,@42H ; 42H, 44H – source page
42H contains 44H ; 40H – destination page

e) CP 20H,#0AAH ; 20H – destination page

NOTE: The above examples also apply to the instructions TCM and TM.

 **PROGRAMMING TIP – Examples of Inter-Page Data Transfer Operations (Continued)**

Example 16.	a)	LD	R0,#0AAH	;	R0 – destination page
	b)	LD	R0,40H	;	40H – source page
				;	R0 – destination page
	c)	LD	40H,R0	;	R0 – source page
				;	40H – destination page
	d)	LD	R0,@R2	;	R2, 50H – source page
			R2 contains 50H	;	R0 – destination page
	e)	LD	@R4,R2	;	R4, R2 – source page
			R4 contains 40H	;	40H – destination page
	f)	LD	40H,41H	;	41H – source page
				;	40H – destination page
	g)	LD	40H,@42H	;	42H, 44H – source page
			42H contains 44H	;	40H – destination page
	h)	LD	45H,#02H	;	45H – destination page
	i)	LD	@40H,#02H	;	40H – source page
			40H contains 44H	;	44H – destination page
	j)	LD	@40H,42H	;	40H, 42H – source page
			40H contains 44H	;	44H – destination page
	k)	LD	R5,#04H(R0)	;	R0, 04H(2 + offset) – source page
			R0 contains 02H	;	R5 – destination page
	l)	LD	#04H(R0),R1	;	R0, R1 – source page
			R0 contains 02H	;	04H – destination page

 **PROGRAMMING TIP – Examples of Inter-Page Data Transfer Operations (Concluded)**

- Example 20.**
- a) MULT 40H,20H ; 20H – source page
 ; 40H, 41H – destination page
 - b) MULT 60H,@20H ; 20H, 40H – source page
 ; 60H, 61H – destination page
 ; 20H contains 40H
 - c) MULT 40H,#02H ; 40H, 41H – destination page
- Example 21.**
- a) POP 00H ; 00H – destination page
 - b) POP @20H ; 20H – source page
 ; 40H – destination page
 ; 20H contains 40H
- Example 22.**
- a) POPUD 00H,@20H ; 20H, 40H – source page
 ; 00H – destination page
 ; 20H contains 40H

NOTE: The above example also applies to the instruction POPUI.

- Example 23.**
- a) PUSH 00H ; 00H – destination page
 - b) PUSH @20H ; 20H, 40H – source page
 ; 40H – destination page
 ; 20H contains 40H

- Example 24.**
- a) PUSHUD @60H,20H ; 60H, 20H – source page
 ; 44H – destination page
 ; 60H contains 44H

NOTE: The above example also applies to the instruction PUSHUI.

- Example 25.**
- a) SWAP 00H ; 00H – destination page
 - b) SWAP @20H ; 20H – source page
 ; 40H – destination page
 ; 20H contains 40H

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 Register addressing mode. The 16-byte working register area, however, can only be accessed using working register addressing. Working register addressing is a function of Register addressing mode (see Chapter 3, "Addressing Modes," for more information).

REGISTER SET 2

The same 64-byte physical space that is used for the 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 should use Register Indirect addressing mode or Indexed addressing mode to access set 2.

For the S3C8847/C8849/P8849, the set 2 address range (C0H–FFH) is accessible on page 0 and page 1. Please note, however, that on page 1, the set 2 locations FCH–FFH are not mapped.

Part of the OSD video RAM is in page 1, set 2 (C0H–FBH), 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 252-byte video RAM area.

PRIME REGISTER SPACE

The lower 192 bytes (00H–BFH) of the S3C8847/C8849/P8849'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 after 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.

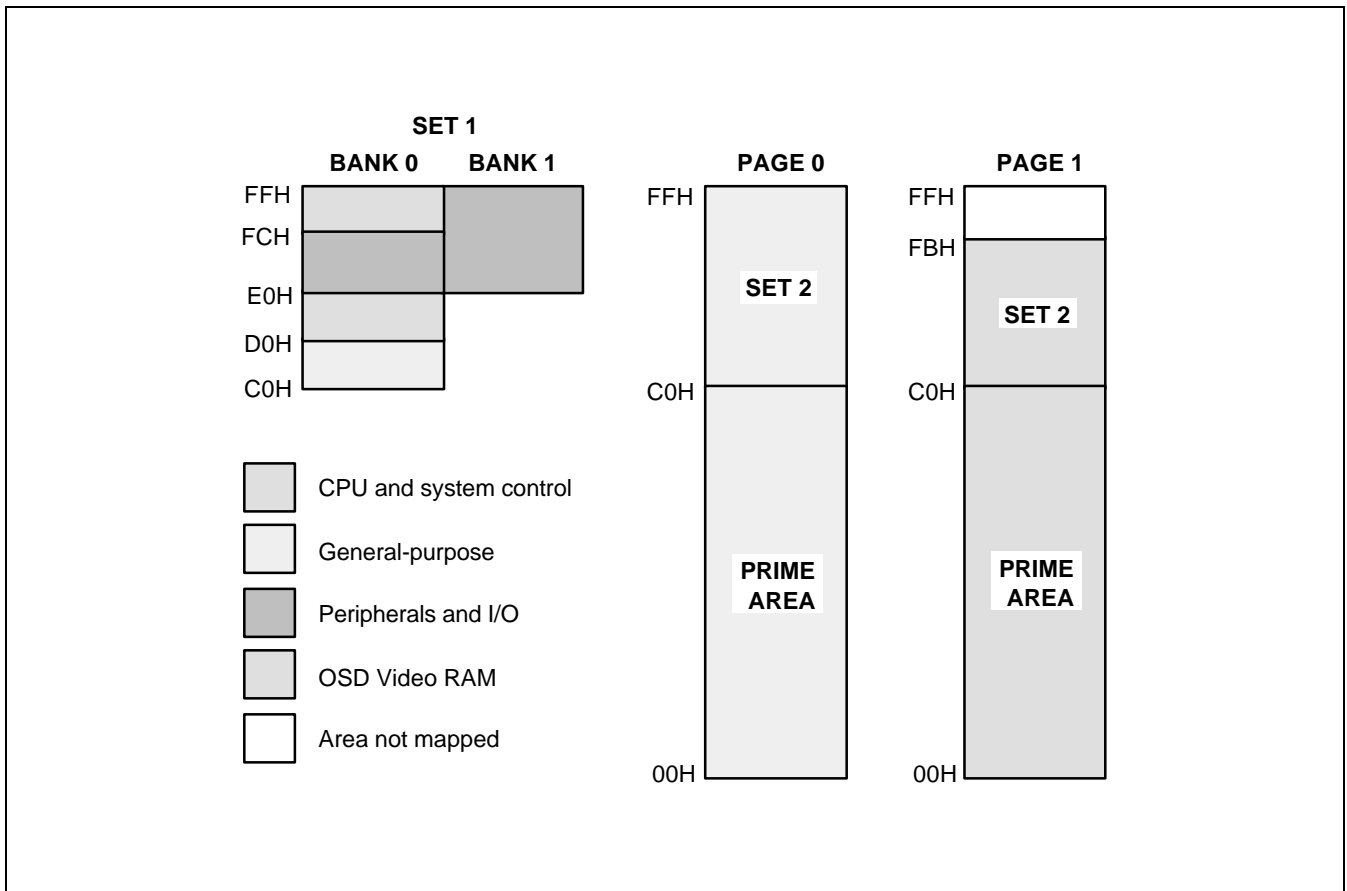


Figure 2-5. Set 1, Set 2, and Prime Area Register Map

WORKING REGISTERS

Instructions can access specific 8-bit registers or 16-bit register pairs using either 4-bit or 8-bit address fields. When 4-bit working register addressing is used, the 256-byte register file is viewed as thirty two 8-byte register groups or "slices." Each slice consists of eight 8-bit registers. When the two 8-bit register pointers, RP1 and RP0, are used, two working register slices can be selected at any time to form a 16-byte working register block. Using the register pointers, you can move this 16-byte register block anywhere in the addressable register file (except for the set 2 area).

The terms *slice* and *block* are used in this manual to help you visualize the size and relative locations of selected working register spaces:

- One working register *slice* is 8 bytes (eight 8-bit working registers; R0–R7 or R8–R15)
- One working register *block* is 16 bytes (sixteen 8-bit working registers; R0–R15)

All the registers in an 8-byte working register slice have the same binary value for their five most significant address bits. This makes it possible for each register pointer to point to one of the 24 slices in the register file. The base addresses for the two selected 8-byte register slices are contained in the register pointers, RP0 and RP1. After a reset, RP0 and RP1 always point to the 16-byte common area in set 1 (C0H–CFH).

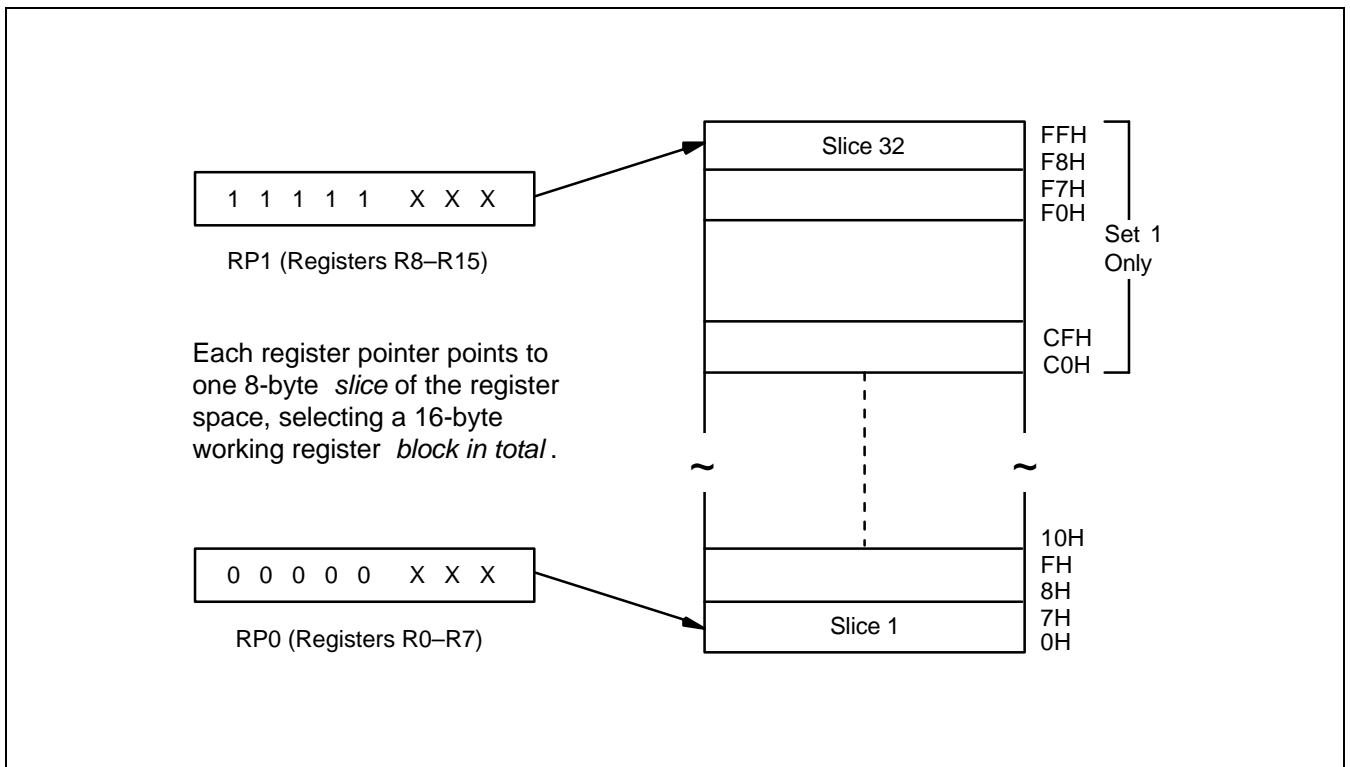


Figure 2-6. 8-Byte Working Register Areas (Slices)

USING THE REGISTER POINTERS

The register pointers, RP0 and RP1, are mapped to the addresses D6H and D7H in set 1. They are used to select two movable 8-byte working register slices in the register file. After a reset, they point to the working register common area: RP0 points to the addresses C0H–C7H, and RP1 points to the addresses C8H–CFH. If you want to change a register pointer value, you should load a new value to RP0 and/or RP1 using an SRP or LD instruction (see Figures 2-7 and 2-8).

With working register addressing, you can only access those locations that are pointed to by the register pointers. Please note that you cannot use the register pointers to select the working register area in set 2, C0H–FFH, because these locations are accessible only using the Indirect Register or Indexed addressing mode.

The selected 16-byte working register block usually consists of two contiguous 8-byte slices. As a general programming guideline, we recommend that RP0 point to the "lower" slice and RP1 point to the "upper" slice (see Figure 2-7). In some cases, it may be necessary to define working register areas in different (non-contiguous) areas of the register file. In Figure 2-8, RP0 points to the "upper" slice and RP1 to the "lower" slice.

Because a register pointer can point to either of the two 8-byte slices in the working register block, you can flexibly define the working register area.

PROGRAMMING TIP — Setting the Register Pointers

SRP	#70H	; RP0 ← 70H, RP1 ← 78H
SRP1	#48H	; RP0 ← no change, RP1 ← 48H
SRP0	#0A0H	; RP0 ← A0H, RP1 ← no change
CLR	RP0	; RP0 ← 00H, RP1 ← no change
LD	RP1,#0F8H	; RP0 ← no change, RP1 ← 0F8H

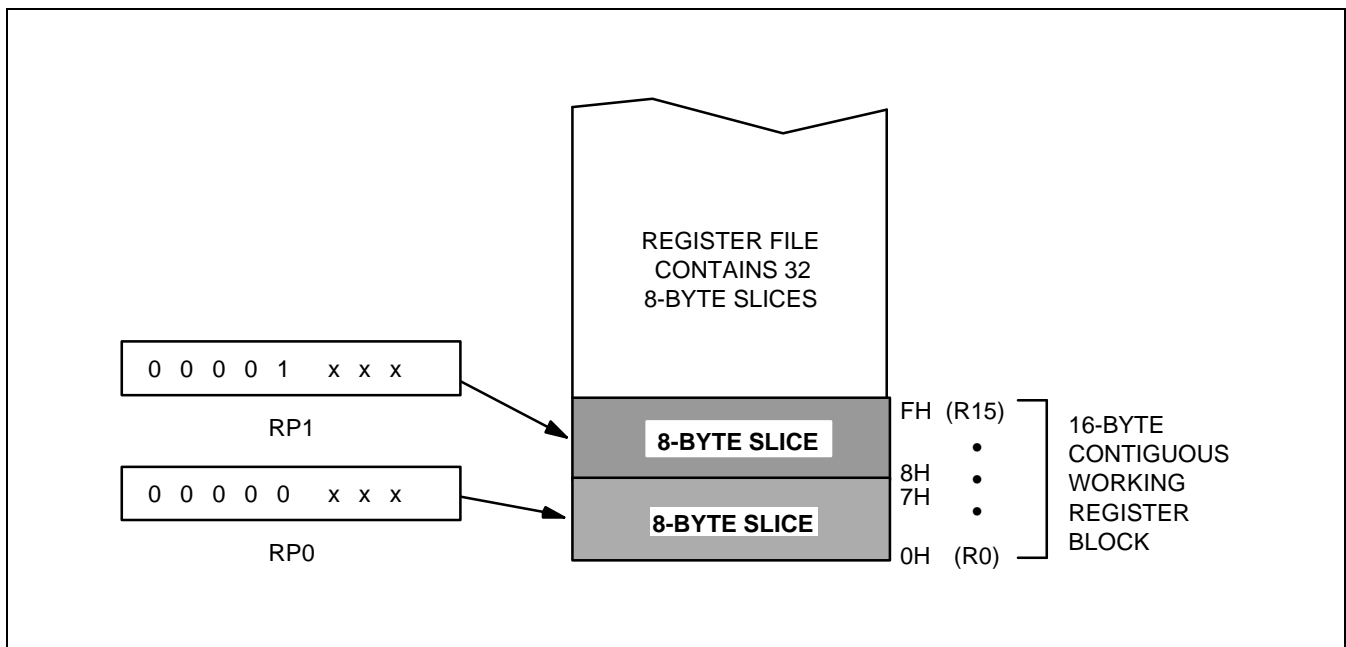


Figure 2-7. Contiguous 16-Byte Working Register Block

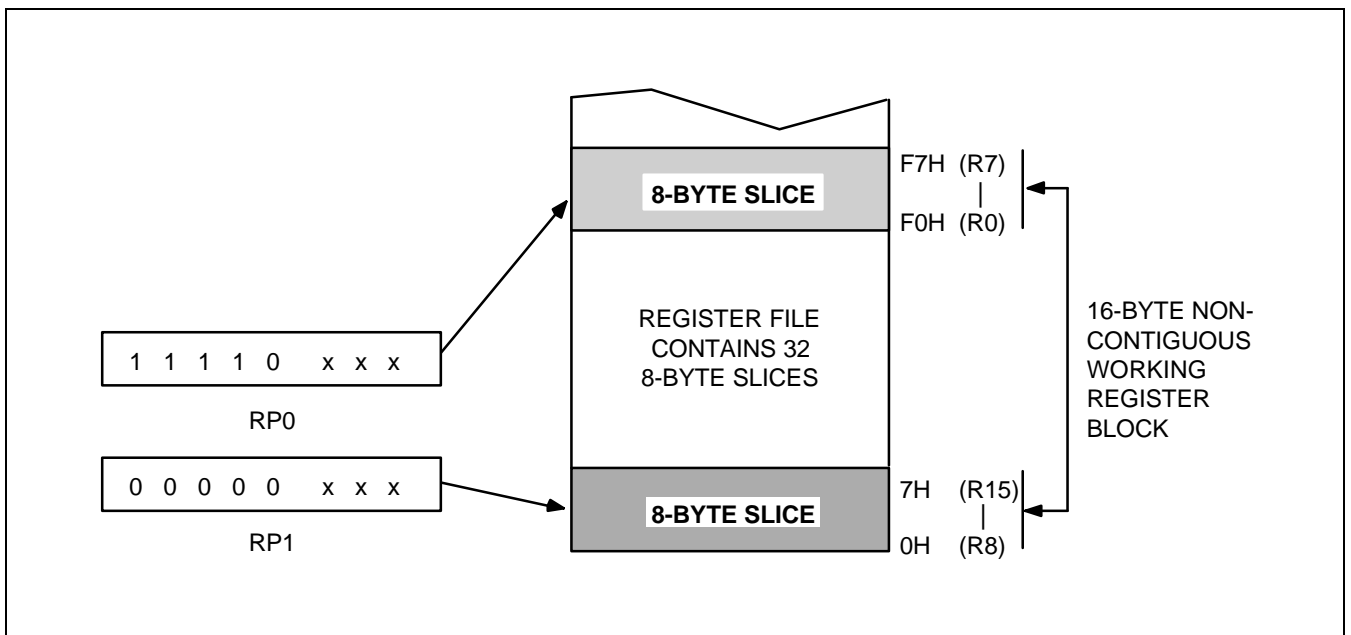


Figure 2-8. Non-Contiguous 16-Byte Working Register Block

PROGRAMMING TIP — Using the RPs to Calculate the Sum of a Series of Registers

Calculate the sum of the registers, 80H–85H, using the register pointer. The register addresses 80H through 85H contain the values 10H, 11H, 12H, 13H, 14H, and 15 H, respectively:

```

SRP0    #80H           ; RP0 ← 80H
ADD     R0,R1          ; R0 ← R0 + R1
ADC     R0,R2          ; R0 ← R0 + R2 + C
ADC     R0,R3          ; R0 ← R0 + R3 + C
ADC     R0,R4          ; R0 ← R0 + R4 + C
ADC     R0,R5          ; R0 ← R0 + R5 + C

```

The sum of these six registers, 6FH, is located in the register R0 (80H). The instruction string used in this example takes 12 bytes of instruction code and its execution time is 36 cycles. If you do not use the register pointer to calculate the sum of these registers, you would have to execute the following instruction sequence:

```

ADD     80H,81H        ; 80H ← (80H) + (81H)
ADC     80H,82H        ; 80H ← (80H) + (82H) + C
ADC     80H,83H        ; 80H ← (80H) + (83H) + C
ADC     80H,84H        ; 80H ← (80H) + (84H) + C
ADC     80H,85H        ; 80H ← (80H) + (85H) + C

```

Here, the sum of the six registers is also located in the register 80H. This instruction string, however, takes 15 bytes of instruction code rather than 12 bytes, and the execution time is 50 cycles rather than 36 cycles.

REGISTER ADDRESSING

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

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

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

Registers are addressed either as a single 8-bit register or a paired 16-bit register. In 16-bit register pairs, the address of the first 8-bit register is always an even number and that of the next register is an odd number. The most significant byte of the 16-bit data is always stored in the even-numbered register; the least significant byte is always stored in the next (+ 1) odd-numbered register.

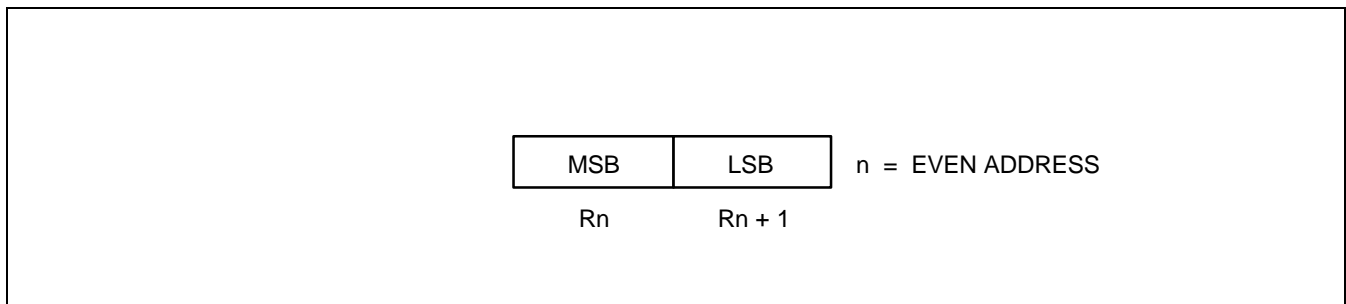


Figure 2-9. 16-Bit Register Pairs

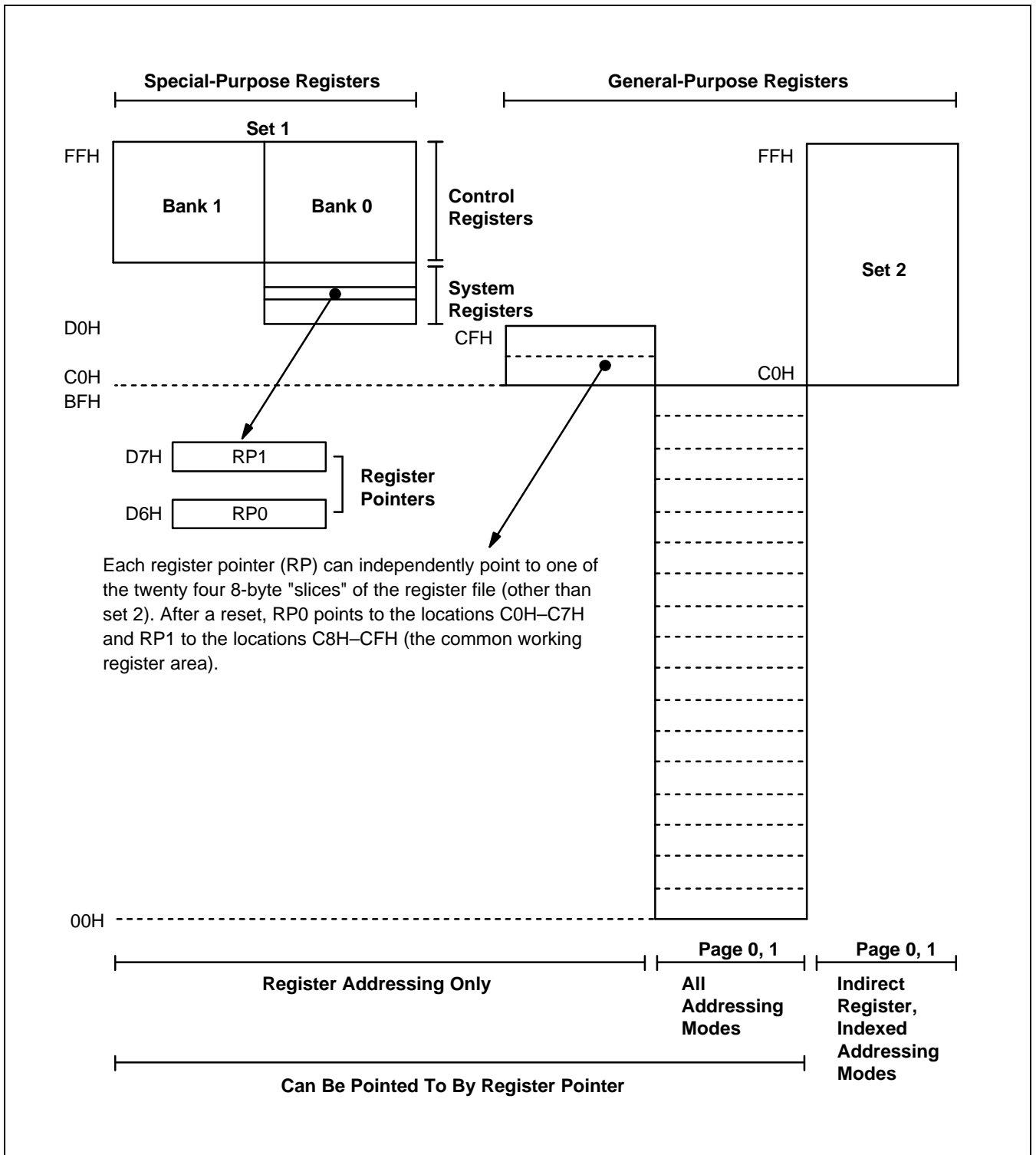


Figure 2-10. Register File Addressing

COMMON WORKING REGISTER AREA (C0H–CFH)

After a reset, the register pointers, RP0 and RP1, automatically point to two 8-byte register slices in set 1, locations C0H–CFH, as the active 16-byte working register block:

- RP0 → C0H–C7H
- RP1 → C8H–CFH

This 16-byte address range is a *common area*. That is, locations in this area can be accessed using working register addressing only.

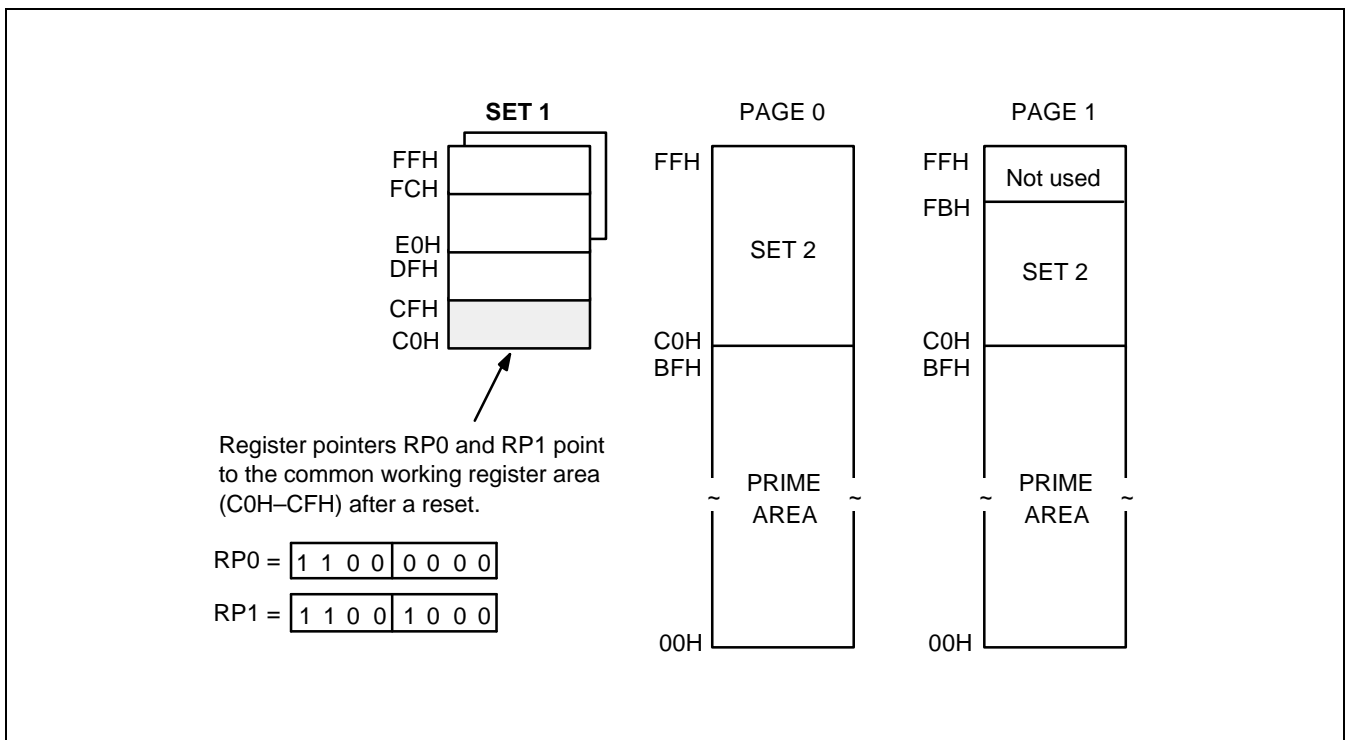


Figure 2-11. Common Working Register Area

PROGRAMMING TIP – Addressing the Common Working Register Area

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

Examples:

- LD 0C2H,40H ; Invalid addressing mode!
 Use working register addressing instead:
 SRP #0C0H
 LD R2,40H ; R2 (C2H) ← the value in location 40H
- ADD 0C3H,#45H ; Invalid addressing mode!
 Use working register addressing instead:
 SRP #0C0H
 ADD R3,#45H ; R3 (C3H) ← R3 + 45H

4-BIT WORKING REGISTER ADDRESSING

Each register pointer defines a movable 8-byte slice of working register space. The address information stored in a register pointer serves as an addressing "window" that enables instructions to access working registers very efficiently using short 4-bit addresses. When an instruction addresses a location in the selected working register area, the address bits are concatenated in the following way to form a complete 8-bit address:

- The high-order bit of the 4-bit address selects one of the register pointers ("0" selects RP0; "1" selects RP1);
- The five high-order bits in the register pointer select an 8-byte slice of the register space;
- The three low-order bits of the 4-bit address select one of the eight registers in the slice.

As shown in Figure 2-12, the net effect of this operation is that the five high-order bits from the register pointer are concatenated with the three low-order bits from the instruction address to form a complete address. As long as the address stored in the register pointer remains unchanged, the three bits from the address will always point to an address in the same 8-byte register slice.

Figure 2-13 shows a typical example of 4-bit working register addressing: the high-order bit of the instruction 'INC R6' is "0", which selects RP0. The five high-order bits stored in RP0 (01110B) are concatenated with the three low-order bits of the instruction's 4-bit address (110B) to produce the register address 76H (01110110B).

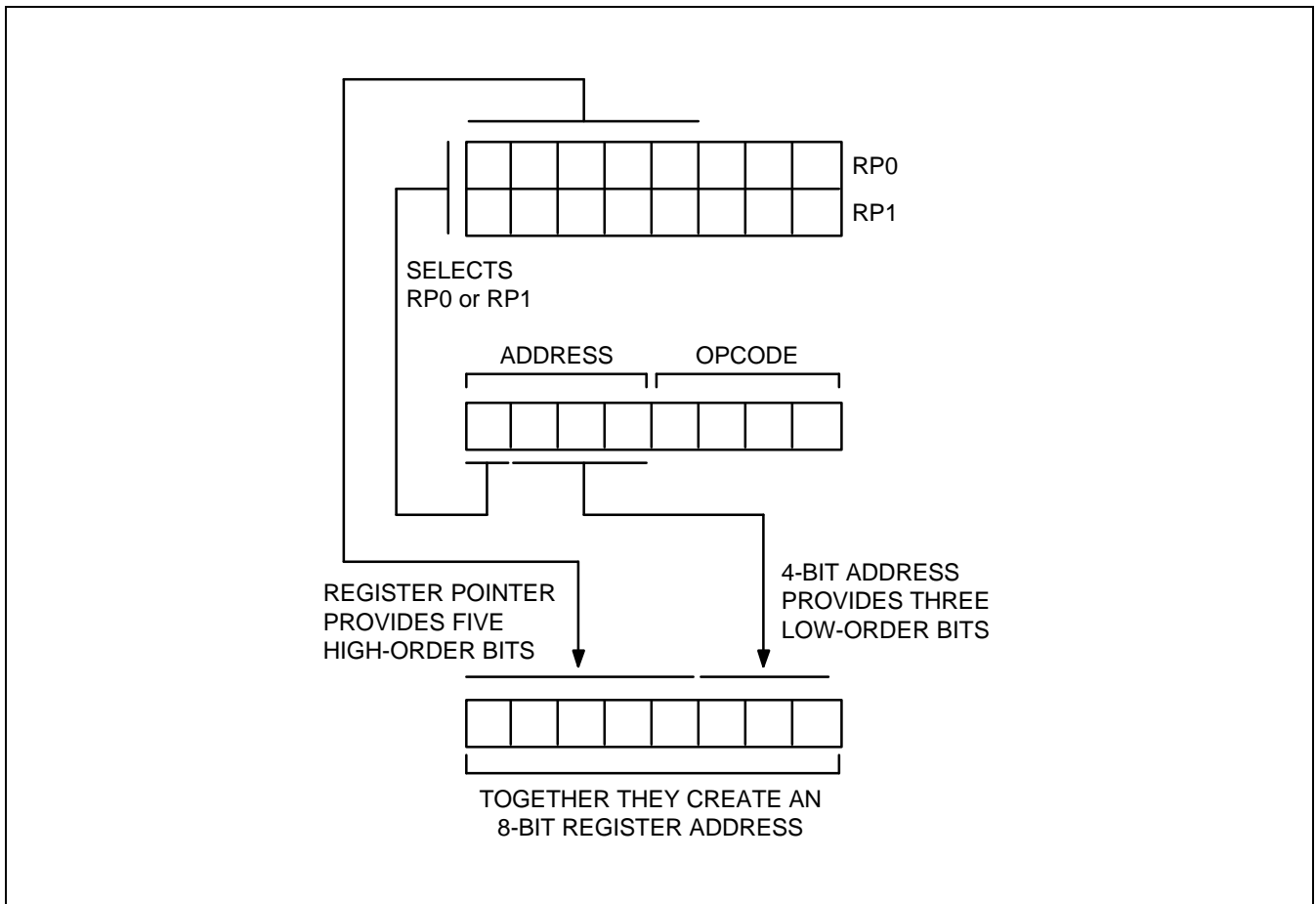


Figure 2-12. 4-Bit Working Register Addressing

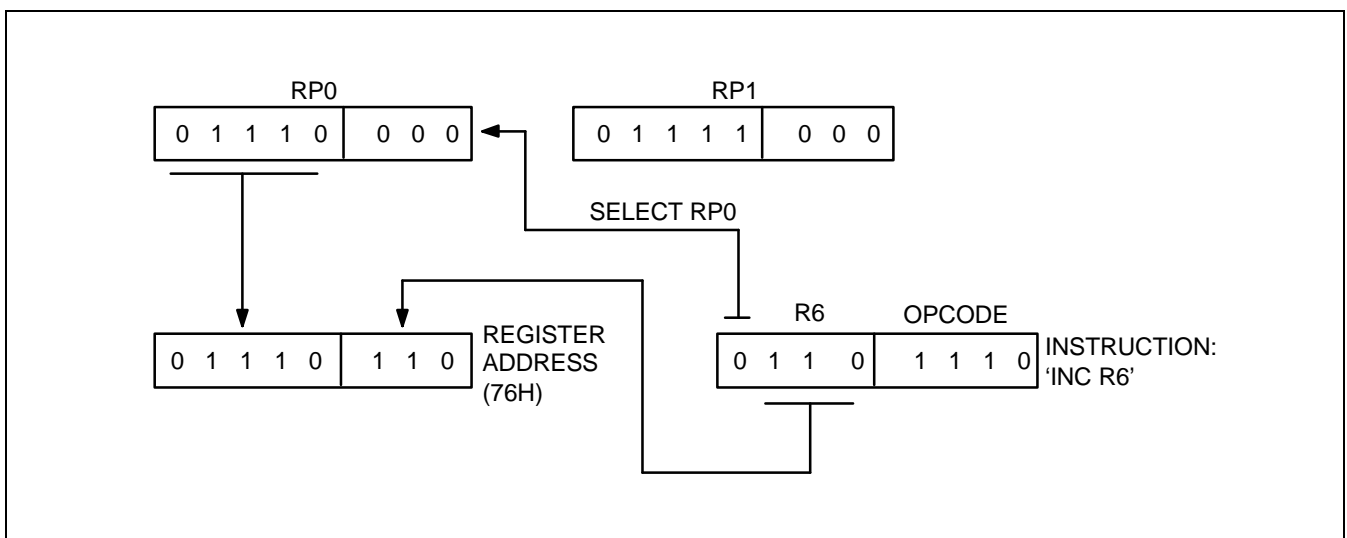


Figure 2-13. 4-Bit Working Register Addressing Example

8-BIT WORKING REGISTER ADDRESSING

You can also use 8-bit working register addressing to access registers in a selected working register area. In order to initiate 8-bit working register addressing, the upper four bits of the instruction address must contain the value 1100B. This 4-bit value (1100B) indicates that the remaining four bits have the same effect as 4-bit working register addressing.

As shown in Figure 2-14, the lower nibble of the 8-bit address is concatenated in much the same way as for 4-bit addressing: bit 3 selects either RP0 or RP1, which then supplies the five high-order bits of the final address, and the three low-order bits of the complete address are provided by the original instruction.

Figure 2-15 shows an example of 8-bit working register addressing: the four high-order bits of the instruction address (1100B) specify 8-bit working register addressing. The fourth bit ("1") selects RP1 and the five high-order bits in RP1 (10100B) become the five high-order bits of the register address. The three low-order bits of the register address (011) are provided by the three low-order bits of the 8-bit instruction address. Together, the five address bits from RP1 and the three address bits from the instruction comprise the complete register address, R163 (10100011B).

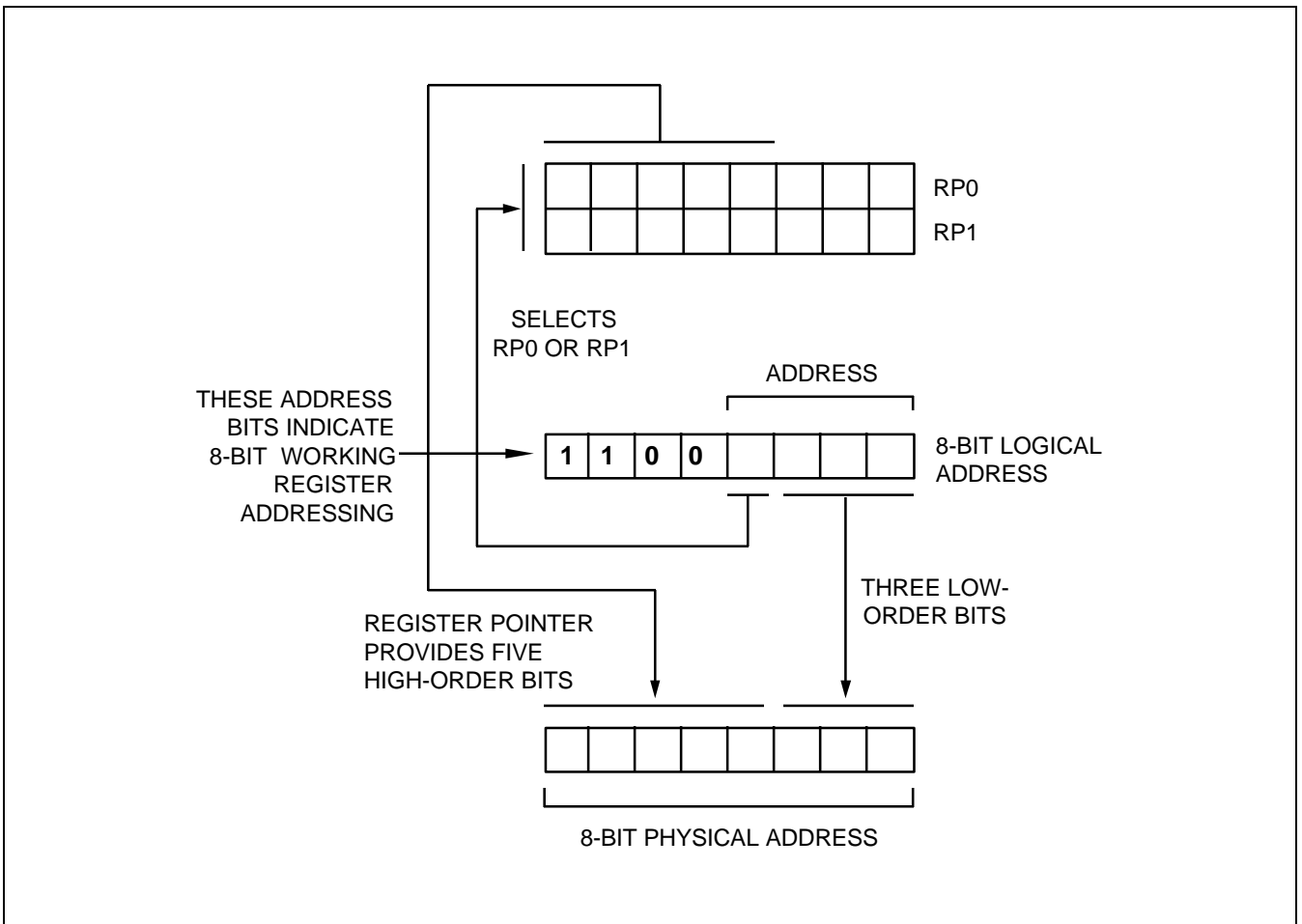


Figure 2-14. 8-Bit Working Register Addressing

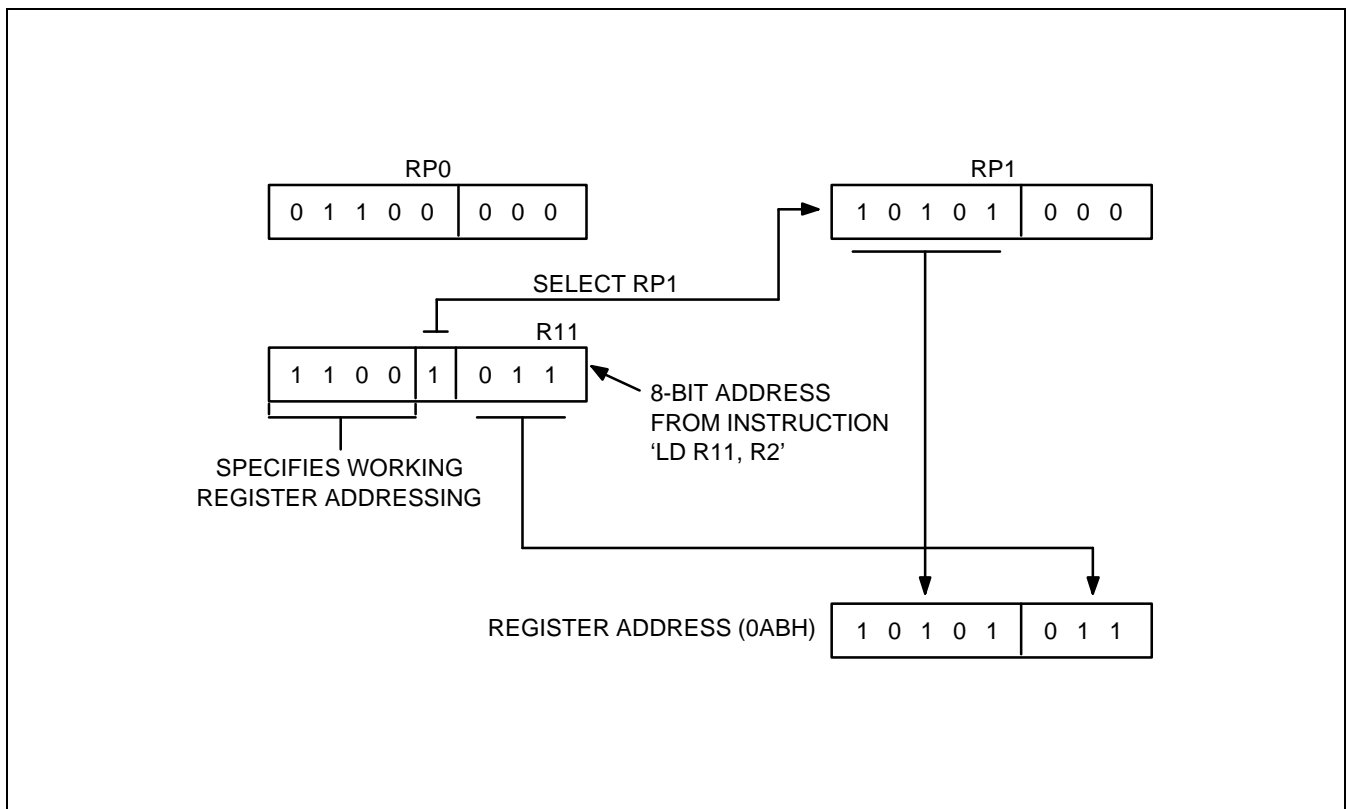


Figure 2-15. 8-Bit Working Register Addressing Example

SYSTEM AND USER STACKS

The S3C8-series microcontrollers use the system stack for subroutine calls and returns, interrupt processing, and data storage. The PUSH and POP instructions support system stack operations. Stack operations in the internal register file and in external data memory are supported by hardware. (The S3C8847/C8849/P8849 do not support an external memory access.) Bit 1 in the external memory timing register EMT selects an internal or external stack area. The 16-bit stack pointer register (SPH, SPL) is used to access an externally defined system stack. An 8-bit stack pointer (SPL) is sufficient for internal stack addressing.

Stack Operations

Return addresses for procedure calls and interrupts and data are stored on the stack. The contents of the PC are saved to stack by a CALL instruction and restored by an RET instruction. When an interrupt occurs, the contents of the PC and the FLAGS register are pushed to the stack. The IRET instruction then pops these values back to their original locations. The stack address is always decremented *before* a push operation and incremented *after* a pop operation. The stack pointer (SP) always points to the stack frame stored on the top of the stack, as shown in Figure 2-16.

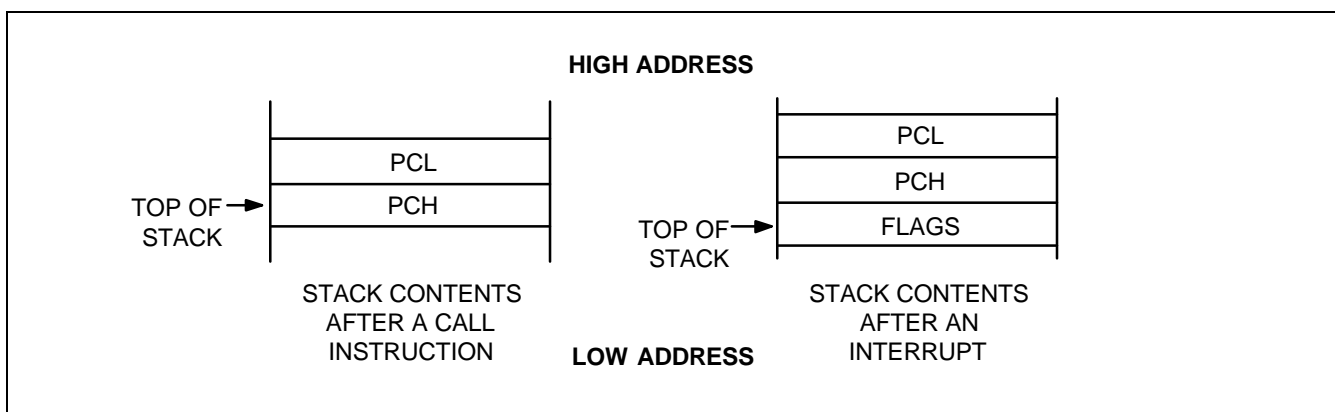


Figure 2-16. Stack Operations

User-Defined Stacks

You can freely define stacks in the internal register file as data storage locations. The instructions, PUSHUI, PUSHUD, POPUI, and POPUD, support user-defined stack operations.

Stack Pointers (SPL, SPH)

The register locations D8H and D9H contain the 16-bit stack pointer (SP) value. The most significant byte of a 16-bit stack address is stored in the SPH register (D8H) and the least significant byte is stored in the SPL register (D9H). Because an external memory interface is not implemented for the S3C8847/C8849/P8849 microcontrollers, a single 8-bit stack pointer (SPL) is sufficient to address stack locations in the internal register file.

After a reset, the stack pointer value is undetermined. The SPL register must then be initialized to an 8-bit value in the range 00H–FFH, page 0.

You can use the SPH register as a general-purpose data register. Please note that when you do so, data stored in SPH may be overwritten if an overflow or underflow of the SPL register occurs during normal stack operations. To prevent this, you can initialize the SPL value to FFH instead of 00H.

 PROGRAMMING TIP – Standard Stack Operations Using PUSH and POP

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

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

3 ADDRESSING MODES

OVERVIEW

Instructions that are stored in program memory are fetched for execution using the program counter. Instructions indicate the operation to be performed and the data to be operated on. *Addressing mode* is used to determine the location of the data operand. The operands specified in SAM87 instructions may be condition codes, immediate data, or a location in the register file, program memory, or data memory.

The SAM87 instruction set supports seven explicit addressing modes. Not all of these addressing modes are available for each instruction. The addressing modes and their symbols are as follows:

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

REGISTER ADDRESSING MODE (R)

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

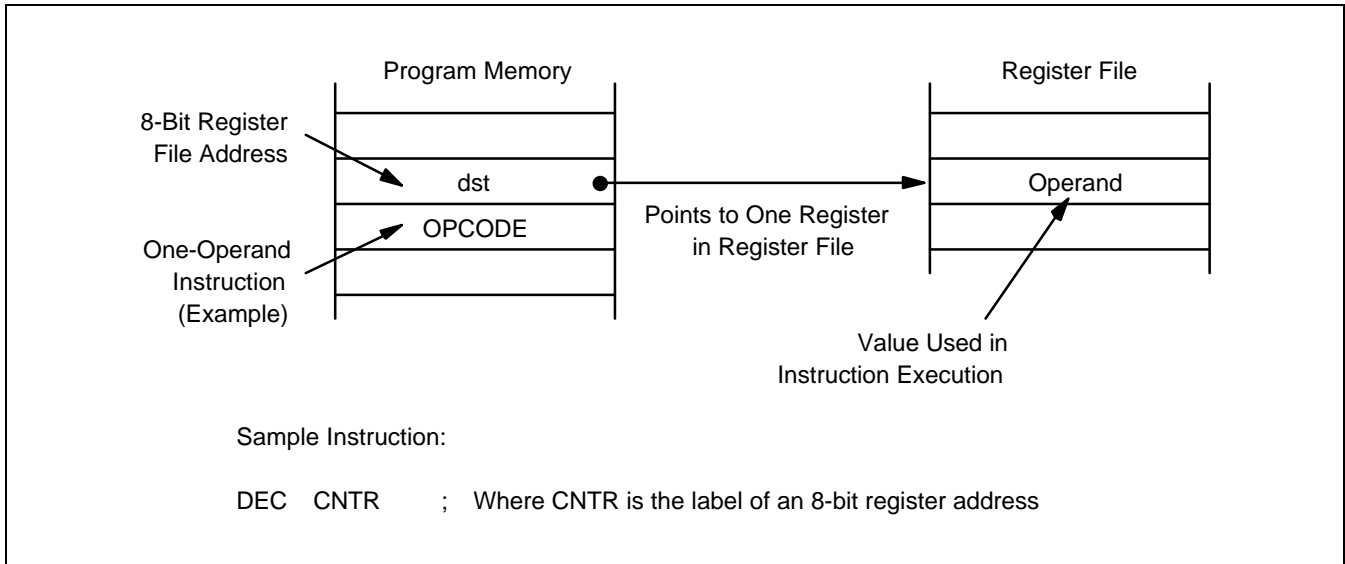


Figure 3-1. Register Addressing

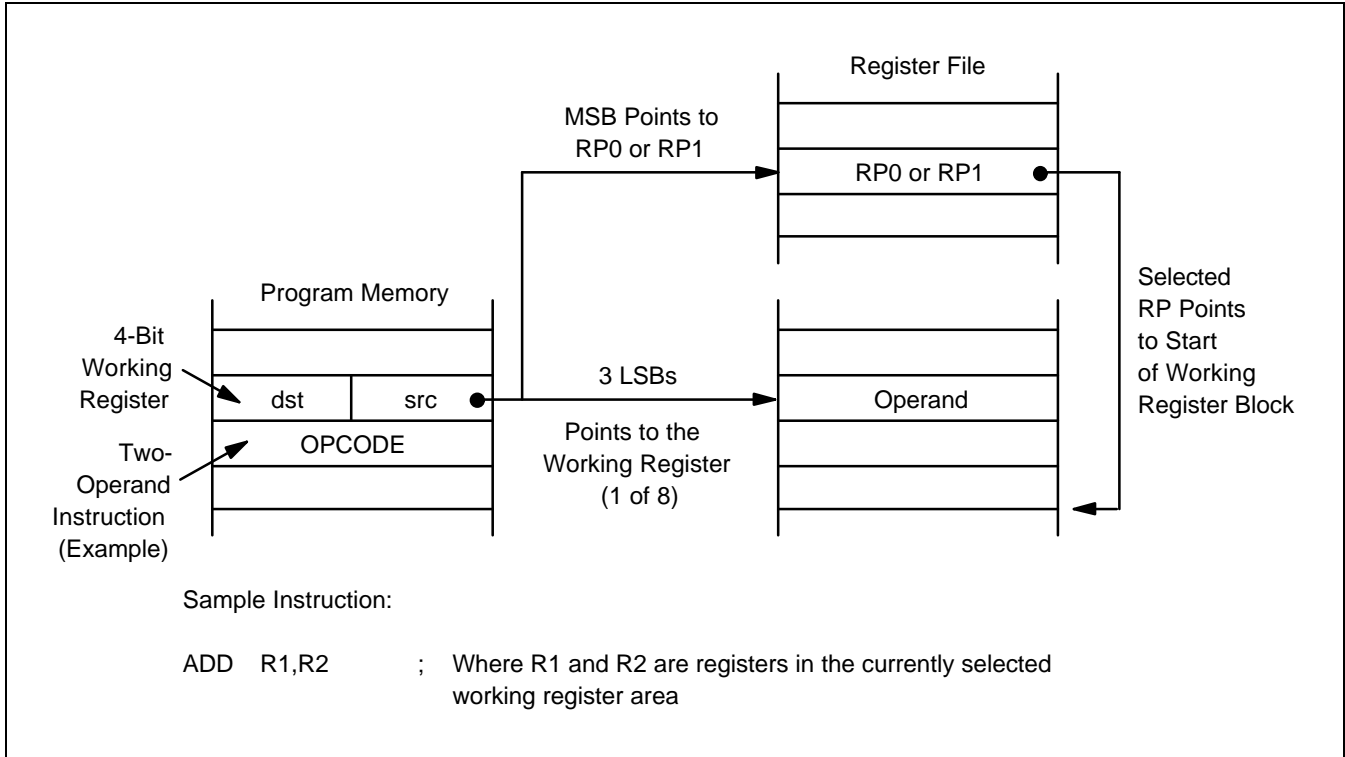


Figure 3-2. Working Register Addressing

INDIRECT REGISTER ADDRESSING MODE (IR)

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

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

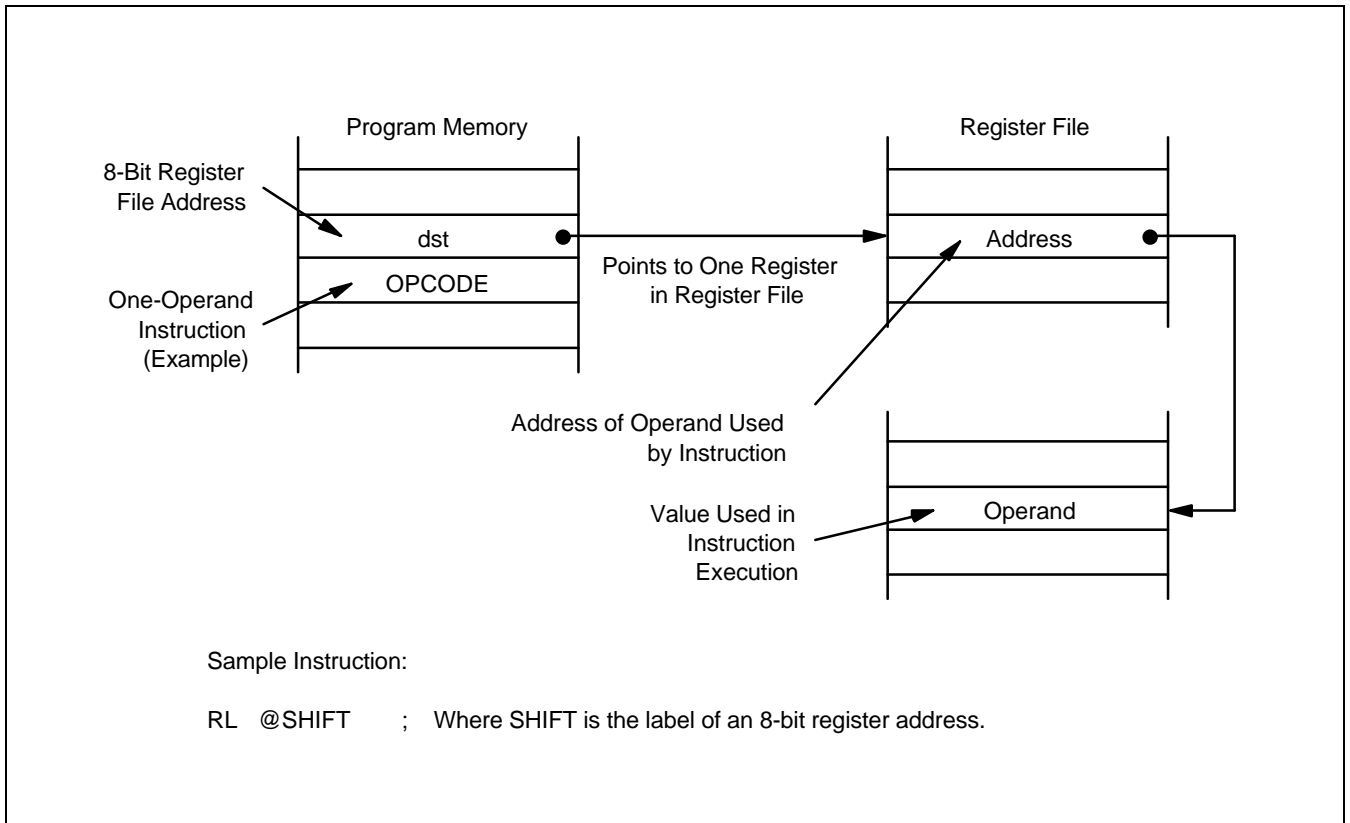


Figure 3-3. Indirect Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Continued)

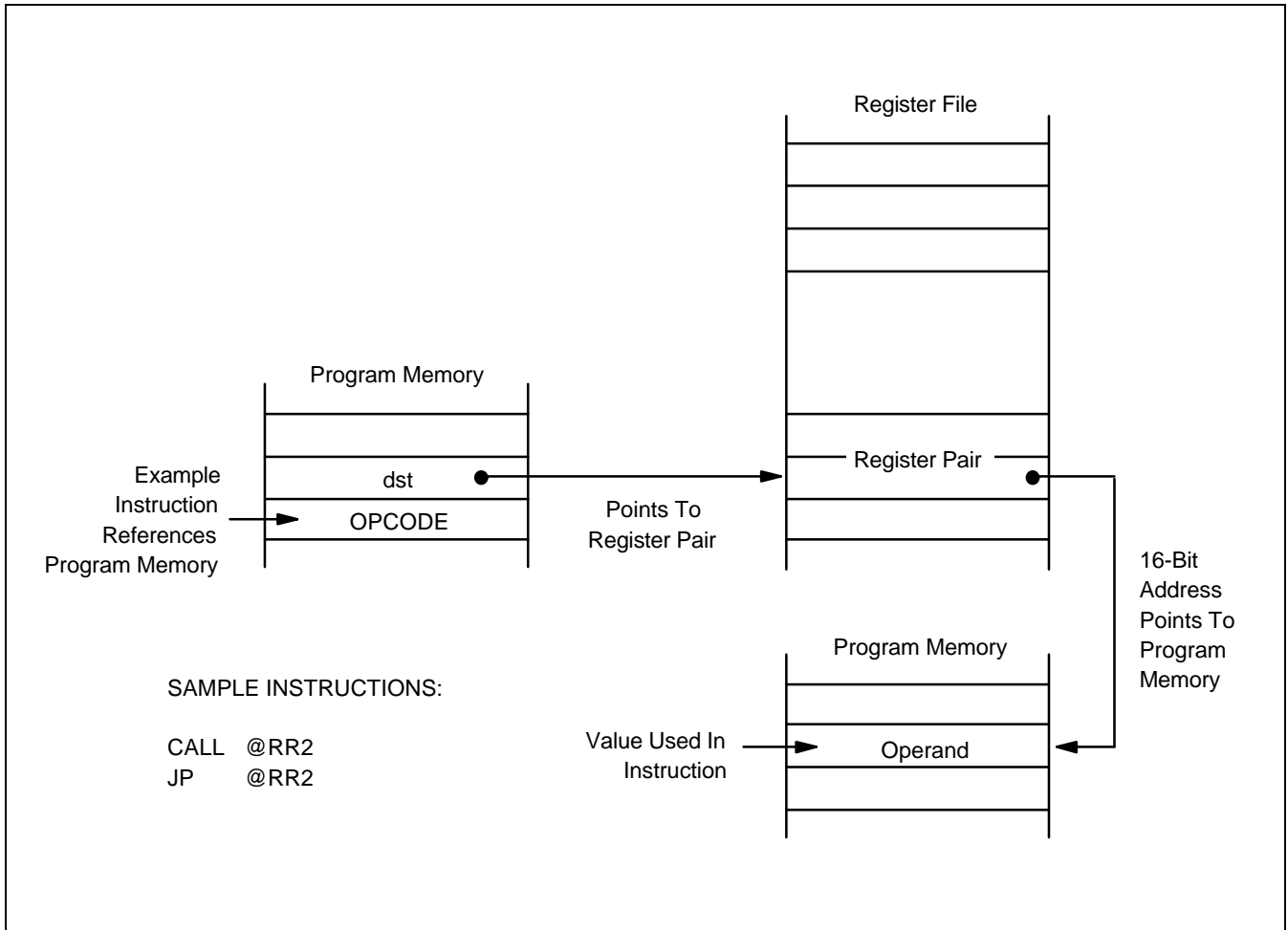


Figure 3-4. Indirect Register Addressing to Program Memory

INDIRECT REGISTER ADDRESSING MODE (Continued)

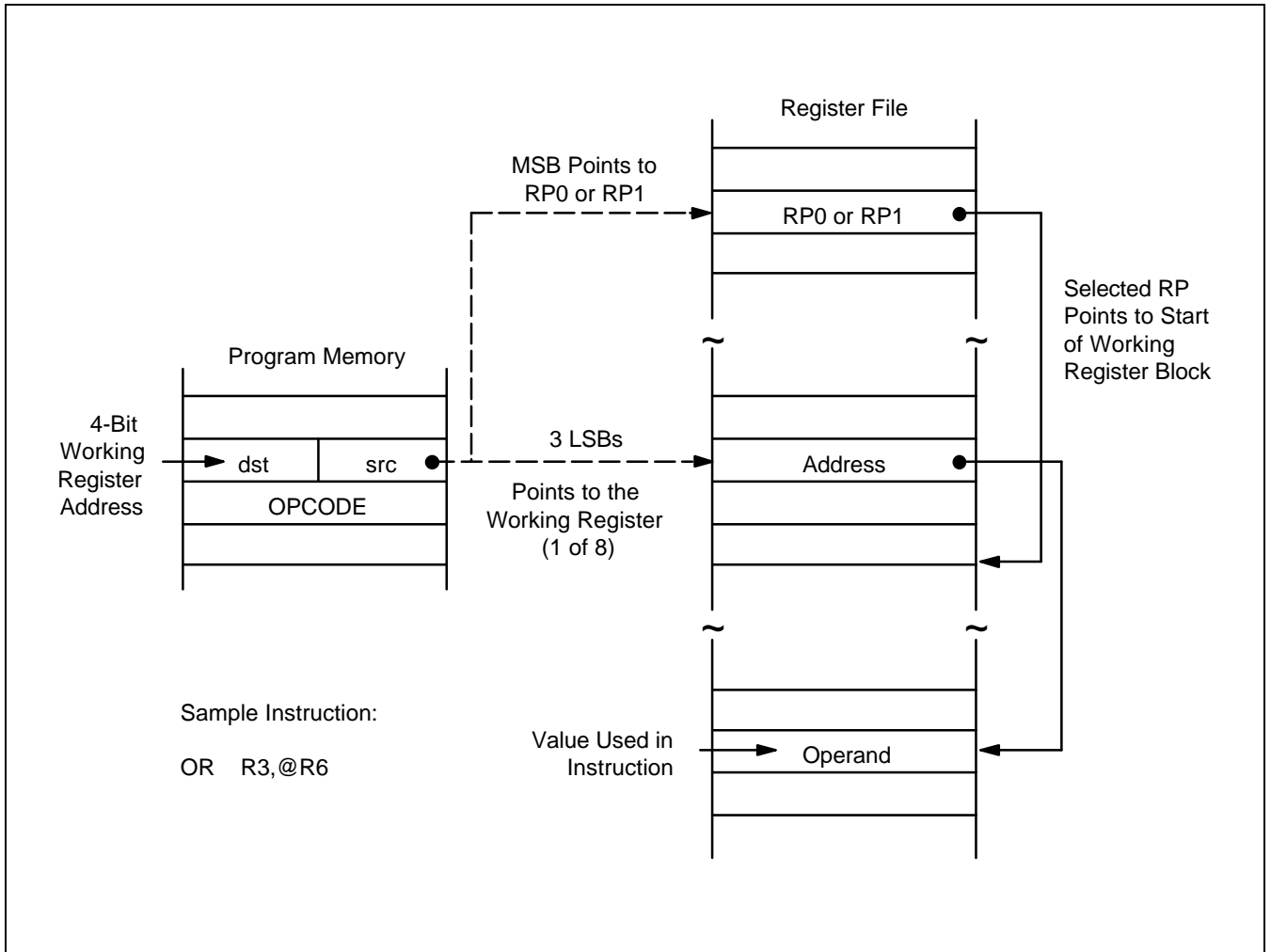


Figure 3-5. Indirect Working Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Concluded)

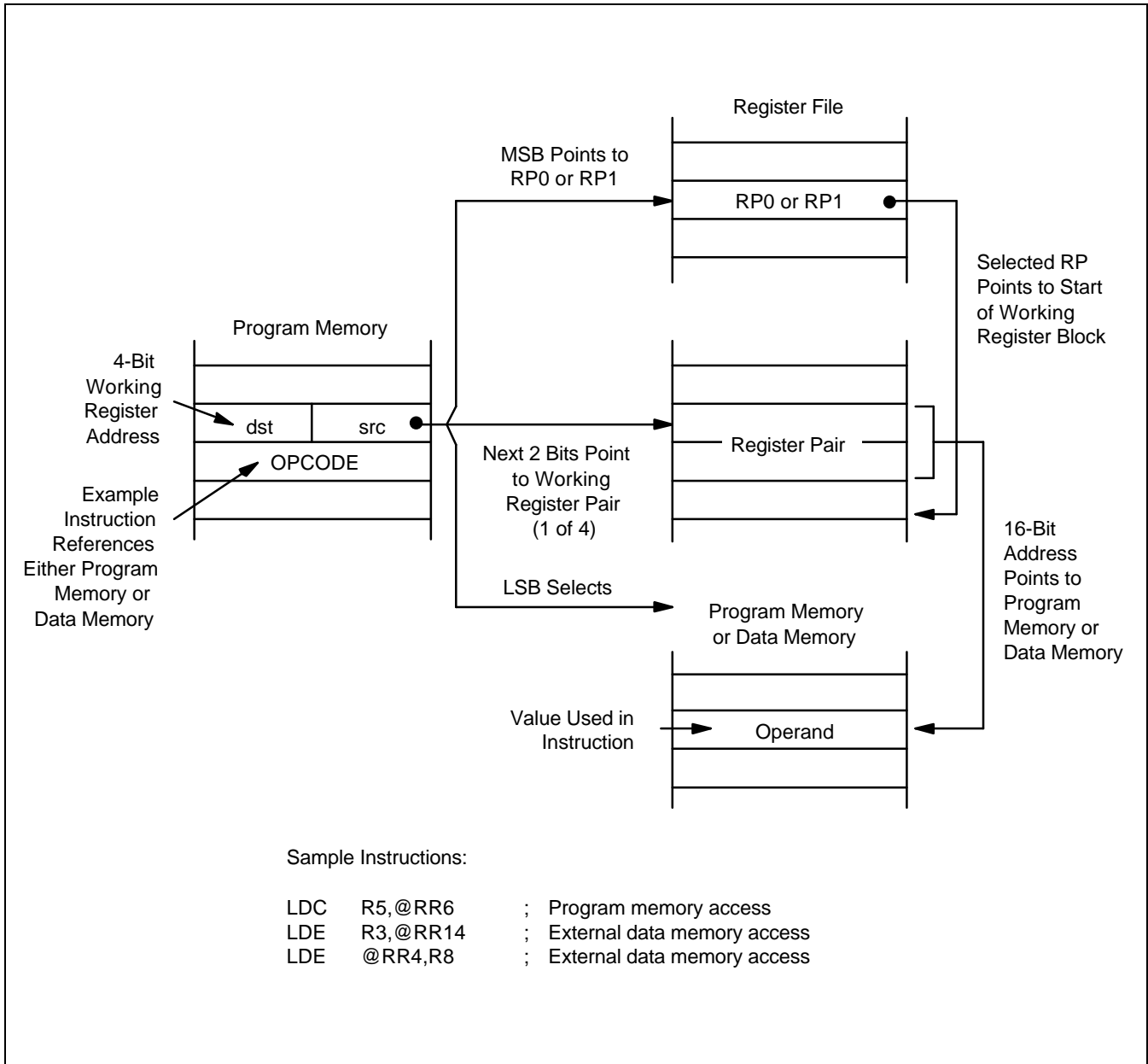


Figure 3-6. Indirect Working Register Addressing to Program or Data Memory

INDEXED ADDRESSING MODE (X)

Indexed (X) addressing mode adds an offset value to a base address during the instruction execution in order to calculate the effective operand address (see Figure 3-7). You can use Indexed addressing mode to access locations in the internal register file or in external memory. You cannot, however, access the locations C0H–FFH in set 1 using Indexed addressing mode.

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

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

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

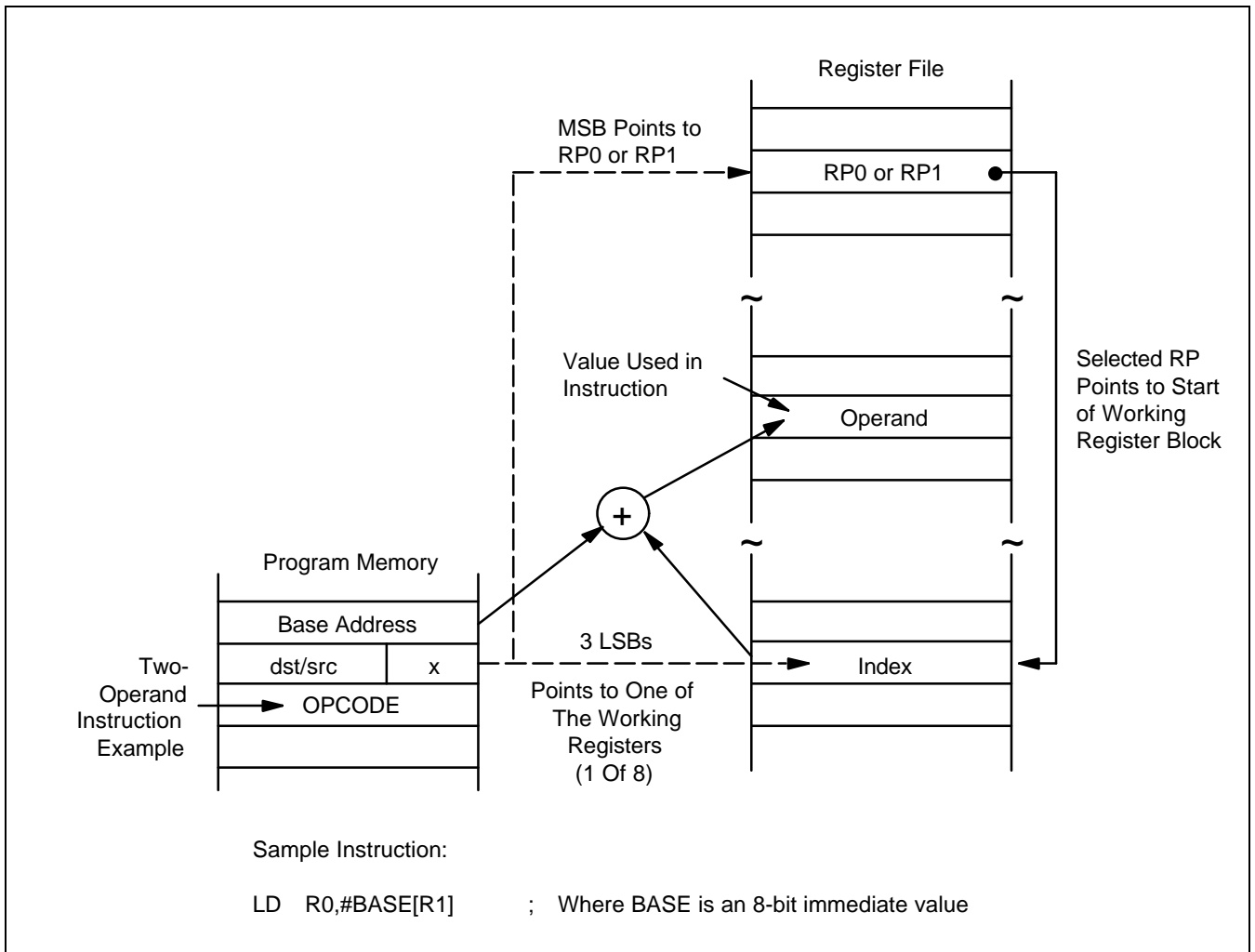


Figure 3-7. Indexed Addressing to Register File

INDEXED ADDRESSING MODE (Continued)

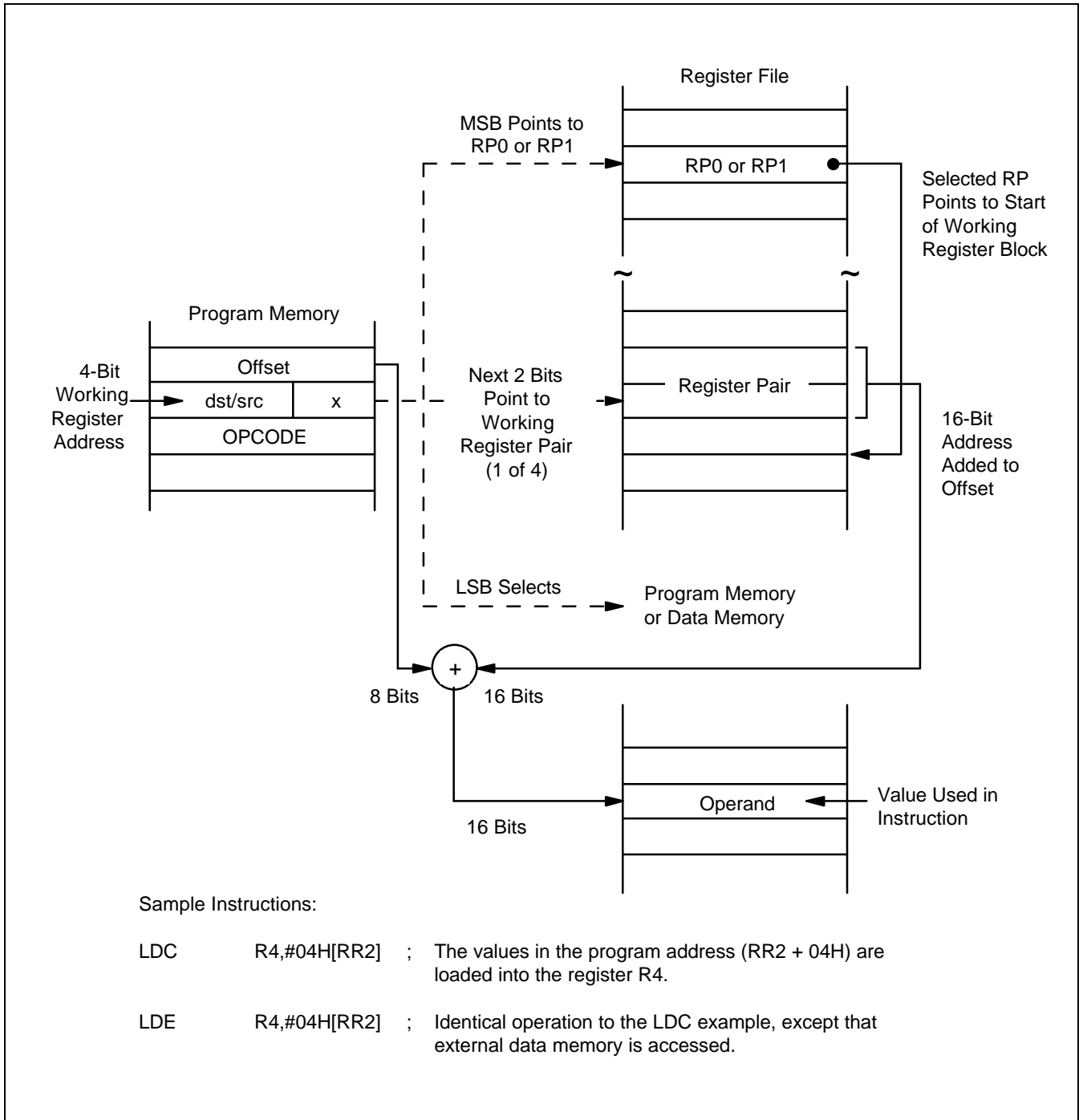


Figure 3-8. Indexed Addressing to Program or Data Memory with Short Offset

INDEXED ADDRESSING MODE (Concluded)

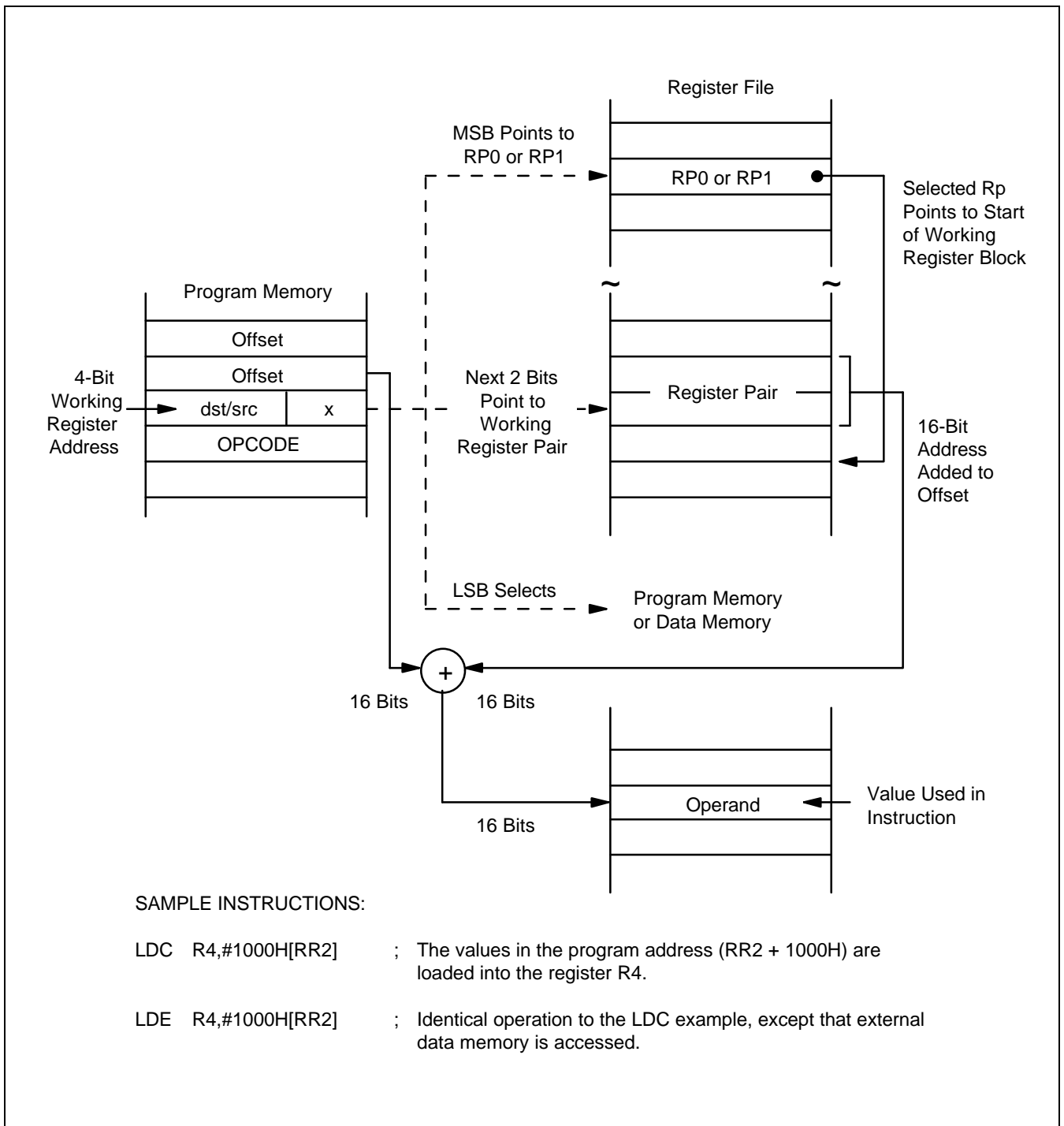


Figure 3-9. Indexed Addressing to Program or Data Memory

DIRECT ADDRESS MODE (DA)

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

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

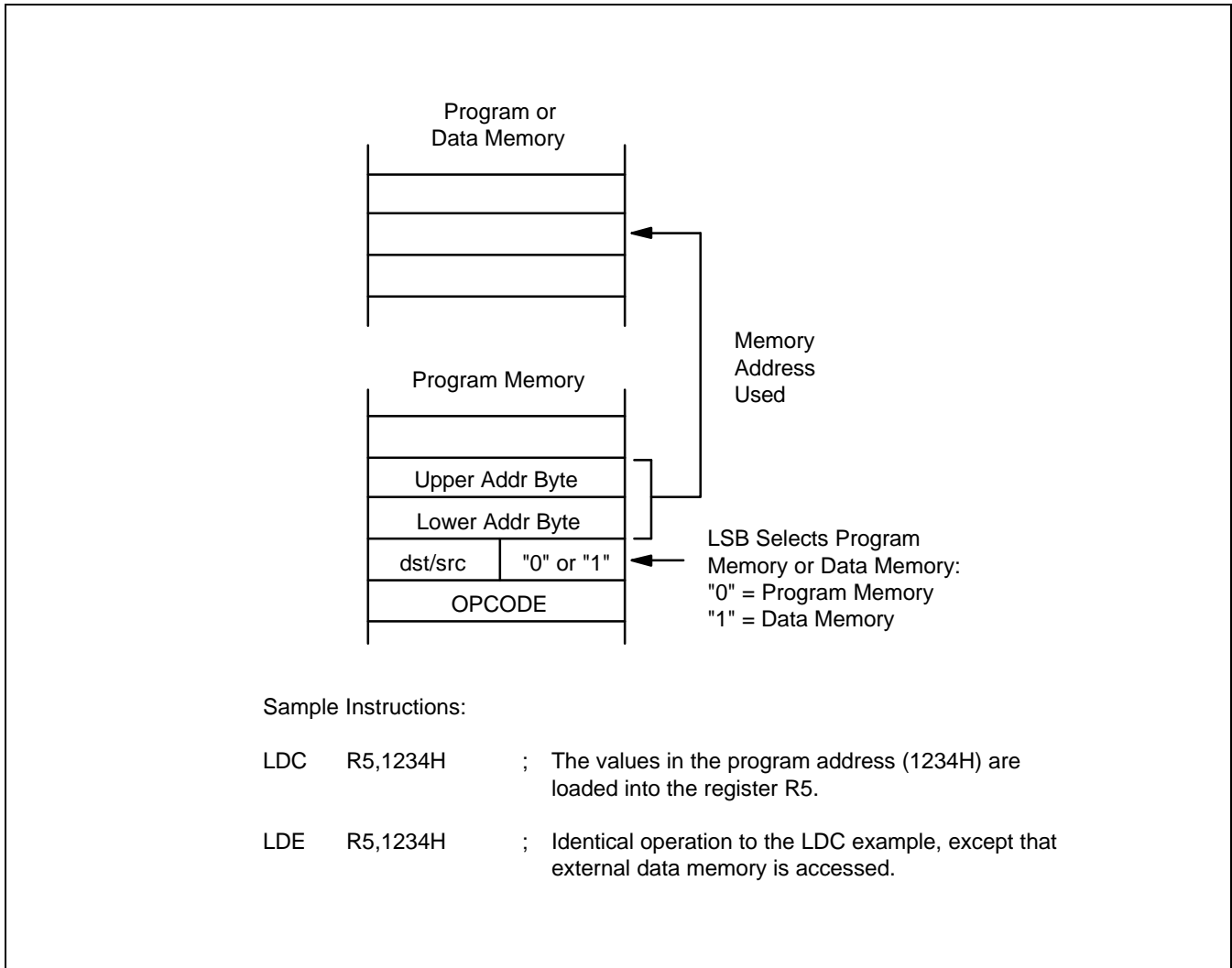


Figure 3-10. Direct Addressing for Load Instructions

DIRECT ADDRESS MODE (Continued)

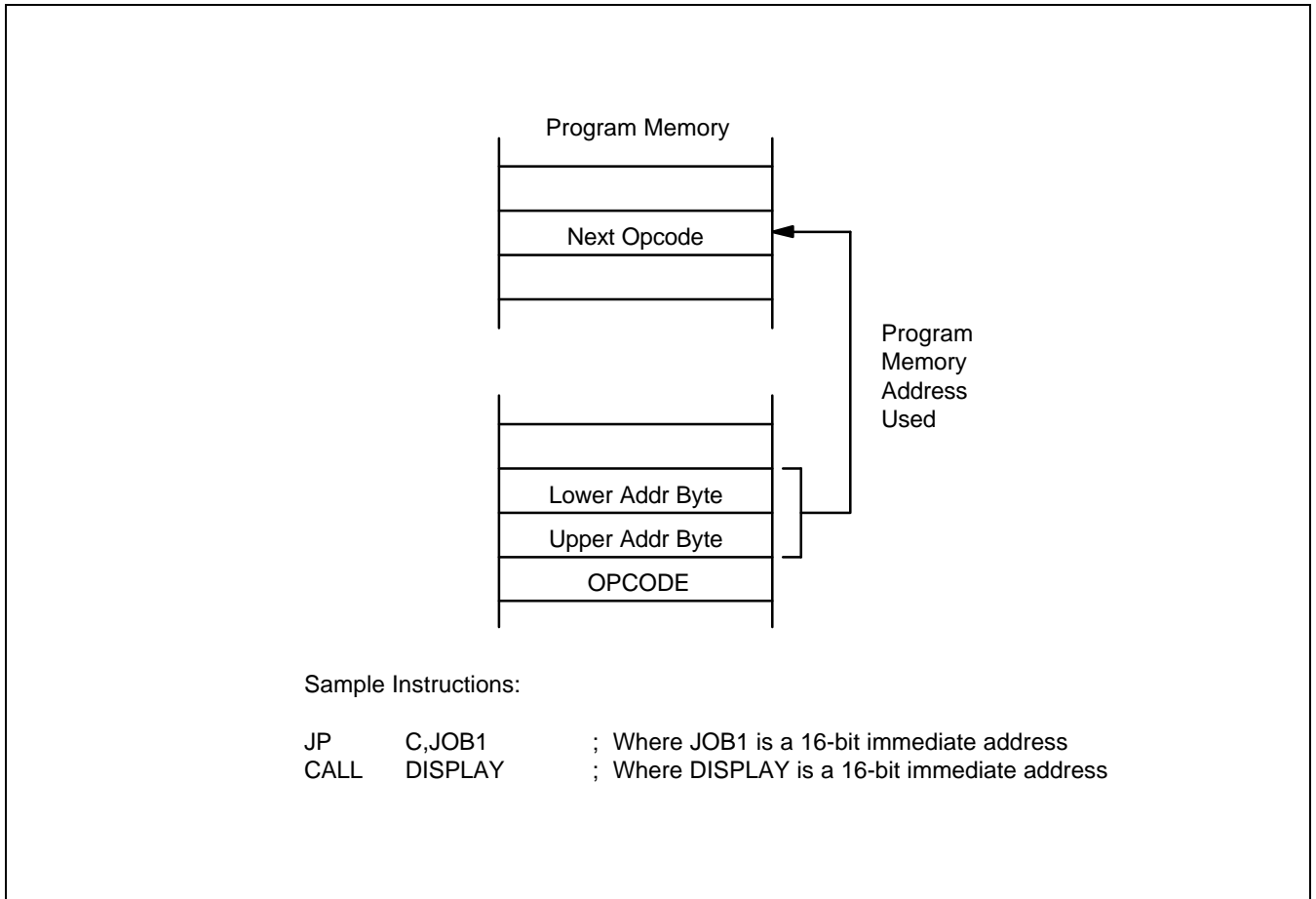


Figure 3-11. Direct Addressing for Call and Jump Instructions

INDIRECT ADDRESS MODE (IA)

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

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

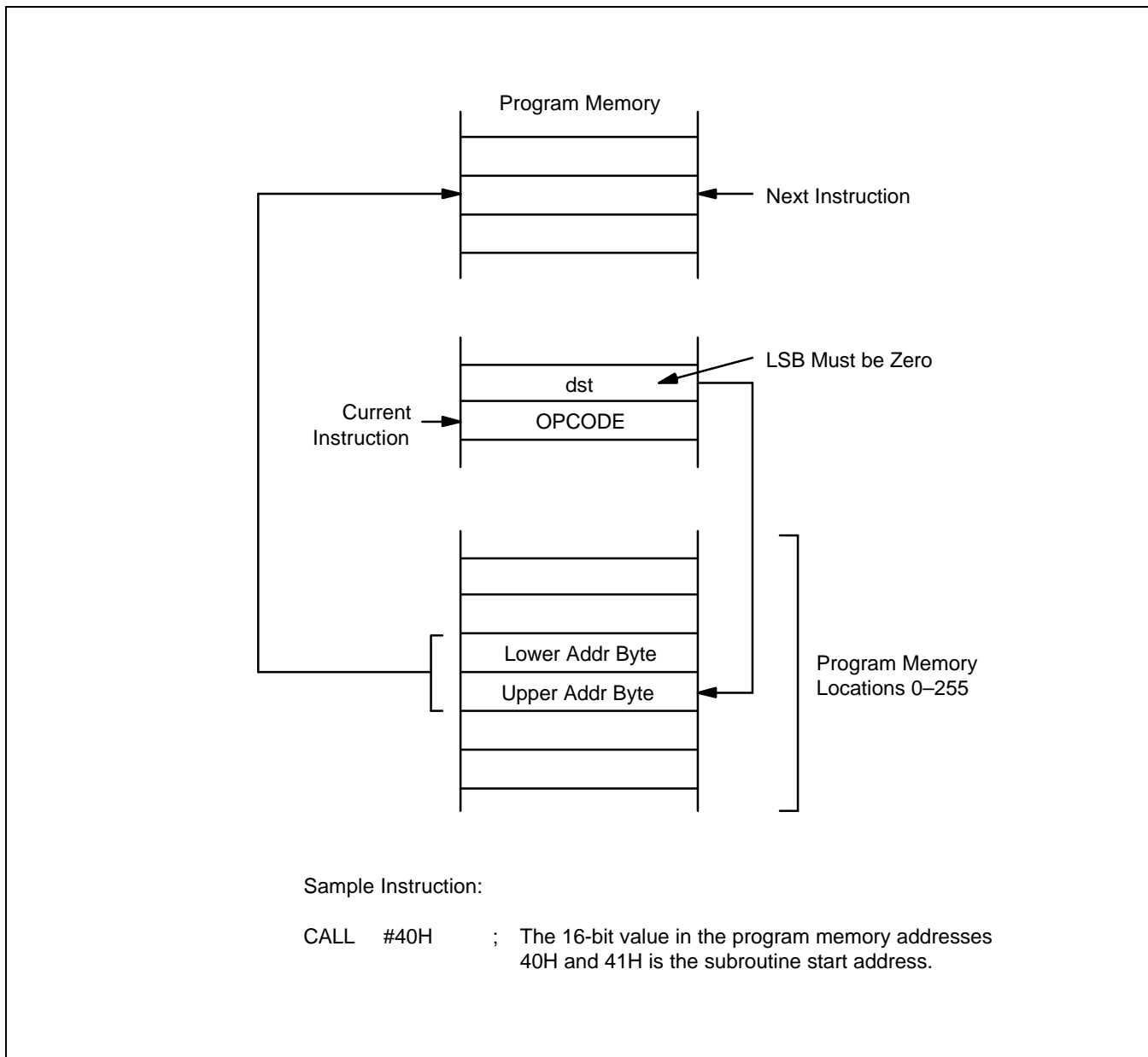


Figure 3-12. Indirect Addressing

RELATIVE ADDRESS MODE (RA)

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

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

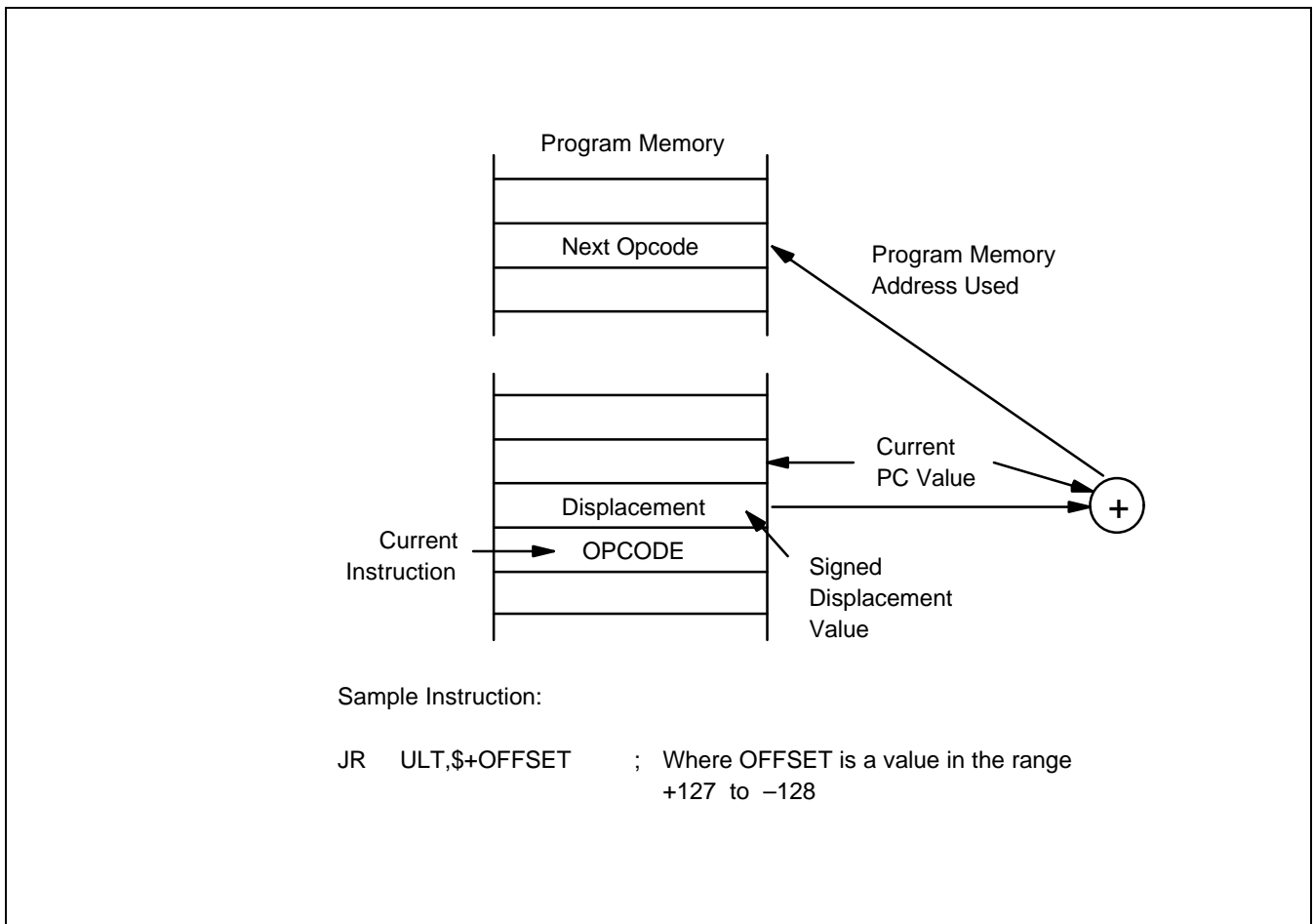


Figure 3-13. Relative Addressing

IMMEDIATE MODE (IM)

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

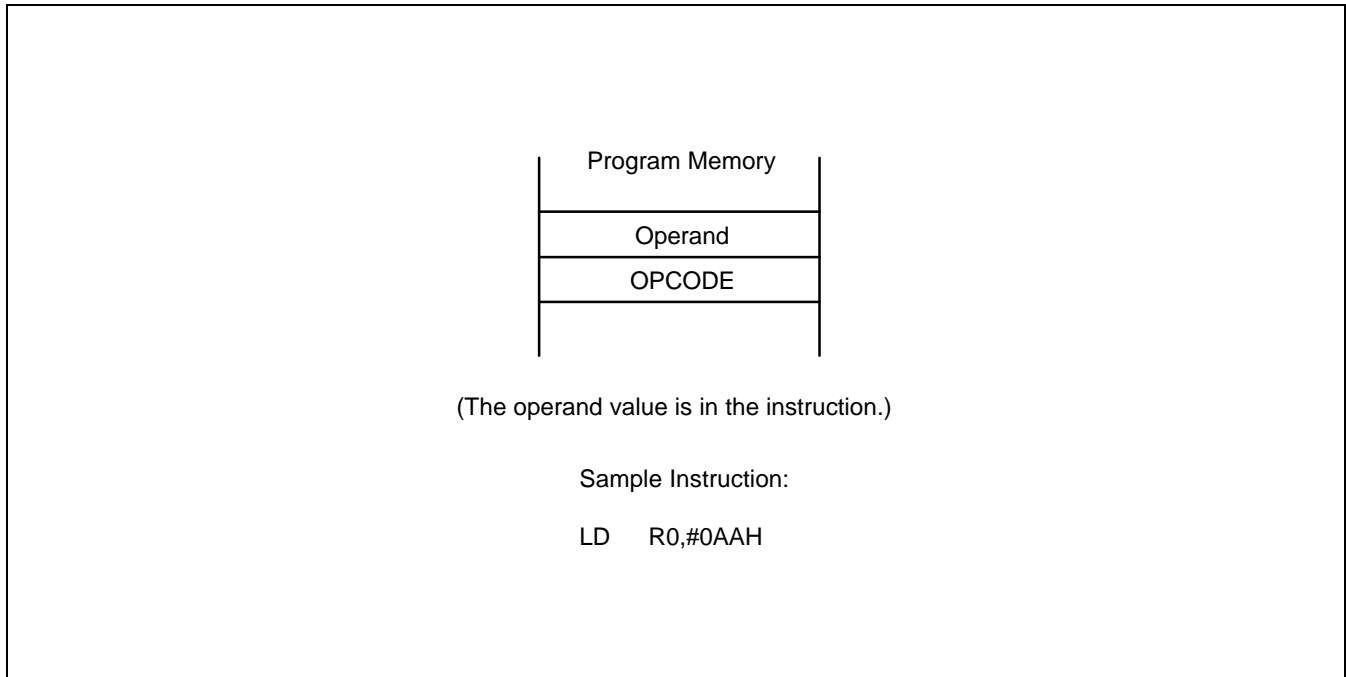


Figure 3-14. Immediate Addressing

4 CONTROL REGISTERS

OVERVIEW

In this chapter, detailed descriptions of the S3C8847/C8849/P8849 control registers are presented in an easy-to-read format. These descriptions will help familiarize you with the mapped locations in the register file. You can also use them as a quick-reference source when writing application programs.

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

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

Table 4-1. Set 1 Registers

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

Table 4-2. Set 1, Bank 0 Registers

Register Name	Mnemonic	Decimal	Hex	R/W
Port 0 data register	P0	224	E0H	R/W
Port 1 data register	P1	225	E1H	R/W
Port 2 data register	P2	226	E2H	R/W
Port 3 data register	P3	227	E3H	R/W
Port 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
Locations EAH in set 1, bank 0, are not mapped.				
Port 3 control register (low byte)	P3CONL	235	EBH	R/W
Locations ECH – EFH in set 1, bank 0, are not mapped.				
Timer A data register	TADATA	240	F0H	R/W
Locations F1H in set 1, bank 0, are not mapped.				
Timer A control register	TACON	242	F2H	R/W
Locations F3H in set 1, bank 0, are not mapped.				

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

Register Name	Mnemonic	Decimal	Hex	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 (note)
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 bit3 – bit0 of the ADCON register are read-only.

Table 4-3. 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 (note)
Halftone signal control register	HTCON	246	F6H	R/W
V-SYNC blank control register	VSBCON	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
OSD color buffer	COLBUF	252	FCH	R/W
Locations FDH–FFH in set 1, bank 1, are not mapped.				

NOTE: Bit7 – bit4 of the DSPCON register are read-only.

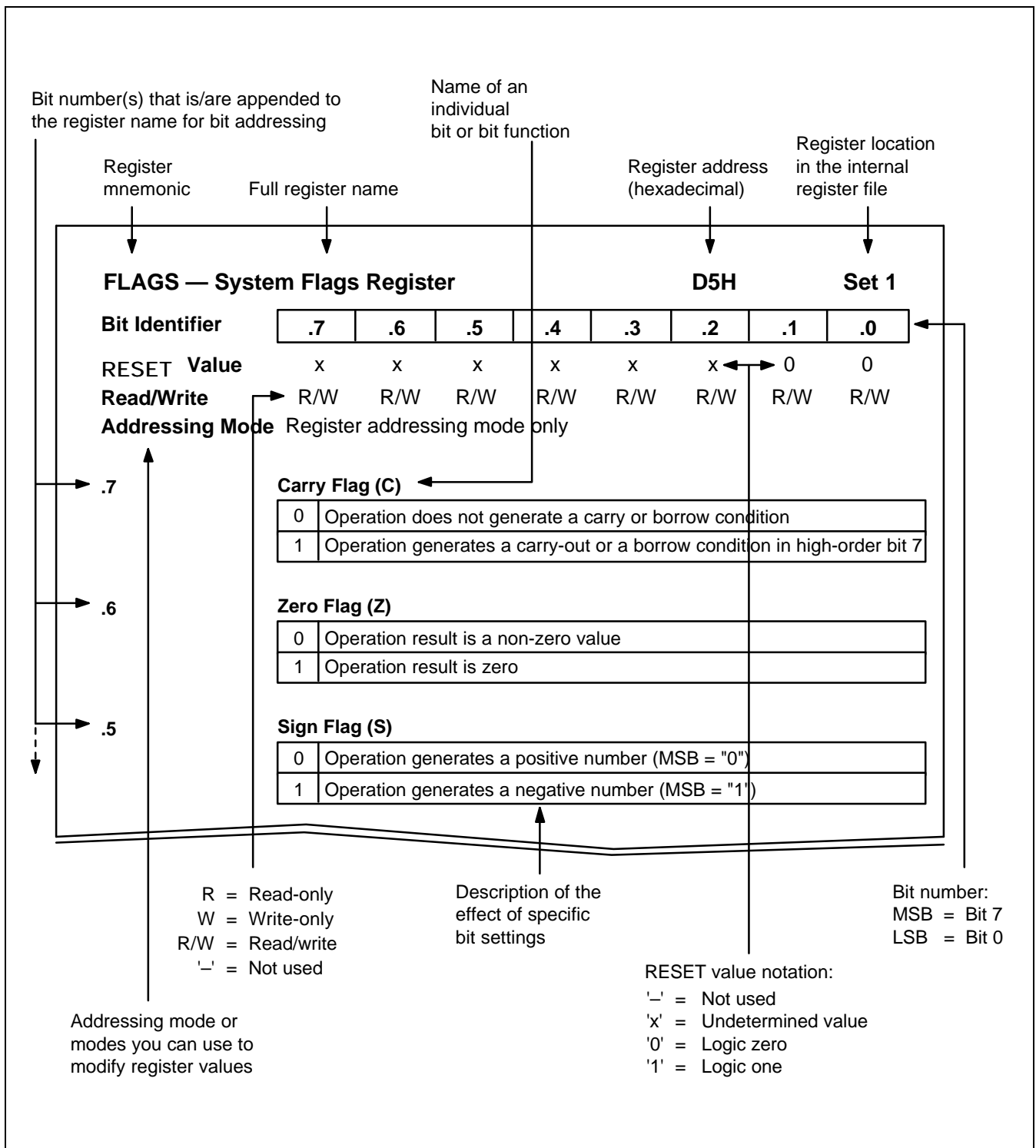


Figure 4-1. Register Description Format

ADCON — A/D Converter Control Register

FAH

Set 1, Bank 0

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

.7 **End-of-Conversion Status Flag (Read-Only)**

0	Conversion now in progress
1	Conversion is complete

.6 Not used for the S3C8847/C8849/P8849.**.5 and .4**

A/D Converter Input Pin Selection Bits		
0	0	ADC0 (P3.0)
0	1	ADC1 (P3.1)
1	0	ADC2 (P0.6)
1	1	ADC3 (P0.7)

.3–.0 **4-Bit Digital Conversion Result (Read-Only)****NOTES:**

1. The end-of-conversion status flag (bit 7) and the 4-bit digital result (lower nibble) are read-only.
2. The analog-to-digital conversion procedure starts when you write bit 4 and bit 5 to select the ADC input pin.

BTCON — Basic Timer Control Register

D3H

Set 1

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

.7–.4

Watchdog Timer Function Disable Code (for Reset)

1	0	1	0	Disable watchdog timer function
Others				Enable watchdog timer function

.3 and .2

Basic Timer Input Clock Selection Bits

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

.1

Basic Timer Counter Clear Bit (note)

0	No effect
1	Clear the basic timer counter value

.0

Clock Divider Clear Bit for Basic Timer and Timer 0 (note)

0	No effect
1	Clear both dividers

NOTE: When you write a "1" to bit 0 or bit 1, the corresponding divider or counter value is cleared to '00H'. The corresponding BTCON bit is then automatically reset by hardware to "0".

CHACON — OSD Character Size Control Register

F0H

Set 1, Bank 1

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

.7 and .6

Vertical Character Size Selection Bits

0	0	Select 'x1' vertical character size
0	1	Select 'x2' vertical character size
1	0	Select 'x3' vertical character size
1	1	Select 'x4' vertical character size

.5 and .4

Horizontal Character Size Selection Bits

0	0	Select 'x1' horizontal character size
0	1	Select 'x2' horizontal character size
1	0	Select 'x3' horizontal character size
1	1	Select 'x4' horizontal character size

.3–.0

Fade Row Address Selection for Rows 0–11 in On-Screen Display

0	0	0	0	Row 0 selected
0	0	0	1	Row 1 selected
0	0	1	0	Row 2 selected
0	0	1	1	Row 3 selected
0	1	0	0	Row 4 selected
0	1	0	1	Row 5 selected
0	1	1	0	Row 6 selected
0	1	1	1	Row 7 selected
1	0	0	0	Row 8 selected
1	0	0	1	Row 9 selected
1	0	1	0	Row 10 selected
1	0	1	1	Row 11 selected
Others				Invalid selection



CLKCON — System Clock Control Register

D4H

Set 1

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

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

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

.6 and .5 **Main Oscillator Stop Control Bits**

0	0	No effect
0	1	No effect
1	0	Stop main oscillator
1	1	No effect

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

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

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

1	0	1	Invalid selection for S3C8847/C8849/P8849
Others			Select main system clock (MCLK)

NOTES:

- After a reset, the slowest clock (divide by 16) is selected as the system clock. To select faster clock speeds, load the appropriate values to CLKCON.3 and CLKCON.4.
- These selection bits are required only for systems that have a main clock and a subsystem clock. The S3C8847 and the S3C8849 microcontrollers have only a main oscillator (and an L-C oscillator for the OSD module). For this reason, the setting '101B' is invalid.

CLMCON — OSD Column Control Register

F3H

Set 1, Bank 1

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

.7–.3

Left Margin Display Position Control Bits (16 + 4 x LMG value of 0–31 dots)

0	0	0	0	0	Left margin = 16 dot clocks
0	0	0	0	1	Left margin = 16 + 4 × 1 dot clock
...					...
1	1	1	1	1	Left margin = 16 + 4 × 31 dot clocks

.2–.0

Inter-Column Spacing Control Selection (0–7 dots)

0	0	0	No inter-column spacing		
0	0	1	Inter-column spacing = 1 dot		
...			...		
1	1	1	Inter-column spacing = 7 dots		

NOTE: To set left margin and inter-column spacing, separate decimal values must be calculated, converted to their binary equivalents, and then written to the CLMCON register.



COLBUF — OSD Character Color Buffer

FCH

Set 1, Bank 1

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

.7–.5

Not used for the S3C8847/C8849/P8849.

.4 **Video RAM Bit-8 Enable Bit**

0	Disable VRAM bit-8
1	Enable VRAM bit-8 (Video ROM code = 256–383)

.3 **H/T and BGRND Enable Bit**

0	Disable H/T and BRGND
1	Enable H/T and BRGND

.2–.0 **Character Color Selection Bits (.2 = Red, .1 = Green, .0 = Blue)**

0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

NOTE: When a character code is written to the OSD video RAM, the color buffer value is automatically loaded to the bits 9–13 in the video RAM. Therefore, in order to change the character color, the color buffer value should be updated first. When a character code is read from the video RAM, the color data (RGB values) of the character is automatically loaded to the color buffer.

COLCON — OSD Background Color Control Register **F4H** **Set 1, Bank 1**

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

.7 **Frame Background Color Enable Bit**

0	Disable frame background color (no background color is displayed)
1	Enable frame background color (turn frame background color on)

.6–4 **Frame Background Color Selection Bits (when .7 = "1")**

0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

.3 **Character Background Color Enable Bit**

0	Disable character background color (no background color is displayed)
1	Enable character background color (turn character background color on)

.2–0 **Character Background Color Selection Bits (when .3 = "1")**

0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

DSPCON — On-Screen Display Control Register

F5H

Set 1, Bank 1

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

.7–.4

OSD ROW counter

.3 **OSD Synchronization Clock Edge Selection Bit (H-Sync, V-Sync)**

0	Use rising clock edges for synchronization
1	Use falling clock edges for synchronization

.2–.1 **Halftone or Background Color Selection Bits ⁽¹⁾**

0	0	Character background color display
0	1	Not used
1	0	Halftone output
1	1	Character halftone and background color display

.0 **On-Screen Display Enable Bit ⁽²⁾**

0	Disable on-screen display (turn L-C oscillator off)
1	Enable on-screen display (turn L-C oscillator on)

NOTES:

- DSPCON.2–1 controls the bit-13 setting of each video RAM value (that is, for individual characters).
- If the OSD function is not being used, we recommend that bit 0 remain cleared to "0" in order to reduce the possibility of noise generated by the L-C oscillator.

EMT — External Memory Timing Register

FEH

Set 1, Bank 0

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

.7 External WAIT Input Function Enable Bit

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

.6 Slow Memory Timing Enable Bit

0	Disable slow memory timing
1	Enable slow memory timing

.5 and .4 Program Memory Automatic Wait Control Bits

0	0	No wait (normal operation)
0	1	Wait one cycle
1	0	Wait two cycles
1	1	Wait three cycles

.3 and .2 Data Memory Automatic Wait Control Bits

0	0	No wait (normal operation)
0	1	Wait one cycle
1	0	Wait two cycles
1	1	Wait three cycles

.1 Stack Area Selection Bit

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

.0 Not used for the S3C8847/C8849/P8849.

NOTE: Because an external interface is not implemented for the S3C8847/C8849/P8849, the EMT values should always be "0".

FADECON — OSD Fade Control Register

F1H

Set 1, Bank 1

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

.7

Not used for the S3C8847/C8849/P8849.

.6 **Fade Function Enable Bit**

0	Enable fade function
1	Disable fade function

.5 **Fade Direction Selection Bit**

0	Fade direction: Before the character matrix
1	Fade direction: After the character matrix

.4–.0 **Fade Line Selection Bits**

0	0	0	0	0	Line 0 selected
0	0	0	0	1	Line 1 selected
0	0	0	1	0	Line 2 selected
0	0	0	1	1	Line 3 selected
0	0	1	0	0	Line 4 selected
0	0	1	0	1	Line 5 selected
0	0	1	1	0	Line 6 selected
0	0	1	1	1	Line 7 selected
0	1	0	0	0	Line 8 selected
0	1	0	0	1	Line 9 selected
0	1	0	1	0	Line 10 selected
0	1	0	1	1	Line 11 selected
0	1	1	0	0	Line 12 selected
0	1	1	0	1	Line 13 selected
0	1	1	1	0	Line 14 selected
0	1	1	1	1	Line 15 selected
1	0	0	0	0	Line 16 selected
1	0	0	0	1	Line 17 selected
Others					Invalid selection

FLAGS — System Flags Register

D5H

Set 1

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

.7

Carry Flag (C)

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

.6

Zero Flag (Z)

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

.5

Sign Flag (S)

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

.4

Overflow Flag (V)

0	Operation result is $\leq +127$ or ≥ -128
1	Operation result is $> +127$ or < -128

.3

Decimal Adjust Flag (D)

0	Add operation has completed
1	Subtraction operation has completed

.2

Half-Carry Flag (H)

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

.1

Fast Interrupt Status Flag (FIS)

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

.0

Bank Address Selection Flag (BA)

0	Bank 0 is selected (using the SB0 instruction)
1	Bank 1 is selected (using the SB1 instruction)

HTCON — Halftone Signal Control Register

F6H

Set 1, Bank 1

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

.7 Halftone Output Polarity Selection Bit (HT Only)

0	Active high (normal halftone output is Low level)
1	Active low (normal halftone output is High level)

.6 RGB Output Polarity Selection Bit

0	Active high (normal RGB polarity is Low level)
1	Active low (normal RGB polarity is High level)

.5 OSD ROW Interrupt Enable Bit

0	Disable the OSD ROW interrupt
1	Enable the OSD ROW interrupt

.4 OSD ROW Interrupt Pending Bit

0	No interrupt pending (when read); clear pending bit (when write)
1	Interrupt is pending (when read); no effect (when write)

.3 Halftone Function Enable Bit

0	Disable the halftone control signal
1	Enable the halftone control signal

.2 Halftone Option Selection Bit

0	Halftone output for character periods only (as selected by video RAM bit-13)
1	Halftone output for all frame periods (regardless of video RAM bit-13 setting)

.1 V-Sync Interrupt Enable Bit

0	Disable the V-sync interrupt
1	Enable the V-sync interrupt

.0 V-Sync Interrupt Pending Bit

0	No OSD ROW interrupt pending bit (when read)
0	Clear pending bit (when write)
1	OSD ROW interrupt is pending (when read)
1	No effect (when write)

IMR — Interrupt Mask Register

DDH

Set 1

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

.7 **Interrupt Priority Level 7 (IRQ7) Enable Bit; V-Sync**

0	Disable IRQ7 interrupt
1	Enable IRQ7 interrupt

.6 **Interrupt Priority Level 6 (IRQ4) Enable Bit; Timer A**

0	Disable IRQ6 interrupt
1	Enable IRQ6 interrupt

.5 Not used for S3C8847/C8849/P8849.**.4** **Interrupt Priority Level 4 (IRQ4) Enable Bit; P1.2 and P1.3 External Interrupt**

0	Disable IRQ4 interrupt
1	Enable IRQ4 interrupt

.3 **Interrupt Priority Level 3 (IRQ3) Enable Bit; CAPA**

0	Disable IRQ3 interrupt
1	Enable IRQ3 interrupt

.2 **Interrupt Priority Level 2 (IRQ2) Enable Bit; OSD ROW Interrupt**

0	Disable IRQ2 interrupt
1	Enable IRQ2 interrupt

.1 **Interrupt Priority Level 1 (IRQ1) Enable Bit; P1.0 and P1.1 External Interrupt**

0	Disable IRQ1 interrupt
1	Enable IRQ1 interrupt

.0 **Interrupt Priority Level 0 (IRQ0) Enable Bit; T0INT (Match)**

0	Disable IRQ0 interrupt
1	Enable IRQ0 interrupt

IPH — Instruction Pointer (High Byte)

DAH

Set 1

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

.7–.0

Instruction Pointer Address (High Byte)

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

IPL — Instruction Pointer (Low Byte)

DBH

Set 1

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

.7 – .0

Instruction Pointer Address (Low Byte)

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

IPR — Interrupt Priority Register

FFH

Set 1, Bank 0

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

.7, .4, and .1

Priority Control Bits for Interrupt Groups A, B, and C ⁽¹⁾

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

.6

Interrupt Group C Priority Control Bit

0	IRQ6 > IRQ7
1	IRQ7 > IRQ6

.5

Not used for the S3C8847/C8849/P8849.	
---------------------------------------	--

.3

Interrupt Sub Group B Priority Control Bit

0	IRQ3 > IRQ4
1	IRQ4 > IRQ3

.2

Interrupt Group B Priority Control Bit

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

.0

Interrupt Group A Priority Control Bit

0	IRQ0 > IRQ1
1	IRQ1 > IRQ0

NOTES:

1. Interrupt group A is IRQ0 and IRQ1; interrupt group B is IRQ2, IRQ3, and IRQ4; interrupt group C is IRQ6 and IRQ7.
2. Interrupt level IRQ5 is not used in the S3C8847/C8849/P8849 interrupt structure. For this reason, IPR.5 is not used.

IRQ — Interrupt Request Register

DCH

Set 1

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

.7 Interrupt Level 7 (IRQ7) Request Pending Bit; V-Sync

0	No IRQ7 interrupt pending
1	IRQ7 interrupt is pending

.6 Interrupt Level 6 (IRQ6) Request Pending Bit; Timer A

0	No IRQ6 interrupt pending
1	IRQ6 interrupt is pending

.5 Not used for the S3C8847/C8849/P8849.**.4 Interrupt Level 4 (IRQ4) Request Pending Bit; P1.2 and P1.3 External Interrupt**

0	No IRQ4 interrupt pending
1	IRQ4 interrupt is pending

.3 Interrupt Level 3 (IRQ3) Request Pending Bit; CAPA

0	No IRQ3 interrupt pending
1	IRQ3 interrupt is pending

.2 Interrupt Level 2 (IRQ2) Request Pending Bit; OSD ROW Interrupt

0	No IRQ2 interrupt pending
1	IRQ2 interrupt is pending

.1 Interrupt Level 1 (IRQ1) Request Pending Bit; P1.0 and P1.1 External Interrupt

0	No IRQ1 interrupt pending
1	IRQ1 interrupt is pending

.0 Interrupt Level 0 (IRQ0) Request Pending Bit; T0INT (Match)

0	No IRQ0 interrupt pending
1	IRQ0 interrupt is pending

NOTE: Interrupt level request pending bits can be polled by software to detect an interrupt request pending condition on any of the seven valid interrupt levels (IRQ0–IRQ3, IRQ6, and IRQ7). Interrupt pending bits are read-only addressable.

P0CONH — Port 0 Control Register (High Byte)

E4H

Set 1, Bank 0

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

.7 and .6**Port 0.7 Configuration Bits**

0	0	Input mode
0	1	ADC Input mode
1	0	Open-drain output mode
1	1	Open-drain output mode

.5 and .4**Port 0.6 Configuration Bits**

0	0	Input mode
0	1	ADC Input mode
1	0	Open-drain output mode
1	1	Open-drain output mode

.3 and .2**Port 0.5 Configuration Bits**

0	0	Input mode
0	1	Input mode
1	0	Push-pull output mode
1	1	Push-pull output mode

.1 and .0**Port 0.4 Configuration Bits**

0	0	Input mode
0	1	Input mode
1	0	Push-pull output mode
1	1	Push-pull output mode

P0CONL — Port 0 Control Register (Low Byte)

E5H

Set 1, Bank 0

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

.7 and .6**Port 0.3 Configuration Bits**

0	0	Input mode
0	1	Input mode
1	0	N-channel open-drain output mode (5 V load)
1	1	Push-pull output mode

.5 and .4**Port 0.2 Configuration Bits**

0	0	Input mode
0	1	Input mode
1	0	N-channel open-drain output mode (5 V load)
1	1	Push-pull output mode

.3 and .2**Port 0.1 Configuration Bits**

0	0	Input mode
0	1	Input mode
1	0	N-channel open-drain output mode (5 V load)
1	1	Push-pull output mode

.1 and .0**Port 0.0 Configuration Bits**

0	0	Input mode
0	1	Input mode
1	0	N-channel open-drain output mode (5 V load)
1	1	Push-pull output mode

P1CONH — Port 1 Control Register (High Byte)

E6H

Set 1, Bank 0

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

.7 and .6**Port 1.7/T0CK Configuration Bits**

0	0	Input mode
0	1	Timer 0 clock Input mode
1	0	Push-pull output mode
1	1	Push-pull output mode

.5 and .4**Port 1.6 Configuration Bits**

0	0	Input mode
0	1	Input mode
1	0	Push-pull output mode
1	1	Push-pull output mode

.3 and .2**Port 1.5 Configuration Bits**

0	0	Input mode
0	1	Input mode
1	0	N-channel open-drain mode (6-volt load capacity)
1	1	N-channel open-drain mode (6-volt load capacity)

.1 and .0**Port 1.4 Configuration Bits**

0	0	Input mode
0	1	Input mode
1	0	N-channel open-drain mode (6-volt load capacity)
1	1	N-channel open-drain mode (6-volt load capacity)

P1CONL — Port 1 Control Register (Low Byte)

E7H

Set 1, Bank 0

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

.7 and .6**Port 1.3 Configuration Bits**

0	0	Input mode; interrupt disabled
0	1	Input mode; interrupt on rising edge
1	0	Input mode; interrupt on falling edge
1	1	N-channel open-drain output mode (6-volt load capacity)

.5 and .4**Port 1.2 Configuration Bits**

0	0	Input mode; interrupt disabled
0	1	Input mode; interrupt on rising edge
1	0	Input mode; interrupt on falling edge
1	1	N-channel open-drain output mode (6-volt load capacity)

.3 and .2**Port 1.1/INT1 Configuration Bits**

0	0	Input mode; interrupt disabled
0	1	Input mode; interrupt on rising edge
1	0	Input mode; interrupt on falling edge
1	1	N-channel open-drain output mode (6-volt load capacity)

.1 and .0**Port 1.0/INT0 Configuration Bits**

0	0	Input mode; interrupt disabled
0	1	Input mode; interrupt on rising edge
1	0	Input mode; interrupt on falling edge
1	1	N-channel open-drain output mode (6-volt load capacity)

P2CONH — Port 2 Control Register (High Byte)

E8H

Set 1, Bank 0

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

.7 and .6

Port 2.7/OSDHT Configuration Bits

0	0	Input mode
0	1	N-channel open-drain output mode (5-volt load capacity)
1	0	Push-pull output mode
1	1	OSD half-tone output mode (push-pull circuit type)

.5 and .4

Port 2.6/T0 Configuration Bits

0	0	Input mode
0	1	N-channel open-drain output mode (5-volt load capacity)
1	0	Push-pull output mode
1	1	Timer 0 output mode (interval or PWM; N-channel open-drain type)

.3 and .2

Port 2.5/PWM0 Configuration Bits

0	0	Input mode
0	1	N-channel open-drain output mode (5-volt load capacity)
1	0	Push-pull output mode
1	1	PWM0 output mode (push-pull circuit type)

.1 and .0

Port 2.4/PWM4 Configuration Bits

0	0	Input mode
0	1	N-channel open-drain output mode (5-volt load capacity)
1	0	Push-pull output mode
1	1	PWM4 output mode (N-channel open-drain type)



P2CONL — Port 2 Control Register (Low Byte)

E9H

Set 1, Bank 0

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

.7 and .6**Port 2.3/PWM3 Configuration Bits**

0	0	Normal input mode
0	1	Normal input mode
1	0	PWM3 output mode; N-channel, open-drain output mode With 5-volt load capacity
1	1	Push-pull output mode

.5 and .4**Port 2.2/PWM2 Configuration Bits**

0	0	Normal input mode
0	1	Normal input mode
1	0	PWM2 output mode; N-channel, open-drain output mode With 5-volt load capacity
1	1	Push-pull output mode

.3 and .2**Port 2.1/PWM1 Configuration Bits**

0	0	Normal input mode
0	1	Normal input mode
1	0	PWM1 output mode; N-channel, open-drain output mode With 5-volt load capacity
1	1	Push-pull output mode

.1 and .0**Port 2.0/PWM5 Configuration Bits**

0	0	Normal input mode
0	1	Normal input mode
1	0	PWM5 output mode; N-channel, open-drain output mode With 5-volt load capacity
1	1	Push-pull output mode

P3CONL — Port 3 Control Register (Low Byte)

EBH

Set 1, Bank 0

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

.7 – .4	No effect
----------------	-----------

.3 and .2

Port 3.1/ADC1 Configuration Bits

0	0	Input mode
0	1	ADC input mode
1	0	Input mode
1	1	N-channel, open-drain output mode with 5-volt load capacity

.1 and .0

Port 3.0/ADC0 Configuration Bits

0	0	Input mode
0	1	ADC input mode
1	0	Input mode
1	1	N-channel, open-drain output mode with 5-volt load capacity

PP — Register Page Pointer**DFH****Set 1**

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

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

0	0	0	0	Destination: page 0
0	0	0	1	Destination: page 1
0	0	1	0	Not used for the S3C8847/C8849/P8849.
• • •				"
1	1	1	1	Not used for the S3C8847/C8849/P8849.

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

0	0	0	0	Source: page 0
0	0	0	1	Source: page 1
0	0	1	0	Not used for the S3C8847/C8849/P8849.
• • •				"
1	1	1	1	Not used for the S3C8847/C8849/P8849.

NOTE: When PP = 11H or 10H and working register area is not in C0H–CFH, DJNZ instruction do not operate.

PWMCON — PWM Control Register

F8H

Set 1, Bank 0

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

.4, .7, and .6

3-Bit Prescaler Value for PWM Counter Input Clock

0	0	0	Non-divided input clock
0	0	1	Divided-by-two input clock
0	1	0	Divided-by-three input clock
0	1	1	Divided-by-four input clock
1	0	0	Divided-by-five input clock
1	0	1	Divided-by-six input clock
1	1	0	Divided-by-seven input clock
1	1	1	Divided-by-eight input clock

.5

PWM Counter Enable Bit

0	Stop PWM counter operation
1	Start (or resume) PWM counter operation

.3

Capture A Interrupt Enable Bit

0	Disable capture A interrupt
1	Enable capture A interrupt

.2

Not used for the S3C8847/C8849/P8849.

.1 and .0

Capture A Module Control Bits

0	0	Disable capture A module
0	1	Capture on falling edges only
1	0	Capture on rising edges only
1	1	Capture on both rising and falling edges



ROWCON — OSD Row Position Control Register

F2H

Set 1, Bank 1

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

.7–.3**Top Margin Display Position Control Value (4 x TMG value of 0–31 dots)**

0	0	0	0	0	Top margin position = 0H
0	0	0	0	1	Top margin position = 4H
...					...
1	1	1	1	1	Top margin position = 124H

.2–.0**Inter-Row Spacing Control Value (0–7H)**

0	0	0	No inter-row spacing
0	0	1	Inter-row spacing = 1H
...			...
1	1	1	Inter-row spacing = 7H

NOTE: To set top margin and inter-row spacing, separate decimal values must be calculated, converted to their binary equivalents, and then written to the ROWCON register.

RP0 — Register Pointer 0

D6H

Set 1

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

.7–.3

Register Pointer 0 Address Value

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

.2–.0

Not used for the S3C8847/C8849/P8849.

RP1 — Register Pointer 1

D7H

Set 1

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

.7–.3

Register Pointer 1 Address Value

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

.2–.0

Not used for the S3C8847/C8849/P8849.

SPH — Stack Pointer (High Byte)**D8H****Set 1**

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

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

The high-byte stack pointer value is the upper 8 bits of the 16-bit stack pointer address (SP15–SP8). The lower byte of the stack pointer value is located in the register SPL (D9H). The SP value is undefined after a reset.

SPL — Stack Pointer (Low Byte)**D9H****Set 1**

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

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

The low-byte stack pointer value is the lower 8 bits of the 16-bit stack pointer address (SP7–SP0). The upper byte of the stack pointer value is located in the register SPH (D8H). The SP value is undefined after a reset.

SYM — System Mode Register

DEH

Set 1

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

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

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

.6–.5 Not used for the S3C8847/C8849/P8849.

.4–.2 Fast Interrupt Level Selection Bits

0	0	0	Level 0 (IRQ0)
0	0	1	Level 1 (IRQ1)
0	1	0	Level 2 (IRQ2)
0	1	1	Level 3 (IRQ3)
1	0	0	Level 4 (IRQ4)
1	0	1	Not used for S3C8847/C8849/P8849.
1	1	0	Level 6 (IRQ6)
1	1	1	Level 7 (IRQ7)

.1 Fast Interrupt Enable Bit

0	Disable fast interrupt processing
1	Enable fast interrupt processing

.0 Global Interrupt Enable Bit ⁽²⁾

0	Disable global interrupt processing
1	Enable global interrupt processing

NOTES:

1. Because the S3C8847/C8849/P8849 microcontrollers do not have an external interface, bit 7 should always be "0".
2. After a reset, the initialization routine must enable global interrupt processing by executing an EI instruction (and not by writing a "1" to SYM.0).

TACON — Timer A Control Register

F2H

Set 1, Bank 0

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

.7–.4

4-Bit Prescaler for Timer A Clock Input				
0	0	0	0	Divide input by 1 (non-divided)
0	0	0	1	Divide input by 2
...			...	
1	1	1	1	Divide input by 16

.3

Timer A Clock Source Selection Bit

0	CPU clock divided by 1000
1	Non-divided CPU clock

.2

Timer A Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.1

Timer A Interrupt Pending Bit

0	No interrupt pending (when read)
0	<i>Clear pending bit (when write)</i>
1	Interrupt is pending (when read)
1	<i>No effect (when write)</i>

.0

Timer A Operating Mode Selection Bit

Not used for the S3C8847/C8849/P8849.	
---------------------------------------	--

T0CON — Timer 0 Control Register

D2H

Set 1

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

.7 and .6

T0 Input Clock Selection Bits

0	0	$f_{OSC}/4096$
0	1	$f_{OSC}/256$
1	0	$f_{OSC}/8$
1	1	External clock (T0CLK)

.5 and .4

T0 Operating Mode Selection Bits

0	0	Interval mode
0	1	PWM mode
1	0	PWM mode
1	1	PWM mode

.3

T0 Counter Clear Bit

0	No effect
1	Clear the T0 counter (when write)

.2

No effect

.1

T0 Interrupt Enable Bit

0	Disable T0 interrupt
1	Enable T0 interrupt

.0

T0 Interrupt Pending Bit

0	No timer 0 interrupt pending
0	Clear timer 0 pending bit (when write)
1	Timer 0 interrupt is pending

VSBCON — V-sunc Blank Control Register

F7H

Set 1, Bank1

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

.7 and .5

Not used for the S3C8847/C8849/P8849.

.4–.0

V-sync Blank Time Control Bits:

0	0	0	0	0	9 Horizontal Sync
...					"
0	1	0	0	1	9 Horizontal Sync
0	1	0	1	0	10 Horizontal Sync
0	1	0	1	1	11 Horizontal Sync
...					...
1	1	1	1	1	31 Horizontal Sync

5 INTERRUPT STRUCTURE

OVERVIEW

The SAM87 interrupt structure has three basic components: levels, vectors, and sources. The CPU recognizes 8 interrupt levels and supports up to 128 interrupt vectors. When a specific interrupt level has more than one vector address, the vector priorities are established in hardware. Each vector can have one or more interrupt sources.

Levels

Levels provide the highest-level method of interrupt priority assignment and recognition. All peripherals and I/O blocks can issue interrupt requests. In other words, peripheral and I/O operations are interrupt-driven. There are eight interrupt levels: IRQ0–IRQ7. Each interrupt level directly corresponds to an interrupt request number (IRQn). The total number of interrupt levels used in the interrupt structure varies from device to device. For the S3C8847 and the S3C8849/P8849 microcontrollers, seven levels are recognized: IRQ0–IRQ4, IRQ6, and IRQ7.

The interrupt level numbers 0 through 7 do not necessarily indicate the relative priority of the levels. They are simply identifiers for the interrupt levels that are recognized by the CPU (IRQ0–IRQ7). The relative priority of different interrupt levels is determined by settings in the interrupt priority register, IPR. Interrupt logic controlled by the IPR settings lets you define additional priority relationship for specific interrupt levels.

Vectors

Each interrupt level can have one or more interrupt vectors, or it may have no vector address assigned at all. The maximum number of vectors that can be supported for a given level is 128. (The actual number of vectors used for the S3C8-series microcontrollers is always much smaller.) If an interrupt level has more than one vector address, the vector priorities are set in hardware. The S3C8847 and the S3C8849/P8849 have 9 vectors, one corresponding to each of the 9 possible sources.

Sources

A source is any peripheral that generates an interrupt. A source can be an external pin or a counter overflow, for example. Each vector can have several interrupt sources. In the S3C8847/C8849/P8849 interrupt structure, each source has its own vector address. When a service routine starts, the respective pending bit is either cleared automatically by hardware or "manually" by the program software. The characteristics of the source's pending mechanism determine which method is used to clear its pending bit.

INTERRUPT TYPES

The three components of the SAM87 interrupt structure described above — levels, vectors, and sources — are combined to determine the interrupt structure of an individual device and to make full use of its available interrupt logic. There are three possible combinations of interrupt structure components, called interrupt types 1, 2, and 3. The types differ in the number of vectors and interrupt sources assigned to each level (see Figure 5-1):

- Type 1: One level (IRQn) + one vector (V_1) + one source (S_1)
- Type 2: One level (IRQn) + one vector (V_1) + multiple sources ($S_1 - S_n$)
- Type 3: One level (IRQn) + multiple vectors ($V_1 - V_n$) + multiple sources ($S_1 - S_n, S_{n+1} - S_{n+m}$)

In the S3C8847 and the S3C8849/P8849 interrupt structure, only interrupt types 1 and 3 are implemented.

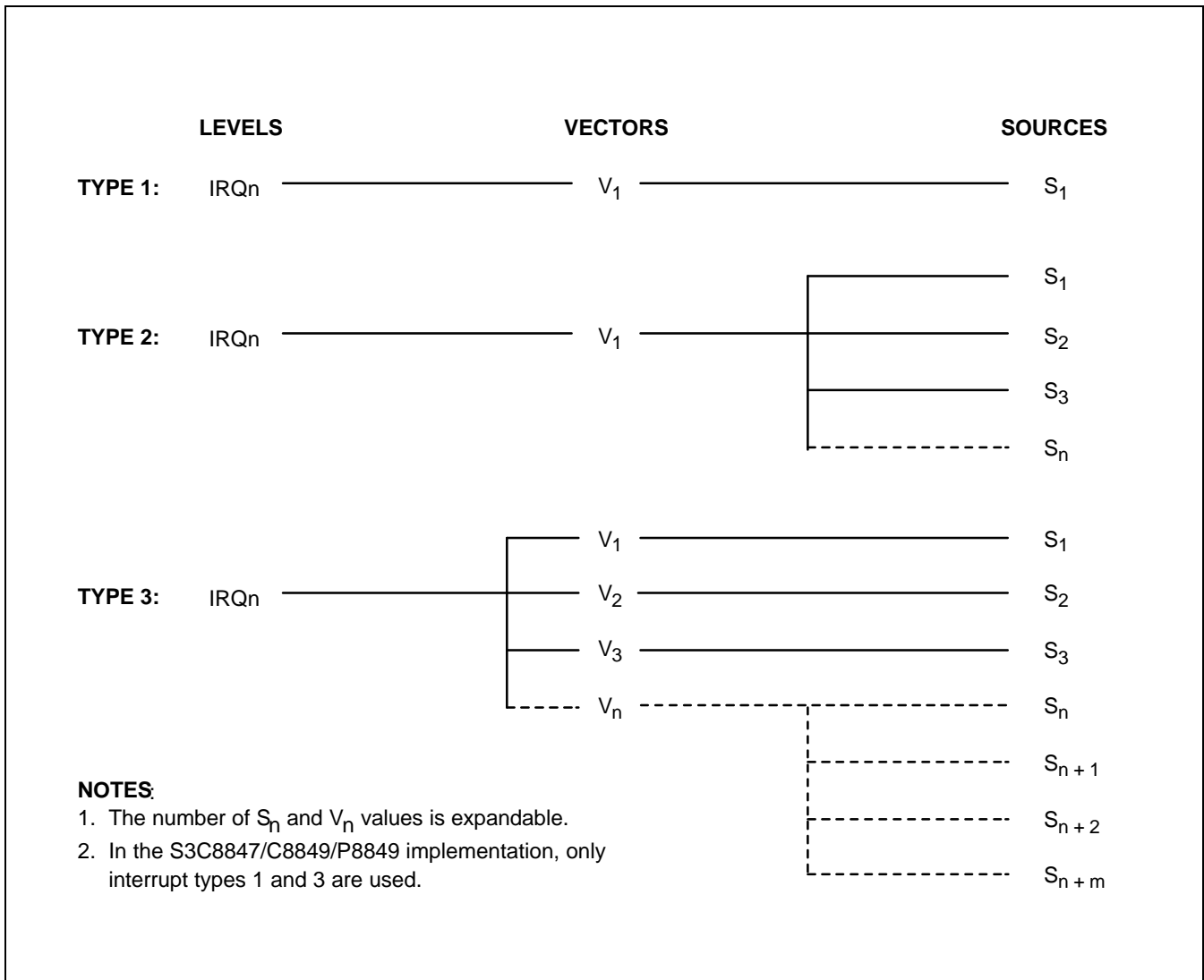


Figure 5-1. S3C8-Series Interrupt Types

S3C8847/C8849/P8849 INTERRUPT STRUCTURE

The S3C8847 and the S3C8849 microcontrollers have 9 standard interrupt sources. Nine different vector addresses are used to support these interrupt sources. Seven of the eight available levels are used for the interrupt structure: IRQ0–IRQ3, IRQ6, and IRQ7. The device-specific interrupt structure is shown in Figure 5-2.

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

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

LEVEL	VECTOR	SOURCE	IDENTIFIER	RESET
IRQ0	FCH	Timer 0 interrupt (match)	T0INT	S/W
IRQ1	C0H <u>0</u>	P1.0 external interrupt	P10INT	H/W
	C2H <u>1</u>	P1.1 external interrupt	P11INT	H/W
IRQ2	C4H	OSD ROW interrupt	ROWINT	S/W
IRQ3	02H	Capture A (8-bit)	CAPA	H/W
IRQ4	C6H <u>0</u>	P1.2 external interrupt	P12INT	H/W
	C8H <u>1</u>	P1.3 external interrupt	P13INT	H/W
IRQ6	BEH	Timer A	TAINT	S/W
IRQ7	D4H	V-sync	VSYNC	S/W

NOTES:

- The interrupt level IRQ5 is not used in the S3C8847/C8849/P8849 interrupt structure.
- For interrupt levels with two or more vectors, the lowest vector address usually has the highest priority. For example, C0H has higher priority (0) than C2H (1) within the level IRQ1. These priorities (see numbers) are hardwired.
- 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 5-2. S3C8847/C8849/P8849 Interrupt Structure

INTERRUPT VECTOR ADDRESSES

Interrupt vector addresses for the S3C8847/C8849/P8849 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.

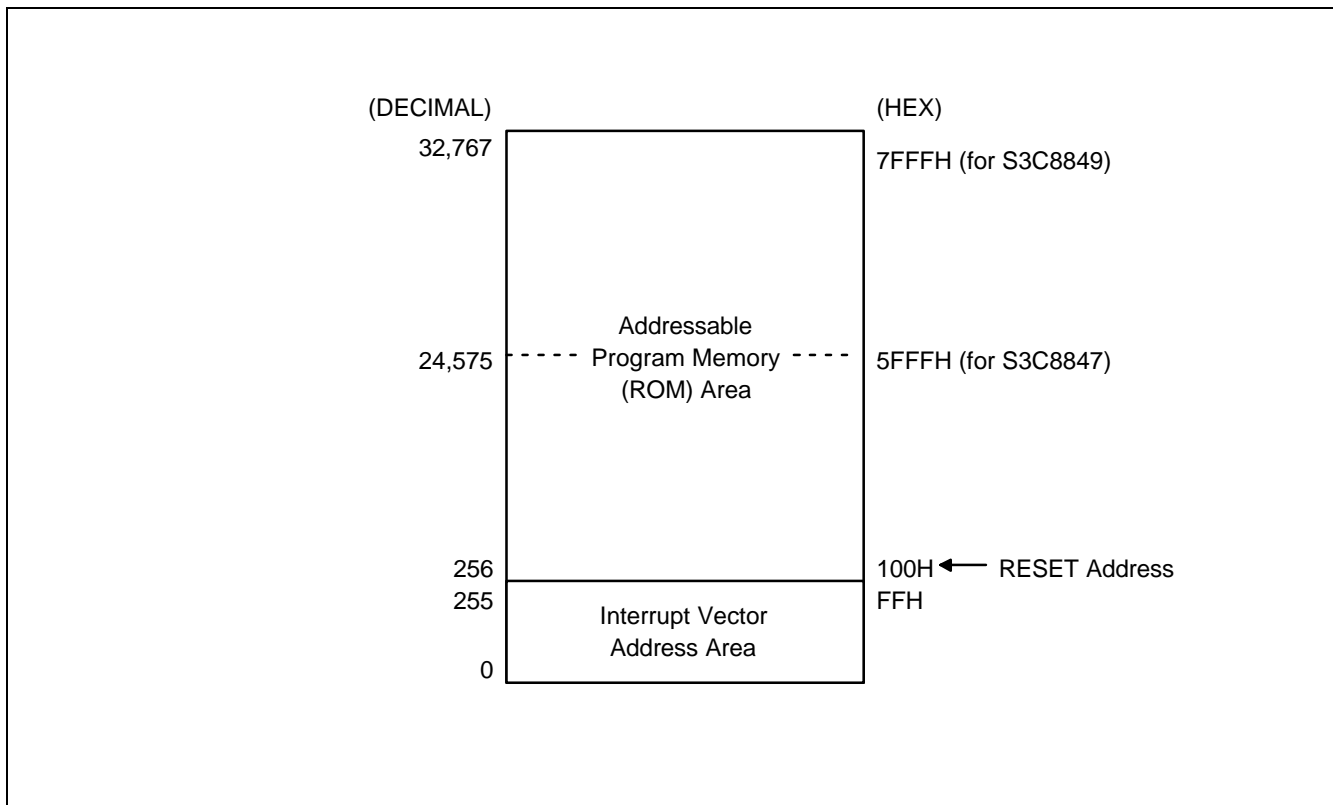


Figure 5-3. ROM Vector Address Area

Table 5-1. S3C8847/C8849/P8849 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)	IRQ0	–		√
212	D4H	V-sync	IRQ7	–		√
200	C8H	P1.3 external interrupt	IRQ4	1	√	
198	C6H	P1.2 external interrupt		0	√	
196	C4H	OSD ROW interrupt	IRQ2	–		√
194	C2H	P1.1 external interrupt	IRQ1	1	√	
192	C0H	P1.0 external interrupt		0	√	
190	BEH	Timer A	IRQ6	–		√
2	02H	Capture A (8-bit)	IRQ3	–	√	

NOTES:

1. Interrupt priorities are identified in inverse order: '0' is the 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 the interrupt level IRQ1, the higher-priority interrupt vector is the P1.0 external interrupt, vector C0H; the lower-priority interrupt within that level is the P1.1 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 the 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 external interface, if implemented.

Table 5-2. 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 S3C8847/C8849/P8849, the seven levels are organized into three groups: A, B, and C. Group A includes IRQ0 and IRQ1, group B is IRQ2, IRQ3, and 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).

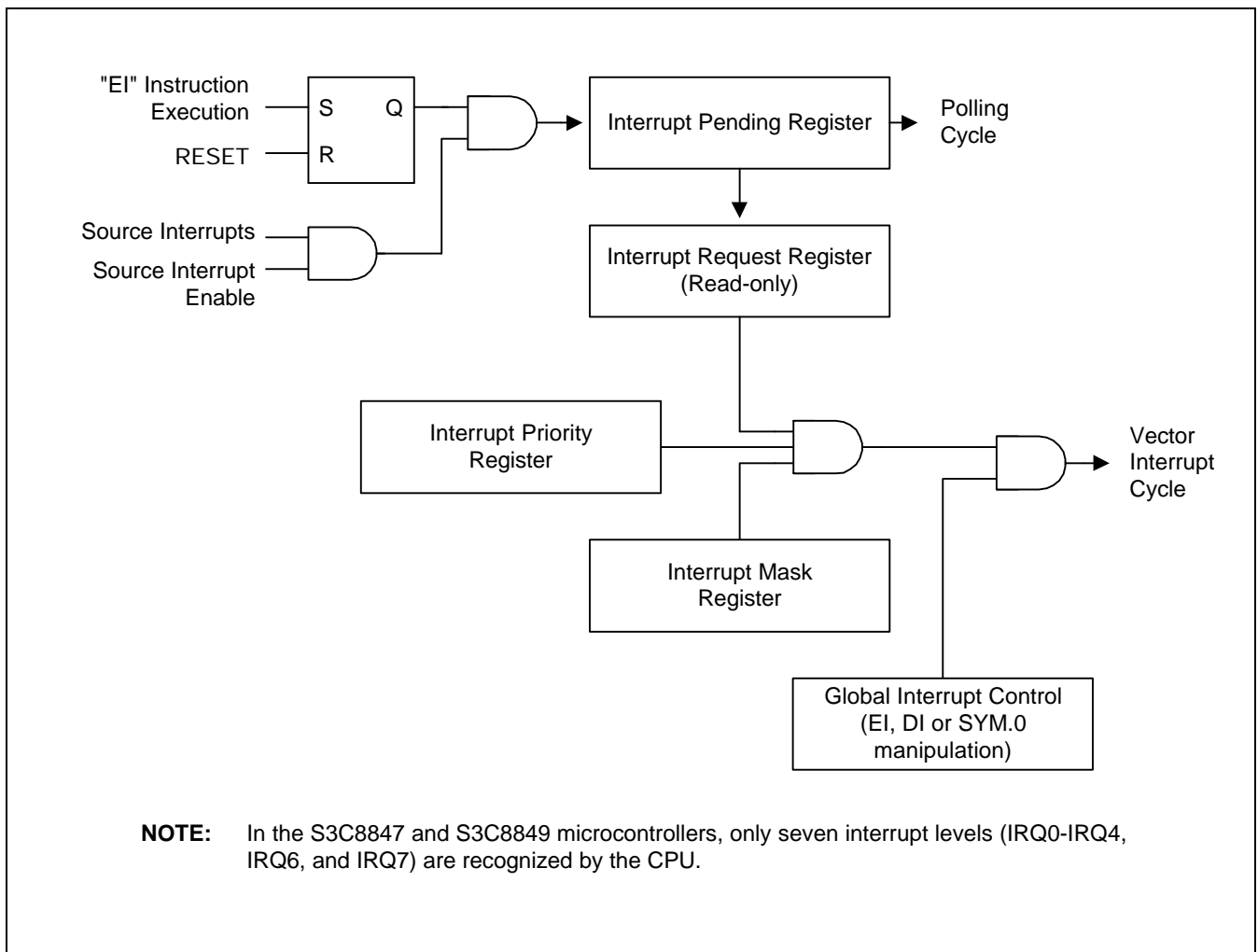


Figure 5-4. Interrupt Function Diagram

PERIPHERAL INTERRUPT CONTROL REGISTERS

For each interrupt source there is a corresponding peripheral control register (or registers) that controls the interrupts generated by the peripheral. These registers and their locations are listed in Table 5-3.

Table 5-3. Interrupt Source Control Registers

Interrupt Source	Interrupt Level	Control Register	Register Location
Timer 0 (match)	IRQ0	T0CON	Set 1, D2H
P1.0 external interrupt P1.1 external interrupt	IRQ1	P1CONL	Set 1, bank 0, E7H
OSD ROW interrupt	IRQ2	HTCON	Set 1, bank 1, E6H
Capture A (8-bit)	IRQ3	PWMCON	Set 1, bank 0, F8H
P1.2 external interrupt P1.3 external interrupt	IRQ4	P1CONL	Set 1, bank 0, E7H
Timer A	IRQ6	TACON	Set 1, bank 0, F2H
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 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 S3C8847/C8849/P8849). A reset clears SYM.0, SYM.1, and SYM.7 to "0"; 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 the normal operation, we recommend using the EI and DI instructions for this purpose.

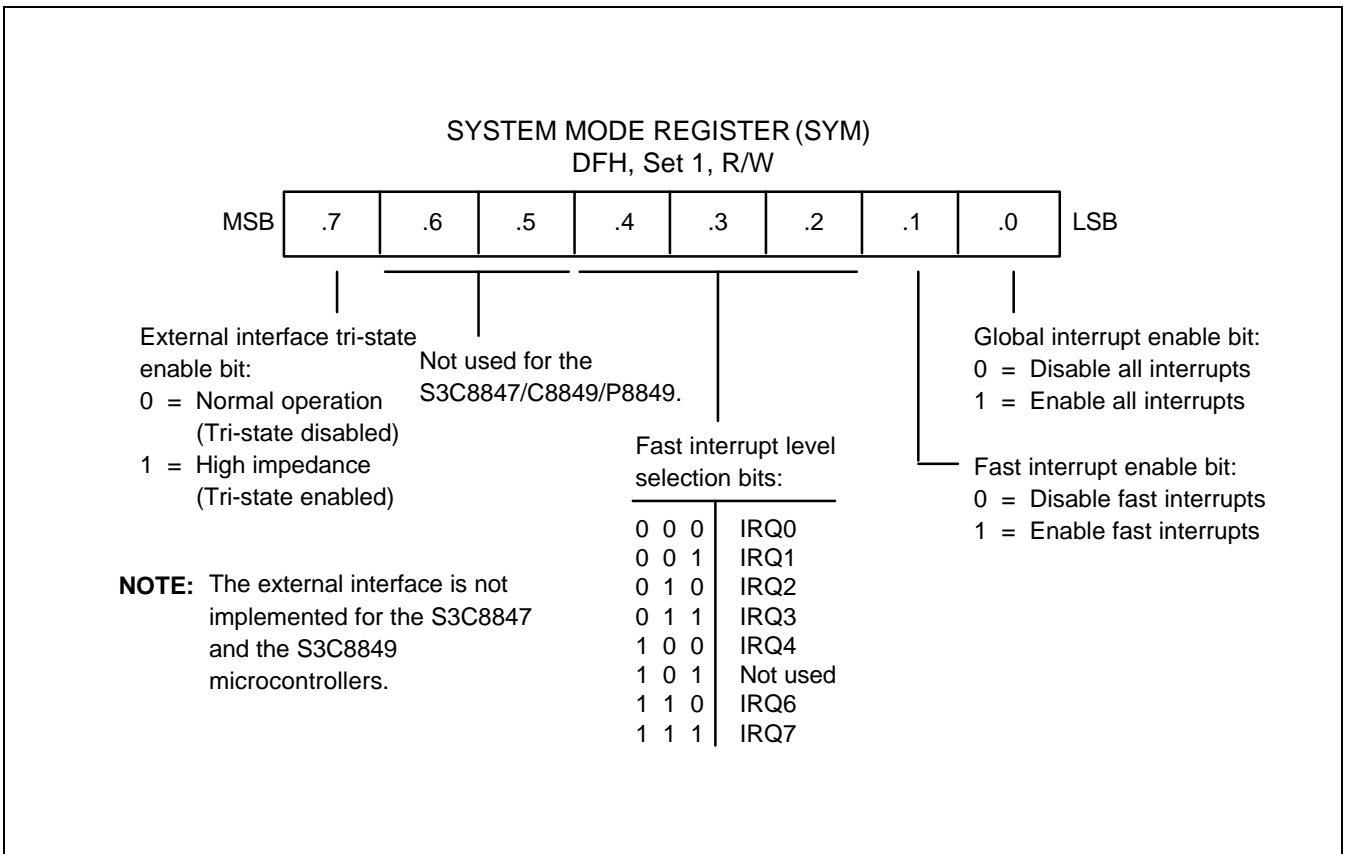


Figure 5-5. System Mode Register (SYM)

INTERRUPT MASK REGISTER (IMR)

The interrupt mask register (IMR) is used to enable or disable interrupt processing for each of the seven interrupt levels used in the S3C8847/C8849/P8849 interrupt structure, IRQ0–IRQ4, IRQ6, and IRQ7. After a reset, all 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 the register location DDH in set 1. Bit values can be read and written by instructions using Register addressing mode.

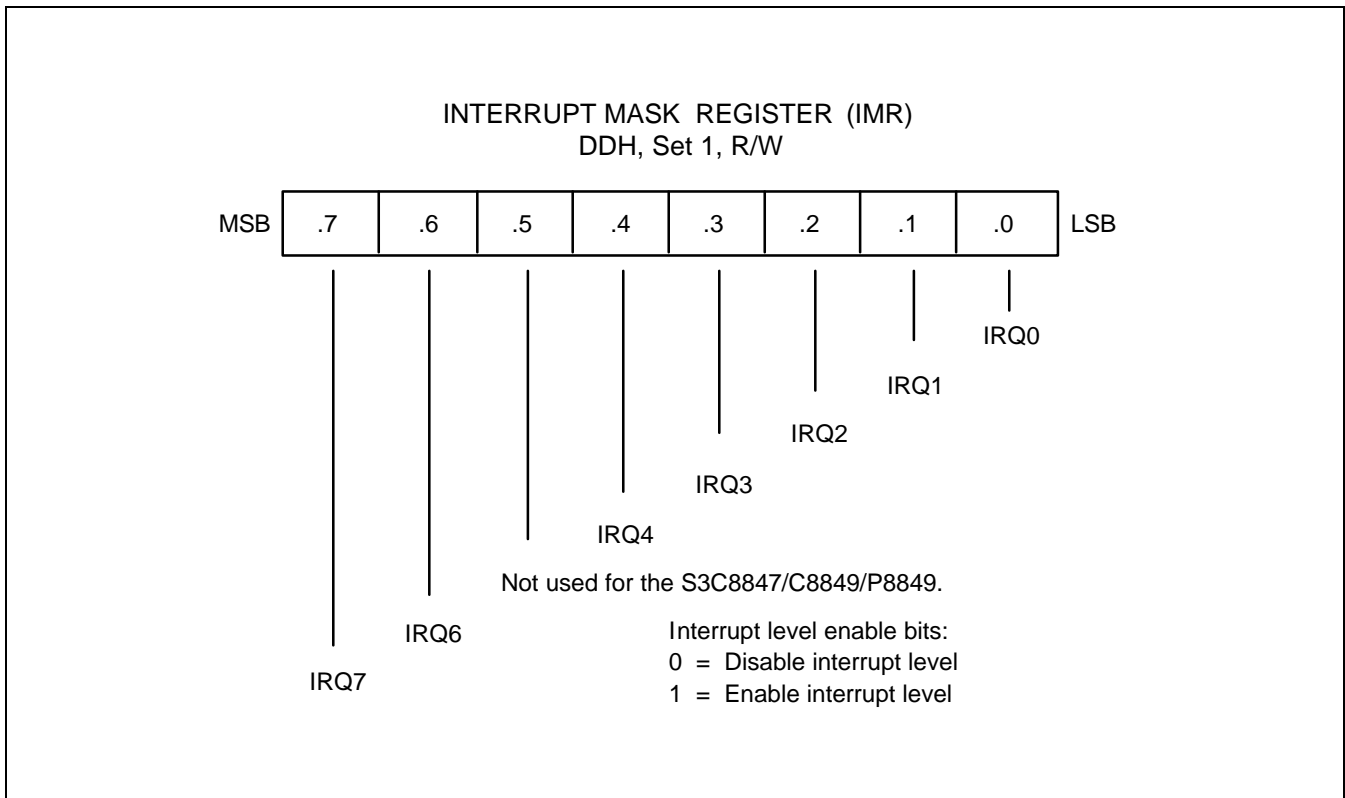


Figure 5-6. Interrupt Mask Register (IMR)

INTERRUPT PRIORITY REGISTER (IPR)

The interrupt priority register, IPR, is used to set the relative priorities of the seven interrupt levels used in the S3C8847/C8849/P8849 interrupt structure. The IPR register is mapped to the 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 5-7). These groups and subgroups are used only for IPR register priority definitions:

- Group A IRQ0, IRQ1
- Group B IRQ2, IRQ3, and 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 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 among interrupt levels 2, 3, and 4. IPR.3 defines possible subgroup B relationship.
- IPR.6 controls the relative priorities of group C interrupts.

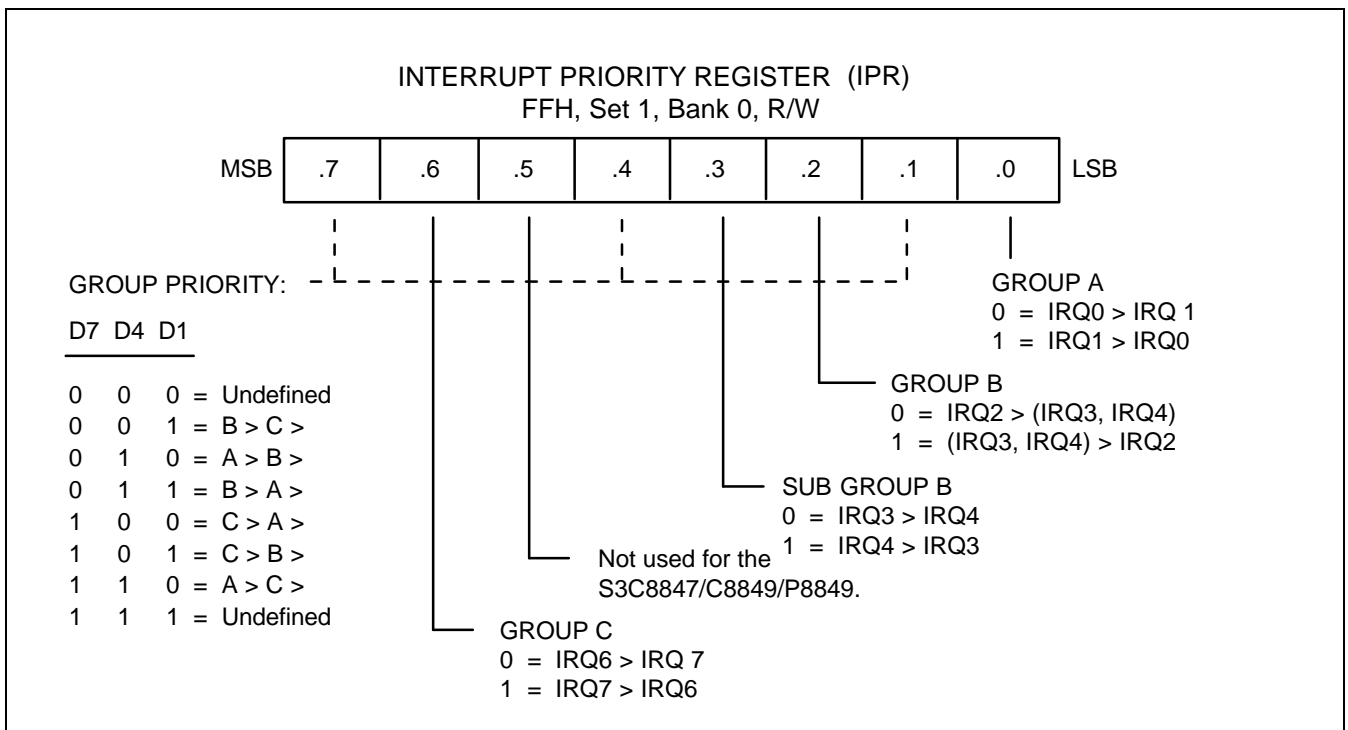


Figure 5-7. Interrupt Priority Register (IPR)

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 S3C8847/C8849/P8849 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 the register location DCH in set 1. IRQ bit values are read-only addressable using Register addressing mode. You can read (test) the contents of the IRQ register at any time using bit or byte addressing to determine the current interrupt request status of specific interrupt levels. After a reset, the IRQ register is cleared to 00H.

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

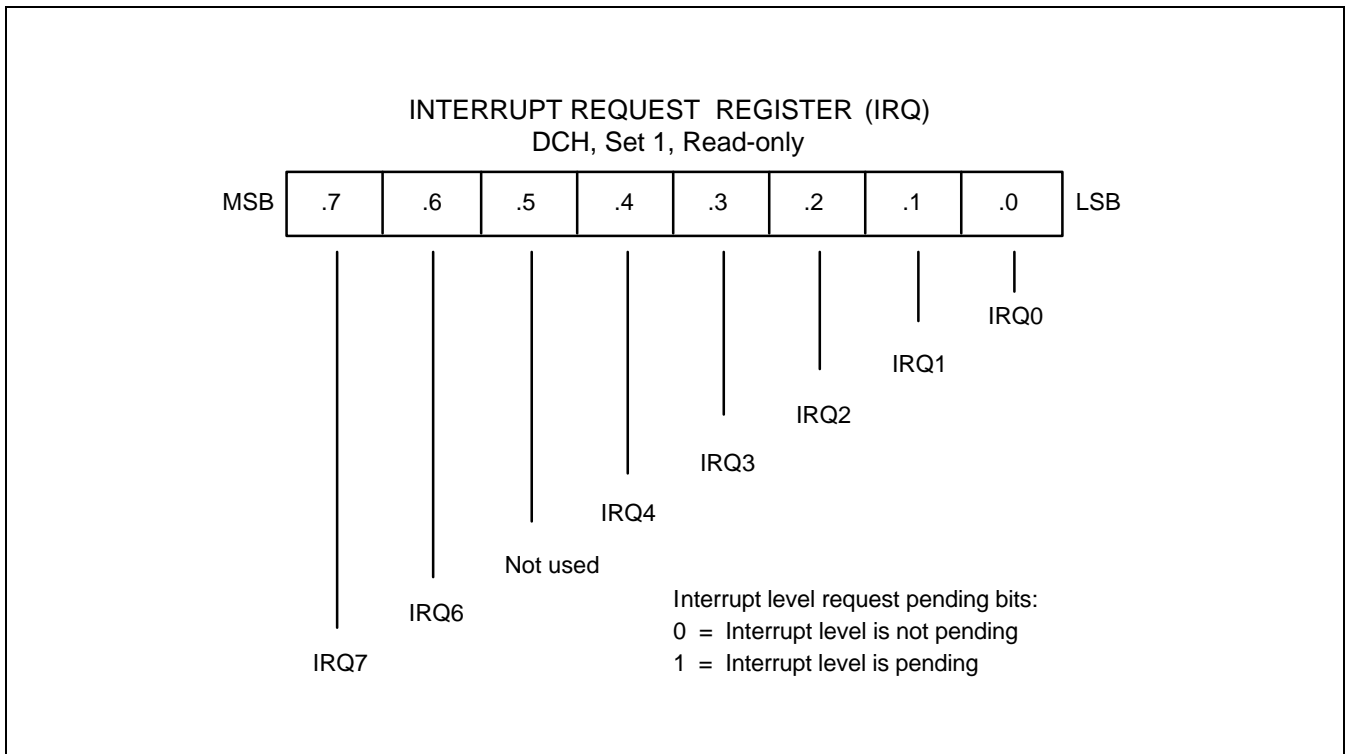


Figure 5-8. Interrupt Request Register (IRQ)

INTERRUPT PENDING FUNCTION TYPES

Overview

There are two types of interrupt pending bits: one is the type that automatically cleared by hardware after the interrupt service routine is acknowledged and executed; the other is the one that 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 S3C8847/C8849/P8849 interrupt structure, the P1.0, P1.1, P1.2 and P1.3 external interrupts, 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 interrupt, the timer A interrupt, the OSD row interrupt 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 is 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 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.
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, return the IMR to its original value by restoring 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 the S3C8-series microcontrollers to control optional high-speed interrupt processing called *fast interrupts*. The IP consists of the 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 S3C8847 and the S3C8849 microcontrollers, 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 5–17 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          ; Enable T0 interrupt
                                   ; Capture mode; trigger on rising signal edges
                                   ; Select fOSC/256 as T0 clock source
      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

```

7

CLOCK CIRCUITS

OVERVIEW

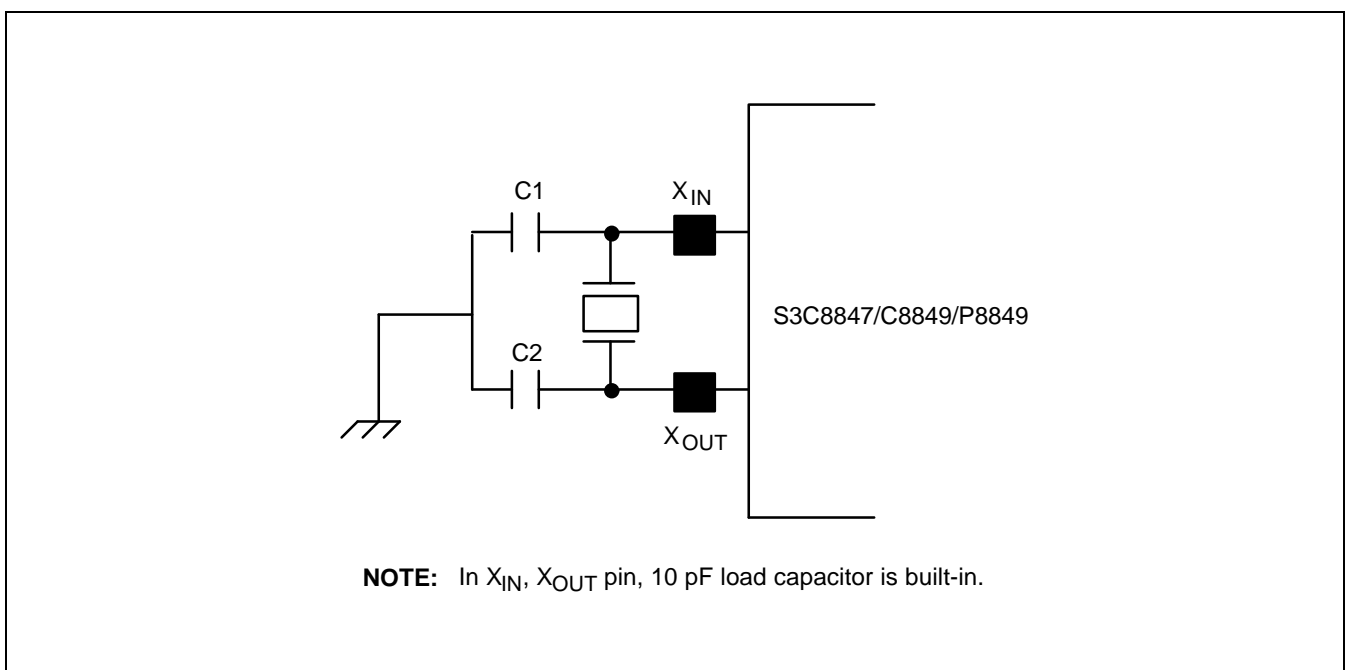
The clock frequency generated by an external crystal or ceramic resonator may range from 0.5 MHz to 8 MHz. The maximum CPU clock frequency is 8 MHz. The X_{IN} and X_{OUT} pins connect the external oscillation source to the on-chip clock circuit.

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

CLOCK STATUS DURING POWER-DOWN MODES

The two power-down modes, Stop mode and Idle mode, affect 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, Timer A counter, PWM, and capture (CAPA), which are inactive. Idle mode is released by a reset or by an interrupt.

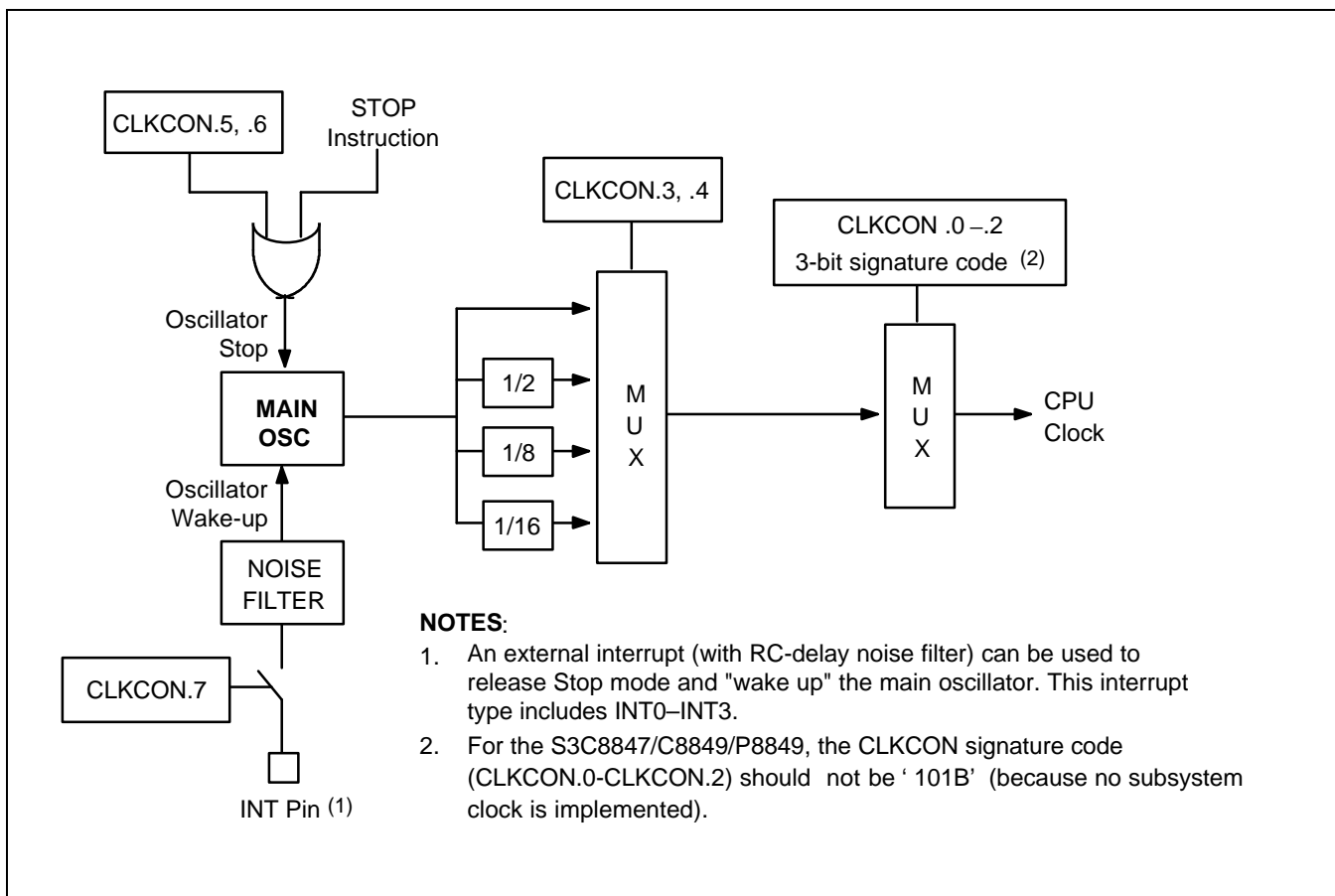


Figure 7-2. System Clock Circuit Diagram

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 S3C8847/C8849/P8849, 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'.

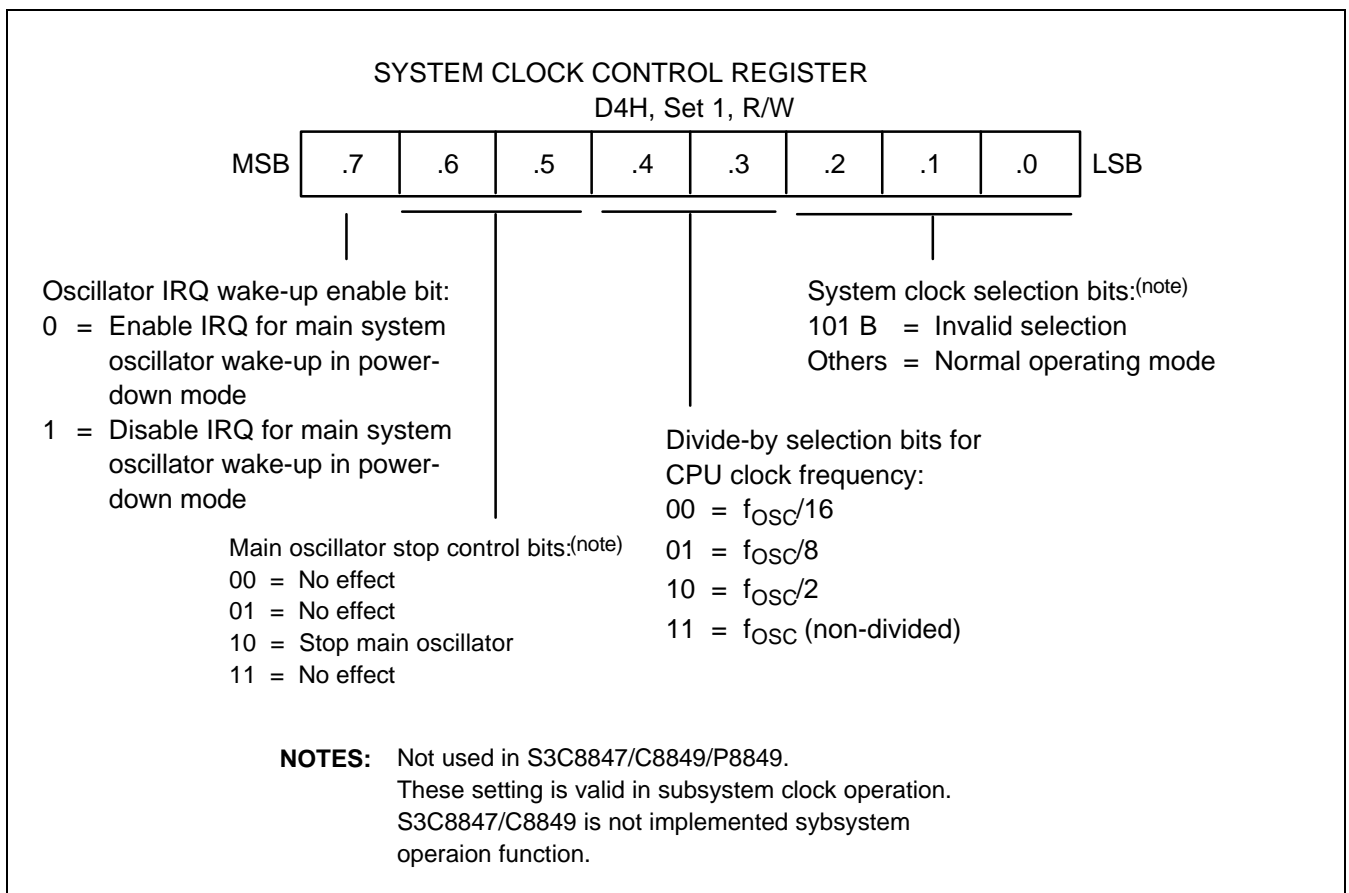


Figure 7-3. System Clock Control Register (CLKCON)

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 to 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 the V-sync input. The recommended L-C clock frequency is 6.5 MHz.

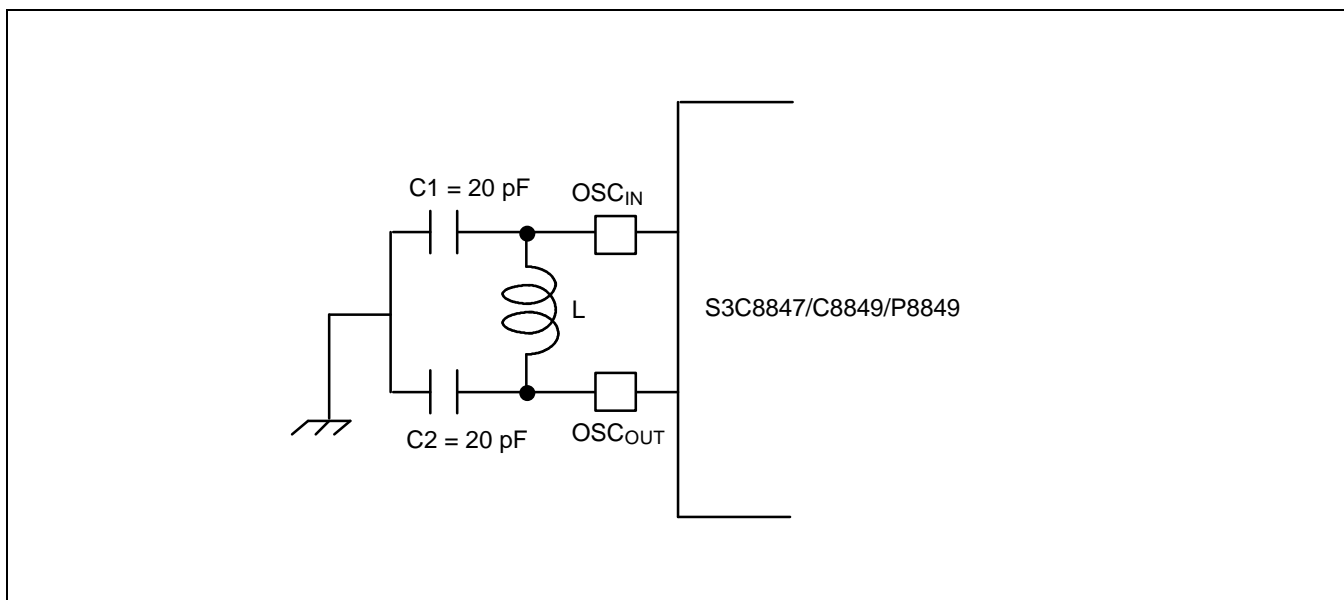


Figure 7-4. L-C Oscillator Circuit for OSD

8

RESET and POWER-DOWN

SYSTEM RESET

OVERVIEW

During a power-on reset, the voltage at V_{DD} is High level and the RESET pin is forced to Low level. The RESET signal is input through a Schmitt trigger circuit where it is then synchronized with the CPU clock. This brings the S3C8847/C8849/P8849 into a normal operating status.

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

When a reset occurs during the normal operation (that is, when V_{DD} and RESET are High level), the RESET pin is forced Low and the reset operation starts. All system and peripheral control registers are 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, 1, 2, and 3 are set to input mode except P1.2–P1.5, these ports set to N-channel open-drain output mode.
- 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 the 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 8-3 list the reset values for CPU and system registers, peripheral control registers, and peripheral data registers after a reset operation. The following notation is used to represent reset values:

- A "1" or a "0" shows the reset bit value as logic one or logic zero, respectively.
- An 'x' means that the bit value is undefined after a reset.
- A dash ('-') means that the bit is either not used or not mapped.

Table 8-1. 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	0
Timer 0 data register	T0DATA	209	D1H	1	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	0
Clock control register	CLKCON	212	D4H	0	0	0	0	0	0	0	0	0
System flags register	FLAGS	213	D5H	x	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	x
Stack pointer (low byte)	SPL	217	D9H	x	x	x	x	x	x	x	x	x
Instruction pointer (high byte)	IPH	218	DAH	x	x	x	x	x	x	x	x	x
Instruction pointer (low byte)	IPL	219	DBH	x	x	x	x	x	x	x	x	x
Interrupt request register	IRQ	220	DCH	0	0	–	0	0	0	0	0	0
Interrupt mask register	IMR	221	DDH	x	x	–	x	x	x	x	x	x
System mode register	SYM	222	DEH	0	–	–	x	x	x	0	0	0
Register page pointer	PP	223	DFH	0	0	0	0	0	0	0	0	0

Table 8-2. 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	0	0	0	0	0	0	0	0	0
Port 1 data register	P1	225	E1H	0	0	0	0	0	0	0	0	0
Port 2 data register	P2	226	E2H	0	0	0	0	0	0	0	0	0
Port 3 data register	P3	227	E3H	–	–	–	–	–	–	–	0	0
Port 0 control register (high byte)	P0CONH	228	E4H	1	1	1	1	0	0	0	0	0
Port 0 control register (low byte)	P0CONL	229	E5H	0	0	0	0	0	0	0	0	0
Port 1 control register (high byte)	P1CONH	230	E6H	0	0	0	0	1	1	1	1	1
Port 1 control register (low byte)	P1CONL	231	E7H	1	1	1	1	1	1	1	1	1
Port 2 control register (high byte)	P2CONH	232	E8H	0	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	0
Locations EAH in set 1, bank 0, are not mapped.												
Port 3 control register (low byte)	P3CONL	235	EBH	–	–	–	–	1	1	1	1	1
Locations ECH–EFH in set 1, bank 0, are not mapped.												
Timer A data register	TADATA	240	F0H	0	0	0	0	0	0	0	0	0
Locations F1H in set 1, bank 0, are not mapped.												
Timer A control register	TACON	242	F2H	0	0	0	0	0	0	0	0	–
Locations F3H in set 1, bank 0, are not mapped.												
PWM0 data register (main byte)	PWM0	244	F4H	1	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	1
PWM1 data register (extension byte)	PWM1EX	247	F7H	0	0	0	0	0	0	–	–	–
PWM control register	PWMCON	248	F8H	0	0	0	0	0	–	0	0	0
Capture A data register	CAPA	249	F9H	0	0	0	0	0	0	0	0	0
A/D converter control register	ADCON	250	FAH	x	–	0	0	x	x	x	x	x
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	0
External memory timing register	EMT	254	FEH	0	0	0	0	0	0	0	0	–
Interrupt priority register	IPR	255	FFH	x	x	–	x	x	x	x	x	x
Locations E0H–EFH in set 1, bank 1, are not mapped.												

Table 8-2. Set 1, Bank 0 Register Values after a Reset (Continued)

Register Name	Mnemonic	Address		Bit Values after a Reset								
		Dec	Hex	7	6	5	4	3	2	1	0	
OSD character size control register	CHACON	240	F0H	0	0	0	0	0	0	0	0	0
OSD fade control register	FADECON	241	F1H	0	0	0	0	0	0	0	0	0
OSD row position control register	ROWCON	242	F2H	0	0	0	0	0	0	0	0	0
OSD column position control register	CLMCON	243	F3H	0	0	0	0	0	0	0	0	0
OSD background color control register	COLCON	244	F4H	0	0	0	0	0	0	0	0	0
On-screen display control register	DSPCON	245	F5H	0	0	0	0	0	0	0	0	0
Halftone signal control register	HTCON	246	F6H	0	0	0	0	0	0	0	0	0
V-sync blank control register	VSBCON	247	F7H	–	–	–	0	1	0	0	0	1
PWM2 data register	PWM2	248	F8H	1	1	1	1	1	1	1	1	1
PWM3 data register	PWM3	249	F9H	1	1	1	1	1	1	1	1	1
PWM4 data register	PWM4	250	FAH	1	1	1	1	1	1	1	1	1
PWM5 data register	PWM5	251	FBH	1	1	1	1	1	1	1	1	1
OSD color buffer	COLBUF	252	FCH	–	–	–	x	x	x	x	x	x

Locations FDH–FFH in set 1, bank 1, are not mapped.

Table 8-3. 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–FBH, page 1	x	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 maximum 10 μ A. 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) from active (Low level) state. 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 the 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 S3C8847/C8849/P8849 interrupt structure that meet this requirement are INTO–INT3 (P1.0–P1.3). 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 a 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, the CPU operations are halted while selected peripherals remain active. During Idle mode, the internal clock signal is gated off to the CPU and all peripherals except the OSD block timer A counter, PWM and capture (CAPA). 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 S3C8847/C8849/P8849 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        02H
DW         CAPA_INT      ; Capture A interrupt

ORG        0BEH
DW         TIMERA_INT   ; Timer A interrupt

ORG        0C0H
DW         P10_INT      ; P1.0 external interrupt
DW         P11_INT      ; P1.1 external interrupt
DW         OSD_ROW_INT  ; OSD ROW interrupt
DW         P12_INT      ; P1.2 external interrupt
DW         P13_INT      ; P1.3 external interrupt

ORG        0D4H
DW         V_SYNC_INT   ; V-sync interrupt

ORG        0FCH
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)

```

; Enable OSD ROW interrupt
LD      HTCON,#2AH      ; Enable V-sync interrupt
LD      DSPCON,#0A0H    ; Disable OSD logic
SB0     ; Select bank 0
LD      PWMCON,#0E9H    ; Prescaler ← 4
; Enable PWM counter
; Enable capture A interrupt
LD      IPR,#0AEH       ; Interrupt priority settings
; IRQ6 > 7 > 3 > 2 > 0 > 1
LD      IMR,#0CCH       ; Enable level 2, 3, 6, and 7 interrupts
LD      P0CONH,#00H     ; Input mode
LD      P0CONL,#0FFH    ; Push-pull output mode
LD      P1CONH,#0FFH    ; Output mode
LD      P1CONL,#00H     ; Input mode
LD      P2CONH,#00H     ; Input mode
LD      P2CONL,#00H     ; Input mode
LD      P3CONL,#00H     ; Input 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
                .
                .
                .
                SB1
                LD      HTCON, #3AH ; Clear pending bit
                POP     RP1
                POP     RP0
                POP     PP
                IRET          ; Return from interrupt service routine

OSD_ROW_INT:
                PUSH    PP
                PUSH    RP0
                PUSH    RP1
                .
                .
                .
                SB1
                LD      HTCON, #2BH ; Clear pending bit
                POP     RP1
                POP     RP0
                POP     PP
                IRET

P10_INT:        ; P1.0 external interrupt
P11_INT:        ; P1.1 external interrupt
P12_INT:        ; P1.2 external interrupt
P13_INT:        ; P1.3 external interrupt
TIMER0_INT:    ; Timer 0 interrupt
                IRET          ; Return from interrupt service routine
    
```

9

I/O PORTS

OVERVIEW

The S3C8847 and the S3C8849/P8849 microcontrollers have four I/O ports with a total of 26 pins. Up to 10 pins can be configured as n-channel open-drain outputs. Of these 10 open-drain pins, 6 pins can withstand loads of up to 6 volts and 4 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 9-1. S3C8847/C8849/P8849 Port Configuration Overview

Port	Configuration Options	Programmability
0	General I/O port, configurable for digital input or push-pull output. Pins P0.6–P0.7 are multiplexed to support alternative function.	Bit programmable
1	General I/O port, configurable for digital input or n-channel open-drain output. Pins 1.0–P1.5 can withstand up to 6-volt loads. Pins 1.0–P1.3 are multiplexed to support alternative functions.	Bit programmable
2	General I/O port, configurable for n-channel open-drain or push-pull output mode by software. Pins can withstand up to 5-volt loads. Each pin has an alternative function.	Bit programmable
3	General 2-bit I/O port, configurable for digital input or n-channel open-drain output. Pins can withstand up to 5 V. P3.0–P3.1 can be alternately used as external interrupt inputs ADC0–ADC1.	Bit programmable

PORT DATA REGISTERS

Data registers for ports 0–3 have the structure shown in Figure 9-1. Table 9-2 gives you an overview of the port data register locations:

Table 9-2. 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

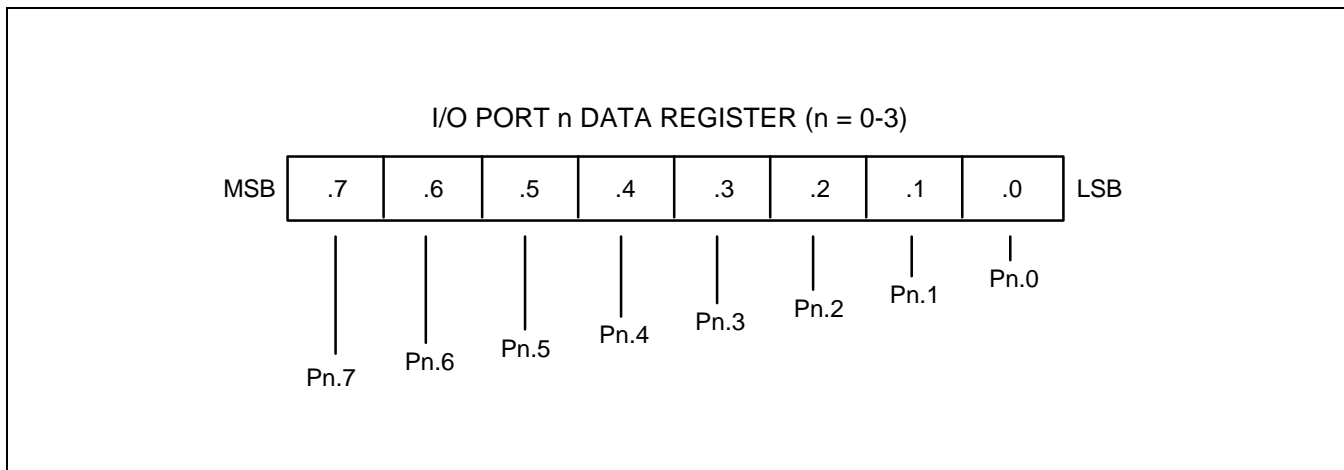


Figure 9-1. Port Data Register Format

PORT 0

Port 0 is a bit-programmable general I/O port. Port 0 is accessed directly by writing or reading the port 0 data register, P0 (E0H, set 1, bank 0).

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 P0CONL controls I/O for the lower byte pins.

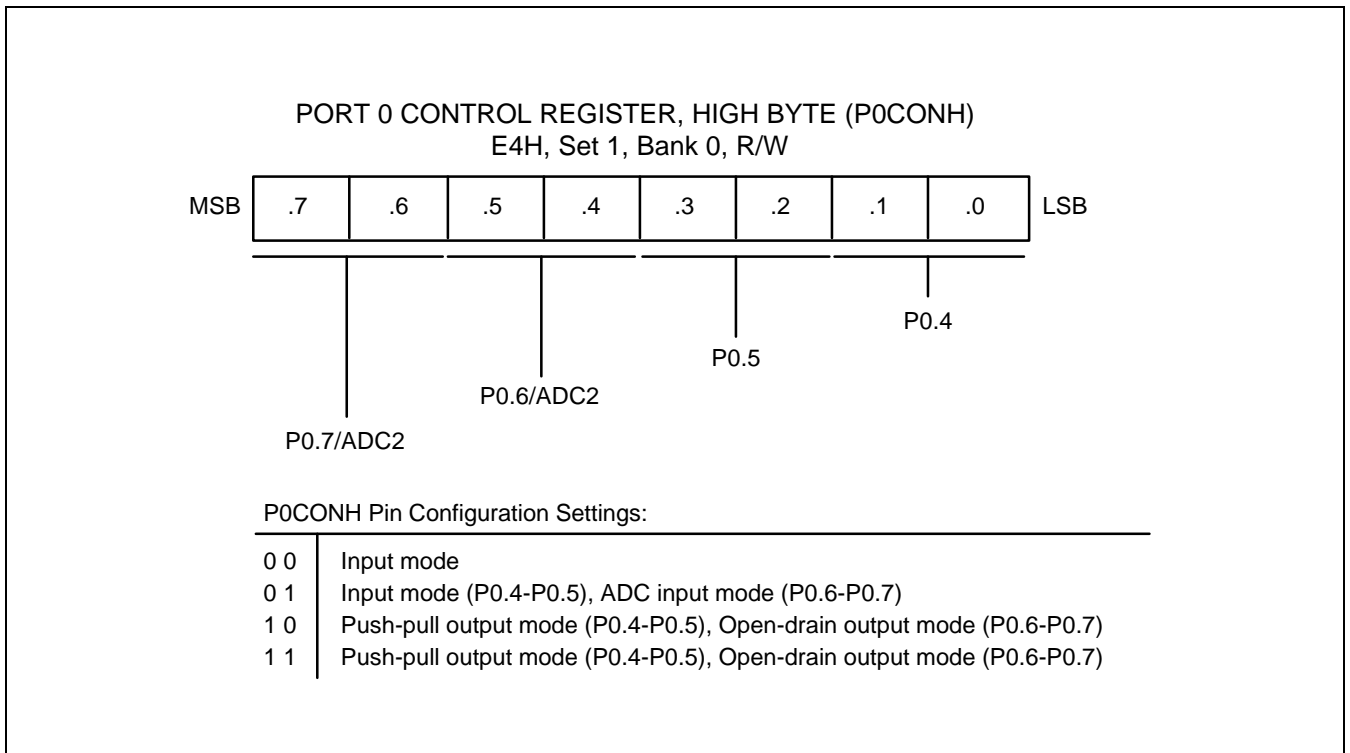


Figure 9-2. Port 0 High-Byte Control Register (P0CONH)

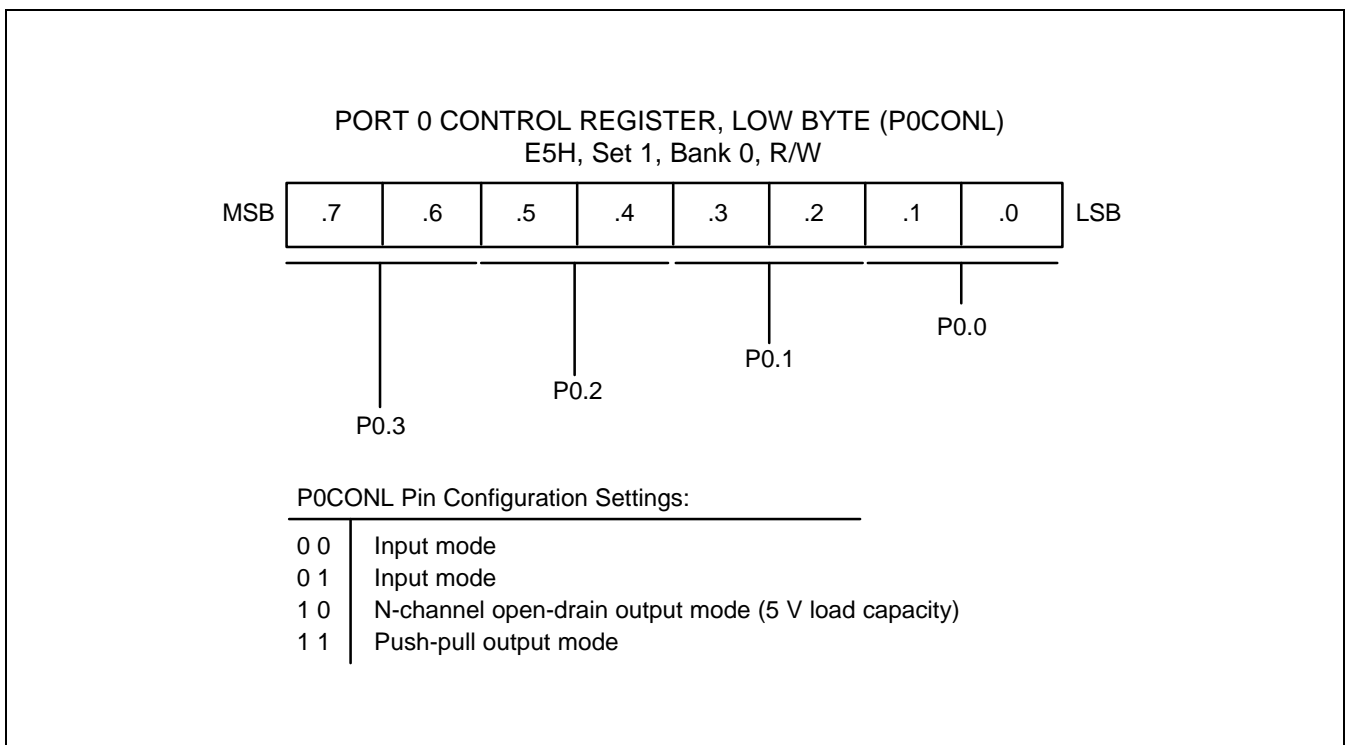


Figure 9-3. Port 0 Low-Byte Control Register (P0CONL)

PORT 1

Port 1 is 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.

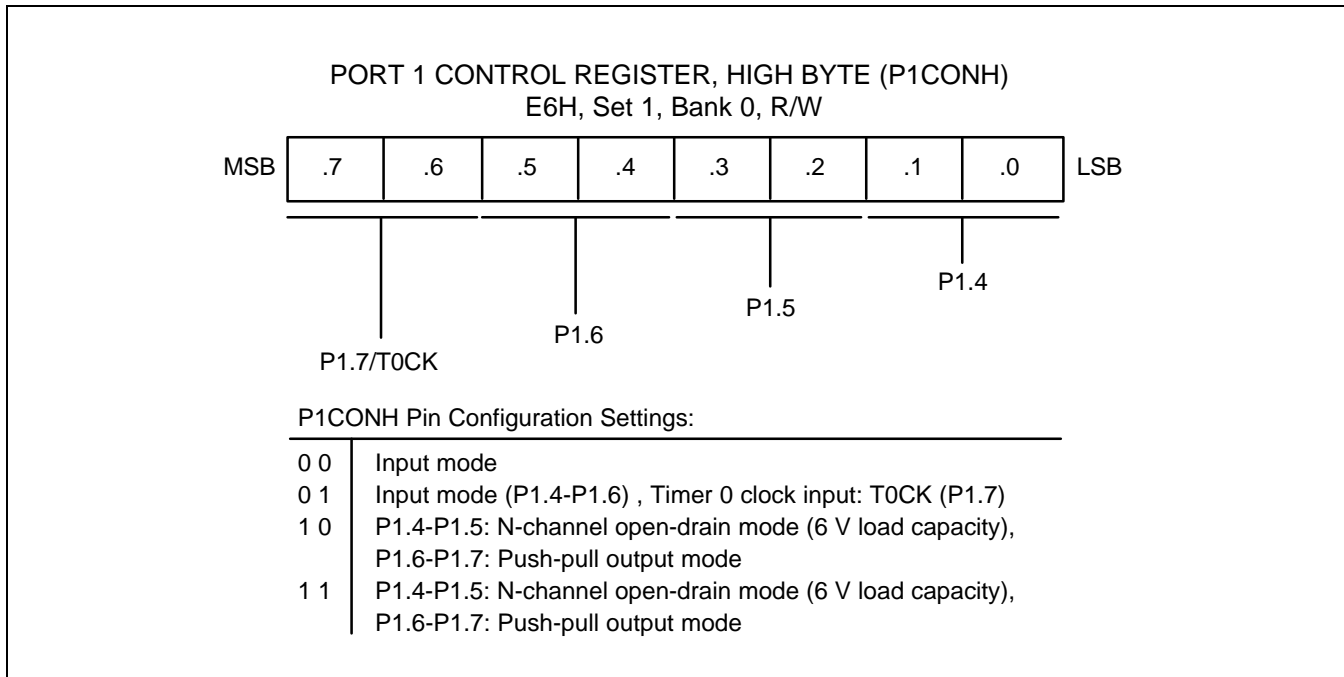


Figure 9-4. Port 1 High-Byte Control Register (P1CONH)

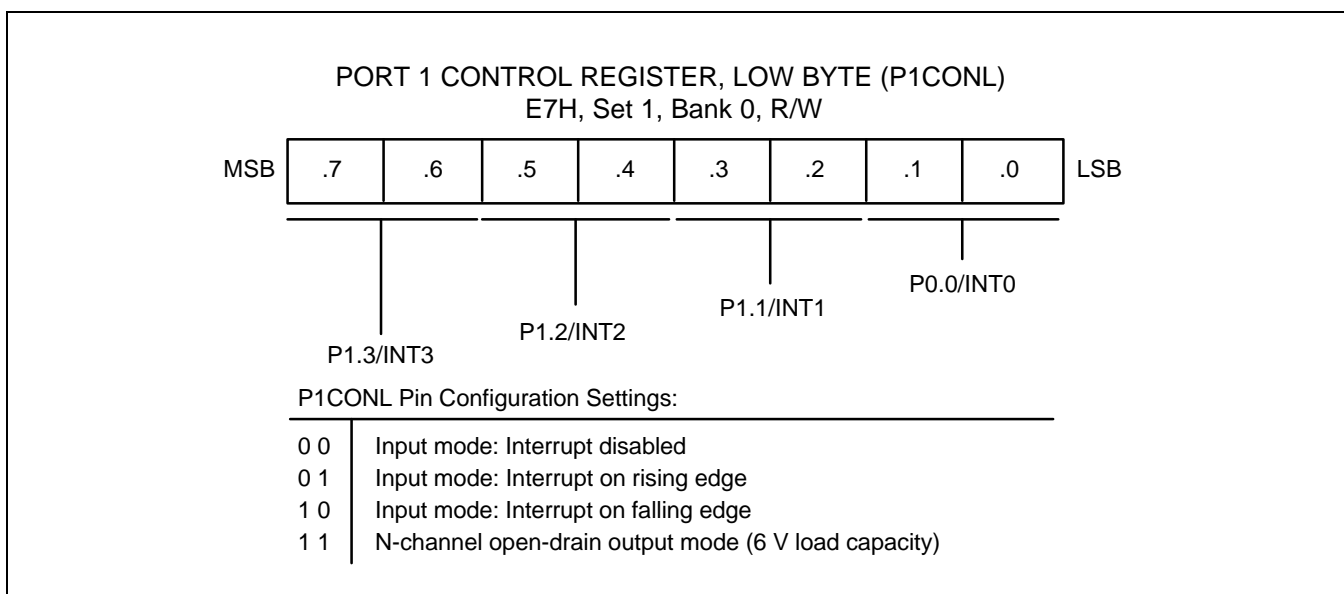


Figure 9-5. Port 1 Low-Byte Control Register (P1CONL)

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 normal input mode (P2.0–P2.3) and input mode (2.4–P2.7). You use P2CONH and P2CONL register settings to configure individual port 2 pins:

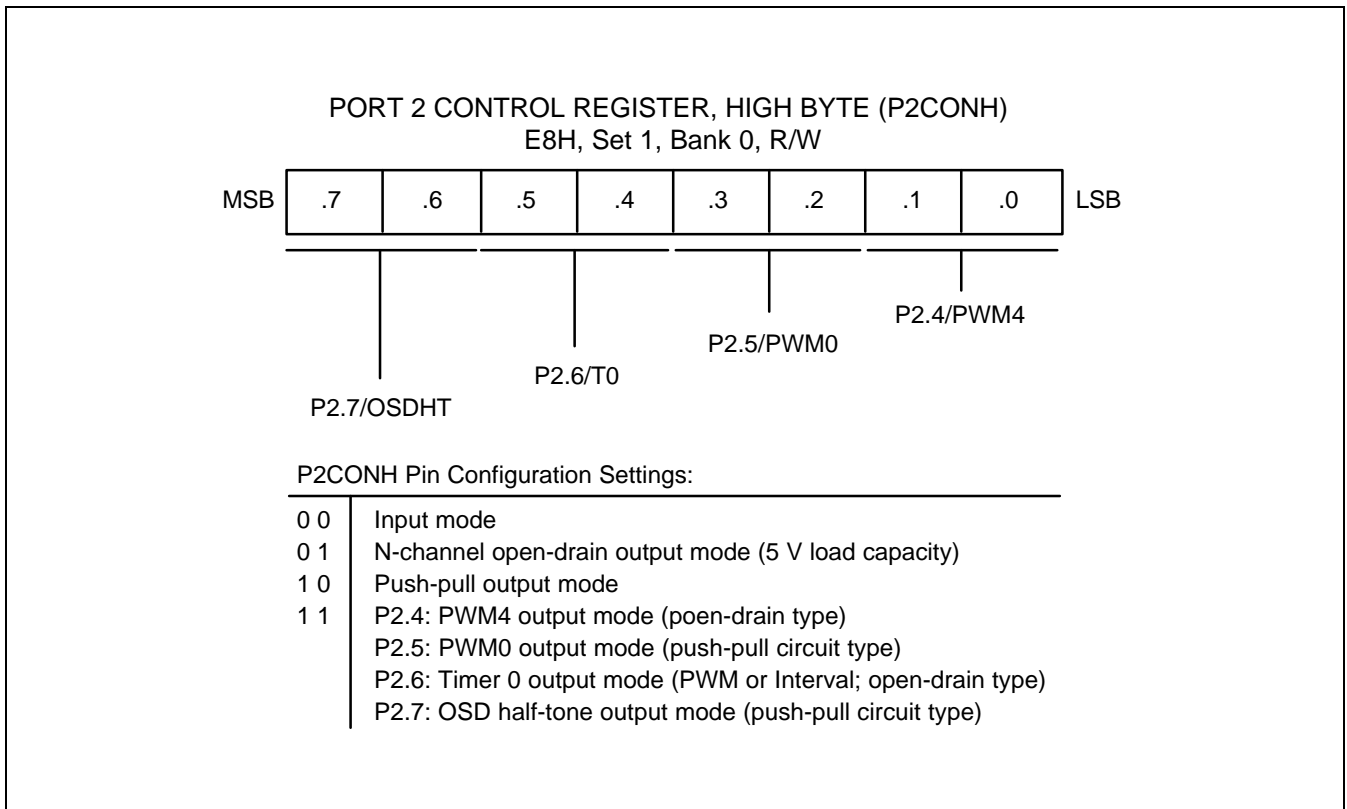


Figure 9-6. Port 2 High-Byte Control Register (P2CONH)

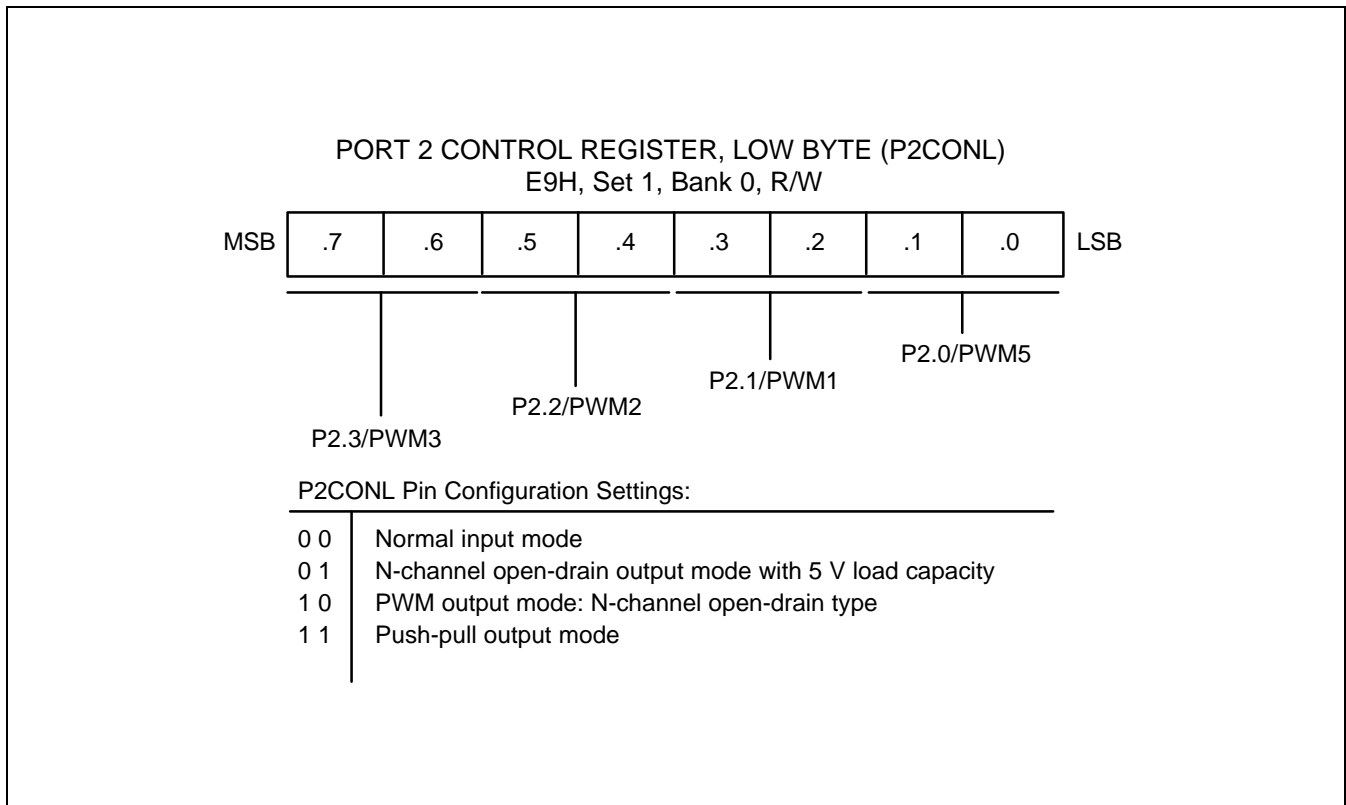


Figure 9-7. Port 2 Low-Byte Control Register (P2CONL)

PORT 3

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

A reset operation sets the P3 data register to '00H', and the port 3 control register to '0FH', configuring the port 3 pins to output (open-drain) mode.

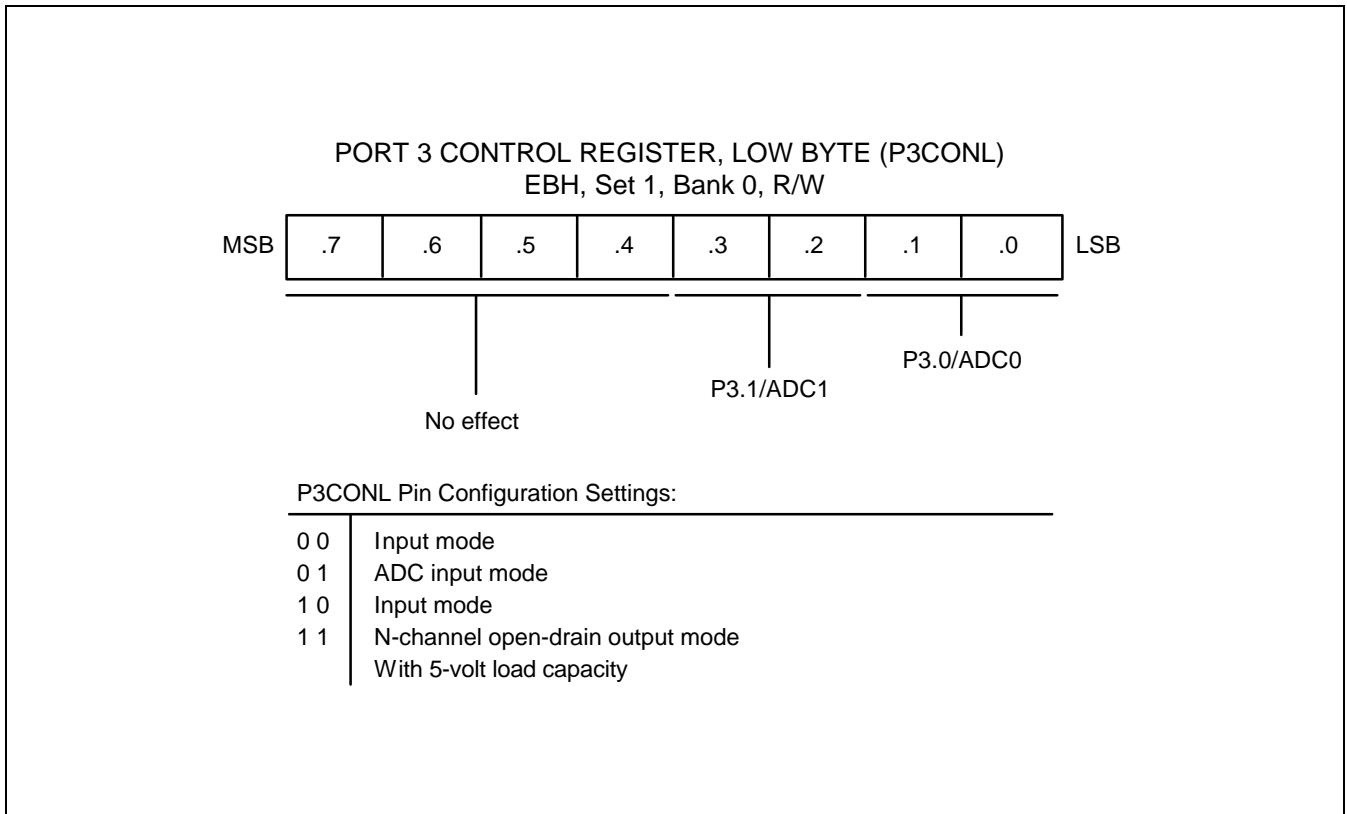


Figure 9-8. Port 3 Low-Byte Control Register (P3CONL)

PROGRAMMING TIP — Configuring I/O Port Pins to Specification

The following sample program shows you how to configure the KS88C8216/C8224/C8316/C8324 I/O ports to specification. The following parameters are given for ports 0, 1, 2, and 3:

- Set P0.0 and P0.1 to input mode
- Set P0.2 and P0.3 to output mode
- Set P0.4 and P0.5 to input mode
- Set P0.6 and P0.7 to open-drain output mode
- Set P1.0–P1.1 to interrupt rising edge mode
- Set P1.2–1.5 to open-drain output mode
- Set P1.6– P1.7 to push-pull output mode
- Set P2.0 and P2.1 to open-drain output mode
- Set P2.2–P2.4 to input mode
- Set P2.6–2.7 to push-pull output mode
- Set P2.5 to PWM0 output mode
- Set P3.0–P3.1 to ADC input mode

```

•
•
•
SB0                ; Select bank 0

LD      P0CONH,#0F0H ; P0.4, P0.5 ← Input mode
                          ; P0.6, P0.7 ← Open-drain output mode
LD      P0CONL,#0F0H ; P0.0, P0.1 ← Input mode
                          ; P0.2, P0.3 ← Output mode

LD      P1CONH,#0FFH ; P1.6–1.7 ← Push-pull output mode
                          ; P1.2–1.5 ← Open-drain output mode
LD      P1CONL,#0F5H ; P1.0, P1.1 ← Interrupt rising edge mode

LD      P2CONH,#0ACH ; P2.4 ← Input mode
                          ; P2.6, 2.7 ← Push-pull output mode
                          ; P2.5 ← PWM0 output mode
LD      P2CONL,#05H  ; P2.0, P2.1 ← Open-drain output mode
                          ; P2.2, P2.3 ← Input mode

LD      P3CONL,#05H  ; P3.0, P3.1 ← ADC input mode
•
•
•

```

PROGRAMMING TIP — Clearing Port 0 Interrupt Pending Bits

This sample program shows you how to clear the interrupt pending bits for port 1. The program parameters are as follows:

- Enable only the interrupt level 1 (IRQ1) for P1.0–P1.1
- Set the interrupt priorities as P1.0 > P1.1

```

                ORG      0C0H
                VECTOR   EXT_INT_P10
                VECTOR   EXT_INT_P11
                .
                .
                .
RESET          ORG      0100H
                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      P1CONL,#0AH ; P1.0, P1.1 ← Input mode; falling edge interrupts
                .
                .
                .
                SRP      #0C0H     ; Set register pointer to 0C0H
                EI                ; Enable interrupts
                .
                .
                .
MAIN          NOP
                NOP
                .
                .
                .
                JP      T,MAIN
                .
                .
                .

```

(Continued on next page)

PROGRAMMING TIP — Clearing Port 1 Interrupt Pending Bits (Continued)

```
EXT_INT_P10:                                ; P1.0 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_P11:                                ; P1.1 external interrupt service
        PUSH        PP
        PUSH        RP0
        PUSH        RP1
        .
        .
        .
        POP         RP1
        POP         RP0
        POP         PP
        IRET
```

10 BASIC TIMER and TIMER 0

MODULE OVERVIEW

The S3C8847 and the S3C8849/P8849 microcontrollers have two default timers: an 8-bit *basic timer (BT)* and an 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 that provides 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)
- Clock frequency divider (f_{OSC} divided by 4096, 256, or 8) with multiplexer
- 8-bit counter (T0CNT), 8-bit comparator, and 8-bit reference data register (T0DATA)
- Timer 0 match 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, clear the basic timer counter and frequency dividers, and 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 the normal operation by writing a "1" to BTCON.1. To clear the frequency dividers for both the basic timer input clock and the timer 0 clock, you should write a "1" to BTCON.0.

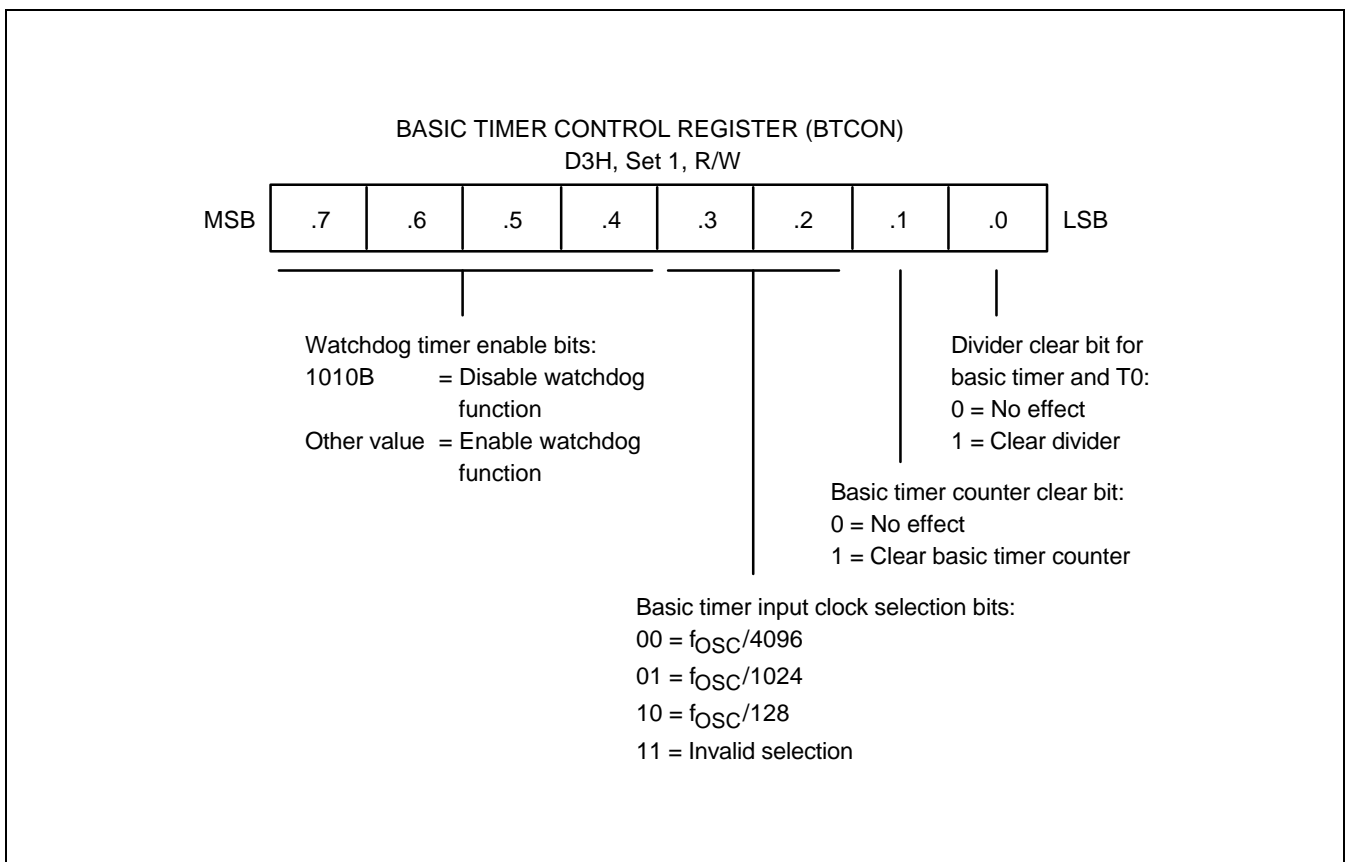


Figure 10-1. Basic Timer Control Register (BTCON)

BASIC TIMER FUNCTION DESCRIPTION

Watchdog Timer Function

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

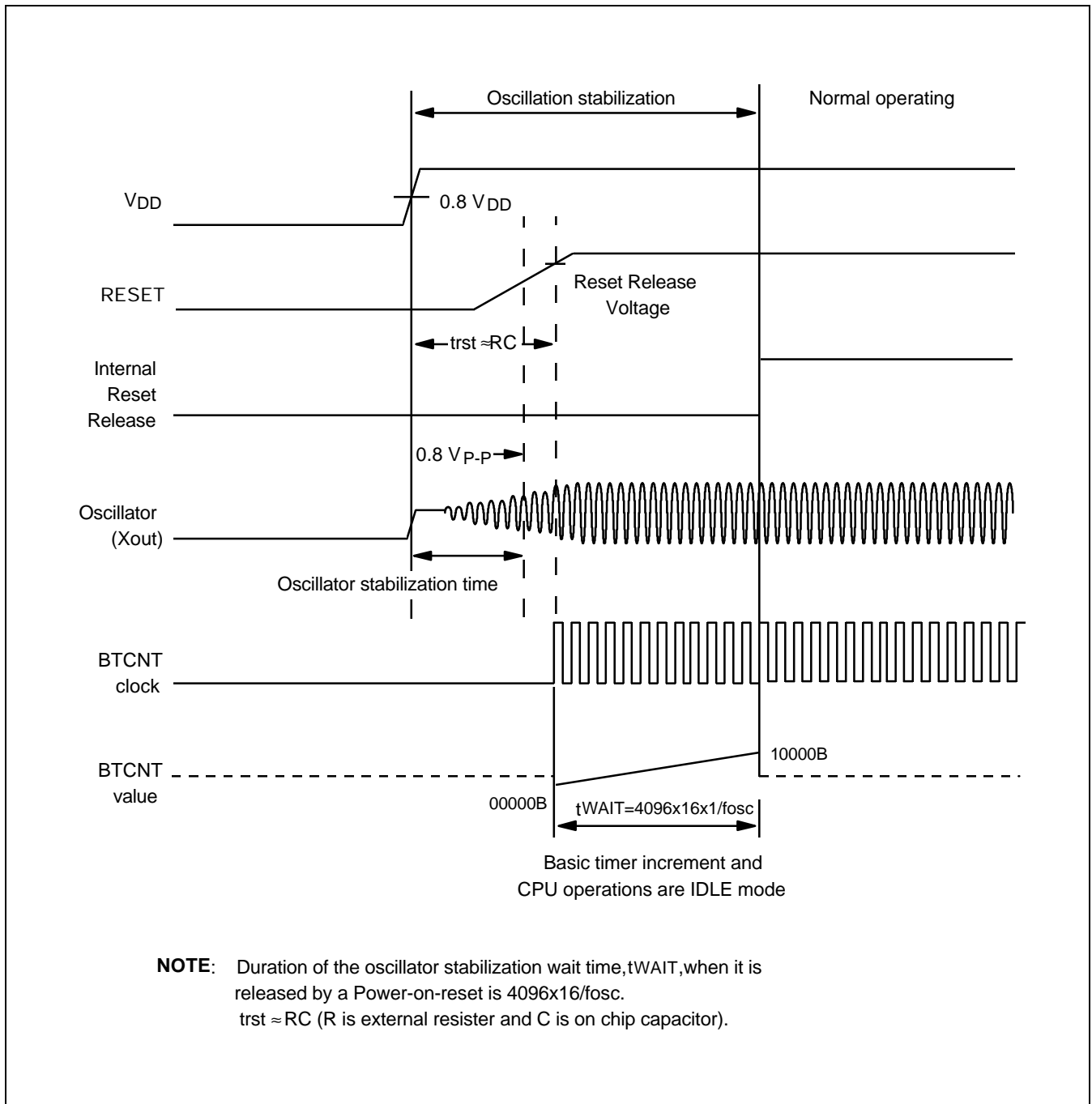


Figure 10-2. Oscillation Stabilization Time on RESET

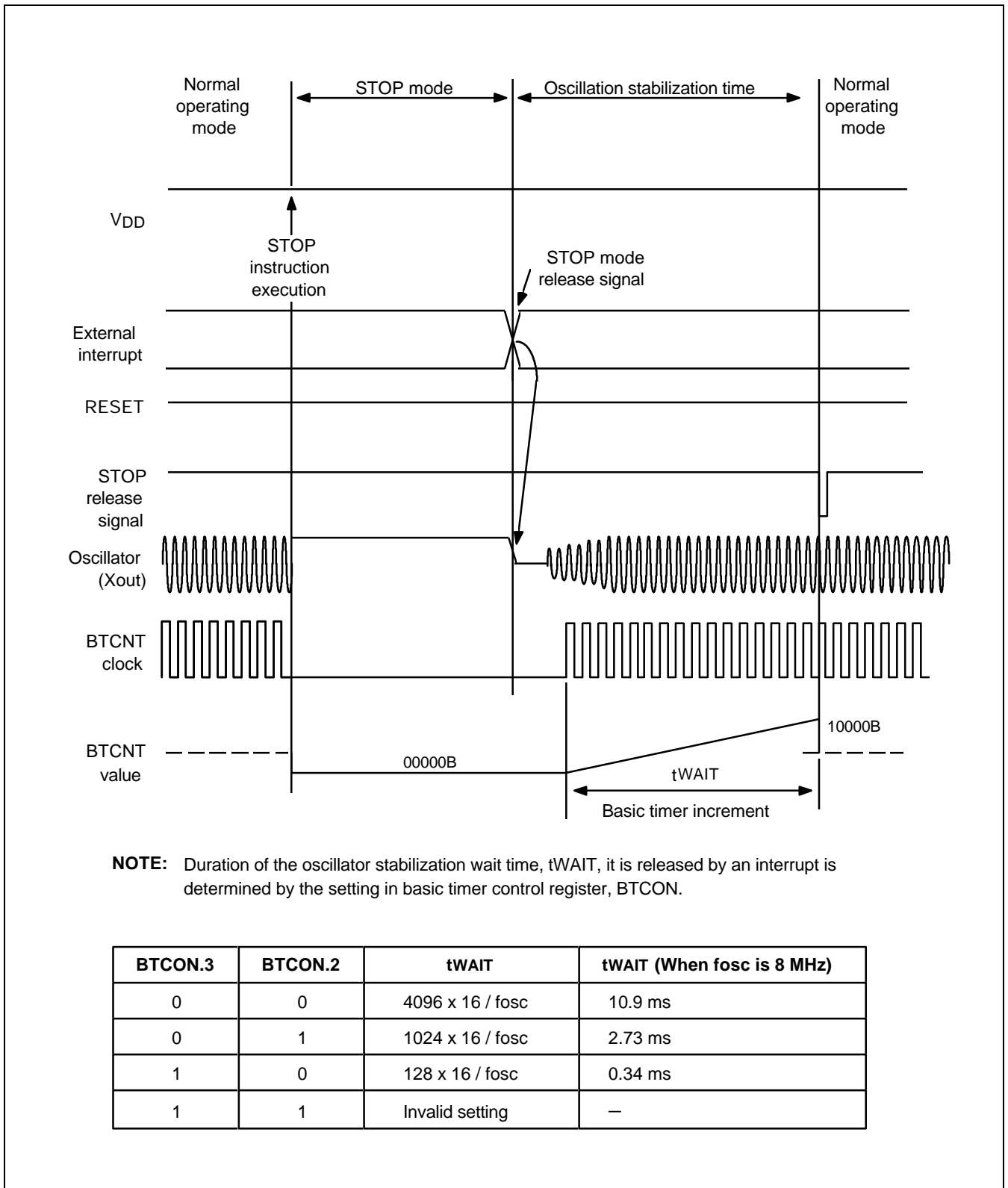


Figure 10-3. Oscillation Stabilization Time on STOP Mode Release

TIMER 0 CONTROL REGISTER (T0CON)

The timer 0 control register, T0CON, is used to select the timer 0 operating mode (interval timer) and input clock frequency, clear the timer 0 counter, and enable the T0 match interrupt. It also contains a pending bit for T0 match interrupt. 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 match interrupt. The T0 counter can be cleared at any time during the normal operation by writing a "1" to T0CON.3.

To enable the T0 match 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.

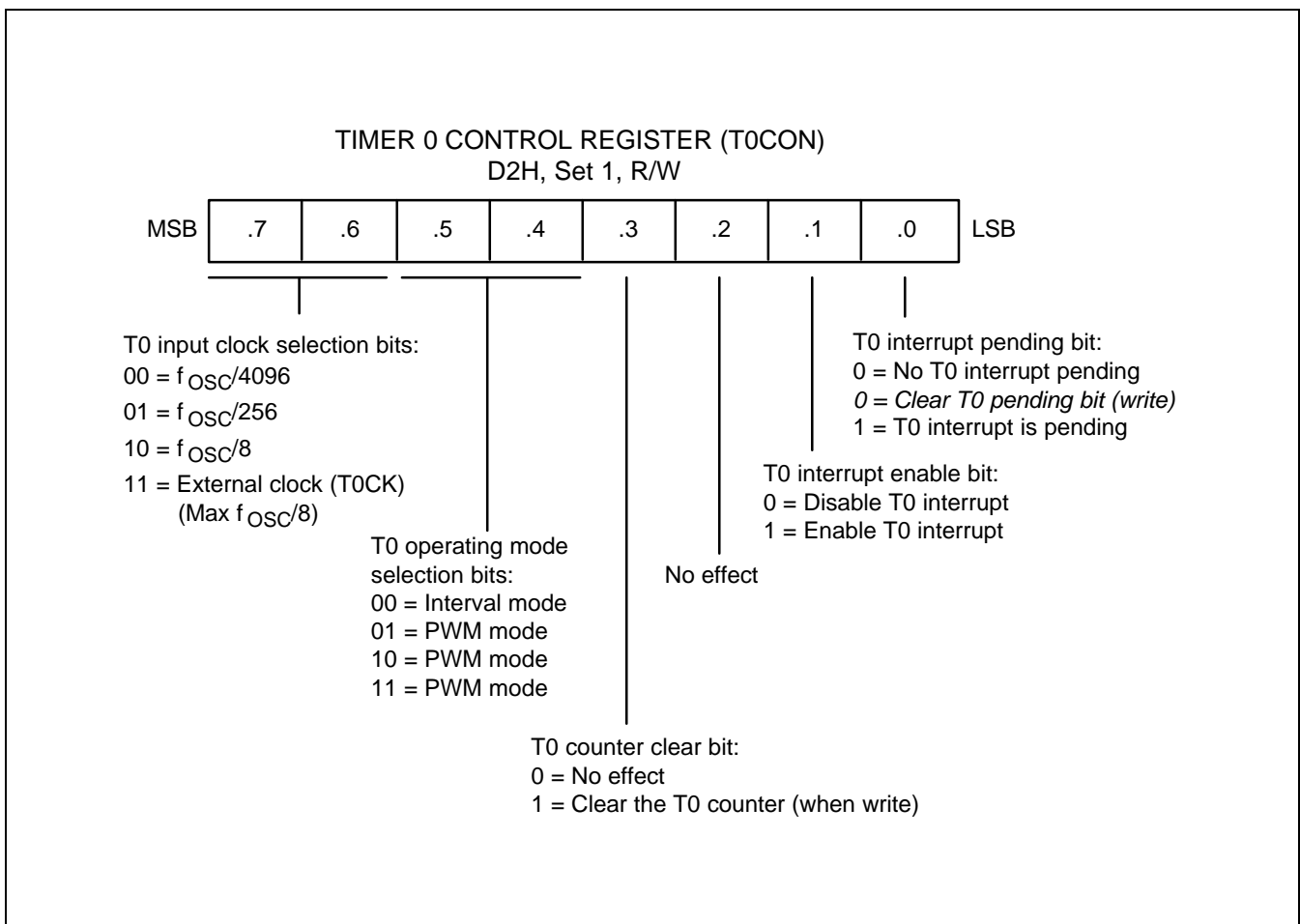


Figure 10-4. Timer 0 Control Register (T0CON)

TIMER 0 FUNCTION DESCRIPTION

T0 Interrupts (IRQ0, Vector FCH)

The T0 module can generate one interrupt: the timer 0 match interrupt (T0INT). T0INT is also in the level IRQ0, vector address: FCH. 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.

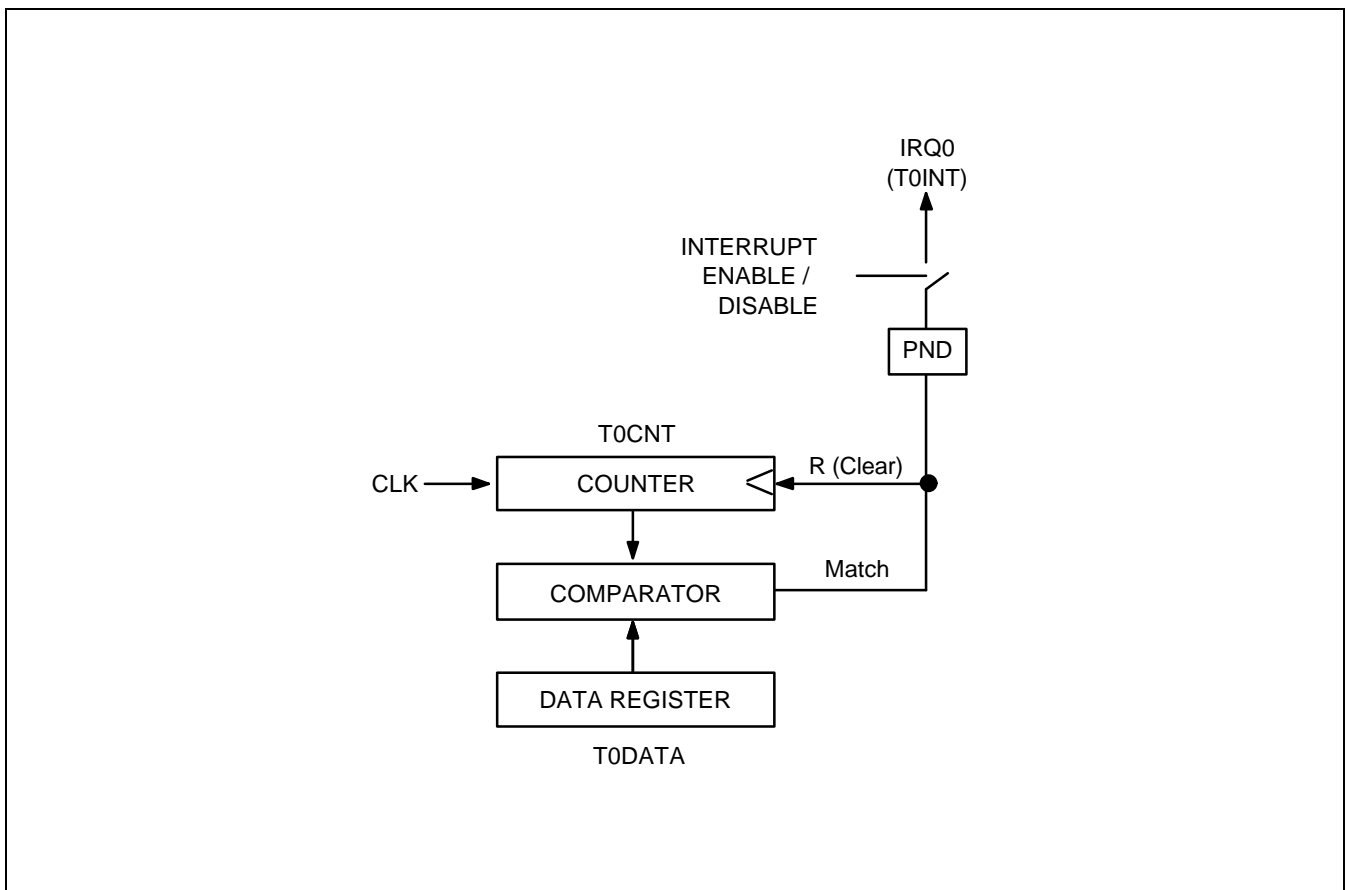


Figure 10-5. Timer 0 Function Diagram (Interval Timer Mode)

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 10-6)

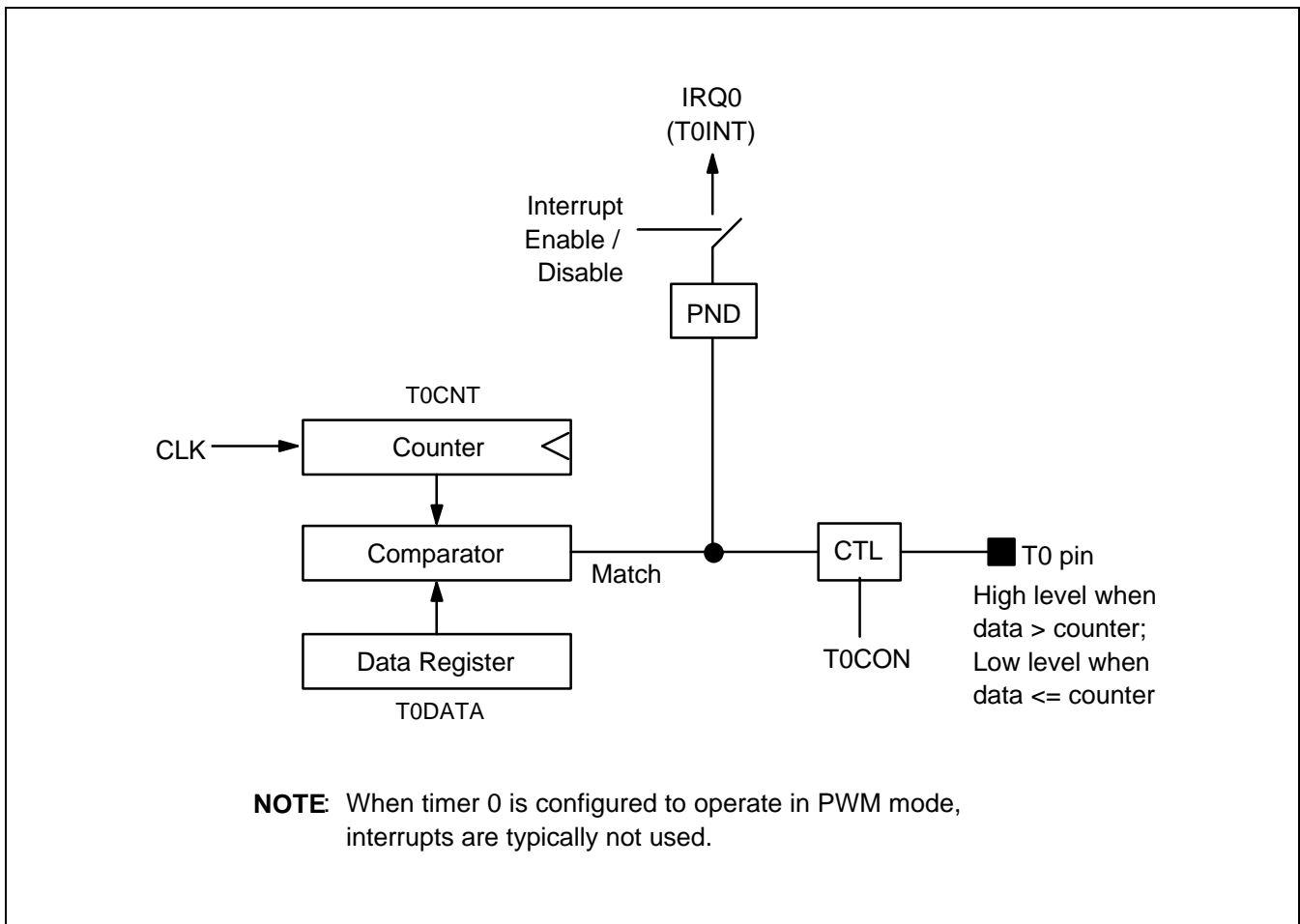


Figure 10-6. Timer 0 Function Diagram (PWM Mode)

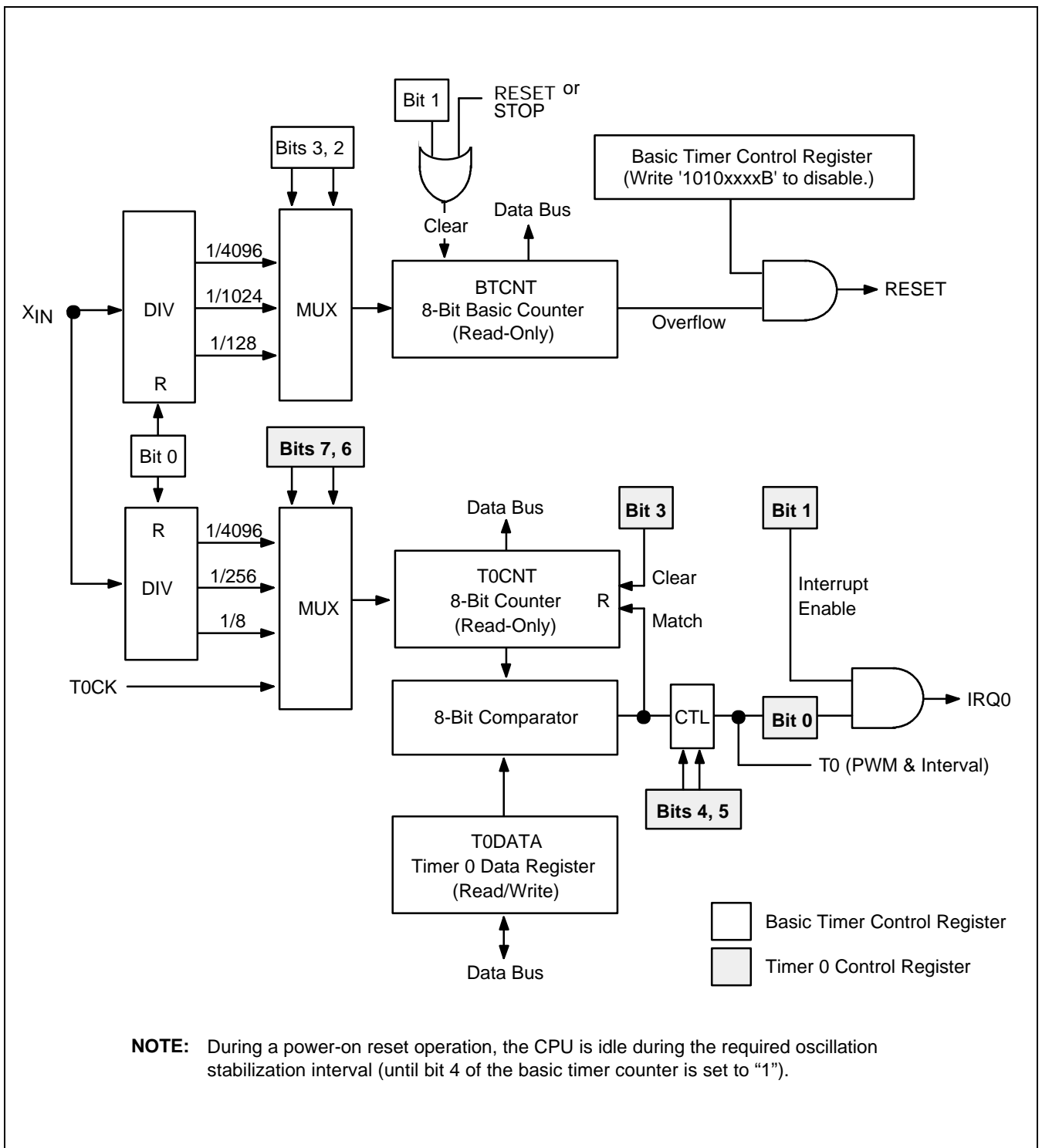



Figure 10-7. Basic Timer and Timer 0 Block Diagram

 **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, determining the frequency of the oscillator clock, and 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      0FCH          ; Timer 0 interrupt (match)
                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
              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)

NO_200MS_SET:
              LD          T0CON,#42H ; Clear pending bit
              POP         RP0        ; Restore register pointer 0 value
              POP         PP         ; Restore page pointer value
              IRET          ; Return from interrupt service routine

```


11

TIMER A

OVERVIEW

The S3C8847 and the S3C8849 microcontrollers have an 8-bit timer/counter (timer A). Each timer has a control register, an 8-bit counter register, an 8-bit data register, an 8-bit comparator. Timer A runs continuously. Counter register addresses are not mapped and they cannot, therefore, be read or written.

TIMER CLOCK INPUT

Timer A has different clock input options. You can select the non-divided CPU clock or the CPU clock divided by 1000. The selected clock input frequency for each timer can be scaled using the 4-bit prescaler that is located in bits 4–7 of the TACON register.

TIMER A INTERRUPT CONTROL

Timer A generate a match signal when the count value is equal to the referenced data value in the TADATA. When the interrupt enable bit is set for timer A, an interrupt is generated whenever a match is detected. The corresponding count register is then cleared and counting resumes. To enable the timer A interrupt, you should set TACON.2 to "1".

The timer A interrupt pending bit is TACON.1. When a timer A pending bit read operation shows a "0" value, no interrupt is pending; when it is "1", an interrupt request is pending. When the request is acknowledged by the CPU and the service routine starts, the pending bit must be cleared by the interrupt service routine. To do this, you must write a "0" to the appropriate bit location.

TIMER A FUNCTION DESCRIPTION

When a match occurs, the timer is reset to zero.

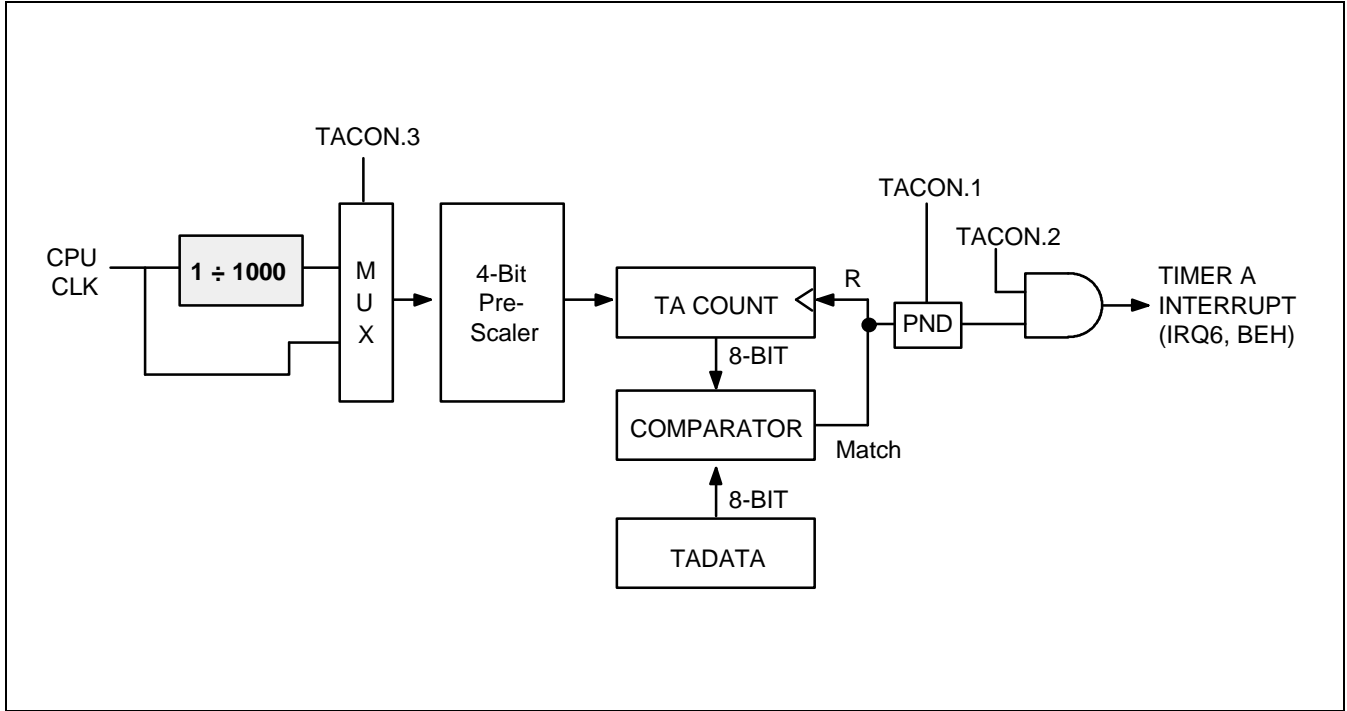


Figure 11-1. Timer A Block Diagram

TIMER A CONTROL REGISTER (TACON)

The timer A control register, TACON, is located at F2H in set 1, bank 0. All bits are read/write addressable. The TACON register settings control four functions:

- Interrupt enable/disable
- Interrupt pending control (read for status, write to clear)
- Clock source selection
- Prescaler (4-bit) for timer clock input

TACON.1 is the pending flag for the timer A interrupt (IRQ6, vector BEH). Application software can poll the TAIP bit to detect timer A interrupt requests. When an interrupt request is acknowledged, the interrupt service routine must clear TACON.1 by writing a "0" to the bit location.

Note that there are two clock source selections for timer A: the CPU clock divided by 1000 or the non-divided CPU clock.

A reset clears TACON to '00H', selecting the CPU clock/1000, and disabling the timer A interrupt.

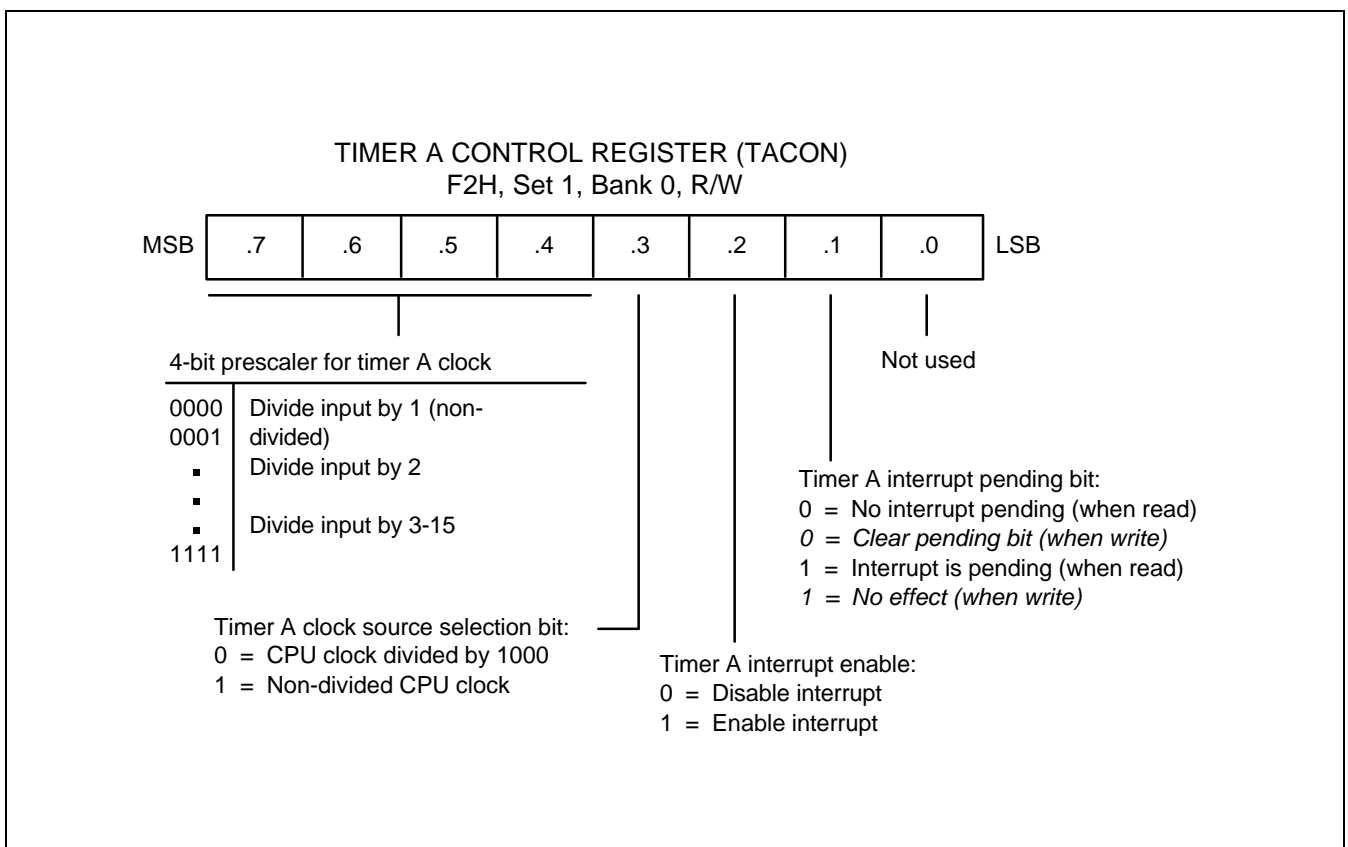


Figure 11-2. Timer A Control Register (TACON)

 **PROGRAMMING TIP — Configuring Timer A**

This example sets timer A to normal interval mode, determining the oscillation frequency of the timer clock, the execution sequence that follows a timer A interrupt. The program parameters are:

- The timer interval is set to 10 milliseconds
- Oscillation frequency = 6 MHz
- General register 70H (page 0) ← 70H + 71H + 72H + 73H + 74H (page 0) is executed after a timer A interrupt

```

                ORG      0BEH          ; Timer A interrupt
                VECTOR  TAIN

RESET          ORG      0100H
                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      TACON,#54H   ; 01010100B
                ; PS ← 5 (divide-by-6)
                ; CPU clock/1000
                ; Select interval mode for timer A
                LD      TADATA,#59H  ; 10-ms interval time
                ; (6 MHz/1000) ÷ (59 + 1) = 100 Hz (10 ms)
                SRP     #0C0H        ; Set register pointer ← 0C0H
                EI              ; Enable interrupts
                .
                .
                .
TAIN           PUSH     PP           ; Save page pointer to stack
                PUSH     RP0        ; Save register pointer 0 to stack
                SB0           ; Select bank 0
                LD      PP,#00H     ; Page pointer ← 00H (select page 0)
                SRP0     #70H       ; RP0 ← 70H
                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,#64H     ; 100 × 10 ms = 1000 ms (1 second)
                JR      ult,NO_1SEC_SET
                BITS     R1.2       ; Bit setting (71.2H)
    
```

(Continued on next page)

 **PROGRAMMING TIP — Configuring Timer A (Continued)**

```
NO_1SEC_SET:
    LD      TACON,#54H      ; Clear pending bit
    POP    RP0              ; Restore register pointer 0 value
    POP    PP               ; Restore page pointer value

    IRET                    ; Return from interrupt service routine
```

12 PWM and CAPTURE

PWM/CAPTURE MODULE

The S3C8847 and the S3C8849 microcontrollers 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 the PWM circuits is controlled by a single control register, PWMCON. PWMCON also contains a 3-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 14-bit incrementing counter. It is used by the 14-bit PWM circuits. To start the counter and enable the PWM circuits, you must 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 3-bit prescaler controls the clock input frequency to the PWM counter. By modifying the prescaler value, you can divide the input clock by one (non-divided), two, three, four, five, six, seven, or eight. The prescaler output is the clock frequency of the PWM counter.

PWM CONTROL REGISTER (PWMCON)

The control register for the PWM module, PWMCON, is located at the register address F8H in set 1, bank 0. Bit settings in the PWMCON register control the following functions:

- 3-bit prescaler for scaling the PWM counter clock
- Stop/start (or resume) the PWM counter operation
- Capture A interrupt enable and capture A edge selection

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

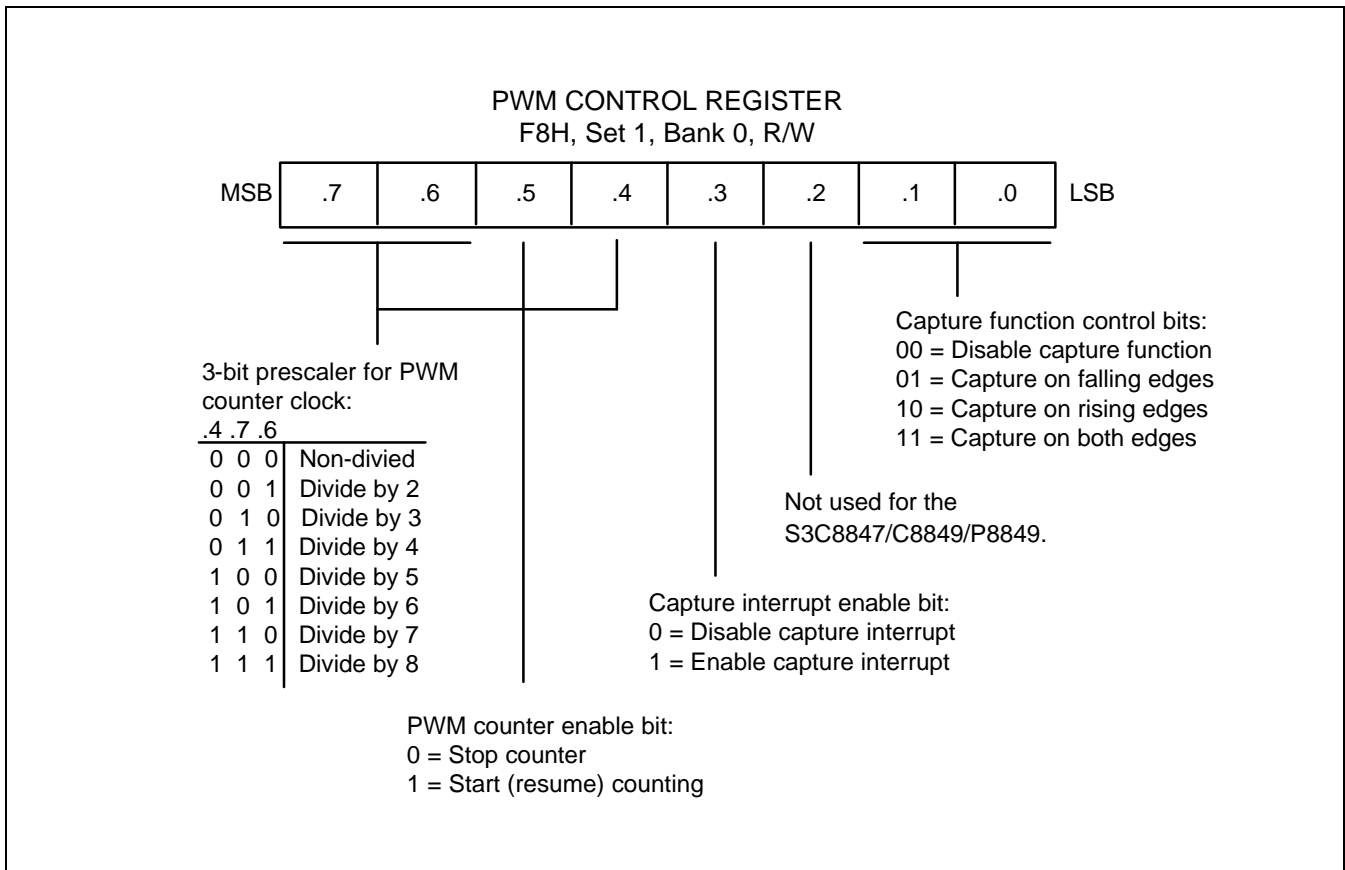


Figure 12-1. PWM Control Register (PWMCON)

PWM2–PWM5

The S3C8847/C8849 microcontrollers have four 8-bit PWM circuits, called PWM2–PWM5. These 8-bit circuits have the following components:

- 14-bit counter with 3-bit prescaler
- 8-bit comparators
- 8-bit PWM data registers (PWM2–PWM5)
- PWM output pins (PWM2–PWM5)

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

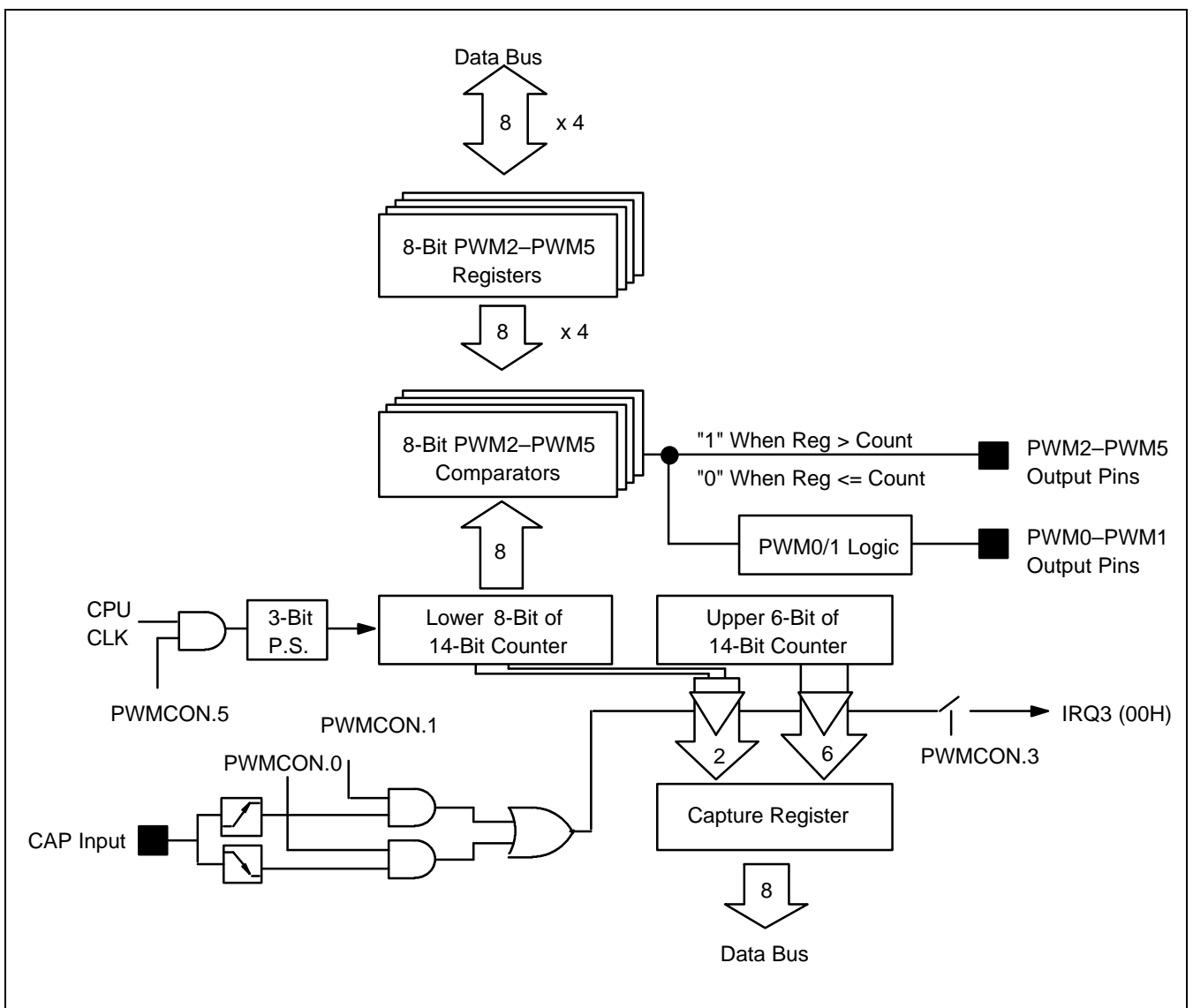


Figure 12-2. Block Diagram for PWM2–PWM5

PWM2–PWM5 FUNCTION DESCRIPTION

All the 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 14-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 the following 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 the four 8-bit PWM circuits. The output of PWM3, PWM4, and PWM5 are delayed by one-half of a counter clock for subsequent clock cycles (see Figure 12-4).

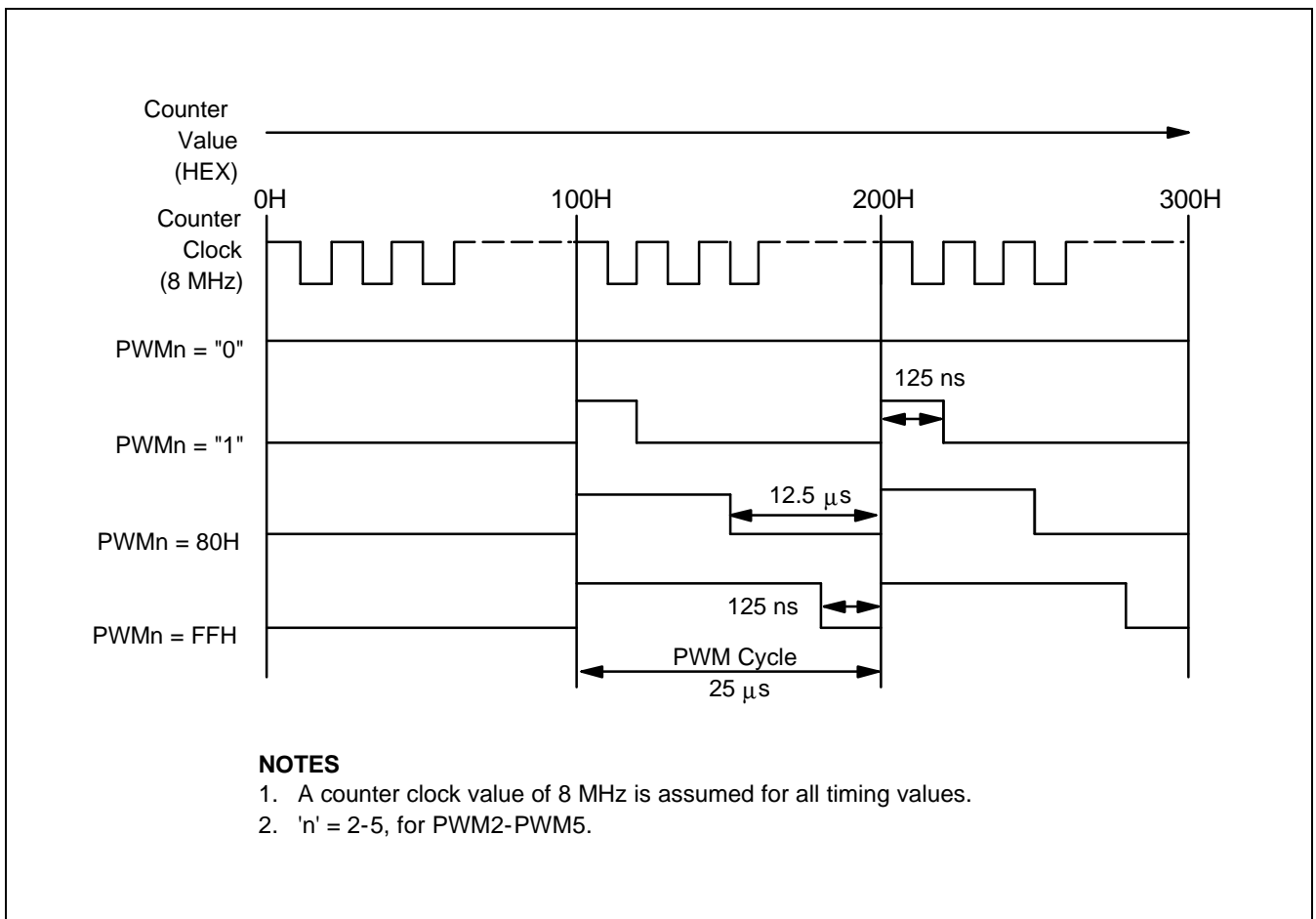


Figure 12-3. PWM Waveforms for PWM2–PWM5

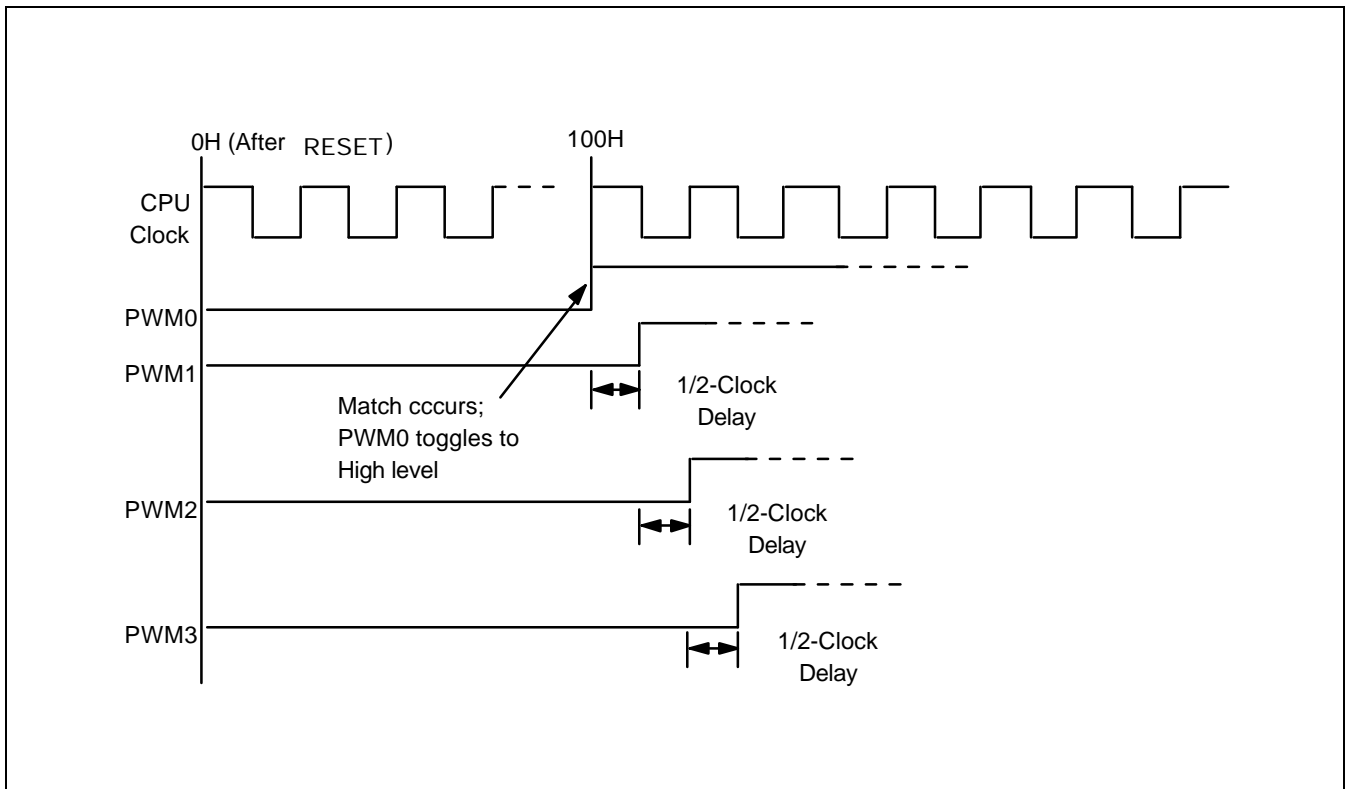


Figure 12-4. PWM Clock to PWM2–PWM5 Output Delays

PWM0–PWM1

The S3C8847 and S3C8849/P8849 pulse width modulation (PWM) module has two 14-bit PWM circuits (PWM0 and PWM1). The 14-bit PWM circuits have the following components:

- 14-bit counter with 3-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 14-bit increasing counter comprised of a lower byte counter and an upper byte counter.

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. PWM0 and PWM1 are read/write addressable.

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

To program the required PWM output, you should 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 should 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 1 (non-divided), 2, 3, 4, 5, 6, 7, or 8.

Because the maximum CPU clock rate for the S3C8847/C8849/P8849 microcontrollers is 8 MHz, the maximum base PWM frequency is 31.25 kHz (8 MHz divided by 256). This assumes a non-divided CPU clock.

Table 12-1. 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 PWM0 basic cycle frame value
	PWM1EX	F7H	6-bit extension ("stretch") value
PWM control register	PWMCON	F8H	PWM0 counter stop/start (resume), and 2-bit prescaler for CPU clock; also contains capture A control settings

PWM0 AND 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, 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 2–7), 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 Table 12-2).

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 and PWM1 output pin. In this way, you can obtain high output resolution at high frequencies.

Table 12-2. PWM Output "Stretch" Values for Extension Registers PWM0EX and PWM1EX

PWM0EX/PWM1EX Bit	"Stretched" Cycle Number
7	1, 3, 5, 7, 9, ..., 55, 57, 59, 61, 63
6	2, 6, 10, 14, ..., 50, 54, 58, 62
5	4, 12, 20, ..., 44, 52, 60
4	8, 24, 40, 56
3	16, 48
2	32
1	Not used
0	Not used

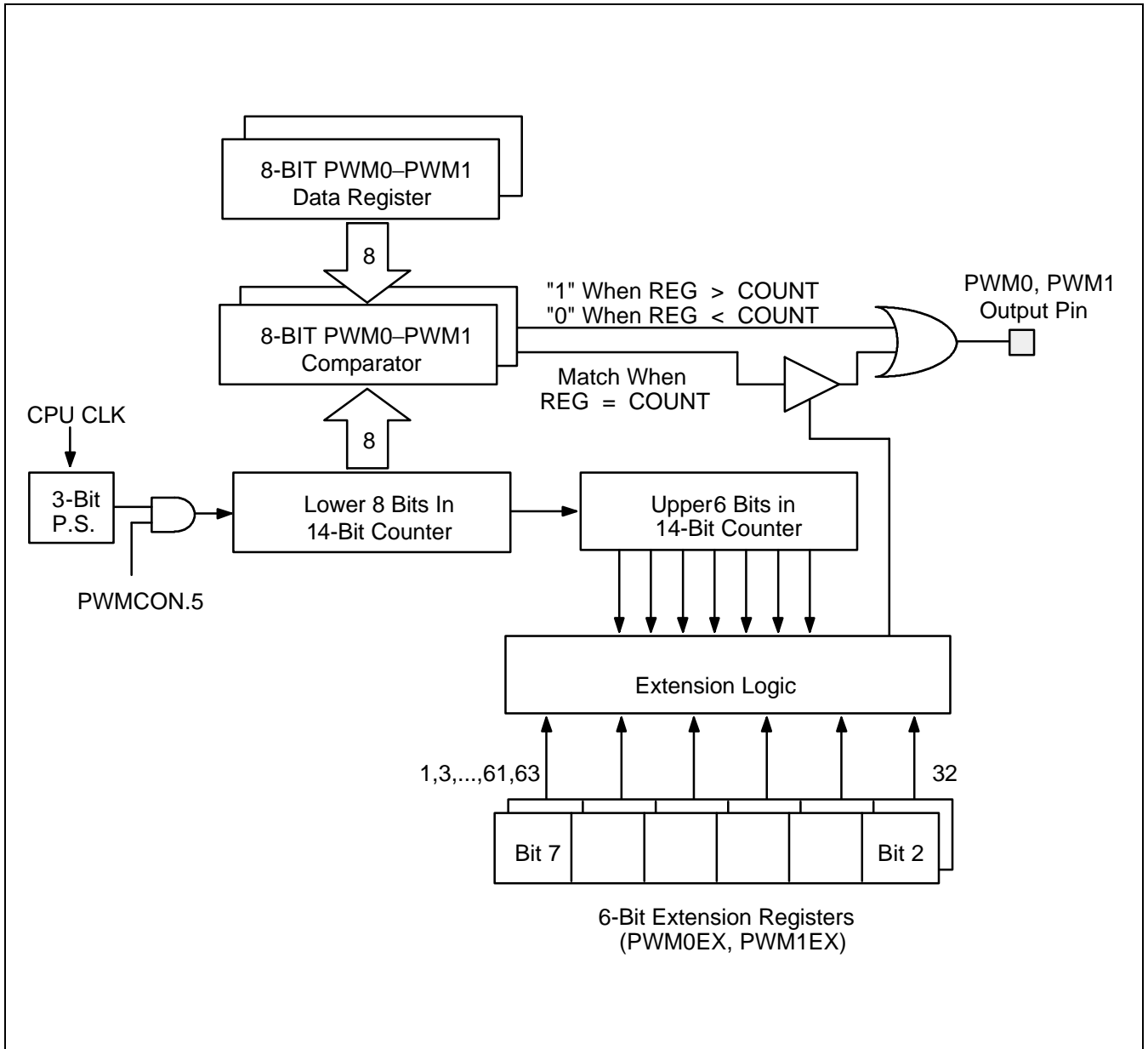


Figure 12-5. Block Diagram for PWM0 and PWM1

PROGRAMMING TIP — Programming PWM0 to Sample Specifications

This example shows how to program the 14-bit pulse-width modulation module, PWM0. The program parameters are as follows:

- The oscillation frequency of the main crystal is 6 MHz
- PWM0 data is in the working register R0
- PWM0EX (PWM0 extension value) is in the 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 $PWM \leftarrow PWM + 12H$
(If an overflow occurs from R0, then $R0 \leftarrow 0FFH$ and $R1 \leftarrow 0FCH$.)
3. If R3.0 = "0", then $PWM \leftarrow PWM - 11H$
(If an underflow occurs from R0, then $R0 \leftarrow 00H$ and $R1 \leftarrow 00H$.)

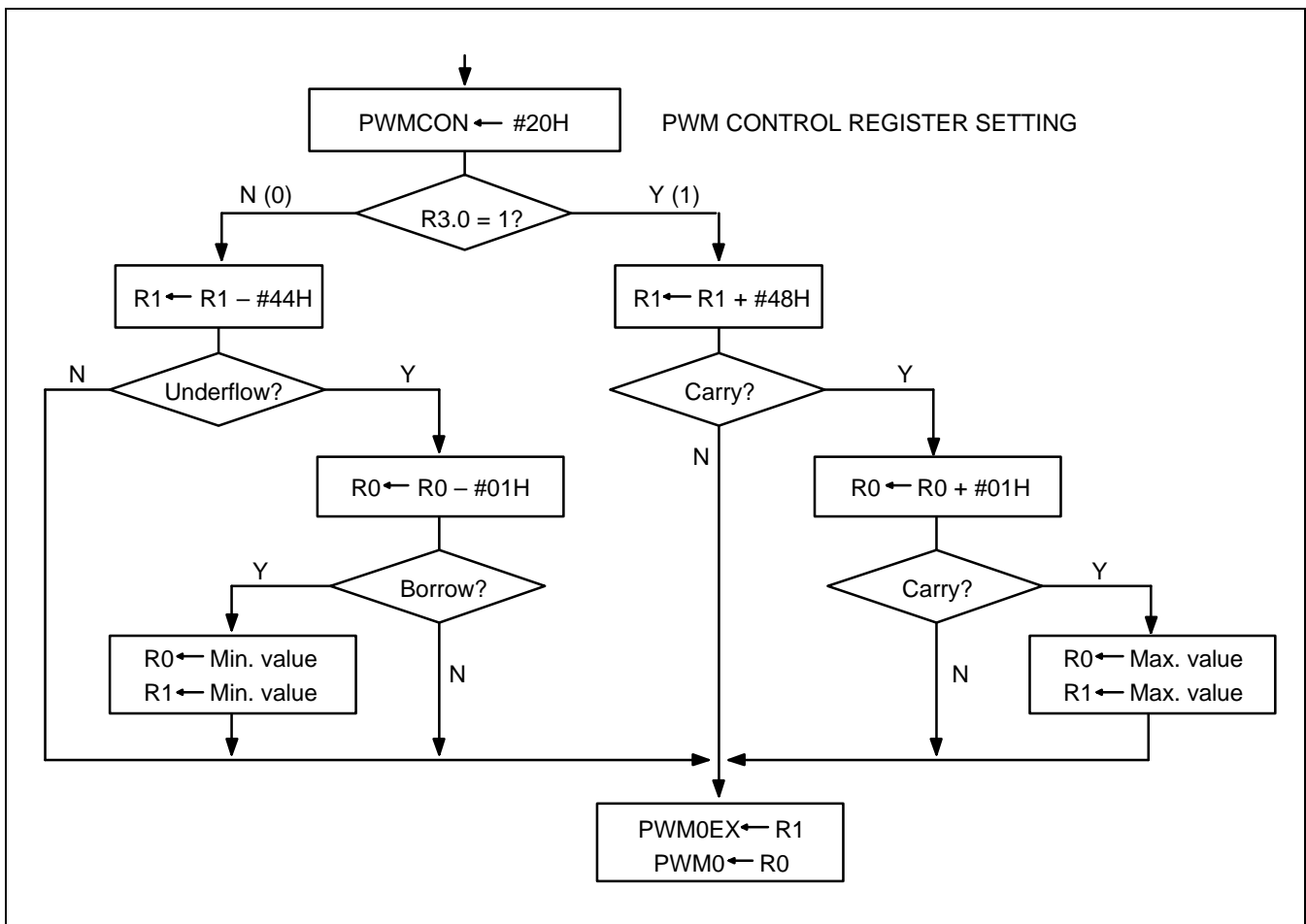


Figure 12-6. Decision Flowchart for PWM0 Programming Tip

 **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 (CAPA/Pin 36)
- 8-bit capture interrupt (IRQ3, vector 02H)

The capture unit captures the upper 8-bit value of the 14-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 in the level 3 (IRQ3) and its vector address is 02H.

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.

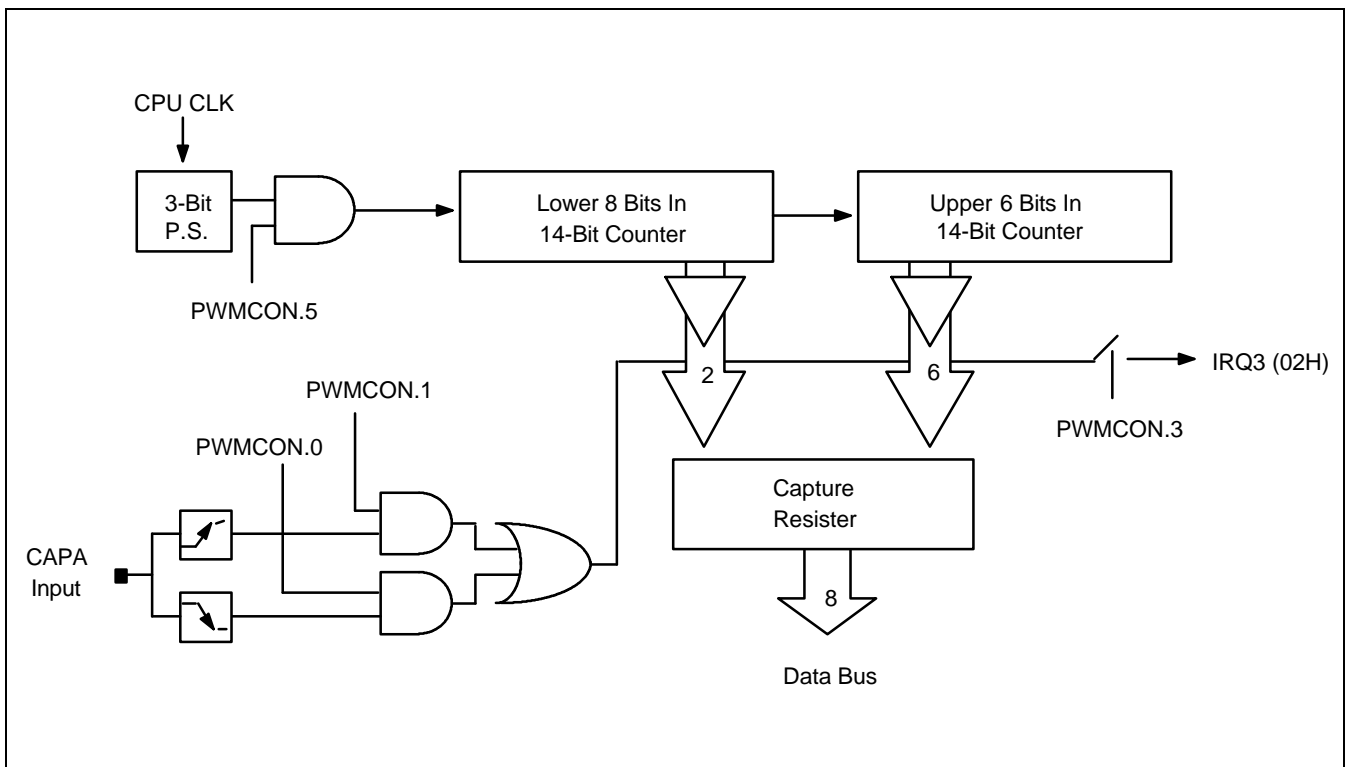


Figure 12-7. Block Diagram for Capture A

PROGRAMMING TIP — Programming the Capture Module to Sample Specifications

This example shows you how to program the S3C8847/C8849/P8849 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 (CAPA) 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 captured 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 $4.35 \text{ ms} < t_H, t_L < 4.6 \text{ ms}$, then set bit zero (LDR) in the register 77H; otherwise clear the zero bit (LDR) in the register 77H.
2. If the interval between two rising signal edges (capture trigger) is $> 30 \text{ ms}$, disregard the capture setting.

Figures 12-4 and 12-5 show decision flowcharts for the sample program.

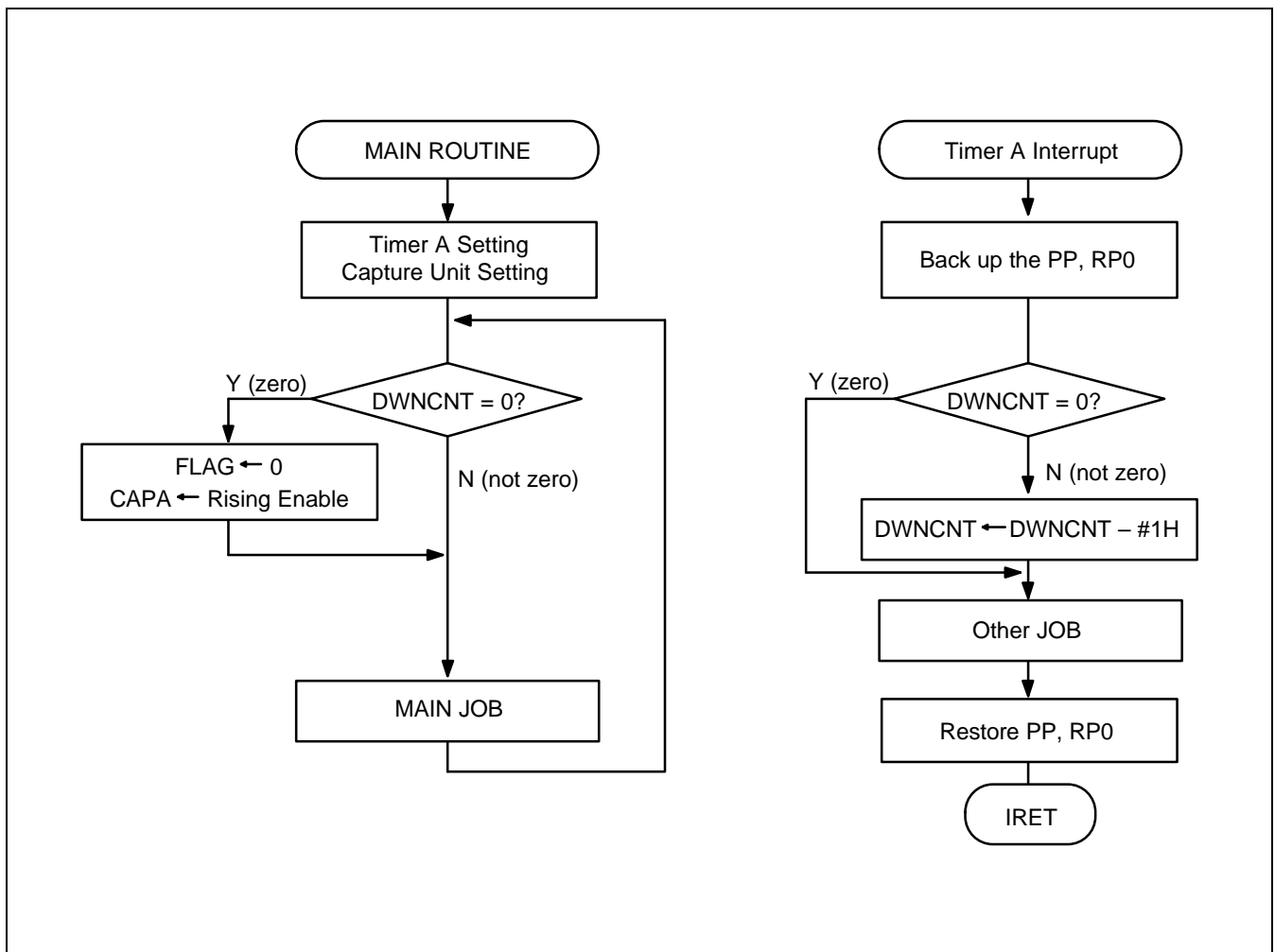


Figure 12-8. Decision Flowchart (Main Routine and Timer A Interrupt)

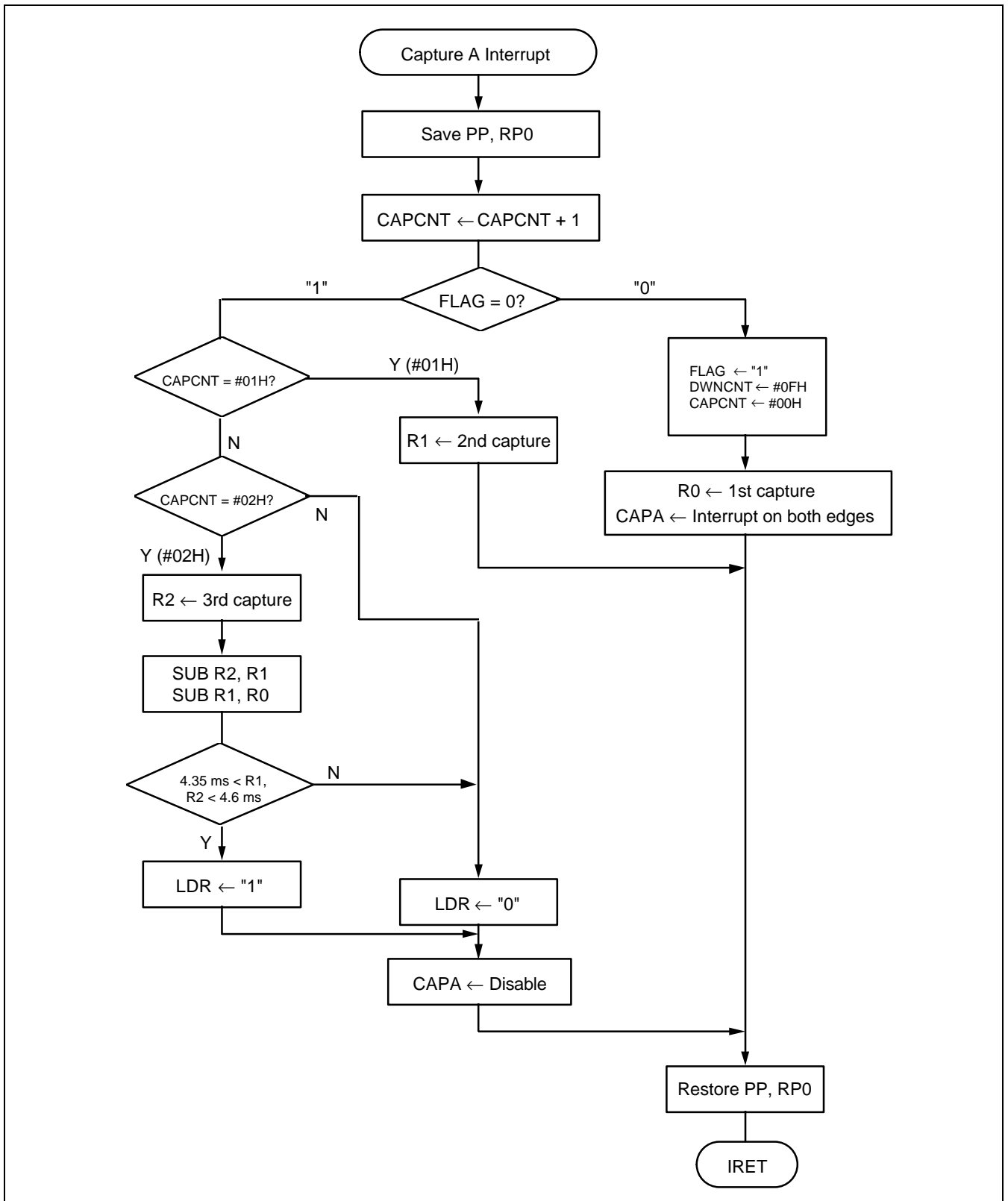


Figure 12-9. Decision Flowchart for Capture A Interrupt

 **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       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:
      •
      •
      •
      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

```

(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
          LD      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

```

(Continued on next page)

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

```

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
          .
          .
          .

```

13 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 252 locations and supports a set of 384 characters. (Two characters are reserved: 00H for the blank function and 17FH for the test pattern.) There are eight display colors.

PATTERN GENERATION SOFTWARE

For application development using the S3C8847/C8849/P8849 microcontrollers, Samsung provides OSD pattern generation software (OSDFONT.exe). You can customize standard OSD patterns contained in this file.

Table 13-1. OSD Function Block Summary

OSD Function Block	Function Description
Video RAM ^(note)	Located in register page 1, the video RAM contains 252 "word" lines. Each line is 13 bits long. Each 13-bit RAM address stores an 9-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–FBH can be accessed using Indirect Register or Indexed addressing mode only.
Character ROM	The character ROM contains an 18-dot × 16-dot matrix data for 384 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 17FH is for a factory test pattern.
Output control logic	Output control logic receives input from the Character ROM, 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, blank, and H/T) are output from the OSD block at pins 22–25, 21.

NOTE: The video RAM can be cleared only by "LD" instruction.

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 252 word lines. Each line is 13 bits long. Of these 13 bits, eight are character display codes (bits 0–8). Bit 12 is the character halftone or character background color display control bit and bits 9–11 are used to determine the red, green, and blue components of the character color.

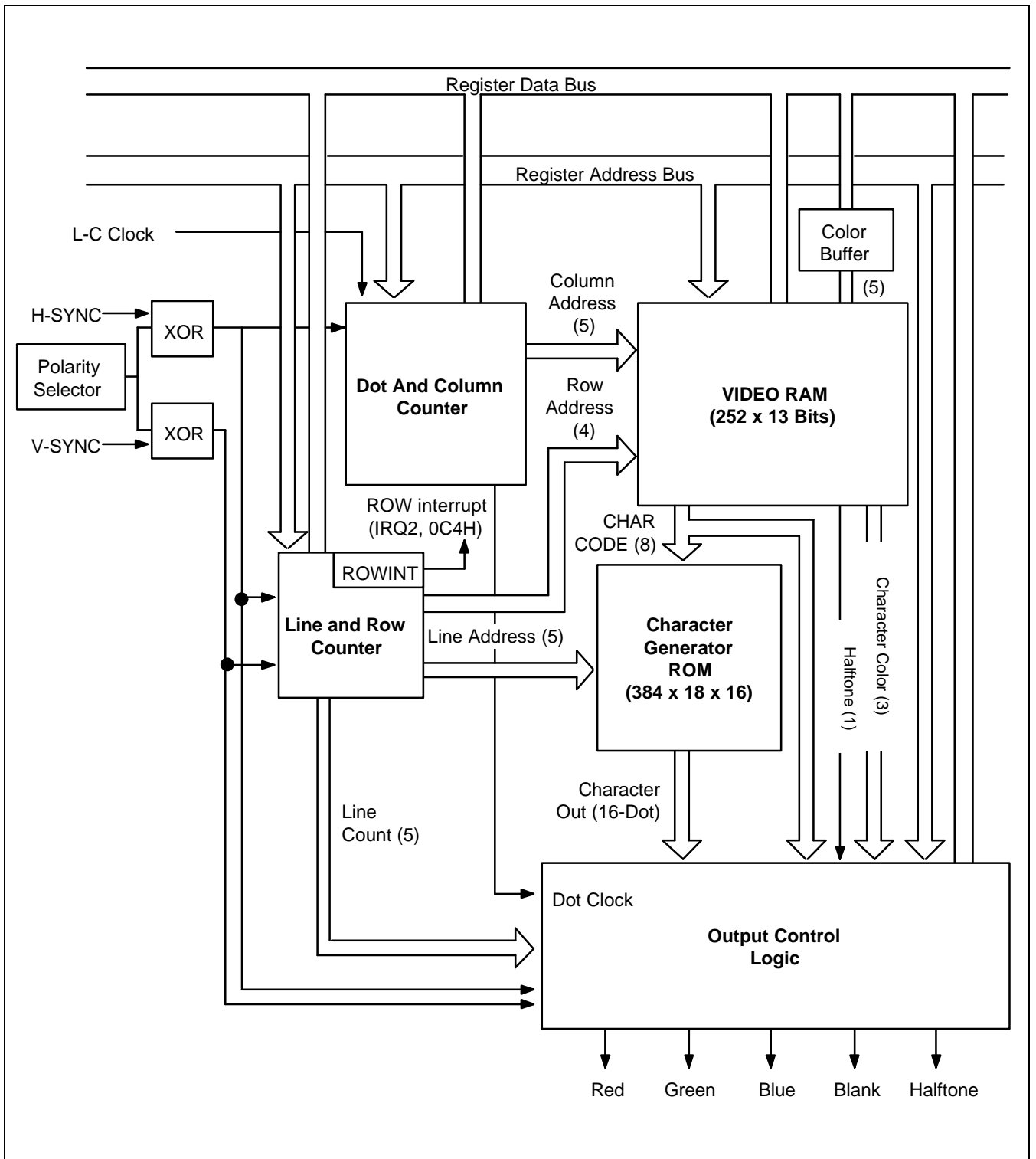


Figure 13-1. On-Screen Display Function Block Diagram

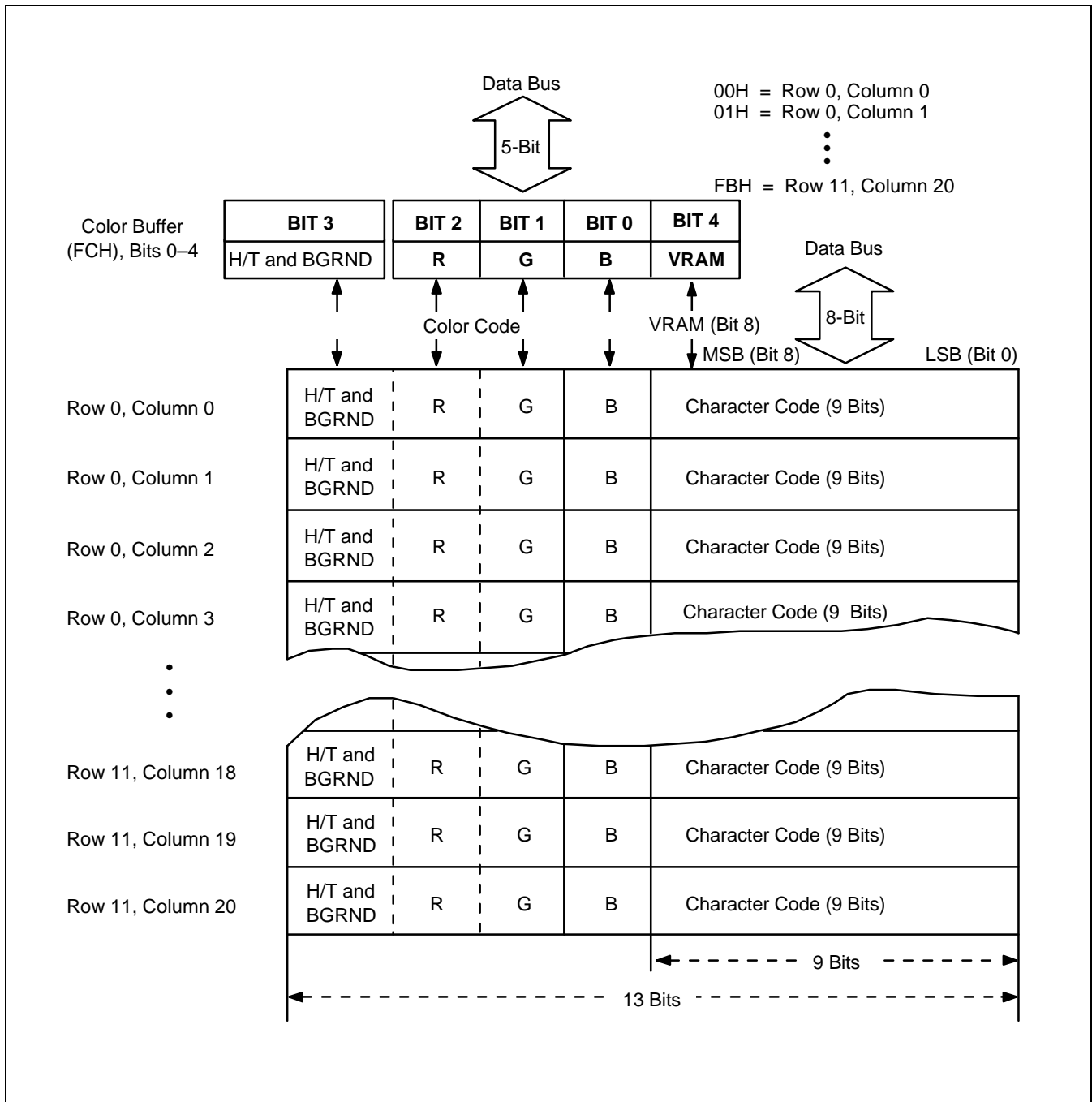


Figure 13-2. On-Screen Display Video RAM Data Organization

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
VSBCON	V-sync blank time control 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 select halftone or background color for character displays, choose the polarity for H-sync and V-sync signal synchronization, and as OSD ROW counter which is read-only (bit4–bit7).

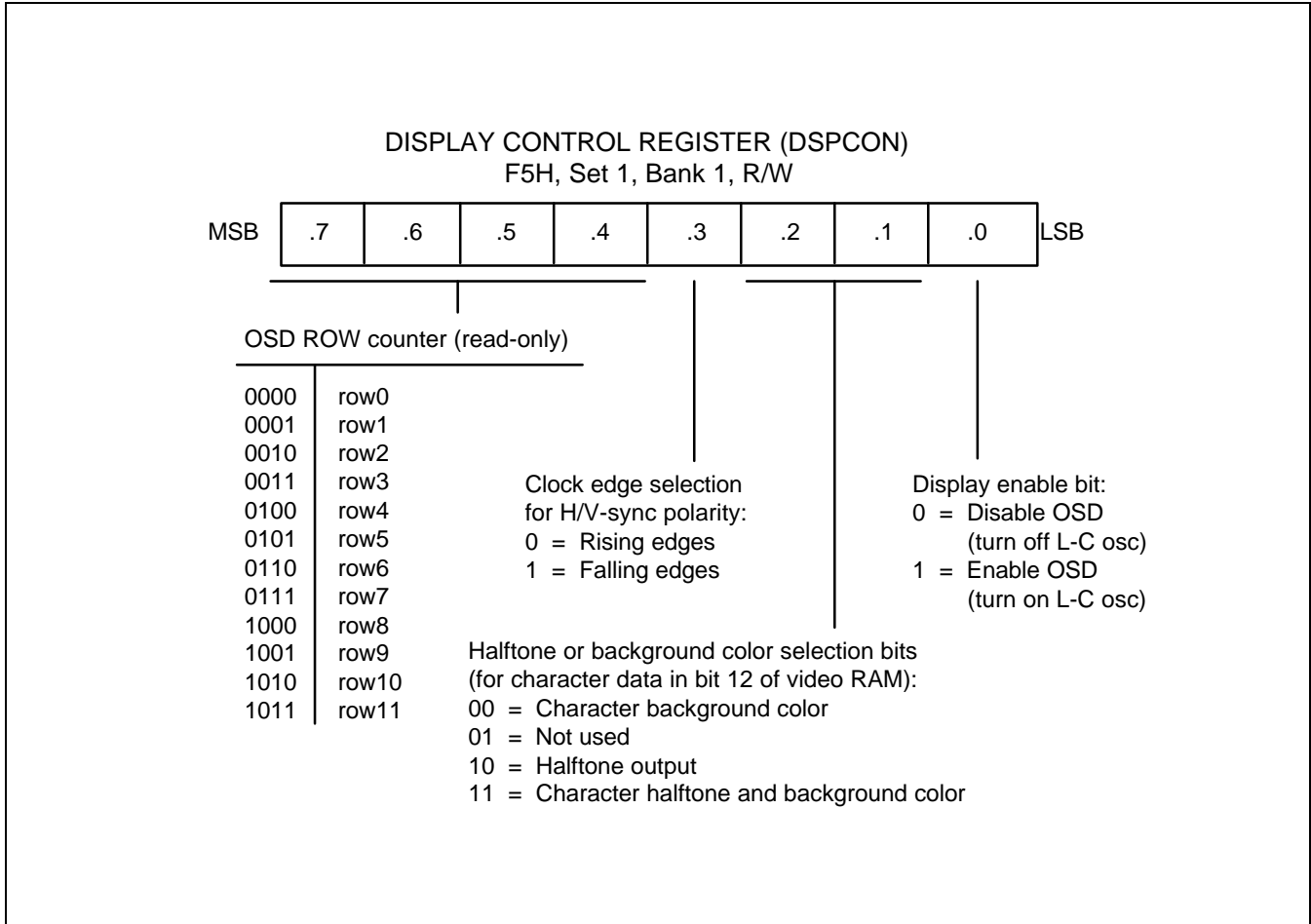


Figure 13-3. OSD Display Control Register (DSPCON)

NOTE

Refer to the PROGRAMMING TIP – Row Interrupt Function of 13-24.

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.

The DSPCON.0 settings determine the on/off condition of L-C oscillation, synchronized with Vsync input. And the OSD output can be turned on or off in OSD row units when L-C oscillation is on. At the point the value of DSPCON.0 is changed from "1" to "0" in the middle of a frame, OSD is disabled (OSD output is off). In this condition, L-C oscillation becomes off at the next Vsync input. When the value is shifted from "0" to "1", OSD is enabled (OSD output is on) and L-C oscillation returns to "on" at the following Vsync input.

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 and DSPCON.1 let you select a halftone or background color display for individual characters. (Which characters are displayed as halftones, or with character background color, or with character halftone and background color, depends on the bit12 settings in the character video RAM data.) When DSPCON.2–.1 = "00", the character background color option is selected; when DSPCON.2–.1 = "10", the character halftone function is selected; when DSPCON.2–.1 = "11", the character halftone and backgroundcolor option are selected; but DSPCON.2–.1 = "01" is not used.

ROW Counter Function

DSPCON.4–DSPCON.7 to the OSD ROW read data. OSD ROW counter indicates the OSD ROW currently displayed. One ROW comprises one character (18 lines) and inter-ROW space (ROWCON.2–.0). The Row counter value for the first ROW after a Vsync input is set to "0".

Character Background Color and Halftone Function Mode

When DSPCON.1 = "0", the character background color or halftone function mode is selected; when DSPCON.1 = "1", the character background and halftone function mode is selected.

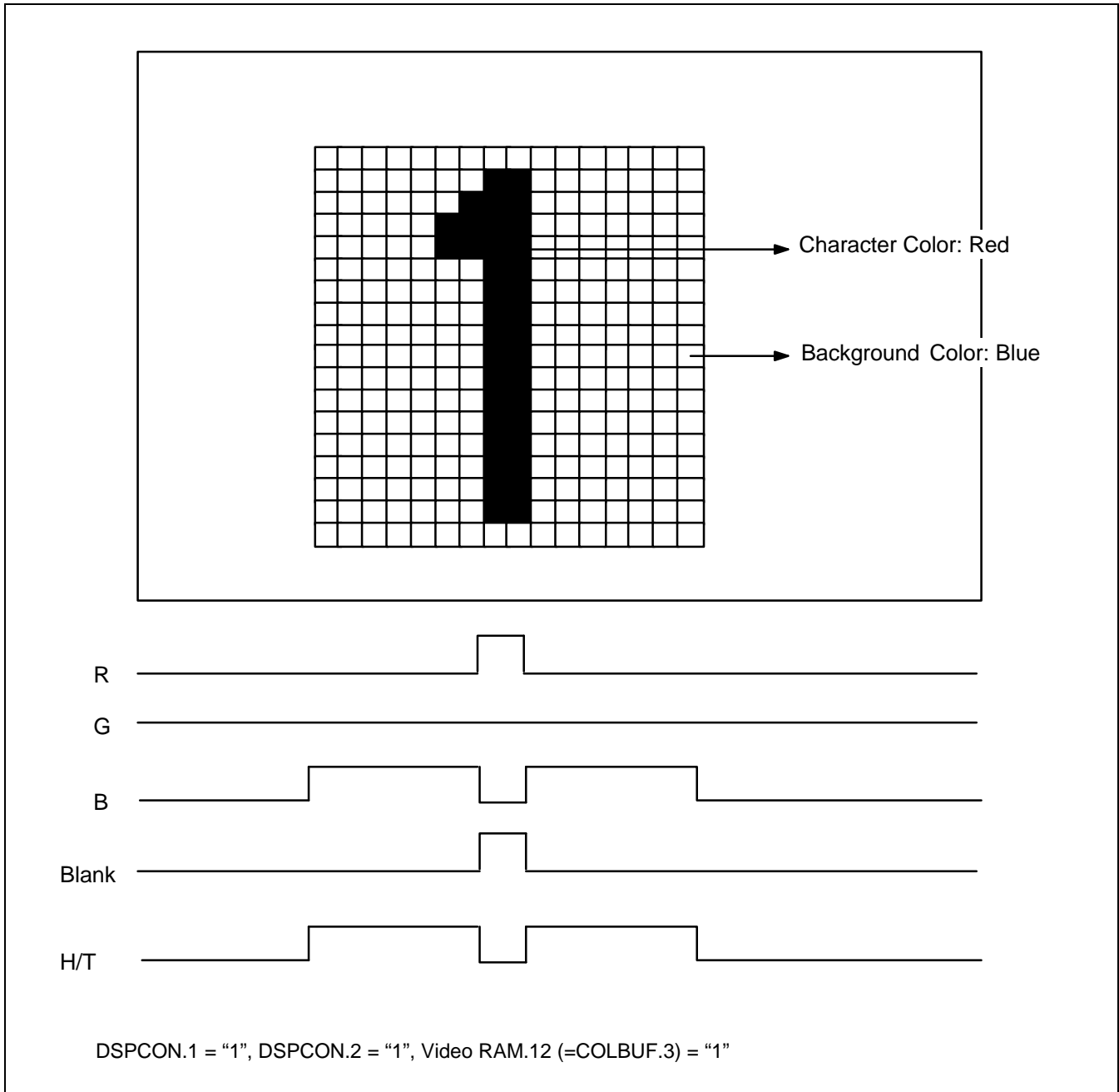


Figure 13-4. Halftone or Character Background Signal Output

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 (0–11) for the character fade function (see Figure 13-5).

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: $\times 1$, $\times 2$, $\times 3$, and $\times 4$. Size ' $\times 1$ ' is the smallest and ' $\times 4$ ' is the largest. For example, to display a ' $\times 1$ ' (horizontal) by ' $\times 1$ ' (vertical) size character, you should clear CHACON.4–CHACON.7 to "0". To display a ' $\times 4$ ' by ' $\times 4$ ' size character, you should set bits 4–7 to '1111B'.

You can also combine different vertical and horizontal size selections to produce flattened or elongated characters (see Figure 13-6).

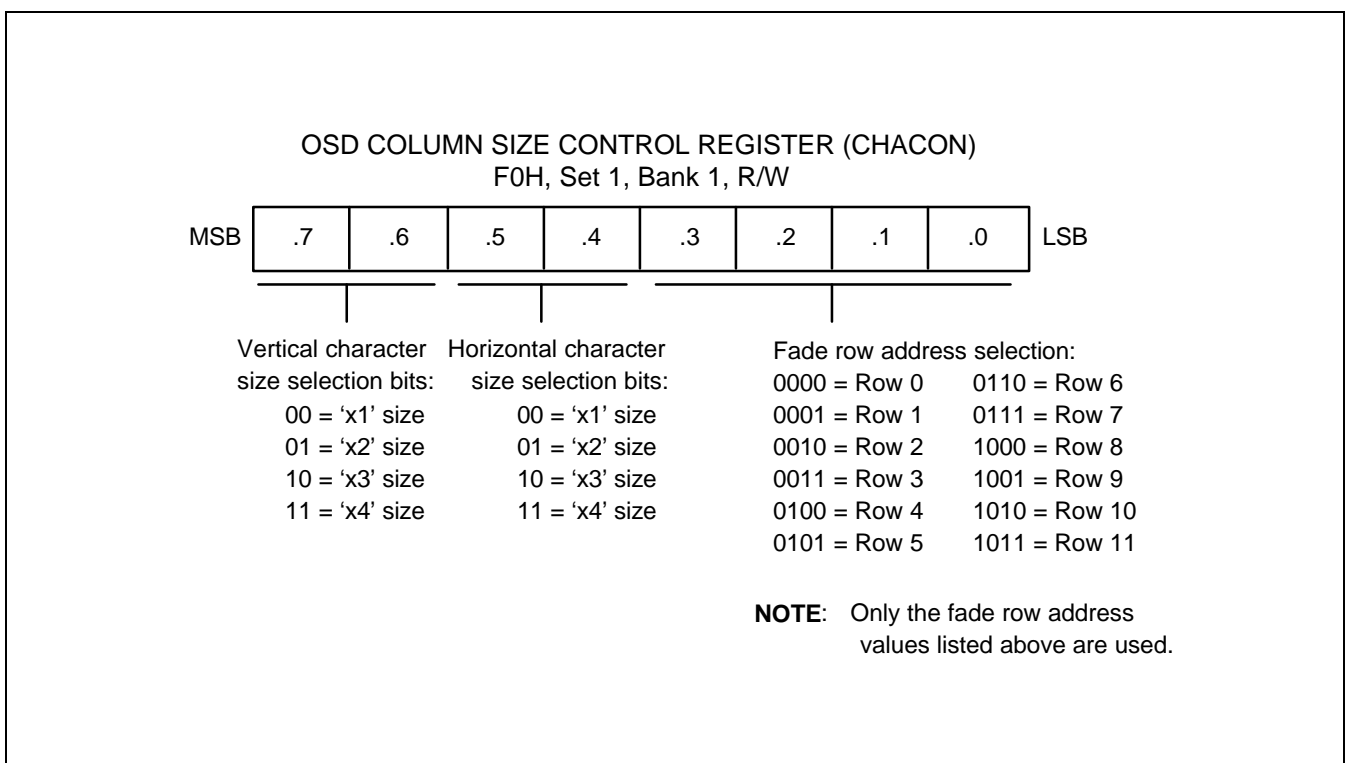


Figure 13-5. OSD Character Size Control Register (CHACON)

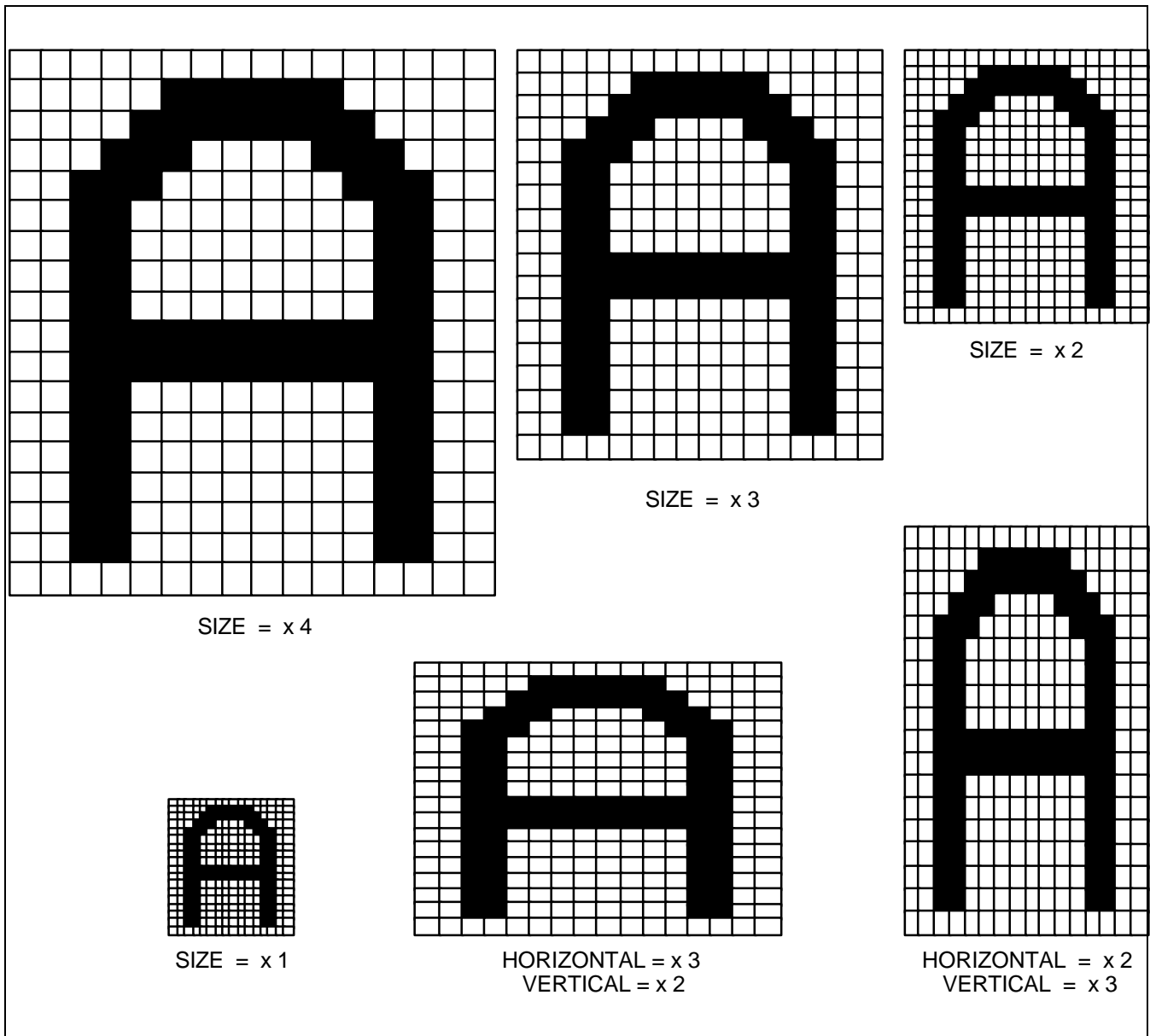


Figure 13-6. OSD Character Sizing Dimensions

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 "appears". 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 twelve rows (0–11) 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 should 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.

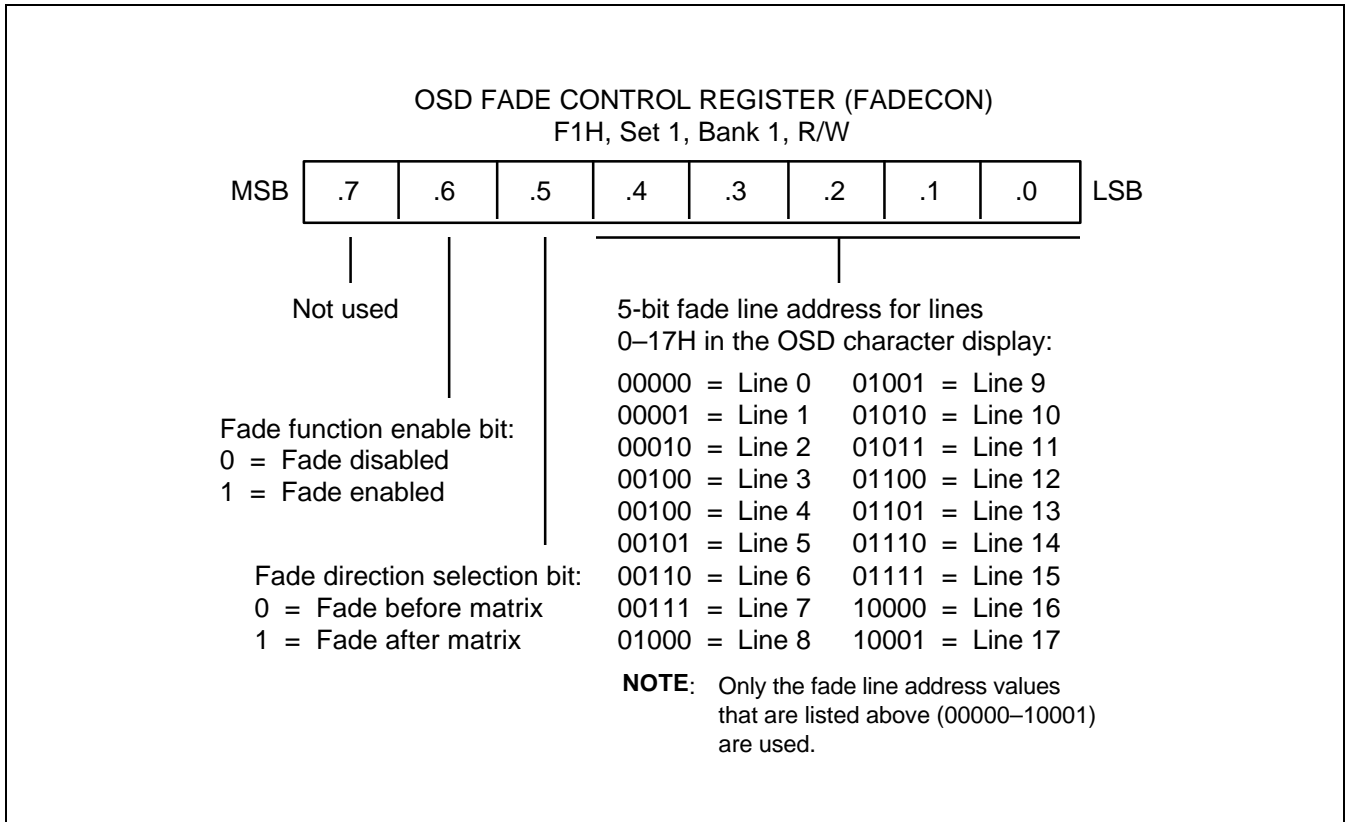


Figure 13-7. OSD Fade Control Register (FADECON)

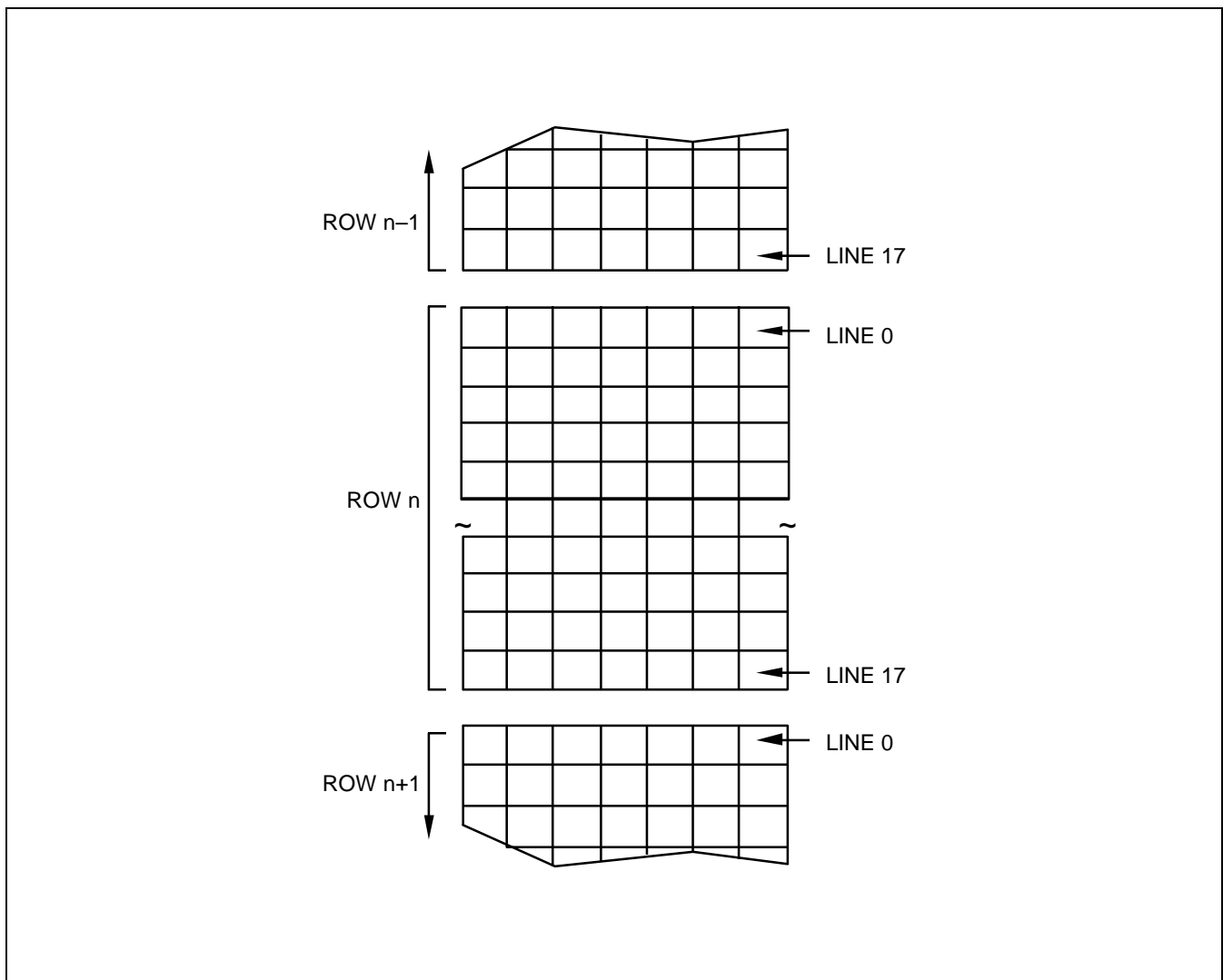


Figure 13-8. Line and Row Addressing Conventions

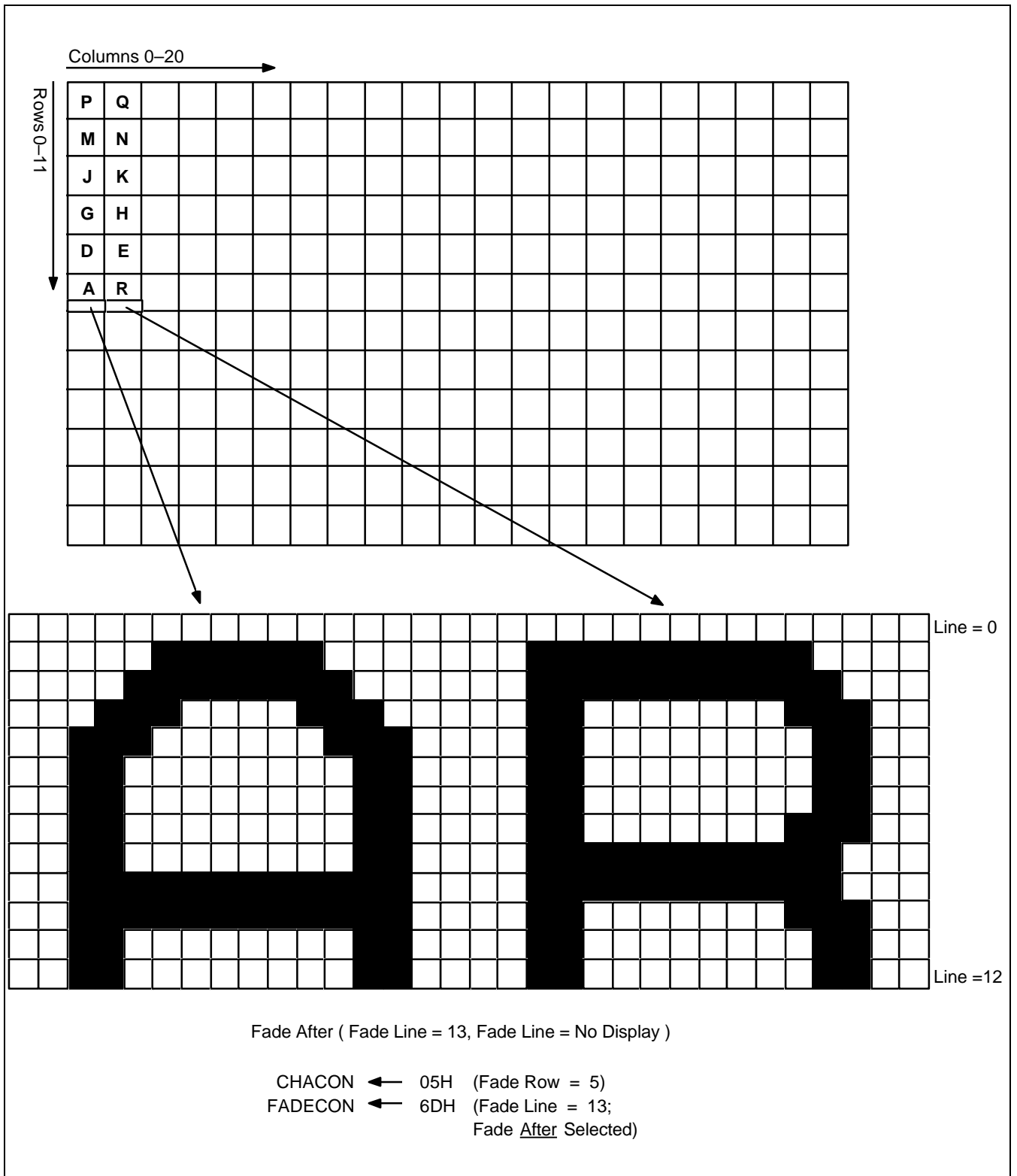


Figure 13-9. OSD Fade Function Example: Fade After

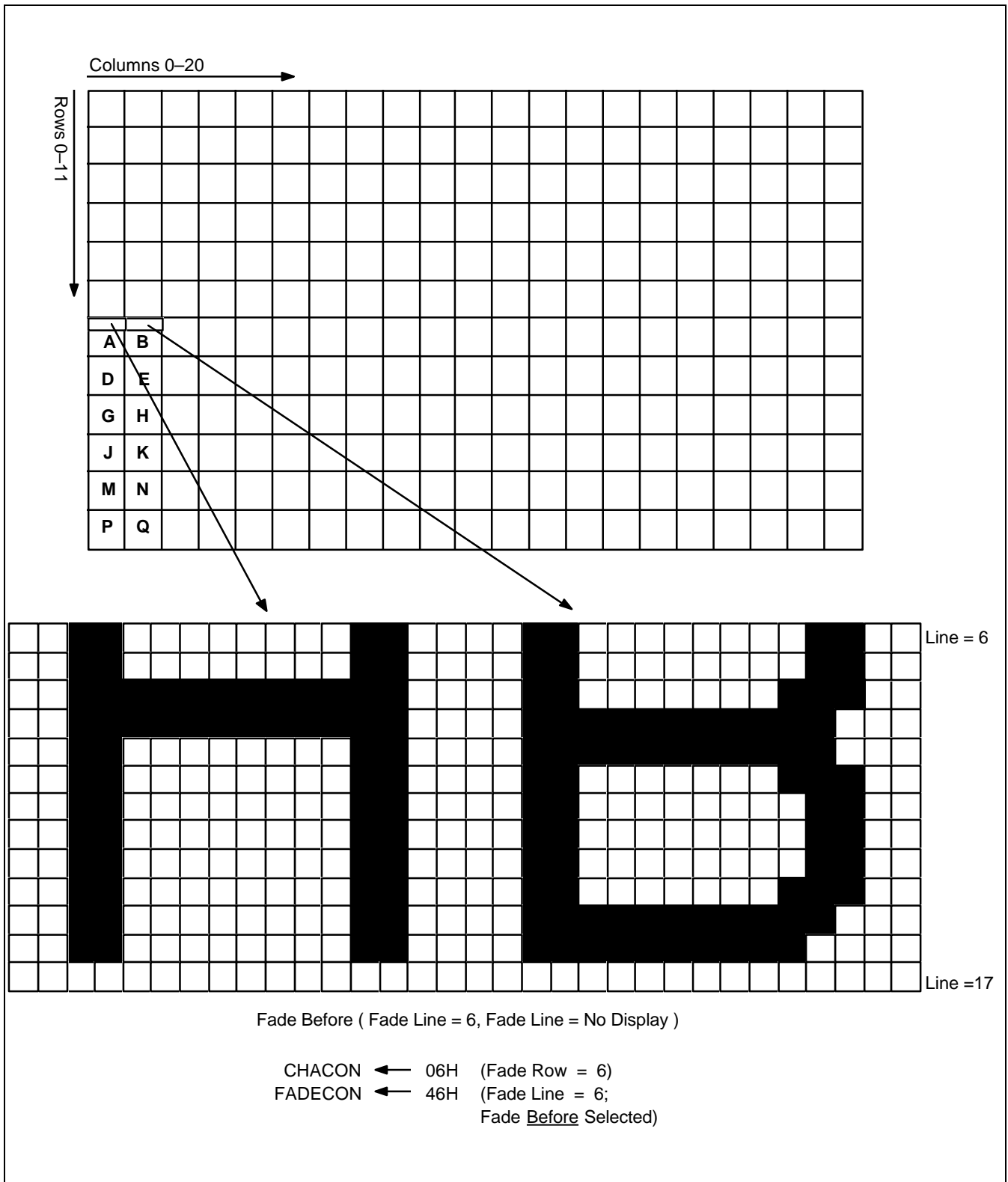


Figure 13-10. OSD Fade Function Example: Fade Before

DISPLAY POSITION CONTROL

The on-screen display has 252 character display positions. There are 21 horizontal columns and 12 vertical rows. Positions can be numbered sequentially from 0–251 (decimal) or from 0–FB (hexadecimal), as shown in Figures 13-11 and 13-12. To control display position, you can adjust the top and left margins and the inter-column and inter-row spacing between characters on the screen.

		COLUMNS 0-20 →																				
																					DECIMAL	
ROWS 0-11 ↓	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	93	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	201	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	240	241	242	243	244	245	246	247	248	249	250	251	

Figure 13-11. 252-Byte On-Screen Character Display Map (Decimal)

		COLUMNS 0-20 →																				
																					HEXADECIMAL	
ROWS 0-11 ↓	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	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	

Figure 13-12. 252-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) to 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.

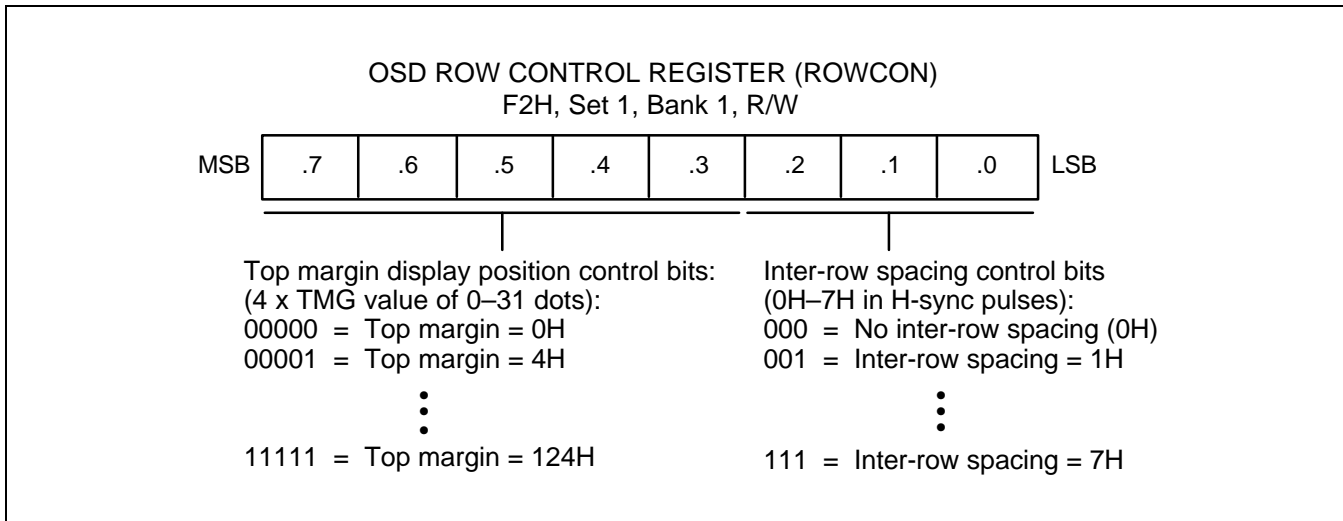


Figure 13-13. OSD Row Control Register (ROWCON)

COLUMN CONTROL REGISTER (CLMCON)

The column control register, CLMCON, controls the left margin and inter-column spacing. *Left margin* is the distance to the character display from the left edge of the display frame. *Inter-column spacing* is the distance (0–7 dots) between 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.

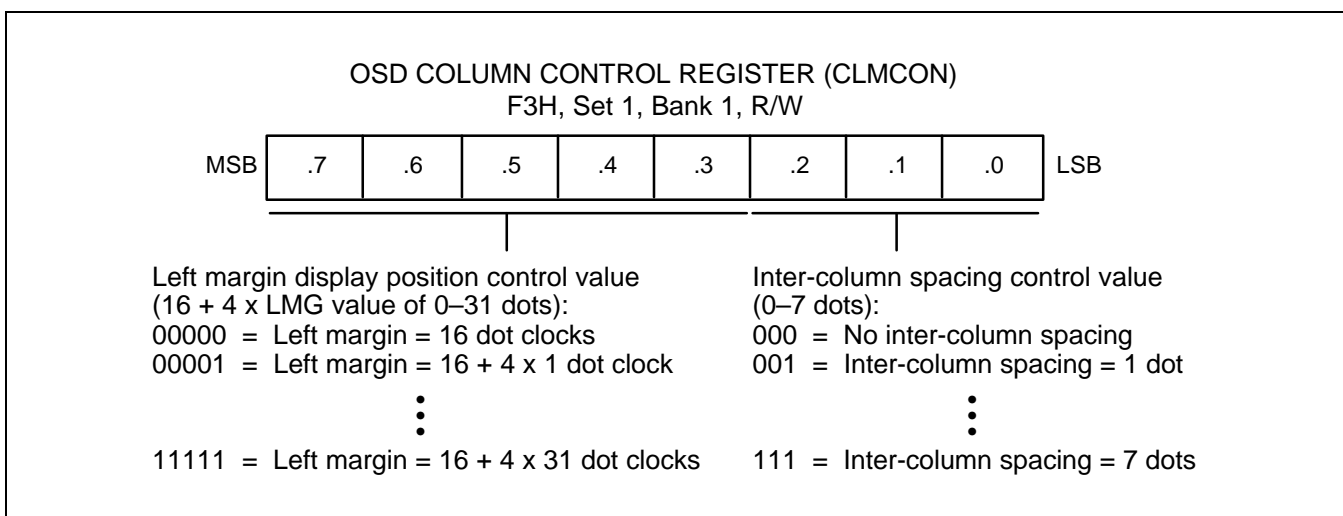


Figure 13-14. OSD Column Control Register (CLMCON)

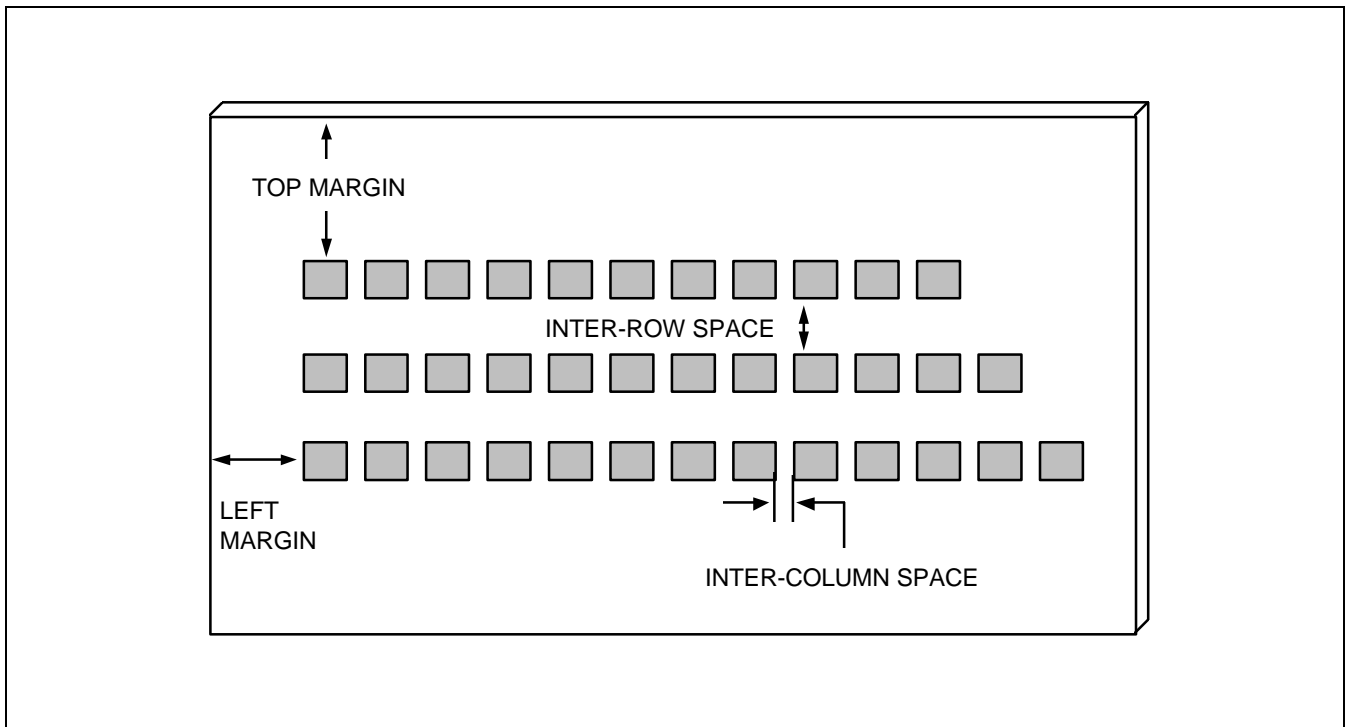


Figure 13-15. OSD Display Formatting and Spacing Conventions

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 can set the top margin at $4 \times$ 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 can set the left margin at $16 + 4 \times$ 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 16 dots.

- Top margin = $4 \times$ (top margin register value) H
- Left margin = $16 + 4 \times$ (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 5-bit value in the OSD video RAM. You can modify the *character color selection bits* only by addressing the OSD color buffer register, COLBUF (FCH, set 1, bank 1). The color selection bits are COLBUF.2 (red), COLBUF.1 (green), COLBUF.0 (blue) and COLBUF.3 (H/T and BGRND). These four bits comprise the RGB value (bits 9, 10, 11) and H/T and BGRND enable bit (bit 12) 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 9-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.

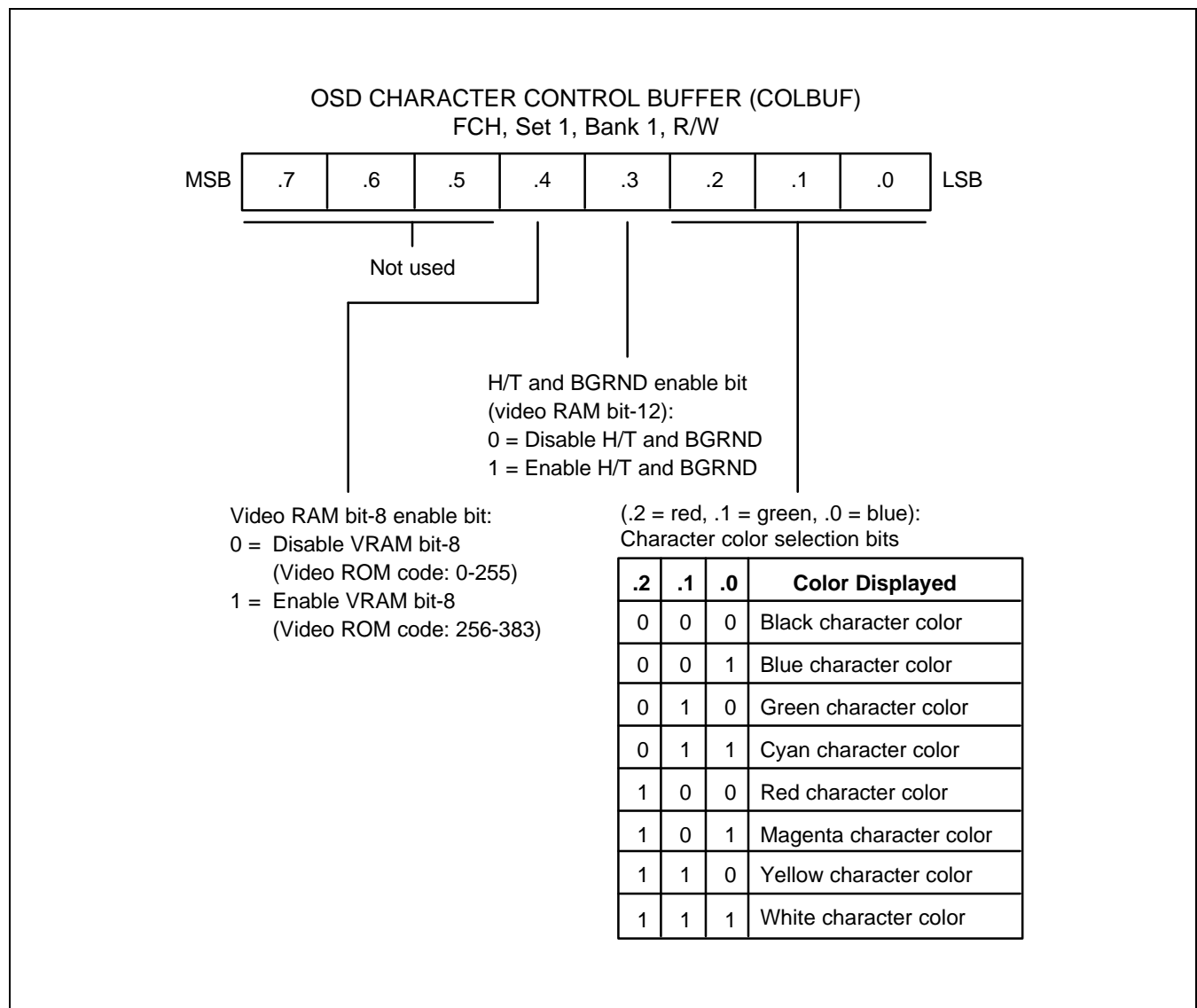


Figure 13-16. OSD Character Color Buffer Register (COLBUF)

BACKGROUND COLOR CONTROL

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

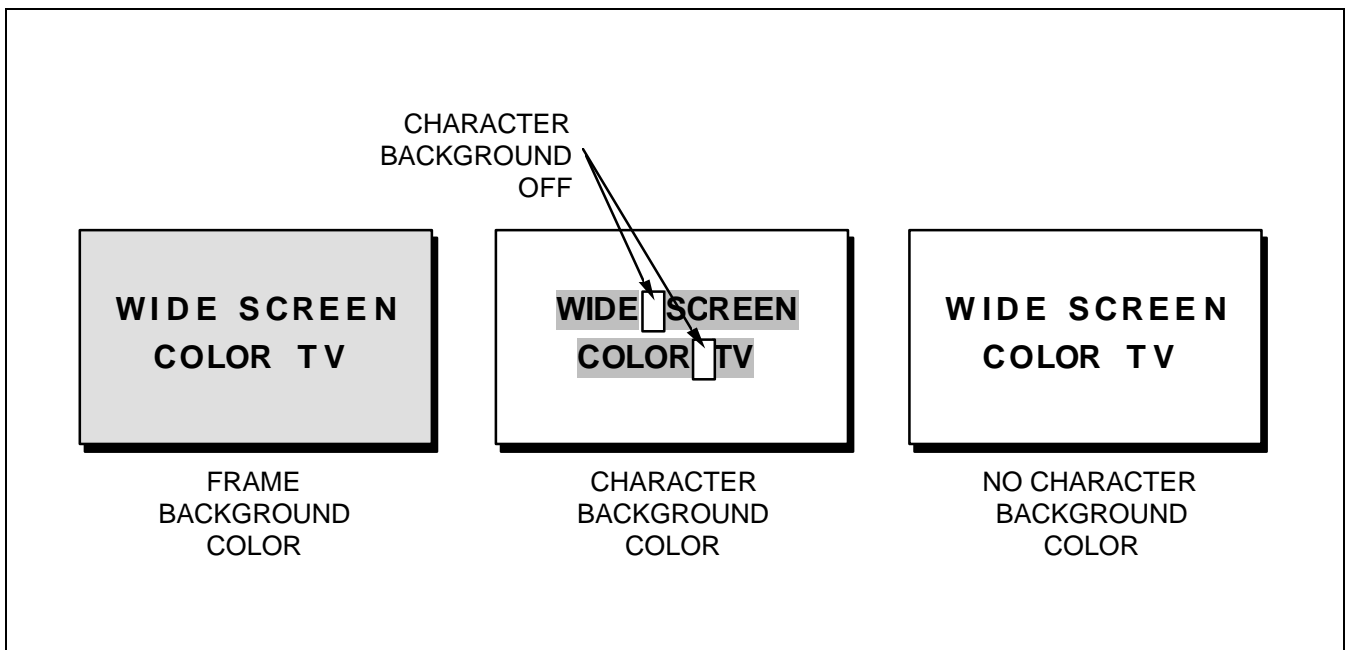


Figure 13-17. Background Color Display Conventions

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 to "0". When COLCON.7 or COLCON.3 is "0", the background display is disabled and no color appears.

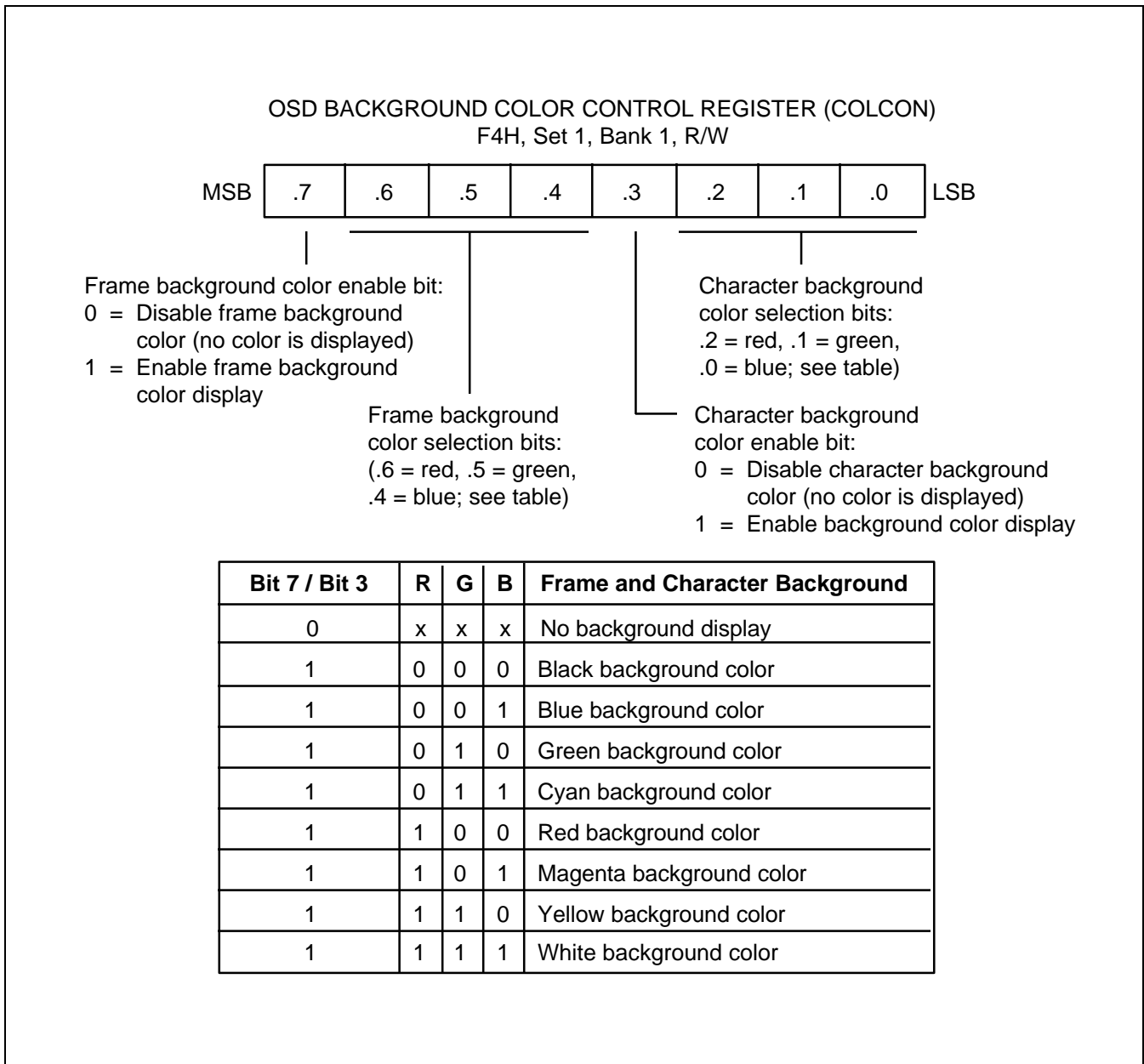


Figure 13-18. OSD Background Color Control Register (COLCON)

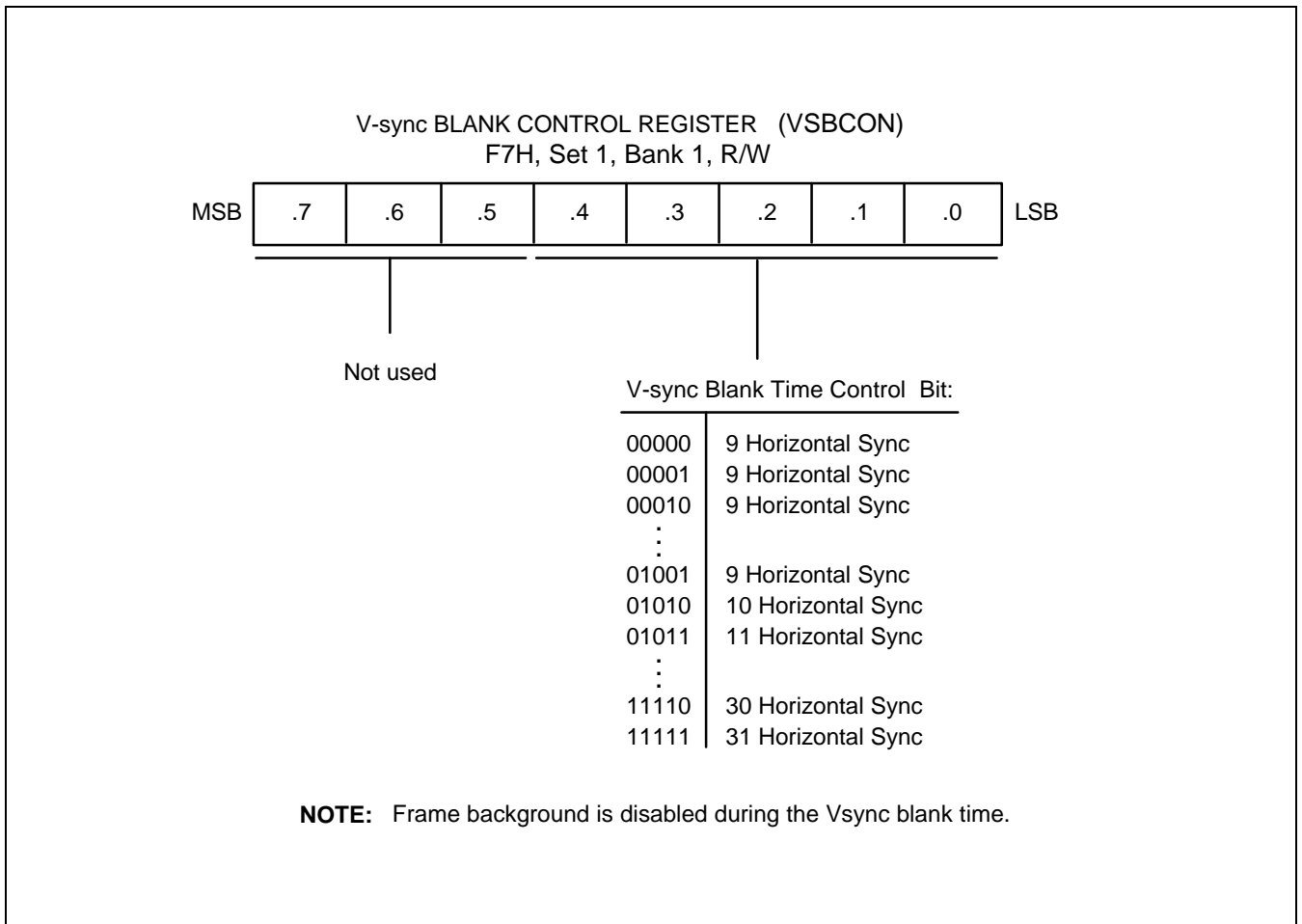


Figure 13-19. V-sync Blank Control Register (VSBCON)

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 12 in the video RAM) or for frame periods (regardless of the bit 12 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 are used for OSD Row interrupt function.

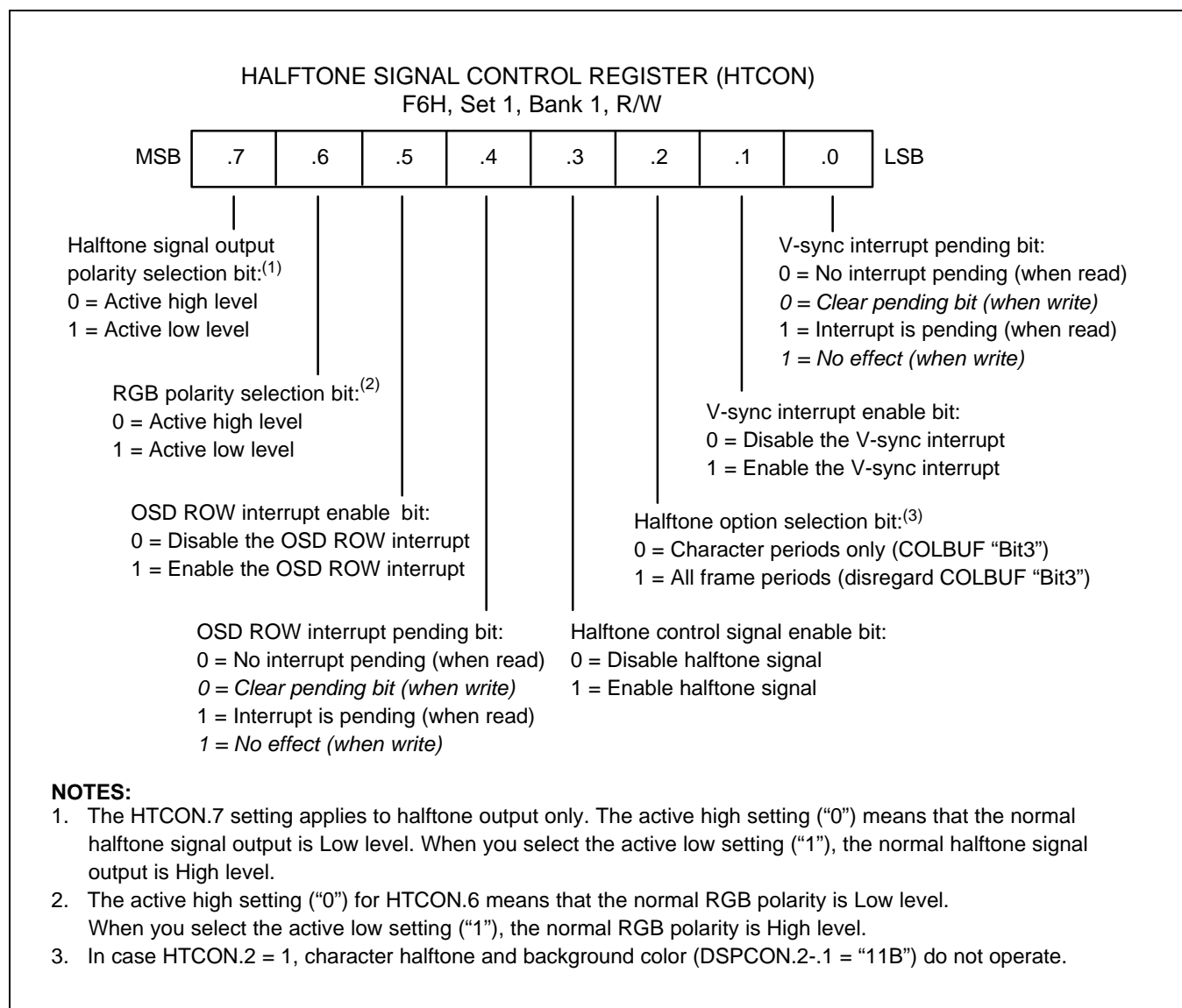


Figure 13-20. Halftone Signal Control Register (HTCON)

OSD ROW Interrupt Control

The S3C8847/C8849/P8849 has a total of 12 OSD display rows. When enabled, an OSD ROW interrupt occurs in the first line of each row. Up to 12 OSD ROW interrupts can be generated, while this number can be reduced according to different settings in top margin (ROWCON.7–.3), inter row space (ROWCON.2–.0), vertical character size (CHACON.7–.6), and Vsync blank time (VSBCON). The ROW counter of DSPCON.7–.4 informs the order of an OSD ROW interrupt occurring within a frame. The first ROW interrupt (DSPCON.7–.4 = “0000B”) occurs at the point the Vsync blank is completed or the top margin starts. An OSD ROW interrupt is generated at the beginning of a ROW (for ROW1 through ROW11), except for ROW0.

OSD ROW interrupt allow different controls to each ROW. If the OSD control register is adjusted in the first OSD ROW interrupt (DSPCON.7–.4 = 0000B) service routine, the new value is applied from ROW1. A change in the 12 th OSD ROW interrupt service routine affects the rows from ROW0.

NOTE: OSD output enable/disable (DSPCON.0) settings are immediately applied. Top margin (ROWCON.7–.3) and VSBCON are applied in accordance with Vsync input signals.

Halftone Option Selection

In character periods only (HTCON.2 = “0”), the character specified in COLBUF.3 may have the halftone function according to the condition of DSPCON.2–.1.

In all frame periods (HTCON.2 = “1”), the entire section can have the function, regardless of the COLBUF.3 condition. In this situation, however, the character halftone and background color (DSPCON.2–.1 = 11B) function is not available.

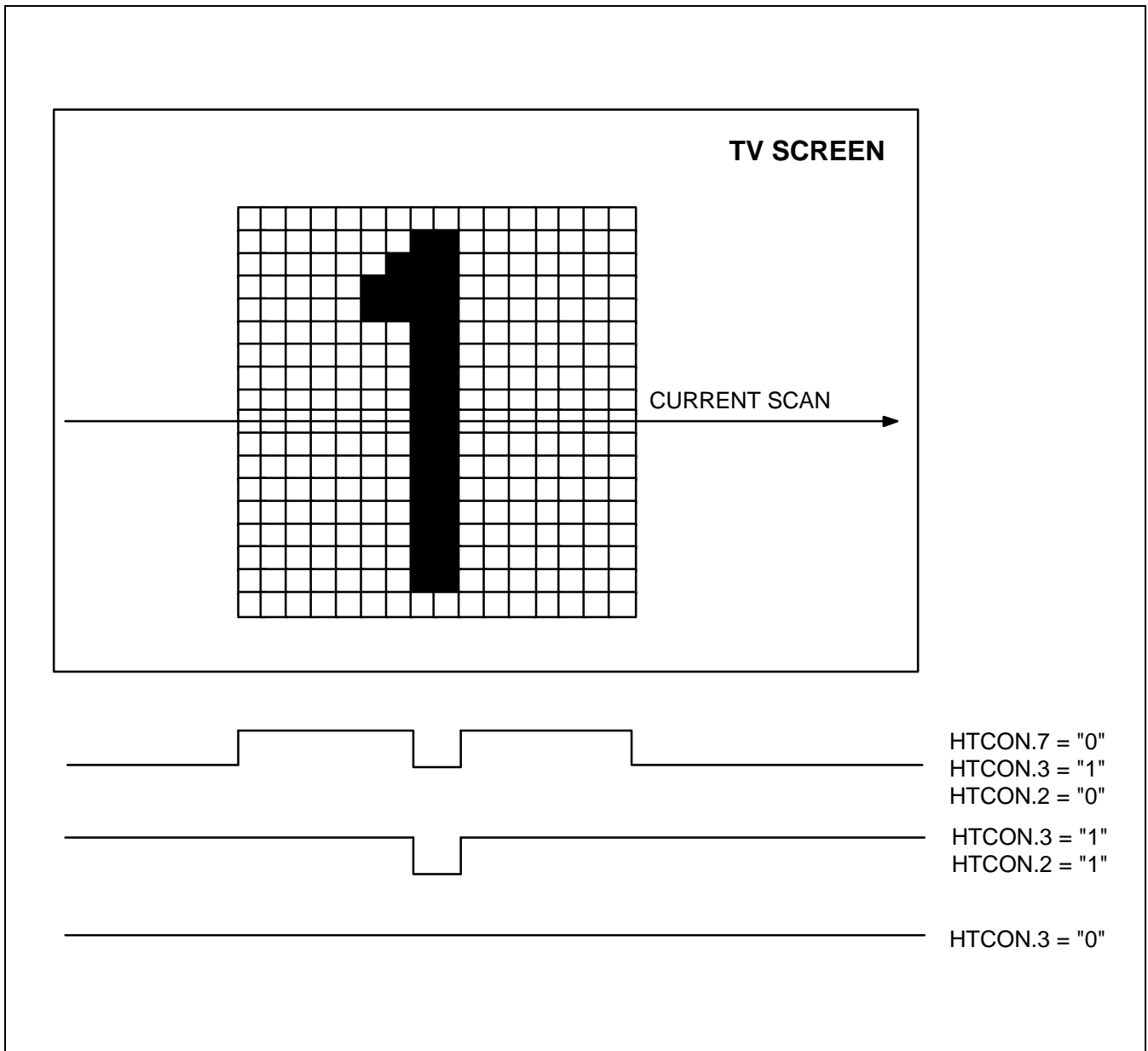


Figure 13-21. Half-tone Control Signal Output Options

PROGRAMMING TIP — Row Interrupt Function

This example shows the effect of the control register setting excluding the HTCON.5,4,1,0 and DSPCON 3,0 occurs in the next row. The sample program should meet the following specifications:

1. The character size of the row 2 must be double-sized (×2).
2. The character size of the other rows must be normal (×1).

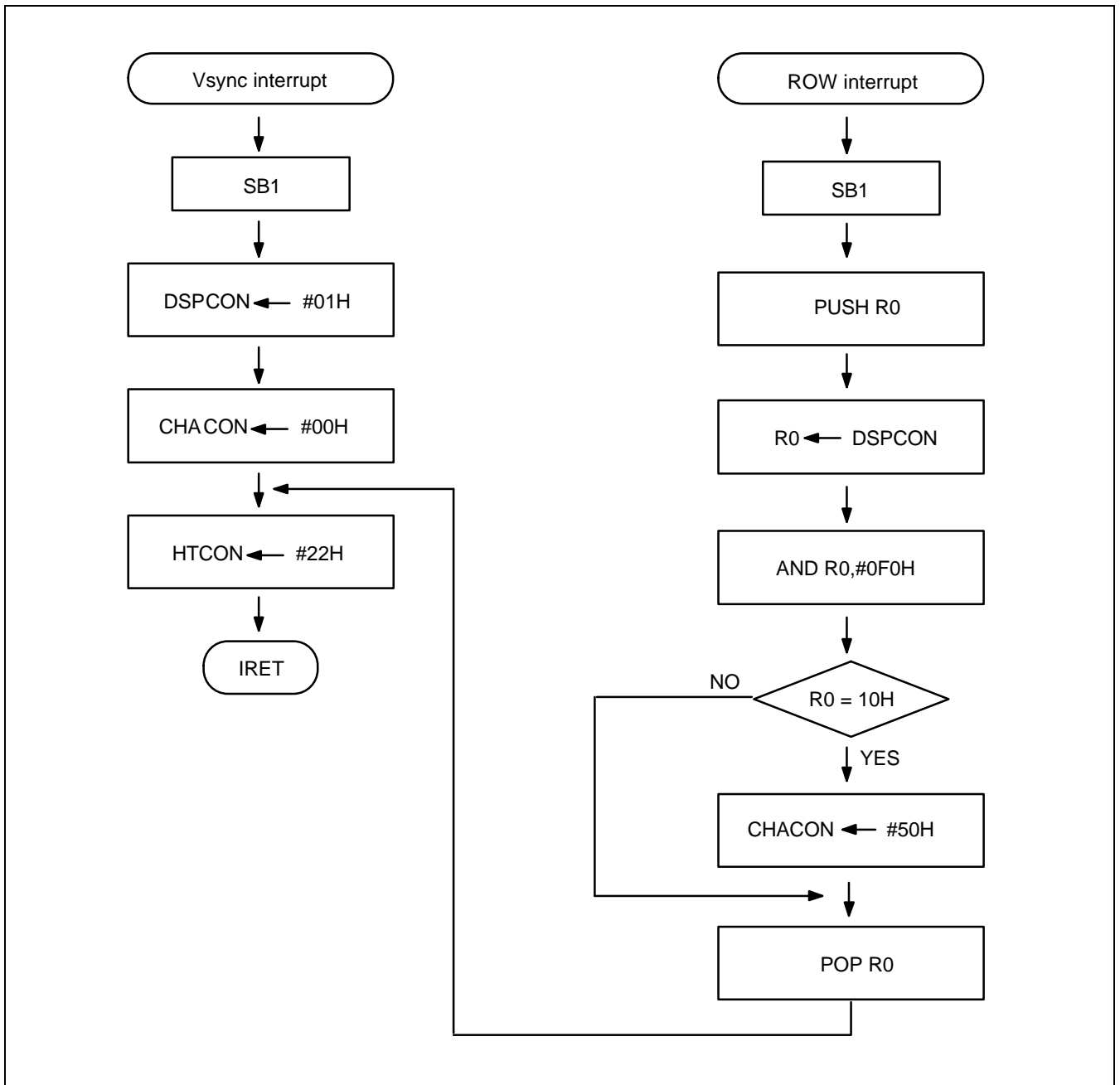


Figure 13-22. Decision Flowchart for Row Interrupt Function Programming Tip

 **PROGRAMMING TIP — Row Interrupt Function**

```

Vsync_int:
    SB1                ; Select bank 1
    LD      DSPCON,#01H ; OSD on, H/V sync rising edge
    LD      CHACON,#01H ; Character size

Interrupt_end:
    LD      HTCON, #22H ; Pending bit clear
    IRET

Row_int:
    SB1                ; Select bank 1
    PUSH   R0          ; Stack ← R0
    LD     R0,DSPCON   ; R0 ← DSPCON data
    AND   R0,#0F0H    ; 11110000b, bit0-bit3 clear
    CP    R0,#10H     ; Row interrupt?
    JR    NE, No_Char_Change
    LD     CHACON,#50H ; Double size character

No_char_change:
    POP   R0
    JR    t, Interrupt_end
  
```

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 0FBH.

```

      •
      •
      •
      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–0FBH
      INC     R0           ;
      CP      R0,#0FBH   ;
      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'.

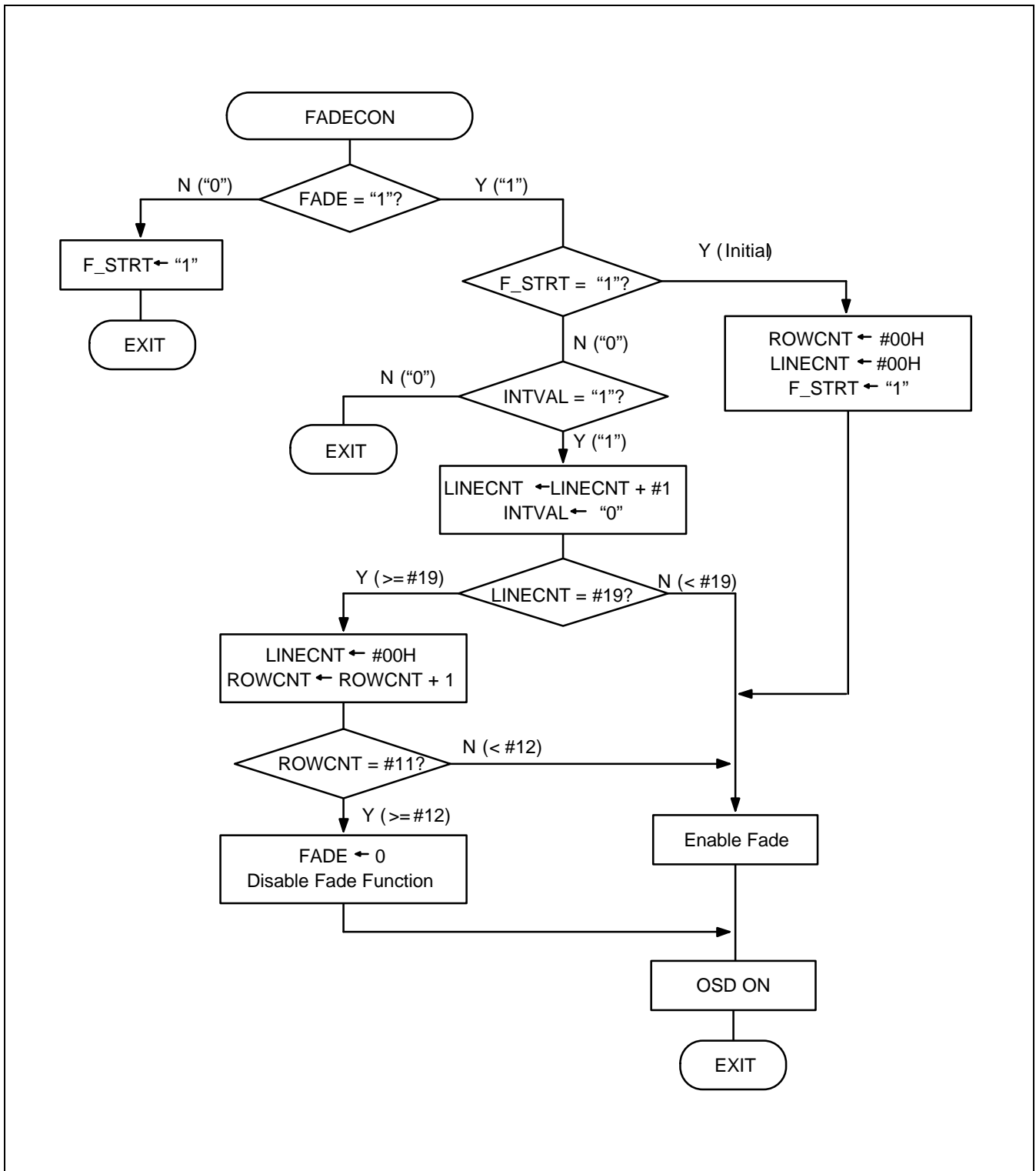


Figure 13-23. Decision Flowchart for Fade Function Programming Tip

 **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,#0DH   ; Row counter < 11?
      JP        ULT,FAD2       ; If row ≤ 12, then jump to FAD2
      LD        R1,#0F1H      ; If row > 12, 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

```

(Continued on next page)

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

```

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
          .
          .
          .

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 the 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 + BG ; 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,#0FH      ; Mask out bit 7 through bit 3 of R2
      LD           COLBUF,R2    ; Color buffer ← complementary color of the character
                                   ; in address 78H
      LD           DSPCON,#0F9H ; OSD module on; negative sync trigger selected
      •
      •
      •

```

(Continued on next page)

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

```

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,#09H         ; Enable OSD; select negative sync trigger
            LD      R1,@R0              ; Video RAM zero address
            INC     R0
            CP      R0,#0FBH            ; Video RAM end?
            JP      UGT,end_halftone
            tm      COLBUF,#08H         ; Check COLBUF.3
            JR      Z,loop_halftone
            LD      HTCON,#0AH          ; Enable halftone
                                           ; Enable V-sync interrupt
            LD      DSPCON,#0DH         ; 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, 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 $\times 3$ and vertical $\times 2$.
2. Enable character background color to the complementary color of the character code in address 0EFH of the video RAM.
3. Enable the frame background; select the color cyan.
4. Set top margin to 16H, inter-row spacing to 1H, left margin to 24 dots, and 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  $\times 3$ , vertical  $\times 2$  for character size
LD      FADECON,#00H ; Disable the fade function
LD      ROWCON,#21H ; Top margin  $\leftarrow$  16H, inter-row space  $\leftarrow$  1H
LD      CLMCON,#1BH ; Left margin  $\leftarrow$  24 dots, inter-column space  $\leftarrow$  3 dots
LD      R3,COLBUF   ; R3  $\leftarrow$  color of the character in address 0EFH
COM     R3          ; R3  $\leftarrow$  not R3
AND     R3,#07H     ; Mask out bit 7 through bit 3 of R3
OR      R3,#0B8H    ; R3  $\leftarrow$  cyan frame background color
LD      COLBUF,R3   ; Enable character and frame background color
LD      DSPCON,#09H ; Falling edge sync trigger, 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 (F7C, 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.

14 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 S3C8847/C8849/P8849 perform 4-bit conversions for one input channel at a time. You select the channel by writing the appropriate 2-bit value to ADCON.5 and ADCON.4. 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.

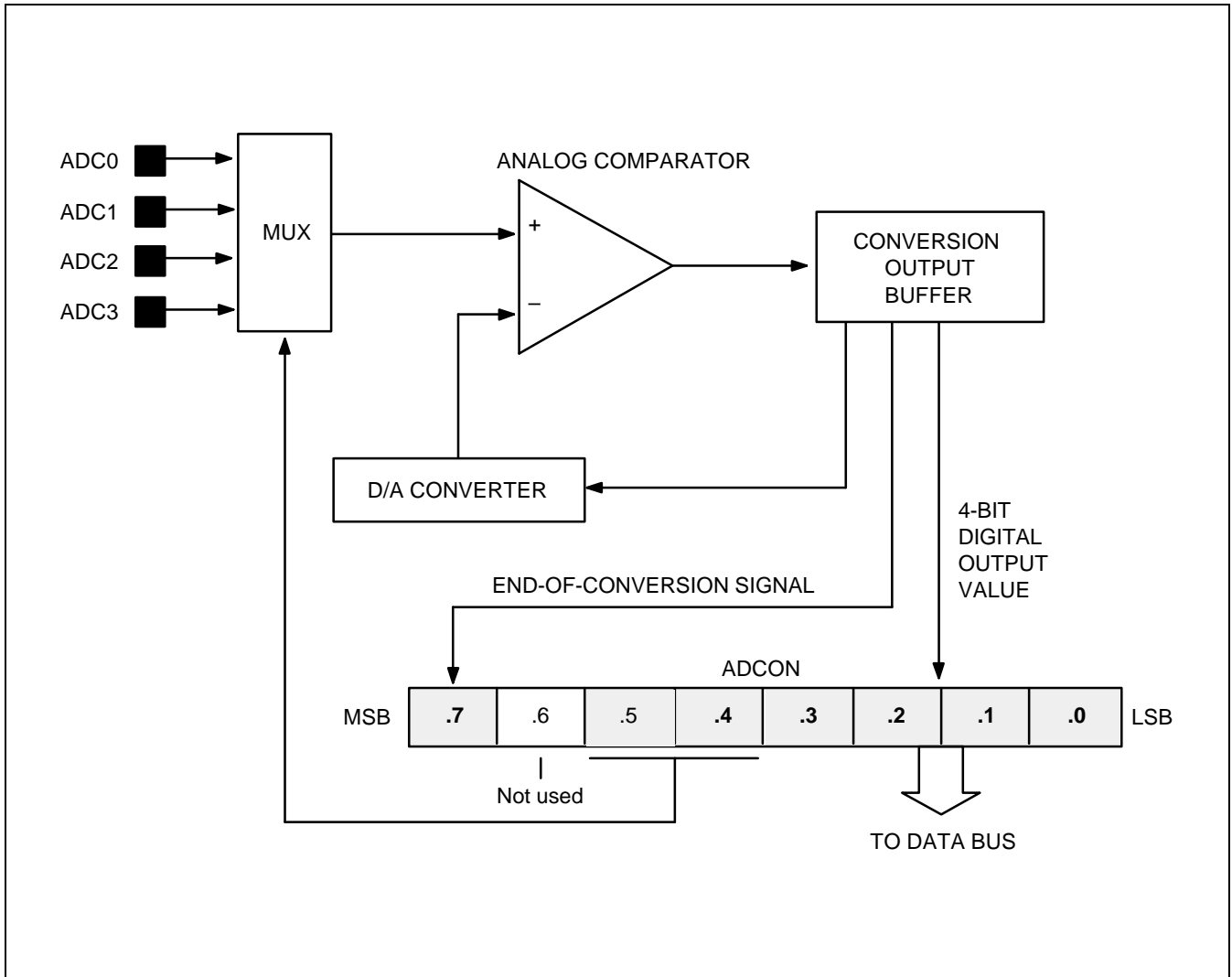


Figure 14-1. A/D Converter Functional Block Diagram

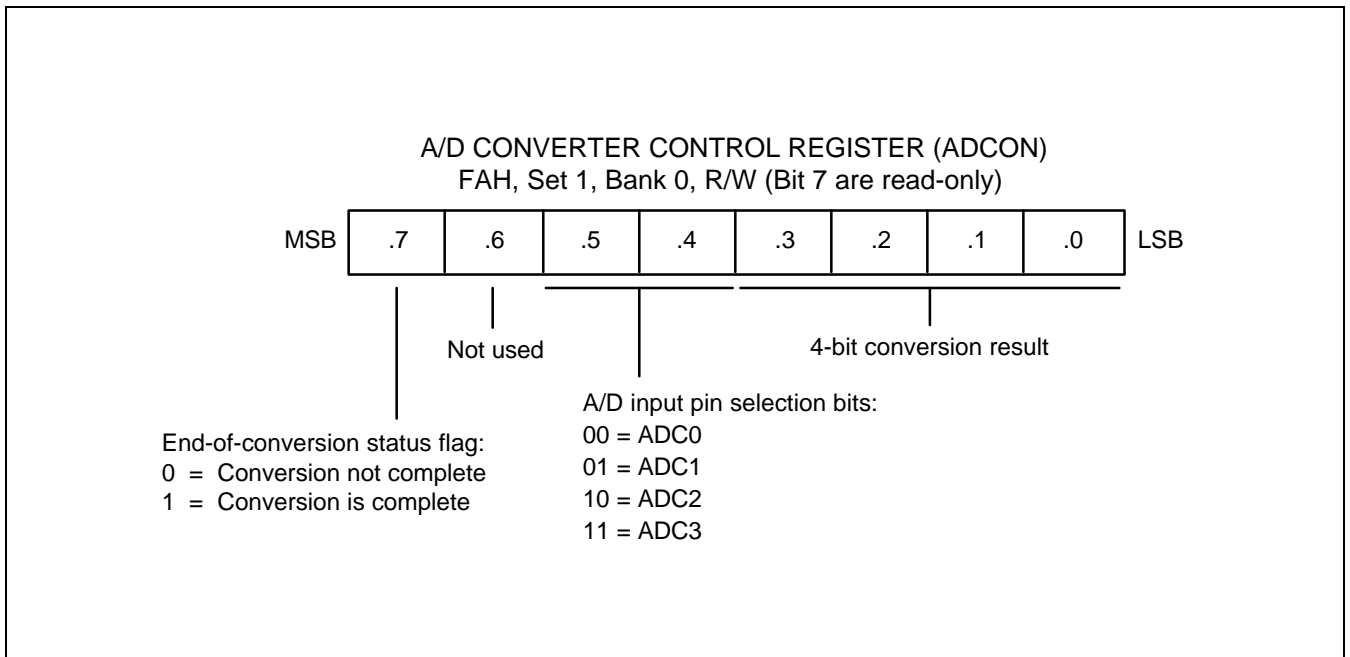


Figure 14-2. A/D Converter Control Register (ADCON)

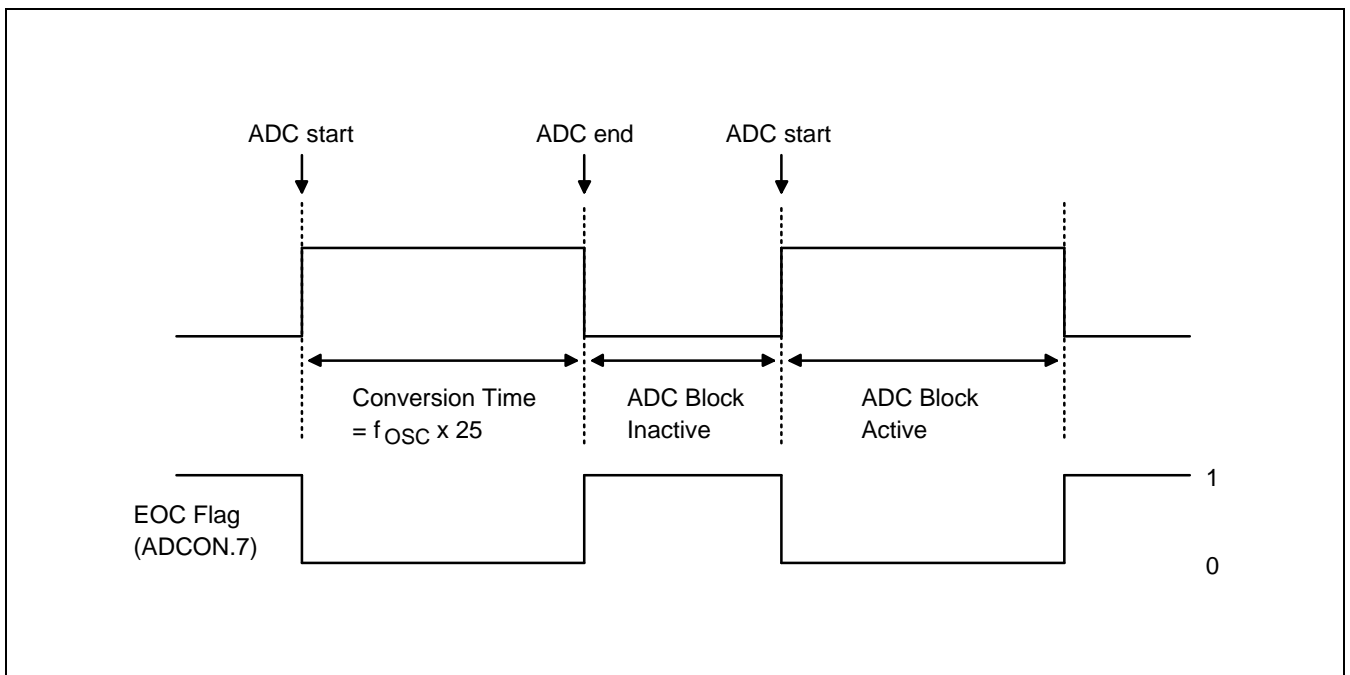


Figure 14-3. A/D Converter Function Timing Diagram

INTERNAL A/D CONVERSION PROCEDURE

1. Select the analog input channel by writing the appropriate value to ADCON.5 and ADCON.4. 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 S3C8847/C8849/P8849, the analog input level must be within the range of 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.

```
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
          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 the median value of the 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

```

(Continued on next page)

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

```

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       ;
        .
        .
        .
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
        .
        .
        .

```

15 ELECTRICAL DATA

OVERVIEW

In this section, the S3C8847 and the S3C8849 electrical characteristics are presented in tables and graphs. The information is arranged in the following order:

- Absolute maximum ratings
- D.C. electrical characteristics
- I/O capacitance
- A.C. electrical characteristics
- Input timing measurement points for t_{NF1} and t_{NF2}
- Data retention supply voltage in Stop mode
- Stop mode release timing when initiated by RESET
- Main oscillator and L-C oscillator frequency
- Clock timing measurement points for X_{IN}
- Main oscillator clock stabilization time (t_{ST})
- A/D converter electrical characteristics
- Characteristic curves

Table 15-1. Absolute Maximum Ratings

 $(T_A = 25\text{ }^\circ\text{C})$

Parameter	Symbol	Conditions	Rating	Unit
Supply Voltage	V_{DD}	–	– 0.3 to + 6.0	V
Input Voltage	V_{I1}	P1.0–P1.5 (open-drain)	– 0.3 to + 7	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 1	+ 100	
		Total pin current for ports 0, 2, and 3	+ 100	
Operating Temperature	T_A	–	– 20 to + 85	$^\circ\text{C}$
Storage Temperature	T_{STG}	–	– 65 to + 150	$^\circ\text{C}$

Table 15-2. D.C. Electrical Characteristics

 $(T_A = -20\text{ }^\circ\text{C}$ to $+85\text{ }^\circ\text{C}$, $V_{DD} = 4.5\text{ V}$ to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input High Voltage	V_{IH1}	All input pins except V_{IH2}	$0.8 V_{DD}$	–	V_{DD}	V
	V_{IH2}	X_{IN} , X_{OUT}	2.7 V			
Input Low Voltage	V_{IL1}	All input pins except V_{IL2}	–	–	$0.2 V_{DD}$	V
	V_{IL2}	X_{IN} , X_{OUT}			1.0 V	
Output High Voltage	V_{OH}	$I_{OH} = -500\text{ }\mu\text{A}$ P0.0–P0.5, P1.6–P1.7, P2 R, G, B, Vblank	$V_{DD} - 0.8$	–	–	V
Output Low Voltage	V_{OL1}	$I_{OL} = 4\text{ mA}$ P0.0–P0.5, P1.6–P1.7	–	–	0.4	V
	V_{OL2}	$I_{OL} = 10\text{ mA}$ P1.4–P1.5	–	–	0.8	
	V_{OL3}	$I_{OL} = 2\text{ mA}$ P1.0–P1.3, P3.0–P3.1, P0.6–P0.7	–	–	0.4	
	V_{OL4}	$I_{OL} = 1\text{ mA}$ R, G, B, Vblank, P2	–	–	0.4	V

Table 15-2. D.C. Electrical Characteristics (Continued)

(T_A = -20 °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} , OSC _{IN} , OSC _{OUT}			10	
	I _{LIH3}	V _{IN} = V _{DD} , X _{IN} , X _{OUT}	2.5	10	20	
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, OSC _{IN} , OSC _{OUT}			– 10	
	I _{LIL3}	V _{IN} = 0 V, X _{IN} , X _{OUT}	– 2.5	– 10	– 20	
Output High Leakage Current	I _{LOH1}	V _{OUT} = V _{DD} All output pins except I _{LOH2}	–	–	3	μA
	I _{LOH2}	V _{OUT} = 6 V P1.0–P1.5			10	
Output Low Leakage Current	I _{LOL}	V _{OUT} = 0 V All output pins	–	–	– 3	μA
Supply Current (note)	I _{DD1}	Normal mode; V _{DD} = 4.5 V to 5.5 V 8-MHz CPU clock	–	7	20	mA
	I _{DD2}	Idle mode; V _{DD} = 4.5 V to 5.5 V 8-MHz CPU clock		2	10	
	I _{DD3}	Stop mode; V _{DD} = 4.5 V to 5.5 V		1	10	μA

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

Table 15-3. Input/output Capacitance

(T_A = -20 °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 15-4. A.C. Electrical Characteristics

(T_A = -20 °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}	P1.0-P1.3	-	350	-	ns
	t _{NF2}	RESET, H-sync, V-sync	-	1000		
	t _{NF3}	Glitch filter (oscillator block)	-	25		
	t _{NF4}	CAPA	-	5	-	t _{CAPA}

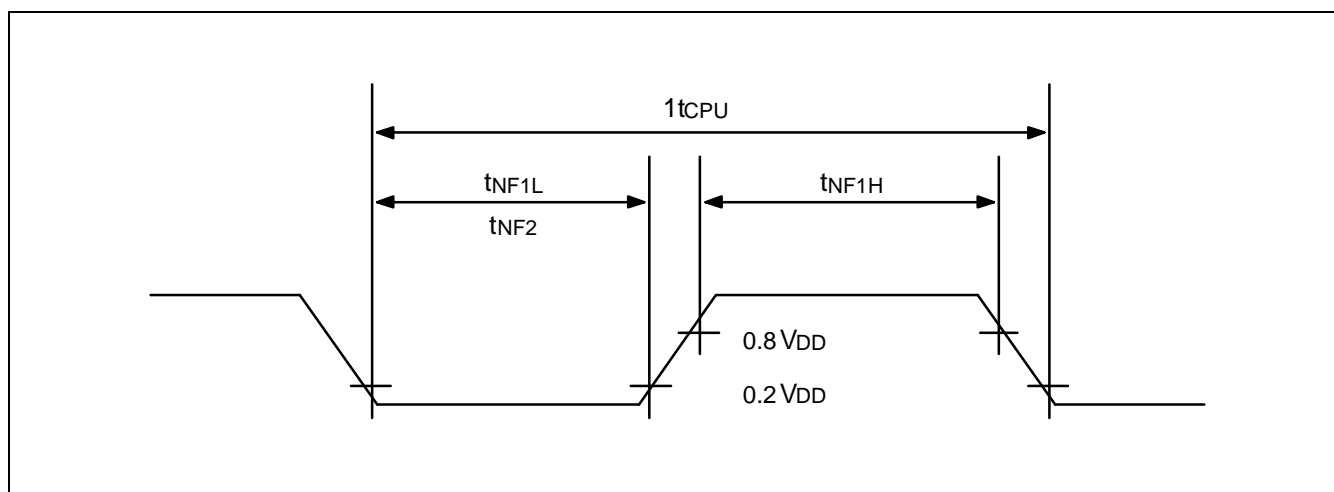
NOTE: f_{CAPA} = f_{OSC}/128Figure 15-1. Input Timing Measurement Points for t_{NF1} and t_{NF2}

Table 15-5. Data Retention Supply Voltage in Stop Mode

($T_A = -20\text{ }^\circ\text{C}$ to $+85\text{ }^\circ\text{C}$)

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, $V_{DDDR} = 2.0\text{ V}$	–	–	5	μA

NOTES:

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

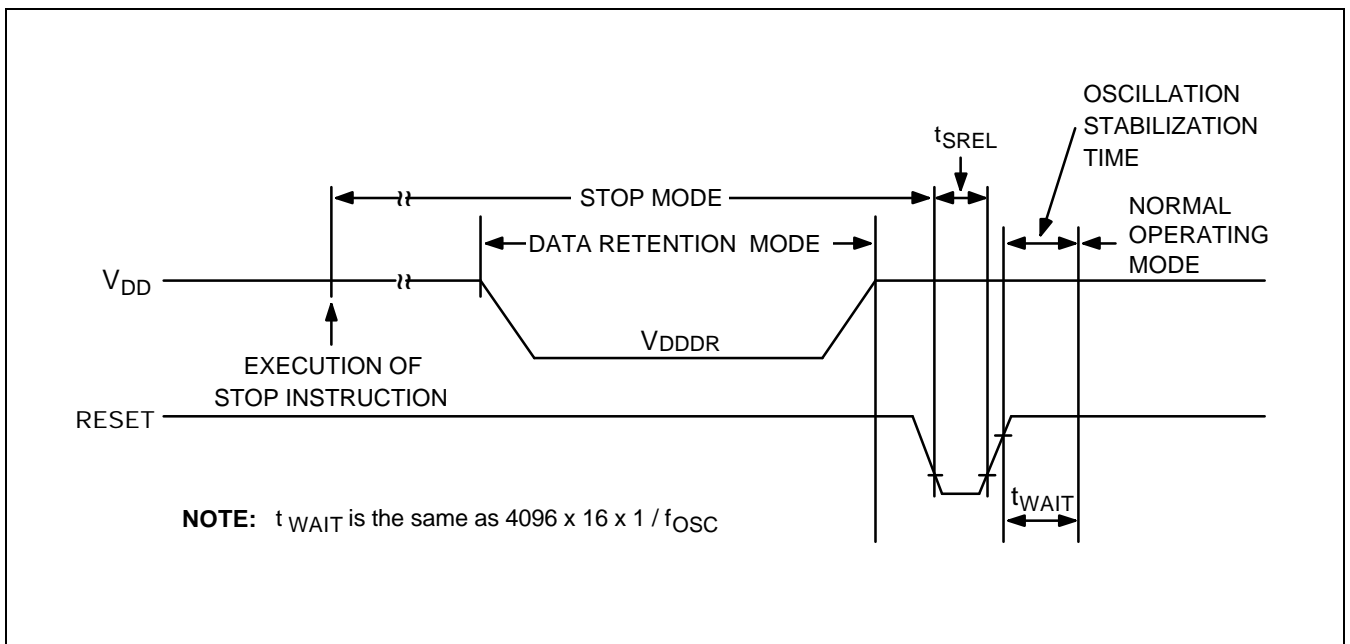
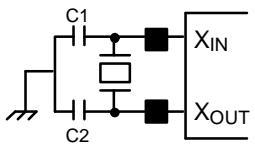
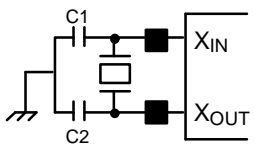
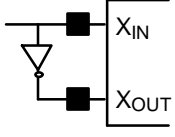
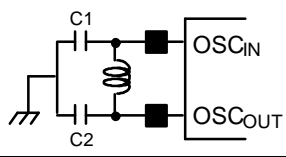


Figure 15-2. Stop Mode Release Timing When Initiated by a RESET

Table 15-6. Main Oscillator and L-C Oscillator Frequency

($T_A = -20\text{ }^\circ\text{C} + 85\text{ }^\circ\text{C}$, $V_{DD} = 4.5\text{ V to } 5.5\text{ V}$)

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		Recommend value; C1 = C2 = 20 pF	5	6.5	8	MHz
CPU Clock Frequency		–	0.032	6.0	8	MHz

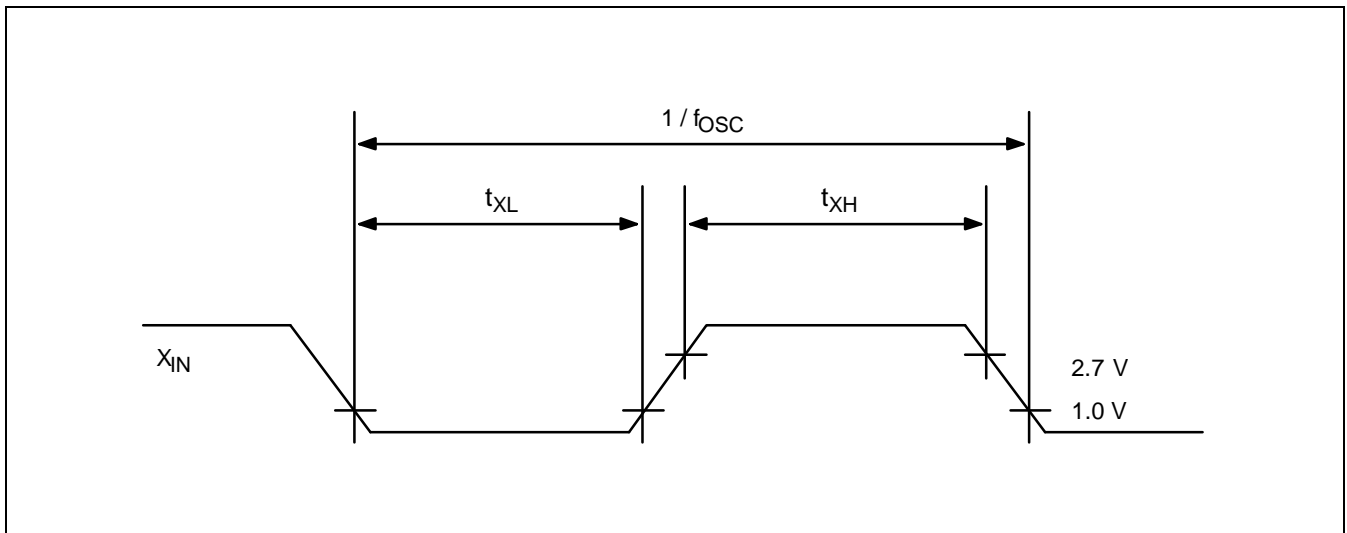


Figure 15-3. Clock Timing Measurement Points for X_{IN}

Table 15-7. Main Oscillator Clock Stabilization Time

(T_A = -20 °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 6.0 V (Oscillation stabilization occurs when V _{DD} is equal to the minimum oscillator voltage range.)	-	-	20	ms
Ceramic					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	-	1000	-	ns
Oscillation Stabilization Wait Time (1)	t _{WAIT}	CPU clock = 8 MHz; Stop mode released by RESET	-	8.3	-	ms
		CPU clock = 8 MHz; Stop mode released by an interrupt		(2)		

NOTES:

- 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.
- The oscillation stabilization interval is determined by the basic timer (BT) input clock setting.

Table 15-8. A/D Converter Electrical Characteristics

(T_A = -20 °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	-	-	± 0.5	LSB
Conversion Time (2)	t _{CON}		t _{CPU} × 25 (3)	-	-	-
Analog Input Voltage	V _{IAN}	-	V _{SS}	-	V _{DD}	V
Analog Input Impedance	R _{AN}	-	2	-	-	MΩ

NOTES:

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

16 MECHANICAL DATA

OVERVIEW

The S3C8847 and the S3C8849 microcontrollers are available in 42-pin SIP package (42-SDIP-600).

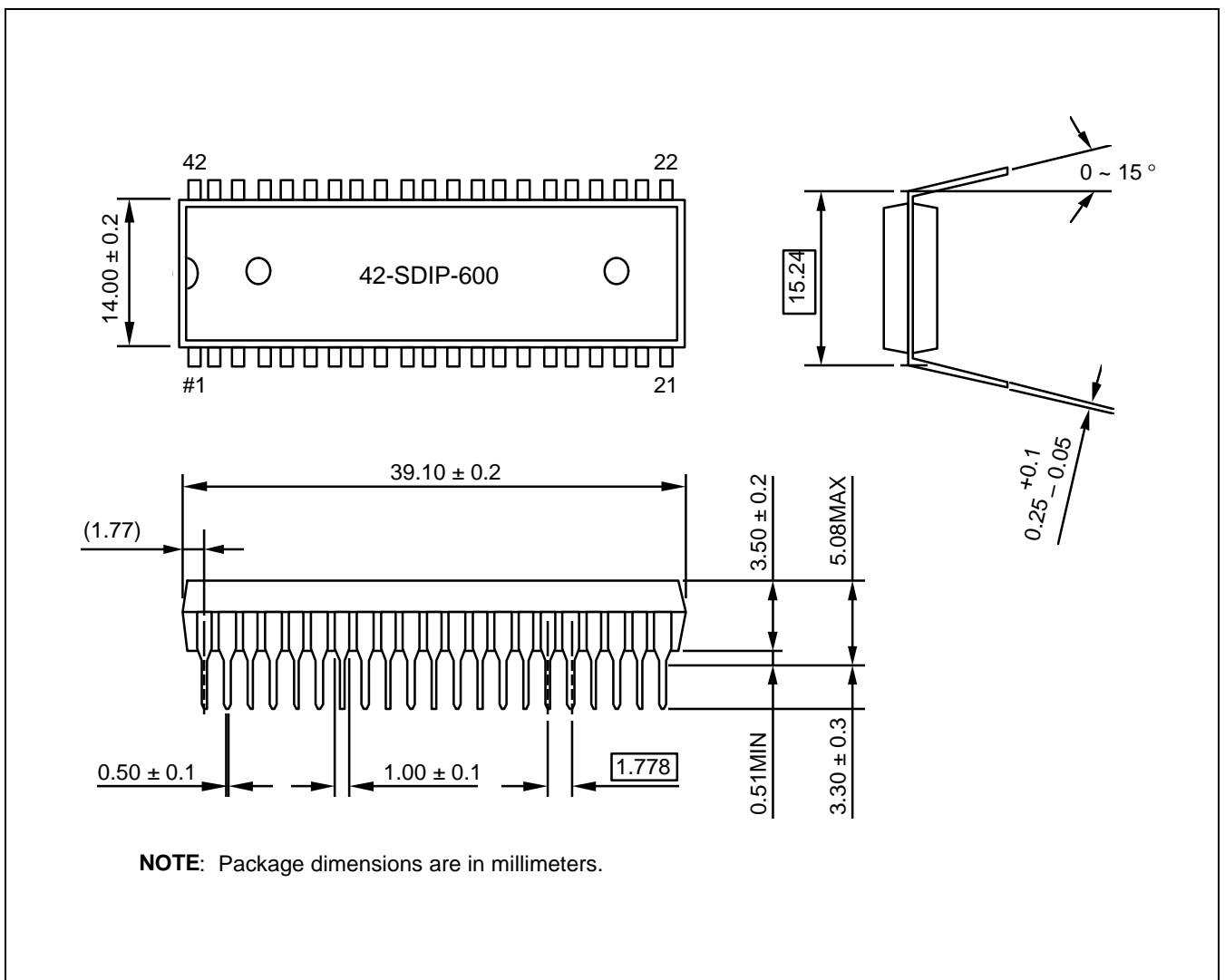


Figure 16-1. 42-Pin SDIP Package Mechanical Data (42-SDIP-600)

17

S3P8849 OTP

OVERVIEW

The S3P8849 single-chip CMOS microcontroller is the OTP (One Time Programmable) version of the S3C8847/C8849 microcontroller. It has an on-chip OTP ROM instead of a masked ROM. The EPROM is accessed by serial data format.

The S3P8849 is fully compatible with the S3C8847/C8849, both in function and pin configuration. The simple programming requirements of the S3P8849 make the device ideal for use as an evaluation chip for the S3C8847/C8849.

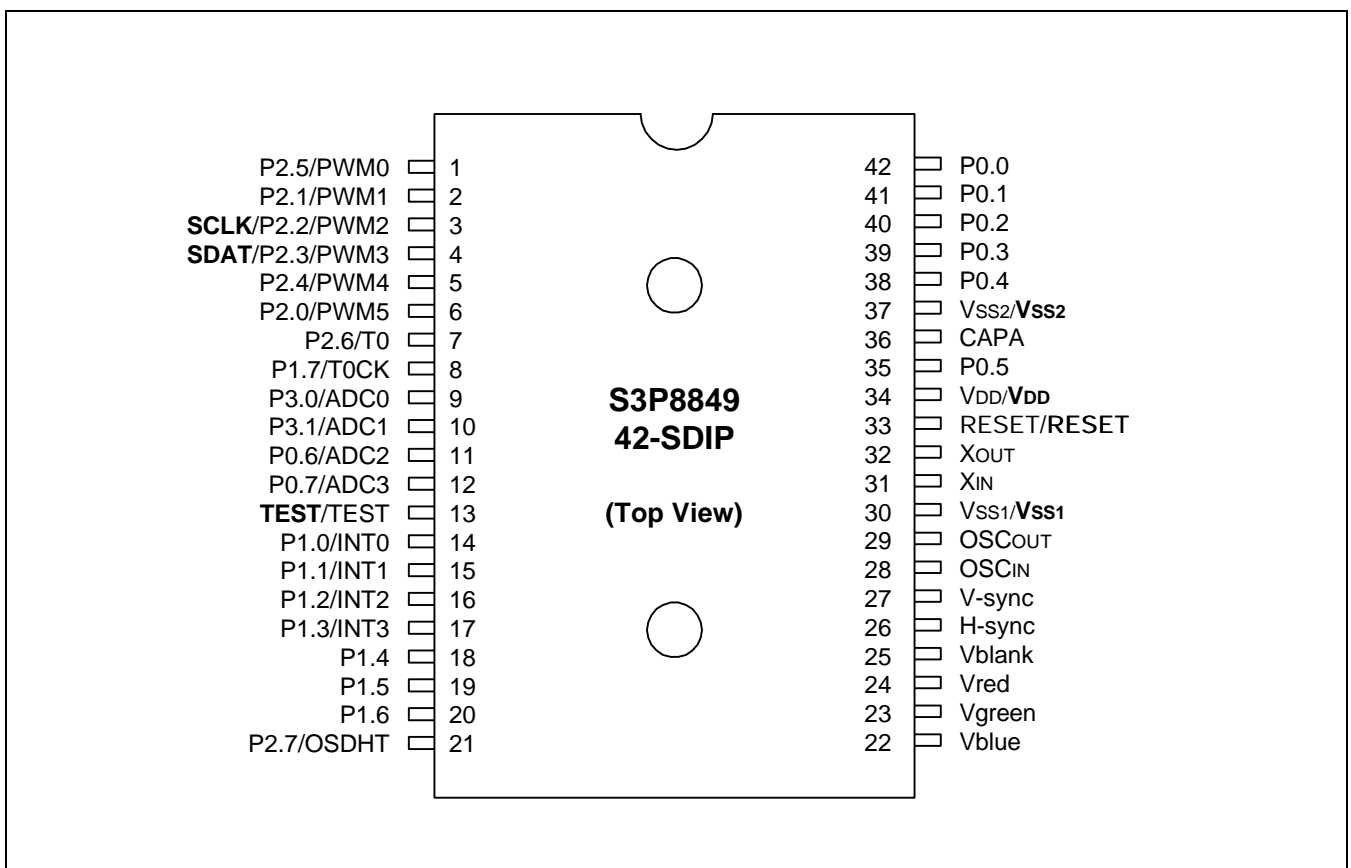


Figure 17-1. S3P8849 Pin Assignment (42-SDIP)

Table 17-1. Descriptions of Pins Used to Read/Write the EPROM (S3P8849)

Main Chip Pin Name	During Programming			
	Pin Name	Pin No.	I/O	Function
P2.3 (Pin 4)	SDAT	4	I/O	Serial data pin (output when reading, Input when writing) Input and push-pull output port can be assigned
P2.2 (Pin 3)	SCLK	3	I/O	Serial clock pin (Input only pin)
TEST	V _{PP} (TEST)	13	I	0 V: operating mode 5 V: test mode 12.5 V: OTP mode
RESET	RESET	33	I	5 V: operating mode, 0 V: OTP mode
V _{DD} /V _{SS}	V _{DD} /V _{SS}	34/30, 37	I	Logic power supply pin.

Table 17-2. Comparison of S3P8849 and S3C8847/C8849 Features

Characteristic	S3P8849	S3C8847/C8849
Program Memory	32-K byte EPROM	24/32-K byte mask ROM
Operating Voltage (V _{DD})	4.5 V to 5.5 V	4.5 V to 5.5 V
OTP Programming Mode	V _{DD} = 5 V, TEST V _{PP} = 12.5 V	–
Pin Configuration	42 SDIP	42 SDIP
EPROM Programmability	User Program 1 time	Programmed at the factory

OPERATING MODE CHARACTERISTICS

When 12.5 V is supplied to the V_{PP} (TEST) pin of the S3P8849, the EPROM programming mode is entered. The operating mode (read, write, or read protection) is selected according to the input signals to the pins listed in Table 16-3 below.

Table 17-3. Operating Mode Selection Criteria

V _{DD}	V _{PP} (TEST)	REG/ MEM	ADDRESS (A15–A0)	R/W	MODE
5 V	5 V	0	0000H	1	EPROM read
	12.5 V	0	0000H	0	EPROM program
	12.5 V	0	0000H	1	EPROM verify
	12.5 V	1	0E3FH	0	EPROM read protection

NOTE: "0" means Low level; "1" means High level.

18 DEVELOPMENT TOOLS

OVERVIEW

Samsung provides a powerful and easy-to-use development support system in turnkey form. The development support system is configured with a host system, debugging tools, and support software. For the host system, any standard computer that operates with MS-DOS as its operating system can be used. One type of debugging tool including hardware and software is provided: the sophisticated and powerful in-circuit emulator, SMDS2+, for S3C7, S3C6, S3C8 families of microcontrollers. The SMDS2+ is a new and improved version of SMDS2. Samsung also offers support software that includes debugger, assembler, and a program for setting options.

SHINE

Samsung Host Interface for in-circuit Emulator, SHINE, is a multi-window based debugger for SMDS2+. SHINE provides pull-down and pop-up menus, mouse support, function/hot keys, and context-sensitive hyper-linked help. It has an advanced, multiple-windowed user interface that emphasizes ease of use. Each window can be sized, moved, scrolled, highlighted, added, or removed completely.

SAMA ASSEMBLER

The Samsung Arrangeable Microcontroller (SAM) Assembler, SAMA, is a universal assembler, and generates object code in standard hexadecimal format. Assembled program code includes the object code that is used for ROM data and required SMDS program control data. To assemble programs, SAMA requires a source file and an auxiliary definition (DEF) file with device specific information.

SASM88

The SASM88 is an relocatable assembler for Samsung's S3C8-series microcontrollers. The SASM88 takes a source file containing assembly language statements and translates into a corresponding source code, object code and comments. The SASM88 supports macros and conditional assembly. It runs on the MS-DOS operating system. It produces the relocatable object code only, so the user should link object file. Object files can be linked with other object files and loaded into memory.

HEX2ROM

HEX2ROM file generates ROM code from HEX file which has been produced by assembler. ROM code must be needed to fabricate a microcontroller which has a mask ROM. When generating the ROM code.(OBJ file) by HEX2ROM, the value 'FF' is filled into the unused ROM area up to the maximum ROM size of the target device automatically.

TARGET BOARDS

Target boards are available for all S3C8-series microcontrollers. All required target system cables and adapters are included with the device-specific target board.

OTPs

One times programmable microcontrollers (OTPs) are under development for S3C8847/C8849 microcontroller.

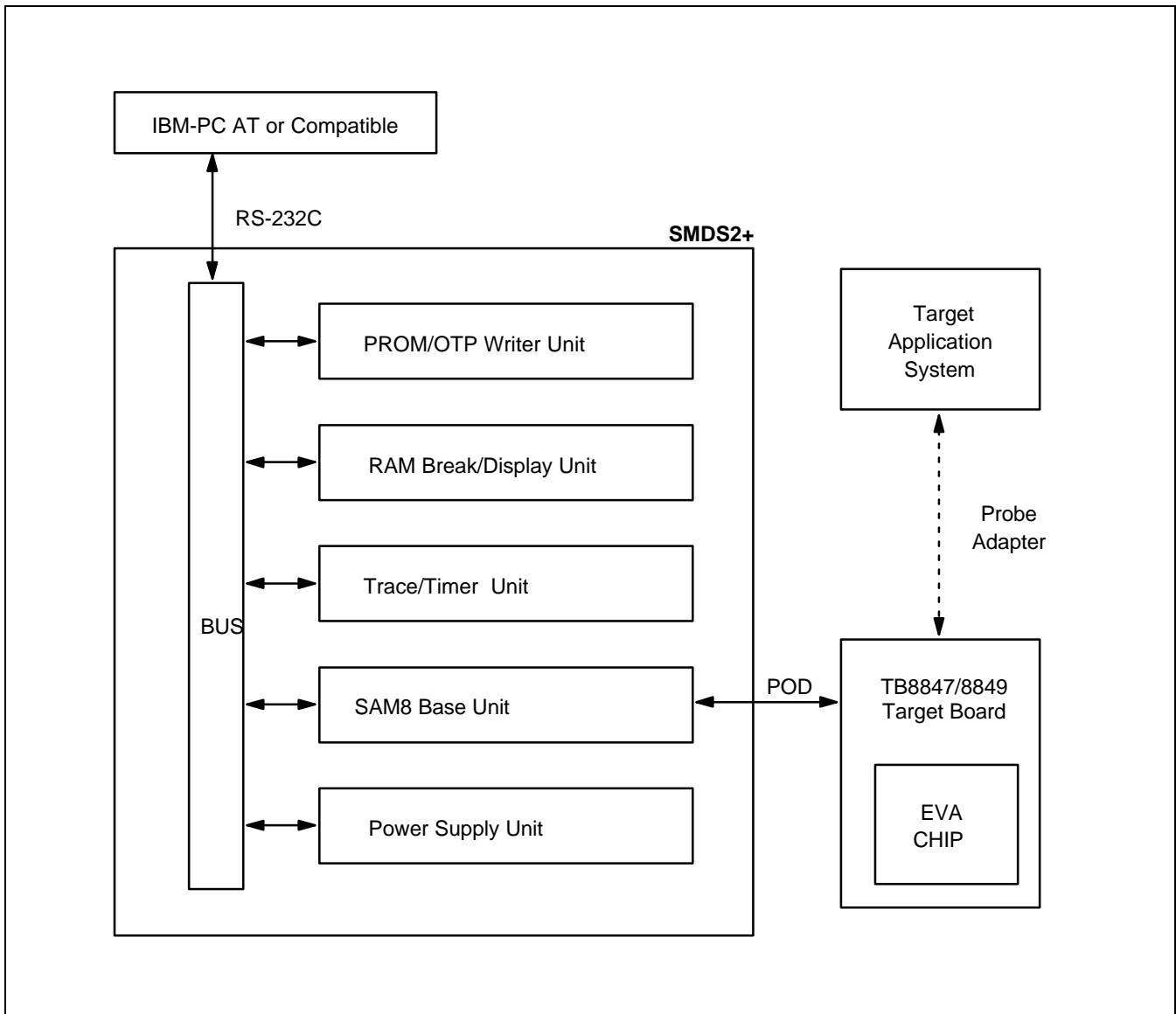


Figure 18-1. SMDS Product Configuration (SMDS2+)

TB8847/8849 TARGET BOARD

The TB8847/8849 target board is used for the S3C8847/C8849/P8849 microcontrollers. It is supported with the SMDS2+. The TB8847/8849 target board can also be used for S3C8847/C8849/P8849.

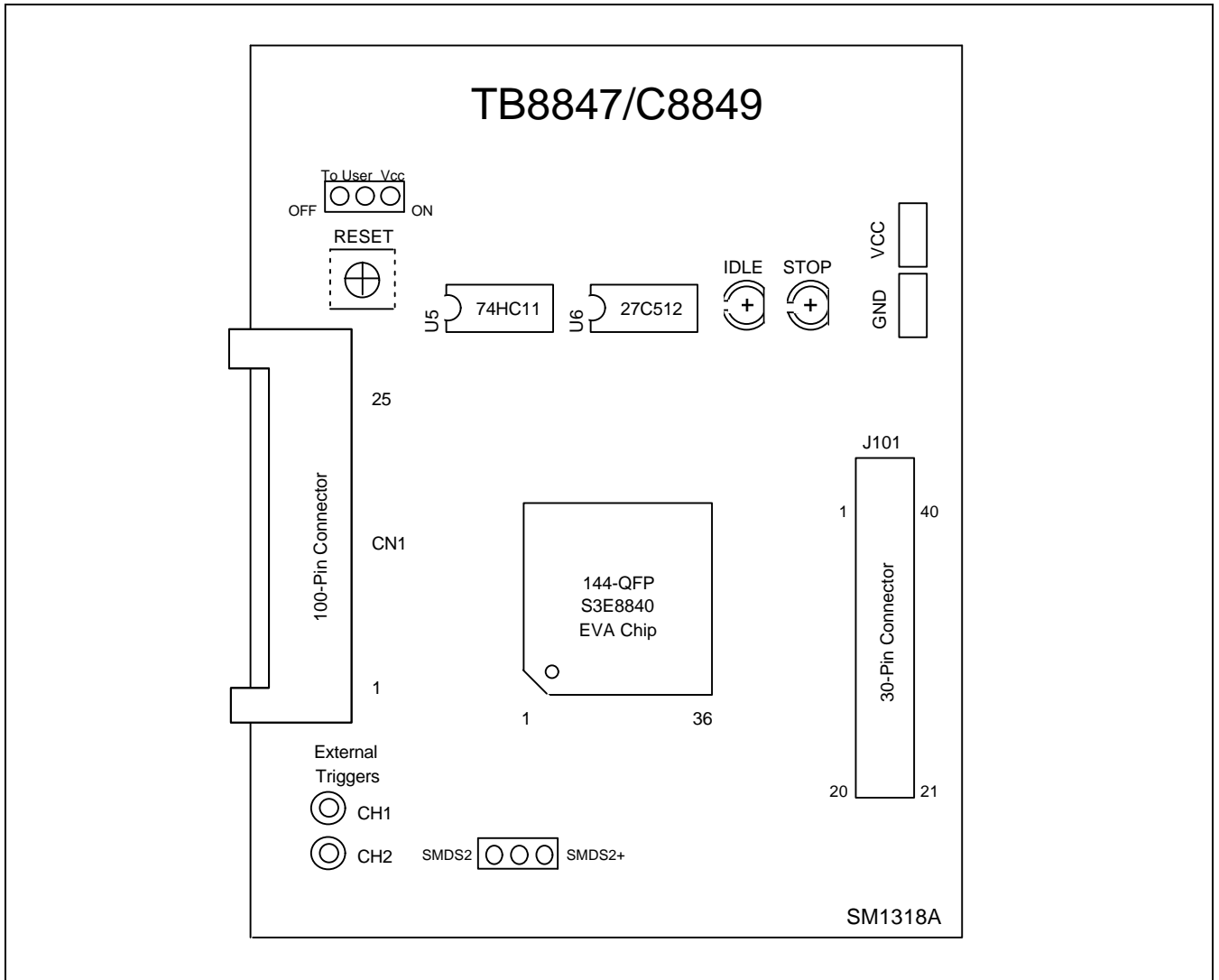

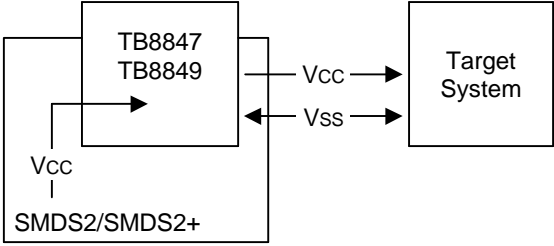

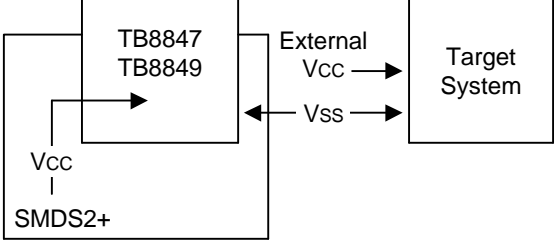


Figure 18-2. TB8847/8849 Target Board Configuration

Table 18-1. Power Selection Settings for TB8847/8849

'To User_Vcc' Settings	Operating Mode	Comments
To User_Vcc OFF  ON		The SMDS2+ main board supplies V_{CC} to the target board (evaluation chip) and the target system.
To User_Vcc OFF  ON		The SMDS2+ main board supplies V_{CC} only to the target board (evaluation chip). The target system must have its own power supply.

NOTE: The following symbol in the 'To User_Vcc' Setting column indicates the electrical short (off) configuration:



SMDS2+ Selection (SAM8)

In order to write data into program memory that is available in SMDS2+, the target board should be selected to be for SMDS2+ through a switch as follows. Otherwise, the program memory writing function is not available.

Table 18-2. The SMDS2 + Tool Selection Setting


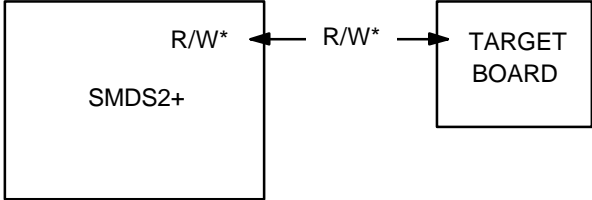
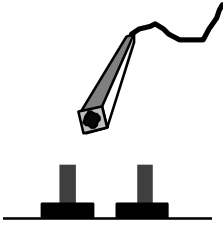
'SW1' Setting	Operating Mode
SMDS2  SMDS2+	

Table 18-3. Using Single Header Pins as the Input Path for External Trigger Sources

Target Board Part	Comments
<p>EXTERNAL TRIGGERS</p> <p>○ CH1</p> <p>○ CH2</p>	<div style="display: flex; align-items: center;">  <div style="margin-left: 20px;"> <p>Connector from external trigger sources of the application system</p> </div> </div> <p>You can connect an external trigger source to one of the two external trigger channels (CH1 or CH2) for the SMDS2+ breakpoint and trace functions.</p>

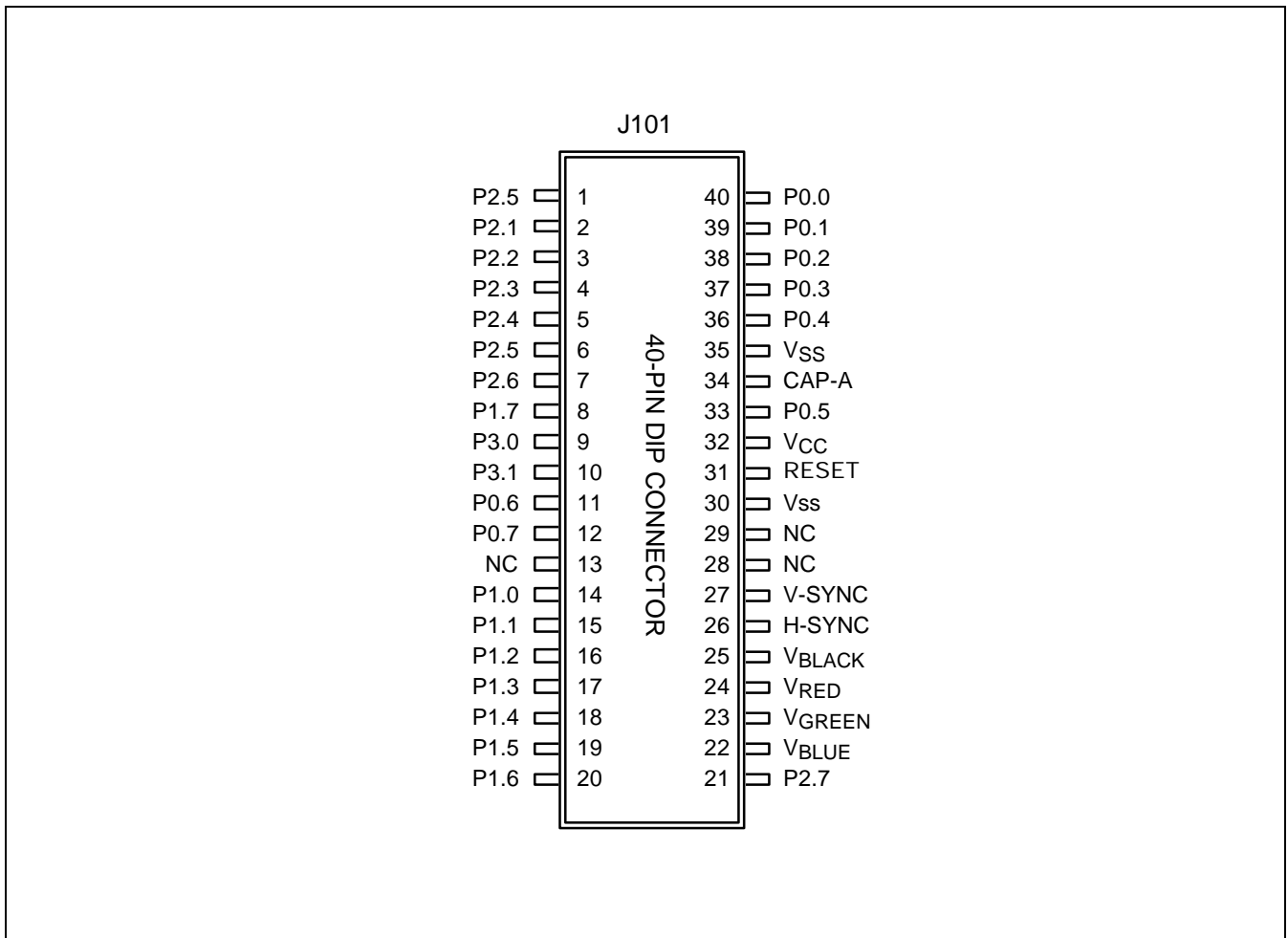


Figure 18-3. 40-Pin DIP Connector J101 for TB8847/8849

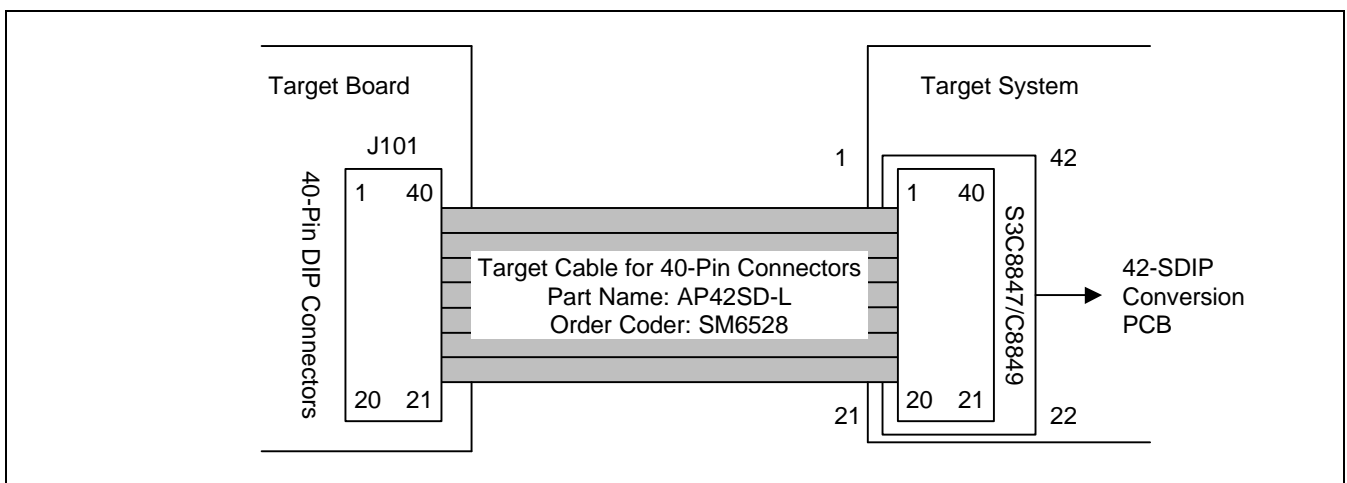


Figure 18-4. S3C8847/C8849 Probe Adapter for 42-SDIP Package