zilog®

A Littelfuse Company

S3 Family 8-Bit Microcontrollers

S3F8S6B MCU

Product Specification

PS032408-0220

PRELIMINARY

www.zilog.com

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer

 Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZILOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZILOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.

S3 and Z8 are trademarks or registered trademarks of Zilog, Inc. All other product or service names are the property of their respective owners.

Revision History

Each instance in this document's revision history reflects a change from its previous edition. For more details, refer to the corresponding page(s) or appropriate links furnished in the table below.

Date	Revision Level	Description	Page
Feb 2020	08	Updated to new Zilog logo.	All
Mar 2018	07	Replaced Figure 23-2 (64-LQFP Package Dimensions); corrected typos in LCD Mode Control Register description	23-2; 4-19
Dec 2016	06	Updated LCD Mode Control Register; added Table 15.1; updated Figure 15-5; updated Table 22.10	4-19; 15-4; 15-6; 22-14
May 2015	05	Updated title and header/footer information	All
Feb 2015	04	Reduced PDF file size	All
Feb 2015	03	Corrected Figure Title in Figure 1-3	1-8
Jan 2015	02	Updated the Third Parties for Development Tools section.	25-7
Mar 2014	01	Original Zilog issue. Deleted Pin Circuit Type A diagram from Product Overview chapter; changed "First interrupt" to "Fast interrupt", Figure 6-1.	1-14; 6-5
Jan 2014	1.00	First release.	
Jun 2013	0.00	First draft.	

Table of Contents

List of Figures

List of Tables

List of Examples

1 Product Overview

1.1 S3C8-Series Microcontrollers

Zilog's S3C8 series of 8-bit single-chip CMOS microcontrollers offers a fast and efficient CPU, a wide range of integrated peripherals, and various mask-programmable ROM sizes. Among the major CPU features are:

- Efficient register-oriented architecture
- Selectable CPU clock sources
- Idle and Stop power-down mode release by interrupts
- Built-in basic timer with watchdog function

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

1.2 S3F8S6B Microcontroller

The S3F8S6B single-chip CMOS microcontrollers are fabricated using the highly advanced CMOS process, based on Zilog's newest CPU architecture.

The S3F8S6B is a microcontroller with a 64KB Flash ROM embedded respectively.

Using a proven modular design approach, Zilog engineers have successfully developed the S3F8S6B by integrating the following peripheral modules with the powerful SAM8 core:

- Seven programmable I/O ports, including six 8-bit ports, and one 6-bit port, for a total of 54 pins
- Sixteen bit-programmable pins for external interrupts
- One 8-bit basic timer for oscillation stabilization and watchdog functions (system reset)
- Three 8-bit timer/counter and two 16-bit timer/counter with selectable operating modes
- Watch timer for real time
- LCD Controller/driver
- A/D converter with 8 selectable input pins
- Synchronous SIO modules
- Two asynchronous UART modules
- Pattern generation module

They are currently available in 64-pin QFP and 64-pin SDIP package

1.3 Features

CPU

- SAM88 RC CPU core

Memory

- Program Memory (ROM)
  - 64K × 8 bits program memory
  - Internal Flash memory (program memory)
    - Sector size: 128 bytes
    - 10 years data retention
    - Fast programming time
    - User program and sector erase available
    - Endurance: 10,000 erase/program cycles
    - External serial programming support
    - Expandable OBPTM (on board program) sector
- Data Memory (RAM)
  - Including LCD display data memory
  - 2,098 × 8 bits data memory

Instruction Set

- 78 instructions
- Idle and stop instructions added for power-down modes

54 I/O Pins

- I/O: 18 pins (Sharing with other signal pins)
- I/O: 36 pins (Sharing with LCD signal outputs)

Interrupts

- 8 interrupt levels and 30 interrupt sources
- Fast interrupt processing feature

8-Bit Basic Timer

- Watchdog timer function
- 4 kinds of clock source

8-Bit Timer/Counter A

- Programmable 8-bit internal timer
- External event counter function
- PWM and capture function

8-Bit Timer/Counter B

- Programmable 8-bit internal timer
- Carrier frequency generator

Two 8-Bit Timer/Counter C

- Programmable 8-bit internal timer
- PWM function

Two 16-Bit Timer/Counter (D0/D1)

- Programmable 16-bit internal timer
- External event counter function
- PWM and capture function

Watch Timer

- Interval time: 1.995 mS, 0.125S, 0.25S, and 0.5S at 32.768 kHz
- 0.5/1/2/4 kHz Selectable buzzer output

LCD Controller/Driver

- 28 segments and 8 common terminals
- 1/2, 1/3, 1/4, and 1/8 duty selectable
- Capacitor or resistor bias selectable
- Regulator and booster circuit for LCD bias

Analog to Digital Converter

- 8-channel analog input
- 10-bit conversion resolution
- 25 μs conversion time

Two Channels UART

- Full-duplex serial I/O interface
- Four programmable operating modes
- Auto generating parity bit

8-bit Serial I/O Interface

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

Pattern Generation Module

- Pattern generation module triggered by timer match signal and software

Low Voltage Reset (LVR)

- Criteria voltage: 1.9 V,2.2 V
- En/Disable by smart option (ROM address: 3FH)

Two Power-Down Modes

- Idle: only CPU clock stops
- Stop: selected system clock and CPU clock stop

Oscillation Sources

- Crystal, ceramic, or RC for main clock
- Main clock frequency: 0.4 MHz to 12.0 MHz
- 32.768 kHz crystal oscillation circuit for sub clock

Instruction Execution Times

- 333 ns at 12.0 MHz fx (Minimum)
- 122.1 μs at 32.768 kHz fxt (Minimum)

Operating Voltage Range

- 1.8 V to 5.5 V at 0.4 to 4.2 MHz
- 2.2 V to 5.5 V at 0.4 to 12.0 MHz

Operating Temperature Range

- – 40 °C to + 85 °C

Package Type

- 64-QFP-1420F, 64-LQFP-1010, 64-SDIP-750

IVC

- Internal Voltage Converter for 5 V operations

Smart Option

- Low Voltage Reset (LVR) level and enable/disable are at your hardwired option (ROM address 3FH)
- ISP related option selectable (ROM address 3EH)

1.4 Block Diagram

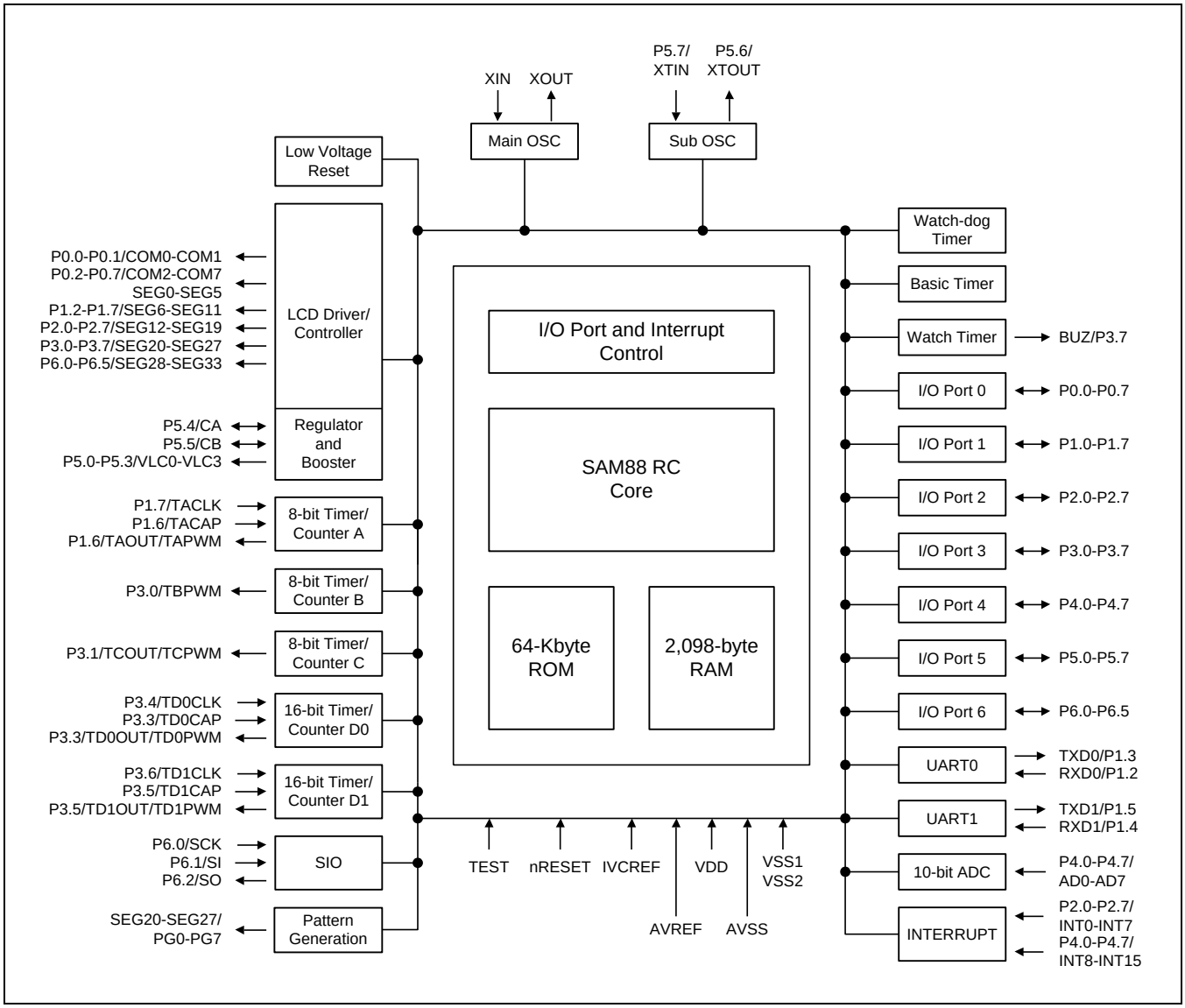


Figure 1-1 Block Diagram

1.5 Pin Assignment

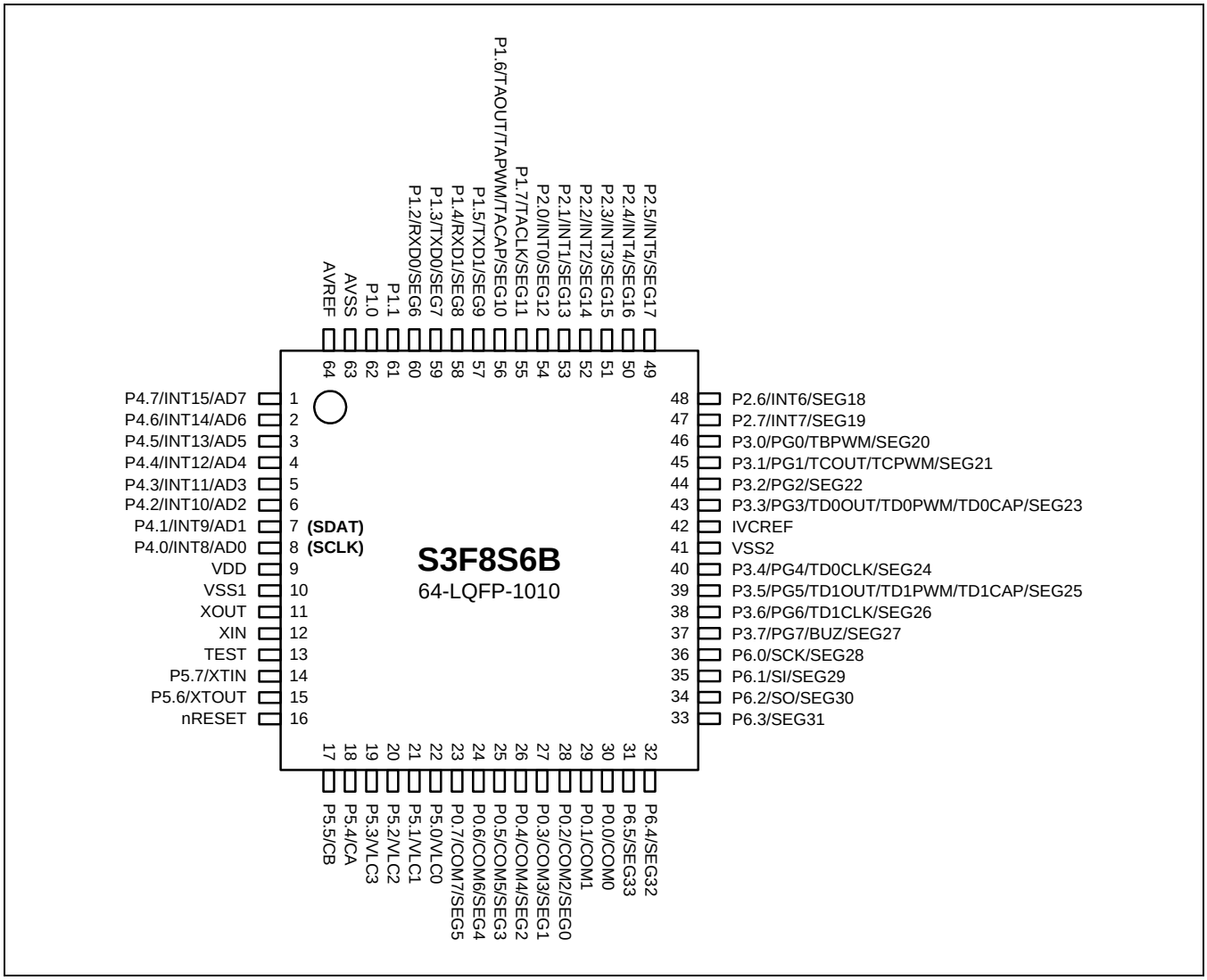


Figure 1-2 S3F8S6B Pin Assignments (64-LQFP-1010)

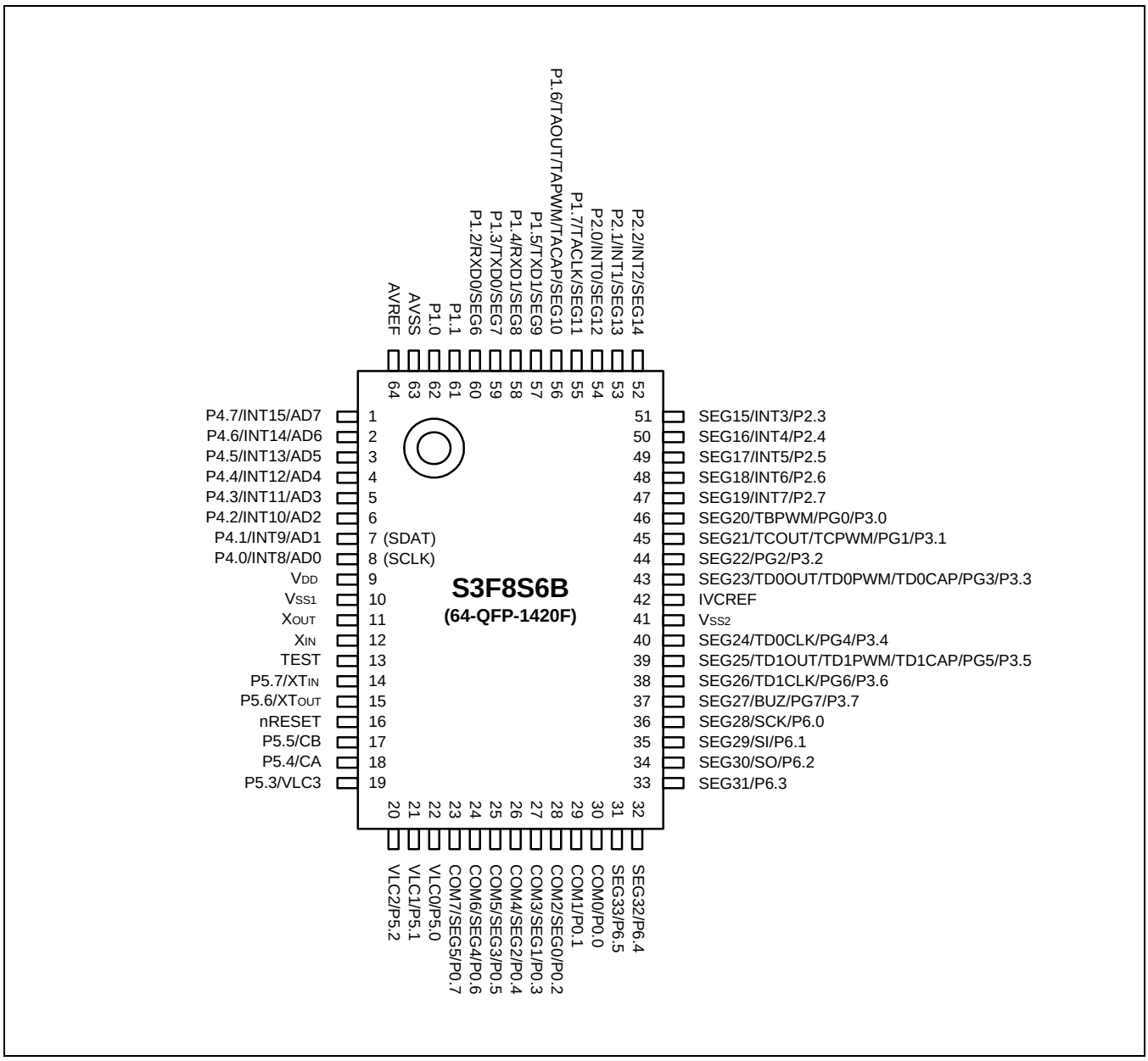


Figure 1-3 S3F8S6B Pin Assignments (64-QFP-1420F)

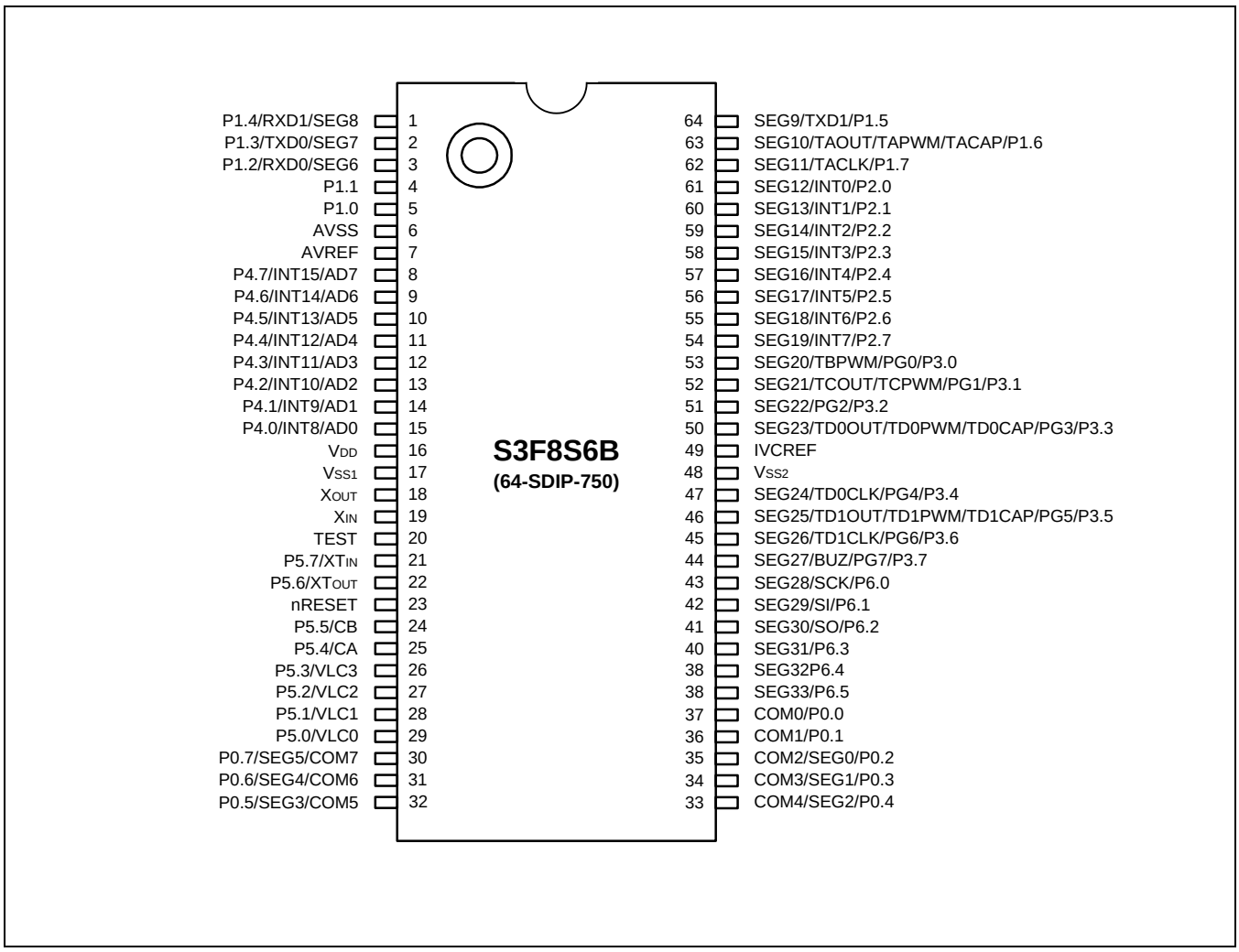


Figure 1-4 S3F8S6B Pin Assignments (64-SDIP-750)

1.6 Pin Descriptions

Table 1.1 S3F8S6B Pin Descriptions

Pin Names	Pin Type	Pin Description	Circuit Type	Pin Numbers (NOTE)	Share Pins
P0.0 P0.1 P0.2–P0.7	I/O	I/O port with bit-programmable pins; Input or push-pull output and software assignable pull-ups.	H-44	30(37) 29(36) 28–23 (35–30)	COM0 COM1 COM2–COM7/ SEG0–SEG5
P1.0 P1.1 P1.2 P1.3 P1.4 P1.5 P1.6 P1.7	I/O	I/O port with bit-programmable pins; Input or push-pull, open-drain output and software assignable pull-ups.	H-42	62(5) 61(4) 60(3) 59(2)Pin 58(1) 57(64) 56(63) 55(62)	– – RxD0/SEG6 TxD0/SEG7 RxD1/SEG8 TxD1/SEG9 TAOUT/TAPW M/TACAP/SEG 10 TACLK/SEG11
P2.0–P2.7	I/O	I/O port with bit-programmable pins; Schmitt trigger input or push-pull output and software assignable pull-ups. Alternately used for external interrupt input (noise filters, interrupt enable and pending control).	H-43	54–47 (61–54)	INT0–INT7/ SEG12–SEG19
P3.0 P3.1 P3.2 P3.3 P3.4 P3.5 P3.6 P3.7	I/O	I/O port with bit-programmable pins; Input or push-pull, open-drain output and software assignable pull-ups.	H-42	46(53) 45(52) 44(51) 43(50) 40(47) 39(46) 38(45) 37(44)	TBPWM/PG0/ SEG20 TCOUT/TCPW M/PG1/SEG21 PG2/SEG22 TD0OUT/ TD0PWM/ TD0CAP/ PG3/SEG23 TD0CLK/PG4/ SEG24 TD1OUT/ TD1PWM/ TD1CAP/ PG5/SEG25 TD1CLK/PG6/ SEG26 BUZ/PG7/ SEG27
P4.0–P4.7	I/O	The P4.2-P4.7 are the I/O port with bit-programmable pins, but the P4.0/P4.1 are the I/O port with two-bits-programmable pins Schmitt trigger input or push-pull output and software assignable pull-ups.	F-16	8–1 (15–8)	AD0–AD7/ INT8–INT15
P5.0–P5.3	I/O	I/O port with bit-programmable pins; Input or push-pull output and software	F-17	22–19 (29–26)	VLC0–VLC3

Pin Names	Pin Type	Pin Description	Circuit Type	Pin Numbers (NOTE)	Share Pins
P5.4 P5.5		assignable pull-ups.		18(25) 17(24)	CA CB
P5.6 P5.7	I/O	I/O port with bit-programmable pins; Input or push-pull output and software assignable pull-ups.	D-2	15(22) 14(21)	XT_{OUT} XT_{IN}
P6.0 P6.1 P6.2 P6.3–P6.5	I/O	I/O port with bit-programmable pins; Input or push-pull, open-drain output and software assignable pull-ups.	H-42	36(43) 35(42) 34(41) 33–31 (40–38)	SCK/SEG28 SI/SEG29 SO/SEG30 SEG31–SEG33
COM0–COM1 COM2–COM7	I/O	LCD common signal output.	H-44	30–29 (37–36) 28–23 (35–30)	P0.0–P0.1 P0.2–P0.7/ SEG0–SEG5
SEG0– SEG5	I/O	LCD segment signal output.	H-44	28–23 (35–30)	P0.2–P0.7/ COM2–COM7
SEG6 SEG7 SEG8 SEG9 SEG10 SEG11	I/O	LCD segment signal output.	H-42	60(3) 59(2) 58(1) 57(64) 56(63) 55(62)	P1.2/RxD0 P1.3/TxD0 P1.4/RxD1 P1.5/TxD1 P1.6/TAOUT/ TAPWM/TACAP P1.7/TACLK
SEG12–SEG19	I/O	LCD segment signal output.	H-43	54–47 (61–54)	P2.0–P2.7/ INT0–INT7
SEG20 SEG21 SEG22 SEG23 SEG24 SEG25 SEG26 SEG27	I/O	LCD segment signal output.	H-42	46(53) 45(52) 44(51) 43(50) 40(47) 39(46) 38(45) 37(44)	P3.0/ PG0/ TBPWM P3.1/PG1/ TCOUT/TCPWM P3.2/PG2 P3.3/PG3/ TD0OUT/ TD0PWM/ TD0CAP P3.4/PG4/TD0CLK P3.5/PG5/ TD1OUT/ TD1PWM/ TD1CAP P3.6/PG6/TD1CLK P3.7/PG7/BUZ
SEG28 SEG29	I/O	LCD segment signal output.	H-42	36(43) 35(42)	P6.0/SCK P6.1/SI

Pin Names	Pin Type	Pin Description	Circuit Type	Pin Numbers (NOTE)	Share Pins
SEG30 SEG31–SEG33				34(41) 33–31 (40–38)	P6.2/SO P6.3–P6.5
VLC0–VLC3	I/O	LCD power supply pins.	F-17	22–19 (29–26)	P5.0–P5.3
CA CB	I/O	Capacitor terminal for voltage booster.	F-17	18(25) 17(24)	P5.4 P5.5
AD0–AD7	I/O	A/D converter analog input channels.	F-16	8–1 (15–8)	P4.0–P4.7/ INT8–INT15
AV_{REF}	–	A/D converter reference voltage.	–	64(7)	–
AV_{SS}	–	A/D converter ground.	–	63(6)	–
PG0–PG3 PG4–PG7	I/O	Pattern generation output.	H-42	46–43 (53–50) 40–37 (47–44)	P3.0–P3.3/ SEG20–SEG23 P3.4–P3.7/ SEG24–SEG27
TxD0 RxD0	I/O	UART 0 data output, input.	H-42	59(2) 60(3)	P1.3/SEG7 P1.2/SEG6
TxD1 RxD1	I/O	UART 1 data output, input.	H-42	57(64) 58(1)	P1.5/SEG9 P1.4/SEG8
TAOUT/ TAPWM	I/O	Timer A clock output and PWM output.	H-42	56(63)	P1.6/SEG10/ TACAP
TACAP	I/O	Timer A capture input.	H-42	56(63)	P1.6/SEG10/ TAOUT/TAPWM
TACLK	I/O	Timer A external clock input.	H-42	55(62)	P1.7/SEG11
TBPWM	I/O	Timer B carrier frequency output.	H-42	46(53)	P3.0/SEG20/ PG0
TCOUT/ TCPWM	I/O	Timer C clock output and PWM output.	H-42	45(52)	P3.1/SEG21/ PG1
TD0OUT/ TD0PWM	I/O	Timer D0 clock output and PWM output.	H-42	43(50)	P3.3/SEG23/ PG3/TD0CAP
TD0CAP	I/O	Timer D0 capture input.	H-42	43(50)	P3.3/SEG23/ PG3/TD0OUT/ TD0PWM
TD0CLK	I/O	Timer D0 external clock input.	H-42	40(47)	P3.4/SEG24/ PG4
TD1OUT/ TD1PWM	I/O	Timer D1 clock output and PWM output.	H-42	39(46)	P3.5/SEG25/ PG5/TD1CAP
TD1CAP	I/O	Timer D1 capture input.	H-42	39(46)	P3.5/SEG25/ PG5/TD1OUT/ TD1PWM
TD1CLK	I/O	Timer D1 external clock input.	H-42	38(45)	P3.6/SEG26/ PG6

Pin Names	Pin Type	Pin Description	Circuit Type	Pin Numbers (NOTE)	Share Pins
BUZ	I/O	Output pin for buzzer signal.	H-42	37(44)	P3.7/SEG27/ PG7
SCK	I/O	Serial interface clock.	H-42	36(43)	P6.0/SEG28
SI	I/O	Serial interface data input.	H-42	35(42)	P6.1/SEG29
SO	I/O	Serial interface data output.	H-42	34(41)	P6.2/SEG30
INT0–INT7	I/O	External interrupts input pins.	H-43	54–47 (61–54)	P2.0–P2.7/ SEG12–SEG19
INT8–INT15	I/O	External interrupts input pins.	F-16	1–8 (8–15)	P4.0–P4.7/ AD0–AD7
nRESET	I	System reset pin	B	16(23)	–
X_{IN} X_{OUT}	–	Main oscillator pins.	–	12(19) 11(18)	–
XT_{IN} XT_{OUT}	–	Crystal oscillator pins for sub clock.	–	14(21) 15(22)	P5.7 P5.6
TEST	I	Test input: it must be connected to VSS	–	13(20)	–
V_{DD}	–	Power supply input pins.	–	9(16)	–
V_{SS1} V_{SS2}	–	Ground pins.	–	10(17) 41(48)	–
IVC_{REF}	–	Internal voltage controller reference input pin. A capacitor (0.1uF) must be connected between IVCREF and V_{SS}.	–	42(49)	–

NOTE: Parentheses indicate pin number for 64-SDIP-750 package.

1.7 Pin Circuits

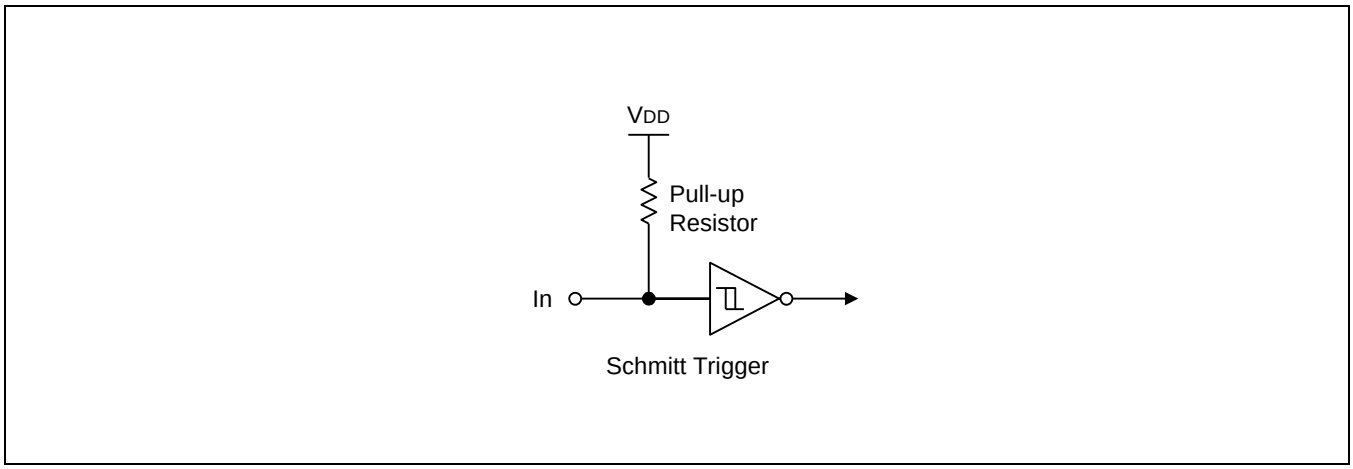


Figure 1-5 Pin Circuit Type B

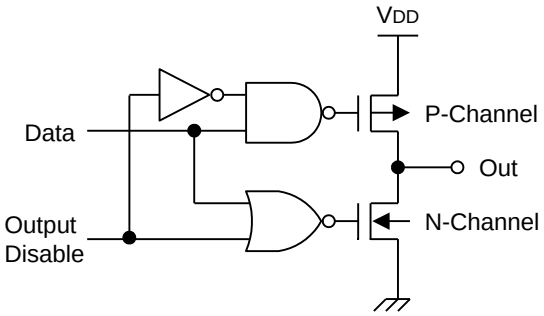


Figure 1-6 Pin Circuit Type C

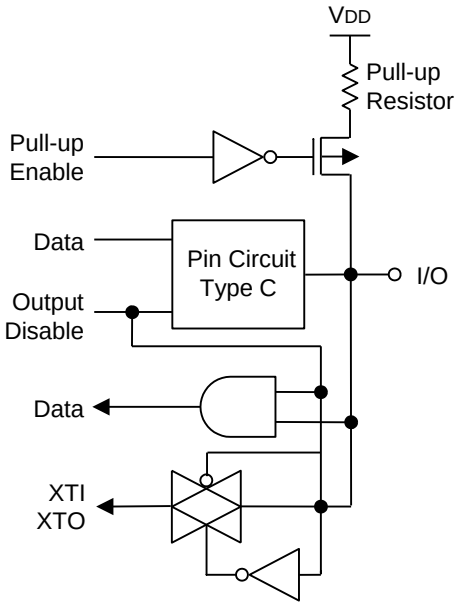


Figure 1-7 Pin Circuit Type D-2 (P5.6 – P5.7)

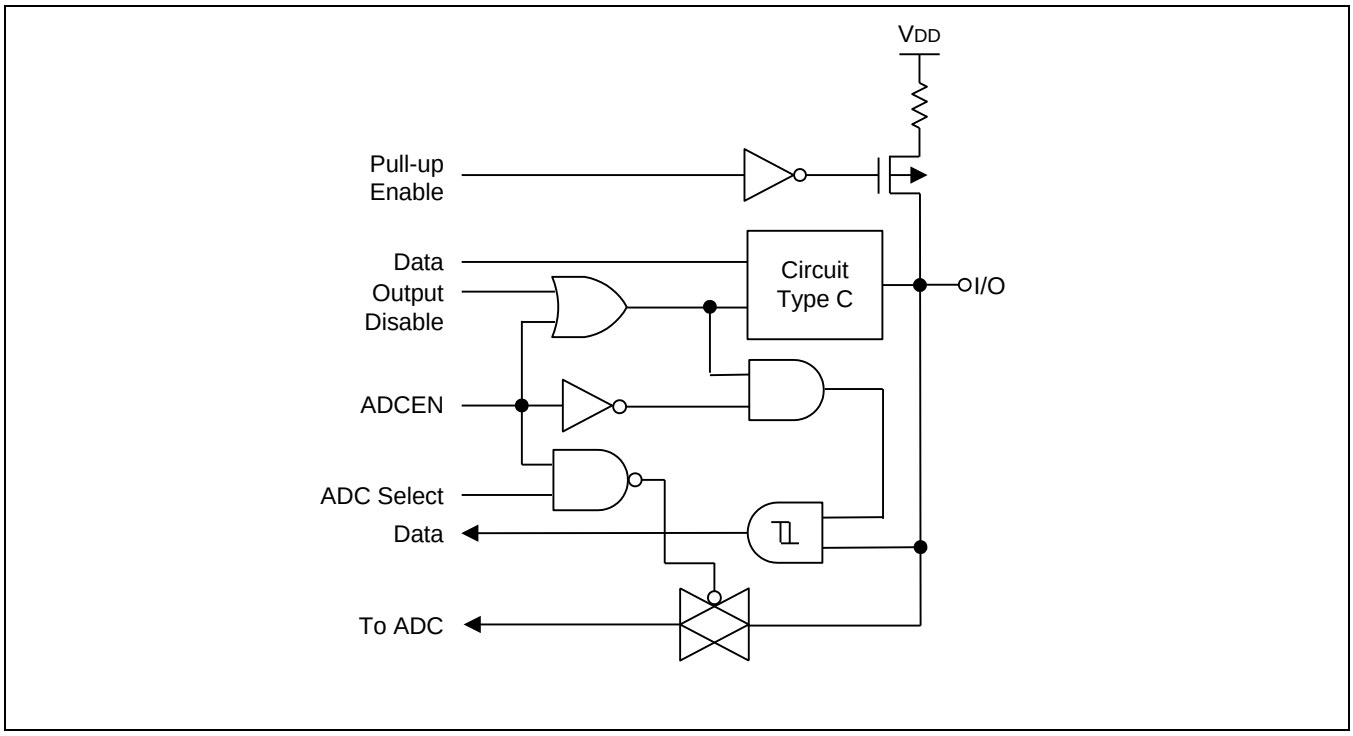


Figure 1-8 Pin Circuit Type F-16 (P4)

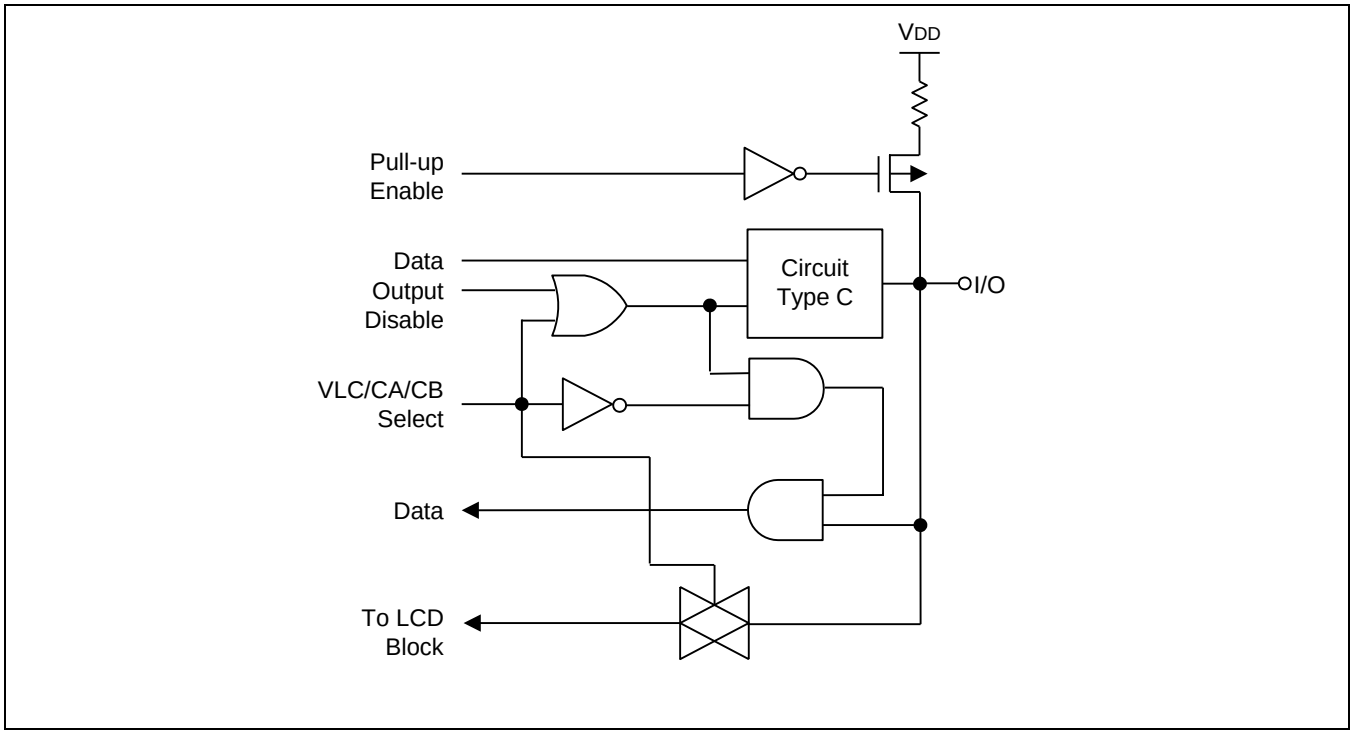


Figure 1-9 Pin Circuit Type F-17 (P5.0 - P5.5)

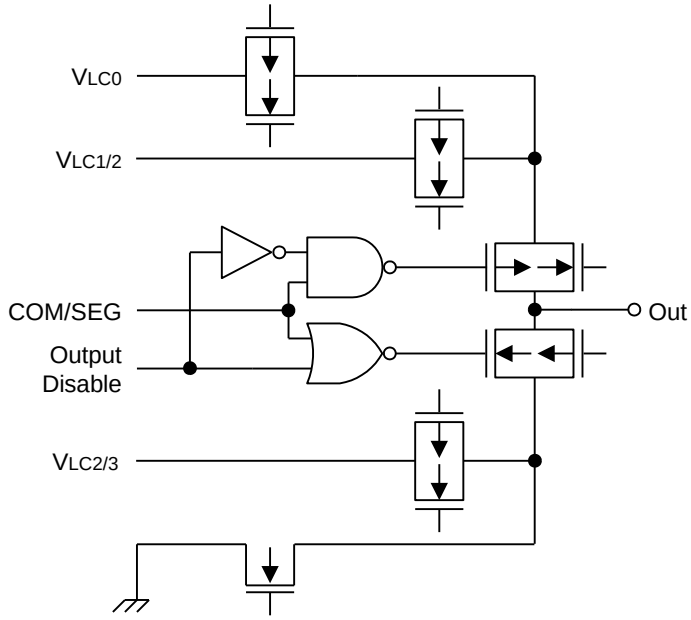


Figure 1-10 Pin Circuit Type H-39

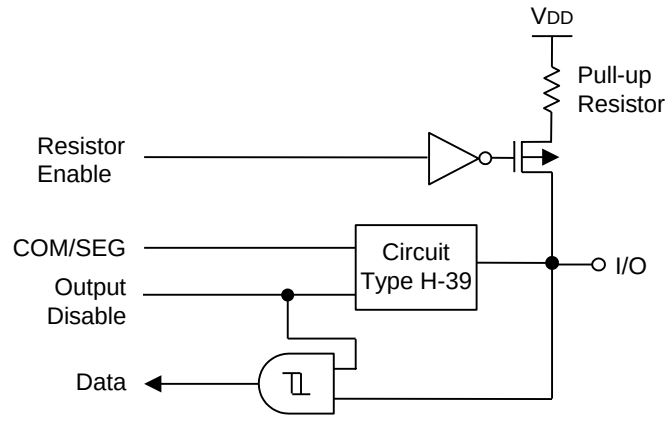


Figure 1-11 Pin Circuit Type H-43 (P2)

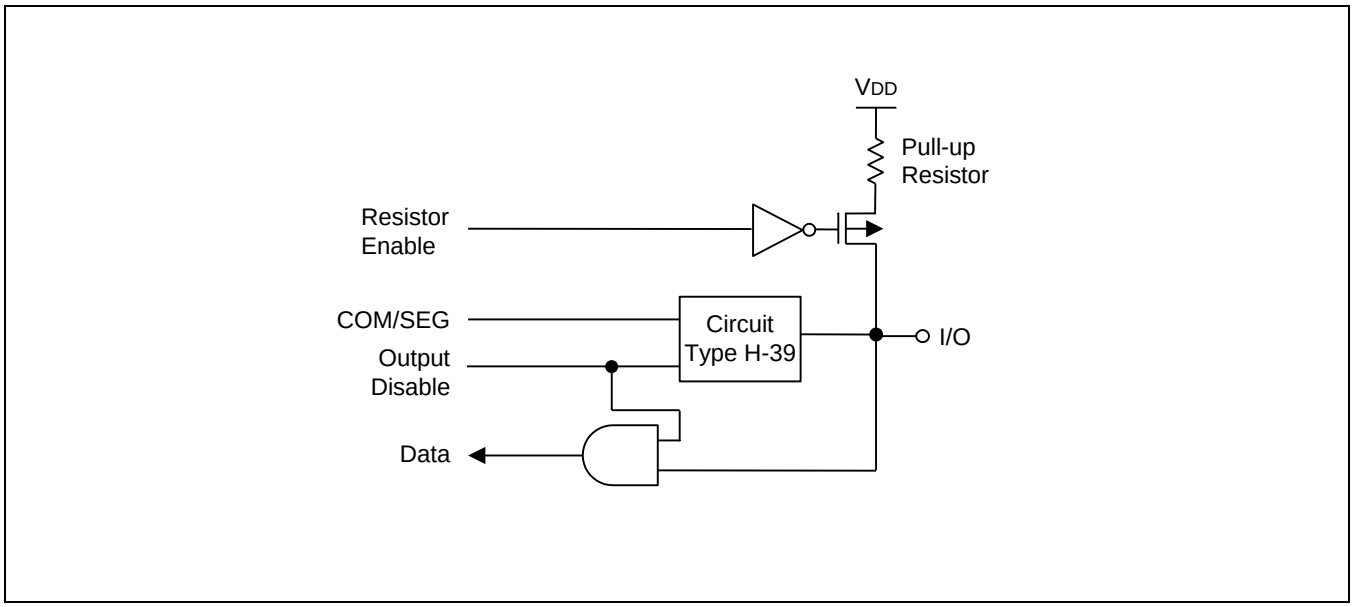


Figure 1-12 Pin Circuit Type H-44 (P0)

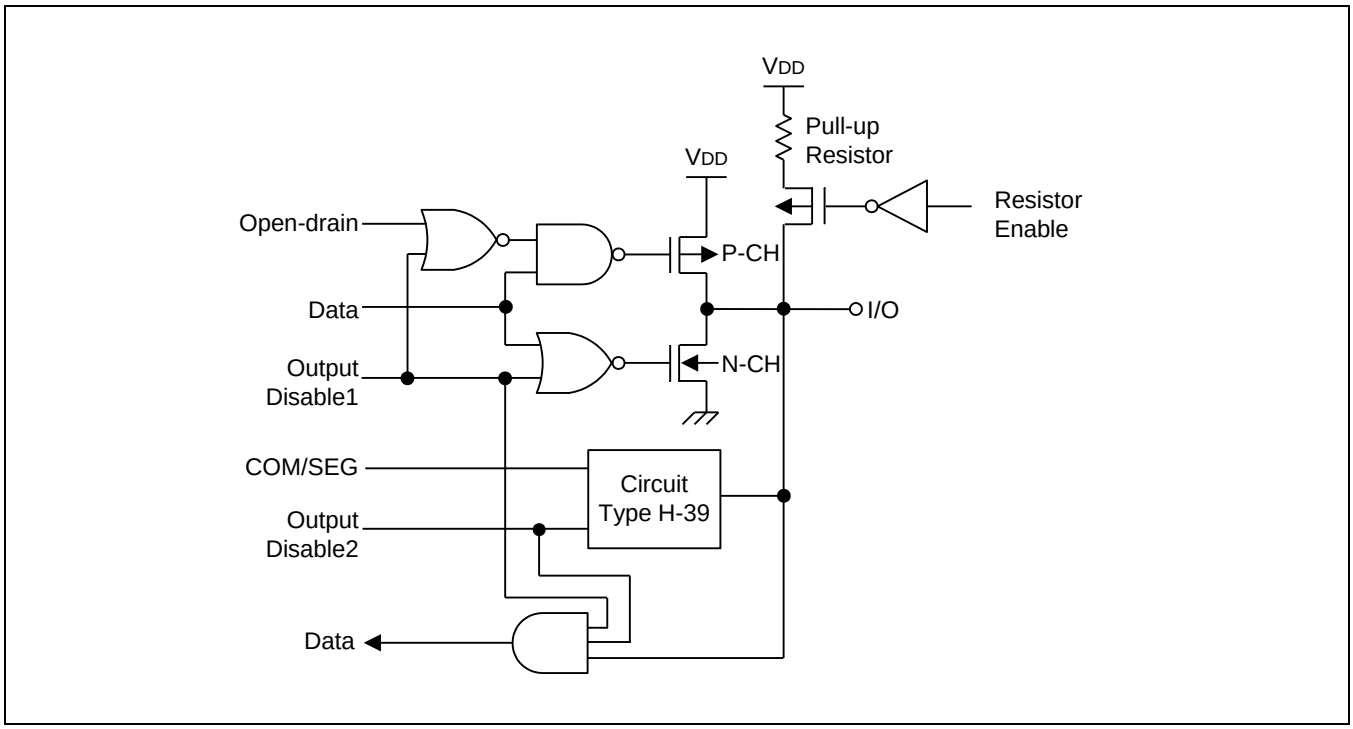


Figure 1-13 Pin Circuit Type H-42 (P1, P3, P6)

2 Address Spaces

2.1 Overview

The S3F8S6B microcontroller has two types of address space:

- Internal program memory (ROM)
- Internal register file

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

The S3F8S6B has an internal 64KB Flash ROM.

The 256 byte physical register space is expanded into an addressable area of 320 bytes using addressing modes.

A 34 byte LCD display register file is implemented.

2.2 Program Memory (ROM)

Program memory (ROM) stores program codes or table data. The S3F8S6B has 64KB internal Flash program memory.

The first 256 bytes of the ROM (0H–0FFH) are reserved for interrupt vector addresses. Unused locations in this address range can be used as normal program memory. If you use the vector address area to store a program code, be careful not to overwrite the vector addresses stored in these locations.

The ROM address at which a program execution starts after a reset is 0100H in the S3F8S6B.

The reset address of ROM can be changed by a smart option in the S3F8S6B (Full-Flash Device). Refer to the chapter 21. Embedded Flash Memory Interface for more detail contents.

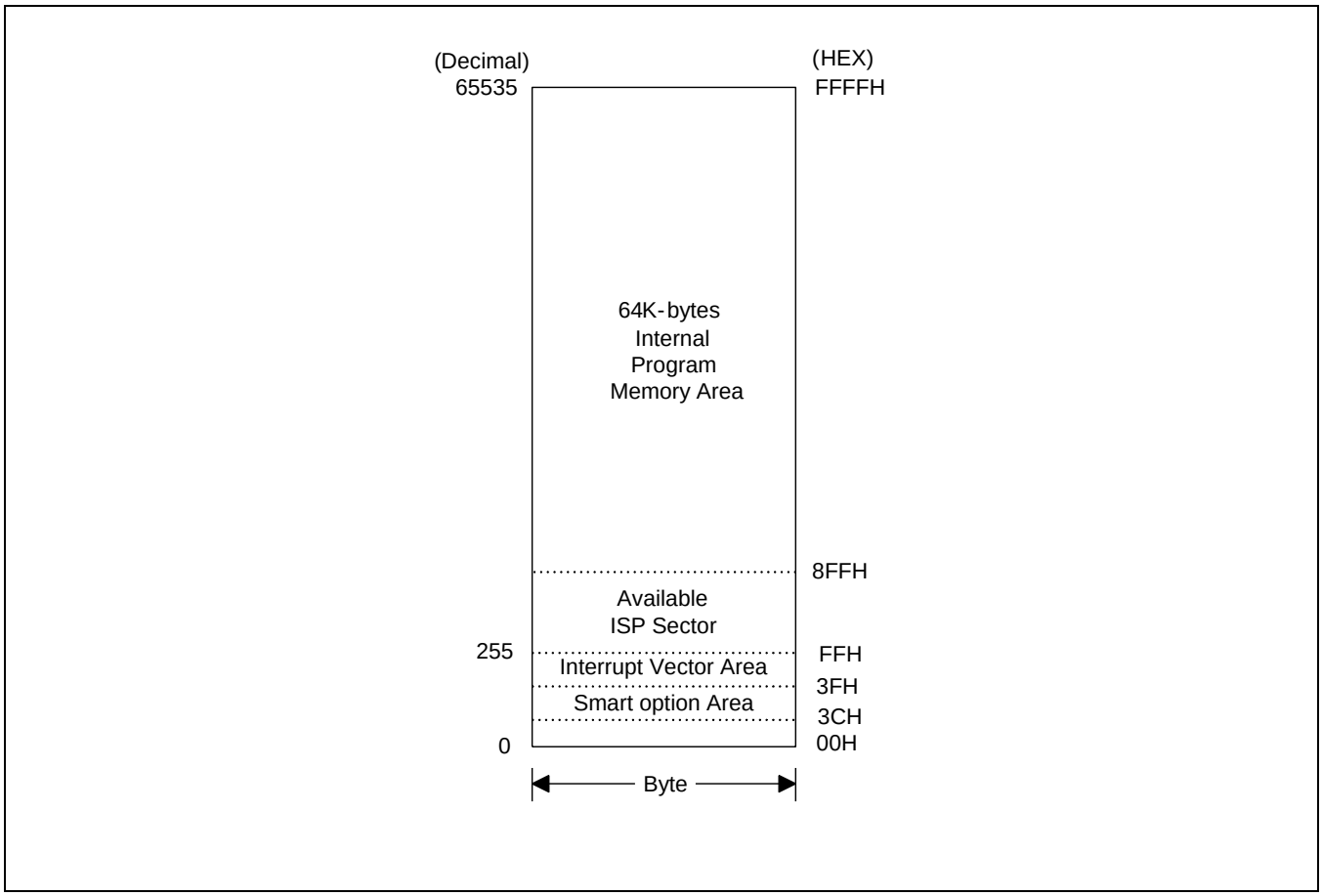
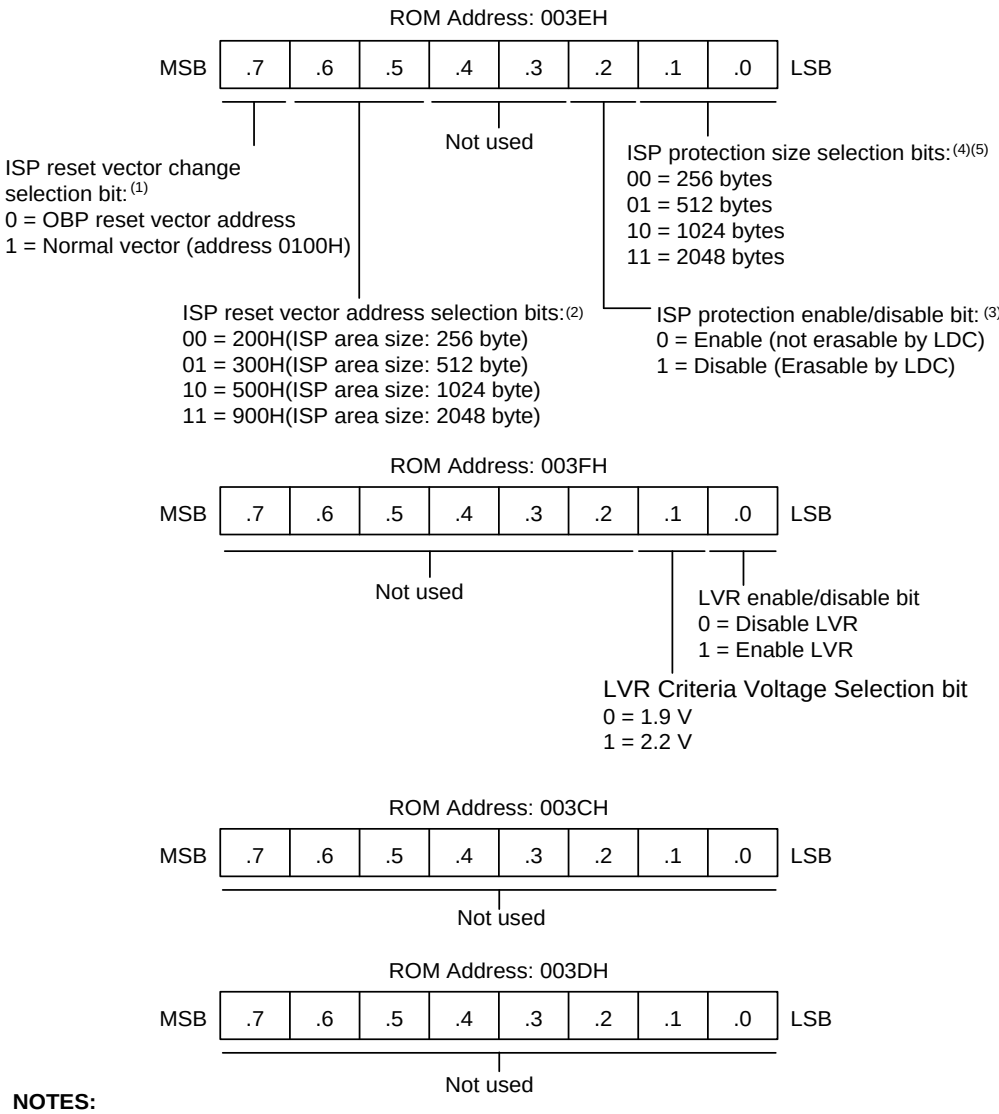


Figure 2-1 Program Memory Address Space

2.2.1 Smart Option



NOTES:

1. By setting ISP reset vector change selection bit (3EH.7) to '0', user can have the available ISP area. If ISP reset vector change selection bit (3EH.7) is '1', 3EH.6 and 3EH.5 are meaningless.
2. If ISP reset vector change selection bit (3EH.7) is '0', user must change ISP reset vector address from 0100H to some address which user want to set reset address (0200H, 0300H, 0500H or 0900H). If the reset vector address is 0200H, the ISP area can be assigned from 0100H to 01FFH (256bytes). If 0300H, the ISP area can be assigned from 0100H to 02FFH (512bytes). If 0500H, the ISP area can be assigned from 0100H to 04FFH (1024bytes). If 0900H, the ISP area can be assigned from 0100H to 08FFH (2048bytes).
3. If ISP protection enable/disable bit is '0', user can't erase or program the ISP area selected by 3EH.1 and 3EH.0 in flash memory.
4. User can select suitable ISP protection size by 3EH.1 and 3EH.0. If ISP protection enable/disable bit (3EH.2) is '1', 3EH.1 and 3EH.0 are meaningless.
5. After selecting ISP reset vector address in selecting ISP protection size, don't select upper than ISP area size.

Figure 2-2 Smart Option

Smart option is the ROM option for start condition of the chip. The ROM address used by smart option is from 003CH to 003FH. The S3F8S6B only use 003EH to 003FH.

When any values are written in the Smart Option area (003CH-003FH) by LDC instruction, the data of the area may be changed but the Smart Option is not affected. The data for Smart Option should be written in the Smart Option area (003CH-003FH) by OTP/MTP programmer (Writer tools)

2.3 Register Architecture

In the S3F8S6B implementation, the upper 64 byte area of register files is expanded two 64 byte areas, called set 1 and set 2. The upper 32 byte area of set 1 is further expanded two 32 byte register banks (bank 0 and bank 1), and the lower 32 byte area is a single 32 byte common area.

In case of S3F8S6B the total number of addressable 8-bit registers is 2,192. Of these 2,192 registers, 13 bytes are for CPU and system control registers, 34 bytes are for LCD data registers, 81 bytes are for peripheral control and data registers, 16 bytes are used as a shared working registers, and 2,048 registers are for general-purpose use, page 0-page 7.

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

The extension of 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 S3F8S6B Register Type Summary

Register Type	Number of Bytes
General-purpose registers (including the 16 byte common working register area, eight 192 byte prime register area, and eight 64 byte set 2 area) LCD data registers CPU and system control registers Mapped clock, peripheral, I/O control, and data registers	2,064 34 13 81
Total Addressable Bytes	2,192

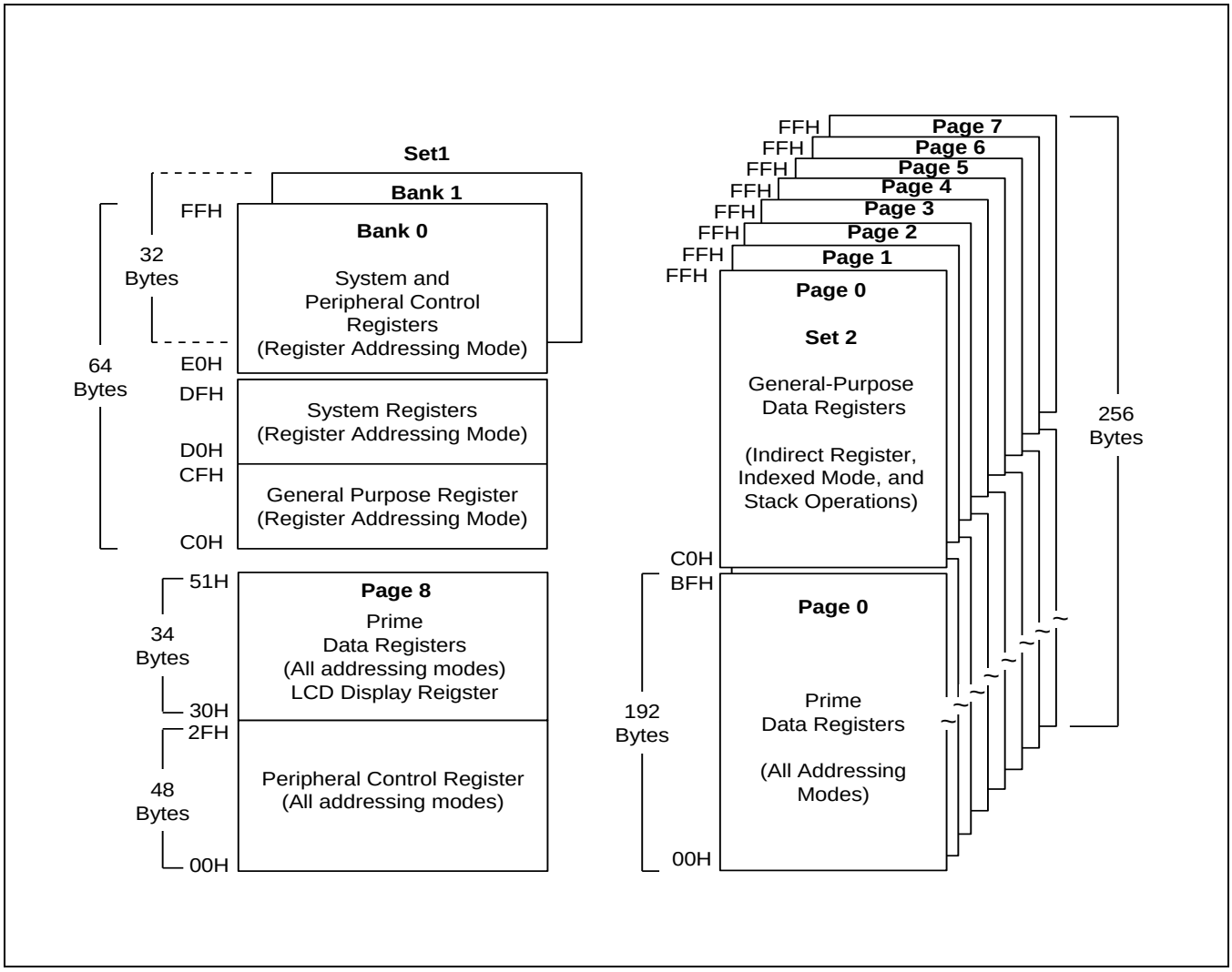


Figure 2-3 Internal Register File Organization (S3F8S6B)

2.3.1 Register Page Pointer (PP)

The S3C8-series architecture supports the logical expansion of the physical 256 byte internal register file (using an 8-bit data bus) into as many as 16 separately addressable register pages. Page addressing is controlled by the register page pointer (PP, DFH). In the S3F8S6B microcontroller, a paged register file expansion is implemented for LCD data registers, and the register page pointer must be changed to address other pages.

After a reset, the page pointer's source value (lower nibble) and the destination value (upper nibble) are always "0000", automatically selecting page 0 as the source and destination page for register addressing.

Register Page Pointer (PP)
DFH ,Set 1, R/W

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

Destination register page selection bits:	
0000	Destination: Page 0
0001	Destination: Page 1
0010	Destination: Page 2
0011	Destination: Page 3
0100	Destination: Page 4
0101	Destination: Page 5
0110	Destination: Page 6
0111	Destination: Page 7
1000	Destination: Page 8
Others	Not used for the S3F8S6B

Source register page selection bits:	
0000	Source: page 0
0001	Source: page 1
0010	Source: page 2
0011	Source: page 3
0100	Source: page 4
0101	Source: page 5
0110	Source: page 6
0111	Source: page 7
1000	Source: page 8
Others	Not used for the S3F8S6B

NOTES:
1. In the S3F8S6B microcontroller, the internal register file is configured as nine pages (Pages 0-7, 8). The pages 0-7 are used for general purpose register file.
2. The page 8 of S3F8S6B is used for LCD data register (30H~51H) and system register (00H~2FH).

Figure 2-4 Register Page Pointer (PP)

Example 2-1 Using the Page Pointer for RAM clear (Page 0, Page 1)

```
              LD      PP, #00H        ;       Destination ← 0, Source ←← 0
              SRP     #0C0H
              LD      R0,#0FFH        ;       Page 0 RAM clear starts
RAMCL0        CLR     @R0
              DJNZ    R0,RAMCL0
              CLR     @R0     ;       R0 = 00H

              LD      PP,#10H ;       Destination ← 1, Source ← 0
              LD      R0,#0FFH        ;       Page 1 RAM clear starts
RAMCL1        CLR     @R0
              DJNZ    R0,RAMCL1
              CLR     @R0     ;       R0 = 00H
```

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

2.3.2 Register Set 1

The term set 1 refers to the upper 64 bytes of the register file, locations C0H–FFH.

The upper 32 byte area of this 64 byte space (E0H–FFH) is expanded two 32 byte register banks, bank 0 and bank 1. The set register bank instructions, SB0 or SB1, are used to address one bank or the other. A hardware reset operation always selects bank 0 addressing.

The upper two 32 byte areas (bank 0 and bank 1) of set 1 (E0H–FFH) contains 56 mapped system and peripheral control registers. The lower 32 byte area contains 16 system registers (D0H–DFH) and a 16 byte common working register area (C0H–CFH). You can use the common working register area as a "scratch" area for data operations being performed in other areas of the register file.

Registers in set 1 location are directly accessible at all times using Register addressing mode. The 16 byte working register area can only be accessed using working register addressing (For more information about working register addressing, please refer to Chapter 3, "Addressing Modes.")

2.3.3 Register Set 2

The same 64 byte physical space that is used for set 1 location C0H–FFH is logically duplicated to add another 64 bytes of register space. This expanded area of the register file is called set 2. For the S3F8S6B, the set 2 address range (C0H–FFH) is accessible on pages 0-7.

The logical division of set 1 and set 2 is maintained by means of addressing mode restrictions. You can use only Register addressing mode to access set 1 location. In order to access registers in set 2, you must use Register Indirect addressing mode or Indexed addressing mode.

The set 2 register area of page 0 is commonly used for stack operations.

2.3.4 Prime Register Space

The lower 192 bytes (00H–BFH) of the S3F8S6B's eight/four/two 256 byte register pages is called prime register area. Prime registers can be accessed using any of the seven addressing modes
(Refer to Chapter 3, "Addressing Modes.")

The prime register area on page 0 is immediately addressable following a reset. In order to address prime registers on pages 0, 1, 2, 3, 4, 5, 6, 7, or 8 you must set the register page pointer (PP) to the appropriate source and destination values.

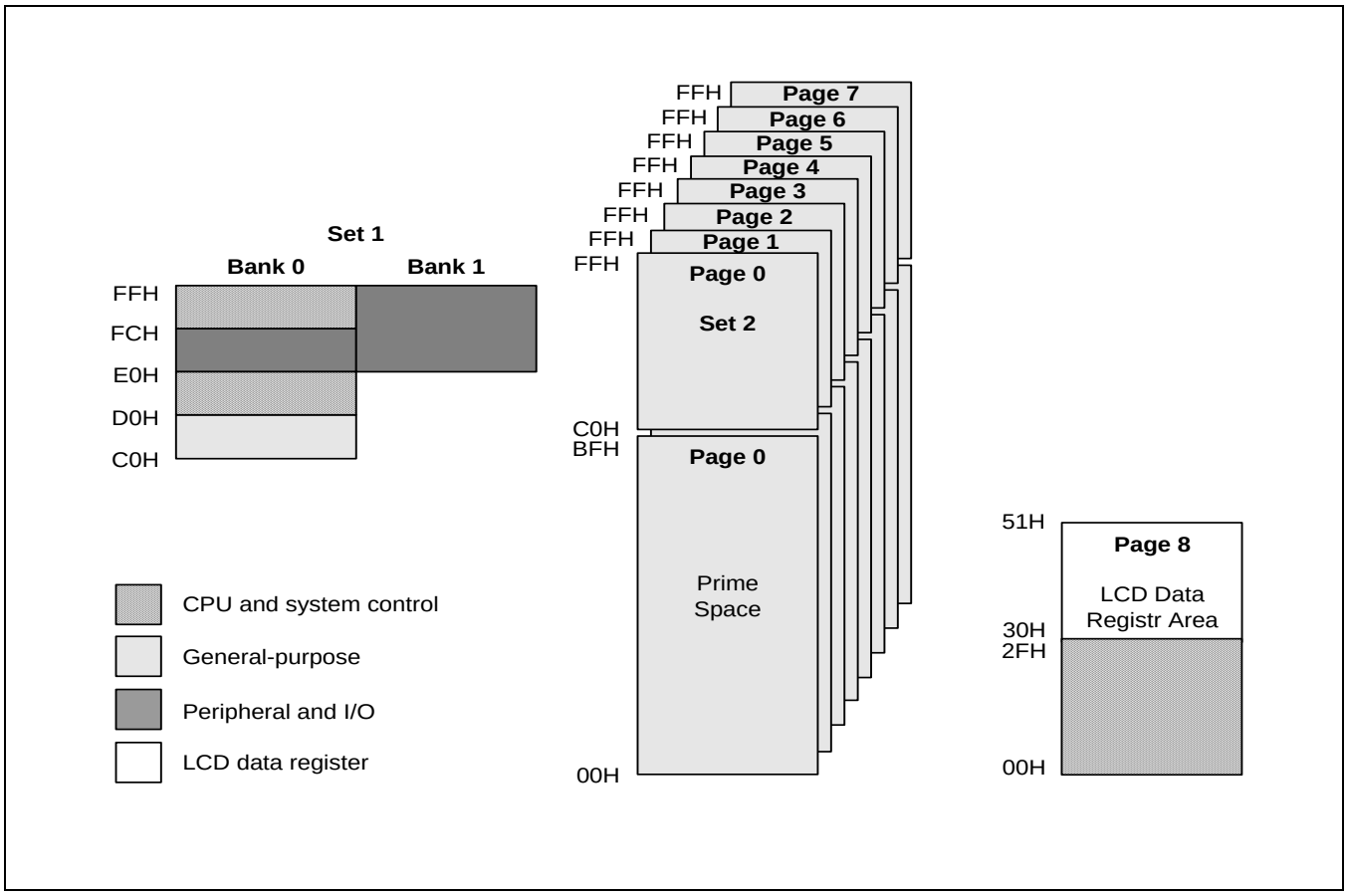


Figure 2-5 Set 1, Set 2, Prime Area Register, and LCD Data Register Map

2.3.5 Working Registers

Instructions can access specific 8-bit registers or 16-bit register pairs using either 4-bit or 8-bit address fields. When 4-bit working register addressing is used, the 256 byte register file can be seen by the programmer as one that consists of 32 8 byte register groups or "slices." Each slice comprises of eight 8-bit registers.

Using the two 8-bit register pointers, RP1 and RP0, two working register slices can be selected at any one time to form a 16 byte working register block. Using the register pointers, you can move this 16 byte register block anywhere in the addressable register file, except 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 register pointers RP0 and RP1.

After a reset, RP0 and RP1 always point to the 16 byte common area in set 1 (C0H–CFH).

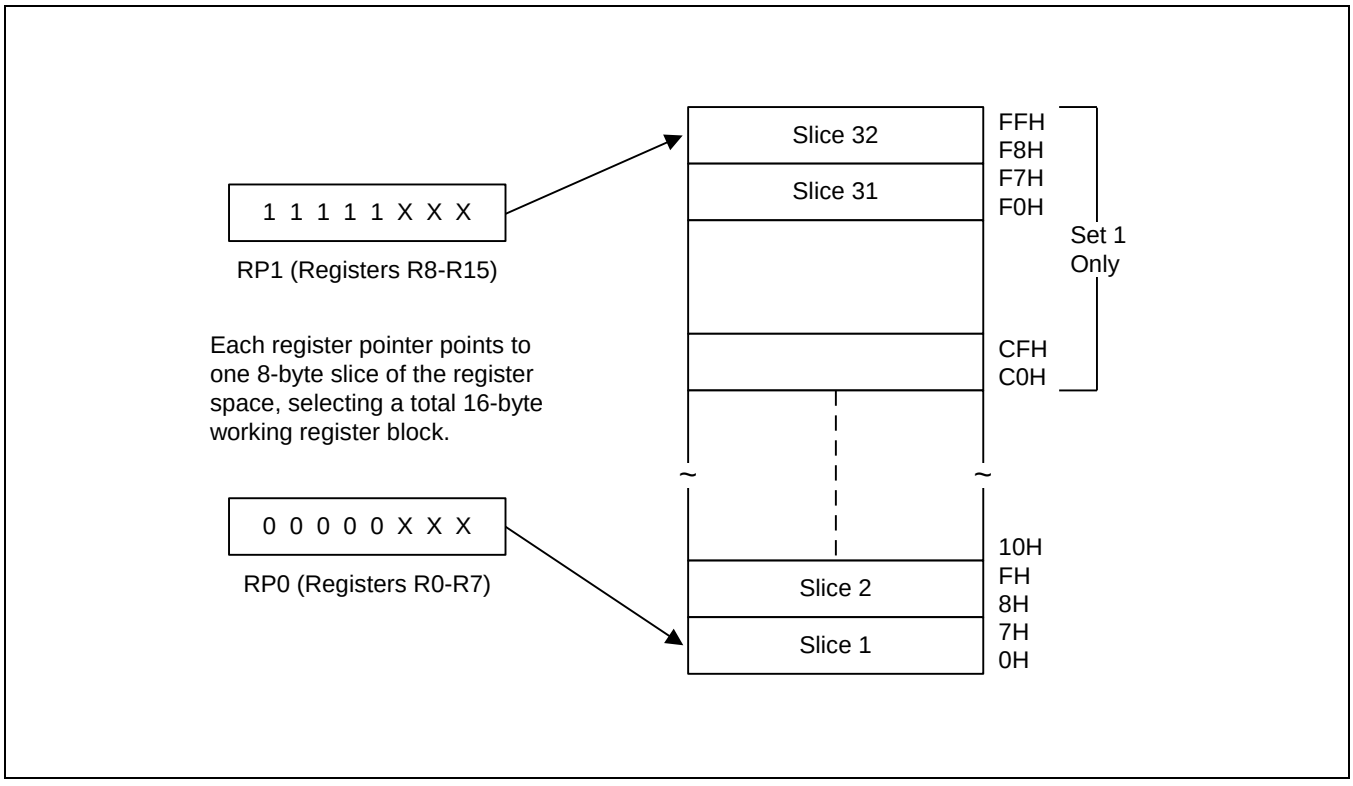


Figure 2-6 8 byte Working Register Areas (Slices)

2.3.6 Using the Register Points

Register pointers RP0 and RP1, mapped to addresses D6H and D7H in set 1, are used to select two movable 8 byte working register slices in the register file. After a reset, they point to the working register common area: RP0 points to addresses C0H–C7H, and RP1 points to addresses C8H–CFH.

To change a register pointer value, you load a new value to RP0 and/or RP1 using an SRP or LD instruction. (see *Figure 2-7* and *Figure 2-8*).

With working register addressing, you can only access those two 8-bit slices of the register file that are currently pointed to by RP0 and RP1. You cannot, however, use the register pointers to select a working register space in set 2, C0H–FFH, because these locations can be accessed only using the Indirect Register or Indexed addressing modes.

The selected 16 byte working register block usually consists of two contiguous 8 byte slices. As a general programming guideline, it is recommended 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 to support program requirements.

Example 2-2 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
```

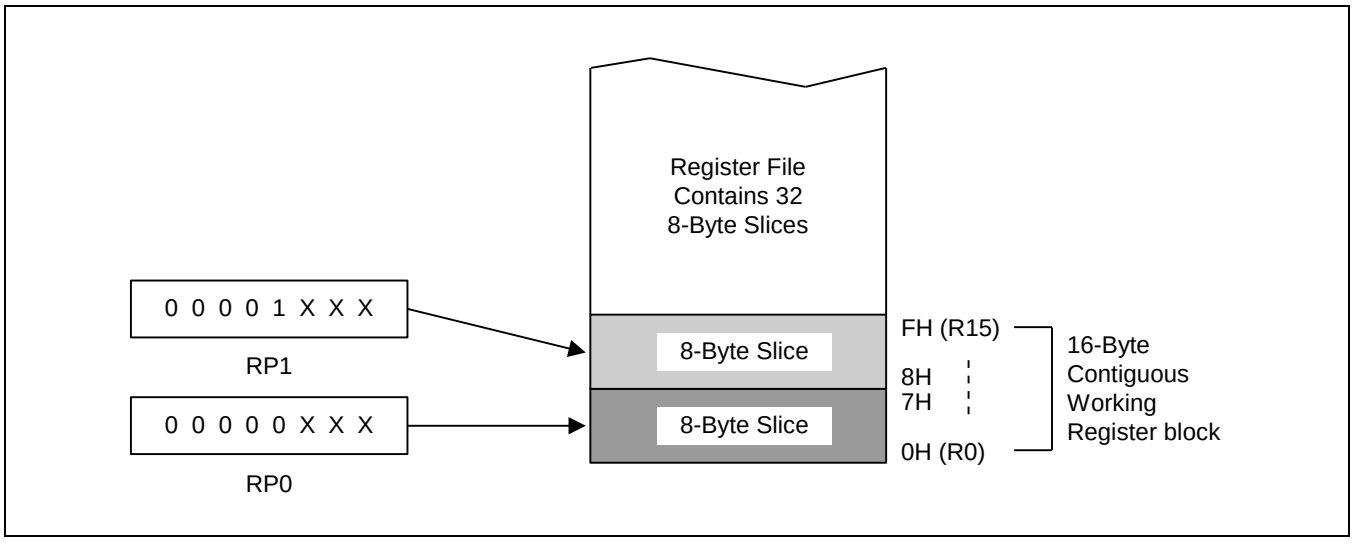


Figure 2-7 Contiguous 16 byte Working Register Block

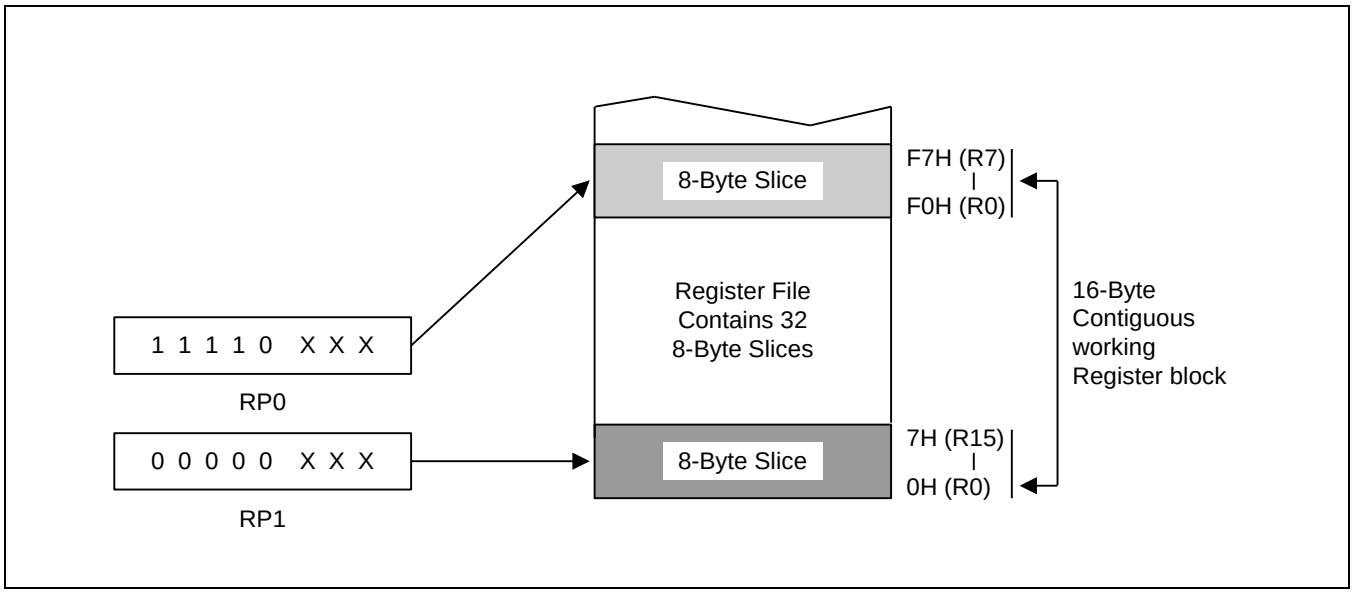


Figure 2-8 Non-Contiguous 16 byte Working Register Block

Example 2-3 Using the RPs to Calculate the Sum of a Series of Registers

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

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

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

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

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

2.4 Register Addressing

The S3C8-series register architecture provides an efficient method of working register addressing that takes full advantage of shorter instruction formats to reduce execution time.

With Register (R) addressing mode, in which the operand value is the content of a specific register or register pair, you can access any location in the register file except for set 2. With working register addressing, you use a register pointer to specify an 8 byte working register space in the register file and an 8-bit register within that space.

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

Working register addressing differs from Register addressing as it uses a register pointer to identify a specific 8 byte working register space in the internal register file and a specific 8-bit register within that space.

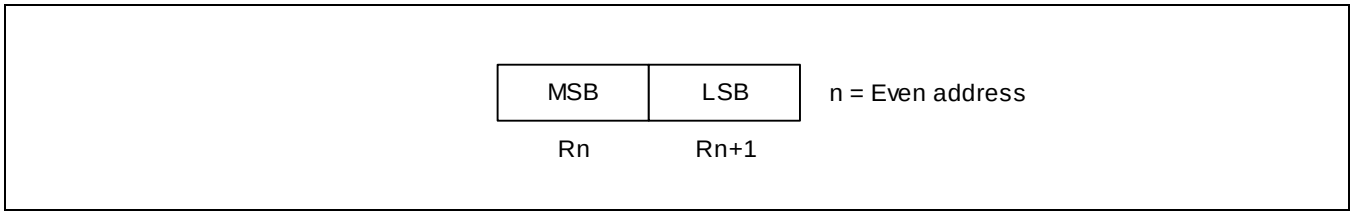


Figure 2-9 16-Bit Register Pair

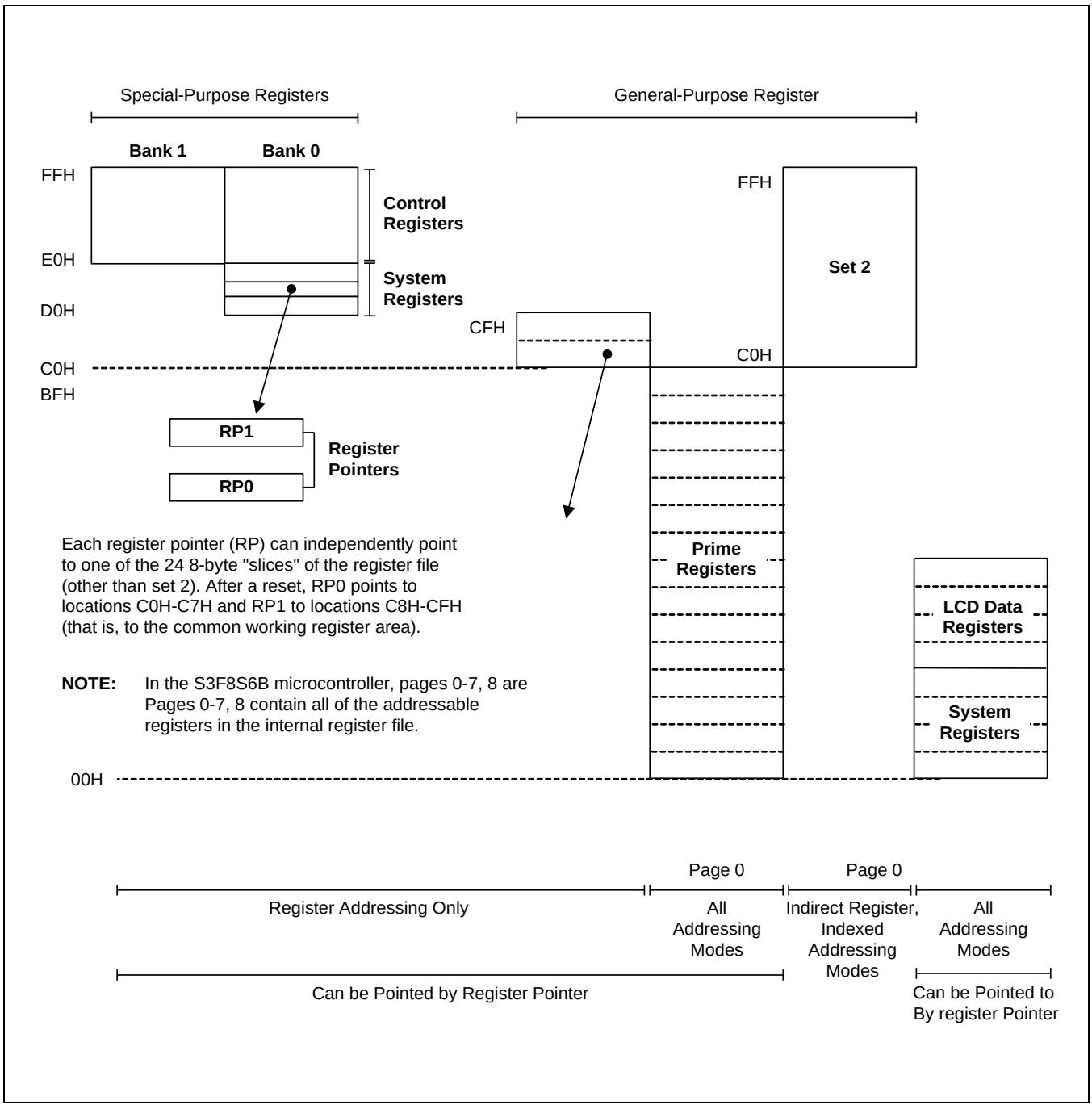


Figure 2-10 Register File Addressing

2.4.1 Common Working Register Area (C0H–CFH)

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

- RP0 → C0H–C7H
- RP1 → C8H–CFH

This 16 byte address range is called common area. That is, locations in this area can be used as working registers by operations that address any location on any page in the register file. Typically, these working registers serve as temporary buffers for data operations between different pages.

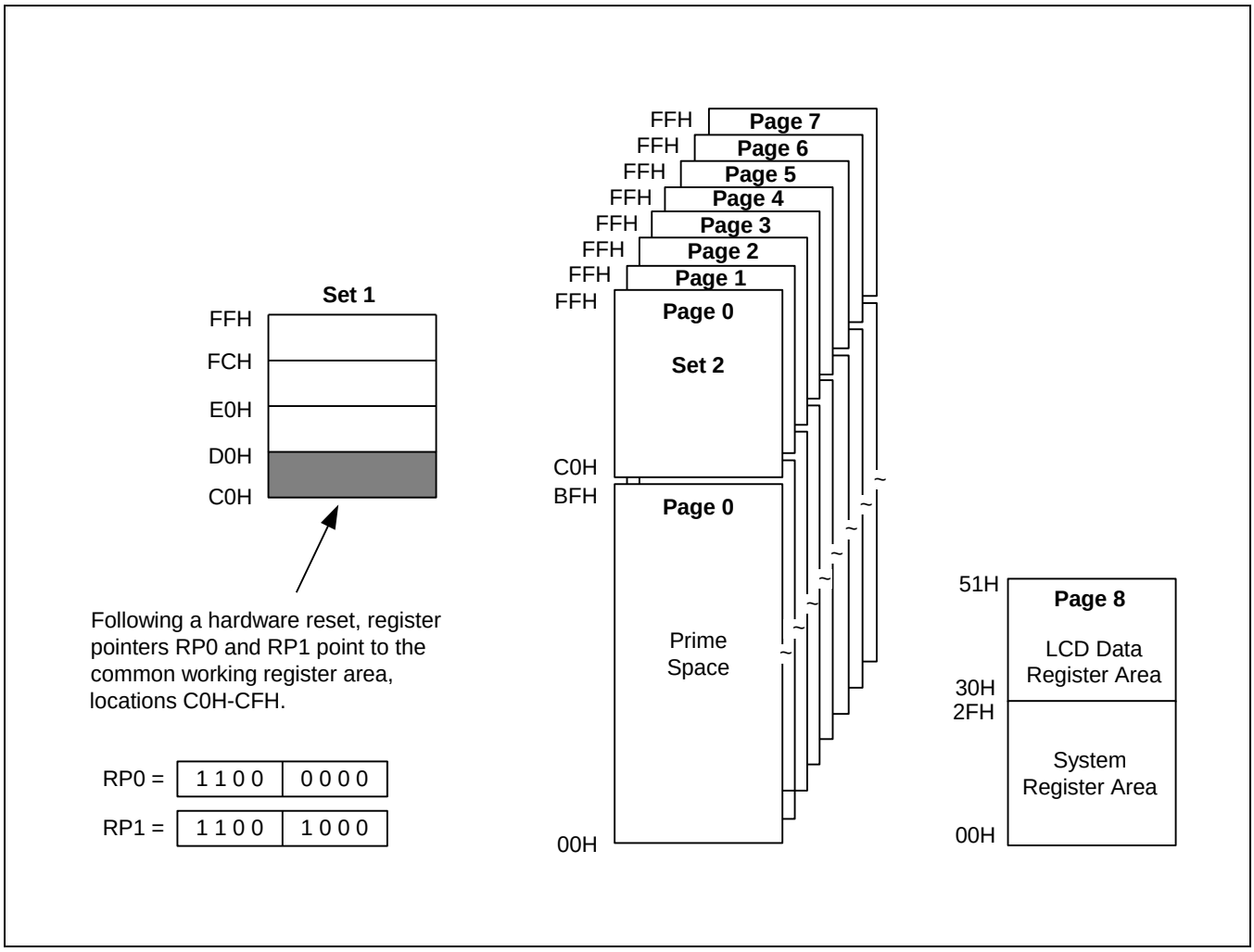


Figure 2-11 Common Working Register Area

Example 2-4 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.

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

Use working register addressing instead:

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

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

Use working register addressing instead:

```
SRP     #0C0H
ADD     R3,#45H         ;       R3 (C3H)  R3 + 45H
```

2.4.2 4-bit Working Register Addressing

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

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

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

Figure 2-13 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).

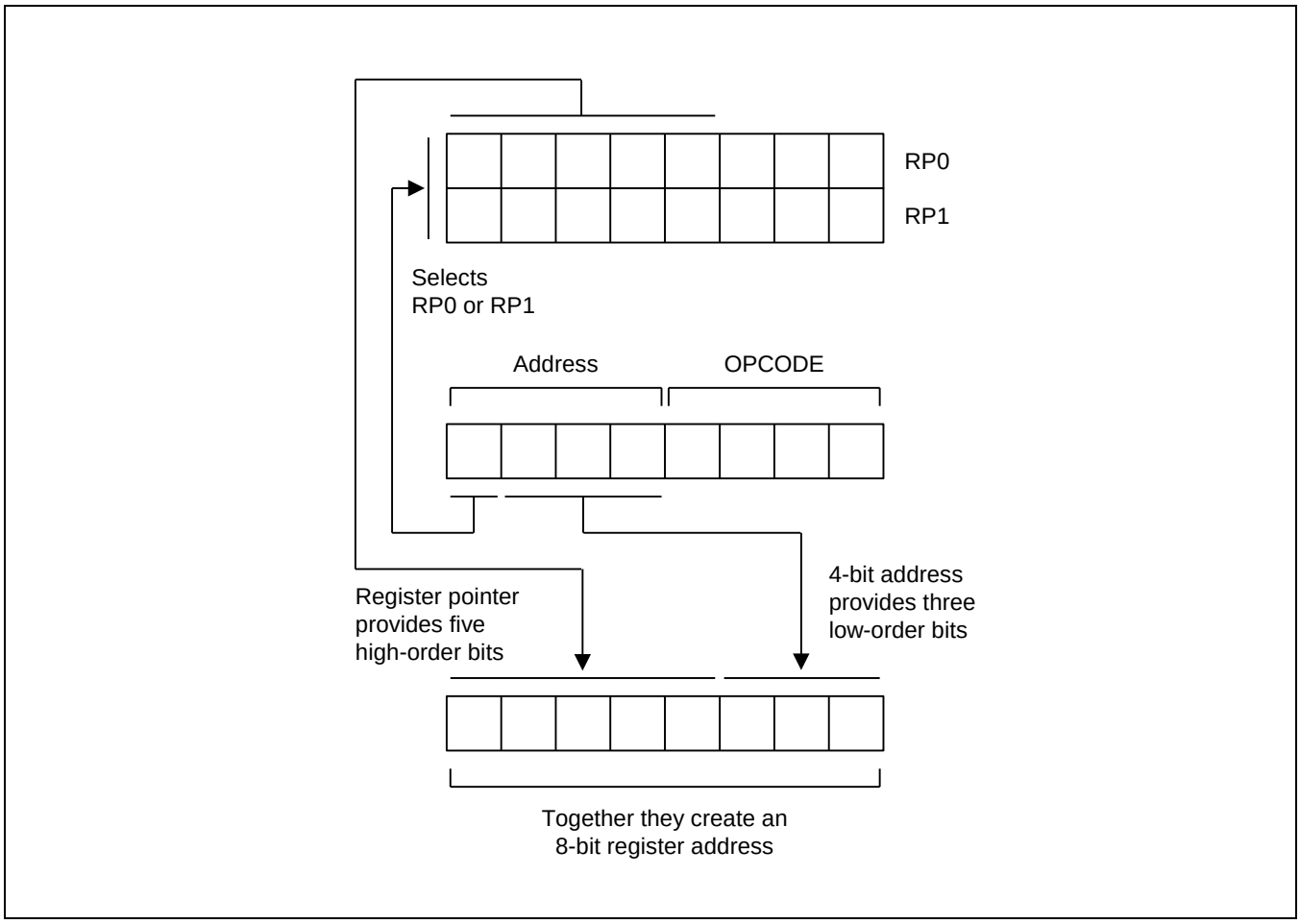


Figure 2-12 4-bit Working Register Addressing

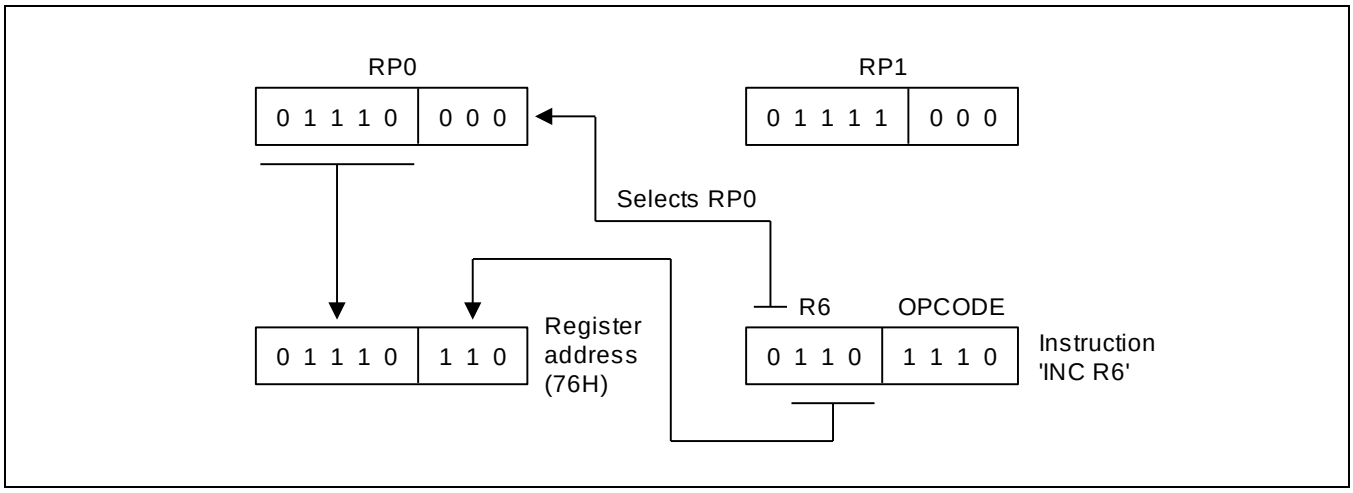


Figure 2-13 4-bit Working Register Addressing Example

2.4.3 8-bit Working Register Addressing

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

As shown in *Figure 2-14*, 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; 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. Bit 4 ("1") selects RP1 and the five high-order bits in RP1 (10101B) become the five high-order bits of the register address. The three low-order bits of the register address (011) are provided by the three low-order bits of the 8-bit instruction address. The five address bits from RP1 and the three address bits from the instruction are concatenated to form the complete register address, 0ABH (10101011B).

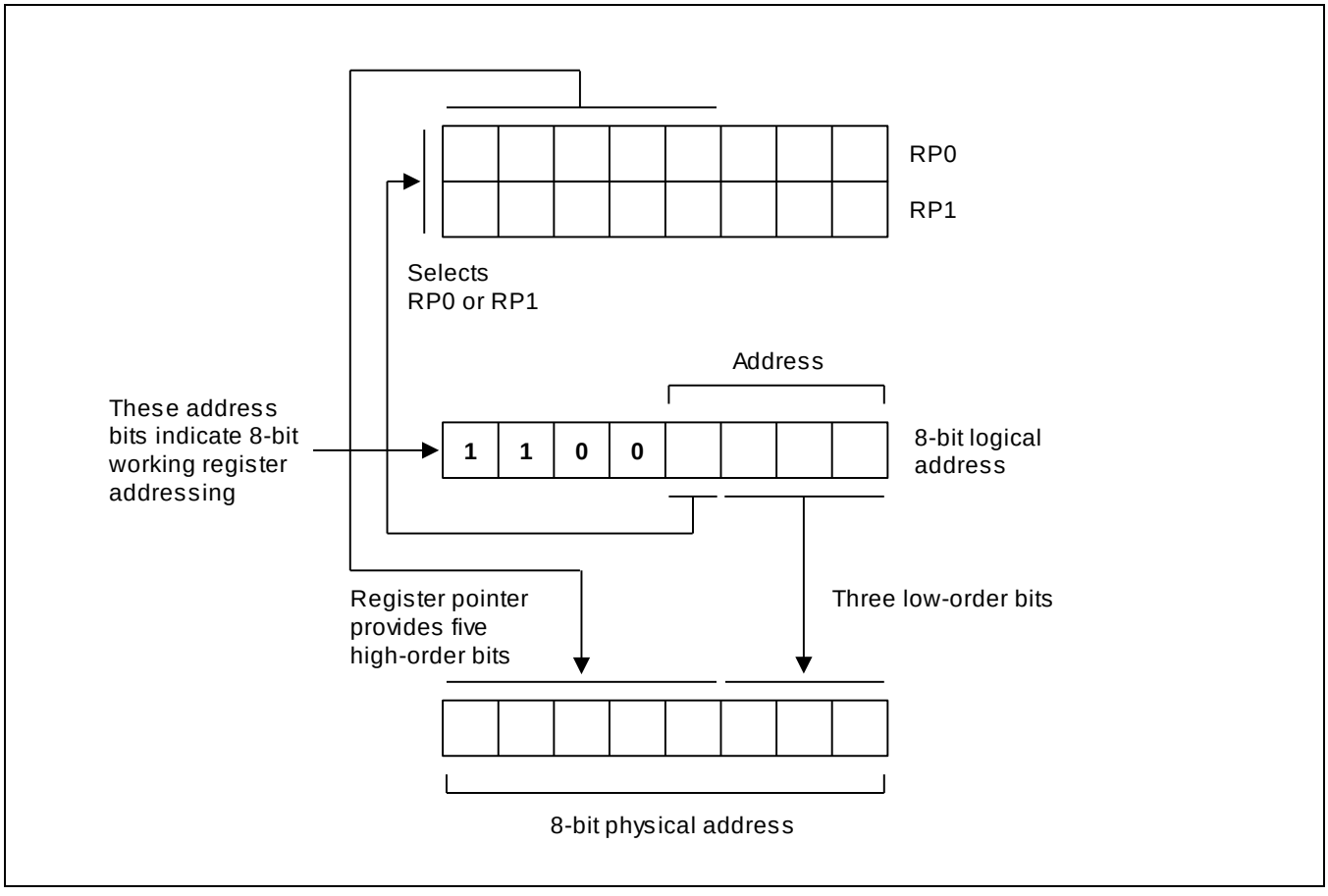


Figure 2-14 8-bit Working Register Addressing

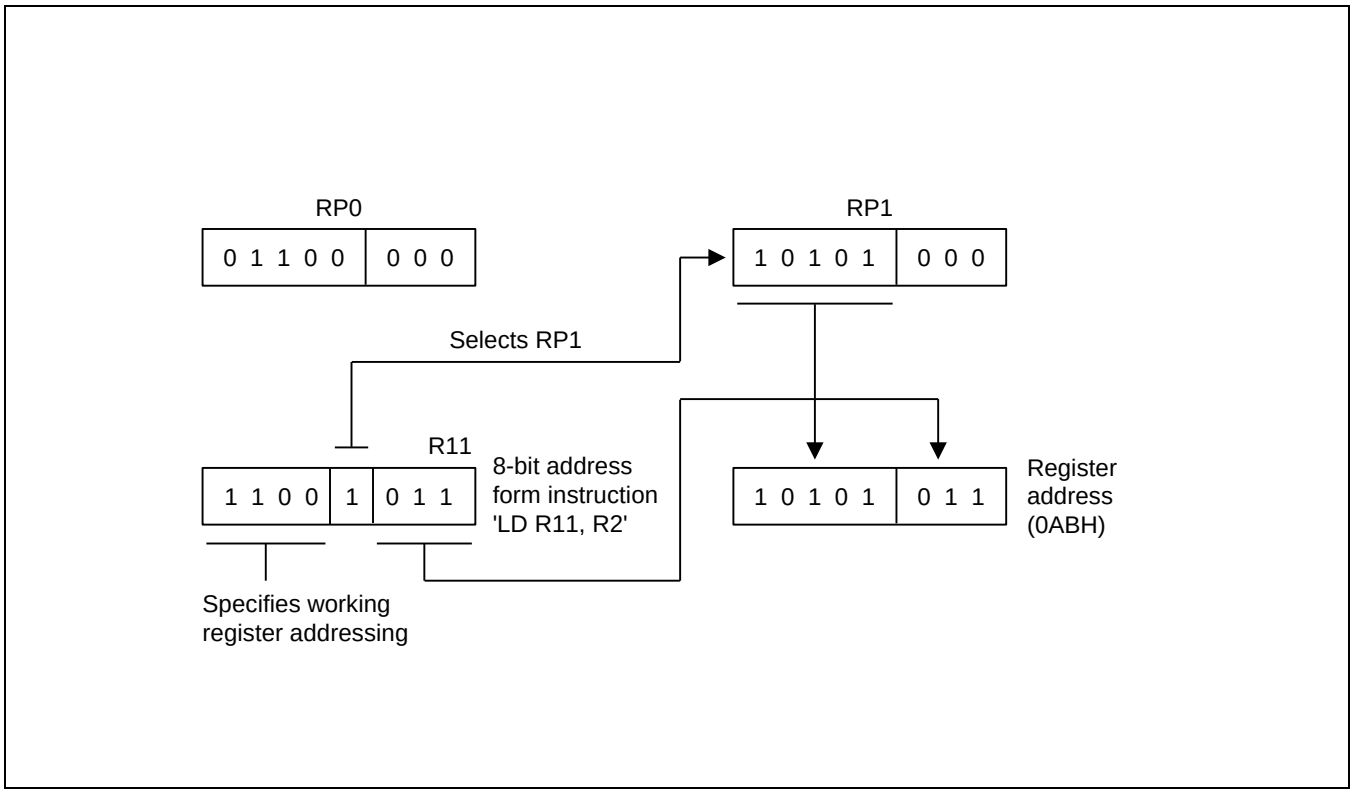


Figure 2-15 8-bit Working Register Addressing Example

2.5 System and User Stack

The S3C8-series microcontrollers use the system stack for data storage, subroutine calls and returns. The PUSH and POP instructions are used to control system stack operations. The S3F8S6B architecture supports stack operations in the internal register file.

2.5.1 Stack Operations

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

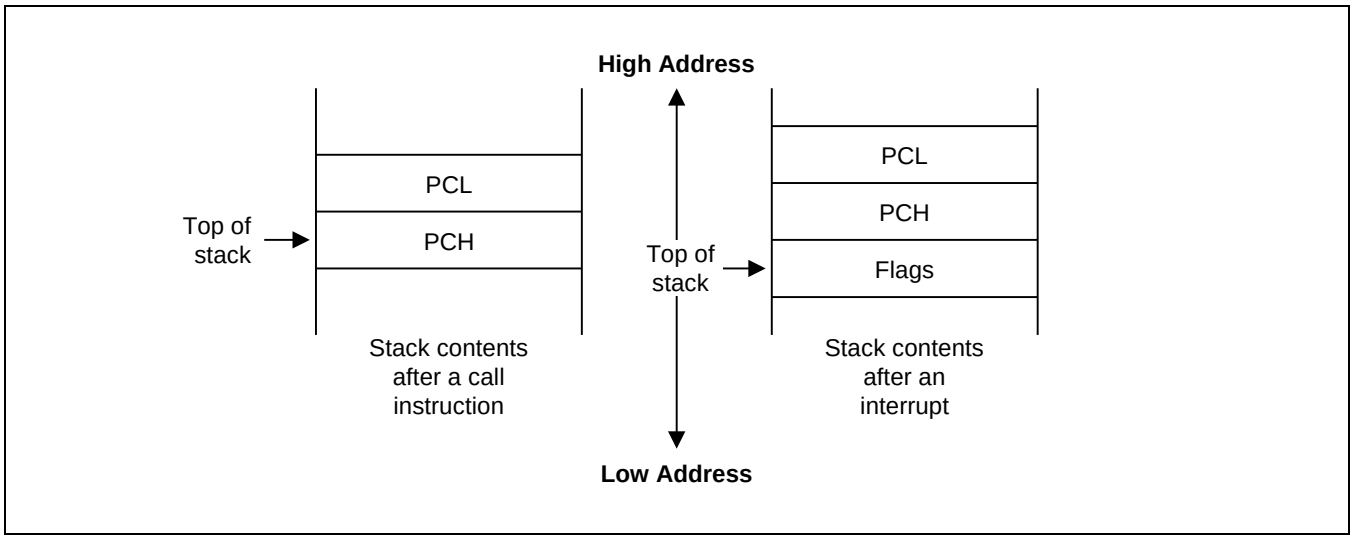


Figure 2-16 Stack Operations

2.5.2 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.

2.5.3 Stack Pointers (SPL, SPH)

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

Because only internal memory space is implemented in the S3F8S6B, the SPL must be initialized to an 8-bit value in the range 00H–FFH. The SPH register is not needed and can be used as a general-purpose register, if necessary.

When the SPL register contains the only stack pointer value (that is, when it points to a system stack in the register file), you can use the SPH register as a general-purpose data register. However, if an overflow or underflow condition occurs as a result of increasing or decreasing the stack address value in the SPL register during normal stack operations, the value in the SPL register will overflow (or underflow) to the SPH register, overwriting any other data that is currently stored there. To avoid overwriting data in the SPH register, you can initialize the SPL value to "FFH" instead of "00H".

Example 2-5 Standard Stack Operations Using PUSH and POP

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

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

3 Addressing Modes

3.1 Overview

Instructions that are stored in program memory are fetched for execution using the program counter. Instructions indicate the operation to be performed and the data to be operated on. Addressing mode is the method used to determine the location of the data operand. The operands specified in SAM88RC instructions may be condition codes, immediate data, or a location in the register file, program memory, or data memory.

The S3C8-series instruction set supports seven explicit addressing modes. Not all of these addressing modes are available for each instruction. The seven addressing modes and their symbols are:

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

3.2 Register Addressing Mode

In Register addressing mode (R), the operand value is the content of a specified register or register pair (see *Figure 3-1*).

Working register addressing differs from Register addressing in that 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*).

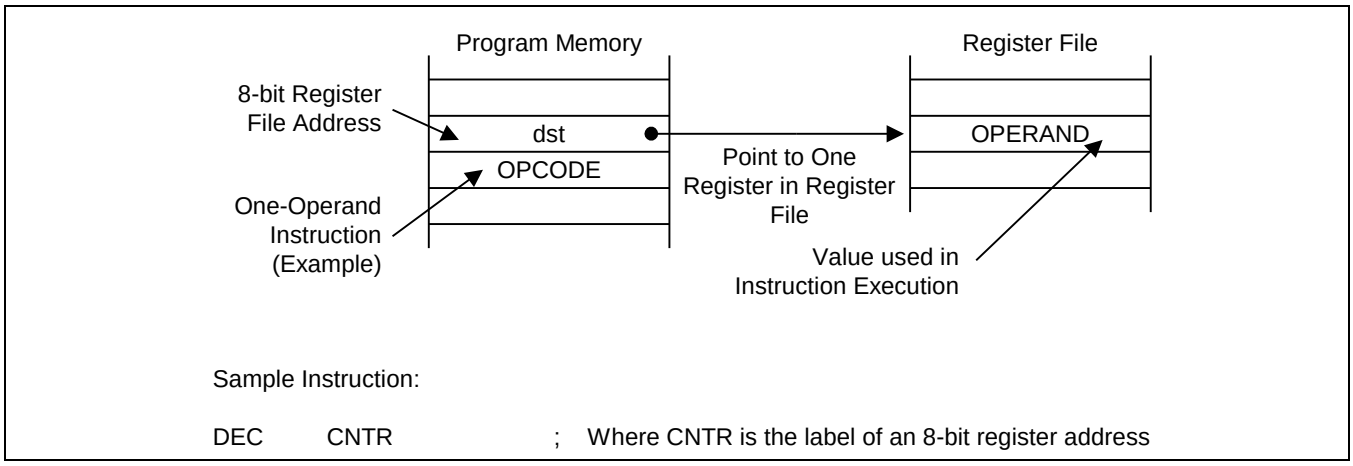


Figure 3-1 Register Addressing

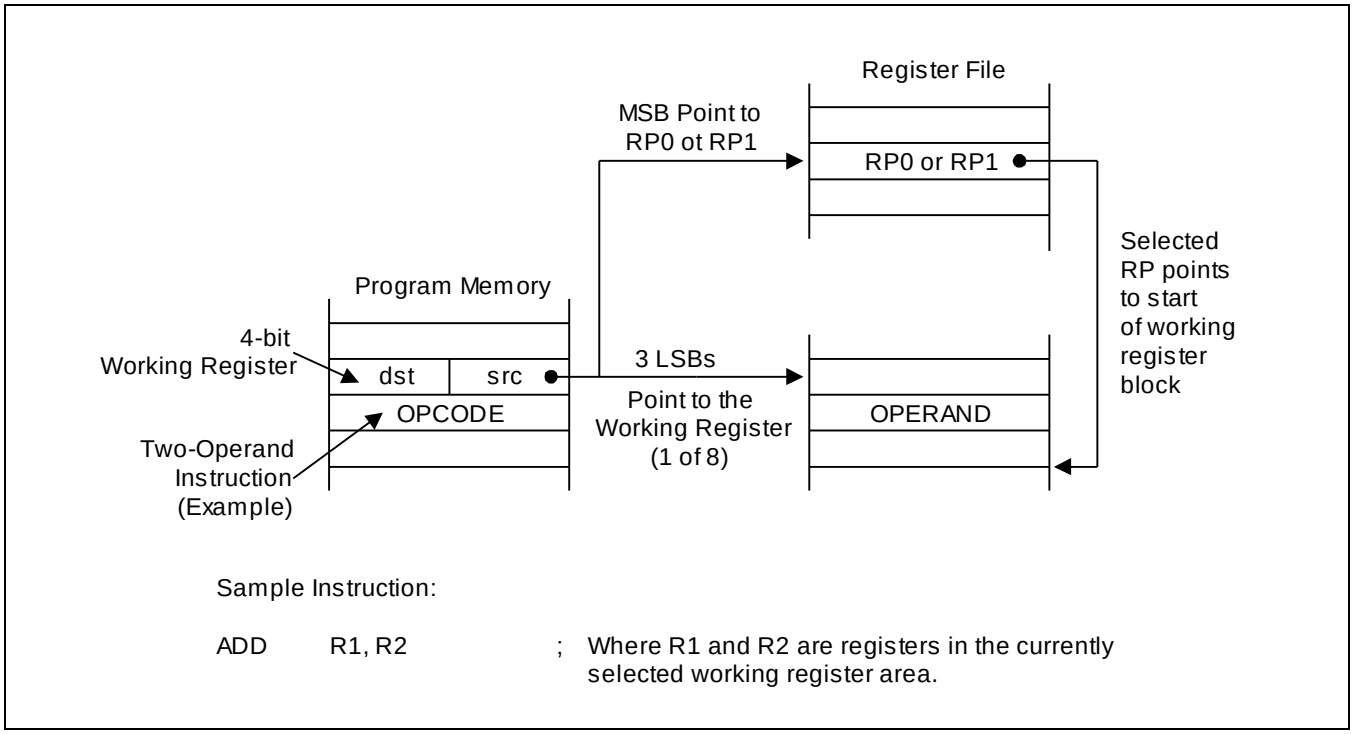
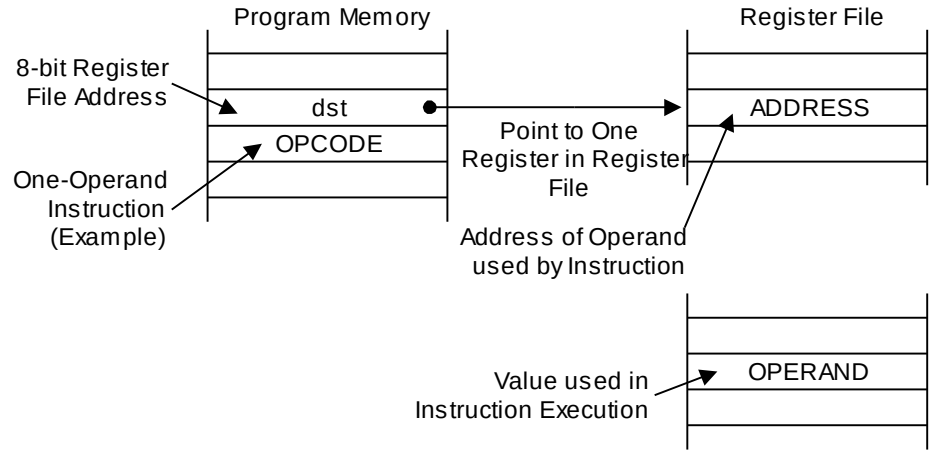


Figure 3-2 Working Register Addressing

3.3 Indirect Register Addressing Mode (IR)

In Indirect Register (IR) addressing mode, the content of the specified register or register pair is the address of the operand. Depending on the instruction used, the actual address may point to a register in the register file, to program memory (ROM), or to an external memory space (see *Figure 3-3*, *Figure 3-4*, *Figure 3-5* and *Figure 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. Please note, however, that you cannot access locations C0H–FFH in set 1 using the Indirect Register addressing mode.



Sample Instruction:

```
RL      @SHIFT          ;  Where SHIFT is the label of an 8-bit register address
```

Figure 3-3 Indirect Register Addressing to Register File

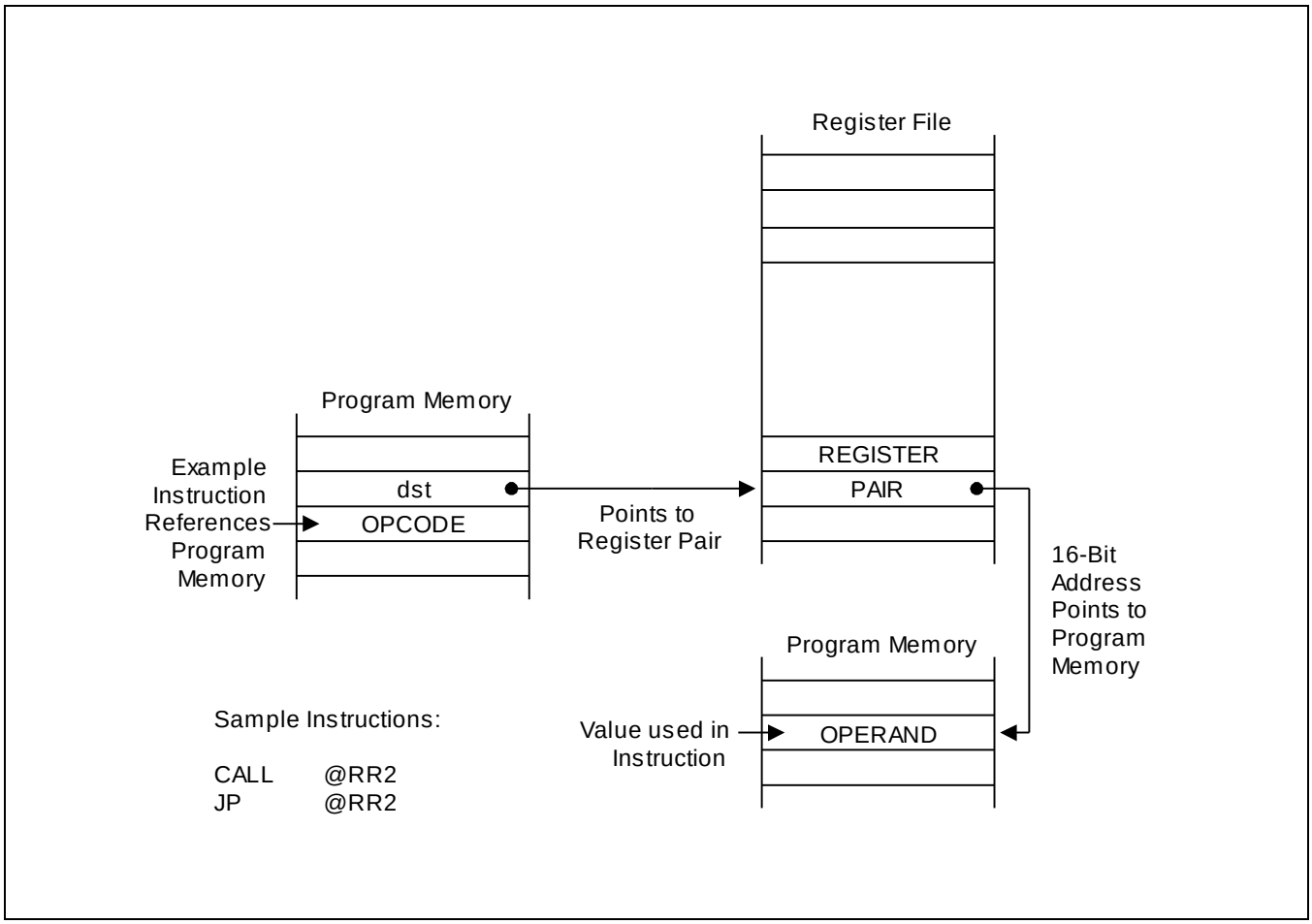


Figure 3-4 Indirect Register Addressing to Program Memory

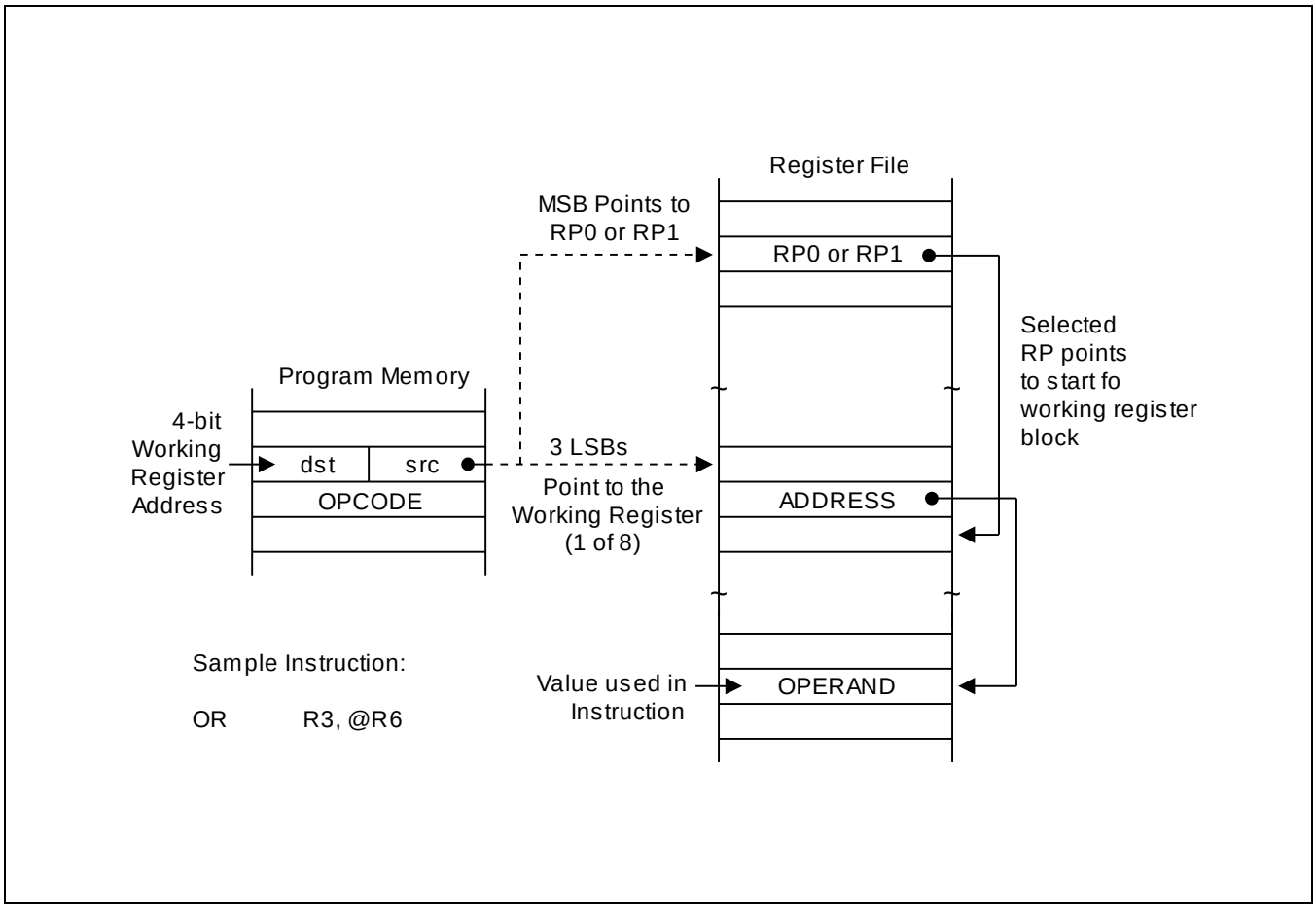


Figure 3-5 Indirect Working Register Addressing to Register File

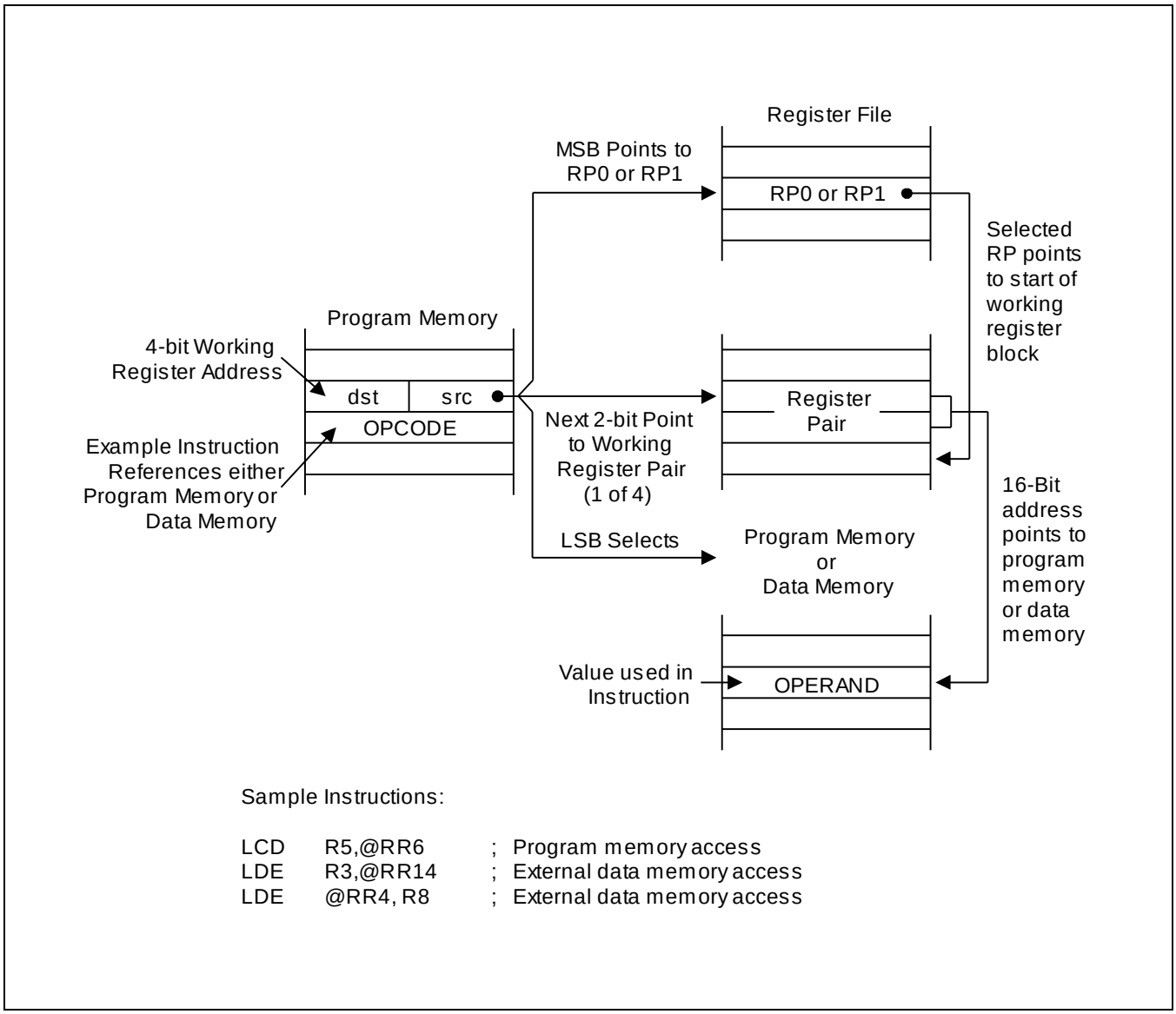


Figure 3-6 Indirect Working Register Addressing to Program or Data Memory

3.4 Indexed Addressing Mode (X)

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

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

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

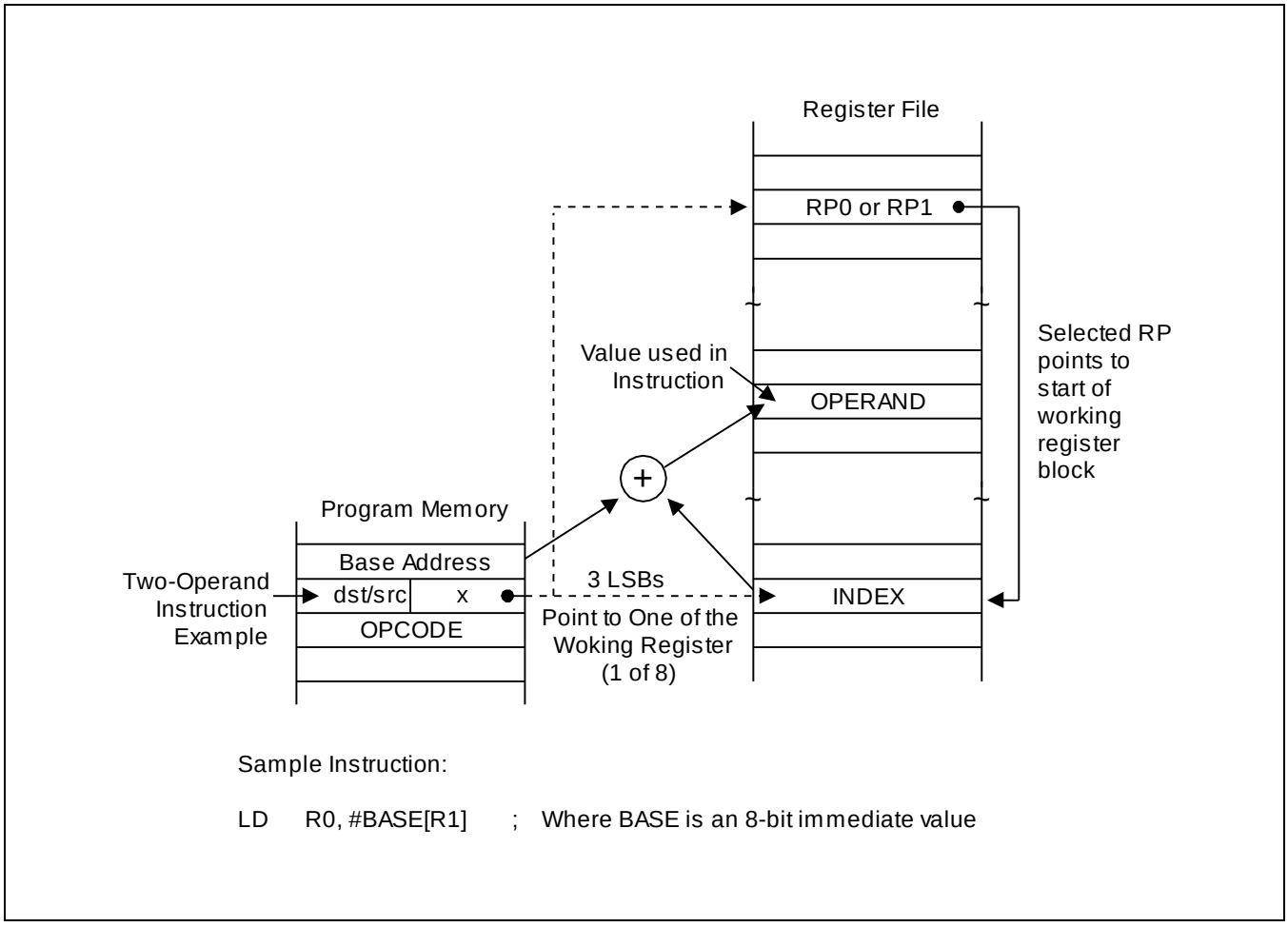


Figure 3-7 Indexed Addressing to Register File

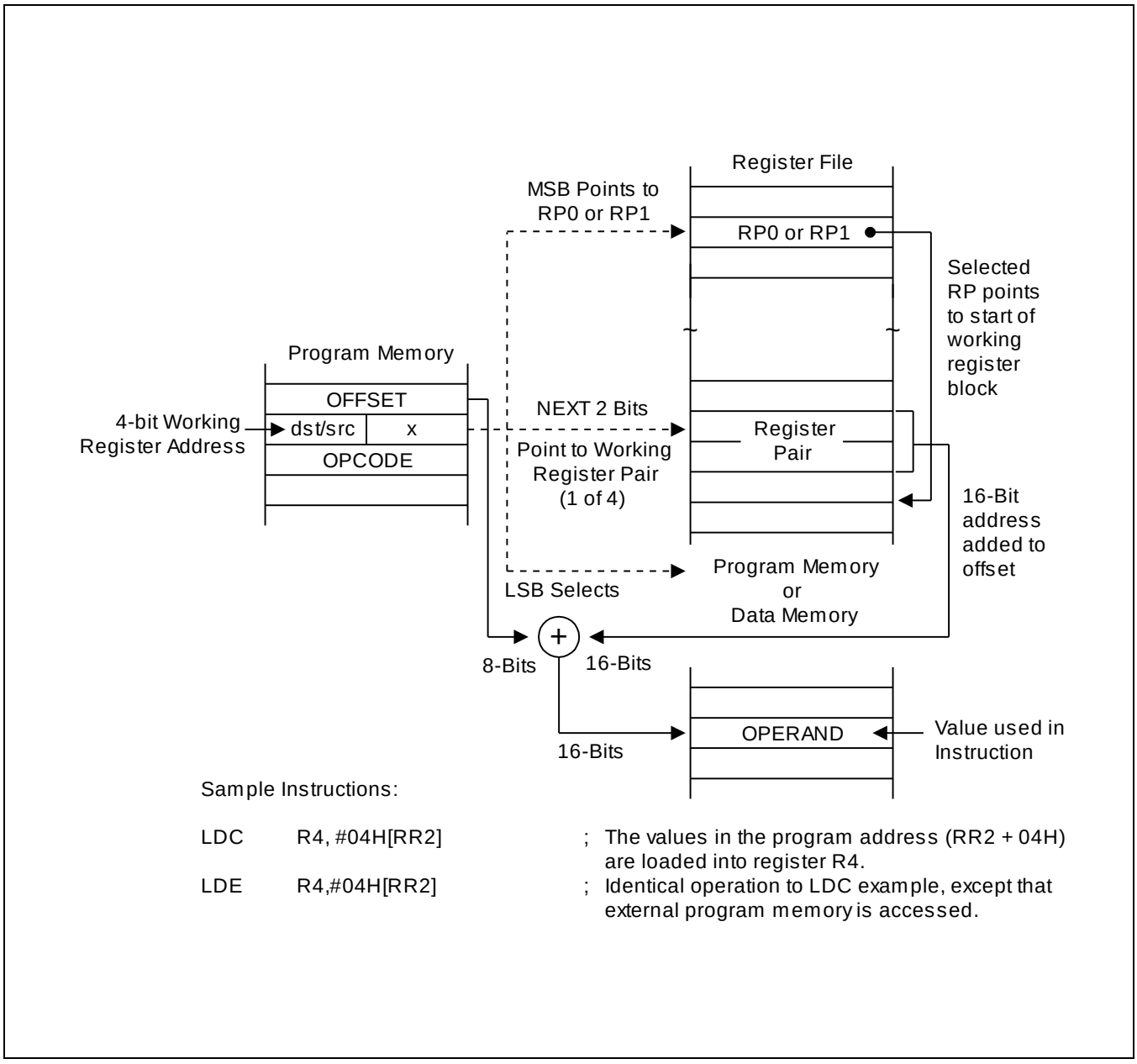


Figure 3-8 Indexed Addressing to Program or Data Memory with Short Offset

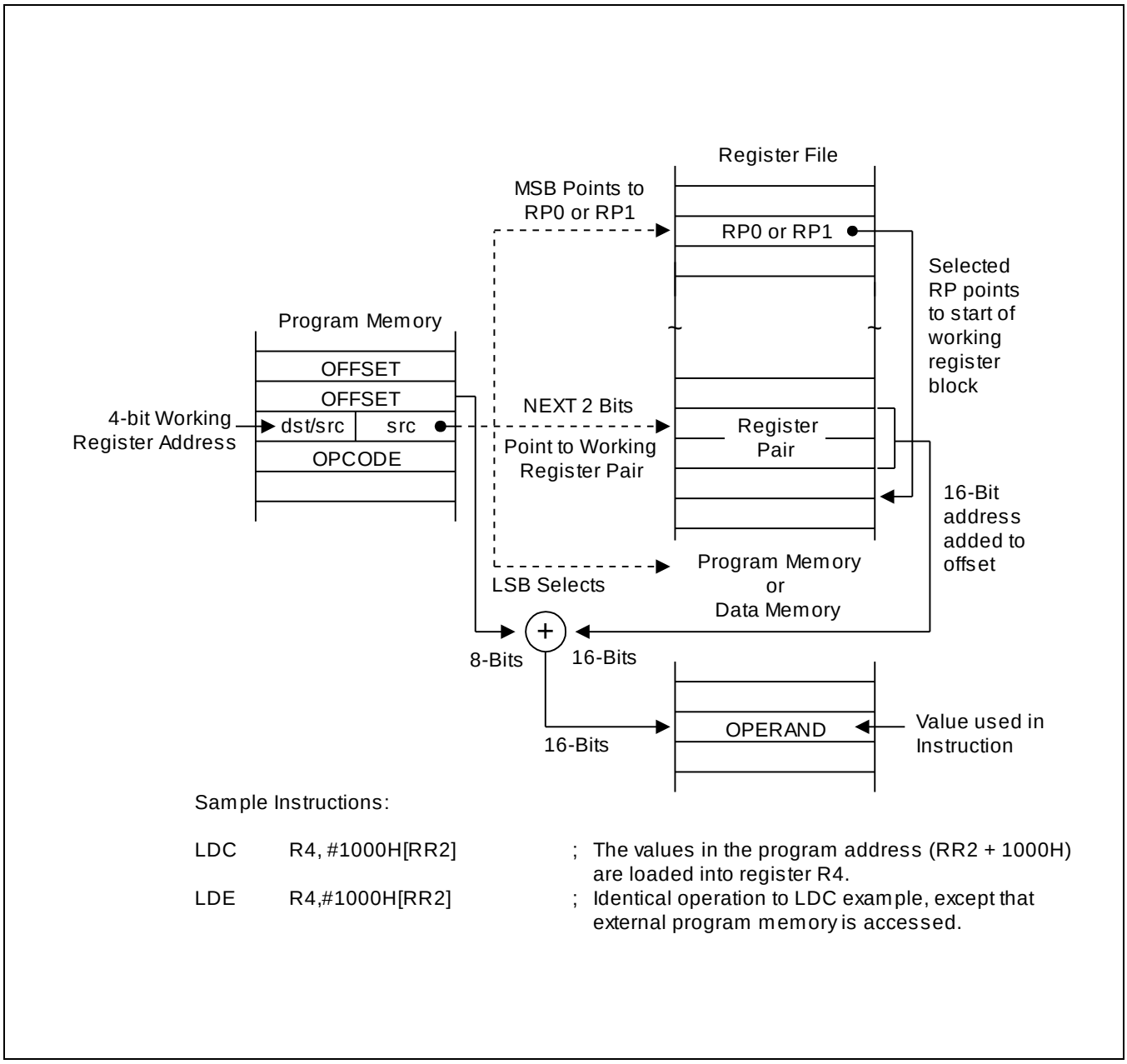


Figure 3-9 Indexed Addressing to Program or Data Memory

3.5 Direct Address Mode (DA)

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

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

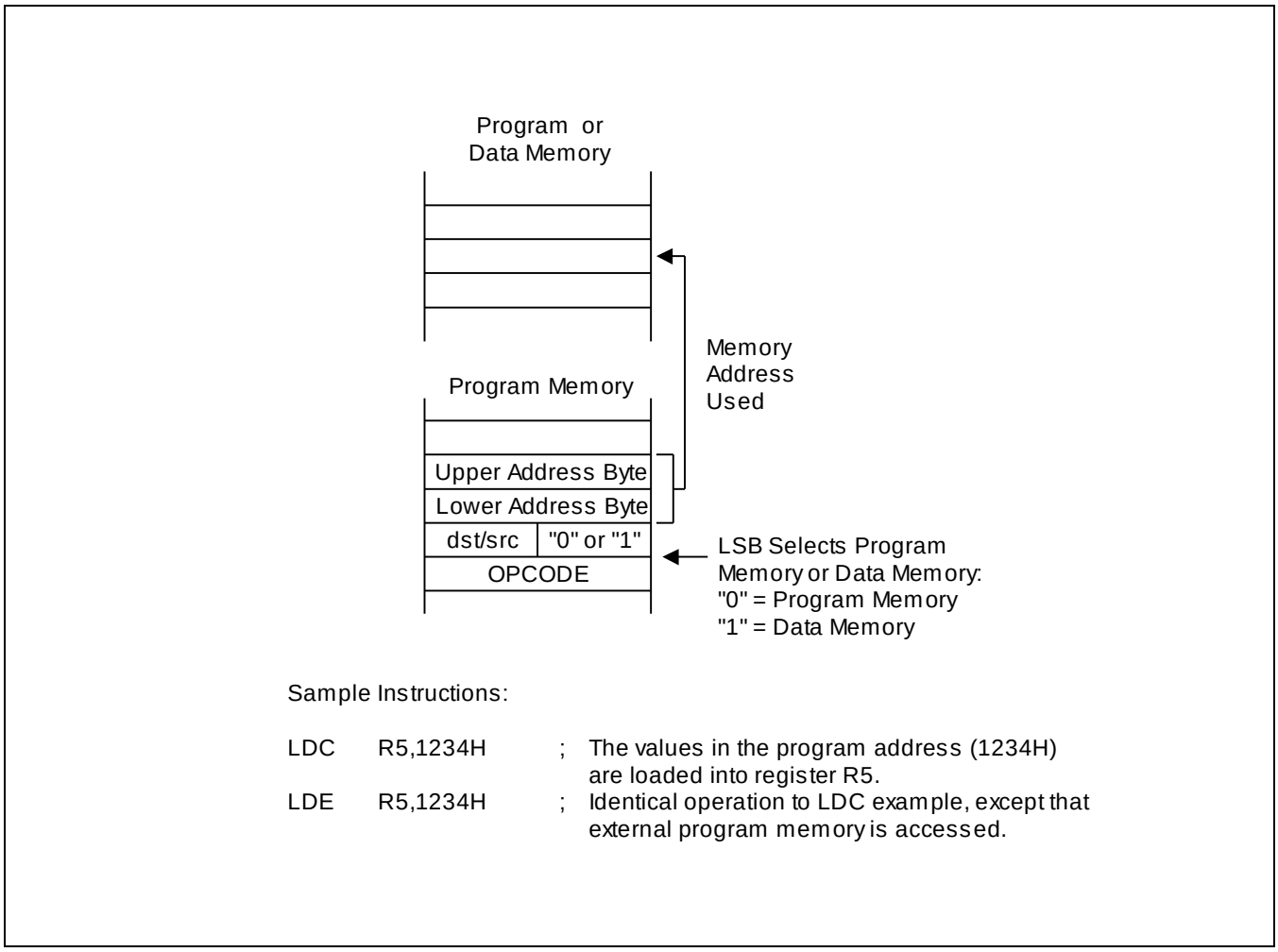
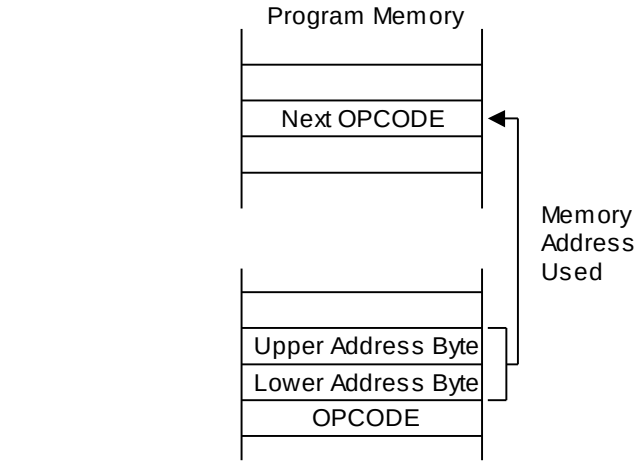


Figure 3-10 Direct Addressing for Load Instructions



Sample Instructions:

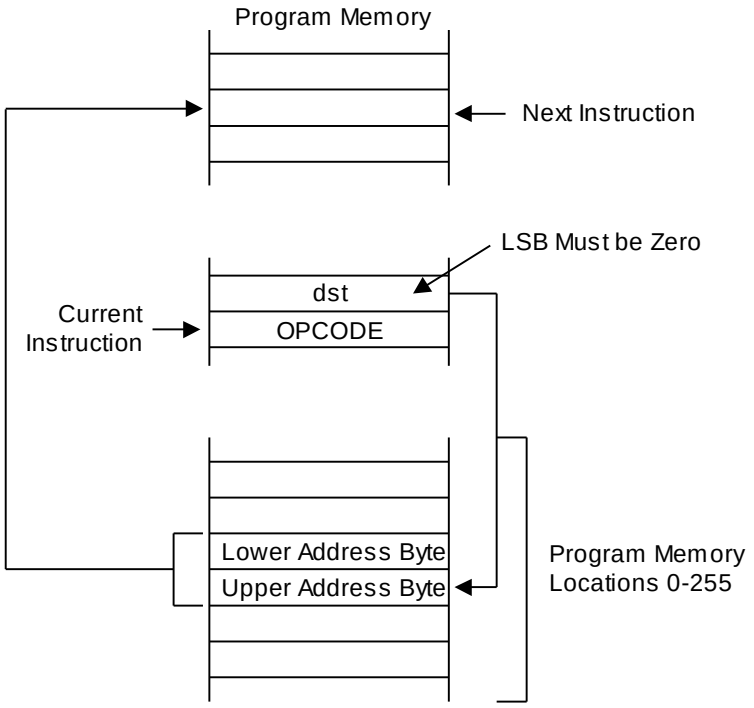
```
JP      C,JOB1          ;  Where JOB1 is a 16-bit immediate address
CALL    DISPLAY         ;  Where DISPLAY is a 16-bit immediate address
```

Figure 3-11 Direct Addressing for Call and Jump Instructions

3.6 Indirect Address Mode (IA)

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

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



Sample Instruction:

```
CALL    #40H        ; The 16-bit value in program memory addresses 40H
                      and 41H is the subroutine start address.
```

Figure 3-12 Indirect Addressing

3.7 Relative Address Mode (RA)

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

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

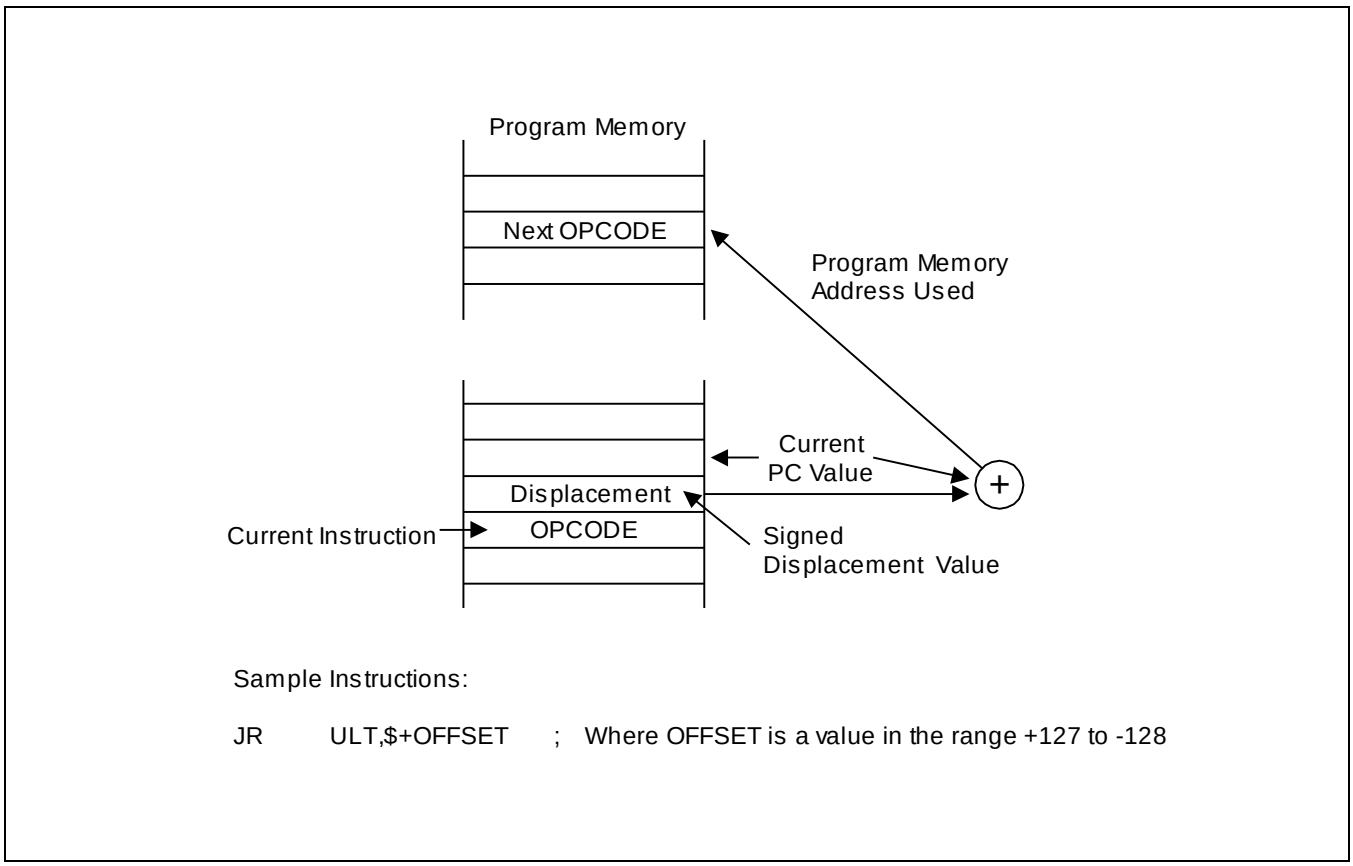


Figure 3-13 Relative Addressing

3.8 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.

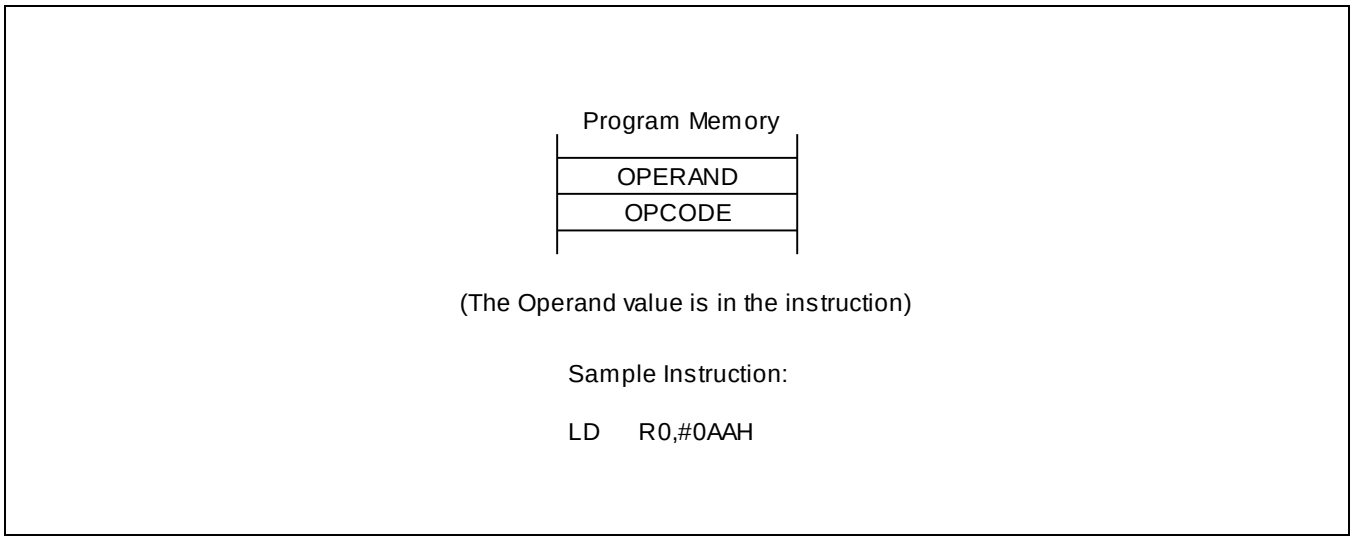


Figure 3-14 Immediate Addressing

4 Control Registers

4.1 Overview

In this chapter, detailed descriptions of the S3F8S6B control registers are presented in an easy-to-read format. You can use this chapter as a quick-reference source when writing application programs. Figure 4-1 illustrates the important features of the standard register description format.

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

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

The locations and read/write characteristics of all mapped registers in the S3F8S6B register file are listed in *Table 4.1*. The hardware reset value for each mapped register is described in Chapter 8, "RESET and Power-Down."

Table 4.1 Set 1 Registers

Register Name	Mnemonic	Decimal	Hex	RW
Basic Timer Control Register	BTCON	211	D3H	RW
System Clock Control Register	CLKCON	212	D4H	RW
System Flags Register	FLAGS	213	D5H	RW
Register Pointer 0	RP0	214	D6H	RW
Register Pointer 1	RP1	215	D7H	RW
Stack Pointer (High Byte)	SPH	216	D8H	RW
Stack Pointer (Low Byte)	SPL	217	D9H	RW
Instruction Pointer (High Byte)	IPH	218	DAH	RW
Instruction Pointer (Low Byte)	IPL	219	DBH	RW
Interrupt Request Register	IRQ	220	DCH	R
Interrupt Mask Register	IMR	221	DDH	RW
System Mode Register	SYM	222	DEH	RW
Register Page Pointer	PP	223	DFH	RW

Table 4.2 Set 1, Bank 0 Registers

Register Name	Mnemonic	Decimal	Hex	RW
A/D Converter Data Register (High Byte)	ADDATAH	208	D0H	R
A/D Converter Data Register (Low Byte)	ADDATAL	209	D1H	R
A/D Converter Control Register	ADCON	210	D2H	RW
Timer A Counter Register	TACNT	224	E0H	R
Timer A Data Register	TADATA	225	E1H	RW
Timer A Control Register	TACON	226	E2H	RW
Timer B Control Register	TBCON	227	E3H	RW
Timer B Data Register (High Byte)	TBDATAH	228	E4H	RW
Timer B Data Register (Low Byte)	TBDATAL	229	E5H	RW
Watch Timer Control Register	WTCON	230	E6H	RW
SIO Control Register	SIOCON	231	E7H	RW
SIO Data Register	SIODATA	232	E8H	RW
SIO Pre-Scaler Register	SIOPS	233	E9H	RW
Timer C Counter Register	TCCNT	234	EAH	R
Timer C Data Register	TCDATA	235	EBH	RW
Timer C Control Register	TCCON	236	ECH	RW
Locations EDH–EFH are not mapped.				
LCD Control Register	LCON	240	F0H	RW
LCD Mode Register	LMOD	241	F1H	RW
Locations F2H–F3H are not mapped.				
Interrupt Pending Register	INTPND	244	F4H	RW
STOP Control Register	STPCON	245	F5H	RW
Flash Memory Sector Address Register (High Byte)	FMSECH	246	F6H	RW
Flash Memory Sector Address Register (Low Byte)	FMSECL	247	F7H	RW
Flash Memory User Programming Enable Register	FMUSR	248	F8H	RW
Flash Memory Control Register	FMCON	249	F9H	RW
Oscillator Control Register	OSCCON	250	FAH	RW
Locations FBH–FCH are not mapped.				
Basic Timer Counter	BTCNT	253	FDH	R
Location FEH is not mapped.				
Interrupt Priority Register	IPR	255	FFH	RW

Table 4.3 Set 1, Bank 1 Registers

Register Name	Mnemonic	Decimal	Hex	RW
Pattern Generation Control Register	PGCON	208	D0H	RW
Pattern Generation Data Register	PGDATA	209	D1H	RW
Location D2H is not mapped.				
Port 0 Control Register (High Byte)	P0CONH	224	E0H	RW
Port 0 Control Register (Low Byte)	P0CONL	225	E1H	RW
Port 1 Control Register (High Byte)	P1CONH	226	E2H	RW
Port 1 Control Register (Low Byte)	P1CONL	227	E3H	RW
Port 1 Pull-up Resistor Enable Register	P1PUR	228	E4H	RW
Port 1 N-Channel Open-drain Mode Register	PNE1	229	E5H	RW
Port 2 Control Register (High Byte)	P2CONH	230	E6H	RW
Port 2 Control Register (Low Byte)	P2CONL	231	E7H	RW
Port 2 Interrupt Control Register (High Byte)	P2INTH	232	E8H	RW
Port 2 Interrupt Control Register (Low Byte)	P2INTL	233	E9H	RW
Port 2 Interrupt Pending Register	P2PND	234	EAH	RW
Port 3 Interrupt Control Register (High Byte)	P3CONH	235	EBH	RW
Port 3 Interrupt Control Register (Middle Byte)	P3CONM	236	ECH	RW
Port 3 Interrupt Control Register (Low Byte)	P3CONL	237	EDH	RW
Port 3 Pull-up Resistor Enable Register	P3PUR	238	EEH	RW
Port 3 N-Channel Open-drain Mode Register	PNE3	239	EFH	RW
Port 5 Control Register (High Byte)	P5CONH	240	F0H	RW
Port 5 Control Register (Low Byte)	P5CONL	241	F1H	RW
Port 6 Control Register (High Byte)	P6CONH	242	F2H	RW
Port 6 Control Register (Low Byte)	P6CONL	243	F3H	RW
Port 6 Pull-up Resistor Enable Register	P6PUR	244	F4H	RW
Port 6 N-Channel Open-drain Mode Register	PNE6	245	F5H	RW
Timer D0 Counter Register (High Byte)	TD0CNTH	246	F6H	R
Timer D0 Counter Register (Low Byte)	TD0CNTL	247	F7H	R
Timer D0 Data Register (High Byte)	TD0DATAH	248	F8H	RW
Timer D0 Data Register (Low Byte)	TD0DATAL	249	F9H	RW
Timer D0 Control Register	TD0CON	250	FAH	RW
Timer D1 Control Register	TD1CON	251	FBH	RW
Timer D1 Counter Register (High Byte)	TD1CNTH	252	FCH	R
Timer D1 Counter Register (Low Byte)	TD1CNTL	253	FDH	R
Timer D1 Data Register (High Byte)	TD1DATAH	254	FEH	RW
Timer D1 Data Register (Low Byte)	TD1DATAL	255	FFH	RW

Table 4.4 Page 8 Registers

Register Name	Mnemonic	Decimal	Hex	RW
Port 0 Data Register	P0	0	00H	RW
Port 1 Data Register	P1	1	01H	RW
Port 2 Data Register	P2	2	02H	RW
Port 3 Data Register	P3	3	03H	RW
Port 4 Data Register	P4	4	04H	RW
Port 5 Data Register	P5	5	05H	RW
Port 6 Data Register	P6	6	06H	RW
Locations 07H–0BH are not mapped.				
Port 4 Control Register (High Byte)	P4CONH	12	0CH	RW
Port 4 Control Register (Low Byte)	P4CONL	13	0DH	RW
Port 4 Interrupt Control Register (High Byte)	P4INTH	14	0EH	RW
Port 4 Interrupt Control Register (Low Byte)	P4INTL	15	0FH	RW
Port 4 Interrupt Pending Register	P4PND	16	10H	RW
Locations 11H–13H are not mapped.				
UART 0 Control Register (High Byte)	UART0CONH	20	14H	RW
UART 0 Control Register (Low Byte)	UART0CONL	21	15H	RW
UART 0 Data Register	UDATA0	22	16H	RW
UART 0 Baud Rate Data Register	BRDATA0	23	17H	RW
UART 1 Control Register (High Byte)	UART1CONH	24	18H	RW
UART 1 Control Register (Low Byte)	UART1CONL	25	19H	RW
UART 1 Data Register	UDATA1	26	1AH	RW
UART 1 Baud Rate Data Register	BRDATA1	27	1BH	RW
Locations 1CH–2FH are not mapped.				

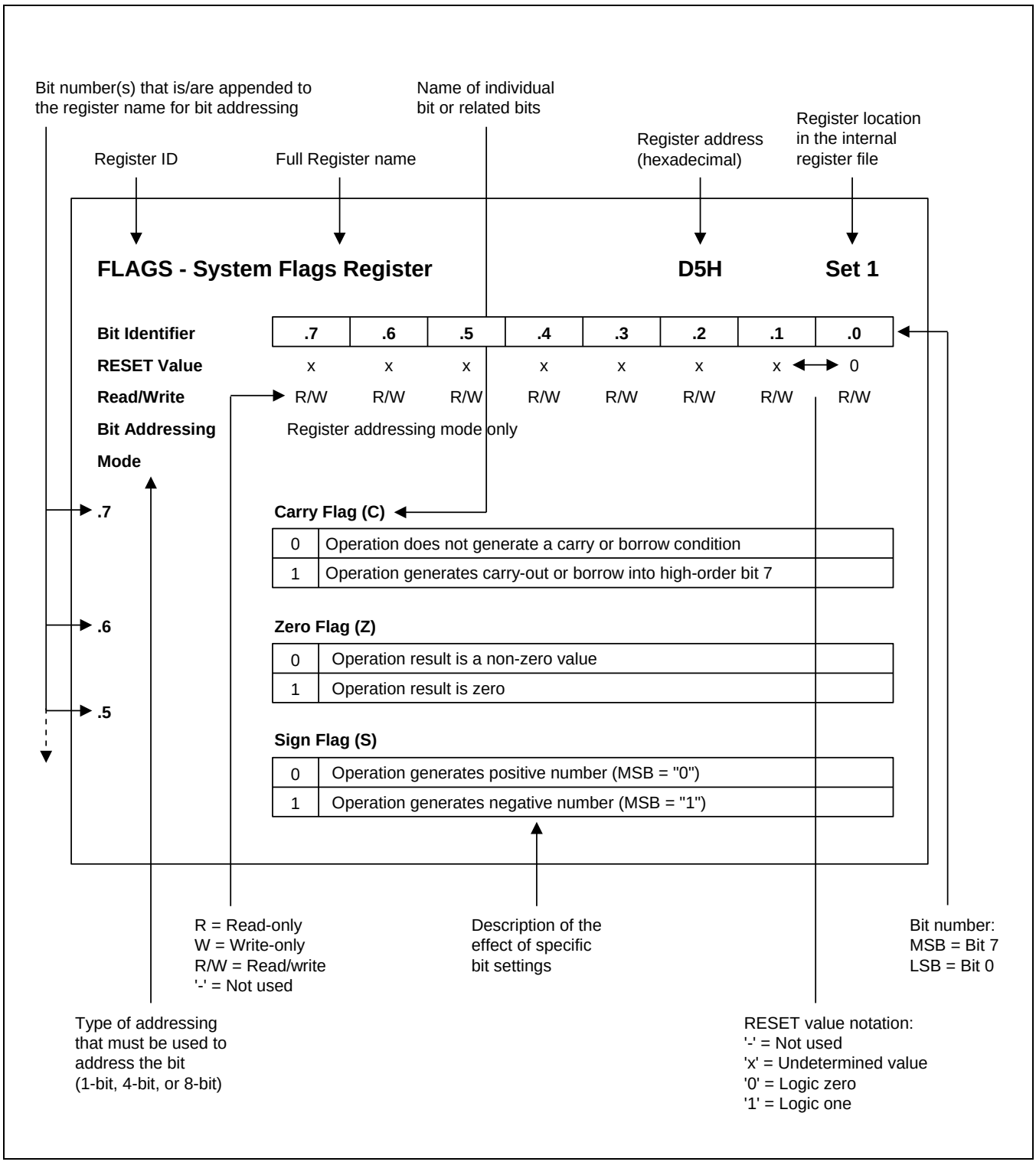


Figure 4-1 Register Description Format

4.1.1 ADCON: A/D Converter Control Register (D2H, 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	–	RW	RW	RW	R	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7 **Not used for the S3F8S6B**

.6–.4 **A/D Input Pin Selection Bits**

0	0	0	AD0
0	0	1	AD1
0	1	0	AD2
0	1	1	AD3
1	0	0	AD4
1	0	1	AD5
1	1	0	AD6
1	1	1	AD7

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

0	Conversion not complete
1	Conversion complete

.2–.1 **Clock Source Selection Bits**

0	0	fxx/16
0	1	fxx/8
1	0	fxx/4
1	1	fxx/1

.0 **Start or Enable Bit**

0	Disable operation
1	Start operation

4.1.2 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

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

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

.3–.2 Basic Timer Input Clock Selection Bits (3)

0	0	fxx/4096
0	1	fxx/1024
1	0	fxx/128
1	1	fxx/16

.1 Basic Timer Counter Clear Bit (1)

0	No effect
1	Clear the basic timer counter value

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

0	No effect
1	Clear both clock frequency dividers

NOTE:

1. When you write a "1" to BTCON.1, the basic timer counter value is cleared to "00H". Immediately following the write operation, the BTCON.1 value is automatically cleared to "0".
2. When you write a "1" to BTCON.0, the corresponding frequency divider is cleared to "00H". Immediately following the write operation, the BTCON.0 value is automatically cleared to "0".
3. The fxx is selected clock for system (main OSC. or sub OSC).

4.1.3 CLKCON: System Clock Control Register (D4H, Set 1)

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

.7 Oscillator IRQ Wake-up Function Bit

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

.6–.5 Not used for the S3F8S6B (must keep always "0")

.4–.3 CPU Clock (System Clock) Selection Bits (note)

0	0	fxx/16
0	1	fxx/8
1	0	fxx/2
1	1	fxx/1

.2–.0 Not used for the S3F8S6B (must keep always "0")

NOTE: After a reset, the slowest clock (divided by 16) is selected as the system clock. To select faster clock speeds, load the appropriate values to CLKCON.3 and CLKCON.4.

4.1.4 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	RW	RW	RW	RW	RW	RW	R	RW
Addressing Mode	Register addressing mode only							

.7 Carry Flag (C)

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

.6 Zero Flag (Z)

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

.5 Sign Flag (S)

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

.4 Overflow Flag (V)

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

.3 Decimal Adjust Flag (D)

0	Add operation completed
1	Subtraction operation completed

.2 Half-Carry Flag (H)

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

.1 Fast Interrupt Status Flag (FIS)

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

.0 Bank Address Selection Flag (BA)

0	Bank 0 is selected
1	Bank 1 is selected

4.1.5 FMCON: Flash Memory Control Register (F9H, Set 1, Bank 0)

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

.7–.4 Flash Memory Mode Selection Bits

0	1	0	1	Programming mode
1	0	1	0	Sector erase mode
0	1	1	0	Hard lock mode
Others				Not available

.3 Sector Erase Status Bit (Read-only)

0	Success sector erase
1	Fail sector erase

.2–.1 Not used for the S3F8S6B

.0 Flash Operation Start Bit

0	Operation stop bit
1	Operation start bit

NOTE: The FMCON.0 will be cleared automatically just after the corresponding operation completed.

4.1.6 FMSECH: Flash Memory Sector Address Register-High Byte (F6H, Set 1, Bank 0)

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

.7–.0	Flash Memory Sector Address Bits (High Byte)
	The 15th-8th bits to select a sector of Flash ROM

NOTE: The high byte Flash memory sector address pointer value is higher eight bits of the 16-bit pointer address.

4.1.7 FMSECL: Flash Memory Sector Address Register-Low Byte (F7H, Set 1, Bank 0)

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

.7	Flash Memory Sector Address Bit (Low Byte)
	The 7th bit to select a sector of Flash ROM

.6–.0	Don't care

NOTE: The low byte Flash memory sector address pointer value is lower eight bits of the 16-bit pointer address.

4.1.8 FMUSR: Flash Memory User Programming Enable 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	0
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.0 Flash Memory User Programming Enable Bits

1	0	1	0	0	1	0	1	Enable user programming mode
Others								Disable user programming mode

4.1.9 IMR: Interrupt Mask Register (DDH, Set 1)

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

.7 Interrupt Level 7 (IRQ7) Enable Bit; External Interrupts P4.0–4.7

0	Disable (mask)
1	Enable (unmask)

.6 Interrupt Level 6 (IRQ6) Enable Bit; External Interrupts P2.0–2.7

0	Disable (mask)
1	Enable (unmask)

.5 Interrupt Level 5 (IRQ5) Enable Bit; UART0/1 Transmit, UART0/1 Receive

0	Disable (mask)
1	Enable (unmask)

.4 Interrupt Level 4 (IRQ4) Enable Bit; Watch Timer, SIO

0	Disable (mask)
1	Enable (unmask)

.3 Interrupt Level 3 (IRQ3) Enable Bit; Timer D0/D1 Match/Capture or Overflow

0	Disable (mask)
1	Enable (unmask)

.2 Interrupt Level 2 (IRQ2) Enable Bit; Timer C Match/Overflow

0	Disable (mask)
1	Enable (unmask)

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

0	Disable (mask)
1	Enable (unmask)

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

0	Disable (mask)
1	Enable (unmask)

NOTE: When an interrupt level is masked, any interrupt requests that may be issued are not recognized by the CPU.

4.1.10 INTPND: Interrupt Pending Register (F4H, Set 1, Bank 0)

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

.7–.6 **Not used for the S3F8S6B**

.5 **Timer D1 Match/Capture Interrupt Pending Bit**

0	No interrupt pending (when read), clear pending bit (when write)
1	Interrupt is pending (when read)

.4 **Timer D1 Overflow Interrupt Pending Bit**

0	No interrupt pending (when read), clear pending bit (when write)
1	Interrupt is pending (when read)

.3 **Timer D0 Match/Capture Interrupt Pending Bit**

0	No interrupt pending (when read), clear pending bit (when write)
1	Interrupt is pending (when read)

.2 **Timer D0 Overflow Interrupt Pending Bit**

0	No interrupt pending (when read), clear pending bit (when write)
1	Interrupt is pending (when read)

.1 **Timer A Match/Capture Interrupt Pending Bit**

0	No interrupt pending (when read), clear pending bit (when write)
1	Interrupt is pending (when read)

.0 **Timer A Overflow Interrupt Pending Bit**

0	No interrupt pending (when read), clear pending bit (when write)
1	Interrupt is pending (when read)

4.1.11 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	RW	RW	RW	RW	RW	RW	RW	RW
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).

4.1.12 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	RW	RW	RW	RW	RW	RW	RW	RW
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).

4.1.13 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	x
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

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

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

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

0	IRQ6 > IRQ7
1	IRQ7 > IRQ6

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

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

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

0	IRQ3 > IRQ4
1	IRQ4 > IRQ3

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

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

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

0	IRQ0 > IRQ1
1	IRQ1 > IRQ0

NOTE: Interrupt group A -IRQ0, IRQ1
Interrupt group B -IRQ2, IRQ3, IRQ4
Interrupt group C -IRQ5, IRQ6, IRQ7

4.1.14 IRQ: Interrupt Request Register (DCH, Set 1)

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

.7 Level 7 (IRQ7) Request Pending Bit; External Interrupts P4.0–4.7

0	Not pending
1	Pending

.6 Level 6 (IRQ6) Request Pending Bit; External Interrupts P2.0–2.7

0	Not pending
1	Pending

.5 Level 5 (IRQ5) Request Pending Bit; UART0/1 Transmit, UART0/1 Receive

0	Not pending
1	Pending

.4 Level 4 (IRQ4) Request Pending Bit; Watch Timer, SIO

0	Not pending
1	Pending

.3 Level 3 (IRQ3) Request Pending Bit; Timer D0/D1 Match/Capture or Overflow

0	Not pending
1	Pending

.2 Level 2 (IRQ2) Request Pending Bit; Timer C Match/Overflow

0	Not pending
1	Pending

.1 Level 1 (IRQ1) Request Pending Bit; Timer B Match

0	Not pending
1	Pending

.0 Level 0 (IRQ0) Request Pending Bit; Timer A Match/Capture or Overflow

0	Not pending
1	Pending

4.1.15 LCON: LCD Control Register (F0H, Set 1, Bank 0)

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

.7–.6 LCD Clock Selection Bits

0	0	fw/2^8 (128 Hz)
0	1	fw/2^7 (256 Hz)
1	0	fw/2^6 (512 Hz)
1	1	fw/2^5 (1024 Hz)

.5–.3 LCD Duty and Bias Selection Bits

0	0	0	1/8 duty, 1/4 bias
0	0	1	1/4 duty, 1/3 bias
0	1	0	1/3 duty, 1/3 bias
0	1	1	1/3 duty, 1/2 bias
1	x	x	1/2 duty, 1/2 bias

.2–.1 LCD Bias Type Selection Bits (note)

0	0	V_{LC0} – V_{LC3}, CA and CB pins are normal I/O pin
0	1	Capacitor bias; V_{LC0}– V_{LC3}, CA and CB pins are bias pin
1	0	Internal resistor bias (The voltage booster is always stopped and cut off); V_{LC0} – V_{LC3} are bias pin, CA, and CB pins are normal pin
1	1	External resistor bias (The voltage booster is always stopped and cut off); V_{LC0} – V_{LC3} are bias pin, CA, and CB pins are normal pin

.0 LCD Display Control Bits

0	All LCD signals are low (The voltage booster is always stopped and cut off)
1	Turn display on (When LCON.2–.1 = "01", Run and connect voltage booster)

NOTE:

1. When LCON.2-.1 are selected to '01', P5.0-.5 are automatically selected to VLCn, CA and CB pin. (n = 0 to 3).
   When LCON.2-.1 are capacitor bias selected, LCON.0 is select to "1" after 1 millisecond delay.
2. When LCON.2-.1 are selected to "10", P5.0-.3 are automatically selected to VLCn. (n = 0 to 3)
3. When LCON.2-.1 are selected to "11", P5.0-.3 are automatically selected to VLCn. (n = 0 to 3)
4. The clock and duty for LCD controller/driver is automatically initialized by hardware, whenever LCON register data value is re-write. So, the LCON register don"t re-write frequently.
5. The P5.3/VLC3-P5.0/VLC0 must be used as LCD bias pins if the LCD block is used. So, the LCON.2-.1 must not be set to "00b" when LCON.0 = 1.

4.1.16 LMOD: LCD Mode Control Register (F1H, Set 1, Bank 0)

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

.7–.3	Not used for the S3F8S6B

In Run, Idle, and Sub Operating Modes

.2–.0 VLCD Voltage Selection Bits (Only when the capacitor bias is selected)

Values			1/4 Bias	1/3 Bias	1/2 Bias
0	0	0	3.6 V	3.15 V	2.20 V
0	0	1	3.8 V	3.375 V	2.40 V
0	1	0	4.0 V	3.60 V	2.60 V
0	1	1	4.2 V	3.825 V	2.80 V
1	0	0	4.4 V	4.050 V	3.00 V
1	0	1	4.6 V	4.275 V	Not available
1	1	0	4.8 V	4.50 V	Not available
1	1	1	5.0 V	Not available	Not available

In Sub Idle and Stop Modes

.2–.0 VLCD Voltage Selection Bits (Only when the capacitor bias is selected)

Values			1/4 Bias	1/3 Bias	1/2 Bias
0	0	0	3.6 V	3.15 V	2.20 V
0	0	1	3.8 V	3.375 V	2.40 V
0	1	0	4.0 V	3.60 V	Not available
0	1	1	4.2 V	Not available	Not available
1	0	0	4.4 V	Not available	Not available
1	0	1	4.6 V	Not available	Not available
1	1	0	4.8 V	Not available	Not available
1	1	1	5.0 V	Not available	Not available

NOTE:

1. The voltage regulator and booster circuit provide constant LCD contrast level.
2. Since the booster clock is generated by watch timer (fw). And the booster clock is almost 32 kHz.

4.1.17 OSCCON: Oscillator Control Register (FAH, Set 1, Bank 0)

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

.7–.4 **Not used for the S3F8S6B**

.3 **Main Oscillator Control Bit**

0	Main oscillator RUN
1	Main oscillator STOP

.2 **Sub Oscillator Control Bit**

0	Sub oscillator RUN
1	Sub oscillator STOP

.1 **Not used for the S3F8S6B**

.0 **System Clock Selection Bit**

0	Select main oscillator for system clock
1	Select sub oscillator for system clock

4.1.18 P0CONH: Port 0 Control Register-High Byte (E0H, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.6 P0.7/SEG5/COM7 Configuration Bits

0	0	Input mode
0	1	Input mode, pull-up
1	0	Alternative function (LCD signal)
1	1	Output mode, push-pull

.5–.4 P0.6/SEG4/COM6 Configuration Bits

0	0	Input mode
0	1	Input mode, pull-up
1	0	Alternative function (LCD signal)
1	1	Output mode, push-pull

.3–.2 P0.5/SEG3/COM5 Configuration Bits

0	0	Input mode
0	1	Input mode, pull-up
1	0	Alternative function (LCD signal)
1	1	Output mode, push-pull

.1–.0 P0.4/SEG2/COM4 Configuration Bits

0	0	Input mode
0	1	Input mode, pull-up
1	0	Alternative function (LCD signal)
1	1	Output mode, push-pull

4.1.19 P0CONL: Port 0 Control Register-Low Byte (E1H, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.6 P0.3/SEG1/COM3 Configuration Bits

0	0	Input mode
0	1	Input mode, pull-up
1	0	Alternative function (LCD signal)
1	1	Output mode, push-pull

.5–.4 P0.2/SEG0/COM2 Configuration Bits

0	0	Input mode
0	1	Input mode, pull-up
1	0	Alternative function (LCD signal)
1	1	Output mode, push-pull

.3–.2 P0.1/COM1 Configuration Bits

0	0	Input mode
0	1	Input mode, pull-up
1	0	Alternative function (LCD signal)
1	1	Output mode, push-pull

.1–.0 P0.0/COM0 Configuration Bits

0	0	Input mode
0	1	Input mode, pull-up
1	0	Alternative function (LCD signal)
1	1	Output mode, push-pull

4.1.20 P1CONH: Port 1 Control Register-High Byte (E2H, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.6 P1.7/TACLK/SEG11 Configuration Bits

0	0	Input mode (TACLK)
0	1	Not available
1	0	Alternative function (LCD signal)
1	1	Output mode

.5–.4 P1.6/TAOUT/TAPWM/TACAP/SEG10 Configuration Bits

0	0	Input mode (TACAP)
0	1	Alternative function (TAOUT/TAPWM)
1	0	Alternative function (LCD signal)
1	1	Output mode

.3–.2 P1.5/TxD1/SEG9 Configuration Bits

0	0	Input mode
0	1	Alternative function (TxD1)
1	0	Alternative function (LCD signal)
1	1	Output mode

.1–.0 P1.4/RxD1/SEG8 Configuration Bits

0	0	Input mode (RxD1)
0	1	Alternative function (RxD1 out)
1	0	Alternative function (LCD signal)
1	1	Output mode

4.1.21 P1CONL: Port 1 Control Register-Low Byte (E3H, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.6 **P1.3/TxD0/SEG7 Configuration Bits**

0	0	Input mode
0	1	Alternative function (TxD0 out)
1	0	Alternative function (LCD signal)
1	1	Output mode

.5–.4 **P1.2/RxD0/SEG6 Configuration Bits**

0	0	Input mode (RxD0)
0	1	Alternative function (RxD0 out)
1	0	Alternative function (LCD signal)
1	1	Output mode

.3–.2 **P1.1 Configuration Bits**

0	0	Input mode
0	1	Not available
1	0	Not available
1	1	Output mode

.1–.0 **P1.0 Configuration Bits**

0	0	Input mode
0	1	Not available
1	0	Not available
1	1	Output mode

4.1.22 P1PUR: Port 1 Pull-up Resistor Enable Register (E4H, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7 P1.7 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.6 P1.6 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.5 P1.5 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.4 P1.4 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.3 P1.3 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.2 P1.2 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.1 P1.1 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.0 P1.0 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

NOTE: A pull-up resistor of port 1 is automatically disabled only when the corresponding pin is selected as push-pull output or alternative function.

4.1.23 PNE1: Port 1 N-channel Open-drain Mode Register (E5H, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7 P1.7 Output Mode Selection Bit

0	Output mode, push-pull
1	Output mode, open-drain

.6 P1.6 Output Mode Selection Bit

0	Output mode, push-pull
1	Output mode, open-drain

.5 P1.5 Output Mode Selection Bit

0	Output mode, push-pull
1	Output mode, open-drain

.4 P1.4 Output Mode Selection Bit

0	Output mode, push-pull
1	Output mode, open-drain

.3 P1.3 Output Mode Selection Bit

0	Output mode, push-pull
1	Output mode, open-drain

.2 P1.2 Output Mode Selection Bit

0	Output mode, push-pull
1	Output mode, open-drain

.1 P1.1 Output Mode Selection Bit

0	Output mode, push-pull
1	Output mode, open-drain

.0 P1.0 Output Mode Selection Bit

0	Output mode, push-pull
1	Output mode, open-drain

4.1.24 P2CONH: Port 2 Control Register-High Byte (E6H, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.6 **P2.7/INT7/SEG19 Configuration Bits**

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (LCD signal)
1	1	Output mode, push-pull

.5-.4 **P2.6/INT6/SEG18 Configuration Bits**

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (LCD signal)
1	1	Output mode, push-pull

.3–.2 **P2.5/INT5/SEG17 Configuration Bits**

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (LCD signal)
1	1	Output mode, push-pull

.1–.0 **P2.4/INT4/SEG16 Configuration Bits**

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (LCD signal)
1	1	Output mode, push-pull

4.1.25 P2CONL: Port 2 Control Register-Low Byte (E7H, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.6 P2.3/INT3/SEG15 Configuration Bits

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (LCD signal)
1	1	Output mode, push-pull

.5–.4 P2.2/INT2/SEG14 Configuration Bits

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (LCD signal)
1	1	Output mode, push-pull

.3–.2 P2.1/INT1/SEG13 Configuration Bits

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (LCD signal)
1	1	Output mode, push-pull

.1–.0 P2.0/INT0/SEG12 Configuration Bits

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (LCD signal)
1	1	Output mode, push-pull

4.1.26 P2INTH: Port 2 Interrupt Control Register-High Byte (E8H, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.6 **P2.7/External interrupt (INT7) Enable Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.5–.4 **P2.6/External interrupt (INT6) Enable Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.3–.2 **P2.5/External interrupt (INT5) Enable Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.1–.0 **P2.4/External interrupt (INT4) Enable Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

4.1.27 P2INTL: Port 2 Interrupt Control Register-Low Byte (E9H, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.6 **P2.3/External interrupt (INT3) Enable Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.5–.4 **P2.2/External interrupt (INT2) Enable Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.3–.2 **P2.1/External interrupt (INT1) Enable Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.1–.0 **P2.0/External interrupt (INT0) Enable Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

4.1.28 P2PND: Port 2 Interrupt Pending Register (EAH, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7 P2.7/External Interrupt (INT7) Pending Bit

0	Clear pending bit (when write)
1	Interrupt request is pending (when read)

.6 P2.6/External Interrupt (INT6) Pending Bit

0	Clear pending bit (when write)
1	Interrupt request is pending (when read)

.5 P2.5/External Interrupt (INT5) Pending Bit

0	Clear pending bit (when write)
1	Interrupt request is pending (when read)

.4 P2.4/External Interrupt (INT4) Pending Bit

0	Clear pending bit (when write)
1	Interrupt request is pending (when read)

.3 P2.3/External Interrupt (INT3) Pending Bit

0	Clear pending bit (when write)
1	Interrupt request is pending (when read)

.2 P2.2/External Interrupt (INT2) Pending Bit

0	Clear pending bit (when write)
1	Interrupt request is pending (when read)

.1 P2.1/External Interrupt (INT1) Pending Bit

0	Clear pending bit (when write)
1	Interrupt request is pending (when read)

.0 P2.0/External Interrupt (INT0) Pending Bit

0	Clear pending bit (when write)
1	Interrupt request is pending (when read)

4.1.29 P3CONH: Port 3 Control Register-High Byte (EBH, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.5 P3.7/BUZ/PG7/SEG27 Configuration Bits

0	0	0	Input mode
0	0	1	Alternative function (PG7)
0	1	0	Alternative function (LCD signal)
0	1	1	Output mode
1	x	x	Alternative function (BUZ)

.4–.3 P3.6/TD1CLK/PG6/SEG26 Configuration Bits

0	0	Input mode (TD1CLK)
0	1	Alternative function (PG6)
1	0	Alternative function (LCD signal)
1	1	Output mode

.2–.0 P3.5/TD1OUT/TD1PWM/TD1CAP/PG5/SEG25 Configuration Bits

0	0	0	Input mode (TD1CAP)
0	0	1	Alternative function (PG5)
0	1	0	Alternative function (LCD signal)
0	1	1	Output mode
1	x	x	Alternative function (TD1OUT/TD1PWM)

4.1.30 P3CONM: Port 3 Control Register-Middle Byte (ECH, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.6 P3.4/TD0CLK/PG4/SEG24 Configuration Bits

0	0	Input mode (TD0CLK)
0	1	Alternative function (PG4)
1	0	Alternative function (LCD signal)
1	1	Output mode

.5–.3 P3.3/TD0OUT/TD0PWM/TD0CAP/PG3/SEG23 Configuration Bits

0	0	0	Input mode (TD0CAP)
0	0	1	Alternative function (PG3)
0	1	0	Alternative function (LCD signal)
0	1	1	Output mode
1	x	x	Alternative function (TD0OUT/TD0PWM)

.2–.0 P3.2/PG2/SEG22 Configuration Bits

0	0	0	Input mode
0	0	1	Alternative function (PG2)
0	1	0	Alternative function (LCD signal)
0	1	1	Output mode
1	x	x	Alternative function

4.1.31 P3CONL: Port 3 Control Register-Low Byte (EDH, Set 1, Bank 1)

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

.7–.6 **Not used for the S3F8S6B**

.5–.3 **P3.1/TCOUT/PG1/SEG21 Configuration Bits**

0	0	0	Input mode
0	0	1	Alternative function (PG1)
0	1	0	Alternative function (LCD signal)
0	1	1	Output mode
1	x	x	Alternative function (TCOUT)

.2–.0 **P3.0/TBPWM/PG0/SEG20 Configuration Bits**

0	0	0	Input mode
0	0	1	Alternative function (PG0)
0	1	0	Alternative function (LCD signal)
0	1	1	Output mode
1	x	x	Alternative function (TBPWM)

4.1.32 P3PUR: Port 3 Pull-up Resistor Enable Register (EEH, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7 P3.7 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.6 P3.6 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.5 P3.5 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.4 P3.4 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.3 P3.3 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.2 P3.2 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.1 P3.1 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.0 P3.0 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

NOTE: A pull-up resistor of port 3 is automatically disabled only when the corresponding pin is selected as push-pull output or alternative function.

4.1.33 PNE3: Port 3 N-channel Open-drain Mode Register (EFH, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7 P3.7 Output Mode Selection Bit

0	Output mode, push-pull
1	Output mode, open-drain

.6 P3.6 Output Mode Selection Bit

0	Output mode, push-pull
1	Output mode, open-drain

.5 P3.5 Output Mode Selection Bit

0	Output mode, push-pull
1	Output mode, open-drain

.4 P3.4 Output Mode Selection Bit

0	Output mode, push-pull
1	Output mode, open-drain

.3 P3.3 Output Mode Selection Bit

0	Output mode, push-pull
1	Output mode, open-drain

.2 P3.2 Output Mode Selection Bit

0	Output mode, push-pull
1	Output mode, open-drain

.1 P3.1 Output Mode Selection Bit

0	Output mode, push-pull
1	Output mode, open-drain

.0 P3.0 Output Mode Selection Bit

0	Output mode, push-pull
1	Output mode, open-drain

4.1.34 P4CONH: Port 4 Control Register-High Byte (0CH, Page 8)

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

.7–.6 P4.7/INT15/AD7 Configuration Bits

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (AD7)
1	1	Output mode, push-pull

.5–.4 P4.6/INT14/AD6 Configuration Bits

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (AD6)
1	1	Output mode, push-pull

.3–.2 P4.5/INT13/AD5 Configuration Bits

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (AD5)
1	1	Output mode, push-pull

.1–.0 P4.4/INT12/AD4 Configuration Bits

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (AD4)
1	1	Output mode, push-pull

4.1.35 P4CONL: Port 4 Control Register-Low Byte (0DH, Page 8)

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

.7–.6 P4.3/INT11/AD3 Configuration Bits

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (AD3)
1	1	Output mode, push-pull

.5–.4 P4.2/INT10/AD2 Configuration Bits

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (AD2)
1	1	Output mode, push-pull

.3–.2 P4.1/INT9/AD1 Configuration Bits

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (AD1)
1	1	Output mode, push-pull

.1–.0 P4.0/INT8/AD0 Configuration Bits

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (AD0)
1	1	Output mode, push-pull

4.1.36 P4INTH: Port 4 Interrupt Control Register-High Byte (0EH, Page 8)

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

.7–.6 **P4.7/External interrupt (INT15) Enable Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.5–.4 **P4.6/External interrupt (INT14) Enable Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.3–.2 **P4.5/External interrupt (INT13) Enable Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.1–.0 **P4.4/External interrupt (INT12) Enable Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

4.1.37 P4INTL: Port 4 Interrupt Control Register-Low Byte (0FH, Page 8)

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

.7–.6 **P4.3/External interrupt (INT11) Enable Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.5–.4 **P4.2/External interrupt (INT10) Enable Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.3–.2 **P4.1/External interrupt (INT9) Enable Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.1–.0 **P4.0/External interrupt (INT8) Enable Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

4.1.38 P4PND: Port 4 Interrupt Pending Register (10H, Page 8)

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

.7 P4.7/External Interrupt (INT15) Pending Bit

0	Clear pending bit (when write)
1	Interrupt request is pending (when read)

.6 P4.6/External Interrupt (INT14) Pending Bit

0	Clear pending bit (when write)
1	Interrupt request is pending (when read)

.5 P4.5/External Interrupt (INT13) Pending Bit

0	Clear pending bit (when write)
1	Interrupt request is pending (when read)

.4 P4.4/External Interrupt (INT12) Pending Bit

0	Clear pending bit (when write)
1	Interrupt request is pending (when read)

.3 P4.3/External Interrupt (INT11) Pending Bit

0	Clear pending bit (when write)
1	Interrupt request is pending (when read)

.2 P4.2/External Interrupt (INT10) Pending Bit

0	Clear pending bit (when write)
1	Interrupt request is pending (when read)

.1 P4.1/External Interrupt (INT9) Pending Bit

0	Clear pending bit (when write)
1	Interrupt request is pending (when read)

.0 P4.0/External Interrupt (INT8) Pending Bit

0	Clear pending bit (when write)
1	Interrupt request is pending (when read)

4.1.39 P5CONH: Port 5 Control Register-High Byte (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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.6 **P5.7/XTIN Configuration Bits**

0	0	Input mode
0	1	Input mode, pull-up
1	0	Alternative function (XT_{IN})
1	1	Output mode, push-pull

.5–.4 **P5.6/XTOUT Configuration Bits**

0	0	Input mode
0	1	Input mode, pull-up
1	0	Alternative function (XT_{OUT})
1	1	Output mode, push-pull

.3–.2 **P5.5/CB (NOTE) Configuration Bits**

0	0	Input mode
0	1	Input mode, pull-up
1	0	Not available
1	1	Output mode, push-pull

.1–.0 **P5.4/CA (NOTE) Configuration Bits**

0	0	Input mode
0	1	Input mode, pull-up
1	0	Not available
1	1	Output mode, push-pull

NOTE: Refer to LCON register in Chapter 15

4.1.40 P5CONL: Port 5 Control Register-Low Byte (F1H, Set 1, Bank 1)

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

.7–.6 **P5.3/VLC3 (NOTE) Configuration Bits**

0	0	Input mode
0	1	Input mode, pull-up
1	0	Not available
1	1	Output mode, push-pull

.5–.4 **P5.2/VLC2 (NOTE) Configuration Bits**

0	0	Input mode
0	1	Input mode, pull-up
1	0	Not available
1	1	Output mode, push-pull

.3–.2 **P5.1/VLC1 (NOTE) Configuration Bits**

0	0	Input mode
0	1	Input mode, pull-up
1	0	Not available
1	1	Output mode, push-pull

.1–.0 **P5.0/VLC0 (NOTE) Configuration Bits**

0	0	Input mode
0	1	Input mode, pull-up
1	0	Not available
1	1	Output mode, push-pull

NOTE: Refer to LCON register in Chapter15

4.1.41 P6CONH: Port 6 Control Register-High Byte (F2H, Set 1, Bank 1)

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

.7–.4 **Not used for the S3F8S6B**

.3–.2 **P6.5/SEG33 Configuration Bits**

0	0	Input mode
0	1	Not available
1	0	Alternative function (LCD signal)
1	1	Output mode

.1–.0 **P6.4/SEG32 Configuration Bits**

0	0	Input mode
0	1	Not available
1	0	Alternative function (LCD signal)
1	1	Output mode

4.1.42 P6CONL: Port 6 Control Register-Low Byte (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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.6 P6.3/SEG31 Configuration Bits

0	0	Input mode
0	1	Not available
1	0	Alternative function (LCD signal)
1	1	Output mode

.5–.4 P6.2/SO/SEG30 Configuration Bits

0	0	Input mode
0	1	Alternative function (SO)
1	0	Alternative function (LCD signal)
1	1	Output mode

.3–.2 P6.1/SI/SEG29 Configuration Bits

0	0	Input mode (SI)
0	1	Not available
1	0	Alternative function (LCD signal)
1	1	Output mode

.1–.0 P6.0/SCK/SEG28 Configuration Bits

0	0	Input mode (SCK)
0	1	Alternative function (SCK out)
1	0	Alternative function (LCD signal)
1	1	Output mode

4.1.43 P6PUR: Port 6 Pull-up Resistor Enable Register (F4H, Set 1, Bank 1)

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

.7–.6 **Not used for the S3F8S6B**

.5 **P6.5 Pull-up Resistor Enable Bit**

0	Pull-up disable
1	Pull-up enable

.4 **P6.4 Pull-up Resistor Enable Bit**

0	Pull-up disable
1	Pull-up enable

.3 **P3.3 Pull-up Resistor Enable Bit**

0	Pull-up disable
1	Pull-up enable

.2 **P6.2 Pull-up Resistor Enable Bit**

0	Pull-up disable
1	Pull-up enable

.1 **P6.1 Pull-up Resistor Enable Bit**

0	Pull-up disable
1	Pull-up enable

.0 **P6.0 Pull-up Resistor Enable Bit**

0	Pull-up disable
1	Pull-up enable

NOTE: A pull-up resistor of port 6 is automatically disabled only when the corresponding pin is selected as push-pull output or alternative function.

4.1.44 PNE6: Port 6 N-channel Open-drain Mode Register (F5H, Set 1, Bank 1)

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

.7–.6 **Not used for the S3F8S6B**

.5 **P6.5 Output Mode Selection Bit**

0	Output mode, push-pull
1	Output mode, open-drain

.4 **P6.4 Output Mode Selection Bit**

0	Output mode, push-pull
1	Output mode, open-drain

.3 **P6.3 Output Mode Selection Bit**

0	Output mode, push-pull
1	Output mode, open-drain

.2 **P6.2 Output Mode Selection Bit**

0	Output mode, push-pull
1	Output mode, open-drain

.1 **P6.1 Output Mode Selection Bit**

0	Output mode, push-pull
1	Output mode, open-drain

.0 **P6.0 Output Mode Selection Bit**

0	Output mode, push-pull
1	Output mode, open-drain

4.1.45 PGCON: Pattern Generation Module Control Register (D0H, Set 1, Bank 1)

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

.7–.4 **Not used for the S3F8S6B**

.3 **S/W Trigger Start Bit**

0	No effect
1	S/W trigger start (auto clear)

.5 **PG Operation Disable/Enable Selection Bit**

0	PG operation disable
1	PG operation Enable

.1–.0 **Detection Voltage Selection Bits**

0	0	Timer A match signal trigging
0	1	Timer B overflow signal trigging
1	0	Timer D0 match signal trigging
1	1	S/W trigging

4.1.46 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.4 Destination Register Page Selection Bits

0	0	0	0	Destination: page 0
0	0	0	1	Destination: page 1
0	0	1	0	Destination: page 2
0	0	1	1	Destination: page 3
0	1	0	0	Destination: page 4
0	1	0	1	Destination: page 5
0	1	1	0	Destination: page 6
0	1	1	1	Destination: page 7
1	0	0	0	Destination: page 8
Others				Not used for the S3F8S6B

.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	Source: page 2
0	0	1	1	Source: page 3
0	1	0	0	Source: page 4
0	1	0	1	Source: page 5
0	1	1	0	Source: page 6
0	1	1	1	Source: page 7
1	0	0	0	Source: page 8
Others				Not used for the S3F8S6B

NOTE:

1. In the S3F8S6B microcontroller, the internal register file is configured as nine pages (Pages 0-7, 8). The pages 0-7 are used for general purpose register file.
2. The page 8 of S3F8S6B is used for LCD data register (30H to 51H) and system register (00H to 2FH).

4.1.47 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	RW	RW	RW	RW	RW	–	–	–
Addressing Mode	Register addressing only							

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

Register pointer 0 can independently point to one of the 256 byte working register areas in the register file. Using the register pointers RP0 and RP1, you can select two 8 byte register slices at one time as active working register space. After a reset, RP0 points to address C0H in register set 1, selecting the 8 byte working register slice C0H–C7H.

.2–.0 **Not used for the S3F8S6B**

4.1.48 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	RW	RW	RW	RW	RW	–	–	–
Addressing Mode	Register addressing only							

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

Register pointer 1 can independently point to one of the 256 byte working register areas in the register file. Using the register pointers RP0 and RP1, you can select two 8 byte register slices at one time as active working register space. After a reset, RP1 points to address C8H in register set 1, selecting the 8 byte working register slice C8H–CFH.

.2 – .0 **Not used for the S3F8S6B**

4.1.49 SIOCON: SIO Control Register (E7H, Set 1, Bank 0)

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

.7 SIO Shift Clock Selection Bit

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

.6 Data Direction Control Bit

0	MSB-first mode
1	LSB-first mode

.5 SIO Mode Selection Bit

0	Receive-only mode
1	Transmit/Receive mode

.4 Shift Clock Edge Selection Bit

0	Tx at falling edges, Rx at rising edges
1	Tx at rising edges, Rx at falling edges

.3 SIO Counter Clear and Shift Start Bit

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

.2 SIO Shift Operation Enable Bit

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

.1 SIO Interrupt Enable Bit

0	Disable SIO Interrupt
1	Enable SIO Interrupt

.0 SIO Interrupt Pending Bit

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

4.1.50 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

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

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

4.1.51 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

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

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

4.1.52 STPCON: Stop Control Register (F5H, Set 1, Bank 0)

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

.7–.0 STOP Control Bits

1 0 1 0 0 1 0 1	Enable stop instruction
Other values	Disable stop instruction

NOTE: Before execute the STOP instruction. You must set this STPCON register as "10100101b". Otherwise the STOP instruction will not execute as well as reset will be generated.

4.1.53 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	RW	–	–	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7 **Not used, But you must keep "0"**

.6–.5 **Not used for the S3F8S6B**

.4–.2 **Fast Interrupt Level Selection Bits (1)**

0	0	0	IRQ0
0	0	1	IRQ1
0	1	0	IRQ2
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

.1 **Fast Interrupt Enable Bit (2)**

0	Disable fast interrupt processing
1	Enable fast interrupt processing

.0 **Global Interrupt Enable Bit (3)**

0	Disable all interrupt processing
1	Enable all interrupt processing

NOTE:

1. You can select only one interrupt level at a time for fast interrupt processing.
2. Setting SYM.1 to "1" enables fast interrupt processing for the interrupt level currently selected by SYM.2–SYM.4.
3. Following a reset, you must enable global interrupt processing by executing an EI instruction (not by writing a "1" to SYM.0).

4.1.54 TACON: Timer A Control Register (E2H, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.5 Timer A Input Clock Selection Bits

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

.4–.3 Timer A Operating Mode Selection Bits

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

.2 Timer A Counter Enable Bit

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

.1 Timer A Match/Capture Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Timer A Overflow Interrupt Enable Bit

0	Disable overflow interrupt
1	Enable overflow interrupt

NOTE: The TACON.2 value is automatically cleared to "0" after being cleared counter.

4.1.55 TBCON: Timer B Control Register (E3H, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.6 Timer B Input Clock Selection Bits

0	0	fxx/1
0	1	fxx/2
1	0	fxx/4
1	1	fxx/8

.5–.4 Timer B Interrupt Time Selection Bits

0	0	Generating after low data is borrowed
0	1	Generating after high data is borrowed
1	0	Generating after low and high data are borrowed
1	1	Not available

.3 Timer B Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.2 Timer B Start/Stop Bit

0	Stop timer B
1	Start timer B

.1 Timer B Mode Selection Bit

0	One-shot mode
1	Repeat mode

.0 Timer B Output Flip-flop Control Bit

0	TBOF is low (TBPWM: low level for low data, high level for high data)
1	TBOF is high (TBPWM: high level for low data, low level for high data)

4.1.56 TCCON: Timer C Control Register (ECH, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7 Timer C Start/Stop Bit

0	Stop Timer C
1	Start Timer C

.6–.4 Timer C 3-Bits Prescaler Bits

0	0	0	Non divided
0	0	1	Divided by 2
0	1	0	Divided by 3
0	1	1	Divided by 4
1	0	0	Divided by 5
1	0	1	Divided by 6
1	1	0	Divided by 7
1	1	1	Divided by 8

.3 Timer C Counter Clear Bit

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

.2 Timer C Mode Selection Bit

0	fxx/1 & PWM mode
1	fxx/64 & interval mode

.1 Timer C Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Timer C Interrupt Pending Bit

0	No interrupt pending (when read), clear pending bit (when write)
1	Interrupt is pending (when read)

NOTE: The TCCON.3 value is automatically cleared to "0" after being cleared counter.

4.1.57 TD0CON: Timer D0 Control Register (FAH, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.5 Timer D0 Input Clock Selection Bits

0	0	0	fxx/1024
0	0	1	fxx/256
0	1	0	fxx/64
0	1	1	fxx/8
1	0	0	fxx/1
1	0	1	External clock (TD0CLK) falling edge
1	1	0	External clock (TD0CLK) rising edge
1	1	1	Counter stop

.4–.3 Timer D0 Operating Mode Selection Bits

0	0	Interval mode (TD0OUT)
0	1	Capture mode (Capture on rising edge, counter running, OVF can occur)
1	0	Capture mode (Capture on falling edge, counter running, OVF can occur)
1	1	PWM mode (OVF and match interrupt can occur)

.2 Timer D0 Counter Clear Bit

0	No effect
1	Clear the timer D0 counter (when write)

.1 Timer D0 Match/Capture Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Timer D0 Overflow Interrupt Enable Bit

0	Disable overflow interrupt
1	Enable overflow interrupt

NOTE: The TD0CON.2 value is automatically cleared to "0" after being cleared counter.

4.1.58 TD1CON: Timer D1 Control Register (FBH, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7–.5 Timer D1 Input Clock Selection Bits

0	0	0	fxx/1024
0	0	1	fxx/256
0	1	0	fxx/64
0	1	1	fxx/8
1	0	0	fxx/1
1	0	1	External clock (TD1CLK) falling edge
1	1	0	External clock (TD1CLK) rising edge
1	1	1	Counter stop

.4–.3 Timer D1 Operating Mode Selection Bits

0	0	Interval mode (TD1OUT)
0	1	Capture mode (Capture on rising edge, counter running, OVF can occur)
1	0	Capture mode (Capture on falling edge, counter running, OVF can occur)
1	1	PWM mode (OVF and match interrupt can occur)

.2 Timer D1 Counter Clear Bit

0	No effect
1	Clear the timer D1 counter (when write)

.1 Timer D1 Match/Capture Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Timer D1 Overflow Interrupt Enable Bit

0	Disable overflow interrupt
1	Enable overflow interrupt

NOTE: The TD1CON.2 value is automatically cleared to "0" after being cleared counter.

4.1.59 UART0CONH: UART 0 Control Register-High Byte (14H, Page 8)

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

.7–.6 UART 0 Mode Selection Bits

0	0	Mode 0: shift register (fU/(16 × (BRDATA0+1)))
0	1	Mode 1: 8-bit UART (fU/(16 × (BRDATA0+1)))
1	0	Mode 2: 9-bit UART (fU/16)
1	1	Mode 3: 9-bit UART (fU/(16 × (BRDATA0+1)))

.5 Multiprocessor Communication Enable Bit (for modes 2 and 3 only)

0	Disable
1	Enable

.4 Serial Data Receive Enable Bit

0	Disable
1	Enable

.3 TB8 (Only when UART0CONL.7 = 0)

Location of the 9th data bit to be transmitted in UART 0 mode 2 or 3 ("0" or "1")

NOTE: If the UART0CONL.7 = 1, This bit is "don't care".

.2 RB8 (Only when UART0CONL.7 = 0)

Location of the 9th data bit that was received in UART 0 mode 2 or 3 ("0" or "1")

NOTE: If the UART0CONL.7 = 1, This bit is "don't care".

.1 UART 0 Receive Interrupt Enable Bit

0	Disable Rx interrupt
1	Enable Rx interrupt

.0 UART 0 Receive Interrupt Pending Bit

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

NOTE:

1. In mode 2 and 3, if the MCE bit is set to "1" then the receive interrupt will not be activated if the received 9th data bit "0". In mode 1, if MCE = "1" the receive interrupt will not be activated if a valid stop bit was not received. In mode 0, the MCE bit should be "0".
2. The descriptions for 8-bit and 9-bit UART mode do not include start and stop bits for serial data receive and transmit.

4.1.60 UART0CONL: UART 0 Control Register-Low Byte (15H, Page 8)

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

.7 UART 0 Transmit Parity-bit Auto-Generation Enable Bit (for modes 2 and 3 only)

0	Disable parity-bit auto-generation
1	Enable parity-bit auto-generation

.6 UART 0 Transmit Parity-bit Selection Bit (for modes 2 and 3 only)

0	Even parity-bit
1	Odd parity-bit

NOTE: If the UART0CONL.7 = 0, This bit is "don't care".

.5 UART 0 Receive Parity-bit Selection Bit (for modes 2 and 3 only)

0	Even parity-bit check
1	Odd parity-bit check

NOTE: If the UART0CONL.7 = 0, This bit is "don't care".

.4 UART 0 Receive Parity-bit Error Status Bit (for modes 2 and 3 only)

0	No parity-bit error
1	Parity-bit error

NOTE: If the UART0CONL.7 = 0, This bit is "don't care".

.3–.2 UART 0 Clock Selection Bits

0	0	fxx/8
0	1	fxx/4
1	0	fxx/2
1	1	fxx/1

.1 UART 0 Transmit Interrupt Enable Bit

0	Disable Tx interrupt
1	Enable Tx interrupt

.0 UART 0 Transmit Interrupt Pending Bit

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

4.1.61 UART1CONH: UART 1 Control Register-High Byte (18H, Page 8)

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

.7–.6 UART 1 Mode Selection Bits

0	0	Mode 0: shift register (fU/(16 × (BRDATA1+1)))
0	1	Mode 1: 8-bit UART (fU/(16 × (BRDATA1+1)))
1	0	Mode 2: 9-bit UART (fU/16)
1	1	Mode 3: 9-bit UART (fU/(16 × (BRDATA1+1)))

.5 Multiprocessor Communication Enable Bit (for modes 2 and 3 only)

0	Disable
1	Enable

.4 Serial Data Receive Enable Bit

0	Disable
1	Enable

.3 TB8 (Only when UART1CONL.7 = 0)

Location of the 9th data bit to be transmitted in UART 1 mode 2 or 3 ("0" or "1")

NOTE: If the UART1CONL.7 = 1, This bit is "don't care".

.2 RB8 (Only when UART1CONL.7 = 0)

Location of the 9th data bit that was received in UART 1 mode 2 or 3 ("0" or "1")

NOTE: If the UART1CONL.7 = 1, This bit is "don't care".

.1 UART 1 Receive Interrupt Enable Bit

0	Disable Rx interrupt
1	Enable Rx interrupt

.0 UART 1 Receive Interrupt Pending Bit

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

NOTE:

1. In mode 2 and 3, if the MCE bit is set to "1" then the receive interrupt will not be activated if the received 9th data bit "0". In mode 1, if MCE = "1" the receive interrupt will not be activated if a valid stop bit was not received. In mode 0, the MCE bit should be "0".
2. The descriptions for 8-bit and 9-bit UART mode do not include start and stop bits for serial data receive and transmit.

4.1.62 UART1CONL: UART 1 Control Register-Low Byte (19H, Page 8)

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

.7 UART 1 Transmit Parity-bit Auto-Generation Enable Bit (for modes 2 and 3 only)

0	Disable parity-bit auto-generation
1	Enable parity-bit auto-generation

.6 UART 1 Transmit Parity-bit Selection Bit (for modes 2 and 3 only)

0	Even parity-bit
1	Odd parity-bit

NOTE: If the UART1CONL.7 = 0, This bit is "don't care".

.5 UART 1 Receive Parity-bit Selection Bit (for modes 2 and 3 only)

0	Even parity-bit check
1	Odd parity-bit check

NOTE: If the UART1CONL.7 = 0, This bit is "don't care".

.4 UART 1 Receive Parity-bit Error Status Bit (for modes 2 and 3 only)

0	No parity-bit error
1	Parity-bit error

NOTE: If the UART1CONL.7 = 0, This bit is "don't care".

.3–.2 UART 1 Clock Selection Bits

0	0	fxx/8
0	1	fxx/4
1	0	fxx/2
1	1	fxx/1

.1 UART 1 Transmit Interrupt Enable Bit

0	Disable Tx interrupt
1	Enable Tx interrupt

.0 UART 1 Transmit Interrupt Pending Bit

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

4.1.63 WTCON: Watch Timer Control Register (E6H, 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	RW	RW	RW	RW	RW	RW	RW	RW
Addressing Mode	Register addressing mode only							

.7 Watch Timer Clock Selection Bit

0	Main system clock divided by 2^7 (fxx/128)
1	Sub system clock (fxt)

.6 Watch Timer Interrupt Enable Bit

0	Disable watch timer interrupt
1	Enable watch timer interrupt

.5–.4 Buzzer Signal Selection Bits

0	0	0.5 kHz
0	1	1 kHz
1	0	2 kHz
1	1	4 kHz

.3–.2 Watch Timer Speed Selection Bits

0	0	Set watch timer interrupt to 0.5s
0	1	Set watch timer interrupt to 0.25s
1	0	Set watch timer interrupt to 0.125s
1	1	Set watch timer interrupt to 1.995 ms

.1 Watch Timer Enable Bit

0	Disable watch timer; Clear frequency dividing circuits
1	Enable watch timer

.0 Watch Timer Interrupt Pending Bit

0	No interrupt pending (when read), clear pending bit (when write)
1	Interrupt is pending (when read)

NOTE: Watch timer clock frequency (fw) is assumed to be 32.768 kHz.

5 Interrupt Structure

5.1 Overview

The S3C8-series interrupt structure has three basic components: levels, vectors, and sources. The SAM8 CPU recognizes up to eight interrupt levels and supports up to 128 interrupt vectors. When a specific interrupt level has more than one vector address, the vector priorities are established in hardware. A vector address can be assigned to one or more sources.

Levels

Interrupt levels are the main unit for interrupt priority assignment and recognition. All peripherals and I/O blocks can issue interrupt requests. In other words, peripheral and I/O operations are interrupt-driven. There are eight possible interrupt levels: IRQ0–IRQ7, also called level 0–level 7. Each interrupt level directly corresponds to an interrupt request number (IRQn). The total number of interrupt levels used in the interrupt structure varies from device to device. The S3F8S6B interrupt structure recognizes eight interrupt levels.

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

Vectors

Each interrupt level can have one or more interrupt vectors, or it may have no vector address assigned at all. The maximum number of vectors that can be supported for a given level is 128 (The actual number of vectors used for S3C8-series devices is always much smaller). If an interrupt level has more than one vector address, the vector priorities are set in hardware. S3F8S6B uses thirty vectors.

Sources

A source is any peripheral that generates an interrupt. A source can be an external pin or a counter overflow. Each vector can have several interrupt sources. In the S3F8S6B interrupt structure, there are thirty possible interrupt sources.

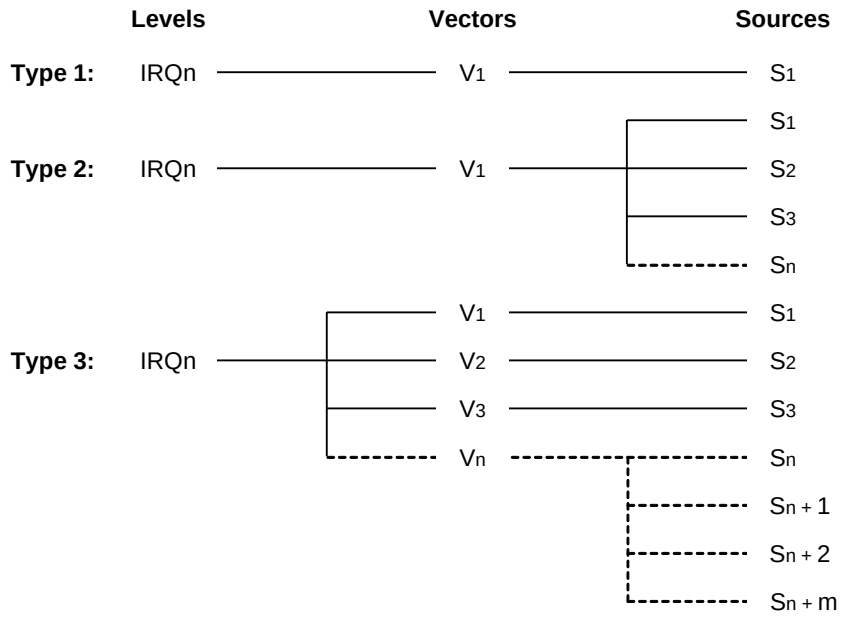
When a service routine starts, the respective pending bit should be either cleared automatically by hardware or cleared "manually" by program software. The characteristics of the source's pending mechanism determine which method would be used to clear its respective pending bit.

5.2 Interrupt Types

The three components of the S3C8 interrupt structure described before — 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 S3F8S6B microcontroller, two interrupt types are implemented.



NOTES:
1. The number of S_n and V_n value is expandable.
2. In the S3F8S6B implementation, interrupt types 1 and 3 are used.

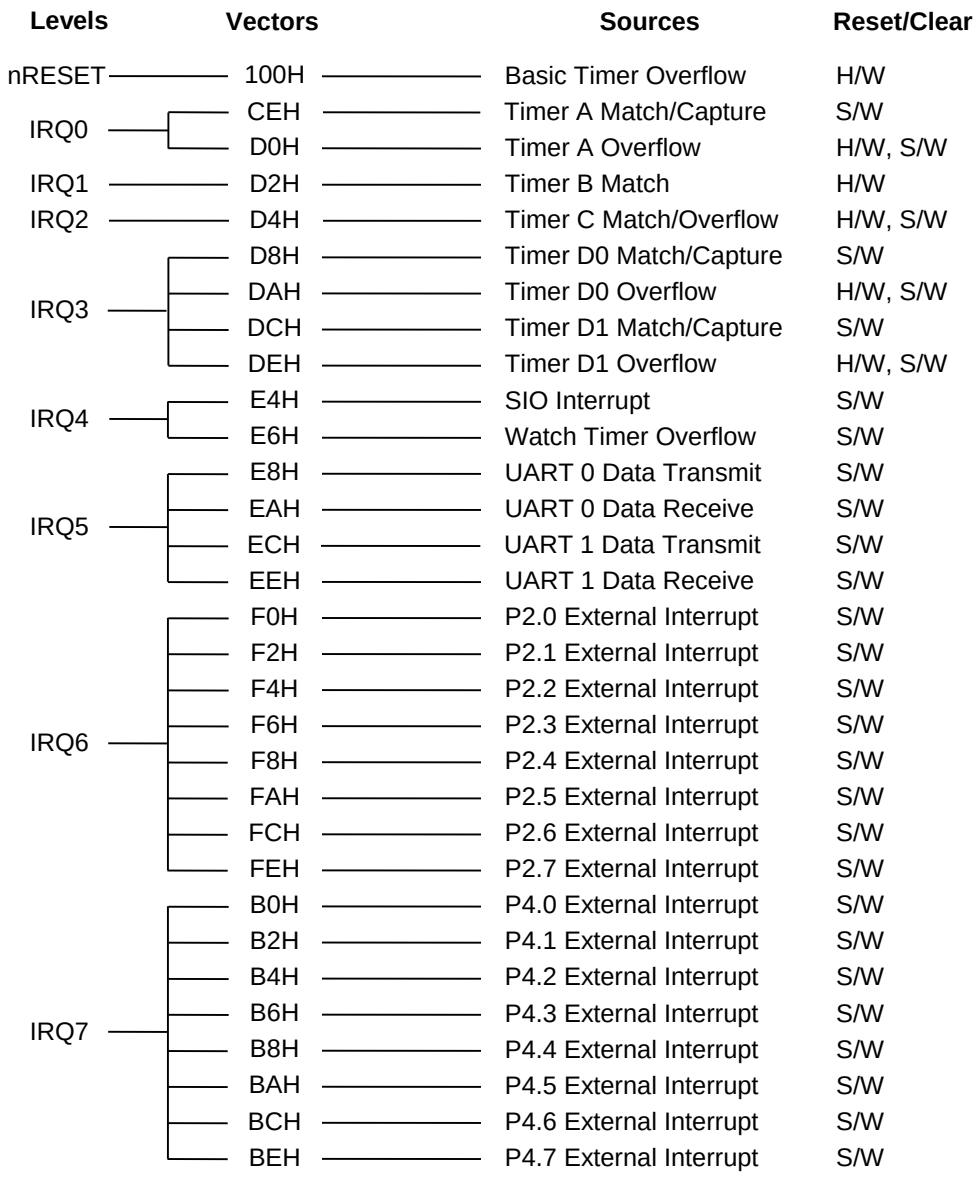
Figure 5-1 S3C8-Series Interrupt Types

5.3 S3F8S6B Interrupt Structure

The S3F8S6B microcontroller supports thirty interrupt sources. All thirty of the interrupt sources have a corresponding interrupt vector address. Eight interrupt levels are recognized by the CPU in this device-specific interrupt structure, as shown in *Figure 5-2*.

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

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



NOTES:

1. Within a given interrupt level, the low vector address has high priority. For example, CEH has higher priority than D0H within the level IRQ0 the priorities within each level are set at the factory.
2. External interrupts are triggered by a rising or falling edge, depending on the corresponding control register setting.

Figure 5-2 S3F8S6B Interrupt Structure

5.4 Interrupt Vector Addresses

All interrupt vector addresses for the S3F8S6B interrupt structure are stored in the vector address area of the internal 64KB ROM, 0H–FFFFH.; (see *Figure 5-3*).

You can allocate unused locations in the vector address area as normal program memory. If you do so, please be careful not to overwrite any of the stored vector addresses (*Table 5.1* lists all vector addresses).

The program reset address in the ROM is 0100H.

The reset address of ROM can be changed by Smart Option in the S3F8S6B (full-Flash device). Refer to the Chapter 21. Embedded Flash Memory Interface for more detailed contents.

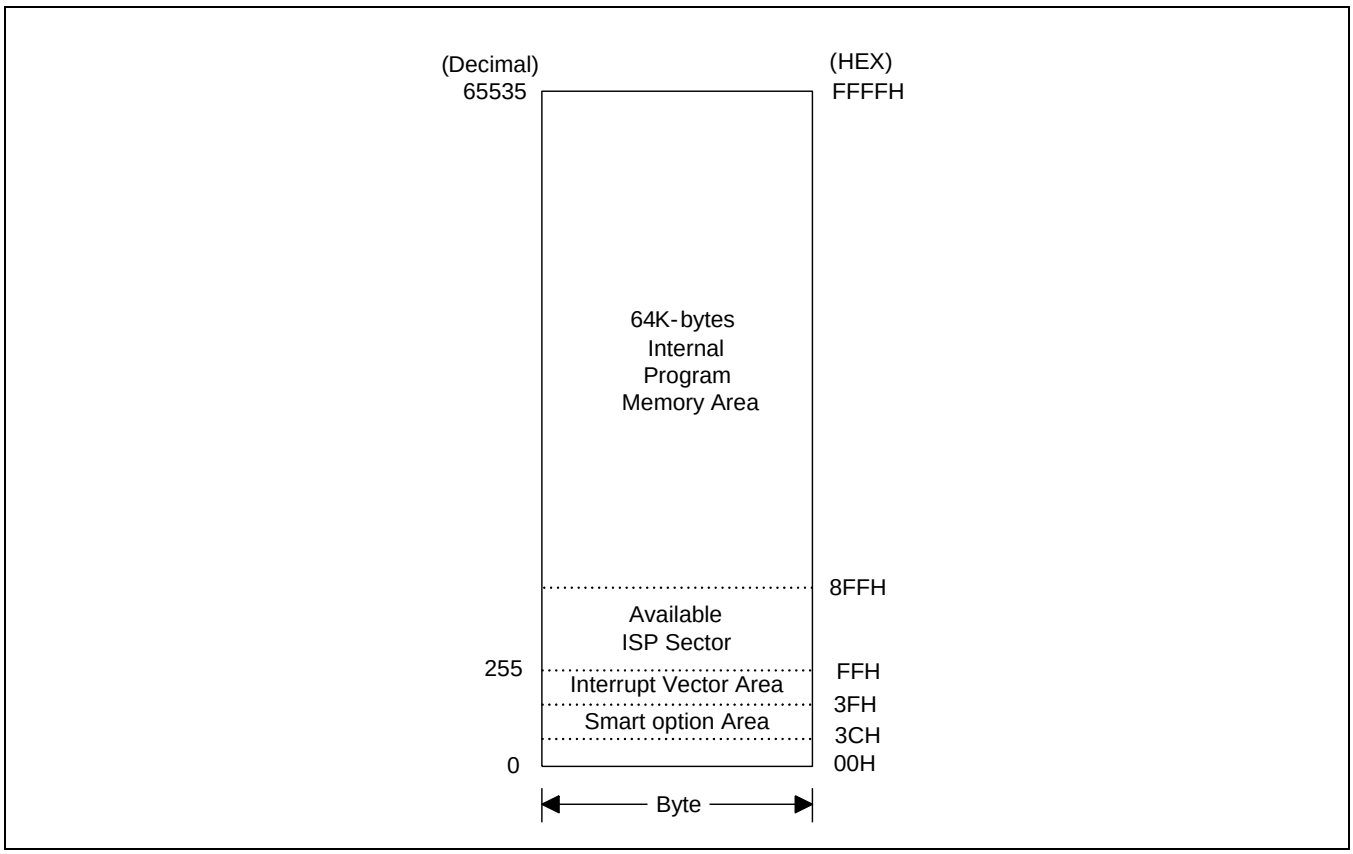


Figure 5-3 ROM Vector Address Area

Table 5.1 Interrupt Vectors

Vector Address		Interrupt Source	Request		Reset/Clear	
Decimal Value	Hex Value		Interrupt Level	Priority in Level	H/W	S/W
256	100H	Basic timer overflow	Reset	–	√	
206	CEH	Timer A match/capture	IRQ0	0		√
208	D0H	Timer A overflow	–	1	√	√
210	D2H	Timer B match	IRQ1	–	√	
212	D4H	Timer C match/overflow	IRQ2	–	√	√
216	D8H	Timer D0 match/capture	IRQ3	0		√
218	DAH	Timer D0 overflow	–	1	√	√
220	DCH	Timer D1 match/capture	–	2		√
222	DEH	Timer D1 overflow	–	3	√	√
228	E4H	SIO interrupt	IRQ4	0		√
230	E6H	Watch timer overflow	–	1		√
232	E8H	UART 0 data transmit	IRQ5	0		√
234	EAH	UART 0 data receive	–	1		√
236	ECH	UART 1 data transmit	–	2		√
238	EEH	UART 1data receive	–	3		√
240	F0H	P2.0 external interrupt	IRQ6	0		√
242	F2H	P2.1 external interrupt	–	1		√
244	F4H	P2.2 external interrupt	–	2		√
246	F6H	P2.3 external interrupt	–	3		√
248	F8H	P2.4 external interrupt	–	4		√
250	FAH	P2.5 external interrupt	–	5		√
252	FCH	P2.6 external interrupt	–	6		√
254	FEH	P2.7 external interrupt	–	7		√
176	B0H	P4.0 external interrupt	IRQ7	0		√
178	B2H	P4.1 external interrupt	–	1		√
180	B4H	P4.2 external interrupt	–	2		√
182	B6H	P4.3 external interrupt	–	3		√
184	B8H	P4.4 external interrupt	–	4		√
186	BAH	P4.5 external interrupt	–	5		√
188	BCH	P4.6 external interrupt	–	6		√
190	BEH	P4.7 external interrupt	–	7		√

NOTE:

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 given level are fixed in hardware.

5.5 Enable/Disable Interrupt Instructions (EI, DI)

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

NOTE: The system initialization routine executed after a reset must always contain an EI instruction to globally enable the interrupt structure.

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.

5.6 System-Level Interrupt Control Registers

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

- The interrupt mask register, IMR, enables (un-masks) or disables (masks) interrupt levels.
- The interrupt priority register, IPR, controls the relative priorities of interrupt levels.
- The interrupt request register, IRQ, contains interrupt pending flags for each interrupt level (as opposed to each interrupt source).
- The system mode register, SYM, enables or disables global interrupt processing (SYM settings also enable fast interrupts and control the activity of external interface, if implemented).

Table 5.2 Interrupt Control Register Overview

Control Register	ID	R/W	Function Description
Interrupt mask register	IMR	R/W	Bit settings in the IMR register enable or disable interrupt processing for each of the eight interrupt levels: IRQ0–IRQ7.
Interrupt priority register	IPR	R/W	Controls the relative processing priorities of the interrupt levels. The seven levels of S3F8S6B are organized into three groups: A, B, and C. Group A is IRQ0 and IRQ1, group B is IRQ2, IRQ3 and IRQ4, and group C is IRQ5, IRQ6, and IRQ7.
Interrupt request register	IRQ	R	This register contains a request pending bit for each interrupt level.
System mode register	SYM	R/W	This register enables/disables fast interrupt processing, dynamic global interrupt processing, and external interface control (An external memory interface is implemented in the S3F8S6B microcontroller).

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

5.7 Interrupt Processing Control Points

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

- Global interrupt enable and disable (by EI and DI instructions or by direct manipulation of SYM.0)
- Interrupt level enable/disable settings (IMR register)
- Interrupt level priority settings (IPR register)
- Interrupt source enable/disable settings in the corresponding peripheral control registers

NOTE: When writing an application program that handles interrupt processing, be sure to include the necessary register file address (register pointer) information.

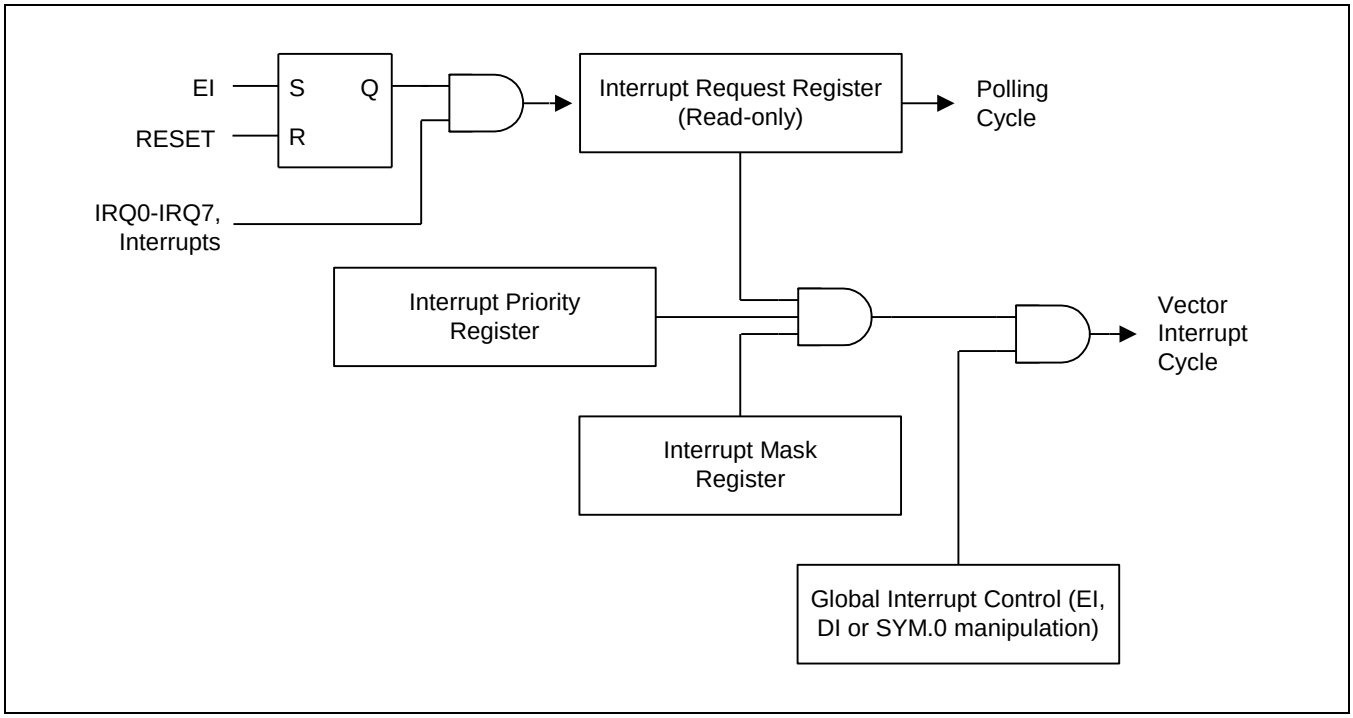


Figure 5-4 Interrupt Function Diagram

5.8 Peripheral Interrupt Control Registers

For each interrupt source there is one or more corresponding peripheral control registers that let you control the interrupt generated by the related peripheral (see *Table 5.3*).

Table 5.3 Interrupt Source Control and Data Registers

Interrupt Source	Interrupt Level	Register(s)	Location(s) in Set 1
Timer A match/capture Timer A overflow	IRQ0	TACON TACNT TADATA	E2H, bank 0 E0H, bank 0 E1H, bank 0
Timer B match	IRQ1	TBCON TBDATAH TBDATAL	E3H, bank 0 E4H, bank 0 E5H, bank 0
Timer C match/overflow	IRQ2	TCCON TCCNT TCDATA	ECH, bank 0 EAH, bank 0 EBH, bank 0
Timer D0 match/capture Timer D0 overflow Timer D1 match/capture Timer D1 overflow	IRQ3	TD0CON TD0CNTH TD0CNTL TD0DATAH TD0DATAL TD1CON TD1CNTH TD1CNTL TD1DATAH TD1DATAL	FAH, bank 1 F6H, bank 1 F7H, bank 1 F8H, bank 1 F9H, bank 1 FBH, bank 1 FCH, bank 1 FDH, bank 1 FEH, bank 1 FFH, bank 1
SIO interrupt Watch timer overflow	IRQ4	SIOCON SIODATA SIOPS WTCON	E7H, bank 0 E8H, bank 0 E9H, bank 0 E6H, bank 0
UART 0 data transmit UART 0 data receive UART 1 data transmit UART 1 data receive	IRQ5	UART0CONH UART0CONL UDATA0 BRDATA0 UART1CONH UART1CONL UDATA1 BRDATA1	14H, page 8 15H, page 8 16H, page 8 17H, page 8 18H, page 8 19H, page 8 1AH, page 8 1BH, page 8
P2.0 external interrupt P2.1 external interrupt P2.2 external interrupt P2.3 external interrupt P2.4 external interrupt P2.5 external interrupt P2.6 external interrupt P2.7 external interrupt	IRQ6	P2CONH P2CONL P2INTH P2INTL P2PND	E6H, bank 1 E7H, bank 1 E8H, bank 1 E9H, bank 1 EAH, bank 1
P4.0 external interrupt P4.1 external interrupt P4.2 external interrupt P4.3 external interrupt P4.4 external interrupt	IRQ7	P4CONH P4CONL P4INTH P4INTL P4PND	0CH, page 8 0DH, page 8 0EH, page 8 0FH, page 8 10H, page 8

Interrupt Source	Interrupt Level	Register(s)	Location(s) in Set 1
P4.5 external interrupt P4.6 external interrupt P4.7 external interrupt			

NOTE: If a interrupt is un-mask (Enable interrupt level) in the IMR register, the pending bit and enable bit of the interrupt should be written after a DI instruction is executed.

During the Port 2 and Port 4 state change, the unexpected external interrupts are occurred.
The unexpected external interrupts are occurred by port 2 and port 4state change. Therefore before port 2 and port 4 states are changed to any value, you can execute the DI, EI instructions and pending bit clear.

The following steps must be taken to change:

1. Use DI instruction.
2. Change P4CONH/L, P4INTH/L
3. Clear Port 4 Interrupt Pending Register (P4PND) to "00000000B"
4. Use EI instruction.

Example 5-1 How to Prevent the Unexpected External Interrupts

1. This example shows how to change from the normal port mode to the interrupt port mode.

```
LD      PP,#10001000B                ; Select page 8
LD      P4CONH,#10101010B            ; P4.7~.4 Alternative function (AD4-AD7)
LD      P4CONL,#10101010B            ; P4.3~.0 Alternative function (AD0-AD3)
•
•
•
DI                    ; for external interrupt setting mode
LD      P4CONH,#01010101B            ; P4.7~.4 Input mode with pull-up for interrupt
LD      P4CONL,# 01010101B           ; P4.3~.0 Input mode with pull-up for interrupt
LD      P4INTH,#01010101B            ; P4.7~.4 Enable interrupt falling edge
LD      P4INTL,# 01010101B           ; P4.3~.0 Enable interrupt falling edge
LD      P4PND,#00000000B             ; P4.7~.0 Interrupt pending bit clear
EI
•
•
```

2. This example shows how to change from the interrupt port mode to the normal port mode.

```
LD      PP,#10001000B                ; Select page 8
LD      P4CONH,# 01010101B           ; P4.7~.4 Input mode with pull-up for interrupt
LD      P4CONL,# 01010101B           ; P4.3~.0 Input mode with pull-up for interrupt
•
•
•
DI                    ; for normal port setting mode
LD      P4CONH,# 10101010B           ; P4.7~.4 Alternative function (AD4-AD7)
LD      P4CONL,# 10101010B           ; P4.3~.0 Alternative function (AD0-AD3)
LD      P4INTH,# 00000000B           ; P4.7~.4 Disable interrupt falling edge
LD      P4INTL,# 00000000B           ; P4.3~.0 Disable interrupt falling edge
LD      P4PND,# 00000000B            ; P4.7~.0 Interrupt pending bit clear
EI
•
```

5.9 System Mode Register (SYM)

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

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

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

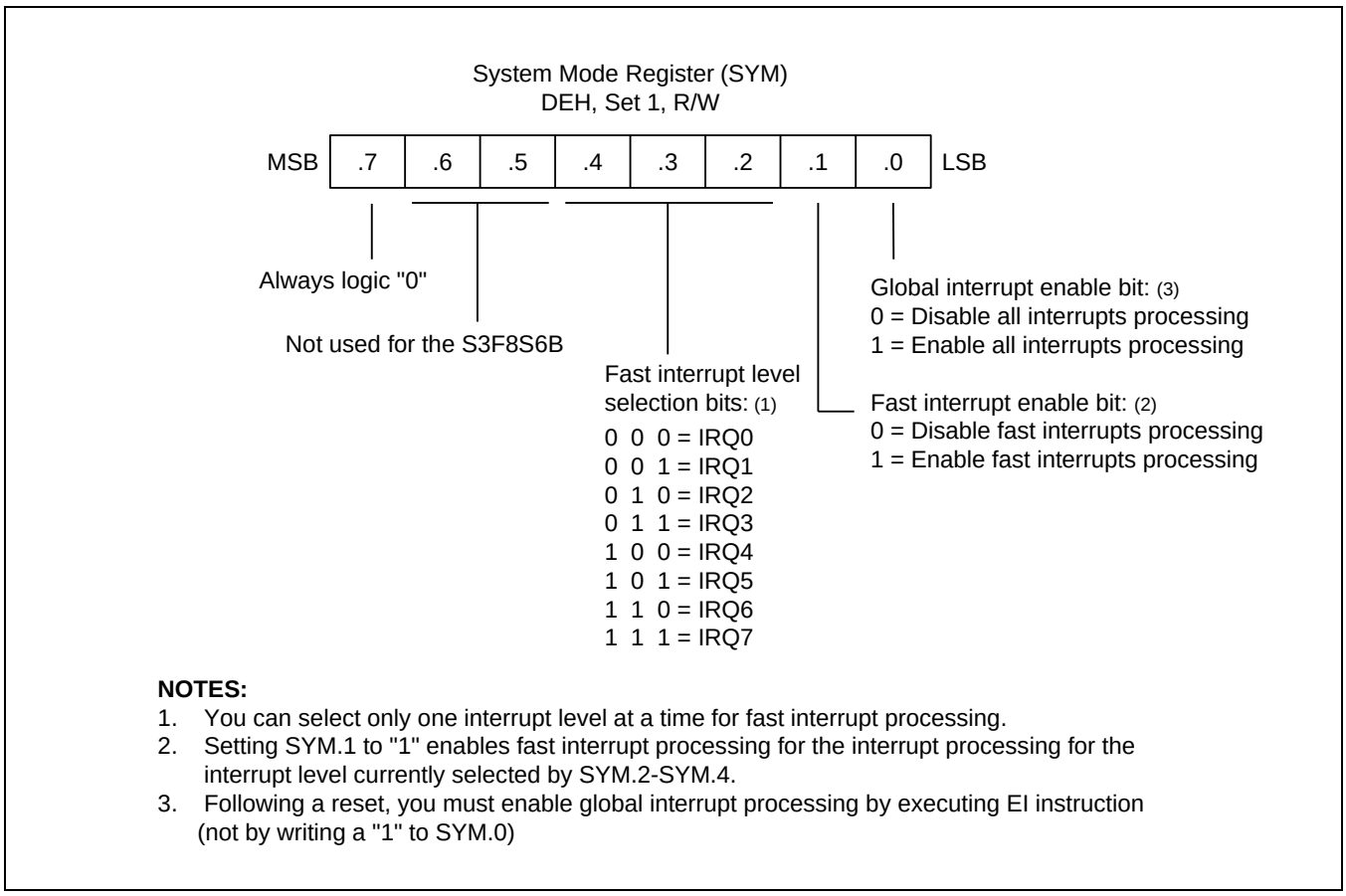


Figure 5-5 System Mode Register (SYM)

5.10 Interrupt Mask Register (IMR)

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

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

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

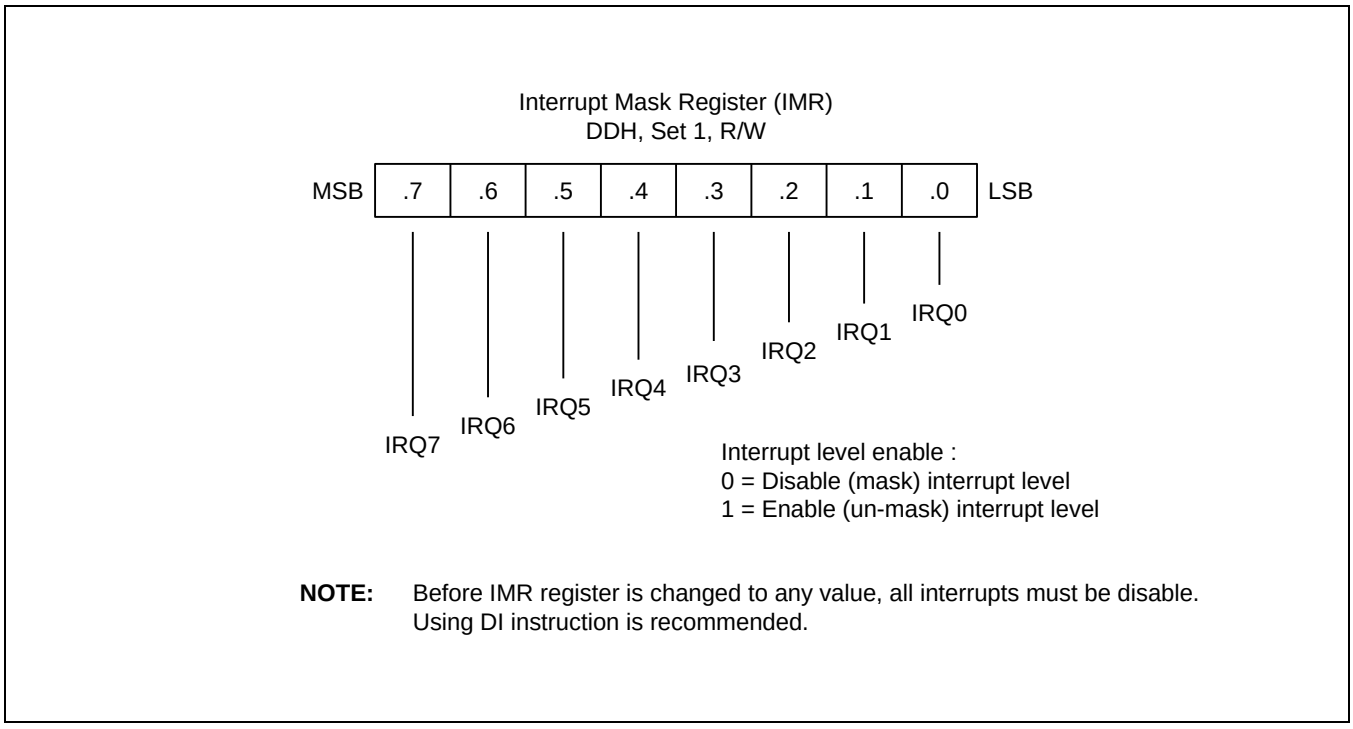


Figure 5-6 Interrupt Mask Register (IMR)

5.11 Interrupt Priority Register (IPR)

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

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

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

- Group A IRQ0, IRQ1
- Group B IRQ2, IRQ3, IRQ4
- Group C IRQ5, IRQ6, IRQ7

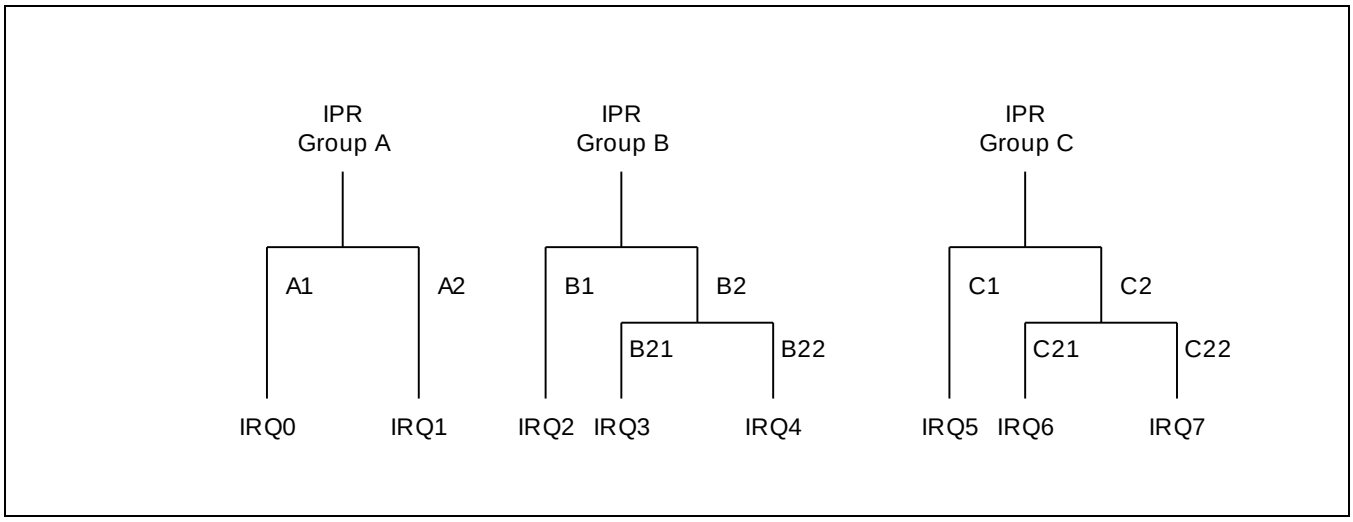


Figure 5-7 Interrupt Request Priority Groups

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

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

- IPR.5 controls the relative priorities of group C interrupts.
- Interrupt group C includes a subgroup that has an additional priority relationship among the interrupt levels 5, 6, and 7. IPR.6 defines the subgroup C relationship. IPR.5 controls the interrupt group C.
- IPR.0 controls the relative priority setting of IRQ0 and IRQ1 interrupts.

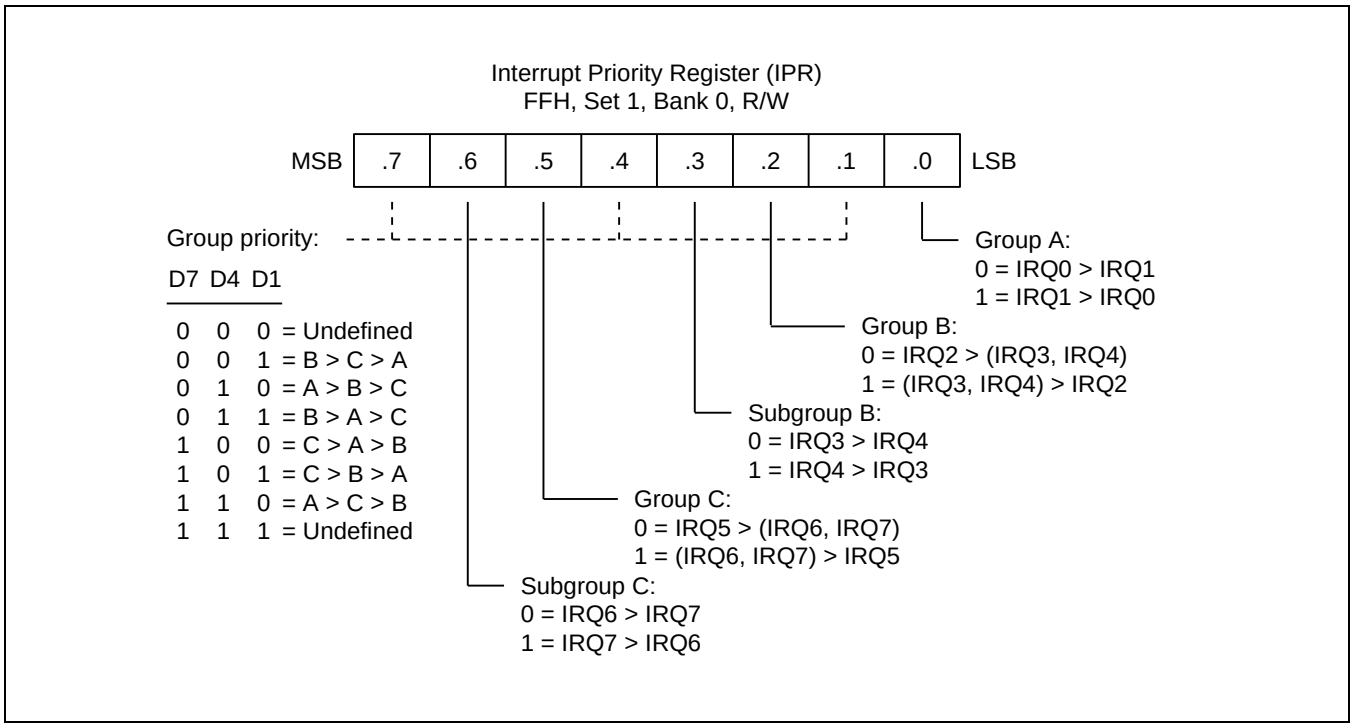


Figure 5-8 Interrupt Priority Register (IPR)

5.12 Interrupt Request Register (IRQ)

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

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

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

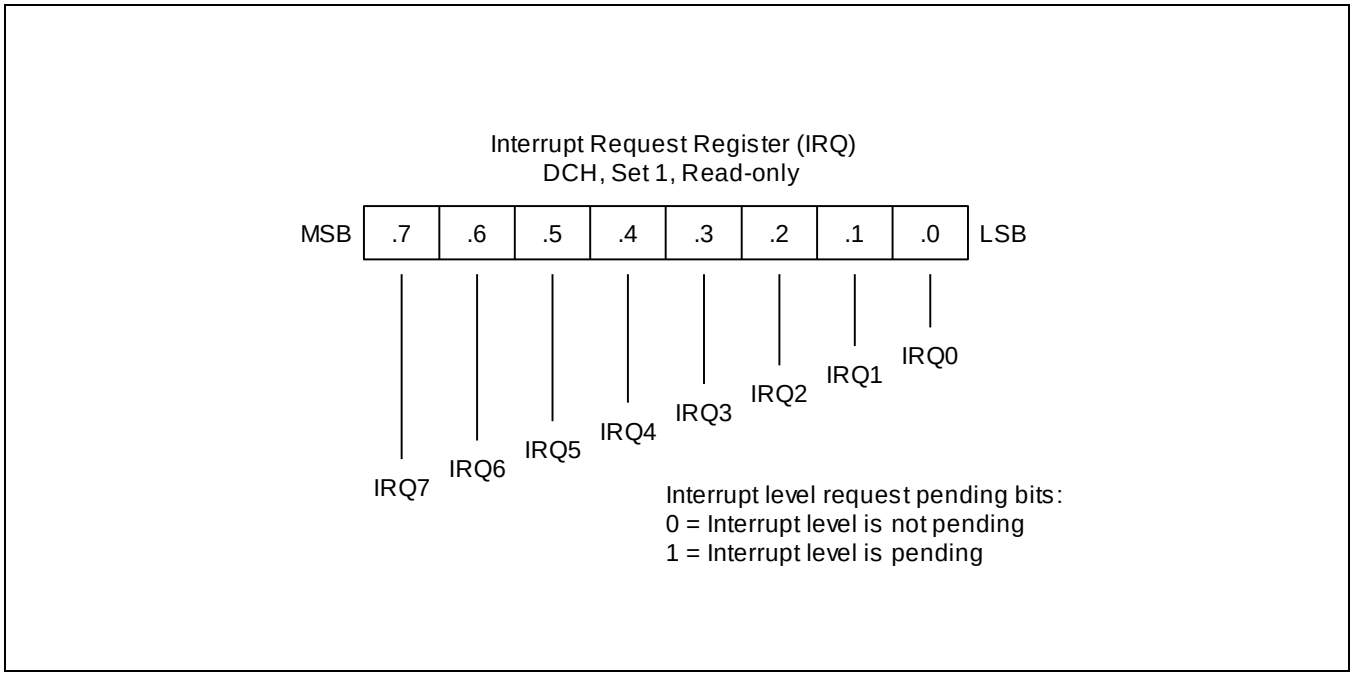


Figure 5-9 Interrupt Request Register (IRQ)

5.13 Interrupt Pending Function Types

5.13.1 Overview

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

5.13.2 Pending Bits Cleared Automatically by Hardware

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

In the S3F8S6B interrupt structure, the timer A overflow interrupt (IRQ0), the timer B match interrupt (IRQ1), the timer C match/overflow interrupt (IRQ2), the timer D0/D1 overflow interrupt (IRQ3) belongs to this category of interrupts in which pending condition is cleared automatically by hardware.

5.13.3 Pending Bits Cleared by the Service Routine

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

Example 5-2 How to Clear an Interrupt Pending Bit

As the following examples are shown, a load instruction should be used to clear an interrupt pending bit.

```
1.    SB1
      LD     P2PND, #11111011B            ; Clear P2.2's interrupt pending bit
      •
      •
      •
      IRET

2.    SB0
      LD     INTPND, #11111101B           ; Clear timer A match/capture interrupt pending bit
      •
      •
      •
      IRET
```

5.14 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 sources interrupt level.
4. The CPU generates an interrupt acknowledge signal.
5. Interrupt logic determines the interrupt's vector address.
6. The service routine starts and the source's pending bit is cleared to "0" (by hardware or by software).
7. The CPU continues polling for interrupt requests.

5.15 Interrupt Service Routines

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

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

When 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 (PC) and status flags to the system stack.
3. Branch to the interrupt vector to fetch the address of the service routine.
4. Pass control to the interrupt service routine.

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

5.16 Generating interrupt Vector Addresses

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

Vectored interrupt processing follows this sequence:

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

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

5.17 Nesting of Vectored Interrupts

It is possible to nest a higher-priority interrupt request while a lower-priority request is being serviced.

To do this, you must follow these steps:

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

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

5.18 Instruction Pointer (IP)

The instruction pointer (IP) is adopted by all the S3C8-series microcontrollers to control the optional high-speed interrupt processing feature called fast interrupts. The IP consists of register pair DAH and DBH. The names of IP registers are IPH (high byte, IP15–IP8) and IPL (low byte, IP7–IP0).

5.19 Fast Interrupt Processing

The feature called fast interrupt processing allows an interrupt within a given level to be completed in approximately 6 clock cycles rather than the usual 16 clock cycles. To select a specific interrupt level for fast interrupt processing, you write the appropriate 3-bit value to SYM.4–SYM.2. Then, to enable fast interrupt processing for the selected level, you set SYM.1 to "1".

Two other system registers support fast interrupt processing:

- The instruction pointer (IP) contains the starting address of the service routine (and is later used to swap the program counter values), and
- When a fast interrupt occurs, the contents of the FLAGS register is stored in an unmapped, dedicated register called FLAGS' ("FLAGS prime").

NOTE: For the S3F8S6B microcontroller, the service routine for any one of the eight interrupts levels: IRQ0–IRQ7 can be selected for fast interrupt processing.

5.20 Procedure for Initiating Fast Interrupts

To initiate fast interrupt processing, follow these steps:

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

5.21 Fast Interrupt Service Routine

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

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

5.22 Relationship to Interrupt Pending Bit Types

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

5.23 Programming Guidelines

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

6 Instruction Set

6.1 Overview

The SAM8 instruction set is specifically designed to support the large register files that are typical of most SAM8 microcontrollers. There are 78 instructions.

The powerful data manipulation capabilities and features of the instruction set include:

- A full complement of 8-bit arithmetic and logic operations, including multiply and divide
- No special I/O instructions (I/O control/data registers are mapped directly into the register file)
- Decimal adjustment included in binary-coded decimal (BCD) operations
- 16-bit (word) data can be incremented and decremented
- Flexible instructions for bit addressing, rotate, and shift operations

6.1.1 Data Types

The SAM8 CPU performs operations on bits, bytes, BCD digits, and two-byte words. Bits in the register file can be set, cleared, complemented, and tested. Bits within a byte are numbered from 7 to 0, where bit 0 is the least significant (right-most) bit.

6.1.2 Register Addressing

To access an individual register, an 8-bit address in the range 0-255 or the 4-bit address of a working register is specified. Paired registers can be used to construct 16-bit data or 16-bit program memory or data memory addresses. For detailed information about register addressing, please refer to Section 2, "Address Spaces."

6.1.3 Addressing Modes

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

Table 6.1 Instruction Group Summary

Mnemonic	Operands	Instruction
Load Instructions		
CLR	dst	Clear
LD	dst,src	Load
LDB	dst,src	Load bit
LDE	dst,src	Load external data memory
LDC	dst,src	Load program memory
LDED	dst,src	Load external data memory and decrement
LDCD	dst,src	Load program memory and decrement
LDEI	dst,src	Load external data memory and increment
LDCI	dst,src	Load program memory and increment
LDEPD	dst,src	Load external data memory with pre-decrement
LDCPD	dst,src	Load program memory with pre-decrement
LDEPI	dst,src	Load external data memory with pre-increment
LDCPI	dst,src	Load program memory with pre-increment
LDW	dst,src	Load word
POP	dst	Pop from stack
POPUD	dst,src	Pop user stack (decrementing)
POPUI	dst,src	Pop user stack (incrementing)
PUSH	src	Push to stack
PUSHUD	dst,src	Push user stack (decrementing)
PUSHUI	dst,src	Push user stack (incrementing)
Arithmetic Instructions		
ADC	dst,src	Add with carry
ADD	dst,src	Add
CP	dst,src	Compare
DA	dst	Decimal adjust
DEC	dst	Decrement
DECW	dst	Decrement word
DIV	dst,src	Divide
INC	dst	Increment
INCW	dst	Increment word
MULT	dst,src	Multiply
SBC	dst,src	Subtract with carry
SUB	dst,src	Subtract
Logic Instructions		
AND	dst,src	Logical AND
COM	dst	Complement

Mnemonic	Operands	Instruction
OR	dst,src	Logical OR
XOR	dst,src	Logical exclusive OR
Program Control Instructions		
BTJRF	dst,src	Bit test and jump relative on false
BTJRT	dst,src	Bit test and jump relative on true
CALL	dst	Call procedure
CPIJE	dst,src	Compare, increment and jump on equal
CPIJNE	dst,src	Compare, increment and jump on non-equal
DJNZ	r,dst	Decrement register and jump on non-zero
ENTER		Enter
EXIT		Exit
IRET		Interrupt return
JP	cc,dst	Jump on condition code
JP	dst	Jump unconditional
JR	cc,dst	Jump relative on condition code
NEXT		Next
RET		Return
WFI		Wait for interrupt
Bit Manipulation Instructions		
BAND	dst,src	Bit AND
BCP	dst,src	Bit compare
BITC	dst	Bit complement
BITR	dst	Bit reset
BITS	dst	Bit set
BOR	dst,src	Bit OR
BXOR	dst,src	Bit XOR
TCM	dst,src	Test complement under mask
TM	dst,src	Test under mask
Rotate and Shift Instructions		
RL	dst	Rotate left
RLC	dst	Rotate left through carry
RR	dst	Rotate right
RRC	dst	Rotate right through carry
SRA	dst	Shift right arithmetic
SWAP	dst	Swap nibbles
CPU Control Instructions		
CCF		Complement carry flag
DI		Disable interrupts

Mnemonic	Operands	Instruction
EI		Enable interrupts
IDLE		Enter Idle mode
NOP		No operation
RCF		Reset carry flag
SB0		Set bank 0
SB1		Set bank 1
SCF		Set carry flag
SRP	src	Set register pointers
SRP0	src	Set register pointer 0
SRP1	src	Set register pointer 1
STOP		Enter Stop mode

6.2 Flags Register (FLAGS)

The flags register FLAGS contains eight bits that describe the current status of CPU operations. Four of these bits, FLAGS.7–FLAGS.4, can be tested and used with conditional jump instructions; two others FLAGS.3 and FLAGS.2 are used for BCD arithmetic.

The FLAGS register also contains a bit to indicate the status of fast interrupt processing (FLAGS.1) and a bank address status bit (FLAGS.0) to indicate whether bank 0 or bank 1 is currently being addressed. FLAGS register can be set or reset by instructions as long as its outcome does not affect the flags, such as, Load instruction.

Logical and Arithmetic instructions such as, AND, OR, XOR, ADD, and SUB can affect the Flags register. For example, the AND instruction updates the Zero, Sign and Overflow flags based on the outcome of the AND instruction. If the AND instruction uses the Flags register as the destination, then simultaneously, two write will occur to the Flags register producing an unpredictable result.

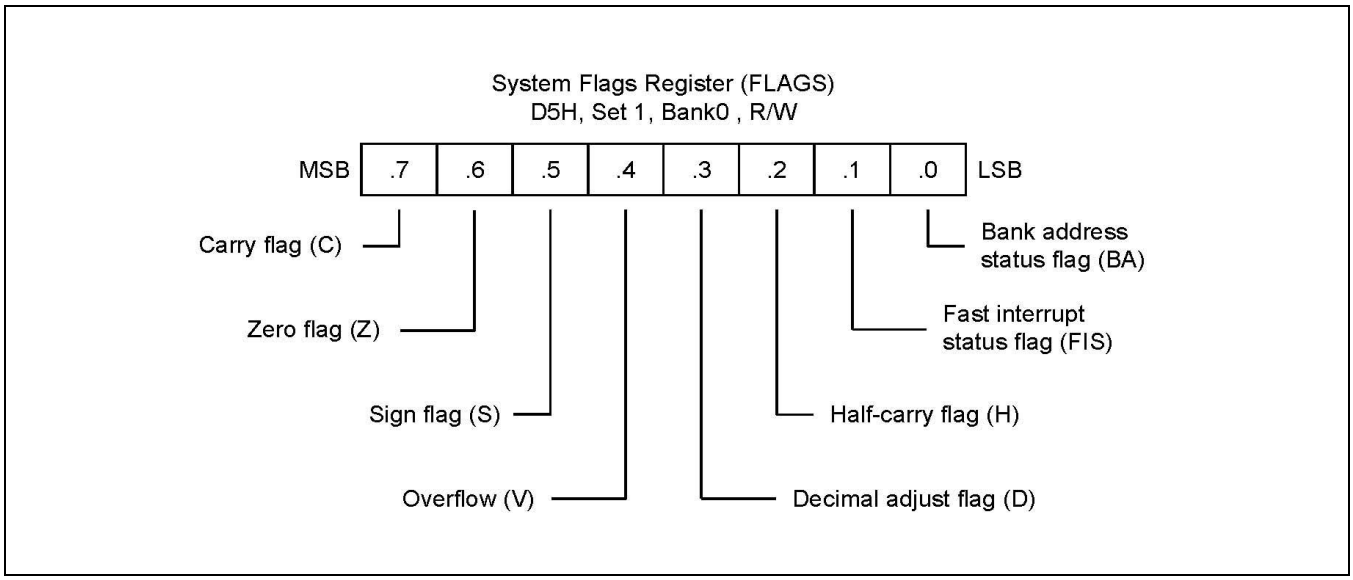


Figure 6-1 System Flags Register (FLAGS)

6.2.1 Flag Descriptions

C Carry Flag (FLAGS.7)

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

Z Zero Flag (FLAGS.6)

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

S Sign Flag (FLAGS.5)

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

V Overflow Flag (FLAGS.4)

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

D Decimal Adjust Flag (FLAGS.3)

The DA bit is used to specify what type of instruction was executed last during BCD operations, so that a subsequent decimal adjust operation can execute correctly. The DA bit is not usually accessed by programmers, and cannot be used as a test condition.

H Half-Carry Flag (FLAGS.2)

The H bit is set to "1" whenever an addition generates a carry-out of bit 3, or when a subtraction borrows out of bit 4. It is used by the Decimal Adjust (DA) instruction to convert the binary result of a previous addition or subtraction into the correct decimal (BCD) result. The H flag is seldom accessed directly by a program.

FIS Fast Interrupt Status Flag (FLAGS.1)

The FIS bit is set during a fast interrupt cycle and reset during the IRET following interrupt servicing. When set, it inhibits all interrupts and causes the fast interrupt return to be executed when the IRET instruction is executed.

BA Bank Address Flag (FLAGS.0)

The BA flag indicates which register bank in the set 1 area of the internal register file is currently selected, bank 0 or bank 1. The BA flag is cleared to "0" (select bank 0) when you execute the SB0 instruction and is set to "1" (select bank 1) when you execute the SB1 instruction.

6.2.2 Instruction Set Notation

Table 6.2 Flag Notation Conventions

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

Table 6.3 Instruction Set Symbols

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

Table 6.4 Instruction Notation Conventions

Notation	Description	Actual Operand Range
cc	Condition code	See list of condition codes in Table 6-6.
r	Working register only	Rn (n = 0 to 15)
rb	Bit (b) of working register	Rn.b (n = 0 to 15, b = 0 to 7)
r0	Bit 0 (LSB) of working register	Rn (n = 0 to 15)
rr	Working register pair	RRp (p = 0, 2, 4, ..., 14)
R	Register or working register	reg or Rn (reg = 0–255, n = 0 to 15)
Rb	Bit "b" of register or working register	reg.b (reg = 0 to 255, b = 0 to 7)
RR	Register pair or working register pair	reg or RRp (reg = 0 to 254, even number only, where p = 0, 2, ..., 14)
IA	Indirect addressing mode	addr (addr = 0 to 254, even number only)
Ir	Indirect working register only	@Rn (n = 0 to 15)
IR	Indirect register or indirect working register	@Rn or @reg (reg = 0 to 255, n = 0 to 15)
Irr	Indirect working register pair only	@RRp (p = 0, 2, ..., 14)
IRR	Indirect register pair or indirect working register pair	@RRp or @reg (reg = 0–254, even only, where p = 0, 2, ..., 14)
X	Indexed addressing mode	#reg [Rn] (reg = 0 to 255, n = 0 to 15)
XS	Indexed (short offset) addressing mode	#addr [RRp] (addr = range – 128 to + 127, where p = 0, 2, ..., 14)
xl	Indexed (long offset) addressing mode	#addr [RRp] (addr = range 0v65535, where p = 0, 2, ..., 14)
da	Direct addressing mode	addr (addr = range 0–65535)
ra	Relative addressing mode	addr (addr = number in the range + 127 to – 128 that is an offset relative to the address of the next instruction)
im	Immediate addressing mode	#data (data = 0 to 255)
iml	Immediate (long) addressing mode	#data (data = range 0 to 65535)

Table 6.5 Opcode Quick Reference

OPCODE MAP									
LOWER NIBBLE (HEX)									
	–	0	1	2	3	4	5	6	7
U	0	DEC R1	DEC IR1	ADD r1,r2	ADD r1,Ir2	ADD R2,R1	ADD IR2,R1	ADD R1,IM	BOR r0–Rb
P	1	RLC R1	RLC IR1	ADC r1,r2	ADC r1,Ir2	ADC R2,R1	ADC IR2,R1	ADC R1,IM	BCP r1.b, R2
P	2	INC R1	INC IR1	SUB r1,r2	SUB r1,Ir2	SUB R2,R1	SUB IR2,R1	SUB R1,IM	BXOR r0–Rb
E	3	JP IRR1	SRP/0/1 IM	SBC r1,r2	SBC r1,Ir2	SBC R2,R1	SBC IR2,R1	SBC R1,IM	BTJR r2.b, RA
R	4	DA R1	DA IR1	OR r1,r2	OR r1,Ir2	OR R2,R1	OR IR2,R1	OR R1,IM	LDB r0–Rb
	5	POP R1	POP IR1	AND r1,r2	AND r1,Ir2	AND R2,R1	AND IR2,R1	AND R1,IM	BITC r1.b
N	6	COM R1	COM IR1	TCM r1,r2	TCM r1,Ir2	TCM R2,R1	TCM IR2,R1	TCM R1,IM	BAND r0–Rb
I	7	PUSH R2	PUSH IR2	TM r1,r2	TM r1,Ir2	TM R2,R1	TM IR2,R1	TM R1,IM	BIT r1.b
B	8	DECW RR1	DECW IR1	PUSHUD IR1,R2	PUSHUI IR1,R2	MULT R2,RR1	MULT IR2,RR1	MULT IM,RR1	LD r1, x, r2
B	9	RL R1	RL IR1	POPUD IR2,R1	POPUI IR2,R1	DIV R2,RR1	DIV IR2,RR1	DIV IM,RR1	LD r2, x, r1
L	A	INCW RR1	INCW IR1	CP r1,r2	CP r1,Ir2	CP R2,R1	CP IR2,R1	CP R1,IM	LDC r1, Irr2, xL
E	B	CLR R1	CLR IR1	XOR r1,r2	XOR r1,Ir2	XOR R2,R1	XOR IR2,R1	XOR R1,IM	LDC r2, Irr2, xL
	C	RRC R1	RRC IR1	CPIJE Ir,r2,RA	LDC r1,Irr2	LDW RR2,RR1	LDW IR2,RR1	LDW RR1,IML	LD r1, Ir2
H	D	SRA R1	SRA IR1	CPIJNE Irr,r2,RA	LDC r2,Irr1	CALL IA1		LD IR1,IM	LD Ir1, r2
E	E	RR R1	RR IR1	LDCD r1,Irr2	LDCI r1,Irr2	LD R2,R1	LD R2,IR1	LD R1,IM	LDC r1, Irr2, xs
X	F	SWAP R1	SWAP IR1	LDCPD r2,Irr1	LDCPI r2,Irr1	CALL IRR1	LD IR2,R1	CALL DA1	LDC r2, Irr1, xs

OPCODE MAP									
LOWER NIBBLE (HEX)									
	–	8	9	A	B	C	D	E	F
U	0	LD r1,R2	LD r2,R1	DJNZ r1,RA	JR cc,RA	LD r1,IM	JP cc,DA	INC r1	NEXT
P	1	↓	↓	↓	↓	↓	↓	↓	ENTER
P	2								EXIT
E	3								WFI
R	4								SB0
	5								SB1
N	6								IDLE
I	7	↓	↓	↓	↓	↓	↓	↓	STOP
B	8								DI
B	9								EI
L	A								RET
E	B								IRET
	C								RCF
H	D	↓	↓	↓	↓	↓	↓	↓	SCF
E	E								CCF
X	F	LD r1,R2	LD r2,R1	DJNZ r1,RA	JR cc,RA	LD r1,IM	JP cc,DA	INC r1	NOP

6.2.3 Condition Codes

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

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

Table 6.6 Condition Codes

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

NOTE:

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

6.3 Instruction Descriptions

This section contains detailed information and programming examples for each instruction in the SAM8 instruction set. Information is arranged in a consistent format for improved readability and for fast referencing.

The following information is included in each instruction description:

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

6.3.1 ADC - Add with carry

ADC dst,src

Operation: dst ← dst + src + c

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

Flags:

C: Set if there is a carry from the most significant bit of the result; cleared otherwise.

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result is negative; cleared otherwise.

V: Set if arithmetic overflow occurs, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise.

D: Always cleared to "0".

H: Set if there is a carry from the most significant bit of the low-order four bits of the result; cleared otherwise.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	12	r	r
		6	13	r	lr
opc | src | dst	3	6	14	R	R
		6	15	R	IR
opc | dst | src	3	6	16	R	IM

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

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

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

6.3.2 ADD - Add

ADD dst,src

Operation: dst ← dst + src

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

Flags:

C: Set if there is a carry from the most significant bit of the result; cleared otherwise.

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result is negative; cleared otherwise.

V: Set if arithmetic overflow occurred, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise.

D: Always cleared to "0".

H: Set if a carry from the low-order nibble occurred.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | src		2	4	02	r	r
				6	03	r	lr
opc	src	dst	3	6	04	R	R
				6	05	R	IR
opc	dst	src	3	6	06	R	IM

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

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

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

6.3.3 AND - Logical AND

AND dst,src

Operation: dst ← dst AND src

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

Flags:

C: Unaffected.

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result bit 7 is set; cleared otherwise.

V: Always cleared to "0".

D: Unaffected.

H: Unaffected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | src		2	4	52	r	r
				6	53	r	lr
opc	src	dst	3	6	54	R	R
				6	55	R	IR
opc	dst	src	3	6	56	R	IM

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

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

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

6.3.4 BAND - Bit AND

BAND dst,src.b

BAND dst.b,src

Operation: dst(0) ← dst(0) AND src(b)
or

dst(b) ← dst(b) AND src(0)

The specified bit of the source (or the destination) is logically ANDed with the zero bit (LSB) of the destination (or source). The resultant bit is stored in the specified bit of the destination. No other bits of the destination are affected. The source is unaffected.

Flags:

C: Unaffected.

Z: Set if the result is "0"; cleared otherwise.

S: Cleared to "0".

V: Undefined.

D: Unaffected.

H: Unaffected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | b | 0	src	3	6	67	r0	Rb
opc	src | b | 1	dst	3	6	67	Rb	r0

NOTE: In the second byte of the 3 byte instruction formats, the destination (or source) address is four bits, the bit address "b" is three bits, and the LSB address value is one bit in length.

Examples: Given: R1 = 07H and register 01H = 05H:

```
BAND    R1,01H.1        →       R1 = 06H, register 01H = 05H
BAND    01H.1,R1        →       Register 01H = 05H, R1 = 07H
```

In the first example, source register 01H contains the value 05H (00000101B) and destination working register R1 contains 07H (00000111B). The statement "BAND R1, 01H.1" ANDs the bit value of the source register ("0") with the bit 0 value of register R1 (destination), leaving the value 06H (00000110B) in register R1.

6.3.5 BCP - Bit Compare

BCP dst,src.b

Operation: dst(0) – src(b)

The specified bit of the source is compared to (subtracted from) bit zero (LSB) of the destination. The zero flag is set if the bits are the same; otherwise it is cleared. The contents of both operands are unaffected by the comparison.

Flags:

C: Unaffected.

Z: Set if the two bits are the same; cleared otherwise.

S: Cleared to "0".

V: Undefined.

D: Unaffected.

H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | b | 0 | src	3	6	17	r0	Rb

NOTE: In the second byte of the instruction format, the destination address is four bits, the bit address "b" is three bits, and the LSB address value is one bit in length.

Example: Given: R1 = 07H and register 01H = 01H:

```
BCP     R1,01H.1        →       R1  =  07H, register 01H  =  01H
```

If destination working register R1 contains the value 07H (00000111B) and the source register 01H contains the value 01H (00000001B), the statement "BCP R1, 01H.1" compares bit one of the source register (01H) and bit zero of the destination register (R1). Because the bit values are not identical, the zero flag bit (Z) is cleared in the FLAGS register (0D5H).

6.3.6 BITC - Bit Complement

BITC dst.b

Operation: dst(b) ← NOT dst(b)

This instruction complements the specified bit within the destination without affecting any other bits in the destination.

Flags:

C: Unaffected.

Z: Set if the result is "0"; cleared otherwise.

S: Cleared to "0".

V: Undefined.

D: Unaffected.

H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst | b | 0	2	4	57	rb

NOTE: In the second byte of the instruction format, the destination address is four bits, the bit address "b" is three bits, and the LSB address value is one bit in length.

Example: Given: R1 = 07H

```
BITC    R1.1    →       R1 = 05H
```

If working register R1 contains the value 07H (00000111B), the statement "BITC R1.1" complements bit one of the destination and leaves the value 05H (00000101B) in register R1. Because the result of the complement is not "0", the zero flag (Z) in the FLAGS register (0D5H) is cleared.

6.3.7 BITR - Bit Reset

BITR dst.b

Operation: dst(b) ← 0

The BITR instruction clears the specified bit within the destination without affecting any other bits in the destination.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst | b | 0	2	4	77	rb

NOTE: In the second byte of the instruction format, the destination address is four bits, the bit address "b" is three bits, and the LSB address value is one bit in length.

Example: Given: R1 = 07H:

```
BITR    R1.1    →       R1 = 05H
```

If the value of working register R1 is 07H (00000111B), the statement "BITR R1.1" clears bit one of the destination register R1, leaving the value 05H (00000101B).

6.3.8 BITS - Bit Set

BITS dst.b

Operation: dst(b) ← 1

The BITS instruction sets the specified bit within the destination without affecting any other bits in the destination.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst | b | 1	2	4	77	rb

NOTE: In the second byte of the instruction format, the destination address is four bits, the bit address "b" is three bits, and the LSB address value is one bit in length.

Example: Given: R1 = 07H:

```
BITS    R1.3    →       R1 = 0FH
```

If working register R1 contains the value 07H (00000111B), the statement "BITS R1.3" sets bit three of the destination register R1 to "1", leaving the value 0FH (00001111B).

6.3.9 BOR - Bit OR

BOR dst,src.b

BOR dst.b,src

Operation: dst(0) ← dst(0) OR src(b)
or
dst(b) ← dst(b) OR src(0)

The specified bit of the source (or the destination) is logically ORed with bit zero (LSB) of the destination (or the source). The resulting bit value is stored in the specified bit of the destination. No other bits of the destination are affected. The source is unaffected.

Flags:

C: Unaffected.

Z: Set if the result is "0"; cleared otherwise.

S: Cleared to "0".

V: Undefined.

D: Unaffected.

H: Unaffected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | b | 0	src	3	6	07	r0	Rb
opc	src | b | 1	dst	3	6	07	Rb	r0

NOTE: In the second byte of the 3 byte instruction formats, the destination (or source) address is four bits, the bit address "b" is three bits, and the LSB address value is one bit.

Examples: Given: R1 = 07H and register 01H = 03H:

```
BOR     R1, 01H.1       →       R1 = 07H, register 01H = 03H
BOR     01H.2, R1       →       Register 01H = 07H, R1 = 07H
```

In the first example, destination working register R1 contains the value 07H (00000111B) and source register 01H the value 03H (00000011B). The statement "BOR R1, 01H.1" logically ORs bit one of register 01H (source) with bit zero of R1 (destination). This leaves the same value (07H) in working register R1.

In the second example, destination register 01H contains the value 03H (00000011B) and the source working register R1 the value 07H (00000111B). The statement "BOR 01H.2, R1" logically ORs bit two of register 01H (destination) with bit zero of R1 (source). This leaves the value 07H in register 01H.

6.3.10 BTJRF - Bit Test, Jump Relative on False

BTJRF dst,src.b

Operation: If src(b) is a "0", then PC ← PC + dst

The specified bit within the source operand is tested. If it is a "0", the relative address is added to the program counter and control passes to the statement whose address is now in the PC; otherwise, the instruction following the BTJRF instruction is executed.

Flags: No flags are affected.

Format:

(Note 1)			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	src | b | 0	dst	3	10	37	RA	rb

NOTE: In the second byte of the instruction format, the source address is four bits, the bit address "b" is three bits, and the LSB address value is one bit in length.

Example: Given: R1 = 07H:

```
BTJRF   SKIP,R1.3            →      PC jumps to SKIP location
```

If working register R1 contains the value 07H (00000111B), the statement "BTJRF SKIP, R1.3" tests bit 3. Because it is "0", the relative address is added to the PC and the PC jumps to the memory location pointed to by the SKIP. (Remember that the memory location must be within the allowed range of + 127 to – 128.)

6.3.11 BTJRT - Bit Test, Jump Relative on True

BTJRT dst,src.b

Operation: If src(b) is a "1", then PC ← PC + dst

The specified bit within the source operand is tested. If it is a "1", the relative address is added to the program counter and control passes to the statement whose address is now in the PC; otherwise, the instruction following the BTJRT instruction is executed.

Flags: No flags are affected.

Format:

(Note 1)			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	src | b | 1	dst	3	10	37	RA	rb

NOTE: In the second byte of the instruction format, the source address is four bits, the bit address "b" is three bits, and the LSB address value is one bit in length.

Example: Given: R1 = 07H:

```
BTJRT   SKIP, R1.1
```

If working register R1 contains the value 07H (00000111B), the statement "BTJRT SKIP, R1.1" tests bit one in the source register (R1). Because it is a "1", the relative address is added to the PC and the PC jumps to the memory location pointed to by the SKIP. (Remember that the memory location must be within the allowed range of + 127 to – 128.)

6.3.12 BXOR - Bit XOR

BXOR dst,src.b

BXOR dst.b,src

Operation: dst(0) ← dst(0) XOR src(b)
or
dst(b) ← dst(b) XOR src(0)

The specified bit of the source (or the destination) is logically exclusive-ORed with bit zero (LSB) of the destination (or source). The result bit is stored in the specified bit of the destination. No other bits of the destination are affected. The source is unaffected.

Flags:

C: Unaffected.

Z: Set if the result is "0"; cleared otherwise.

S: Cleared to "0".

V: Undefined.

D: Unaffected.

H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | b | 0 | src	3	6	27	r0	Rb
opc | src | b | 1 | dst	3	6	27	Rb	r0

NOTE: In the second byte of the 3 byte instruction formats, the destination (or source) address is four bits, the bit address "b" is three bits, and the LSB address value is one bit in length.

Examples: Given: R1 = 07H (00000111B) and register 01H = 03H (00000011B):

```
BXOR    R1,01H.1        →       R1 = 06H, register 01H = 03H
BXOR    01H.2,R1        →       Register 01H = 07H, R1 = 07H
```

In the first example, destination working register R1 has the value 07H (00000111B) and source register 01H has the value 03H (00000011B). The statement "BXOR R1, 01H.1" exclusive-ORs bit one of register 01H (source) with bit zero of R1 (destination). The result bit value is stored in bit zero of R1, changing its value from 07H to 06H. The value of source register 01H is unaffected.

6.3.13 CALL - Call Procedure

CALL dst

Operation:

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

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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	3	14	F6	DA
opc | dst	2	12	F4	IRR
opc | dst	2	14	D4	IA

Examples: Given: R0 = 35H, R1 = 21H, PC = 1A47H, and SP = 0002H:

```
CALL    3521H   →       SP = 0000H
                        (Memory locations 0000H = 1AH, 0001H = 4AH, where
                        4AH is the address that follows the instruction.)
CALL    @RR0    →       SP = 0000H (0000H = 1AH, 0001H = 49H)
CALL    #40H    →       SP = 0000H (0000H = 1AH, 0001H = 49H)
```

In the first example, if the program counter value is 1A47H and the stack pointer contains the value 0002H, the statement "CALL 3521H" pushes the current PC value onto the top of the stack. The stack pointer now points to memory location 0000H. The PC is then loaded with the value 3521H, the address of the first instruction in the program sequence to be executed.

If the contents of the program counter and stack pointer are the same as in the first example, the statement "CALL @RR0" produces the same result except that the 49H is stored in stack location 0001H (because the two-byte instruction format was used). The PC is then loaded with the value 3521H, the address of the first instruction in the program sequence to be executed. Assuming that the contents of the program counter and stack pointer are the same as in the first example, if program address 0040H contains 35H and program address 0041H contains 21H, the statement "CALL #40H" produces the same result as in the second example.

6.3.14 CCF - Complement Carry Flag

CCF

Operation: C ← NOT C

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

Flags: **C:** Complemented.

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	EF

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

```
CCF
```

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

6.3.15 CLR - Clear

CLR dst

Operation: dst ← "0"

The destination location is cleared to "0".

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	B0	R
		4	B1	IR

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

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

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

6.3.16 COM - Complement

COM dst

Operation: dst ← NOT dst

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

Flags:

C: Unaffected.

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result bit 7 is set; cleared otherwise.

V: Always reset to "0".

D: Unaffected.

H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	60	R
		4	61	IR

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

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

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

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

6.3.17 CP - Compare

CP dst,src

Operation: dst – src

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

Flags:

C: Set if a "borrow" occurred (src > dst); cleared otherwise.

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result is negative; cleared otherwise.

V: Set if arithmetic overflow occurred; cleared otherwise.

D: Unaffected.

H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	A2	r	r
		6	A3	r	lr
opc | src | dst	3	6	A4	R	R
		6	A5	R	IR
opc | dst | src	3	6	A6	R	IM

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

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

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

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

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

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

6.3.18 CPIJE - Compare, Increment, and Jump on Equal

CPIJE dst,src,RA

Operation: If dst – src = 0", PC ← PC + RA
Ir ← Ir + 1

The source operand is compared to (subtracted from) the destination operand. If the result is "0", the relative address is added to the program counter and control passes to the statement whose address is now in the program counter. Otherwise, the instruction immediately following the CPIJE instruction is executed. In either case, the source pointer is incremented by one before the next instruction is executed.

Flags: No flags are affected.

Format:

				Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	src	dst	RA	3	12	C2	r	Ir

NOTE: Execution time is 18 cycles if the jump is taken or 16 cycles if it is not taken.

Example: Given: R1 = 02H, R2 = 03H, and register 03H = 02H:

```
CPIJE   R1,@R2,SKIP     →      R2 = 04H, PC jumps to SKIP location
```

In this example, working register R1 contains the value 02H, working register R2 the value 03H, and register 03 contains 02H. The statement "CPIJE R1,@R2,SKIP" compares the @R2 value 02H (00000010B) to 02H (00000010B). Because the result of the comparison is equal, the relative address is added to the PC and the PC then jumps to the memory location pointed to by SKIP. The source register (R2) is incremented by one, leaving a value of 04H. (Remember that the memory location must be within the allowed range of + 127 to – 128.)

6.3.19 CPIJNE - Compare, Increment, and Jump on Non-Equal

CPIJNE dst,src,RA

Operation: If dst – src "0", PC ← PC + RA
Ir ← Ir + 1

The source operand is compared to (subtracted from) the destination operand. If the result is not "0", the relative address is added to the program counter and control passes to the statement whose address is now in the program counter; otherwise the instruction following the CPIJNE instruction is executed. In either case the source pointer is incremented by one before the next instruction.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | src | dst | RA	3	12	D2	r	Ir

NOTE: Execution time is 18 cycles if the jump is taken or 16 cycles if it is not taken.

Example: Given: R1 = 02H, R2 = 03H, and register 03H = 04H:

```
CPIJNE  R1,@R2,SKIP     →       R2 = 04H, PC jumps to SKIP location
```

Working register R1 contains the value 02H, working register R2 (the source pointer) the value 03H, and general register 03 the value 04H. The statement "CPIJNE R1,@R2,SKIP" subtracts 04H (00000100B) from 02H (00000010B). Because the result of the comparison is non-equal, the relative address is added to the PC and the PC then jumps to the memory location pointed to by SKIP. The source pointer register (R2) is also incremented by one, leaving a value of 04H. (Remember that the memory location must be within the allowed range of + 127 to – 128.)

6.3.20 DA - Decimal Adjust

DA dst

Operation: dst ← DA dst

The destination operand is adjusted to form two 4-bit BCD digits following an addition or subtraction operation. For addition (ADD, ADC) or subtraction (SUB, SBC), the following table indicates the operation performed. (The operation is undefined if the destination operand was not the result of a valid addition or subtraction of BCD digits):

Instruction	Carry Before DA	Bits 4–7 Value (Hex)	H Flag Before DA	Bits 0–3 Value (Hex)	Number Added to Byte	Carry After DA
	0	0–9	0	0–9	00	0
	0	0–8	0	A–F	06	0
	0	0–9	1	0–3	06	0
ADD	0	A–F	0	0–9	60	1
ADC	0	9–F	0	A–F	66	1
	0	A–F	1	0–3	66	1
	1	0–2	0	0–9	60	1
	1	0–2	0	A–F	66	1
	1	0–3	1	0–3	66	1
	0	0–9	0	0–9	00 = – 00	0
SUB	0	0–8	1	6–F	FA = – 06	0
SBC	1	7–F	0	0–9	A0 = – 60	1
	1	6–F	1	6–F	9A = – 66	1

Flags:

C: Set if there was a carry from the most significant bit; cleared otherwise (see table).

Z: Set if result is "0"; cleared otherwise.

S: Set if result bit 7 is set; cleared otherwise.

V: Undefined.

D: Unaffected.

H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	40	R
		4	41	IR

Example: Given: Working register R0 contains the value 15 (BCD), working register R1 contains 27 (BCD), and address 27H contains 46 (BCD):

```
ADD     R1,R0   ;       C ← "0", H ← "0", Bits 4–7 = 3, bits 0–3 = C, R1 ← 3CH
DA      R1      ;       R1 ← 3CH + 06
```

If addition is performed using the BCD values 15 and 27, the result should be 42. The sum is incorrect, however, when the binary representations are added in the destination location using standard binary arithmetic:

```
    0 0 0 1     0 1 0 1  15
+   0 0 1 0     0 1 1 1  27
    0 0 1 1     1 1 0 0  = 3CH
```

The DA instruction adjusts this result so that the correct BCD representation is obtained:

```
    0 0 1 1     1 1 0 0
+   0 0 0 0     0 1 1 0
    0 1 0 0     0 0 1 0  = 42
```

Assuming the same values given above, the statements

```
SUB     27H,R0  ;       C ←"0", H ← "0", Bits 4–7 = 3, bits 0–3 = 1
DA      @R1     ;       @R1 ← 31–0
```

leave the value 31 (BCD) in address 27H (@R1).

6.3.21 DEC - Decrement

DEC dst

Operation: dst ← dst – 1

The contents of the destination operand are decremented by one.

Flags:

C: Unaffected.

Z: Set if the result is "0"; cleared otherwise.

S: Set if result is negative; cleared otherwise.

V: Set if arithmetic overflow occurred; cleared otherwise.

D: Unaffected.

H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	00	R
		4	01	IR

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

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

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

6.3.22 DECW - Decrement Word

DECW dst

Operation: dst ← dst – 1

The contents of the destination location (which must be an even address) and the operand following that location are treated as a single 16-bit value that is decremented by one.

Flags:

C: Unaffected.

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result is negative; cleared otherwise.

V: Set if arithmetic overflow occurred; cleared otherwise.

D: Unaffected.

H: Unaffected.

Format:

opc	dst

Bytes	Cycles	Opcode (Hex)	Addr Mode dst
2	8	80	RR
	8	81	IR

Examples: Given: R0 = 12H, R1 = 34H, R2 = 30H, register 30H = 0FH, and register 31H = 21H:

```
DECW    RR0     →       R0 = 12H, R1 = 33H
DECW    @R2     →→      Register 30H = 0FH, register 31H = 20H
```

In the first example, destination register R0 contains the value 12H and register R1 the value 34H. The statement "DECW RR0" addresses R0 and the following operand R1 as a 16-bit word and decrements the value of R1 by one, leaving the value 33H.

NOTE: A system malfunction may occur if you use a Zero flag (FLAGS.6) result together with a DECW instruction. To avoid this problem, we recommend that you use DECW as shown in the following example:

```
LOOP:   DECW    RR0
        LD      R2,R1
        OR      R2,R0
        JR      NZ,LOOP
```

6.3.23 DI - Disable Interrupts

DI

Operation: SYM (0) ← 0

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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	8F

Example: Given: SYM = 01H:

```
DI
```

If the value of the SYM register is 01H, the statement "DI" leaves the new value 00H in the register and clears SYM.0 to "0", disabling interrupt processing.

Before changing IMR, interrupt pending and interrupt source control register, be sure DI state.

6.3.24 DIV - Divide (Unsigned)

DIV dst,src

Operation: dst ÷ src
dst (UPPER) ← REMAINDER
dst (LOWER) ← QUOTIENT

The destination operand (16 bits) is divided by the source operand (8 bits). The quotient (8 bits) is stored in the lower half of the destination. The remainder (8 bits) is stored in the upper half of the destination. When the quotient is ≥ 2^8, the numbers stored in the upper and lower halves of the destination for quotient and remainder are incorrect. Both operands are treated as unsigned integers.

Flags:

C: Set if the V flag is set and quotient is between 2^8 and 2^9 –1; cleared otherwise.

Z: Set if divisor or quotient = "0"; cleared otherwise.

S: Set if MSB of quotient = "1"; cleared otherwise.

V: Set if quotient is ≥ 2^8 or if divisor = "0"; cleared otherwise.

D: Unaffected.

H: Unaffected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	src	dst	3	26/10	94	RR	R
				26/10	95	RR	IR
				26/10	96	RR	IM

NOTE: Execution takes 10 cycles if the divide-by-zero is attempted; otherwise it takes 26 cycles.

Examples: Given: R0 = 10H, R1 = 03H, R2 = 40H, register 40H = 80H:

```
DIV     RR0,R2          →       R0 = 03H, R1 = 40H
DIV     RR0,@R2         →       R0 = 03H, R1 = 20H
DIV     RR0,#20H        →       R0 = 03H, R1 = 80H
```

In the first example, destination working register pair RR0 contains the values 10H (R0) and 03H (R1), and register R2 contains the value 40H. The statement "DIV RR0, R2" divides the 16-bit RR0 value by the 8-bit value of the R2 (source) register. After the DIV instruction, R0 contains the value 03H and R1 contains 40H. The 8-bit remainder is stored in the upper half of the destination register RR0 (R0) and the quotient in the lower half (R1).

6.3.25 DJNZ - Decrement and Jump if Non-Zero

DJNZ r,dst

Operation: r ← r – 1

If r ≠ 0, PC ← PC + dst

The working register being used as a counter is decremented. If the contents of the register are not logic zero after decrementing, the relative address is added to the program counter and control passes to the statement whose address is now in the PC. The range of the relative address is + 127 to – 128, and the original value of the PC is taken to be the address of the instruction byte following the DJNZ statement.

NOTE: In case of using DJNZ instruction, the working register being used as a counter should be set at the one of location 0C0H to 0CFH with SRP, SRP0, or SRP1 instruction.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
r | opc | dst	2	8 (jump taken) 8 (no jump)	rA r = 0 to F	RA

Example: Given: R1 = 02H and LOOP is the label of a relative address:

```
SRP     #0C0H
DJNZ    R1,LOOP
```

DJNZ is typically used to control a "loop" of instructions. In many cases, a label is used as the destination operand instead of a numeric relative address value. In the example, working register R1 contains the value 02H, and LOOP is the label for a relative address.

The statement "DJNZ R1, LOOP" decrements register R1 by one, leaving the value 01H. Because the contents of R1 after the decrement are non-zero, the jump is taken to the relative address specified by the LOOP label.

6.3.26 EI - Enable Interrupts

EI

Operation: SYM (0) ← 1

An EI instruction sets bit zero of the system mode register, SYM.0 to "1". This allows interrupts to be serviced as they occur (assuming they have highest priority). If an interrupt's pending bit was set while interrupt processing was disabled (by executing a DI instruction), it will be serviced when you execute the EI instruction.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	9F

Example: Given: SYM = 00H:

```
EI
```

If the SYM register contains the value 00H, that is, if interrupts are currently disabled, the statement "EI" sets the SYM register to 01H, enabling all interrupts. (SYM.0 is the enable bit for global interrupt processing.)

6.3.27 ENTER - Enter

ENTER

Operation:

SP ← SP – 2
@SP ← IP
IP ← PC
PC ← @IP
IP ← IP + 2

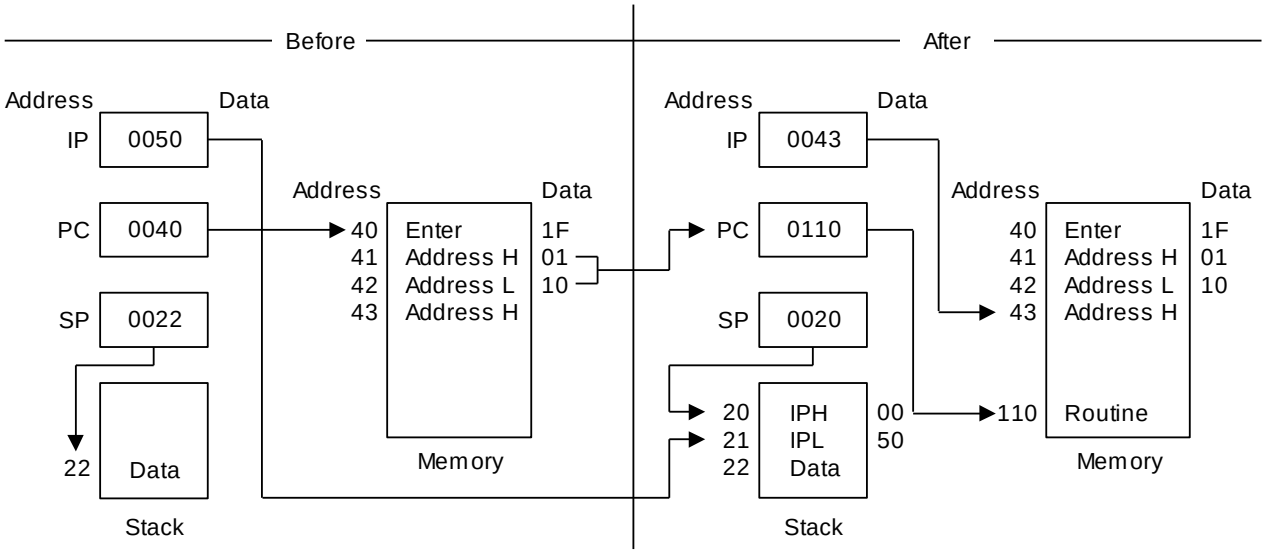
This instruction is useful when implementing threaded-code languages. The contents of the instruction pointer are pushed to the stack. The program counter (PC) value is then written to the instruction pointer. The program memory word that is pointed to by the instruction pointer is loaded into the PC, and the instruction pointer is incremented by two.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	14	1F

Example: The diagram below shows one example of how to use an ENTER statement.



6.3.28 EXIT - Exit

EXIT

Operation:

IP ← @SP
SP ← SP + 2
PC ← @IP
IP ← IP + 2

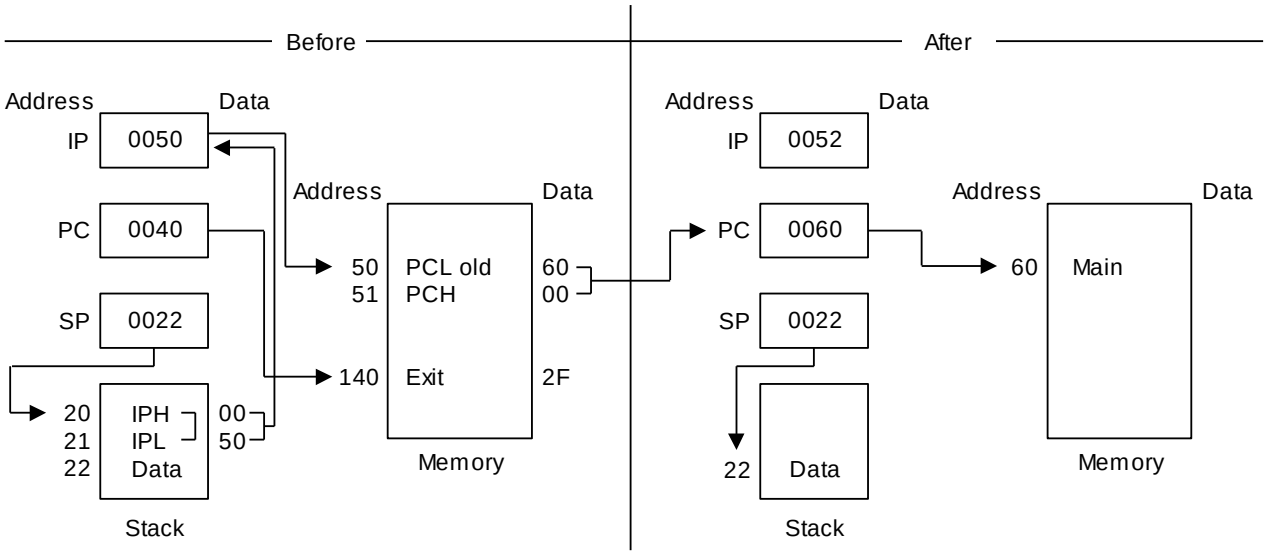
This instruction is useful when implementing threaded-code languages. The stack value is popped and loaded into the instruction pointer. The program memory word that is pointed to by the instruction pointer is then loaded into the program counter, and the instruction pointer is incremented by two.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	14 (internal stack) 16 (internal stack)	2F

Example: The diagram below shows one example of how to use an EXIT statement.



6.3.29 IDLE - Idle Operation

IDLE

Operation:

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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	1	4	6F	–	–

Example: The instruction

```
IDLE
```

stops the CPU clock but not the system clock.

6.3.30 INC - Increment

INC dst

Operation: dst ← dst + 1

The contents of the destination operand are incremented by one.

Flags:

C: Unaffected.

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result is negative; cleared otherwise.

V: Set if arithmetic overflow occurred; cleared otherwise.

D: Unaffected.

H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
dst | opc	1	4	rE r = 0 to F	r
opc | dst	2	4	20	R
		4	21	IR

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

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

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

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

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

6.3.31 INCW - Increment Word

INCW dst

Operation: dst ← dst + 1

The contents of the destination (which must be an even address) and the byte following that location are treated as a single 16-bit value that is incremented by one.

Flags:

C: Unaffected.

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result is negative; cleared otherwise.

V: Set if arithmetic overflow occurred; cleared otherwise.

D: Unaffected.

H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	8	A0	RR
		8	A1	IR

Examples: Given: R0 = 1AH, R1 = 02H, register 02H = 0FH, and register 03H = 0FFH:

```
INCW    RR0     →       R0 = 1AH, R1 = 03H
INCW    @R1     →       Register 02H = 10H, register 03H = 00H
```

In the first example, the working register pair RR0 contains the value 1AH in register R0 and 02H in register R1. The statement "INCW RR0" increments the 16-bit destination by one, leaving the value 03H in register R1. In the second example, the statement "INCW @R1" uses Indirect Register (IR) addressing mode to increment the contents of general register 03H from 0FFH to 00H and register 02H from 0FH to 10H.

NOTE: A system malfunction may occur if you use a Zero (Z) flag (FLAGS.6) result together with an INCW instruction. To avoid this problem, we recommend that you use INCW as shown in the following example:

```
LOOP:   INCW    RR0
        LD      R2,R1
        OR      R2,R0
        JR      NZ,LOOP
```

6.3.32 IRET - Interrupt Return

IRET IRET (Normal) IRET (Fast)

Operation:

FLAGS ← @SP
SP ← SP + 1
PC ← @SP
SP ← SP + 2
SYM(0) ← 1

PC ↔ IP
FLAGS ← FLAGS'
FIS ← 0

This instruction is used at the end of an interrupt service routine. It restores the flag register and the program counter. It also re-enables global interrupts. A "normal IRET" is executed only if the fast interrupt status bit (FIS, bit one of the FLAGS register, 0D5H) is cleared (= "0"). If a fast interrupt occurred, IRET clears the FIS bit that was set at the beginning of the service routine.

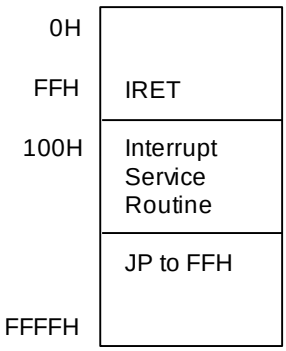
Flags: All flags are restored to their original settings (that is, the settings before the interrupt occurred).

Format:

IRET (Normal)	Bytes	Cycles	Opcode (Hex)
opc	1	10 (internal stack) 12 (internal stack)	BF

IRET (Fast)	Bytes	Cycles	Opcode (Hex)
opc	1	6	BF

Example: In the figure below, the instruction pointer is initially loaded with 100H in the main program before interrupts are enabled. When an interrupt occurs, the program counter and instruction pointer are swapped. This causes the PC to jump to address 100H and the IP to keep the return address. The last instruction in the service routine normally is a jump to IRET at address FFH. This causes the instruction pointer to be loaded with 100H "again" and the program counter to jump back to the main program. Now, the next interrupt can occur and the IP is still correct at 100H.



NOTE: In the fast interrupt example above, if the last instruction is not a jump to IRET, you must pay attention to the order of the last two instructions. The IRET cannot be immediately proceeded by a clearing of the interrupt status (as with a reset of the IPR register).

6.3.33 JP - Jump

JP cc,dst (Conditional)

JP dst (Unconditional)

Operation: If cc is true, PC ← dst

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

Flags: No flags are affected.

Format: (1)

(2)	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
cc | opc | dst	3	8	ccD cc = 0 to F	DA
opc | dst	2	8	30	IRR

NOTE:

1. The 3 byte format is used for a conditional jump and the 2 byte format for an unconditional jump.
2. In the first byte of the three-byte instruction format (conditional jump), the condition code and the opcode are both four bits.

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

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

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

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

6.3.34 JR - Jump Relative

JR cc,dst

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

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

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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
(1) cc | opc dst	2	6	ccB	RA
			cc = 0 to F	

NOTE: In the first byte of the two-byte instruction format, the condition code and the opcode are each four bits.

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

```
JR      C,LABEL_X       →       PC  =  1FF7H
```

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

6.3.35 LD - Load

LD dst,src

Operation: dst ← src

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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
dst | opc / src	2	4	rC	r	IM
		4	r8	r	R
src | opc / dst	2	4	r9 (r = 0 to F)	R	r
opc / dst | src	2	4	C7	r	lr
		4	D7	lr	r
opc / src / dst	3	6	E4	R	R
		6	E5	R	IR
opc / dst / src	3	6	E6	R	IM
		6	D6	IR	IM
opc / src / dst	3	6	F5	IR	R
opc / dst | src / x	3	6	87	r	x [r]
opc / src | dst / x	3	6	97	x [r]	r

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

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

6.3.36 LDB - Load Bit

LDB dst,src.b

LDB dst.b,src

Operation: dst(0) ← src(b)
or
dst(b) ← src(0)

The specified bit of the source is loaded into bit zero (LSB) of the destination, or bit zero of the source is loaded into the specified bit of the destination. No other bits of the destination are affected. The source is unaffected.

Flags: No flags are affected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | b | 0	src	3	6	47	r0	Rb
opc	src | b | 1	dst	3	6	47	Rb	r0

NOTE: In the second byte of the instruction formats, the destination (or source) address is four bits, the bit address "b" is three bits, and the LSB address value is one bit in length.

Examples: Given: R0 = 06H and general register 00H = 05H:

```
LDB     R0,00H.2        →       R0 = 07H, register 00H = 05H
LDB     00H.0,R0        →       R0 = 06H, register 00H = 04H
```

In the first example, destination working register R0 contains the value 06H and the source general register 00H the value 05H. The statement "LD R0, 00H.2" loads the bit two value of the 00H register into bit zero of the R0 register, leaving the value 07H in register R0.

In the second example, 00H is the destination register. The statement "LD 00H.0, R0" loads bit zero of register R0 to the specified bit (bit zero) of the destination register, leaving 04H in general register 00H.

6.3.37 LDC/LDE - Load Memory

LDC/LDE dst,src

Operation: dst ← src

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

Flags: No flags are affected.

Format:

					Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
1.	opc	dst | src			2	10	C3	r	Irr
2.	opc	src | dst			2	10	D3	Irr	r
3.	opc	dst | src	XS		3	12	E7	r	XS [rr]
4.	opc	src | dst	XS		3	12	F7	XS [rr]	r
5.	opc	dst | src	XL_L	XL_H	4	14	A7	r	XL [rr]
6.	opc	src | dst	XL_L	XL_H	4	14	B7	XL [rr]	r
7.	opc	dst | 0000	DA_L	DA_H	4	14	A7	r	DA
8.	opc	src | 0000	DA_L	DA_H	4	14	B7	DA	r
9.	opc	dst | 0001	DA_L	DA_H	4	14	A7	r	DA
10.	opc	src | 0001	DA_L	DA_H	4	14	B7	DA	r

NOTE:

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

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

```
LDC     R0,@RR2          ;R0 ← contents of program memory location 0104H
                         ;R0 = 1AH, R2 = 01H, R3 = 04H
LDE     R0,@RR2          ;R0 ← contents of external data memory location 0104H
                         ;R0 = 2AH, R2 = 01H, R3 = 04H
LDC (NOTE) @RR2,R0       ;11H (contents of R0) is loaded into program memory
                         ;location 0104H (RR2),
                         ;working registers R0, R2, R3 → no change
LDE     @RR2,R0          ;11H (contents of R0) is loaded into external data memory
                         ;location 0104H (RR2),
                         ;working registers R0, R2, R3 → no change
LDC     R0,#01H[RR2]     ;R0 ← contents of program memory location 0105H
                         ;(01H + RR2),
                         ;R0 = 6DH, R2 = 01H, R3 = 04H
LDE     R0,#01H[RR2]     ;R0 ← contents of external data memory location 0105H
                         ;(01H + RR2), R0 = 7DH, R2 = 01H, R3 = 04H
LDC (NOTE) #01H[RR2],R0;11H (contents of R0) is loaded into program memory location
                         ;0105H (01H + 0104H)
LDE     #01H[RR2],R0     ;11H (contents of R0) is loaded into external data memory
                         ;location 0105H (01H + 0104H)
LDC     R0,#1000H[RR2]   ;R0 ← contents of program memory location 1104H
                         ;(1000H + 0104H), R0 = 88H, R2 = 01H, R3 = 04H
LDE     R0,#1000H[RR2]   ;R0 ← contents of external data memory location 1104H
                         ;(1000H + 0104H), R0 = 98H, R2 = 01H, R3 = 04H
LDC     R0,1104H         ;R0 ← contents of program memory location 1104H, R0 = 88H
LDE     R0,1104H         ;R0 ← contents of external data memory location 1104H,
                         ;R0 = 98H
LDC (NOTE) 1105H,R0      ;11H (contents of R0) is loaded into program memory location
                         ;1105H, (1105H) ← 11H
LDE     1105H,R0         ;11H (contents of R0) is loaded into external data memory
                         ;location 1105H, (1105H) ← 11H
```

NOTE: These instructions are not supported by masked ROM type devices.

6.3.38 LDCD/LDED - Load Memory and Decrement

LDCD/LDED dst,src

Operation: dst ← src

rr ← rr – 1

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

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

Flags: No flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | src	2	10	E2	r	Irr

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

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

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

6.3.39 LDCI/LDEI - Load Memory and Increment

LDCI/LDEI dst,src

Operation: dst ← src

rr ← rr + 1

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

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

Flags: No flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | src	2	10	E3	r	Irr

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

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

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

6.3.40 LDCPD/LDEPD - Load Memory with Pre-Decrement

LDCPD/ LDEPD dst,src

Operation: rr ← rr – 1

dst ← src

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

LDCPD refers to program memory and LDEPD refers to external data memory. The assembler makes "Irr" an even number for program memory and an odd number for external data memory.

Flags: No flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	src | dst	2	14	F2	Irr	r

Examples: Given: R0 = 77H, R6 = 30H, and R7 = 00H:

```
LDCPD   @RR6,R0         ;       (RR6 ← RR6 – 1)
                        ;       77H (contents of R0) is loaded into program memory location
                        ;       2FFFH (3000H – 1H)
                        ;       R0 = 77H, R6 = 2FH, R7 = 0FFH

LDEPD   @RR6,R0         ;       (RR6 ← RR6 – 1)
                        ;       77H (contents of R0) is loaded into external data memory
                        ;       location 2FFFH (3000H – 1H)
                        ;       R0 = 77H, R6 = 2FH, R7 = 0FFH
```

6.3.41 LDCPI/LDEPI - Load Memory with Pre-Increment

LDCPI/ LDEPI dst,src

Operation: rr ← rr + 1

dst ← src

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

LDCPI refers to program memory and LDEPI refers to external data memory. The assembler makes "Irr" an even number for program memory and an odd number for data memory.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | src | dst	2	14	F3	Irr	r

Examples: Given: R0 = 7FH, R6 = 21H, and R7 = 0FFH:

```
LDCPI   @RR6,R0         ;       (RR6 ← RR6 + 1)
                        ;       7FH (contents of R0) is loaded into program memory
                        ;       location 2200H (21FFH + 1H)
                        ;       R0 = 7FH, R6 = 22H, R7 = 00H

LDEPI   @RR6,R0         ;       (RR6 ← RR6 + 1)
                        ;       7FH (contents of R0) is loaded into external data memory
                        ;       location 2200H (21FFH + 1H)
                        ;       R0 = 7FH, R6 = 22H, R7 = 00H
```

6.3.42 LDW - Load Word

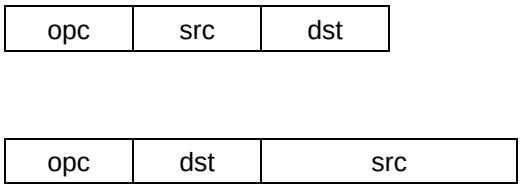
LDW dst,src

Operation: dst ← src

The contents of the source (a word) are loaded into the destination. The contents of the source are unaffected.

Flags: No flags are affected.

Format:



	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc src dst	3	8	C4	RR	RR
		8	C5	RR	IR
opc dst src	4	8	C6	RR	IML

Examples: Given: R4 = 06H, R5 = 1CH, R6 = 05H, R7 = 02H, register 00H = 1AH, register 01H = 02H, register 02H = 03H, and register 03H = 0FH:

```
LDW     RR6,RR4         →       R6 = 06H, R7 = 1CH, R4 = 06H, R5 = 1CH
LDW     00H,02H         →       Register 00H = 03H, register 01H = 0FH,
                                register 02H = 03H, register 03H = 0FH
LDW     RR2,@R7         →       R2 = 03H, R3 = 0FH,
LDW     04H,@01H        →       Register 04H = 03H, register 05H = 0FH
LDW     RR6,#1234H      →       R6 = 12H, R7 = 34H
LDW     02H,#0FEDH      →       Register 02H = 0FH, register 03H = 0EDH
```

In the second example, please note that the statement "LDW 00H, 02H" loads the contents of the source word 02H, 03H into the destination word 00H, 01H. This leaves the value 03H in general register 00H and the value 0FH in register 01H.

The other examples show how to use the LDW instruction with various addressing modes and formats.

6.3.43 MULT - Multiply (Unsigned)

MULT dst,src

Operation: dst ← dst × src

The 8-bit destination operand (even register of the register pair) is multiplied by the source operand (8 bits) and the product (16 bits) is stored in the register pair specified by the destination address. Both operands are treated as unsigned integers.

Flags:

C: Set if result is > 255; cleared otherwise.

Z: Set if the result is "0"; cleared otherwise.

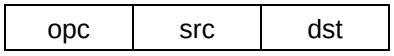
S: Set if MSB of the result is a "1"; cleared otherwise.

V: Cleared.

D: Unaffected.

H: Unaffected.

Format:



Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
3	22	84	RR	R
	22	85	RR	IR
	22	86	RR	IM

Examples: Given: Register 00H = 20H, register 01H = 03H, register 02H = 09H, register 03H = 06H:

```
MULT    00H, 02H      →     Register 00H = 01H, register 01H = 20H, register 02H = 09H
MULT    00H, @01H     →     Register 00H = 00H, register 01H = 0C0H
MULT    00H, #30H     →     Register 00H = 06H, register 01H = 00H
```

In the first example, the statement "MULT 00H, 02H" multiplies the 8-bit destination operand (in the register 00H of the register pair 00H, 01H) by the source register 02H operand (09H). The 16-bit product, 0120H, is stored in the register pair 00H, 01H.

6.3.44 NEXT - Next

NEXT

Operation: PC ← @ IP

IP ← IP + 2

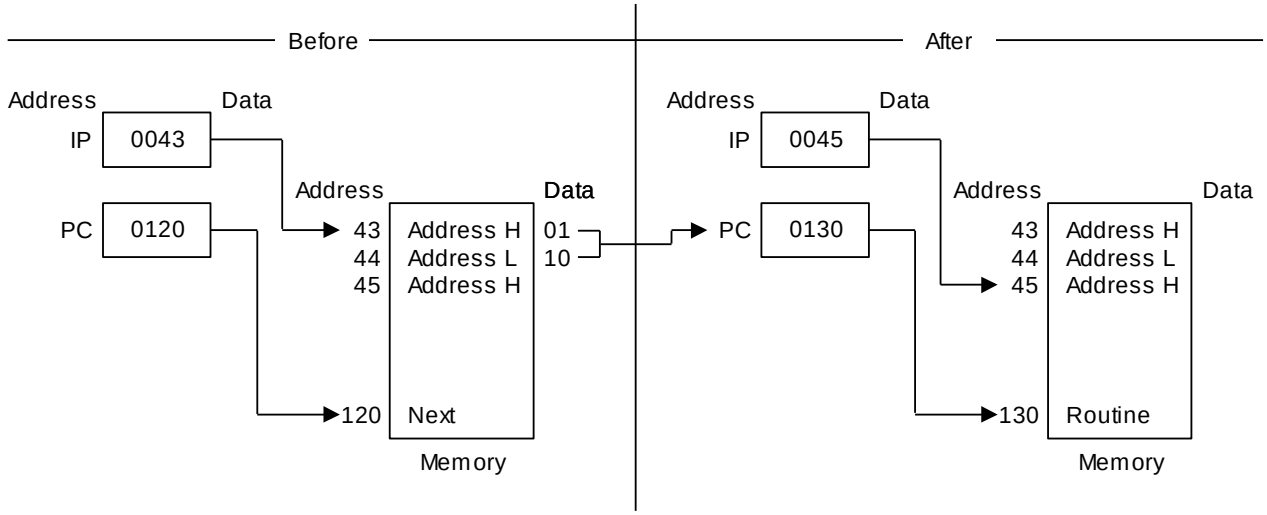
The NEXT instruction is useful when implementing threaded-code languages. The program memory word that is pointed to by the instruction pointer is loaded into the program counter. The instruction pointer is then incremented by two.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	10	0F

Example: The following diagram shows one example of how to use the NEXT instruction.



6.3.45 NOP - No Operation

NOP

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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	FF

Example: When the instruction

```
NOP
```

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

6.3.46 OR - Logical OR

OR dst,src

Operation: dst ← dst OR src

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

Flags:

C: Unaffected.

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result bit 7 is set; cleared otherwise.

V: Always cleared to "0".

D: Unaffected.

H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	42	r	r
		6	43	r	lr
opc | src | dst	3	6	44	R	R
		6	45	R	IR
opc | dst | src	3	6	46	R	IM

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

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

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

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

6.3.47 POP - Pop from Stack

POP dst

Operation: dst ← @SP

SP ← SP + 1

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

Flags: No flags affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc dst	2	8	50	R
		8	51	IR

Examples: Given: Register 00H = 01H, register 01H = 1BH, SPH (0D8H) = 00H, SPL (0D9H) = 0FBH, and stack register 0FBH = 55H:

```
POP     00H     →       Register 00H = 55H, SP = 00FCH
POP     @00H    →       Register 00H = 01H, register 01H = 55H, SP = 00FCH
```

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

6.3.48 POPUD - Pop User Stack (Decrementing)

POPUD dst,src

Operation: dst ← src

IR ← IR – 1

This instruction is used for user-defined stacks in the register file. The contents of the register file location addressed by the user stack pointer are loaded into the destination. The user stack pointer is then decremented.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | src | dst	3	8	92	R	IR

Example: Given: Register 00H = 42H (user stack pointer register), register 42H = 6FH, and register 02H = 70H:

```
POPUD   02H,@00H        →     Register 00H = 41H, register 02H = 6FH, register 42H = 6FH
```

If general register 00H contains the value 42H and register 42H the value 6FH, the statement "POPUD 02H,@00H" loads the contents of register 42H into the destination register 02H. The user stack pointer is then decremented by one, leaving the value 41H.

6.3.49 POPUI - Pop User Stack (Incrementing)

POPUI dst,src

Operation: dst ← src

IR ← IR + 1

The POPUI instruction is used for user-defined stacks in the register file. The contents of the register file location addressed by the user stack pointer are loaded into the destination. The user stack pointer is then incremented.

Flags: No flags are affected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	src	dst	3	8	93	R	IR

Example: Given: Register 00H = 01H and register 01H = 70H:

```
POPUI   02H,@00H        →     Register 00H = 02H, register 01H = 70H, register 02H = 70H
```

If general register 00H contains the value 01H and register 01H the value 70H, the statement "POPUI 02H,@00H" loads the value 70H into the destination general register 02H. The user stack pointer (register 00H) is then incremented by one, changing its value from 01H to 02H.

6.3.50 PUSH - Push To Stack

PUSH src

Operation: SP ← SP – 1

@SP ← src

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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc src	2	8 (internal clock)	70	R
		8 (external clock)		
		8 (internal clock)		
		8 (external clock)	71	IR

Examples: Given: Register 40H = 4FH, register 4FH = 0AAH, SPH = 00H, and SPL = 00H:

```
PUSH    40H     →      Register 40H = 4FH, stack register 0FFH = 4FH,
SPH = 0FFH, SPL = 0FFH

PUSH    @40H    →      Register 40H = 4FH, register 4FH = 0AAH, stack register
0FFH = 0AAH, SPH = 0FFH, SPL = 0FFH
```

In the first example, if the stack pointer contains the value 0000H, and general register 40H the value 4FH, the statement "PUSH 40H" decrements the stack pointer from 0000 to 0FFFFH. It then loads the contents of register 40H into location 0FFFFH and adds this new value to the top of the stack.

6.3.51 PUSHUD - Push User Stack (Decrementing)

PUSHUD dst,src

Operation: IR ← IR – 1

dst ← src

This instruction is used to address user-defined stacks in the register file. PUSHUD decrements the user stack pointer and loads the contents of the source into the register addressed by the decremented stack pointer.

Flags: No flags are affected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst	src	3	8	82	IR	R

Example: Given: Register 00H = 03H, register 01H = 05H, and register 02H = 1AH:

```
PUSHUD  @00H,01H        →       Register 00H = 02H, register 01H = 05H, register 02H = 05H
```

If the user stack pointer (register 00H, for example) contains the value 03H, the statement "PUSHUD @00H, 01H" decrements the user stack pointer by one, leaving the value 02H. The 01H register value, 05H, is then loaded into the register addressed by the decremented user stack pointer.

6.3.52 PUSHUI - Push User Stack (Incrementing)

PUSHUI dst,src

Operation: IR ← IR + 1

dst ← src

This instruction is used for user-defined stacks in the register file. PUSHUI increments the user stack pointer and then loads the contents of the source into the register location addressed by the incremented user stack pointer.

Flags: No flags are affected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst	src	3	8	83	IR	R

Example: Given: Register 00H = 03H, register 01H = 05H, and register 04H = 2AH:

```
PUSHUI  @00H,01H        →       Register 00H = 04H, register 01H = 05H, register 04H = 05H
```

If the user stack pointer (register 00H, for example) contains the value 03H, the statement "PUSHUI @00H, 01H" increments the user stack pointer by one, leaving the value 04H. The 01H register value, 05H, is then loaded into the location addressed by the incremented user stack pointer.

6.3.53 RCF - Reset Carry Flag

RCF RCF

Operation: C ← 0

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

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

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	CF

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

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

6.3.54 RET - Return

RET

Operation: PC ← @SP

SP ← SP + 2

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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	8 (internal stack) 10 (internal stack)	AF

Example: Given: SP = 00FCH, (SP) = 101AH, and PC = 1234:

```
RET     →       PC = 101AH, SP = 00FEH
```

The statement "RET" pops the contents of stack pointer location 00FCH (10H) into the high byte of the program counter. The stack pointer then pops the value in location 00FEH (1AH) into the PC's low byte and the instruction at location 101AH is executed. The stack pointer now points to memory location 00FEH.

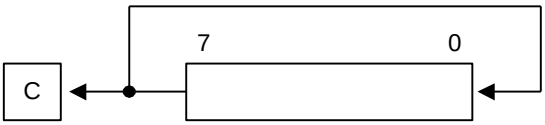
6.3.55 RL - Rotate Left

RL dst

Operation:

C ← dst (7)
dst (0) ← dst (7)
dst (n + 1) ← dst (n), n = 0 to 6

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



Flags:

C: Set if the bit rotated from the most significant bit position (bit 7) was "1".

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result bit 7 is set; cleared otherwise.

V: Set if arithmetic overflow occurred; cleared otherwise.

D: Unaffected.

H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	90	R
		4	91	IR

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

```
RL      00H     →       Register 00H = 55H, C = "1"
RL      @01H    →       Register 01H = 02H, register 02H = 2EH, C = "0"
```

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

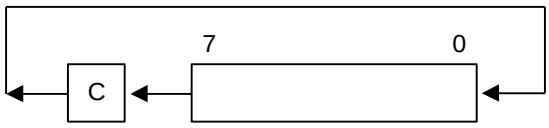
6.3.56 RLC - Rotate Left Through Carry

RLC dst

Operation:

dst (0) ← C
C ← dst (7)
dst (n + 1) ← dst (n), n = 0 to 6

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



Flags:

C: Set if the bit rotated from the most significant bit position (bit 7) was "1".

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result bit 7 is set; cleared otherwise.

V: Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.

D: Unaffected.

H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	10	R
		4	11	IR

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

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

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

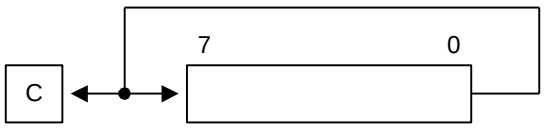
6.3.57 RR - Rotate Right

RR dst

Operation:

C ← dst (0)
dst (7) ← dst (0)
dst (n) ← dst (n + 1), n = 0 to 6

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



Flags:

C: Set if the bit rotated from the least significant bit position (bit zero) was "1".

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result bit 7 is set; cleared otherwise.

V: Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.

D: Unaffected.

H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	E0	R
		4	E1	IR

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

```
RR      00H     □       Register 00H = 98H, C = "1"
RR      @01H    □       Register 01H = 02H, register 02H = 8BH, C = "1"
```

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

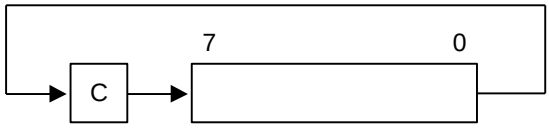
6.3.58 RRC - Rotate Right Through Carry

RRC dst

Operation:

dst (7) ← C
C ← dst (0)
dst (n) ← dst (n + 1), n = 0 to 6

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



Flags:

C: Set if the bit rotated from the least significant bit position (bit zero) was "1".

Z: Set if the result is "0" cleared otherwise.

S: Set if the result bit 7 is set; cleared otherwise.

V: Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.

D: Unaffected.

H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	C0	R
		4	C1	IR

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

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

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

6.3.59 SB0 - Select Bank 0

SB0

Operation: BANK ← 0

The SB0 instruction clears the bank address flag in the FLAGS register (FLAGS.0) to logic zero, selecting bank 0 register addressing in the set 1 area of the register file.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	4F

Example: The statement

```
SB0
```

clears FLAGS.0 to "0", selecting bank 0 register addressing.

6.3.60 SB1 - Select Bank 1

SB1

Operation: BANK ← 1

The SB1 instruction sets the bank address flag in the FLAGS register (FLAGS.0) to logic one, selecting bank 1 register addressing in the set 1 area of the register file. (Bank 1 is not implemented in some S3C8-series microcontrollers.)

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	5F

Example: The statement

```
SB1
```

sets FLAGS.0 to "1", selecting bank 1 register addressing, if implemented.

6.3.61 SBC - Subtract with Carry

SBC dst,src

Operation: dst ← dst – src – c

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

Flags:

C: Set if a borrow occurred (src > dst); cleared otherwise.

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result is negative; cleared otherwise.

V: Set if arithmetic overflow occurred, that is, if the operands were of opposite sign and the sign of the result is the same as the sign of the source; cleared otherwise.

D: Always set to "1".

H: Cleared if there is a carry from the most significant bit of the low-order four bits of the result; set otherwise, indicating a "borrow".

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src		2	4	32	r	r
			6	33	r	lr
opc | src | dst		3	6	34	R	R
			6	35	R	IR
opc | dst | src		3	6	36	R	IM

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

```
SBC     R1,R2          □      R1 = 0CH, R2 = 03H
SBC     R1,@R2 □              R1 = 05H, R2 = 03H, register 03H = 0AH
SBC     01H,02H□              Register 01H = 1CH, register 02H = 03H
SBC     01H,@02H       □      Register 01H = 15H,register 02H = 03H, register 03H = 0AH
SBC     01H,#8AH       □      Register 01H =  5H; C, S, and V = "1"
```

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

6.3.62 SCF - Set Carry Flag

SCF

Operation: C ← 1

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

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

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	DF

Example: The statement

```
SCF
```

sets the carry flag to logic one.

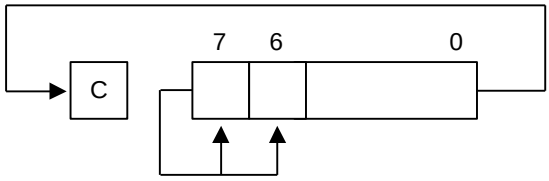
6.3.63 SRA - Shift Right Arithmetic

SRA dst

Operation:

dst (7) ← dst (7)
C ← dst (0)
dst (n) ← dst (n + 1), n = 0 to 6

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



Flags:

C: Set if the bit shifted from the LSB position (bit zero) was "1".

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result is negative; cleared otherwise.

V: Always cleared to "0".

D: Unaffected.

H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	D0	R
		4	D1	IR

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

```
SRA     00H     □       Register 00H = 0CD, C = "0"
SRA     @02H    □       Register 02H = 03H, register 03H = 0DEH, C = "0"
```

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

6.3.64 SRP/SRP0/SRP1 - Set Register Pointer

SRP src

SRP0 src

SRP1 src

Operation:

If src (1) = 1 and src (0) = 0 then:	RP0 (3–7)	□	src (3–7)
If src (1) = 0 and src (0) = 1 then:	RP1 (3–7)	□	src (3–7)
If src (1) = 0 and src (0) = 0 then:	RP0 (4–7)	□	src (4–7),
	RP0 (3) □		0
	RP1 (4–7)	□	src (4–7),
	RP1 (3) □		1

The source data bits one and zero (LSB) determine whether to write one or both of the register pointers, RP0 and RP1. Bits 3–7 of the selected register pointer are written unless both register pointers are selected. RP0.3 is then cleared to logic zero and RP1.3 is set to logic one.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode src
opc src	2	4	31	IM

Examples: The statement

```
SRP  #40H
```

Sets register pointer 0 (RP0) at location 0D6H to 40H and register pointer 1 (RP1) at location 0D7H to 48H.

The statement "SRP0 #50H" sets RP0 to 50H, and the statement "SRP1 #68H" sets RP1 to 68H.

6.3.65 STOP - Stop Operation

STOP

Operation:

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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	1	4	7F	–	–

Example: The statement

```
STOP
```

halts all microcontroller operations.

6.3.66 SUB - Subtract

SUB dst,src

Operation: dst ← dst – src

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

Flags:

C: Set if a "borrow" occurred; cleared otherwise.

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result is negative; cleared otherwise.

V: Set if arithmetic overflow occurred, that is, if the operands were of opposite signs and the sign of the result is of the same as the sign of the source operand; cleared otherwise.

D: Always set to "1".

H: Cleared if there is a carry from the most significant bit of the low-order four bits of the result; set otherwise indicating a "borrow".

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	22	r	r
		6	23	r	lr
opc | src | dst	3	6	24	R	R
		6	25	R	IR
opc | dst | src	3	6	26	R	IM

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

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

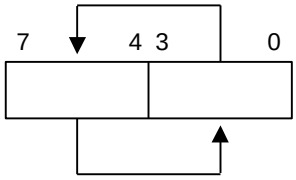
In the first example, if working register R1 contains the value 12H and if register R2 contains the value 03H, the statement "SUB R1,R2" subtracts the source value (03H) from the destination value (12H) and stores the result (0FH) in destination register R1.

6.3.67 SWAP - Swap Nibbles

SWAP dst

Operation: dst (0 – 3) ↔ dst (4 – 7)

The contents of the lower four bits and upper four bits of the destination operand are swapped.



Flags:

C: Undefined.

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result bit 7 is set; cleared otherwise.

V: Undefined.

D: Unaffected.

H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	F0	R
		4	F1	IR

Examples: Given: Register 00H = 3EH, register 02H = 03H, and register 03H = 0A4H:

```
SWAP    00H     □       Register 00H = 0E3H
SWAP    @02H    □       Register 02H = 03H, register 03H = 4AH
```

In the first example, if general register 00H contains the value 3EH (00111110B), the statement "SWAP 00H" swaps the lower and upper four bits (nibbles) in the 00H register, leaving the value 0E3H (11100011B).

6.3.68 TCM - Test Complement Under Mask

TCM dst,src

Operation: (NOT dst) AND src

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

Flags:

C: Unaffected.

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result bit 7 is set; cleared otherwise.

V: Always cleared to "0".

D: Unaffected.

H: Unaffected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | src		2	4	62	r	r
				6	63	r	lr
opc	src	dst	3	6	64	R	R
				6	65	R	IR
opc	dst	src	3	6	66	R	IM

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

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

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

6.3.69 TM - Test Under Mask

TM dst,src

Operation: dst AND src

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

Flags:

C: Unaffected.

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result bit 7 is set; cleared otherwise.

V: Always reset to "0".

D: Unaffected.

H: Unaffected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | src		2	4	72	r	r
				6	73	r	lr
opc	src	dst	3	6	74	R	R
				6	75	R	IR
opc	dst	src	3	6	76	R	IM

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

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

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

6.3.70 WFI - Wait for Interrupt

WFI

Operation:

The CPU is effectively halted until an interrupt occurs, except that DMA transfers can still take place during this wait state. The WFI status can be released by an internal interrupt, including a fast interrupt .

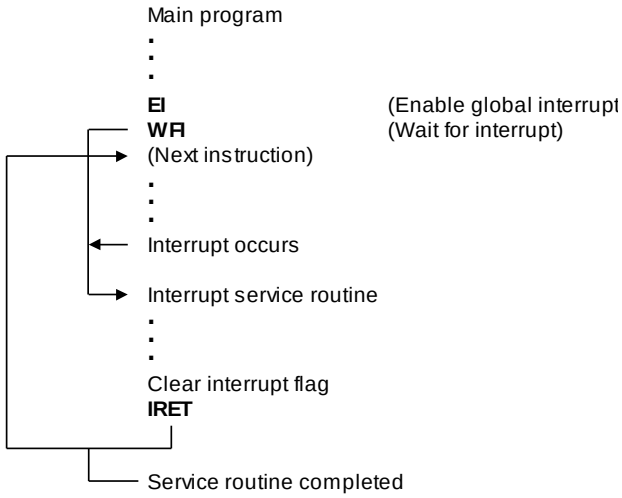
Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4n	3F

NOTE: (n = 1, 2, 3, …)

Example: The following sample program structure shows the sequence of operations that follow a "WFI" statement:



6.3.71 XOR - Logical Exclusive OR

XOR dst,src

Operation: dst ← dst XOR src

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

Flags:

C: Unaffected.

Z: Set if the result is "0"; cleared otherwise.

S: Set if the result bit 7 is set; cleared otherwise.

V: Always reset to "0".

D: Unaffected.

H: Unaffected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | src		2	4	B2	r	r
				6	B3	r	lr
opc	src	dst	3	6	B4	R	R
				6	B5	R	IR
opc	dst	src	3	6	B6	R	IM

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

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

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

7 Clock Circuit

7.1 Overview

The S3F8S6B microcontroller has two oscillator circuits: a main clock and a sub clock circuit. The CPU and peripheral hardware operate on the system clock frequency supplied through these circuits. The maximum CPU clock frequency of S3F8S6B is determined by CLKCON register settings.

7.1.1 System Clock Circuit

The system clock circuit has the following components:

- External crystal, ceramic resonator, RC oscillation source, or an external clock source
- Oscillator stop and wake-up functions
- Programmable frequency divider for the CPU clock (fxx divided by 1, 2, 8, or 16)
- System clock control register, CLKCON
- Oscillator control register, OSCCON and STOP control register, STPCON

7.1.2 CPU Clock Notation

In this document, the following notation is used for descriptions of the CPU clock;

- fx: main clock
- fxt: sub clock
- fxx: selected system clock

7.2 Main Oscillator Circuits

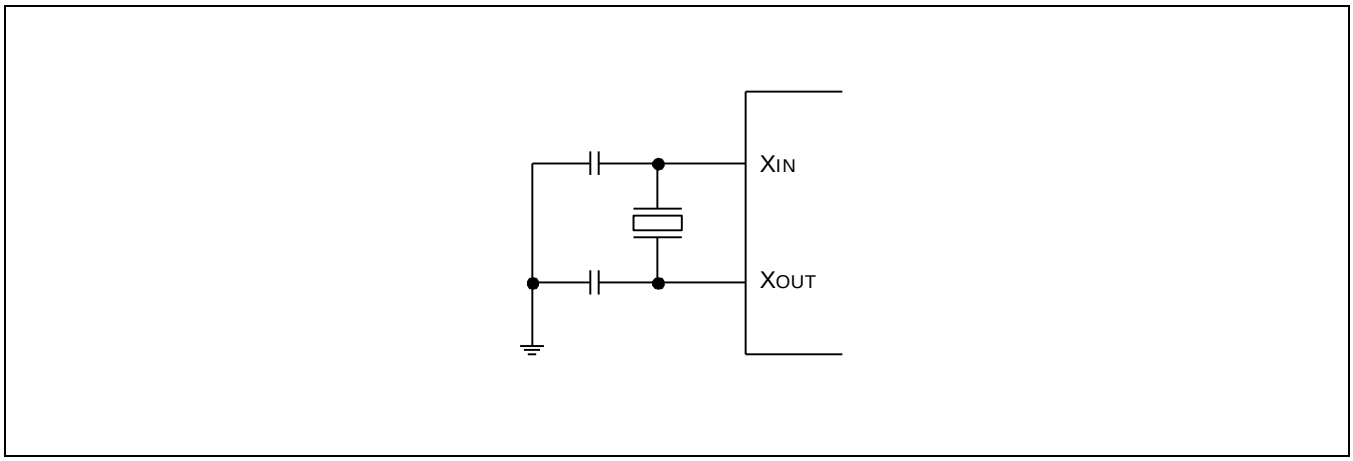


Figure 7-1 Crystal/Ceramic Oscillator (fX)

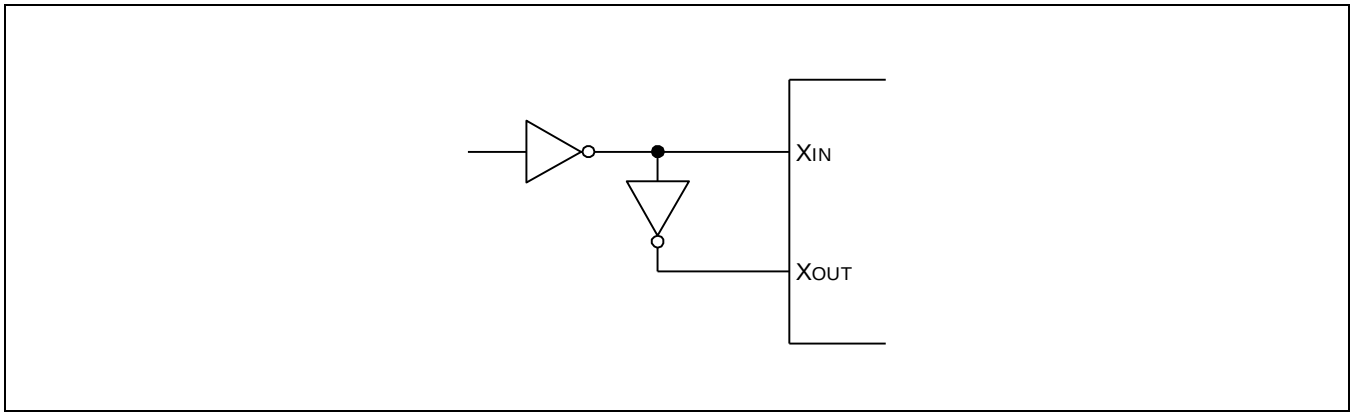


Figure 7-2 External Oscillator (fX)

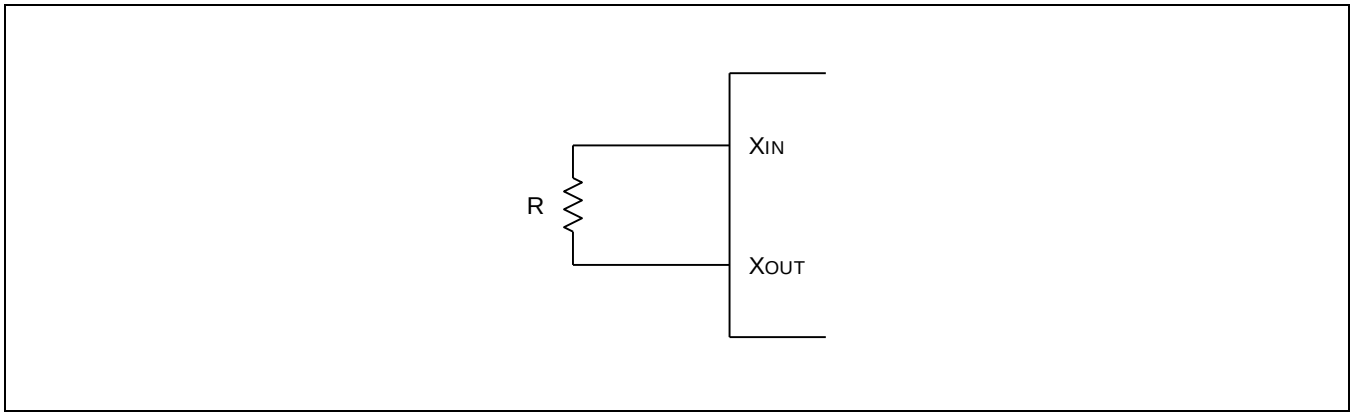


Figure 7-3 RC Oscillator (fX)

7.3 Sub Oscillator Circuits

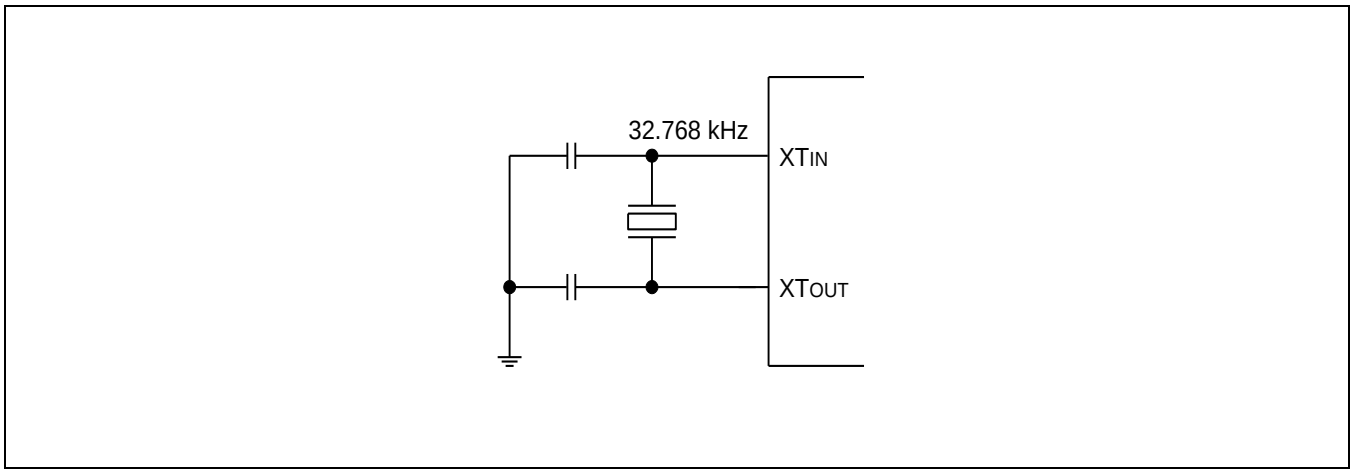


Figure 7-4 Crystal Oscillator (fxt)

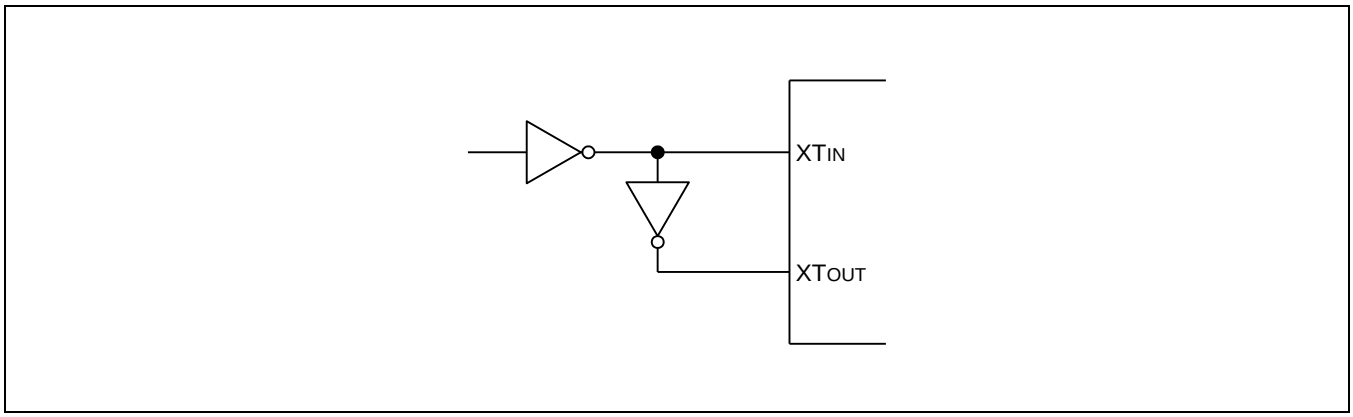


Figure 7-5 External Oscillator (fxt)

7.4 Clock Status During Power-Down Modes

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

- In Stop mode, the main oscillator is halted. Stop mode is released, and the oscillator is started, by a reset operation or an external interrupt (with RC delay noise filter), and can be released by internal interrupt too when the sub-system oscillator is running and watch timer is operating with sub-system clock.
- In Idle mode, the internal clock signal is gated to the CPU, but not to interrupt structure, timers and timer/counters. Idle mode is released by a reset or by an external or internal interrupt.

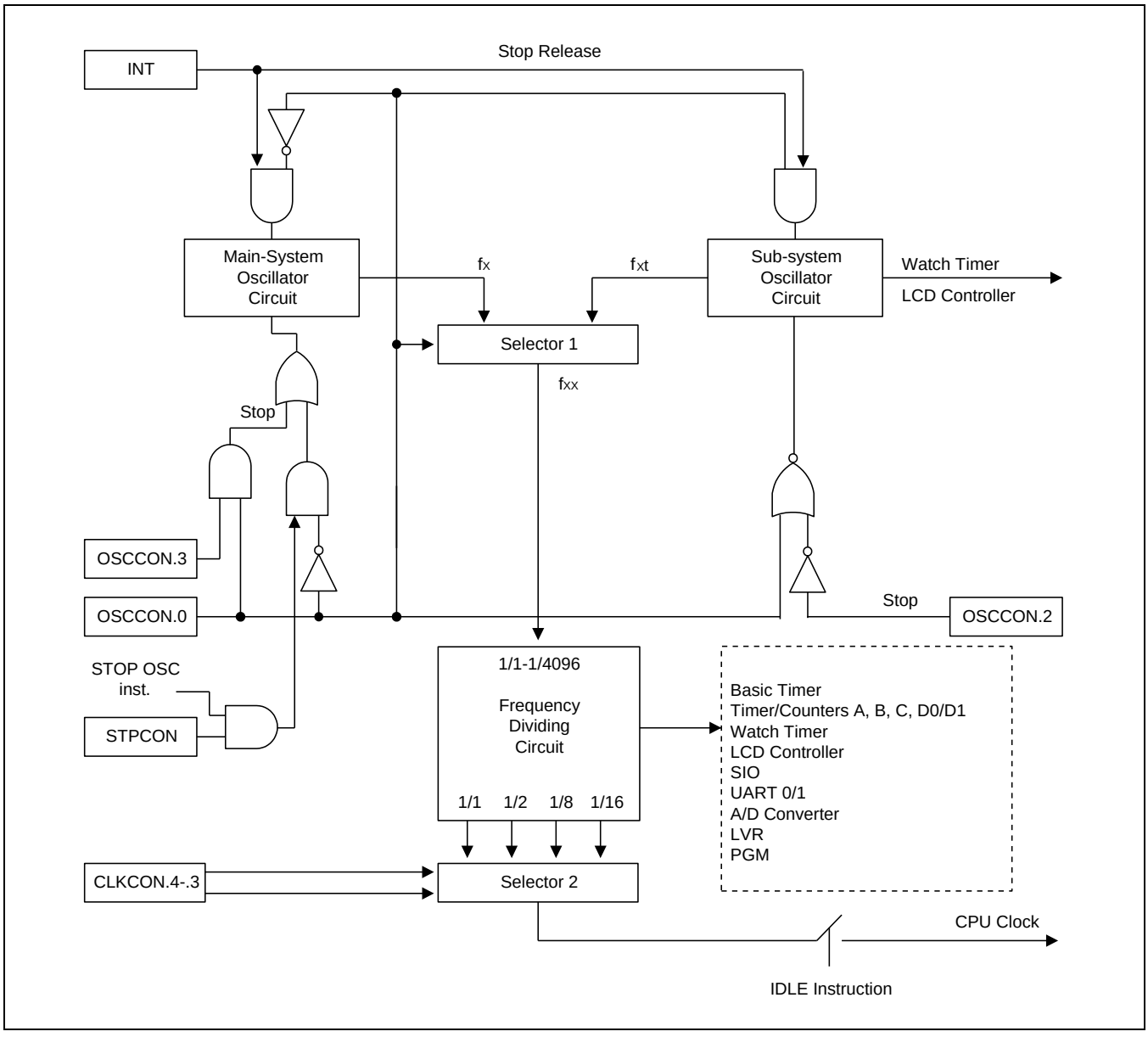


Figure 7-6 System Clock Circuit Diagram

7.5 System Clock Control Register (CLKCON)

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

- Oscillator frequency divide-by value

After the main oscillator is activated, and the fxx/16 (the slowest clock speed) is selected as the CPU clock. If necessary, you can then increase the CPU clock speed fxx/8, fxx/2, or fxx/1.

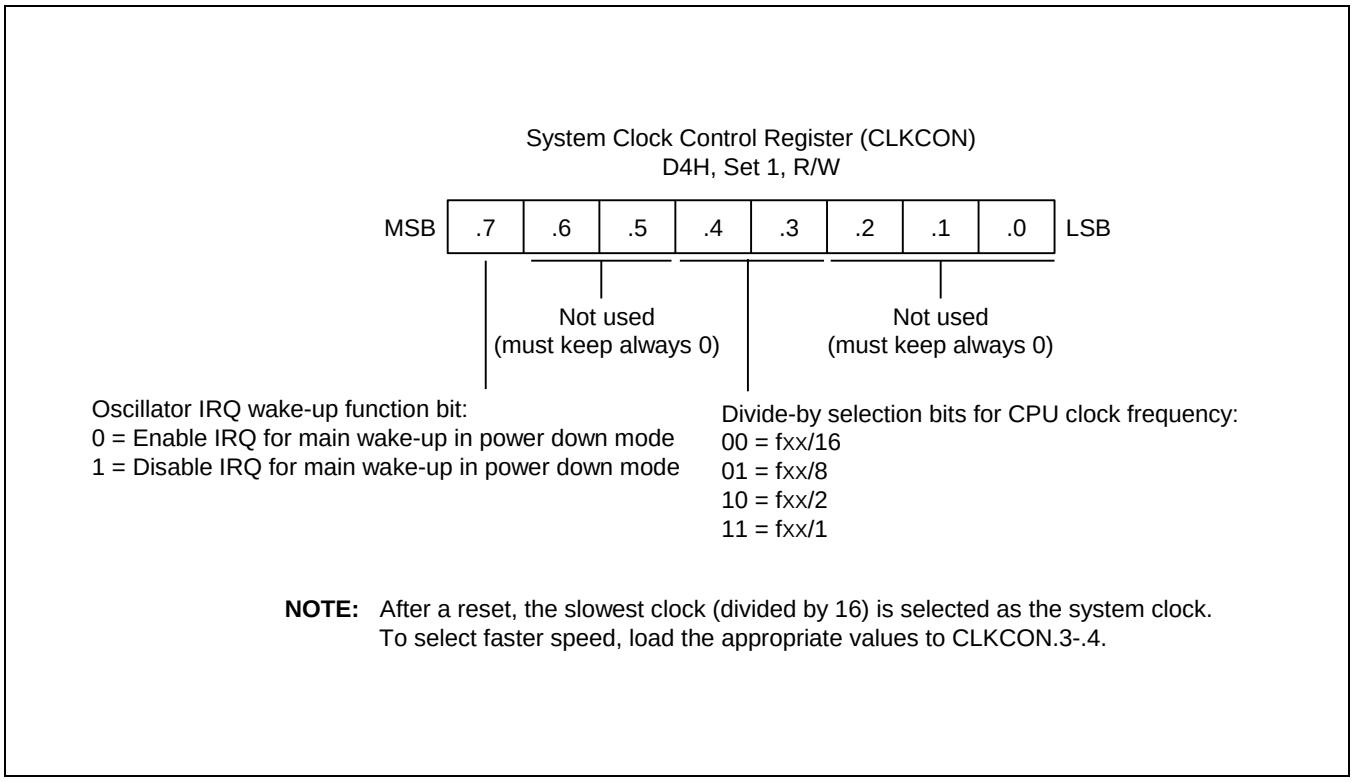


Figure 7-7 System Clock Control Register (CLKCON)

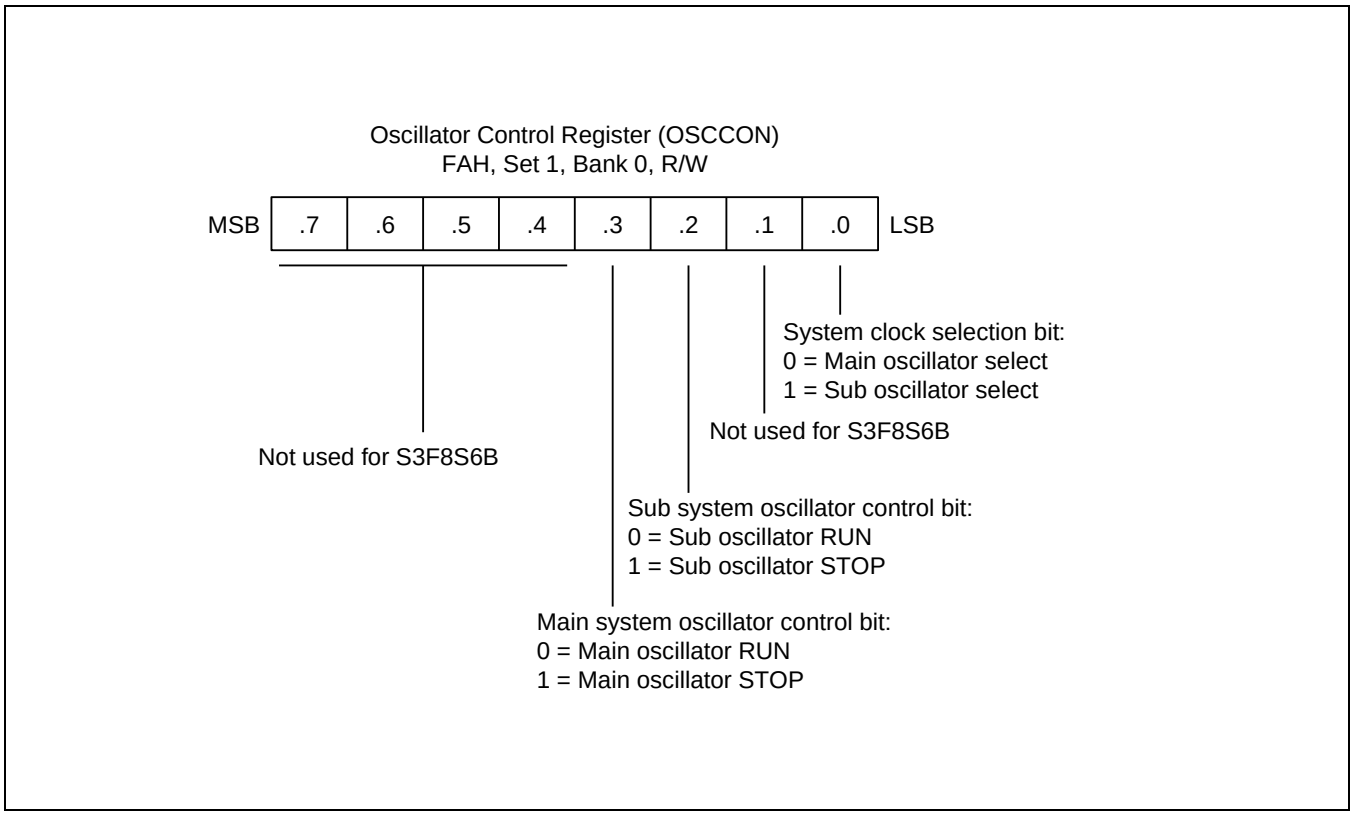


Figure 7-8 Oscillator Control Register (OSCCON)

7.6 Stop Control Register (STPCON)

The STOP control register, STPCON, is located in the bank 0 of set1, address F5H. It is read/write addressable and has the following functions:

- Enable/Disable STOP instruction

After a reset, the STOP instruction is disabled, because the value of STPCON is "other values".
If necessary, you can use the STOP instruction by setting the value of STPCON to "10100101B".

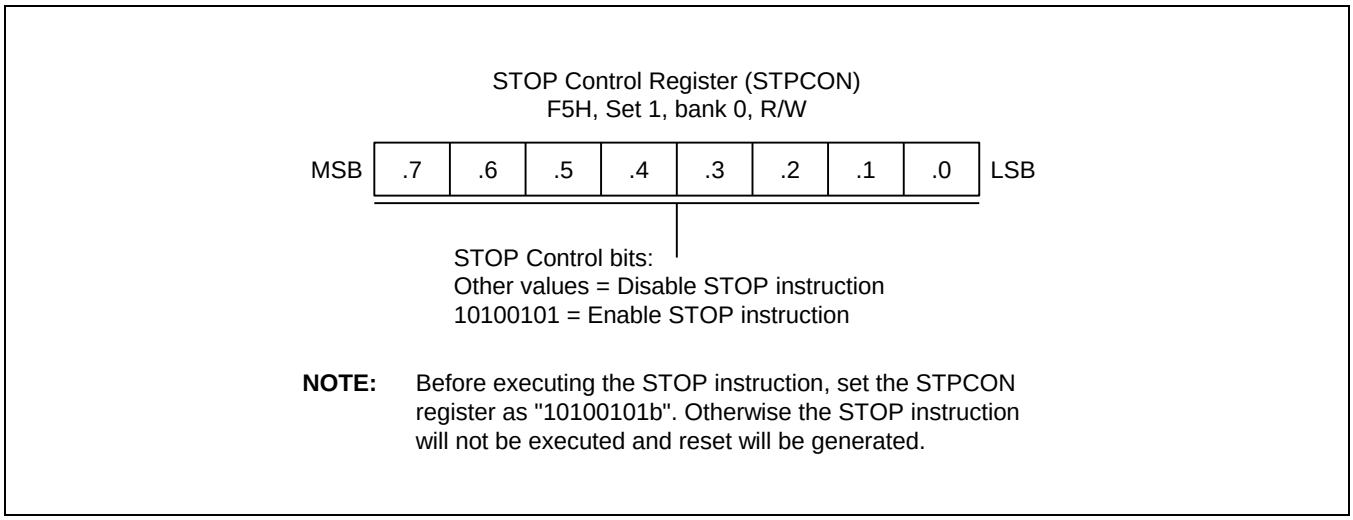


Figure 7-9 STOP Control Register (STPCON)

Example 7-1 How to Use Stop Instruction

This example shows how to go Stop mode when a main clock is selected as the system clock.

```
LD      STOPCON,#1010010B           ;       Enable STOP instruction
STOP                                ;       Enter Stop mode
NOP
NOP
NOP                                 ;       Release Stop mode
LD      STOPCON,#00000000B          ;       Disable STOP instruction
```

7.7 Switching the CPU Clock

Data loading in the oscillator control register, OSCCON, determine whether a main or a sub clock is selected as the CPU clock, and also how this frequency is to be divided by setting CLKCON. This makes it possible to switch dynamically between main and sub clocks and to modify operating frequencies.

OSCCON.0 selects the main clock (fx) or the sub clock (fxt) for the CPU clock. OSCCON .3 start or stop main clock oscillation and OSCCON.2 start or stop sub clock oscillation. CLKCON.4–.3 controls the frequency divider circuit, and divides the selected fxx clock by 1, 2, 8 and 16. If the sub clock (fxt) is selected for system clock, the CLKCON.4–.3 must be set to "11".

For example, you are using the default CPU clock (normal operating mode and a main clock of fx/16) and you want to switch from the fx clock to a sub clock and to stop the main clock. To do this, you need to set CLKCON.4-.3 to "11", OSCCON.0 to "1", and OSCCON.3 to "1" by turns. This switches the clock from fx to fxt and stops main clock oscillation.

The following steps must be taken to switch from a sub clock to the main clock: first, set OSCCON.3 to "0" to enable main clock oscillation. Then, after a certain number of machine cycles have elapsed, select the main clock by setting OSCCON.0 to "0".

Example 7-2 Switching the CPU clock

1. This example shows how to change from the main clock to the sub clock:

```
MA2SUB  OR      CLKCON,#18H             ;       Non-divided clock for system clock
                LD      OSCCON,#01H     ;       Switches to the sub clock
                CALL    DLY16           ;       Delay 16 ms
                OR      OSCCON,#08H     ;       Stop the main clock oscillation
                RET
```

2. This example shows how to change from sub clock to main clock:

```
SUB2MA  AND     OSCCON,#07H             ;       Start the main clock oscillation
                CALL    DLY16           ;       Delay 16 ms
                AND     OSCCON,#06H     ;       Switch to the main clock
                RET
DLY16           SRP     #0C0H
                LD      R0,#20H
DEL             NOP
                DJNZ    R0,DEL
                RET
```

8 RESET and Power-Down

8.1 System Reset

8.1.1 Overview

During a power-on reset, the voltage at V_{DD} goes to High level and the nRESET pin is forced to Low level. The nRESET signal is input through a schmitt trigger circuit where it is then synchronized with the CPU clock. This procedure brings the S3F8S6B into a known operating status.

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

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

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

- All interrupt is disabled.
- The watchdog function (basic timer) is enabled.
- Ports 0-6 and set to input mode, and all pull-up resistors are disabled for the I/O port.
- Peripheral control and data register settings are disabled and reset to their default hardware values.
- The program counter (PC) is loaded with the program reset address in the ROM, 0100H.
- When the programmed oscillation stabilization time interval has elapsed, the instruction stored in ROM location 0100H (and 0101H) is fetched and executed at normal mode by smart option.
- The reset address at ROM can be changed by Smart Option in the S3F8S6B (full-Flash device). Refer to "The Chapter 21. Embedded Flash Memory Interface" for more detailed contents.

8.1.2 Normal Mode Reset Operation

In normal mode, the Test pin is tied to V_{SS}. A reset enables access to the 64KB on-chip ROM. (The external interface is not automatically configured).

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

8.1.3 Hardware Reset Values

Table 8-1, 8-2, 8-3, 8-4 list the reset values for CPU and system registers, peripheral control registers, and peripheral data registers following a reset operation. The following notation is used to represent reset values:

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

Table 8.1 S3F8S6B Set 1 Register and Values After RESET

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

NOTE:

1. An "x" means that the bit value is undefined following reset.
2. A dash ('–') means that the bit is neither used nor mapped, but the bit is read as "0".

Table 8.2 S3F8S6B Set 1, Bank 0 Register and Values after RESET

Register Name	Mnemonic	Address		Bit Values after RESET							
		Dec	Hex	7	6	5	4	3	2	1	0
A/D Converter Data Register (High Byte)	ADDATAH	208	D0H	x	x	x	x	x	x	x	x
A/D Converter Data Register (Low Byte)	ADDATAL	209	D1H	–	–	–	–	–	–	x	x
A/D Converter Control Register	ADCON	210	D2H	–	0	0	0	0	0	0	0
Timer A Counter Register	TACNT	224	E0H	0	0	0	0	0	0	0	0
Timer A Data Register	TADATA	225	E1H	1	1	1	1	1	1	1	1
Timer A Control Register	TACON	226	E2H	0	0	0	0	0	0	0	0
Timer B Control Register	TBCON	227	E3H	0	0	0	0	0	0	0	0
Timer B Data Register (High Byte)	TBDATAH	228	E4H	1	1	1	1	1	1	1	1
Timer B Data Register (Low Byte)	TBDATAL	229	E5H	1	1	1	1	1	1	1	1
Watch Timer Control Register	WTCON	230	E6H	0	0	0	0	0	0	0	0
SIO Control Register	SIOCON	231	E7H	0	0	0	0	0	0	0	0
SIO Data Register	SIODATA	232	E8H	0	0	0	0	0	0	0	0
SIO Pre-scaler Register	SIOPS	233	E9H	0	0	0	0	0	0	0	0
Timer C Counter Register	TCCNT	234	EAH	0	0	0	0	0	0	0	0
Timer C Data Register	TCDATA	235	EBH	1	1	1	1	1	1	1	1
Timer C Control Register	TCCON	236	ECH	0	0	0	0	0	0	0	0
Locations EDH–EFH are not mapped											
LCD Control Register	LCON	240	F0H	0	0	0	0	0	0	0	0
LCD Mode Register	LMOD	241	F1H	–	–	–	–	–	0	0	0
Locations F2H–F3H are not mapped											
Interrupt Pending Register	INTPND	244	F4H	–	–	0	0	0	0	0	0
STOP control register	STPCON	245	F5H	0	0	0	0	0	0	0	0
Flash Memory Sector Address Register (High Byte)	FMSECH	246	F6H	0	0	0	0	0	0	0	0
Flash Memory Sector Address Register (Low Byte)	FMSECL	247	F7H	0	0	0	0	0	0	0	0
Flash Memory User Programming Enable Register	FMUSR	248	F8H	0	0	0	0	0	0	0	0
Flash Memory Control Register	FMCON	249	F9H	0	0	0	0	0	–	–	0
Oscillator Control Register	OSCCON	250	FAH	–	–	–	–	0	0	–	0
Locations FBH–FCH are not mapped											
Basic Timer Counter	BTCNT	253	FDH	0	0	0	0	0	0	0	0
Location FEH are not mapped											
Interrupt Priority Register	IPR	255	FFH	x	x	x	x	x	x	x	x

Table 8.3 S3F8S6B Set 1, Bank 1 Register and Values after RESET

Register Name	Mnemonic	Address		Bit Values after RESET							
		Dec	Hex	7	6	5	4	3	2	1	0
Pattern Generation Control Register	PGCON	208	D0H	–	–	–	–	0	0	0	0
Pattern Generation Data Register	PGDATA	209	D1H	0	0	0	0	0	0	0	0
Location D2H is not mapped.											
Port 0 Control Register (High Byte)	P0CONH	224	E0H	0	0	0	0	0	0	0	0
Port 0 Control Register (Low Byte)	P0CONL	225	E1H	0	0	0	0	0	0	0	0
Port 1 Control Register (High Byte)	P1CONH	226	E2H	0	0	0	0	0	0	0	0
Port 1 Control Register (Low Byte)	P1CONL	227	E3H	0	0	0	0	0	0	0	0
Port 1 Pull-up Resistor Enable Register	P1PUR	228	E4H	0	0	0	0	0	0	0	0
Port 1 N-Channel Open-drain Mode Register	PNE1	229	E5H	0	0	0	0	0	0	0	0
Port 2 Control Register (High Byte)	P2CONH	230	E6H	0	0	0	0	0	0	0	0
Port 2 Control Register (Low Byte)	P2CONL	231	E7H	0	0	0	0	0	0	0	0
Port 2 Interrupt Control Register (High Byte)	P2INTH	232	E8H	0	0	0	0	0	0	0	0
Port 2 Interrupt Control Register (Low Byte)	P2INTL	233	E9H	0	0	0	0	0	0	0	0
Port 2 Interrupt Pending Register	P2PND	234	EAH	0	0	0	0	0	0	0	0
Port 3 Control Register (High Byte)	P3CONH	235	EBH	0	0	0	0	0	0	0	0
Port 3 Control Register (Middle Byte)	P3CONM	236	ECH	0	0	0	0	0	0	0	0
Port 3 Control Register (Low Byte)	P3CONL	237	EDH	–	–	0	0	0	0	0	0
Port 3 Pull-up Resistor Enable Register	P3PUR	238	EEH	0	0	0	0	0	0	0	0
Port 3 N-Channel Open-drain Mode Register	PNE3	239	EFH	0	0	0	0	0	0	0	0
Port 5 Control Register (High Byte)	P5CONH	240	F0H	0	0	0	0	0	0	0	0
Port 5 Control Register (Low Byte)	P5CONL	241	F1H	0	0	0	0	0	0	0	0
Port 6 Control Register (High Byte)	P6CONH	242	F2H	–	–	–	–	0	0	0	0
Port 6 Control Register (Low Byte)	P6CONL	243	F3H	0	0	0	0	0	0	0	0
Port 6 Pull-up Resistor Enable Register	P6PUR	244	F4H	–	–	0	0	0	0	0	0
Port 6 N-Channel Open-drain Mode Register	PNE6	245	F5H	–	–	0	0	0	0	0	0
Timer D0 Counter Register (High Byte)	TD0CNTH	246	F6H	0	0	0	0	0	0	0	0
Timer D0 Counter Register (Low Byte)	TD0CNTL	247	F7H	0	0	0	0	0	0	0	0
Timer D0 Data Register (High Byte)	TD0DATAH	248	F8H	1	1	1	1	1	1	1	1
Timer D0 Data Register (Low Byte)	TD0DATAL	249	F9H	1	1	1	1	1	1	1	1

Register Name	Mnemonic	Address		Bit Values after RESET							
		Dec	Hex	7	6	5	4	3	2	1	0
Timer D0 Control Register	TD0CON	250	FAH	0	0	0	0	0	0	0	0
Timer D1 Control Register	TD1CON	251	FBH	0	0	0	0	0	0	0	0
Timer D1 Counter Register (High Byte)	TD1CNTH	252	FCH	0	0	0	0	0	0	0	0
Timer D1 Counter Register (Low Byte)	TD1CNTL	253	FDH	0	0	0	0	0	0	0	0
Timer D1 Data Register (High Byte)	TD1DATAH	254	FEH	1	1	1	1	1	1	1	1
Timer D1 Data Register (Low Byte)	TD1DATAL	255	FFH	1	1	1	1	1	1	1	1

NOTE:

1. An "x' means that the bit value is undefined following reset.
2. A dash ('–') means that the bit is neither used nor mapped, but the bit is read as "0".

Table 8.4 S3F8S6B Page 8 Register and Values After RESET

Register Name	Mnemonic	Address		Bit Values After RESET							
		Dec	Hex	7	6	5	4	3	2	1	0
Port 0 Data Register	P0	0	00H	0	0	0	0	0	0	0	0
Port 1 Data Register	P1	1	01H	0	0	0	0	0	0	0	0
Port 2 Data Register	P2	2	02H	0	0	0	0	0	0	0	0
Port 3 Data Register	P3	3	03H	0	0	0	0	0	0	0	0
Port 4 Data Register	P4	4	04H	0	0	0	0	0	0	0	0
Port 5 Data Register	P5	5	05H	0	0	0	0	0	0	0	0
Port 6 Data Register	P6	6	06H	x	x	0	0	0	0	0	0
Locations 07H–0BH are not mapped.											
Port 4 Control Register (High Byte)	P4CONH	12	0CH	0	0	0	0	0	0	0	0
Port 4 Control Register (Low Byte)	P4CONL	13	0DH	0	0	0	0	0	0	0	0
Port 4 Interrupt Control Register (High Byte)	P4INTH	14	0EH	0	0	0	0	0	0	0	0
Port 4 Interrupt Control Register (Low Byte)	P4INTL	15	0FH	0	0	0	0	0	0	0	0
Port 4 Interrupt Pending Register	P4PND	16	10H	0	0	0	0	0	0	0	0
Locations 11H–13H are not mapped.											
UART 0 Control Register (High Byte)	UART0CONH	20	14H	0	0	0	0	0	0	0	0
UART 0 Control Register (Low Byte)	UART0CONL	21	15H	0	0	0	0	0	0	0	0
UART 0 Data Register	UDATA0	22	16H	x	x	x	x	x	x	x	x
UART 0 Baud Rate Data Register	BRDATA0	23	17H	1	1	1	1	1	1	1	1
UART 1 Control Register (High Byte)	UART1CONH	24	18H	0	0	0	0	0	0	0	0
UART 1 Control Register (Low Byte)	UART1CONL	25	19H	0	0	0	0	0	0	0	0
UART 1 Data Register	UDATA1	26	1AH	x	x	x	x	x	x	x	x
UART 1 Baud Rate Data Register	BRDATA1	27	1BH	1	1	1	1	1	1	1	1
Locations 1CH–2FH are not mapped.											

NOTE:

1. An "x" means that the bit value is undefined following reset.
2. A dash ('–') means that the bit is neither used nor mapped, but the bit is read as "0".

8.2 Power-Down Modes

8.2.1 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 3 μA. All system functions stop when the clock "freezes", but data stored in the internal register file is retained. Stop mode can be released in one of two ways: by a reset or by interrupts, for more details see Figure 7-6.

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

8.2.1.1 Using nRESET to Release Stop Mode

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

8.2.1.2 Using an External Interrupt to Release Stop Mode

External interrupts with an RC-delay noise filter circuit can be used to release Stop mode. Which interrupt you can use to release Stop mode in a given situation depends on the microcontroller's current internal operating mode. The external interrupts in the S3F8S6B interrupt structure that can be used to release Stop mode are:

- External interrupts P2.0–P2.7, P4.0–P4.7 (INT0–INT15)

Please note the following conditions for Stop mode release:

- If you release Stop mode using an external interrupt, the current values in system and peripheral control registers are unchanged except STPCON register.
- If you use an internal or external interrupt for Stop mode release, you can also program the duration of the oscillation stabilization interval. To do this, you must make the appropriate control and clock settings before entering Stop mode.
- When the Stop mode is released by external interrupt, the CLKCON.4 and CLKCON.3 bit-pair setting remains unchanged and the currently selected clock value is used.
- The external interrupt is serviced when the Stop mode release occurs. Following the IRET from the service routine, the instruction immediately following the one that initiated Stop mode is executed.

8.2.1.3 Using an Internal Interrupt to Release Stop Mode

Activate any enabled interrupt, causing Stop mode to be released. Other things are same as using external interrupt.

8.2.1.4 How to Enter into Stop Mode

Handling STPCON register then writing STOP instruction (keep the order).

```
LD      STPCON,#10100101B
STOP
NOP
NOP
NOP
```

8.2.2 Idle Mode

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

There are two ways to release Idle mode:

1. Execute a reset. All system and peripheral control registers are reset to their default values and the contents of all data registers are retained. The reset automatically selects the slow clock fxx/16 because CLKCON.4 and CLKCON.3 are cleared to "00B". If interrupts are masked, a reset is the only way to release Idle mode.

2. Activate any enabled interrupt, causing Idle mode to be released. When you use an interrupt to release Idle mode, the CLKCON.4 and CLKCON.3 register values remain unchanged, and the currently selected clock value is used. The interrupt is then serviced. When the return-from-interrupt (IRET) occurs, the instruction immediately following the one that initiated Idle mode is executed.

9 I/O Port

9.1 Overview

The S3F8S6B microcontroller has seven bit-programmable I/O ports, P0–P6. The port 6 is a 6-bit port and the others are 8-bit ports. This gives a total of 54 I/O pins. Each port can be flexibly configured to meet application design requirements. The CPU accesses ports by directly writing or reading port registers. No special I/O instructions are required. *Table 9.1* provides a general overview of the S3F8S6B I/O port functions.

Table 9.1 S3F8S6B Port Configuration Overview

Port	Configuration Options
0	1-bit programmable I/O port. Input or push-pull output mode selected by software; software assignable pull-ups. P0.0–P0.7 can alternately be used as outputs for LCD COM/SEG.
1	1-bit programmable I/O port. Input or push-pull, open-drain output mode selected by software; software assignable pull-ups. Alternately P1.0–P1.7 can be used as TxD0, RxD0, TxD1, RxD1, TAOUT/TAPWM/TACAP, TACLK or LCD SEG.
2	1-bit programmable I/O port. Schmitt trigger input or push-pull output mode selected by software; software assignable pull-ups. P2.0–P2.7 can be used as inputs for external interrupts INT0–INT7 (with noise filter, interrupt enable and pending control). Alternatively P2.0-P2.7 can be used as LCD SEG.
3	1-bit programmable I/O port. Input or push-pull, open-drain output mode selected by software; software assignable pull-ups. Alternately P3.0–P3.7 can be used as TBPWM/PG0, TCOUT/PG1, PG2, TD0OUT/TD0PWM/TD0CAP/PG3, TD0CLK/PG4, TD1OUT/TD1PWM/TD1CAP/PG5, TD1CLK/PG6, BUZ/PG7 or LCD SEG.
4	The P4.2-P4.7 are the I/O port with bit-programmable pins, but the P4.0/P4.1 are the I/O port with two-bits-programmable pins. Schmitt trigger input or push-pull in output mode selected by software; software assignable pull-ups. P4.0–P4.7 can be used as inputs for external interrupts INT8–INT15 (with noise filter, interrupt enable and pending control). Alternatively P4.0-P4.7 can be used as AD0–AD7.
5	1-bit programmable I/O port. Input or push-pull output mode selected by software; software assignable pull-ups. P5.0–P5.7 can alternately be used as V_{LC0}–V_{LC3}, CA, CB, XT_{OUT}, XT_{IN}.
6	1-bit programmable I/O port. Input or push-pull, open-drain output mode selected by software; software assignable pull-ups. P6.0–P6.5 can alternately be used as SCK, SI, SO, or LCD SEG.

9.2 Port Data Registers

Table 9.2 gives you an overview of the register locations of all twelve S3F8S6B I/O port data registers. Data registers for ports 0, 1, 2, 3, 4, 5 and 6 have the general format shown in Figure 9-1.

Table 9.2 Port Data Register Summary

Register Name	Mnemonic	Decimal	Hex	Location	RW
Port 0 data register	P0	0	00H	Page 8	RW
Port 1 data register	P1	1	01H	Page 8	RW
Port 2 data register	P2	2	02H	Page 8	RW
Port 3 data register	P3	3	03H	Page 8	RW
Port 4 data register	P4	4	04H	Page 8	RW
Port 5 data register	P5	5	05H	Page 8	RW
Port 6 data register	P6	6	06H	Page 8	RW

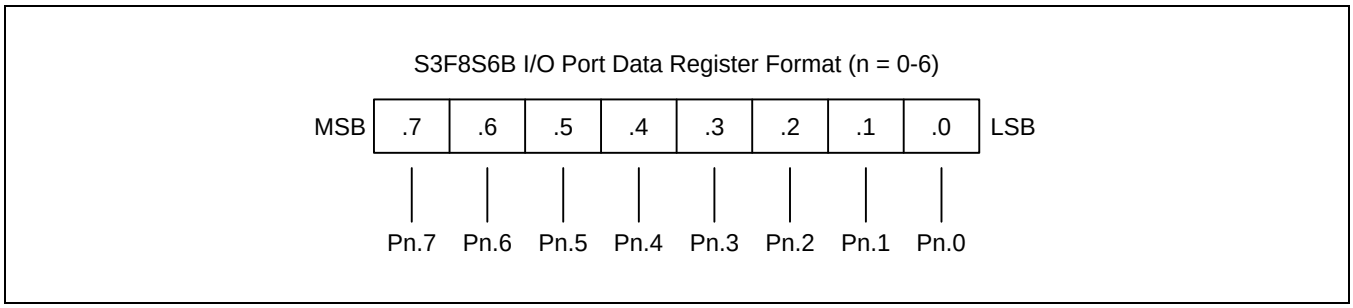


Figure 9-1 S3F8S6B I/O Port Data Register Format

9.2.1 Port 0

Port 0 is an 8-bit I/O port with individually configurable pins. Port 0 pins are accessed directly by writing or reading the port 0 data register, P0 at location 00H in page 8. P0.0–P0.7 can serve as inputs (with or without pull-ups), and push-pull outputs. And, they can serve as segment pins for LCD also.

9.2.1.1 Port 0 Control Register (P0CONH, P0CONL)

Port 0 has two 8-bit control registers: P0CONH for P0.4-P0.7 and P0CONL for P0.0-P0.3. A reset clears the P0CONH and P0CONL registers to "00H", configuring all pins to input mode. You use control registers settings to select input (with or without pull-ups) or push-pull output mode and enable the alternative functions.

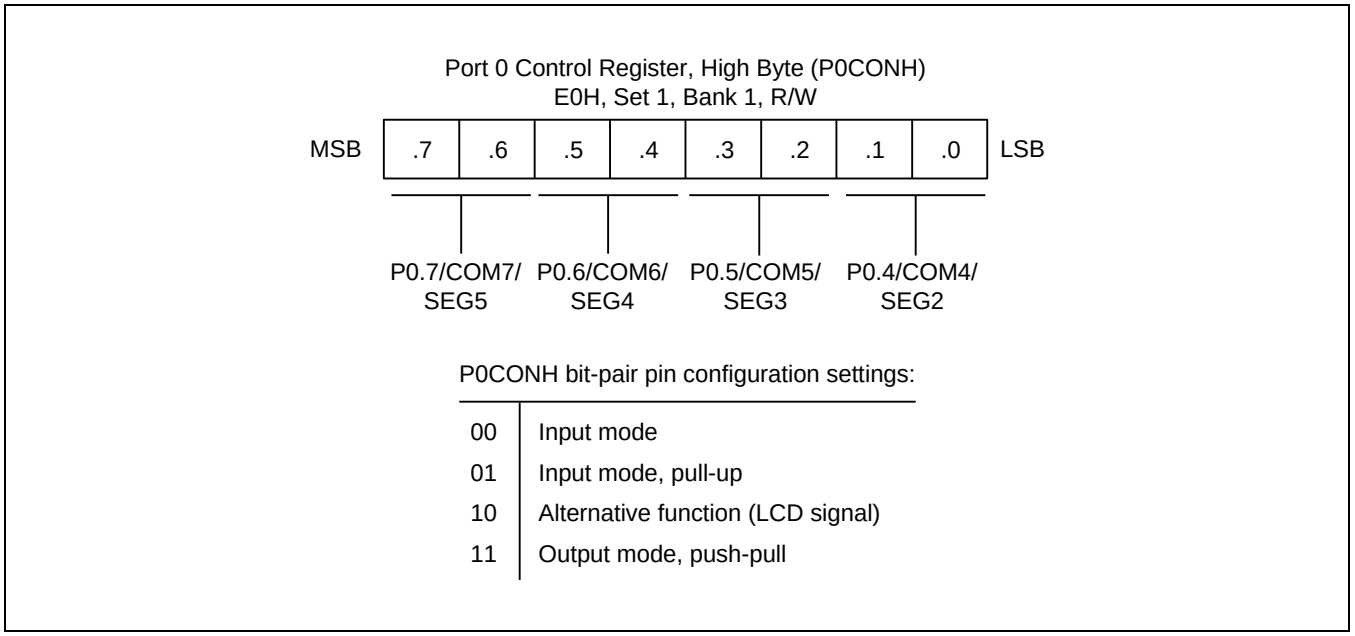


Figure 9-2 Port 0 High-Byte Control Register (P0CONH)

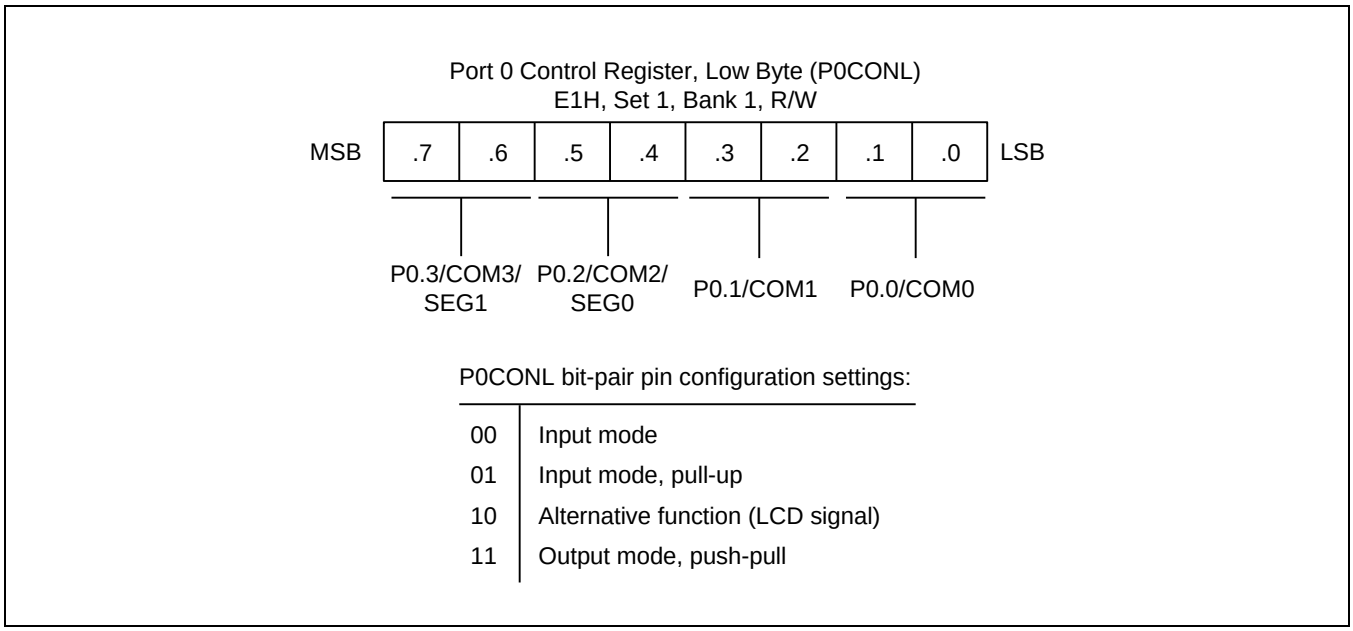


Figure 9-3 Port 0 Low-Byte Control Register (P0CONL)

9.2.2 Port 1

Port 1 is an 8-bit I/O port with individually configurable pins. Port 1 pins are accessed directly by writing or reading the port 1 data register, P1 at location 01H in page 8. P1.0–P1.7 can serve as inputs (with or without pull-ups), and outputs (push pull or open-drain). And the P1.7–P1.2 can serve as segment pins for LCD or you can configure the following alternative functions:

- Low-byte pins (P1.0-P1.3): RxD0, TxD0
- High-byte pins (P1.4-P1.7): RxD1, TxD1, TAOUT/TAPWM/TACAP, TACLK

9.2.2.1 Port 1 Control Register (P1CONH, P1CONL)

Port 1 has two 8-bit control registers: P1CONH for P1.4–P1.7 and P1CONL for P1.0–P1.3. A reset clears the P1CONH and P1CONL registers to "00H", configuring all pins to input mode. You use control registers settings to select input or output mode and enable the alternative functions.

When programming the port, please remember that any alternative peripheral I/O function you configure using the port 1 control registers must also be enabled in the associated peripheral module.

9.2.2.2 Port 1 Pull-Up Resistor Enable Register (P1PUR)

Using the port 1 pull-up resistor enable register, P1PUR (E4H, set1, bank1), you can configure pull-up resistors to individual port 1 pins.

9.2.2.3 Port 1 N-Channel Open-Drain Mode Register (PNE1)

Using the port 1 n-channel open-drain mode register, PNE1 (E5H, set1, bank1), you can configure push-pull or open-drain output mode to individual port 1 pins.

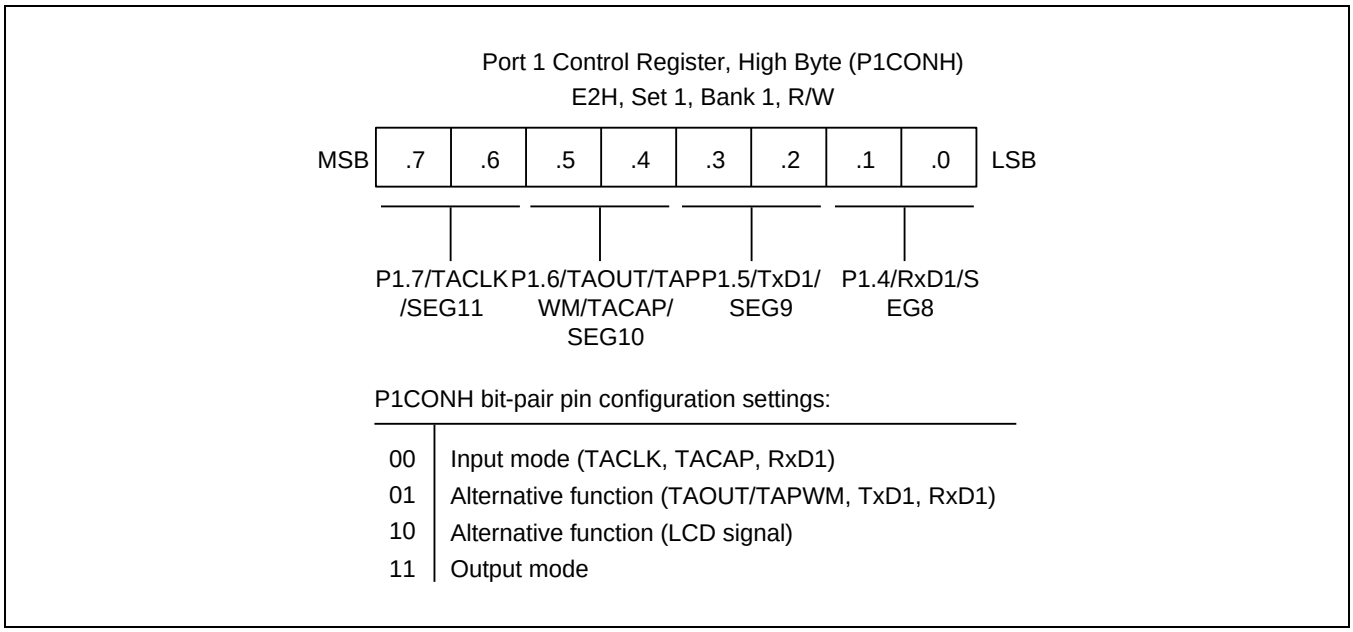


Figure 9-4 Port 1 High-Byte Control Register (P1CONH)

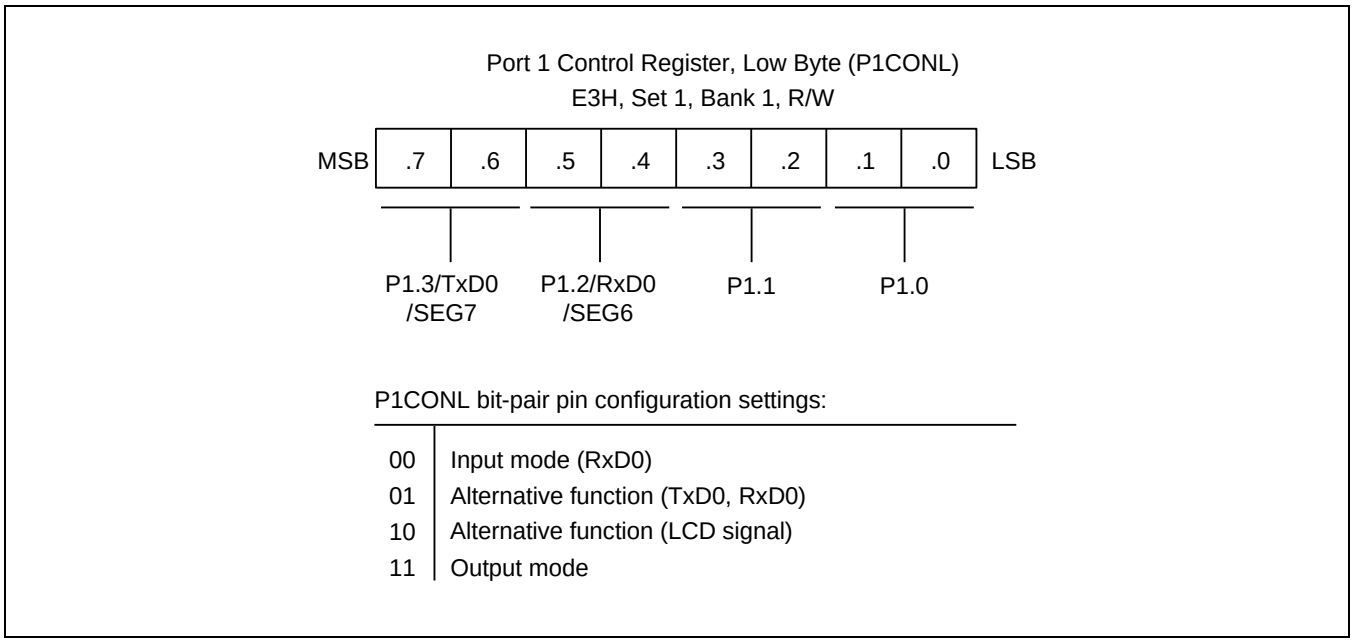
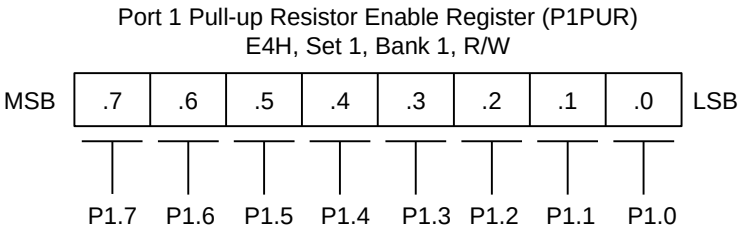


Figure 9-5 Port 1 Low-Byte Control Register (P1CONL)

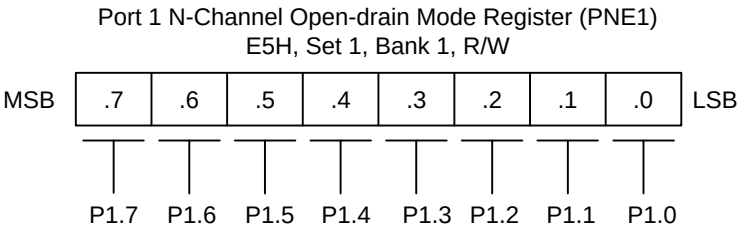


P1PUR bit configuration settings:

0	Disable Pull-up Resistor
1	Enable Pull-up Resistor

NOTE: A pull-up resistor of port 1 is automatically disabled only when the corresponding pin is selected as push-pull output or alternative function.

Figure 9-6 Port 1 Pull-up Resistor Enable Register (P1PUR)



PNE1 bit configuration settings:

0	Push-pull output mode
1	Open-drain output mode

Figure 9-7 Port 1 N-Channel Open-drain Mode Register (PNE1)

9.2.3 Port 2

Port 2 is an 8-bit I/O port with individually configurable pins. Port 2 pins are accessed directly by writing or reading the port 2 data register, P2 at location 02H in page 8. P2.0–P2.7 can serve as inputs (with or without pull-ups), and push pull outputs. And the P2.7–P2.0 can serve as segment pins for LCD or you can configure the following alternative functions:

- Low-byte pins (P2.0–P2.3): INT0-INT3
- High-byte pins (P2.4–P2.7): INT4-INT7

9.2.3.1 Port 2 Control Register (P2CONH, P2CONL)

Port 2 has two 8-bit control registers: P2CONH for P2.4-P2.7 and P2CONL for P2.0-P2.3. A reset clears the P2CONH and P2CONL registers to "00H", configuring all pins to input mode. In input mode, three different selections are available:

- Schmitt trigger input with interrupt generation on falling signal edges.
- Schmitt trigger input with interrupt generation on rising signal edges.
- Schmitt trigger input with interrupt generation on falling/rising signal edges.

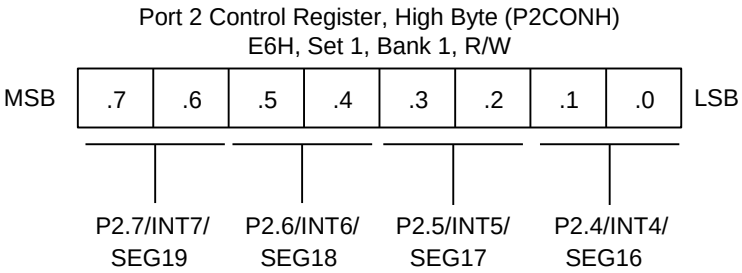
When programming the port, please remember that any alternative peripheral I/O function you configure using the port 2 control registers must also be enabled in the associated peripheral module.

9.2.3.2 Port 2 Interrupt Enable and Pending Registers (P2INTH, P2INTL, P2PND)

To process external interrupts at the port 2 pins, the additional control registers are provided: the port 2 interrupt enable register P2INTH (high byte, E8H, set 1, bank 1), P2INTL (Low byte, E9H, set1, bank1) and the port 2 interrupt pending register P2PND (EAH, set 1, bank 1).

The port 2 interrupt pending register P2PND lets you check for interrupt pending conditions and clear the pending condition when the interrupt service routine has been initiated. The application program detects interrupt requests by polling the P2PND register at regular intervals.

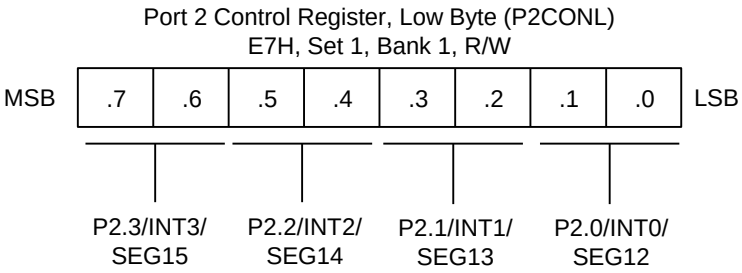
When the interrupt enable bit of any port 2 pin is "1", a rising or falling signal edge at that pin will generate an interrupt request. The corresponding P2PND bit is then automatically set to "1" and the IRQ level goes low to signal the CPU that an interrupt request is waiting. When the CPU acknowledges the interrupt request, application software must the clear the pending condition by writing a "0" to the corresponding P2PND bit.



P2CONH bit-pair pin configuration settings:

00	Schmitt trigger input mode
01	Schmitt trigger input mode, pull-up
10	Alternative function (LCD signal)
11	Push-pull output mode

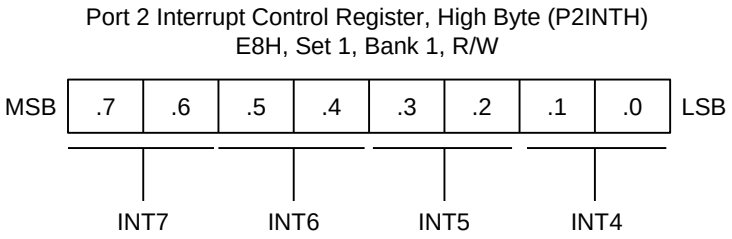
Figure 9-8 Port 2 High-Byte Control Register (P2CONH)



P2CONL bit-pair pin configuration settings:

00	Schmitt trigger input mode
01	Schmitt trigger input mode, pull-up
10	Alternative function (LCD signal)
11	Output mode, push-pull

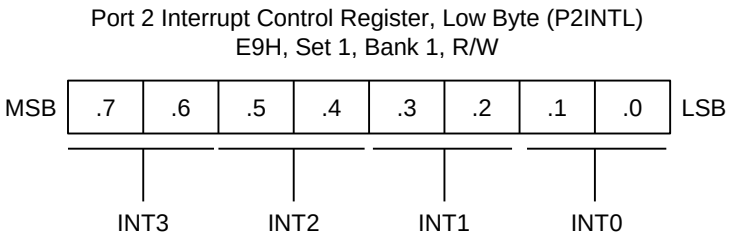
Figure 9-9 Port 2 Low-Byte Control Register (P2CONL)



P2INTH bit-pair pin configuration settings:

00	Disable interrupt
01	Enable interrupt by falling edge
10	Enable interrupt by rising edge
11	Enable interrupt by both falling and rising edge

Figure 9-10 Port 2 High-Byte Interrupt Control Register (P2INTH)



P2INTL bit-pair pin configuration settings:

00	Disable interrupt
01	Enable interrupt by falling edge
10	Enable interrupt by rising edge
11	Enable interrupt by both falling and rising edge

Figure 9-11 Port 2 Low-Byte Interrupt Control Register (P2INTL)

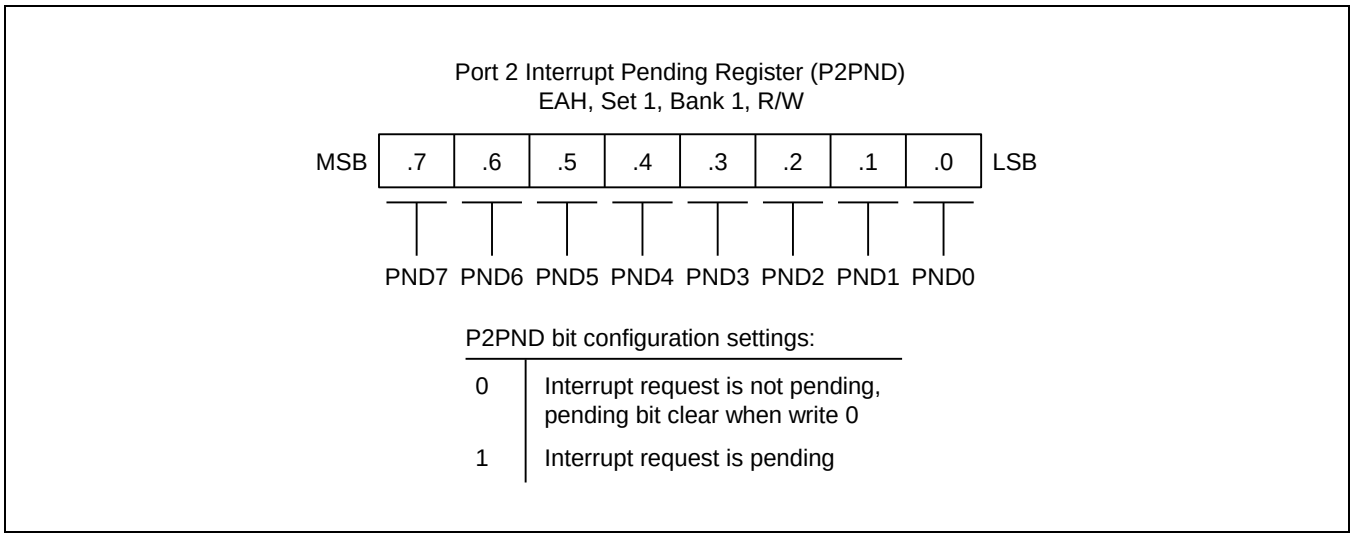


Figure 9-12 Port 2 Interrupt Pending Register (P2PND)

9.2.4 Port 3

Port 3 is an 8-bit I/O port with individually configurable pins. Port 3 pins are accessed directly by writing or reading the port 3 data register, P3 at location 03H in page 8. P3.0–P3.7 can serve as inputs (with or without pull-ups), and outputs (push pull or open-drain). And the P3.7–P3.0 can serve as segment pins for LCD or you can configure the following alternative functions:

- Low-byte pins (P3.0–P3.1): TBPWM/PG0, TCOUT/PG1
- Middle-byte pins (P3.2–P3.4): PG2, TD0OUT/TD0PWM/TD0CAP/PG3, TD0CLK/PG4
- High-byte pins (P3.5–P3.7): TD1OUT/TD1PWM/TD1CAP/PG5, TD1CLK/PG6, BUZ/PG7

9.2.4.1 Port 3 Control Registers (P3CONH, P3CONM P3CONL)

Port 3 has three 8-bit control registers: P3CONH for P3.5–P3.7, P3CONM for P3.2–P3.4 and P3CONL for P3.0–P3.1. A reset clears the P3CONH, P3CONM and P3CONL registers to "00H", configuring all pins to input mode. You use control registers settings to select input or output mode, and enable the alternative functions.

When programming the port, please remember that any alternative peripheral I/O function you configure using the port 3 control registers must also be enabled in the associated peripheral module.

9.2.4.2 Port 3 Pull-Up Resistor Enable Register (P3PUR)

Using the port 3 pull-up resistor enable register, P3PUR (EEH, set1, bank1), you can configure pull-up resistors to individual port 3 pins.

9.2.4.3 Port 3 N-Channel Open-Drain Mode Register (PNE3)

Using the port 3 n-channel open-drain mode register, PNE3 (EFH, set1, bank1), you can configure push-pull or open-drain output mode to individual port 3 pins.

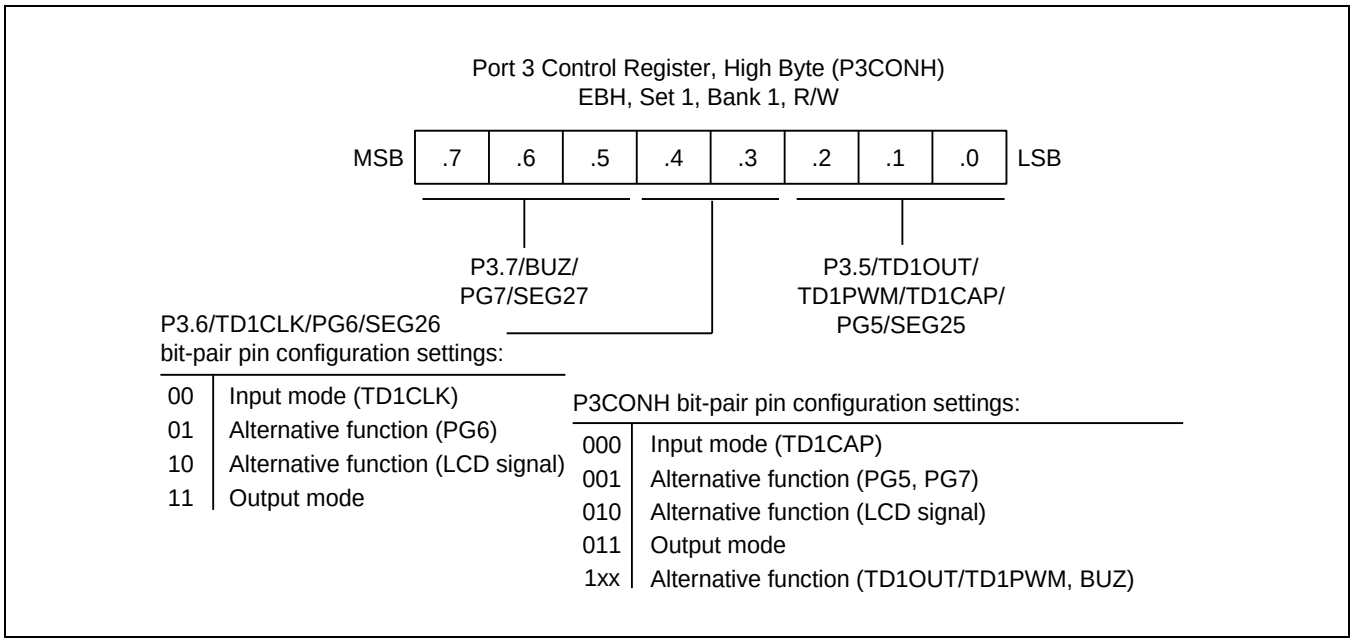


Figure 9-13 Port 3 High-Byte Control Register (P3CONH)

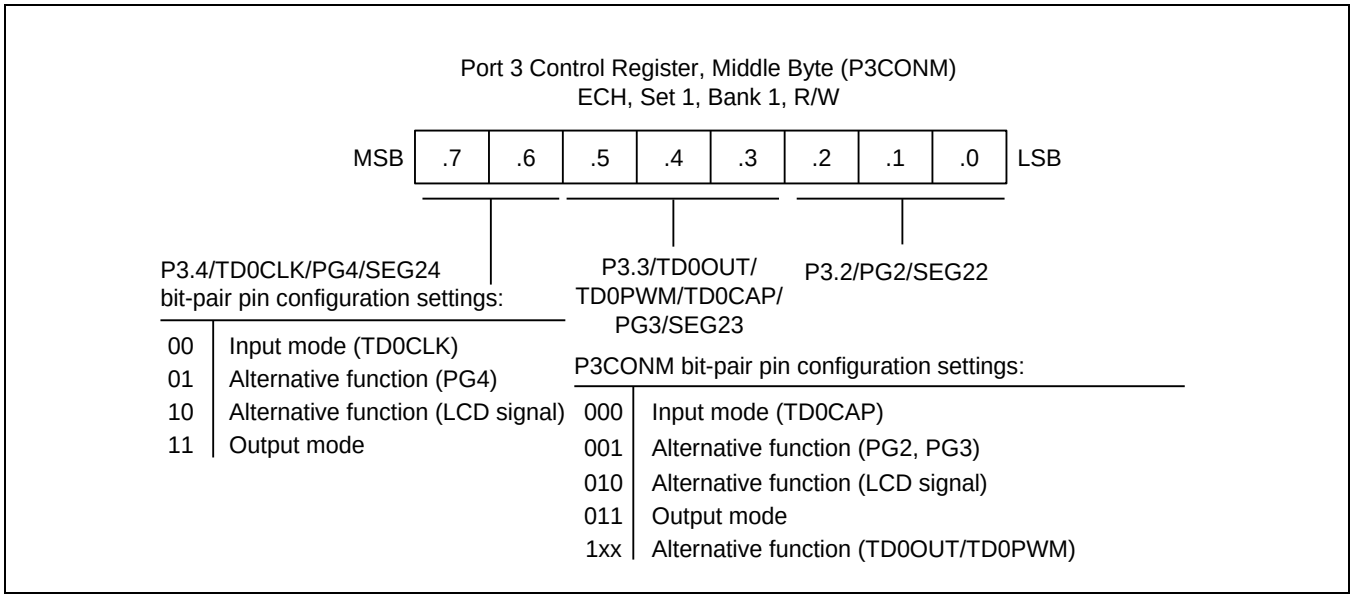
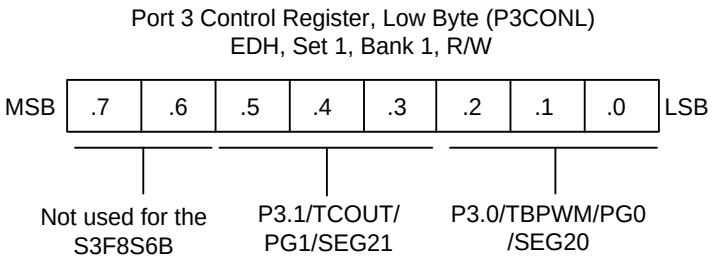


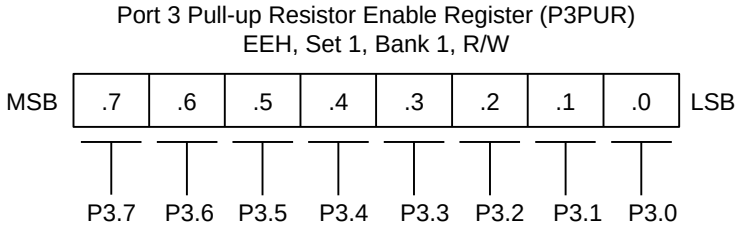
Figure 9-14 Port 3 Middle-Byte Control Register (P3CONM)



P3CONL bit-pair pin configuration settings:

000	Input mode
001	Alternative function (PG0, PG1)
010	Alternative function (LCD signal)
011	Output mode
1xx	Alternative function (TBPWM, TCOUT)

Figure 9-15 Port 3 Low-Byte Control Register (P3CONL)



P3PUR bit configuration settings:

0	Disable Pull-up Resistor
1	Enable Pull-up Resistor

NOTE: A pull-up resistor of port 3 is automatically disabled only when the corresponding pin is selected as push-pull output or alternative function.

Figure 9-16 Port 3 Pull-Up Resistor Enable Register (P3PUR)

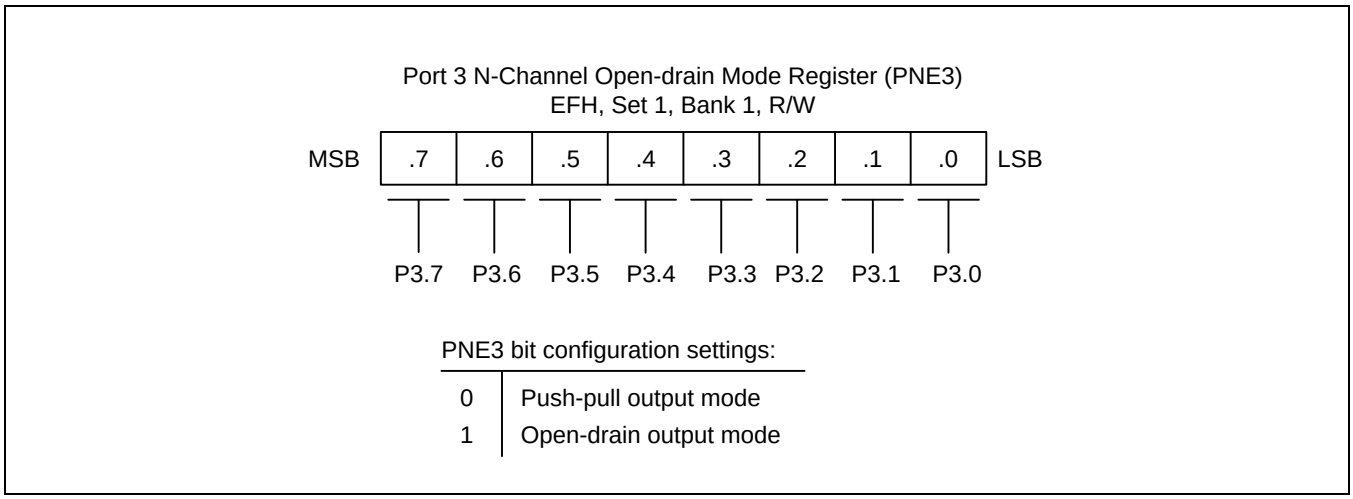


Figure 9-17 Port 3 N-Channel Open-Drain Mode Register (PNE3)

9.2.5 Port 4

Port 4 is an 8-bit I/O port that can be used for general purpose I/O as A/D converter inputs, AD0-AD7. Port 4 pins are accessed directly by writing or reading the port 4 data register, P4 at location 04H in page 8. P4.0–P4.7 can serve as inputs (with or without pull-ups), and push-pull outputs or you can configure the following alternative functions:

- Low-byte pins (P4.0–P4.3): INT8-INT11/AD0-AD3
- High-byte pins (P4.4–P4.7): INT12-INT15/AD4-AD7

9.2.5.1 Port 4 Control Register (P4CONH, P4CONL)

Port 4 has two 8-bit control registers: P4CONH for P4.4-P4.7 and P4CONL for P4.0-P4.3. A reset clears the P4CONH and P4CONL registers to "00H", configuring all pins to input mode. In input mode, three different selections are available:

- Schmitt trigger input with interrupt generation on falling signal edges.
- Schmitt trigger input with interrupt generation on rising signal edges.
- Schmitt trigger input with interrupt generation on falling/rising signal edges.

When programming the port, please remember that any alternative peripheral I/O function you configure using the port 4 control registers must also be enabled in the associated peripheral module.

9.2.5.2 Port 4 Interrupt Enable and Pending Registers (P4INTH, P4INTL, P4PND)

To process external interrupts at the port 4 pins, the additional control registers are provided: the port 4 interrupt enable register P4INTH (high byte, 0EH, page 8), P4INTL (Low byte, 0FH, page 8) and the port 4 interrupt pending register P4PND (10H, page 8).

The port 4 interrupt pending register P4PND lets you check for interrupt pending conditions and clear the pending condition when the interrupt service routine has been initiated. The application program detects interrupt requests by polling the P4PND register at regular intervals.

When the interrupt enable bit of any port 4 pin is "1", a rising or falling signal edge at that pin will generate an interrupt request. The corresponding P4PND bit is then automatically set to "1" and the IRQ level goes low to signal the CPU that an interrupt request is waiting. When the CPU acknowledges the interrupt request, application software must the clear the pending condition by writing a "0" to the corresponding P4PND bit.

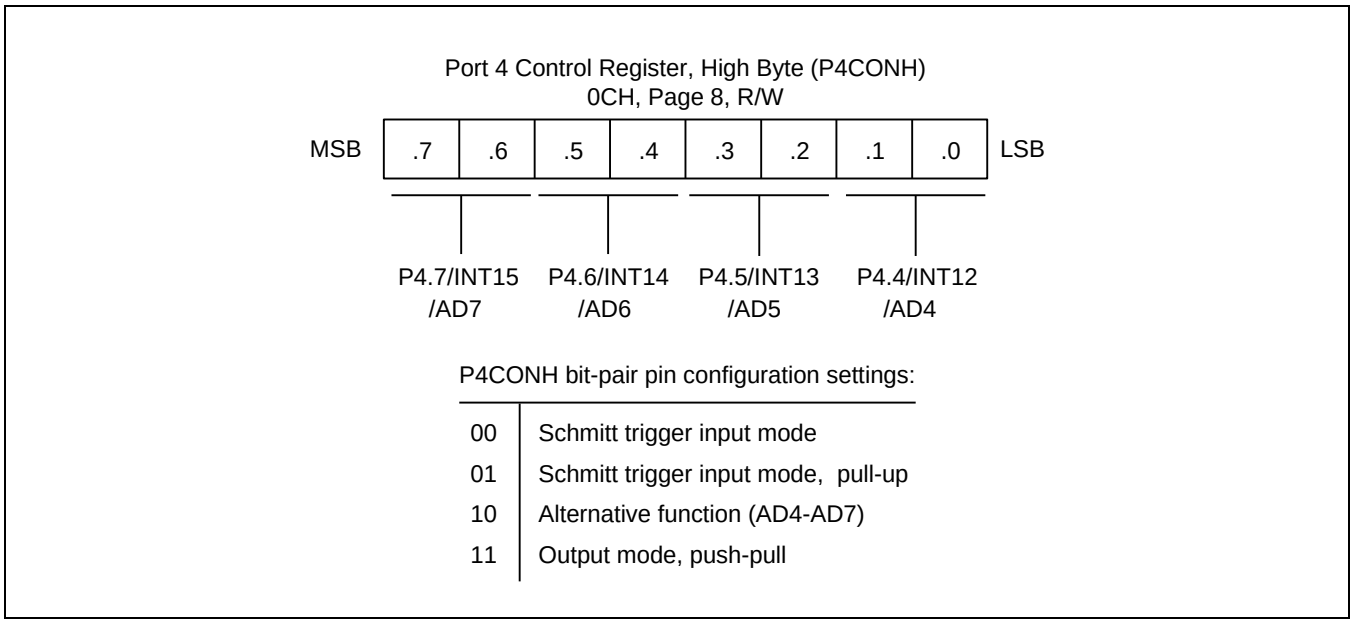


Figure 9-18 Port 4 High-Byte Control Register (P4CONH)

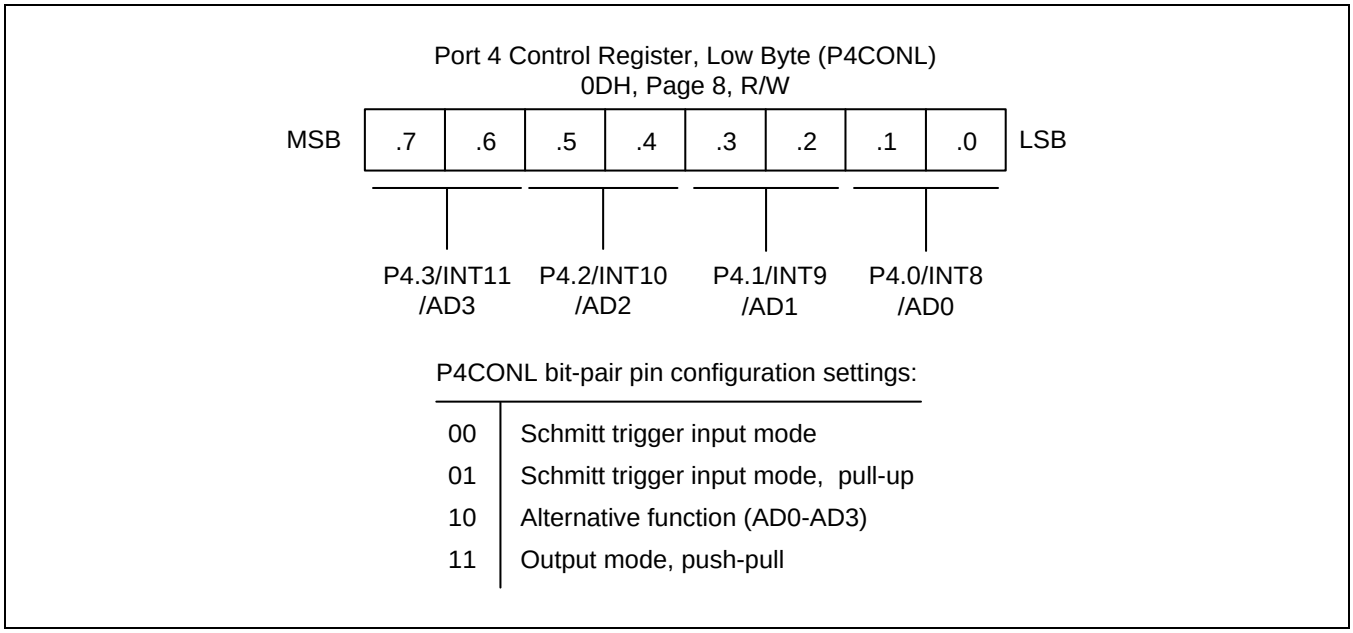
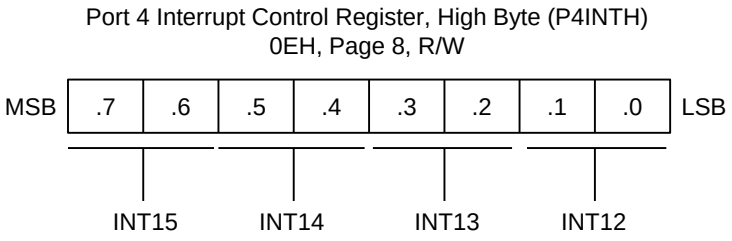


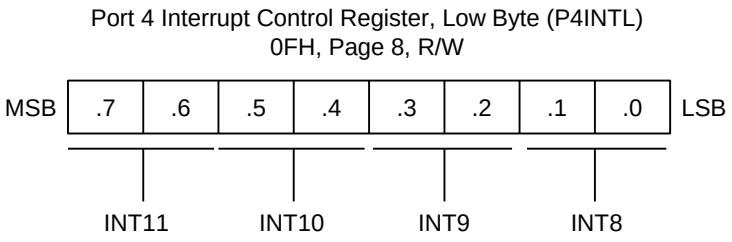
Figure 9-19 Port 4 Low-Byte Control Register (P4CONL)



P4INTH bit-pair pin configuration settings:

00	Disable interrupt
01	Enable interrupt by falling edge
10	Enable interrupt by rising edge
11	Enable interrupt by both falling and rising edge

Figure 9-20 Port 4 High-Byte Interrupt Control Register (P4INTH)



P4INTL bit-pair pin configuration settings:

00	Disable interrupt
01	Enable interrupt by falling edge
10	Enable interrupt by rising edge
11	Enable interrupt by both falling and rising edge

Figure 9-21 Port 4 Low-Byte Interrupt Control Register (P4INTL)

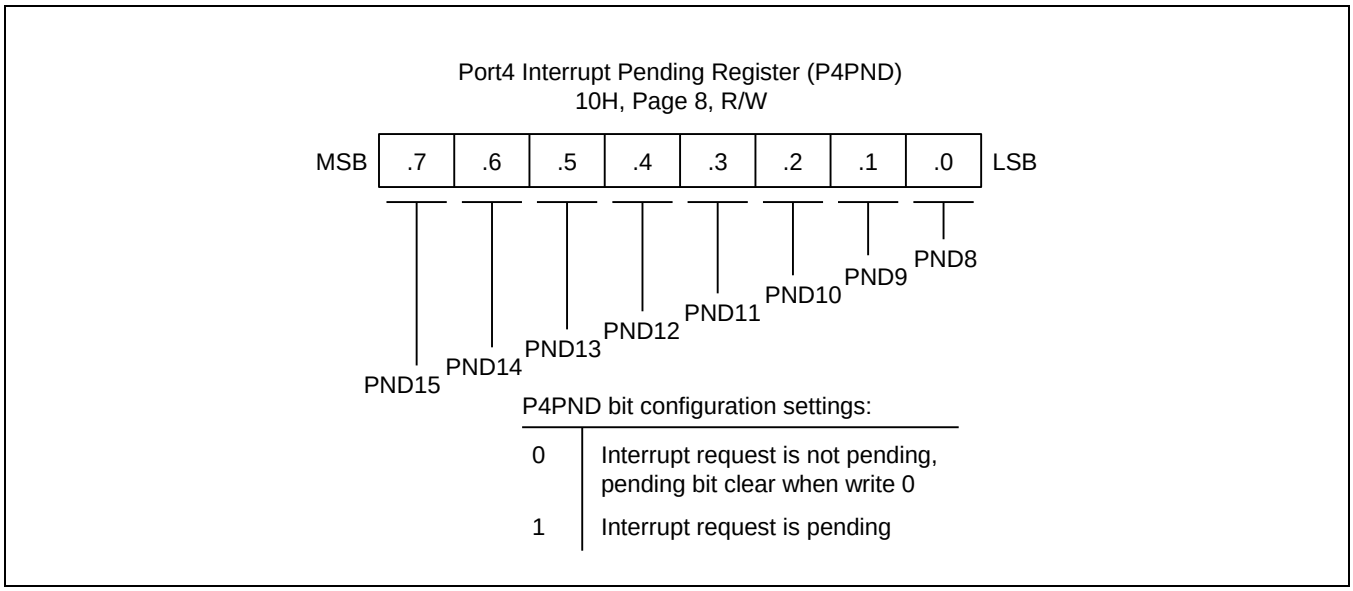


Figure 9-22 Port 4 Interrupt Pending Register (P4PND)

9.2.6 Port 5

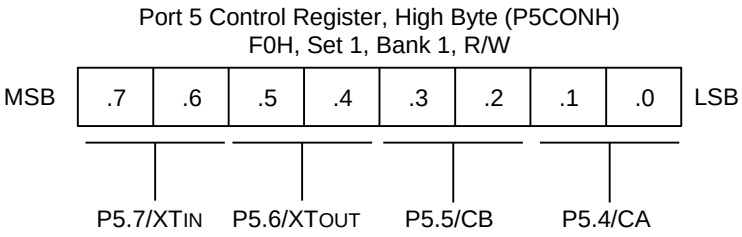
Port 5 is an 8-bit I/O port with individually configurable pins. Port 5 pins are accessed directly by writing or reading the port 5 data register, P5 at location 05H in page 8. P5.0–P5.7 can serve as inputs (with or without pull-ups) and push-pull outputs or you can configure the following alternative functions:

- Low-byte pins (P5.0–P5.3): V_{LC0}–V_{LC3}
- High-byte pins (P5.4–P5.7): CA, CB, XTOUT, XTIN

9.2.6.1 Port 5 Control Registers (P5CONH, P5CONL)

Port 5 has two 8-bit control registers: P5CONH for P5.4–P5.7 and P5CONL for P5.0–P5.3. A reset clears the P5CONH and P5CONL registers to "00H", configuring all pins to input mode. You use control registers settings to select input (with or without pull-ups) or push-pull output mode and enable the alternative functions.

When programming the port, please remember that any alternative peripheral I/O function you configure using the port 5 control registers must also be enabled in the associated peripheral module.

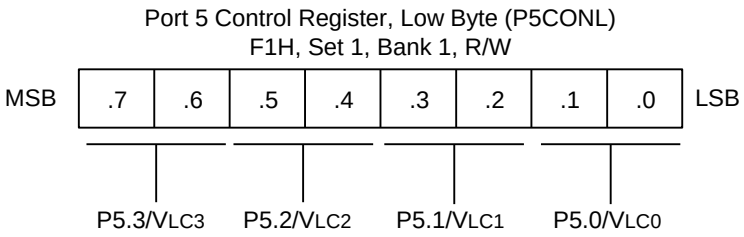


P5CONH bit-pair pin configuration settings:

00	Input mode
01	Input mode, pull-up
10	Alternative function (XTOUT, XTIN)
11	Output mode, push-pull

NOTE: Refer to LCON register in Chapter 15.

Figure 9-23 Port 5 High-Byte Control Register (P5CONH)



P5CONL bit-pair pin configuration settings:

00	Input mode
01	Input mode, pull-up
10	Not available
11	Output mode, push-pull

NOTE: Refer to LCON register in Chapter 15.

Figure 9-24 Port 5 Low-Byte Control Register (P5CONL)

9.2.7 Port 6

Port 6 is an 8-bit I/O port with individually configurable pins. Port 6 pins are accessed directly by writing or reading the port 6 data register, P6 at location 06H in page 8. P6.0–P6.5 can serve as inputs (with or without pull-ups), and outputs (push pull or open-drain). And the P6.5–P6.0 can serve as segment pins for LCD or you can configure the following alternative functions:

- Low-byte pins (P6.0-P6.3): SCK, SI, SO

9.2.7.1 Port 6 Control Register (P6CONH, P6CONL)

Port 6 has two 8-bit control registers: P6CONH for P6.4–P6.5 and P6CONL for P6.0–P6.3. A reset clears the P6CONH and P6CONL registers to "00H", configuring all pins to input mode. You use control registers settings to select input or output mode and enable the alternative functions.

When programming the port, please remember that any alternative peripheral I/O function you configure using the port 6 control registers must also be enabled in the associated peripheral module.

9.2.7.2 Port 6 Pull-Up Resistor Enable Register (P6PUR)

Using the port 6 pull-up resistor enable register, P6PUR (F4H, set1, bank1), you can configure pull-up resistors to individual port 6 pins.

9.2.7.3 Port 6 N-Channel Open-Drain Mode Register (PNE6)

Using the port 6 n-channel open-drain mode register, PNE6 (F5H, set1, bank1), you can configure push-pull or open-drain output mode to individual port 6 pins.

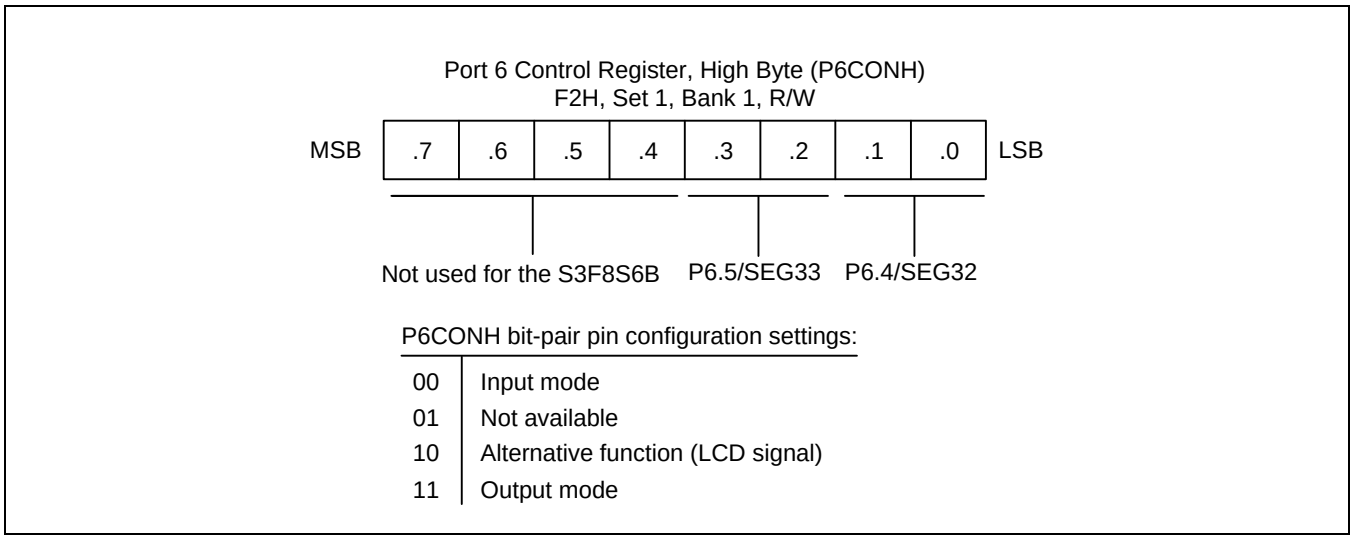


Figure 9-25 Port 6 High-Byte Control Register (P6CONH)

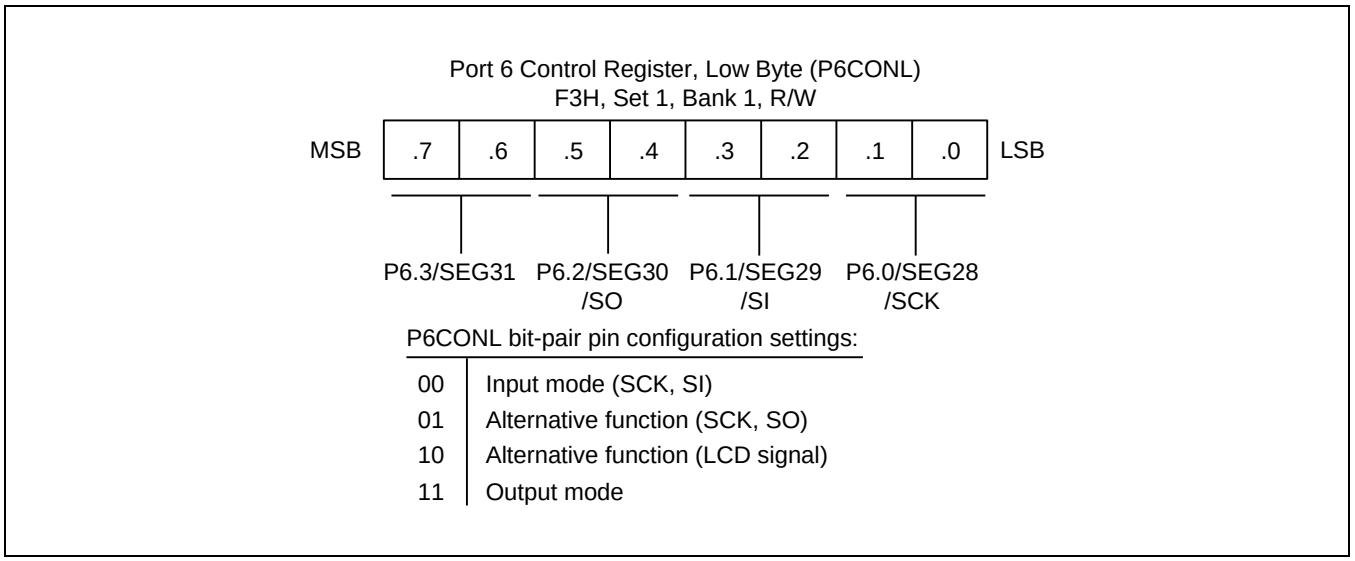
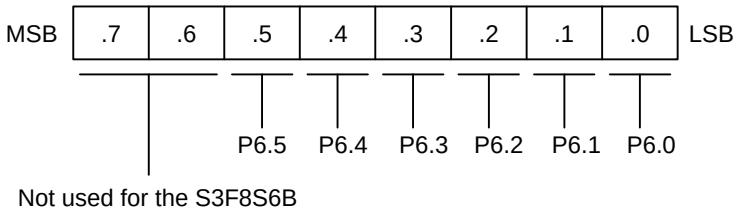


Figure 9-26 Port 6 Low-Byte Control Register (P6CONL)

Port 6 Pull-up Resistor Enable Register (P6PUR)
F4H, Set 1, Bank 1, R/W



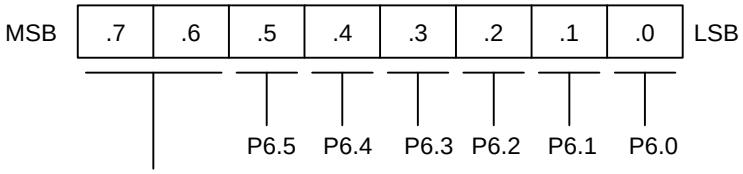
P6PUR bit configuration settings:

0	Disable Pull-up Resistor
1	Enable Pull-up Resistor

NOTE: A pull-up resistor of port 6 is automatically disabled only when the corresponding pin is selected as push-pull output or alternative function.

Figure 9-27 Port 6 Pull-up Resistor Enable Register (P6PUR)

Port 6 N-Channel Open-drain Mode Register (PNE6)
F5H, Set 1, Bank 1, R/W



PNE6 bit configuration settings:

0	Push-pull output mode
1	Open-drainl output mode

Figure 9-28 Port 6 N-Channel Open-drain Mode Register (PNE6)

10 Basic Timer

10.1 Overview

S3F8S6B has an 8-bit basic timer.

10.1.1 Basic Timer (BT)

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

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

The functional components of the basic timer block are:

- Clock frequency divider (fxx divided by 4096, 1024, 128, or 16) with multiplexer
- 8-bit basic timer counter, BTCNT (set 1, Bank 0, FDH, read-only)
- Basic timer control register, BTCON (set 1, D3H, read/write)

10.2 Basic Timer Control Register (BTCON)

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

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

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

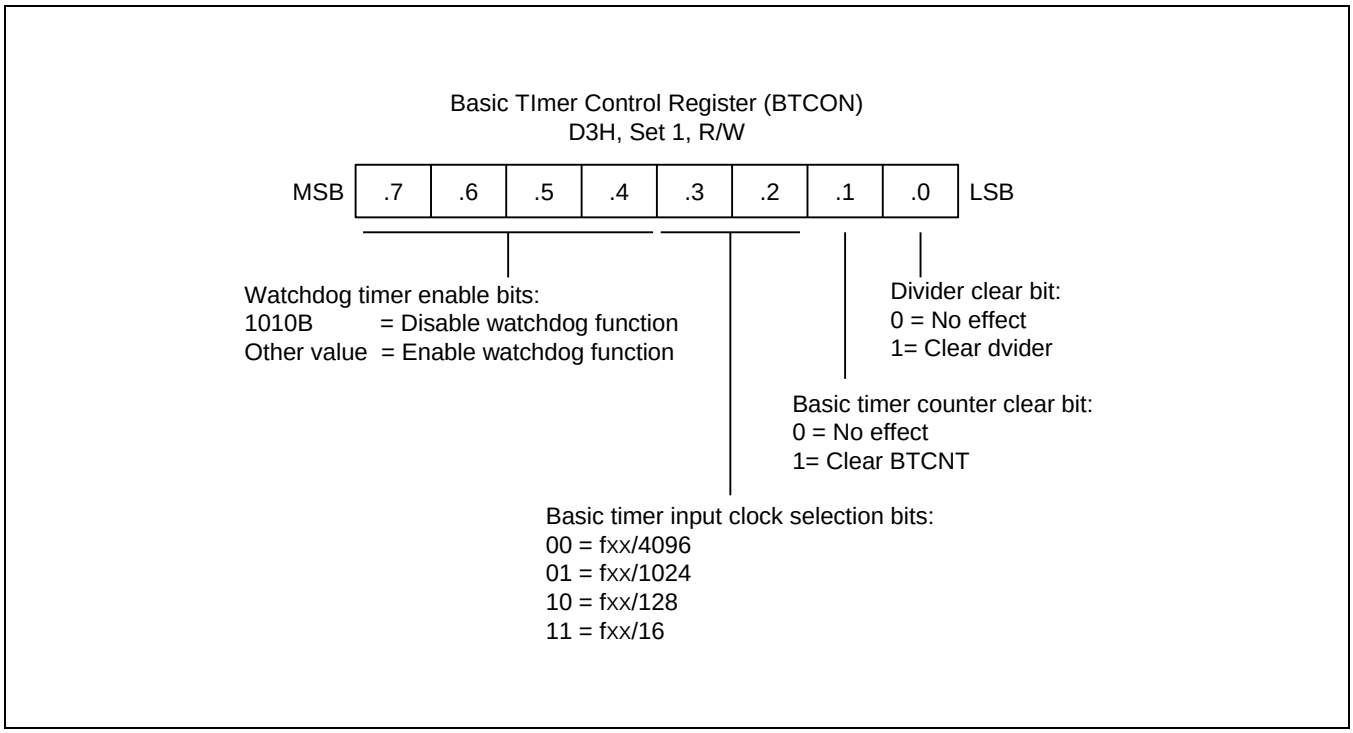


Figure 10-1 Basic Timer Control Register (BTCON)

10.3 Basic Timer Function Description

10.3.1 Watchdog Timer Function

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

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

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

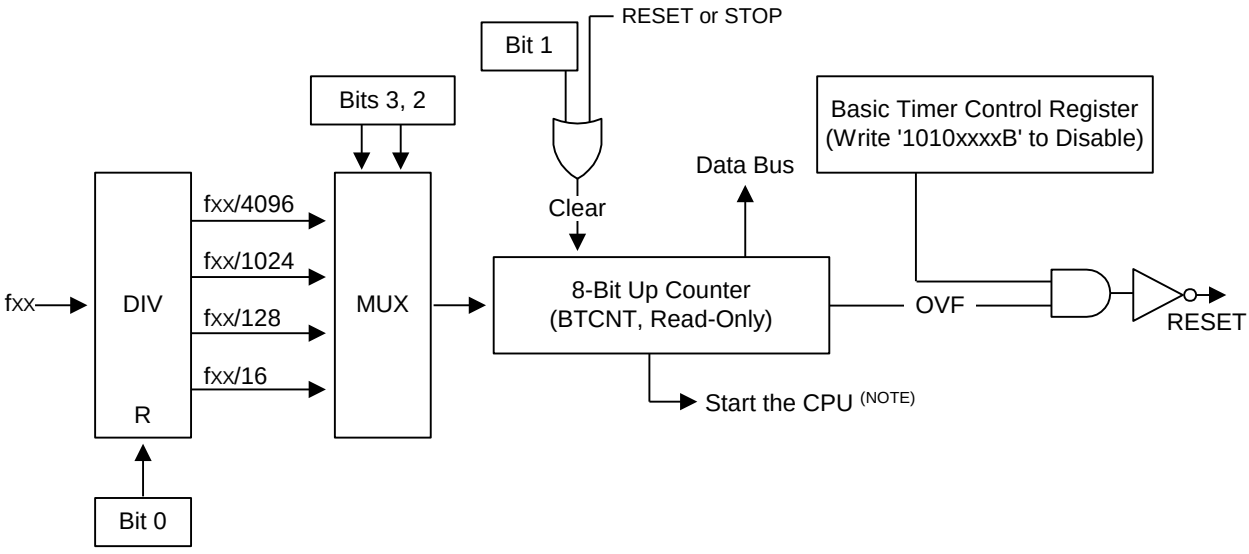
10.3.2 Oscillation Stabilization Interval Timer Function

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

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

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

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



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

Figure 10-2 Basic Timer Block Diagram

11 8-Bit Timer A/B

11.1 8-Bit Timer A

11.1.1 Overview

The 8-bit timer A is an 8-bit general-purpose timer/counter. Timer A has three operating modes, one of which you select using the appropriate TACON setting:

- Interval timer mode (Toggle output at TAOUT pin)
- Capture input mode with a rising or falling edge trigger at the TACAP pin
- PWM mode (TAPWM)

Timer A has the following functional components:

- Clock frequency divider (fxx divided by 1024, 256, 64, 8 or 1) with multiplexer
- External clock input pin (TACLK)
- 8-bit counter (TACNT), 8-bit comparator, and 8-bit reference data register (TADATA)
- I/O pins for capture input (TACAP) or PWM or match output (TAPWM, TAOUT)
- Timer A overflow interrupt (IRQ0 vector D0H) and match/capture interrupt (IRQ0 vector CEH) generation
- Timer A control register, TACON (set 1, Bank 0, E2H, read/write)

11.1.2 Timer A Control Register (TACON)

You use the timer A control register, TACON, to

- Select the timer A operating mode (interval timer, capture mode, or PWM mode)
- Select the timer A input clock frequency
- Clear the timer A counter, TACNT
- Enable the timer A overflow interrupt or timer A match/capture interrupt

TACON is located in set 1, Bank 0 at address E2H, and is read/write addressable using Register addressing mode.

A reset clears TACON to '00H'. This sets timer A to normal interval timer mode, selects an input clock frequency of fxx/1024, and disables all timer A interrupts. You can clear the timer A counter at any time during normal operation by writing a "1" to TACON.2.

The timer A overflow interrupt (TAOVF) is interrupt level IRQ0 and has the vector address D0H. When a timer A overflow interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware or must be cleared by software.

To enable the timer A match/capture interrupt (IRQ0, vector CEH), you must write TACON.1 to "1". To detect a match/capture interrupt pending condition, the application program polls INTPND.1. When a "1" is detected, a timer A match or capture interrupt is pending. When the interrupt request has been serviced, the pending condition must be cleared by software by writing a "0" to the timer A match/capture interrupt pending bit, INTPND.1.

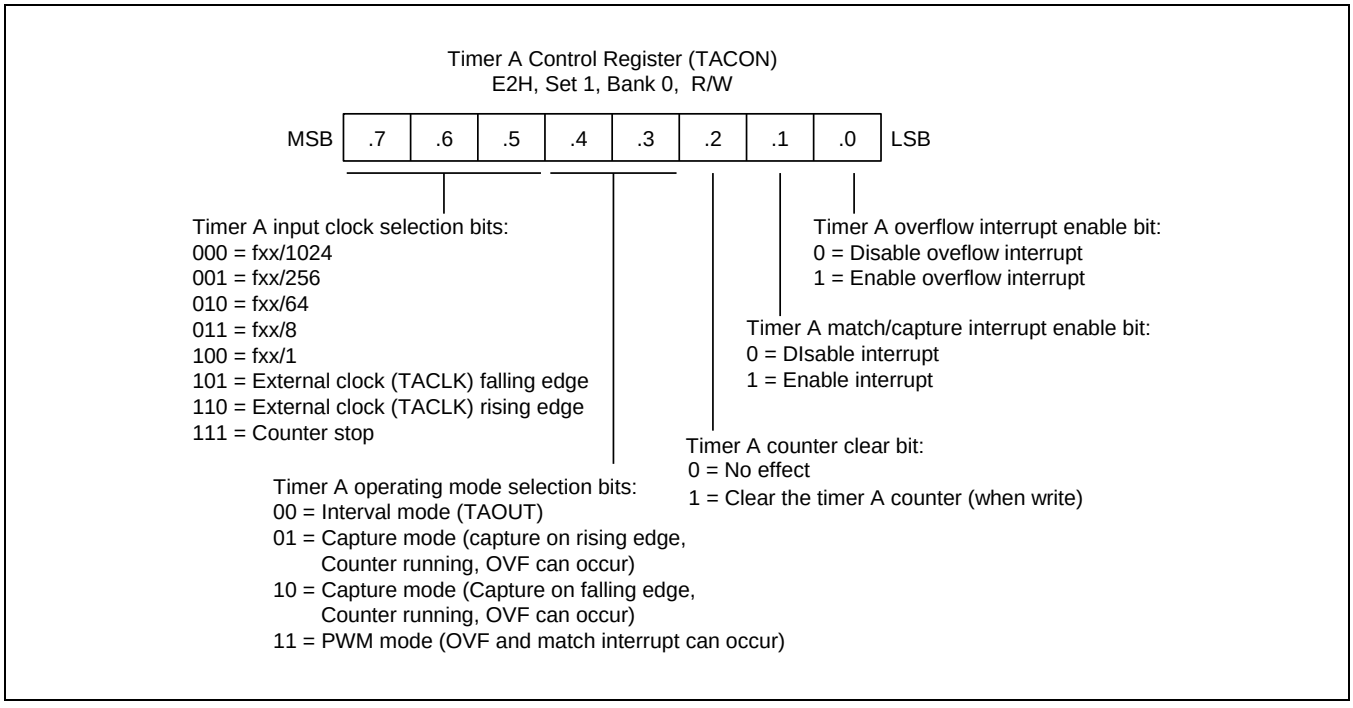


Figure 11-1 Timer A Control Register (TACON)

11.1.3 Timer A Function Description

11.1.3.1 Timer A Interrupts (IRQ0, Vectors CEH and D0H)

The timer A can generate two interrupts: the timer A overflow interrupt (TAOVF), and the timer A match/capture interrupt (TAINT). TAOVF is interrupt level IRQ0, vector D0H. TAINT also belongs to interrupt level IRQ0, but is assigned the separate vector address, CEH.

A timer A overflow interrupt pending condition is automatically cleared by hardware when it has been serviced or should be cleared by software in the interrupt service routine by writing a "0" to the INTPND.0 interrupt pending bit. However, the timer A match/capture interrupt pending condition must be cleared by the application's interrupt service routine by writing a "0" to the INTPND.1 interrupt pending bit.

11.1.3.2 Interval Timer Mode

In interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer A reference data register, TADATA. The match signal generates a timer A match interrupt (TAINT, vector CEH) and clears the counter.

If, for example, you write the value "10H" to TADATA, the counter will increment until it reaches "10H". At this point, the timer A interrupt request is generated, the counter value is reset, and counting resumes. With each match, the level of the signal at the timer A output pin is inverted (see *Figure 11-2*).

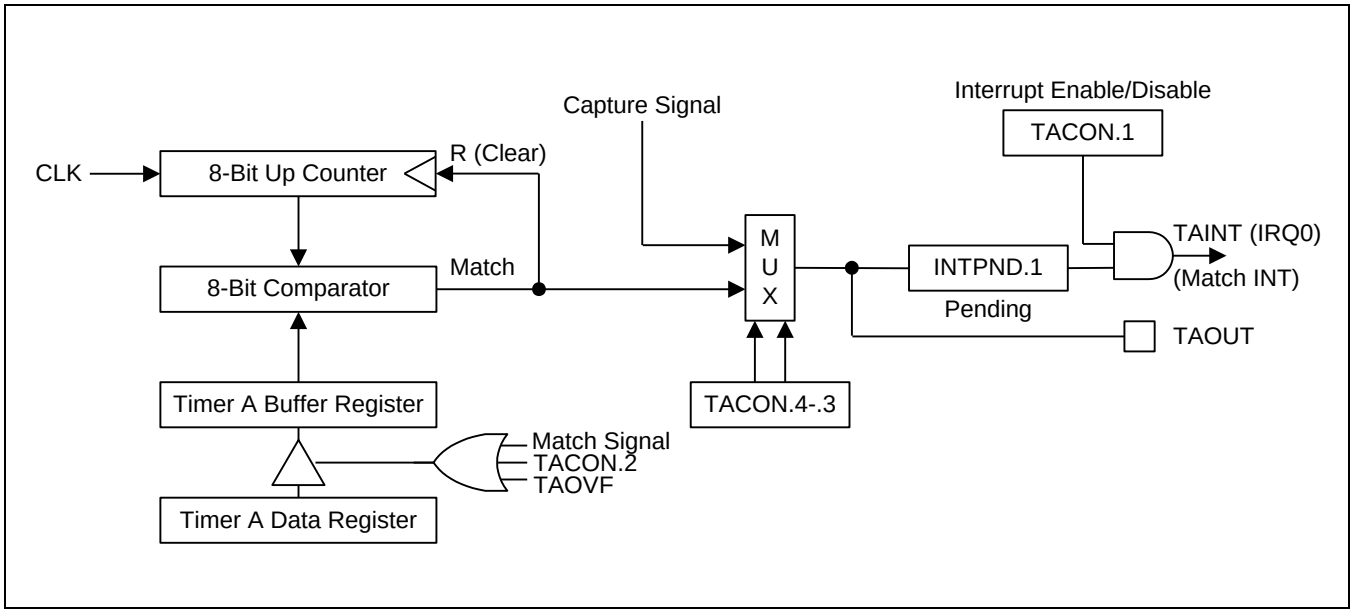


Figure 11-2 Simplified Timer A Function Diagram: Interval Timer Mode

11.1.3.3 Pulse Width Modulation Mode

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

Although you can use the match signal to generate a timer A overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the TAPWM pin is held to Low level as long as the reference data value is less than or equal to (≤) the counter value and then the pulse is held to High level for as long as the data value is greater than (>) the counter value. One pulse width is equal to $t_{CLK} \times 256$ (see *Figure 11-3*).

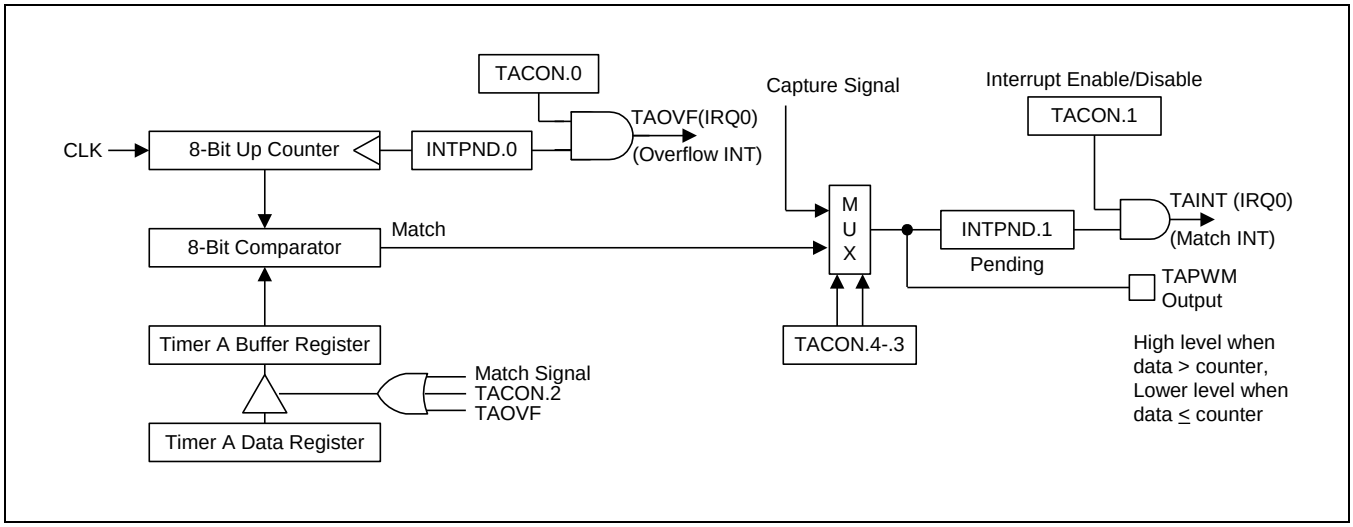


Figure 11-3 Simplified Timer A Function Diagram: PWM Mode

11.1.3.4 Capture Mode

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

Timer A also gives you capture input source: the signal edge at the TACAP pin. You select the capture input by setting the values of the timer A capture input selection bits in the port 1 control register, P1CONH.5–.4, (set 1, bank 1, E2H). When P1CONH.5–.4 is "00" the TACAP input is selected.

Both kinds of timer A interrupts can be used in capture mode: the timer A overflow interrupt is generated whenever a counter overflow occurs; the timer A match/capture interrupt is generated whenever the counter value is loaded into the timer A data register.

By reading the captured data value in TADATA, and assuming a specific value for the timer A clock frequency, you can calculate the pulse width (duration) of the signal that is being input at the TACAP pin (see *Figure 11-4*).

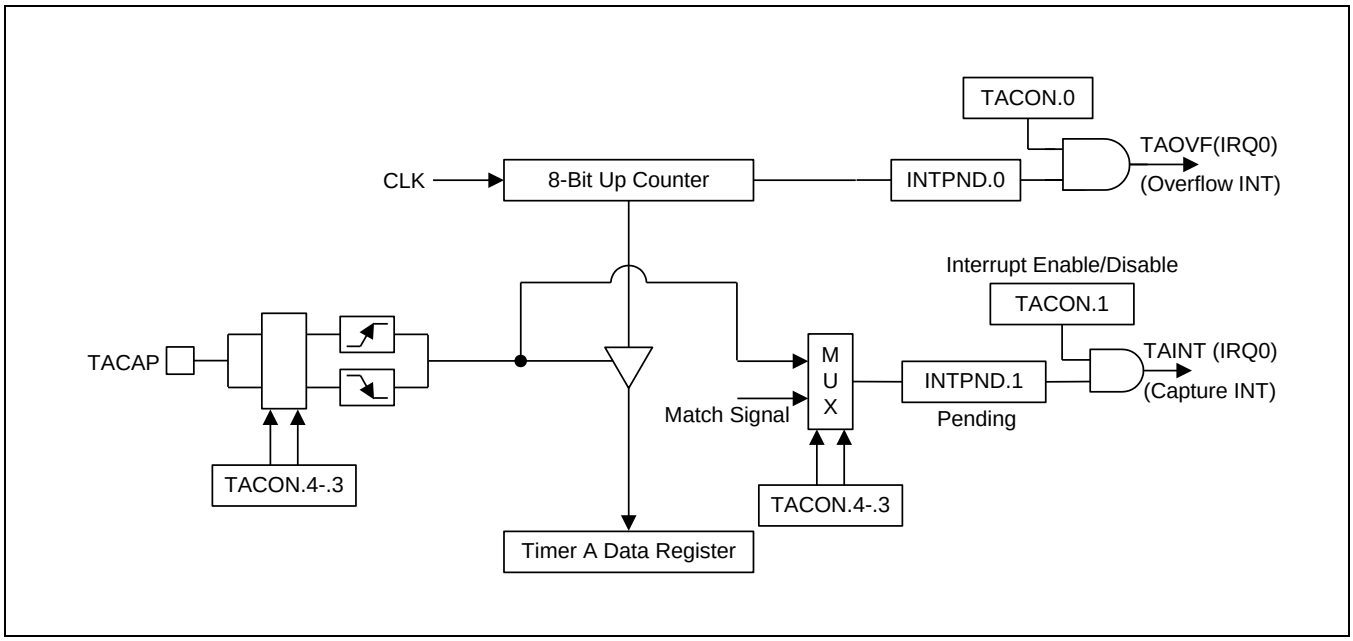


Figure 11-4 Simplified Timer A Function Diagram: Capture Mode

11.1.4 Block Diagram

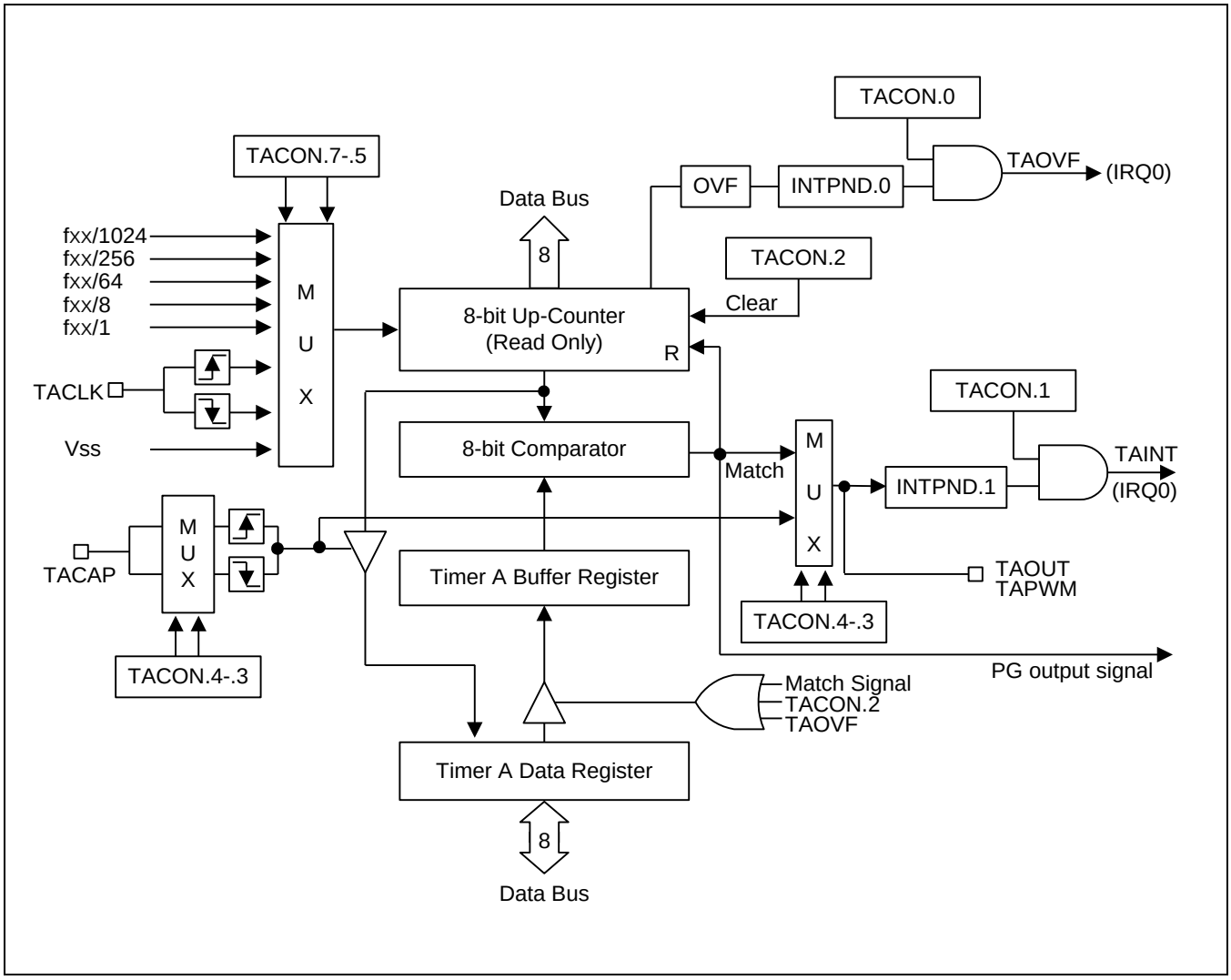


Figure 11-5 Timer A Functional Block Diagram

11.2 8-Bit Timer B

11.2.1 Overview

The S3F8S6B micro-controller has an 8-bit counter called timer B. Timer B, which can be used to generate the carrier frequency of a remote controller signal.

Timer B has two functions:

- As a normal interval timer, generating a timer B interrupt at programmed time intervals.
- To supply a clock source to the 8-bit timer/counter module, timer B, for generating the timer B overflow interrupt.

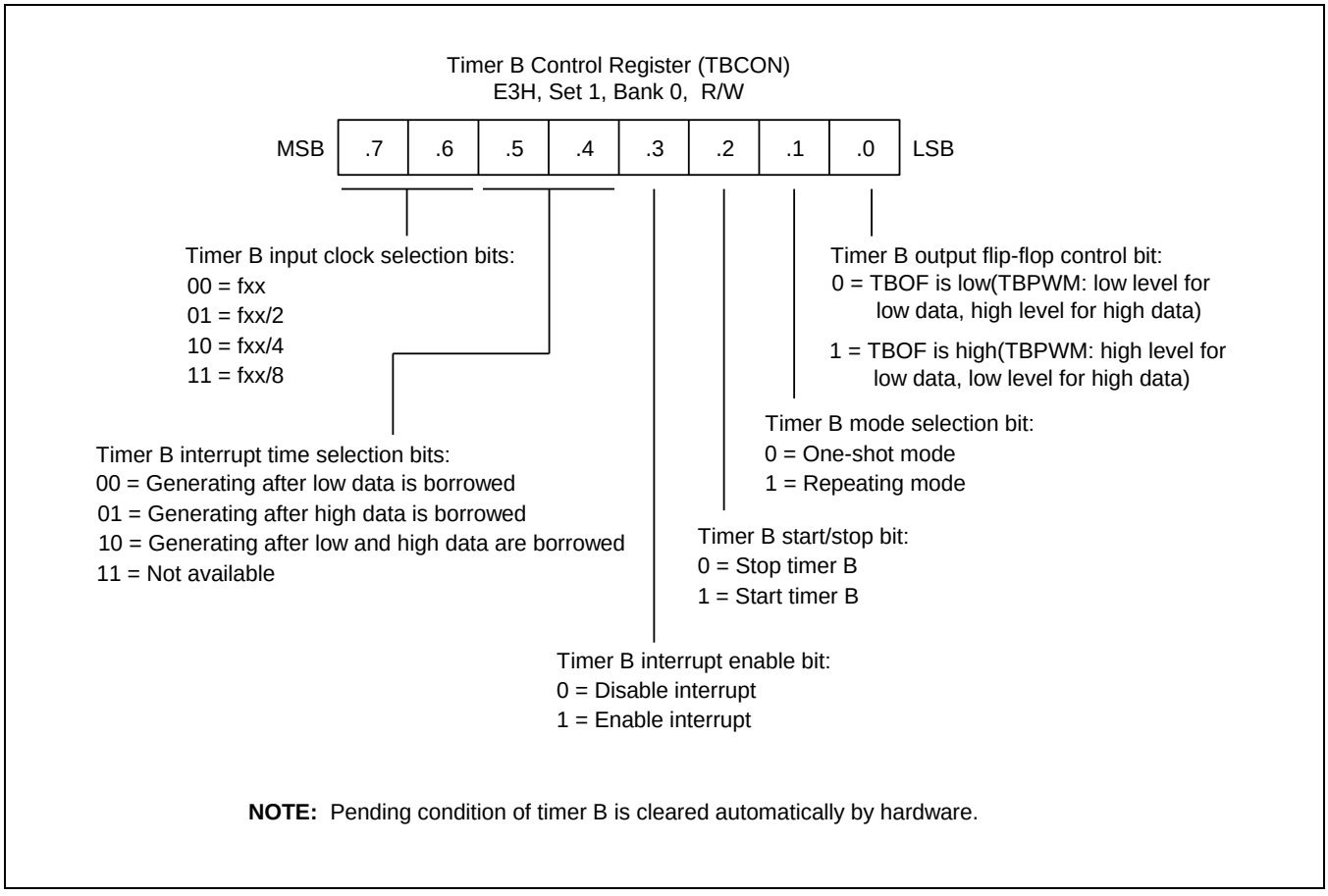
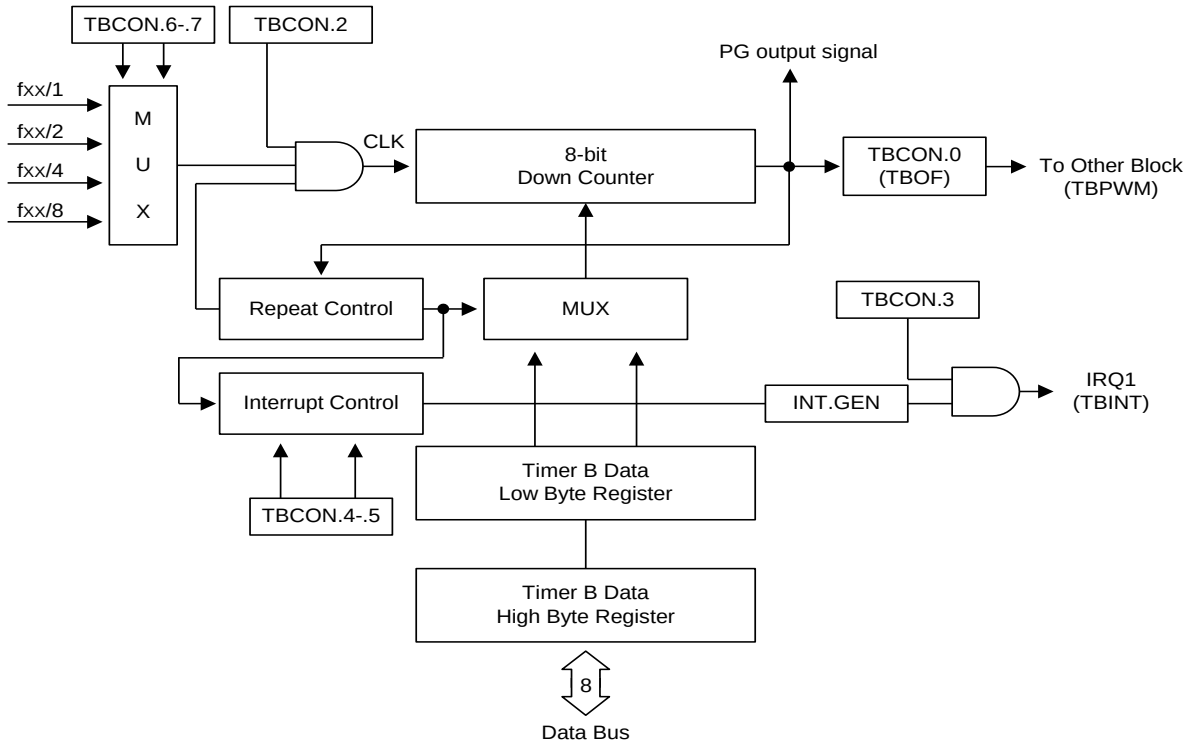


Figure 11-6 Timer B Control Register

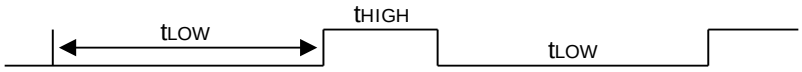
11.2.2 Block diagram



NOTE: The value of the TBDATAL register is loaded into the 8-bit counter when the operation of the timer B starts. If a borrow occurs in the counter, the value of the TBDATAH register is loaded into the 8-bit counter. However, if the next borrow occurs, the value of the TBDATAL register is loaded into the 8-bit counter.

Figure 11-7 Timer B Functional Block Diagram

11.2.3 Timer b PULSE WIDTH CALCULATIONS



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

When TBOF = 0,
t_{LOW} = (TBDATAL + 2) × 1/fx, 0H < TBDATAL < 100H, where fx = The selected clock.
t_{HIGH} = (TBDATAH + 2) × 1/fx, 0H < TBDATAH < 100H, where fx = The selected clock.

When TBOF = 1,
t_{LOW} = (TBDATAH + 2) × 1/fx, 0H < TBDATAH < 100H, where fx = The selected clock.
t_{HIGH} = (TBDATAL + 2) × 1/fx, 0H < TBDATAL < 100H, where fx = The selected clock.

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

When TBOF = 0,
t_{LOW} = 24 μs = (TBDATAL + 2) /fx = (TBDATAL + 2) × 1 μs, TBDATAL = 22.
t_{HIGH} = 15 μs = (TBDATAH + 2) /fx = (TBDATAH + 2) × 1 μs, TBDATAH = 13.

When TBOF = 1,
t_{HIGH} = 15 μs = (TBDATAL + 2) /fx = (TBDATAL + 2) × 1 μs, TBDATAL = 13.
t_{LOW} = 24 μs = (TBDATAH + 2) /fx = (TBDATAH + 2) × 1 μs, TBDATAH = 22.

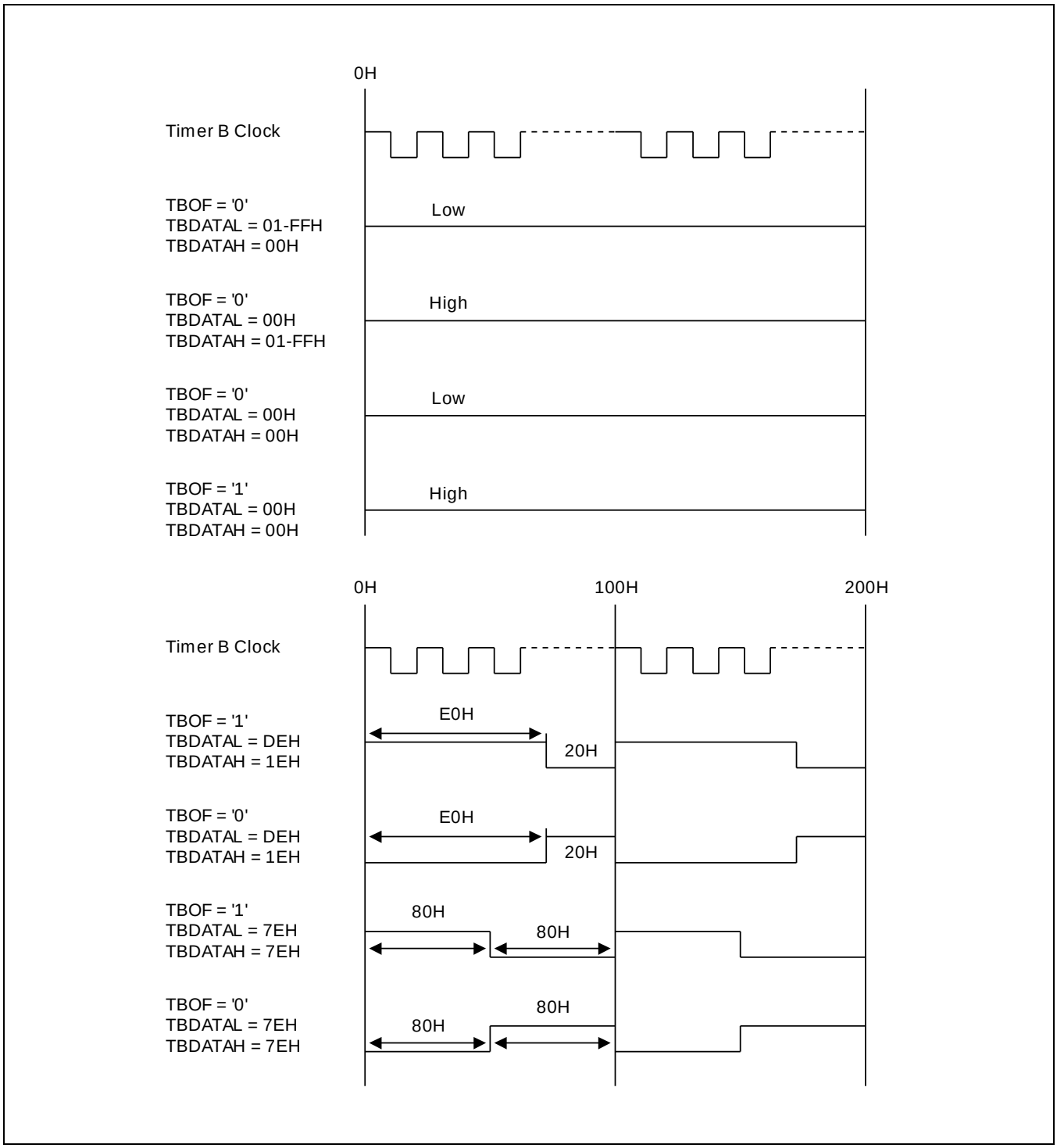
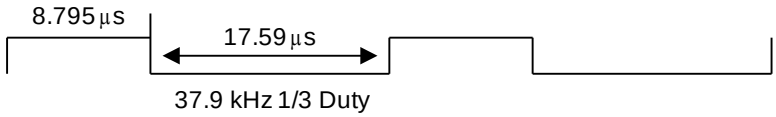


Figure 11-8 Timer B Output Flip-Flop Waveforms in Repeat Mode

Example 11-1 To Generate 38 kHz, 1/3 Duty Signal Through P3.0

This example sets Timer B to the repeat mode, sets the oscillation frequency as the Timer B clock source, and TBDATAH and TBDATAL to make a 38 kHz, 1/3 Duty carrier frequency. The program parameters are:



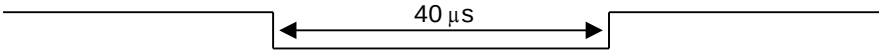
- Timer B is used in repeat mode
- Oscillation frequency is 4 MHz (0.25□□s)
- TBDATAH = 8.795 μs/0.25 μs = 35.18, TBDATAL = 17.59 μs/0.25 μs = 70.36
- Set P3.0 to TBPWM mode.

```
      ORG     0100H                ; Reset address
START DI
      •
      •
      •
      LD      TBDATAL,#(70-2)      ; Set 17.5□□s
      LD      TBDATAH,#(35-2)      ; Set 8.75□□s
      LD      TBCON,#00000111B     ; Clock Source ← fxx
                                   ; Disable Timer B interrupt.
                                   ; Select repeat mode for Timer B.
                                   ; Start Timer B operation.
                                   ; Set Timer B Output flip-flop (TBOF) high.

      OR      P3CONL,#07H          ; Set P3.0 to TBPWM mode.
                                   ; This command generates 38 kHz, 1/3 duty pulse signal
                                     through P3.0.
      •
      •
      •
```

Example 11-2 To Generate a One Pulse Signal Through P3.0

This example sets Timer B to the one shot mode, sets the oscillation frequency as the Timer B clock source, and TBDATAH and TBDATAL to make a 40 μs width pulse. The program parameters are:



- Timer B is used in one shot mode
- Oscillation frequency is 4 MHz (1 clock = 0.25 μs)
- TBDATAH = 40 μs/0.25 μs = 160, TBDATAL = 1
- Set P3.0 to TBPWM mode

```
      ORG     0100H                       ;       Reset address
START DI
      •
      •
      •
      LD      TBDATAH,# (160-2)           ; Set 40□□s
      LD      TBDATAL,# 1                 ; Set any value except  00H
      LD      TBCON,#00000001B            ; Clock Source ← fOSC
                                          ; Disable Timer B interrupt.
                                          ; Select one shot mode for Timer B.
                                          ; Stop Timer B operation.
                                          ; Set Timer B output flip-flop (TBOF) high
      OR      P3CONL, #07H                ; Set P3.0 to TBPWM mode.
      •
      •
Pulse_out:    LD      TBCON,#00000101B    ; Start Timer B operation
                                          ; To make the pulse at this point.
      •                                   ; After the instruction is executed, 0.75 □s is required
      •                                   ; before the falling edge of the pulse starts.
      •
```

12 8-Bit Timer C

12.1 8-Bit Timer C

12.1.1 Overview

The 8-bit timer C is an 8-bit general-purpose timer/counter.

Timer C has two operating mode, you can select one of them using the appropriate TCCON setting:

- Interval timer mode (Toggle output at TCOUT pin), only match interrupt occurs
- PWM mode (TCPWM pin), match and overflow interrupt can occur

Timer C has the following functional components:

- Clock frequency divider with multiplexer
- 8-bit counter, 8-bit comparator, and 8-bit reference data register (TCDATA)
- PWM or match output (TCOUT/TCPWM)
- Timer C match/overflow interrupt (IRQ2, vector D4H) generation
- Timer C control register, TCCON (set 1, bank0, ECH, read/write)

12.1.2 Timer c Control Register (TcCON)

You use the timer C control register, TCCON, to

- Select the timer C operating mode (fxx/1 & PWM mode or fxx/64 & interval mode)
- Select the timer C 3-bits prescaler
- Clear the timer C counter, TCCNT
- Enable the timer C match/overflow interrupt
- Start the timer C

TCCON is located in set 1, Bank 0 at address ECH, and is read/write addressable using Register addressing mode.

A reset clears TCCON to '00H'. This sets timer C to fxx/1&PWM timer mode, selects a 3-bits prescaler of non divided, stop timer C and disables all timer C interrupts. You can clear the timer C counter at any time during normal operation by writing a "1" to TCCON.3.

To enable the timer C match/overflow interrupt (IRQ2, vector D4H), you must write TCCON.7 and TCCON.1 to "1". To generate the exact time interval, you should write TCCON.3 and 0, which cleared counter and interrupt pending bit. To detect an interrupt pending condition when TCINT is disabled, the application program poll pending bit, TCCON.0. When a "1" is detected, a timer C match/overflow interrupt is pending. When the TCINT sub-routine has been serviced, the pending condition is cleared automatically by hardware or must be cleared by software by writing a "0" to the timer C interrupt pending bit, TCCON.0.

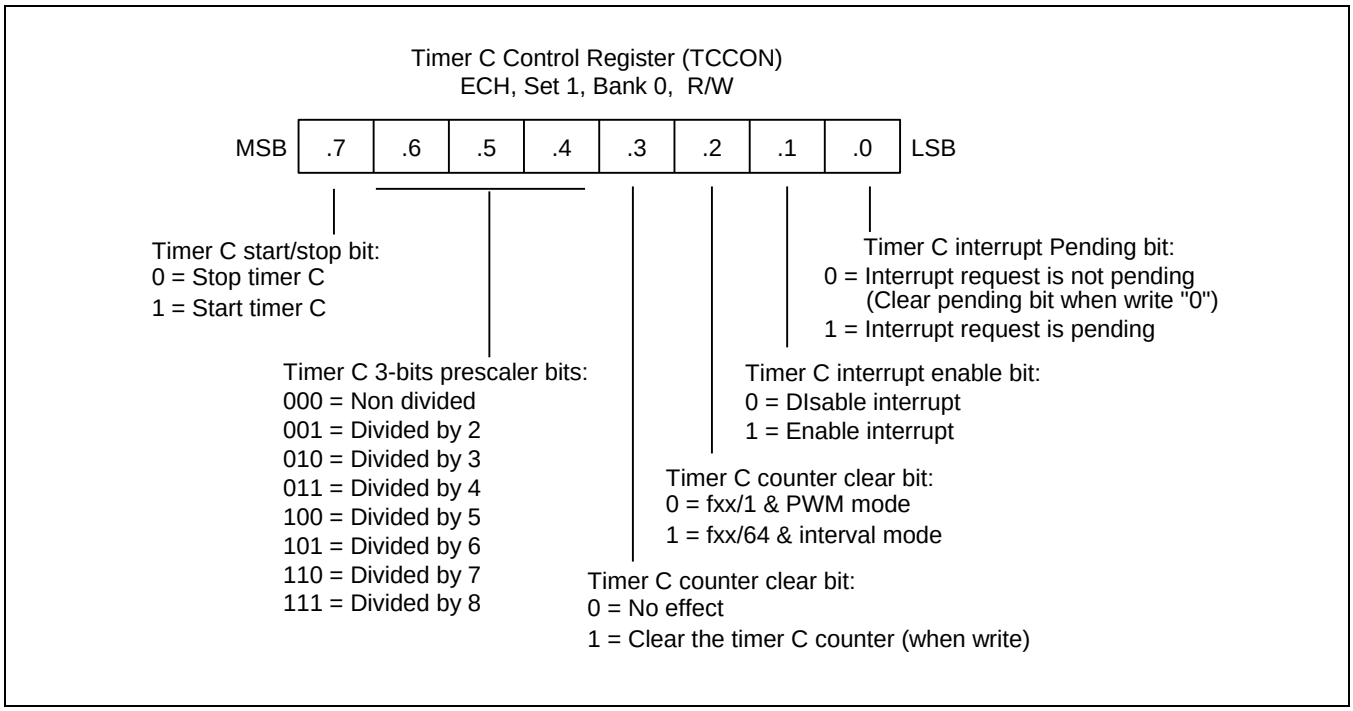
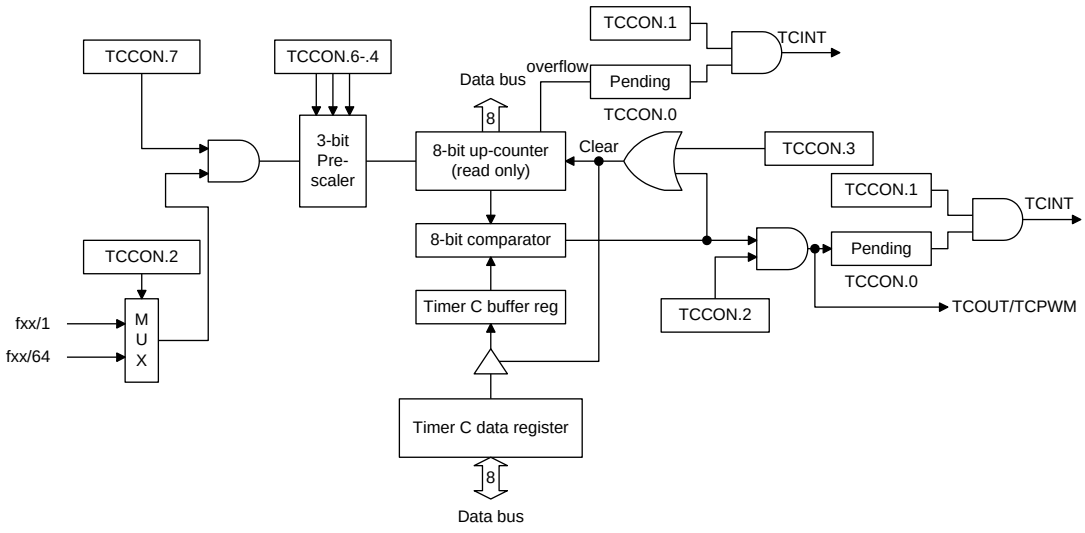


Figure 12-1 Timer C Control Register (TCCON)

12.1.3 Block Diagram



NOTE: When PWM mode, match signal cannot clear counter.

Figure 12-2 Timer C Function Block Diagram

13 16-Bit Timer D0/D1

13.1 16-Bit Timer D0

13.1.1 Overview

The 16-bit timer D0 is an 16-bit general-purpose timer.

Timer D0 has three operating modes, one of which you select using the appropriate TD0CON, TD0CON setting is

- Interval timer mode (Toggle output at TD0OUT pin)
- Capture input mode with a rising or falling edge trigger at the TD0CAP pin
- PWM mode (TD0PWM); PWM output shares their output port with TD0OUT pin

Timer D0 has the following functional components:

- Clock frequency divider (fxx divided by 1024, 256, 64, 8, 1) with multiplexer
- External clock input pin (TD0CLK)
- A 16-bit counter (TD0CNTH/L), a 16-bit comparator, and two 16-bit reference data register (TD0DATAH/L)
- I/O pins for capture input (TD0CAP), or match output (TD0OUT)
- Timer D0 overflow interrupt (IRQ3, vector D8H) and match/capture interrupt (IRQ3, vector DAH) generation
- Timer D0 control register, TD0CON (set 1, Bank 1, FAH, read/write)

13.1.2 Timer D0 Control Register (TD0CON)

You use the timer D0 control register, TD0CON, to

- Select the timer D0 operating mode (interval timer, capture mode, or PWM mode)
- Select the timer D0 input clock frequency
- Clear the timer D0 counter, TD0CNTH/TD0CNTL
- Enable the timer D0 overflow interrupt or timer D0 match/capture interrupt

TD0CON is located in set 1 and bank 1 at address FAH, and is read/write addressable using Register addressing mode.

A reset clears TD0CON to "00H". This sets timer D0 to normal interval timer mode, selects an input clock frequency of fxx/1024, and disables all timer D0 interrupts. To disable the counter operation, please set TD0CON.7-.5 to 111B. You can clear the timer D0 counter at any time during normal operation by writing a "1" to TD0CON.2.
The timer D0 overflow interrupt (TD0OVF) is interrupt level IRQ3 and has the vector address DAH. When a timer D0 overflow interrupt occurs and is serviced interrupt (IRQ3, vector DAH), you must write TD0CON.0 to "1". When a timer D0 overflow interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware or must be cleared by software.

To enable the timer D0 match/capture interrupt (IRQ3, vector D8H), you must write TD0CON.1 to "1". To detect a match/capture interrupt pending condition, the application program polls INTPND.3. When a "1" is detected, a timer D0 match or capture interrupt is pending. When the interrupt request has been serviced, the pending condition must be cleared by software by writing a "0" to the timer D0 match/capture interrupt pending bit, INTPND.3.

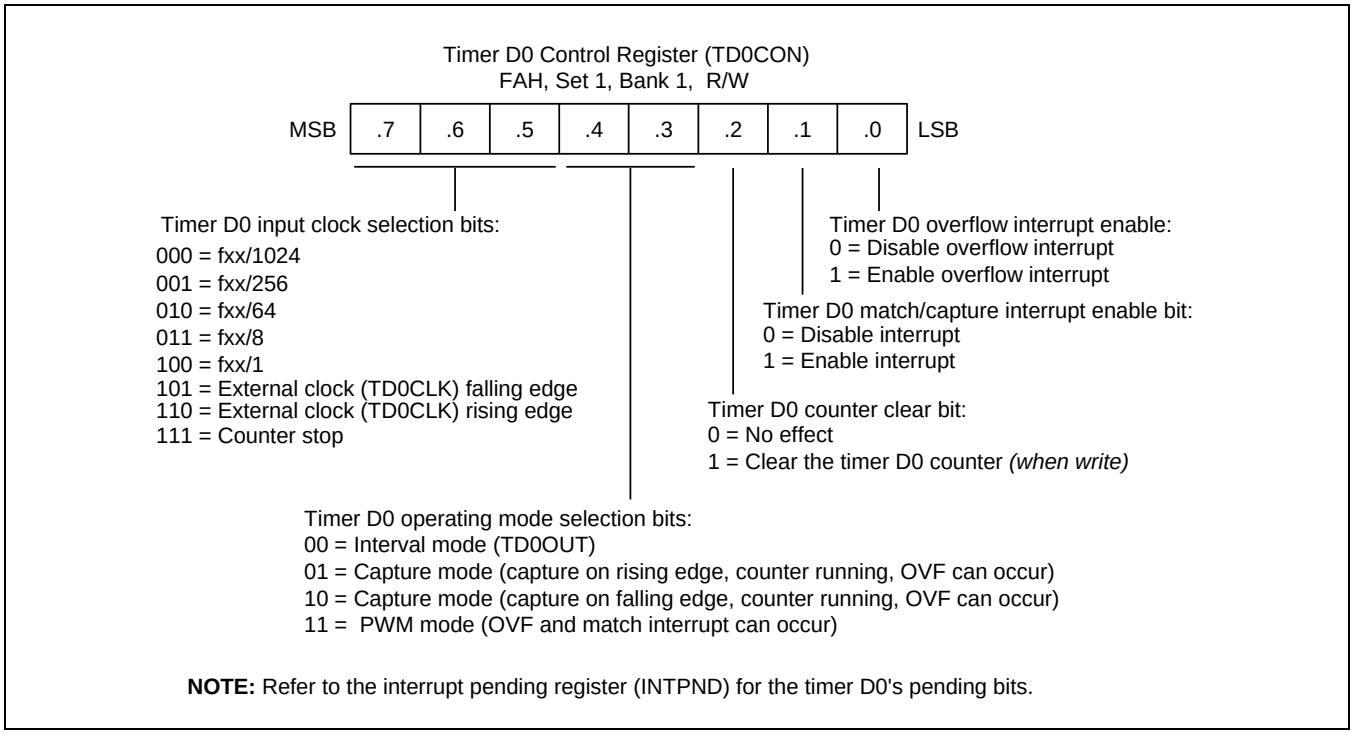


Figure 13-1 Timer D0 Control Register (TD0CON)

13.1.3 Timer D0 Function Description

13.1.3.1 Timer D0 Interrupts (IRQ3, Vectors D8H and DAH)

The timer D0 can generate two interrupts: the timer D0 overflow interrupt (TD0OVF), and the timer D0 match/capture interrupt (TD0INT). TD0OVF is belongs to interrupt level IRQ3, vector DAH. TD0INT also belongs to interrupt level IRQ3, but is assigned the separate vector address, D8H.

A timer D0 overflow interrupt pending condition is automatically cleared by hardware when it has been serviced or should be cleared by software in the interrupt service routine by writing a "0" to the INTPND.2 interrupt pending bit. However, the timer D0 match/capture interrupt pending condition must be cleared by the application's interrupt service routine by writing a "0" to the INTPND.3 interrupt pending bit.

13.1.3.2 Interval Timer Mode

In interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer D0 reference data register, TD0DATAH/TD0DATAL. The match signal generates a timer D0 match interrupt (TD0INT, vector D8H) and clears the counter.

If, for example, you write the value "1087H" to TD0DATAH/TD0DATAL, the counter will increment until it reaches "1087H". At this point, the timer D0 interrupt request is generated, the counter value is reset, and counting resumes. With each match, the level of the signal at the timer D0 output pin is inverted (see *Figure 13-2*).

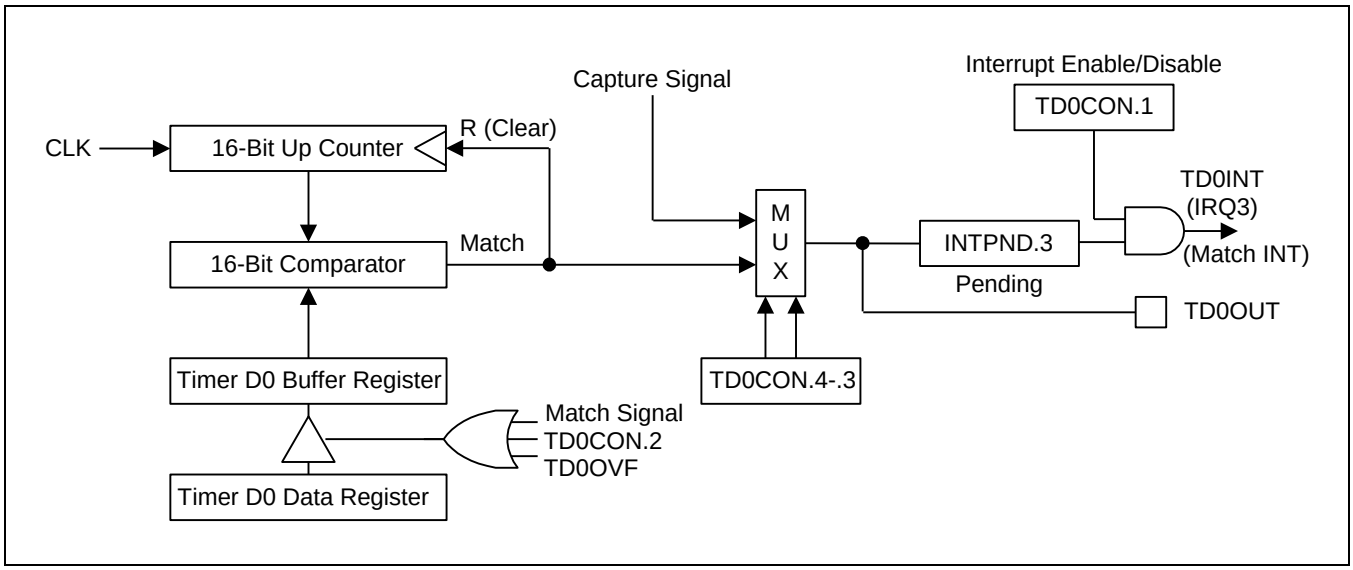


Figure 13-2 Simplified Timer D0 Function Diagram: Interval Timer Mode

13.1.3.3 Pulse Width Modulation Mode

Pulse width modulation (PWM) mode lets you program the width (duration) of the pulse that is output at the TD0PWM pin. As in interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer D0 data register. In PWM mode, however, the match signal does not clear the counter. Instead, it runs continuously, overflowing at "FFFFH", and then continues incrementing from "0000H".

Although you can use the match signal to generate a timer D0 overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the TD0PWM pin is held to Low level as long as the reference data value is less than or equal to (≤) the counter value and then the pulse is held to High level for as long as the data value is greater than (>) the counter value. One pulse width is equal to $t_{CLK} \times 65536$ (see *Figure 13-3*).

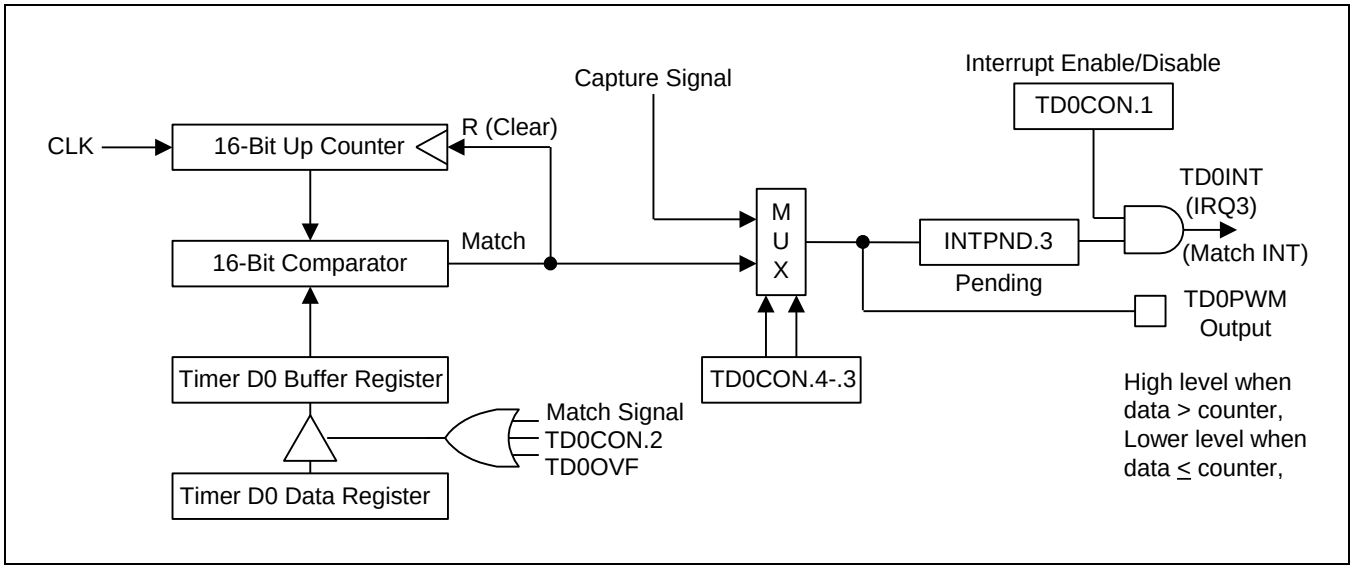


Figure 13-3 Simplified Timer D0 Function Diagram: PWM Mode

13.1.3.4 Capture Mode

In capture mode, a signal edge that is detected at the TD0CAP pin opens a gate and loads the current counter value into the timer D0 data register. You can select rising or falling edges to trigger this operation.

Timer D0 also gives you capture input source: the signal edge at the TD0CAP pin. You select the capture input by setting the values of the timer D0 capture input selection bits in the port 3 control register, P3CONM (set 1, bank 1, ECH). When P3CONM.5–.3 is "000", the TD0CAP input is selected.

Both kinds of timer D0 interrupts can be used in capture mode: the timer D0 overflow interrupt is generated whenever a counter overflow occurs; the timer D0 match/capture interrupt is generated whenever the counter value is loaded into the timer D0 data register.

By reading the captured data value in TD0DATAH/TD0DATAL, and assuming a specific value for the timer D0 clock frequency, you can calculate the pulse width (duration) of the signal that is being input at the TD0CAP pin (see *Figure 13-4*).

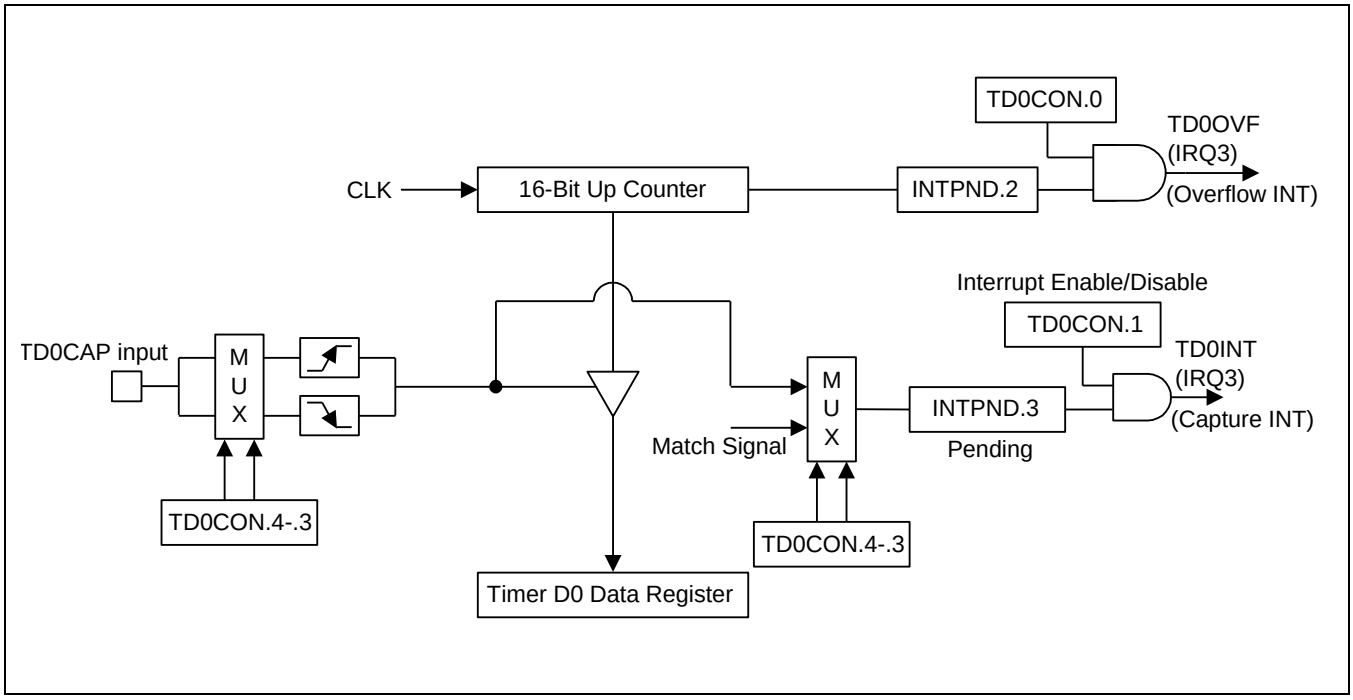


Figure 13-4 Simplified Timer D0 Function Diagram: Capture Mode

13.1.4 Block Diagram

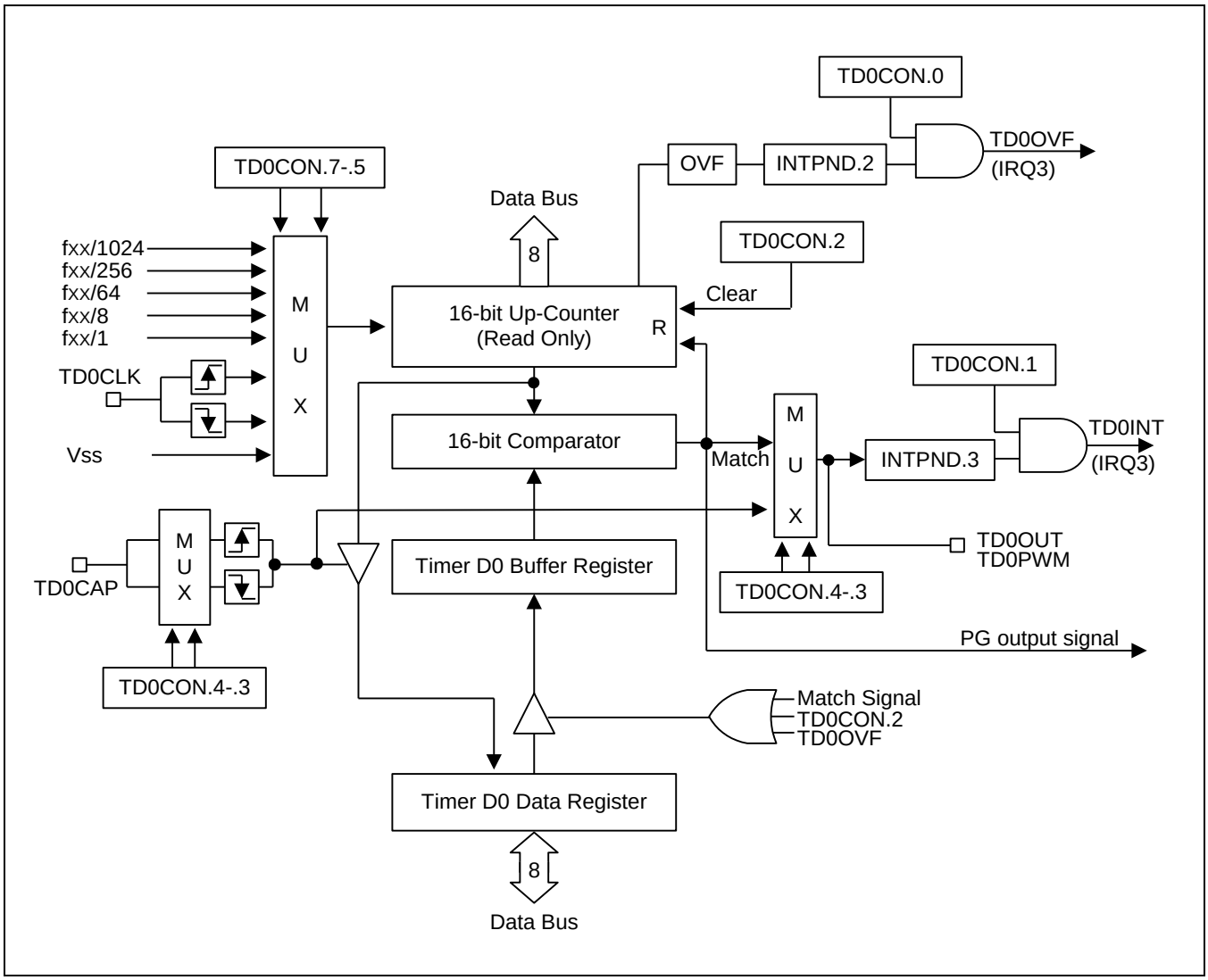


Figure 13-5 Timer D0 Functional Block Diagram

13.2 16-bit timer D1

13.2.1 Overview

The 16-bit timer D1 is a 16-bit general-purpose timer.

Timer D1 has three operating modes, one of which you select using the appropriate TD1CON, TD1CON setting is

- Interval timer mode (Toggle output at TD1OUT pin)
- Capture input mode with a rising or falling edge trigger at the TD1CAP pin
- PWM mode (TD1PWM); PWM output shares their output port with TD1OUT pin

Timer D1 has the following functional components:

- Clock frequency divider (fxx divided by 1024, 256, 64, 8, 1) with multiplexer
- External clock input pin (TD1CLK)
- A 16-bit counter (TD1CNTH/L), a 16-bit comparator, and two 16-bit reference data register (TD1DATAH/L)
- I/O pins for capture input (TD1CAP), or match output (TD1OUT)
- Timer D1 overflow interrupt (IRQ3, vector DEH) and match/capture interrupt (IRQ3, vector DCH) generation
- Timer D1 control register, TD1CON (set 1, Bank 1, FBH, read/write)

13.2.2 Timer D1 Control Register (TD1CON)

You use the timer D1 control register, TD1CON, to

- Select the timer D1 operating mode (interval timer, capture mode, or PWM mode)
- Select the timer D1 input clock frequency
- Clear the timer D1 counter, TD1CNTH/TD1CNTL
- Enable the timer D1 overflow interrupt or timer D1 match/capture interrupt

TD1CON is located in set 1 and bank 1 at address FBH, and is read/write addressable using Register addressing mode.

A reset clears TD1CON to '00H'. This sets timer D1 to normal interval timer mode, selects an input clock frequency of fxx/1024, and disables all timer D1 interrupts. To disable the counter operation, please set TD1CON.7-.5 to 111B. You can clear the timer D1 counter at any time during normal operation by writing a "1" to TD1CON.2.
The timer D1 overflow interrupt (TD1OVF) is interrupt level IRQ3 and has the vector address DEH. When a timer D1 overflow interrupt occurs and is serviced interrupt (IRQ3, vector DEH), you must write TD1CON.0 to "1". When a timer D1 overflow interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware or must be cleared by software.

To enable the timer D1 match/capture interrupt (IRQ3, vector DCH), you must write TD1CON.1 to "1". To detect a match/capture interrupt pending condition, the application program polls INTPND.5. When a "1" is detected, a timer D1 match or capture interrupt is pending. When the interrupt request has been serviced, the pending condition must be cleared by software by writing a "0" to the timer D1 match/capture interrupt pending bit, INTPND.5.

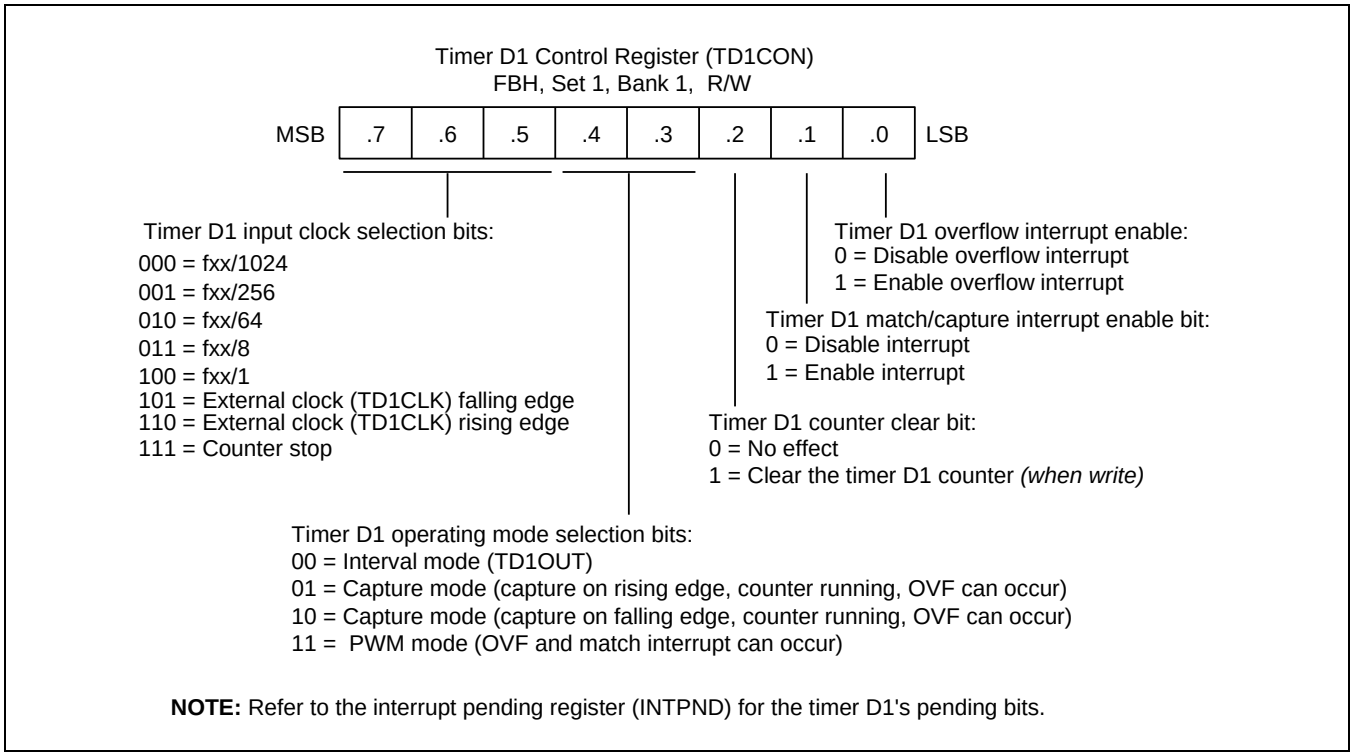


Figure 13-6 Timer D1 Control Register (TD1CON)

13.2.3 Timer D1 Function Description

13.2.3.1 Timer D1 Interrupts (IRQ3, Vectors DCH and DEH)

The timer D1 can generate two interrupts: the timer D1 overflow interrupt (TD1OVF), and the timer D1 match/capture interrupt (TD1INT). TD1OVF is belongs to interrupt level IRQ3, vector DEH. TD1INT also belongs to interrupt level IRQ3, but is assigned the separate vector address, DCH.

A timer D1 overflow interrupt pending condition is automatically cleared by hardware when it has been serviced or should be cleared by software in the interrupt service routine by writing a "0" to the INTPND.4 interrupt pending bit. However, the timer D1 match/capture interrupt pending condition must be cleared by the application's interrupt service routine by writing a "0" to the INTPND.5 interrupt pending bit.

13.2.3.2 Interval Timer Mode

In interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer D1 reference data register, TD1DATAH/TD1DATAL. The match signal generates a timer D1 match interrupt (TD1INT, vector DCH) and clears the counter.

If, for example, you write the value "1087H" to TD1DATAH/TD1DATAL, the counter will increment until it reaches "1087H". At this point, the timer D1 interrupt request is generated, the counter value is reset, and counting resumes. With each match, the level of the signal at the timer D1 output pin is inverted (see *Figure 13-7*).

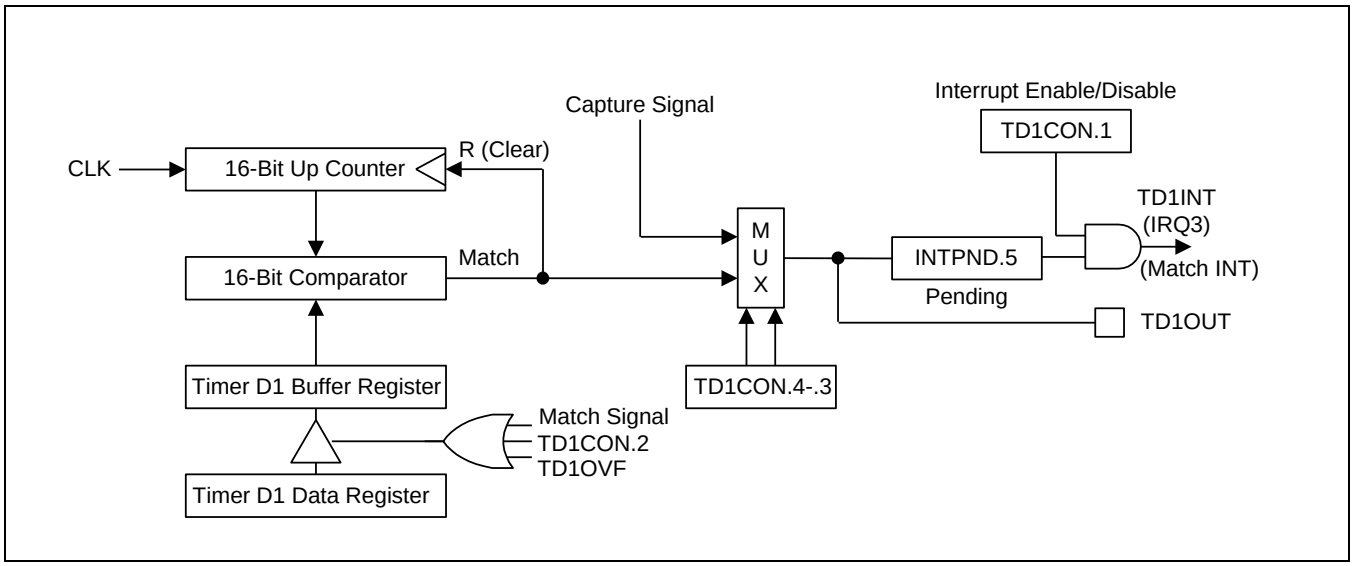


Figure 13-7 Simplified Timer D1 Function Diagram: Interval Timer Mode

13.2.3.3 Pulse Width Modulation Mode

Pulse width modulation (PWM) mode lets you program the width (duration) of the pulse that is output at the TD1PWM pin. As in interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer D1 data register. In PWM mode, however, the match signal does not clear the counter. Instead, it runs continuously, overflowing at "FFFFH", and then continues incrementing from "0000H".

Although you can use the match signal to generate a timer D1 overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the TD1PWM pin is held to Low level as long as the reference data value is less than or equal to (≤) the counter value and then the pulse is held to High level for as long as the data value is greater than (>) the counter value. One pulse width is equal to $t_{CLK} \times 65536$ (see *Figure 13-8*).

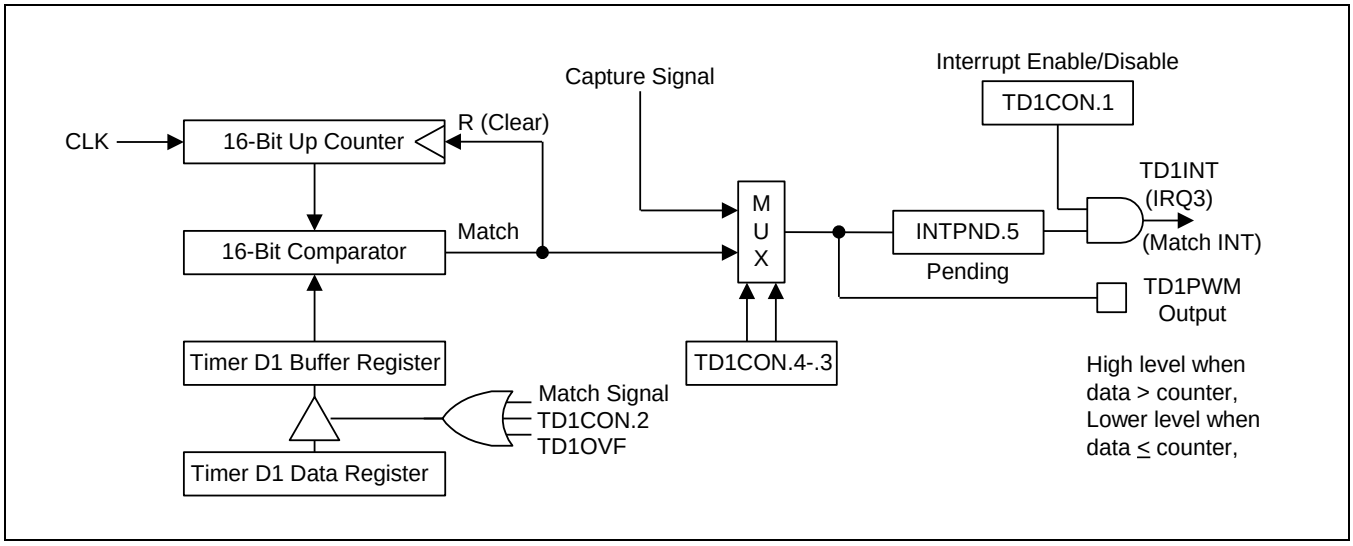


Figure 13-8 Simplified Timer D1 Function Diagram: PWM Mode

13.2.3.4 Capture Mode

In capture mode, a signal edge that is detected at the TD1CAP pin opens a gate and loads the current counter value into the timer D1 data register. You can select rising or falling edges to trigger this operation.

Timer D1 also gives you capture input source: the signal edge at the TD1CAP pin. You select the capture input by setting the values of the timer D1 capture input selection bits in the port 3 control register, P3CONH (set 1, bank 1, EBH). When P3CONH.2–.0 is "000", the TD1CAP input is selected.

Both kinds of timer D1 interrupts can be used in capture mode: the timer D1 overflow interrupt is generated whenever a counter overflow occurs; the timer D1 match/capture interrupt is generated whenever the counter value is loaded into the timer D1 data register.

By reading the captured data value in TD1DATAH/TD1DATAL, and assuming a specific value for the timer D1 clock frequency, you can calculate the pulse width (duration) of the signal that is being input at the TD1CAP pin (see *Figure 13-9*).

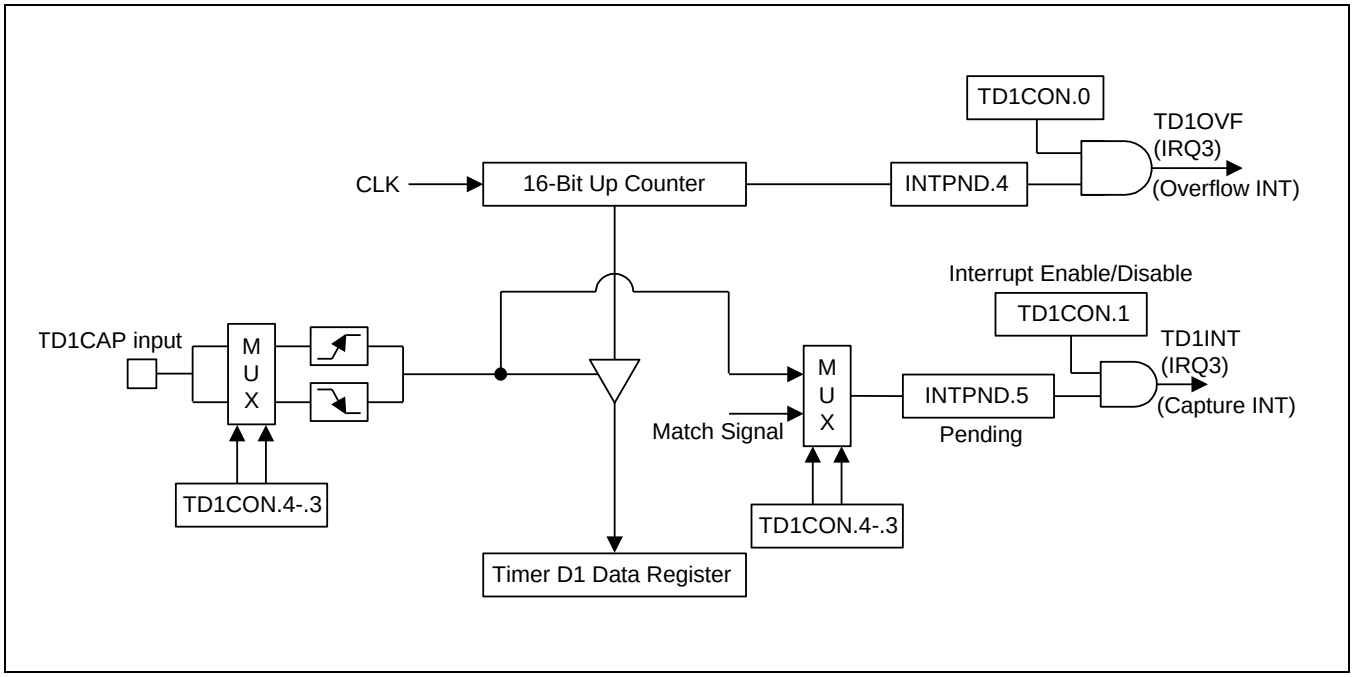


Figure 13-9 Simplified Timer D1 Function Diagram: Capture Mode

13.2.4 Block Diagram

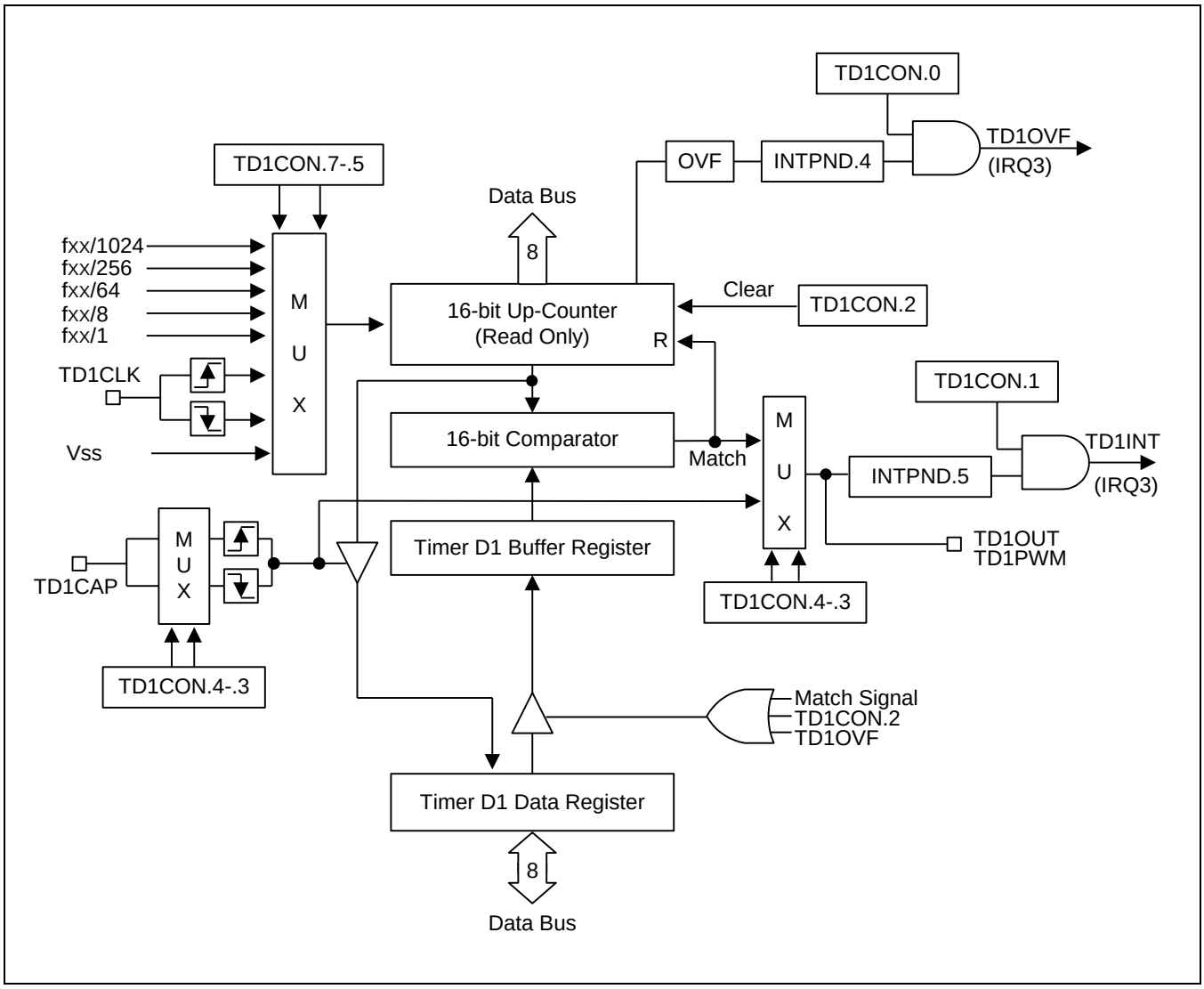


Figure 13-10 Timer D1 Functional Block Diagram

14 Watch Timer

14.1 Overview

Watch timer functions include real-time and watch-time measurement and interval timing for the system clock. To start watch timer operation, set bit 1 of the watch timer control register, WTCON.1 to "1".
And if you want to service watch timer overflow interrupt (IRQ4, vector E6H), then set the WTCON.6 to "1".
The watch timer overflow interrupt pending condition (WTCON.0) must be cleared by software in the application's interrupt service routine by means of writing a "0" to the WTCON.0 interrupt pending bit.
After the watch timer starts and elapses a time, the watch timer interrupt pending bit (WTCON.0) is automatically set to "1", and interrupt requests commence in 1.995 ms, 0.125, 0.25 and 0.5-second intervals by setting Watch timer speed selection bits (WTCON.3–.2).

The watch timer can generate a steady 0.5 kHz, 1 kHz, 2 kHz, or 4 kHz signal to BUZ output pin for Buzzer. By setting WTCON.3 and WTCON.2 to "11b", the watch timer will function in high-speed mode, generating an interrupt every 1.995 ms. High-speed mode is useful for timing events for program debugging sequences.

The watch timer supplies the clock frequency for the LCD controller (f_{LCD}). Therefore, if the watch timer is disabled, the LCD controller does not operate.

Watch timer has the following functional components:

- Real Time and Watch-Time Measurement
- Using a Main Clock Source or Sub clock
- Clock Source Generation for LCD Controller (f_{LCD})
- I/O pin for Buzzer Output Frequency Generator (BUZ)
- Timing Tests in High-Speed Mode
- Watch timer overflow interrupt (IRQ4, vector E6H) generation
- Watch timer control register, WTCON (set 1, bank 0, E6H, read/write)

14.2 Watch Timer CONTROL Register (WTCON)

The watch timer control register, WTCON is used to select the watch timer interrupt time and Buzzer signal, to enable or disable the watch timer function. It is located in set 1, bank 0 at address E6H, and is read/write addressable using register addressing mode.

A reset clears WTCON to "00H". This disable the watch timer.
So, if you want to use the watch timer, you must write appropriate value to WTCON.

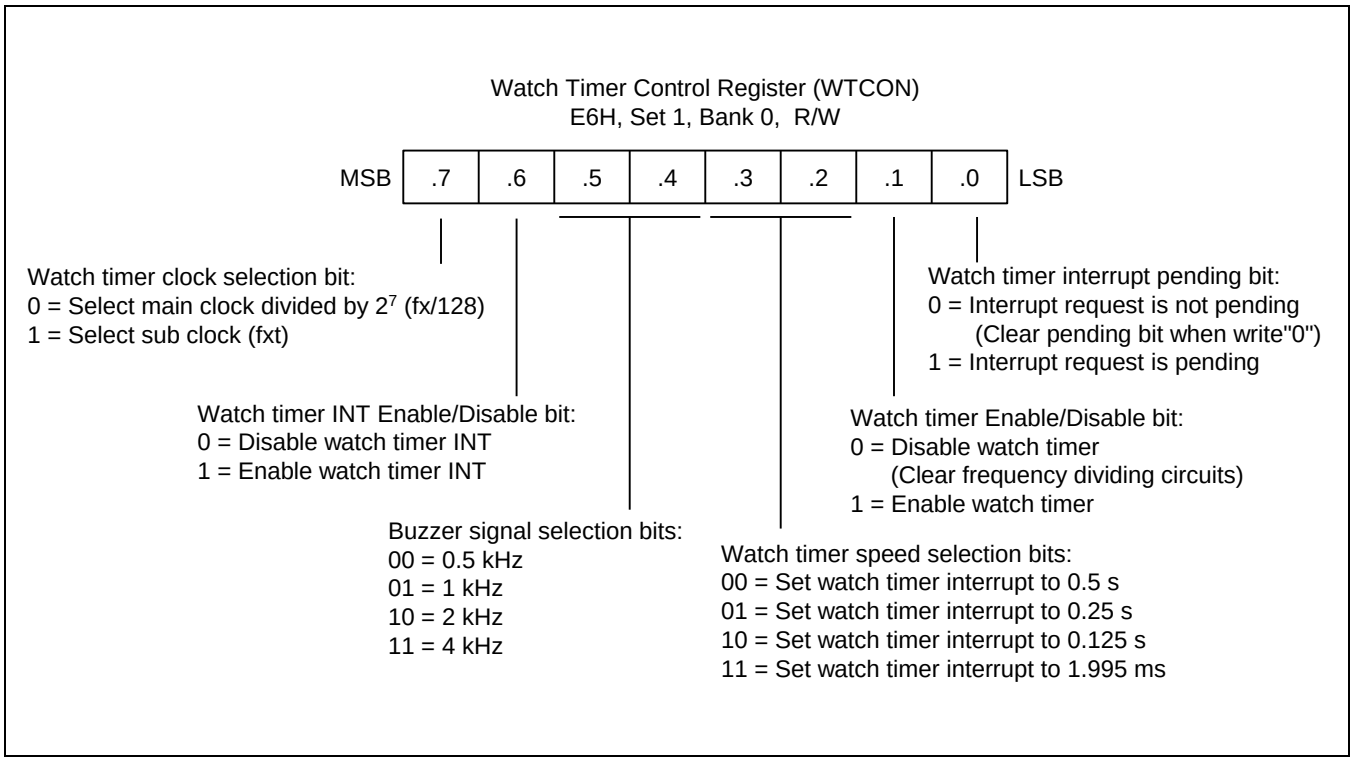


Figure 14-1 Watch Timer Control Register (WTCON)

14.3 Watch Timer Circuit Diagram

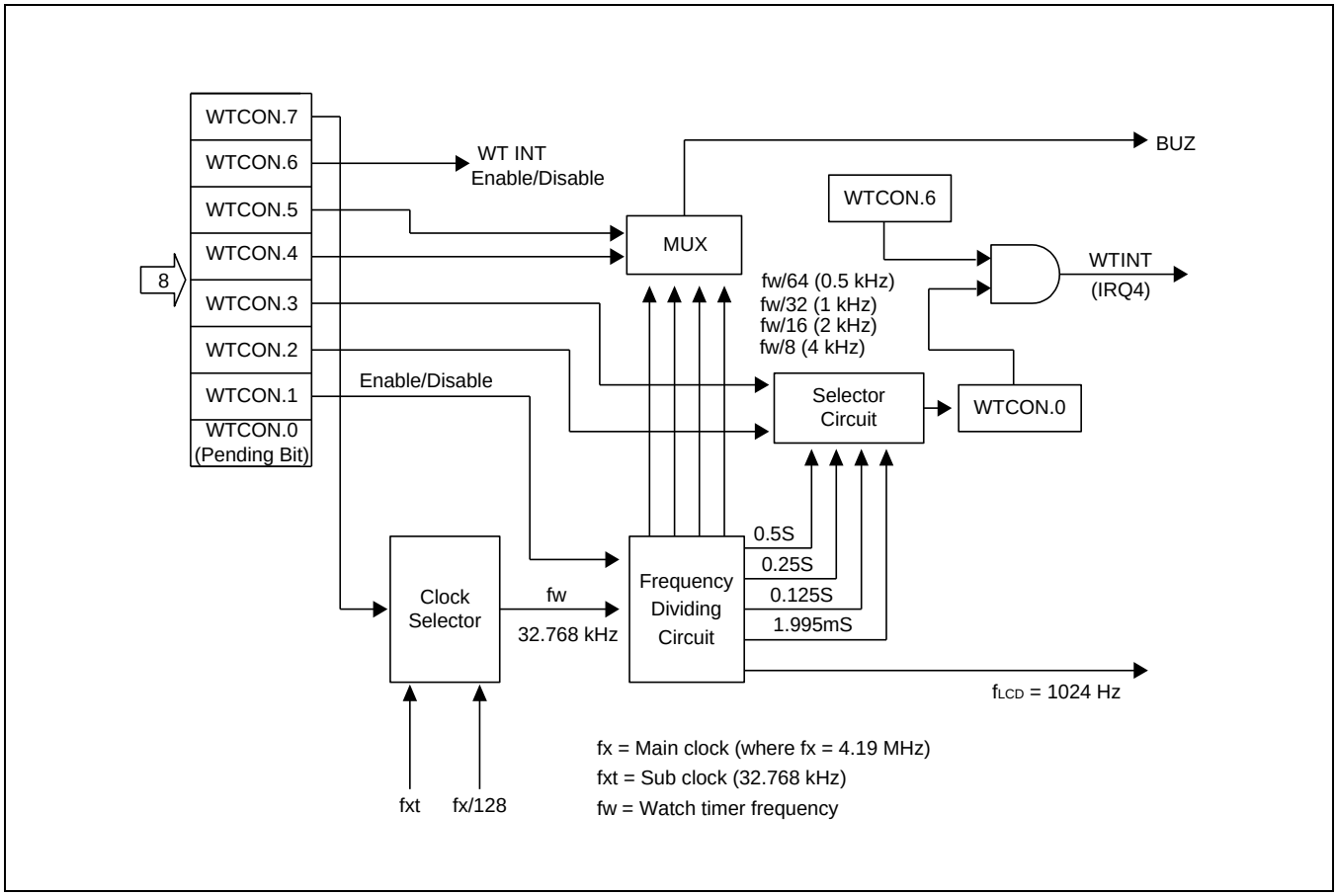


Figure 14-2 Watch Timer Circuit Diagram

15 LCD Controller/Driver

15.1 Overview

The S3F8S6B microcontroller can directly drive an up-to-224-dot (28 segments × 8 commons) LCD panel.

Its LCD block has the following components:

- LCD controller/driver
- Display RAM (30H–51H) for storing display data in page 8
- 6 common/segment output pins (COM2/SEG0–COM7/SEG5)
- 64 segment output pins (SEG6–SEG33)
- 2 common output pins (COM0–COM1)
- Four LCD operating power supply pins (V_{LC0}–V_{LC3})
- LCD bias by internal/external register and capacitor bias

The LCD control register, LCON, is used to turn the LCD display on and off, select LCD clock frequency, LCD duty and bias, and LCD bias type. The LCD mode control register, LMOD, is used to selects VLCD voltage. Data written to the LCD display RAM can be automatically transferred to the segment signal pins without any program control. When a subsystem clock is selected as the LCD clock source, the LCD display is enabled even in the main clock Stop or Idle modes.

LCD data stored in the display RAM locations are transferred to the segment signal pins automatically without program control.

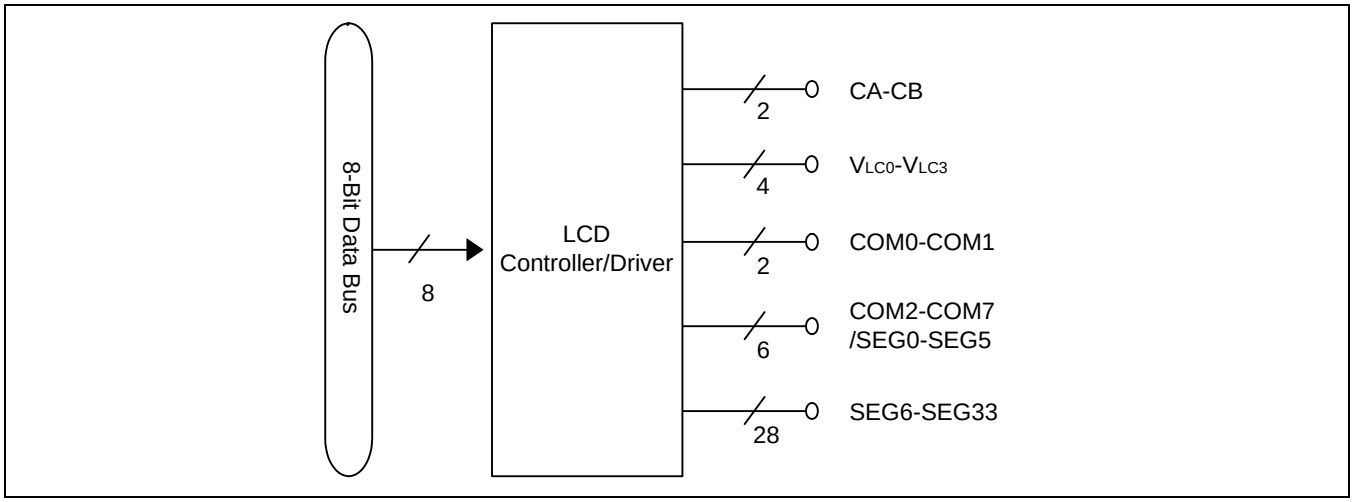


Figure 15-1 LCD Function Diagram

15.2 LCD Circuit Diagram

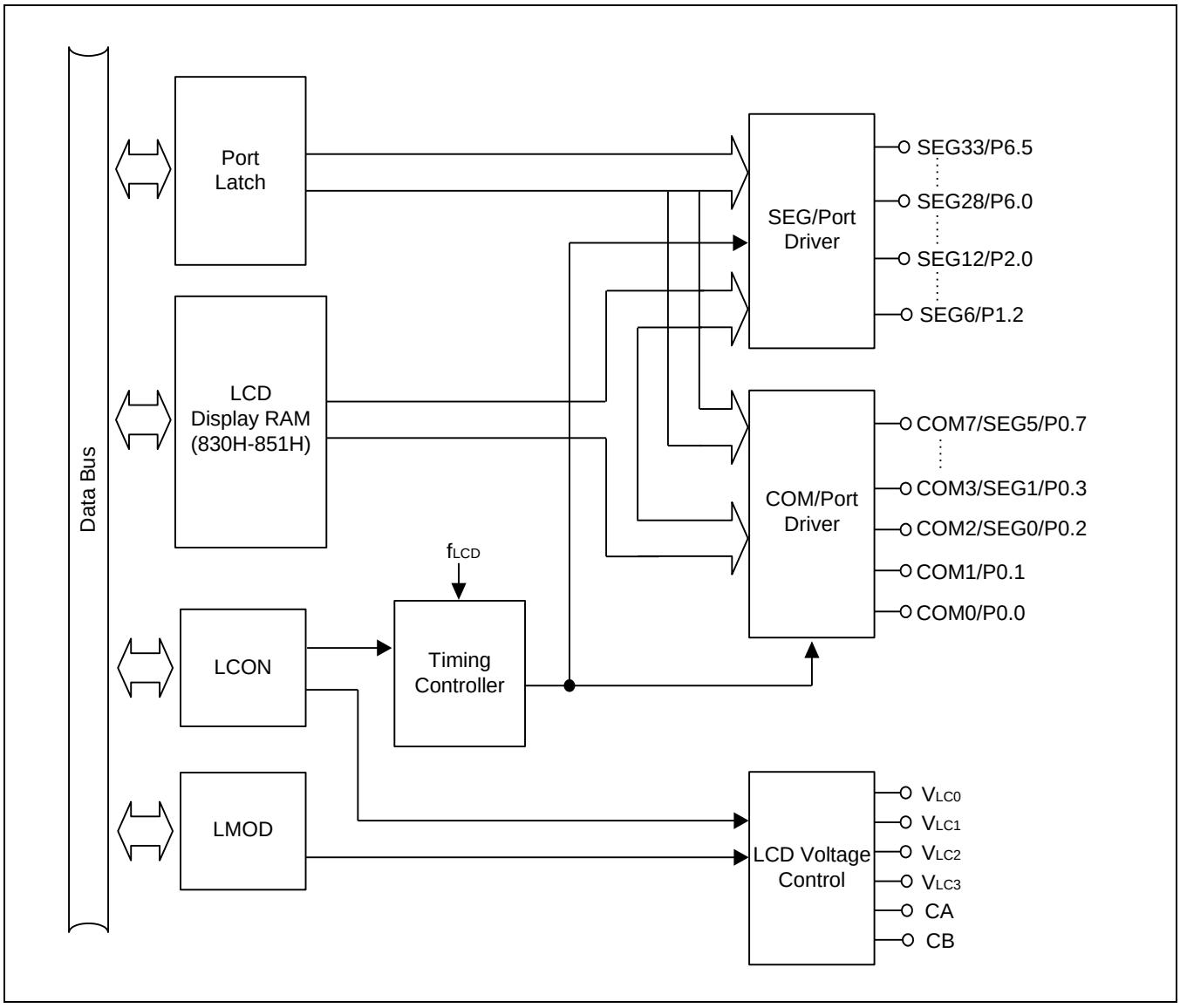


Figure 15-2 LCD Circuit Diagram

15.3 LCD RAM Address Area

RAM addresses of 30H - 75H page 8 are used as LCD data memory. These locations can be addressed by 1-bit or 8-bit instructions. When the bit value of a display segment is "1", the LCD display is turned on; When the bit value is "0", the display is turned off.

Display RAM data are sent out through the segment pins, SEG0–SEG33, using the direct memory access (DMA) method that is synchronized with the f_{LCD} signal. RAM addresses in this location that are not used for LCD display can be allocated to general-purpose use.

COM	Bit	SEG0	SEG1	SEG2	SEG3	SEG4	------	SEG32	SEG33
COM0	.0	830H	831H	832H	833H	834H	------	850H	851H
COM1	.1								
COM2	.2								
COM3	.3								
COM4	.4								
COM5	.5								
COM6	.6								
COM7	.7								

Figure 15-3 LCD Display Data RAM Organization

15.4 LCD control register (lCON)

A LCON is located in set1, bank0 at address F0H, and is read/write addressable using register addressing mode. It has the following control functions.

- LCD duty and bias selection
- LCD clock selection
- LCD display control
- Internal/External LCD dividing resistors or capacitor bias selection

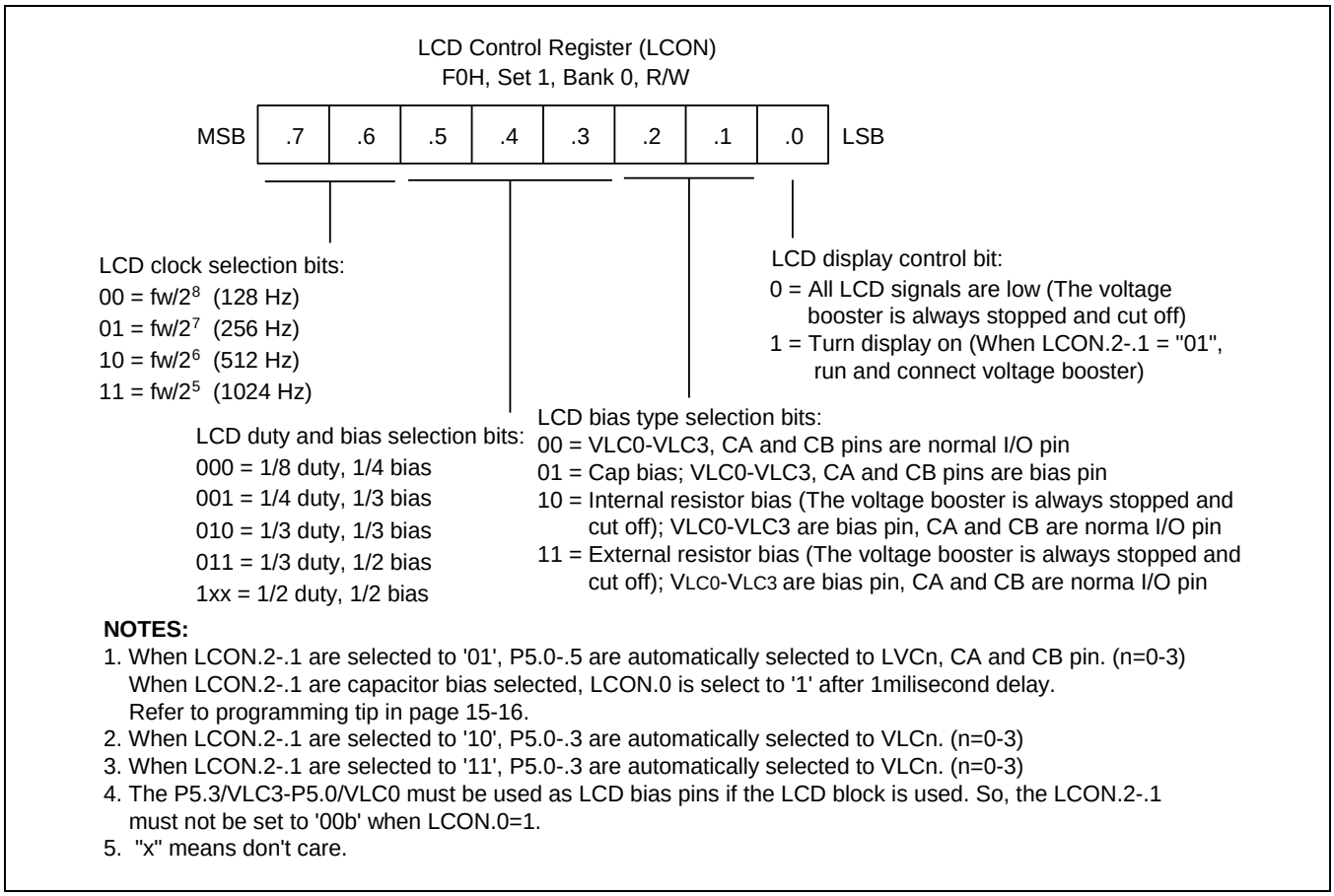
The LCON register is used to turn the LCD display on/off, to select duty and bias, to select LCD clock and LCD bias type. A reset clears the LCON registers to "00H", configuring turns off the LCD display, select 1/8 duty and 1/4 bias, select 128Hz for LCD clock, and no-select any LCD bias type.

The LCD clock signal determines the frequency of COM signal scanning of each segment output. Since the LCD clock is generated by watch timer clock (fw), the watch timer should be enabled when the LCD display is turned on. The LCD frame rate is the LCD clock frequency times the duty cycle (LCD clock * LCD duty), as shown in Table 15.1.

Table 15.1. LCD Frame Rate

LCD Clock Frequency	LCD Frame Frequency					Units
	Static	1/2 Duty	1/3 Duty	1/4 Duty	1/8 Duty	
128	128	64	43	32	16	Hz
256	256	128	85	64	32	
512	512	256	171	128	64	
1024	1024	512	341	256	128	

NOTE: The clock and duty for LCD controller/driver is automatically initialized by hardware, whenever the LCON register data value is rewritten. Therefore, the LCON register does not rewrite frequently.



NOTES:

1. When LCON.2-.1 are selected to '01', P5.0-.5 are automatically selected to LVCn, CA and CB pin. (n=0-3) When LCON.2-.1 are capacitor bias selected, LCON.0 is select to '1' after 1milisecond delay. Refer to programming tip in page 15-16.
2. When LCON.2-.1 are selected to '10', P5.0-.3 are automatically selected to VLCn. (n=0-3)
3. When LCON.2-.1 are selected to '11', P5.0-.3 are automatically selected to VLCn. (n=0-3)
4. The P5.3/VLC3-P5.0/VLC0 must be used as LCD bias pins if the LCD block is used. So, the LCON.2-.1 must not be set to '00b' when LCON.0=1.
5. "x" means don't care.

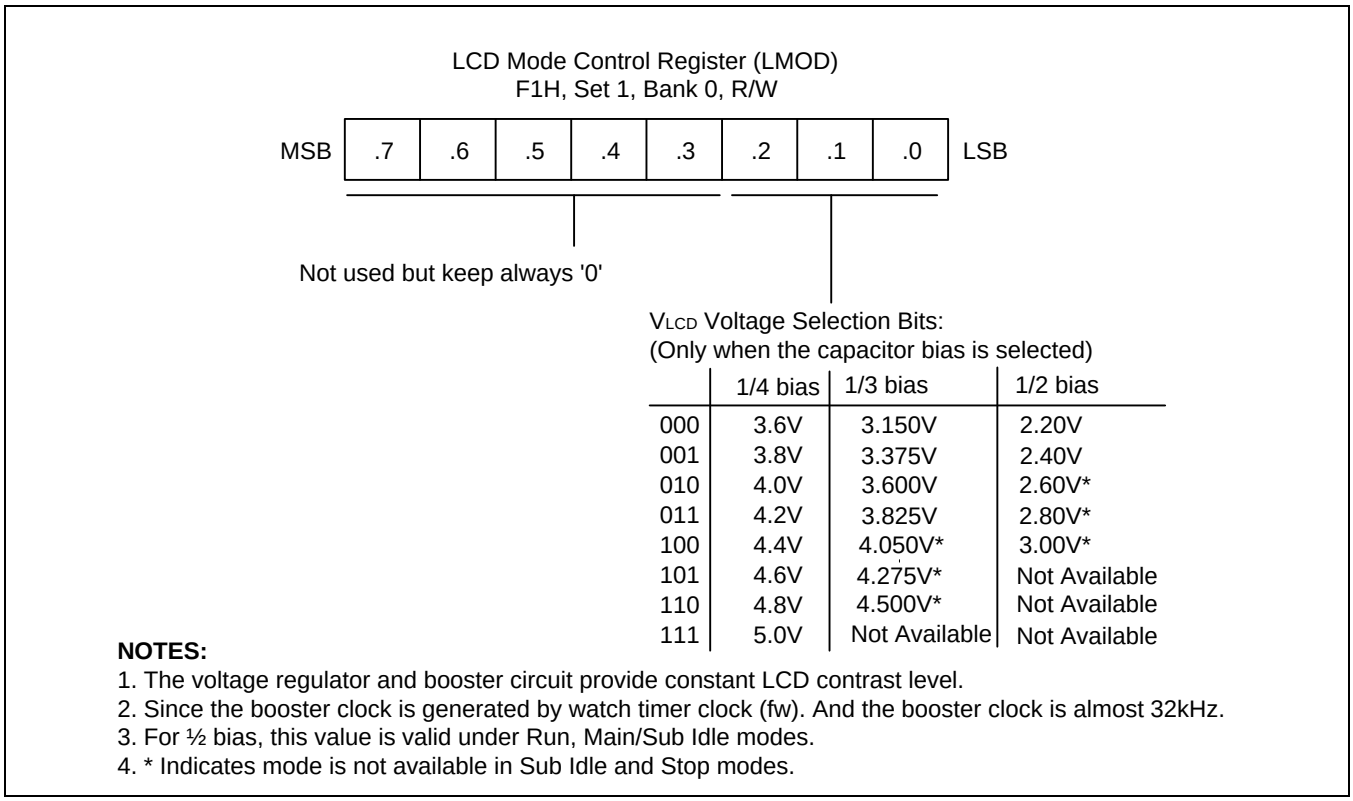
Figure 15-4 LCD Control Register (LCON)

15.5 LCD MODE Control Register (LMOD)

A LMOD is located in set 1, bank 0 at address F1H, and is read/write addressable using Register addressing mode. It has the following control functions.

- V_{LCD} voltage selection

The LMOD register is used to select VLCD voltage. A reset clears the LMOD registers to "00H", configuring select 3.6 V (when 1/4 bias) for VLCD voltage.

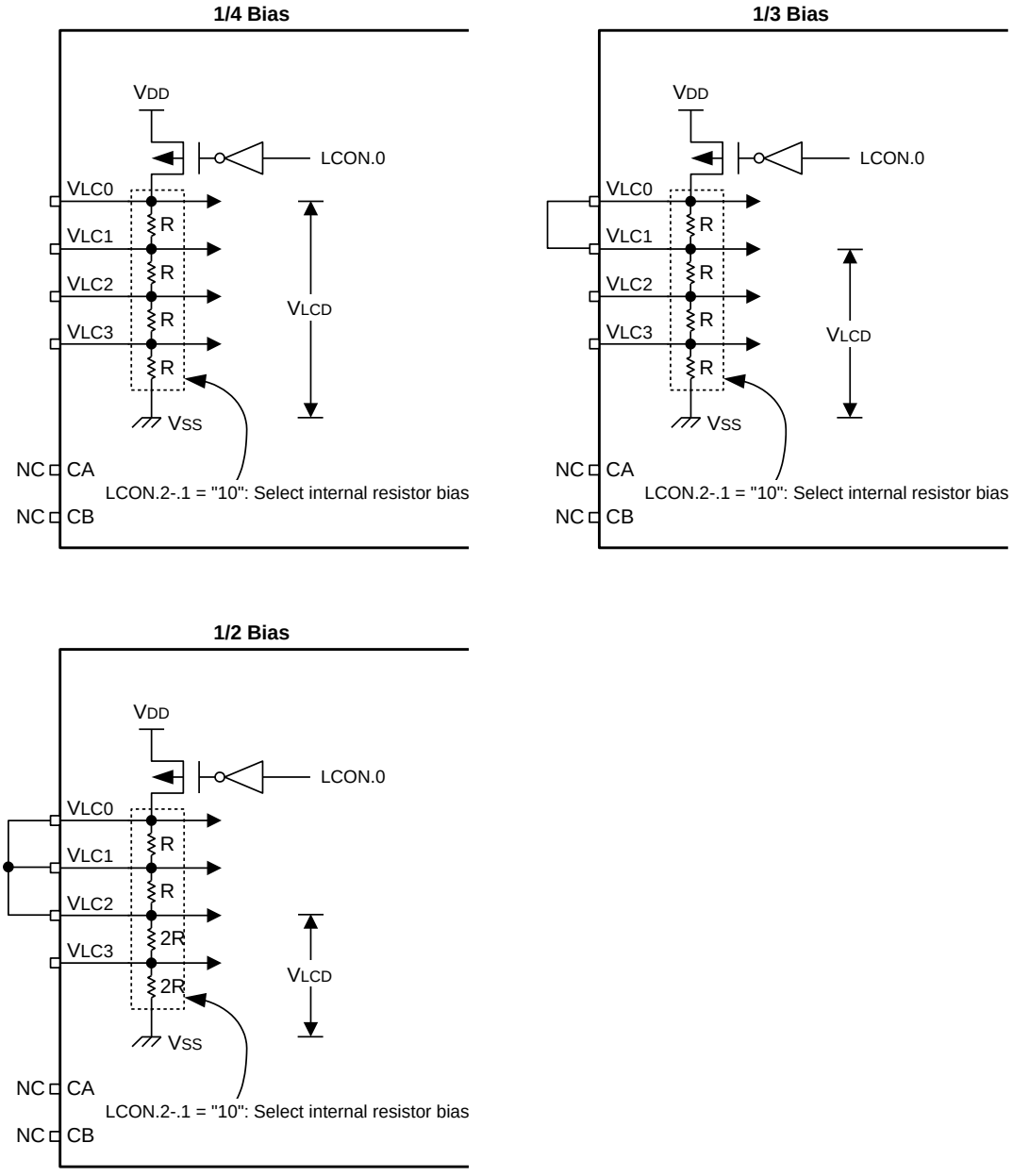


	1/4 bias	1/3 bias	1/2 bias
000	3.6V	3.150V	2.20V
001	3.8V	3.375V	2.40V
010	4.0V	3.600V	2.60V*
011	4.2V	3.825V	2.80V*
100	4.4V	4.050V*	3.00V*
101	4.6V	4.275V*	Not Available
110	4.8V	4.500V*	Not Available
111	5.0V	Not Available	Not Available

NOTES:
1. The voltage regulator and booster circuit provide constant LCD contrast level.
2. Since the booster clock is generated by watch timer clock (fw). And the booster clock is almost 32kHz.
3. For ½ bias, this value is valid under Run, Main/Sub Idle modes.
4. * Indicates mode is not available in Sub Idle and Stop modes.

Figure 15-5 LCD Mode Control Register (LMOD)

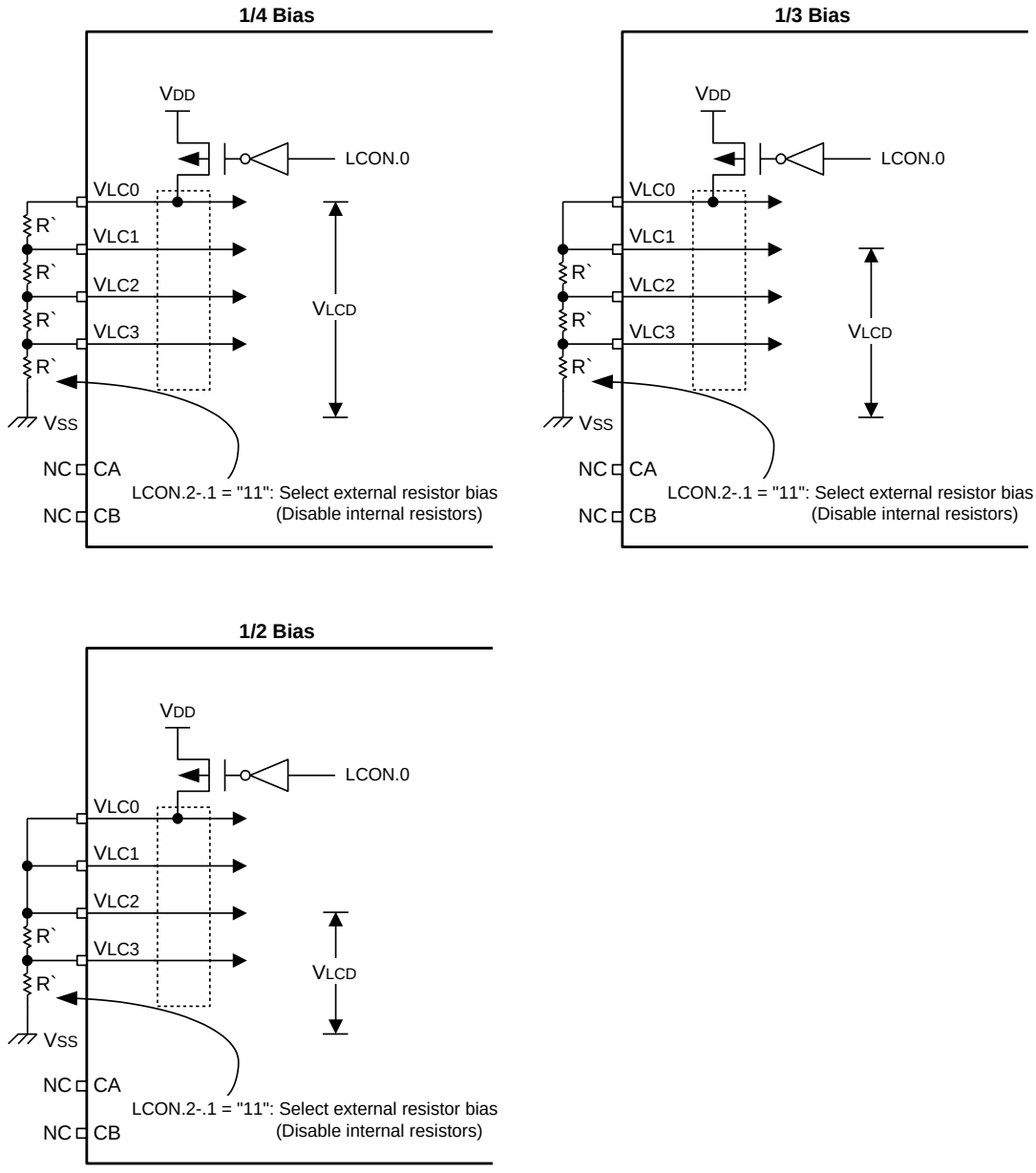
15.6 Internal Resistor Bias Pin Connection



NOTES:
1. The CA and CB pins are not connected when the internal resistor bias is selected.
2. V_{LC0} and V_{LC1} should be connected at 1/3 bias.
3. V_{LC0}, V_{LC1}, and V_{LC2} should be connected at 1/2 bias.

Figure 15-6 Internal Resistor Bias Pin Connection

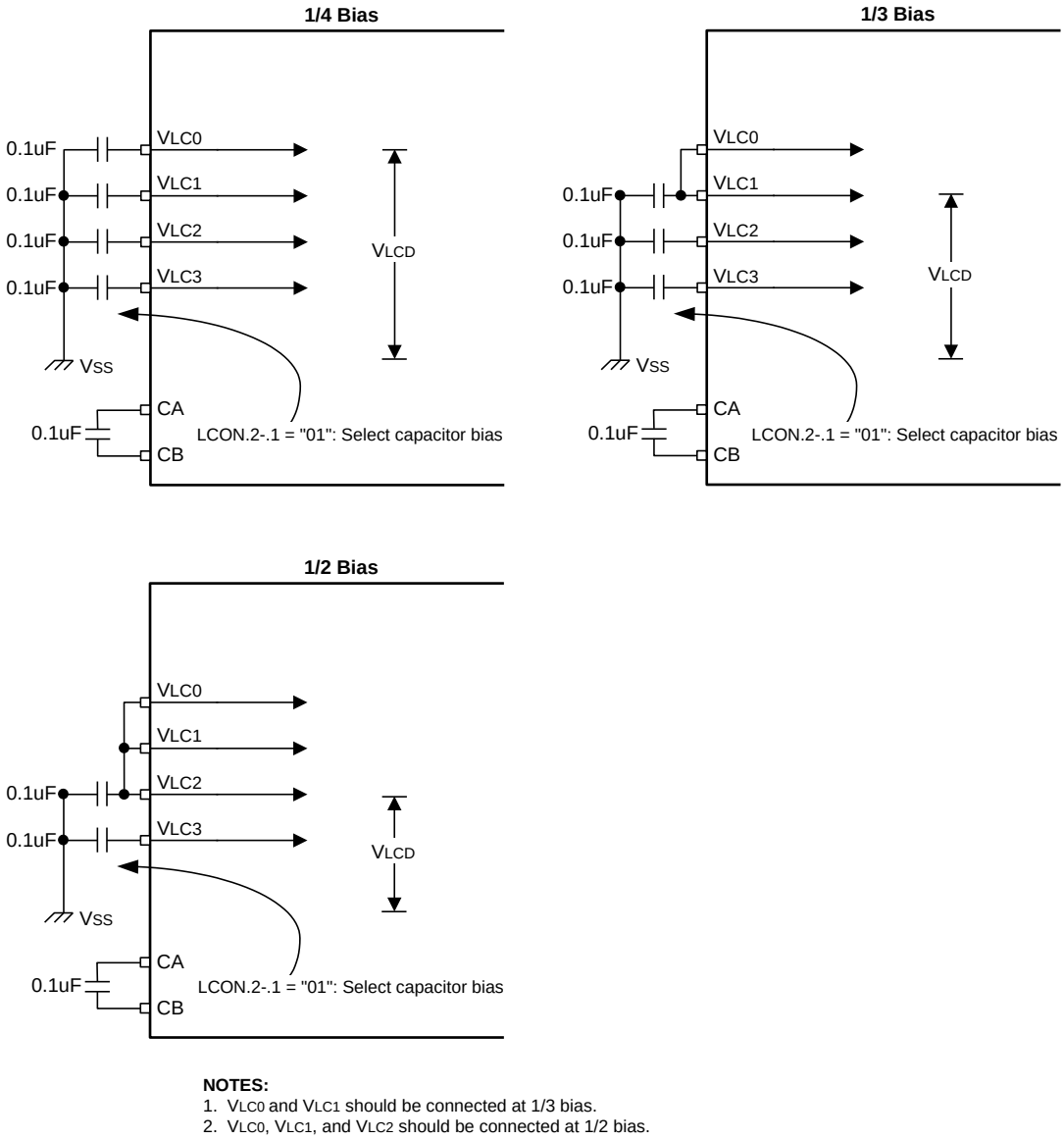
15.7 External Resistor Bias Pin Connection



NOTES:
1. The CA and CB pins are not connected when the external resistor bias is selected.
2. V_{LC0} and V_{LC1} should be connected at 1/3 bias.
3. V_{LC0}, V_{LC1}, and V_{LC2} should be connected at 1/2 bias.

Figure 15-7 External Resistor Bias Pin Connection

15.8 Capacitor Bias Pin Connection



NOTES:
1. V_{LC0} and V_{LC1} should be connected at 1/3 bias.
2. V_{LC0}, V_{LC1}, and V_{LC2} should be connected at 1/2 bias.

Figure 15-8 Capacitor Bias Pin Connection

15.9 Common (COM) Signals

The common signal output pin selection (COM pin selection) varies according to the selected duty cycle.

- In 1/8 duty mode, COM0-COM7 (SEG6–SEG33) pins are selected.
- In 1/4 duty mode, COM0-COM3 (SEG2–SEG33) pins are selected.
- In 1/3 duty mode, COM0-COM2 (SEG1–SEG33) pins are selected.
- In 1/2 duty mode, COM0-COM1 (SEG0–SEG33) pins are selected.

15.10 Segment (SEG) Signals

The 34 LCD segment signal pins are connected to corresponding display RAM locations at page 8. Bits of the display RAM are synchronized with the common signal output pins.

When the bit value of a display RAM location is "1", a select signal is sent to the corresponding segment pin. When the display bit is "0", a 'no-select' signal to the corresponding segment pin.

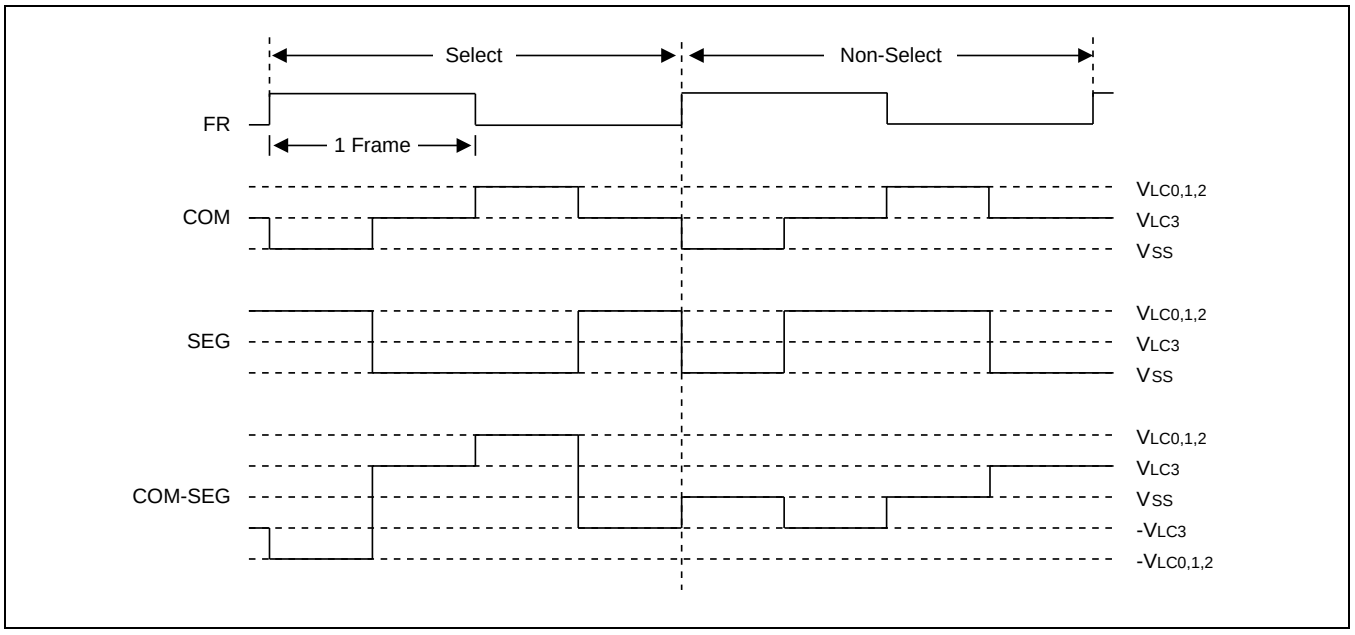


Figure 15-9 Select/No-Select Signal in 1/2 Duty, 1/2 Bias Display Mode

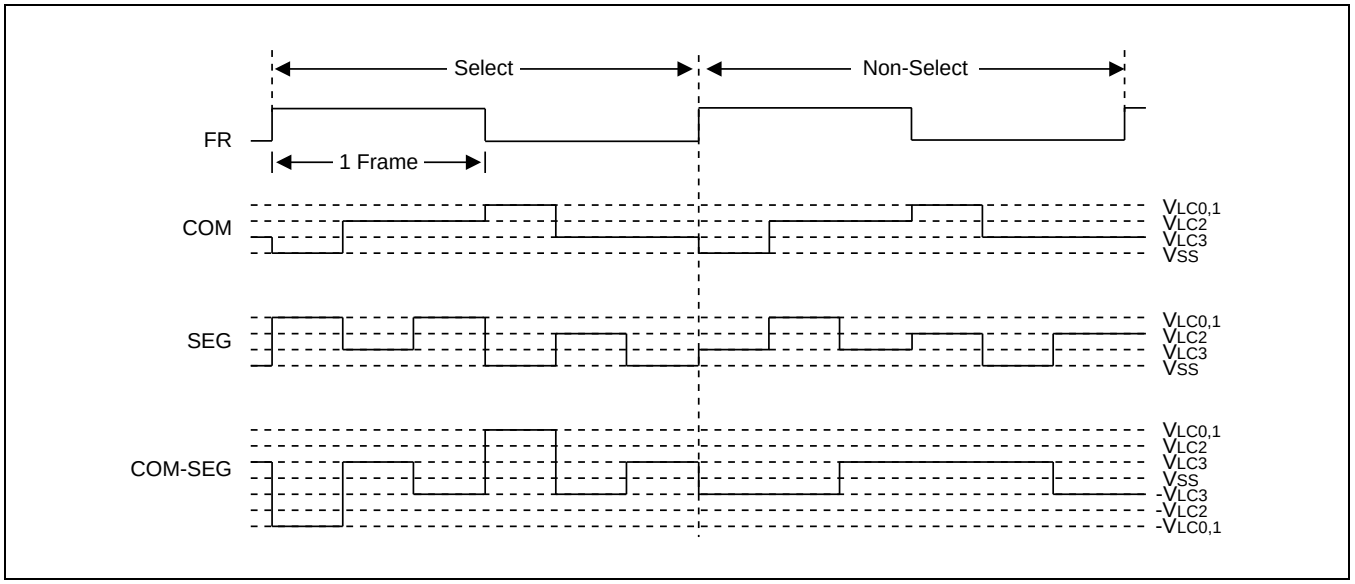


Figure 15-10 Select/No-Select Signal in 1/3 Duty, 1/3 Bias Display Mode

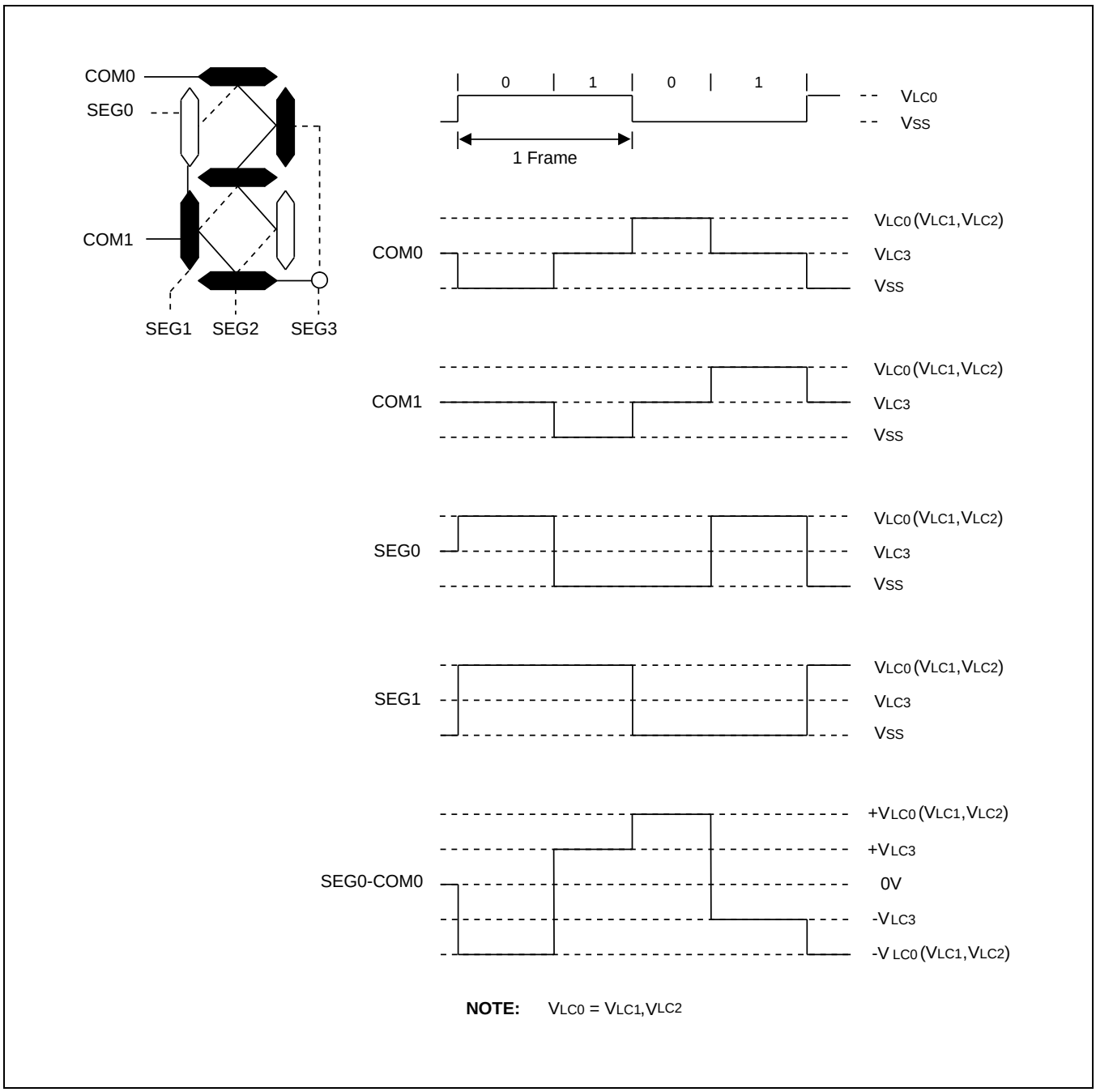


Figure 15-11 LCD Signal Waveforms (1/2 Duty, 1/2 Bias)

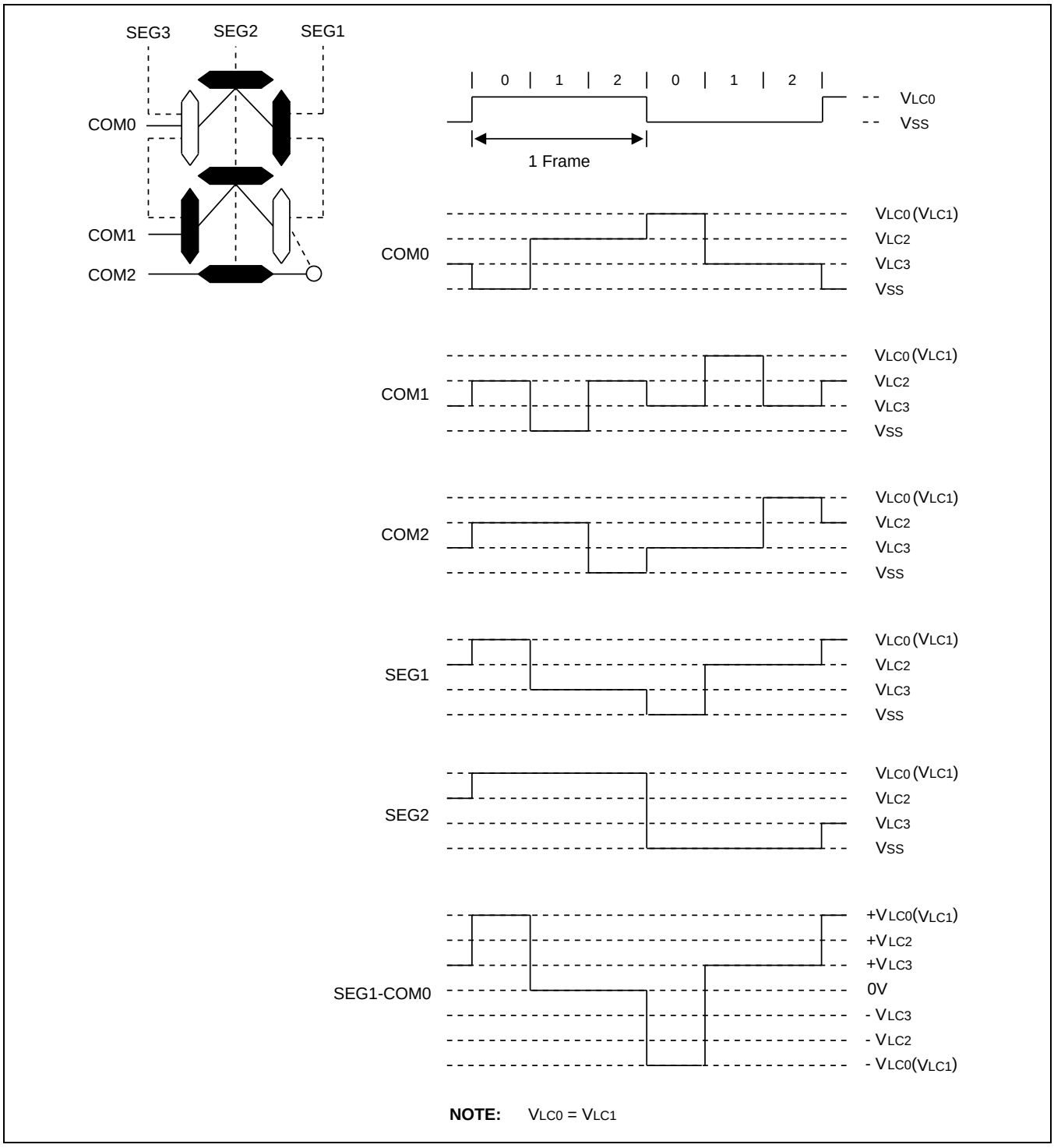


Figure 15-12 LCD Signal Waveforms (1/3 Duty, 1/3 Bias)

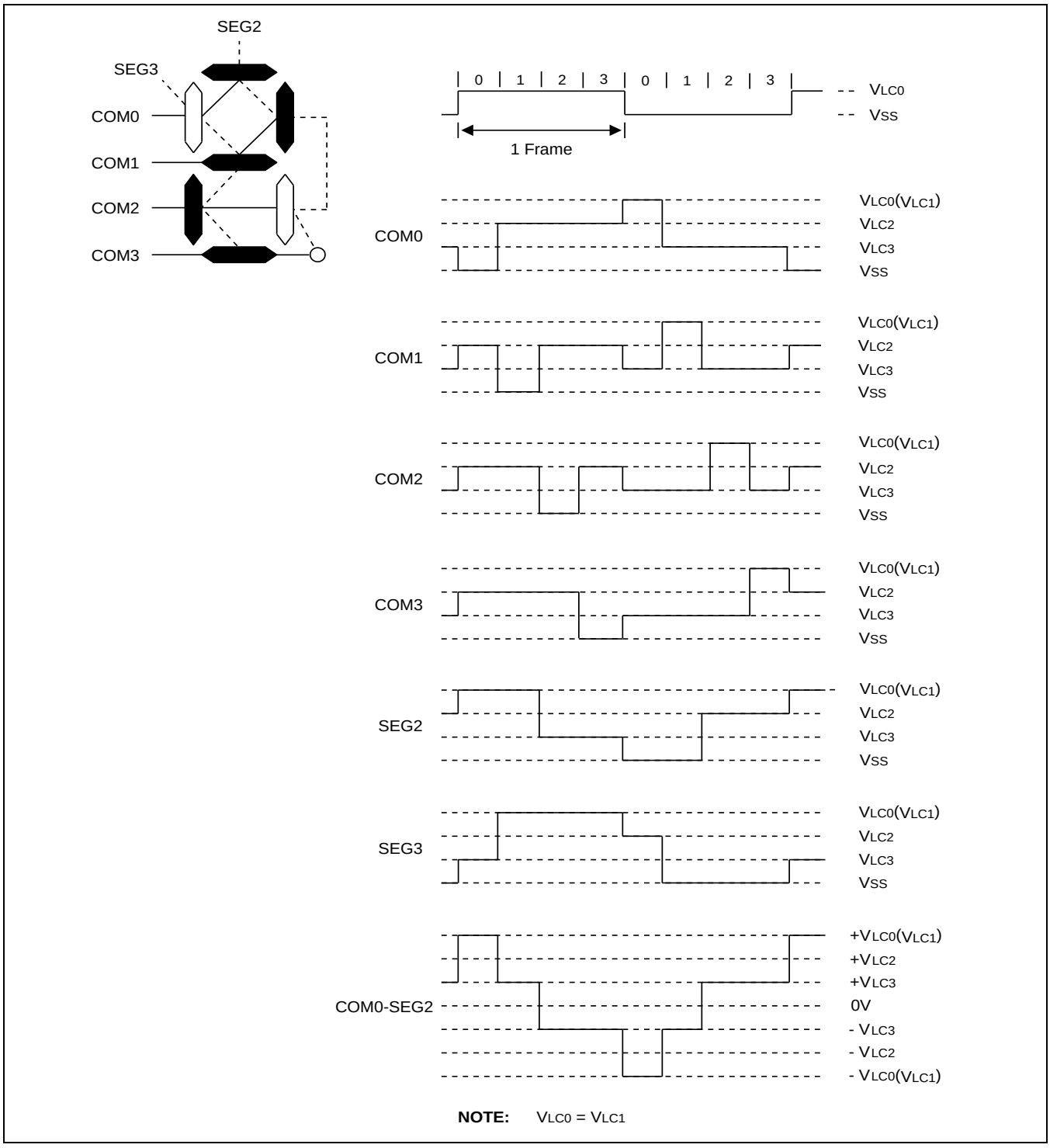
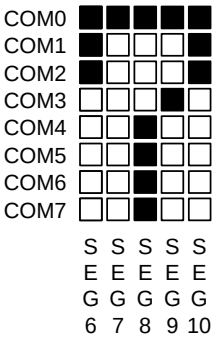
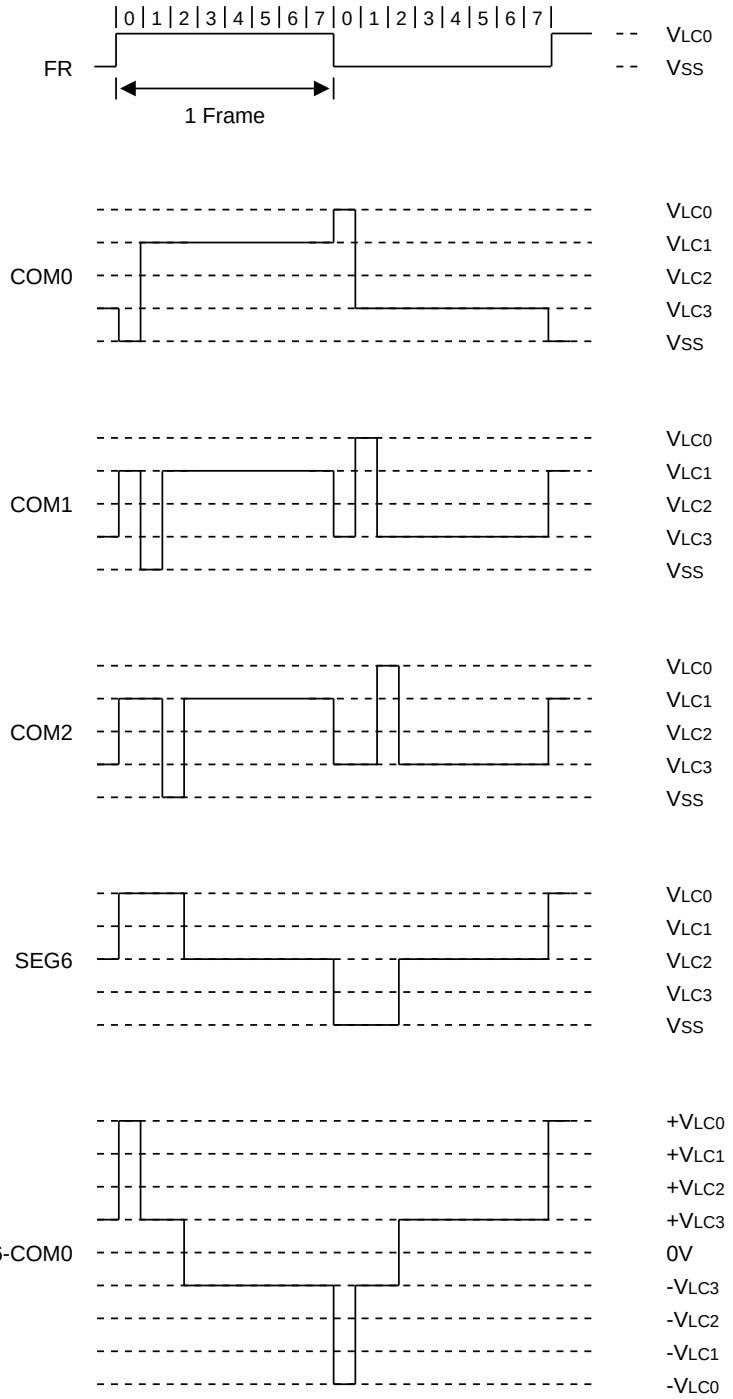


Figure 15-13 LCD Signal Waveforms (1/4 Duty, 1/3 Bias)


COM0
COM1
COM2
COM3
COM4
COM5
COM6
COM7
SEG6 SEG7 SEG8 SEG9 SEG10


0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
FR
1 Frame
VLC0
VSS
COM0
VLC1
VLC2
VLC3
COM1
COM2
SEG6
SEG6-COM0
+VLC0
+VLC1
+VLC2
+VLC3
0V
-VLC3
-VLC2
-VLC1
-VLC0

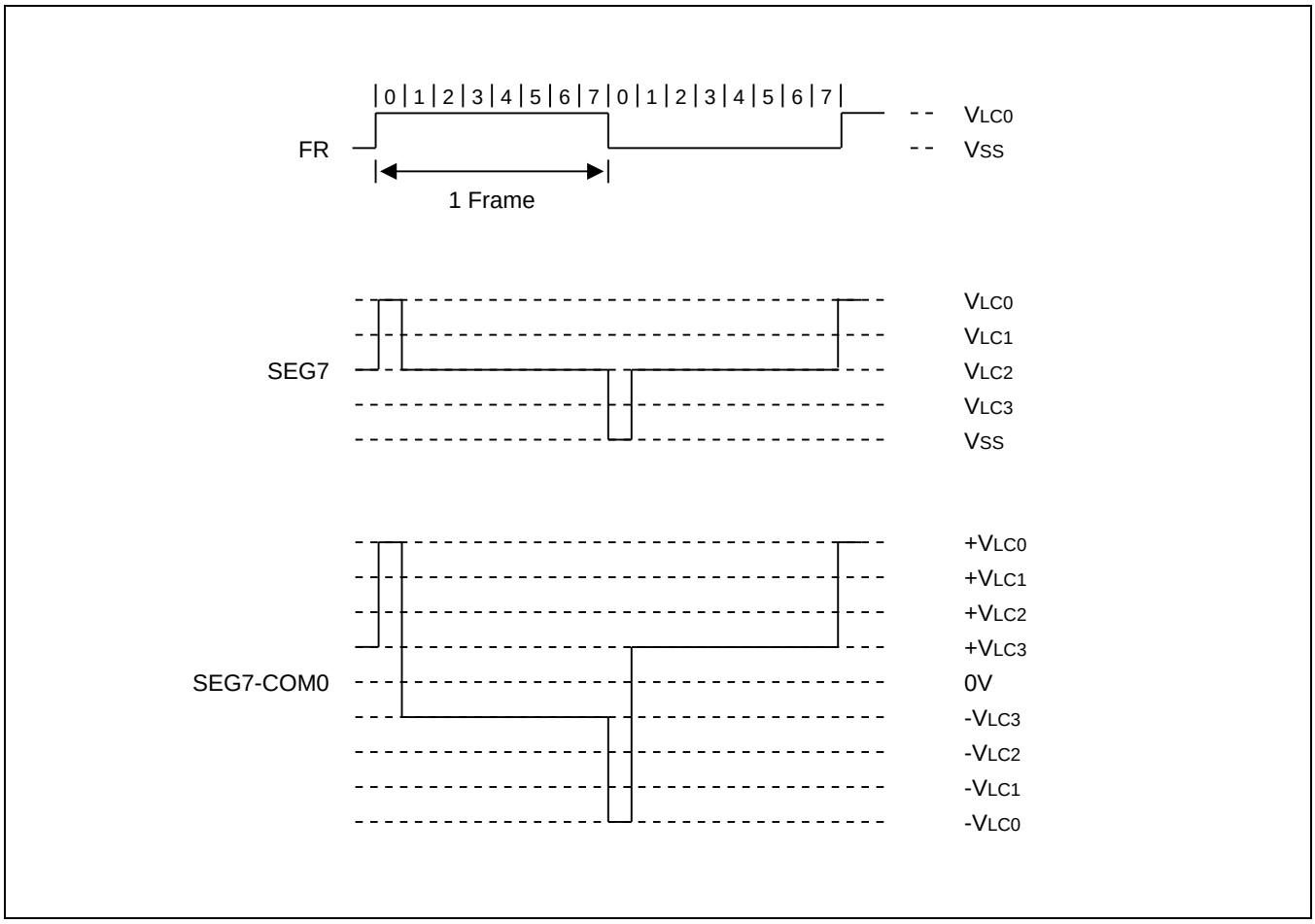


Figure 15-14 LCD Signal Waveforms (1/8 Duty, 1/4 Bias)

Example 15-1 LCD Display on, After Capacitor Bias Selected

This example shows how to display on after capacitor bias selected:

```
                    .
                    .
                    .

LCD_cap_bias
              AND     LCON,#0FBH
              OR      LCON,#02H       ; Capacitor bias select
              CALL    Delay1mS        ; Delay 1ms
              OR      LCON,#01H       ; Turn on LCD display
              RET
                    .
                    .
                    .

Delay1mS      LD      R0,#20H
DEL   NOP
              DJNZ    R0,DEL
              RET
```

16 10-Bit Analog-To-Digital Converter

16.1 Overview

The 10-bit A/D converter (ADC) module uses successive approximation logic to convert analog levels entering at one of the eight input channels to equivalent 10-bit digital values. The analog input level must lie between the AV_{REF} and AV_{SS} values. The A/D converter has the following components:

- Analog comparator with successive approximation logic
- D/A converter logic (resistor string type)
- ADC control register (ADCON)
- Eight multiplexed analog data input pins (AD0–AD7)
- 10-bit A/D conversion data output register (ADDATAH/L)
- 8-bit digital input port (Alternately, I/O port.)
- AV_{REF} and AV_{SS} pins, AV_{SS} is internally connected to V_{SS}

16.2 Function Description

To initiate an analog-to-digital conversion procedure, at the first you must set ADCEN signal for ADC input enable at port 4, the pin set with alternative function can be used for ADC analog input. And you write the channel selection data in the A/D converter control register ADCON.4–.6 to select one of the eight analog input pins (AD0–7) and set the conversion start or disable bit, ADCON.0. The read-write ADCON register is located in set 1, bank 0 at address D2H. The pins which are not used for ADC can be used for normal I/O.

During a normal conversion, ADC logic initially sets the successive approximation register to 200H (the approximate half-way point of an 10-bit register). This register is then updated automatically during each conversion step. The successive approximation block performs 10-bit conversions for one input channel at a time. You can dynamically select different channels by manipulating the channel selection bit value (ADCON.6–.4) in the ADCON register. To start the A/D conversion, you should set the start bit, ADCON.0. When a conversion is completed, ADCON.3, the end-of-conversion (EOC) bit is automatically set to 1 and the result is dumped into the ADDATAH/L register where it can be read. The A/D converter then enters an Idle state. Remember to read the contents of ADDATAH/L before another conversion starts. Otherwise, the previous result will be overwritten by the next conversion result.

NOTE: Because the A/D converter has no sample-and-hold circuitry, it is very important that fluctuation in the analog level at the AD0–AD7 input pins during a conversion procedure be kept to an absolute minimum. Any change in the input level, perhaps due to noise, will invalidate the result. If the chip enters to STOP or IDLE mode in conversion process, there will be a leakage current path in A/D block. You must use STOP or IDLE mode after ADC operation is finished.

16.3 Conversion Timing

The A/D conversion process requires 4 steps (4 clock edges) to convert each bit and 10 clocks to set-up A/D conversion. Therefore, total of 50 clocks are required to complete an 10-bit conversion: When fxx/8 is selected for conversion clock with an 8 MHz fxx clock frequency, one clock cycle is 1 us. Each bit conversion requires 4 clocks, the conversion rate is calculated as follows:

4 clocks/bit × 10 bits + set-up time = 50 clocks, 50 clock × 1 μs = 50 μs at 1 MHz

16.4 A/D Converter Control Register (ADCON)

The A/D converter control register, ADCON, is located at address D2H in set 1, bank 0. It has three functions:

- Analog input pin selection (ADCON.6–.4)
- End-of-conversion status detection (ADCON.3)
- ADC clock selection (ADCON.2–.1)
- A/D operation start or disable (ADCON.0)

After a reset, the start bit is turned off. You can select only one analog input channel at a time. Other analog input pins (AD0–AD7) can be selected dynamically by manipulating the ADCON.4–6 bits. And the pins not used for analog input can be used for normal I/O function.

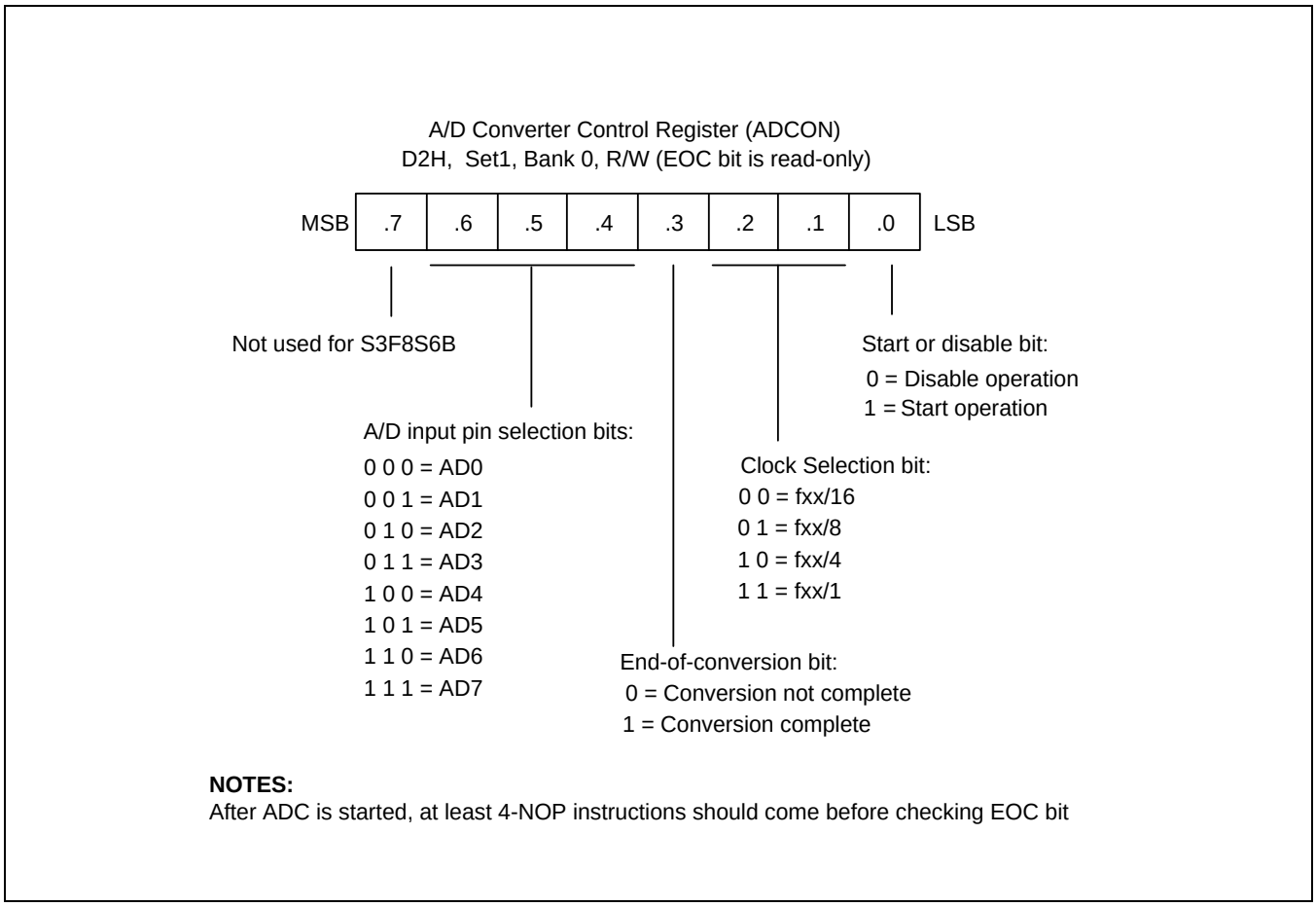


Figure 16-1 A/D Converter Control Register (ADCON)

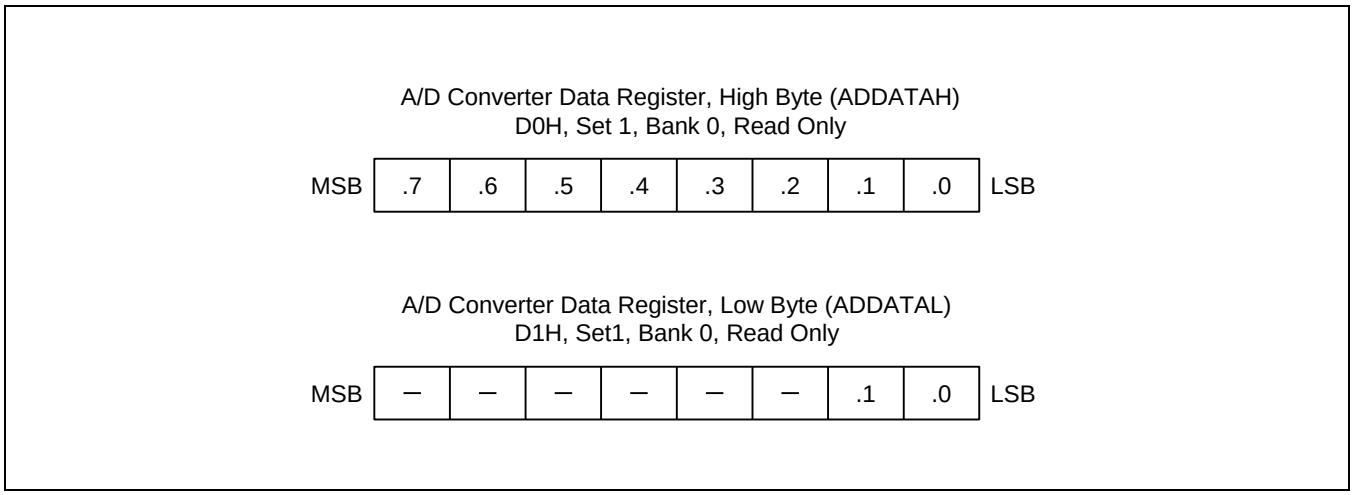


Figure 16-2 A/D Converter Data Register (ADDATAH/L)

16.5 Internal Reference Voltage Levels

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

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

16.6 Block Diagram

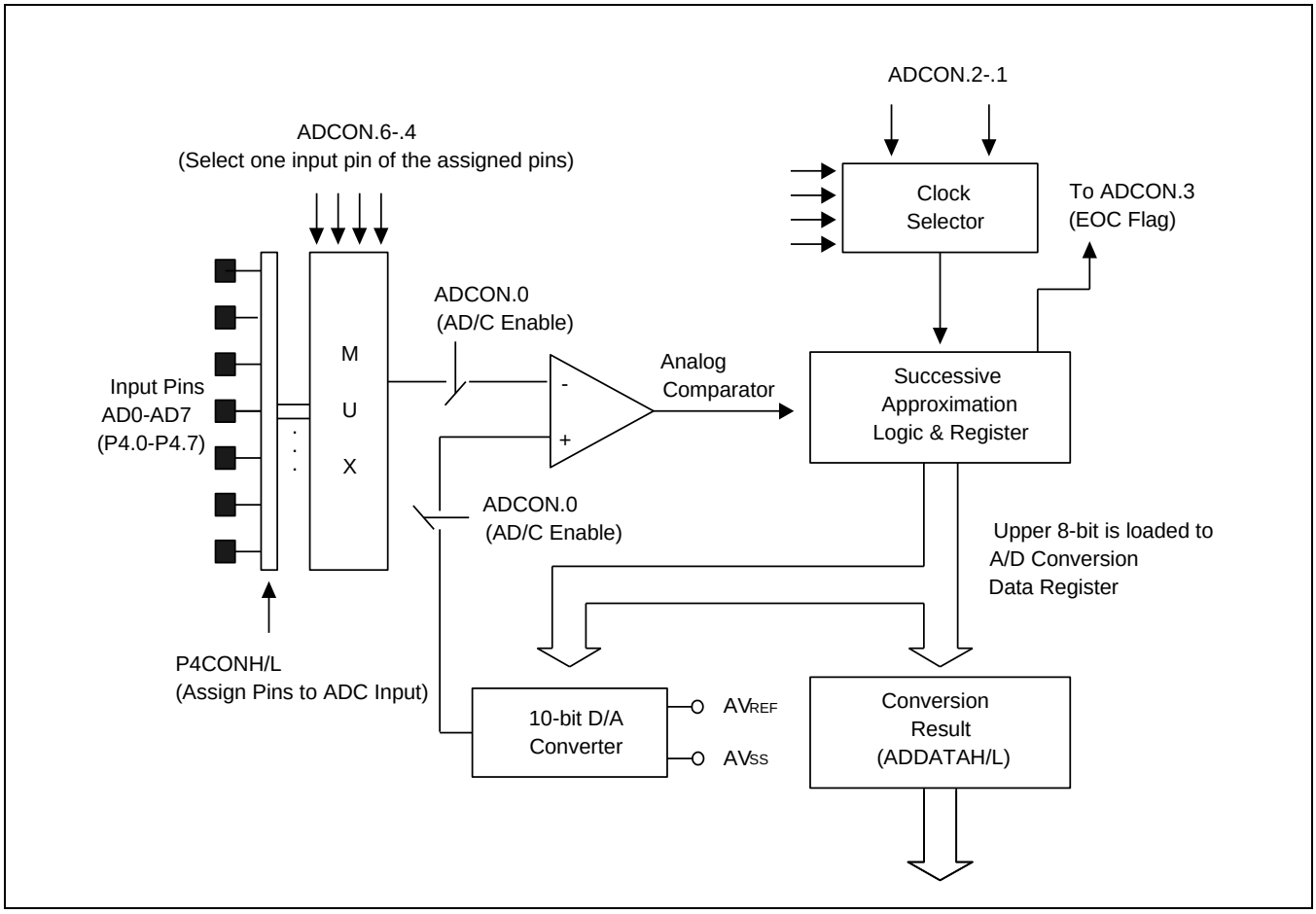


Figure 16-3 A/D Converter Functional Block Diagram

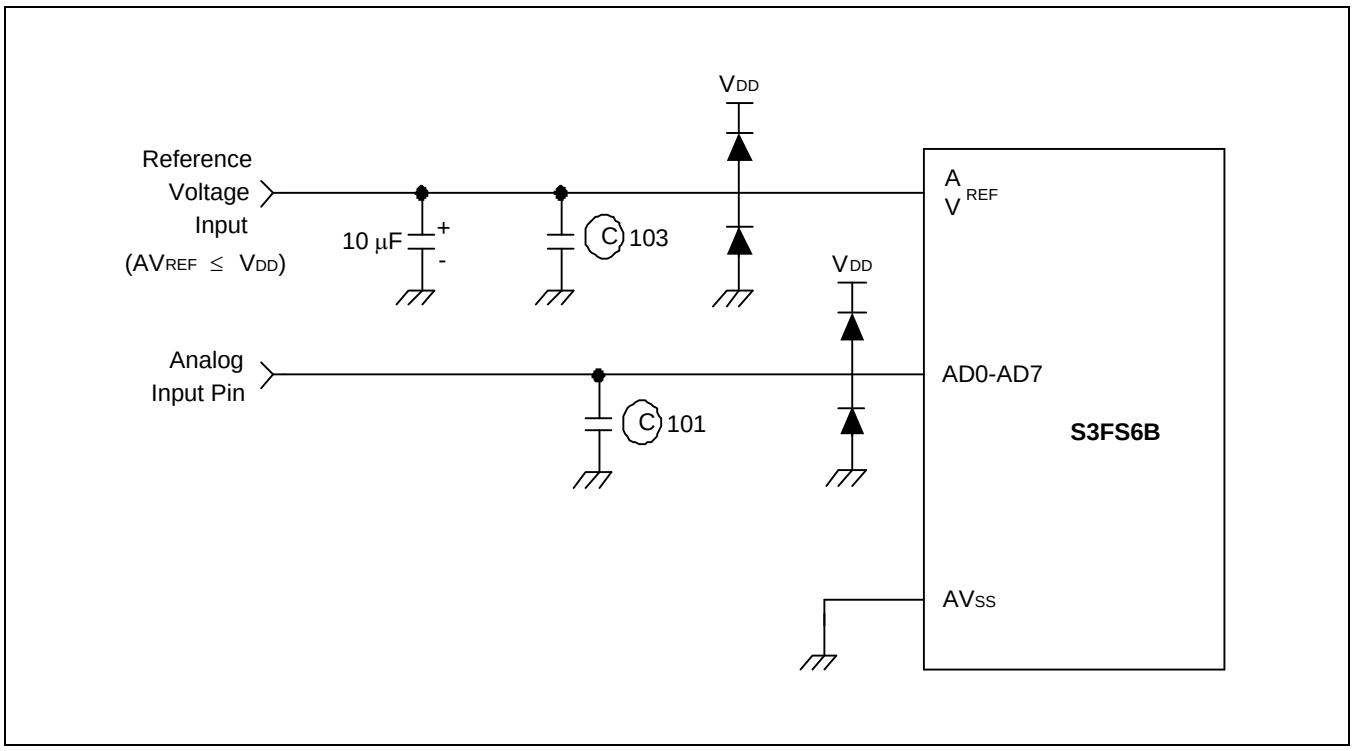


Figure 16-4 Recommended A/D Converter Circuit for Highest Absolute Accuracy

17 Serial I/O Interface

17.1 Overview

Serial I/O module, SIO can interface with various types of external device that require serial data transfer.

The components of each SIO function block are:

- 8-bit control register (SIOCON)
- Clock selector logic
- 8-bit data buffer (SIODATA)
- 8-bit pre-scaler (SIOPS)
- 3-bit serial clock counter
- Serial data I/O pins (SI, SO)
- Serial clock input/output pins (SCK)

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

17.2 Programming Procedure

To program the SIO modules, follow these basic steps:

1. Configure the I/O pins at port (SO, SCK, SI) by loading the appropriate value to the P6CONL register if necessary.
2. Load an 8-bit value to the SIOCON control register to properly configure the serial I/O module. In this operation, SIOCON.2 must be set to "1" to enable the data shifter.
3. For interrupt generation, set the serial I/O interrupt enable bit (SIOCON.1) to "1".
4. When you transmit data to the serial buffer, write data to SIODATA and set SIOCON.3 to 1, the shift operation starts.
5. When the shift operation (transmit/receive) is completed, the SIO pending bit (SIOCON.0) is set to "1" and an SIO interrupt request is generated.

17.3 SIO Control Register (SIOCON)

The control register for serial I/O interface module, SIOCON, is located at E7H in set 1, bank 0. It has the control settings for SIO module.

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

A reset clears the SIOCON value to "00H". This configures the corresponding module with an internal clock source at the SCK, selects receive-only operating mode, and clears the 3-bit counter. The data shift operation and the interrupt are disabled. The selected data direction is MSB-first.

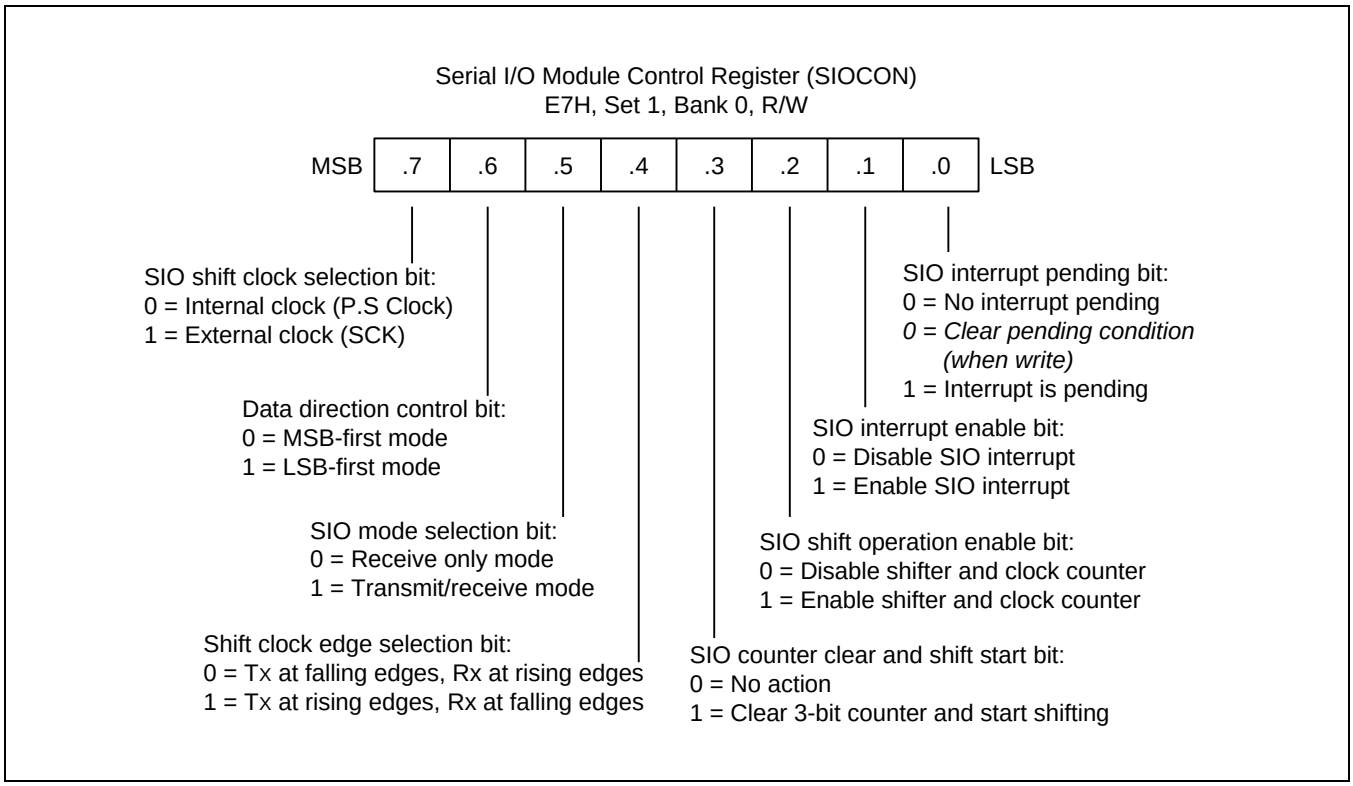


Figure 17-1 Serial I/O Module Control Registers (SIOCON)

17.4 SIO Pre-Scaler Register (SIOPS)

The control register for serial I/O interface module, SIOPS, is located at E9H in set 1, bank 0.

The value stored in the SIO pre-scaler register, SIOPS, lets you determine the SIO clock rate (baud rate) as follows:

Baud rate = Input clock (fxx/4)/(Pre-scaler value + 1), or SCK input clock, where the input clock is fxx/4

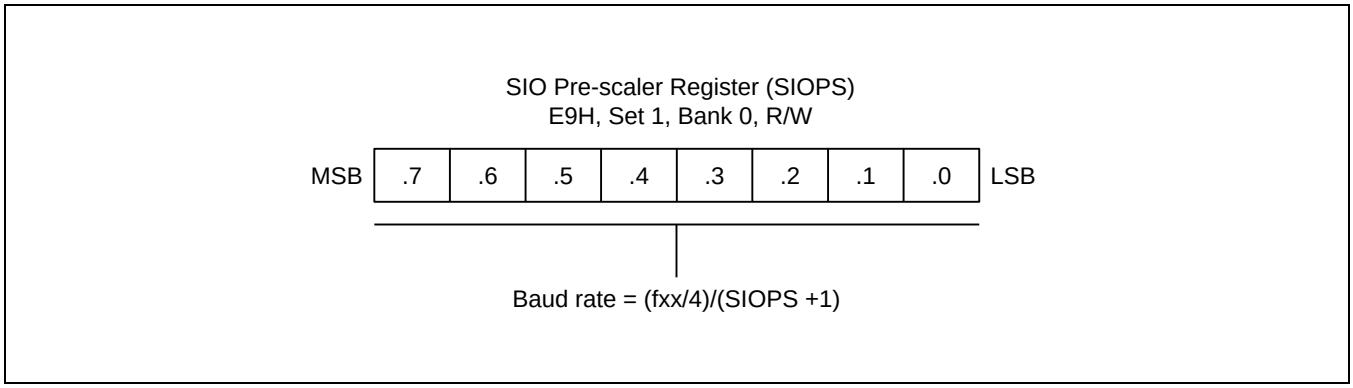


Figure 17-2 SIO Pre-scaler Register (SIOPS)

17.5 Block Diagram

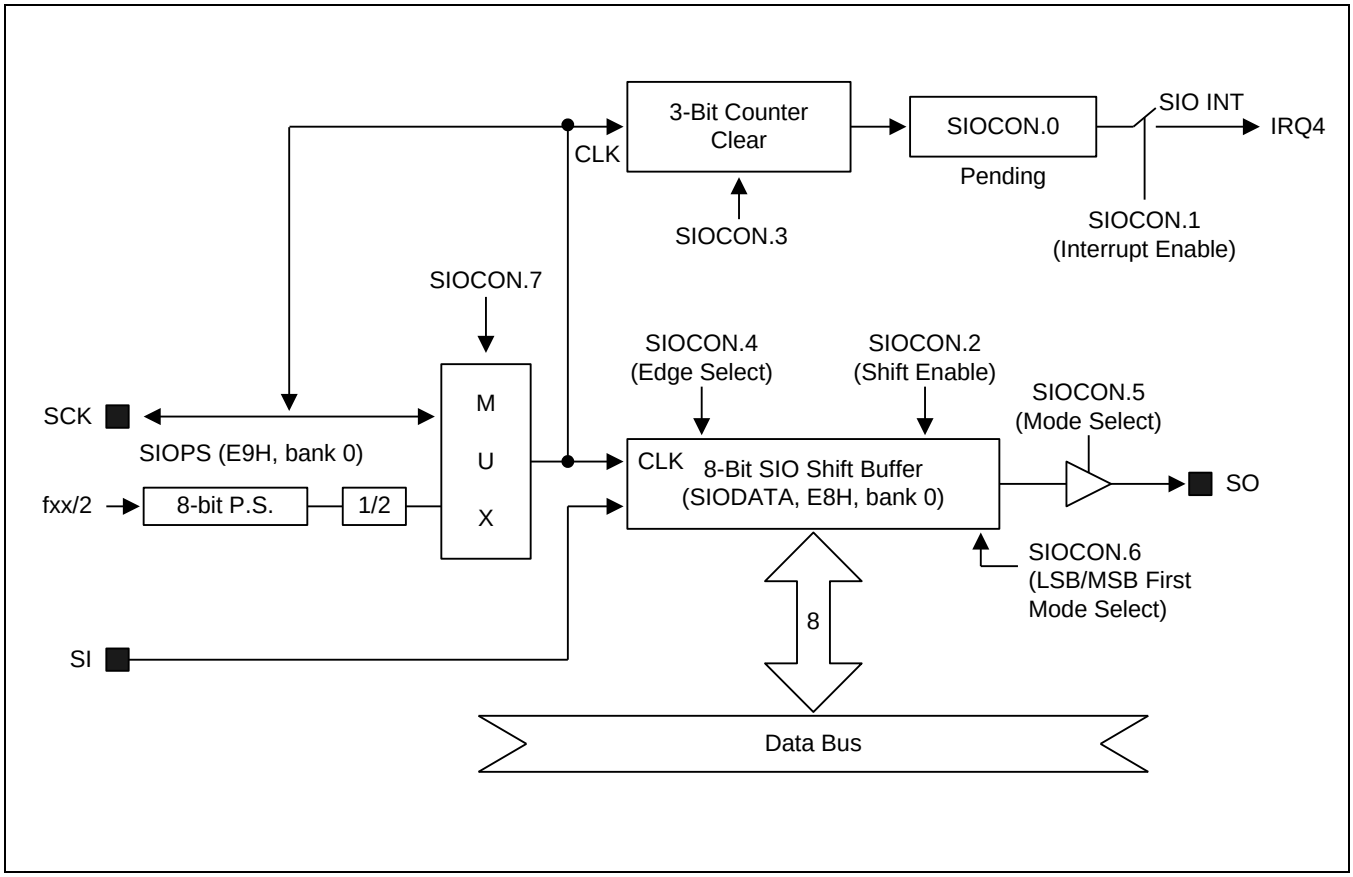


Figure 17-3 SIO Functional Block Diagram

17.6 Serial I/O Timing Diagram

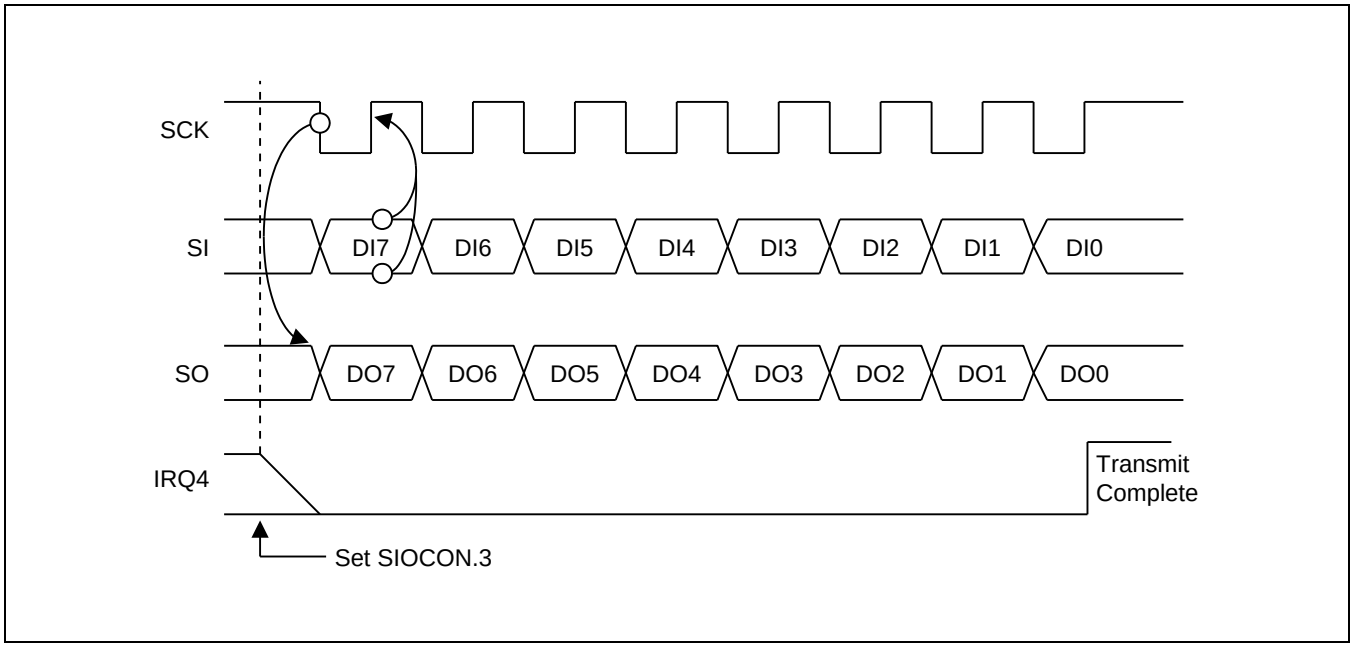


Figure 17-4 Serial I/O Timing in Transmit/Receive Mode (Tx at falling, SIOCON.4 = 0)

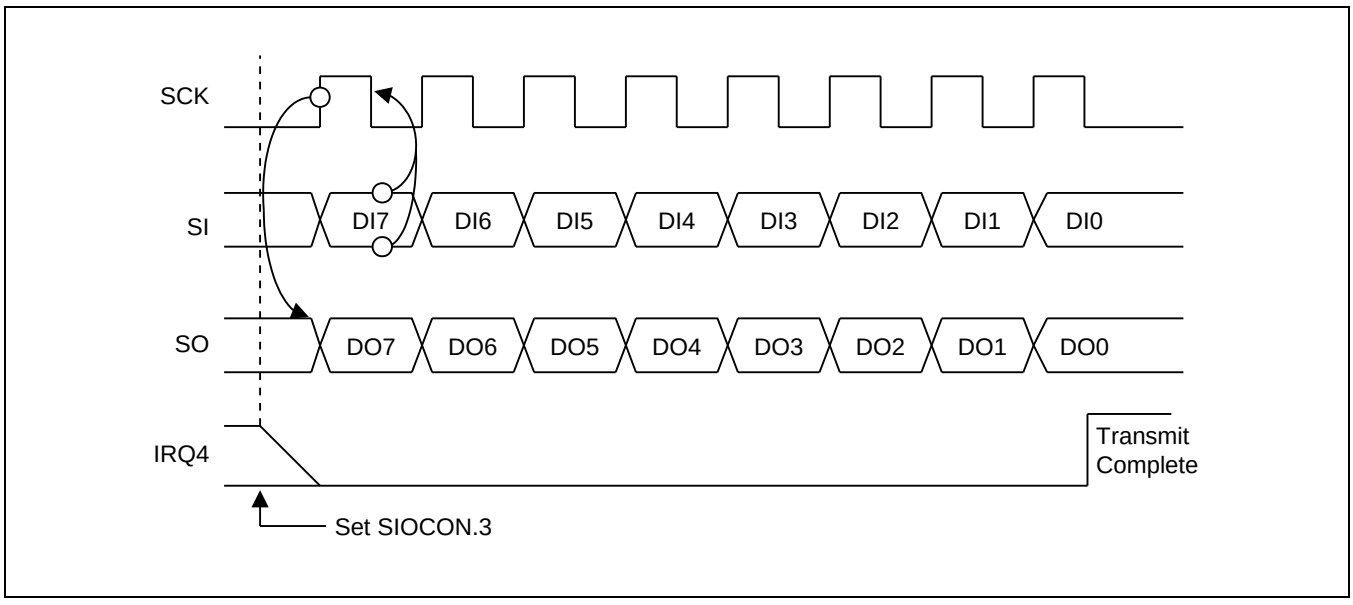


Figure 17-5 Serial I/O Timing in Transmit/Receive Mode (Tx at rising, SIOCON.4 = 1)

18 UART 0

18.1 Overview

The UART 0 block has a full-duplex serial port with programmable operating modes:

There is one synchronous mode and three UART (Universal Asynchronous Receiver/Transmitter) modes:

- Serial I/O with baud rate of $f_U/(16 \times (BRDATA0 + 1))$
- 8-bit UART mode; variable baud rate
- 9-bit UART mode; $f_U/16$
- 9-bit UART mode, variable baud rate

UART 0 receive and transmit buffers are both accessed via the data register, UDATA0, is page 8 at address 16H. Writing to the UART data register loads the transmit buffer; reading the UART data register accesses a physically separate receive buffer.

When accessing a receive data buffer (shift register), reception of the next byte can begin before the previously received byte has been read from the receive register. However, if the first byte has not been read by the time the next byte has been completely received, one of the bytes will be lost.

In all operating modes, transmission is started when any instruction (usually a write operation) uses the UDATA0 register as its destination address. In mode 0, serial data reception starts when the receive interrupt pending bit (UART0CONH.0) is "0" and the receive enable bit (UART0CONH.4) is "1". In mode 1, 2, and 3, reception starts whenever an incoming start bit ("0") is received and the receive enable bit (UART0CONH.4) is set to "1".

18.2 Programming Procedure

To program the UART 0 modules, follow these basic steps:

1. Configure P1.2 and P1.3 to alternative function (RxD0 (P1.2), TxD0 (P1.3)) for UART module by setting the P1CONL register to appropriately value.
2. Load an 8-bit value to the UART0CONH/L control register to properly configure the UART I/O module.
3. For interrupt generation, set the UART 0 I/O interrupt enable bit (UART0CONH.1 or UART0CONL.1) to "1".
4. When you transmit data to the UART 0 buffer, write data to UDATA0, the shift operation starts.
5. When the shift operation (receive/transmit) is completed, UART 0 pending bit (UART0CONH.0 or UART0CONL.0) is set to "1" and an UART 0 interrupt request is generated.

18.3 UART 0 High-Byte Control Register (UART0CONh)

The control register for the UART 0 is called UART0CONH in page 8 at address 14H.

It has the following control functions:

- Operating mode and baud rate selection
- Multiprocessor communication and interrupt control
- Serial receive enable/disable control
- 9^{th} data bit location for transmit and receive operations (modes 2 and 3 only)
- UART 0 receive interrupt control

A reset clears the UART0CONH value to "00H". So, if you want to use UART 0 module, you must write appropriate value to UART0CONH.

18.4 UART 0 Low-Byte Control Register (UART0CONl)

The control register for the UART 0 is called UART0CONL in page 8 at address 15H.

It has the following control functions:

- UART 0 transmit and receive parity-bit control
- UART 0 clock selection
- UART 0 transmit interrupt control

A reset clears the UART0CONL value to "00H". So, if you want to use UART 0 module, you must write appropriate value to UART0CONL.

UART 0 Control Register, High Byte (UART0CONH)
14H, Page 8, R/W

Operating mode and baud rate selection bits: (see table below)

Multiprocessor communication(1) enable bit (for modes 2 and 3 only):
0 = Disable
1 = Enable

Serial data receive enable bit:
0 = Disable
1 = Enable

TB8(3) (Only when UART0CONL.7 = 0):
Location of the 9th data bit to be transmitted in UART 0 mode 2 or 3 ("0" or "1")

RB8(3) (Only when UART0CONL.7 = 0):
Location of the 9th data bit that was received in UART 0 mode 2 or 3 ("0" or "1")

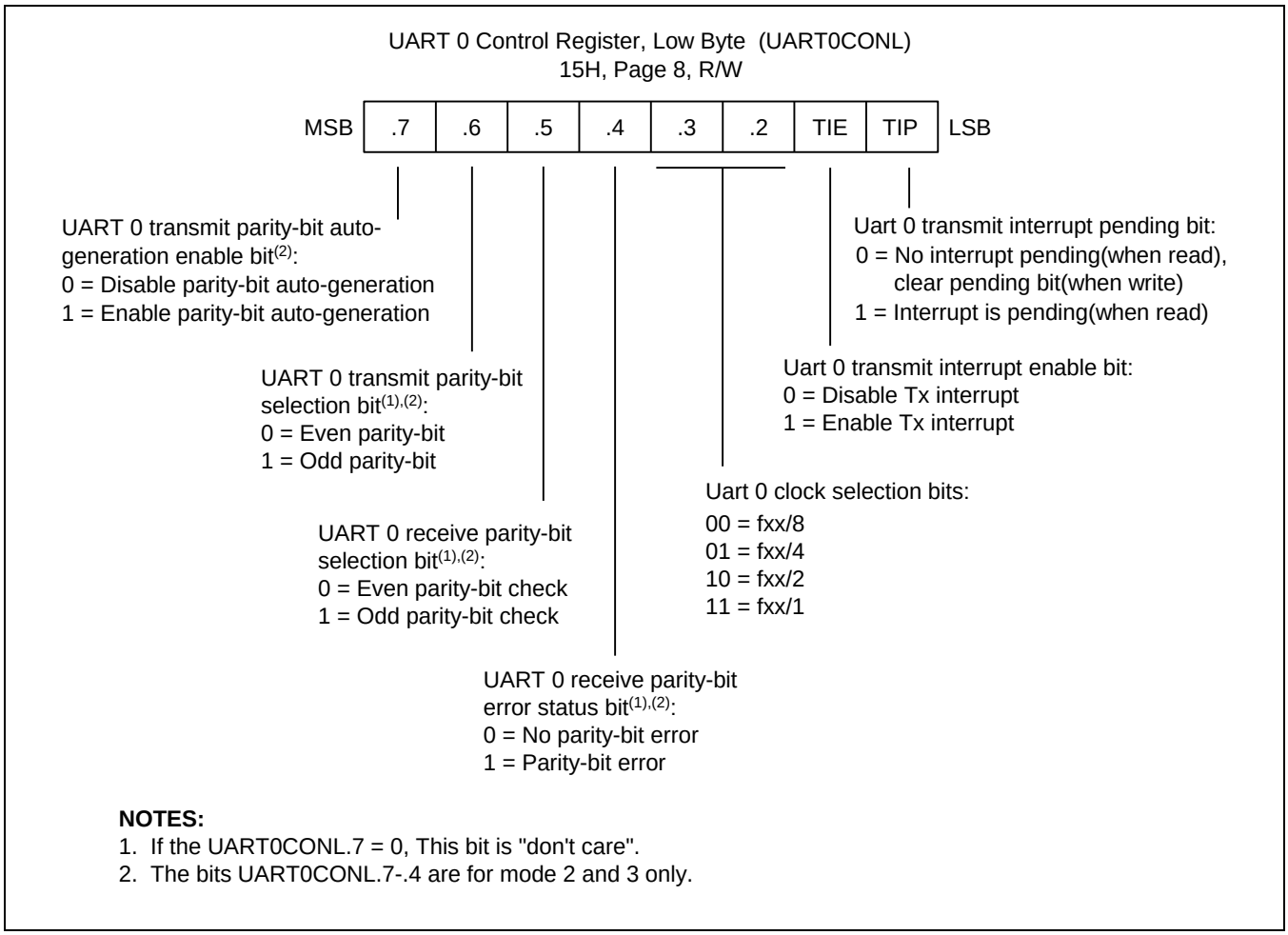
Uart 0 receive interrupt enable bit:
0 = Disable Rx interrupt
1 = Enable Rx interrupt

Uart 0 receive interrupt pending bit:
0 = No interrupt pending(when read), clear pending bit(when write)
1 = Interrupt is pending(when read)

MS1	MS0	Mode	Description(2)	Baud Rate
0	0	0	Shift register	(fu/(16 x (BRDATA0 + 1)))
0	1	1	8-bit UART	(fu/(16 x (BRDATA0 + 1)))
1	0	2	9-bit UART	(fu /16)
1	1	3	9-bit UART	(fu/(16 x (BRDATA0 + 1)))

NOTES:
1. In mode 2 or 3, if the UART0CONH.5 bit is set to "1" then the receive interrupt will not be activated if the received 9th data bit is "0". In mode 1, if UART0CONH.5 = "1" then the receive interrut will not be activated if a valid stop bit was not received. In mode 0, the UART0CONH.5 bit should be "0"
2. The descriptions for 8-bit and 9-bit UART mode do not include start and stop bits for serial data receive and transmit.
3. If the UART0CONL.7 = 1, This bit is "don't care".

Figure 18-1 UART 0 High Byte Control Register (UART0CONH)



NOTES:
1. If the UART0CONL.7 = 0, This bit is "don't care".
2. The bits UART0CONL.7-.4 are for mode 2 and 3 only.

Figure 18-2 UART 0 Low Byte Control Register (UART0CONL)

18.5 UART 0 Interrupt Pending bits

In mode 0, the receive interrupt pending bit UART0CONH.0 is set to "1" when the 8th receive data bit has been shifted. In mode 1, the UART0CONH.0 bit is set to "1" at the halfway point of the stop bit's shift time. In mode 2, or 3, the UART0CONH.0 bit is set to "1" at the halfway point of the RB8 bit's shift time. When the CPU has acknowledged the receive interrupt pending condition, the UART0CONH.0 bit must then be cleared by software in the interrupt service routine.

In mode 0, the transmit interrupt pending bit UART0CONL.0 is set to "1" when the 8th transmit data bit has been shifted. In mode 1, 2, or 3, the UART0CONL.0 bit is set at the start of the stop bit. When the CPU has acknowledged the transmit interrupt pending condition, the UART0CONL.0 bit must then be cleared by software in the interrupt service routine.

18.6 UART 0 Data Register (UDATA0)

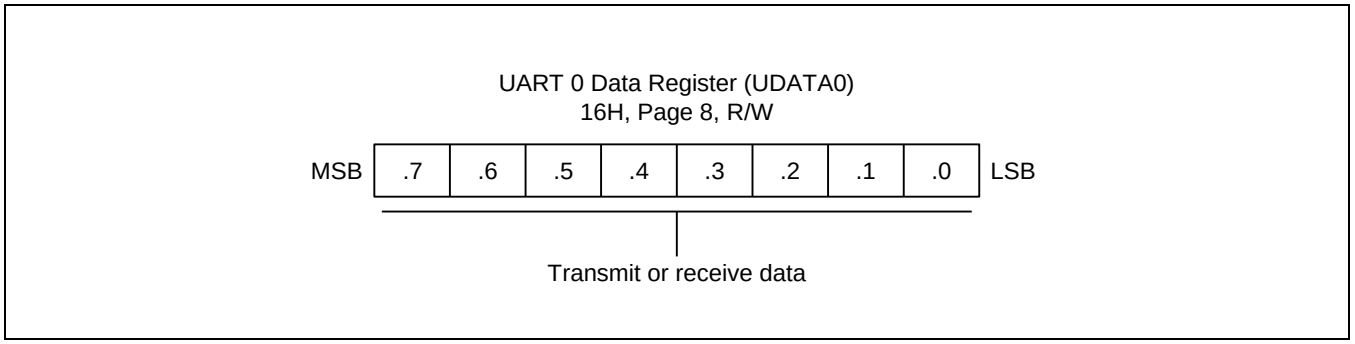


Figure 18-3 UART 0 Data Register (UDATA0)

18.7 UART 0 Baud Rate Data Register (BRDATA0)

The value stored in the UART 0 baud rate register, BRDATA0, lets you determine the UART clock rate (baud rate).

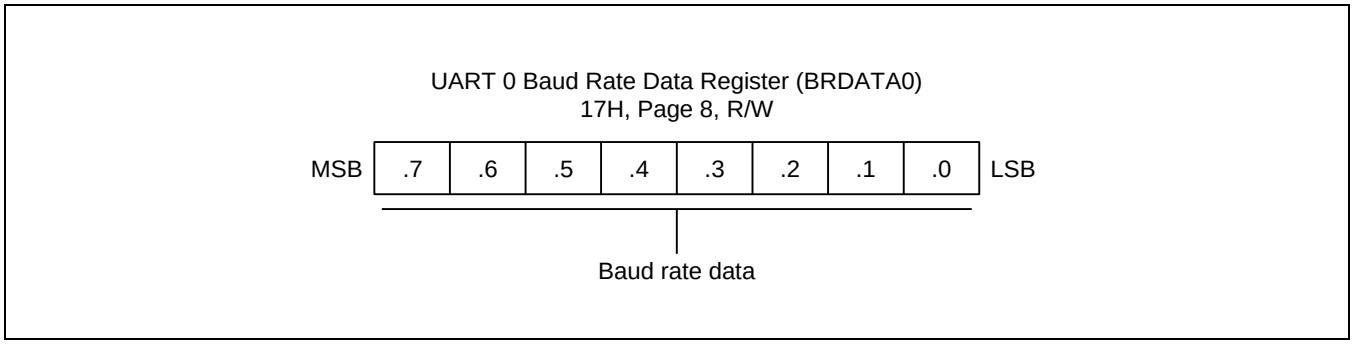


Figure 18-4 UART 0 Baud Rate Data Register (BRDATA0)

18.8 Baud Rate Calculations

18.8.1 Mode 0 Baud Rate Calculation

In mode 0, the baud rate is determined by the UART baud rate data register, BRDATA0 in page 8 at address 17H: Mode 0 baud rate = $f_U/(16 \times (BRDATA0 + 1))$.

18.8.2 Mode 2 Baud Rate Calculation

The baud rate in mode 2 is fixed at the f_U clock frequency divided by 16: Mode 2 baud rate = $f_U/16$

18.8.3 Modes 1 and 3 Baud Rate Calculation

In modes 1 and 3, the baud rate is determined by the UART baud rate data register, BRDATA0 in page 8 at address 17H: Mode 1 and 3 baud rate = $f_U/(16 \times (BRDATA0 + 1))$

Table 18.1 Commonly Used Baud Rates Generated by BRDATA0

Mode	Baud Rate	UART Clock (f_U)	BRDATA0	
			Decimal	Hexadecimal
Mode 2	0.5 MHz	8 MHz	x	x
Mode 0 Mode 1 Mode 3	230,400 Hz	11.0592 MHz	02	02H
	115,200 Hz	11.0592 MHz	05	05H
	57,600 Hz	11.0592 MHz	11	0BH
	38,400 Hz	11.0592 MHz	17	11H
	19,200 Hz	11.0592 MHz	35	23H
	9,600 Hz	11.0592 MHz	71	47H
	4,800 Hz	11.0592 MHz	143	8FH
	62,500 Hz	10 MHz	09	09H
	9,615 Hz	10 MHz	64	40H
	38,461 Hz	8 MHz	12	0CH
	12,500 Hz	8 MHz	39	27H
	19,230 Hz	4 MHz	12	0CH
	9,615 Hz	4 MHz	25	19H

18.9 Block Diagram

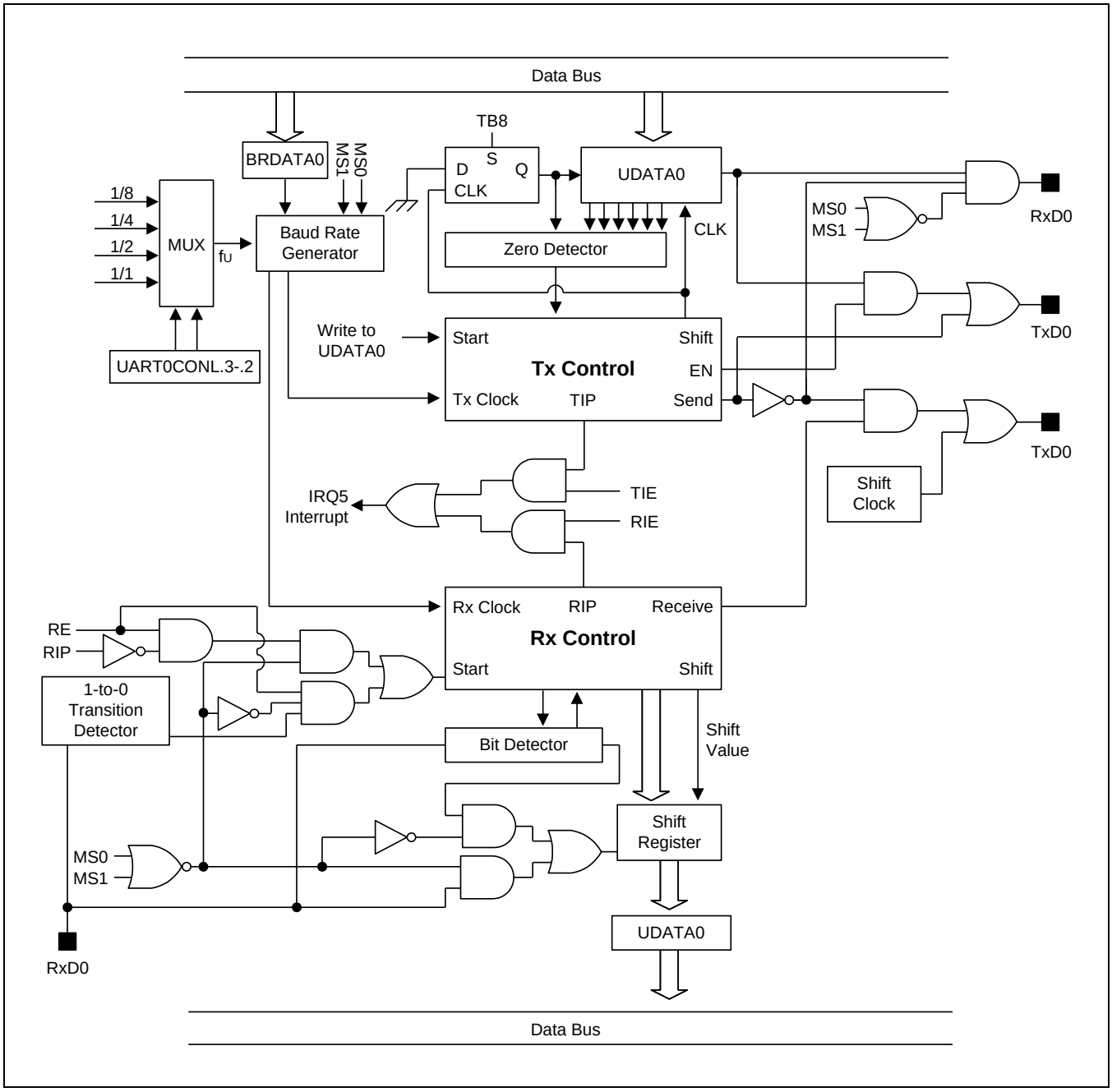


Figure 18-5 UART 0 Functional Block Diagram

18.10 uart 0 Mode 0 Function Description

In mode 0, UART 0 is input and output through the RxD0 (P1.2) pin and TxD0 (P1.3) pin outputs the shift clock. Data is transmitted or received in 8-bit units only. The LSB of the 8-bit value is transmitted (or received) first.

18.10.1 Mode 0 Transmit Procedure

1. Select the UART 0 clock, UART0CONL.3 and .2.
2. Clear the UART 0 transmit parity-bit auto generation enable bit (UART0CONL.7).
3. Select mode 0 by setting UART0CONH.7 and .6 to "00B".
4. Write transmission data to the shift register UDATA0 (16H, page 8) to start the transmission operation.

18.10.2 Mode 0 Receive Procedure

1. Select the UART 0 clock, UART0CONL.3 and .2.
2. Clear the UART 0 transmit parity-bit auto generation enable bit (UART0CONL.7).
3. Select mode 0 by setting UART0CONH.7 and .6 to "00B".
4. Clear the receive interrupt pending bit (UART0CONH.0) by writing a "0" to UART0CONH.0.
5. Set the UART receive enable bit (UART0CONH.4) to "1".
6. The shift clock will now be output to the TxD0 (P1.3) pin and will read the data at the RxD0 (P1.2) pin. A UART 0 receive interrupt occurs when UART0CONH.1 is set to "1".

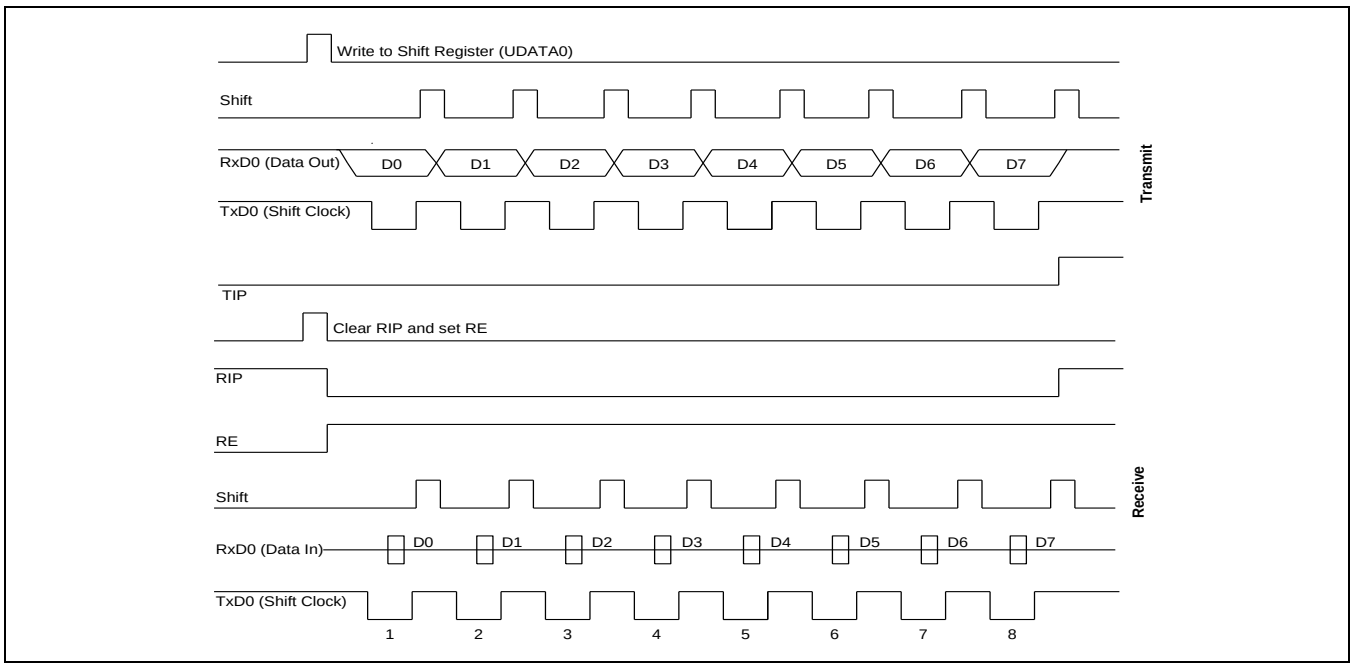


Figure 18-6 Timing Diagram for Serial Port Mode 0 Operation

18.11 Serial Port Mode 1 Function Description

In mode 1, 10-bits are transmitted (through the TxD0 (P1.3) pin) or received (through the RxD0 (P1.2) pin).

Each data frame has three components:

- Start bit ("0")
- 8 data bits (LSB first)
- Stop bit ("1")

The baud rate for mode 1 is variable.

18.11.1 Mode 1 Transmit Procedure

1. Select the UART 0 clock, UART0CONL.3 and .2.
2. Clear the UART 0 transmit parity-bit auto generation enable bit (UART0CONL.7).
3. Select the baud rate to be generated by BRDATA0.
4. Select mode 1 (8-bit UART) by setting UART0CONH bits 7 and 6 to "01B".
5. Write transmission data to the shift register UDATA0 (16H, page 8). The start and stop bits are generated automatically by hardware.

18.11.2 Mode 1 Receive Procedure

1. Select the UART 0 clock, UART0CONL.3 and .2.
2. Clear the UART 0 transmit parity-bit auto generation enable bit (UART0CONL.7).
3. Select the baud rate to be generated by BRDATA0.
4. Select mode 1 and set the RE (Receive Enable) bit in the UART0CONH register to "1".
5. The start bit low ("0") condition at the RxD0 (P1.2) pin will cause the UART 0 module to start the serial data receive operation.

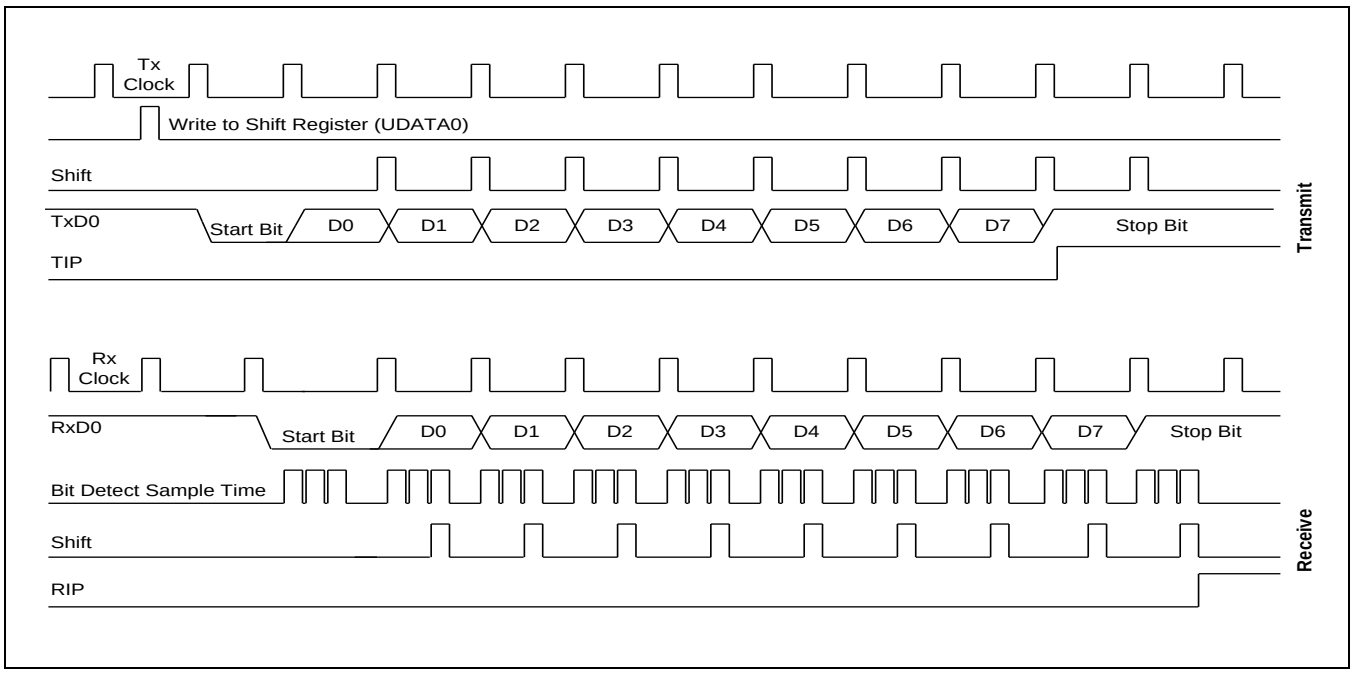


Figure 18-7 Timing Diagram for Serial Port Mode 1 Operation

18.12 Serial Port Mode 2 Function Description

In mode 2, 11-bits are transmitted (through the TxD0 (P1.3) pin) or received (through the RxD0 (P1.2) pin). Each data frame has four components:

- Start bit ("0")
- 8 data bits (LSB first)
- Programmable 9th data bit
- Stop bit ("1")

The 9th data bit to be transmitted can be assigned a value of "0" or "1" by writing the TB8 bit (UART0CONH.3). When receiving, the 9th data bit that is received is written to the RB8 bit (UART0CONH.2), while the stop bit is ignored. The baud rate for mode 2 is $f_U/16$ clock frequency.

18.12.1 Mode 2 Transmit Procedure

1. Select the UART 0 clock, UART0CONL.3 and .2.
2. Select the UART 0 transmit parity-bit auto generation enable or disable bit (UART0CONL.7).
3. Select mode 2 (9-bit UART) by setting UART0CONH bits 7 and 6 to "10B". Also, select the 9th data bit to be transmitted by writing TB8 to "0" or "1".
4. Write transmission data to the shift register, UDATA0 (16H, page 8), to start the transmit operation.

18.12.2 Mode 2 Receive Procedure

1. Select the UART 0 clock, UART0CONL.3 and .2.
2. Select the UART 0 transmit parity-bit auto generation enable or disable bit (UART0CONL.7).
3. Select mode 2 and set the receive enable bit (RE) in the UART0CONH register to "1".
4. The receive operation starts when the signal at the RxD0 (P1.2) pin goes to low level.

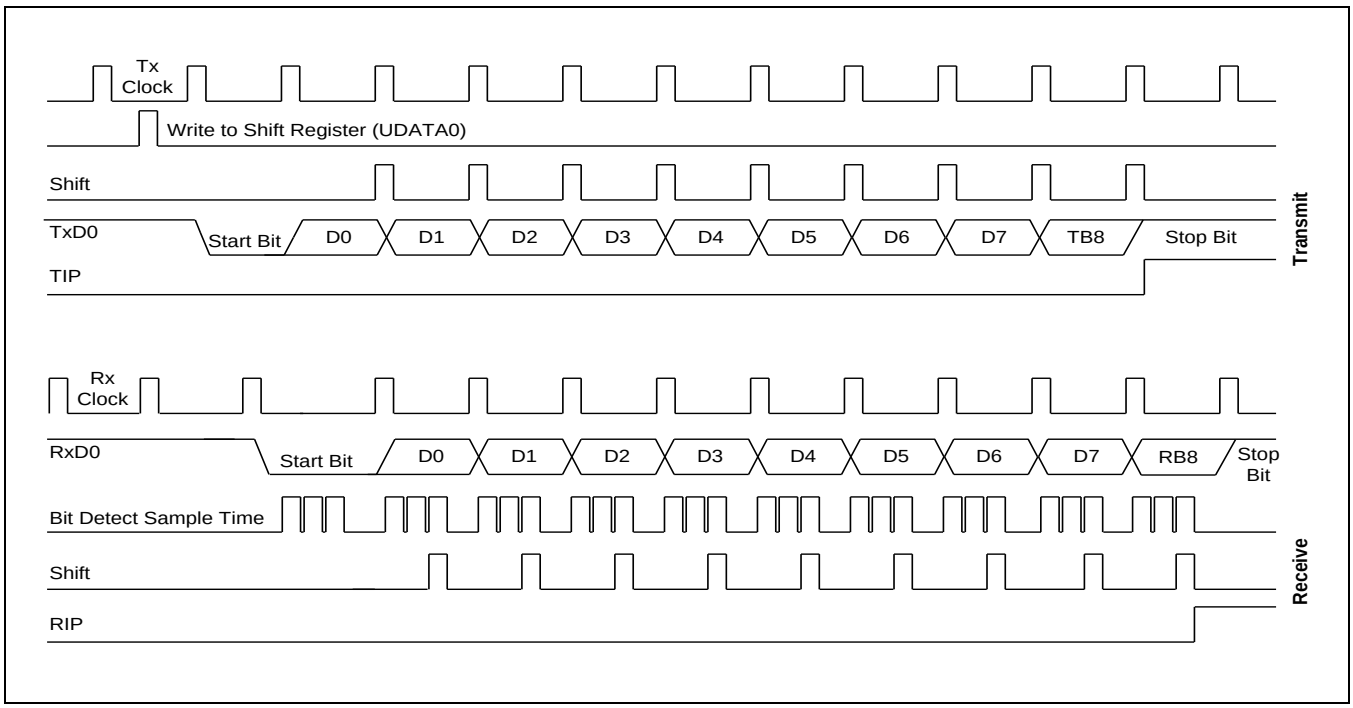


Figure 18-8 Timing Diagram for Serial Port Mode 2 Operation

18.13 Serial Port Mode 3 Function Description

In mode 3, 11-bits are transmitted (through the TxD0 (P1.3) pin) or received (through the RxD0 (P1.2) pin). Mode 3 is identical to mode 2 except for baud rate, which is variable.

Each data frame has four components:

- Start bit ("0")
- 8 data bits (LSB first)
- Programmable 9th data bit
- Stop bit ("1")

18.13.1 Mode 3 Transmit Procedure

1. Select the UART 0 clock, UART0CONL.3 and .2.
2. Select the UART 0 transmit parity-bit auto generation enable or disable bit (UART0CONL.7).
3. Select mode 3 operation (9-bit UART) by setting UART0CONH bits 7 and 6 to "11B". Also, select the 9th data bit to be transmitted by writing UART0CONH.3 (TB8) to "0" or "1".
4. Write transmission data to the shift register, UDATA0 (16H, page 8), to start the transmit operation.

18.13.2 Mode 3 Receive Procedure

1. Select the UART 0 clock, UART0CONL.3 and .2.
2. Select the UART 0 transmit parity-bit auto generation enable or disable bit (UART0CONL.7).
3. Select mode 3 and set the RE (Receive Enable) bit in the UART0CONH register to "1".
4. The receive operation will be started when the signal at the RxD0 (P1.2) pin goes to low level.

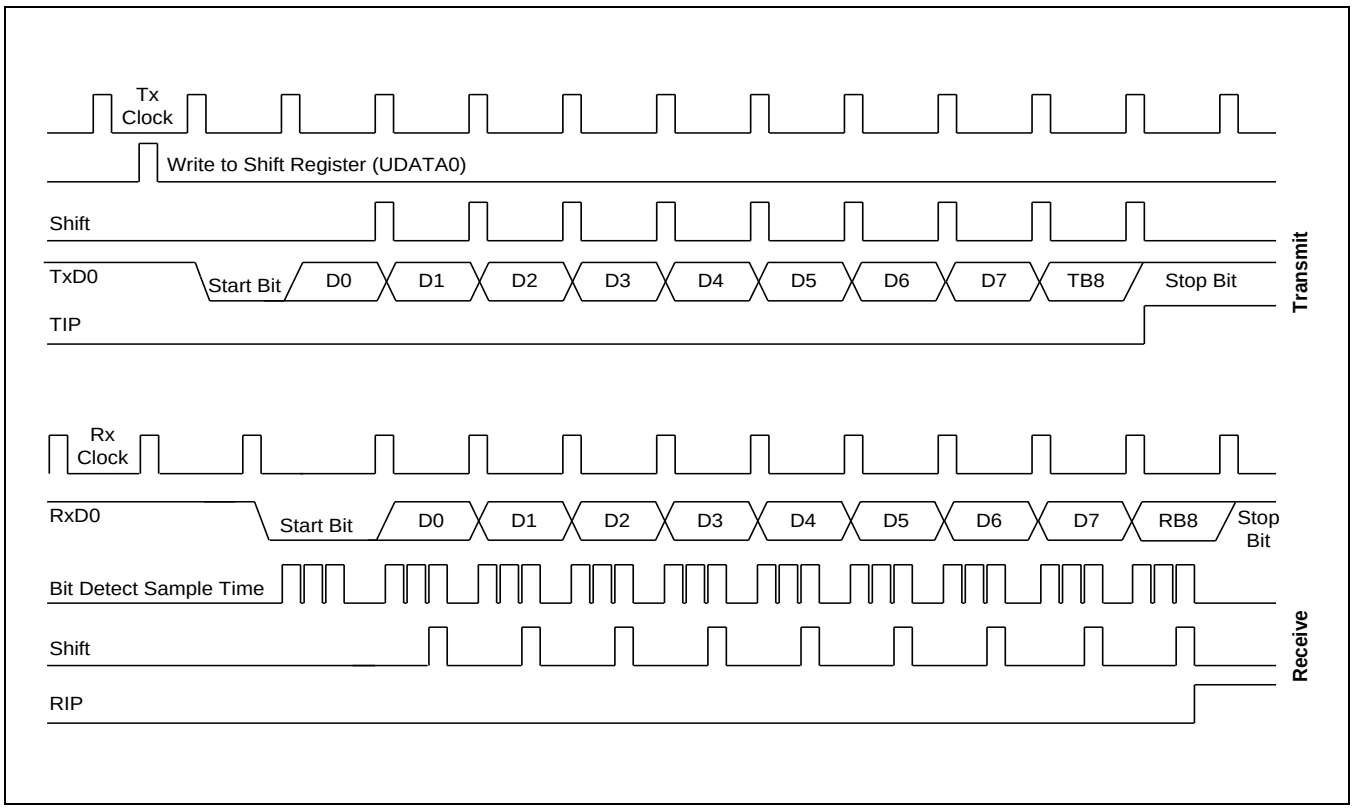


Figure 18-9 Timing Diagram for Serial Port Mode 3 Operation

18.14 Serial Communication for Multiprocessor Configurations

The S3F8-series multiprocessor communication features lets a "master" S3F8S6B send a multiple-frame serial message to a "slave" device in a multi- S3F8S6B configuration. It does this without interrupting other slave devices that may be on the same serial line.

This feature can be used only in UART modes 2 or 3. In these modes 2 and 3, 9 data bits are received. The 9th bit value is written to RB8 (UART0CONH.2). The data receive operation is concluded with a stop bit. You can program this function so that when the stop bit is received, the serial interrupt will be generated only if RB8 = "1".

To enable this feature, you set the MCE bit in the UART0CONH register. When the MCE bit is "1", serial data frames that are received with the 9th bit = "0" do not generate an interrupt. In this case, the 9th bit simply separates the address from the serial data.

18.14.1 Sample Protocol for Master/Slave Interaction

When the master device wants to transmit a block of data to one of several slaves on a serial line, it first sends out an address byte to identify the target slave. Note that in this case, an address byte differs from a data byte: In an address byte, the 9th bit is "1" and in a data byte, it is "0".

The address byte interrupts all slaves so that each slave can examine the received byte and see if it is being addressed. The addressed slave then clears its MCE bit and prepares to receive incoming data bytes.

The MCE bits of slaves that were not addressed remain set, and they continue operating normally while ignoring the incoming data bytes.

While the MCE bit setting has no effect in mode 0, it can be used in mode 1 to check the validity of the stop bit. For mode 1 reception, if MCE is "1", the receive interrupt will be issue unless a valid stop bit is received.

18.14.2 Setup Procedure for Multiprocessor Communications

Follow these steps to configure multiprocessor communications:

1. Set all S3F8S6B devices (masters and slaves) to UART mode 2 or 3.
2. Write the MCE bit of all the slave devices to "1".
3. The master device's transmission protocol is:
   - First byte: the address identifying the target slave device (9^{th} bit = "1")
   - Next bytes: data (9^{th} bit = "0")
   -
4. When the target slave receives the first byte, all of the slaves are interrupted because the 9th data bit is "1". The targeted slave compares the address byte to its own address and then clears its MCE bit in order to receive incoming data. The other slaves continue operating normally.

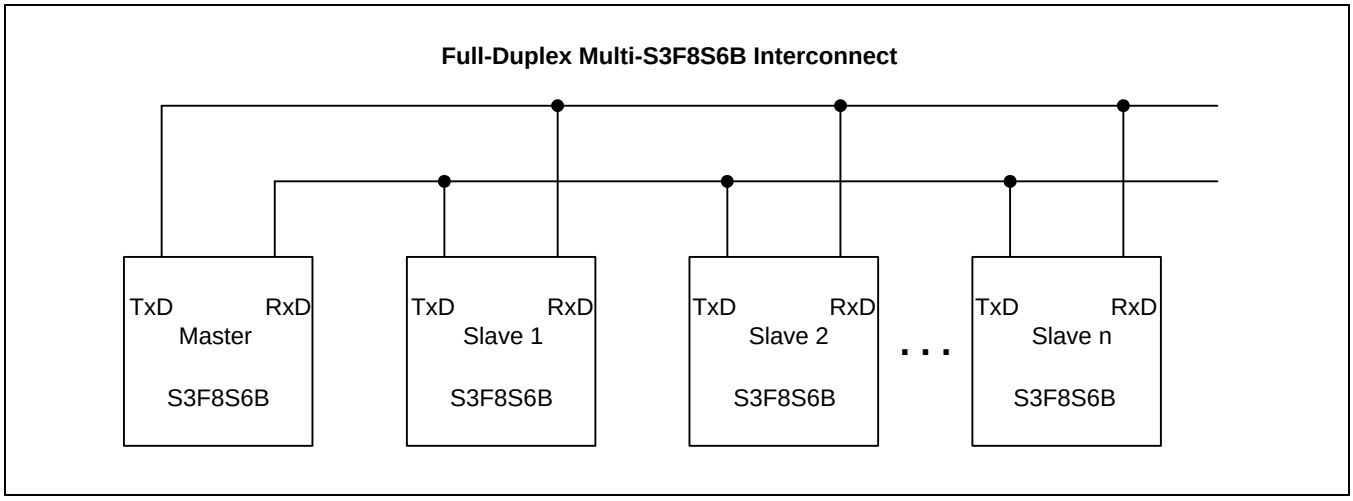


Figure 18-10 Connection Example for Multiprocessor Serial Data Communications

19 UART 1

19.1 Overview

The UART 1 block has a full-duplex serial port with programmable operating modes: There is one synchronous mode and three UART (Universal Asynchronous Receiver/Transmitter) modes:

- Serial I/O with baud rate of $f_U/(16 \times (BRDATA1+1))$
- 8-bit UART mode; variable baud rate
- 9-bit UART mode; $f_U/16$
- 9-bit UART mode, variable baud rate

UART 1 receive and transmit buffers are both accessed via the data register, UDATA1, is page 8 at address 1AH. Writing to the UART data register loads the transmit buffer; reading the UART data register accesses a physically separate receive buffer.

When accessing a receive data buffer (shift register), reception of the next byte can begin before the previously received byte has been read from the receive register. However, if the first byte has not been read by the time the next byte has been completely received, one of the bytes will be lost.

In all operating modes, transmission is started when any instruction (usually a write operation) uses the UDATA1 register as its destination address. In mode 0, serial data reception starts when the receive interrupt pending bit (UART1CONH.0) is "0" and the receive enable bit (UART1CONH.4) is "1". In mode 1, 2, and 3, reception starts whenever an incoming start bit ("0") is received and the receive enable bit (UART1CONH.4) is set to "1".

19.2 Programming Procedure

To program the UART 1 modules, follow these basic steps:

1. Configure P1.4 and P1.5 to alternative function (RxD1 (P1.4), TxD1 (P1.5)) for UART module by setting the P1CONH register to appropriately value.
2. Load an 8-bit value to the UART1CONH/L control register to properly configure the UART I/O module.
3. For interrupt generation, set the UART 1 I/O interrupt enable bit (UART1CONH.1 or UART1CONL.1) to "1".
4. When you transmit data to the UART 1 buffer, write data to UDATA1, the shift operation starts.
5. When the shift operation (receive/transmit) is completed, UART 1 pending bit (UART1CONH.0 or UART1CONL.0) is set to "1" and an UART 1 interrupt request is generated.

19.3 UART 1 High-Byte Control Register (UART1CONh)

The control register for the UART 1 is called UART1CONH in page 8 at address 18H.

It has the following control functions:

- Operating mode and baud rate selection
- Multiprocessor communication and interrupt control
- Serial receive enable/disable control
- 9^{th} data bit location for transmit and receive operations (modes 2 and 3 only)
- UART 1 receive interrupt control

A reset clears the UART1CONH value to "00H". So, if you want to use UART 1 module, you must write appropriate value to UART1CONH.

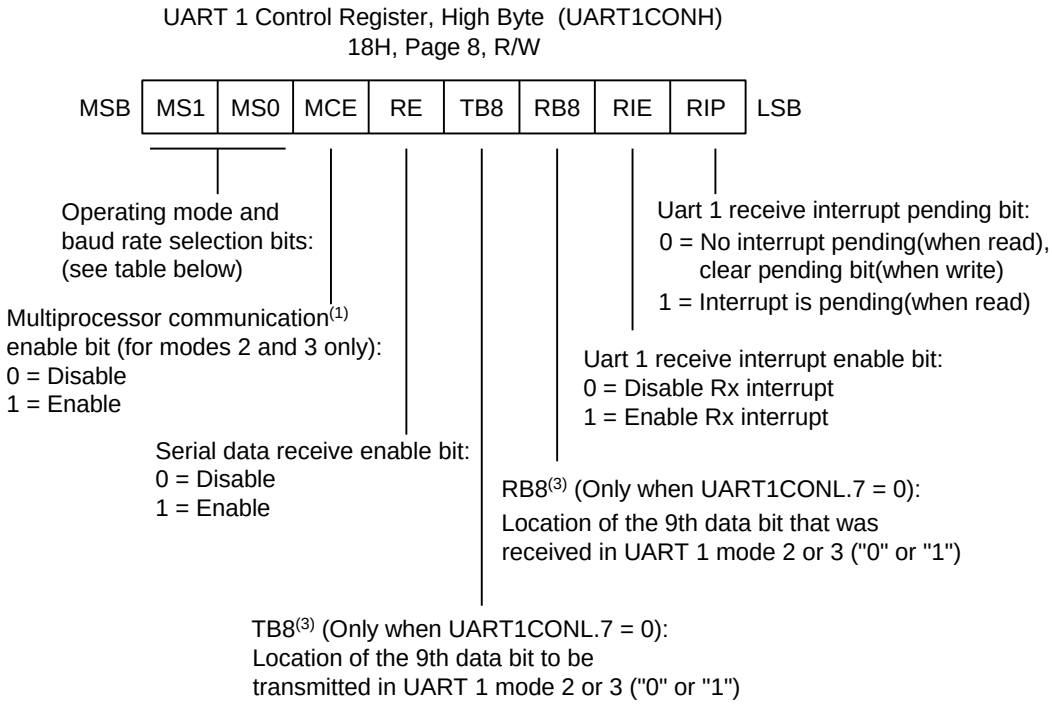
19.4 UART 1 Low-Byte Control Register (UART1CONl)

The control register for the UART 1 is called UART1CONL in page 8 at address 19H.

It has the following control functions:

- UART 1 transmit and receive parity-bit control
- UART 1 clock selection
- UART 1 transmit interrupt control

A reset clears the UART1CONL value to "00H". So, if you want to use UART 1 module, you must write appropriate value to UART1CONL.

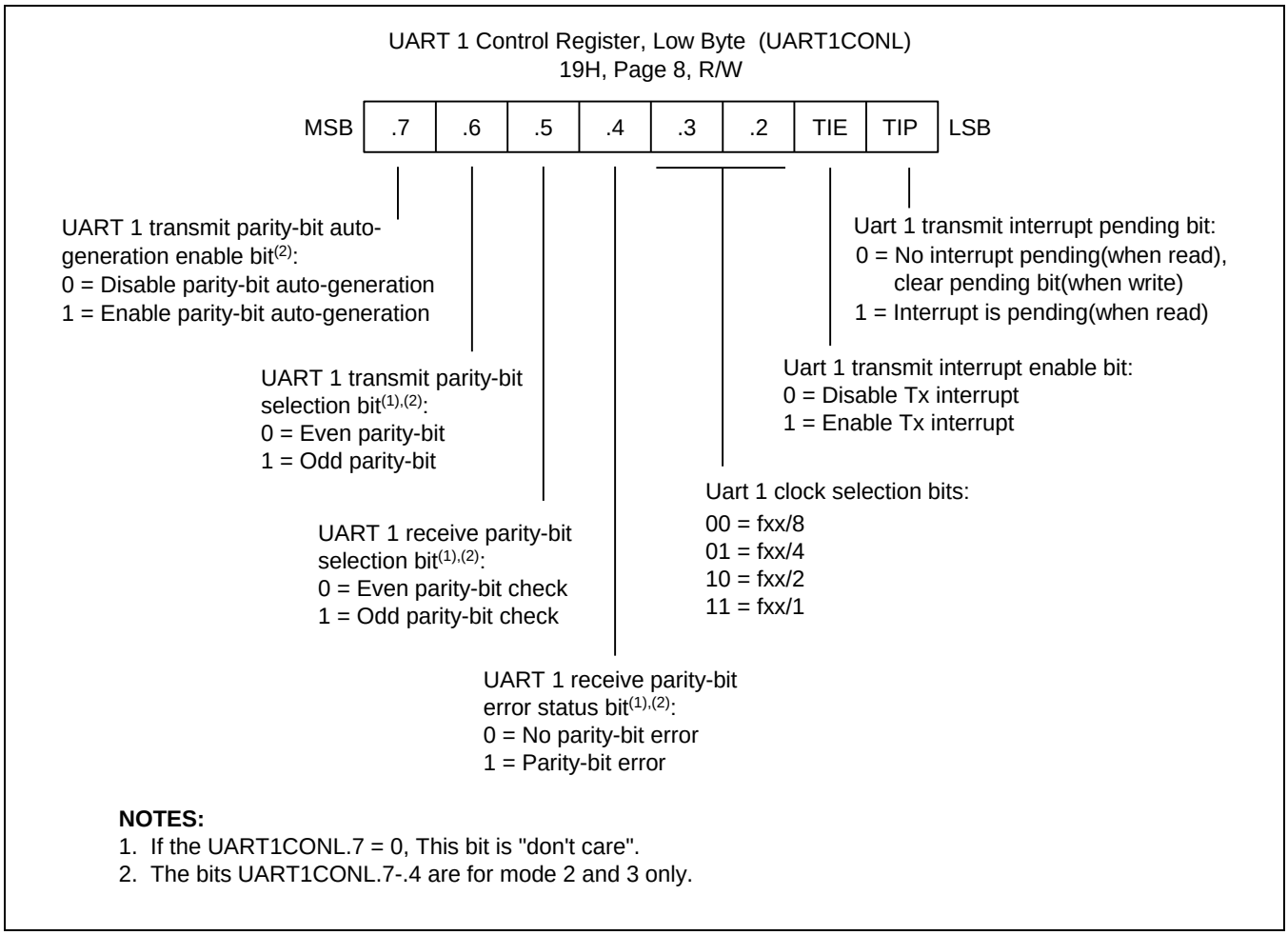


MS1	MS0	Mode	Description(2)	Baud Rate
0	0	0	Shift register	(fu/(16 x (BRDATA1 + 1)))
0	1	1	8-bit UART	(fu/(16 x (BRDATA1 + 1)))
1	0	2	9-bit UART	(fu /16)
1	1	3	9-bit UART	(fu/(16 x (BRDATA1 + 1)))

NOTES:

1. In mode 2 or 3, if the UART1CONH.5 bit is set to "1" then the receive interrupt will not be activated if the received 9th data bit is "0". In mode 1, if UART1CONH.5 = "1" then the receive interrut will not be activated if a valid stop bit was not received. In mode 0, the UART1CONH.5 bit should be "0"
2. The descriptions for 8-bit and 9-bit UART mode do not include start and stop bits for serial data receive and transmit.
3. If the UART1CONL.7 = 1, This bit is "don't care".

Figure 19-1 UART 1 High Byte Control Register (UART1CONH)



NOTES:
1. If the UART1CONL.7 = 0, This bit is "don't care".
2. The bits UART1CONL.7-.4 are for mode 2 and 3 only.

Figure 19-2 UART 1 Low Byte Control Register (UART1CONL)

19.5 UART 1 Interrupt Pending Bits

In mode 0, the receive interrupt pending bit UART1CONH.0 is set to "1" when the 8th receive data bit has been shifted. In mode 1, the UART1CONH.0 bit is set to "1" at the halfway point of the stop bit's shift time. In mode 2, or 3, the UART1CONH.0 bit is set to "1" at the halfway point of the RB8 bit's shift time. When the CPU has acknowledged the receive interrupt pending condition, the UART1CONH.0 bit must then be cleared by software in the interrupt service routine.

In mode 0, the transmit interrupt pending bit UART1CONL.0 is set to "1" when the 8th transmit data bit has been shifted. In mode 1, 2, or 3, the UART1CONL.0 bit is set at the start of the stop bit. When the CPU has acknowledged the transmit interrupt pending condition, the UART1CONL.0 bit must then be cleared by software in the interrupt service routine.

19.6 UART 1 Data Register (UDATA1)

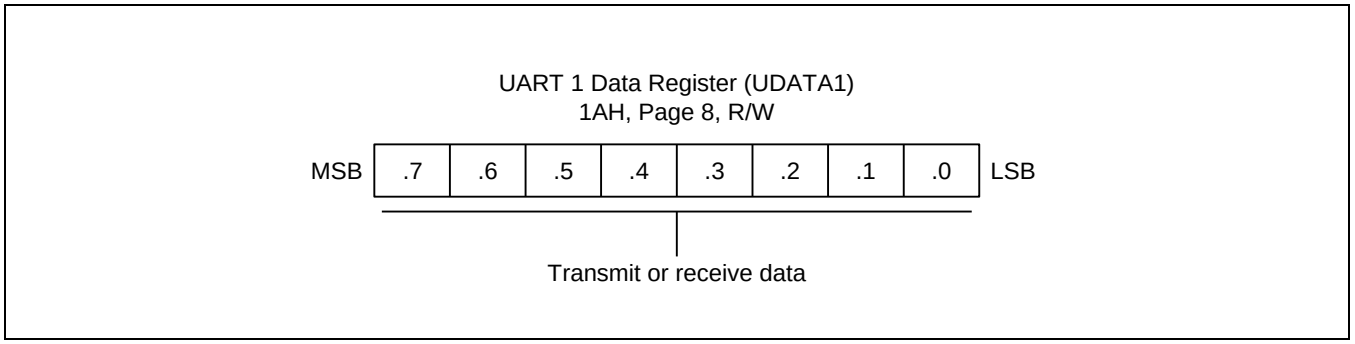


Figure 19-3 UART 1 Data Register (UDATA1)

19.7 UART 1 Baud Rate Data Register (BRDATA1)

The value stored in the UART 1 baud rate register, BRDATA1, lets you determine the UART clock rate (baud rate).

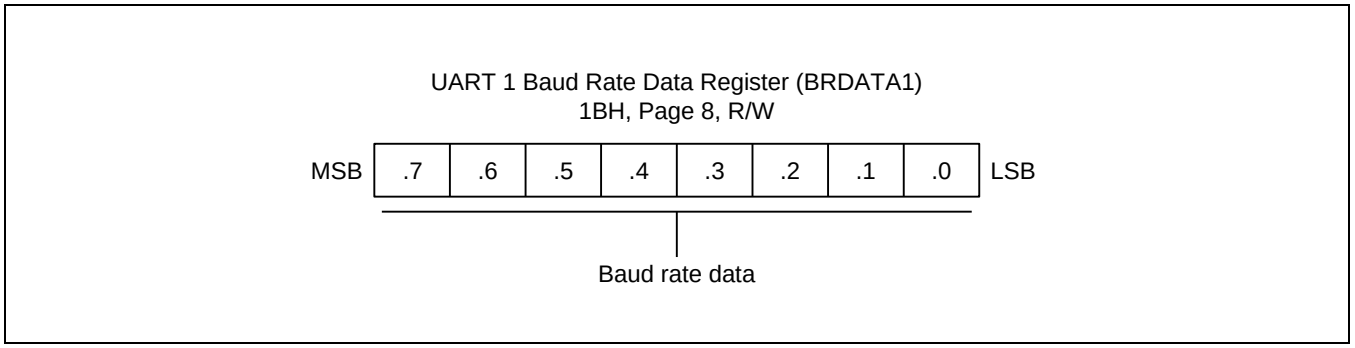


Figure 19-4 UART 1 Baud Rate Data Register (BRDATA1)

19.8 Baud Rate Calculations

19.8.1 Mode 0 Baud Rate Calculation

In mode 0, the baud rate is determined by the UART baud rate data register, BRDATA1 in page 8 at address 1BH: Mode 0 baud rate = $f_U/(16 \times (BRDATA1 + 1))$.

19.8.2 Mode 2 Baud Rate Calculation

The baud rate in mode 2 is fixed at the f_U clock frequency divided by 16: Mode 2 baud rate = $f_U/16$

19.8.3 Modes 1 and 3 Baud Rate Calculation

In modes 1 and 3, the baud rate is determined by the UART baud rate data register, BRDATA1 in page 8 at address 1BH: Mode 1 and 3 baud rate = $f_U/(16 \times (BRDATA1 + 1))$

Table 19.1 Commonly Used Baud Rates Generated by BRDATA1

Mode	Baud Rate	UART Clock (f_U)	BRDATA1	
			Decimal	Hexadecimal
Mode 2	0.5 MHz	8 MHz	x	x
Mode 0 Mode 1 Mode 3	230,400 Hz	11.0592 MHz	02	02H
	115,200 Hz	11.0592 MHz	05	05H
	57,600 Hz	11.0592 MHz	11	0BH
	38,400 Hz	11.0592 MHz	17	11H
	19,200 Hz	11.0592 MHz	35	23H
	9,600 Hz	11.0592 MHz	71	47H
	4,800 Hz	11.0592 MHz	143	8FH
	62,500 Hz	10 MHz	09	09H
	9,615 Hz	10 MHz	64	40H
	38,461 Hz	8 MHz	12	0CH
	12,500 Hz	8 MHz	39	27H
	19,230 Hz	4 MHz	12	0CH
	9,615 Hz	4 MHz	25	19H

19.9 Block Diagram

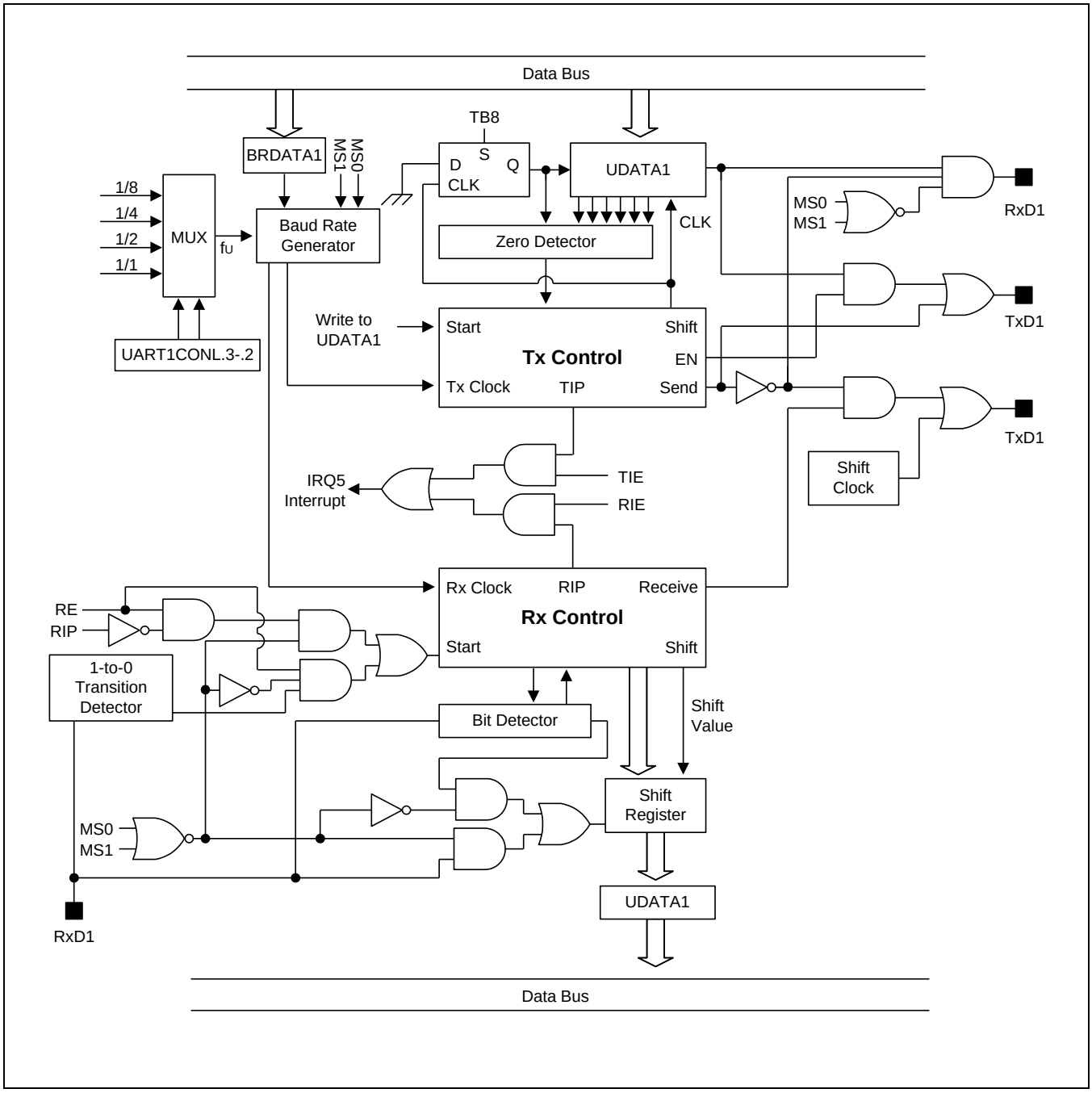


Figure 19-5 UART 1 Functional Block Diagram

19.10 UART 1 Mode 0 Function Description

In mode 0, UART 1 is input and output through the RxD1 (P1.4) pin and TxD1 (P1.5) pin outputs the shift clock. Data is transmitted or received in 8-bit units only. The LSB of the 8-bit value is transmitted (or received) first.

19.10.1 Mode 0 Transmit Procedure

1. Select the UART 1 clock, UART1CONL.3 and .2.
2. Clear the UART 1 transmit parity-bit auto generation enable bit (UART1CONL.7).
3. Select mode 0 by setting UART1CONH.7 and .6 to "00B".
4. Write transmission data to the shift register UDATA1 (1AH, page 8) to start the transmission operation.

19.10.2 Mode 0 Receive Procedure

1. Select the UART 1 clock, UART1CONL.3 and .2.
2. Clear the UART 1 transmit parity-bit auto generation enable bit (UART1CONL.7).
3. Select mode 0 by setting UART1CONH.7 and .6 to "00B".
4. Clear the receive interrupt pending bit (UART1CONH.0) by writing a "0" to UART1CONH.0.
5. Set the UART receive enable bit (UART1CONH.4) to "1".
6. The shift clock will now be output to the TxD1 (P1.5) pin and will read the data at the RxD1 (P1.4) pin. A UART 1 receive interrupt occurs when UART1CONH.1 is set to "1".

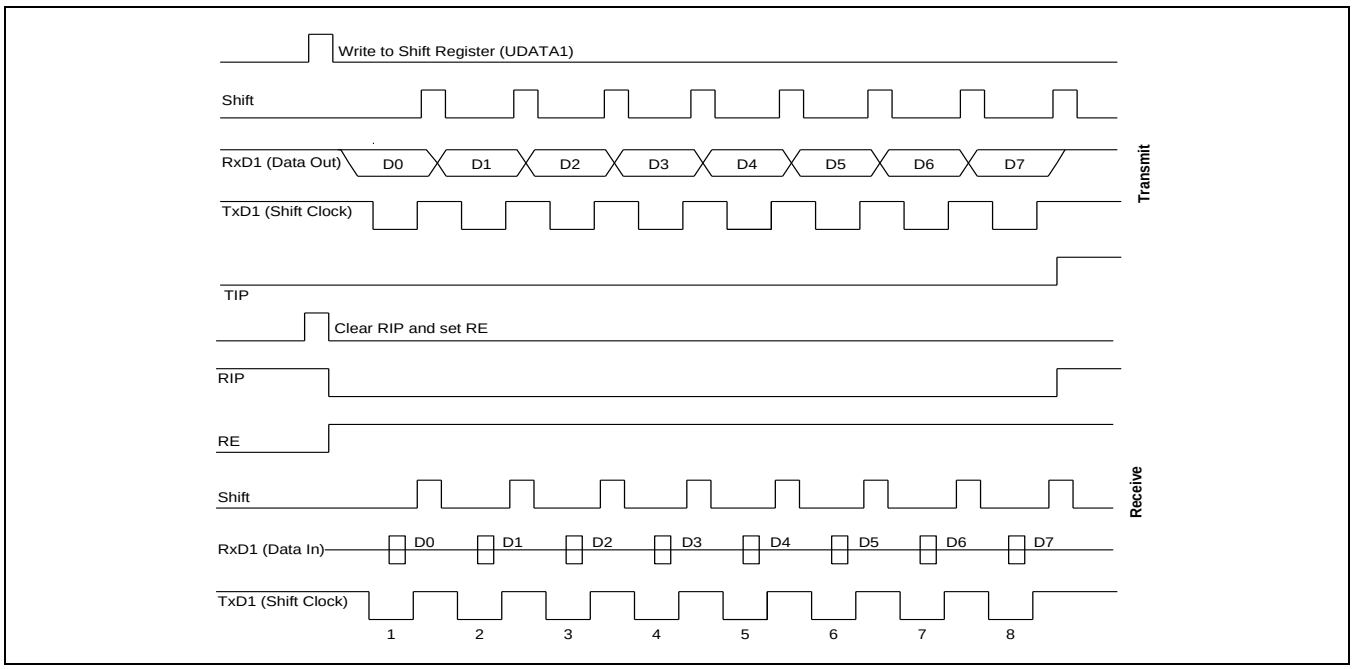


Figure 19-6 Timing Diagram for Serial Port Mode 0 Operation

19.11 Serial Port Mode 1 Function Description

In mode 1, 10-bits are transmitted (through the TxD1 (P1.5) pin) or received (through the RxD1 (P1.4) pin). Each data frame has three components:

- Start bit ("0")
- 8 data bits (LSB first)
- Stop bit ("1")

The baud rate for mode 1 is variable.

19.11.1 Mode 1 Transmit Procedure

1. Select the UART 1 clock, UART1CONL.3 and .2.
2. Clear the UART 1 transmit parity-bit auto generation enable bit (UART1CONL.7).
3. Select the baud rate to be generated by BRDATA1.
4. Select mode 1 (8-bit UART) by setting UART1CONH bits 7 and 6 to "01B".
5. Write transmission data to the shift register UDATA1 (1AH, page 8). The start and stop bits are generated automatically by hardware.

19.11.2 Mode 1 Receive Procedure

1. Select the UART 1 clock, UART1CONL.3 and .2.
2. Clear the UART 1 transmit parity-bit auto generation enable bit (UART1CONL.7).
3. Select the baud rate to be generated by BRDATA1.
4. Select mode 1 and set the RE (Receive Enable) bit in the UART1CONH register to "1".
5. The start bit low ("0") condition at the RxD1 (P1.4) pin will cause the UART 1 module to start the serial data receive operation.

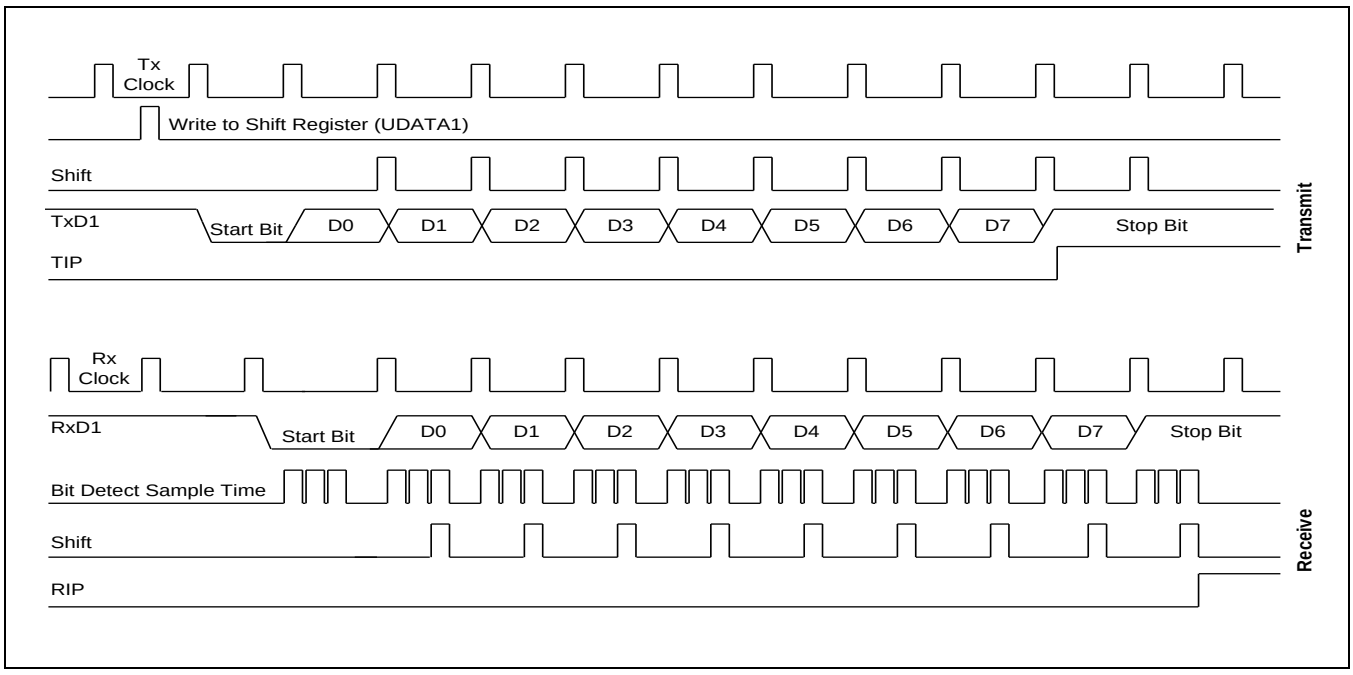


Figure 19-7 Timing Diagram for Serial Port Mode 1 Operation

19.12 Serial Port Mode 2 Function Description

In mode 2, 11-bits are transmitted (through the TxD1 (P1.5) pin) or received (through the RxD1 (P1.4) pin). Each data frame has four components:

- Start bit ("0")
- 8 data bits (LSB first)
- Programmable 9th data bit
- Stop bit ("1")

The 9th data bit to be transmitted can be assigned a value of "0" or "1" by writing the TB8 bit (UART1CONH.3). When receiving, the 9th data bit that is received is written to the RB8 bit (UART1CONH.2), while the stop bit is ignored. The baud rate for mode 2 is fU/16 clock frequency.

19.12.1 Mode 2 Transmit Procedure

1. Select the UART 1 clock, UART1CONL.3 and .2.
2. Select the UART 1 transmit parity-bit auto generation enable or disable bit (UART1CONL.7).
3. Select mode 2 (9-bit UART) by setting UART1CONH bits 7 and 6 to "10B". Also, select the 9th data bit to be transmitted by writing TB8 to "0" or "1".
4. Write transmission data to the shift register, UDATA1 (1AH, page 8), to start the transmit operation.

19.12.2 Mode 2 Receive Procedure

1. Select the UART 1 clock, UART1CONL.3 and .2.
2. Select the UART 1 transmit parity-bit auto generation enable or disable bit (UART1CONL.7).
3. Select mode 2 and set the receive enable bit (RE) in the UART1CONH register to "1".
4. The receive operation starts when the signal at the RxD1 (P1.4) pin goes to low level.

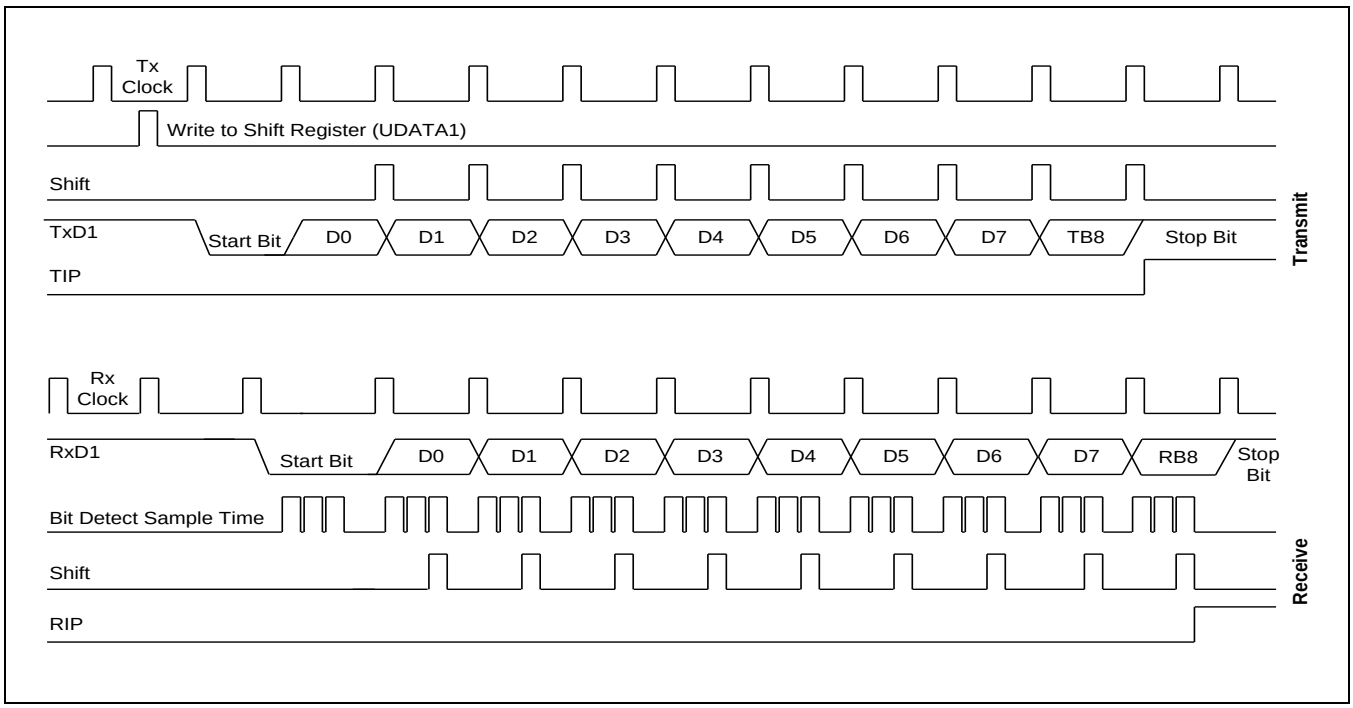


Figure 19-8 Timing Diagram for Serial Port Mode 2 Operation

19.13 Serial Port Mode 3 Function Description

In mode 3, 11-bits are transmitted (through the TxD1 (P1.5) pin) or received (through the RxD1 (P1.4) pin). Mode 3 is identical to mode 2 except for baud rate, which is variable.

Each data frame has four components:

- Start bit ("0")
- 8 data bits (LSB first)
- Programmable 9th data bit
- Stop bit ("1")

19.13.1 Mode 3 Transmit Procedure

1. Select the UART 1 clock, UART1CONL.3 and .2.
2. Select the UART 1 transmit parity-bit auto generation enable or disable bit (UART1CONL.7).
3. Select mode 3 operation (9-bit UART) by setting UART1CONH bits 7 and 6 to "11B". Also, select the 9th data bit to be transmitted by writing UART1CONH.3 (TB8) to "0" or "1".
4. Write transmission data to the shift register, UDATA1 (1AH, page 8), to start the transmit operation.

19.13.2 Mode 3 Receive Procedure

1. Select the UART 1 clock, UART1CONL.3 and .2.
2. Select the UART 1 transmit parity-bit auto generation enable or disable bit (UART1CONL.7).
3. Select mode 3 and set the RE (Receive Enable) bit in the UART1CONH register to "1".
4. The receive operation will be started when the signal at the RxD1 (P1.4) pin goes to low level.

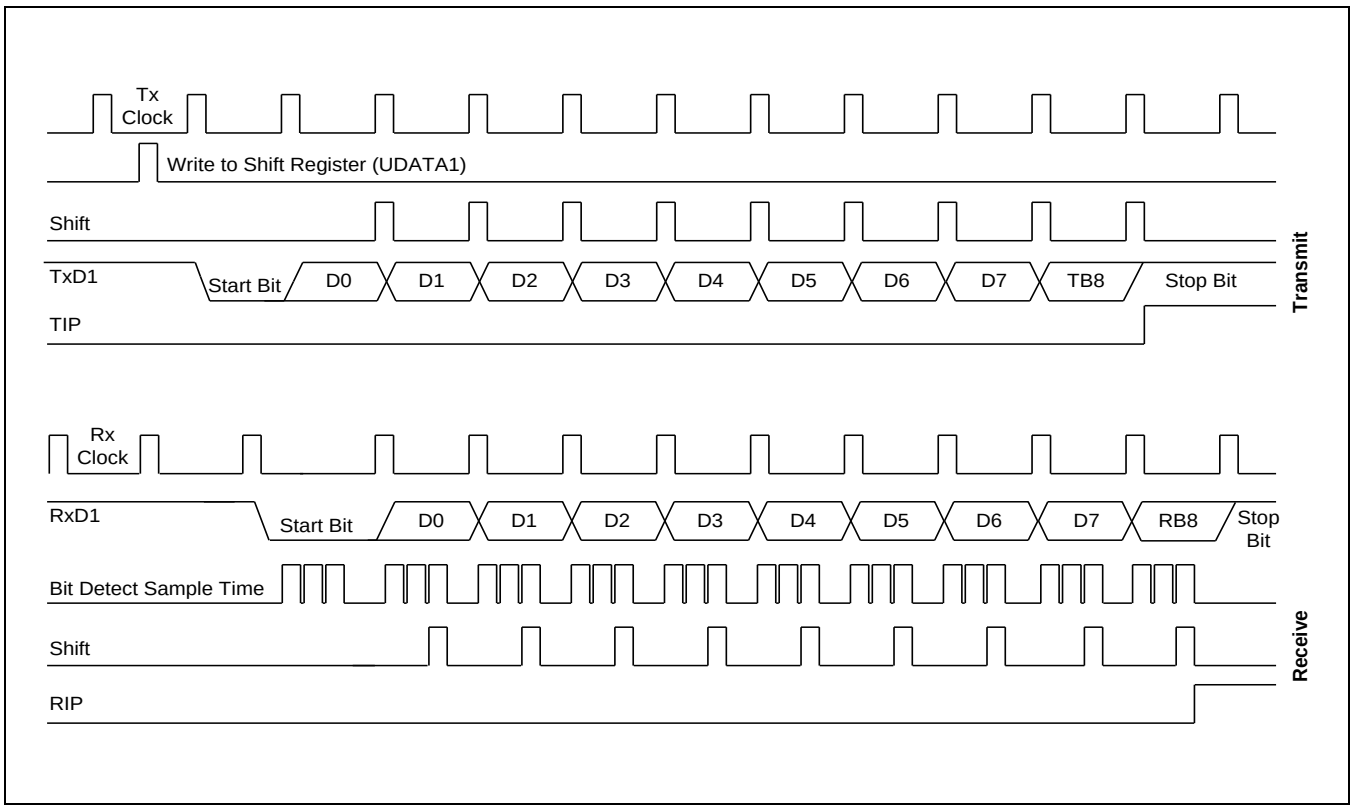


Figure 19-9 Timing Diagram for Serial Port Mode 3 Operation

19.14 Serial Communication for Multiprocessor Configurations

The S3F-series multiprocessor communication features lets a "master" S3F8S6B send a multiple-frame serial message to a "slave" device in a multi- S3F8S6B configuration. It does this without interrupting other slave devices that may be on the same serial line.

This feature can be used only in UART modes 2 or 3. In these modes 2 and 3, 9 data bits are received. The 9th bit value is written to RB8 (UART1CONH.2). The data receive operation is concluded with a stop bit. You can program this function so that when the stop bit is received, the serial interrupt will be generated only if RB8 = "1".

To enable this feature, you set the MCE bit in the UART1CONH register. When the MCE bit is "1", serial data frames that are received with the 9th bit = "0" do not generate an interrupt. In this case, the 9th bit simply separates the address from the serial data.

19.14.1 Sample Protocol for Master/Slave Interaction

When the master device wants to transmit a block of data to one of several slaves on a serial line, it first sends out an address byte to identify the target slave. Note that in this case, an address byte differs from a data byte: In an address byte, the 9th bit is "1" and in a data byte, it is "0".

The address byte interrupts all slaves so that each slave can examine the received byte and see if it is being addressed. The addressed slave then clears its MCE bit and prepares to receive incoming data bytes.

The MCE bits of slaves that were not addressed remain set, and they continue operating normally while ignoring the incoming data bytes.

While the MCE bit setting has no effect in mode 0, it can be used in mode 1 to check the validity of the stop bit. For mode 1 reception, if MCE is "1", the receive interrupt will be issue unless a valid stop bit is received.

19.14.2 Setup Procedure for Multiprocessor Communications

Follow these steps to configure multiprocessor communications:

1. Set all S3F8S6B devices (masters and slaves) to UART mode 2 or 3.
2. Write the MCE bit of all the slave devices to "1".
3. The master device's transmission protocol is:
   - First byte: the address identifying the target slave device (9th bit = "1")
   - Next bytes: data (9th bit = "0")

4. When the target slave receives the first byte, all of the slaves are interrupted because the 9th data bit is "1". The targeted slave compares the address byte to its own address and then clears its MCE bit in order to receive incoming data. The other slaves continue operating normally.

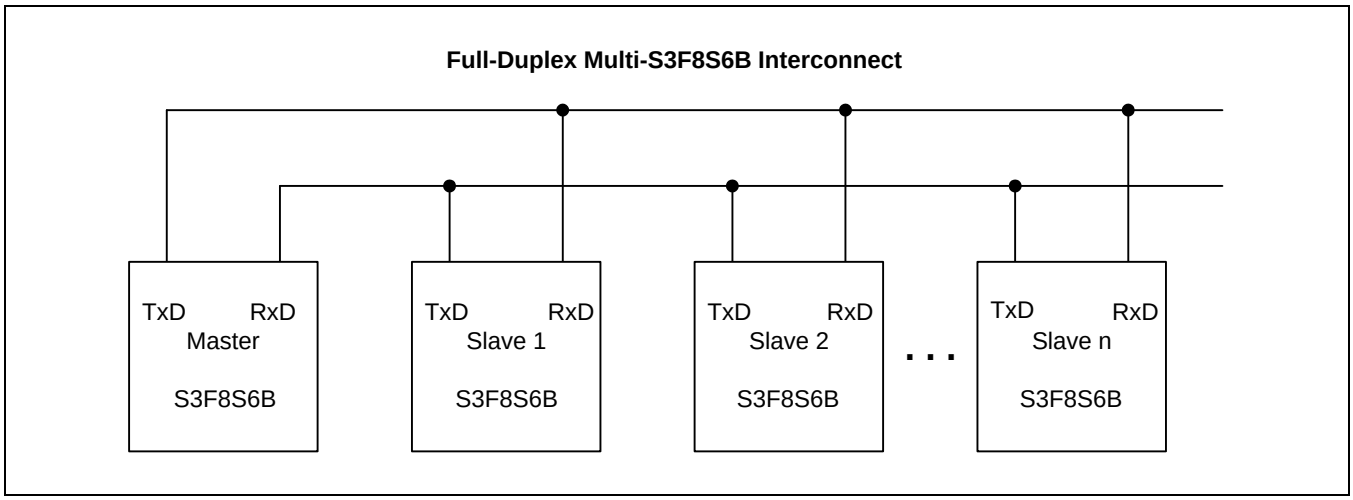


Figure 19-10 Connection Example for Multiprocessor Serial Data Communications

20 Pattern Generation Module

20.1 Overview

20.1.1 PAttern Gneration Flow

You can output up to 8-bit through P3.0-P3.7 by tracing the following sequence. First of all, you have to change the PGDATA into what you want to output. And then you have to set the PGCON to enable the pattern generation module and select the triggering signal. From now, bits of PGDATA are on the P3.0-P3.7 whenever the selected triggering signal happens.

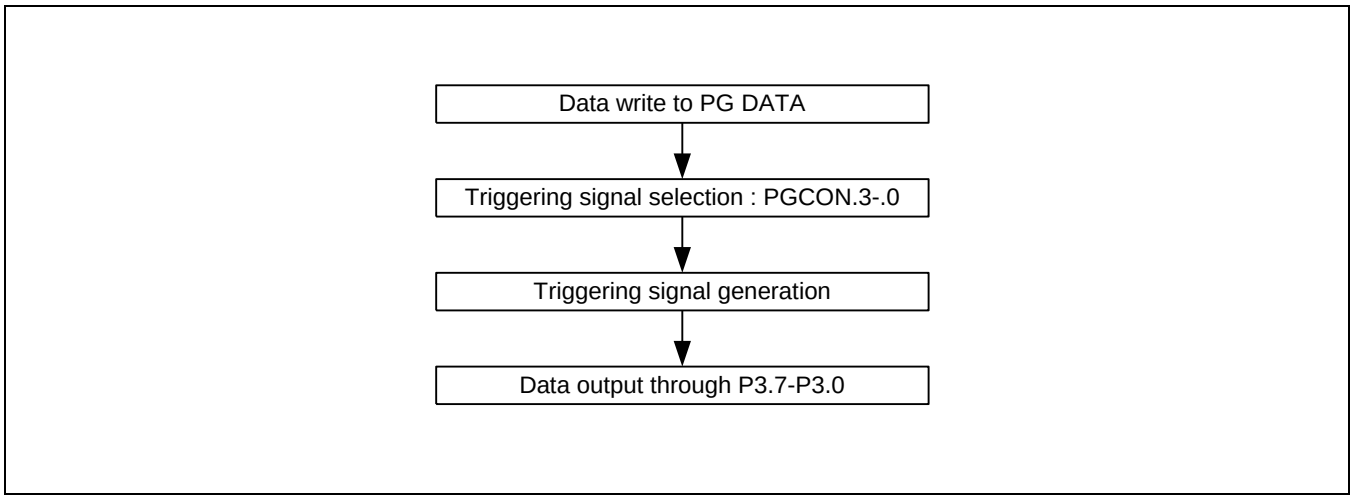


Figure 20-1 Pattern Generation Flow

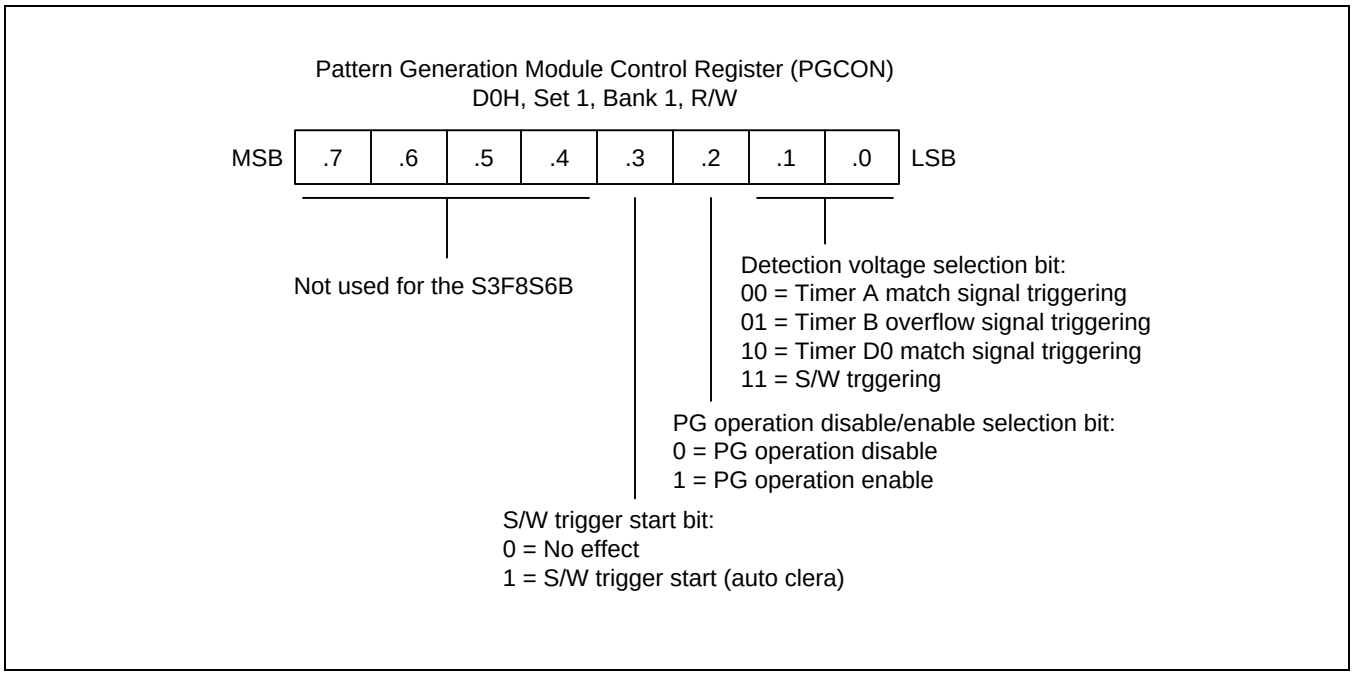


Figure 20-2 Pattern Generation Control Register (PGCON)

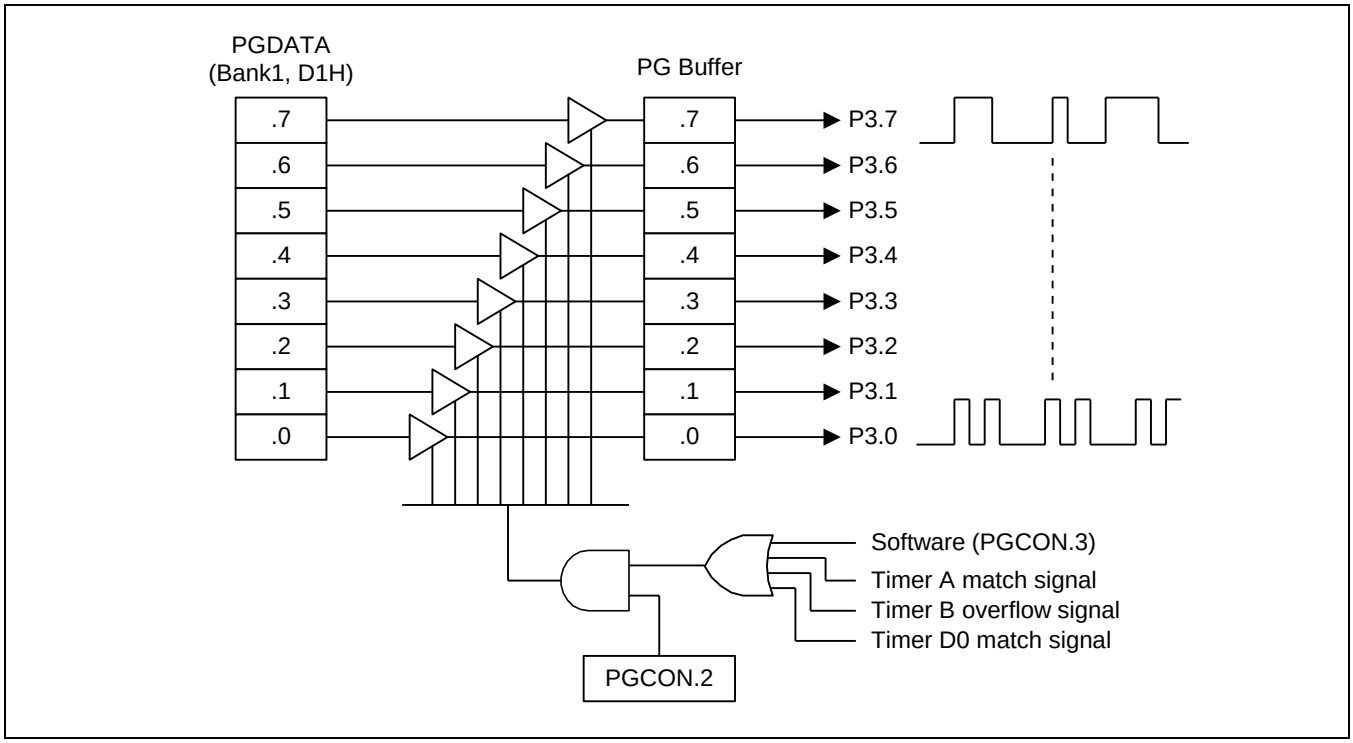


Figure 20-3 Pattern Generation Circuit Diagram

Example 20-1 Using the Pattern Generation

```
      ORG     0000h

      ORG     0100h
INITIAL:
      SB0
      LD      SYM, #00h             ;       Disable Global/Fast interrupt → SYM
      LD      IMR, #01h             ;       Enable IRQ0 interrupt
      LD      SPH, #0h              ;       High byte of stack pointer → SPH
      LD      SPL, #0FFh            ;       Low byte of stack pointer → SPL
      LD      BTCON, #10100011b     ;       Disable Watchdog
      LD      CLKCON, #00011000b    ;       Non-divided (fxx)

      SB1
      LD      P3CONH,#00101001b     ;       Enable PG output
      LD      P3CONM,#01001001b     ;       Enable PG output
      LD      P3CONL,#00001001b     ;       Enable PG output
      SB0
      EI
MAIN:
      NOP
      NOP

      SB1
      LD      PGDATA, #10101010b    ;  PG data setting
      OR      PGCON, #00000100b     ;  Triggering by Timer A match then pattern data are output
      SB0

      NOP
      NOP

      JR      T,MAIN


      .END
```

21 Embedded Flash Memory Interface

21.1 Overview

The S3F8S6B has an on-chip Flash memory internally instead of masked ROM. The Flash memory is accessed by "LDC" instruction and the type of sector erase and a byte programmable Flash, a user can program the data in a Flash memory area any time you want. The S3F8S6B's embedded 64KB memory has two operating features as below:

- User Program Mode
- Tool Program Mode: Refer to the chapter 24. S3F8S6B Flash MCU.

21.2 User Program Mode

This mode supports sector erase, byte programming, byte read and one protection mode (Hard lock protection). The read protection mode is available only in Tool Program mode. So in order to make a chip into read protection, you need to select a read protection option when you program a initial your code to a chip by using Tool Program mode by using a programming tool.

The S3F8S6B has the pumping circuit internally; therefore, 12.5V into V_{PP} (Test) pin is not needed. To program a Flash memory in this mode several control registers will be used. There are four kind functions – programming, reading, sector erase and hard lock protection

NOTE:

1. The user program mode cannot be used when the CPU operates with the subsystem clock.
2. Be sure to execute the DI instruction before starting user program mode. The user program mode checks the interrupt request register (IRQ). If an interrupt request is generated, user program mode is stopped.
3. User program mode is also stopped by an interrupt request that is masked even in the DI status. To prevent this, Be disable the interrupt by using the each peripheral interrupt enable bit.

21.2.1 Flash Memory Control Registers (User Program Mode)

21.2.1.1 Flash Memory Control Register

FMCON register is available only in user program mode to select the Flash Memory operation mode; sector erase, byte programming, and to make the Flash memory into a hard lock protection.

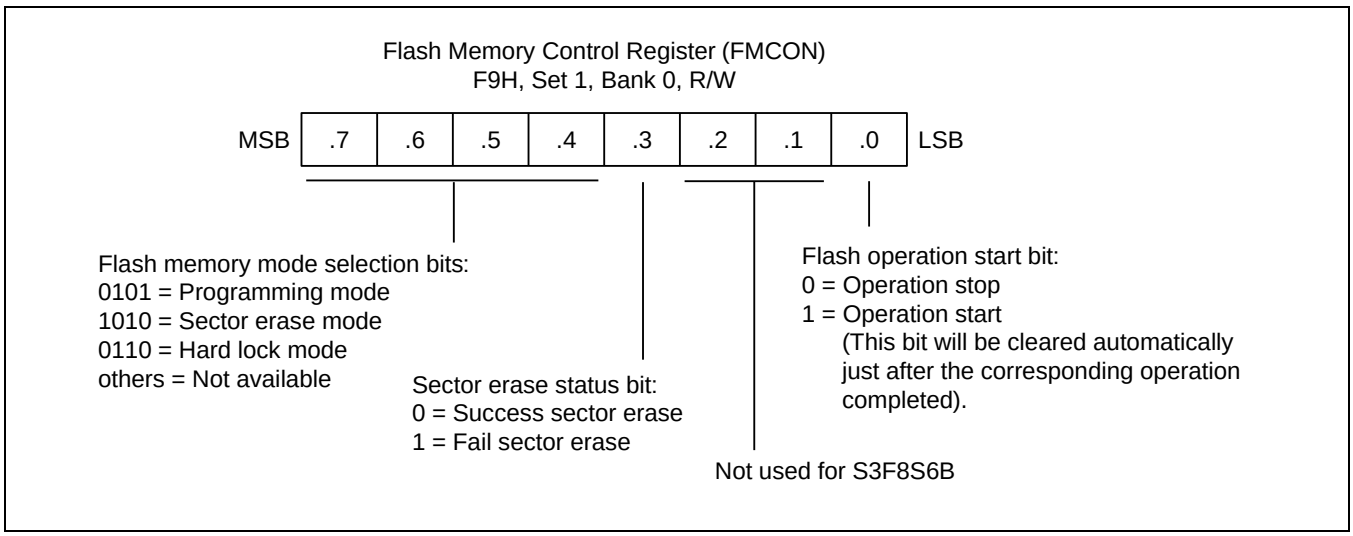


Figure 21-1 Flash Memory Control Register (FMCON)

The bit0 of FMCON register (FMCON.0) is a start bit for Erase and Hard Lock operation mode. Therefore, operation of Erase and Hard Lock mode is activated when you set FMCON.0 to "1". Additionally, wait a time of Erase (Sector erase) or Hard lock to complete its operation before a byte programming or a byte read of same sector area by using "LDC" instruction. When you read or program a byte data from or into Flash memory, this bit is not needed to manipulate.

The sector erase status bit is read only. Even if IMR bits are "0", the interrupt is serviced during the operation of "Sector erase", when the each peripheral interrupt enable bit is set "1" and interrupt pending bit is set "1". If an interrupt is requested during the operation of "Sector erase", the operation of "Sector erase" is discontinued, and the interrupt is served by CPU. Therefore, the sector erase status bit should be checked after executing "Sector erase". The "sector erase" operation is success if the bit is logic "0", and is failure if the bit is logic "1".

NOTE: When the ID code, "A5H", is written to the FMUSR register. A mode of sector erase, user program, and hard lock may be executed unfortunately. So, it should be careful of the above situation.

21.2.1.2 Flash Memory User Programming Enable Register

The FMUSR register is used for a safety operation of the Flash memory. This register will protect undesired erase or program operation from malfunctioning of CPU caused by an electrical noise.

After reset, the user-programming mode is disabled, because the value of FMUSR is "00000000B" by reset operation. If necessary to operate the Flash memory, you can use the user programming mode by setting the value of FMUSR to "10100101B". The other value of "10100101b", User Program mode is disabled.

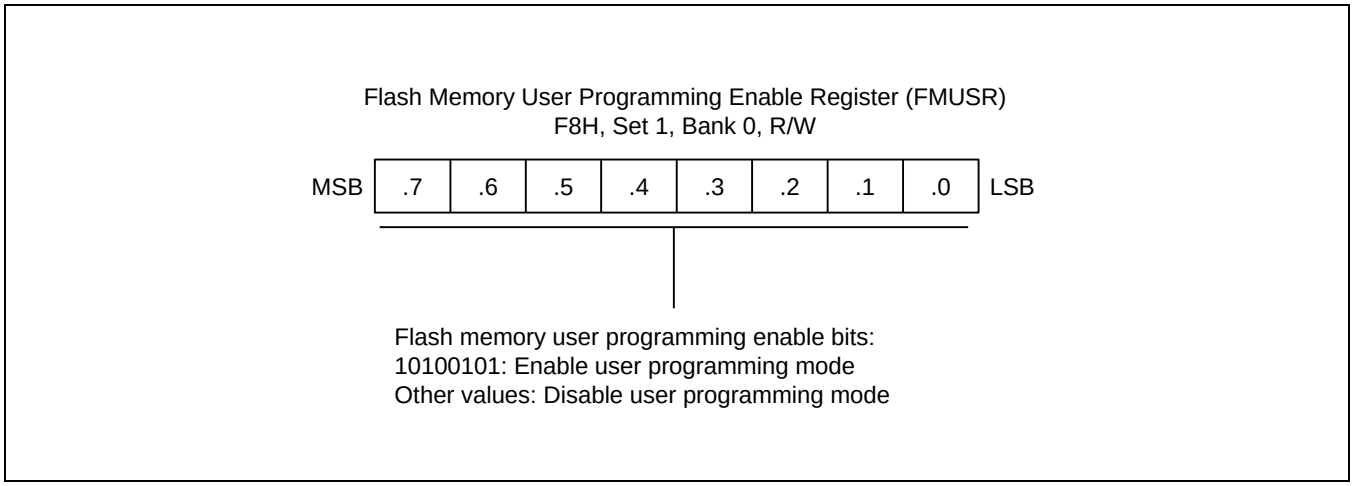


Figure 21-2 Flash Memory User Programming Enable Register (FMUSR)

21.2.1.3 Flash Memory Sector Address Registers

There are two sector address registers for addressing a sector to be erased. The FMSECL (Flash Memory Sector Address Register Low Byte) indicates the low byte of sector address and FMSECH (Flash Memory Sector Address Register High Byte) indicates the high byte of sector address.

The FMSECH is needed for S3F8S6B because it has 512 sectors. One sector consists of 128 bytes. Each sector's address starts XX00H or XX80H that is a base address of sector is XX00H or XX80H. So FMSECL register 6-0 don't mean whether the value is '1' or '0'. We recommend that the simplest way is to load sector base address into FMSECH and FMSECL register.

When programming the Flash memory, you should write data after loading sector base address located in the target address to write data into FMSECH and FMSECL register. If the next operation is also to write data, you should check whether next address is located in the same sector or not. In case of other sectors, you must load sector address to FMSECH and FMSECL register according to the sector.

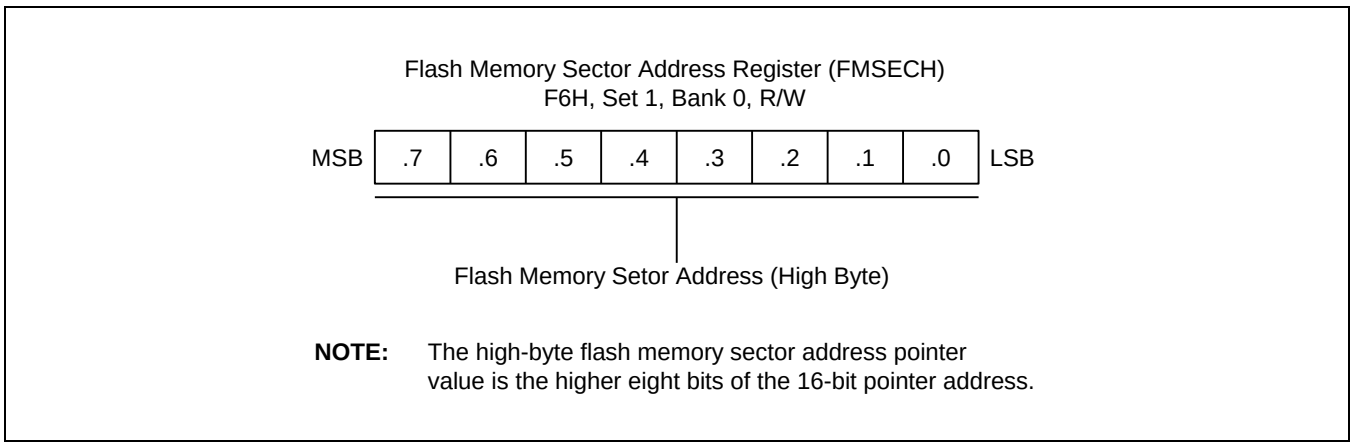


Figure 21-3 Flash Memory Sector Address Register High Byte (FMSECH)

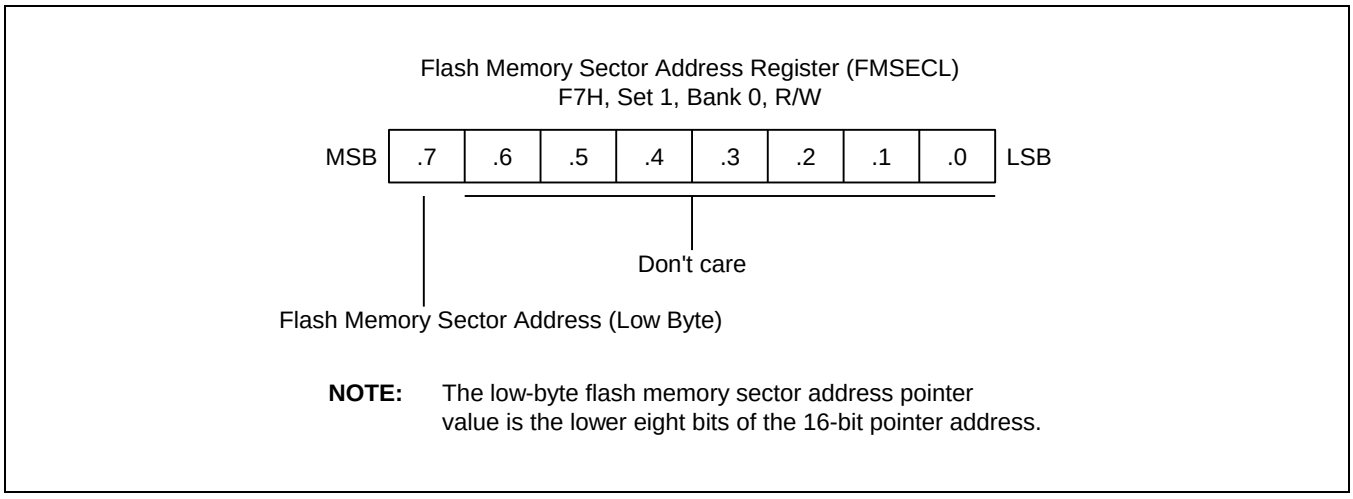


Figure 21-4 Flash Memory Sector Address Register Low Byte (FMSECL)

21.3 ISP™ (On-Board Programming) Sector

ISP™ sectors located in program memory area can store On Board Program software (Boot program code for upgrading application code by interfacing with I/O port pin). The ISP™ sectors can not be erased or programmed by LDC instruction for the safety of On Board Program software.

The ISP sectors are available only when the ISP enable/disable bit is set 0, that is, enable ISP at the Smart Option. If you don't like to use ISP sector, this area can be used as a normal program memory (can be erased or programmed by LDC instruction) by setting ISP disable bit ("1") at the Smart Option. Even if ISP sector is selected, ISP sector can be erased or programmed in the Tool Program mode, by Serial programming tools.

The size of ISP sector can be varied by settings of Smart Option. You can choose appropriate ISP sector size according to the size of On Board Program software.

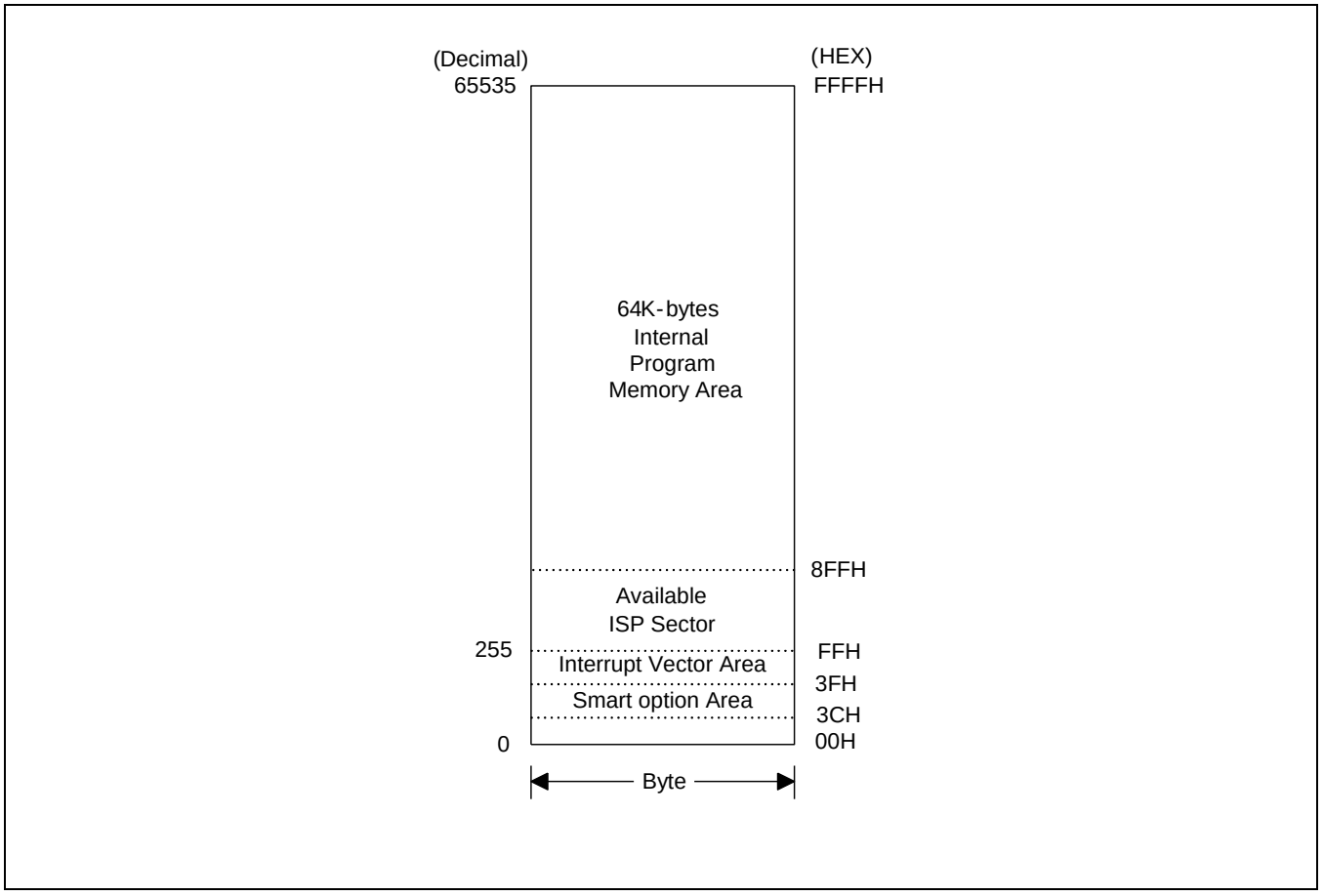


Figure 21-5 Program Memory Address Space

Table 21.1 ISP Sector Size

Smart Option (003EH) ISP size selection bit			Area of ISP sector	ISP sector size
Bit 2	Bit 1	Bit 0		
1	x	x	–	0
0	0	0	100H – 1FFH (256 Byte)	256 Bytes
0	0	1	100H – 2FFH (512 Byte)	512 Bytes
0	1	0	100H – 4FFH (1024 Byte)	1024 Bytes
0	1	1	100H – 8FFH (2048 Byte)	2048 Bytes

NOTE: The area of the ISP sector selected by Smart Option bit (003EH.2 – 003EH.0) can not be erased and programmed by LDC instruction in User Program mode.

21.3.1 ISP Reset Vector and ISP Sector Size

If you use ISP sectors by setting the ISP Enable/Disable bit to "0" and the Reset Vector Selection bit to "0" at the Smart Option, you can choose the reset vector address of CPU as shown in *Table 21.2* by setting the ISP Reset Vector Address Selection bits.

Table 21.2 Reset Vector Address

Smart Option (003EH) ISP Reset Vector Address Selection Bit			Reset Vector Address After POR	Usable Area for ISP Sector	ISP Sector Size
Bit 7	Bit 6	Bit 5			
1	x	x	0100H	–	–
0	0	0	0200H	100H – 1FFH	256 Bytes
0	0	1	0300H	100H – 2FFH	512 Bytes
0	1	0	0500H	100H – 4FFH	1024 Bytes
0	1	1	0900H	100H – 8FFH	2048 Bytes

NOTE: The selection of the ISP reset vector address by Smart Option (003EH.7 – 003EH.5) is not dependent of the selection of ISP sector size by Smart Option (003EH.2 – 003EH.0).

21.4 Sector Erase

User can erase a Flash memory partially by using sector erase function only in User Program Mode. The only unit of Flash memory to be erased and programmed in User Program Mode is called sector.

The program memory of S3F8S6B is divided into 512 sectors for unit of erase and programming. Every sector has all 128 byte sizes of program memory areas. So each sector should be erased first to program a new data (byte) into a sector.

Minimum 10 ms delay time for erase is required after setting sector address and triggering erase start bit (FMCON.0). Sector Erase is not supported in Tool Program Modes (MDS mode tool or Programming tool).

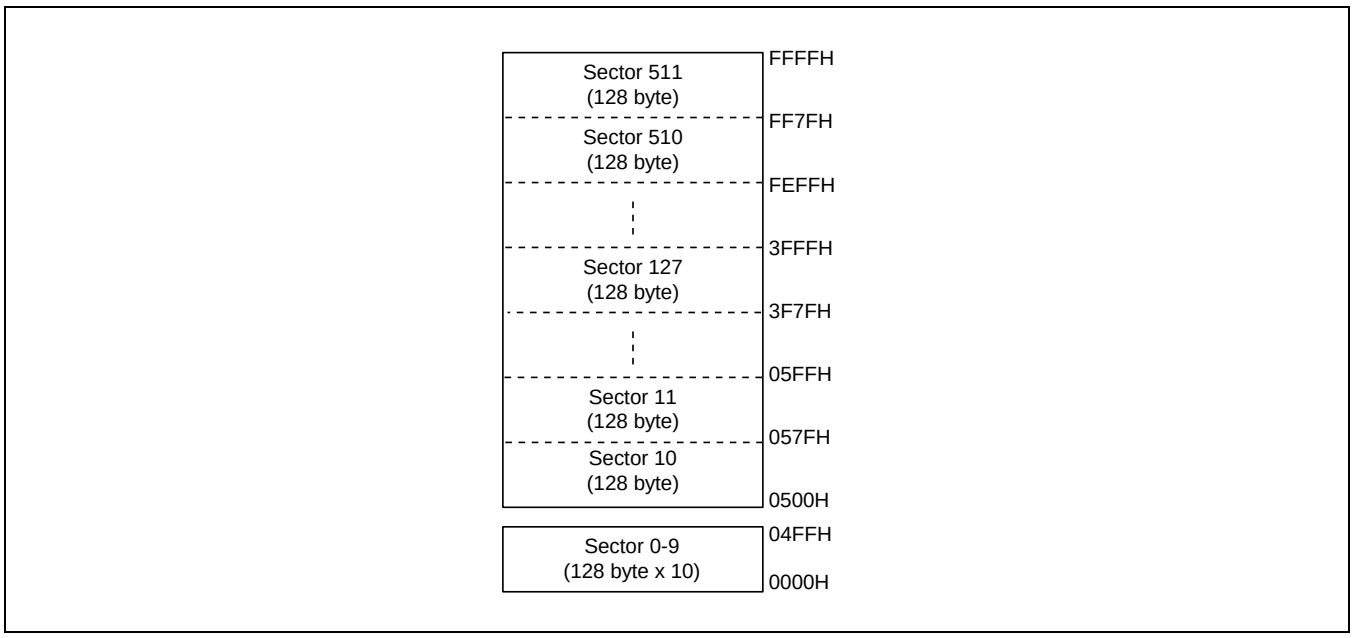


Figure 21-6 Sector Configurations in User Program Mode

The Sector Erase Procedure in User Program Mode

1. If the procedure of Sector Erase needs to be stopped by any interrupt, set the appropriately bit of Interrupt ask Enable Register (IMR) and the appropriately peripheral interrupt enable bit. Otherwise clear all bits of nterrupt Mask Enable Register (IMR) and all peripheral interrupt enable bits.
2. Set Flash Memory User Programming Enable Register (FMUSR) to "10100101B".
3. Set Flash Memory Sector Address Register (FMSECH/ FMSECL).
4. Check user's ID code (written by user)
5. Set Flash Memory Control Register (FMCON) to "10100001B".
6. Set Flash Memory User Programming Enable Register (FMUSR) to "00000000B".
7. Check the "Sector erase status bit" whether "Sector erase" is success or not.

Example 21-1 Sector Erase

```
                •
                •

                SB0
reErase:        LD      FMUSR,Temp0             ; User Program mode enable
                                                ; Temp0 = #0A5H
                                                ; Temp0 variable is must be setting another routine
                LD      FMSECH,#10H
                LD      FMSECL,#00H             ; Set sector address (1000H-107FH)
                CP      UserID_Code,#User_value ; Check user's ID code (written by user)
                                                ; User_value is any value by user
                JR      NE,Not_ID_Code          ; If not equal, jump to Not_ID_Code
                LD      FMCON,Temp1             ; Start sector erase
                                                ; Temp1 = #0A1H
                                                ; Temp1 variable is must be setting another routine
                NOP                             ; Dummy Instruction, This instruction must be
                                                  needed
                NOP                             ; Dummy Instruction, This instruction must be
                                                  needed
                LD      FMUSR,#0                ; User Program mode disable
                TM      FMCON,#00001000B        ; Check "Sector erase status bit"
                JR      NZ,reErase              ; Jump to reErase if fail
                •
                •
                •
Not_ID_Code:
                SB0
                LD      FMUSR,#0                ; User Program mode disable
                LD      FMCON,#0                ; Sector erase mode disable
                •
                •
                •
```

NOTE: In case of Flash User Mode, the Tmep0 to Temp1's data values are must be setting another routine. Temp0 to Temp (n) variables are should be defined by user.

21.5 Programming

A Flash memory is programmed in one byte unit after sector erase. And for programming safety's sake, must set FMSECH and FMSECL to Flash memory sector value.

The write operation of programming starts by "LDC" instruction.
You can write until 128 byte, because this Flash sector's limit is 128 byte.
So, if you written 128 byte, must reset FMSECH and FMSECL.

The Program Procedure in User Program Mode

1. Must erase sector before programming.
2. Set Flash Memory User Programming Enable Register (FMUSR) to "10100101B".
3. Set Flash Memory Sector Register (FMSECH, FMSECL) to sector value of write address.
4. Load a Flash memory upper address into upper register of pair working register.
5. Load a Flash memory lower address into lower register of pair working register.
6. Load a transmission data into a working register.
7. Check user's ID code (written by user)
8. Set Flash Memory Control Register (FMCON) to "01010001B".
9. Load transmission data to Flash memory location area on 'LDC' instruction by indirectly addressing mode
10. Set Flash Memory User Programming Enable Register (FMUSR) to "00000000B".

Example 21-2 Programming

```
      •
      •

      SB0
      LD          FMUSR,Temp0              ; User Program mode enable
                                           ; Temp0 = #0A5H
                                           ; Temp0 variable is must be setting another routine
      LD          FMSECH,#17H
      LD          FMSECL,#80H              ; Set sector address (1780H-17FFH)
      LD          R2,#17H                  ; Set a ROM address in the same sector 1780H-17FFH
      LD          R3,#84H
      LD          R4,#78H                  ; Temporary data
      CP          UserID_Code,#User_value  ; Check user's ID code (written by user)
                                           ; User_value is any value by user
      JR          NE,Not_ID_Code           ; If not equal, jump to Not_ID_Code
      LD          FMCON,Temp1              ; Start program
                                                 ; Temp1 = #51H
                                           ; Temp1 variable is must be setting another routine

      LDC         @RR2,R4                  ; Write the data to a address of same sector(1784H)
      NOP                                  ; Dummy Instruction, This instruction must be
                                             needed
      LD          FMUSR,#0                 ; User Program mode disable
      •
      •
      •
      •

Not_ID_Code:
      SB0
      LD          FMUSR,#0                 ; User Program mode disable
      LD          FMCON,#0                 ; Programming mode disable
      •
      •
      •
      •
```

NOTE: In case of Flash User Mode, the Tmep0 to Temp1's data values are must be setting another routine. Temp0 to Temp(n) variables are should be defined by user.

21.6 Reading

The read operation of programming starts by "LDC" instruction.

The Reading Procedure in User Program Mode

1. Load a Flash memory upper address into upper register of pair working register.
2. Load a Flash memory lower address into lower register of pair working register.
3. Load receive data from Flash memory location area on "LDC' instruction by indirectly addressing mode

Example 21-3 Reading

```
      •
      •

      LD            R2,#3H              ; Load Flash memory upper address
                                        ; to upper of pair working register
      LD            R3,#0               ; Load Flash memory lower address
                                        ; to lower pair working register
LOOP: LDC           R0,@RR2             ; Read data from Flash memory location
                                        ; (Between 300H and 3FFH)
      INC           R3
      CP            R3,#0H
      JP            NZ,LOOP
      •
      •
      •
      •
```

21.7 Hard Lock Protection

User can set Hard Lock Protection by write "0110" in FMCON.7-4. If this function is enabled, the user cannot write or erase the data in a Flash memory area. This protection can be released by the chip erase execution (in the tool program mode).

In terms of user program mode, the procedure of setting Hard Lock Protection is following that. Whereas in tool mode the manufacturer of serial tool writer could support Hardware Protection. Please refer to the manual of serial program writer tool provided by the manufacturer.

The Hard Lock Protection Procedure in User program mode

1. Set Flash Memory User Programming Enable Register (FMUSR) to "10100101B".
2. Check user's ID code (written by user)
3. Set Flash Memory Control Register (FMCON) to "01100001B".
4. Set Flash Memory User Programming Enable Register (FMUSR) to "00000000B".

Example 21-4 Hard Lock Protection

```
        •
        •

                SB0
                LD      FMUSR,Temp0                 ; User Program mode enable
                                                    ; Temp0 = #0A5H
                                                    ; Temp0 variable is must be setting another routine
                CP      UserID_Code,#User_value     ; Check user's ID code (written by user)
                                                    ; User_value is any value by user
                JR      NE,Not_ID_Code              ; If not equal, jump to Not_ID_Code
                LD      FMCON,Temp1                 ; Hard Lock mode set & start
                                                    ; Temp1 = #61H
                                                    ; Temp1 variable is must be setting another routine
                NOP                                 ; Dummy Instruction, This instruction must be
                                                      needed
                LD      FMUSR,#0                    ; User Program mode disable
                •
                •
                •
                •
Not_ID_Code:
                SB0
                LD      FMUSR,#0                    ; User Program mode disable
                LD      FMCON,#0                    ; Hard Lock Protection mode disable
                •
                •
                •
                •
```

NOTE: In case of Flash User Mode, the Tmep0 to Temp1's data values are must be setting another routine. Temp0 to Temp(n) variables are should be defined by user.

22 Electrical Data

22.1 Overview

In this chapter, S3F8S6B electrical characteristics are presented in tables and graphs.

The information is arranged in the following order:

- Absolute maximum ratings
- Input/output capacitance
- D.C. electrical characteristics
- A.C. electrical characteristics
- Oscillation characteristics
- Oscillation stabilization time
- Data retention supply voltage in Stop mode
- LVR timing characteristics
- Serial I/O timing characteristics
- A/D converter electrical characteristics
- LCD capacitor bias electrical characteristics
- UART timing characteristics
- Internal Flash ROM electrical characteristics
- Operating voltage range

22.2 Absolute Maximum Ratings

Table 22.1 Absolute Maximum Ratings

(T_A = 25 °C)

Parameter	Symbol	Conditions	Rating	Unit
Supply voltage	V_{DD}	–	– 0.3 to + 6.5	V
Input voltage	V_I	Ports 0-6	– 0.3 to V_{DD} + 0.3	
Output voltage	V_O	–	– 0.3 to V_{DD} + 0.3	
Output current high	I_{OH}	One I/O pin active	– 15	mA
		All I/O pins active	– 60	
Output current low	I_{OL}	One I/O pin active	+ 30 (Peak value)	
		Total pin current for ports	+ 100 (Peak value)	
Operating temperature	T_A	–	– 40 to + 85	°C
Storage temperature	T_{STG}	–	– 65 to + 150	

22.3 D.C. Electrical Characteristics

Table 22.2 D.C. Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Operating voltage	V_{DD}	fx = 0.4 – 4.2 MHz, fxt = 32.768 kHz	1.8	–	5.5	V
		fx = 0.4 – 12.0 MHz	2.2	–	5.5	
Input high voltage	V_{IH1}	All input pins except V_{IH2}	0.7 V_{DD}	–	V_{DD}	
	V_{IH2}	P2, P4	0.8 V_{DD}		V_{DD}	
	V_{IH3}	nRESET	0.8 V_{DD}		V_{DD}	
	V_{IH4}	X_{IN}, X_{OUT}	V_{DD} – 0.1		V_{DD}	
Input low voltage	V_{IL1}	All input pins except V_{IL2}	–	–	0.3 V_{DD}	
	V_{IL2}	P2, P4			0.2 V_{DD}	
	V_{IL3}	nRESET			0.2 V_{DD}	
	V_{IL4}	X_{IN}, X_{OUT}			0.1	
Output high voltage	V_{OH}	V_{DD} = 4.5 V to 5.5 V I_{OH} = –1 mA All output ports	V_{DD} – 1.5	–	–	V
Output low voltage	V_{OL1}	V_{DD} = 4.5 V to 5.5 V I_{OL} = 10 mA All output ports	–	–	2.0	
Input high leakage current	I_{LIH1}	V_{IN} = V_{DD} All input pins except ILIH2	–	–	3	µA
	I_{LIH2}	V_{IN} = V_{DD} X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}			20	
Input low leakage current	I_{LIL1}	V_{IN} = 0 V All input pins except for nRESET, I_{LIL2}	–	–	– 3	
	I_{LIL2}	V_{IN} = 0 V X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}			– 20	
Output high leakage current	I_{LOH}	V_{OUT} = V_{DD} All output pins	–	–	3	
Output low leakage current	I_{LOL}	V_{OUT} = 0 V All output pins	–	–	– 3	

(T_A = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
LCD voltage dividing resistor	R_{LCD}	T_A = 25 °C	40	65	90	kΩ
Oscillator feed back resistors	R_{OSC1}	V_{DD} = 5 V, T_A= 25 °C X_{IN} = V_{DD}, X_{OUT} = 0 V	420	850	1700	
	R_{OSC2}	V_{DD} = 5 V, T_A= 25 °C XT_{IN} = V_{DD}, XT_{OUT} = 0 V	2200	4500	9000	
Pull-up resistor	R_{L1}	V_{IN} = 0 V; V_{DD} = 5 V Ports 0–6, T_A = 25 °C	25	50	100	
		V_{IN} = 0 V; V_{DD} = 3 V Ports 0 to 6, T_A = 25 °C	50	100	150	
	R_{L2}	V_{IN} = 0 V; V_{DD} = 5 V T_A = 25 °C, nRESET	150	250	400	
		V_{IN} = 0 V; V_{DD} = 3 V T_A = 25 °C, nRESET	300	500	700	
Middle output voltage (1)	V_{LC1}	V_{DD} = 2.7 V to 5.5 V LCD clock = 0 Hz, V_{LC0} = V_{DD}	0.75 V_{DD} – 0.2	0.75 V_{DD}	0.75 V_{DD} + 0.2	V
	V_{LC2}		0.50 V_{DD} – 0.2	0.5 V_{DD}	0.5 V_{DD} + 0.2	
	V_{LC3}		0.25 V_{DD} – 0.2	0.25 V_{DD}	0.25 V_{DD} + 0.2	
|V_{LCD} – COMi| Voltage drop (i = 0 to 7)	V_{DC}	– 15 μA per common pin	–	–	120	mV
|V_{LCD} – SEGx| Voltage drop (x = 0 to 33)	V_{DS}	– 15 μA per segment pin	–	–	120	

(T_A = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions		Min.	Typ.	Max.	Unit
Supply current (1)	I_{DD1} (2)	Run mode: V_{DD} = 5.0 V Crystal oscillator C1 = C2 = 22 pF	12.0 MHz	–	2.2	4.0	mA
			4.2 MHz		1.2	2.0	
		V_{DD} = 3.0V	4.2 MHz		0.8	1.5	
	I_{DD2} (2)	Idle mode: V_{DD} = 5.0 V Crystal oscillator C1 = C2 = 22 pF	12.0 MHz	–	1.3	2.3	
			4.2 MHz		0.8	1.5	
		V_{DD} = 3.0 V	4.2 MHz		0.4	0.8	
	I_{DD3} (3)	Sub Operating mode: V_{DD} = 3.0 V, T_A = 25 °C 32 kHz crystal oscillator		–	80.0	120.0	µA
		Run mode: V_{DD} = 3.0 V, T_A = 25 °C 32 kHz crystal oscillator C = 0.1 µF, No panel load Cap bias LCD on			85.0	140.0	
	I_{DD4} (3)	Sub Idle mode: V_{DD} = 3.0 V, T_A = 25 °C 32 kHz crystal oscillator		–	6.0	15.0	
		Idle mode: V_{DD} = 3.0 V, T_A = 25°C 32 kHz crystal oscillator C = 0.1 µF, No panel load Cap bias LCD on			10.0	20.0	
	I_{DD5} (4)	Stop mode: V_{DD} = 5.0 V, T_A = 25°C		–	0.3	3.0	
		Stop mode: V_{DD} = 5.0 V, T_A = +85°C		–	3.0	8.0	
		Stop mode: V_{DD} = 5.0 V, T_A = -40 to +85°C		–	–	8.0	

NOTE: It is middle output voltage when the V_{DD} and V_{LC0} pin are connected.

1. Supply current does not include current drawn through internal pull-up resistors, LCD voltage dividing resistors, the LVR block, and external output current loads.
2. I_{DD1} and I_{DD2} include a power consumption of sub clock oscillation.
3. I_{DD3} and I_{DD4} are the current when the main clock oscillation stops and the sub clock is used.
4. I_{DD5} is the current when the main and sub clock oscillation stops.
5. Every value in this table is measured when bits 4-3 of the system clock control register (CLKCON.4–.3) is set to 11B.

22.4 A.C. Electrical Characteristics

Table 22.3 A.C. Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Interrupt input high, low width (P2, P4)	t_{INTH}, t_{INTL}	All interrupt, V_{DD} = 5 V	500	–	–	ns
nRESET input low width	t_{RSL}	Input, V_{DD} = 5 V	10	–	–	µs

NOTE: If width of interrupt or reset pulse is greater than min. value, pulse is always recognized as valid pulse.

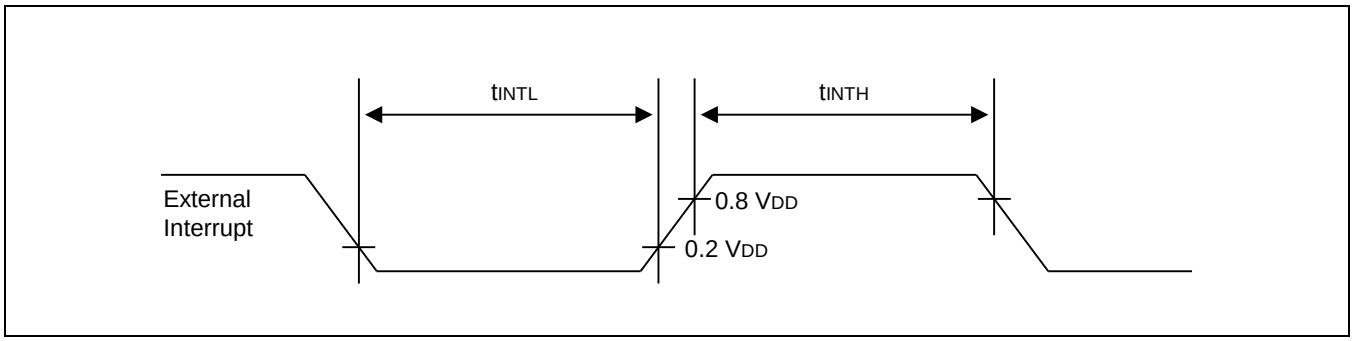


Figure 22-1 Input Timing for External Interrupts

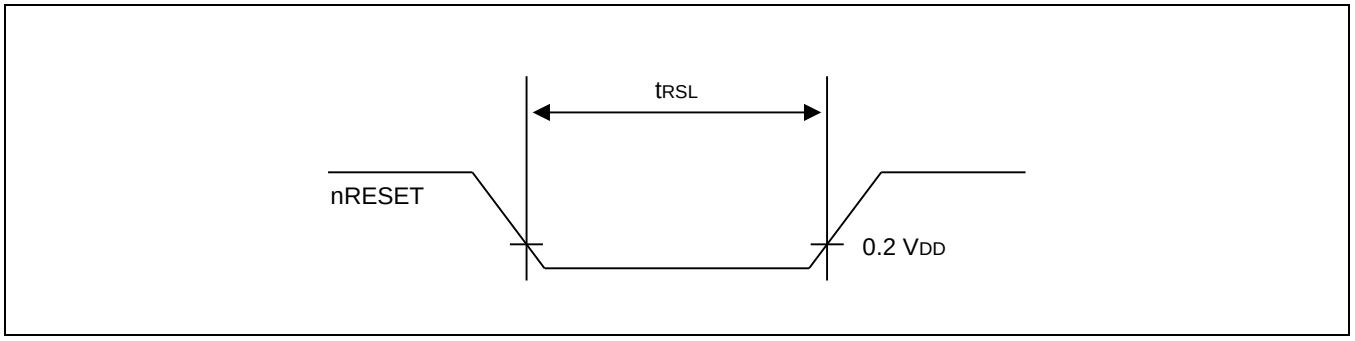


Figure 22-2 Input Timing for nRESET

22.5 Input/Output Capacitance

Table 22.4 Input/Output Capacitance

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

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

22.6 Data Retention Supply Voltage in Stop Mode

Table 22.5 Data Retention Supply Voltage in Stop Mode

(T_A = – 40 °C to + 85 °C)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Data retention supply voltage	V_{DDDR}	–	1.8	–	5.5	V
Data retention supply current	I_{DDDR}	Stop mode, T_A = 25 °C V_{DDDR} = 1.8V Disable LVR block	–	–	1	µA

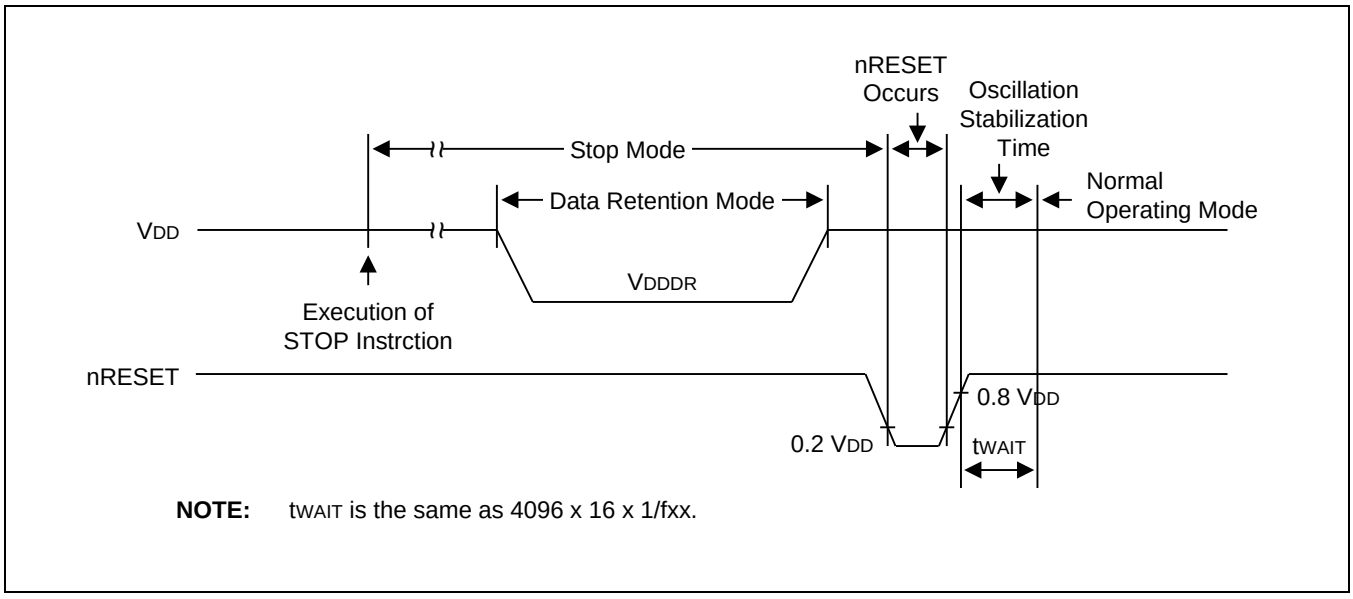


Figure 22-3 Stop Mode Release Timing Initiated by nRESET

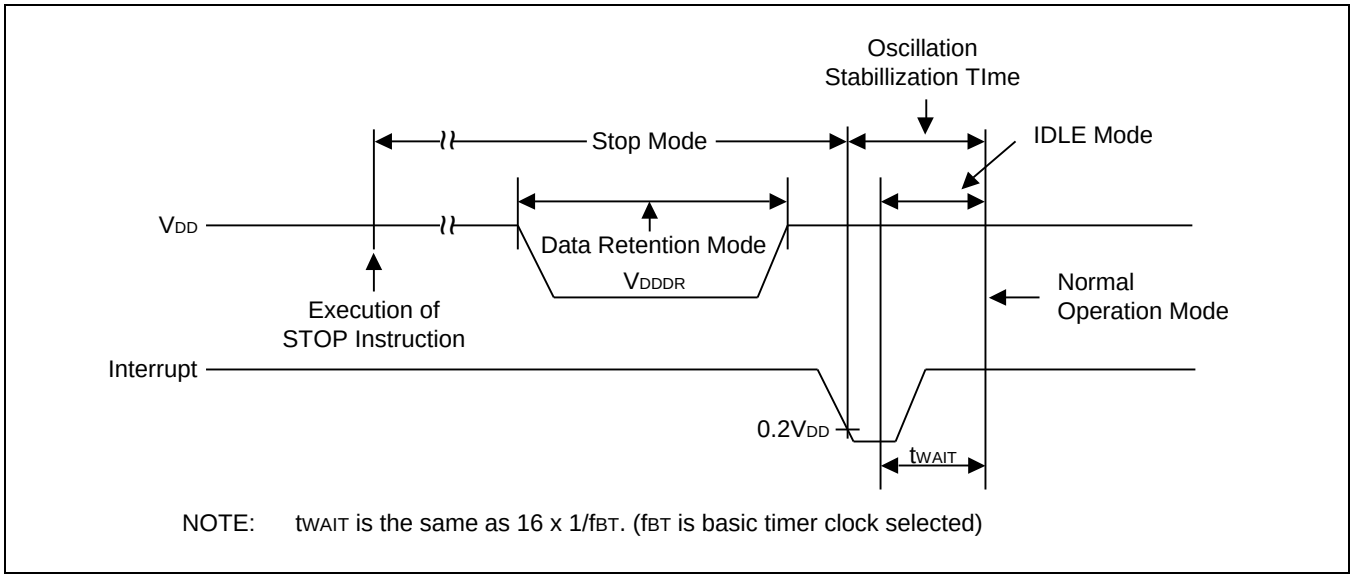


Figure 22-4 Stop Mode Release Timing Initiated by Interrupts

22.7 A/D Converter Electrical Characteristics

Table 22.6 A/D Converter Electrical Characteristics

(T_A = – 25 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Resolution	–	–	–	10	–	bit
Total accuracy	–	AVREF = 5.12 V V_{SS} = 0 V CPU clock = 12.0 MHz	–	–	± 3	LSB
Integral linearity error	ILE		–	–	± 2	
Differential linearity error	DLE			–	± 1	
Offset error of top	EOT			± 1	± 3	
Offset error of bottom	EOB			± 1	± 3	
Conversion time (1)	T_{CON}	–	25	–	–	µS
Analog input voltage	V_{IAN}	–	AV_{SS}	–	AV_{REF}	V
Analog input impedance	R_{AN}	–	2	–	–	MΩ
Analog reference voltage	AV_{REF}	–	1.8	–	V_{DD}	V
Analog ground	AV_{SS}	–	V_{SS}	–	V_{SS} + 0.3	
Analog input current	I_{ADIN}	V_{DD} = 5.0 V	–	–	10	µA
Analog block current (2)	I_{ADC}	V_{DD} = 5.0 V	–	0.5	1.5	mA
		V_{DD} = 5.0 V When power down mode		100	500	nA

NOTE:

1. "Conversion time" is the time required from the moment a conversion operation starts until it ends.
2. I_{ADC} is an operating current during A/D converter.

22.8 Low Voltage Reset Electrical Characteristics

Table 22.7 Low Voltage Reset Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Test Condition	Min.	Typ.	Max.	Unit
Voltage of LVR	V_{LVR}	T_A = 25 °C	1.8	1.9	2.0	V
			2.0	2.2	2.4	
V_{DD} voltage rising time	t_R	–	10	–	–	µS
V_{DD} voltage off time	t_{OFF}	–	0.5	–	–	S
Hysteresis	ΔV	–	–	50	150	mV
Current consumption	I_{LVR}	V_{DD} = 3.0 V	–	30	60	µA

NOTE: The current of LVR circuit is consumed when LVR is enabled by "Smart Option".

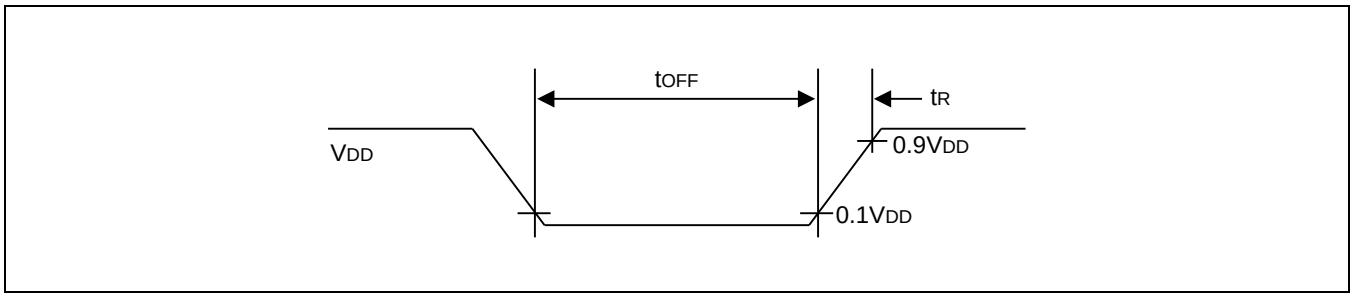


Figure 22-5 LVR (Low Voltage Reset) Timing

22.9 Synchronous SIO Electrical Characteristics

Table 22.8 Synchronous SIO Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
SCK Cycle time	t_{KCY}	External SCK source	1,000	–	–	ns
		Internal SCK source	1,000			
SCK high, low width	t_{KH}, t_{KL}	External SCK source	500			
		Internal SCK source	t_{KCY}/2-50			
SI setup time to SCK high	t_{SIK}	External SCK source	250			
		Internal SCK source	250			
SI hold time to SCK high	t_{KSI}	External SCK source	400			
		Internal SCK source	400			
Output delay for SCK to SO	t_{KSO}	External SCK source	–		300	
		Internal SCK source			250	

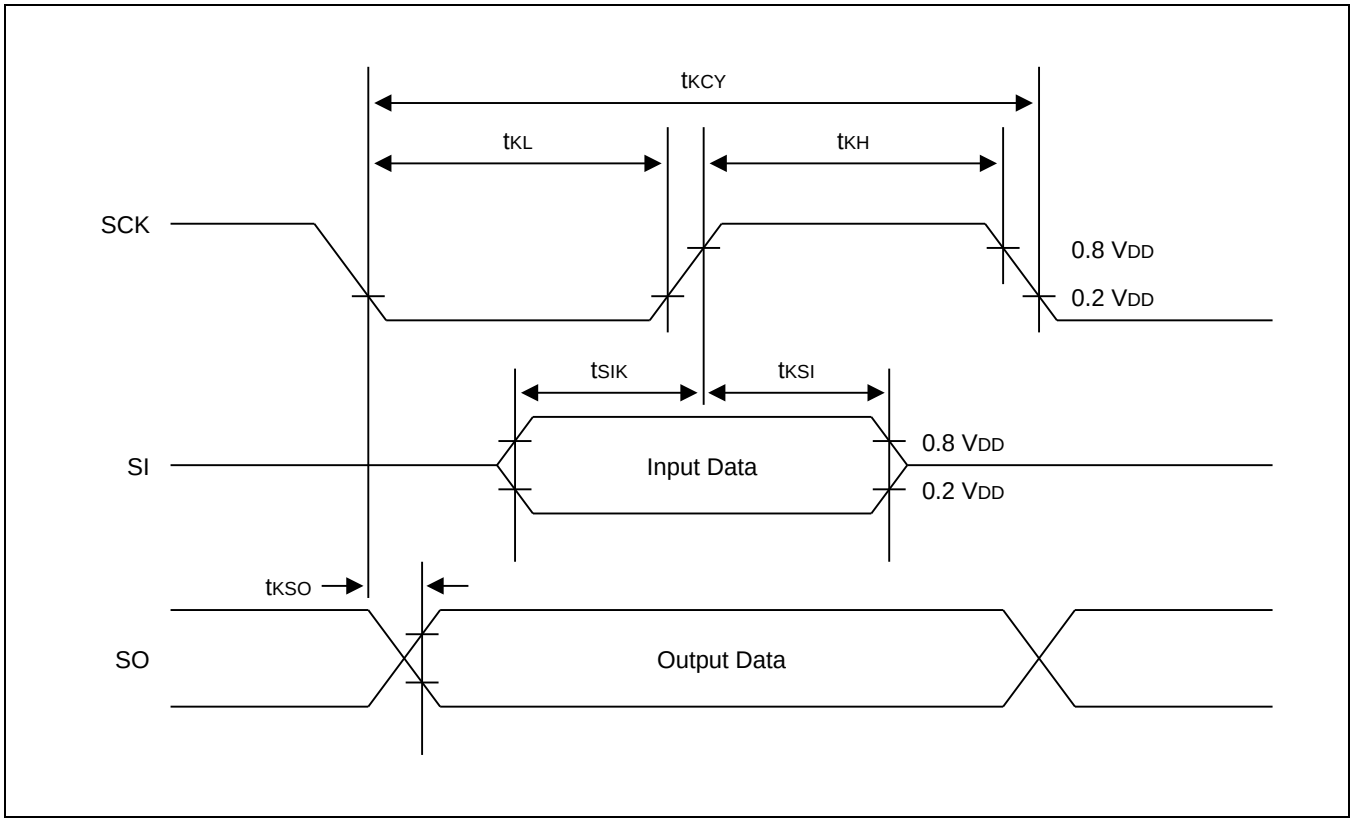


Figure 22-6 Serial Data Transfer Timing

22.10 UART Timing Characteristics

Table 22.9 UART Timing Characteristics in Mode 0 (12.0 MHz)

(T_A = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V, Load Capacitance = 80 pF)

Parameter	Symbol	Min.	Typ.	Max.	Unit
Serial port clock cycle time	t_{SCK}	1,160	$t_{CPU} \times 16$	1,500	ns
Output data setup to clock rising edge	t_{S1}	500	$t_{CPU} \times 13$	–	
Clock rising edge to input data valid	t_{S2}	–	–	500	
Output data hold after clock rising edge	t_{H1}	t_{CPU} – 50	t_{CPU}	–	
Input data hold after clock rising edge	t_{H2}	0	–	–	
Serial port clock High, Low level width	t_{HIGH}, t_{LOW}	450	t_{CPU} □ 8	890	

NOTE:

1. All timings are in nanoseconds (ns) and assume a 12.0-MHz CPU clock frequency.
2. The unit t_{CPU} means one UART clock period.

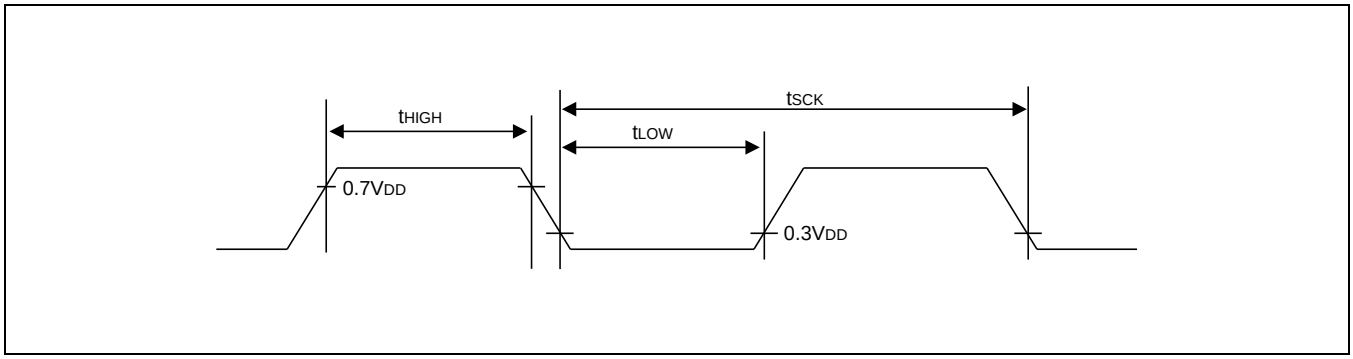
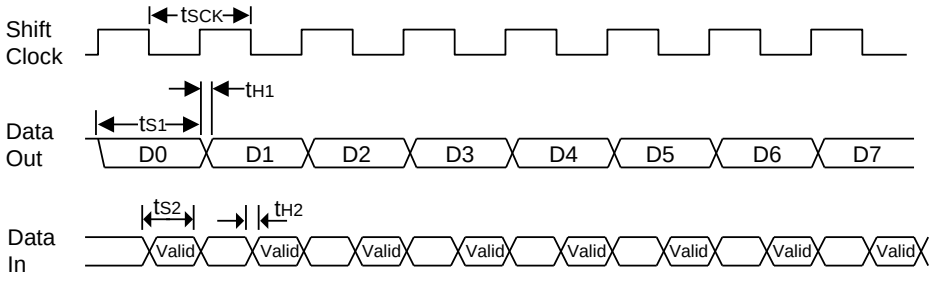


Figure 22-7 Waveform for UART Timing Characteristics



NOTE: The symbols shown in this diagram are defined as follows:

f_{SCK}	Serial port clock cycle time
t_{S1}	Output data setup to clock rising edge
t_{S2}	Clock rising edge to input data valid
t_{H1}	Output data hold after clock rising edge
t_{H2}	Input data hold after clock rising edge

Figure 22-8 Timing Waveform for the UART Module

22.11 LCD Capacitor Bias Electrical Characteristics

Table 22.10 LCD Capacitor Bias Electrical Characteristics (Normal and Idle Mode)

(T_A = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions		Min.	Typ.	Max.	Unit
Liquid crystal drive voltage	V_{LC3}	Connect a 1MΩ load resistance between V_{LC3} and V_{SS} V_{DD} = 2.0V to 5.5V (No panel load) 1/4 Bias	LMOD.2-.0 = 0	Typ.×0.85	0.90	Typ.×1.15	V
			LMOD.2-.0 = 1		0.95		
			LMOD.2-.0 = 2		1.00		
			LMOD.2-.0 = 3		1.05		
			LMOD.2-.0 = 4		1.10		
			LMOD.2-.0 = 5		1.15		
			LMOD.2-.0 = 6		1.20		
			LMOD.2-.0 = 7		1.25		
	V_{LC3}	1/3 Bias	LMOD.2-.0 = 0	Typ.×0.85	1.05	Typ.×1.15	
			LMOD.2-.0 = 1		1.125		
			LMOD.2-.0 = 2		1.20		
			LMOD.2-.0 = 3		1.275		
			LMOD.2-.0 = 4		1.350*		
			LMOD.2-.0 = 5		1.425*		
			LMOD.2-.0 = 6		1.50*		
	V_{LC3}	1/2 Bias	LMOD.2-.0 = 0	Typ.×0.85	1.10	Typ.×1.15	
			LMOD.2-.0 = 1		1.20		
			LMOD.2-.0 = 2		1.30*		
			LMOD.2-.0 = 3		1.40*		
			LMOD.2-.0 = 4		1.50*		
	V_{LC2}	Connect a 1 MΩ load resistance between V_{LC2} and V_{SS} (No panel load)		$2\times V_{LC3}\times 0.9$	–	$2\times V_{LC3}\times 1.1$	
	V_{LC1}	Connect a 1 MΩ load resistance between V_{LC1} and V_{SS} (No panel load)		$3\times V_{LC3}\times 0.9$	–	$3\times V_{LC3}\times 1.1$	
	V_{LC0}	Connect a 1 MΩ load resistance between V_{LC0} and V_{SS} (No panel load)		$4\times V_{LC3}\times 0.9$	–	$4\times V_{LC3}\times 1.1$	

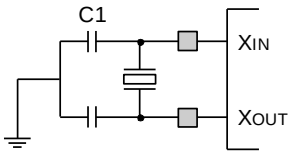
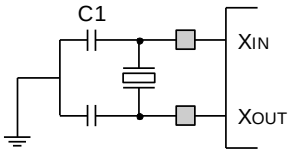
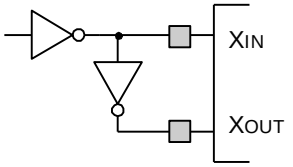
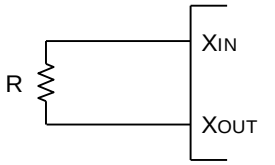
NOTE: It is characteristics when a 1 MΩ load resistance is connected to only a selected symbol (V_{LC0} - V_{LC3}) node at V_{DD} = 2.0 V to 5.5 V. The value of V_{LCN} (N = 0 to 2) is determined by VLC3 that is measured.

* Unsupported modes for Sub Idle and Stop mode operation.

22.12 Main Oscillator Characteristics

Table 22.11 Main Oscillator Characteristics

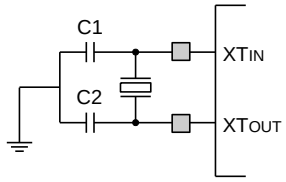
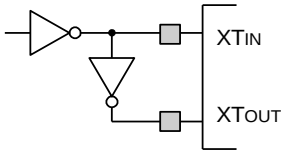
(T_A = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Oscillator	Clock Configuration	Parameter	Test Condition	Min.	Typ.	Max.	Units
Crystal		Main oscillation frequency	2.2 V – 5.5 V	0.4	–	12.0	MHz
			1.8 V – 5.5 V	0.4	–	4.2	
Ceramic Oscillator		Main oscillation frequency	2.2 V – 5.5 V	0.4	–	12.0	
			1.8 V – 5.5 V	0.4	–	4.2	
External Clock		X_{IN} input frequency	2.2 V – 5.5 V	0.4	–	12.0	
			1.8 V – 5.5 V	0.4	–	4.2	
RC Oscillator		Frequency	5.0V	0.4	–	2	MHz
			3.0V	0.4	–	1	

22.13 Sub Oscillation Characteristics

Table 22.12 Sub Oscillation Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Oscillator	Clock Configuration	Parameter	Test Condition	Min.	Typ.	Max.	Units
Crystal		Sub oscillation frequency	1.8 V – 5.5 V	–	32.768	–	kHz
External clock		XT_{IN} input frequency	1.8 V – 5.5 V	32	–	100	

22.14 Main Oscillation Stabilization Time

Table 22.13 Main Oscillation Stabilization Time

(T_A = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Oscillator	Test Condition	Min.	Typ.	Max.	Unit
Crystal	fx > 1 MHz Oscillation stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	40	ms
Ceramic		–	–	10	
External clock	X_{IN} input high and low width (t_{XH}, t_{XL})	62.5	–	1250	ns

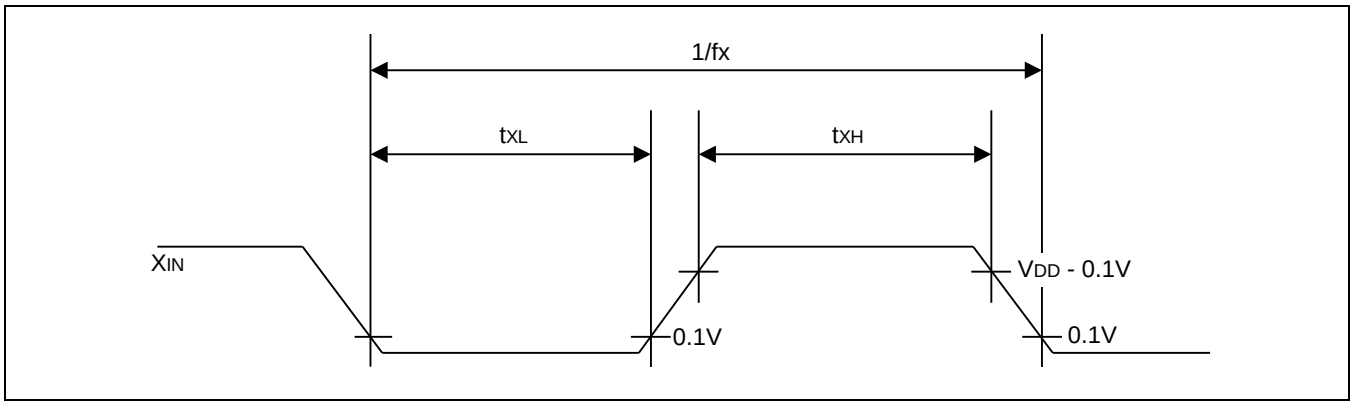


Figure 22-9 Clock Timing Measurement at X_{IN}

22.15 Sub Oscillation Stabilization Time

Table 22.14 Sub Oscillation Stabilization Time

(T_A = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Oscillator	Test Condition	Min.	Typ.	Max.	Unit
Crystal	–	–	–	10	s
External clock	XT_{IN} input high and low width (t_{XTH}, t_{XTL})	5	–	15	µs

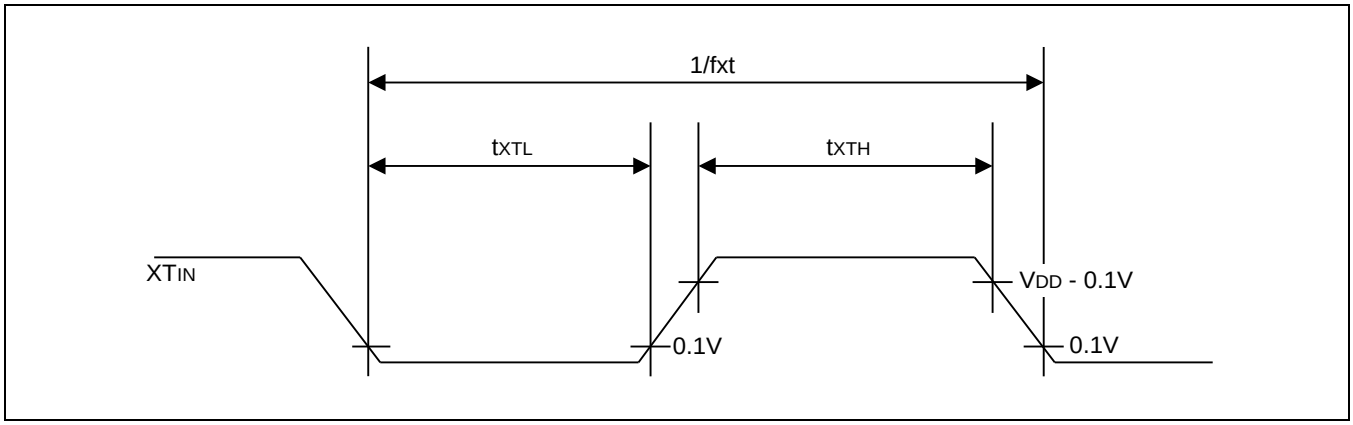
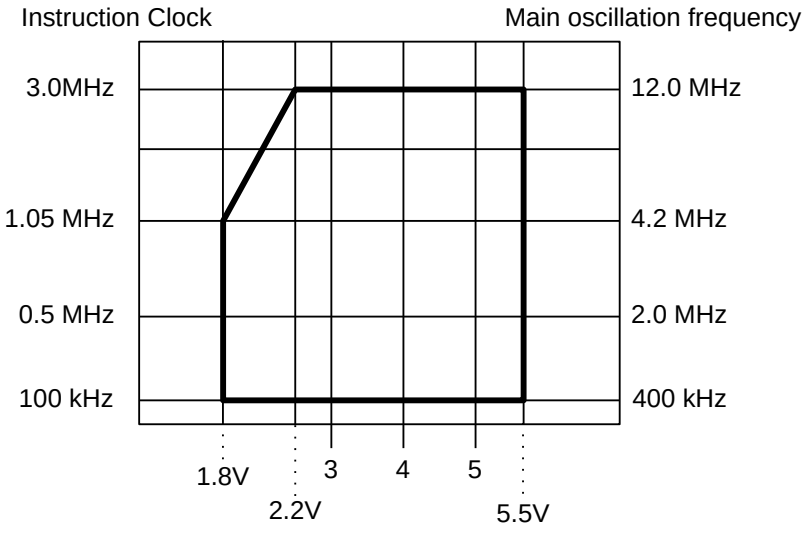


Figure 22-10 Clock Timing Measurement at XT_{IN}

22.16 Operating Voltage Range



CPU Clock = 1/4n x oscillator frequency (n = 1, 2, 8, 16)

Figure 22-11 Operating Voltage Range

22.17 Internal Flash ROM Electrical Characteristics

Table 22.15 Internal Flash ROM Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Programming time (1)	Ftp	–	20	25	30	μs
Chip erasing Time (2)	Ftp1		32	50	70	ms
Sector erasing Time (3)	Ftp2		4	8	12	ms
Read frequency	fR	–	–	–	12	MHz
Number of writing/erasing	FN_{WE}	–	10,000 (4)	–	–	Times

NOTE:

1. The Programming time is the time during which one byte (8-bit) is programmed.
2. The Chip erasing time is the time during which all 64KB block is erased.
3. The Sector erasing time is the time during which all 128 byte block is erased.
4. The Chip erasing is available in Tool Program Mode only.

23 Mechanical Data

23.1 Overview

The S3F8S6B microcontroller is currently available in 64-pin-QFP/LQFP/SDIP package.

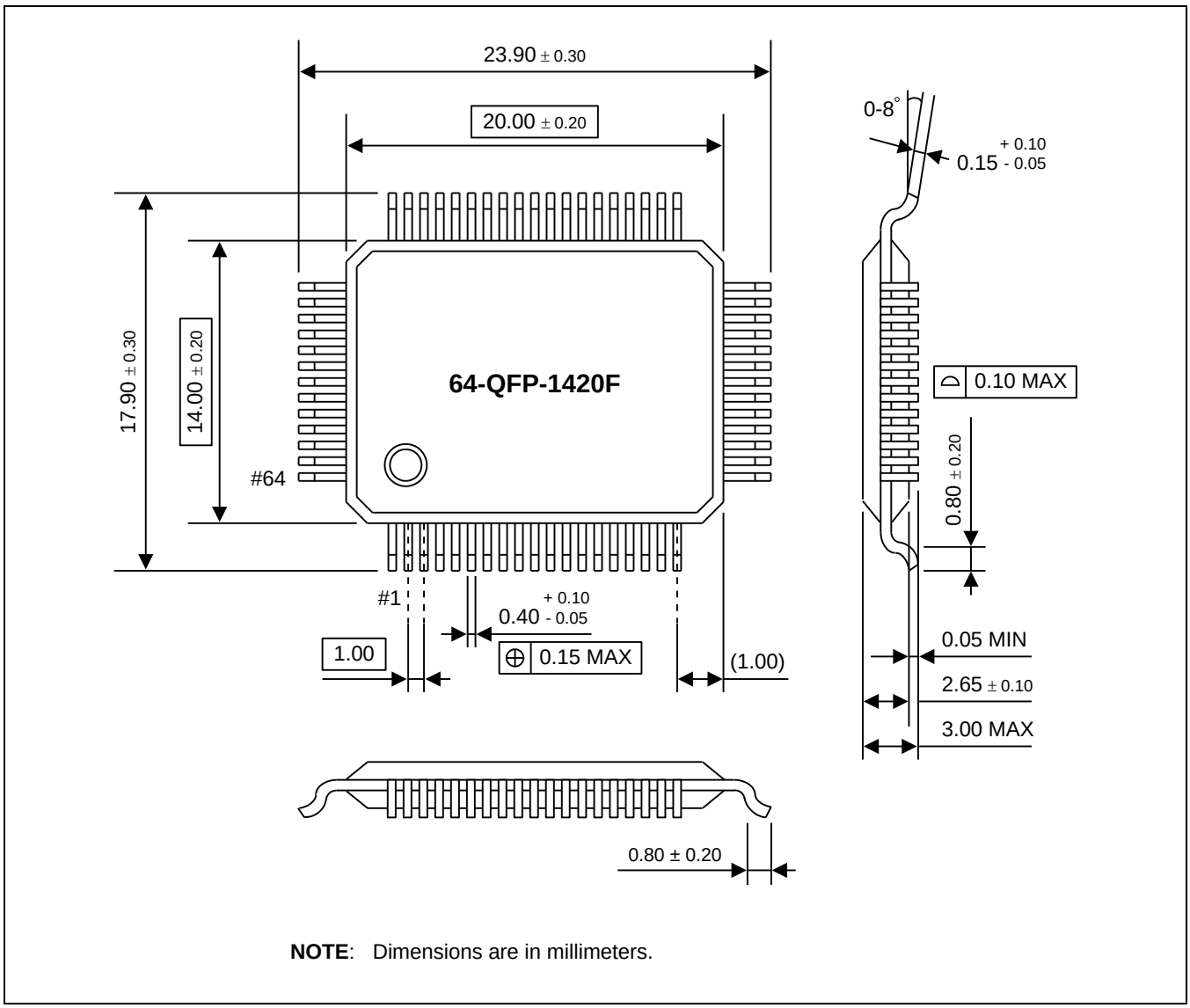
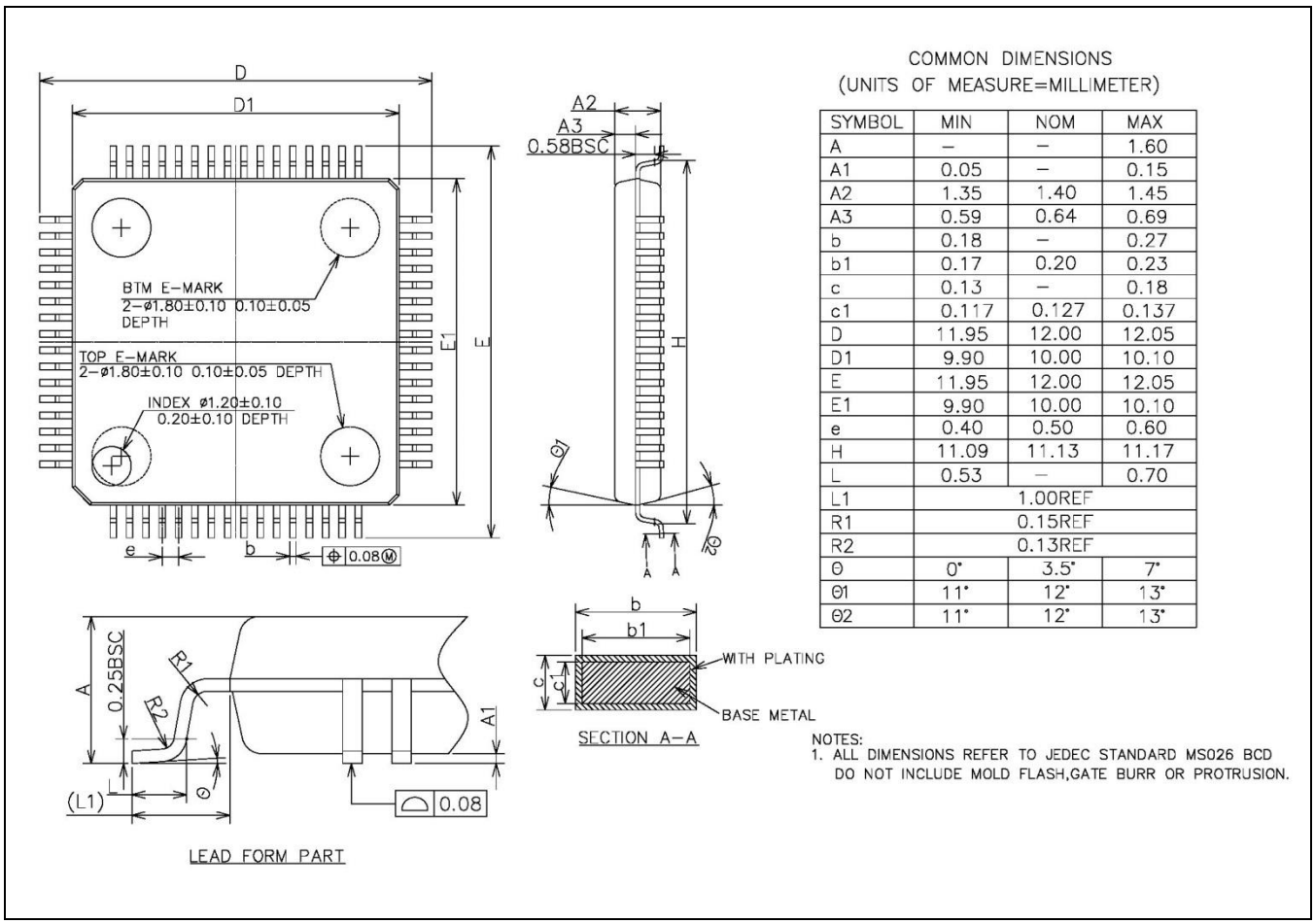


Figure 23-1 Package Dimensions (64-QFP-1420F)



SYMBOL	MIN	NOM	MAX
A	–	–	1.60
A1	0.05	–	0.15
A2	1.35	1.40	1.45
A3	0.59	0.64	0.69
b	0.18	–	0.27
b1	0.17	0.20	0.23
c	0.13	–	0.18
c1	0.117	0.127	0.137
D	11.95	12.00	12.05
D1	9.90	10.00	10.10
E	11.95	12.00	12.05
E1	9.90	10.00	10.10
e	0.40	0.50	0.60
H	11.09	11.13	11.17
L	0.53	–	0.70
L1	1.00REF		
R1	0.15REF		
R2	0.13REF		
θ	0°	3.5°	7°
θ1	11°	12°	13°
θ2	11°	12°	13°

NOTES:
1. ALL DIMENSIONS REFER TO JEDEC STANDARD MS026 BCD
DO NOT INCLUDE MOLD FLASH,GATE BURR OR PROTRUSION.

Figure 23-2 Package Dimensions (64-LQFP-1010)

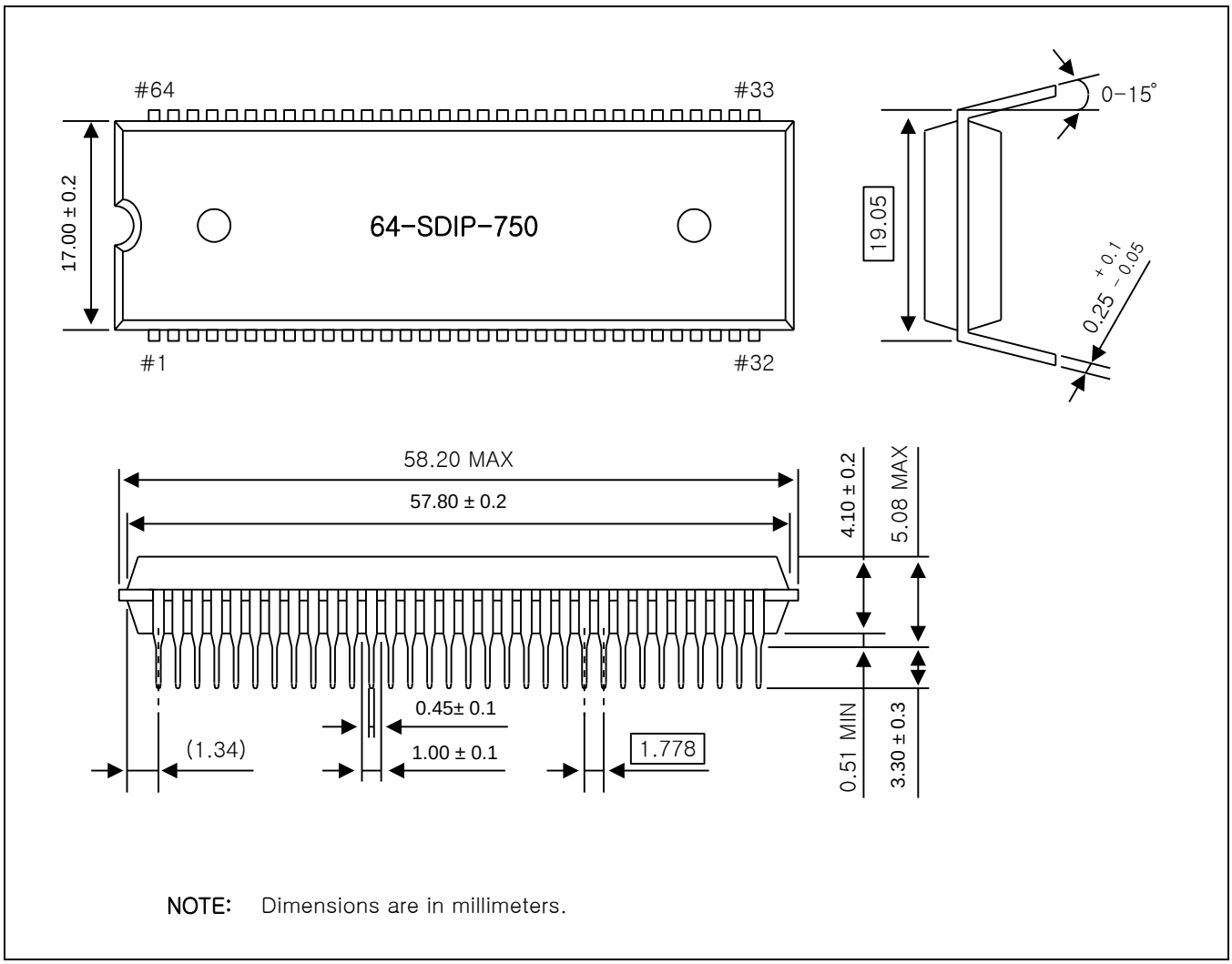


Figure 23-3 Package Dimensions (64-SDIP-750)

24 S3F8S6B Flash MCU

24.1 Overview

The S3F8S6B single-chip CMOS microcontroller is the Flash MCU. It has an on-chip Flash MCU ROM. The Flash ROM is accessed by serial data format.

NOTE: This chapter is about the Tool Program Mode of Flash MCU. If you want to know the User Program Mode, refer to the chapter 21. Embedded Flash Memory Interface.

24.2 Pin Assignments

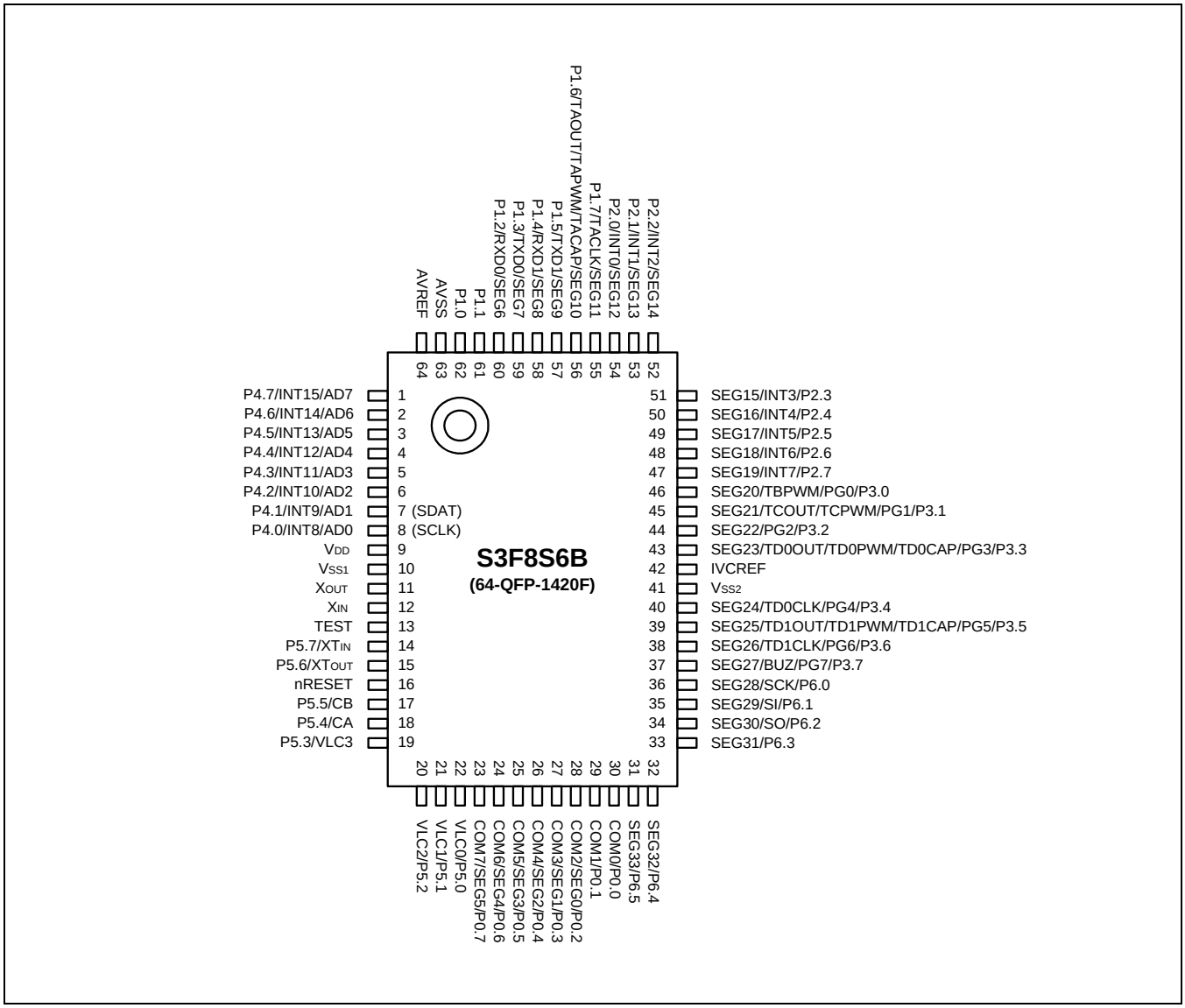


Figure 24-1 S3F8S6B Pin Assignments (64-QFP-1420F)

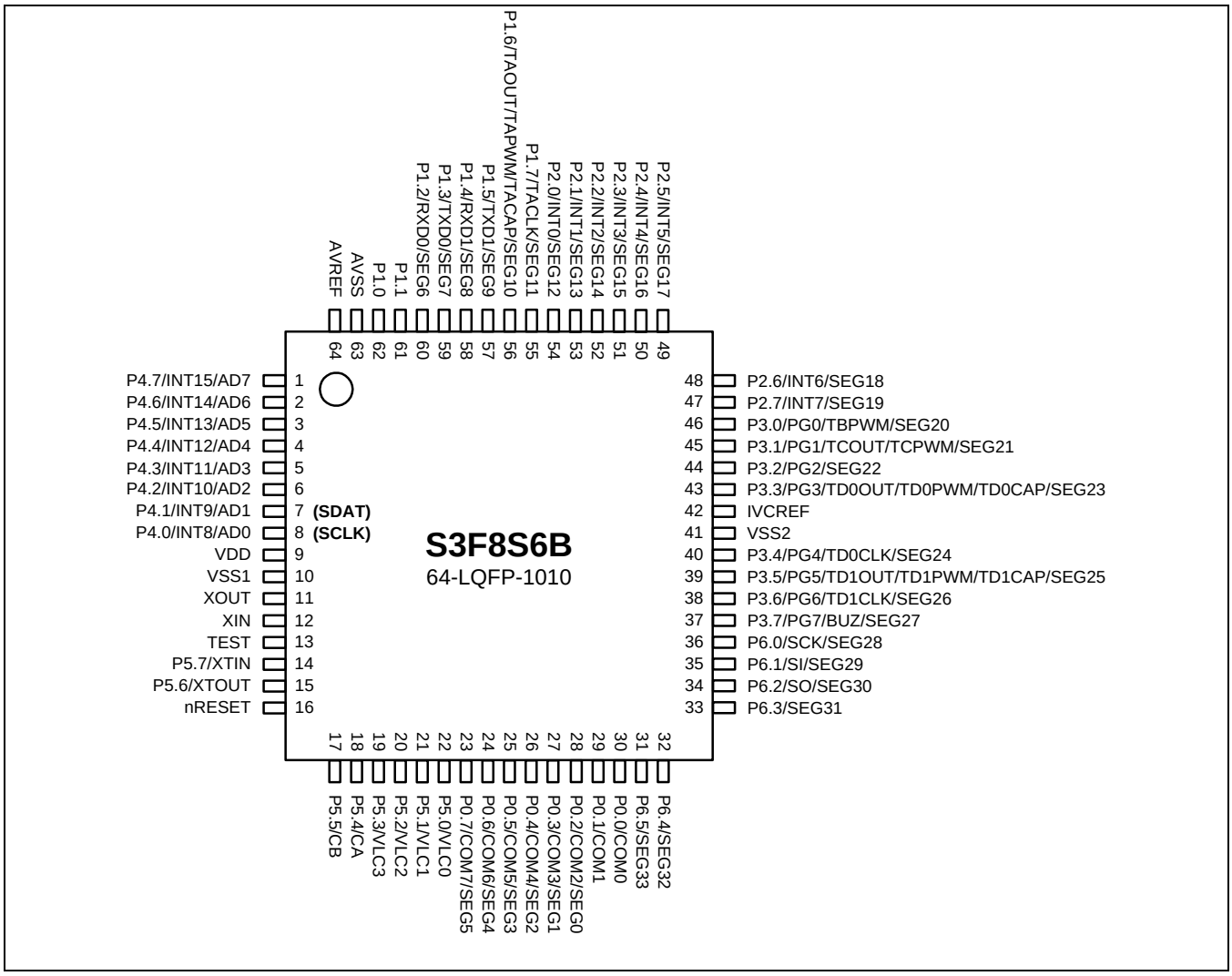


Figure 24-2 S3F8S6B Pin Assignments (64-LQFP-1010)

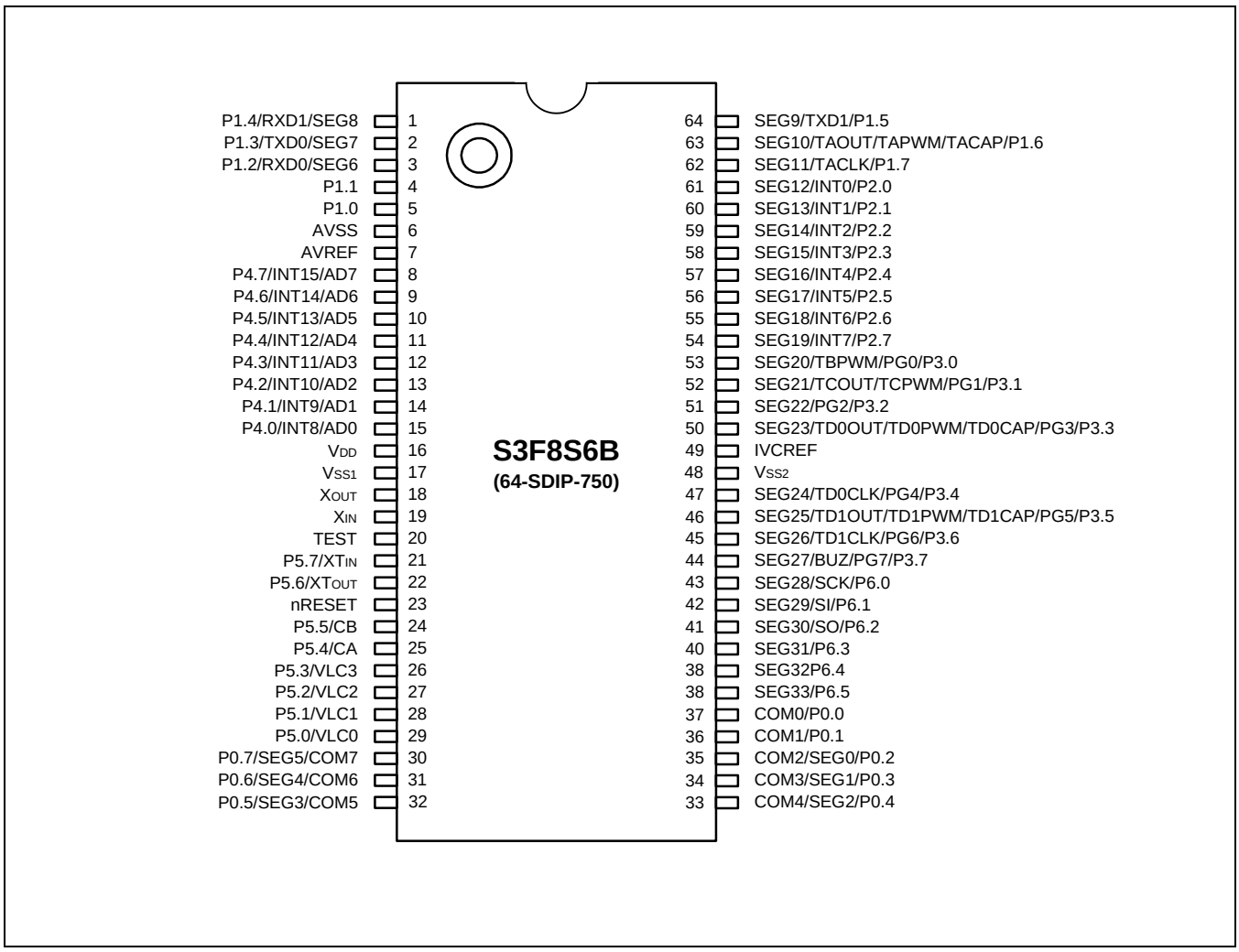


Figure 24-3 S3F8S6B Pin Assignments (64-SDIP-750)

Table 24.1 Descriptions of Pins Used to Read/Write the Flash ROM

Main Chip Pin Name	During Programming			
	Pin Name	Pin No.	I/O	Function
P4.1	SDAT	7(14)	I/O	Serial data pin. Output port when reading and input port when writing. Can be assigned as a Input/push-pull output port.
P4.0	SCLK	8(15)	I/O	Serial clock pin. Input only pin.
TEST	V_{PP}	13(20)	I	Tool mode selection when TEST/ V_{PP} pin sets Logic value "1". If user uses the Flash writer tool mode (ex.spw2+ etc..), user should be connected TEST/V_{PP} pin to V_{DD}. (S3F8S6B supplies high voltage 12.5V by internal high voltage generation circuit.)
nRESET	nRESET	16(23)	I	Chip Initialization
IVCREF	IVCREF	42(49)	–	A capacitor (0.1 μF) must be connected between IVCREF and V_{SS}.
V_{DD}, V_{SS1}	V_{DD}, V_{SS}	9(16), 10(17)	–	Power supply pin for logic circuit. V_{DD} should be tied to 5V during programming.

NOTE: Parentheses indicate pin number for 64-SDIP-750 package.

Test Pin Voltage

The TEST pin on socket board for MTP writer must be connected to VDD (5.0V) with RC delay as the figure 24-3 (only when SPW 2+ and GW-pro2 are used to). The TEST pin on socket board must not be connected Vpp (12.5V) which is generated from MTP Writer. So the specific socket board for S3F8S6B *must be used, when writing or erasing using MTP writer.*

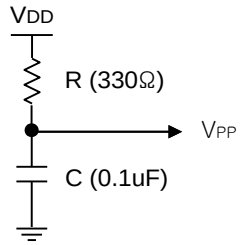


Figure 24-4 RC Delay Circuit

24.3 On Board Writing

The S3F8S6B needs only 6 signal lines including VDD and GND pins for writing internal Flash memory with serial protocol. Therefore the on-board writing is possible if the writing signal lines are considered when the PCB of application board is designed.

24.3.1 Circuit Design Guide

At the Flash writing, the writing tool needs 6 signal lines that are GND, VDD, nRESET, TEST, SDAT and SCLK. When you design the PCB circuits, you should consider the usage of these signal lines for the on-board writing.

In case of TEST pin, normally test pin is connected to GND but in writing mode the programming these two cases, a resistor should be inserted between the TEST pin and GND. The nRESET, SDAT and SCLK should be treated under the same consideration.

Please be careful to design the related circuit of these signal pins because rising/falling timing of VPP, SCLK and SDAT is very important for proper programming.

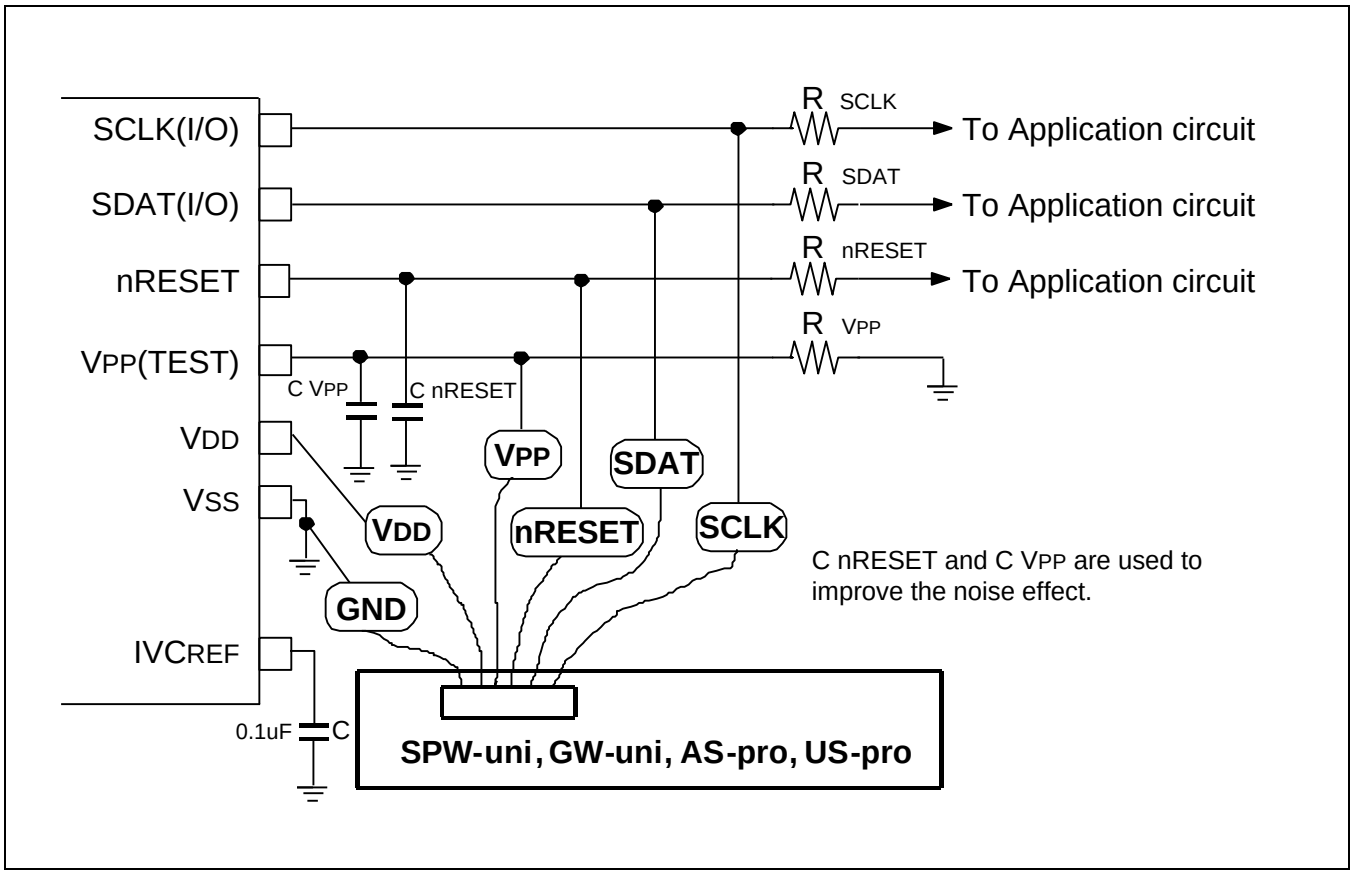


Figure 24-5 PCB Design Guide for on Board Programming

NOTE: If writer tool is the SPW 2+ and GW-pro2, reference to the page 24-4.

24.3.2 Reference Table for Connection

Table 24.2 Reference Table for Connection

Pin Name	I/O mode in Applications	Resistor (Need)	Required value
V_{PP} (TEST)	Input	Yes	RVpp is 10 kΩ to 50 kΩ. CVpp is 0.01 μF to 0.02 μF.
nRESET	Input	Yes	RnRESET is 2 kΩ to 5 kΩ. CnRESET is 0.01 μF to 0.02 μF.
SDAT (I/O)	Input	Yes	RSDAT is 2 kΩ to 5 kΩ.
	Output	No (NOTE)	–
SCLK (I/O)	Input	Yes	RSCLK is 2 kΩ to 5 kΩ.
	Output	No (NOTE)	–

NOTE:

1. In on-board writing mode, very high-speed signal will be provided to pin SCLK and SDAT. And it will cause some damages to the application circuits connected to SCLK or SDAT port if the application circuit is designed as high speed response such as relay control circuit. If possible, the I/O configuration of SDAT, SCLK pins had better be set to input mode.
2. The value of R, C in this table is recommended value. It varies with circuit of system.

25 Development Tools

25.1 Overview

Zilog provides a powerful and easy-to-use development support system on a turnkey basis. This development support system is composed of a host system, debugging tools, and supporting software. Any standard computer running Windows 7 (32-/64-bit), Windows Vista (32-/64-bit), and Windows XP operating systems can be used as a host

A sophisticated debugging tool is provided both in hardware and software: the powerful in-circuit emulator, OPENice-i500 and SK-1200, for the S3C7-, S3C9- and S3C8- microcontroller families. Zilog also offers supporting software that includes, debugger, an assembler, and a program for setting options.

25.1.1 Target Boards

Target boards are available for all the S3C8/S3F8-series microcontrollers. All the required target system cables and adapters are included on the device-specific target board. TB8S6B is a specific target board for the development of application systems using S3F8S6B.

25.1.2 Programming Socket Adapter

When you program S3F8S6B's Flash memory by using an emulator or OTP/MTP writer, you need a specific programming socket adapter for S3F8S6B.

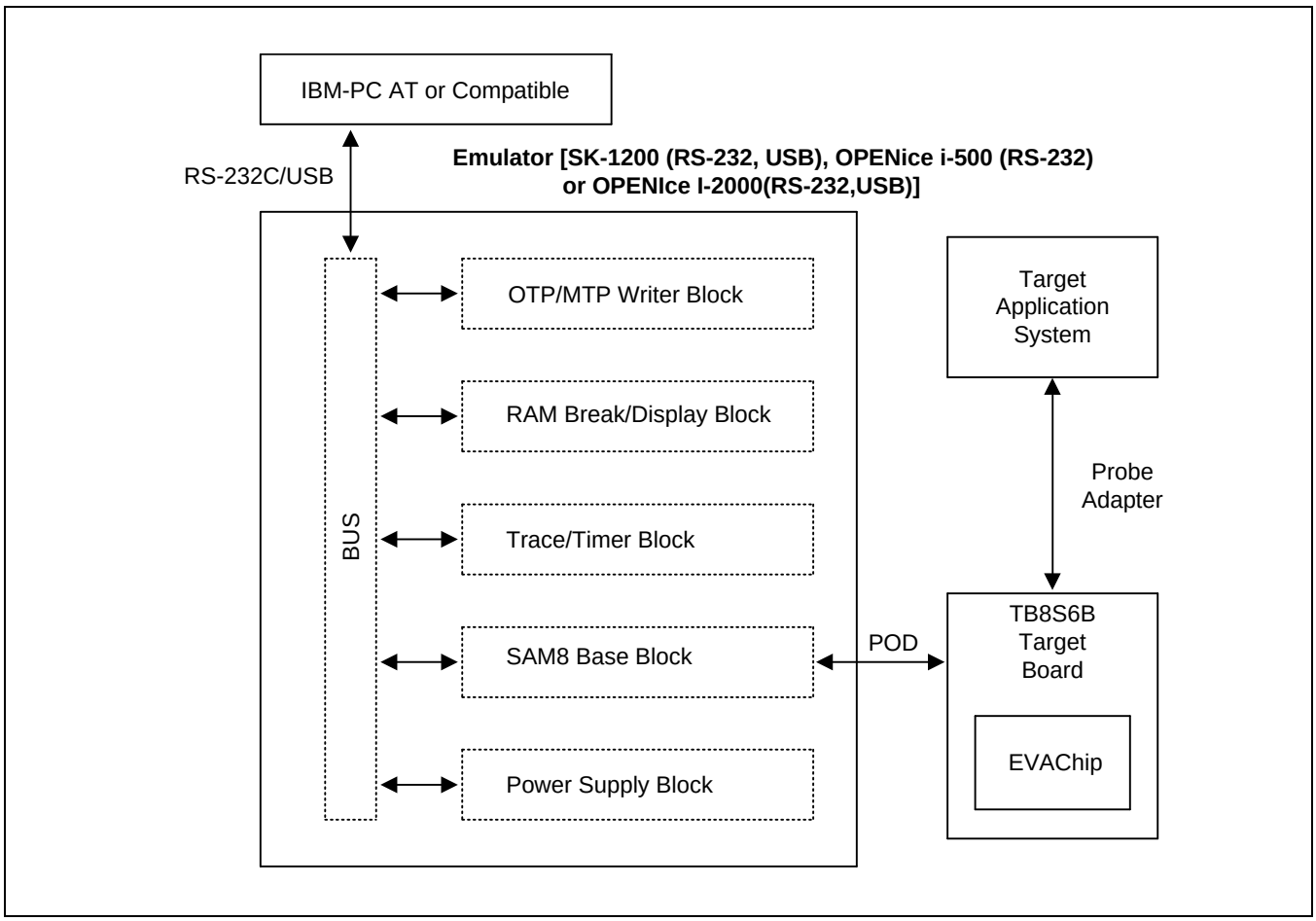


Figure 25-1 Emulator Product Configuration

25.2 TB8S6B Target Board

The TB8S6B target board can be used for development of the S3F8S6B microcontroller.

The TB8S6B target board is operated as target CPU with Emulator (SK-1200, OPENice-i500/2000)).

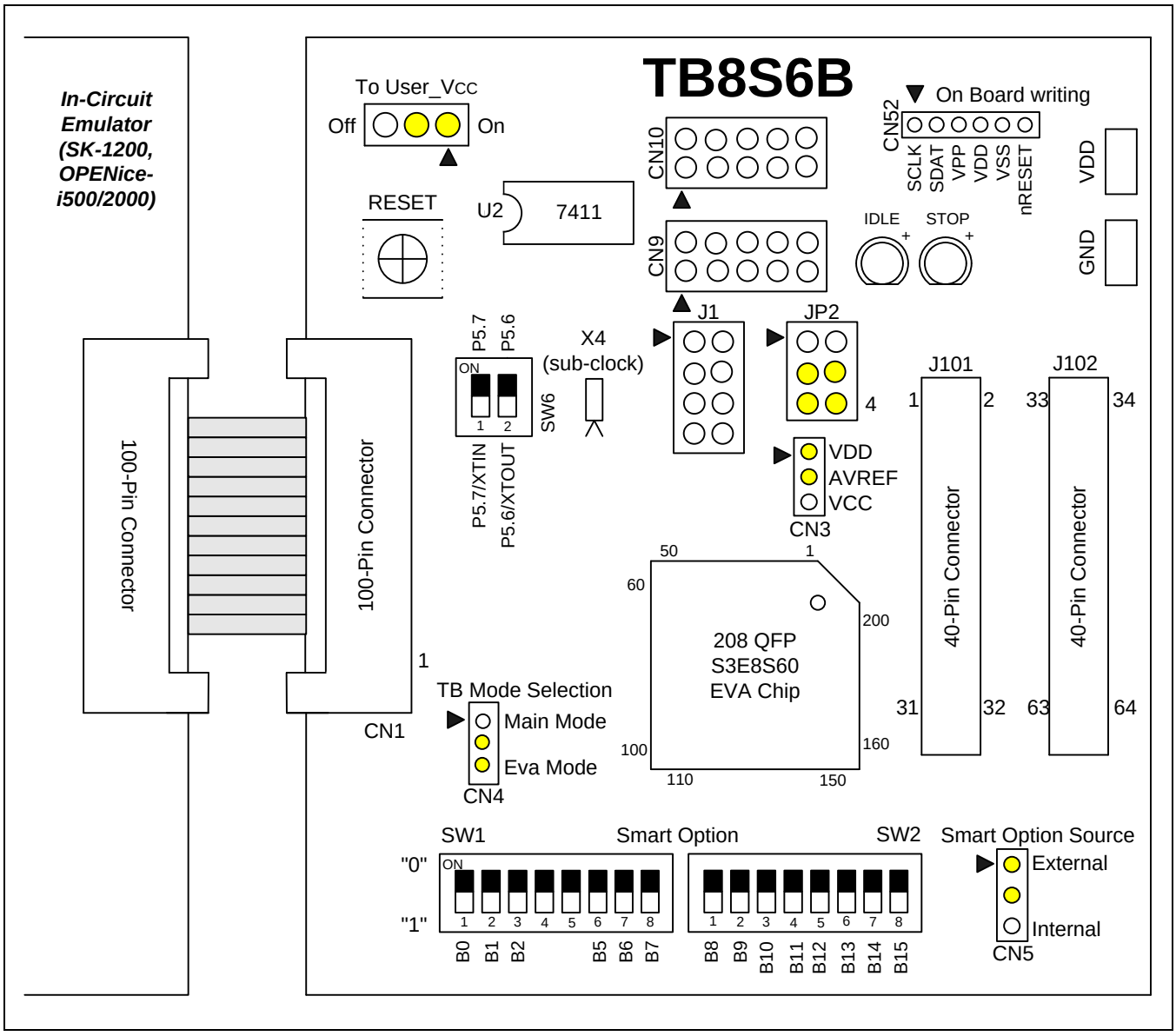


Figure 25-2 TB8S6B Target Board Configuration

NOTE: The symbol "◄" marks start point of jumper signals.

Table 25.1 Components of TB8S6B

Symbols	Usage	Description
CN1	100-pin connector	Connection between emulator and TB8S6B target board.
J101, J102	40-pin connector	Connection between target board and user application system
RESET	Push button	Generation low active reset signal to S3E8S60 EVA-chip
VCC, GND	POWER connector	External power connector for TB8S6B
STOP, IDLE LED	STOP/IDLE Display	Indicate the status of STOP or IDLE of S3E8S60 EVA-chip on TB8S6B target board

Table 25.2 Setting of the Jumper in TB8S6B

JP#	Description	1-2 Connection	2-3 Connection	Default Setting
CN3	AVREF power source	VDD	User power	Join 1-2
JP2	Clock source selection	When using the internal clock source which is generated from Emulator, join connector 2-3 and 4-5 pin. If user wants to use the external clock source like a crystal, user should change the jumper setting from 1-2 to 5-6 and connect J1 to an external clock source.		Emulator 2-3 4-5
J1	External clock source	Connecting points for external clock source		
CN4	Target board mode selection	MAIN-Mode	EVA-Mode	Join 2-3
CN5	Smart option source selection	The Smart Option is selected by external smart option switch (SW1&SW2)	The Smart Option is selected by internal smart option area (003EH–0003FH of ROM). But this selection is not available.	Join 1-2
SW1, SW2	Smart option selection	The Smart Option can be selected by this switch when the Smart Option source is selected by external. The B7–B0 are comparable to the 003EH.7–.0. The B15–B8 are comparable to the 003FH.7–.0. Refer to the page 2-3.		
CN9	LCD bias	Need capacitors When capacitor bias mode		
CN10		Need resistors When external resistor bias mode		
To User_Vcc	Target System is supplied V_{DD}	Target Board is supplied V_{DD} from user System.	Target Board is not supplied V_{DD} from user System.	Join 1-2

- IDLE LED
  This is LED is ON when the evaluation chip (S3E8S60) is in Idle mode.
- STOP LED
  This LED is ON when the evaluation chip (S3E8S60) is in Stop mode.

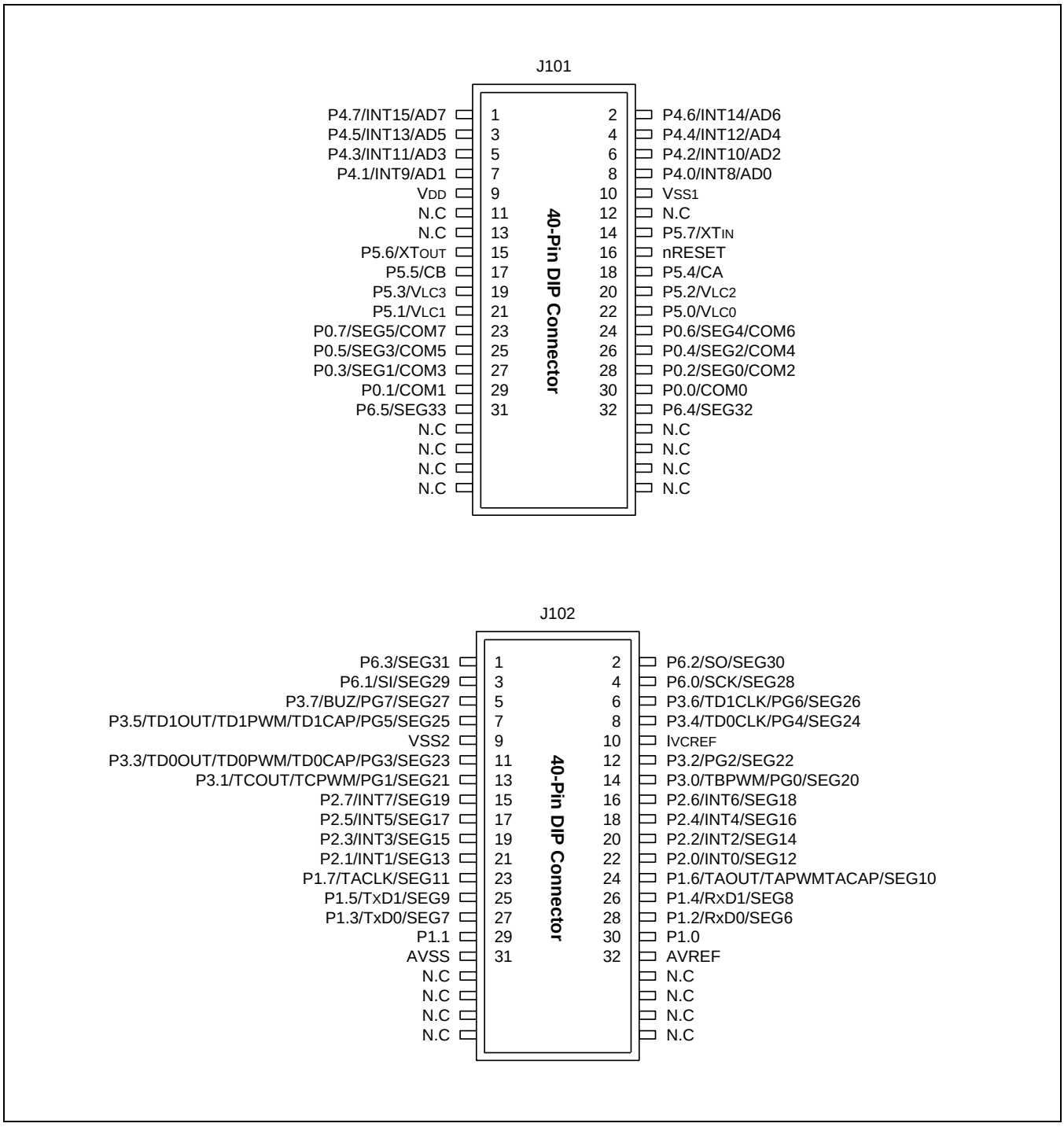


Figure 25-3 40-Pin Connectors (J101, J102) for TB8S6B

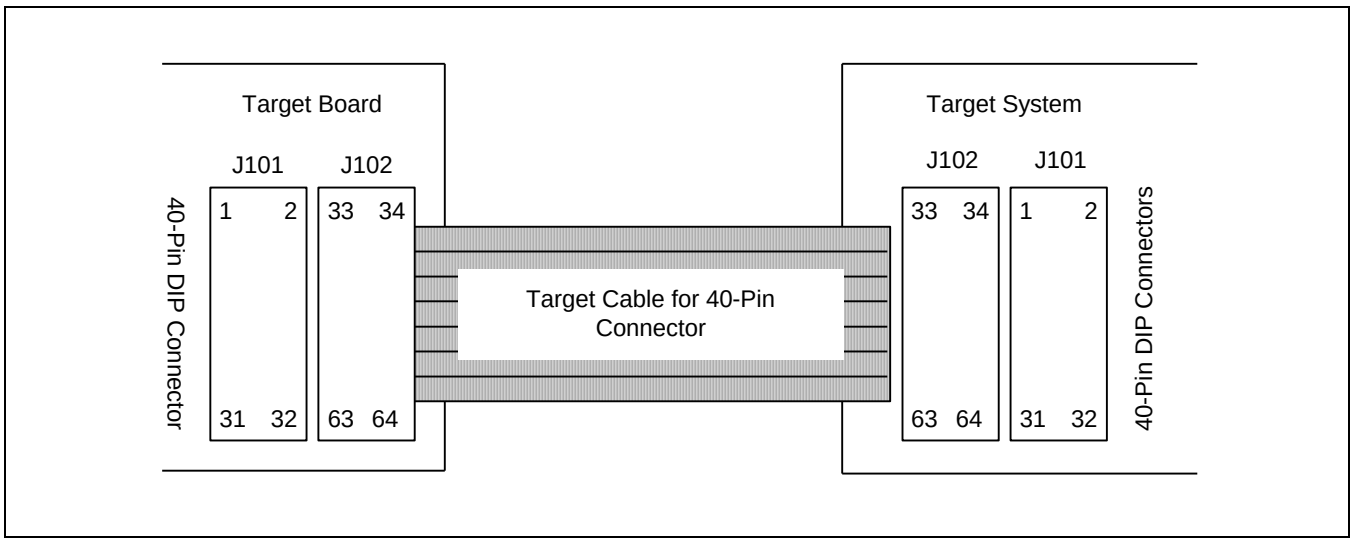


Figure 25-4 S3E8S60 Cables for 64-QFP Package

25.2.1 Third Parties for Development Tools

Zilog provides a complete line of development tools that support the S3 Family of Microcontrollers. With long experience in developing MCU systems, these third party firms are bonafide leaders in MCU development tool technology.

25.2.2 In-Circuit Emulators

- OPENice-i500/2000
- SK-1200 SmartKit

25.2.3 OTP/MTP Programmers

- GW-Uni2
- AS-Pro2
- Elnec programmers

To obtain the S3 Family development tools that will satisfy your S3F80QB development objectives, contact your local Zilog Sales Office, or visit Zilog's Third Party Tools page to review our list of third party tool suppliers.