THE DATASHEET OF

S3F8S39XZZ-LO89

www.win-source.net 0086-755-83957316

zilog

Embedded in Life

An IXYS Company

S3 Family 8-Bit Microcontrollers

S3F8S39/S3F8S35

Product Specification

PS031407-0219

S3 Microcontrollers

Copyright ©2019 Zilog®, Inc. All rights reserved.
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

©2019 Zilog, Inc. All rights reserved. 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concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded.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 2019	07	Corrected description for the Internal Oscillator Calibration Data Register and updated Figure 7-5.	7-4
Jul 2018	06	Added two 32-LQFP packages (S3F8S35EZZ, S3F8S39EZZ) to Chapter 27. Ordering Information; Updated Part Number Suffix Designation guide; Added Table 23-5 for Oscillator Characteristics for these parts; Corrected typos.	27-1; 27-2; 23-7; Various
Jan 2018	05	Added 32-LQFP package ─ updated Figures 1-3 and 25-2; added Package Outline drawing; updated Ordering Information.	1-4, 1-6, 25-2; 24-4; 27-1
May 2017	04	Added Zilog Library-based Development Platform and updated to most current 3rd party tools. Added Chapter 27, Ordering Information Changed package type description from ELP to QFN. Removed 'Preliminary' from footer.	CH 26 27-1 Various All
Jan 2015	03	Updated the Third Parties for Development Tools section.	26-12
Apr 2014	02	Updated to include front matter, PDF bookmarks for easier navigation.	All
Aug 2013	01	Original Zilog issue.	All

Table of Contents

Table of Contents 5

List of Figures 14

List of Tables 18

List of Examples 20

1 Product Overview 1-1

1.1 S3F8 Series Microcontrollers 1-1
1.2 S3F8S39/S3F8S35 Microcontrollers 1-1
1.3 Features 1-2
1.3.1 CPU 1-2
1.3.2 Memory 1-2
1.3.3 Instruction Set 1-2
1.3.4 26 I/O Pins 1-2
1.3.5 Interrupts 1-2
1.3.6 Timers and Timer/Counters 1-2
1.3.7 Watch Timer 1-3
1.3.8 Analog to Digital Converter 1-3
1.3.9 Two Channels UART 1-3
1.3.10 Multi-Master IIC-Bus 1-3
1.3.11 SPI 1-3
1.3.12 Two Power-Down Modes 1-3
1.3.13 Oscillation Sources 1-4
1.3.14 Instruction Execution Time 1-4
1.3.15 Built-In Reset Circuit (LVR) 1-4
1.3.16 Low Voltage Detect Circuit (LVD) 1-4
1.3.17 Operating Voltage Range 1-4
1.3.18 Package Type 1-4
1.3.19 Operating Temperature Range 1-4
1.3.20 Smart Option 1-4
1.4 Block Diagram 1-5
1.5 Pin Assignments 1-5
1.6 Pin Descriptions 1-7
1.7 Pin Circuit Diagrams 1-9

2 Address Spaces 2-1

2.1 Overview 2-1
2.2 Program Memory (ROM) 2-2
2.3 Smart Option 2-3
2.4 Register Architecture 2-5
2.5 Register Page Pointer 2-7
2.6 Register Set 1 2-8
2.7 Register Set 2 2-8
2.8 Prime Register Space 2-9
2.9 Working Registers 2-10
2.10 Using the Register Points 2-11
2.11 Register Addressing 2-13
2.12 Common Working Register Area 2-15
2.13 4-Bit Working Register Addressing 2-17
2.14 8-Bit Working Register Addressing 2-19
2.15 System and User Stack 2-21

2.15.1 Stack Operations 2-21
2.15.2 User-Defined Stacks 2-21
2.15.3 Stack Pointers (SPL, SPH) 2-22

3 Addressing Modes 3-1

3.1 Overview 3-1
3.2 Register Addressing Mode (R) 3-2
3.3 Indirect Register Addressing Mode (IR) 3-3
3.4 Indexed Addressing Mode (X) 3-7
3.5 Direct Address Mode (DA) 3-10
3.6 Indirect Address Mode (IA) 3-12
3.7 Relative Address Mode (RA) 3-13
3.8 Immediate Mode (IM) 3-14

4 Control Registers 4-1

4.1 Overview 4-1
4.1.1 ADCON-A/D Converter Control Register (D2H, SET1) 4-8
4.1.2 BTCON-Basic Timer Control Register (D3H, SET1) 4-9
4.1.3 CLKCON-Clock Control Register (D4H, SET1) 4-10
4.1.4 FLAGS-System Flags Register (D5H, SET1) 4-11
4.1.5 FMCON-Flash Memory Control Register (04H, PAGE8) 4-12
4.1.6 FMSECH-Flash Memory Sector Address Register (High Byte) (01H, PAGE8) 4-13
4.1.7 FMSECL-Flash Memory Sector Address Register (Low Byte) (02H, PAGE8) 4-13
4.1.8 FMUSR-Flash Memory User Programming Enable Register (03H, PAGE8) 4-13
4.1.9 ICCR-Multi-Master IIC-Bus Clock Control Register (05H, PAGE8) 4-14
4.1.10 ICCR-Multi-Master IIC-Bus Clock Control Register (05H, PAGE8) 4-15
4.1.11 IMR-Interrupt Mask Register (DDH, SET1) 4-16
4.1.12 IPH-Instruction Pointer (High Byte) (DAH, SET1) 4-17
4.1.13 IPL-Instruction Pointer (Low Byte) (DBH, SET1) 4-17
4.1.14 IPR-Interrupt Priority Register (FFH, BANK0) 4-18
4.1.15 IRQ-Interrupt Request Register (DCH, SET1) 4-19
4.1.16 LVDCON-Low Voltage Detector Control Register (FFH, BANK 1) 4-20
4.1.17 OSCCALCON-Internal oscillator Calibration Control Register (18H, PAGE8) 4-21
4.1.18 P0CONH-Port 0 Control Register High Byte (E0H, BANK1) 4-22
4.1.19 P0CONL-Port 0 Control Register Low Byte (E1H, BANK1) 4-23
4.1.20 P0PUR-Port 0 Pull Up Resistor Enable Register (E2H, BANK1) 4-24
4.1.21 P1CONL-Port 1 Control Register Low Byte (E3H, BANK1) 4-25
4.1.22 P2CONH-Port 2 Control Register High Byte (E4H, BANK1) 4-26
4.1.23 P2CONL-Port 2 Control Register Low Byte (E5H, BANK1) 4-27
4.1.24 P2INT-Port 2 Interrupt Control Register (E6H, BANK1) 4-28
4.1.25 P2PUR-Port 2 Pull-Up Resistor Enable Register (E7H, BANK1) 4-29
4.1.26 P3CONH-Port 3 Control Register High Byte (E8H, BANK1) 4-30
4.1.27 P3CONL-Port 3 Control Register Low Byte (E9H, BANK1) 4-31
4.1.28 P3INT-Port 3 Interrupt Control Register (EAH, BANK1) 4-32
4.1.29 P3PUR-Port 3 Pull-Up Resistor Enable Register (EBH, BANK1) 4-33
4.1.30 PNE3-Port 3 N-Channel Open-Drain Mode Register (ECH, BANK1) 4-34
4.1.31 PINTPND-Interrupt Pending Register (F1H, BANK1) 4-35
4.1.32 PP-Register Page Pointer (DFH, SET1) 4-36
4.1.33 RESETID-Reset Source Indicating Register (00H, PAGE8) 4-37
4.1.34 ROSCCON-Ring Oscillator Control Register (FBH, BANK0) 4-38
4.1.35 RP0-Register Pointer 0 (D6H, SET1) 4-39
4.1.36 RP1-Register Pointer 1 (D7H, SET1) 4-39
4.1.37 SPH-Stack Pointer High Byte (D8H, SET1) 4-40
4.1.38 SPL-Stack Pointer Low Byte (D9H, SET1) 4-40
4.1.39 SPICON-SPI Control Register (F2H, BANK1) 4-41
4.1.40 SPISTAT-SPI Status Register (F3H, BANK1) 4-42

4.1.41 STPCON-Stop Mode Control Register (FDH, BANK1) 4-43
4.1.42 SWTCON-Stop Wake-Up Timer Control Register (FAH, Bank 0) 4-44
4.1.43 SYM-System Mode Register (DEH, SET1) 4-45
4.1.44 TACON-Timer A Control Register (E2H, BANK0) 4-46
4.1.45 TAPS-TA PreScaler (E3H, BANK0) 4-47
4.1.46 TBCON-Timer B Control Register (09H, Page 8) 4-48
4.1.47 TBTRG-Timer B Trigger Control Register (0EH, Page 8) 4-49
4.1.48 T0CON-Timer 0 Control Register (ECH, BANK0) 4-50
4.1.49 T0PS-T0 PreScalar (EDH, BANK0) 4-51
4.1.50 T1CON-Timer 1 Control Register (F2H, BANK0) 4-52
4.1.51 T1PS-T1 PreScalar Register (F3H, BANK0) 4-53
4.1.52 T2CON-Timer 2 Control Register (F8H, BANK0) 4-54
4.1.53 T2PS-T2 PreScalar Register (F9H, BANK0) 4-55
4.1.54 UART0CONH-UART 0 Control Register High Byte (F5H, BANK1) 4-56
4.1.55 UART0CONL-UART 0 Control Register Low Byte (F6H, BANK1) 4-57
4.1.56 UART1CONH-UART 1Control Register High Byte (F9H, BANK1) 4-58
4.1.57 UART1CONL-UART 1 Control Register Low Byte (FAH, BANK1) 4-59
4.1.58 WTCON-Watch Timer Control Register (FEH, BANK1) 4-60

5 Interrupt Structure 5-1

5.1 Overview 5-1
5.1.1 Levels 5-1
5.1.2 Vectors 5-1
5.1.3 Sources 5-1
5.2 Interrupt Types 5-2
5.3 S3F8S39/S3F8S35 Interrupt Structure 5-3
5.3.1 Interrupt Vector Addresses 5-4
5.3.2 Enable/Disable Interrupt Instructions 5-4
5.4 System-Level Interrupt Control Registers 5-5
5.5 Interrupt Processing Control Points 5-6
5.6 Peripheral Interrupt Control Registers 5-7
5.7 System Mode Register 5-9
5.8 Interrupt Mask Register 5-10
5.9 Interrupt Priority Register 5-11
5.10 Interrupt Request Register 5-13
5.11 Interrupt Pending Function Types 5-14
5.11.1 Overview 5-14
5.11.2 Pending Bits Cleared Automatically by Hardware 5-14
5.11.3 Pending Bits Cleared by the Service Routine 5-14
5.12 Interrupt Source Polling Sequence 5-15
5.12.1 Interrupt Service Routines 5-15
5.12.2 Generating Interrupt Vector Addresses 5-16
5.12.3 Nesting of Vectored Interrupts 5-16
5.12.4 Instruction Pointer 5-16
5.12.5 Fast Interrupt Processing 5-17
5.12.6 Procedure for Initiating Fast Interrupts 5-17
5.12.7 Fast Interrupt Service Routine 5-17
5.12.8 Relationship to Interrupt Pending Bit Types 5-18
5.13 Programming Guidelines 5-18

6 Instruction Set 6-1

6.1 Overview 6-1
6.1.1 Data Types 6-1
6.1.2 Register Addressing 6-1
6.1.3 Addressing Modes 6-1
6.2 Flags Register 6-5

6.3 Flag Descriptions 6-6
6.4 Instruction Set Notation 6-7
6.5 Condition Codes 6-11
6.6 Instruction Descriptions 6-12
6.6.1 ADC-Add with Carry 6-13
6.6.2 ADD-Add 6-14
6.6.3 AND-Logical AND 6-15
6.6.4 BAND-Bit AND 6-16
6.6.5 BCP-Bit Compare 6-17
6.6.6 BITC-Bit Complement 6-18
6.6.7 BITR-Bit Reset 6-19
6.6.8 BITS-Bit Set 6-20
6.6.9 BOR-Bit OR 6-21
6.6.10 BTJRF-Bit Test, Jump Relative on False 6-22
6.6.11 BTJRT-Bit Test, Jump Relative on True 6-23
6.6.12 BXOR-Bit XOR 6-24
6.6.13 CALL-Call Procedure 6-25
6.6.14 CCF-Complement Carry Flag 6-26
6.6.15 CLR-Clear 6-27
6.6.16 COM-Complement 6-28
6.6.17 CP-Compare 6-29
6.6.18 CPIJE-Compare, Increment, and Jump on Equal 6-30
6.6.19 CPIJNE-Compare, Increment, and Jump on Non-Equal 6-31
6.6.20 DA-Decimal Adjust 6-32
6.6.21 DA-Decimal Adjust 6-33
6.6.22 DEC-Decrement 6-34
6.6.23 DECW-Decrement Word 6-35
6.6.24 DI-Disable Interrupts 6-36
6.6.25 DIV-Divide (Unsigned) 6-37
6.6.26 DJNZ-Decrement and Jump if Non-Zero 6-38
6.6.27 EI-Enable Interrupts 6-39
6.6.28 ENTER-Enter 6-40
6.6.29 EXIT-Exit 6-41
6.6.30 IDLE-Idle Operation 6-42
6.6.31 INC-Increment 6-43
6.6.32 INCW-Increment Word 6-44
6.6.33 IRET-Interrupt Return 6-45
6.6.34 JP-Jump 6-46
6.6.35 JR-Jump Relative 6-47
6.6.36 LD-Load 6-48
6.6.37 LD-Load 6-49
6.6.38 LDB-Load Bit 6-50
6.6.39 LDC/LDE-Load Memory 6-51
6.6.40 LDC/LDE-Load Memory 6-52
6.6.41 LDCD/LDED-Load Memory and Decrement 6-53
6.6.42 LDCI/LDEI-Load Memory and Increment 6-54
6.6.43 LDCPD/LDEPD-Load Memory with Pre-Decrement 6-55
6.6.44 LDCPI/LDEPI-Load Memory with Pre-Increment 6-56
6.6.45 LDW-Load Word 6-57
6.6.46 MULT-Multiply (Unsigned) 6-58
6.6.47 NEXT-Next 6-59
6.6.48 NOP-No Operation 6-60
6.6.49 OR-Logical OR 6-61
6.6.50 POP-Pop from Stack 6-62
6.6.51 POPUD-Pop User Stack (Decrementing) 6-63
6.6.52 POPUI-Pop User Stack (Incrementing) 6-64
6.6.53 PUSH-Push to Stack 6-65

6.6.54 PUSHUD-Push User Stack (Decrementing) ... 6-66
6.6.55 PUSHUI-Push User Stack (Incrementing) ... 6-67
6.6.56 RCF-Reset Carry Flag ... 6-68
6.6.57 RET-Return ... 6-69
6.6.58 RL-Rotate Left ... 6-70
6.6.59 RLC-Rotate Left Through Carry ... 6-71
6.6.60 RR-Rotate Right ... 6-72
6.6.61 RRC-Rotate Right through Carry ... 6-73
6.6.62 SB0-Select Bank 0 ... 6-74
6.6.63 SB1-Select Bank 1 ... 6-75
6.6.64 SBC-Subtract With Carry ... 6-76
6.6.65 SCF-Set Carry Flag ... 6-77
6.6.66 SRA-Shift Right Arithmetic ... 6-78
6.6.67 SRP/SRP0/SRP1-Set Register Pointer ... 6-79
6.6.68 STOP-Stop Operation ... 6-80
6.6.69 SUB-Subtract ... 6-81
6.6.70 SWAP-Swap Nibbles ... 6-82
6.6.71 TCM-Test Complement under Mask ... 6-83
6.6.72 TM-Test under Mask ... 6-84
6.6.73 WFI-Wait for Interrupt ... 6-85
6.6.74 XOR-Logical Exclusive OR ... 6-86

7 Clock Circuit ... 7-1

7.1 Overview ... 7-1
7.2 C2.Clock Status during Power-Down Modes ... 7-2
7.3 System Clock Control Register ... 7-2
7.4 Internal Oscillator Calibration Control Register ... 7-3
7.5 Internal Oscillator Calibration Data Register ... 7-4
7.6 Ring Oscillator Control Register ... 7-5
7.7 STOP Control Register ... 7-6

8 RESET and Power-Down ... 8-1

8.1 Overview ... 8-1
8.1.1 MCU Initialization Sequence ... 8-3
8.2 Power-Down Modes ... 8-4
8.2.1 Stop Mode ... 8-4
8.2.2 Idle Mode ... 8-5
8.3 Hardware Reset Values ... 8-6

9 I/O Ports ... 9-1

9.1 Overview ... 9-1
9.2 Port Data Registers ... 9-2
9.3 Port 0 ... 9-3
9.3.1 Port 0 Control Register ... 9-3
9.4 Port 1 ... 9-5
9.4.1 Port 1 Control Register Low Byte (P1CONL) ... 9-5
9.5 Port 2 ... 9-6
9.5.1 Port 2 Control Register (P2CONH, P2CONL) ... 9-6
9.5.2 Port 2 Pull-Up Resistor Enable Register (P2PUR) ... 9-6
9.6 Port 3 ... 9-9
9.6.1 Port 3 Control Register (P3CONH, P3CONL) ... 9-9
9.6.2 Port 3 Interrupt Enable and Pending Registers ... 9-9
9.6.3 Port 3 Pull-Up Resistor Enable Register ... 9-9
9.6.4 Port 3 N-Channel Open-drain Mode Register ... 9-9

10 Basic Timer ... 10-1

10.1 Overview 10-1
10.1.1 Basic Timer (BT) 10-1
10.2 Basic Timer Control Register (BTCON) 10-2
10.3 Basic Timer Function Description 10-3
10.3.1 Watchdog Timer Function 10-3
10.3.2 Oscillation Stabilization Interval Timer Function 10-3

11 8-Bit Timer A 11-1

11.1 Overview 11-1
11.2 Timer A Control Register (TACON) 11-2
11.3 Function Description 11-4
11.3.1 Timer A Interrupts 11-4
11.3.2 Interval Timer Function 11-4
11.3.3 Pulse Width Modulation Mode 11-4
11.3.4 Capture Mode 11-5
11.4 Block Diagram 11-6

12 16-Bit Timer 0/1/2 12-1

12.1 Overview 12-1
12.2 Timer 0 Control Register 12-2
12.3 Block Diagram 12-4
12.4 Overview 16-Bit Timer 1 12-5
12.5 Timer 1 Control Register 12-6
12.6 Timer 1 Function Description 12-8
12.6.1 Timer 1 Interrupts (IRQ3, Vectors D8H and DAH) 12-8
12.6.2 Interval Timer Mode 12-8
12.6.3 Pulse Width Modulation Mode 12-8
12.6.4 Capture Mode 12-9
12.7 Block Diagram 12-10
12.8 Overview 16-Bit Timer 2 12-11
12.9 Timer 2 Control Register 12-12
12.10 Timer 2 Function Description 12-14
12.10.1 Timer 2 Interrupts (IRQ3, Vectors DCH and DEH) 12-14
12.10.2 Interval Timer Mode 12-14
12.10.3 Pulse Width Modulation Mode 12-14
12.10.4 Capture Mode 12-15
12.11 Block Diagram 12-16

13 16-Bit Timer B 13-1

13.1 Overview 13-1
13.2 Block Diagram 13-4
13.2.1 Timer B Pulse Width Calculations 13-5

14 Watch Timer 14-1

14.1 Overview 14-1
14.1.1 Watch Timer Control Register 14-2
14.1.2 Watch Timer Circuit Diagram 14-3

15 Stop Wake-Up Timer 15-1

15.1 Overview 15-1
15.1.1 Function Description 15-1
15.1.2 Stop Wake-Up Timer Control Register 15-2
15.1.3 Stop Mode Wake Up 15-4

16 A/D Converter 16-1

16.1 Overview 16-1
16.1.1 Using A/D Pins for Standard Digital Input 16-2
16.1.2 A/D Converter Control Register 16-2
16.1.3 Internal Reference Voltage Levels 16-3
16.1.4 Conversion Timing 16-5
16.1.5 Internal A/D Conversion Procedure 16-5

17 Serial Peripheral Interface 17-1

17.1 Overview 17-1
17.2 Operation as an SPI Master 17-3
17.3 Master SCK Selection 17-3
17.4 Operation as an SPI Slave 17-4
17.5 SPI Status and Control 17-4
17.6 SPI Interrupt 17-5
17.7 SPI System Errors 17-6
17.8 SPI Control Register 17-7
17.9 SPI Status Register 17-8
17.10 SPI Data Register 17-9

18 Inter Integrated Circuit-Bus Interface 18-1

18.1 Overview 18-1
18.1.1 Multi-Master IIC-Bus Control Register 18-2
18.1.2 Multi-Master IIC-Bus Control/Status Register 18-4
18.1.3 Multi-Master IIC-Bus Transmit/Receive Data Shift Register 18-5
18.1.4 Multi-Master IIC-Bus Address Register 18-6
18.2 Block Diagram 18-7
18.3 The IIC-Bus Interface 18-8
18.4 Start and Stop Conditions 18-8
18.5 Data Transfer Formats 18-10
18.6 ACK Signal Transmission 18-11
18.7 Read-Write Operations 18-14
18.8 Bus Arbitration Procedures 18-14
18.9 Abort Conditions 18-14
18.10 Configuring the IIC-Bus 18-14

19 UART 0 19-1

19.1 Overview 19-1
19.2 Programming Procedure 19-2
19.2.1 UART 0 High Byte Control Register (UART0CONH) 19-2
19.2.2 UART 0 Low Byte Control Register 19-2
19.2.3 UART 0 Interrupt Pending Bits 19-5
19.2.4 UART 0 Data Register 19-5
19.2.5 UART 0 Baud Rate Data Register 19-5
19.3 Baud Rate Calculations 19-6
19.3.1 Mode 0 Baud Rate Calculation 19-6
19.3.2 Mode 2 Baud Rate Calculation 19-6
19.3.3 Modes 1 and 3 Baud Rate Calculation 19-6
19.4 Block Diagram 19-7
19.5 UART 0 Mode 0 Function Description 19-8
19.5.1 Mode 0 Transmit Procedure 19-8
19.5.2 Mode 0 Receive Procedure 19-8
19.6 Serial Port Mode 1 Function Description 19-10
19.6.1 Mode 1 Transmit Procedure 19-10
19.6.2 Mode 1 Receive Procedure 19-10
19.7 Serial Port Mode 2 Function Description 19-12
19.7.1 Mode 2 Transmit Procedure 19-12

19.7.2 Mode 2 Receive Procedure 19-12
19.8 Serial Port Mode 3 Function Description 19-14
19.8.1 Mode 3 Transmit Procedure 19-14
19.8.2 Mode 3 Receive Procedure 19-14
19.9 Serial Communication for Multiprocessor Configurations 19-16
19.9.1 Sample Protocol for Master/Slave Interaction 19-16
19.9.2 Setup Procedure for Multiprocessor Communications 19-17

20 UART 1 20-1

20.1 Overview 20-1
20.2 Programming Procedure 20-2
20.2.1 UART 1 High Byte Control Register (UART1CONH) 20-2
20.2.2 UART 1 Low Byte Control Register 20-2
20.2.3 UART 1 Interrupt Pending Bits 20-5
20.2.4 UART 1 Data Register 20-5
20.2.5 UART 1 Baud Rate Data Register 20-5
20.3 Baud Rate Calculations 20-6
20.3.1 Mode 0 Baud Rate Calculation 20-6
20.3.2 Mode 2 Baud Rate Calculation 20-6
20.3.3 Modes 1 and 3 Baud Rate Calculation 20-6
20.4 Block Diagram 20-7
20.5 UART 1 Mode 0 Function Description 20-8
20.5.1 Mode 0 Transmit Procedure 20-8
20.5.2 Mode 0 Receive Procedure 20-8
20.6 Serial Port Mode 1 Function Description 20-10
20.6.1 Mode 1 Transmit Procedure 20-10
20.6.2 Mode 1 Receive Procedure 20-10
20.7 Serial Port Mode 2 Function Description 20-12
20.7.1 Mode 2 Transmit Procedure 20-12
20.7.2 Mode 2 Receive Procedure 20-12
20.8 Serial Port Mode 3 Function Description 20-14
20.8.1 Mode 3 Transmit Procedure 20-14
20.8.2 Mode 3 Receive Procedure 20-14
20.9 Serial Communication for Multiprocessor Configurations 20-16
20.9.1 Sample Protocol for Master/Slave Interaction 20-16
20.9.2 Setup Procedure for Multiprocessor Communications 20-17

21 Embedded Flash Memory Interface 21-1

21.1 Overview 21-1
21.2 User Program Mode 21-1
21.2.1 Flash Memory Control Register 21-2
21.2.2 Flash Memory User Programming Enable Register 21-3
21.2.3 Flash Memory Sector Address Registers 21-4
21.3 ISP™ (On-Board Programming) Sector 21-5
21.4 ISP Reset Vector and ISP Sector Size 21-6
21.5 Sector Erase 21-7
21.6 Programming 21-9
21.7 Reading 21-10
21.8 Hard Lock Protection 21-11

22 Low Voltage Detect and Low Voltage Reset 22-1

22.1 Low Voltage Reset 22-1
22.2 Low Voltage Detect 22-3

23 Electrical Data 23-1

23.1 Overview 23-1

23.2 Absolute Maximum Ratings 23-2
23.3 DC Electrical Characteristics 23-3
23.4 AC Electrical Characteristics 23-5
23.5 Oscillator Characteristics 23-6

24 Mechanical Data 24-1

24.1 Overview 24-1

25 S3F8S39/S3F8S35 Flash MCU 25-1

25.1 Overview 25-1
25.2 Pin Assignments 25-2
25.3 On Board Writing 25-4
25.3.1 Circuit Design Guide 25-4
25.3.2 Reference Table for Connection 25-5

26 Development Tools 26-1

26.1 Overview 26-1
26.2 Emulator-based Development System 26-1
26.2.1 Host Software 26-2
26.2.2 Target Boards 26-2
26.2.3 Optional Probe Adapter Cable(s) and Application Board 26-11
26.3 Zilog Library-based Development Platform 26-12
26.3.1 Zilog Developer Platform Components 26-12
26.3.2 Compatibility with 3rd Party Tools 26-14
26.3.3 Benefits and Limitations of Zilog Development Tools 26-14
26.4 Development Tools 26-15

27 Ordering Information 27-1

27.1 Part Number Suffix Designations 27-1

List of Figures

Figure Number	Title	Page Number
Figure 1-1	S3F8S39/S3F8S35 Block Diagram	1-5
Figure 1-2	S3F8S39/S3F8S35 Pin Assignment (32-SOP/32-SDIP)	1-6
Figure 1-3	S3F8S39/S3F8S35 Pin Assignment (32-QFN/32-LQFP)	1-6
Figure 1-4	Pin Circuit Type A	1-9
Figure 1-5	Pin Circuit Type B	1-9
Figure 1-6	Pin Circuit Type C	1-10
Figure 1-7	Pin Circuit Type D-5 (P3.4-P3.5)	1-10
Figure 1-8	Pin Circuit Type D-6 (P2.0-P2.3, P3.6-P3.7)	1-11
Figure 1-9	Pin Circuit Type E (P3.0-P3.3)	1-11
Figure 1-10	Pin Circuit Type E-1 (P0, P1)	1-12
Figure 1-11	Pin Circuit Type E-2 (P2.4-P2.7)	1-12
Figure 2-1	Program Memory Address Space	2-2
Figure 2-2	Smart Option	2-3
Figure 2-3	Internal Register File Organization (S3F8S39/S3F8S35)	2-6
Figure 2-4	Register PP	2-7
Figure 2-5	Set 1, Set 2, Prime Area Register Map	2-9
Figure 2-6	8 Byte Working Register Areas (Slices)	2-10
Figure 2-7	Contiguous 16 Byte Working Register Block	2-11
Figure 2-8	Non-Contiguous 16 Byte Working Register Block	2-12
Figure 2-9	16-Bit Register Pair	2-13
Figure 2-10	Register File Addressing	2-14
Figure 2-11	Common Working Register Area	2-15
Figure 2-12	4-Bit Working Register Addressing	2-17
Figure 2-13	4-Bit Working Register Addressing Example	2-18
Figure 2-14	8-Bit Working Register Addressing	2-19
Figure 2-15	8-Bit Working Register Addressing Example	2-20
Figure 2-16	Stack Operations	2-21
Figure 3-1	Register Addressing	3-2
Figure 3-2	Working Register Addressing	3-2
Figure 3-3	Indirect Register Addressing to Register File	3-3
Figure 3-4	Indirect Register Addressing to Program Memory	3-4
Figure 3-5	Indirect Working Register Addressing to Register File	3-5
Figure 3-6	Indirect Working Register Addressing to Program or Data Memory	3-6
Figure 3-7	Indexed Addressing to Register File	3-7
Figure 3-8	Indexed Addressing to Program or Data Memory with Short Offset	3-8
Figure 3-9	Indexed Addressing to Program or Data Memory	3-9
Figure 3-10	Direct Addressing for Load Instructions	3-10
Figure 3-11	Direct Addressing for Call and Jump Instructions	3-11
Figure 3-12	Indirect Addressing	3-12
Figure 3-13	Relative Addressing	3-13
Figure 3-14	Immediate Addressing	3-14
Figure 4-1	Register Description Format	4-7
Figure 5-1	S3C8/S3F8 Series Interrupt Types	5-2
Figure 5-2	S3F8S39/S3F8S35 Interrupt Structure	5-3
Figure 5-3	ROM Vector Address Area	5-4
Figure 5-4	Interrupt Function Diagram	5-6
Figure 5-5	System Mode Register (SYM)	5-9
Figure 5-6	Interrupt Mask Register (IMR)	5-10
Figure 5-7	Interrupt Request Priority Groups	5-11
Figure 5-8	Interrupt Priority Register (IPR)	5-12

Figure 5-9 Interrupt Request Register (IRQ) 5-13
Figure 6-1 System Flags Register (FLAGS) 6-5
Figure 7-1 Main Oscillator Circuit (RC Oscillator with Internal Capacitor) 7-1
Figure 7-2 Main Oscillator Circuit (Crystal/Ceramic Oscillator) 7-1
Figure 7-3 System Clock Control Register (CLKCON) 7-2
Figure 7-4 Internal Oscillator Calibration Control Register (OSCCALCON) 7-3
Figure 7-5 Internal Oscillator Calibration Data Register (OSCCALDATA) 7-4
Figure 7-6 Ring Oscillator Control Register (ROSCCON) 7-5
Figure 7-7 STOP Control Register (STPCON) 7-6
Figure 7-8 System Clock Circuit Diagram 7-7
Figure 8-1 Low Voltage Reset Circuit 8-2
Figure 8-2 Reset Block Diagram 8-3
Figure 8-3 Timing for S3F8S39/S3F8S35 after reset 8-3
Figure 9-1 Port 0 Control Register High Byte (P0CONH) 9-3
Figure 9-2 Port 0 Control Register Low Byte (P0CONL) 9-4
Figure 9-3 Pull-up Resistor Enable Register (P0PUR) 9-4
Figure 9-4 Port 1 Control Register Low Byte 9-5
Figure 9-5 Port 2 High-Byte Control Register (P2CONH) 9-6
Figure 9-6 Port 2 Low-Byte Control Register (P2CONL) 9-7
Figure 9-7 Port 2 Interrupt Control Register (P2INT) 9-7
Figure 9-8 Port 2 Pull-Up Resistor Enable Register (P2PUR) 9-8
Figure 9-9 Port 3 High-Byte Control Register (P3CONH) 9-10
Figure 9-10 Port 3 Low-Byte Control Register (P3CONL) 9-10
Figure 9-11 Port 3 Interrupt Control Register (P3INT) 9-11
Figure 9-12 Port 3 & Port 0 Interrupt Pending Register (PINTPND) 9-11
Figure 9-13 Port 3 Pull-Up Resistor Enable Register (P3PUR) 9-12
Figure 9-14 Port 3 N-Channel Open-Drain Mode Register (PNE3) 9-12
Figure 10-1 Basic Timer Control Register (BTCON) 10-2
Figure 10-2 Basic Timer Block Diagram 10-4
Figure 11-1 Timer A Control Register (TACON) 11-2
Figure 11-2 Timer A Prescaler Register (TAPS) 11-3
Figure 11-3 Timer A Functional Block Diagram 11-6
Figure 12-1 Timer 0 Control Register (T0CON) 12-2
Figure 12-2 Timer 0 Prescaler Register (T0PS) 12-3
Figure 12-3 Timer 0 Functional Block Diagram 12-4
Figure 12-4 Timer 1 Control Register (T1CON) 12-6
Figure 12-5 Timer 1 Prescaler Register (T1PS) 12-7
Figure 12-6 Timer 1 Functional Block Diagram 12-10
Figure 12-7 Timer 2 Control Register (T2CON) 12-12
Figure 12-8 Timer 2 Prescaler Register (T2PS) 12-13
Figure 12-9 Timer 2 Functional Block Diagram 12-16
Figure 13-1 Timer B Control Register 13-1
Figure 13-2 Timer B Trigger Register (TBTRG) 13-2
Figure 13-3 Timer B Data Registers (TBDATA0H/L) 13-2
Figure 13-4 Timer B Data Registers (TBDATA1H/L) 13-3
Figure 13-5 Timer B Functional Block Diagram 13-4
Figure 13-6 Timer B Output Flip-Flop Waveforms in Repeat Mode 13-6
Figure 14-1 Watch Timer Control Register (WTCON) 14-2
Figure 14-2 Watch Timer Circuit Diagram 14-3
Figure 15-1 Stop Wake-Up Timer Control Register (SWTCON) 15-2
Figure 15-2 Stop Wake-Up Timer Functional Block Diagram 15-3
Figure 15-3 Stop Wake-Up Timer Operation Sequence 15-4
Figure 16-1 A/D Converter Control Register (ADCON) 16-2
Figure 16-2 A/D Converter Circuit Diagram 16-3
Figure 16-3 A/D Converter Data Register (ADDATAH/L) 16-3
Figure 16-4 A/D Converter Timing Diagram 16-4
Figure 16-5 Recommended A/D Converter Circuit for Highest Absolute Accuracy 16-5

Figure 17-1 SPI Block Diagram 17-2
Figure 17-2 SPI Data Timing 17-2
Figure 17-3 SPI Control Register (SPICON) 17-7
Figure 17-4 SPI Status Register (SPISTAT) 17-8
Figure 17-5 SPI Data Register (SPIDATA) 17-9
Figure 18-1 Multi-Master IIC-Bus Control Register (ICCR) 18-2
Figure 18-2 Multi-Master IIC-Bus Control/Status Register (ICSR) 18-4
Figure 18-3 Multi-Master IIC-Bus Tx/Rx Data Shift Register (IDSR) 18-5
Figure 18-4 Multi-Master IIC-Bus Address Register (IAR) 18-6
Figure 18-5 IIC-Bus Block Diagram 18-7
Figure 18-6 Start and Stop Conditions 18-8
Figure 18-7 Input Data Protocol 18-9
Figure 18-8 Interrupt Pending Information 18-9
Figure 18-9 IIC-Bus Interface Data Formats 18-10
Figure 18-10 Acknowledge Response from Receiver 18-11
Figure 18-11 Write Operation Sequence 18-12
Figure 18-12 Read Operation Sequence 18-13
Figure 19-1 UART 0 High Byte Control Register (UART0CONH) 19-3
Figure 19-2 UART 0 Low Byte Control Register (UART0CONL) 19-4
Figure 19-3 UART 0 Data Register (UDATA0) 19-5
Figure 19-4 UART 0 Baud Rate Data Register (BRDATA0) 19-5
Figure 19-5 UART 0 Functional Block Diagram 19-7
Figure 19-6 Timing Diagram for Serial Port Mode 0 Operation 19-9
Figure 19-7 Timing Diagram for Serial Port Mode 1 Operation 19-11
Figure 19-8 Timing Diagram for Serial Port Mode 2 Operation 19-13
Figure 19-9 Timing Diagram for Serial Port Mode 3 Operation 19-15
Figure 19-10 Connection Example for Multiprocessor Serial Data Communications 19-17
Figure 20-1 UART 1 High Byte Control Register (UART1CONH) 20-3
Figure 20-2 UART 1 Low Byte Control Register (UART1CONL) 20-4
Figure 20-3 UART 1 Data Register (UDATA1) 20-5
Figure 20-4 UART 1 Baud Rate Data Register (BRDATA1) 20-5
Figure 20-5 UART 1 Functional Block Diagram 20-7
Figure 20-6 Timing Diagram for Serial Port Mode 0 Operation 20-9
Figure 20-7 Timing Diagram for Serial Port Mode 1 Operation 20-11
Figure 20-8 Timing Diagram for Serial Port Mode 2 Operation 20-13
Figure 20-9 Timing Diagram for Serial Port Mode 3 Operation 20-15
Figure 20-10 Connection Example for Multiprocessor Serial Data Communications 20-17
Figure 21-1 Flash Memory Control Register (FMCON) 21-2
Figure 21-2 Flash Memory User Programming Enable Register (FMUSR) 21-3
Figure 21-3 Flash Memory Sector Address Register High Byte (FMSECH) 21-4
Figure 21-4 Flash Memory Sector Address Register Low Byte (FMSECL) 21-4
Figure 21-5 Program Memory Address Space 21-5
Figure 21-6 Sector Configurations in User Program Mode 21-7
Figure 22-1 Low Voltage Reset Circuit 22-2
Figure 22-2 Low Voltage Detect (LVD) Block Diagram 22-3
Figure 22-3 Low Voltage Detect Control Register (LVDCON) 22-4
Figure 23-1 Input Timing Measurement Points 23-5
Figure 23-2 Operating Voltage Range 23-8
Figure 23-3 Schmitt Trigger Input Characteristics Diagram 23-9
Figure 23-4 Stop Mode Release Timing When Initiated by a RESET 23-10
Figure 23-5 Waveform for UART Timing Characteristics 23-12
Figure 23-6 LVR Reset Timing 23-12
Figure 23-7 The Circuit Diagram to Improve EFT Characteristics 23-14
Figure 24-1 32-SOP-450A Package Dimensions 24-1
Figure 24-2 32-SDIP-400 Package Dimensions 24-2
Figure 24-3 32-Pin QFN Package Dimension 24-3
Figure 25-1 S3F8S39/S3F8S35 Pin Assignments (32-SOP/32-SDIP) 25-2

Figure 25-2 S3F8S39/S3F8S35 Pin Assignments (32-QFN/32-LQFP) 25-2
Figure 25-3 RC Delay Circuit 25-3
Figure 25-4 PCB Design Guide for On-Board Programming 25-4
Figure 26-1 Emulator-based Development System Configuration 26-1
Figure 26-2 TB8S19/8S28/8S39 Target Board Configuration 26-3
Figure 26-3 DIP Switch for Smart Option 26-9
Figure 26-4 50-Pin Connector for TB8S19/8S28/8S39 26-10
Figure 26-5 TB8S19/8S28/8S39 Probe Adapter Cable and Application Board 26-11

List of Tables

Table Number	Title	Page Number
Table 1-1	Pin Descriptions of 32-SOP (32-SDIP)	1-7
Table 1-2	Pin Descriptions of 32-QFN (32-LQFP)	1-8
Table 2-1	S3F8S39/S3F8S35 Register Type Summary	2-5
Table 4-1	System and Peripheral Control Registers Set 1	4-2
Table 4-2	System and Peripheral Control Registers Set1 Bank 0	4-3
Table 4-3	System and Peripheral Control Registers Set1 Bank 1	4-4
Table 4-4	System and Peripheral Control Registers Page 8	4-6
Table 5-1	Interrupt Control Register Overview	5-5
Table 5-2	Interrupt Source Control and Data Registers	5-7
Table 6-1	Instruction Group Summary	6-2
Table 6-2	Flag Notation Conventions	6-7
Table 6-3	Instruction Set Symbols	6-7
Table 6-4	Instruction Notation Conventions	6-8
Table 6-5	Opcode Quick Reference (0–7)	6-9
Table 6-6	Opcode Quick Reference (8–F)	6-10
Table 6-7	Condition Codes	6-11
Table 8-1	S3F8S39/S3F8S35 Set 1 Registers Values after Reset	8-6
Table 8-2	System and Peripheral Control Registers (Set 1, Bank 0)	8-7
Table 8-3	System and Peripheral Control Registers (Set 1, Bank 1)	8-8
Table 8-4	System and Peripheral Control Registers Page 8	8-9
Table 9-1	S3F8S39/S3F8S35 Port Configuration Overview	9-1
Table 9-2	Port Data Register Summary	9-2
Table 15-1	Stop Wake-Up Timer Prescaler Select	15-3
Table 17-1	SPI Pin Assignment	17-5
Table 17-2	SCK Rate Selection	17-5
Table 18-1	Sample Timing Calculations for the IIC-Bus Transmit Clock (SCL)	18-3
Table 19-1	Commonly Used Baud Rates Generated by BRDATA0	19-6
Table 20-1	Commonly Used Baud Rates Generated by BRDATA1	20-6
Table 21-1	ISP Sector Size	21-6
Table 21-2	Reset Vector Address	21-6
Table 23-1	Absolute Maximum Ratings	23-2
Table 23-2	DC Electrical Characteristics	23-3
Table 23-3	AC Electrical Characteristics	23-5
Table 23-4	Oscillator Characteristics (S3F8S35XZZ, S3F8S39XZZ)	23-6
Table 23-5	Oscillator Characteristics (S3F8S35EZZ, S3F8S39EZZ)	23-7
Table 23-6	Oscillation Stabilization Time	23-7
Table 23-7	Data Retention Supply Voltage in Stop Mode	23-10
Table 23-8	A/D Converter Electrical Characteristics	23-11
Table 23-9	UART Timing Characteristics in Mode 0 (12 MHz)	23-12
Table 23-10	Flash Memory AC Electrical Characteristics	23-13
Table 23-11	ESD Characteristics	23-15
Table 23-12	LVD Circuit Characteristics	23-15
Table 23-13	LVR Circuit Characteristics	23-15
Table 25-1	Descriptions of Pins Used to Read/Write the Flash in Tool Program Mode	25-3
Table 25-2	Reference Table for Connection	25-5
Table 26-1	Components of TB8S19/8S28/8S39	26-4
Table 26-2	Device Selection Settings for TB8S19/8S28/8S39	26-5
Table 26-3	Power Selection Settings for TB8S19/8S28/8S39	26-6
Table 26-4	SMDS2+ Tool Selection Setting	26-6
Table 26-5	Using Single Header Pins to Select Clock Source/PWM/Operation Mode	26-7

Table 26-6 Using Single Header Pins as the Input Path for External Trigger Sources 26-8

List of Examples

Example Number	Title	Page Number
Example 2-1	Setting the Register Pointers	2-11
Example 2-2	Using the RPs to Calculate the Sum of a Series of Registers	2-12
Example 2-3	Addressing the Common Working Register Area	2-16
Example 2-4	Standard Stack Operations Using PUSH and POP	2-22
Example 13-1	To Generate 38 kHz, 1/3 Duty Signal through P2.1	13-7
Example 13-2	To Generate A One Pulse Signal through P2.1	13-8
Example 15-1	Using Stop Wake-Up Timer	15-5
Example 16-1	Configuring A/D Converter	16-6
Example 21-1	Sector Erase	21-8
Example 21-2	Program	21-9
Example 21-3	Reading	21-10
Example 21-4	Hard Lock Protection	21-11

1 Product Overview

1.1 S3F8 Series Microcontrollers

Zilog's S3 Family of 8-bit single-chip CMOS microcontrollers offers a fast and efficient CPU, a wide range of integrated peripherals, and various mask-programmable Read only memory (ROM) sizes. Address/data bus architecture and a large number of bit-configurable Input/Output (I/O) ports provide a flexible programming environment for applications with varied memory and I/O requirements. It includes timer/counters with selectable operating modes to support real-time operations.

1.2 S3F8S39/S3F8S35 Microcontrollers

The S3F8S39/S3F8S35 single-chip CMOS microcontrollers are fabricated using the highly advanced CMOS process technology based on Zilog's latest CPU architecture. The S3F8S39/S3F8S35 is a microcontroller with an embedded 32K/16K byte full-flash ROM.

By using a proven modular design approach, Zilog engineers have successfully developed the S3F8S39/S3F8S35 by integrating these peripheral modules with the powerful SAM8 RC core:

- Four configurable I/O ports (26 pin)
- Twenty six interrupt sources and eight interrupt levels
- Eight bit-programmable pins for external interrupts
- One watchdog timer function (Basic Timer)
- One 8-bit basic timer for oscillation stabilization
- Two 8-bit timer/counters and three 16-bit timer/counters with selectable operating modes
- A Stop Wake-Up Timer to wake up CPU from Stop Mode
- Watch timer for real time
- Two asynchronous UART modules
- One SPI module
- One IIC module
- Analog to digital converter with 16 input channels and 10-bit resolution
- One BUZ for programmable frequency output

The S3F8S39/S3F8S35 microcontrollers are ideal for use in a wide range of home applications that requires simple timer/counter, ADC, and so on. They are currently available in 32-pin SOP/SDIP and 32-pin QFN packages.

Comparison	S3F8S39	S3F8S35
Flash Size (Bytes)	32K	16K

1.3 Features

1.3.1 CPU

- SAM8RC CPU core

1.3.2 Memory

Features of internal multi-time program Full-Flash memory are:

- 32K × 8-bit program memory (S3F8S39)
- 16K × 8-bit program memory (S3F8S35)
  - Sector size: 128 Bytes
  - User programmable by "LDC" instruction
  - Sector erase available
  - Fast programming time
  - External serial programming support
  - Endurance: 10,000 erase/program cycles
  - 10 Years data retention

- Data Memory (RAM)
  - 1,040 × 8 bit data memory

1.3.3 Instruction Set

- Seventy eight instructions
- Idle and Stop instructions

1.3.4 26 I/O Pins

- Twenty six normal I/O pins for 32 pin

1.3.5 Interrupts

- Eight interrupt levels and 26 interrupt sources
- Fast interrupt processing feature

1.3.6 Timers and Timer/Counters

- One programmable 8-bit Basic Timer (BT) for oscillation stabilization control or watchdog timer function.
- One 8-bit timer/counter (Timer A) with three operating modes:
  - Interval Mode
  - Capture Mode
  - PWM Mode

- One 16-bit timer/counter (Timer 0) with Interval Mode and PWM Mode.
- Two 16-bit timer/counter (Timer 1/2) with three operating modes:
  - Interval Mode
  - Capture Mode
  - PWM Mode
- One 16-bit Timer B with one-shot-pulse output mode and repeat output mode. It can be triggered by external input or software.
- One Stop Wake-Up Timer with Ring oscillator as its clock source.

1.3.7 Watch Timer

- Interval time: 1.995 ms, 0.125s, 0.25s, and 0.5s at 32.768 kHz
- 0.5/1/2/4 kHz buzzer output selectable

1.3.8 Analog to Digital Converter

- 16-channel analog input
- 10-bit conversion resolution

1.3.9 Two Channels UART

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

1.3.10 Multi-Master IIC-Bus

- Serial Peripheral Interface
- Serial, 8-bit Data Transfers
- Programmable Clock Prescale

1.3.11 SPI

- Support Master and Slave Mode
- Programmable Clock Prescale

1.3.12 Two Power-Down Modes

- **Idle Mode**: Stops only CPU clock
- **Stop Mode**: Stops system clock and CPU clock

1.3.13 Oscillation Sources

- Main clock frequency: 0.4 MHz-12.0 MHz
- External RC for main clock
- Internal RC: 8 MHz (typ.), 4 MHz (typ.), 1 MHz (typ.), 0.5 MHz (typ.)
- On-chip free running Ring oscillator with 32 kHz frequency for 16-bit Timer 1.

1.3.14 Instruction Execution Time

- 333 ns at fx = 12 MHz (minimum, main clock)

1.3.15 Built-In Reset Circuit (LVR)

- Low-Voltage check to reset system
- V_{LVR} = 1.9/2.3/3.0/3.9 V (by the Smart Option)

1.3.16 Low Voltage Detect Circuit (LVD)

- Flag or Interrupt for voltage drop detection
- Programmable detect voltage: 2.1/2.5/3.2/4.1 V
- Enable/Disable software selectable

1.3.17 Operating Voltage Range

- 1.8 V to 5.5 V at 4 MHz (main clock)
- 2.7 V to 5.5 V at 12 MHz (main clock)

1.3.18 Package Type

- 32-pin SDIP
- 32-pin SOP
- 32-pin QFN
  32-pin LQFP

1.3.19 Operating Temperature Range

- –40°C to +85°C

1.3.20 Smart Option

- ISP-related option selectable (ROM address 3EH)
- Oscillator selection, LVR selection (ROM address 3FH)

1.4 Block Diagram

Figure 1-1 illustrates the block diagram of S3F8S39/S3F8S35.

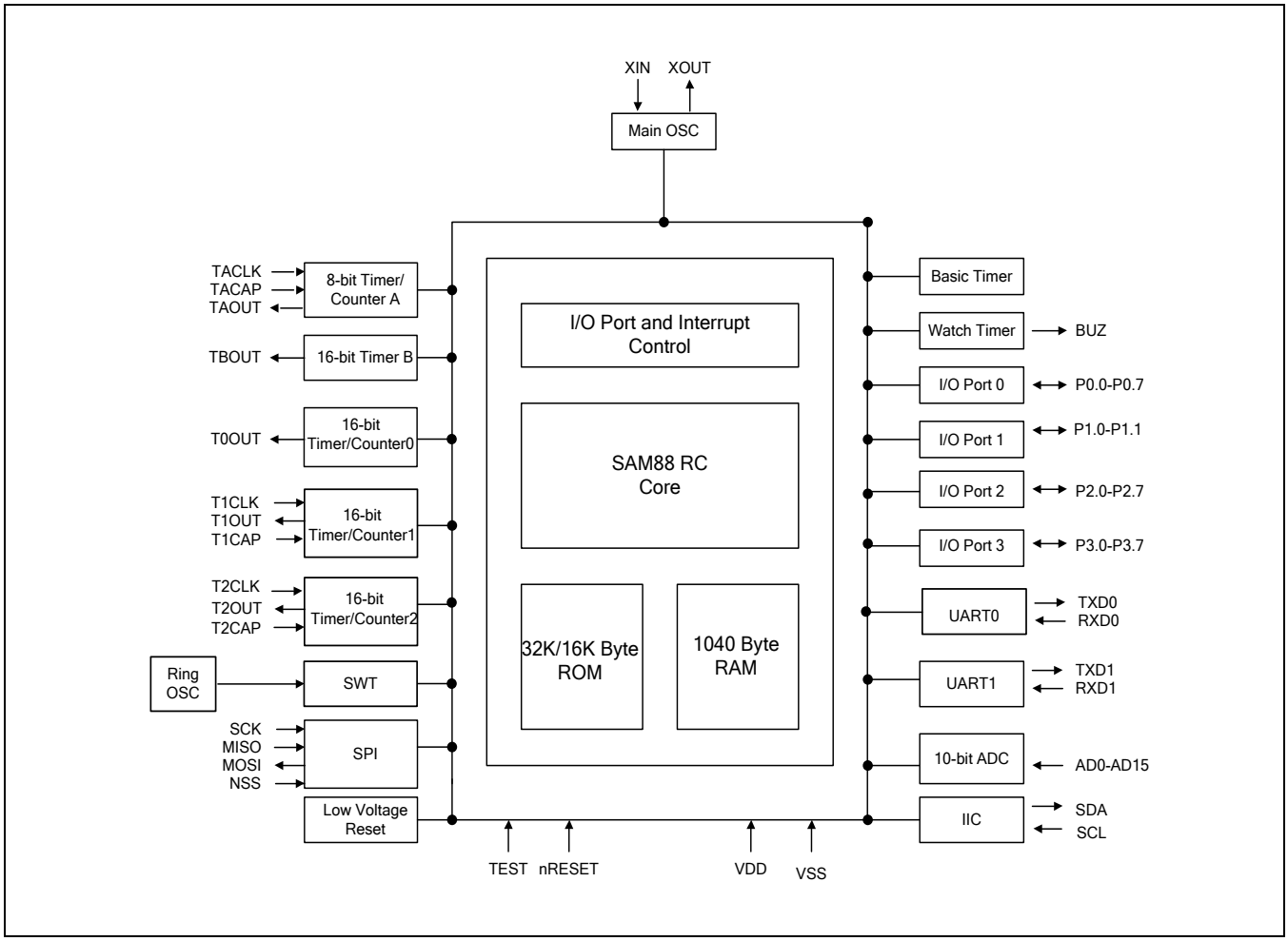


Figure 1-1 S3F8S39/S3F8S35 Block Diagram

1.5 Pin Assignments

Figure 1-2 illustrates the S3F8S39/S3F8S35 pin assignment (32-pin SOP/32-pin SDIP).

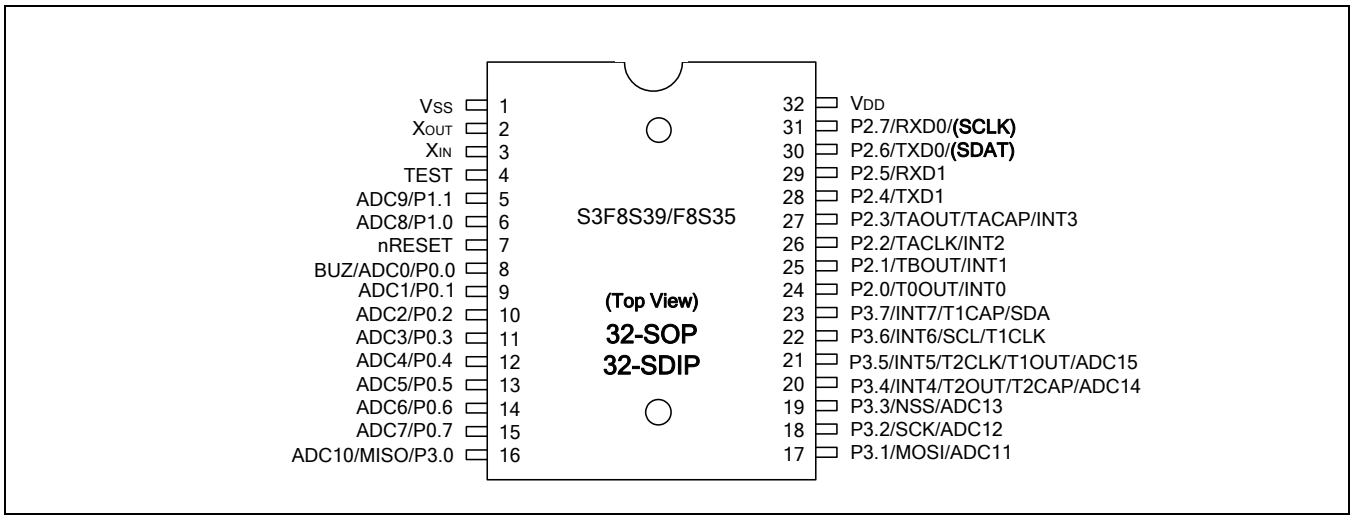


Figure 1-2 S3F8S39/S3F8S35 Pin Assignment (32-SOP/32-SDIP)

Figure 1-3 illustrates the S3F8S39/S3F8S35 pin assignment (32-pin QFN/32-pin LQFP).

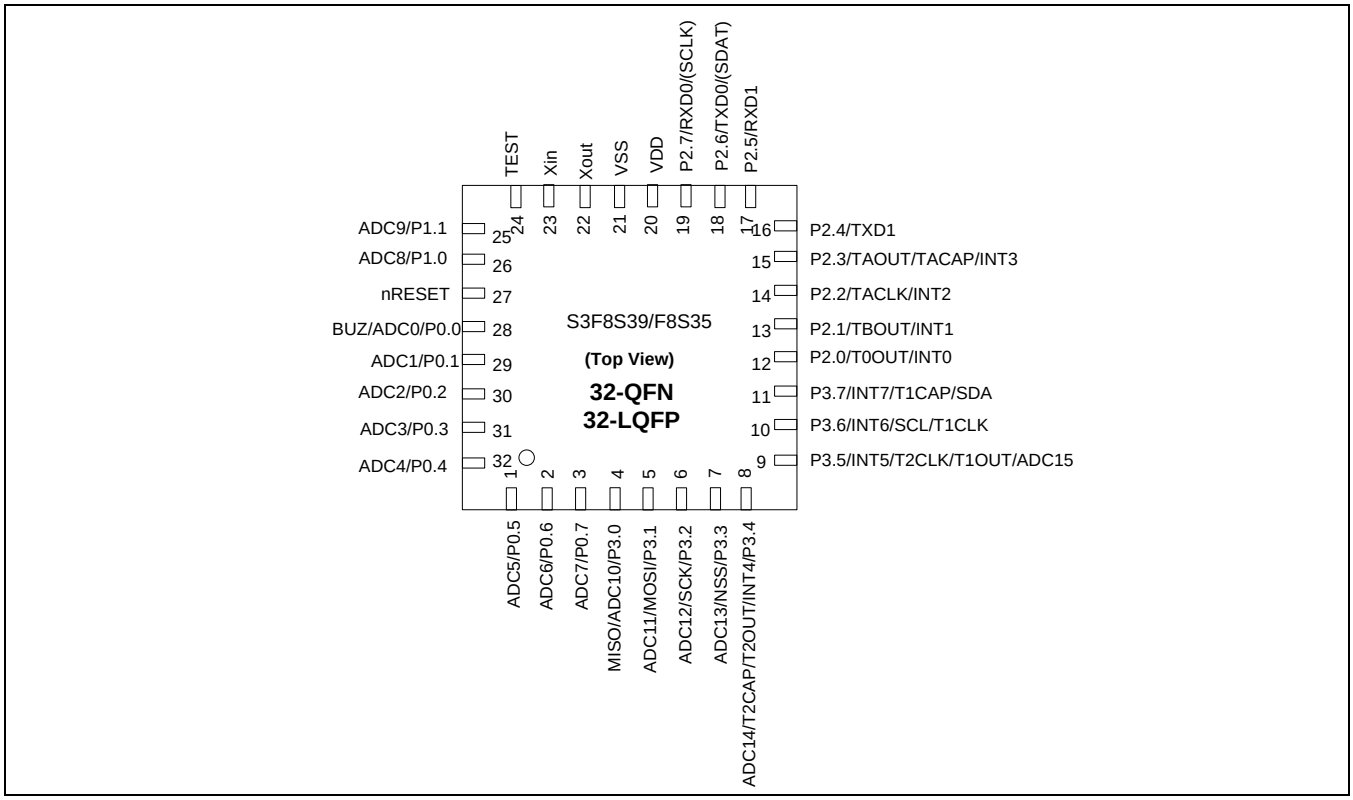


Figure 1-3 S3F8S39/S3F8S35 Pin Assignment (32-QFN/32-LQFP)

1.6 Pin Descriptions

Table 1-1 describes the pin descriptions of 32-pin SOP (32-pin SDIP).

Table 1-1 Pin Descriptions of 32-SOP (32-SDIP)

Pin Names	Pin Type	Pin Description	Circuit Type	Pin No.	Shared Functions
P0.0 P0.1-P0.3	I/O	I/O port with bit-programmable pins Input or push-pull output and software assignable pull-ups.	E-1	8 (28) 9-11 (29-31)	ADC0/BUZ ADC1-ADC3
P0.4-P0.7				12-15 (32,1-3)	ADC4 -ADC7
P1.0	I/O	I/O port with bit-programmable pins Input or push-pull output and software assignable pull-ups.	E-1	6 (26)	ADC8
P1.1				5 (25)	ADC9
P2.0	I/O	Input or push-pull, open-drain output and software assignable pull-up. I/O port with bit-programmable pins	D-6 E-2	24 (12)	T0OUT/INT0
P2.1				25 (13)	TBOUT/INT1
P2.2				26 (14)	TACLK/INT2
P2.3		–		27 (15)	TAOUT/TACAP/INT3
P2.4		–		28 (16)	TXD1
P2.5		–		29 (17)	RXD1
P2.6		–		30 (18)	TXD0
P2.7		–		31 (19)	RXD0
P3.0	I/O	I/O port with bit-programmable pins Schmitt trigger input or push-pull, open-drain output and software assignable pull-ups.	E D-5 D-6	16 (4)	ADC10/MISO
P3.1				17 (5)	ADC11/MOSI
P3.2				18 (6)	ADC12/ SCK
P3.3				19 (7)	ADC13/NSS
P3.4				20 (8)	INT4/T2OUT/ T2CAP/ADC14
P3.5				21 (9)	INT5/T2CLK/ T1OUT/ADC15
P3.6				22 (10)	INT6/SCL/T1CLK
P3.7				23 (11)	INT7/T1CAP/ SDA

NOTE: Parentheses indicate pin number for 32-QFN package.

Table 1-2 describes the pin descriptions of 32-pin QFN (32-pin LQFP).

Table 1-2 Pin Descriptions of 32-QFN (32-LQFP)

Pin Names	Pin Type	Pin Description	Circuit Type	Pin No.	Shared Functions
AD0-AD7	I/O	A/D converter analog input channels	E-1 E	8-15 (28-32, 1-3)	P0.0/BUZ-P0.7
AD8-AD9				6-5 (25-26)	P1.0-P1.1
AD10-AD15				16-21 (4-9)	P3.0/MISO, P3.1/MOSI, P3.2/SCK, P3.3/NSS, P3.4/INT4/T2OUT/T2CAP, P3.5/INT5/T2CLK/T1OUT
TXD0	I/O	UART data output and input	E-2	30 (18)	P2.6
RXD0				31 (19)	P2.7
TXD1				28 (16)	P2.4
RXD1				29 (17)	P2.5
SDA SCL	I/O	IIC data output and input	D-6	23 (11) 22 (10)	P3.7/INT7/T1CAP P3.6/INT6/T1CLK
TAOUT	I/O	Timer A clock output and PWM output	D-6	27 (15)	P2.3/TACAP
TACAP	I/O	Timer A capture input	D-6	27 (15)	P2.3/TAOUT
TACLK	I/O	Timer A external clock input	D-6	26 (14)	P2.2
TBOUT	I/O	Timer B carrier frequency output	D-6	25 (13)	P2.1
T0OUT	I/O	Timer 0 clock output	D-6	24 (12)	P2.0
T1OUT	I/O	Timer 1 clock output and PWM output	D-5	21 (9)	P3.5/INT5/T2CLK/ADC15
T1CAP	I/O	Timer 1capture input.	D-6	23 (11)	P3.7/INT7/SDA
T2OUT	I/O	Timer 2 clock output and PWM output	D-5	20 (8)	P3.4/INT4/ T2CAP/ADC14
T2CAP	I/O	Timer 2 capture input	D-5	20 (8)	P3.4/INT4/T2OUT/ADC14
T2CLK	I/O	Timer 2 external clock input	D-5	21 (9)	P3.5/INT5/T1OUT/ADC15
BUZ	I/O	Output pin for buzzer signal	E-1	8 (28)	P0.0/ADC0
SCK	I/O	SPI interface clock	E	18 (6)	P3.2/ADC12
MISO	I/O	SPI Master input slave output	E	16 (4)	P3.0/ADC10
MOSI	I/O	SPI Master output slave input	E	17 (5)	P3.1/ADC11
NSS	O	Slave selection	E	19 (7)	P3.3/ADC13
INT0-INT3	I	External interrupt input pins	D-6	24-27 (12-15)	P2.0/T0OUT, P2.1/TBOUT, P2.2/TACLK, P2.3/TAOUT/TACAP
INT4-INT7			D-5 D-6	20-23 (8-11)	P3.4-P3.7
nRESET	I	System reset pin	B	7 (27)	–

Pin Names	Pin Type	Pin Description	Circuit Type	Pin No.	Shared Functions
X_{IN}, X_{OUT}	–	Main oscillator pins	–	3 (23), 2 (22)	–
TEST	I	Test input: it should be connected to V_{SS}	–	4 (24)	–
V_{DD}	–	Power supply input pin	–	32 (20)	–
V_{SS}	–	Ground pins	–	1 (21)	–

NOTE: Parentheses indicate pin number for 32-QFN package.

1.7 Pin Circuit Diagrams

Figure 1-4 illustrates the pin circuit type A.

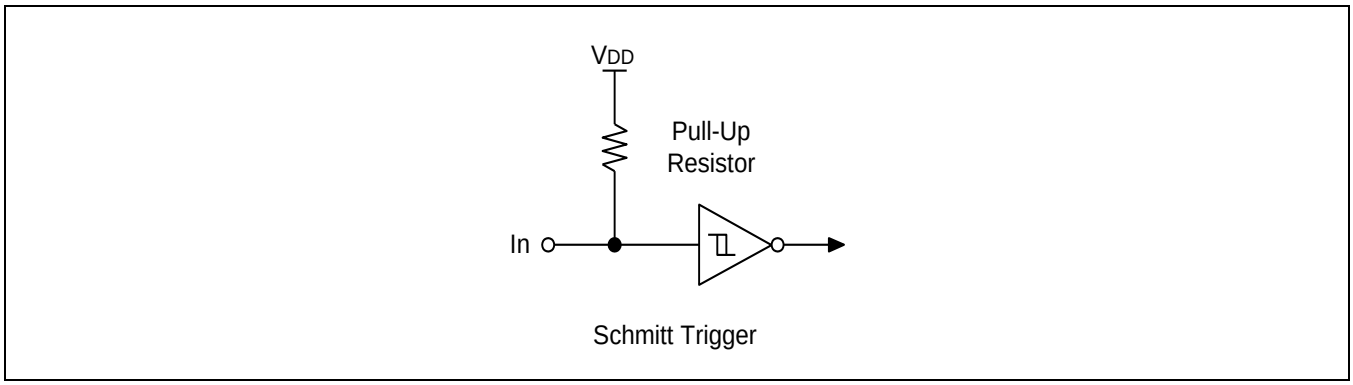


Figure 1-4 Pin Circuit Type A

Figure 1-5 illustrates the pin circuit type B.

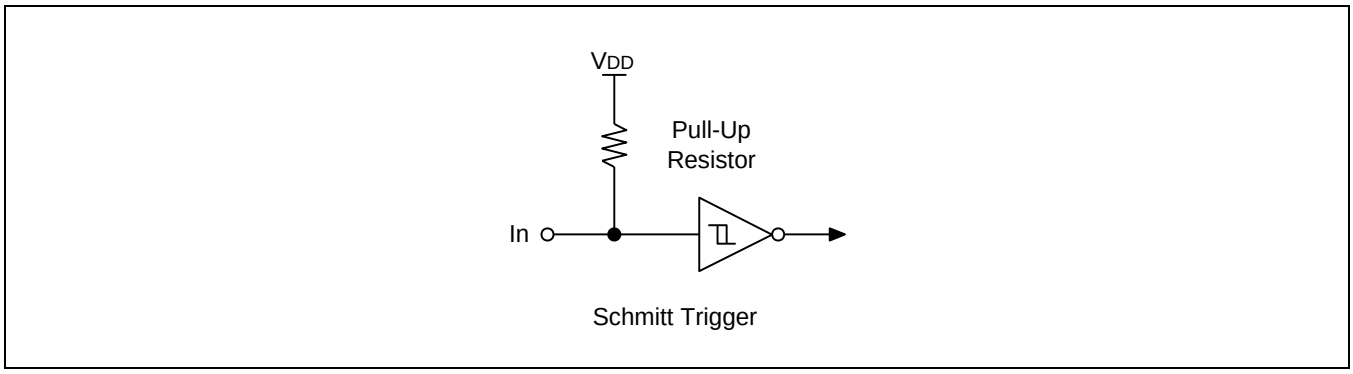


Figure 1-5 Pin Circuit Type B

Figure 1-6 illustrates the pin circuit type C.

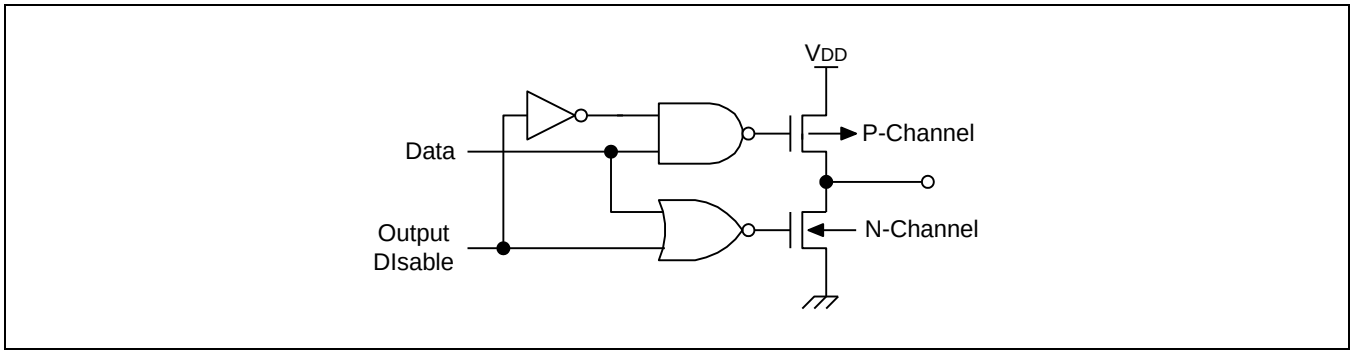


Figure 1-6 Pin Circuit Type C

Figure 1-7 illustrates the pin circuit type D (P3.4-P3.5).

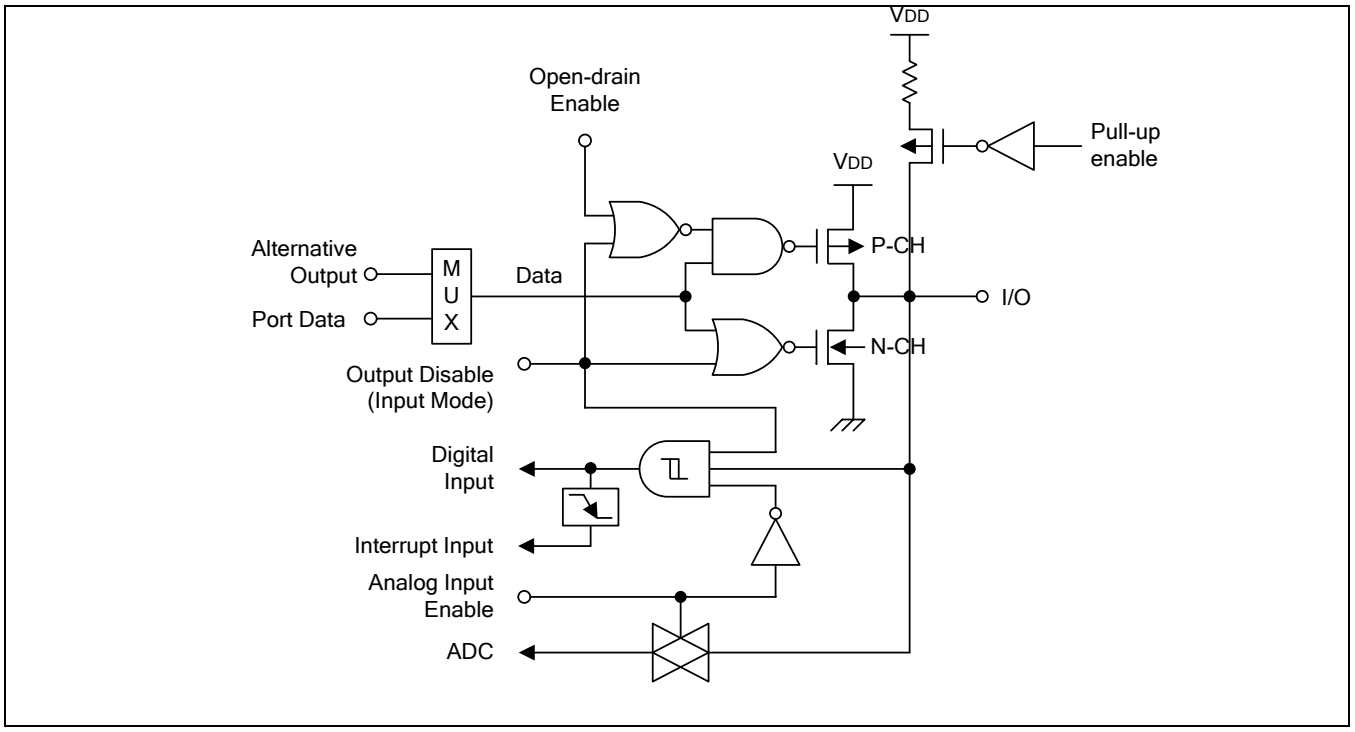


Figure 1-7 Pin Circuit Type D-5 (P3.4-P3.5)

Figure 1-8 illustrates the pin circuit type D-6 (P2.0-P2.3, P3.6-P3.7).

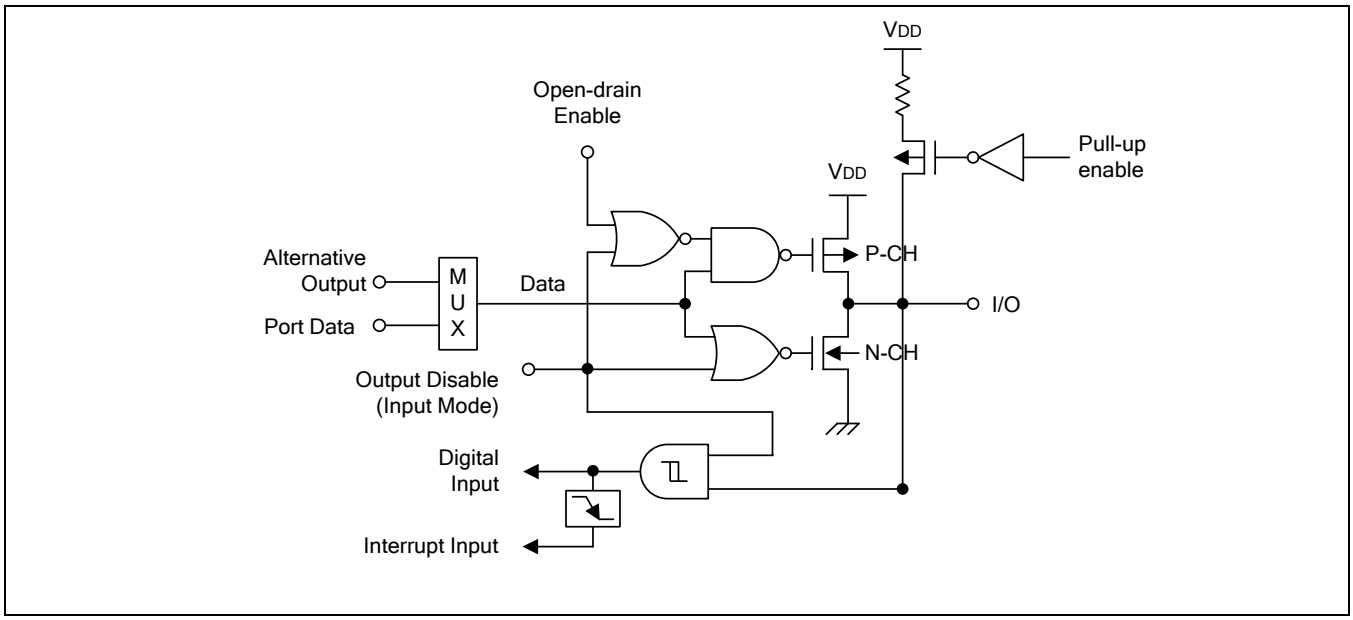


Figure 1-8 Pin Circuit Type D-6 (P2.0-P2.3, P3.6-P3.7)

Figure 1-9 illustrates the pin circuit type E (P3.0-P3.3).

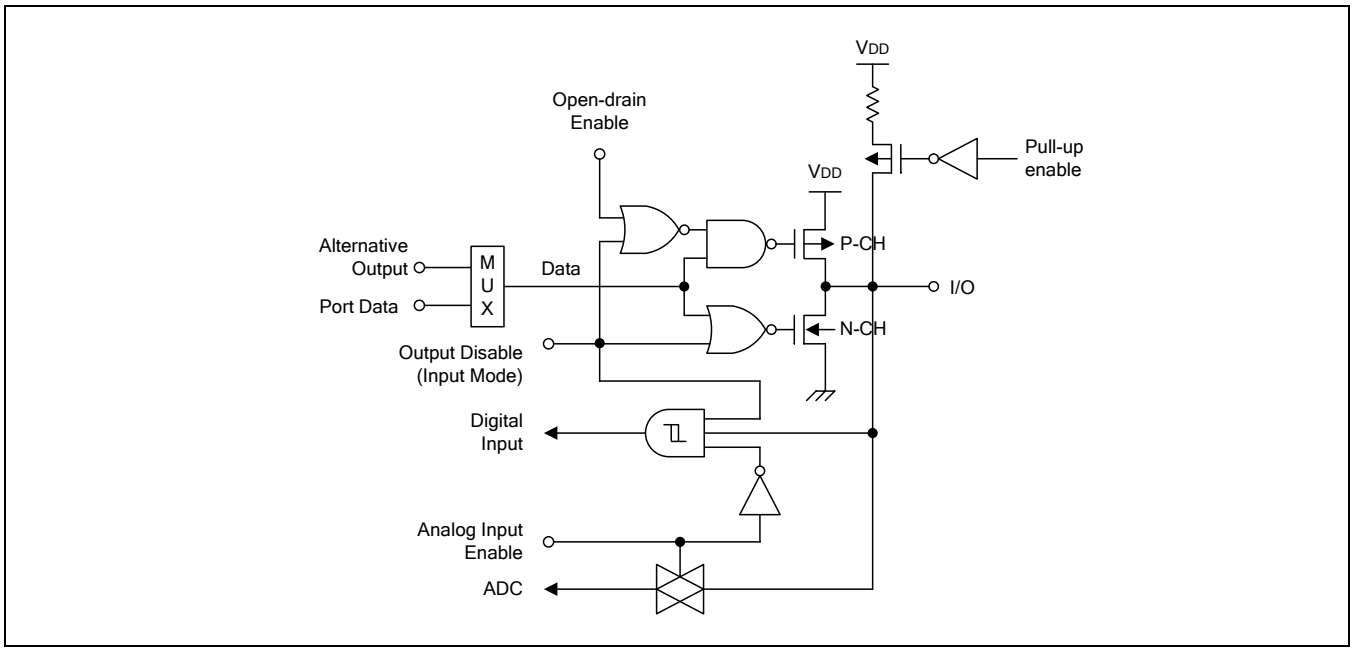


Figure 1-9 Pin Circuit Type E (P3.0-P3.3)

Figure 1-10 illustrates the pin circuit type E-1 (P0, P1).

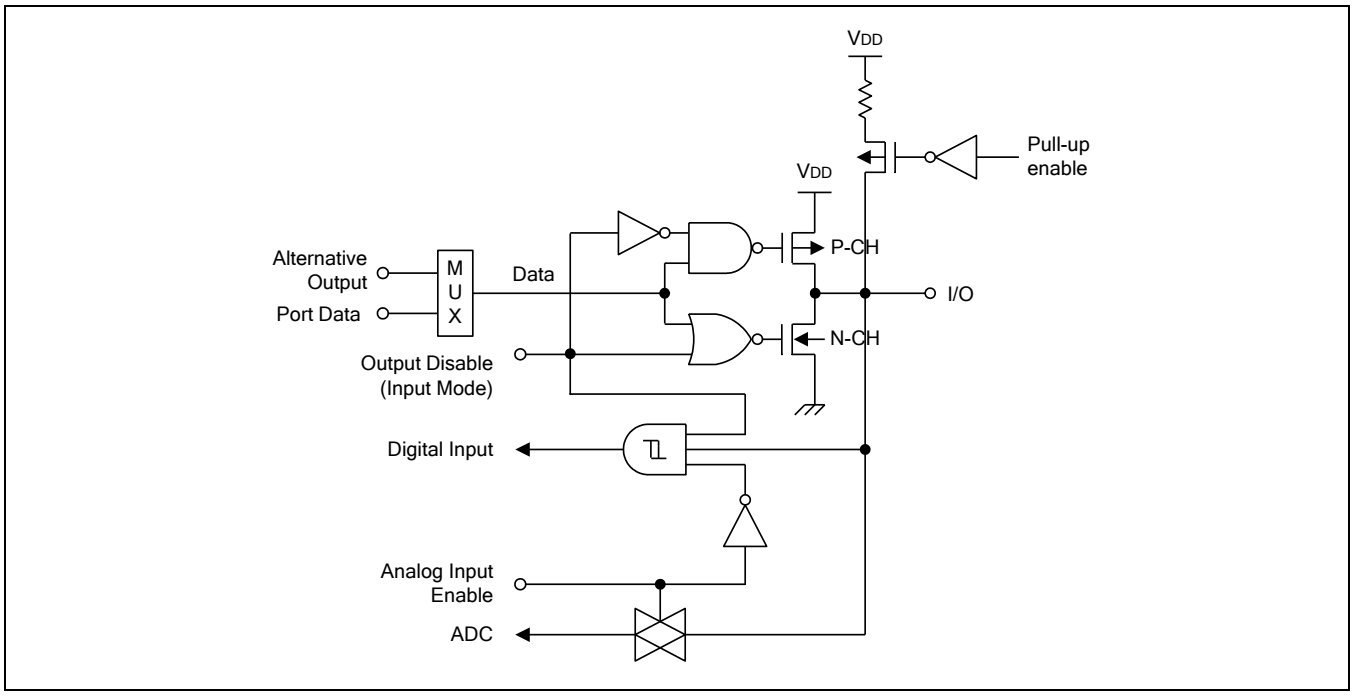


Figure 1-10 Pin Circuit Type E-1 (P0, P1)

Figure 1-11 illustrates the pin circuit type E-2 (P2.4-P2.7).

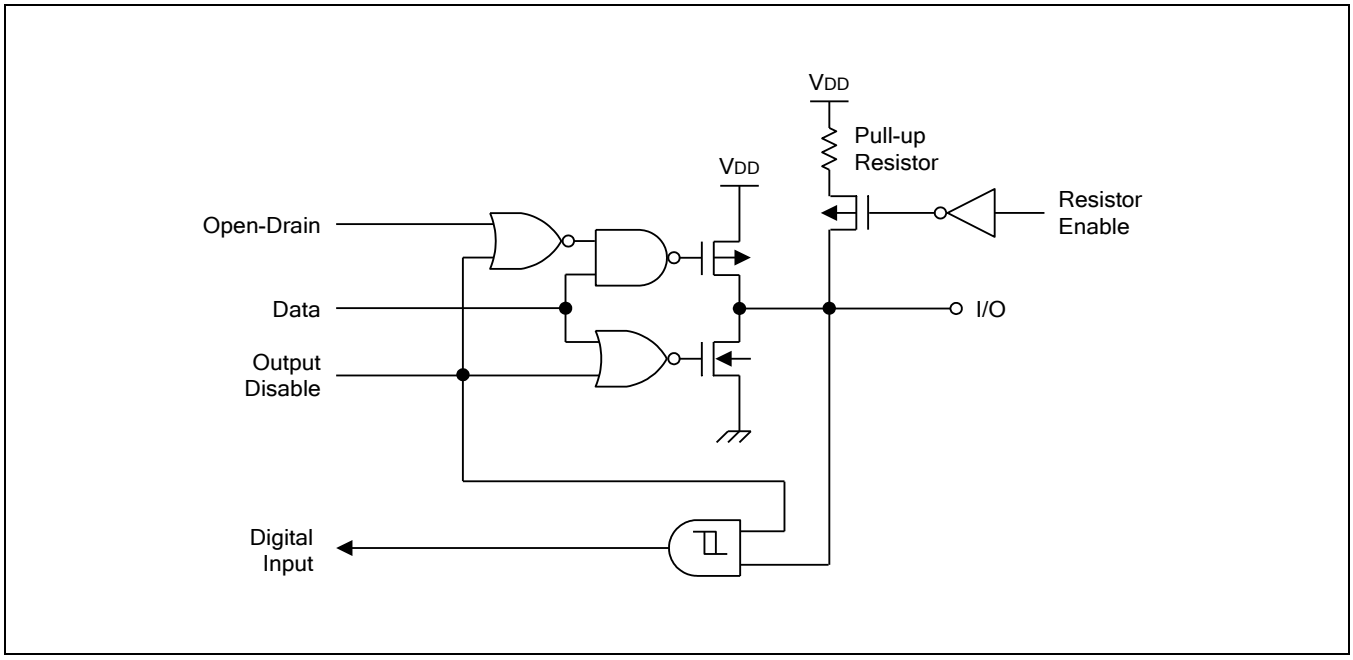


Figure 1-11 Pin Circuit Type E-2 (P2.4-P2.7)

2 Address Spaces

2.1 Overview

The S3F8S39/S3F8S35 microcontroller has two types of address spaces:

- Internal program memory (Read only 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 internal register file.

The S3F8S39/S3F8S35 microcontroller has 32K/16K bytes of on-chip program memory that it configures as the Internal ROM mode.

The S3F8S39/S3F8S35 microcontroller has 1126 general-purpose registers in its internal register file. It maps 86 bytes in the register file for system and peripheral control functions.

2.2 Program Memory (ROM)

Program memory (ROM) stores program codes or table data. The S3F8S39/S3F8S35 microcontroller has 32K/16K bytes of internal multi time programmable (MTP) program memory (Refer to *Figure 2-1* for more information).

It reserves the first 256 bytes of the ROM (0H–0FFH) for interrupt vector addresses. You can use unused locations (except 3CH, 3DH, 3EH, 3FH) in this address range as normal program memory. If you use the vector address area to store a program code, ensure not to overwrite the vector addresses stored in these locations.

It uses 3CH, 3DH, 3EH, and 3FH as Smart Option ROM cell.

The default program Reset address in the ROM is 0100H.

Figure 2-1 illustrates the program memory address space.

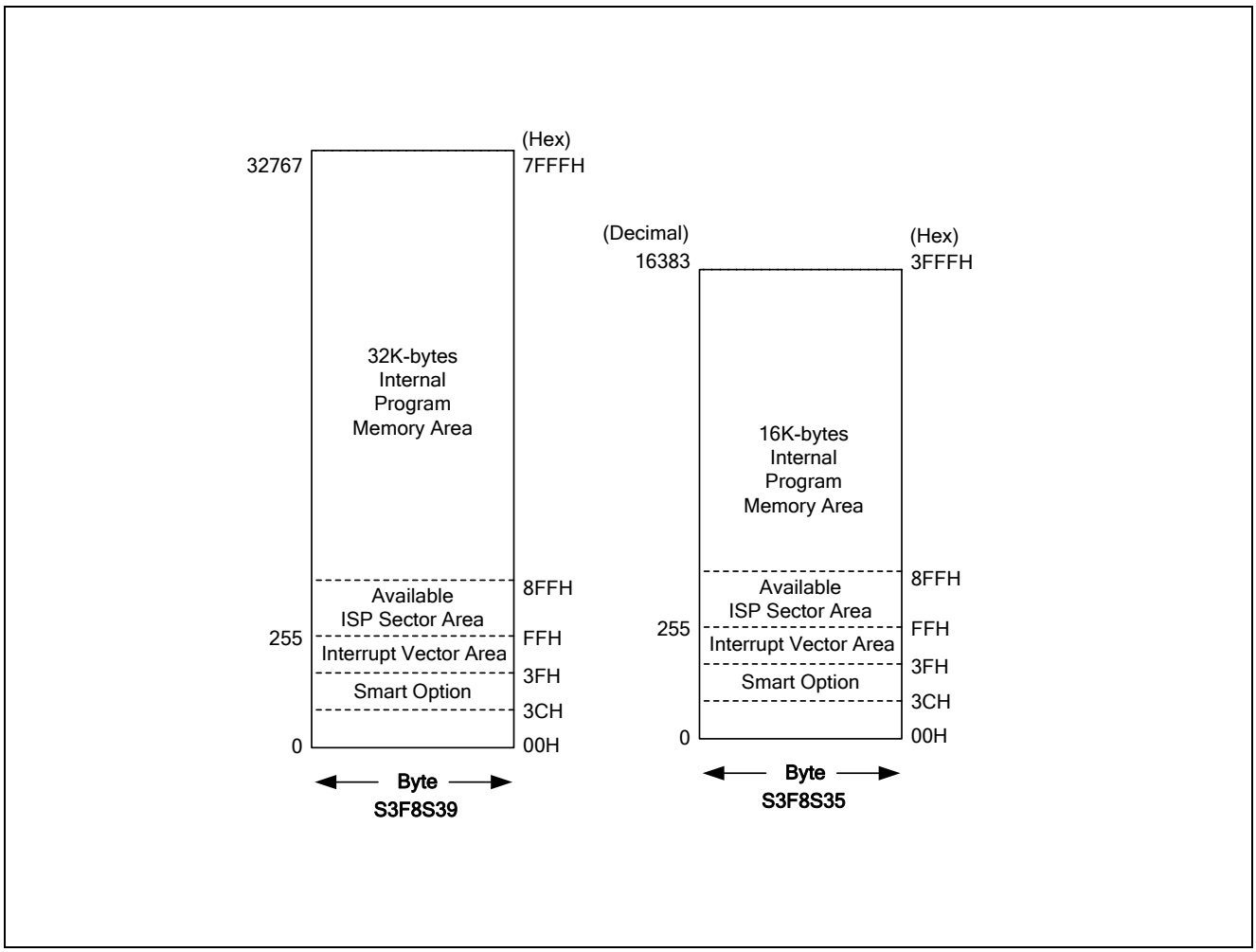
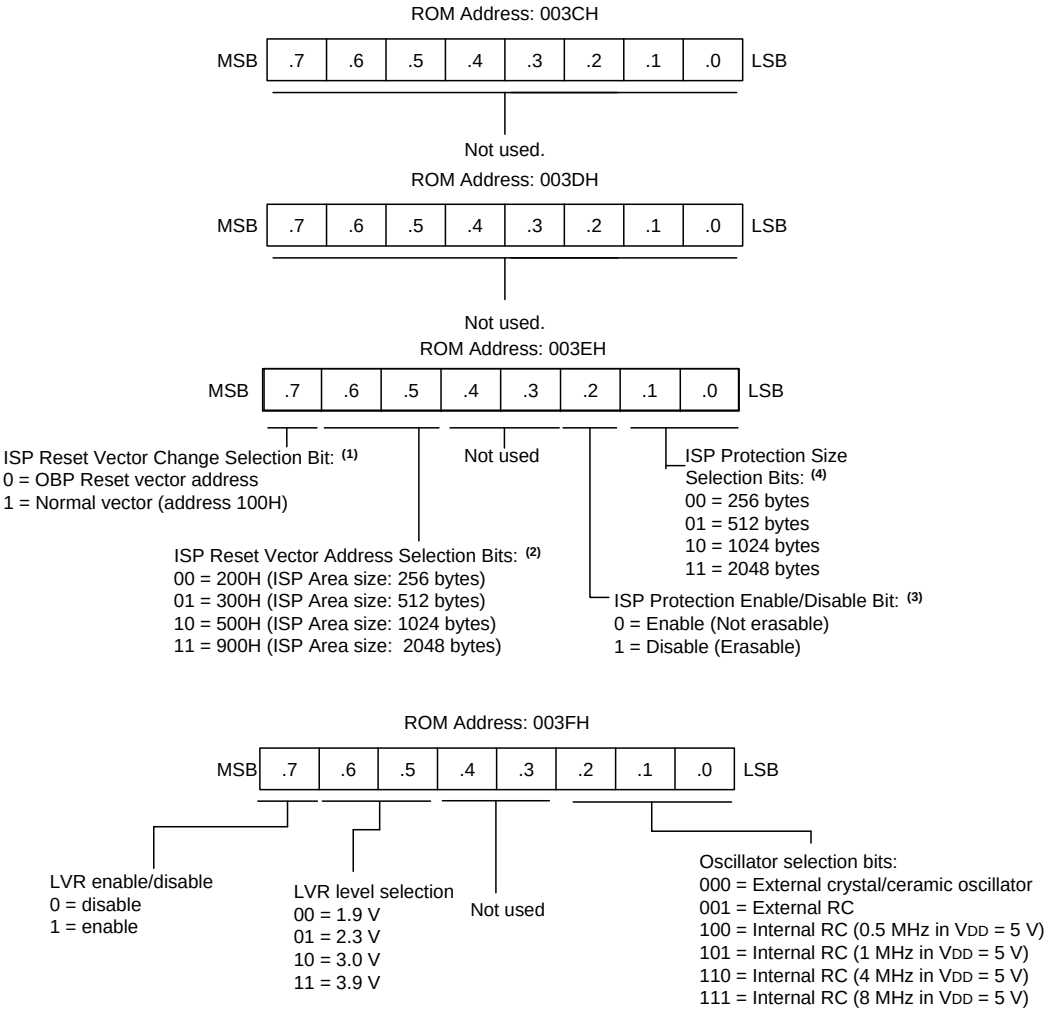


Figure 2-1 Program Memory Address Space

2.3 Smart Option

Figure 2-2 illustrates the 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. The unused bits of 3EH, 3FH must be set to '1'.

Figure 2-2 Smart Option

Smart Option is the ROM option for start condition of the chip. Smart Option that the ROM address uses is from 003EH to 003FH. It does not use 003CH and 003DH in S3F8S39/S3F8S35.

When LDC instruction writes any values in the Smart Option area (003CH–003FH), it may change the data of the area. However, it does not affect the Smart Option. OTP/MTP programmer (Writer tools) should write the data for Smart Option in the Smart Option area (003CH–003FH).

2.4 Register Architecture

In the S3F8S39/S3F8S35 microcontroller implementation, it expands the upper 64 byte area of register files into two 64 byte areas. This expansion is called as set 1 and set 2. It further expands the upper 32 byte area of set 1 into 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 S3F8S39/S3F8S35 microcontroller the total number of addressable 8-bit registers is 1128. Of these 1128 registers, 13 bytes are for CPU and system control registers, 70 bytes are for peripheral control and data registers, it uses 16 bytes as a shared working registers, and 1040 registers are for general-purpose use.

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

Various addressing mode restrictions, the select bank instructions, SB0 and SB1, and the register page pointer (PP) support the extension of register space into separately addressable areas (sets, banks and pages).

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

Table 2-1 describes the specific register types and the area (in bytes) that they occupy in the register file.

Table 2-1 S3F8S39/S3F8S35 Register Type Summary

Register Type	Number of Bytes
System and peripheral registers	88
General-purpose registers (including the 16 byte common working register area, four 192 byte prime register area and four 64 byte set 2 area)	1040
Total Addressable Bytes	1128

Figure 2-3 illustrates the internal register file organization (S3F8S39/S3F8S35).

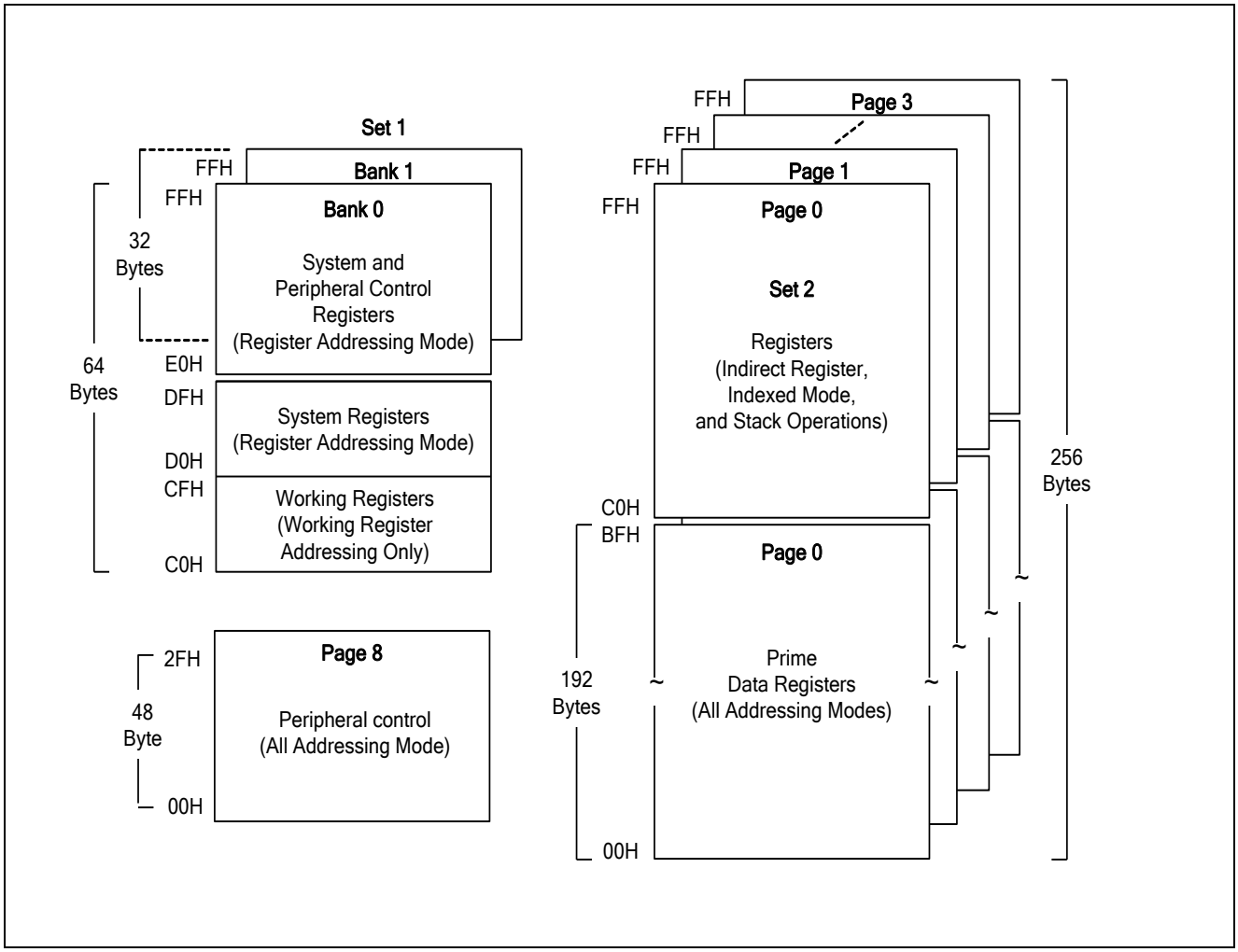


Figure 2-3 Internal Register File Organization (S3F8S39/S3F8S35)

2.5 Register Page Pointer

By using an 8-bit data bus, the S3F8-series architecture supports the logical expansion of the physical 256 byte internal register file into 6 separately addressable register pages. The register page pointer (PP) controls page addressing.

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

Figure 2-4 illustrates the register PP.

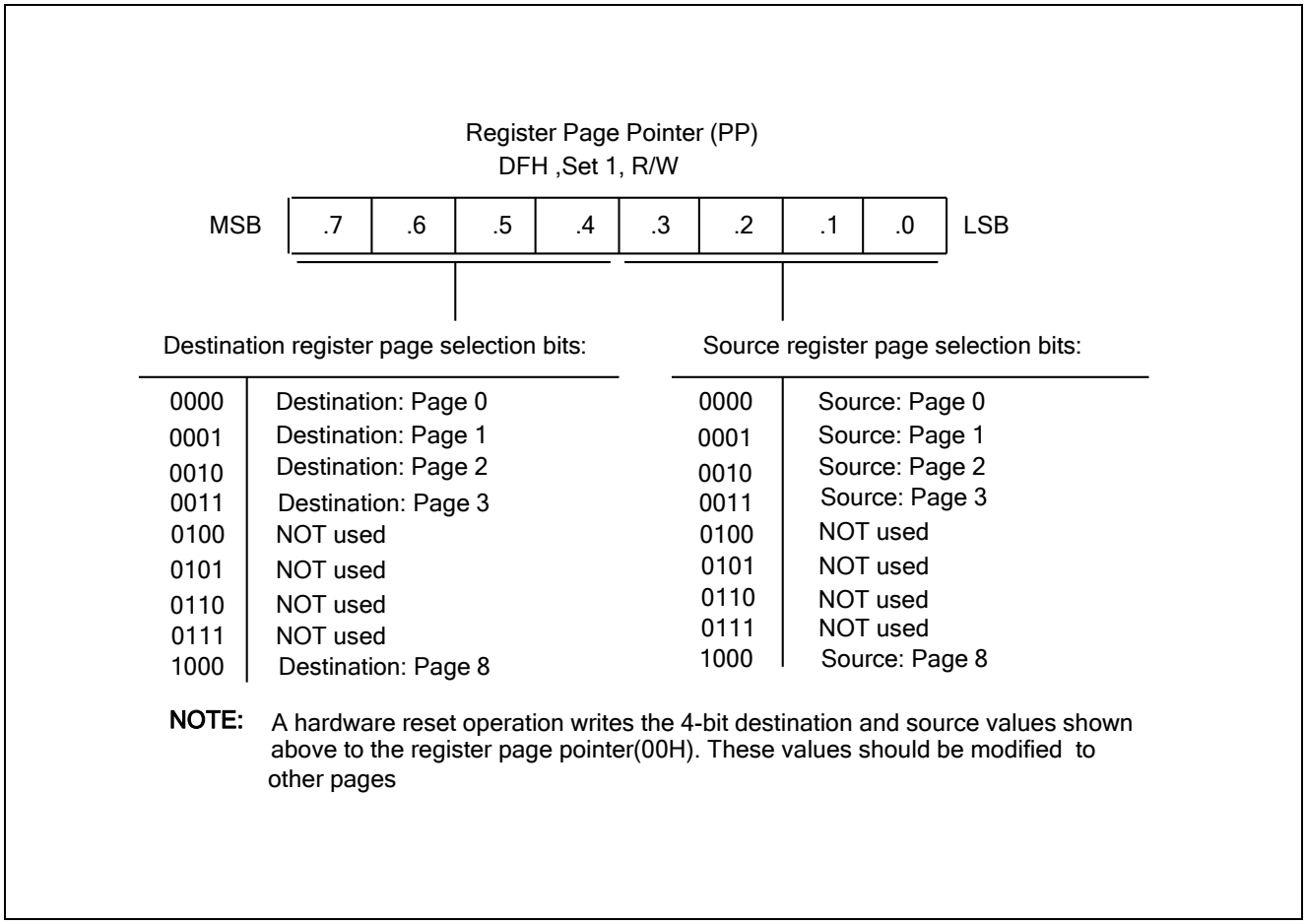


Figure 2-4 Register PP

2.6 Register Set 1

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

It expands the upper 32 byte area of this 64 byte space (E0H–FFH) into two 32 byte register banks, bank 0 and bank 1. It uses the set register bank instructions, SB0 or SB1, it uses the set register bank instructions, SB0 or SB1 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 46 mapped system and peripheral control registers. The lower 32 byte area contains 14 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 that it performs in other areas of the register file.

You can access the registers in set 1 location directly at any time by using register addressing mode. You can access the 16 byte working register area by using working register addressing only. Refer to Chapter 3 "Addressing Modes for more information about working register addressing.

2.7 Register Set 2

It logically duplicates to add another 64 bytes of register space when the same 64 byte physical space uses set 1 locations, C0H to FFH. This expanded area of the register file is called set 2. For S3F8S39/S3F8S35 microcontroller, you can access the set 2 address range (C0H–FFH) on register pages 0-3.

Addressing mode restrictions maintains the logical division of set 1 and set 2. You can use only addressing mode register to access set 1 location. To access registers in set 2, use indirect addressing mode or Indexed addressing mode register.

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

2.8 Prime Register Space

The lower 192 bytes (00H–BFH) of the S3F8S39/S3F8S35's 256 byte register pages is called prime register area. You can access prime registers by using any of the seven addressing modes. Refer to Chapter 3 "Addressing Modes" for more information.

It addresses the prime register area on page 0 immediately after a reset. To address prime registers on pages 0, 1, 2 and 3 you should set the register PP to the appropriate source and destination values.

Figure 2-5 illustrates the set 1, set 2 and prime area register map.

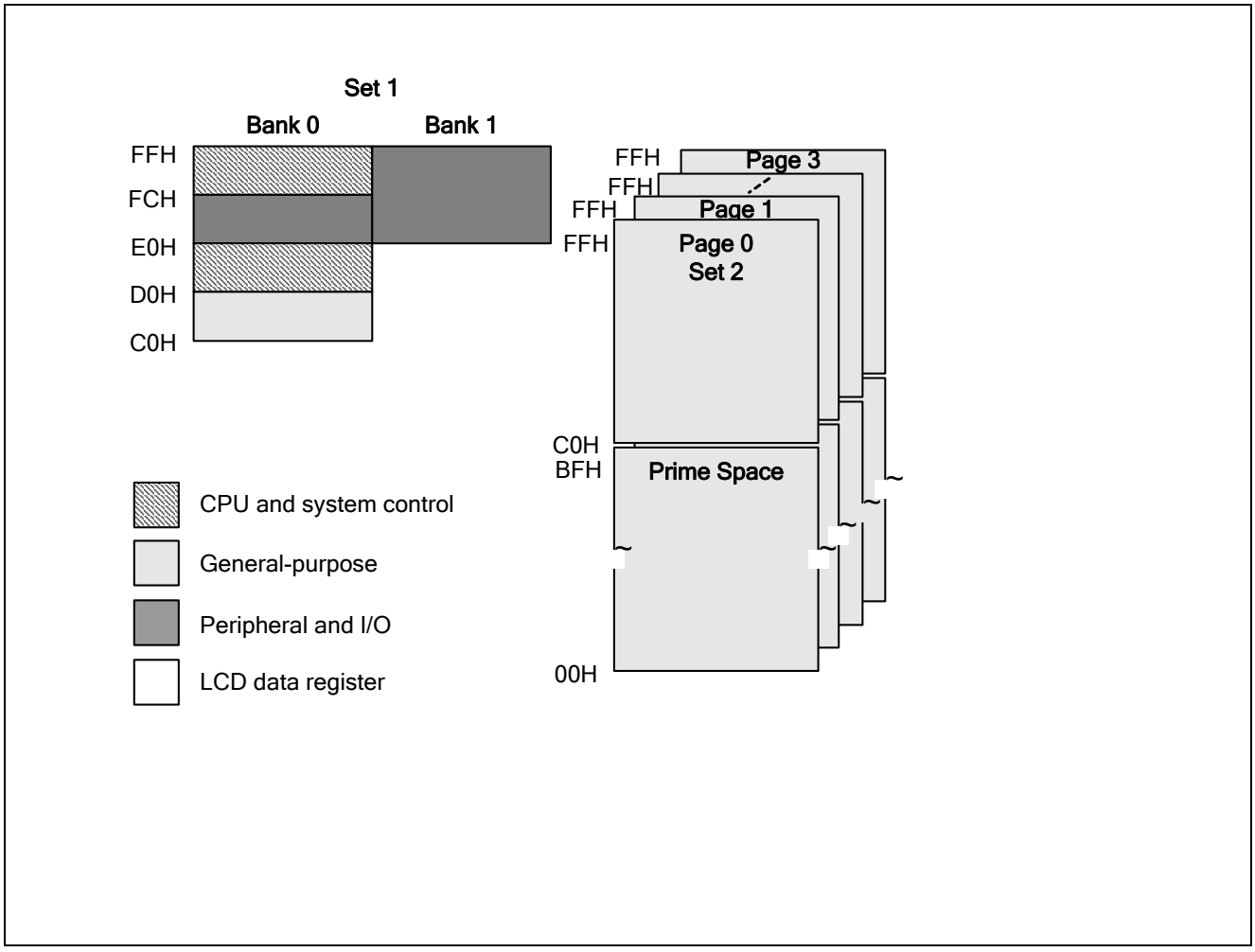


Figure 2-5 Set 1, Set 2, Prime Area Register Map

2.9 Working Registers

Instructions can access specific 8-bit registers or 16-bit register pairs using either 4-bit or 8-bit address fields. When it uses 4-bit working register addressing, the programmer can see the 256 byte register file that consists of thirty-two 8 byte register groups or "slices." Each slice consists of eight 8-bit registers.

Using the two 8-bit register pointers, RP1 and RP0, you can select the two working register slices at any time to form a 16-byte working register block. By 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 helps you to 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 similar 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. It contains the base addresses for the two selected 8-byte register slices 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 illustrates 8 byte working register areas (slices).

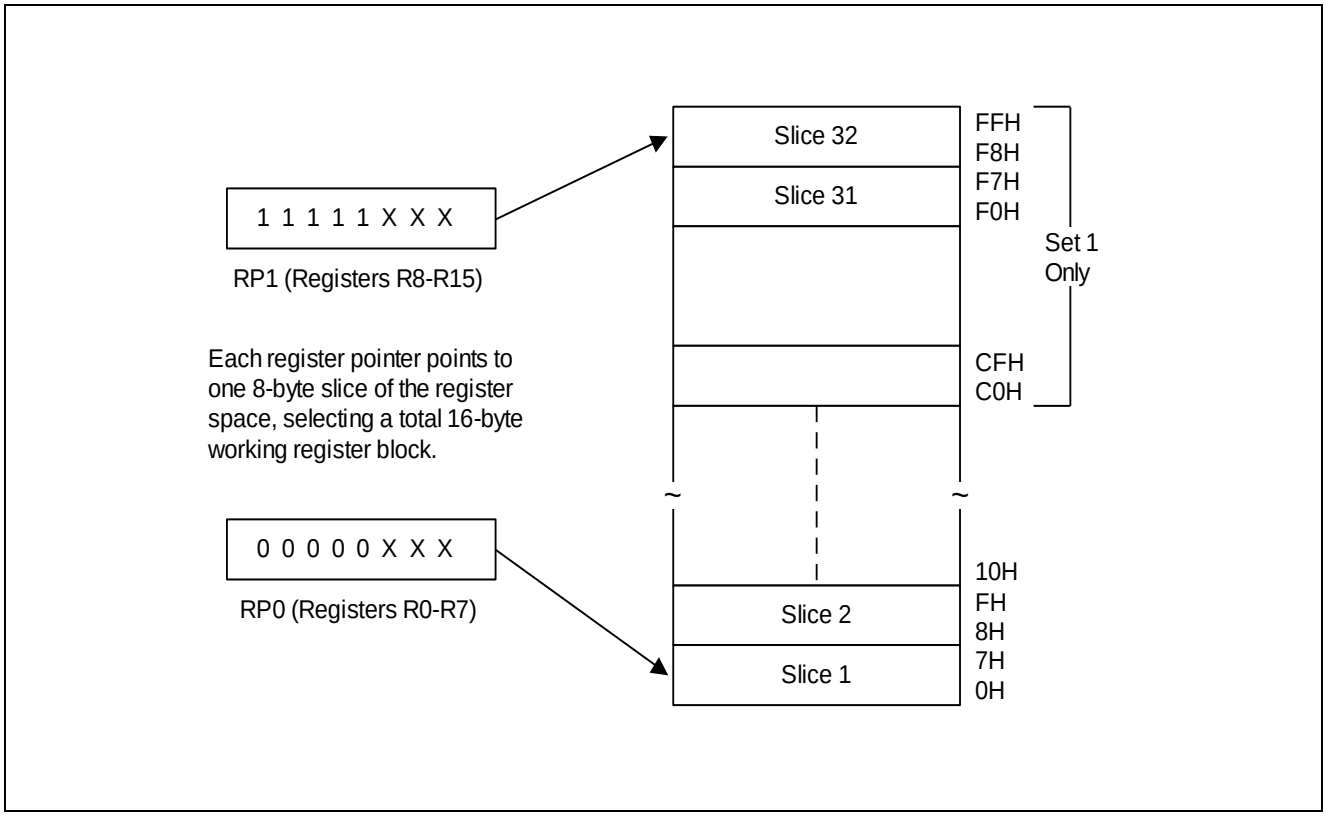


Figure 2-6 8 Byte Working Register Areas (Slices)

2.10 Using the Register Points

It uses register pointers RP0 and RP1 that it maps to address D6H and D7H in set 1 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, load a new value to RP0 and RP1 by using an SRP or LD instruction. Refer to *Figure 2-7* and *Figure 2-8* for more information.

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. However, you cannot use the register pointers to select a working register space in set 2, C0H–FFH, because you can access these locations only by 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, RP0 should point to the "lower" slice and RP1 should point to the "upper" slice (Refer to *Figure 2-8* for more information). In some cases, it may be necessary to define working register areas in different (non-contiguous) areas of the register file. In *Figure 2-9*, RP0 points to the "upper" slice and RP1 to the "lower" slice.

With the register pointer pointing 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 2-1 shows register pointer setting.

Example 2-1 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 illustrates contiguous 16 byte working register block.

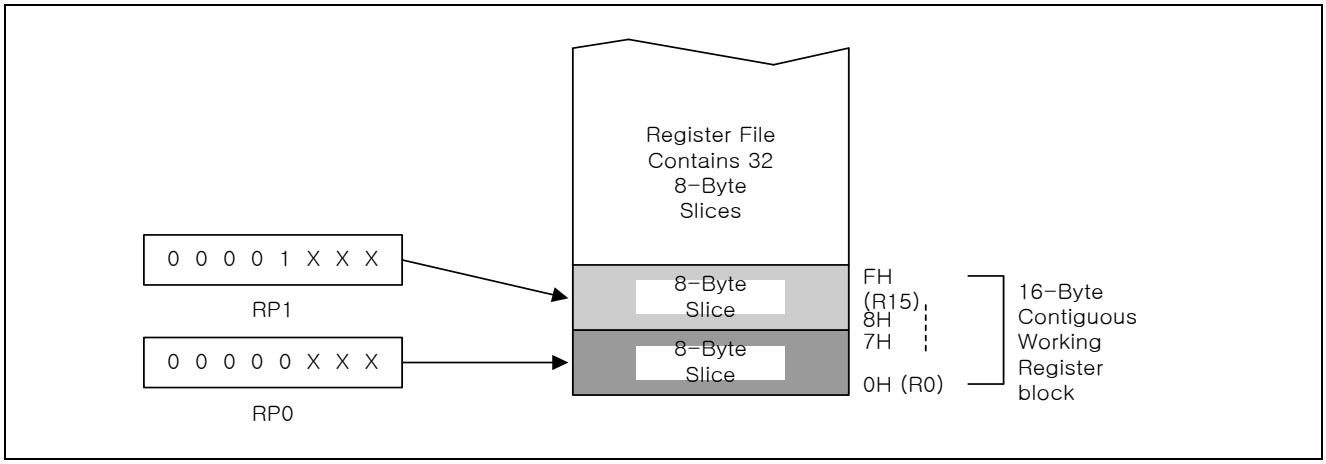


Figure 2-7 Contiguous 16 Byte Working Register Block

Figure 2-8 illustrates the non-contiguous 16 byte working register block.

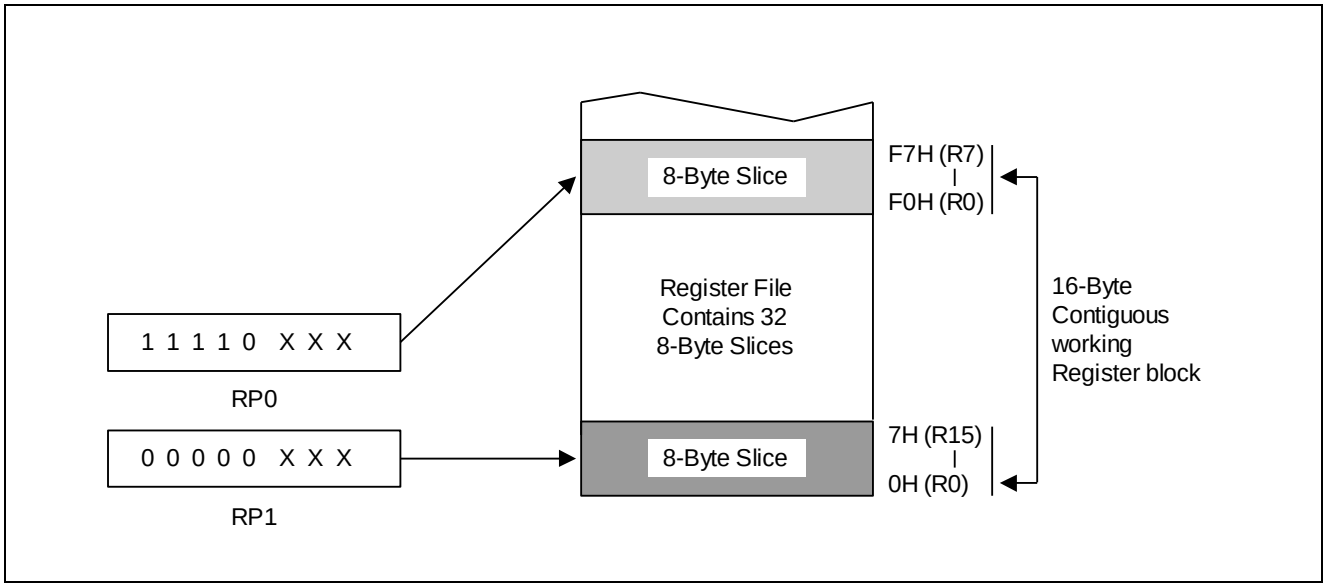


Figure 2-8 Non-Contiguous 16 Byte Working Register Block

Example 2-2 shows how to use the RPs to calculate the sum of a series of registers.

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

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

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

The sum of these six registers, 6FH, is located in the register R0 (80H). The instruction string that it uses 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.

2.11 Register Addressing

The S3C8-series register architecture provides an efficient method of working register addressing. This 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 can use:

- A register pointer to specify an 8 byte working register space in the register file.
- An 8-bit register within that space.

You can address the registers 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. The least significant byte is always stored in the next (+ 1) odd-numbered register.

Working register addressing is different 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 illustrates the 16-bit register pair.

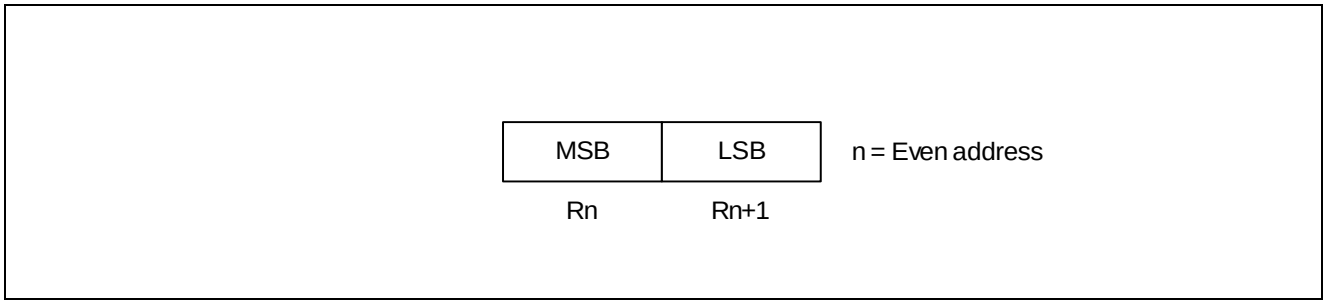


Figure 2-9 16-Bit Register Pair

Figure 2-10 illustrates the register file addressing.

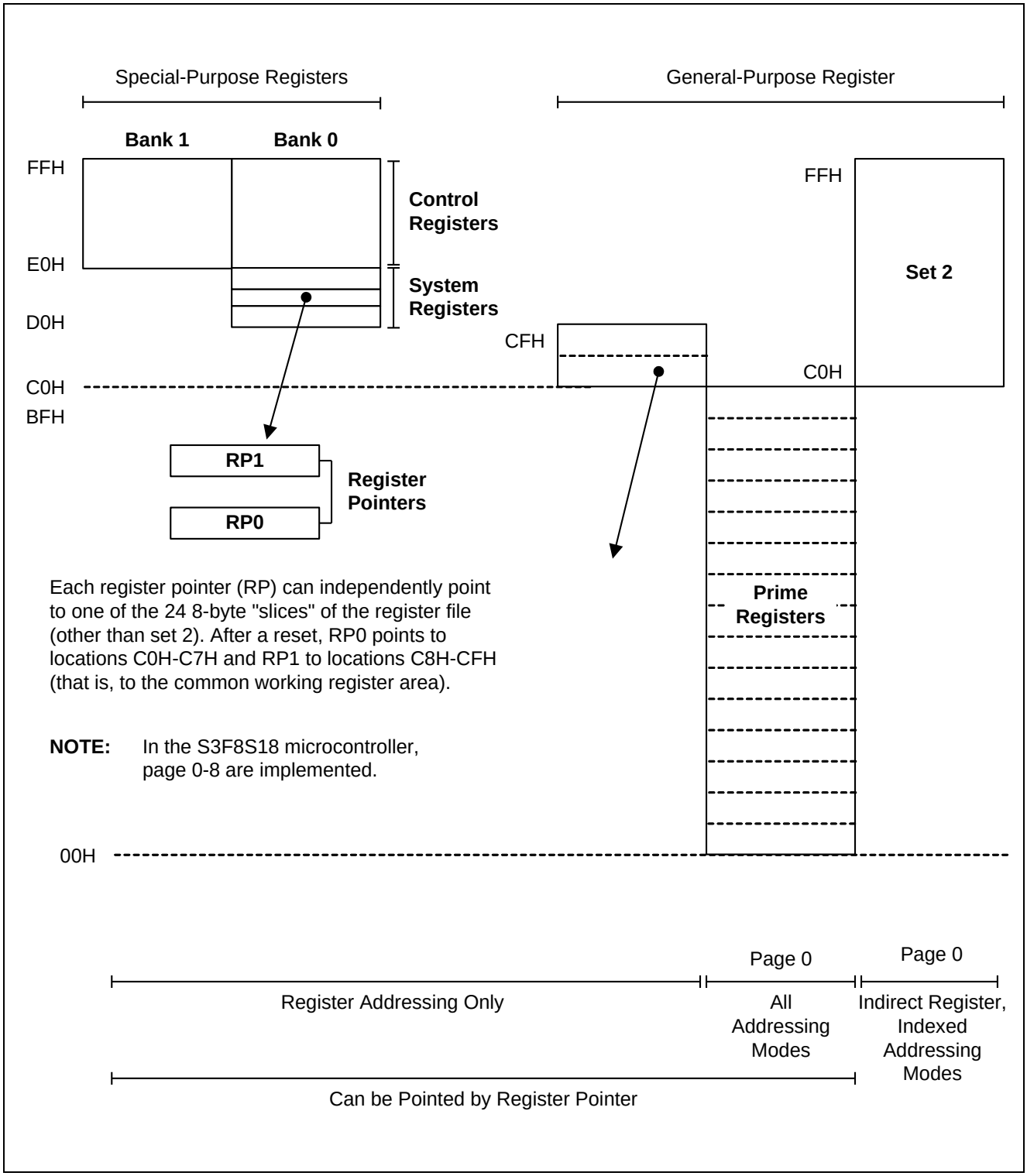


Figure 2-10 Register File Addressing

2.12 Common Working Register Area

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. These register slices are:

- RP0 → C0H–C7H
- RP1 → C8H–CFH

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

Figure 2-11 illustrates the common working register area.

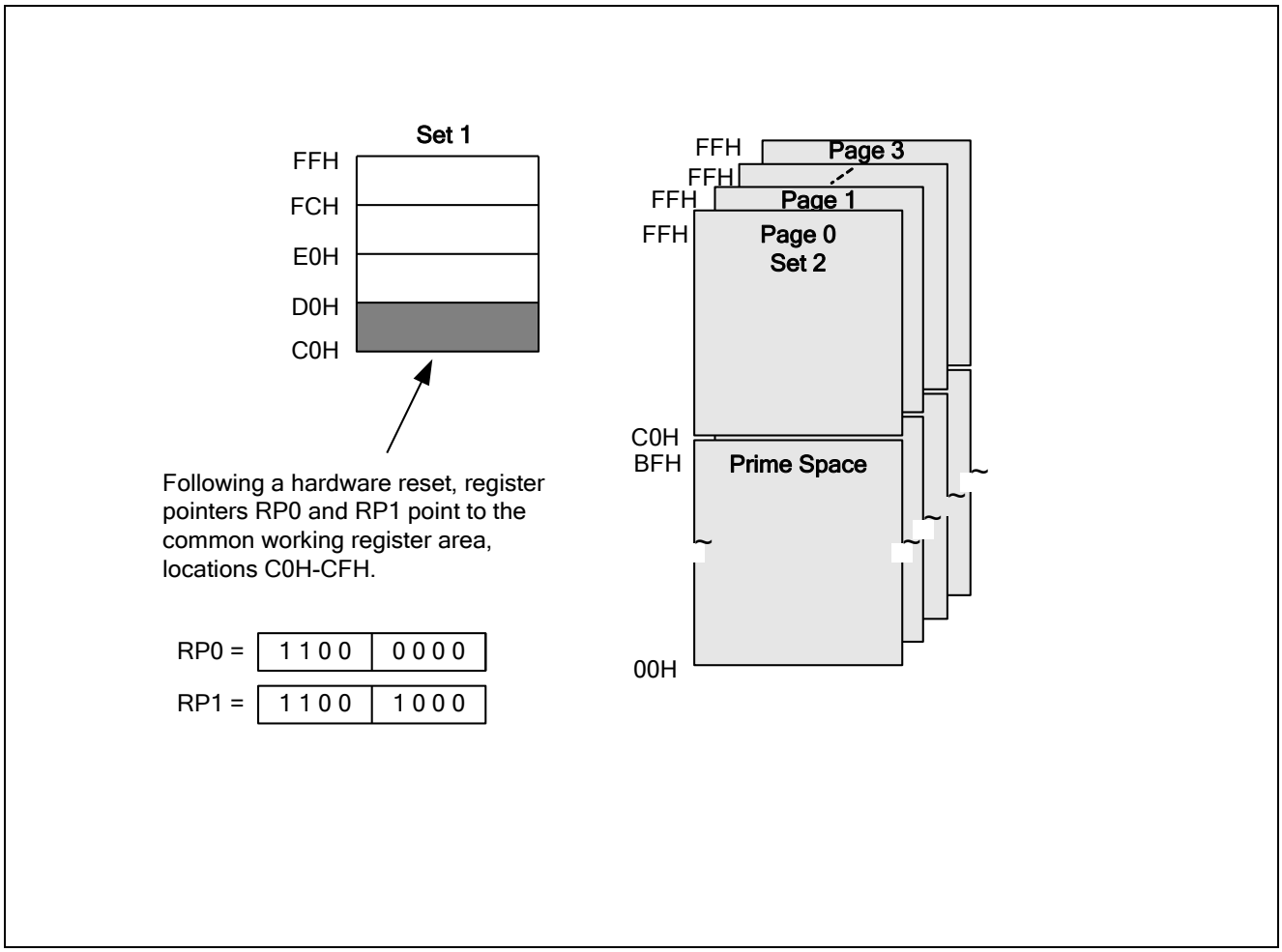


Figure 2-11 Common Working Register Area

As Example 2-3**Error! Not a valid bookmark self-reference.** shows, you should access working registers in the common area, locations C0H-CFH, by using working register addressing mode only.

Example 2-3 Addressing the Common Working Register Area

```
Examples 1.

      LD      0C2H, 40H                    ; Invalid addressing mode!

      Use working register addressing instead:

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

Examples 2.

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

      Use working register addressing instead:

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

2.13 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". This makes it possible for instructions to access working registers efficiently by using short 4-bit addresses. When an instruction addresses a location in the selected working register area, the address bits are concatenated in this manner 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 *Figure 2-12* illustrates, it concatenates the result of this operation is that the five high-order bits from the register pointer 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 always points to an address in the same 8-byte register slice.

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

Figure 2-12 illustrates the 4-bit working register addressing.

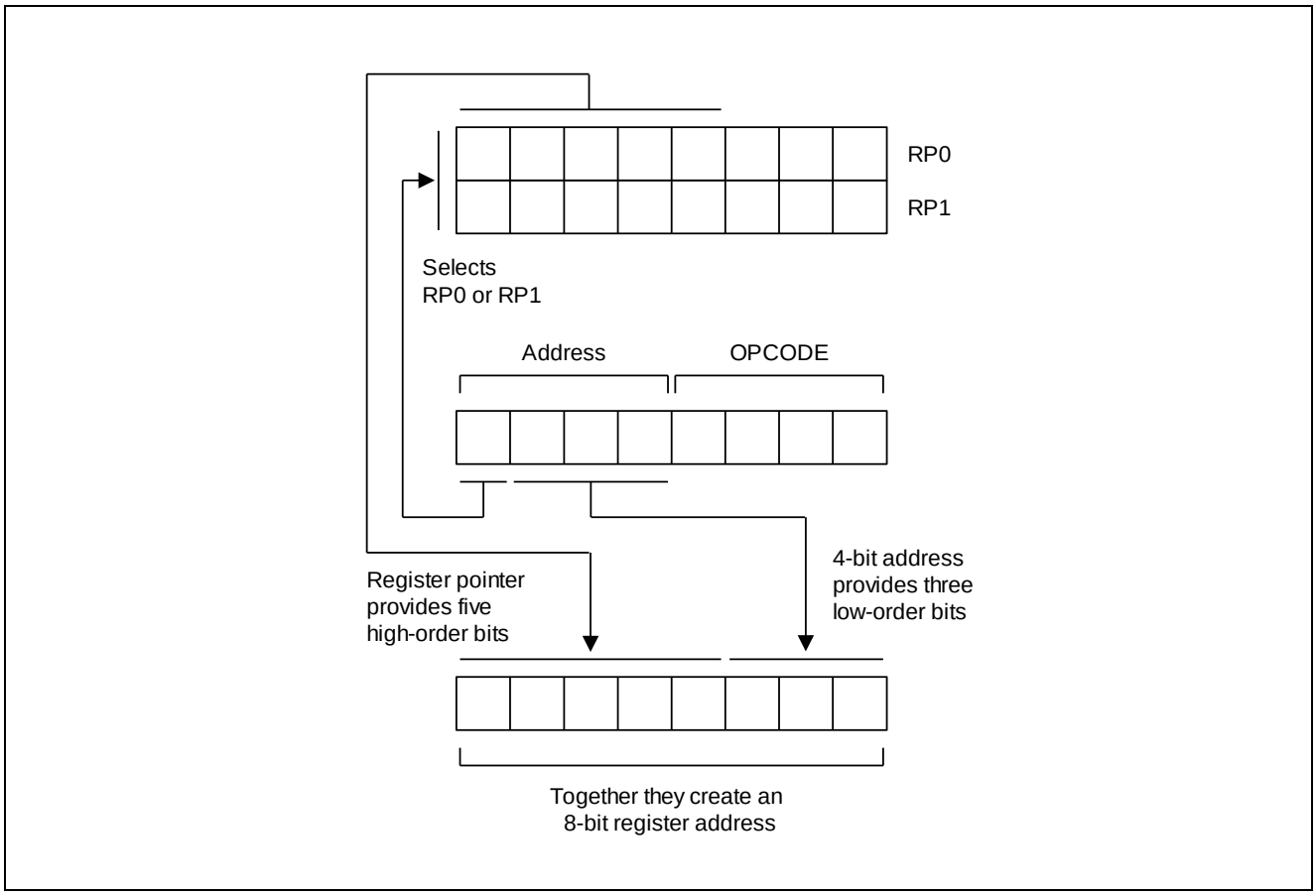


Figure 2-12 4-Bit Working Register Addressing

Figure 2-13 illustrates the 4-bit working register addressing example.

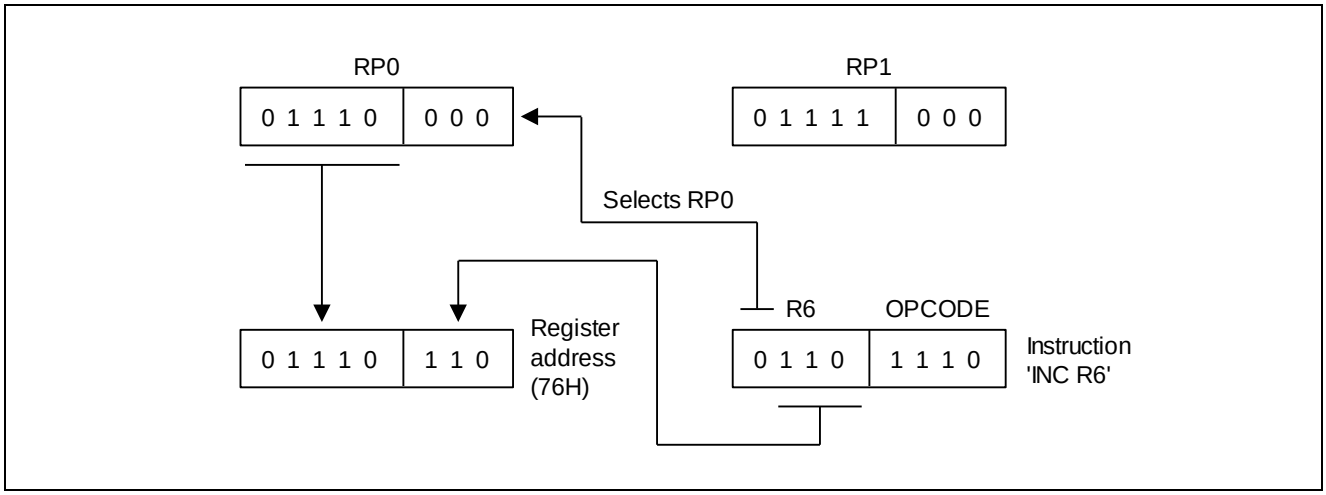


Figure 2-13 4-Bit Working Register Addressing Example

2.14 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 should contain the 4-bit value, "1100B." The value indicates that the remaining four bits have the same effect as 4-bit working register addressing.

As *Figure 2-14* illustrates, it concatenates the lower nibble of the 8-bit address in similar manner 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 original instruction provides the three low-order bits of the complete address.

Figure 2-15 illustrates 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 8-bit instruction address provide the three low-order bits of the register address (011). It concatenates the five address bits from RP1 and the three address bits from the instruction to form the complete register address, 0ABH (10101011B).

Figure 2-14 illustrates the 8-Bit working register addressing.

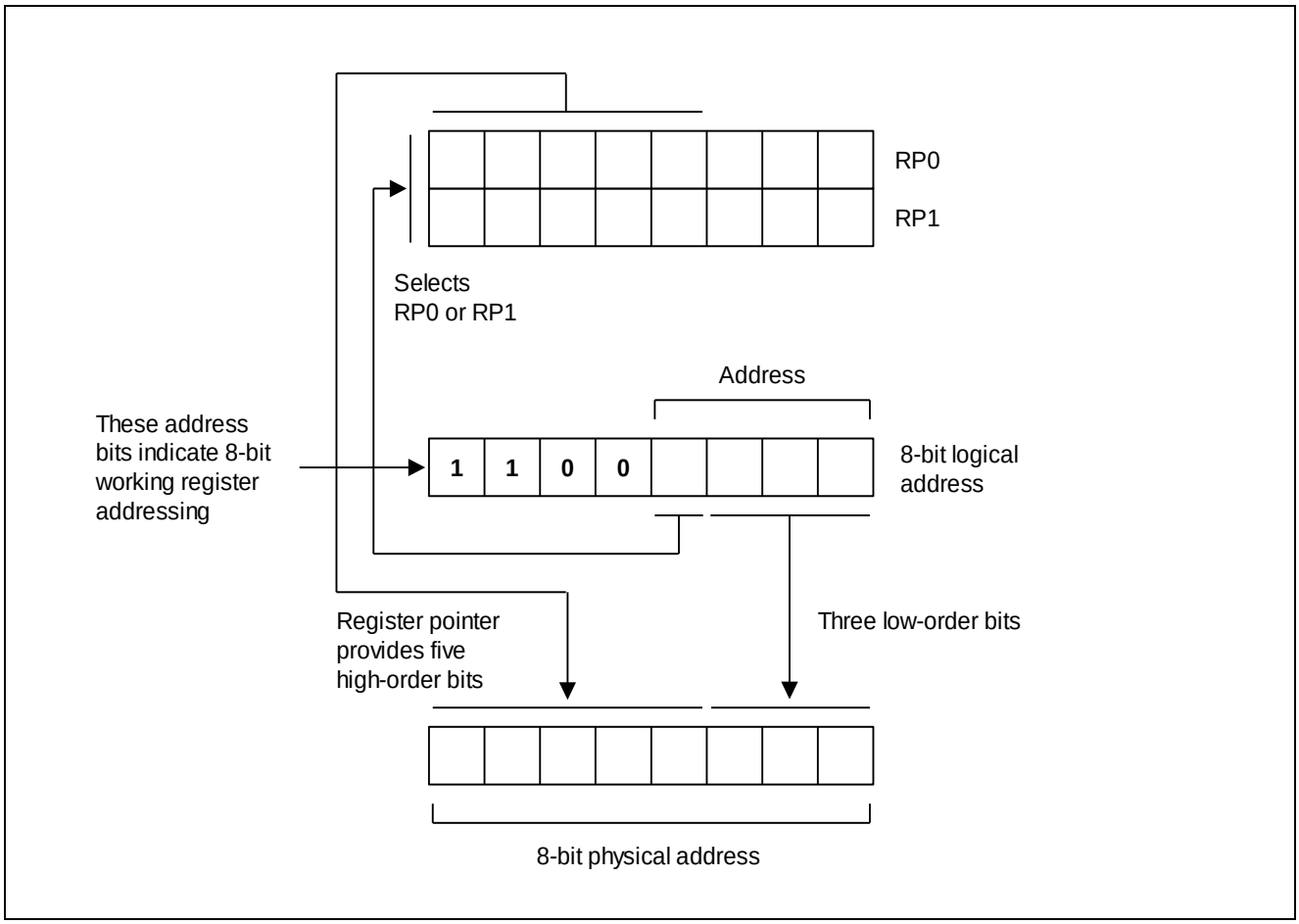


Figure 2-14 8-Bit Working Register Addressing

Figure 2-15 illustrates the 8-Bit working register addressing example.

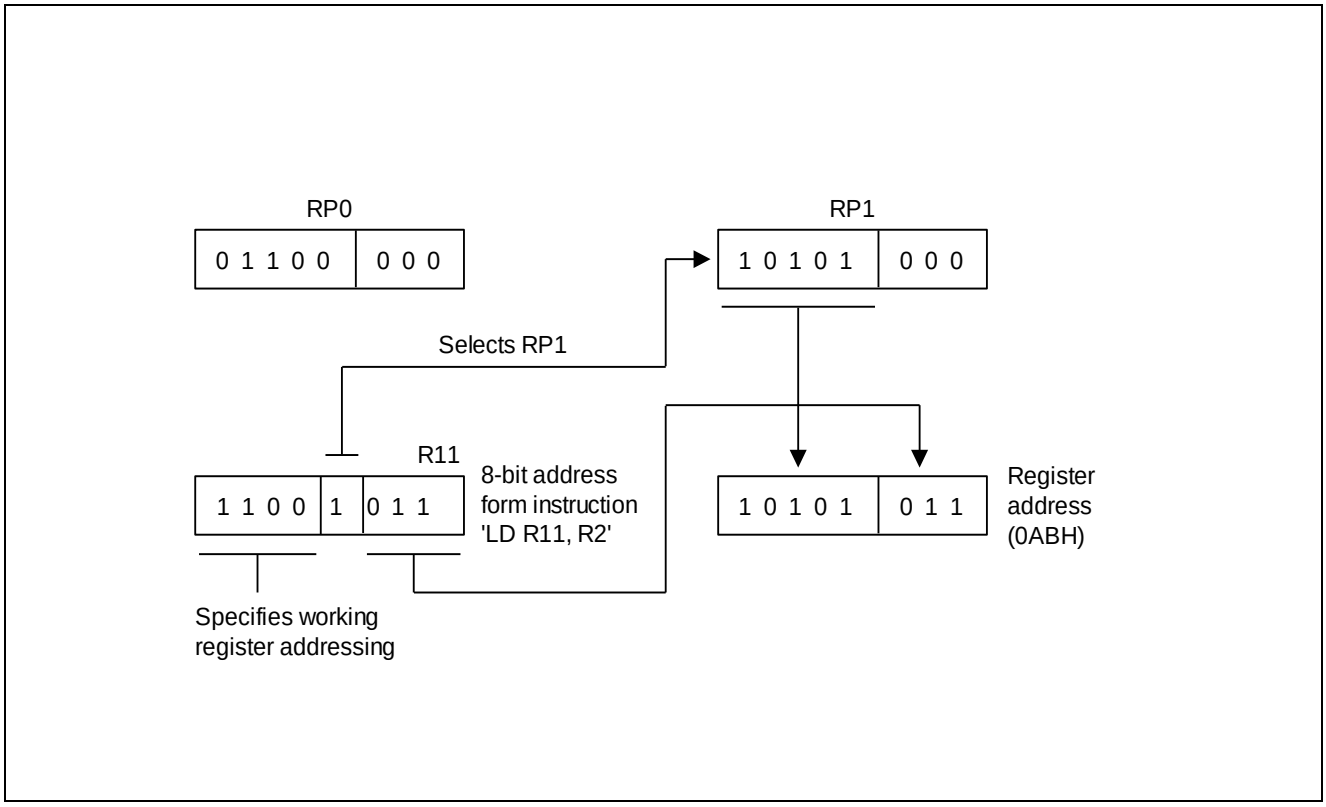


Figure 2-15 8-Bit Working Register Addressing Example

2.15 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 S3F8S39/S3F8S35 architecture supports stack operations in the internal register file.

2.15.1 Stack Operations

It stores return addresses for procedure calls, interrupts, and data on the stack. A CALL instruction saves the contents of the PC to stack and RET instruction restores. When an interrupt occurs, it pushes the contents of the PC and the FLAGS register to the stack. The IRET instruction then pops these values back to their original locations. It always decreases the stack address value by one before a push operation and increases by one after a pop operation. The stack pointer (SP) always points to the stack frame stored on the top of the stack, as *Figure 2-16* illustrates.

Figure 2-16 illustrates the stack operations.

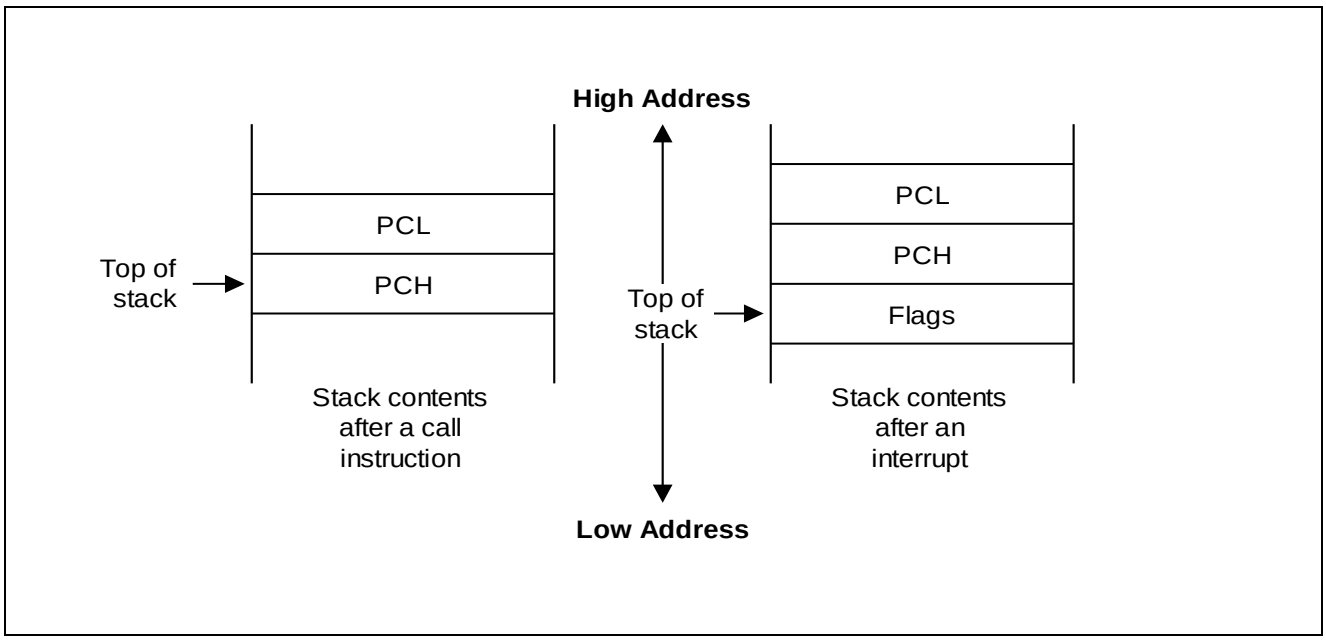


Figure 2-16 Stack Operations

2.15.2 User-Defined Stacks

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

2.15.3 Stack Pointers (SPL, SPH)

Register locations D8H and D9H contain the 16-bit 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 it only implements internal memory space in the S3F8S39/S3F8S35 microcontroller, it should initialize to an 8-bit value in the range 00H–FFH. The SPH register is not required and it can use it as a general-purpose register, if necessary.

When the SPL register contains only SP 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 overflows (or underflows) to the SPH register. This occurs by overwriting any other data that is currently stored there. To avoid overwriting data in the SPH register, initialize the SPL value to "FFH" instead of "00H".

Example 2-4 shows how to perform stack operations in the internal register file by using PUSH and POP instructions:

Example 2-4 Standard Stack Operations Using PUSH and POP

```
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 SAM8RC instructions may be condition codes, immediate data, or a location in the register file, program memory, or data memory.

The S3C-series instruction set supports seven addressing modes. Not all of these addressing modes are available for each instruction.

The seven addressing modes with 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 (R)

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

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* for more information.

Figure 3-1 illustrates the register addressing.

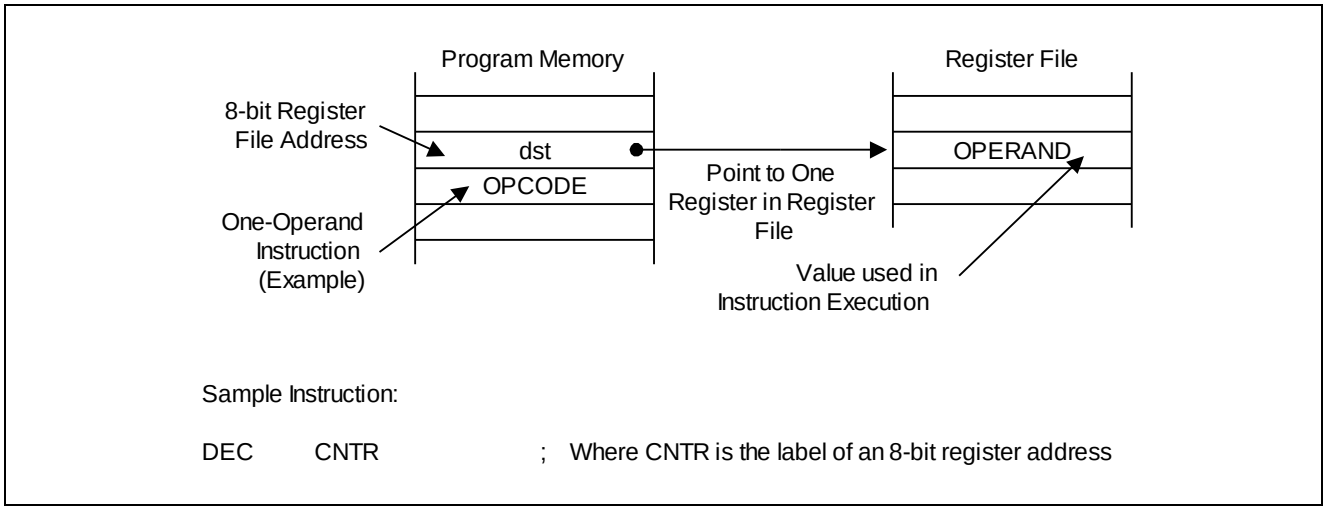


Figure 3-1 Register Addressing

Figure 3-2 illustrates the working register addressing.

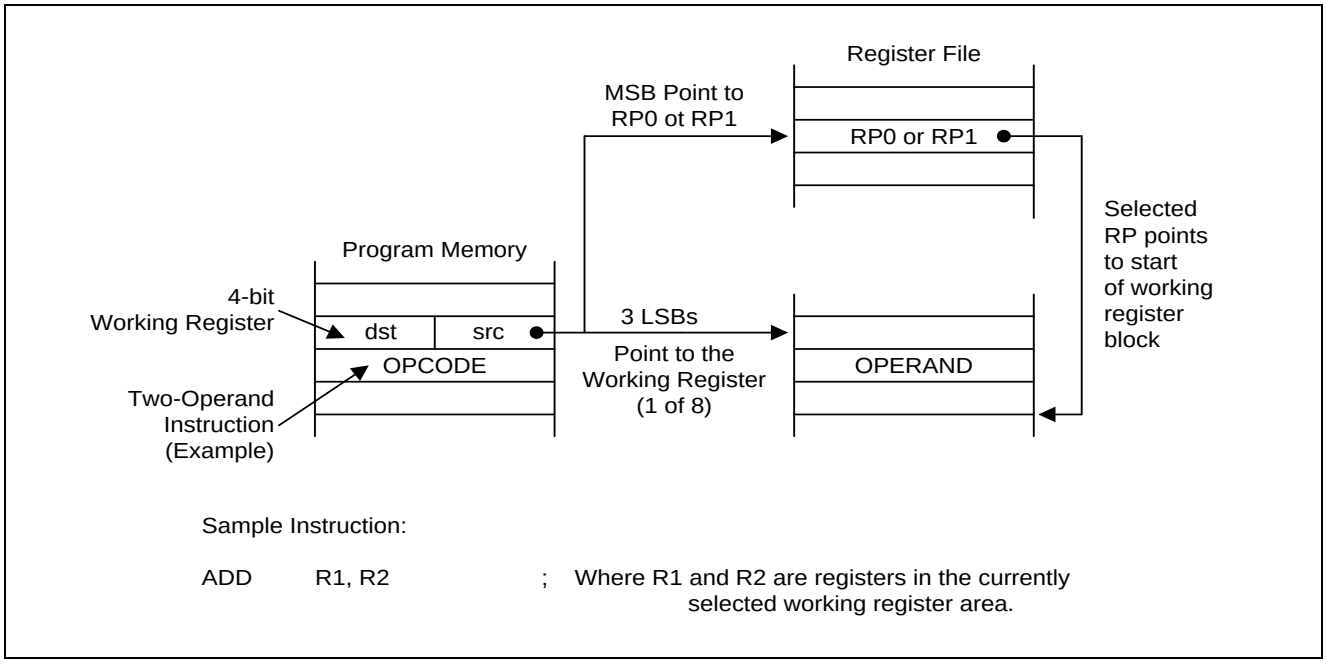


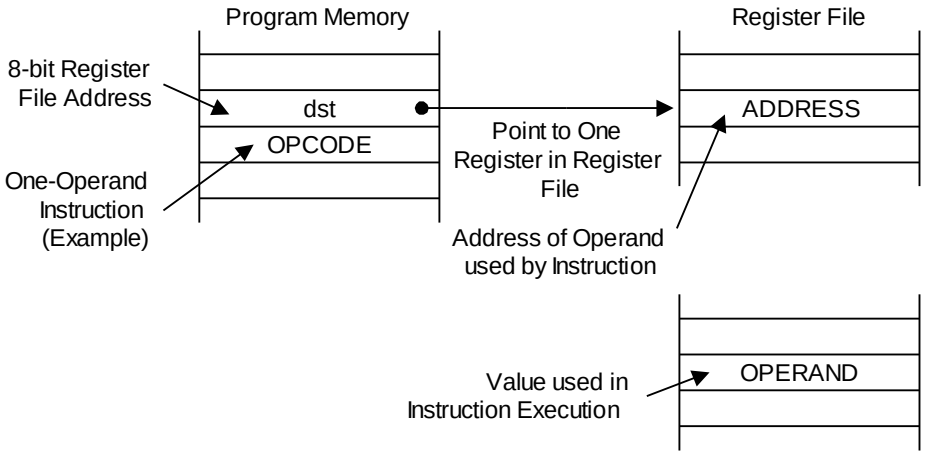
Figure 3-2 Working Register Addressing

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. (Refer to *Figure 3-3* through *Figure 3-6* for more information).

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

Figure 3-3 illustrates the indirect register addressing to register file.



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 illustrates the indirect register addressing to program memory.

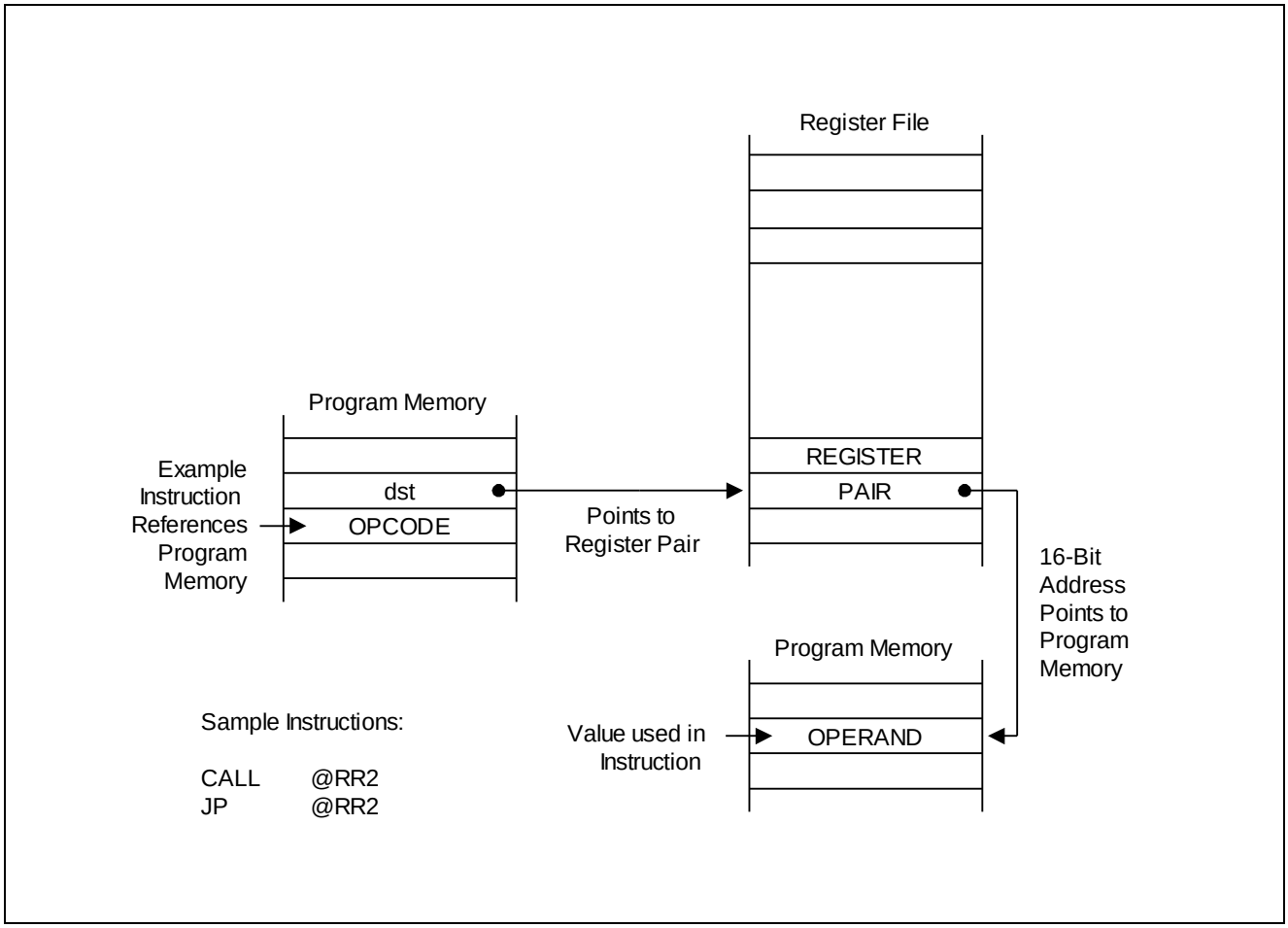


Figure 3-4 Indirect Register Addressing to Program Memory

Figure 3-5 illustrates the indirect working register addressing to register file.

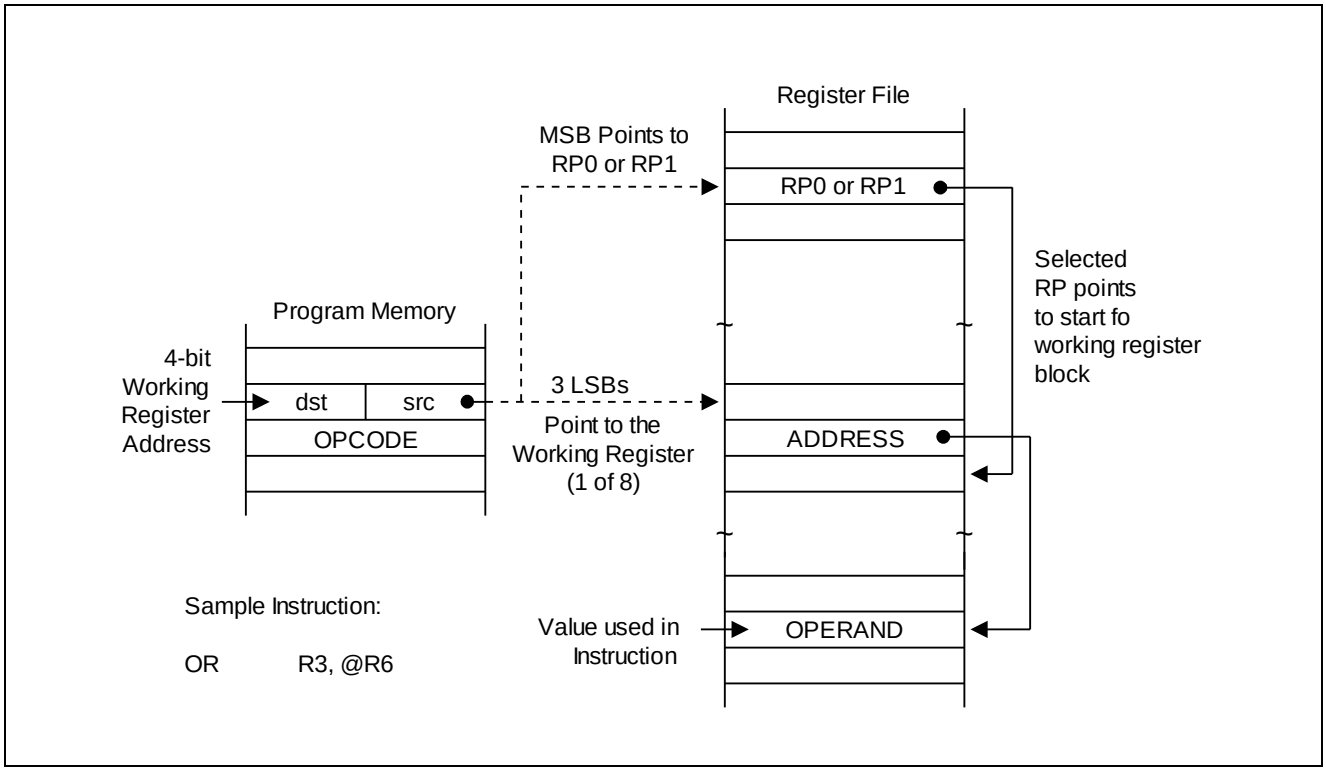


Figure 3-5 Indirect Working Register Addressing to Register File

Figure 3-6 illustrates the indirect working register addressing to program or data memory.

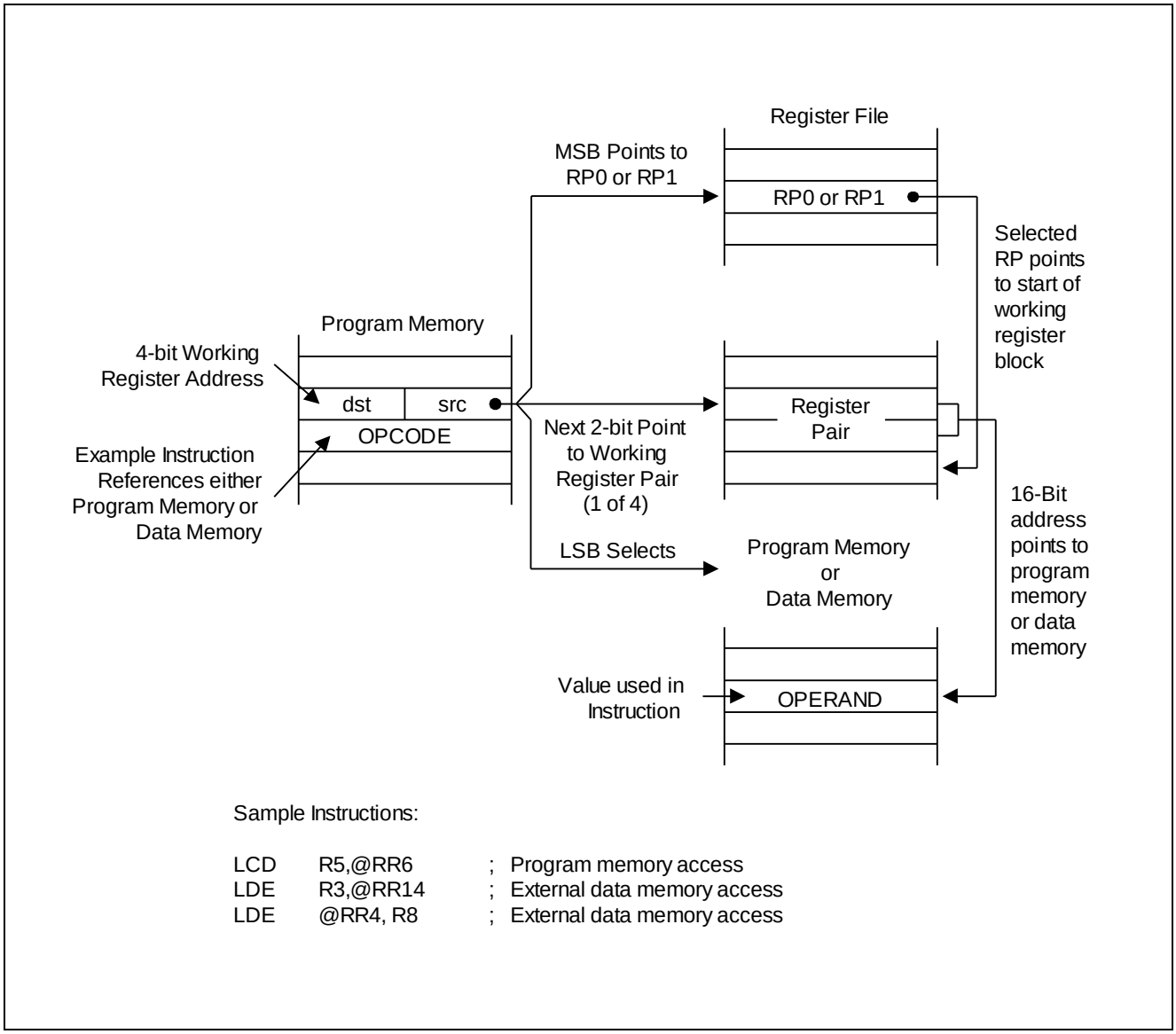


Figure 3-6 Indirect Working Register Addressing to Program or Data Memory

3.4 Indexed Addressing Mode (X)

The Indexed adds an offset value to a base address during instruction execution to calculate the effective operand address (Refer to *Figure 3-7* for more information). You can use Indexed addressing mode to access locations in the internal register file or in external memory. 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 (Refer to *Figure 3-8* for more information).

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 (Refer to *Figure 3-9* for more information).

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

Figure 3-7 illustrates the indexed addressing to register file.

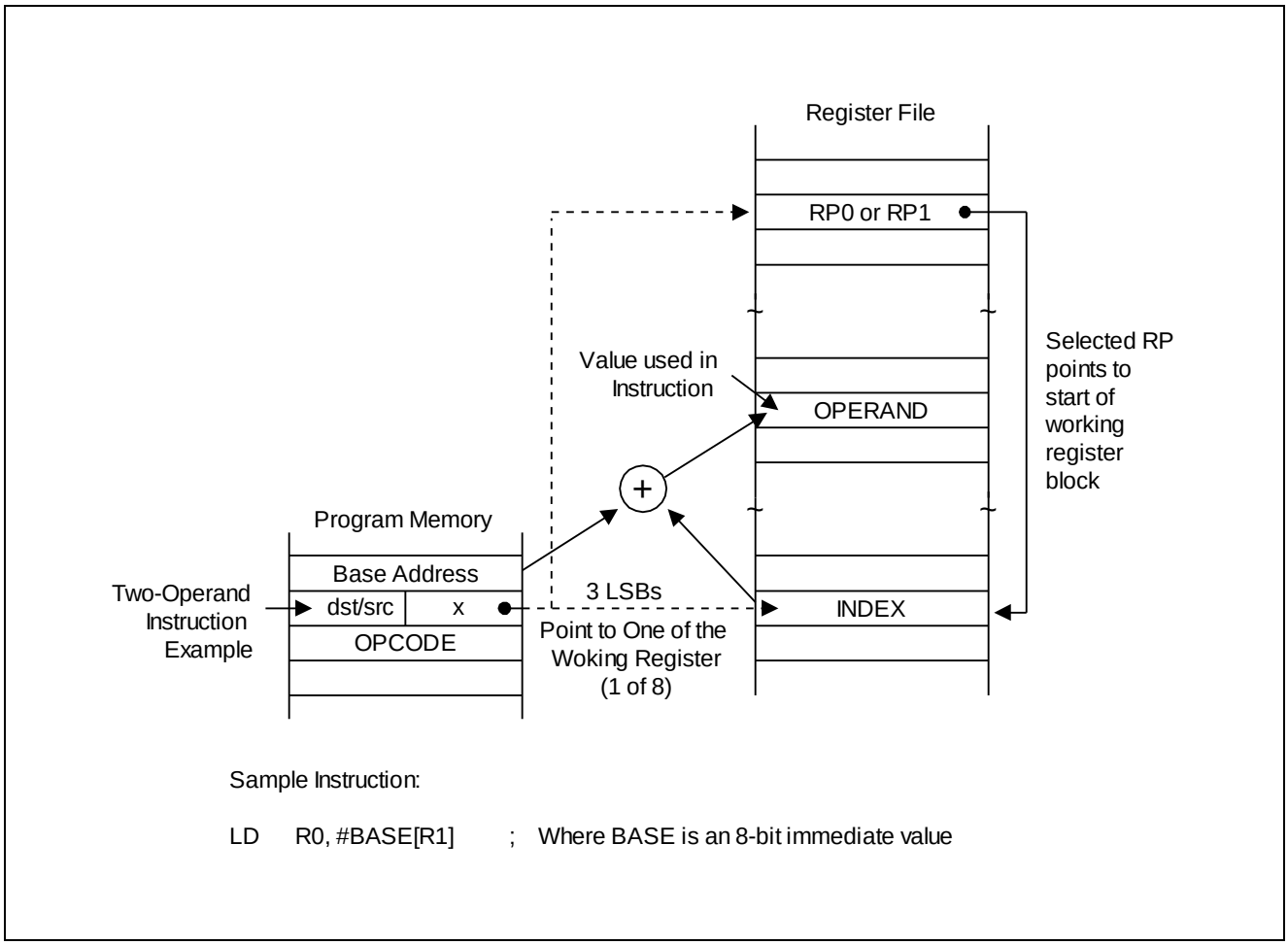


Figure 3-7 Indexed Addressing to Register File

Figure 3-8 illustrates the indexed addressing to program or data memory with short offset.

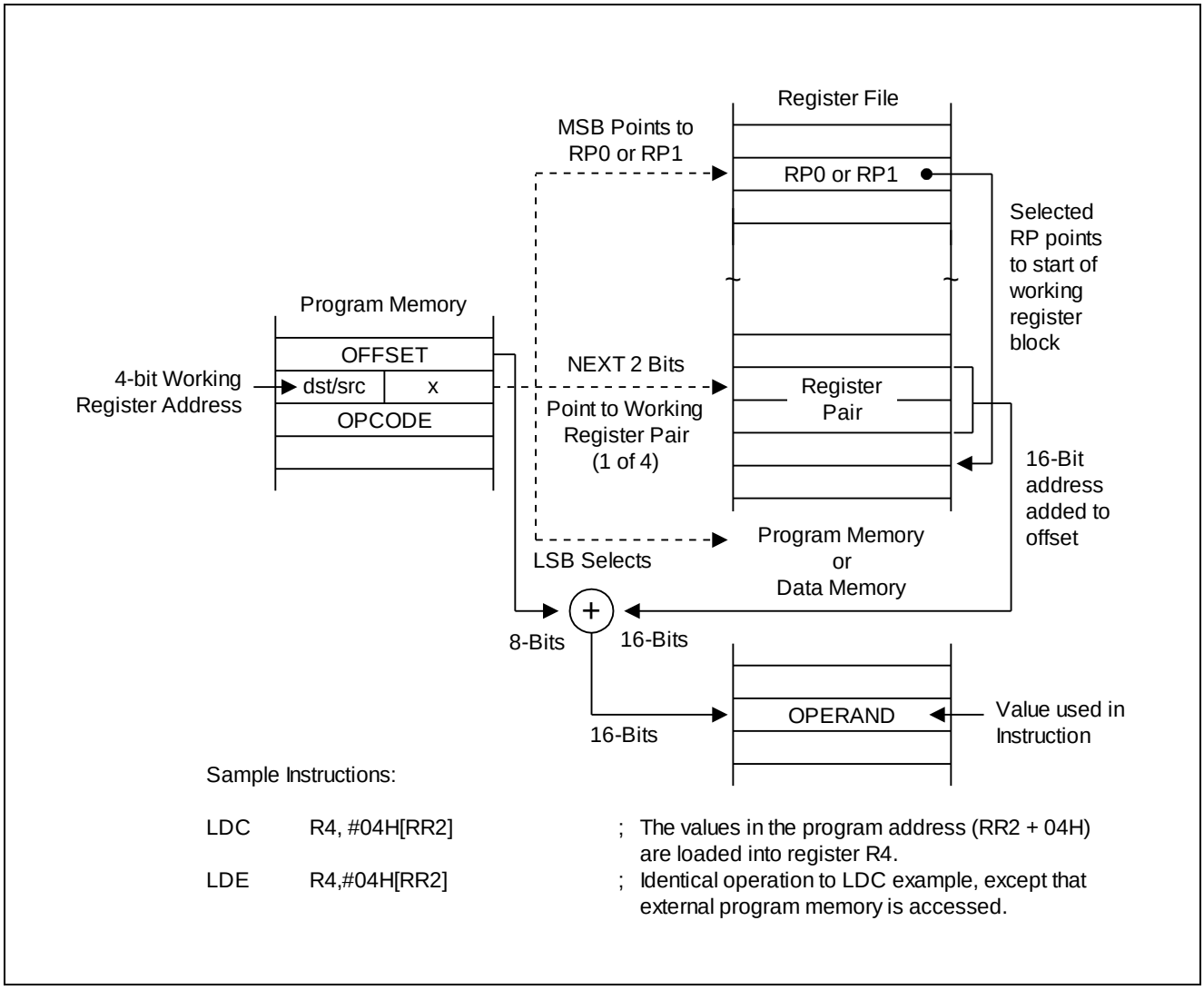


Figure 3-8 Indexed Addressing to Program or Data Memory with Short Offset

Figure 3-9 illustrates the indexed addressing to program or data memory.

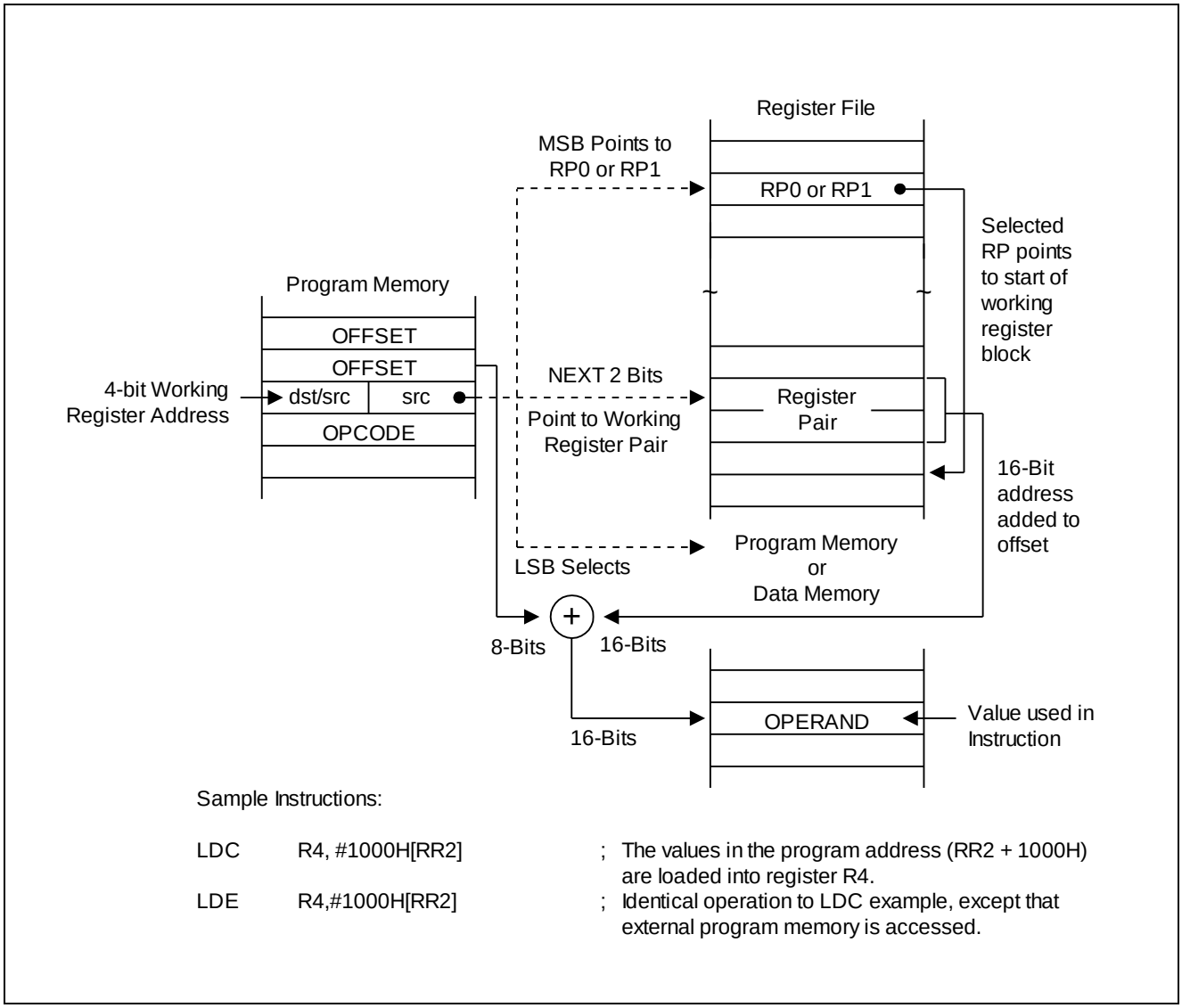


Figure 3-9 Indexed Addressing to Program or Data Memory

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.

If it implements, 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).

Figure 3-10 illustrates the direct addressing for load instructions.

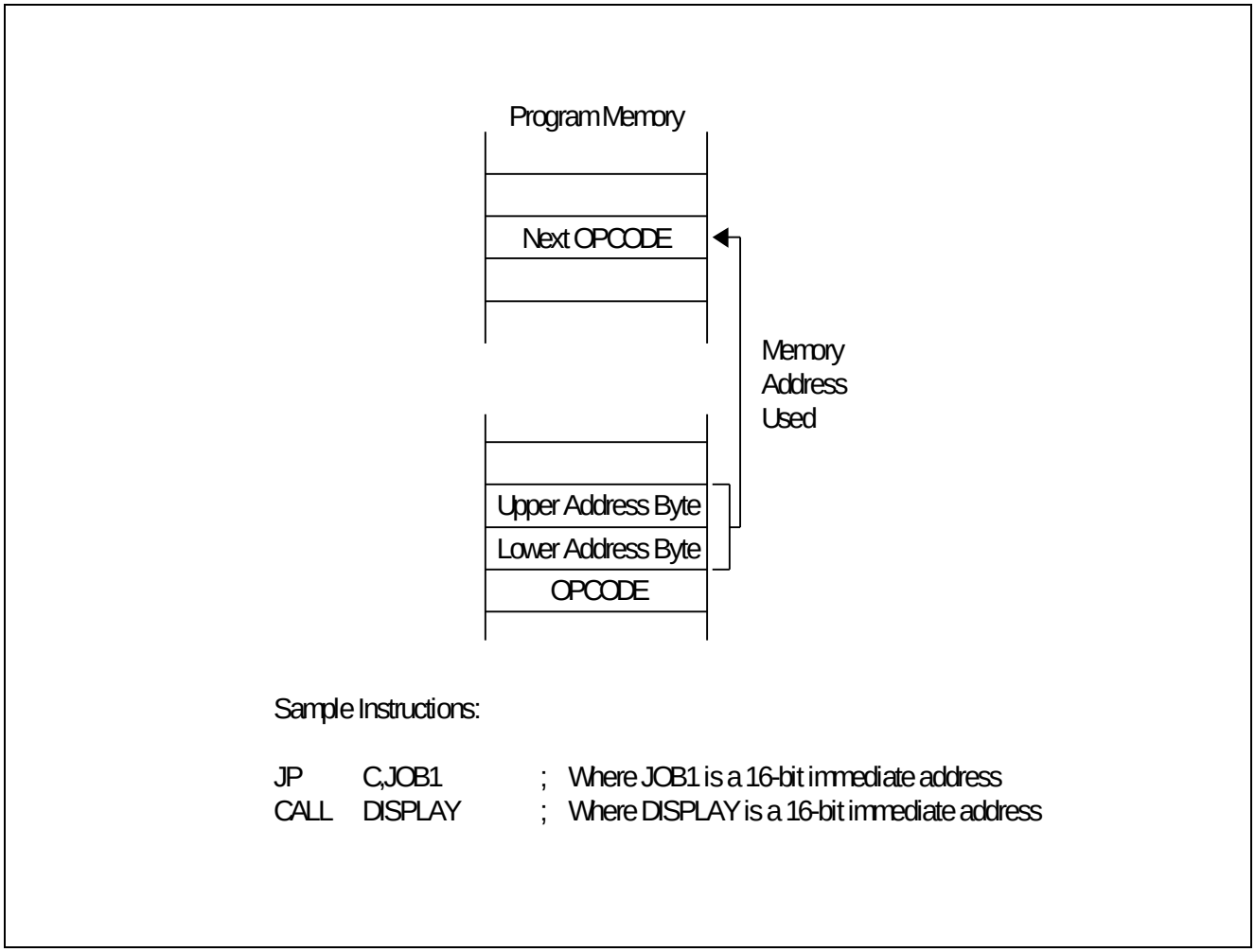


Figure 3-10 Direct Addressing for Load Instructions

Figure 3-11 illustrates the direct addressing for call and jump instructions.

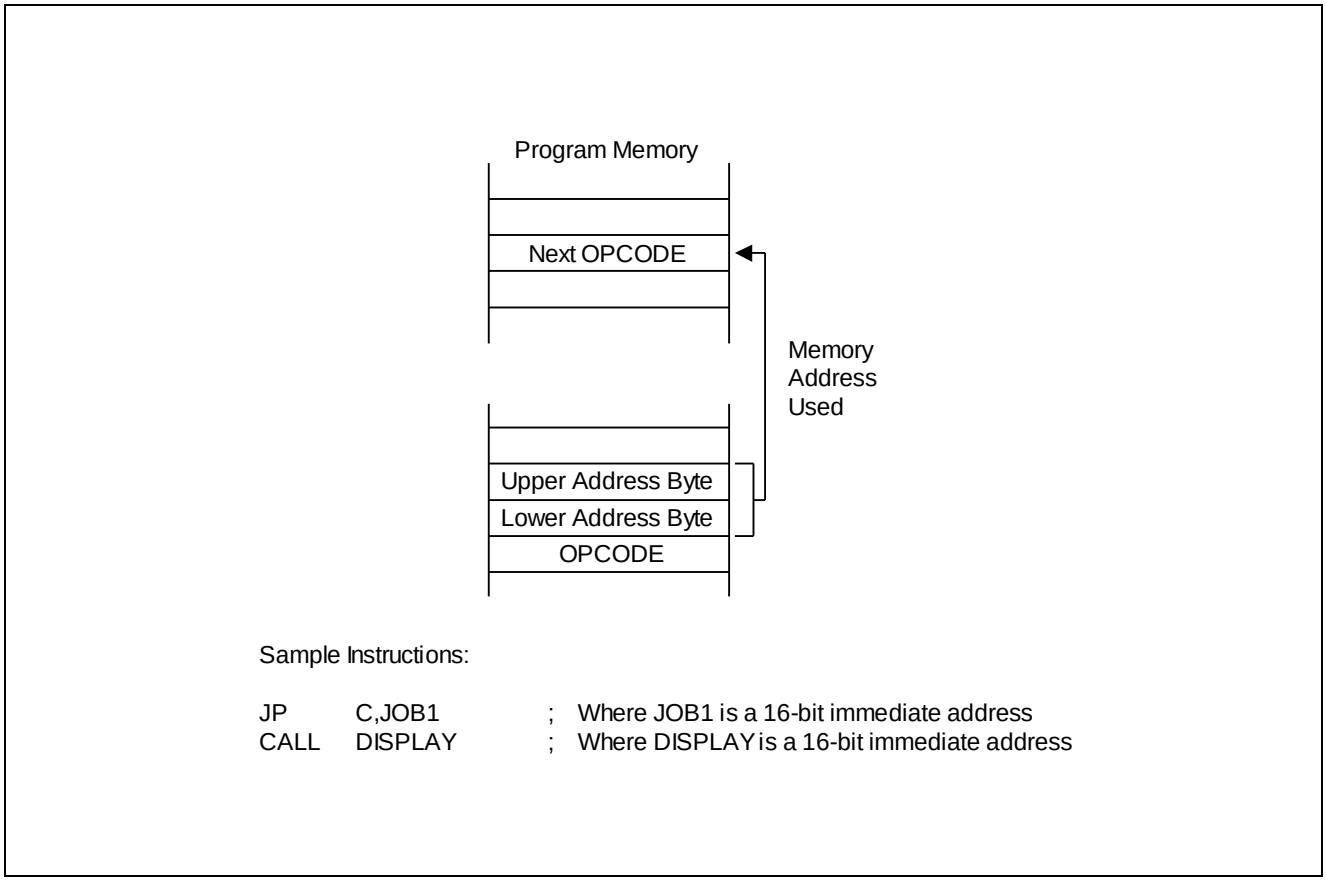


Figure 3-11 Direct Addressing for Call and Jump Instructions

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. It supplies only an 8-bit address in the instruction. It assumes the upper bytes of the destination address to be all zeros.

Figure 3-12 illustrates the indirect addressing.

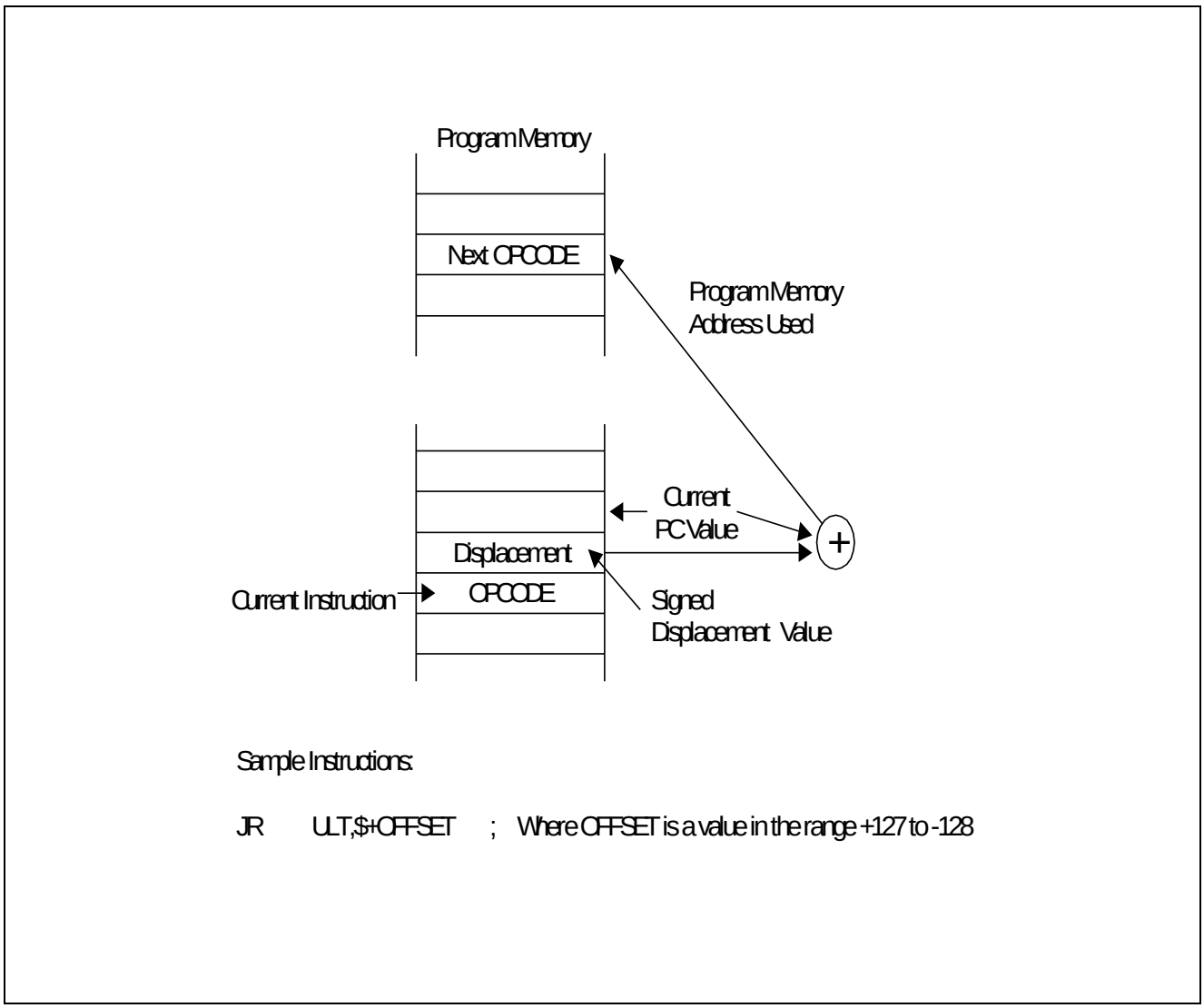


Figure 3-12 Indirect Addressing

3.7 Relative Address Mode (RA)

In Relative Address (RA) mode, a two-complement signed displacement between – 128 and + 127 is specified in the instruction. Then it adds the displacement value to the current PC value. The result is the address of the next instruction to execute. 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 illustrates the relative addressing.

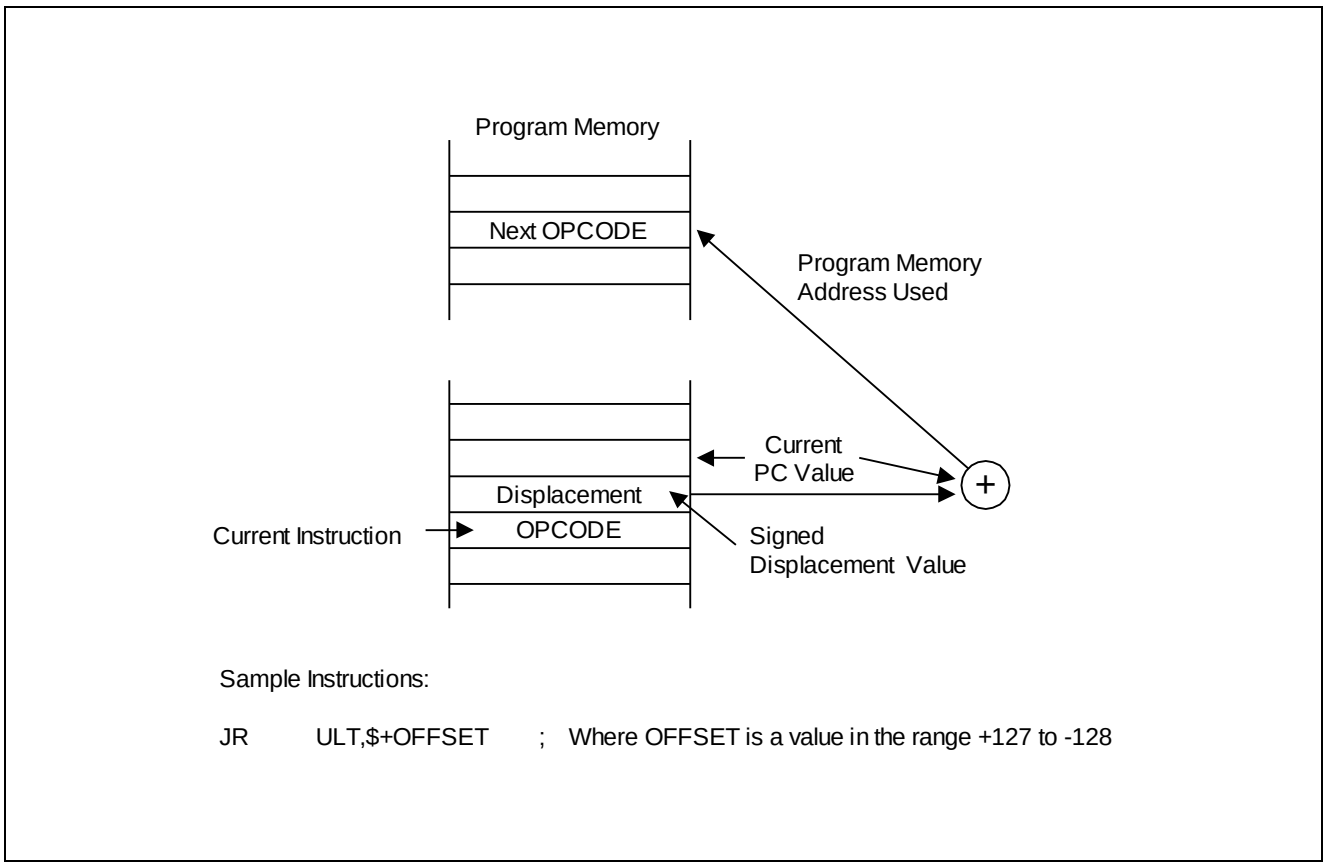


Figure 3-13 Relative Addressing

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. The operand may be 1 byte or one word in length, depending on the instruction it uses. Immediate addressing mode is useful for loading constant values into registers.

Figure 3-14 illustrates the immediate addressing.

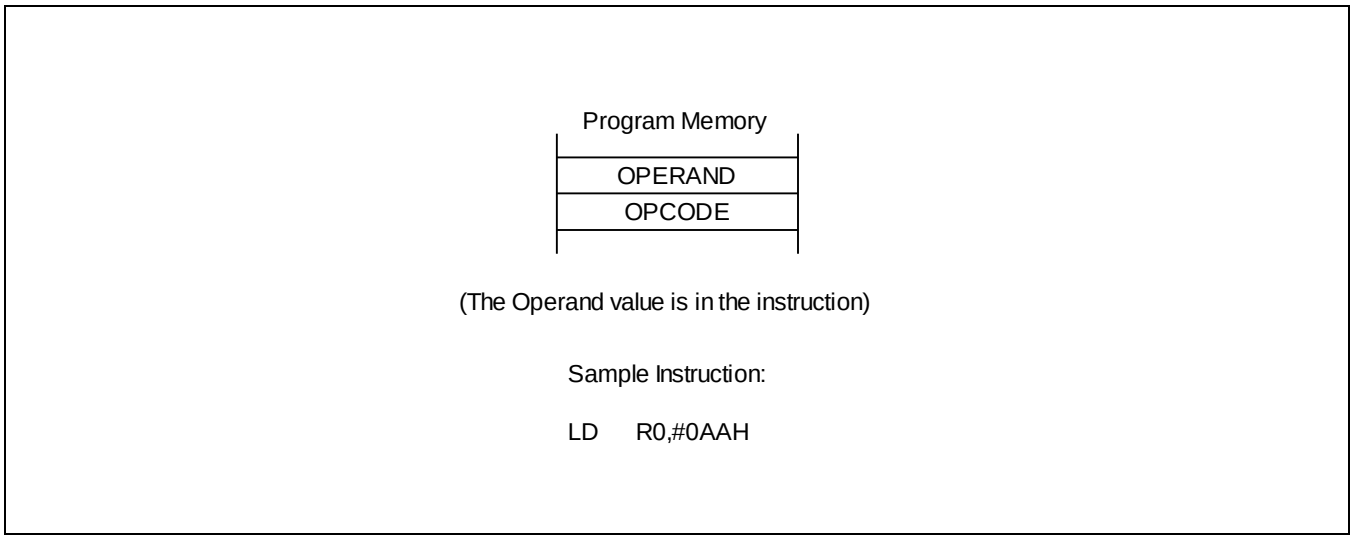


Figure 3-14 Immediate Addressing

4 Control Registers

4.1 Overview

This section describes the S3F8S39/S3F8S35 control registers in an easy-to-read format. These descriptions help you to get familiarize with the mapped locations in the register file. You can also use them as a quick-reference source when writing application programs.

Table 4-1 lists the system and peripheral registers. *Figure 4-1* illustrates the important features of the standard register description format.

Control register descriptions are alphabetically arranged according to register mnemonic. You can refer to Part II of this manual for more information about control registers.

Table 4-1 lists the system and peripheral control registers set 1.

Table 4-1 System and Peripheral Control Registers Set 1

Register Name	Mnemonic	Address		R/W	nRESET Value (Bit)							
		Decimal	Hex		7	6	5	4	3	2	1	0
A/D Converter Data register (High Byte)	ADDATAH	208	D0H	R	x	x	x	x	x	x	x	x
A/D Converter Data register (Low Byte)	ADDATAL	209	D1H	R	–	–	–	–	–	–	x	x
A/D Converter Control register	ADCON	210	D2H	R/W	0	0	0	0	0	0	0	0
Basic Timer Control register	BTCON	211	D3H	R/W	0	0	0	0	0	0	0	0
System Clock Control register	CLKCON	212	D4H	R/W	0	–	–	0	0	–	–	–
System Flags register	FLAGS	213	D5H	R/W	x	x	x	x	x	x	0	0
Register Pointer 0	RP0	214	D6H	R/W	1	1	0	0	0	–	–	–
Register Pointer 1	RP1	215	D7H	R/W	1	1	0	0	1	–	–	–
Stack Pointer (High Byte)	SPH	216	D8H	R/W	x	x	x	x	x	x	x	x
Stack Pointer (Low Byte)	SPL	217	D9H	R/W	x	x	x	x	x	x	x	x
Instruction Pointer (High Byte)	IPH	218	DAH	R/W	x	x	x	x	x	x	x	x
Instruction Pointer (Low Byte)	IPL	219	DBH	R/W	x	x	x	x	x	x	x	x
Interrupt Request register	IRQ	220	DCH	R	0	0	0	0	0	0	0	0
Interrupt Mask register	IMR	221	DDH	R/W	x	x	x	x	x	x	x	x
System Mode register	SYM	222	DEH	R/W	–	–	–	x	x	x	0	0
Register Page pointer	PP	223	DFH	R/W	0	0	0	0	0	0	0	0

NOTE:

1. An "x" means that the bit value is undefined following reset.
2. A dash ("–") refers to the bit that is neither used nor mapped. However, it reads the bit as "0".

Table 4-2 lists the system and peripheral control registers Set1 Bank 0.

Table 4-2 System and Peripheral Control Registers Set1 Bank 0

Register Name	Mnemonic	Address		R/W	nRESET Value (Bit)							
		Decimal	Hex		7	6	5	4	3	2	1	0
Timer A Counter register	TACNT	224	E0H	R	0	0	0	0	0	0	0	0
Timer A Data register	TADATA	225	E1H	R/W	1	1	1	1	1	1	1	1
Timer A Control register	TACON	226	E2H	R/W	0	0	0	0	0	0	0	0
Timer A Prescaler	TAPS	227	E3H	R/W	0	–	–	–	0	0	0	0
Location E4H to E7H is not mapped												
Timer 0 Counter register (High Byte)	T0CNTH	232	E8H	R	0	0	0	0	0	0	0	0
Timer 0 Counter register (Low Byte)	T0CNTL	233	E9H	R	0	0	0	0	0	0	0	0
Timer 0 Data register (High Byte)	T0DATAH	234	EAH	R/W	1	1	1	1	1	1	1	1
Timer 0 Data register (Low Byte)	T0DATAL	235	EBH	R/W	1	1	1	1	1	1	1	1
Timer 0 Control register	T0CON	236	ECH	R/W	0	0	0	0	0	0	0	0
Timer 0 Prescaler register	T0PS	237	EDH	R/W	–	–	–	–	0	0	0	0
Timer 1 Counter register (High Byte)	T1CNTH	238	EEH	R	0	0	0	0	0	0	0	0
Timer 1 Counter register (Low Byte)	T1CNTL	239	EFH	R	0	0	0	0	0	0	0	0
Timer 1 Data register (High Byte)	T1DATAH	240	F0H	R/W	1	1	1	1	1	1	1	1
Timer 1 Data register (Low Byte)	T1DATAL	241	F1H	R/W	1	1	1	1	1	1	1	1
Timer 1 Control register	T1CON	242	F2H	R/W	0	0	0	0	0	0	0	0
Timer 1 Prescaler register	T1PS	243	F3H	R/W	0	–	–	–	0	0	0	0
Timer 2 Counter register (High Byte)	T2CNTH	244	F4H	R	0	0	0	0	0	0	0	0
Timer 2 Counter register (Low Byte)	T2CNTL	245	F5H	R	0	0	0	0	0	0	0	0
Timer 2 Data register (High Byte)	T2DATAH	246	F6H	R/W	1	1	1	1	1	1	1	1
Timer 2 Data register (Low Byte)	T2DATAL	247	F7H	R/W	1	1	1	1	1	1	1	1
Timer 2 Control register	T2CON	248	F8H	R/W	0	0	0	0	0	0	0	0
Timer 2 Prescaler register	T2PS	249	F9H	R/W	0	–	–	–	0	0	0	0
Stop Wake-Up Timer Control register	SWTCON	250	FAH	R/W	0	–	0	–	0	0	0	0
Ring oscillator Control register	ROSCCON	251	FBH	R/W	0	–	–	–	–	–	–	–
Location FCH is not mapped												
Basic Timer Counter register	BTCNT	253	FDH	R	0	0	0	0	0	0	0	0
Location FEH is not mapped												
Interrupt Priority register	IPR	255	FFH	R/W	x	x	x	x	x	x	x	x

Table 4-3 lists the system and peripheral control registers Set1 Bank 1.

Table 4-3 System and Peripheral Control Registers Set1 Bank 1

Register Name	Mnemonic	Address		R/W	nRESET Value (Bit)							
		Decimal	Hex		7	6	5	4	3	2	1	0
Port 0 Control register (High Byte)	P0CONH	224	E0H	R/W	0	0	0	0	0	0	0	0
Port 0 Control register (Low Byte)	P0CONL	225	E1H	R/W	0	0	0	0	0	0	0	0
Port 0 Pull-up Resistor Enable register	P0PUR	226	E2H	R/W	0	0	0	0	0	0	0	0
Port 1 Control register (Low Byte)	P1CONL	227	E3H	R/W	–	–	–	–	0	0	0	0
Port 2 Control register (High Byte)	P2CONH	228	E4H	R/W	0	0	0	0	0	0	0	0
Port 2 Control register (Low Byte)	P2CONL	229	E5H	R/W	0	0	0	0	0	0	0	0
Port 2 Interrupt Control register	P2INT	230	E6H	R/W	0	0	0	0	0	0	0	0
Port 2 Pull-up Resistor Enable register	P2PUR	231	E7H	R/W	0	0	0	0	0	0	0	0
Port 3 Control Register (High Byte)	P3CONH	232	E8H	R/W	0	0	0	0	0	0	0	0
Port 3 Control Register (Low Byte)	P3CONL	233	E9H	R/W	0	0	0	0	0	0	0	0
Port 3 Interrupt Control register	P3INT	234	EAH	R/W	0	0	0	0	0	0	0	0
Port 3 Pull-up Resistor Enable register	P3PUR	235	EBH	R/W	0	0	0	0	0	0	0	0
Port 3 N-Channel Open-Drain mode register	PNE3	236	ECH	R/W	0	0	0	0	0	0	0	0
Port 0 Data register	P0	237	EDH	R/W	0	0	0	0	0	0	0	0
Port 1 Data register	P1	238	EEH	R/W	–	–	–	–	–	–	0	0
Port 2 Data register	P2	239	EFH	R/W	0	0	0	0	0	0	0	0
Port 3 Data register	P3	240	F0H	R/W	0	0	0	0	0	0	0	0
External Interrupt Pending register	PINTPND	241	F1H	R/W	0	0	0	0	0	0	0	0
SPI Control register	SPICON	242	F2H	R/W	0	0	0	0	0	0	0	0
SPI Status register	SPISTAT	243	F3H	R/W	0	0	0	–	–	–	–	0
SPI Data register	SPIDATA	244	F4H	R/W	1	1	1	1	1	1	1	1
UART0 Control register (High Byte)	UART0CONH	245	F5H	R/W	0	0	0	0	0	0	0	0
UART0 Control register (Low Byte)	UART0CONL	246	F6H	R/W	0	0	0	0	0	0	0	0
UART0 Data register	UDATA0	247	F7H	R/W	x	x	x	x	x	x	x	x
UART0 Baud Rate Data register	BRDATA0	248	F8H	R/W	1	1	1	1	1	1	1	1
UART1 Control register	UART1CONH	249	F9H	R/W	0	0	0	0	0	0	0	0

Register Name	Mnemonic	Address		R/W	nRESET Value (Bit)							
		Decimal	Hex		7	6	5	4	3	2	1	0
(High Byte)												
UART1 Control register (Low Byte)	UART1CONL	250	FAH	R/W	0	0	0	0	0	0	0	0
UART1 Data register	UDATA1	251	FBH	R/W	x	x	x	x	x	x	x	x
UART1 Baud Rate Data register	BRDATA1	252	FCH	R/W	1	1	1	1	1	1	1	1
STOP Control register	STPCON	253	FDH	R/W	0	0	0	0	0	0	0	0
Watch Timer Control register	WTCON	254	FEH	R/W	–	0	0	0	0	0	0	0
LVD Control register	LVDCON	255	FFH	R/W	1	–	0	0	0	–	0	0

Table 4-4 lists the system and peripheral control registers Page 8.

Table 4-4 System and Peripheral Control Registers Page 8

Register Name	Mnemonic	Address		R/W	nRESET Value (Bit)							
		Decimal	Hex		7	6	5	4	3	2	1	0
Reset ID register	RESETID	0	00H	R/W	Refer to detail description							
Flash Memory Sector Address register (High Byte)	FMSECH	1	01H	R/W	0	0	0	0	0	0	0	0
Flash Memory Sector Address register (Low Byte)	FMSECL	2	02H	R/W	0	0	0	0	0	0	0	0
Flash Memory User Programming enable register	FMUSR	3	03H	R/W	0	0	0	0	0	0	0	0
Flash Memory Control register	FMCON	4	04H	R/W	0	0	0	0	0	–	–	0
IIC Control register	ICCR	5	05H	R/W	0	0	0	0	1	1	1	1
IIC Status register	ICSR	6	06H	R/W	0	0	0	0	0	0	0	0
IIC Data Shift register	IDSR	7	07H	R/W	x	x	x	x	x	x	x	x
IIC Address register	IAR	8	08H	R/W	x	x	x	x	x	x	x	x
16-bit Timer B Control register	TBCON	9	09H	R/W	0	0	0	0	0	0	0	0
Timer B Reference Data register 0 (High Byte)	TBDATA0H	10	0AH	R/W	1	1	1	1	1	1	1	1
Timer B Reference Data register 0 (Low Byte)	TBDATA0L	11	0BH	R/W	1	1	1	1	1	1	1	1
Timer B Reference Data register 1 (High Byte)	TBDATA1H	12	0CH	R/W	1	1	1	1	1	1	1	1
Timer B Reference Data register 1 (Low Byte)	TBDATA1L	13	0DH	R/W	1	1	1	1	1	1	1	1
Timer B Trigger Control register	TBTRG	14	0EH	R/W	0	0	0	0	0	0	0	0
Location 0FH to 16H are not mapped												
Internal oscillator Calibration Data register	OSCCALDATA	23	17H	R/W	–	Refer to (NOTE)						
Internal oscillator Calibration Control register	OSCCALCON	24	18H	R/W	0	0	0	0	0	0	0	0

NOTE: The reset value of OSCCALDATA register is factory calibrated. Different chips may have different reset value.

Figure 4-1 illustrates the register description format.

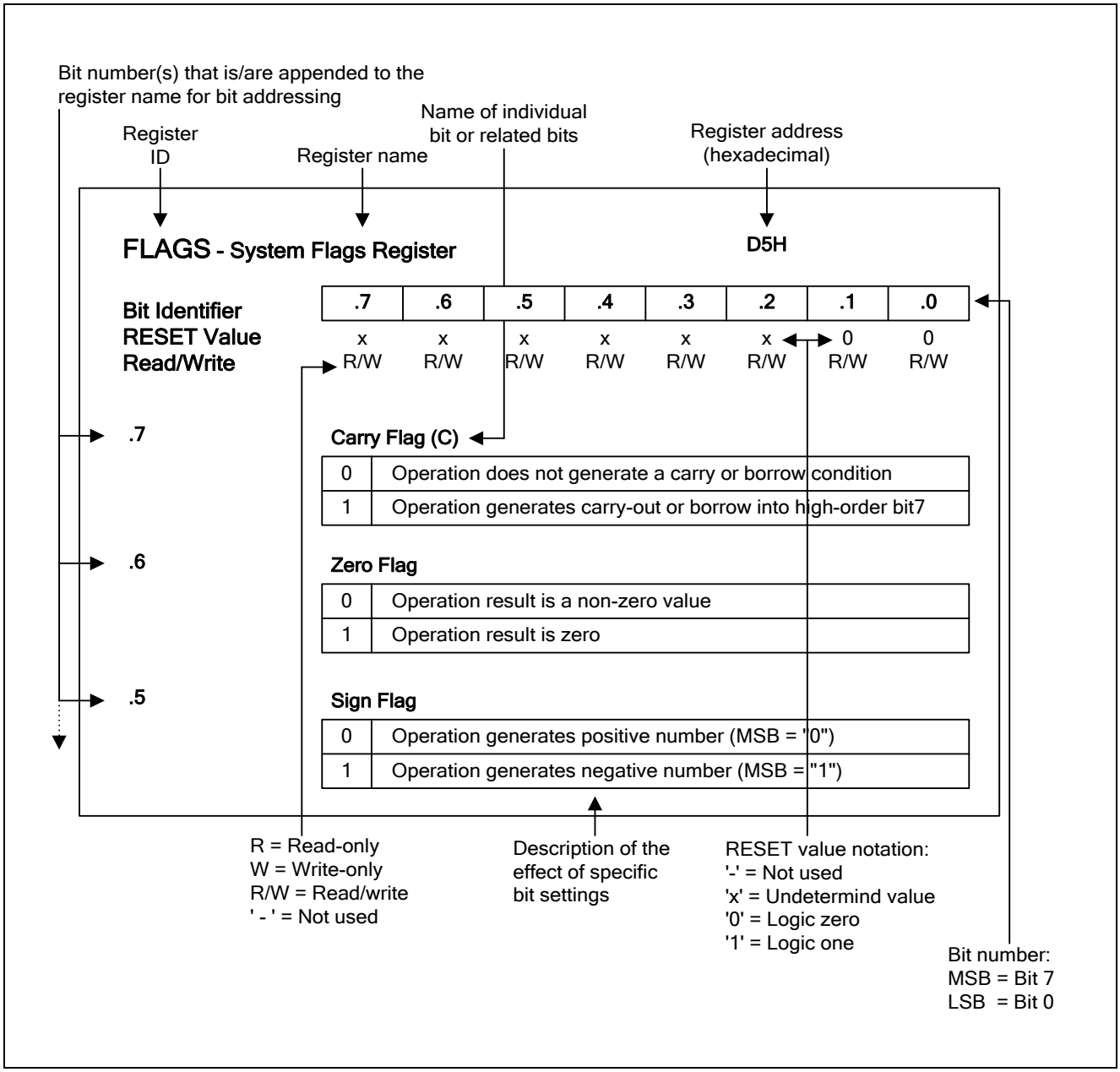


Figure 4-1 Register Description Format

4.1.1 ADCON-A/D Converter Control Register (D2H, SET1)

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

.7– .4	A/D Converter Input Selection Bits				
	0	0	0	0	Select AD0
	0	0	0	1	Select AD1
	0	0	1	0	Select AD2
	0	0	1	1	Select AD3
	0	1	0	0	Select AD4
	0	1	0	1	Select AD5
	0	1	1	0	Select AD6
	0	1	1	1	Select AD7
	1	0	0	0	Select AD8
	1	0	0	1	Select AD9
	1	0	1	0	Select AD10
	1	0	1	1	Select AD11
	1	1	0	0	Select AD12
	1	1	0	1	Select AD13
	1	1	1	0	Select AD14
	1	1	1	1	Select AD15

.3	End-of-conversion Bit (Read-only)	
	0	Conversion not completed
	1	Conversion completed

.2– .1	A/D Converter Clock Selection Bits		
	0	0	fxx/16
	0	1	fxx/8
	1	0	fxx/4
	1	1	fxx/1 (fxx < 4 MHz)

.0	A/D Converter Start or Disable Bit	
	0	Disable operation
	1	Start operation

NOTE: Maximum ADC clock input = 4 MHz.

4.1.2 BTCON-Basic Timer Control Register (D3H, SET1)

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

.7– .4	**Watchdog Timer Function Disable Code (For System Reset)**				
	1	0	1	0	Disable watchdog timer function
	Other values				Enable watchdog timer function
.3– .2	**Basic Timer Clock Selection Bits**				
	0	0	fxx/4096		
	0	1	fxx/1024		
	1	0	fxx/128		
	1	1	Invalid selection		
.1	**Basic Timer Counter Clear Bit**				
	0	No effect			
	1	Clears the basic timer counter value (Automatically cleared to "0" after being cleared basic timer counter)			
.0	**Clock Frequency Divider Clear Bit for Basic Timer and Timer Counters**				
	0	No effect			
	1	Clear clock frequency dividers (Automatically cleared to "0" after being cleared clock frequency dividers)			

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

4.1.3 CLKCON-Clock Control Register (D4H, SET1)

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
nRESET Value	0	–	–	0	0	–	–	–
Read/Write	R/W	–	–	R/W	R/W	–	–	–
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**	
.4– .3	**CPU Clock (System Clock) Selection Bits** (NOTE)	
	0 0	fxx/16
	0 1	fxx/8
	1 0	fxx/2
	1 1	fxx
.2– .0	**Not used**	

NOTE: After a reset, it selects the slowest clock (divided by 16) 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, SET1)

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

.7	Carry Flag (C)	
	0	Operation does not generate a carry or borrow condition
	1	Operation generates a carry-out or borrow into high-order bit 7

.6	Zero Flag (Z)	
	0	Operation result is a non-zero value
	1	Operation result is zero

.5	Sign Flag (S)	
	0	Operation generates a positive number (MSB = "0")
	1	Operation generates a negative number (MSB = "1")

.4	Overflow Flag (V)	
	0	Operation result is ≤ +127 and ≥ 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 (04H, PAGE8)

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
Reset Value	0	0	0	0	0	–	–	0
Read/Write	R/W	R/W	R/W	R/W	R	–	–	R/W
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
	Other values				Not available

.3	Sector Erase Status Bit	
	0	Success sector erase
	1	Fail sector erase

.2– .1	Not used

.0	Flash Operation Start Bit	
	0	Operation stop
	1	Operation start (This bit will be cleared automatically just after the corresponding operator completed).

4.1.6 FMSECH-Flash Memory Sector Address Register (High Byte) (01H, PAGE8)

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
Reset Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							
.7– .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 the higher eight bits of the 16-bit pointer address.

4.1.7 FMSECL-Flash Memory Sector Address Register (Low Byte) (02H, PAGE8)

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
Reset Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							
.7	Flash Memory Sector Address Bit (Low Byte)							
	The 7th bit to select a sector of flash ROM							
.6– .0	Bits 6-0							
	Don't care							

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

4.1.8 FMUSR-Flash Memory User Programming Enable Register (03H, PAGE8)

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
Reset Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							
.7– .0	Flash Memory User Programming Enable Bits							
	1 0 1 0 0 1 0 1	Enable user programming mode						
	Other values	Disable user programming mode						

4.1.9 ICCR-Multi-Master IIC-Bus Clock Control Register (05H, PAGE8)

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
nRESET Value	0	0	0	0	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							
.7	**Acknowledgement Enable Bit**							
	0	Acknowledgement disable mode						
	1	Acknowledgement enable mode						
.6	**Tx Clock Selection Bit**							
	0	fosc/16						
	1	fosc/512						
.5	**Multi-master IIC-Bus Tx/Rx Interrupt Enable Bit**							
	0	Disable						
	1	Enable						
.4	**IIC Interrupt Pending Bit**							
	0	Interrupt request is not pending (when read) ; pending bit clear when write 0						
	1	Interrupt request is pending (when read)						
.3– .0	**ICCR.3-0: Transmit Clock 4-Bit Prescaler Bits**							
	SCL clock = IICLK/(CCR < 3:0 > + 1) where, IICLK = fosc/16 when IICR.6 is "0", IICLK = fosc/512 when ICCR.6 is "1"							

4.1.10 ICCR-Multi-Master IIC-Bus Clock Control Register (05H, PAGE8)

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
nRESET Value	0	0	0	0	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							
.7	**Acknowledgement Enable Bit**							
	0	Acknowledgement disable mode						
	1	Acknowledgement enable mode						
.6	**Tx Clock Selection Bit**							
	0	fosc/16						
	1	fosc/512						
.5	**Multi-master IIC-Bus Tx/Rx Interrupt Enable Bit**							
	0	Disable						
	1	Enable						
.4	**IIC Interrupt Pending Bit**							
	0	Interrupt request is not pending; (when read) pending bit clear when write 0						
	1	Interrupt request is pending (when read)						
.3– .0	**ICCR.3-0: Transmit Clock 4-Bit Prescaler Bits**							
	SCL clock = IICLK/(CCR < 3:0 > + 1) where, IICLK = fosc/16 when IICR.6 is "0", IICLK = fosc/512 when ICCR.6 is "1"							

4.1.11 IMR-Interrupt Mask Register (DDH, SET1)

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

.7	**Interrupt Level 7 (IRQ7)**	
	0	Disable (Mask)
	1	Enable (Unmask)
.6	**Interrupt Level 6 (IRQ6)**	
	0	Disable (Mask)
	1	Enable (Unmask)
.5	**Interrupt Level 5 (IRQ5)**	
	0	Disable (Mask)
	1	Enable (Unmask)
.4	**Interrupt Level 4 (IRQ4)**	
	0	Disable (Mask)
	1	Enable (Unmask)
.3	**Interrupt Level 3 (IRQ3)**	
	0	Disable (Mask)
	1	Enable (Unmask)
.2	**Interrupt Level 2 (IRQ2)**	
	0	Disable (Mask)
	1	Enable (Unmask)
.1	**Interrupt Level 1 (IRQ1)**	
	0	Disable (Mask)
	1	Enable (Unmask)
.0	**Interrupt Level 0 (IRQ0)**	
	0	Disable (Mask)
	1	Enable (Unmask)

NOTE: When an interrupt level is masked, the CPU does not recognize any interrupt requests that may be issued.

4.1.12 IPH-Instruction Pointer (High Byte) (DAH, SET1)

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

4.1.13 IPL-Instruction Pointer (Low Byte) (DBH, SET1)

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

4.1.14 IPR-Interrupt Priority Register (FFH, BANK0)

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

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

1. Interrupt Group A-IRQ0, IRQ1
2. Interrupt Group B-IRQ2, IRQ3, IRQ4
3. Interrupt Group C-IRQ5, IRQ6, IRQ7

4.1.15 IRQ-Interrupt Request Register (DCH, SET1)

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

.7	**Level 7 (IRQ7) Request Pending Bit;**	
	0	Not pending
	1	Pending
.6	**Level 6 (IRQ6) Request Pending Bit;**	
	0	Not pending
	1	Pending
.5	**Level 5 (IRQ5) Request Pending Bit;**	
	0	Not pending
	1	Pending
.4	**Level 4 (IRQ4) Request Pending Bit;**	
	0	Not pending
	1	Pending
.3	**Level 3 (IRQ3) Request Pending Bit;**	
	0	Not pending
	1	Pending
.2	**Level 2 (IRQ2) Request Pending Bit;**	
	0	Not pending
	1	Pending
.1	**Level 1 (IRQ1) Request Pending Bit;**	
	0	Not pending
	1	Pending
.0	**Level 0 (IRQ0) Request Pending Bit;**	
	0	Not pending
	1	Pending

4.1.16 LVDCON-Low Voltage Detector Control Register (FFH, BANK 1)

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

.7	**LVD Enable/Disable Bit**	
	0	Disable LVD
	1	Enable LVD
.6	**Not used**	
.5	**LVD Output Bit (Read Only)**	
	0	$V_{DD} > V_{LVD}$
	1	$V_{DD} < V_{LVD}$
.4	**LVD Interrupt Enable/Disable Bit**	
	0	Disable LVD interrupt
	1	Enable LVD interrupt
.3–.2	**Not used**	

.1– .0	**Detection Voltage Level Selection Bits**		
	0	0	V_{LVD} = 4.1 V
	0	1	V_{LVD} = 3.2 V
	1	0	V_{LVD} = 2.5 V
	1	1	V_{LVD} = 2.1 V

4.1.17 OSCCALCON-Internal oscillator Calibration Control Register (18H, PAGE8)

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

.7– .0	Internal oscillator Calibration Control Bits	
	10100101	Enable OSCCALDATA register write
	Other values	Disable OSCCALDATA register write

NOTE: Before writing OSCCALDATA register, set this OSCCALCON register as "10100101b".

4.1.18 P0CONH-Port 0 Control Register High Byte (E0H, BANK1)

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

.7– .6	**P0.7/ AD7 Configuration Bits**		
	0	0	Schmitt trigger input mode
	0	1	Not used
	1	0	Alternative function (AD7)
	1	1	Push-pull output mode
.5– .4	**P0.6/ AD6 Configuration Bits**		
	0	0	Schmitt trigger input mode
	0	1	Not used
	1	0	Alternative function (AD6)
	1	1	Push-pull output mode
.3– .2	**P0.5/ AD5 Configuration Bits**		
	0	0	Schmitt trigger input mode
	0	1	Not used
	1	0	Alternative function (AD5)
	1	1	Push-pull output mode
.1– .0	**P0.4/ AD4 Configuration Bits**		
	0	0	Schmitt trigger input mode
	0	1	Not used
	1	0	Alternative function (AD4)
	1	1	Push-pull output mode

4.1.19 P0CONL-Port 0 Control Register Low Byte (E1H, BANK1)

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

.7– .6	**P0.3/AD3 Configuration Bits**		
	0	0	Schmitt trigger input mode
	0	1	Not used
	1	0	Alternative function (AD3)
	1	1	Push-pull output mode
.5– .4	**P0.2 /AD2 Configuration Bits**		
	0	0	Schmitt trigger input mode
	0	1	Not used
	1	0	Alternative function (AD2)
	1	1	Push-pull output mode
.3– .2	**P0.1 /AD1 Configuration Bits**		
	0	0	Schmitt trigger input mode
	0	1	Not used
	1	0	Alternative function (AD1)
	1	1	Push-pull output mode
.1– .0	**P0.0/AD0 Configuration Bits**		
	0	0	Schmitt trigger input mode
	0	1	Alternative function (BUZ)
	1	0	Alternative function (AD0)
	1	1	Push-pull output mode

4.1.20 P0PUR-Port 0 Pull Up Resistor Enable Register (E2H, BANK1)

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

.7	**P0.7's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.6	**P0.6's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.5	**P0.5's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.4	**P0.4's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.3	**P0.3's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.2	**P0.2's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.1	**P0.1's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.0	**P0.0's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor

NOTE: A pull-up resistor of port 0 is automatically disabled only when it selects the corresponding pin as push-pull output or alternative function

4.1.21 P1CONL-Port 1 Control Register Low Byte (E3H, BANK1)

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

.7– .4	**Not used**		
.3– .2	**P1.1/ADC9 Configuration Bits**		
	0	0	Schmitt trigger input mode
	0	1	Schmitt trigger input mode; pull-up
	1	0	Alternative function (AD9)
	1	1	Push-pull output mode
.1– .0	**P1.0/ADC8 Configuration Bits**		
	0	0	Schmitt trigger input mode
	0	1	Schmitt trigger input mode; pull-up
	1	0	Alternative function (AD8)
	1	1	Push-pull output mode

4.1.22 P2CONH-Port 2 Control Register High Byte (E4H, BANK1)

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

.7– .6	**P2.7/RXD0 Configuration Bits**		
	0	0	Schmitt trigger input mode (RXD0)
	0	1	Alternative function (RXD0 out)
	1	0	Open-drain output mode
	1	1	Push-pull output mode
.5– .4	**P2.6/TXD0 Configuration Bits**		
	0	0	Schmitt trigger input mode
	0	1	Alternative function (TXD0)
	1	0	Open-drain output mode
	1	1	Push-pull output mode
.3– .2	**P2.5/RXD1 Configuration Bits**		
	0	0	Schmitt trigger input mode (RXD1)
	0	1	Alternative function (RXD1 out)
	1	0	Open-drain output mode
	1	1	Push-pull output mode
.1– .0	**P2.4/TXD1 Configuration Bits**		
	0	0	Schmitt trigger input mode
	0	1	Alternative function (TXD1)
	1	0	Open-drain output mode
	1	1	Push-pull output mode

4.1.23 P2CONL-Port 2 Control Register Low Byte (E5H, BANK1)

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

.7– .6	P2.3/TAOUT/TACAP Configuration Bits		
	0	0	Schmitt trigger input mode (TACAP)
	0	1	Alternative function (TAOUT)
	1	0	Open-drain output mode
	1	1	Push-pull output mode
.5– .4	P2.2/TACLK Configuration Bits		
	0	0	Schmitt trigger input mode (TACLK)
	0	1	Not available
	1	0	Open-drain output mode
	1	1	Push-pull output mode
.3– .2	P2.1/TBOUT Configuration Bits		
	0	0	Schmitt trigger input mode
	0	1	Alternative function (TBOUT)
	1	0	Open-drain output mode
	1	1	Push-pull output mode
.1– .0	P2.0/T0OUT Configuration Bits		
	0	0	Schmitt trigger input mode
	0	1	Alternative function (T0OUT)
	1	0	Open-drain output mode
	1	1	Push-pull output mode

4.1.24 P2INT-Port 2 Interrupt Control Register (E6H, BANK1)

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

.7– .6	**P2.3 (INT3) External Interrupt Configuration Bit**		
	0	0	Disable interrupt
	0	1	Enable interrupt; interrupt on falling edge
	1	0	Enable interrupt; interrupt on rising edge
	1	1	Enable interrupt; interrupt on rising or falling edge
.5– .4	**P2.2 (INT2) External Interrupt Configuration Bit**		
	0	0	Disable interrupt
	0	1	Enable interrupt; interrupt on falling edge
	1	0	Enable interrupt; interrupt on rising edge
	1	1	Enable interrupt; interrupt on rising or falling edge
.3– .2	**P2.1 (INT1) External Interrupt Configuration Bit**		
	0	0	Disable interrupt
	0	1	Enable interrupt; interrupt on falling edge
	1	0	Enable interrupt; interrupt on rising edge
	1	1	Enable interrupt; interrupt on rising or falling edge
.1– .0	**P2.0 (INT0) External Interrupt Configuration Bit**		
	0	0	Disable interrupt
	0	1	Enable interrupt; interrupt on falling edge
	1	0	Enable interrupt; interrupt on rising edge
	1	1	Enable interrupt; interrupt on rising or falling edge

4.1.25 P2PUR-Port 2 Pull-Up Resistor Enable Register (E7H, BANK1)

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

.7	**P2.7's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.6	**P2.6's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.5	**P2.5's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.4	**P2.4's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.3	**P2.3's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.2	**P2.2's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.1	**P2.1's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.0	**P2.0's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor

NOTE: A pull-up resistor of port 2 is automatically disabled only when the corresponding pin is selected as push-pull output or alternative function

4.1.26 P3CONH-Port 3 Control Register High Byte (E8H, BANK1)

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

.7– .6	**P3.7/INT7/T1CAP/SDA Configuration Bits**		
	0	0	Schmitt trigger input mode (T1CAP)
	0	1	Alternative function (IIC signal SDA) NOTE
	1	0	Not used
	1	1	Output mode
.5– .4	**P3.6/INT6/SCL/T2CLK Configuration Bits**		
	0	0	Schmitt trigger input mode (T1CLK)
	0	1	Alternative function (IIC signal SCL)NOTE
	1	0	Not used
	1	1	Output mode
.3– .2	**P3.5/INT5/T2CLK/T1OUT/ADC15 Configuration Bits**		
	0	0	Schmitt trigger input mode (T2CLK)
	0	1	Alternative function (T1OUT)
	1	0	Alternative function (AD15)
	1	1	Output mode
.1– .0	**P3.4/INT4/T2OUT/T2CAP/ADC14 Configuration Bits**		
	0	0	Schmitt trigger input mode (T2CAP)
	0	1	Alternative function (T2OUT)
	1	0	Alternative function (AD14)
	1	1	Output mode

NOTE: If you want to you use IIC open drain type, you should set PNE3.7 and PNE3.6 to "1".

4.1.27 P3CONL-Port 3 Control Register Low Byte (E9H, BANK1)

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

.7– .6	P3.3/NSS/ADC13 Configuration Bits		
	0	0	Schmitt trigger input mode
	0	1	Alternative function (NSS)
	1	0	Alternative function (AD13)
	1	1	Output mode
.5– .4	P3.2/SCK/ADC12 Configuration Bits		
	0	0	Schmitt trigger input mode
	0	1	Alternative function (SCK)
	1	0	Alternative function (AD12)
	1	1	Output mode
.3– .2	P3.1/MOSI/ADC11 Configuration Bits		
	0	0	Schmitt trigger input mode
	0	1	Alternative function (MOSI)
	1	0	Alternative function (AD11)
	1	1	Output mode
.1– .0	P3.0/MISO/ADC10 Configuration Bits		
	0	0	Schmitt trigger input mode
	0	1	Alternative function (MISO)
	1	0	Alternative function (AD10)
	1	1	Output mode

4.1.28 P3INT-Port 3 Interrupt Control Register (EAH, BANK1)

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

.7– .6	P3.7 (INT7) External Interrupt Configuration Bit		
	0	0	Disable interrupt
	0	1	Enable interrupt; interrupt on falling edge
	1	0	Enable interrupt; interrupt on rising edge
	1	1	Enable interrupt; interrupt on rising or falling edge
.5– .4	P3.6 (INT6) External Interrupt Configuration Bit		
	0	0	Disable interrupt
	0	1	Enable interrupt; interrupt on falling edge
	1	0	Enable interrupt; interrupt on rising edge
	1	1	Enable interrupt; interrupt on rising or falling edge
.3– .2	P3.5 (INT5) External Interrupt Configuration Bit		
	0	0	Disable interrupt
	0	1	Enable interrupt; interrupt on falling edge
	1	0	Enable interrupt; interrupt on rising edge
	1	1	Enable interrupt; interrupt on rising or falling edge
.1– .0	P3.4 (INT4) External Interrupt Configuration Bit		
	0	0	Disable interrupt
	0	1	Enable interrupt; interrupt on falling edge
	1	0	Enable interrupt; interrupt on rising edge
	1	1	Enable interrupt; interrupt on rising or falling edge

4.1.29 P3PUR-Port 3 Pull-Up Resistor Enable Register (EBH, BANK1)

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

.7	**P3.7's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.6	**P3.6's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.5	**P3.5's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.4	**P3.4's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.3	**P3.3's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.2	**P3.2's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.1	**P3.1's Pull-up Resistor Enable Bit**	
	0	Disable pull-up resistor
	1	Enable pull-up resistor
.0	**P3.0's Pull-up Resistor Enable Bit**	
	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.

4.1.30 PNE3-Port 3 N-Channel Open-Drain Mode Register (ECH, BANK1)

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

.7	P3.7's Output Mode Selection Bit	
	0	Push-pull output mode
	1	Open-drain output mode
.6	P3.6's Output Mode Selection Bit	
	0	Push-pull output mode
	1	Open-drain output mode
.5	P3.5's Output Mode Selection Bit	
	0	Push-pull output mode
	1	Open-drain output mode
.4	P3.4's Output Mode Selection Bit	
	0	Push-pull output mode
	1	Open-drain output mode
.3	P3.3's Output Mode Selection Bit	
	0	Push-pull output mode
	1	Open-drain output mode
.2	P3.2's Output Mode Selection Bit	
	0	Push-pull output mode
	1	Open-drain output mode
.1	P3.1's Output Mode Selection Bit	
	0	Push-pull output mode
	1	Open-drain output mode
.0	P3.0's Output Mode Selection Bit	
	0	Push-pull output mode
	1	Open-drain output mode

4.1.31 PINTPND-Interrupt Pending Register (F1H, BANK1)

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

.7	**P3.7 (INT7) External Interrupt Pending Bit**	
	0	Interrupt request is not pending; (when read) pending bit clear when write 0
	1	Interrupt request is pending (when read)
.6	**P3.6 (INT6) External Interrupt Pending Bit**	
	0	Interrupt request is not pending; (when read) pending bit clear when write 0
	1	Interrupt request is pending (when read)
.5	**P3.5 (INT5) External Interrupt Pending Bit**	
	0	Interrupt request is not pending; (when read) pending bit clear when write 0
	1	Interrupt request is pending (when read)
.4	**P3.4 (INT4) External Interrupt Pending Bit**	
	0	Interrupt request is not pending; (when read) pending bit clear when write 0
	1	Interrupt request is pending (when read)
.3	**P2.3 (INT3) External Interrupt Pending Bit**	
	0	Interrupt request is not pending; (when read) pending bit clear when write 0
	1	Interrupt request is pending (when read)
.2	**P2.2 (INT2) External Interrupt Pending Bit**	
	0	Interrupt request is not pending; (when read) pending bit clear when write 0
	1	Interrupt request is pending (when read)
.1	**P2.1 (INT1) External Interrupt Pending Bit**	
	0	Interrupt request is not pending; (when read) pending bit clear when write 0
	1	Interrupt request is pending (when read)
.0	**P2.0 (INT0) External Interrupt Pending Bit**	
	0	Interrupt request is not pending; (when read) pending bit clear when write 0
	1	Interrupt request is pending (when read)

4.1.32 PP-Register Page Pointer (DFH, SET1)

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

.7– .4	Destination Register Page Selection Bits				
	0	0	0	0	Destination: page 0
	0	0	0	1	Destination: page 1
	0	0	1	0	Destination: page 2
	0	0	1	1	Destination: page 3
	1	0	0	0	Destination: page 8
	Other values				**Not used**

.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
	1	0	0	0	Source: page 8
	Other values				**Not used**

NOTE: In the S3F8S39/S3F8S35 microcontroller, the internal register file is configured as 5 pages (Pages 0-3, 8). The pages 0-3 are used for general purpose register file.

4.1.33 RESETID-Reset Source Indicating Register (00H, PAGE8)

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
Read/Write	–	–	–	R/W	–	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							
.7– .3	Not used							
.2	nReset pin Indicating Bit							
	0	Reset is not generated by nReset pin (when read)						
	1	Reset is generated by nReset pin (when read)						
.1	Basic Timer Reset Indicating Bit							
	0	Reset is not generated by Basic Timer (when read)						
	1	Reset is generated by Basic Timer (when read)						
.0	LVR Indicating Bit							
	0	Reset is not generated by LVR (when read)						
	1	Reset is generated by LVR (when read)						

State of RESETID Depends on Reset Source								
	.7	.6	.5	.4	.3	.2	.1	.0
LVR	–	–	–	–	–	0	0	1
Basic Timer or nReset pin	–	–	–	–	–	(3)	(3)	(2)

NOTE:

1. To clear an indicating register, write a "0" to indicating flag bit; writing a "1" to a reset indicating flag (RESETID.0-2) has no effect.
2. When a LVR reset occurs, it sets RESETID.0 and it clears all the other bits to "0" simultaneously.
3. When a basic timer or nRESET pin reset occurs, the corresponding bit will be set, but all other indicating bits won't be changed.

4.1.34 ROSCCON-Ring Oscillator Control Register (FBH, BANK0)

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
nRESET Value	0	–	–	–	–	–	–	–
Read/Write	R/W	–	–	–	–	–	–	–
Addressing Mode	Register addressing mode only							
.7	Ring Oscillator Enable Bit							
	0	Disable ring oscillator						
	1	Enable ring oscillator						
.6– .0	Not used							

4.1.35 RP0-Register Pointer 0 (D6H, SET1)

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
nRESET Value	1	1	0	0	0	–	–	–
Read/Write	R/W	R/W	R/W	R/W	R/W	–	–	–
Addressing Mode	Register addressing only							
.7– .3	Register Pointer 0 Address Value							
–	Register pointer 0 can independently point to one of the 256 byte working register areas in the register file. By using the register pointers RP0 and RP1, you can select two 8 byte register slices at a time as active working register space. After a reset, RP0 points to address C0H in register set 1 by selecting the 8 byte working register slice C0H–C7H.							
.2– .0	Not used							

4.1.36 RP1-Register Pointer 1 (D7H, SET1)

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
nRESET Value	1	1	0	0	1	–	–	–
Read/Write	R/W	R/W	R/W	R/W	R/W	–	–	–
Addressing Mode	Register addressing only							
.7– .3	Register Pointer 1 Address Value							
–	Register pointer 1 can independently point to one of the 256 byte working register areas in the register file. By using the register pointers RP0 and RP1, you can select two 8 byte register slices at a time as active working register space. After a reset, RP1 points to address C8H in register set 1 by selecting the 8 byte working register slice C8H-CFH.							
.2– .0	Not used							

4.1.37 SPH-Stack Pointer High Byte (D8H, SET1)

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

4.1.38 SPL-Stack Pointer Low Byte (D9H, SET1)

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

4.1.39 SPICON-SPI Control Register (F2H, BANK1)

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

.7	**SPI Interrupt Enable/Disable Bit**	
	0	Disable SPI Interrupt
	1	Enable SPI Interrupt
.6	**SPI Enable Bit**	
	0	Disable SPI
	1	Enable SPI
.5	**Data Order Selection Bit**	
	0	Least Significant Bit (LSB) First
	1	Most Significant Bit (MSB) First
.4	**Master/Slave Mode Selection Bit**	
	0	Slave Mode
	1	Master Mode
.3	**Clock Polarity Bit**	
	0	Clock Low when Idle
	1	Clock High when Idle
.2	**Clock Phase Bit**	
	0	Sample on the leading edge of SPCK
	1	Sample on the trailing edge of SPCK

.1– .0	**SPCK Rate Selection Bit**		
	0	0	fosc/4
	0	1	fosc/16
	1	0	fosc/64
	1	1	fosc/256

4.1.40 SPISTAT-SPI Status Register (F3H, BANK1)

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

.7	**SPI Interrupt Pending Bit**	
	0	No pending
	1	Interrupt pending
.6	**Write Collision Bit**	
	0	No write collision
	1	Write collision
.5	**Mode Fault Bit**	
	0	No Mode fault
	1	Mode fault
.4– .1	**Not used**	
	0	Slave Mode
	1	Master Mode
.0	**Double SPI Speed Bit**	
	0	Single
	1	Double when in Master Mode

4.1.41 STPCON-Stop Mode Control Register (FDH, BANK1)

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

.7– .0	Watchdog Timer Function Enable Bit	
	10100101	Enable STOP instruction
	Other values	Disable STOP instruction

NOTE:

1. Before executing the STOP instruction, set this STPCON register as "10100101b"
2. When STOPCON register is not # 0A5H value, if you use STOP instruction, it changes PC to reset address.

4.1.42 SWTCON-Stop Wake-Up Timer Control Register (FAH, Bank 0)

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

.7	**Stop Wake-Up Timer Enable bit**	
	0	Disable Stop Wake-Up Timer
	1	Enable Stop Wake-Up Timer
.6	**Not used**	
.5	**Stop Wake-Up Timer Interrupt Enable bit**	
	0	Disable interrupt
	1	Enable interrupt
.4	**Not used**	
.3– .0	**Stop Wake-Up Timer prescaler bits (SWTPSB)**	
	SWTCLK = $F_{CLK}/(2^{SWTPSB[3-0]})$	

NOTE:

1. Prescaler values (SWTPSB) above 12 are NOT valid.
2. Bit 6 and .4 must kept as default value "0".

4.1.43 SYM-System Mode Register (DEH, SET1)

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

.7– .5	**Not used**			
.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" Enable fast interrupt processing for the interrupt level currently selected by SYM.2-SYM.4.
3. Following a reset, you should enable global interrupt processing by executing an EI instruction (not by writing a "1" to SYM.0).

4.1.44 TACON-Timer A Control Register (E2H, BANK0)

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

.7– .6	**Timer A Operating Mode Selection Bits**		
	0	0	Interval Mode (TAOUT mode)
	0	1	Capture Mode (Capture on rising edge, counter running, OVERFLOW can occur)
	1	0	Capture Mode (Capture on falling edge, counter running, OVERFLOW can occur)
	1	1	PWM Mode (OVERFLOW interrupt can occur)
.5	**Timer A Counter Clear Bit**		
	0		No effect
	1		Clears the Timer A counter (After clearing, return to zero)
.4	**Timer A Start/Stop Bit**		
	0		Stop Timer A
	1		Start Timer A
.3	**Timer A Match/Capture Interrupt Enable Bit**		
	0		Disable interrupt
	1		Enable interrupt
.2	**Timer A Overflow Interrupt Enable Bit**		
	0		Disable interrupt
	1		Enable interrupt
.1	**Timer A Match Interrupt Pending Bit**		
	0		No interrupt pending (clears pending bit when write)
	1		Interrupt pending
.0	**Timer A Overflow Interrupt Pending Bit**		
	0		No interrupt pending (clears pending bit when write)
	1		Interrupt pending

4.1.45 TAPS-TA PreScaler (E3H, BANK0)

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

.7	**Timer A Clock Source Selection**	
	0	Internal clock source
	1	External clock source from TACLK
.6– .5	**Not used**	
.3– .0	**Timer A Prescaler Bits**	
	TAPS = TA clock/($2^{TAPS[3-0]}$) Prescaler values above 12 are invalid	

4.1.46 TBCON-Timer B Control Register (09H, Page 8)

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

.7– .5	**Timer B Prescaler Bits**		
	TBPS = TB clock/($2^{TBPS[7-5]}$)		
.4– .3	**Timer B Interrupt Time Selection Bits**		
	0	0	Disable interrupt
	0	1	Generating after data0 is borrowed
	1	0	Generating after data1 is borrowed
	1	1	Generating after data0 and data1 are borrowed
.2	**Timer B Enable/Disable Bit**		
	0	Disable Timer B	
	1	Enable 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 (TBOUT: Low level for low data, high level for high data)	
	1	TBOF is high (TBOUT: High level for low data, low level for high data)	

4.1.47 TBTRG-Timer B Trigger Control Register (0EH, Page 8)

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

.7– .6	**Emergency Detection Pin Selection Bits: (Rising edge)**		
	0	0	Disable
	0	1	P30
	1	0	P31
	1	1	P32
.5– .4	**Trigger Source Selection Bits:**		
	0	0	P24
	0	1	P25
	1	0	P26
	1	1	P27
.3– .2	**Trigger Pin Valid Edge Selection Bits:**		
	0	0	Falling edge
	0	1	Rising edge
	1	x	Falling and rising edge
.1	**Timer B Trigger Control by H/W or S/W:**		
	0		Hardware control(external input trigger)
	1		Software control (internal trigger)
.0	**Timer B trigger control by software: (Write only, automatically cleared after write "1")**		
	0		Without one-shot pulse output trigger
	1		one-shot pulse is triggered

4.1.48 T0CON-Timer 0 Control Register (ECH, BANK0)

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

.7– .6	**Timer 0 Operating Mode Selection Bits**		
	0	0	Interval Mode (TAOUT mode)
	0	1	Not used
	1	0	Not used
	1	1	PWM Mode (OVF interrupt can occur)
.5	**Timer 0 Counter Clear Bit**		
	0		No effect
	1		Clears the timer 0 counter (after clearing, return to zero)
.4	**Timer 0 Start/Stop Bit**		
	0		Stop Timer 0
	1		Start Timer 0
.3	**Timer 0 Match Interrupt Enable Bit**		
	0		Disable interrupt
	1		Enable interrupt
.2	**Timer 0 Overflow Interrupt Enable Bit**		
	0		Disable interrupt
	1		Enable interrupt
.1	**Timer 0 Match Interrupt pending Bit**		
	0		No interrupt pending (clears pending bit when write)
	1		Interrupt pending
.0	**Timer 0 Overflow Interrupt pending Bit**		
	0		No interrupt pending (clears pending bit when write)
	1		Interrupt pending

4.1.49 T0PS-T0 PreScalar (EDH, BANK0)

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	–	–	–	0	0	0	0
Read/Write	–	–	–	–	R/W	R/W	R/W	R/W
.7– .5	Not used							
.3– .0	Timer 0 Prescaler Bits							
	T0PS = T0 clock/($2^{T0PS[3-0]}$) Prescaler values above 12 are invalid							

4.1.50 T1CON-Timer 1 Control Register (F2H, BANK0)

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

.7– .6	**Timer 1 Operating Mode Selection Bits**		
	0	0	Interval Mode (T1OUT mode)
	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 interrupt can occur)
.5	**Timer 1 Counter Clear Bit**		
	0		No effect
	1		Clears the timer 1 counter (after clearing, return to zero)
.4	**Timer 1 Start/Stop Bit**		
	0		Stop Timer 1
	1		Start Timer 1
.3	**Timer 1 Match/Capture Interrupt Enable Bit**		
	0		Disable interrupt
	1		Enable interrupt
.2	**Timer 1 Overflow Interrupt Enable Bit**		
	0		Disable interrupt
	1		Enable interrupt
.1	**Timer 1 Match Interrupt Pending Bit**		
	0		No interrupt pending (clears pending bit when write)
	1		Interrupt pending
.0	**Timer 1 Overflow Interrupt Pending Bit**		
	0		No interrupt pending (clears pending bit when write)
	1		Interrupt pending

4.1.51 T1PS-T1 PreScalar Register (F3H, BANK0)

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

.7	**Timer 1 Clock Source Selection**	
	0	Internal clock source
	1	External clock source from T1CLK
.6– .5	**Not used**	
.3– .0	**Timer 1 Prescaler Bits**	
	T1PS = T1 clock/ ($2^{T1PS[3-0]}$) Prescaler values above 12 are invalid	

4.1.52 T2CON-Timer 2 Control Register (F8H, BANK0)

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

.7– .6	**Timer 2 Operating Mode Selection Bits**		
	0	0	Interval Mode (T2OUT mode)
	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 interrupt can occur)
.5	**Timer 2 Counter Clear Bit**		
	0		No effect
	1		Clears the timer 2 counter (after clearing, return to zero)
.4	**Timer 2 Start/Stop Bit**		
	0		Stop Timer 2
	1		Start Timer 2
.3	**Timer 2 Match/Capture Interrupt Enable Bit**		
	0		Disable interrupt
	1		Enable interrupt
.2	**Timer 2 Overflow Interrupt Enable Bit**		
	0		Disable interrupt
	1		Enable interrupt
.1	**Timer 2 Match Interrupt Pending Bit**		
	0		No interrupt pending (clears pending bit when write)
	1		Interrupt pending
.0	**Timer 2 Overflow Interrupt Pending Bit**		
	0		No interrupt pending (clears pending bit when write)
	1		Interrupt pending

4.1.53 T2PS-T2 PreScalar Register (F9H, BANK0)

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

.7	**Timer 2 Clock Source Selection**	
	0	Internal clock source
	1	External clock source from T2CLK
.6– .5	**Not used**	
.3– .0	**Timer 2 Prescaler Bits**	
	T2PS = T2 clock/ ($2^{T2PS[3-0]}$) Prescaler values above 12 are invalid	

4.1.54 UART0CONH-UART 0 Control Register High Byte (F5H, BANK1)

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

.7– .6	**UART 0 Mode Selection Bits**		
	0	0	Mode 0: Shift register (fxx/(16 × (BR0DATA + 1)))
	0	1	Mode 1: 8-bit UART (fxx/(16 × (BR0DATA + 1)))
	1	0	Mode 2: 9-bit UART (fxx/16)
	1	1	Mode 3: 9-bit UART (fxx/(16 × (BR0DATA + 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 to be 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), clears pending bit (when write)	
	1	Interrupt is pending (When read)	

4.1.55 UART0CONL-UART 0 Control Register Low Byte (F6H, BANK1)

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

.7	**UART 0 Transmit Parity bit Autogeneration Enable Bit**		
	0	Disable parity bit autogeneration	
	1	Enable parity bit autogeneration	
.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), clears pending bit (when write)	
	1	Interrupt is pending (when read)	

4.1.56 UART1CONH-UART 1Control Register High Byte (F9H, BANK1)

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

.7– .6	**UART 1 Mode Selection Bits**		
	0	0	Mode 0: Shift register (fxx/(16 × (BR0DATA + 1)))
	0	1	Mode 1: 8-bit UART (fxx/(16 × (BR0DATA + 1)))
	1	0	Mode 2: 9-bit UART (fxx/16)
	1	1	Mode 3: 9-bit UART (fxx/(16 × (BR0DATA + 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 0 mode 2 or 3 ("0" or "1")		
	NOTE: If the UART0CONL.7 = 1, This bit is "don't care".		
.2	**RB8 (Only When UART1CONL.7 = 0)**		
	Location of the 9th data bit to be 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 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), clears pending bit (when write)	
	1	Interrupt is pending (when read)	

4.1.57 UART1CONL-UART 1 Control Register Low Byte (FAH, BANK1)

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

.7	**UART 1 Transmit Parity bit Autogeneration Enable Bit**		
	0	Disable parity bit autogeneration	
	1	Enable parity bit autogeneration	
.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), clears pending bit (when write)	
	1	Interrupt is pending (when read)	

4.1.58 WTCON-Watch Timer Control Register (FEH, BANK1)

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

.7	Not used		
.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	1.0 kHz
	1	0	2.0 kHz
	1	1	4.0 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 1.995ms
.1	Watch Timer Enable Bit		
	0	Disable watch timer; clear frequency dividing circuits	
	1	Enable watch timer	
.0	Watch Timer Interrupt Pending Bit		
	0	Interrupt request is not pending; (when read) pending bit clear when write 0	
	1	Interrupt request is pending (when read)	

5 Interrupt Structure

5.1 Overview

The S3C8/S3F8-series interrupt structure has three basic components:

- Levels
- Vectors
- Sources

The SAM8RC CPU recognizes up to eight interrupt levels and supports up to 128 interrupt vectors. When a specific interrupt level has multiple vector addresses, the vector priorities are established in hardware. A vector address can be assigned to one or more sources.

5.1.1 Levels

Interrupt levels are the main unit for interrupt priority assignment and recognition. All peripherals and I/O blocks can issue interrupt requests. That means, 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 that it uses in the interrupt structure varies from device to device. The S3F8S39/S3F8S35 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 settings in the interrupt priority register (IPR) determine the relative priority of different priority levels. The IPR settings that control interrupt group and subgroup logic let you define more complex priority relationships between different levels.

5.1.2 Vectors

Each interrupt level can have one or more interrupt vectors or it may have no vector address assigned. The maximum number of vectors that can be supported for a given level is 128 (The actual number of vectors it uses for S3C8/S3F8-series devices is always much smaller). If an interrupt level has multiple vector addresses, the vector priorities are set in hardware. The S3F8S39/S3F8S35 uses 26 vectors.

5.1.3 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 S3F8S39/S3F8S35 interrupt structure there are 25 possible interrupt sources. That means every source has its own vector.

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

5.2 Interrupt Types

The three components of the S3C8/S3F8 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 (Refer to *Figure 5-1* for more information)

- 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 S3F8S39/S3F8S35 microcontroller, two interrupt types are implemented.

Figure 5-1 illustrates the S3C8/S3F8-series interrupt types.

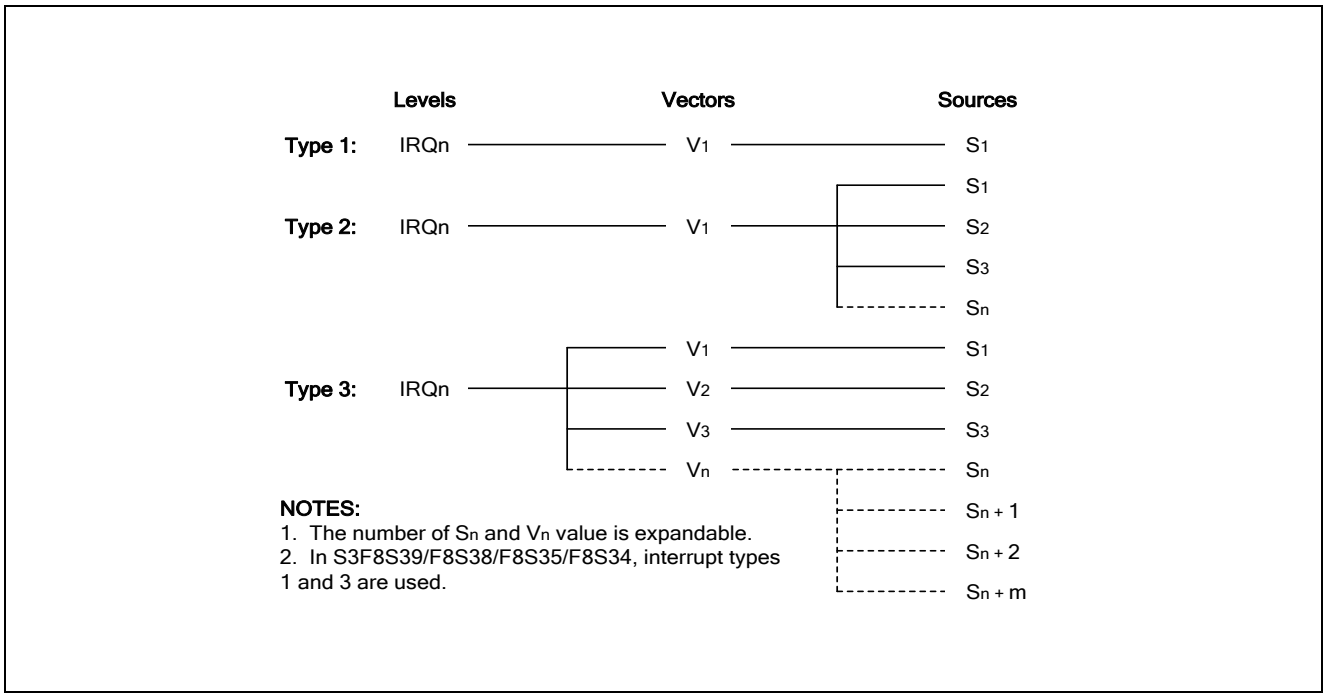


Figure 5-1 S3C8/S3F8 Series Interrupt Types

5.3 S3F8S39/S3F8S35 Interrupt Structure

The S3F8S39/S3F8S35 microcontroller supports 26 interrupt sources. Every interrupt source has a corresponding interrupt address. The CPU recognizes eight interrupt levels in this device-specific interrupt structure as illustrated in *Figure 5-2*.

When multiple interrupt levels are active, the 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.

Figure 5-2 illustrates the S3F8S39/S3F8S35 microcontroller interrupt structure.

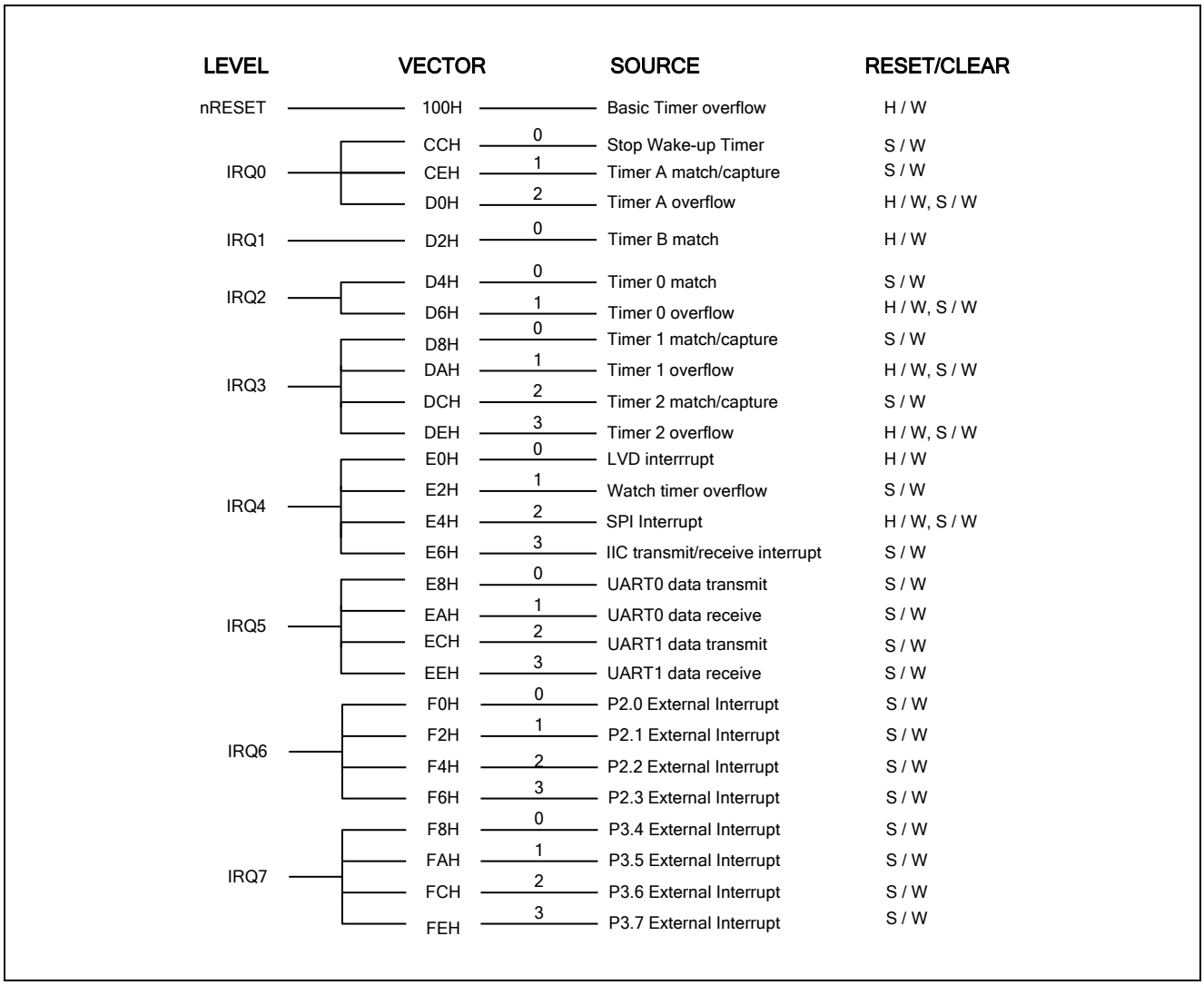


Figure 5-2 S3F8S39/S3F8S35 Interrupt Structure

5.3.1 Interrupt Vector Addresses

All interrupt vector addresses for the S3F8S39/S3F8S35 interrupt structure is stored in the vector address area of the first 256 bytes of the program memory (ROM).

You can allocate unused locations in the vector address area as normal program memory. If you do so, ensure not overwrite any of the stored vector addresses.

The default program reset address in the ROM is 0100H.

Figure 5-3 illustrates the ROM vector address area.

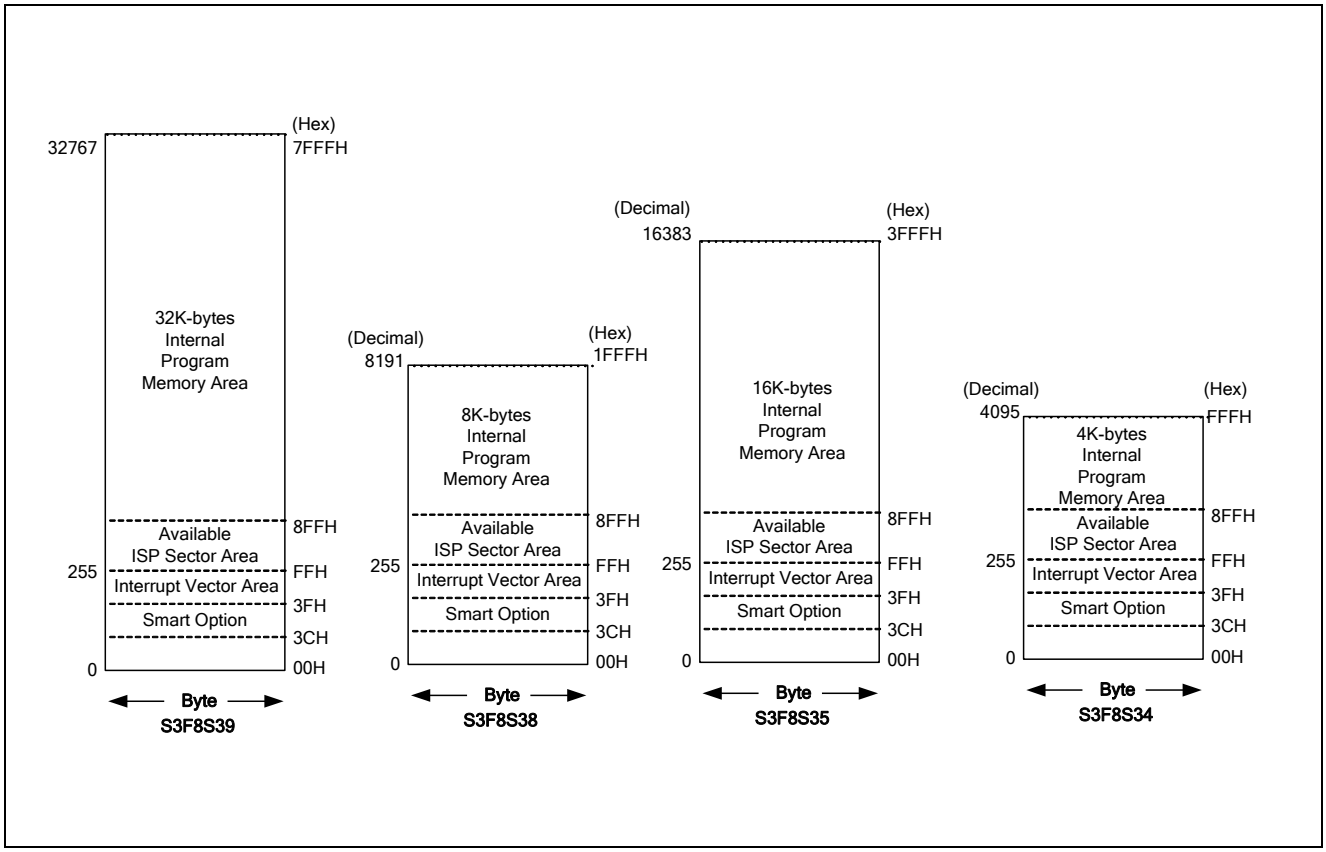


Figure 5-3 ROM Vector Address Area

5.3.2 Enable/Disable Interrupt Instructions

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 should always contain an EI instruction to globally enable the interrupt structure.

During the normal operation, you can execute the disable interrupt (DI) instruction at any time to globally disable interrupt processing. The EI and DI instructions change the value of bit 0 in the SYM register.

5.4 System-Level Interrupt Control Registers

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

- The Interrupt Mask Register (IMR) enables (unmasks) 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 5-1 describes the overview of interrupt control register.

Table 5-1 Interrupt Control Register Overview

Control Register	ID	R/W	Function Description
Interrupt mask register	IMR	R/W	Bit settings in the IMR register enables or disables processing of interrupt for each of the eight interrupt levels: IRQ0–IRQ7.
Interrupt priority register	IPR	R/W	Controls the relative processing priorities of the interrupt levels. The eight levels of S3F8S39/S3F8S35 are organized into three groups: • Group A is IRQ0 and IRQ1 • Group B is IRQ2, IRQ3, and IRQ4 • 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 and dynamic global interrupt processing.

NOTE: Disable all interrupts before IMR register is changed to any value. Use DI instruction here.

5.5 Interrupt Processing Control Points

You can control interrupt processing either globally or by specific interrupt level and source.

The system-level control points in the interrupt structure are:

- Global interrupt enable and disable (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: While writing an application program that handles interrupt processing, ensure to include the necessary register file address (register pointer) information.

Figure 5-4 illustrates the diagram of interrupt function.

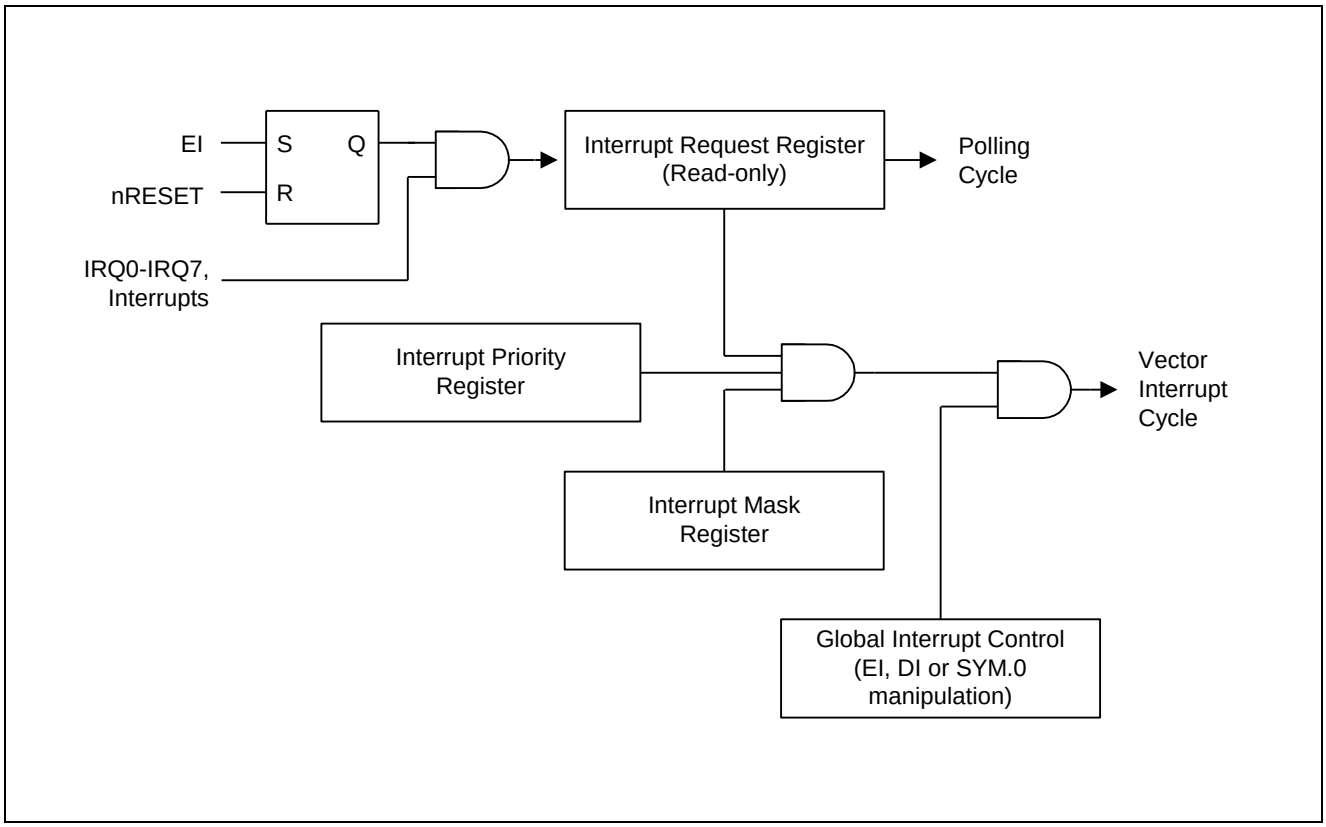


Figure 5-4 Interrupt Function Diagram

5.6 Peripheral Interrupt Control Registers

For each interrupt source there is one or more corresponding peripheral control registers that enable you to control the interrupt that related peripheral generates. Refer to *Table 5-2* for more information.

Table 5-2 lists interrupt source control and data registers.

Table 5-2 Interrupt Source Control and Data Registers

Interrupt Source	Interrupt Level	Register (s)	Location (s)
Timer A overflow Timer A match/capture Stop Wake-Up Timer	IRQ0	TACON TAPS TADATA TACNT SWTCON	E2H, BANK0 E3H, BANK0 E1H, BANK0 E0H, BANK0 FAH, BANK0
Timer B match	IRQ1	TBCON TBDATA0H TBDATA0L TBDATA1H TBDATA1L TBTRG	09H, Page 8 0AH, Page 8 0BH, Page 8 0CH, Page 8 0DH, Page 8 0EH, Page 8
Timer 0 overflow Timer 0 match/capture	IRQ2	T0CON T0DATAH T0DATAL T0CNTH T0CNTL T0PS	ECH, BANK0 EAH, BANK0 EBH, BANK0 E8H, BANK0 E9H, BANK0 EDH, BANK0
Timer 1 overflow Timer 1 match/capture Timer 2 overflow Timer 2 match/capture	IRQ3	T1CON T1DATAH T1DATAL T1CNTH T1CNTL T1PS T2CON T2DATAH T2DATAL T2CNTH T2CNTL T2PS	F2H, BANK0 F0H, BANK0 F1H, BANK0 EEH, BANK0 EFH, BANK0 F3H, BANK0 F8H, BANK0 F6H, BANK0 F7H, BANK0 F4H, BANK0 F5H, BANK0 F9H, BANK0
LVD Interrupt Watch Timer overflow SPI Interrupt IIC transmit/receive Interrupt	IRQ4	LVDCON WTCON SPICCON SPISTAT SPIDATA ICCR ICSR IDSR IAR	FFH, BANK1 FEH, BANK1 F2H, BANK1 F3H, BANK1 F4H, BANK1 05H, PAGE8 06H, PAGE8 07H, PAGE8 08H, PAGE8
UART0 data transmit UART0 data receive UART1 data transmit UART1 data receive	IRQ5	UART0CONH UART0CONL UDATA0 BRDATA0 UART1CONH UART1CONL	F5H, BANK1 F6H, BANK1 F7H, BANK1 F8H, BANK1 F9H, BANK1 FAH, BANK1

Interrupt Source	Interrupt Level	Register (s)	Location (s)
		UDATA1 BRDATA1	FBH, BANK1 FCH, BANK1
P2.0 External Interrupt P2.1 External Interrupt P2.2 External Interrupt P2.3 External Interrupt	IRQ6	P2CONL P2INT PINTPND	E5H, BANK1 E6H, BANK1 F1H, BANK1
P3.4 External Interrupt P3.5 External Interrupt P3.6 External Interrupt P3.7 External Interrupt	IRQ7	P3CONH P3INT PINTPND	E8H, BANK1 EAH, BANK1 F1H, BANK1

NOTE: If an interrupt unmasks (enable interrupt level) in the IMR register, execute a DI instruction before clearing the pending bit or changing the enable bit of the corresponding interrupt.

5.7 System Mode Register

The system mode register (SYM) (DEH, Set1), is used to globally enable and disable interrupt processing and to control fast interrupt processing Refer to *Figure 5-5* for more information.

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. To enable interrupt processing, include an enable interrupt (EI) instruction 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, use EI and DI instructions for this purpose.

Figure 5-5 illustrates the system mode register.

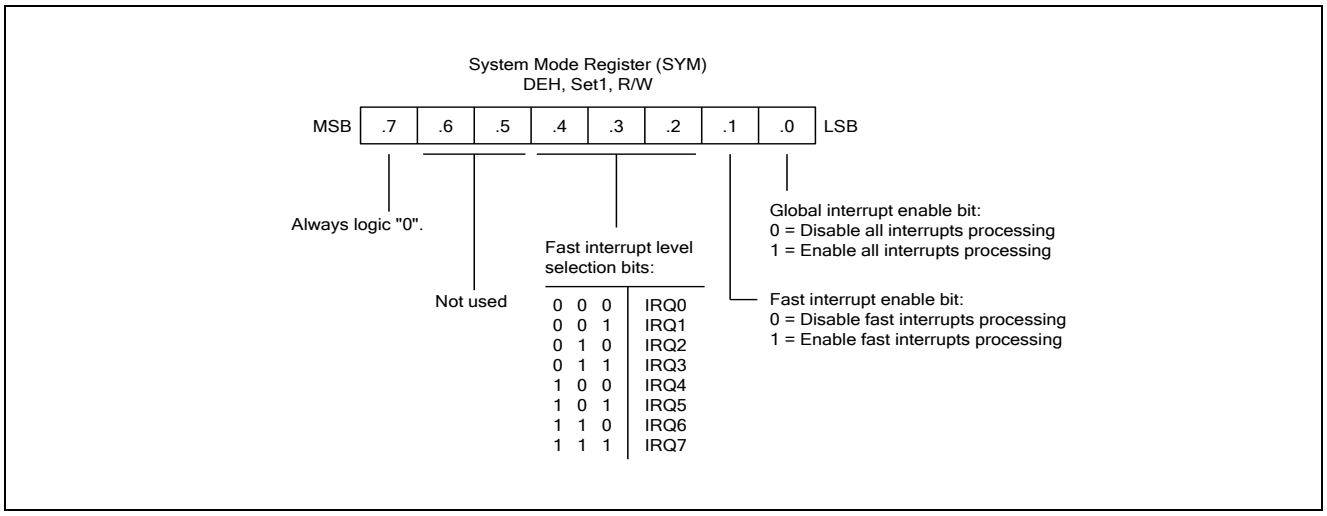


Figure 5-5 System Mode Register (SYM)

5.8 Interrupt Mask Register

The interrupt mask register (IMR) (DDH, Set1) 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, Set1. Bit values can be read and written by instructions using the Register addressing mode.

Figure 5-6 illustrates the interrupt mask register.

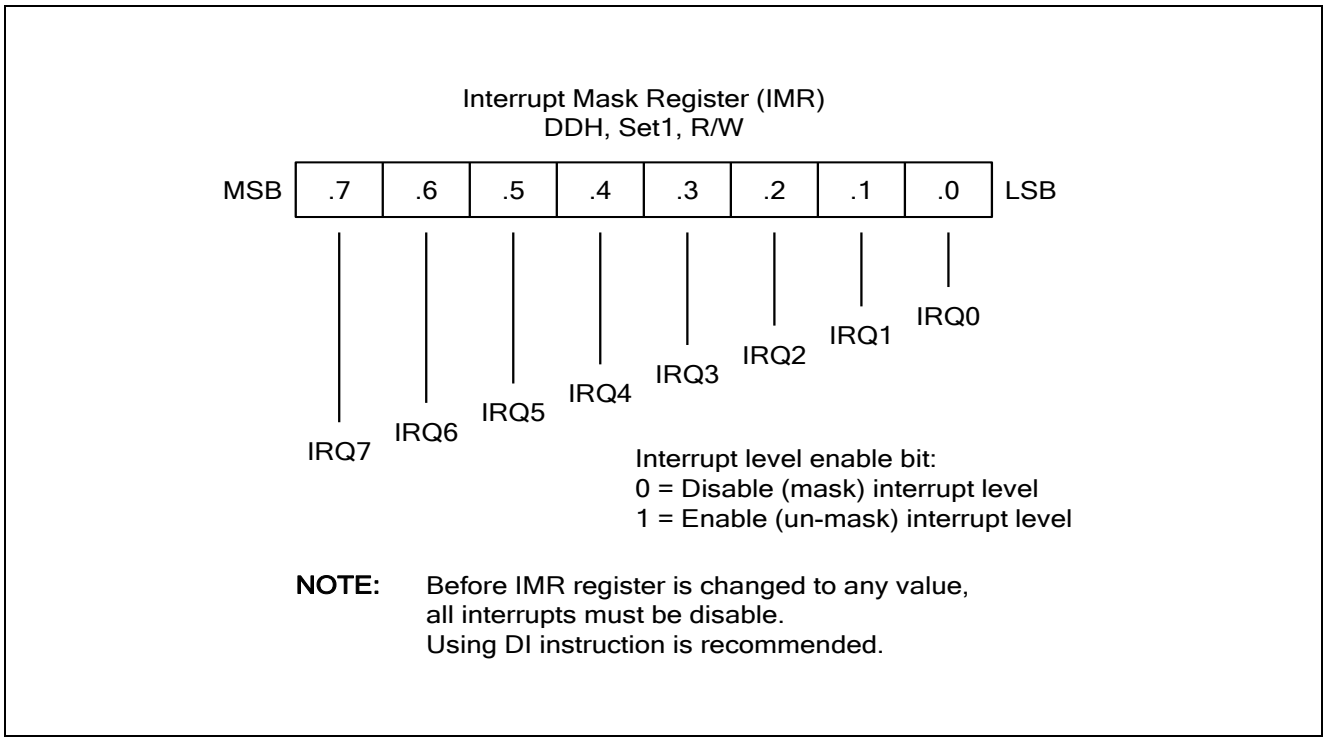


Figure 5-6 Interrupt Mask Register (IMR)

5.9 Interrupt Priority Register

The interrupt priority register (IPR) (FFH, Set 1, Bank 0) is used to set the relative priorities of the interrupt levels in the interrupt structure of microcontroller. 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 Refer to *Figure 5-7* for more information.

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

Figure 5-7 illustrates the interrupt request priority groups.

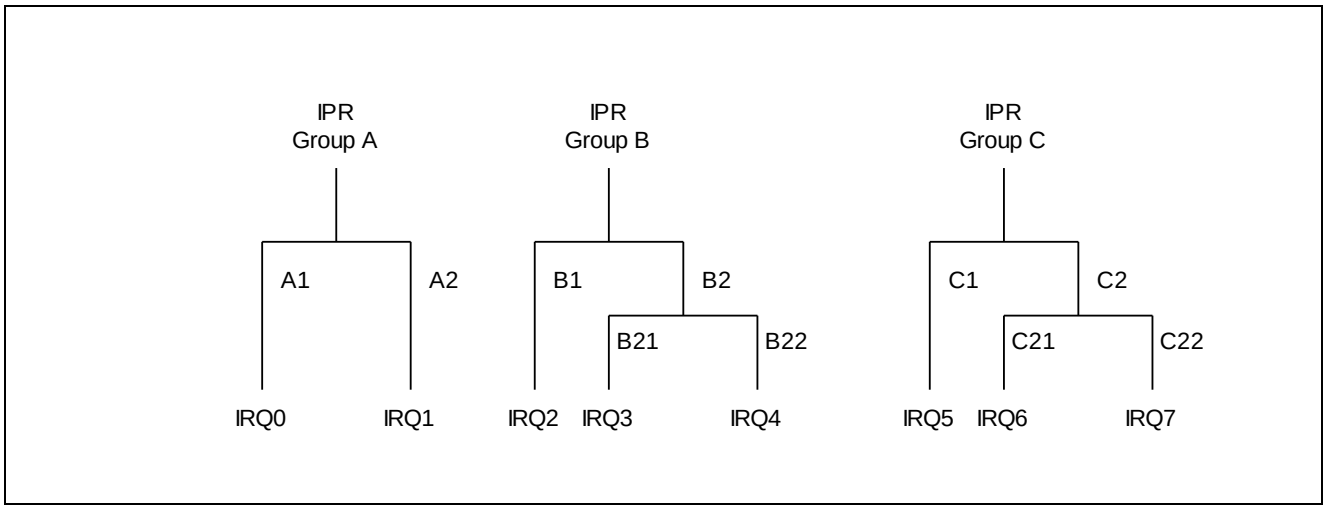


Figure 5-7 Interrupt Request Priority Groups

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 select the group relationship B > C > A. The setting "101B" selects the relationship C > B > A.

The functions of other IPR bit settings are:

- 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. The IPR.6 defines the subgroup C relationship. The IPR.5 controls the interrupt group C.
- IPR.0 controls the relative priority setting of IRQ0 and IRQ1 interrupts.

Figure 5-8 illustrates the interrupt priority register.

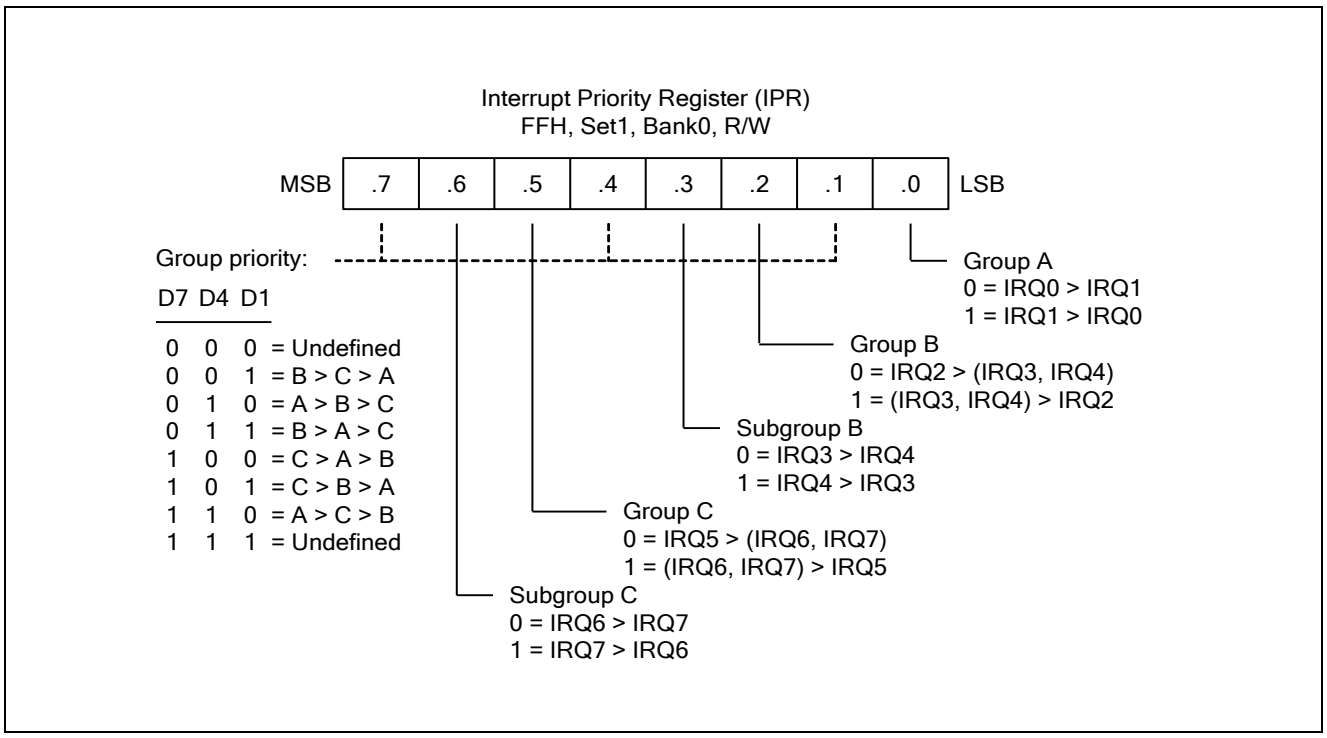


Figure 5-8 Interrupt Priority Register (IPR)

5.10 Interrupt Request Register

You can poll bit values in the interrupt request register (IRQ) (DCH, Set1) to monitor interrupt request status for all levels in the interrupt structure of microcontroller. Each bit corresponds to the interrupt level of the same number: Bit 0 to IRQ0, bit 1 to IRQ1 and so on. "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.

The IRQ bit values are read-only addressable and uses register addressing mode. You can read (test) the contents of the IRQ register 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). When 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. This way, you can determine which events occurred while the interrupt structure was globally disabled.

Figure 5-9 illustrates the interrupt request register.

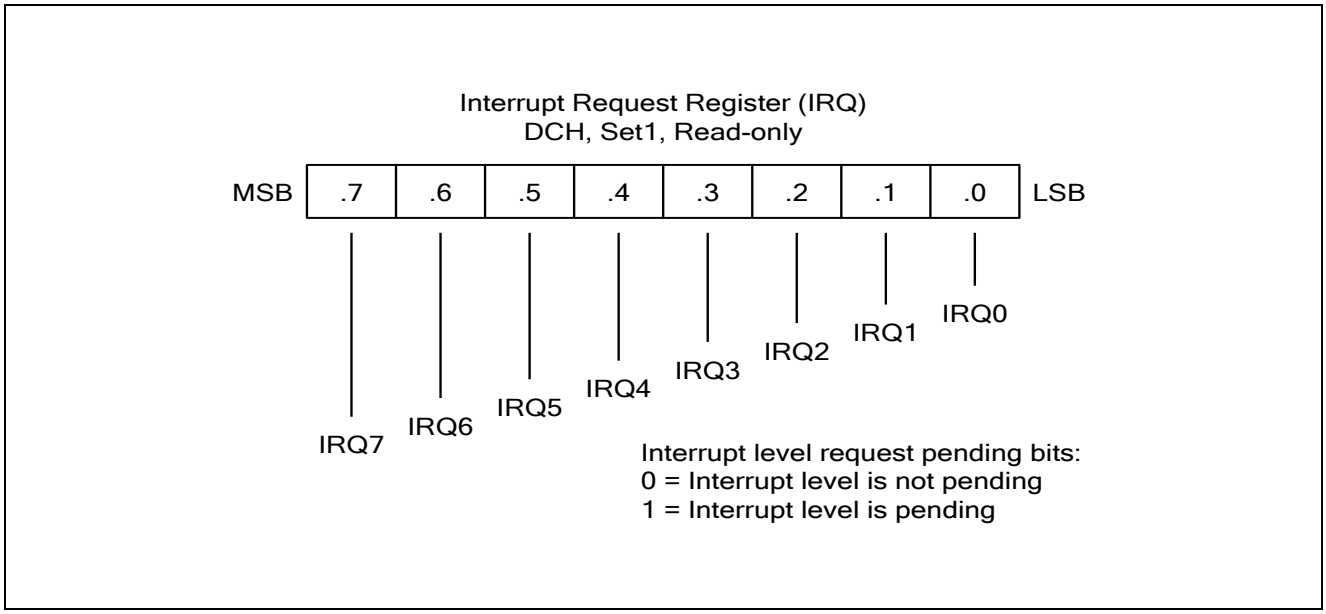


Figure 5-9 Interrupt Request Register (IRQ)

5.11 Interrupt Pending Function Types

This section includes:

- Overview
- Pending bits cleared automatically by hardware
- Pending bits cleared by the service routine

5.11.1 Overview

There are two types of interrupt pending bits:

- One type that is automatically cleared by hardware after interrupt service routine is acknowledged and executed.
- Another type that must be cleared in the interrupt service routine.

5.11.2 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 S3F8S39/S3F8S35 interrupt structure, TimerA, Timer0, Timer1, and Timer2 overflow interrupts, TimerB match interrupt and LVD interrupt belong to this category of interrupts, in which pending bits can be cleared automatically by hardware.

5.11.3 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 should 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.

5.12 Interrupt Source Polling Sequence

The sequence of interrupt request polling and servicing is:

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 verifies the interrupt level of source.
4. The CPU generates an interrupt acknowledge signal.
5. Interrupt logic determines the vector address of interrupt.
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.

5.12.1 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 level is 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 processing sequence:

1. Resets (clears to "0") the interrupt enable bit in the SYM register (SYM.0) to disable all subsequent interrupts.
2. Saves the program counter (PC) and status flags to the system stack.
3. Branches to the interrupt vector to fetch the address of the service routine.
4. Passes 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.

5.12.2 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.

To process vectored interrupt:

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.

5.12.3 Nesting of Vectored Interrupts

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

To accomplish, execute these steps:

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

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

5.12.4 Instruction Pointer

All the S3C8/S3F8-series microcontrollers adopt instruction pointer (IP) 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).

5.12.5 Fast Interrupt Processing

Fast interrupt processing enables an interrupt within a given level to be completed in approximately six clock cycles rather than the usual 16 clock cycles. To select a specific interrupt level for fast interrupt processing, write the appropriate 3-bit value to SYM.4-SYM.2. Next, to enable fast interrupt processing for the selected level, set SYM.1 to "1".

Two other system registers support fast interrupt processing:

- The IP that 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 are stored in an unmapped, dedicated register called FLAGS' ("FLAGS prime").

NOTE: For the S3F8S39/S3F8S35 microcontroller, the service routine for any one of the eight interrupts levels: IRQ0–IRQ7, can be selected for fast interrupt processing.

5.12.6 Procedure for Initiating Fast Interrupts

To initiate fast interrupt processing:

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

5.12.7 Fast Interrupt Service Routine

When an interrupt occurs in the level that it selects for fast interrupt processing, these 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.

5.12.8 Relationship to Interrupt Pending Bit Types

As described previously in Section *5.1.3 Sources*, 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 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 software.

5.13 Programming Guidelines

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, load the IP with a new start address when the fast interrupt service routine ends. Refer to IRET instruction in Chapter 6 for more information.

6 Instruction Set

6.1 Overview

The SAM8RC instruction set supports 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 are:

- Supports a full complement of 8-bit arithmetic and logic operations that includes multiplication and division
- No special I/O instructions (I/O control/data registers are mapped directly into the register file)
- Includes decimal adjustment in binary-coded decimal (BCD) operations
- 16-bit (word) data can be incremented and decremented
- Provides 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. Refer to Chapter 2 "Address Spaces" for more information about register addressing.

6.1.3 Addressing Modes

There are seven explicit addressing modes, namely:

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

Refer to Chapter 3 "Addressing Modes" for more information on these addressing modes.

Figure 6-1 describes the Instruction Group summary.

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 predecrement
LDCPD	dst, src	Load program memory with predecrement
LDEPI	dst, src	Load external data memory with preincrement
LDCPI	dst, src	Load program memory with preincrement
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
EI	–	Enable interrupts
IDLE	–	Enter Idle Mode

Mnemonic	Operands	Instruction
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

The flags register (FLAGS) contains 8 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.

FLAGS also contain 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 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. For example, the AND instruction updates the Zero, Sign and Overflow flags on the basis of outcome of the AND instruction. When the AND instruction uses the FLAGS as the destination, two writes occur simultaneously to the FLAGS that produces an unpredictable result.

Figure 6-1 illustrates the system Flags register (FLAGS).

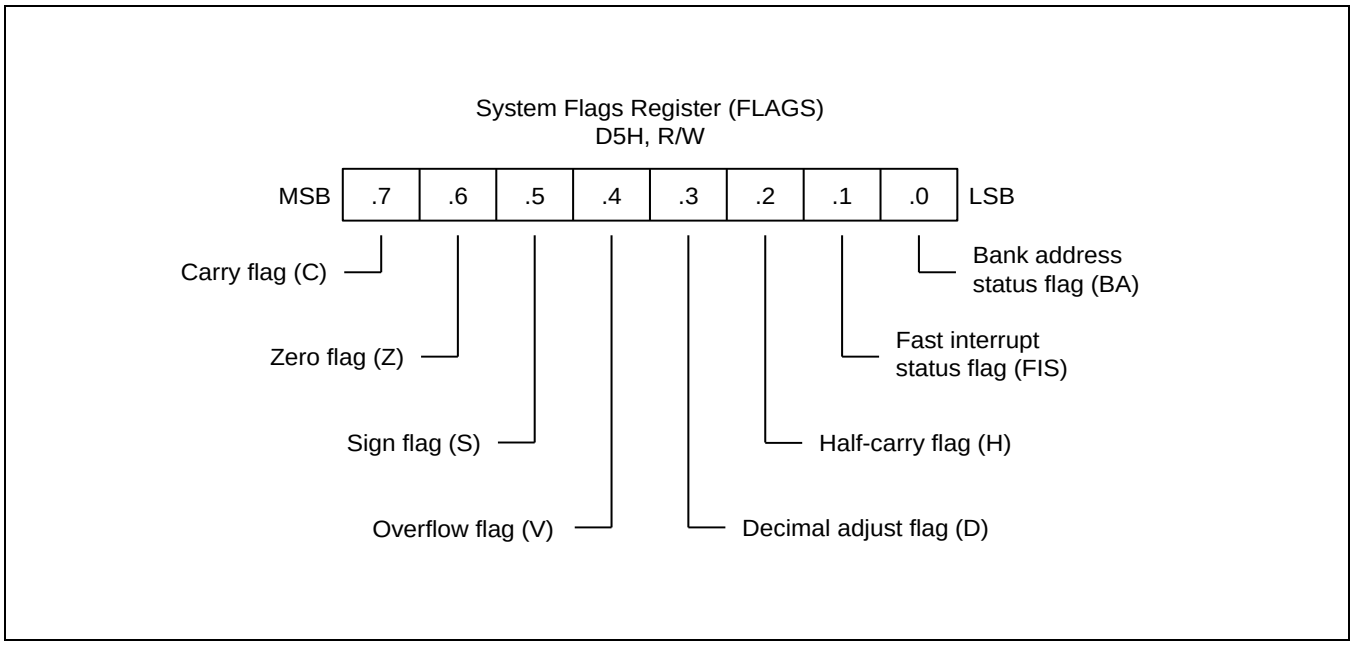


Figure 6-1 System Flags Register (FLAGS)

6.3 Flag Descriptions

C	**Carry Flag (FLAGS.7)** The C flag is set to "1" when the result from an arithmetic operation generates carry-out from or 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" when 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 also clears to "0" following logic operations.
D	**Decimal Adjust Flag (FLAGS.3)** The DA bit is used to specify what type of instruction that it executed last during BCD operations so that a subsequent decimal adjust operation can execute correctly. Usually, programmers cannot access DA and cannot use 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. A program seldom accesses the H flag.
FIS	**Fast Interrupt Status Flag (FLAGS.1)** The FIS bit is set during a fast interrupt cycle and resets during the Interrupt Return (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 it currently selects in the set 1 area of the internal register file, 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.

6.4 Instruction Set Notation

Table 6-2 describes the Flag Notation conventions.

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 describes the Instruction Set symbols.

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 describes the Instruction Notation conventions.

Table 6-4 Instruction Notation Conventions

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

Table 6-5 lists the Opcode quick reference (0–7).

Table 6-5 Opcode Quick Reference (0–7)

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

Table 6-6 lists the Opcode Quick references (8–F).

Table 6-6 Opcode Quick Reference (8–F)

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

6.5 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.

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

Table 6-7 describes the conditional codes.

Table 6-7 Condition Codes

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

NOTE:

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

6.6 Instruction Descriptions

This section describes information and programming examples for each instruction in the SAM8 instruction set. It arranges information in a consistent format for improved readability and for fast referencing.

It includes the information listed in each instruction description:

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

6.6.1 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: Sets if there is a carry from the most significant bit of the result; otherwise it clears.

Z: Sets if the result is "0"; otherwise it clears.

S: Sets if the result is negative; otherwise it clears.

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

D: Always clears to "0".

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

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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 10Hby leaving 14H in register R1.

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

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

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

V: Sets 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: Sets if a carry from the low-order nibble occurs.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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 by leaving the value 15H in register R1.

6.6.3 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: Does not affect.

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

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

V: Always cleared to "0".

D: Does not affect.

H: Does not affect.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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 by leaving the value 02H in register R1.

6.6.4 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: Does not affect.

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

S: Clears to "0".

V: Does not define.

D: Does not define.

H: Does not define.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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 4 bits, the bit address "b" is 3 bits, and the LSB address value is 1 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 1 value of the source register ("0") with the bit 0 value of register R1 (destination) by leaving the value 06H (00000110B) in register R1.

6.6.5 BCP-Bit Compare

BCP dst, src.b

Operation: dst (0)-src (b)

The specified bit of the source is compared to (subtracts from) bit zero (LSB) of the destination. The zero flag is set when the bits are the same; otherwise it clears. The comparison does not affect the contents of both operands.

Flags: **C**: Does not affect.

Z: Sets if two bits are similar; otherwise it clears.

S: Clears to "0".

V: Does not define.

D: Does not define.

H: Does not define.

Format:

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

NOTE: In the second byte of the instruction format, the destination address is 4 bits, the bit address "b" is 3 bits, and the LSB address value is 1 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 1 one of the source register (01H) and bit 0 of the destination register (R1). As the bit values are not identical, the zero flag bit (Z) is cleared in the FLAGS (0D5H).

6.6.6 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: Does not affect.

Z: Sets if the result is "0"; Otherwise clears.

S: Clears to "0".

V: Does not define.

D: Does not define.

H: Does not define.

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 4 bits, the bit address "b".is 3 bits, and the LSB address value is 1 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 (0D5H) is cleared.

6.6.7 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: Does not affect.

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 4 bits, the bit address "b" is 3 bits, and the LSB address value is 1 bit in length.

Example: Given: R1 = 07H:

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

When the value of working register R1 is 07H (00000111B), the statement "BITR R1.1" clears bit1 of the destination register R1by leaving the value 05H (00000101B).

6.6.8 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 4 bits, the bit address "b" is 3 bits, and the LSB address value is 1 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 3 of the destination register R1 to "1" by leaving the value 0FH (00001111B).

6.6.9 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: Does not affect.

Z: Sets if the result is "0"; otherwise it clears.

S: Clears to "0".

V: Does not affect.

D: Does not affect.

H: Does not affect.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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 4 bits, the bit address "b" is 3 bits, and the LSB address value is 1 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 1 of register 01H (source) with bit 0 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 2 of register 01H (destination) with bit 0 of R1 (source). This leaves the value 07H in register 01H.

6.6.10 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: Does not affect any flags.

Format:

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

NOTE: In the second byte of the instruction format, the source address is 4 bits, the bit address "b" is 3 bits, and the LSB address value is 1 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. As it is "0", the relative address is added to the PC and the PC jumps to the memory location pointed to the SKIP. (Ensure that the memory location must be within the allowed range of + 127 to 128.)

6.6.11 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: Does not affect any flags.

Format:

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

NOTE: In the second byte of the instruction format, the source address is 4 bits, the bit address "b" is 3 bits, and the LSB address value is 1 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 1 in the source register (R1). As 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. (Ensure that the memory location must be within the allowed range of + 127 to 128.)

6.6.12 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. It does not affect other bits of destination. It does not affect the source.

Flags:

C: Does not affect.

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

S: Clears to "0".

V: Does not define.

D: Does not affect.

H: Does not affect.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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       →       R = 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 1 of register 01H (source) with bit 0 of R1 (destination). The result bit value is stored in bit 0 of R1, changing its value from 07H to 06H. It does not affect the value of source register 01H.

6.6.13 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: Does not affect any flags.

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, when 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 similar 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, when the program address 0040H contains 35H and program address 0041H contains 21H, the statement "CALL #40H" produces the same result as in the second example.

6.6.14 CCF-Complement Carry Flag

CCF

Operation: C ← NOT C

Complements the carry flag (C). When 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:** Complements.

Does not affect any other flags.

Format:

						Bytes	Cycles	Opcode (Hex)
	opc					1	4	EF

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

CCF

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

6.6.15 CLR-Clear

CLR dst

Operation: dst ← "0"

Clears the destination location to "0".

Flags: Does not affect any other flags..

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.

6.6.16 COM-Complement

COM dst

Operation: dst ← NOT dst

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

Flags:

C: Does not affect.

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: Does not affect.

H: Does not affect.

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: it changes all logic ones to logic zeros, and vice-versa, leaving the value 0F8H (11111000B).

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

6.6.17 CP-Compare

CP dst, src

Operation: dst-src

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

Flags:

C: Sets if a "borrow" occurs (src > dst); otherwise it clears.

Z: Sets if the result is "0"; otherwise it clears.

S: Sets if the result is negative; otherwise it clears.

V: Sets if arithmetic overflow occurred; otherwise it clears.

D: Does not affect.

H: Does not affect.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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). As 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". 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.

6.6.18 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 (subtracts from) the destination operand. When the result is "0", the relative address is added to the program counter and the 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: Does not affect any flags.

Format:

							Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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 contains the value 03H, and register 03 contains 02H. The statement "CPIJE R1, @R2, SKIP" compares the @R2 value 02H (00000010B) to 02H (00000010B). As the result of the comparison is equal, the relative address is added to the PC and the PC then jumps to the memory location that SKIP points to. The source register (R2) is incremented by one, leaving a value of 04H. (Ensure that the memory location is within the allowed range of + 127 to – 128.)

6.6.19 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 (subtracts from) the destination operand. When 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: Does not affect any flags.

Format:

							Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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 contains the value 04H. The statement "CPIJNE R1, @R2, SKIP" subtracts 04H (00000100B) from 02H (00000010B). As 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 that SKIP points to. The source pointer register (R2) is also incremented by one, leaving a value of 04H. (Ensure that the memory location is within the allowed range of + 127 to – 128.)

6.6.20 DA-Decimal Adjust

DA dst

Operation: dst ← DA dst

The destination operand is adjusted to form two 4-bit BCD digits after an addition or subtraction operation. For addition (ADD, ADC) or subtraction (SUB, SBC). the following table indicates the operation that it performs (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: Sets if there was a carry from the most significant bit; otherwise it clears (see table).

Z: Sets if result is "0"; otherwise it clears.

S: Sets if result bit 7 is set; otherwise it clears.

V: Does not define.

D: Does not affect.

H: Does not affect.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst
	opc	dst				2	4	40	R
							4	41	IR

6.6.21 DA-Decimal Adjust

DA (Continued)

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 by using the BCD values 15 and 27, the result should be 42. However, the sum is incorrect, however, when the binary representations are added in the destination location by 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 provided previously, 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).

6.6.22 DEC-Decrement

DEC dst

Operation: dst ← dst-1

The contents of the destination operand are decremented by one.

Flags:

C: Does not affect.

Z: Sets if the result is "0"; otherwise it clears.

S: Sets if result is negative; otherwise it clears.

V: Sets if arithmetic overflow occurred; otherwise it clears.

D: Does not affect.

H: Does not affect.

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 by1, leaving the value 02H. In the second example, the statement "DEC @R1" decrements the value 10H contained in the destination register 03H by 1, leaving the value 0FH.

6.6.23 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: Does not affect.

Z: Sets if the result is "0"; otherwise it clears.

S: Sets if the result is negative; otherwise it clears.

V: Sets if arithmetic overflow occurred; otherwise it clears.

D: Does not affect.

H: Does not affect.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst
	opc	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 contains the value 34H. The statement "DECW RR0" addresses R0 and the next operand R1 as a 16-bit word and decrements the value of R1 by1, 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, that you should use DECW as shown in this example:

```
LOOP:   DECW    RR0
        LD      R2, R1
        OR      R2, R0
        JR      NZ, LOOP
```

6.6.24 DI-Disable Interrupts

DI

Operation: SYM (0) ← 0

Bit zero of the system mode control register, SYM.0, is cleared to "0" by globally disabling all interrupt processing. Interrupt requests continues to set their respective interrupt pending bits, however the CPU does not service them when it disables the interrupt processing.

Flags: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)
	opc					1	4	8F

Example: Given: SYM = 01H:

DI

When 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" by disabling the interrupt processing.

Before changing IMR, interrupt pending and interrupt source control register, ensure that it is in the DI state.

6.6.25 DIV-Divide (Unsigned)

DIV dst, src

Operation: dst ÷ src

dst (UPPER) ← REMAINDER

dst (LOWER) ← QUOTIENT

The source operand (8 bits) divides the destination operand (16 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 that it stores in the upper and lower halves of the destination for quotient and remainder are incorrect. Both operands are treated as unsigned integers.

Flags:

C: Sets if the V flag is set and quotient is between 2^8 and 2^9 1; otherwise it clears.

Z: Set sif divisor or quotient = "0"; otherwise it clears.

S: Sets if MSB of quotient = "1"; otherwise it clears.

V: Sets if quotient is ≥ 2^8 or if divisor = "0"; otherwise it clears.

D: Does not affect.

H: Does not affect.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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).

6.6.26 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 after the DJNZ statement.

NOTE: In case of using DJNZ instruction, the working register that it uses as a counter should be set at one of locations 0C0H to 0CFH with SRP, SRP0, or SRP1 instruction.

Flags: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst
	r | opc	dst				2	8 (Jump taken)	rA	RA
							8 (No jump)	r = 0 to F	

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 by leaving the value 01H. As the contents of R1 after the decrement are non-zero, the jump is taken to the relative address that the LOOP label specifies.

6.6.27 EI-Enable Interrupts

EI

Operation: SYM (0) ← 1

An EI instruction sets bit 0 of the system mode register, SYM.0 to "1". This enables interrupts to be serviced as they occur (assuming they have highest priority). If the pending bit of an interrupt is set while interrupt processing was disabled (by executing a DI instruction), it will be serviced when you execute the EI instruction.

Flags: Does not affect any flags.

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 by enabling all interrupts. (SYM.0 is the enable bit for global interrupt processing.)

6.6.28 ENTER-Enter

ENTER

Operation:

SP ← SP-2
@SP ← IP
IP ← PC
PC ← @IP
IP ← IP + 2

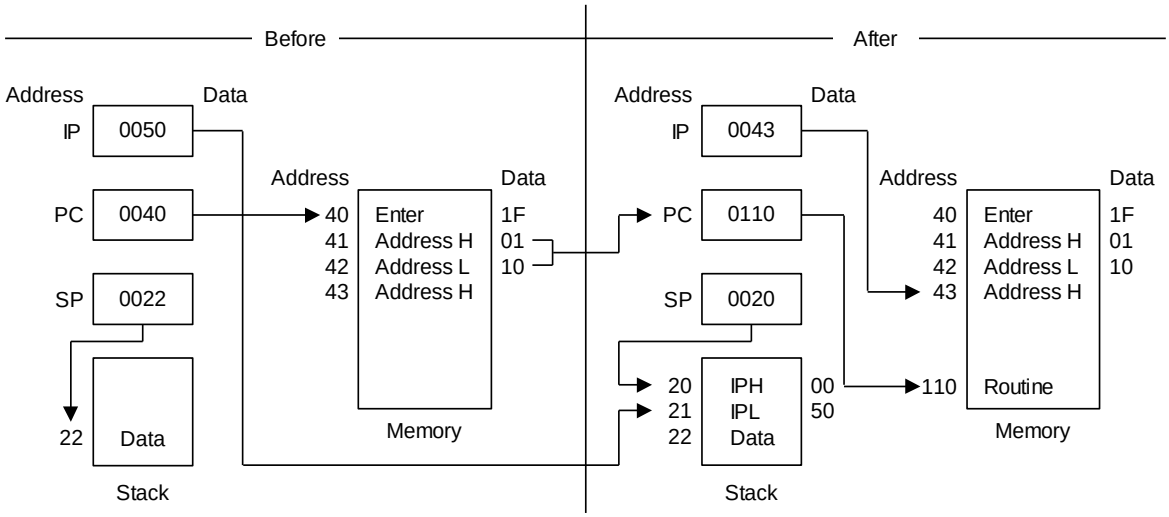
This instruction is useful when implementing threaded-code languages. The contents of the instruction pointer are pushed to the stack. The program counter (PC) value is then written to the instruction pointer. The program memory word that is pointed to by the instruction pointer is loaded into the PC, and the instruction pointer is incremented by two.

Flags: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)
	opc					1	14	1F

Example: The diagram below shows one example of how to use an ENTER statement.



6.6.29 EXIT-Exit

EXIT

Operation:

IP ← @SP
SP ← SP + 2
PC ← @IP
IP ← IP + 2

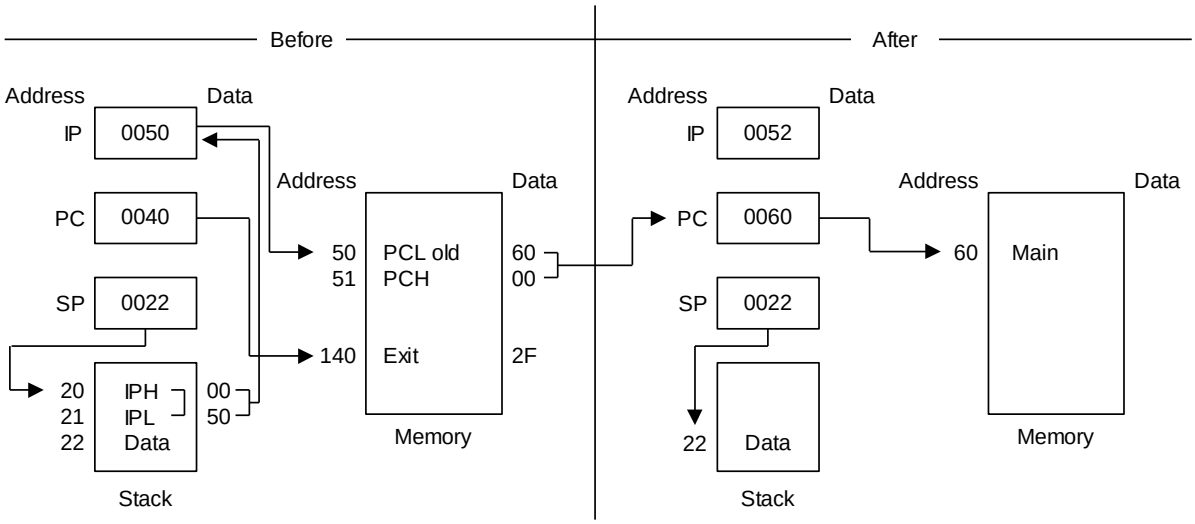
This instruction is useful when implementing threaded-code languages. The stack value is popped and loaded into the instruction pointer. The program memory word that is pointed to by the instruction pointer is then loaded into the program counter, and the instruction pointer is incremented by two.

Flags: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)
	opc					1	14 (Internal stack)	2F

Example: The diagram below shows one example of how to use an EXIT statement.



6.6.30 IDLE-Idle Operation

IDLE

Operation:

The IDLE instruction stops the CPU clock while allowing system clock oscillation to continue. An Interrupt request (IRQ) or external reset operation release idle.

Flags: Does not affect any flags.

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.

6.6.31 INC-Increment

INC dst

Operation: dst ← dst + 1

The contents of the destination operand are incremented by one.

Flags:

C: Does not affect.

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

S: Set if the result is negative; otherwise it clears.

V: Set if arithmetic overflow occurred; otherwise it clears.

D: Does not affect.

H: Does not affect.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst
	dst | opc					1	4	rE	r
								r = 0 to F	
	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.

6.6.32 INCW-Increment Word

INCW dst

Operation: dst ← dst + 1

It treats the contents of the destination (which must be an even address) and the byte following that location as a single 16-bit value. This value is incremented by one.

Flags: **C**: Does not affect.

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

S: Set if the result is negative; otherwise it clears.

V: Set if arithmetic overflow occurred; otherwise it clears.

D: Does not affect.

H: Does not affect.

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, you should use INCW as shown :

```
LOOP:   INCW    RR0
        LD      R2, R1
        OR      R2, R0
        JR      NZ, LOOP
```

6.6.33 IRET-Interrupt Return

IRET IRET (Normal) IRET (Fast)

Operation:

FLAGS ← @SPPC ↔ IP
SP ← SP + 1 FLAGS ← FLAGS
PC ← @SP FIS ← 0
SP ← SP + 2
SYM (0) ← 1

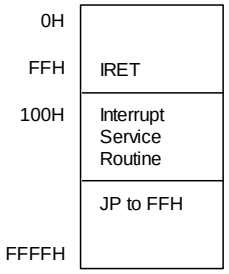
This instruction is used at the end of an interrupt service routine. It restores the flag register and the program counter. It also reenables global interrupts. A "normal IRET" is executed only if the fast interrupt status bit (FIS, bit one of the FLAGS register, 0D5H) is cleared (= "0"). If a fast interrupt occurred, IRET clears the FIS bit that was set at the beginning of the service routine.

Flags: Restores all flags to their original settings (that is, the settings before the interrupt occurs).

Format:

	IRET (Normal)					Bytes	Cycles	Opcode (Hex)
	opc					1	10 (Internal stack)	BF
							12 (Internal stack)	
	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 this fast interrupt example in the figure 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 preceded by a clearing of the interrupt status (as with a reset of the IPR register).

6.6.34 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 when the condition that condition code (cc) specifies 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: Does not affect any flags.

Format: (1)

	(2)				Bytes	Cycles	Opcode (Hex)	Addr Mode dst
	cc | opc	dst			3	8	ccD	DA
							cc = 0 to F	
	opc	dst			2	8	30	IRR

NOTE:

1. You can use the 3 byte format 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 you can set the carry flag 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 to the next 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 by leaving the value 0120H.

6.6.35 JR-Jump Relative

JR cc, dst

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

When 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. (Refer to the list of condition codes for more information).

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: Does not affect any flags.

Format:

	(NOTE)				Bytes	Cycles	Opcode (Hex)	Addr Mode dst
	cc | opc	dst			2	6	ccB	RA
							cc = 0 to F	

NOTE: In the first byte of the 2 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 you set the carry flag (that is, if the condition code is true), the statement "JR C, LABEL_X" passes control to the statement whose address is currently in the PC. Otherwise, the program instruction following the JR would be executed.

6.6.36 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: Does not affect any flags.

Format:

					Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
	dst | opc	src			2	4	rC	r	IM
						4	r8	r	R
	src | opc	dst			2	4	r9	R	r
							r = 0 to F		
	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

6.6.37 LD-Load

LD (Continued)

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

```
LD      R0, #10H        →      R0 = 10H
LD      R0, 01H         →      R0 = 20H, register 01H = 20H
LD      01H, R0         →      Register 01H = 01H, R0 = 01H
LD      R1, @R0         →      R1 = 20H, R0 = 01H
LD      @R0, R1         →      R0 = 01H, R1 = 0AH, register 01H = 0AH
LD      00H, 01H        →      Register 00H = 20H, register 01 = 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      @0H, 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
```

6.6.38 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 0 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: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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 4 bits, the bit address 'b' is 3 bits, and the LSB address value is 1-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 by 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 by leaving 04H in general register 00H.

6.6.39 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 od number for data memory.

Flags: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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 1 byte each.
3. For formats 5 and 6, the destination address "XL[rr]" and the source address "XL[rr]" are 2 byte each.
4. The DA and r source values for formats 7 and 8 are used to address program memory. You can use the second set of values, used in formats 9 and 10, to address data memory.

6.6.40 LDC/LDE-Load Memory

LDC/LDE (Continued)

Examples: Given: R0 = 11H, R = 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, R         ; 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       ; 0 ← 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: Masked ROM type devices do not support these instructions.

6.6.41 LDCD/LDED-Load Memory and Decrement

LDCD/LDED dst, src

Operation: dst ← src

rr ← rr – 1

You can uses these instructions for user stacks or block transfers of data from program or data memory to the register file. A working register pair addresses specifies the address of the memory location. 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 and LDED refer to external data memory respectively. The assembler makes "Irr" an even number for program memory and an odd number for data memory.

Flags: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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
```

6.6.42 LDCI/LDEI-Load Memory and Increment

LDCI/LDEI dst, src

Operation: dst ← src

rr ← rr + 1

You can use these instructions for user stacks or block transfers of data from program or data memory to the register file. A working register pair specifies the address of the memory location. 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 and LDEI refer to external data memory respectively. The assembler makes "Irr" even for program memory and odd for data memory.

Flags: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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
```

6.6.43 LDCPD/LDEPD-Load Memory with Pre-Decrement

LDCPD/ LDEPD dst, src

Operation: rr ← rr – 1

dst ← src

You can use these instructions for block transfers of data from program or data memory from the register file. A working register pair specifies the address of the memory location. It decrements the address of the memory location. The contents of the source location are then loaded into the destination location. The contents of the source are unaffected.

LDCPD and LDEPD refer to external data memory respectively. The assembler makes "Irr" an even number for program memory and an odd number for external data memory.

Flags: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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
```

6.6.44 LDCPI/LDEPI-Load Memory with Pre-Increment

LDCPI/
LDEPI dst, src

Operation: rr ← rr + 1

dst ← src

You can these instructions for block transfers of data from program or data memory from the register file. A working register pair specifies the address of the memory location. It increments the address of the memory location. 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: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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
```

6.6.45 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: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode 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, 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.

6.6.46 MULT-Multiply (Unsigned)

MULT dst, src

Operation: dst ← dst × src

The source operand (8 bits) multiplies the 8-bit destination operand (even register of the register pair) It stores the product (16 bits) in the register pair that destination address specifies.

Flags:

C: Set if result is > 255; otherwise it clears.

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

S: Set if MSB of the result is a "1"; otherwise it clears.

V: Cleared.

D: Does not affect.

H: Does not affect.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
	opc	src	dst			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.

6.6.47 NEXT-Next

NEXT

Operation: PC ← @ IP

IP ← IP + 2

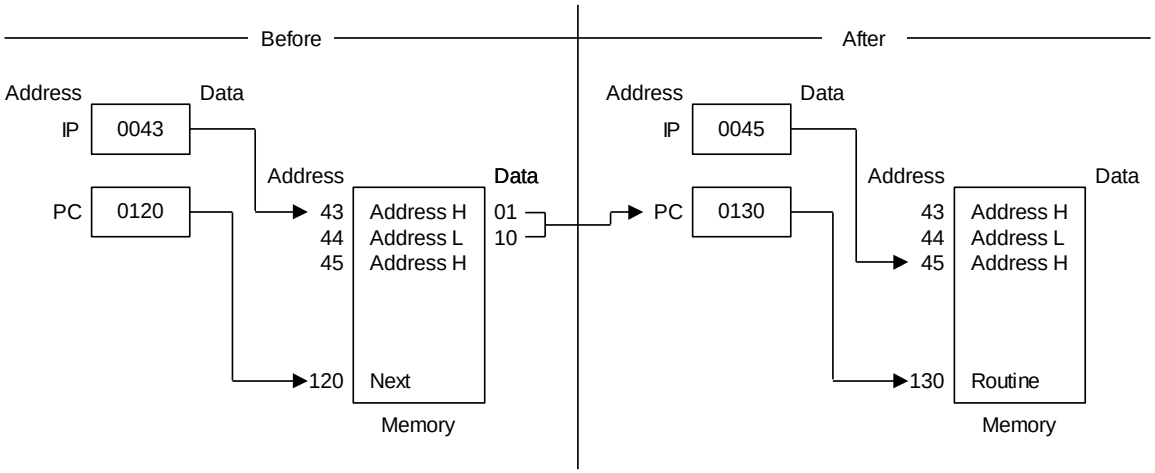
The NEXT instruction is useful when implementing threaded-code languages. The program memory word that is pointed to by the instruction pointer is loaded into the program counter. The instruction pointer is then incremented by two.

Flags: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)
	opc					1	10	0F

Example: The following diagram shows one example of how to use the NEXT instruction.



6.6.48 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: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)
	opc					1	4	FF

Example: When the instruction

NOP
is encountered in a program, operation does not occur. Instead, there is a delay in instruction execution time.

6.6.49 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 it stores a "0".

Flags:

C: Does not affect.

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

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

V: Always cleared to "0".

D: Does not affect.

H: Does not affect.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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, when working register R0 contains the value 15H and register R1 contains 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.

6.6.50 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.

6.6.51 POPUD-Pop User Stack (Decrementing)

POPUD dst, src

Operation: dst ← src

IR ← IR – 1

You can use this instruction for user-defined stacks in the register file. It loads the contents of the register file location that the user stack pointer addresses into the destination. The user stack pointer is then decremented.

Flags: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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 contains 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 oneby leaving the value 41H.

6.6.52 POPUI-Pop User Stack (Incrementing)

POPUI dst, src

Operation: dst ← src

IR ← IR + 1

You can use the POPUI instruction 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: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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. Next, it increments the user stack pointer (register 00H) by one, changing its value from 01H to 02H.

6.6.53 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 The decremented stack pointer addresses the location. The operation then adds the new value to the top of the stack.

Flags: Does not affect any flags.

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, when the stack pointer contains the value 0000H, and general register 40H contains 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.

6.6.54 PUSHUD-Push User Stack (Decrementing)

PUSHUD dst, src

Operation: IR ← IR – 1

dst ← src

You can use this instruction 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. The decremented stack pointer addresses the register.

Flags: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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 oneby leaving the value 02H. It then loads the 01H register value, 05H, into the register. The decremented user stack pointer addresses the register.

6.6.55 PUSHUI-Push User Stack (Incrementing)

PUSHUI dst, src

Operation: IR ← IR + 1

dst ← src

You can use this instruction 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. The incremented user stack pointer addresses the register location.

Flags: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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 oneby leaving the value 04H. It then loads the 01H register value, 05H, into the location. The incremented user stack pointer addresses the location.

6.6.56 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".

Does not affect any other flags.

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.

6.6.57 RET-Return

RET

Operation: PC ← @SP

SP ← SP + 2

Normally, you can use the RET instruction to return to the previously executing procedure at the end of a procedure. A CALL instruction enters the procedure. The contents of the location that the stack pointer addresses are popped into the program counter. The next statement that is executed is the one that the new program counter value addresses.

Flags: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)
	opc					1	8 (Internal stack)	AF
							10 (Internal stack)	

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 low byte of PC and it executes the instruction at location 101AH. The stack pointer now points to memory location 00FEH.

6.6.58 RL-Rotate Left

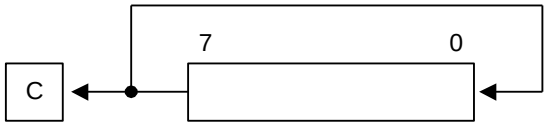
RL dst

Operation: C ← dst (7)

dst (0) ← dst (7)

dst (n + 1) ← dst (n), n = 0-6

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



Flags:

C: Set if the bit rotated from the most significant bit position (bit 7) was "1".

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

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

V: Set if arithmetic overflow occurred; otherwise it clears.

D: Does not affect.

H: Does not affect.

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

6.6.59 RLC-Rotate Left Through Carry

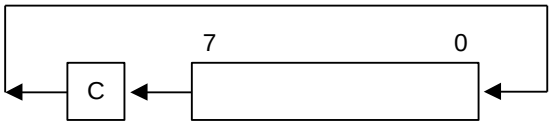
RLC dst

Operation: dst (0) ← C

C ← dst (7)

dst (n + 1) ← dst (n), n = 0-6

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



Flags:

C: Sets if the bit rotated from the most significant bit position (bit 7) was "1".

Z: Sets if the result is "0"; otherwise it clears.

S: Sets if the result bit 7 is set; otherwise it clears.

V: Sets if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; otherwise it clears.

D: Does not affect.

H: Does not affect.

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, when general register 00H has the value 0AAH (10101010B), the statement "RLC 00H" rotates 0AAH 1 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 0 of register 00Hby leaving the value 55H (01010101B). The MSB of register 00H resets the carry flag to "1" and sets the overflow flag.

6.6.60 RR-Rotate Right

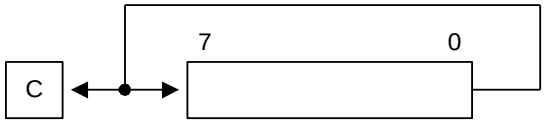
RR dst

Operation: C ← dst (0)

dst (7) ← dst (0)

dst (n) ←dst (n + 1), n = 0-6

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



Flags:

C: Sets if the bit rotated from the least significant bit position (bit zero) was "1".

Z: Sets if the result is "0"; otherwise it clears.

S: Sets if the result bit 7 is set; otherwise it clears.

V: Sets if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; otherwise it clears.

D: Does not affect.

H: Does not affect.

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, when general register 00H contains the value 31H (00110001B), the statement "RR 00H" rotates this value one bit position to the right. It moves the initial value of bit zero to bit 7by 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".

6.6.61 RRC-Rotate Right through Carry

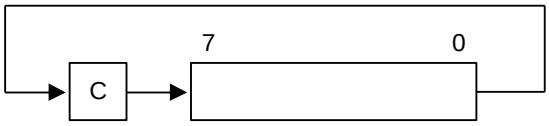
RRC dst

Operation: dst (7) ← C

C ← dst (0)

dst (n)← dst (n + 1), n = 0-6

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



Flags:

C: Sets if the bit rotated from the least significant bit position (bit zero) was "1".

Z: Sets if the result is "0" otherwise it clears.

S: Sets if the result bit 7 is set; otherwise it clears.

V: Sets if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; otherwise it clears.

D: Does not affect.

H: Does not affect.

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, when general register 00H contains the value 55H (01010101B), the statement "RRC 00H" rotates this value 1 bit position to the right. The initial value of bit 0 ("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".

6.6.62 SB0-Select Bank 0

SB0

Operation: BANK ← 0

The SB0 instruction clears the bank address flag in the FLAGS (FLAGS.0) to logic 0 by selecting the bank 0 register addressing in the set 1 area of the register file.

Flags: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)
	opc					1	4	4F

Example: The statement

SB0

clears FLAGS.0 to "0" by selecting the bank 0 register addressing.

6.6.63 SB1-Select Bank 1

SB1

Operation: BANK ← 1

The SB1 instruction sets the bank address flag in the FLAGS (FLAGS.0) to logic one by selecting bank 1 register addressing in the set 1 area of the register file. (It does not implement bank 1 in some KS88-series microcontrollers.)

Flags: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)
	opc					1	4	5F

Example: The statement

SB1

sets FLAGS.0 to "1" by selecting the bank 1 register addressing, when implemented.

6.6.64 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. You can perform subtraction 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: Sets if a borrow occurred (src > dst); otherwise it clears.

Z: Sets if the result is "0"; otherwise it clears.

S: Sets if the result is negative; otherwise it clears.

V: Sets 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; otherwise it clears.

D: Always set to "1".

H: Clears when 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	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 = 95H; C, S, and V = "1"
```

In the first example, when 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.

6.6.65 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".

Does not affect any other flags.

Format:

						Bytes	Cycles	Opcode (Hex)
	opc					1	4	DF

Example: The statement

SCF

sets the carry flag to logic 1.

6.6.66 SRA-Shift Right Arithmetic

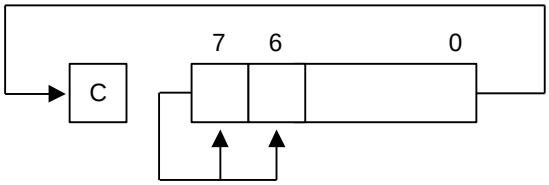
SRA dst

Operation: dst (7) ← dst (7)

C ← dst (0)

dst (n) ← dst (n + 1), n = 0-6

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



Flags:

C: Sets if the bit shifted from the LSB position (bit zero) was "1".

Z: Sets if the result is "0"; otherwise it clears.

S: Sets if the result is negative; otherwise it clears.

V: Always cleared to "0".

D: Does not affect.

H: Does not affect.

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, when general register 00H contains the value 9AH (10011010B), the statement "SRA 00H" shifts the bit values in register 00H right 1 bit position. Bit zero ("0") clears the C flag and bit 7 ("1") is then shifted into the bit 6 position (bit 7 does not change). This leaves the value 0CDH (11001101B) in destination register 00H.

6.6.67 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 1 and 0 (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: Does not affect any flags.

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.

6.6.68 STOP-Stop Operation

STOP

Operation:

The STOP instruction stops both the CPU 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. An external reset operation or external interrupts release the Stop Mode. For the reset operation, the RESET pin must be held to Low level until the required oscillation stabilization interval has elapsed.

Flags: Does not affect any flags.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
	opc					1	4	7F	–	–

Example: The statement

STOP

Halts all microcontroller operations.

6.6.69 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: Sets if a "borrow" occurred; otherwise it clears.

Z: Sets if the result is "0"; otherwise it clears.

S: Sets if the result is negative; otherwise it clears.

V: Sets 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; otherwise it clears.

D: Always set to "1".

H: Clears 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	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"
```

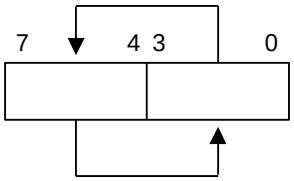
In the first example, when working register R1 contains the value 12H and when register R2 contains the value 03H, the statement "SUB R1, R2" subtracts the source value (03H) from the destination value (12H) and stores the result (0FH) in destination register R1.

6.6.70 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: Does not define.

Z: Sets if the result is "0"; otherwise it clears.

S: Sets if the result bit 7 is set; otherwise it clears.

V: Does not define.

D: Does not affect.

H: Does not affect.

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, when general register 00H contains the value 3EH (00111110B), the statement "SWAP 00H" swaps the lower and upper four bits (nibbles) in the 00H registerby leaving the value 0E3H (11100011B).

6.6.71 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 1 value. You can specify the bits to be tested 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. You can verify the zero (Z) flag to determine the result. The destination and source operands are unaffected.

Flags:

C: Does not affect.

Z: Sets if the result is "0"; otherwise it clears.

S: Sets if the result bit 7 is set; otherwise it clears.

V: Always clears to "0".

D: Does not affect.

H: Does not affect.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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, when working register R0 contains the value 0C7H (11000111B) and register R1 the value 02H (00000010B), the statement "TCM R0, R1" tests bit 1 in the destination register for a "1" value. Because the mask value corresponds to the test bit, the Z flag is set to logic 1 and can be tested to determine the result of the TCM operation.

6.6.72 TM-Test under Mask

TM dst, src

Operation: dst AND src

This instruction tests selected bits in the destination operand for a logic 0 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: Does not affect.

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

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

V: Always reset to "0".

D: Does not affect.

H: Does not affect.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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 when working register R0 contains the value 0C7H (11000111B) and register R1 the value 02H (00000010B), the statement "TM R0, R1" tests bit 1 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 0 and can be tested to determine the result of the TM operation.

6.6.73 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 that includes a fast interrupt.

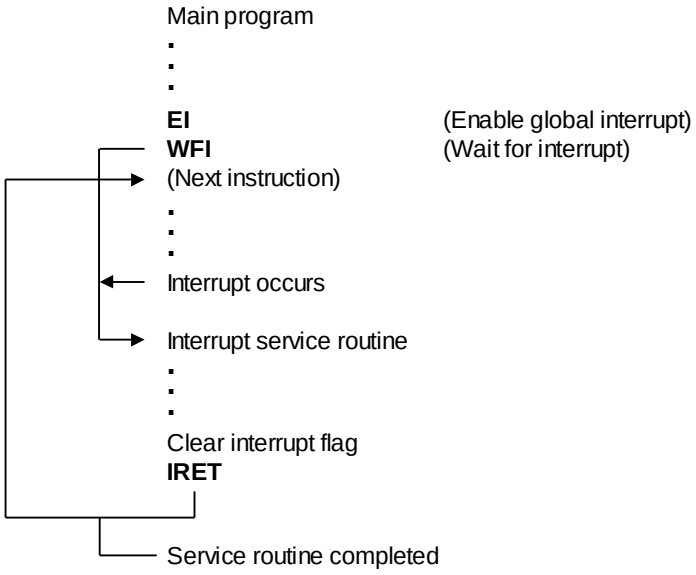
Flags: Does not affect any flags.

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:



6.6.74 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, it stores a "0" bit.

Flags:

C: Does not affect.

Z: Sets if the result is "0"; otherwise it clears.

S: Sets if the result bit 7 is set; otherwise it clears.

V: Always reset to "0".

D: Does not affect.

H: Does not affect.

Format:

						Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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 when working register R0 contains the value 0C7H and when 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

By Smart Option (0x3FH.2 in ROM), you can select internal RC oscillator, external RC oscillator, or external oscillator. An internal RC oscillator source provides 8 MHz, 4 MHz, 1 MHz, or 0.5 MHz depending on Smart Option.

An external RC oscillation source provides 4 MHz clock for S3F8S39/S3F8S35. An internal capacitor supports the RC oscillator circuit. An external crystal or ceramic oscillation source provides a maximum 12 MHz clock. The X_{IN} and X_{OUT} pin connect the oscillation source to the on-chip clock circuit.

Figure 7-1 illustrates the simplified external RC oscillator.

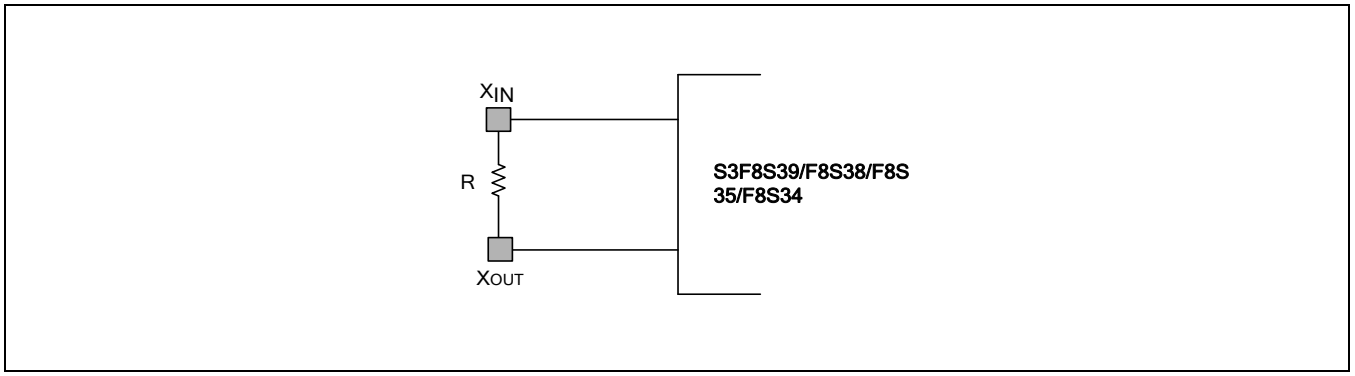


Figure 7-1 Main Oscillator Circuit (RC Oscillator with Internal Capacitor)

Figure 7-2 illustrates the simplified crystal/ceramic oscillator circuits.

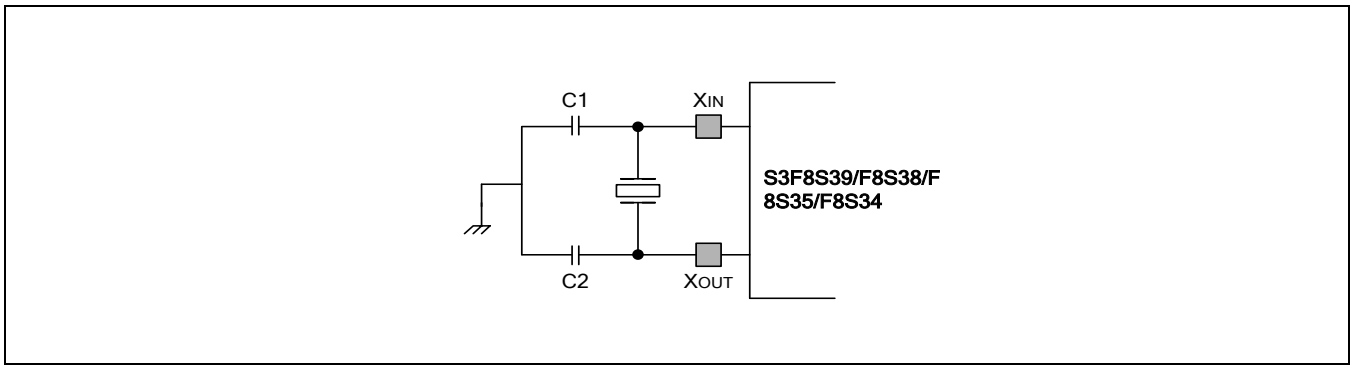


Figure 7-2 Main Oscillator Circuit (Crystal/Ceramic Oscillator)

7.2 C2.Clock Status during Power-Down Modes

The two power-down modes, Stop Mode and Idle Mode, affect clock oscillation. In:

- Stop Mode, the main oscillator "freezes", halting the CPU and peripherals. The contents of the register file and current system register values are retained. Stop Mode is released, and the oscillator started, by a reset operation or by an external interrupt with RC-delay noise filter (for S3F8S39/S3F8S35, INT0–INT7).
- Idle Mode, the internal clock signal is gated off to the CPU, but does not interrupt control and the timer. The current CPU status is preserved, including stack pointer, program counter, and flags. Data in the register file is retained. Idle Mode is released by a reset or by an interrupt (external or internally-generated).

7.3 System Clock Control Register

The system clock control register (CLKCON) is located in location D4H. It is read/write addressable and has these functions:

- Oscillator IRQ wake-up enable/disable (CLKCON.7)
- Oscillator frequency divide-by value: Non-divided, 2, 8, or 16 (CLKCON.4 and CLKCON.3)

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

After a reset, the external interrupt oscillator wake-up function is enabled, and the fOSC/16 (the slowest clock speed) is selected as the CPU clock. If necessary, you can increase the CPU clock speed to fOSC, fOSC/2, fOSC/8 or fOSC/16.

Figure 7-3 illustrates the system clock control register.

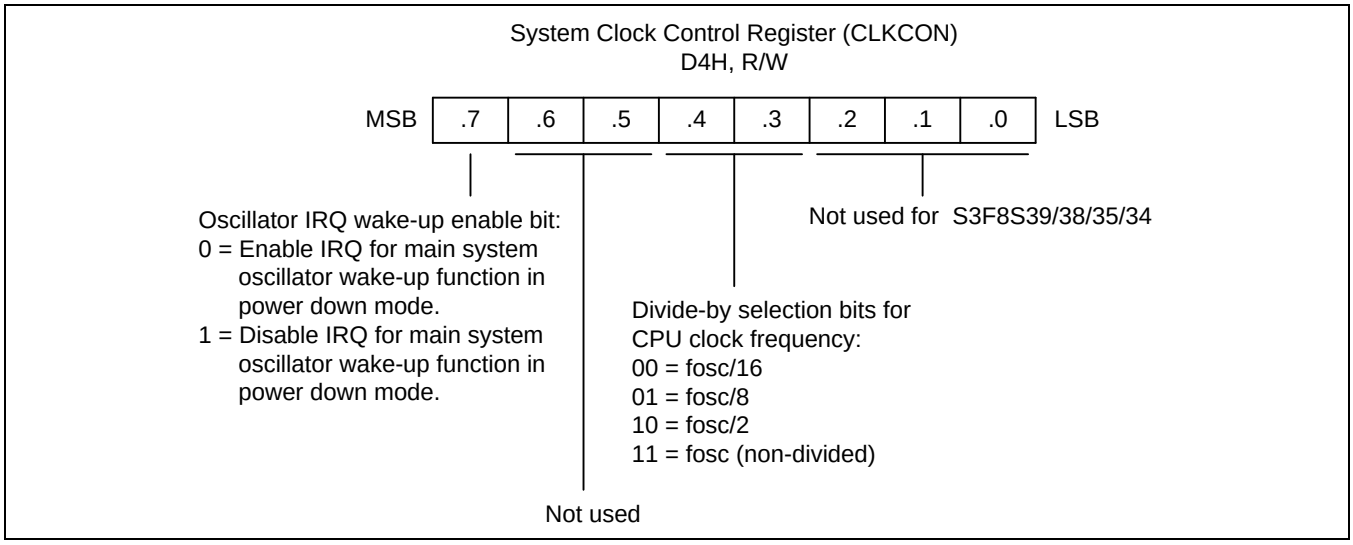


Figure 7-3 System Clock Control Register (CLKCON)

7.4 Internal Oscillator Calibration Control Register

The internal oscillator calibration control register, OSCCALCON, is located in location 18H, Page8 of RAM address. It is read/write addressable and other function is

enable/disable OSCCALDATA register written.

After a reset, the OSCCALDATA register written is disabled, because the value of OSCCALCON is "other values".

If necessary, you can write the OSCCALDATA register by setting the value of OSCCALCON to "10100101B".

Figure 7-4 illustrates the internal oscillator calibration control register.

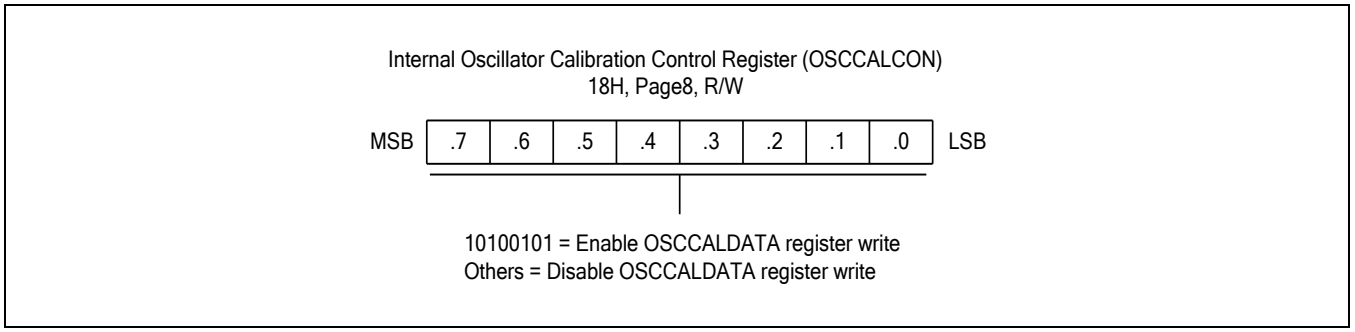


Figure 7-4 Internal Oscillator Calibration Control Register (OSCCALCON)

7.5 Internal Oscillator Calibration Data Register

The internal oscillator calibration data register (OSCCALDATA) is in location 17H, Page8 of RAM address. It is read/write addressable and its other function is to calibrate the internal oscillator.

Whenever OSCCALDATA.6 = ‘1’, such as after a reset, a factory calibrated value is loaded to OSCCALDATA.5-.0. This value is factory calibrated at VDD = 5 V, 25ºC. If used under a different condition, you can calibrate the internal oscillator by writing different values to the OSCCALDATA register. Setting OSCCALDATA.5-.0 to a lower value (minimum ‘000000’), results in a higher frequency, while setting the OSCCALDATA.5-.0 to a higher value (maximum ‘111111’), results in a lower frequency. To write OSCCALDATA.5-.0, perform the following steps:

1. Write "10100101" to the OSCCALCON register to enable OSCCALDATA register data writing.
2. AND OSCCALDATA with “10111111’ to clear OSCCALDATA.6 to ‘0’.
3. Write the OSCCALDATA register with the desired value ensuring OSCCALDATA.6 remains ‘0’.

Figure 7-5 illustrates the internal oscillator calibration data register.

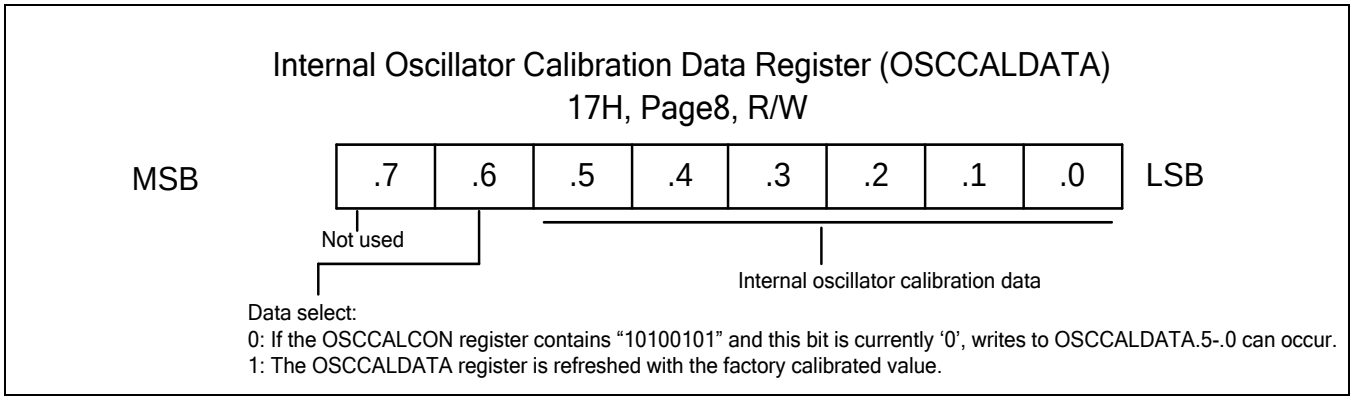


Figure 7-5 Internal Oscillator Calibration Data Register (OSCCALDATA)

7.6 Ring Oscillator Control Register

The Ring oscillator control register (ROSCCON) is located at Page8 of RAM address, at address 0CH.

It is read/write addressable and other function is ring oscillator enable bit.

Set ROSCCON.7 to "1" to enable the ring oscillator and also set ring oscillator as STOP wake-up time's clock source. Set ROSCCON.7 to '0' to disable the ring oscillator.

Figure 7-6 illustrates the ring oscillator control register.

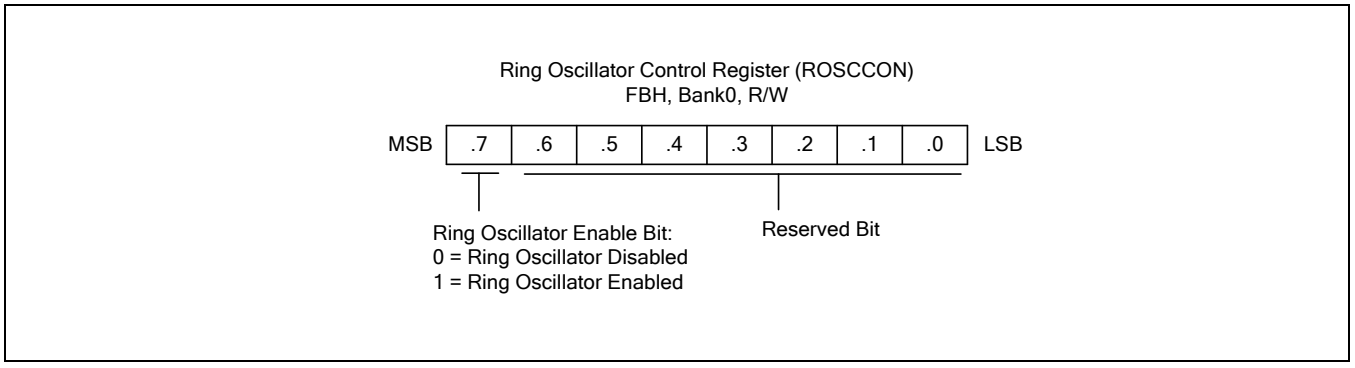


Figure 7-6 Ring Oscillator Control Register (ROSCCON)

7.7 STOP Control Register

The STOP control register (STPCON) is located in the bank 1 of set1, address FBH.

It is read/write addressable and other function is

- 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 7-7 illustrates the STOP control register.

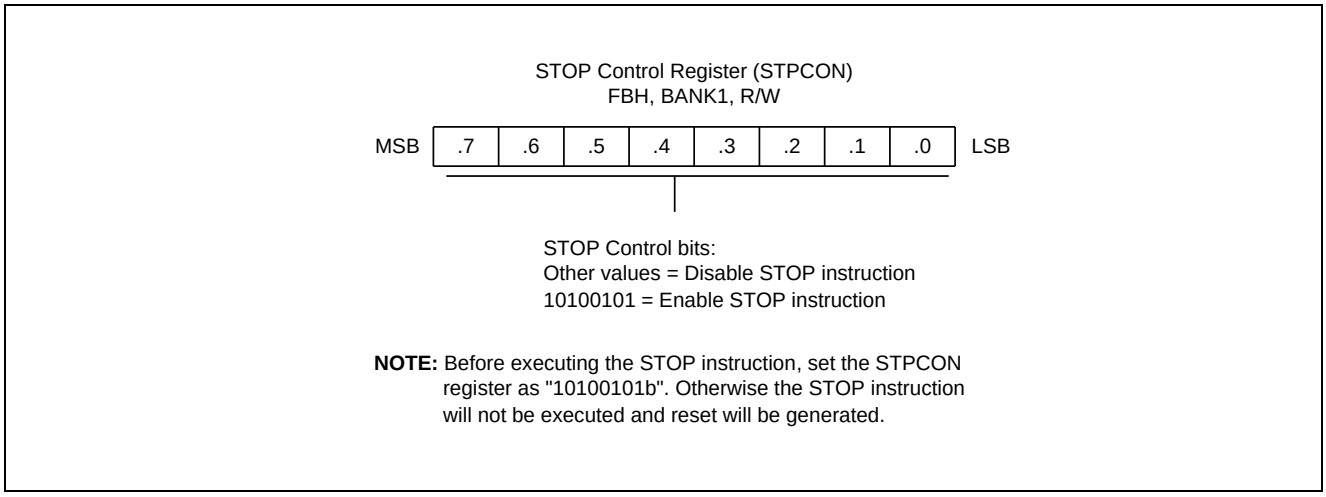
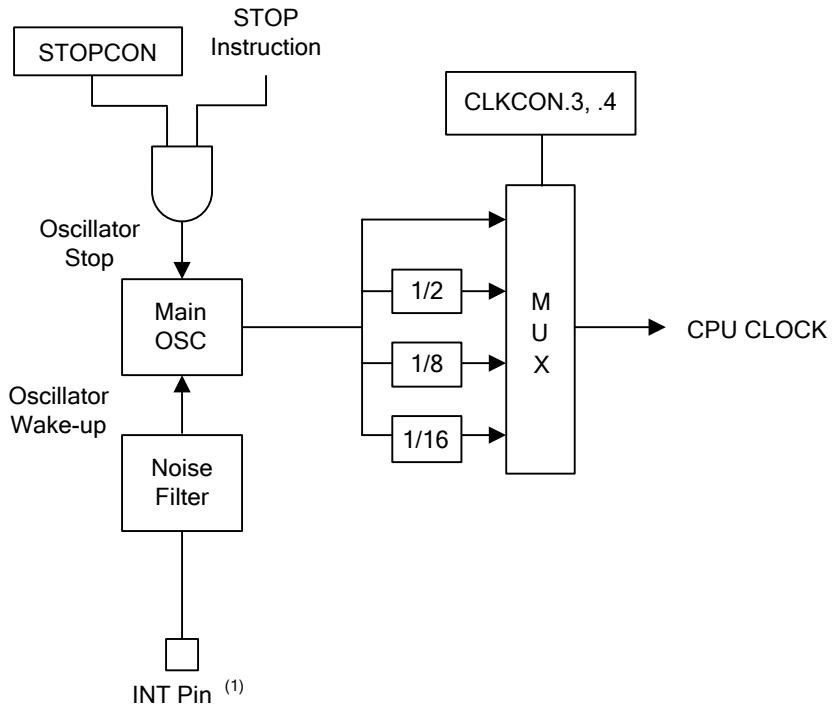


Figure 7-7 STOP Control Register (STPCON)

Figure 7-8 illustrates the circuit diagram of a system clock.



NOTES:

1. An external interrupt with an RC-delay noise filter (for the S3F8S39/F8S38/F8S35/F8S34 INT0-7) is fixed to release stop mode and "wake up" the main oscillator.
2. Because the S3F8S39/F8S38/F8S35/F8S34 has no subsystem clock, the 3-bit CLKCON signature code (CLKCON.2-CLKCON.0) is no meaning.

Figure 7-8 System Clock Circuit Diagram

8 RESET and Power-Down

8.1 Overview

By Smart Option (3FH.7 in ROM), you can select internal reset (LVR) or external RESET.

You can reset S3F8S39/S3F8S35 in four ways. They are:

- By external power-on-reset
- By the external nRESET input pin pulled low
- By digital basic timer timing out
- By Low Voltage Reset (LVR)

During an external power-on reset, the voltage at V_{DD} is High level and the nRESET pin is forced to Low level. The nRESET signal is an input through a schmitt trigger circuit where it is then synchronized to CPU clock. This brings the S3F8S39/S3F8S35 into a known operating status. To ensure correct start-up, you should take care that it does not release nRESET signal before the V_{DD} level is sufficient to allow MCU operation at the chosen frequency.

The nRESET pin must be held as low level for a minimum time interval after the power supply comes within tolerance to allow time for internal CPU clock oscillation to stabilize.

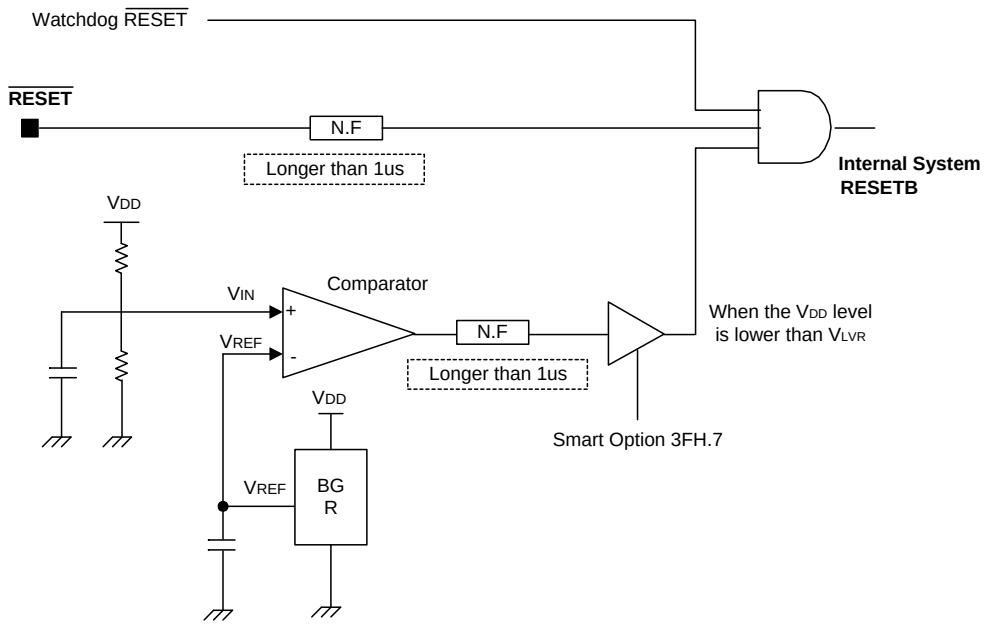
When a reset occurs during normal operation (with both V_{DD} and nRESET at High level), it forces the signal at the nRESET pin to low and the reset operation starts. Next, it sets all system and peripheral control registers to their default hardware reset values (Refer to *Table 8-1* for more information.)

The MCU provides a watchdog timer function to ensure graceful recovery from software malfunction. If watchdog timer is not refreshed before an end-of-counter condition (overflow) is reached, the internal reset will be activated.

The on-chip Low Voltage Reset features static reset when supply voltage is below a reference value (Typical 1.9, 2.3, 3.0, 3.9 V). With this feature, you can remove the external reset circuit while keeping the application safety. As long as the supply voltage is below the reference value, there is an internal and static reset. The MCU can start only when the supply voltage rises over the reference value.

When you calculate power consumption, remember that you should add a static current of LVR circuit a CPU operating current in any operating modes such as STOP, IDLE, and normal RUN mode.

Figure 8-1 illustrates the low voltage reset circuit.



NOTES:
1. The target of voltage detection level is the one you selected at smart option 3FH.
2. BGR is Band Gap voltage Reference

Figure 8-1 Low Voltage Reset Circuit

NOTE: To program the duration of the oscillation stabilization interval, you should 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.

8.1.1 MCU Initialization Sequence

These are the sequence of events that occur during a Reset operation:

- Disables all interrupts.
- Enables watchdog function (basic timer).
- Sets ports 0-4 to input mode
- Peripheral control and data registers reset to their initial values (Refer to *Table 8-1* for more information.)
- Loads the program counter with the ROM reset address, 0100H or other values set by the Smart Option.
- When the programmed oscillation stabilization time interval has elapsed, the address stored in the first and second bytes of reset address in ROM is fetched and executed.

Figure 8-2 illustrates the block diagram of Reset.

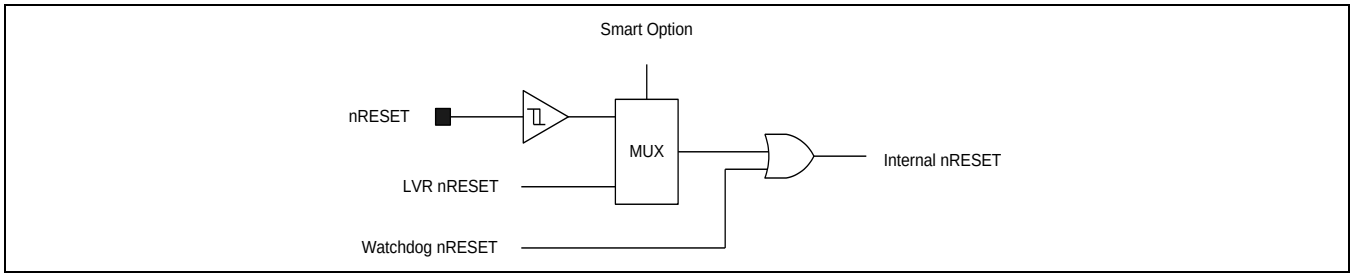


Figure 8-2 Reset Block Diagram

Figure 8-3 illustrates the timing for S3F8S39/S3F8S35 after Reset.

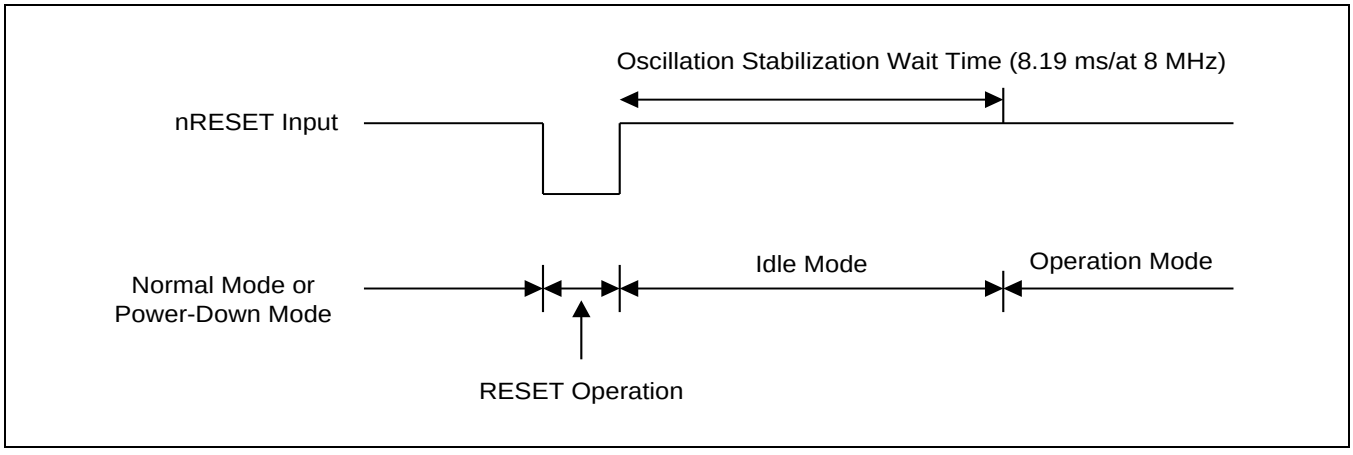


Figure 8-3 Timing for S3F8S39/S3F8S35 after reset

8.2 Power-Down Modes

This section includes:

- Stop Mode
- Idle Mode

8.2.1 Stop Mode

The instruction STOP (opcode 7FH) invokes Stop Mode). In Stop Mode, it halts the operation of the CPU and all peripherals. That is, the on-chip main oscillator stops and the supply current reduces to less than 2 μA when it enables Low Voltage Reset (LVR). All system functions are halted when the clock "freezes", but it retains the data stored in the internal register. Stop Mode can be released in one of two ways:

- nRESET signal
- External interrupt

NOTE: Before executing the STOP instruction, you should set STPCON register to "10100101B".

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 then reset to their default values and the contents of all data registers are retained. A Reset operation automatically selects a slow clock (f_x/16) because it clears CLKCON.3 and CLKCON.4 to "00B". After the oscillation stabilization interval has elapsed, the CPU executes the system initialization routine by fetching the 16-bit address stored in the first and second bytes of reset address (configured by the Smart Option) in ROM.

8.2.1.2 Using an External Interrupt to Release Stop Mode

You can use external interrupts with an RC-delay noise filter circuit to release Stop Mode. (You cannot use clock-related external interrupts.). External interrupts INT0–INT7 in the S3F8S39/S3F8S35 interrupt structure meet this criterion.

Note that when an external interrupt releases the Stop Mode, it does not change the values in system and peripheral control registers. When you use an interrupt to release Stop Mode, it does not change the values of CLKCON.3 and CLKCON.4 register. It uses the currently selected clock value. Therefore, you can program the duration of the oscillation stabilization interval by putting the appropriate value to BTCON register before entering Stop Mode.

The external interrupt is serviced when the Stop Mode release occurs. After return from the interrupt service routine (IRET), the instruction next to STOP is immediately executed.

8.2.2 Idle Mode

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

There are two ways to release IDLE mode:

1. Execute a Reset. It resets all system and peripheral control registers to their default values and it retains the contents of all data registers. The Reset automatically selects a slow clock (fxx/16) because it clears CLKCON.3 and CLKCON.4 to "00B". If it masks the interrupts, 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, it does not change the values of CLKCON.3 and CLKCON.4 register, and it uses the currently selected clock value. The interrupt is then serviced. After return from the interrupt service routine (IRET), the instruction next to STOP is immediately executed.

NOTE:

1. Only external interrupts that are not clock-related can be used to release Stop Mode. To release IDLE mode, however, ou can use any type of interrupt (that is, internal or external).
2. Before entering the Stop Mode or IDLE mode, you must disable ADC. Otherwise, the STOP or IDLE current will be increased significantly.

8.3 Hardware Reset Values

Hardware reset value is 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 lists the S3F8S39/S3F8S35 Set 1 registers values after reset.

Table 8-1 S3F8S39/S3F8S35 Set 1 Registers Values after Reset

Register name	Mnemonic	Address Hex	R/W	Reset Value (Bit)							
				7	6	5	4	3	2	1	0
A/D Converter Data Register (High Byte)	ADDATAH	D0H	R	x	x	x	x	x	x	x	x
A/D Converter Data Register (Low Byte)	ADDATAL	D1H	R	–	–	–	–	–	–	x	x
A/D Converter Control Register	ADCON	D2H	R/W	0	0	0	0	0	0	0	0
Basic Timer Control Register	BTCON	D3H	R/W	0	0	0	0	0	0	0	0
System Clock control Register	CLKCON	D4H	R/W	0	–	–	0	0	–	–	–
System Flags Register	FLAGS	D5H	R/W	x	x	x	x	x	x	0	0
Register Pointer 0	RP0	D6H	R/W	1	1	0	0	0	–	–	–
Register Pointer 1	RP1	D7H	R/W	1	1	0	0	1	–	–	–
Stack Pointer (High byte)	SPH	D8H	R/W	x	x	x	x	x	x	x	x
Stack Pointer (Low byte)	SPL	D9H	R/W	x	x	x	x	x	x	x	x
Instruction Pointer (High Byte)	IPH	DAH	R/W	x	x	x	x	x	x	x	x
Instruction Pointer (Low Byte)	IPL	DBH	R/W	x	x	x	x	x	x	x	x
Interrupt Request Register	IRQ	DCH	R	0	0	0	0	0	0	0	0
Interrupt Mask Register	IMR	DDH	R/W	x	x	x	x	x	x	x	x
System Mode Register	SYM	DEH	R/W	–	–	–	x	x	x	0	0
Register Page Pointer	PP	DFH	R/W	0	0	0	0	0	0	0	0

NOTE: –: Not mapped or not used, x: Undefined

Table 8-2 lists the system and peripheral control registers Set1 Bank 0.

Table 8-2 System and Peripheral Control Registers (Set 1, Bank 0)

Register Name	Mnemonic	Address	R/W	Bit Values After Reset							
–	–	Hex	–	7	6	5	4	3	2	1	0
Timer A Counter Register	TACNT	E0H	R	0	0	0	0	0	0	0	0
Timer A Data Register	TADATA	E1H	R/W	1	1	1	1	1	1	1	1
Timer A Control Register	TACON	E2H	R/W	0	0	0	0	0	0	0	0
Timer A Prescaler	TAPS	E3H	R/W	0	–	–	–	0	0	0	0
Locations E4H to E7H are not mapped											
Timer 0 Counter Register (High Byte)	T0CNTH	E8H	R	0	0	0	0	0	0	0	0
Timer 0 Counter Register (Low Byte)	T0CNTL	E9H	R	0	0	0	0	0	0	0	0
Timer 0 Data Register (High Byte)	T0DATAH	EAH	R/W	1	1	1	1	1	1	1	1
Timer 0 Data Register (Low Byte)	T0DATAL	EBH	R/W	1	1	1	1	1	1	1	1
Timer 0 Control Register	T0CON	ECH	R/W	0	0	0	0	0	0	0	0
Timer 0 Prescaler	T0PS	EDH	R/W	–	–	–	–	0	0	0	0
Timer 1 Counter Register (High Byte)	T1CNTH	EEH	R	0	0	0	0	0	0	0	0
Timer 1 Counter Register (Low Byte)	T1CNTL	EFH	R	0	0	0	0	0	0	0	0
Timer 1 Data Register (High Byte)	T1DATAH	F0H	R/W	1	1	1	1	1	1	1	1
Timer 1 Data Register (Low Byte)	T1DATAL	F1H	R/W	1	1	1	1	1	1	1	1
Timer 1 Control Register	T1CON	F2H	R/W	0	0	0	0	0	0	0	0
Timer 1 Prescaler	T1PS	F3H	R/W	0	–	–	–	0	0	0	0
Timer 2 Counter Register (High Byte)	T2CNTH	F4H	R	0	0	0	0	0	0	0	0
Timer 2 Counter Register (Low Byte)	T2CNTL	F5H	R	0	0	0	0	0	0	0	0
Timer 2 Data Register (High Byte)	T2DATAH	F6H	R/W	1	1	1	1	1	1	1	1
Timer 2 Data Register (Low Byte)	T2DATAL	F7H	R/W	1	1	1	1	1	1	1	1
Timer 2 Control Register	T2CON	F8H	R/W	0	0	0	0	0	0	0	0
Timer 2 Prescaler	T2PS	F9H	R/W	0	–	–	–	0	0	0	0
Stop Wake-Up Timer Control Register	SWTCON	FAH	R/W	0	–	0	–	0	0	0	0
Ring oscillator Control Register	ROSCCON	FBH	R/W	0	0	0	0	0	0	0	0
Locations FCH are not mapped											
Basic timer counter	BTCNT	FDH	R	0	0	0	0	0	0	0	0
Location FEH is not Mapped											
Interrupt Priority Register	IPR	FFH	R/W	x	x	x	x	x	x	x	x

NOTE: –: Not mapped or not used, x: Undefined.

Table 8-3 lists the system and peripheral control registers Set 1 Bank 1.

Table 8-3 System and Peripheral Control Registers (Set 1, Bank 1)

Register name	Mnemonic	Address Hex	R/W	Reset Value (Bit)							
				7	6	5	4	3	2	1	0
–	P0CONH	E0H	R/W	0	0	0	0	0	0	0	0
Port 0 Control Register (Low Byte)	P0CONL	E1H	R/W	0	0	0	0	0	0	0	0
Port 0 Pull-up Resistor Enable Register	P0PUR	E2H	R/W	0	0	0	0	0	0	0	0
Port 1 Control Register (Low Byte)	P1CONL	E3H	R/W	–	–	–	–	0	0	0	0
Port 2 Control Register (High Byte)	P2CONH	E4H	R/W	0	0	0	0	0	0	0	0
Port 2 Control Register (Low Byte)	P2CONL	E5H	R/W	0	0	0	0	0	0	0	0
Port 2 Interrupt Control Register	P2INT	E6H	R/W	0	0	0	0	0	0	0	0
Port 2 Pull-up Resistor Enable Register	P2PUR	E7H	R/W	0	0	0	0	0	0	0	0
Port 3 Control Register (High Byte)	P3CONH	E8H	R/W	0	0	0	0	0	0	0	0
Port 3 Control Register (Low Byte)	P3CONL	E9H	R/W	0	0	0	0	0	0	0	0
Port 3 Interrupt Control Register	P3INT	EAH	R/W	0	0	0	0	0	0	0	0
Port 3 Pull-up Resistor Enable Register	P3PUR	EBH	R/W	0	0	0	0	0	0	0	0
Port 3 N-Channel Open-drain Mode Register	PNE3	ECH	R/W	0	0	0	0	0	0	0	0
Port 0 Data Register	P0	EDH	R/W	0	0	0	0	0	0	0	0
Port 1 Data Register	P1	EEH	R/W	–	–	–	–	–	–	0	0
Port 2 Data Register	P2	EFH	R/W	0	0	0	0	0	0	0	0
Port 3 Data Register	P3	F0H	R/W	0	0	0	0	0	0	0	0
External Interrupt Pending Register	PINTPND	F1H	R/W	0	0	0	0	0	0	0	0
SPI Control Register	SPICON	F2H	R/W	0	0	0	0	0	0	0	0
SPI Status Register	SPISTAT	F3H	R/W	0	0	0	-	-	-	-	0
SPI Data Register	SPIDATA	F4H	R/W	1	1	1	1	1	1	1	1
UART0 Control Register (High Byte)	UART0CONH	F5H	R/W	0	0	0	0	0	0	0	0
UART0 Control Register (Low Byte)	UART0CONL	F6H	R/W	0	0	0	0	0	0	0	0
UART0 Data Register	UDATA0	F7H	R/W	x	x	x	x	x	x	x	x
UART0 Baud Rate Data Register	BRDATA0	F8H	R/W	1	1	1	1	1	1	1	1
UART1 Control Register (High Byte)	UART1CONH	F9H	R/W	0	0	0	0	0	0	0	0
UART1 Control Register (Low Byte)	UART1CONL	FAH	R/W	0	0	0	0	0	0	0	0
UART1 Data Register	UDATA1	FBH	R/W	x	x	x	x	x	x	x	x
UART1 Baud Rate Data Register	BRDATA1	FCH	R/W	1	1	1	1	1	1	1	1
STOP Control Register	STPCON	FDH	R/W	0	0	0	0	0	0	0	0
Watch Timer Control Register	WTCON	FEH	R/W	0	0	0	0	0	0	0	0

Register name	Mnemonic	Address Hex	R/W	Reset Value (Bit)							
				7	6	5	4	3	2	1	0
LVD Control Register	LVDCON	FFH	R/W	1	–	0	0	0	–	0	0

NOTE: – : Not mapped or not used, read "0"; x: Undefined

Table 8-4 lists the system and peripheral control registers page 8.

Table 8-4 System and Peripheral Control Registers Page 8

Register Name	Mnemonic	Address Hex	R/W	nRESET Value (Bit)							
				7	6	5	4	3	2	1	0
Reset ID	RESETID	00H	R/W	Refer to detail description							
Flash Memory Sector Address Register (High Byte)	FMSECH	01H	R/W	0	0	0	0	0	0	0	0
Flash Memory Sector Address Register (Low Byte)	FMSECL	02H	R/W	0	0	0	0	0	0	0	0
Flash Memory User Programming Enable Register	FMUSR	03H	R/W	0	0	0	0	0	0	0	0
Flash Memory Control Register	FMCON	04H	R/W	0	0	0	0	0	–	–	0
IIC Control Register	ICCR	05H	R/W	0	0	0	0	1	1	1	1
IIC Status Register	ICSR	06H	R/W	0	0	0	0	0	0	0	0
IIC Data Shift Register	IDSR	07H	R/W	x	x	x	x	x	x	x	x
IIC Address Register	IAR	08H	R/W	x	x	x	x	x	x	x	x
16-Bit Timer B Control Register	TBCON	09H	R/W	0	0	0	0	0	0	0	0
Timer B Reference Data Register 0 (High Byte)	TBDATA0H	0AH	R/W	1	1	1	1	1	1	1	1
Timer B Reference Data Register 0 (Low Byte)	TBDATA0L	0BH	R/W	1	1	1	1	1	1	1	1
Timer B Reference Data Register 1 (High Byte)	TBDATA1H	0CH	R/W	1	1	1	1	1	1	1	1
Timer B Reference Data Register 1 (Low Byte)	TBDATA1L	0DH	R/W	1	1	1	1	1	1	1	1
Timer B Trigger Control Register	TBTRG	0EH	R/W	0	0	0	0	0	0	0	0
Location 0FH to16H are not mapped											
Internal oscillator Calibration Data Register	OSCCALDATA	23	17H	–	(NOTE)						
Internal oscillator Calibration Control Register	OSCCALCON	24	18H	0	0	0	0	0	0	0	0

NOTE: The reset value of OSCCALDATA register is factory calibrated. Different chips may have different reset value.

9 I/O Ports

9.1 Overview

The S3F8S39/S3F8S35 microcontroller has four bit-programmable I/O ports P0, P1, P2, P3, which gives a total of 26 I/O pins. You can flexibly configure each port to meet application design requirements. The CPU accesses ports by directly writing or reading port registers. It does not require special I/O instructions.

Table 9-1 gives a general overview of the S3F8S39/S3F8S35 I/O port functions.

Table 9-1 S3F8S39/S3F8S35 Port Configuration Overview

Port	Configuration Options
0	I/O port with bit-programmable pins. You can configure it to Schmitt trigger input or push-pull output mode. Pull-up resistors can be assigned by software. You can also assign pins individually as alternative function pins.
1	I/O port with bit-programmable pins. You can configure it to Schmitt trigger input or push-pull output mode. Pull-up resistors can be assigned by software. You can also assign pins individually as alternative function pins.
2	I/O port with bit-programmable pins. You can configure it to Schmitt trigger input mode, push-pull output mode or open-drain output mode. You can also assign pins individually as alternative function pins.
3	I/O port with bit-programmable pins. You can configure it to Schmitt trigger input mode, push-pull output mode or open-drain output mode. You can also assign pins individually as alternative function pins.

9.2 Port Data Registers

Table 9-2 gives you an overview of register locations of all four S3F8S39/S3F8S35 I/O port data registers. Data registers for ports 0, 1, 2 and 3 have the general format as shown in Figure 9-1.

Table 9-2 describes the port data register summary.

Table 9-2 Port Data Register Summary

Register Name	Mnemonic	Decimal	Hex	Location	R/W
Port 0 data register	P0	237	EDH	Set1, Bank1	R/W
Port 1 data register	P1	238	EEH	Set1, Bank1	R/W
Port 2 data register	P2	239	EFH	Set1, Bank1	R/W
Port 3 data register	P3	240	F0H	Set1, Bank1	R/W

9.3 Port 0

Port 0 is an 8-bit I/O Port that you can use in two ways:

- General-purpose I/O
- Alternative function

You can access Port 0 directly by writing or reading the port 0 data register, P0 at location EDH, Set1 Bank1.

9.3.1 Port 0 Control Register

You can configure Port 0 pins individually by bit-pair settings in two control registers location: P0CONH (high byte, E0H, Set1 Bank1) and P0CONL (low byte, E1H, Set1 Bank1).

When you select output mode, a push-pull circuit is configured. Many different selections are available:

- Schmitt trigger input mode
- Output mode(Push-pull or Open-drain)
- Alternative function: BUZ output-BUZ
- Alternative function: ADC input-ADC0–ADC7

Figure 9-1 illustrates the port 0 control register high byte.

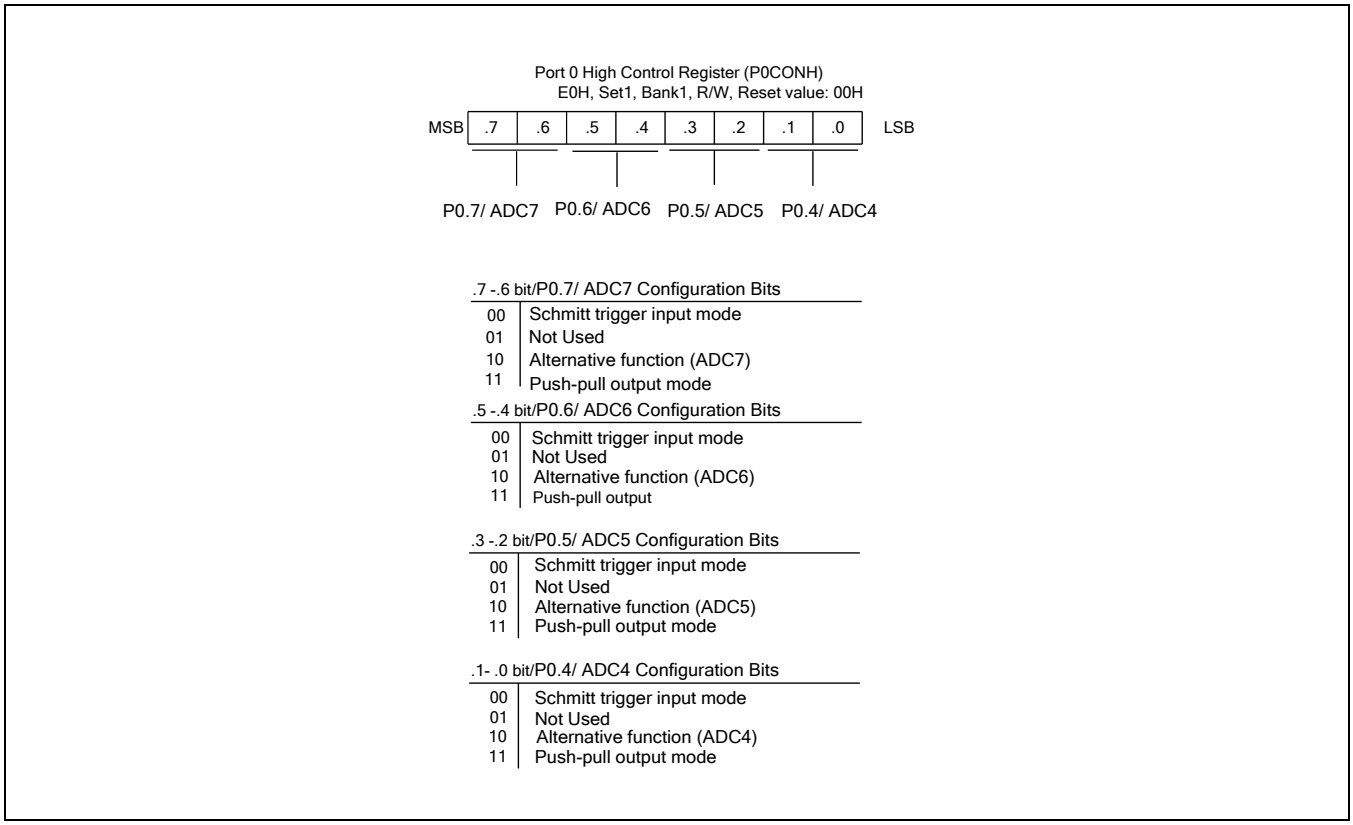


Figure 9-1 Port 0 Control Register High Byte (P0CONH)

Figure 9-2 illustrates the port 0 control register low byte.

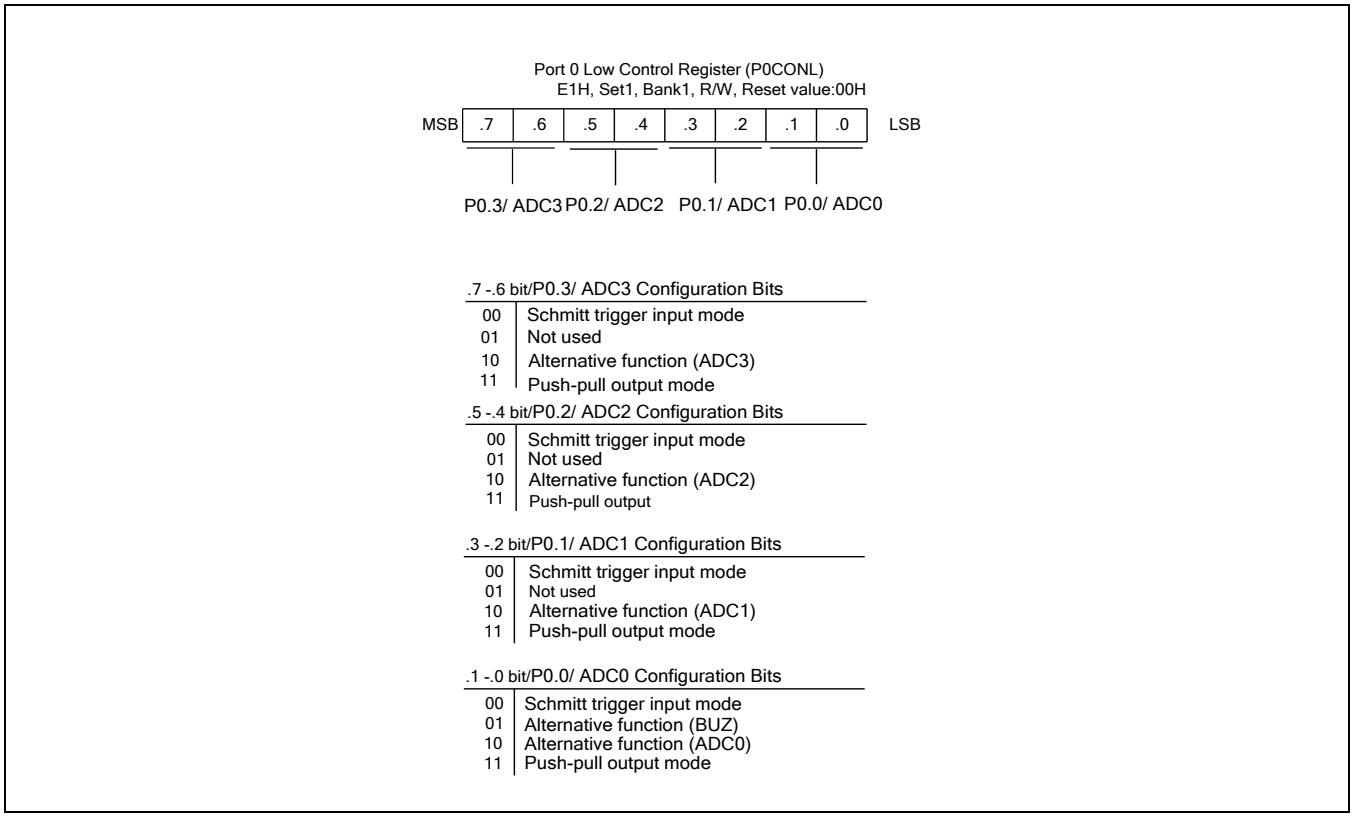


Figure 9-2 Port 0 Control Register Low Byte (P0CONL)

Figure 9-3 illustrates the pull-up resistor enable register.

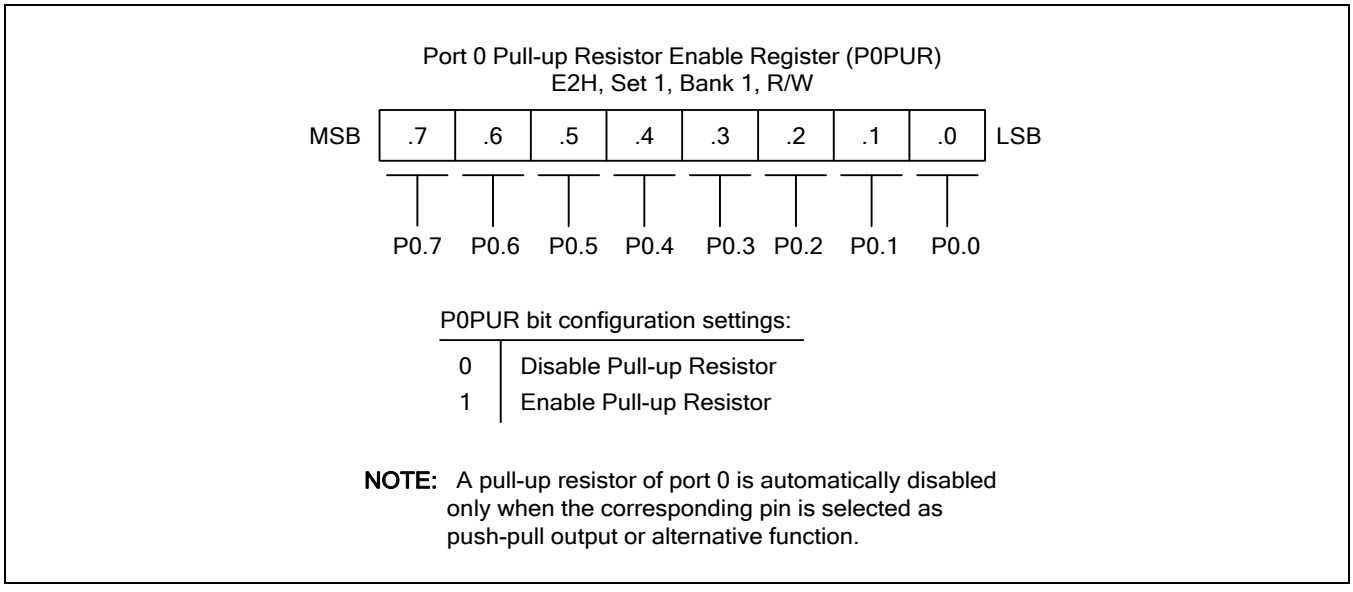


Figure 9-3 Pull-up Resistor Enable Register (P0PUR)

9.4 Port 1

Port 1 is a 2-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 EEH in set 1, and bank1. P1.0–P1.1 can serve as Schmitt trigger input (with or without pull-ups), push-pull outputs and alternative functions ADC8–ADC9.

9.4.1 Port 1 Control Register Low Byte (P1CONL)

Port 1 has one 8-bit control registers: P1CONL for P1.1–P1.0. A reset clears the P1CONL registers to "00H", configuring all pins to Schmitt trigger input mode. You use control registers settings to select Schmitt trigger input or output mode, enable pull-up resistors, select push-pull mode, and enable the alternative functions.

When programming the port, remember that any alternative peripheral I/O function you configure using the port 1 control registers should also be enabled in the associated peripheral module.

Figure 9-4 illustrates the port 1 control register low byte.

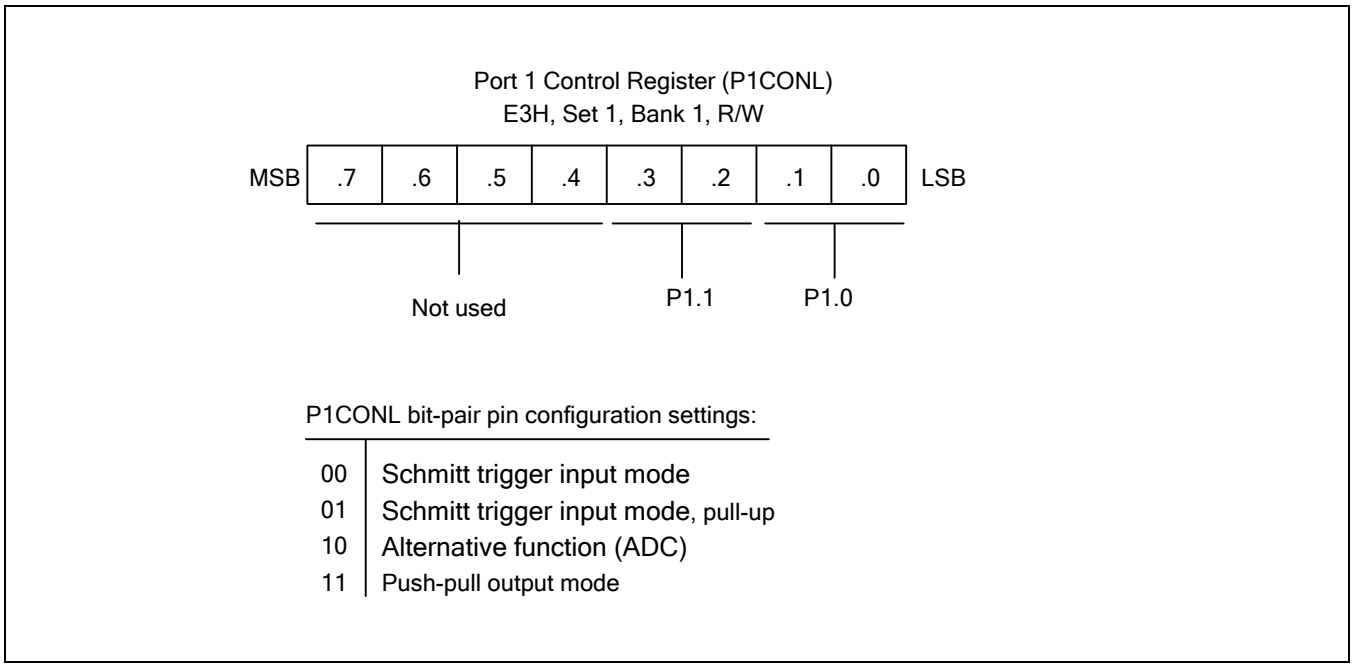


Figure 9-4 Port 1 Control Register Low Byte

9.5 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 EEH in set 1, bank 1. P2.0–P2.7 can serve as inputs (with or without pull-ups) and outputs (push pull or open-drain). For P2.7–P2.0, you can configure these alternative functions:

- Low-byte pins (P2.0–P2.3): T0OUT, TBOUT, TACLK, TACAP, TAOUT, and INT0–INT3
- High-byte pins (P2.4–P2.7): TxD1, RxD1, TxD0, and RxD0

9.5.1 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 Schmitt trigger input mode. You use control registers settings to select input or output mode, enable pull-up resistors, select push-pull or open drain output mode, and enable the alternative functions.

When programming the port, remember that any alternative peripheral I/O function you configure using the port 2 control registers should also be enabled in the associated peripheral module.

9.5.2 Port 2 Pull-Up Resistor Enable Register (P2PUR)

Using the port 2 pull-up resistor enable register, P2PUR (E7H, set1, bank1), you can configure pull-up resistors to individual port 2 pins.

Figure 9-5 illustrates the port 2 high-byte control register.

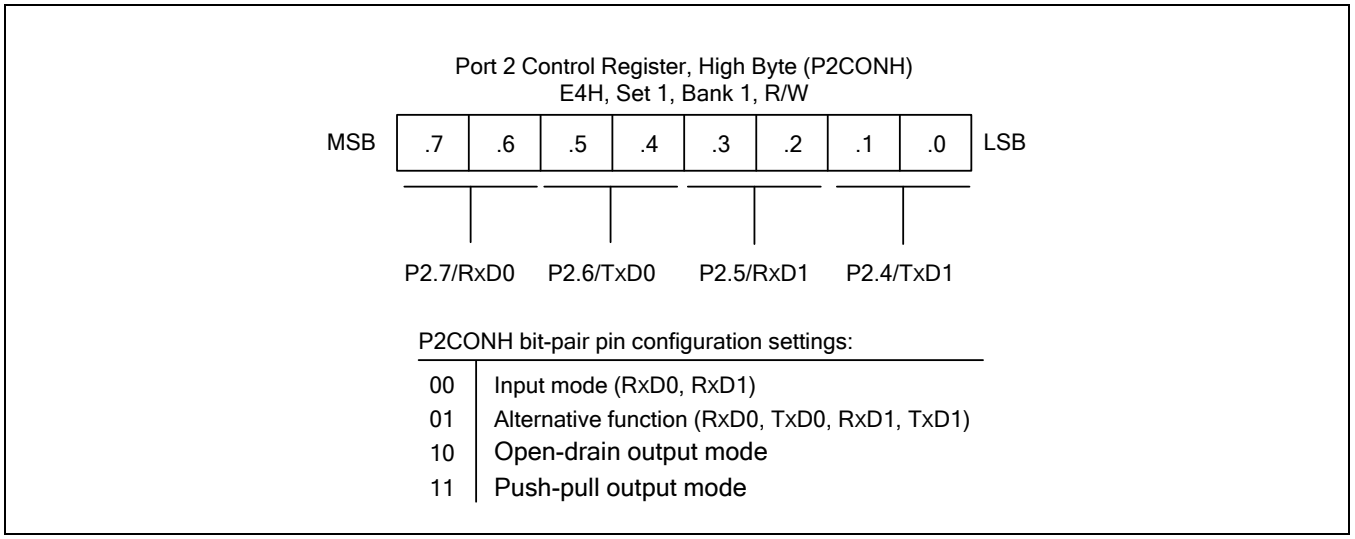


Figure 9-5 Port 2 High-Byte Control Register (P2CONH)

Figure 9-6 illustrates the port 2 low-byte control register.

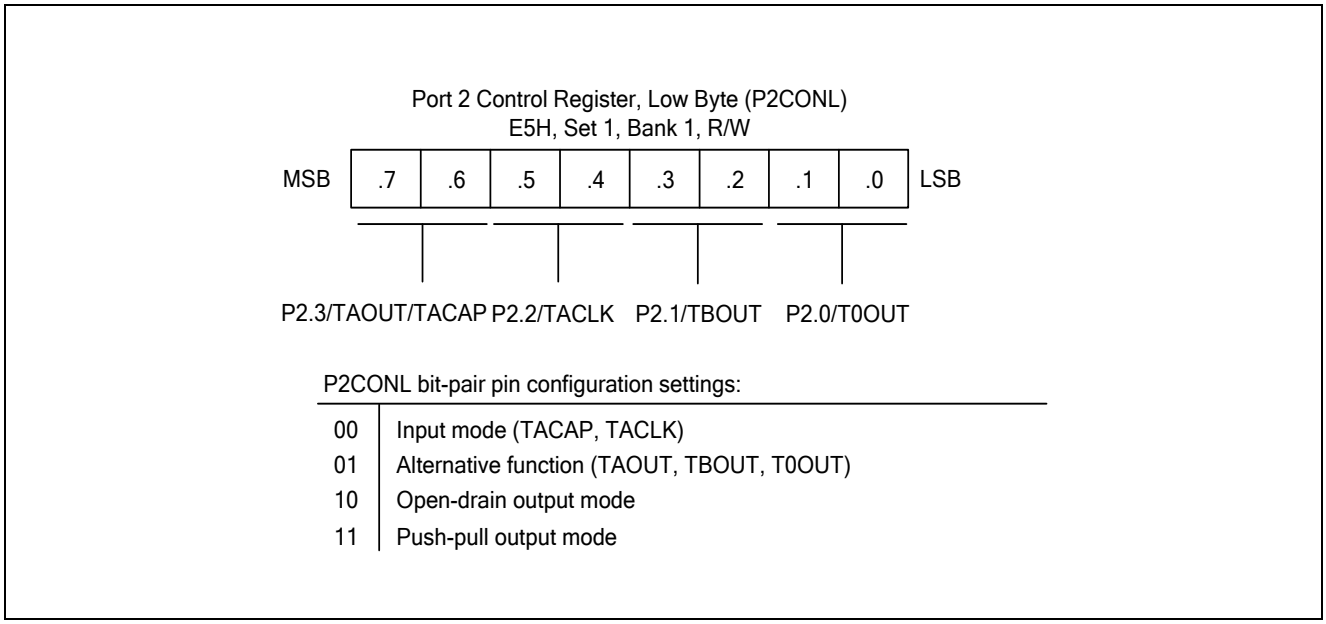


Figure 9-6 Port 2 Low-Byte Control Register (P2CONL)

Figure 9-7 illustrates the port 2 interrupt control register.

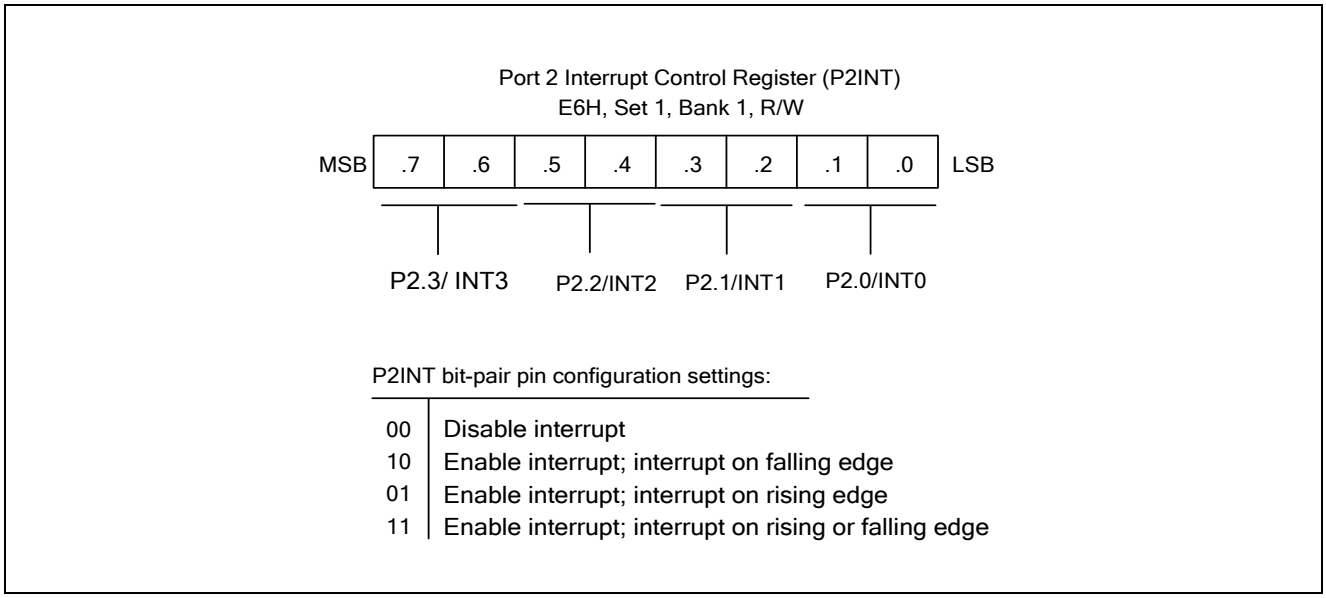


Figure 9-7 Port 2 Interrupt Control Register (P2INT)

Figure 9-8 illustrates the port 2 pull-up resistor enable register.

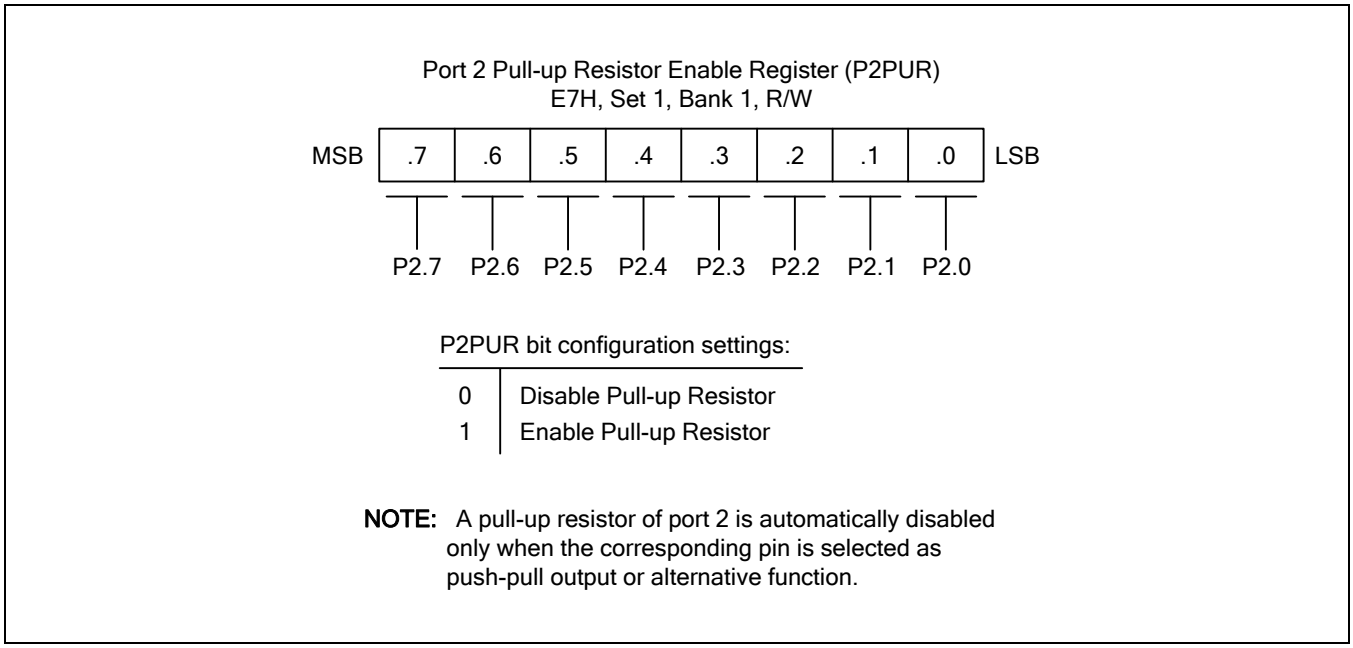


Figure 9-8 Port 2 Pull-Up Resistor Enable Register (P2PUR)

9.6 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 F0H in set 1, bank 1. P3.0–P3.7 can serve as inputs (with or without pull-ups), and outputs (push pull or open-drain). For the P3.7–P3.0, you can configure the alternative functions:

- Low-byte pins (P3.0–P3.3): MISO, MOSI, SCK, NSS, and ADC10–ADC13
- High-byte pins (P3.4–P3.7): INT4/T2OUT/ T2CAP/ADC14, SDA, INT5/T2CLK/T1OUT/ADC15, INT6/SCL, and INT7/T1CAP/SDA

9.6.1 Port 3 Control Register (P3CONH, P3CONL)

Port 3 has two 8-bit control registers: P3CONH for P3.4-P3.7 and P3CONL for P3.0-P3.3. A reset clears the P3CONH and P3CONL 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, remember that any alternative peripheral I/O function you configure using the port 3 control registers should also be enabled in the associated peripheral module.

9.6.2 Port 3 Interrupt Enable and Pending Registers

To process external interrupts at the port 3 pins, the additional control registers are provided: The port 3 interrupt enable register P3INT (EAH, set 1, bank 1) and the port 3 interrupt pending register PINTPND (F1H, set 1, bank 1).

The port 3 interrupt pending register PINTPND lets you check for interrupt pending conditions and clears the pending condition when the interrupt service routine has been initiated. The application program detects interrupt requests by polling the PINTPND register at regular intervals.

When the interrupt enable bit of any port 3 pin is "1", rising or falling signal edge at that pin generates an interrupt request. The corresponding PINTPND 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 should clear the pending condition by writing a "0" to the corresponding PINTPND bit.

9.6.3 Port 3 Pull-Up Resistor Enable Register

Using the port 3 pull-up resistor enable register, P3PUR (EBH, set1, bank1), you can configure pull-up resistors to individual port 3 pins.

9.6.4 Port 3 N-Channel Open-drain Mode Register

Using the port 3 n-channel open-drain mode register, PNE3 (ECH, set1, bank1), you can configure push-pull or open-drain output mode to individual port 3 pins.

Figure 9-9 illustrates the port 3 high-byte control register.

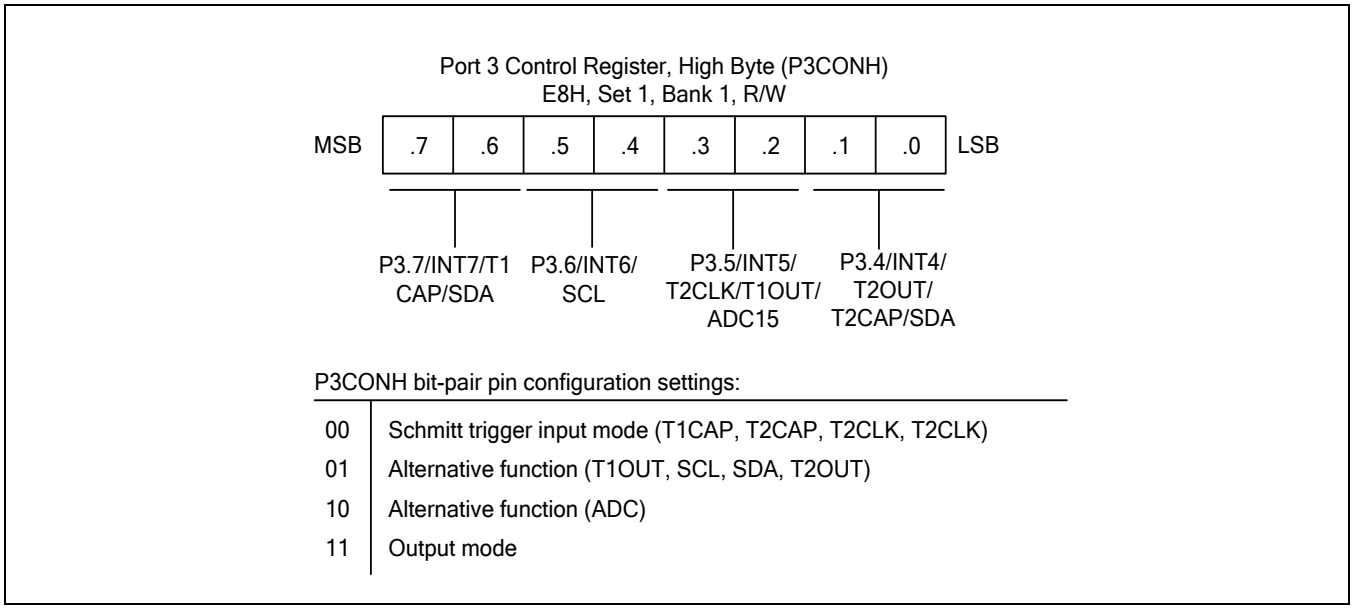


Figure 9-9 Port 3 High-Byte Control Register (P3CONH)

Figure 9-10 illustrates the port 3 low-byte control register.

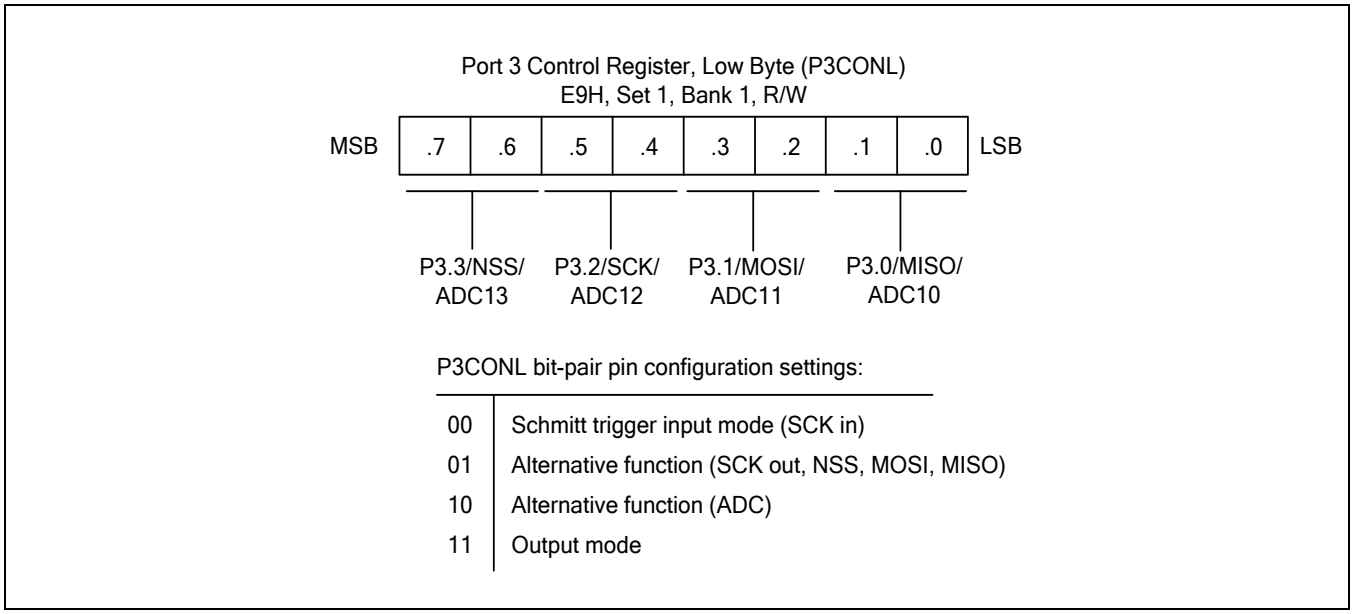


Figure 9-10 Port 3 Low-Byte Control Register (P3CONL)

Figure 9-11 illustrates the Port 3 interrupt control register.

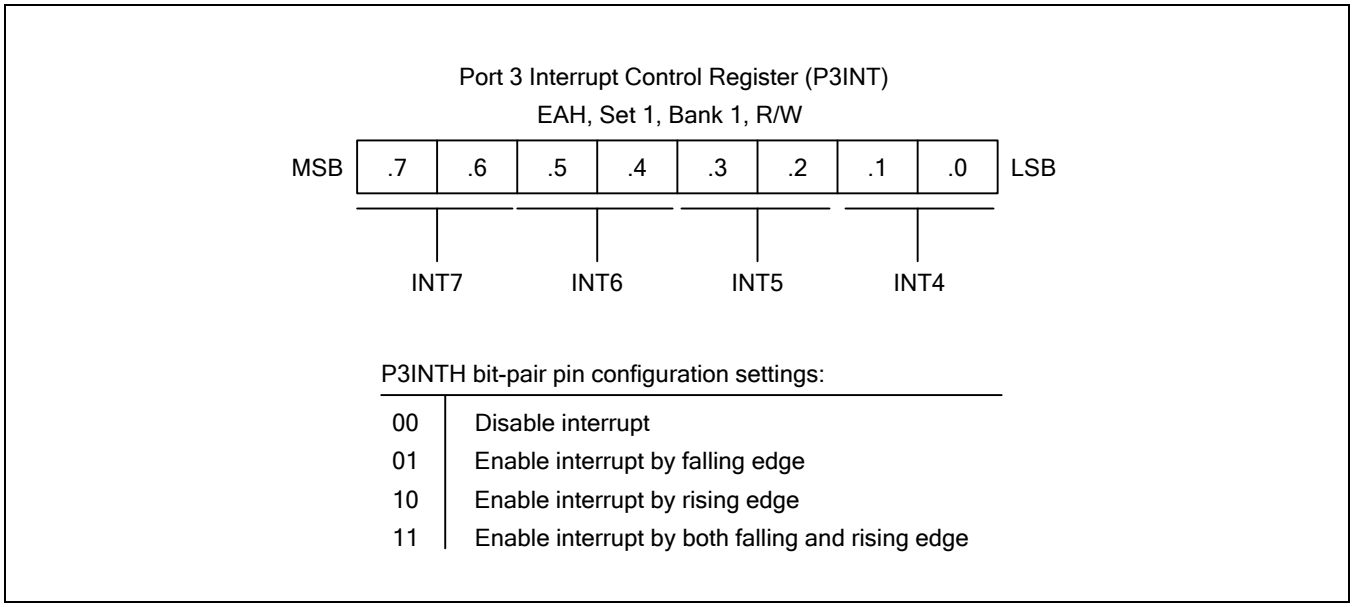


Figure 9-11 Port 3 Interrupt Control Register (P3INT)

Figure 9-12 illustrates the port 3 & port 0 interrupt pending register.

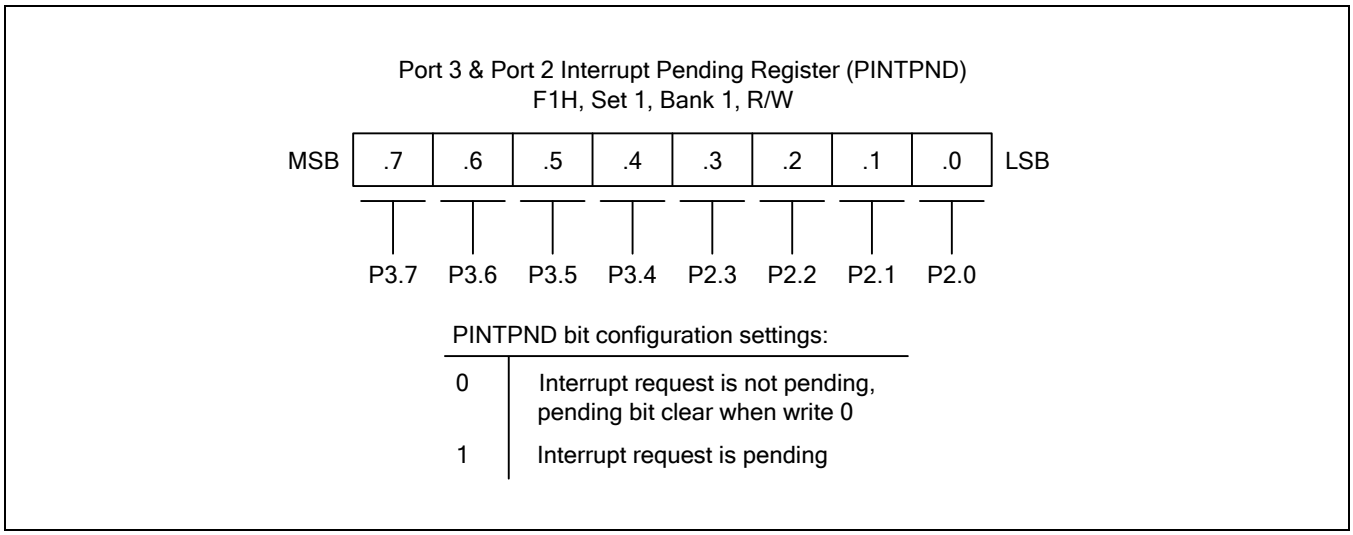


Figure 9-12 Port 3 & Port 0 Interrupt Pending Register (PINTPND)

Figure 9-13 illustrates the port 3 pull-up resistor enable register.

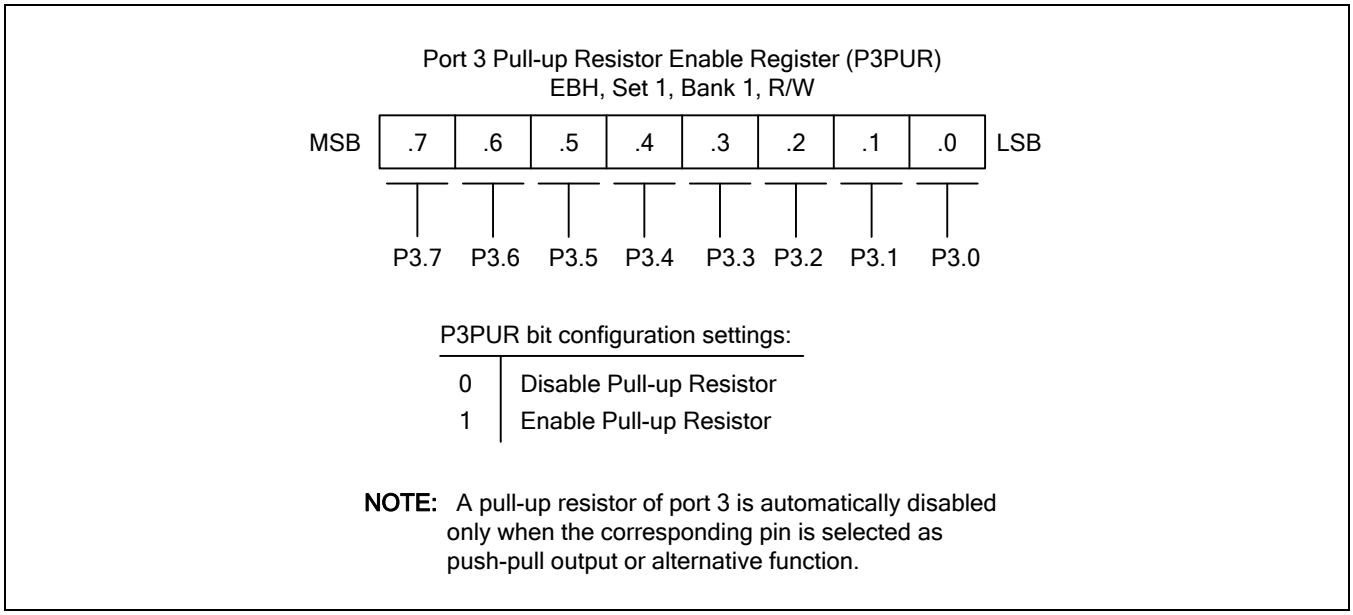


Figure 9-13 Port 3 Pull-Up Resistor Enable Register (P3PUR)

Figure 9-14 illustrates the port 3 n-channel open-drain mode register.

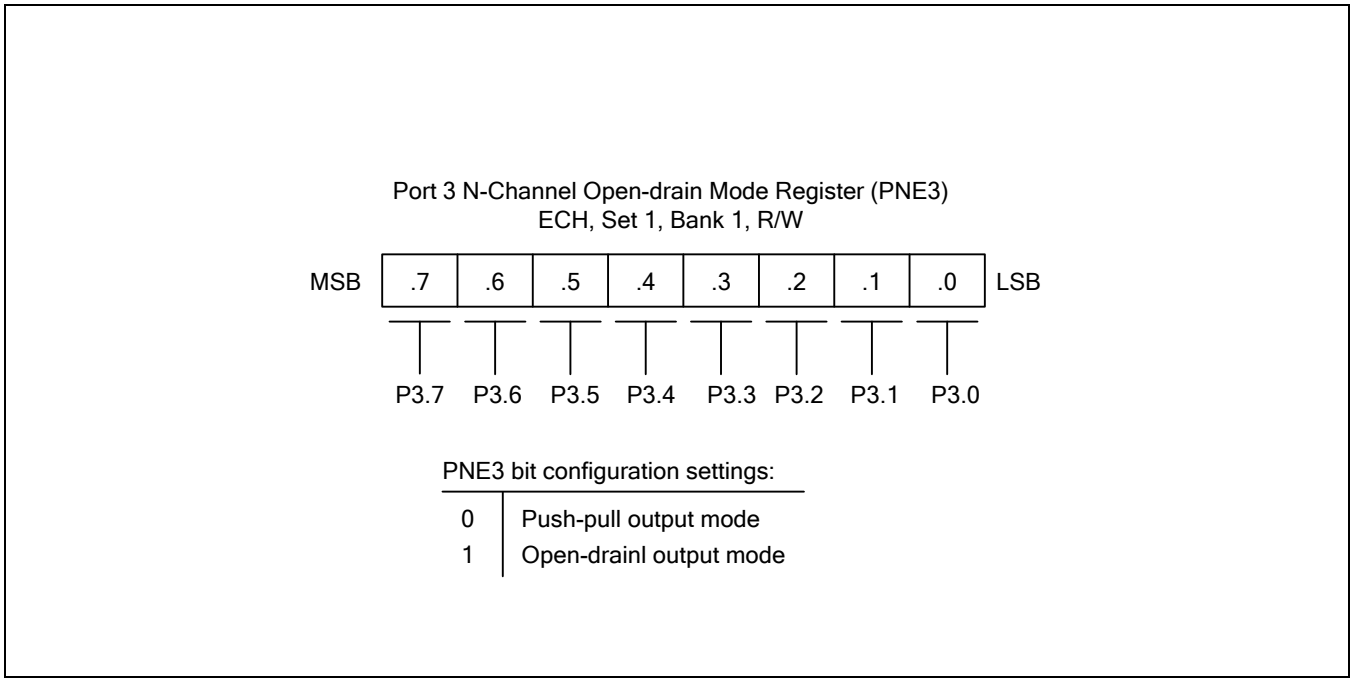


Figure 9-14 Port 3 N-Channel Open-Drain Mode Register (PNE3)

10 Basic Timer

10.1 Overview

S3F8S39/S3F8S35 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) 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. It also enables or disables 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 should write the signature code "1010B" to the basic timer register control bits BTCON.7–BTCON.4.

You can clear the 8-bit basic timer counter, BTCNT (set 1, bank 0, FDH), 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 illustrates the basic timer control register.

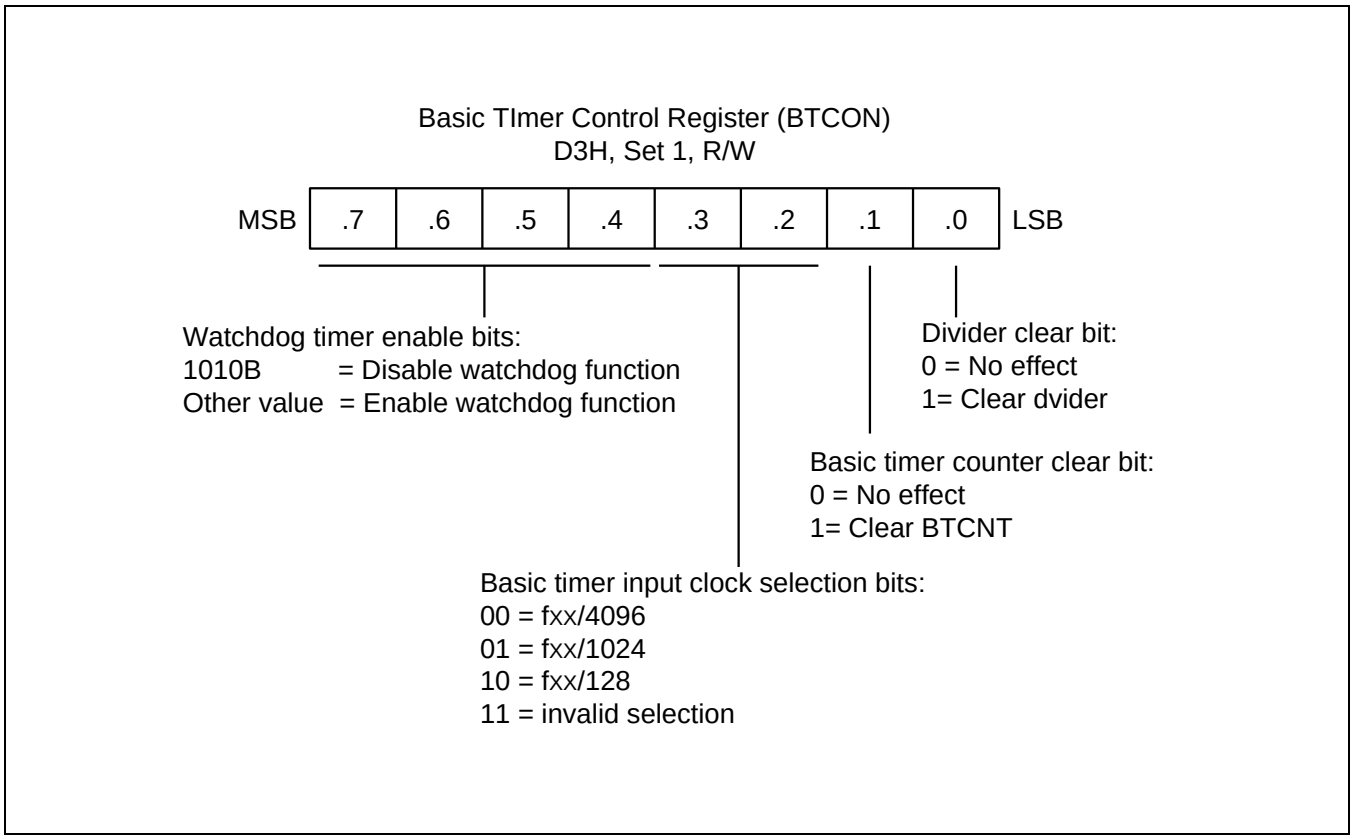


Figure 10-1 Basic Timer Control Register (BTCON)

10.3 Basic Timer Function Description

This section includes:

- Watchdog timer function
- Oscillation stabilization interval timer function

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 enables 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 should prevent the overflow, and the accompanying reset operation, from occurring, To do this you should clear BTCNT value (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 occurs, initiating a reset. In other words, during the normal operation, a BTCNT clear instruction always beak the basic timer overflow loop (a bit 7 overflow of the 8-bit basic timer counter, BTCNT). If a malfunction occurs, a reset is triggered automatically.

10.3.2 Oscillation Stabilization Interval Timer Function

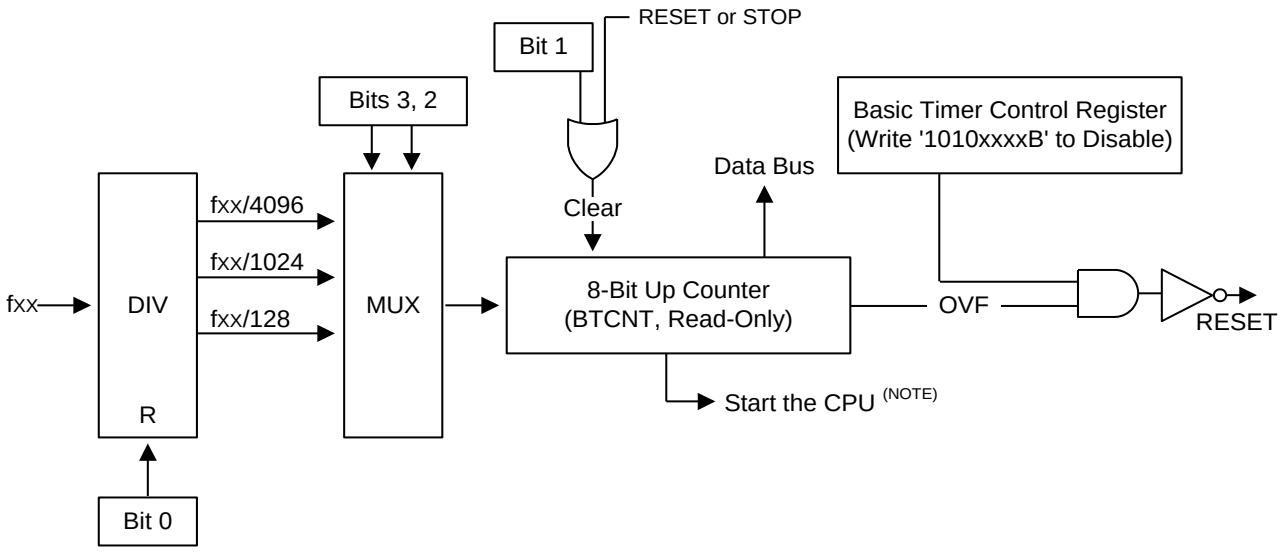
You can also use the basic timer to program a specific oscillation stabilization interval after a reset or when an external interrupt releases Stop Mode.

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 it elapses the stabilization interval and to gate the clock signal off to the CPU so that it can resume the normal operation.

In summary, the events that 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 occurs, the basic timer counter increases 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. When a BTCNT.4 overflow occurs, the normal CPU operation resumes.

Figure 10-2 illustrates the basic timer block diagram.



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

11.1 Overview

The 8-bit Timer A is an 8-bit general-purpose timer/counter. Timer A has three operating modes, you can select one of the modes using appropriate TACON setting. The settings are:

- Interval timer mode (Toggle output at TAOUT pin)
- Capture input mode with a rising or falling edge trigger at the TACAP pin
- PWM Mode (TAOUT)

The functional components of Timer A are:

- Clock frequency divider with prescaler (TAPS)
- 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 (TAOUT)
- Timer A overflow interrupt (IRQ0 vector D0H) and matches/captures interrupt (IRQ0 vector CEH) generation
- Timer A control register, TACON (set 1, Bank 0, E2H, and Read/Write)

11.2 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 and 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, 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.5.

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 should write TACON.3 to "1". To detect a match/capture interrupt pending condition, the application program polls TACON.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, TACON.1.

*Figure 11-1*illustrates the Timer A control register.

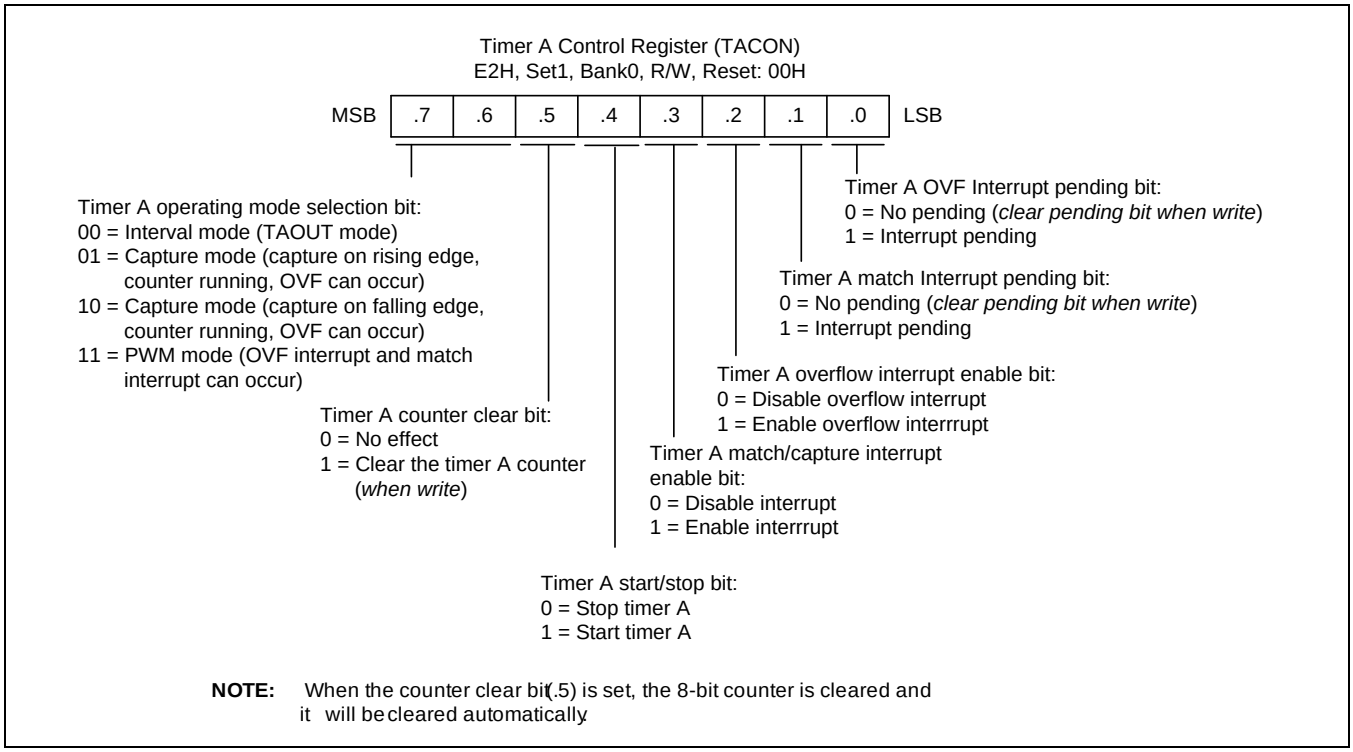


Figure 11-1 Timer A Control Register (TACON)

Figure 11-2 illustrates the Timer A prescaler register.

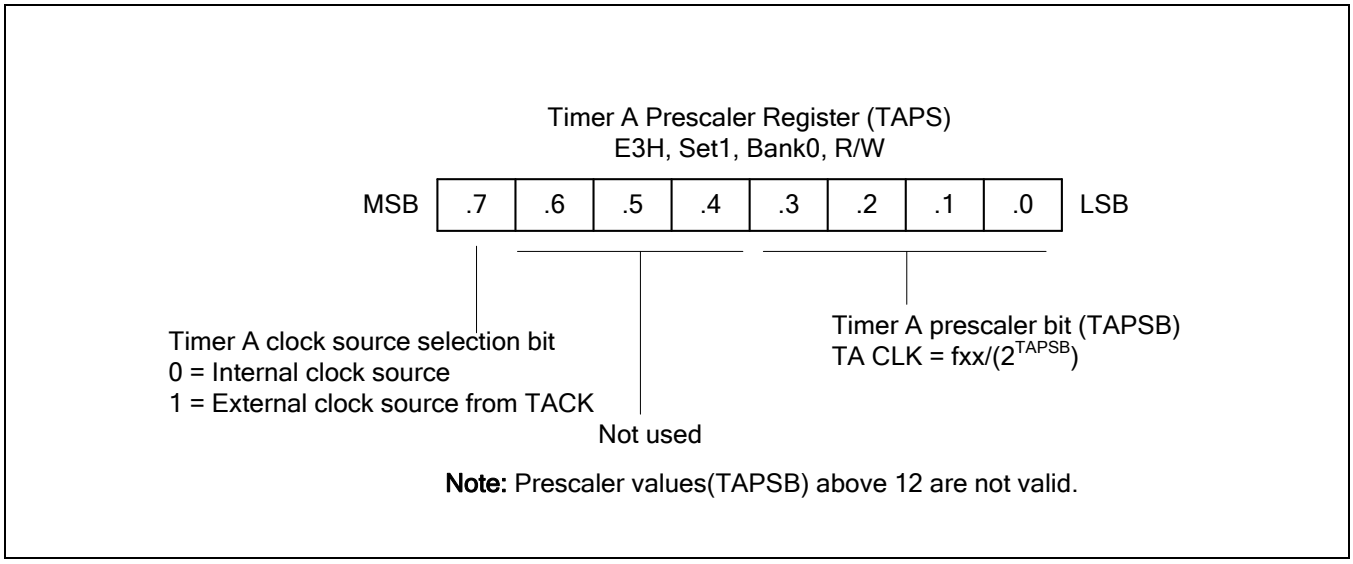


Figure 11-2 Timer A Prescaler Register (TAPS)

11.3 Function Description

This section includes:

- Timer A interrupts
- Interval timer function
- Pulse width modulation mode
- Capture Mode

11.3.1 Timer A Interrupts

The Timer A module can generate two interrupts:

- Timer A overflow interrupt (TAOVF)
- Timer A match/ capture interrupt (TAINT)

Timer A overflow interrupt can be cleared by both software and hardware, and match/capture interrupt pending conditions are cleared by software when it has been serviced.

11.3.2 Interval Timer Function

The Timer A module can generate an interrupt: the Timer A match interrupt (TAINT).

When Timer A interrupt occurs and is serviced by the CPU, the pending condition should be cleared by software.

In interval timer mode, a match signal is generated and TAOUT is toggled 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 and clears the counter.

If, for example, you write the value 10H to TADATA and 0BH to TACON, the counter increments until it reaches 10H. At this point the TA interrupt request is generated; resets the counter value, and resumes counting.

11.3.3 Pulse Width Modulation Mode

Pulse width modulation (PWM) mode lets you program the width (duration) of the pulse that is output at the TAOUT 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 0xFFH, 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 TAOUT pin is held to Low level as long as the reference data value is less than or equal to (≤) the counter value the pulse is held to High level for as long as the data value is greater than (>) the counter value. One pulse width is equal to $t_{CLK} \times 256$.

11.3.4 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 value of the Timer A capture input selection bit in P2CONL, (E5H, Set1 Bank1).
When P2CONL.7–6 is 00, the TACAP input or normal input is selected.

you can use both types of Timer A interrupts in capture mode: the Timer A overflow interrupt is generated whenever a counter overflow occurs; the Timer A match/capture interrupt is generated when 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.

11.4 Block Diagram

Figure 11-3 illustrates the Timer A functional block diagram.

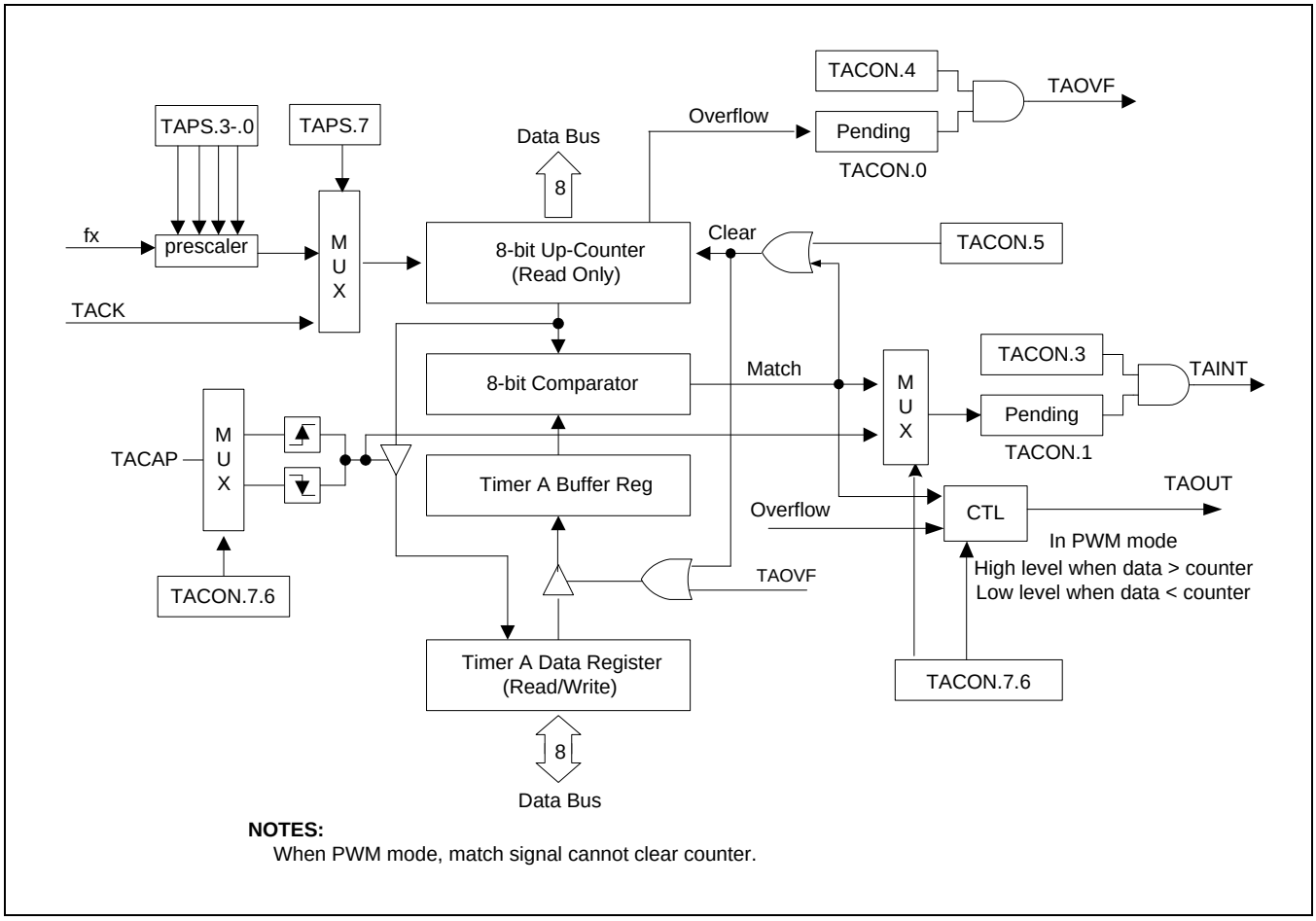


Figure 11-3 Timer A Functional Block Diagram

12 16-Bit Timer 0/1/2

12.1 Overview

The 16-bit timer 0 is a 16-bit general-purpose timer/counter.

Timer 0 has two operating modes:

- Interval timer mode (Toggle output at T0OUT pin); Only match interrupt occurs
- Pulse Width modulation (PWM) mode (T0OUT/T0PWM pin); Match and overflow interrupt can occur

You can select one of these modes by using appropriate T0CON setting.

The functional components Timer 0 are:

- Clock frequency divider with prescaler (T0PS)
- 16-bit counter, 16-bit comparator, and 16-bit reference data register (T0DATAH/L)
- PWM or match output (T0OUT)
- Timer 0 overflow interrupt (IRQ2, vector D6H) and match interrupt (IRQ2, vector D4H) generation
- Timer 0 control register, T0CON (set 1, bank0, ECH, read/write)

12.2 Timer 0 Control Register

You can use the timer 0 control register, T0CON, to:

- Select the timer 0 operating mode (PWM Mode or interval mode)
- Select the timer 0 4 bits prescaler
- Clear the timer 0 counter, T0CNT
- Enable the timer 0 match/overflow interrupt
- Start the timer 0

T0CON is located in set 1, Bank 0 at address ECH, and is read/write addressable that uses the Register addressing mode.

A reset clears T0CON to "00H". This sets timer 0 to interval mode, selects fxx, stops timer 0, and disables all timer 0 interrupts. You can clear the timer 0 counter at any time during normal operation by writing a "1" to T0CON.3.

To enable the timer 0 match/overflow interrupt (IRQ2, vector D4H), you should write T0CON.7 and T0CON.5 to "1". To generate the exact time interval, you should write T0CON.5 and 1, which clears counter and interrupt pending bit. To detect an interrupt pending condition when T0INT is disabled, the application program should poll the pending bit, T0CON.2.When a "1" is detected, a timer 0 match/overflow interrupt is pending. When the T0INT sub-routine has been serviced, the pending condition must be cleared by software by writing a "0" to the timer 0 interrupt pending bit, T0CON.1.

Figure 12-1 illustrates the Timer 0 control register.

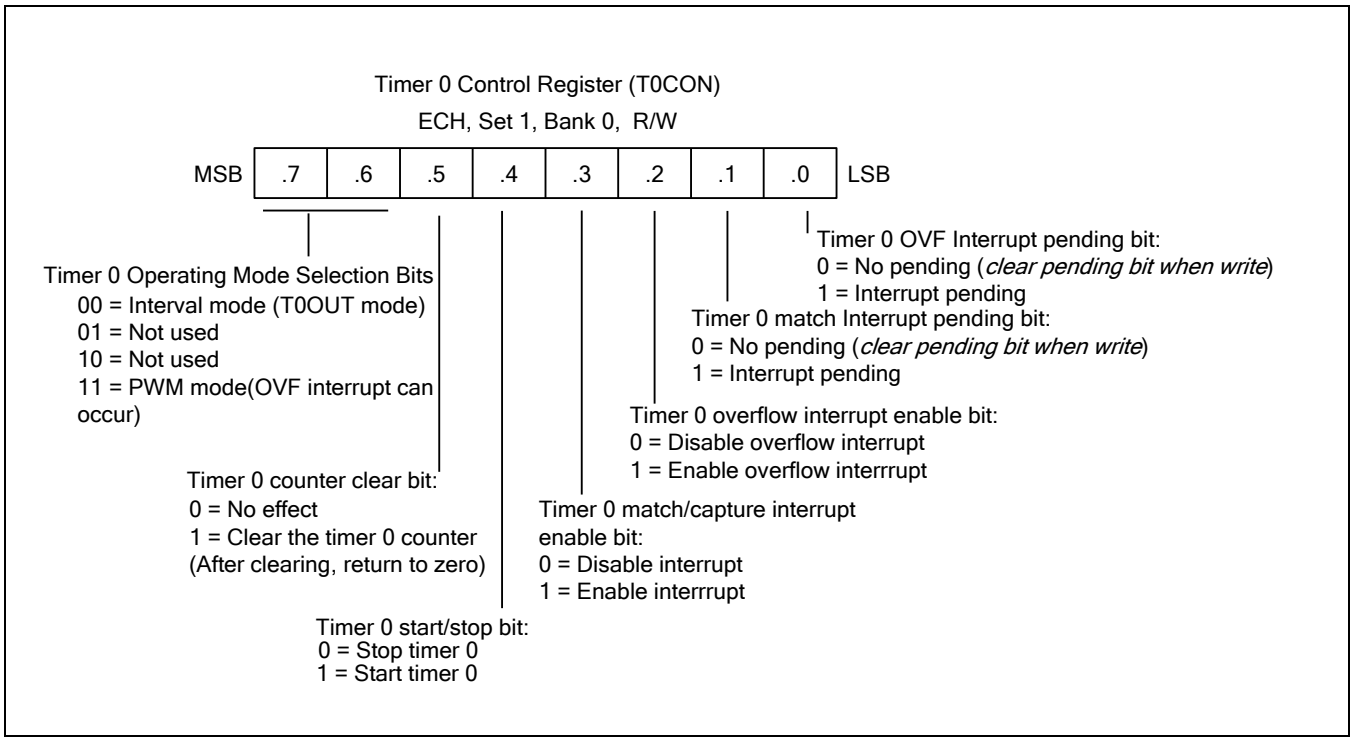


Figure 12-1 Timer 0 Control Register (T0CON)

Figure 12-2 illustrates the Timer 0 prescaler register.

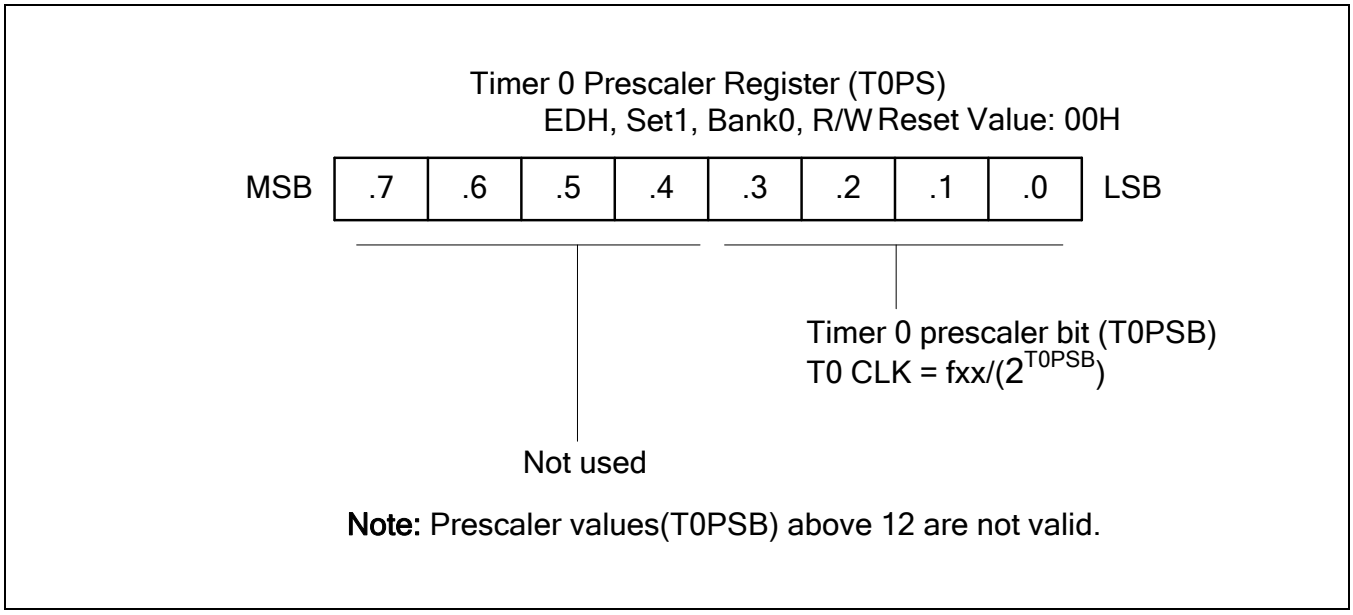


Figure 12-2 Timer 0 Prescaler Register (T0PS)

12.3 Block Diagram

Figure 12-3 illustrates the functional block diagram of Timer 0.

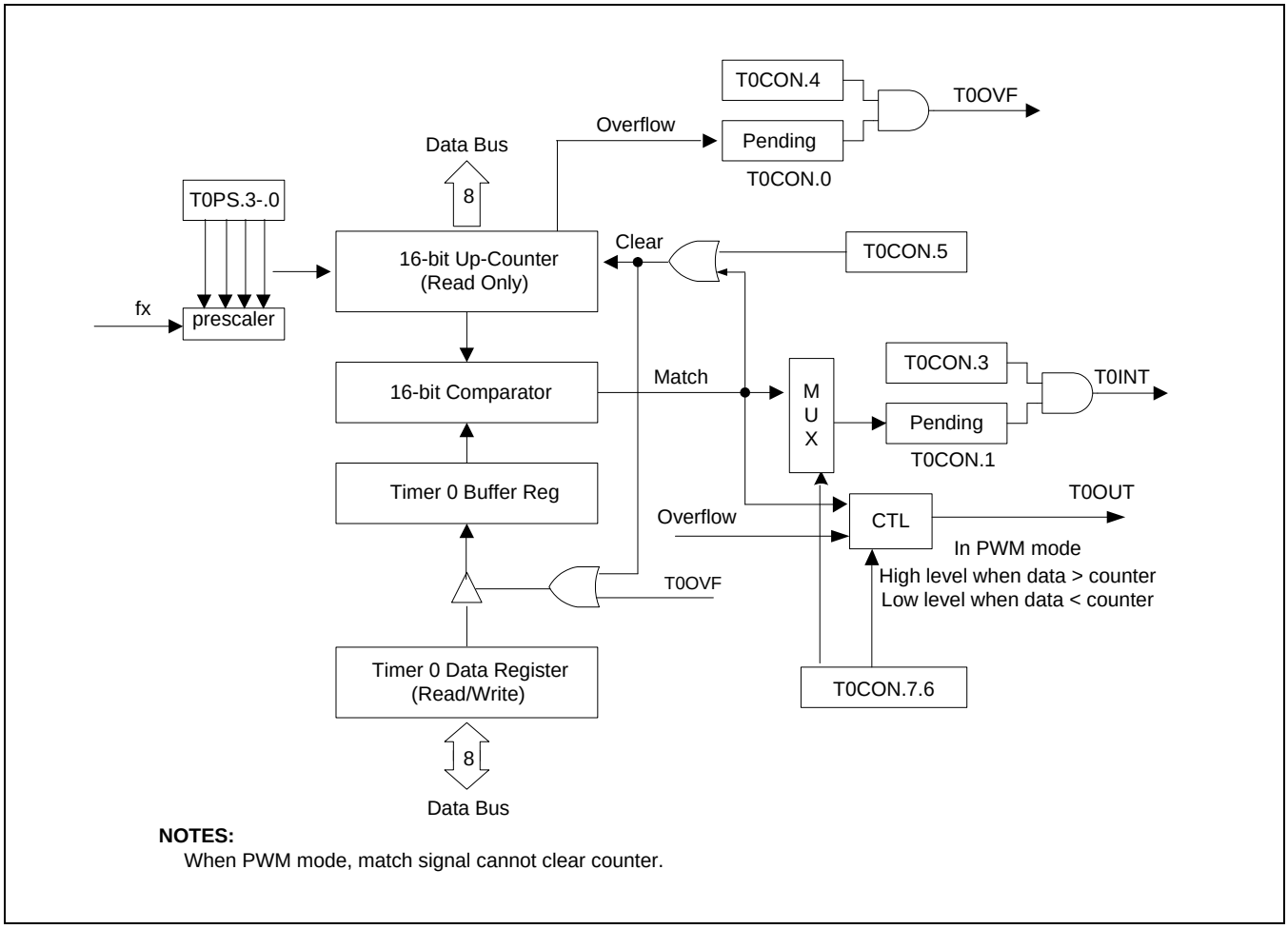


Figure 12-3 Timer 0 Functional Block Diagram

12.4 Overview 16-Bit Timer 1

The 16-bit timer 1 is a 16-bit general-purpose timer.

Timer 1 has three operating modes:

- Interval timer mode (Toggle output at T1OUT pin)
- Capture input mode with a rising or falling edge trigger at the T1CAP pin
- PWM Mode: PWM output shares their output port with T1OUT pin

You can select one of these modes can by using the appropriate T1CON setting.

The functional components Timer 1 are:

- External clock input pin (T1CLK)
- Clock frequency divider with prescaler (T1PS)
- A 16-bit counter (T1CNTH/L), a 16-bit comparator, and two 16-bit reference data register (T1DATAH/L)
- I/O pins for capture input (T1CAP), or match output (T1OUT)
- Timer 1 overflow interrupt (IRQ3, vector DAH) and match/capture interrupt (IRQ3, vector D8H) generation
- Timer 1 control register, T1CON (set 1, Bank 0, F2H, read/write)

12.5 Timer 1 Control Register

You use the timer 1 control register, T1CON, to:

- Select the timer 1 operating mode (interval timer, capture mode, or PWM Mode)
- Select the timer 1 input clock frequency (T1PS)
- Clear the timer 1 counter, T1CNTH/T 1CNTL
- Enable the timer 1 overflow interrupt or timer 1 match/capture interrupt

T1CON is located in set 1 and bank 0 at address F2H, and is read/write addressable that uses the Register addressing mode.

A reset clears T1CON to "00H". This sets timer 1 to normal interval timer mode. It selects an input clock frequency of fxx, and disables all timer 1 interrupts. To disable the counter operation, set T1CON.4 to "0". You can clear the timer 1 counter at any time during normal operation by writing a "1" to T1CON.5.

The timer 1 overflow interrupt (T1OVF) is interrupt level IRQ3 and has the vector address 0xDAH. When a timer 1 match/capture interrupt occurs and is serviced interrupt (IRQ3, vector DAH), you should write T1CON.1 to "0". When a timer 1 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 1 match/capture interrupt (IRQ3, vector D8H), you should write T1CON.3 to "1". To detect a match/capture interrupt pending condition, the application program polls T1CON.1. When a "1" is detected, a timer 1 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 1 match/capture interrupt pending bit, T1CON.1.

Figure 12-4 illustrates the Timer 1 control register.

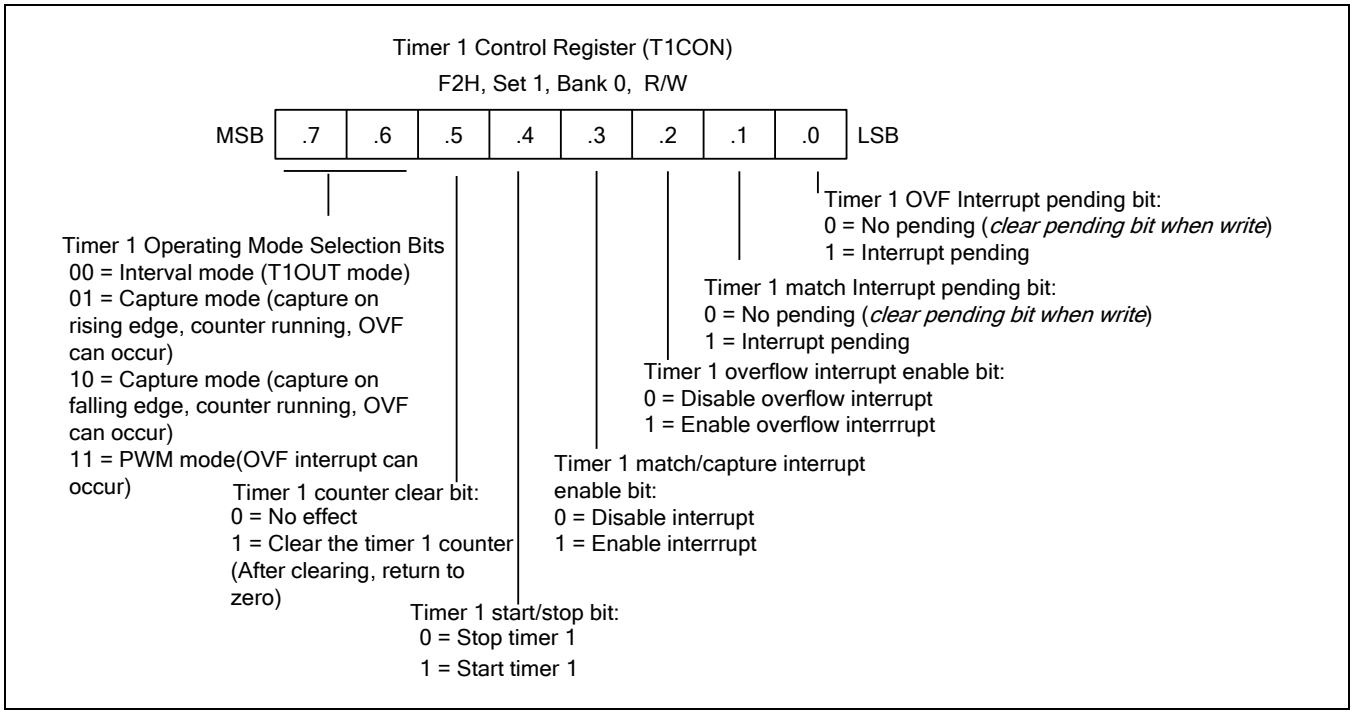


Figure 12-4 Timer 1 Control Register (T1CON)

Figure 12-5 illustrates the Timer 1 prescaler register.

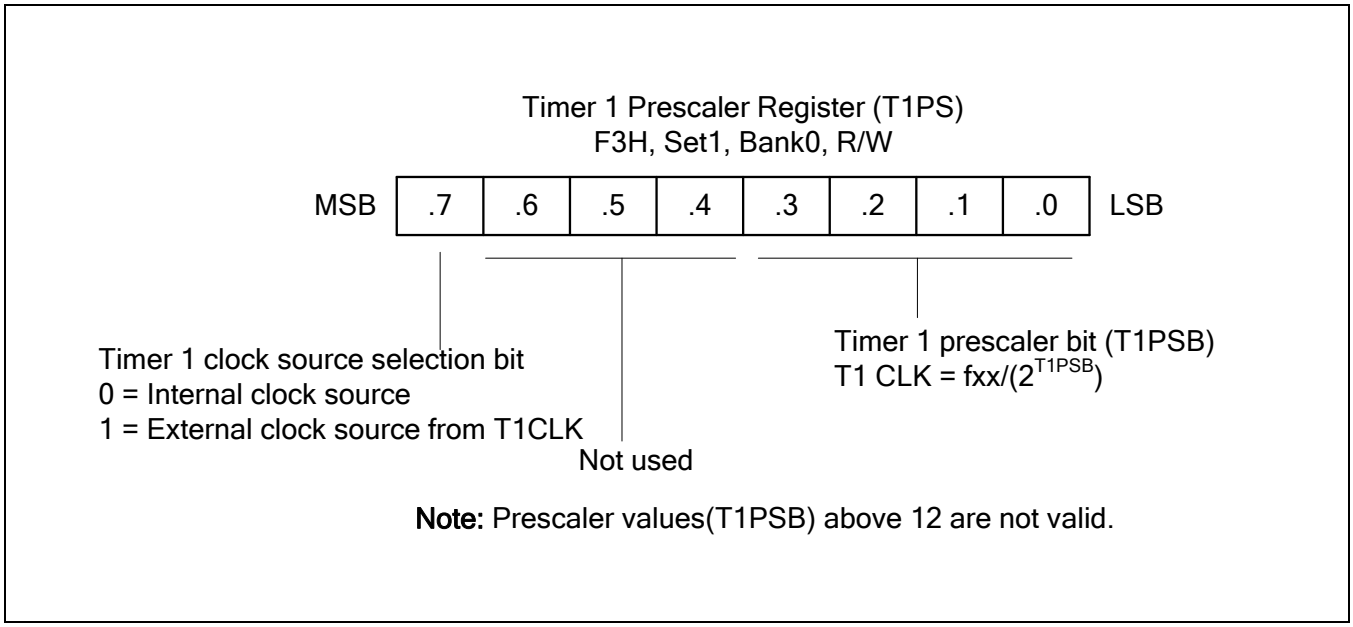


Figure 12-5 Timer 1 Prescaler Register (T1PS)

12.6 Timer 1 Function Description

This section includes:

- Timer 1 interrupts (IRQ3, Vectors D8H and DAH)
- Interval timer mode
- Pulse width modulation mode
- Capture Mode

12.6.1 Timer 1 Interrupts (IRQ3, Vectors D8H and DAH)

The timer 1 generates two interrupts:

- Timer 1 overflow interrupt (T1OVF)
- Timer 1 match/capture interrupt (T1INT)

T1OVF belongs to interrupt level IRQ3, vector DAH. T1INT belongs to interrupt level IRQ3, but is assigned the separate vector address, D8H.

A timer 1 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 T1CON.0. However, the timer 1 match/capture interrupt pending condition must be cleared by the application's interrupt service routine by writing a "0" to the T1CON.1.

12.6.2 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 1 reference data register, T1DATAH/T1DATAL. The match signal generates a timer 1 match interrupt (T1INT, vector D8H) and clears the counter.

If, for example, you write the value "1087H" to T1DATAH/T1DATAL, the counter will increment until it reaches "1087H". At this point, the timer 1 interrupt request is generated, resets the counter value, and resumes counting. With each match, the level of the signal at the timer 1 output pin is inverted.

12.6.3 Pulse Width Modulation Mode

PWM Mode enables you to program the width (duration) of the pulse that is output at the T1OUT pin. As in interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer 1 data register. However in PWM Mode, the match signal does not clear the counter. Instead, it runs continuously, overflowing at "FFFFH", and then continues to increment from "0000H".

Although you can use the match signal to generate a timer 1 overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the T1PWM pin is held to Low level as long as the reference data value is less than or equal to (≤) the counter value the pulse is held to High level for as long as the data value is greater than (>) the counter value. One pulse width is equal to $t_{CLK} \times 65536$.

12.6.4 Capture Mode

In capture mode, a signal edge that is detected at the T1CAP pin opens a gate and loads the current counter value into the timer 1 data register. You can select rising or falling edges to trigger this operation.

Timer 1 also provides you with capture input source: the signal edge at the T1CAP pin. You can select the capture input by setting the values of the timer 1 capture input selection bits in the port 3 control register, P3CONH.7–6 (set 1, bank 1, E8H). When P3CONH.7–6 is "00", the T1CAP input is selected.

Both kinds of timer 1 interrupts can be used in capture mode: the timer 1 overflow interrupt is generated whenever a counter overflow occurs. The timer 1 match/capture interrupt is generated whenever the counter value is loaded into the timer 1 data register.

By reading the captured data value in T1DATAH/T1DATAL, and assuming a specific value for the timer 1 clock frequency, you can calculate the pulse width (duration) of the signal that is being input at the T1CAP pin.

12.7 Block Diagram

Figure 12-6 illustrates the functional block diagram of Timer 1.

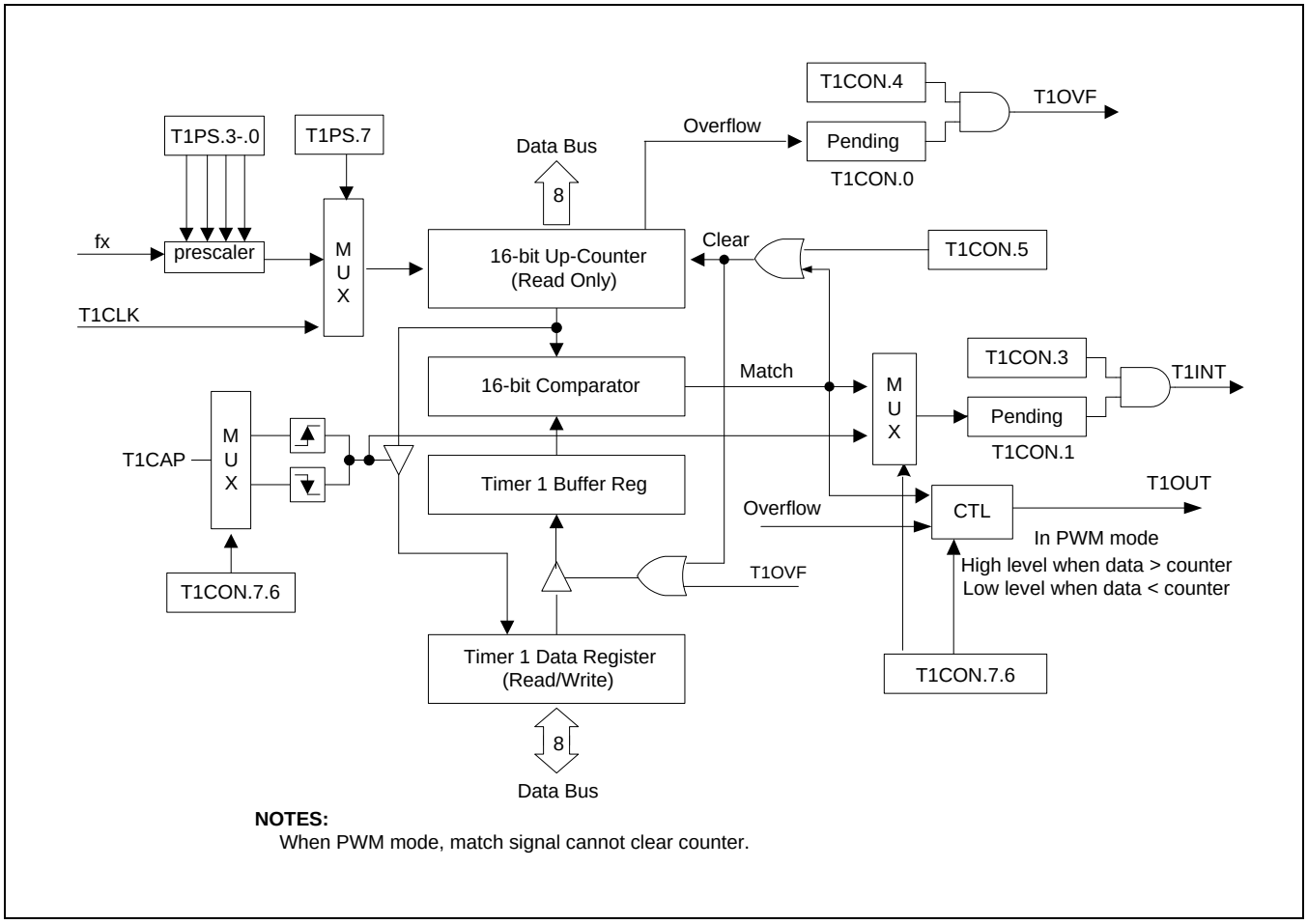


Figure 12-6 Timer 1 Functional Block Diagram

12.8 Overview 16-Bit Timer 2

The 16-bit timer 2 is a 16-bit general-purpose timer.

Timer 2 has three operating modes:

- Interval timer mode (Toggle output at T2OUT pin)
- Capture input mode with a rising or falling edge trigger at the T2CAP pin
- PWM Mode (T2PWM): PWM output shares their output port with T2OUT pin

You can select of these modes by using the appropriate T2CON setting.

The functional components of Timer 2 are:

- Clock frequency divider with prescaler(T2PS)
- External clock input pin (T2CLK)
- A 16-bit counter (T2CNTH/L), a 16-bit comparator, and two 16-bit reference data register (T2DATAH/L)
- I/O pins for capture input (T2CAP), or match output (T2OUT)
- Timer 2 overflow interrupt (IRQ3, vector DEH) and match/capture interrupt (IRQ3, vector DCH) generation
- Timer 2 control register, T2CON (set 1, Bank 1, F8H, read/write)

12.9 Timer 2 Control Register

You use the timer 2 control register, T2CON, to:

- Select the timer 2 operating mode (interval timer, capture mode, or PWM Mode)
- Select the timer 2 input clock frequency(T2PS)
- Clear the timer 2 counter, T2CNTH/T 2CNTL
- Enable the timer 2 overflow interrupt or timer 2 match/capture interrupt

T2CON is located in set 1 and bank 1 at address F8H, and is read/write addressable that uses Register addressing mode.

A reset clears T2CON to "00H". This sets timer 1 to normal interval timer mode, selects an input clock frequency of fxx, and disables all timer 2 interrupts. To disable the counter operation, set T2CON.4 to "0". You can clear the timer 2 counter at any time during normal operation by writing a "1" to T2CON.5.

The timer 2 overflow interrupt (T2OVF) is interrupt level IRQ3 and has the vector address 0xDEH. When a timer 2 match/capture interrupt occurs and is serviced interrupt (IRQ3, vector DEH), you should write T2CON.1 to "0". When a timer 2 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 2 match/capture interrupt (IRQ3, vector DCH), you should write T2CON.3 to "1". To detect a match/capture interrupt pending condition, the application program polls T2CON.1. When a "1" is detected, a timer 2 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 2 match/capture interrupt pending bit, T2CON.1.

Figure 12-7 illustrates the Timer 2 control register.

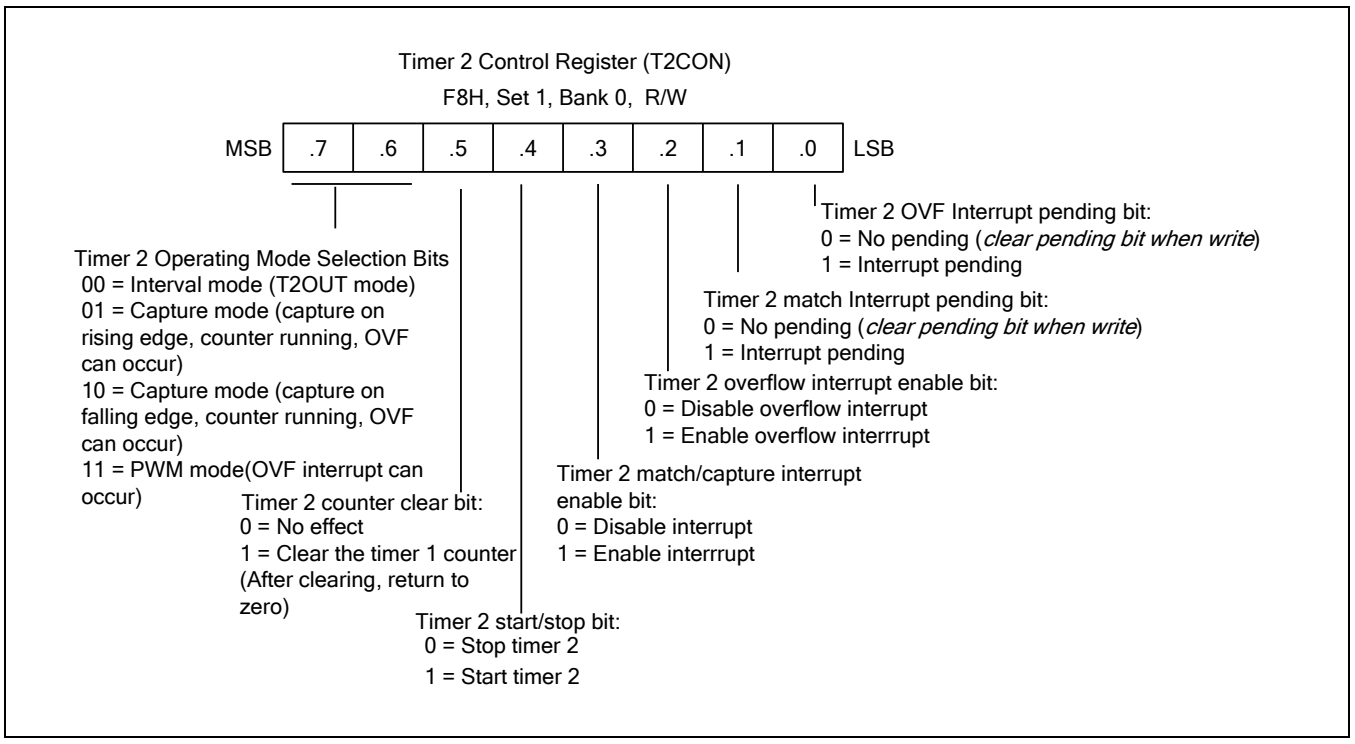


Figure 12-7 Timer 2 Control Register (T2CON)

Figure 12-8 illustrates the Timer 2 prescaler register.

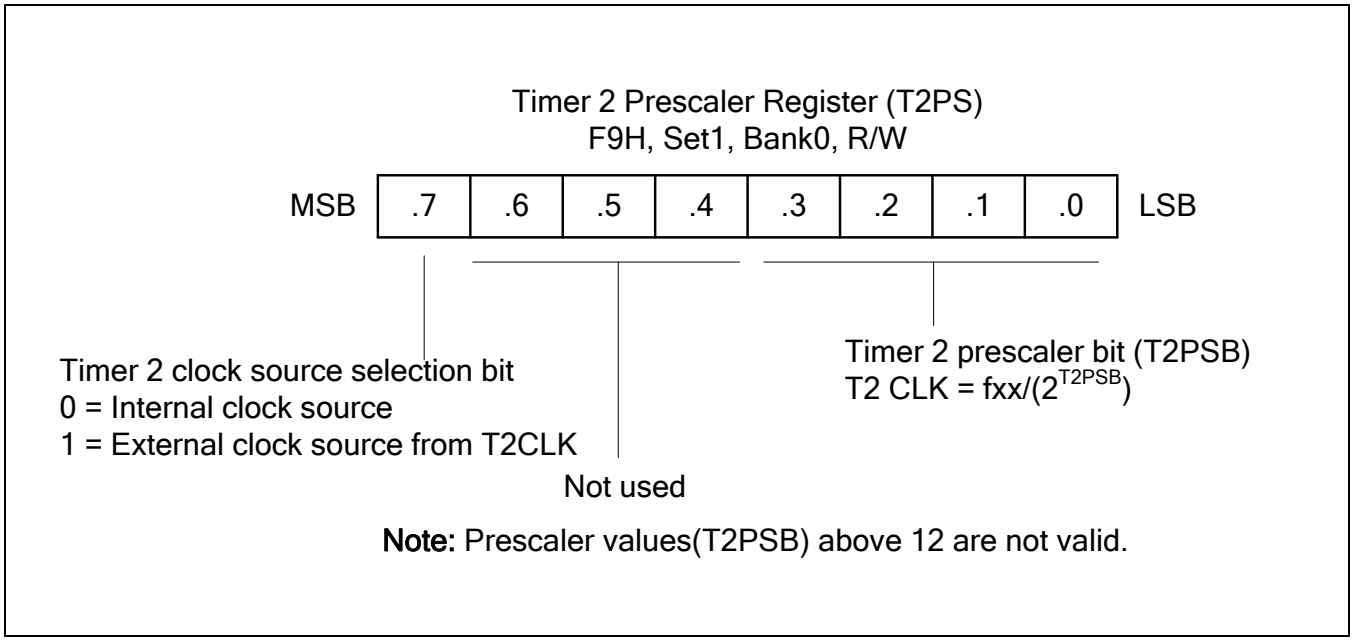


Figure 12-8 Timer 2 Prescaler Register (T2PS)

12.10 Timer 2 Function Description

This section includes:

- Timer 2 interrupts (IRQ3, Vectors DCH and DEH)
- Interval timer mode
- Pulse width modulation mode
- Capture Mode

12.10.1 Timer 2 Interrupts (IRQ3, Vectors DCH and DEH)

The timer 2 can generate two interrupts:

- Timer 2 overflow interrupt (T2OVF)
- Timer 2 match/capture interrupt (T2INT)

T2OVF belongs to interrupt level IRQ3, vector DEH. T2INT belongs to interrupt level IRQ3, but is assigned the separate vector address, DCH.

A timer 2 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 T2CON.0. However, the timer 2 match/capture interrupt pending condition must be cleared by the application's interrupt service routine by writing a "0" to the T2CON.1.

12.10.2 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 2 reference data register, T2DATAH/T2DATAL. The match signal generates a timer 2 match interrupt (T2INT, vector DCH) and clears the counter.

If, for example, you write the value "1087H" to T2DATAH/T2DATAL, the counter increments until it reach "1087H". At this point, the timer 2 interrupt request is generated, resets the counter value, and resumes counting. With each match, the level of the signal at the timer 2 output pin is inverted.

12.10.3 Pulse Width Modulation Mode

PWM Mode enables you to program the width (duration) of the pulse that is output at the T2OUT pin. As in interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer 2 data register. However, in PWM Mode, the match signal does not clear the counter. Instead, it runs continuously, overflowing at "FFFFH", and then continues to increment from "0000H".

Although you can use the match signal to generate a timer 2 overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the T2OUT pin is held to Low level as long as the reference data value is less than or equal to ($\leq$) the counter value. 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 $t_{CLK} \times 65536$.

12.10.4 Capture Mode

In capture mode, a signal edge that is detected at the T2CAP pin opens a gate and loads the current counter value into the timer 2 data register. You can select rising or falling edges to trigger this operation.

Timer 2 also provides you with capture input source: the signal edge at the T2CAP pin. You can select the capture input by setting the values of the timer 2 capture input selection bits in the port 3 control register, P3CONH.3–2 (set 1, bank 1, E4H). When P3CONH.3–2 is "00", the T2CAP input is selected.

Both kinds of timer 2 interrupts can be used in capture mode: the timer 2 overflow interrupt is generated whenever a counter overflow occurs. The timer 2 match/capture interrupt is generated whenever the counter value is loaded into the timer 2 data register.

By reading the captured data value in T2DATAH/T2DATAL and assuming a specific value for the timer 2 clock frequency, you can calculate the pulse width (duration) of the signal that is being input at the T2CAP pin.

12.11 Block Diagram

Figure 12-9 illustrates the functional block diagram of Timer 2.

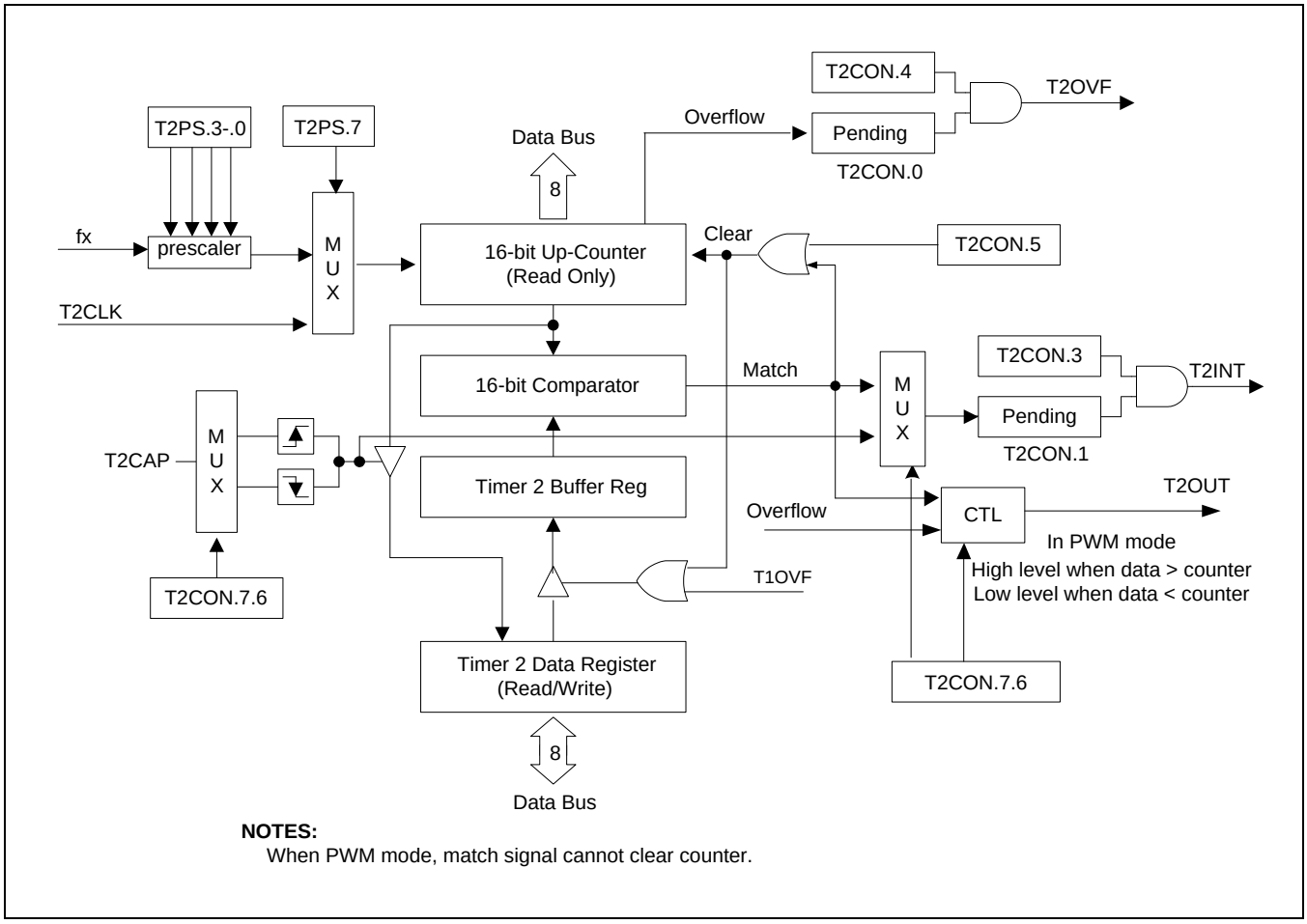


Figure 12-9 Timer 2 Functional Block Diagram

13 16-Bit Timer B

13.1 Overview

The S3F8S39/S3F8S35 micro-controller has a 16-bit counter called Timer B. You can use this to generate the carrier frequency of a remote controller signal.

Timer B has four functions:

- As a normal interval timer, generating a Timer B interrupt at programmed time intervals.
- To generate a programmable carrier pulse for a remote control signal at TBOUT.
- To generate one shot pulse triggered by external input signal or software.
- Emergency detection by external input trigger and stop Pulse width modulation (PWM) output automatically.

Figure 13-1 illustrates the Timer B control register.

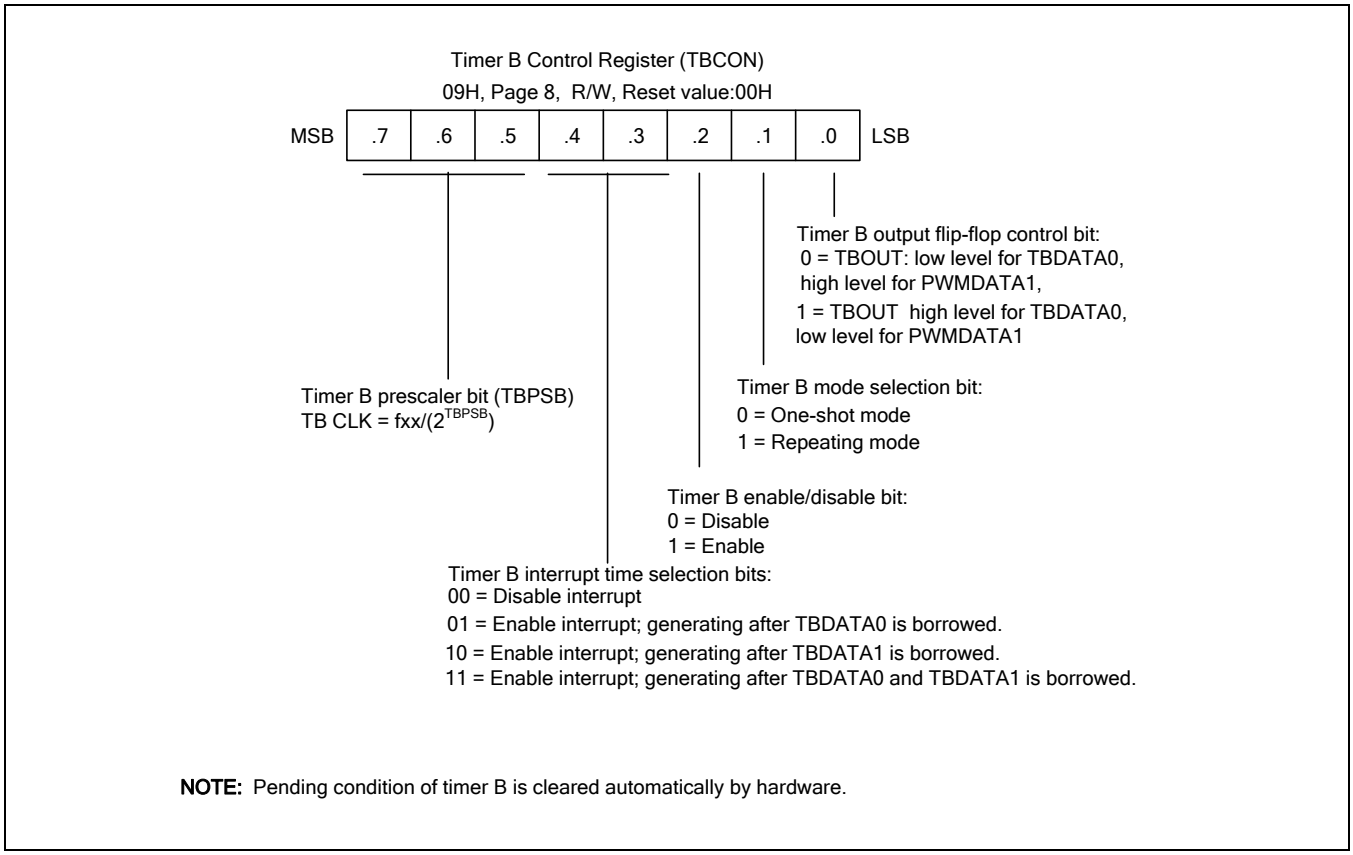


Figure 13-1 Timer B Control Register

Figure 13-2 illustrates the Timer B trigger register (TBTRG).

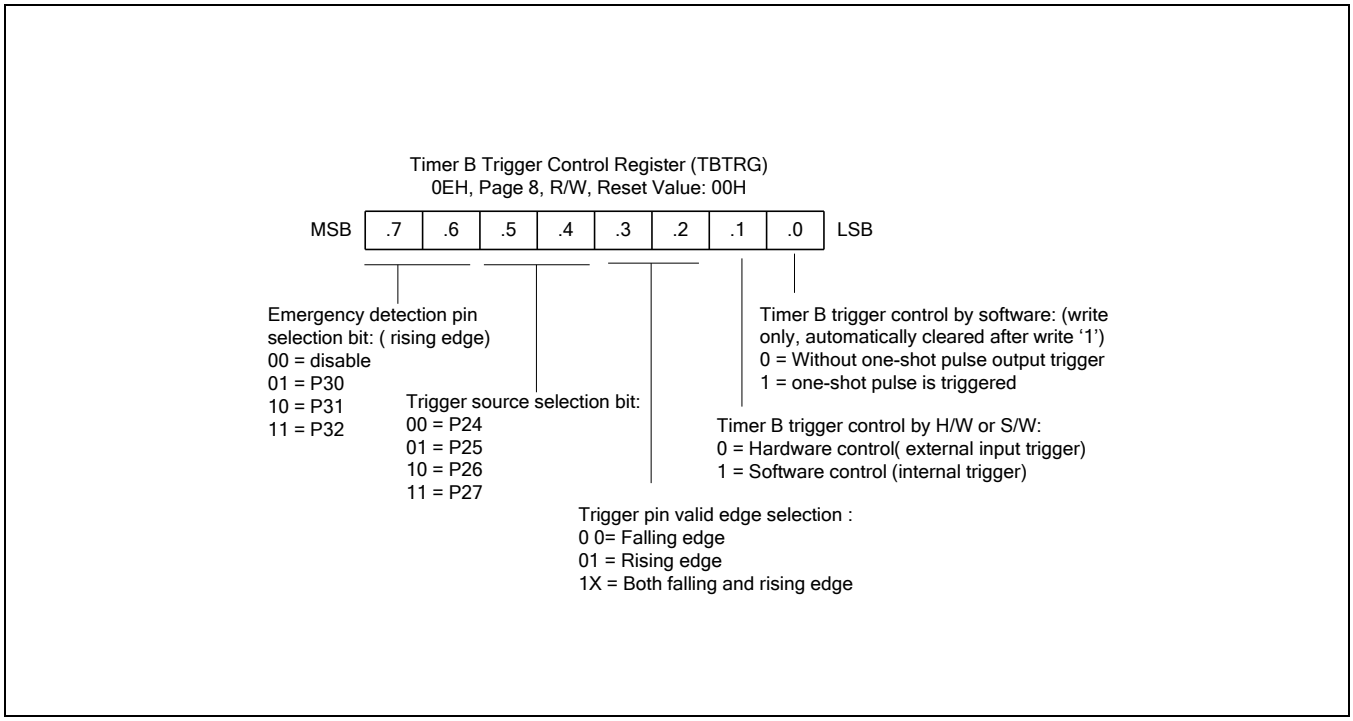


Figure 13-2 Timer B Trigger Register (TBTRG)

Figure 13-3 illustrates the Timer B data registers (TBDATA0H/L).

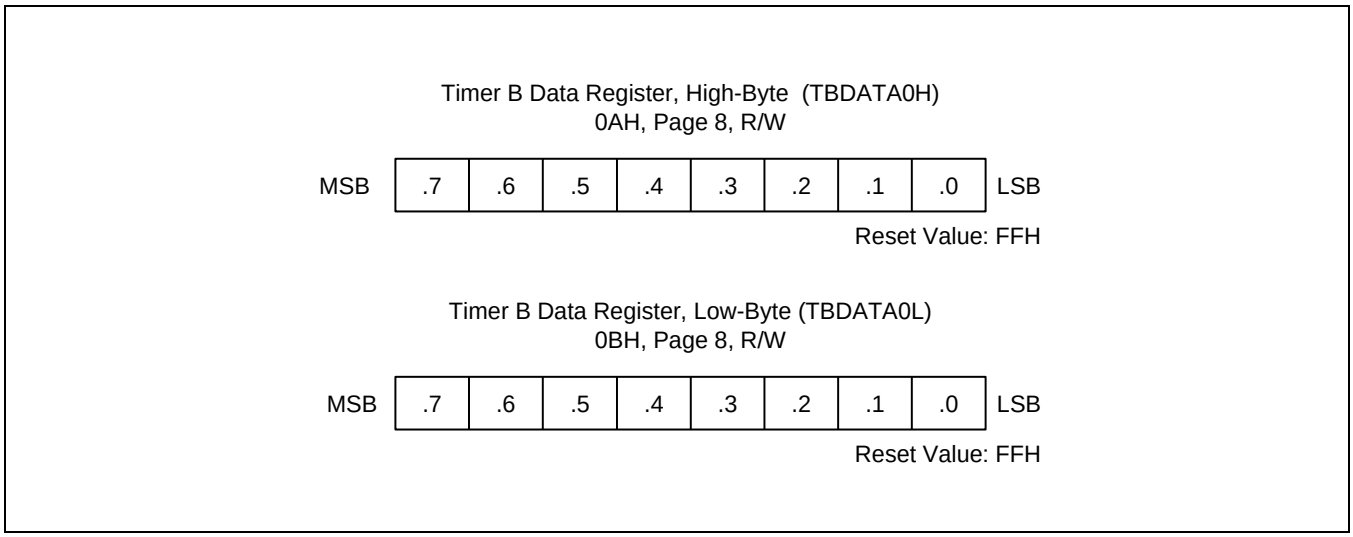


Figure 13-3 Timer B Data Registers (TBDATA0H/L)

Figure 13-4 illustrates the Timer B data registers (TBDATA1H/L).

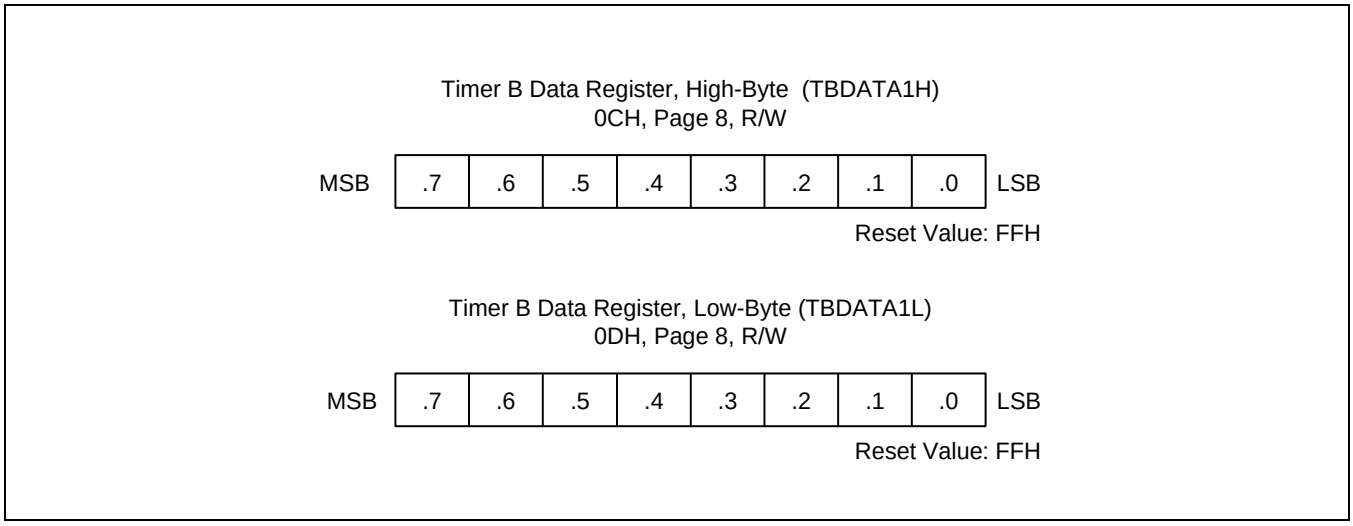
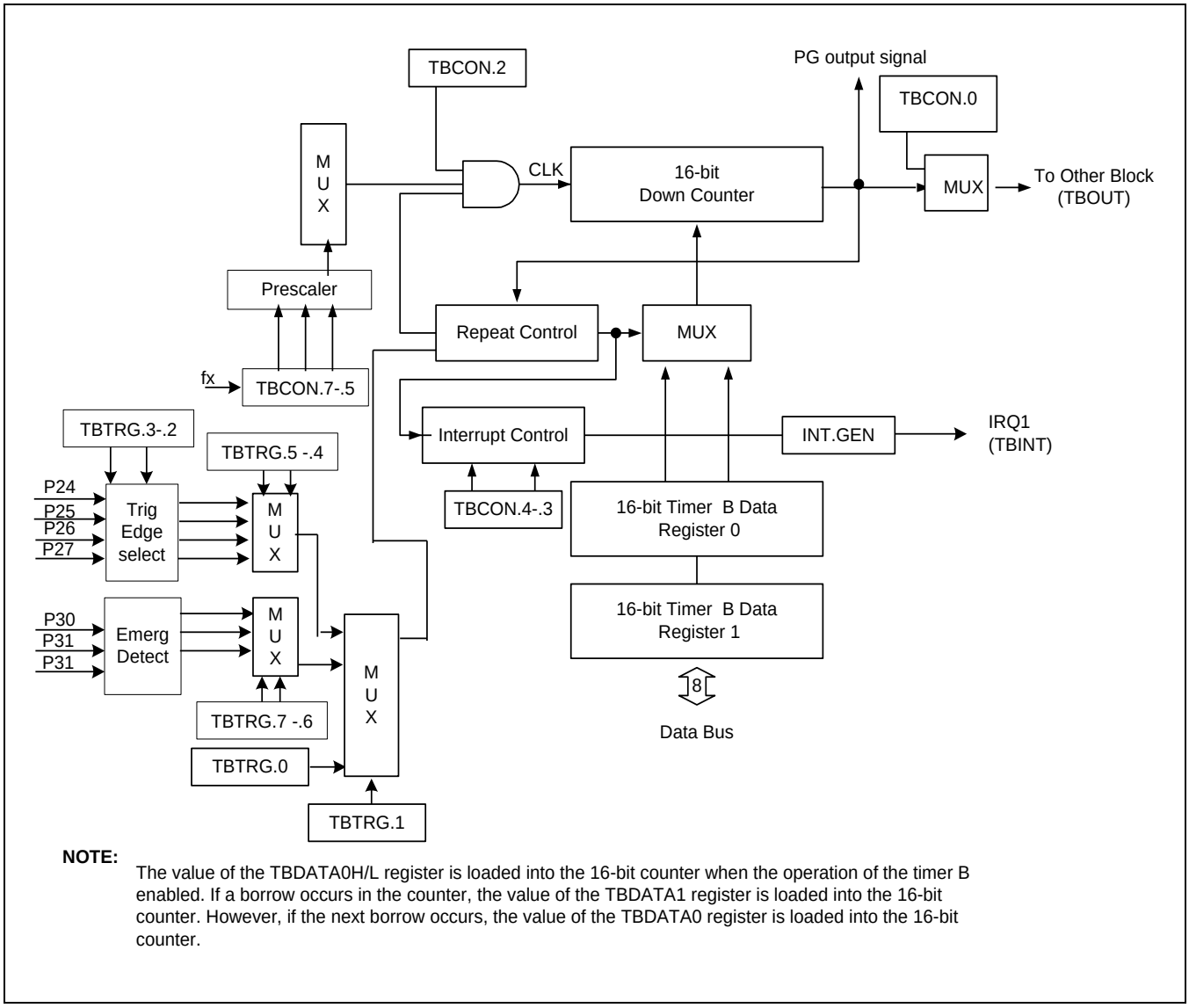


Figure 13-4 Timer B Data Registers (TBDATA1H/L)

13.2 Block Diagram

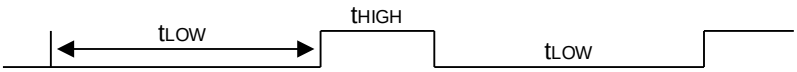
Figure 13-5 illustrates the functional block diagram of Timer B.



NOTE: The value of the TBDATA0H/L register is loaded into the 16-bit counter when the operation of the timer B enabled. If a borrow occurs in the counter, the value of the TBDATA1 register is loaded into the 16-bit counter. However, if the next borrow occurs, the value of the TBDATA0 register is loaded into the 16-bit counter.

Figure 13-5 Timer B Functional Block Diagram

13.2.1 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} = (TBDATA0 + 1) × 1/fx, 0H < TBDATA0 < 10000H, where fx = The selected clock.

t_{HIGH} = (TBDATA1 + 1) × 1/fx, 0H < TBDATA1< 10000H, where fx = The selected clock.

When TBOF = 1,

t_{LOW} = (TBDATA1 + 1) × 1/fx, 0H < TBDATA1 < 10000H, where fx = The selected clock.

t_{HIGH} = (TBDATA0 + 1) × 1/fx, 0H < TBDATA0 < 10000H, where fx = The selected clock.

To make t_{LOW} = 24 μs and t_{HIGH} = 15 μs. f_{OSC} = 4 MHz, fx = 4 MHz/4 = 1 MHz

When TBOF = 0,

t_{LOW} = 24 μs = (TBDATA0 + 1/fx = (TBDATA0 + 1) × 1 μs, TBDATA0 = 23.

t_{HIGH} = 15 μs = (TBDATA1 + 1/fx = (TBDATA1 + 1) × 1 μs, TBDATA1 = 14.

When TBOF = 1,

t_{HIGH} = 15 μs = (TBDATA0 + 1/fx = (TBDATA0 + 1) × 1 μs, TBDATA0 = 14.

t_{LOW} = 24 μs = (TBDATA1 + 1/fx = (TBDATA1 + 1) × 1 μs, TBDATA1 = 23.

Figure 13-6 illustrates the Timer B output flip-flop waveforms in Repeat Mode.

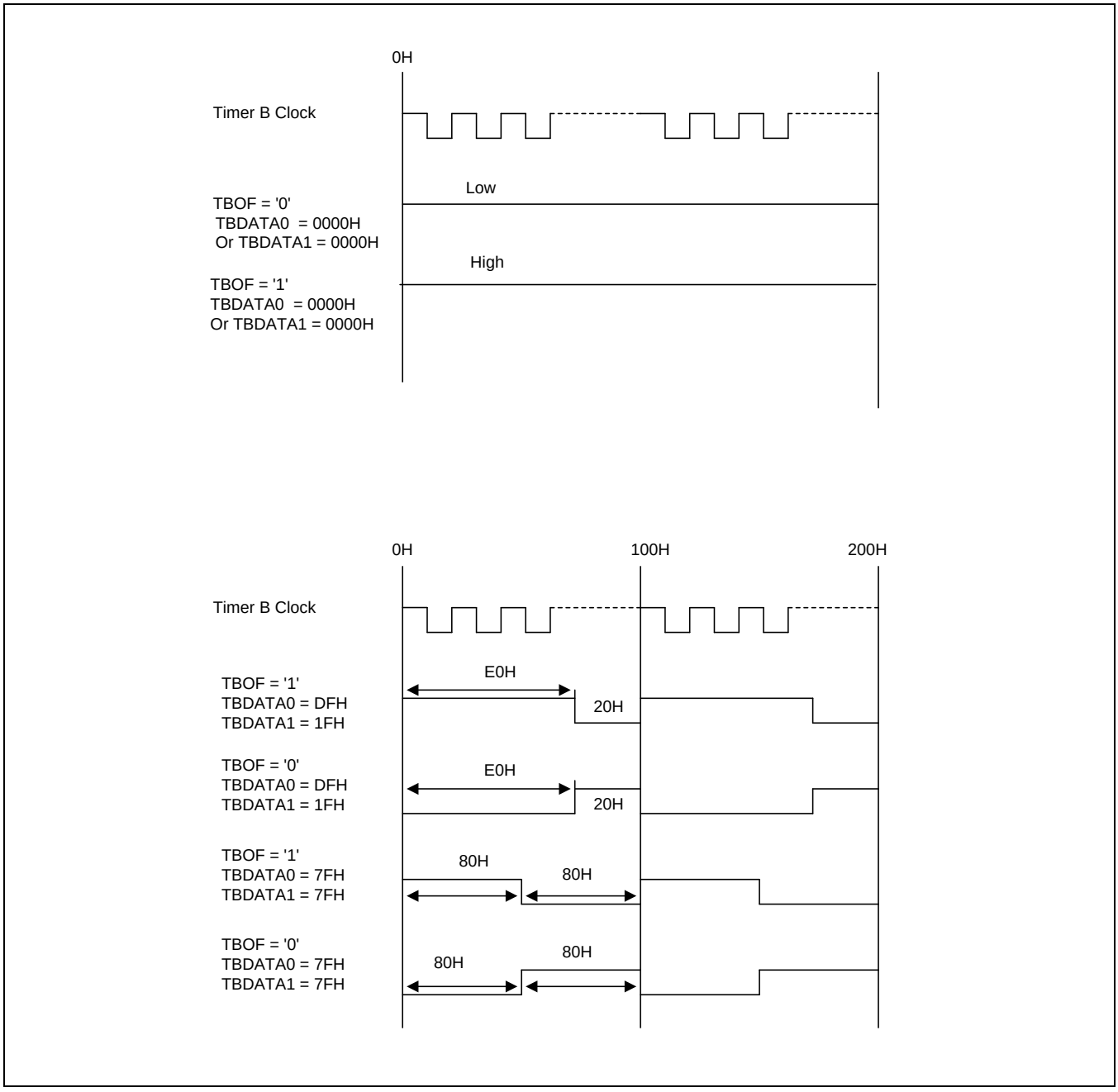
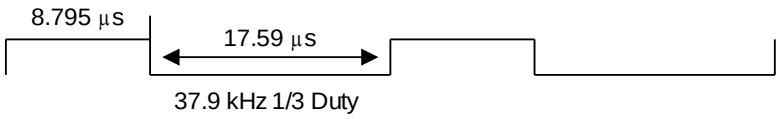


Figure 13-6 Timer B Output Flip-Flop Waveforms in Repeat Mode

Example 13-1 To Generate 38 kHz, 1/3 Duty Signal through P2.1

This example sets Timer B to the repeat mode, it sets the oscillation frequency as the Timer B clock source, and 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 μs)
- TBDATA0 = 17.59 μs/0.25 μs = 70.36, TBDATA1 = 8.795 μs/0.25 μs = 35.18
- Sets P2.1 to TBOUT mode

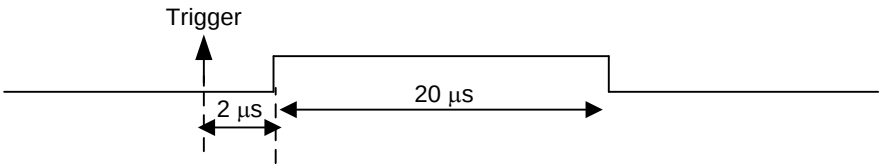
```
      ORG     0100H                ; Reset address
START DI
      •
      •
      •
      LD      TBDATA0H, #0         ; Set 17.5 μs
      LD      TBDATA0L, #(70-1)
      LD      TBDATA1H, #0         ; Set 8.75 μs
      LD      TBDATA1, #(35-1)
      LD      TBCON, #00000111B    ; Clock Source ← fxx
      LD      TBTRG, #0            ; Disable Timer B interrupt.
                                   ; Select repeat mode for Timer B.
                                   ; Start Timer B operation.
                                   ; Set Timer B Output flip-flop (TBOF) high.

      OR      P2CONL, #00000100B   ; Set P2.1 to TBOUT mode.
                                   ; This command generates 38 kHz, 1/3 duty pulse signal
                                     through P2.1.
      •
      •
      •
```

Example 13-2 To Generate A One Pulse Signal through P2.1

This example sets Timer B to the one shot mode, it sets the oscillation frequency as the Timer B clock source, and make a 38 kHz, 1/3 Duty carrier frequency.

The program parameters are:



- Timer B is used in one shot mode
- Oscillation frequency is 4 MHz (1 clock = 0.25 μs)
- Delay time from trigger detection is 2 μs
- TBDATA1 = 20 μs/0.25 μs = 80, TBDATA0 = 2 μs/0.25 = 8
- Sets P2.1 to TBPWM Mode

```
      ORG     0100H                       ;       Reset address
START DI
      •
      •
      •
      LD      TBDATA1H, #0                ;       Set 20 μs
      LD      TBDATA1L, #(80-1)
      LD      TBDATA0H, #0                ;       Set 2 μs
      LD      TBDATA0L, #(8-1)
      LD      TBCON, #00000101B           ; Clock Source ← fOSC
                                          ; Disable Timer B interrupt.
                                          ; Select one shot mode for Timer B.
                                          ; Enable Timer B operation.
                                          ; Set Timer B output flip-flop (TBOF) high
              OR      P2CONL, #00000100B  ; Set P2.1 to TBOUT mode.
Pulse_out:    LD      TBTRG, #0000010B    ; Set valid trigger source P0.0 falling edge.
                                          ; Trigger by software
```

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". If you want to service watch timer overflow interrupt (IRQ4, vector E2H), then set the WTCON.6 to "1". Software could clear the watch timer overflow interrupt pending condition in the interrupt service routine by writing a "0" to the interrupt pending bit (WTCON.0).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". 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 functions in high-speed mode by generating an interrupt every 1.995 ms. High-speed mode is useful for timing events for program debugging sequences.

The components of watch timer are:

- Real Time and Watch-Time Measurement
- Using a Main Clock Source
- I/O pin for Buzzer Output Frequency Generator (BUZ)
- Timing Tests in High-Speed Mode
- Watch timer overflow interrupt (IRQ4, vector E2H) generation
- Watch timer control register, WTCON (set 1, bank 1, FEH, Read/Write)

14.1.1 Watch Timer Control Register

You use the watch timer control register (WTCON) to:

- Select the watch timer interrupt time and Buzzer signal,
- Enable or disable the watch timer function.

WTCON is located in set 1, bank 1 at address FEH, and is read/write addressable using register addressing mode.

A reset clears WTCON to "00H". This disables the watch timer.

Therefore, if you want to use the watch timer, you should write appropriate value to WTCON.

Figure 14-1 illustrates the watch timer control register.

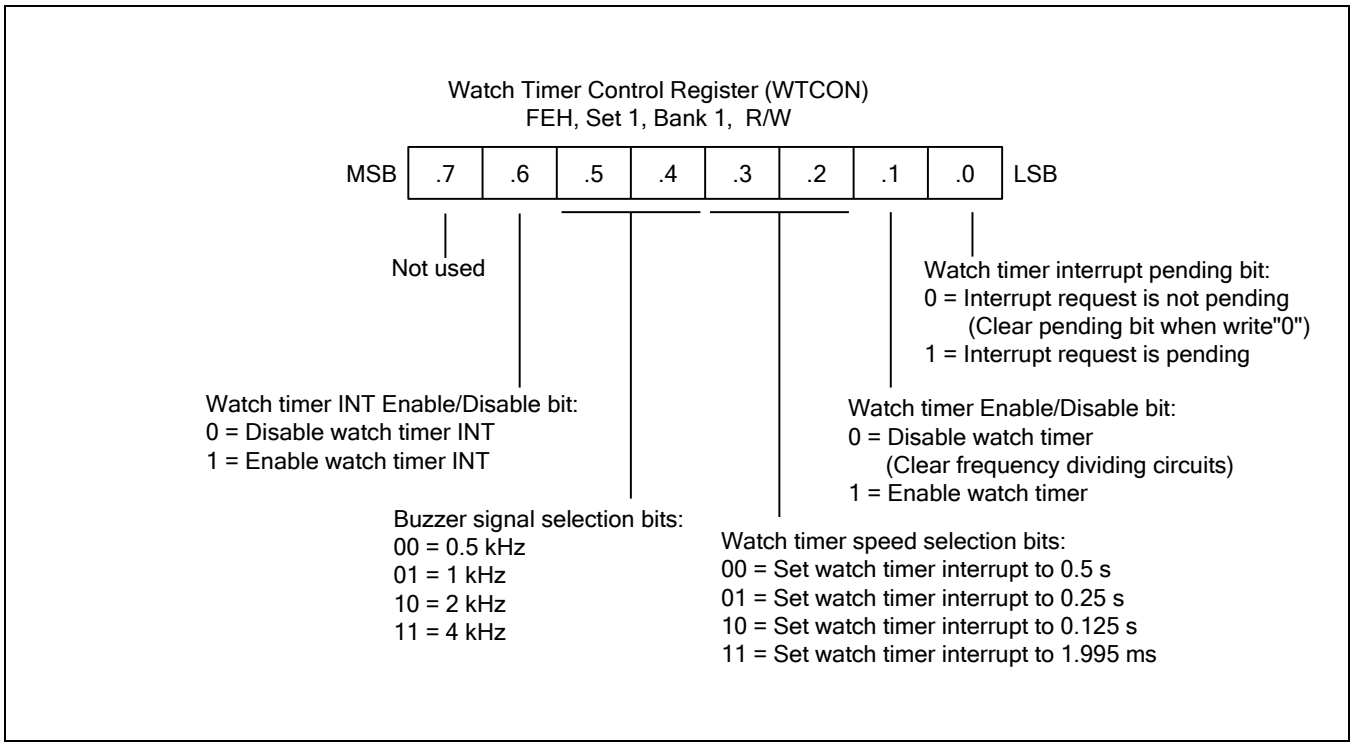


Figure 14-1 Watch Timer Control Register (WTCON)

14.1.2 Watch Timer Circuit Diagram

Figure 14-2 illustrates the circuit diagram of watch timer.

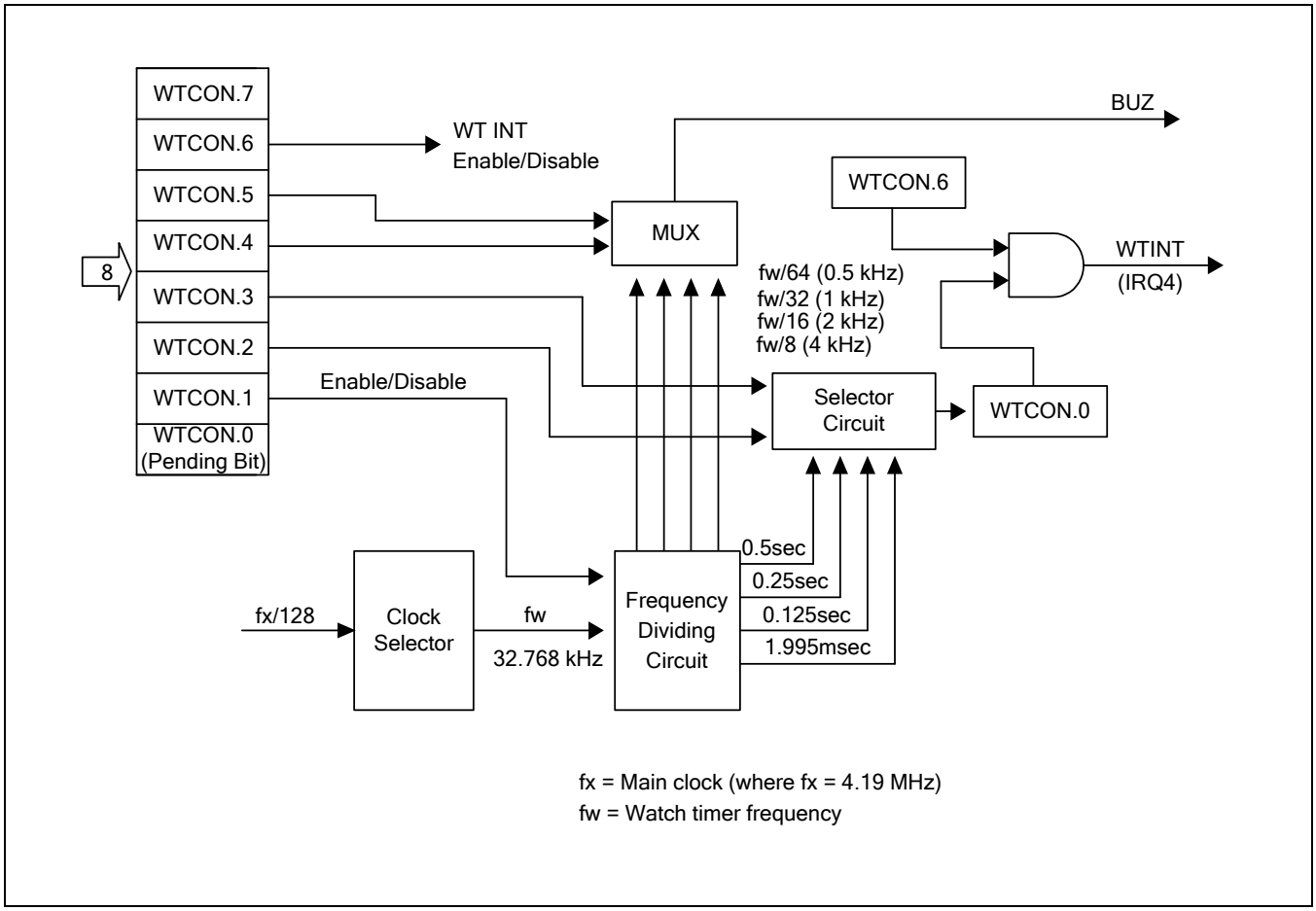


Figure 14-2 Watch Timer Circuit Diagram

15 Stop Wake-Up Timer

15.1 Overview

The S3F8S39/S3F8S35 has a 10-bit Stop Wake-Up Timer (SWT). The features of SWT are:

- Use this only in Stop Mode
- Use Ring oscillator as clock source
- Generates interrupt to wake up CPU from Stop Mode
- Selectable clock divider

15.1.1 Function Description

The SWT works only in Stop Mode. A 32 kHz internal Ring Oscillator works as its clock source in Stop Mode. The SWT can wake up CPU from Stop Mode when the counter reaches 0x3FF. And then, CPU executes the instruction next to STOP instruction, after SWT interrupt service routine returns.

15.1.2 Stop Wake-Up Timer Control Register

You can use the Stop Wake-Up Timer Control Register (SWTCON) to:

- Enable/Disable Stop Wake-Up Timer
- Interrupt Enable/Disable
- Program the clock prescaler for Stop Wake-Up Timer

SWTCON is located at address FAH, Bank 0, Set 1, and is Read/Write addressable by using Register addressing mode. A reset clears SWTCON to "00H". This disables Stop Wake-Up Timer and Stop Wake-Up interrupt.

You can enable Stop Wake-Up Timer by writing a "1" to SWTCON.7. SWTCON.5 controls the Stop Wake-Up Timer interrupt. If it is set, then it generates Stop Wake-Up timer interrupt when counter reaches to 0x3FF.

You can use SWTCON.3-0to set the clock prescaler of the Stop Wake-Up Timer. You should not keep the clock prescaler value of SWTPS more than 12. Any value more than 12 is invalid.

Figure 15-1 illustrates the Stop Wake-Up Timer Control Register (SWTCON).

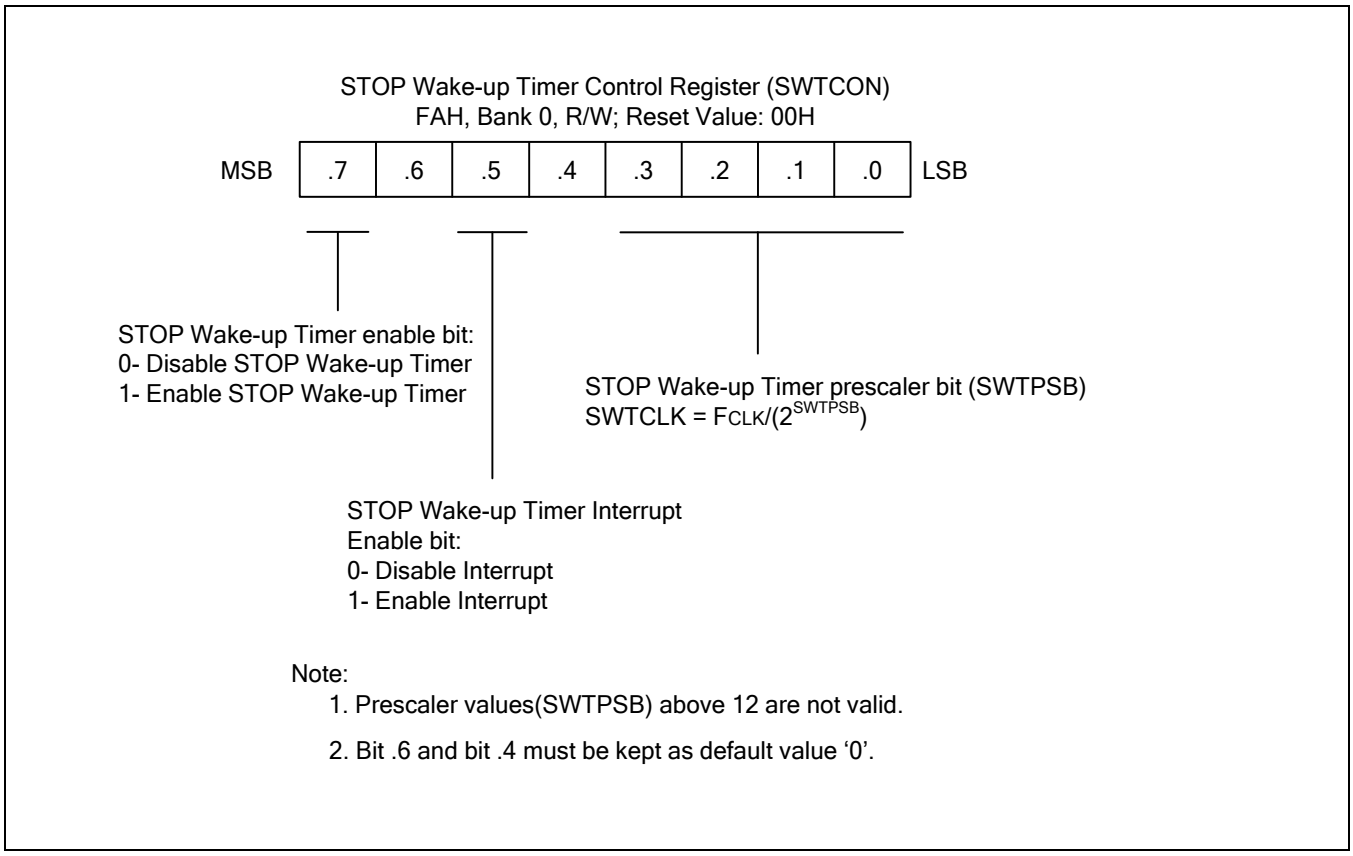


Figure 15-1 Stop Wake-Up Timer Control Register (SWTCON)

Figure 15-2 illustrates the functional block diagram of Stop Wake-Up Timer.

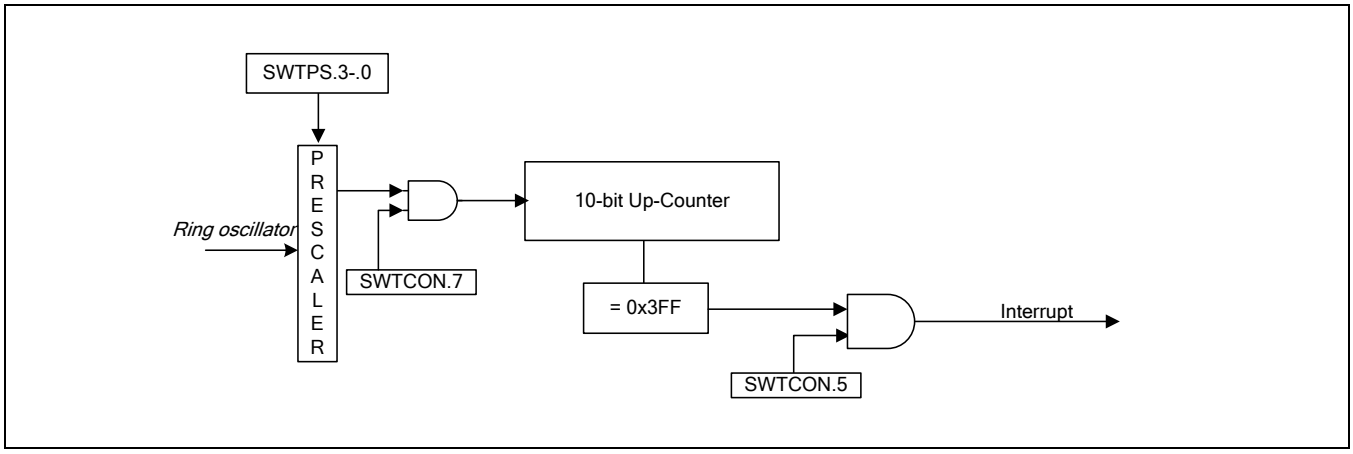


Figure 15-2 Stop Wake-Up Timer Functional Block Diagram

Table 15-1 lists the Stop Wake-Up Timer prescaler select.

Table 15-1 Stop Wake-Up Timer Prescaler Select

SWTCON.3	SWTCON.2	SWTCON.1	SWTCON.0	Number of 32 kHz Ring Oscillator Cycles	Typical Time-out
0	0	0	0	2048	64 ms
0	0	0	1	4096	128 ms
0	0	1	0	8109	256 ms
0	0	1	1	16384	512 ms
0	1	0	0	32768	1024 ms
0	1	0	1	65536	2048 ms
0	1	1	0	131072	4096 ms
0	1	1	1	262144	8192 ms
1	0	0	0	524288	16384 ms
1	0	0	1	1048576	32768 ms
1	0	1	0	2097152	65536 ms

15.1.3 Stop Mode Wake Up

Before entering Stop Mode, you should follow these steps to set up the Stop Mode Wake-up Timer:

1. Enables Ring oscillator by setting ROSCCON to 0xc0.
2. Enables Stop Mode Wake-up Timer and its interrupt. Set the prescaler to determine the wake-up time.
3. Enters the Stop Mode immediately after SWT start.

Then it stops the CPU and Stop Wake-Up Timer continues counting until it reaches 0x3FF (10-bit).The SWT interrupt wakes up CPU and executes its interrupt service routine.

1. Disables Stop Wake-Up Timer interrupt.
2. Does not allow other instructions.
3. Interrupt returns.

Figure 15-3 illustrates the SWT operation sequence.

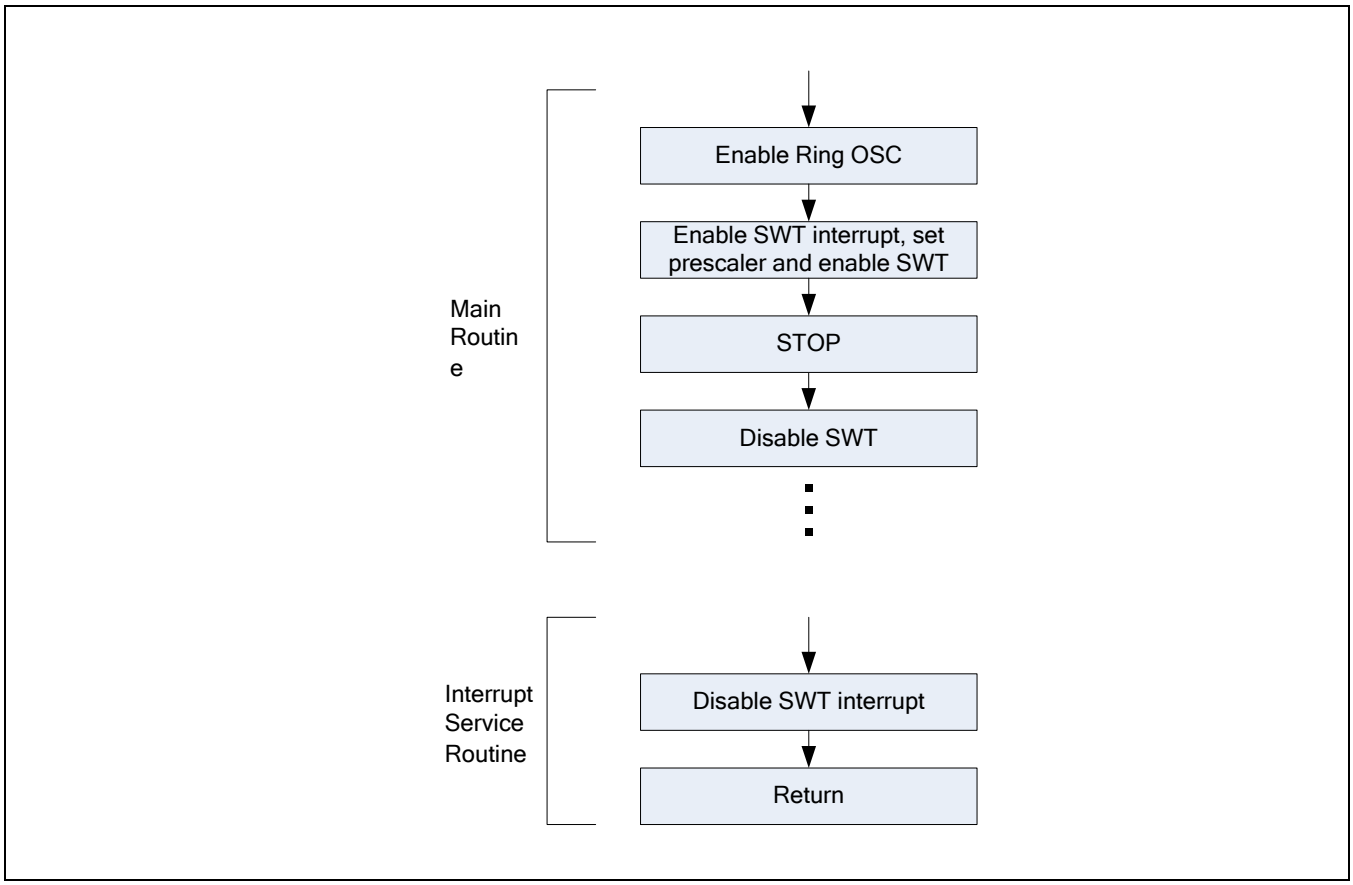


Figure 15-3 Stop Wake-Up Timer Operation Sequence

Example 15-1 Using Stop Wake-Up Timer

```
;------<< Interrupt Vector Address >>
              VECTOR  CCH, INT_SWT

;------<< Smart Option >>
       ORG     003CH
       DB      0FFH                          ; 003CH, must be initialized to FF
       DB      0FFH                          ; 003DH, must be initialized to FF
       DB      0FFH                          ; 003EH, Normal reset vector, ISP disabled
       DB      018H                          ; 003FH, LVR disabled, external oscillator
       ORG     0100H
RESET: DI                                    ; disable interrupt
       LD      BTCON, #10100011B             ; Watchdog disable
       •
       •
       •
       EI                                    ; Enable interrupt

;------<< Main loop >>
MAIN:  •
       •
       •
       SB1
       LD      STPCON, #01011010B
       SB0
       LD      ROSCCON, #11000000B           ; Ring oscillator enabled
       LD      SWTCON, #10100000B            ; SWT enabled, interrupt enabled, SWTPSB = 0
       STOP                                  ; Immediately after SWT enable
       NOP
       NOP
       AND     SWTCON, #00001111B            ; Disable SWT
       •
       •
       •
       JR      t, MAIN

;-----<< Interrupt Service Routine >>
INT_SWT:
       AND     SWTCON, #00001111B            ; Disable SWT interrupt
       IRET
```

16 A/D 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 sixteen input channels to equivalent 10-bit digital values. The analog input level must lie between the V_{DD} and V_{SS} values.

The components A/D converter contains:

- Analog comparator with successive approximation logic
- Digital to analog (D/A) converter logic
- ADC Control register (ADCON)
- 16 multiplexed analog data input pins (ADC0–ADC15)
- 10-bit A/D conversion data output register (ADDATAH/L):

The procedure to initiate an analog-to-digital conversion is:

1. Write the channel selection data in the ADCON register to select one of the 16 analog input pins (ADCn, n = 0–15)
2. Set the conversion start or enable bit, ADCON.0. The Read/Write ADCON register is placed in D2H, Set1.

To initiate an analog-to-digital conversion procedure, you write the channel selection data in the A/D converter control register ADCON to select one of the sixteen analog input pins (ADCn, n = 0–15) and set the conversion start or enable bit, ADCON.0. The read-write ADCON register is located at address D2H, Set1.

During a normal conversion, ADC logic initially sets the successive approximation register to 200H (the approximate half-way point of a 10-bit register). It then updates this register 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.7–4) in the ADCON register. To start the A/D conversion, you should set the enable bit, ADCON.0. When it completes the conversion, ACON.3, it automatically sets the end-of-conversion (EOC) bit to 1 and then dumps the result into the ADDATA register where it can be read. The A/D converter then enters an idle state. Remember to read the contents of ADDATA before another conversion starts. Otherwise, it overwrites the previous result with the next conversion result.

NOTE: Because the ADC does not use sample-and-hold circuitry, it is important that any fluctuations in the analog level at the ADC0–ADC15 input pins during a conversion procedure should be kept to an absolute minimum. Any change in the input level, perhaps due to circuit noise, invalidates the result.

16.1.1 Using A/D Pins for Standard Digital Input

You can alternatively use the inputs pins of ADC module as digital input in port 0, port 1, and P3.0–P3.5.

16.1.2 A/D Converter Control Register

The A/D converter control register (ADCON), is located at address D2H.

ADCON has four functions:

- Bits 7-4 select an analog input pin (ADC0–ADC15).
- Bit 3 indicates the status of the A/D conversion.
- Bits 2-1 select a conversion speed.
- Bit 0 starts the A/D conversion.

You can select only one analog input channel at a time. . You can dynamically select any one of the sixteen analog input pins (ADC0–ADC15) by manipulating the 4-bit value for ADCON.7–ADCON.4.

Figure 16-1 illustrates the A/D Converter Control Register (ADCON).

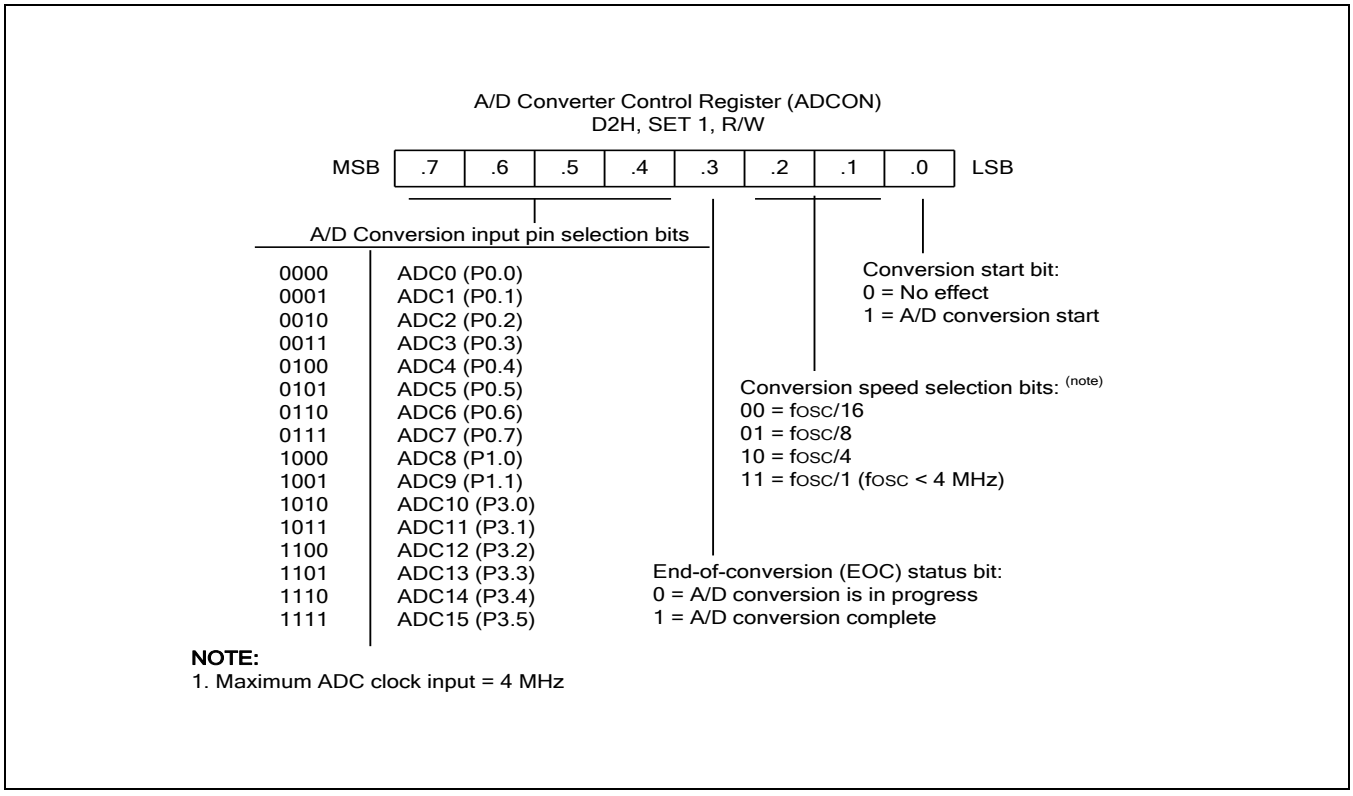


Figure 16-1 A/D Converter Control Register (ADCON)

16.1.3 Internal Reference Voltage Levels

In the ADC function block, it compares the analog input voltage level to the reference voltage. The analog input level should remain within the range V_{SS} to V_{DD}.

It generates different reference voltage levels internally along the resistor tree during the analog conversion process for each conversion step. The reference voltage level for the first bit conversion is always 1/2 V_{DD}.

Figure 16-2 illustrates the circuit diagram of A/D Converter.

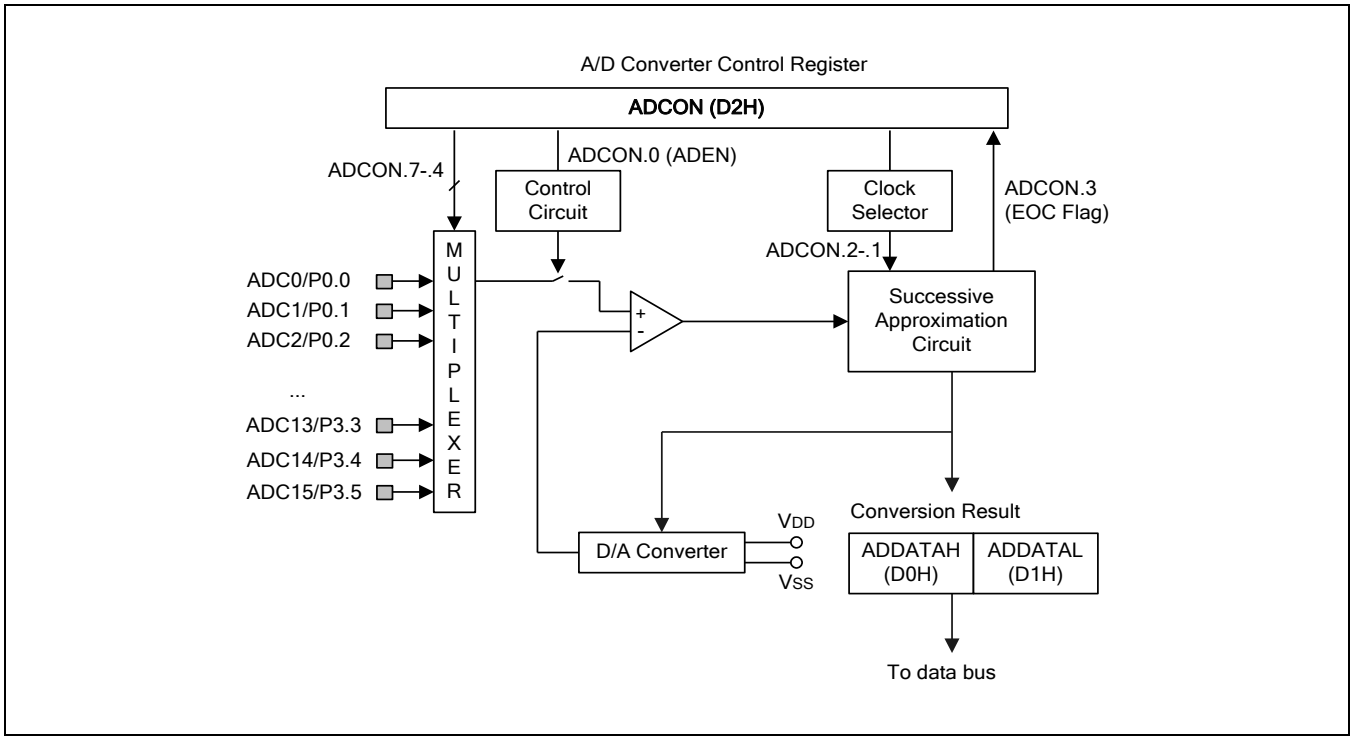


Figure 16-2 A/D Converter Circuit Diagram

Figure 16-3 illustrates the A/D Converter Data Register (ADDATAH/L).

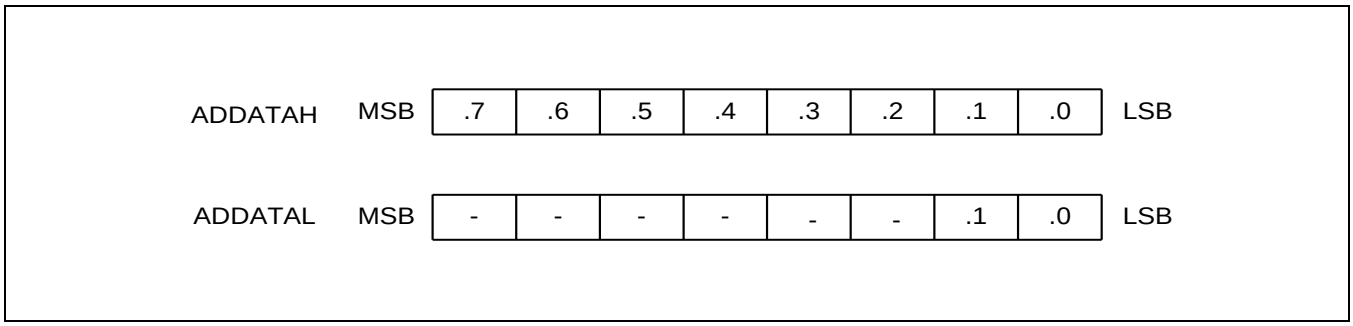


Figure 16-3 A/D Converter Data Register (ADDATAH/L)

Figure 16-4 illustrates the timing diagram of A/D Converter.

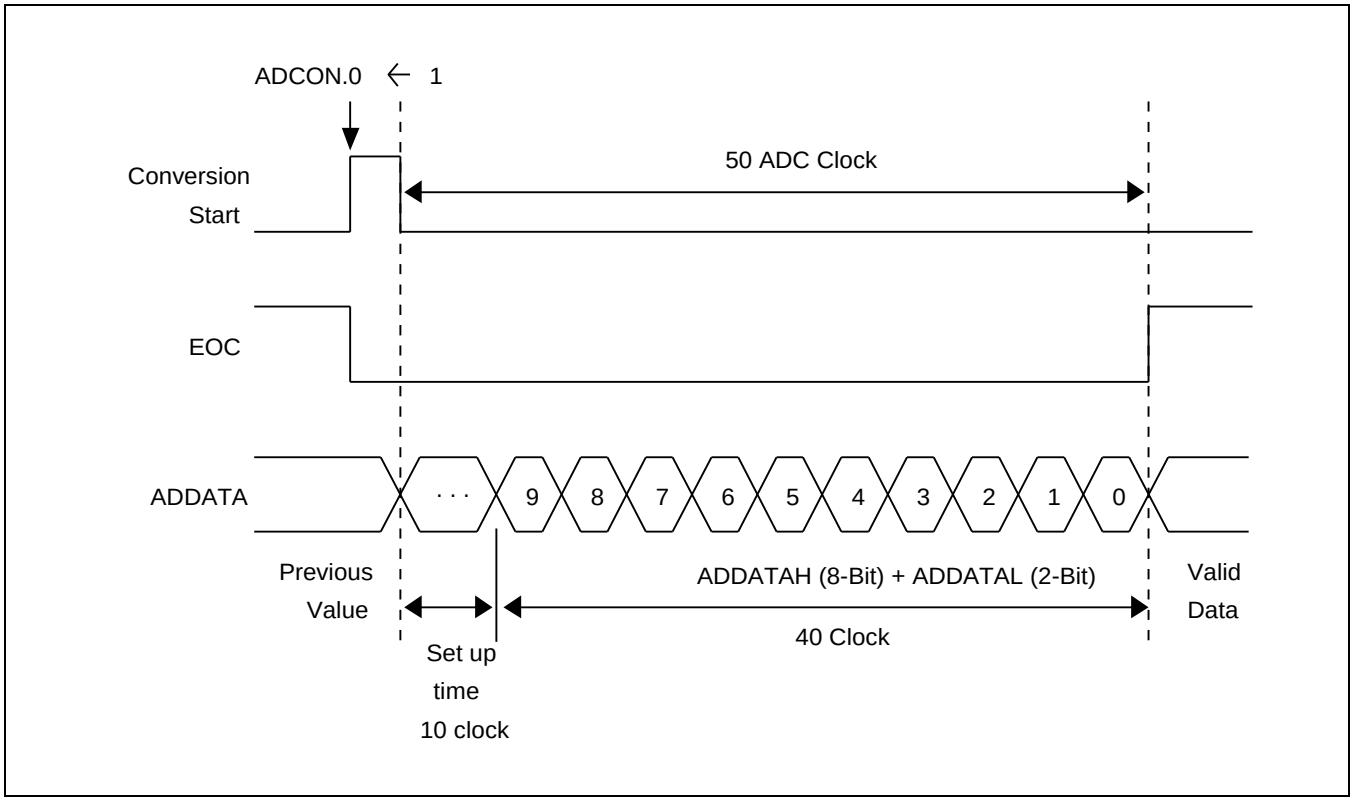


Figure 16-4 A/D Converter Timing Diagram

16.1.4 Conversion Timing

The A/D conversion process requires four steps (four clock edges) to convert each bit and 10 clocks to step-up A/D conversion. Therefore, it requires total of 50 clocks to complete a 10-bit conversion: With a 12 MHz CPU clock frequency, one clock cycle is 333 ns (4/fxx). If each bit conversion requires four clocks, it calculates the conversion rate as:

- 4 clocks/bit x 10-bits + step-up time (10 clock) = 50 clocks
- 50 clock × 333 μs = 16.65 μs at 12 MHz, 1 clock time = 4/fxx (assuming ADCON.2–1 = 10)

16.1.5 Internal A/D Conversion Procedure

The procedure for internal A/D Conversion is:

1. Analog input should remain between the voltage range of V_{SS} and V_{DD}.
2. Configure the analog input pins to input mode by making the appropriate settings in P0CONH, P0CONL, P1CONL, P3CONH, and P3CONL registers.
3. Before the conversion operation starts, you should select one of the sixteen input pins (ADC0–ADC15) by writing the appropriate value to the ADCON register.
4. When it completes the conversion, (16 clocks have elapsed), the EOC flag is set to "1", so that a check can be made to verify that the conversion was successful.
5. Loads the converted digital value to the output register, ADDATAH (8-bit) and ADDATAL (2-bit), and then the ADC module enters an idle state.
6. You can now read the digital conversion result from the ADDATAH and ADDATAL register.

Figure 16-5 illustrates the recommended A/D Converter circuit for highest absolute accuracy.

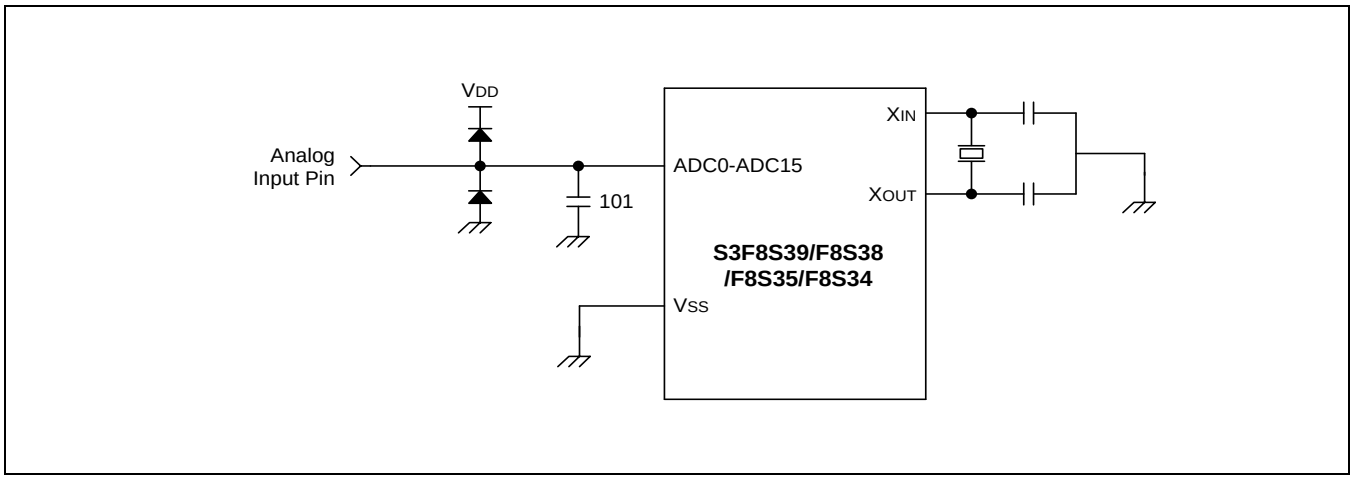


Figure 16-5 Recommended A/D Converter Circuit for Highest Absolute Accuracy

Example 16-1 Configuring A/D Converter

```
;--------------<< Interrupt Vector Address >>
              VECTOR  D4H, INT_TIMER0
;--------------<< Smart Option >>
              ORG     003CH
              DB      0FFH                    ; 003CH, must be initialized to FF
              DB      0FFH                    ; 003DH, must be initialized to FF
              DB      0FFH                    ; 003EH,
              DB      0FFH                    ; 003FH, enable LVR 3.9 V, internal RC oscillator 8MHz
              ORG     0100H
RESET:        DI                              ; Disable interrupt
              LD      BTCON,#10100011B        ; Watchdog disable
              •
              •
              •
              LD      P0CONH,#10101010B       ; Configure P0.4-P0.7 AD input
              LD      P0CONL,#10101010B       ; Configure P0.0-P0.3 AD input

              EI                              ; Enable interrupt
;--------------<< Main loop >>
MAIN:         •
              •
              •
              CALL    AD_CONV                 ; Subroutine for AD conversion
              •
              •
              •
              JR      t, MAIN ;
AD_CONV:      LD      ADCON, #00000001B       ; Select analog input channel → P0.0
                                                select conversion speed → fOSC/16
                                                set conversion start bit
              NOP
                                              ; If you select conversion speed to fOSC/16
                                                at least one NOP must be included

CONV_LOOP:    TM      ADCON, #00001000B       ; Check EOC flag
              JR      Z, CONV_LOOP            ; If EOC flag = 0, jump to CONV_LOOP until EOC flag = 1
              LD      R0, ADDATAH             ; High 8 bits of conversion result are stored
                                              ; to ADDATAH register
              LD      R1, ADDATAL             ; Low 2 bits of conversion result are stored
                                              ; to ADDATAL register
              LD      ADCON, #00010011B       ; Select analog input channel → P0.1
                                              ; Select conversion speed → fOSC/8
                                              ; Set conversion start bit
CONV_LOOP2:   TM      ADCON, #00001000B       ; Check EOC flag
              JR      Z, CONV_LOOP2
              LD      R2, ADDATAH
              LD      R3, ADDATAL
              •
              •
              •
              RET                             ;

INT_TIMER0:   •                               ;
              •                               ;
              •                               ; Pending bit clear
              IRET                            ;
              •
              •
              END
```

17 Serial Peripheral Interface

17.1 Overview

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the Micro controller (MCU) and peripheral devices or between several Zilog devices.

- Full-duplex, 4-wire Synchronous Data Transfer
- Master or Slave Operation
- Least Significant Bit (LSB) First or Most Significant Bit (MSB) First Data Transfer
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Mode Fault Flag bit
- Wake-up from Idle Mode
- Double Speed Master SPI Mode

The S3F8S39/S3F8S35 SPI circuit supports byte serial transfers in either Master or Slave modes. *Figure 17-1*. Illustrates the block diagram of the SPI circuit the block contains buffer for receive data for maximum flexibility and throughput. You can configure S3F8S39/S3F8S35as either an SPI Master or Slave. The external interface consists of Master-Out/Slave-In (MOSI), Master-In/Slave-Out (MISO), Serial Clock (SCK), and Slave Select (NSS). You get the receive buffer contents by reading the SPI Data register (Refer to *Figure 17-5* for more information).

You can activate the SPI modes by setting the appropriate bits in the SPI Control Register (Refer to *Figure 17-5* for more information).

Figure 17-1 illustrates the block diagram of SPI.

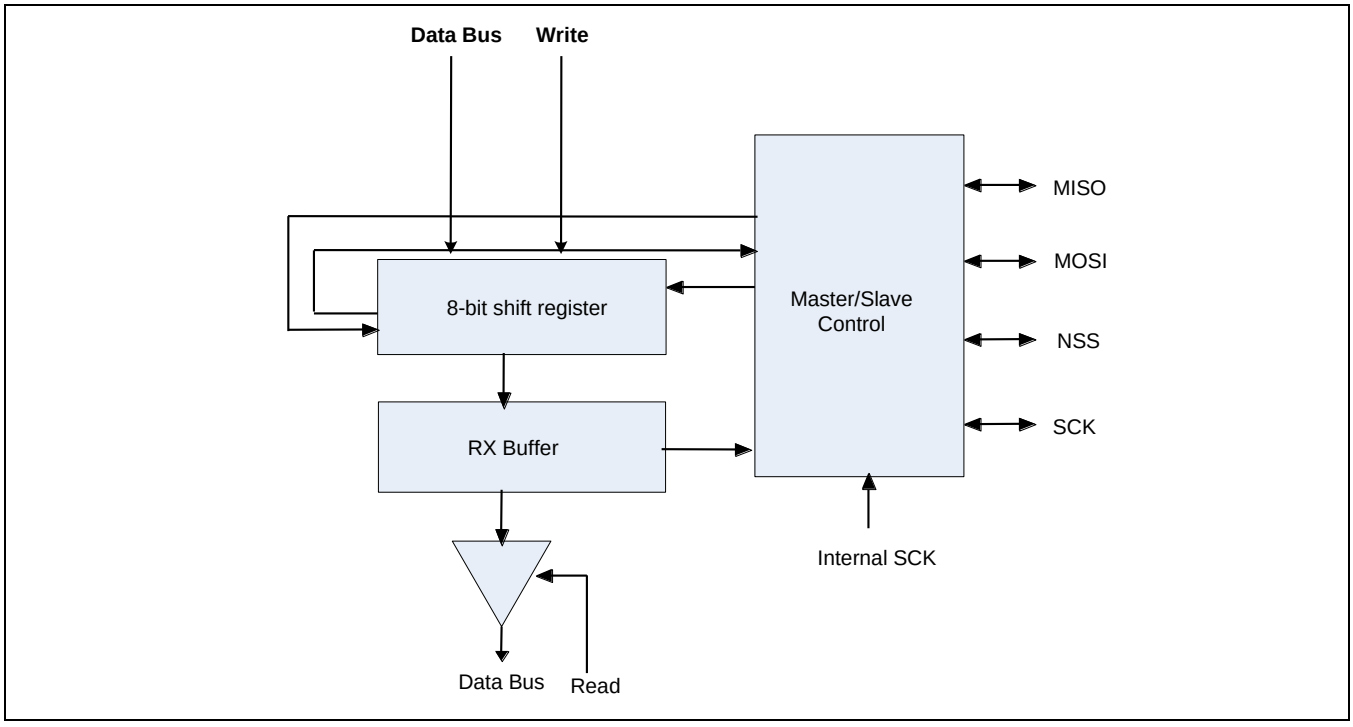


Figure 17-1 SPI Block Diagram

Figure 17-2 illustrates the data timing of SPI.

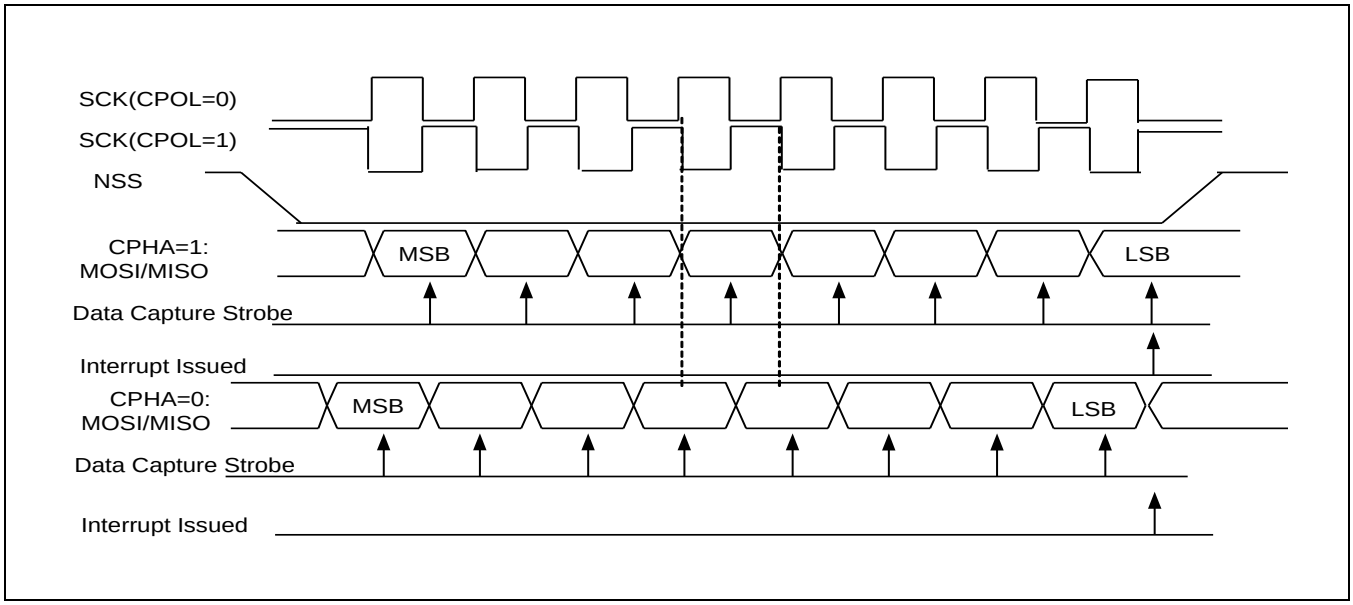


Figure 17-2 SPI Data Timing

17.2 Operation as an SPI Master

SPI Master can initiate a byte/data transfer and this is done by the Master writing to the SPI Data register. The Master shifts out 8 bits of data along with the serial clock SCK for the Slave. The Master's outgoing byte is replaced with an incoming one from a Slave device. When the last bit is received, the shift register contents are transferred to the Receive Buffer and an interrupt is generated. The receive data must be read form the SPI Data Register before it transfers the next bye of data to the receive buffer or data is lost.

When operating as a Master, an active LOW Slave Select (NSS) must be generated to enable a Slave for a byte transfer. This Slave Select is generated under firmware control, and is not part of the SPI internal hardware. any available General purpose input output pin (GPIO) can be used for the Master's Slave Select Output.

When the Master writes to the SPI Data Register and if the shift register is not busy shifting a previous byte, then it loads the data into the shift register and begins the shifting. If the shift register is busy, then it generates a write collision error. The transfer continues undisturbed, and the write data that caused the error is not written to the shifter.

The byte shifting and SCK generation are handled by the hardware (based on firmware selection of the clock source). Data is shifted out on the MOSI pin and the serial clock is output on the SCK pin. Data is received from the slave on the MISO pin.

17.3 Master SCK Selection

The Master SCK is programmable to one of eight clock settings, as described in *Table 17-2*. The frequency is selected with the Clock Select Bits of SPI control register and Double SPI Speed Bit of SPI status register. The hardware provides eight output clocks on the SCK pin for each byte transfer. Clock phase and polarity are selected by the CPHA and CPOL control bits (Refer to *Figure 17-3* for more information).

17.4 Operation as an SPI Slave

In Slave mode, the chip receives SCK from an external master on pin P3.2. Data from the master is shifted in on the MOSI pin, while data is being shifted out of the slave on the MISO pin. Additionally, it should assert the active LOW Slave Select to enable the slave for transmit. The Slave Select pin is P3.3. These pins are automatically configured by enabling SPI Enable bit.

In Slave mode, the data written to the SPI Data Register will be loaded into the shift register, only if the Slave Select is asserted (NSS LOW) and the shift register is not busy shifting a previous byte. If the register is busy, a write collision error is generated. The transfer continues undisturbed, and the write data that caused the error is not written to the shifter. If the Slave Select is not active when the data is loaded, data is not transferred to the shift register until Slave Select is asserted.

If the Slave Select is de-asserted before a byte transfer is complete, the transfer is aborted and no interrupt is generated. Whenever Slave Select is asserted, the data is automatically reloaded into the shift register.

You must select the clock phase and polarity to match the SPI master, using the control bits of SPICON (Refer to *Figure 17-3* for more information).

The SPI slave logic continues to operate in suspend, so if it enables the SPI interrupt, the device can go into suspend during a SPI slave transaction, and it wakes up at the interrupt that signals the end of the byte transfer.

17.5 SPI Status and Control

Figure 17-3 illustrates the SPI control register. *Figure 17-2* illustrates the timing diagram that shows the clock and data states for the various SPI modes.

17.6 SPI Interrupt

For SPI, an interrupt request is generated after a byte is received or transmitted. After the interrupt, the received data byte can be read from the SPI Data Register and the SPI interrupt flag bit is high.

Table 17-1 describes the SPI pin assignment.

Table 17-1 SPI Pin Assignment

SPI Function	GPIO Pin	Comment
Slave Select (NSS)	P3.3	For Master Mode, Firmware sets NSS, can be used as GPIO pin. For Slave Mode, NSS is an active LOW input.
Master Out, Slave In (MOSI)	P3.1	Data output for master, data input for slave.
Master In, Slave Out (MISO)	P3.0	Data input for master, data output for slave.
SCK	P3.2	SPI Clock: Output for master, input for slave.

Table 17-2 lists the SCK rate selection.

Table 17-2 SCK Rate Selection

SPICON.1–0	SPISTAT.0	SCK Rate
00	1	Fosc/2
00	0	Fosc/4
01	1	Fosc/8
01	0	Fosc/16
10	1	Fosc/32
10	0	Fosc/64
11	1	Fosc/128
11	0	Fosc/256

17.7 SPI System Errors

Two system errors can be detected by the SPI system. The first type of error arises in a multiple-master system when multiple SPI devices simultaneously try to be a master. This error is called a mode fault. The second type of error, write collision, indicates that an attempt was made to write data to the SPIDATA, while a transfer was in progress.

When the SPI system is configured as a master and the NSS input line goes to active low, a mode fault error has occurred-usually because two devices have attempted to act as master at the same time. In cases where more than one device is concurrently configured as a master, there is a chance of contention between two pin drivers. For push-pull CMOS drivers, this contention can cause permanent damage. The mode fault mechanism attempts to protect the device by disabling the drivers. It clears the master/slave selection bit in the SPICON and all four P3CON control bits that are associated with the SPI. If NSS is an input and is driven low when the SPI is in Master mode, this will also set the SPI Interrupt Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPIDATA).

Other precautions may need to be taken to prevent driver damage. If two devices are made masters simultaneously, mode fault does not help protect either one unless one of them selects the other as slave. The amount of damage possible depends on the length of time both devices attempt to act as master.

A write collision error occurs if it writes the SPIDATA when a transfer is in progress. Because it does not double buffer the SPIDATA in the transmit direction, writes to SPIDATA cause data to be written directly into the SPI shift register. Because this write corrupts any transfer in progress, it generates a write collision error. The transfer continues undisturbed, and the write data that caused the error is not written to the shifter.

A write collision is normally a slave error because a slave has no control over when a master initiates a transfer. A master knows when a transfer is in progress, so there is no reason for a master to generate a write-collision error, although the SPI logic can detect write collisions in both master and slave devices.

17.8 SPI Control Register

The control register for the SPI is called SPI Control Register (SPICON) at address F4H, Bank1.

It has the following control functions:

- Operating mode and SCK rate selection
- Clock Phase and Clock Polarity selection
- Data order selection
- SPI Enable/Disable
- SPI Interrupt Enable/Disable

A reset clears the SPICON value to "00H". So, if you want to use SPI module, you must write appropriate value to SPICON.

Figure 17-3 illustrates the SPI Control Register (SPICON).

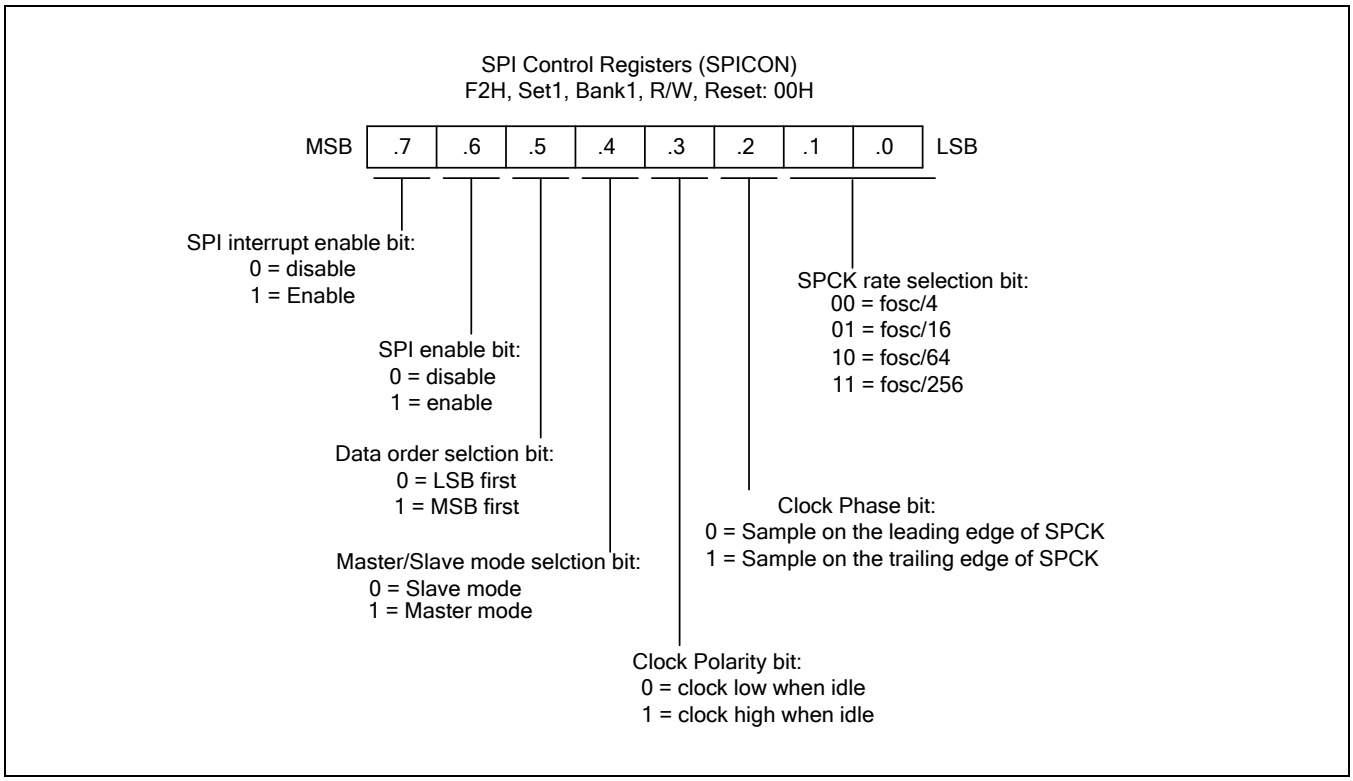


Figure 17-3 SPI Control Register (SPICON)

17.9 SPI Status Register

Two system errors can be detected by the SPI system. The first type of error arises in a multiple-master system when multiple SPI devices simultaneously try to be a master. This error is called a mode fault. The second type of error, write collision, indicates that an attempt was made to write data to the SPDR, while a transfer was in progress.

The control register for the SPI is called SPISTAT at address F5H, Bank1.

It has the following control functions:

- Double SPI speed
- SPI interrupt flag
- Write collision flag
- Mode fault flag

Clearing the Write Collision bit is accomplished by reading the SPISTAT (with Write Collision bit set) followed by an access of SPIDATA.

To clear the Mode fault bit, read the SPISTAT (with Mode Fault bit set), then write to the SPICON.

SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, it clears the SPIF bit by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPIDATA).

Figure 17-4 illustrates the SPI Status Register (SPISTAT).

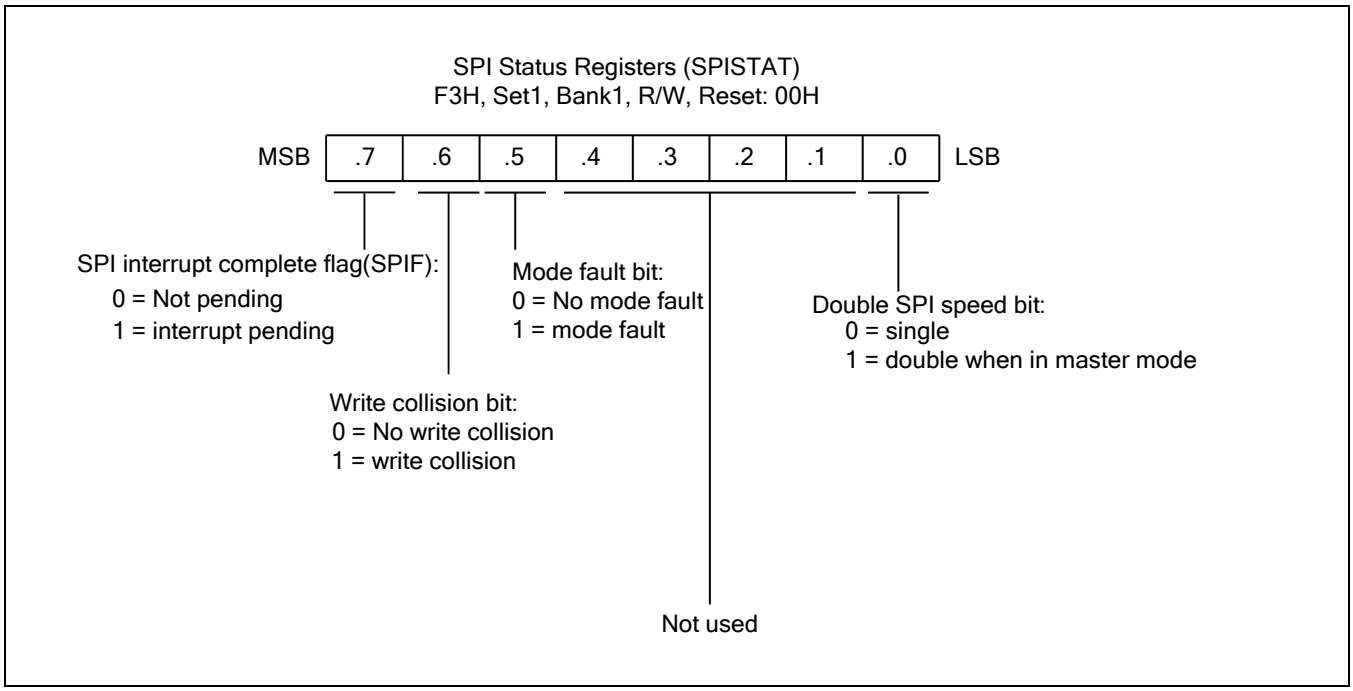


Figure 17-4 SPI Status Register (SPISTAT)

17.10 SPI Data Register

This register holds the SPI Data. The Firmware writes this register for transmitting data to External SPI Module. The Firmware reads the register to get data from the external SPI module.

SPIDATA is located at address F6H and is R/W addressable.

Figure 17-5 illustrates the SPI Data Register (SPIDATA).

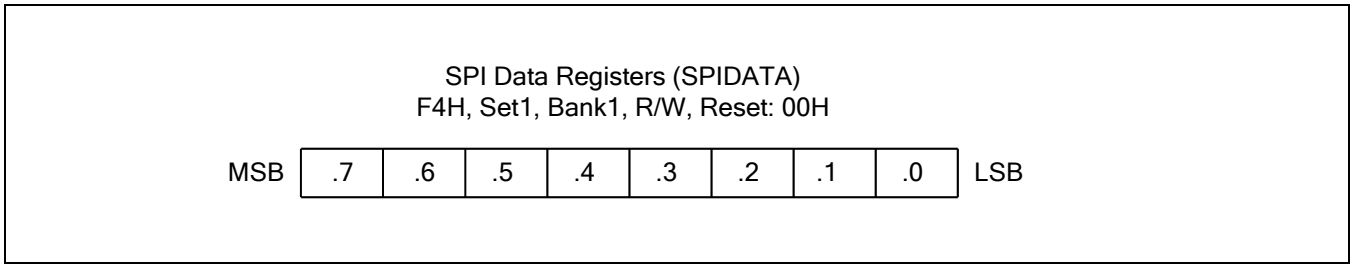


Figure 17-5 SPI Data Register (SPIDATA)

18 Inter Integrated Circuit-Bus Interface

18.1 Overview

The S3F8S39/S3F8S35 microcontrollers support a multi-master IIC-bus serial interface. A dedicated serial data line (SDA) and a serial clock line (SCL) transfer information between bus masters and peripheral devices that connects to the IIC-bus. The SDA and SCL lines are bi-directional.

In multi-master IIC-bus mode, multiple S3F8S39/S3F8S35 microcontrollers can receive or transmit serial data to or from slave devices. The master S3F8S39/S3F8S35 which initiates a data transfer over the IIC-bus is responsible for terminating the transfer. S3F8S39/S3F8S35 microcontrollers support standard bus arbitration functions.

To control multi-master IIC-bus operations, you should write the values to these registers:

- IIC-bus control register (ICCR)
- IIC-bus control/status register (ICSR)
- IIC-bus Tx/Rx data shift register (IDSR)
- IIC-bus address register (IAR)

When the IIC-bus is free, the SDA and SCL lines are both at High level. A High-to-Low transition of SDA initiates a Start condition. A Low-to-High transition of SDA initiates a Stop condition, while SCL remains steady at High level.

The bus master always generates Start and Stop conditions. A 7-bit address value in the first data byte that is put onto the bus after the Start condition is initiated determines which slave device the bus master selects. The eighth bit determines the direction of the transfer (read or write).

Every data byte that is put onto the SDA line must total 8 bits. The number of bytes that can be sent or received per bus transfer operation is unlimited. Data is always sent as most-significant bit (MSB) first and every byte must be immediately followed by an acknowledge (ACK) bit.

18.1.1 Multi-Master IIC-Bus Control Register

The multi-master IIC-bus control register, ICCR, is located at address FCH. It is Read/Write addressable.

ICCR settings control these IIC-bus functions:

- Enables or suppresses CPU acknowledge signal (ACK)Selects IIC-bus clock source (fosc/16 or fosc/512)
- Transmits/receives interrupt enable or disable
- Transmits/receives interrupt pending control
- Provide 4-bit prescaler for the serial transmit clock (SCL)

In the S3F8S39/S3F8S35 interrupt structure, the IIC-bus Tx/Rx interrupt is assigned level IRQ4, vector E6H. To enable this interrupt, you should set ICCR.5 to "1". Program software then polls the IIC-bus Tx/Rx interrupt pending bit (ICCR.4) to detect IIC-bus receive or transmit requests. When the CPU acknowledges the interrupt request from the IIC-bus, the interrupt service routine should clear the interrupt pending condition by writing a "0" to ICCR.4.

The IIC-bus clock source selection (fosc/16 or fosc/512) and the 4-bit prescaler value in the ICCR register determines the SCL frequency (Refer to *Figure 18-1* for more information).

Figure 18-1 illustrates the multi-master IIC-bus control register.

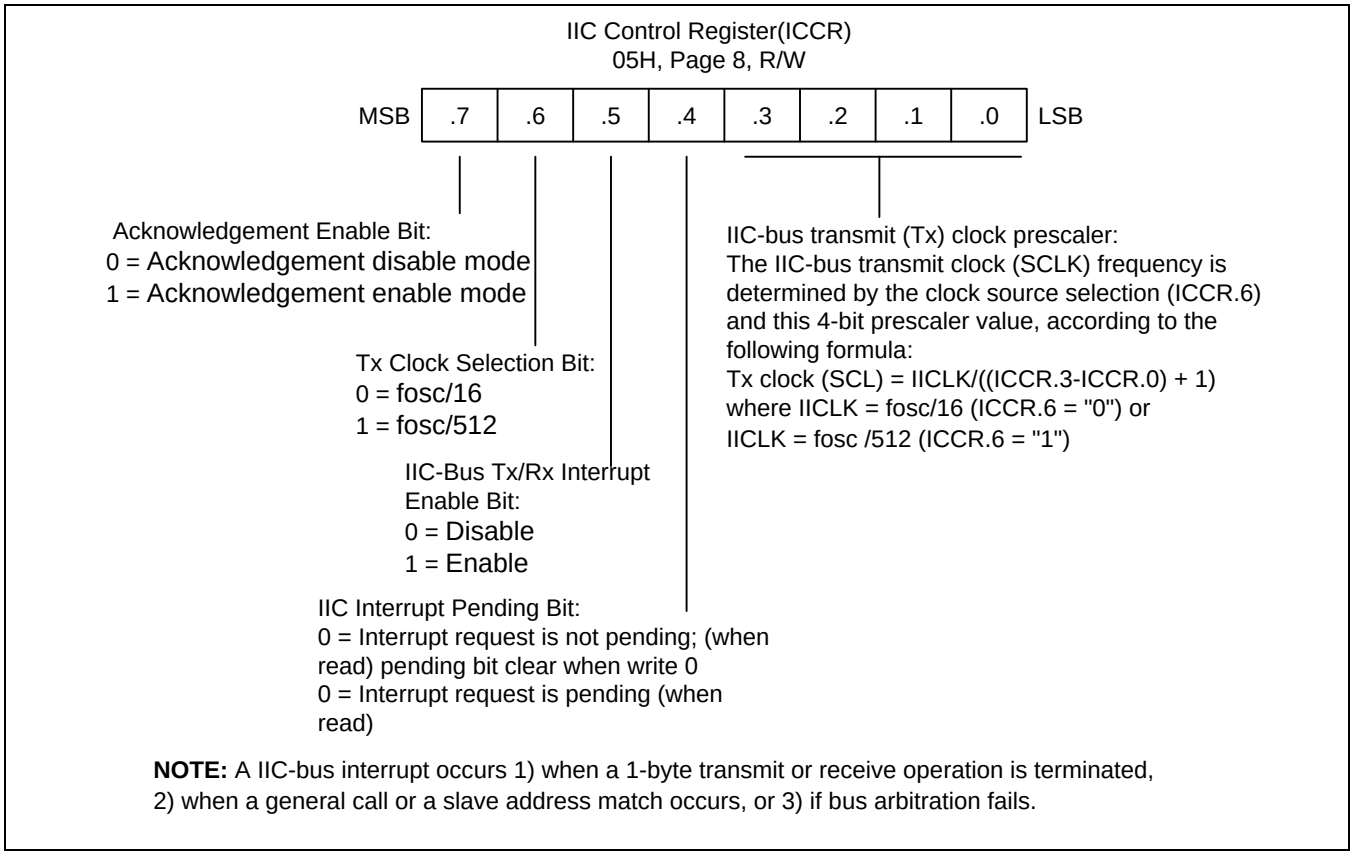


Figure 18-1 Multi-Master IIC-Bus Control Register (ICCR)

Table 18-1 lists the sample timing calculations for the IIC-bus transmit clock.

Table 18-1 Sample Timing Calculations for the IIC-Bus Transmit Clock (SCL)

ICCR.3-ICCR.0 Value	IICLK (ICCR.3-ICCR.0 Settings + 1)	(fosc = 8 MHz) ICCR.6 = 0 (fosc/16) IICLK = 500 kHz	(fosc = 8 MHz) ICCR.6 = 1 (fosc/512) IICLK = 15.625 kHz
0000	IICLK/1	400 kHz (NOTE)	15.625 kHz
0001	IICLK/2	250 kHz	7.1825 kHz
0010	IICLK/3	16.7kHz	5.2038 kHz
0011	IICLK/4	125 kHz	3.9063 kHz
0100	IICLK/5	100 kHz	3.1250 kHz
0101	IICLK/6	83.3 kHz	2.6042 kHz
0110	IICLK/7	71.4 kHz	2.2321 kHz
0111	IICLK/8	62.5 kHz	1.9531 kHz
1000	IICLK/9	55.6 kHz	1.7361 kHz
1001	IICLK/10	50 kHz	1.5625 kHz
1010	IICLK/11	45.5 kHz	1.4205 kHz
1011	IICLK/12	41.7 kHz	1.3021 kHz
1100	IICLK/13	38.5 kHz	1.2019 kHz
1101	IICLK/14	35.7 kHz	1.1160 kHz
1110	IICLK/15	33.3 kHz	1.0417 kHz
1111	IICLK/16	31.25 kHz	0.9766 kHz

NOTE: Maximum IICLK = 400 kHz.

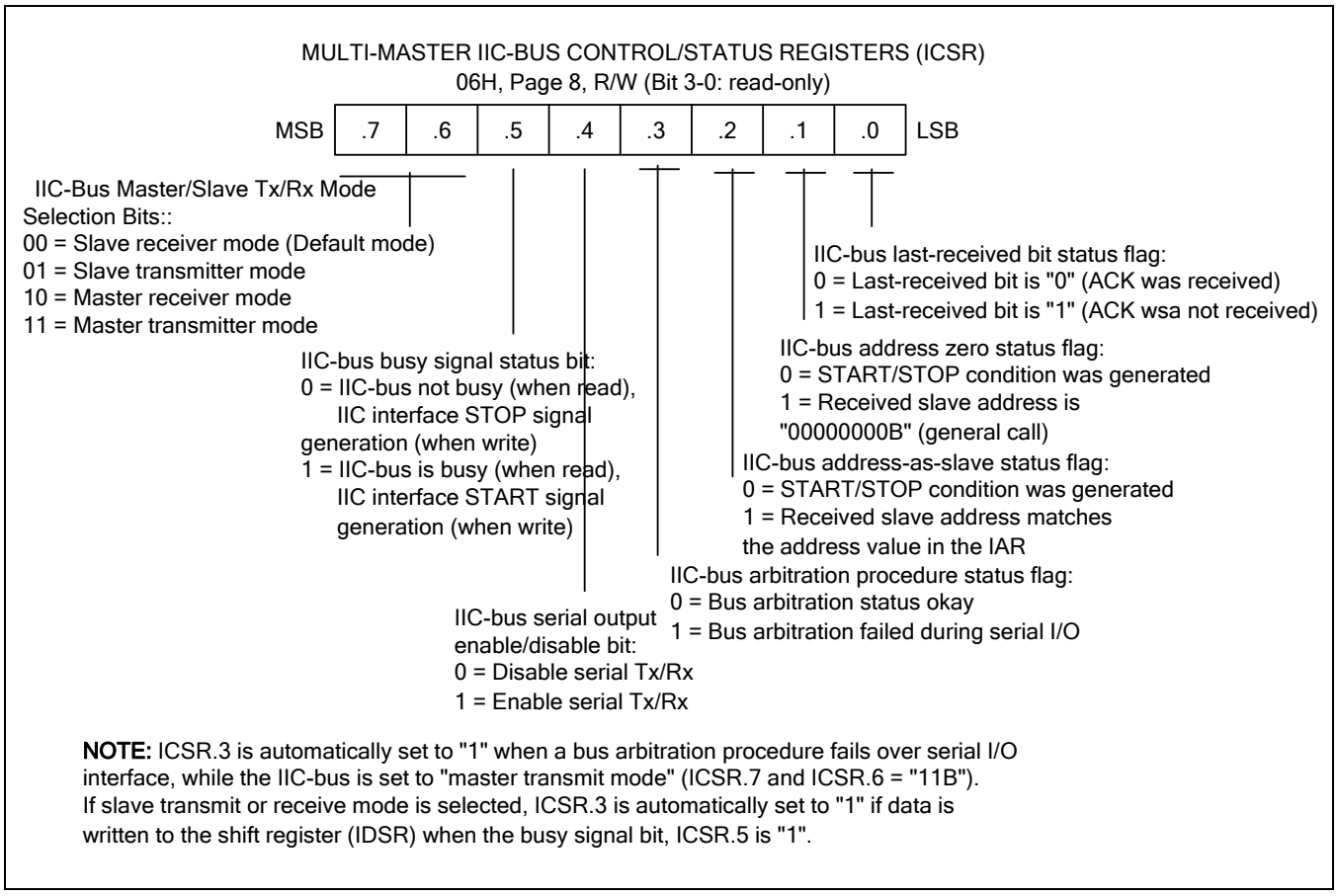
18.1.2 Multi-Master IIC-Bus Control/Status Register

The multi-master IIC-bus control/status register (ICSR) is located at address FDH, BANK1. 4 bits in this register, ICSR.3-ICSR.0, are read-only status flags.

ICSR register settings are used to control or monitor the IIC-bus functions (Refer to *Figure 18-2* for more information). The IIC-bus functions are:

- Selects master/slave transmit or receive mode Provide an IIC-bus busy status flag
- Enables/disables Serial output Provide failed bus arbitration procedure status flag
- Provide Slave address/address register match or general call received status flag
- Provide Slave address 00000000B (general call) received status flag
- Provide last received bit status flag (not ACK = "1", ACK = "0")

Figure 18-2 illustrates the multi-master IIC-bus control/status register.



NOTE: ICSR.3 is automatically set to "1" when a bus arbitration procedure fails over serial I/O interface, while the IIC-bus is set to "master transmit mode" (ICSR.7 and ICSR.6 = "11B"). If slave transmit or receive mode is selected, ICSR.3 is automatically set to "1" if data is written to the shift register (IDSR) when the busy signal bit, ICSR.5 is "1".

Figure 18-2 Multi-Master IIC-Bus Control/Status Register (ICSR)

18.1.3 Multi-Master IIC-Bus Transmit/Receive Data Shift Register

The IIC-bus data shift register, IDSR, is located at address F5H. In a transmit operation, data that is written to the IDSR is transmitted serially and MSB first. (For receive operations, the input data is written into the IDSR register LSB first.)

The ICSR.4 setting enables or disables serial transmit/receive operations. When ICSR.4 = "1", data can be written to the shift register. However, you can read the IIC-bus shift register at any time, irrespective of the current ICSR.4 setting.

Figure 18-3 illustrates the multi-master IIC-bus Tx/Rx data shift register.

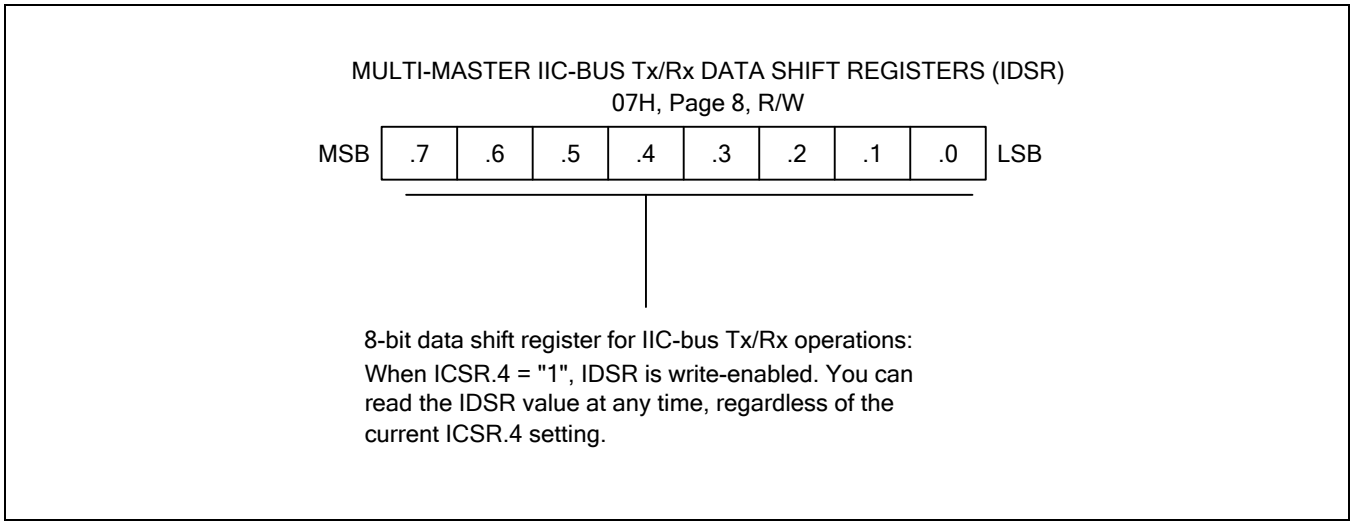


Figure 18-3 Multi-Master IIC-Bus Tx/Rx Data Shift Register (IDSR)

18.1.4 Multi-Master IIC-Bus Address Register

The address register for the IIC-bus interface (IAR) is located at address F4H. It is used to store a latched 7-bit slave address. This address is mapped to IAR.7-IAR.1; bit 0 is not used (Refer to *Figure 18-4* for more information).

The latched slave address is compared to the next received slave address. If a match condition is detected, and if the latched value is 00000000B, a general call status is detected.

Figure 18-4 illustrates the multi-master IIC-bus address register.

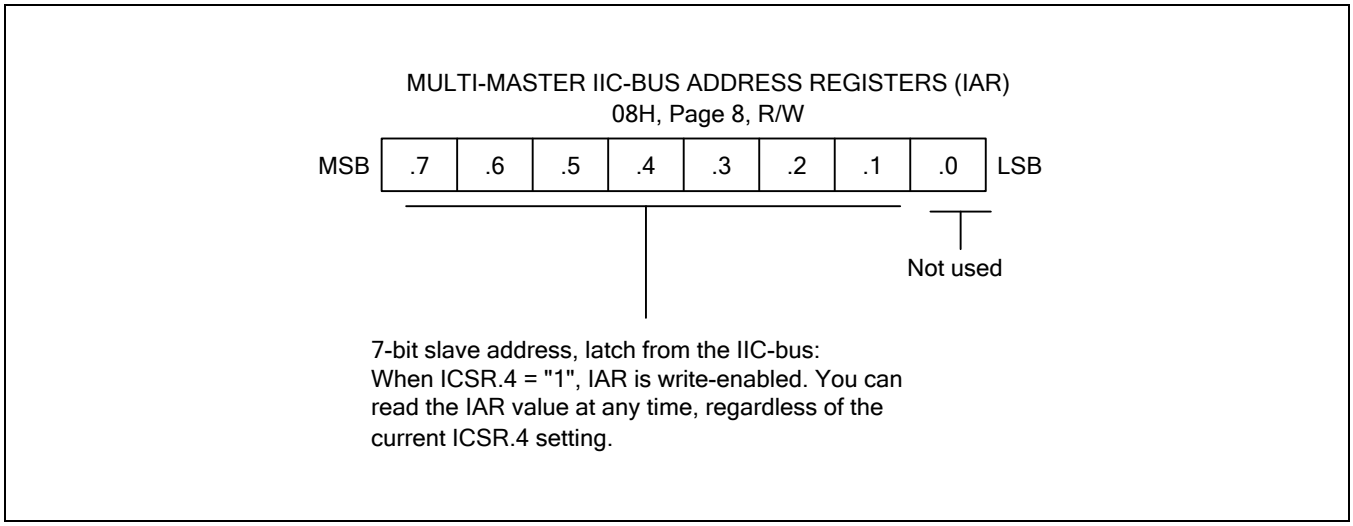


Figure 18-4 Multi-Master IIC-Bus Address Register (IAR)

18.2 Block Diagram

Figure 18-5 illustrates the block diagram of IIC-bus.

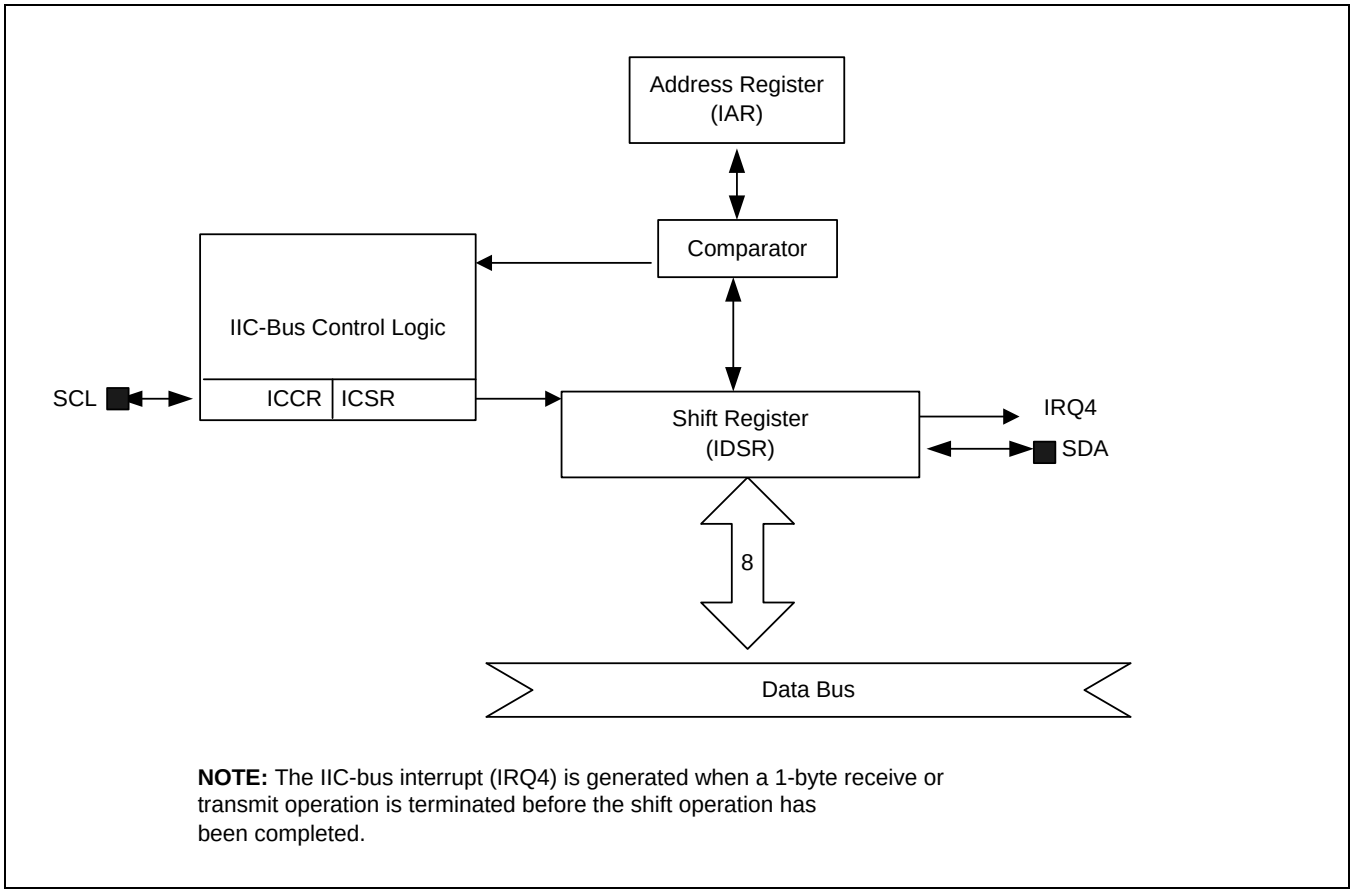


Figure 18-5 IIC-Bus Block Diagram

18.3 The IIC-Bus Interface

The S3F8S39/S3F8S35 IIC-bus interface has four operating modes:

- Master transmitter mode
- Master receive mode
- Slave transmitter mode
- Slave receive mode

Functional relationships between these operating modes are described in these sections.

18.4 Start and Stop Conditions

When the IIC-bus interface is inactive, it is in slave mode. Therefore, the interface is therefore always in slave mode when a start condition is detected on the SDA line. (A start condition is a High-to-Low transition of the SDA line while the clock signal, SCL, is High level.) When the interface enters master mode, it initiates a data transfer and generates the SCL signal.

A start condition initiates a 1 byte serial data transfer over the SDA line and a stop condition ends the transfer. (A stop condition is a Low-to-High transition of the SDA line while SCL is High level.) Start and stop conditions are always generated by the master. The IIC-bus is "busy" when a start condition is generated. After a few clocks a stop condition is generated, the IIC-bus is again "free".

When a master initiates a start condition, it sends its slave address onto the bus. The address byte consists of a 7-bit address and a 1-bit transfer direction indicator (that is, write or read). If bit 8 is "0", a transmit operation (write) is indicated; if bit 8 is "1", a request for data (read) is indicated.

The master ends the indicated transfer operation by transmitting a stop condition. If the master wants to continue sending data over the bus, it can generate another slave address and another start condition. In this way, you can perform read write operations in various formats.

Figure 18-6 illustrates the start and stop conditions.

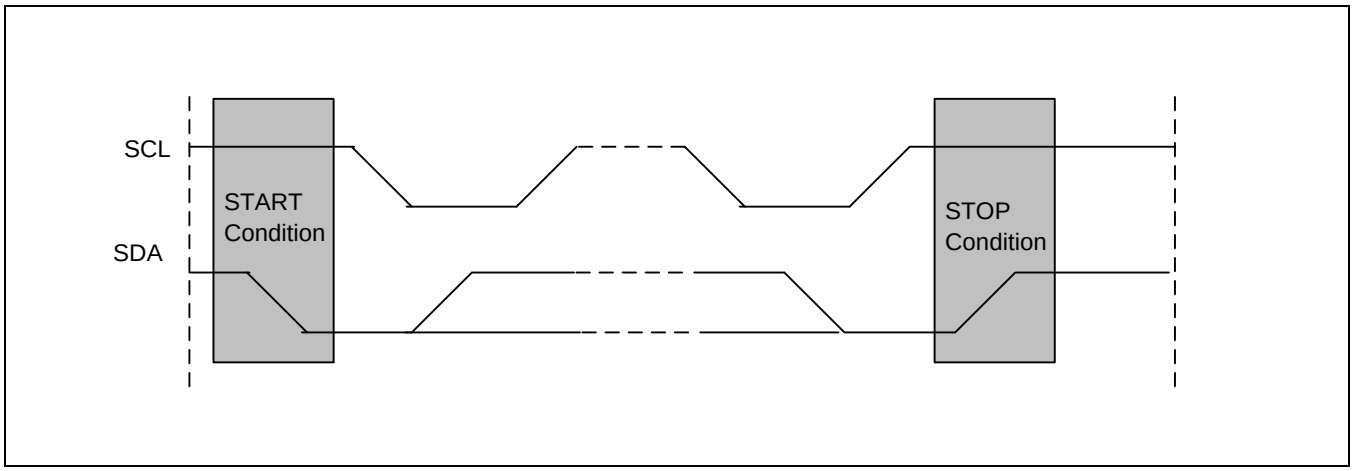


Figure 18-6 Start and Stop Conditions

Figure 18-7 illustrates the input data protocol.

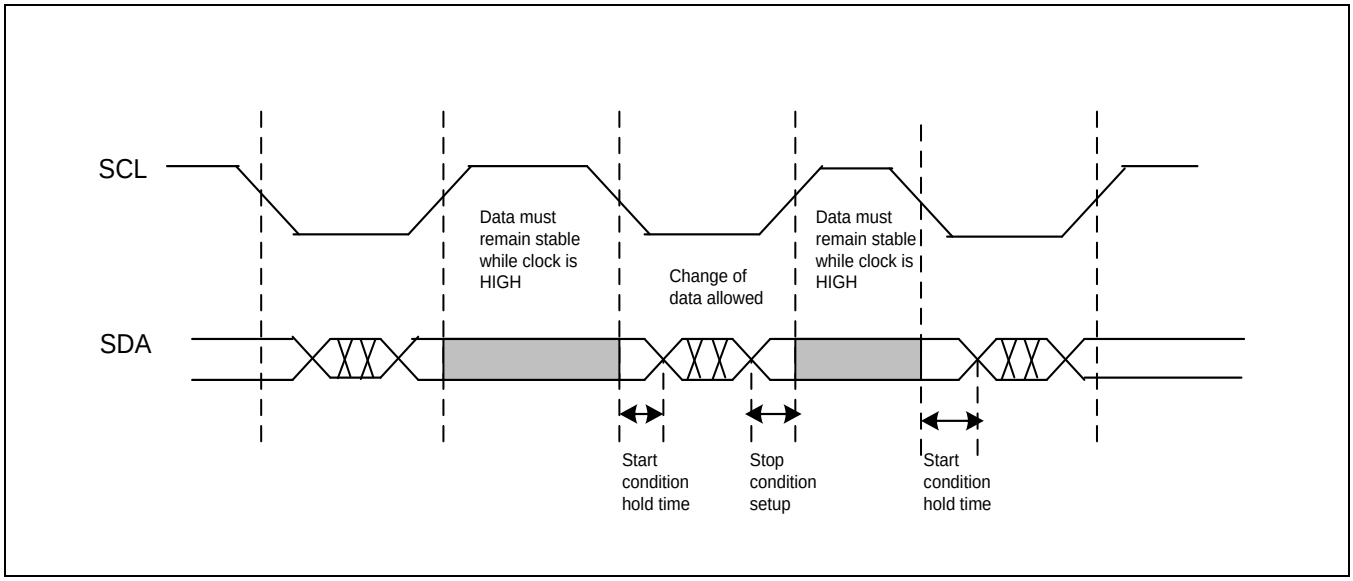
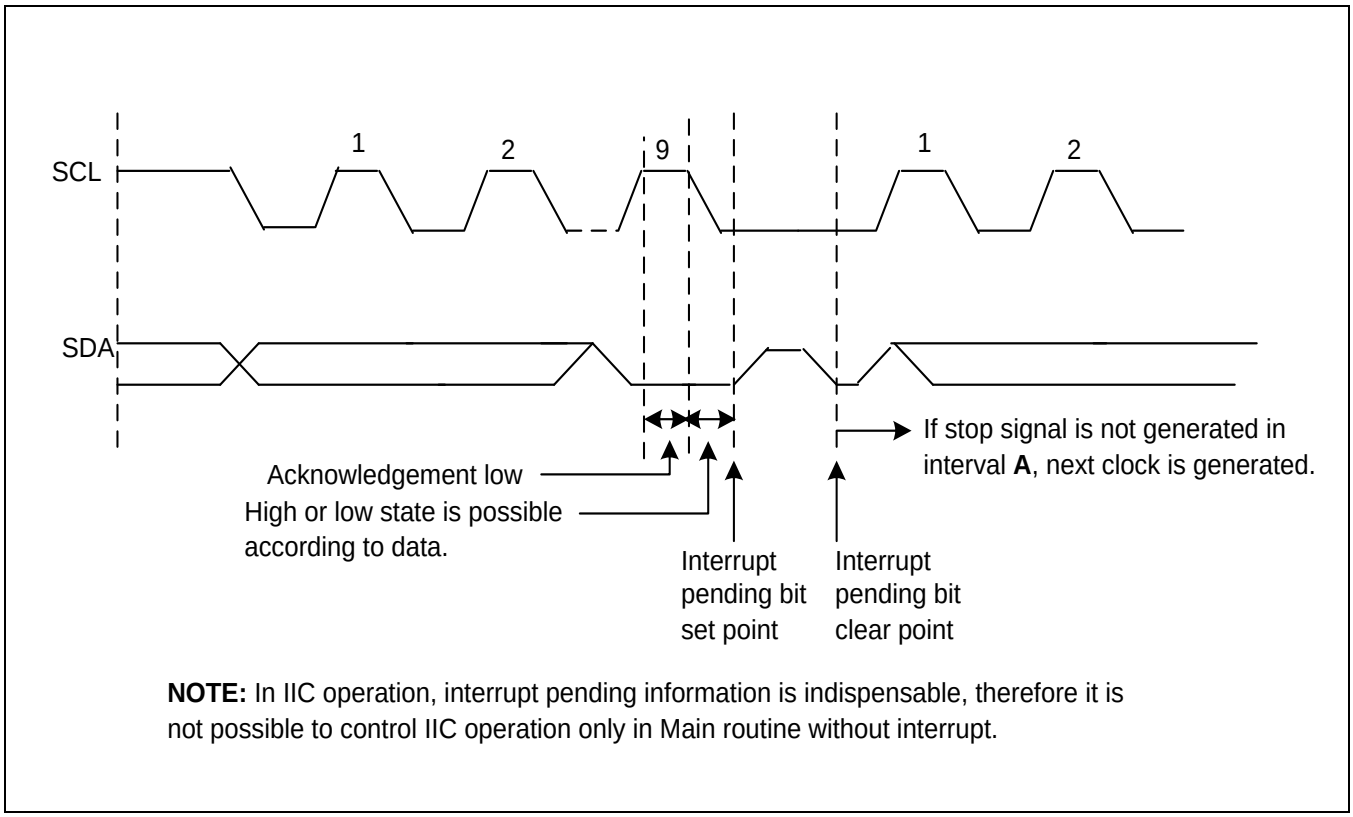


Figure 18-7 Input Data Protocol

Figure 18-8 illustrates the interrupt pending information.



NOTE: In IIC operation, interrupt pending information is indispensable, therefore it is not possible to control IIC operation only in Main routine without interrupt.

Figure 18-8 Interrupt Pending Information

18.5 Data Transfer Formats

Every byte that you put on the SDA line must be 8 bits in length. The number of bytes which you can transmit per transfer is unlimited. The first byte following a start condition is the address byte. The master transmits this address byte when the IIC-bus is operating in master mode. Each byte must be followed by an acknowledge (ACK) bit. Serial data and addresses are always sent MSB first.

Figure 18-9 illustrates the IIC-bus interface data formats.

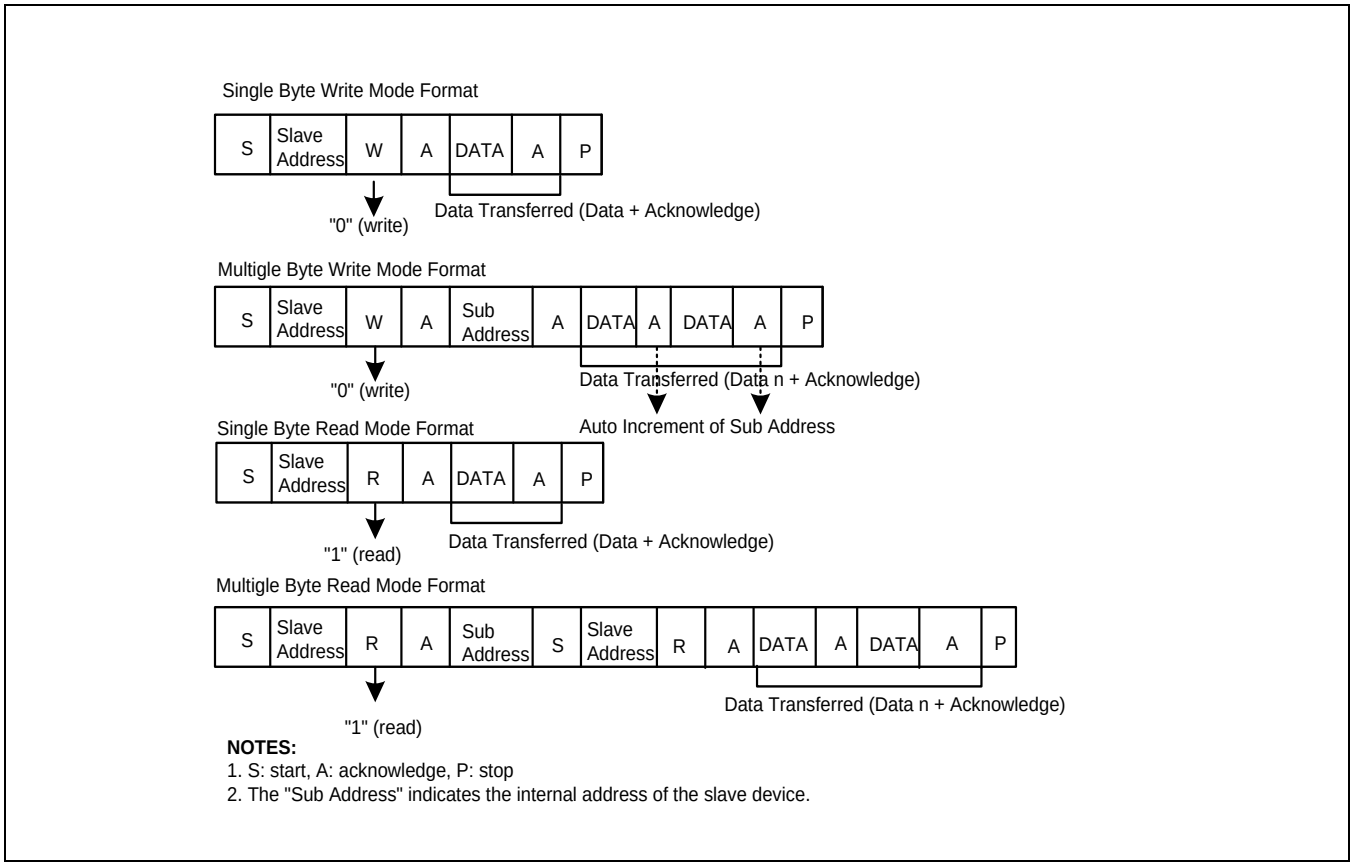


Figure 18-9 IIC-Bus Interface Data Formats

18.6 ACK Signal Transmission

To complete a 1-byte transfer operation, the receiver should send an ACK bit to the transmitter. The ACK pulse occurs at the ninth clock of the SCL line (eight clocks are required to complete the one-byte transfer). The clock pulse required for the transmission of the ACK bit is always generated by the master.

The transmitter releases the SDA line (that is, it sends the SDA line High) when the ACK clock pulse is received. The receiver must drive the SDA line Low during the ACK clock pulse so that SDA is Low during the High period of the ninth SCL pulse.

You can enable and disable the ACK bit transmit function by software (ICCR.7). However, the ACK pulse on the ninth clock of SCL is required to complete a 1 byte data transfer operation.

Figure 18-10 illustrates the acknowledge response from receiver.

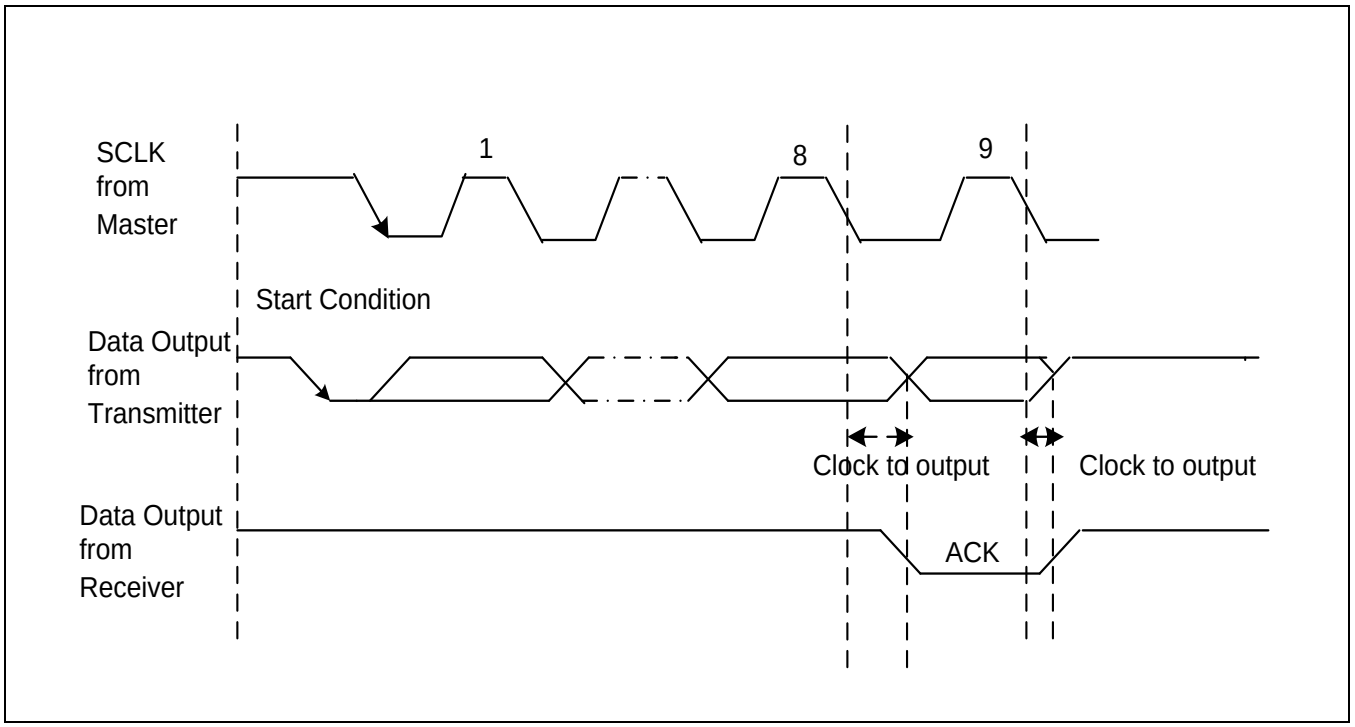


Figure 18-10 Acknowledge Response from Receiver

Figure 18-11 illustrates the write operation sequence.

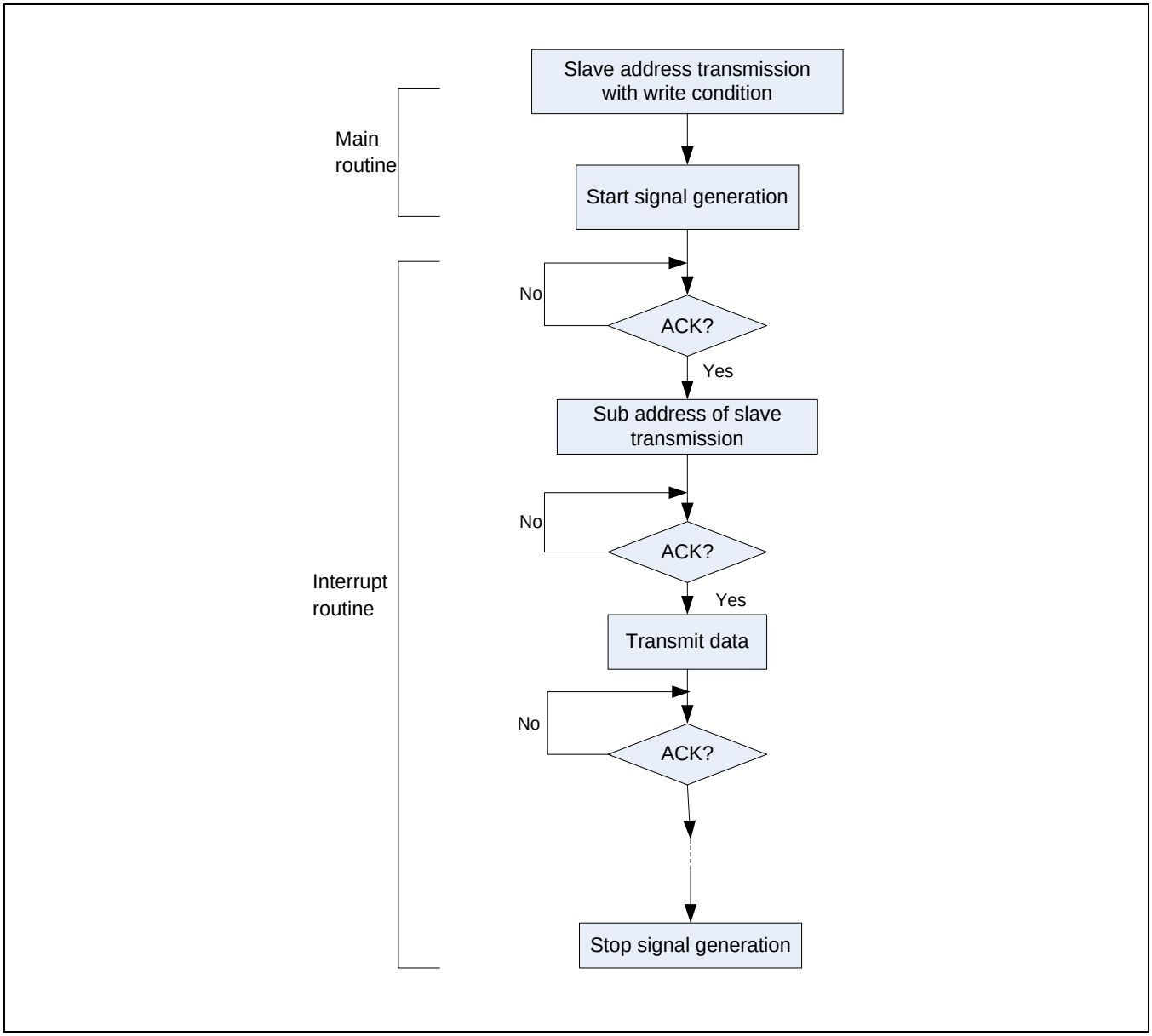


Figure 18-11 Write Operation Sequence

Figure 18-12 illustrates the read operation sequence.

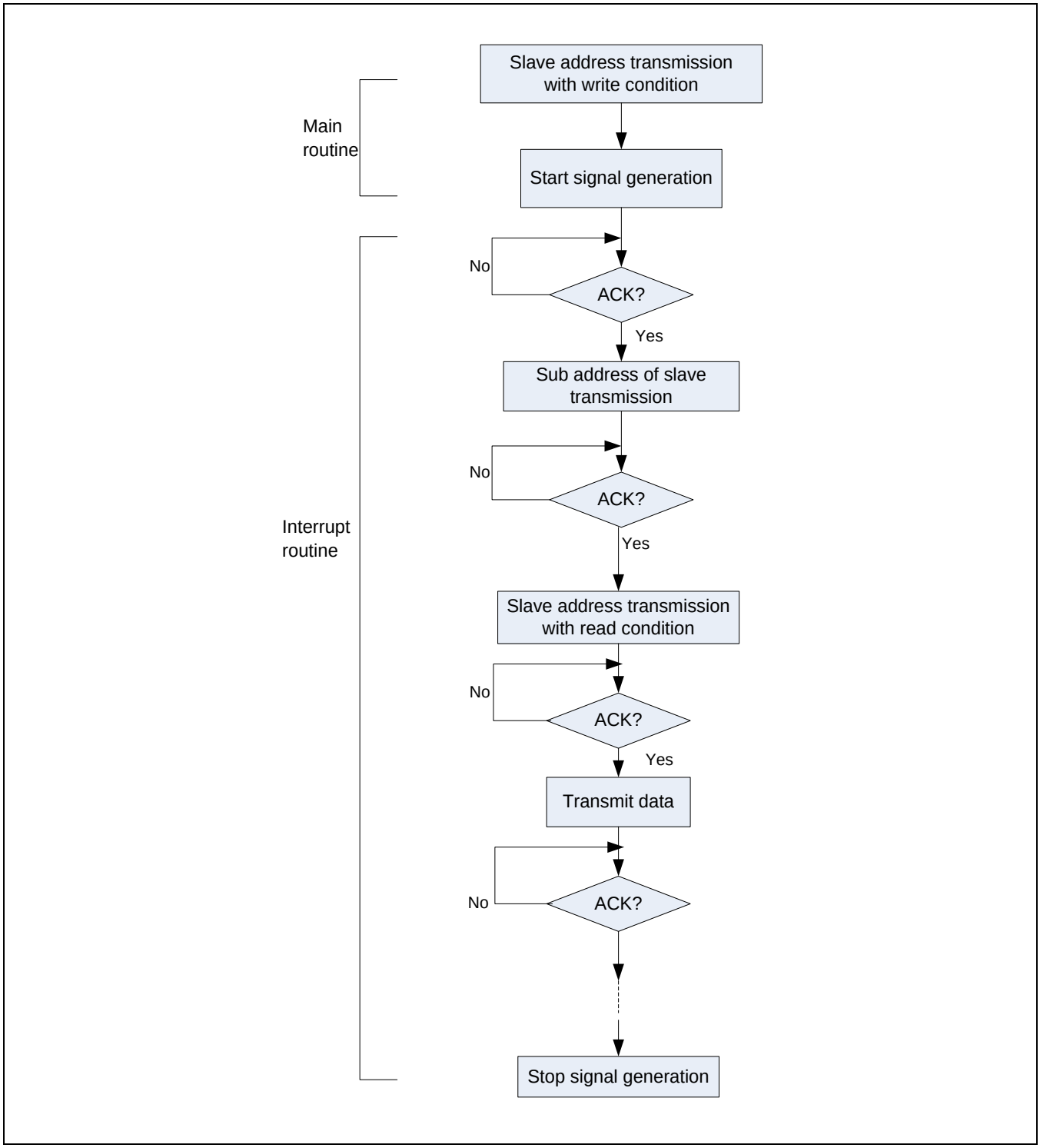


Figure 18-12 Read Operation Sequence

18.7 Read-Write Operations

When operating in transmitter mode, the IIC-bus interface interrupt routine waits for the master (the S3F8S39/S3F8S35) to write a data byte into the IIC-bus data shift register (IDSR). To perform this, it holds the SCL line Low prior to transmission.

In receive mode, the IIC-bus interface waits for the master to read the byte from the IIC-bus data shift register (IDSR). It performs this by holding the SCL line Low after the complete reception of a data byte.

18.8 Bus Arbitration Procedures

Arbitration takes place on the SDA line to prevent contention on the bus between two masters. If a master with a SDA High level detects another master with an SDA active Low level, it will not initiate a data transfer because the current level on the bus does not correspond to its own level. The master that loses the arbitration can generate SCL pulses only until the end of the last-transmitted data byte. The arbitration procedure can continue while data continues to be transferred over the bus.

The first stage of arbitration is the comparison of address bits. If a master loses the arbitration during the addressing stage of a data transfer, it is possible that the master that won the arbitration attempts to address the master that lost. In this case, the losing master should immediately switch to slave receiver mode.

18.9 Abort Conditions

If a slave receiver does not acknowledge the slave address, it should hold the level of the SDA line High. This signals the master to generate a stop condition and to abort the transfer.

If a master receiver involves in the aborted transfer, it should also signal the end of the slave transmit operation. It performs this by not generating an ACK after the last data byte received from the slave. The slave transmitter must then release the SDA to allow a master to generate a stop condition.

18.10 Configuring the IIC-Bus

To control the frequency of the serial clock (SCL), you should program the 4-bit prescaler value in the ICCR register. The IIC-bus interface address is stored in IAR. By default, the IIC-bus interface address is an unknown value.

19 UART 0

19.1 Overview

The Universal Asynchronous Receiver/Transmitter (UART) 0 block has a full-duplex serial port with programmable operating modes.

There is one synchronous mode and three UART Modes. They are:

- 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

You can access both UART 0 receive and transmit buffers through the data register, UDATA0 (Set 1, Bank 0 at address F7H). Writing to the UART data register loads the transmit buffer. Reading the UART 0 data register accesses a physically separate receive buffer.

When accessing a receive data buffer (shift register), reception of the next byte can begin before CPU reads the previously received byte 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".

19.2 Programming Procedure

To program the UART 0 modules:

1. Configure P2.7 and P2.6 to alternative function (RxD0 (P2.7), TxD0 (P2.6)) for UART 0 module by setting the P2CONH register to appropriately value.
2. Load an 8-bit value to the UART0CONH/L control register to properly configure the UART 0 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) completes, CPU sets UART 0 pending bit (UART0CONH.0 or UART0CONL.0) to "1" and generates an UART 0 interrupt request.

19.2.1 UART 0 High Byte Control Register (UART0CONH)

The control register for the UART 0 is called UART0CONH in Set 1, Bank 1 at address F5H.

The control functions of UART0CONH are:

- Selects operating mode and baud rate.
- Supports multiprocessor communication and interrupt control.
- Enables/disables serial receive control.
- Ninth data bit location for transmit and receive operations (modes 2 and 3 only).
- Receives UART 0 interrupt control.

A reset clears the UART0CONH value to "00H". So, if you want to use UART 0 module, you should write appropriate value to UART0CONH.

19.2.2 UART 0 Low Byte Control Register

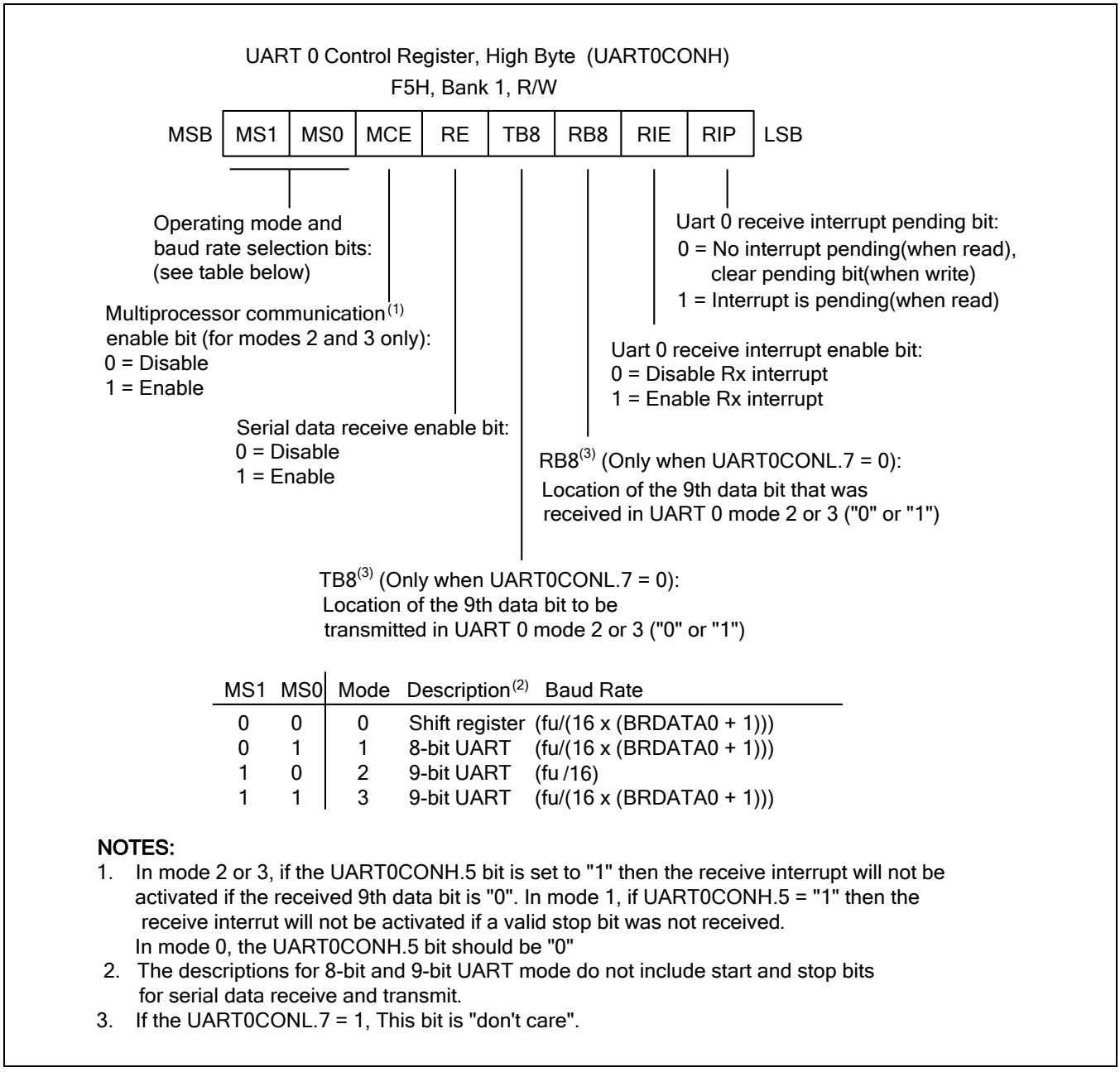
The control register for the UART 0 is called UART0CONL located in Set 1, Bank 1 at address F6H.

The control functions of UART0CONL are:

- Selects UART 0 transmit and receive parity bit
- Selects UART 0 clock
- UART 1 transmit interrupt control

A reset clears the UART0CONL value to "00H". Therefore, if you want to use UART 0 module, you should write appropriate value to UART0CONL.

Figure 19-1 illustrates the UART 0 high byte control register.



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 19-1 UART 0 High Byte Control Register (UART0CONH)

Figure 19-2 illustrates the UART 0 low byte control register.

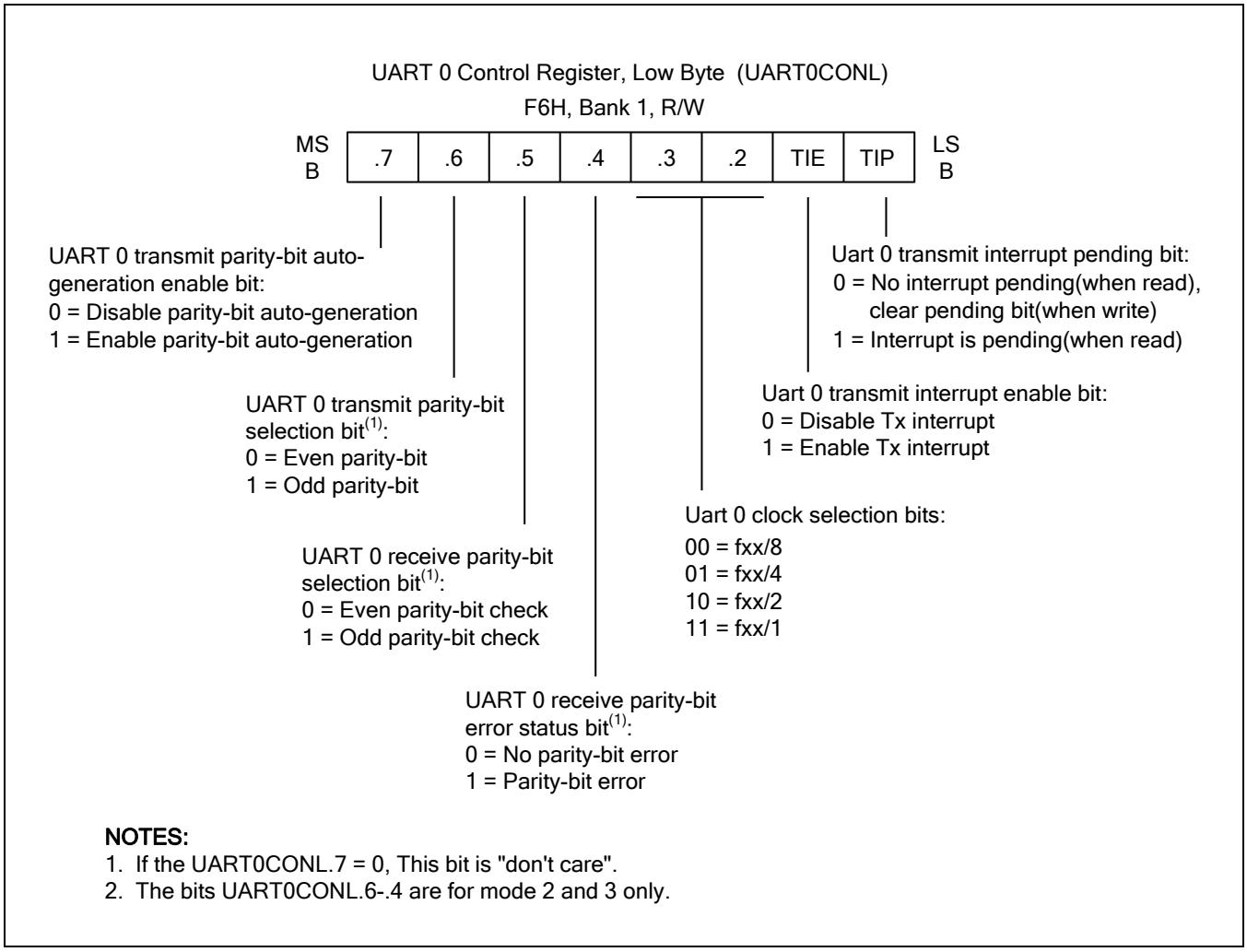


Figure 19-2 UART 0 Low Byte Control Register (UART0CONL)

19.2.3 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 shift time of stop bit. In mode 2, or 3, the UART0CONH.0 bit is set to "1" at the halfway point of the shift time of RB8 bit. When the CPU acknowledges 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 acknowledges the transmit interrupt pending condition, the UART0CONL.0 bit must then be cleared by software in the interrupt service routine.

19.2.4 UART 0 Data Register

Figure 19-3 illustrates the UART 0 data register.

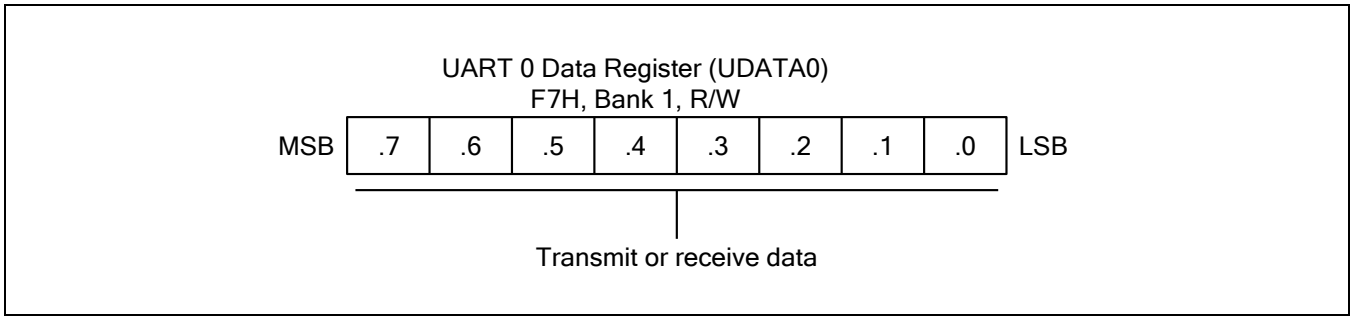


Figure 19-3 UART 0 Data Register (UDATA0)

19.2.5 UART 0 Baud Rate Data Register

The value stored in the UART 0 baud rate register (BRDATA0), enables you to determine the UART 0 clock rate (baud rate).

Figure 19-4 illustrates the UART 0 baud rate data register.

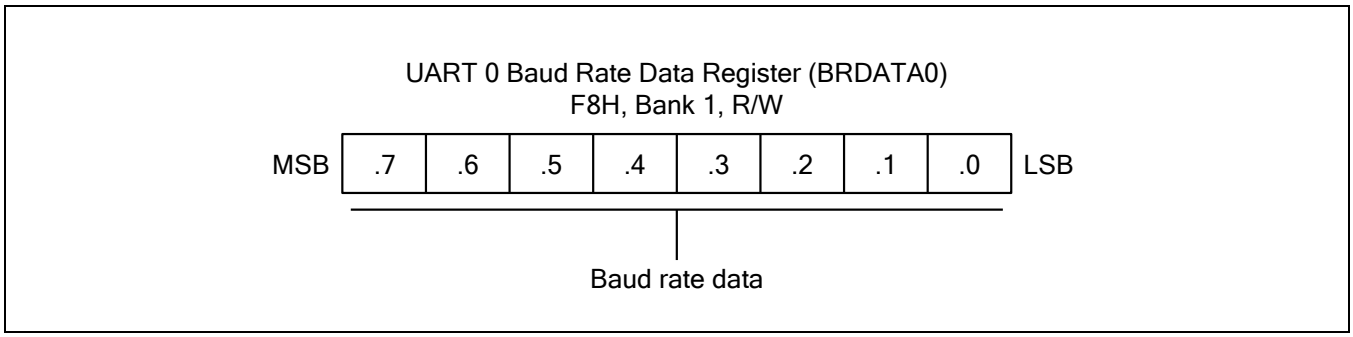


Figure 19-4 UART 0 Baud Rate Data Register (BRDATA0)

19.3 Baud Rate Calculations

This section includes:

- Mode 0 baud rate calculation
- Mode 2 baud rate calculation
- Mode 1 and 3 baud rate calculation

19.3.1 Mode 0 Baud Rate Calculation

In mode 0, the baud rate is determined by the UART 0 baud rate data register, BRDATA0 in Bank 1 at address F8H: Mode 0 baud rate = $f_U/(16 \times (BRDATA0 + 1))$.

19.3.2 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 = $f_U/16$

19.3.3 Modes 1 and 3 Baud Rate Calculation

In modes 1 and 3, the baud rate is determined by the UART 0 baud rate data register, BRDATA0 in Bank 1 at address F8H: Mode 1 and 3 baud rate = $f_U/(16 \times (BRDATA0 + 1))$

Table 19-1 lists the commonly used baud rates generated by BRDATA0.

Table 19-1 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

19.4 Block Diagram

Figure 19-5 illustrates the functional block diagram of UART 0.

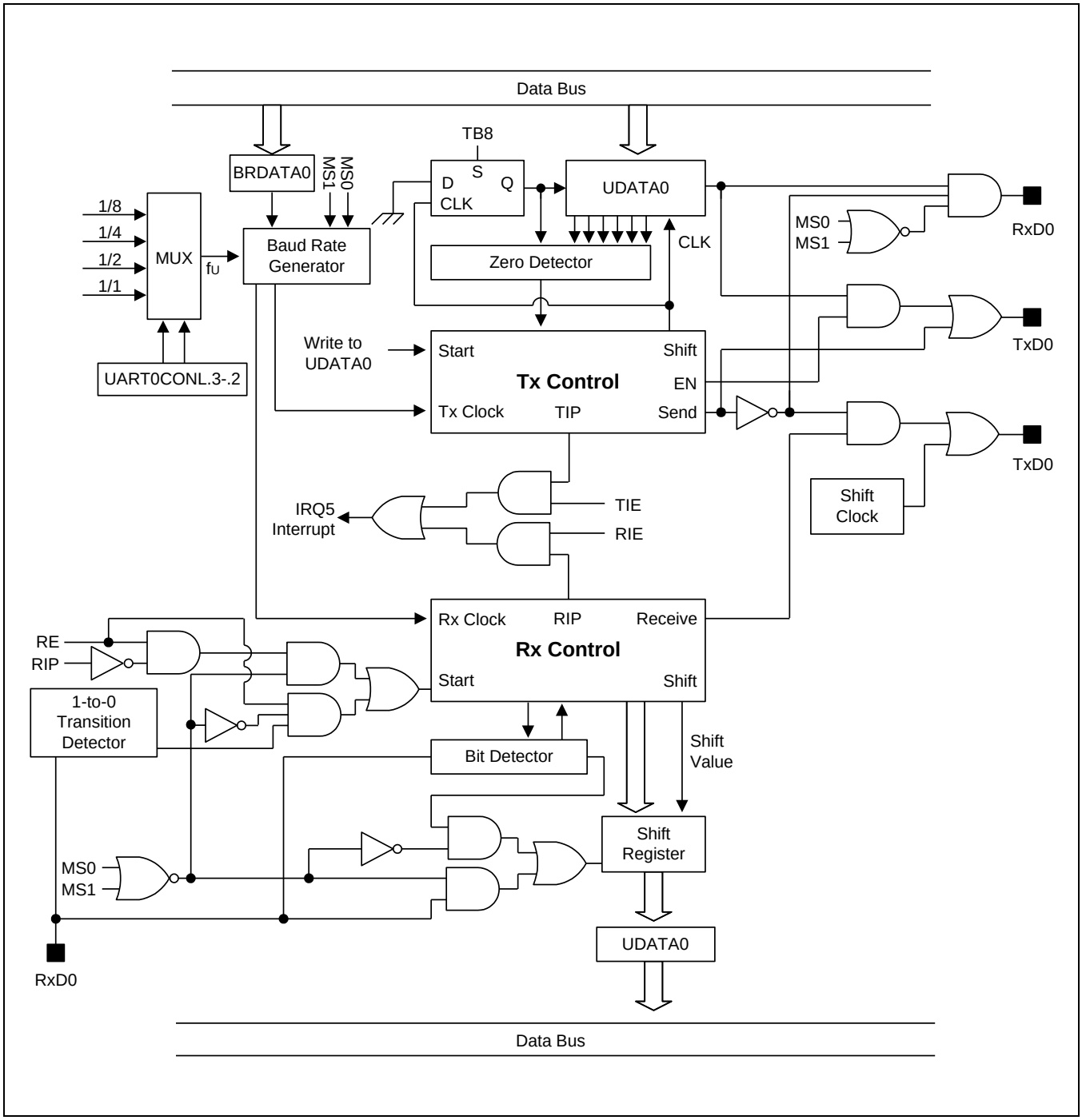


Figure 19-5 UART 0 Functional Block Diagram

19.5 UART 0 Mode 0 Function Description

In mode 0, UART 0 is input and output through the RxD0 (P2.7) pin and TxD0 (P2.6) 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.

19.5.1 Mode 0 Transmit Procedure

The steps of Mode 0 transmit procedure are:

1. Select the UART 0 clock, UART0CONL.3 and .2.
2. Set the UART 0 transmit parity bit autogeneration enable or disable bit (UART0CONL.7).
3. Select mode 0 by setting UART0CONH.7 and .6 to "00B".
4. Write transmission data to the shift register UDATA0 (F0H, set 1, bank 0) to start the transmission operation.

19.5.2 Mode 0 Receive Procedure

The steps of Mode 0 receive procedure are:

1. Select the UART 0 clock, UART0CONL.3 and .2.
2. Set the UART 0 transmit parity bit autogeneration enable or disable 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 0 receive enable (RE) bit (UART0CONH.4) to "1".
6. The shift clock will now be output to the TxD0 (P2.6) pin and will read the data at the RxD0 (P2.7) pin. A UART 0 receive interrupt occurs when UART0CONH.1 is set to "1".

Figure 19-6 illustrates the timing diagram for serial port Mode 0 operation.

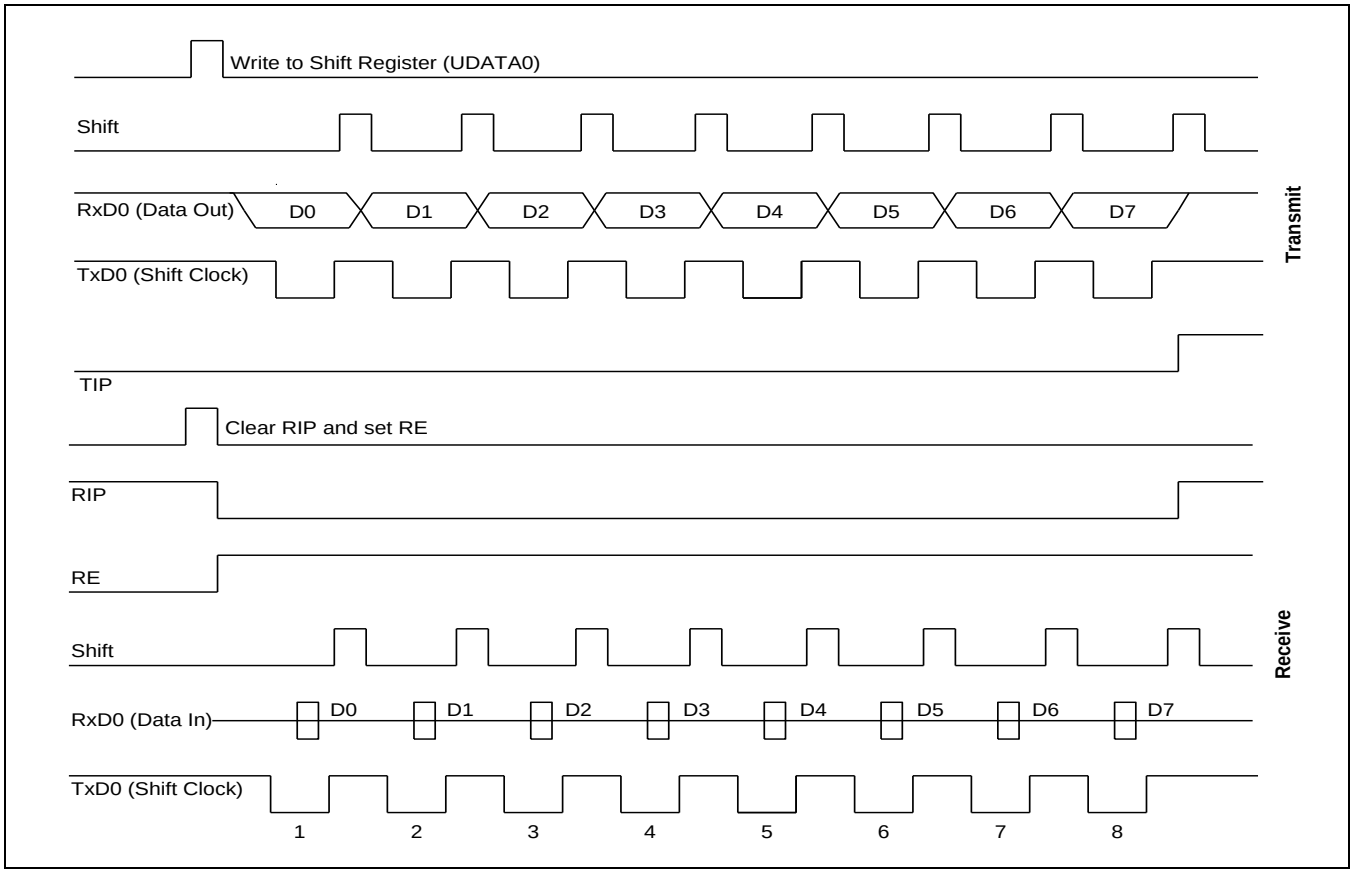


Figure 19-6 Timing Diagram for Serial Port Mode 0 Operation

19.6 Serial Port Mode 1 Function Description

In mode 1, 10 bits are transmitted (through the TxD0 (P2.6) pin) or received (through the RxD0 (P2.7) pin).

The three components of each data frame are:

- Start bit ("0")
- 8 data bits (LSB first)
- Stop bit ("1")

The baud rate for mode 1 is variable.

19.6.1 Mode 1 Transmit Procedure

The steps of Mode 1 transmit procedure are:

1. Select the UART 0 clock, UART0CONL.3 and .2.
2. Set the UART 0 transmit parity bit autogeneration enable or disable bit (UART0CONL.7).
3. Select the baud rate that BRDATA0 generates.
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 (F0H, set 1, bank 0). The start and stop bits are generated automatically by hardware.

19.6.2 Mode 1 Receive Procedure

The steps of Mode 1 receive procedure are:

1. Select the UART 0 clock, UART0CONL.3 and .2.
2. Set the UART 0 transmit parity bit autogeneration enable or disable bit (UART0CONL.7).
3. Select the baud rate to be generated by BRDATA0.
4. Select mode 1 and set the Receive Enable (RE) bit in the UART0CONH register to "1".
5. The start bit low ("0") condition at the RxD0 (P2.7) pin will cause the UART 0 module to start the serial data receive operation.

Figure 19-7 illustrates the timing diagram for serial port Mode 1 operation.

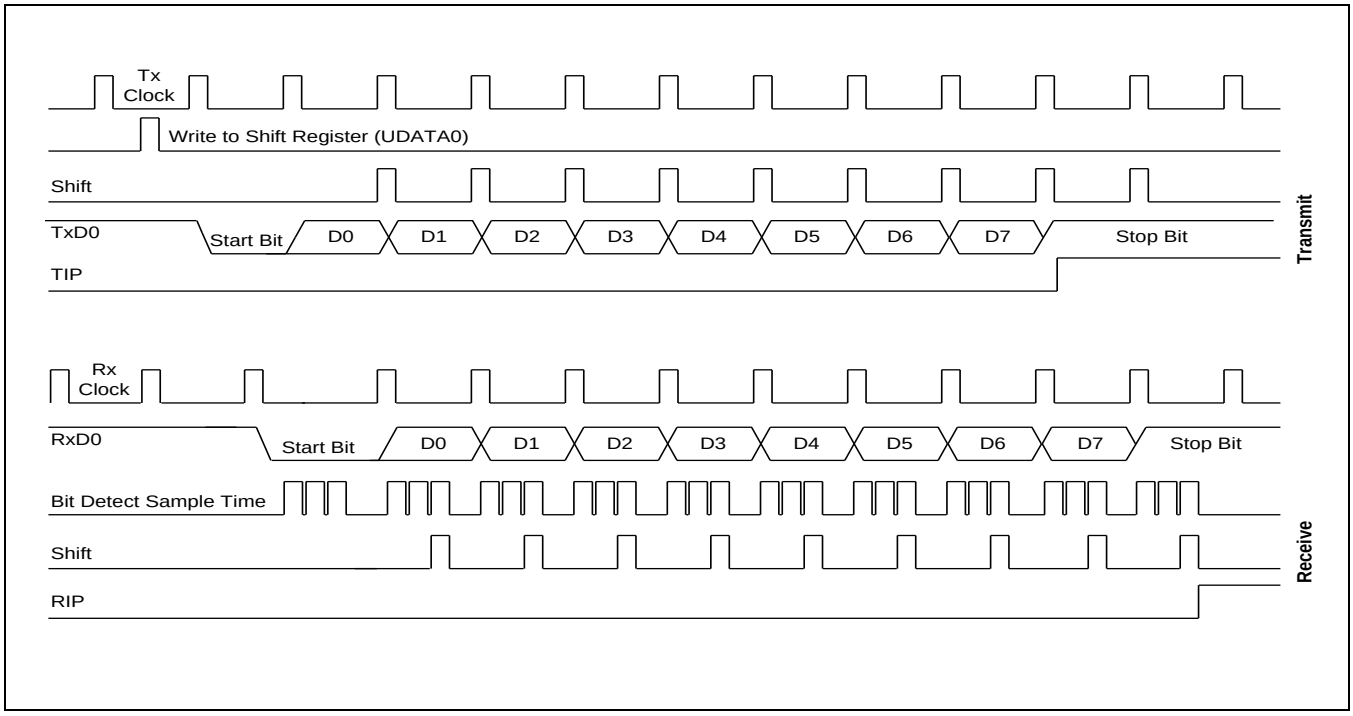


Figure 19-7 Timing Diagram for Serial Port Mode 1 Operation

19.7 Serial Port Mode 2 Function Description

In mode 2, 11 bits are transmitted (through the TxD0 (P2.6) pin) or received (through the RxD0 (P2.7) pin).

The four components of each data frame are:

- 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 ninth 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.

19.7.1 Mode 2 Transmit Procedure

The steps for Mode 2 transmit procedure are:

1. Select the UART 0 clock, UART0CONL.3 and .2.
2. Set the UART 0 transmit parity bit autogeneration 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 (F0H, set 1, bank 0), to start the transmit operation.

19.7.2 Mode 2 Receive Procedure

The steps for Mode 2 receive procedure are:

1. Select the UART 0 clock, UART0CONL.3 and .2.
2. Set the UART 0 transmit parity bit autogeneration enable or disable bit (UART0CONL.7).
3. Select mode 2 and the receive enable bit (RE) in the UART0CONH register to "1".
4. The receive operation starts when the signal at the RxD0 (P2.7) pin enters to low level.

Figure 19-8 illustrates the timing diagram for serial port Mode 2 operation.

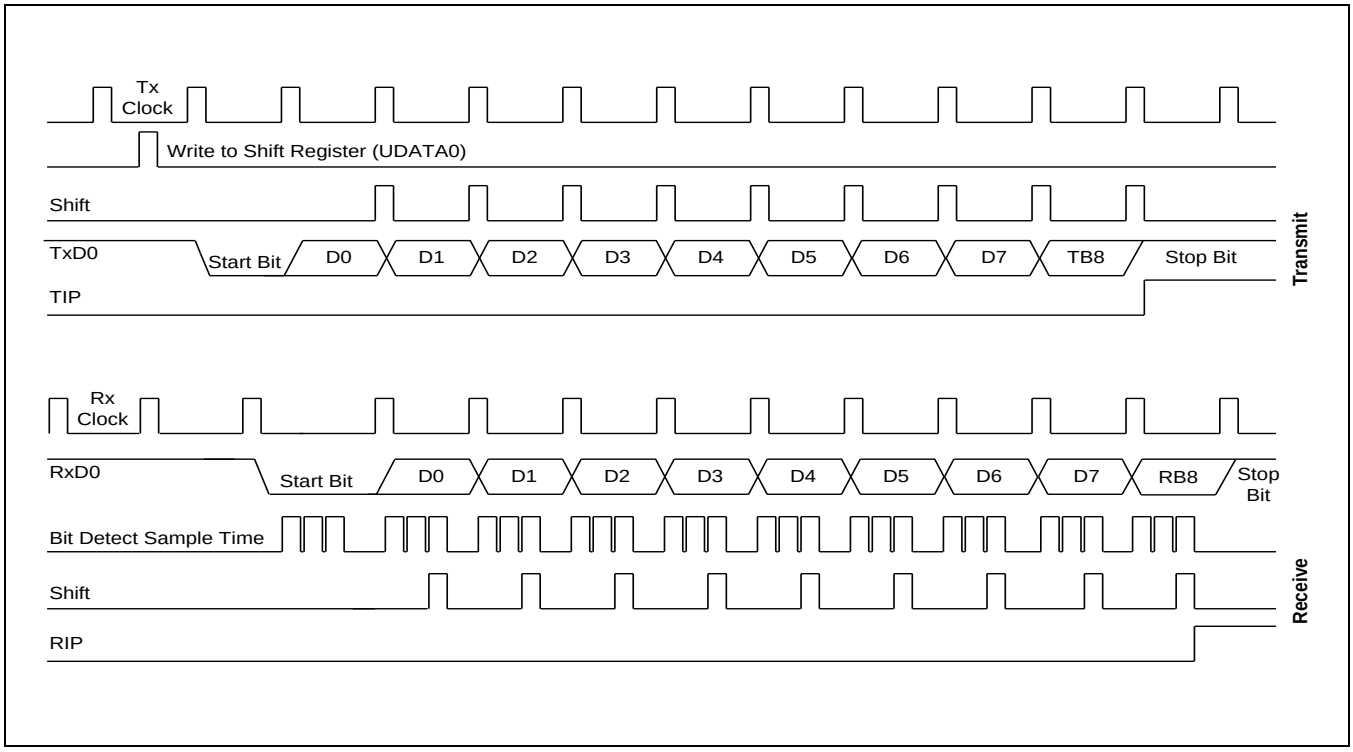


Figure 19-8 Timing Diagram for Serial Port Mode 2 Operation

19.8 Serial Port Mode 3 Function Description

In mode 3, 11 bits are transmitted (through the TxD0 (P2.6) pin) or received (through the RxD0 (P2.7) pin). Mode 3 is identical to mode 2 except for baud rate, which is variable.

The four components of each data frame are:

- Start bit ("0")
- 8 data bits (LSB first)
- Programmable 9th data bit
- Stop bit ("1")

19.8.1 Mode 3 Transmit Procedure

To transmit Mode 3:

1. Select the UART 0 clock, UART0CONL.3 and .2.
2. Set the UART 0 transmit parity bit autogeneration 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 (F0H, set 1, bank 0), to start the transmit operation.

19.8.2 Mode 3 Receive Procedure

The steps for Mode 3 receive procedure are:

1. Select the UART 0 clock, UART0CONL.3 and .2.
2. Set the UART 0 transmit parity bit autogeneration enable or disable bit (UART0CONL.7).
3. Select mode 3 and set the Receive Enable (RE) bit in the UART0CONH register to "1".
4. The receive operation will be started when the signal at the RxD0 (P2.7) pin enters to low level.

Figure 19-9 illustrates the timing diagram for serial port Mode 3 operation.

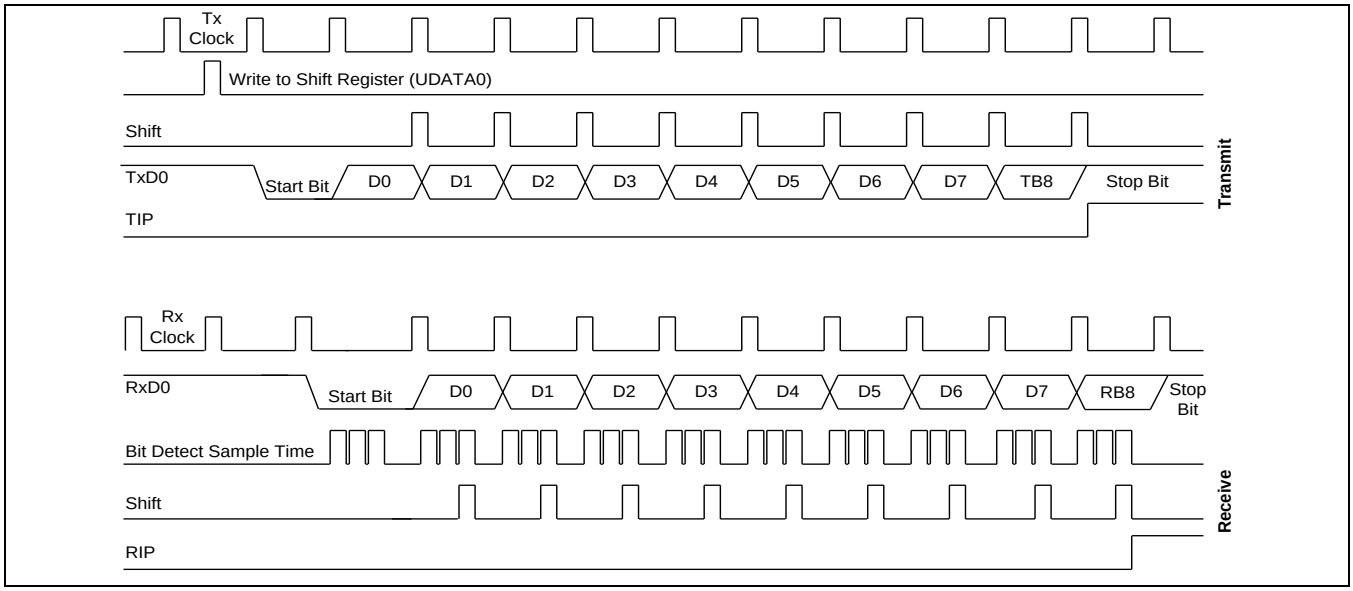


Figure 19-9 Timing Diagram for Serial Port Mode 3 Operation

19.9 Serial Communication for Multiprocessor Configurations

The S3F8-series multiprocessor communication feature enables a "master" S3F8S39/S3F8S35 to send a multiple-frame serial message to a "slave" device in a multi- S3F8S39/S3F8S35 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 should set the MCE bit in the UART0CONH register. When the MCE bit is "1", the 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.

19.9.1 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, you can use it in mode 1 to verify 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.

19.9.2 Setup Procedure for Multiprocessor Communications

The steps to configure multiprocessor communications are:

1. Set all S3F8S39/S3F8S35 devices (masters and slaves) to UART 0 mode 2 or 3.
2. Write the MCE bit of all the slave devices to "1".
3. The transmission protocol of master device 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 illustrates the connection example of multiprocessor serial data communications.

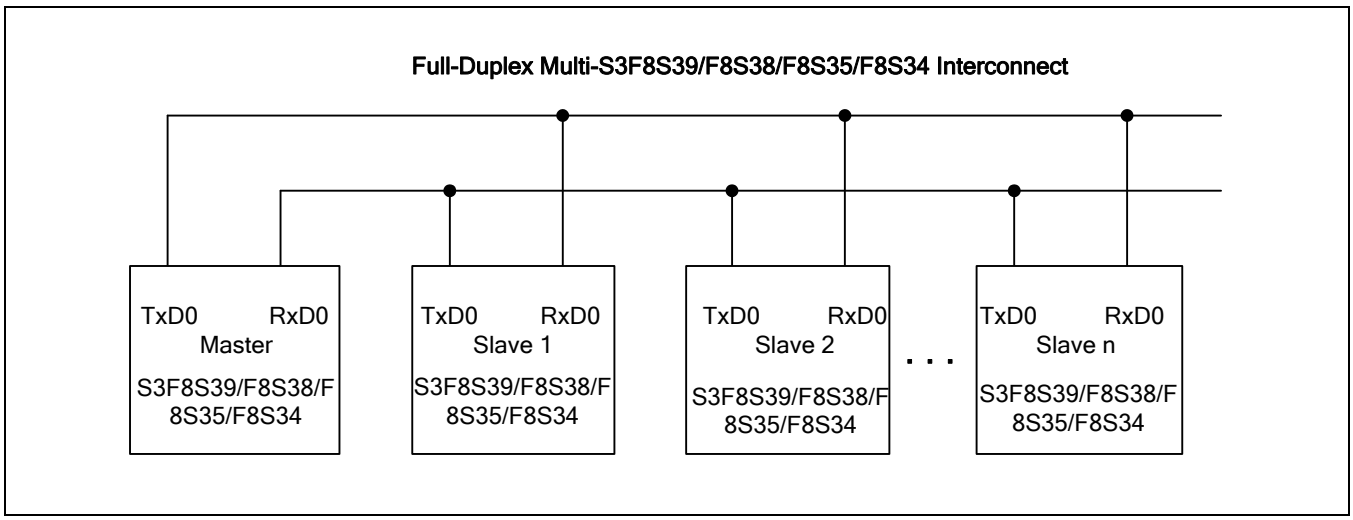


Figure 19-10 Connection Example for Multiprocessor Serial Data Communications

20 UART 1

20.1 Overview

The Universal Asynchronous Receiver/Transmitter (UART)1 block has a full-duplex serial port with programmable operating modes:

There is one synchronous mode and three Universal Asynchronous Receiver/Transmitter (UART) 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

You can access both UART 1 receive and transmit buffers through the data register, UDATA1 (Set 1, Bank 1 at address FBH). Writing to the UART data register loads the transmit buffer. Reading the UART 1 data register accesses a physically separate receive buffer.

When accessing a receive data buffer (shift register), reception of the next byte can begin before CPU reads the previously received byte 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".

20.2 Programming Procedure

To program the UART 1 modules:

1. Configure P2.5 and P2.4 to alternative function (RxD1 (P2.5), TxD1 (P2.4)) for UART 1 module by setting the P2CONH/L register to appropriately value.
2. Load an 8-bit value to the UART1CONH/L control register to properly configure the UART 1 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) completes, CPU sets UART 1 pending bit (UART1CONH.0 or UART1CONL.0) to "1" and generates an UART 1 interrupt request.

20.2.1 UART 1 High Byte Control Register (UART1CONH)

The control register for the UART 1 is called UART1CONH in Set 1, Bank 1 at address F9H.

The control functions of UART1CONH are:

- Selects operating mode and baud rate.
- Supports multiprocessor communication and interrupt control
- Enables/disables serial receive control.
- Ninth 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 should write appropriate value to UART1CONH.

20.2.2 UART 1 Low Byte Control Register

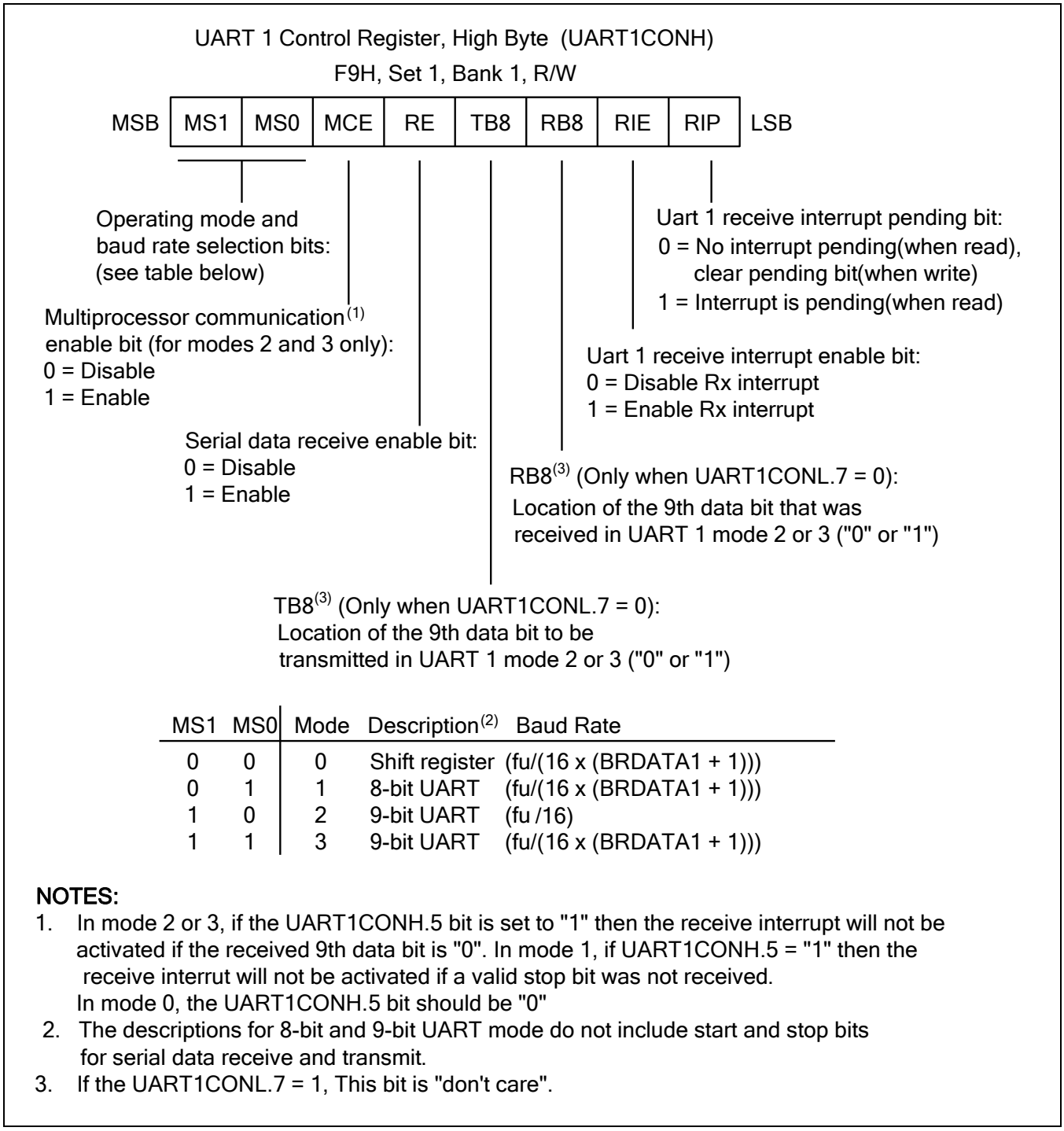
The control register for the UART 1 is called UART1CONL located in Set 1, Bank 1 at address FAH.

The control functions of UART1CONL are:

- Selects UART 1 transmit and receive parity bit
- Selects UART 1 clock
- UART 1 transmit interrupt control

A reset clears the UART1CONL value to "00H". Therefore, if you want to use UART 1 module, you should write appropriate value to UART1CONL.

Figure 20-1 illustrates the UART 1 high byte control register.



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 20-1 UART 1 High Byte Control Register (UART1CONH)

Figure 20-2 illustrates the UART 1 low byte control register.

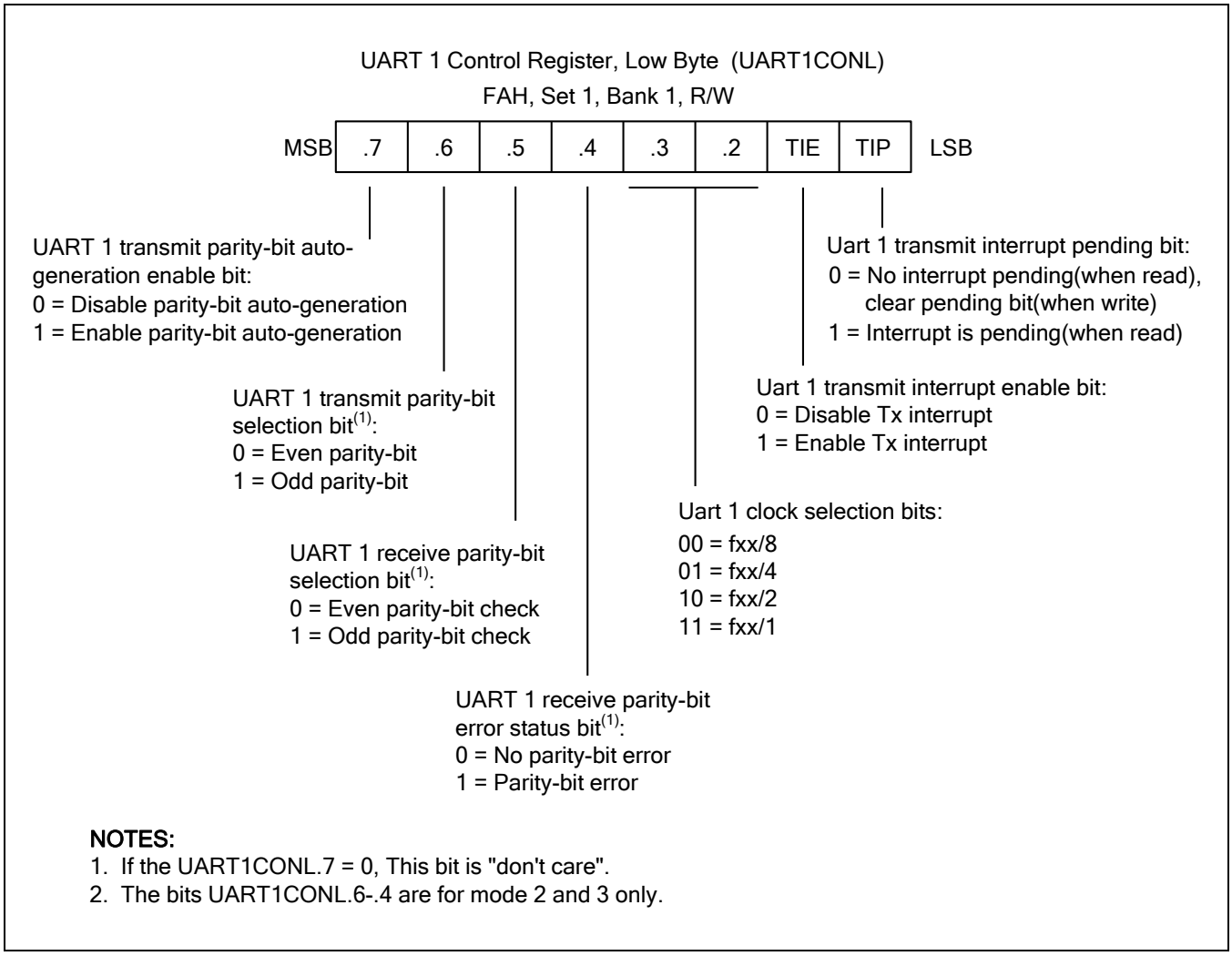


Figure 20-2 UART 1 Low Byte Control Register (UART1CONL)

20.2.3 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 shift time of stop bit. In mode 2, or 3, the UART1CONH.0 bit is set to "1" at the halfway point of the shift time of RB8 bit. When the CPU acknowledges 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 acknowledges the transmit interrupt pending condition, the UART1CONL.0 bit must then be cleared by software in the interrupt service routine.

20.2.4 UART 1 Data Register

Figure 20-3 illustrates the UART 1 data register.

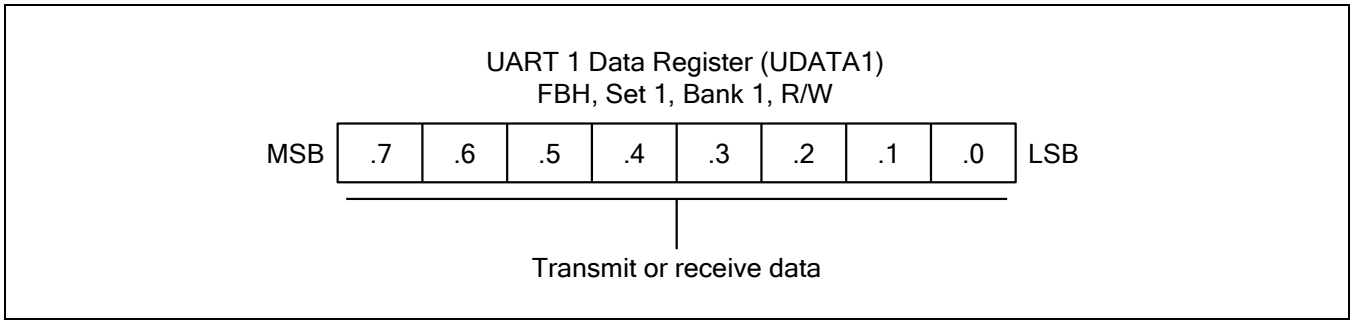


Figure 20-3 UART 1 Data Register (UDATA1)

20.2.5 UART 1 Baud Rate Data Register

The value stored in the UART 1 baud rate register, BRDATA1, enables you determine the UART 1 clock rate (baud rate).

Figure 20-4 illustrates the UART 1 baud rate data register.

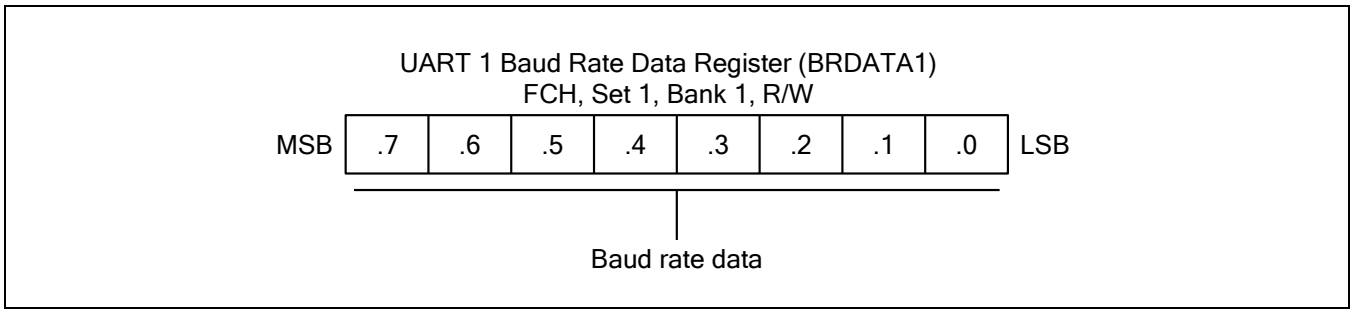


Figure 20-4 UART 1 Baud Rate Data Register (BRDATA1)

20.3 Baud Rate Calculations

This section includes:

- Mode 0 baud rate calculation
- Mode 2 baud rate calculation
- Mode 1 and 3 baud rate calculation

20.3.1 Mode 0 Baud Rate Calculation

In mode 0, the baud rate is determined by the UART 1 baud rate data register, BRDATA1 in set 1, bank 1 at address FCH: Mode 0 baud rate = $f_U/(16 \times (BRDATA1 + 1))$.

20.3.2 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 = $f_U/16$

20.3.3 Modes 1 and 3 Baud Rate Calculation

In modes 1 and 3, the baud rate is determined by the UART 1 baud rate data register, BRDATA1 in set 1, bank 1 at address FCH: Mode 1 and 3 baud rate = $f_U/(16 \times (BRDATA1 + 1))$

Table 20-1 lists the commonly used baud rates generated by BRDATA1.

Table 20-1 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

20.4 Block Diagram

Figure 20-5 illustrates the functional block diagram of UART 1.

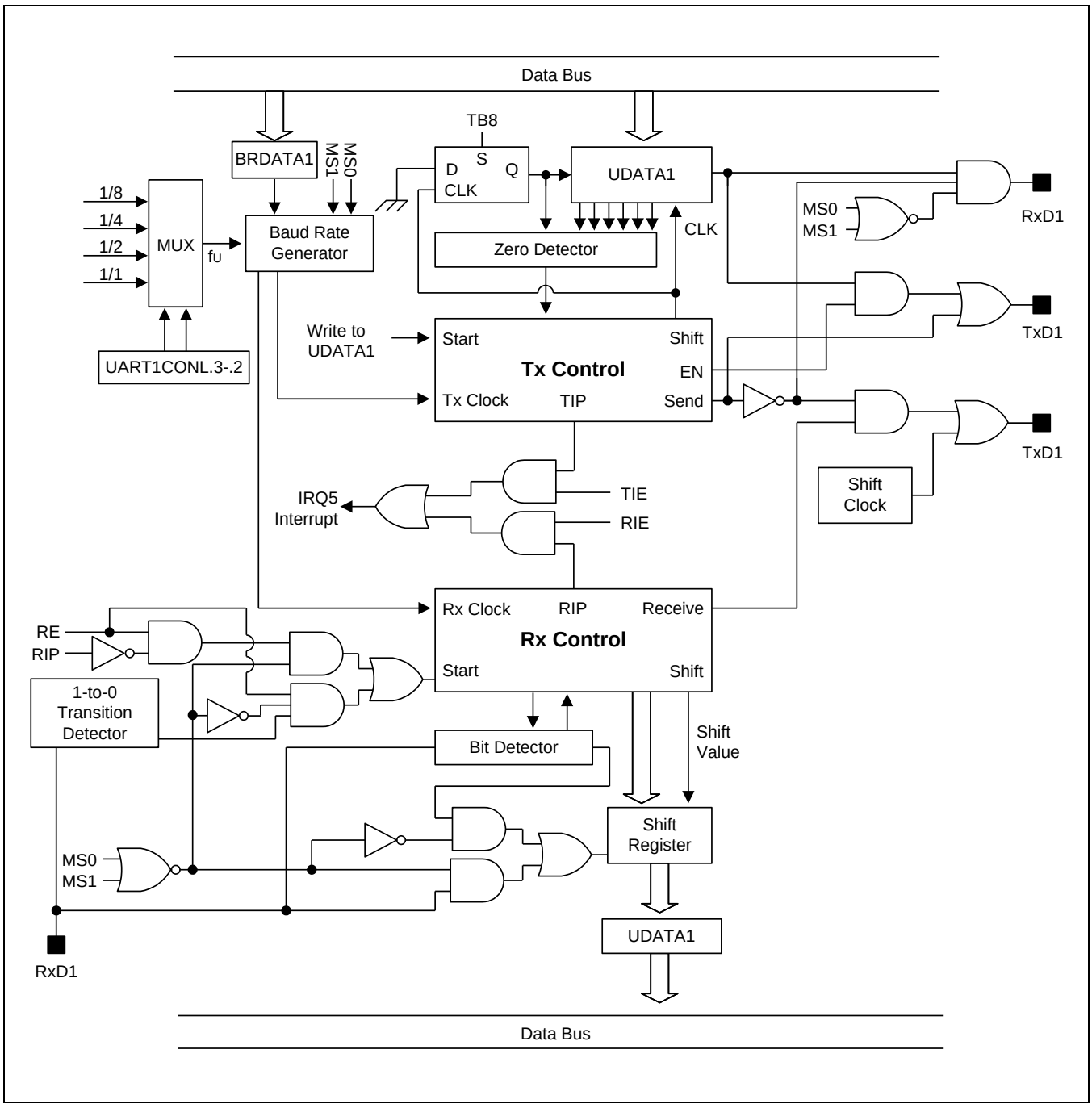


Figure 20-5 UART 1 Functional Block Diagram

20.5 UART 1 Mode 0 Function Description

In mode 0, UART 1 is input and output through the RxD1 (P2.5) pin and TxD1 (P2.4) 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.

20.5.1 Mode 0 Transmit Procedure

The steps of Mode 0 transmit procedure are:

1. Select the UART 1 clock, UART1CONL.3 and .2.
2. Set the UART 1 transmit parity bit autogeneration enable or disable bit (UART1CONL.7).
3. Select mode 0 by setting UART1CONH.7 and .6 to "00B".
4. Write transmission data to the shift register UDATA1 (FBH, Set 1, Bank 1) to start the transmission operation.

20.5.2 Mode 0 Receive Procedure

The steps of Mode 0 receive procedure are:

1. Select the UART 1 clock, UART1CONL.3 and .2.
2. Set the UART 1 transmit parity bit autogeneration enable or disable 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 1 receive enable (RE) bit (UART1CONH.4) to "1".
6. The shift clock will now be output to the TxD1 (P2.4) pin and will read the data at the RxD1 (P2.5) pin. A UART 1 receive interrupt occurs when UART1CONH.1 is set to "1".

Figure 20-6 illustrates the timing diagram for serial port Mode 0 operation.

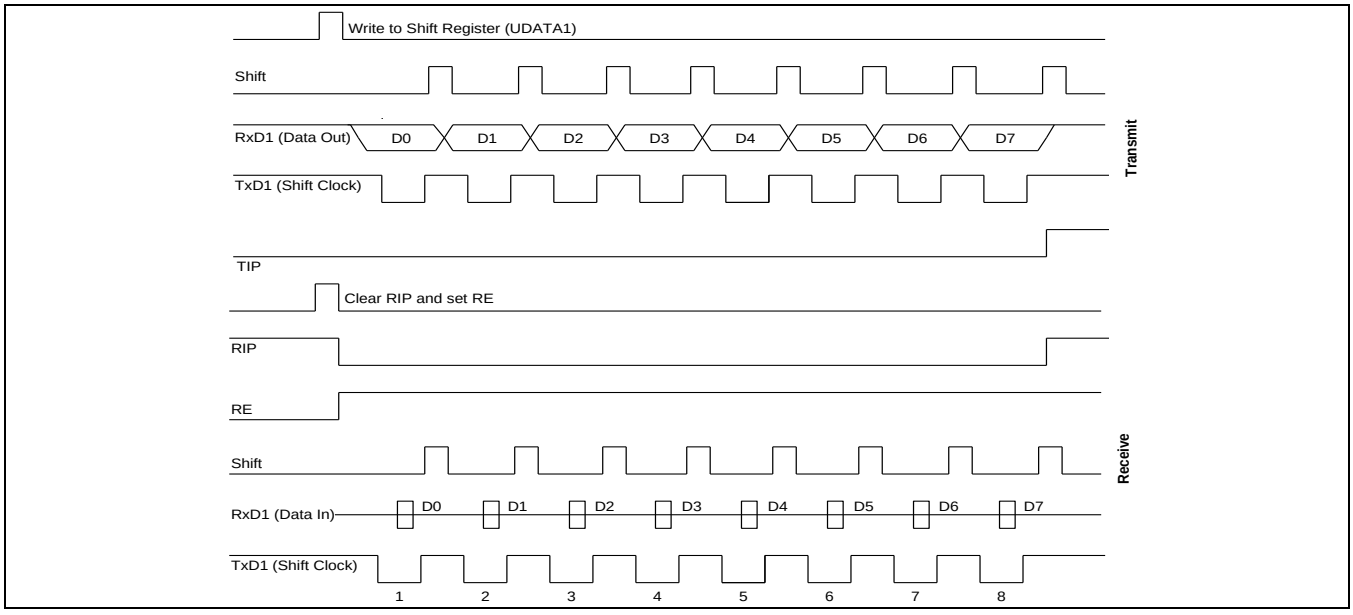


Figure 20-6 Timing Diagram for Serial Port Mode 0 Operation

20.6 Serial Port Mode 1 Function Description

In mode 1, 10 bits are transmitted (through the TxD1 (P2.4) pin) or received (through the RxD1 (P2.5) pin).

The three components of each data frame are:

- Start bit ("0")
- 8 data bits (LSB first)
- Stop bit ("1")

The baud rate for mode 1 is variable.

20.6.1 Mode 1 Transmit Procedure

The steps of Mode 1 transmit procedure are:

1. Select the UART 1 clock, UART1CONL.3 and .2.
2. Set the UART 1 transmit parity bit autogeneration enable or disable bit (UART1CONL.7).
3. Select the baud rate that BRDATA1 generates.
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 (FBH, Set 1, Bank 1). The start and stop bits are generated automatically by hardware.

20.6.2 Mode 1 Receive Procedure

The steps of Mode 1 receive procedure are:

1. Select the UART 1 clock, UART1CONL.3 and .2.
2. Set the UART 1 transmit parity bit autogeneration enable or disable bit (UART1CONL.7).
3. Select the baud rate that BRDATA1 generates.
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 (P2.5) pin will cause the UART 1 module to start the serial data receive operation.

Figure 20-7 illustrates the timing diagram for serial port Mode 1 operation.

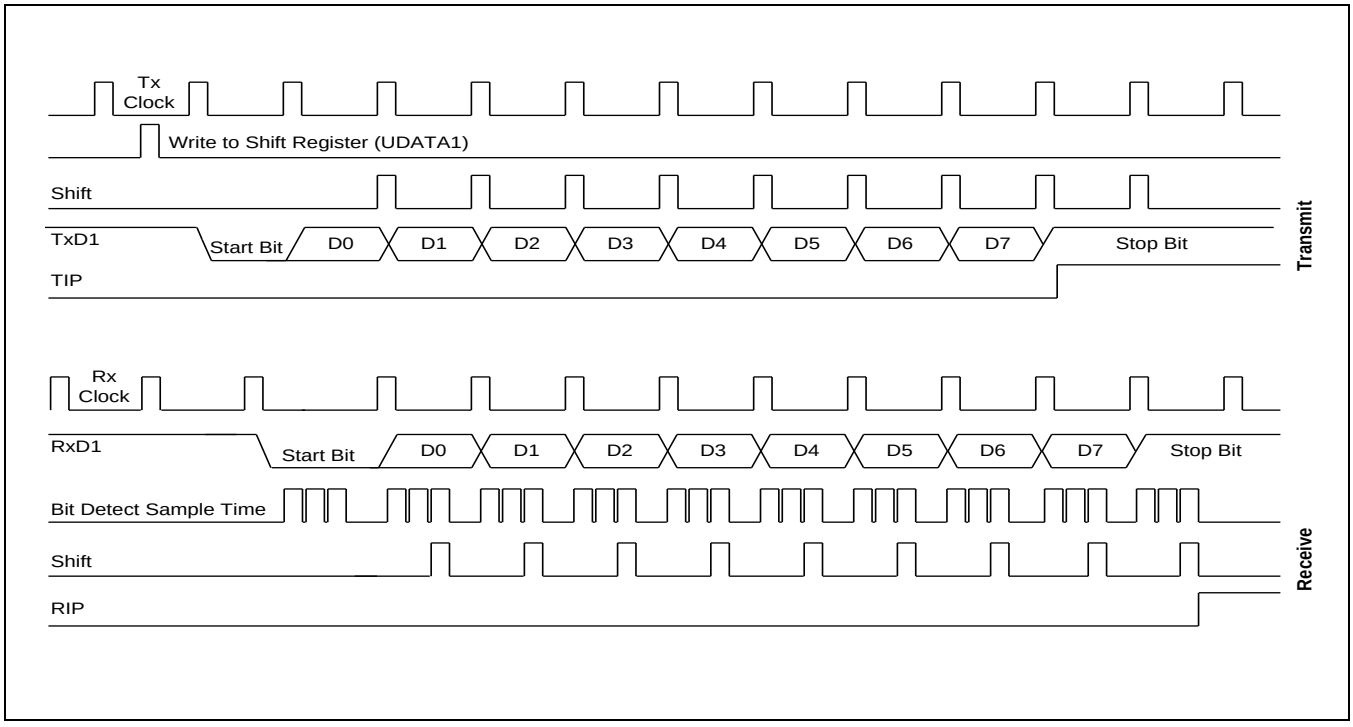


Figure 20-7 Timing Diagram for Serial Port Mode 1 Operation

20.7 Serial Port Mode 2 Function Description

In mode 2, 11 bits are transmitted (through the TxD1 (P2.4) pin) or received (through the RxD1 (P2.5) pin).

The four components of each frame are:

- 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 $f_U/16$ clock frequency.

20.7.1 Mode 2 Transmit Procedure

The steps for Mode 2 transmit procedure are:

1. Select the UART 1 clock, UART1CONL.3 and .2.
2. Set the UART 1 transmit parity bit autogeneration 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 (FBH, Set 1, Bank 1), to start the transmit operation.

20.7.2 Mode 2 Receive Procedure

The steps for Mode 2 receive procedure are:

1. Select the UART 1 clock, UART1CONL.3 and .2.
2. Set the UART 1 transmit parity bit autogeneration 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 (P2.5) pin enters to low level.

Figure 20-8 illustrates the timing diagram for serial port Mode 2 operation.

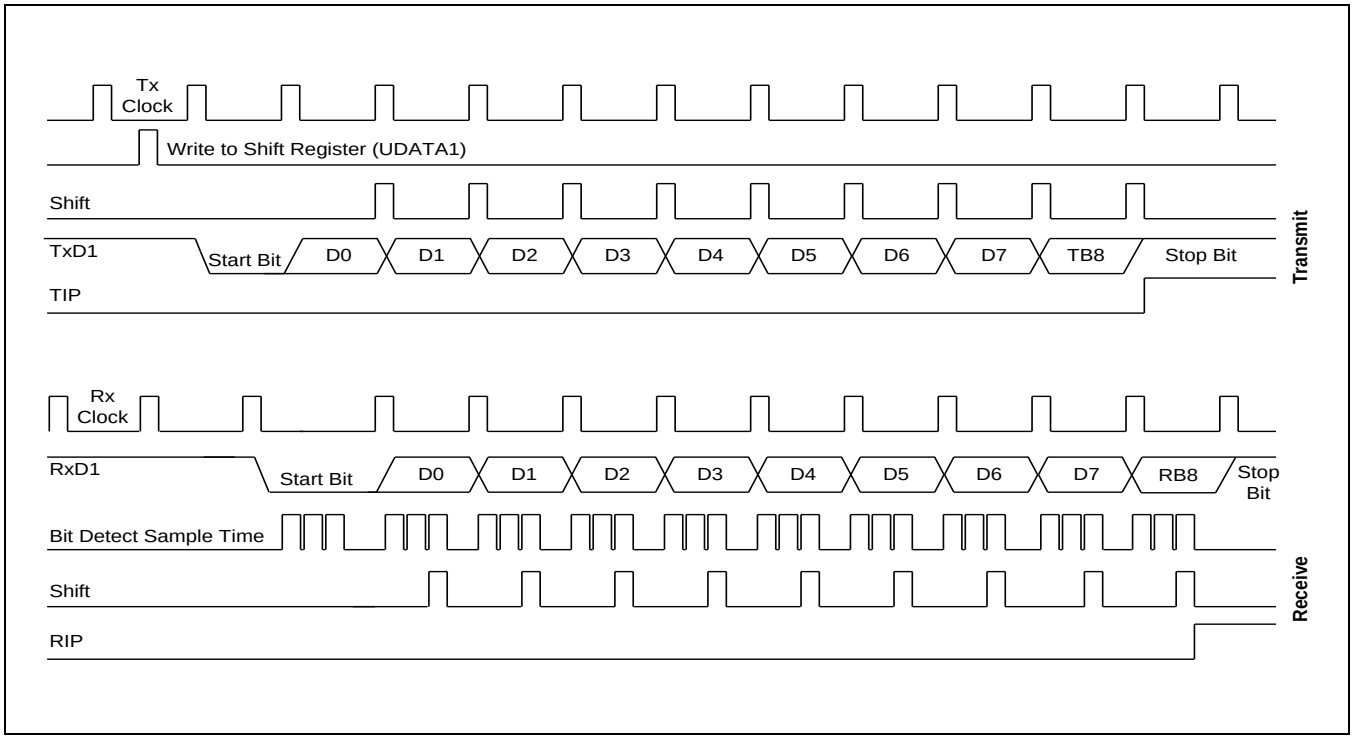


Figure 20-8 Timing Diagram for Serial Port Mode 2 Operation

20.8 Serial Port Mode 3 Function Description

In mode 3, 11 bits are transmitted (through the TxD1 (P2.4) pin) or received (through the RxD1 (P2.5) pin). Mode 3 is identical to mode 2 except for baud rate, which is variable.

The four components of each data frame are:

- Start bit ("0")
- 8 data bits (LSB first)
- Programmable 9th data bit
- Stop bit ("1")

20.8.1 Mode 3 Transmit Procedure

The steps for Mode 3 transmit procedure are:

1. Select the UART 1 clock, UART1CONL.3 and .2.
2. Set the UART 1 transmit parity bit autogeneration 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 (FBH, Set 1, Bank 1), to start the transmit operation.

20.8.2 Mode 3 Receive Procedure

The steps for Mode 3 receive procedure are:

1. Select the UART 1 clock, UART1CONL.3 and .2.
2. Set the UART 1 transmit parity bit autogeneration enable or disable bit (UART1CONL.7).
3. Select mode 3 and set the Receive Enable (RE) bit in the UART1CONH register to "1".
4. The receive operation will be started when the signal at the RxD1 (P2.5) pin enters to low level.

Figure 20-9 illustrates the timing diagram for serial port Mode 3 operation.

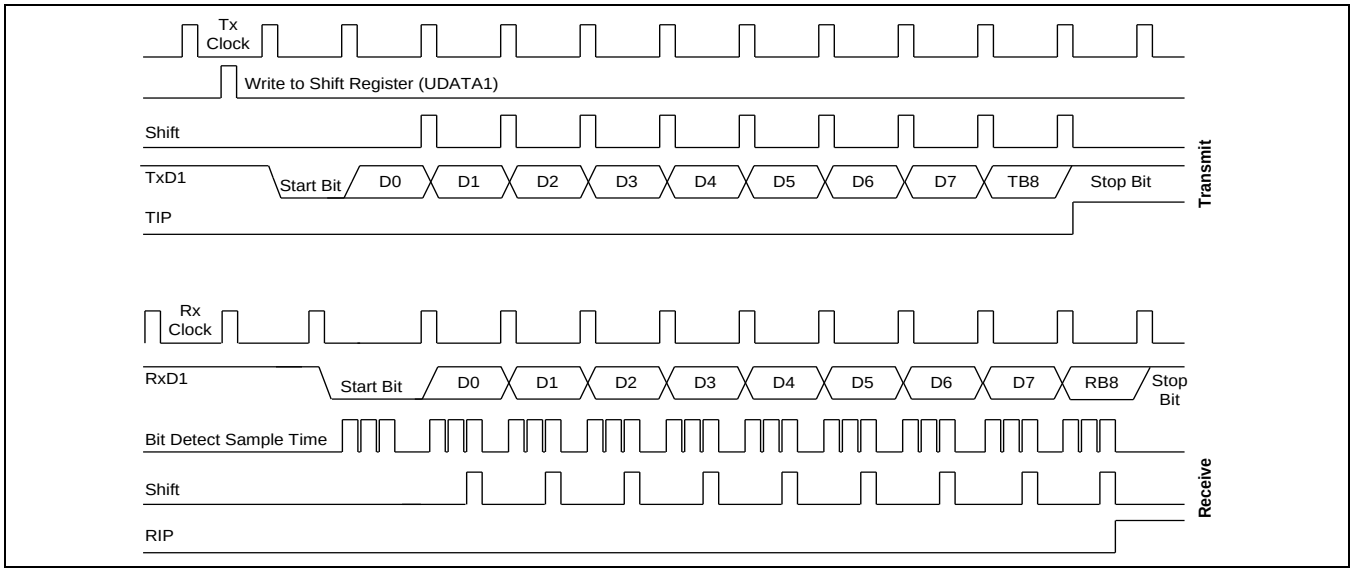


Figure 20-9 Timing Diagram for Serial Port Mode 3 Operation

20.9 Serial Communication for Multiprocessor Configurations

The S3F8-series multiprocessor communication feature enables a "master" S3F8S39/S3F8S35 to send a multiple-frame serial message to a "slave" device in a multi- S3F8S39/S3F8S35 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 should set the MCE bit in the UART1CONH register. When the MCE bit is "1", the 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.

20.9.1 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, you can use it in mode 1 to verify 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.

20.9.2 Setup Procedure for Multiprocessor Communications

The steps to configure multiprocessor communications are:

1. Set all S3F8S39/S3F8S35 devices (masters and slaves) to UART 1 mode 2 or 3.
2. Write the MCE bit of all the slave devices to "1".
3. The transmission protocol of master device 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 20-10 illustrates the connection example of multiprocessor serial data communications.

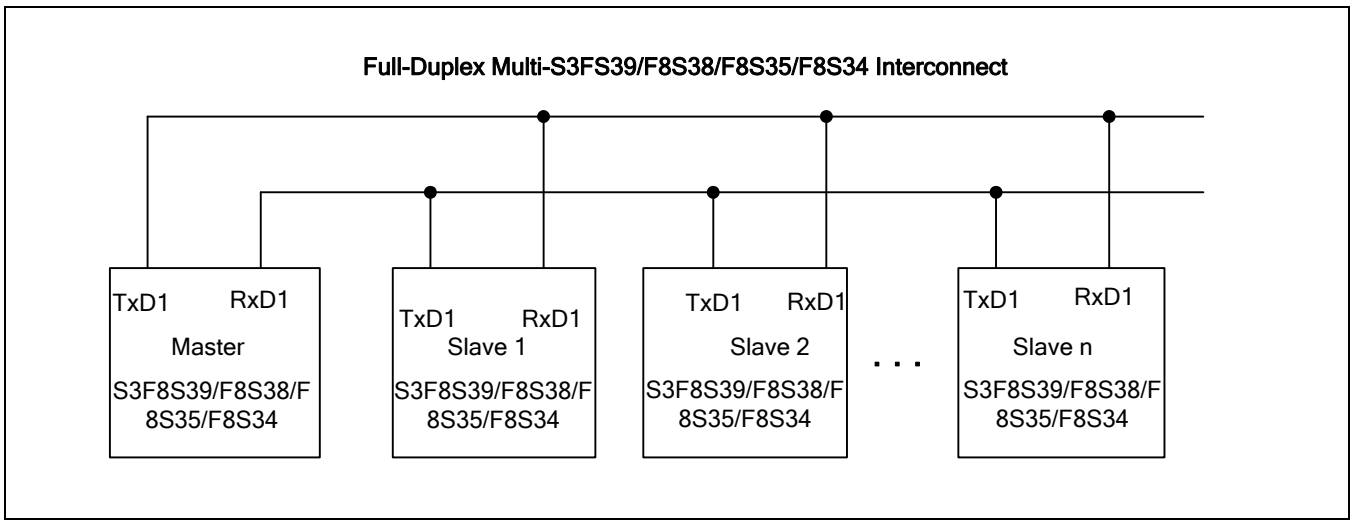


Figure 20-10 Connection Example for Multiprocessor Serial Data Communications

21 Embedded Flash Memory Interface

21.1 Overview

S3F8S39/S3F8S35 has an on-chip flash memory internally instead of masked ROM. The flash memory is accessed by "LDC" instruction and this is a sector erasable and byte programmable flash. You can program the data in a flash memory area any time.

The two operating features of S3F8S39/S3F8S35's embedded 32K/16K-byte memory:

- Tool Program Mode (Refer to Chapter 25 for more information)
- User Program Mode

21.2 User Program Mode

This mode supports:

- Sector erase
- Byte programming
- Byte read
- Protection mode (Hard lock protection).

The read protection mode is available only in Tool Program mode. Therefore, to make a chip into read protection, you must select a read protection option when you program by using a programming tool.

S3F8S39/S3F8S35 has the pumping circuit internally. Therefore, 12.5 V into Vpp (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
- Hard lock protection

NOTE:

1. Ensure to execute the DI instruction before starting user program mode. The user program mode verifies The interrupt request register (IRQ). When an interrupt request is generated, user program mode stops.
2. User program mode also stops by an interrupt request that even masked in the DI status. To Prevent this; disable the interrupt by using each peripheral interrupt enable bit.

21.2.1 Flash Memory Control Register

Flash memory control register (FMCON) register is available only in user program mode to:

- Select the Flash Memory operation mode
- Select sector erase
- Select byte programming
- Make the flash memory into a hard lock protection.

Figure 21-1 illustrates the flash memory control register.

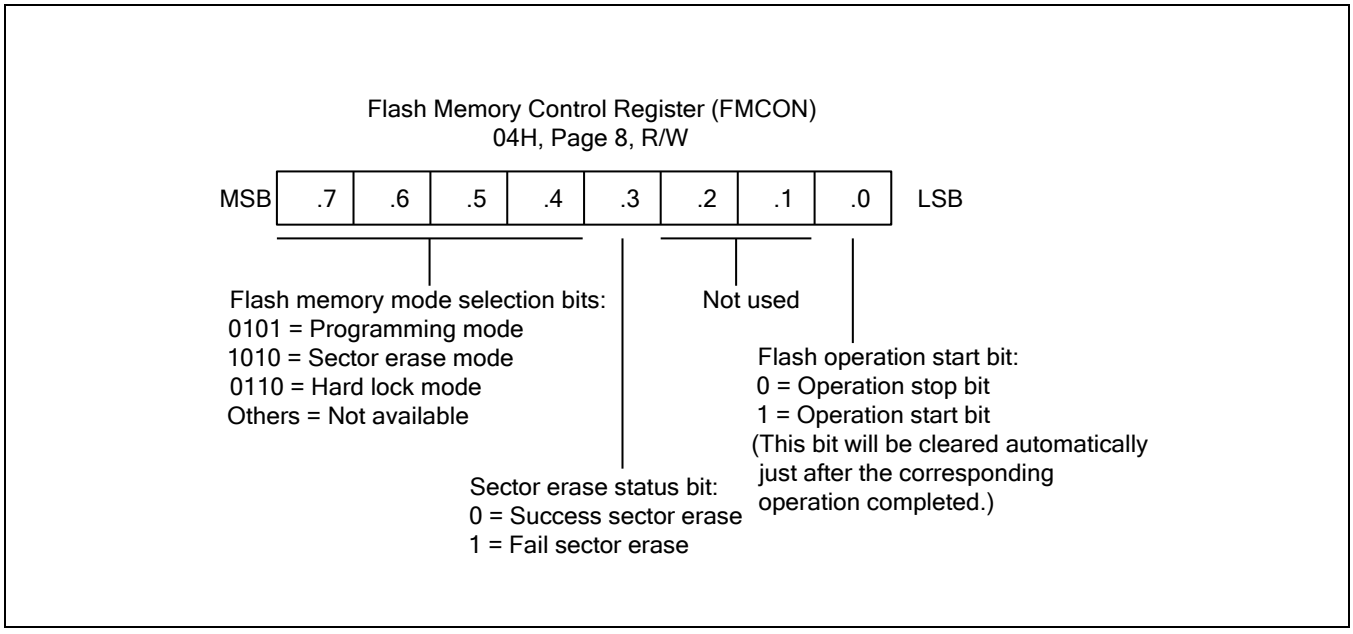


Figure 21-1 Flash Memory Control Register (FMCON)

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 Erase (Sector erase) or hard lock instruction need execution time. Therefore, you should insert wait time 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. If an interrupt is requested during the operation of "Sector erase", the operation of "Sector erase" is discontinues, and the interrupt is served by CPU. Therefore, you should verify the status bit after executing "Sector erase". The "sector erase" operation is successful when the bit is logic "0", and is failure if the bit is logic "1".

21.2.2 Flash Memory User Programming Enable Register

The FMUSR register is used for a safety operation of the flash memory. This register protects undesired erase or program operation from malfunctioning of CPU that is due to electrical noise.

After reset, the user-programming mode is disabled, because the value of FMUSR is "00000000B" by reset operation. To operate the flash memory, you can use the user programming mode by setting the value of FMUSR to "10100101B". If FMUSR is set to value other than "10100101b", User Program mode is disabled.

Figure 21-2 illustrates the flash memory user programming enable register.

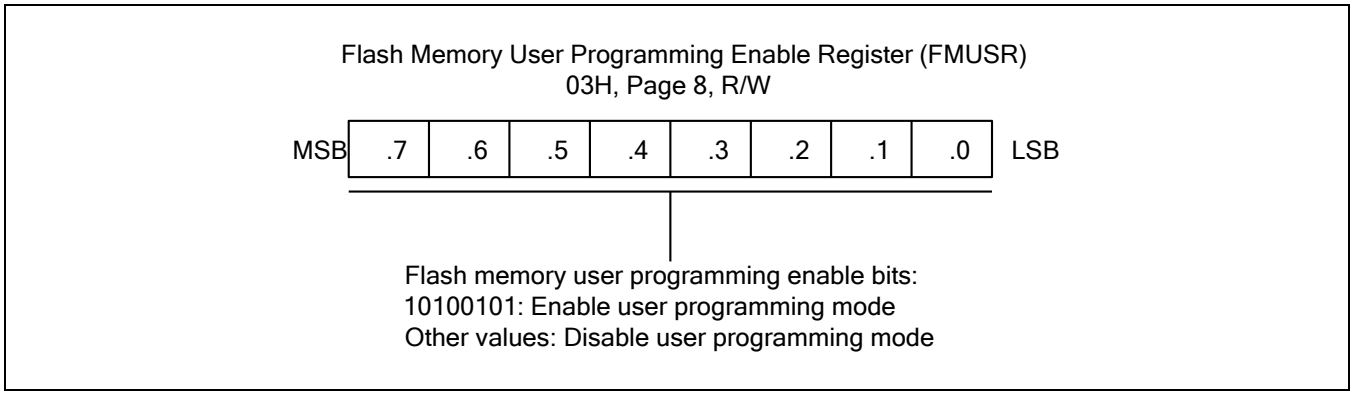


Figure 21-2 Flash Memory User Programming Enable Register (FMUSR)

21.2.3 Flash Memory Sector Address Registers

There are two sector address registers for addressing a sector to be erased. The Flash Memory Sector Address Register Low Byte (FMSECL) indicates the low byte of sector address. The Flash Memory Sector Address Register High Byte (FMSECH) indicates the high byte of sector address. The FMSECH is needed for S3F8S39/S3F8S35 because it has 128 sectors, respectively. One sector consists of 128 bytes. Each address of sector starts with XX00H or XX80H, that is, a base address of sector is XX00H or XX80H. Therefore, FMSECL register 6-0 does not indicate whether the value is "1" or "0".

It is recommend to load the 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 verify whether next address is located in the same sector or not. In case of other sectors, you should load sector address to FMSECH and FMSECL register according to the sector.

Figure 21-3 illustrates the flash memory sector address register high byte.

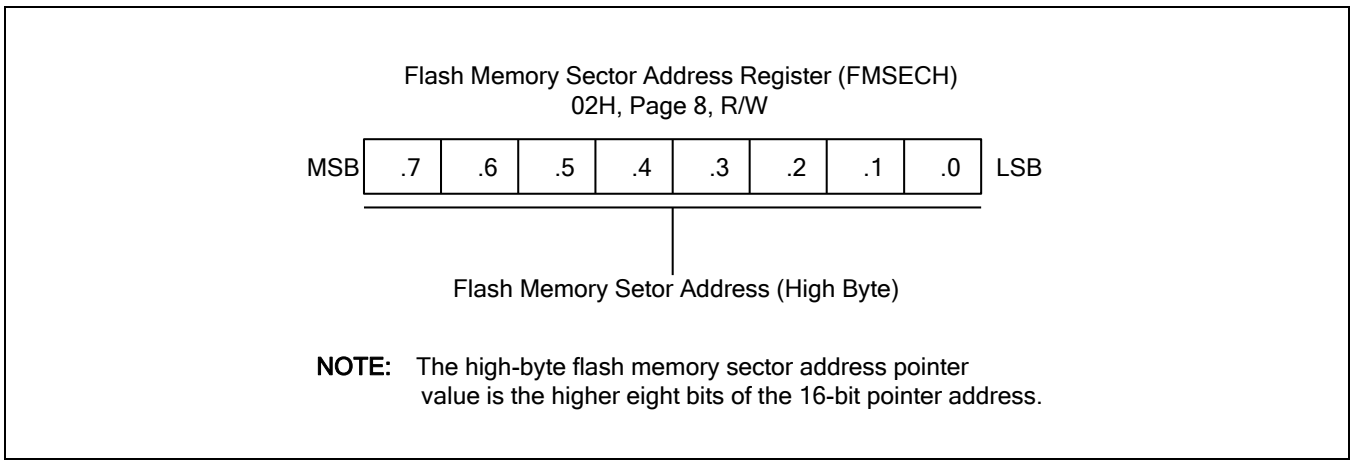


Figure 21-3 Flash Memory Sector Address Register High Byte (FMSECH)

Figure 21-4 illustrates the flash memory sector address register low byte.

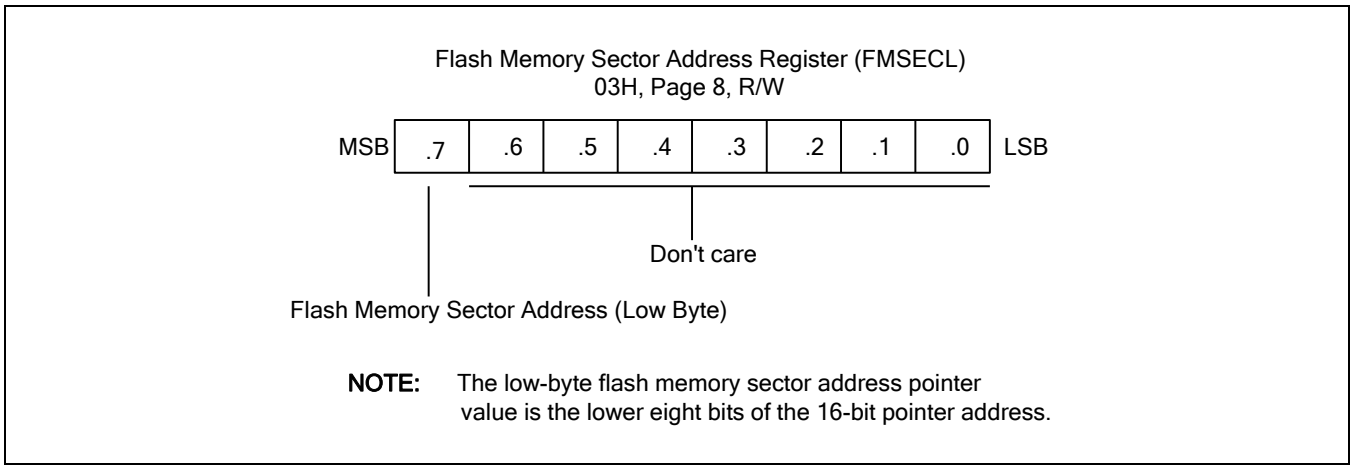


Figure 21-4 Flash Memory Sector Address Register Low Byte (FMSECL)

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 cannot 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 do not want to use ISP sector, you can use this area as a normal program memory (can erase or program 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 using Serial programming tools.

The size of ISP sector can vary by settings of Smart Option. You can select appropriate ISP sector size according to the size of On Board Program software.

Figure 21-5 illustrates the program memory address space.

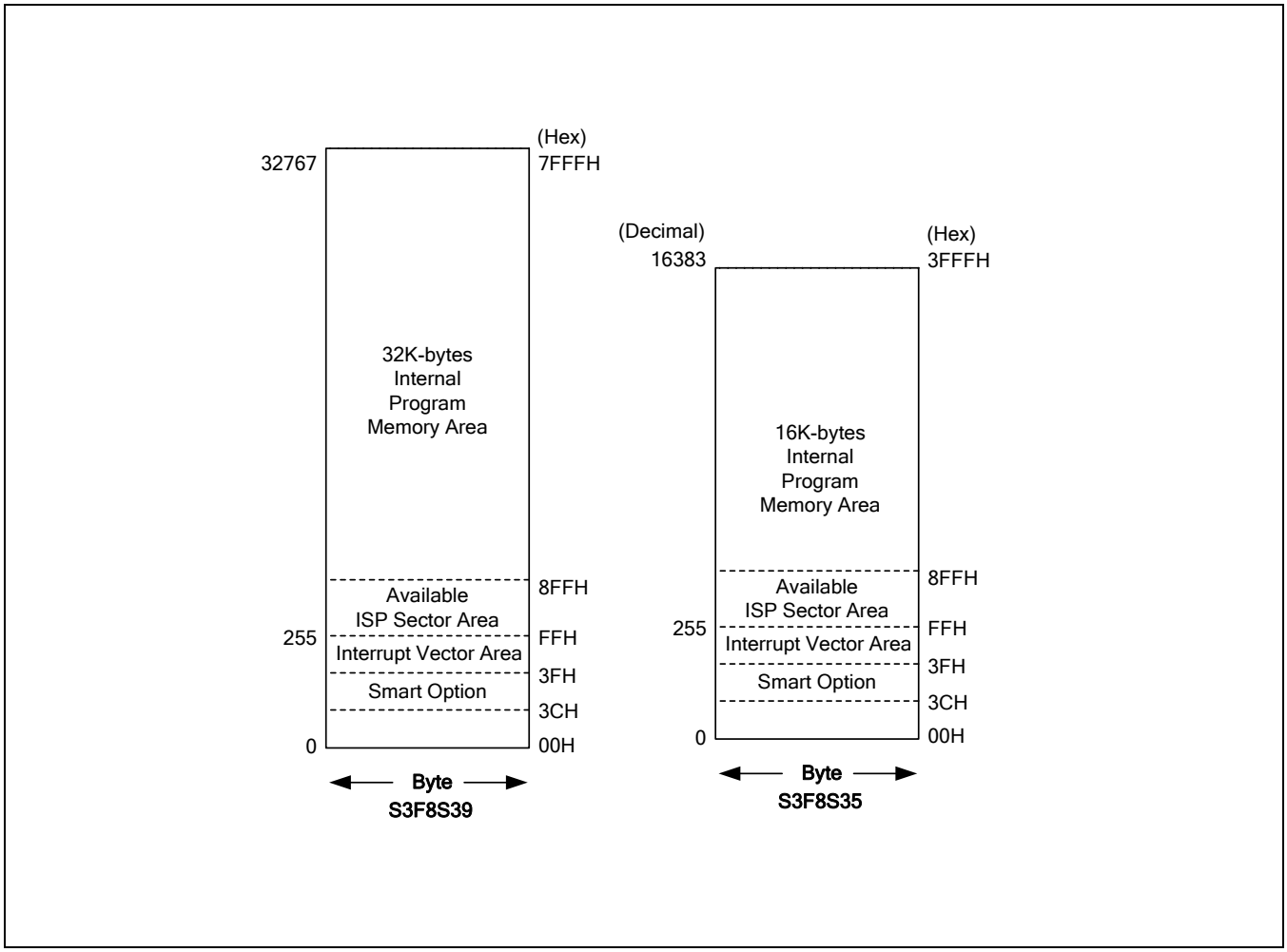


Figure 21-5 Program Memory Address Space

Table 21-1 lists the ISP sector size.

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	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) cannot be erased and programmed by LDC instruction in User Program mode.

21.4 ISP Reset Vector and ISP Sector Size

At the Smart Option, when you use ISP sectors by setting the ISP Enable/Disable bit to "0" and the Reset Vector Selection bit to "0", you can select the reset vector address of CPU as listed in Table 21-2. You can do this by setting the ISP Reset Vector Address Selection bits.

Table 21-2 lists the reset vector address.

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 the Smart Option (003EH.7–003EH.5) does not depend on the selection of ISP sector size by the Smart Option (003EH.2–003EH.0).

21.5 Sector Erase

You can erase a flash memory partially by using sector erase function only in User Program Mode. The only unit of flash memory that you can erase or program in User Program Mode is called sector.

The program memory of S3F8S39/S3F8S35 is divided into 256/128 sectors for unit of erase and programming, respectively. Every sector has all 128 byte sizes of program memory areas. Therefore, you should erase each sector first to program a new data (byte) into a sector. It requires minimum 10ms delay time for erase after setting sector address and triggering erase start bit (FMCON.0). Tool Program Modes (MDS mode tool or Programming tool) does not supports sector erase.

Figure 21-6 illustrates the sector configurations in user program mode.

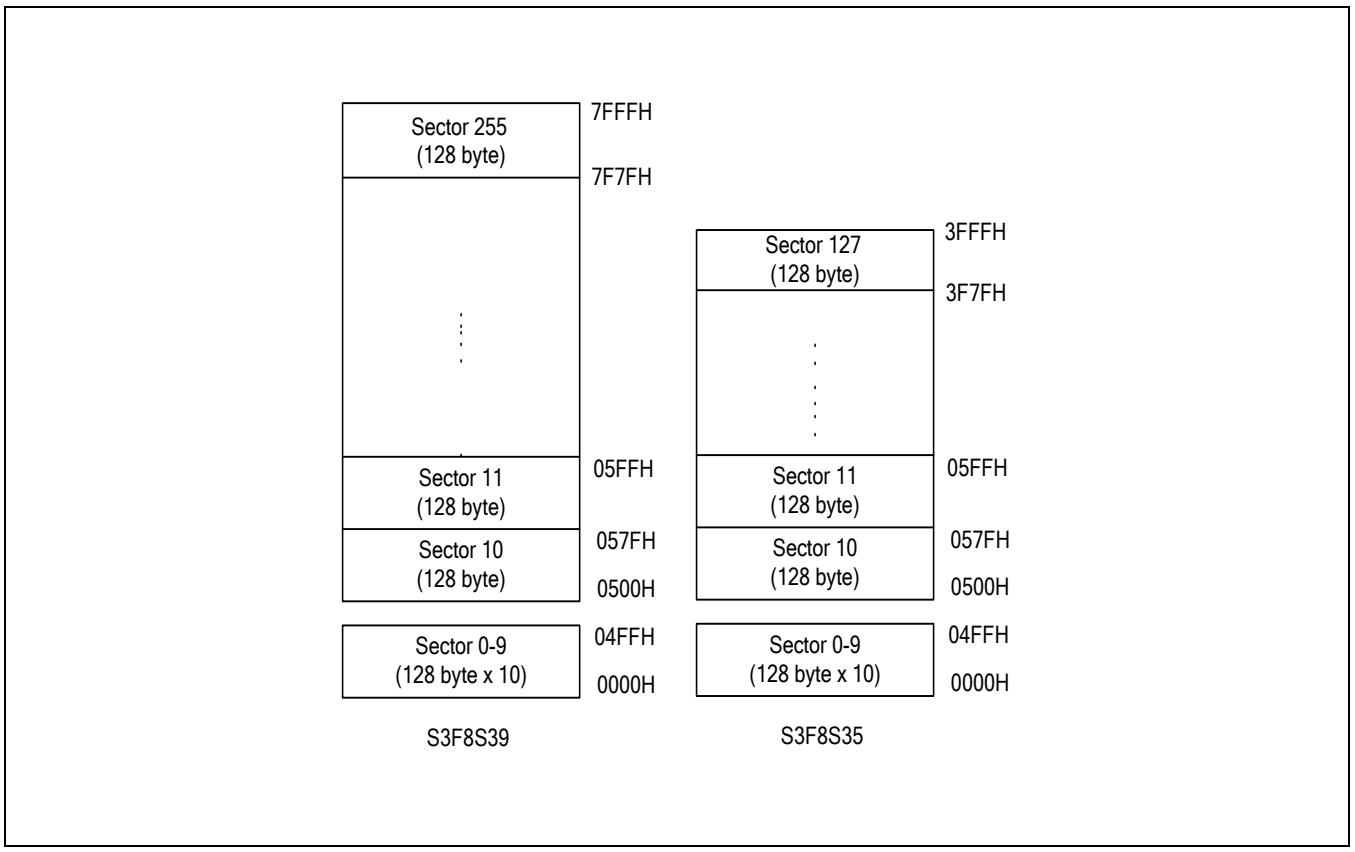


Figure 21-6 Sector Configurations in User Program Mode

The Sector Program Procedure in User Program Mode

1. Set Flash Memory User Programming Enable Register (FMUSR) to "10100101B".
2. Set Flash Memory Sector Address Register (FMSECH/ FMSECL).
3. Set Flash Memory Control Register (FMCON) to "10100001B".
4. Set Flash Memory User Programming Enable Register (FMUSR) to "00000000B".
5. Verify the "Sector erase status bit" whether "Sector erase" is success or not.

Example 21-1 shows the sector erase.

Example 21-1 Sector Erase

```
                •
                •
                SB1
reErase:        LD      FMUSR, #0A5H        ; User Program mode enable
                LD      FMSECH, #10H
                LD      FMSECL, #00H        ; Set sector address (1000H-107FH)
                LD      FMCON, #10100001B   ; Start sector erase
                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
```

21.6 Programming

A flash memory is programmed in 1 byte unit after sector erase. And programming for safety's sake, you should 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 the limit of flash sector is 128 byte. Therefore when you write 128 byte, you should reset FMSECH and FMSECL.

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 Control Register (FMCON) to "01010001B".
4. Set Flash Memory Sector Register (FMSECH, FMSECL) to sector value of write address.
5. Load a transmission data into a working register.
6. Load a flash memory upper address into upper register of pair working register.
7. Load a flash memory lower address into lower register of pair working register.
8. Load transmission data to flash memory location area on "LDC" instruction by indirectly addressing mode Set Flash Memory User Programming Enable Register (FMUSR) to "00000000B".

Example 21-2 shows the program procedure.

Example 21-2 Program

```
•
•
SB1
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
LD              FMUSR, #0A5H            ; User Program mode enable
LD              FMCON, #01010001B       ; Start program
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
```

21.7 Reading

The read operation of programming starts by "LDC" instruction.

Program 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 shows the read operation.

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
                •
                •
                •
                •
```

21.8 Hard Lock Protection

You can set Hard Lock Protection by writing "0110" in FMCON7-4. If this function is enabled, then you 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 or user program mode). Following chip erase execution. Whereas in tool mode the manufacturer of serial tool writer could support hardware Protection. Please refer to manual of serial program writer tool provided by the manufacturer.

Program procedure in User program Mode

1. Set Flash Memory User Programming Enable Register (FMUSR) to "10100101B".
2. Set Flash Memory Control Register (FMCON) to "01100001B".
3. Set Flash Memory User Programming Enable Register (FMUSR) to "00000000B".

Example 21-4 shows the hard lock protection.

Example 21-4 Hard Lock Protection

```
•
•
SB1
LD          FMUSR, #0A5H          ; User Program mode enable
LD          FMCON, #01100001B     ; Hard Lock mode set & start
NOP                               ; Dummy Instruction, This instruction must be
                                    needed
LD          FMUSR, #0             ; User Program mode disable
•
•
```

22 Low Voltage Detect and Low Voltage Reset

22.1 Low Voltage Reset

You can select internal RESET (LVR) or external RESET by using Smart Option (3FH.7 in ROM).

You can reset S3F8S39/S3F8S35 in four ways:

- By external power-on-reset
- By the external reset input pin pulled low
- By the digital watchdog timing out
- By the Low Voltage Reset (LVR) circuit

During an external power-on reset, the voltage V_{DD} is High level and the nRESET pin is forced Low level. The nRESET signal is input through a Schmitt trigger circuit where it is synchronized with the CPU clock. This brings the S3F8S39/S3F8S35 into a known operating status. To ensure correct start-up, you should ensure that reset signal is not released before the V_{DD} level is sufficient to allow MCU operation at the chosen frequency.

You should hold nRESET pin to Low level for a minimum time interval after the power supply comes within tolerance to allow time for internal CPU clock oscillation to stabilize.

When a reset occurs during normal operation (with both V_{DD} and nRESET at High level), it forces the signal at the nRESET pin to Low and the reset operation starts. All system and peripheral control registers are then set to their default hardware reset values Refer to *Table 8-1* for more information.

The MCU provides a watchdog timer function to ensure graceful recovery from software malfunction. If it does not refresh watchdog timer before it reaches end-of-counter condition (overflow), then it activates the internal reset.

The S3F8S39/S3F8S35 has a built-in low voltage reset circuit. It provides detection of power voltage drop of external V_{DD} input level to prevent MCU from malfunctioning in an unstable MCU power level. This voltage detector works for the reset operation of MCU. This low voltage reset includes an analog comparator and Vref circuit. Hardware internally sets the value of a detection voltage. The on-chip LVR, features static reset when supply voltage is below a reference voltage value (Typical 1.9/2.3/3.0/3.9 V). This feature can remove external reset circuit while keeping the application safe. As long as the supply voltage is below the reference value, an internal static RESET will be triggered. The MCU can start only when the supply voltage rises over the reference voltage.

When you calculate power consumption, remember that a static current of LVR circuit should be added a CPU operating current in any operating modes such as STOP, IDLE, and normal RUN mode.

Figure 22-1 illustrates the Low Voltage Reset Circuit.

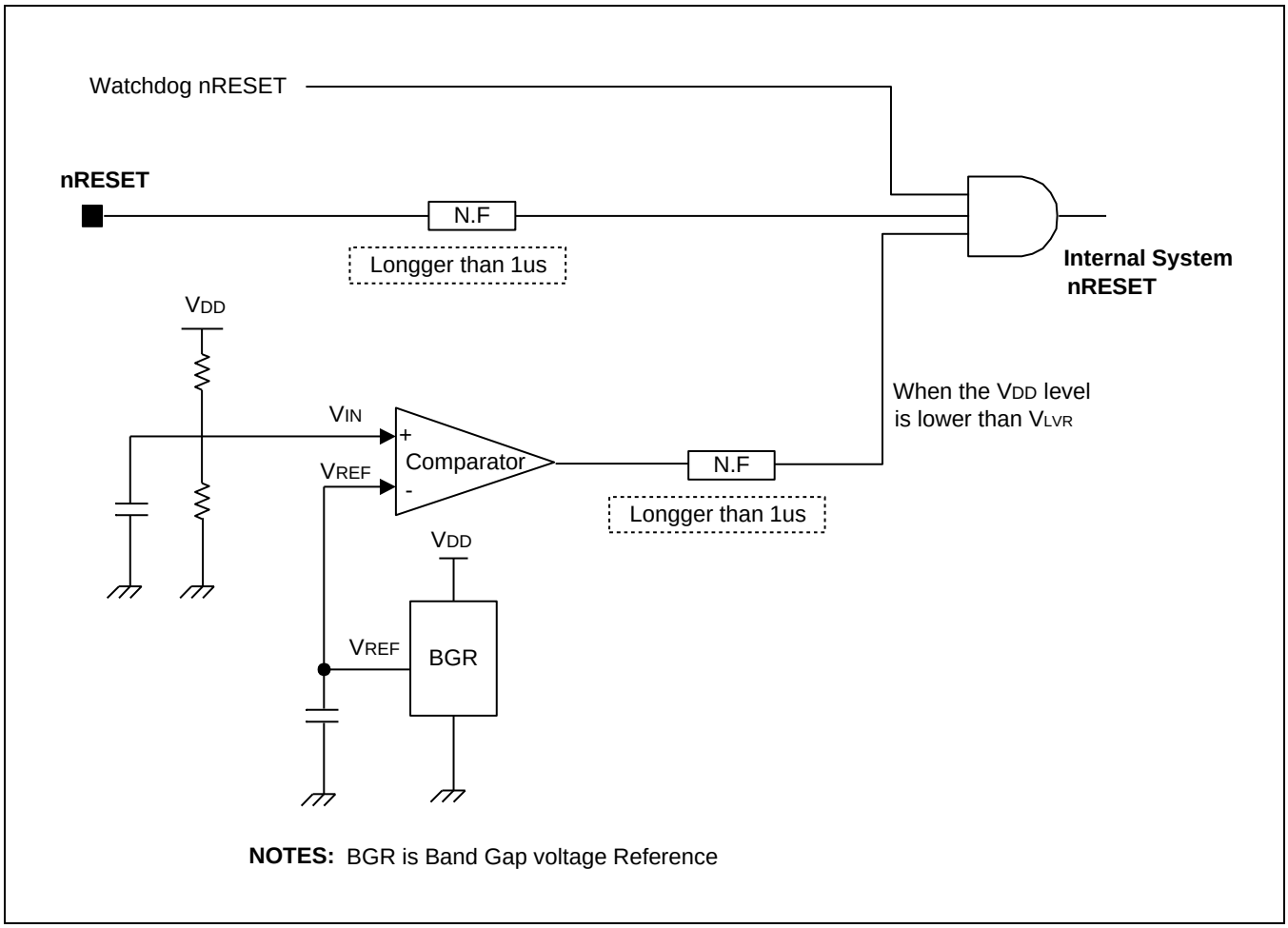


Figure 22-1 Low Voltage Reset Circuit

NOTE: To program the duration of the oscillation stabilization interval, 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), disable it by writing "1010B" to the upper nibble of BTCON.

22.2 Low Voltage Detect

The S3F8S39/S3F8S35 micro-controller has a built-in Low Voltage Detector (LVD) circuit that allows LVD interrupt and LVD_FLAG detection of power voltage. The S3F8S39/S3F8S35 has four options in LVD voltage level according to the LVDCON register (Refer to *Figure 22-3* for more information).

- Low voltage detect level Flag Bit (LVD_FLAG): 2.1, 2.5, 3.2, and 4.1 V (Typical) ± 100 mV

LVD block can be enabled or disabled by LVDCON.7. If you enable the LVD interrupt, when $V_{DD} < V_{LVD}$, it generates an LVD interrupt.

The LVD block of S3F8S39/S3F8S35 consists of one comparator and a resistor string. The comparator is for LVD detection.

The output of comparator makes LVD indicator flag bit "1" or "0". That it uses to indicate low voltage level. When the power voltage is below the LVD level, the bit 5 of LVDCON register is set "1". When the power voltage is above the LVD level, the bit 5 of LVDCON register is set '0' automatically. You can use LVDCON.5 flag bit to indicate low battery in applications.

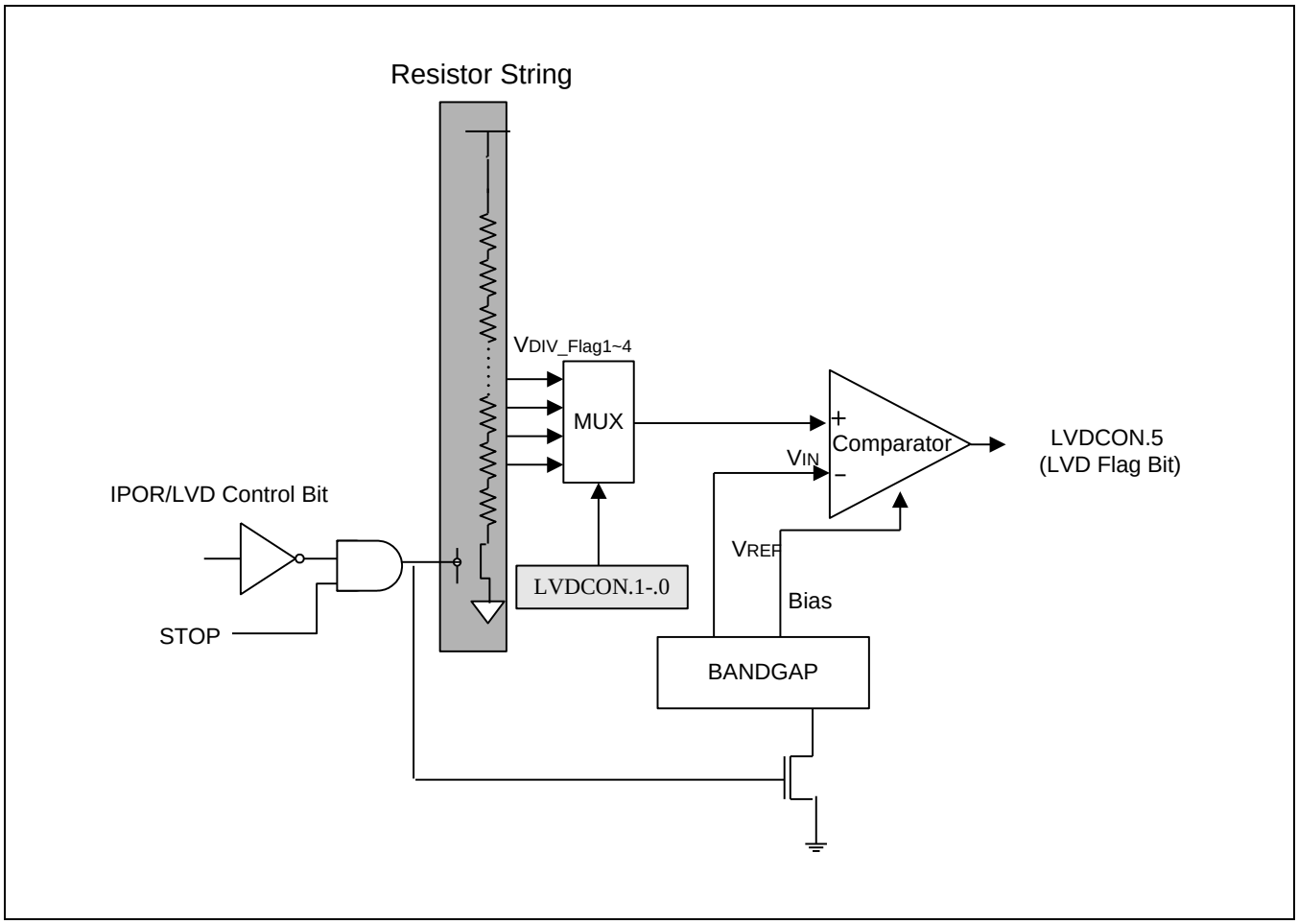


Figure 22-2 Low Voltage Detect (LVD) Block Diagram

Figure 22-3 illustrates the Low Voltage Detect Control Register (LVDCON).

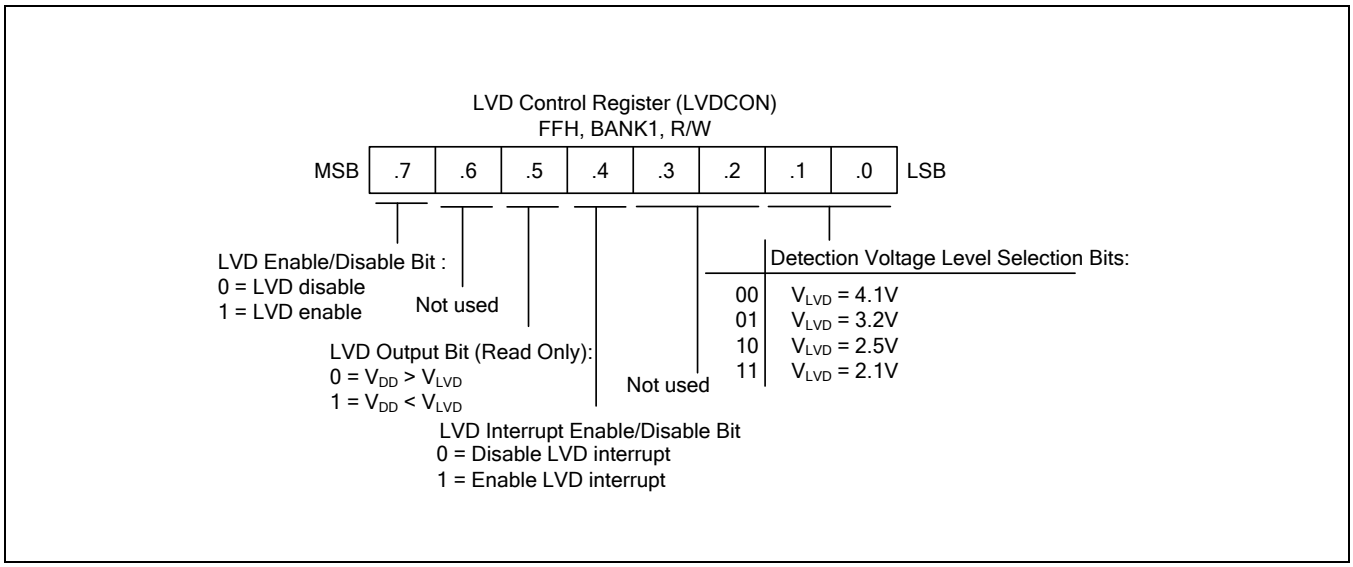


Figure 22-3 Low Voltage Detect Control Register (LVDCON)

23 Electrical Data

23.1 Overview

In this section, the following S3F8S39/S3F8S35 electrical characteristics are presented in tables and graphs:

- Absolute maximum ratings
- D.C. electrical characteristics
- A.C. electrical characteristics
- Input timing measurement points
- Oscillator characteristics
- Oscillation stabilization time
- Operating voltage range
- Schmitt trigger input characteristics
- Data retention supply voltage in Stop Mode
- Stop Mode release timing when initiated by a RESET
- A/D converter electrical characteristics
- Universal Asynchronous Receiver/Transmitter (UART) characteristics
- Low Voltage Reset (LVR) circuit characteristics
- LVR reset timing
- LVD circuit characteristics
- Full-Flash memory characteristics
- Electrostatic Discharge (ESD) Characteristics

23.2 Absolute Maximum Ratings

Table 23-1 lists the absolute maximum ratings.

Table 23-1 Absolute Maximum Ratings

(T_A = 25 °C)

Parameter	Symbol	Conditions	Rating	Unit
Supply voltage	V_{DD}	–	– 0.3 to + 6.5	V
Input voltage	V_I	All ports	– 0.3 to V_{DD} + 0.3	
Output voltage	V_O	All output ports	– 0.3 to V_{DD} + 0.3	
Output current high	I_{OH}	One I/O pin active	– 25	mA
–	–	All I/O pins active	– 80	–
Output current low	I_{OL}	One I/O pin active	+ 30	mA
–	–	All I/O pins active	+ 100	–
Operating temperature	T_A	–	–40 to +85	°C
Storage temperature	T_{STG}	–	– 65 to + 150	

23.3 DC Electrical Characteristics

Table 23-2 lists the DC electrical characteristics.

Table 23-2 DC Electrical Characteristics

(T_A = –40°C to +85 °C, V_{DD} = 1.8 to 5.5 V)

Parameter	Symbol	Conditions		Min.	Typ.	Max.	Unit
Operating voltage	V_{DD}	T_A = –40°C to +85°C		1.8	–	5.5	V
Main crystal or ceramic frequency	fmain	V_{DD} = 2.7 to 5.5 V		0.4	–	12	MHz
		V_{DD} = 1.8 to 2.7 V		0.4	–	4	
Input high voltage	V_{IH1}	Ports 0, 1, 2, 3 and nRESET	V_{DD} = 1.8 to 5.5 V	0.8 V_{DD}	–	V_{DD}	V
	V_{IH2}	X_{IN} and X_{OUT}		V_{DD} – 0.1			–
Input low voltage	V_{IL1}	Ports 0, 1, 2, 3 and nRESET	V_{DD} = 1.8 to 5.5 V	–	–	0.2 V_{DD}	V
	V_{IL2}	X_{IN} and X_{OUT}				0.1	–
Output high voltage	V_{OH}	I_{OH} = – 1 mA Ports 0, 1, 2, 3	V_{DD} = 4.5 to 5.5 V	V_{DD} – 1.0	–	–	V
Output low voltage	V_{OL}	I_{OL} = 15 mA Ports 0, 1, 2, 3	V_{DD} = 4.5 to 5.5 V	–	0.4	2.0	
Input high leakage current	I_{LIH1}	All input except I_{LIH2}	V_{IN} = V_{DD}	–	–	1	µA
	I_{LIH2}	X_{IN}	V_{IN} = V_{DD}			20	–
Input low leakage current	I_{LIL1}	All input except I_{LIL2}	V_{IN} = 0 V	–	–	– 1	µA
	I_{LIL2}	X_{IN}	V_{IN} = 0 V			– 20	–
Output high leakage current	I_{LOH}	All output pins	V_{OUT} = V_{DD}	–	–	2	µA
Output low leakage current	I_{LOL}	All output pins	V_{OUT} = 0 V	–	–	– 2	
Pull-up resistors	R_{P1}	V_{IN} = 0 V, Ports 0, 1, 2, 3	V_{DD} = 5 V T_A = 25 °C	25	50	100	kΩ
	R_{P2}	V_{IN} = 0 V, nRESET	V_{DD} = 5 V T_A = 25 °C	125	250	500	
Supply current	I_{DD1}	Run mode 12 MHz CPU clock	V_{DD} = 4.5 to 5.5 V	–	2.2	3	mA
		Run mode 4 MHz CPU clock	V_{DD} = 4.5 to 5.5 V	–	1.2	1.5	
		Run mode 1 MHz CPU clock	V_{DD} = 3.0 V	–	0.4	1	
	I_{DD2}	Idle Mode 12 MHz CPU clock	V_{DD} = 4.5 to 5.5 V	–	1.3	2.3	–
		Idle Mode 4 MHz CPU clock	V_{DD} = 4.5 to 5.5 V		0.8	1	
		Idle Mode	V_{DD} = 3.0 V	–	0.27	0.54	mA

Parameter	Symbol	Conditions		Min.	Typ.	Max.	Unit
		1 MHz CPU clock					
	I_{DD3}	Stop Mode	V_{DD} = 4.5 to 5.5 V	–	0.4	2.3	µA
			V_{DD} = 4.5 to 5.5 V (Ring Oscillator on, Stop Wake-Up Timer on)		0.6	2.5	
			V_{DD} = 4.5 to 5.5 V (LVR enable)		30	50	

NOTE:

1. Supply current does not include current drawn through internal pull-up resistors, the LVR block, LVD block, ADC block and external output current loads.
2. I_{DD3} is current when main clock oscillation stops.

23.4 AC Electrical Characteristics

Table 23-3 lists the AC electrical characteristics.

Table 23-3 AC Electrical Characteristics

(T_A = –40°C to +85°C, V_{DD} = 1.8 to 5.5 V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Interrupt input high, low width	t_{INTH} t_{INTL}	INT0, INT1 V_{DD} = 5 V ± 10 %	500	–	–	ns
RESET input low width	t_{RSL}	Input V_{DD} = 5 V ± 10 %	10	–	–	µs

Figure 23-1 illustrates the input timing measurement points.

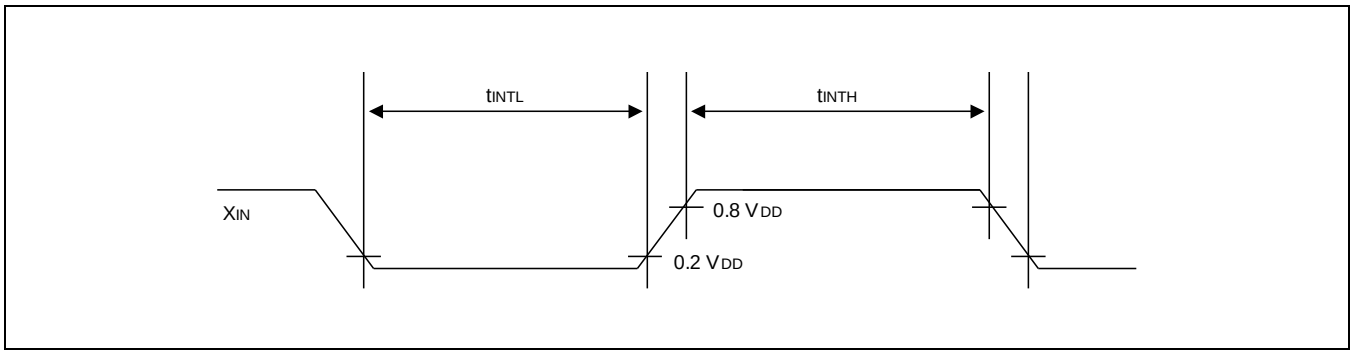


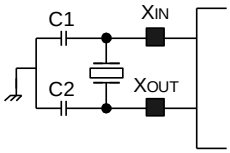
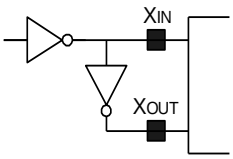
Figure 23-1 Input Timing Measurement Points

23.5 Oscillator Characteristics

Table 23-4 and *Table 23-5* list the oscillator characteristics.

Table 23-4 Oscillator Characteristics (S3F8S35XZZ, S3F8S39XZZ)

(T_A = –40°C to +85°C, V_{DD} = 1.8 to 5.5 V)

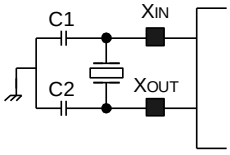
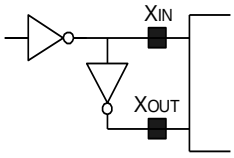
Oscillator	Clock Circuit	Test Condition	Min.	Typ.	Max.	Unit
Main crystal or ceramic		V_{DD} = 2.7 to 5.5 V	0.4	–	12	MHz
		V_{DD} = 1.8 to 2.7 V	0.4	–	4	
External clock (Main System)		V_{DD} = 2.7 to 5.5 V	0.4	–	12	
		V_{DD} = 1.8 to 2.7 V	0.4	–	4	
Tolerance of internal RC oscillator (8/4/2/0.5 MHz)	–	T_A = 25 °C V_{DD} = 1.8 to 5.0 V	–	± 0.5	± 1	%
		T_A = –40°C to +85°C	–	–	± 3.5	
Internal ring oscillator	–	V_{DD} = 5 V T_A = 25 °C	16.384	32.768	49.152	kHz
External RC oscillator	–	V_{DD} = 5 V	–	4	–	MHz

NOTE:

1. Refer to figure of Operating Voltage Range for more information.
2. Refer to Chapter7 for more information on user calibration of internal RC oscillator.

Table 23-5 Oscillator Characteristics (S3F8S35EZZ, S3F8S39EZZ)

(T_A = –40°C to +85°C, V_{DD} = 1.8 to 5.5 V)

Oscillator	Clock Circuit	Test Condition	Min.	Typ.	Max.	Unit
Main crystal or ceramic		V_{DD} = 2.7 to 5.5 V	0.4	–	12	MHz
		V_{DD} = 1.8 to 2.7 V	0.4	–	4	
External clock (Main System)		V_{DD} = 2.7 to 5.5 V	0.4	–	12	
		V_{DD} = 1.8 to 2.7 V	0.4	–	4	
Tolerance of internal RC oscillator (8/4/2/0.5 MHz)	–	T_A = 25 °C V_{DD} = 1.8 to 5.0 V	–	± 0.5	± 1	%
		T_A = 0°C to +60°C V_{DD} = 3.0 to 3.3 V	–	–	± 2	
Internal ring oscillator	–	V_{DD} = 5 V T_A = 25 °C	16.384	32.768	49.152	kHz
External RC oscillator	–	V_{DD} = 5 V	–	4	–	MHz

NOTE:

1. Refer to figure of Operating Voltage Range for more information.
2. Refer to Chapter7 for more information on user calibration of internal RC oscillator.

Table 23-6 lists the oscillation stabilization time.

Table 23-6 Oscillation Stabilization Time

(T_A = –40°C to +85°C, V_{DD} = 1.8 to 5.5 V)

Oscillator	Test Condition	Min.	Typ.	Max.	Unit
Main crystal	f_{OSC} > 1.0 MHz	–	–	20	ms
Main ceramic	Oscillation stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	10	
External clock (main system)	X_{IN} input high and low width (t_{XH}, t_{XL})	25	–	500	ns
Oscillator stabilization wait time	t_{WAIT} when released by a reset (1)	–	$2^{16}/f_{OSC}$	–	ms
	t_{WAIT} when released by an interrupt (2)	–	–	–	

NOTE:

1. f_{OSC} is the oscillator frequency.
2. The duration of the oscillator stabilization wait time, t_{WAIT}, when it is released by an interrupt is determined by the settings in the basic timer control register, BTCON.

Figure 23-2 illustrates the operating voltage range.

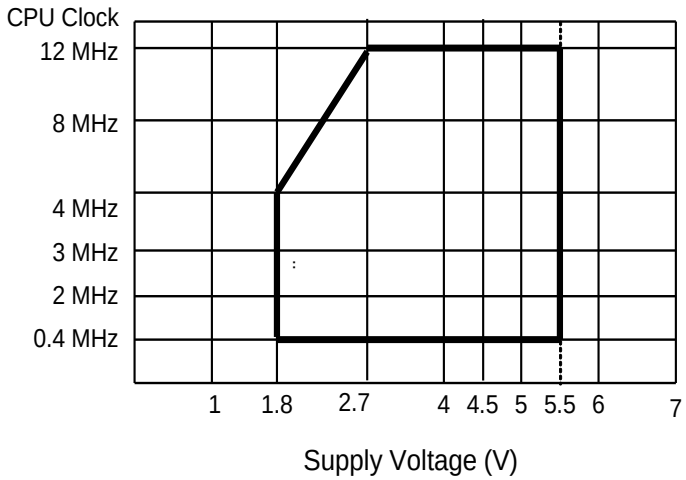


Figure 23-2 Operating Voltage Range

Figure 23-3 illustrates the Schmitt trigger input characteristics diagram.

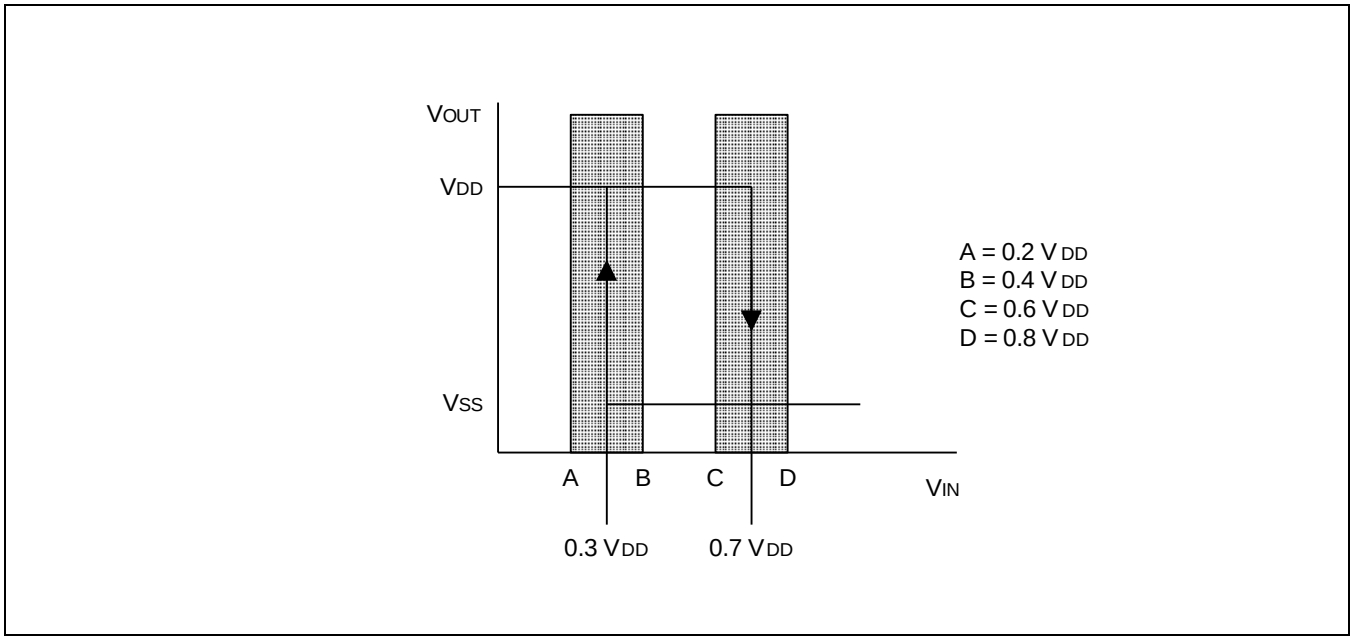


Figure 23-3 Schmitt Trigger Input Characteristics Diagram

Table 23-7 lists the data retention supply voltage in Stop Mode.

Table 23-7 Data Retention Supply Voltage in Stop Mode

(T_A = –40°C to +85°C, V_{DD} = 1.8 to 5.5 V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Data retention supply voltage	V_{DDDR}	Stop Mode	1.0	–	5.5	V
Data retention supply current	I_{DDDR}	Stop Mode; V_{DDDR} = 2.0 V	–	–	1	µA

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

Figure 23-4 illustrates the Stop Mode release timing when initiated by a RESET.

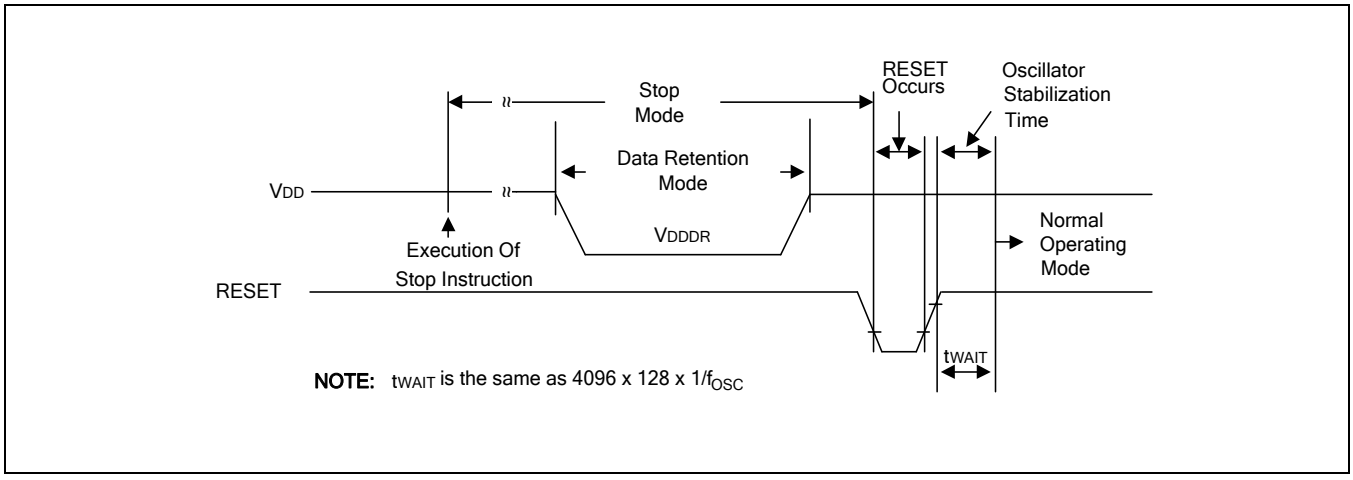


Figure 23-4 Stop Mode Release Timing When Initiated by a RESET

Table 23-8 lists the A/D Converter electrical characteristics.

Table 23-8 A/D Converter Electrical Characteristics

(T_A = –40°C to +85°C, V_{DD} = 2.7 to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Test Conditions	Min.	Typ.	Max.	Unit
Resolution	–	–	–	10	–	bit
Total accuracy	–	V_{DD} = 5.12 V CPU clock = 10 MHz V_{SS} = 0 V	–	–	± 3 (1)	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 (2)	t_{CON}	–	–	20	–	µs
Analog input voltage	V_{IAN}	–	V_{SS}	–	V_{DD}	V
Analog input impedance	R_{AN}	–	2	1000	–	MΩ
Analog input current	I_{ADIN}	V_{DD} = 5 V	–	–	10	µA
Analog block current (3)	I_{ADC}	V_{DD} = 5 V	–	0.5	1.5	–
		V_{DD} = 3 V		0.15	0.45	
		V_{DD} = 5 V power down mode	–	100	500	–

NOTE:

1. The total accuracy is 3LSB (maximum) at V_{DD} = 2.7 –5.5 V, It is for design guidance only and are not tested in production.
2. "Conversion time" is the time that it requires from the moment a conversion operation starts until it ends.
3. I_{ADC} is operating current during A/D conversion.

Table 23-9 lists the UART timing characteristics in Mode 0 (12 MHz).

Table 23-9 UART Timing Characteristics in Mode 0 (12 MHz)

(T_A = –40°C to +85°C, 2.0 to 5.5 V, Load capacitance = 80 pF)

Parameter	Symbol	Min.	Typ.	Max.	Unit
Serial port clock cycle time	t_{SCK}	500	$T0_{PU} \times 6$	700	ns
Output data setup to clock rising edge	t_{S1}	300	$T0_{PU} \times 5$	–	
Clock rising edge to input data valid	t_{S2}	–	–	300	
Output data hold after clock rising edge	t_{H1}	$T0_{PU} – 50$	$T0_{PU}$	–	
Input data hold after clock rising edge	t_{H2}	0	–	–	
Serial port clock High, Low level width	t_{HIGH}, t_{LOW}	200	$T0_{PU} \times 3$	400	

NOTE:
1. All timings are in nanoseconds (ns) and assume a 10 MHz CPU clock frequency.
2. The unit T0PU means one CPU clock period.

Figure 23-5 illustrates the waveform for UART timing characteristics.

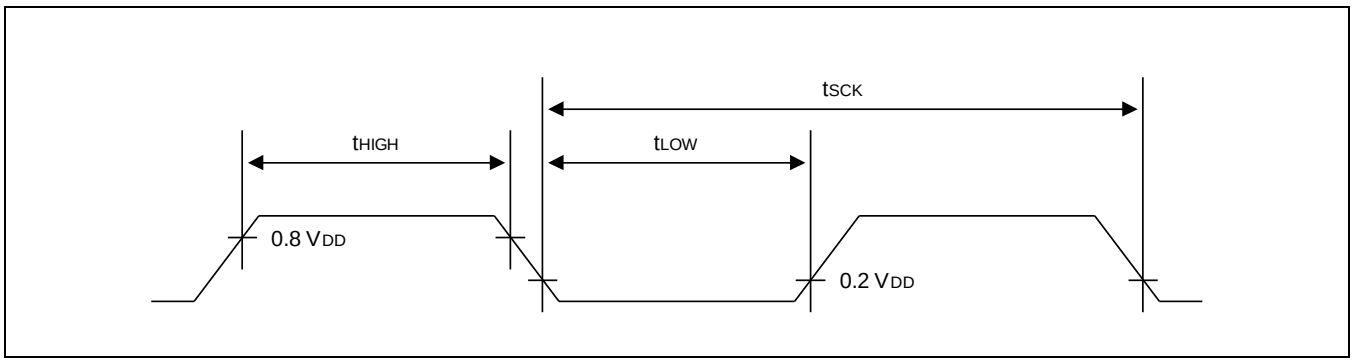


Figure 23-5 Waveform for UART Timing Characteristics

Figure 23-6 illustrates the LVR reset timing.

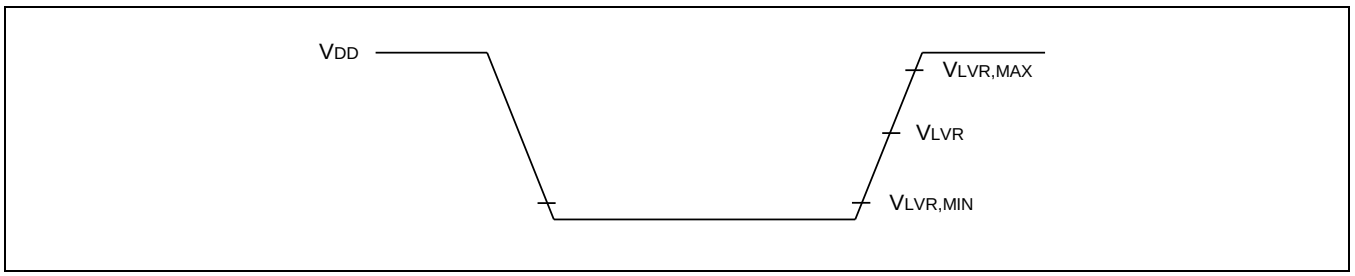


Figure 23-6 LVR Reset Timing

Table 23-10 lists the flash memory AC electrical characteristics.

Table 23-10 Flash Memory AC Electrical Characteristics

(T_A = –40°C to +85°C at V_{DD} = 1.8 to 5.5 V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Flash Erase/Write/Read voltage	Fewrv	V_{DD}	1.8	5.0	5.5	V
Programming time (1)	Ftp	–	20	–	30	µs
Chip erasing time (2)	Ftp1		32	–	70	mS
Sector erasing time (3)	Ftp2		4	–	12	
Data access time	FtRS	V_{DD} = 2.0 V	–	250	–	nS
Number of writing/erasing	FNwe	–	10,000	–	–	Times
Data retention	Ftdr	–	10	–	–	Years

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 entire program memory 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.

Figure 23-7 illustrates the circuit diagram to improve EFT characteristics.

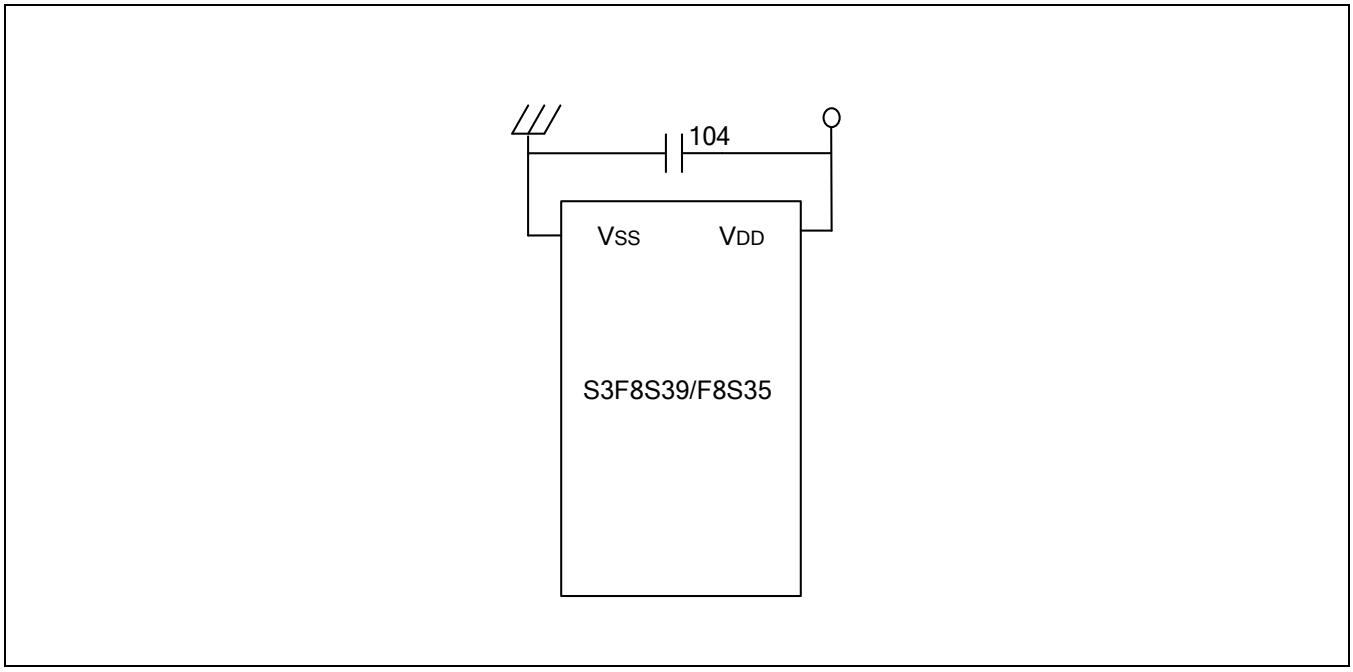


Figure 23-7 The Circuit Diagram to Improve EFT Characteristics

NOTE: To improve EFT characteristics, use power capacitor near S3F8S39/S3F8S35 as illustrated in *Figure 23-7*.

Table 23-11 lists the ESD characteristics.

Table 23-11 ESD Characteristics

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Electrostatic discharge	V_{ESD}	HBM	2000	–	–	V
		MM	200	–	–	
		CDM	500	–	–	

Table 23-12 lists the LVD circuit characteristics.

Table 23-12 LVD Circuit Characteristics

(T_A = –40 to 85 °C, V_{DD} = 1.8 to 5.5 V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
LVD detect voltage	V_{LVD0}	–	4.0	4.1	4.2	V
–	V_{LVD1}		3.1	3.2	3.3	
–	V_{LVD2}		2.4	2.5	2.6	
–	V_{LVD3}		2.0	2.1	2.2	

Table 23-13 lists the LVR circuit characteristics

Table 23-13 LVR Circuit Characteristics

(T_A = –40 to 85 °C, V_{DD} = 1.8 to 5.5 V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
LVR detect voltage	V_{LVR0}	–	3.8	3.9	4.0	V
–	V_{LVR1}		2.9	3.0	3.1	
–	V_{LVR2}		2.2	2.3	2.4	
–	V_{LVR3}		1.8	1.9	2.0	

24 Mechanical Data

24.1 Overview

The S3F8S39/S3F8S35 is available in these packages:

- 32-pin SOP package (Zilog: 32-SOP-450A)
- 32-pin SDIP package (Zilog: 32-SDIP-400)
- 32-pin QFN package (Zilog: 32-QFN-0505)
- 32-pin LQFP package (Zilog: 32-LQFP)

Package dimensions are illustrated in *Figure 24-1*, *Figure 24-2*, *Figure 24-3*, and *Figure 24-4*.

Figure 24-1 shows the package dimensions of 32-pin SOP-450A.

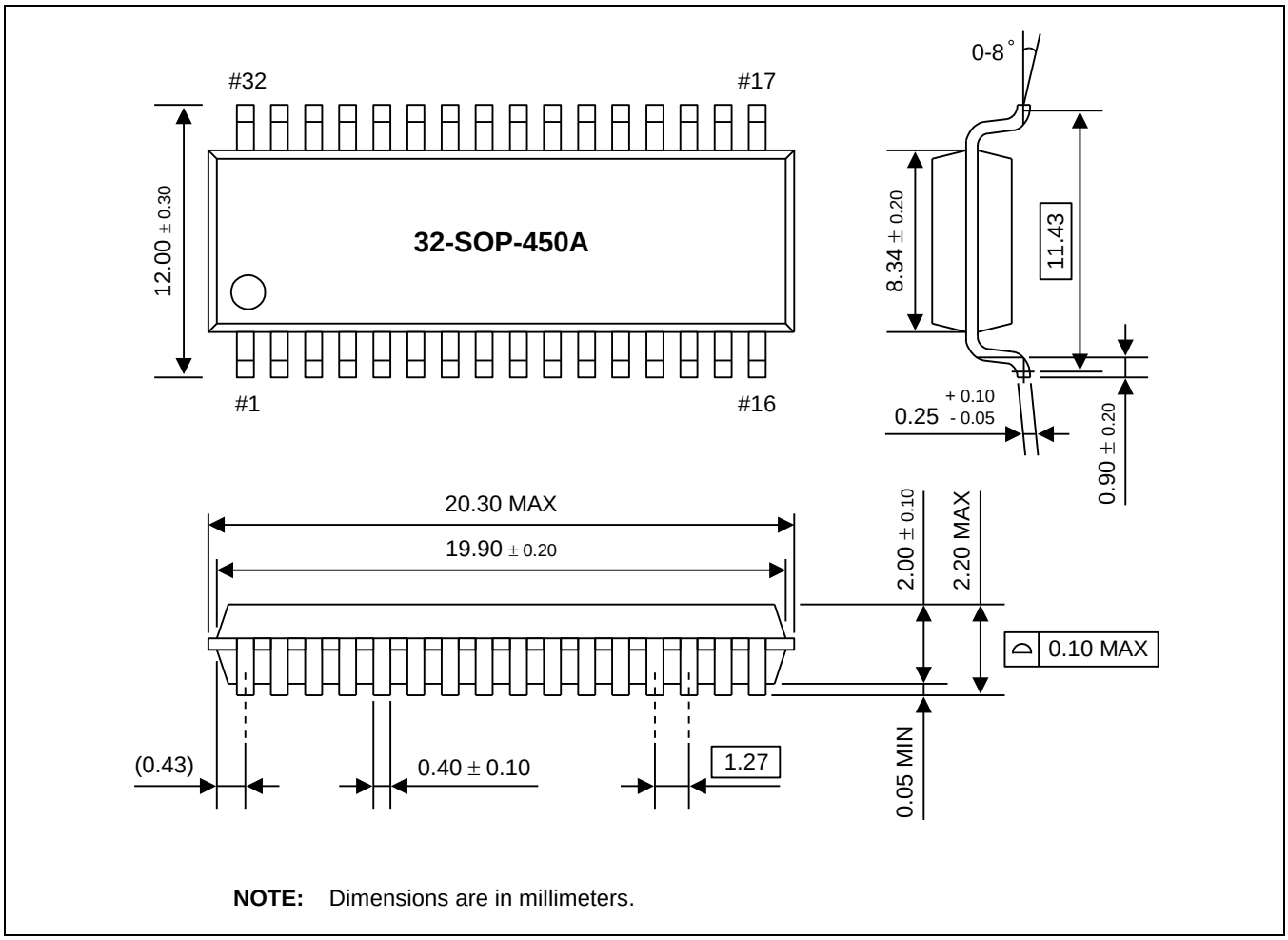
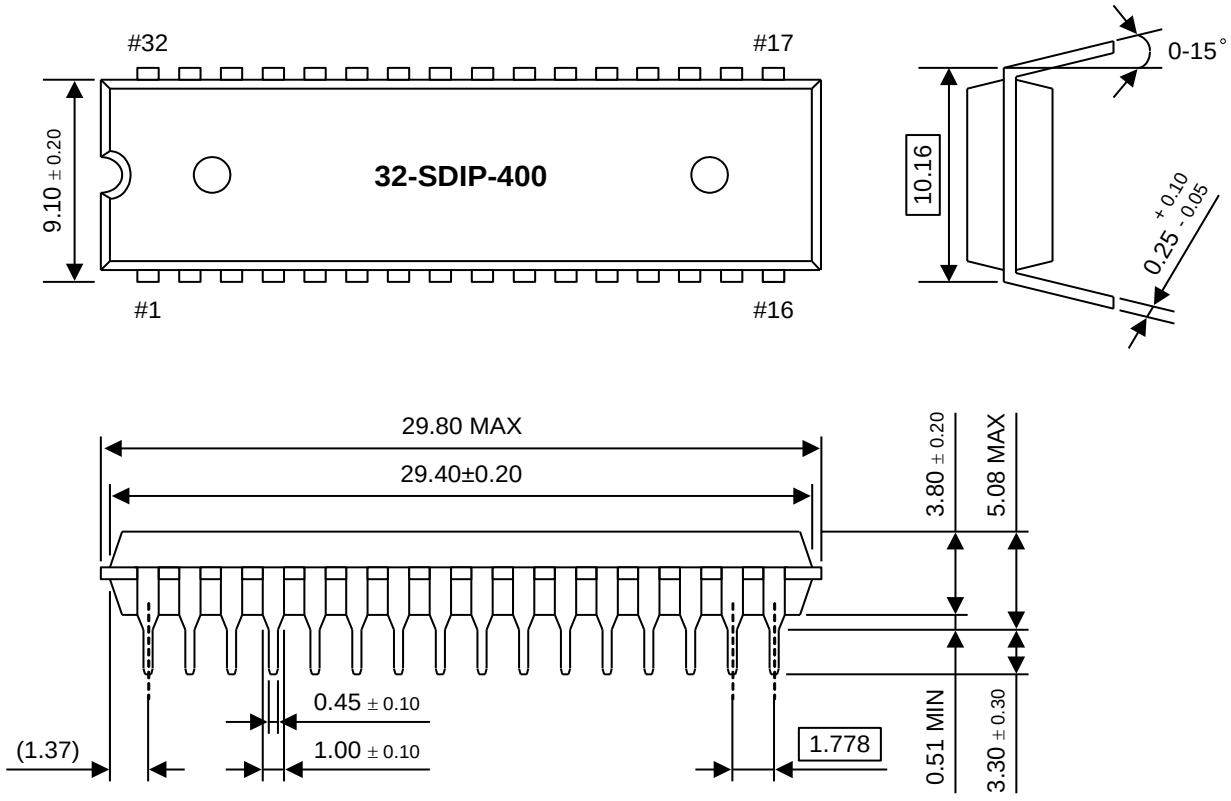


Figure 24-1 32-SOP-450A Package Dimensions

Figure 24-2 illustrates the package dimensions of 32-pin SDIP-400.



NOTE: Dimensions are in millimeters.

Figure 24-2 32-SDIP-400 Package Dimensions

Figure 24-3 illustrates the package dimensions of 32-Pin QFN.

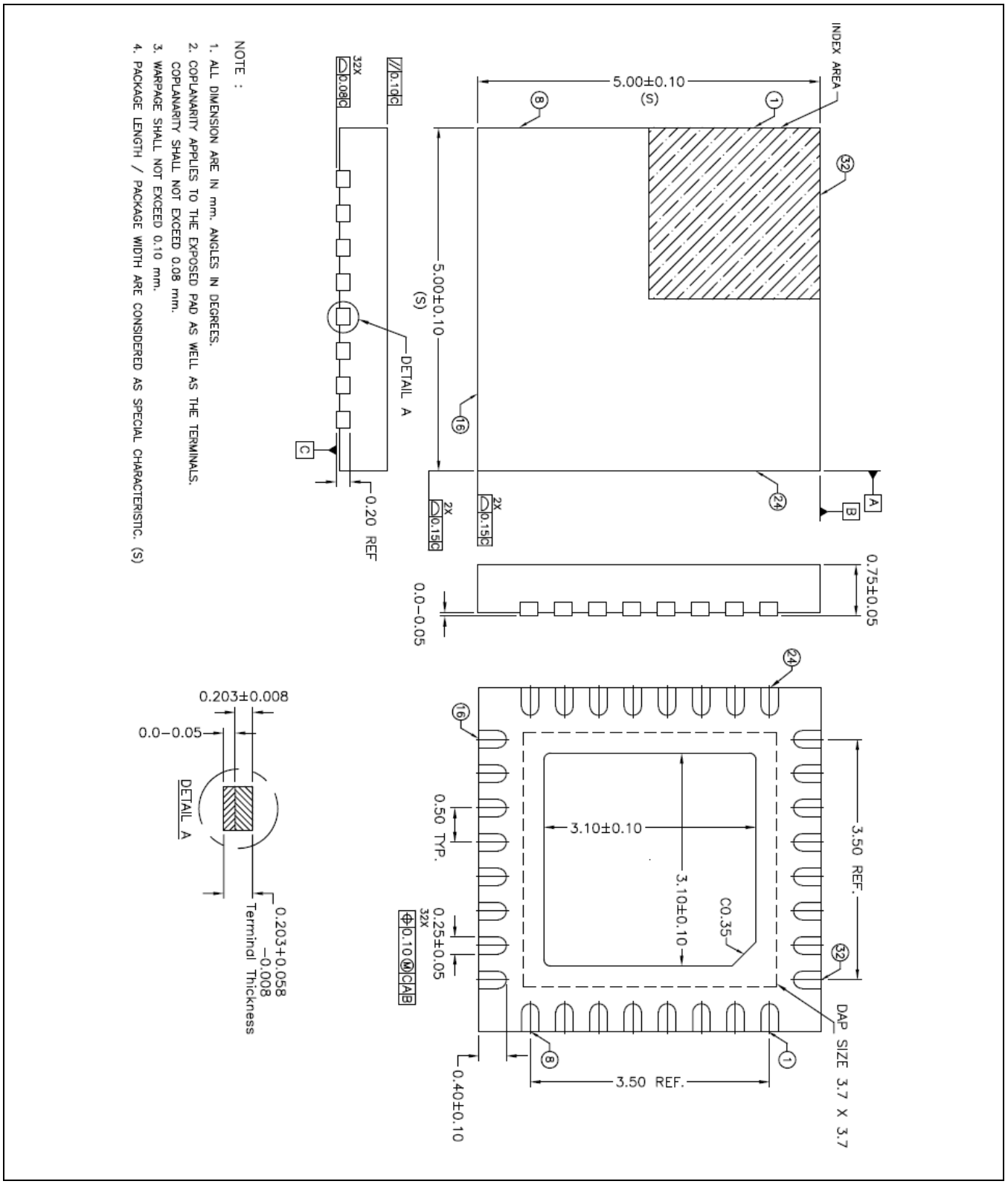


Figure 24-3 32-Pin QFN Package Dimension

Figure 24-4 illustrates the package dimensions of 32-Pin LQFP.

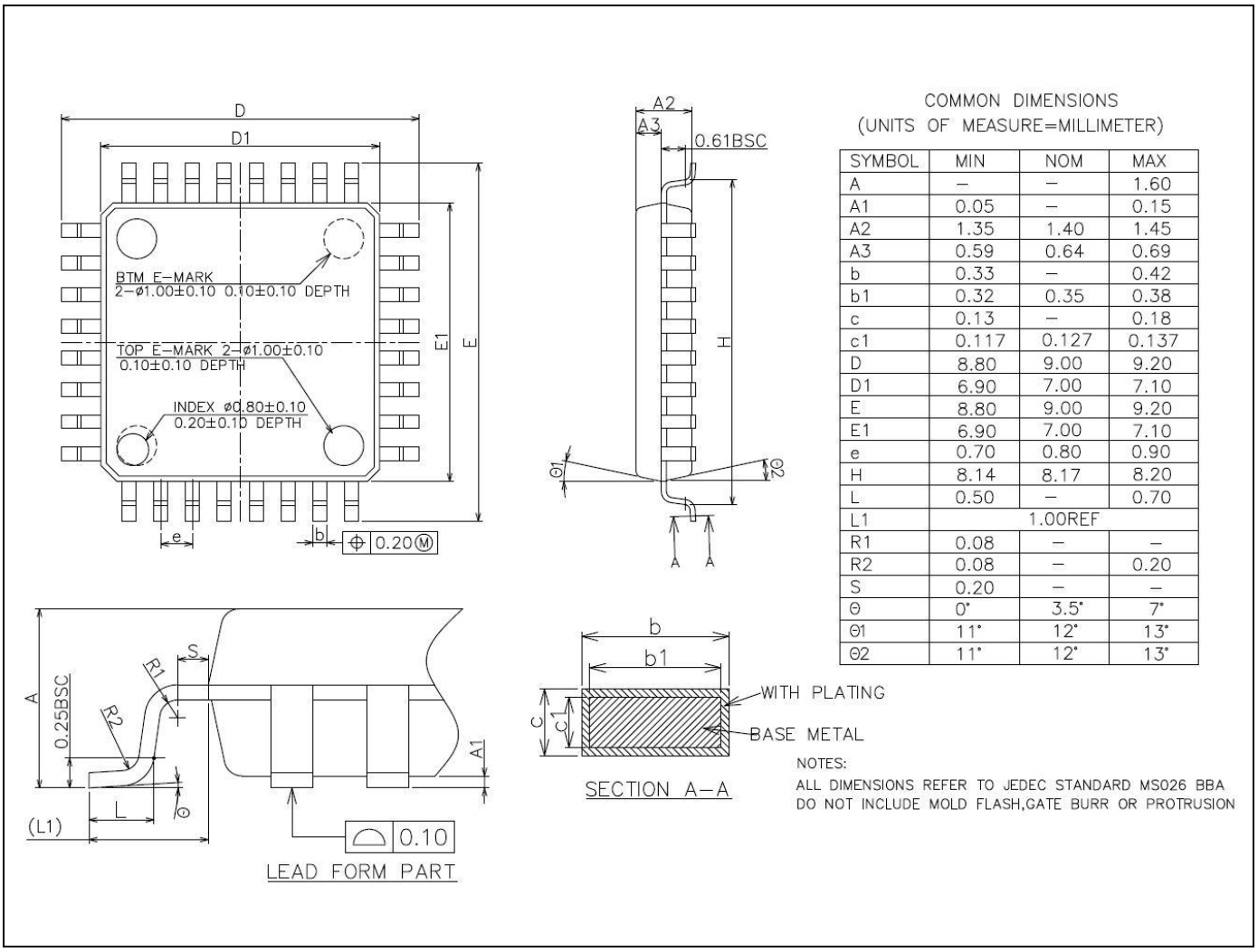


Figure 24-4 32-Pin LQFP Package Dimension

25 S3F8S39/S3F8S35 Flash MCU

25.1 Overview

The S3F8S39/S3F8S35 single-chip microcontrollers feature on-chip Flash ROM, which is accessed via the serial data format.

NOTE: This chapter describes Tool Program Mode. Refer to Chapter 21, “Embedded Flash Memory Interface,” to learn more about User Program Mode.

25.2 Pin Assignments

Figure 25-1 illustrates the S3F8S39/S3F8S35 Pin Assignments (32-pin SOP/32-pin SDIP).

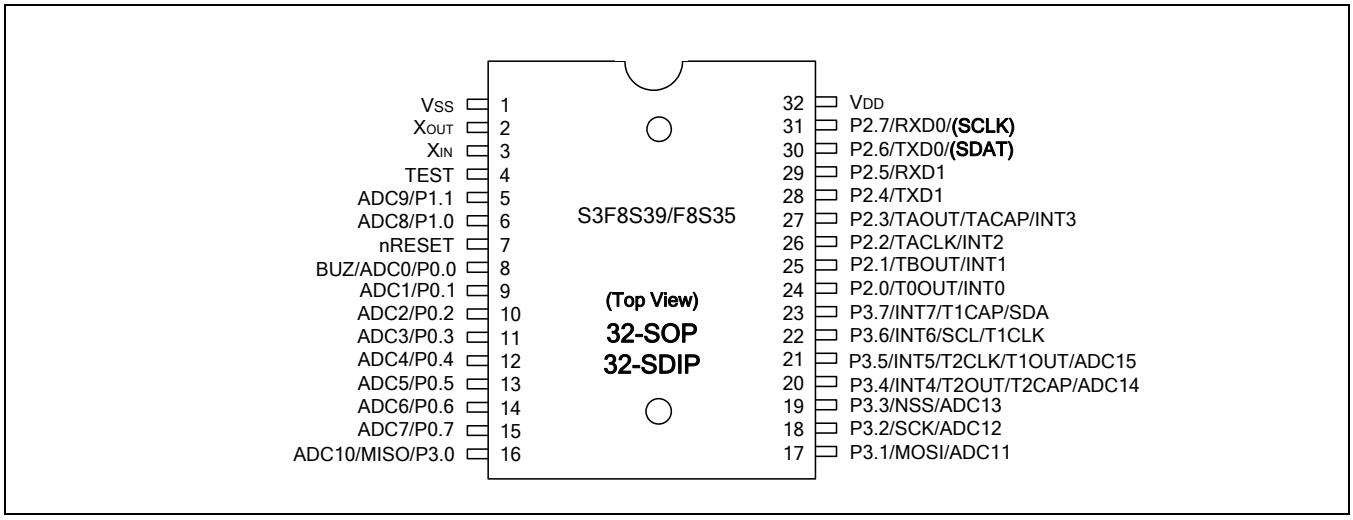


Figure 25-1 S3F8S39/S3F8S35 Pin Assignments (32-SOP/32-SDIP)

Figure 25-2 illustrates the S3F8S39/S3F8S35 Pin Assignments (32-pin QFN/32-pin LQFP).

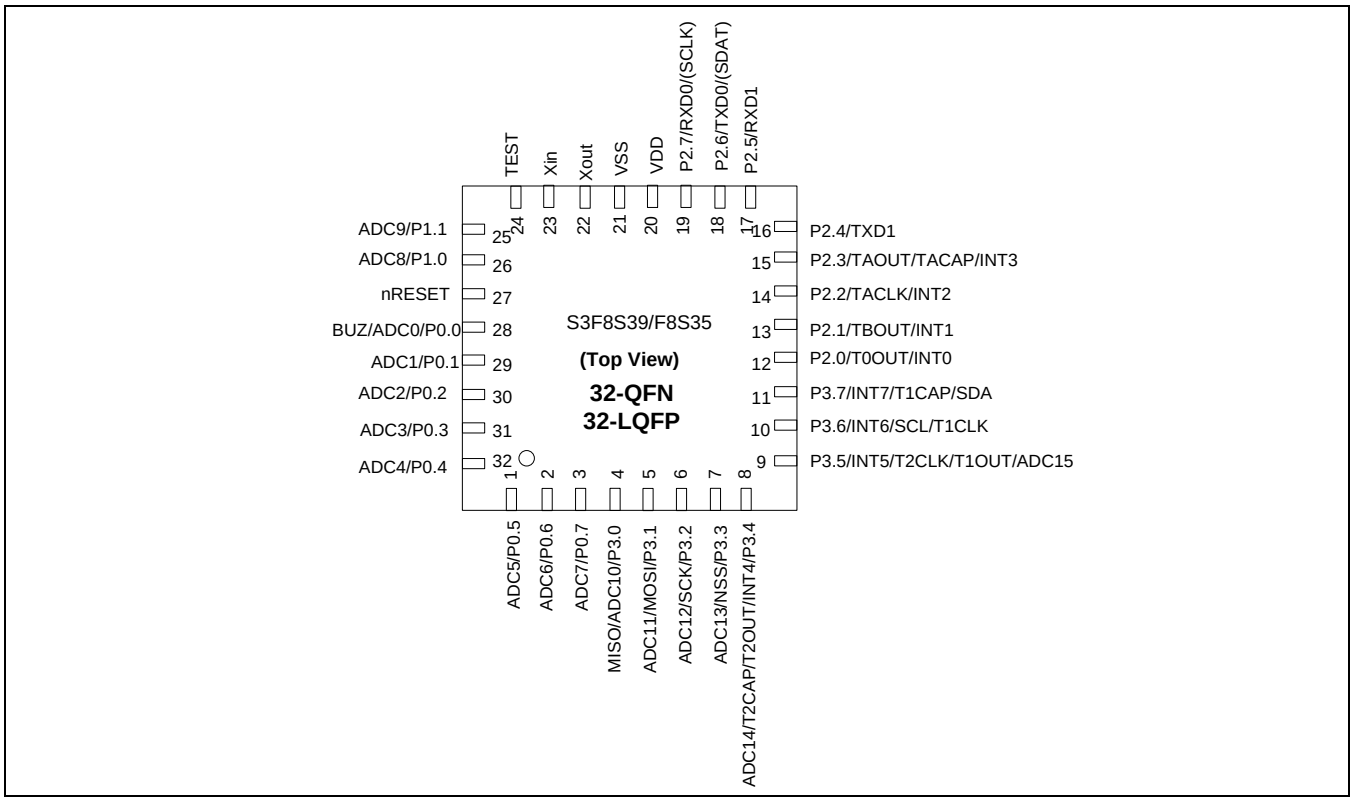


Figure 25-2 S3F8S39/S3F8S35 Pin Assignments (32-QFN/32-LQFP)

Table 25-1 describes the functions of pins used to Read/Write the Flash in Tool Program Mode.

Table 25-1 Descriptions of Pins Used to Read/Write the Flash in Tool Program Mode

Normal Chip	During Programming			
Pin Name	**Pin Name**	**Pin No.**	**I/O**	**Function**
P2.6	SDAT	30 (18)	I/O	Serial data pin. Output port when reading and input port when writing. Can be assigned as an input or push-pull output port.
P2.7	SCLK	31 (19)	I	Serial clock pin. Input only pin.
TEST	V_{PP}	4 (24)	I	Tool mode selection when TEST/Vpp pin sets Logic value "1". When user uses the Flash writer tool mode (ex.spw2+ etc.), user should connect TEST/V_{PP} pin to V_{DD}. (S3F8S39/S3F8S35 supplies high voltage 11 V by internal high voltage generation circuit.)
nRESET	nRESET	7 (27)	I	Chip Initialization
V_{DD}, V_{SS}	V_{DD}, V_{SS}	32 (20), 1 (21)	–	Power supply pin for logic circuit. You should tie V_{DD} to 5.0 V during programming.

NOTE: Parentheses indicate pin number for 32-QFN package.

Test Pin Voltage

The TEST pin on socket board for MTP writer must be connected to V_{DD} (5.0 V) with RC delay as illustrated in Figure 25-3 (only when SPW 2 + and GW-pro2 are used). The TEST pin on socket board must not be connected Vpp (12.5 V) which is generated from MTP Writer. Therefore, when you write or erase using MTP writer, you should use the specific socket board for S3F8S39/S3F8S35.

Figure 25-3 illustrates the RC delay circuit.

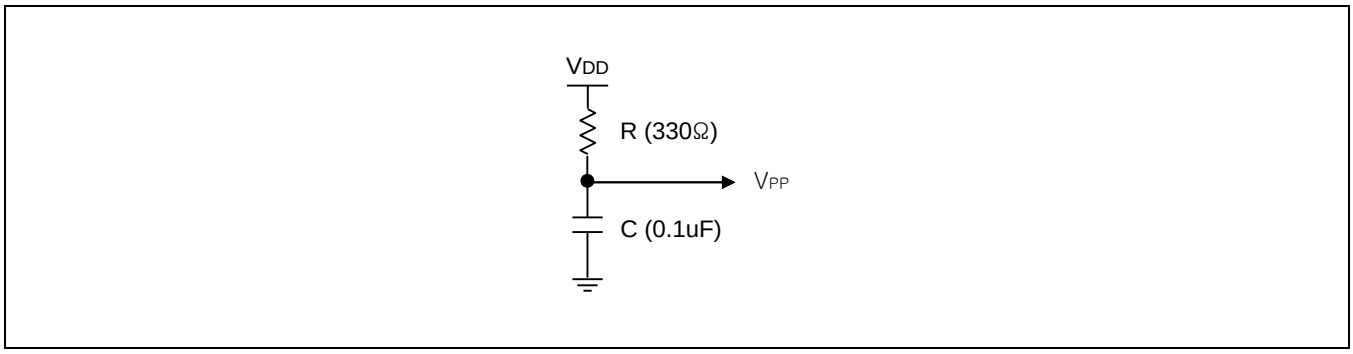


Figure 25-3 RC Delay Circuit

25.3 On Board Writing

The S3F8S39/S3F8S35 requires only six signal lines V_{DD} and V_{SS} 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.

25.3.1 Circuit Design Guide

At the flash writing, the writing tool requires six signal lines that are V_{SS}, V_{DD}, 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 V_{SS} but in writing mode, a resistor should be inserted between the TEST pin and V_{SS}. You should treat nRESET, SDAT, and SCLK under the same consideration.

You should be careful to design the circuit that these signal pins relate because rising/falling timing of V_{PP}, SCLK, and SDAT is very important for proper programming.

Figure 25-4 illustrates the PCB design guide for on-board programming.

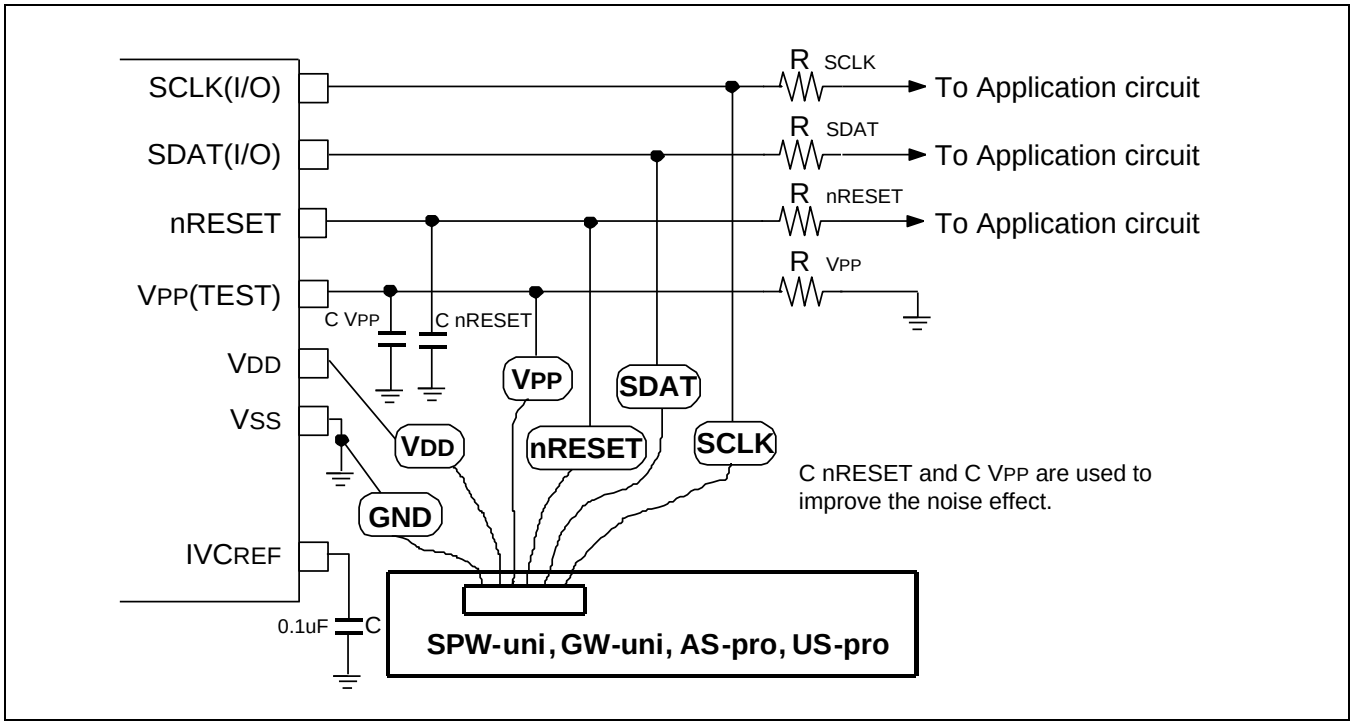


Figure 25-4 PCB Design Guide for On-Board Programming

25.3.2 Reference Table for Connection

Table 25-2 lists the reference table for connection.

Table 25-2 Reference Table for Connection

Pin Name	I/O Mode in Applications	Resistor (Need)	Required Value
V_{PP} (TEST)	Input	Yes	R_{Vpp} is 10 kΩ to 50 kΩ. C_{Vpp} is 0.01 μF to 0.02 μF.
nRESET	Input	Yes	R_{nRESET} is 2 kΩ to 5 kΩ. C_{nRESET} is 0.01 μF to 0.02 μF.
SDAT (I/O)	Input	Yes	R_{SDAT} is 2 kΩ to 5 kΩ.
	Output	No (NOTE)	–
SCLK (I/O)	Input	Yes	R_{SCLK} is 2 kΩ to 5 kΩ.
	Output	No (NOTE)	–

NOTE: In the on-board writing mode, very high-speed signal will be provided to pin SCLK and SDAT. It also causes some damages to the application circuits that are connected to SCLK or SDAT port if the application circuit is designed as high speed response such as relay control circuit. If possible, set the I/O configuration of SDAT and SCLK pins to input mode.
The value of R and C in this table is a recommended value. It varies with circuit of system.

26 Development Tools

26.1 Overview

Zilog offers software and hardware tools for S3 application development. Alternatively, a complete suite of 3rd party tools can be used. Applications targeting S3F8-series microcontrollers can use either the low-cost Zilog library-based Development Platform toolset or more sophisticated 3rd party emulator-based development tools. Applications targeting S3C8-series microcontrollers typically require the use of 3rd party emulator-based development tools.

Section 26.2 describes using 3rd party emulators (such as the OPENice i500 or i2000) to interface with a device-specific target board for application development on S3C8-series (or S3F8-series) microcontrollers. Section 26.3 describes the Zilog library-based Development Platform for Flash-based S3F8-series microcontrollers.

26.2 Emulator-based Development System

Figure 26-1 shows an emulator-based development system utilizing an emulator to interface with an application board through a Zilog-provided Target Board.

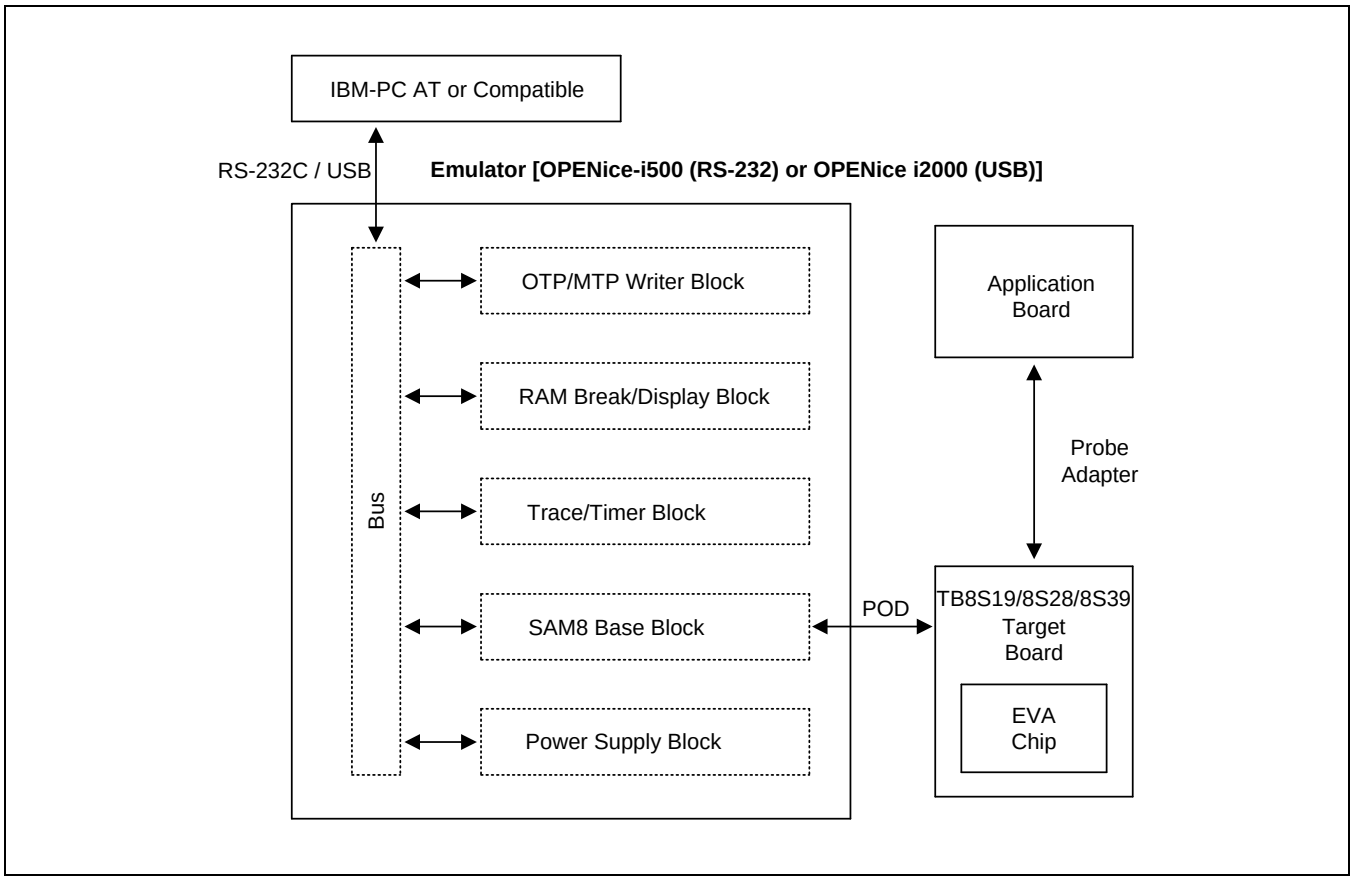


Figure 26-1 Emulator-based Development System Configuration

The S3 Emulator Based Development System includes the components listed in the following sections.

26.2.1 Host Software

Host software is required to create and debug S3 application programs in C or assembly language. The host software program converts the application source code into an executable format that is downloaded into the evaluation (EVA) chip on the target board for program execution/debugging. Optionally, the probe adapter cable(s) can be connected between the target board and the application board to debug program interaction with components on the application board.

Zilog provides the Zilog Developer Studio (ZDS) software suite host software package free of charge for any PC running a supported version of the Windows operating system. Alternatively, 3rd party host software packages (such as the IAR Embedded Workbench host software package) are available for purchase from vendor websites. The ZDS S3 software package is available for free download from the Zilog website.

26.2.2 Target Boards

Target boards are available for all S3C8/S3F8-series microcontrollers. Each target board includes the cables and adapters necessary to interface with an application board. The target board can be used with a 3rd party emulator to enable application debugging with or without an application board. Alternatively, the emulator can be used to program the target MCU on the application board using the supplied 10- circuit programming cable. The TB8S19/8S28/8S39 target board can be used with application boards targeting the S3F8S19, S3F8S28, and S3F8S39 MCUs.

Figure 26-2 shows how the TB8S19/8S28/8S39 Target Board is configured. The symbol " ◀ " marks the starting point of the jumper signals.

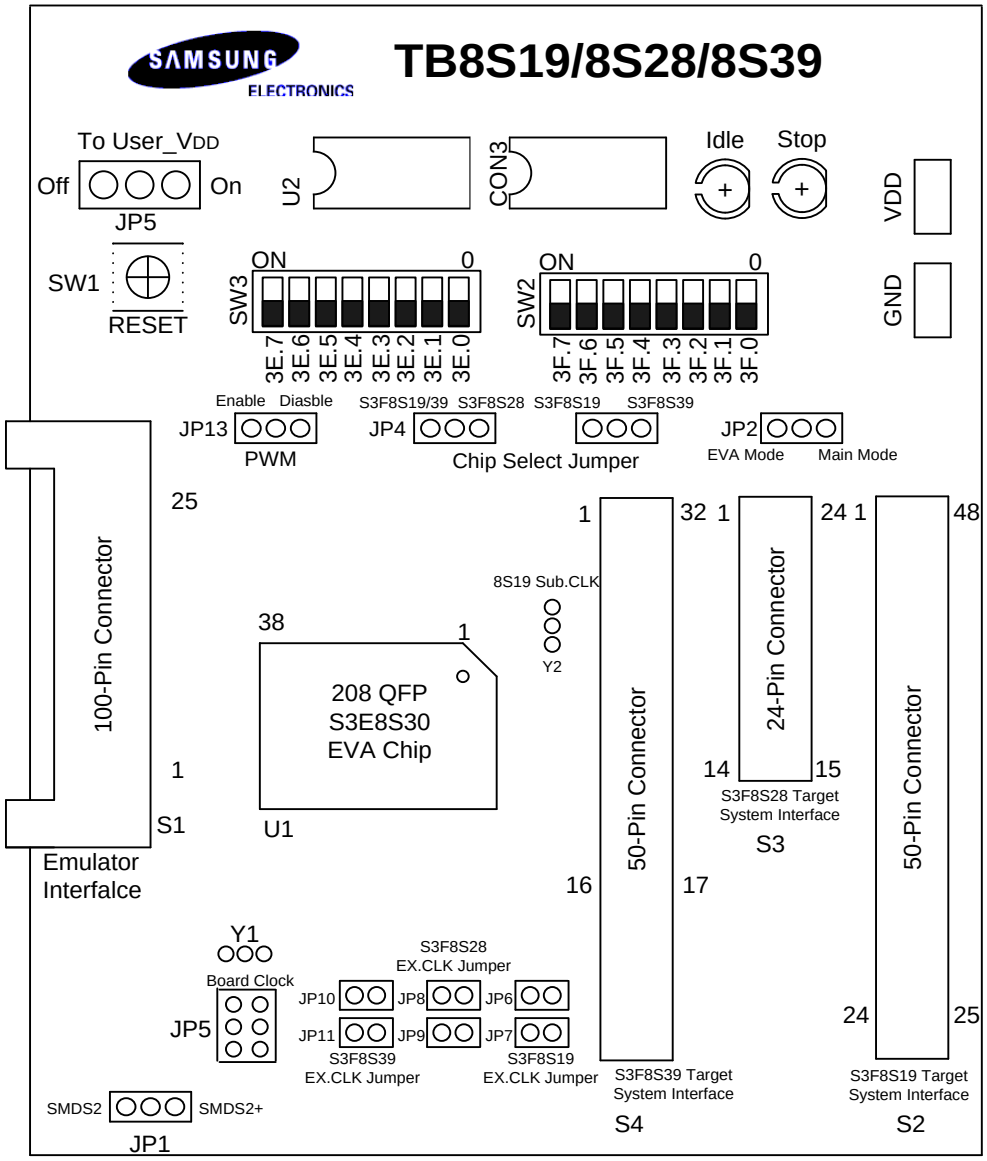


Figure 26-2 TB8S19/8S28/8S39 Target Board Configuration

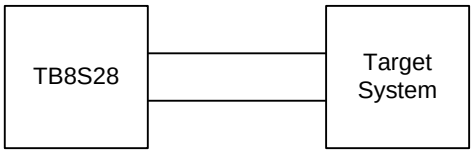
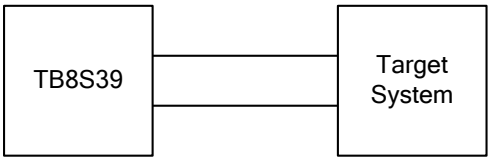
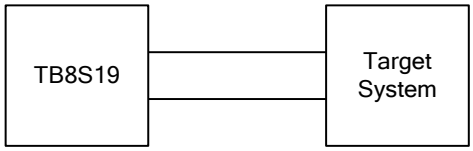
Table 26-1 describes the components of TB8S19/8S28/8S39.

Table 26-1 Components of TB8S19/8S28/8S39

Symbols	Usage	Description
JP3, JP4	Device Selection	Selection of device: S3F8S19, S3F8S28, S3F8S39.
JP10, JP11	Ex.CLK selection	Set external clock connect to S3F8S39/S3F8S35 EVA-chip.
JP12	User's Power selection	Selection of Power to User.
JP2	MODE Selection	Selection of Eva/Main-chip mode of S3F8S39/S3F8S35 EVA-chip.
JP1	Emulator selection	Selection of SMDS2/SMDS2+.
JP5	Clock Source Selection	Selection of debug with internal/external clock
SW2, SW3	8-pin switch	Smart Option setting for S3F8S39/S3F8S35 EVA-chip
S1	100-pin connector	Connection between emulator and TB8S8S39 target board.
S3	50-pin connector	Connection between target board and user application system
RESET	Push button	Generation low active reset signal to S3F8S39/S3F8S35 EVA-chip.
VCC, GND	POWER connector	External power connector for TB8S19/8S28/8S39.
IDLE, STOP LED	STOP/IDLE Display	Indicate the status of STOP or IDLE of S3F8S39/S3F8S35 EVA-chip on TB8S19/8S28/8S39 target board
JP3	PWM selection	Selection of PWM enable/disable.

Table 26-2 lists the device selection settings for TB8S19/8S28/8S39.

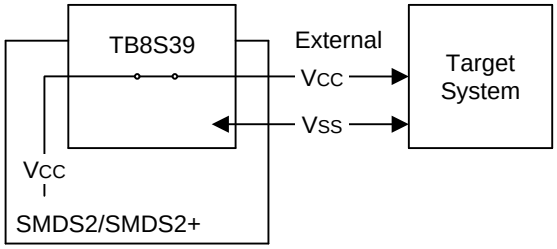
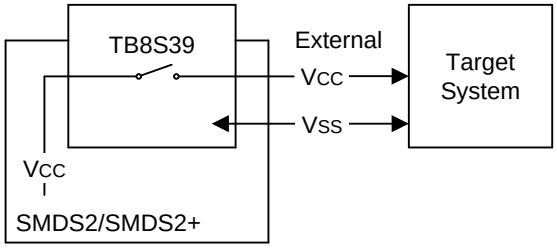
Table 26-2 Device Selection Settings for TB8S19/8S28/8S39

"Device Selection" Settings	Operating Mode	Comments
Device Selection:JP4 8S19/39 ○ ● ● 8S28		Operate with TB8S28
Device Selection JP4 8S19/8S39 ● ● ○ 8S28 JP3 8S19 ○ ● ● 8S39		Operate with TB8S39
Device Selection JP4 8S19/39 ● ● ○ 8S28 JP3 8S19 ● ● ○ 8S39		Operate with TB8S19

NOTE: The following symbol in the "8S28" Setting column indicates the electrical short (off) configuration:

Table 26-3 lists the power selection settings for TB8S19/8S28/8S39.

Table 26-3 Power Selection Settings for TB8S19/8S28/8S39

"To User_Vcc" Settings	Operating Mode	Comments
To user_VDD off on		The SMDS2/SMDS2+ main board supplies V_{DD} to the target board (evaluation chip) and the target system.
To user_VDD off on		The SMDS2/SMDS2+ main board supplies V_{DD} only to the target board (evaluation chip). The target system must have its own power supply.

26.2.2.1 SMDS2+ Selection (SAM8)

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

Table 26-4 describes the SMDS2+ tool selection setting.

Table 26-4 SMDS2+ Tool Selection Setting

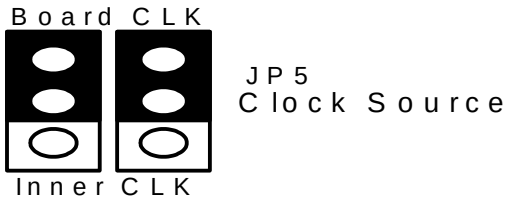
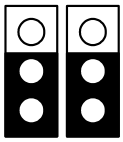
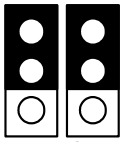
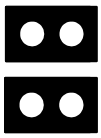
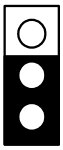
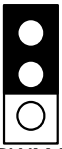
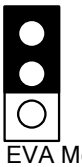
"JP4" Setting	Operating Mode
SMDS2 SMDS2+	

Table 26-5 describes the Using Single Header Pins to Select Clock Source/PWM/Operation Mode.

Table 26-5 Using Single Header Pins to Select Clock Source/PWM/Operation Mode

Target Board Part	Comments
Board CLK  JP5 Clock Source Inner CLK	Use SMDS2/SMDS2+ internal clock source as the system clock. Default Setting
Board CLK  JP5 Clock Source Inner CLK	Use external crystal or ceramic oscillator as the system clock.
 JP10 JP11	Connect Clock to S3F8S39. (JP6-9 off)
PWM Enable  JP13 PWM Disable	PWM function is DISABLED.
PWM Enable  JP13 PWM Disable	PWM function is ENABLED. Default Setting
Main Mode  JP2 EVA Mode	The S3E8S30 run in main mode, just same as S3F8S39/S3F8S35. The debug interface is not available.

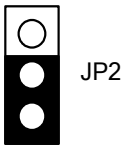
Target Board Part	Comments
Main Mode  JP2 EVA Mode	The S3E8S30 run in EVA mode, available. When debug program, set the jumper in this mode. Default Setting

Table 26-6 describes the using single header pins as the input path for external trigger sources.

Table 26-6 Using Single Header Pins as the Input Path for External Trigger Sources

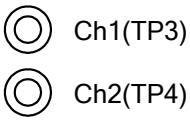
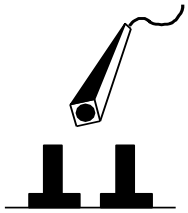
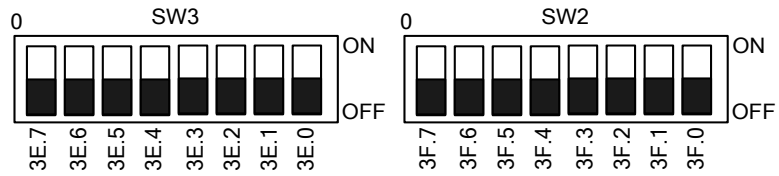
Target Board Part	Comments
External Triggers  Ch1(TP3) Ch2(TP4)	 Connector from External Trigger Sources of the Application System You can connect an external trigger source to one of the two external trigger channels (CH1 or CH2) for the SK-1000/SMDS2+ breakpoint and trace functions.

Figure 26-3 illustrates the DIP switch for Smart Option.



ON	Low
OFF	High (Default)

NOTE:
1. For EVA chip, smart option is determined by DIP switch not software.
2. Please keep the reserved bits as default value (high).

Figure 26-3 DIP Switch for Smart Option

- IDLE LED

  This LED is ON when the evaluation chip (S3E8S30) is in idle mode.

- STOP LED

  This LED is ON when the evaluation chip (S3E8S30) is in Stop Mode.

Figure 26-4 illustrates the 24-Pin connector for TB8S19/8S28/8S39.

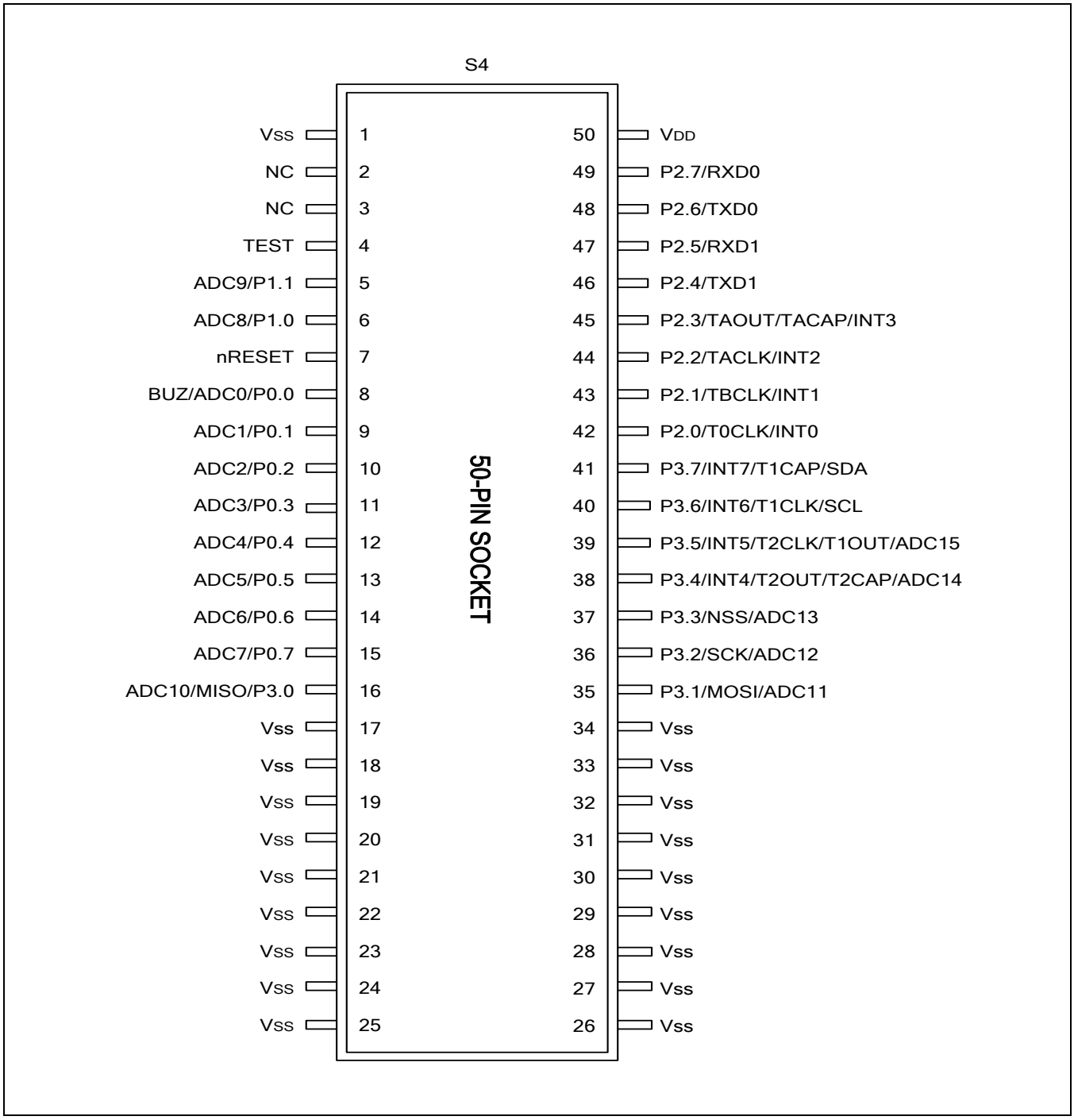


Figure 26-4 50-Pin Connector for TB8S19/8S28/8S39

26.2.3 Optional Probe Adapter Cable(s) and Application Board

The target board can be connected to a customer-designed application board using the optional probe adapter cable(s), as shown in *Figure 26-5*. This allows the EVA chip on the target board to interact with components on the application board while debugging the application.

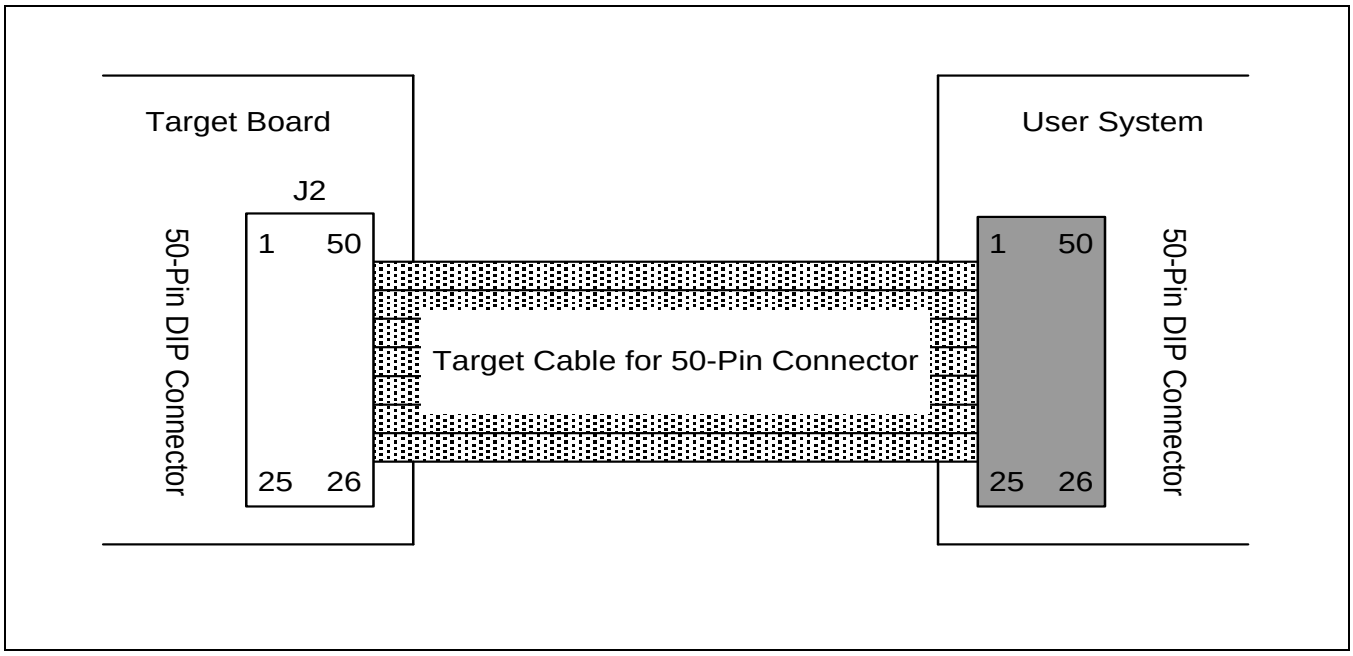


Figure 26-5 TB8S19/8S28/8S39 Probe Adapter Cable and Application Board

26.3 Zilog Library-based Development Platform

The Zilog developer platform is a suite of low-cost highly-integrated software and hardware tools for any PC running a supported version of Windows. The developer platform is composed of three components – the host Integrated Development Environment (IDE) software, the S3 Flash In-System Programmer (ISP) II USB interface, and a development board with a standard 10-pin ISP II connector. Together, these tools cost only a fraction of the price of most other 3rd party compilers, programmers/ emulators, or target boards.

Features include:

- Very low cost development tools
- Easy setup
- Source-level debugging using the application hardware board

26.3.1 Zilog Developer Platform Components

Figure 26-6 shows the simplicity of connecting all of the components of the Zilog developer platform.

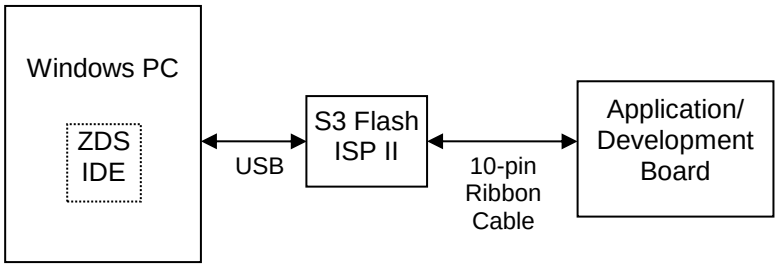


Figure 26-6 Zilog Development Platform

26.3.1.1 ZDS IDE

The Zilog Developer Studio (ZDS) Integrated Development Environment (IDE) is a suite of software tools that run on a Windows-based host PC. These tools include an editor used to create application programs in C or assembly, a compiler, assembler, a linker used to convert the application source code into an executable program image, and a debugger that allows the developer to single-step their application source code while it is executing on the actual target HW platform.

ZDS is completely free of charge and available from the Zilog website. For more information about the features of the ZDS IDE, please refer to the Zilog Developer Studio Help file integrated within the ZDS IDE by clicking the Help Topics item available through the IDE's Help menu, or by pressing F1 on the PC keyboard.

26.3.1.2 S3 Flash ISP II

The Zilog S3 Flash ISP II is a low cost hardware interface between the PC and the application board or Zilog development board. The ISP II connects to the Windows PC through a USB cable and connects to the application or development board through a 10-pin ribbon cable. ZDS uses the ISP II to access Flash memory on the S3 target for read, erase, and program operations. Additionally, ZDS can use the S3 Flash ISP II to debug applications built with a Zilog-provided debug library.

26.3.1.3 Application/Development Board

The S3 Flash ISP II communicates with the S3 microcontroller on a Zilog development board, or a customer application board, through a 10-pin ribbon cable. This requires the application or development board design to include the components shown in *Figure 26-7*.

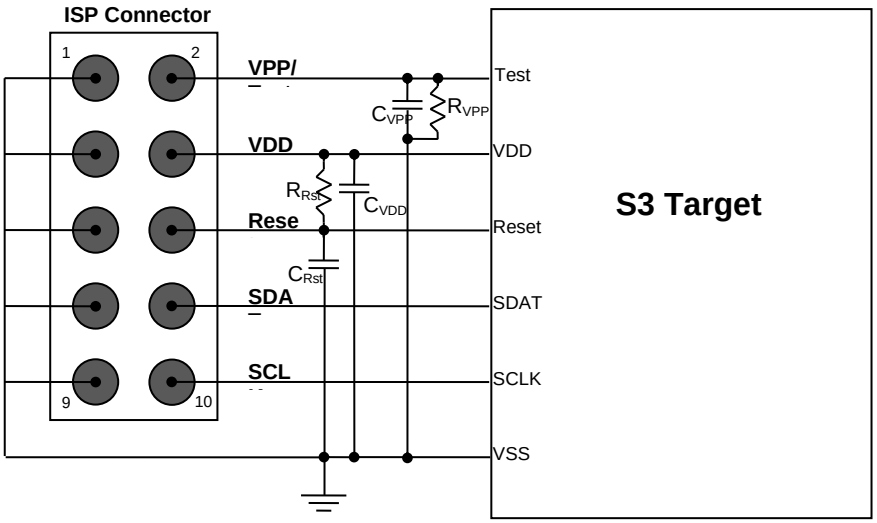


Figure 26-7 PCB Design Guide for In System Programming

Some S3 devices have a VPP/Test pin shared with a GPIO pin which can also be configured as the Reset pin. When designing a PCB that requires In-System Programming support for S3 devices with a shared VPP/ Reset pin, do not connect the Reset signal (pin 6) from the 10-pin ISP II connector to the S3 MCU. Instead, connect the MCU VPP/ Reset pin to the Test/ VPP signal (pin 2) of the ISP II connector with R_{RST} and C_{RST}. In this instance, it is not necessary to include R_{VPP} or C_{VPP}.

Table 26-7 shows the recommended values for the passive components in the ISP II circuit of Figure 26.6.

Table 26-7 ISP II Circuit Recommended Values

ISP Signal (Pin Number)	Passive Component	Notes
VPP/ Test (2)	C_{VPP} = 0.1 uF R_{VPP} = 10K	If the S3 MCU has a shared VPP/Reset pin, connect the ISP II VPP/ Test pin to the MCU VPP/Test pin.
VDD (4)	C_{VDD} = 0.1 uF	
Reset (6)	C_{RST} = 0.1 uF R_{RST} = 40K	
SDAT (8) SCLK (10)		The ZDS IDE and S3 Flash ISP II cannot be used to debug applications that use the GPIO pins associated with the SCLK & SDAT signals. In this instance, it is only possible to access Flash Memory in the target S3 MCU.
GND (1,3,5,7,9)		Connect all odd number pins of the ISP connector to GND on the target board and S3 MCU

Refer to the schematic diagram in the appropriate Zilog Development Kit User Manual for a complete reference design that includes an ISP II interface circuit applicable to a particular series of S3 devices. Zilog recommends keeping the traces connecting SCLK and SDAT to the ISP II connector as short as possible.

26.3.2 Compatibility with 3rd Party Tools

The Zilog IDE can also be used with 3rd party development tools. For example, the ZDS IDE can program a Hex file generated by a 3rd party compiler such as the IAR Embedded Workbench using the Zilog S3 Flash ISP II or a 3rd party programmer such as the OPENice-i2000 emulator. Information regarding 3rd party development tools can be found in section 26.4.

26.3.3 Benefits and Limitations of Zilog Development Tools

Zilog development tools provide a low cost turnkey solution capable of creating and debugging S3 applications on Zilog development boards or customer application boards. Debugging applications on a particular S3 target typically requires the application to be built with a Zilog-provided debug library that is capable of interfacing with the S3 Flash ISP II. The debug library consumes some amount of code space on the S3 target depending on the set of debugging features supported by the particular debug library linked to the application.

The ZDS IDE and S3 Flash ISP II can be used to program Flash memory on all Zilog S3 microcontrollers; however, single-step debugging support may not be available for every series of Zilog S3 microcontroller. For more information regarding the debugging features available on a particular S3 microcontroller, refer to the S3 ISP II Interface Debug Library chapter of the Zilog Developer Studio Help file available within the ZDS S3 IDE.

26.4 Development Tools

Zilog, in conjunction with third parties, 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.

In-Circuit emulator:

- YIC - OPENice-i500/2000

OPENice-i500 	YIC System • TEL: 82-31-278-0461 • FAX: 82-31-278-0463 • E-mail: support@yicsystem.com • URL: *http://www.yicsystem.com*
OPENice-i2000 	YIC System • TEL: 82-31-278-0461 • FAX: 82-31-278-0463 • E-mail : support@yicsystem.com • URL: *http://www.yicsystem.com*

Zilog Library based Development Tools:

- Zilog – S3USBISP000ZACG S3 Flash ISP II

S3USBISP000ZACG 	Zilog • TEL: (408) 457-9000 • FAX: (408) 416-0223 • E-mail:s3sales@zilog.com • URL: *http://www.zilog.com*

Programmers (Writer):

- Seminix - GW-uni2
- C&A Tech – GW-Pro2
- Elnec – BeeHive series
- Zilog – S3 Flash ISP II

	GW-uni2 **Gang Programmer for OTP/MTP/FLASH MCU** • Support all SAMSUNG OTP and MTP devices with SAMSUNG standard serial protocol format • Program up to 8 devices at one time • Operation mode: 1.PC base 2.Stand-alone (no PC) • Very fast programming speed: OTP(2 Kbps) MTP(10 Kbps) • Maximum buffer memory:100 Mbyte • Hex data file download via USB port from PC • Support simple GUI (Graphical User Interface) • Support data format: Intel hex, SAMSUNG hex, Binary • Device information can be set by a device part number • LCD Display (Stand-alone mode operation) - Display an operation state • Touch key (Stand-alone mode operation) • System upgradeable - The system firmware can be upgraded simply by user	Seminix • TEL: 82-31-703-7891 • FAX: 82-31-702-7869 • E-mail: sales@seminix.com • URL: *http://www.seminix.com*
	GW-Pro Gang Programmer • Programming of 8 MCUs at a time • Fast programming speed (2 Kbyte/sec) • Possible without PC (standalone) • Search operation based on a PC • Enough features to support Gang Programmer • Off data is also preserved • Key Lock function to prevent malfunction • Good and bad quantity counter • Program completion notification (sound) • Easy-to-use (PC) menu	C & A Technology • TEL: 02-2612-9027 • E-mail : jhc115@cnatech.com • URL: *http://www.cnatech.com*

	Beehive204 • Four independent universal programming sites • Two BeeHive 204 multiprogrammers can be attached to one PC to better utilize programming workplace • Extremely fast programming, one of the fastest programmers in this category. Sustainable programming speed greater than 5 Mbytes per second • Powerful independent pin driver circuit for each and every pin of the programmer • In-circuit programming capability through ISP connector • Very low voltage support for the latest Flash memory chips • ESD protection on each pin of the socket's USB (up to 480 Mbit/s) interface to PC • Comfortable and easy-to-use control program; works with all versions of MS Windows from Windows XP to Windows 10 (32-bit and 64-bit)	Elnec • TEL: +421-51-7734328 • FAX: +421-51-7732797 • E-mail:tech2@elnec.com • URL: *http://www.elnec.com*
	S3 Flash In-System Programmer II Zilog's S3 Flash ISP II provides an interface between any development or application board with an S3 microcontroller device to the high-speed USB port of a PC on which Zilog Developer Studio II for S3 Family devices (ZDS II – S3) is installed. The ISP II allows the Flash memory space on any S3 Family device to be programmed, and also offers limited debugging capabilities when used together with the Zilog Debug Library. The following features are available with the S3 Flash ISP II when using ZDS II for S3 Family devices: • Download code to Flash and begin to program execution • Break program execution arbitrarily • Single-step debug the application, view/edit memory and S3 special function registers. Resume normal program operation after a breakpoint • Insert multiple breakpoints in a program at compile/assembly time	Zilog • TEL: (408) 457-9000 • FAX: (408) 416-0223 • E-mail:s3sales@zilog.com • URL: *http://www.zilog.com*

To obtain the S3 Family development tools that will satisfy your S3F8S39/S3F8S35 MCU 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.

27 Ordering Information

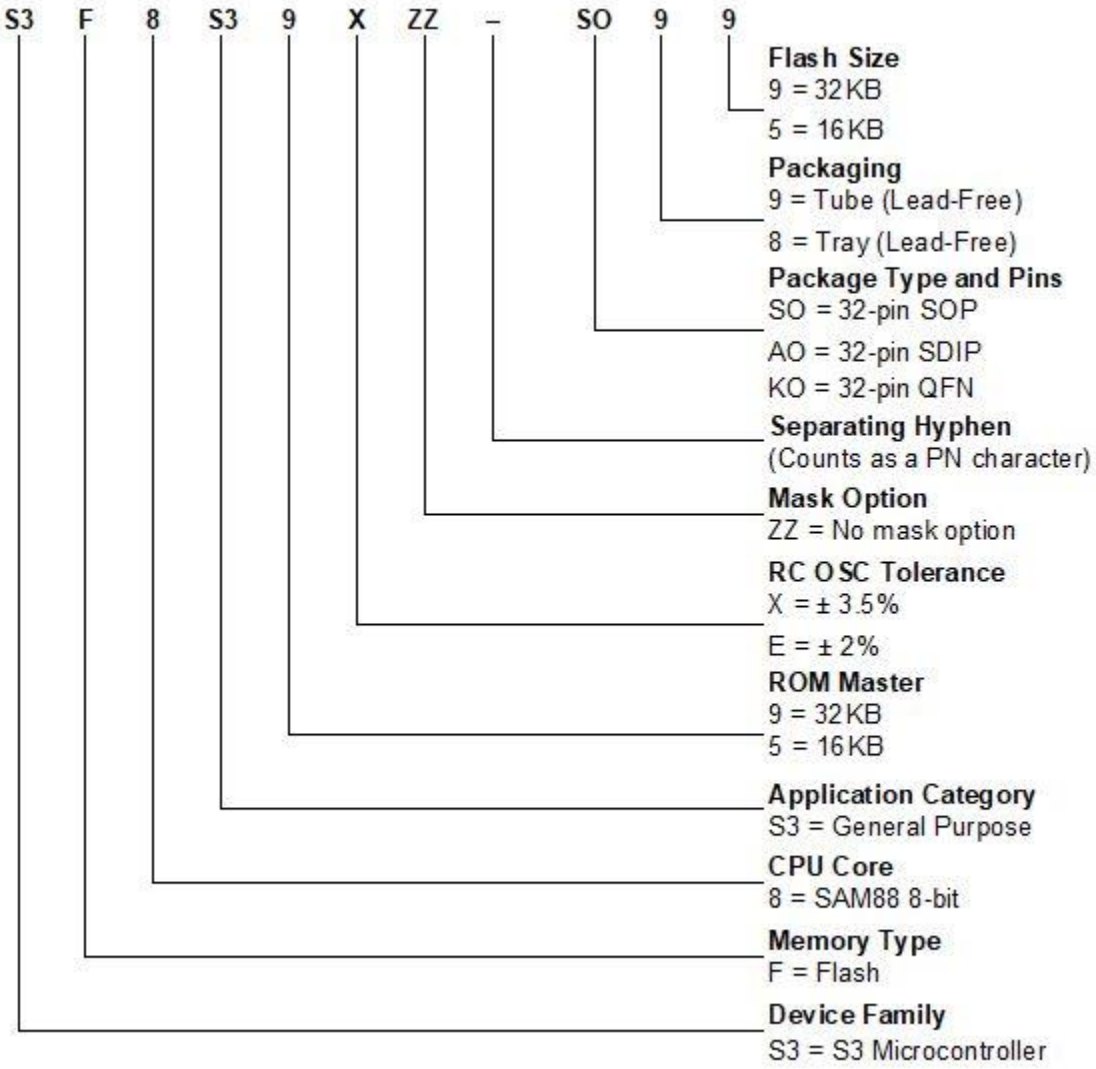
Table 27-1 identifies the basic features and package styles available for the S3F8S39/S3F8S35 MCU.

Table 27-1 Ordering Information for the S3F8S39/S3F8S35 MCU

Device	Flash Size	RAM Size	GPIO	Bit-Programmable Ports	RC OSC Tolerance, Max (Note 1)	Package
S3F8S35XZZ-SO95	16 KB	1040 B	26	4	±3.5%	32-pin SOP
S3F8S35XZZ-AO95	16 KB	1040 B	26	4	±3.5%	32-pin SDIP
S3F8S35XZZ-KO85	16 KB	1040 B	26	4	±3.5%	32-pin QFN
S3F8S35XZZ-EO85	16 KB	1040 B	26	4	±3.5%	32-pin LQFP
S3F8S35EZZ-EO85	16 KB	1040 B	26	4	±2%	32-pin LQFP
S3F8S39XZZ-SO99	32 KB	1040 B	26	4	±3.5%	32-pin SOP
S3F8S39XZZ-AO99	32 KB	1040 B	26	4	±3.5%	32-pin SDIP
S3F8S39XZZ-KO89	32 KB	1040 B	26	4	±3.5%	32-pin QFN
S3F8S39XZZ-EO89	32 KB	1040 B	26	4	±3.5%	32-pin LQFP
S3F8S39EZZ-EO89	32 KB	1040 B	26	4	±2%	32-pin LQFP

27.1 Part Number Suffix Designations

Zilog part numbers consist of a number of components. For example, part number S3F8S39XZZ-SO99 is an unmasked 8-bit general purpose MCU with 32 KB of Flash memory in a 32-pin SOP package.


S3 F 8 S3 9 X ZZ – SO 9 9
Flash Size
9 = 32 KB
5 = 16 KB
Packaging
9 = Tube (Lead-Free)
8 = Tray (Lead-Free)
Package Type and Pins
SO = 32-pin SOP
AO = 32-pin SDIP
KO = 32-pin QFN
Separating Hyphen
(Counts as a PN character)
Mask Option
ZZ = No mask option
RC OSC Tolerance
X = ± 3.5%
E = ± 2%
ROM Master
9 = 32 KB
5 = 16 KB
Application Category
S3 = General Purpose
CPU Core
8 = SAM88 8-bit
Memory Type
F = Flash
Device Family
S3 = S3 Microcontroller

Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View S3F8S39XZZ-LO89 on WIN SOURCE
- Zilog Information

Optimize Your Supply Chain with WIN SOURCE Solutions

- Global Sourcing Solution
- Obsolete Management
- Cost Control Management
- Shortage Management
- Alternative Solution
- Excess Inventory Management