



S3 Family 8-Bit Microcontrollers

S3F80PB MCU

Product Specification

PS032104-0417





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

©2017 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concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded.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
Apr 2017	04	Added Zilog Library-based Development Platform and updated to most current 3rd party tools; Changed package type description from ELP to QFN.	CH 20 Multiple
Aug 2015	03	Replaced Figure 4. Pin Assignments, 32-Pin ELP Package with updated pin-out, updated Table 1. Pin Descriptions, 32-Pin SOP/ELP Packages.	7 , 8
Dec 2014	02	Added pin assignments for the 32-pin ELP and 44-pin ELP packages (Figures 4 and 5); modified Tables 1 and 2 to include pin descriptions for the 32-pin and 44-pin ELP packages; modified pin circuit diagrams, Figures 6 through 11; modified Noncontiguous 16-Byte Working Register Block (Figure 19); modified Op Code Quick Reference (changed I _{rr} to I _r in D2; CPIJNE, Table 26); corrected Figure 50 to imply hex values; modified D.C. Electrical Characteristics (Table 94); deleted Slew Rate of LVD data from Low Voltage Detect Circuit (Table 96); updated to Zilog style.	7 , 10 , 13–18 , 29 , 91 , 151 , 300 , 302
Mar 2014	01	Original Zilog issue.	All

Table of Contents

Revision History	iii
Table of Contents	iv
List of Figures	xi
List of Tables	xiv
Chapter 1. Overview	1
1.1. S3F80PB Microcontroller	1
1.2. Features	1
1.3. Block Diagram	4
1.4. Pin Assignments	6
1.5. Pin Circuits	13
Chapter 2. Address Space	19
2.1. Program Memory	19
2.1.1. Smart Option	21
2.2. Register Architecture	23
2.2.1. Register Page Pointer	25
2.2.2. Register Set1	26
2.2.3. Register Set2	27
2.2.4. Prime Register Space	27
2.2.5. Working Registers	28
2.2.6. Using the Register Pointers	29
2.3. Register Addressing	31
2.3.1. Common Working Register Area	33
2.3.2. 4-Bit Working Register Addressing	34
2.3.3. 8-Bit Working Register Addressing	36
2.4. Register Pointer Registers	39
2.5. System and User Stacks	40
2.5.1. Stack Operations	40
2.5.2. User-Defined Stacks	40
2.5.3. Stack Pointers	41
2.6. Stack Pointer Register	42
Chapter 3. Addressing Modes	43
3.1. Register Addressing Mode	44
3.2. Indirect Register Addressing Mode	45
3.3. Indexed Addressing Mode	48

3.4.	Direct Address Mode	52
3.5.	Indirect Address Mode	53
3.6.	Relative Address Mode	54
3.7.	Immediate Mode	56
Chapter 4.	Control Registers	57
4.1.	External Memory Timing Register	60
Chapter 5.	Interrupt Structure	61
5.1.	Levels	61
5.2.	Vectors	61
5.3.	Sources	61
5.4.	Interrupt Types	62
5.5.	Interrupt Vector Addresses	65
5.6.	Enable/Disable Interrupt Instructions	67
5.7.	Interrupt Processing Control Points	68
5.8.	Peripheral Interrupt Control Registers	70
5.9.	System Mode Register	71
5.10.	Interrupt Mask Register	72
5.11.	Interrupt Priority Register	74
5.12.	Interrupt Request Register	76
5.13.	Interrupt Pending Function Types	77
5.14.	Interrupt Source Polling Sequence	78
5.15.	Interrupt Service Routines	78
5.16.	Generating Interrupt Vector Addresses	79
5.17.	Nesting of Vectored Interrupts	79
5.18.	Instruction Pointer	80
5.19.	Fast Interrupt Processing	81
5.19.1.	Procedure for Initiating Fast Interrupts	81
5.19.2.	Fast Interrupt Service Routine	81
5.19.3.	Programming Guidelines	82
Chapter 6.	CPU Instructions	83
6.1.	Data Types	83
6.2.	Register Addressing	83
6.3.	Addressing Modes	83
6.4.	Instruction Summary	84
6.5.	Flags Register	86
6.6.	Instruction Set Notation	89

Chapter 7. Op Code Maps	91
7.1. Condition Codes	93
7.2. Instruction Set	94
Add with Carry	95
Add	97
Logical AND	99
BAND-Bit AND	101
Bit Compare	102
Bit Complement	103
Bit Reset	104
Bit Set	105
Bit OR	106
Bit Test, Jump Relative on False	108
Bit Test, Jump Relative on True	109
Bit XOR	110
Call Procedure	111
Complement Carry Flag	113
Clear	114
Complement	115
Compare	116
Compare, Increment, and Jump on Equal	118
Compare, Increment, and Jump on NonEqual	119
Decimal Adjust	120
Decrement	122
Decrement Word	123
Disable Interrupts	124
Divide (Unsigned)	125
Decrement and Jump if NonZero	127
Enable Interrupts	128
Enter	129
Exit	130
Idle Operation	131
Increment	132
Increment Word	133
Interrupt Return	135
Jump	137
Jump Relative	138
Load	139
Load Bit	141
Load Memory	142
Load Memory and Decrement	145

Load Memory and Increment	146
Load Memory with PreDecrement	147
Load Memory with PreIncrement	148
Load Word	149
Multiply (Unsigned)	150
Next	151
No Operation	152
Logical OR	153
Pop from Stack	155
Pop User Stack (Decrementing)	156
Pop User Stack (Incrementing)	157
Push to Stack	158
Push User Stack (Decrementing)	159
Push User Stack (Incrementing)	160
Reset Carry Flag	161
Return	162
Rotate Left	163
Rotate Left through Carry	165
Rotate Right	167
Rotate Right through Carry	169
Select Bank0	171
Select Bank1	172
Subtract with Carry	173
Set Carry Flag	175
Shift Right Arithmetic	176
Set Register Pointer	178
Stop Operation	179
Subtract	180
Swap Nibbles	182
Test Complement under Mask	183
Test Under Mask	185
Wait for Interrupt	187
Logical Exclusive OR	188
Chapter 8. Clock, Power, and Reset Circuits	190
8.1. System Clock Circuit	190
8.2. Clock Status During Power-Down Modes	191
8.3. System Clock Control Register	192
Chapter 9. Reset	195
9.1. Reset Sources	195

9.2. Reset Mechanism	197
9.3. External Reset Pin	198
9.4. Watchdog Timer Reset	198
9.5. LVD Reset	198
9.6. Internal Power-On Reset	199
9.7. External Interrupt Reset	200
9.8. Stop Error Detection & Recovery	201
9.9. External Reset Pin	201
9.10. Power-Down Modes	201
9.10.1. Idle Mode	202
9.10.2. Idle Mode Release	202
9.10.3. Backup Mode	203
9.10.4. Stop Mode	205
9.10.5. Sources to Release Stop Mode	206
9.10.6. Using nRESET Pin to Release Stop Mode	206
9.10.7. Using IPOR to Release Stop Mode	207
9.10.8. Using an External Interrupt to Release Stop Mode	207
9.10.9. Stop Error Detect and Recovery	207
9.11. System Reset Operation	208
9.11.1. Hardware Reset Values	209
9.12. Recommendation for Unused Pins	214
9.13. Reset Source Indicating Register	216
Chapter 10. I/O Ports	217
10.1. Port Data Registers	218
10.2. Port 0	219
10.2.1. Port 1	225
10.2.2. Port 2	229
10.3. Port 3	236
Chapter 11. Basic Timer and Timer 0	244
11.1. Basic Timer	245
11.2. Basic Timer Control Register	246
11.3. Timer 0 Control Register	248
Chapter 12. Timer 1	256
12.1. Timer 1 Control Register	258
12.2. Timer 1 Overflow Interrupt	260
12.3. Timer 1 Capture Interrupt	260
12.4. Timer 1 Match Interrupt	261

Chapter 13. Counter A	263
13.1. Counter A Control Register	264
13.1.1. Counter A Pulse Width Calculations	266
Chapter 14. Timer 2	270
14.1. Timer 2 Control Register	272
14.2. Timer 2 Overflow Interrupt	274
14.3. Timer 2 Capture Interrupt	274
14.4. Timer 2 Match Interrupt	275
Chapter 15. Embedded Flash Memory Interface	277
15.1. Flash ROM Configuration	277
15.2. User Program Mode	278
15.3. ISP On-Board Programming Sector	278
15.4. ISP Reset Vector and ISP Sector Size	280
15.5. Flash Memory Control Registers	281
15.5.1. Flash Memory User Programming Enable Register	281
15.5.2. Flash Memory Sector Address Registers	282
15.6. Sector Erase Operations	283
15.6.1. The Sector Erase Procedure in User Program Mode	284
15.7. Program Operations	287
15.8. Read Operations	293
15.8.1. Hard Lock Protection	293
Chapter 16. Low Voltage Detection	295
16.1. LVD Flag	296
16.2. Low Voltage Detection Control Register	297
16.3. Low Voltage Detection Flag Selection Register	298
Chapter 17. Electrical Characteristics	299
Chapter 18. Mechanical Data	309
Chapter 19. Flash Programming	313
19.1. Test Pin Voltage	314
19.2. Operating Mode Characteristics	314
Chapter 20. Development Tools	315
20.1. Development System Configuration	315
20.2. TB80PB Target Board	316
20.3. Probe Adapter	320
20.4. Third Parties for Development Tools	320



Chapter 21. Ordering Information 321
 21.0.1. Part Number Suffix Designations 321
Customer Support 322

List of Figures

Figure 1.	32-Pin SOP/QFN Block Diagram	4
Figure 2.	44-Pin QFP/QFN Block Diagram	5
Figure 3.	Pin Assignments, 32-Pin SOP Package	6
Figure 4.	Pin Assignments, 32-Pin QFN Package	7
Figure 5.	Pin Assignments, 44-Pin QFP and QFN Packages	10
Figure 6.	Port 0 Pin Circuit, Type 1	13
Figure 7.	Port 1, Port 4, P3.4 and P3.5 Pin Circuit, Type 2	14
Figure 8.	Port 2 Pin Circuit, Type 3	15
Figure 9.	Port 3.0 Pin Circuit, Type 4	16
Figure 10.	P3.1 Pin Circuit, Type 5	17
Figure 11.	P3.2 and P3.3 Pin Circuit, Type 6	18
Figure 12.	nRESET Pin Circuit, Type 7	18
Figure 13.	Program Memory Address Space	20
Figure 14.	Smart Option	22
Figure 15.	Internal Register File Organization	25
Figure 16.	Set1, Set2, and Prime Area Register Map	27
Figure 17.	8-Byte Working Register Areas	28
Figure 18.	Contiguous 16-Byte Working Register Block	29
Figure 19.	Noncontiguous 16-Byte Working Register Block	29
Figure 20.	16-Bit Register Pair	31
Figure 21.	Register File Addressing	32
Figure 22.	Common Working Register Area	33
Figure 23.	4-Bit Working Register Addressing	35
Figure 24.	4-Bit Working Register Addressing Example	36
Figure 25.	8-Bit Working Register Addressing	37
Figure 26.	8-Bit Working Register Addressing Example	38
Figure 27.	Stack Operations	40
Figure 28.	Register Addressing	44
Figure 29.	Working Register Addressing	44
Figure 30.	Indirect Register Addressing to Register File	45
Figure 31.	Indirect Register Addressing to Program Memory	46
Figure 32.	Indirect Register Addressing to Register File	47
Figure 33.	Indirect Register Addressing to Program or Data Memory	48
Figure 34.	Indexed Addressing to Register File	49

Figure 35. Indexed Addressing to Program or Data Memory with Short Offset	50
Figure 36. Indexed Addressing to Program or Data Memory	51
Figure 37. Direct Addressing for Load Instructions	52
Figure 38. Direct Addressing for Call and Jump Instructions	53
Figure 39. Indirect Addressing	54
Figure 40. Relative Addressing	55
Figure 41. Immediate Addressing	56
Figure 42. S3 Family Interrupt Types	62
Figure 43. S3F80PB Interrupt Structure	64
Figure 44. ROM Vector Address Area	65
Figure 45. Interrupt Function Diagram	69
Figure 46. Interrupt Request Priority Groups	74
Figure 47. How to Use an ENTER Statement	129
Figure 48. How to Use an EXIT Statement	130
Figure 49. Instruction Pointer	136
Figure 50. How to Use the NEXT Instruction	151
Figure 51. Rotate Left	163
Figure 52. Rotate Left through Carry	165
Figure 53. Rotate Right	167
Figure 54. Rotate Right through Carry	169
Figure 55. Shift Right	176
Figure 56. Swap Nibbles	182
Figure 57. Sample Program Structure	187
Figure 58. Main Oscillator Circuit	190
Figure 59. External Clock Circuit	191
Figure 60. System Clock Circuit Diagram	192
Figure 61. Power Circuit	193
Figure 62. nRESET Circuit	194
Figure 63. Reset Sources of the S3F80PB MCU	196
Figure 64. Reset Block Diagram of the S3F80PB MCU	197
Figure 65. Reset Block Diagram by LVD for the S3F80PB MCU in Stop Mode	198
Figure 66. Timing Diagram for Internal Power-On Reset Circuit	199
Figure 67. Reset Timing Diagram for the S3F80PB MCU in Stop Mode by IPOR	200
Figure 68. Block Diagram for Backup Mode	203
Figure 69. Timing Diagram for Backup Mode Input and Released by LVD	204
Figure 70. Timing Diagram for Backup Mode Input in Stop Mode	205
Figure 71. S3F80PB I/O Port Data Register Format	219

Figure 72.	Basic Timer and Timer 0 Block Diagram	244
Figure 73.	Simplified Timer 0 Function Diagram: Interval Timer Mode	251
Figure 74.	Simplified Timer 0 Function Diagram: PWM Mode	252
Figure 75.	Simplified Timer 0 Function Diagram: Capture Mode	253
Figure 76.	Timer 1 Block Diagram	256
Figure 77.	Simplified Timer 1 Function Diagram: Capture Mode	261
Figure 78.	Simplified Timer 1 Function Diagram: Interval Timer Mode	262
Figure 79.	Counter A Block Diagram	263
Figure 80.	Counter A Pulse Width	266
Figure 81.	Counter A Output Flip-Flop Waveforms in Repeat Mode	267
Figure 82.	38 kHz, 1/3 Duty Carrier Frequency	267
Figure 83.	One-Shot Mode	268
Figure 84.	Timer 2 Block Diagram	270
Figure 85.	Simplified Timer 2 Function Diagram: Capture Mode	275
Figure 86.	Simplified Timer 2 Function Diagram: Interval Timer Mode	276
Figure 87.	Program Memory Address Space	279
Figure 88.	Sector Configurations in User Program Mode	284
Figure 89.	Sector Erase Routine in User Program Mode	285
Figure 90.	Byte Program Flowchart in User Program Mode	288
Figure 91.	Program Flowchart in User Program Mode	289
Figure 92.	Low Voltage Detect Block Diagram	295
Figure 93.	Stop Mode to Normal Mode Timing Diagram, #1 of 2	303
Figure 94.	Stop Mode to Normal Mode Timing Diagram, #2 of 2	304
Figure 95.	Input Timing for External Interrupts, Ports 0 and 2	304
Figure 96.	Input Timing for Reset (nRESET Pin)	305
Figure 97.	Operating Voltage Range	307
Figure 98.	32-Pin SOP Package Dimension	309
Figure 99.	32-Pin QFN Package Dimension	310
Figure 100.	44-Pin QFP Package Dimension	311
Figure 101.	44-Pin QFN Package Dimension	312
Figure 102.	Development System Configuration	315
Figure 103.	TB80P0 Target Board Configuration	316
Figure 104.	50-Pin Connector Pin Assignment, User System	319
Figure 105.	TB80PB Probe Adapter Cable	320

List of Tables

Table 1.	Pin Descriptions, 32-Pin SOP/QFN Packages	8
Table 2.	Pin Descriptions, 44-Pin QFP/QFN Packages	11
Table 3.	S3F80PB Register Types	24
Table 4.	Register Page Pointer	26
Table 5.	Register Pointer 0	39
Table 6.	Register Pointer 1	39
Table 7.	Stack Pointer Low Byte	42
Table 8.	Set1, Bank0 Mapped Registers	57
Table 9.	Set1, Bank1 Mapped Registers	59
Table 10.	External Memory Timing Register	60
Table 11.	S3F80PB Interrupt Vectors	66
Table 12.	Interrupt Control Register Overview	68
Table 13.	Vectored Interrupt Source Control and Data Registers	70
Table 14.	System Mode Register	71
Table 15.	Interrupt Mask Register	73
Table 16.	Interrupt Priority Register	75
Table 17.	Interrupt Request Register	76
Table 18.	Instruction Pointer High Byte	80
Table 19.	Instruction Pointer Low Byte	80
Table 20.	Instruction Summary	84
Table 21.	System Flags Register	87
Table 22.	Flags	88
Table 23.	Flag Notation Conventions	89
Table 24.	Instruction Set Symbols	89
Table 25.	Instruction Notation Conventions	90
Table 26.	Op Code Quick Reference (0–7)	91
Table 27.	Op Code Quick Reference (8–F)	92
Table 28.	Condition Codes	93
Table 29.	DA Instruction	120
Table 30.	System Clock Control Register	193
Table 31.	Falling and Rising Time of Operating Voltage	194
Table 32.	Reset Conditions in Stop Mode	201
Table 33.	Stop Control Register	208

Table 34.	Set 1, Bank0 Register Values After Reset	210
Table 35.	Set1, Bank1 Register Values After Reset	212
Table 36.	Smart Option Reset Generation	213
Table 37.	Guideline for Unused Pins to Reduced Power Consumption	214
Table 38.	Summary of Each Mode	215
Table 39.	Reset Source Indicating Register	216
Table 40.	State of RESETID Depends on Reset Source	216
Table 41.	S3F80PB Port Configuration Overview (44-Pin QFP, QFN)	217
Table 42.	S3F80PB Port Configuration Overview (32-Pin SOP, QFN)	218
Table 43.	Port Data Register Summary	219
Table 44.	Port 0 Control	220
Table 45.	Port 0 Control Low Byte Register	221
Table 46.	Port 0 External Interrupt Enable Register	222
Table 47.	Port 0 External Interrupt Pending Register	223
Table 48.	Port 0 Pull-Up Resistor Enable Register	224
Table 49.	Port 1 Control High Byte Register	226
Table 50.	Port 1 Control Low Byte Register	227
Table 51.	Port 1 Output Pull-Up Resistor Enable Register	228
Table 52.	Port 2 Control High Byte Register	229
Table 53.	Port 2 Control Low Byte Register	230
Table 54.	Port 2 External Interrupt Enable Register	231
Table 55.	Port 2 Output Mode Selection Register	232
Table 56.	Port 2 External Interrupt Pending Register	233
Table 57.	Port 2 Pull-Up Resistor Enable Register	234
Table 58.	Port 3 Control Register	236
Table 59.	Function Description and Pin Assignment of P3CON	237
Table 60.	Port 3 Output Pull-Up Resistor Enable Register	238
Table 61.	Port 3[4:5] Control Register	239
Table 62.	Port 4 Control Register	240
Table 63.	Port 4 Control High Byte Register	241
Table 64.	Port 4 Control Low Byte Register	242
Table 65.	Port 4 Output Pull-Up Resistor Enable Register	243
Table 66.	Basic Timer Control Register	247
Table 67.	Timer 0 Control Register	249
Table 68.	Timer 0 Reference Data Register	250
Table 69.	Timer 1 Control Register	258

Table 70.	Timer 1 Counter High Byte Register	259
Table 71.	Timer 1 Counter Low Byte Register	259
Table 72.	Timer 1 Data High Byte Register	259
Table 73.	Timer 1 Data Low Byte Register	260
Table 74.	Counter A Control Register	264
Table 75.	Counter A Data High Byte Register	265
Table 76.	Counter A Data Low Byte Register	265
Table 77.	Timer 2 Control Register	272
Table 78.	Timer 2 Counter High Byte Register	273
Table 79.	Timer 2 Counter Low Byte Register	273
Table 80.	Timer 2 Data High Byte Register	273
Table 81.	Timer 2 Data Low Byte Register	274
Table 82.	Reset Vector Address	280
Table 83.	ISP Sector Size	280
Table 84.	Flash Memory Control Register	281
Table 85.	Flash Memory User Programming Enable Register	282
Table 86.	Flash Memory Sector Address High Byte Register	283
Table 87.	Flash Memory Sector Address Low Byte Register	283
Table 88.	LVD Voltage Gap Characteristics	296
Table 89.	LVD Flag Gap Characteristics	297
Table 90.	LVD Enable Time	297
Table 91.	LVD Control Register	297
Table 92.	LVD Flag Level Selection Register	298
Table 93.	Absolute Maximum Ratings	300
Table 94.	DC Electrical Characteristics	300
Table 95.	LVD Adder Current in Backup Mode	301
Table 96.	Low Voltage Detect Circuit	302
Table 97.	Low Voltage Detect Circuit	302
Table 98.	LVD Enable Time	302
Table 99.	Power On Reset Circuit	303
Table 100.	Data Retention Supply Voltage in Stop Mode	303
Table 101.	AC Electrical Characteristics	304
Table 102.	Oscillation Characteristics	305
Table 103.	Input/Output Capacitance	306
Table 104.	Oscillation Stabilization Time	306
Table 105.	AC Electrical Characteristics for Internal Flash ROM	307



Table 106. ESD Characteristics 308

Table 107. Flash Operations in Tool Program Mode 313

Table 108. Operating Mode Selection Criteria 314

Table 109. TB80PB Jumper Settings 317

Table 110. TB80PB Target Board LEDs 318

Table 111. Ordering Information for the S3F80PB MCU 321

Chapter 1. Overview

Zilog's S3 Family of 8-bit single-chip microcontrollers offers a fast and efficient CPU, a wide range of integrated peripherals, and multiple Flash memory sizes. Important CPU features include:

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

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

1.1. S3F80PB Microcontroller

The S3F80PB MCU features 63KB of Flash ROM memory. Using a proven modular design approach, the S3F80PB MCU was developed by integrating the following peripheral modules with the SAM8 RC core:

- Internal LVD circuit and 16-bit programmable pins for external interrupts
- One 8-bit basic timer for oscillation stabilization and watchdog function (system reset)
- One 8-bit timer/counter with three operating modes
- Two 16-bit timer/counters with selectable operating modes
- One 8-bit counter with automatic reload function and one-shot or repeat control

The S3F80PB MCU is a versatile general-purpose microcontroller that is especially suitable for use as remote transmitter controller. It is currently available in 32-pin SOP, 32-pin QFN, 44-pin QFP, 44-pin QFN, and Pellet (Die) packages.

1.2. Features

The S3F80PB MCU offers the following features:

- SAM8 RC CPU core
- Program memory:
 - 63 KB internal Flash memory
 - 10 years data retention
 - Endurance: 10,000 erase/program cycles
 - Byte-programmable
 - User-programmable by LDC instruction
- Executable memory: 1KB RAM
- Data memory: 272-byte general-purpose RAM
- Instruction set:
 - 78 instructions
 - Idle and Stop instructions added for power-down modes
- Instruction execution time: 500ns at 8MHz f_{OSC} (minimum)
- 24 interrupt sources with 18 vectors and 9 levels
- I/O ports:
 - Four 8-bit I/O ports (P0–P2, P4) and one 6-bit I/O port (P3) for a total of 38-bit programmable pins (44-pin QFP)
 - Three 8-bit I/O ports (P0–P2) and one 2-bit I/O port (P3) for a total of 26-bit programmable pins (32-pin SOP)
 - Three 8-bit n-channel open-drain pins (P1, P2, and P4) and one 2-bit n-channel open-drain pin (P3; 44-pin QFP)
 - Two 8-bit n-channel open-drain pins (P1, P2) and one 2-bit n-channel open-drain pin (P3; 32-pin SOP)
- Carrier frequency generator: one 8-bit counter with automatic reload function and one-shot or repeat control (Counter A)
- Basic timer and timer/counters:
 - One programmable 8-bit basic timer (BT) for oscillation stabilization control or watchdog timer (software reset) function
 - One 8-bit timer/counter (Timer 0) with three operating modes: Interval, Capture and PWM

- One 16-bit timer/counter (Timer 1) with two operating modes: Interval and Capture
- One 16-bit timer/counter (Timer 2) with two operating modes: Interval and Capture
- Backup Mode:
 - When V_{DD} is lower than V_{LVD} and LVD is ON, the S3F80PB MCU enters Backup Mode to block oscillation
 - When the reset pin is lower than the input low voltage (V_{IL}), the S3F80PB MCU enters Backup Mode to block oscillation and reduce current consumption
- Low Voltage Detect circuit:
 - Low voltage detect to enter Backup Mode and Reset
 - 1.65V (typ.) ± 50 mV.
 - Low voltage detect to control the LVD_Flag bit at 1.90V, 2.00V, 2.10V, and 2.20V (typ.) ± 100 mV (selectable).
 - LVD Reset enable:
 - When voltage at V_{DD} is falling and passes V_{LVD} , the S3F80PB MCU enters Backup Mode
 - When voltage at V_{DD} is rising, a reset pulse is generated at $V_{DD} > V_{LVD}$
 - LVD Stop Mode disable: if voltage at V_{DD} does not fall to V_{POR} , a reset pulse is not generated
- Operating temperature range: -25°C to $+85^{\circ}\text{C}$
- Operating voltage range: 1.60V to 3.6V at 1 to 8MHz
- Packages:
 - 32-pin SOP
 - 32-pin QFN
 - 44-pin QFP
 - 44-pin QFN

1.3. Block Diagram

Figure 1 shows a block diagram for the S3F80PB MCU's, 32-pin SOP and 32-pin QFN-packages.

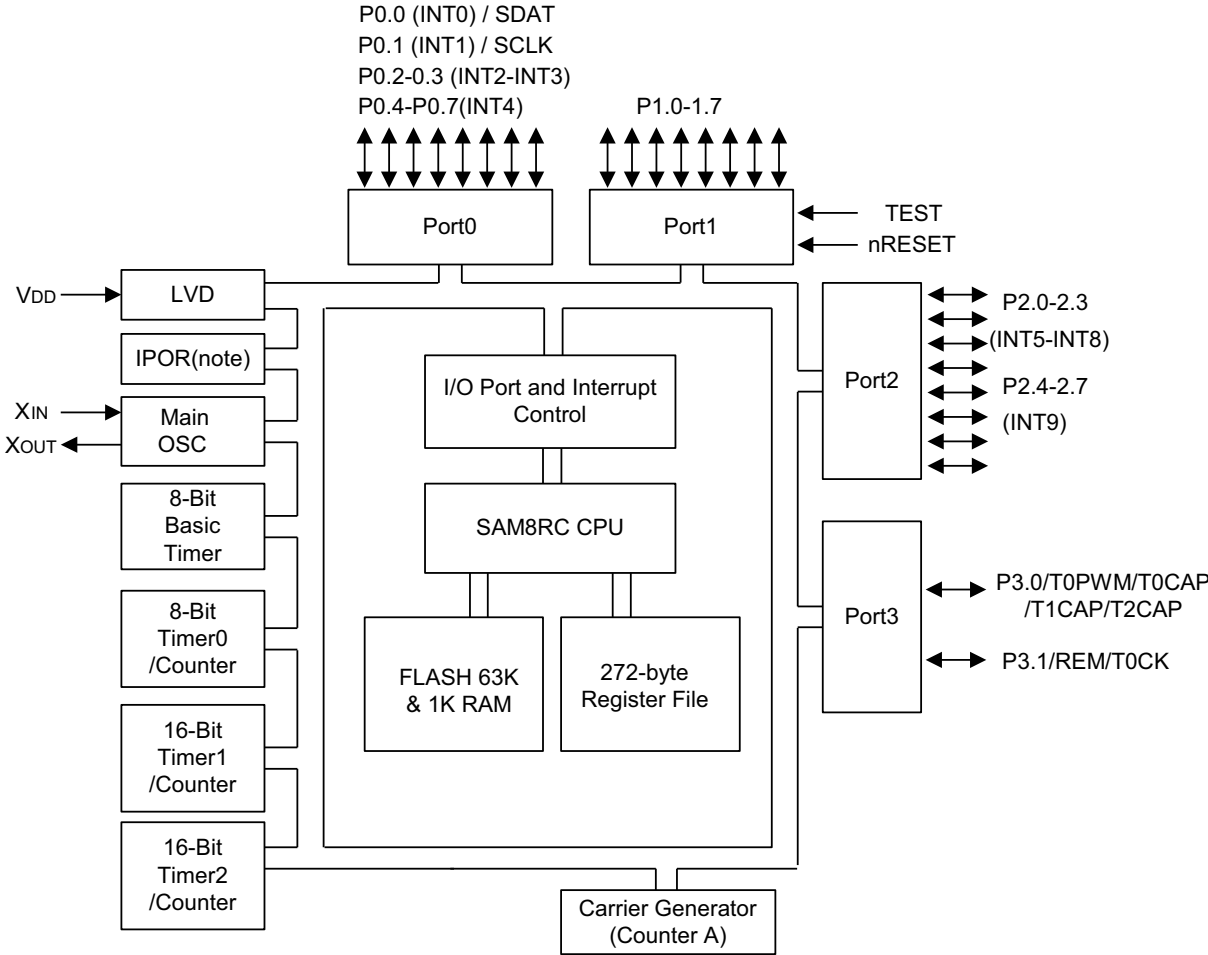


Figure 1. 32-Pin SOP/QFN Block Diagram

Figure 2 shows a block diagram for the S3F80PB MCU's, 44-pin QFP and 44-pin QFN-packages.

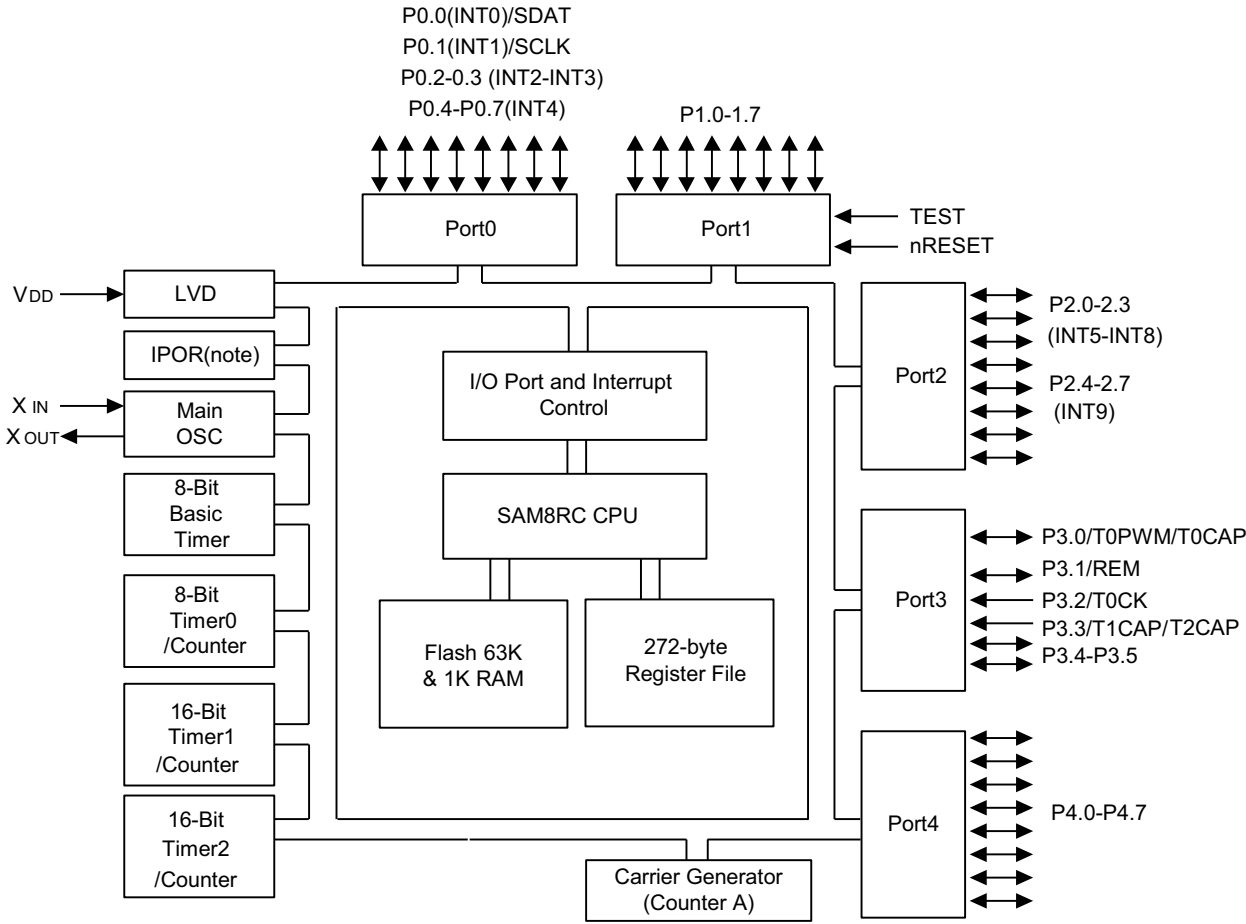


Figure 2. 44-Pin QFP/QFN Block Diagram

1.4. Pin Assignments

Figures 3 and 4 show the pin assignments for the 32-pin SOP and QFN packages, respectively.

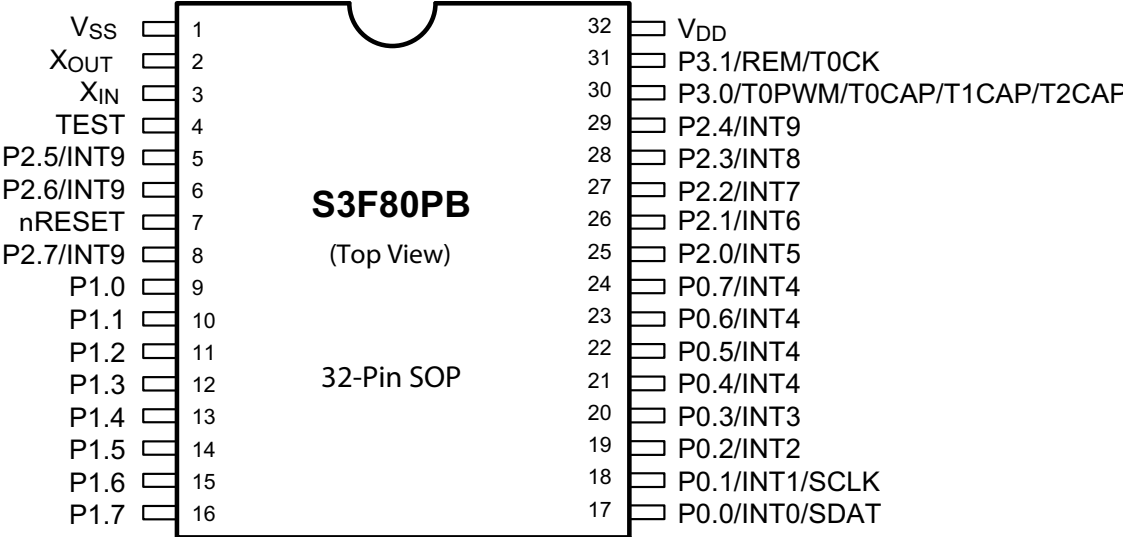


Figure 3. Pin Assignments, 32-Pin SOP Package

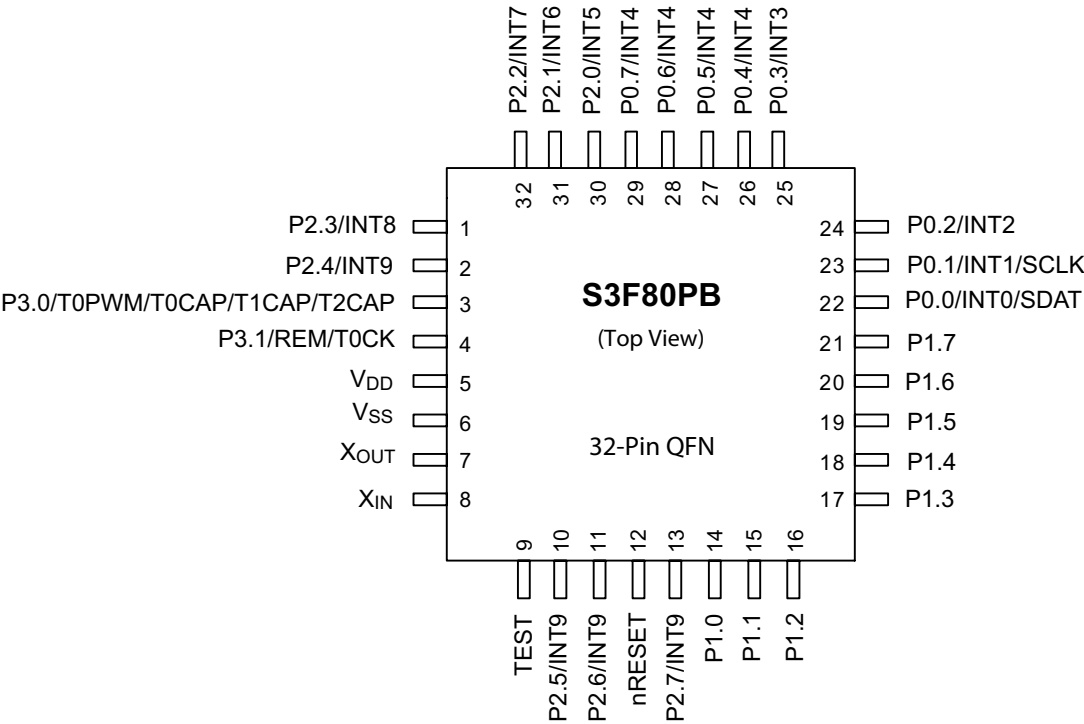


Figure 4. Pin Assignments, 32-Pin QFN Package

Table 1 identifies each pin in the S3F80PB MCU’s 32-pin SOP and 32-Pin QFN packages.

Table 1. Pin Descriptions, 32-Pin SOP/QFN Packages

Pin Name	Pin Description	Pin Type	Circuit Type	SOP Pkg. Pins	QFN Pkg. Pins	Shared Functions
P0.0–P0.7	I/O port with bit-programmable pins. Configurable to Input or Push-Pull Output Mode. Pull-up resistors can be assigned by software. Pins can be assigned individually as external interrupt inputs with noise filters, interrupt enable/disable, and interrupt pending control. In the tool mode, P0.0 and P0.1 are assigned as serial MTP interface pins; SDAT and SCLK.	I/O	1	17–24	22–29	External Interrupts INT0–INT3 INT4, SDAT, SCLK
P1.0–P1.7	I/O port with bit-programmable pins. Configurable to Input Mode or Output Mode. Pin circuits are either push-pull or n-channel open-drain type.	I/O	2	9–16	14–21	–
P2.0–P2.7	I/O port with bit-programmable pins. Configurable to Input Mode, Push-Pull Output Mode, or n-channel Open-Drain Output Mode. Pull-up resistors can be assigned by software. Pins can be assigned individually as external interrupt inputs with noise filters, interrupt enable/disable, and interrupt pending control.	I/O	3	25–29, 5, 6, 8	30–32, 1–2, 10–11, 13	External Interrupts INT5–INT8, INT9
P3.0	I/O port with a bit-programmable pin. Configurable to Input Mode, Push-Pull Output Mode, or n-channel Open-Drain Output Mode. Input Mode with a pull-up resistor can be assigned by software. This Port 3 pin features high current drive capability. Also P3.0 can be assigned individually as an output pin for T0PWM or input pin for T0CAP/ T1CAP/T2CAP.	I/O	4	30	3	T0PWM/T0CAP/ T1CAP/T2CAP

Table 1. Pin Descriptions, 32-Pin SOP/QFN Packages (Continued)

Pin Name	Pin Description	Pin Type	Circuit Type	SOP Pkg. Pins	QFN Pkg. Pins	Shared Functions
P3.1	I/O port with a bit-programmable pin. Configurable to Input Mode, Push-Pull Output Mode, or n-channel Open-Drain Output Mode. Input Mode with a pull-up resistor can be assigned by software. This Port 3 pin features high current drive capability. Also P3.1 can be assigned individually as an output pin for REM or input pin for T0CK.	I/O	5	31	4	REM/T0CK
X _{OUT} , X _{IN}	System clock input and output pins	–	–	2, 3	7, 8	–
nRESET	System reset signal input pin and Backup Mode input. Zilog recommends adding a 0.1 μF capacitor between nRESET pin and V _{SS} for better noise immunity.	I	7	7	12	–
TEST	Test signal input pin. If on board programming is required, Zilog recommends to add a 0.1 μF capacitor between the TEST pin and V _{SS} for better noise immunity; otherwise, connect the TEST pin to V _{SS} directly.	I	–	4	9	–
V _{DD}	Power supply input pin	–	–	32	5	–
V _{SS}	Ground Pin	–	–	1	6	–

Figure 5 shows the pin assignments for both the 44-pin QFP and 44-pin QFN packages.

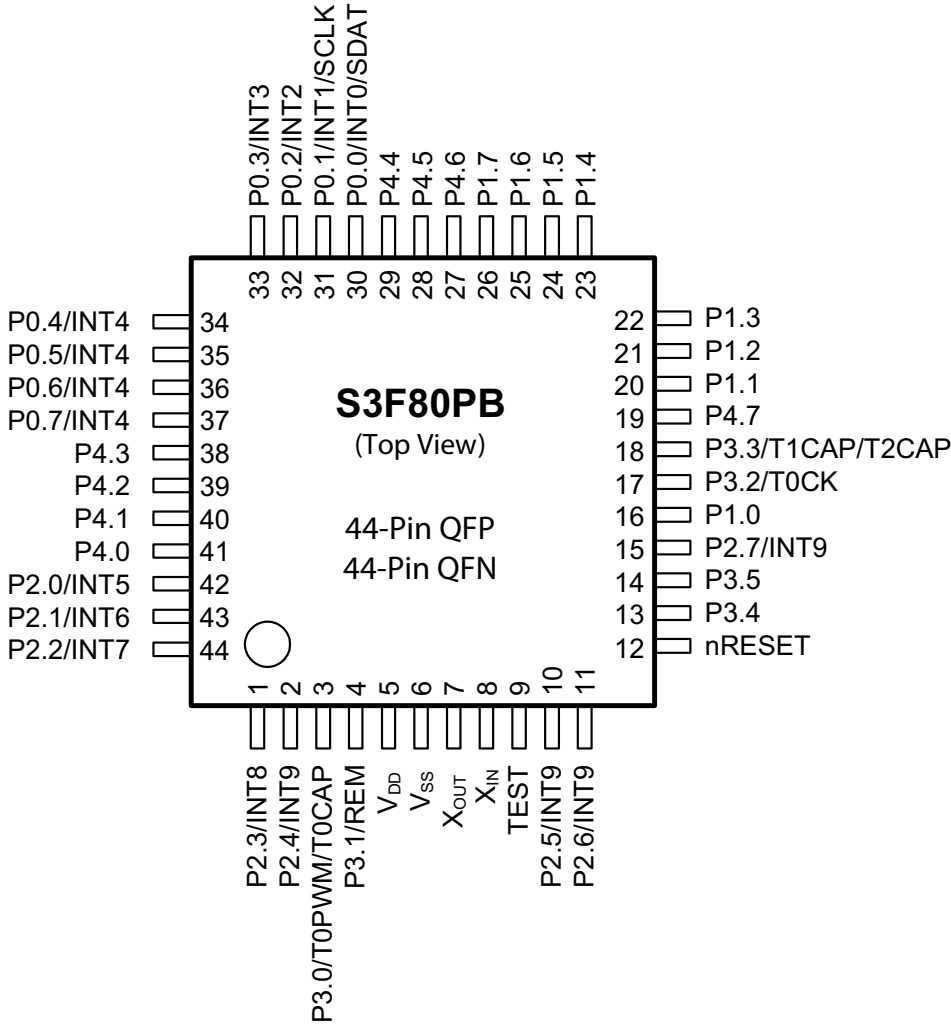


Figure 5. Pin Assignments, 44-Pin QFP and QFN Packages

Table 2 identifies each pin in the S3F80PB MCU’s 44-pin QFP and 44-Pin QFN packages.

Table 2. Pin Descriptions, 44-Pin QFP/QFN Packages

Pin Name	Pin Description	Pin Type	Circuit Type	QFP Pkg. Pins	QFN Pkg. Pins	Shared Functions
P0.0–P0.7	I/O port with bit-programmable pins. Configurable to Input or Push-Pull Output Mode. Pull-up resistors can be assigned by software. Pins can be assigned individually as external interrupt inputs with noise filters, interrupt enable/disable, and interrupt pending control. In Tool Mode, P0.0 and P0.1 are assigned as serial MTP interface pins; SDAT and SCLK.	I/O	1	30–37	30–37	External Interrupts INT0–INT3, INT4, SDAT, SCLK
P1.0–P1.7	I/O port with bit-programmable pins. Configurable to Input Mode or Output Mode. Pin circuits are either Push-Pull or n-Channel Open-Drain type.	I/O	2	16, 20–26	16, 20–26	–
P2.0–P2.3 P2.4–P2.7	I/O port with bit-programmable pins. Configurable to Input Mode, Push-Pull Output Mode, or n-Channel Open-Drain Output Mode. Pull-up resistors can be assigned by software. Pins can be assigned individually as external interrupt inputs with noise filters, interrupt enable/disable, and interrupt pending control.	I/O	3	42, 44, 1, 2, 10, 11, 15	42–44, 1, 2, 10, 11, 15	External Interrupts INT5–INT8, INT9
P3.0	I/O port with a bit-programmable pin. Configurable to Input Mode, Push-Pull Output Mode, or n-Channel Open-Drain Output Mode. Input Mode with a pull-up resistor can be assigned by software. This Port 3 pin features high current drive capability. Also P3.0 can be assigned individually as an output pin for T0PWM or input pin for T0CAP.	I/O	4	3	3	T0PWM/T0CAP

Table 2. Pin Descriptions, 44-Pin QFP/QFN Packages (Continued)

Pin Name	Pin Description	Pin Type	Circuit Type	QFP Pkg. Pins	QFN Pkg. Pins	Shared Functions
P3.1	I/O port with a bit-programmable pin. Configurable to Input Mode, Push-Pull Output Mode, or n-Channel Open-Drain Output Mode. Input Mode with a pull-up resistor can be assigned by software. This Port 3 pin features high current drive capability. Also P3.1 can be assigned individually as an output pin for REM.	I/O	5	4	4	REM
P3.2–P3.3	C-MOS Input port with a pull-up resistor	I	6	17, 18	17, 18	T0CK T1CAP/T2CAP
P3.4–P3.5	I/O port with bit-programmable pins. Configurable to input mode or output mode. Pin circuits are either push-pull or n-channel open-drain type. Pull-up resistors can be assigned by software.	I/O	2	13, 14	13, 14	–
P4.0–P4.7	I/O port with bit-programmable pins. Configurable to input mode or output mode. Pin circuits are either push-pull or n-channel open-drain type.	I/O	2	38–41, 27–29, 19	38–41, 27–29, 19	–
X _{OUT} , X _{IN}	System clock input and output pins	–	–	7, 8	7, 8	–
nRESET	System reset signal input pin and Backup Mode input. Zilog recommends adding a 0.1 μ F capacitor between nRESET pin and V _{SS} for better noise immunity.	I	7	12	12	–
TEST	Test signal input pin. If on board programming is required, Zilog recommends adding a 0.1 μ F capacitor between the TEST pin and V _{SS} for better noise immunity; otherwise, connect the TEST pin to V _{SS} directly.	I	–	9	9	–
V _{DD}	Power supply input pin	–	–	5	5	–
V _{SS}	Ground Pin	–	–	6	6	–

1.5. Pin Circuits

Figure 6 shows the Type 1 pin circuit for Port 0.

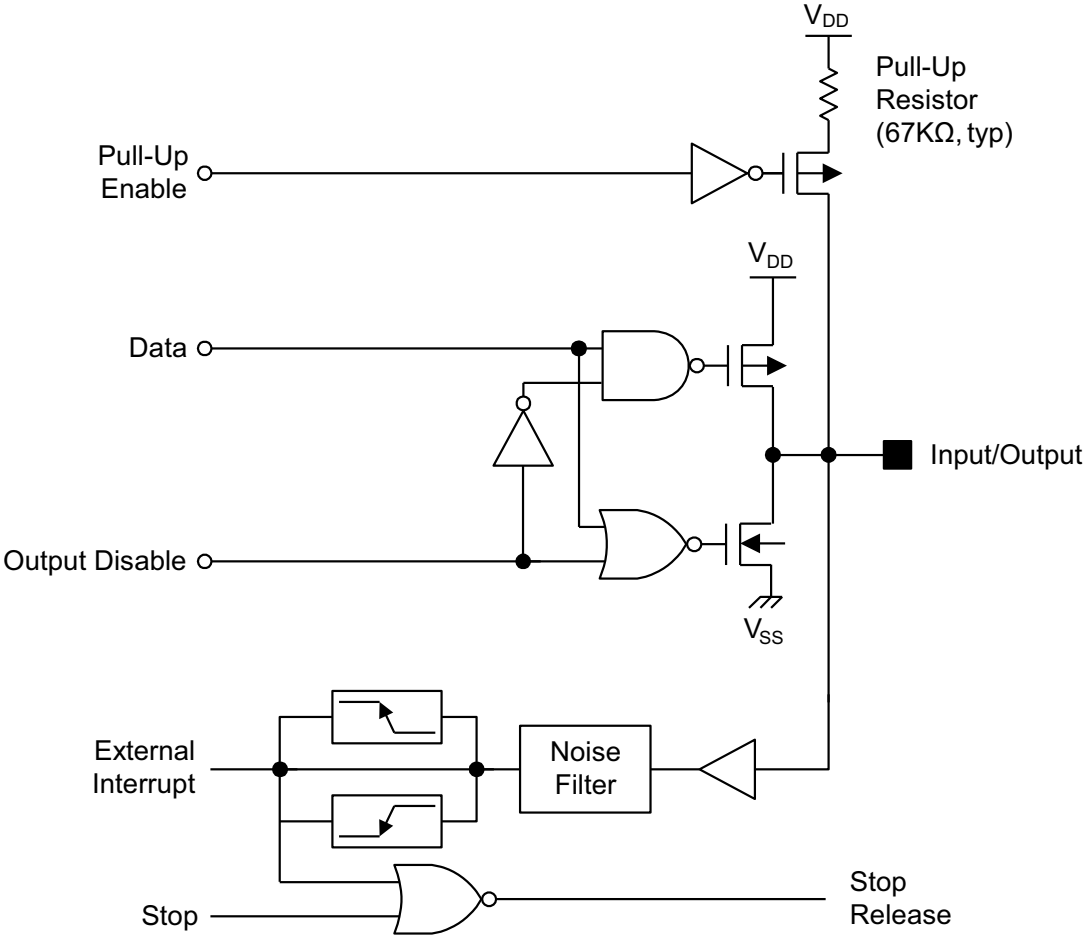


Figure 6. Port 0 Pin Circuit, Type 1

Figure 7 shows the Type 2 pin circuit for Port 1, Port 4, P3.4, and P3.5.

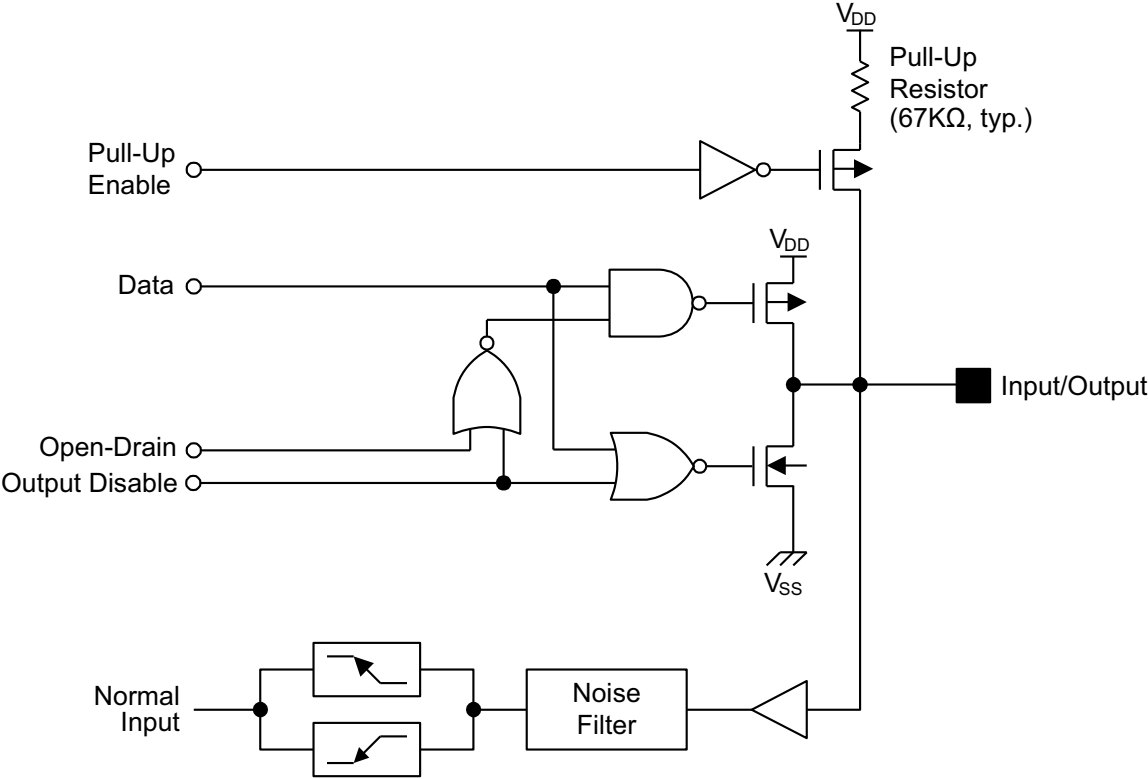


Figure 7. Port 1, Port 4, P3.4 and P3.5 Pin Circuit, Type 2

Figure 8 illustrates the Type 3 pin circuit for Port 2.

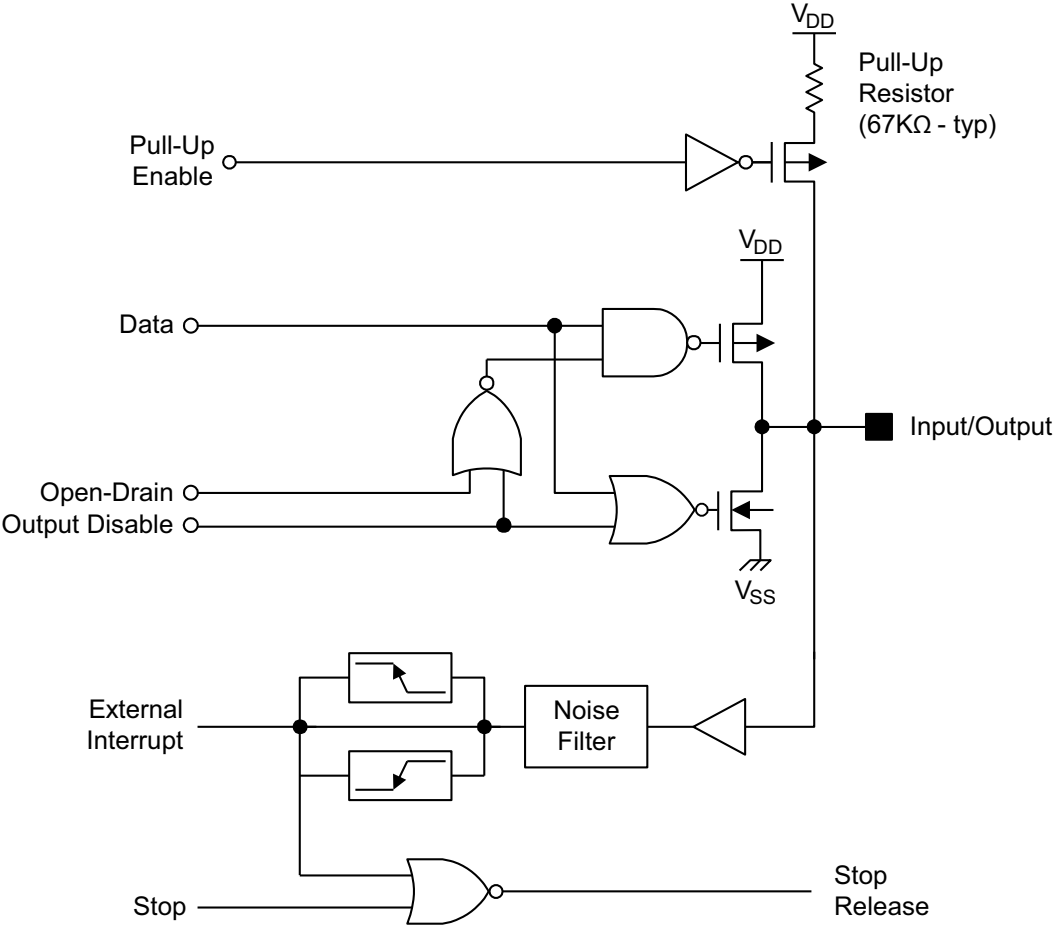


Figure 8. Port 2 Pin Circuit, Type 3

Figure 9 shows the Type 4 pin circuit for Port 3.0.

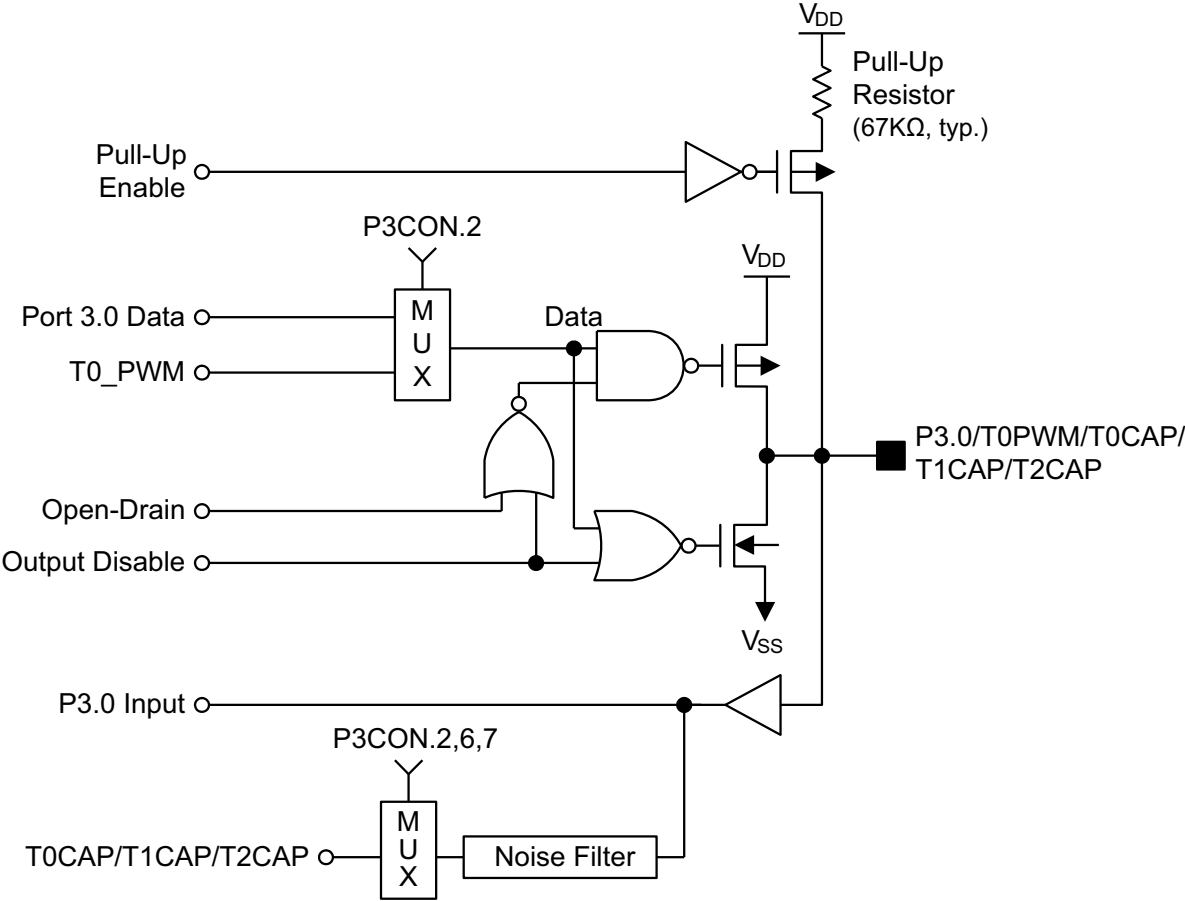


Figure 9. Port 3.0 Pin Circuit, Type 4

Figure 10 shows the Type 5 pin circuit for P3.1.

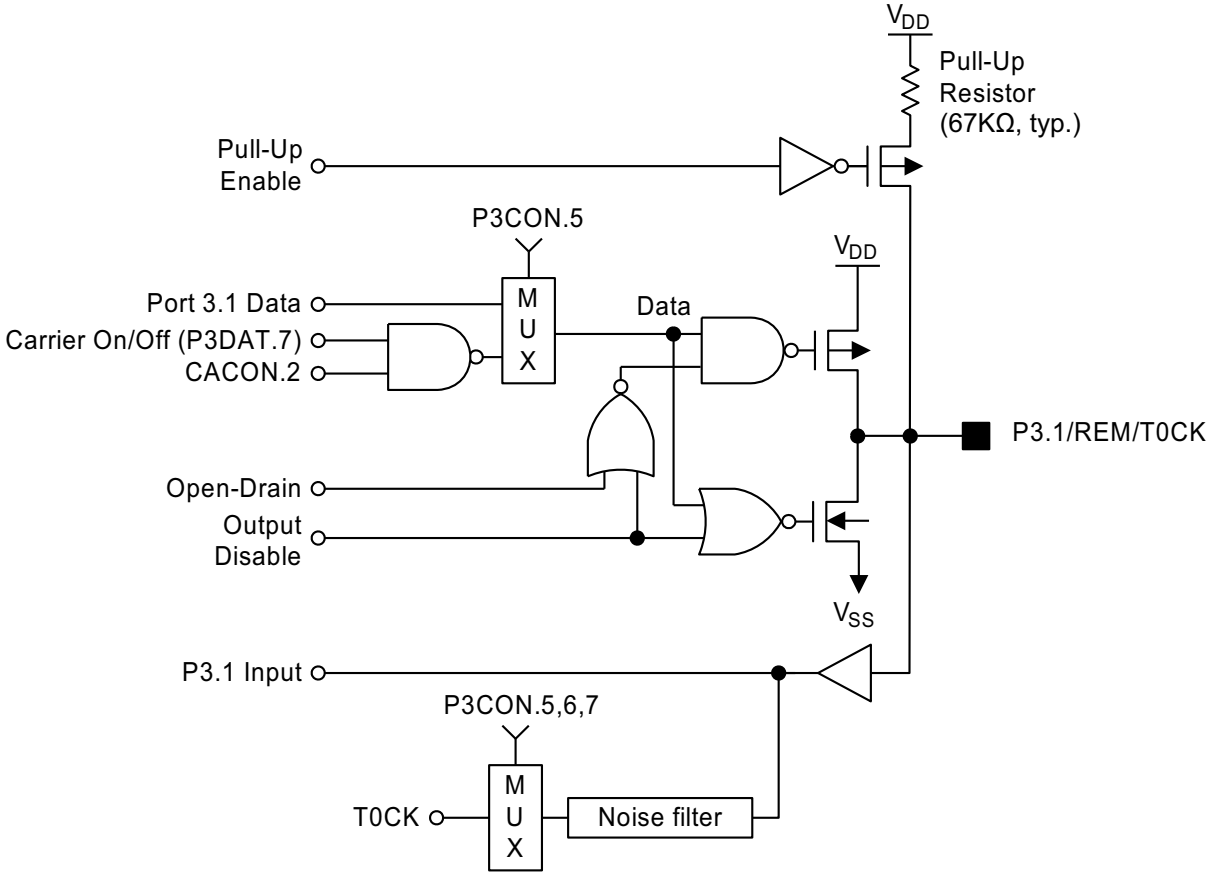


Figure 10. P3.1 Pin Circuit, Type 5

Figure 11 shows the Type 6 pin circuit for P3.2 and P3.3.

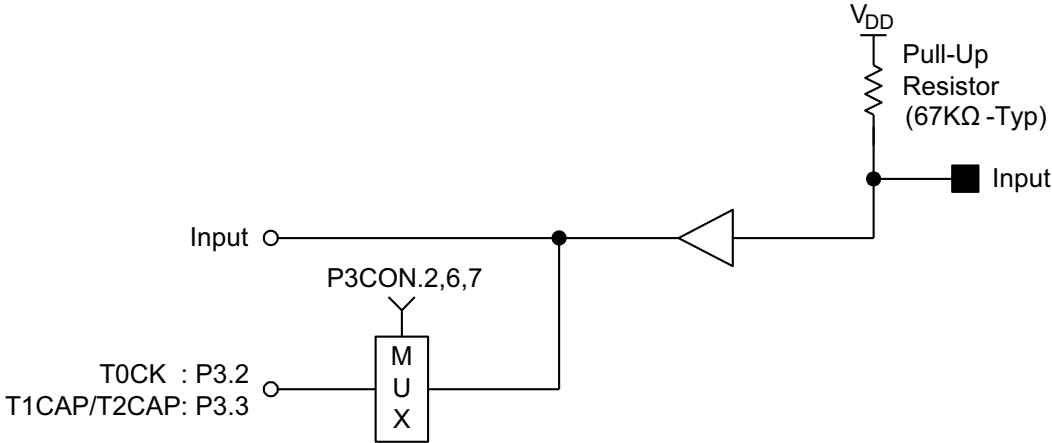


Figure 11. P3.2 and P3.3 Pin Circuit, Type 6

Figure 12 shows the Type 7 pin circuit for nRESET.

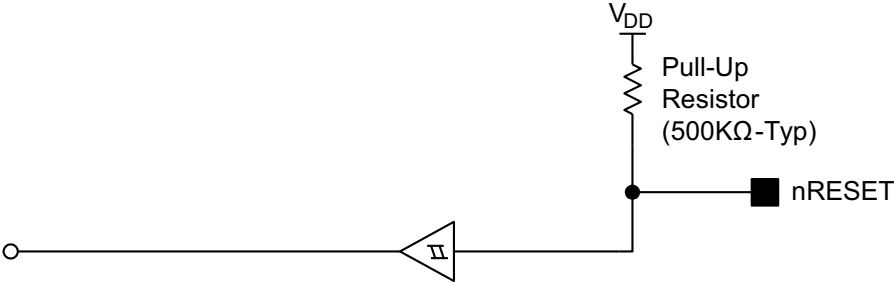


Figure 12. nRESET Pin Circuit, Type 7

Chapter 2. Address Space

The S3F80PB microcontroller features two types of address spaces:

- Internal program (Flash) memory
- Internal register file

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

The S3F80PB MCU features 63KB of programmable internal Flash ROM. An external memory interface is not implemented.

The internal register file contains 333 mapped registers; of these, 272 bytes are designated for general-purpose use. This number includes a 16-byte working register common area that is used as a scratch area for data operations, a 192-byte prime register area, and a 64-byte area, Set2, that is also used for stack operations. Twenty-two 8-bit registers are used for CPU and system control; 39 registers are mapped peripheral control and data registers.

2.1. Program Memory

Program memory stores program code or table data. The S3F80PB MCU includes a memory map option which is 63KB of internal programmable Flash memory and 2KB of executable RAM.

The program memory address range is therefore 0000h–FFFFh of Flash memory, as shown in Figure 13.

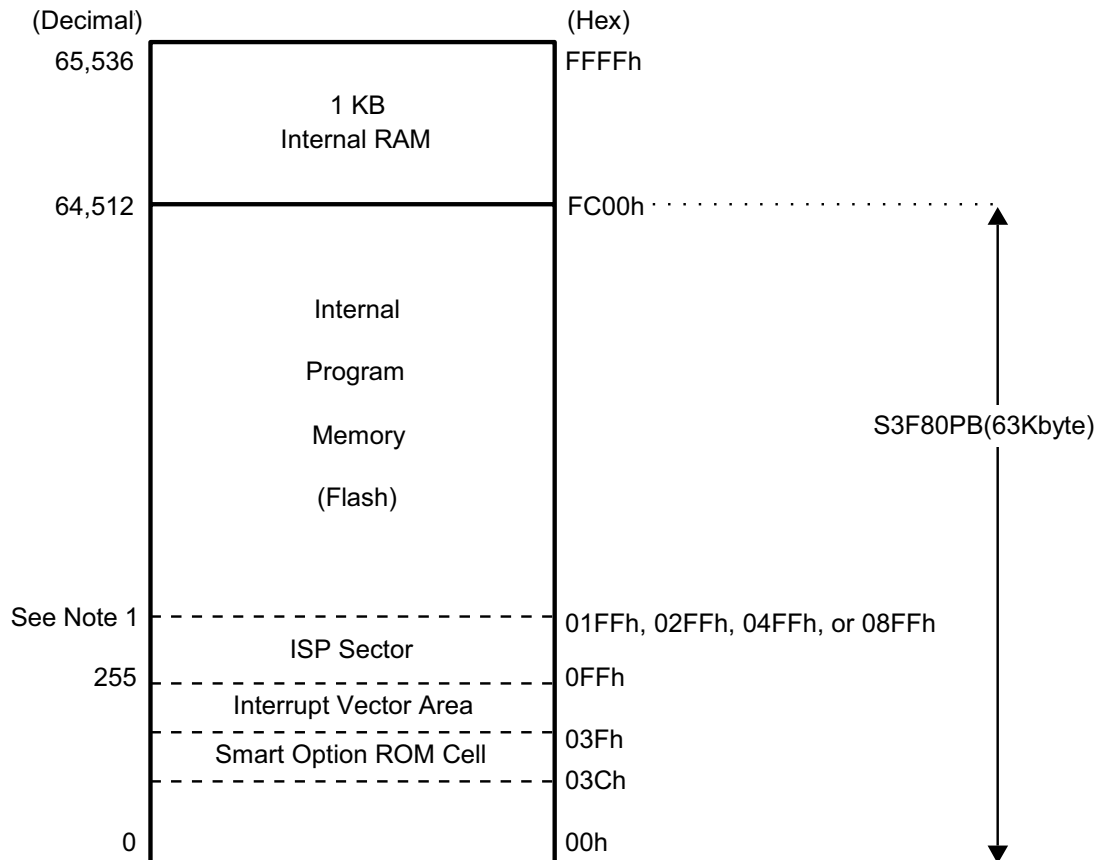


Figure 13. Program Memory Address Space

- **Notes:**
1. In Figure 13, the size of the ISP sector can be varied by the Smart Option. According to the Smart Option settings, the ISP reset vector address can be changed to one of the following addresses: 200h, 300h, 500h, or 900h.
 2. The ISP sector can store on-board program software. To learn more, see the [Embedded Flash Memory Interface](#) chapter on page 277.

The first 256 bytes of the program memory (0h–0FFh) are reserved for interrupt vector addresses. Unused locations (in the 0000h–00FFh address range, with the exception of 03Ch, 03Dh, 03Eh, and 03Fh) can be used as normal program memory. The 03Ch, 03Dh,

03Eh, and 03Fh locations are used as a Smart Option ROM cell. Avoid overwriting vector addresses stored in these locations when using the vector address area to store program code.

By default, the program memory address at which program execution starts after reset is 0100h. The reset vector address can be changed by setting the Smart Option when using ISP sectors for ISP software storage; see Figure 14 and the Smart Option section, which follow.

2.1.1. Smart Option

The *Smart Option*, diagrammed in Figure 14, is the program memory option for setting the starting condition of the S3F80PB MCU. The program memory addresses used by the Smart Option are from 003Ch to 003Fh. The S3F80PB MCU uses only addresses in the range 003Eh to 003Fh.

Users can write any value to the unused addresses (003Ch and 003Dh). The default value of the Smart Option bits in program memory is 0FFh; the normal reset vector address, 100h, is ISP protection-disabled. Before executing program memory code, set the Smart Option bits according to the preferred hardware option.

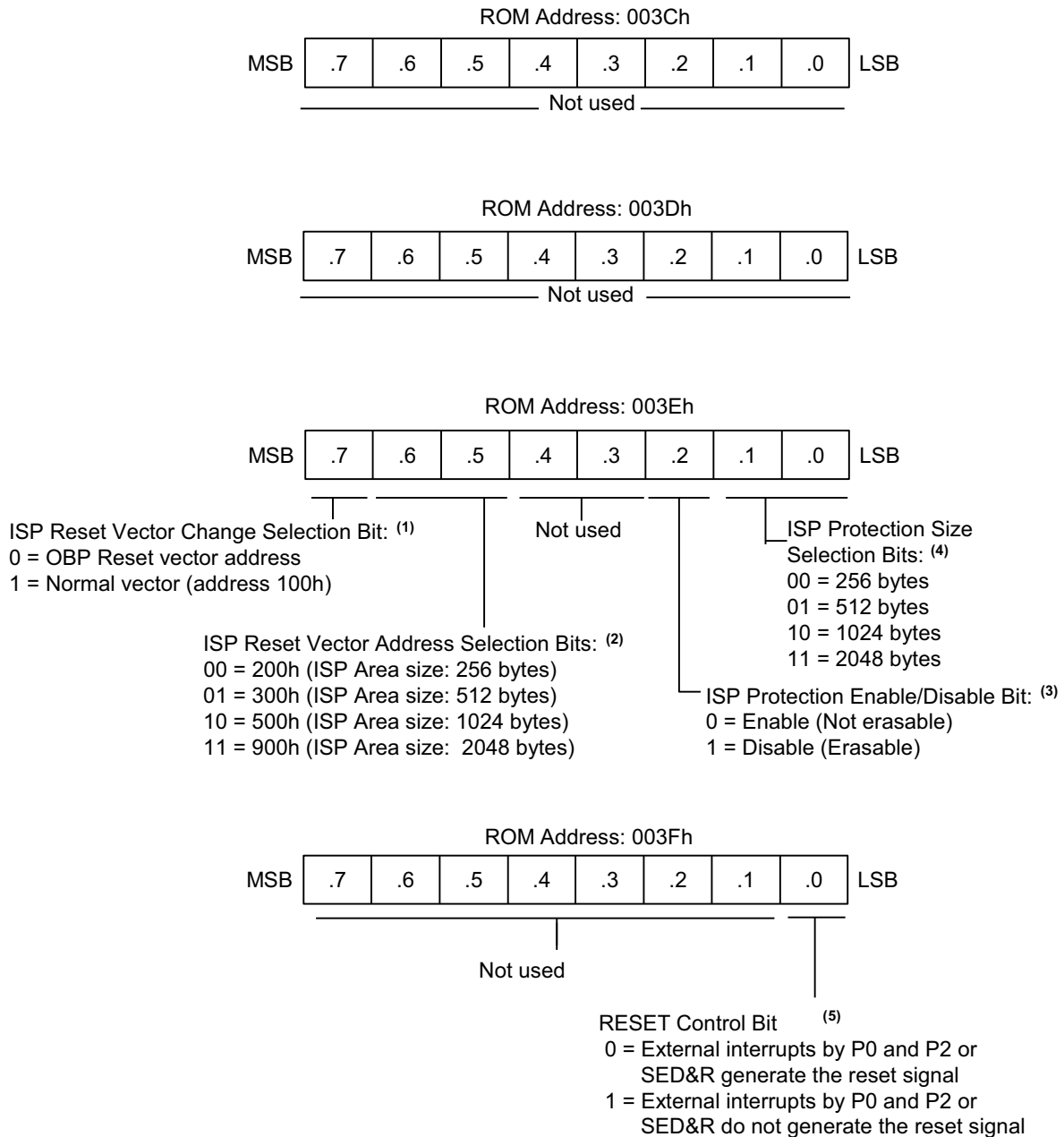


Figure 14. Smart Option

-
- **Notes:**
1. In Figure 14, by setting ISP reset vector change selection bit (3Eh. 7) to 0, the ISP area becomes available. If this bit is 1, 3Eh. 6 and 3Eh. 5 are rendered meaningless.
 2. If the ISP reset vector change selection bit (3Eh. 7) is 0, the user must change the ISP reset vector address from 0100h to an address at which the user prefers to set a reset (i.e., 0200h, 0300h, 0500h, or 0900h). If the reset vector address is 0200h, the ISP area can be assigned from 0100h to 01FFh (an area of 256 bytes). If 0300h, the ISP area can be assigned from 0100h to 02FFh (512 bytes). If 0500h, the ISP area can be assigned from 0100h to 04FFh (1024 bytes). If 0900h, the ISP area can be assigned from 0100h to 08FFh (2048 bytes).
 3. If the ISP protection enable/disable bit is 0, the ISP area selected by 3Eh. 1 and 3Eh. 0 cannot be erased or programmed in Flash memory.
 4. A suitable ISP protection size can be selected using 3Eh. 1 and 3Eh. 0. If the ISP protection enable/disable bit (3Eh. 2) is 1, 3Eh. 1 and 3Eh. 0 are rendered meaningless.
 5. External interrupts can be used to release Stop Mode. When the RESET control bit (3Fh. 0) is 0 and external interrupts are enabled, these external interrupts wake the MCU from Stop Mode and generate a reset signal. Any falling edge input signals of P0 or P2.4–P2.7 can wake the MCU from Stop Mode and generate this reset signal. When the RESET control bit (3Fh. 0) is 1, the S3F80PB MCU only releases Stop Mode and does not generate a reset signal.
-

2.2. Register Architecture

In the S3F80PB implementation, the upper 64-byte area of register files is expanded to two 64-byte areas, called *Set1* and *Set2*. The upper 32-byte area of *Set1* is further expanded into two 32-byte register banks (i.e., Bank0 and Bank1). The lower 32-byte area is a single 32-byte common area.

In the S3F80PB MCU, the total number of addressable 8-bit registers is 333. Of these 333 registers, 22 bytes are designated for the CPU and system control registers, 39 bytes are designated for the peripheral control and data registers, 16 bytes are used as shared working registers, and 272 registers are designated for general-purpose use.

The extension of the register space into separately-addressable areas (i.e., sets and banks) is supported by multiple addressing mode restrictions: the select bank instructions, SB0 and SB1.

Specific register types and the area occupied in the S3F80PB internal register space are summarized in Table 3.

Table 3. S3F80PB Register Types

Register Type	Number of Bytes
General-purpose registers, including the 16-byte common working register area, the 64-byte Set2 area and 192-byte prime register area of Page 0.	272
CPU and system control registers	22
Mapped clock, peripheral, and I/O control and data registers (Bank0: 27 registers, Bank1: 12 registers)	39
Total addressable bytes	333

Figure 15 shows the organization of the Internal Register File.

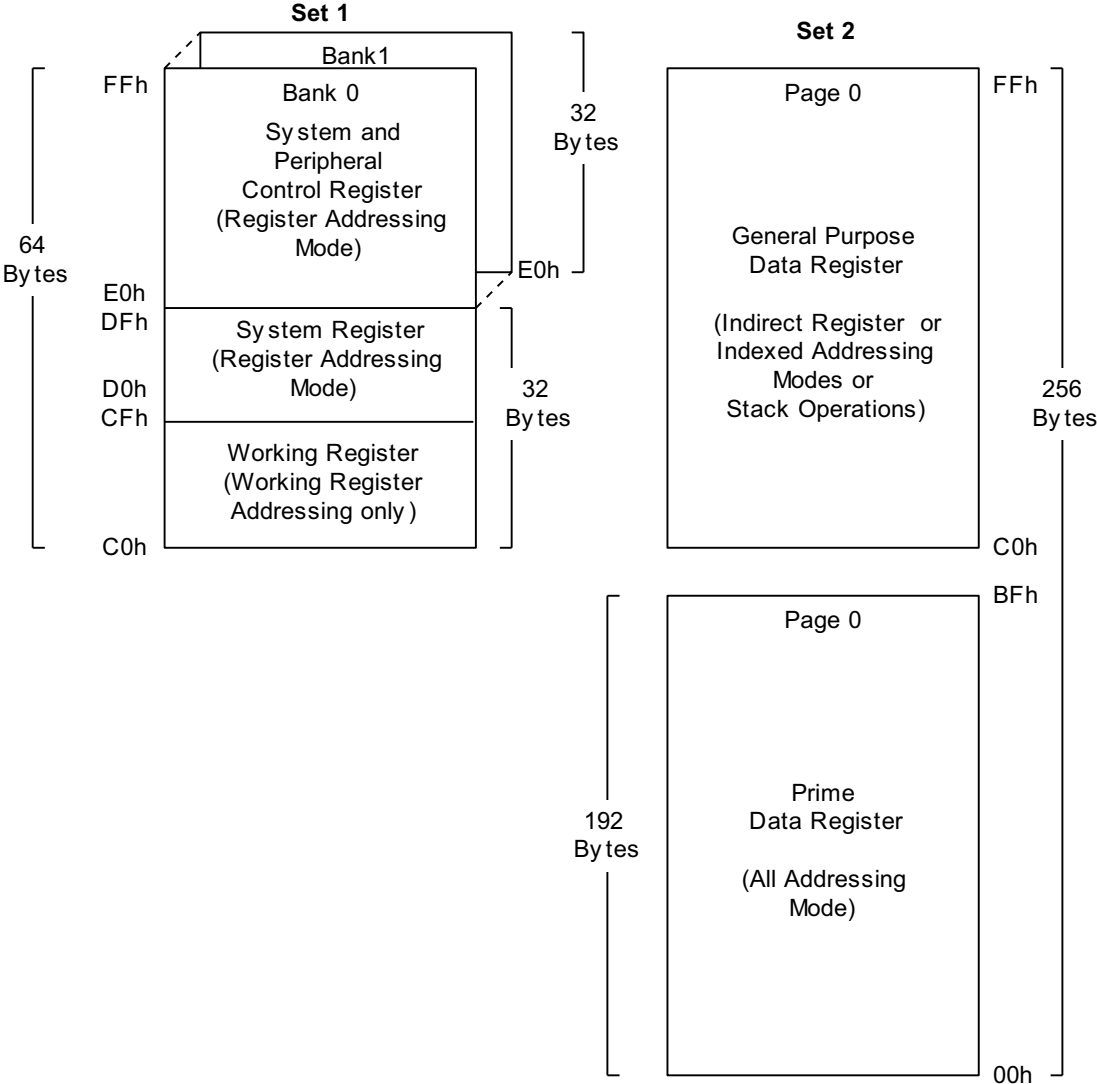


Figure 15. Internal Register File Organization

2.2.1. Register Page Pointer

The S3 Family architecture supports the logical expansion of the physical 333-byte internal register files (using an 8-bit data bus) into as many as 16 separately addressable register pages. Page addressing is controlled by the Register Page Pointer (PP; Set 1, Bank 0,

DFh). In the S3F80PB microcontroller, a paged register file expansion is not implemented, and the register page pointer settings therefore always point to Page 0.

Following a reset, the page pointer's source value (i.e., lower nibble) and destination value (i.e., upper nibble) are always 0000, automatically. Therefore, the S3F80PB MCU always selects Page 0 as the source and destination page for register addressing. These Page Pointer (PP) Register settings, as shown in Table 4, should not be modified during normal operation.

Table 4. Register Page Pointer (PP; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					DFh			

Note: R/W = read/write.

Bit	Description
[7:4]	Destination Register Page Selection Bits 0000: Destination: Page 0.*
[3:0]	Source Register Page Selection Bits 0000: Source: Page 0.*

Note: In the S3F80PB microcontroller, a hardware reset operation writes the 4-bit destination and source values shown above to the register page pointer. These values should not be modified to address other pages.

2.2.2. Register Set1

The term *Set1* refers to the upper 64 bytes of the register file, locations C0h–FFh.

The upper 32-byte area of this 64-byte space (E0h–FFh) is divided into two 32-byte register banks, Bank0 and Bank1. The Set Register's SB0 or SB1 bank instructions are used to address one bank or the other. In the S3F80PB microcontroller, Bank1 is implemented. A hardware reset operation always selects Bank0 addressing.

The upper two 32-byte area of Set1, Bank0, (E0h–FFh) contain 31 mapped system and peripheral control registers. Additionally, the upper 32-byte area of Set1, Bank1 (E0h–FFh) contains 16 mapped peripheral control registers. The lower 32-byte area contains 15 system registers (D0h–DFh) and a 16-byte common working register area (C0h–CFh). Use the common working register area as a scratch area for data operations being performed in other areas of the register file.

Registers in the Set1 location are directly accessible at all times using the Register Addressing Mode. The 16-byte working register area can only be accessed using working register addressing. To learn more about working register addressing, see the [Addressing Modes](#) chapter on page 36.

2.2.3. Register Set2

The same 64-byte physical space that is used for Set1 location C0h–FFh is logically duplicated to add another 64 bytes of register space. This expanded area of the register file is called *Set2*. The Set2 locations (C0h–FFh) are accessible on Page 0 in the S3F80PB MCU’s register space.

The logical division of Set1 and Set2 is maintained by means of addressing mode restrictions: Use only Register Addressing Mode to access Set1 locations; to access registers in Set2, you must use Register Indirect Addressing Mode or Indexed Addressing Mode. The Set2 register area is commonly used for stack operations.

2.2.4. Prime Register Space

The lower 192 bytes of the 256-byte physical internal register file (00h–BFh) are called the *prime register space* or, more simply, the *prime area*. Access the registers in this address using any addressing mode. (In other words, there is no addressing mode restriction for these registers, as is the case for Set1 and Set2 registers.) The prime register area on Page 0 is immediately addressable following a reset.

Figure 16 shows a map of the Set1, Set2, and prime register space.

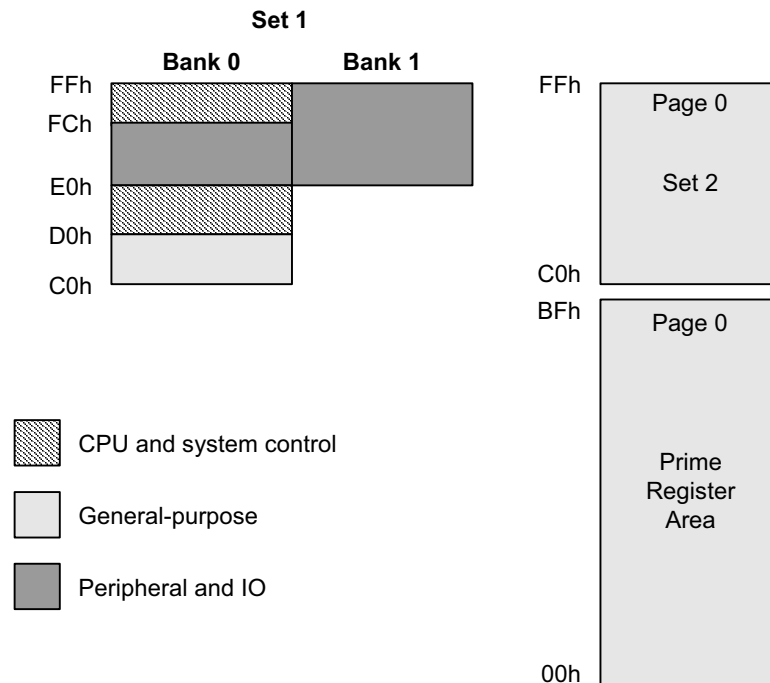


Figure 16. Set1, Set2, and Prime Area Register Map

2.2.5. Working Registers

Instructions can access specific 8-bit registers or 16-bit register pairs using either 4-bit or 8-bit address fields. When the 4-bit working register addressing is used, the 256-byte register file can be seen by the programmer as consisting 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, two working register slices can be selected at any one time to form a 16-byte working register block. Using the register pointers, move this 16-byte register block anywhere in the addressable register file, except for the Set2 area.

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

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

All of the registers in an 8-byte working register slice feature the same binary value for their five most significant address bits, thereby making it possible for each register pointer to point to one of the 24 slices in the register file. The base addresses for the two selected 8-byte register slices are contained in register pointers RP0 and RP1.

After a reset, RP0 and RP1 always point to the 16-byte common area in Set1 (C0h–CFh). Figure 17 illustrates the 8-byte working register areas (i.e., slices).

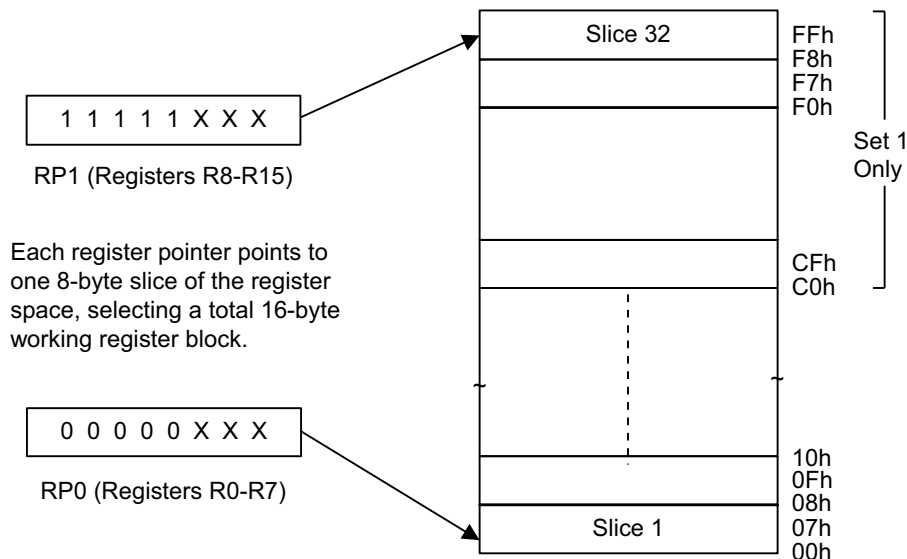


Figure 17. 8-Byte Working Register Areas

2.2.6. Using the Register Pointers

Register pointers RP0 and RP1, mapped to addresses D6h and D7h in Set1, are used to select two movable 8-byte working register slices in the register file. After a reset, they point to the working register common area: RP0 points to addresses C0h–C7h, and RP1 points to addresses C8h–CFh.

To change a register pointer value, load a new value to RP0 and/or RP1 using an SRP or LD instruction; see Figures 18 and 19.

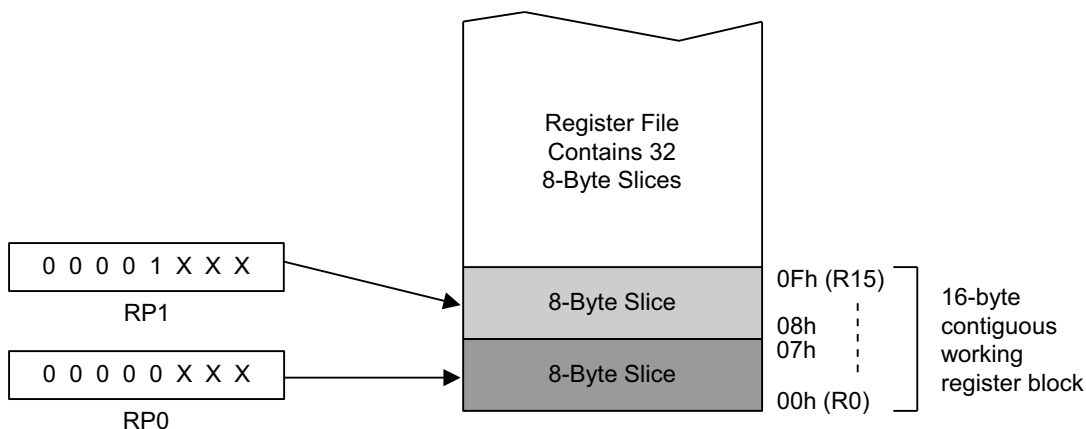


Figure 18. Contiguous 16-Byte Working Register Block

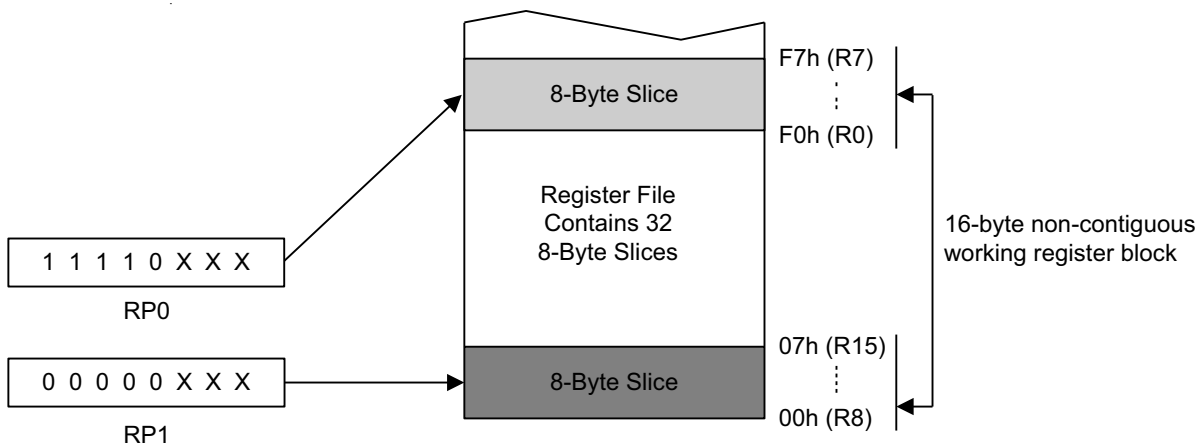


Figure 19. Noncontiguous 16-Byte Working Register Block

With working register addressing, only access those two 8-bit slices of the register file that are currently pointed to by RP0 and RP1. Register pointers cannot, however, be used to select working register spaces in Set2, C0h to FFh, because these locations can be accessed only using the Indirect Register or Indexed Addressing modes.

The selected 16-byte working register block usually consists of two contiguous 8-byte slices. As a general programming guideline, Zilog recommends that RP0 point to the lower slice and RP1 point to the upper slice; see [Figure 17](#) on page 28. In some cases, it can be necessary to define working register areas in different (i.e., noncontiguous) areas of the register file. In [Figure 18](#), RP0 points to the *upper* slice and RP1 to the *lower* slice.

Because a register pointer can point to the either of the two 8-byte slices in the working register block, define the working register area very flexibly to support program requirements, as indicated in the following two examples.

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
```

Using Register Pointers to Calculate the Sum of a Series of Registers

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

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

The sum of these six registers, 6Fh, is located in the register R0 (80h). The instruction string used in this example takes 12 bytes of instruction code and includes an execution time of 36 cycles. If the register pointer is not used to calculate the sum of these registers, the following instruction sequence must 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
```

As a result, the sum of the six registers is now located in register 80h. However, this instruction string requires 15 bytes of instruction code instead of 12 bytes, and its execution time is 50 cycles instead of 36 cycles.

2.3. Register Addressing

The S3 Family register architecture provides an efficient method of working register addressing that takes full advantage of shorter instruction formats to reduce execution time.

With Register (R) Addressing Mode, in which the operand value is the content of a specific register or register pair, access all locations in the register file except for Set2. With working register addressing, use a register pointer to specify an 8-byte working register space in the register file and an 8-bit register within that space.

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

Working register addressing differs from register addressing because 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; see Figures 20 and 21.

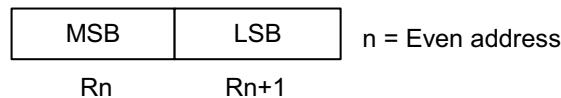


Figure 20. 16-Bit Register Pair

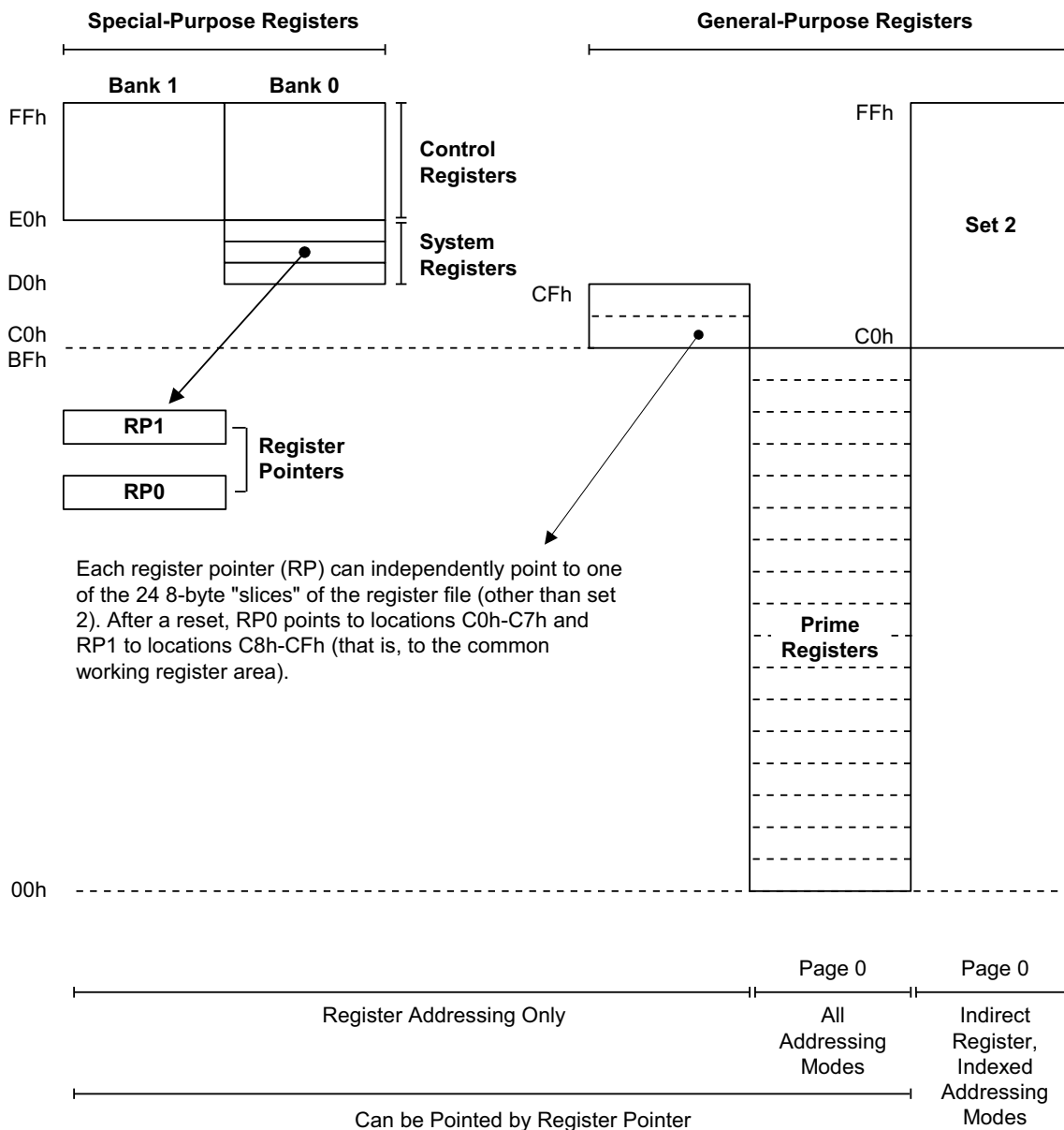


Figure 21. Register File Addressing

► **Note:** As shown in Figure 21, only Page 0 is implemented in the S3F80PB microcontroller. Page 0 contains all of the addressable registers in the internal register file.

2.3.1. Common Working Register Area

After a reset, register pointers RP0 and RP1 automatically select two 8-byte register slices in Set1, locations C0h to CFh, as the active 16-byte working register block, as shown below.

- RP0 → C0h–C7h
- RP1 → C8h–CFh

This 16-byte address range is called the *common working area*. Essentially, locations in this area can be used as working registers by operations that address any location on any page in the register file. Typically, these working registers serve as temporary buffers for data operations. See Figure 22.

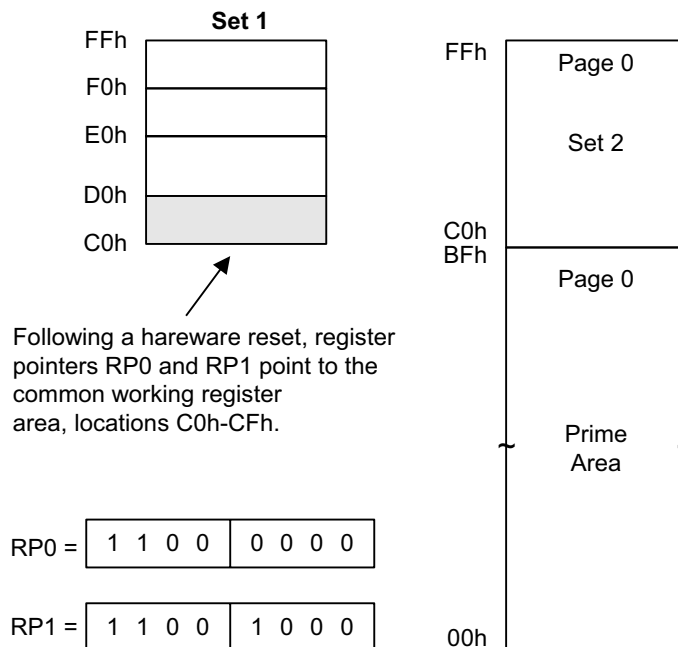


Figure 22. Common Working Register Area

2.3.1.1. Addressing the Common Working Register Area

As the following two examples show, working registers in the common area, locations C0h to CFh, are accessed using working Register Addressing Mode only.

Example 1

```
LD    0C2h, 40h    ; Invalid addressing mode!
```

Use working register addressing instead:

```
SRP   #0C0h
LD    R2, 40h      ; R2 (C2h) ← the value in location 40h
```

Example 2

```
ADD   0C3h, #45h   ; Invalid addressing mode!
```

Use working register addressing instead:

```
SRP   #0C0h
ADD   R3, #45h     ; R3 (C3h) ← R3 + 45h
```

2.3.2. 4-Bit Working Register Addressing

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

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

As shown in Figure 23, the result of this operation is that the five high-order bits from the register pointer are concatenated with the three low-order bits from the instruction address to form the complete address. As long as the address stored in the register pointer remains unchanged, the three bits from the address will always point to an address in the same 8-byte register slice.

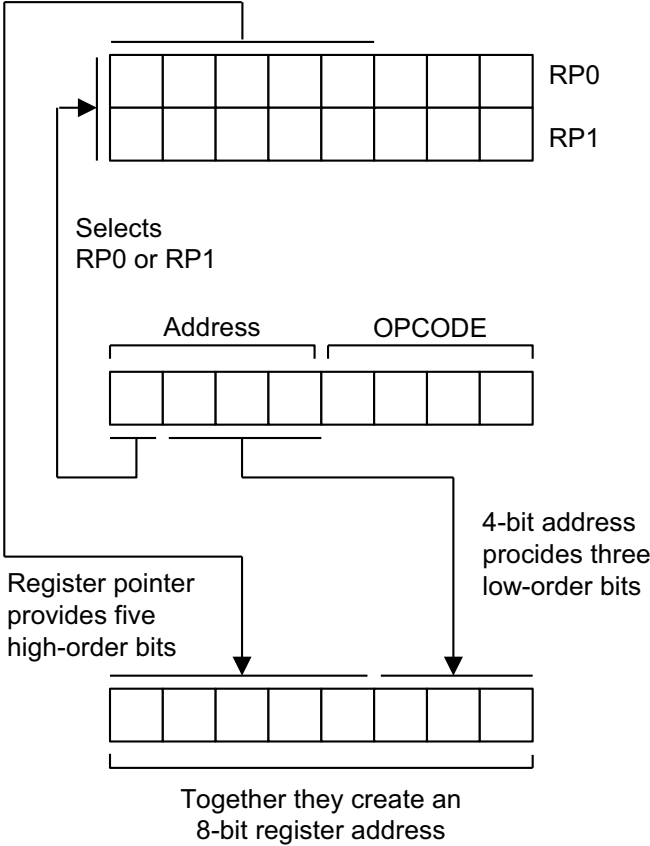


Figure 23. 4-Bit Working Register Addressing

Figure 24 shows a typical example of 4-bit working register addressing. The high-order bit of the instruction INC-R6 is 0, which selects RP0. The five high-order bits stored in RP0 (01110b) are concatenated with the three low-order bits of the instruction's 4-bit address (110b) to produce the register address 76h (01110110b).

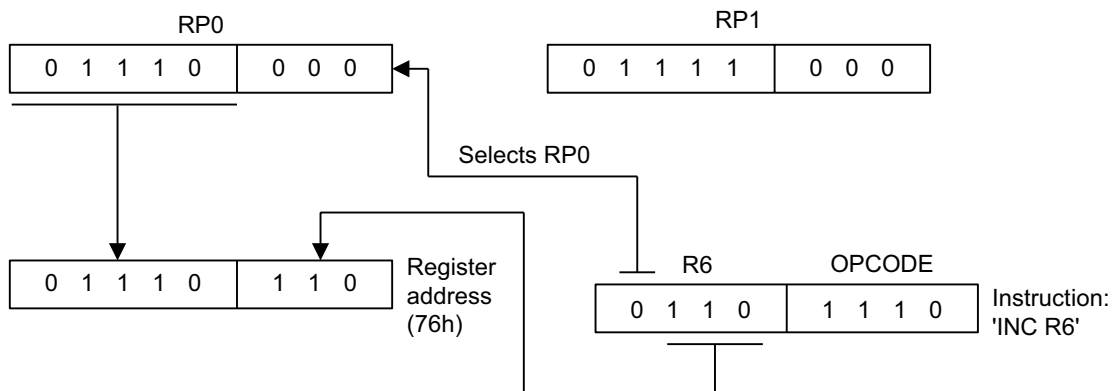


Figure 24. 4-Bit Working Register Addressing Example

2.3.3. 8-Bit Working Register Addressing

Use the 8-bit working register addressing to access registers in a selected working register area. To initiate 8-bit working register addressing, the upper four bits of the instruction address must contain the 4-bit value, 1100b. This value indicates that the remaining four bits contain the same effect as 4-bit working register addressing.

As shown in Figure 25, the lower nibble of the 8-bit address is concatenated in much the same way as for 4-bit addressing: Bit 3 selects either RP0 or RP1, which then supplies the five high-order bits of the final address. The three low-order bits of the complete address are provided by the original instruction.

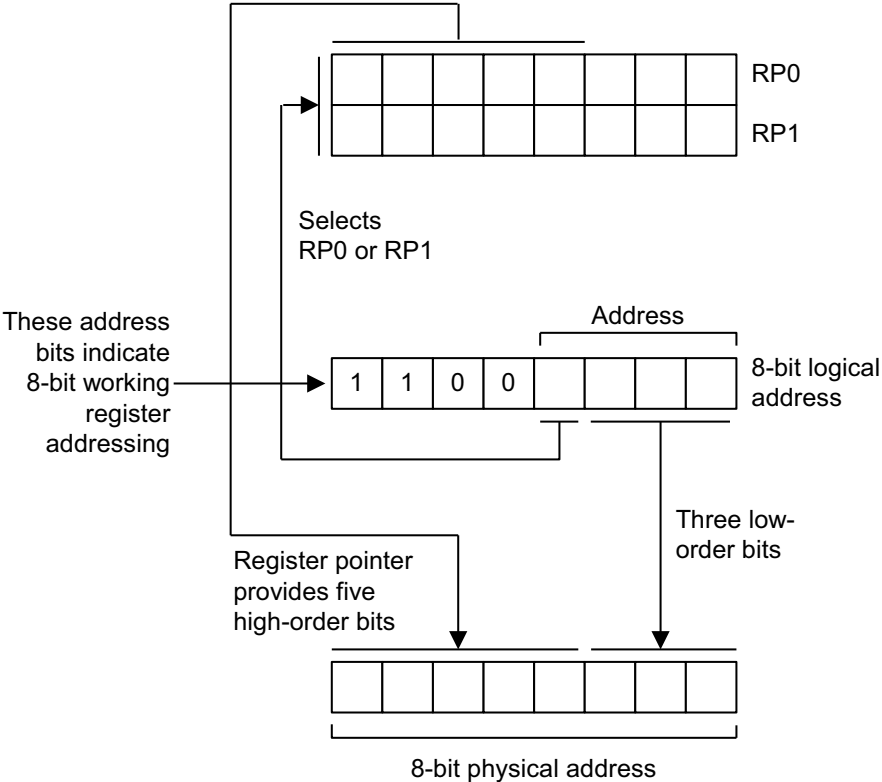


Figure 25. 8-Bit Working Register Addressing

Figure 26 shows an example of 8-bit working register addressing. The four high-order bits of the instruction address (1100b) specify 8-bit working register addressing. Bit 4 (1) selects RP1 and the five high-order bits in RP1 (10101b) become the five high-order bits of the register address. The three low-order bits of the register address (011) are provided by the three low-order bits of the 8-bit instruction address. The five-address bits from RP1 and the three address bits from the instruction are concatenated to form the complete register address, 0ABh (10101011b).

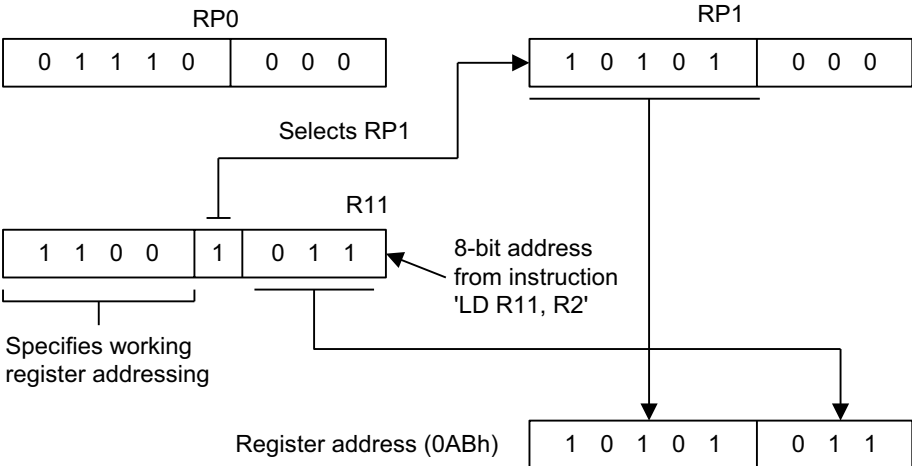


Figure 26. 8-Bit Working Register Addressing Example

2.4. Register Pointer Registers

The contents of the Register Pointer 0 (RP0) and Register Pointer 1 (RP1) registers are described in Tables 5 and 6.

Table 5. Register Pointer 0 (RP0; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	1	1	0	0	0	–	–	–
R/W	R/W	R/W	R/W	R/W	R/W	–	–	–
Address	D6h							

Note: R/W = read/write.

Bit	Description
[7:3]	Register Pointer 0 Address Value Register Pointer 0 can independently point to one of the 248-byte working register areas in the register file. Using the register pointers RP0 and RP1, select two 8-byte register slices at one time as an active working register space. After a reset, RP0 points to address C0h in register Set1, Bank0, selecting the 8-byte working register slice C0h–C7h.
[2:0]	Reserved These bits are reserved and must be set to 000.

Table 6. Register Pointer 1 (RP1; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	1	1	0	0	1	–	–	–
R/W	R/W	R/W	R/W	R/W	R/W	–	–	–
Address	D7h							

Note: R/W = read/write.

Bit	Description
[7:3]	Register Pointer 1 Address Value Register Pointer 1 can independently point to one of the 248-byte working register areas in the register file. Using the register pointers RP0 and RP1, select two 8-byte register slices at one time as active working register space. After a reset, RP1 points to address C8h in register Set1, Bank0, selecting the 8-byte working register slice C8h–CFh.
[2:0]	Reserved These bits are reserved and must be set to 000.

2.5. System and User Stacks

S3 Family microcontrollers use the system stack for subroutine calls and returns and to store data. The PUSH and POP instructions are used to control system stack operations. The S3F80PB architecture supports stack operations in the internal register file.

2.5.1. Stack Operations

Return addresses for procedure calls, interrupts and data are stored on the stack. The contents of the PC are saved to stack by a CALL instruction and restored by the RET instruction. When an interrupt occurs, the contents of the PC and the Flags registers are pushed to the stack. The IRET instruction then pops these values back to their original locations. The stack address value is always decreased by one before a push operation and increased by one after a pop operation. The Stack Pointer (SP) always points to the stack frame stored on the top of the stack, as shown in Figure 27.

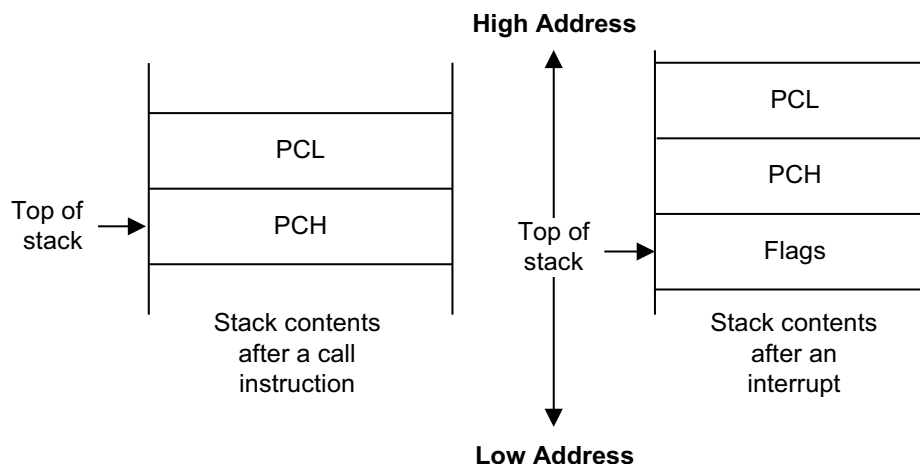


Figure 27. Stack Operations

2.5.2. User-Defined Stacks

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

2.5.3. Stack Pointers

Register location D9h contains the 8-bit stack pointer (SPL) that is used for system stack operations. After a reset, this SPL value is undetermined. Because only a 256-byte internal memory is implemented in the S3F80PB MCU, the SPL must be initialized to an 8-bit value in the range 00h–FFh.

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

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
```

2.6. Stack Pointer Register

The contents of the Stack Pointer Low Byte (SPL) Register are described in Table 7.

Table 7. Stack Pointer Low Byte (SPL; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	x	x	x	x	x	x	x	x
R/W					R/W			
Address					D9h			

Note: R/W = read/write.

Bit	Description
[7:0]	Stack Pointer Address Low Byte The SP value is undefined following a reset.

Chapter 3. Addressing Modes

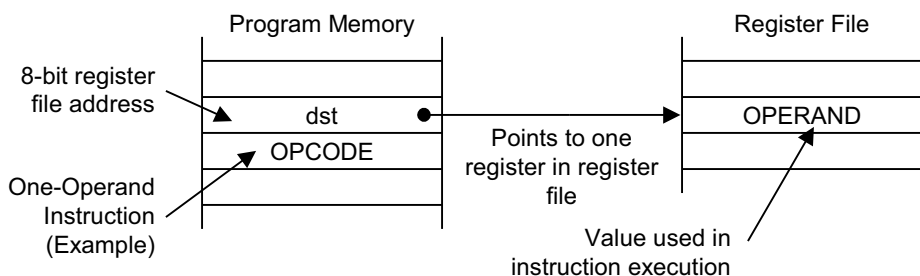
The program counter is used to fetch instructions that are stored in program memory for execution. Instructions indicate the operation to be performed and the data to be operated on. Addressing Mode is the method used to determine the location of the data operand. The operands specified in instructions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

The S3 Family instruction set supports the following seven explicit addressing modes; not all of these addressing modes are available for each instruction.

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

3.1. Register Addressing Mode

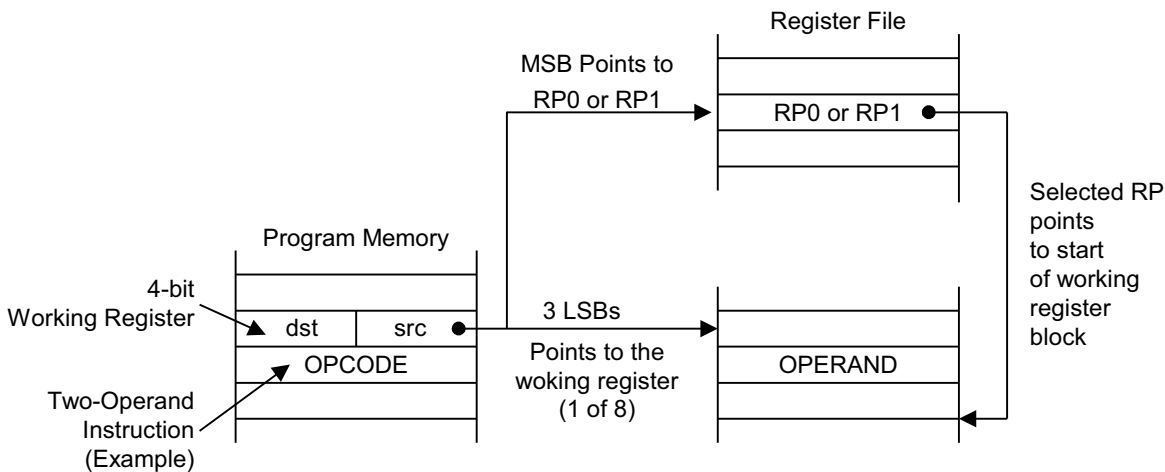
In Register Addressing Mode (R), the operand is the content of a specified register or register pair; see Figure 28. Working register addressing differs from register addressing because it uses a register pointer to specify an 8-byte working register space in the register file and an 8-bit register within that space; see Figure 29.



Sample Instruction:

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

Figure 28. Register Addressing



Sample Instruction:

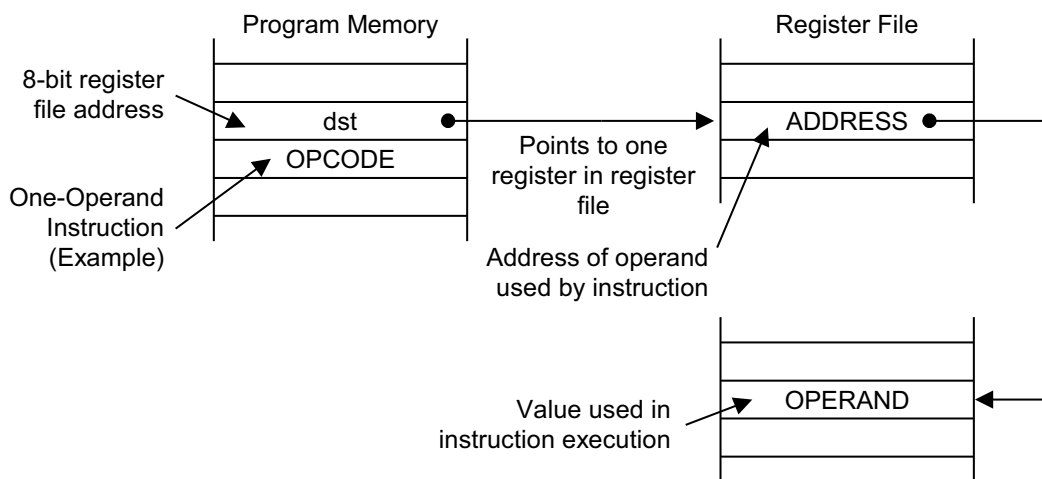
ADD R1, R2 ; Where R1 and R2 are registers in the currently selected working register area.

Figure 29. Working Register Addressing

3.2. Indirect Register Addressing Mode

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 can point to a register in the register file, to program memory (ROM), or to an external memory space, if implemented; see Figures 30 through 33.

Use any 8-bit register to indirectly address another register. Any 16-bit register pair can be used to indirectly address another memory location. Remember, however, that locations C0h–FFh in Set1 cannot be accessed using Indirect Register Addressing Mode.



Sample Instruction:

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

Figure 30. Indirect Register Addressing to Register File

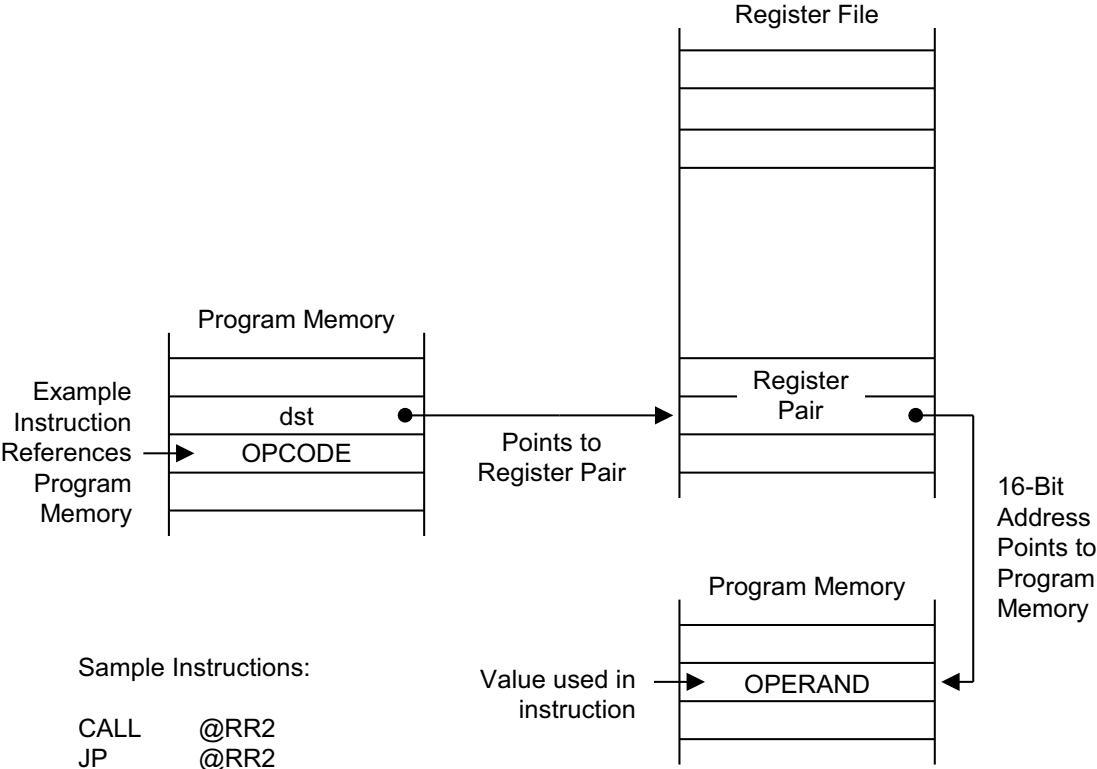


Figure 31. Indirect Register Addressing to Program Memory

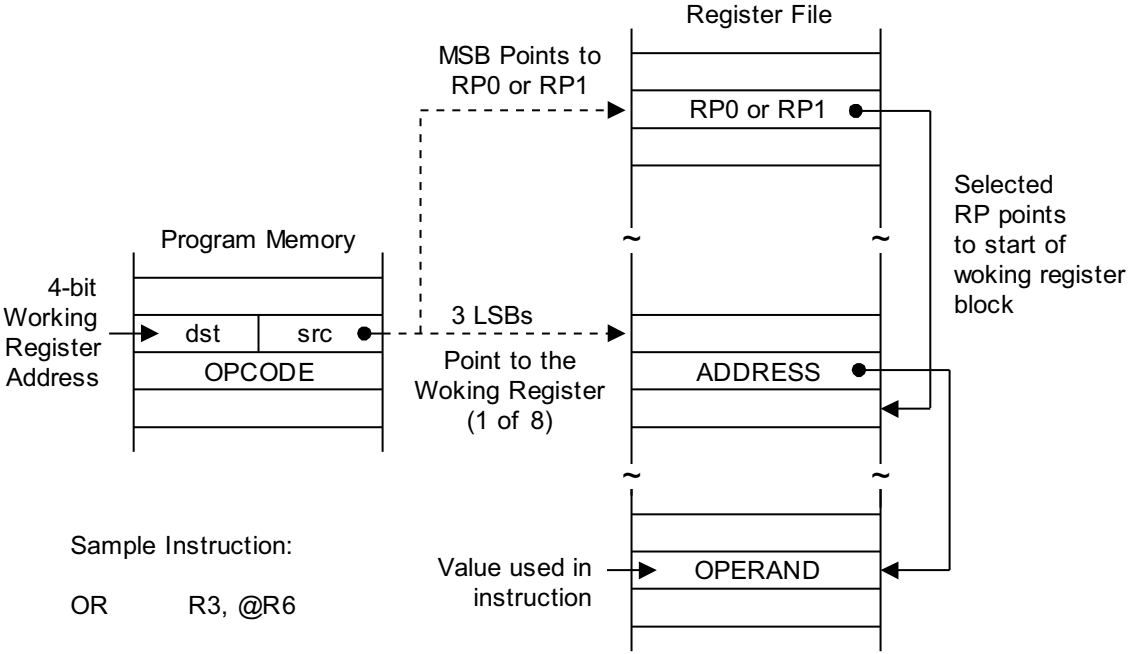
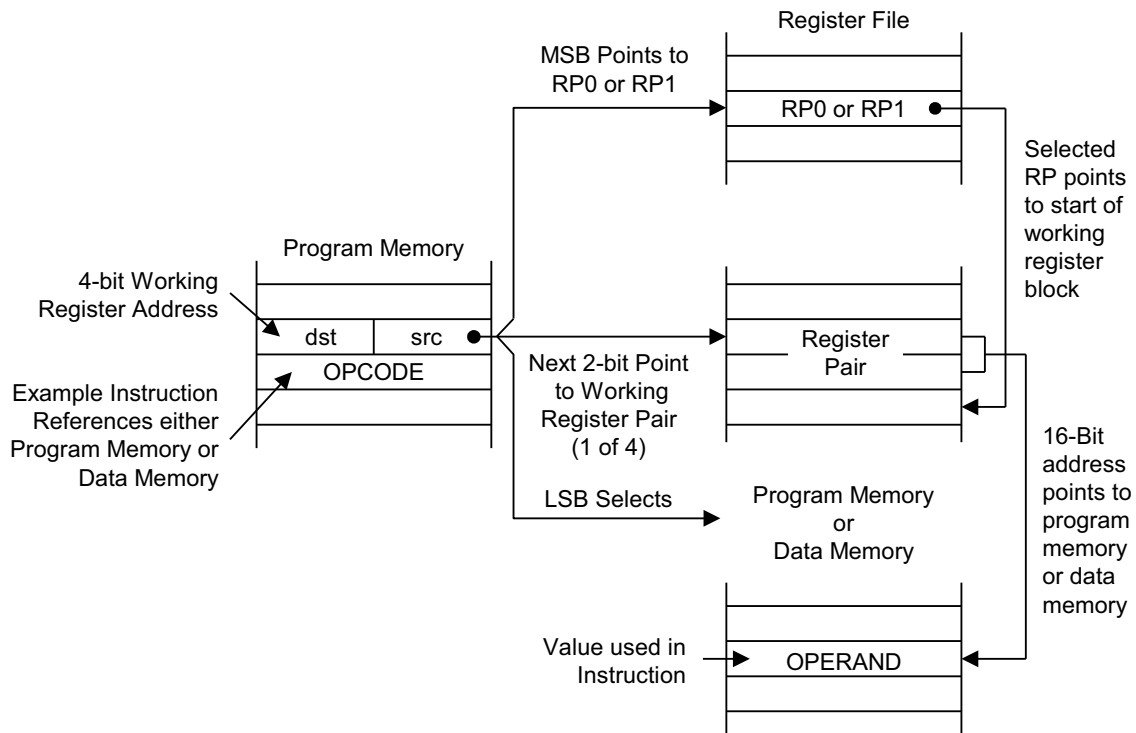


Figure 32. Indirect Register Addressing to Register File



Sample Instructions:

```
LCD    R5,@RR6      ; Program memory access
LDE    R3,@RR14     ; External data memory access
LDE    @RR4, R8     ; External data memory access
```

Figure 33. Indirect Register Addressing to Program or Data Memory

► **Note:** The LDE command is not available because an external interface is not implemented for the S3F80PB MCU.

3.3. Indexed Addressing Mode

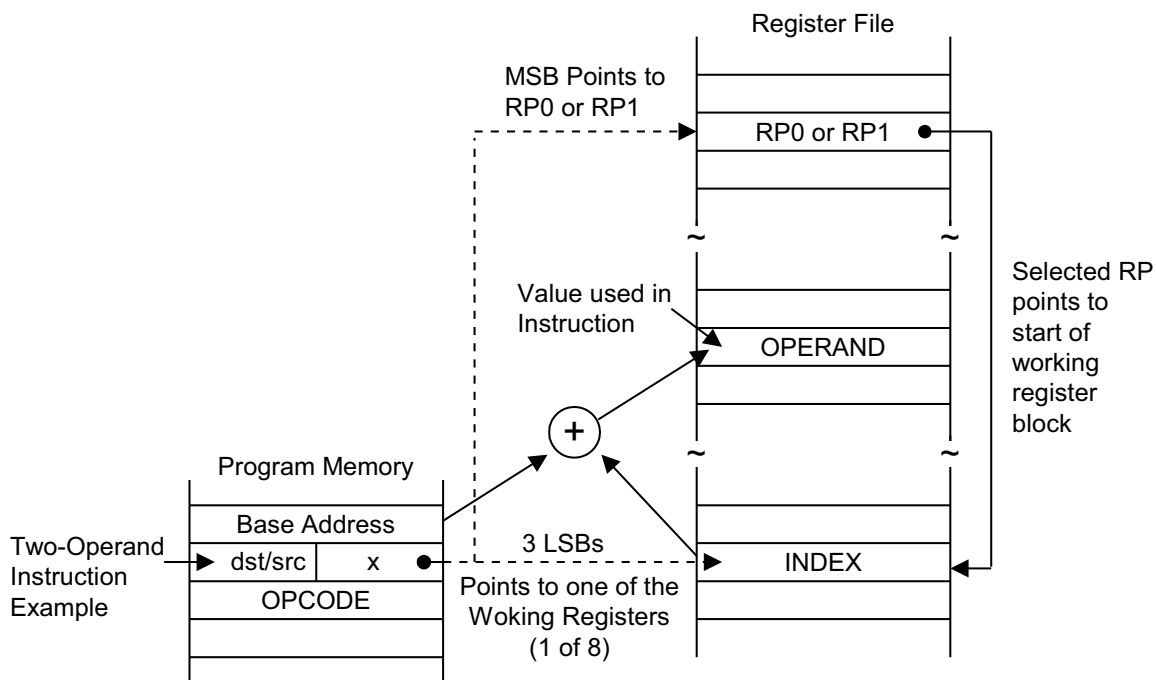
Indexed (X) Addressing Mode adds an offset value to a base address during the instruction execution to calculate the effective operand address; see [Figure 34](#) on page 49. Use Indexed Addressing Mode to access locations in the internal register file or in external

memory (if implemented). The C0h–FFh locations in Set1 cannot be selected using indexed addressing.

In short offset Indexed Addressing Mode, the 8-bit displacement is treated as a signed integer in the range –128 to +127. This displacement applies to external memory accesses only; see [Figure 35](#) on page 50.

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 provided in the instruction is then added to the base address; see [Figure 36](#) on page 51.

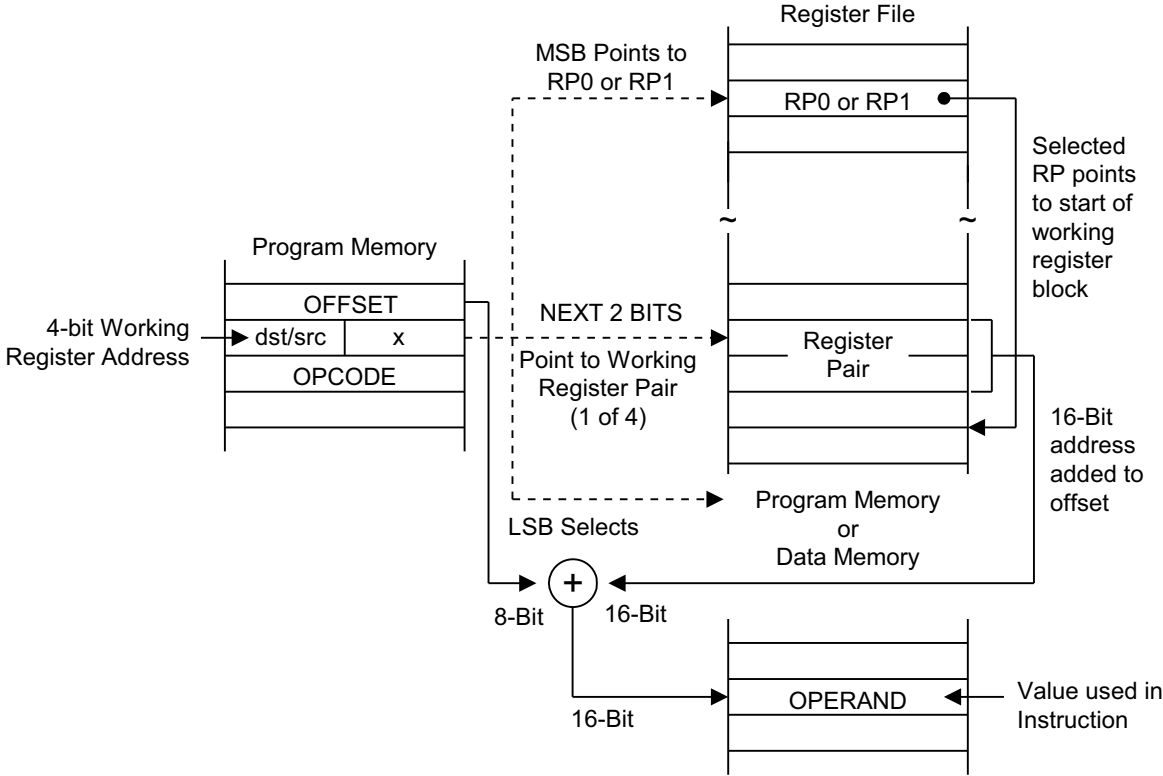
The only instruction that supports Indexed Addressing Mode for the internal register file is the Load instruction (LD). The LDC and LDE instructions support Indexed Addressing Mode for internal program memory and for external data memory (if implemented).



Sample Instruction:

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

Figure 34. Indexed Addressing to Register File

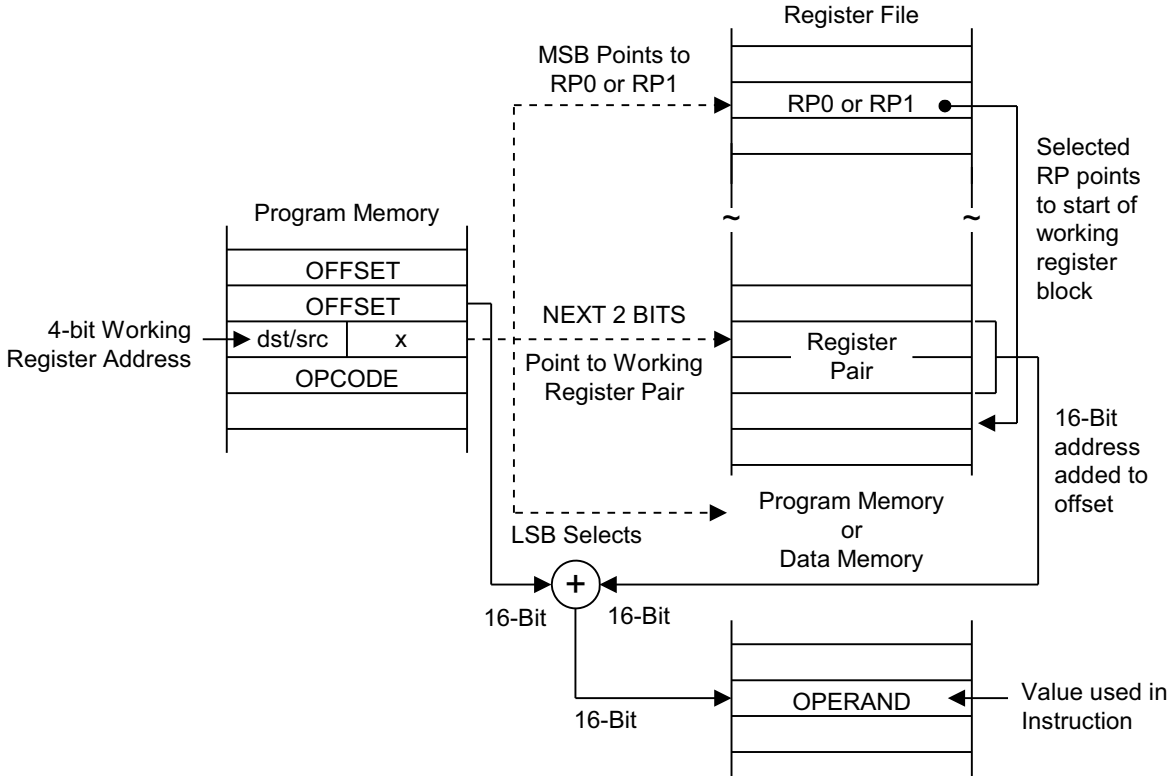


Sample Instructions:

```
LDC    R4, #04h[RR2]    ; The values in the program address (RR2 + 04h)
                        ; are loaded into register R4.
LDE    R4, #04h[RR2]    ; Identical operation to LDC example, except that
                        ; external data memory is accessed.
```

Figure 35. Indexed Addressing to Program or Data Memory with Short Offset

► **Note:** LDE command is not available because an external interface is not implemented for the S3F80PB MCU.



Sample Instructions:

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

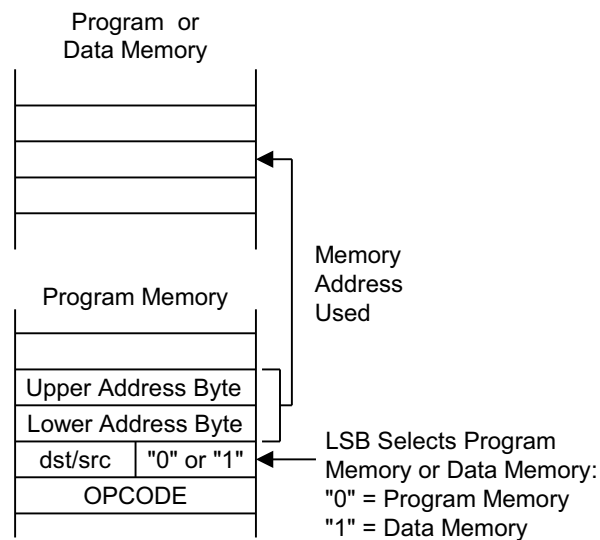
Figure 36. Indexed Addressing to Program or Data Memory

► **Note:** LDE command is not available because an external interface is not implemented for the S3F80PB MCU.

3.4. Direct Address Mode

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

The LDC and LDE instructions can use Direct Address Mode to specify the source or destination address for Load operations to program memory (LDC) or to external data memory (LDE), if implemented. See Figures 37 and 38.

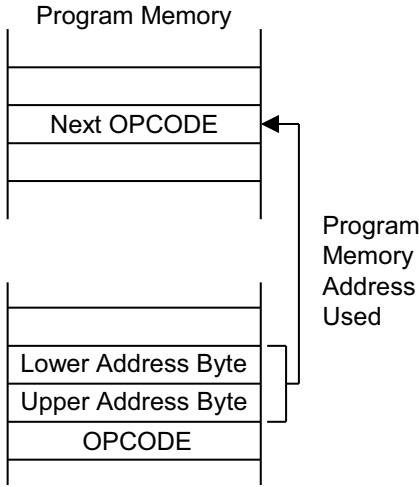


Sample Instructions:

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

Figure 37. Direct Addressing for Load Instructions

► **Note:** The LDE command is not available because an external interface is not implemented for the S3F80PB MCU.



Sample Instructions:

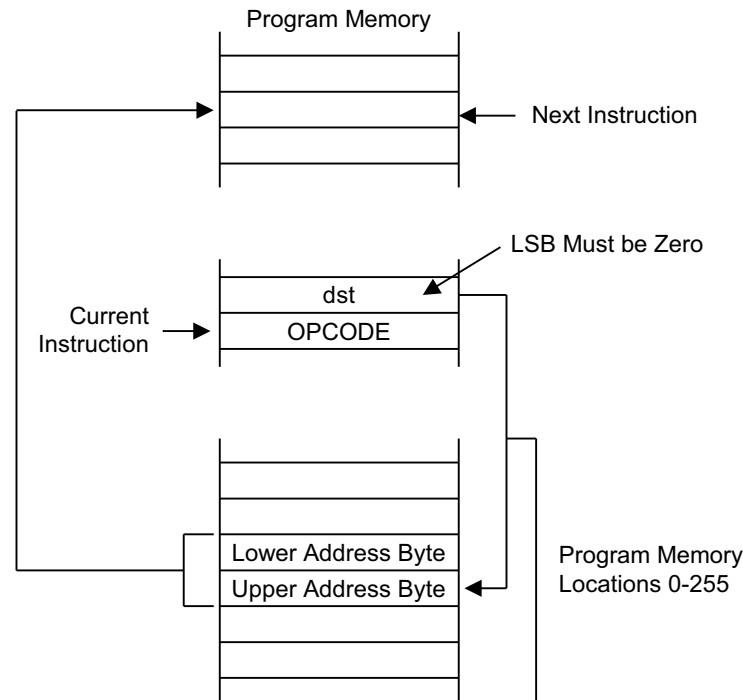
```
JP      C, JOB1          ; Where JOB1 is a 16-bit immediate address  
CALL   DISPLAY          ; Where DISPLAY is a 16-bit immediate address
```

Figure 38. Direct Addressing for Call and Jump Instructions

3.5. Indirect Address Mode

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 Indirect Address Mode.

Because Indirect Address Mode assumes that the operand is located in the lowest 256 bytes of program memory, only an 8-bit address is supplied in the instruction; the upper bytes of the destination address are assumed to all be zeros. See Figure 39.



Sample Instruction:

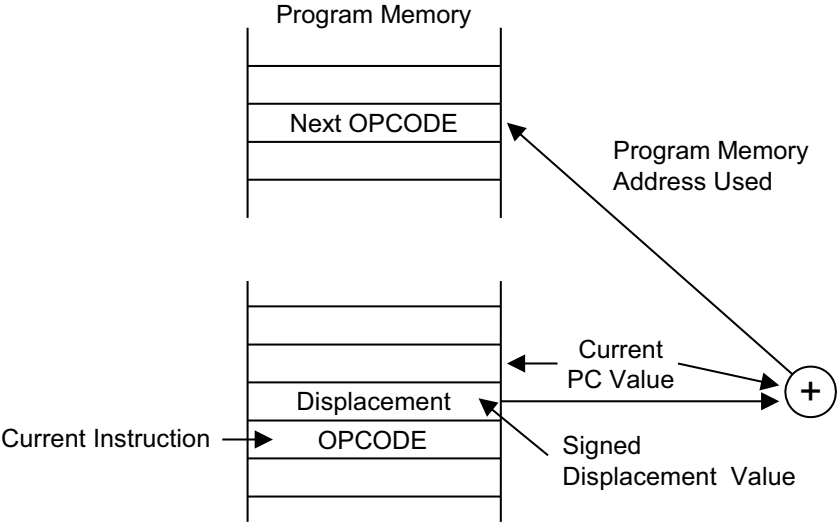
```
CALL    #40h    ; The 16-bit value in program memory addresses 40h
                and 41h is the subroutine start address.
```

Figure 39. Indirect Addressing

3.6. Relative Address Mode

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

Several program control instructions use the Relative Address Mode to perform conditional jumps. The instructions that support RA addressing are BTJRF, BTJRT, DJNZ, CPIJE, CPIJNE, and JR. See Figure 40.



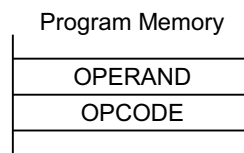
Sample Instructions:

```
JR    ULT,$+OFFSET    ; Where OFFSET is a value in the range +127 to -128
```

Figure 40. Relative Addressing

3.7. Immediate Mode

In Immediate (IM) Mode, the operand value used in the instruction is the value supplied in the operand field itself. The operand can be one byte or one word in length, depending on the instruction used. Immediate Addressing Mode is useful for loading constant values into registers. See Figure 41.



(The operand value is in the instruction)

Sample Instruction:

```
LD R0,#0AAh
```

Figure 41. Immediate Addressing

Chapter 4. Control Registers

This chapter describes the S3F80PB MCU's control registers. Data and counter registers are not described in this chapter; information about all registers used by a specific peripheral is presented in the corresponding chapters.

Table 8 identifies the names, mnemonics, decimal and hex equivalents, and read/write settings of the Set1, Bank0 mapped registers.

Table 8. Set1, Bank0 Mapped Registers

Register Name	Mnemonic	Page #	Decimal	Hex	R/W
Timer 0 Counter	T0CNT		208	D0h	R*
Timer 0 Data	T0DATA	250	209	D1h	R/W
Timer 0 Control	T0CON	249	210	D2h	R/W
Basic Timer Control	BTCON	247	211	D3h	R/W
Clock Control	CLKCON	193	212	D4h	R/W
System Flags	FLAGS	88	213	D5h	R/W
Register Pointer 0	RP0	39	214	D6h	R/W
Register Pointer 1	RP1	39	215	D7h	R/W
Location D8h is not mapped.					
Stack Pointer Low Byte	SPL	42	217	D9h	R/W
Instruction Pointer High Byte	IPH	80	218	DAh	R/W
Instruction Pointer Low Byte	IPL	80	219	DBh	R/W
Interrupt Request	IRQ	76	220	DCh	R*
Interrupt Mask	IMR	73	221	DDh	R/W
System Mode	SYM	71	222	DEh	R/W
Register Page Pointer	PP	26	223	DFh	R/W
Port 0 Data	P0	219	224	E0h	R/W
Port 1 Data	P1	219	225	E1h	R/W
Port 2 Data	P2	219	226	E2h	R/W
Port 3 Data	P3	219	227	E3h	R/W
Port 4 Data	P4	219	228	E4h	R/W
Port 2 Interrupt Enable	P2INT	231	229	E5h	R/W
Port 2 Interrupt Pending	P2PND	233	230	E6h	R/W
Port 0 Pull-Up Resistor Enable	P0PUR	224	231	E7h	R/W

Note: *A read-only register cannot be used as a destination for the OR, AND, LD, or LDB instructions.

Table 8. Set1, Bank0 Mapped Registers (Continued)

Register Name	Mnemonic	Page #	Decimal	Hex	R/W
Port 0 Control High Byte	P0CONH	220	232	E8h	R/W
Port 0 Control Low Byte	P0CONL	221	233	E9h	R/W
Port 1 Control High Byte	P1CONH	226	234	EAh	R/W
Port 1 Control Low Byte	P1CONL	227	235	EBh	R/W
Port 2 Control High Byte	P2CONH	229	236	ECh	R/W
Port 2 Control Low Byte	P2CONL	230	237	EDh	R/W
Port 2 Pull-Up Enable	P2PUR	234	238	EEh	R/W
Port 3 Control	P3CON	236	239	EFh	R/W
Port 4 Control	P4CON	240	240	F0h	R/W
Port 0 Interrupt Enable	P0INT	222	241	F1h	R/W
Port 0 Interrupt Pending	P0PND	223	242	F2h	R/W
Counter A Control	CACON	264	243	F3h	R/W
Counter A Data High Byte	CADATAH	265	244	F4h	R/W
Counter A Data Low Byte	CADATAL	265	245	F5h	R/W
Timer 1 Counter High Byte	T1CNTH	259	246	F6h	R*
Timer 1 Counter Low Byte	T1CNTL	259	247	F7h	R*
Timer 1 Data High Byte	T1DATAH	259	248	F8h	R/W
Timer 1 Data Low Byte	T1DATAL	260	249	F9h	R/W
Timer 1 Control	T1CON	258	250	FAh	R/W
Stop Control	STOPCON	208	251	FBh	W
Location FCh is not mapped.					
Basic Timer Counter	BTCNT		253	FDh	R*
External Memory Timing	EMT	60	254	FEh	R/W
Interrupt Priority	IPR	75	255	FFh	R/W

Note: *A read-only register cannot be used as a destination for the OR, AND, LD, or LDB instructions.

Table 9 identifies the names, mnemonics, decimal and hex equivalents, and read/write settings of the Bank1, Set1 mapped registers.

Table 9. Set1, Bank1 Mapped Registers

Register Name	Mnemonic	Page #	Decimal	Hex	R/W
LVD Control	LVDCON	297	224	E0h	R/W
Port 3 [4:5] Control	P345CON	239	225	E1h	R/W
Port 4 Control High Byte	P4CONH	241	226	E2h	R/W
Port 4 Control Low Byte	P4CONL	242	227	E3h	R/W
Timer 2 Counter High Byte	T2CNTH	273	228	E4h	R*
Timer 2 Counter Low Byte	T2CNTL	273	229	E5h	R*
Timer 2 Data High Byte	T2DATAH	273	230	E6h	R/W
Timer 2 Data Low Byte	T2DATAL	274	231	E7h	R/W
Timer 2 Control	T2CON	272	232	E8h	R/W
Location E9h is not mapped.					
Location EAh is not mapped.					
Location EBh is not mapped.					
Flash Memory Sector Address High Byte	FMSECH	283	236	ECh	R/W
Flash Memory Sector Address Low Byte	FMSECL	283	237	EDh	R/W
Flash Memory User Programming Enable	FMUSR	282	238	EEh	R/W
Flash Memory Control	FMCON	281	239	EFh	R/W
Reset Indicating	RESETID	216	240	F0h	R/W
LVD Flag Selection	LVDSSEL	298	241	F1h	R/W
Port 1 Output Mode Pull-Up Enable	P1OUTPU	228	242	F2h	R/W
Port 2 Output Mode Selection	P2OUTMD	232	243	F3h	R/W
Port 3 Output Mode Pull-Up Enable	P3OUTPU	238	244	F4h	R/W
Port 4 Output Mode Pull-Up Enable	P4OUTPU	243	245	F5h	R/W
Not mapped in the F6h to 0FFh address range.					
Note: *Read-only registers cannot be used as destinations for the OR, AND, LD, or LDB instructions.					

4.1. External Memory Timing Register

The contents of the External Memory Timing (EMT) Register are described in Table 10.

Table 10. External Memory Timing Register (EMT; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	1	1	1	1	1	0	–
R/W				R/W				–
Address								FEh

Note: R/W = read/write.

Bit	Description
[7]	External WAIT Input Function Enable Bit 0: Disable WAIT input function for external device. 1: Enable WAIT input function for external device.
[6]	Slow Memory Timing Enable Bit 0: Disable slow memory timing. 1: Enable slow memory timing.
[5:4]	Program Memory Automatic Wait Control Bits 00: No wait. 01: Wait one cycle. 10: Wait two cycles. 11: Wait three cycles.
[3:2]	Data Memory Automatic Wait Control Bits 00: No wait. 01: Wait one cycle. 10: Wait two cycles. 11: Wait three cycles.
[1]	Stack Area Selection Bit 0: Select internal register file area. 1: Select external data memory area.
[0]	Reserved This bit is reserved and must be set to 0.

Note: The EMT Register is not used in the S3F80PB MCU, because an external peripheral interface is not implemented. The program initialization routine should clear the EMT Register to 00h following a reset. Modification of EMT values during normal operation can cause a system malfunction.

Chapter 5. Interrupt Structure

The S3 Family interrupt structure features three basic components: levels, vectors, and sources. The SAM8RC CPU recognizes up to eight interrupt levels and supports up to 128 interrupt vectors. When a specific interrupt level contains more than one vector address, the vector priorities are established in hardware. A vector address can be assigned to one or more sources.

5.1. Levels

Interrupt levels are the main unit for interrupt priority assignment and recognition. All peripherals and I/O blocks can issue interrupt requests. In other words, peripheral and I/O operations are interrupt-driven. There are eight possible interrupt levels: IRQ0–IRQ7, also called *Level 0–Level 7*. Each interrupt level directly corresponds to an interrupt request number (IRQn). The total number of interrupt levels used in the interrupt structure varies from device to device. The S3F80PB interrupt structure recognizes eight interrupt levels (IRQ0–IRQ7) with hardware reset.

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

5.2. Vectors

Each interrupt level can contain one or more interrupt vectors, or it cannot contain a vector address at all. The maximum number of vectors that can be supported for an assigned level is 128. (The actual number of vectors used for the S3 Family devices is always much smaller.) If an interrupt level contains more than one vector address, the vector priorities are set in hardware. The S3F80PB MCU uses eighteen vectors. Two vector addresses are shared by four interrupt sources.

5.3. Sources

A source is any peripheral that generates an interrupt. For example, a source can be an external pin or a counter overflow. Each vector can contain several interrupt sources. In the S3F80PB interrupt structure, there are 24 possible interrupt sources.

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

5.4. Interrupt Types

The three components of the S3 Family interrupt structure described above—levels, vectors, and sources are combined to determine the interrupt structure of an individual device and to make full use of its available interrupt logic. There are three possible combinations of interrupt structure components, called *interrupt types 1, 2, and 3*. These three types differ in the number of vectors and interrupt sources assigned to each level; see Figure 42.

- Type 1: One level (IRQn) + one vector (V1) + one source (S1)
- Type 2: One level (IRQn) + one vector (V1) + multiple sources (S1–Sn)
- Type 3: One level (IRQn) + multiple vectors (V1–Vn) + multiple sources (S1–Sn, Sn + 1–Sn + m)

In the S3F80PB microcontroller, all three interrupt types are implemented.

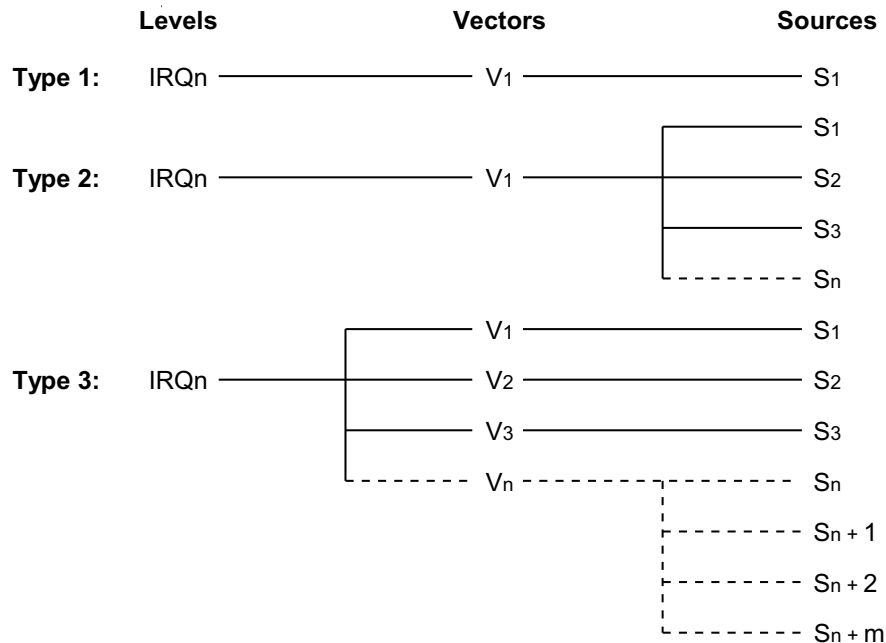


Figure 42. S3 Family Interrupt Types

► **Note:** The number of Sn and Vn value is expandable.

The S3F80PB microcontroller supports twenty-four interrupt sources. Sixteen of the interrupt sources contain a corresponding interrupt vector address; the remaining eight interrupt sources share two vector addresses. Eight interrupt levels are recognized by the CPU in this device-specific interrupt structure, as shown in Figure 43.

When multiple interrupt levels are active, the Interrupt Priority (IPR) Register 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.

Levels(9)	Vectors(18)	Sources(24)	Reset/Clear
RESET	100h	Basic timer overflow	H/W
IRQ0	FCh	Timer 0 match/capture	S/W
	FAh	Timer 0 overflow	H/W
IRQ1	F6h	Timer 1 match/capture	S/W
	F4h	Timer 1 overflow	H/W
IRQ2	ECh	Counter A	H/W
IRQ3	F2h	Timer 2 match/capture	S/W
	F0h	Timer 2 overflow	H/W
IRQ4	D6h	P2.3 external interrupt	S/W
	D4h	P2.2 external interrupt	S/W
	D2h	P2.1 external interrupt	S/W
	D0h	P2.0 external interrupt	S/W
IRQ5	D8h	P2.7 external interrupt	S/W
		P2.6 external interrupt	S/W
		P2.5 external interrupt	S/W
		P2.4 external interrupt	S/W
IRQ6	E6h	P0.3 external interrupt	S/W
	E4h	P0.2 external interrupt	S/W
	E2h	P0.1 external interrupt	S/W
	E0h	P0.0 external interrupt	S/W
IRQ7	E8h	P0.7 external interrupt	S/W
		P0.6 external interrupt	S/W
		P0.5 external interrupt	S/W
		P0.4 external interrupt	S/W

Figure 43. S3F80PB Interrupt Structure

► **Note:** The reset interrupt vector address (Basic Timer overflow) can be varied by the Smart Option.

5.5. Interrupt Vector Addresses

All interrupt vector addresses for the S3F80PB interrupt structure are stored in the vector address area of the internal program memory ROM, 00h–FFh; see Figure 44.

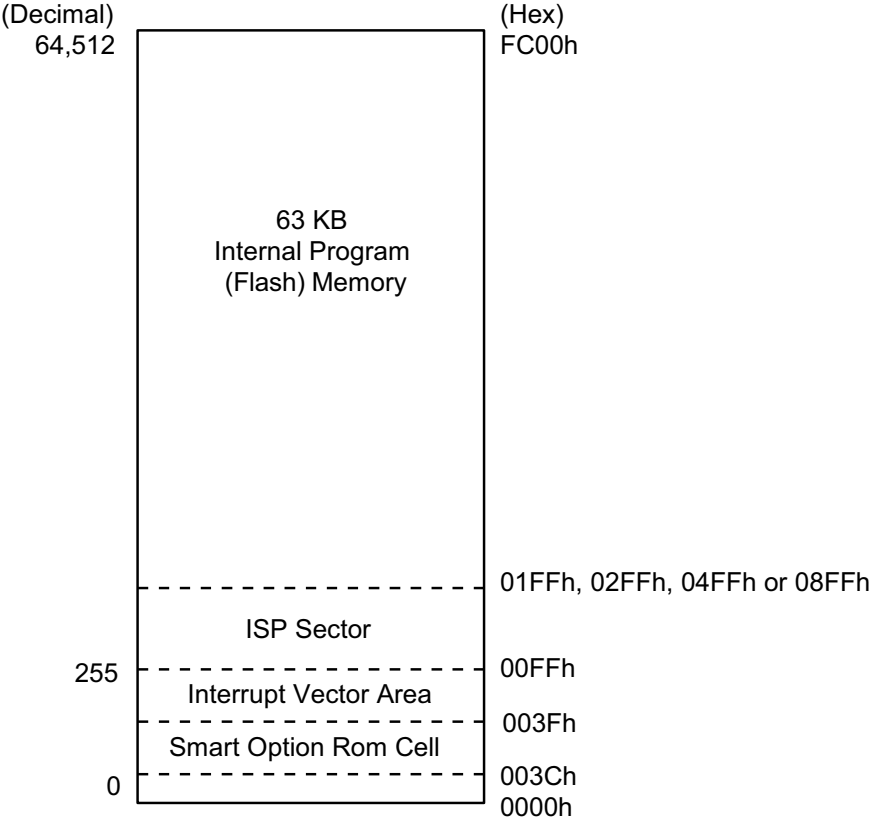


Figure 44. ROM Vector Address Area

Allocate unused locations in the vector address area as normal program memory. However, be careful not to overwrite any of the stored vector addresses, which are listed in Table 11.

Table 11. S3F80PB Interrupt Vectors

Vector Address		Interrupt Source	Request		Reset/Clear	
Decimal Value	Hex Value		Interrupt Level	Priority in Level	Hardware	Software
256	100h	Basic Timer overflow/POR	RESET	–	√	–
252	FCh	Timer 0 match/capture	IRQ0	1	–	√
250	FAh	Timer 0 overflow	–	0	√	–
246	F6h	Timer 1 match/capture	IRQ1	1	–	√
244	F4h	Timer 1 overflow	–	0	–	–
236	ECh	Counter A	IRQ2	–	–	–
242	F2h	Timer 2 match/capture	IRQ3	1	–	√
240	F0h	Timer 2 overflow	–	0	–	–
232	E8h	P0.7 external interrupt	IRQ7	–	–	√
232	E8h	P0.6 external interrupt	–	–	–	√
232	E8h	P0.5 external interrupt	–	–	–	√
232	E8h	P0.4 external interrupt	–	–	–	√
230	E6h	P0.3 external interrupt	IRQ6	3	–	√
228	E4h	P0.2 external interrupt	–	2	–	√
226	E2h	P0.1 external interrupt	–	1	–	√
224	E0h	P0.0 external interrupt	–	0	–	√
216	D8h	P2.7 external interrupt	IRQ5	–	–	√
216	D8h	P2.6 external interrupt	–	–	–	√
216	D8h	P2.5 external interrupt	–	–	–	√
216	D8h	P2.4 external interrupt	–	–	–	√
214	D6h	P2.3 external interrupt	IRQ4	3	–	√
212	D4h	P2.2 external interrupt	–	2	–	√
210	D2h	P2.1 external interrupt	–	1	–	√
208	D0h	P2.0 external interrupt	–	0	–	√

Notes:

1. Interrupt priorities are identified in inverse order: 0 is highest priority, 1 is the next highest, etc.
2. If two or more interrupts are within the same interrupt level, the interrupt with the lowest vector address usually receives priority over an interrupt with a higher vector address. The priorities within a assigned interrupt level are fixed in hardware.
3. A Reset (i.e., a reset of the basic timer overflow or a POR) interrupt vector address can be changed by the Smart Option; see [Figure 14](#) on page 22.

The program reset address in ROM is 0100h. This reset address can be changed using the Smart Option; see [Figure 14](#) on page 22 and/or the [Embedded Flash Memory Interface](#) chapter on page 277.

5.6. Enable/Disable Interrupt Instructions

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

► **Note:** The system initialization routine that is executed following a reset must always contain an EI instruction to globally enable the interrupt structure.

During normal operation, execute the Disable Interrupts (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. Although it is possible to manipulate SYM.0 directly to enable or disable interrupts, Zilog recommends using the EI and DI instructions instead.

5.6.0.1. System-Level Interrupt Control Registers

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

- The Interrupt Mask (IMR) Register enables (unmasks) or disables (masks) interrupt levels
- The Interrupt Priority (IPR) Register controls the relative priorities of interrupt levels
- The Interrupt Request (IRQ) Register contains interrupt pending flags for each interrupt level (as opposed to each interrupt source)
- The System Mode (SYM) Register enables or disables global interrupt processing (SYM settings also enable fast interrupts and control the activity of external interface, if implemented)

A summary of these interrupt control registers is provided in Table 12.

Table 12. Interrupt Control Register Overview

Control Register	Mnemonic	R/W	Function Description
Interrupt Mask Register	IMR	R/W	Bit settings in the IMR Register enable or disable interrupt processing for each of the eight interrupt levels: IRQ0–IRQ7.
Interrupt Priority Register	IPR	R/W	Controls the relative processing priorities of the interrupt levels. The eight levels of the S3F80PB MCU are organized into three groups: A, B and C. Group A is IRQ0 and IRQ1, group B is IRQ2, IRQ3, IRQ4 and group C is IRQ5, IRQ6 and IRQ7.
Interrupt Request Register	IRQ	R	This register contains a request pending bit for each interrupt level.
System Mode Register	SYM	R/W	A dynamic global interrupt processing enables/disables, fast interrupt processing, and external interface control; an external memory interface is not implemented in the S3F80PB microcontroller.

5.7. Interrupt Processing Control Points

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

- Global interrupt enable and disable by EI and DI instructions or by a 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

Figure 45 diagrams the functions of these combined interrupt processes.

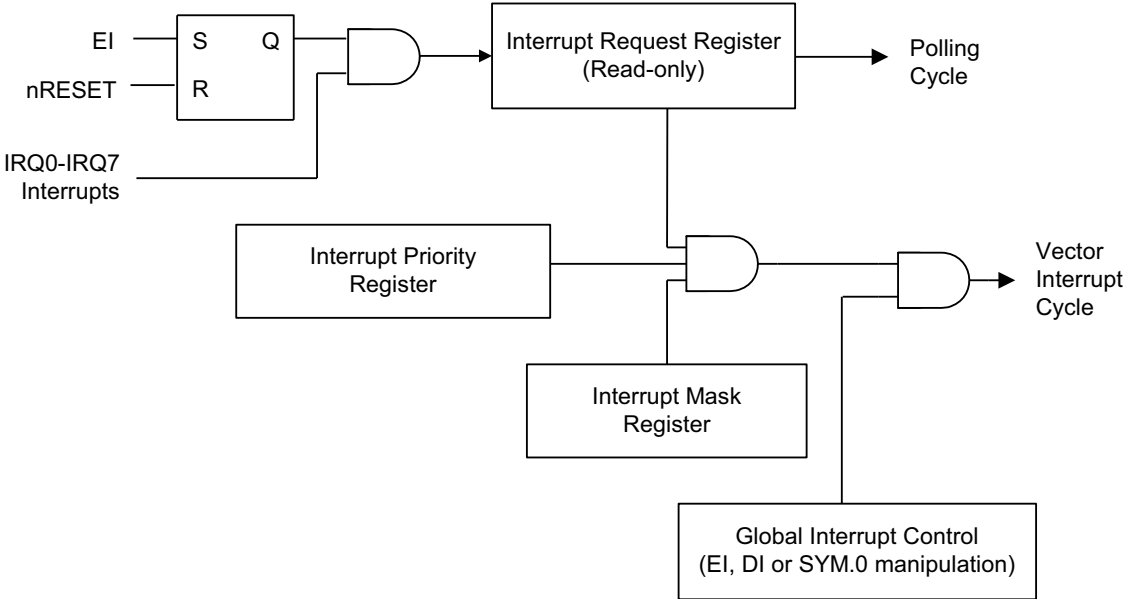


Figure 45. Interrupt Function Diagram

► **Note:** When writing the part of your application program that handles the processing of interrupts, be sure to include the necessary register file address (i.e., register pointer) information.

5.8. Peripheral Interrupt Control Registers

For each interrupt source there is one or more corresponding peripheral control registers that let you control the interrupt generated by that peripheral; see Table 13.

Table 13. Vectored Interrupt Source Control and Data Registers

Interrupt Source	Interrupt Level	Register(s)	Location(s) in Set1	Bank
Timer 0 match/capture or Timer 0 overflow	IRQ0	T0CON ¹ T0DATA	D2h D1h	Bank0
Timer 1 match/capture or Timer 1 overflow	IRQ1	T1CON ¹ T1DATAH, T1DATAL	FAh F8h, F9h	Bank0
Counter A	IRQ2	CACON CADATAH, CADATAL	F3h F4h, F5h	Bank0
Timer 2 match/capture or Timer 2 overflow	IRQ3	T2CON ¹ T2DATAH, T2DATAL	E8h E6h, E7h	Bank1
P0.7 external interrupt P0.6 external interrupt P0.5 external interrupt P0.4 external interrupt	IRQ7	P0CONH P0INT P0PND	E8h F1h F2h	Bank0
P0.3 external interrupt P0.2 external interrupt P0.1 external interrupt P0.0 external interrupt	IRQ6	P0CONL P0INT P0PND	E9h F1h F2h	Bank0
P2.7 external interrupt P2.6 external interrupt P2.5 external interrupt P2.4 external interrupt	IRQ5	P2CONH P2INT P2PND	ECh E5h E6h	Bank0
P2.3 external interrupt P2.2 external interrupt P2.1 external interrupt P2.0 external interrupt	IRQ4	P2CONL P2INT P2PND	EDh E5h E6h	Bank0

Note: Because the Timer 0, Timer 1, and Timer 2 overflow interrupts are cleared by hardware, the T0CON, T1CON and T2CON registers control only the enable/disable functions. The T0CON, T1CON and T2CON registers contain enable/disable and pending bits for the Timer 0, Timer 1, and Timer 2 match/capture interrupts, respectively. If an interrupt is unmasked (i.e., at an enable interrupt level) in the IMR Register, the pending bit and enable bit of the interrupt should be written after a DI instruction is executed.

5.9. System Mode Register

The System Mode (SYM) Register (DEh, Set1, Bank0) is used to globally enable and disable interrupt processing and to control fast interrupt processing; see Table 14.

A reset clears SYM.7, SYM.1 and SYM.0 to 0. The 3-bit value, SYM.4–SYM.2, is for fast interrupt level selection and undetermined values after reset. SYM.6 and SYM.5 are not used.

The instructions EI and DI enable and disable global interrupt processing, respectively, by modifying the bit 0 value of the SYM Register. An Enable Interrupts (EI) instruction must be included in the initialization routine, which follows a reset operation to enable interrupt processing. Although it is possible to manipulate SYM.0 directly to enable and disable interrupts during normal operation, Zilog recommends using the EI and DI instructions for this purpose.

Table 14. System Mode Register (SYM; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	–	–	x	x	x	0	0
R/W	R/W	–	–	R/W	R/W	R/W	R/W	R/W
Address	DEh							

Note: R/W = read/write.

Bit	Description
[7]	Tri-State External Interface Control Bit¹ 0: Normal operation (i.e., disable tri-state operation). 1: Set external interface lines to high impedance (i.e., enable tri-state operation).
[6:5]	Reserved These bits are reserved and must be set to 00. ²

Note:

1. Because an external interface is not implemented for the S3F80PB MCU, SYM.7 must always be 0. When the Stop Mode is released, the Stop Control (STOPCON) Register value is cleared automatically.
2. Although the SYM Register is not used, SYM.5 should always be 0. If you accidentally write a 1 to this bit during normal operation, a system malfunction can occur.
3. Select only one interrupt level at a time for fast interrupt processing.
4. Setting SYM.1 to 1 enables fast interrupt processing for the interrupt level currently selected by SYM.2–SYM.4.
5. Following a reset, you must enable global interrupt processing by executing an EI instruction (not by writing a 1 to SYM.0).

Bit	Description (Continued)
[4:2]	Fast Interrupt Level Selection Bits³ 000: IRQ0. 001: IRQ1. 010: IRQ2. 011: IRQ3. 100: IRQ4. 101: IRQ5. 110: IRQ6. 111: IRQ7.
[1]	Fast Interrupt Enable Bit⁴ 0: Disable fast interrupt processing. 1: Enable fast interrupt processing.
[0]	Global Interrupt Enable Bit⁵ 0: Disable global interrupt processing. 1: Enable global interrupt processing.

Note:

1. Because an external interface is not implemented for the S3F80PB MCU, SYM.7 must always be 0. When the Stop Mode is released, the Stop Control (STOPCON) Register value is cleared automatically.
2. Although the SYM Register is not used, SYM.5 should always be 0. If you accidentally write a 1 to this bit during normal operation, a system malfunction can occur.
3. Select only one interrupt level at a time for fast interrupt processing.
4. Setting SYM.1 to 1 enables fast interrupt processing for the interrupt level currently selected by SYM.2–SYM.4.
5. Following a reset, you must enable global interrupt processing by executing an EI instruction (not by writing a 1 to SYM.0).

5.10. Interrupt Mask Register

The Interrupt Mask (IMR) Register (DDh, Set1, Bank0) 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, etc. When the IMR bit of an interrupt level is cleared to 0, interrupt processing for that level is disabled (i.e., masked). When a level's IMR bit is set to 1, interrupt processing for the level is enabled (i.e., not masked).

The IMR Register is mapped to register location DDh in set1 and Bank0. Bit values can be read and written by instructions using the Register Addressing Mode. See Table 15.

Table 15. Interrupt Mask Register (IMR; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	x	x	x	x	x	x	x	x
R/W					R/W			
Address					DDh			

Note: R/W = read/write.

Bit	Description
[7]	Interrupt Level 7 (IRQ7) Enable Bit; External Interrupts P0.7–P0.4 0: Disable (Mask). 1: Enable (Unmask).
[6]	Interrupt Level 6 (IRQ6) Enable Bit; External Interrupts P0.3–P0.0 0: Disable (Mask). 1: Enable (Unmask).
[5]	Interrupt Level 5 (IRQ5) Enable Bit; External Interrupts P2.7–P2.4 0: Disable (Mask). 1: Enable (Unmask).
[4]	Interrupt Level 4 (IRQ4) Enable Bit; External Interrupts P2.3–P2.0 0: Disable (Mask). 1: Enable (Unmask).
[3]	Interrupt Level 3 (IRQ3) Enable Bit; Timer 2 Match/Capture or Overflow 0: Disable (Mask). 1: Enable (Unmask).
[2]	Interrupt Level 2 (IRQ2) Enable Bit; Counter A Interrupt 0: Disable (Mask). 1: Enable (Unmask).
[1]	Interrupt Level 1 (IRQ1) Enable Bit; Timer 1 Match/Capture or Overflow 0: Disable (Mask). 1: Enable (Unmask).
[0]	Interrupt Level 0 (IRQ0) Enable Bit; Timer 0 Match/Capture or Overflow 0: Disable (Mask). 1: Enable (Unmask).

5.11. Interrupt Priority Register

The Interrupt Priority (IPR) Register (FFh, Set1, Bank0) is used to set the relative priorities of the interrupt levels used in the microcontroller's interrupt structure. After a reset, all IPR bit values are undetermined and must be written to their required settings by the initialization routine.

When more than one interrupt source is active, the source with the highest priority level is serviced first. If both sources belong to the same interrupt level, the source with the lowest vector address usually receives priority; this priority is fixed in hardware.

To support programming of the relative interrupt level priorities, interrupts are organized into groups and subgroups by the interrupt logic. These groups and subgroups are used only by IPR logic for the IPR Register priority definitions, as shown in Figure 46.

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

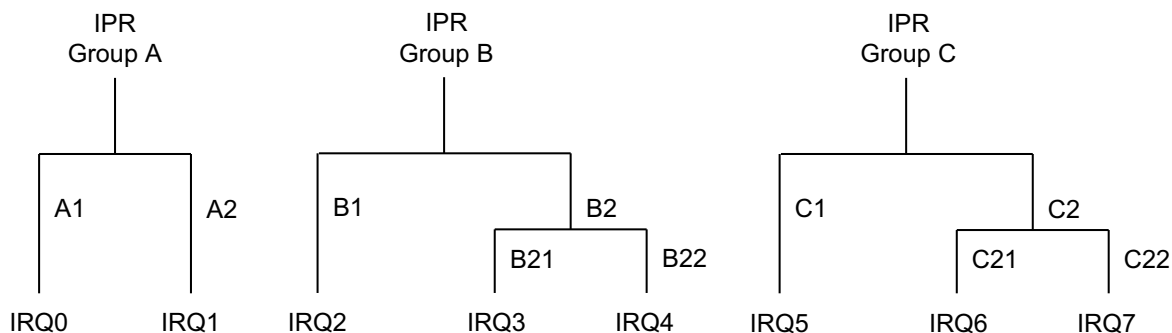


Figure 46. Interrupt Request Priority Groups

In the Interrupt Priority Register, IPR.7, IPR.4 and IPR.1 control the relative priority of interrupt groups A, B, and C. For example, a setting of 001b for these bits would select the group relationship B > C > A; a setting of 101b would select the relationship C > B > A.

The functions of the other IPR bit settings are:

- IPR.5 controls the relative priorities of group C interrupts.
- Interrupt group B includes a subgroup to provide an additional priority relationship between for interrupt levels 2, 3, and 4. IPR.3 defines the possible subgroup B relationships. IPR.2 controls interrupt group B.

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

The contents of the Interrupt Priority Register are described in Table 16.

Table 16. Interrupt Priority Register (IPR; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0	
Reset	x	x	x	x	x	x	x	x	
R/W					R/W				
Address					FEh				

Note: R/W = read/write.

Bit	Description
[7, 4, 1]	Priority Control Bits for Interrupt Groups A, B, and C 000: Group priority undefined. 001: B > C > A. 010: A > B > C. 011: B > A > C. 100: C > A > B. 101: C > B > A. 110: A > C > B. 111: 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: *The S3F80PB MCU interrupt structure uses eight levels: IRQ0–IRQ7.

5.12. Interrupt Request Register

Bit values in the Interrupt Request (IRQ) Register (DCh, Set1, Bank0) can be polled to monitor interrupt request status for all levels in the microcontroller’s interrupt structure. Each bit corresponds to the interrupt level of the same number: bit 0 to IRQ0, bit 1 to IRQ1, etc. A 0 indicates that no interrupt request is currently being issued for that level; a 1 indicates that an interrupt request is generated for that level.

IRQ bit values are read-only addressable using Register Addressing Mode. The contents of the IRQ Register can be read (i.e., tested) at any time using bit or byte addressing to determine the current interrupt request status of specific interrupt levels. After a reset, all IRQ status bits are cleared to 0.

IRQ Register values can be polled even if a DI instruction has been executed (i.e., if global interrupt processing is disabled). If an interrupt occurs while the interrupt structure is disabled, the CPU will not service it. However, the interrupt request can still be detected by polling the IRQ Register. In this way, the events that occur while the interrupt structure is globally disabled can be determined.

The contents of the Interrupt Request Register are described in Table 17.

Table 17. Interrupt Request Register (IRQ; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R							
Address	DCh							

Note: R = read only.

Bit	Description
[7]	Level 7 (IRQ7) Request Pending Bit; External Interrupts P0.7–P0.4 0: Not pending. 1: Pending.
[6]	Level 6 (IRQ6) Request Pending Bit; External Interrupts P0.3–P0.0 0: Not pending. 1: Pending.
[5]	Level 5 (IRQ5) Request Pending Bit; External Interrupts P2.7–P2.4 0: Not pending. 1: Pending.
[4]	Level 4 (IRQ4) Request Pending Bit; External Interrupts P2.3–P2.0 0: Not pending. 1: Pending.

Bit	Description (Continued)
[3]	Level 3 (IRQ3) Request Pending Bit; Timer 2 Match/Capture or Overflow 0: Not pending. 1: Pending.
[2]	Level 2 (IRQ2) Request Pending Bit; Counter A Interrupt 0: Not pending. 1: Pending.
[1]	Level 1 (IRQ1) Request Pending Bit; Timer 1 Match/Capture or Overflow 0: Not pending. 1: Pending.
[0]	Level 0 (IRQ0) Request Pending Bit; Timer 0 Match/Capture or Overflow 0: Not pending. 1: Pending.

5.13. Interrupt Pending Function Types

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

5.13.0.1. Pending Bits Cleared Automatically by Hardware

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

In the S3F80PB MCU's interrupt structure, the Timer 0 overflow interrupt (IRQ0), the Timer 1 overflow interrupt (IRQ1), the Timer 2 overflow interrupt (IRQ3), and the Counter A interrupt (IRQ2) belong to this category of interrupts in which a pending condition is cleared automatically by hardware.

5.13.0.2. Pending Bits Cleared by the Service Routine

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

In the S3F80PB interrupt structure, pending conditions for all interrupt sources except the Timer 0 overflow interrupt, the Timer 1 overflow interrupt, the Timer 2 overflow interrupt and the Counter A borrow interrupt, must be cleared by the interrupt service routine.

5.14. Interrupt Source Polling Sequence

Interrupt request polling and servicing occurs via the following sequence:

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

5.15. Interrupt Service Routines

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

- Interrupt processing must be globally enabled (EI, SYM.0 = 1)
- The interrupt level must be enabled (i.e., the IMR Register must be unmasked)
- The interrupt level must contain the highest priority if more than one level is currently requesting service
- The interrupt must be enabled at the interrupt's source (i.e, the peripheral control register)

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

1. Reset (i.e., clear to 0) the interrupt enable bit in the SYM Register (SYM.0) to disable all subsequent interrupts.
2. Save the program counter (PC) and status flags to the system stack.
3. Branch to the interrupt vector to fetch the address of the service routine.
4. Pass control to the interrupt service routine.

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

5.16. Generating Interrupt Vector Addresses

The interrupt vector area in ROM (with the exception of Smart Option ROM cells 003Ch, 003Dh, 003Eh, and 003Fh) contains the addresses of interrupt service routines that correspond to each level in the interrupt structure.

Vectored interrupt processing follows this sequence:

1. Push the program counter's low-byte value to the stack.
2. Push the program counter's high-byte value to the stack.
3. Push the Flags 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 00h–FFh.

5.17. Nesting of Vectored Interrupts

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

To nest a higher-priority interrupt request, follow these steps:

1. Push the current 8-bit Interrupt Mask (IMR) Register 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 the EI instruction occurs.
4. When the lower-priority interrupt service routine ends, restore the IMR to its original value by returning the previous mask value from the stack (POP IMR).
5. Execute an IRET.

Depending on the application, simplification of the above procedure to some extent is possible.

5.18. Instruction Pointer

The Instruction Pointer (IP) is used by all of the S3 Family microcontrollers to control the optional high-speed interrupt processing feature called *fast interrupts*. The IP consists of register pair IPH (DAh Set1, Bank0) and IPL (DBh Set1, Bank0). The IP Register names are IPH (high byte, IP15–IP8) and IPL (low byte, IP7–IP0).

The contents of the Instruction Pointer High Byte (IPH) Register are described in Table 18.

Table 18. Instruction Pointer High Byte (IPH; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	x	x	x	x	x	x	x	x
R/W					R/W			
Address					DAh			

Note: R/W = read/write.

Bit	Description
[7:0]	<p>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).</p>

The contents of the Instruction Pointer Low Byte (IPL) Register are described in Table 19.

Table 19. Instruction Pointer Low Byte (IPL; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	x	x	x	x	x	x	x	x
R/W					R/W			
Address					DBh			

Note: R/W = read/write.

Bit	Description
[7:0]	<p>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).</p>

5.19. Fast Interrupt Processing

This feature lets you specify that an interrupt within an assigned interrupt level be completed in approximately six clock cycles instead of the usual 22 clock cycles. To select a specific interrupt level for fast interrupt processing, you 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 interrupts processing:

- The instruction pointer (IP) contains the starting address of the service routine (and is later used to swap the program counter values)
- When a fast interrupt occurs, the contents of the Flags Register are stored in an un-mapped, dedicated register called *Flags'* (a.k.a. Flags prime)

► **Note:** For the S3F80PB microcontroller, the service routine for any one of the eight interrupt levels: IRQ0–IRQ7, can be selected for fast interrupt processing.

5.19.1. Procedure for Initiating Fast Interrupts

To initiate fast interrupt processing, follow these steps:

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

5.19.2. Fast Interrupt Service Routine

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

1. The contents of the instruction pointer and the PC are swapped.
2. The Flags Register values are written to the Flags' (i.e., 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' (i.e., Flags prime) is copied automatically back to the Flags Register.
7. The fast interrupt status bit in FLAGS is cleared automatically.

5.19.3. 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 fast interrupts are being used, load the IP with a new start address when the fast interrupt service routine ends.

Chapter 6. CPU Instructions

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

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

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

6.1. 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, in which bit 0 is the least significant (i.e., right-most) bit.

6.2. 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. To learn more about register addressing, see the [Address Space](#) chapter on page 16.

6.3. Addressing Modes

There are seven explicit addressing modes: Register (R), Indirect Register (IR), Indexed (X), Direct (DA), Relative (RA), Immediate (IM) and Indirect (IA). To learn more about these addressing modes, see the [Addressing Modes](#) chapter on page 43.

6.4. Instruction Summary

All instructions are summarized by type, operand and description in Table 20.

Table 20. Instruction Summary

Mnemonic	Operands	Description
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

Table 20. Instruction Summary (Continued)

Mnemonic	Operands	Description
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
OR	dst, src	Logical OR
XOR	dst, src	Logical exclusive OR
Program Control Instruction		
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 nonequal
DJNZ	r, dst	Decrement register and jump on nonzero
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
BTR	dst	Bit reset
BITS	dst	Bit set
BOR	dst, src	Bit OR
BXOR	dst, src	Bit XOR

Table 20. Instruction Summary (Continued)

Mnemonic	Operands	Description
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
NOP	–	No operation
RCF	–	Reset carry flag
SB0	–	Set Bank0
SB1	–	Set Bank1
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

6.5. Flags Register

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

The Flags Register also contains a bit to indicate the status of fast interrupt processing (FLAGS.1) and a bank address status bit (FLAGS.0) to indicate whether Bank0 or Bank1 is currently being addressed. Flags Register can be set or reset by instructions as long as its outcome does not affect the flags, such as, Load instruction.

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

The contents of the Flags Register are described in Table 21.

Table 21. System Flags Register (FLAGS; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	x	x	x	x	x	x	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Address	D5h							

Note: R/W = read/write.

Bit	Description
[7]	Carry Flag Bit (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 Bit (Z) 0: Operation result is a nonzero value. 1: Operation result is zero.
[5]	Sign Flag Bit (S) 0: Operation generates a positive number (MSB = 0). 1: Operation generates a negative number (MSB = 1).
[4]	Overflow Flag Bit (V) 0: Operation result is $\leq +127$ or -128 . 1: Operation result is $> +127$ or < -128 .
[3]	Decimal Adjust Flag Bit (D) 0: Add operation completed. 1: Subtraction operation completed.
[2]	Half-Carry Flag Bit (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 Bit (FIS) 0: Interrupt return (IRET) in progress when read. 1: Fast interrupt service routine in progress when read.
[0]	Bank Address Selection Flag Bit (BA) 0: Bank0 is selected. 1: Bank1 is selected.

Table 22 describes each of the flags managed by the Flags Register.

Table 22. Flags

Flag	Description
C	Carry Flag (FLAGS.7) The C flag is set to 1 if the result from an arithmetic operation generates a carry-out from or a borrow to the bit 7 position (MSB). After rotate and shift operations, it contains the most recent value shifted out of the specified register. Program instructions can set, clear, or complement the carry flag.
Z	Zero Flag (FLAGS.6) For arithmetic and logic operations, the Z flag is set to 1 if the result of the operation is zero. For operations that test register bits, and for shift and rotate operations, the Z flag is set to 1 if the result is logic 0.
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 0 indicates a positive number and a logic 1 indicates a negative number.
V	Overflow Flag (FLAGS.4) The V flag is set to 1 when the result of a two's-complement operation is greater than +127 or less than -128. It is also cleared to 0 following logic operations.
D	Decimal Adjust Flag (FLAGS.3) The DA bit is used to specify what type of instruction is executed at the end during BCD operations so that a subsequent decimal adjust operation can execute correctly. The DA bit is not usually accessed by programmers, and cannot be used as a test condition.
H	Half-Carry Flag (FLAGS.2) The H bit is set to 1 whenever an addition generates a carry-out of bit 3, or when a subtraction borrows out of bit 4. It is used by the Decimal Adjust (DA) instruction to convert the binary result of a previous addition or subtraction into the correct decimal (BCD) result. The H flag is seldom accessed directly by a program.
FIS	Fast Interrupt Status Flag (FLAGS.1) The FIS bit is set during a fast interrupt cycle and reset during the IRET following interrupt servicing. When this flag is set, it inhibits all interrupts and allows the fast interrupt return to be executed when the IRET instruction is executed.
BA	Bank Address Flag (FLAGS.0) The BA flag indicates which register bank in the Set1 area of the internal register file is currently selected, Bank0 or Bank1. The BA flag is cleared to 0 (select Bank0) when you execute the SB0 instruction and is set to 1 (select Bank1) when you execute the SB1 instruction.

6.6. Instruction Set Notation

Table 23 lists the conventions used for each of the flags managed by the Instruction Set. Symbols for the Instruction Set are listed in Table 24; conditions for these symbols are described in Table 25.

Table 23. 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 0.
1	Set to logic 1.
*	Set to cleared according to operation.
–	Value is unaffected.
x	Value is undefined.

Table 24. 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	Op code.

Table 25. Instruction Notation Conventions

Notation	Description	Actual Operand Range
cc	Condition code	See list of condition codes in Table 28 on page 93
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, in which 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, in which 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, in which p = 0, 2,.., 14)
xl	Indexed (Long Offset) Addressing Mode	#addr [RRp] (addr = range 0–65535, in which 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)

Chapter 7. Op Code Maps

Tables 26 and 27 provide quick reference op code maps to addresses 0–7 and 8–F, respectively.

Table 26. Op Code Quick Reference (0–7)

		Op Code Map							
		Lower Nibble (Hex)							
		0	1	2	3	4	5	6	7
U	0	DEC R1	DEC IR1	ADD r1, r2	ADD r1, lr2	ADD R2, R1	ADD IR2, R1	ADD R1, IM	BOR r0–Rb
P	1	RLC R1	RLC IR1	ADC r1, r2	ADC r1, lr2	ADC R2, R1	ADC IR2, R1	ADC R1, IM	BCP r1.b, R2
P	2	INC R1	INC IR1	SUB r1, r2	SUB r1, lr2	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, lr2	SBC R2, R1	SBC IR2, R1	SBC R1, IM	BTJR r2.b, RA
R	4	DA R1	DA IR1	OR r1, r2	OR r1, lr2	OR R2, R1	OR IR2, R1	OR R1, IM	LDB r0–Rb
	5	POP R1	POP IR1	AND r1, r2	AND r1, lr2	AND R2, R1	AND IR2, R1	AND R1, IM	BITC r1.b
N	6	COM R1	COM IR1	TCM r1, r2	TCM r1, lr2	TCM R2, R1	TCM IR2, R1	TCM R1, IM	BAND r0–Rb
I	7	PUSH R2	PUSH IR2	TM r1, r2	TM r1, lr2	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, lr2	CP R2, R1	CP IR2, R1	CP R1, IM	LDC r1, lrr2, xL
E	B	CLR R1	CLR IR1	XOR r1, r2	XOR r1, lr2	XOR R2, R1	XOR IR2, R1	XOR R1, IM	LDC r2, lrr2, xL
	C	RRC R1	RRC IR1	CPIJE lr, r2, RA	LDC r1, lrr2	LDW RR2, RR1	LDW IR2, RR1	LDW RR1, IML	LD r1, lr2
H	D	SRA R1	SRA IR1	CPIJNE lr, r2, RA	LDC r2, lrr1	CALL IA1		LD IR1, IM	LD lr1, r2
E	E	RR R1	RR IR1	LDCD r1, lrr2	LDCI r1, lrr2	LD R2, R1	LD R2, IR1	LD R1, IM	LDC r1, lrr2, xs
X	F	SWAP R1	SWAP IR1	LDCPD r2, lrr1	LDCPI r2, lrr1	CALL IRR1	LD IR2, R1	CALL DA1	LDC r2, lrr1, xs

Table 27. Op Code Quick Reference (8–F)

Op Code 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

7.1. Condition Codes

The op code of a conditional jump always contains a 4-bit field called the *condition code* (cc), which specifies under which conditions it is to execute the jump. For example, a conditional jump with a condition code of *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.

These condition codes are listed in Table 28.

Table 28. Condition Codes

Mnemonic	Binary	Description	Flags Set
F	0000	Always false	–
T	1000	Always true	–
C	0111 ¹	Carry	C = 1
NC	1111 ¹	No carry	C = 0
Z	0110 ¹	Zero	Z = 1
NZ	1110 ¹	Not zero	Z = 0
PL	1101	Plus	S = 0
MI	0101	Minus	S = 1
OV	0100	Overflow	V = 1
NOV	1100	No overflow	V = 0
EQ	0110 ¹	Equal	Z = 1
NE	1110 ¹	Not equal	Z = 0
GE	1001	Greater than or equal	(S XOR V) = 0
LT	0001	Less than	(S XOR V) = 1
GT	1010	Greater than	(Z OR (S XOR V)) = 0
LE	0010	Less than or equal	(Z OR (S XOR V)) = 1
UGE	1111 ^{1&2}	Unsigned greater than or equal	C = 0
ULT	0111 ^{1&2}	Unsigned less than	C = 1
UGT	1011 ²	Unsigned greater than	(C = 0 AND Z = 0) = 1
ULE	0011 ²	Unsigned less than or equal	(C OR Z) = 1

Note:

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

7.2. Instruction Set

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

The following information is included in each instruction description:

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

Add with Carry

ADC dst, src

Operation $dst \leftarrow dst + src + c$

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

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

Format	Bytes	Cycles	Op Code (Hex)	Address Mode				
				dst	src			
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px;">opc</td> <td style="padding: 2px;">dst src</td> </tr> </table>	opc	dst src	2	4	12	r	r	
	opc	dst src						
13	r	lr						
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px;">opc</td> <td style="padding: 2px;">src</td> <td style="padding: 2px;">dst</td> </tr> </table>	opc	src	dst	3	6	14	R	R
	opc	src	dst					
15	R	IR						
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px;">opc</td> <td style="padding: 2px;">dst</td> <td style="padding: 2px;">src</td> </tr> </table>	opc	dst	src	3	6	16	R	IM
opc	dst	src						

Example Assume that 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	R1, R2	→	R1 = 14h, R2 = 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 1, and the source working register R2 contains the value 03h. The *ADC R1, R2* statement adds 03h and the carry flag value (1) to the destination value 10h, leaving 14h in register R1.

Add

ADD dst, src

Operation $dst \leftarrow dst + src$

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 additions are performed.

Flags

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

Format		Bytes	Cycles	Op Code (Hex)	Address 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

Example Assume that 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 *ADD R1, R2* statement adds 03h to 12h, leaving the value 15h in register R1.

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 1s; otherwise a 0 bit value is stored. The contents of the source are unaffected.

Flags

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

Format			Bytes	Cycles	Op Code (Hex)	Address Mode dst	src		
	<table border="1" style="display: inline-table;"> <tr> <td>opc</td> <td>dst src</td> </tr> </table>		opc	dst src	2	4	52	r	r
opc	dst src								
			6	53	r	lr			
<table border="1" style="display: inline-table;"> <tr> <td>opc</td> <td>src</td> <td>dst</td> </tr> </table>		opc	src	dst	3	6	54	R	R
opc	src	dst							
			6	55	R	IR			
<table border="1" style="display: inline-table;"> <tr> <td>opc</td> <td>dst</td> <td>src</td> </tr> </table>		opc	dst	src	3	6	56	R	IM
opc	dst	src							

Example Assume that 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 *AND R1, R2* statement

logically ANDs the source operand 03h with the destination operand value 12h, leaving the value 02h in register R1.

BAND-Bit AND

BAND dst, src.b

BAND dst.b, src

Operation $\text{dst}(0) \leftarrow \text{dst}(0) \text{ AND } \text{src}(b)$
or
 $\text{dst}(b) \leftarrow \text{dst}(b) \text{ AND } \text{src}(0)$

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

Flags

- C** Unaffected.
- Z** Set if the result is 0; cleared otherwise.
- S** Cleared to 0.
- V** Undefined.
- D** Unaffected.
- H** Unaffected.

Format			Bytes	Cycles	Op Code (Hex)	Address 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 four bits, the bit address b is three bits, and the LSB address value is one bit in length.

Example Assume that 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 *BAND R1, 01h.1* statement ANDs the bit 1 value of the source register (0) with the bit 0 value of register R1 (destination), leaving the value 06h (00000110b) in register R1.

Bit Compare

BCP dst, src.b

Operation dst(0)–src(b)

The specified bit of the source is compared to (subtracted from) bit 0 (LSB) of the destination. The zero flag is set if the bits are the same; otherwise it is cleared. The contents of both operands are unaffected by the comparison.

Flags

- C** Unaffected.
- Z** Set if the two bits are the same; cleared otherwise.
- S** Cleared to 0.
- V** Undefined.
- D** Unaffected.
- H** Unaffected.

Format				Bytes	Cycles	Op Code	Address Mode	
						(Hex)	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 four bits, the bit address b is three bits, and the LSB address value is one bit in length.

Example Assume that 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 *BCP R1, 01h.1* statement compares bit 1 of the source register (01h) and bit 0 of the destination register (R1). Because the bit values are not identical, the zero flag bit (Z) is cleared in the Flags Register (0D5h).

Bit Complement

BITC dst.b

Operation $\text{dst}(b) \leftarrow \text{NOT } \text{dst}(b)$

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

Flags

- C** Unaffected.
- Z** Set if the result is 0; cleared otherwise.
- S** Cleared to 0.
- V** Undefined.
- D** Unaffected.
- H** Unaffected.

Format	Bytes	Cycles	Op Code (Hex)	Address Mode dst		
<table border="1"> <tr> <td>opc</td> <td>dst b 0</td> </tr> </table>	opc	dst b 0	2	4	57	rb
opc	dst b 0					

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

Example Assume that $R1 = 07h$.

BITC R1.1 → R1 = 05h

If working register R1 contains the value 07h (00000111b), the *BITC R1.1* statement complements bit 1 of the destination and leaves the value 05h (00000101b) in register R1. Because the result of the complement is not 0, the zero flag (Z) in the Flags Register (0D5h) is cleared.

Bit Reset

BITR dst.b

Operation dst(b) ← 0

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

Flags No flags are affected.

Format	Bytes	Cycles	Op Code (Hex)	Address Mode dst		
<table border="1" style="display: inline-table;"> <tr> <td style="padding: 2px;">opc</td> <td style="padding: 2px;">dst b 0</td> </tr> </table>	opc	dst b 0	2	4	77	rb
opc	dst b 0					

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

Example Assume that R1 = 07h.

BITR R1.1 → R1 = 05h

If the value of working register R1 is 07h (00000111b), the *BITR R1.1* statement clears bit 1 of the destination register R1, leaving the value 05h (00000101b).

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	Op Code (Hex)	Address Mode		
<table border="1"> <tr> <td>opc</td> <td>dst b 1</td> </tr> </table>	opc	dst b 1	2	4	77	dst rb
opc	dst b 1					

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

Example Assume that R1 = 07h.

BITR R1.3 → R1 = 0Fh

If working register R1 contains the value 07h (00000111b), the *BITS R1.3* statement sets bit 3 of the destination register R1 to 1, leaving the value 0Fh (00001111b).

Bit OR

BOR dst, src.b

BOR dst.b, src

Operation $dst(0) \leftarrow dst(0) \text{ OR } src(b)$
or
 $dst(b) \leftarrow dst(b) \text{ OR } src(0)$

The specified bit of the source (or the destination) is logically ORed with bit 0 (LSB) of the destination (or the source). The resulting bit value is stored in the specified bit of the destination. No other bits of the destination are affected. The source is unaffected.

Flags

- C** Unaffected.
- Z** Set if the result is 0; cleared otherwise.
- S** Cleared to 0.
- V** Undefined.
- D** Unaffected.
- H** Unaffected.

Format			Bytes	Cycles	Op Code (Hex)	Address 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 four bits, the bit address b is three bits, and the LSB address value is one bit.

Example Assume that 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 *BOR R1, 01h.1* statement logically ORs bit 1 of register 01h (source) with bit 0 of R1 (destination), which 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 *BOR 01h,2, R1* statement logically ORs bit 2 of register 01h (destination) with bit 0 of R1 (source) which leaves the value 07h in register 01h.

Bit Test, Jump Relative on False

BTJRF dst, src.b

Operation If src(b) is a 0, then $PC \leftarrow PC + dst$

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

Flags No flags are affected.

Format			Bytes	Cycles	Op Code (Hex)	Address 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 four bits, the bit address b is three bits, and the LSB address value is one bit in length.

Example Assume that $R1 = 07h$.

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

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

Bit Test, Jump Relative on True

BTJRT dst, src.b

Operation If src(b) is a 1, then $PC \leftarrow PC + dst$

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 for which the address is now in the PC; otherwise, the instruction following the BTJRT instruction is executed.

Flags No flags are affected.

Format	Bytes	Cycles	Op Code (Hex)	Address Mode				
				dst	src			
<table border="1"> <tr> <td>opc</td> <td>src b 1</td> <td>dst</td> </tr> </table>	opc	src b 1	dst	3	10	37	RA	rb
opc	src b 1	dst						

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

Example Assume that $R1 = 07h$.

BTJRT SKIP, R1.1

If working register R1 contains the value 07h (00000111b), the *BTJRT SKIP, R1.1* statement tests bit 1 in the source register (R1). Because it is a 1, the relative address is added to the PC and the PC jumps to the memory location pointed to by the SKIP. (Remember that the memory location must be within the allowed range of +127 to -128.)

Bit XOR

BXOR dst, src.b

BXOR dst.b, src

Operation $\text{dst}(0) \leftarrow \text{dst}(0) \text{ XOR } \text{src}(b)$
or
 $\text{dst}(b) \leftarrow \text{dst}(b) \text{ XOR } \text{src}(0)$

The specified bit of the source (or the destination) is logically exclusive-ORed with bit 0 (LSB) of the destination (or source). The result bit is stored in the specified bit of the destination. No other bits of the destination are affected. The source is unaffected.

Flags

- C** Unaffected.
- Z** Set if the result is 0; cleared otherwise.
- S** Cleared to 0.
- V** Undefined.
- D** Unaffected.
- H** Unaffected.

Format				Bytes	Cycles	Op Code	Address Mode	
						(Hex)	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.

Example Assume that R1 = 07h (00000111b) and register 01h = 03h (00000011b).

BXOR R1, 01h.1 → R1 = 06h, register 01h = 03h
 BXOR 01h.2, R1 → Register 01h = 07h, R1 = 07h

In the first example, destination working register R1 contains the value 07h (00000111b) and source register 01h contains the value 03h (00000011b). The *BXOR R1, 01h.1* statement 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. The value of source register 01h is unaffected.

Call Procedure

CALL	dst		
Operation	SP	←	SP-1
	@SP	←	PCL
	SP	←	SP-1
	@SP	←	PCH
	PC	←	dst

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

Flags No flags are affected.

Format		Bytes	Cycles	Op Code (Hex)	Address Mode
opc	dst	3	14	F6	DA
opc	dst	2	12	F4	IRR
opc	dst	2	14	D4	IA

Example Assume that R0 = 35h, R1 = 21h, PC = 1A47h, and SP = 0002h.

CALL	3521h	→	SP = 0000h (Memory locations 0000h = 1Ah, 0001h = 4Ah, in which 4Ah is the address that follows the instruction.)
CALL	@RR0	→	SP = 0000h (0000h = 1Ah, 0001h = 49h)
CALL	#40h	→	SP = 0000h (0000h = 1Ah, 0001h = 49h)

In the first example, if the program counter value is 1A47h and the stack pointer contains the value 0002h, the *CALL 3521h* statement pushes the cur-

rent PC value onto the top of the stack. The stack pointer now points to memory location 0000h. The PC is then loaded with the value 3521h, the address of the first instruction in the program sequence to be executed.

If the contents of the program counter and stack pointer are the same as in the first example, the *CALL @RR0* statement produces the same result except that the 49h is stored in stack location 0001h (because the two-byte instruction format is used). The PC is then loaded with the value 3521h, the address of the first instruction in the program sequence to be executed. Assuming that the contents of the program counter and stack pointer are the same as in the first example, if program address 0040h contains 35h and program address 0041h contains 21h, the *CALL #40h* statement produces the same result as in the second example.

Complement Carry Flag

CCF

Operation $C \leftarrow \text{NOT } C$

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

Flags **C** Complemented.
No other flags are affected.

Format	Bytes	Cycles	Op Code (Hex)
opc	1	4	EF

Example Assume that the carry flag = 0.

CCF

If the carry flag = 0, the CCF instruction complements it in the Flags Register (0D5h), changing its value from logic 0 to logic 1.

Clear

CCF dst

Operation dst ← 0

The destination location is cleared to 0.

Flags No flags are affected.

Format

		Bytes	Cycles	Op Code (Hex)	Addr Mode
opc	dst	2	4	B0	R
			4	B1	IR

Example

Assume that 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 *CLR 00h* statement clears the destination register 00h value to 00h. In the second example, the *CLR @01h* statement uses Indirect Register (IR) Addressing Mode to clear the 02h register value to 00h.

Complement

COM dst

Operation dst ← NOT dst

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

Flags

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

Format

		Bytes	Cycles	Op Code (Hex)	Address Mode		
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px 10px;">opc</td> <td style="padding: 2px 10px;">dst</td> </tr> </table>	opc	dst		2	4	60	R
	opc	dst					
			4	61	IR		

Example

Assume that 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 *COM R1* statement complements all of the bits in R1: all logic 1s are changed to logic 0s, and vice-versa, leaving the value 0F8h (11111000b).

In the second example, Indirect Register (IR) Addressing Mode is used to complement the value of destination register 07h (11110001b), leaving the new value 0Eh (00001110b).

Compare

CP dst, src

Operation dst ← src

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

Flags

- C** Set if a borrow occurred (src > dst); cleared otherwise.
- Z** Set if the result is 0; cleared otherwise.
- S** Set if the result is negative; cleared otherwise.
- V** Set if arithmetic overflow occurred; cleared otherwise.
- D** Unaffected.
- H** Unaffected.

Format	Bytes	Cycles	Op Code (Hex)	Address Mode				
				dst	src			
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px;">opc</td> <td style="padding: 2px;">dst src</td> </tr> </table>	opc	dst src	2	4	A2	r	r	
	opc	dst src						
A3	r	lr						
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px;">opc</td> <td style="padding: 2px;">src</td> <td style="padding: 2px;">dst</td> </tr> </table>	opc	src	dst	3	6	A4	R	R
	opc	src	dst					
A5	R	IR						
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px;">opc</td> <td style="padding: 2px;">dst</td> <td style="padding: 2px;">src</td> </tr> </table>	opc	dst	src	3	6	A6	R	IM
opc	dst	src						

Example 1. Assume that 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 *CP R1, R2* statement subtracts the R2 value (source/subtrahend) from the R1 value (destination/minuend). Because a borrow occurs and the difference is negative, C and S are 1.

2. Assume that 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 *CP R1, R2* statement generates $C = 1$ and the JP instruction does not jump to the SKIP location. After the *LD R3, R1* statement executes, the value 06h remains in working register R3.

Compare, Increment, and Jump on Equal

CPIJE dst, src, RA

Operation If $dst - src = 0$, $PC \leftarrow PC + RA$

$Ir \leftarrow Ir + 1$

The source operand is compared to (subtracted from) the destination operand. If the result is 0, the relative address is added to the program counter and control passes to the statement for which the address is now in the program counter. Otherwise, the instruction immediately following the CPIJE instruction is executed. In either case, the source pointer is incremented by one before the next instruction is executed.

Flags No flags are affected.

Format

				Bytes	Cycles	Op Code (Hex)	Address Mode dst	src
opc	dst src	RA		3	12	C2	r	lr

Note: Execution time is 18 cycles if the jump is taken or 16 cycles if it is not taken.

Example

Assume that $R1 = 02h$, $R2 = 03h$, and register $03h = 02h$.

CPIJE R1, @R2, SKIP → R2 = 04h, PC jumps to SKIP location

In this example, working register R1 contains the value 02h, working register R2 the value 03h, and register 03 contains 02h. The *CPIJE R1, @R2, SKIP* statement compares the @R2 value 02h (00000010b) to 02h (00000010b). Because the result of the comparison is equal, the relative address is added to the PC and the PC then jumps to the memory location pointed to by SKIP. The source register (R2) is incremented by one, leaving a value of 04h. (Remember that the memory location must be within the allowed range of +127 to -128.)

Compare, Increment, and Jump on NonEqual

CPIJNE dst, src, RA

Operation If $dst - src = 0$, $PC \leftarrow PC + RA$

$Ir \leftarrow Ir + 1$

The source operand is compared to (subtracted from) the destination operand. If the result is 0, the relative address is added to the program counter and control passes to the statement for which the address is now in the program counter. Otherwise, the instruction immediately following the CPIJNE instruction is executed. In either case, the source pointer is incremented by one before the next instruction is executed.

Flags No flags are affected.

Format

				Bytes	Cycles	Op Code	Address Mode	
						(Hex)	dst	src
opc	src	dst	RA	3	12	D2	r	lr

Note: Execution time is 18 cycles if the jump is taken or 16 cycles if it is not taken.

Example

Assume that $R1 = 02h$, $R2 = 03h$, and register $03h = 04h$.

`CPIJE R1, @R2, SKIP` → $R2 = 04h$, PC jumps to SKIP location

Working register R1 contains the value 02h, working register R2 (the source pointer) the value 03h, and general register 03 the value 04h. The `CPIJNE R1, @R2, SKIP` statement subtracts 04h (00000100b) from 02h (00000010b). Because the result of the comparison is nonequal, the relative address is added to the PC and the PC then jumps to the memory location pointed to by SKIP. The source pointer register (R2) is also incremented by one, leaving a value of 04h. (Remember that the memory location must be within the allowed range of +127 to -128.)

Decimal Adjust

DA dst

Operation dst ← DA dst

The destination operand is adjusted to form two 4-bit BCD digits following an addition or subtraction operation. For addition (ADD, ADC) or subtraction (SUB, SBC), Table 29 indicates the operations performed. These operations are undefined if the destination operand is not the result of a valid addition or subtraction of BCD digits.

Table 29. DA Instruction

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
ADD	0	0–9	0	0–9	00	0
ADC	0	0–8	0	A–F	06	0
	0	0–9	1	0–3	06	0
	0	A–F	0	0–9	60	1
	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
	SUB	0	0–9	0	0–9	00 =00
SBC	0	0–8	1	6–F	FA =06	0
	1	7–F	0	0–9	A0 =60	1
	1	6–F	1	6–F	9A =66	1

Flags

- C Set if there is a carry from the most significant bit; cleared otherwise (see Table 29).
- Z Set if result is 0; cleared otherwise.
- S Set if result bit 7 is set; cleared otherwise.
- V Undefined.
- D Unaffected.
- H Unaffected.

Format

		Bytes	Cycles	Op Code (Hex)	Address Mode
opc	dst	2	4	40	R
			4	41	IR

Example

Assume that Working register R0 contains the value 15 (BCD), working register R1 contains 27 (BCD), and address 27h contains 46 (BCD).

```
ADD    R1, R0    ; C ← 0, H ← 0, Bits 4–7 = 3, bits 0–3 = C, R1 ← 3Ch
DA     R1        ; R1 ← 3Ch + 06
```

If addition is performed using the BCD values 15 and 27, the result should be 42. The sum is incorrect, however, when the binary representations are added in the destination location using standard binary arithmetic:

$$\begin{array}{r}
 \\
 0001 \\
 + 0101 \\
 \hline
 0011 = 3Ch
 \end{array}$$

The DA instruction adjusts this result so that the correct BCD representation is obtained:

$$\begin{array}{r}
 \\
 0011 \\
 + 0110 \\
 \hline
 0100 = 42
 \end{array}$$

Assuming the same values which are presented above, the following statements leave the value 31 (BCD) in address 27h (@R1).

```
SUB    27h, R0    ; C ← 0, H ← 0, Bits 4–7 = 3, bits 0–3 = 1
DA     @R1        ; @R1 ← 31–0
```

Decrement

DEC dst

Operation $dst \leftarrow dst - 1$

The contents of the destination operand are decremented by one.

Flags

- C** Unaffected.
- Z** Set if the result is 0; cleared otherwise.
- S** Cleared to 0.
- V** Undefined.
- D** Unaffected.
- H** Unaffected.

Format		Bytes	Cycles	Op Code (Hex)	Address Mode
opc	dst	2	4	00	R
			4	01	IR

Example Assume that 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 *DEC R1* statement decrements the hexadecimal value by one, leaving the value 02h. In the second example, the *DEC @R1* statement decrements the value 10h contained in the destination register 03h by one, leaving the value 0Fh.

Decrement Word

DECW dst

Operation $dst \leftarrow dst - 1$

The contents of the destination location (which must be an even address) and the operand following that location are treated as a single 16-bit value that is decremented by one.

Flags

- C** Unaffected.
- Z** Set if the result is 0; cleared otherwise.
- S** Set if the result is negative; cleared otherwise.
- V** Set if arithmetic overflow occurred; cleared otherwise.
- D** Unaffected.
- H** Unaffected.

Format		Bytes	Cycles	Op Code (Hex)	Address Mode dst
opc	dst	2	8	80	RR
			8	81	IR

Example Assume that R0 = 12h, R1 = 34h, R2 = 30h, register 30h = 0Fh, and register 31h = 21h.

```
DECW RR0      → R0 = 12h, R1 = 33h
DECW @R2     → Register 30h = 0Fh, register 31h = 20h
```

In the first example, destination register R0 contains the value 12h and register R1 the value 34h. The *DECW RR0* statement addresses R0 and the following operand R1 as a 16-bit word and decrements the value of R1 by one, leaving the value 33h.

Note A system malfunction can occur if you use a Zero flag (FLAGS.6) result together with a DECW instruction. To avoid this problem, Zilog recommends that you use DECW as shown in the following example:

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

Disable Interrupts

DI

Operation SYM (0) ← 0

Bit 0 of the System Mode Control Register, SYM.0, is cleared to 0, globally disabling all interrupt processing. Interrupt requests will continue to set their respective interrupt pending bits; however, the CPU will not service them while interrupt processing is disabled.

Flags No flags are affected.

Format

	Bytes	Cycles	Op Code (Hex)
opc	1	4	8F

Example Assume that SYM = 01h.

DI

If the value of the SYM Register is 01h, the *DI* statement leaves the new value 00h in the register and clears SYM.0 to 0, disabling interrupt processing.

Execute a DI instruction prior to changing the Interrupt Mask (IMR), Interrupt Pending (IPR), and Interrupt Request (IRQ) Registers.

Divide (Unsigned)

DIV dst, src

Operation dst ÷ src

dst (UPPER) ← REMAINDER

dst (LOWER) ← QUOTIENT

The destination operand (16 bits) is divided by the source operand (8 bits). The quotient (8 bits) is stored in the lower half of the destination. The remainder (8 bits) is stored in the upper half of the destination. When the quotient is $\geq 2^8$, the numbers stored in the upper and lower halves of the destination for quotient and remainder are incorrect. Both operands are treated as unsigned integers.

Flags

- C** Set if the V flag is set and quotient is between 2^8 and $2^9 - 1$; cleared otherwise.
- Z** Set if divisor or quotient = 0; cleared otherwise.
- S** Set if MSB of quotient = 1; cleared otherwise.
- V** Set if quotient is $\geq 2^8$ or if divisor = 0; cleared otherwise.
- D** Unaffected.
- H** Unaffected.

Format

			Bytes	Cycles	Op Code (Hex)	Address Mode dst	src	
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px 10px;">opc</td> <td style="padding: 2px 10px;">src</td> <td style="padding: 2px 10px;">dst</td> </tr> </table>	opc	src	dst	3	26/10	94	RR	R
	opc	src	dst					
		26/10	95					
	26/10	96	RR	IM				

Note: Execution takes 10 cycles if the divide-by-zero is attempted; otherwise it takes 26 cycles.

Example

Assume that 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 *DIV RR0, R2* statement 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).

Decrement and Jump if NonZero

DJNZ r, dst

Operation $r \leftarrow r - 1$

If $r \neq 0$, $PC \leftarrow PC + \text{dst}$

The working register being used as a counter is decremented. If the contents of the register are not logic 0 after decrementing, the relative address is added to the program counter and control passes to the statement for which the address is now in the PC. The range of the relative address is +127 to -128, and the original value of the PC is taken to be the address of the instruction byte following the *DJNZ* statement.

► **Note:** If using DJNZ instruction, the working register being used as a counter should be set at the one of location 0C0h to 0CFh with SRP, SRP0, or SRP1 instruction.

Flags No flags are affected.

Format	Bytes	Cycles	Op Code (Hex)	Address Mode			
<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>r</td><td>opc</td><td>dst</td></tr></table>	r	opc	dst	2	8 (jump taken) 8 (no jump)	rA r = 0 to F	dst RA
r	opc	dst					

Example Assume that 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 *DJNZ R1, LOOP* statement decrements register R1 by one, leaving the value 01h. Because the contents of R1 after the decrement are nonzero, the jump is taken to the relative address specified by the LOOP label.

Enable Interrupts

EI

Operation SYM (0) ← 1

An EI instruction sets bit 0 of the System Mode Register, SYM.0, to 1. This instruction allows interrupts to be serviced as they occur (assuming they contain highest priority). If an interrupt's pending bit is set while interrupt processing is disabled (by executing a DI instruction), it will be serviced when you execute the EI instruction.

Flags No flags are affected.

Format

	Bytes	Cycles	Op Code (Hex)
opc	1	4	9F

Example Assume that SYM = 00h.

EI

If the SYM Register contains the value 00h, i.e., if interrupts are currently disabled, the *EI* statement sets the SYM Register to 01h, enabling all interrupts. (SYM.0 is the enable bit for global interrupt processing.)

Enter

ENTER

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

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

Flags No flags are affected.

Format	Bytes	Cycles	Op Code (Hex)
opc	1	14	1F

Example Figure 47 shows an example of how to use an *ENTER* statement.

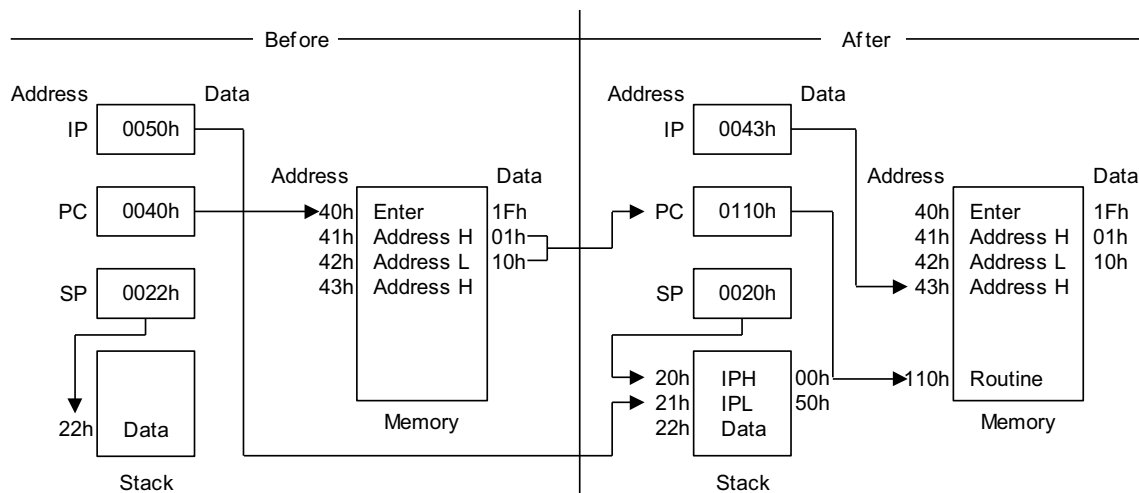


Figure 47. How to Use an ENTER Statement

Exit

EXIT

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

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

Flags No flags are affected.

Format	Bytes	Cycles	Op Code
			(Hex)
opc	1	14 (internal stack) 16 (internal stack)	2F

Example Figure 48 shows an example of how to use an *EXIT* statement.

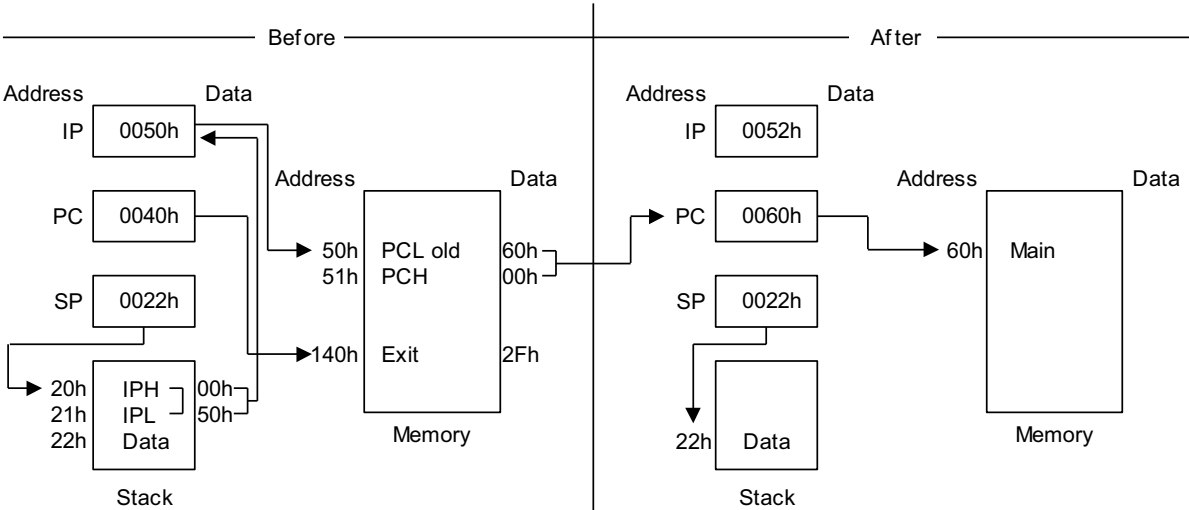


Figure 48. How to Use an EXIT Statement

Idle Operation

IDLE

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

Flags No flags are affected.

Format	Bytes	Cycles	Op Code (Hex)	
<table border="1"><tr><td>opc</td></tr></table>	opc	1	4	6F
opc				

Example The Idle instruction stops the CPU clock but not the system clock.

Increment

INC dst

Operation dst ← dst + 1

The contents of the destination operand are incremented by one.

Flags

- C** Unaffected.
- Z** Set if the result is 0; cleared otherwise.
- S** Set if the result is negative; cleared otherwise.
- V** Set if arithmetic overflow occurred; cleared otherwise.
- D** Unaffected.
- H** Unaffected.

Format	Bytes	Cycles	Op Code (Hex)	Address Mode
dst src	1	4	rE r = 0 to F	dst r
opc dst	2	4 4	20 21	R IR

Example Assume that 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 *INC R0* statement leaves the value 1Ch in that same register.

The next example shows the effect of an INC instruction 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.

Increment Word

INCW dst

Operation $dst \leftarrow dst + 1$

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

Flags

- C** Unaffected.
- Z** Set if the result is 0; cleared otherwise.
- S** Set if the result is negative; cleared otherwise.
- V** Set if arithmetic overflow occurred; cleared otherwise.
- D** Unaffected.
- H** Unaffected.

Format

		Bytes	Cycles	Op Code (Hex)	Address Mode
opc	dst	2	8	A0	RR
			8	A1	IR

Example Assume that 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 *INCW RR0* statement increments the 16-bit destination by one, leaving the value 03h in register R1. In the second example, the *INCW @R1* statement 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 can occur if you use a Zero (Z) flag (FLAGS.6) result together with an INCW instruction. To avoid this problem, Zilog recommends that you use INCW as shown in the following example:

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

Interrupt Return

IRET	IRET (Normal), IRET (FAST)			
Operation	IRET(Normal)	FLAGS	←	@SP
		SP	←	SP + 1
		PC	←	@SP
		SP	←	SP + 2
		SYM (0)	←	1
	IRET (Fast)	PC	↔	IP
		FLAGS	←	FLAGS
		FIS	←	0

This instruction is used at the end of an interrupt service routine. It restores the Flags Register and the program counter. It also reenables global interrupts. A *normal IRET* is executed only if the fast interrupt status bit (FIS, bit 1 of the Flags Register, 0D5h) is cleared (= 0). If a fast interrupt occurred, IRET clears the FIS bit that is set at the beginning of the service routine.

Flags All flags are restored to their original settings (i.e., the settings before the interrupt occurred).

Format	IRET (Normal)	Bytes	Cycles	Op Code (Hex)
		opc	1	10 (internal stack) 12 (internal stack)
	IRET (Fast)	Bytes	Cycles	Op Code
	opc	1	6	BF

Example In Figure 49, 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 which allows the PC to jump to address 100h and the IP to keep the return address.

The final instruction in the service routine normally is a jump to IRET at address FFh which allows 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.

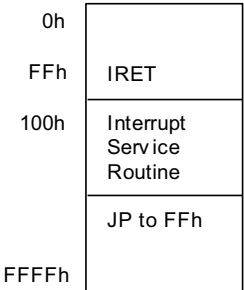


Figure 49. Instruction Pointer

► **Note:** In the fast interrupt example above, if the final instruction is not a jump to IRET, you must pay attention to the order of the final two instructions. The IRET cannot be immediately preceded by a clearing of the interrupt status (as with a reset of the IPR Register).

Jump

JP cc, dst (Conditional)

JP dst (Unconditional)

Operation If cc is true, PC ← dst

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

Flags No flags are affected.

Format

		Bytes ¹	Cycles	Op Code (Hex)	Address Mode dst
cc opc ²	dst	3	8	ccD	DA
			6	cc = 0 to F	
opc	src	2	8	30	IRR

Notes:

1. The 3-byte format is used for a conditional jump and the 2-byte format for an unconditional jump.
2. In the first byte of the three-byte instruction format (conditional jump), the condition code and the op code are both four bits.

Example Assume that the carry flag (C) = 1, register 00 = 01h and register 01 = 20h.

JP C, LABEL_W → LABEL_W = 1000h, PC = 1000h

JP @00h → PC = 0120h

The first example shows a conditional JP. Assuming that the carry flag is 1, the JP C, LABEL_W statement 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 pass to the statement immediately following the JP instruction.

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

Jump Relative

JR cc, dst

Operation If cc is true, $PC \leftarrow PC + dst$

If the condition specified by the condition code (cc) is true, the relative address is added to the program counter and control passes to the statement for which the address is now in the program counter; otherwise, the instruction following the JR instruction is executed; see the list of condition codes in [Table 28](#) on page 93.

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

Flags No flags are affected.

Format

	Bytes	Cycles	Op Code (Hex)	Address Mode
*				dst
cc opc dst	2	6	ccB	RA
			cc = 0 to F	

Note: *In the first byte of the two-byte instruction format, the condition code and the op code are each four bits.

Example Assume that the carry flag = 1 and LABEL_X = 1FF7h.

JR C, LABEL_X → PC = 1FF7h

If the carry flag is set (i.e., if the condition code is true), the JR C, LABEL_X statement passes control to the statement for which the address is now in the PC. Otherwise, the program instruction following the JR would be executed.

Load

LD dst, src

Operation dst ← src

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

Flags No flags are affected.

Format

			Bytes	Cycles	Op Code (Hex)	Address Mode dst	src
dst src	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

Example

Assume that R0 = 01h, R1 = 0Ah, register 00h = 01h, register 01h = 20h, register 02h = 02h, LOOP = 30h and register 3Ah = 0FFh.

```
LD R0, #10h      → R0 = 10h
LD R0, 01h      → R0 = 20h, register 01h = 20h
LD 01h, R0      → Register 01h = 01h, R0 = 01h
LD R1, @R0      → R1 = 20h, R0 = 01h
LD @R0, R1      → R0 = 01h, R1 = 0Ah, register 01h = 0Ah
LD 00h, 01h     → Register 00h = 20h, register 01h = 20h
LD 02h, @00h    → Register 02h = 20h, register 00h = 01h
LD 00h, #0Ah    → Register 00h = 0Ah
LD @00h, #10h   → Register 00h = 01h, register 01h = 10h
LD @00h, 02h    → Register 00h = 01h, register 01h = 02, register
                 02h = 02h
LD R0, #LOOP[R1] → R0 = 0FFh, R1 = 0Ah
LD #LOOP[R0], R1 → Register 31h = 0Ah, R0 = 01h, R1 = 0Ah
```

Load Bit

LDB dst, src.b

LDB dst.b, src

Operation $\text{dst}(0) \leftarrow \text{src}(b)$
or
 $\text{dst}(b) \leftarrow \text{src}(0)$

The specified bit of the source is loaded into bit 0 (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 No flags are affected.

Format			Bytes	Cycles	Op Code (Hex)	Address 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 four bits, the bit address b is three bits, and the LSB address value is one bit in length.

Example Assume that 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 *LD R0, 00h.2* statement loads the bit 2 value of the 00h register into bit 0 of the R0 Register, leaving the value 07h in register R0.

In the second example, 00h is the destination register. The *LD 00h.0, R0* statement loads bit 0 of register R0 to the specified bit (bit 0) of the destination register, leaving 04h in general register 00h.

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 causes lrr or rr values to be even numbers for program memory and odd numbers for data memory.

Flags No flags are affected.

Format

		Bytes	Cycles	Op Code (Hex)	Address Mode	
					dst	src
opc	dst src	2	10	C3	r	lrr
opc	src dst	2	10	D3	lrr	r
opc	dst src XS	3	12	E7	r	XS[rr]
opc	src dst XS	3	12	F7	XS[rr]	r
opc	dst src XL _L XL _H	4	14	A7	r	XL[rr]
opc	src dst XL _L XL _H	4	14	B7	XL[rr]	r
opc	dst 0000 DA _L DA _H	4	14	A7	r	DA
opc	src 0000 DA _L DA _H	4	14	B7	DA	r

Note:

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

				Bytes	Cycles	Op Code (Hex)	Address Mode	
opc	dst	0001	DA _L DA _H	4	14	A7	dst	src
opc	src	0001	DA _L DA _H	4	14	B7	DA	r

Note:

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

Example

Assume that R0 = 11h, R1 = 34h, R2 = 01h, R3 = 04h. Program memory locations 0103h = 4Fh, 0104h = 1A, 0105h = 6Dh, and 1104h = 88h. External data memory locations 0103h = 5Fh, 0104h = 2Ah, 0105h = 7Dh, and 1104h = 98h.

LDC	R0, @RR2	; R0 ← contents of program memory location 0104h, R0 = Ah, R2 = 01h, R3 = 04h
LDE	R0, @RR2	; R0 ← contents of external data memory location 0104h, R0 = 2Ah, R2 = 01h, R3 = 04h
LDC*	@RR2, R0	; 11h (contents of R0) is loaded into program memory location 0104h (RR2), working registers R0, R2, R3 → no change
LDE	@RR2, R0	; 11h (contents of R0) is loaded into external data memory location 0104h (RR2), working registers R0, R2, R3 → no change
LDC	R0, #01h[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*	#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

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

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*	1105h, R0	; 11h (contents of R0) is loaded into program memory location 1105h, (1105h) ← 11h
LDE	1105h, R0	; 11h (contents of R0) is loaded into external data memory location 1105h, (1105h) ← 11h

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

Load Memory and Decrement

LDCD/LDED dst, src

Operation dst ← src

rr ← rr – 1

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

LDCD references program memory and LDED references external data memory. The assembler causes Irr to be an even number for program memory and an odd number for data memory.

Flags No flags are affected.

Format

		Bytes	Cycles	Op Code (Hex)	Address Mode dst	src
opc	dst src	2	10	E2	r	Irr

Example

Assume that 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.

Load Memory and Increment

LDCI/LDEI dst, src

Operation dst ← src

rr ← rr + 1

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

LDCI refers to program memory and LDEI refers to external data memory. The assembler causes Irr to be an even number for program memory and an odd number for data memory.

Flags No flags are affected.

Format

		Bytes	Cycles	Op Code (Hex)	Address Mode dst	src
opc	dst src	2	10	E3	r	Irr

Example

Assume that 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.

Load Memory with PreDecrement

**LDCPD/
LDEPD** dst, src

Operation $rr \leftarrow rr - 1$
 $dst \leftarrow src$

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

LDCPD refers to program memory and LDEPD refers to external data memory. The assembler causes *Irr* to be an even number for program memory and an odd number for external data memory.

Flags No flags are affected.

Format	Bytes	Cycles	Op Code (Hex)	Address Mode dst src
opc dst src	2	14	F2	Irr r

Example Assume that 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.

Load Memory with PreIncrement

**LDCPI/
LDEPI** dst, src

Operation rr ← rr + 1
dst ← src

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

LDCPI refers to program memory and LDEPI refers to external data memory. The assembler causes Irr to be an even number for program memory and an odd number for data memory.

Flags No flags are affected.

Format		Bytes	Cycles	Op Code (Hex)	Address Mode dst	src
opc	dst src	2	14	F3	lrr	r

Example Assume that 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.

LDEPD @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.

Load Word

LDW dst, src

Operation dst ← src

The contents of the source (a word) are loaded into the destination. The contents of the source are unaffected.

Flags No flags are affected.

Format

			Bytes	Cycles	Op Code (Hex)	Address Mode dst	src
opc	src	dst	3	8	C4	RR	RR
					C5	RR	IR
opc	dst	src	4	8	C6	RR	IML

Example

Assume that R4 = 06h, R5 = 1Ch, R6 = 05h, R7 = 02h, register 00h = Ah, 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 *LDW 00h, 02h* statement loads the contents of the source word 02h, 03h into the destination word 00h, 01h. This instruction 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 multiple addressing modes and formats.

Multiply (Unsigned)

MULT dst, src

Operation $dst \leftarrow dst \times src$

The 8-bit destination operand (even register of the register pair) is multiplied by the source operand (8 bits) and the product (16 bits) is stored in the register pair specified by the destination address. Both operands are treated as unsigned integers.

Flags

- C** Set if result is > 255; cleared otherwise.
- Z** Set if the result is 0; cleared otherwise.
- S** Set if MSB of the result is a 1; cleared otherwise.
- V** Cleared.
- D** Unaffected.
- H** Unaffected.

Format

			Bytes	Cycles	Op Code (Hex)	Address Mode dst	src	
<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px 10px;">opc</td> <td style="padding: 2px 10px;">src</td> <td style="padding: 2px 10px;">dst</td> </tr> </table>	opc	src	dst	3	22	84	RR	R
	opc	src	dst					
	22	85	RR					
22	86	RR	IM					

Example

Assume that 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 01h, #30h → Register 00h = 06h, register 01h = 00h.

In the first example, the *MULT 00h, 02h* statement 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.

Next

NEXT

Operation $PC \leftarrow @ IP$
 $IP \leftarrow IP + 2$

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

Flags No flags are affected.

Format	Bytes	Cycles	Op Code (Hex)
opc	1	10	0F

Example Figure 50 shows an example of how to use the NEXT instruction.

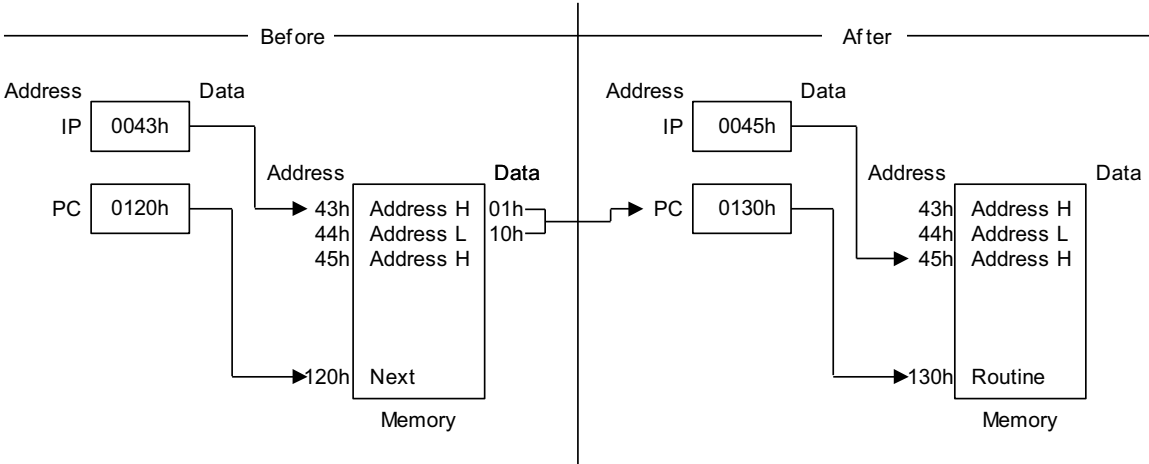


Figure 50. How to Use the NEXT Instruction

No Operation

NOP

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

Flags No flags are affected.

Format

	Bytes	Cycles	Op Code (Hex)
opc	1	4	FF

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

Logical OR

OR dst, src

Operation dst ← dst OR src

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

Flags

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

Format		Bytes	Cycles	Op Code (Hex)	Address Mode dst	src	
opc	dst src	2	4	42	r	r	
			6	43	r	lr	
opc	src	3	6	44	R	R	
			6	45	R	IR	
opc	dst	src	3	6	46	R	IM

Example Assume that R0 = 15h, R1 = 2Ah, R2 = 01h, register 00h = 08h, register 01h = 37h and register 08h = 8Ah.

```

OR   R0, R1      →   R0 = 3Fh, R1 = 2Ah.
OR   R0, @R2     →   R0 = 37h, R2 = 01h, register 01h = 37h.
OR   00h, 01h   →   Register 00h = 3Fh, register 01h = 37h.
OR   01h, @00h  →   Register 00h = 08h, register 01h = 0BFh.
OR   00h, #02h  →   Register 00h = 0Ah.

```

In the first example, if working register R0 contains the value 15h and register R1 the value 2Ah, the *OR R0, R1* statement 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 multiple addressing modes and formats.

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 are affected.

Format		Bytes	Cycles	Op Code (Hex)	Address Mode
opc	dst	2	8	50	R
			8	51	IR

Example Assume that 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 *POP 00h* statement 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.

Pop User Stack (Decrementing)

POPUD dst, src

Operation dst ← src
IR ← IR – 1

This instruction is used for user-defined stacks in the register file. The contents of the register file location addressed by the user stack pointer are loaded into the destination. The user stack pointer is then decremented.

Flags No flags are affected.

Format			Bytes	Cycles	Op Code (Hex)	Address Mode	
						dst	src
opc	src	dst	3	8	92	R	IR

Example Assume that Register 00h = 42h (user stack pointer register), register 42h = 6Fh and register 02h = 70h.

POPUD 02h, @00h → Register 00h = 41h, register 02h = 6Fh, register 42h = 6Fh.

If general register 00h contains the value 42h and register 42h the value 6Fh, the *POPUD 02h, @00h* statement loads the contents of register 42h into the destination register 02h. The user stack pointer is then decremented by one, leaving the value 41h.

Pop User Stack (Incrementing)

POPUI dst, src

Operation dst ← src
IR ← IR + 1

The POPUI instruction is used for user-defined stacks in the register file. The contents of the register file location addressed by the user stack pointer are loaded into the destination. The user stack pointer is then incremented.

Flags No flags are affected.

			Bytes	Cycles	Op Code (Hex)	Address Mode	
						dst	src
opc	src	dst	3	8	93	R	IR

Example Assume that 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 *POPUI 02h, @00h* statement loads the value 70h into the destination general register 02h. The user stack pointer (register 00h) is then incremented by one, changing its value from 01h to 02h.

Push to Stack

PUSH src

Operation $SP \leftarrow SP - 1$

$@SP \leftarrow src$

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

Flags No flags are affected.

Format

		Bytes	Cycles	Op Code (Hex)	Address Mode dst
opc	src	2	8 (internal clock)	70	R
			8 (external clock)		
			8 (internal clock)	71	IR
			8 (external clock)		

Example Assume that Register 40h = 4Fh, register 4Fh = 0AAh, SPH = 00h and SPL = 00h.

PUSH 40h → Register 40h = 4Fh, stack register 0FFh = 4Fh, SPH = 0FFh, SPL = 0FFh.

PUSH @40h → Register 40h = 4Fh, register 4Fh = 0AAh, stack register 0FFh = 0AAh, SPH = 0FFh, SPL = 0FFh.

In the first example, if the stack pointer contains the value 0000h, and general register 40h the value 4Fh, the *PUSH 40h* statement 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.

Push User Stack (Decrementing)

PUSHUD dst, src

Operation $IR \leftarrow IR - 1$
 $dst \leftarrow src$

This instruction is used to address user-defined stacks in the register file. PUSHUD decrements the user stack pointer and loads the contents of the source into the register addressed by the decremented stack pointer.

Flags No flags are affected.

Format			Bytes	Cycles	Op Code (Hex)	dst	src
opc	dst	src	3	8	82	IR	R

Example Assume that 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 *PUSHUD @00h, 01h* statement decrements the user stack pointer by one, leaving the value 02h. The 01h register value, 05h, is then loaded into the register addressed by the decremented user stack pointer.

Push User Stack (Incrementing)

PUSHUI dst, src

Operation IR ← IR + 1
dst ← src

This instruction is used for user-defined stacks in the register file. PUSHUI increments the user stack pointer and then loads the contents of the source into the register location addressed by the incremented user stack pointer.

Flags No flags are affected.

Format

			Bytes	Cycles	Op Code (Hex)	dst	src
opc	dst	src	3	8	83	IR	R

Example Assume that 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 *PUSHUI @00h, 01h* statement increments the user stack pointer by one, leaving the value 04h. The 01h register value, 05h, is then loaded into the location addressed by the incremented user stack pointer.

Reset Carry Flag

RCF RCF

Operation $C \leftarrow 0$

The carry flag is cleared to logic 0, regardless of its previous value.

Flags **C** Cleared to 0.
No other flags are affected.

Format

	Bytes	Cycles	Op Code (Hex)
opc	1	4	CF

Example Assume that $C = 1$ or 0 . The RCF instruction clears the carry flag (C) to logic 0.

Return

RET

Operation $PC \leftarrow @SP$
 $SP \leftarrow SP + 2$

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

Flags No flags are affected.

Format	Bytes	Cycles	Op Code (Hex)
opc	1	8 (internal stack) 10 (internal stack)	AF

Example Assume that $SP = 00FCh$, $(SP) = 101Ah$, and $PC = 1234$.

RET \rightarrow $PC = 101Ah$, $SP = 00FEh$.

The *RET* statement pops the contents of stack pointer location $00FCh$ ($10h$) into the high byte of the program counter. The stack pointer then pops the value in location $00FEh$ ($1Ah$) into the PC's low byte and the instruction at location $101Ah$ is executed. The stack pointer now points to memory location $00FEh$.

Rotate Left

RL dst

Operation $C \leftarrow \text{dst}(7)$
 $\text{dst}(0) \leftarrow \text{dst}(7)$
 $\text{dst}(n + 1) \leftarrow \text{dst}(n), n = 0 - 6$

The contents of the destination operand are rotated left one bit position. The initial value of bit 7 is moved to the bit 0 (LSB) position and also replaces the carry flag.

Figure 51 shows how bits rotate left.

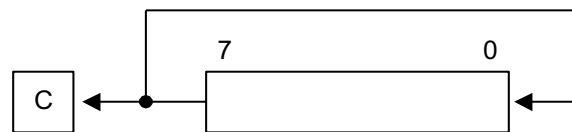


Figure 51. Rotate Left

Flags

- C** Set if the bit rotated from the most significant bit position (bit 7) is 1.
- Z** Set if the result is 0; cleared otherwise.
- S** Set if the result bit 7 is set; cleared otherwise.
- V** Set if arithmetic overflow occurred; cleared otherwise.
- D** Unaffected.
- H** Unaffected.

Format

		Bytes	Cycles	Op Code (Hex)	Address Mode		
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px 10px;">opc</td> <td style="padding: 2px 10px;">dst</td> </tr> </table>	opc	dst		2	4	90	R
	opc	dst					
			4	91	IR		

Example

Assume that Register 00h = 0AAh, register 01h = 02h and register 02h = 17h.

RL 00h → Register 00h = 55h, C = 1.

RL @01h → Register 01h = 02h, register 02h = 2Eh, C = 0.

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

Rotate Left through Carry

RCL dst

Operation $dst(0) \leftarrow C$

$C \leftarrow dst(7)$

$dst(n + 1) \leftarrow dst(n), n = 0 - 6$

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 0.

Figure 52 shows how bits rotate left through carry.

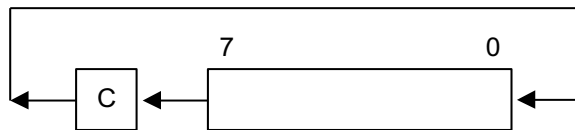


Figure 52. Rotate Left through Carry

Flags

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

Format

		Bytes	Cycles	Op Code (Hex)	Address Mode
opc	dst	2	4	10	R
			4	11	IR

Example

Assume that register 00h = 0AAh, register 01h = 02h, and register 02h = 17h, C = 0.

RCL 00h Register 00h = 54h, C = 1.

RCL @01h Register 01h = 02h, register 02h = 2Eh, C = 0.

In the first example, if general register 00h contains the value 0AAh (10101010b), the *RLC 00h* statement rotates 0AAh one bit position to the left. The initial value of bit 7 sets the carry flag and the initial value of the C flag replaces bit 0 of register 00h, leaving the value 55h (01010101b). The MSB of register 00h resets the carry flag to 1 and sets the overflow flag.

Rotate Right

RR dst

Operation $C \leftarrow \text{dst}(0)$
 $\text{dst}(7) \leftarrow \text{dst}(0)$
 $\text{dst}(n) \leftarrow \text{dst}(n + 1), n = 0-6$

The contents of the destination operand are rotated right one bit position. The initial value of bit 0 (LSB) is moved to bit 7 (MSB) and also replaces the carry flag (C).

Figure 53 shows how bits rotate right.

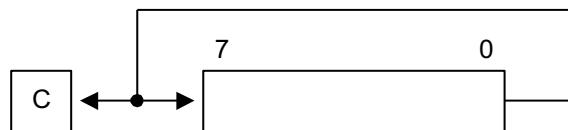


Figure 53. Rotate Right

Flags

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

Format

		Bytes	Cycles	Op Code (Hex)	Address Mode	
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px 10px;">opc</td> <td style="padding: 2px 10px;">dst</td> </tr> </table>	opc	dst	2	4	E0	R
	opc	dst				
4	4	E1	IR			

Example

Assume that register 00h = 31h, register 01h = 02h, and register 02h = 17h.

RR 00h Register 00h = 98h, C = 1.
 RR @01h Register 01h = 02h, register 02h = 8Bh, C = 1.

In the first example, if general register 00h contains the value 31h (00110001b), the *RR 00h* statement rotates this value one bit position to the right. The initial value of bit 0 is moved to bit 7, leaving the new value 98h (10011000b) in the destination register. The initial bit 0 resets the C flag to 1 and the sign flag and overflow flag are also 1.

Rotate Right through Carry

RRC dst

Operation $dst(7) \leftarrow C$

$C \leftarrow dst(0)$

$dst(n) \leftarrow dst(n + 1), n = 0-6$

The contents of the destination operand and the carry flag are rotated right one bit position. The initial value of bit 0 (LSB) replaces the carry flag; the initial value of the carry flag replaces bit 7 (MSB).

Figure 54 shows how bits rotate right through carry.

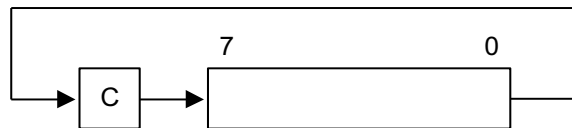


Figure 54. Rotate Right through Carry

Flags

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

Format

		Bytes	Cycles	Op Code (Hex)	Address Mode
opc	dst	2	4	C0	R
			4	C1	IR

Example

Assume that Register 00h = 55h, register 01h = 02h, register 02h = 17h, and C = 0.

RRC 00h Register 00h = 2Ah, C = 1.

RRC @01h Register 01h = 02h, register 02h = 0Bh, C = 1.

In the first example, if general register 00h contains the value 55h (01010101b), the *RRC 00h* statement rotates this value one 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 instruction leaves the new value 2Ah (00101010b) in destination register 00h. The sign flag and overflow flag are both cleared to 0.

Select Bank0

SB0

Operation BANK ← 0

The SB0 instruction clears the bank address flag in the Flags Register (FLAGS.0) to logic 0, selecting Bank0 register addressing in the Set1 area of the register file.

Flags No flags are affected.

Format

	Bytes	Cycles	Op Code (Hex)
opc	1	4	4F

Example The *SB0* statement clears FLAGS.0 to 0, selecting Bank0 register addressing.

Select Bank1

SB1

Operation BANK ← 1

The SB1 instruction sets the bank address flag in the Flags Register (FLAGS.0) to logic 1, selecting Bank1 register addressing in the Set1 area of the register file. (Bank1 is not implemented in some KS88-series microcontrollers.)

Flags No flags are affected.

Format

	Bytes	Cycles	Op Code (Hex)
opc	1	4	5F

Example The *SB1* statement sets FLAGS.0 to 1, selecting Bank1 register addressing, if implemented.

Subtract with Carry

SBC dst, src

Operation $dst \leftarrow dst - src - c$

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

Flags

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

Format		Bytes	Cycles	Op Code (Hex)	Address Mode dst	src	
opc	dst src	2	4	32	r	r	
			6	33	r	lr	
opc	src	3	6	34	R	R	
			6	35	R	IR	
opc	dst	src	3	6	36	R	IM

Example Assume that $R1 = 10h$, $R2 = 03h$, $C = 1$, register $01h = 20h$, register $02h = 03h$ and register $03h = 0Ah$.

SBC R1, R2 R1 = 0Ch, R2 = 03h.

SBC R1, @R2 R1 = 05h, R2 = 03h, register 03h = 0Ah.

SBC 01h, 02h Register 01h = 1Ch, register 02h = 03h.

SBC 01h, @02h Register 01h = 15h, register 02h = 03h, register 03h = 0Ah.
SBC 01h, #8Ah Register 01h = 95h; C, S, and V = 1.

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

Set Carry Flag

SCF

Operation $C \leftarrow 1$

The carry flag (C) is set to logic 1, regardless of its previous value.

Flags **C** Set to 1.

No other flags are affected.

Format

	Bytes	Cycles	Op Code (Hex)
opc	1	4	DF

Example The *SCF* statement sets the carry flag to logic 1.

Shift Right Arithmetic

SRA dst

Operation $dst(7) \leftarrow dst(7)$

$C \leftarrow dst(0)$

$dst(n) \leftarrow dst(n + 1), n = 0-6$

An arithmetic shift-right of one bit position is performed on the destination operand. Bit 0 (the LSB) replaces the carry flag. The value of bit 7 (the sign bit) is unchanged and is shifted into bit position 6.

Figure 55 shows how bits shift right.

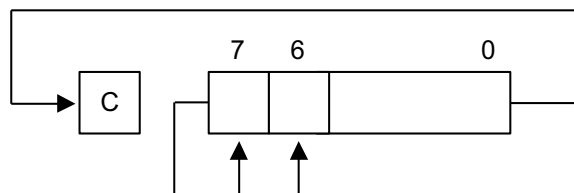


Figure 55. Shift Right

Flags

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

Format

		Bytes	Cycles	Op Code (Hex)	Address Mode
opc	dst	2	4	D0	R
			4	D1	IR

Example

Assume that Register 00h = 9Ah, register 02h = 03h, register 03h = 0BCh, and C = 1.

SRA 00h Register 00h = 0CD, C = 0.
SRA @02h Register 02h = 03h, register 03h = 0DEh, C = 0.

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

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) = 1 and src(0) = 0 then:	RP1(3–7)	src (3–7)
	If src(1) = 1 and src(0) = 0 then:	RP0 (4–7)	src (4–7)
		RP0 (3)	0
		RP1 (4–7)	src (4–7)
		RP1 (3)	1

The sources data bits one and zero (LSB) determine whether to write one or both of the register pointers, RP0 and RP1. Bits 3–7 of the selected register pointer are written unless both register pointers are selected. RP0.3 is then cleared to logic 0 and RP1.3 is set to logic 1.

Flags No flags are affected.

Format			Bytes	Cycles	Op Code	Address Mode
					(Hex)	src
	opc	src	2	4	31	IM

Example The *SRP #40h* statement sets register pointer 0 (RP0) at location 0D6h to 40h and register pointer 1 (RP1) at location 0D7h to 48h.

The *SRP0 #50h* statement sets RP0 to 50h, and the *SRP1 #68h* statement sets RP1 to 68h.

Stop Operation

STOP

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

Flags No flags are affected.

Format

	Bytes	Cycles	Op Code (Hex)	Address Mode dst	src
opc	1	4	7F	–	–

Example The *Stop* statement halts all microcontroller operations.

Subtract

SUB dst, src

Operation $dst \leftarrow dst - src$

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

Flags

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

Format	Bytes		Cycles	Op Code (Hex)	Address Mode			
	opc	dst src			dst	src		
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px;">opc</td> <td style="padding: 2px;">dst src</td> </tr> </table>	opc	dst src	2	4	22	r	r	
	opc	dst src						
r	lr							
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px;">opc</td> <td style="padding: 2px;">src</td> <td style="padding: 2px;">dst</td> </tr> </table>	opc	src	dst	3	6	24	R	R
	opc	src	dst					
25	R	IR						
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px;">opc</td> <td style="padding: 2px;">dst</td> <td style="padding: 2px;">src</td> </tr> </table>	opc	dst	src	3	6	26	R	IM
opc	dst	src						

Example Assume that R1 = 12h, R2 = 03h, register 01h = 21h, register 02h = 03h, register 03h = 0Ah.

SUB	R1, R2	R1 = 0Fh, R2 = 03h.
SUB	R1, @R2	R1 = 08h, R2 = 03h.
SUB	01h, 02h	Register 01h = 1Eh, register 02h = 03h.
SUB	01h, @02h	Register 01h = 17h, register 02h = 03h.

SUB 01h, #90h Register 01h = 91h; C, S, and V = 1.

SUB 01h, #65h Register 01h = 0BCh; C and S = 1, V = 0.

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

Swap Nibbles

SRA 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.

Figure 56 shows how to swap nibbles.

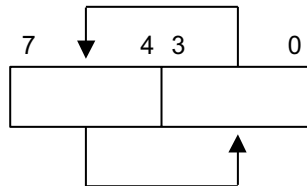


Figure 56. Swap Nibbles

Flags

- C** Undefined.
- Z** Set if the result is 0; cleared otherwise.
- S** Set if the result bit 7 is set; cleared otherwise.
- V** Undefined.
- D** Unaffected.
- H** Unaffected.

Format

		Bytes	Cycles	Op Code (Hex)	Address Mode		
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px 10px;">opc</td> <td style="padding: 2px 10px;">dst</td> </tr> </table>	opc	dst		2	4	F0	R
	opc	dst					
			4	F1	IR		

Example

Assume that Register 00h = 3Eh, register 02h = 03h, and register 03h = 0A4h.

SWAP 00h Register 00h = 0E3h.

SWAP @02h Register 02h = 03h, register 03h = 4Ah.

In the first example, if general register 00h contains the value 3Eh (00111110b), the *SWAP 00h* statement swaps the lower and upper four bits (nibbles) in the 00h register, leaving the value 0E3h (11100011b).

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

Flags

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

Format		Bytes	Cycles	Op Code (Hex)	Address Mode dst	src
opc	dst src	2	4	62	r	r
			6	63	r	lr
opc	src	3	6	64	R	R
			6	65	R	IR
opc	dst	src	3	66	R	IM

Example Assume that R0 = 0C7h, R1 = 02h, R2 = 12h, register 00h = 2Bh, register 01h = 02h and register 02h = 23h.

TCM	R0, R1	R0 = 0C7h, R1 = 02h, Z = 1.
TCM	R0, @R1	R0 = 0C7h, R1 = 02h, register 02h = 23h, Z = 0.
TCM	00h, 01h	Register 00h = 2Bh, register 01h = 02h, Z = 1.
TCM	00h, @01h	Register 00h = 2Bh, register 01h = 02h, register 02h = 23h, Z = 1.
TCM	00h, #34	Register 00h = 2Bh, Z = 0.

In the first example, if working register R0 contains the value 0C7h (11000111b) and register R1 the value 02h (0000010b), the *TCM R0, R1* statement 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.

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

Format		Bytes	Cycles	Op Code (Hex)	Address Mode dst	src	
opc	dst src	2	4	72	r	r	
			6	73	r	lr	
opc	src	3	6	74	R	R	
			6	75	R	IR	
opc	dst	src	3	6	76	R	IM

Example Assume that R0 = 0C7h, R1 = 02h, R2 = 18h, register 00h = 2Bh, register 01h = 02h and register 02h = 23h.

TM	R0, R1	R0 = 0C7h, R1 = 02h, Z = 0.
TM	R0, @R1	R0 = 0C7h, R1 = 02h, register 02h = 23h, Z = 0.
TM	00h, 01h	Register 00h = 2Bh, register 01h = 02h, Z = 0.
TM	00h, @01h	Register 00h = 2Bh, register 01h = 02h, register 02h = 23h, Z = 0.
TM	00h, #54h	Register 00h = 2Bh, Z = 1.

In the first example, if working register R0 contains the value 0C7h (11000111b) and register R1 the value 02h (0000010b), the *TM R0, R1* statement 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.

Wait for Interrupt

WFI

Operation The CPU is effectively halted until an interrupt occurs, except that DMA transfers can still take place during this wait state. The WFI status can be released by an internal interrupt, including a fast interrupt.

Flags No flags are affected.

Format	Bytes	Cycles	Op Code (Hex)
opc	1	4n	3F

Note: n = 1, 2, 3, etc.

Example Figure 57 presents a sample program structure that depicts the sequence of operations that follow a *WFI* statement.

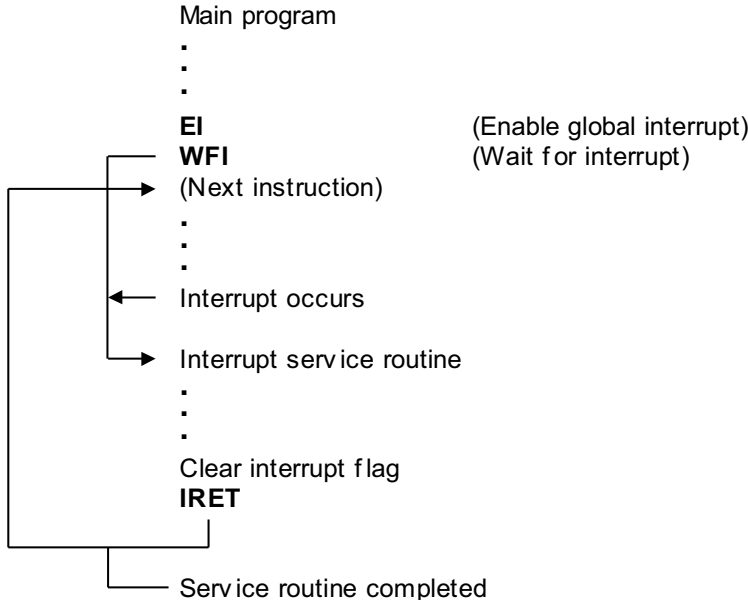


Figure 57. Sample Program Structure

Logical Exclusive OR

XOR dst, src

Operation dst ← dst XOR src

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

Flags

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

Format		Bytes	Cycles	Op Code (Hex)	Address Mode dst	src	
opc	dst src	2	4	B2	r	r	
			6	B3	r	lr	
opc	src	3	6	B4	R	R	
			6	B5	R	IR	
opc	dst	src	3	6	B6	R	IM

Example Assume that R0 = 0C7h, R1 = 02h, R2 = 18h, register 00h = 2Bh, register 01h = 02h and register 02h = 23h.

XOR	R0, R1	R0 = 0C5h, R1 = 02h.
XOR	R0, @R1	R0 = 0E4h, R1 = 02h, register 02h = 23h.
XOR	00h, 01h	Register 00h = 29h, register 01h = 02h.
XOR	00h, @01h	Register 00h = 08h, register 01h = 02h, register 02h = 23h.
XOR	00h, #54h	Register 00h = 7Fh.

In the first example, if working register R0 contains the value 0C7h and if register R1 contains the value 02h, the *XOR R0, R1* statement logically exclusive-ORs the R1 value with the R0 value and stores the result (0C5h) in the destination register, R0.

Chapter 8. Clock, Power, and Reset Circuits

The clock frequency for the S3F80PB MCU can be generated by an external crystal or supplied by an external clock source. This clock frequency can range from 1 MHz to 8 MHz. The maximum CPU clock frequency, as determined by CLKCON Register, is 8 MHz. The X_{IN} and X_{OUT} pins connect the external oscillator or clock source to the on-chip clock circuit.

Typically, application systems contain a resistor and two separate capacitors across the power pins to suppress high-frequency noise and provide bulk charge storage for the overall system. When the nRESET pin input goes High after V_{DD} has risen above the LVD threshold voltage, the reset operation is released. External reset circuit must be attached in the application systems.

8.1. System Clock Circuit

The main system clock circuit, shown in Figure 58, features the following components:

- External crystal or ceramic resonator oscillation source (or an external clock)
- Oscillator stop and wake-up functions
- Programmable frequency divider for the CPU clock (f_{OSC} divided by 1, 2, 8, or 16)
- Clock Circuit Control Register (CLKCON)

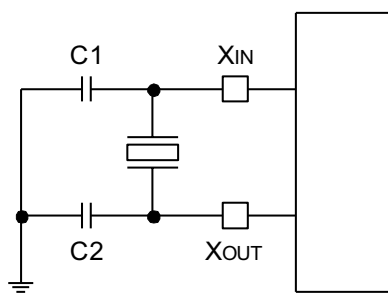


Figure 58. Main Oscillator Circuit

The circuit for the external clock is shown in Figure 59.

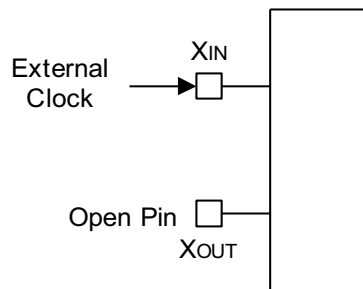


Figure 59. External Clock Circuit

8.2. Clock Status During Power-Down Modes

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

- In Stop Mode, the main oscillator is halted. When the Stop Mode is released, the oscillator starts by a reset operation or by an external interrupt. To enter Stop Mode, the Stop Control (STOPCON) Register must be loaded with value, #0A5h before Stop instruction execution. After recovering from Stop Mode by a reset or an external interrupt, the STOPCON Register is automatically cleared.
- In Idle Mode, the internal clock signal is gated away from the CPU, but continues to be supplied to the interrupt structure, Timer 0, Timer 1, Counter A, etc. Idle Mode is released by a reset or by an interrupt (external or internally generated).

A block diagram of the system clock circuit is shown in Figure 60.

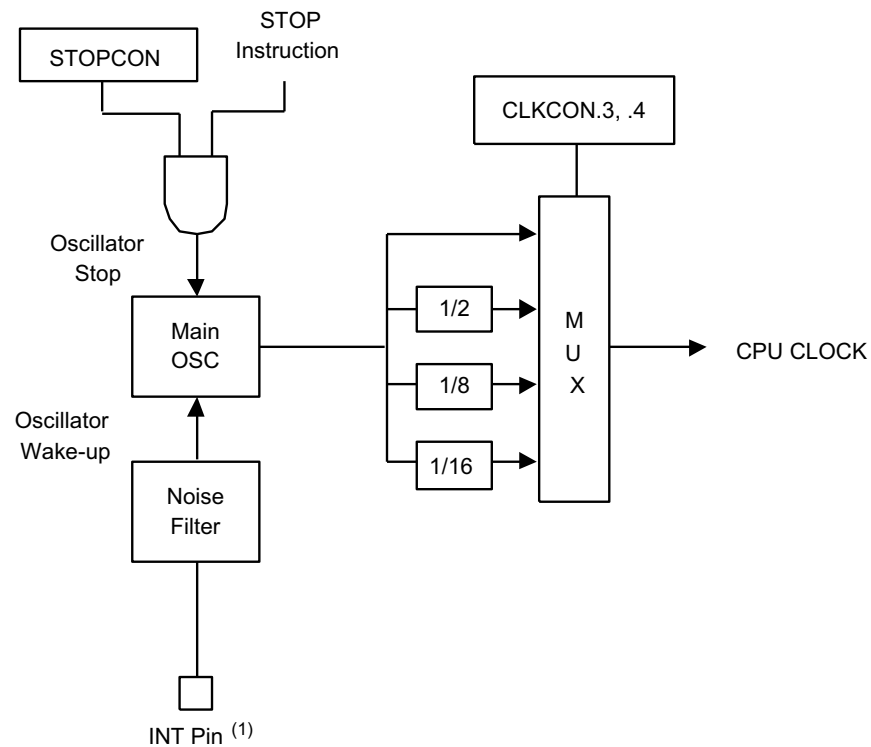


Figure 60. System Clock Circuit Diagram

-
- **Notes:**
1. An external interrupt with an RC-delay noise filter (for the S3F80PB MCU INT0–9) is fixed to release Stop Mode and awaken the main oscillator.
 2. The 3-bit CLKCON signature code (CLKCON2–CLKCON.0) has no meaning because the S3F80PB does not contain a subsystem clock.
-

8.3. System Clock Control Register

The System Clock Control (CLKCON) Register is located in address D4h, Set1, Bank0. This register is read/write-addressable and offers the following functions:

- Oscillator frequency divide-by value

The CLKCON.7–.5 and CLKCON.2–.0 bits are not used on the S3F80PB MCU. After a reset, the main oscillator is activated, and $f_{OSC/16}$ (i.e., the slowest clock speed) is selected as the CPU clock. If necessary, increase the CPU clock speed to f_{OSC} , $f_{OSC/2}$, $f_{OSC/8}$, or $f_{OSC/16}$. The contents of the System Clock Control (CLKCON) Register are described in Table 30.

Table 30. System Clock Control Register (CLKCON; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	D4h							

Note: R/W = read/write.

Bit	Description
[7:5]	Reserved These bits are reserved and must be set to 000.
[4:3]	CPU Clock (System Clock) Selection Bits* 00: $f_{OSC} / 16$. 01: $f_{OSC} / 8$. 10: $f_{OSC} / 2$. 11: f_{OSC} (not divided).
[2:0]	Reserved These bits are reserved and must be set to 000.

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

Typically, application systems contain a resistor and two separate capacitors across the power pins. R1 and C1 must be located as near to the MCU's power pins as practical to suppress high-frequency noise. C2 should be a bulk electrolytic capacitor to provide bulk charge storage for the overall system. Zilog recommends setting $R1 = 10\Omega$, $C1 = 0.1\mu F$ and $C2 = 100\mu F$. The V_{DD} power circuit is shown in Figure 61.

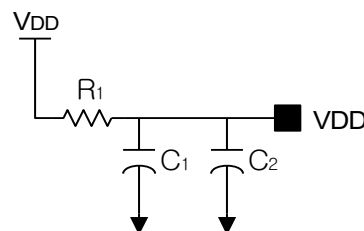


Figure 61. Power Circuit

When the nRESET pin input goes High, the reset operation is released. External reset circuit must be attached in the application systems for initialization. Zilog recommends setting $R1 = 1M\Omega$ and $C1 = 0.1\mu F$. The nRESET circuit is shown in Figure 62.

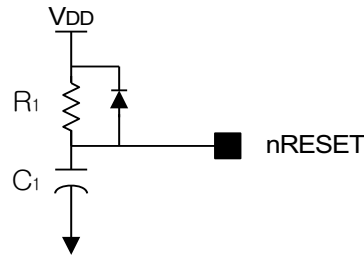


Figure 62. nRESET Circuit

Table 31 presents guidelines for setting the stability of the V_{DD} rise and fall times.

Table 31. Falling and Rising Time of Operating Voltage

V_{DD} Slope	Min.	Typ.	Max.	Unit
R_{VF} 3.6V to 0.0V	–	–	20	V/ms
R_{VR} 0.0V to 3.6V	–	–	4	

Note: R_{VF} = falling; R_{VR} = rising.

Chapter 9. Reset

Resetting the MCU is the function to start processing by generating reset signal using several reset schemes. During reset, most control and status are forced to initial values and the program counter is loaded from the reset vector. If using the S3F80PB MCU, the reset vector can be changed by the Smart Option, as discussed in the [Smart Option](#) section on page 21.

9.1. Reset Sources

The S3F80PB MCU features six different system reset sources as following:

The External Reset Pin (nRESET). When the nRESET pin transiting from the low input level of the reset pin (V_{IL}) to the high input level of the reset pin (V_{IH}), the reset pulse is generated on the condition of $V_{DD} \geq V_{LVD}$ in any operation mode.

Watchdog Timer (WTD). When the watchdog timer enables in normal operation, a reset is generated whenever the basic timer overflow occurs.

Low Voltage Detect (LVD). When a change in V_{DD} affects LVD operation during normal operating mode, a reset occurs.

Internal Power-On Reset (IPOR). When the V_{DD} is changed in condition for IPOR operation, a reset is generated.

External Interrupt (INT0–INT9). When the RESET control bit is set to 0 (i.e., using the Smart Option at address 03Fh), the S3F80PB MCU is in Stop Mode, and external interrupts are enabled, any external interrupts by P0 and P2 will generate a reset signal.

Stop Error Detection & Recovery (SED & R). When the RESET control bit is set to 0 (i.e., using the Smart Option bit[7] at 03Fh) and the MCU is in a stopped or abnormal state, the falling edge input of P0 or P2.4–P2.7 generates the reset signal regardless of external interrupt enable or disable settings.

These reset sources are diagrammed in Figure 63; source lines labeled 1 through 6 correspond to the following stipulations:

1. The rising edge detection of the LVD circuit while rising of V_{DD} passes the level of V_{LVD} .
2. When the POR circuit detects V_{DD} below V_{POR} , reset is generated by internal power-on reset.
3. Basic timer overflow for the watchdog timer; see the [Basic Timer and Timer 0](#) chapter on page 261.

4. The reset pulse generation by transiting of reset pin (nRESET) from low level to high level on the condition that V_{DD} is higher level state than V_{LVD} (Low level Detect Voltage).
5. When the RESET control bit (via the Smart Option at address 03Fh) is set to 0 and the S3F80PB MCU is in Stop Mode, external interrupts input via P0 and P2 generate a reset signal.
6. When the RESET control bit (via the Smart Option at address 03Fh) are set to 0 and the S3F80PB MCU is in Stop Mode or an abnormal state, the falling edge input of P0 and P2.4–P2.7 generates the reset signal regardless of external interrupt enable/disable settings.

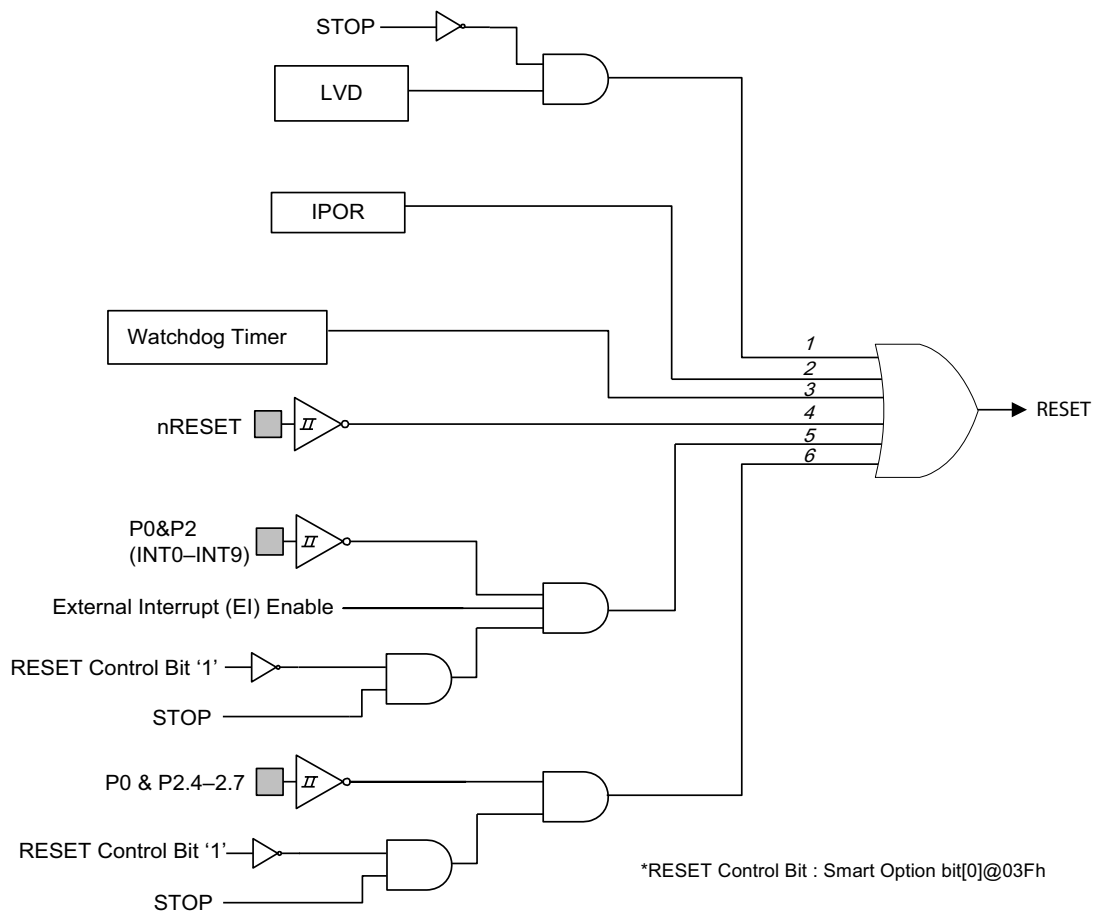


Figure 63. Reset Sources of the S3F80PB MCU

Figure 64 shows a block diagram of the S3F80PB MCU's Reset functions.

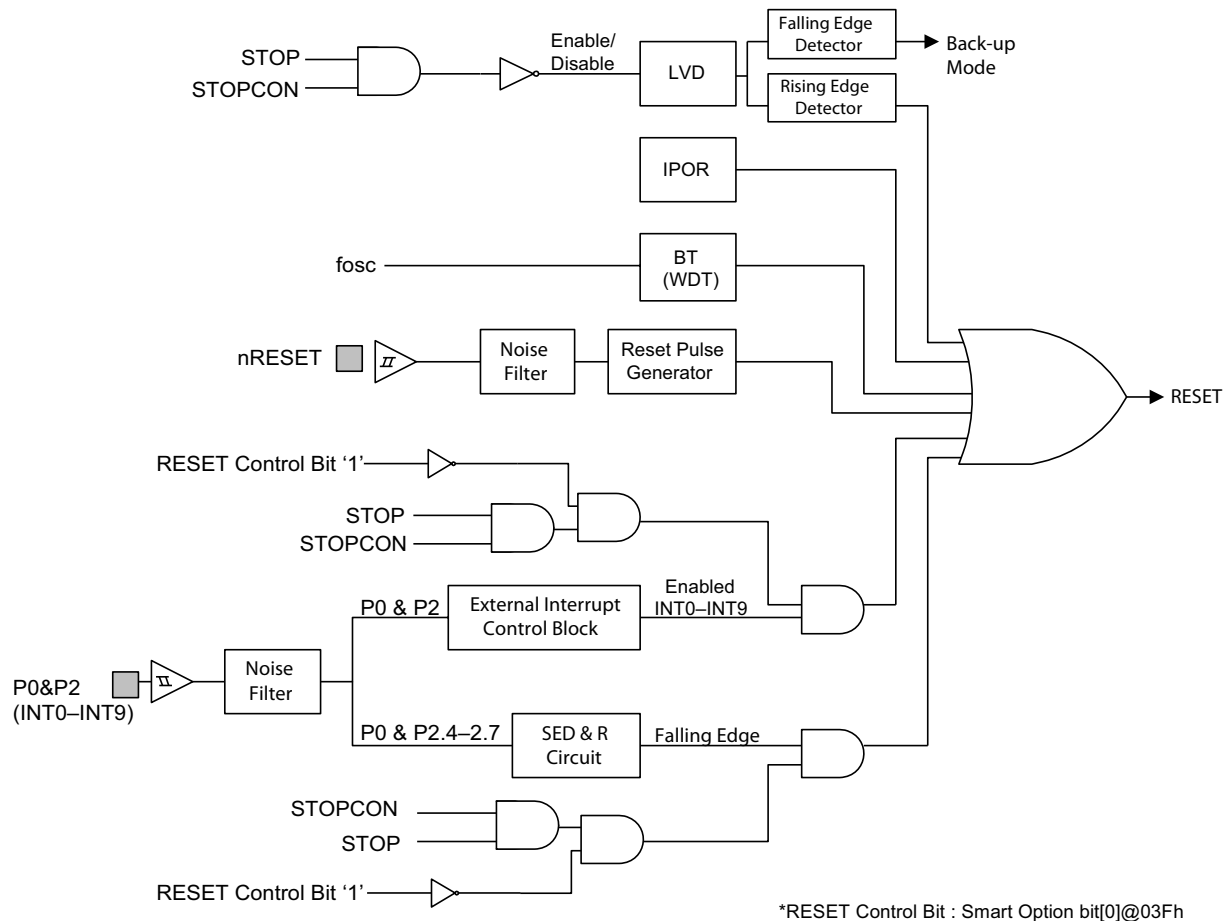


Figure 64. Reset Block Diagram of the S3F80PB MCU

9.2. Reset Mechanism

The interlocking work of the reset pin and the LVD circuit supplies two operating modes: Backup Mode input, and system reset input. Backup Mode input automatically causes a chip stop, when the reset pin is set to low level or the voltage at V_{DD} is lower than V_{LVD} . When the reset pin is at a high state and the LVD circuit detects the rising edge of V_{DD} on the point V_{LVD} , the reset pulse generator causes a reset pulse, and system reset occurs. When the operating mode is Stop Mode, the LVD circuit is disabled to reduce the current consumption under $5 \mu A$ (at $V_{DD} = 3.6V$). Therefore, although the voltage at V_{DD} is lower than V_{LVD} , the S3F80PB MCU does not go into Backup Mode when the operating state is in Stop Mode and the reset pin is at a High level (i.e., $V_{RESET} > V_{IH}$).

9.3. External Reset Pin

When the nRESET pin transiting from V_{IL} (i.e., the low input level of the reset pin) to V_{IH} (the high input level of the reset pin), the reset pulse is generated on the condition that $V_{DD} \geq V_{LVD}$.

9.4. Watchdog Timer Reset

A watchdog timer that can enable recovery from an abnormal function to normal operation is built into the S3F80PB MCU. This watchdog timer generates a system reset signal if the Basic Timer Counter (BTCNT) is not cleared within a specific time by the program. To learn more about the watchdog timer function, see the [Basic Timer and Timer 0](#) chapter on page 261.

9.5. LVD Reset

The Low Voltage Detect Circuit (LVD) is built on the S3F80PB device to generate a system reset. LVD is disabled in Stop Mode. When the voltage at V_{DD} falls and passes V_{LVD} , the S3F80PB MCU enters Backup Mode at the moment $V_{DD} = V_{LVD}$. As the voltage at V_{DD} rises, the reset pulse occurs at the moment $V_{DD} \geq V_{LVD}$. Figure 65 shows the Reset block diagram in Stop Mode.

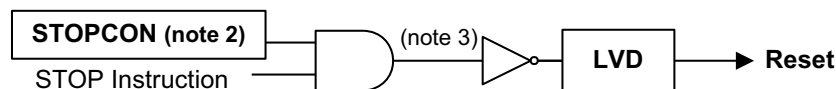


Figure 65. Reset Block Diagram by LVD for the S3F80PB MCU in Stop Mode

-
- **Notes:**
1. LVD is disabled in Stop Mode. LVD always operates in any other operation modes.
 2. The CPU can enter Stop Mode by setting the Stop Control (STOPCON) Register into 0A5h before executing a Stop instruction.
 3. This signal is output as it relates to Stop Mode. If STOPCON contains 0A5h and a Stop instruction is executed, the output signal allows the S3F80PB MCU to enter Stop Mode. Therefore, of two statuses, one is Stop Mode; the other is not Stop Mode.
-

9.6. Internal Power-On Reset

The power-on reset circuit is built on the S3F80PB device. When the power is initially applied to the MCU, or when V_{DD} drops below V_{POR} , the POR circuit holds the MCU in reset until V_{DD} has risen above the V_{LVD} level.

Figure 66 shows the Timing diagram for the internal power-on reset circuit.

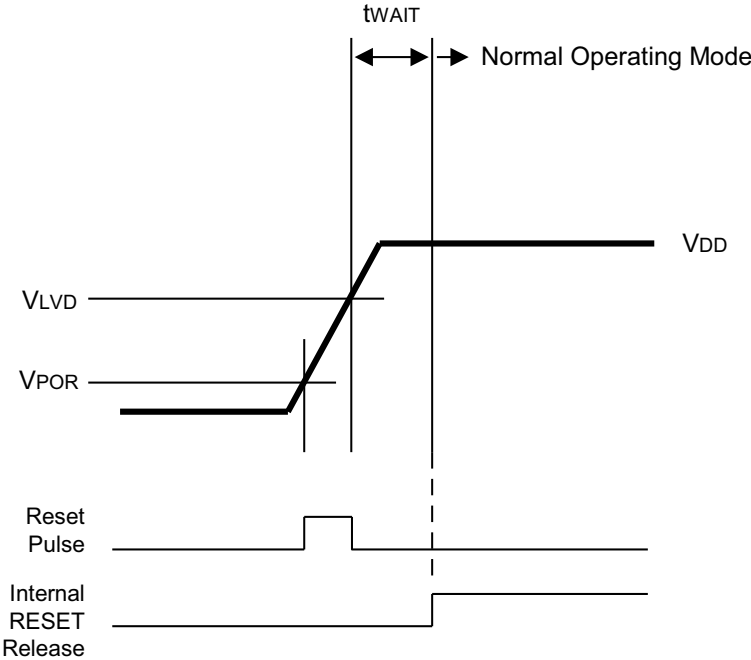


Figure 66. Timing Diagram for Internal Power-On Reset Circuit

Figure 67 shows the Reset Timing diagram for the S3F80PB MCU in Stop Mode by IPOR.

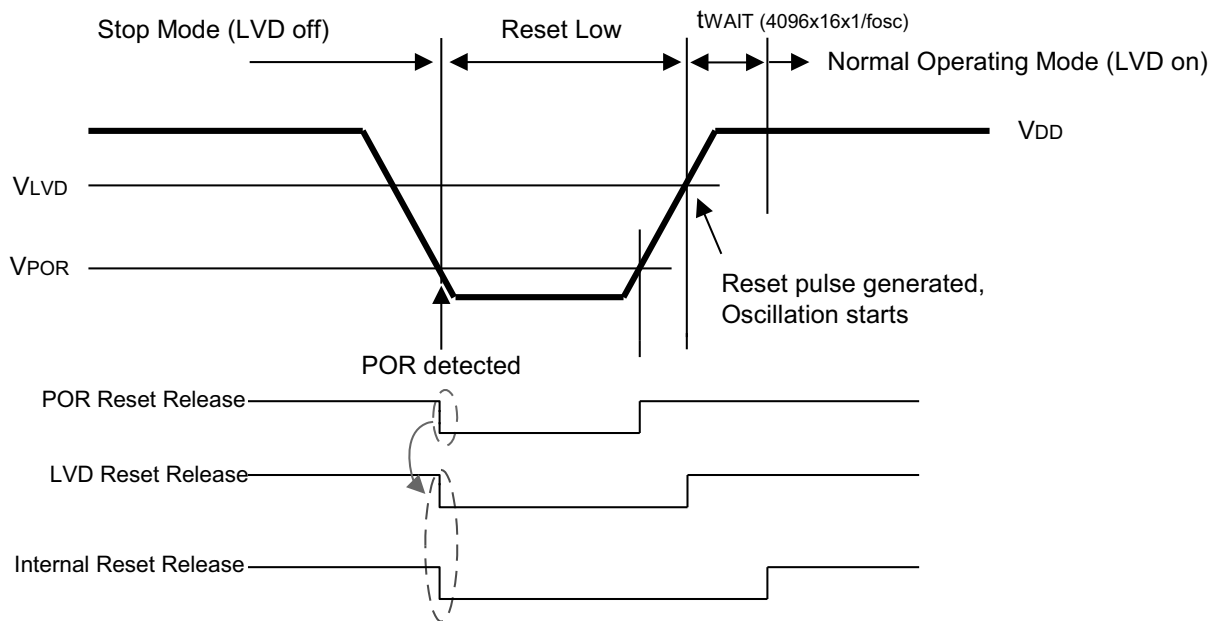


Figure 67. Reset Timing Diagram for the S3F80PB MCU in Stop Mode by IPOR

► **Note:** If $V_{\text{RESET}} > V_{\text{IH}}$, the operating status is in Stop Mode and the LVD circuit is disabled in the S3F80PB MCU.

9.7. External Interrupt Reset

When the RESET control bit (via the Smart Option at address 03Fh) is set to 0 and the chip is in Stop Mode, if an external interrupt occurs among the enabled external interrupt sources, from INT0 to INT9, a reset signal is generated.

9.8. Stop Error Detection & Recovery

When the RESET control bit (via the Smart Option at address 03Fh) is set to 0 and the chip is in a stop or abnormal state, the falling edge input of P0 and P2.4–P2.7 generate the reset signal.

9.9. External Reset Pin

When the nRESET pin transiting from V_{IL} (i.e., the low input level of the reset pin) to V_{IH} (the high input level of the reset pin), the reset pulse is generated on the condition of $V_{DD} \geq V_{LVD}$ in any operational mode. Table 32 lists the possible reset conditions.

Table 32. Reset Conditions in Stop Mode

Condition				
Slope of V_{DD}	V_{DD}	The Voltage Level of Reset Pin (V_{RESET})	Reset Source	System Reset
Rising up from $V_{POR} < V_{DD} < V_{LVD}$	$V_{DD} \geq V_{LVD}$	$V_{RESET} \geq V_{IH}$	–	No system reset
	$V_{DD} > V_{LVD}$	$V_{RESET} < V_{IH}$	–	No system reset
	$V_{DD} < V_{LVD}$	Transition from $V_{reset} < V_{IL}$ to $V_{IH} < V_{RESET}$	–	No system reset
Rising up from $V_{DD} < V_{POR}$	$V_{DD} \geq V_{LVD}$	$V_{RESET} \geq V_{IH}$	Internal POR	System reset occurs
	$V_{DD} > V_{LVD}$	$V_{RESET} < V_{IH}$	–	No system reset
	$V_{DD} < V_{LVD}$	Transition from $V_{RESET} < V_{IL}$ to $V_{IH} < V_{RESET}$	–	No system reset
Standstill ($V_{DD} \geq V_{LVD}$)	$V_{DD} \geq V_{LVD}$	Transition from $V_{RESET} < V_{IL}$ to $V_{IH} < V_{RESET}$	Reset pin	System reset occurs

9.10. Power-Down Modes

The power-down modes on the S3F80PB MCU consists of the following modes:

- Idle Mode
- Backup Mode
- Stop Mode

9.10.1. Idle Mode

Idle Mode is invoked by the instruction Idle (op code 6Fh). In Idle Mode, CPU operations are halted while some peripherals remain active. During Idle Mode, the internal clock signal is gated away from the CPU and from all but the following peripherals, which remain active:

- Interrupt logic
- Basic timer
- Timer 0
- Timer 1
- Timer 2
- Counter A

The I/O port pins retain the state (i.e., input or output) they were in at the time Idle Mode is entered.

9.10.2. Idle Mode Release

Idle Mode can be released in one of two ways:

1. Execute a reset. All system and peripheral control registers are reset to their default values and the contents of all data registers are retained. The reset automatically selects the slowest clock (1/16) because of the hardware reset value for the CLKCON Register. If all interrupts are masked in the IMR Register, a reset is the only way to release Idle Mode.
2. Activate any enabled internal or external interrupt. When using an interrupt to release Idle Mode, the 2-bit CLKCON.4/CLKCON.3 value remains unchanged, and the currently selected clock value is used. The interrupt is then serviced. When the return-from-interrupt condition (IRET) occurs, the instruction immediately following the one which initiated Idle Mode is executed.

► **Note:** Only external interrupts built in to the pin circuit can be used to release Stop Mode. To release Idle Mode, = use either an external interrupt or an internally-generated interrupt.

9.10.3. Backup Mode

For reducing current consumption, the S3F80PB MCU enters Backup Mode. If the external reset pin is in a low state or a falling level of V_{DD} is detected by LVD circuit on the point of V_{LVD} , the chip enters Backup Mode. The CPU and peripheral operations are stopped, but the LVD is enabled. Because of the oscillation stop, the supply current is reduced. The chip cannot be released from Backup Mode by any interrupt. The only way to release Backup Mode is the system-reset operation by interactive work of the reset pin and the LVD circuit. The watchdog timer system reset does not occur in Backup Mode.

Figure 68 shows the block diagram for Backup Mode.

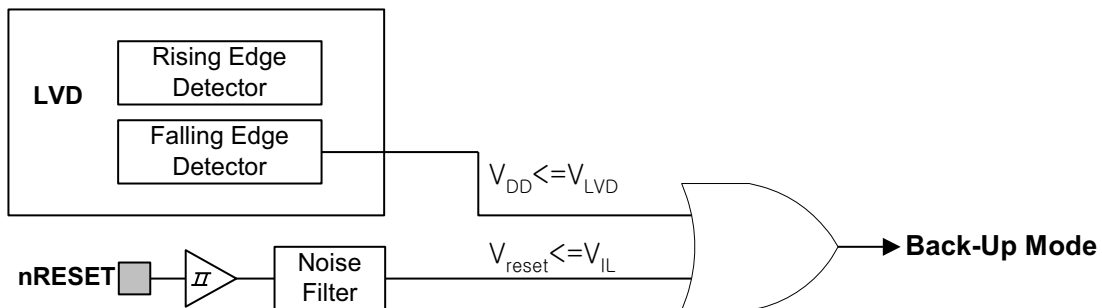


Figure 68. Block Diagram for Backup Mode

Figure 69 shows the Timing diagram for Backup Mode input and released by LVD.

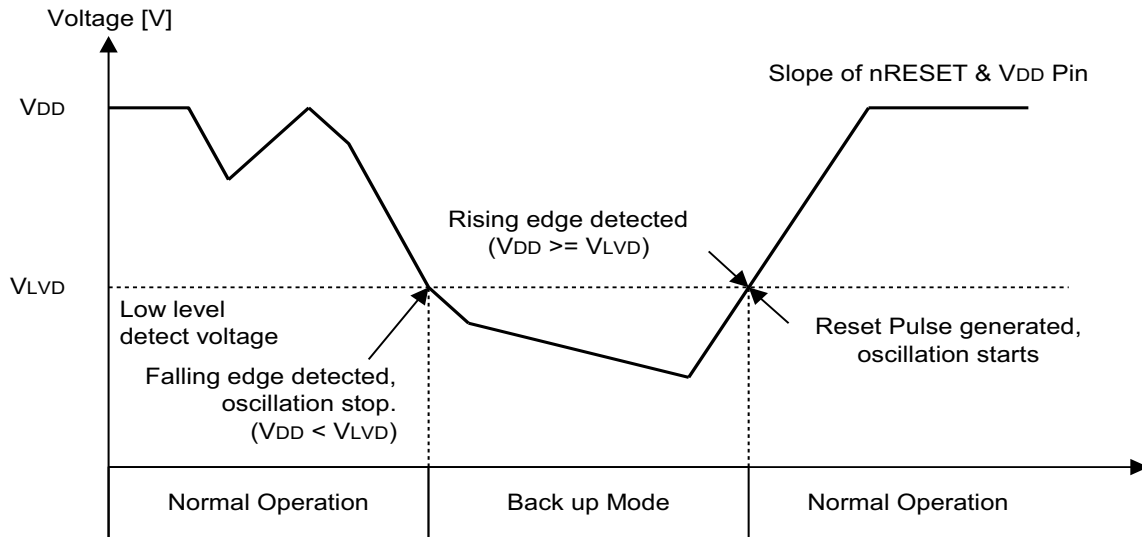


Figure 69. Timing Diagram for Backup Mode Input and Released by LVD

- **Notes:**
1. When the rising edge is detected by LVD circuit, Backup Mode is released (V_{LVD} is less than V_{DD}).
 2. When the falling edge is detected by LVD circuit, Backup Mode is activated (V_{LVD} is greater than V_{DD}).

Figure 70 shows the Timing diagram for Backup Mode input in Stop Mode.

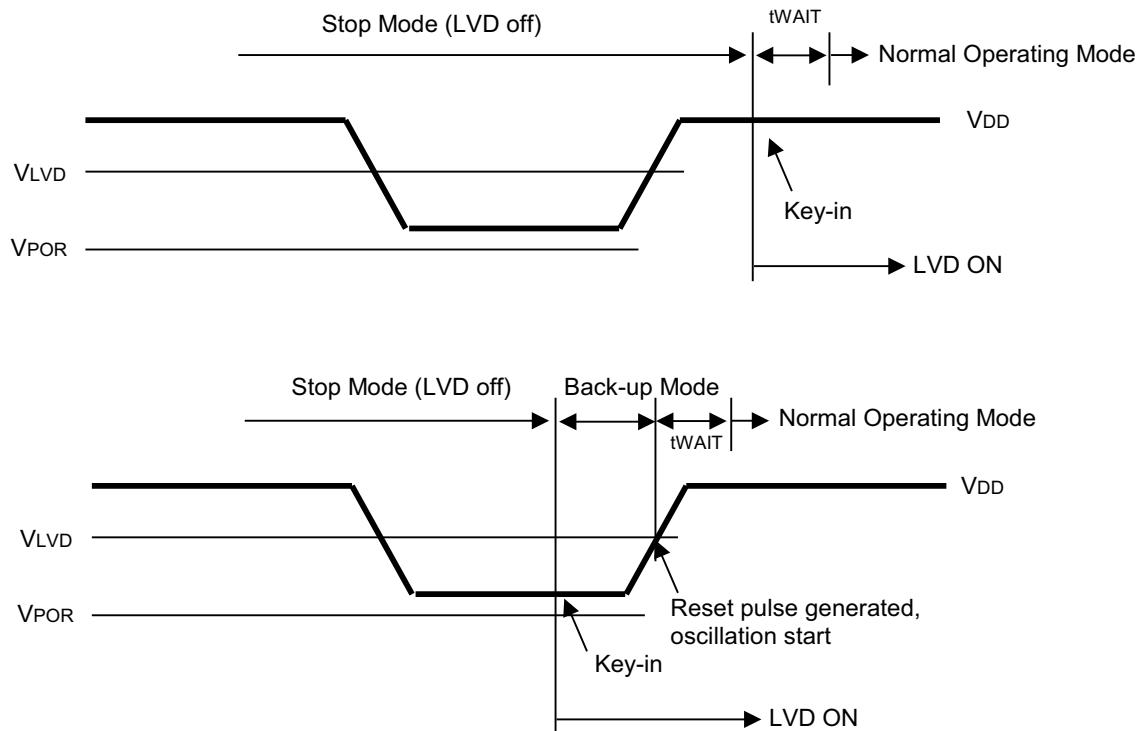


Figure 70. Timing Diagram for Backup Mode Input in Stop Mode

9.10.4. Stop Mode

Stop Mode is invoked by executing the Stop instruction after setting the Stop Control (STOPCON) Register. In Stop Mode, the operation of the CPU and all peripherals is halted. Essentially, the on-chip main oscillator stops and the current consumption can be reduced. All system functions stop when the clock freezes, but data stored in the internal register file is retained. Stop Mode can be released in one of two ways: by a system reset or by an external interrupt. After releasing from Stop Mode, the value of STOPCON is cleared automatically.

The following code segment shows an example of how to enter Stop Mode.

```

ORG      0000h          ; Reset address
.
.
.
JP      T, START
    
```

```
ENTER_STOP:
    LD        STOPCON, #0A5h
    STOP
    NOP
    NOP
    NOP
    RET

    ORG      0100h-3
    JP      T, START
    ORG 0100h                ; Reset address
START: LD    BTCON, #03      ; Clear basic timer
                                ; counter.
    .
    .
    .
MAIN: NOP
    .
    .
    .
    CALL    ENTER_STOP      ; Enter Stop Mode
    .
    .
    .
    LD      BTCON,#02h      ; Clear basic timer counter.
    JP      T, MAIN
    .
    .
    .
```

9.10.5. Sources to Release Stop Mode

Stop Mode is released when the following sources go active:

- System reset by external reset pin (nRESET)
- System reset by internal Power-On Reset (IPOR)
- External interrupt (INT0–INT9)
- SED & R circuit

9.10.6. Using nRESET Pin to Release Stop Mode

Stop Mode is released when the system reset signal goes active by nRESET Pin. All system and peripheral control registers are reset to their default hardware values and the contents of all data registers are retained. When the oscillation stabilization interval has

elapsed, the CPU starts the system initialization routine by fetching the program instruction stored in reset address.

9.10.7. Using IPOR to Release Stop Mode

Stop Mode is released when the system reset signal goes active by internal power-on reset (IPOR). All system and peripheral control registers are reset to their default hardware values and contents of all data registers are unknown states. When the oscillation stabilization interval has elapsed, the CPU starts the system initialization routine by fetching the program instruction stored in reset address.

9.10.8. Using an External Interrupt to Release Stop Mode

External interrupts can be used to release Stop Mode. When the RESET control bit is set to 0 (via the Smart Option at address 03Fh) and external interrupt is enabled, the S3F80PB MCU is released from Stop Mode and generates reset signal. Conversely, when the RESET control bit is 1 (via the Smart Option at address 03Fh), the S3F80PB MCU is only released from Stop Mode and does not generate reset signal. To wake-up from Stop Mode by external interrupt from INT0 to INT9, external interrupt should be enabled by setting corresponding control registers or instructions.

Note the following conditions regarding a Stop Mode release:

- If Stop Mode is released using an external interrupt, the current values in system and peripheral control registers are unchanged.
- If an external interrupt is being used for Stop Mode release, program the duration of the oscillation stabilization interval. To program the duration, you must make the appropriate control and clock settings before entering Stop Mode.
- If an interrupt is being used to release Stop Mode, the bit-pair setting for CLKCON.4/CLKCON.3 remains unchanged and the currently selected clock value is used.

9.10.9. Stop Error Detect and Recovery

The Stop Error Detect and Recovery (SED & R) circuit is used to release Stop Mode and prevent any abnormal Stop Mode that occurs due to a phenomenon known as *battery bouncing*. The circuit executes two functions in relation to the internal logic of P0 and P2.4–P2.7. One of these two functions is to release from stop status by switching the level of the input port (P0 or P2.4–P2.7); the other is to prevent the S3F80PB MCU from entering Stop Mode when the S3F80PB MCU is in an abnormal status, as noted below.

Release From Stop Mode. When the RESET control bit is set to 0 (via the Smart Option at address 03Fh), if falling edge input signal enters in through Port 0 or P2.4–P2.7, the S3F80PB MCU is released from Stop Mode and generates reset signal. Conversely, when

RESET control bit is 1 (via the Smart Option at address 03Fh), the S3F80PB MCU is only released Stop Mode, reset does not occur. When the falling edge of a pin on Port 0 and P2.4–P2.7 is entered, the S3F80PB MCU is released from Stop Mode even though external interrupt is disabled.

Preventing The Chip From Entering Abnormal Stop Mode. The circuit detects the abnormal status by checking the port (P0 and P2.4–P2.7) status. If the S3F80PB MCU is in abnormal status, it is prevented from entering Stop Mode.

The contents of the Stop Control (STOPCON) Register are described in Table 33.

Table 33. Stop Control Register (STOPCON; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W
Address	FBh							

Note: W = write only.

Bit	Description
[7:0]	Stop Control Register Enable Bits 10100101: Enable Stop Mode. If any other value, then disable Stop Mode.

Note:

1. To enter Stop Mode, the Stop Control (STOPCON) Register must be enabled just before a Stop instruction.
2. When the Stop Mode is released, the STOPCON value is cleared automatically.
3. It is prohibited to write another value into STOPCON.

► **Note:** In case the P2.0–2.3, SED & R circuit is not implemented. Therefore, although 4 pins (P2.0–2.3) feature the falling edge input signal in Stop Mode, if external interrupt is disabled, the stop state on the S3F80PB MCU is unchanged. Do not use Stop Mode if you are using an external clock source because X_{IN} input must be cleared internally to V_{SS} to reduce current leakage.

9.11. System Reset Operation

System reset starts the oscillation circuit, synchronize chip operation with CPU clock, and initialize the internal CPU and peripheral modules. This procedure brings the S3F80PB MCU into a known operating status. To allow time for internal CPU clock oscillation to stabilize, the reset pulse generator must be held to active level for a minimum time interval after the power supply comes within tolerance. The minimum required reset operation for

a oscillation stabilization time is 16 oscillation clocks. All system and peripheral control registers are subsequently reset to their default hardware values, which are listed in Table 34.

In summary, the following sequence of events occurs during a reset operation:

- All interrupts are disabled
- The watchdog function (Basic Timer) is enabled
- Port 0, 2 and 3 are set to Input Mode and all pull-up resistors are disabled for the I/O port pin circuits
- Peripheral control and data register settings are disabled and reset to their default hardware values
- The program counter (PC) is loaded with the program reset address in ROM, 0100h
- When the programmed oscillation stabilization time interval has elapsed, the instruction stored in reset address is fetched and executed

► **Note:** To program the duration of the oscillation stabilization interval, make the appropriate settings to the Basic Timer Control (BTCON) Register before entering Stop Mode. Additionally, if the basic timer watchdog function (which causes a system reset if a basic timer counter overflow occurs) is not required, disable it by writing 1010b to the upper nibble of BTCON. However, Zilog recommends that you use BTCON to prevent the possibility of chip malfunction.

9.11.1. Hardware Reset Values

Table 34 lists the Set 1, Bank0 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 the reset values in this table:

- A 1 or a 0 shows the reset bit value as logic 1 or logic 0, 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 (however, a 0 is read from the bit position)

Table 34. Set1, Bank0 Register Values After Reset

Register Name	Mnemonic	Address		Bit Values after Reset								
		Dec	Hex	7	6	5	4	3	2	1	0	
Timer 0 Counter Register	T0CNT	208	D0h	0	0	0	0	0	0	0	0	0
Timer 0 Data Register	T0DATA	209	D1h	1	1	1	1	1	1	1	1	1
Timer 0 Control Register	T0CON	210	D2h	0	0	0	0	0	0	0	0	0
Basic Timer Control Register	BTCN	211	D3h	0	0	0	0	0	0	0	0	0
Clock Control Register	CLKCON	212	D4h	0	0	0	0	0	0	0	0	0
System Flags Register	FLAGS	213	D5h	x	x	x	x	x	x	x	0	0
Register Pointer 0	RP0	214	D6h	1	1	0	0	0	–	–	–	–
Register Pointer 1	RP1	215	D7h	1	1	0	0	1	–	–	–	–
Location D8h (SPH) is not mapped.												
Stack Pointer Low Byte	SPL	217	D9h	x	x	x	x	x	x	x	x	x
Instruction Pointer High Byte	IPH	218	DAh	x	x	x	x	x	x	x	x	x
Instruction Pointer Low Byte	IPL	219	DBh	x	x	x	x	x	x	x	x	x
Interrupt Request Register (read only)	IRQ	220	DCh	0	0	0	0	0	0	0	0	0
Interrupt Mask Register	IMR	221	DDh	x	x	x	x	x	x	x	x	x
System Mode Register	SYM	222	DEh	0	–	–	x	x	x	0	0	0
Register Page Pointer	PP	223	DFh	0	0	0	0	0	0	0	0	0
Port 0 Data Register	P0	224	E0h	0	0	0	0	0	0	0	0	0
Port 1 Data Register	P1	225	E1h	0	0	0	0	0	0	0	0	0
Port 2 Data Register	P2	226	E2h	0	0	0	0	0	0	0	0	0
Port 3 Data Register	P3	227	E3h	0	–	0	0	1	1	0	0	0
Port 4 Data Register	P4	228	E4h	0	0	0	0	0	0	0	0	0
Port 2 Interrupt Enable Register	P2INT	229	E5h	0	0	0	0	0	0	0	0	0
Port 2 Interrupt Pending Register	P2PND	230	E6h	0	0	0	0	0	0	0	0	0
Port 0 Pull-Up Enable Register	P0PUR	231	E7h	0	0	0	0	0	0	0	0	0
Port 0 Control High Byte	P0CONH	232	E8h	0	0	0	0	0	0	0	0	0
Port 0 Control Low Byte	P0CONL	233	E9h	0	0	0	0	0	0	0	0	0
Port 1 Control High Byte	P1CONH	234	EAh	1	1	1	1	1	1	1	1	1
Port 1 Control Low Byte	P1CONL	235	EBh	0	0	0	0	0	0	0	0	0

Notes:

1. Although the SYM Register is not used, SYM.5 should always be 0. If you accidentally write a 1 to this bit during normal operation, a system malfunction can occur.
2. Except for T0CNTH, T0CNTL, IRQ, T1CNTH, T1CNTL, T2CNTH, T2CNTL, and BTCN, which are read only, all registers in Set1 are read/write-addressable.
3. Read-only registers cannot be used as destination fields for the OR, AND, LD, and LDB instructions.

Table 34. Set1, Bank0 Register Values After Reset (Continued)

Register Name	Mnemonic	Address		Bit Values after Reset								
		Dec	Hex	7	6	5	4	3	2	1	0	
Port 2 Control High Byte	P2CONH	236	ECh	0	0	0	0	0	0	0	0	0
Port 2 Control Low Byte	P2CONL	237	EDh	0	0	0	0	0	0	0	0	0
Port 2 Pull-Up Enable	P2PUR	238	EEh	0	0	0	0	0	0	0	0	0
Port 3 Control	P3CON	239	EFh	0	0	0	0	0	0	0	0	0
Port 4 Control	P4CON	240	F0h	0	0	0	0	0	0	0	0	0
Port 0 Interrupt Enable	P0INT	241	F1h	0	0	0	0	0	0	0	0	0
Port 0 Interrupt Pending	P0PND	242	F2h	0	0	0	0	0	0	0	0	0
Counter A Control	CACON	243	F3h	0	0	0	0	0	0	0	0	0
Counter A Data High Byte	CADATAH	244	F4h	1	1	1	1	1	1	1	1	1
Counter A Data Low Byte	CADATAL	245	F5h	1	1	1	1	1	1	1	1	1
Timer 1 Counter High Byte	T1CNTH	246	F6h	0	0	0	0	0	0	0	0	0
Timer 1 Counter Low Byte	T1CNTL	247	F7h	0	0	0	0	0	0	0	0	0
Timer 1 Data High Byte	T1DATAH	248	F8h	1	1	1	1	1	1	1	1	1
Timer 1 Data Low Byte	T1DATAL	249	F9h	1	1	1	1	1	1	1	1	1
Timer 1 Control	T1CON	250	FAh	0	0	0	0	0	0	0	0	0
Stop Control	STOPCON	251	FBh	0	0	0	0	0	0	0	0	0
Location FCh is not mapped; it is used for factory testing.												
Basic Timer Counter	BTCNT	253	FDh	0	0	0	0	0	0	0	0	0
External Memory Timing	EMT	254	FEh	0	1	1	1	1	1	1	0	–
Interrupt Priority	IPR	255	FFh	x	x	x	x	x	x	x	x	x

Notes:

1. Although the SYM Register is not used, SYM.5 should always be 0. If you accidentally write a 1 to this bit during normal operation, a system malfunction can occur.
2. Except for T0CNTH, T0CNTL, IRQ, T1CNTH, T1CNTL, T2CNTH, T2CNTL, and BTCNT, which are read only, all registers in Set1 are read/write-addressable.
3. Read-only registers cannot be used as destination fields for the OR, AND, LD, and LDB instructions.

Table 35 lists the Set1, Bank1 reset values for CPU and system registers, peripheral control registers, and peripheral data registers following a reset operation.

Table 35. Set1, Bank1 Register Values After Reset

Register Name	Mnemonic	Address		Bit Values after Reset								
		Dec	Hex	7	6	5	4	3	2	1	0	
LVD Control	LVDCON	224	E0h	–	–	–	–	–	–	–	–	0
Port 3[4:5] Control	P345CON	225	E1h	0	1	0	1	0	0	0	0	0
Port 4 Control High Byte	P4CONH	226	E2h	1	1	1	1	1	1	1	1	1
Port 4 Control Low Byte	P4CONL	227	E3h	1	1	1	1	1	1	1	1	1
Timer 2 Counter High Byte	T2CNTH	228	E4h	0	0	0	0	0	0	0	0	0
Timer 2 Counter Low Byte	T2CNTL	229	E5h	0	0	0	0	0	0	0	0	0
Timer 2 Data High Byte	T2DATAH	230	E6h	1	1	1	1	1	1	1	1	1
Timer 2 Data Low Byte	T2DATAL	231	E7h	1	1	1	1	1	1	1	1	1
Timer 2 Control	T2CON	232	E8h	0	0	0	0	0	0	0	0	0
Locations E9h to EBh are not mapped.												
Flash Memory Sector Address High Byte	FMSECH	236	ECh	0	0	0	0	0	0	0	0	0
Flash Memory Sector Address Low Byte	FMSECL	237	EDh	0	0	0	0	0	0	0	0	0
Flash Memory User Programming Enable	FMUSR	238	EEh	0	0	0	0	0	0	0	0	0
Flash Memory Control	FMCON	239	EFh	0	0	0	0	–	–	–	–	0
Reset Indicating	RESETID	240	F0h	See the Control Registers chapter on page 57								
LVD Flag Level Selection	LVDSSEL	243	F1h	0	0	–	–	–	–	–	–	–
Port 1 Output Mode Pull-Up Enable	P1OUTPU	244	F2h	0	0	0	0	0	0	0	0	0
Port 2 Output Mode Selection	P2OUTMD	245	F3h	0	0	0	0	0	0	0	0	0
Port 3 Output Mode Pull-Up Enable	P3OUTPU	246	F4h	–	–	0	0	–	–	0	0	0
Port 4 Output Mode Pull-Up Enable	P4OUTPU	247	F5h	0	0	0	0	0	0	0	0	0

Notes:

1. P345CON will be initialized as 50h to set P3.4 and P3.5 into open drain output mode after reset operation.
2. S3F80PB has P4CONH, P4CONL and P4CON as port 4 control registers. P4CONH and P4CONL will be initialized as the C-MOS input with pull up mode after reset. On the other hand, P4CON will be initialized as open-drain output mode. After reset, status of Port 4 is decided by P345CON.0 bit. So port4 reset status will be initialized as open-drain output mode.

Table 36 describes the reset generation according to the condition of the Smart Option.

Table 36. Smart Option Reset Generation

Mode	Reset Source	Smart Option 1 st Bit @ 3Fh				
		1		0		
Normal Operating	Reset Pin	O	Reset	O	Reset	
	Watchdog Timer Enable	O	Reset	O	Reset	
	IPOR	O	Reset	O	Reset	
	LVD	O	Reset	O	Reset	
	External Interrupt (EI) P0 and P2	X	External ISR	X	External ISR	
	External Interrupt (DI) P0 and P2	X	Continue	X	Continue	
Stop Mode	Reset Pin	O	Reset	O	Reset	
	Watchdog Timer Enable	X	Stop	X	Stop	
	IPOR	O	Stop Release and Reset	O	Stop Release and Reset	
	LVD	X	Stop	X	Stop	
	External Interrupt (EI Enable) P0 and P2	X	Stop Release and External ISR	O	Stop Release and Reset	
	SED & R	P0 & P2.4–P2.7	X	Stop Release and Continue	O	Stop Release and Reset
		P2.0–P2.3	X	Stop	X	Stop

Notes:

1. X means that a corresponding reset source do not generate reset signal. O means that a corresponding reset source generates reset signal.
2. Reset means that reset signal is generated and chip reset occurs.
3. Continue means that it executes the next instruction continuously without ISR execution.
4. External ISR means that chip executes the interrupt service routine of generated external interrupt source.
5. Stop means that the S3F80PB MCU is in a stop state.
6. Stop Release and External ISR means that chip executes the external interrupt service routine of generated external interrupt source after Stop released.
7. Stop Release and Continue means that executes the next instruction continuously after Stop released.

9.12. Recommendation for Unused Pins

To reduce overall power consumption, configure unused pins according to the guidelines listed in Table 37.

Table 37. Guideline for Unused Pins to Reduced Power Consumption

Pin Name	Recommend	Example
Port 0	<ul style="list-style-type: none"> Set Input Mode. Enable pull-up resistor. No Connection for pins. 	<ul style="list-style-type: none"> P0CONH ← # 00h or 0FFh P0CONL ← # 00h or 0FFh P0PUR ← # 0FFh
Port 1	<ul style="list-style-type: none"> Set Open-Drain Output Mode. Set P1 Data Register to # 00h. Disable pull-up resistor. No Connection for pins. 	<ul style="list-style-type: none"> P1CONH ← # 55h P1CONL ← # 55h P1 ← # 00h
Port 2	<ul style="list-style-type: none"> Set Push-Pull Output Mode. Set P2 Data Register to # 00h. Disable pull-up resistor. No Connection for pins. 	<ul style="list-style-type: none"> P2CONH ← # 0AAh P2CONL ← # 0AAh P2 ← # 00h P2PUR ← # 00h
P3.0–3.1	<ul style="list-style-type: none"> Set Push-Pull Output Mode. Set P3 Data Register to # 00h. No Connection for pins. 	<ul style="list-style-type: none"> P3CON ← # 11010010b P3 ← # 00h
P3.2–3.3	<ul style="list-style-type: none"> – 	<ul style="list-style-type: none"> No connection
P3.4–3.5	<ul style="list-style-type: none"> Set Push-Pull output mode Set P3.4 and P3.5 data registers to #00h No connection for pins 	<ul style="list-style-type: none"> P345CON← # A0h P3 ← # 00h
Port 4	<ul style="list-style-type: none"> Set Push-Pull output mode Set P4 Data Register to #00h No connection for pins 	<ul style="list-style-type: none"> P4CONH← # 0AAh P4CONL← # 0AAh P4← # 00h
Test	<ul style="list-style-type: none"> Connect to V_{SS}. 	–

Table 38 summarizes the status of the backup, reset, and stop modes.

Table 38. Summary of Each Mode

Item/Mode	Backup	Reset Status	Stop
Approach Condition	<ul style="list-style-type: none"> External nRESET pin is low level state or V_{DD} is lower than V_{LVD}. 	<ul style="list-style-type: none"> External nRESET pin is on rising edge. The rising edge at V_{DD} is detected by LVD circuit when $V_{DD} \geq V_{LVD}$. Watchdog timer overflow signal is activated. 	<ul style="list-style-type: none"> STOPCON ← # A5h Stop (LD STOPCON, # 0A5h) (Stop)
Port Status	<ul style="list-style-type: none"> All I/O ports are at floating status. All of the ports enter Input Mode but are blocked. Disable all pull-up resistors except P3.2 and P3.3. 	<ul style="list-style-type: none"> All I/O ports are at floating status. Disable all pull-up resistors except P3.2 and P3.3. 	<ul style="list-style-type: none"> All of the ports keep the previous status. Output port data is not changed.
Control Register	<ul style="list-style-type: none"> All control registers and system registers are initialized as a list; see Table 34 on page 210. 	<ul style="list-style-type: none"> All control registers and system registers are initialized as a list; see Table 34 on page 210. 	–
Releasing Condition	<ul style="list-style-type: none"> External nRESET pin is high (rising edge). The rising edge of LVD circuit is generated. 	<ul style="list-style-type: none"> After passing an oscillation warm-up time. 	<ul style="list-style-type: none"> External interrupt, or reset SED & R Circuit.
Others	<ul style="list-style-type: none"> There is no current consumption in the chip. 	<ul style="list-style-type: none"> There can be input leakage current in the chip. 	<ul style="list-style-type: none"> Depends on the control program.

9.13. Reset Source Indicating Register

The contents of the Reset Source Indicating (RESETID) Register are described in Table 39. The state of the RESETID depends on the source of the reset; see Table 40.

Table 39. Reset Source Indicating Register (RESETID; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
R/W	–	–	–	R/W	R/W	R/W	R/W	R/W
Address	F0h							

Note: R/W = read/write.

Bit	Description
[7:5]	Reserved These bits are reserved and must be set to 000.
[4]	nRESET Pin Indicating Bit 0: Reset is not generated by nRESET pin when read. 1: Reset is generated by nRESET pin when read.
[3]	Key-In Reset Indicating Bit 0: Reset is not generated by P0, P2 external INT or SED & R when read. 1: Reset is generated by P0, P2 external INT or SED & R when read.
[2]	WDT Reset Indicating Bit 0: Reset is not generated by WDT when read. 1: Reset is generated by WDT when read.
[1]	LVD Reset Indicating Bit 0: Reset is not generated by LVD when read. 1: Reset is generated by LVD when read.
[0]	POR Reset Indicating Bit 0: Reset is not generated by POR when read. 1: Reset is generated by POR when read.

Table 40. State of RESETID Depends on Reset Source

Bit	7	6	5	4	3	2	1	0
POR	–	–	–	0	0	0	See Note 1	See Note 1
LVD	–	–	–	0	0	0	See Note 1	See Note 2
WDT, Key-in, or nReset Pin	–	–	–		3		See Note 2	See Note 2

Note:

1. To clear an indicating register, write a 0 to indicating flag bit. Writing a 1 to an reset indicating flag (RESETID.0–4) has no effect.
2. Not affected by any other reset.
3. Bits corresponding to sources that are active at the time of reset will be set.

Chapter 10. I/O Ports

The S3F80PB microcontroller features two packages.

The 44-pin QFP package type has a total of 38 I/O pins. Among these are five bit-programmable I/O ports, P0–P4. Four ports, P0–P2 and P4, are 8-bit ports; P3 is a 6-bit port.

The 32-pin SOP package type has a total of 26 I/O pins. Three ports, P0–P2, are 8-bit ports; P3 is a 2-bit port.

Each port is bit-programmable and can be flexibly configured to meet application design requirements. The CPU accesses ports by directly writing or reading port registers. No special I/O instructions are required.

For IR applications, Port 0, Port 1, and Port 2 are usually configured to the keyboard matrix and Port 3 is used to IR drive pins.

Tables 41 and 42 provide a general overview of the S3F80PB MCU's I/O port functions for the 44-pin QFP/QFN and the 32-pin SOP/QFN packages.

Table 41. S3F80PB Port Configuration Overview (44-Pin QFP, QFN)

Port	Configuration Options
Port 0	8-bit general-purpose I/O port; input or push-pull output; external interrupt input on falling edges, rising edges, or both edges; all P0 pin circuits feature noise filters and Interrupt Enable/Disable (POINT) Register. The Pending Control (POPND) Register and pull-up resistors can be assigned to individual P0 pins using the P0PUR register settings. This port is dedicated for key input in IR controller applications.
Port 1	8-bit general-purpose I/O port; input with or without pull-ups, open-drain output, or push-pull output. This port is dedicated for key output in IR controller applications.
Port 2	8-bit general-purpose I/O port; input, open-drain output, or push-pull output. The P2 pins, P2.0–P2.7, can be used as external interrupt inputs and contain noise filters. The P2INT register is used to enable/disable interrupts and P2PND bits can be polled by software for interrupt pending control. Pull-up resistors can be assigned to individual P2 pins using P2PUR register settings.
P3.0–P3.1	P3.0 is configured for input functions (Input mode, with or without pull-up, for normal input or T0CAP) or output functions (push-pull or open-drain output mode, for normal output or T0PWM). P3.1 is configured for input functions (Input mode, with or without pull-up, for normal input) or output functions (push-pull or open-drain output mode, for normal output or REM function). P3.1 is dedicated for the IR drive pin and P3.0 can be used for the indicator LED drive.
P3.2–P3.3	P3.2 is configured as an input only pin with pull-up resistor (for normal input or T0CK function). P3.3 is configured as an input only pin with pull-up resistor (for normal input, T1CAP function, or T2CAP function). P3.3 can be used for the IR signal capture pin with T1CAP function or T2CAP function.

Table 41. S3F80PB Port Configuration Overview (44-Pin QFP, QFN) (Continued)

Port	Configuration Options
P3.4–P3.5	2-bit general-purpose I/O port; input without or with pull-up, open-drain output, or push-pull output.
P3.7	P3.7 is not configured for the I/O pin and it is only used to control carrier signal on/off.
Port 4	8-bit general-purpose I/O port; input without or with pull-up, open-drain output, or push-pull output. This port is dedicated for key outputs in the IR controller application.

Table 42. S3F80PB Port Configuration Overview (32-Pin SOP, QFN)

Port	Configuration Options
Port 0	8-bit general-purpose I/O port; input or push-pull output; external interrupt input on falling edges, rising edges, or both edges; all P0 pin circuits feature noise filters and Interrupt Enable/Disable (POINT) Register and the Pending Control (P0PND) Register; pull-up resistors can be assigned to individual P0 pins using the P0PUR register settings. This port is dedicated for key input in IR controller applications.
Port 1	8-bit general-purpose I/O port; input with or without pull-ups, open-drain output, or push-pull output. This port is dedicated for key output in IR controller applications.
Port 2	8-bit general-purpose I/O port; input, open-drain output, or push-pull output. The P2 pins, P2.0–P2.7, can be used as external interrupt inputs and contain noise filters. The P2INT register is used to enable/disable interrupts and P2PND bits can be polled by software for interrupt pending control. Pull-up resistors can be assigned to individual P2 pins using P2PUR register settings.
P3.0–P3.1	2-bit I/O port; P3.0 and P3.1 are configured input functions (Input Mode, with or without pull-up, for T0CK, T0CAP, or T1CAP) or output functions (Push-Pull or Open-Drain Output Mode, or for REM and T0PWM). P3.1 is dedicated for the IR drive pin and P3.0 can be used for indicator LED drive.
P3.7	P3.7 is not configured for the I/O pin and it is only used to control carrier signal on/off.

10.1. Port Data Registers

Table 43 provides an overview of the register locations of all four S3F80PB I/O port data registers. Data registers for ports 0, 1, 2, 3, and 4 feature the general format shown in Figure 71.

Table 43. Port Data Register Summary

Register Name	Mnemonic	Decimal	Hex	Location	Read/Write
Port 0 Data Register	P0	224	E0h	Set1, Bank0	R/W
Port 1 Data Register	P1	225	E1h	Set1, Bank0	R/W
Port 2 Data Register	P2	226	E2h	Set1, Bank0	R/W
Port 3 Data Register	P3	227	E3h	Set1, Bank0	R/W
Port 4 Data Register	P4	228	E4h	Set1, Bank0	R/W

► **Note:** The data register for Port 3 (P3) contains 6 bits for P3.0–P3.5 and an additional status bit (P3.7) to toggle the carrier signal on or off.

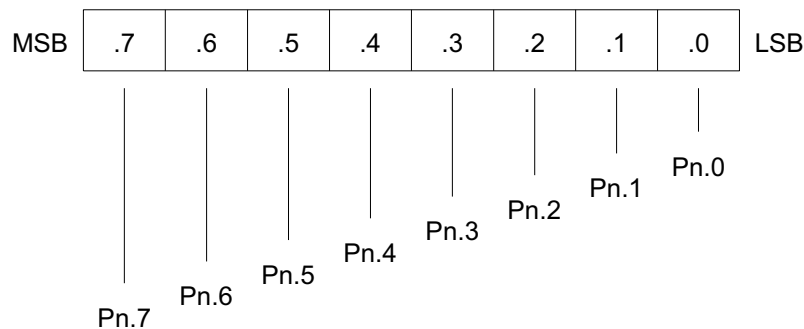


Figure 71. S3F80PB I/O Port Data Register Format

► **Note:** Because Port 3 is a 6-bit I/O port, the Port 3 Data Register only contains values for P3.0–P3.5. The P3 Register also contains a special carrier on/off bit, P3.7; see the [Port 3](#) section on page 236 for details. All other S3F80PB I/O ports are 8-bit ports.

10.2. Port 0

Port 0 is an 8-bit I/O port with individually-configurable pins that are accessed directly by writing or reading the Port 0 Data Register, P0, at location F0h in Set1, Bank1. P0.0–P0.7 can serve as inputs (with or without pull-ups) and push-pull outputs.

10.2.0.1. Port 0 Control Registers

Port 0 provides two 8-bit control registers, P0CONH for P0.4–P0.7 and P0CONL for P0.0–P0.3 (see Tables 44 and 45). A reset clears these P0CONH and P0CONL registers to 00h, configuring all pins to Input Mode. When Port 0 is in Input Mode, the following three 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

Table 44. Port 0 Control (P0CONH; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					E8h			

Note: R/W = read/write.

Bit	Description
[7:6]	<p>P0.7/INT4 Mode Selection Bits</p> <p>00: CMOS Input Mode; interrupt on falling edges. 01: CMOS Input Mode; interrupt on rising and falling edges. 10: Push-Pull Output Mode. 11: CMOS Input Mode; interrupt on rising edges.</p>
[5:4]	<p>P0.6/INT4 Mode Selection Bits</p> <p>00: CMOS Input Mode; interrupt on falling edges. 01: CMOS Input Mode; interrupt on rising and falling edges. 10: Push-Pull Output Mode. 11: CMOS Input Mode; interrupt on rising edges.</p>

Notes:

1. The INT4 external interrupts at the P0.7–P0.4 pins share the same interrupt level (IRQ7) and interrupt vector address (E8h).
2. Assign pull-up resistors to individual Port 0 pins by making the appropriate settings to the P0PUR Register. (P0PUR.7–P0PUR.4).

Bit	Description (Continued)
[3:2]	P0.5/INT4 Mode Selection Bits 00: CMOS Input Mode; interrupt on falling edges. 01: CMOS Input Mode; interrupt on rising and falling edges. 10: Push-Pull Output Mode. 11: CMOS Input Mode; interrupt on rising edges.
[1:0]	P0.4/INT4 Mode Selection Bits 00: CMOS Input Mode; interrupt on falling edges. 01: CMOS Input Mode; interrupt on rising and falling edges. 10: Push-Pull Output Mode. 11: CMOS Input Mode; interrupt on rising edges.

Notes:

1. The INT4 external interrupts at the P0.7–P0.4 pins share the same interrupt level (IRQ7) and interrupt vector address (E8h).
2. Assign pull-up resistors to individual Port 0 pins by making the appropriate settings to the P0PUR Register. (P0PUR.7–P0PUR.4).

Table 45. Port 0 Control Low Byte Register (P0CONL; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					E9h			

Note: R/W = read/write.

Bit	Description
[7:6]	P0.3/INT3 Mode Selection Bits 00: CMOS Input Mode; interrupt on falling edges. 01: CMOS Input Mode; interrupt on rising and falling edges. 10: Push-Pull Output Mode. 11: CMOS Input Mode; interrupt on rising edges.
[5:4]	P0.2/INT2 Mode Selection Bits 00: CMOS Input Mode; interrupt on falling edges. 01: CMOS Input Mode; interrupt on rising and falling edges. 10: Push-Pull Output Mode. 11: CMOS Input Mode; interrupt on rising edges.

Note: The INT3–INT0 external interrupts at P0.3–P0.0 are interrupt level IRQ6. Each interrupt contains a separate vector address. Assign pull-up resistors to individual Port 0 pins by making the appropriate settings to the P0PUR Register. (P0PUR.3–P0PUR.0).

Bit	Description (Continued)
[3:2]	P0.1/INT1 Mode Selection Bits 00: CMOS Input Mode; interrupt on falling edges. 01: CMOS Input Mode; interrupt on rising and falling edges. 10: Push-Pull Output Mode. 11: CMOS Input Mode; interrupt on rising edges.
[1:0]	P0.0/INT0 Mode Selection Bits 00: CMOS Input Mode; interrupt on falling edges. 01: CMOS Input Mode; interrupt on rising and falling edges. 10: Push-Pull Output Mode. 11: CMOS Input Mode; interrupt on rising edges.

Note: The INT3–INT0 external interrupts at P0.3–P0.0 are interrupt level IRQ6. Each interrupt contains a separate vector address. Assign pull-up resistors to individual Port 0 pins by making the appropriate settings to the P0PUR Register. (P0PUR.3–P0PUR.0).

The contents of the Port 0 External Interrupt Enable (P0INT) Register are described in Table 46.

Table 46. Port 0 External Interrupt Enable Register (P0INT; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					F1h			

Note: R/W = read/write.

Bit	Description
[7]	P0.7 External Interrupt (INT4) Enable Bit 0: Disable interrupt. 1: Enable interrupt.
[6]	P0.6 External Interrupt (INT4) Enable Bit 0: Disable interrupt. 1: Enable interrupt.
[5]	P0.5 External Interrupt (INT4) Enable Bit 0: Disable interrupt. 1: Enable interrupt.
[4]	P0.4 External Interrupt (INT4) Enable Bit 0: Disable interrupt. 1: Enable interrupt.
[3]	P0.3 External Interrupt (INT3) Enable Bit 0: Disable interrupt. 1: Enable interrupt.

Bit	Description (Continued)
[2]	P0.2 External Interrupt (INT2) Enable Bit 0: Disable interrupt. 1: Enable interrupt.
[1]	P0.1 External Interrupt (INT1) Enable Bit 0: Disable interrupt. 1: Enable interrupt.
[0]	P0.0 External Interrupt (INT0) Enable Bit 0: Disable interrupt. 1: Enable interrupt.

The contents of the Port 0 External Interrupt Pending (P0PND) Register are described in Table 47.

Table 47. Port 0 External Interrupt Pending Register (P0PND; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					F2h			

Note: R/W = read/write.

Bit	Description
[7]	P0.7 External Interrupt (INT4) Pending Flag Bit* 0: No P0.7 external interrupt pending when read. 1: P0.7 external interrupt is pending when read.
[6]	P0.6 External Interrupt (INT4) Pending Flag Bit 0: No P0.6 external interrupt pending when read. 1: P0.6 external interrupt is pending when read.
[5]	P0.5 External Interrupt (INT4) Pending Flag Bit 0: No P0.5 external interrupt pending when read. 1: P0.5 external interrupt is pending when read.
[4]	P0.4 External Interrupt (INT4) Pending Flag Bit 0: No P0.4 external interrupt pending when read. 1: P0.4 external interrupt is pending when read.
[3]	P0.3 External Interrupt (INT3) Pending Flag Bit 0: No P0.3 external interrupt pending when read. 1: P0.3 external interrupt is pending when read.
[2]	P0.2 External Interrupt (INT2) Pending Flag Bit 0: No P0.2 external interrupt pending when read. 1: P0.2 external interrupt is pending when read.

Bit	Description
[1]	P0.1 External Interrupt (INT1) Pending Flag Bit 0: No P0.1 external interrupt pending when read. 1: P0.1 external interrupt is pending when read.
[0]	P0.0 External Interrupt (INT0) Pending Flag Bit 0: No P0.0 external interrupt pending when read. 1: P0.0 external interrupt is pending when read.

Note: *To clear an interrupt pending condition, write a 0 to the appropriate pending flag bit. Writing a 1 to an interrupt pending flag (P0PND.7–0) has no effect.

These control register settings can be used to select Input Mode (with or without pull-ups) or to select Push-Pull Output Mode and enable the alternative functions.

When programming the port, note that any alternative peripheral I/O function you configure using the Port 0 Control registers must also be enabled in the associated peripheral module.

Pull-up resistors can be assigned to the pin circuits of individual pins in Port 0. To assign these pull-up resistors, enter the appropriate settings to the Port 0 Pull-Up Resistor Enable (P0PUR) Register. This register is located in Set 1, Bank 0 at location E7h and is read/write accessible using Register Addressing Mode.

A pull-up resistor can be assigned to the Port 1 and Port 4 pins using the basic port configuration settings in the P1CONH, P1CONL, P4CONH, and P4CONL registers. Pull-up resistors can also be assigned to the Port 3 pins P3.0, P3.1, P3.4, and P3.5 in Input Mode using the basic port configuration settings in the P3CON and P345CON registers. P3.2–P3.3 are configured as input only pins with a pull-up resistor.

Table 48 lists the contents of the Port 0 Pull-Up Resistor Enable Register.

Table 48. Port 0 Pull-Up Resistor Enable Register (P0PUR; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					E7h			

Note: R/W = read/write.

Bit	Description
[7]	P0.7 Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.

Bit	Description (Continued)
[6]	P0.6 Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[5]	P0.5 Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[4]	P0.4 Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[3]	P0.3 Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[2]	P0.2 Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[1]	P0.1 Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[0]	P0.0 Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.

- **Notes:**
1. Pull-up resistors can be assigned to the Port 3 pins, P3.0 and P3.1, by modifying the appropriate settings in the Port 3 Control (P3CON) Register.
 2. Pull-up resistors can be assigned to the Port 3 pins, P3.4 and P3.5, by modifying the appropriate settings in the Port 3[4:5] and Port 3[6:7] Control (P345CON) Register.
 3. Pull-up resistors can be assigned to the P1 and P4 pins by modifying the appropriate settings in the Port 1 Control registers P1CONL and P1CONH, and in the Port 4 control registers P4CONL and P4CONH, respectively.

10.2.1. Port 1

Port 1 is an 8-bit I/O port with individually-configurable pins. Port 1 pins are accessed directly by writing or reading the Port 1 Data (P1) Register at location F1h in Set1, Bank1. P1.0–P1.7 can serve as inputs (with or without pull-ups) and outputs (push-pull or open-drain).

10.2.1.1. Port 1 Control Registers

Port 1 contains two 8-bit control registers: P1CONH for P1.4–P1.7 and P1CONL for P1.0–P1.3 (see Tables 49 and 50). A reset clears these P1CONH and P1CONL registers to 00h, configuring all pins to Input Mode. The control registers settings can be used to select Input Mode (with or without pull-ups) or Output Mode.

When programming the port, note that any alternative peripheral I/O function you configure using the Port 1 Control registers must also be enabled in the associated peripheral module.

Table 49. Port 1 Control High Byte Register (P1CONH; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	1	1	1	1	1	1	1	1
R/W					R/W			
Address					EAh			

Note: R/W = read/write.

Bit	Description
[7:6]	P1.7 Mode Selection Bits 00: CMOS Input Mode. 01: Open-Drain Output Mode. 10: Push-Pull Output Mode. 11: CMOS input with Pull-Up Mode.
[5:4]	P1.6 Mode Selection Bits 00: CMOS Input Mode. 01: Open-Drain Output Mode. 10: Push-Pull Output Mode. 11: CMOS input with Pull-Up Mode.
[3:2]	P1.5 Mode Selection Bits 00: CMOS Input Mode. 01: Open-Drain Output Mode. 10: Push-Pull Output Mode. 11: CMOS input with Pull-Up Mode.
[1:0]	P1.4 Mode Selection Bits 00: CMOS Input Mode. 01: Open-Drain Output Mode. 10: Push-Pull Output Mode. 11: CMOS input with Pull-Up Mode.

The contents of the Port 1 Control Low Byte (P1CONL) Register are described in Table 50.

Table 50. Port 1 Control Low Byte Register (P1CONL; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					EBh			

Note: R/W = read/write.

Bit	Description
[7:6]	P1.3 Mode Selection Bits 00: CMOS Input Mode. 01: Open-Drain Output Mode. 10: Push-Pull Output Mode. 11: CMOS input with Pull-Up Mode.
[5:4]	P1.2 Mode Selection Bits 00: CMOS Input Mode. 01: Open-Drain Output Mode. 10: Push-Pull Output Mode. 11: CMOS input with Pull-Up Mode.
[3:2]	P1.1 Mode Selection Bits 00: CMOS Input Mode. 01: Open-Drain Output Mode. 10: Push-Pull Output Mode. 11: CMOS input with Pull-Up Mode.
[1:0]	P1.0 Mode Selection Bits 00: CMOS Input Mode. 01: Open-Drain Output Mode. 10: Push-Pull Output Mode. 11: CMOS input with Pull-Up Mode.

The contents of the Port 1 Output Pull-Up Resistor Enable (P1OUTPU) Register are described in Table 51.

Table 51. Port 1 Output Pull-Up Resistor Enable Register (P1OUTPU; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					F2h			

Note: R/W = read/write.

Bit	Description
[7]	P1.7 Output Mode Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[6]	P1.6 Output Mode Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[5]	P1.5 Output Mode Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[4]	P1.4 Output Mode Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[3]	P1.3 Output Mode Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[2]	P1.2 Output Mode Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[1]	P1.1 Output Mode Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[0]	P1.0 Output Mode Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.

10.2.2. Port 2

Port 2 is an 8-bit I/O port with individually-configurable pins which are accessed directly by writing or reading the Port 2 Data (P2) Register at location F2h in Set1, Bank1. P2.0–P2.7 can serve as inputs (with or without pull-ups) and push-pull outputs.

10.2.2.1. Port 2 Control Registers

Port 2 provides two 8-bit control registers, P2CONH for P2.4–P2.7 and P2CONL for P2.0–P2.3 (see Tables 52 and 53). A reset clears these P2CONH and P2CONL registers to 00h, configuring all pins to Input Mode. Use control register settings to select Input Mode (with or without pull-ups) or to select Push-Pull Output Mode and enable the alternative functions.

When programming the port, note that any alternative peripheral I/O function you configure using the Port 2 Control registers must also be enabled in the associated peripheral module.

Table 52. Port 2 Control High Byte Register (P2CONH; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					ECh			

Note: R/W = read/write.

Bit	Description
[7:6]	<p>P2.7/INT9 Mode Selection Bits</p> <p>00: CMOS Input Mode; interrupt on falling edges. 01: CMOS Input Mode; interrupt on rising and falling edges. 10: Output Mode; push-pull or open-drain output; see the Port 2 Output Mode Selection Register on page 232. 11: CMOS Input Mode; interrupt on rising edges.</p>
[5:4]	<p>P2.6/INT9 Mode Selection Bits</p> <p>00: CMOS Input Mode; interrupt on falling edges. 01: CMOS Input Mode; interrupt on rising and falling edges. 10: Output Mode; push-pull or open-drain output; see the Port 2 Output Mode Selection Register on page 232. 11: CMOS Input Mode; interrupt on rising edges.</p>
[3:2]	<p>P2.5/INT9 Mode Selection Bits</p> <p>00: CMOS Input Mode; interrupt on falling edges. 01: CMOS Input Mode; interrupt on rising and falling edges. 10: Output Mode; push-pull or open-drain output; see the Port 2 Output Mode Selection Register on page 232. 11: CMOS Input Mode; interrupt on rising edges.</p>

Bit	Description (Continued)
[1:0]	<p>P2.4/INT9 Mode Selection Bits</p> <p>00: CMOS Input Mode; interrupt on falling edges. 01: CMOS Input Mode; interrupt on rising and falling edges. 10: Output Mode; push-pull or open-drain output; see the Port 2 Output Mode Selection Register on page 232. 11: CMOS Input Mode; interrupt on rising edges.</p>
<p>Note: Pull-up resistors can be assigned to individual Port 2 pins by making the appropriate settings to the P2PUR Control Register, location EEh, Set 1, Bank 0.</p>	

The contents of the Port 2 Control Low Byte (P2CONL) Register are described in Table 53.

Table 53. Port 2 Control Low Byte Register (P2CONL; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					EDh			

Note: R/W = read/write.

Bit	Description
[7:6]	<p>P2.3/INT8 Mode Selection Bits</p> <p>00: CMOS Input Mode; interrupt on falling edges. 01: CMOS Input Mode; interrupt on rising edges and falling edges. 10: Output Mode; push-pull or open-drain output; see the Port 2 Output Mode Selection Register on page 232. 11: CMOS Input Mode; interrupt on rising edges.</p>
[5:4]	<p>P2.2/INT7 Mode Selection Bits</p> <p>00: CMOS Input Mode; interrupt on falling edges. 01: CMOS Input Mode; interrupt on rising edges and falling edges. 10: Output Mode; push-pull or open-drain output; see the Port 2 Output Mode Selection Register on page 232. 11: CMOS Input Mode; interrupt on rising edges.</p>
[3:2]	<p>P2.1/INT6 Mode Selection Bits</p> <p>00: CMOS Input Mode; interrupt on falling edges. 01: CMOS Input Mode; interrupt on rising edges and falling edges. 10: Output Mode; push-pull or open-drain output; see the Port 2 Output Mode Selection Register on page 232. 11: CMOS Input Mode; interrupt on rising edges.</p>

Bit	Description (Continued)
[1:0]	P2.0/INT5 Mode Selection Bits 00: CMOS Input Mode; interrupt on falling edges. 01: CMOS Input Mode; interrupt on rising edges and falling edges. 10: Output Mode; push-pull or open-drain output; see the Port 2 Output Mode Selection Register on page 232. 11: CMOS Input Mode; interrupt on rising edges.
Note: Pull-up resistors can be assigned to individual Port 2 pins by making the appropriate settings to the P2PUR Control Register, location EEh, Set 1, Bank 0.	

The contents of the Port 2 External Interrupt Enable (P2INT) Register are described in Table 54.

Table 54. Port 2 External Interrupt Enable Register (P2INT; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					E5h			

Note: R/W = read/write.

Bit	Description
[7]	P2.7 External Interrupt (INT9) Enable Bit 0: Disable interrupt. 1: Enable interrupt.
[6]	P2.6 External Interrupt (INT9) Enable Bit 0: Disable interrupt. 1: Enable interrupt.
[5]	P2.5 External Interrupt (INT9) Enable Bit 0: Disable interrupt. 1: Enable interrupt.
[4]	P2.4 External Interrupt (INT9) Enable Bit 0: Disable interrupt. 1: Enable interrupt.
[3]	P2.3 External Interrupt (INT8) Enable Bit 0: Disable interrupt. 1: Enable interrupt.
[2]	P2.2 External Interrupt (INT7) Enable Bit 0: Disable interrupt. 1: Enable interrupt.

Bit	Description (Continued)
[1]	P2.1 External Interrupt (INT6) Enable Bit 0: Disable interrupt. 1: Enable interrupt.
[0]	P2.0 External Interrupt (INT5) Enable Bit 0: Disable interrupt. 1: Enable interrupt.

The contents of the Port 2 Output Mode Selection (P2OUTMD) Register are described in Table 55.

Table 55. Port 2 Output Mode Selection Register (P2OUTMD; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					F3h			

Note: R/W = read/write.

Bit	Description
[7]	P2.7 Output Mode Selection Bit 0: Push-Pull Output Mode. 1: Open-Drain Output Mode.
[6]	P2.6 Output Mode Selection Bit 0: Push-Pull Output Mode. 1: Open-Drain Output Mode.
[5]	P2.5 Output Mode Selection Bit 0: Push-Pull Output Mode. 1: Open-Drain Output Mode.
[4]	P2.4 Output Mode Selection Bit 0: Push-Pull Output Mode. 1: Open-Drain Output Mode.
[3]	P2.3 Output Mode Selection Bit 0: Push-Pull Output Mode. 1: Open-Drain Output Mode.
[2]	P2.2 Output Mode Selection Bit 0: Push-Pull Output Mode. 1: Open-Drain Output Mode.

Bit	Description (Continued)
[1]	P2.1 Output Mode Selection Bit 0: Push-Pull Output Mode. 1: Open-Drain Output Mode.
[0]	P2.0 Output Mode Selection Bit 0: Push-Pull Output Mode. 1: Open-Drain Output Mode.

The contents of the Port 2 External Interrupt Pending (P2PND) Register are described in Table 56.

Table 56. Port 2 External Interrupt Pending Register (P2PND; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					E6h			

Note: R/W = read/write.

Bit	Description
[7]	P2.7 External Interrupt (INT9) Pending Flag Bit* 0: No P2.7 external interrupt pending when read. 1: P2.7 external interrupt is pending when read.
[6]	P2.6 External Interrupt (INT9) Pending Flag Bit 0: No P2.6 external interrupt pending when read. 1: P2.6 external interrupt is pending when read.
[5]	P2.5 External Interrupt (INT9) Pending Flag Bit 0: No P2.5 external interrupt pending when read. 1: P2.5 external interrupt is pending when read.
[4]	P2.4 External Interrupt (INT9) Pending Flag Bit 0: No P2.4 external interrupt pending when read. 1: P2.4 external interrupt is pending when read.
[3]	P2.3 External Interrupt (INT8) Pending Flag Bit 0: No P2.3 external interrupt pending when read. 1: P2.3 external interrupt is pending when read.
[2]	P2.2 External Interrupt (INT7) Pending Flag Bit 0: No P2.2 external interrupt pending when read. 1: P2.2 external interrupt is pending when read.
[1]	P2.1 External Interrupt (INT6) Pending Flag Bit 0: No P2.1 external interrupt pending when read. 1: P2.1 external interrupt is pending when read.

Bit	Description (Continued)
[0]	P2.0 External Interrupt (INT5) Pending Flag Bit 0: No P2.0 external interrupt pending when read. 1: P2.0 external interrupt is pending when read.

Note: *To clear an interrupt pending condition, write a 0 to the appropriate pending flag bit. Writing a 1 to an interrupt pending flag (P2PND.0–7) has no effect.

Pull-up resistors can be assigned to the pin circuits of individual pins in Port 2. To assign these pull-up resistors, enter the appropriate settings to the Port 2 Pull-Up Resistor Enable (P2PUR) Register. This register is located in Set 1, Bank 0 at location EEh and is read/write accessible using Register Addressing Mode.

A pull-up resistor can be assigned to the Port 1 and Port 4 pins using the basic port configuration settings in the P1CONH, P1CONL, P4CONH, and P4CONL registers. Pull-up resistors can also be assigned to the Port 3 pins P3.0, P3.1, P3.4, and P3.5 in Input Mode using the basic port configuration settings in the P3CON and P345CON registers. P3.2–P3.3 are configured as input only pins with a pull-up resistor.

Table 57 lists the contents of the Port 2 Pull-Up Resistor Enable Register.

Table 57. Port 2 Pull-Up Resistor Enable Register (P2PUR; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					EEh			

Note: R/W = read/write.

Bit	Description
[7]	P2.7 Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[6]	P2.6 Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[5]	P2.5 Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[4]	P2.4 Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.

Bit	Description (Continued)
[3]	P2.3 Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[2]	P2.2 Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[1]	P2.1 Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[0]	P2.0 Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.

- **Notes:**
1. Pull-up resistors can be assigned to the Port 3 pins, P3.0 and P3.1, by modifying the appropriate settings in the Port 3 Control (P3CON) Register.
 2. Pull-up resistors can be assigned to the Port 3 pins, P3.4 and P3.5, by modifying the appropriate settings in the Port 3[4:5] and Port 3[6:7] Control (P345CON) Register.
 3. Pull-up resistors can be assigned to the P1 and P4 pins by modifying the appropriate settings in the Port 1 Control registers P1CONL and P1CONH, and in the Port 4 control registers P4CONL and P4CONH, respectively.

10.3. Port 3

The contents of the Port 3 Control (P3CON) Register are described in Table 58. The Port 3 Control (P3CON) Register's functions and port assignments are described in Table 59.

Table 58. Port 3 Control Register (P3CON; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					EFh			

Note: R/W = read/write.

Bit	Description
[7:6]	<p>Package Selection and Alternative Function Select Bits</p> <p>00: 32-pin package: P3.0: T0PWM/T0CAP/T1CAP. P3.1: REM/T0CK.</p> <p>01–11: 44-pin package: P3.0: T0PWM/T0CAP. P3.1: REM. P3.2: T0CK. P3.3: T1CAP/T2CAP.</p>
[5]	<p>P3.1 Function Selection Bit</p> <p>0: Normal I/O selection. 1: Alternative function enable (REM/T0CK).</p>
[4:3]	<p>P3.1 Mode Selection Bits</p> <p>00: Schmitt trigger Input Mode. 01: Open- drain Output Mode. 10: Push pull Output Mode. 11: Schmitt trigger input with pull-up resistor.</p>
[2]	<p>Function Selection Bit for P3.0 and P3.3</p> <p>0: Normal I/O selection. 1: Alternative function enable: P3.0: T0PWM/T0CAP. P3.3: T1CAP/T2CAP.</p>

Bit	Description
[1:0]	P3.0 Mode Selection Bits 00: Schmitt trigger Input Mode. 01: Open- drain Output Mode. 10: Push pull Output Mode. 11: Schmitt trigger input with pull-up resistor.

Note:

- The Port 3 Data Register, P3, at location E3h, Set1, Bank0, contains seven bit values which correspond to the following Port 3 pin functions (bit 6 is not used for the S3F80PB MCU):
 - Port 3, bit[7]: carrier signal on (1) or off (0).
 - Port 3, bit[1:0]: P3.1/REM/T0CK pin, bit 0: P3.0/T0PWM/T0CAP/T1CAP pin.
 - Port 3, bit[3:2]: P3.3, P3.2 are selected only to input pin with pull-up resistor automatically.
 - Port 3, bit[5:4]: P3.5, P3.4 are selected into digital I/O by setting P345CON Register at E1h, Set1, Bank1.
- The alternative function enable/disable are enabled in accordance with function selection bit (bit[5] and bit[2]).
- The pin assign for alternative functions can be selectable relating to mode selection bit (bit0, 1, 2, 3, 4 and 5).
- See [Table 59](#) on page 237 for specific alternative function examples and pin assignments for P3CON.

Table 59. Function Description and Pin Assignment of P3CON (44-Pin Packages)

P3CON							Function Description and Assignment to P3.0–P3.3			
B5	B4	B3	B2	B1	B0	P3.0	P3.1	P3.2	P3.3	
0	x	x	0	x	x	Normal I/O	Normal I/O	Normal Input	Normal Input	
0	x	x	1	0	0	T0_CAP	Normal I/O	Normal Input	T1CAP/Normal Input	
0	x	x	1	1	1	T0_CAP	Normal I/O	Normal Input	T1CAP/Normal Input	
0	x	x	1	0	1	T0PWM	Normal I/O	Normal Input	T1CAP/Normal Input	
0	x	x	1	1	0	T0PWM	Normal I/O	Normal Input	T1CAP/Normal Input	
1	0	0	0	x	x	Normal I/O	Normal Input	T0CK	Normal Input	
1	1	1	0	x	x	Normal I/O	Normal Input	T0CK	Normal Input	
1	0	1	0	x	x	Normal I/O	REM	T0CK	Normal Input	
1	1	0	0	x	x	Normal I/O	REM	T0CK	Normal Input	
1	0	0	1	0	0	T0_CAP	Normal Input	T0CK/Normal Input	T1CAP/Normal Input	
1	1	1	1	1	1	T0_CAP	Normal Input	T0CK/Normal Input	T1CAP/Normal Input	
1	0	1	1	0	1	T0PWM	REM	T0CK/Normal Input	T1CAP/Normal Input	
1	1	0	1	1	0	T0PWM	REM	T0CK/Normal Input	T1CAP/Normal Input	
1	0	0	1	0	1	T0PWM	Normal Input	T0CK/Normal Input	T1CAP/Normal Input	
1	1	1	1	1	0	T0PWM	Normal Input	T0CK/Normal Input	T1CAP/Normal Input	
1	0	1	1	0	0	T0_CAP	REM	T0CK/Normal Input	T1CAP/Normal Input	
1	1	0	1	1	1	T0_CAP	REM	T0CK/Normal Input	T1CAP/Normal Input	

The contents of the Port 3 Output Pull-Up Resistor Enable (P3OUTPU) Register are described in Table 60.

Table 60. Port 3 Output Pull-Up Resistor Enable Register (P3OUTPU; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	–	–	0	0	–	–	0	0
R/W	–	–	R/W	R/W	–	–	R/W	R/W
Address	F4h							

Note: R/W = read/write.

Bit	Description
[7:6]	Reserved These bits are reserved and must be set to 00.
[5]	P3.5 Output Mode Pull-up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[4]	P3.4 Output Mode Pull-up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[3:2]	Reserved These bits are reserved and must be set to 00.
[1]	P3.1 Output Mode Pull-up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[0]	P3.0 Output Mode Pull-up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.

The contents of the Port 3[4:5] Control (P345CON) Register are described in Table 61.

Table 61. Port 3[4:5] Control Register (P345CON; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	1	0	1	–	–	–	0
R/W	R/W	R/W	R/W	R/W	–	–	–	R/W
Address	E1h							

Note: R/W = read/write.

Bit	Description
[7:6]	P3.5 Mode Selection Bits 00: CMOS input mode. 01: Open-drain output mode. 10: Push-pull output mode 11: CMOS input with pull up mode.
[5:4]	P3.4 Mode Selection Bits 00: CMOS input mode. 01: Open-drain output mode. 10: Push-pull output mode 11: CMOS input with pull up mode.
[3:1]	Reserved These bits are reserved and must be set to 000.
[0]	Port 4 Control Register Selection Bit 0: P4CON Register selection. 1: P4CONH/P4CONL Register selection.

Note: After a CPU reset, P3.4 and P3.5 will be Open-drain output mode by the reset value of P345CON register at E1h, Set1, and Bank1. P345CON will be initialized as 50h to set P3.4 into the open-drain output mode after reset operation. Port 4 Control Register (P4CON) will be selected by the reset value of P345CON.0 bit. If Port 4 input and output modes are being used, set P345CON.0 to 1.

The contents of the Port 4 Control (P4CON) Register are described in Table 62.

Table 62. Port 4 Control Register (P4CON; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					F0h			

Note: R/W = read/write.

Bit	Description
[7]	P4.7 Mode Selection Bit 0: Open-Drain Output Mode. 1: Push-Pull Output Mode.
[6]	P4.6 Mode Selection Bit 0: Open-Drain Output Mode. 1: Push-Pull Output Mode.
[5]	P4.5 Mode Selection Bit 0: Open-Drain Output Mode. 1: Push-Pull Output Mode.
[4]	P4.4 Mode Selection Bit 0: Open-Drain Output Mode. 1: Push-Pull Output Mode.
[3]	P4.3 Mode Selection Bit 0: Open-Drain Output Mode. 1: Push-Pull Output Mode.
[2]	P4.2 Mode Selection Bit 0: Open-Drain Output Mode. 1: Push-Pull Output Mode.
[1]	P4.1 Mode Selection Bit 0: Open-Drain Output Mode. 1: Push-Pull Output Mode.
[0]	P4.0 Mode Selection Bit 0: Open-Drain Output Mode. 1: Push-Pull Output Mode.

The contents of the Port 4 Control High Byte (P4CONH) Register are described in Table 63.

Table 63. Port 4 Control High Byte Register (P4CONH; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	1	1	1	1	1	1	1	1
R/W					R/W			
Address					E2h			

Note: R/W = read/write.

Bit	Description
[7:6]	P4.7 Mode Selection Bits 00: CMOS Input Mode. 01: CMOS Input Mode. 10: Output Mode. 11: CMOS Input Mode.
[5:4]	P4.6 Mode Selection Bits 00: CMOS Input Mode. 01: CMOS Input Mode. 10: Output Mode. 11: CMOS Input Mode.
[3:2]	P4.5 Mode Selection Bits 00: CMOS Input Mode. 01: CMOS Input Mode. 10: Output Mode. 11: CMOS Input Mode.
[1:0]	P4.4 Mode Selection Bits 00: CMOS Input Mode. 01: CMOS Input Mode. 10: Output Mode. 11: CMOS Input Mode.

Note: After a CPU reset, P4.7–P4.4 will be reset to CMOS inputs with Pull-Up Mode.

The contents of the Port 4 Control Low Byte (P4CONL) Register are described in Table 64.

Table 64. Port 4 Control Low Byte Register (P4CONL; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	1	1	1	1	1	1	1	1
R/W					R/W			
Address					E3h			

Note: R/W = read/write.

Bit	Description
[7:6]	P4.3 Mode Selection Bits 00: CMOS Input Mode. 01: Open-Drain Output Mode. 10: Push-Pull Output Mode. 11: CMOS Input with Pull-Up Mode.
[5:4]	P4.2 Mode Selection Bits 00: CMOS Input Mode. 01: Open-Drain Output Mode. 10: Push-Pull Output Mode. 11: CMOS Input with Pull-Up Mode.
[3:2]	P4.1 Mode Selection Bits 00: CMOS Input Mode. 01: Open-Drain Output Mode. 10: Push-Pull Output Mode. 11: CMOS Input with Pull-Up Mode.
[1:0]	P4.0 Mode Selection Bits 00: CMOS Input Mode. 01: Open-Drain Output Mode. 10: Push-Pull Output Mode. 11: CMOS Input with Pull-Up Mode.

Note: After a CPU reset, P4.3–P4.0 will be reset to CMOS inputs with Pull-Up Mode.

The contents of the Port 4 Output Pull-Up Resistor Enable (P4OUTPU) Register are described in Table 65.

Table 65. Port 4 Output Pull-Up Resistor Enable Register (P4OUTPU; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					F5h			

Note: R/W = read/write.

Bit	Description
[7]	P4.7 Output Mode Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[6]	P4.6 Output Mode Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[5]	P4.5 Output Mode Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[4]	P4.4 Output Mode Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[3]	P4.3 Output Mode Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[2]	P4.2 Output Mode Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[1]	P4.1 Output Mode Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[0]	P4.0 Output Mode Pull-Up Resistor Enable Bit 0: Disable pull-up resistor. 1: Enable pull-up resistor.

Chapter 11. Basic Timer and Timer 0

The S3F80PB MCU contains two default timers: an 8-bit basic timer and an 8-bit general-purpose timer/counter. The 8-bit timer/counter is called *Timer 0* (T0). A block diagram of these two timers is shown in Figure 72.

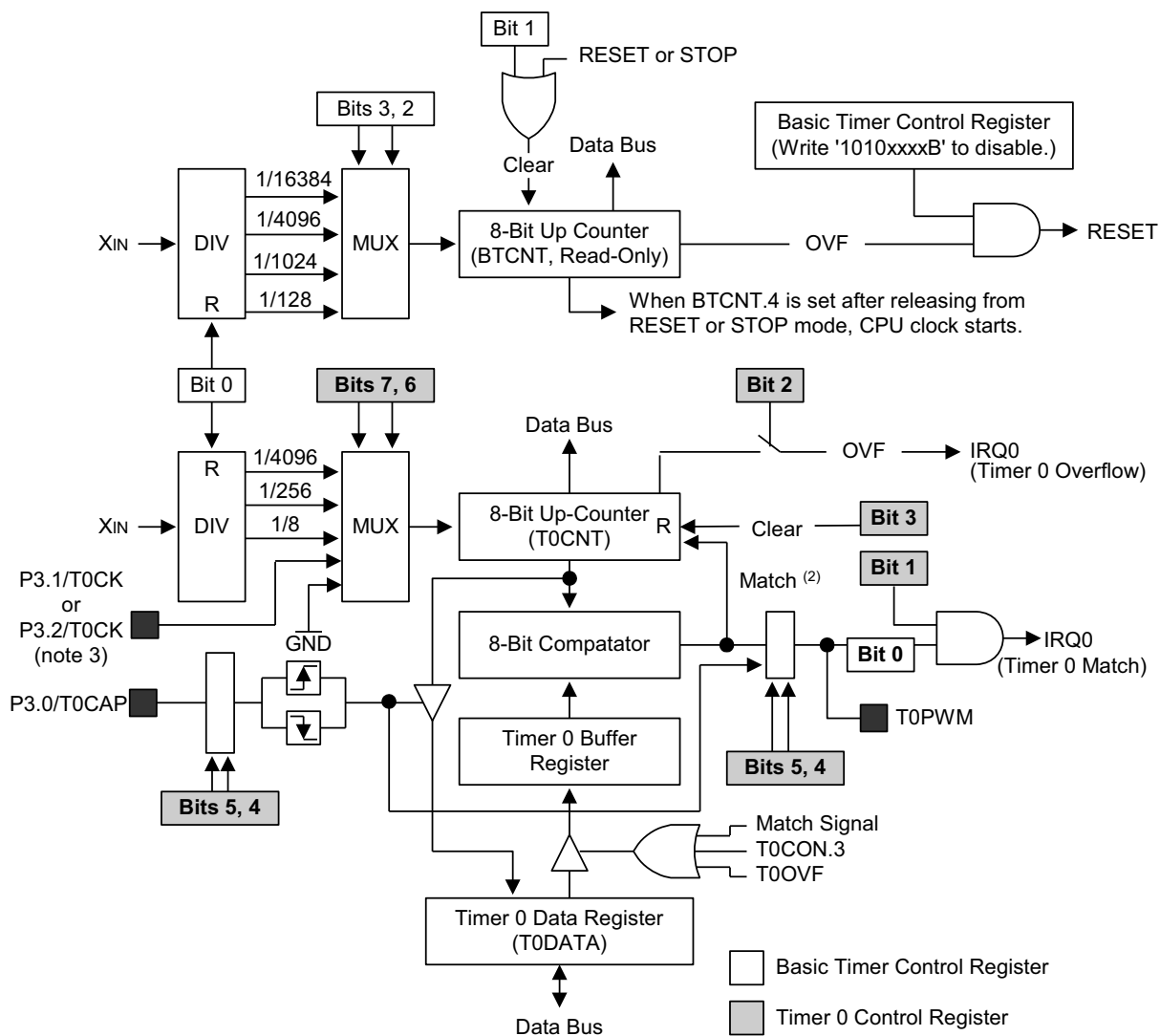


Figure 72. Basic Timer and Timer 0 Block Diagram

-
- **Notes:**
1. During a power-on reset operation, the CPU is idle during the required oscillation stabilization interval (i.e., until bit 4 of the basic timer counter overflows)
 2. Available only while using internal mode.
 3. The external clock source is P3.1/T0CK in the 32-pin package, or P3.2/T0CK in 44-pin packages.
-

11.1. Basic Timer

Use the Basic Timer (BT) in two different ways:

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

The functional components of the basic timer block are:

- Clock frequency divider (f_{OSC} divided by 16384, 4096, 1024 or 128) with multiplexer
- 8-bit Basic Timer Counter, BTCNT (FDh, Set1, Bank0, read only)
- Basic Timer Control Register, BTCN (D3h, Set1, Bank0, R/W)

11.1.0.1. Timer 0

Timer 0 contains three operating modes which are selected using the appropriate T0CON setting:

- Interval Timer Mode
- Capture Input Mode with a rising or falling edge trigger at the P3.0 pin
- PWM Mode

Timer 0 contains the following functional components:

- Clock frequency divider (f_{OSC} divided by 4096, 256, or 8) with multiplexer
- External clock input pin (T0CK)
- 8-bit Timer 0 Counter (T0CNT), 8-bit comparator, and 8-bit Reference Data (T0DATA) Register
- I/O pins for capture input (T0CAP) or match output

- Timer 0 overflow interrupt (IRQ0, vector FAh) and match/capture interrupt (IRQ0, vector FCh) generation
- Timer 0 Control Register, T0CON (D2h, Set1, Bank0, R/W)

► **Note:** The CPU clock should be faster than both the basic timer clock and the Timer 0 clock.

11.2. Basic Timer Control Register

The Basic Timer Control Register, BTCON, is used to select the input clock frequency, to clear the basic timer counter and frequency dividers, and to enable or disable the watchdog timer function. BTCON is located in Set1 and Bank0, address D3h, and is read/write-addressable using Register Addressing Mode.

A reset clears BTCON to 00h, enabling the watchdog function and selecting a basic timer clock frequency of $f_{OSC}/4096$. To disable the watchdog function, you must write the signature code 1010b to the basic timer register control bits BTCON.7–BTCON.4. For improved reliability, using the watchdog timer function is recommended in remote controllers and hand-held product applications.

The contents of the Basic Timer Control (BTCN) Register are detailed in Table 66.

Table 66. Basic Timer Control Register (BTCN; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address	D3h							

Note: R/W = read/write.

Bit	Description
[7:4]	Watchdog Timer Function Enable Bits (For System Reset) 0000–1001: Enable watchdog timer function. 1010: Disable watchdog timer function. 1011–1111: Enable watchdog timer function.
[3:2]	Basic Timer Input Clock Selection Bits 00: $f_{OSC}/4096$. 01: $f_{OSC}/1024$. 10: $f_{OSC}/128$. 11: $f_{OSC}/16384$.
[1]	Basic Timer Counter Clear Bit¹ 0: No effect. 1: Clear the basic timer counter value.
[0]	Clock Frequency Divider Clear Bit for Basic Timer and Timer 0² 0: No effect. 1: Clear both block frequency dividers.

Notes:

1. When writing a 1 to BTCN.1, the basic timer counter value is cleared to 00h. Immediately following the write operation, the BTCN.1 value is automatically cleared to 0.
2. When writing a 1 to BTCN.0, the corresponding frequency divider is cleared to 00h. Immediately following the write operation, the BTCN.0 value is automatically cleared to 0.

Watchdog Timer Function. Program the basic timer overflow signal (BTOVF) to generate a reset by setting BTCN.7–BTCN.4 to any value other than 1010b. (The 1010b value disables the watchdog function.) A reset clears BTCN to 00h, automatically enabling the watchdog timer function. A reset also selects the CPU clock (as determined by the current CLKCON Register setting), divided by 4096, as the BT clock.

A reset is generated whenever the basic timer overflow occurs. During normal operation, the application program must prevent the overflow, and the accompanying reset operation, from occurring. To prevent the overflow and accompanying reset operation, the BTCNT value must be cleared (i.e., by writing a 1 to BTCN.1) at regular intervals.

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

Oscillation Stabilization Interval Timer Function. Use the basic timer to program a specific oscillation stabilization interval following a reset or when Stop Mode is released by an external interrupt.

In Stop Mode, whenever a reset or an external interrupt occurs, the oscillator starts. The BTCNT value then starts increasing at the rate of $f_{OSC}/4096$ (i.e., for reset), or at the rate of the preset clock source (i.e., for an external interrupt). When the BTCNT.3 overflows, a signal is generated to indicate that the stabilization interval has elapsed and to gate the clock signal off to the CPU so that it can resume normal operation.

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

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

11.3. Timer 0 Control Register

Use the Timer 0 Control (T0CON) Register to:

- Select the Timer 0 operating mode (interval timer, capture, or PWM)
- Select the Timer 0 input clock frequency
- Clear the Timer 0 Counter (T0CNT) Register
- Enable the Timer 0 overflow interrupt or the Timer 0 match/capture interrupt
- Clear Timer 0 match/capture interrupt, pending conditions

The Timer 0 Control (T0CON) Register, described in Table 67, is located in Set1, Bank0, at address D2h, and is read/write-addressable using Register Addressing Mode.

Table 67. Timer 0 Control Register (T0CON; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					D2h			

Note: R/W = read/write.

Bit	Description
[7:6]	Timer 0 Input Clock Selection Bits 00: $f_{OSC}/4096$. 01: $f_{OSC}/256$. 10: $f_{OSC}/8$. 11: External clock input (At the T0CK pin, P3.1 or P3.2).
[5:4]	Timer 0 Operating Mode Selection Bits 00: Interval Timer Mode; counter cleared by match signal. 01: Capture Mode; rising edges, counter running, OVF interrupt can occur. 10: Capture Mode; falling edges, counter running, OVF interrupt can occur. 11: PWM Mode; match and OVF interrupt can occur.
[3]	Timer 0 Counter Clear Bit 0: No effect when written. 1: Clear T0 counter, T0CNT when written.
[2]	Timer 0 Overflow Interrupt Enable Bit* 0: Disable T0 overflow interrupt. 1: Enable T0 overflow interrupt.
[1]	Timer 0 Match/Capture Interrupt Enable Bit 0: Disable T0 match/capture interrupt. 1: Enable T0 match/capture interrupt.
[0]	Timer 0 Match/Capture Interrupt Pending Flag Bit 0: No T0 match/capture interrupt pending when read. 0: Clear T0 match/capture interrupt pending condition when write. 1: T0 match/capture interrupt is pending when read. 1: No effect when write.

Note: *A Timer 0 overflow interrupt pending condition is automatically cleared by hardware. However, the Timer 0 match/capture interrupt, IRQ0, vector FCh, must be cleared by the interrupt service routine (software).

A reset clears T0CON to 00h, sets Timer 0 to normal Interval Timer Mode, selects an input clock frequency of $f_{OSC}/4096$, and disables all Timer 0 interrupts. The Timer 0 counter can be cleared at any time during normal operation by writing a 1 to T0CON.3.

The Timer 0 overflow interrupt (T0OVF) is interrupt level IRQ0 and contains the vector address FAh. When a Timer 0 overflow interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware.

To enable the Timer 0 match/capture interrupt (IRQ0, vector FCh), write T0CON.1 to 1. To detect a match/capture interrupt pending condition, the application program polls T0CON.0. When a 1 is detected, a Timer 0 match or capture interrupt is pending. When the interrupt request is serviced, the pending condition must be cleared by software by writing a 0 to the Timer 0 interrupt pending bit, T0CON.0.

Timer 0 Interrupts (IRQ0, Vectors FAh and FCh). The Timer 0 module can generate two interrupts: the Timer 0 overflow interrupts (T0OVF) and the Timer 0 match/capture interrupt (T0INT). T0OVF is a level IRQ0 interrupt with a vector address of FAh. T0INT also belongs to interrupt level IRQ0, but is assigned the separate vector address FCh.

A pending condition on the Timer 0 overflow interrupt (T0OVF) is automatically cleared by hardware when it is serviced. This T0INT pending condition must, however, be cleared by the application's interrupt service routine by writing a 1 to the T0CON.0 interrupt pending bit.

Interval Timer Mode. In Interval Timer Mode, a match signal is generated when the counter value is identical to the value written to the T0 Reference Data (T0DATA) Register; see Table 68. The match signal generates a Timer 0 match interrupt (T0INT, vector FCh) and clears the counter.

Table 68. Timer 0 Reference Data Register (T0DATA; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset								FFh
R/W								R/W
Address								D1h

Note: R/W = read/write.

Bit	Description
[7:0]	Timer 0 Reference Data Register

If, for example, you write the value 10h to T0DATA and the value 0Bh to T0CON, the counter will increment until it reaches 10h. At this point, a T0 interrupt request is generated, and after the counter value is reset, counting resumes. With each match, the level of the signal at the Timer 0 output pin is inverted, as shown in Figure 73.

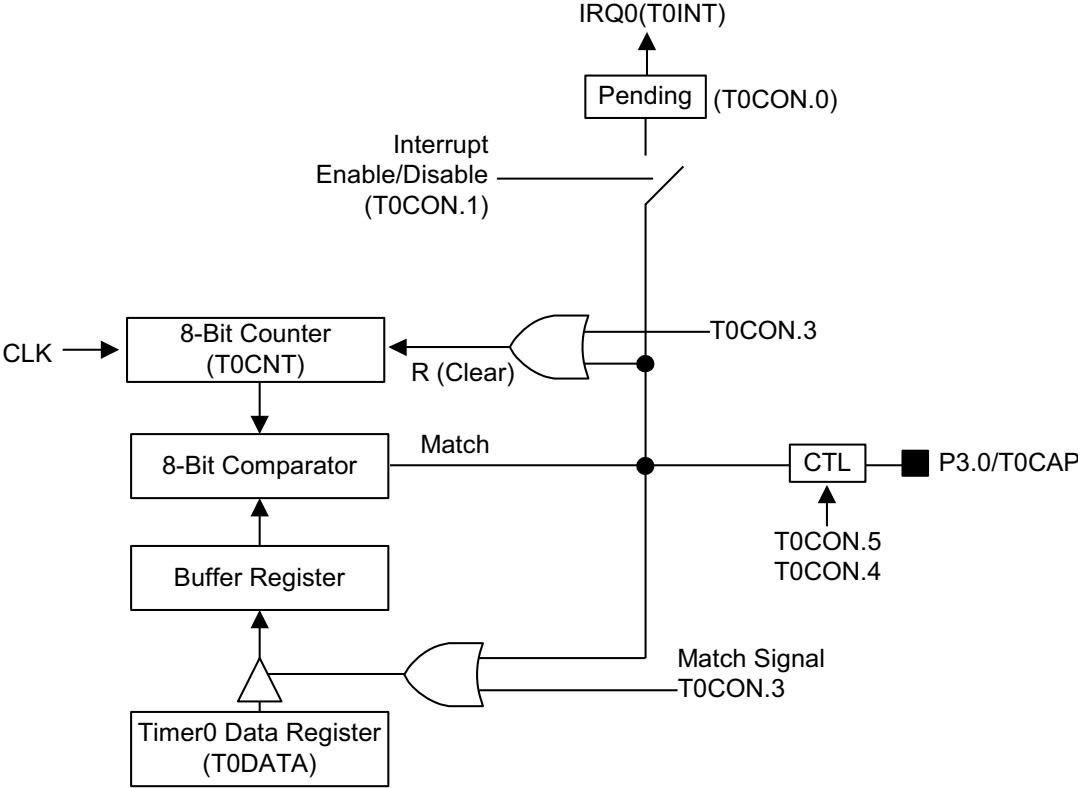


Figure 73. Simplified Timer 0 Function Diagram: Interval Timer Mode

Pulse Width Modulation Mode. Pulse Width Modulation (PWM) Mode lets you program the width (duration) of the pulse that is output at the TOPWM pin. As in Interval Timer Mode, a match signal is generated when the counter value is identical to the value written to the Timer 0 Data Register. In PWM Mode, however, the match signal does not clear the counter. Instead, it runs continuously, overflowing at FFh, and then continues incrementing from 00h.

Although it is possible to use the match signal to generate a Timer 0 overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the TOPWM pin is held to low level as long as the reference data value is less than or equal to (\leq) the counter value and then the pulse is held to high level for as long as the data value is greater than ($>$) the counter value. One pulse width is equal to $t_{CLK} \times 256$, as shown in Figure 74.

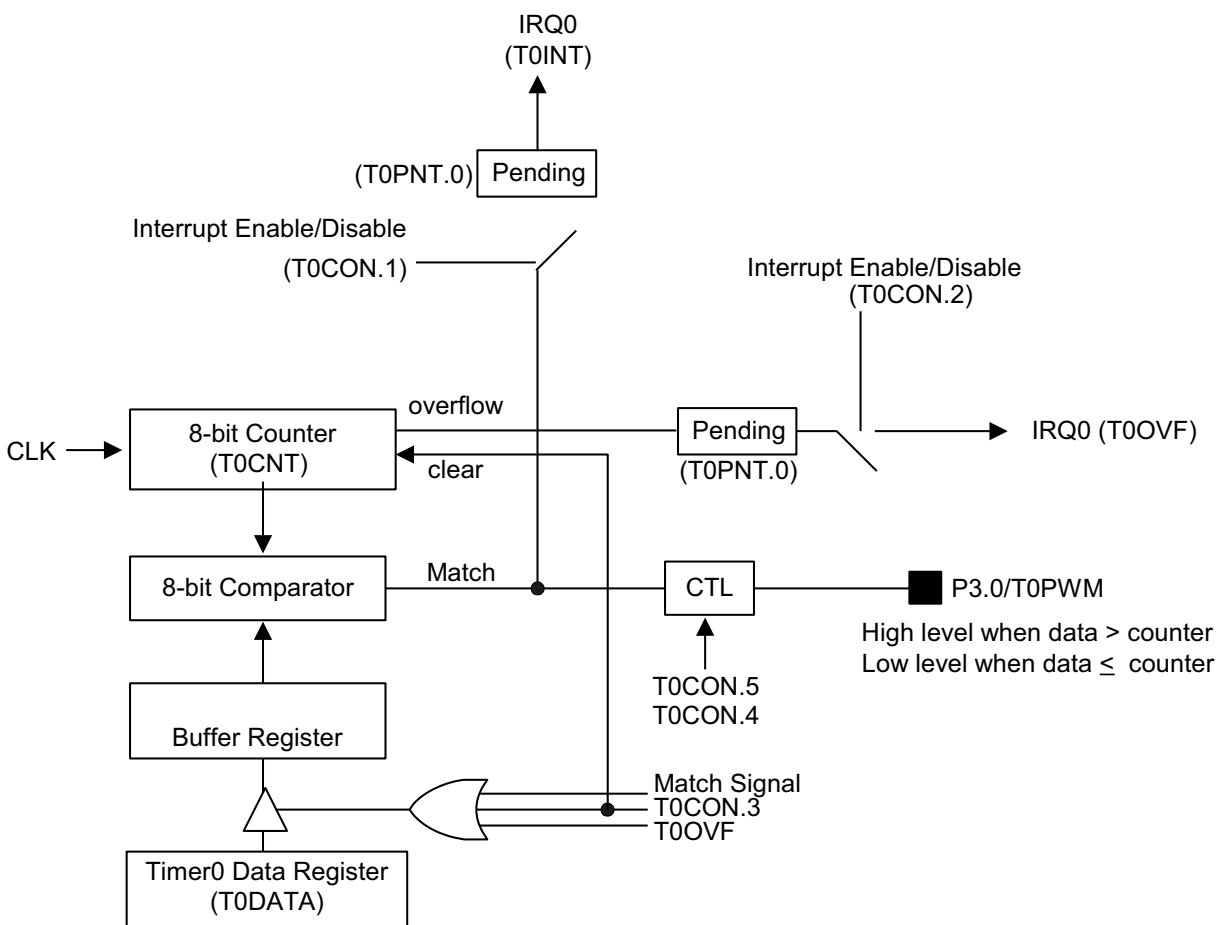


Figure 74. Simplified Timer 0 Function Diagram: PWM Mode

Capture Mode. In Capture Mode, a signal edge that is detected at the T0CAP pin opens a gate and loads the current counter value into the T0 Data Register. Select rising or falling edges to trigger this operation.

Timer 0 also provides a capture input source: the signal edge at the T0CAP pin. Select the capture input by setting the value of the Timer 0 capture input selection bit in the Port 3 Control Register, P3CON.2, (Set1, Bank0, EFh). When the P3CON.2 is 1, the T0CAP input is selected. When the P3CON.2 is set to 0, normal I/O port (P3.0) is selected.

Both types of Timer 0 interrupts can be used in Capture Mode: the Timer 0 overflow interrupt is generated whenever a counter overflow occurs; the Timer 0 match/capture interrupt is generated whenever the counter value is loaded into the T0 Data Register.

By reading the captured data value in T0DATA, and assuming a specific value for the Timer 0 clock frequency, calculate the pulse width (i.e., duration) of the signal that is being input at the T0CAP pin, as shown in Figure 75.

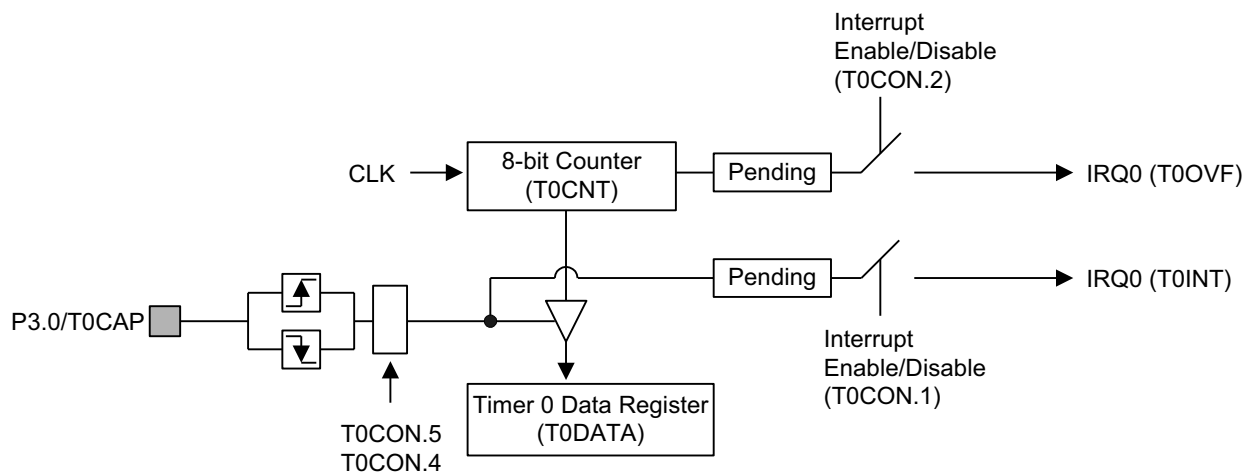


Figure 75. Simplified Timer 0 Function Diagram: Capture Mode

The following example shows how to configure the basic timer to sample specifications.

```

ORG      0100h

Reset DI          ; Disable all interrupts
LD       BTCON, #0AAh ; Disable the watchdog timer
LD       CLKCON, #18h ; Nondivided clock
CLR      SYM      ; Disable global and fast
           ; interrupts
CLR      SPL      ; Stack pointer low byte → 0
           ; Stack area starts at 0FFh
    
```

```

.
.
.
SRP      #0C0h          ; Set register pointer → 0C0h
EI                          ; Enable interrupts
.
.
.
MAIN LD      BTCON, #52h      ; Enable the watchdog timer
                                ; Basic timer clock: fOSC/4096
                                ; Clear basic timer counter

NOP
NOP
.
.
.
JP      T, MAIN
.
.
.

```

The sample program in the next example sets Timer 0 to Interval Timer Mode, sets the frequency of the oscillator clock, and determines the execution sequence that follows a Timer 0 interrupt. The program parameters are as follows:

- Timer 0 is used in Interval Mode; the timer interval is set to 4 milliseconds
- Oscillation frequency is 6MHz
- General register 60h (Page 0) → 60h + 61h + 62h + 63h + 64h (Page 0) is executed after a Timer 0 interrupt

```

VECTOR  00FAh, T0OVER      ; Timer 0 overflow interrupt
VECTOR  00FCh, T0INT      ; Timer 0 match/capture interrupt
ORG      0100h
RESET: DI                  ; Disable all interrupts
LD      BTCON, #0AAh      ; Disable the watchdog timer
LD      CLKCON, #18h      ; Select nondivided clock
CLR     SYM                ; Disable global and fast
                                ; interrupts
CLR     SPL                ; Stack pointer low byte → 0
                                ; Stack area starts at 0FFh
.
.
.
LD      T0CON, #4Bh        ; Write 00100101b
                                ; Input clock is fOSC/256

```

```

; Interval Timer Mode
; Enable the Timer 0 interrupt
; Disable the Timer 0 overflow
; interrupt
LD      TODATA, #5Dh      ; Set timer interval to 4
; milliseconds
; (6MHz/256)/(93+1)=0.25kHz (4ms)
SRP    #0C0h            ; Set register pointer → 0C0h
EI      ; Enable interrupts
.
.
.
TOINT: PUSH  RP0        ; Save RP0 to stack
        SRP0    #60h    ; RP0 ← 60h
        INC     R0      ; R0 ← R0 +1
        ADD    R2, R0   ; R2 ← R2 + R0
        ADC    R3, R2   ; R3 ← R3 + R2 + Carry
        ADC    R4, R0   ; R4 ← R4 + R0 + Carry
        CP     R0, #32h ; 50 × 4 = 200ms
        JR     ULT, NO_200MS_SET
        BITS   R1.2    ; Bit setting (61.2h)
NO_200MS_SET
        LD     TOCON, #42h ; Clear pending bit
        POP    RP0      ; Restore register pointer 0 value
TOOVER IRET             ; Return from interrupt service
; routine

```


Chapter 12. Timer 1

The S3F80PB microcontroller features a 16-bit timer/counter called *Timer 1* (T1). For universal remote controller applications, Timer 1 can be used to generate the envelope pattern for the remote controller signal. A block diagram of Timer 1 is shown in Figure 76.

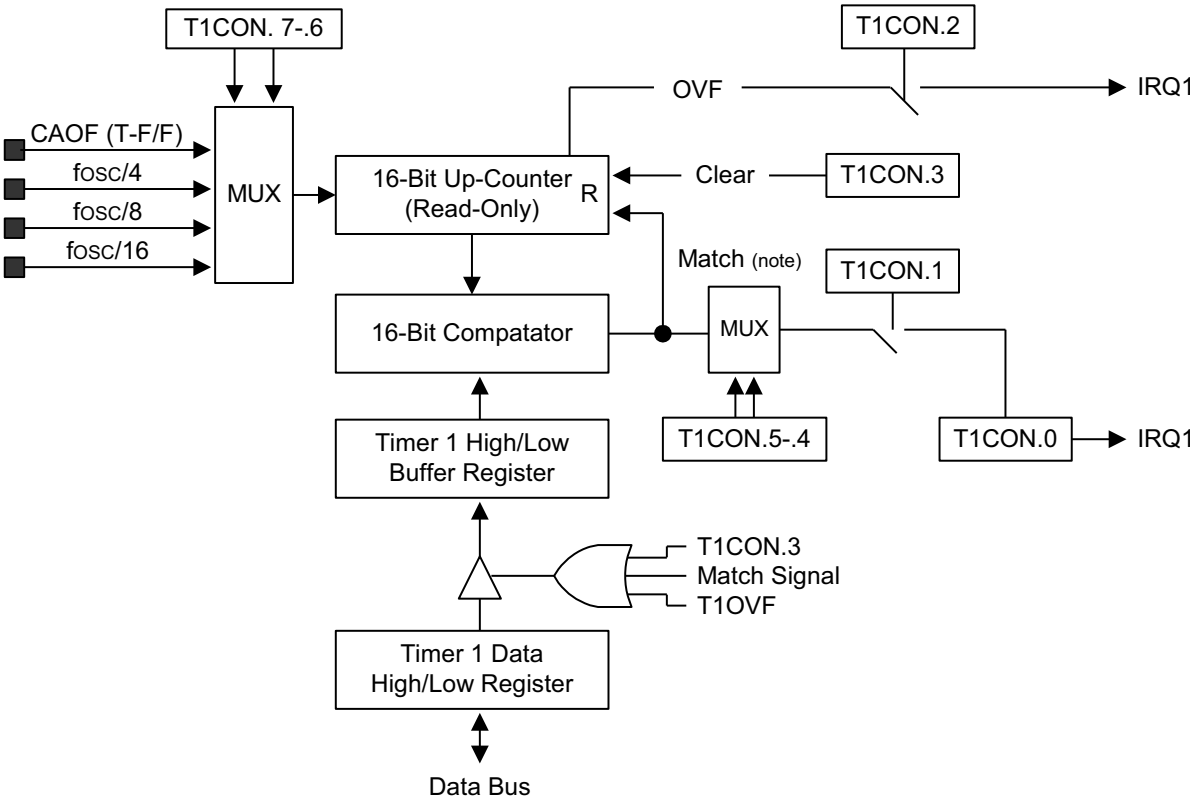


Figure 76. Timer 1 Block Diagram

► **Note:** The Match signal only occurs in Interval Mode.

Timer 1 features the following components.

- One control register, T1CON (FAh, Set1, Bank0, R/W)
- Two 8-bit counter registers, T1CNTH and T1CNTL (F6h and F7h, Set1, Bank0, read only)
- Two 8-bit reference data registers, T1DATAH and T1DATAL (F8h and F9h, Set1, Bank0, R/W)
- One 16-bit comparator

Select one of the following clock sources as the Timer 1 clock:

- Oscillator frequency (f_{OSC}) divided by 4, 8, or 16
- Internal clock input from the Counter A module (Counter A flip/flop output)

Timer 1 can be used in three ways:

- As a normal free run counter, generating a Timer 1 overflow interrupt (IRQ1, vector F4h) at programmed time intervals
- To generate a Timer 1 match interrupt (IRQ1, vector F6h) when the 16-bit Timer 1 count value matches the 16-bit value written to the reference data registers
- To generate a Timer 1 capture interrupt (IRQ1, vector F6h) when a triggering condition exists at the P3.2 pin on the 44-pin package and at the P3.0 pin on the 32-pin package (i.e., select a rising edge, a falling edge, or both edges as the trigger)

In the S3F80PB MCU's interrupt structure, the Timer 1 overflow interrupt receives higher priority than the Timer 1 match or capture interrupts.

► **Note:** The CPU clock should be faster than the Timer 1 clock.

12.1. Timer 1 Control Register

The Timer 1 Control (T1CON) Register, described in Table 69, is located in Set1, Bank0, FAh and is read/write-addressable.

Table 69. Timer 1 Control Register (T1CON; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					FAh			

Note: R/W = read/write.

Bit	Description
[7:6]	Timer 1 Input Clock Selection Bits 00: $f_{OSC} / 4$. 01: $f_{OSC} / 8$. 10: $f_{OSC} / 16$. 11: Internal clock. (Counter A Flip-Flop, T-FF).
[5:4]	Timer 1 Operating Mode Selection Bits 00: Interval Timer Mode (Counter cleared by match signal). 01: Capture Mode (Rising edges, counter running, OVF can occur). 10: Capture Mode (Falling edges, counter running, OVF can occur). 11: Capture Mode (Rising and falling edges, counter running, OVF can occur).
[3]	Timer 1 Counter Clear Bit 0: No effect when write. 1: Clear T ₁ counter, T1CNT when write.
[2]	Timer 1 Overflow Interrupt Enable Bit* 0: Disable T ₁ overflow interrupt. 1: Enable T ₁ overflow interrupt.
[1]	Timer 1 Match/Capture Interrupt Enable Bit 0: Disable T ₁ match/capture interrupt. 1: Enable T ₁ match/capture interrupt.
[0]	Timer 1 Match/Capture Interrupt Pending Flag Bit 0: No T ₁ match/capture interrupt pending when read. 0: Clear T ₁ match/capture interrupt pending condition when write. 1: T ₁ match/capture interrupt is pending when read. 1: No effect when write.

Note: * A timer 1 overflow interrupt pending condition is automatically cleared by hardware. However, the timer 1 match/capture interrupt, IRQ1, vector F6h, must be cleared by the interrupt service routine (SW).

T1CON contains control settings for the following T1 functions:

- Timer 1 input clock selection
- Timer 1 operating mode selection
- Timer 1 16-bit downcounter clear
- Timer 1 overflow interrupt enable/disable
- Timer 1 match or capture interrupt enable/disable
- Timer 1 interrupt pending control (i.e., read for status, write to clear)

A reset operation clears T1CON to 00h, selecting f_{OSC} divided by 4 as the T1 clock, configuring Timer 1 as a normal interval timer, and disabling the Timer 1 interrupts.

The Timer 1 registers, T1CNTH, T1CNTL, T1DATAH, T1DATAL (Set1, Bank0), are described in Tables 70 through 73.

Table 70. Timer 1 Counter High Byte Register

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	F6h							

Note: R = read only; reset value = 00h.

Table 71. Timer 1 Counter Low Byte Register

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	F7h							

Note: R = read only; reset value = 00h.

Table 72. Timer 1 Data High Byte Register

Bit	7	6	5	4	3	2	1	0
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F8h							

Note: R/W = read/write; reset value = FFh.

Table 73. Timer 1 Data Low Byte Register

Bit	7	6	5	4	3	2	1	0
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F9h							

Note: R/W = read/write; reset value = FFh.

12.2. Timer 1 Overflow Interrupt

Timer 1 can be programmed to generate an overflow interrupt (IRQ1, F4h) whenever an overflow occurs in the 16-bit upcounter. When the Timer 1 overflow interrupt enable bit, T1CON.2, is set to 1, an overflow interrupt is generated each time the 16-bit upcounter reaches FFFFh. After the interrupt request is generated, the counter value is automatically cleared to 00h and up counting resumes. Clear/reset the 16-bit counter value at any time during program operation by writing a 1 to T1CON.3.

12.3. Timer 1 Capture Interrupt

Timer 1 can be used to generate a capture interrupt (IRQ1, vector F6h) whenever a triggering condition is detected at the P3.0 pin. The T1CON.5 and T1CON.4 bit-pair setting is used to select the trigger condition for Capture Mode operation: rising edges, falling edges, or both signal edges.

In Capture Mode, program software can poll the Timer 1 match/capture interrupt pending bit, T1CON.0, to detect when a Timer 1 capture interrupt pending condition exists (T1CON.0 = 1). When the interrupt request is acknowledged by the CPU and the service

routine starts, the interrupt service routine for vector F6h must clear the interrupt pending condition by writing a 0 to T1CON.0. See Figure 77.

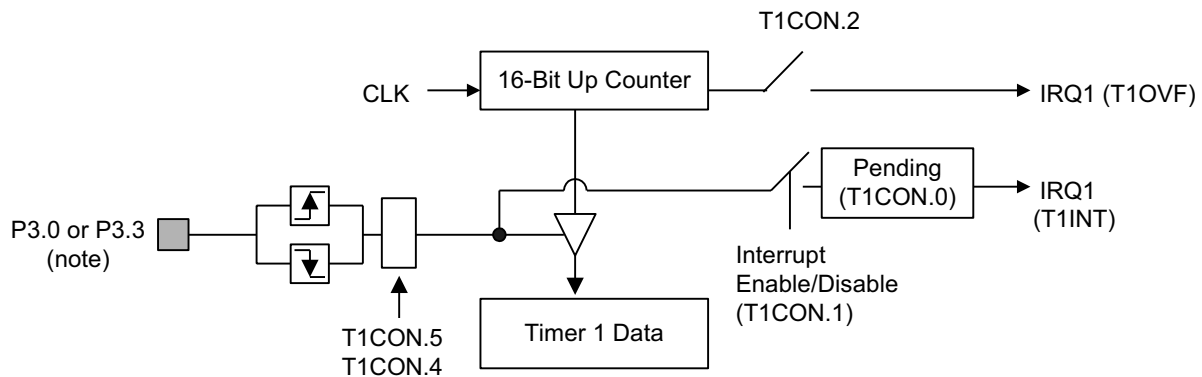


Figure 77. Simplified Timer 1 Function Diagram: Capture Mode

► **Note:** P3.0 is assigned as T1CAP function for the 32-pin package and P3.3 is assigned as T1CAP function for the 44-pin packages.

12.4. Timer 1 Match Interrupt

In addition to generating capture interrupts, Timer 1 can also be used to generate a match interrupt (IRQ1, vector F6h) whenever the 16-bit counter value matches the value that is written to the Timer 1 reference data registers, T1DATAH and T1DATAH. When a match condition is detected by the 16-bit comparator, the match interrupt is generated, the counter value is cleared, and up counting resumes from 00h.

In Match Mode, program software can poll the Timer 1 match/capture interrupt pending bit, T1CON.0, to detect when a Timer 1 match interrupt pending condition exists (T1CON.0 = 1). When the interrupt request is acknowledged by the CPU and the service

routine starts, the interrupt service routine for vector F6h must clear the interrupt pending condition by writing a 0 to T1CON.0. See Figure 78.

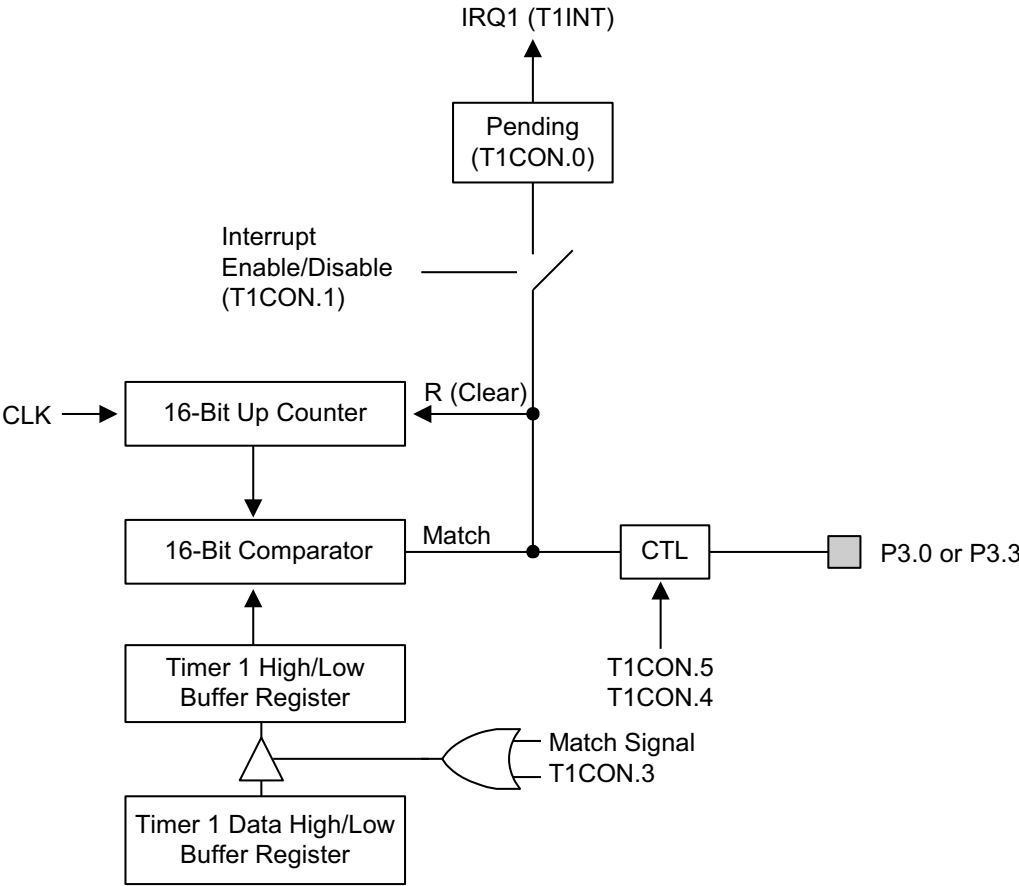


Figure 78. Simplified Timer 1 Function Diagram: Interval Timer Mode

Chapter 13. Counter A

The S3F80PB MCU features an 8-bit counter, Counter A, which can be used to generate a carrier frequency. Counter A features the following components, which can be seen in Figure 79.

- Counter A Control (CACON) Register
- 8-bit downcounter with automatic reload function
- Two 8-bit reference data registers (CADATAH, CADATAL)

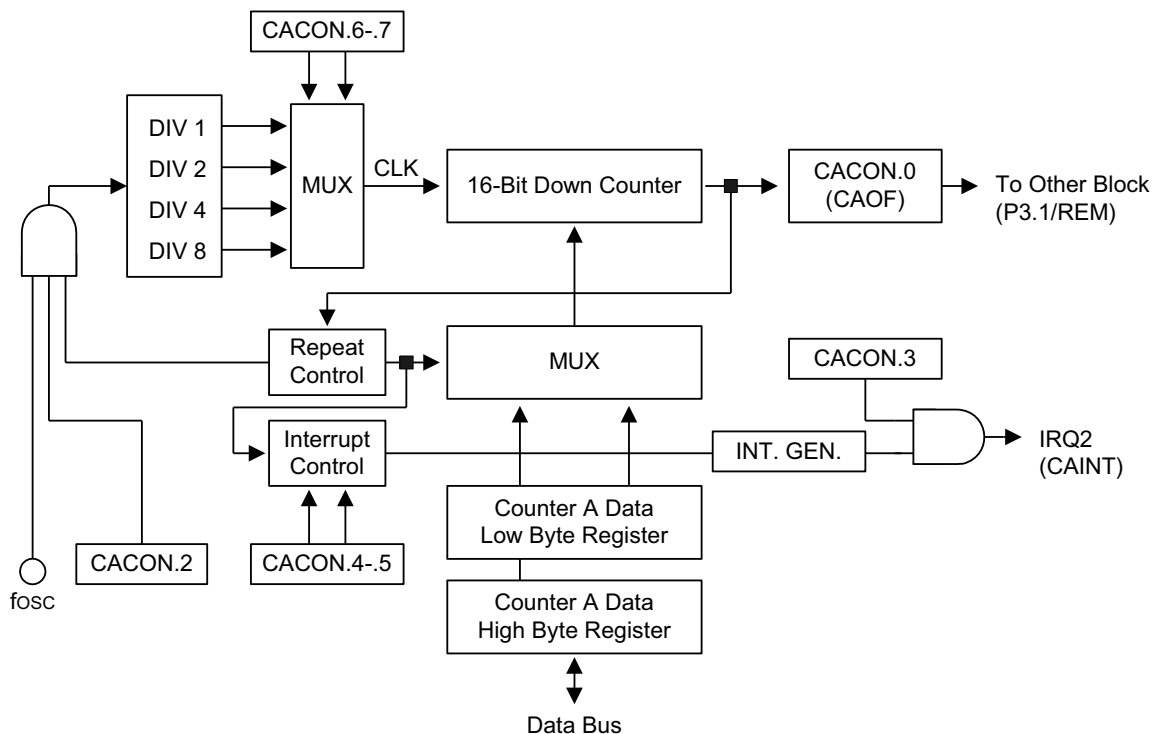


Figure 79. Counter A Block Diagram

► **Note:** The value of the CADATAL Register is loaded into the 8-bit counter when the operation of the Counter A starts. If a borrow occurs, the value of this register is loaded into the 8-bit counter. However, if the next borrow occurs, the value of the CADATAL Register is loaded into the 8-bit counter.

Counter A offers the following two functions:

- A normal interval timer, generating a Counter A interrupt (IRQ2, vector ECh) at programmed time intervals
- Supplies a clock source to the 16-bit timer/counter module, Timer 1, to generate Timer 1 overflow interrupts

► **Note:** The CPU clock should be faster than the Counter A clock.

13.1. Counter A Control Register

The Counter A Control (CACON) Register, described in Table 74, is located in F3h, Set1, Bank0, and is read/write-addressable.

Table 74. Counter A Control Register (CACON; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					F3h			

Note: R/W = read/write.

Bit	Description
[7:6]	Counter A Input Clock Selection Bits 00: f_{OSC} . 01: $f_{OSC}/2$. 10: $f_{OSC}/4$. 11: $f_{OSC}/8$.
[5:4]	Counter A Interrupt Timing Selection Bits 00: Elapsed time for Low data value. 01: Elapsed time for High data value. 10: Elapsed time for combined Low and High data values. 11: Reserved.
[3]	Counter A Interrupt Enable Bit 0: Disable interrupt. 1: Enable interrupt.
[2]	Counter A Start Bit 0: Stop Counter A. 1: Start Counter A.

Bit	Description (Continued)
[1]	Counter A Mode Selection Bit 0: One-Shot Mode. 1: Repeating Mode.
[0]	Counter A Output Flip-Flop Control Bit 0: Flip-Flop Low level (T–FF = Low). 1: Flip-Flop High level (T–FF = High).

CACON contains control settings for the following functions:

- Counter A clock source selection
- Counter A interrupt enable/disable
- Counter A interrupt pending control (i.e., read for status, write to clear)
- Counter A interrupt time selection

The Counter A Data registers (Set1, Bank0), CADATAH, CADATAL are described in Tables 75 and 76.

Table 75. Counter A Data High Byte Register

Bit	7	6	5	4	3	2	1	0
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F4h							

Note: R/W = read/write.

Table 76. Counter A Data Low Byte Register

Bit	7	6	5	4	3	2	1	0
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F5h							

Note: R/W = read/write.

13.1.1. Counter A Pulse Width Calculations

Figure 80 presents an example waveform consisting of a low period time (t_{LOW}) and a high period time (t_{HIGH}).

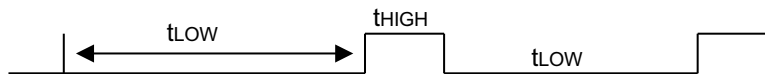


Figure 80. Counter A Pulse Width

To generate the waveform in Figure 80, observe the following calculations.

When CAOF = 0:

$$t_{LOW} = (CADATAL + 2) \times 1/F_x. \quad 0h < CADATAL < 100h, \text{ in which } F_x = \text{the selected clock}$$

$$t_{HIGH} = (CADATAH + 2) \times 1/F_x. \quad 0h < CADATAH < 100h, \text{ in which } F_x = \text{the selected clock}$$

When CAOF = 1:

$$t_{LOW} = (CADATAH + 2) \times 1/F_x. \quad 0h < CADATAH < 100h, \text{ in which } F_x = \text{the selected clock}$$

$$t_{HIGH} = (CADATAL + 2) \times 1/F_x. \quad 0h < CADATAL < 100h, \text{ in which } F_x = \text{the selected clock}$$

For t_{LOW} to result in 24 μ s and t_{HIGH} to result in 15 μ s, set $f_{OSC} = 4$ MHz and set $F_x = 4\text{MHz}/4 = 1$ MHz, as the following two methods show.

Method 1. When CAOF = 0:

$$t_{LOW} = 24 \mu\text{s} = (CADATAL + 2)/F_x = (CADATAL + 2) \times 1 \mu\text{s}, \quad CADATAL = 22$$

$$t_{HIGH} = 15 \mu\text{s} = (CADATAH + 2)/F_x = (CADATAH + 2) \times 1 \mu\text{s}, \quad CADATAH = 13$$

Method 2. When CAOF = 1:

$$t_{HIGH} = 15 \mu\text{s} = (CADATAL + 2)/F_x = (CADATAL + 2) \times 1 \mu\text{s}, \quad CADATAL = 13$$

$$t_{LOW} = 24 \mu\text{s} = (CADATAH + 2)/F_x = (CADATAH + 2) \times 1 \mu\text{s}, \quad CADATAH = 22$$

In Figures 81 and 82, and the code example that follows, Counter A is set to a repeat mode, the oscillation frequency is set as the Counter A clock source, and CADATAH and CADATAL are set to exhibit a 38kHz, 1/3 Duty carrier frequency. The program parameters are:

- Counter A is used in Repeat Mode
- Oscillation frequency is 4MHz (0.25 μ s)
- CADATAH = 8.795 μ s/0.25 μ s = 35.18, CADATAL = 17.59 μ s/0.25 μ s = 70.36
- Set P3.1 CMOS push-pull output and CAOF Mode
- 44-pin package

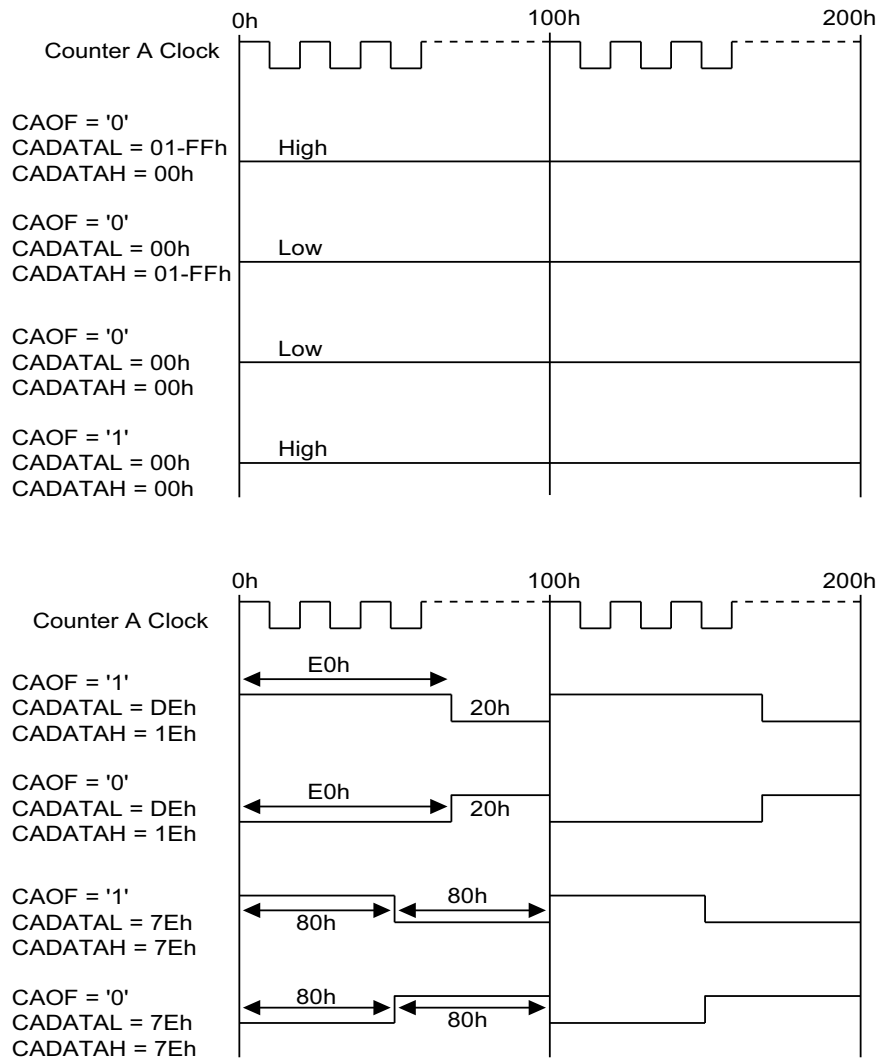


Figure 81. Counter A Output Flip-Flop Waveforms in Repeat Mode

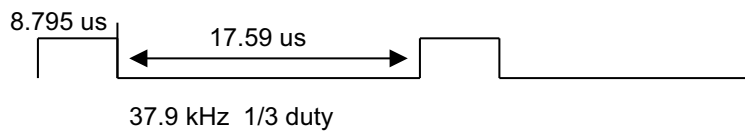


Figure 82. 38 kHz, 1/3 Duty Carrier Frequency

```

ORG          0100h          ; Reset address
START: DI
.
.
.
LD          CADATAL, #(70-2) ; Set 17.5ms
LD          CADATAH, #(35-2) ; Set 8.75ms
LD          P3CON, #11110010b ; Set P3 to CMOS push-pull
                                   ; output
                                   ; Set P3.1 to REM output
LD          CACON, #00000110b ; Clock Source → fOSC
                                   ; Disable Counter A
                                   ; interrupt.
                                   ; Select Repeat Mode for
                                   ; Counter A
                                   ; Start Counter A
                                   ; operation.
                                   ; Set Counter A output
                                   ; Flip-Flop (CAOF) high.
LD          P3, #80h          ; Set P3.7 (Carrier On/
                                   ; Off) to high.
                                   ; This command generates
                                   ; 38kHz, 1/3 duty pulse
                                   ; signal through P3.1

```

In this next example (see Figure 83), Counter A is set to One-Shot Mode, the oscillation frequency is set as the Counter A clock source, and CADATAH and CADATAL are set to exhibit a 40 μs-width pulse. The program parameters are:

- Counter A is used in One-Shot Mode
- Oscillation frequency is 4MHz (1 clock = 0.25 μs)
- CADATAH = 40 μs/0.25 μs = 160, CADATAL = 1
- Set P3.1 CMOS push-pull output and CAOF Mode
- 44-pin package

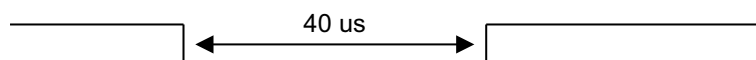


Figure 83. One-Shot Mode

```

ORG          0100h          ; Reset address

```

```

START: DI
.
.
.
    LD        CADATAL, #(160-2)    ; Set 40ms
    LD        CADATAH, #1          ; Set any value except 00h
    LD        P3CON, #11110010b    ; Set P3 to CMOS push-pull
                                        ; output
                                        ; Set P3.1 to REM output
    LD        CACON, #00000110b    ; Clock Source → fOSC
                                        ; Disable Counter A
                                        ; interrupt.
                                        ; Select One-Shot Mode for
                                        ; Counter A
                                        ; Stop Counter A
                                        ; operation.
                                        ; Set Counter A Output
                                        ; Flip-Flop (CAOF) high.
    LD        P3, #80h             ; Set P3.7 (Carrier On/
                                        ; Off) to high.
.
.
.
Pulse_out:
    LD        CACON, #00000101b    ; Start Counter A
                                        ; operation to make the
                                        ; pulse at this point.
                                        ; After the instruction is
                                        ; executed, 0.75ms is
                                        ; required before the
                                        ; falling edge of the
                                        ; pulse starts.

```

Chapter 14. Timer 2

The S3F80PB microcontroller's 16-bit timer/counter, Timer 2 (T2) can be used in universal remote controller applications to generate an envelope pattern for the remote controller signal. Timer 2 features the following components:

- One control register, T2CON (E8h, Set1, Bank1, R/W)
- Two 8-bit counter registers, T2CNTH and T2CNTL (E4h and E5h, Set1, Bank1, read-only)
- Two 8-bit reference data registers, T2DATAH and T2DATAL (E6h and E7h, Set1, Bank1, R/W)
- One 16-bit comparator

Figure 84 presents the Timer 2 block diagram.

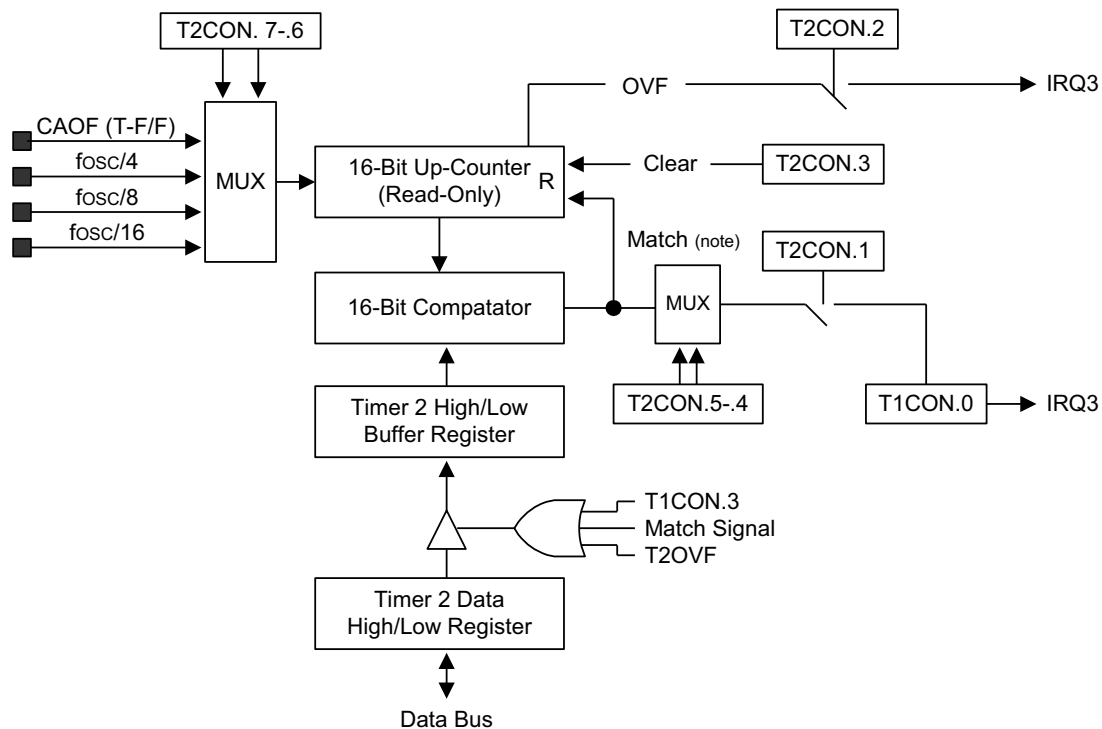


Figure 84. Timer 2 Block Diagram

► **Note:** The Match signal only occurs in Interval Mode.

Select one of the following clock sources as the Timer 2 clock:

- Oscillator frequency (f_{OSC}) divided by 4, 8, or 16
- Internal clock input from the Counter A module (Counter A flip/flop output)

Use Timer 2 in three ways:

- As a normal free run counter, generating a Timer 2 overflow interrupt (IRQ3, vector F0h) at programmed time intervals
- To generate a Timer 2 match interrupt (IRQ3, vector F2h) when the 16-bit Timer 2 count value matches the 16-bit value written to the reference data registers
- To generate a Timer 2 capture interrupt (IRQ3, vector F2h) when a triggering condition exists at the P3.2 pin for the 44-pin package and at the P3.0 pin for the 32-pin package (i.e., select a rising edge, a falling edge, or both edges as the trigger)

► **Note:** The CPU clock should be faster than the Timer 2 clock.

14.1. Timer 2 Control Register

The Timer 2 Control (T2CON) Register, shown in Table 77, is located in address E8h, Bank1, Set1 and is read/write-addressable.

Table 77. Timer 2 Control Register (T2CON; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					E8h			

Note: R/W = read/write.

Bit	Description
[7:6]	Timer 2 Input Clock Selection Bits 00: $f_{OSC} / 4$. 01: $f_{OSC} / 8$. 10: $f_{OSC} / 16$. 11: Internal clock. Counter A Flip-Flop, T–FF.
[5:4]	Timer 2 Operating Mode Selection Bits 00: Interval Timer Mode. Counter cleared by match signal. 01: Capture Mode. Rising edges, counter running, OVF can occur. 10: Capture Mode. Falling edges, counter running, OVF can occur. 11: Capture Mode. Rising and falling edges, counter running, OVF can occur.
[3]	Timer 2 Counter Clear Bit 0: No effect when write. 1: Clear T2 counter, T2CNT when write.
[2]	Timer 2 Overflow Interrupt Enable Bit* 0: Disable T2 overflow interrupt. 1: Enable T2 overflow interrupt.
[1]	Timer 2 Match/Capture Interrupt Enable Bit 0: Disable T2 match/capture interrupt. 1: Enable T2 match/capture interrupt.
[0]	Timer 2 Match/Capture Interrupt Pending Flag Bit 0: No T2 match/capture interrupt pending when read. 0: Clear T2 match/capture interrupt pending condition when write. 1: T2 match/capture interrupt is pending when read. 1: No effect when write.

Note: *A Timer 2 overflow interrupt pending condition is automatically cleared by hardware. However, the Timer 2 match/ capture interrupt, IRQ3, vector F2h, must be cleared by the interrupt service routine (software).

T2CON contains control settings for the following T2 functions:

- Timer 2 input clock selection
- Timer 2 operating mode selection
- Timer 2 16-bit downcounter clear
- Timer 2 overflow interrupt enable/disable
- Timer 2 match or capture interrupt enable/disable
- Timer 2 interrupt pending control (read for status, write to clear)

A reset operation clears T2CON to 00h, selects f_{OSC} divided by 4 as the T2 clock, configures Timer 2 as a normal interval timer, and disables the Timer 2 interrupts.

In the S3F80PB MCU's interrupt structure, the Timer 2 overflow interrupt receives higher priority than the Timer 2 match or capture interrupts; see Tables 78 through 81.

Table 78. Timer 2 Counter High Byte Register

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	E4h							

Note: R = read only.

Table 79. Timer 2 Counter Low Byte Register

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	E5h							

Note: R = read only.

Table 80. Timer 2 Data High Byte Register

Bit	7	6	5	4	3	2	1	0
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	E6h							

Note: R/W = read/write.

Table 81. Timer 2 Data Low Byte Register

Bit	7	6	5	4	3	2	1	0
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	E7h							

Note: R/W = read/write.

14.2. Timer 2 Overflow Interrupt

Timer 2 can be programmed to generate an overflow interrupt (IRQ3, F0h) whenever an overflow occurs in the 16-bit upcounter. When the Timer 2 overflow interrupt enable bit, T2CON.2, is set to 1, an overflow interrupt is generated each time the 16-bit upcounter reaches FFFFh. After the interrupt request is generated, the counter value is automatically cleared to 00h and up counting resumes. By writing a 1 to T2CON.3, the 16-bit counter value can be cleared/reset at any time during program operation.

14.3. Timer 2 Capture Interrupt

Timer 2 can be used to generate a capture interrupt (IRQ3, vector F2h) whenever a triggering condition is detected at the P3.0 pin for the 32-pin package and at the P3.3 pin for the 44-pin package. The T2CON.5 and T2CON.4 bit-pair setting is used to select the trigger condition for Capture Mode operation: rising edges, falling edges, or both signal edges.

In Capture Mode, program software can poll the Timer 2 match/capture interrupt pending bit, T2CON.0, to detect when a Timer 2 capture interrupt pending condition exists (T2CON.0 = 1). When the interrupt request is acknowledged by the CPU and the service routine starts, the interrupt service routine for vector F2h must clear the interrupt pending condition by writing a 0 to T2CON.0.

Figure 85 illustrates Timer 2 Capture Mode.

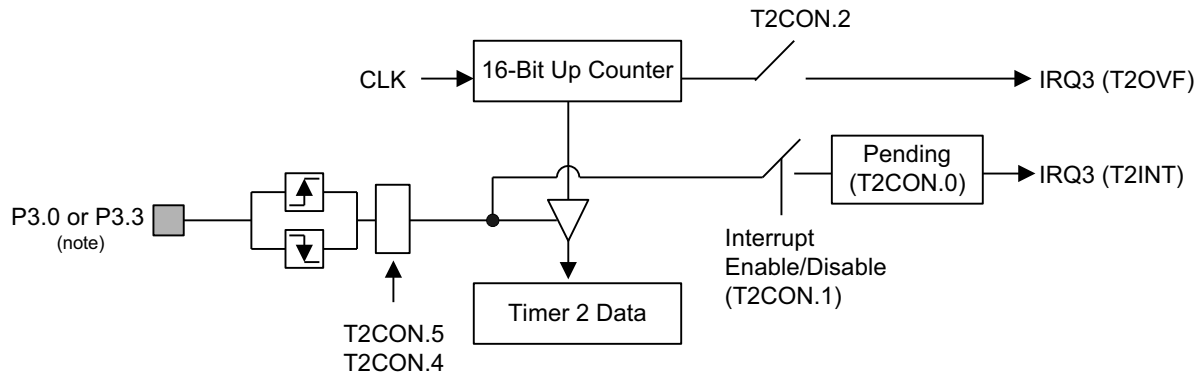


Figure 85. Simplified Timer 2 Function Diagram: Capture Mode

► **Note:** P3.0 is assigned as the T2CAP function for the 32-pin package, and P3.3 is assigned as the T2CAP function for the 44-pin package.

14.4. Timer 2 Match Interrupt

In addition to generating capture interrupts, Timer 2 can also be used to generate a match interrupt (IRQ3, vector F2h) whenever the 16-bit counter value matches the value that is written to the Timer 2 reference data registers, T2DATAH and T2DATAH. When a match condition is detected by the 16-bit comparator, the match interrupt is generated, the counter value is cleared, and up counting resumes from 00h.

In Match Mode, program software can poll the Timer 2 match/capture interrupt pending bit, T2CON.0, to detect when a Timer 2 match interrupt pending condition exists (T2CON.0 = 1). When the interrupt request is acknowledged by the CPU and the service routine starts, the interrupt service routine for vector F2h must clear the interrupt pending condition by writing a 0 to T2CON.0.

Figure 86 illustrates the Timer 2 Interval Timer Mode.

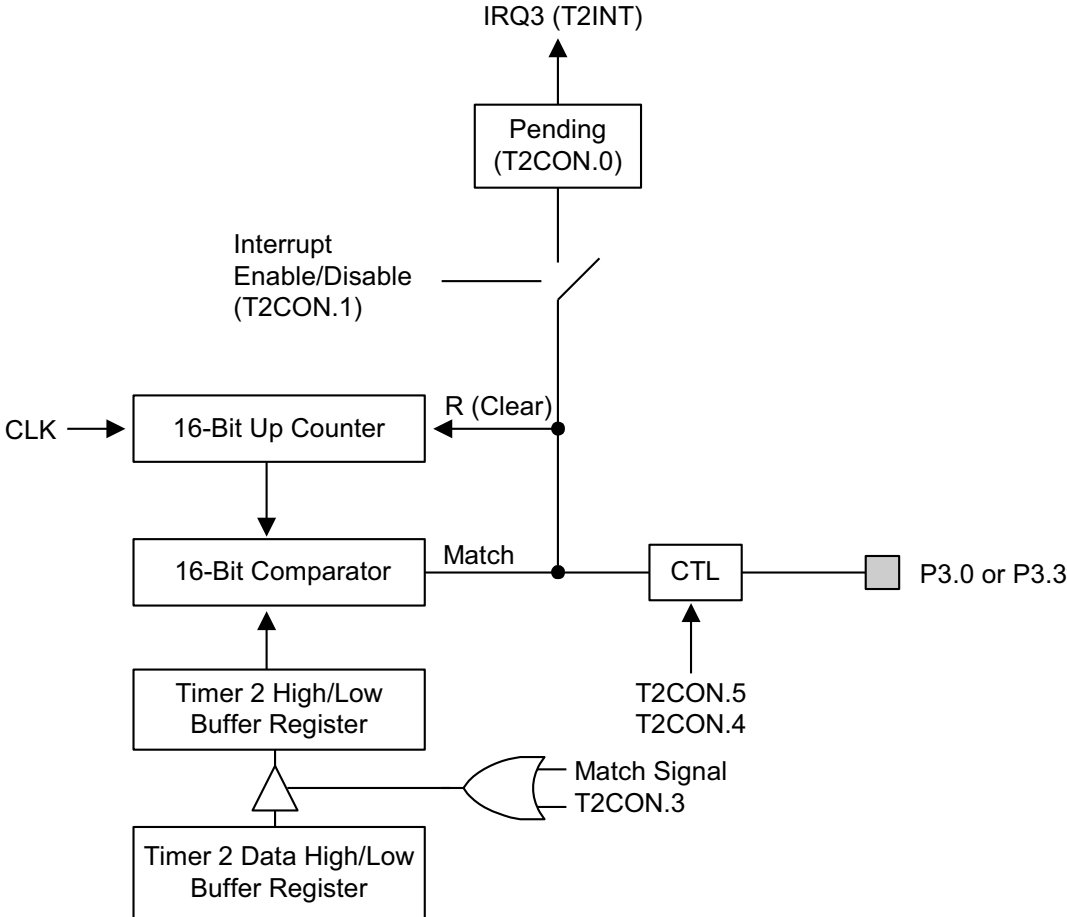


Figure 86. Simplified Timer 2 Function Diagram: Interval Timer Mode

Chapter 15. Embedded Flash Memory Interface

The S3F80PB MCU features an on-chip internal Flash memory instead of a masked ROM; this memory is accessible by an LDC instruction. The Flash sector is erasable and byte-programmable; data can be programmed in a Flash memory area at any time. This embedded Flash memory offers the following two operating features:

- User Program Mode
- Tool Program Mode

► **Note:** To learn more about Tool Program Mode; see the [Flash Programming](#) chapter on page 313.

15.1. Flash ROM Configuration

S3F80PB MCU Flash memory consists of 504 sectors. Each sector consists of 128 bytes. Therefore, the total size of Flash memory is 504×128 bytes (63 KB). User can erase Flash memory by a sector unit at a time and write the data into Flash memory by a byte unit at a time.

- 63KB of internal Flash memory
- Sector size: 128 bytes
- 10 years data retention
- Fast programming time:
 - Sector erase: 4 ms (minimum)
 - Byte programming: 20 μ s (minimum)
- Byte-programmable
- User programmable by LDC instruction
- 128-byte sector erase available
- External serial programming support
- Endurance: 10,000 erase/program cycles (minimum)
- Expandable onboard program (OBP)

15.2. User Program Mode

User Program Mode supports sector erase, byte programming, byte read, and one protection mode, Hard Lock protection. The S3F80PB MCU also features an internal pumping circuit to generate high voltage; therefore, 12.5 V into a V_{PP} (i.e., test) pin is not required. To program Flash memory in this mode, several control registers are used. There are four functions: programming, reading, sector erase, and hard lock protection.

15.3. ISP On-Board Programming Sector

ISP sectors located in program memory area can store on-board program software (i.e., boot program code for upgrading application code by interfacing with an 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 Set0, i.e., enable ISP at the Smart Option. If you prefer not to use the ISP sector method, this area can be used as normal program memory (i.e., can be erased or programmed by the LDC instruction) by setting the ISP disable bit (1) with the Smart Option. Even if the ISP sector is selected, the ISP sector can be erased or programmed in Tool Program Mode using a serial programming tool.

The size of the ISP sector can be adjusted using Smart Option settings; to learn more, see [Figure 13](#) on page 20. The Program Memory Address Space is illustrated in Figure 87.

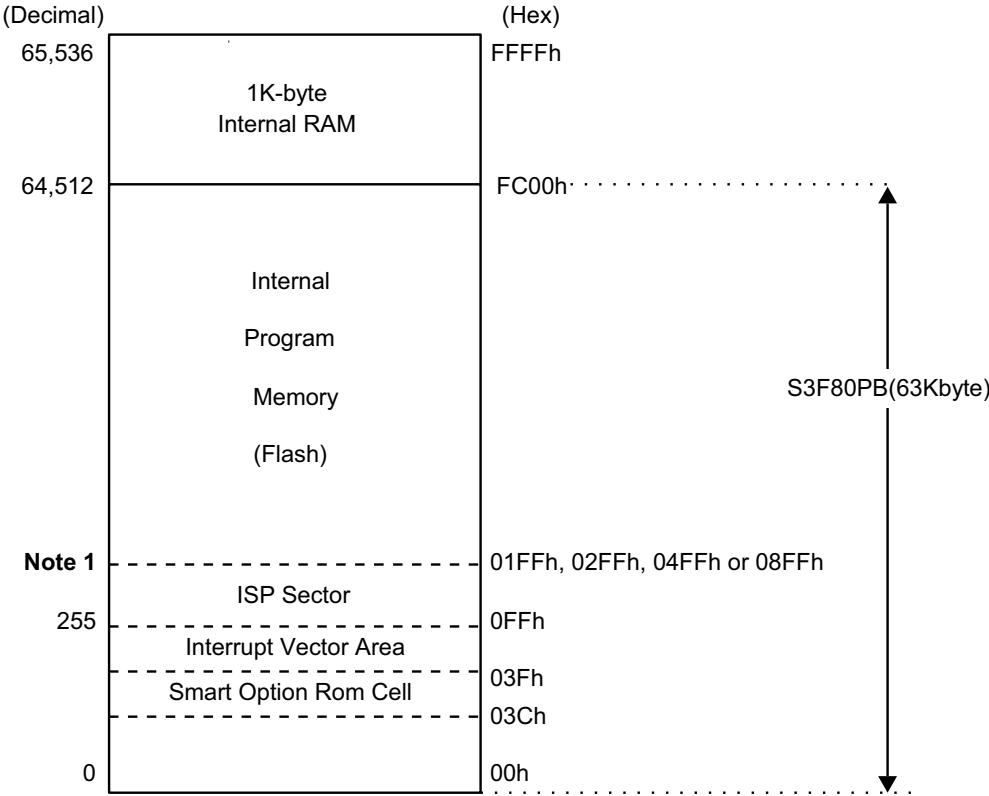


Figure 87. Program Memory Address Space

-
- **Notes:**
1. The size of the ISP sector can be varied by using the Smart Option. According to the Smart Option setting related to the ISP, the ISP reset vector address can be changed to one of the following addresses: 200h, 300h, 500h, or 900h).
 2. The ISP sector can store on-board program software. To learn more, see the [Embedded Flash Memory Interface](#) chapter on page 277.
-

15.4. ISP Reset Vector and ISP Sector Size

When the ISP sectors are being used by setting the ISP enable/disable bit to 0 and the reset vector selection bit to 0 at the Smart Option (see the [Embedded Flash Memory Interface](#) chapter on page 277), choose the reset vector address of the CPU as shown in Table 82 by setting the ISP reset vector address selection bits; also see the list of ISP sector sizes in Table 83.

Table 82. Reset Vector Address

Smart Option (003Eh) ISP Reset Vector Address Selection Bit			Reset Vector Address after POR	Usable Area for	
Bit 7	Bit 6	Bit 5		ISP Sector	ISP Sector Size
1	x	x	0100h	0	0
0	0	0	0200h	100h–1FFh	256 bytes
0	0	1	0300h	100h–2FFh	512 bytes
0	1	0	0500h	100h–4FFh	1024 bytes
0	1	1	0900h	100h–8FFh	2048 bytes

Note: The selection of the ISP reset vector address by Smart Option (003Eh.7–003Eh.5) is not dependent of the selection of ISP sector size by Smart Option (003Eh.2–003Eh.0).

Table 83. 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 bytes)	256 bytes
0	0	1	100h–2FFh (512 bytes)	512 bytes
0	1	0	100h–4FFh (1024 bytes)	1024 bytes
0	1	1	100h–8FFh (2048 bytes)	2048 bytes

Note: The area of the ISP sector selected by Smart Option bits (3Eh.2–3Eh.0) cannot be erased and programmed by LDC instruction in User Program Mode.

15.5. Flash Memory Control Registers

The Flash Memory Control (FMCON) Register, shown in Table 84, is available only in User Program Mode to select the mode of Flash memory operations; i.e., sector erase and byte programming, and to provide Hard Lock Protection for Flash memory.

Table 84. Flash Memory Control Register (FMCON; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	–	–	–	0
R/W	R/W	R/W	R/W	R/W	–	–	–	R/W
Address	EFh							

Note: R/W = read/write.

Bit	Description
[7:4]	Flash Memory Mode Selection Bits 0001–0100: Reserved. 0101: Programming Mode. 0110: Hard Lock Mode. 0111–1001: Reserved. 1010: Erase Mode. 1011–1111: Reserved.
[3:1]	Reserved These bits are reserved and must be set to 000.
[0]	Flash Operation Start Bit (Available for Erase and Hard Lock) 0: Operation stop. 1: Operation start (automatic clear bit).

Note: Hard Lock Mode is one of the Flash protection modes. To learn more about Hard Lock Mode, see the [Hard Lock Protection](#) section on page 293.

Bit 0 of the FMCON Register, FMCON.0, initiates the Erase and Hard Lock Protection operations. As a result, operation of Erase and Hard Lock Protection is activated when setting FMCON.0 to 1. If FMCON.0 is set to 1 for erasing, the CPU is stopped automatically to allow for an erasure period (minimum 10ms). After this period, the CPU is automatically restarted. When reading or programming a byte of data from or into Flash memory, this bit is not required to be manipulated.

15.5.1. Flash Memory User Programming Enable Register

The Flash Memory User Programming Enable (FMUSR) Register, shown in Table 85, manages the safe operation of Flash memory. This register will protect an undesired erase or program operation from CPU malfunctions caused by electrical noise. After reset, User

Program Mode is disabled because the value of FMUSR is 00000000b as a result of the reset operation. If it is necessary to operate Flash memory, enable User Program Mode by setting the value of FMUSR to 10100101b. Any value written to FMUSR other than 10100101b disables User Program Mode.

Table 85. Flash Memory User Programming Enable Register (FMUSR; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					EEh			

Note: R/W = read/write.

Bit	Description
[7:0]	Flash Memory User Programming Enable Bits 10100101: Enable User Program Mode. Other values: Disable User Program Mode.

Notes:

1. To enable Flash memory user programming, write 10100101b to FMUSR.
2. To disable Flash memory operation, write other value except 10100101b into FMUSR.

15.5.2. Flash Memory Sector Address Registers

There are two sector address registers for erasing or programming Flash memory. The Flash Memory Sector Address Low Byte (FMSECL) Register indicates the low byte of the sector address, and the Flash Memory Address Sector High Byte (FMSECH) Register indicates the high byte of the sector address. The S3F80PB MCU requires the FMSECH Register because it contains 512 sectors.

Each Flash memory sector consists of 128 bytes. Each sector's address starts at xx00h or xx80h; for example, the base address of the sector is xx00h or xx80h. Therefore, bits 6–0 of FMSECL do not effect whether the value is 1 or 0. Zilog recommends using this method as the simplest way to load the sector base address into the FMSECH and FMSECL registers. When programming Flash memory, user should program after loading a sector base address, which is located in the destination address to write data into the FMSECH and FMSECL registers. If the next operation is also to write one byte data, user should check whether next destination address is located in the same sector or not. If using other sectors, user should load sector address to FMSECH and FMSECL Register according to the sector. To learn more, see the three examples that begin [on page 286](#).

The contents of these Flash Memory Sector Address registers are described in Tables 86 and 87.

Table 86. Flash Memory Sector Address High Byte Register (FMSECH; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					ECh			

Note: R/W = read/write.

Bit	Description
[7:0]	Flash Memory Sector Address High Byte

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

Table 87. Flash Memory Sector Address Low Byte Register (FMSECL; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W					R/W			
Address					EDh			

Note: R/W = read/write.

Bit	Description
[7:0]	Flash Memory Sector Address Low Byte

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

15.6. Sector Erase Operations

Flash memory can be partially erased by using the sector erase functions in User Program Mode only. Sectors are the only units of Flash memory that can be erased in User Program Mode.

Program memory on the S3F80PB MCU is divided into 256 sectors (32KB). Every sector contains all 128 bytes and should first be erased prior to programming a new data byte into Flash memory. A minimum of 10ms delay time is required prior to an erase and after setting the sector address and triggering the erase start bit (FMCON.0). Sector erase is not supported in Tool Program modes (i.e., when using MDS mode or programming tools).

Figure 88 portrays how sectors are mapped in User Program Mode.

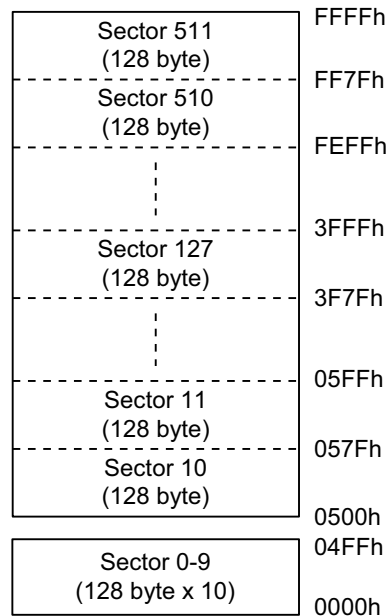


Figure 88. Sector Configurations in User Program Mode

15.6.1. The Sector Erase Procedure in User Program Mode

Observe the following procedure to perform a sector erase in User Program Mode.

1. Set the Flash Memory User Programming Enable (FMUSR) Register to 10100101b.
2. Set the Flash Memory Sector Address (FMSECH and FMSECL) registers.
3. Set the Flash Memory Control (FMCON) Register to 10100001b.
4. Set the Flash Memory User Programming Enable (FMUSR) Register to 00000000b.

Figure 89 shows the flow of the sector erase routine in User Program Mode.

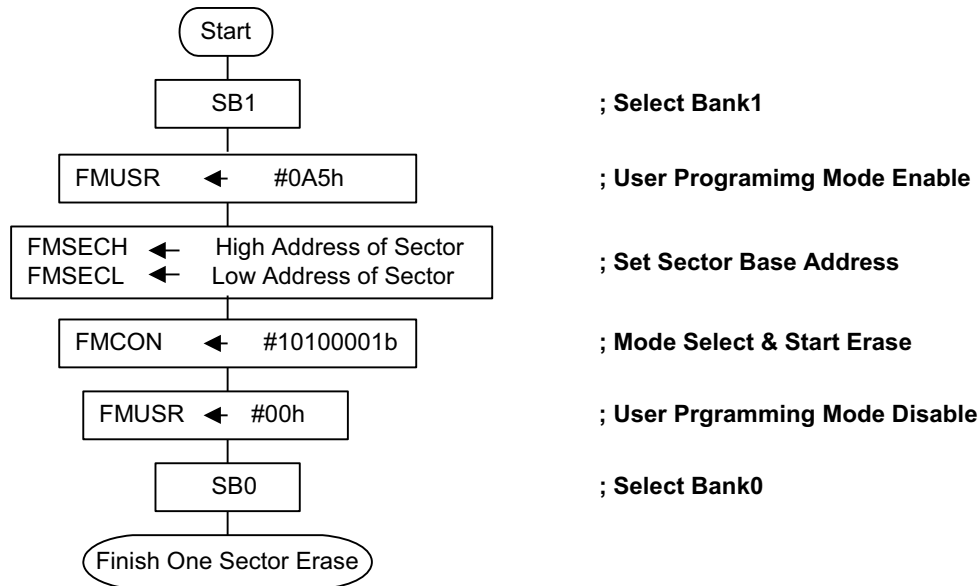


Figure 89. Sector Erase Routine in User Program Mode

-
- **Notes:**
1. If the user erases a sector selected by the Flash Memory Sector Address (FMSECH and FMSECL) registers, FMUSR should be enabled just before starting a sector erase operation. To erase a sector, the Flash Operation start bit of the FMCON Register is written from Operation Stop 0 to Operation Start 1. This bit will be cleared automatically just after the corresponding operation is completed. In other words, when the S3F80PB MCU is in a condition such that its Flash memory user programming enable bits are enabled and it executes the start operation of a sector erase, it will achieve the result of erasing the selected sector per the user's purposes, and the Flash operation start bit of the FMCON Register will clear automatically.
 2. If the user executes a sector erase operation with FMUSR disabled, then the Flash operation start bit (FMCON.0) remains High, signalling a start operation; this bit is not cleared when the next instruction is executed. Therefore, the user should be careful to set FMUSR when executing a sector erase to not affect other Flash sectors.
-

The following code example shows how to erase one sector.

```

ERASE_ONESECTOR:
    SB1
    LD    FMUSR, #0A5h           ; User Program Mode enable
    LD    FMSECH, #40h          ; Set sector address
                                   ; 4000h, sector 128
    LD    FMSECL, #00h          ; Among sector 0 to 511
    LD    FMCON, #10100001b     ; Select Erase Mode Enable
                                   ; & Start sector erase

ERASE_STOP:
    LD    FMUSR, #00h           ; User Program Mode
                                   ; disable
    SB0

```

This next code example shows how to erase the contents of a Flash memory space that ranges from sector (n) to sector (n + m).

```

;; Predefine the number of sector to erase

    LD    SecNumH, #00h         ; Set sector number
    LD    SecNumL, #128        ; Selection the sector128
                                   ; (base address 4000h)
    LD    R6, #01h             ; Set the sector range (m)
                                   ; to erase
    LD    R7, #7Dh             ; Into High-byte (R6) and
                                   ; Low-byte (R7)
    LD    R2, SecNumH
    LD    R3, SecNumL

ERASE_LOOP: CALL SECTOR_ERASE
    XOR    P4, #11111111b      ; Display ERASE_LOOP cycle
    INCW  RR2
    LD    SecNumH, R2
    LD    SecNumL, R3
    DECW  RR6
    LD    R8, R6
    OR    R8, R7
    CP    R8, #00h
    JP    NZ, ERASE_LOOP

SECTOR_ERASE:
    LD    R12, SecNumH
    LD    R14, SecNumL
    MULT  RR12, #80h           ; Calculation the base

```

```

; address of a target
; sector
MULT RR14, #80h ; The size of one sector
; is 128 bytes
ADD R13, R14 ; BTJRF FLAGS.7, NOCARRY
; INC R12

NOCARRY:
LD R10, R13
LD R11, R15
ERASE_START:
SB1
LD FMUSR, #0A5h ; User program mode enable
LD FMSECH, R10 ; Set sector address
LD FMSECL, R11
LD FMCON, #10100001B ; Select erase mode enable
; and start sector erase

ERASE_STOP:
LD FMUSR, #00h ; User Program Mode
; disable

SB0
RET

```

15.7. Program Operations

After a sector erase, Flash memory is programmed in one-byte units. Programming write operations begin with the LDC instruction.

15.7.0.1. Programming in User Program Mode

Observe the following procedure to program Flash memory in User Program Mode.

1. Erase all target sectors before programming.
2. Set the Flash Memory User Programming Enable (FMUSR) Register to 10100101b.
3. Set the Flash Memory Control (FMCON) Register to 0101000Xb.
4. Set the Flash Memory Sector Address (FMSECH and FMSECL) registers to the sector base address of the destination address to write data.
5. Load transmission data into a working register.
6. Load the upper addresses of Flash memory into the upper register of the working register pair.
7. Load the lower addresses of Flash memory into the lower register of the working register pair.

8. Load transmission data to a Flash memory location on the LDC instruction through Indirect Addressing Mode.
9. Set the Flash Memory User Programming Enable (FMUSR) Register to 00000000b.

► **Note:** In Programming Mode, FMCON.0's value can be either 0 or 1.

Figure 90 shows the flow of the byte programming routine in User Program Mode.

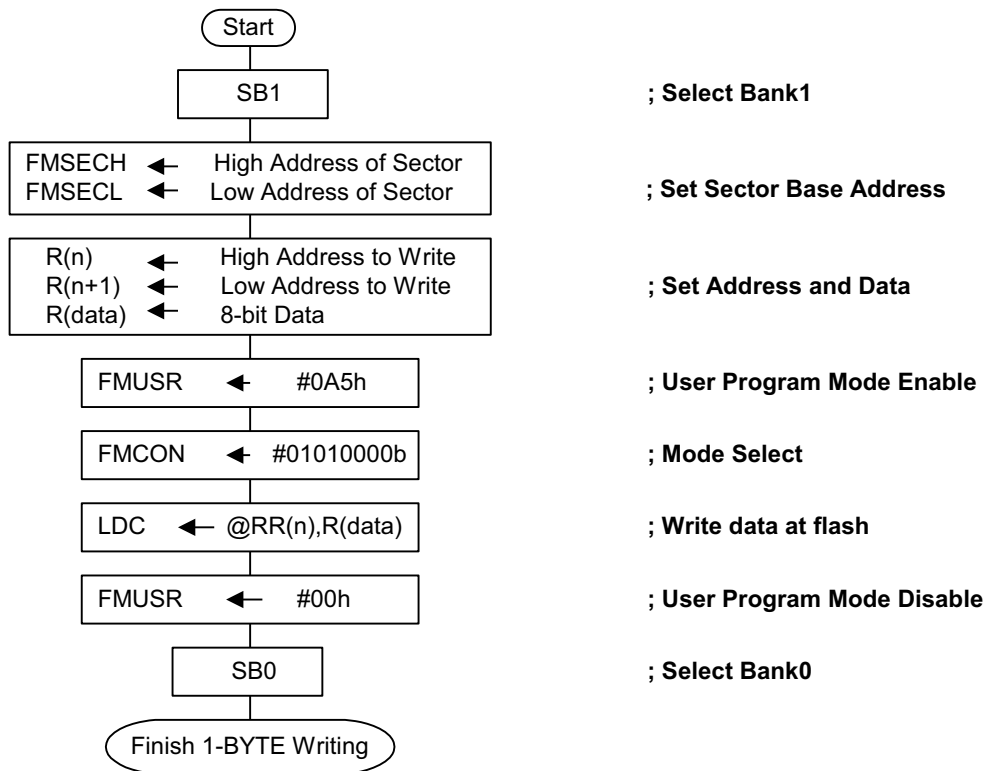


Figure 90. Byte Program Flowchart in User Program Mode

Figure 91 shows the overall program flow in User Program Mode.

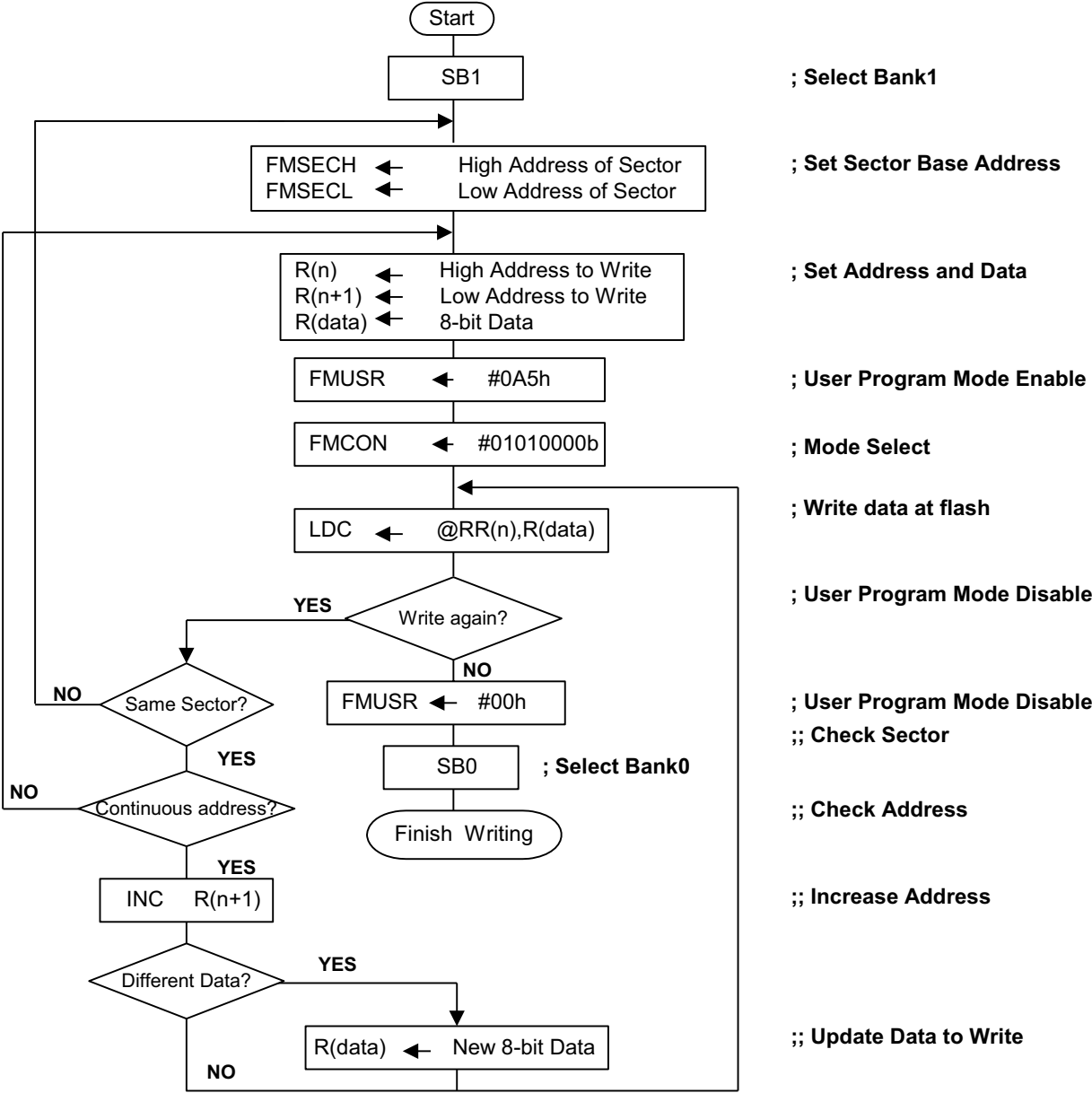


Figure 91. Program Flowchart in User Program Mode

The following example shows how to perform one-byte Flash programming.

```

WR_BYTE:           ; Write data AAh to destination address 4010h
    SB1
    LD      FMUSR, #0A5h           ; User Program Mode enable
    LD      FMCON, #01010000b     ; Selection programming
                                           ; mode
    LD      FMSECH, #40h          ; Set the base address of
                                           ; sector (4000h)
    LD      FMSECL, #00h
    LD      R9, #0AAh             ; Load data AA to write
    LD      R10, #40h             ; Load the upper address
                                           ; of Flash memory into the
                                           ; upper register of the
                                           ; working register pair
    LD      R11, #10h            ; Load Flash memory lower
                                           ; address into lower
                                           ; register of pair working
                                           ; register
    LDC     @RR10, R9             ; Write data AAh at Flash
                                           ; memory location (4010h)
    LD      FMUSR, #00h           ; User Program Mode
                                           ; disable
    SB0
  
```

This next example shows how to program the same Flash memory sector.

```

WR_INSECTOR:      ; RR10 Address copy (R10-
                                           ; high address, R11-low
                                           ; address)
    LD      R0, #40h
    SB1
    LD      FMUSR, #0A5h           ; User Program Mode enable
    LD      FMCON, #01010000b     ; Selection programming
                                           ; mode and Start
                                           ; programming
    LD      FMSECH, #40h          ; Set the base address of
                                           ; sector located in target
                                           ; address to write data
    LD      FMSECL, #00h          ; The sector 128's base
                                           ; address is 4000h
    LD      R9, #33h             ; Load data 33h to write
    LD      R10, #40h            ; Load Flash memory upper
                                           ; address into upper
                                           ; register of pair working
                                           ; register
    LD      R11, #40h            ; Load Flash memory lower
  
```

```

; address into lower
; register of pair working
; register
WR_BYTE:
    LDC        @RR10, R9        ; Write data 33h at Flash
                                ; memory location
    INC        R11              ; Reset address in the
                                ; same sector by INC
                                ; instruction
    DJNZ       R0, WR_BYTE      ; Check whether the end
                                ; address for programming
                                ; reach 407Fh or not
    LD         FMUSR, #00h      ; User Program Mode
                                ; disable
    SB0

```

A third example shows how to program the Flash memory spaces located in other sectors.

```

WR_INSECTOR2:
    LD         R0, #40h
    LD         R1, #40h
    SB1
    LD         FMUSR, #0A5h      ; User Program Mode enable
    LD         FMCON, #01010000b ; Selection programming
                                ; mode and Start
                                ; programming
    LD         FMSECH, #01h     ; Set the base address of
                                ; sector located in target
                                ; address to write data
    LD         FMSECL, #00h     ; The sector 2's base
                                ; address is 100h
    LD         R9, #0CCh        ; Load data CCh to write
    LD         R10, #01h        ; Load Flash memory upper
                                ; address into upper
                                ; register of pair working
                                ; register
    LD         R11, #40h        ; Load Flash memory lower
                                ; address into lower
                                ; register of pair working
                                ; register

    CALL      WR_BYTE
    LD         R0, #40h

WR_INSECTOR50:
    LD         FMSECH, #19h     ; Set the base address of

```

```

; sector located in target
; address to write data
LD      FMSECL, #00h      ; The sector 50's base
; address is 1900h
LD      R9, #55h         ; Load data 55h to write
LD      R10, #19h        ; Load Flash memory upper
; address into upper
; register of pair working
; register
LD      R11, #40h        ; Load Flash memory lower
; address into lower
; register of pair working
; register
CALL    WR_BYTE

WR_INSECTOR128:
LD      FMSECH, #40h     ; Set the base address of
; sector located in target
; address to write data
LD      FMSECL, #00h     ; The sector 128's base
; address is 4000h
LD      R9, #0A3h        ; Load data A3h to write
LD      R10, #40h        ; Load Flash memory upper
; address into upper
; register of pair working
; register
LD      R11, #40h        ; Load Flash memory lower
; address into lower
; register of pair working
; register

WR_BYTE1:
LDC     @RR10, R9        ; Write data A3h at Flash
; memory location
INC     R11
DJNZ   R1, WR_BYTE1

LD      FMUSR, #00h      ; User Program mode
; disable
SB0

WR_BYTE:
LDC     @RR10, R9        ; Write data written by R9
; at Flash memory location
INC     R11
DJNZ   R0, WR_BYTE
RET

```

15.8. Read Operations

The read operation is initiated by the LDC instruction. Observe the following procedure to program read operations in User Program Mode.

1. Load the upper Flash memory addresses into the upper register of the working register pair.
2. Load the lower Flash memory addresses into the lower register of the working register pair.
3. Load received data from Flash memory location area on the LDC instruction through Indirect Addressing Mode.

The following example shows how to perform read programming.

```
LD          R2, #03h          ; Load Flash memory's
                              ; upper address to upper
                              ; register of
                              ; pair working register
LD          R3, #00h          ; Load Flash memory's
                              ; lower address to lower
                              ; register of pair working
                              ; register

Loop: LDC   R0, @RR2          ; Read data from Flash
                              ; memory location (between
                              ; 300h and 3FFh)

INC         R3
CP          R3, #0FFh
JP         NZ, LOOP
```

15.8.1. Hard Lock Protection

The Hard Lock Protection function prevents changes to data in Flash memory. Hard Lock Protection can be set by writing 0110b to FMCON7–4. If this function is enabled, the user cannot write or erase the data within Flash memory. This protection can be released by executing a chip erase in Tool Program Mode. The procedure to set a Hard Lock Protection in the User Program Mode is listed below. Hardware Protection could be supported by the serial tool writer manufacture. Consult the documentation for the particular manufacturer's serial program writer tool you use.

Observe the following procedure to set Hard Lock Protection in User Program Mode.

1. Set Flash Memory User Programming Enable (FMUSR) Register to 10100101b.
2. Set Flash Memory Control (FMCON) Register to 01100001b.

3. Set Flash Memory User Programming Enable (FMUSR) Register to 00000000b.

The following example shows how to set Hard Lock Protection.

```
SB1
LD      FMUSR, #0A5h      ; User Program Mode enable
LD      FMCON, #01100001b ; Select Hard Lock Mode
                          ; and Start protection
LD      FMUSR, #00h      ; User Program Mode
                          ; disable
SB0
```

Chapter 16. Low Voltage Detection

The S3F80PB microcontroller features a built-in Low Voltage Detection (LVD) circuit that allows LVD and LVD_FLAG detection of power voltage.

- Operating Frequency: 1 to 8MHz
- Low voltage detect level for Backup Mode and Reset (LVD): 1.65 V (typ.) \pm 50mV
- Low voltage detect level for Flash Flag Bit (LVD_FLAG): 1.90, 2.00, 2.10, 2.20V (typ.) \pm 100mV

After power-on, LVD block is always enabled. LVD block is only disable when executed Stop instruction. The LVD block on the S3F80PB MCU consists of two comparators and a resistor string. One of comparators is for LVD detection, and the other is for LVD_FLAG detection. A block diagram of the LVD circuit is shown in Figure 92.

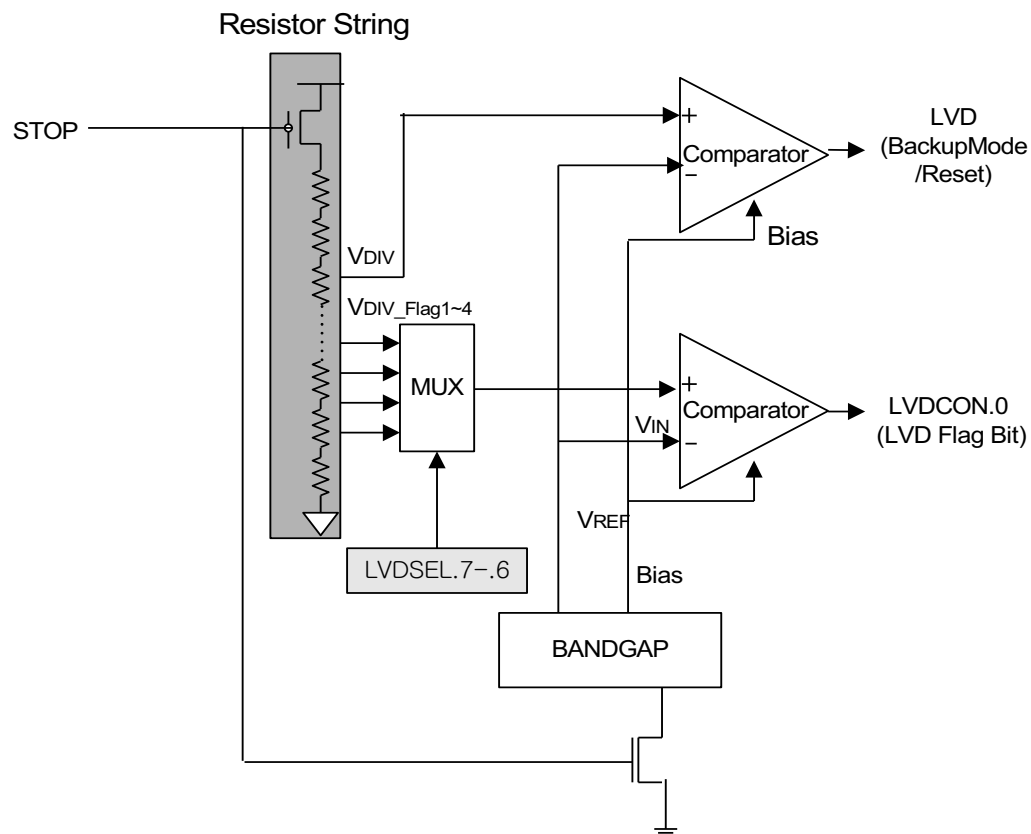


Figure 92. Low Voltage Detect Block Diagram

The LVD circuit supplies two operating modes by one comparator: Backup Mode input and system reset input. With the LVD circuit, the S3F80PB MCU can enter Backup Mode and generate a reset signal via LVD level detection. When the LVD circuit detects falling power levels, the S3F80PB MCU enters Backup Mode.

The Backup Mode input automatically causes a chip stop. When the LVD circuit detects rising power levels, a system reset occurs. When the reset pin is at a high state and the LVD circuit detects a rising edge of V_{DD} on the V_{LVD} point, the reset pulse generator causes a reset pulse, and a system reset occurs. This reset by the LVD circuit is one of the S3F80PB MCU's reset sources.

► **Note:** LVD is a parameter that can be defined as low-level voltage detection when entering or exiting Backup Mode.

16.1. LVD Flag

The LVD indicator flag, LVD_FLAG, is a parameter that represents low-level voltage detection. Output from another comparator will cause the LVD_FLAG bit to become a 1 or a 0. When the power voltage is below the LVD_FLAG level, bit 0 of the LVDCON Register is 1. When the power voltage is above the LVD_FLAG level, bit 0 of LVDCON Register is automatically set to 0. Essentially, when this flag is 0, a low voltage level is indicated. As a result, LVDCON.0 can be used as a flag bit to indicate a low battery in IR and other applications.

The LVD_GAPn voltage gaps (in which $n = 1$ to 4) between LVD and LVD_FLAGn (in which $n = 1$ to 4) feature an ± 80 mV distribution. LVD and LVD_FLAGn are not overlapped. See Tables 88 through 90.

Table 88. LVD Voltage Gap Characteristics

Symbol	Min.	Typ.	Max.	Unit
LVD_GAP1	170	250	330	mV
LVD_GAP2	270	350	430	mV
LVD_GAP3	370	450	530	mV
LVD_GAP4	470	550	630	mV

Table 89. LVD Flag Gap Characteristics

Symbol	Min.	Typ.	Max.	Unit
Gap between LVD_Flag1 and LVD_Flag2	50	100	150	mV
Gap between LVD_Flag2 and LVD_Flag3	50	100	150	mV
Gap between LVD_Flag3 and LVD_Flag4	50	100	150	mV

Table 90. LVD Enable Time ($T_A = -25^{\circ}\text{C}$ to $+85^{\circ}\text{C}$)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
LVD enable time	t_{LVD}	$V_{DD} = 1.4\text{V}$	–	–	50	μs

In Stop Mode, the LVD operation turns off. When an external interrupt occurs, the LVD operation requires t_{LVD} during $50\mu\text{s}$ (maximum) to wake up. If V_{DD} is below V_{LVD} after this external interrupt, the S3F80PB MCU enters Backup Mode. Because the t_{LVD} period is not enough to initiate oscillation, the S3F80PB MCU is not operating an abnormal state.

16.2. Low Voltage Detection Control Register

In the Low Voltage Detection Control (LVDCON) Register, the LVDCON.0 flag bit indicates a low battery status in IR applications, among others. When an LVD circuit detects the LVD_FLAG, the LVDCON.0 flag bit is automatically set. The reset value of LVDCON is #00h. The contents of the LVDCON Register are described in Table 91.

Table 91. LVD Control Register (LVDCON; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	–	–	–	–	–	–	–	0
R/W	–	–	–	–	–	–	–	R/W
Address	E0h							

Note: R/W = read/write.

Bit	Description
[7:1]	Reserved These bits are reserved and must be set to 0000000.
[0]	LVD Flag Indicator Bit 0: $V_{DD} \geq \text{LVD_FLAG Level}$. 1: $V_{DD} < \text{LVD_FLAG Level}$.

Note: When the LVD detects LVD_FLAG level, LVDCON.0 flag bit is set automatically. When V_{DD} is greater than or equal to zero, the LVDCON.0 flag bit is cleared automatically.

16.3. Low Voltage Detection Flag Selection Register

The Low Voltage Detection Flag Selection (LVDSEL) Register is used to select the LVD flag level. The reset value of LVDSEL is #00h.

The contents of the LVDCON Register are described in Table 92.

Table 92. LVD Flag Level Selection Register (LVDSEL; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	–	–	–	–	–	–
R/W	R/W	R/W	–	–	–	–	–	–
Address	F1h							

Note: R/W = read/write.

Bit	Description
[7:6]	LVD Flag Level Selection Bits 00: LVD_FLAG Level = 1.90V. 01: LVD_FLAG Level = 2.00V. 10: LVD_FLAG Level = 2.10V. 11: LVD_FLAG Level = 2.20V.
[5:0]	Reserved These bits are reserved and must be set to 000000.

Chapter 17. Electrical Characteristics

In this chapter, the S3F80PB MCU's electrical characteristics are presented in the following tables and charts.

- Absolute Maximum Ratings – see [Table 93](#) on page 300
- DC Electrical Characteristics – see [Table 94](#) on page 300
- Low Voltage Detect Circuit – see [Table 96](#) on page 302
- LVD Enable Time – see [Table 98](#) on page 302
- Power On Reset Circuit – see [Table 99](#) on page 303
- Data Retention Supply Voltage in Stop Mode – see [Table 100](#) on page 303
- Stop Mode to Normal Mode Timing Diagrams – see [Figure 93](#) on page 303 and [Figure 94](#) on page 304
- AC Electrical Characteristics – see [Table 101](#) on page 304
- Input Timing for External Interrupts – see [Figure 95](#) on page 304
- Input Timing for Reset – see [Figure 96](#) on page 305
- Oscillation Characteristics – see [Table 102](#) on page 305
- Input/Output Capacitance – see [Table 103](#) on page 306
- Oscillation Stabilization Time – see [Table 104](#) on page 306
- Operating Voltage Range – see [Figure 97](#) on page 307
- AC Electrical Characteristics for Internal Flash ROM – see [Table 105](#) on page 307
- ESD Characteristics – see [Table 106](#) on page 308

Table 93. Absolute Maximum Ratings ($T_A = 25^\circ\text{C}$)

Parameter	Symbol	Conditions	Rating TBD	Unit
Supply Voltage	V_{DD}	–	–0.3 to +3.8	V
Input Voltage	V_{IN}	–	–0.3 to $V_{DD} + 0.3$	
Output Voltage	V_O	All output pins	–0.3 to $V_{DD} + 0.3$	
Output Current High	I_{OH}	One I/O pin active	–18	mA
		All I/O pins active	–60	
Output Current Low	I_{OL}	One I/O pin active	+30	mA
		All I/O pins active	+150	
Operating Temperature	T_A	–	0 to +70	$^\circ\text{C}$
Storage Temperature	T_{STG}	–	–65 to +150	$^\circ\text{C}$

Table 94. DC Electrical Characteristics ($T_A = -25^\circ\text{C}$ to $+85^\circ\text{C}$, $V_{DD} = 1.60\text{V}$ to 3.6V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Operating Voltage	V_{DD}	$f_{OSC} = 4\text{MHz}, 8\text{MHz}$	1.60	–	3.6	V
Input High Voltage	V_{IH1}	All input pins except V_{IH2} and V_{IH3}	$0.8 V_{DD}$	–	V_{DD}	V
	V_{IH2}	nRESET	0.85V	–	V_{DD}	
	V_{IH3}	X_{IN}	$V_{DD} - 0.3$	–	V_{DD}	
Input Low Voltage	V_{IL1}	All input pins except V_{IL2} and V_{IL3}	0	–	$0.2 V_{DD}$	V
	V_{IL2}	nRESET	0	–	$0.2 V_{DD}$	
	V_{IL3}	X_{IN}	0	–	0.3	
Output High Voltage	V_{OH1}	$V_{DD} = 1.70\text{V}$, $I_{OH} = -6\text{mA}$, Port 3.1 only	$V_{DD} - 0.7$	–	–	V
	V_{OH2}	$V_{DD} = 1.70\text{V}$, $I_{OH} = -2.2\text{mA}$, P3.0 and P2.0–2.3	$V_{DD} - 0.7$	–	–	
	V_{OH3}	$V_{DD} = 1.70\text{V}$, $I_{OH} = -1\text{mA}$, Port 0, Port 1, P2.4–2.7, P3.4–3.5, and Port 4	$V_{DD} - 1.0$	–	–	
Output Low Voltage	V_{OL1}	$V_{DD} = 1.70\text{V}$, $I_{OL} = 8\text{mA}$, Port 3.1 only	–	0.4	0.5	V
	V_{OL2}	$V_{DD} = 1.70\text{V}$, $I_{OL} = 5\text{mA}$, P3.0 and P2.0–2.3	–	0.4	0.5	
	V_{OL3}	$V_{DD} = 1.70\text{V}$, $I_{OL} = 2\text{mA}$, Port 0, Port 1, P2.4–2.7, P3.4–3.5, and Port 4	–	0.4	1.0	

Note:

1. Supply current does not include current drawn through internal pull-up resistors or external output current loads.
2. $I_{DD11/12}$ include Flash operating current (for Flash erase/write/read operation).

Table 94. DC Electrical Characteristics (Continued) ($T_A = -25^\circ\text{C}$ to $+85^\circ\text{C}$, $V_{DD} = 1.60\text{V}$ to 3.6V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Input High Leakage Current	I_{LIH1}	$V_{IN} = V_{DD}$; all input pins except I_{LIH2} and X_{OUT}	–	–	1	μA
	I_{LIH2}	$V_{IN} = V_{DD}$, X_{IN}	–	–	20	
Input Low Leakage Current	I_{LIL1}	$V_{IN} = 0\text{V}$, all input pins except I_{LIL2} and X_{OUT}	–	–	–1	μA
	I_{LIL2}	$V_{IN} = 0\text{V}$, X_{IN}	–	–	–20	
Output High Leakage Current	I_{LOH}	$V_{OUT} = V_{DD}$, all output pins	–	–	1	μA
Output Low Leakage Current	I_{LOL}	$V_{OUT} = 0\text{V}$, all output pins	–	–	–1	μA
Pull-Up Resistors	R_{L1}	$V_{IN} = 0\text{V}$, $V_{DD} = 2.35\text{V}$, $T_A = 25^\circ\text{C}$, Ports 0–4	44	67	95	$\text{k}\Omega$
	R_{L2}	$V_{IN} = 0\text{V}$, $V_{DD} = 2.35\text{V}$, $T_A = 25^\circ\text{C}$, nRESET	150	500	1000	$\text{k}\Omega$
Feedback Resistor	R_{FD}	$V_{IN} = V_{DD}$, $V_{DD} = 2.35\text{V}$, $T_A = 25^\circ\text{C}$, $X_{IN} = V_{DD}$, $X_{OUT} = 0\text{V}$	300	700	1500	$\text{k}\Omega$
Supply Current ¹	I_{DD11}	Operating Mode ² ; $V_{DD} = 3.6\text{V}$; 8MHz crystal	–	3	6	mA
	I_{DD12}	Operating Mode ² ; $V_{DD} = 3.6\text{V}$; 4MHz crystal	–	1.5	3	mA
	I_{DD21}	Idle Mode; $V_{DD} = 3.6\text{V}$; 8MHz crystal	–	1	2	mA
	I_{DD22}	Idle Mode; $V_{DD} = 3.6\text{V}$; 4MHz crystal	–	0.5	1	mA
	I_{DD3}	Stop Mode; LVD off, $V_{DD} = 3.6\text{V}$	–	0.7	5	μA

Note:

1. Supply current does not include current drawn through internal pull-up resistors or external output current loads.
2. $I_{DD11/12}$ include Flash operating current (for Flash erase/write/read operation).

The adder by LVD on current in Backup Mode is $18\mu\text{A}$ under the conditions shown in Table 95. Backup Mode voltage is V_{DD} between LVD and POR.

Table 95. LVD Adder Current in Backup Mode

Conditions	Min.	Typ.	Max.	Unit
LVD on current in Backup Mode $V_{DD} = 1.60\text{V}$	–	18	35	μA

Table 96. Low Voltage Detect Circuit ($T_A = -25^{\circ}\text{C}$ to $+85^{\circ}\text{C}$)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Hysteresis Voltage of the LVD	ΔV	–	–	100	200	mV
Low Level Detect Voltage for Backup Mode	LVD	–	1.60	1.65	1.70	V
Low Level Detect Voltage for Flag Indicator	LVD_FLAG1	–	1.80	1.90	2.00	V
	LVD_FLAG2	–	1.90	2.00	2.10	V
	LVD_FLAG3	–	2.00	2.10	2.20	V
	LVD_FLAG4	–	2.10	2.20	2.30	V

As indicated in Table 97, the voltage gaps (LVD_GAPn [n = 1 to 4]) between LVD and LVD FLAGn (n = 1 to 4) feature $\pm 80\text{mV}$ distribution. LVD and LVD FLAGn (n = 1 to 4) are not overlapped. The variation of the LVD FLAGn (n = 1 to 4) and LVD is always shifted in the same direction. Essentially, if one chip exhibits a positive tolerance (for example, $+50\text{mV}$) in LVD FLAG, LVD has a positive tolerance.

Table 97. Low Voltage Detect Circuit ($T_A = -25^{\circ}\text{C}$ to $+85^{\circ}\text{C}$)

Symbol	Min.	Typ.	Max.	Unit
LVD_GAP1	170	250	330	mV
LVD_GAP2	270	350	430	mV
LVD_GAP3	370	450	530	mV
LVD_GAP4	470	550	630	mV
GAP Between LVD_Flag1 and LVD_Flag2	50	100	150	mV
GAP Between LVD_Flag2 and LVD_Flag3	50	100	150	mV
GAP Between LVD_Flag3 and LVD_Flag4	50	100	150	mV

Table 98. LVD Enable Time ($T_A = -25^{\circ}\text{C}$ to $+85^{\circ}\text{C}$)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
LVD enable time	t_{LVD}	$V_{\text{DD}} = 1.4\text{V}$	–	–	50	μs

In Stop Mode, LVD turns off. When an external interrupt occurs, LVD requires t_{LVD} during max. $50\mu s$ to wake up. If V_{DD} is below V_{LVD} after external interrupt, chip enters Backup Mode. Because t_{LVD} time is not enough to start oscillation, chip is not operated to abnormal state.

Table 99. Power On Reset Circuit ($T_A = -25^\circ C$ to $+85^\circ C$)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Power on reset (POR) Voltage	V_{POR}	–	0.8	1.1	1.4	V

Table 100. Data Retention Supply Voltage in Stop Mode ($T_A = -25^\circ C$ to $+85^\circ C$)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Data Retention Supply Voltage	V_{DDDR}	–	0.8	–	3.6	V
Data Retention Supply Current	I_{DDDR}	$V_{DDDR} = 1.0V$ Stop Mode	–	–	1	μA

Note: Data Retention Supply Current means that the minimum supplied current for data retention. When the battery voltage is not sufficient (i.e., the supply current is $< 1\mu A$), the data retention could be not be guaranteed.

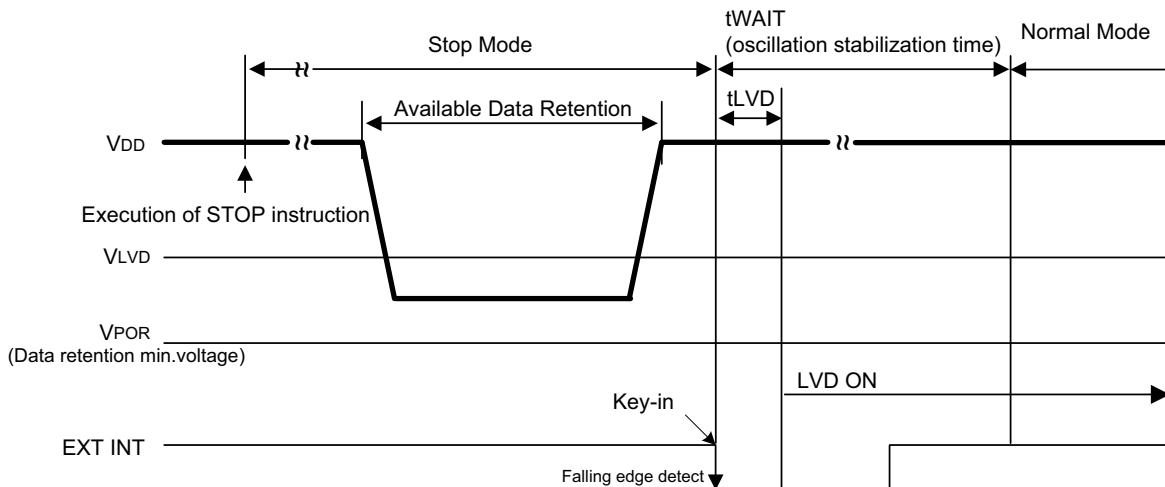


Figure 93. Stop Mode to Normal Mode Timing Diagram, #1 of 2

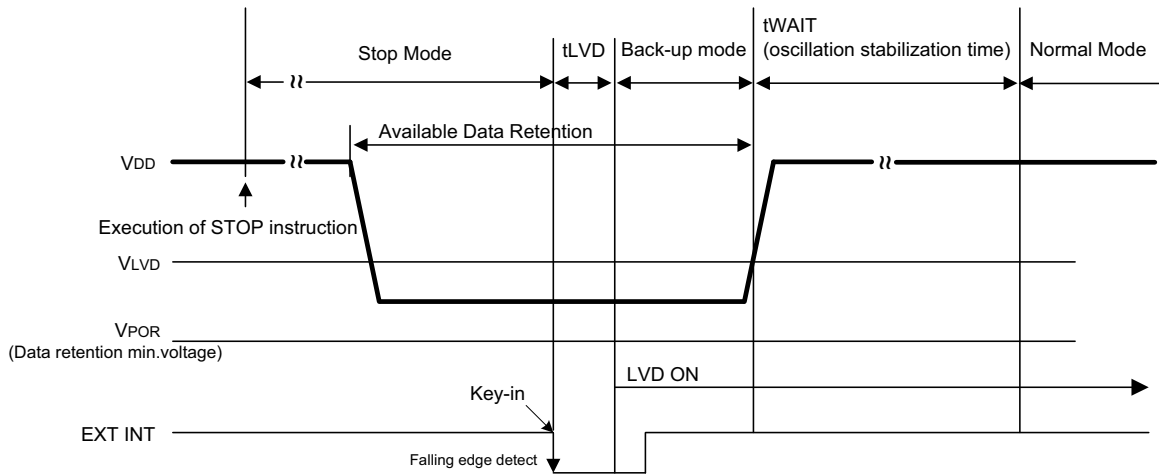


Figure 94. Stop Mode to Normal Mode Timing Diagram, #2 of 2

Table 101. AC Electrical Characteristics ($T_A = -25^{\circ}\text{C}$ to $+85^{\circ}\text{C}$)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Interrupt Input High, Low Width	t_{INTH} , t_{INTL}	P0.0–P0.7, P2.0–P2.7 $V_{\text{DD}} = 3.6\text{V}$	200	300	–	ns
nRESET Input Low Width	t_{RSL}	Input $V_{\text{DD}} = 3.6\text{V}$	1000	–	–	

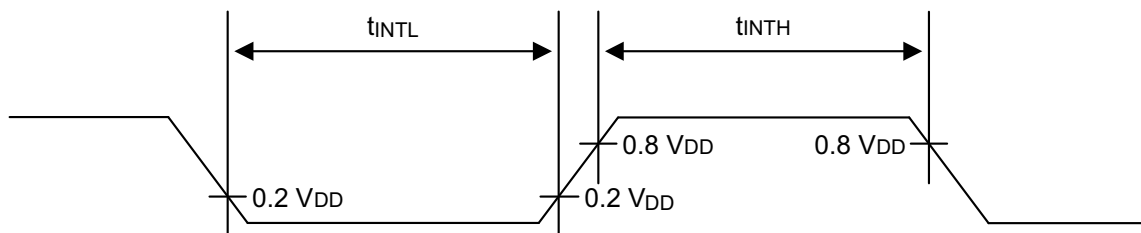


Figure 95. Input Timing for External Interrupts, Ports 0 and 2

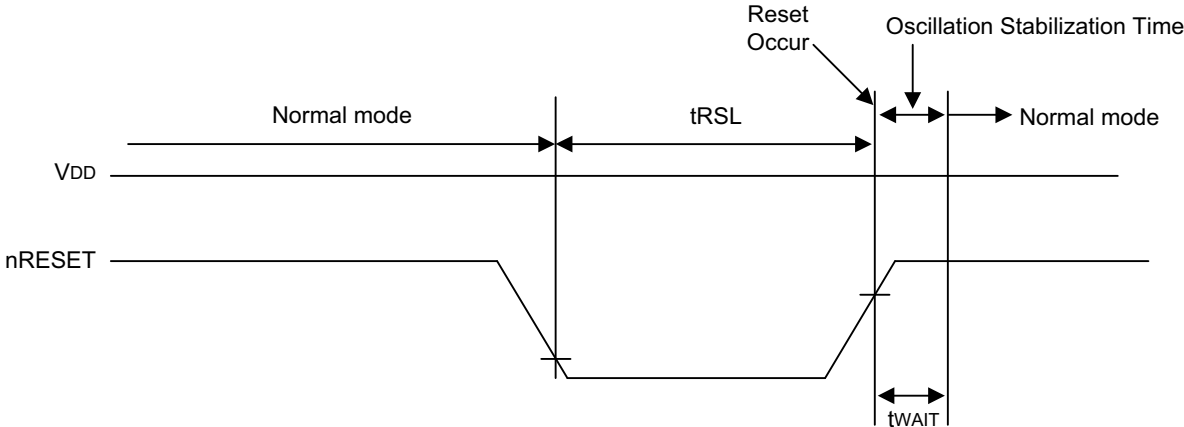


Figure 96. Input Timing for Reset (nRESET Pin)

► **Note:** In Figure 96, T_{WAIT} is the same as $4096 \times 16 \times 1/f_{OSC}$.

Table 102. Oscillation Characteristics ($T_A = -25^{\circ}C$ to $+85^{\circ}C$)

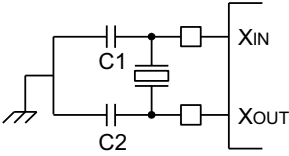
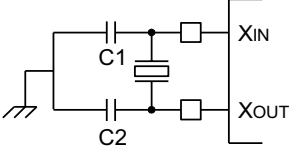
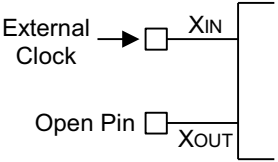
Oscillator	Clock Circuit	Conditions	Min.	Typ.	Max.	Unit
Crystal		CPU clock oscillation frequency	1	–	8	MHz
Ceramic		CPU clock oscillation frequency	1	–	8	MHz
External Clock		X_{IN} input frequency	1	–	8	MHz

Table 103. Input/Output Capacitance ($T_A = -25^{\circ}\text{C}$ to $+85^{\circ}\text{C}$)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Input Capacitance	C_{IN}	$f = 1\text{ MHz}$	–	–	10	pF
Output Capacitance	C_{OUT}	$V_{DD} = 0\text{ V}$, unmeasured pins are connected to V_{SS} .				
I/O Capacitance	C_{IO}					

Table 104. Oscillation Stabilization Time ($T_A = -25^{\circ}\text{C}$ to $+85^{\circ}\text{C}$, $V_{DD} = 3.6\text{ V}$)

Parameter	Conditions	Min.	Typ.	Max.	Unit
Main crystal	$f_{OSC} > 1\text{ MHz}$	–	–	20	ms
Main ceramic	Oscillation stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	10	ms
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	–	–	–	ms

Note:

- f_{OSC} is the oscillator frequency.
- The duration of the oscillation stabilization time (t_{WAIT}) when it is released by an interrupt is determined by the setting in the Basic Timer Control Register, BTCON.

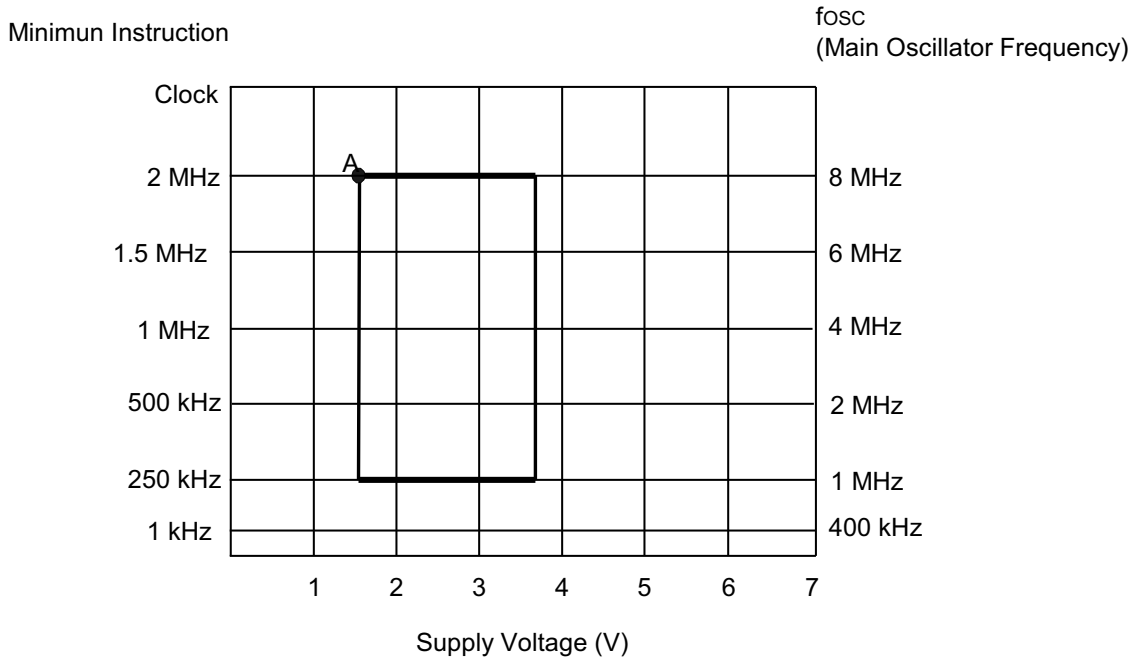


Figure 97. Operating Voltage Range

► **Note:** In Figure 97, the minimum instruction clock = $1/4n \times$ oscillator frequency, in which $n = 1, 2, 8,$ or 16 ; A = 1.65V at 8MHz.

Table 105. AC Electrical Characteristics for Internal Flash ROM ($T_A = -25^\circ\text{C}$ to $+85^\circ\text{C}$)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Flash Erase/Write/Read Voltage	Fewrv	V_{DD}	1.60	3.3	3.6	V
Programming Time ¹	Ftp	–	20	–	30	μs
Sector Erasing Time ²	Ftp 1	–	4	–	12	ms
Chip Erasing Time ³	Ftp 2	–	32	–	70	ms
Data Access Time	Ft _{RS}	$V_{DD} = 2.0\text{V}$	–	250	–	ns

Note:

1. The programming time is the period during which one (8-bit) byte is programmed.
2. The sector erasing time is the period during which all 128 bytes of one sector block are erased.
3. For the S3F80PB MCU, chip erasure is available in Tool Program Mode only.

Table 105. AC Electrical Characteristics for Internal Flash ROM ($T_A = -25^{\circ}\text{C}$ to $+85^{\circ}\text{C}$)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Number of Writing/Erasing	FNwe	–	10,000	–	–	Times
Data Retention	Ftdr	–	10	–	–	Years

Note:

1. The programming time is the period during which one (8-bit) byte is programmed.
2. The sector erasing time is the period during which all 128 bytes of one sector block are erased.
3. For the S3F80PB MCU, chip erasure is available in Tool Program Mode only.

Table 106. ESD Characteristics

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Electrostatic Discharge	V_{ESD}	HBM	2000	–	–	V
		MM	200	–	–	V
		CDM	500	–	–	V

Note: If on board programming is required, it is recommended that add a 0.1 μF capacitor between the TEST pin and V_{SS} for better noise immunity; otherwise, connect the TEST pin to V_{SS} directly. It is recommended also that add a 0.1 μF capacitor between nRESET pin and V_{SS} for better noise immunity.

Chapter 18. Mechanical Data

The S3F80PB microcontroller is currently available in a 32-pin SOP, 32-pin QFN, 44-pin QFP, and 44-pin QFN packages shown in Figures 98 through 101, respectively.

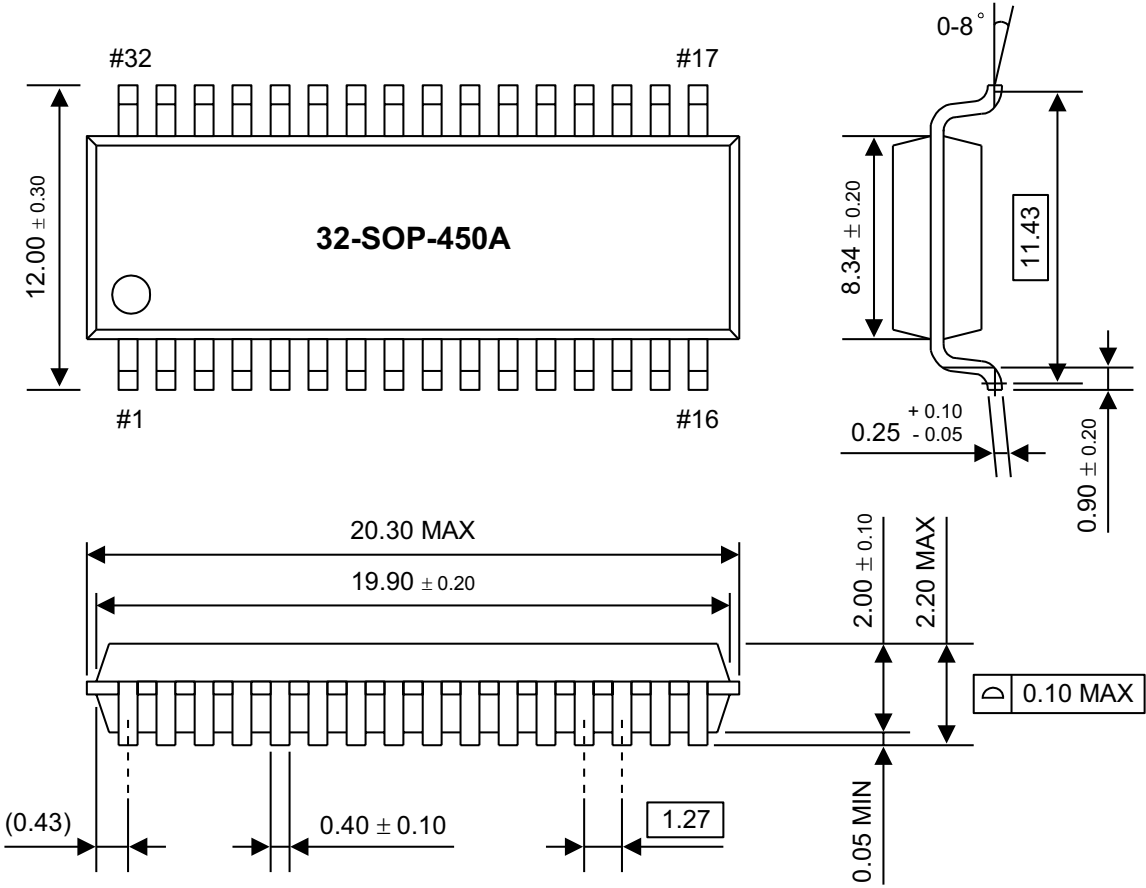


Figure 98. 32-Pin SOP Package Dimension

► **Note:** In Figure 98, dimensions shown are in millimeters.

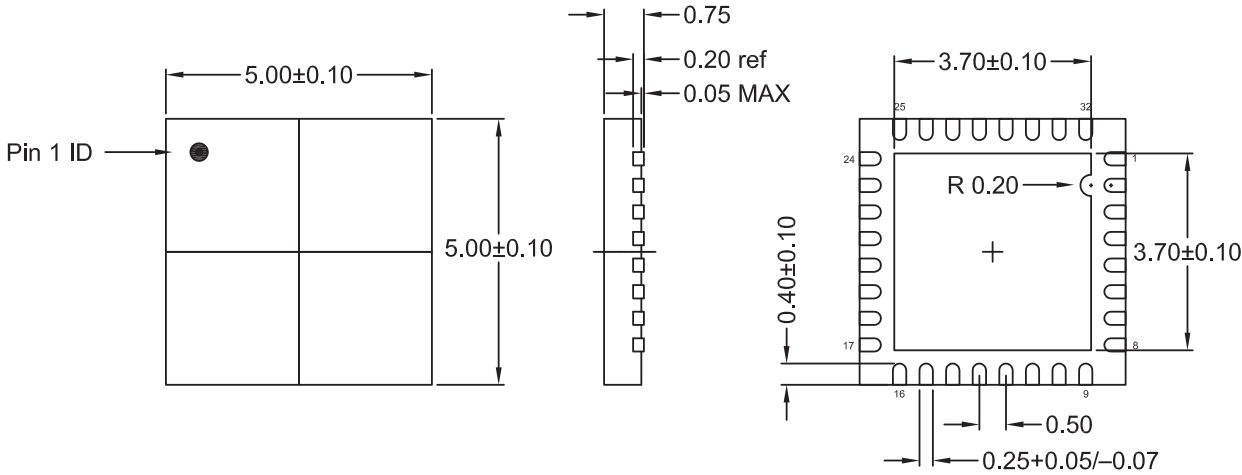


Figure 99. 32-Pin QFN Package Dimension

► **Note:** In Figure 99, dimensions shown are in millimeters.

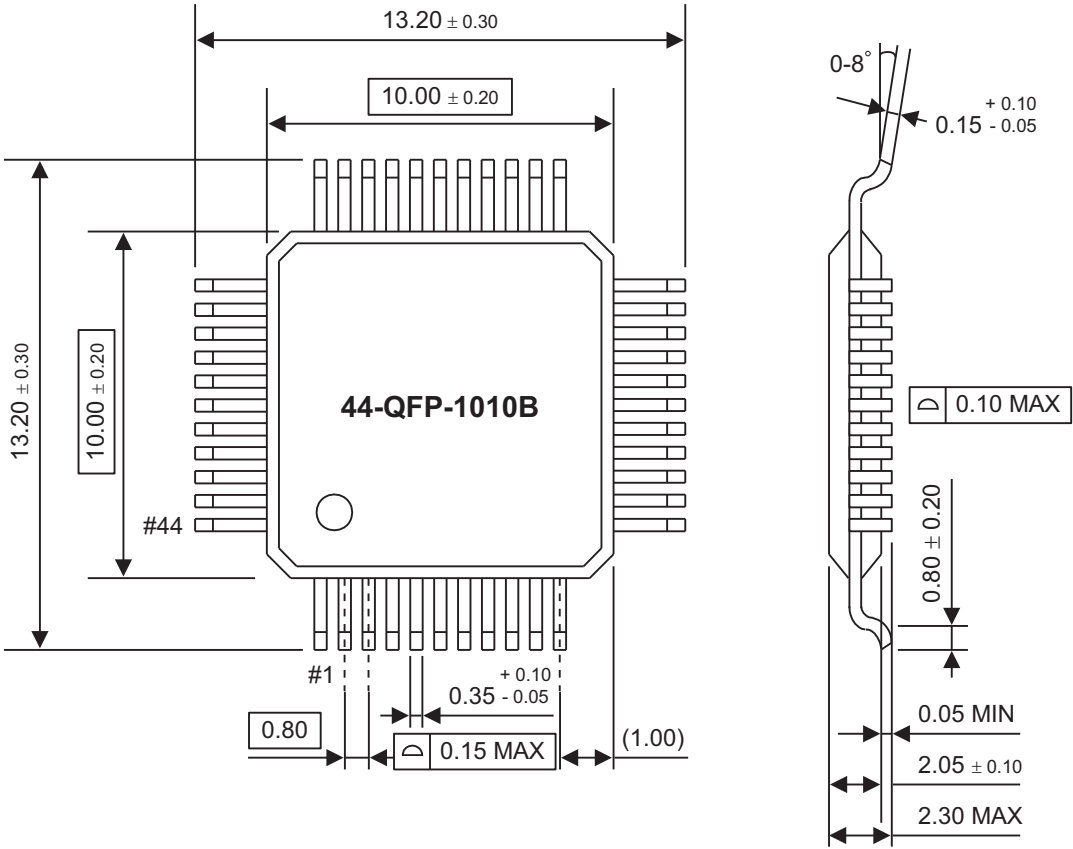


Figure 100. 44-Pin QFP Package Dimension

► **Note:** In Figure 100, dimensions shown are in millimeters.

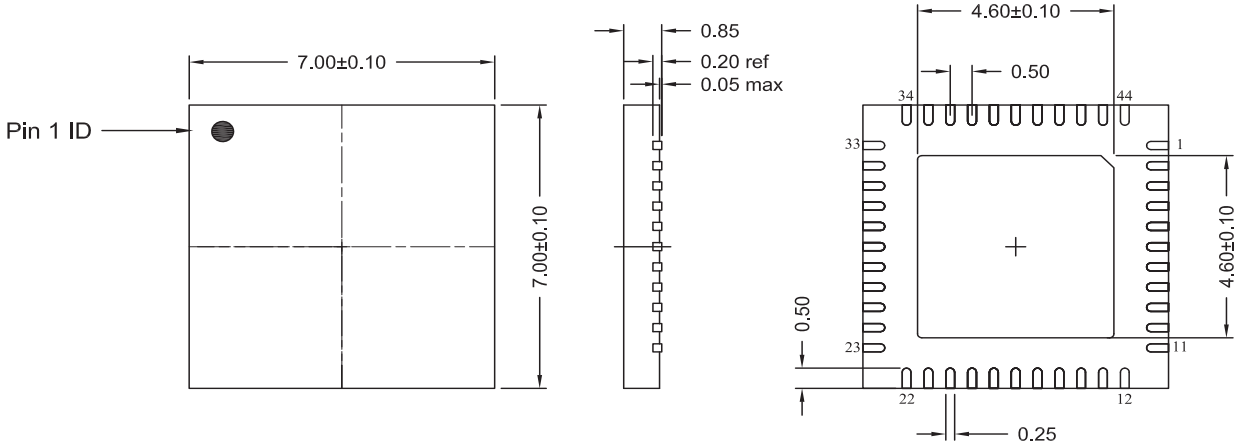


Figure 101. 44-Pin QFN Package Dimension

► **Note:** In Figure 101, dimensions shown are in millimeters.

Chapter 19. Flash Programming

This chapter discusses the Tool Program Mode of the S3F80PB single-chip MCU, which features on-chip Flash memory accessible via the serial data format.

► **Note:** See the [Embedded Flash Memory Interface](#) chapter on page 277 to learn more about User Program Mode.

Table 107 lists the pins used to read, write, and erase Flash memory when operating in Tool Program Mode.

Table 107. Flash Operations in Tool Program Mode

Normal Chip Pin Name	Pin Name	During Programming				I/O	Function
		32-Pin SOP Pkg. Pins	32-Pin QFN Pkg. Pins	44-Pin QFP Pkg. Pins	44-Pin QFN Pkg. Pins		
P0.0	SDAT	17	21	30	30	I/O	Serial data pin. Output port when reading and input port when writing. SDAT (P0.0) can be assigned as an input or push-pull output port.
P0.1	SCLK	18	22	31	31	I	Serial clock pin. Input only pin.
TEST	TEST	4	8	9	9	I	Tool Mode selection when the TEST pin sets Logic value 1. If user uses the Flash writer Tool Mode (for example, spw2+, etc.), user should connect the TEST pin to V _{DD} . (The S3F80PB MCU supplies high voltage 12.5V by internal high voltage generation circuit.)
nRESET	nRESET	7	11	12	12	I	Chip Initialization.
V _{DD}	V _{DD}	32	4	5	5	–	Power supply pin for logic circuit. V _{DD} should be tied to +3.3V during programming.
V _{SS}	V _{SS}	1	5	6	6	–	System ground voltage.

19.1. Test Pin Voltage

The TEST pin on the socket board for the OTP/MTP writer must be connected to V_{DD} (3.3 V). This TEST pin must not be connected to V_{PP} (12.5 V), which is generated from the OTP/MTP Writer. Therefore, the specific socket aboard the S3F80PB MCU must be used when writing or erasing using an OTP/MTP writer.

19.2. Operating Mode Characteristics

When 3.3 V is supplied to the TEST pin of the S3F80PB MCU, the Flash ROM Programming Mode is entered. The operating mode (read, write, or read protection) is selected according to the input signals to the pins listed in Table 108.

Table 108. Operating Mode Selection Criteria

V_{DD}	Test	REG/nMEM	Address (A15–A0)	R/W	Mode
3.3V	3.3V	0	0000h	1	Flash ROM read.
	3.3V	0	0000h	0	Flash ROM program.
	3.3V	1	0E3Fh	0	Flash ROM read protection.

Note: 0 refers to low level. 1 refers to high level.

Chapter 20. Development Tools

20.1. 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 21.2 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 21.3 describes the Zilog library-based Development Platform for Flash-based S3F8-series microcontrollers

20.2. Emulator-based Development System

Figure 102 shows an emulator-based development system utilizing an emulator to interface with an application board through a Zilog-provided Target Board.

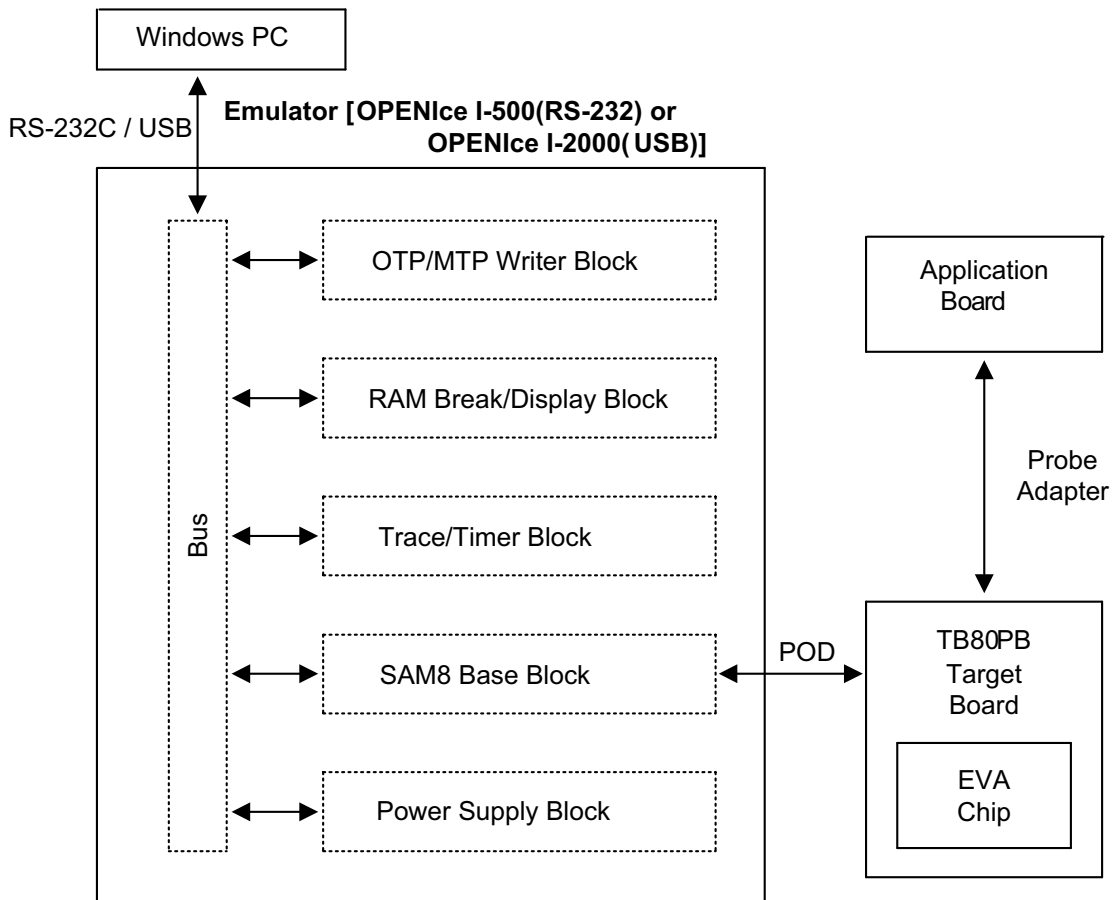


Figure 102. Emulator-based Development System Configuration

The S3 Emulator Based Development System includes the components listed in the following sections.

20.2.1. 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.

20.2.2. 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 TB80PB target board can be used with application boards targeting the S3F80PB MCU.

Figure 103 shows how the TP80PB Target Board is configured. The symbol ◀ marks the starting point of the jumper signals.

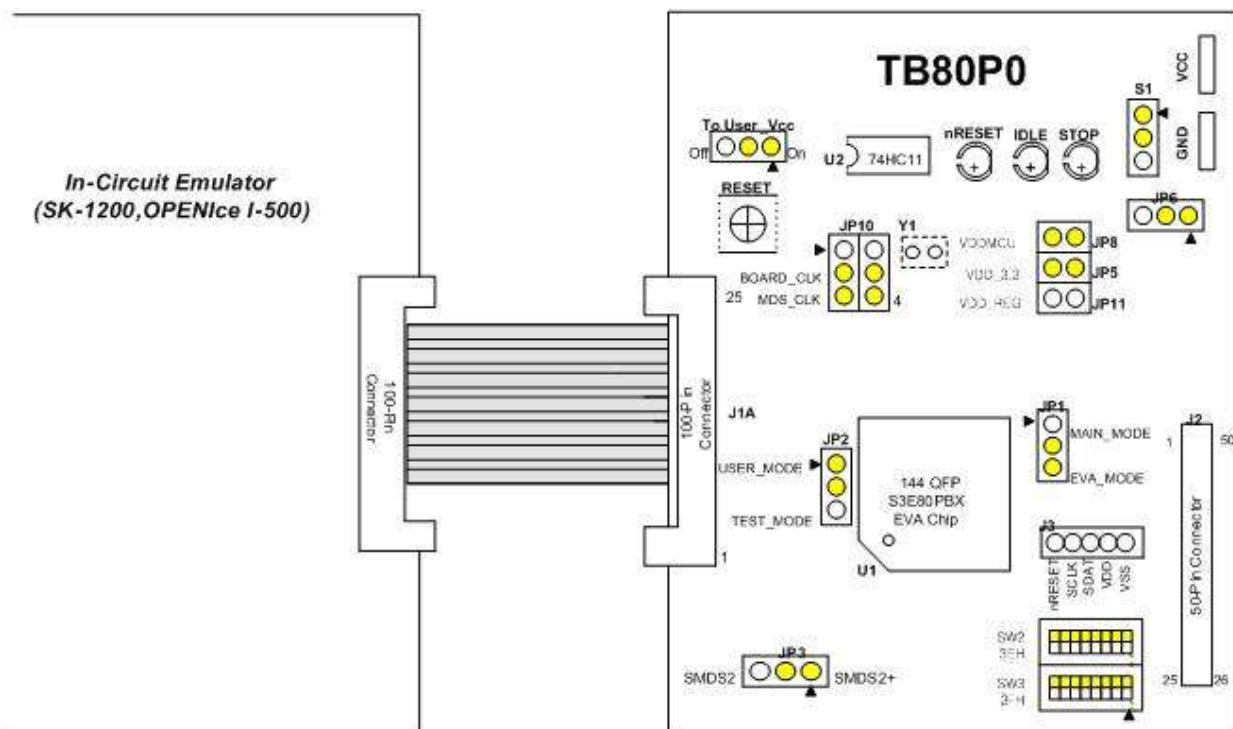


Figure 103. TB80P0 Target Board Configuration

- **Notes:** 1. The TB80PB target board should normally be supplied 3.3V. For target board operation, the power supply from the emulator should therefore also be set to 3.3V. If the power supply from the emulator is set to 5V, activate a 3.3V regulator on the target board by setting the related jumpers shown in Figure 103.
2. The ◀ symbol marks the starting point of the jumper signals.

Table 109 lists the operation selection settings for the TB80PB Target Board.

Table 109. TB80PB Jumper Settings

JP#	Description	1–2 Connection	2–3 Connection	Default Setting
S1	Target board power source	Use JP7 (V _{CC})	Not Connected	Join 1–2
JP1	Target board mode selection	H: Main Mode (not supported)	L: EVA Mode	Join 2–3
JP2	Operation mode	H: User Mode	L: Test Mode (not supported)	Join 1–2
JP3	MDS version	SMDS2	SMDS2+,SK-1200,OPENIce I-500	Join 2–3
JP4 (To User_V _{CC})	Target System is supplied V _{DD}	Target System is supplied V _{DD} from user system.	Target System is not supplied V _{DD} from user system.	ON setting
JP5	Board power connection	Board power connection		Connect
JP6	When supplied 5V in target board, generation of 3.3V using regulator	In selecting a 3.3V emulator, do not use a 3.3V regulator.	In selecting a 3.3V emulator, do not use a 3.3V regulator.	Join 1–2
JP7, JP9	Power connector	JP7: V _{CC} ; JP9: GND		–
JP8	80PB V _{DD} power connection	80PB V _{DD} power connection		Connect
JP10	Clock source selection	When using the internal clock source which is generated from Emulator, join connector 2–3 and 4–5 pin. If user wants to use the external clock source as a crystal, user should change the jumper setting from 1–2 to 5–6 and connect Y1 to an external clock source.		Emulator 2–3 4–5

Table 109. TB80PB Jumper Settings (Continued)

JP#	Description	1–2 Connection	2–3 Connection	Default Setting
JP11	Regulator 3.3V output connection	Connection between regulator out voltage and 80PB's Power VDD when using the regulator. When debugging with an emulator, JP11 is not required to be connected.		No Connect
JP11, 12	Not used for TB80PB		–	No Connect
SW1	Generation low active reset signal to S3F80PB EVA chip	Push switch		–
SW2	Smart Option at address 3Eh	Dip switch for the Smart Option. This byte is mapped to address 3Eh for special functions; see the Embedded Flash Memory Interface chapter on page 277.		No Connect
SW3	Smart Option at address 3Fh	Dip switch for the Smart Option. This byte is mapped to address 3Fh for special functions; see the Embedded Flash Memory Interface chapter on page 277.		–
X1	External Clock Source	Connecting points for external clock source		Not connected

The TB80PB Target Board features the LEDs listed in Table 110.

Table 110. TB80PB Target Board LEDs

nRESET LED	This LED is OFF when the Reset switch is ON.
Idle LED	This LED is ON when the evaluation chip (S3E80PB) is in Idle Mode.
Stop LED	This LED is ON when the evaluation chip (S3E80PB) is in Stop Mode.

Pin assignments for the TB80PB Target Board are shown in Figure 104.

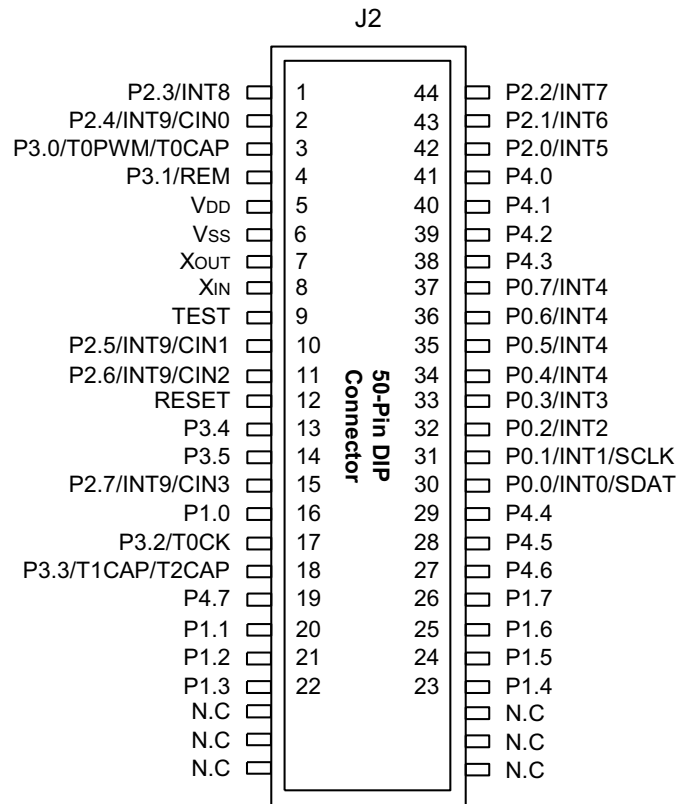


Figure 104. 50-Pin Connector Pin Assignment, User System

► **Note:** In Figure 104, N.C = No Connection.

The TB80PB Target Board should normally be supplied with 3.3V. Therefore, the power supply from the emulator should be set to 3.3V for target board operation. If the power supply from the emulator is set to 5V, activate a 3.3V regulator on the TB80PB Target Board by setting the related jumpers; see Table 109.

20.2.3. 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 105. This allows the EVA chip on the target board to interact with components on the application board while debugging the application.

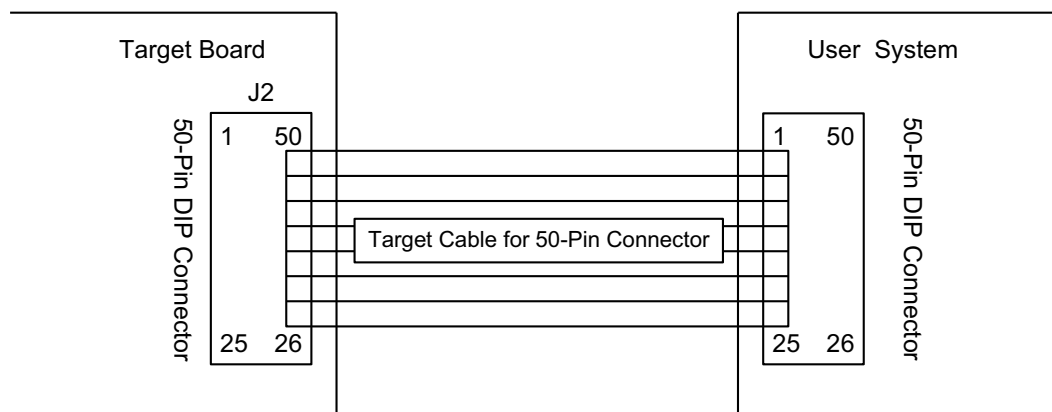


Figure 105. TB80PB Probe Adapter Cable and Application Board

20.3. 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

20.3.1. Zilog Developer Platform Components

Figure 106 shows the simplicity of connecting all of the components of the Zilog developer platform.

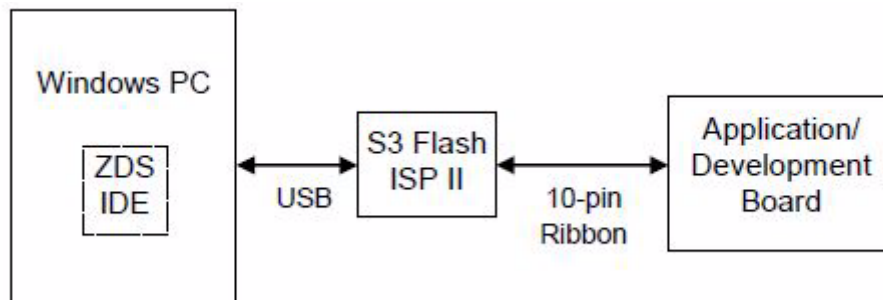


Figure 106. Zilog Development Platform

20.3.1.1. 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.

20.3.1.2. 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.

20.3.1.3. 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 107.

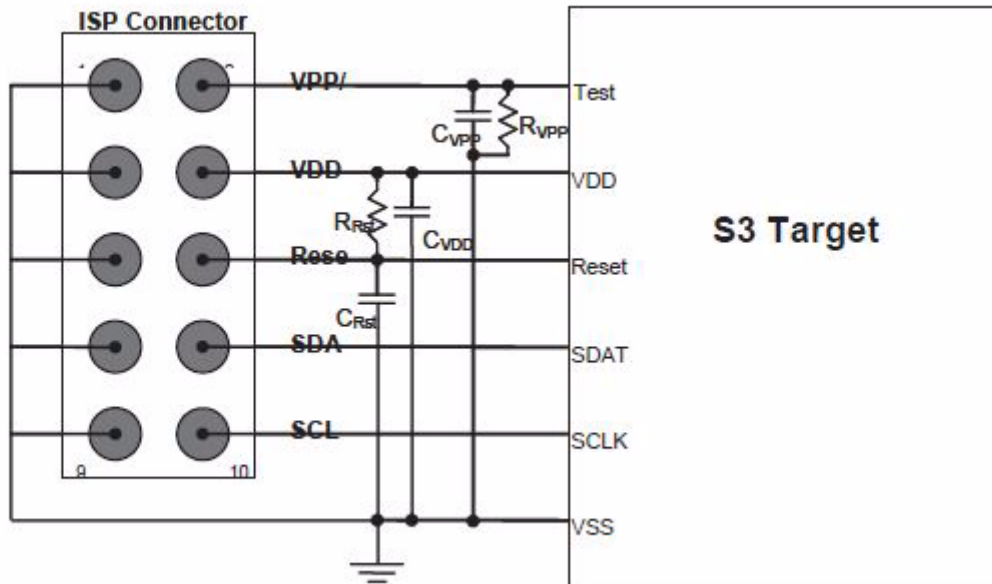


Figure 107. PCB Design Guide for In-System Programming

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 RRST and CRST. In this instance, it is not necessary to include RVPP or CVPP.

Table 111 shows the recommended values for the passive components in the ISP II circuit of Figure 107.

Table 111. ISP II Circuit Recommended Values

ISp Signal (Pin Number)	Passive Component	Notes
VPP/ Test (2)	CVPP = 0.1 uF RVPP = 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)	CVDD = 0.1 uF	
Reset (6)	CRST = 0.1 uF RRST = 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.

20.3.2. 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 OPEN-ice-i2000 emulator. For information regarding 3rd party development tools, see the [Development Tools](#) section on page 325.

20.3.3. 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 depend-

ing 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.

20.3.4. Development Tools

Zilog, in conjunction with third parties, provides a complete line of development tools that support the S3 Family of Microcontrollers. With extensive experience in developing MCU systems, these third party firms are bonafide leaders in MCU development tool technology.

In-Circuit Emulators

- [YIC](#) – OPENice-i500/2000

OPENice-i500	YIC System
	<ul style="list-style-type: none"> • TEL: 82-31-278-0461 • FAX: 82-31-278-0463 • E-mail: support@yicsystem.com • URL: http://www.yicsystem.com
OPENice-i2000	YIC System
	<ul style="list-style-type: none"> • 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](#) – S3 Flash In System Programmer II

Programmers (Writer)

- [Seminix](#) – GW-uni2
- [C & A Tech](#) – GW-Pro2
- [Eltec](#) – 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](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](http://www.elnec.com)

S3 Flash In-System Programmer II

Zilog



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 debugging of 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

- TEL: (408) 457-9000
- FAX: (408) 416-0223
- E-mail: s3sales@zilog.com
- URL: <http://www.zilog.com>

Chapter 21. Ordering Information

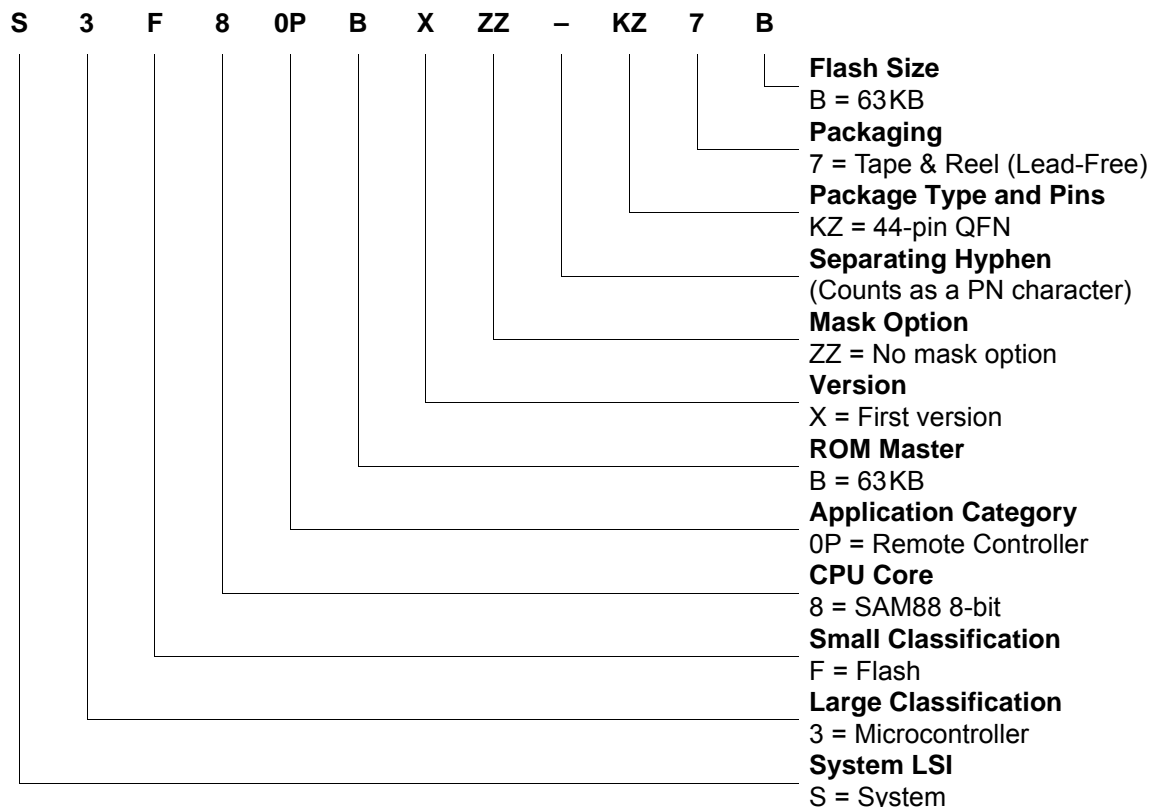
Table 112 identifies the basic features and package styles available for the S3F80PB MCU.

Table 112. Ordering Information for the S3F80PB MCU

Device	Flash Size	RAM Size	GPIO	Bit-Programmable Ports	Package
S3F80PBXZZ-QZ8B	63KB	1296 B	38	4	44-pin QFP
S3F80PBXZZ-KZ7B	63KB	1296 B	38	4	44-pin QFN
S3F80PBXZZ-SO9B	63KB	1296 B	26	4	32-pin SOP
S3F80PBXZZ-KO7B	63KB	1296 B	26	4	32-pin QFN
S3F80PBXZZ-C0CB	63KB	1296 B	38	4	Pellet (Die)

21.0.1. Part Number Suffix Designations

Zilog part numbers consist of a number of components. For example, part number S3F80PBXZZ-KZ7B is an unmasked 8-bit general-purpose MCU with 63 KB of Flash memory in a 44-pin QFP package.



Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at <http://support.zilog.com>.

To learn more about this product, find additional documentation, or to discover other facts about Zilog product offerings, please visit the Zilog Knowledge Base at <http://zilog.com/kb> or consider participating in the Zilog Forum at <http://zilog.com/forum>.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at <http://www.zilog.com>.