S3 Family 8-Bit Microcontrollers

S3F80P5 MCU

Product Specification

PS031904-0115

PRELIMINARY

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

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
Jan 2015	04	Revised the Ordering Information table to reflect the Thin 24-Pin ELP package as a Tape & Reel offering.	304
Nov 2014	03	Added mechanical drawing of the 0.55 mm 24-pin ELP package.	294
Oct 2014	02	Minor correction to S3F80P5 MCU Block Diagram (Figure 1); modified pin circuit diagrams, Figures 4 through 8; corrected Noncontiguous 16-Byte Working Register Block (Figure 15); corrected D2 opcode, Op Code Quick Reference (0–7) (Table 20); corrected How to Use the NEXT Instruction (Figure 46); modified Falling and Rising Rate of Operating Voltage (Table 31); added V_{OH} specifications to DC Electrical Characteristics (Table 85); added tolerance footnote to Oscillation Characteristics (Table 91); updated to Zilog style guidelines.	4, 8–12, 22, 74, 145, 188, 284, 288
Mar 2014	01	Original Zilog issue.	All

Table of Contents

List of Figures

List of Tables

Chapter 1. Overview

Zilog's S3F8 Series 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 contain one or more interrupt sources and vectors. Fast interrupt processing (within a minimum of four CPU clocks) can be assigned to specific interrupt levels.

1.1. S3F80P5 Microcontroller

The S3F80P5 MCU features 18KB of Flash ROM. Using a proven modular design approach, the S3F80P5 MCU was developed by integrating the following peripheral modules with the SAM88 core:

- Internal LVD circuit and 9 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 auto-reload function and one-shot or repeat control

The S3F80P5 MCU is a versatile general-purpose microcontroller that is especially suitable for use in applications requiring remote-control transmission. The S3F80P5 MCU is available in the 24-pin SOP and 24-pin ELP packages.

1.2. Features

The S3F80P5 MCU offers the following features:

- SAM8 RC CPU core
- Program memory (ROM)

  - 18 KB program memory
  - Internal Flash (program) memory
    - Sector size: 128 bytes
    - 10 years data retention
    - Fast programming time; user program and sector erase available
    - Endurance: 10,000 erase/program cycles
    - Byte-programmable
    - User-programmable via LDC instruction
    - External serial programming support
    - Expandable On-Board Program (OBP) sector

- Executable memory: 1 KB RAM
- Data memory: 272-byte general-purpose RAM
- Instruction set
  - 78 instructions
  - IDLE and STOP instructions added for power-down modes

- Instruction execution time: 500 ns at 8 MHz f_{OSC} (minimum)
- Interrupts: 17 interrupt sources with 14 vectors and 7 levels
- 19 programmable I/O ports in both of the 24-pin packages (SOP/ELP):
  - Two 8-bit I/O ports (P0, P1)
  - One 1-bit port (P2)
  - One 2-bit port (P3)

- Carrier frequency generator:
  - One 8-bit counter with autoreload function
  - 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 Interval, Capture, and PWM modes
  - One 16-bit timer/counter (Timer 1) with Interval and Capture modes
  - One 16-bit timer/counter (Timer 2) with Interval and Capture modes

- Backup Mode:
  - When V_{DD} is lower than V_{LVD}, LVD is ON and the MCU enters Backup Mode to block oscillation

- Low Voltage Detect circuit:
  - Low voltage detection to enter Backup Mode and Reset; 1.65 V (Typ.) ±50 mV
  - Low voltage detection to control the LVD_Flag bit; 1.90 V, 2.00 V, 2.10 V, 2.20 V (Typ.) ±100 mV (selectable)
  - LVD Reset is enabled during operation when the V_{DD} is falling and passing V_{LVD}; the MCU enters Backup Mode; when the V_{DD} is rising, a reset pulse is generated at $V_{DD} > V_{LVD}$.
  - LVD is disabled in Stop Mode; V_{DD} is not falling to V_{POR}, a reset pulse is not generated

- Operating temperature range: –25 °C to +85 °C
- Operating voltage range: 1.60 V to 3.6 V at 1MHz to 8 MHz
- Packages:
  - 24-pin SOP
  - 24-pin ELP
  - Pellet (Die)

1.3. Block Diagram

Figure 1 shows a block diagram for the S3F80P5 MCU.

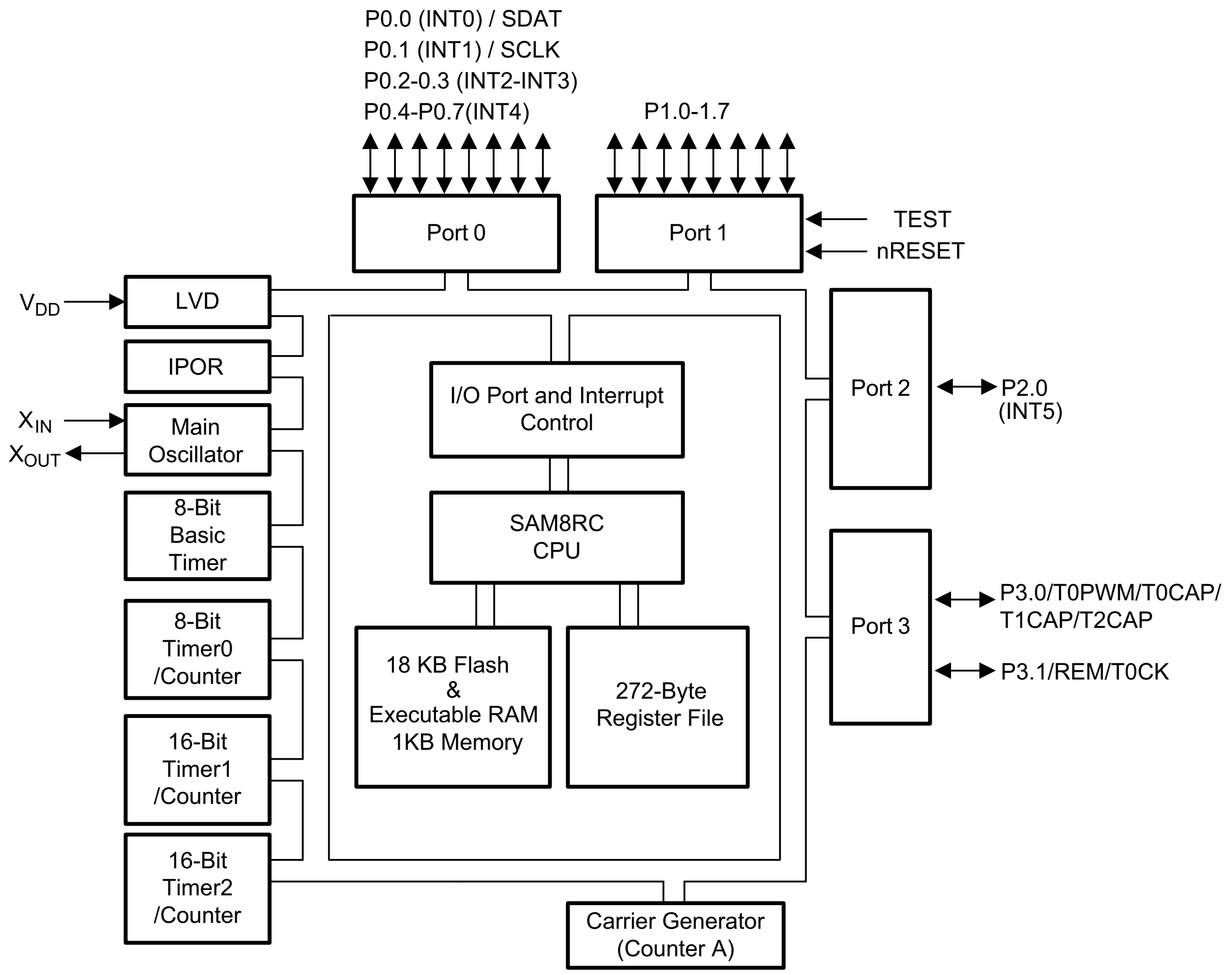


Figure 1. S3F80P5 MCU Block Diagram

1.4. Pin Assignments

Figures 2 and 3 show the pin assignments for the 24-pin SOP and 24-pin ELP packages, respectively.

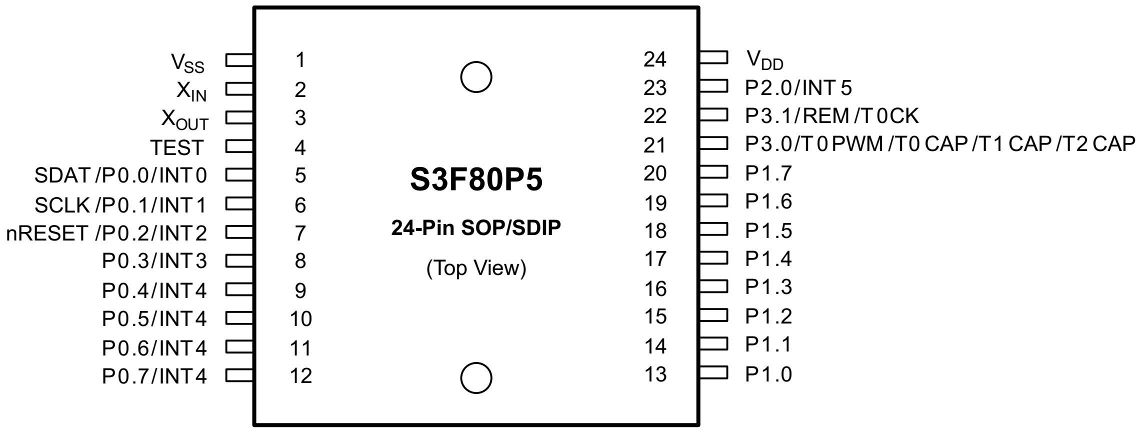


Figure 2. Pin Assignments, 24-Pin SOP

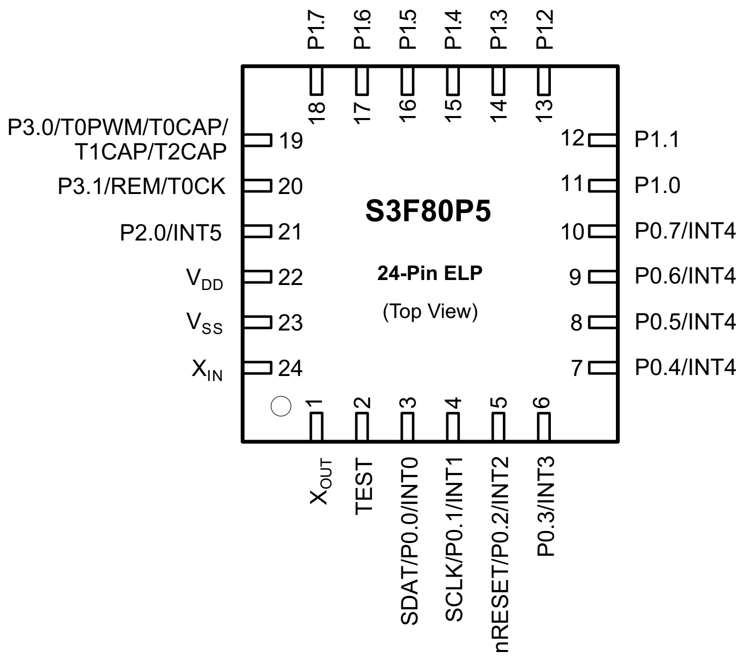


Figure 3. Pin Assignments, 24-Pin ELP Package

1.5. Pin Descriptions

Table 1 identifies each pin in each of the S3F80P5 MCU's 24-pin packages.

Table 1. S3F80P5 MCU Pin Descriptions

				Pin Numbers		Shared Functions	
Pin Names	**Pin Description**	**Pin Type**	**Circuit Type**	**24-Pin SOP**	**24-Pin ELP**	**24-Pin SOP**	**24-Pin ELP**
P0.0–P0.7	I/O port with bit-programmable pins. Configurable to input or push-pull output mode. Pull-up resistors are assignable by software. Pins can be assigned individually as external interrupt inputs with noise filters, interrupt enable/disable, and interrupt pending control. SED & R circuit built in P0 for stop releasing. In Tool Mode, P0.0 and P0.1 are assigned as serial MTP interface pins; SDAT and SCLK.	I/O	1	5–12	3–10	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	13–20	11–18	–	–
P2.0	I/O port with bit-programmable pins; Schmitt trigger input or push-pull output and software-assignable pull-ups. Alternately used for external interrupt input (noise filters, interrupt enable and pending control).	I/O	3	23	21	External Interrupts	INT5
P3.0	I/O port with 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 contains 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	26	19	T0PWM/T0CAP/T1CAP/T2CAP	

Table 1. S3F80P5 MCU Pin Descriptions (Continued)

				Pin Numbers		Shared Functions	
Pin Names	**Pin Description**	**Pin Type**	**Circuit Type**	**24-Pin SOP**	**24-Pin ELP**	**24-Pin SOP**	**24-Pin ELP**
P3.1	I/O port with 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 contains 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	27	20	REM/T0CK	
X_{IN} X_{OUT}	System clock input and output pins.	–	–	2, 3	24, 1	–	
TEST	Test signal input pin. If onboard programming is required, Zilog recommends adding a 0.1µF capacitor between the TEST pin and V_{SS} for better noise immunity; otherwise, connect the TEST pin to V_{SS} directly.	I	–	4	2	–	
V_{DD}	Power supply input pin.	–	–	28	22	–	
V_{SS}	Ground pin.	–	–	1	23	–	

1.6. Pin Circuits

Figure 4 shows the Type 1 pin circuit for Port 0.

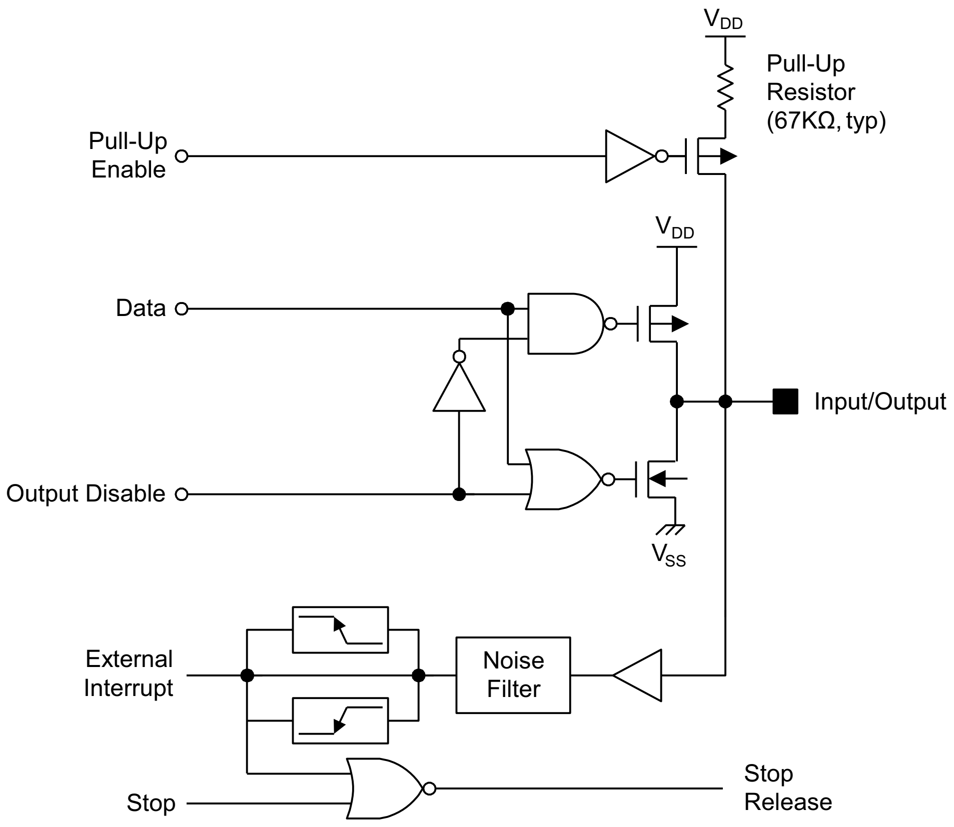


Figure 4. Pin Circuit Type 1, Port 0

Figure 5 shows the Type 2 pin circuit for Port 1.

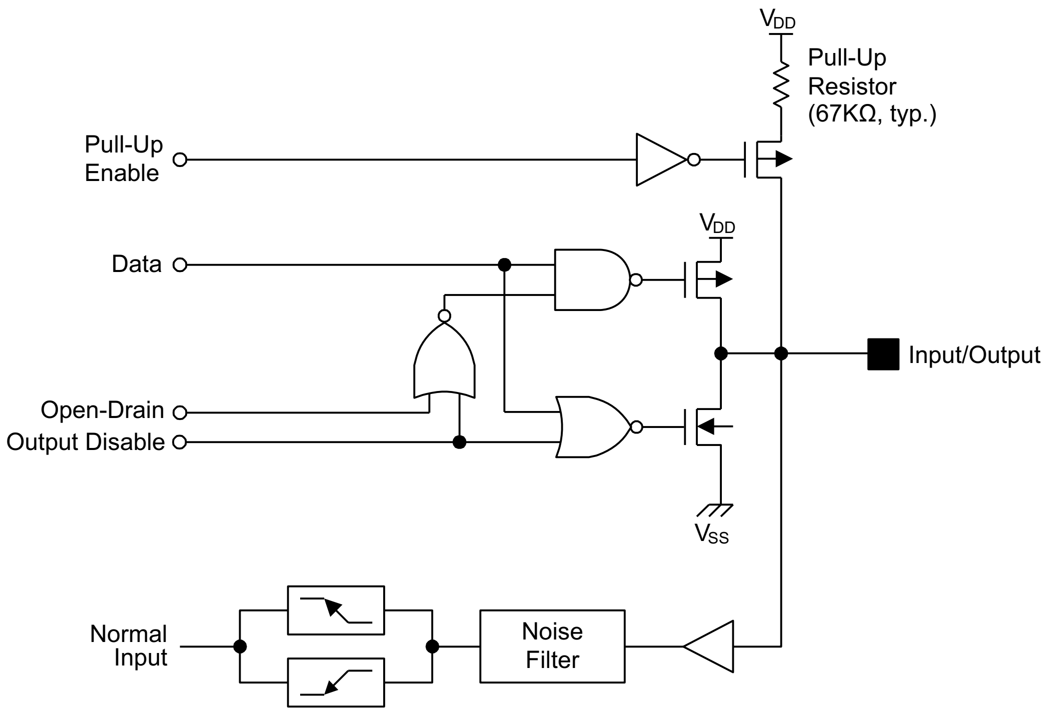


Figure 5. Pin Circuit Type 2, Port 1

Figure 6 shows Type 2 pin circuit for Port 2.

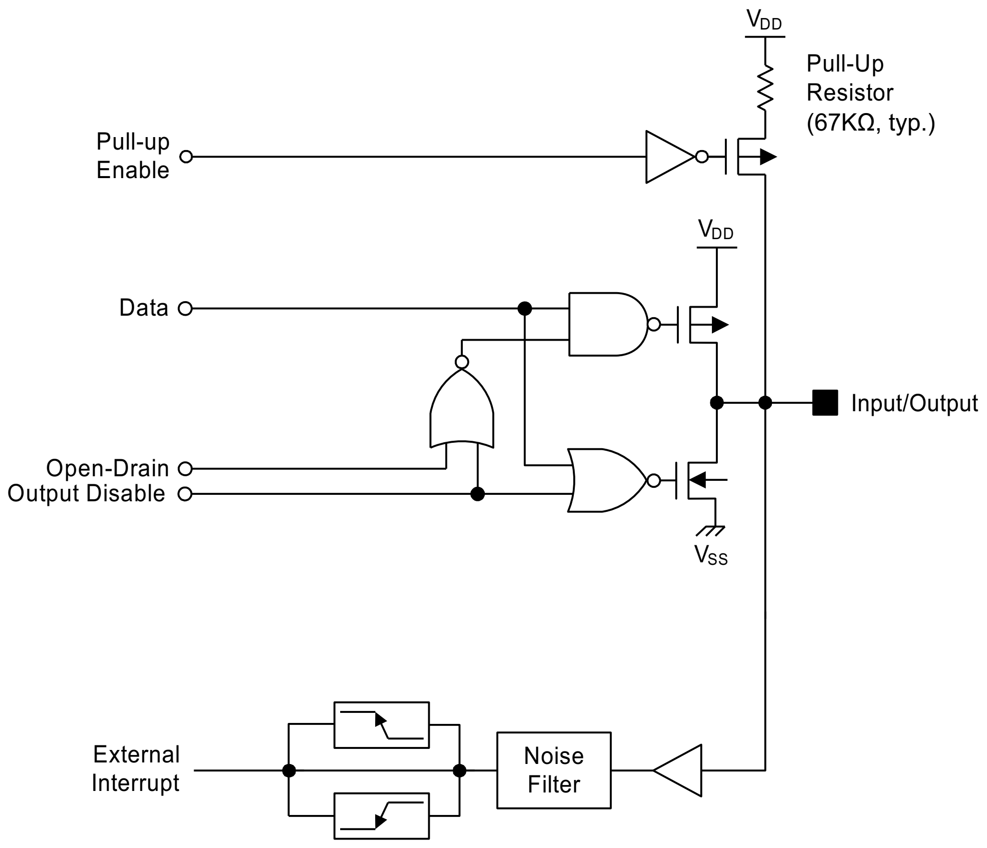


Figure 6. Pin Circuit Type 2, Port 2

Figure 7 shows the Type 4 pin circuit for Port 3, bit 0.

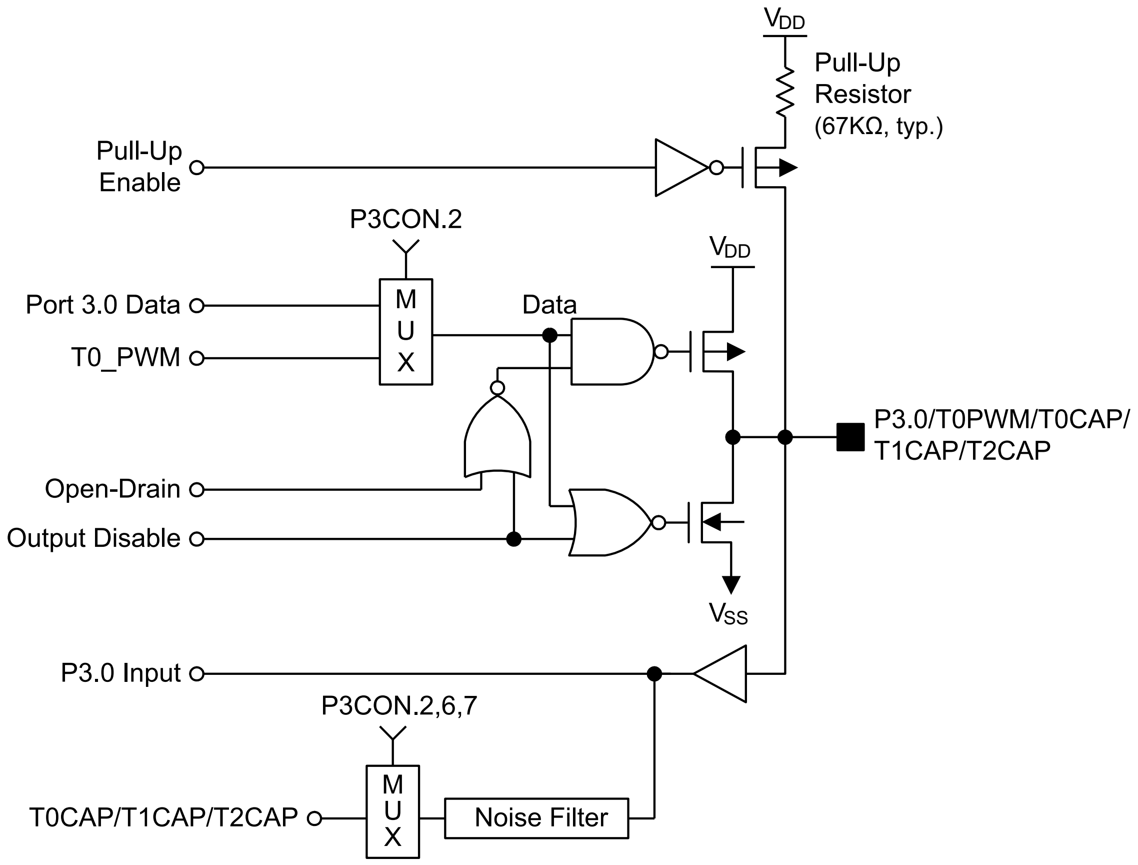


Figure 7. Pin Circuit Type 4, Port 3.0

Figure 8 shows the Type 5 pin circuit for Port 3, bit 1.

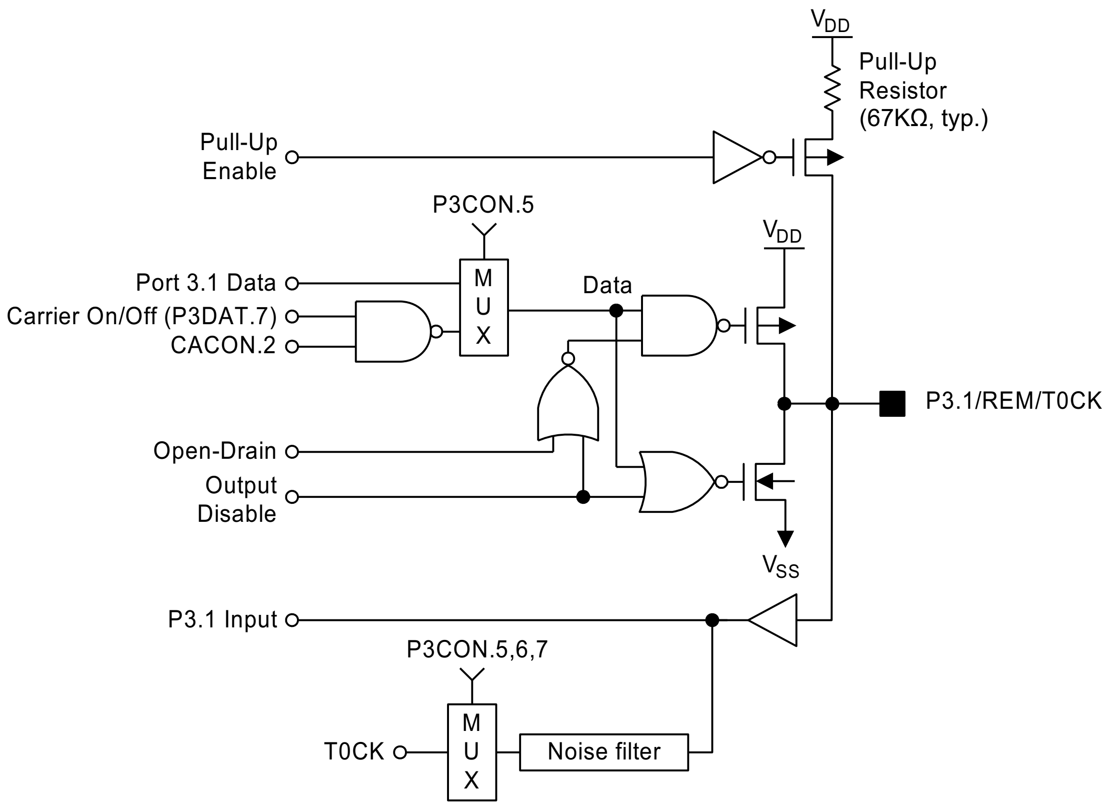


Figure 8. Pin Circuit Type 5, Port 3.1

Chapter 2. Address Space

The S3F80P5 microcontroller features the following two types of address space:

- Internal program (Flash) ROM
- Internal register file

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

The S3F80P5 MCU features 18KB of programmable internal Flash ROM; an external memory interface is not implemented.

There are 333 mapped registers in the internal register file. 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, and 39 registers are mapped peripheral control and data registers.

2.1. Program Memory

Program memory (ROM) stores program code or table data. The S3F80P5 MCU features 18KB internal programmable Flash memory. The address range for this Flash program memory is `0000h–47FFh`, as shown in Figure 9.

The first 256 bytes of the ROM space, `0h–0FFh`, are reserved for interrupt vector addresses. Unused locations in this address range (i.e., addresses in the range `0000h–00FFh` except `03Ch`, `03Dh`, `03Eh`, and `03Fh`) can be used as normal program memory. These exceptions at addresses `03Ch`, `03Dh`, `03Eh`, and `03Fh` are used as Smart Option ROM cells.

The program memory address at which program execution starts after a reset is `0100h` by default. If you use the ISP sectors for ISP software storage, the reset vector address can be changed by setting the Smart Option, as shown in Figure 10 on page 15.

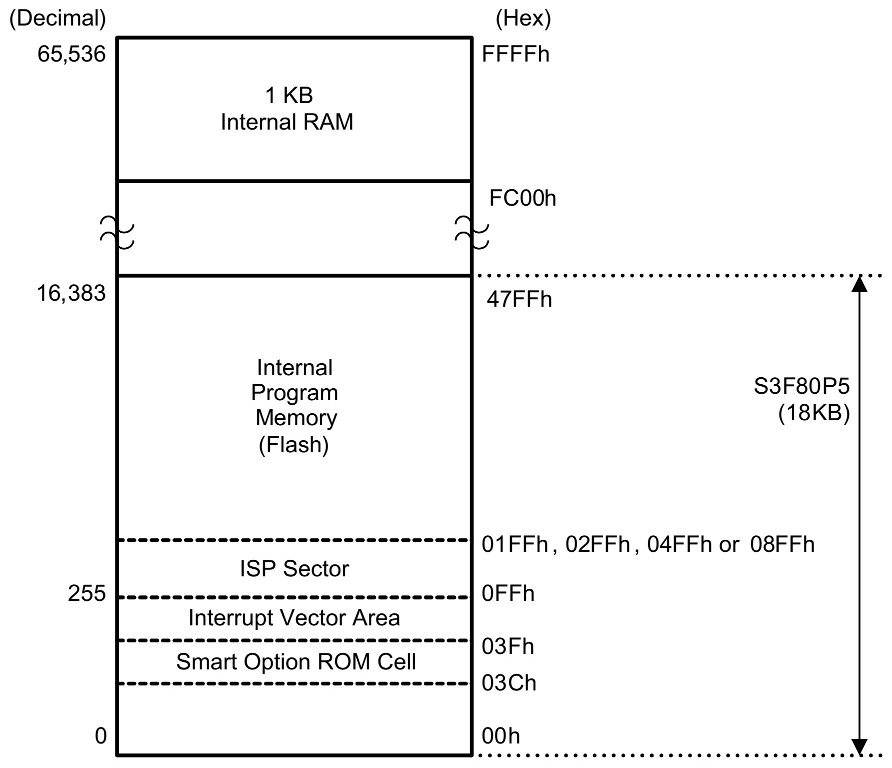


Figure 9. Program Memory Address Space

Notes:
1. In Figure 9, the size of the ISP sector can be adjusted by using the Smart Option, shown in Figure 10. According to its Smart Option setting, the ISP reset vector address can be changed to addresses `200h`, `300h`, `500h`, or `900h`.
2. The ISP sector can store onboard program software. To learn more, refer to the Embedded Flash Memory Interface chapter on page 262.

2.1.1. Smart Option

The *Smart Option*, diagrammed in Figure 10, is the ROM option for the start condition of the MCU. The ROM address used for the Smart Option ranges from `003Ch` to `003Fh`. The S3F80P5 MCU uses only addresses in the range `003Eh` to `003Fh`.

Any value can be written in the unused addresses `003Ch` and `003Dh`. The default value of the Smart Option bits in program memory is `0FFh` (i.e., normal reset vector address `100h`, ISP protection disabled). Before program memory code is executed, the Smart Option bits can be set according to the user's appropriate hardware option.

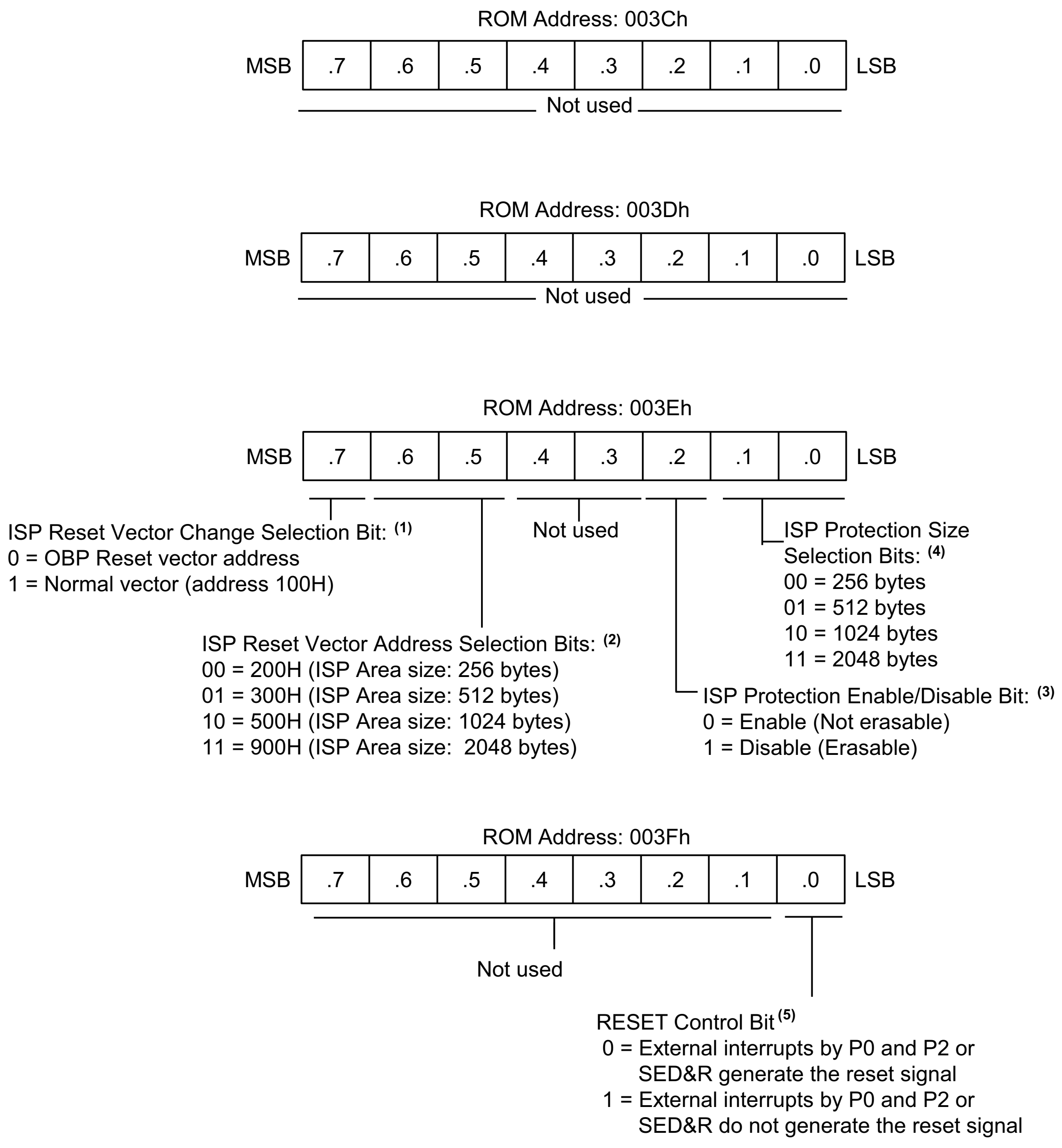

ROM Address: 003Ch
MSB
.7
.6
.5
.4
.3
.2
.1
.0
LSB
Not used
ROM Address: 003Dh
MSB
.7
.6
.5
.4
.3
.2
.1
.0
LSB
Not used
ROM Address: 003Eh
MSB
.7
.6
.5
.4
.3
.2
.1
.0
LSB
ISP Reset Vector Change Selection Bit: (1)
0 = OBP Reset vector address
1 = Normal vector (address 100H)
Not used
ISP Protection Size Selection Bits: (4)
00 = 256 bytes
01 = 512 bytes
10 = 1024 bytes
11 = 2048 bytes
ISP Reset Vector Address Selection Bits: (2)
00 = 200H (ISP Area size: 256 bytes)
01 = 300H (ISP Area size: 512 bytes)
10 = 500H (ISP Area size: 1024 bytes)
11 = 900H (ISP Area size: 2048 bytes)
ISP Protection Enable/Disable Bit: (3)
0 = Enable (Not erasable)
1 = Disable (Erasable)
ROM Address: 003Fh
MSB
.7
.6
.5
.4
.3
.2
.1
.0
LSB
Not used
RESET Control Bit (5)
0 = External interrupts by P0 and P2 or SED&R generate the reset signal
1 = External interrupts by P0 and P2 or SED&R do not generate the reset signal

Figure 10. Smart Option

Notes:

1. In Figure 10, 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 for which the user prefers to set a reset address (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` in Flash memory cannot be erased or programmed.
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 P0 falling edge input signals can wake the MCU from Stop Mode and generate this reset signal. When the RESET control bit (`3Fh.0`) is 1, the S3F80P5 MCU is only released from Stop Mode; a reset signal is not generated.

2.2. Register Architecture

In the S3F80P5 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 to two 32-byte register banks (Bank0 and Bank1), and the lower 32-byte area is a single 32-byte common area.

In the S3F80P5 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 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 register space into separately addressable areas (i.e., sets and banks) is supported by multiple addressing mode restrictions governed by the select bank instructions, SB0 and SB1.

Specific register types and the area occupied in the S3F80P5 MCU's register file are summarized in Table 2.

Table 2. S3F80P5 Register Types

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

Figure 11 shows the organization of the internal register file.

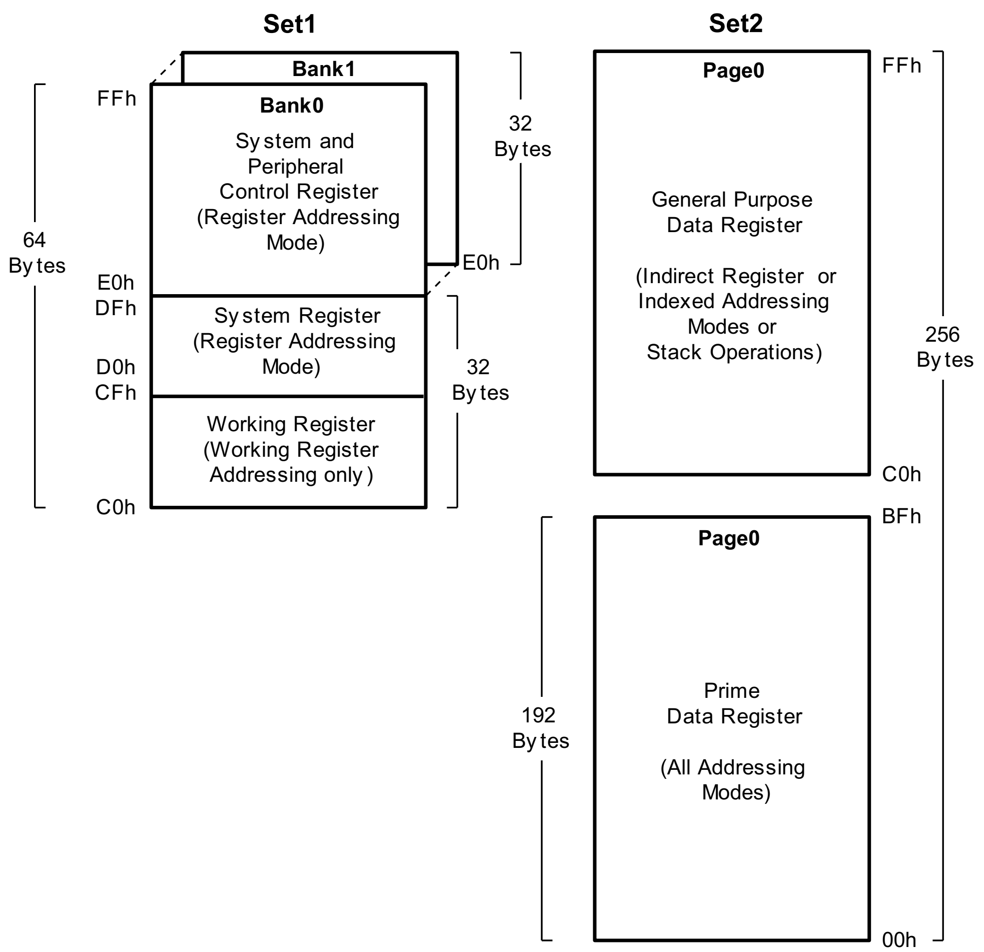

Set1
Set2
Bank1
Bank0
System and Peripheral Control Register (Register Addressing Mode)
System Register (Register Addressing Mode)
Working Register (Working Register Addressing only)
FFh
E0h
DFh
D0h
CFh
C0h
64 Bytes
32 Bytes
E0h
32 Bytes
Page0
General Purpose Data Register
(Indirect Register or Indexed Addressing Modes or Stack Operations)
FFh
C0h
BFh
Page0
Prime Data Register
(All Addressing Modes)
192 Bytes
256 Bytes
00h

Figure 11. Internal Register File Organization

2.2.1. Register Page Pointer

The S3F8 Series 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 at address `DFh`). In the S3F80P5 microcontroller, a paged register file expansion is not implemented; as a result, the register page pointer must always point to Page 0.

After a reset, the page pointer's source value (i.e., its lower nibble) and destination value (its upper nibble) are always `0000`, automatically selecting Page 0 as the source and destination page for register addressing. These Page Pointer (PP) Register settings should not be modified during normal operation.

The contents of the Register Page Pointer (PP) Register are shown in Table 3.

Table 3. 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	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	DFh							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Bit	**Description**
[7:4]	**Destination Register Page Selection Bits** 0000: Destination: Page 0. 0001–1111: Reserved; must be written to 0.
[3:0]	**Source Register Page Selection Bits** 0000: Source: Page 0. 0001–1111: Reserved; must be written to 0.
Note:	In the S3F80P5 MCU, a paged expansion of the internal register file is not implemented. For this reason, only Page 0 settings are valid. Register page pointer values for the source and destination register page are automatically set to 0000b following a hardware reset. These values should not be changed during normal operation.

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. A hardware reset operation always selects Bank0 addressing.

The upper two 32-byte Set1, Bank0 areas contain 31 mapped systems 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`). `E0h–FFh`. 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 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 35.

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 address range (`C0h–FFh`) is accessible on Page 0 in the S3F80P5 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; however, to access registers in Set2, Register Indirect Addressing Mode or Indexed Addressing Mode must be used.

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*. Prime registers can be accessed using any addressing mode; i.e., there is no addressing mode restriction for these registers, as is the case for the Set1 and Set2 registers. To learn more, see the Addressing Modes chapter on page 35.

The prime register area on Page 0 is immediately addressable following a reset.

Figure 12 shows a map of the Set1, Set2, and prime register spaces.

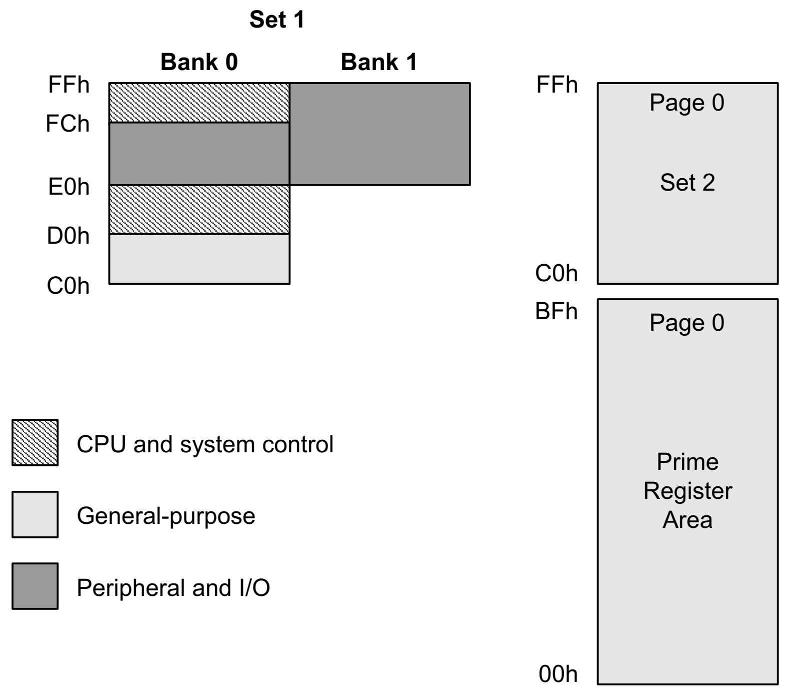


Figure 12. Set1, Set2, and Prime Area Register Map

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 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.

When using the two 8-bit register pointers, RP1 and RP0, two working register slices can be selected at any time to form a 16-byte working register block. When using these 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 document to help readers visualize the size and relative locations of selected working register spaces, as follows:

- 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 contain 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 13 illustrates the 8-byte working register areas (i.e., slices).

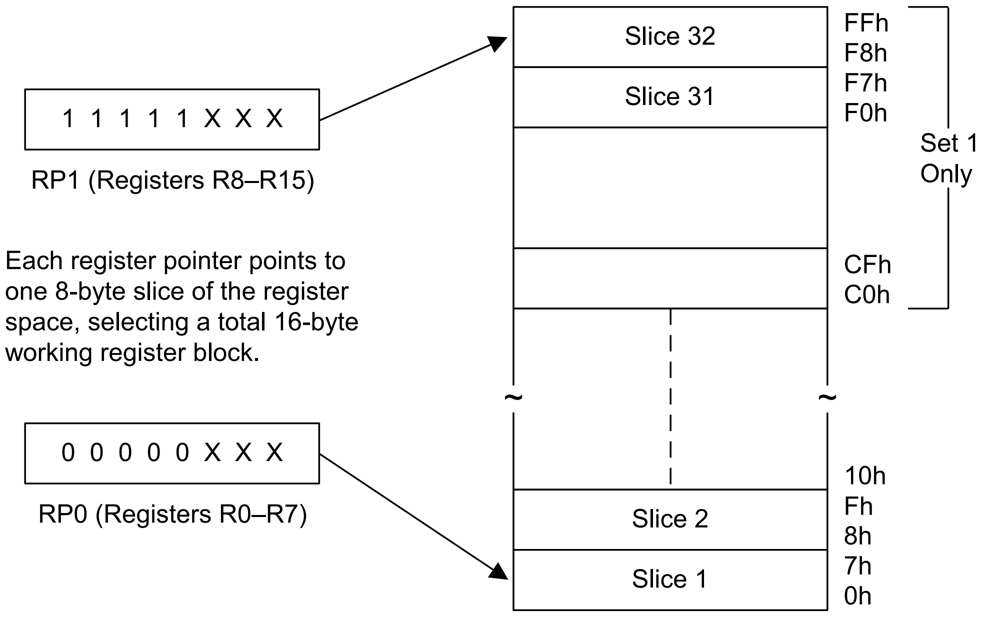


Figure 13. 8-Byte Working Register Areas

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 14 and 15.

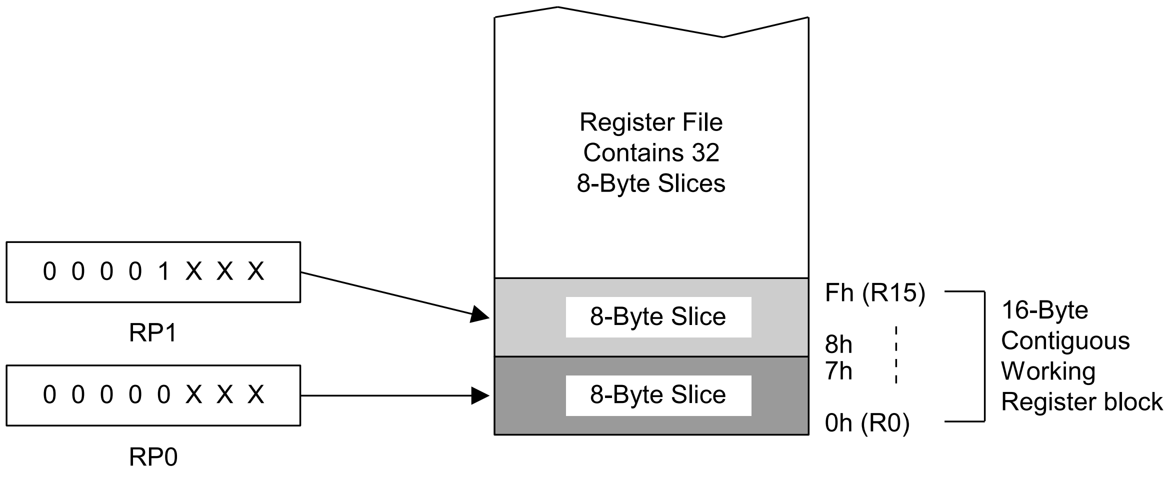


Figure 14. Contiguous 16-Byte Working Register Block

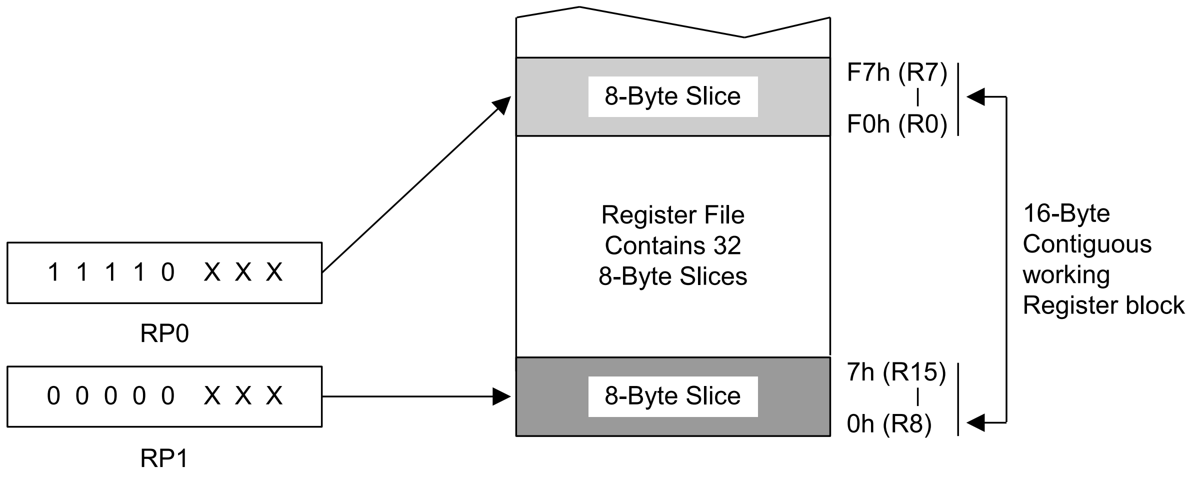


Figure 15. Noncontiguous 16-Byte Working Register Block

With working register addressing, only access those two 8-bit slices of the register file that are currently pointed to by RP0 and RP1. However, it is not possible to use the register pointers to select a working register space 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 13 on page 21. In some cases, it can be necessary to define working register areas in different (noncontiguous) areas of the register file. In Figure 14, 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.

Example 1. Setting the Register Pointers

```
SRP    #70h         ; RP0 ← 70h, RP1 ← 78h
SRP1   #48h         ; RP0 ← no change, RP1 ← 48h
SRP0   #0A0h        ; RP0 ← A0h, RP1 ← no change
CLR    RP0          ; RP0 ← 00h, RP1 ← no change
LD     RP1, #0F8h   ; RP0 ← no change, RP1 ← 0F8h
```

Example 2. 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 R0 Register (`80h`). The instruction string used in this example takes 12 bytes of instruction code, and its execution time is 36 cycles. If the register pointer is not used to calculate the sum of these registers, the following instruction sequence 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 also located in register `80h`. However, this instruction string takes 15 bytes of instruction code rather than 12 bytes, and its execution time is 50 cycles rather than 36 cycles.

2.3. Register Addressing

The S3F8 Series register architecture provides an efficient method of working register addressing that takes full advantage of shorter instruction formats to reduce execution time.

With Register (R) Addressing Mode, in which an operand value in contained in 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 16 and 17.

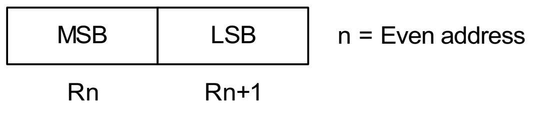
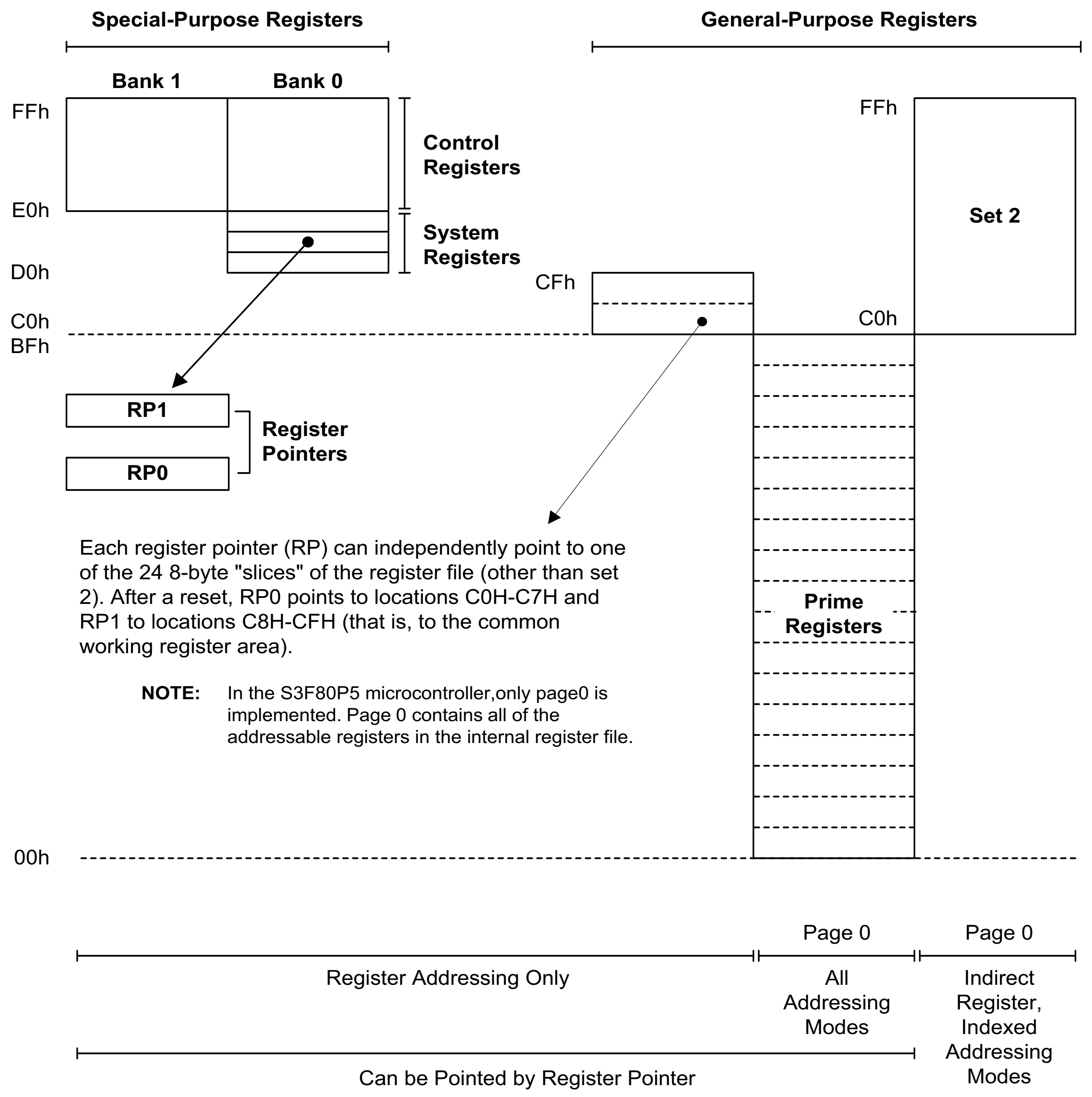


Figure 16. 16-Bit Register Pair



Each register pointer (RP) can independently point to one of the 24 8-byte "slices" of the register file (other than set 2). After a reset, RP0 points to locations C0H-C7H and RP1 to locations C8H-CFH (that is, to the common working register area).

NOTE: In the S3F80P5 microcontroller,only page0 is implemented. Page 0 contains all of the addressable registers in the internal register file.

Figure 17. Register File Addressing

2.3.1. Common Working Register Area

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

- RP0 → `C0h`–`C7h`

- RP1 → `C8h–CFh`

This 16-byte address range is called the *common working area*. Locations in this area can be used as working registers by operations that address any location on any page in the register file. Typically, these working registers serve as temporary buffers for data operations between different pages. See Figure 18.

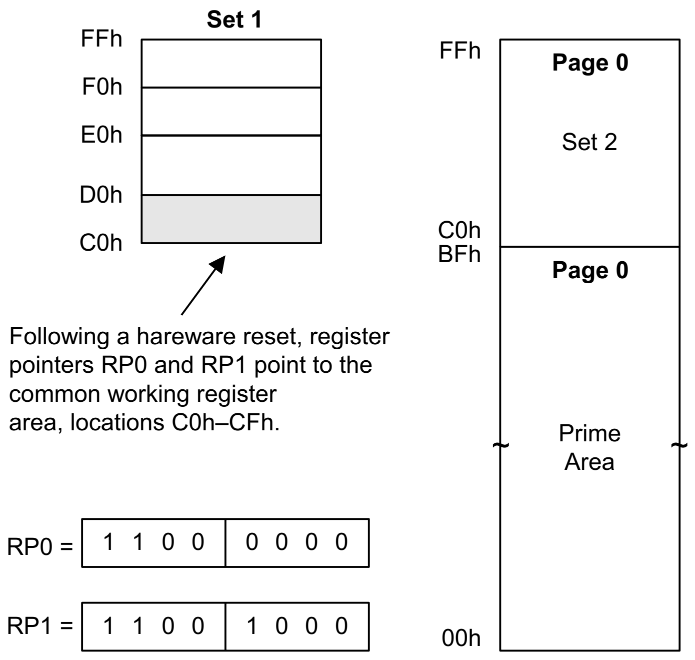

Set 1
FFh
F0h
E0h
D0h
C0h
Following a hareware reset, register pointers RP0 and RP1 point to the common working register area, locations C0h–CFh.
RP0 = 1 1 0 0 0 0 0 0
RP1 = 1 1 0 0 1 0 0 0
FFh
Page 0
Set 2
C0h
BFh
Page 0
Prime Area
00h

Figure 18. Common Working Register Area

2.3.1.1. Addressing the Common Working Register Area

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

Example 3.

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

Use working register addressing instead:

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

Example 4.

```
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 enables instructions to access working registers very efficiently using short 4-bit addresses. When an instruction addresses a location in the selected working register area, the address bits are concatenated in the following way to form a complete 8-bit address:

- The high-order bit of the 4-bit address selects one of the register pointers (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 19, 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 20 shows a typical example of 4-bit working register addressing. The high-order bit of the INC R6 instruction 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`).

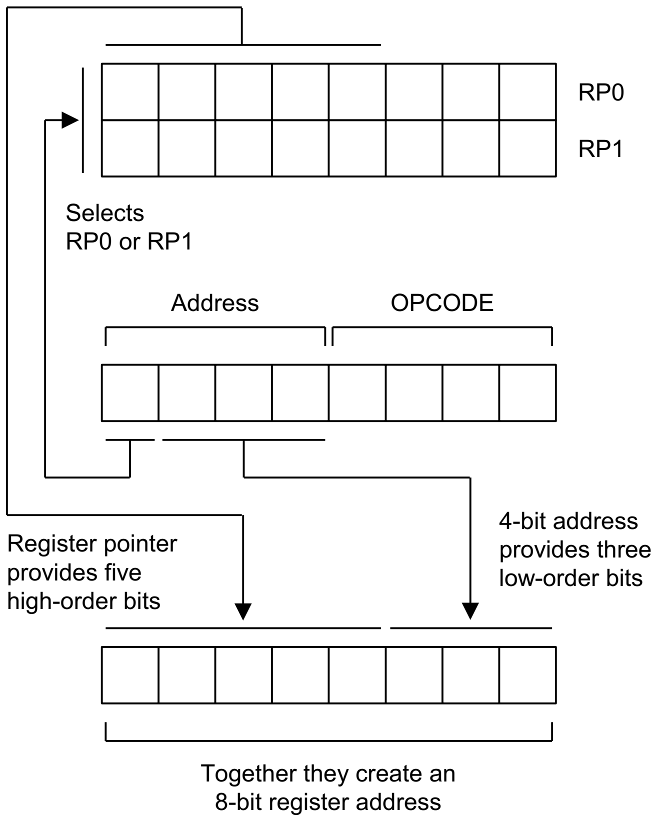


Figure 19. 4-Bit Working Register Addressing

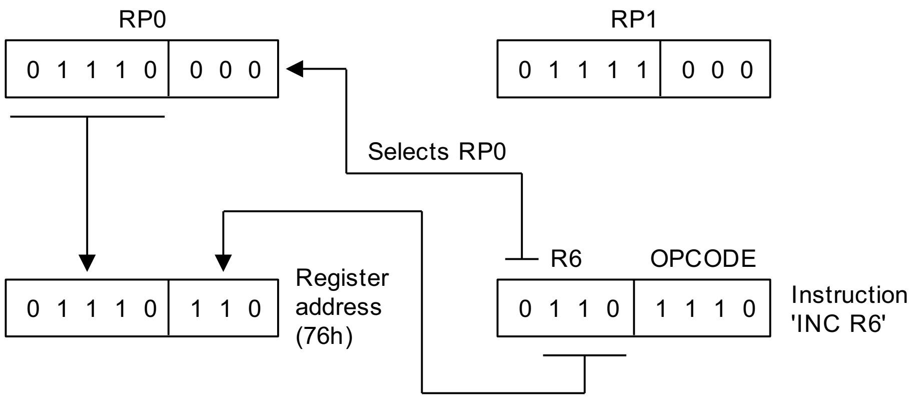


Figure 20. 4-Bit Working Register Addressing Example

2.3.3. 8-Bit Working Register Addressing

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

As shown in Figure 21, 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.

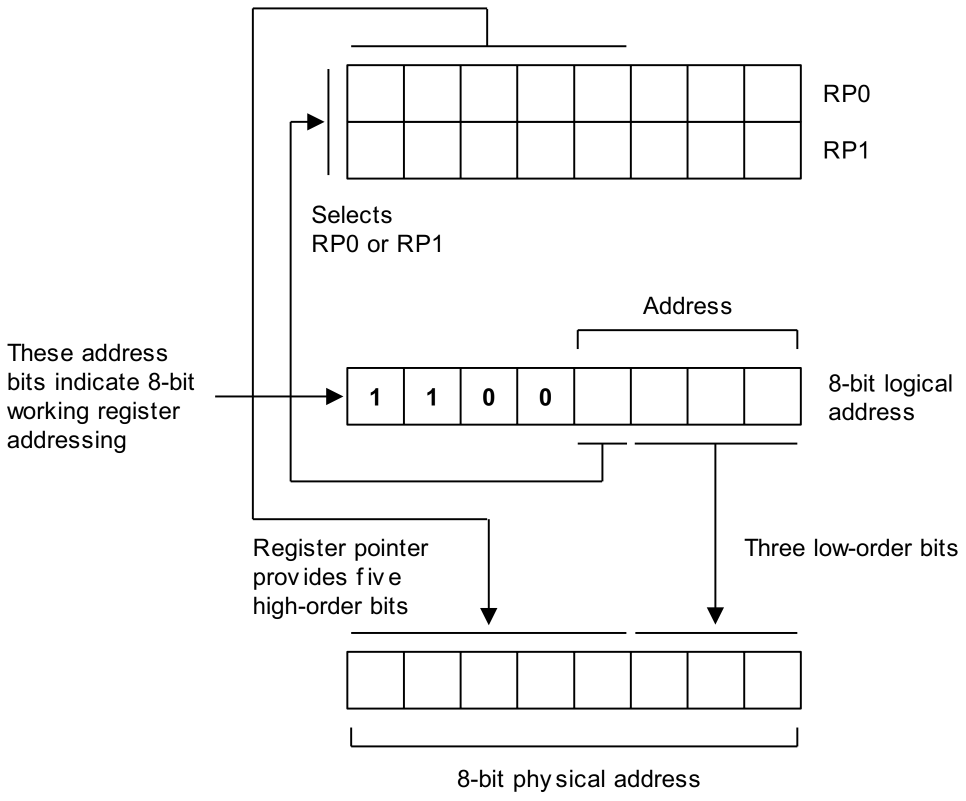

RP0
RP1
Selects RP0 or RP1
Address
These address bits indicate 8-bit working register addressing
1 1 0 0
8-bit logical address
Register pointer provides five high-order bits
Three low-order bits
8-bit physical address

Figure 21. 8-Bit Working Register Addressing

Figure 22 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`).

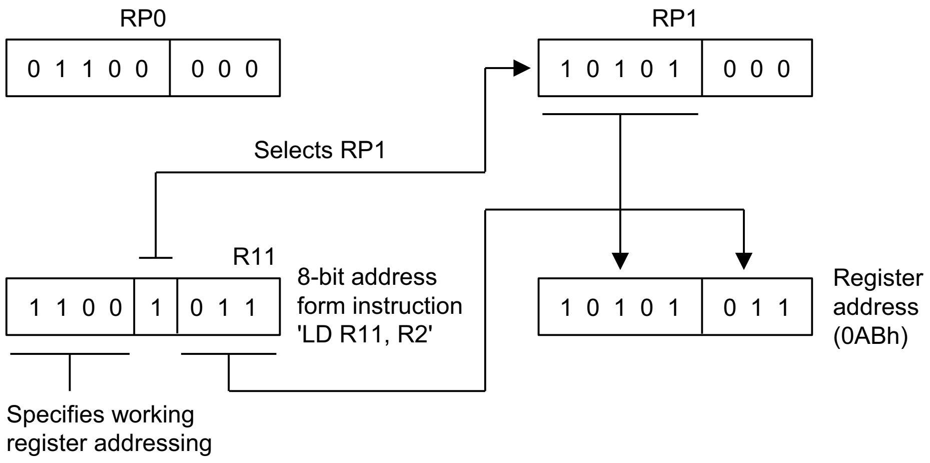


Figure 22. 8-Bit Working Register Addressing Example

2.3.4. Register Pointer Registers

The contents of the Register Pointer 0 (RP0) and Register Pointer 1 (RP1) Registers are described in Tables 4 and 5.

Table 4. Register Pointer 0 (RP0; Set1)

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							
Mode	Register Addressing Mode only							
Note: R = read only; 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 register pointers RP0 and RP1, select two 8-byte register slices at one time as the active working register space. After a reset, RP0 points to C0h in register Set1, Bank0 selecting the 8-byte working register slice C0h–C7h.
[2:0]	**Reserved; must be written to 0.**

Table 5. 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							
Mode	Register Addressing Mode only							
Note: R = read only; 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 register pointers RP0 and RP1, select two 8-byte register slices at one time as the active working register space. After a reset, RP1 points to C8h in register Set1, Bank0 selecting the 8-byte working register slice C8h–CFh.
[2:0]	**Reserved; must be written to 0**

2.4. System and User Stack

S3F8 Series 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 S3F80P5 architecture supports stack operations in the internal register file.

2.4.1. 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 23.

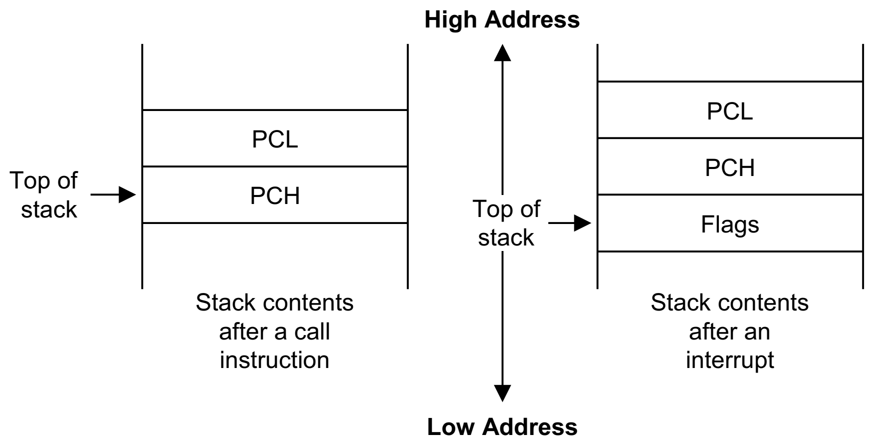

High Address
Top of stack
PCL
PCH
Stack contents after a call instruction
Top of stack
PCL
PCH
Flags
Stack contents after an interrupt
Low Address

Figure 23. Stack Operations

2.4.1.1. User-Defined Stacks

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

2.4.1.2. Stack Pointers

Register location `D9h` contains the 8-bit stack pointer (SPL) that is used for system stack operations. After a reset, the SP value is undetermined.

Because only 256KB of internal memory space is implemented in the S3F80P5 MCU, the SPL must be initialized to an 8-bit value in the range `00h–FFh`.

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

Example 5. 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
```

The contents of the Stack Pointer Low Byte (SPL) Register are shown in Table 6.

Table 6. 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							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

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

Chapter 3. Addressing Modes

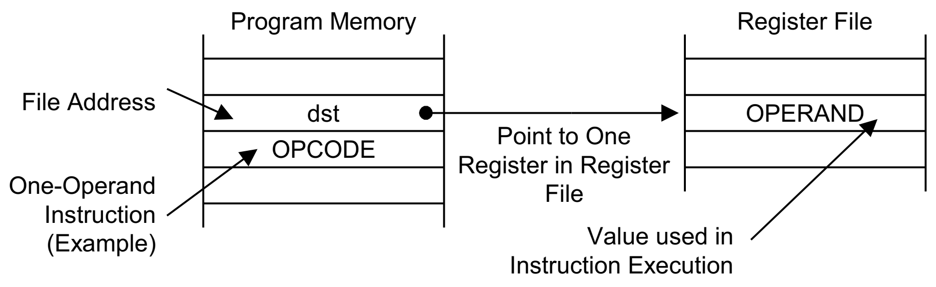
The program counter is used to fetch instructions that are stored in program memory for execution. Instructions indicate the operation to be performed and the data to be operated on. Addressing Mode is the method used to determine the location of the data operand. The operands specified in instructions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

The S3F8 Series 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

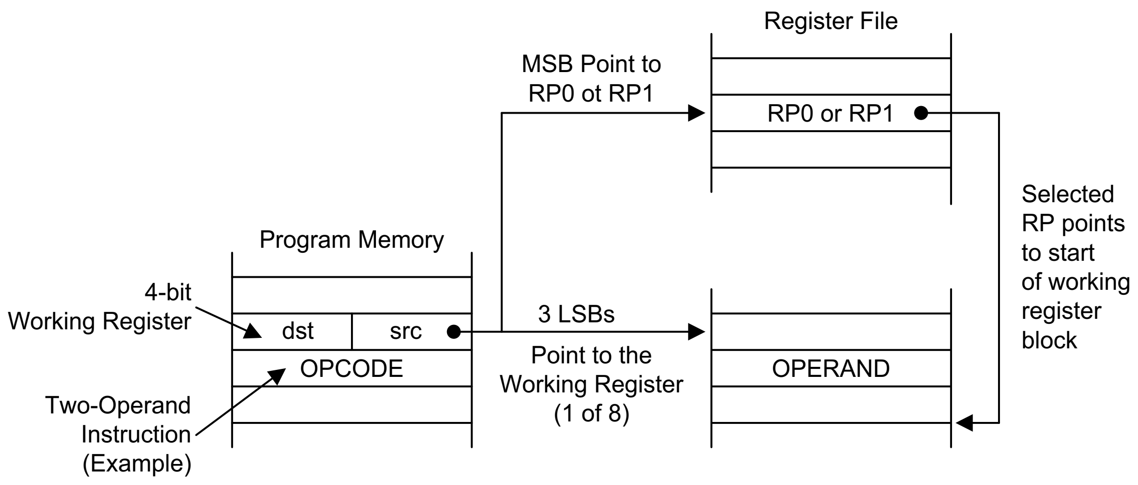
In Register Addressing Mode (R), the operand is the content of a specified register or register pair; see Figure 24. Working register addressing differs from register addressing because it uses a register pointer to specify an 8-byte working register space in the register file and an 8-bit register within that space; see Figure 25.


Program Memory
Register File
File Address
dst
OPCODE
OPERAND
Point to One Register in Register File
One-Operand Instruction (Example)
Value used in Instruction Execution

Sample Instruction:

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

Figure 24. Register Addressing


Register File
MSB Point to RP0 ot RP1
RP0 or RP1
Selected RP points to start of working register block
Program Memory
4-bit Working Register
dst
src
3 LSBs
Point to the Working Register (1 of 8)
OPCODE
OPERAND
Two-Operand Instruction (Example)

Sample Instruction:

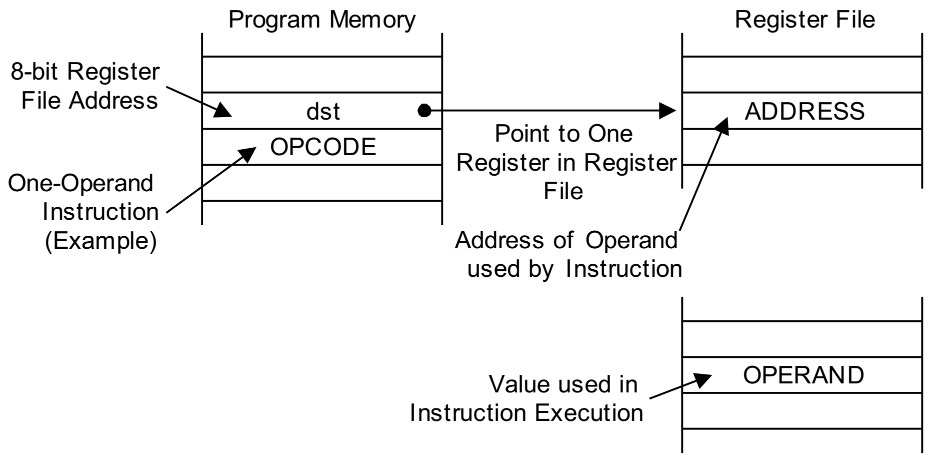
```
ADD     R1, R2        ;   Where R1 and R2 are registers in the currently
                              selected working register area.
```

Figure 25. Working Register Addressing

3.2. Indirect Register Addressing Mode

In Indirect Register (IR) Addressing Mode, the contents of the specified register or register pair represent 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 26 through 29.

Use 8-bit register can be used to indirectly address another register, and 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.


Program Memory
Register File
8-bit Register File Address
dst
OPCODE
One-Operand Instruction (Example)
Point to One Register in Register File
ADDRESS
Address of Operand used by Instruction
Value used in Instruction Execution
OPERAND

Sample Instruction:

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

Figure 26. Indirect Register Addressing to Register File

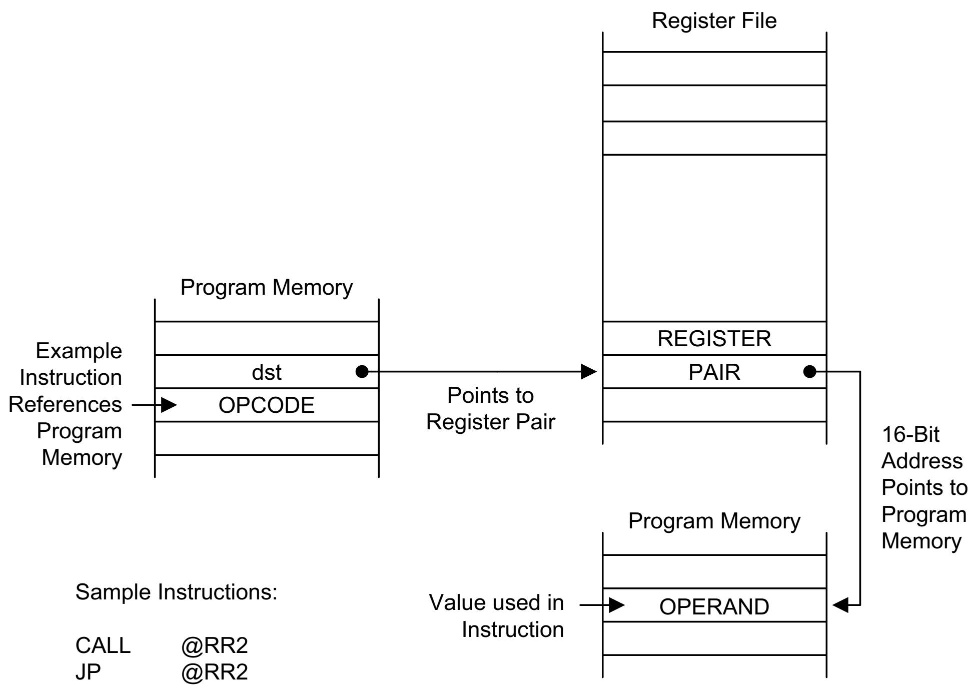


Figure 27. Indirect Register Addressing to Program Memory

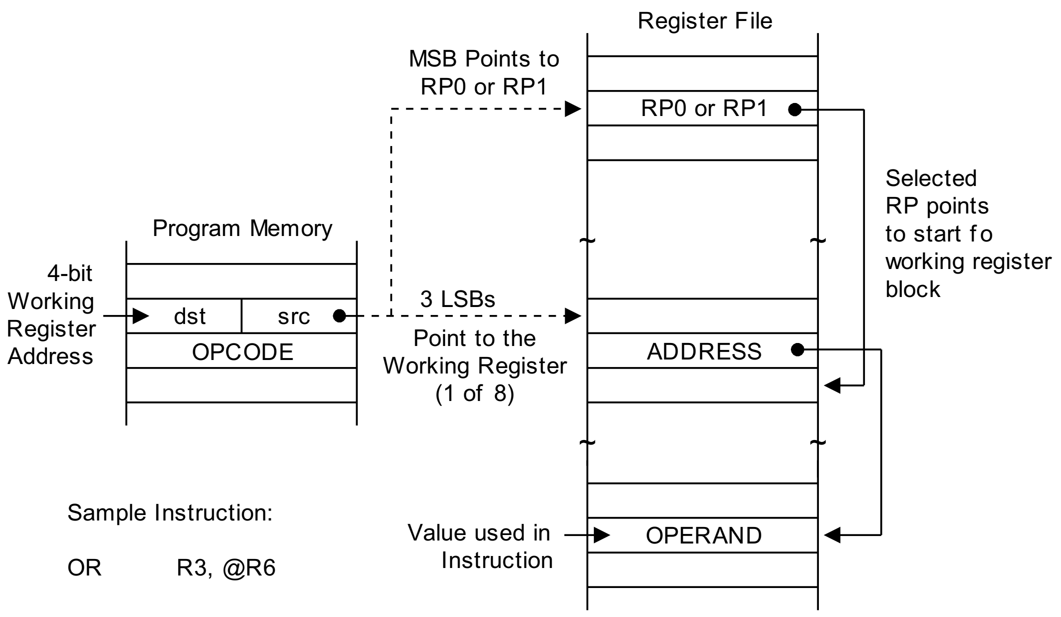
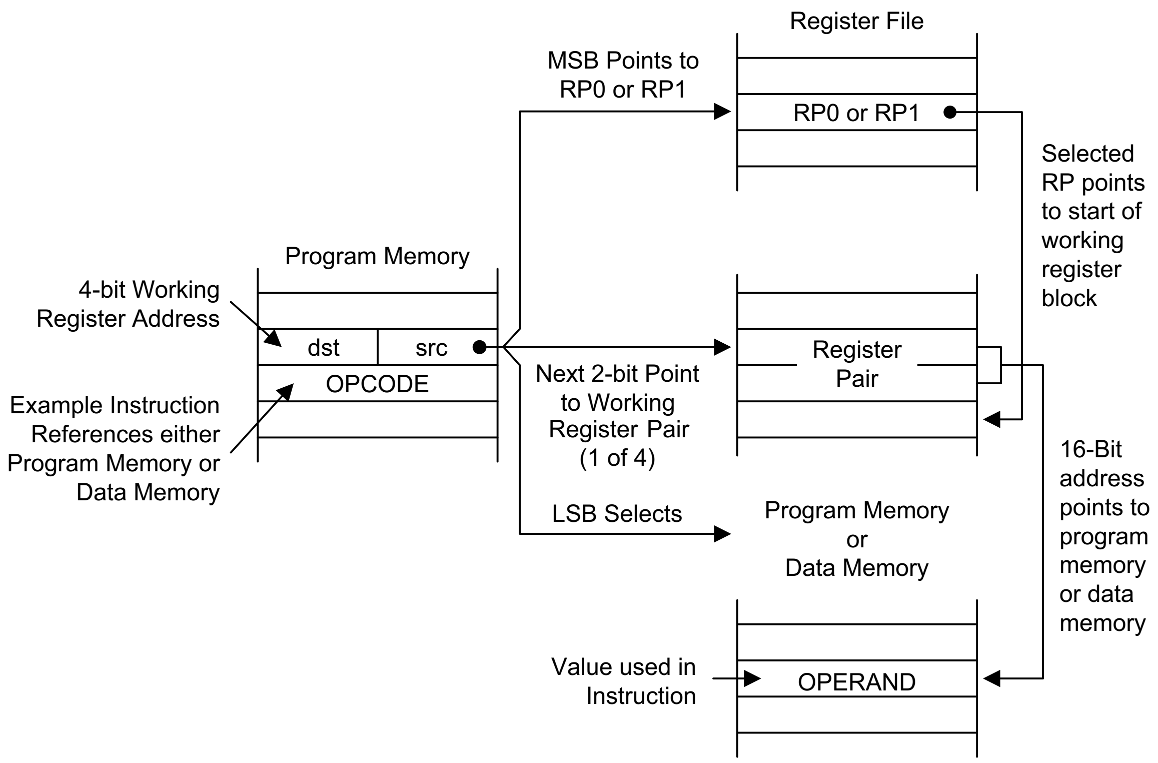

Register File
MSB Points to RP0 or RP1
RP0 or RP1
Selected RP points to start fo working register block
Program Memory
4-bit Working Register Address
dst
src
OPCODE
3 LSBs
Point to the Working Register (1 of 8)
ADDRESS
Value used in Instruction
OPERAND
Sample Instruction:
OR R3, @R6

Figure 28. Indirect Working Register Addressing to Register File


Register File
MSB Points to RP0 or RP1
RP0 or RP1
Selected RP points to start of working register block
Program Memory
4-bit Working Register Address
dst
src
OPCODE
Example Instruction References either Program Memory or Data Memory
Next 2-bit Point to Working Register Pair (1 of 4)
Register Pair
16-Bit address points to program memory or data memory
LSB Selects
Program Memory or Data Memory
Value used in Instruction
OPERAND

Sample Instructions:

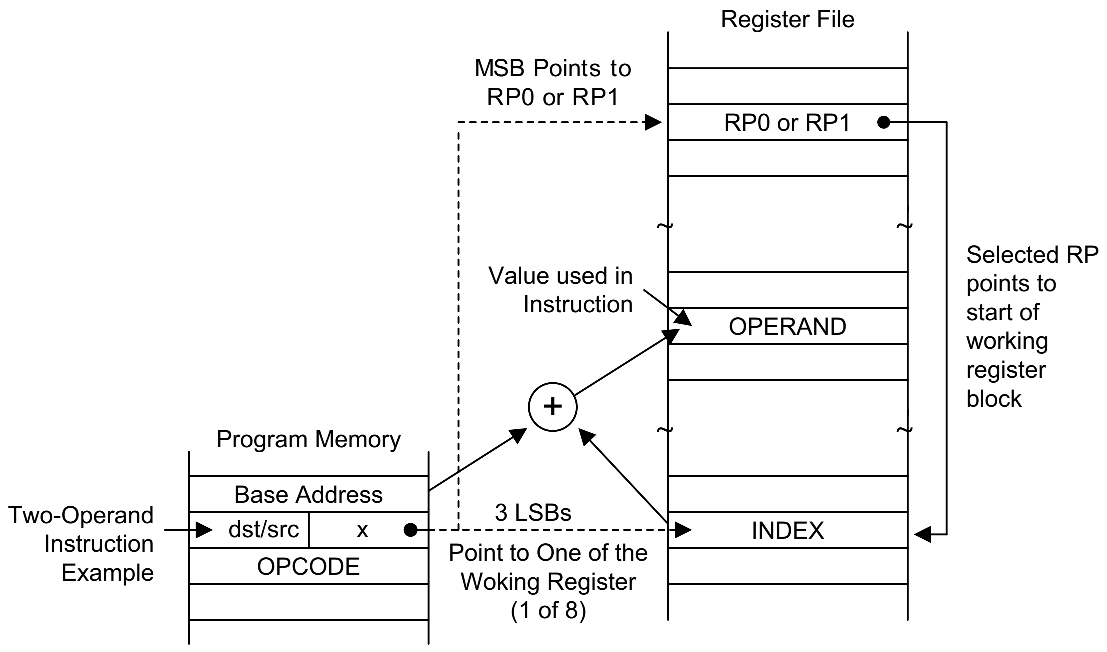
LCD	R5,@RR6	; Program memory access
LDE	R3,@RR14	; External data memory access
LDE	@RR4, R8	; External data memory access

Figure 29. Indirect Working Register Addressing to Program or Data Memory

Note: The LDE command is included in Figure 29 for illustrative purposes only. It is not available for the S3F80P5 MCU because an external interface is not implemented.

3.3. Indexed Addressing Mode

Indexed (X) Addressing Mode adds an offset value to a base address during instruction execution to calculate the effective operand address; see Figure 30. Use Indexed Addressing Mode to access locations in the internal register file or in external memory. However, locations `C0h`–`FFh` in Set1 cannot be accessed using Indexed Addressing Mode.

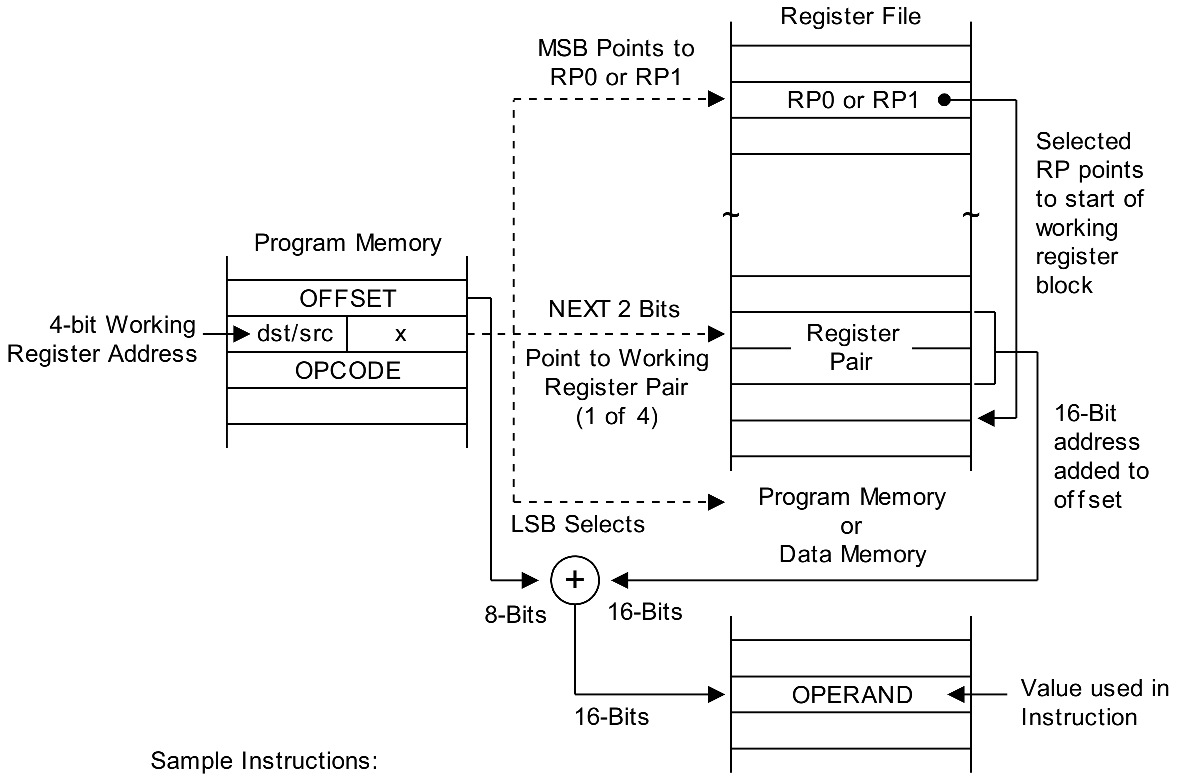

Register File
MSB Points to RP0 or RP1
RP0 or RP1
Selected RP points to start of working register block
Value used in Instruction
OPERAND
+
Program Memory
Base Address
Two-Operand Instruction Example
dst/src
x
3 LSBs
INDEX
OPCODE
Point to One of the Woking Register (1 of 8)

Sample Instruction:

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

Figure 30. Indexed Addressing to Register File

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 31.


Register File
MSB Points to RP0 or RP1
RP0 or RP1
Selected RP points to start of working register block
Program Memory
OFFSET
4-bit Working Register Address
dst/src
x
OPCODE
NEXT 2 Bits
Point to Working Register Pair (1 of 4)
Register Pair
16-Bit address added to offset
LSB Selects
Program Memory or Data Memory
+
8-Bits
16-Bits
16-Bits
OPERAND
Value used in Instruction

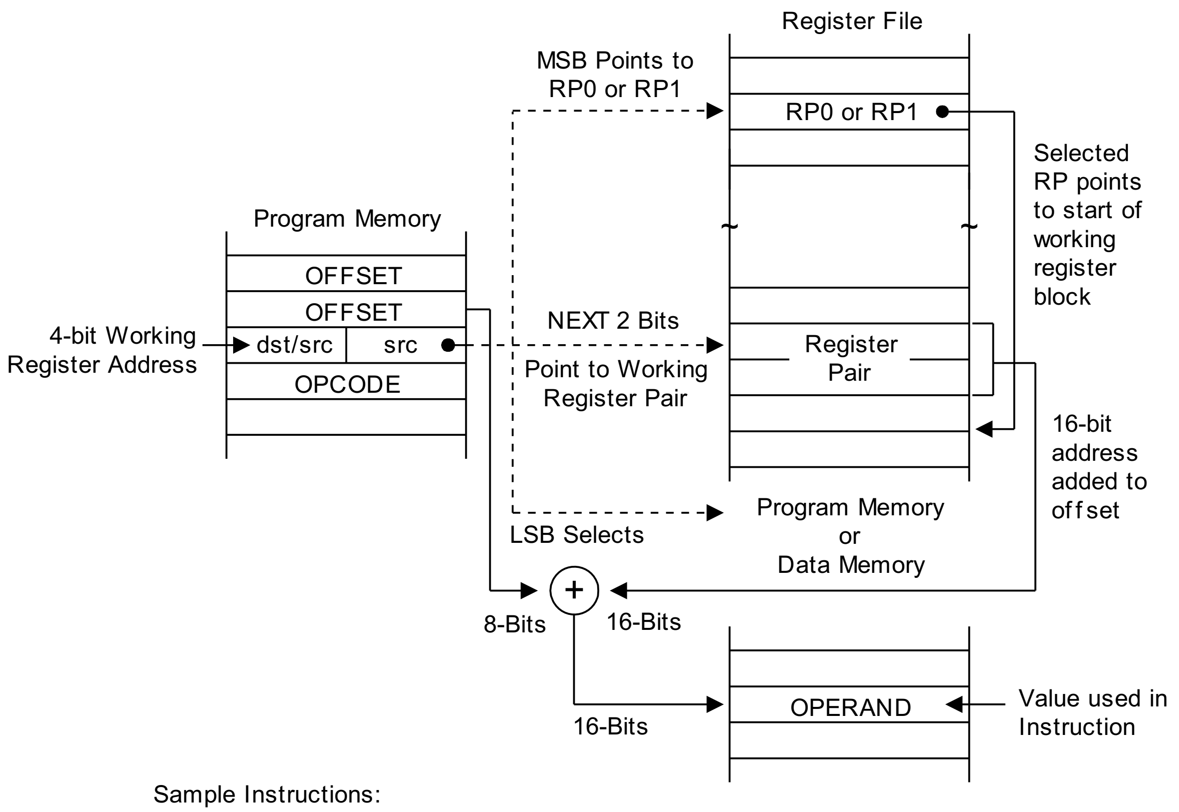
Sample Instructions:

```
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 program memory is accessed.
```

Figure 31. Indexed Addressing to Program or Data Memory with Short Offset

Note: The LDE command is included in Figure 31 for illustrative purposes only. It is not available for the S3F80P5 MCU because an external interface is not implemented.

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 32.


Register File
MSB Points to RP0 or RP1
RP0 or RP1
Selected RP points to start of working register block
Program Memory
OFFSET
OFFSET
4-bit Working Register Address
dst/src
src
OPCODE
NEXT 2 Bits
Point to Working Register Pair
Register Pair
16-bit address added to offset
LSB Selects
Program Memory or Data Memory
+
8-Bits
16-Bits
16-Bits
OPERAND
Value used in Instruction

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 program memory is accessed.

Figure 32. Indexed Addressing to Program or Data Memory

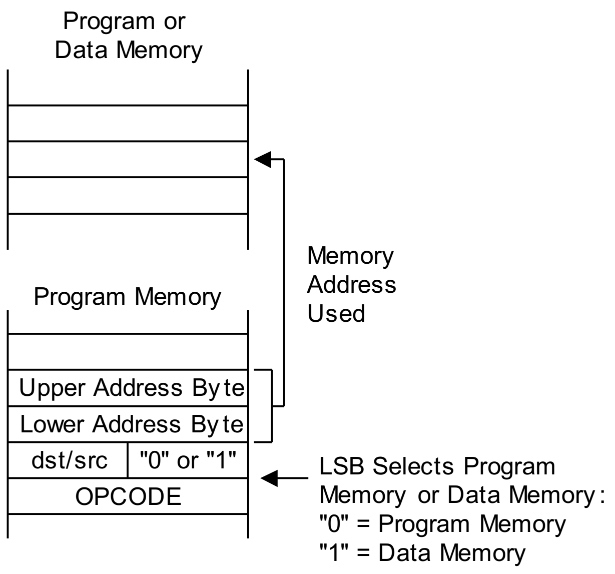
Note: The LDE command is included in Figure 32 for illustrative purposes only. It is not available for the S3F80P5 MCU because an external interface is not implemented.

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).

3.4. Direct Address Mode

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

The LDC and LDE instructions can use Direct Address Mode to specify the source or destination address for Load operations to program memory (LDC) or to external data memory (LDE), if implemented. See Figures 33 and 34.

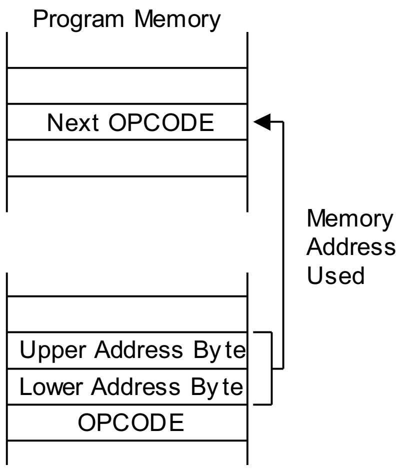


Sample Instructions:

```
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 program memory is accessed.
```

Figure 33. Direct Addressing for Load Instructions

Note: The LDE command is included in Figure 33 for illustrative purposes only. It is not available for the S3F80P5 MCU because an external interface is not implemented.



Sample Instructions:

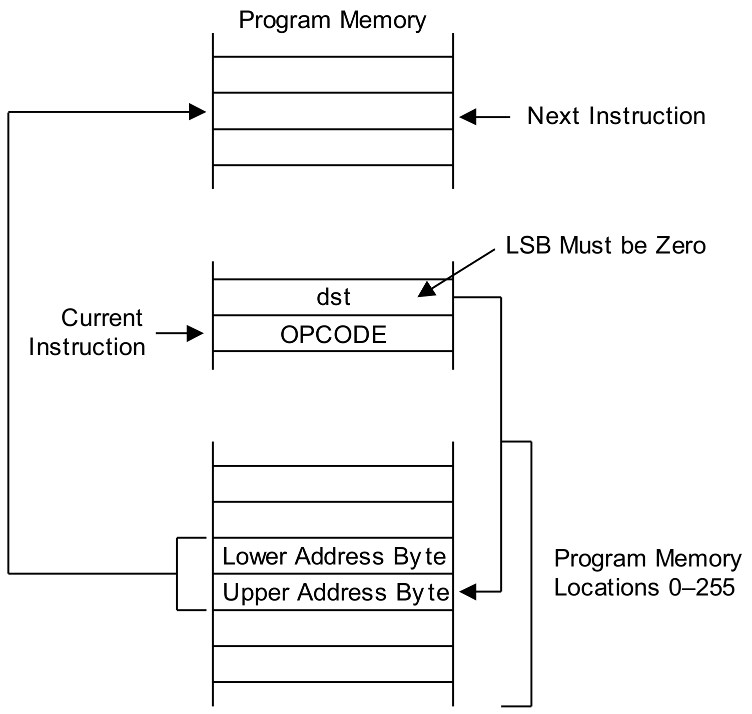
```
JP      C,JOB1       ;   Where JOB1 is a 16-bit immediate address
CALL    DISPLAY      ;   Where DISPLAY is a 16-bit immediate address
```

Figure 34. 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 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 be all zeros. See Figure 35.



Sample Instruction:

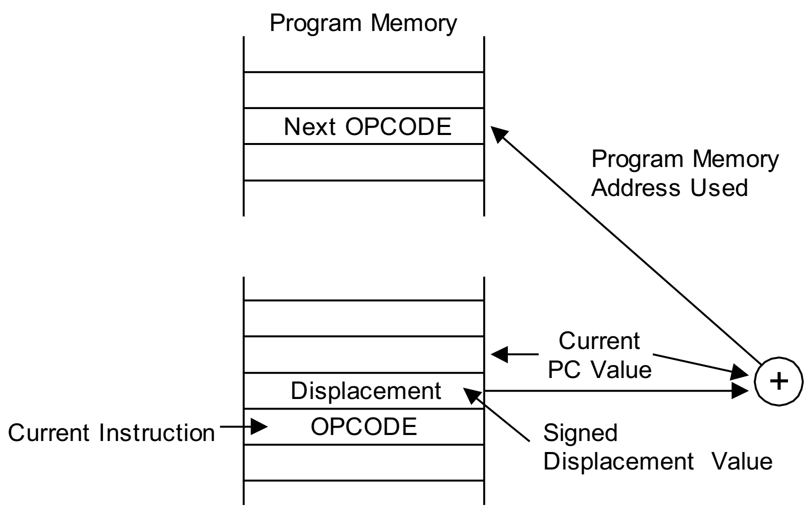
```
CALL    #40h        ;  The 16-bit value in program memory addresses 40h
                       and 41h is the subroutine start address.
```

Figure 35. 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. This displacement value is then added to the current PC value. The result is the address of the next instruction to be executed. Before this addition occurs, the PC contains the address of the instruction immediately following the current instruction.

Several program control instructions use Relative Address Mode to perform conditional jumps. The instructions that support RA addressing are BTJRF, BTJRT, DJNZ, CPIJE, CPIJNE, and JR. See Figure 36.



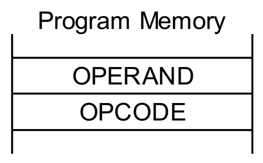
Sample Instructions:

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

Figure 36. Relative Addressing

3.7. Immediate Mode

In Immediate (IM) Addressing 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 37.



(The operand value is in the instruction)

Sample Instruction:

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

Figure 37. Immediate Addressing

Chapter 4. Control Registers

This chapter describes the S3F80P5 MCU's control registers. The hardware reset value for each mapped register tabulated in this chapter is described in the Reset and Power-Down chapter on page 189. Data and counter registers are not described in this chapter; information about all registers used by a specific peripheral is presented in corresponding chapters.

Table 7 identifies the names, mnemonics, decimal and hex equivalents, and read/write settings of the Page 8 registers; click to the linked page to review the contents of each.

Table 7. Set1, Bank0 Mapped Registers

Register Name	Page #	Mnemonic	Decimal	Hex	R/W
Timer 0 Counter	–	T0CNT	208	D0h	R*
Timer 0 Data	236	T0DATA	209	D1h	R/W
Timer 0 Control	235	T0CON	210	D2h	R/W
Basic Timer Control	231	BTCON	211	D3h	R/W
Clock Control	187	CLKCON	212	D4h	R/W
System Flags	82	FLAGS	213	D5h	R/W
Register Pointer 0	31	RP0	214	D6h	R/W
Register Pointer 1	32	RP1	215	D7h	R/W
Location D8h is not mapped.					
Stack Pointer Low Byte	34	SPL	217	D9h	R/W
Instruction Pointer High Byte	70	IPH	218	DAh	R/W
Instruction Pointer Low Byte	70	IPL	219	DBh	R/W
Interrupt Request	65	IRQ	220	DCh	R
Interrupt Mask	62	IMR	221	DDh	R/W
System Mode	60	SYM	222	DEh	R/W
Register Page Pointer	18	PP	223	DEh	R/W
Port 0 Data	211	P0	224	E0h	R/W
Port 1 Data	211	P1	225	E1h	R/W
Port 2 Data	211	P2	226	E2h	R/W
Port 3 Data	211	P3	227	E3h	R/W
Location E4h is reserved.					
Port 2 Interrupt Enable	224	P2INT	229	E5h	R/W
Port 2 Interrupt Pending	225	P2PND	230	E6h	R/W
Note: A read-only register cannot be used as a destination for the OR, AND, LD, or LDB instructions.					

Table 7. Set1, Bank0 Mapped Registers (Continued)

Register Name	Page #	Mnemonic	Decimal	Hex	R/W
Port 0 Pull-Up Resistor Enable	218	P0PUR	231	E7h	R/W
Port 0 Control High Byte	218	P0CONH	232	E8h	R/W
Port 0 Control Low Byte	214	P0CONL	233	E9h	R/W
Port 1 Control High Byte	219	P1CONH	234	EAh	R/W
Port 1 Control Low Byte	221	P1CONL	235	EBh	R/W
Location ECh is reserved.					
Port 2 Control Low Byte	223	P2CONL	237	EDh	R/W
Port 2 Pull-Up Enable	225	P2PUR	238	EEh	R/W
Port 3 Control	226	P3CON	239	EFh	R/W
Location F0h is reserved.					
Port 0 Interrupt Enable	215	P0INT	241	F1h	R/W
Port 0 Interrupt Pending	216	P0PND	242	F2h	R/W
Counter A Control	249	CACON	243	F3h	R/W
Counter A Data High Byte	250	CADATAH	244	F4h	R/W
Counter A Data Low Byte	250	CADATAL	245	F5h	R/W
Timer 1 Counter High Byte	246	T1CNTH	246	F6h	R
Timer 1 Counter Low Byte	246	T1CNTL	247	F7h	R
Timer 1 Data High Byte	246	T1DATAH	248	F8h	R/W
Timer 1 Data Low Byte	246	T1DATAL	249	F9h	R/W
Timer 1 Control	245	T1CON	250	FAh	R/W
Stop Control	208	STOPCON	251	FBh	R/W
Location FCh is not mapped.					
Basic Timer Counter	–	BTCNT	253	FDh	R
External Memory Timing	73	EMT	254	FEh	R/W
Interrupt Priority	64	IPR	255	FFh	R/W

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

Table 8 identifies the names, mnemonics, decimal and hex equivalents, and read/write settings of the Set1, Bank1 registers; click to the linked page to review the contents of each.

Table 8. Set1, Bank1 Mapped Registers

Register Name	Page #	Mnemonic	Decimal	Hex	R/W
LVD Control	282	LVDCON	224	E0h	R/W
Location E1h is reserved.					
Location E2h is reserved.					
Location E3h is reserved.					
Timer 2 Counter High Byte	261	T2CNTH	228	E4h	R
Timer 2 Counter Low Byte	261	T2CNTL	229	E5h	R
Timer 2 Data High Byte	261	T2DATAH	230	E6h	R/W
Timer 2 Data Low Byte	261	T2DATAL	231	E7h	R/W
Timer 2 Control	260	T2CON	232	E8h	R/W
Location E9h is reserved.					
Location EAh is reserved.					
Location EBh is reserved.					
Flash Memory Sector Address High Byte	268	FMSECH	236	ECh	R/W
Flash Memory Sector Address Low Byte	268	FMSECL	237	EDh	R/W
Flash Memory User Programming Enable	267	FMUSR	238	EEh	R/W
Flash Memory Control	266	FMCON	239	EFh	R/W
Reset Indicating	208	RESETID	240	F0h	R/W
LVD Flag Selection	282	LVDSEL	241	F1h	R/W
Port 1 Output Mode Pull-Up Enable	222	P1OUTPU	242	F2h	R/W
Port 1 Output Mode Selection	224	P2OUTMD	243	F3h	R/W
Port 1 Output Mode Pull-Up Enable	228	P3OUTPU	244	F4h	R/W
Locations F5h–FFh are not mapped.					
Note: A read-only register cannot be used as a destination for the OR, AND, LD, or LDB instructions.					

Chapter 5. Interrupt Structure

The S3F8 Series interrupt structure contains three basic components: levels, vectors, and sources. The SAM8 RC 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 (however, IRQ5 is reserved on the S3F80PB MCU), also called *Level 0–Level 7* (however, Level 5 is reserved on the S3F80PB MCU). 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 S3F80P5 MCU's interrupt structure recognizes eight interrupt levels.

The interrupt level numbers 0 through 7 do not necessarily indicate the relative priority of the levels; they are 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 (IPR) Register. 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 can contain no vector address assigned at all. The maximum number of vectors that can be supported for any level is 128. (The actual number of vectors used for S3F8 Series devices is always much smaller.) If an interrupt level contains more than one vector address, the vector priorities are set in hardware. The S3F80P5 MCU uses fourteen vectors; one vector address is 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 S3F80P5 MCU's interrupt structure, there are 17 possible interrupt sources.

When a service routine starts, the respective pending bit is either cleared automatically by hardware or 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 S3F8 Series interrupt structure described previously – 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, as follows.

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

In the S3F80P5 microcontroller, all three interrupt types are implemented, as indicated in Figure 38.

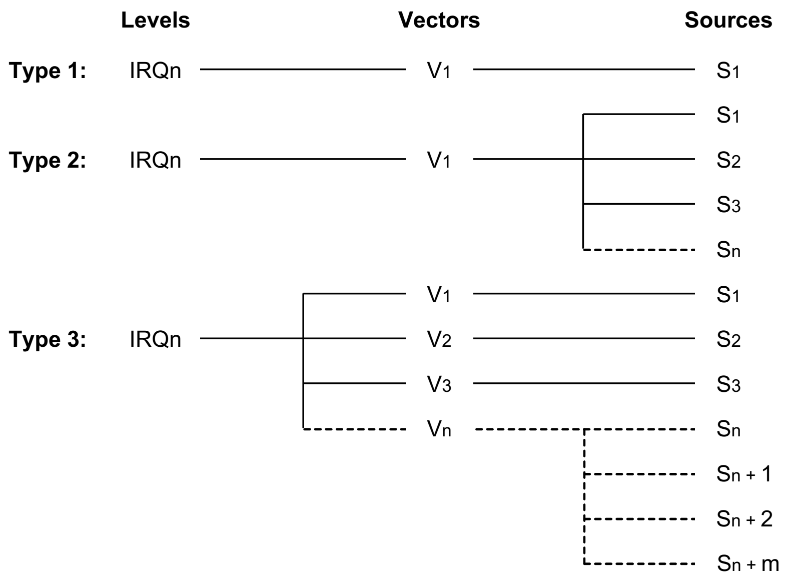


Figure 38. S3F8 Series Interrupt Types

Note: In Figure 38, the number of S_n and V_n values is expandable.

The S3F80P5 microcontroller supports seventeen interrupt sources. Thirteen of these interrupt sources contain a corresponding interrupt vector address; the remaining four interrupt sources share one vector address. Seven interrupt levels are recognized by the CPU in this device-specific interrupt structure, as shown in Figure 39.

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).

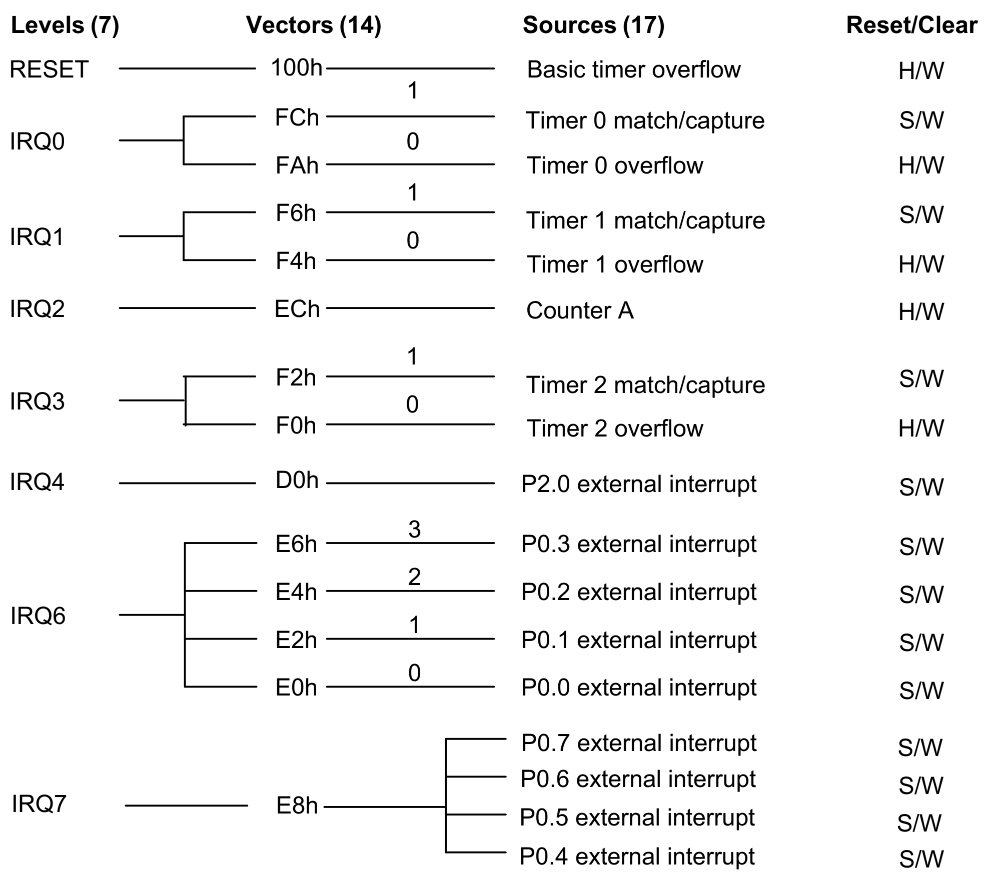


Figure 39. S3F80P5 Interrupt Structure

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 the 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.

Note: in Figure 39, the reset interrupt vector address (i.e., basic timer overflow) can be varied by means of the Smart Option.

5.5. Interrupt Vector Addresses

All interrupt vector addresses for the S3F80P5 interrupt structure are stored in the vector address area of internal ROM in the address range `00h–FFFh`; see Figure 40.

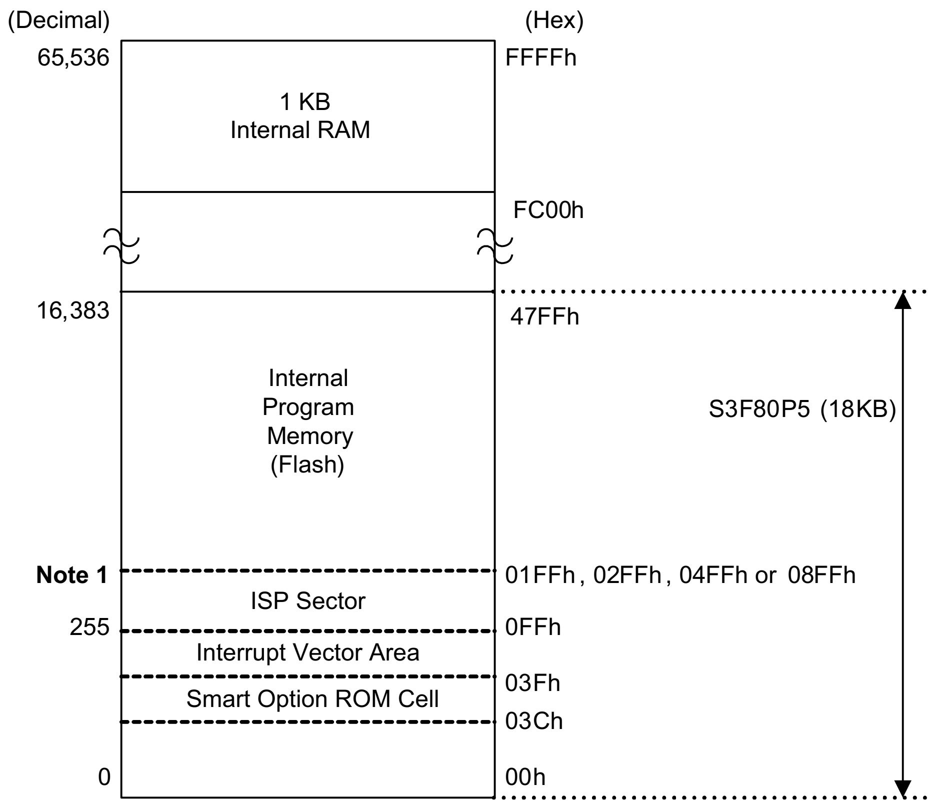


Figure 40. ROM Vector Address Area

Note: in Figure 40, the size of the ISP sector can be varied by means of the Smart Option; see Figure 10 on page 15. 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`).

Unused locations in the vector address area should be allocated as normal program memory. However, be careful not to overwrite any of the stored vector addresses, which are listed in Table 9.

Table 9. Interrupt Vectors

Vector Address			Request		Reset/Clear	
Decimal Value	**Hex Value**	**Interrupt Source**	**Interrupt Level**	**Priority in Level**	**Hardware**	**Software**
256	100h	Basic timer overflow	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		√
208	D0h	P2.0 external interrupt	IRQ4	–		

Note: Interrupt priorities are identified in inverse order: 0 is the highest priority, 1 is the next-highest priority, etc. If two or more interrupts within the same level contend, the interrupt with the lowest vector address usually has priority over an interrupt with a higher vector address. Priorities within a presented level are fixed in hardware. Resets of the basic timer overflow or POR interrupt vector address can be changed via the Smart Option.

The program reset address in the ROM is `0100h`. This reset address can be changed by means of the Smart Option; see Figure 10 on page 15.

5.6. Enable/Disable Interrupt Instructions

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

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

During normal operation, the Disable Interrupt (DI) instruction can be executed at any time to globally disable interrupt processing. The EI and DI instructions change the value of bit 0 in the SYM Register.

5.7. System-Level Interrupt Control Registers

In addition to 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 10.

Table 10. 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. IRQ5 is reserved in the S3F80P5 MCU.
Interrupt Priority Register	IPR	R/W	Controls the relative processing priorities of the interrupt levels. The eight levels of the S3F80P5 MCU are organized into three groups: A, B, and C. Group A is IRQ0 and IRQ1, Group B is IRQ2, IRQ3, and IRQ4, and Group C is IRQ5, IRQ6, and IRQ7.

Table 10. Interrupt Control Register Overview

Control Register	Mnemonic	R/W	Function Description
Interrupt Request Register	IRQ	R	This register contains a request pending bit for each interrupt level.
System Mode Register	SYM	R/W	This register enables/disables fast interrupt processing, dynamic global interrupt processing, and external interface control (an external memory interface is not implemented in the S3F80P5 microcontroller).

5.8. Interrupt Processing Control Points

Interrupt processing can be controlled in either of two ways: either 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 41 diagrams the functions of these combined interrupt processes.

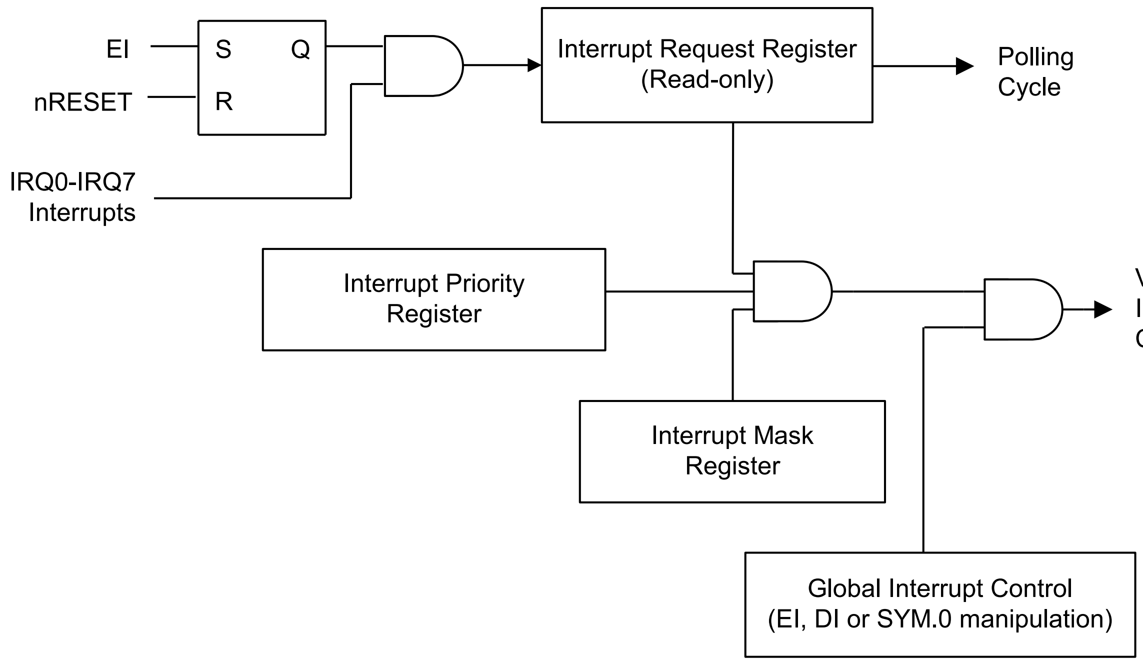

EI
nRESET
S
Q
R
IRQ0-IRQ7 Interrupts
Interrupt Request Register (Read-only)
Polling Cycle
Interrupt Priority Register
Interrupt Mask Register
Vector Interrupt Cycle
Global Interrupt Control (EI, DI or SYM.0 manipulation)

Figure 41. Interrupt Function Diagram

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

5.9. Peripheral Interrupt Control Registers

For each interrupt source, there are one or more corresponding peripheral control registers that control the interrupts generated by their corresponding peripherals. Because the Timer 0, Timer 1, and Timer 2 overflow interrupts are cleared by hardware, the T0CON, T1CON, and T2CON registers only control enable and disable functions. These three registers contain enable/disable and pending bits for the Timer 0, Timer 1, and Timer 2 match/capture interrupts, respectively; see Table 11.

Table 11. Vectored Interrupt Source Control and Data Registers

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

Note: If an interrupt is unmasked in the IMR Register (as determined by its Enable Interrupt level), the pending and enable bits of the interrupt should be written after a DI instruction is executed.

5.10. System Mode Register

The System Mode (SYM) Register (DEh, Set1, Bank0), shown in Table 12, is used to globally enable and disable interrupt processing and to control fast interrupt processing. A reset clears SYM.7, SYM.1, and SYM.0 to 0. The 3-bit value for fast interrupt level selection, SYM.4–SYM.2, is undetermined after reset. SYM.6 and SYM.5 are not used.

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

Table 12. 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							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Bit	Description
[7]	**Tri-State External Interface Control Bit**[1] 0: Normal operation (i.e., tri-state operation is disabled). 1: Set external interface lines to high impedance to enable tri-state operation.
[6:5]	**Reserved**[2] **Not used in the S3F80P5 MCU.**
[4:2]	**Fast Interrupt Level Selection Bits**[3] 000: IRQ0. 001: IRQ1. 010: IRQ2. 011: IRQ3. 100: IRQ4. 101: Not used in the S3F80P5 MCU. 110: IRQ6. 111: IRQ7.
[1]	**Fast Interrupt Enable Bit**[4] 0: Disable fast interrupt processing. 1: Enable fast interrupt processing.
[0]	**Global Interrupt Enable Bit**[5] 0: Disable global interrupt processing. 1: Enable global interrupt processing.

Notes:
1. Because an external interface is not implemented for the S3F80P5, SYM.7 must always be 0.
2. Although the SYM Register is not used, SYM.5 should always be 0. If a 1 is accidentally written to this bit during normal operation, a system malfunction can occur.
3. Only one interrupt level at a time can be selected 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, global interrupt processing must be enabled by executing an EI instruction (not by writing a 1 to SYM.0).

5.11. Interrupt Mask Register

The Interrupt Mask (IMR) Register (`DDh`, Set1, Bank0), shown in Table 13, 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 (masked). When setting a level's IMR bit to 1, interrupt processing for the level is enabled (not masked).

The IMR Register is mapped to register location `DDh` in Set1, Bank0. Bit values can be read and written by instructions using Register Addressing Mode.

Before changing values in the IMR Register, first disable all interrupts with the DI instruction.

Table 13. Interrupt Mask Register (IMR; Set1)

Bit	7	6	5	4	3	2	1	0
Reset	x	x	–	x	x	x	x	x
R/W	R/W	R/W	–	R/W	R/W	R/W	R/W	R/W
Address	DDh							
Mode	Register Addressing Mode only							

Note: R = read only; 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]	Reserved Not used in the S3F80P5 MCU.
[4]	**Interrupt Level 4 (IRQ4) Enable Bit; External Interrupt P2.0** 0: Disable (Mask). 1: Enable (Unmask).
[3]	**Interrupt Level 3 (IRQ3) Enable Bit; Timer 2 Match 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 or Overflow** 0: Disable (Mask). 1: Enable (Unmask).
[0]	**Interrupt Level 0 (IRQ0) Enable Bit; Timer 0 Match or Overflow** 0: Disable (Mask). 1: Enable (Unmask).

5.12. 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 therefore 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 two sources belong to the same interrupt level, the source with the lower vector address usually has 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, listed below, are used only by IPR logic for the IPR Register priority definitions; see Figure 42.

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

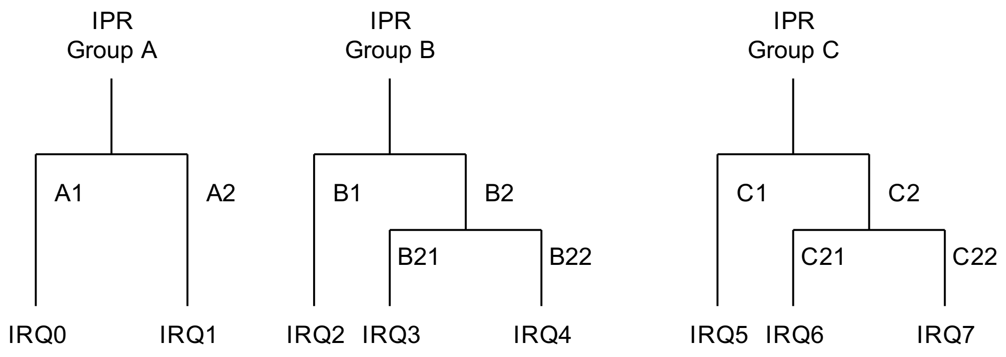


Figure 42. Interrupt Request Priority Groups

As Figure 42 shows, 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:

- Interrupt Group B features a subgroup to provide an additional priority relationship between 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 (IPR) Register are shown in Table 14.

Table 14. 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	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FFh							
Mode	Register Addressing Mode only							

Note: R = read only; 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]	**Reserved Not used in the S3F80P5 MCU.**
[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 S3F80P5 interrupt structure uses the seven levels IRQ0–IRQ7; however, IRQ5 is reserved.

5.13. Interrupt Request Register

Bit values in the Interrupt Request (IRQ) Register, shown in Table 15, 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 (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 is executed (i.e., if global interrupt processing is disabled). If an interrupt occurs while the interrupt structure is disabled, the CPU can not service it. Detect the interrupt request by polling the IRQ Register. In this way, it is possible to determine which events occurred while the interrupt structure is globally disabled.

Table 15. Interrupt Request Register (IPQ; 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							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

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]	**Reserved Not used in the S3F80P5 MCU.**
[4]	**Level 4 (IRQ4) Request Pending Bit; External Interrupt P2.0** 0: Not pending. 1: Pending.
[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.

5.14. 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.

5.14.1. 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 S3F80P5 MCU's interrupt structure, the Timer A overflow interrupt (IRQ0), the Timer B match interrupt (IRQ1), the Timer C overflow interrupt (IRQ2), and the Timer D0/D1 overflow interrupt (IRQ3) belong to this category of interrupts in which a pending condition is cleared automatically by hardware.

5.14.2. 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; i.e., a 0 must be written to the corresponding pending bit location in the source's mode or control register.

In the S3F80P5 MCU's interrupt structure, pending conditions for all interrupt sources, with the exception of the Timer 0 and Timer 1 overflow interrupts and the Counter A borrow interrupt, must be cleared by the interrupt service routine.

The contents of the Interrupt Pending (INTPND) Register are described in Table 16. Before changing values in the IMR Register, first disable all interrupts with the DI instruction.

Table 16. Interrupt Pending Register (INTPND; Set1, Bank0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	–	–	x	x	x	x	x	x
R/W	–	–	R/W	R/W	R/W	R/W	R/W	R/W
Address	DDh							
Mode	Register Addressing Mode only							

Note: R = read only; R/W = read/write.

Bit	Description
[7:6]	**Reserved; must be written to 0**
[5]	**Timer D1 Match/Capture Interrupt Pending Bit** 0: No interrupt pending (when read), clear pending bit (when write). 1: Interrupt is pending (when read).
[4]	**Timer D1 Overflow Interrupt Pending Bit** 0: No interrupt pending (when read), clear pending bit (when write). 1: Interrupt is pending (when read).
[3]	**Timer D0 Match/Capture Interrupt Pending Bit** 0: No interrupt pending (when read), clear pending bit (when write). 1: Interrupt is pending (when read).
[2]	**Timer D0 Overflow Interrupt Pending Bit** 0: No interrupt pending (when read), clear pending bit (when write). 1: Interrupt is pending (when read).
[1]	**Timer A Match/Capture Interrupt Pending Bit** 0: No interrupt pending (when read), clear pending bit (when write). 1: Interrupt is pending (when read).
[0]	**Timer A Overflow Interrupt Pending Bit** 0: No interrupt pending (when read), clear pending bit (when write). 1: Interrupt is pending (when read).

5.15. 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 the 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 (by hardware or by software).
7. The CPU continues polling for interrupt requests.

5.16. Interrupt Service Routines

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

- Interrupt processing must be globally enabled (i.e., EI, SYM.0 = 1)
- The interrupt level must be enabled (IMR Register)
- 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)

When 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 (SYM.0) Register 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). This IRET restores the PC and status flags and sets SYM.0 to 1, allowing the CPU to process the next interrupt request.

5.17. Generating Interrupt Vector Addresses

The interrupt vector area in ROM (`00h–FFh`) contains the addresses of interrupt service routines that correspond to each level in the interrupt structure. Vectored interrupt processing observes the following 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`.

5.18. Nesting of Vectored Interrupts

It is possible to nest a higher-priority interrupt request while a lower-priority request is being serviced, as shown in the following sequence.

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 is processed if an 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, the above procedure can be simplified to some extent.

5.19. Instruction Pointer Register

The Instruction Pointer (IP) Register is used by all S3F8 Series microcontrollers to control the optional high-speed interrupt processing feature called *fast interrupts*. The IP consists of register pair `DAh` and `DBh`. The IP Register names are IPH (high byte, IP15–IP8) and IPL (low byte, IP7–IP0).

The contents of the Instruction Pointer High Byte (IPH) and Instruction Pointer Low Byte (IPL) Registers are described in Tables 17 and 18.

Table 17. Instruction Pointer High Byte (IPH; Set1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	x	x	x	x	x	x	x	x
R/W	R/W							
Address	DAh							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Bit	**Description**
[7:0]	**Instruction Pointer Address High Byte** The high-byte instruction pointer value is the upper eight bits of the 16-bit instruction pointer address (IP15–IP8). The lower byte of the IP address is located in the IPL Register (DBh).

Table 18. Instruction Pointer Low Byte (IPL; Set1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	x	x	x	x	x	x	x	x
R/W	R/W							
Address	DBh							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Bit	**Description**
[7:0]	**Instruction Pointer Address Low Byte** The low-byte instruction pointer value is the lower eight bits of the 16-bit instruction pointer address (IP7–IP0). The upper byte of the IP address is located in the IPH Register (DAh).

5.20. Fast Interrupt Processing

This feature lets you specify that an interrupt within a presented level be completed in approximately 6 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), and
- When a fast interrupt occurs, the contents of the Flags Register are stored in an unmapped, dedicated register called *Flags'* (a.k.a. Flags prime).

Note: For the S3F80P5 microcontroller, the service routine for any one of the seven interrupt levels: IRQ0–IRQ7(IRQ5 is reserved), can be selected for fast interrupt processing.

5.20.1. Procedure for Initiating Fast Interrupts

To initiate fast interrupt processing, observe the following procedure:

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

5.20.2. 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′ (a.k.a. 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′ is automatically copied back to the Flags Register.

7. The fast interrupt status bit in Flags is cleared automatically.

5.20.3. Programming Guidelines

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

5.21. External Memory Timing Register

The External Memory Timing (EMT) Register is described in Table 19.

Table 19. 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	R/W	R/W	R/W	R/W	R/W	R/W	–
Address	FEh							
Mode	Register Addressing Mode only							

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** Not used in the S3F80P5 MCU; must be set to 0.

Note: The EMT Register is not used in the S3F80P5 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 6. Op Code Maps

Tables 20 and 21 provide quick reference op code maps to addresses 0–7 and 8–F, respectively.

Table 20. Op Code Quick Reference (0–7)

Op Code Map									
Lower Nibble (Hex)									
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Upper Nibble (Hex)	0	DEC R1	DEC IR1	ADD r1, r2	ADD r1, Ir2	ADD R2, R1	ADD IR2, R1	ADD R1, IM	BOR r0–Rb
	1	RLC R1	RLC IR1	ADC r1, r2	ADC r1, Ir2	ADC R2, R1	ADC IR2, R1	ADC R1, IM	BCP r1.b, R2
	2	INC R1	INC IR1	SUB r1, r2	SUB r1, Ir2	SUB R2, R1	SUB IR2, R1	SUB R1, IM	BXOR r0–Rb
	3	JP IRR1	SRP/0/1 IM	SBC r1, r2	SBC r1, Ir2	SBC R2, R1	SBC IR2, R1	SBC R1, IM	BTJR r2.b, RA
	4	DA R1	DA IR1	OR r1, r2	OR r1, Ir2	OR R2, R1	OR IR2, R1	OR R1, IM	LDB r0–Rb
	5	POP R1	POP IR1	AND r1, r2	AND r1, Ir2	AND R2, R1	AND IR2, R1	AND R1, IM	BITC r1.b
	6	COM R1	COM IR1	TCM r1, r2	TCM r1, Ir2	TCM R2, R1	TCM IR2, R1	TCM R1, IM	BAND r0–Rb
	7	PUSH R2	PUSH IR2	TM r1, r2	TM r1, Ir2	TM R2, R1	TM IR2, R1	TM R1, IM	BIT r1.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
	9	RL R1	RL IR1	POPUD IR2, R1	POPUI IR2, R1	DIV R2, RR1	DIV IR2, RR1	DIV IM, RR1	LD r2, x, r1
	A	INCW RR1	INCW IR1	CP r1, r2	CP r1, Ir2	CP R2, R1	CP IR2, R1	CP R1, IM	LDC r1, Irr2, xL
	B	CLR R1	CLR IR1	XOR r1, r2	XOR r1, Ir2	XOR R2, R1	XOR IR2, R1	XOR R1, IM	LDC r2, Irr2, xL
	C	RRC R1	RRC IR1	CPIJE Ir, r2, RA	LDC r1, Irr2	LDW RR2, RR1	LDW IR2, RR1	LDW RR1, IML	LD r1, Ir2
	D	SRA R1	SRA IR1	CPIJNE Ir, r2, RA	LDC r2, Irr1	CALL IA1		LD IR1, IM	LD Ir1, r2
	E	RR R1	RR IR1	LDCD r1, Irr2	LDCI r1, Irr2	LD R2, R1	LD R2, IR1	LD R1, IM	LDC r1, Irr2, xs
	F	SWAP R1	SWAP IR1	LDCPD r2, Irr1	LDCPI r2, Irr1	CALL IRR1	LD IR2, R1	CALL DA1	LDC r2, Irr1, xs

Table 21. Op Code Quick Reference (8–F)

		Op Code Map							
		Lower Nibble (Hex)							
		8	**9**	**A**	**B**	**C**	**D**	**E**	**F**
Upper Nibble (Hex)	0	LD r1,R2	LD r2,R1	DJNZ r1,RA	JR cc, RA	LD r1,IM	JP cc, DA	INC r1	NEXT
	1	↓	↓	↓	↓	↓	↓	↓	ENTER
	2								EXIT
	3								WFI
	4								SB0
	5								SB1
	6								IDLE
	7	↓	↓	↓	↓	↓	↓	↓	STOP
	8								DI
	9								EI
	A								RET
	B								IRET
	C								RCF
	D	↓	↓	↓	↓	↓	↓	↓	SCF
	E								CCF
	F	LD r1, R2	LD r2, R1	DJNZ r1, RA	JR cc, RA	LD r1, IM	JP cc, DA	INC r1	NOP

6.1. Condition Codes

The op code of a conditional jump always contains a 4-bit field called the *condition code* (cc) that 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 22.

Table 22. Condition Codes[1]

Mnemonic	Binary	Description	Flags Set
F	0000	Always false	–
T	1000	Always true	–
C[2]	0111	Carry	C = 1
NC[2]	1111	No carry	C = 0
Z[2]	0110	Zero	Z = 1
NZ[2]	1110	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[2]	0110	Equal	Z = 1
NE[2]	1110	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[2]	1111	Unsigned greater than or equal	C = 0
ULT[2]	0111	Unsigned less than	C = 1
UGT	1011	Unsigned greater than	(C = 0 AND Z = 0) = 1
ULE	0011	Unsigned less than or equal	(C OR Z) = 1

Note:
1. For operations involving unsigned numbers, the special condition codes UGE, ULT, UGT, and ULE must be used.
2. 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.

Chapter 7. 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 SAM8 instruction set include:

- A full complement of 8-bit arithmetic and logic operations, including multiply and divide
- I/O control/data registers are mapped directly into the register file; no special I/O instructions are required
- 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

7.1. 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 (right-most) bit.

7.2. 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 can be 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.

7.3. 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 35.

7.4. Instruction Summary

All instructions are summarized by type, operand, and description in Table 23.

Table 23. Instruction Group Summary

Mnemonic	Operands	Instruction
Load Instructions		
CLR	dst	Clear
LD	dst,src	Load
LDB	dst,src	Load bit
LDE	dst,src	Load external data memory
LDC	dst,src	Load program memory
LDED	dst,src	Load external data memory and decrement
LDCD	dst,src	Load program memory and decrement
LDEI	dst,src	Load external data memory and increment
LDCI	dst,src	Load program memory and increment
LDEPD	dst,src	Load external data memory with predecrement
LDCPD	dst,src	Load program memory with predecrement
LDEPI	dst,src	Load external data memory with preincrement
LDCPI	dst,src	Load program memory with preincrement
LDW	dst,src	Load word
POP	dst	Pop from stack
POPUD	dst,src	Pop user stack (decrementing)
POPUI	dst,src	Pop user stack (incrementing)
PUSH	src	Push to stack
PUSHUD	dst,src	Push user stack (decrementing)
PUSHUI	dst,src	Push user stack (incrementing)
Arithmetic Instructions		
ADC	dst,src	Add with carry
ADD	dst,src	Add
CP	dst,src	Compare
DA	dst	Decimal adjust
DEC	dst	Decrement
DECW	dst	Decrement word

Table 23. Instruction Group Summary (Continued)

Mnemonic	Operands	Instruction
DIV	dst,src	Divide
Arithmetic Instructions (continued)		
INC	dst	Increment
INCW	dst	Increment word
MULT	dst,src	Multiply
SBC	dst,src	Subtract with carry
SUB	dst,src	Subtract
Logic Instructions		
AND	dst,src	Logical AND
COM	dst	Complement
OR	dst,src	Logical OR
XOR	dst,src	Logical exclusive OR
Program Control Instructions		
BTJRF	dst,src	Bit test and jump relative on false
BTJRT	dst,src	Bit test and jump relative on true
CALL	dst	Call procedure
CPIJE	dst,src	Compare, increment and jump on equal
CPIJNE	dst,src	Compare, increment and jump on non-equal
DJNZ	r,dst	Decrement register and jump on non-zero
ENTER	–	
EXIT	–	
IRET	–	
JP	cc,dst	Jump on condition code
JP	dst	Jump unconditional
JR	cc,dst	Jump relative on condition code
NEXT	–	Next
RET	–	Return
WFI	–	Wait for interrupt
Bit Manipulation Instructions		
BAND	dst,src	Bit AND
BCP	dst,src	Bit compare
BITC	dst	Bit complement
BITR	dst	Bit reset

Table 23. Instruction Group Summary (Continued)

Mnemonic	Operands	Instruction
BITS	dst	Bit set
Bit Manipulation Instructions (continued)		
BOR	dst,src	Bit OR
BXOR	dst,src	Bit XOR
TCM	dst,src	Test complement under mask
TM	dst,src	Test under mask
Rotate and Shift Instructions		
RL	dst	Rotate left
RLC	dst	Rotate left through carry
RR	dst	Rotate right
RRC	dst	Rotate right through carry
SRA	dst	Shift right arithmetic
SWAP	dst	Swap nibbles
CPU Control Instructions		
CCF	–	Complement carry flag
DI	–	Disable interrupts
EI	–	Enable interrupts
IDLE	–	Enter Idle mode
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

7.5. Flags Register

The Flags (FLAGS) Register, shown in Table 24, 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. The Flags Register can be set or reset by instructions as long as its outcome does not affect the flags (e.g., a 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 writes will occur to the Flags Register, thereby producing an unpredictable result.

Table 24. 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							
Mode	Register Addressing Mode only							
Note: R = read only; 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 $\geq$ –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 25 describes each of the flags managed by the Flags Register.

Table 25. Flag Descriptions

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 final 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 the 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 of the BCD operations, to ensure that a subsequent decimal adjust operation can execute correctly. The DA bit is not usually accessed by programmers, and cannot be used as a test condition.
H	**Half-Carry Flag (FLAGS.2)** The H bit is set to 1 whenever an addition generates a carry-out of bit 3, or when a subtraction borrows out of bit 4. It is used by the Decimal Adjust (DA) instruction to convert the binary result of a previous addition or subtraction into the correct decimal (BCD) result. The H flag is seldom accessed directly by a program.
FIS	**Fast Interrupt Status Flag (FLAGS.1)** The FIS bit is set during a fast interrupt cycle and reset during the IRET following interrupt servicing. When set, it inhibits all interrupts and 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.

7.6. Instruction Set Notation

Table 26 lists the conventions used for each of the flags managed by the Instruction Set. Symbols for the Instruction Set are listed in Table 27; conditions for these symbols are described in Table 28.

Table 26. 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 27. 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 28. Instruction Notation Conventions

Notation	Description	Actual Operand Range
cc	Condition code	See list of condition codes in Table 22 on page 76
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)

7.7. Instruction Set

This section provides programming examples for each instruction in the SAM88 instruction set. Information is arranged in a consistent format for improved readability and for fast referencing.

The following information is included in each instruction description:

- Instruction name (mnemonic)
- Full instruction name
- Source/destination format of the instruction operand
- Shorthand notation of the instruction’s operation
- Textual description of the instruction's effect
- Specific flag settings affected by the instruction
- 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 ← 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

Format	Bytes	Cycles	Op Code (Hex)	Address Mode dst	Address Mode src
opc | dst | src	2	4	12	r	r
		6	13	r	Ir
opc | src | dst	3	6	14	R	R
		6	15	R	IR
opc | dst | src	3	6	16	R	IM

Examples

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	01h, @02h	→	Register 01h = 2Bh, register 02h = 03h
ADC	01h, #11h	→	Register 01h = 32h

In the first of the above five ADC examples, destination register R1 contains the value `10h`, the carry flag is set to 1, and the source working register R2 contains the value `03h`. The *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 ← 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**	**Address Mode**	
			(Hex)	**dst**	**src**
opc | dst | src	2	4	02	r	r
		6	03	r	Ir
opc | src | dst	3	6	04	R	R
		6	05	R	IR
opc | dst | src	3	6	06	R	IM

Examples 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 of the above five ADD examples, 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	Address Mode src
opc | dst | src	2	4	52	r	r
		6	53	r	Ir
opc | src | dst	3	6	54	R	R
		6	55	R	IR
opc | dst | src	3	6	56	R	IM

Examples 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 of the above five AND examples, 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.

Bit AND

BAND dst, src.b

BAND dst.b, src

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

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

Flags

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

Format

	Bytes	Cycles	Op Code (Hex)	Address Mode dst	Address Mode src
opc | dst | b | 0 | src	3	6	67	r0	Rb
opc | src | b | 1 | dst	3	6	67	Rb	r0

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

Examples 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 of the above two BAND examples, 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 the R1 Register (destination), leaving the value `06h` (`00000110b`) in the R1 Register.

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 (Hex)	Address Mode dst	Address Mode src
opc | dst | b | 0 | src	3	6	17	r0	Rb

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

Example 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 dst(b) ← NOT dst(b)

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

Flags

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

Format

	Bytes	Cycles	Op Code (Hex)	Address Mode dst
opc | dst | b | 0	2	4	57	rb

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

Example 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
opc | dst | b | 0	2	4	77	rb

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

Example 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 dst
opc | dst | b | 1	2	4	77	rb

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

Example 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) ← dst(0) OR src(b)
or
dst(b) ← dst(b) OR src(0)

The specified bit of the source (or the destination) is logically ORed with bit 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	Address Mode	
			(Hex)	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.

Examples

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 of the above two BOR examples, 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). This leaves the same value (`07h`) in working register R1.

In the second example, destination register `01h` contains the value `03h` (`00000011b`) and the source working register R1 the value `07h` (`00000111b`). The *BOR 01h.2, R1* statement logically ORs bit 2 of register `01h` (destination) with bit 0 of R1 (source). This 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 ← 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	Address Mode src
opc | dst | b | 1 | dst	3	10	37	RA	rb

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

Example 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 ← 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	Address Mode src
opc	src | b | 1	dst	3	10	37	RA	rb

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

Example 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 dst(0) ← dst(0) XOR src(b)
or
dst(b) ← dst(b) XOR src(0)

The specified bit of the source (or the destination) is logically exclusive-ORed with bit 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.

Examples 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 of the above two BXOR examples, 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 dst
opc | dst	3	14	F6	DA
opc | dst	2	12	F4	IRR
opc | dst	2	14	D4	IA

Examples 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 of the above three CALL examples, if the program counter value is `1A47h` and the stack pointer contains the value `0002h`, the *CALL 3521h* statement pushes the current PC value onto the top of the stack. The stack pointer now points to memory location `0000h`. The PC is then loaded with the value `3521h`, the address of the first instruction in the program sequence to be executed.

If the contents of the program counter and stack pointer are the same as in the first example, the *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 ← 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 dst
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 dst
opc | dst	2	4	60	R
		4	61	IR

Examples Assume that R1 = `07h` and register `07h` = `0F1h`.

COM	R1	→	R1 = 0F8h
COM	@R1	→	R1 = 07h, register 07h = 0Eh

In the first of the above two COM examples, 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**	**Address Mode**	
			(Hex)	**dst**	**src**
opc | dst | src	2	4	A2	r	r
		6	A3	r	Ir
opc | src | dst	3	6	A4	R	R
		6	A5	R	IR
opc | dst | src	3	6	A6	R	IM

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 ← PC + RA

Ir ← Ir + 1

The source operand is compared to (subtracted from) the destination operand. If the result is 0, the relative address is added to the program counter and control passes to the statement 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	Ir

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

CPIJE dst, src, RA

Operation If dst – src = 0, PC ← PC + RA

Ir ← Ir + 1

The source operand is compared to (subtracted from) the destination operand. If the result is 0, the relative address is added to the program counter and control passes to the statement 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	Address Mode src
opc	dst | src	RA	3	12	C2	r	Ir

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

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

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 ADC	0	0–9	0	0–9	00	0
	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 SBC	0	0–9	0	0–9	00 =–00	0
	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 dst
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:

```
          0001    0101            15
+         0010    0111            27
         ---------------
          0011    1100     =      3Ch
```

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

```
          0011    1100
+         0000    0110
         ---------------
          0100    0010     =      42
```

Assuming the same values provided 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 ← 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 dst
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 of the above two DEC examples, 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 ← 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

Examples 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 of the above two DECW examples, 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 ≥ 28, 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 ≥ 28 or if divisor = 0; cleared otherwise.
D	Unaffected.
H	Unaffected.

Format

				Bytes	**Cycles**	**Op Code**	**Address Mode**	
						(Hex)	**dst**	**src**
opc	src	dst		3	26/10	94	RR	R
					26/10	95	RR	IR
					26/10	96	RR	IM

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

Examples 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 of the above three DIV examples, 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 ← r – 1

If r ≠0, PC ← PC + dst

The working register being used as a counter is decremented. If the contents of the register are not logic 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: When using a DJNZ instruction, the working register being used as a counter should be set at location `0C0h` to `0CFh` with SRP, SRP0, or SRP1 instruction.

Flags No flags are affected.

Format

	Bytes	Cycles	Op Code (Hex)	Address Mode dst
r | opc | dst	2	8 (jump taken)	rA	RA
		8 (no jump)	r = 0 to F	

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 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 43 shows an example of how to use an *ENTER* statement.

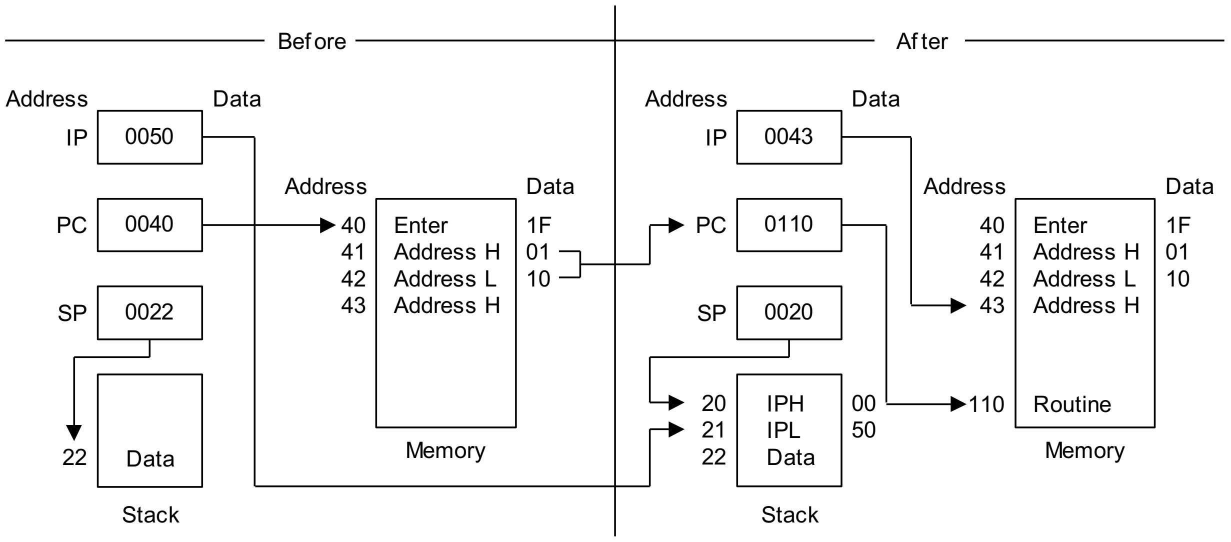


Figure 43. How to Use an ENTER Statement

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 44 shows an example of how to use an *EXIT* statement.

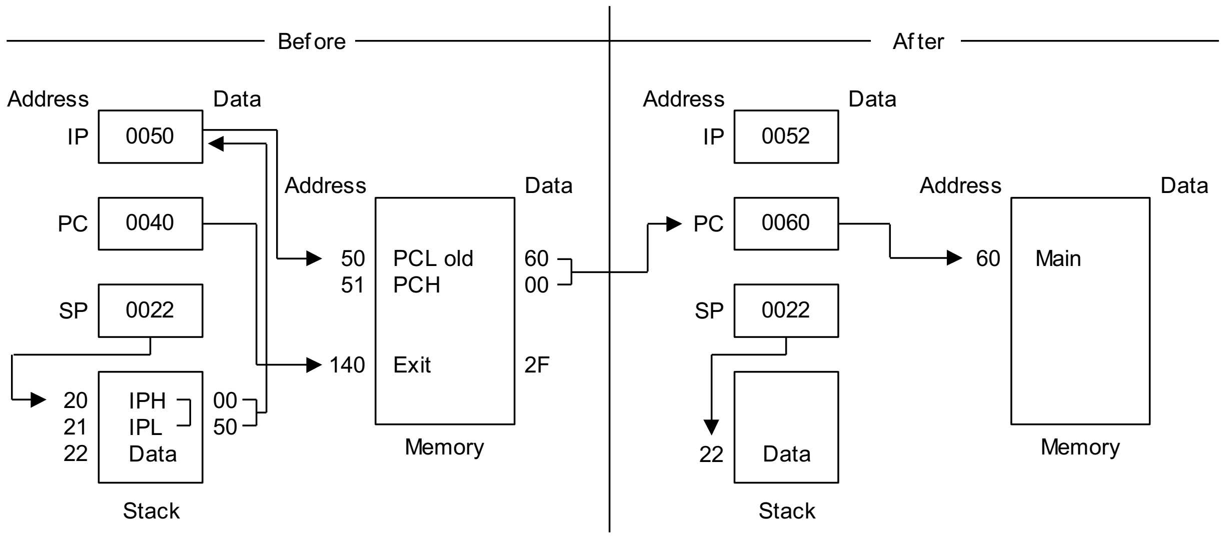


Figure 44. How to Use an EXIT Statement

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)
opc	1	4	6F

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**
dst | src	1	4	rE r = 0 to F	r
opc | dst	2	4	20	R
		4	21	IR

Examples 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 of the above three INC examples, 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 an INC instruction has on register `00h`, assuming that it contains the value `0Ch`.

In the third example, INC is used in Indirect Register (IR) Addressing Mode to increment the value of register `1Bh` from `0Fh` to `10h`.

Increment Word

INCW dst

Operation dst ← dst + 1

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

Flags

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

Format

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

Examples 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 of the above two INCW examples, 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

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

This instruction is used at the end of an interrupt service routine. It restores the 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)	BF
		12 (internal stack)	

IRET (Fast)	Bytes	Cycles	Op Code
opc	1	6	BF

Example In the Figure 45, 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, causing the PC to jump to address `100h` and the IP to keep the return address.

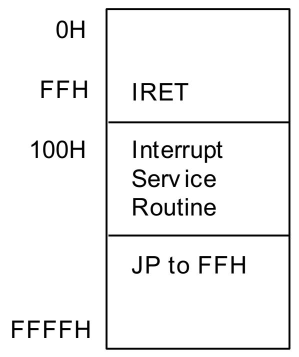


Figure 45. Instruction Pointer

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

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

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.

Examples 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 of the above two JP examples shows a conditional JP. Assuming that the carry flag is set to 1, the *JP C, LABEL_W* statement replaces the contents of the PC with the value `1000h` and transfers control to that location. If the carry flag is not set, the control is passed 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 ← 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 22 on page 76.

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

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	Address Mode	
					(Hex)	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	Ir
				4	D7	Ir	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 = `01`h, R1 = `0A`h, register `00`h = `01`h, register `01`h = `20`h, register `02`h = `02`h, LOOP = `30`h and register `3A`h = `0FF`h.

LD	R0, #10h	⟶	R0 = 10h
LD	R0, 01h	⟶	R0 = 20h, register 01h = 20h
LD	01h, R0	⟶	Register 01h = 01h, R0 = 01h

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

Load Bit

LDB dst, src.b

LDB dst.b, src

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

The specified bit of the source is loaded into bit 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

Format			Bytes	Cycles	Op Code (Hex)	Address Mode dst	Address Mode src
opc	dst | b | 0	src	3	6	47	r0	Rb
opc	src | b | 1	dst	3	6	47	Rb	r0

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

Examples 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 of the above two LDB examples, 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 Irr or rr values to be even numbers for program memory and odd numbers for data memory.

Flags No flags are affected.

Format

Format	Bytes	Cycles	Op Code (Hex)	Address Mode dst	Address Mode src
opc | dst | src	2	10	C3	r	Irr
opc | src | dst	2	10	D3	Irr	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
opc | dst | 0001 | DA_L | DA_H	4	14	A7	r	DA

				Bytes	Cycles	Op Code (Hex)	Address Mode dst	Address Mode 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	Address Mode 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	Address Mode 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 ← rr – 1

dst ← src

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

LDCPD refers to program memory and LDEPD refers to external data memory. The assembler 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	Address Mode	
				(Hex)	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	Address Mode src
opc | dst | src	2	14	F3	Irr	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	Address Mode	
			(Hex)	dst	src
opc src dst	3	8	C4	RR	RR
		8	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 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 ← dst × src

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

Flags

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

Format

	Bytes	Cycles	Op Code	Address Mode	
			(Hex)	dst	src
opc | src | dst	3	22	84	RR	R
		22	85	RR	IR
		22	86	RR	IM

Examples 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, @02h	→	Register 00h = 06h, register 01h = 00h

In the first of the above three MULT examples, 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 ← @ IP

IP ← IP + 2

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

Flags No flags are affected.

Format

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

Example Figure 46 shows an example of how to use the NEXT instruction.

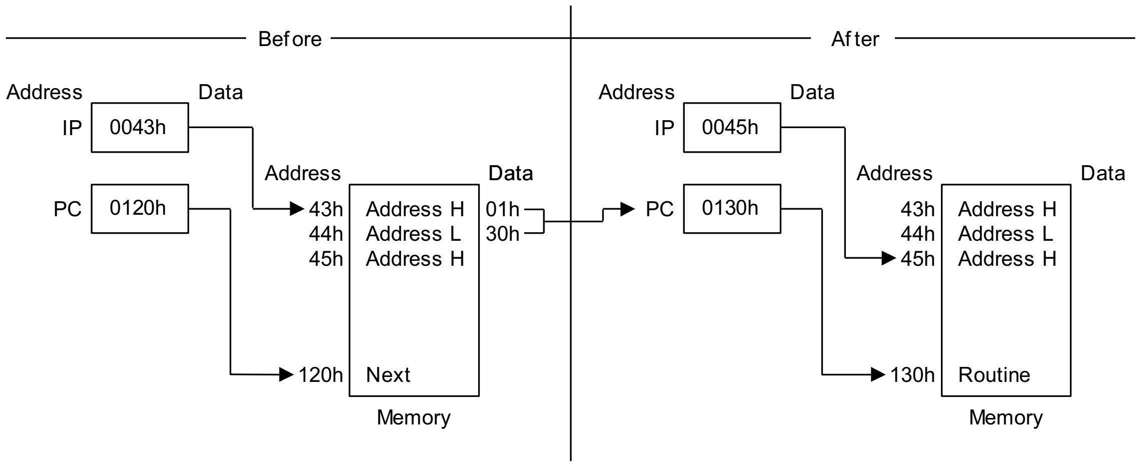


Figure 46. How to Use the NEXT Instruction

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	Address Mode src
opc | dst | src	2	4	42	r	r
		6	43	r	Ir
opc | src | dst	3	6	44	R	R
		6	45	R	IR
opc | dst | src	3	6	46	R	IM

Examples 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 of the above five OR examples, 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

ADC 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 dst
opc | dst	2	8	50	R
		8	51	IR

Examples 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 of the above two POP examples, 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.

Format

			Bytes	Cycles	Op Code	Address Mode	
					(Hex)	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 ← SP – 1

@SP ← src

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

Flags No flags are affected.

Format

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

Examples 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 of the above two PUSH examples, 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 ← IR – 1

dst ← src

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

Flags No flags are affected.

Format

			Bytes	Cycles	Op Code (Hex)	Address Mode dst	Address Mode 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)	Address Mode dst	Address Mode 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 ← 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 ← @SP

SP ← SP + 2

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

Flags

No flags are affected.

Format

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

Example

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

RET → 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 ← dst(7)

dst(0) ← dst(7)

dst(n + 1) ← dst(n), n = 0 – 6

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

Figure 47 shows how bits rotate left.

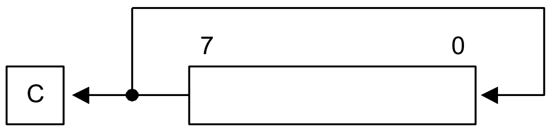


Figure 47. Rotate Left

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 dat**
opc | dst	2	4	90	R
		4	91	IR

Examples

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 of the above two RL examples, 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) ← C

C ← dst(7)

dst(n + 1) ← dst(n), n = 0 – 6

The contents of the destination operand with the carry flag are rotated left one bit position. The initial value of bit 7 replaces the carry flag (C); the initial value of the carry flag replaces bit 0.

Figure 48 shows how bits rotate left through carry.

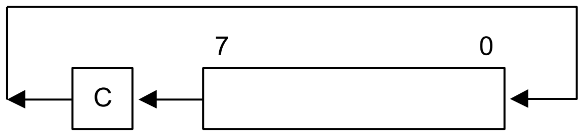


Figure 48. Rotate Left through Carry

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

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

Examples

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

RLC	00h	Register 00h = 54h, C = 1
RLC	@01h	Register 01h = 02h, register 02h = 2Eh, C = 0

In the first of the above two RLC examples, 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 ← dst(0)

dst(7) ← dst(0)

dst(n) ← dst(n + 1), n = 0 – 6

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

Figure 49 shows how bits rotate right.

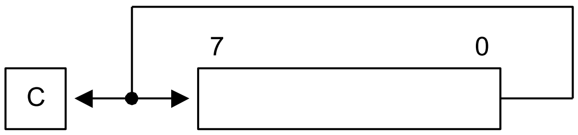


Figure 49. Rotate Right

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 dst
opc | dst	2	4	E0	R
		4	E1	IR

Examples 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 of the above two RR examples, 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 also resets the C flag to 1 and the sign flag and overflow flag are also set to 1.

Rotate Right through Carry

RRC dst

Operation

dst(7) ← C

C ← dst(0)

dst(n) ← dst(n + 1), n = 0 – 6

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

Figure 50 shows how bits rotate right through carry.

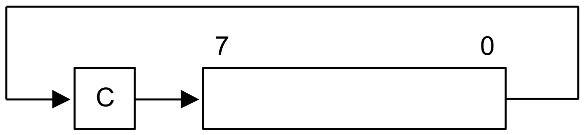


Figure 50. Rotate Right through Carry

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 dst
opc | dst	2	4	C0	R
		4	C1	IR

Examples

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 of the above two RRC examples, 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 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 ← 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

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

Examples 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 of the above five SBC examples, 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 ← 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) ← dst(7)

C ← dst(0)

dst(n) ← dst(n + 1), n = 0 – 6

An arithmetic shift-right of one bit position is performed on the destination operand. Bit 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 51 shows how bits shift right.

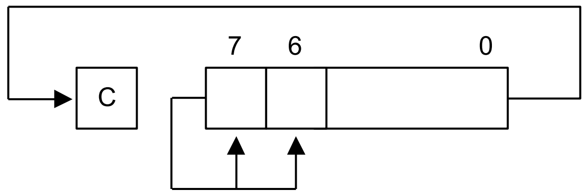


Figure 51. Shift Right

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 dst
opc | dst	2	4	D0	R
		4	D1	IR

Examples

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 of the above two SRA examples, 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 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 (Hex)	Address Mode 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	Address Mode src
opc	1	4	7F	–	–

Example The STOP statement halts all microcontroller operations.

Subtract

SUB dst, src

Operation dst ← dst – src

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

Flags

C	Set if a borrow occurred; cleared otherwise.
Z	Set if the result is 0; cleared otherwise.
S	Set if the result is negative; cleared otherwise.
V	Set if arithmetic overflow occurred, 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

Format	Bytes	Cycles	Op Code (Hex)	Address Mode dst	Address Mode src
opc | dst | src	2	4	22	r	r
		6	23	r	Ir
opc | src | dst	3	6	24	R	R
		6	25	R	IR
opc | dst | src	3	6	26	R	IM

Examples 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 of the above six SUB examples, 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 52 shows how to swap nibbles.

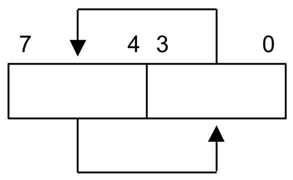


Figure 52. Swap Nibbles

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 dst
opc | dst	2	4	F0	R
		4	F1	IR

Examples 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 of the above two SWAP examples, 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	Address Mode src
opc | dst | src	2	4	62	r	r
		6	63	r	Ir
opc | src | dst	3	6	64	R	R
		6	65	R	IR
opc | dst | src	3	6	66	R	IM

Examples 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 of the above five TCM examples, if working register R0 contains the value `0C7h` (`11000111b`) and register R1 the value `02h` (`00000010b`), 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	Address Mode src
opc | dst | src	2	4	72	r	r
		6	73	r	Ir
opc | src | dst	3	6	74	R	R
		6	75	R	IR
opc | dst | src	3	6	76	R	IM

Examples 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, #54	Register 00h = 2Bh, Z = 1

In the first of the above five TM examples, if working register R0 contains the value `0C7h` (`11000111b`) and register R1 the value `02h` (`00000010b`), 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 occur 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 53 presents a sample program structure that depicts the sequence of operations that follow a *WFI* statement.

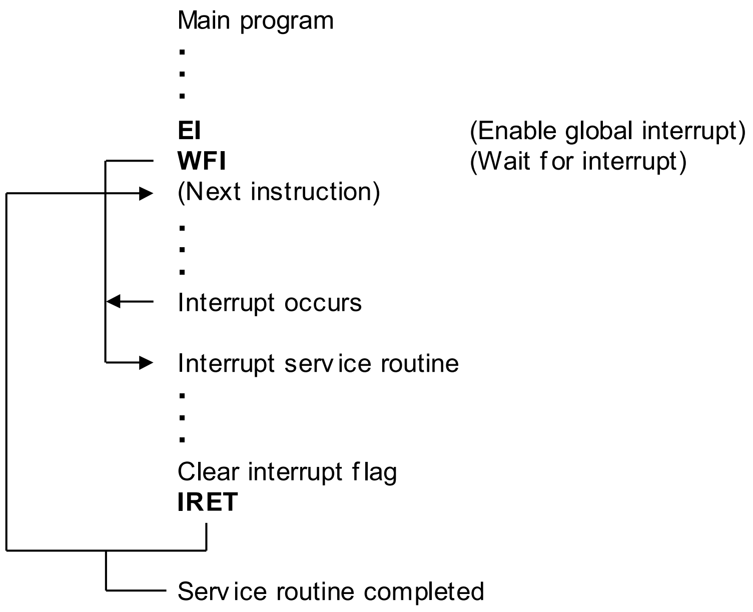


Figure 53. Sample Program Structure

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

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

Examples 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 of the above five XOR examples, 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 Circuit

The clock frequency for the S3F80P5 can be generated by an external crystal or supplied by an external clock source. The clock frequency for the S3F80P5 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 resister and two separate capacitors across the power pins to suppress high frequency noise and provide bulk charge storage for the overall system. These main oscillator circuits are shown in Figures 54 through 57.

8.1. System Clock Circuit

The system clock circuit features the following components:

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

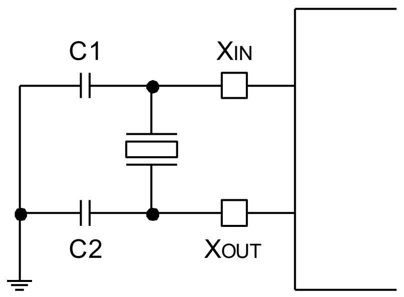


Figure 54. Crystal/Ceramic Main Oscillator Circuit

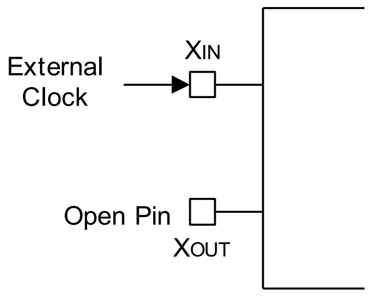


Figure 55. External Clock Circuit

8.2. Clock Status During Power-Down Modes

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

- In Stop Mode, the main oscillator is halted. When Stop Mode is released, the oscillator is started 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 the Stop instruction execution. After recovering from Stop Mode by a reset or an external interrupt, STOPCON Register is automatically cleared.
- The internal clock signal is gated to the CPU in Idle Mode, but not to the interrupt structure, timers, and timer/counters. Idle Mode is released by a reset or by an external or internal interrupt.

A block diagram of the system clock circuit is shown in Figure 56.

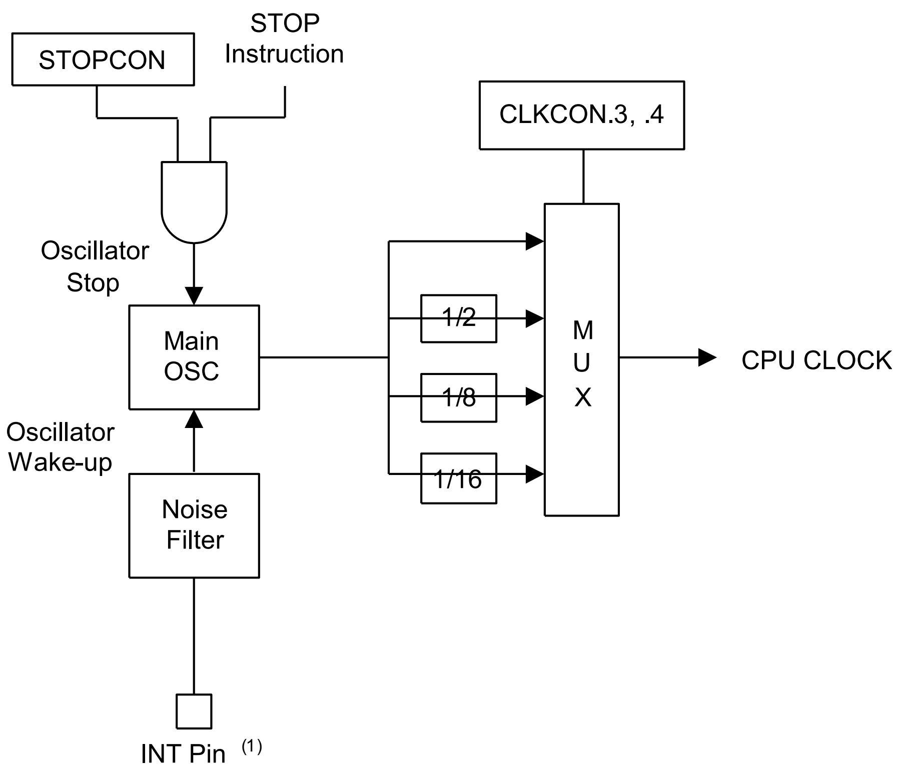


Figure 56. System Clock Circuit Diagram

Notes:
1. An external interrupt with an RC-delay noise filter (for the S3F80P5 INT0-5) is fixed to release Stop Mode and wake up the main oscillator.
2. Because the S3F80P5 contains no subsystem clock, the 3-bit CLKCON signature code (CLKCON.2-CLKCON.0) is meaningless.

8.3. System Clock Control Register

The System Clock Control (CLKCON) Register, shown in Table 30, is located at address `D4h`, Set1, Bank0. This register is read/write-addressable and offers an oscillator frequency divide-by value.

The CLKCON.7–.5 and CLKCON.2–.0 bits are not used in the S3F80P5 MCU. After a reset, the main oscillator is activated, and the f_{OSC}/16 (the slowest clock speed) is selected as the CPU clock. If necessary, the CPU clock speed can be increased to f_{OSC}, $f_{OSC}/2$, $f_{OSC}/8$,or $f_{OSC}/16$.

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	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	D4h							
Mode	Register Addressing Mode only							

Note: R = read only; R/W = read/write.

Bit	**Description**
[7:5]	**Reserved** Must be written to 0.
[4:3]	**CPU System Clock Selection Bits**[1] 00: $f_{OSC}/16$. 01: $f_{OSC}/8$. 10: $f_{OSC}/2$. 11: f_{OSC} (not divided).
[2:0]	**Reserved** Must be written to 0.

Notes:
1. After a reset, the slowest clock (divided by 16) is selected as the system clock. To select faster CPU clock speeds, load the appropriate values to CLKCON.3 and CLKCON.4.
2. Selection bits CLKCON.0–.2 are required only for systems that contain a main clock and a subsystem clock. The S3F80P5 MCU only uses a main oscillator clock circuit. For this reason, the 101b setting is invalid.

Typically, application systems contain a resister and two separate capacitors across the power pins as shown in Figure 57. R1 and C1 are located as near to the MCU 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. We recommend that R1 = 10Ω, C1 = 0.1 µF and C2 = 100 µF.

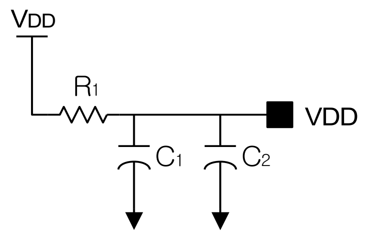


Figure 57. Power Circuit (V_{DD})

Table 31. Falling and Rising Rate 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 and Power-Down

This chapter discusses system reset and power-down modes. 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. A reset vector can be changed by the Smart Option in the S3F80P5 MCU; see Figure 10 on page 15.

9.1. Reset Sources

The S3F80P5 features five different system reset sources as following.

- Watchdog Timer (WDT): When the watchdog timer is enabled in the normal operating mode, a reset is generated whenever a basic timer overflow occurs.
- Low Voltage Detect (LVD): A reset occurs when the V_{DD} is changed in condition for LVD operation in the normal operating mode.
- Internal Power-On Reset (IPOR): A reset is generated when the V_{DD} condition is changed in IPOR operation.
- External Interrupt (INT0-INT5): When the RESET Control Bit is set to 0 (Smart Option @ `03Fh`) and the MCU is in Stop Mode, external interrupts by P0 and P2.0 generate the reset signal if the external interrupt is enabled.
- Stop Error Detection and Recovery (SED & R): When the RESET Control Bit is set to 0 (Smart Option bit [7] @ `03Fh`) and the MCU is in stop or abnormal state, the falling edge input of P0 generates the reset signal regardless of external interrupt enable or disable.

Figure 58 illustrates the RESET sources of the S3F80P5.

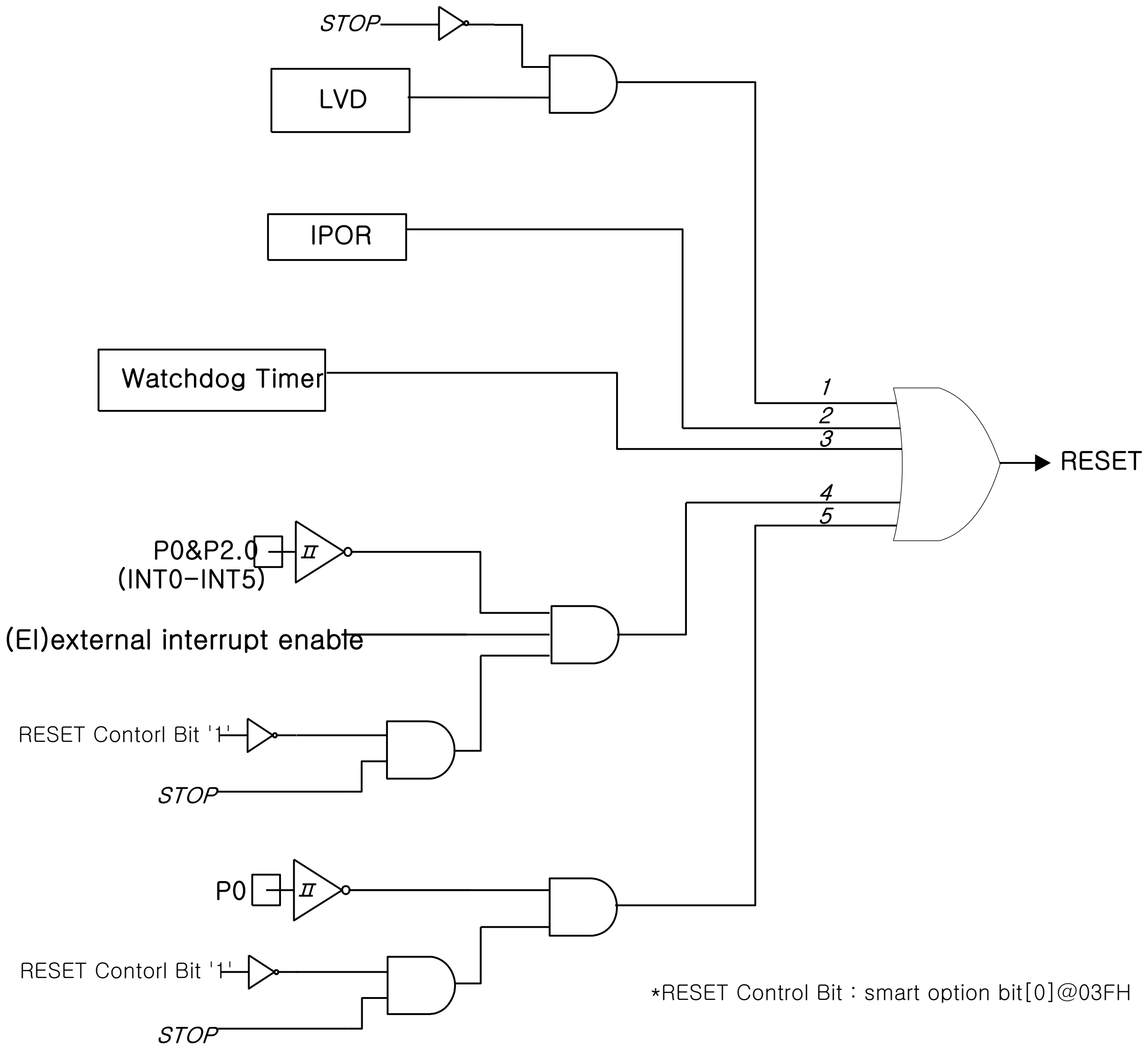


Figure 58. RESET Sources of the S3F80P5

Notes: 1. The rising edge detection of LVD circuit while rising of V_{DD} passes the level of V_{LVD}.

2. When POR circuit detects V_{DD} below V_{POR}, a reset is generated by the internal power-on reset.

3. Basic Timer over-flow for watchdog timer. Refer to the Basic Timer and Timer 0 chapter on page 229.
4. When RESET Control Bit (Smart Option @ `03Fh`) is set to 0 and the MCU is in Stop Mode, external interrupt input by P0 and P2.0 generates the reset signal.
5. When RESET Control Bit (Smart Option @ `03Fh`) are set to 0 and the MCU is in Stop Mode or abnormal state, the falling edge input of P0 generates the reset signal regardless of external interrupt enable/disable.Hardware Reset Values.

Figure 59 illustrates the RESET block diagram of the S3F80P5.

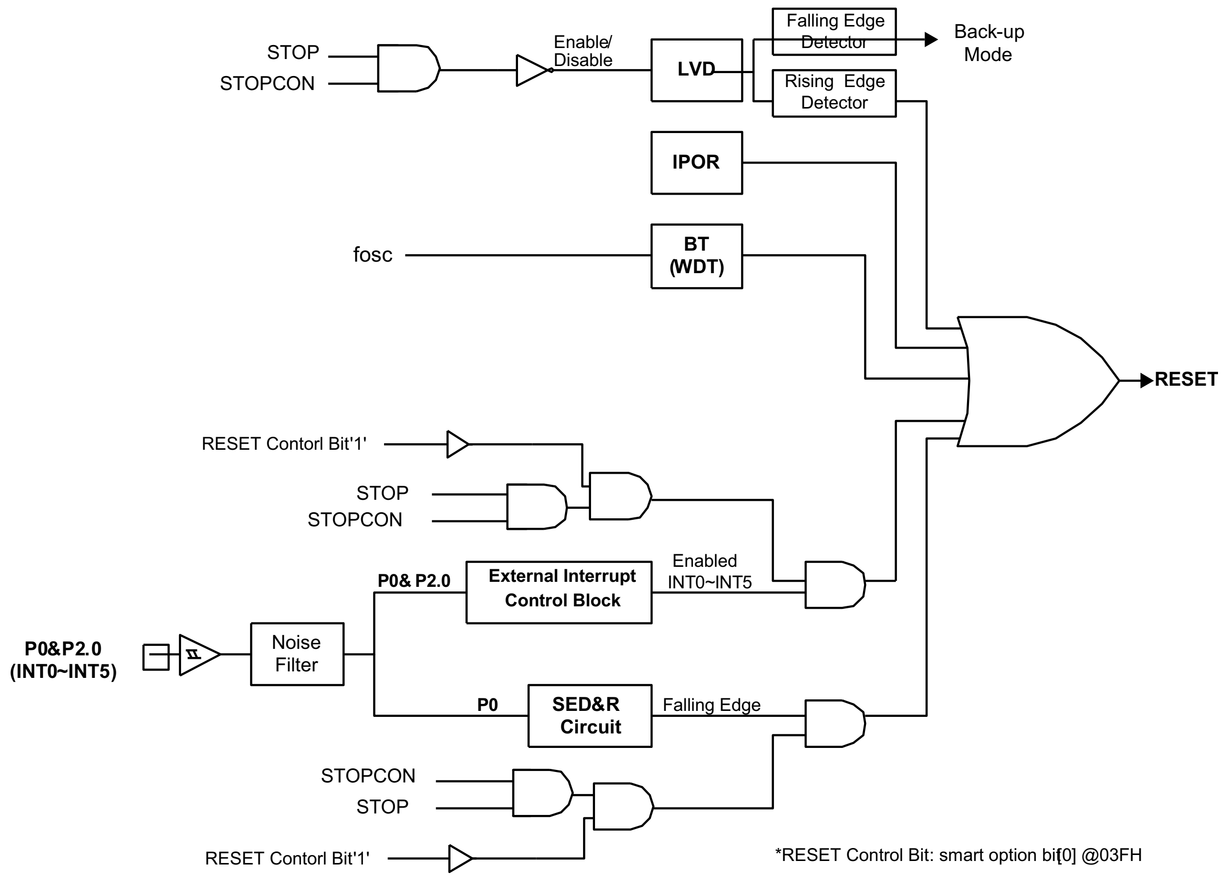


Figure 59. RESET Block Diagram of the S3F80P5

9.2. Reset Mechanism

The interlocking work of the reset pin and LVD circuit supplies two operating modes: Backup Mode input, and system reset input. Backup Mode input automatically creates a chip stop state when the voltage at V_{DD} is lower than V_{LVD}. When the LVD circuit detects rising edge of V_{DD} on the point V_{LVD}, the reset pulse generator generates a reset pulse, and system reset occurs. When the operating mode is in Stop Mode, the LVD circuit is disabled to reduce the current consumption under 5 µA (at V_{DD} = 3.6 V). Therefore, although the voltage at V_{DD} is lower than V_{LVD}, the MCU does not go into Backup Mode when the operating state is in Stop Mode.

9.3. Watchdog Timer Reset

A watchdog timer that can recover to normal operation from abnormal function is built in S3F80P5. The watchdog timer generates a system reset signal if the Basic Timer Counter (BTCNT) is not cleared within a specific time by program, see the Basic Timer chapter on page 230.

9.4. LVD Reset

The Low Voltage Detect Circuit (LVD) is built in the S3F80P5 MCU to generate a system reset. The LVD is disabled in Stop Mode. When the voltage at V_{DD} is falling down and passing V_{LVD}, the MCU goes into Backup Mode at the moment $V_{DD} = V_{LVD}$. As the voltage at V_{DD} is rising up, the reset pulse is occurred at the moment $V_{DD} \geq V_{LVD}$.

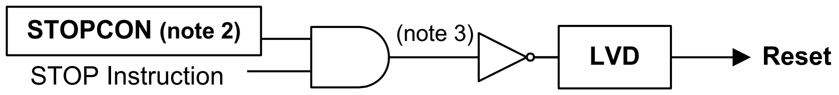


Figure 60. RESET Block Diagram by LVD for the S3F80P5 in Stop Mode

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 Stop Control (STOPCON) Register to `0A5h` before executing the STOP instruction. See Figure 60.
3. The signal is output relating to Stop Mode. If STOPCON is set to `0A5h`, and the Stop instruction is executed, the output signal forces the S3F80P5 to enter Stop Mode. There are two statuses; one is Stop Mode, the other is not Stop Mode. See Figure 60.

9.5. Internal Power-On Reset

The power-on reset circuit is built into the S3F80P5 MCU. When the power is initially applied to the MCU, or when the V_{DD} drops below the V_{POR}, the POR circuit holds the MCU in reset until the V_{DD} has risen above the V_{LVD} level. See Figures 61 and 62.

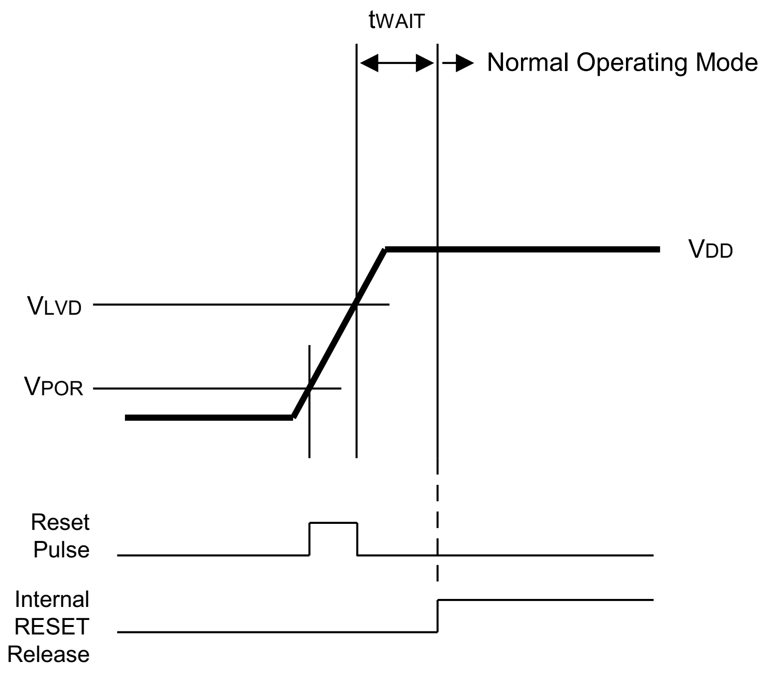


Figure 61. Timing Diagram for Internal Power-On Reset Circuit

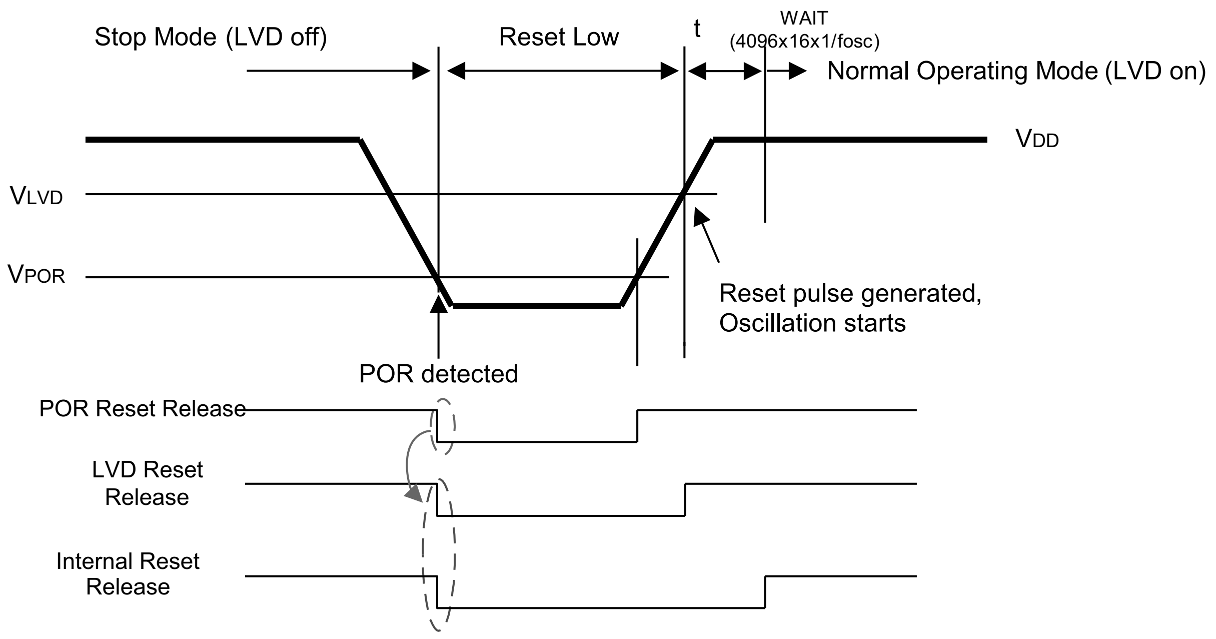


Figure 62. Reset Timing Diagram for the S3F80P5 in Stop Mode by IPOR

Note: If $V_{RESET} > V_{IH}$, the operating status is in Stop Mode and the LVD circuit is disabled in the S3F80P5 MCU.

9.6. External Interrupt Reset

When the RESET Control bit (Smart Option @ `03Fh`) is set to 0 and the MCU is in Stop Mode, an external interrupt occurring among the enabled external interrupt sources, from INT0 to INT5, can generate the reset signal.

9.7. Stop Error Detection and Recovery

When the RESET Control bit (Smart Option bit [0] @ `03Fh`) is set to 0 and the MCU is in a stop or abnormal state, the falling edge input of P0 generates the reset signal (See Table 32).

Table 32. Reset Condition in Stop Mode

Condition			
Slope of V_{DD}	**V_{DD}**	**Reset Source**	**System Reset**
Rising up from $V_{POR} < V_{DD} < V_{LVD}$	$V_{DD} \geq V_{LVD}$	–	No system reset
Rising up from $V_{DD} < V_{POR}$	$V_{DD} \geq V_{LVD}$	Internal POR	System reset occurs

9.8. Power-Down Modes

The three power down modes for the S3F80P5 MCU are:

- Idle Mode
- Back- up mode
- Stop Mode

9.8.1. Idle Mode

Idle mode is invoked by the IDLE instruction (op code `6Fh`). In Idle mode, the 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
- Timer2
- Counter A

The I/O port pins retain the state (input or output) they had at the time Idle Mode was entered.

9.8.1.1. Idle Mode Release

Release Idle Mode in one of two ways:

- 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.

- Activate any enabled interrupt; internal or external. 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.

9.8.2. Backup Mode

For reducing current consumption, the S3F80P5 MCU enters Backup Mode. If a falling level of V_{DD} is detected by the LVD circuit on the point of V_{LVD}, the MCU goes into Backup Mode. The CPU and peripheral operations are stopped, but the LVD is enabled. Because of the oscillation stop, the supply current is reduced. In Backup Mode, MCU cannot be released from stop state by any interrupt. The only way to release Backup Mode is through a system-reset operation by interactive work of the LVD circuit. The system reset for the watchdog timer does not occur in back up mode (see Figures 63 through 65).

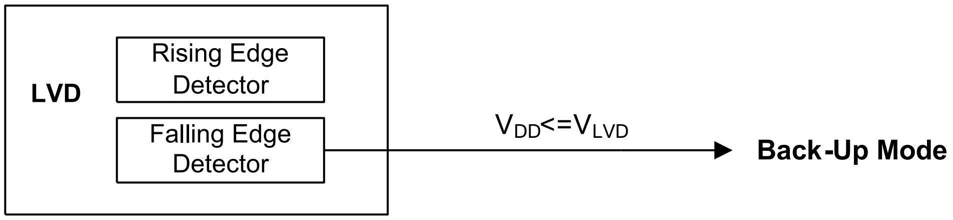


Figure 63. Block Diagram for Backup Mode

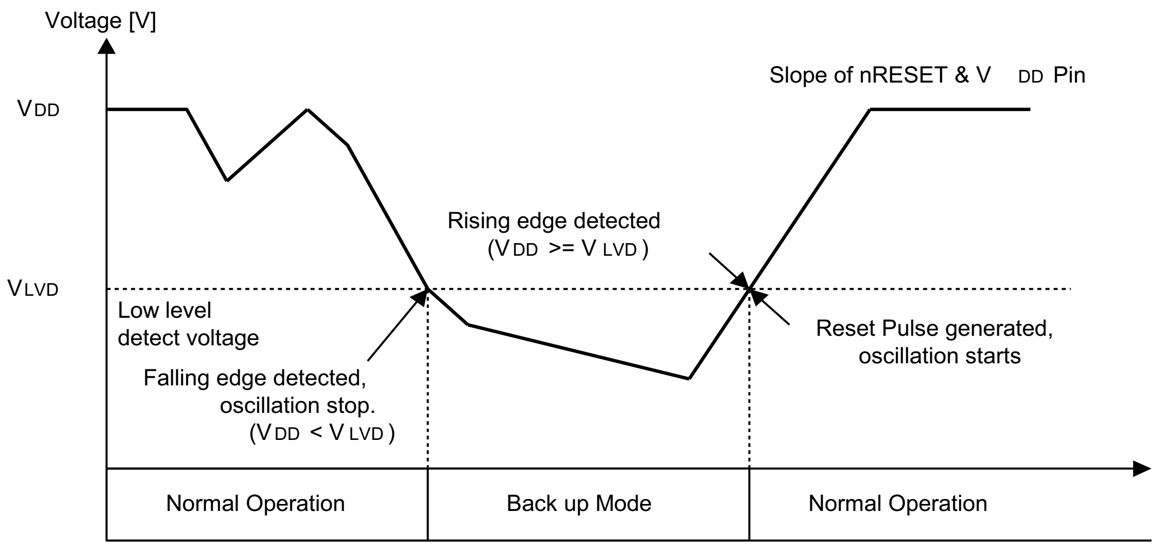


Figure 64. Timing Diagram for Backup Mode Input and Released by LVD

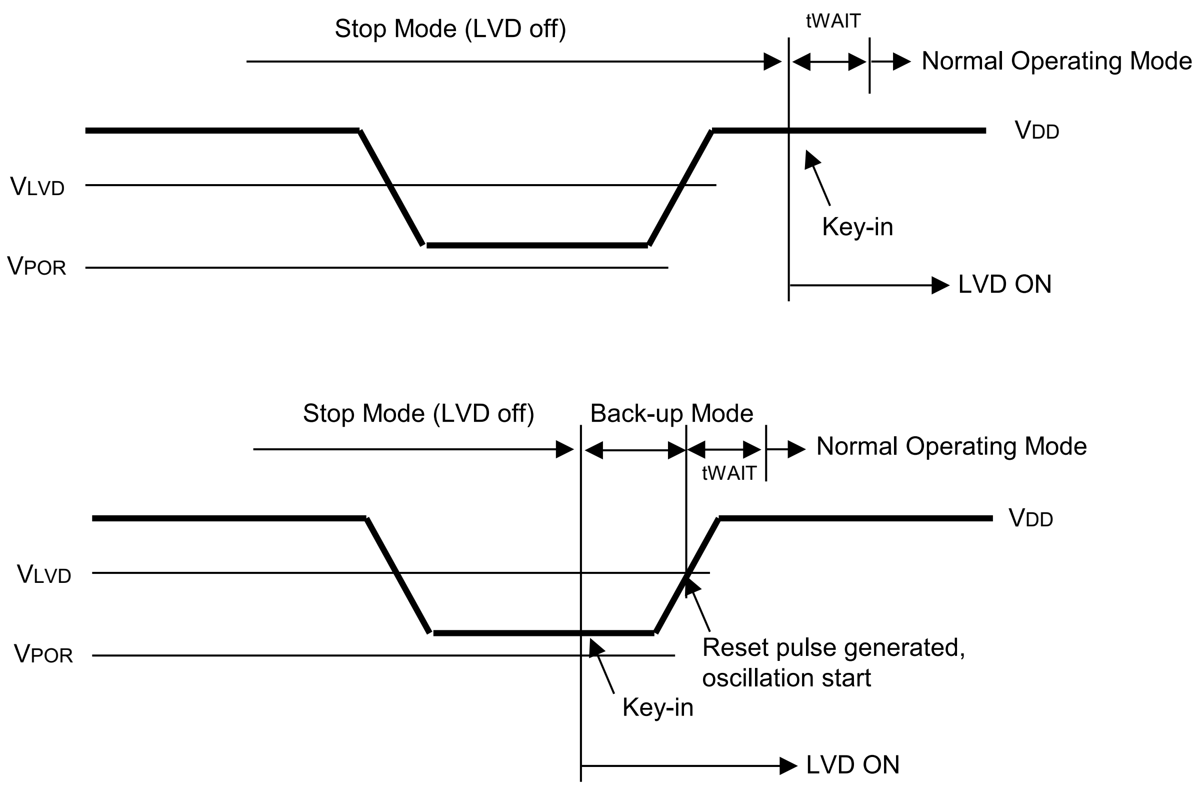


Figure 65. Timing Diagram for Backup Mode Input in Stop Mode

Notes: 1. When a rising edge is detected by the LVD circuit, Backup Mode is released. (V_{LVD} = V_{DD})

2. When a falling edge is detected by the LVD circuit, Backup Mode is activated (V_{LVD} > V_{DD}

9.8.3. Stop Mode

Stop Mode is invoked by executing a STOP instruction after setting the Stop Control (STOPCON) Register. In Stop Mode, the operation of the CPU and all peripherals is halted; i.e., the on-chip main oscillator stops and the current consumption can be reduced. All system functions stop when the clock freezes, but data which is 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 the Stop Control (STOPCON) Register is cleared automatically.

Example 6. The following routine shows 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
            •
            •
            •
```

9.8.3.1. Sources to Release Stop Mode

Stop Mode is released when the following sources go active:

- System Reset by Internal Power-On Reset (IPOR)
- External Interrupt (INT0¡VINT5)
- SED & R circuit

9.8.3.2. Using IPOR to Release Stop Mode

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

9.8.3.3. 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 (Smart Option @ `03Fh`) and an external interrupt is enabled, the S3F80P5 MCU is released from Stop Mode and generates a reset signal. Conversely, when the RESET Control bit is set to 1 (Smart Option @ `03Fh`), the S3F80P5 MCU is released from Stop Mode but does not generate a reset signal. To wake-up from Stop Mode by an external interrupt from INT0 to INT5, the external interrupt should be enabled by setting the corresponding control registers or instructions.

The following bullets are conditions for Stop Mode release:

- When releasing Stop Mode by using an external interrupt, the current values in the system and peripheral control registers are unchanged.
- When using an external interrupt for Stop Mode release, it is possible to program the duration of the oscillation stabilization interval. To perform this action, make the appropriate control and clock settings before entering Stop Mode.
- When using an interrupt to release Stop Mode, the bit-pair setting for CLKCON.4/CLKCON.3 remains unchanged and the currently selected clock value is used.

9.8.3.4. Stop Error Detect and Recovery

The Stop Error Detect and Recovery (SED & R) circuit is used to release Stop Mode and prevent an abnormal Stop Mode that can occur by battery bouncing. SED & R executes two functions in relation to the internal logic of P0. One function is to release from Stop status by switching the level of the input port (P0), and the other function is to prevent the MCU from entering Stop Mode when the MCU is in an abnormal status.

The SED & R circuit performs the following functions:

- Releasing from Stop Mode
- When RESET Control bit is set to 0 (Smart Option @ `03Fh`), if the falling edge input signal is entered through Port 0, the S3F80P5 MCU is released from Stop Mode and generates a reset signal. Conversely, when the RESET Control bit is set to 1 (Smart Option @ `03Fh`), the S3F80P5 MCU is released Stop Mode but a reset does not occur. When the falling edge of a pin on Port 0 is entered, the MCU is released from Stop Mode even though the external interrupt is disabled.
- Keeping the MCU from entering an abnormal Stop Mode
- This circuit detects the abnormal status by checking the port (P0) status. If the MCU is in an abnormal status SED & R keeps the MCU from entering Stop Mode.

Note: The SED & R circuit is not implemented in P2.0. Although 1pin, in this case, P2.0, contains the falling edge input signal in Stop Mode, if an external interrupt is disabled, the Stop state of S3F80P5 MCU is unchanged. Stop Mode should not be used when an external clock source is being used because the X_{IN} input must be cleared internally for the V_{SS} to reduce current leakage.

9.8.4. System Reset Operation

The system reset starts the oscillation circuit, synchronizes the MCU operation with the CPU clock, and initializes the internal CPU and peripheral modules. This procedure brings the S3F80P5 MCU into a known operating status. To allow time for the internal CPU clock oscillation to stabilize, the reset pulse generator must be held to an active level for a minimum time interval after the power supply is within tolerance. The minimum required reset operation for an oscillation stabilization time is 16 oscillation clocks. All system and peripheral control registers are then reset to their default hardware values (See Table 33).

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

- All interrupts are disabled.
- The watchdog function (Basic Timer) is enabled.
- Ports 0, 2 and 3 are set to 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; see Table 33
- The program counter (PC) is loaded with the program reset address `0100h` in the ROM.
- When the programmed oscillation stabilization time interval has elapsed, the instruction stored in the 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 register, BTCON, before entering Stop Mode. Additionally, by writing `0101b` to the upper nibble of BTCON the basic timer watchdog function (which causes a system reset if a basic timer counter overflow occurs), can be disabled, but Zilog recommends using it to prevent the MCU from malfunctioning.

9.8.5. Hardware Reset Values

Tables 33 through 35 list the reset values for CPU and system registers, peripheral control registers, and peripheral data registers following a reset operation.

The following notation is used to represent these reset values:

- 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 33. Set1 Bank0 Register Values After RESET

		Address		**Bit Values After Reset**							
Register Name	**Mnemonic**	**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
Timer 0 Data Register	T0DATA	209	D1h	1	1	1	1	1	1	1	1
Timer 0 Control Register	T0CON	210	D2h	0	0	0	0	0	0	0	0
Basic Timer Control Register	BTCON	211	D3h	0	0	0	0	0	0	0	0
Clock Control Register	CLKCON	212	D4h	0	0	0	0	0	0	0	0
System Flags Register	FLAGS	213	D5h	x	x	x	x	x	x	0	0
Register Pointer 0	RP0	214	D6h	1	1	0	0	0	–	–	–
Register Pointer 1	RP1	215	D7h	1	1	0	0	1	–	–	–
Location D8h is not mapped.											
Stack Pointer (low byte)	SPL	217	D9h	x	x	x	x	x	x	x	x
Instruction Pointer (high byte)	IPH	218	DAh	x	x	x	x	x	x	x	x
Instruction Pointer (low byte)	IPL	219	DBh	x	x	x	x	x	x	x	x
Interrupt Request Register (read only)	IRQ	220	DCh	0	0	0	0	0	0	0	0
Interrupt Mask Register	IMR	221	DDh	x	x	x	x	x	x	x	x
System Mode Register	SYM	222	DEh	0	–	–	x	x	x	0	0
Register Page Pointer	PP	223	DFh	0	0	0	0	0	0	0	0
Port 0 Data Register	P0	224	E0h	0	0	0	0	0	0	0	0

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

Table 33. Set1 Bank0 Register Values After RESET (Continued)

		Address		Bit Values After Reset							
Register Name	**Mnemonic**	**Dec**	**Hex**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Port 1 Data Register	P1	225	E1h	0	0	0	0	0	0	0	0
Port 2 Data Register	P2	226	E2h	0	0	0	0	0	0	0	0
Port 3 Data Register	P3	227	E3h	0	–	0	0	1	1	0	0
Location E4h is not mapped.											
Port 2 Interrupt Enable Register	P2INT	229	E5h	0	0	0	0	0	0	0	0
Port 2 Interrupt Pending Register	P2PND	230	E6h	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
Port 0 Control Register (high byte)	P0CONH	232	E8h	0	0	0	0	0	0	0	0
Port 0 Control Register (low byte)	P0CONL	233	E9h	0	0	0	0	0	0	0	0
Port 1 Control Register (high byte)	P1CONH	234	EAh	1	1	1	1	1	1	1	1
Port 1 Control Register (low byte)	P1CONL	235	EBh	0	0	0	0	0	0	0	0
Location ECh is not mapped.											
Port 2 Control Register (low byte)	P2CONL	237	EDh	0	0	0	0	0	0	0	0
Port 2 Pull-up Enable Register	P2PUR	238	EEh	0	0	0	0	0	0	0	0
Port 3 Control Register	P3CON	239	EFh	0	0	0	0	0	0	0	0
Location F0h is not mapped.											
Port 0 Interrupt Enable Register	P0INT	241	F1h	0	0	0	0	0	0	0	0
Port 0 Interrupt Pending Register	P0PND	242	F2h	0	0	0	0	0	0	0	0
Counter A Control Register	CACON	243	F3h	0	0	0	0	0	0	0	0
Counter A Data Register (high byte)	CADATAH	244	F4h	1	1	1	1	1	1	1	1
Counter A Data Register (low byte)	CADATAL	245	F5h	1	1	1	1	1	1	1	1
Timer 1 Counter Register (high byte)	T1CNTH	246	F6h	0	0	0	0	0	0	0	0
Timer 1 Counter Register (low byte)	T1CNTL	247	F7h	0	0	0	0	0	0	0	0
Timer 1 Data Register (high byte)	T1DATAH	248	F8h	1	1	1	1	1	1	1	1
Timer 1 Data Register (low byte)	T1DATAL	249	F9h	1	1	1	1	1	1	1	1

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

Table 33. Set1 Bank0 Register Values After RESET (Continued)

		Address		Bit Values After Reset							
Register Name	**Mnemonic**	**Dec**	**Hex**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Timer 1 Control Register	T1CON	250	FAh	0	0	0	0	0	0	0	0
Stop Control Register	STOPCON	251	FBh	0	0	0	0	0	0	0	0
Location FCh is not mapped. (for factory test)											
Basic Timer Counter	BTCNT	253	FDh	0	0	0	0	0	0	0	0
External Memory Timing Register	EMT	254	FEh	0	1	1	1	1	1	0	–
Interrupt Priority Register	IPR	255	FFh	x	x	x	x	x	x	x	x

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

Table 34. Set1, Bank1 Register Values After RESET

		Address		Bit Values After Reset							
Register Name	**Mnemonic**	**Dec**	**Hex**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LVD Control Register	LVDCON	224	E0h	–	–	–	–	–	–	–	0
Location E1h is not mapped.											
Location E2h is not mapped.											
Location E3h is not mapped.											
Timer 2 Counter Register (high byte)	T2CNTH	228	E4h	0	0	0	0	0	0	0	0
Timer 2 Counter Register (low byte)	T2CNTL	229	E5h	0	0	0	0	0	0	0	0
Timer 2 Data Register (high byte)	T2DATAH	230	E6h	1	1	1	1	1	1	1	1
Timer 2 Data Register (low byte)	T2DATAL	231	E7h	1	1	1	1	1	1	1	1
Timer 2 Control Register	T2CON	232	E8h	0	0	0	0	0	0	0	0
Locations E9h–EBh are not mapped.											
Flash Memory Sector Address Register (high byte)	FMSECH	236	ECh	0	0	0	0	0	0	0	0
Flash Memory Sector Address Register (low byte)	FMSECL	237	EDh	0	0	0	0	0	0	0	0
Flash Memory User Programming Enable Register	FMUSR	238	EEh	0	0	0	0	0	0	0	0

Table 34. Set1, Bank1 Register Values After RESET (Continued)

		Address		Bit Values After Reset							
Register Name	**Mnemonic**	**Dec**	**Hex**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Flash Memory Control Register	FMCON	239	EFh	0	0	0	0	–	–	–	0
Reset Indicating Register	RESETID	240	F0h	Refer to the Control Registers chapter on page 47							
LVD Flag Level Selection Register	LVDSEL	243	F1h	0	0	–	–	–	–	–	–
Port 1 Output Mode Pull-up Enable Register	P1OUTPU	244	F2h	0	0	0	0	0	0	0	0
Port 2 Output Mode Selection Register	P2OUTMD	245	F3h	0	0	0	0	0	0	0	0
Port 3 Output Mode Pull-up Enable Register	P3OUTPU	246	F4h	–	–	0	0	–	–	0	0

Table 35. Reset Generation According to the Condition of Smart Option

		Smart Option 1st Bit @ 3Fh			
Mode	**Reset Source**	**1**		**0**	
Normal Operating	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

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

Table 35. Reset Generation According to the Condition of Smart Option

Mode	Reset Source		Smart Option 1st Bit @ 3Fh			
			1		0	
Stop Mode	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 Reset	O	Stop Release and Reset
	SED & R	P0	X	Stop Release and Reset	O	Stop Release and Reset
		P2.0	X	STOP	X	STOP

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

9.8.6. Recommendation for Unused Pins

To reduce overall power consumption, configure unused pins according to the guideline description as listed in Table 36.

Table 36. Page 8 Register Values After RESET

Pin Name	Recommend	Example

Port 0	Set Input Mode
	Enable Pull-Up Resistor
	No Connection for Pins
		P0CONH ← #00h or 0FFh
		P0CONL ← #00h or 0FFh
		P0PUR ← #0FFh

Port 1	Set Open-drain Output Mode
	Set P1 Data Register to #00h
	Disable Pull-Up Resistor
	No Connection for Pins
		P1CONH ← #55h
		P1CONL ← #55h
		P1 ← #00h

Port 2.0	Set Push-pull Output Mode
	Set P2 Data Register to #00h
	Disable Pull-Up Resistor
	No Connection for Pins
		P2CONL ← #0AAh
		P2 ← #00h
		P2PUR ← #00h

Table 36. Page 8 Register Values After RESET (Continued)

Pin Name	Recommend	Example

Ports 3.0–3.1	Set Push-pull Output Mode
	Set P3 Data Register to #00h
	No Connection for Pins
		P3CON ← #11010010b
		P3 ← #00h

Test	Connect to V_{SS}	–

9.8.7. Summary Table of Backup Mode, Stop Mode, and Reset Status

Table 37 lists a summary of each mode for the approach condition, port status, control register, releasing condition and other statuses.

Table 37. Summary of Each Mode

Item/Mode	Back-up	Reset Status	Stop
Approach Condition	V_{DD} is lower than V_{LVD}.	The rising edge at V_{DD} is detected by LVD circuit (When $V_{DD} \geq V_{LVD}$). Watchdog timer overflow signal is activated.	STOPCON← #A5h STOP (LD STOPCON, #0A5h) (STOP).
Port Status	All I/O ports are floating status. All of the ports become input mode but are blocked. Disable all pull-up resistors.	All I/O ports are floating status. Disable all pull-up resistors.	All of the ports keep the previous status. Output port data is not changed.
Control Register	All control registers and system registers are initialized as listed in Table 33 on page 202.	All control registers and system registers are initialized as listed in Table 33 on page 202.	–
Releasing Condition	Rising edge of the LVD circuit is generated.	Occurs after passing an oscillation warm-up time.	External interrupt or reset SED & R circuit.
Others	There is no consumption in MCU.	There can be input leakage current in MCU.	Stop Mode dependent on control program.

9.8.7.1. Stop Control Register

The Stop Control (STOPCON) Register is presented in Table 38.

Table 38. 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							
Mode	Register Addressing Mode only							
Note: W = write only.								

Bit	**Description**
[7:0]	**Stop Control Register Enable Bits** 10100101: Enable Stop Mode. All other bits: Disable Stop Mode.

Notes:
1. To enter Stop Mode, the Stop Control Register must be enabled just before a STOP instruction.
2. When Stop Mode is released, the STOPCON value is cleared automatically.
3. Writing additional values into STOPCON is prohibited.

9.8.7.2. Reset Indicating Register

The Reset Indicating (RESETID) Register is presented in Table 39; also see Table 40.

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

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	0	0	0	0	0	0	0	0
R/W	–	–	–	R/W	R/W	R/W	R/W	R/W
Address	F0h							
Mode	Register Addressing Mode only							
Note: R/W = read/write.								

Bit	**Description**
[7:4]	**Reserved** Not used in the S3F80P5 MCU; must be set to 0000.
[3]	**Key-In Reset Indicating Bit** 0: Reset is not generated by P0, P2 external interrupts. 1: Reset is generated by P0, P2 external interrupts.

Bit	Description (Continued)
[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).

The state of a RESETID depends on its reset source, as indicated in Table 40.

Table 40. State of RESETID Depends on Reset Source[1]

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

Notes:
1. To clear an indicating register, write a 0 to the indicating flag bit. Writing a 1 to a reset indicating flag (RESETID.0–.3) 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 S3F80P5 microcontroller features four bit-programmable I/O ports, P0–P3. Two ports (P0 an d P1) are 8-bit ports, P2 is a 1-bit port, and P3 is a 2-bit port, for a total of 19 I/O pins. Each port is bit-programmable and can be flexibly configured to meet application design requirements. The CPU accesses ports by directly writing or reading the port registers; no special I/O instructions are required. For IR applications, Port 0 and Port 1 are usually configured to the keyboard matrix, Port 2 is a normal I/O pin and Port 3 is used to IR drive pins.

Table 41 provides a general overview of the S3F80P5 MCU's I/O port functions.

Table 41. S3F80P5 Port Configuration Overview

Port	Configuration Options
Port 0	8-bit general-purpose I/O port; Input or Push-pull output modes; external interrupt input on falling edges, rising edges, or both edges; all P0 pin circuits contain noise filters, interrupt enable/disable the (P0INT) Register and the Pending Control (P0PND) Register. Pull-up resistors can be assigned to individual P0 pins using P0PUR register settings. This port is dedicated for key input in an IR controller application.
Port 1	8-bit general-purpose I/O port; Input with or without pull-up, open-drain output, or push-pull output. This port is dedicated for key output in an IR controller application.
Port 2	1-bit general-purpose I/O port; input, Push-pull output, or Open-drain output. P2.0 can be used as an external interrupt inputs which contains 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 the P2PUR Register settings.
Port 3.0-Port 3.1	2-bit I/O port; P3.0 and P3.1 are configured input functions (input mode, with or without pull-up, for T0CK, T0CAP and T1CAP) or output functions (push-pull or open-drain output mode, and for REM and T0PWM). P3.1 is dedicated for an IR drive pin and P3.0 can be used for indicator LED drive.

10.1. Port Data Registers

Table 42 provides an overview of the register locations of all four S3F80P5 I/O port data registers. Data registers for ports 0, 1, 2, 3, and 4 feature the general format shown in Figure 66.

Table 42. 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

Note: The data register for Port 3, P3, contains 2 bits for P3.0–P3.1, and an additional status bit (P3.7) for carrier signal on/off.

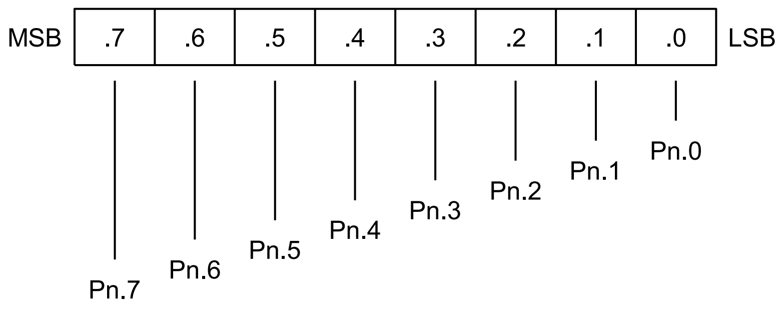


Figure 66. S3F80P5 I/O Port Data Register Format

Note: Because Port 3 is a 2-bit I/O port, the Port 3 Data Register only contains values for P3.0-P3.1. The P3 Register also contains a special carrier on/off bit (P3.7).

10.2. Port 0 Registers

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, or configure the following alternative functions:

- Low-byte pins (P0.0–P0.3): INT7/AD8, X_{TOUT}, X_{TIN}
- High-byte pins (P0.4–P0.7): AD7/COM7, AD6/COM6, AD5/COM5, AD4/COM4

Port 0 provides two 8-bit control registers, P0CONH for P0.4–P0.7 and P0CONL for P0.0–P0.3 (see Tables 43 and 44). A reset clears these P0CONH and P0CONL registers to `00h`, configuring all pins to Input Mode. When P0.3 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 43. Port 0 Control Register High Byte (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 = read only; R/W = read/write.

Bit	**Description**
[7:6]	**P0.7/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).

Bit	Description (Continued)
[5:4]	**P0.6/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.
[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 44. 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 = read only; 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.
[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 45.

Table 45. 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 = read only; 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.
[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 46.

Table 46. 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 = read only; 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.
[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 registers 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.

Note: When programming the port, any alternative peripheral I/O function you configure using the Port 0 Control registers must also be enabled in the associated peripheral module.

Assign pull-up resistors to the pin circuits of individual pins in Port 0 and Port 1. To perform this action, make the appropriate settings to the corresponding pull-up resistor enable registers; P0PUR. These registers are located in Set1, Bank0 at location E7h, and are read/write accessible using register addressing mode. Assign a pull-up resistor to the Port 3 pins, P3.0–P3.1, in the input mode using the basic port configuration setting in the P3CON registers.

Table 47. Port 0 Pull-Up Resistor Enable (P0PUR) Register (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							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Bit	**Description**
[7]	**P0.7 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[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.
Note: Pull-up resistors can be assigned to the Port 3 pins, P3.0 and P3.1 by making the appropriate setting the Port 3 Control (P3CON) Register.	

10.3. Port 1 Registers

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), can be configured to the following alternative functions:

- Low-byte pins (P1.0–P1.3): AD3/TBPWM, AD2/BUZ, AD1/RXD1, AD0/TXD1
- High-byte pins (P1.4–P1.7): TCOUT/TCPWM/COM3, COM2, COM1, COM0

Port 1 has two 8-bit control registers: P1CONH for P1.4–P1.7 and P1CONL for P1.0–P1.3 (see Tables 48 and 49). 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 and enable the alternative functions.

Note: When programming the port,any alternative peripheral I/O function you configure using the Port 1 Control registers must also be enabled in the associated peripheral module.

Table 48. Port 1 Control Register High Byte (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 = read only; 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.

Note: The P1CONH reset value is 0FFh. After a reset, the initial values of ports 1.4–.7 become CMOS input with pull-up mode.

Bit	Description
[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.

Note: The P1CONH reset value is 0FFh. After a reset, the initial values of ports 1.4–.7 become CMOS input with pull-up mode.

The contents of the Port 1 Control Low Byte (P1CONL) Register are described in Table 49.

Table 49. Port 1 Control Register Low Byte (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 = read only; 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 50.

Table 50. 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 = read only; 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.

10.4. Port 2 Registers

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, or can be configured to the following alternative functions:

- Low-byte pins (P2.0–P2.3): PG0/SEG3, PG1/SEG4, PG2/SEG5, PG3/SEG6

Port 2 provides one 8-bit control register, P2CONL for P2.0–P2.3 (see Table 51). 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.

Note: When programming the port, any alternative peripheral I/O function you configure using the Port 2 Control registers must also be enabled in the associated peripheral module.

The contents of the Port 2 Control Low Byte (P2CONL) Register are described in Table 51.

Table 51. Port 2 Control Register Low Byte (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 = read only; R/W = read/write.

Bit	**Description**
[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 224. 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, Set1, Bank0.

The contents of the Port 2 External Interrupt Enable (P2INT) Register are described in Table 52.

Table 52. 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 = read only; R/W = read/write.								

Bit	**Description**
[0]	**P2.0 External Interrupt (INT4) Enable Bit** 0: Disable interrupt. 1: Enable interrupt.

The contents of the Port 2 Output Mode Selection (P2OUTMD) Register are described in Table 53.

Table 53. 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 = read only; R/W = read/write.								

Bit	**Description**
[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 54.

Table 54. 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 = read only; R/W = read/write.								

Bit	**Description**
[0]	**P2.0 External Interrupt (INT4) Pending Flag Bit** No P2.0 external interrupt pending when read. 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 rending flag (P2PND.0–7) has no effect.

The contents of the Port 2 Pull-Up Resistor Enable (P2PUR) Register are described in Table 55.

Table 55. 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 = read only; R/W = read/write.								

Bit	**Description**
[0]	**P2.0 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.

10.5. Port 3 Registers

The contents of the Port 3 Control (P3CON) Register are described in Table 56. The Port 3 Control (P3CON) Register's functions and port assignments are described in Table 56.

Table 56. 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 = read only; R/W = read/write.

Bit	**Description**
[7:6]	**Package Selection and Alternative Function Select Bits** 00**:** 24 pin package. P3.0: T0PWM/T0CAP/T1CAP P3.1: REM/T0CK Others**:** Not available in the S3F80P9 MCU.
[5]	**P3.1 Function Selection Bit** 0: Normal I/O selection. 1: Alternative function enable (REM/T0CK).
[4:3]	**P3.1 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.
[2]	**Function Selection Bit for P3.0** 0: Normal I/O selection. 1: Alternative function enable (P3.0: T0PWM/T0CAP/T1CAP).
[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:
1. 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 S3F80P5 MCU):
   a. Port 3, bit[7]: carrier signal on (1) or off (0).
   b. Port 3, bit[1:0]: P3.1/REM/T0CK pin, bit 0: P3.0/T0PWM/T0CAP/T1CAP pin.
2. The alternative function enable/disable are enabled in accordance with function selection bit (bit[5] and bit[2]).

Table 57 lists specific examples of the alternative functions and pin assignments according to each control bit of the P3CON in the 24-pin package.

Table 57. Function Description and Pin Assignment of P3CON (24-Pin Package)

P3CON						Function Description and Assignment to P3.0–P3.3	
B5	**B4**	**B3**	**B2**	**B1**	**B0**	**P3.0**	**P3.1**
0	x	x	0	x	x	Normal I/O	Normal I/O
0	x	x	1	0	0	T0_CAP/T1_CAP	Normal I/O
0	x	x	1	1	1	T0_CAP/T1_CAP	Normal I/O
0	x	x	1	0	1	T0PWM	Normal I/O
0	x	x	1	1	0	T0PWM	Normal I/O
1	0	0	0	x	x	Normal I/O	TOCK
1	1	1	0	x	x	Normal I/O	TOCK
1	0	1	0	x	x	Normal I/O	REM
1	1	0	0	x	x	Normal I/O	REM
1	0	0	1	0	0	T0_CAP/T1_CAP	TOCK
1	1	1	1	1	1	T0_CAP/T1_CAP	TOCK
1	0	1	1	0	1	T0PWM	REM
1	1	0	1	1	0	T0PWM	REM
1	0	0	1	0	1	T0PWM	Normal Input
1	1	1	1	1	0	T0PWM	Normal Input
1	0	1	1	0	0	T0_CAP/T1_CAP	REM
1	1	0	1	1	1	T0_CAP/T1_CAP	REM

The contents of the Port 3 Output Pull-Up Resistor Enable (P3OUTPU) Register are described in Table 58.

Table 58. Port 3 Output Pull-Up Resistor Enable Register (P3OUTPU; Set1, Bank1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	–	–	–	–	–	–	0	0
R/W	–	–	–	–	–	–	R/W	R/W
Address	F4h							
Note: R = read only; R/W = read/write.								

Bit	**Description**
[7:2]	**Reserved; 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.

Chapter 11. Basic Timer and Timer 0

The S3F80P5 MCUcontains two default timers: the 8-bit basic timer and the 8-bit general-purpose timer/counter. The 8-bit timer/counter is called *Timer 0*. A block diagram of the basic timer and Timer 0 is shown in Figure 67.

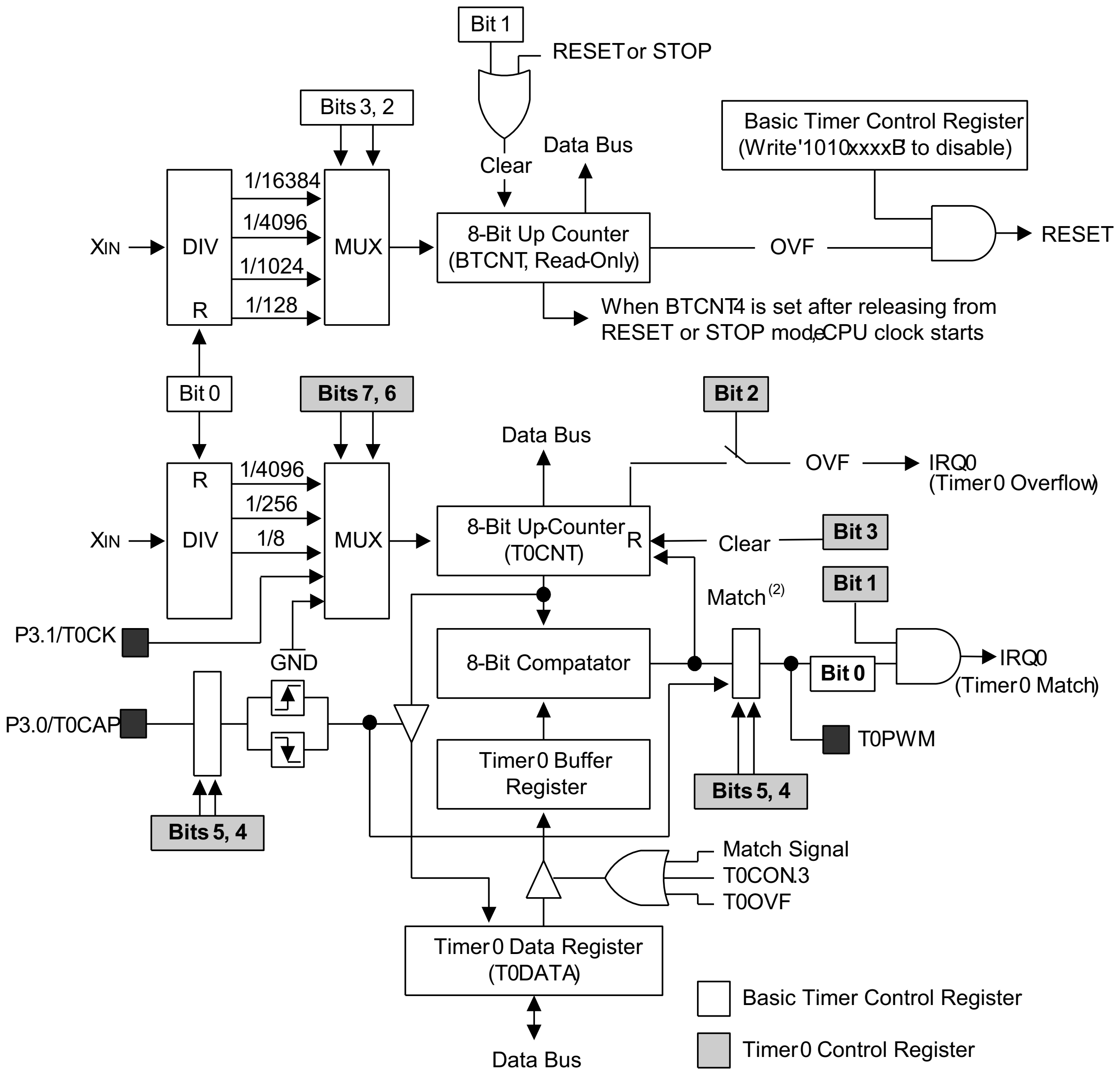


Figure 67. Basic Timer and Timer 0 Block Diagram

Note: In reference to Figure 67, and during a power-on reset operation, the CPU is idle during the required oscillation stabilization interval (until bit 4 of the basic timer counter overflows).

11.1. Basic Timer

The Basic Timer (BT) can be used 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, BTCON (`D3h`, Set1, Bank0, R/W)
- 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 basic timer clock and Timer 0 clock.

11.2. Basic Timer Control Register

The Basic Timer Control (BTCON) Register is used to select the input clock frequency, to clear the basic timer counter and frequency dividers, and to enable or disable the watchdog timer function. It is located in Set1, Bank0, address D3h, and is read/write-addressable using register addressing mode.

A reset clears BTCON to `00h`. This enables the watchdog function and selects a basic timer clock frequency of 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.

Table 59. Basic Timer Control (BTCON) Register (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							
Mode	Register Addressing Mode only							
Note: R = read only; 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 Bit** 0: No effect. 1: Clear the basic timer counter value.
[0]	**Clock Frequency Divider Clear Bit for Basic Timer and Timer 0** 0: No effect. 1: Clear both block frequency dividers.

Notes:
1. When a 1 is written to BTCON.1, the basic timer counter value is cleared to `00h`. Immediately following the write operation, the BTCON.1 value is automatically cleared to 0.
2. When a 1 is written to BTCON.0, the corresponding frequency divider is cleared to `00h`. Immediately following the write operation, the BTCON.0 value is automatically cleared to 0.

11.2.1. Basic Timer Function Description

This section discusses functions of the Watchdog Timer, the Oscillation Stabilization Interval Timer, the Timer 0 Control Register, Interval Timer Mode, Pulse Width Modulation mode, and Capture Mode.

11.2.1.1. Watchdog Timer Function

Program the basic timer overflow signal (BTOVF) to generate a reset by setting BTCON.7–BTCON.4 to any value other than `1010b`. (The `1010b` value disables the watchdog function.) A reset clears BTCON to `00h`, automatically enabling the watchdog timer function. A reset also selects the CPU clock (as determined by the current CLKCON register setting), divided by 4096, as the BT clock.

A reset is generated whenever the basic timer overflow occurs. During normal operation, the application program must prevent overflow, and the accompanying reset operation, from occurring by clearing the BTCNT value (by writing a 1 to BTCON.1) at regular intervals.

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

11.2.1.2. 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 (for reset), or at the rate of the preset clock source (for an external interrupt). When 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 to ensure that it can resume normal operation.

In summary, the following events occur when Stop Mode is released:

1. During Stop Mode, a power-on reset or an external interrupt occurs to trigger a Stop Mode release, and oscillation starts.
2. If a power-on reset occurs, the basic timer counter 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.

11.2.1.3. Configuring the Basic Timer

The following routine 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  ; Non-divided clock
        CLR   SYM           ; Disable global and fast interrupts
        CLR   SPL           ; Stack pointer low byte → 0
                            ; Stack area starts at 0FFh
        •
        •
        •
        SRP    #0C0h         ; Set register pointer → 0C0h
        EI                   ; Enable interrupts
        •
        •
        •
MAIN    LD     BTCON, #52h   ; Enable the watchdog timer
                             ; Basic timer clock: fOSC/4096
                             ; Clear basic timer counter
        NOP
        NOP
        •
        •
        •
        JP     T, MAIN
        •
        •
        •
```

11.3. Timer 0

Timer 0 contains three operating modes, which can be 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 register (T0DATA)
- I/O pins for capture input (T0CAP) or match output

11.3.1. Timer 0 Control Register

Use the Timer 0 Control (T0CON) Register, to:

- Select the Timer 0 operating mode (interval timer, capture mode, or PWM mode)
- Select the Timer 0 input clock frequency
- Clear the Timer 0 counter, T0CNT

- Enable the Timer 0 overflow interrupt or Timer 0 match/capture interrupt
- Clear timer0 match/capture interrupt pending conditions

T0CON is located in Set1, Bank0, at address `D2h`, and is read/write-addressable using register addressing mode.

A reset clears T0CON to `00h`. This sets Timer 0 to normal interval timer mode, selects an input clock frequency of f_{OSC}/4096, and disables all Timer 0 interrupts. Clear the Timer 0 counter 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 timer0 overflow interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware.

To enable the Timer 0 mach/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 timer0 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 timer0 interrupt pending bit, T0CON.0.

Table 60. Timer 0 Control (T0CON) Register (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							
Mode	Register Addressing Mode only							
Note: R = read only; 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 write). 1: Clear T0 counter, T0CNT (when write).
[2]	**Timer 0 Overflow Interrupt Enable Bit**[1] 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]	**Clock Frequency Divider Clear Bit for Basic Timer and Timer 0** 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 (S/W).

Table 61. Timer 0 Data (T0DATA) Register (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	D1h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

11.3.2. Timer 0 Function Description

This section describes the features and functions of the Timer A interrupt.

11.3.3. Timer 0 Interrupts

Timer 0 can generate two interrupts: the Timer 0 overflow interrupt (T0OVF) and the Timer 0 match/capture interrupt (T0INT). T0OVF occurs at interrupt level IRQ0, vector `FAh`. T0INT also occurs at IRQ0, but is assigned the separate vector address `FCh`.

A Timer 0 overflow interrupt (T0OVF) pending condition is automatically cleared by hardware when the interrupt is serviced. The 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.

11.3.4. Interval Timer Mode

In the interval timer mode, a match signal is generated when the counter value is identical to the value written to the T0 reference data register, T0DATA. The match signal generates a Timer 0 match interrupt (T0INT, vector `FCh`) and clears the counter.

If, for example, you write the value `10h` to T0DATA, `0Bh` to T0CON, the counter will increment until it reaches `10h`. At this point, the 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; see Figure 68.

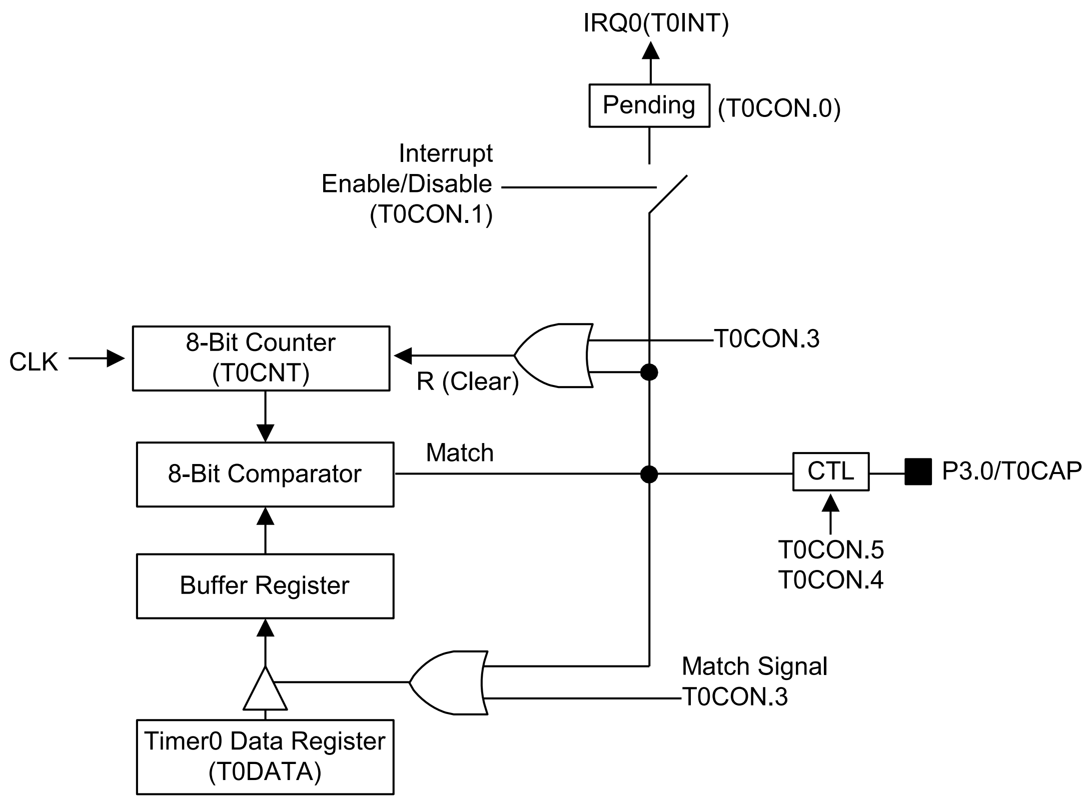


Figure 68. Simplified Timer 0 Function Diagram: Interval Timer Mode

11.3.5. Pulse Width Modulation Mode

Pulse Width Modulation (PWM) Mode lets you program the width (duration) of the pulse that is output at the T0PWM 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`, then continues incrementing from `00h`.

Although the match signal can be used to generate a Timer A overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the T0PWM pin is held Low as long as the reference data value is less than or equal to (≤) the counter value, the pulse is then held High for as long as the data value is greater than (>) the counter value. One pulse width is equal to t_{CLK} x 256; see Figure 69.

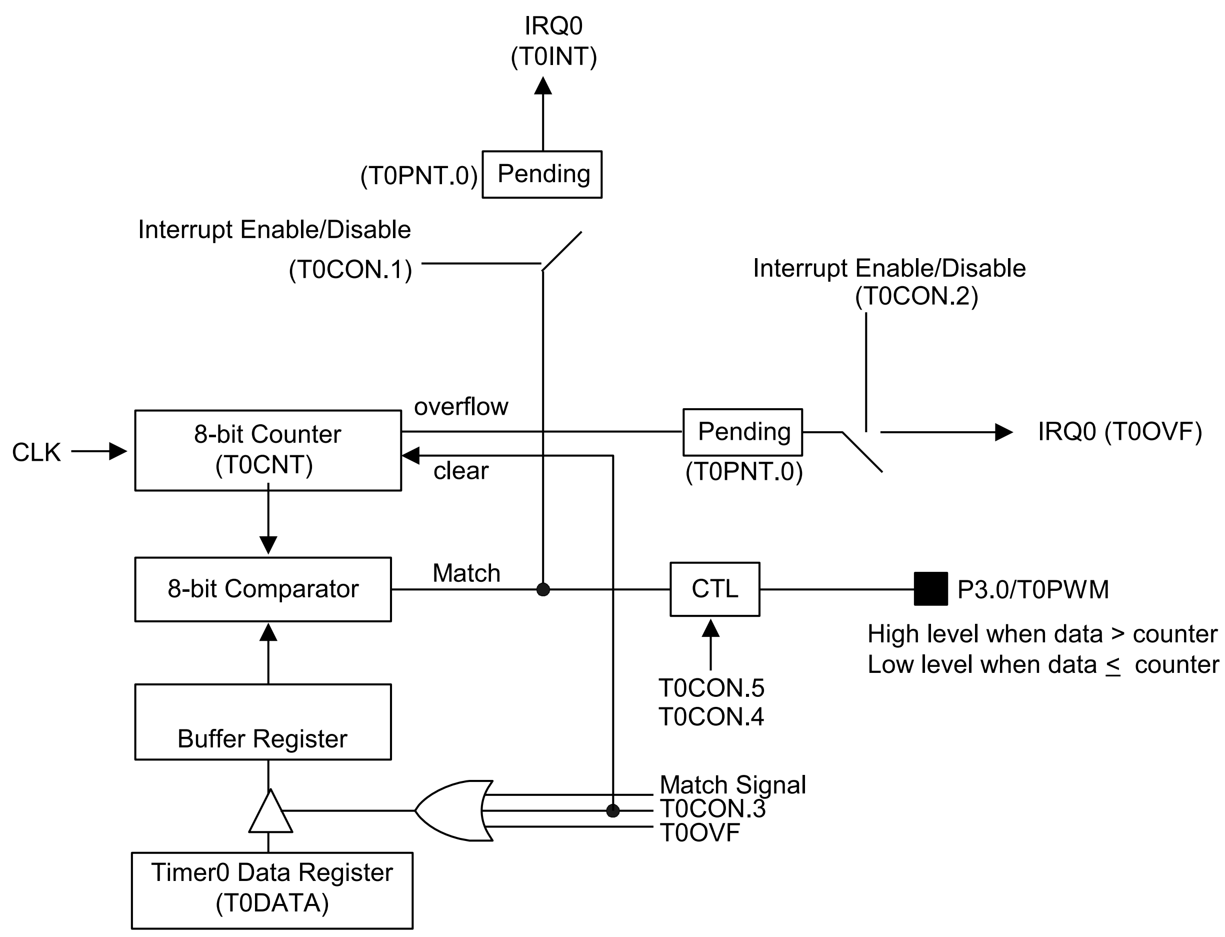


Figure 69. Simplified Timer 0 Function Diagram: PWM Mode

11.3.6. 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 P3CON.2 is 1, the T0CAP input is selected. When P3CON.2 is set to 0, normal I/O port (P3.0) is selected.

Both kinds 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 (duration) of the signal that is being input at the T0CAP pin; see Figure 70.

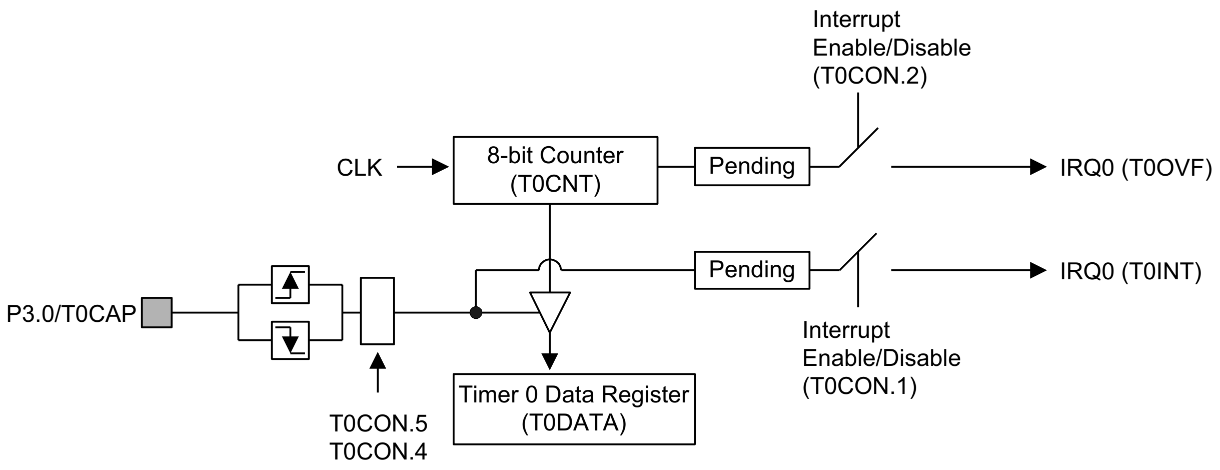

Interrupt Enable/Disable (T0CON.2)
CLK
8-bit Counter (T0CNT)
Pending
IRQ0 (T0OVF)
Pending
IRQ0 (T0INT)
P3.0/T0CAP
Interrupt Enable/Disable (T0CON.1)
Timer 0 Data Register (T0DATA)
T0CON.5
T0CON.4

Figure 70. Simplified Timer 0 Function Diagram: Capture Mode

11.3.6.1. Programming Timer 0

The following sample program sets Timer 0 to interval timer mode, sets the frequency of the oscillator clock, and determines the execution sequence which follows a Timer 0 interrupt.

The program parameters are as follows:

- Timer 0 is used in Interval Mode; the timer interval is set to 4 milliseconds
- Oscillation frequency is 6 MHz
- General register 60h (page 0) → 60h + 61h + 62h + 63h + 64h (page 0) is executed after a Timer 0 interrupt

```
            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 non-divided 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      T0DATA, #5Dh        ; Set timer interval to 4
                                        ; milliseconds
                                        ; (6MHz/256) ÷ (93 + 1) = 0.25 kHz
                                        ; (4 ms)

            SRP     #0C0h               ; Set register pointer → 0C0h
            EI                          ; Enable interrupts
            •
            •
            •
T0INT       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 = 200 ms
            JR      ULT, NO_200MS_SET
            BITS    R1.2                ; Bit setting (61.2h)
NO_200MS_SET:
            LD      T0CON, #42h         ; Clear pending bit
            POP     RP0                 ; Restore register pointer 0 value

T0OVER IRET                             ; Return from interrupt service
                                        ; routine
```

Chapter 12. Timer 1

The S3F80P5 microcontroller contains 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.

Timer 1 contains 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 P3.0 (Select a rising edge, a falling edge, or both edges as the trigger).

In the S3F80P5 interrupt structure, the Timer 1 overflow interrupt has higher priority than the Timer 1 match or capture interrupt.

Note: The CPU clock should be faster than the Timer 1 clock.

Figure 71 illustrates the block diagram for Timer 1.

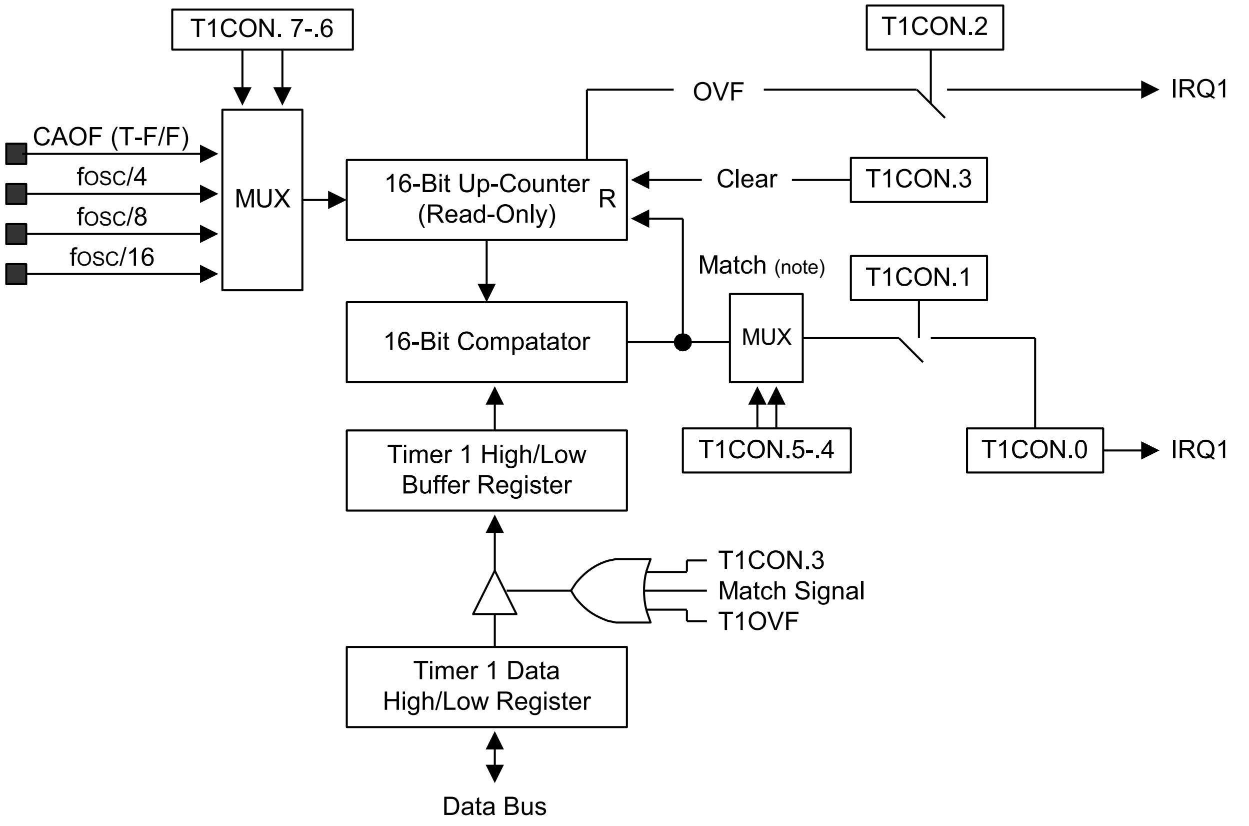


Figure 71. Timer 1 Block Diagram

Note: Match signal only occurs in Interval Mode.

12.1. Timer 1 Overflow Interrupt

Timer 1 can be programmed to generate an overflow interrupt (IRQ1, `F4h`) whenever an overflow occurs in the 16-bit up counter. When Timer 1 overflow interrupt enable bit, T1CON.2, is set to 1, the overflow interrupt is generated each time the 16-bit up counter 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 T1CON.3, the 16-bit counter value can be cleared/reset at any time during the program operation.

12.2. 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, the 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.

Figure 72 illustrates the simplified Timer 1 function diagram in capture mode.

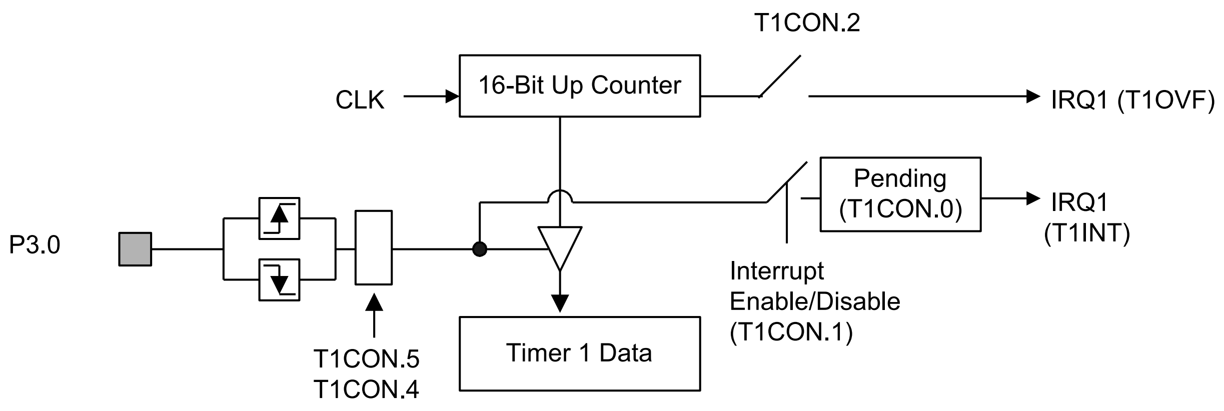


Figure 72. Simplified Timer 1 Function Diagram: Capture Mode

12.3. Timer 1 Match Interrupt

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 T1DATAL. 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 73.

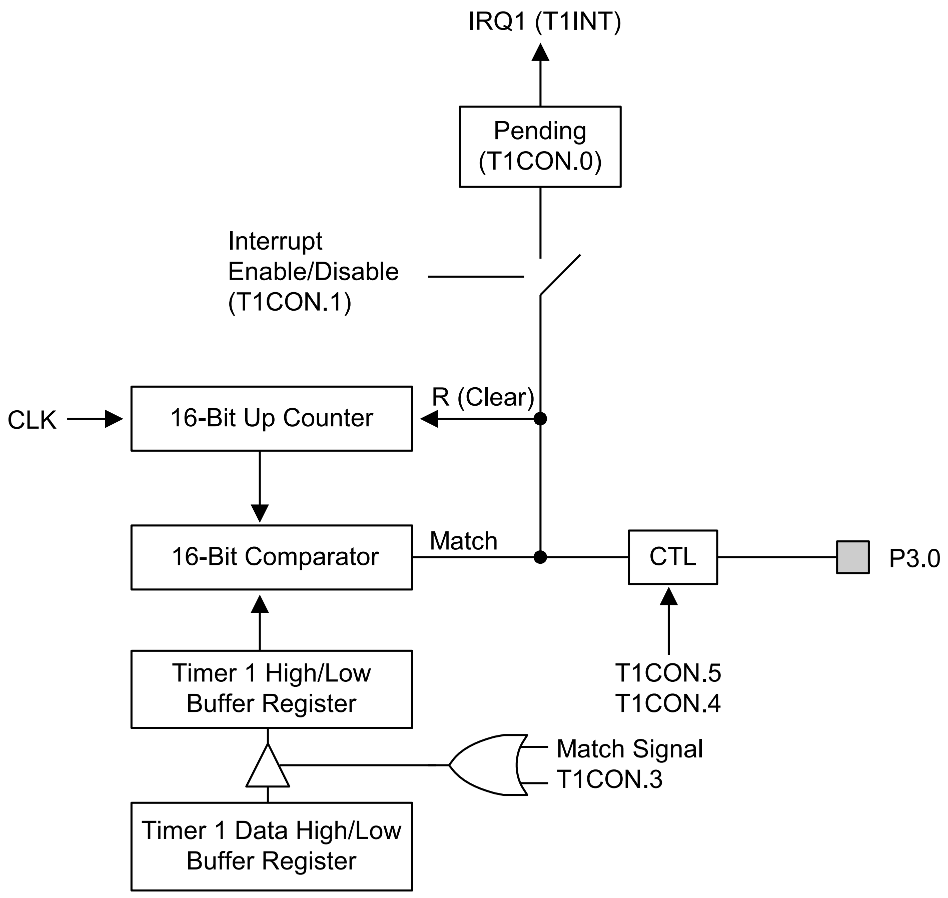


Figure 73. Simplified Timer 1 Function Diagram: Interval Timer Mode

12.4. Timer 1 Control Register

The Timer 1 Control Register, T1CON, is located in Set1, FAh, Bank0 and is read/write-addressable. T1CON contains control settings for the following T1 functions:

- Timer 1 input clock selection
- Timer 1 operating mode selection
- Timer 1 16-bit down counter clear
- Timer 1 overflow interrupt enable/disable
- Timer 1 match or capture interrupt enable/disable

- Timer 1 interrupt pending control (read for status, write to clear)

A reset operation clears T1CON to `00h`, selecting $f_{OSC}/4$ as the T1 clock, configuring Timer 1 as a normal interval Timer, and disabling the Timer 1 interrupts; see Table 62.

Table 62. Timer 1 Control (T1CON) Register (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							
Mode	Register Addressing Mode only							
Note: R = read only; 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 interrupt can occur). 10: Capture mode (falling edges, counter running, OVF interrupt can occur). 11: Capture mode (rising and falling edges, counter running, OVF interrupt can occur).
[3]	**Timer 1 Counter Clear Bit** 0: No effect (when write). 1: Clear T1 counter, T1CNT (when write).
[2]	**Timer 1 Overflow Interrupt Enable Bit**[1] 0: Disable T1 overflow interrupt. 1: Enable T1 overflow interrupt.
[1]	**Timer 1 Match/Capture Interrupt Enable Bit** 0: Disable T1 match/capture interrupt. 1: Enable T1 match/capture interrupt.
[0]	**Timer 1 Match/Capture Interrupt Pending Flag Bit** 0: No T1 match/capture interrupt pending (when read). 0: Clear T1 match/capture interrupt pending condition (when write). 1: T1 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 (i.e., software).

Tables 63 through 66 list the available counter and data registers for Timer 1.

Table 63. Timer 1 Counter High-Byte (T1CNTH) Register (Set1, Bank0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	0	0	0	0	0	0	0	0
R/W	R							
Address	F6h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Table 64. Timer 1 Counter Low-Byte (T1CNTL) Register (Set1, Bank0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	0	0	0	0	0	0	0	0
R/W	R							
Address	F7h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Table 65. Timer 1 Data High-Byte (T1DATAH) Register (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	F8h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Table 66. Timer 1 Data Low-Byte (T1DATAL) Register (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	F9h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Chapter 13. Counter A

The S3F80P5 microcontroller contains one 8-bit counter called *Counter A*. Counter A, which can be used to generate the carrier frequency, contains the following components (See Figure 74):

- Counter A control register, CACON
- 8-bit down counter with auto-reload function
- Two 8-bit reference data registers, CADATAH and CADATAL

Counter A contains two functions:

- As a normal interval timer, generating a Counter A interrupt (IRQ2, vector `ECh`) at programmed time intervals.
- To supply a clock source to the 16-bit timer/counter module, Timer 1, for generating the Timer 1 overflow interrupt.

Note: The CPU clock should be faster than count A clock.

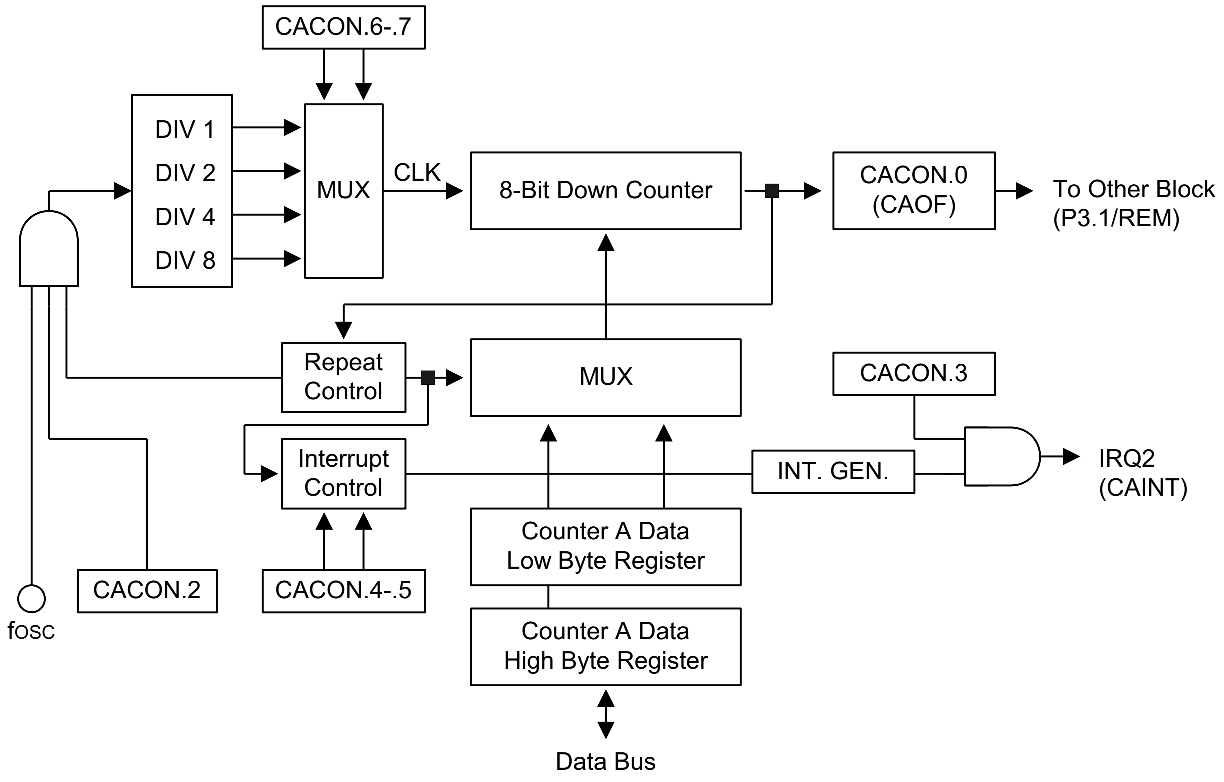


Figure 74. Counter A Block Diagram

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 the CADATAH 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.

13.1. Counter A Control Register

The Counter A Control (CACON) Register is located in `F3h`, Set1, Bank0, and is read/write-addressable.

CACON contains control settings for the following functions (See Table 67):

- Counter A clock source selection

- Counter A interrupt enable/disable
- Counter A interrupt pending control (read for status, write to clear)
- Counter A interrupt time selection

Table 67. Counter A Control (CACON) Register (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							
Mode	Register Addressing Mode only							
Note: R = read only; 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 Time Selection Bits** 00: Elapsed time for low data value. 01: Elapsed time for high data value. 10: Elapsed time for high and low data value. 11: Invalid setting.
[3]	**Counter A Enable Bit** 0: Disable interrupt. 1: Enable interrupt.
[2]	**Counter A Start/Stop Bit** 0: Stop counter A. 1: Start counter A.
[1]	**Counter A Mode Selection Bit** 0: One-Shot Mode. 1: Repeating mode.
[0]	**Counter A Output Flip-Flop Control Bit (CAOF)** 0: T-F/F is low. 1: T-F/F is high.

Tables 68 and 69 list the available data registers for Counter A.

Table 68. Counter A Data High-Byte (CADATAH) Register (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	F4h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Table 69. Counter A Data Low-Byte (CADATAL) Register (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	F5h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

13.1.1. Counter A Pulse Width Calculations

Figure 75 illustrates the pulse width calculations for Counter A.

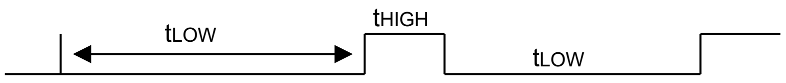


Figure 75. Counter A Pulse Width Calculations

To generate the above repeated waveform consisted of low period time, t_{LOW}, and high period time, t_{HIGH}:

When CAOF = 0,

t_{LOW} = (CADATAL + 2) × 1/Fx. `0h` < CADATAL < `100h`, in which Fx = the selected clock.

t_{HIGH} = (CADATAH + 2) × 1/Fx. `0h` < CADATAH < `100h`, in which Fx = the selected clock.

When CAOF = 1,

t_{LOW} = (CADATAH + 2) × 1/Fx. `0h` < CADATAH < `100h`, in which Fx = the selected clock.

t_{HIGH} = (CADATAL + 2) × 1/Fx. `0h` < CADATAL < `100h`, in which Fx = the selected clock.

To make t_{LOW} = 24 µs and t_{HIGH} = 15µs. f_{OSC} = 4 MHz, FX = 4 MHz/4 = 1 MHz

[Method 1]

When CAOF = 0,

t_{LOW} = 24 µs = (CADATAL+2)/FX = (CADATAL+2) × 1µs, CADATAL = 22.

t_{HIGH} = 15 µs = (CADATAH+2)/FX = (CADATAH+2) × 1µs, CADATAH = 13.

[Method 2]

When CAOF = 1,

t_{HIGH} = 15 µs = (CADATAL+2)/FX = (CADATAL+2) × 1µs, CADATAL = 13.

t_{LOW} = 24 µs = (CADATAH+2)/FX = (CADATAH+ 2) × 1µs, CADATAH = 22.

Figure 76 illustrates the Counter A output Flip-Flop waveforms when the MCU is in Repeat mode.

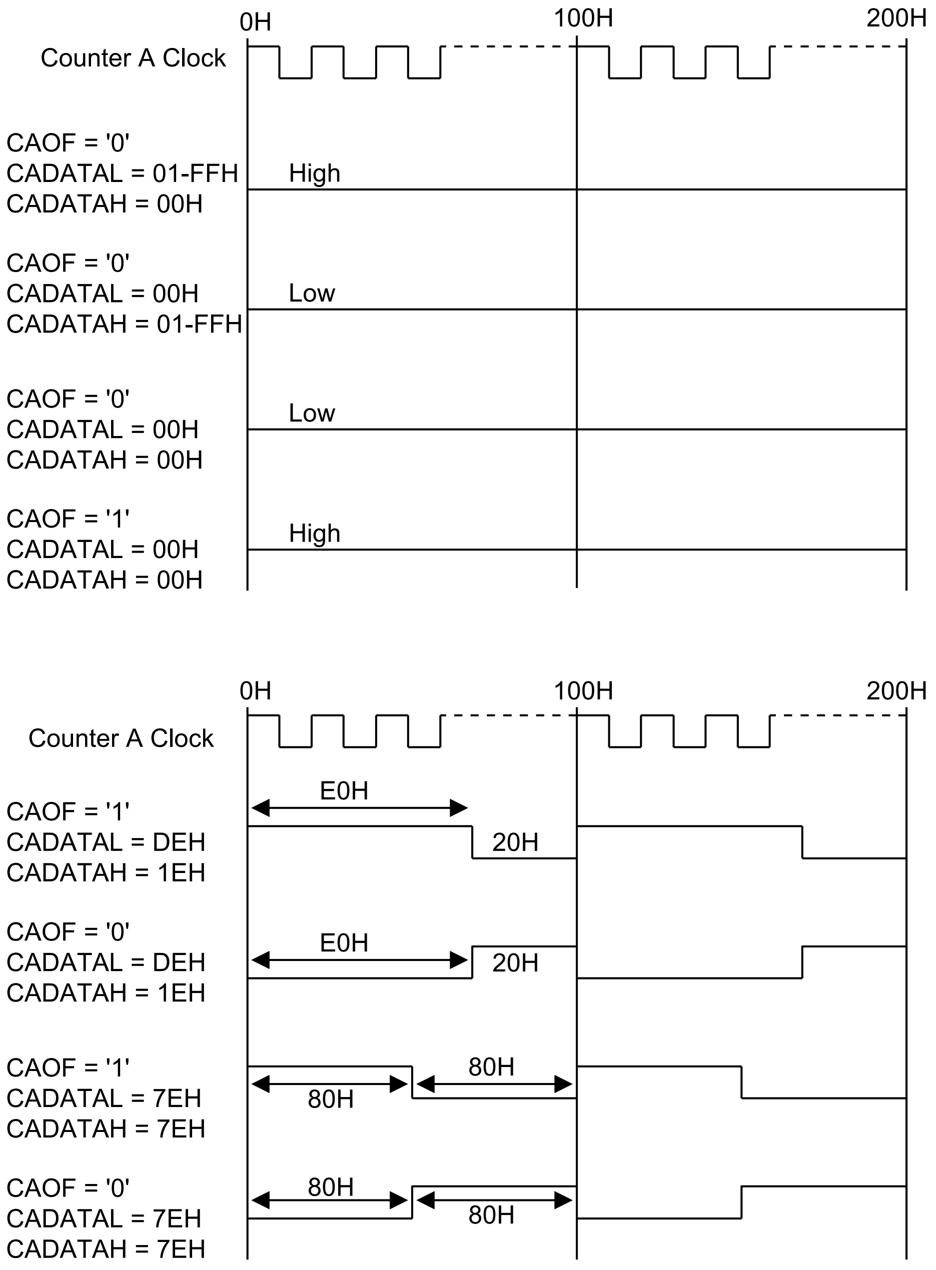


Figure 76. Counter A Output Flip-Flop Waveforms in Repeat Mode

13.1.1.1. Generate 38 kHz, 1/3 Duty Signal Through P3.1

This example sets Counter A to the repeat mode, sets the oscillation frequency as the Counter A clock source, and CADATAH and CADATAL to make a 38 kHz, 1/3 Duty carrier frequency. Figure 77 illustrates the wave function for generating 38 kHz, 1/3 Duty Signal Through P3.1.

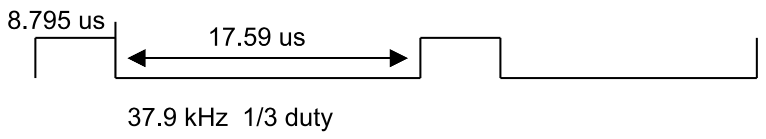


Figure 77. 38 kHz, 1/3 Duty Signal Through P3.1

The program parameters are:

- Counter A is used in repeat mode
- Oscillation frequency is 4 MHz (0.25 µs)
- CADATAH = 8.795 µs/0.25 µs = 35.18
- CADATAL = 17.59 µs/0.25 µs = 70.36
- Set P3.1 CMOS push-pull output and CAOF Mode.

```
        ORG    0100h              ; Reset address
START:  DI
        •
        •
        •
        LD     CADATAL,#(70-2)    ; Set 17.5 ms
        LD     CADATAH,#(35-2)    ; Set 8.75 ms

        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 38 kHz, 1/3
                                  ; duty pulse signal through P3.1.
        •
        •
        •
```

13.1.1.2. Generate a One-Pulse Signal Through P3.1

This example sets Counter A to the One-Shot Mode, sets the oscillation frequency as the Counter A clock source, and CADATAH and CADATAL to make a 40 µs width pulse. Figure 78 illustrates the wave form when generating a one-pulse signal through P3.1.

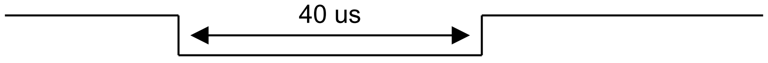


Figure 78. Generate a One-Pulse Signal Through P3.1

The program parameters are:

- Counter A is used in one-shot mode
- Oscillation frequency is 4 MHz (1 clock = 0.25 µs)
- CADATAH = 40 µs/0.25 µs = 160
- CADATAL = 1
- Set P3.1 CMOS push-pull output and CAOF mode

```
            ORG    0100h               ; Reset address
START:      DI

            •
            •

            LD     CADATAH,#(160-2)    ; Set 40 ms
            LD     CADATAL,#1          ; Set any value except 00h

            LD     P3CON,#11110010B    ; Set P3 to CMOS push-pull output.
                                       ; Set P3.1 to REM output

            LD     CACON,#00000001B    ; 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, 0.75 ms is required
                                        ; before the falling edge of
                                        ; the pulse starts.
                        •
                        •
                        •
```

Chapter 14. Timer 2

The S3F80P5 microcontroller contains a 16-bit timer/counter called *Timer 2* (T2). Timer 2 can be used to generate the envelope pattern for the remote controller signal for universal remote controller applications.

- Timer 2 contains 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

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)

Timer 2 can be used 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 P3.0 (select a rising edge, a falling edge, or both edges as the trigger).

The Timer 2 overflow interrupt contains a higher priority than the Timer 2 match or capture interrupt in the S3F80P5 interrupt structure; see Figure 79.

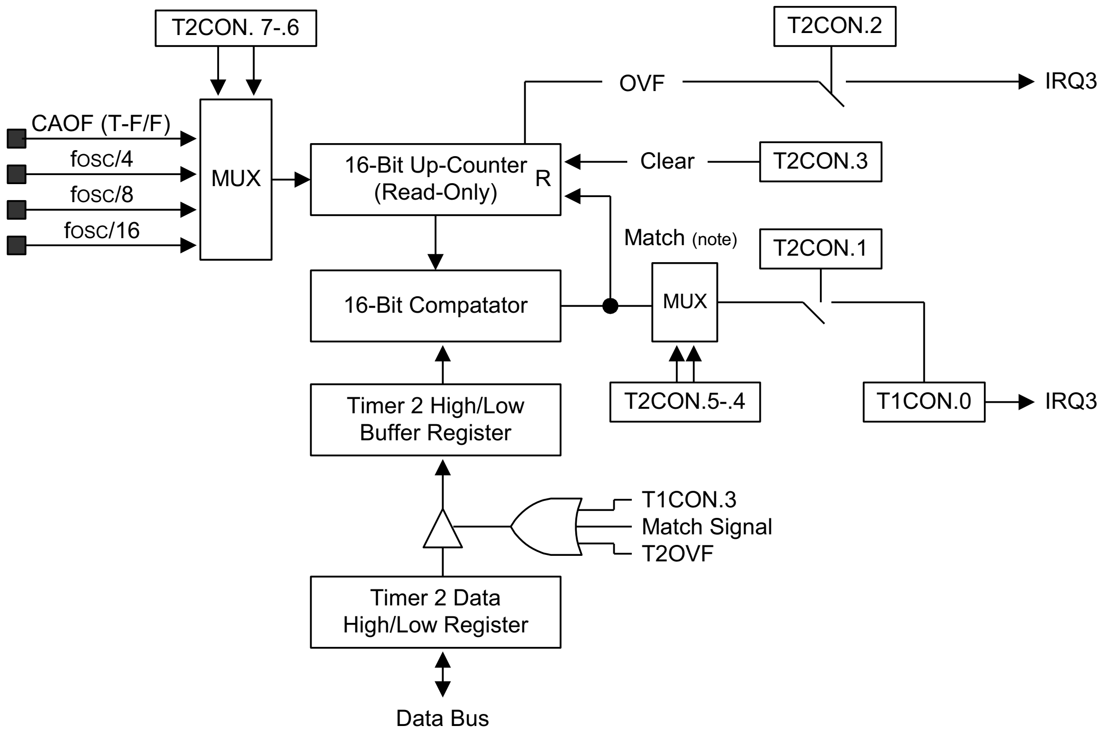


Figure 79. Timer 2 Block Diagram

Notes: 1. A match signal only occurs in Interval Mode.
2. The CPU clock should be faster than the Timer 2 clock.

14.1. Timer 2 Overflow Interrupt

Timer 2 can be programmed to generate an overflow interrupt (IRQ3, `F0h`) whenever an overflow occurs in the 16-bit up counter. When you set the Timer 2 overflow interrupt enable bit, T2CON.2, to 1, the overflow interrupt is generated each time the 16-bit up counter 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, clear/reset the 16-bit counter value at any time during program operation.

14.2. 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 32-pin package and P3.3 pin for 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; see Figure 80.

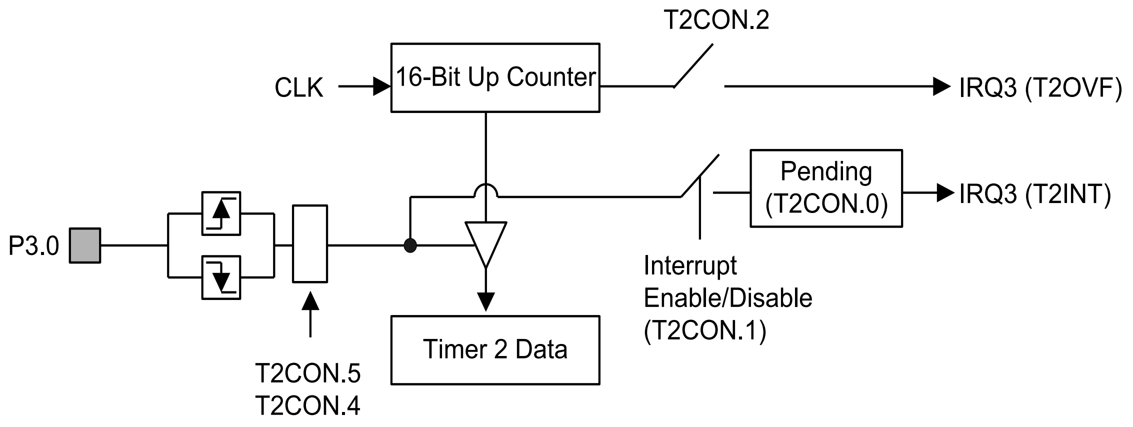


Figure 80. Simplified Timer 2 Function Diagram: Capture Mode

14.3. Timer 2 Match Interrupt

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 T2DATAL. 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; see Figure 81.

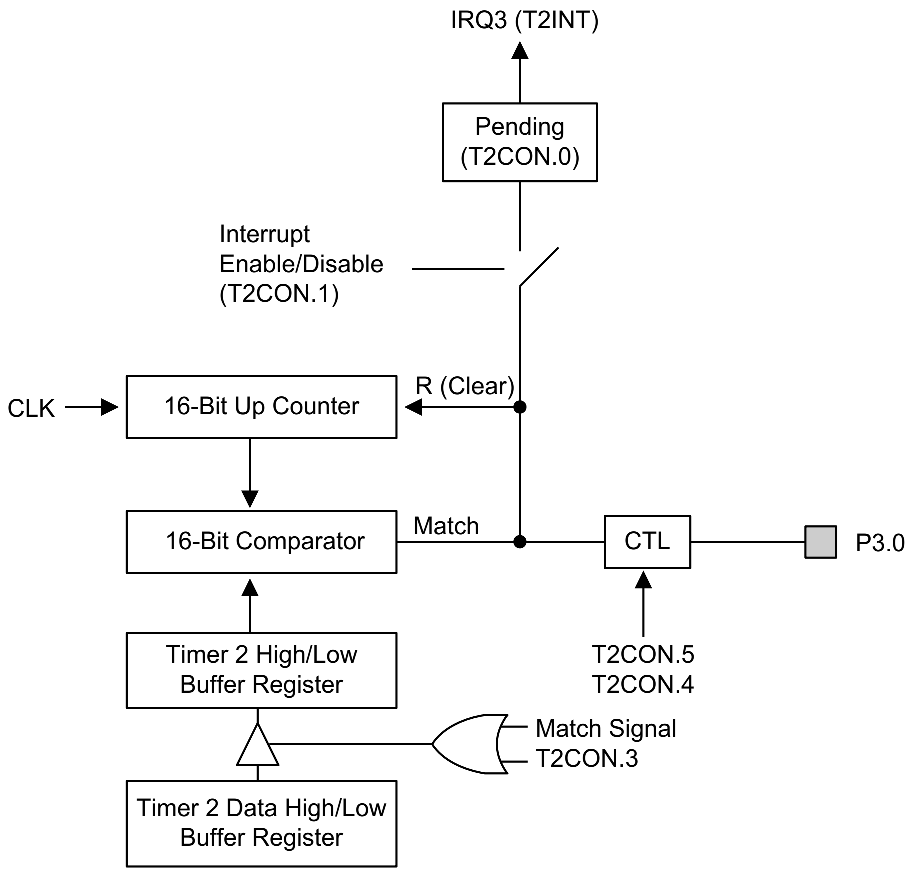


Figure 81. Simplified Timer 2 Function Diagram: Interval Timer Mode

14.4. Timer 2 Control Register

The Timer 2 Control (T2CON) Register is located in address `E8h`, Bank1, Set1 and is read/write-addressable. T2CON contains control settings for the following T2 functions:

- Timer 2 input clock selection
- Timer 2 operating mode selection
- Timer 2 16-bit down counter 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`, selecting f_{OSC} divided by 4 as the T2 clock, configuring Timer 2 as a normal interval timer, and disabling the Timer 2 interrupts; see Table 70.

Table 70. Timer 2 Control (T2CON) Register (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							
Mode	Register Addressing Mode only							
Note: R = read only; 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 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 (S/W).

The available Timer 2 Counter and Data registers for the S3F80P5 MCU are listed in Tables 71 through 74.

Table 71. Timer 2 Counter High Byte (T2CNTH) Register (Set1, Bank1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	0	0	0	0	0	0	0	0
R/W	R							
Address	E4h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Table 72. Timer 2 Counter Low Byte (T2CNTL) Register (Set1, Bank1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	0	0	0	0	0	0	0	0
R/W	R							
Address	E5h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Table 73. Timer 2 Data High Byte (T2DATAH) Register (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	E6h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Table 74. Timer 2 Data Low Byte (T2DATAL) Register (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	E7h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Chapter 15. Embedded Flash Memory Interface

The S3F80P5 MCU contains internal on-chip Flash memory instead of masked ROM. This Flash memory space is sector-erasable and byte-programmable, and is accessed by the *LDC* instruction. Users can program data in this Flash memory area at any time.

The S3F80P5 MCU contains embedded 18KB memory featuring the following two operating modes:

- User Program Mode
- Tool Program Mode: see the S3F80P5 Tool Programming chapter on page 295

15.1. Flash ROM Configuration

S3F80P5 Flash memory consists of 144 sectors. Each sector consists of 128 bytes. Therefore, the total size of Flash memory is 128 × 144 bytes (18KB). Users can erase this Flash memory one sector unit at a time, then write data into Flash memory one byte unit at a time.

- 18KB internal Flash memory
- Sector size: 128 bytes
- 10 years data retention
- Fast programming time:
  - Sector Erase: 4ms (min.)
  - Byte Program: 20µs (min.)

- Byte programmable
- User programmable by LDC instruction
- 128-byte sector erase available
- External serial programming support
- Endurance: 10,000 erase/program cycles (min.)
- Expandable on-board programming

15.2. User Program Mode

This mode supports sector erase, byte programming, byte read and one protection mode (Hard Lock Protection). The S3F80P5 MCU contains an internal pumping circuit to generate high voltage; therefore, supplying 12.5V to the V_{PP} (test) pin is not required. To program a Flash memory in this mode, several control registers are used. There are four kind functions in User Program Mode – programming, reading, sector erase, and one protection mode (hard lock protection).

15.2.1. ISP Sectors

ISP sectors located in the program memory area can store onboard program software (boot program code for upgrading application code by interfacing with I/O port pin). The ISP sectors can not be erased or programmed by the LDC instruction for the safety of the onboard program software.

The ISP sectors are available only when the ISP enable/disable bit is set to 0; i.e., enable ISP at the Smart Option. This area can be used as a normal program memory (can be erased or programmed by LDC instruction) by setting ISP disable bit to 1 at the Smart Option, if electing not to use ISP sector. Even if ISP sector is selected, ISP sector can be erased or programmed in Tool Program Mode by serial programming tools.

The size of ISP sector can be varied by settings of Smart Option (refer to Figure 10 on page 15). Choose appropriate ISP sector size according to the size of onboard program software; see Figure 82 and Table 75.

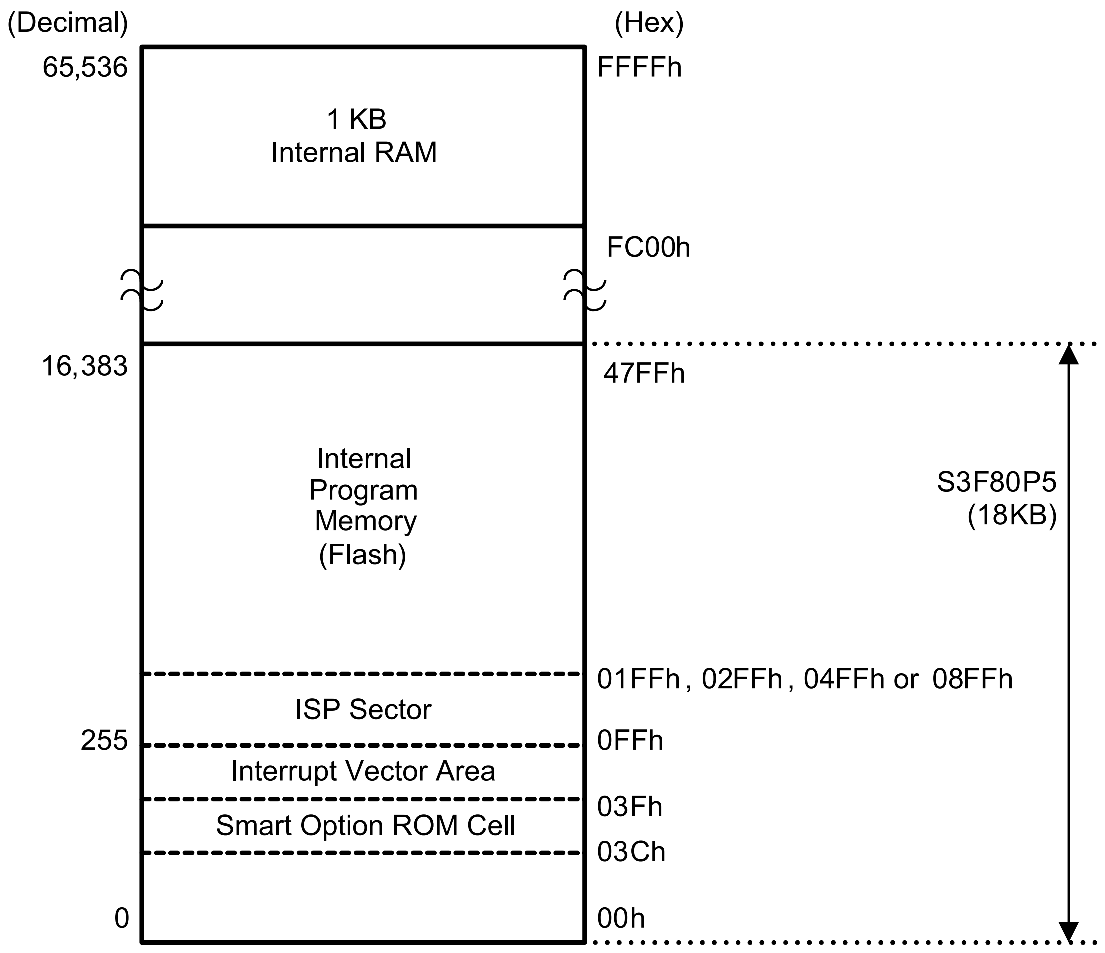


Figure 82. Program Memory Address Space

Table 75. ISP Sector Size

Smart Option (003Eh) ISP Size Selection Bit				
Bit 2	**Bit 1**	**Bit 0**	**Area of ISP Sector**	**ISP Sector Size**
1	x	x	0	0
0	0	0	100h–1FFh (256 bytes)	256 bytes
0	0	1	100h–2FFh (512 bytes)	512 bytes

Note: The area of the ISP sector selected by the Smart Option bit (3Eh.2 – 3Eh.0) can not be erased and programmed by the LDC instruction in User Program Mode.

Table 75. ISP Sector Size

Smart Option (003Eh) ISP Size Selection Bit				
Bit 2	**Bit 1**	**Bit 0**	**Area of ISP Sector**	**ISP Sector Size**
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 the Smart Option bit (3Eh.2 – 3Eh.0) can not be erased and programmed by the LDC instruction in User Program Mode.

15.2.2. ISP Reset Vector and ISP Sector Size

Use ISP sectors by setting the ISP enable/disable bit to 0 and the reset vector selection bit to 0 at the Smart Option, choose the reset vector address of the CPU as listed in Table 76 by setting the ISP reset vector address selection bits; see Figure 10 on page 15.

Table 76. Reset Vector Address

Smart Option (003Eh) ISP Reset Address Selection Bit					
Bit 7	**Bit 6**	**Bit 5**	**Reset Vector Address after IPOR**	**Usable Area for 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 the Smart Option (003Eh.7 – 003Eh.5) is not dependent of the selection of ISP sector size by the Smart Option (003Eh.2 – 003Eh.0).

15.3. Flash Memory Control Registers: User Program Mode

This section describes Flash memory control registers in User Program Mode.

15.3.1. Flash Memory Control Register

The Flash Memory Control (FMCON) Register is available only in User Program Mode to select the Flash memory operation mode; sector erase, byte programming, and to set Flash memory into a hard lock protection mode; see Table 77.

Table 77. Flash Memory Control (FMCON) Register (Set1, Bank1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset	0	0	0	0	–	–	–	0
R/W	R/W							
Address	EFh							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Bit	**Description**
[7:4]	**Flash Memory Mode Selection Bits** 0000–0100: Reserved; must be written to 0. 0101: Programming Mode. 0110: Hard Lock Mode.* 0111– 1001: Reserved; must be written to 0. 1010: Erase Mode. 1011–1111: Reserved; must be written to 0.
[3:1]	**Reserved, not used in the S3F80P5 MCU.**
[3]	**Flash Operation Start Bit (available for Erase and Hard Lock modes only)** 0: Operation stop. 1: Operation start (auto clear bit).
Note: *Hard Lock Mode is one of the Flash memory protection modes.	

FMCON Register bit 0 (FMCON.0) defines the operational startup of the Erase and Hard Lock Protection functions. Therefore, operation of Erase and Hard Lock Protection is activated when FMCON.0 is set to 1. When writing FMCON.0 to 1 for erasing, CPU is stopped automatically for erasing time (min.10ms). After erasing time, CPU is restarted automatically. When reading or programing a byte data from or into Flash memory, this bit is not required to manipulate.

15.3.1.1. Flash Memory User Programming Enable Register

The Flash Memory User Programming Enable (FMUSR) Register is used for the safe operation of Flash memory space. This register protects an improper erase or program operation from malfunctioning of the CPU caused by electrical noise. After reset, User Program Mode is disabled because the value of FMUSR is `00000000b` upon reset. If the operation of Flash memory is necessary, configure User Program Mode by setting the value of FMUSR to `10100101b`. User Program Mode is disabled for FMUSR values other than 10100101b; see Table 78.

Table 78. Flash Memory User Programming Enable (FMUSR) Register (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							
Mode	Register Addressing Mode only							

Note: R = read only; R/W = read/write.

Bit	Description
[7:0]	**Flash Memory User Programming Enable Bit** 10100101: Enable user programming mode. All other values: Disable user programming 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.

15.3.1.2. Flash Memory Sector Address Registers

There are two sector address registers for the erase or programming Flash memory. The Flash Memory Sector Address Low Byte (FMSECL) Register indicates the low byte of sector address and the Flash Memory Address Sector High Byte (FMSECH) Register indicates the high byte of sector address; see Tables 79 and 80.

Each sector consists of 128-bytes. Each sector's address starts `XX00h` or `XX80h`; i.e., a base address of sector is `XX00h` or `XX80h`. Therefore the value of the FMSECL Register bits .6– .0 are *don't care* and do not correspond to the values of either *1* or *0*. To write data into the FMSECH and FMSECL registers when programming Flash memory, the user should program after loading a sector base address, which is located in the destination address. If the next operation is also to write one byte data, the user should check whether the next destination address is located in the same sector or not. Should the address be located in another sector, the user should load a sector address to the FMSECH and FMSECL registers according to the sector; refer to the Sector Erase Examples section on page 271.

Table 79. Flash Memory Sector Address High Byte (FMSECH) Register (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							
Mode	Register Addressing Mode only							
Note: R = read only; 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 80. Flash Memory Sector Address Low Byte (FMSECL) Register (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							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Bit	**Description**
[7:0]	**Flash Memory Sector Address (High Byte)**
Note: The low-byte Flash memory sector address pointer value is the lower eight bits of the 16-bit pointer address.	

15.3.1.3. Sector Erase

Erase Flash memory partially by using the sector erase function only in User Program Mode. The only unit of Flash memory to be erased in User Program Mode is a sector.

The 18KB Flash program memory of the S3F80P5 MCU is divided into 144 sectors; each sector contains 128 bytes. Therefore, the sector to be located destination address should be erased first to program a new data (one byte) into Flash memory. A minimum of 10 ms delay time for the erase is required after setting sector address and triggering erase start bit (FMCON.0). Sector erase is not supported in tool program modes (i.e., using either the MDS mode tool or the programming tool).

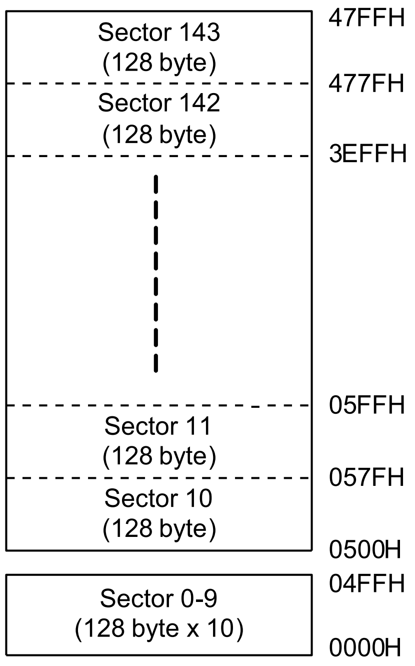


Figure 83. Sector Configurations in User Program Mode

Observe the following procedure to perform a sector erase in User Program Mode; the flow for this routine is shown in Figure 84.

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`.

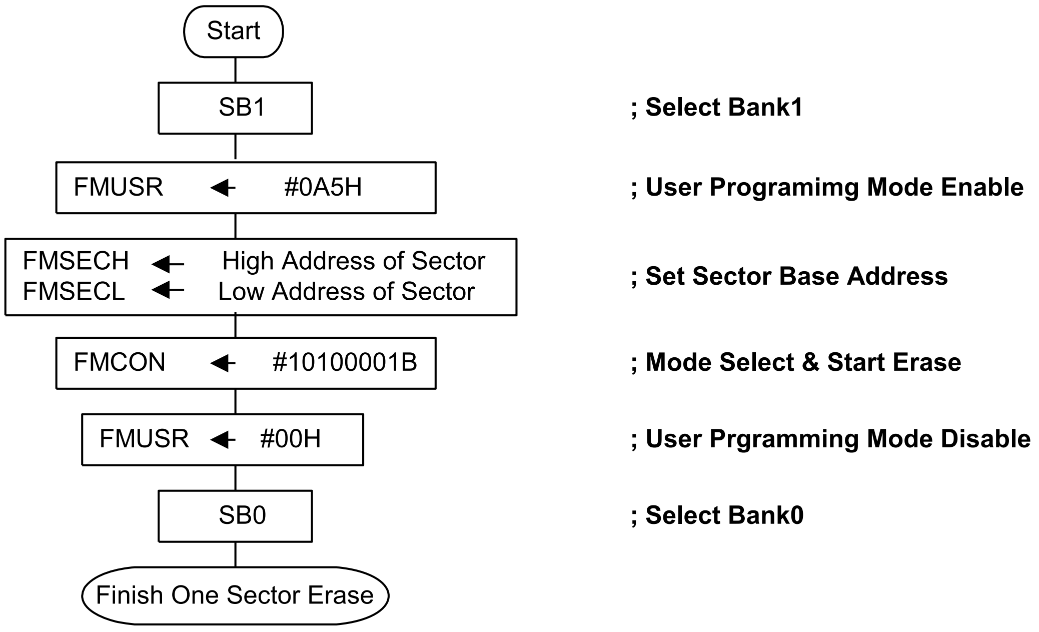

Start
SB1
FMUSR ← #0A5H
FMSECH ← High Address of Sector
FMSECL ← Low Address of Sector
FMCON ← #10100001B
FMUSR ← #00H
SB0
Finish One Sector Erase
; Select Bank1
; User Programimg Mode Enable
; Set Sector Base Address
; Mode Select & Start Erase
; User Prgramming Mode Disable
; Select Bank0

Figure 84. Sector Erase Flowchart in User Program Mode

Notes:

1. If the user erases a sector selected by Flash Memory Sector Address Register FMSECH and FMSECL, the FMUSR should be enabled just before starting sector erase operation. To erase a sector, Flash Operation Start Bit of FMCON register is written from operation stop `0` to operation start `1`. The FMUSR bit will be cleared automatically just after the corresponding operation is completed. In other words, when the S3F80P5 MCU is in the condition that flash memory user programming enable bit is enabled and executes start operation of sector erase, the MCU will get the result of erasing the selected sector as user's a purpose and Flash Operation Start Bit of FMCON register is also clear automatically.
2. If the user executes sector erase operation with FMUSR disabled, FMCON.0 bit, Flash Operation Start Bit remains high, which starts the operation. FMUSR is not cleared even though next instruction is executed. User should be careful setting the FMUSR when executing sector erase, to ensure that no other flash sectors are effected.

15.3.1.4. Sector Erase Examples

Case 1: Erase One Sector

```
ERASE_ONESECTOR:
            SB1
            LD    FMUSR, #0A5h         ; User Program Mode enable
            LD    FMSECH, #            ; Set sector address 4000h,
                                       ; sector 128
            LD    FMSECL, #00h         ; Among sector 0~511
            LD    FMCON, # 10100001b   ; Select Erase Mode enable
                                       ; and Start sector erase

ERASE_STOP  LD    FMUSR, #00h          ; User Program Mode disable
            SB0
```

Case 2: Erase Flash Memory Space from Sector (n) to sector (n+m)

```
Predefine the number of sectors to erase
              LD    SecNumH, #00h        ; Set the sector number.
              LD    SecNumL, # 128       ; Select 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    SecNumL
ERASE_LOOP    CALL  SECTOR_ERASE
              XOR   P4, #11111111b       ; Display ERASE_LOOP cycle
              INCW  RR@
              LD    SecNumH, R2
              LD    SecNumL, R3
              DECW  RR6
              LD    R8, R6
              OR    R8, R7
              CP    R8, #00h
              JP
              ?
              ?
SECTOR_ERASE
              LD    R12, SecNumH
              LD    R14, SecNumL
              MULT  RR12, #80h           ; Calculate the base
                                         ; address of a target
                                         ; sector
              MULT  RR14, #80h           ; The size of one sector
                                         ; is 128 bytes.
              ADD   R1, 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, R11h
              LD    FMCON, # 10100001b  ; Select Erase Mode enable
                                         ; and start sector erase
ERASE_STOP
              LD    FMUSR, #00h          ; User Program Mode
                                         ; disable
              SB0
              RET
```

15.3.1.5. Programming

Flash memory is programmed in a one-byte unit after the sector erase. The write operation of programming starts with an LDC instruction. Observe the following procedure to program in User Program Mode:

1. Erase all target sectors prior to 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 a Flash memory upper address into the upper register of the working register pair.
7. Load a Flash memory lower address into the lower register of the working register pair.
8. Load transmission data to a Flash memory location area on the LDC instruction by Indirect Addressing Mode.
9. Set the Flash Memory User Programming Enable (FMUSR) Register to `00000000b`.

Note: An FMCON.0 value of 0 or 1 is not important in programming mode.

Figures 85 and 86 illustrate the flow of the User Program Mode routine.

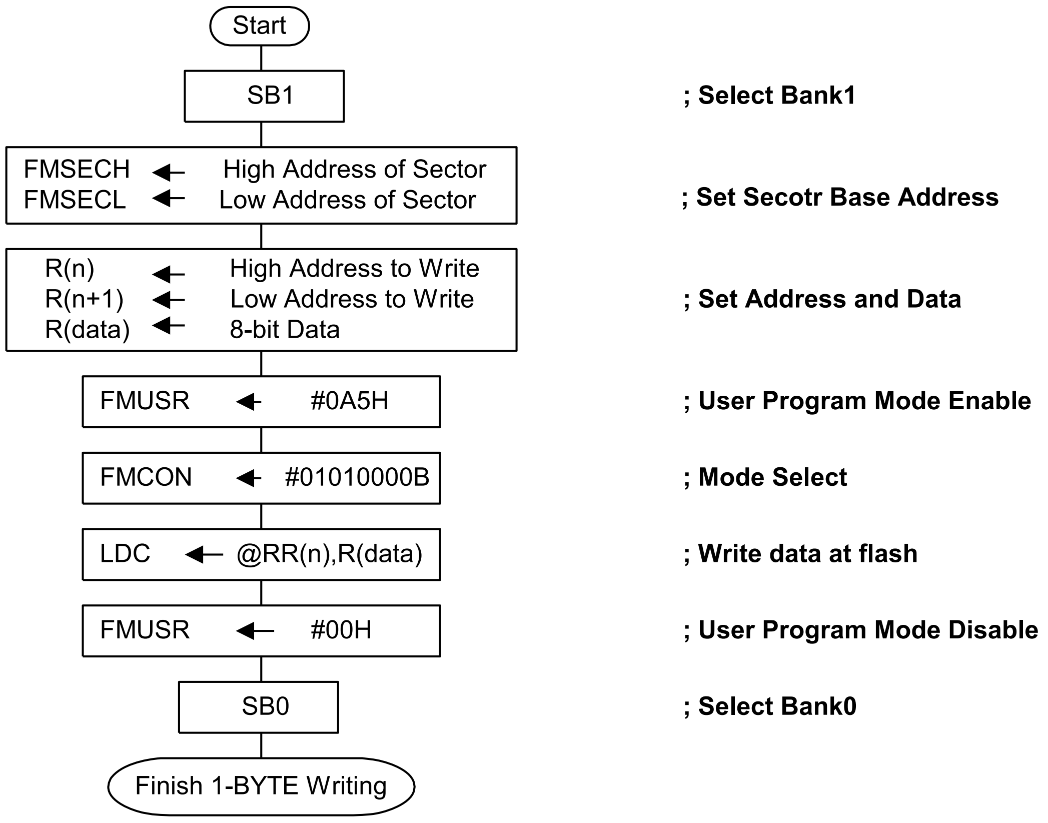
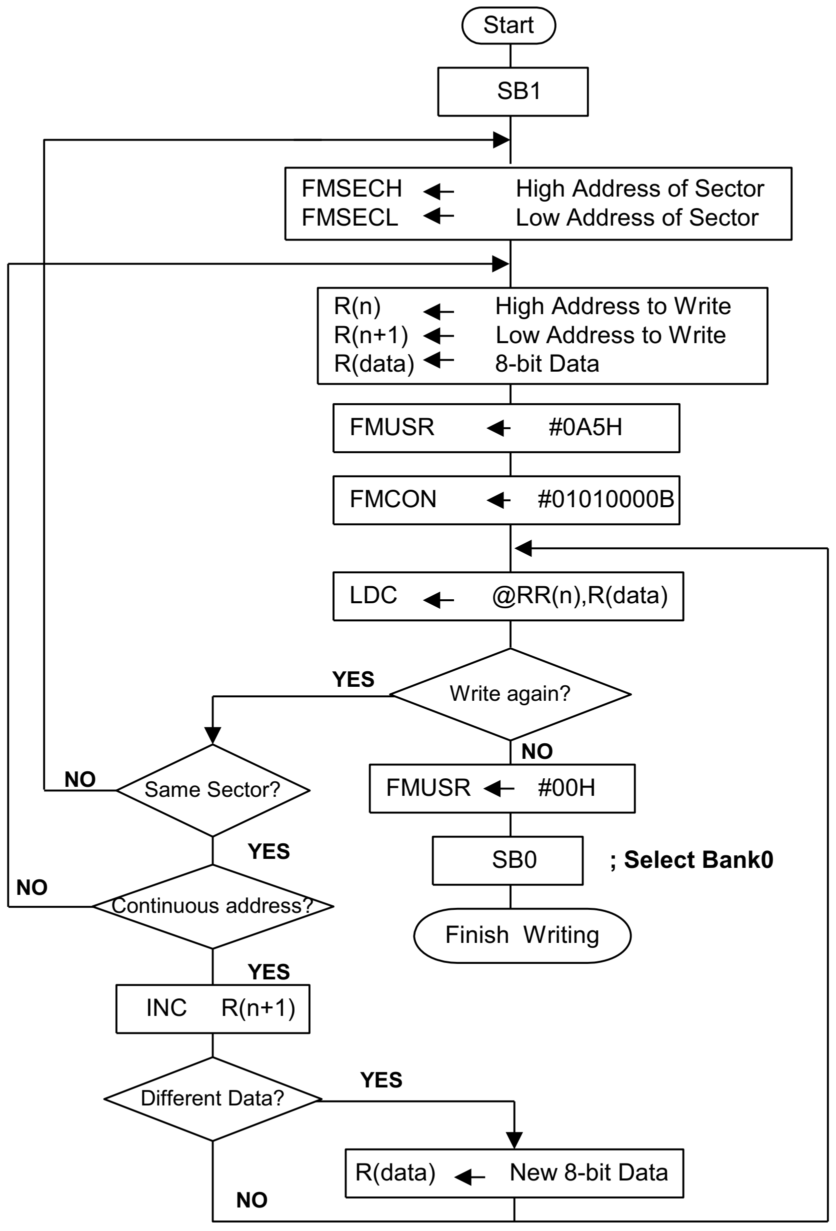


Figure 85. Byte Program Flowchart in a User Program Mode



; Select Bank1

; Set Secotr Base Address

; Set Address and Data

; User Program Mode Enable

; Mode Select

; Write data at flash

; User Program Mode Disable

; User Program Mode Disable
;; Check Sector

;; Check Address

;; Increse Address

;; Update Data to Write

Figure 86. Program Flowchart in a User Program Mode

15.3.1.6. Programming Example Cases

Case 1: One-Byte 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 Flash memory upper
                                       ; address into upper register
                                       ; pair of the working register
          LD     R11,#10h              ; Load Flash memory lower
                                       ; address into lower register
                                       ; pair of the working register
          LDC    @RR10, R9             ; Write data AAh at Flash
                                       ; memory location 4010h

          LD     FMUSR, #00h           ; User Program Mode disable
          SB0
```

Case 2: Programming in the Same 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 the working
                                         ; register pair.
            LD     R11, #40h             ; Load Flash memory lower
                                         ; address into lower
                                         ; register of the working
                                         ; register pair.
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
```

Case 3: Programming to a Flash Memory Space Located in Another Sector

```
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
                                        ; the sector located in
                                        ; the target address to
                                        ; write data
              LD     FMSECL, #00h       ; The base address of
                                        ; sector 2 is 100h
              LD     R9, #0CCh          ; Load data CCh to write
              LD     R10, #01h          ; Load Flash memory upper
                                        ; address into upper
                                        ; register of the pair
                                        ; working register.
              LD     R11, #40h          ; Load Flash memory lower
                                        ; address into lower
                                        ; register of the pair
                                        ; working register.
              CALL   WR_BYTE
              LD     R0, #40h
```

```
WR_INSECTOR50
              LD    FMSECH, #19h        ; Set the base address of
                                          the sector located in
                                          the target address to
                                          write data
              LD    FMSECL, #00h        ; The base address of
                                          sector 50 is 1900h
              LD    R9, #55h            ; Load data 55h to write
              LD    R10, #19h           ; Load Flash memory upper
                                          address into upper
                                          register of the pair
                                          working register.
              LD    R11, #40h           ; Load Flash memory lower
                                          address into lower
                                          register of the pair
                                          working register.
              CALL  WR_BYTE
WR_INSECTOR128LD    FMSECH, #40h        ; Set the base address of
                                          the sector located in
                                          the target address to
                                          write data
              LD    FMSECL, #00h        ; The base address of
                                          sector 128 is 4000h
              LD    R9, #0A3h           ; Load data A3h to write
              LD    R10, #40h           ; Load Flash memory upper
                                          address into upper
                                          register of the pair
                                          working register.
              LD    R11, #40h           ; Load Flash memory lower
                                          address into lower
                                          register of the 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
```

15.3.1.7. Reading

The read operation starts with an LDC instruction. Observe the following procedure to program in User Program Mode.

1. Load a Flash memory upper address into the upper register of the working register pair.
2. Load a Flash memory lower address into the lower register of the working register pair.
3. Load receive data from Flash memory location area on the LDC instruction by using indirect addressing mode.

Example 7. Reading Example

```
      LD    R2, #03h  ; Load Flash memory upper address into upper
                      ; register of the working register pair.
      LD    R3, #00h  ; Load Flash memory lower address into lower
                      ; register of the working register pair.
LOOP  LDC   R0, @RR2  ; Read data from Flash memory location
                      ; (between 33h and 3FFh).
      INC   R3, #0FFh
      JP    NZ, LOOP
```

15.3.1.8. Hard Lock Protection

Set Hard Lock Protection by writing `0110b` in FMCON7–4. This function prevents the changing of data in a Flash memory area. If this function is enabled, writing or erasing the data in a Flash memory area is not possible.

Hard Lock Protection can be released by executing a chip erase in Tool Program Mode. The procedure for setting the Hard Lock Protection mode is performed after the User Program Mode. In tool mode, the manufacturer of the serial tool writer could support Hardware Protection. Please refer to the manual of the serial program writer tool provided by the manufacturer.

Programming Procedure in User Program Mode

1. Set the Flash Memory User Programming Enable (FMUSR) Register to `10100101b`.
2. Set the Flash Memory Control (FMCON) Register to `01100001b`.
3. Set the Flash Memory User Programming Enable (FMUSR) Register to `00000000b`.

Example 8. 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 Detector

The S3F80P5 microcontroller contains a built in Low Voltage Detector (LVD) circuit, which allows LVD and LVD_FLAG detection of power voltage. The S3F80P5 MCU contains two options in LVD and LVD_FLAG voltage level according to the operating frequency to be set by the Smart Option (Refer to Figure 10 on page 15).

Operating Frequency 8 MHz:

- Low voltage detect level for Backup Mode and Reset (LVD): 1.65 V (Typ.) ±50 mV
- Low voltage detect level for Flash Flag Bit (LVD_FLAG): 1.90, 2.00, 2.10, 2.20 V (Typ.) ±100 mV

After a power-on, the LVD block is always enabled. the LVD block is only disabled when a stop instruction is executed. The LVD block of S3F80P5 MCU consists of two comparators and a resistor string. One of comparators is used for LVD detection, and the other is used for LVD_FLAG detection.

16.1. LVD

The LVD circuit supplies two operating modes using one comparator: Backup Mode input and system reset input. The S3F80P5 MCU can enter the Backup Mode and generate the reset signal by the LVD level detection using the LVD circuit. When the LVD circuit detects the LVD level in falling power, the S3F80P5 MCU enters the Backup Mode. Backup Mode input automatically creates a chip stop state. When the LVD circuit detects the LVD level in rising power, the system reset occurs. This reset by LVD circuit is one of the S3F80P5 MCU reset sources; see Figure 59 on page 191.

16.2. LVD FLAG

The output for the other comparator allows the LVD indicator flag bit to return a 1 or a 0. This result is used to indicate a low voltage level. When the power voltage is below the LVD_FLAG level, bit 0 of the LVDCON Register is set to 1. When the power voltage is above the LVD_FLAG level, bit 0 of the LVDCON Register is set to 0 automatically. LVDCON.0 can be used flag bit to indicate low battery in IR application or others.

Notes:
1. LVD is a symbol of parameter that means Low Level Detect Voltage for Backup Mode.
2. LVD_FLAG is a symbol of parameter that means Low Level Detect Voltage for the Flag Indicator.

3. The voltage gaps (LVD_GAPn (n = 1 to 4)) between LVD and LVD FLAGn (n = 1 to 4) include a ± 80 mV distribution. LVD and LVD FLAGn (n = 1 to 4) are not overlapped.

Symbol	Min.	Type	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

Symbol	Min.	Type	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 81 lists the LVD enable time for the S3F80P5 MCU.

Table 81. LVD Enable Time

Parameter	Symbol	Conditions	Min.	Type	Max.	Unit
LVD Enable Time	t_{LVD}	V_{DD}= 1.4V	–	–	50	µs
Note: T_A= –25°C to 85°C						

The LVD turns off in Stop Mode. When an external interrupt occurs, the LVD requires t_{LVD} during max.50µs to wake up. If V_{DD} is below V_{LVD} after external interrupt, chip goes into back-up. Because t_{LVD} time is not enough to start oscillation, chip is not operated to abnormal state; see Figure 87.

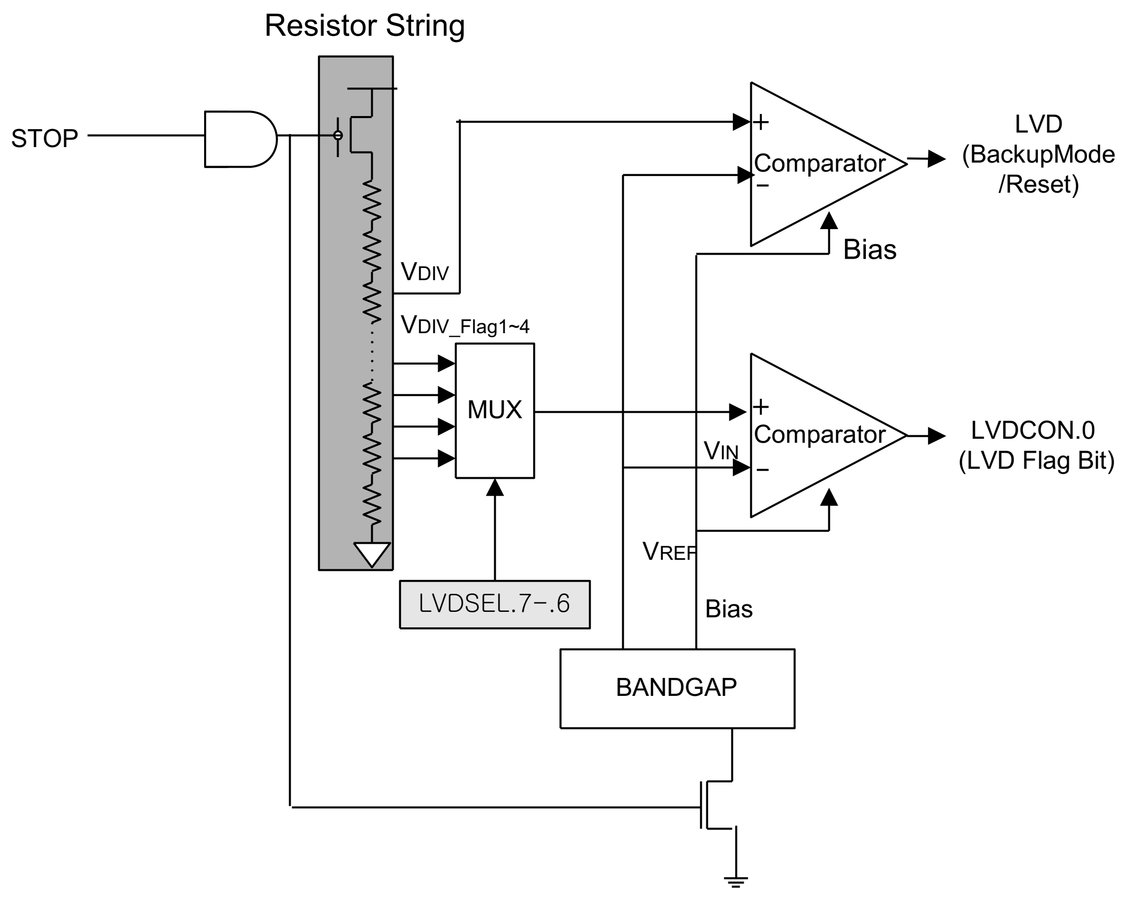


Figure 87. Low Voltage Detect (LVD) Block Diagram

16.2.0.1. Low Voltage Detector Control Register

In the Low Voltage Detector Control (LVDCON) Register, bit 0 (LVDCON.0) is used as the flag bit to indicate a low battery condition in IR and other applications. When the LVD circuit detects this LVD_FLAG, the LVDCON.0 flag bit is set automatically. The reset value of LVDCON is `#00h`; see Table 82.

Table 82. LVD Control (LVDCON) Register (Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	–	–	–	–	–	–	–	0
R/W	R/W							
Address	E0h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Bit	Description
[7:1]	**Reserved, not used in the S3F80P5 MCU.**
[0]	**LVD flag indicator bit.** 0: $V_{DD} \geq$ LVD_FLAG level. 1: $V_{DD} <$ FLAG level.
Note:	When the LVD detects the LVD_FLAG level, the LVDCON.0 flag bit is set automatically. When the V_{DD} is upper LVD_FLAG level, the LVDCON.0 flag bit is cleared automatically.

16.2.0.2. Low Voltage Detector Flag Selection Register

The Low Voltage Detector Flag Selection (LVDSEL) Register is used to select the LVD flag level. The reset value of LVDSEL is `#00h`; see Table 83.

Table 83. LVD Flag Selection (LVDSEL) Register (Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	–	–	–	–	–	–
R/W	R/W							
Address	F1h							
Mode	Register Addressing Mode only							
Note: R = read only; 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, not used in the S3F80P5 MCU.**

Chapter 17. Electrical Data

In this section, electrical characteristics for the S3F80P5 are presented in tables and graphs.

The information is arranged in the following order:

- Absolute Maximum Ratings – see Table 84 on page 283
- D.C. Electrical Characteristics – see Table 85 on page 284
- Characteristics of Low Voltage Detect Circuit – see Table 86 on page 285
- Low Voltage Detect Enable Time – see Table 87 on page 286
- Power On Reset Circuit – see Table 88 on page 286
- Data Retention Supply Voltage in Stop Mode – see Table 89 on page 287
- Stop Mode to Normal Mode Timing Diagrams – see Figures 88 and 89, beginning on page 288
- A.C. Electrical Characteristics – see Table 90 on page 288
- Input Timing for External Interrupts – see Figure 90 on page 288
- Oscillation Characteristics – see Table 91 on page 289
- Input/Output Capacitance – see Table 92 on page 289
- Oscillation Stabilization Time – see Table 93 on page 289
- Operating Voltage Range – see Figure 91 on page 290
- A.C. Electrical Characteristics for Internal Flash ROM – see Table 94 on page 291
- Electrostatic Discharge Characteristics – see Table 95 on page 291

Table 84. Absolute Maximum Ratings (T_A = 25°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	

Table 84. Absolute Maximum Ratings (T_A = 25°C) (Continued)

Parameter	Symbol	Conditions	Rating TBD	Unit
Operating Temperature	T_A	–	–25 to +85	°C
Storage Temperature	T_{STG}	–	–65 to +150	°C

Table 85. DC Electrical Characteristics (T_A = –25°C to +85°C, V_{DD} = 1.60V to 3.6V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Operating Voltage	V_{DD}	f_{OSC} = 4MHz, 8MHz	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_{IH3}	X_{IN}	V_{DD}–0.3	–	V_{DD}	
Input Low Voltage	V_{IL1}	All input pins except V_{IL3}	0	–	0.2 V_{DD}	
	V_{IL3}	X_{IN}	–	–	0.3	
Output High Voltage	V_{OH1}	V_{DD} = 1.70V, I_{OH} = –6mA, Port 3.1 only	V_{DD}–0.7	–	–	
		V_{DD} = 3.3V, I_{OH} = –15mA, Port 3.1 only	V_{DD}–0.7	–	–	
	V_{OH2}	V_{DD} = 1.70V, I_{OH} = –2.2mA, P3.0 and P2.0	V_{DD}–0.7	–	–	
		V_{DD} = 3.3V, I_{OH} = –6.6mA, P3.0 and P2.0	V_{DD}–0.7	–	–	
	V_{OH3}	V_{DD} = 1.70V, I_{OH} = –1mA, Port 0 and Port 1	V_{DD}–1.0	–	–	
		V_{DD} = 3.3V, I_{OH} = –3 mA, Port 0 and Port 1	V_{DD}–0.7	–	–	
Output Low Voltage	V_{OL1}	V_{DD} = 1.70V, I_{OL} = 8 mA, Port 3.1 only	–	0.4	0.5	
	V_{OL2}	V_{DD} = 1.70V, I_{OL} = 5 mA, P3.0 and P2.0	–	0.4	0.5	
	V_{OL3}	V_{DD} = 1.70V, I_{OL} = 2 mA, Port 0 and Port 1	–	0.4	1.0	
Input High Leakage Current	I_{LIH1}	V_{IN} = V_{DD}; all input pins except I_{LIH2}, X_{OUT}	–	–	1	µA
	I_{LIH2}	V_{IN} = V_{DD}, X_{IN}	–	–	20	
Input Low Leakage Current	I_{LIL1}	V_{IN} = 0V, all input pins except I_{LIL2}, X_{OUT}	–	–	–1	
	I_{LIL2}	V_{IN} = 0V, X_{IN}	–	–	–20	
Output High Leakage Current	I_{LOH}	V_{OUT} = V_{DD}, all output pins	–	–	1	

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 (Flash erase/write/read operation).
3. The adder by LVD on current in Backup Mode is 18 µA.
4. Backup Mode voltage is V_{DD} between LVD and POR.
5. LVD on current in Backup Mode V_{DD} = 1.60V; typ. = 18 µA, max. = 35 µA.

Table 85. DC Electrical Characteristics (T_A = –25°C to +85°C, V_{DD} = 1.60V to 3.6V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Output Low Leakage Current	I_{LOL}	V_{OUT} = 0V, all output pins	–	–	–1	µA
Pull-Up Resistors	R_{L1}	V_{IN} = 0V, V_{DD} = 2.35V, T_A = 25 °C, Ports 0–4 except P3.2/P3.3	44	67	95	kΩ
	R_{L2}	V_{IN} = 0V, V_{DD} = 2.35V, T_A = 25 °C, nRESET	150	500	1000	kΩ
Feedback Resistor	Rfd	V_{IN} = V_{DD}, V_{DD} = 1.80V, T_A = 25 °C, X_{IN}	300	700	1500	kΩ
Supply Current[1]	I_{DD1}	Operating Mode[2]; V_{DD} = 3.6V 8MHz crystal	–	3	6	mA
	I_{DD2}	Idle Mode; V_{DD} = 3.6V; 8MHz crystal	–	1	2	
	I_{DD3}	Stop Mode; LVD OFF, V_{DD} = 3.6V	–	0.7	5	µA
	I_{DD12}	Operating Mode, V_{DD} = 3.6V; 4MHz crystal	–	1.5	3	mA
	I_{DD22}	Idle Mode, V_{DD} = 3.6V; 4MHz crystal	–	0.5	1	
LVD on current in Backup Mode when V_{DD} = 1.60V			–	18	35	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 (Flash erase/write/read operation).
3. The adder by LVD on current in Backup Mode is 18 µA.
4. Backup Mode voltage is V_{DD} between LVD and POR.
5. LVD on current in Backup Mode V_{DD} = 1.60V; typ. = 18 µA, max. = 35 µA.

Table 86. Low Voltage Detect Circuit (T_A = –25°C to +85°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
	LVD_GAP1	–	170	250	330	mV
	LVD_GAP2	–	270	350	430	mV

Note: The LVD_GAPn [n = 1 to 4] voltage gaps between LVD and LVD FLAGn (n = 1 to 4) feature ±80mV distribution. LVD and LVD FLAGn (n = 1 to 4) are not overlapped. The variation of the LVD FLAGn (n = 1 to 4) and LVD always is shifted in same direction. Essentially, if one chip exhibits a positive tolerance (for example, +50mV) in LVD FLAG, LVD has a positive tolerance.

Table 86. Low Voltage Detect Circuit (T_A = –25°C to +85°C) (Continued)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Low Level Detect Voltage for Flag Indicator (cont'd.)	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

Note: The LVD_GAPn [n = 1 to 4] voltage gaps between LVD and LVD FLAGn (n = 1 to 4) feature ±80mV distribution. LVD and LVD FLAGn (n = 1 to 4) are not overlapped. The variation of the LVD FLAGn (n = 1 to 4) and LVD always is shifted in same direction. Essentially, if one chip exhibits a positive tolerance (for example, +50mV) in LVD FLAG, LVD has a positive tolerance.

Table 87. LVD Enable Time (T_A = –25°C to +85°C)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
LVD enable time	t_{LVD}	V_{DD} = 1.4V	–	–	50	µs

In Stop Mode, LVD turns off. When external interrupt occurs, LVD requires t_{LVD} during max.50µs 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 88. Power On Reset Circuit (T_A = –25°C to +85°C)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Power on reset (POR) Voltage	V_{POR}	–	0.8	1.1	1.4	V

Table 89. Data Retention Supply Voltage in Stop Mode (T_A = –25°C to +85°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 µA), the data retention could be not be guaranteed.

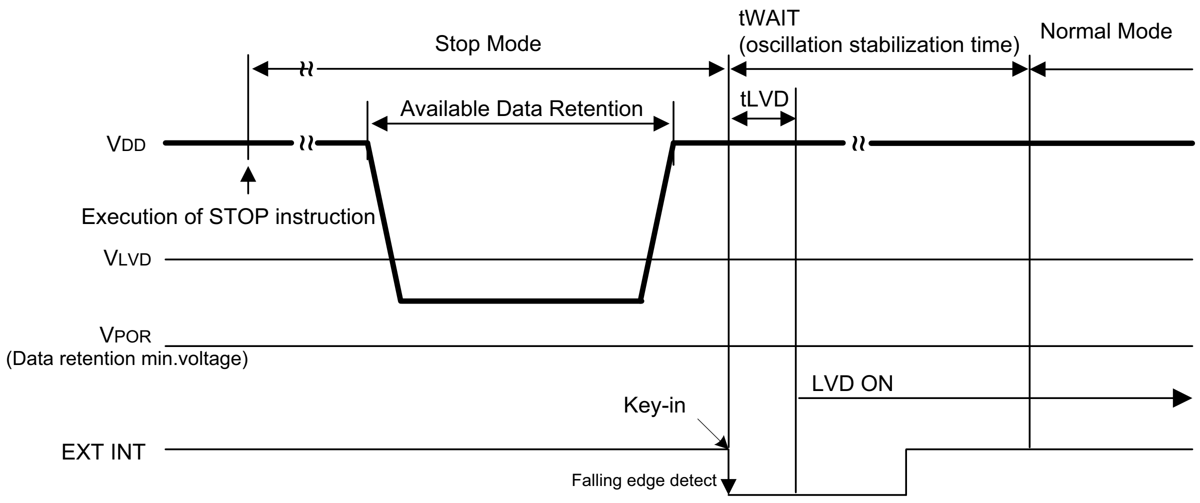


Figure 88. Stop Mode to Normal Mode Timing Diagram 1

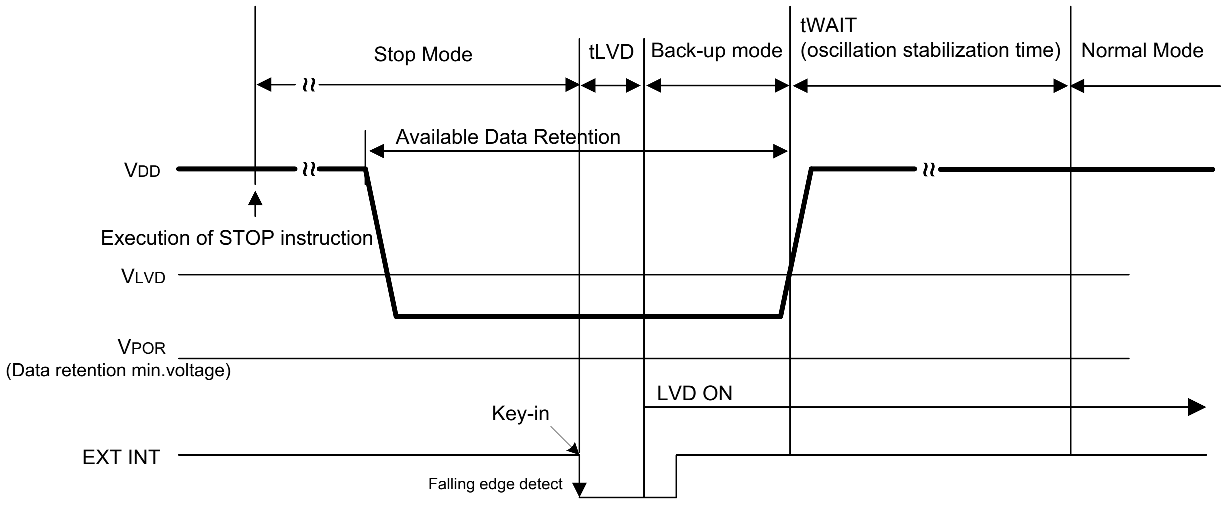


Figure 89. Stop Mode to Normal Mode Timing Diagram 2

Table 90. AC Electrical Characteristics (T_A = –25°C to +85°C)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Interrupt Input High, Low Width	t_{INTH}, t_{INTL}	P0.0–P0.7, P2.0 V_{DD} = 3.6V	200	300	–	ns

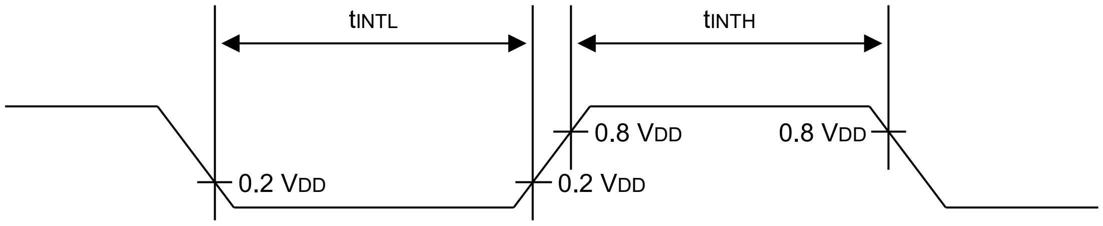
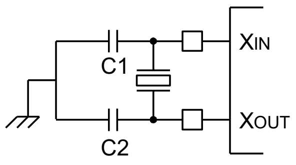
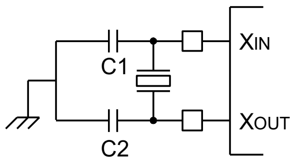
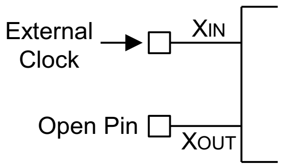


Figure 90. Input Timing for External Interrupts (Port 0 and Port 2)

Table 91. Oscillation Characteristics (T_A = –25°C to +85°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

Note: *Includes an oscillator tolerance of ±1%.

Table 92. Input/Output Capacitance (T_A = –25°C to +85°C)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Input Capacitance	C_{IN}	f = 1MHz V_{DD} = 0V, unmeasured pins are connected to V_{SS}.	–	–	10	pF
Output Capacitance	C_{OUT}				–	
I/O Capacitance	C_{IO}				–	

Table 93. Oscillation Stabilization Time (T_A = –25°C to +85°C, V_{DD} = 1.8V to 3.6V)

Parameter	Conditions	Min.	Typ.	Max.	Unit
Main crystal	f_{OSC} > 1MHz	–	–	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:
1. f_{OSC} is the oscillator frequency.
2. 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.
3. Oscillation tolerance is ± 1%.

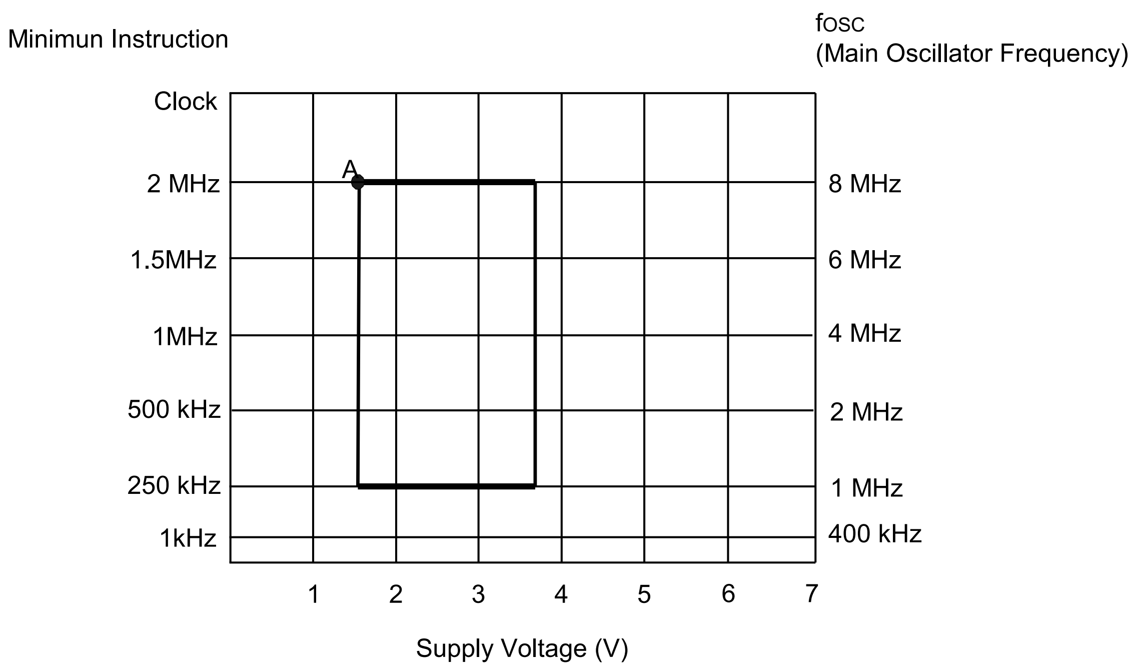


Figure 91. Operating Voltage Range of S3F80P5

Note: Minimum Instruction Clock = 1/4n x oscillator frequency (n = 1, 2, 8, or 16) A: 1.65 V, 8 MHz

Table 94. AC Electrical Characteristics for Internal Flash ROM (T_A = –25°C to +85°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 V	–	250	–	ns

Note:
1. The programming time is the period during which one byte (8-bit) is programmed.
2. The sector erasing time is the period during which all 128 bytes of one sector block are erased.
3. For the S3F80P5 MCU, chip erasure is available in Tool Program Mode only.

Table 94. AC Electrical Characteristics for Internal Flash ROM (T_A = –25°C to +85°C) (Continued)

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 byte (8-bit) is programmed.
2. The sector erasing time is the period during which all 128 bytes of one sector block are erased.
3. For the S3F80P5 MCU, chip erasure is available in Tool Program Mode only.

Table 95. 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 also recommended to add a 0.1 µF capacitor between nRESET pin and V_{SS} for better noise immunity.

Chapter 18. Mechanical Data

The S3F80P5 microcontroller is currently available in 24-pin SOP, 24-pin ELP and DIE packages shown in Figures 92 and 93, respectively.

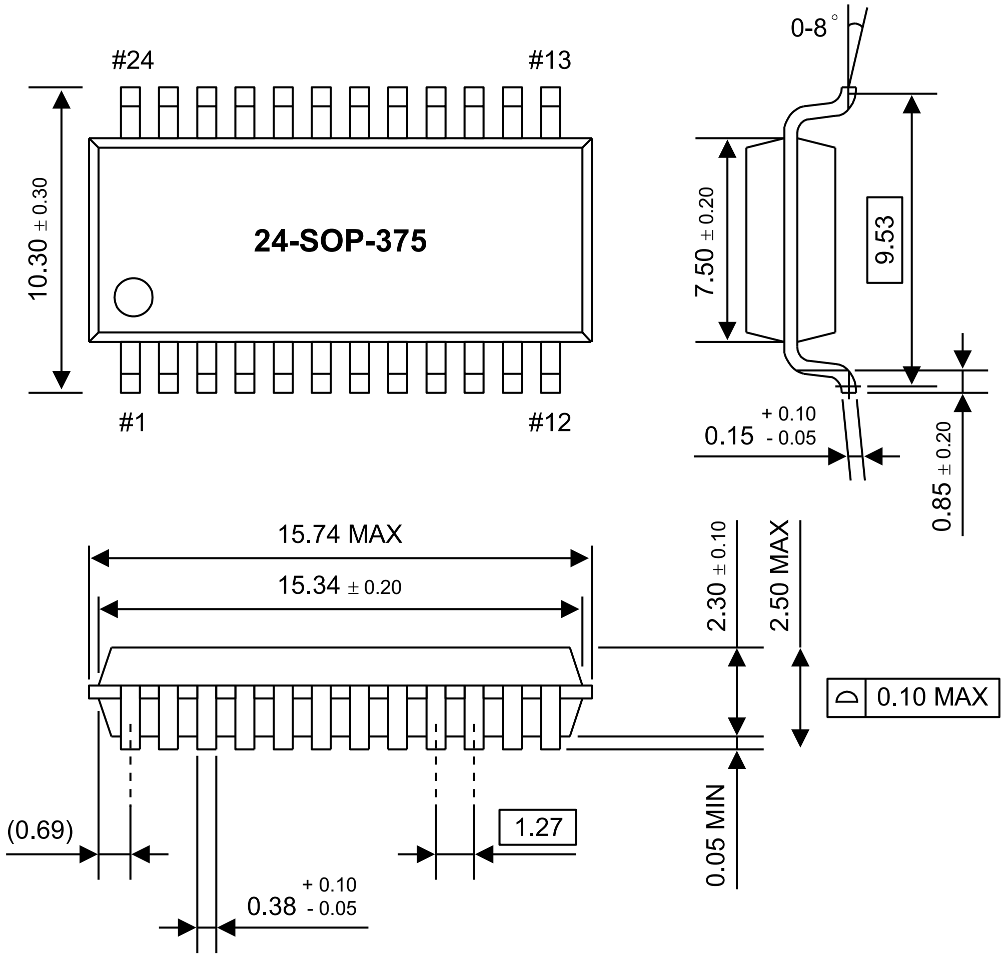
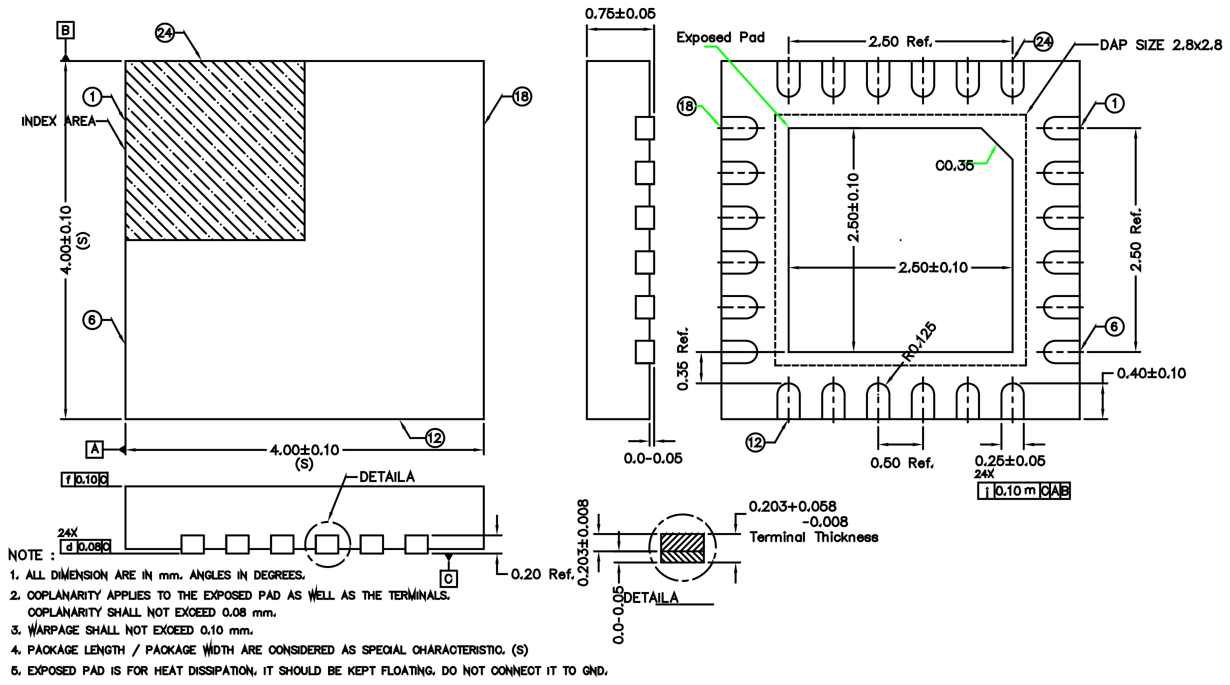


Figure 92. 24-Pin SOP Package Mechanical Data

Note: Dimensions are in millimeters.



NOTE :
1. ALL DIMENSION ARE IN mm. ANGLES IN DEGREES.
2. COPLANARITY APPLIES TO THE EXPOSED PAD AS WELL AS THE TERMINALS. COPLANARITY SHALL NOT EXCEED 0.08 mm.
3. WARPAGE SHALL NOT EXCEED 0.10 mm.
4. PACKAGE LENGTH / PACKAGE WIDTH ARE CONSIDERED AS SPECIAL CHARACTERISTIC. (S)
5. EXPOSED PAD IS FOR HEAT DISSIPATION. IT SHOULD BE KEPT FLOATING. DO NOT CONNECT IT TO GND.

Figure 93. 24-Pin ELP Package Mechanical Data

Note: Dimensions are in millimeters.

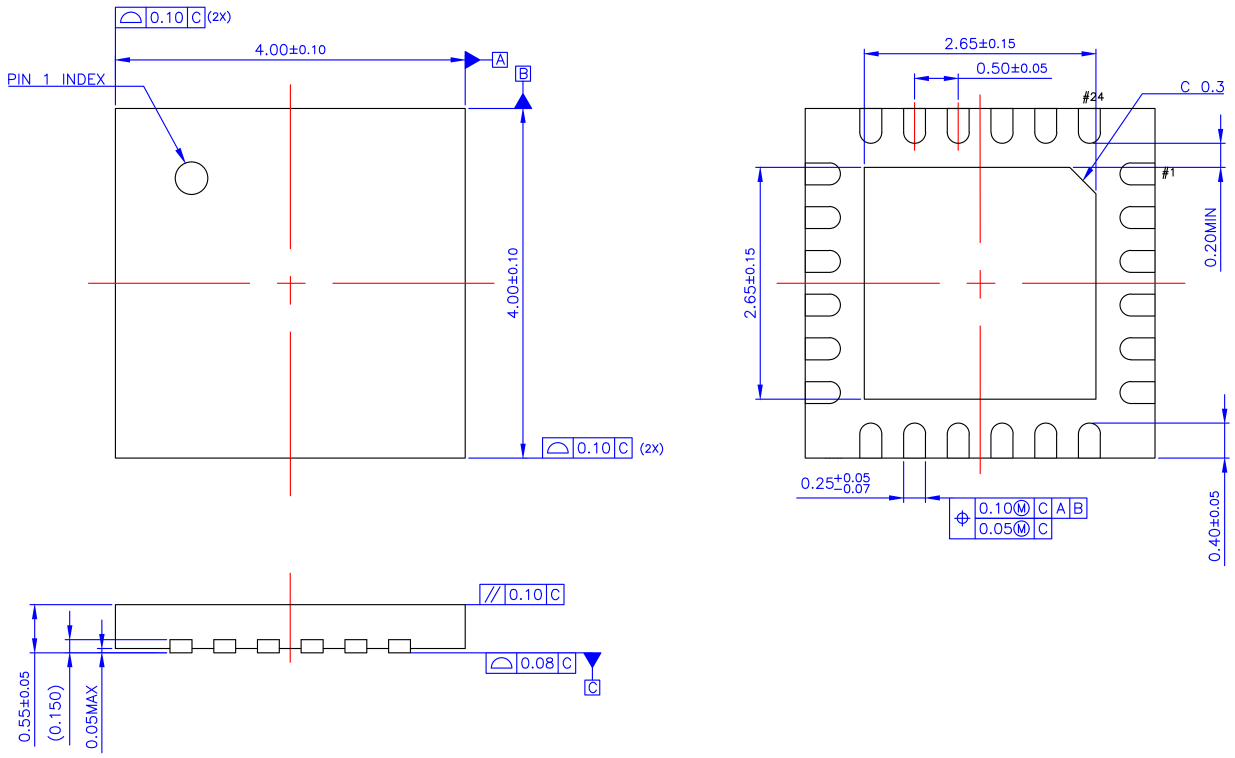


Figure 94. Thin 24-Pin ELP Package Mechanical Data

Note: The thin (0.55 mm) ELP package shown in Figure 94 is designated by a *K* in the 12th character of the part number listed in the Ordering Information table; see Table 100 on page 304.

Chapter 19. S3F80P5 Tool Programming

The S3F80P5 single-chip CMOS microcontroller is a Flash MCU which contains an on-chip Flash MCU ROM. The Flash ROM is accessed by serial data format.

Note: See the the Embedded Flash Memory Interface chapter on page 262 to learn more about User Program Mode.

19.1. Pin Assignments

Figures 95 and 96 illustrate the pin assignments for the S3F80P5 MCU's 24-pin SOP and 24-pin ELP packages respectively.

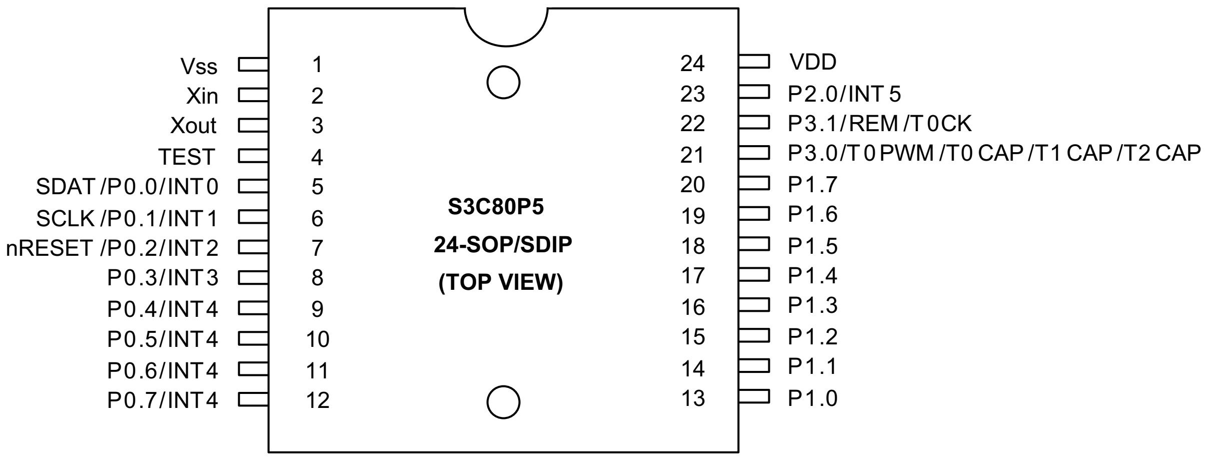


Figure 95. Pin Assignment Diagram (24-Pin SOP Package)

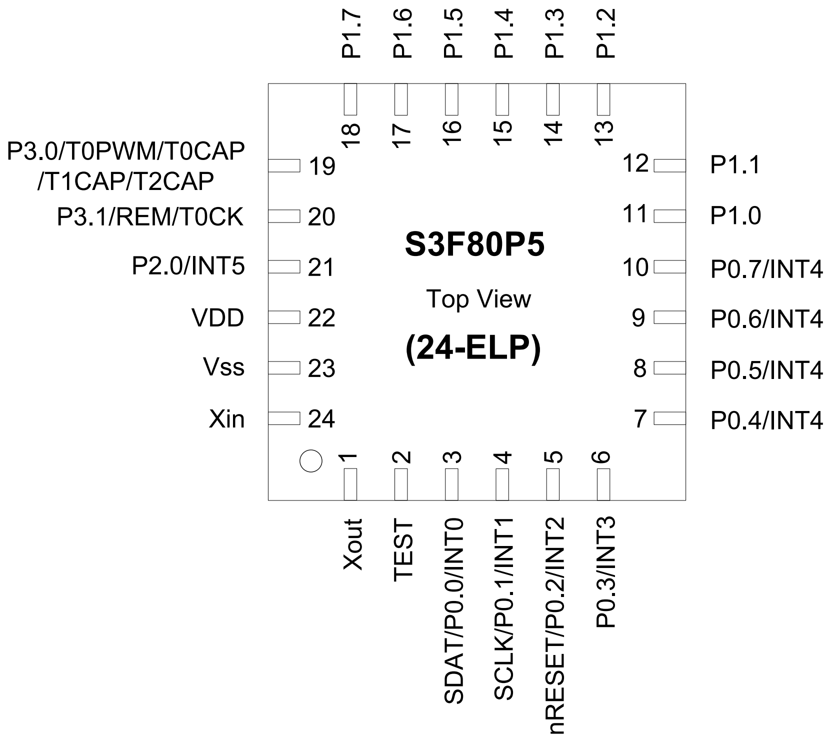


Figure 96. Pin Assignment Diagram (24-Pin ELP Package)

Table 96 lists the pins used to read, write, and erase Flash memory when operating in Tool Program Mode.

Table 96. Flash Operations in Tool Program Mode

Normal Chip	During Programming			
Pin Name	**Pin Name**	**Pin No.**	**I/O**	**Function**
P0.0	SDAT	5(3)	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	6(4)	I	Serial clock pin. Input only pin.
TEST	TEST	4(2)	I	Tool Mode selection when the TEST pin sets Logic value 1. If the user employs the Flash Writer Tool Mode (e.g., spw2+, etc.), the user should connect the TEST pin to V_{DD}. (The S3F80P5 MCU supplies high voltage 12.5V by internal high voltage generation circuit.)

Table 96. Flash Operations in Tool Program Mode (Continued)

Normal Chip Pin Name	During Programming			
	Pin Name	Pin No.	I/O	Function
nRESET	nRESET	7(5)	I	Chip Initialization.
V_{DD}, V_{SS}	V_{DD}, V_{SS}	24(22) 1(23)	–	Power supply pin for logic circuit. V_{DD} should be tied to +3.3V during programming.

19.1.1. Test Pin Voltage

The TEST pin on the socket board for the OTP/MTP writer must be connected to the V_{DD} (3.3 V). The TEST pin on socket board must not be connected to the VPP (12.5 V) which is generated from the OTP/MTP Writer. The specific socket board for the S3F80P5 MCU must be used when writing or erasing using OTP/MTP writer.

19.1.2. Operating Mode Characteristics

When 3.3 V is supplied to the TEST pin of the S3F80P5 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 97.

Table 97. 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.

Notes:
1. 0: Low level.
2. 1: High level.

Chapter 20. Development Tools

Zilog provides a powerful and easy-to-use development support system on a turnkey basis. This development support system is composed of a host system, debugging tools, and supporting software. Any standard computer running Windows 7 (32-/64-bit), Windows Vista (32-/64-bit), and Windows XP operating systems can be used as a host.

A sophisticated debugging tool is provided in both hardware and software formats: the powerful OPENice-i500/i2000 in-circuit emulator and the SK-1200 SmartKit. Zilog also offers supporting software that includes a debugger, an assembler, and a program for setting options.

20.1. Development System Configuration

Figure 97 shows the basic configuration of the development system.

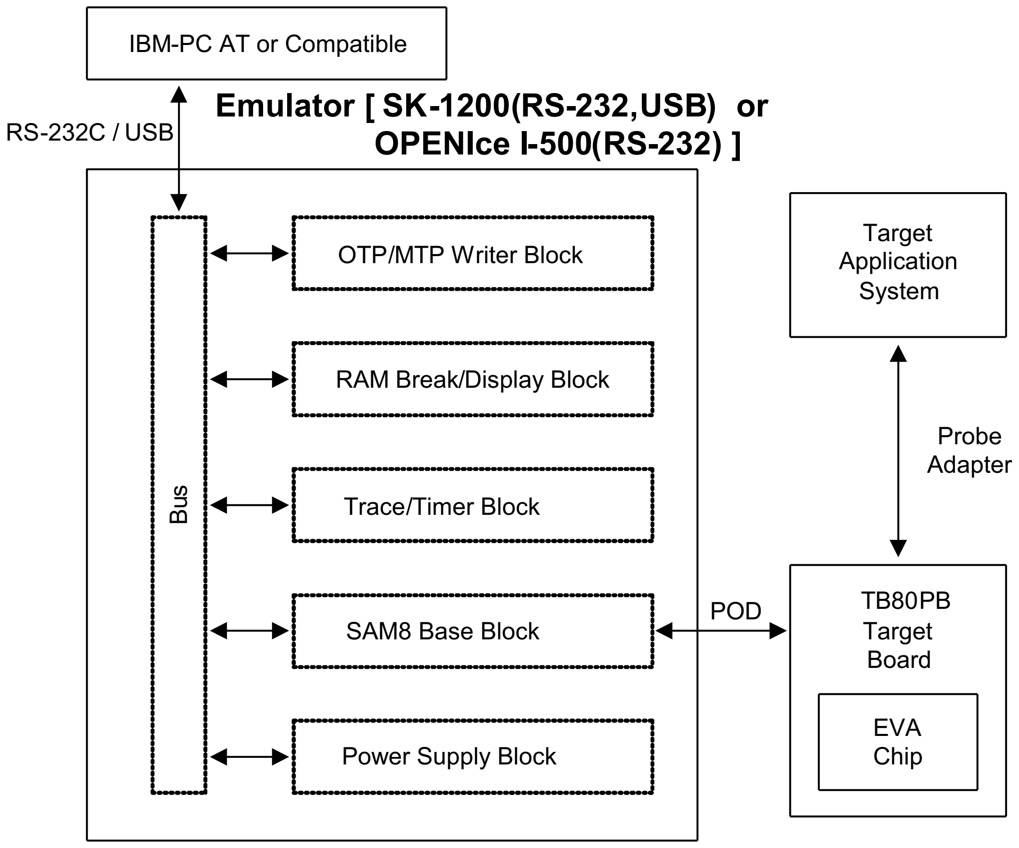


Figure 97. Development System Configuration

20.2. Target Board

The TB80PB target board can be used for development of the S3F80P5 MCU.

The TB80PB target board is operated as a target CPU with an Emulator (SK-1200, OPEN-Ice I-500) (see Figure 98)

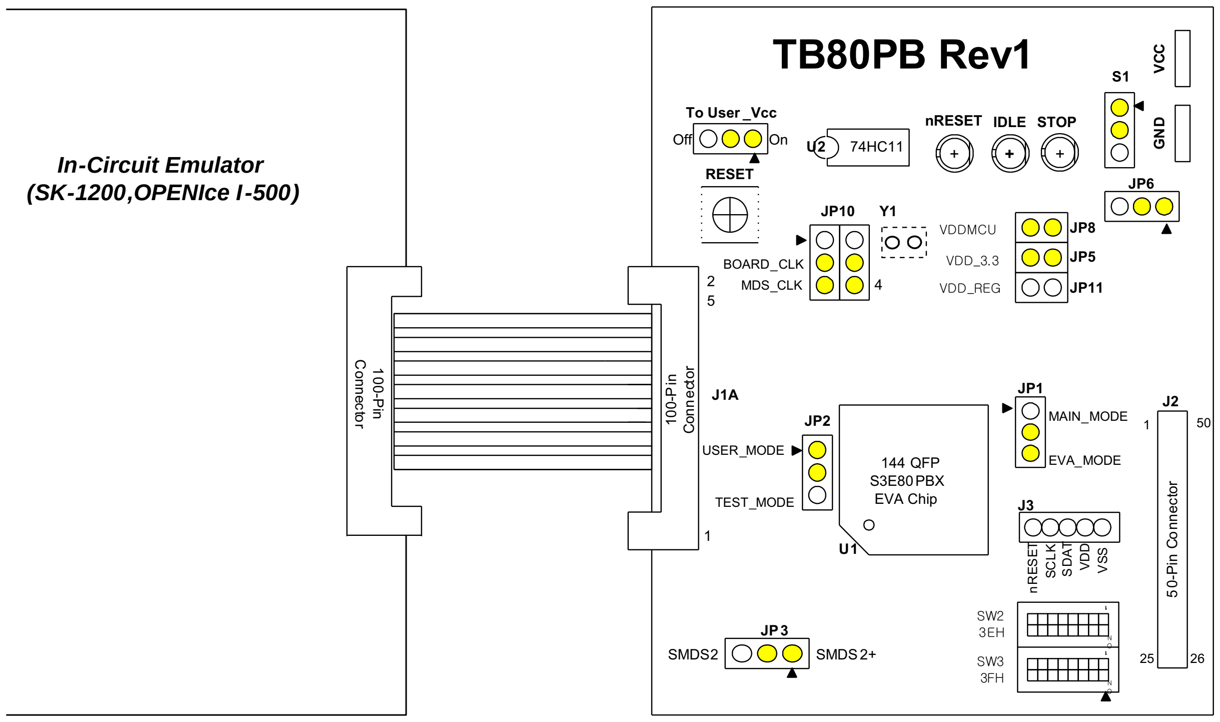


Figure 98. TB80PB Target Board Configuration

1. The TB80PB should be supplied with 3.3 V normally. The power supply from the Emulator should be set to 3.3 V for the target board operation. If the power supply from the Emulator is set to 5 V, activate the 3.3 V regulator on the TB80PB by setting the related jumpers (See Figure 98).
2. 2. The symbol ▶ marks start point of jumper signals.

Table 98 lists the power selection settings for the TB8S6B Target Board.

Table 98. TB8S6B Target Board 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	L: EVA-Mode	Join 2-3
JP2	Operation Mode	H: User-Mode	L: Test-Mode	Join 1-2
JP3	MDS version	SMDS2	SMDS2+, SK-1200, OPEN-Ice 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 peripheral power connection	Board peripheral power connection		Connect
JP6	When 5V is supplied to the target board, generate 3.3V using a regulator.	When using a 3.3V emulator, do not use a 3.3V regulator.	When using a 5V emulator, use a 3.3V regulator.	Join 2-3
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 the user prefers to use the external clock source as a crystal, the 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
JP11, 12	Not used for the TB80PB target board.		–	Not connected
SW1	Generation low active reset signal to S3F80PB EVA chip	Push switch		–
SW2	Smart option at address 3Eh	Dip switch for Smart Option. This 1byte is mapped address 3Eh for special function. Refer to Figure 10 on page 15.		
SW3	Smart option at address 3Fh	Dip switch for Smart Option. This 1byte is mapped address 3Fh for special function. Refer to Figure 10 on page 15.		
Y1	External clock source	Connecting points for external clock source.		

Table 98. TB8S6B Target Board Jumper Settings (Continued)

JP#	Description	1-2 Connection	2-3 Connection	Default Setting
J3	Header for Flash serial programming signals	To program an internal Flash, connect the signals with Flash writer tool.		J3
To User_ V_{CC}	Target system is supplied V_{DD}	Target Board is not supplied V_{DD} from user System.	Target Board is supplied V_{DD} from user System.	Join 2-3

The TB80PB Target Board features the LEDs listed in Table 99.

Table 99. 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.

Figure 99 shows the 50-pin connector pin assignment for the user system to program the S3F80P5 MCU.

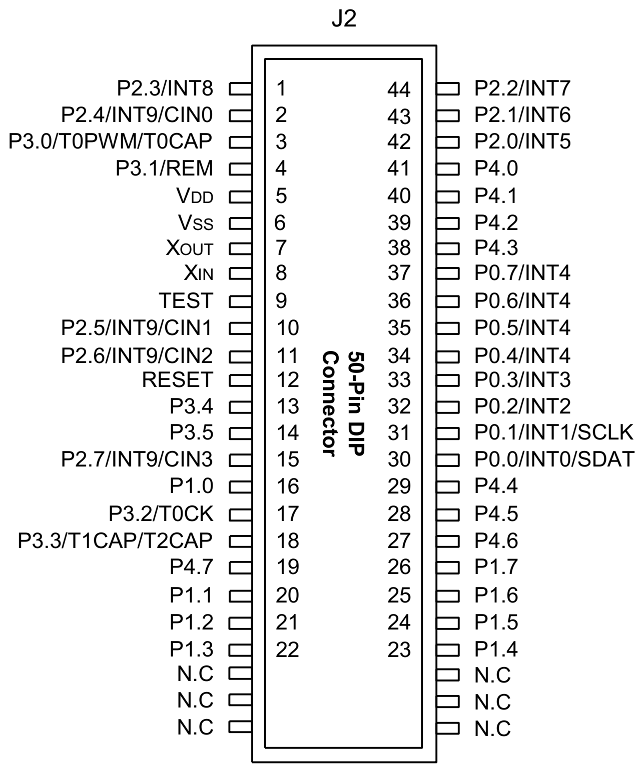


Figure 99. 50-Pin Connector Pin Assignment for User System

Note: N.C means No Connection.

20.3. Programming Socket Adapter

When you program S3F80P5 MCU’s Flash memory by using an emulator or OTP/MTP writer, a specific programming socket adapter for the S3F80P5 MCU is required.

20.4. Probe Adapter

S3F80P9 MCU Flash memory can be programmed by using an emulator or an OTP/MTP writer that requires the TB80PB Probe Adapter cable illustrated in Figure 100.

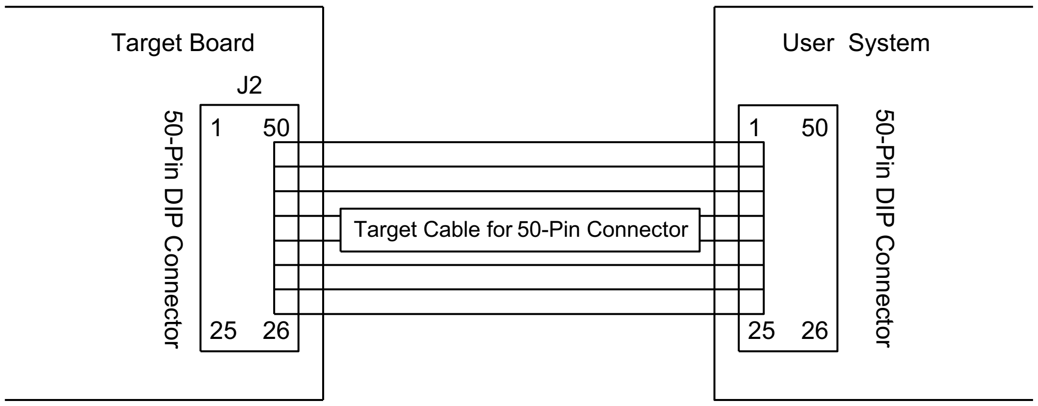


Figure 100. TB80PB Probe Adapter Cable

20.5. Third Parties for Development Tools

Zilog provides a complete line of development tools that support the S3 Family of Microcontrollers. With long experience in developing MCU systems, these third party firms are bonafide leaders in MCU development tool technology.

In-circuit emulators:

- OPENice-i500/2000
- SK-1200 SmartKit

OTP/MTP Programmers:

- GW-Uni2
- AS-Pro2
- Elnec programmers

To obtain the S3 Family development tools that will satisfy your S3F80P5 development objectives, contact your local Zilog Sales Office, or visit Zilog's Third Party Tools page to review our list of third party tool suppliers.

Chapter 21. Ordering Information

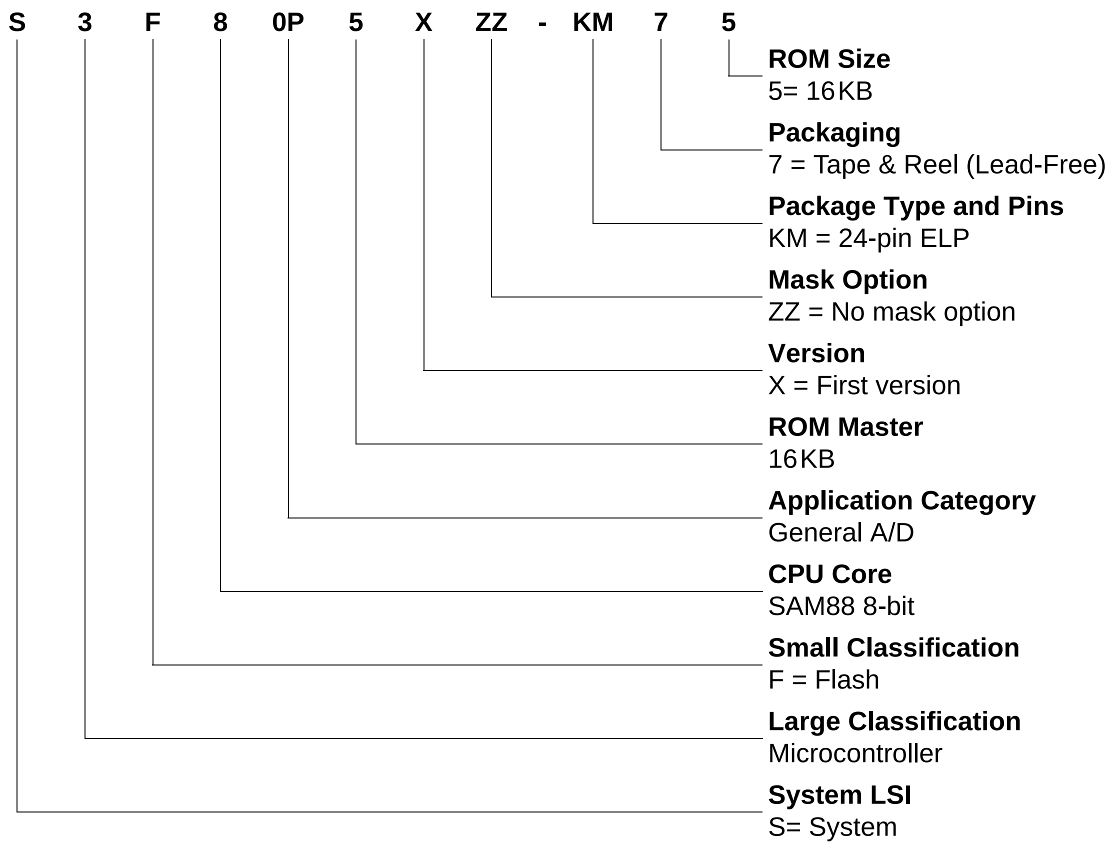
Table 100 identifies the basic features and package styles available for the S3F80P5 MCU.

Table 100. Ordering Information for the S3F80P5 MCU

Device	Flash Size	RAM Size	Interrupts	GPIO	Package
S3F80P5XZZ-SM95	18KB	1296B	16	19	24-pin SOP
S3F80P5XZZ-LM85	18KB	1296B	16	19	24-pin ELP
S3F80P5XZZ-KM75	18KB	1296B	16	19	Thin 24-pin ELP
S3F80P5XZZ-C0C5	18KB	1296B	16	19	Pellet (Die)

21.1. Part Number Suffix Designations

Zilog part numbers consist of a number of components. For example, part number S3F80P5XZZ-KM75 is an unmasked 8-bit MCU with 18KB of Flash memory in a 24-pin ELP package and built using lead-free solder.



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 facets 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.