S3 Family 8-Bit Microcontrollers

S3F94C8/S3F94C4

Product Specification

PS031503-0115

P R E L I M I N A R Y

S³ Microcontrollers

⚠ **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	03	Updated the Third Parties for Development Tools section.	17-9
May 2014	02	Corrected R value in Figure 8-2 and following note to 40KΩ; entire document updated to include front matter, PDF bookmarks for easier navigation.	8-3
Aug 2013	01	Original Zilog issue.	All

Chip Handling Guide

Precaution Against Electrostatic Discharge

When handling semiconductor devices, be sure that the environment is protected against static electricity.

1. Operators should wear anti-static clothing and use earth band.

2. All objects that come in direct contact with devices should be made of materials that do not produce static electricity that would cause damage.

3. Equipment and work table must be earthed.

4. Ionizer is recommended to remove electron charge.

Contamination
Be sure to use semiconductor products in the environment that may not be exposed to dust or dirt adhesion.

Temperature/Humidity
Semiconductor devices are sensitive to environment temperature and humidity. High temperature or humidity may deteriorate semiconductor device's characteristics. Therefore avoid storage or use in such conditions.

Mechanical Shock
Care should be exercised not to apply excessive mechanical shock or force on semiconductor device.

Chemical
Do not expose semiconductor device to chemical because reaction to chemical may cause deterioration of device characteristics.

Light Protection
In case of non-EMC (Epoxy Molding Compound) package, do not expose semiconductor IC to strong light. It may cause device's malfunction. (But, some special products which utilize the light or have security function are excepted from this guide)

Radioactive, Cosmic and X-Ray
Semiconductor devices can be influenced by radioactive, cosmic ray or X-ray. Radioactive, cosmic and X-ray may cause soft error during device operation. Therefore semiconductor devices must be shielded under environment that may be exposed to radioactive, cosmic ray or X-ray.

Electromagnetic Susceptibility (EMS)
Note that semiconductor device's characteristics may be affected by strong electromagnetic wave or magnetic field during operation under insufficient PCB circuit design for EMS.

Table of Contents

List of Figures

List of Tables

1 Product Overview

1.1 SAM88RCRI Microcontrollers

Zilog's SAM88RCRI series of 8-bit single-chip CMOS microcontrollers offers a fast and efficient CPU, a wide range of integrated peripherals, and various programmable ROM sizes. Important CPU features include:

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

A address/data bus architecture and a large number of bit-configurable I/O ports provide a flexible programming environment for applications with varied memory and I/O requirements. Timer/counters with selectable operating modes are included to support real-time operations.

1.2 S3F94C8/S3F94C4 Microcontroller

The S3F94C8/S3F94C4 single-chip 8-bit microcontroller is designed for useful A/D converter application field. The S3F94C8/S3F94C4 single-chip CMOS microcontroller is fabricated using a highly advanced CMOS process and is based on Zilog's powerful SAM88RCRI CPU architecture. Stop and idle (power-down) modes were implemented to reduce power consumption.

- The S3F94C8 is a microcontroller with a **8**KB multi-time-programmable Full Flash ROM embedded.
- The S3F94C4 is a microcontroller with a **4**KB multi-time-programmable Full Flash ROM embedded.

The S3C94C8/S3F94C4 is a versatile general-purpose microcontrollers that is ideal for use in a wide range of electronics applications requiring simple timer/counter, PWM. In addition, the S3F94C8/S3F94C4 advanced CMOS technology provides for low power consumption and wide operating voltage range.

Using the SAM88RCRI design approach, the following peripherals were integrated with the powerful core:

- Three configurable I/O ports (18 pins)
- Four interrupt sources with One vectors and one interrupt level
- 0ne 8-bit timer/counter with time interval modes.
- Analog to digital converter with nine input channels (MAX) and 10-bit resolution
- One PWM output with three optional mode: 8-bit (6+2); 12-bit (6+6); 14-bit (8+6);

The S3F94C8/S3F94C4 microcontroller is ideal for use in a wide range of electronic applications requiring simple timer/counter, PWM, ADC. They are currently available in 20 DIP Package, 20/16-pin SOP Package, 20 SSOP Package and 16 TSSOP Package.

1.3 Features

1.3.1 CPU

SAM88RCRI CPU core

1.3.2 Memory

Internal multi-time program Full-Flash memory:
- 8K×8 bits program memory(S3F94C8)
  4K×8 bits program memory(S3F94C4)
    - √ Sector size: 128 Bytes
    - √ User programmable by 'LDC' instruction
    - √ Sector erase available
    - √ Fast programming time
    - √ External serial programming support
    - √ Endurance: 10,000 erase/program cycles
    - √ 10 Years data retention
- 208-byte general-purpose register area

1.3.3 Instruction Set

- 41 instructions
- Idle and Stop instructions added for power-down modes

1.3.4 Instruction Execution Time

400 ns at 10 MHz f_{OSC} (minimum)

1.3.5 Interrupts

1 interrupt levels and 4 interrupt sources (*2 external interrupts and 2 internal interrupts*)

1.3.6 General I/O

- Three I/O ports (Max 18 pins)
- Bit-programmable ports

1.3.7 1-Channel High-Speed PWM with Three Selectable Resolutions

- 8-bit PWM: 6-bit base + 2-bit extension
- 12-bit PWM: 6-bit base + 6-bit extension
- 14-bit PWM: 8-bit base + 6-bit extension

1.3.8 Timer/Counters

- One 8-bit basic timer for watchdog function

- One 8-bit timer/counter with time interval modes

1.3.9 A/D Converter

Nine analog input pins (MAX)
- 10-bit conversion resolution

1.3.10 Oscillation Frequency

- 0.4 MHz to 10 MHz external crystal oscillator
- Typical 4MHz external RC oscillator
- Internal RC: 3.2MHz typ., 0.5MHz typ., VDD = 5V

1.3.11 Built-in RESET Circuit (LVR)

Low-Voltage check to make system reset
- V_{LVR} = 1.9/2.3/3.0/3.6/3.9 V (by Smart Option)

1.3.12 Smart Option

- LVR enable/disable
- Oscillator selection

1.3.13 Operating Temperature Range

- –40°C to +85°C

1.3.14 Operating Voltage Range

- 1.8 V to 5.5 V @ 0.4 - 4M Hz(LVR disable)
- LVR to 5.5V @ 0.4 - 4M Hz(LVR enable)
- 2.7 V to 5.5V @ 0.4 -10M Hz

1.3.15 Package Types

- 20-DIP-300A
- 20-SOP-375
- 20-SSOP-225
- 16-SOP-225
- 16-TSSOP-0044

Table 1-1. S3F94C8/S3F94C4 Pin Descriptions

Device	Operating Temperature Range	Internal RC Temperature Range	Internal RC Tolerance
S3F94C8XZZ/S3F94C4XZZ	-40°C to $+85^{\circ}$C	-40°C to $+85^{\circ}$C	1% @ 5V, 25°C

1.4 Block Diagram
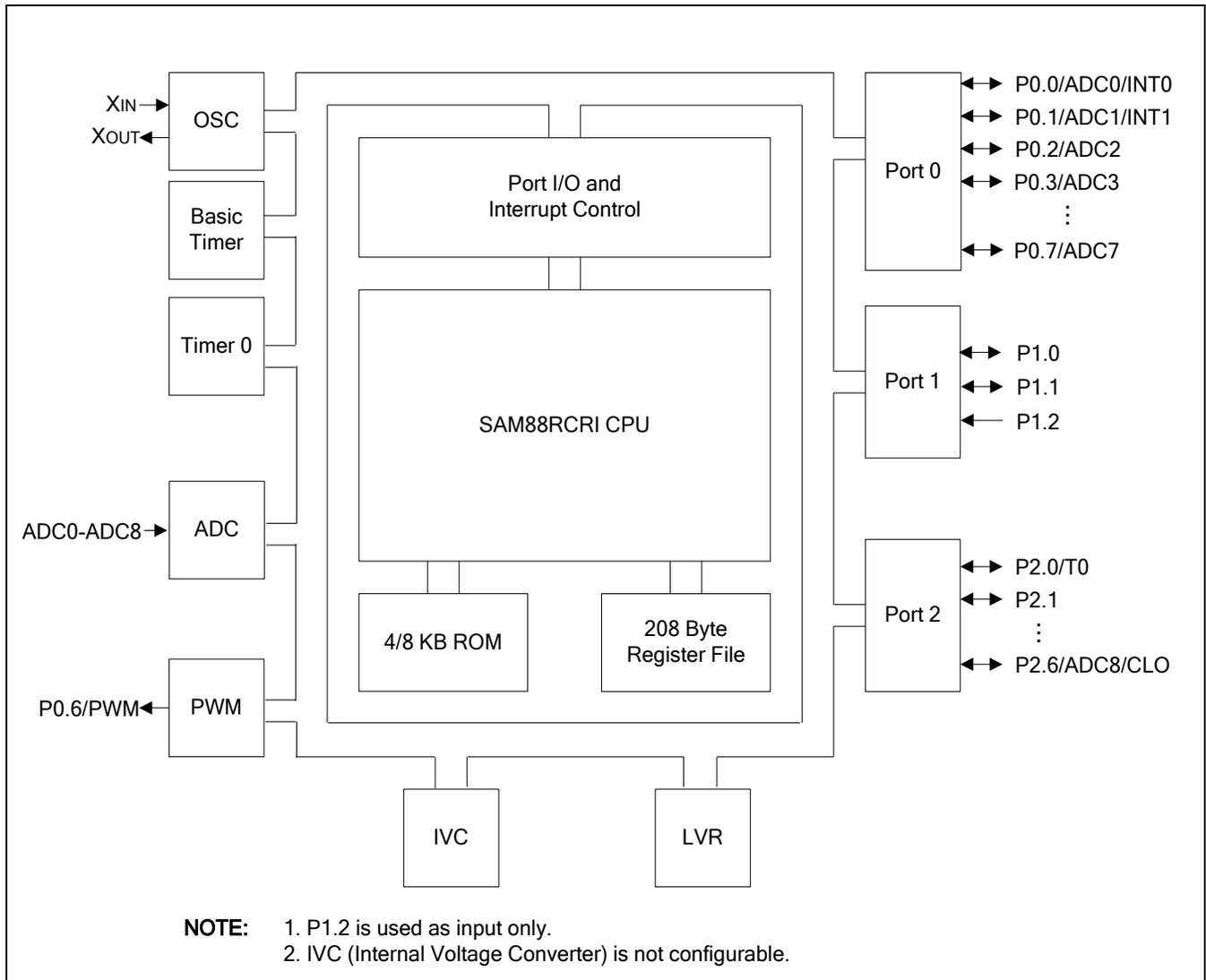
Figure 1-1. Block Diagram

1.5 Pin Assignments

Vss	1	20	VDD
XIN/P1.0	2	19	P0.0/ADC0/INT0 (SCLK)
XOUT/P1.1	3	18	P0.1/ADC1/INT1 (SDAT)
(VPP) nRESET/P1.2	4	17	P0.2/ADC2
T0/P2.0	5	16	P0.3/ADC3
P2.1	6	15	P0.4/ADC4
P2.2	7	14	P0.5/ADC5
P2.3	8	13	P0.6/ADC6/PWM
P2.4	9	12	P0.7/ADC7
P2.5	10	11	P2.6/ADC8/CLO

S3F94C8/F94C4

(20-DIP-300A/
20-SOP-375 /
20-SSOP-225)
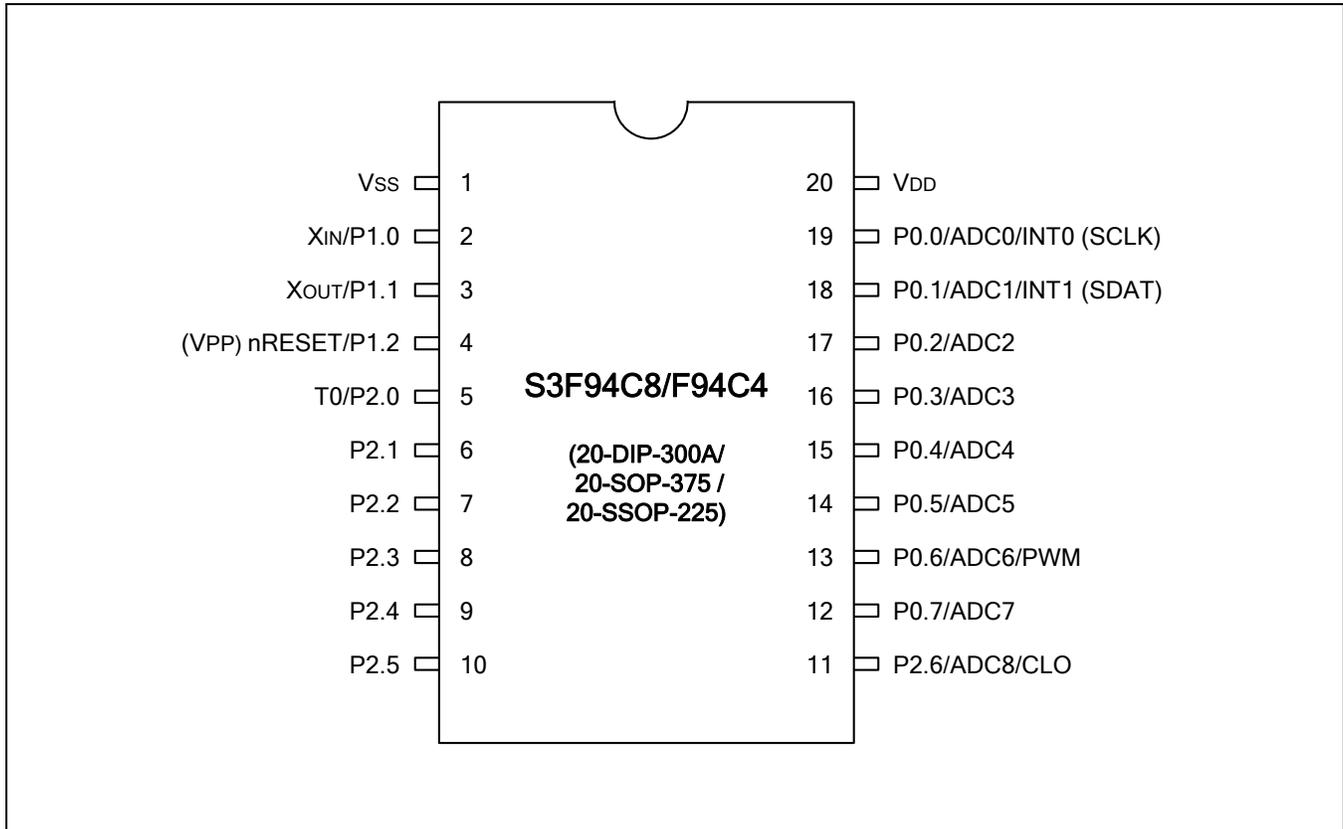
Figure 1-2. Pin Assignment Diagram, 20-Pin DIP/SOP/SSOP Packages
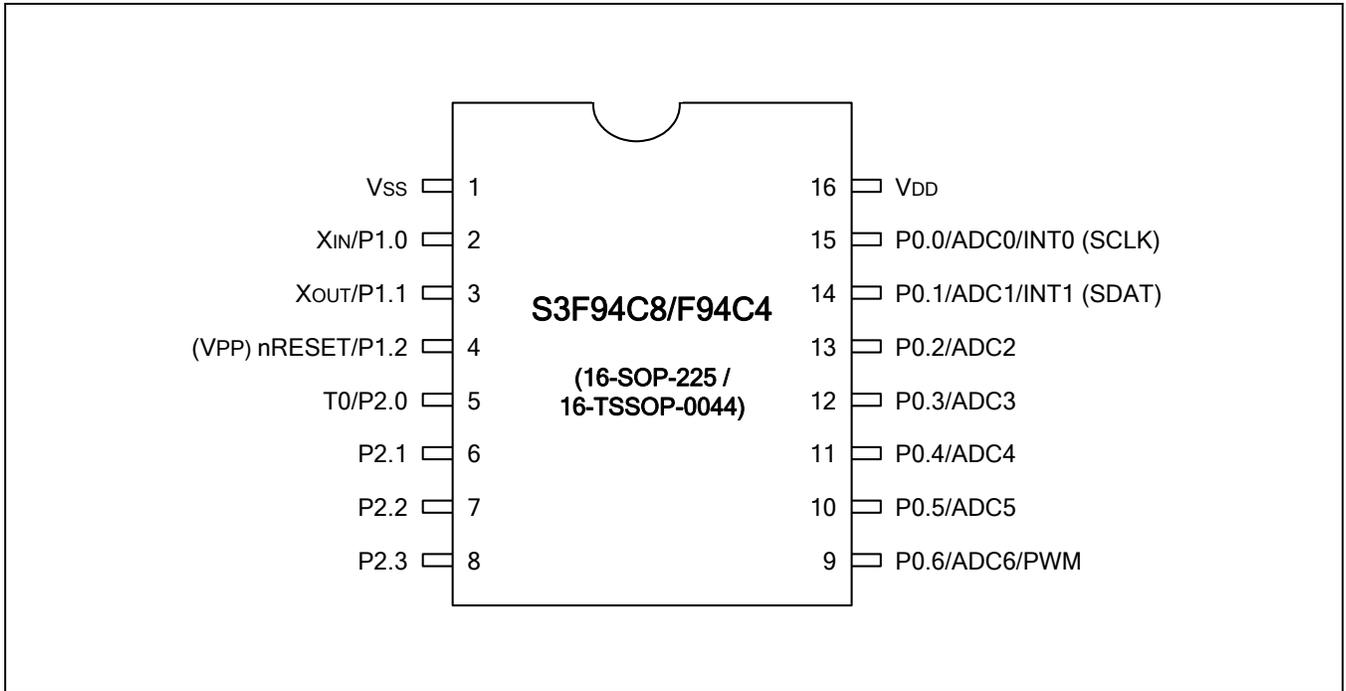
Figure 1-3. Pin Assignment Diagram, 16-Pin SOP/TSSOP Packages

1.6 Pin Descriptions

Table 1-2. S3F94C8/S3F94C4 Pin Descriptions

Pin Name	Input/ Output	Pin Description	Pin Type	Share Pins
P0.0–P0.7	I/O	Bit-programmable I/O port for Schmitt trigger input or push-pull output. Pull-up resistors are assignable by software. Port 0 pins can also be used as A/D converter input, PWM output or external interrupt input.	E-1	ADC0–ADC7 INT0/INT1/ PWM
P1.0–P1.1	I/O	Bit-programmable I/O port for Schmitt trigger input or push-pull, open-drain output. Pull-up resistors or pull-down resistors are assignable by software.	E-2	X_{IN}, X_{OUT}
P1.2	I	Schmitt trigger input port	B[1]	RESET
P2.0–P2.6	I/O	Bit-programmable I/O port for Schmitt trigger input or push-pull, open-drain output. Pull-up resistors are assignable by software.	E	– ADC8/CLO T0
X_{IN}, X_{OUT}	–	Crystal/Ceramic, or RC oscillator signal for system clock.		P1.0–P1.1
nRESET	I	Internal LVR or external RESET	B	P1.2
V_{DD}, V_{SS}	–	Voltage input pin and ground		–
CLO	O	System clock output port	E	P2.6
INT0–INT1	I	External interrupt input port	E-1	P0.0, P0.1
PWM	O	14-Bit high speed PWM output	E-1	P0.6
T0	O	Timer 0 match output	E-1	P2.0
ADC0–ADC8	I	A/D converter input	E-1 E	P0.0–P0.7 P2.6

Table 1-3. Descriptions of Pins Used to Read/Write the Flash ROM

Main Chip Pin Name	During Programming			
	Pin Name	Pin No.	I/O	Function
P0.1	SDAT	18 (20-pin) 14 (16-pin)	I/O	Serial data pin (output when reading, Input when writing) Input and push-pull output port can be assigned.
P0.0	SCLK	19 (20-pin) 15 (16-pin)	I	Serial clock pin (input only pin).
RESET/P1.2	V_{PP}	4	I	Power supply pin for Tool mode entering (indicates that MTP enters into the Tool mode). When 11 V is applied, MTP is in Tool mode.
V_{DD}/V_{SS}	V_{DD}/V_{SS}	20 (20-pin), 16 (16-pin) 1 (20-pin), 1 (16-pin)	I	Logic power supply pin.

1.7 Pin Circuits
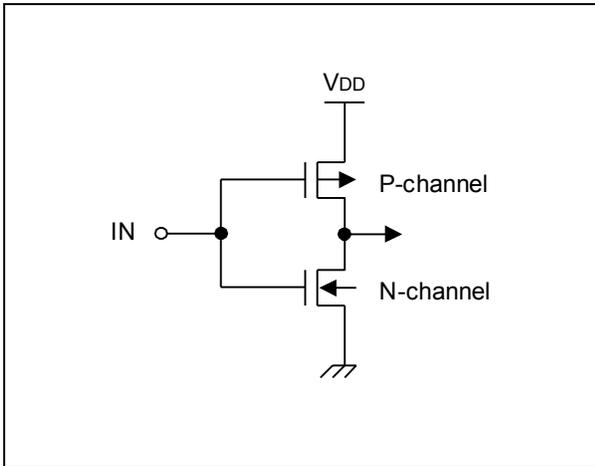
Figure 1-4. Pin Circuit Type A
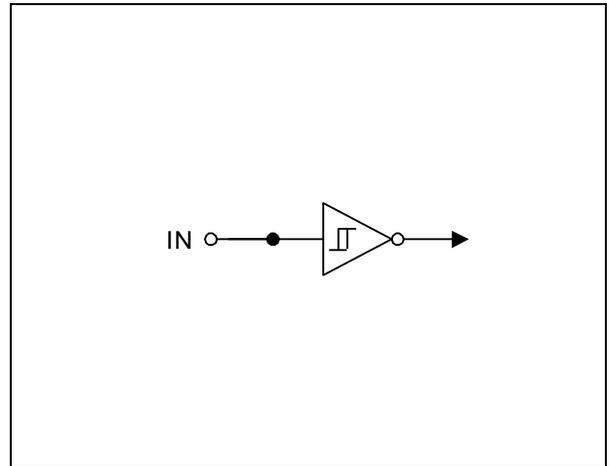
Figure 1-5. Pin Circuit Type B (P1.2)
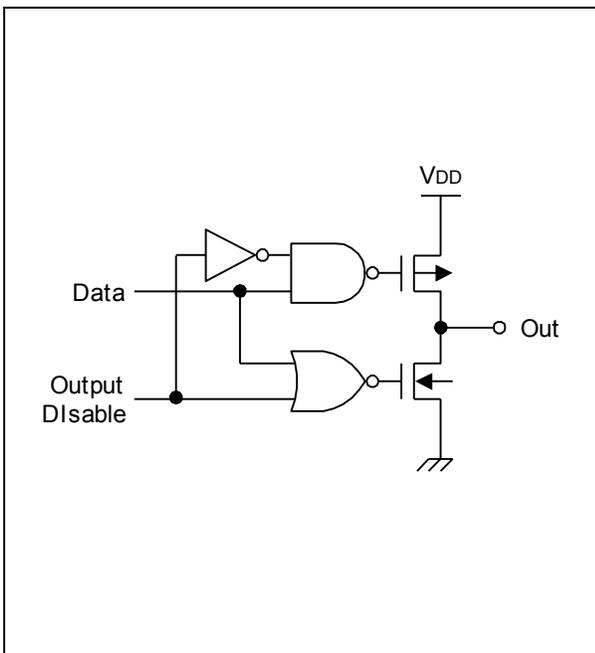
Figure 1-6. Pin Circuit Type C
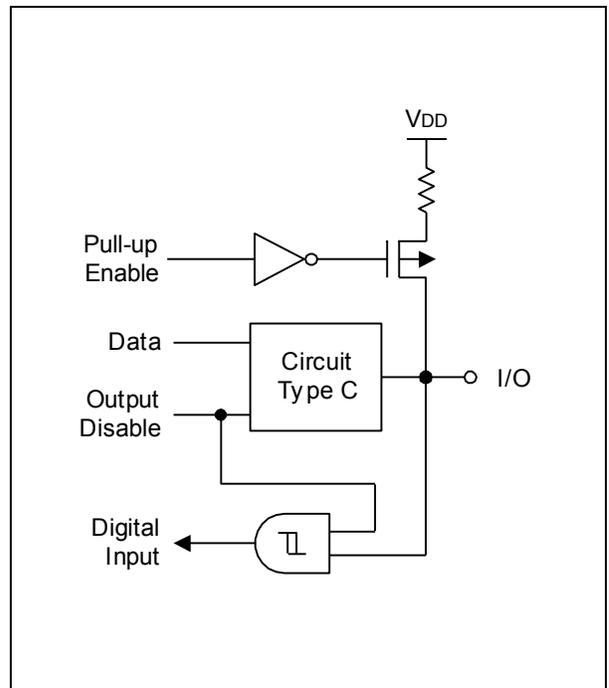
Figure 1-7. Pin Circuit Type D
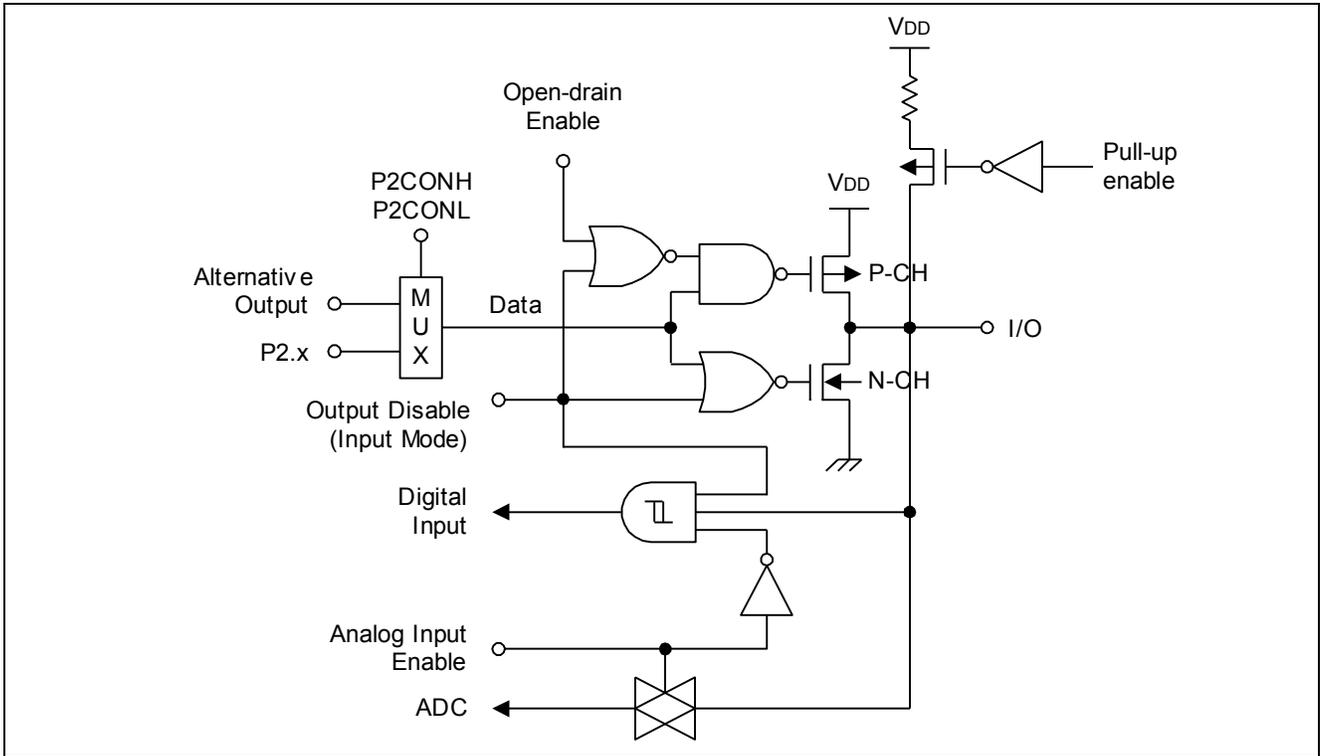
Figure 1-8. Pin Circuit Type E (Port 2)
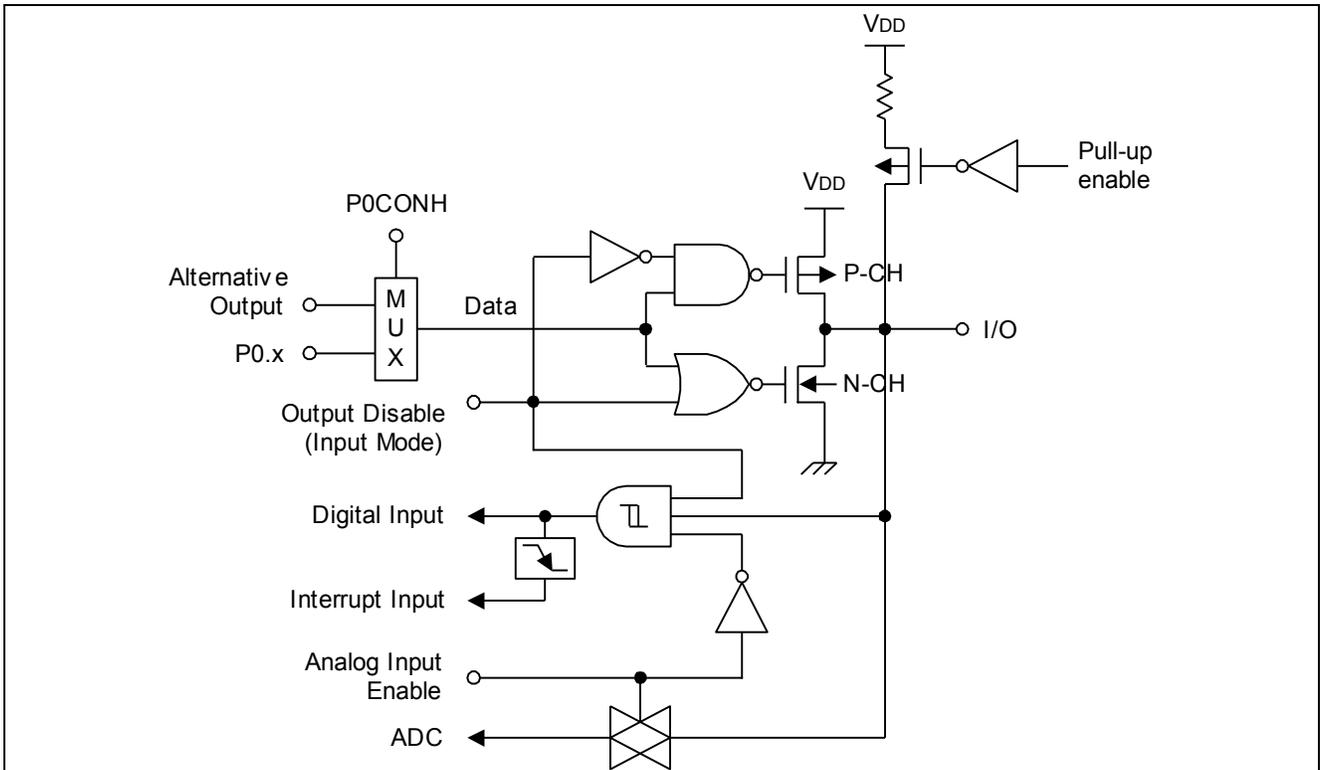
Figure 1-9. Pin Circuit Type E-1 (Port 0)
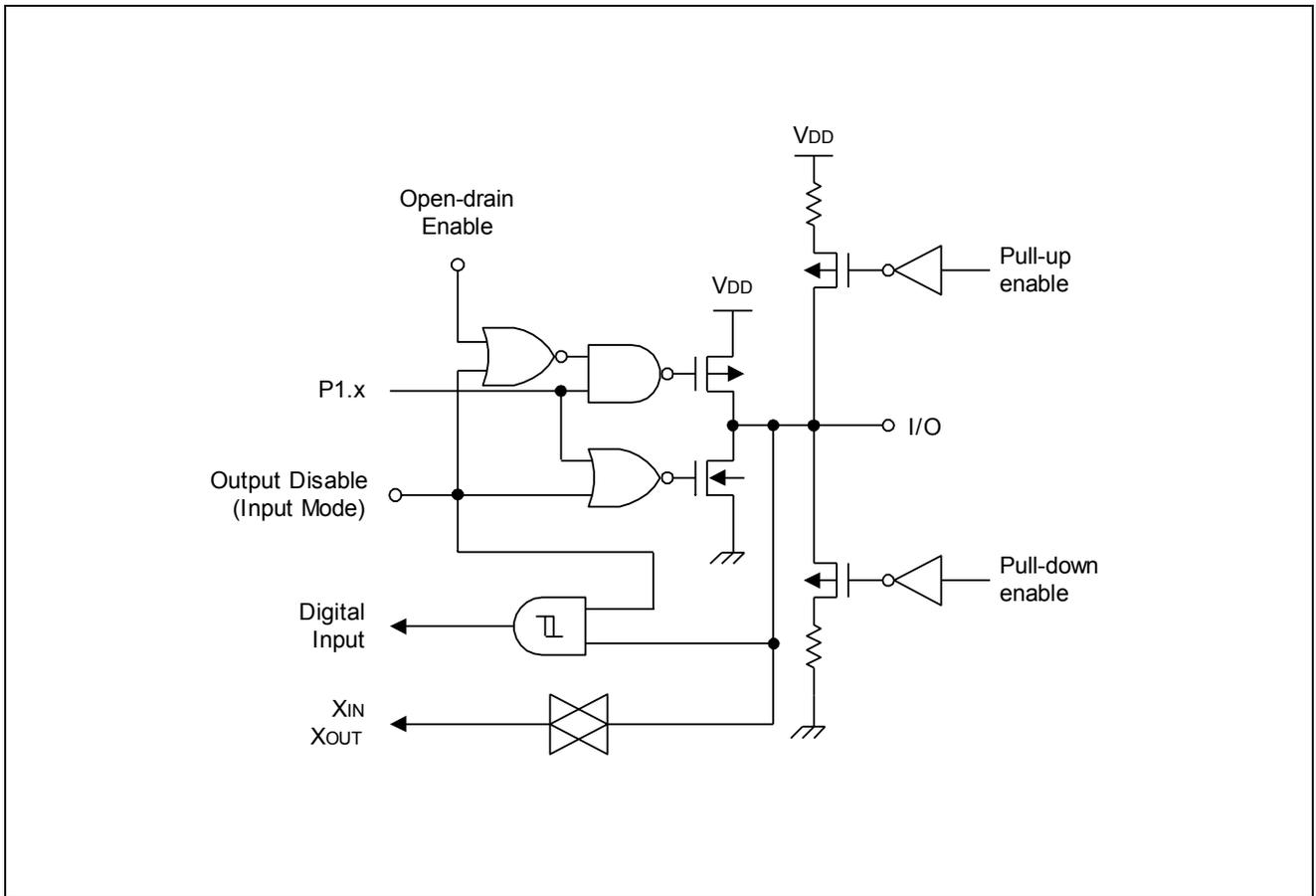
Figure 1-10. Pin Circuit Type E-2 (P1.0-P1.1)

2 Address Spaces

2.1 Overview

The S3F94C8/S3F94C4 microcontroller has two kinds of address space:

Internal full Flash program memory (ROM)

Internal register file

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

The S3F94C8/S3F94C4 have 8KB and 4KB of multi-time-programmable full Flash program memory: which is configured as the Internal ROM mode, all of the 4K/8K internal program memory is used.

The S3F94C8/S3F94C4 microcontroller features 208 general-purpose registers in its internal register file. 32 bytes in the register file are mapped for system and peripheral control functions.

2.2 Program Memory (ROM)

2.2.1 Normal Operating Mode

The S3F94C8/S3F94C4 has 8KB and 4KB of internal multi-time-programmable full Flash program memory. The program memory address range is therefore 0H–1FFFH and 0H–0FFFH.

The first 2 bytes of the ROM (0000H–0001H) are interrupt vector address.

Unused locations (0002H–00FFH except 3CH, 3DH, 3EH, and 3FH) can be used as normal program memory. 3CH, 3DH, 3EH, 3FH is used as Smart Option ROM cell.

The program Reset address in the ROM is 0100H.

```
(Decimal)                                    (HEX)
  8.191 ┌──────────────────────────────┐    1FFFH
        │                              │    (S3F94C8)
        │                              │
        │        8-Kbyte               │
        │        Program               │
        │        Memory                │
        │        (Flash)               │
        │                              │
  4.095 ├ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┤    0FFFH
        │                              │    (S3F94C4)
        │        4-Kbyte               │
        │        Program               │
        │        Memory                │
        │        (Flash)               │
        │                              │
    256 ├──────────────────────────────┤    0100H
        │       Program Start          │
     64 ├──────────────────────────────┤    0040H
        │   Smart option ROM cell      │
     60 ├──────────────────────────────┤    003CH
      2 ├──────────────────────────────┤    0002H
        │       Interrupt Vector       │    0001H
      1 │                              │
      0 └──────────────────────────────┘    0000H
```
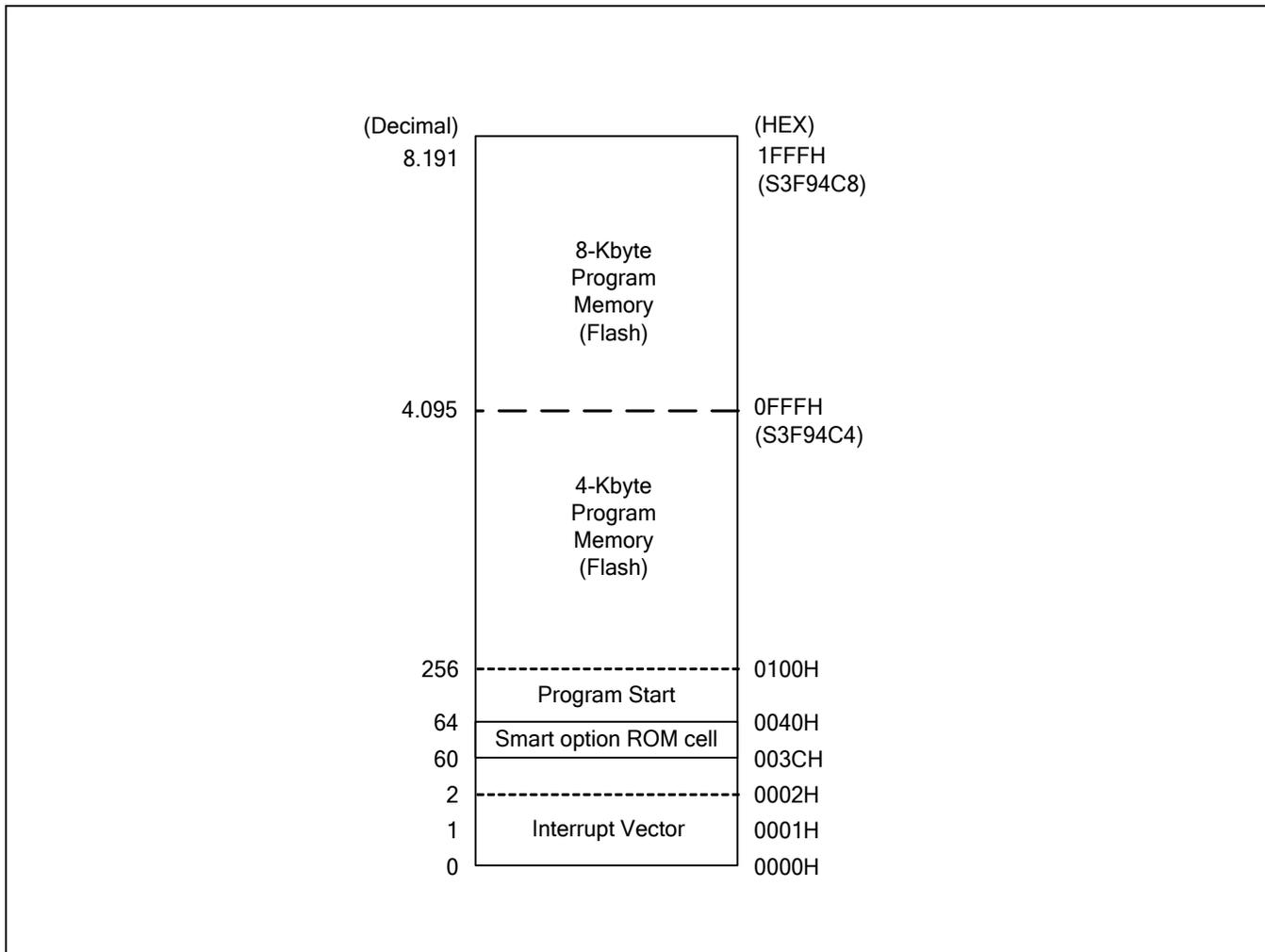
Figure 2-1. Program Memory Address Space

2.3 Smart Option

Smart Option is the ROM option for starting condition of the chip. The ROM addresses used by Smart Option are from 003CH to 003FH. The S3F94C8/S3F94C4 only use 003EH, 003FH. Not used ROM address 003CH, 003DH should be initialized to be initialized to 00H. The default values of ROM 003EH, 003FH are FFH (LVR enable, internal RC oscillator).

ROM Address: 003CH

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

Must be initialized to 00H.

ROM Address: 003DH

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

Must be initialized to 00H.

ROM Address: 003EH

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

LVR enable/disable bit:
0 = Disable
1 = Enable

LVR level selection bits:
10100 = 1.9 V
11001 = 2.3 V
10010 = 3.0 V
00111 = 3.6 V
01100 = 3.9 V

Not used

ROM Address: 003FH

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

Not used.

Oscillator selection bits:
00 = External crystal/ceramic oscillator
01 = External RC
10 = Internal RC (0.5 MHz in V_{DD} = 5 V)
11 = Internal RC (3.2 MHz in V_{DD} = 5 V)

NOTES:
1. When you use external oscillator, P1.0, P1.1 must be set to output port to prevent current consumption.
2. The value of unused bits of 3EH, 3FH is don't care.
3. When LVR is enabled: P1.2/nRESET is used as input port; and LVR level must be set to appropriate value, not default value;

Figure 2-2. Smart Option

☞ PROGRAMMING TIP — Smart Option Setting

```
;           << Interrupt Vector Address >>

            ORG         0000H
            Vector      00H, INT_94C8        ; S3F94C8/S3F94C4 has only one interrupt vector


;           << Smart Option Setting >>

            ORG         003CH
            DB          00H                  ; 003CH, must be initialized to 0.
            DB          00H                  ; 003DH, must be initialized to 0.
            DB          0E4H                 ; 003EH, enable LVR (2.3 V)
            DB          03H                  ; 003FH, Internal RC (3.2 MHz in V_DD = 5 V)


;           << Reset >>

            ORG         0100H
            RESET:      DI
                        •
                        •
                        •
```

2.4 Register Architecture

The upper 64 bytes of the S3F94C8/S3F94C4's internal register file are addressed as working registers, system control registers and peripheral control registers. The lower 192 bytes of internal register file (00H–BFH) is called the *general purpose register space*.

240 registers in this space can be accessed; 208 are available for general-purpose use.

In case of S3F94C8/S3F94C4 the total number of addressable 8-bit registers is 240. Of these 240 registers, 32 bytes are for CPU and system control registers and peripheral control and data registers, **16** bytes are used as shared working registers, and 192 registers are for general-purpose use.

For many SAM88RCRI microcontrollers, the addressable area of the internal register file is further expanded by additional register pages at the general purpose register space (00H–BFH: page0). This register file expansion is not implemented in the S3F94C8/S3F94C4, however.

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

Table 2-1. Register Type Summary

Register Type	Number of Bytes
CPU and system control registers	11
Peripheral, I/O, and clock control and data registers	21
General-purpose registers (including the 16-bit common working register area)	208
Total Addressable Bytes	240

FFH

Peripheral Control
Registers

64 Bytes of   E0H
Common Area   DFH

System Control
Registers

D0H
CFH

Working Registers

C0H
BFH

192 Bytes

General Purpose
Register File
and Stack Area

00H
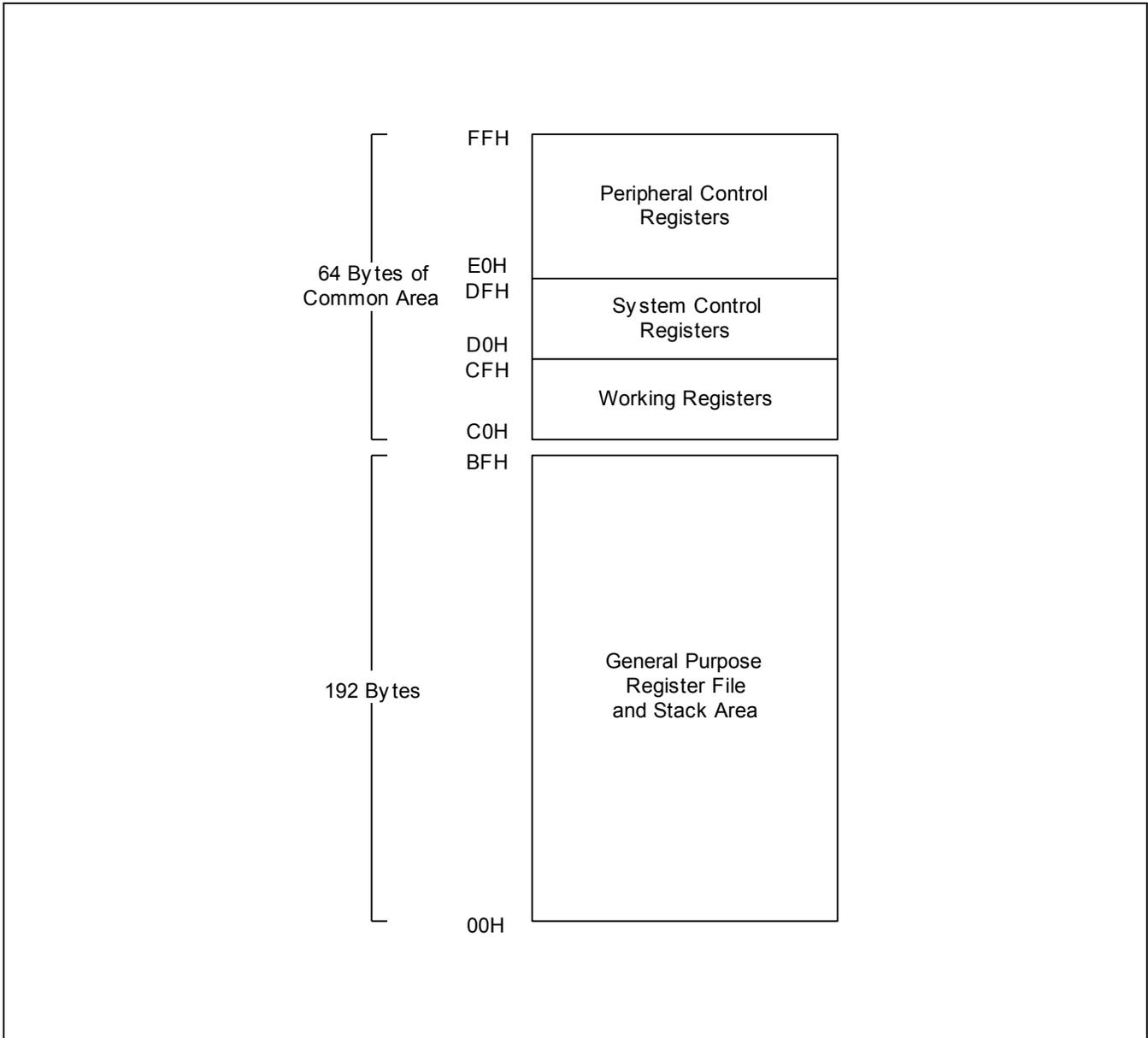
Figure 2-3. Internal Register File Organization

2.5 Common Working Register Area (C0H–CFH)

The SAM88RCRI register architecture provides an efficient method of working register addressing that takes full advantage of shorter instruction formats to reduce execution time.

This16-byte address range is called common area. That is, locations in this area can be used as working registers by operations that address any location on any page in the register file. Typically, these working registers serve as temporary buffers for data operations between different pages. However, because the S3F94C8/S3F94C4 uses only page 0, you can use the common area for any internal data operation.

The working register addressing mode and indirect register addressing mode can be used to access this area.

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

MSB	LSB	n = Even address
Rn	Rn+1	
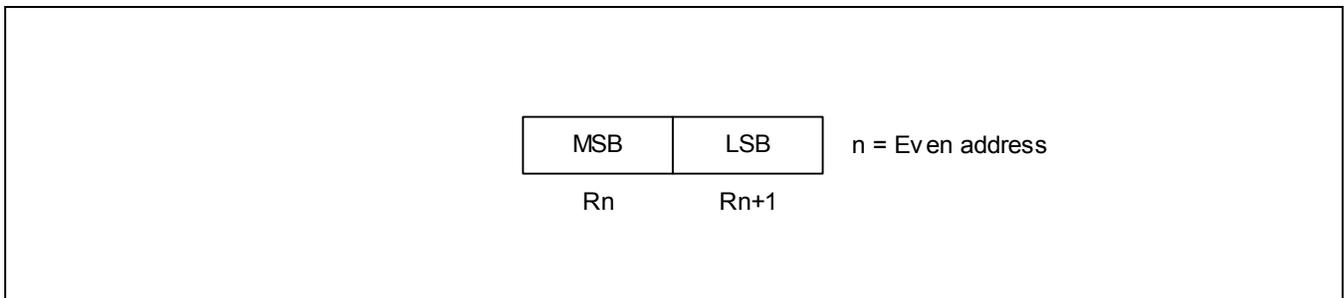
Figure 2-4. 16-Bit Register Pairs

☞ **PROGRAMMING TIP — Addressing the Common Working Register Area**

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

```
Examples:  1. LD     0C2H, 40H           ; Invalid addressing mode!
              Use working register addressing instead:
              LD     R2, 40H             ; R2 (C2H) □  the value in location 40H

           2. ADD    0C3H, #45H          ; Invalid addressing mode!
              Use working register addressing instead:
              ADD    R3, #45H            ; R3 (C3H) □  R3 + 45H
```

2.6 System Stack

S3F9-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 S3F94C8/S3F94C4 architecture supports stack operations in the internal register file.

2.6.1 Stack Operations

Return addresses for procedure calls and interrupts and data are stored on the stack. The contents of the PC are saved to stack by a CALL instruction and restored by the RET instruction. When an interrupt occurs, the contents of the PC and the FLAGS register are pushed to the stack. The IRET instruction then pops these values back to their original locations. The stack address is always decremented *before* a push operation and incremented *after* a pop operation. The stack pointer (SP) always points to the stack frame stored on the top of the stack, as shown in Figure 2-5.
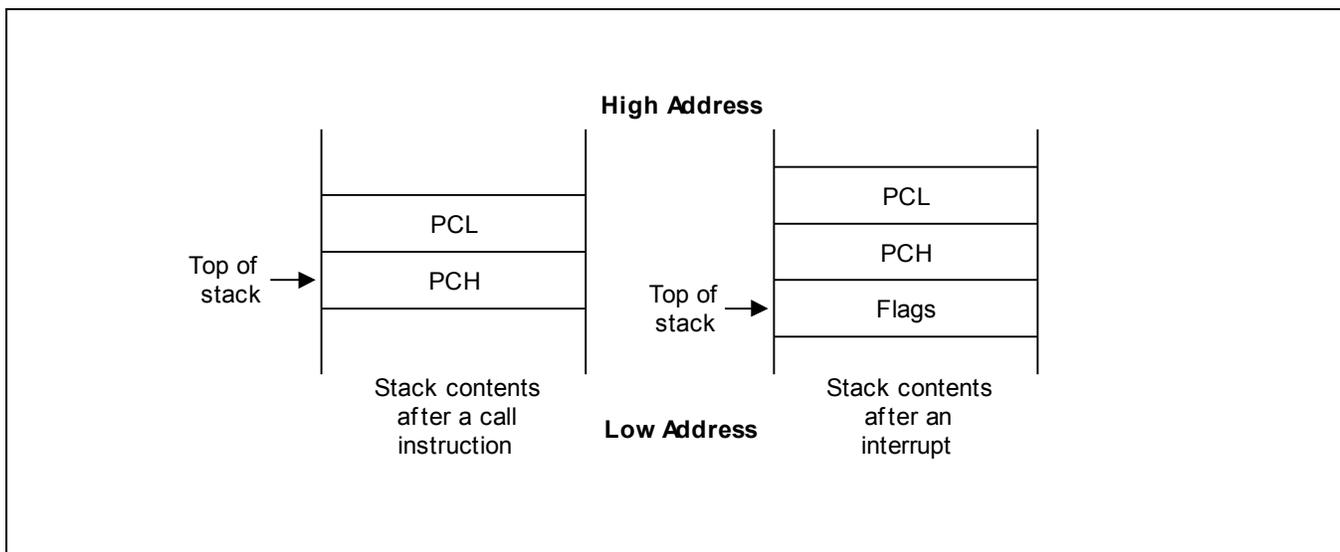
Figure 2-5. Stack Operations

2.6.2 Stack Pointer (SP)

Register location D9H contains the 8-bit stack pointer (SP) that is used for system stack operations. After a reset, the SP value is undetermined.

Because only internal memory 192 bytes space is implemented in the S3F94C8/S3F94C4, the SP must be initialized to an 8-bit value in the range 00H–0C0H.

NOTE

> In case a Stack Pointer is initialized to 00H, it is decreased to FFH when stack operation starts. This means that a Stack Pointer access invalid stack area. We recommend that a stack pointer is initialized to C0H to set upper address of stack to BFH.

☞ PROGRAMMING TIP — Standard Stack Operations Using PUSH and POP

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

```
LD       SP,#0C0H       ;  SP ← C0H (Normally, the SP is set to C0H by the
                        ;   initialization routine)
.
.
.
PUSH     SYM            ;  Stack address 0BFH ← SYM
PUSH     R15            ;  Stack address 0BEH ← R15
PUSH     20H            ;  Stack address 0BDH ← 20H
PUSH     R3             ;  Stack address 0BCH ← R3
.
.
.
POP      R3             ;  R3  ← Stack address 0BCH
POP      20H            ;  20H ← Stack address 0BDH
POP      R15            ;  R15 ← Stack address 0BEH
POP      SYM            ;  SYM ← Stack address 0BFH
```

3 Addressing Modes

3.1 Overview

Instructions that are stored in program memory are fetched for execution using the program counter. Instructions indicate the operation to be performed and the data to be operated on. *Addressing mode* is the method used to determine the location of the data operand. The operands specified in SAM88RCRI instructions may be condition codes, immediate data, or a location in the register file, program memory, or data memory.

The SAM88RCRI instruction set supports six explicit addressing modes. Not all of these addressing modes are available for each instruction. The addressing modes and their symbols are as follows:

Register (R)

Indirect Register (IR)

Indexed (X)

Direct Address (DA)

Relative Address (RA)

Immediate (IM)

3.2 Register Addressing Mode (R)

In Register addressing mode, the operand is the content of a specified register (see Figure 3-1). Working register addressing differs from Register addressing because it uses an 16-byte working register space in the register file and an 4-bit register within that space (see Figure 3-2).

Program Memory

Register File

8-Bit Register
File Address

dst

OPCODE

One-Operand
Instruction
(Example)

Point to one
register in register
file

OPERAND

Value used in
Instruction Execution

Sample Instruction:

DEC   CNTR   ;   Where CNTR is the label of an 8-bit register address
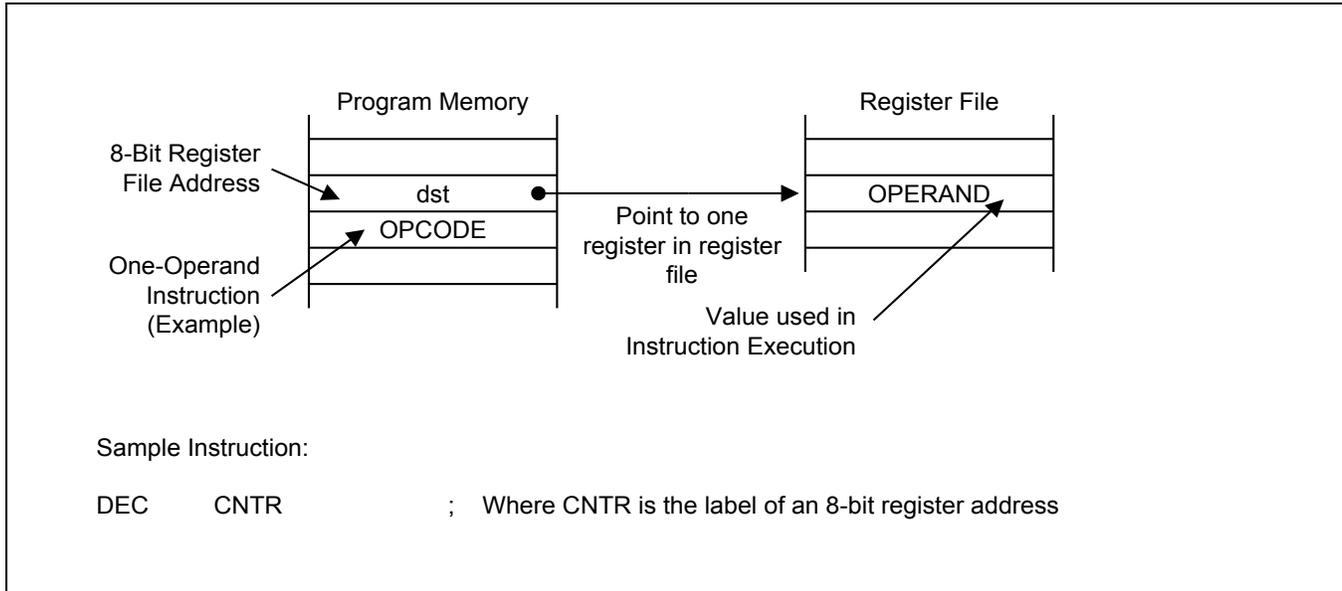
Figure 3-1. Register Addressing

Register File

CFH

Program Memory

4-Bit
Working Register

dst   src

OPCODE

Two-Operand
Instruction
(Example)

4 LSBs

Point to the
working register
(1 of 16)

OPERAND

C0H

Sample Instruction:

ADD   R1, R2   ;   Where R1 and R2 are registers in the currently selected
                                 working register area.
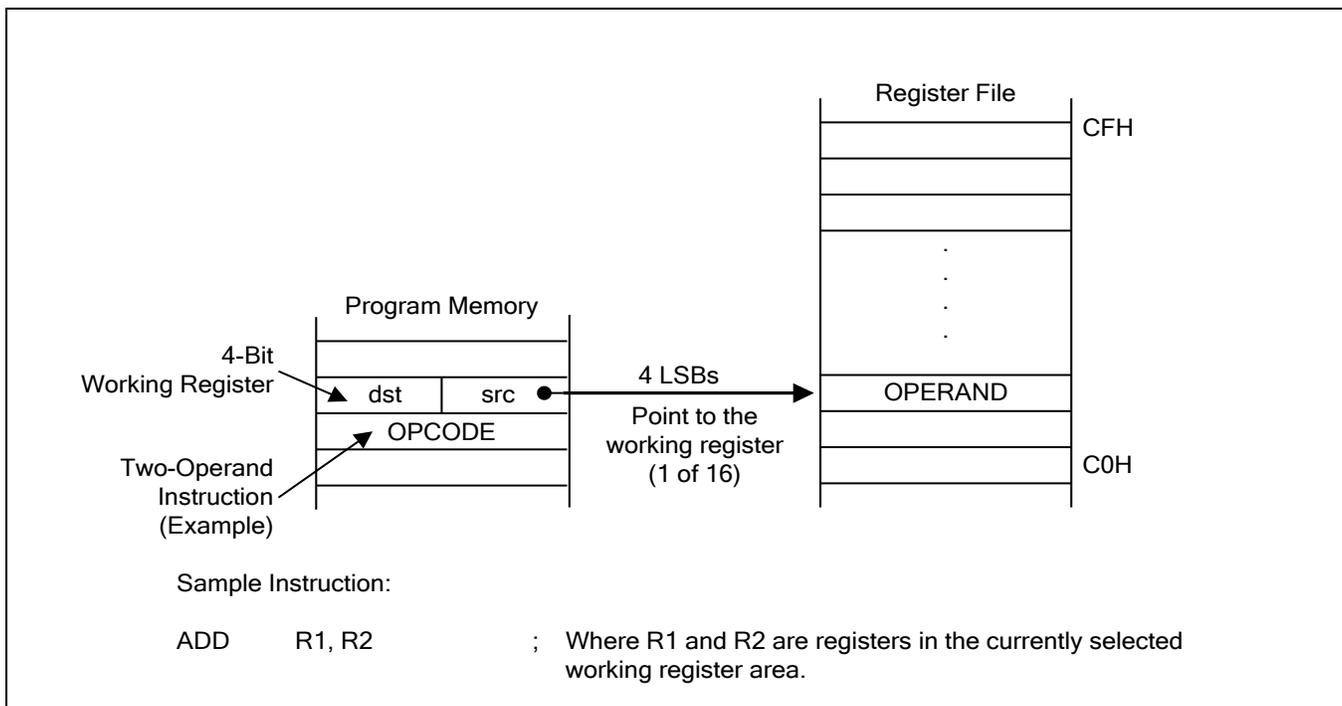
Figure 3-2. Working Register Addressing

3.3 Indirect Register Addressing Mode (IR)

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

You can use any 8-bit register to indirectly address another register. Any 16-bit register pair can be used to indirectly address another memory location.

Program Memory

Register File

8-Bit Register
File Address

dst

ADDRESS

Point to one
register in register
file

OPCODE

One-Operand
Instruction (Example)

Address of operand
used by instruction

OPERAND

Value used in
instruction execution

Sample Instruction:

RL   @SHIFT   ;   Where SHIFT is the label of an 8-bit register ddress
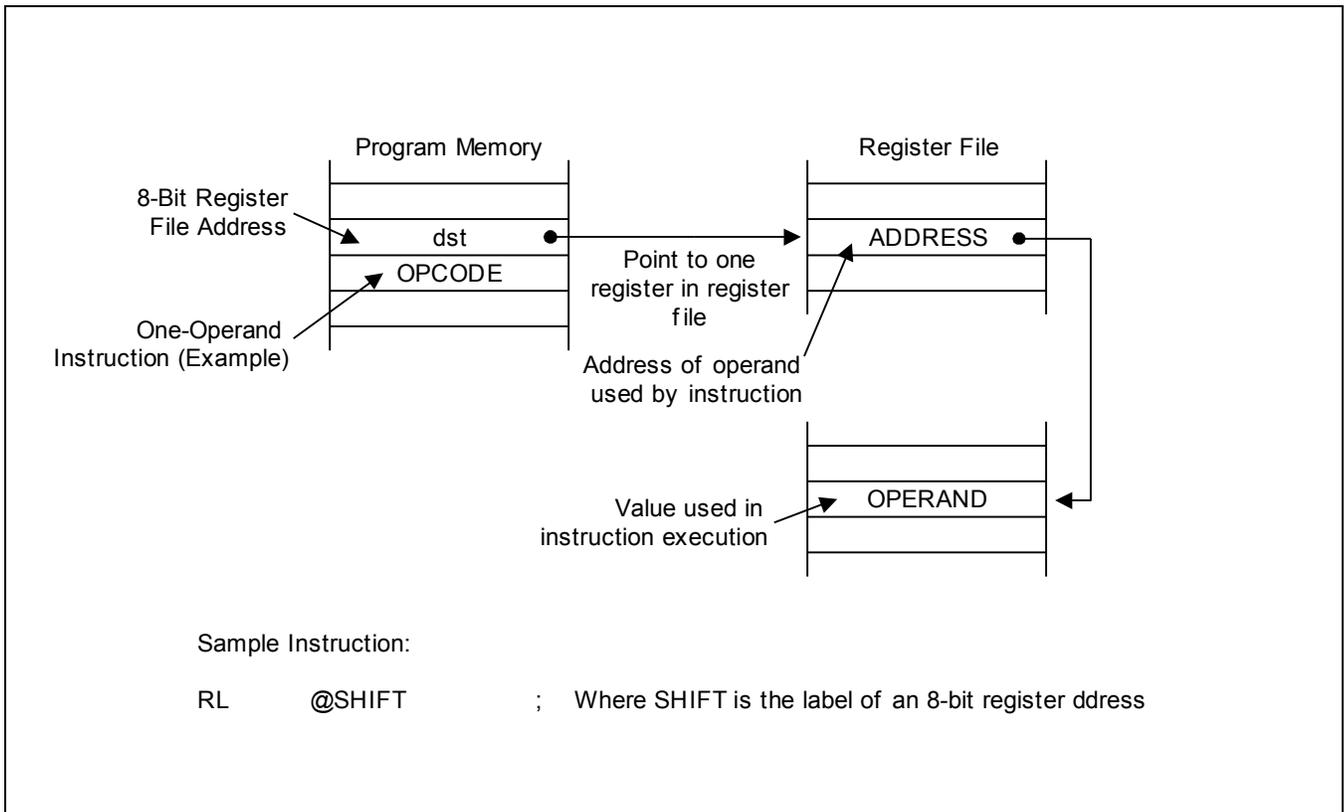
Figure 3-3. Indirect Register Addressing to Register File
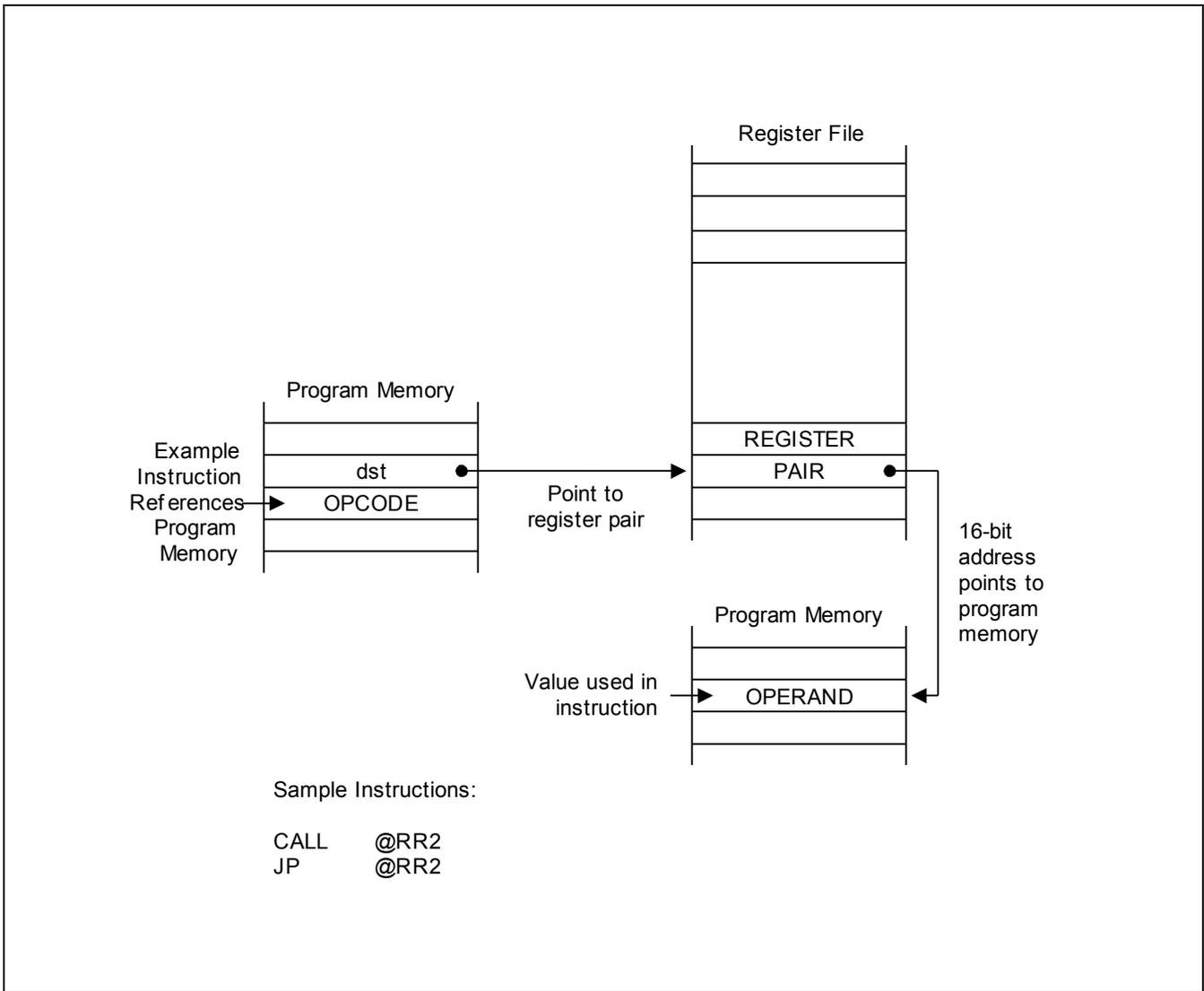
Figure 3-4. Indirect Register Addressing to Program Memory
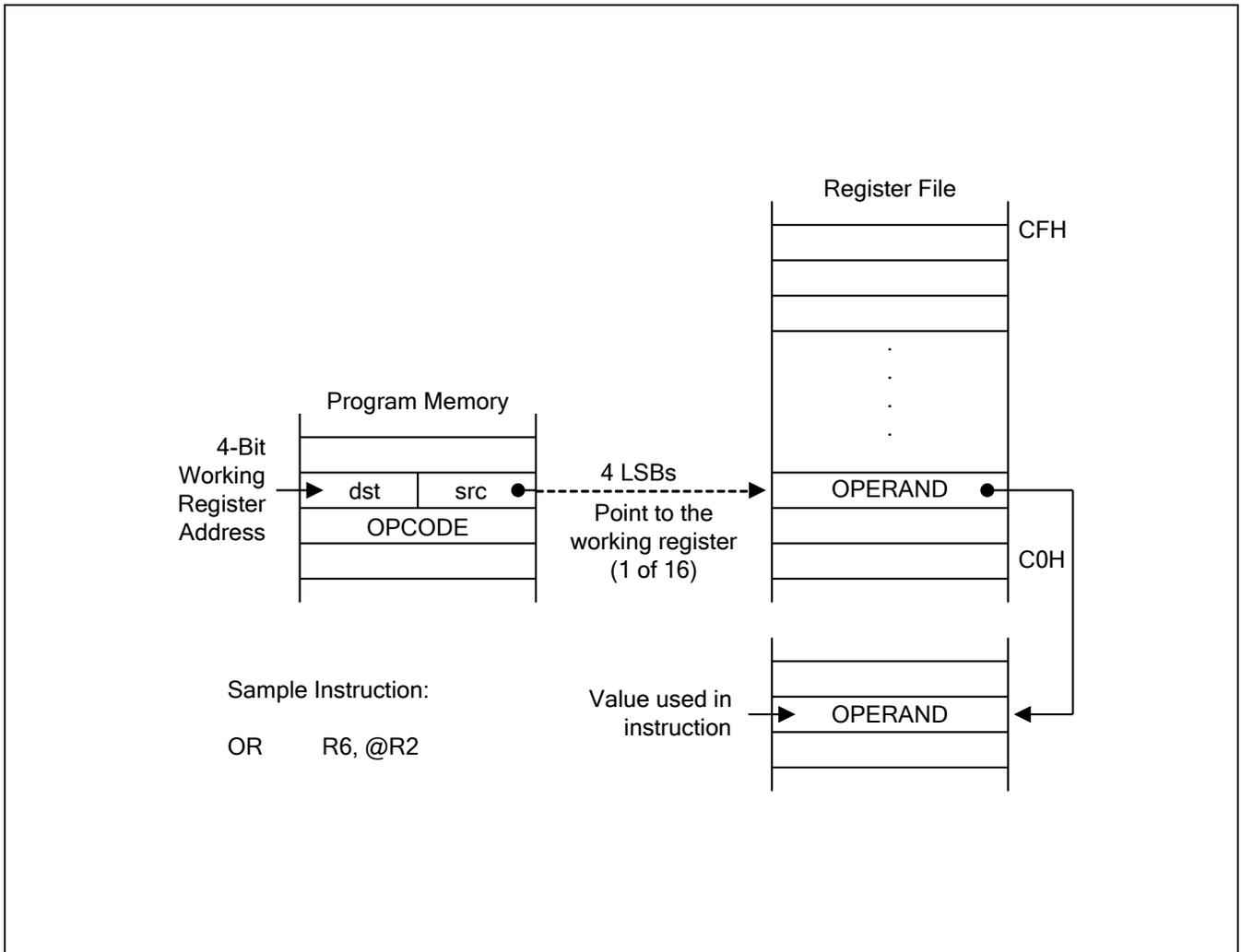
Figure 3-5. Indirect Working Register Addressing to Register File
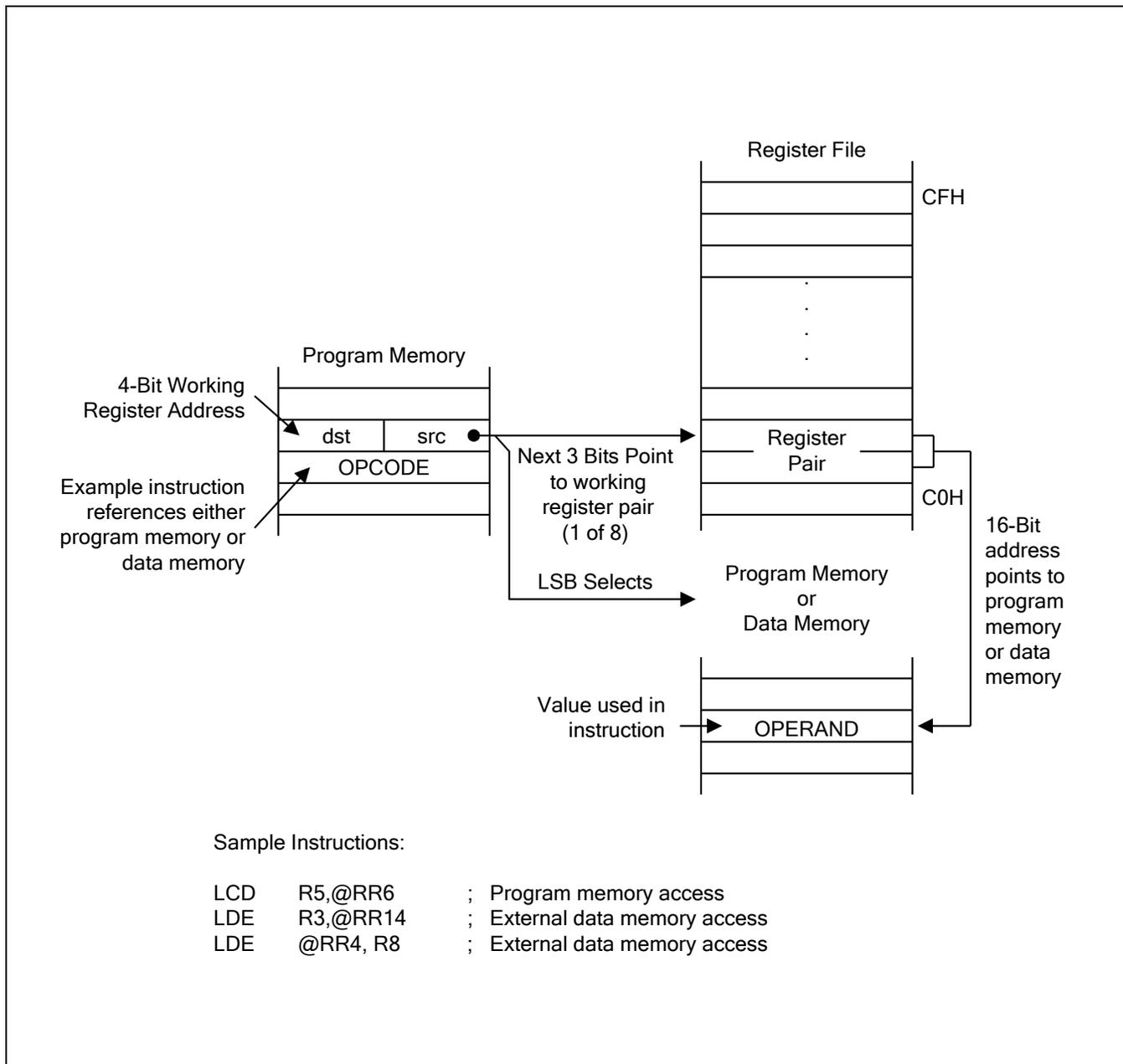
Figure 3-6. Indirect Working Register Addressing to Program or Data Memory

3.4 Indexed Addressing Mode (X)

Indexed (X) addressing mode adds an offset value to a base address during instruction execution in order to calculate the effective operand address (see Figure 3-7). You can use Indexed addressing mode to access locations in the internal register file or in external memory.

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

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

The only instruction that supports Indexed addressing mode for the internal register file is the Load instruction (LD). The LDC and LDE instructions support Indexed addressing mode for internal program memory, external program memory, and for external data memory, when implemented.
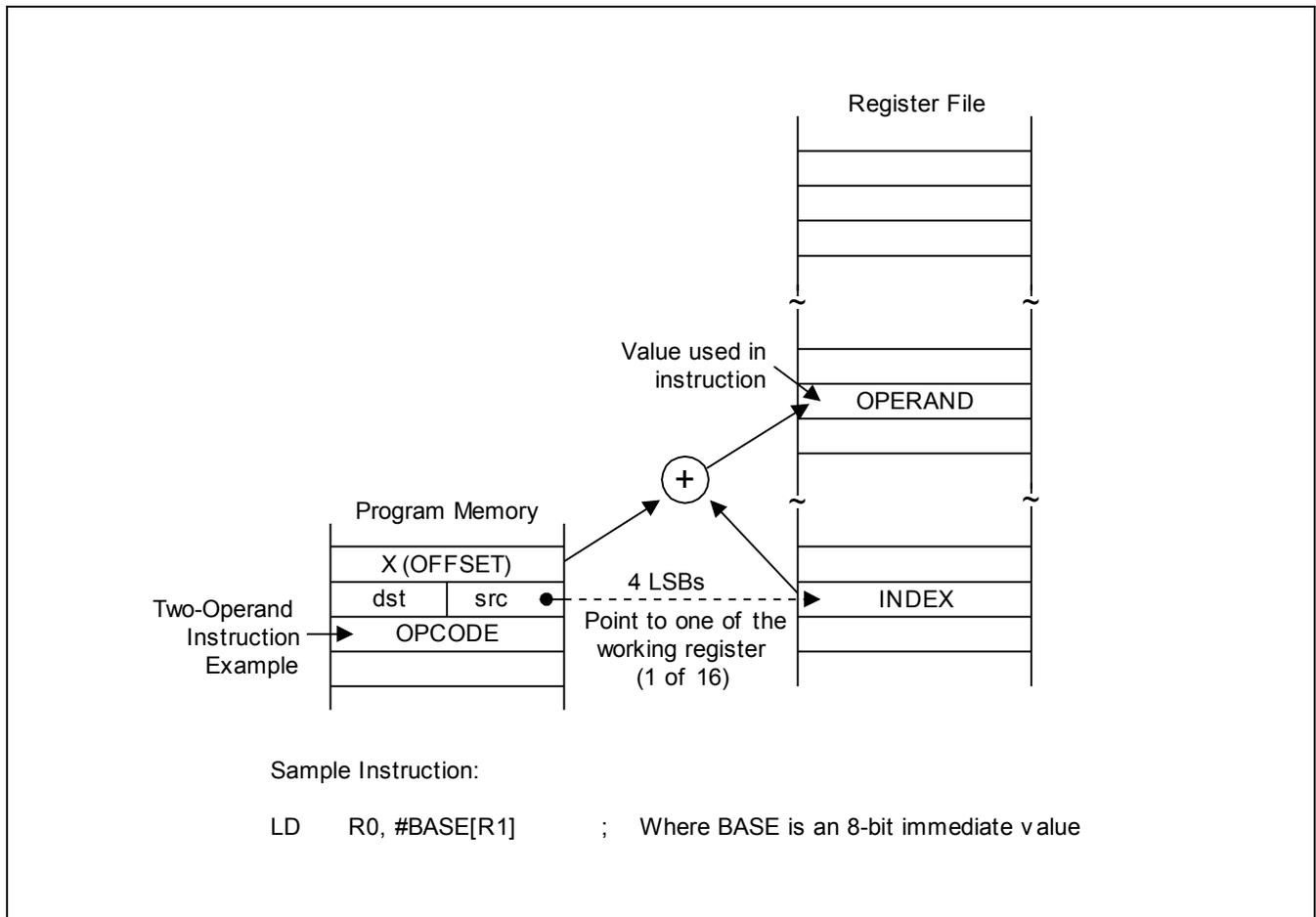
Figure 3-7. Indexed Addressing to Register File
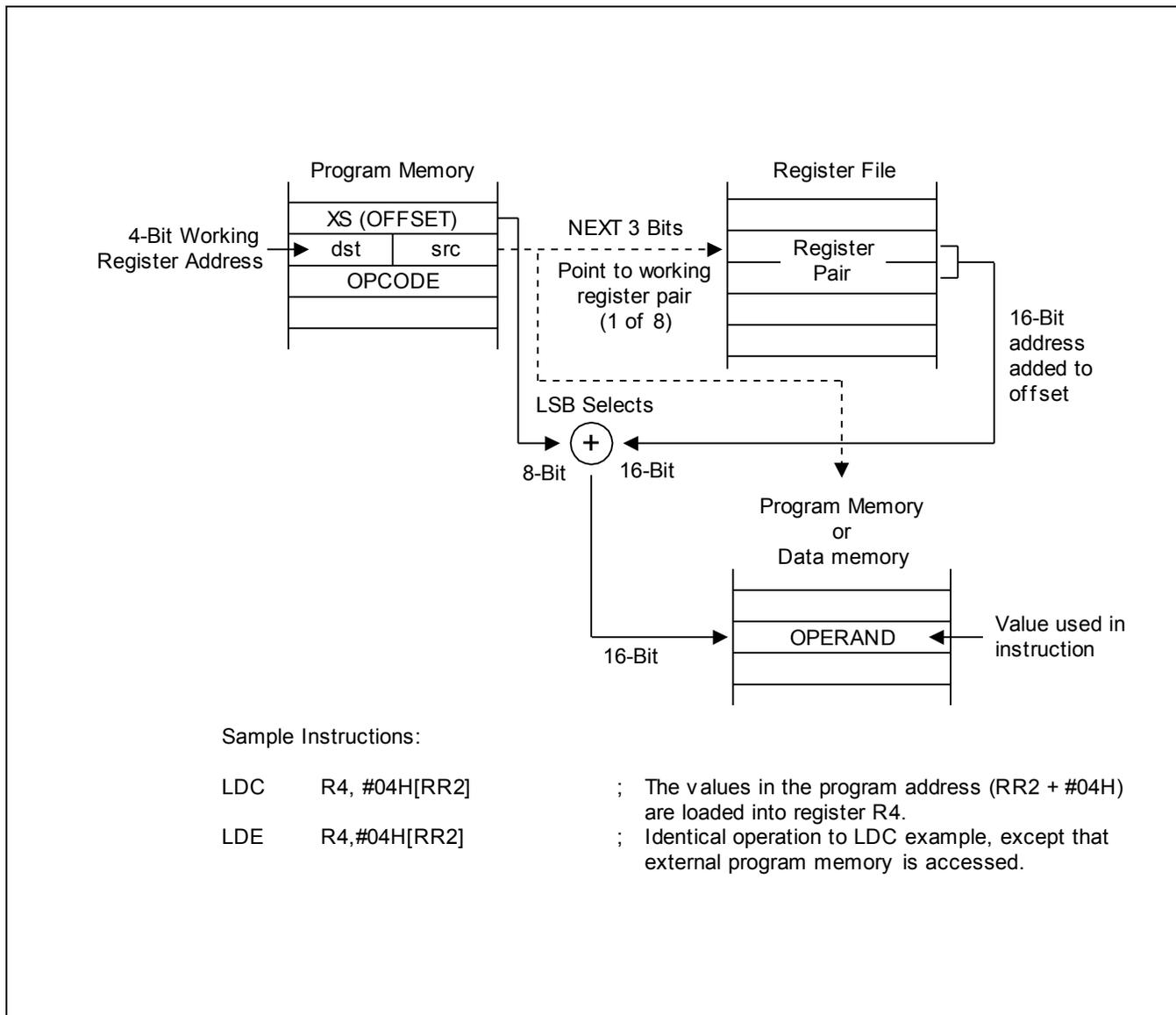
Figure 3-8. Indexed Addressing to Program or Data Memory with Short Offset
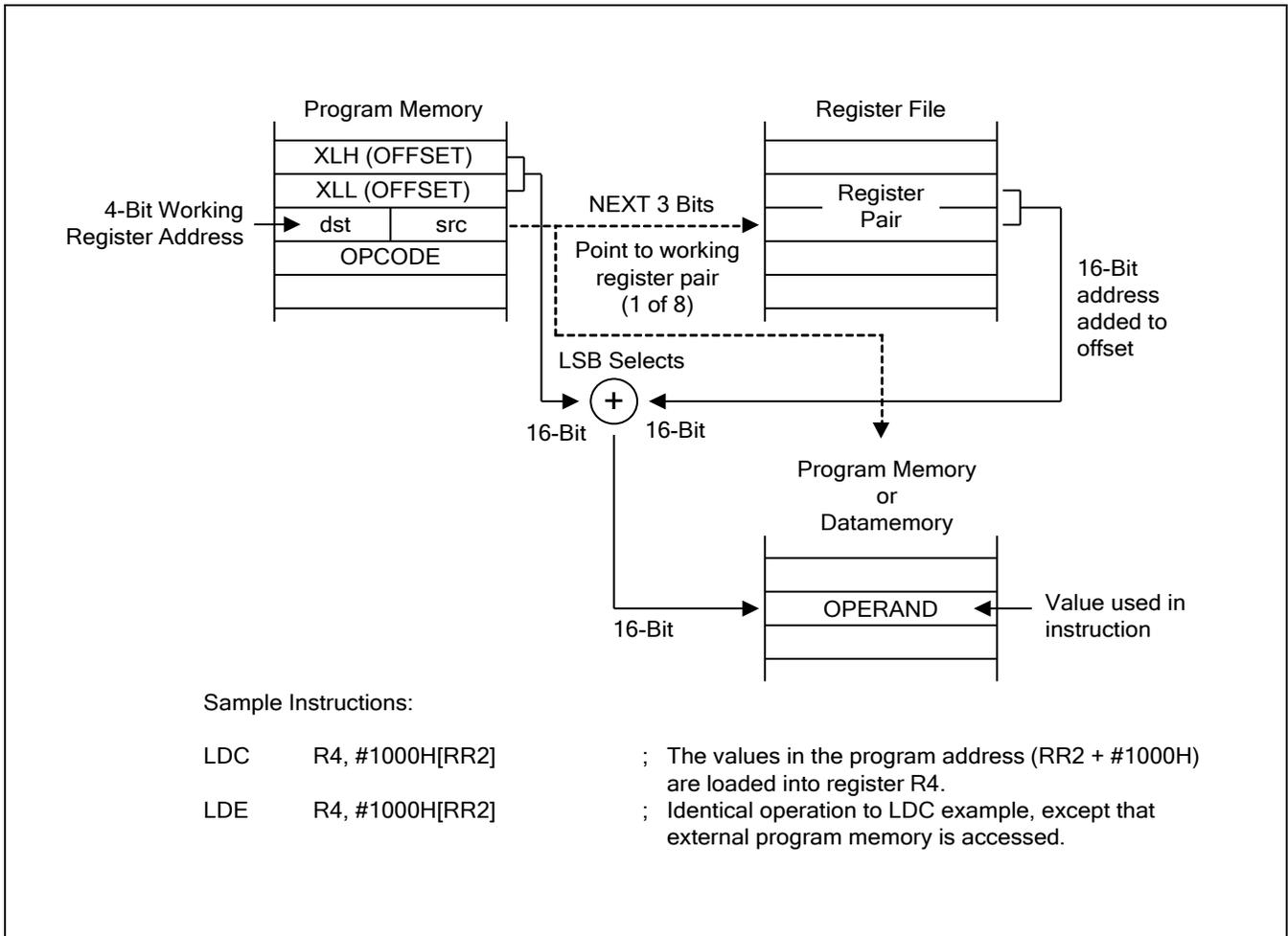
Figure 3-9. Indexed Addressing to Program or Data Memory with Long Offset

3.5 Direct Address Mode (DA)

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

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

Program or
Data Memory

Program Memory

Memory
Address
Used

| Upper Address Byte |
| Lower Address Byte |

dst/src | "0" or "1"

OPCODE

LSB Selects Program
Memory or Data Memory:
"0" = Program Memory
"1" = Data Memory

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 3-10. Direct Addressing for Load Instructions

Program Memory

Next OPCODE

Program
Memory
Address
Used

Lower Address Byte
Upper Address Byte
OPCODE

Sample Instructions:

JP   C,JOB1   ;   Where JOB1 is a 16-bit immediate address
CALL   DISPLAY   ;   Where DISPLAY is a 16-bit immediate address
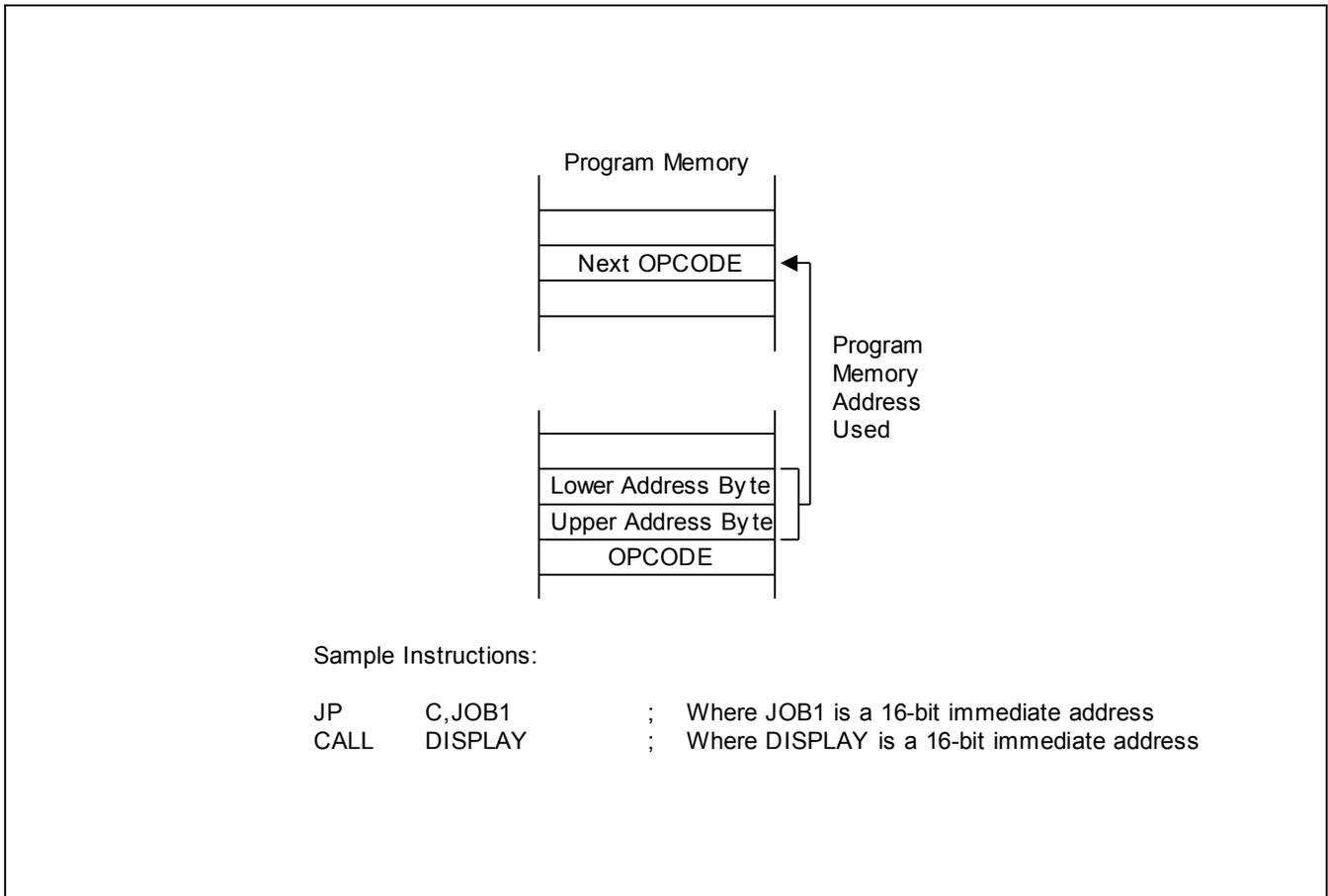
Figure 3-11. Direct Addressing for Call and Jump Instructions

3.6 Relative Address Mode (RA)

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

The instruction that support RA addressing is JR.
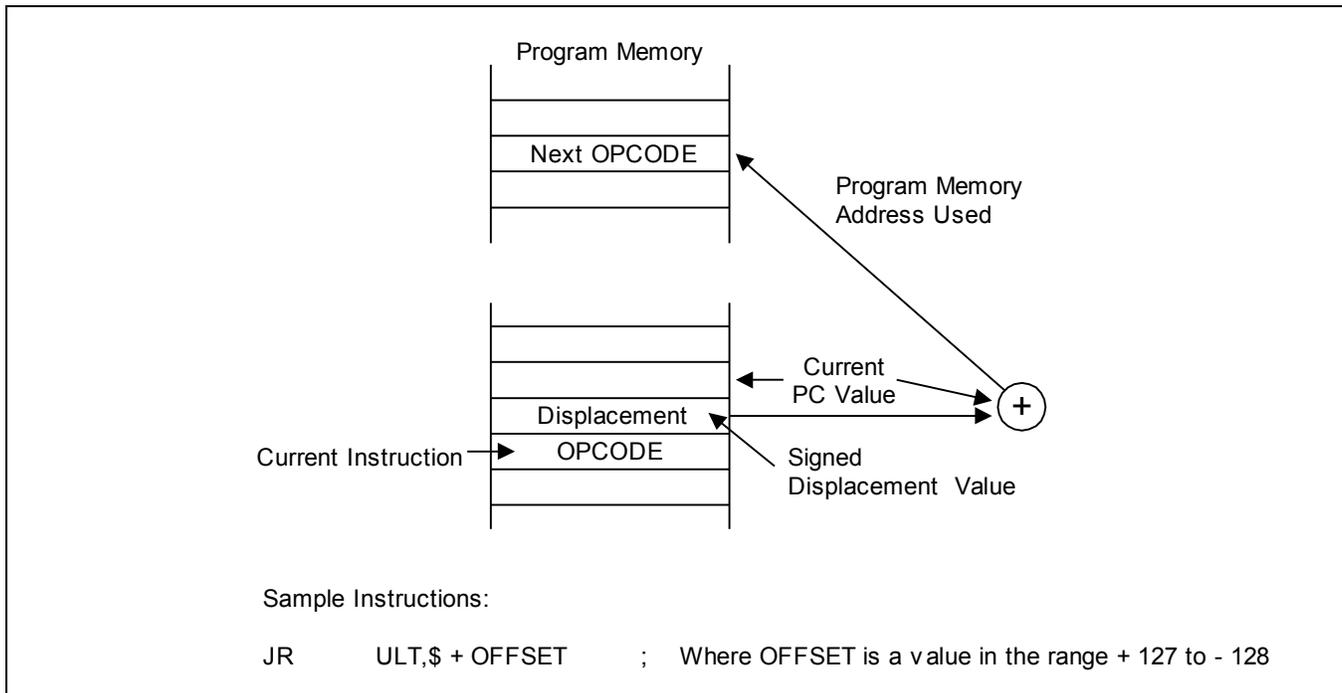
Figure 3-12. Relative Addressing

3.7 Immediate Mode (IM)

In Immediate (IM) addressing mode, the operand value used in the instruction is the value supplied in the operand field itself. Immediate addressing mode is useful for loading constant values into registers.
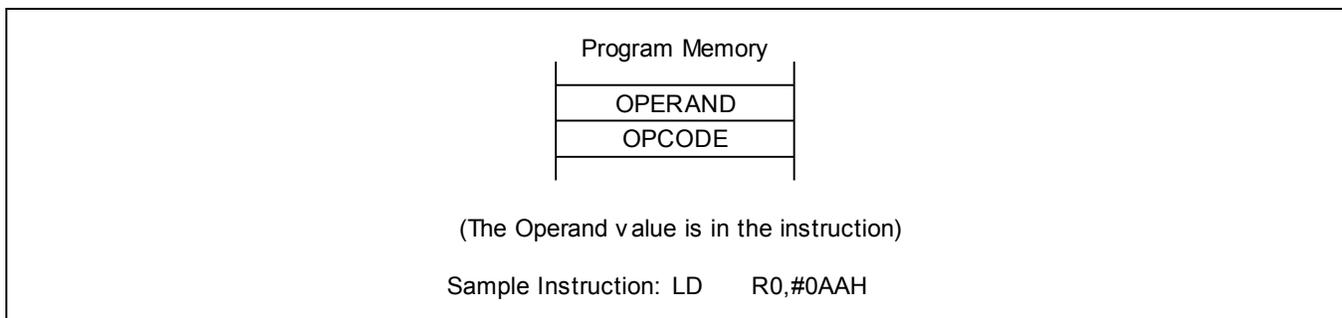
Figure 3-13. Immediate Addressing

4 Control Registers

4.1 Overview

In this section, detailed descriptions of the S3F94C8/S3F94C4 control registers are presented in an easy-to-read format. These descriptions will help familiarize you with the mapped locations in the register file. You can also use them as a quick-reference source when writing application programs.

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

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

Table 4-1. System and Peripheral Control Registers

Register Name	Mnemonic	Address & Location		RESET Value (Bit)							
		Address	R/W	7	6	5	4	3	2	1	0
Timer 0 Counter Register	T0CNT	D0H	R	0	0	0	0	0	0	0	0
Timer 0 Data Register	T0DATA	D1H	R/W	1	1	1	1	1	1	1	1
Timer 0 Control Register	T0CON	D2H	R/W	0	0	–	–	0	–	0	0
Location D3H is not mapped											
Clock Control Register	CLKCON	D4H	R/W	0	–	–	0	0	–	–	–
System Flags Register	FLAGS	D5H	R/W	x	x	x	x	–	–	–	–
Locations D6H–D8H are not mapped											
Stack Pointer Register	SP	D9H	R/W	x	x	x	x	x	x	x	x
Location DAH is not mapped											
MDS Special Register	MDSREG	DBH	R/W	0	0	0	0	0	0	0	0
Basic Timer Control Register	BTCON	DCH	R/W	0	0	0	0	0	0	0	0
Basic Timer Counter Register	BTCNT	DDH	R	0	0	0	0	0	0	0	0
Test Mode Control Register	FTSTCON	DEH	W	–	–	0	0	0	0	0	0
System Mode Register	SYM	DFH	R/W	–	–	–	–	0	0	0	0

NOTES:

1. – : Not mapped or not used, x: Undefined
2. The register, FTSTCON, is no use. Its value should always be '00H' during the normal operation.

Table 4-1. System and Peripheral Control Registers (Continued)

Register Name	Mnemonic	Address Hex	R/W	7	6	5	4	3	2	1	0
				Bit Values After RESET							
Port 0 Data Register	P0	E0H	R/W	0	0	0	0	0	0	0	0
Port 1 Data Register	P1	E1H	R/W	–	–	–	–	–	0	0	0
Port 2 Data Register	P2	E2H	R/W	–	0	0	0	0	0	0	0
Locations E3H–E5H are not mapped											
Port 0 Control High Byte Register	P0CONH	E6H	R/W	0	0	0	0	0	0	0	0
Port 0 Control Register	P0CONL	E7H	R/W	0	0	0	0	0	0	0	0
Port 0 Interrupt Pending Register	P0PND	E8H	R/W	–	–	–	–	0	0	0	0
Port 1 Control Register	P1CON	E9H	R/W	0	0	–	–	0	0	0	0
Port 2 Control High Byte Register	P2CONH	EAH	R/W	–	0	0	0	0	0	0	0
Port 2 Control Low Byte Register	P2CONL	EBH	R/W	0	0	0	0	0	0	0	0
Flash Memory Control Register	FMCON	ECH	R/W	0	0	0	0	–	–	–	0
Flash Memory User Programming Enable Register	FMUSR	EDH	R/W	0	0	0	0	0	0	0	0
Flash Memory Sector Address High Byte Register	FMSECH	EEH	R/W	0	0	0	0	0	0	0	0
Flash Memory Sector Address Low Byte Register	FMSECL	EFH	R/W	0	0	0	0	0	0	0	0
PWM Data Register 1	PWMDATA1	F0H	R/W	0	0	0	0	0	0	0	0
PWM Extension Register	PWMEX	F1H	R/W	0	0	0	0	0	0	0	0
PWM Data Register	PWMDATA	F2H	R/W	0	0	0	0	0	0	0	0
PWM Control Register	PWMCON	F3H	R/W	0	0	–	0	0	0	0	0
Stop Control Register	STOPCON	F4H	R/W	0	0	0	0	0	0	0	0
Locations F5H–F6H are not mapped											
A/D Control Register	ADCON	F7H	R/W	0	0	0	0	0	0	0	0
A/D Converter Data High Byte Register	ADDATAH	F8H	R	x	x	x	x	x	x	x	x
A/D Converter Data Low Byte Register	ADDATAL	F9H	R	0	0	0	0	0	0	x	x
Locations FAH–FFH are not mapped											

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

Bit number(s) that is/are appended to the
register name for bit addressing

Name of individual
bit or related bits

Register
ID

Register name

Register address
(hexadecimal)

FLAGS **- System Flags Register**   **D5H**

Bit Identifier

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

RESET Value

x	x	x	x	x	x	0	0

Read/Write

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7

Carry Flag (C)

0	Operation dose not generate a carry or borrow condition
1	Operation generates carry-out or borrow into high-order bit7

.6

Zero Flag

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

.5

Sign Flag

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

R = Read-only
W = Write-only
R/W = Read/write
' - ' = Not used

Description of the
effect of specific
bit settings

RESET value notation:
'-' = Not used
'x' = Undetermind value
'0' = Logic zero
'1' = Logic one

Bit number:
MSB = Bit 7
LSB = Bit 0

Figure 4-1. Register Description Format

4.1.1 ADCON — A/D Converter Control Register (F7H)

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

.7–.4   A/D Converter Input Pin Selection Bits

0	0	0	0	ADC0 (P0.0)
0	0	0	1	ADC1 (P0.1)
0	0	1	0	ADC2 (P0.2)
0	0	1	1	ADC3 (P0.3)
0	1	0	0	ADC4 (P0.4)
0	1	0	1	ADC5 (P0.5)
0	1	1	0	ADC6 (P0.6)
0	1	1	1	ADC7 (P0.7)
1	0	0	0	ADC8 (P2.6)
1	0	0	1	Connected with GND internally
1	0	1	0	Connected with GND internally
1	0	1	1	Connected with GND internally
1	1	0	0	Connected with GND internally
1	1	0	1	Connected with GND internally
1	1	1	0	Connected with GND internally
1	1	1	1	Connected with GND internally

.3   End-of-Conversion Status Bit

0	A/D conversion is in progress
1	A/D conversion complete

.2–.1   Clock Source Selection Bit **(note)**

0	0	$f_{OSC}/16$ ($f_{OSC} \leq 10$ MHz)
0	1	$f_{OSC}/8$ ($f_{OSC} \leq 10$ MHz)
1	0	$f_{OSC}/4$ ($f_{OSC} \leq 10$ MHz)
1	1	$f_{OSC}/1$ ($f_{OSC} \leq 4$ MHz)

.0   Conversion Start Bit

0	No meaning
1	A/D conversion start

NOTE: Maximum ADC clock input = **4 MHz**.

4.1.2 BTCON — Basic Timer Control Register (DCH)

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

.7–.4   **Watchdog Timer Function Enable Bit**

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

.3–.2   **Basic Timer Input Clock Selection Code**

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

.1   **Basic Timer 8-Bit Counter Clear Bit**

0	No effect
1	Clear the basic timer counter value

.0   **Basic Timer and Timer 0 Divider Clear Bit**

0	No effect
1	Clear both dividers

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

4.1.3 CLKCON — Clock Control Register (D4H)

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

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

0	Enable IRQ for main system oscillator wake-up function
1	Disable IRQ for main system oscillator wake-up function

.6–.5

Not used for S3F94C8/S3F94C4

.4–.3   **Divided by Selection Bits for CPU Clock Frequency**

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

.2–.0

Not used for S3F94C8/S3F94C4

4.1.4 FLAGS — System Flags Register (D5H)

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

.7   **Carry Flag (C)**

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

.6   **Zero Flag (Z)**

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

.5   **Sign Flag (S)**

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

.4   **Overflow Flag (V)**

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

.3–.0   Not used for S3F94C8/S3F94C4

NOTE: The unused bits .3-.0 should always be kept as 0 in normal operation; otherwise it may be cause error execution.

4.1.5 FMCON — Flash Memory Control Register (ECH)

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

.7–.4   Flash Memory Mode Selection Bits

0	1	0	1	Programming mode
1	0	1	0	Sector erase mode
0	1	1	0	Hard lock mode
Other values				Not available

.3–.1

Not used for the S3F94C8/S3F94C4

.0   Flash Operation Start Bit

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

4.1.6 FMSECH — Flash Memory Sector Address High Byte Register (EEH)

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

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

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

4.1.7 FMSECL — Flash Memory Sector Address Low Byte Register (EFH)

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

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

.6–.0	Bits 6–0
	Don't care

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

4.1.8 FMUSR — Flash Memory User Programming Enable Register (EDH)

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

.7–.0   Flash Memory User Programming Enable Bits

1 0 1 0 0 1 0 1	Enable user programming mode
Other values	Disable user programming mode

4.1.9 P0CONH — Port 0 Control High Byte Register (E6H)

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

.7–.6   Port 0, P0.7/ADC7 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	A/D converter input (ADC7); Schmitt trigger input off

.5–.4   Port 0, P0.6/ADC6/PWM Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Alternative function (PWM output)
1	0	Push-pull output
1	1	A/D converter input (ADC6); Schmitt trigger input off

.3–.2   Port 0, P0.5/ADC5 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	A/D converter input (ADC5); Schmitt trigger input off

.1–.0   Port 0, P0.4/ADC4 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	A/D converter input (ADC4); Schmitt trigger input off

4.1.10 P0CONL — Port 0 Control Low Byte Register (E7H)

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

.7–.6   Port 0, P0.3/ADC3 Configuration Bits

0	0	Schmitt trigger input
0	1	Schmitt trigger input; pull-up enable
1	0	Push-pull output
1	1	A/D converter input (ADC3); Schmitt trigger input off

.5–.4   Port 0, P0.2/ADC2 Configuration Bits

0	0	Schmitt trigger input
0	1	Schmitt trigger input; pull-up enable
1	0	Push-pull output
1	1	A/D converter input (ADC2); Schmitt trigger input off

.3–.2   Port 0, P0.1/ADC1/INT1 Configuration Bits

0	0	Schmitt trigger input/falling edge interrupt input
0	1	Schmitt trigger input; pull-up enable/falling edge interrupt input
1	0	Push-pull output
1	1	A/D converter input (ADC1); Schmitt trigger input off

.1–.0   Port 0, P0.0/ADC0/INT0 Configuration Bits

0	0	Schmitt trigger input/falling edge interrupt input
0	1	Schmitt trigger input; pull-up enable/falling edge interrupt input
1	0	Push-pull output
1	1	A/D converter input (ADC0); Schmitt trigger input off

4.1.11 P0PND — Port 0 Interrupt Pending Register (E8H)

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

.7–.4	Not used for the S3F94C8/S3F94C4

.3	Port 0.1/ADC1/INT1 Interrupt Enable Bit	
	0	INT1 falling edge interrupt disable
	1	INT1 falling edge interrupt enable

.2	Port 0.1/ADC1/INT1 Interrupt Pending Bit	
	0	No interrupt pending (when read)
	0	Pending bit clear (when write)
	1	Interrupt is pending (when read)
	1	No effect (when write)

.1	Port 0.0/ADC0/INT0 Interrupt Enable Bit	
	0	INT0 falling edge interrupt disable
	1	INT0 falling edge interrupt enable

.0	Port 0.0/ADC0/INT0 Interrupt Pending Bit	
	0	No interrupt pending (when read)
	0	Pending bit clear (when write)
	1	Interrupt pending (when read)
	1	No effect (when write)

4.1.12 P1CON — Port 1 Control Register (E9H)

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

.7   Part 1.1 N-channel open-drain Enable Bit

0	Configure P1.1 as a push-pull output
1	Configure P1.1 as a n-channel open-drain output

.6   Port 1.0 N-channel open-drain Enable Bit

0	Configure P1.0 as a push-pull output
1	Configure P1.0 as a n-channel open-drain output

.5–.4   Not used for S3F94C8/S3F94C4

.3–.2   Port 1, P1.1 Interrupt Pending Bits

0	0	Schmitt trigger input;
0	1	Schmitt trigger input; pull-up enable
1	0	Output
1	1	Schmitt trigger input; pull-down enable

.1–.0   Port 1, P1.0 Configuration Bits

0	0	Schmitt trigger input;
0	1	Schmitt trigger input; pull-up enable
1	0	Output
1	1	Schmitt trigger input; pull-down enable

NOTE: When you use external oscillator, P1.0, P1.1 must be set to output port to prevent current consumption.

4.1.13 P2CONH — Port 2 Control High Byte Register (EAH)

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

.7	Not used for the S3F94C8/S3F94C4

.6–.4   Port 2, P2.6/ADC8/CLO Configuration Bits

0	0	0	Schmitt trigger input; pull-up enable
0	0	1	Schmitt trigger input
0	1	x	ADC input
1	0	0	Push-pull output
1	0	1	Open-drain output; pull-up enable
1	1	0	Open-drain output
1	1	1	Alternative function; CLO output

.3–.2   Port 2, 2.5 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	Open-drain output

.1–.0   Port 2, 2.4 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	Open-drain output

NOTE: When noise problem is important issue, you had better not use CLO output.

4.1.14 P2CONL — Port 2 Control Low Byte Register (EBH)

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

.7–.6   Part 2, P2.3 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	Open-drain output

.5–.4   Port 2, P2.2 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	Open-drain output

.3–.2   Port 2, P2.1 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	Open-drain output

.1–.0   Port 2, P2.0 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	T0 match output

4.1.15 PWMCON — PWM Control Register (F3H)

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

.7–.6   PWM Input Clock Selection Bits

0	0	$f_{OSC}/64$
0	1	$f_{OSC}/8$
1	0	$f_{OSC}/2$
1	1	$f_{OSC}/1$

.5

Not used for S3F94C8/S3F94C4

.4   PWMDATA Reload Interval Selection Bit

0	Reload from extension upcounter overflow
1	Reload from base upcounter overflow

.3   PWM Counter Clear Bit

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

.2   PWM Counter Enable Bit

0	Stop counter
1	Start (Resume countering)

.1   PWM Overflow Interrupt Enable Bit (8-Bit Overflow)

0	Disable interrupt
1	Enable interrupt

.0   PWM Overflow Interrupt Pending Bit

0	No interrupt pending (when read)

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

NOTES:

1. PWMCON.3 is not auto-cleared. You must pay attention when clear pending bit. (refer to page 11-12).

2. PWMCON.5 should always be set to 0.

4.1.16 PWMEX — PWM Extension Register (F1H)

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

.7–.2   PWM Extension Bits

PWM extension bits for 6+6 resolution and 8+6 resolution; Not used in 6+2 resolution

.1–.0   PWM Base/extension Control bits:

0	0	Base 6-bit (PWMDATA.7-.2) + Extension 2-bit (PWMDATA.1-.0)
1	0	
0	1	Base 6-bit (PWMDATA1.5-.0) + Extension 6-bit (PWMEX.7-.2)
1	1	Base 8-bit (PWMDATA1.7-.0) + Extension 6-bit (PWMEX.7-.2)

4.1.17 STOPCON — Stop Mode Control Register (E4H)

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

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

NOTE: When the STOPCON Register is not #0A5H value, if you use STOP instruction, PC is changed to reset address.

4.1.18 SYM — System Mode Register (DFH)

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

.7–.4

Not used for S3F94C8/S3F94C4

.3   Global Interrupt Enable Bit

0	Disable all interrupts
1	Enable all interrupt

.2–.0   Page Select Bits

0	0	0	Page 0
0	0	1	Page 1 (Not used for S3F94C8/S3F94C4)
0	1	0	Page 2 (Not used for S3F94C8/S3F94C4)
0	1	1	Page 3 (Not used for S3F94C8/S3F94C4)

4.1.19 T0CON — Timer 0 Control Register (D2H)

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

.7–.6   Timer 0 Input Clock Selection Bits

0	0	$f_{OSC}/4096$
0	1	$f_{OSC}/256$
1	0	$f_{OSC}/8$
1	1	$f_{OSC}/1$

.5–.4

Not used for the S3F94C8/S3F94C4

.3   Timer 0 Counter Clear Bit

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

.2

Not used for the S3F94C8/S3F94C4

.1   Timer 0 Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0   Timer 0 Interrupt Pending Bit (Match interrupt)

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

NOTES:

1. T0CON.3 is not auto-cleared. You must pay attention when clear pending bit. (refer to page 10-12)
2. To use T0 match output, set T0CON.3 to 1. (refer to page 10-7)

5 Interrupt Structure

5.1 Overview

The SAM88RCRI interrupt structure has two basic components: a vector, and sources. The number of interrupt sources can be serviced through an interrupt vector which is assigned in ROM address 0000H.
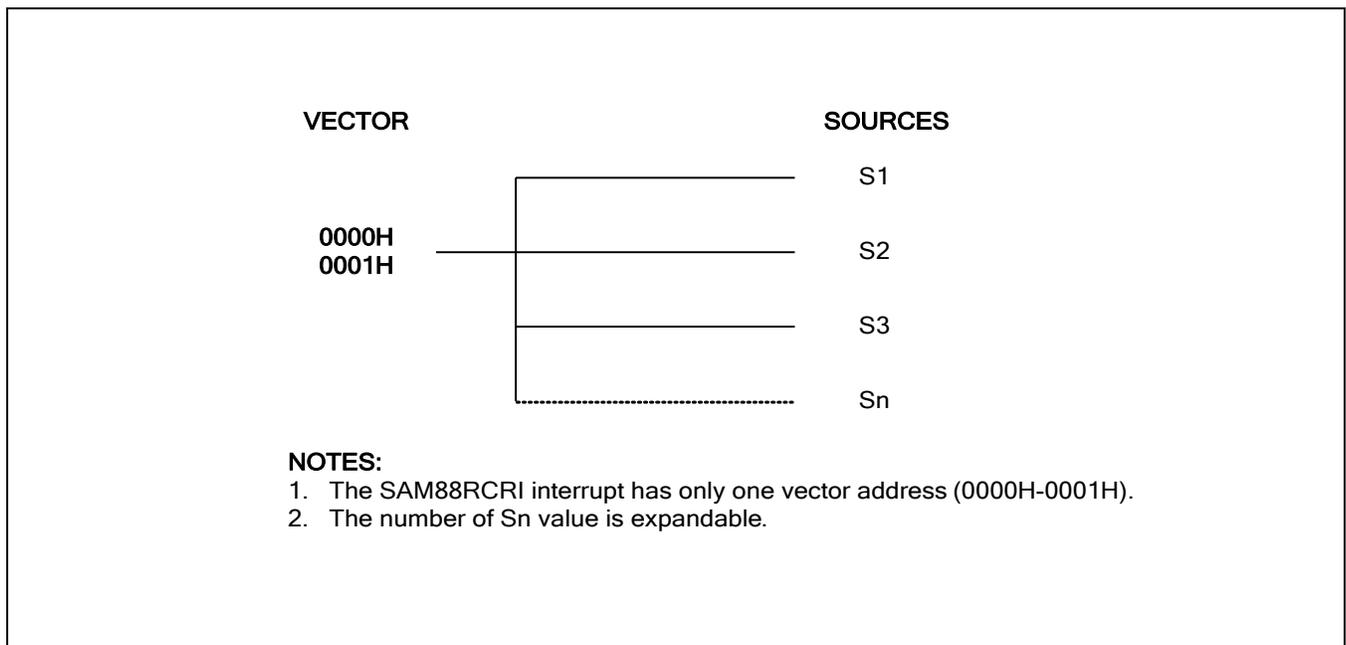
Figure 5-1. S3F9-Series Interrupt Type

5.2 Interrupt Processing Control Points

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

Global interrupt enable and disable (by EI and DI instructions)

Interrupt source enable and disable settings in the corresponding peripheral control register(s)

5.3 Enable/Disable Interrupt Instructions (EI, DI)

The system mode register, SYM (DFH), is used to enable and disable interrupt processing.

SYM.3 is the enable and disable bit for global interrupt processing respectively, by modifying SYM.3.

NOTE

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

Although you can manipulate SYM.3 directly to enable and disable interrupts during normal operation, we recommend that you use the EI and DI instructions for this purpose.

5.4 Interrupt Pending Function Types

When the interrupt service routine has executed, the application program's service routine must clear the appropriate pending bit before the return from interrupt subroutine (IRET) occurs.

5.5 Interrupt Priority

Because there is not an interrupt priority register in SAM88RCRI, the order of service is determined by a sequence of source which is executed in interrupt service routine.
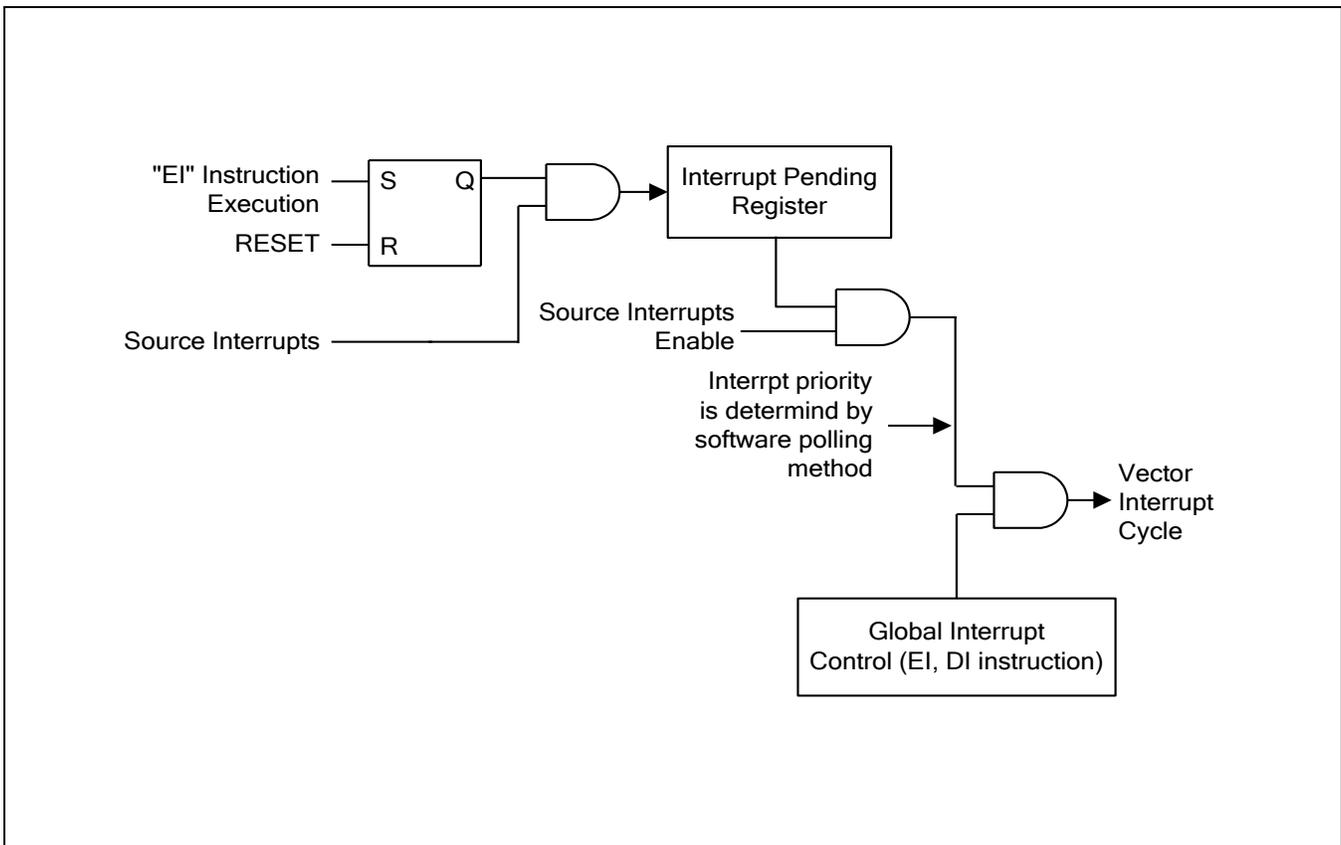
Figure 5-2. Interrupt Function Diagram

5.6 Interrupt Source Service Sequence

The interrupt request polling and servicing sequence is as follows:
1. A source generates an interrupt request by setting the interrupt request pending bit to 1.
2. The CPU generates an interrupt acknowledge signal.
3. The service routine starts and the source's pending flag is cleared to 0 by software.
    4. Interrupt priority must be determined by software polling method.

5.7 Interrupt Service Routines

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

Interrupt processing must be enabled (EI, SYM.3 = 1)

Interrupt must be enabled at the interrupt's source (peripheral control register)

If all of the above conditions are met, the interrupt request is acknowledged at the end of the instruction cycle. The CPU then initiates an interrupt machine cycle that completes the following processing sequence:
1. Reset (clear to 0) the global interrupt enable bit in the SYM register (DI, SYM.3 = 0) to disable all subsequent interrupts.
2. Save the program counter and status flags to stack.
3. Branch to the interrupt vector to fetch the service routine's address.
4. Pass control to the interrupt service routine.

When the interrupt service routine is completed, an Interrupt Return instruction (IRET) occurs. The IRET restores the PC and status flags and sets SYM.3 to 1 (EI), allowing the CPU to process the next interrupt request.

5.8 Generating interrupt Vector Addresses

The interrupt vector area in the ROM contains the address of the interrupt service routine. Vectored interrupt processing follows this sequence:
1. Push the program counter's low-byte value to stack.
2. Push the program counter's high-byte value to stack.
3. Push the FLAGS register values to stack.
4. Fetch the service routine's high-byte address from the vector address 0000H.
5. Fetch the service routine's low-byte address from the vector address 0001H.
6. Branch to the service routine specified by the 16-bit vector address.

5.9 S3F94C8/S3F94C4 Interrupt Structure

The S3F94C8/S3F94C4 microcontroller has four peripheral interrupt sources:

PWM overflow

Timer 0 match

P0.0 external interrupt

P0.1 external interrupt
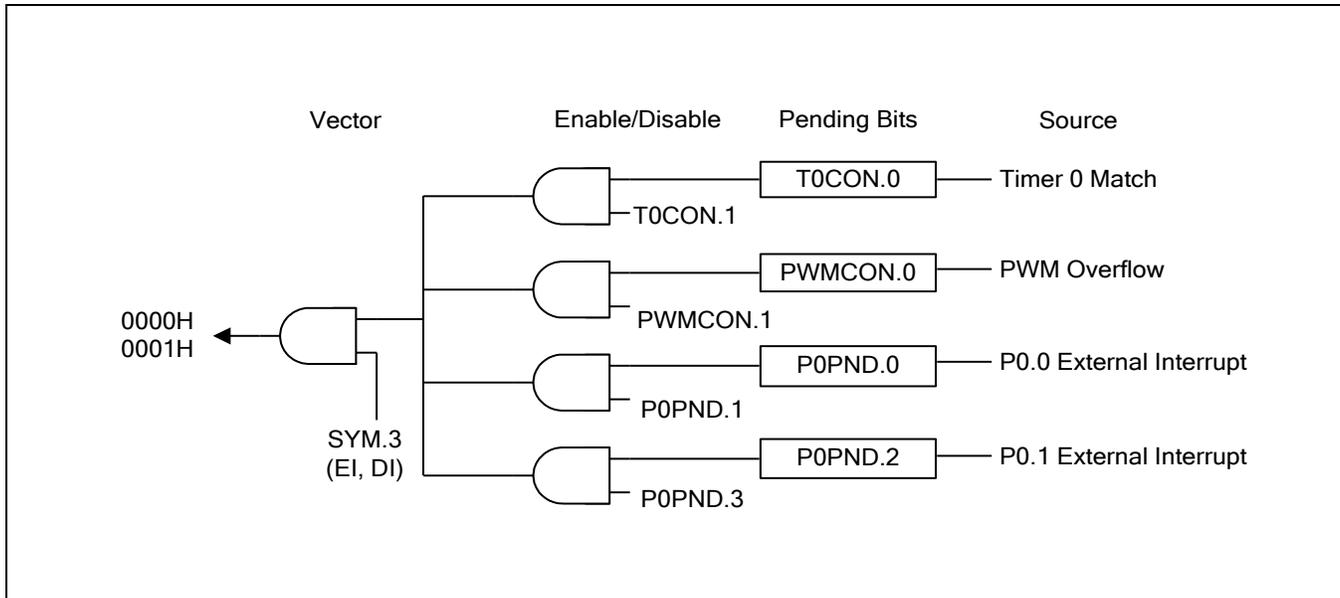
Figure 5-3. S3F94C8/S3F94C4 Interrupt Structure

5.10   Peripheral Interrupt Control Registers

For each interrupt source there is one or more corresponding peripheral control registers that let you control the interrupt generated by the related peripheral (see Table 5-1).

Table 5-1. Interrupt Source Control and Data Registers

Interrupt Source	Register(s)	Register Location(s)
P0.0 external interrupt P0.1 external interrupt	P0CONL P0PND	E7H E8H
Timer 0 match interrupt	T0CON T0DATA	D2H D1H
PWM overflow interrupt	PWMCON PWMDATA PWMDATA1	F3H F2H F0H

6 Instruction Set

6.1 Overview

The SAM88RCRI instruction set is designed to support the large register file. It includes a full complement of 8-bit arithmetic and logic operations. There are 41 instructions. No special I/O instructions are necessary because I/O control and data registers are mapped directly into the register file. Flexible instructions for bit addressing, rotate, and shift operations complete the powerful data manipulation capabilities of the SAM88RCRI instruction set.

6.2 Register Addressing

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

6.3 Addressing Modes

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

Table 6-1. Instruction Group Summary

Mnemonic	Operands	Instruction
	Load Instructions	
CLR	dst	Clear
LD	dst,src	Load
LDC	dst,src	Load program memory
LDE	dst,src	Load external data memory
LDCD	dst,src	Load program memory and decrement
LDED	dst,src	Load external data memory and decrement
LDCI	dst,src	Load program memory and increment
LDEI	dst,src	Load external data memory and increment
POP	dst	Pop from stack
PUSH	src	Push to stack
	Arithmetic Instructions	
ADC	dst,src	Add with carry
ADD	dst,src	Add
CP	dst,src	Compare
DEC	dst	Decrement
INC	dst	Increment
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

Table 6-1. Instruction Group Summary (Continued)

Mnemonic	Operands	Instruction
		Program Control Instructions
CALL	dst	Call procedure
IRET		Interrupt return
JP	cc,dst	Jump on condition code
JP	dst	Jump unconditional
JR	cc,dst	Jump relative on condition code
RET		Return
		Bit Manipulation Instructions
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
		CPU Control Instructions
CCF		Complement carry flag
DI		Disable interrupts
EI		Enable interrupts
IDLE		Enter Idle Mode
NOP		No operation
RCF		Reset carry flag
SCF		Set carry flag
STOP		Enter Stop Mode

6.4 Flags Register

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

FLAGS register can be set or reset by instructions as long as its outcome does not affect the flags, such as, Load instruction. Logical and Arithmetic instructions such as, AND, OR, XOR, ADD, and SUB can affect the Flags register. For example, the AND instruction updates the Zero, Sign and Overflow flags based on the outcome of the AND instruction. If the AND instruction uses the Flags register as the destination, then simultaneously, two write will occur to the Flags register producing an unpredictable result.
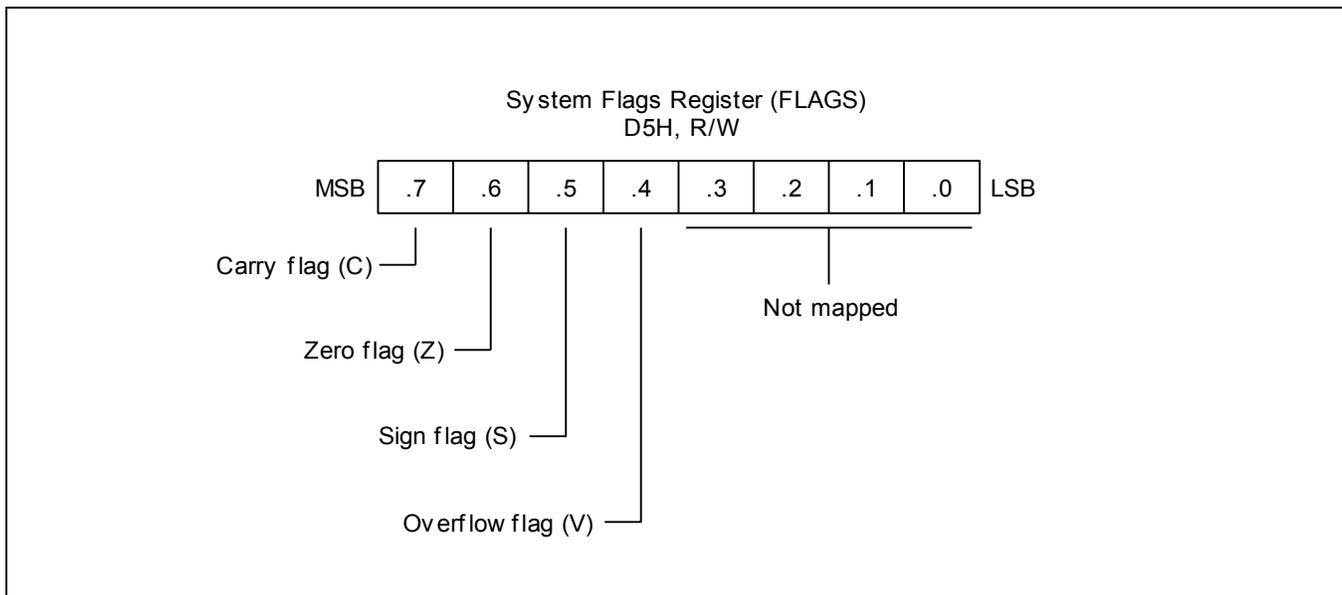
Figure 6-1. System Flags Register (FLAGS)

6.5 Flag Descriptions

6.5.1 Overflow Flag (FLAGS.4, V)

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.

6.5.2 Sign Flag (FLAGS.5, S)

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

6.5.3 Zero Flag (FLAGS.6, Z)

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

6.5.4 Carry Flag (FLAGS.7, C)

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

6.6 Instruction Set Notation

Table 6-2. Flag Notation Conventions

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

Table 6-3. Instruction Set Symbols

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

Table 6-4. Instruction Notation Conventions

Notation	Description	Actual Operand Range
cc	Condition code	See list of condition codes in Table 6-6.
r	Working register only	Rn (n = 0–15)
rr	Working register pair	RRp (p = 0, 2, 4, ..., 14)
R	Register or working register	reg or Rn (reg = 0–255, n = 0–15)
RR	Register pair or working register pair	reg or RRp (reg = 0–254, even number only, where p = 0, 2, ..., 14)
lr	Indirect working register only	@Rn (n = 0–15)
IR	Indirect register or indirect working register	@Rn or @reg (reg = 0–255, n = 0–15)
Irr	Indirect working register pair only	@RRp (p = 0, 2, ..., 14)
IRR	Indirect register pair or indirect working register pair	@RRp or @reg (reg = 0–254, even only, where p = 0, 2, ..., 14)
X	Indexed addressing mode	#reg[Rn] (reg = 0–255, n = 0–15)
XS	Indexed (short offset) addressing mode	#addr[RRp] (addr = range –128 to +127, where p = 0, 2, ..., 14)
xl	Indexed (long offset) addressing mode	#addr [RRp] (addr = range 0–8191, where p = 0, 2, ..., 14)
da	Direct addressing mode	addr (addr = range 0–8191)
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)

Table 6-5. Opcode Quick Reference

OPCODE MAP									
LOWER NIBBLE (HEX)									
	–	0	1	2	3	4	5	6	7
U	0	DEC R1	DEC IR1	ADD r1,r2	ADD r1,Ir2	ADD R2,R1	ADD IR2,R1	ADD R1,IM	
P	1	RLC R1	RLC IR1	ADC r1,r2	ADC r1,Ir2	ADC R2,R1	ADC IR2,R1	ADC R1,IM	
P	2	INC R1	INC IR1	SUB r1,r2	SUB r1,Ir2	SUB R2,R1	SUB IR2,R1	SUB R1,IM	
E	3	JP IRR1		SBC r1,r2	SBC r1,Ir2	SBC R2,R1	SBC IR2,R1	SBC R1,IM	
R	4			OR r1,r2	OR r1,Ir2	OR R2,R1	OR IR2,R1	OR R1,IM	
	5	POP R1	POP IR1	AND r1,r2	AND r1,Ir2	AND R2,R1	AND IR2,R1	AND R1,IM	
N	6	COM R1	COM IR1	TCM r1,r2	TCM r1,Ir2	TCM R2,R1	TCM IR2,R1	TCM R1,IM	
I	7	PUSH R2	PUSH IR2	TM r1,r2	TM r1,Ir2	TM R2,R1	TM IR2,R1	TM R1,IM	
B	8								LD r1, x, r2
B	9	RL R1	RL IR1						LD r2, x, r1
L	A			CP r1,r2	CP r1,Ir2	CP R2,R1	CP IR2,R1	CP R1,IM	LDC r1, Irr2, xL
E	B	CLR R1	CLR IR1	XOR r1,r2	XOR r1,Ir2	XOR R2,R1	XOR IR2,R1	XOR R1,IM	LDC r2, Irr2, xL
	C	RRC R1	RRC IR1		LDC r1,Irr2				LD r1, Ir2
H	D	SRA R1	SRA IR1		LDC r2,Irr1			LD IR1,IM	LD Ir1, r2
E	E	RR R1	RR IR1	LDCD r1,Irr2	LDCI r1,Irr2	LD R2,R1	LD R2,IR1	LD R1,IM	LDC r1, Irr2, xs
X	F					CALL IRR1	LD IR2,R1	CALL DA1	LDC r2, Irr1, xs

Table 6-5. Opcode Quick Reference (Continued)

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

6.7 Condition Codes

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

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

Table 6-6. Condition Codes

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

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

6.8 Instruction Descriptions

This section contains detailed information and programming examples for each instruction in the SAM87RI instruction set. Information is arranged in a consistent format for improved readability and for fast referencing. The following information is included in each instruction description:

Instruction name (mnemonic)

Full instruction name

Source/destination format of the instruction operand

Shorthand notation of the instruction's operation

Textual description of the instruction's effect

Specific flag settings affected by the instruction

Detailed description of the instruction's format, execution time, and addressing mode(s)

Programming example(s) explaining how to use the instruction

6.8.1 ADC — Add with Carry

ADC   **dst,src**

Operation: dst ← dst + src + c

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

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

S: Set if the result is 0; cleared otherwise.

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

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

Format:

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

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

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

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

6.8.2 ADD — Add

ADD   **dst,src**

Operation: dst ← dst + src

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

Flags:

C: Set if there is a carry from the most significant bit of the result; cleared otherwise.
Z: Set if the result is 0; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise.

Format:

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

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

ADD   R1,R2   →   R1 = 15H, R2 = 03H

ADD   R1,@R2   →   R1 = 1CH, R2 = 03H

ADD   01H,02H   →   Register 01H = 24H, register 02H = 03H

ADD   01H,@02H   →   Register 01H = 2BH, register 02H = 03H

ADD   01H,#25H   →   Register 01H = 46H

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

6.8.3 AND — Logical AND

AND   **dst,src**

Operation: dst ← dst AND src

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

Flags:

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

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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: Given: R1 = 12H, R2 = 03H, register 01H = 21H, register 02H = 03H, register 03H = 0AH:

AND   R1,R2   →   R1 = 02H, R2 = 03H

AND   R1,@R2   →   R1 = 02H, R2 =   03H

AND   01H,02H   →   Register 01H = 01H, register 02H = 03H

AND   01H,@02H   →   Register 01H = 00H, register 02H = 03H

AND   01H,#25H   →   Register 01H = 21H

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

6.8.4 CALL — Call Procedure

CALL   **dst**

Operation: SP   ←   SP – 1

@SP   ←   PCL
SP   ←   SP –1
@SP   ←   PCH
PC   ←   dst

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

Flags: No flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc	dst	3	14	F6	DA
opc	dst	2	12	F4	IRR

Examples: Given: R0 = 15H, R1   = 21H, PC = 1A47H, and SP = 0B2H:

CALL   1521H   →   SP = 0B0H
(Memory locations 00H = 1AH, 01H = 4AH, where 4AH
is the address that follows the instruction.)

CALL   @RR0   →   SP = 0B0H (00H = 1AH, 01H = 49H)

In the first example, if the program counter value is 1A47H and the stack pointer contains the value 0B2H, the statement "CALL   1521H" pushes the current PC value onto the top of the stack. The stack pointer now points to memory location 00H. The PC is then loaded with the value 1521H, the address of the first instruction in the program sequence to be executed.

If the contents of the program counter and stack pointer are the same as in the first example, the statement "CALL   @RR0" produces the same result except that the 49H is stored in stack location 01H (because the two-byte instruction format was used). The PC is then loaded with the value 1521H, the address of the first instruction in the program sequence to be executed.

6.8.5 CCF — Complement Carry Flag

CCF

Operation: C ← NOT C

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

Flags: **C:** Complemented.

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	EF

Example: Given: The carry flag = 0:

CCF

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

6.8.6 CLR — Clear

CLR   **dst**

Operation: dst ← 0

The destination location is cleared to 0.

Flags: No flags are affected.

Format:

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

Examples: Given: Register 00H = 4FH, register 01H = 02H, and register 02H = 5EH:

CLR   00H   →   Register 00H = 00H

CLR   @01H   →   Register 01H = 02H, register 02H = 00H

In Register (R) addressing mode, the statement "CLR   00H" clears the destination register 00H value to 00H. In the second example, the statement "CLR   @01H" uses Indirect Register (IR) addressing mode to clear the 02H register value to 00H.

6.8.7 COM — Complement

COM   **dst**

Operation: dst ← NOT   dst

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

Flags: **C:** Unaffected.

    Z: Set if the result is 0; cleared otherwise.
    S: Set if the result bit 7 is set; cleared otherwise.
    V: Always reset to 0.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc	dst	2	4	60	R
			4	61	IR

Examples: Given: R1 = 07H and register 07H = 0F1H:

COM   R1   →   R1 = 0F8H

COM   @R1   →   R1 = 07H, register 07H = 0EH

In the first example, destination working register R1 contains the value 07H (00000111B). The statement "COM   R1" complements all the bits in R1: all logic ones are changed to logic zeros, and vice-versa, leaving the value 0F8H (11111000B).

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

6.8.8 CP — Compare

CP   **dst,src**

Operation: dst – src

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

Flags: **C:** Set if a "borrow" occurred (src > dst); cleared otherwise.

Z: Set if the result is 0; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the operands were of opposite signs and the sign of the result is of the same as the sign of the source operand; cleared otherwise.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode 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

Examples: 1. Given: R1 = 02H and   R2 = 03H:

        CP   R1,R2   → Set the C and S flags

Destination working register R1 contains the value 02H and source register R2 contains the value 03H. The statement "CP   R1,R2" subtracts the R2 value (source/subtrahend) from the R1 value (destination/minuend). Because a "borrow" occurs and the difference is negative, C and S are 1.

2. Given: R1 = 05H and R2 = 0AH:

        CP   R1,R2
        JP   UGE,SKIP
        INC   R1
SKIP   LD   R3,R1

In this example, destination working register R1 contains the value 05H which is less than the contents of the source working register R2 (0AH). The statement "CP   R1,R2" generates C = 1 and the JP instruction does not jump to the SKIP location. After the statement "LD   R3,R1" executes, the value 06H remains in working register R3.

6.8.9 DEC — Decrement

DEC   **dst**

Operation: dst ← dst – 1

The contents of the destination operand are decremented by one.

Flags: **C:** Unaffected.

Z: Set if the result is 0; cleared otherwise.
S: Set if result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, dst value is –128 (80H) and result value is
+127 (7FH); cleared otherwise.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc	dst	2	4	00	R
			4	01	IR

Examples: Given: R1 = 03H and register 03H = 10H:

DEC   R1   →   R1 = 02H

DEC   @R1   →   Register 03H = 0FH

In the first example, if working register R1 contains the value 03H, the statement "DEC   R1"
decrements the hexadecimal value by one, leaving the value 02H. In the second example, the
statement "DEC @R1" decrements the value 10H contained in the destination register 03H by one,
leaving the value 0FH.

6.8.10 DI — Disable Interrupts

DI

Operation: SYM (3)   ← 0

Bit zero of the system mode register, SYM.3, is cleared to 0, globally disabling all interrupt processing. Interrupt requests will continue to set their respective interrupt pending bits, but the CPU will not service them while interrupt processing is disabled.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	8F

Example: Given: SYM = 08H:

DI

If the value of the SYM register is 08H, the statement "DI" leaves the new value 00H in the register and clears SYM.3 to 0, disabling interrupt processing.

6.8.11 EI — Enable Interrupts

EI

Operation:	SYM (3) $\leftarrow$ 1

An EI instruction sets bit 2 of the system mode register, SYM.3 to 1. This allows interrupts to be serviced as they occur. If an interrupt's pending bit was set while interrupt processing was disabled (by executing a DI instruction), it will be serviced when you execute the EI instruction.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	9F

Example: Given: SYM = 00H:

EI

If the SYM register contains the value 00H, that is, if interrupts are currently disabled, the statement "EI" sets the SYM register to 08H, enabling all interrupts. (SYM.3 is the enable bit for global interrupt processing.)

6.8.12 IDLE — Idle Operation

IDLE

Operation:

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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	1	4	6F	–	–

Example: The instruction

IDLE
NOP
NOP
NOP

stops the CPU clock but not the system clock.

6.8.13 INC — Increment

INC	**dst**

Operation: dst ← dst + 1

The contents of the destination operand are incremented by one.

Flags: **C:** Unaffected.

Z: Set if the result is 0; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is dst value is +127 (7FH) and result is –128 (80H); cleared otherwise.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
dst | opc	1	4	rE	r
			r = 0 to F	
opc | dst	2	4	20	R
		4	21	IR

Examples: Given: R0 = 1BH, register 00H = 0CH, and register 1BH = 0FH:

INC	R0	→	R0 = 1CH
INC	00H	→	Register 00H = 0DH
INC	@R0	→	R0 = 1BH, register 01H = 10H

In the first example, if destination working register R0 contains the value 1BH, the statement "INC R0" leaves the value 1CH in that same register.

The next example shows the effect an INC instruction has on register 00H, assuming that it contains the value 0CH.

In the third example, INC is used in Indirect Register (IR) addressing mode to increment the value of register 1BH from 0FH to 10H.

6.8.14 IRET — Interrupt Return

IRET   IRET

Operation: FLAGS ← @SP
                   SP   ← SP + 1
                   PC   ← @SP
                   SP   ← SP + 2
                   SYM(2) ← 1

This instruction is used at the end of an interrupt service routine. It restores the flag register and the program counter. It also reenables global interrupts.

Flags: All flags are restored to their original settings (that is, the settings before the interrupt occurred).

Format:

IRET (Normal)	Bytes	Cycles	Opcode (Hex)
opc	1	10	BF
		12	

6.8.15 JP — Jump

JP   cc,dst   **(Conditional)**

JP   dst   **(Unconditional)**

Operation: If   cc   is true, PC   ← dst

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

Flags: No flags are affected.

Format: [1]

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
(2)				
cc | opc | dst	3	8	ccD	DA
			cc = 0 to F	
opc | dst	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: Given: The carry flag (C) = 1, register 00 = 01H, and register 01 = 20H:

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

JP   @00H   →   PC = 0120H

The first example shows a conditional JP. Assuming that the carry flag is set to 1, the statement "JP   C,LABEL_W" replaces the contents of the PC with the value 1000H and transfers control to that location. Had the carry flag not been set, control would then have passed to the statement immediately following the JP instruction.

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

6.8.16 JR — Jump Relative

JR   **cc,dst**

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

If the condition specified by   the condition code (cc) is true, the relative address is added to the program counter and control passes to the statement whose address is now in the program counter; otherwise, the instruction following the JR instruction is executed (See list of condition codes).

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

Flags: No flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst
(note)					
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: Given: The carry flag = 1 and LABEL_X = 1FF7H:

JR   C,LABEL_X →   PC = 1FF7H

If the carry flag is set (that is, if the condition code is true), the statement "JR   C,LABEL_X" will pass control to the statement whose address is now in the PC. Otherwise, the program instruction following the JR would be executed.

6.8.17 LD — Load

LD	dst,src
Operation:	dst ← src

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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
dst | opc src	2	4	rC	r	IM
		4	r8	r	R
src | opc dst	2	4	r9	R	r
		r = 0 to F			
opc dst | src	2	4	C7	r	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

6.8.18 LD — Load

LD	**(Continued)**

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

LD	R0,#10H	$\rightarrow$	R0 = 10H
LD	R0,01H	$\rightarrow$	R0 = 20H, register 01H = 20H
LD	01H,R0	$\rightarrow$	Register 01H = 01H, R0 = 01H
LD	R1,@R0	$\rightarrow$	R1 = 20H, R0 = 01H
LD	@R0,R1	$\rightarrow$	R0 = 01H, R1 = 0AH, register 01H = 0AH
LD	00H,01H	$\rightarrow$	Register 00H = 20H, register 01H = 20H
LD	02H,@00H	$\rightarrow$	Register 02H = 20H, register 00H = 01H
LD	00H,#0AH	$\rightarrow$	Register 00H = 0AH
LD	@00H,#10H	$\rightarrow$	Register 00H = 01H, register 01H = 10H
LD	@00H,02H	$\rightarrow$	Register 00H = 01H, register 01H = 02, register 02H = 02H
LD	R0,#LOOP[R1]	$\rightarrow$	R0 = 0FFH, R1 = 0AH
LD	#LOOP[R0],R1	$\rightarrow$	Register 31H = 0AH, R0 = 01H, R1 = 0AH

6.8.19 LDC/LDE — Load Memory

LDC/LDE   **dst,src**

Operation: dst ← src

This instruction loads a byte from program or data memory into a working register or vice-versa. The source values are unaffected. LDC refers to program memory and LDE to data memory. The assembler makes "Irr" or "rr" values an even number for program memory and odd an odd number for data memory.

Flags: No flags are affected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
1.	opc	dst | src	2	10	C3	r	Irr
2.	opc	src | dst	2	10	D3	Irr	r
3.	opc	dst | src	XS	3	12	E7	r	XS [rr]
4.	opc	src | dst	XS	3	12	F7	XS [rr]	r
5.	opc	dst | src	XL_L	XL_H	4	14	A7	r	XL [rr]
6.	opc	src | dst	XL_L	XL_H	4	14	B7	XL [rr]	r
7.	opc	dst | 0000	DA_L	DA_H	4	14	A7	r	DA
8.	opc	src | 0000	DA_L	DA_H	4	14	B7	DA	r
9.	opc	dst | 0001	DA_L	DA_H	4	14	A7	r	DA

10.	opc	src | 0001	DA~L~	DA~H~	4	14	B7	DA	r

NOTES:

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.

6.8.20 LDC/LDE — Load Memory

LDC/LDE   **(Continued)**

Examples: Given: R0 = 11H, R1 = 34H, R2 = 01H, R3 = 04H, R4 = 00H, R5 = 60H; Program memory locations 0061 = AAH, 0103H = 4FH, 0104H = 1A, 0105H = 6DH, and 1104H = 88H. External data memory locations 0061H = BBH, 0103H = 5FH, 0104H = 2AH, 0105H = 7DH, and 1104H = 98H:

LDC	R0,@RR2	;	R0 ← contents of program memory location 0104H
		;	R0 = 1AH, R2 = 01H, R3 = 04H
LDE	R0,@RR2	;	R0 ← contents of external data memory location 0104H
		;	R0 = 2AH, R2 = 01H, R3 = 04H
LDC (note)	@RR2,R0	;	11H (contents of R0) is loaded into program memory
		;	location 0104H (RR2),
		;	working registers R0, R2, R3 → no change
LDE	@RR2,R0	;	11H (contents of R0) is loaded into external data memory
		;	location 0104H (RR2),
		;	working registers R0, R2, R3 → no change
LDC	R0,#01H[RR4]	;	R0 ← contents of program memory location 0061H
		;	(01H + RR4),
		;	R0 = AAH, R2 = 00H, R3 = 60H
LDE	R0,#01H[RR4]	;	R0 ← contents of external data memory location 0061H
		;	(01H + RR4), R0 = BBH, R4 = 00H, R5 = 60H
LDC (note) memory location	#01H[RR4],R0 ;		11H (contents of R0) is loaded into program
		;	0061H (01H + 0060H)
LDE	#01H[RR4],R0	;	11H (contents of R0) is loaded into external data memory
		;	location 0061H (01H + 0060H)
LDC	R0,#1000H[RR2]	;	R0 ← contents of program memory location 1104H
		;	(1000H + 0104H), R0 = 88H, R2 = 01H, R3 = 04H
LDE	R0,#1000H[RR2]	;	R0 ← contents of external data memory location 1104H
		;	(1000H + 0104H), R0 = 98H, R2 = 01H, R3 = 04H
LDC	R0,1104H	;	R0 ← contents of program memory location 1104H, R0 = 88H
LDE	R0,1104H	;	R0 ← contents of external data memory location 1104H,
		;	R0 = 98H
LDC (note)	1105H,R0	;	11H (contents of R0) is loaded into program memory location
		;	1105H, (1105H) ← 11H
LDE	1105H,R0	;	11H (contents of R0) is loaded into external data memory
		;	location 1105H, (1105H) ← 11H

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

6.8.21 LDCD/LDED — Load Memory and Decrement

LDCD/LDED   **dst,src**

Operation: dst ← src

rr ← rr − 1

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

LDCD references program memory and LDED references external data memory. The assembler makes "Irr" an even number for program memory and an odd number for data memory.

Flags: No flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	dst | src	2	10	E2	r	Irr

Examples: Given: R6 = 10H, R7 = 33H, R8 = 12H, program memory location 1033H = 0CDH, and external data memory location 1033H = 0DDH:

LDCD   R8,@RR6   ; 0CDH (contents of program memory location 1033H) is loaded
                       ; into R8 and RR6 is decremented by one
                       ; R8 = 0CDH, R6 = 10H, R7 = 32H (RR6 ← RR6 − 1)

LDED   R8,@RR6   ; 0DDH (contents of data memory location 1033H) is loaded
                       ; into R8 and RR6 is decremented by one (RR6 ← RR6 − 1)
                       ; R8 = 0DDH, R6 = 10H, R7 = 32H

6.8.22 LDCI/LDEI — Load Memory and Increment

LDCI/LDEI   **dst,src**

Operation: dst ← src

rr ← rr + 1

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

LDCI refers to program memory and LDEI refers to external data memory. The assembler makes "Irr" even for program memory and odd for data memory.

Flags: No flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	dst | src	2	10	E3	r	Irr

Examples: Given: R6 = 10H, R7 = 33H, R8 = 12H, program memory locations 1033H = 0CDH and 1034H = 0C5H; external data memory locations 1033H = 0DDH and 1034H = 0D5H:

LDCI   R8,@RR6   ; 0CDH (contents of program memory location 1033H) is loaded
                           ; into R8 and RR6 is incremented by one (RR6 ← RR6 + 1)
                           ; R8 = 0CDH, R6 = 10H, R7 = 34H

LDEI   R8,@RR6   ; 0DDH (contents of data memory location 1033H) is loaded
                           ; into R8 and RR6 is incremented by one (RR6 ← RR6 + 1)
                           ; R8 = 0DDH, R6 = 10H, R7 = 34H

6.8.23 NOP — No Operation

NOP

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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	FF

Example: When the instruction

NOP

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

6.8.24 OR — Logical OR

OR   **dst,src**

Operation: dst ← dst OR src

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

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

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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: Given: R0 = 15H, R1 = 2AH, R2 = 01H, register 00H = 08H, register 01H = 37H, and register 08H = 8AH:

OR   R0,R1   →   R0 = 3FH, R1 = 2AH

OR   R0,@R2   →   R0 = 37H, R2 = 01H, register 01H = 37H

OR   00H,01H   →   Register 00H = 3FH, register 01H = 37H

OR   01H,@00H   →   Register 00H = 08H, register 01H = 0BFH

OR   00H,#02H   →   Register 00H = 0AH

In the first example, if working register R0 contains the value 15H and register R1 the value 2AH, the statement "OR   R0,R1" logical-ORs the R0 and R1 register contents and stores the result (3FH) in destination register R0.

The other examples show the use of the logical OR instruction with the various addressing modes and formats.

6.8.25 POP — Pop From Stack

POP   **dst**

Operation: dst ← @SP

SP ← SP + 1

The contents of the location addressed by the stack pointer are loaded into the destination. The stack pointer is then incremented by one.

Flags: No flags affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc	dst	2	8	50	R
			8	51	IR

Examples: Given: Register 00H = 01H, register 01H = 1BH, SP (0D9H) = 0BBH, and stack register 0BBH = 55H:

POP   00H   →   Register 00H = 55H, SP = 0BCH

POP   @00H   →   Register 00H = 01H, register 01H = 55H, SP = 0BCH

In the first example, general register 00H contains the value 01H. The statement "POP   00H" loads the contents of location 0BBH (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 0BCH.

6.8.26 PUSH — Push To Stack

PUSH   **src**

Operation: SP ← SP − 1

@SP ← src

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

Flags: No flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
opc	src	2	8	70	R
			8	71	IR

Examples: Given: Register 40H = 4FH, register 4FH = 0AAH, SP = 0C0H:

PUSH   40H   →   Register 40H = 4FH, stack register 0BFH = 4FH,
                                            SP = 0BFH

PUSH   @40H   →   Register 40H = 4FH, register 4FH = 0AAH, stack register
                                            0BFH = 0AAH, SP = 0BFH

In the first example, if the stack pointer contains the value 0C0H, and general register 40H the value 4FH, the statement "PUSH   40H" decrements the stack pointer from 0C0 to 0BFH. It then loads the contents of register 40H into location 0BFH. Register 0BFH then contains the value 4FH and SP points to location 0BFH.

6.8.27 RCF — Reset Carry Flag

RCF   RCF

Operation: C ← 0

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

Flags: C: Cleared to 0.

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	CF

Example: Given: C = 1 or 0:

The instruction RCF clears the carry flag (C) to logic zero.

6.8.28 RET — Return

RET

Operation: PC ← @SP

SP ← SP + 2

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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	8	AF
		10	

Example: Given: SP = 0BCH, (SP) = 101AH, and PC = 1234:

RET → PC = 101AH, SP = 0BEH

The statement "RET" pops the contents of stack pointer location 0BCH (10H) into the high byte of the program counter. The stack pointer then pops the value in location 0BDH (1AH) into the PC's low byte and the instruction at location 101AH is executed. The stack pointer now points to memory location 0BEH.

6.8.29 RL — Rotate Left

RL   **dst**

Operation: C   ← dst (7)

dst (0)   ← dst (7)

dst (n   +   1)   ← dst (n),   n = 0–6

The contents of the destination operand are rotated left one bit position. The initial value of bit 7 is moved to the bit zero (LSB) position and also replaces the carry flag.
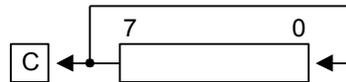
Flags:
C: Set if the bit rotated from the most significant bit position (bit 7) was 1.
Z: Set if the result is 0; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
opc	dst	2	4	90	R
			4	91	IR

Examples: Given: Register 00H = 0AAH, register 01H = 02H and register 02H = 17H:

RL   00H   →   Register 00H = 55H, C = 1

RL   @01H   →   Register 01H = 02H, register 02H = 2EH, C = 0

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

6.8.30 RLC — Rotate Left Through Carry

RLC   **dst**

Operation: dst (0) ← C

C ← dst (7)

dst (n + 1) ← dst (n), n = 0–6

The contents of the destination operand with the carry flag are rotated left one bit position. The initial value of bit 7 replaces the carry flag (C); the initial value of the carry flag replaces bit zero.
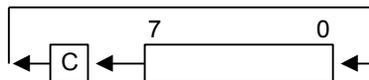
Flags:
   C: Set if the bit rotated from the most significant bit position (bit 7) was 1.
   Z: Set if the result is 0; cleared otherwise.
   S: Set if the result bit 7 is set; cleared otherwise.
   V: Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc	dst	2	4	10	R
			4	11	IR

Examples: Given: Register 00H = 0AAH, register 01H = 02H, and register 02H = 17H, C = 0:

RLC   00H   →   Register 00H = 54H, C = 1

RLC   @01H   →   Register 01H = 02H, register 02H = 2EH, C   = 0

In the first example, if general register 00H has the value 0AAH (10101010B), the statement "RLC 00H" rotates 0AAH one bit position to the left. The initial value of bit 7 sets the carry flag and the initial value of the C flag replaces bit zero of register 00H, leaving the value 55H (01010101B). The MSB of register 00H resets the carry flag to 1 and sets the overflow flag.

6.8.31 RR — Rotate Right

RR   **dst**

Operation: C ← dst (0)

dst (7) ← dst (0)

dst (n) ← dst (n + 1), n = 0–6

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

Flags:
C: Set if the bit rotated from the least significant bit position (bit zero) was 1.
Z: Set if the result is 0; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc	dst	2	4	E0	R
			4	E1	IR

Examples: Given: Register 00H = 31H, register 01H = 02H, and register 02H = 17H:

RR   00H   →   Register 00H = 98H, C = 1

RR   @01H   →   Register 01H = 02H, register 02H = 8BH, C = 1

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

6.8.32 RRC — Rotate Right Through Carry

RRC   **dst**

Operation: dst (7) ← C

C ← dst (0)

dst (n) ← dst (n + 1), n = 0–6

The contents of the destination operand and the carry flag are rotated right one bit position. The initial value of bit zero (LSB) replaces the carry flag; the initial value of the carry flag replaces bit 7 (MSB).
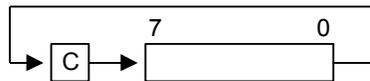
Flags:
- **C:** Set if the bit rotated from the least significant bit position (bit zero) was 1.
- **Z:** Set if the result is 0 cleared otherwise.
- **S:** Set if the result bit 7 is set; cleared otherwise.
- **V:** Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	C0	R
		4	C1	IR

Examples: Given: Register 00H = 55H, register 01H = 02H, register 02H = 17H, and C = 0:

RRC   00H   →   Register 00H = 2AH, C = 1

RRC   @01H   →   Register 01H = 02H, register 02H = 0BH, C = 1

In the first example, if general register 00H contains the value 55H (01010101B), the statement "RRC   00H" rotates this value one bit position to the right. The initial value of bit zero (1) replaces the carry flag and the initial value of the C flag (1) replaces bit 7. This   leaves the new value 2AH (00101010B) in destination register 00H. The sign flag and overflow flag are both cleared to 0.

6.8.33 SBC — Subtract With Carry

SBC   **dst,src**

Operation: dst ← dst − src − c

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

Flags:

C: Set if a borrow occurred (src > dst); cleared otherwise.
Z: Set if the result is 0; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the operands were of opposite sign and the sign of the result is the same as the sign of the source; cleared otherwise.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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: Given: R1 = 10H, R2 = 03H, C = 1, register 01H = 20H, register 02H = 03H, and register 03H = 0AH:

SBC   R1,R2   →   R1 = 0CH, R2 = 03H

SBC   R1,@R2   →   R1 = 05H, R2 = 03H, register 03H = 0AH

SBC   01H,02H   →   Register 01H = 1CH, register 02H = 03H

SBC   01H,@02H   →   Register 01H = 15H,register 02H = 03H, register 03H = 0AH

SBC   01H,#8AH   →   Register 01H = 95H; C, S, and V = 1

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

6.8.34 SCF — Set Carry Flag

SCF

Operation: C ← 1

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

Flags:

   C: Set to 1.
            No other flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)
opc		1	4	DF

Example: The statement

SCF

sets the carry flag to logic one.

6.8.35 SRA — Shift Right Arithmetic

SRA   **dst**

Operation: dst (7)   ← dst (7)

C   ← dst (0)

dst (n)   ← dst (n   +   1), n = 0–6

An arithmetic shift-right of one bit position is performed on the destination operand. Bit zero (the LSB) replaces the carry flag. The value of bit 7 (the sign bit) is unchanged and is shifted into bit position 6.
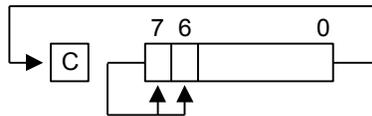
Flags:
   C: Set if the bit shifted from the LSB position (bit zero) was 1.
   Z: Set if the result is 0; cleared otherwise.
   S: Set if the result is negative; cleared otherwise.
   V: Always cleared to 0.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
opc	dst	2	4	D0	R
			4	D1	IR

Examples: Given: Register 00H = 9AH, register 02H = 03H, register 03H = 0BCH, and C = 1:

SRA   00H   →   Register 00H = 0CD, C = 0

SRA   @02H   →   Register 02H = 03H, register 03H = 0DEH, C = 0

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

6.8.36 STOP — Stop Operation

STOP

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

Flags: No flags are affected.

Format:

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

Example: The statement

LD   STOPCON, #0A5H

STOP

NOP

NOP

NOP

halts all microcontroller operations. When the STOPCON Register is not #0A5H value, if you use STOP instruction, PC is changed to reset address.

6.8.37 SUB — Subtract

SUB **dst,src**

Operation: dst ← dst – src

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

Flags:
C: Set if a "borrow" occurred; cleared otherwise.
Z: Set if the result is 0; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the operands were of opposite signs and the sign of the result is of the same as the sign of the source operand; cleared otherwise.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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: Given: R1 = 12H, R2 = 03H, register 01H = 21H, register 02H = 03H, register 03H = 0AH:

SUB	R1,R2	→	R1 = 0FH, R2 = 03H
SUB	R1,@R2	→	R1 = 08H, R2 = 03H
SUB	01H,02H	→	Register 01H = 1EH, register 02H = 03H
SUB	01H,@02H	→	Register 01H = 17H, register 02H = 03H
SUB	01H,#90H	→	Register 01H = 91H; C, S, and V = 1
SUB	01H,#65H	→	Register 01H = 0BCH; C and S = 1, V = 0

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

6.8.38 TCM — Test Complement Under Mask

TCM   **dst,src**

Operation: (NOT dst)   AND   src

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

Flags:
   C: Unaffected.
   Z: Set if the result is 0; cleared otherwise.
   S: Set if the result bit 7 is set; cleared otherwise.
   V: Always cleared to 0.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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: Given: R0 = 0C7H, R1 = 02H, R2 = 12H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

TCM	R0,R1	→	R0 = 0C7H, R1 = 02H, Z = 1
TCM	R0,@R1	→	R0 = 0C7H, R1 = 02H, register 02H = 23H, Z = 0
TCM	00H,01H	→	Register 00H = 2BH, register 01H = 02H, Z = 1
TCM	00H,@01H	→	Register 00H = 2BH, register 01H = 02H, register 02H = 23H, Z = 1
TCM	00H,#34	→	Register 00H = 2BH, Z = 0

In the first example, if working register R0 contains the value 0C7H (11000111B) and register R1 the value 02H (00000010B), the statement "TCM   R0,R1" tests bit one in the destination register for a 1 value. Because the mask value corresponds to the test bit, the Z flag is set to logic one and can be tested to determine the result of the TCM operation.

6.8.39 TM — Test Under Mask

TM   **dst,src**

Operation: dst   AND   src

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

Flags:
    C: Unaffected.
    Z: Set if the result is 0; cleared otherwise.
    S: Set if the result bit 7 is set; cleared otherwise.
    V: Always reset to 0.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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: Given: R0 = 0C7H, R1 = 02H, R2 = 18H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

TM	R0,R1	→	R0 = 0C7H, R1 = 02H, Z = 0
TM	R0,@R1	→	R0 = 0C7H, R1 = 02H, register 02H = 23H, Z = 0
TM	00H,01H	→	Register 00H = 2BH, register 01H = 02H, Z = 0
TM	00H,@01H	→	Register 00H = 2BH, register 01H = 02H, register 02H = 23H, Z = 0
TM	00H,#54H	→	Register 00H = 2BH, Z = 1

In the first example, if working register R0 contains the value 0C7H (11000111B) and register R1 the value 02H (00000010B), the statement "TM   R0,R1" tests bit one in the destination register for a 0 value. Because the mask value does not match the test bit, the Z flag is cleared to logic zero and can be tested to determine the result of the TM operation.

6.8.40 XOR — Logical Exclusive OR

XOR   **dst,src**

Operation: dst ← dst XOR src

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

Flags:
   C: Unaffected.
   Z: Set if the result is 0; cleared otherwise.
   S: Set if the result bit 7 is set; cleared otherwise.
   V: Always reset to 0.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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: Given: R0 = 0C7H, R1 = 02H, R2 = 18H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

XOR	R0,R1	→	R0 = 0C5H, R1 = 02H
XOR	R0,@R1	→	R0 = 0E4H, R1 = 02H, register 02H = 23H
XOR	00H,01H	→	Register 00H = 29H, register 01H = 02H
XOR	00H,@01H	→	Register 00H = 08H, register 01H = 02H, register 02H = 23H
XOR	00H,#54H	→	Register 00H = 7FH

In the first example, if working register R0 contains the value 0C7H and if register R1 contains the value 02H, the statement "XOR R0,R1" logically exclusive-ORs the R1 value with the R0 value and stores the result (0C5H) in the destination register R0.

7 Clock Circuit

7.1 Overview

By using the Smart Option (3FH.1 – .0 in ROM), users can select an internal RC oscillator, an external RC oscillator, or an external oscillator. In using internal oscillator, XIN (P1.0), XOUT (P1.1) can be used by normal I/O pins. An internal RC oscillator source provides a typical 3.2 MHz or 0.5 MHz (in VDD = 5 V) depending on Smart Option.

An external RC oscillation source provides a typical 4MHz clock for S3F94C8/S3F94C4. An internal capacitor supports the RC oscillator circuit. An external crystal or ceramic oscillation source provides a maximum 10 MHz clock. The XIN and XOUT pins connect the oscillation source to the on-chip clock circuit. Simplified external RC oscillator and crystal/ceramic oscillator circuits are shown in Figures 7-1 and 7-2. When you use external oscillator, P1.0, P1.1 must be set to output port to prevent current consumption.
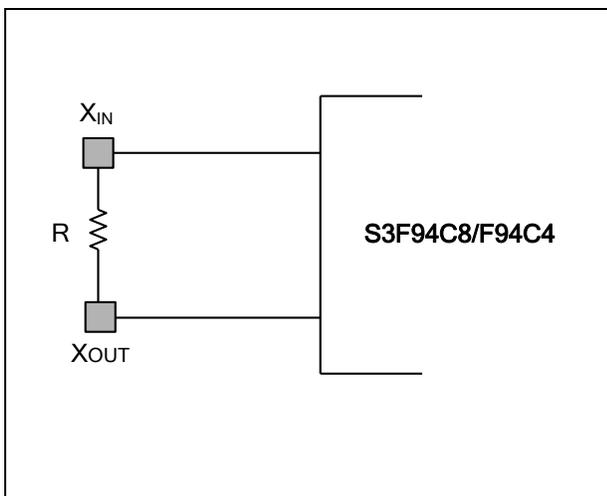
**Figure 7-1.   Main Oscillator Circuit
(RC Oscillator with Internal Capacitor)**
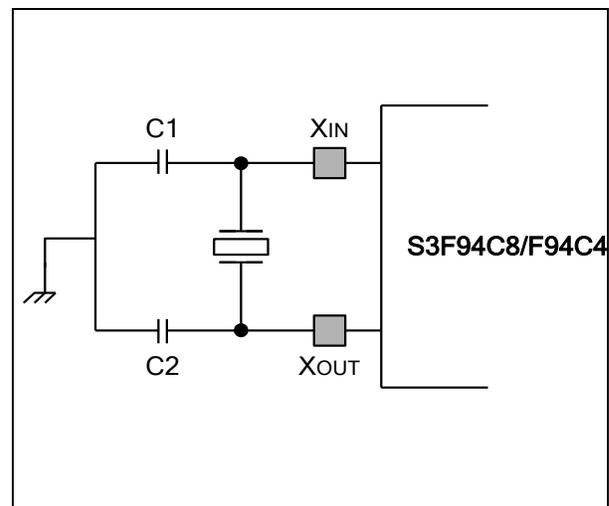
**Figure 7-2.   Main Oscillator Circuit
(Crystal/Ceramic Oscillator)**

7.2 Main Oscillator Logic

To increase processing speed and to reduce clock noise, non-divided logic is implemented for the main oscillator circuit. For this reason, very high-resolution waveforms (square signal edges) must be generated in order for the CPU to efficiently process logic operations.

7.2.1 Clock Status During Power-Down Modes

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

In Stop Mode, the main oscillator "freezes", halting the CPU and peripherals. The contents of the register file and current system register values are retained. Stop Mode is released, and the oscillator started, by a reset operation or by an external interrupt with RC-delay noise filter (for S3F94C8/S3F94C4, INT0–INT1).

In Idle Mode, the internal clock signal is gated off to the CPU, but not to interrupt control and the timer. The current CPU status is preserved, including stack pointer, program counter, and flags. Data in the register file is retained. Idle Mode is released by a reset or by an interrupt (external or internally-generated).

7.2.2 System Clock Control Register

The System Clock Control Register, CLKCON, is located in location D4H. It is read/write addressable and has the following functions:

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

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

After a reset, the external interrupt oscillator wake-up function is enabled, and the f_{OSC}/16 (the slowest clock speed) is selected as the CPU clock. If necessary, you can then increase the CPU clock speed to f_{OSC}, f_{OSC}/2 or f_{OSC}/8.
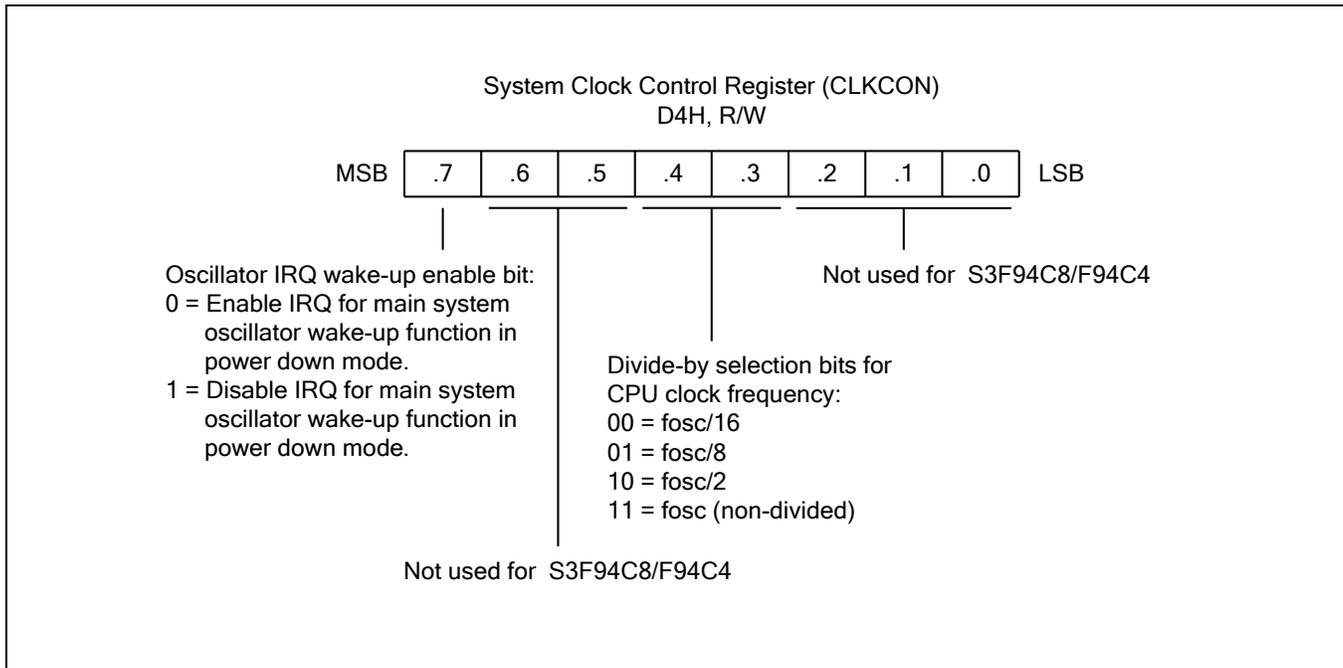
Figure 7-3. System Clock Control Register (CLKCON)
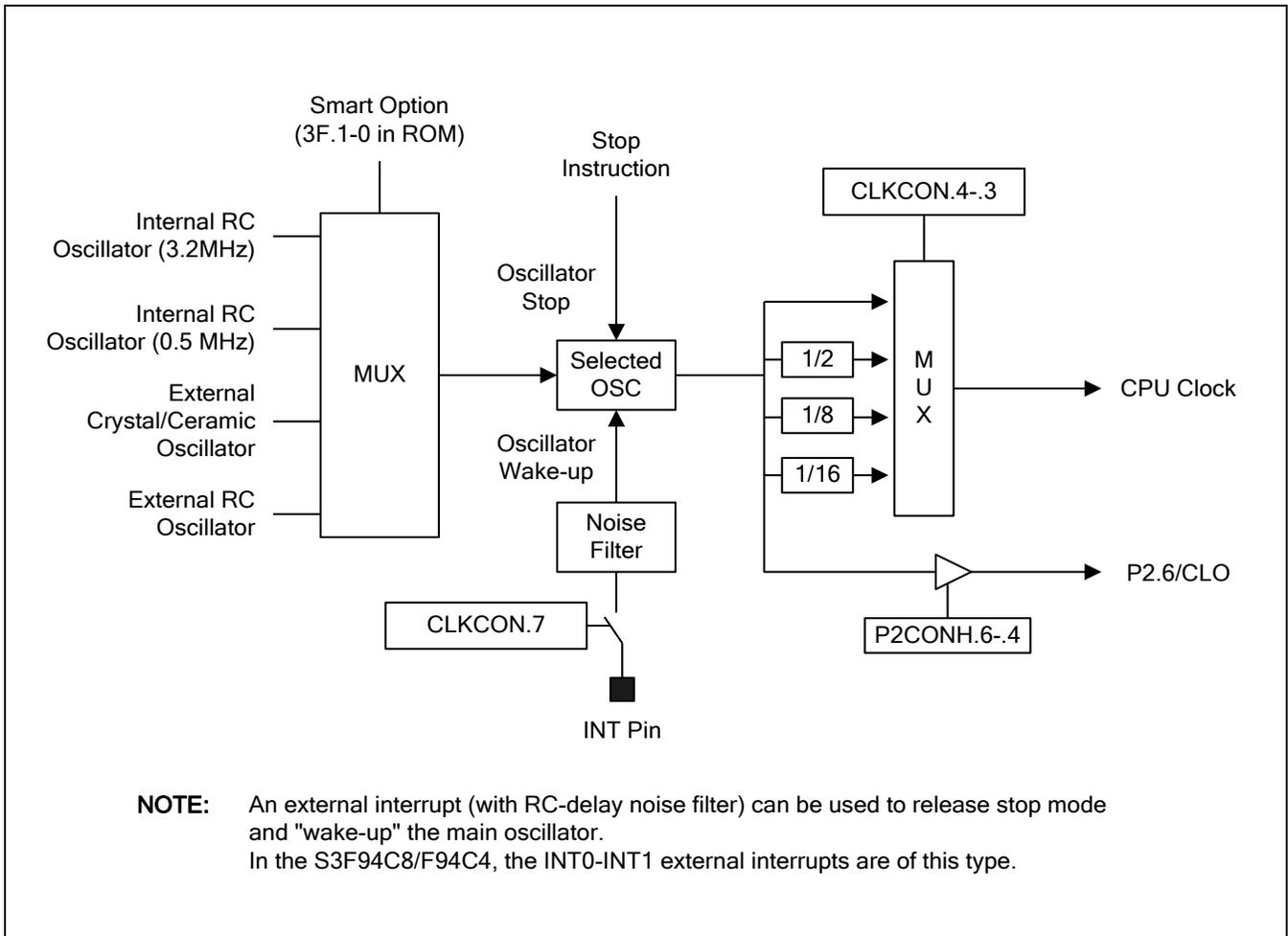
Figure 7-4. System Clock Circuit Diagram

8 RESET and Power-Down

8.1 System Reset

By Smart Option (3EH.7 in ROM), user can select internal RESET (LVR) or external RESET. In using internal RESET (LVR), nRESET pin (P1.2) can be used by normal I/O pin.

The S3F94C8/S3F94C4 can be RESET in four ways:

By external power-on-reset

By the external nRESET input pin pulled low

By the digital watchdog peripheral timing out

By by Low Voltage Reset (LVR)

During a external power-on reset, the voltage at V_{DD} is High level and the nRESET pin is forced to Low level. The nRESET signal is input through a Schmitt trigger circuit where it is then synchronized with the CPU clock. This brings the S3F94C8/S3F94C4 into a known operating status. To ensure correct start-up, the user should take care that nRESET signal is not released before the V_{DD} level is sufficient to allow MCU operation at the chosen frequency.

The nRESET pin must be held to Low level for a minimum time interval after the power supply comes within tolerance in order to allow time for internal CPU clock oscillation to stabilize. The minimum required oscillation stabilization time for a reset is approximately 52.4 ms (@ $2^{19}/f_{OSC}$, f_{OSC} = 10 MHz).

When a reset occurs during normal operation (with both V_{DD} and nRESET at High level), the signal at the nRESET pin is forced Low and the Reset operation starts. All system and peripheral control registers are then set to their default hardware Reset values (see Table 8-1).

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

The on-chip Low Voltage Reset, features static Reset when supply voltage is below a reference value (Typ. 1.9, 2.3, 3.0, 3.6, 3.9 V). Thanks to this feature, external reset circuit can be removed while keeping the application safety. As long as the supply voltage is below the reference value, there is a internal and static RESET. The MCU can start only when the supply voltage rises over the reference value.

When you calculate power consumption, please remember that a static current of LVR circuit should be added a CPU operating current in any operating modes such as Stop, Idle, and normal RUN mode when LVR enable in Smart Option.
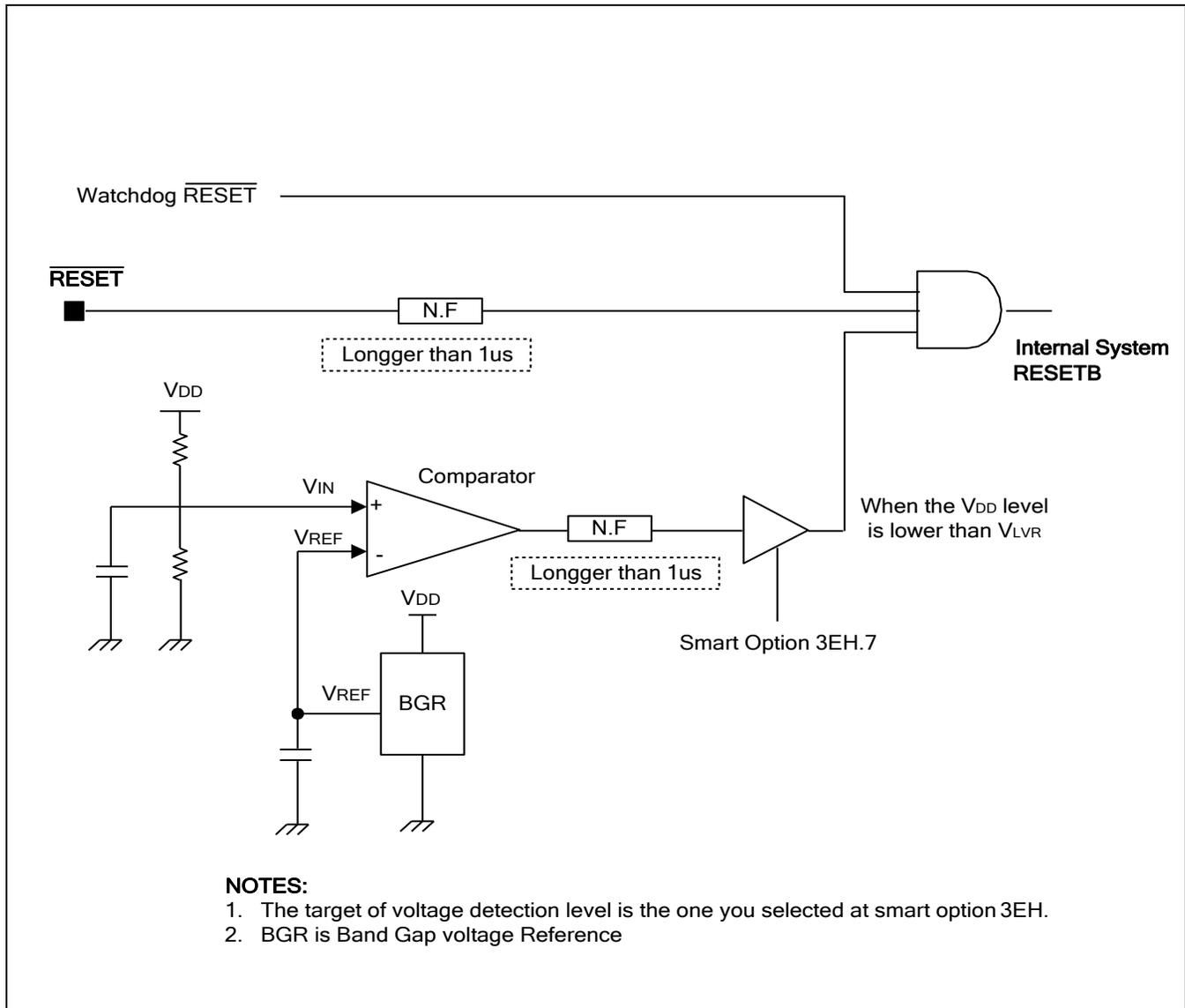
Figure 8-1. Low Voltage Reset Circuit

NOTE

To program the duration of the oscillation stabilization interval, you must make the appropriate settings to the Basic Timer Control Register, BTCON, before entering Stop Mode. Additionally, if you do not want to use the basic timer watchdog function (which causes a system reset if a basic timer counter overflow occurs), you can disable it by writing "1010B" to the upper nibble of BTCON.

8.1.1 External RESET pin

When the nRESET pin transiting from V_{IL} (low input level of reset pin) to V_{IH} (high input level of reset pin), the reset pulse is generated.
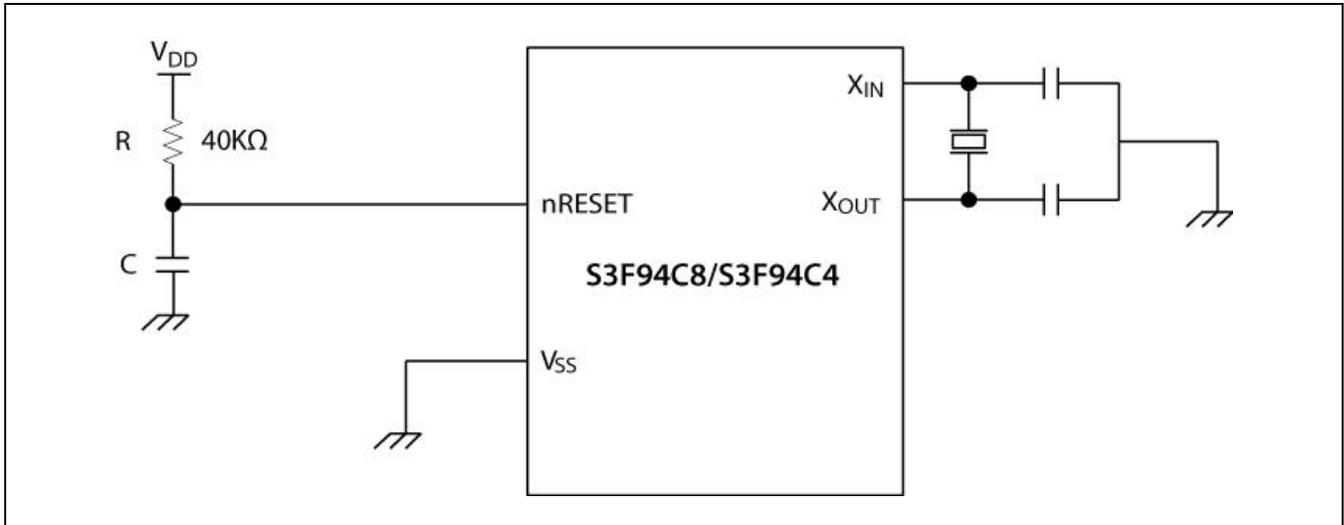
Figure 8-2. Recommended External RESET Circuit

NOTE

In Figure 8-2, to ensure that the voltage drop across Resistor R does not violate reset pulse detection, Zilog recommends that the resistance value of R be less than 40KΩ.

8.1.2 MCU Initialization Sequence

The following sequence of events occurs during a Reset operation:

All interrupts are disabled.

The watchdog function (basic timer) is enabled.

Ports 0–2 are set to input mode

Peripheral control and data registers are disabled and reset to their initial values (see Table 8-1).

The program counter is loaded with the ROM reset address, 0100H.

When the programmed oscillation stabilization time interval has elapsed, the address stored in ROM location 0100H (and 0101H) is fetched and executed.
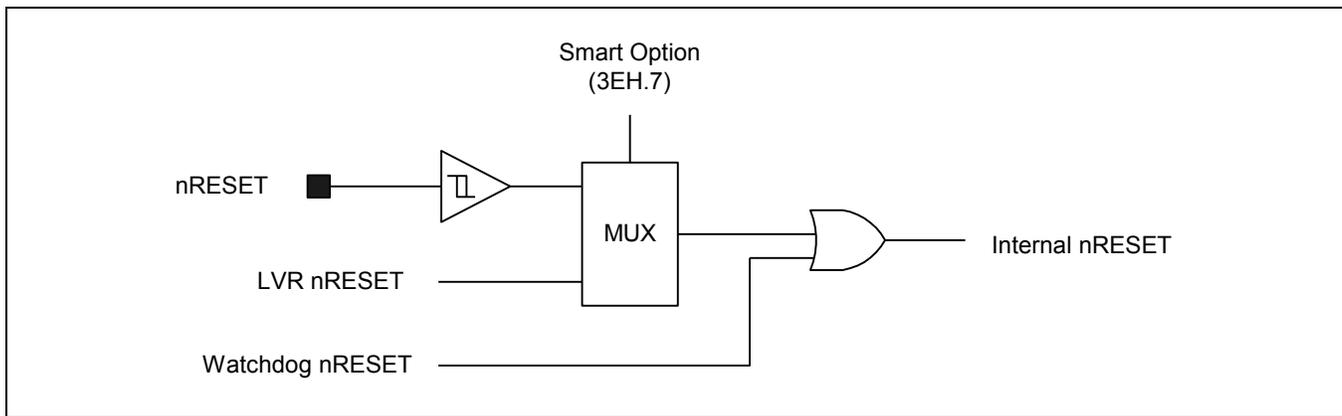
Figure 8-3. Reset Block Diagram
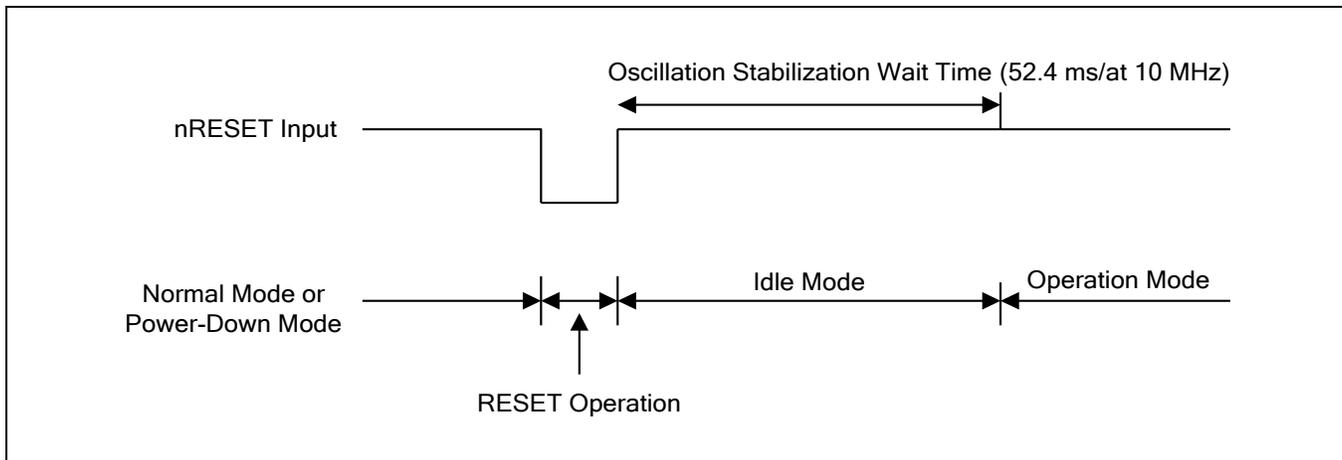
Figure 8-4. Timing for S3F94C8/S3F94C4 After RESET

8.2 Power-Down Modes

8.2.1 Stop Mode

Stop Mode is invoked by the instruction STOP (opcode 7FH). In Stop Mode, the operation of the CPU and all peripherals is halted. That is, the on-chip main oscillator stops and the supply current is reduced to less than $4\mu A$ except that the LVR(Low Voltage Reset) is enable. All system functions are halted when the clock "freezes", but data stored in the internal register file is retained. Stop Mode can be released in one of two ways: by a nRESET signal or by an external interrupt.

NOTE

Before executing the STOP instruction, set the STOPCON Register as "10100101b".

8.2.2 Using RESET to Release Stop Mode

Stop Mode is released when the nRESET signal is released and returns to High level. All system and peripheral control registers are then Reset to their default values and the contents of all data registers are retained. A Reset operation automatically selects a slow clock (f_{OSC}/16) because CLKCON.3 and CLKCON.4 are cleared to "00B".

After the oscillation stabilization interval has elapsed, the CPU executes the system initialization routine by fetching the 16-bit address stored in ROM locations 0100H and 0101H.

8.2.3 Using an External Interrupt to Release Stop Mode

External interrupts with an RC-delay noise filter circuit can be used to release Stop Mode (Clock-related external interrupts cannot be used). External interrupts INT0-INT1 in the S3F94C8/S3F94C4 interrupt structure meet this criterion.

Note that when Stop Mode is released by an external interrupt, the current values in system and peripheral control registers are not changed. When you use an interrupt to release Stop Mode, the CLKCON.3 and CLKCON.4 register values remain unchanged, and the currently selected clock value is used. If you use an external interrupt for Stop Mode Release, you can also program the duration of the oscillation stabilization interval. To do this, you must put the appropriate value to BTCON register *before* entering Stop Mode.

The external interrupt is serviced when the Stop Mode Release occurs. Following the IRET from the service routine, the instruction immediately following the one that initiated Stop Mode is executed.

8.2.4 Idle Mode

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

There are two ways to release Idle Mode:

Execute a Reset. All system and peripheral control registers are Reset to their default values and the contents of all data registers are retained. The Reset automatically selects a slow clock (f_{OSC}/16) because CLKCON.3 and CLKCON.4 are cleared to "00B". If interrupts are masked, a Reset is the only way to release Idle Mode.

Activate any enabled interrupt, causing Idle Mode to be released. When you use an interrupt to release Idle Mode, the CLKCON.3 and CLKCON.4 register values remain unchanged, and the currently selected clock value is used. The interrupt is then serviced. Following the IRET from the service routine, the instruction immediately following the one that initiated Idle Mode is executed.

NOTES

1. Only external interrupts that are not clock-related can be used to release Stop Mode. To release Idle Mode; however, any type of interrupt (that is, internal or external) can be used.
2. Before entering Stop or Idle modes, the ADC must be disabled; otherwise, Stop or Idle current will be

significantly increased.

8.3 Hardware Reset Values

Table 8-1 lists the values for CPU and system registers, peripheral control registers, and peripheral data registers following a Reset operation in normal operating mode.

A 1 or a 0 shows the Reset bit value as logic one or logic zero, respectively.

An "x" means that the bit value is undefined following a reset.

A dash ("–") means that the bit is either not used or not mapped.

Table 8-1. Register Values After a Reset

Register Name	Mnemonic	Address & Location		RESET Value (Bit)							
		Address	R/W	7	6	5	4	3	2	1	0
Timer 0 Counter Register	T0CNT	D0H	R	0	0	0	0	0	0	0	0
Timer 0 Data Register	T0DATA	D1H	R/W	1	1	1	1	1	1	1	1
Timer 0 Control Register	T0CON	D2H	R/W	0	0	–	–	0	–	0	0
Location D3H is not mapped											
Clock Control Register	CLKCON	D4H	R/W	0	–	–	0	0	–	–	–
System Flags Register	FLAGS	D5H	R/W	x	x	x	x	–	–	–	–
Locations D6H–D8H are not mapped											
Stack Pointer Register	SP	D9H	R/W	x	x	x	x	x	x	x	x
Location DAH is not mapped											
MDS Special Register	MDSREG	DBH	R/W	0	0	0	0	0	0	0	0
Basic Timer Control Register	BTCON	DCH	R/W	0	0	0	0	0	0	0	0
Basic Timer Counter Register	BTCNT	DDH	R	0	0	0	0	0	0	0	0
Test Mode Control Register	FTSTCON	DEH	W	–	–	0	0	0	0	0	0
System Mode Register	SYM	DFH	R/W	–	–	–	–	0	0	0	0

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

Table 8-1. Register Values After a Reset (Continued)

Register Name	Mnemonic	Address Hex	R/W	7	6	5	4	3	2	1	0
				header: Bit Values After RESET							
Port 0 Data Register	P0	E0H	R/W	0	0	0	0	0	0	0	0
Port 1 Data Register	P1	E1H	R/W	–	–	–	–	–	0	0	0
Port 2 Data Register	P2	E2H	R/W	–	0	0	0	0	0	0	0
Locations E3H–E5H are not mapped											
Port 0 Control High Byte Register	P0CONH	E6H	R/W	0	0	0	0	0	0	0	0
Port 0 Control Register	P0CON	E7H	R/W	0	0	0	0	0	0	0	0
Port 0 Interrupt Pending Register	P0PND	E8H	R/W	–	–	–	–	0	0	0	0
Port 1 Control Register	P1CON	E9H	R/W	0	0	–	–	0	0	0	0
Port 2 Control High Byte Register	P2CONH	EAH	R/W	–	0	0	0	0	0	0	0
Port 2 Control Low Byte Register	P2CONL	EBH	R/W	0	0	0	0	0	0	0	0
Flash memory Control Register	FMCON	ECH	R/W	0	0	0	0	0	–	–	0
Flash memory user programming enable register	FMUSR	EDH	R/W	0	0	0	0	0	0	0	0
Flash memory sector address High Byte Register	FMSECH	EEH	R/W	0	0	0	0	0	0	0	0
Flash memory sector address register (low byte)	FMSECL	EFH	R/W	0	0	0	0	0	0	0	0
PWM Data Register 1	PWMDATA1	F0H	R/W	0	0	0	0	0	0	0	0
PWM Extension Register	PWMEX	F1H	R/W	0	0	0	0	0	0	0	0
PWM Data Register	PWMDATA	F2H	R/W	0	0	0	0	0	0	0	0
PWM Control Register	PWMCON	F3H	R/W	0	0	–	0	0	0	0	0
Stop Control Register	STOPCON	F4H	R/W	0	0	0	0	0	0	0	0
Locations F5H–F6H are not mapped											
A/D Control Register	ADCON	F7H	R/W	0	0	0	0	0	0	0	0
A/D converter Data Register (High)	ADDATAH	F8H	R	x	x	x	x	x	x	x	x
A/D converter Data Register (Low)	ADDATAL	F9H	R	0	0	0	0	0	0	x	x
Locations FAH–FFH are not mapped											

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

☞ **PROGRAMMING TIP — Sample S3F94C8/S3F94C4 Initialization Routine**

```
;-------------<< Interrupt Vector Address >>
         ORG        0000H
         VECTOR     00H,INT_94C4            ;  S3F94C8/S3F94C4 has only one interrupt vector

;-------------<< Smart Option >>
         ORG        003CH
         DB         00H                     ;  003CH, must be initialized to 0
         DB         00H                     ;  003DH, must be initialized to 0
         DB         0E7H                    ;  003EH, enable LVR (2.3 V)
         DB         03H                     ;  003FH, internal RC (3.2 MHz in V_DD = 5 V)

;-------------<< Initialize System and Peripherals >>
         ORG        0100H
RESET:   DI                                 ;  disable interrupt
         LD         BTCON,#10100011B        ;  Watch-dog disable
         LD         CLKCON,#00011000B       ;  Select non-divided CPU clock
         LD         SP,#0C0H                ;  Stack pointer must be set

         LD         P0CONH,#10101010B       ;
         LD         P0CONL,#10101010B       ;  P0.0–P0.7 push-pull output
         LD         P1CON,#00001010B        ;  P1.0–P1.1 push-pull output
         LD         P2CONH,#01001010B       ;
         LD         P2CONL,#10101010B       ;  P2.0–P2.6 push-pull output

;-------------<< Timer 0 settings >>
         LD         T0DATA,#50H             ;  CPU = 3.2 MHz, interrupt interval = 6.4 msec
         LD         T0CON,#01001010B        ;  f_OSC/256, Timer 0 interrupt enable

;-------------<< Clear all data registers from 00h to 5FH >>
         LD         R0,#0                   ;  RAM clear
RAM_CLR: CLR        @R0                     ;
         INC        R0                      ;
         CP         R0,#0BFH                ;
         JP         ULE,RAM_CLR

;-------------<< Initialize other registers >>
         •
         •
         •
         EI                                 ;  Enable interrupt
```

☞ **PROGRAMMING TIP — Sample S3F94C8/S3F94C4 Initialization Routine (Continued)**

```
;-------------<< Main loop >>
MAIN:       NOP                                 ;  Start main loop
            LD        BTCON,#02H                ;  Enable watchdog function
                                                ;  Basic counter (BTCNT) clear
            •
            •
            CALL      KEY_SCAN                  ;
            •
            •
            •
            CALL      LED_DISPLAY               ;
            •
            •
            •
            CALL      JOB                       ;
            •
            •
            •
            JR        T,MAIN                    ;

;-------------<< Subroutines >>
KEY_SCAN: NOP                                   ;
            •
            •
            •
            RET

LED_DISPLAY: NOP                                ;
            •
            •
            •
            RET

JOB:        NOP                                 ;
            •
            •
            •
            RET
```

☞ **PROGRAMMING TIP — Sample S3F94C8/S3F94C4 Initialization Routine (Continued)**

```
;--------------<< Interrupt Service Routines >>       ;  Interrupt enable bit and pending bit check
INT_94C4:   TM          T0CON,#00000010B    ;  Timer 0 interrupt enable check
            JR          Z,NEXT_CHK1         ;
            TM          T0CON,#00000001B    ;  If Timer 0 interrupt was occurred,
            JP          NZ,INT_TIMER 0      ;  T0CON.0 bit would be set.


NEXT_CHK1:
            TM          PWMCOM,#00000010B   ;  PWM overflow interrupt enable check
            JR          Z,NEXT_CHK2         ;
            TM          P0PND,#00000001B    ;
            JP          NZ,PWMOVF_INT       ;


NEXT_CHK2:
            TM          P0PND,#00000010B    ;  INT0 interrupt enable check
            JR          Z,NEXT_CHK3         ;
            TM          P0PND,#00000001B    ;
            JP          NZ,INT0_INT         ;


NEXT_CHK3:
            TM          P0PND,#00001000B    ; INT1 interrupt enable check
            JP          Z,END_INT           ;
            TM          P0PND,#00000100B    ;
            JP          NZ,INT1_INT         ;
            IRET                            ;  Interrupt return


END_INT     ; IRET


;--------------< Timer 0 interrupt service routine >
INT_TIMER 0:
            •                               ;
            •
            AND         T0CON,#11110110B    ;  Pending bit clear
            IRET                            ;  Interrupt return


;--------------< PWM overflow interrupt service routine >
PWMOVF_INT:
            •
            •
            AND         PWMCON,#11110110B   ;  Pending bit clear
            IRET                            ;  Interrupt return
```

☞ **PROGRAMMING TIP — Sample S3F94C8/S3F94C4 Initialization Routine (Continued)**

```
;-------------< External interrupt0 service routine >
INT0_INT:    •
             •
             AND          P0PND,#11111110B       ;   INT0 Pending bit clear
             IRET                                ;   Interrupt return


;-------------< External interrupt1 service routine >
INT1_INT:    •
             •
             AND          P0PND,#11111011B       ;   INT1 Pending bit clear
             IRET                                ;   Interrupt return
             •
             •
             END                                 ;
```

9

I/O Ports

9.1 Overview

The S3F94C8/S3F94C4 has three I/O ports: with 18 pins total. You access these ports directly by writing or reading port data register addresses.

All ports can be configured as LED drive (High current output: typical 10 mA).

Table 9-1. S3F94C8/S3F94C4 Port Configuration Overview

Port	Function Description	Programmability
0	Bit-programmable I/O port for Schmitt trigger input or push-pull output. Pull-up resistors are assignable by software. Port 0 pins can also be used as alternative function. (ADC input, external interrupt input).	Bit
1	Bit-programmable I/O port for Schmitt trigger input or push-pull, open-drain output. Pull-up or pull-down resistors are assignable by software. Port 1 pins can also oscillator input/output or reset input by Smart Option. P1.2 is input only.	Bit
2	Bit-programmable I/O port for Schmitt trigger input or push-pull, open-drain output. Pull-up resistor are assignable by software. Port 2 can also be used as alternative function (ADC input, CLO, T0 clock output)	Bit

9.2 Port Data Registers

Table 9-2 gives you an overview of the port data register names, locations, and addressing characteristics. Data registers for ports 0-2 have the structure shown in Figure 9-1.

Table 9-2. Port Data Register Summary

Register Name	Mnemonic	Hex	R/W
Port 0 Data Register	P0	E0H	R/W
Port 1 Data Register	P1	E1H	R/W
Port 2 Data Register	P2	E2H	R/W

NOTE: A reset operation clears the P0–P2 Data Register to "00H".

I/O Port n Data Register (n = 0-2)

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

Pn.7 Pn.6 Pn.5 Pn.4 Pn.3 Pn.2 Pn.1 Pn.0
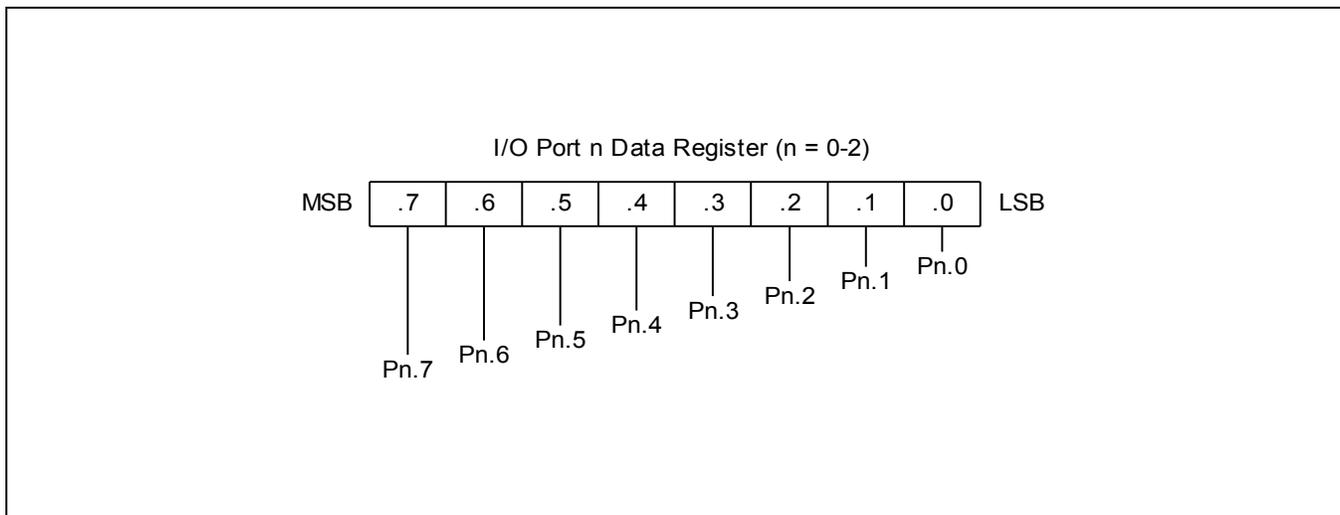
Figure 9-1. Port Data Register Format

9.2.1 Port 0

Port 0 is a bit-programmable, general-purpose, I/O ports. You can select normal input or push-pull output mode. In addition, you can configure a pull-up resistor to individual pins using control register settings.
It is designed for high-current functions such as LED direct drive. Part 0 pins can also be used as alternative functions (ADC input, external interrupt input and PWM output).

Two control resisters are used to control Port 0: P0CONH (E6H) and P0CONL (E7H).

You access Port 0 directly by writing or reading the corresponding port data register, P0 (E0H).

Figure 9-2. Port 0 Circuit Diagram

Port 0 Control Register (High Byte)
E6H, R/W

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

[.7-.6] Port, P0.7/ADC7 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = A/D converter input (ADC7); schmitt trigger input off

[.5-.4] Port 0, P0.6/ADC6/PWM Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Alternative function (PWM output)
1 0 = Push-pull output
1 1 = A/D converter input (ADC6); schmitt trigger input off

[.3-.2] Port 0, P0.5/ADC5 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = A/D converter input (ADC5); schmitt trigger input off

[.1-.0] Port 0, P0.4/ADC4 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = A/D converter input (ADC4); schmitt trigger input off

Figure 9-3. Port 0 Control Register (P0CONH, High Byte)

Port 0 Control Register (Low Byte)
E7H, R/W

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

[.7-.6] Port 0, P0.3/ADC3 Configuration Bits
0 0 = Schmitt trigger input
0 1 = Schmitt trigger input; pull-up enable
1 0 = Push-pull output
1 1 = A/D converter input (ADC3); Schmitt trigger input off

[.5-.4] Port 0, P0.2/ADC2 Configuration Bits
0 0 = Schmitt trigger input
0 1 = Schmitt trigger input; pull-up enable
1 0 = Push-pull output
1 1 = A/D converter input (ADC2); Schmitt trigger input off

[.3-.2] Port 0, P0.1/ADC1/INT1 Configuration Bits
0 0 = Schmitt trigger input/falling edge interrupt input
0 1 = Schmitt trigger input; pull-up enable/falling edge interrupt input
1 0 = Push-pull output
1 1 = A/D converter input (ADC1); Schmitt trigger input off

[.1-.0] Port 0, P0.0/ADC0/INT0 Configuration Bits
0 0 = Schmitt trigger input/falling edge interrupt input
0 1 = Schmitt trigger input; pull-up enable/falling edge interrupt input
1 0 = Push-pull output
1 1 = A/D converter input (ADC0); Schmitt trigger input off

Figure 9-4. Port 0   Control Register (P0CONL, Low Byte)

Port 0 Interrupt Pending Register
E8H, R/W

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

[.7-.4] Not used for S3F94C8/F94C4

[.3] Port 0.1/ADC1/INT1, Interrupt Enable Bit
0 = INT1 falling edge interrupt disable
1 = INT1 falling edge interrupt enable

[.2] Port 0.1/ADC1/INT1, Interrupt Pending Bit
0 = No interrupt pending (when read)
0 = Pending bit clear (when write)
1 = Interrupt is pending (when read)
1 = No effect (when write)

[.1] Port 0.0/ADC0/INT0, Interrupt Enable Bit
0 = INT0 falling edge interrupt disable
1 = INT0 falling edge interrupt enable

[.0] Port 0.0/ADC0/INT0, Interrupt Pending Bit
0 = No interrupt pending (when read)
0 = Pending bit clear (when write)
1 = Interrupt is pending (when read)
1 = No effect (when write)

Figure 9-5. Port 0 Interrupt Pending Registers (P0PND)

PORT 1

Port 1, is a 3-bit I/O port with individually configurable pins. It can be used for general I/O port (Schmitt trigger input mode, push-pull output mode or n-channel open-drain output mode). In addition, you can configure a pull-up and pull-down resistor to individual pin using control register settings. It is designed for high-current functions such as LED direct drive.P1.0, P1.1 are used for oscillator input/output by Smart Option. Additionally, P1.2 is used for RESET pin by Smart Option (LVR disable).

NOTE:

When P1.2 is configured as a general I/O port, it can be used only for Schmitt trigger input. P1.2 is also shared with VPP pin for Flash Programming, so it have intrinsic internal pull-down resistor (about 300Kohm), Please consider about the pull-down resistor when it used as I/O port.

One control register is used to control Port 1: P1CON (E9H).You address Port 1 bits directly by writing or reading the Port 1 Data Register, P1 (E1H). <u>When you use external oscillator, P1.0, P1.1 must be set to output port to prevent current consumption.</u>
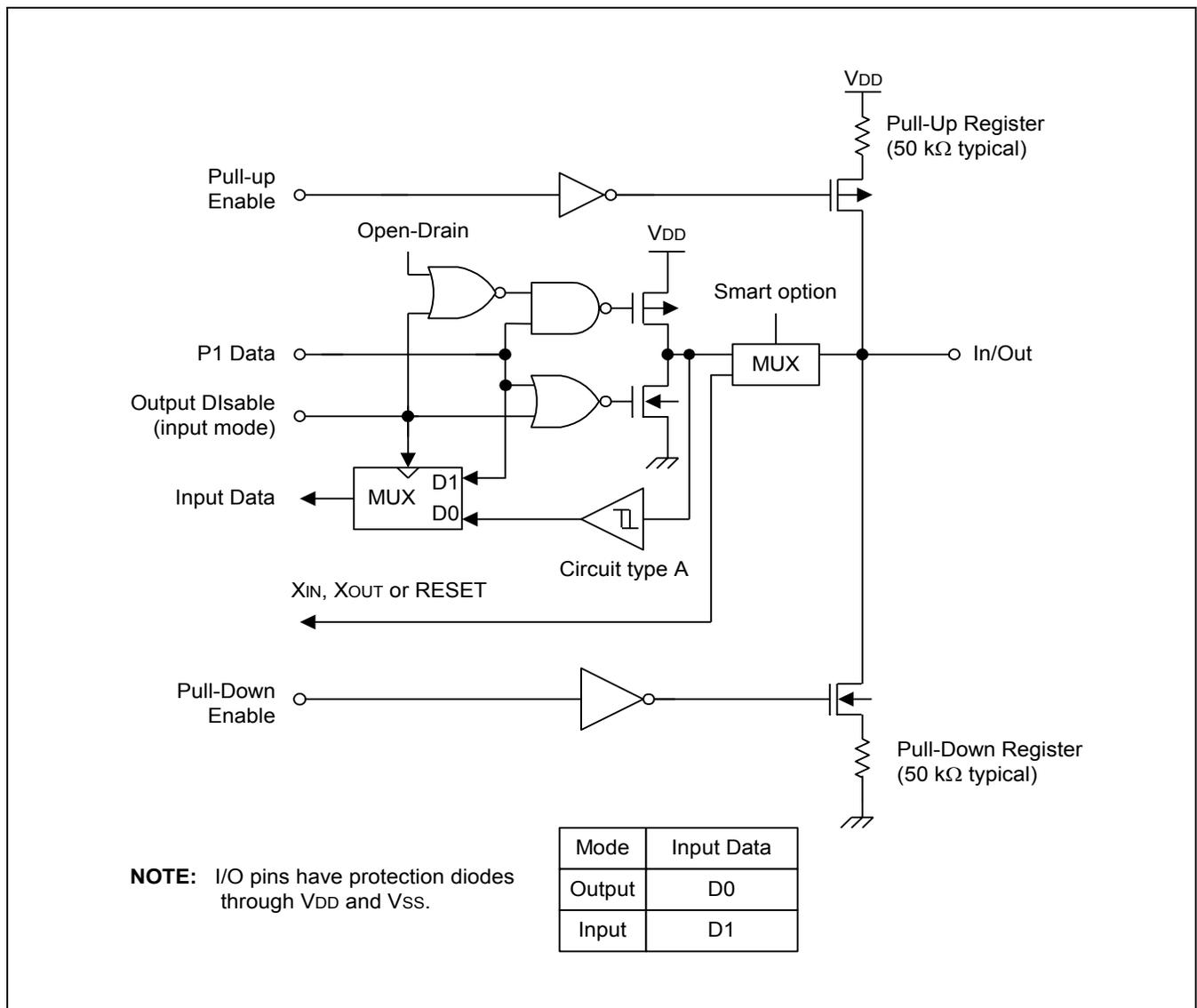
Figure 9-6. Port 1 Circuit Diagram

Port 1 Control Register
E9H, R/W

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

[.7] Port 1.1 N-Channel Open-Drain Enable Bit
0 = Configure P1.1 as a push-pull output
1 = Configure P1.1 as a n-channel open-drain output

[.6] Port 1.0 N-Channel Open-Drain Enable Bit
0 = Configure P1.0 as a push-pull output
1 = Configure P1.0 as a N-channel open-drain output

[.5-.4] Not used for S3F94C8/F94C4

[.3-.2] Port 1, P1.1 Configuration Bits
0 0 = Schmitt trigger input;
0 1 = Schmitt trigger input; pull-up enable
1 0 = Push-pull output
1 1 = Schmitt trigger input; pull-down enable

[.1-.0] Port 1, P1.0 Configuration Bits
0 0 = Schmitt trigger input;
0 1 = Schmitt trigger input; pull-up enable
1 0 = Push-pull output
1 1 = Schmitt trigger input; pull-down enable

NOTE: 1.When you use external oscillator, P1.0, P1.1 must be set to
output port to prevent current consumption.
2. when you enable LVR in smart option, P1.2(nRESET/VPP)
can be and can only be used as input port.

Figure 9-7. Port 1 Control Register (P1CON)

9.2.2 Port 2

Port 2 is a 7-bit I/O port with individually configurable pins. It can be used for general I/O port (Schmitt trigger input mode, push-pull output mode or N-channel open-drain output mode). You can also use some pins of Port 2 ADC input, CLO output and T0 clock output. In addition, you can configure a pull-up resistor to individual pins using control register settings. It is designed for high-current functions such as LED direct drive.

You address Port 2 bits directly by writing or reading the Port 2 Data Register, P2 (E2H). The Port 2 Control Register, P2CONH and P2CONL is located at addresses EAH, EBH respectively.
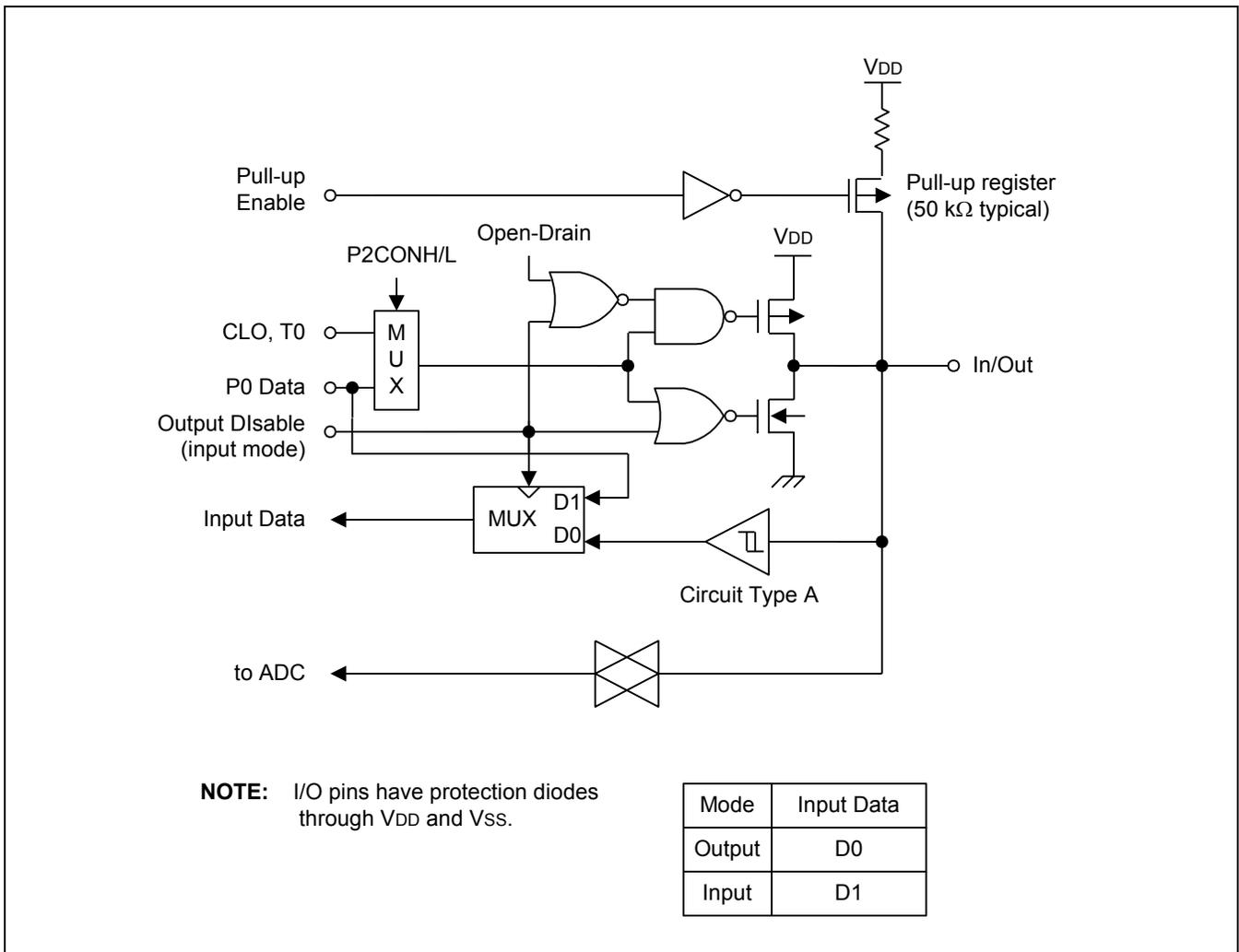
Figure 9-8. Port 2 Circuit Diagram

Port 2 Control Register (High Byte)
EAH, R/W

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

[.7] Not sued for S3F94C8/F94C4

[.6-.4] Port 2, P2.6/ADC8/CLO Configuration Bits
0 0 0 = Schmitt trigger input; pull-up enable
0 0 1 = Schmitt trigger input
0 1 x = ADC input
1 0 0 = Push-pull output
1 0 1 = Open-drain output; pull-up enable
1 1 0 = Open-drain output
1 1 1 = Alternative function; CLO output

[.3-.2] Port 2, P2.5 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = Open-drain output

[.1-.0] Port 2, P2.4 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = Open-drain output

NOTE: When noise problem is important issue, you had better not
use CLO output

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

Port 2 Control Register (Low Byte)
EBH, R/W

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

[.7-.6] Port 2, P2.3 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = Open-drain output

[.5-.4] Port 2, P2.2 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = Open-drain output

[.3-.2] Port 2, P2.1 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = Open-drain output

[.1-.0] Port 2, P2.0 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = T0 match output

Figure 9-10. Port 2 Control Register (P2CONL, Low Byte)

10 Basic Timer and Timer 0

10.1 Module Overview

The S3F94C8/S3F94C4 has two default timers: an 8-bit basic timer, one 8-bit general-purpose timer/counter, called Timer 0.

10.1.1 Basic Timer (BT)

You can use the basic timer (BT) in two different ways:

As a watchdog timer to provide an automatic Reset mechanism in the event of a system malfunction.

To signal the end of the required oscillation stabilization interval after a Reset or a Stop Mode Release.
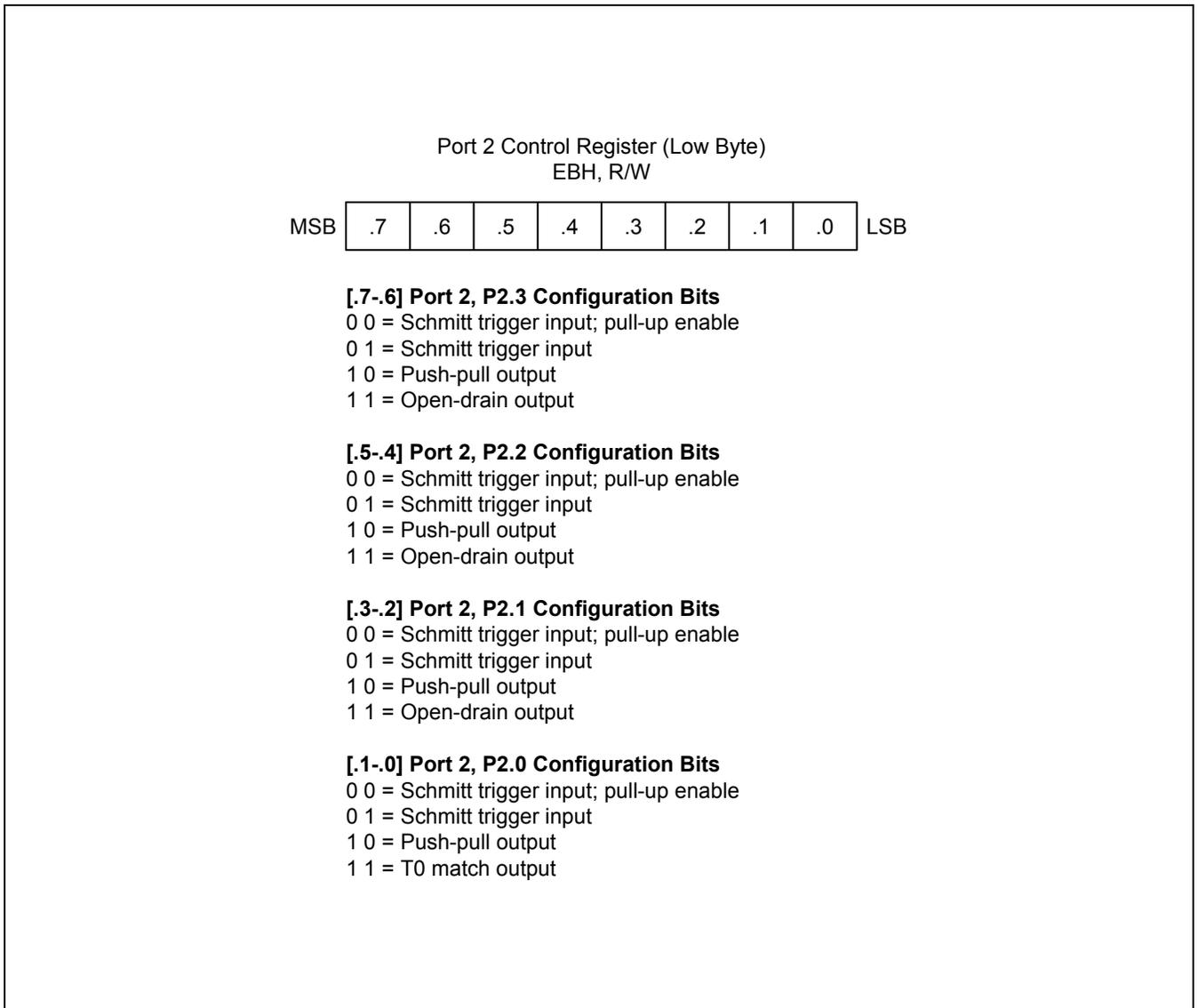

The functional components of the basic timer block are:

Clock frequency divider (f_{OSC} divided by 4096, 1024, or 128) with multiplexer

8-bit basic timer counter, BTCNT (DDH, read-only)

Basic Timer Control Register, BTCON (DCH, read/write)

10.1.2 Timer 0

Timer 0 has the following functional components:

Clock frequency divider (f_{OSC} divided by 4096, 256, 8, or f_{OSC}) with multiplexer

8-bit counter (T0CNT), 8-bit comparator, and 8-bit data register (T0DATA)

Timer 0 Control Register (T0CON)

10.2   Basic Timer

10.2.1 Basic Timer Control Register

The Basic Timer Control Register, BTCON, is used to select the input clock frequency, to clear the basic timer counter and frequency dividers, and to enable or disable the watchdog timer function.

A Reset clears BTCON to "00H". This enables the watchdog function and selects a basic timer clock frequency of $f_{OSC}/4096$. To disable the watchdog function, you must write the signature code "1010B" to the basic timer register control bits BTCON.7–BTCON.4.

The 8-bit basic timer counter, BTCNT, can be cleared during normal operation by writing a 1 to BTCON.1. To clear the frequency dividers for both the basic timer input clock and the Timer 0 clock, you write a 1 to BTCON.0.

Basic Timer Control Register (BTCON)
DCH, R/W

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

Watchdog timer enable bits:
1010B = Disable watchdog function
Other value = Enable watchdog function

Divider clear bit for basic timer and timer 0:
0 = No effect
1 = Clear both dividers

Basic timer counter clear bits:
0 = No effect
1 = Clear basic timer counter

Basic timer input clock selection bits:
00 = fosc/4096
01 = fosc/1024
10 = fosc/128
11 = Invalid selection

NOTE: When you write a 1 to BTCON.0 (or BTCON.1), the basic timer divider (or basic timer counter) is cleared. The bit is then cleared automatically to 0.
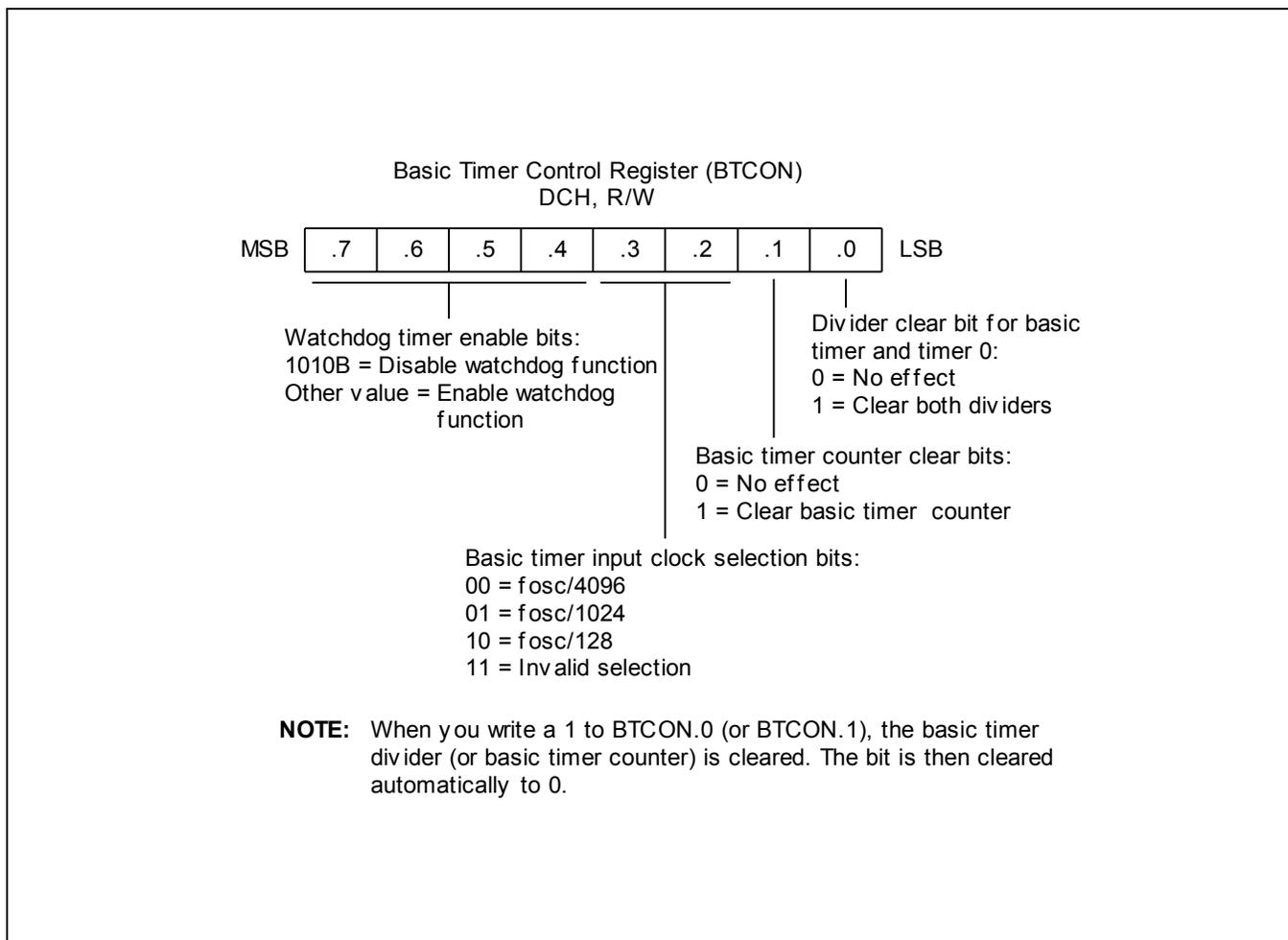
Figure 10-1. Basic Timer Control Register (BTCON)

10.3   Basic Timer Function Description

10.3.1 Watchdog Timer Function

You can program the basic timer overflow signal (BTOVF) to generate a Reset by setting BTCON.7–BTCON.4 to any value other than "1010B" (The "1010B" value disables the watchdog function). A Reset clears BTCON to "00H", automatically enabling the watchdog timer function. A Reset also selects the oscillator clock divided by 4096 as the BT clock.

A Reset whenever a basic timer counter overflow occurs. During normal operation, the application program must prevent the overflow, and the accompanying reset operation, from occurring. To do this, the BTCNT value must be cleared (by writing a 1 to BTCON.1) at regular intervals.

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

10.3.2 Oscillation Stabilization Interval Timer Function

You can also use the basic timer to program a specific oscillation stabilization interval following a Reset or when Stop Mode has been released by an external interrupt.

In Stop Mode, whenever a Reset or an external interrupt occurs, the oscillator starts. The BTCNT value then starts increasing at the rate of f_{OSC}/4096 (for Reset), or at the rate of the preset clock source (for an external interrupt).

When **BTCNT.7** is set, a signal is generated to indicate that the stabilization interval has elapsed and to gate the clock signal off to the CPU so that it can resume normal operation.

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

During Stop Mode, an external power-on Reset or an external interrupt occurs to trigger the Stop Mode Release and oscillation starts.

If an external power-on Reset occurred, the basic timer counter will increase at the rate of f_{OSC}/4096. If an external interrupt is used to release Stop Mode, the BTCNT value increases at the rate of the preset clock source.

Clock oscillation stabilization interval begins and continues until bit 7 of the basic timer counter is set.

When a BTCNT.7 is set, normal CPU operation resumes.

Figure 10-2 and 10-3 shows the oscillation stabilization time on RESET and Stop Mode Release
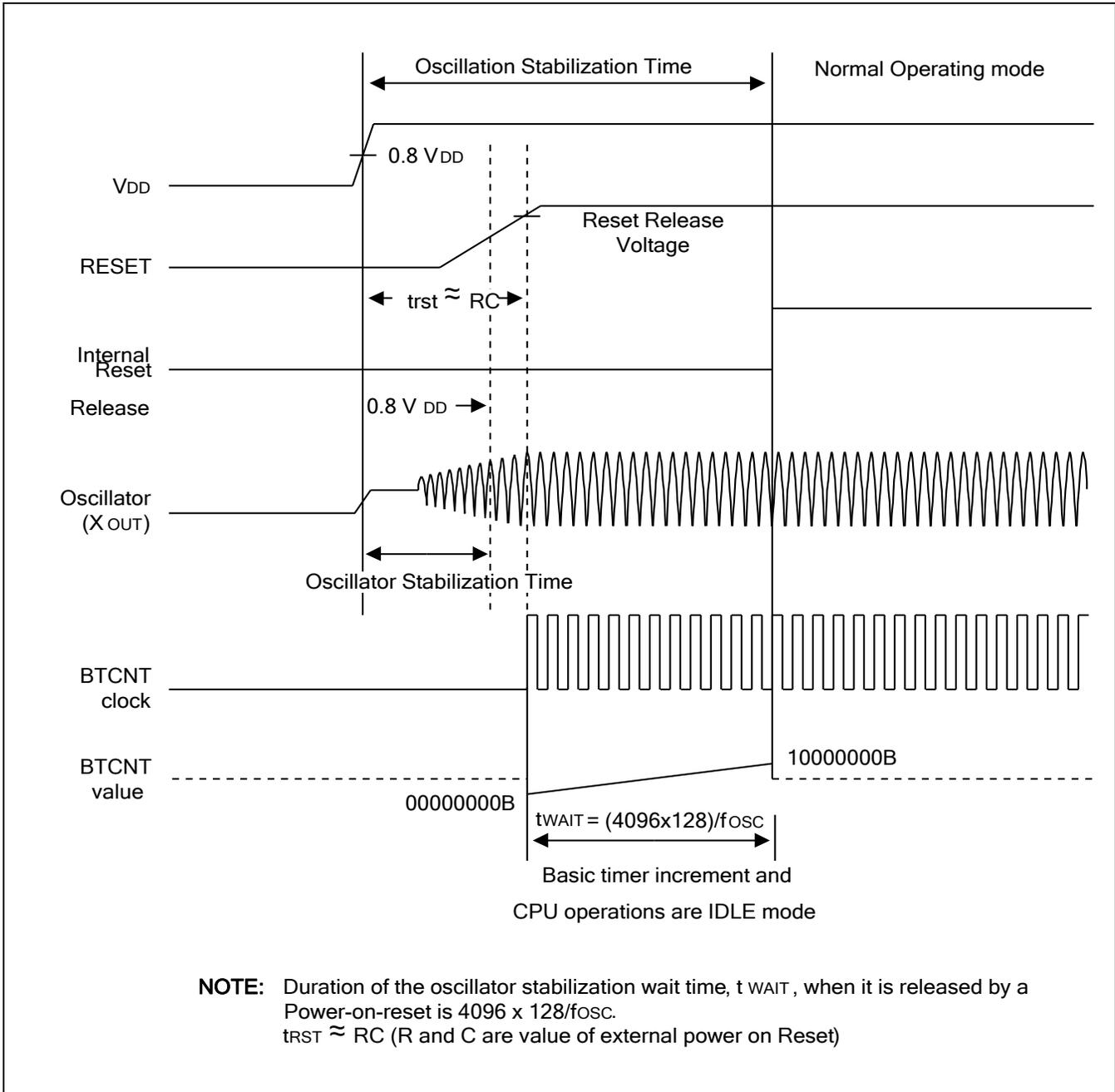
Figure 10-2. Oscillation Stabilization Time on RESET
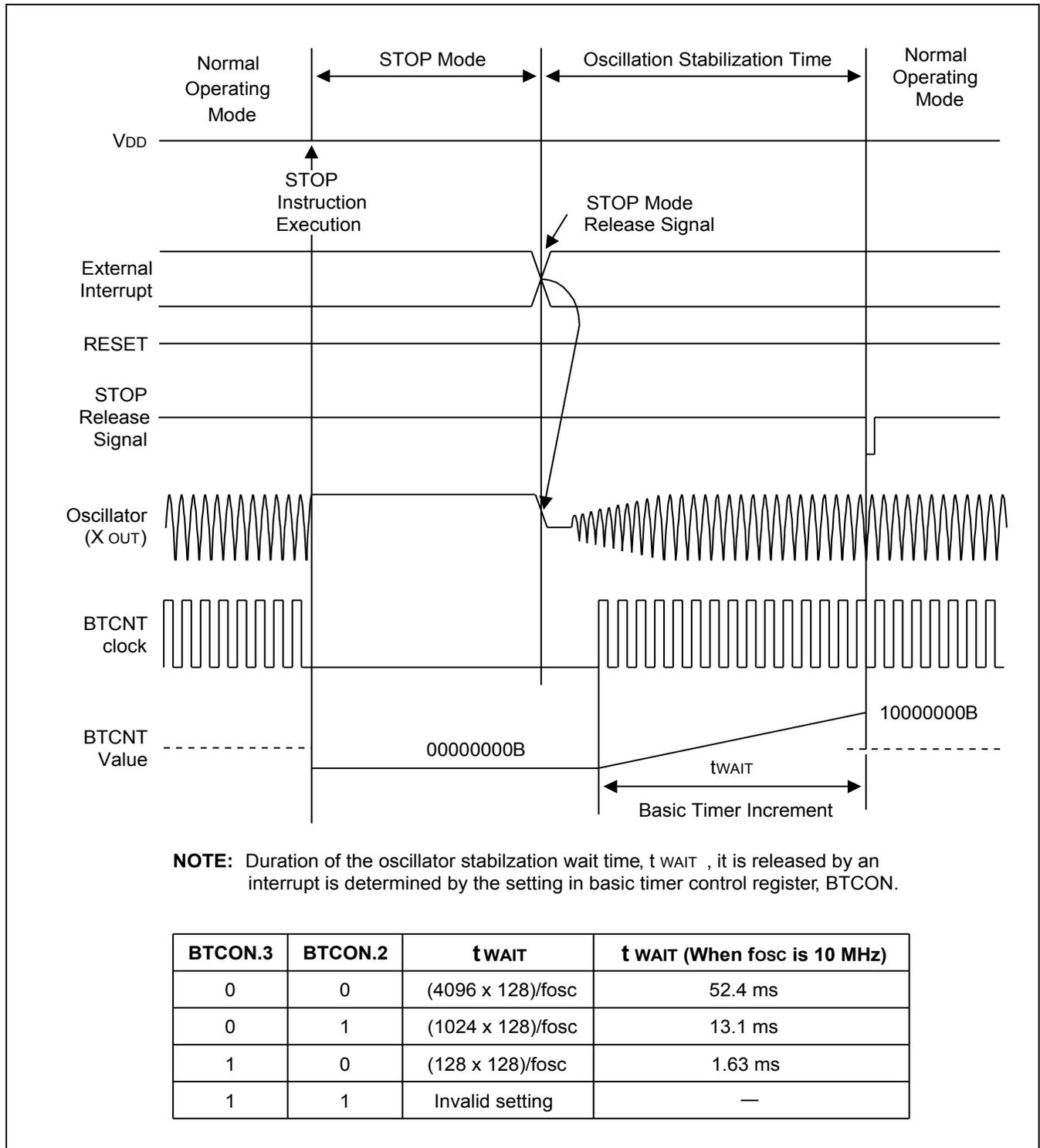
Figure 10-3. Oscillation Stabilization Time on Stop Mode Release

NOTE: Duration of the oscillator stabilzation wait time, t WAIT , it is released by an interrupt is determined by the setting in basic timer control register, BTCON.

BTCON.3	BTCON.2	t WAIT	t WAIT (When fosc is 10 MHz)
0	0	(4096 x 128)/fosc	52.4 ms
0	1	(1024 x 128)/fosc	13.1 ms
1	0	(128 x 128)/fosc	1.63 ms
1	1	Invalid setting	—

☞ **PROGRAMMING TIP — Configuring the Basic Timer**

The following example shows how to configure the basic timer to sample specification.

```
            ORG        0000H
            VECTOR     00H, INT_94C4          ;  S3F94C8/S3F94C4 has only one interrupt vector
```

;-------------<< Smart Option >>

```
            ORG        003CH
            DB         00H                    ;  003CH, must be initialized to 0
            DB         00H                    ;  003DH, must be initialized to 0
            DB         0E7H                   ;  003EH, enable LVR (2.3 V)
            DB         03H                    ;  003FH, internal RC (3.2 MHz in V_DD = 5 V)
```

;-------------<< Initialize System and Peripherals >>

```
            ORG        0100H
RESET:      DI                                ;  Disable interrupt
            LD         CLKCON, #00011000B     ;  Select non-divided CPU clock
            LD         SP, #0C0H              ;  Stack pointer must be set
            •
            •
            LD         BTCON,#02H             ;  Enable watchdog function
                                              ;  Basic timer clock: f_OSC/4096
                                              ;  Basic counter (BTCNT) clear
            •
            •
            •
            EI                                ;  Enable interrupt
```

;-------------<< Main loop >>

```
MAIN:       •
            LD         BTCON, #02H            ;  Enable watchdog function
                                              ;  Basic counter (BTCNT) clear
            •
            •
            •
            JR         T, MAIN                ;
```

;-------------<< Interrupt Service Routines >>

```
INT_94C4:   •                                ;  Interrupt enable bit and pending bit check
            •                                ;
            •                                ;  Pending bit clear
            IRET                             ;
            •
            •
            END                              ;
```

10.4   Timer 0

10.4.1 Timer 0 Control Registers

The Timer 0 Control Register, T0CON, is used to select the Timer 0 operating mode (interval timer) and input clock frequency to clear the Timer 0 counter and to enable the T0 match interrupt. It also contains a pending bit for T0 match interrupts.

A Reset clears T0CON to "00H". This sets Timer 0 to normal Interval Timer Mode, selects an input clock frequency of f_{OSC}/4096, and disables the T0 match interrupts. The T0 counter can be cleared at any time during normal operation by writing a 1 to T0CON.3.
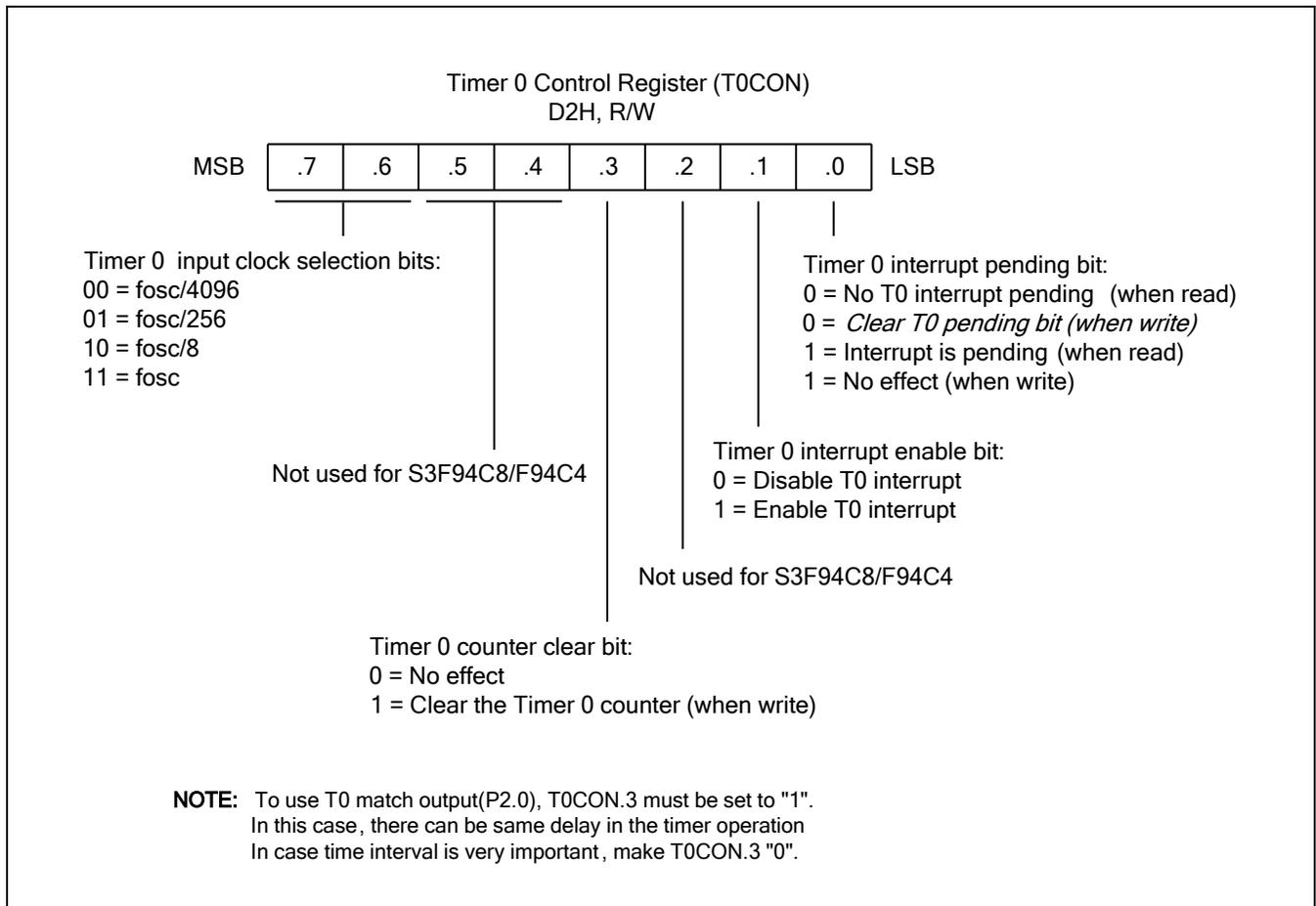
Timer 0 Control Register (T0CON)
D2H, R/W

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

Timer 0 input clock selection bits:
00 = fosc/4096
01 = fosc/256
10 = fosc/8
11 = fosc

Timer 0 interrupt pending bit:
0 = No T0 interrupt pending (when read)
0 = *Clear T0 pending bit (when write)*
1 = Interrupt is pending (when read)
1 = No effect (when write)

Not used for S3F94C8/F94C4

Timer 0 interrupt enable bit:
0 = Disable T0 interrupt
1 = Enable T0 interrupt

Not used for S3F94C8/F94C4

Timer 0 counter clear bit:
0 = No effect
1 = Clear the Timer 0 counter (when write)

NOTE: To use T0 match output(P2.0), T0CON.3 must be set to "1".
In this case, there can be same delay in the timer operation
In case time interval is very important, make T0CON.3 "0".

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

10.5   Timer 0 Function Description

10.5.1 Interval Timer Mode

In Interval Timer Mode, a match signal is generated when the counter value is identical to the value written to the Timer 0 Reference Data Register, T0DATA. The match signal generates a Timer 0 match interrupt (T0INT, vector 00H) and then clears the counter. If, for example, you write the value "10H" to T0DATA, the counter will increment until it reaches "10H". At this point, the Timer 0 interrupt request is generated; the counter value is reset and counting resumes.
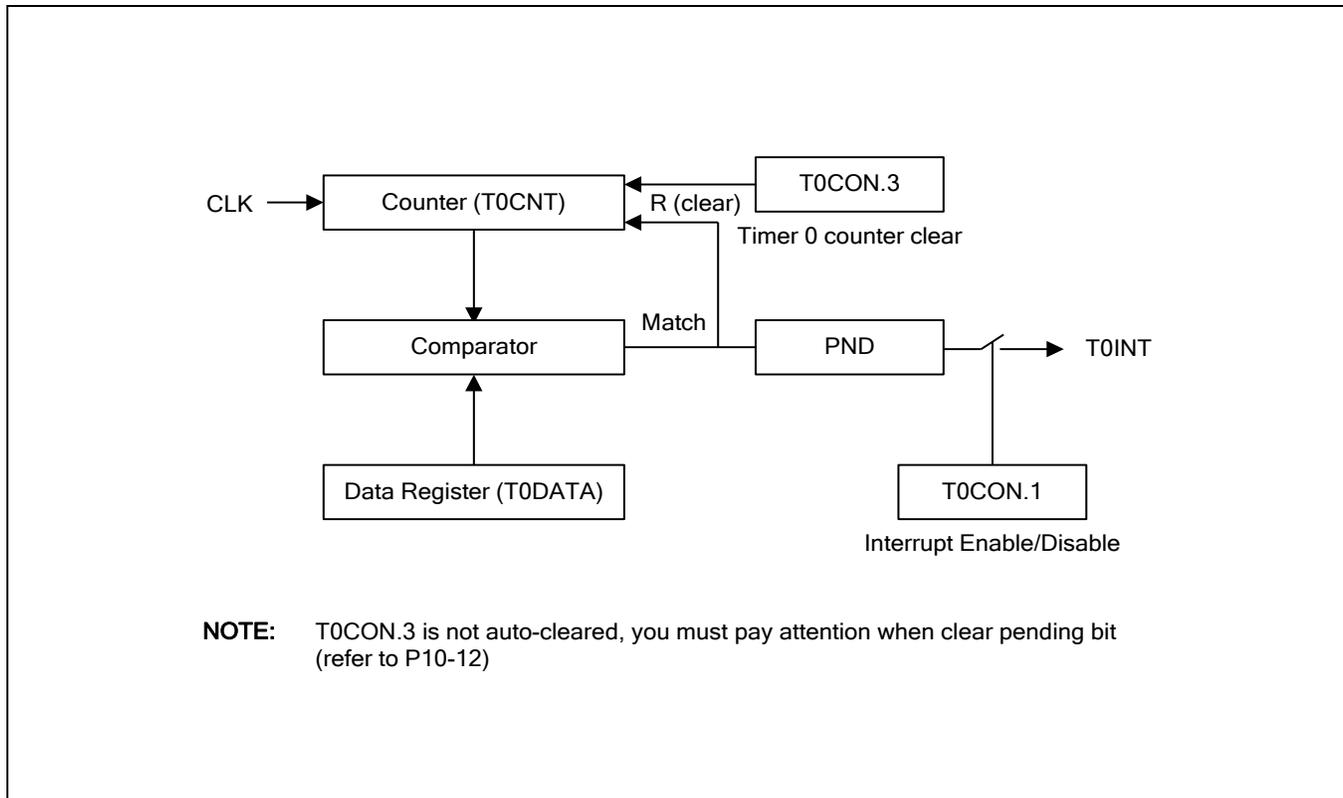

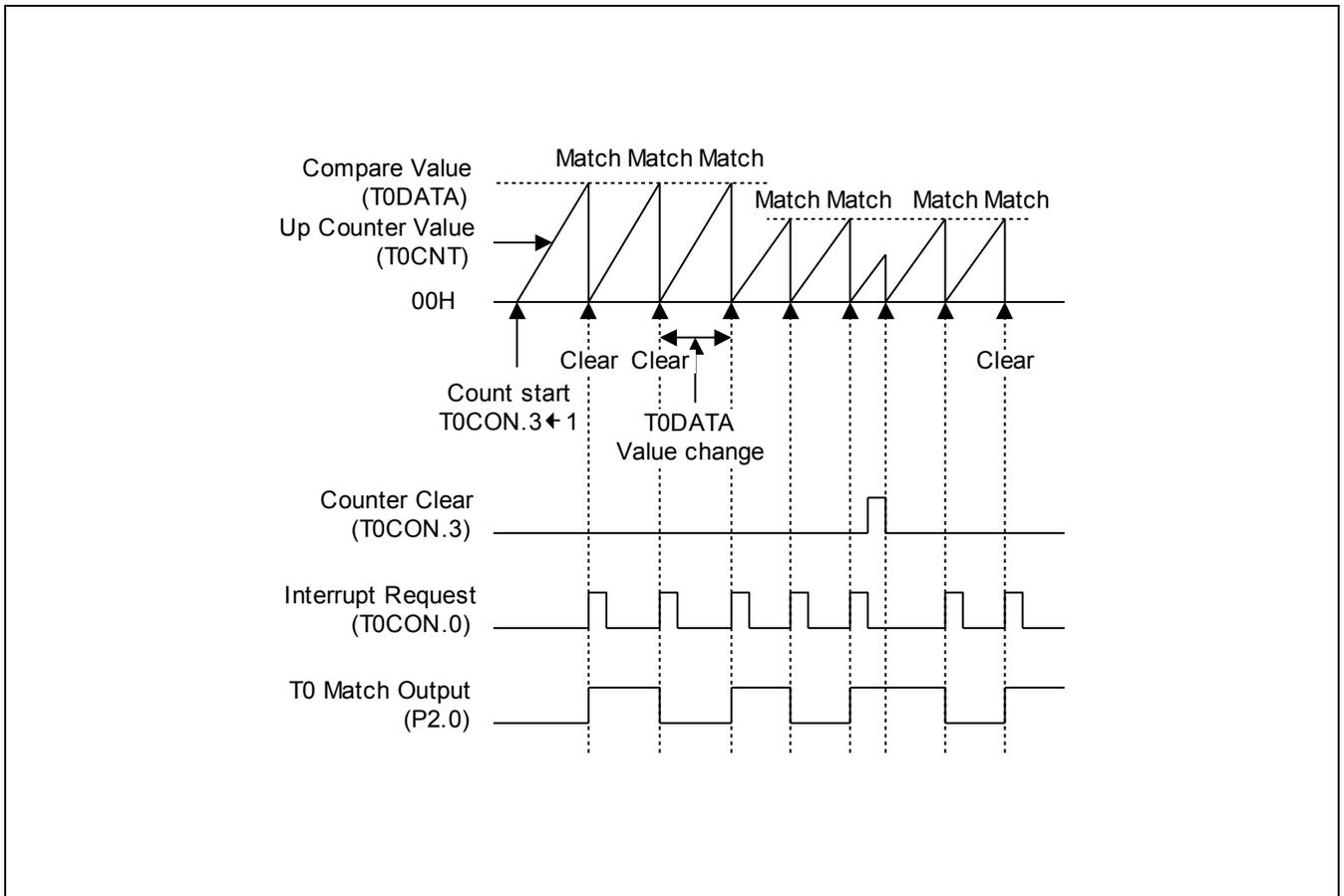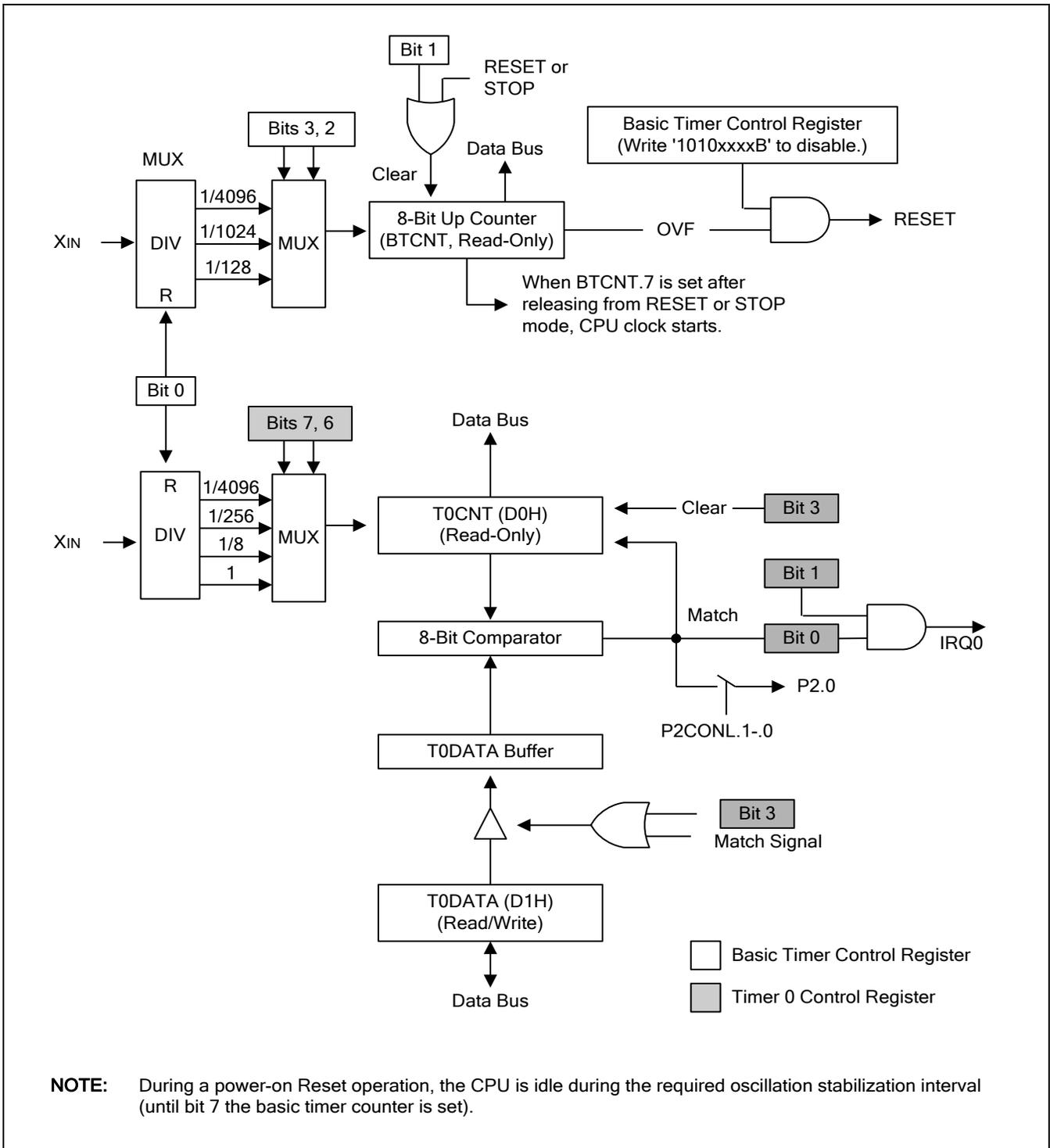**Figure 10-5. Simplified Timer 0 Function Diagram (Interval Timer Mode)**

Figure 10-6. Timer 0 Timing Diagram

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

☞ PROGRAMMING TIP1 – Configuring Timer 0 (Interval Mode)

The following sample program sets Timer 0 to Interval Timer Mode.

```
            ORG         0000H
            VECTOR      00H, INT_94C4           ;   S3F94C8/S3F94C4 has only one interrupt vector

            ORG         003CH
            DB          00H                     ;   003CH, must be initialized to 0
            DB          00H                     ;   003DH, must be initialized to 0
            DB          0E7H                    ;   003EH, enable LVR (2.3 V)
            DB          03H                     ;   003FH, internal RC (3.2 MHz in V_DD = 5 V)

            ORG         0100H
RESET:      DI                                  ;   Disable interrupt
            LD          BTCON,#10100011B        ;   Watchdog disable
            LD          CLKCON,#00011000B       ;   Select non-divided CPU clock
            LD          SP,#0C0H                ;   Set stack pointer
            LD          P0CONH,#10101010B       ;
            LD          P0CONL,#10101010B       ;   P0.0–0.7 push-pull output
            LD          P1CON,#00001010B        ;   P1.0–P1.1 push-pull output
            LD          P2CONH,#01001010B       ;
            LD          P2CONL,#10101010B       ;   P2.0–P2.6 push-pull output

;--------------<< Timer 0 settings >>

            LD          T0DATA, #50H            ;   CPU = 3.2 MHz, interrupt interval = 4 msec
            LD          T0CON, #01001010B       ;   f_OSC/256, Timer 0 interrupt enable
            •
            •
            •
            EI                                  ;   Enable interrupt
;--------------<< Main loop >>
MAIN:       NOP                                 ;   Start main loop
            •
            •
            •
            CALL        LED_DISPLAY             ;   Sub-block module
            •
            •
            •
            CALL        JOB                     ;   Sub-block module
            •
            •
            •
            JR          T, MAIN                 ;
```

☞ **PROGRAMMING TIP1 – Configuring Timer 0 (Interval Mode) (Continued)**

```
LED_DISPLAY:    NOP                              ;
                •                                ;
                •                                ;
                •                                ;
                RET                              ;
JOB:            NOP                              ;
                •                                ;
                •                                ;
                •                                ;
                RET                              ;
;-------------<< Interrupt Service Routines >>
INT_94C4:       TM      T0CON,#00000010B         ;  Interrupt enable check
                JR      Z,NEXT_CHK1              ;
                TM      T0CON, #00000001B        ;  If Timer 0 interrupt was occurred,
                JP      NZ,INT_TIMER 0           ;  T0CON.0 bit would be set.
NEXT_CHK1:      •                                ;  Interrupt enable bit and pending bit check
                •                                ;
                •                                ;
                IRET                             ;
INT_TIMER 0:                                     ;  Timer 0 interrupt service routine
                •
                •
                •
                AND     T0CON, #11110110B        ;   Pending bit clear
                IRET                             ;
                •
                •
                END                              ;
```

11

Pulse Width Modulation

11.1 Overview

This microcontroller has the PWM circuit. The PWM can be configured as one of these three resolutions:

8-bit resolution: 6-bit base + 2-bit extension

12-bit resolution: 6-bit base + 6-bit extension

14-bit resolution: 8-bit base + 6-bit extension

These three resolutions are mutually exclusive; only one resolution can work at any time. And which resolution is used is selected by PWMEX.1-.0.

The operation of all PWM circuit is controlled by a single control register, PWMCON.

The PWM counter is an incrementing counter. It is used by the PWM circuits. To start the counter and enable the PWM circuits, set PWMCON.2 to 1. If the counter is stopped, it retains its current count value; when restarted, it resumes counting from the retained count value. When there is a need to clear the counter set PWMCON.3 to 1.

You can select a clock for the PWM counter by set PWMCON.6-.7. Clocks which you can select are $f_{OSC}/64$, $f_{OSC}/8$, $f_{OSC}/2$, $f_{OSC}/1$.

11.2 Function Description

11.2.1 PWM

The PWM circuits have the following components:

PWM mode selection (PWMEX.1-.0)

Base comparator and extension cycle circuit

Base reference data registers (PWMDATA, PWMDATA1)

Extension data registers (PWMEX)

PWM output pins (P0.6/PWM)

11.2.2 PWM Counter

The PWM counter is an incrementing counter comprised of a lower base counter and an upper extension counter.

To determine the PWM module's base operating frequency, the lower base counter is compared to the PWM base data register value. In order to achieve higher resolutions, the extension bits of the upper counter can be used to modulate the "stretch" cycle. To control the "stretching" of the PWM output duty cycle at specific intervals, the extended counter value is compared with the value that you write to the module's extension bits.

11.2.3 PWM Data and Extension Registers

PWM (duty) data consist of base data bits and extension data bits; determine the output value generated by the PWM circuit. For each PWM resolution, the location of base data bits and extension data bits are different combination of register PWMDATA (F2H), PWMDATA1 (F0H) and PWMEX (F1H):

8-bit resolution, 6-bit base + 2-bit extension:

- Base 6 data bits: PWMDATA.7-.2
- Extension 2 bits: PWMDATA.1-.0

12-bit resolution, 6-bit base + 6-bit extension:

- Base 6 data bits: PWMDATA1.5-.0
- Extension   6 bits: PWMEX.7-.2

14-bit resolution, 8-bit base + 6-bit extension:

- Base 8 data bits: PWMDATA1.7-.0.
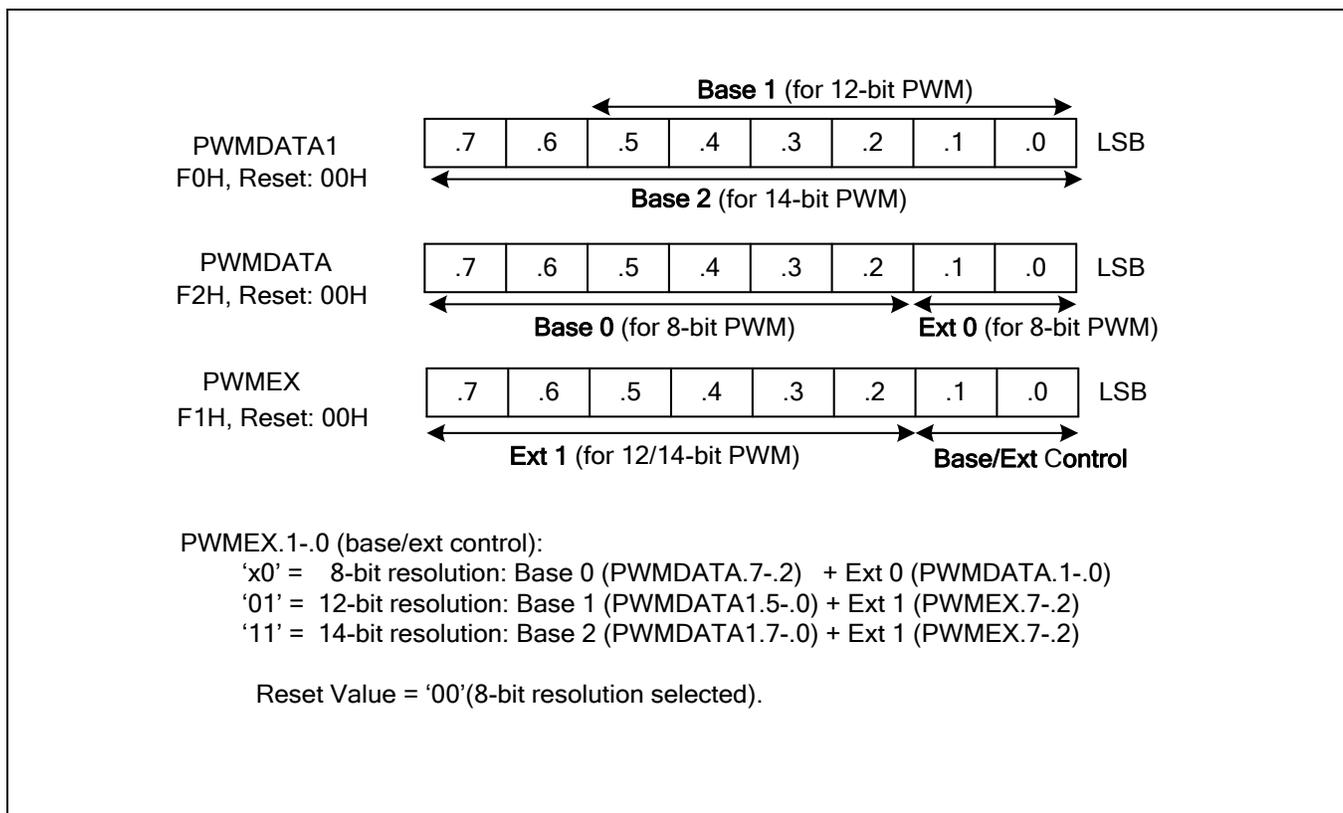- Extension 6 bits: PWMEX.7-.2

Figure 11-1. PWM Data and Extension Registers

To program the required PWM output, you load the appropriate initialization values into the data registers (PWMDATA) and the extension registers (PWMEX). To start the PWM counter, or to resume counting, set PWMCON.2 to 1.

A reset operation disables all PWM output. The current counter value is retained when the counter stops. When the counter starts, counting resumes at the retained value.

11.2.4 PWM Clock Rate

The timing characteristic of PWM output is based on the f_{OSC} clock frequency. The PWM counter clock value is

determined by the setting of PWMCON.6–.7.

Table 11-1. PWM Control and Data Registers

Register Name	Mnemonic	Address	Function
PWM data registers	PWMDATA	F2H	PWM waveform output setting registers.
	PWMDATA1	F0H	
	PWMEX	F1H	
PWM control registers	PWMCON	F3H	PWM counter stop/start (resume), and f_{OSC} clock settings

11.2.5 PWM Function Description

The PWM output signal toggles to Low level whenever the lower base counter matches the reference value stored in the module's data register (PWMDATA). If the value in the PWMDATA register is not zero, an overflow of the lower counter causes the PWM output to toggle to High level. In this way, the reference value written to the data register determines the module's base duty cycle.

The value in the extension counter is compared with the extension settings in the extension data bits. This extension counter value, together with extension logic and the PWM module's extension bits, is then used to "stretch" the duty cycle of the PWM output. The "stretch" value is one extra clock period at specific intervals, or cycles (see Table 11-2).

If, for example, in 8-bit base + 6-bit extension mode, the value in the extension register is '04H', the 32nd cycle will be one pulse longer than the other 63 cycles. If the base duty cycle is 50 %, the duty of the 32nd cycle will therefore be "stretched" to approximately 51% duty. For example, if you write 80H to the extension register, all odd-numbered cycles will be one pulse longer. If you write FCH to the extension register, all cycles will be stretched by one pulse except the 64th cycle. PWM output goes to an output buffer and then to the corresponding PWM output pin. In this way, you can obtain high output resolution at high frequencies.

11.2.6 PWM Output Waveform

6-bit base + 2-bit extension mode:

Table 11-2. PWM output "stretch" Values for Extension Data bits Ext0 (PWMDATA.1–.0)

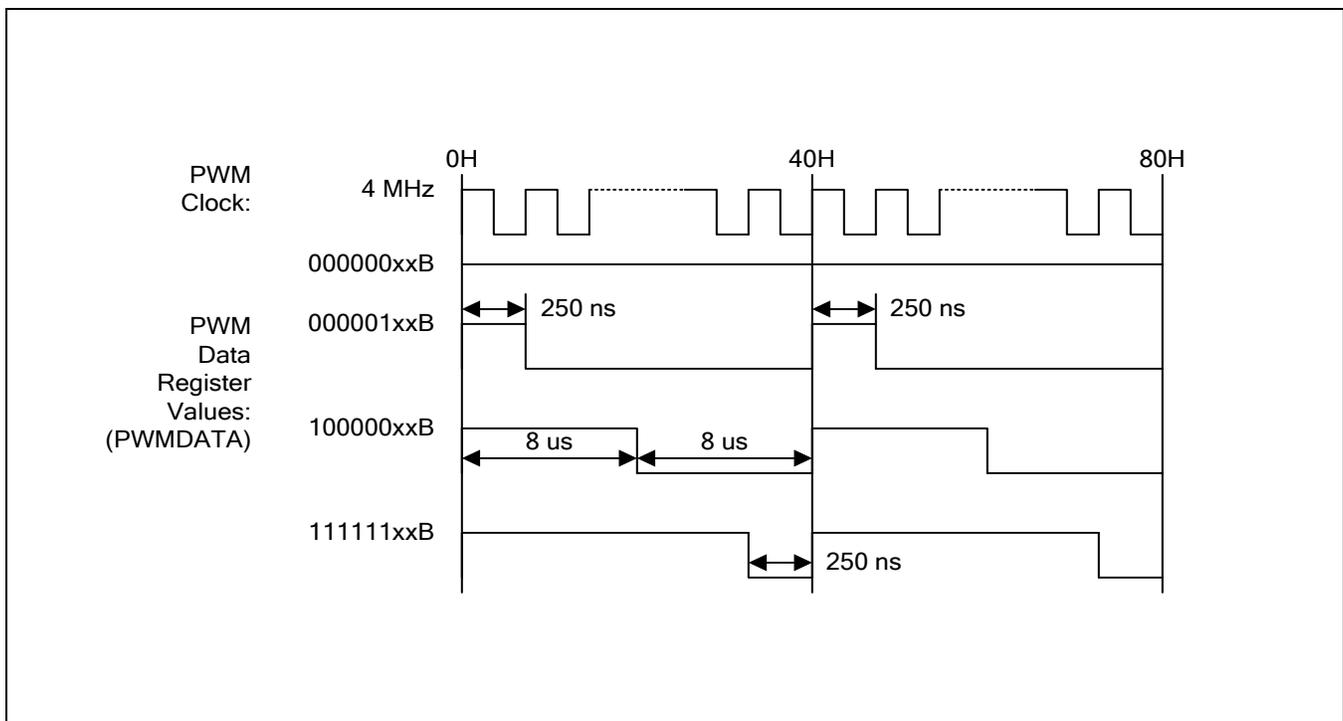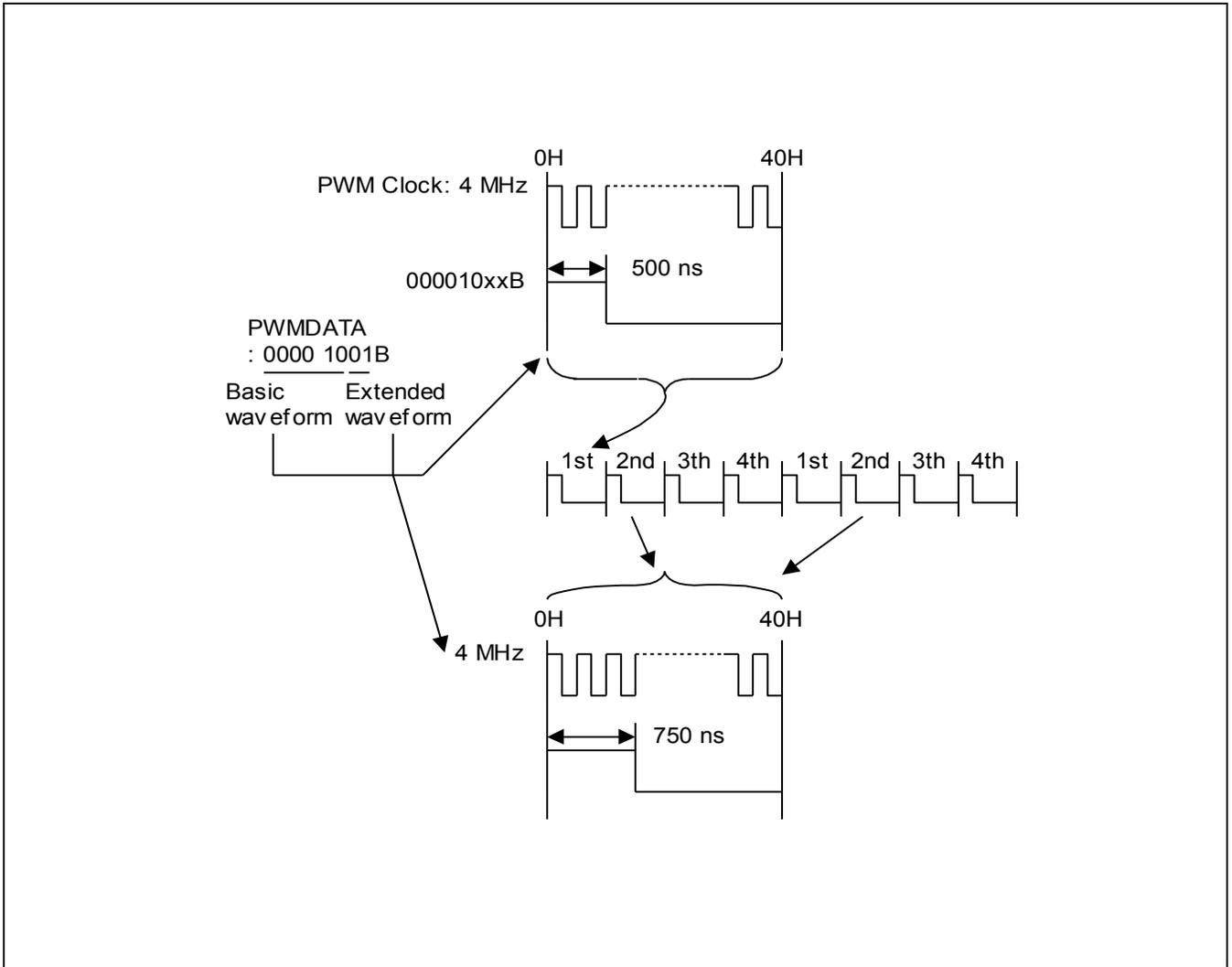| PWMDATA Bit (Bit1–Bit0)	"Stretched" Cycle Number
00	–
01	2
10	1, 3
11	1, 2, 3

Figure 11-2. PWM Basic Waveform (6-bit base)

Figure 11-3. Extended PWM Waveform (6-bit base + 2-bit extension)

6-bit base + 6-bit extension mode:

Table 11-3. PWM output "stretch" Values for Extension Data bits Ext1 (PWMEX.7-.2)

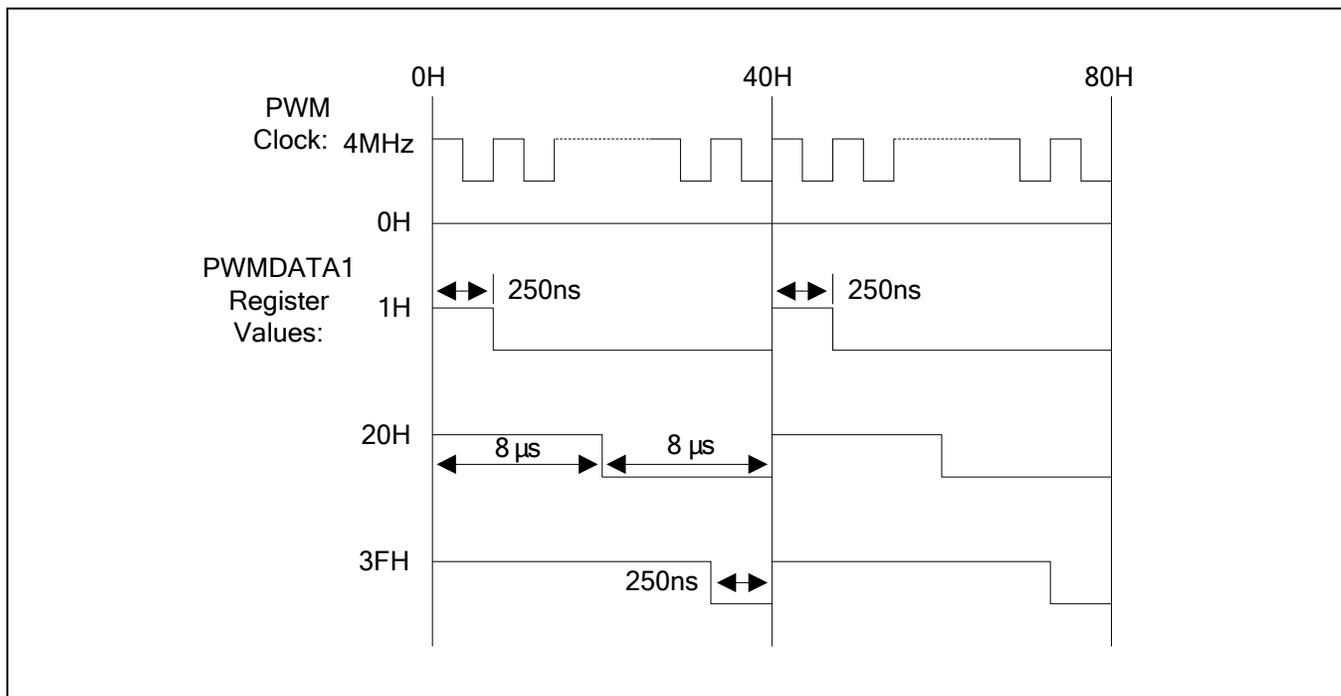| PWMEX Bit	"Stretched" Cycle Number
7	1, 3, 5, 7, 9, . . . , 55, 57, 59, 61, 63
6	2, 6, 10, 14, . . . , 50, 54, 58, 62
5	4, 12, 20, . . . , 44, 52, 60
4	8, 24, 40, 56
3	16, 48
2	32

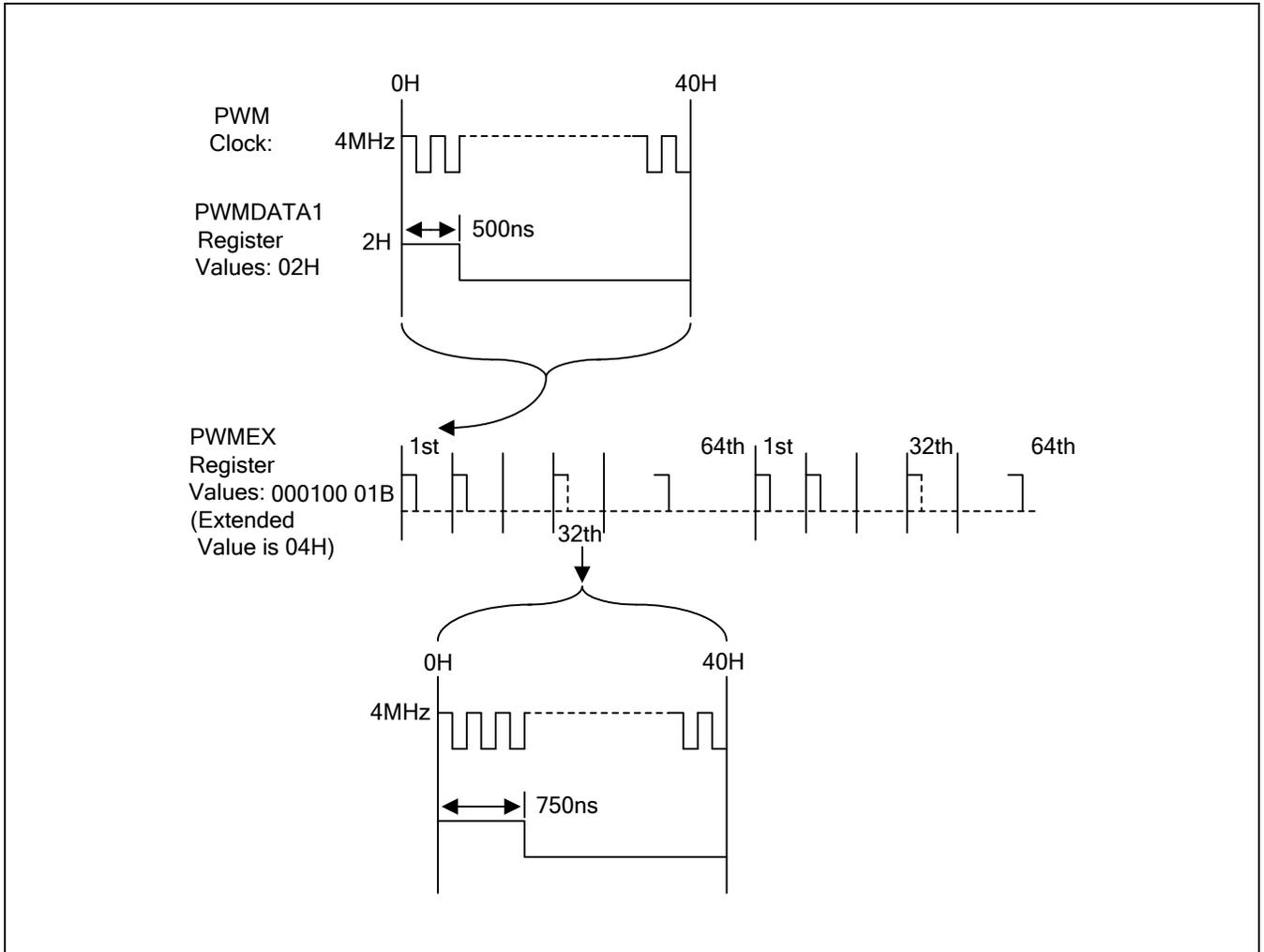**Figure 11-4. PWM Basic Waveform (6-bit base)**

Figure 11-5. Extended PWM Waveform (6-bit base + 6-bit extension)

8-bit base + 6-bit extension mode:

Table 11-4. PWM output "stretch" Values for Extension Data bits Ext1 (PWMEX.7-.2)

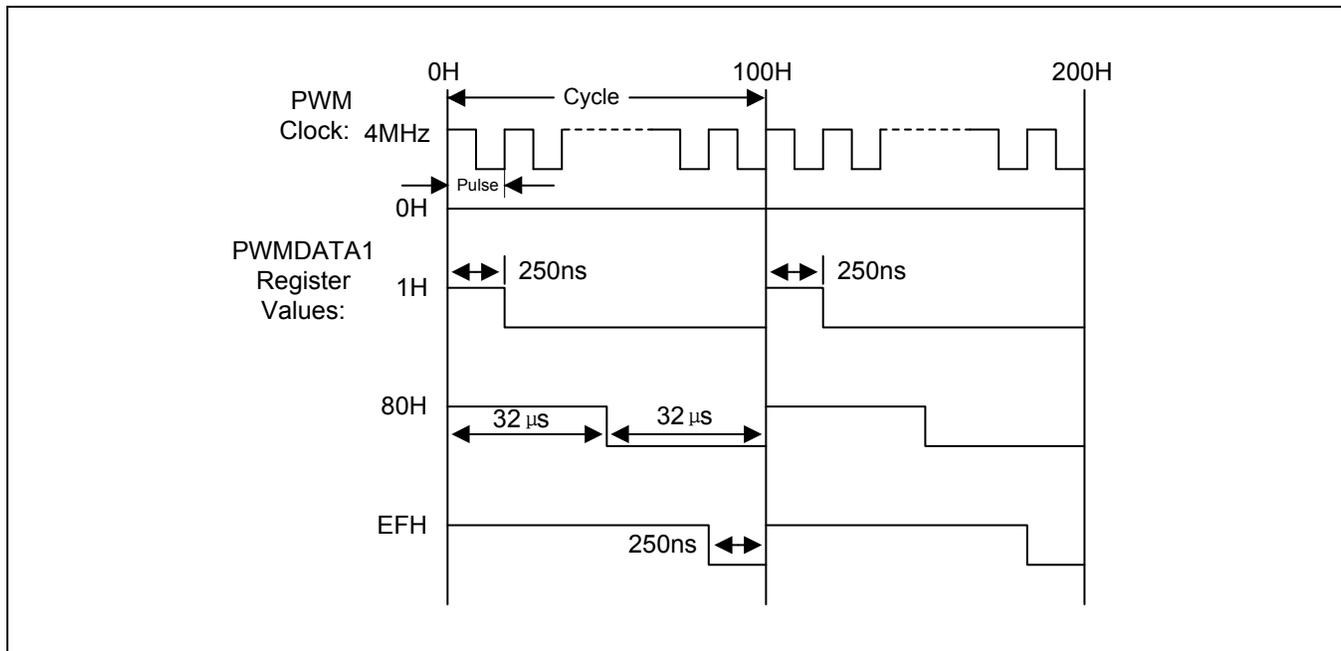| PWMEX Bit	"Stretched" Cycle Number
7	1, 3, 5, 7, 9, . . . , 55, 57, 59, 61, 63
6	2, 6, 10, 14, . . . , 50, 54, 58, 62
5	4, 12, 20, . . . , 44, 52, 60
4	8, 24, 40, 56
3	16, 48
2	32

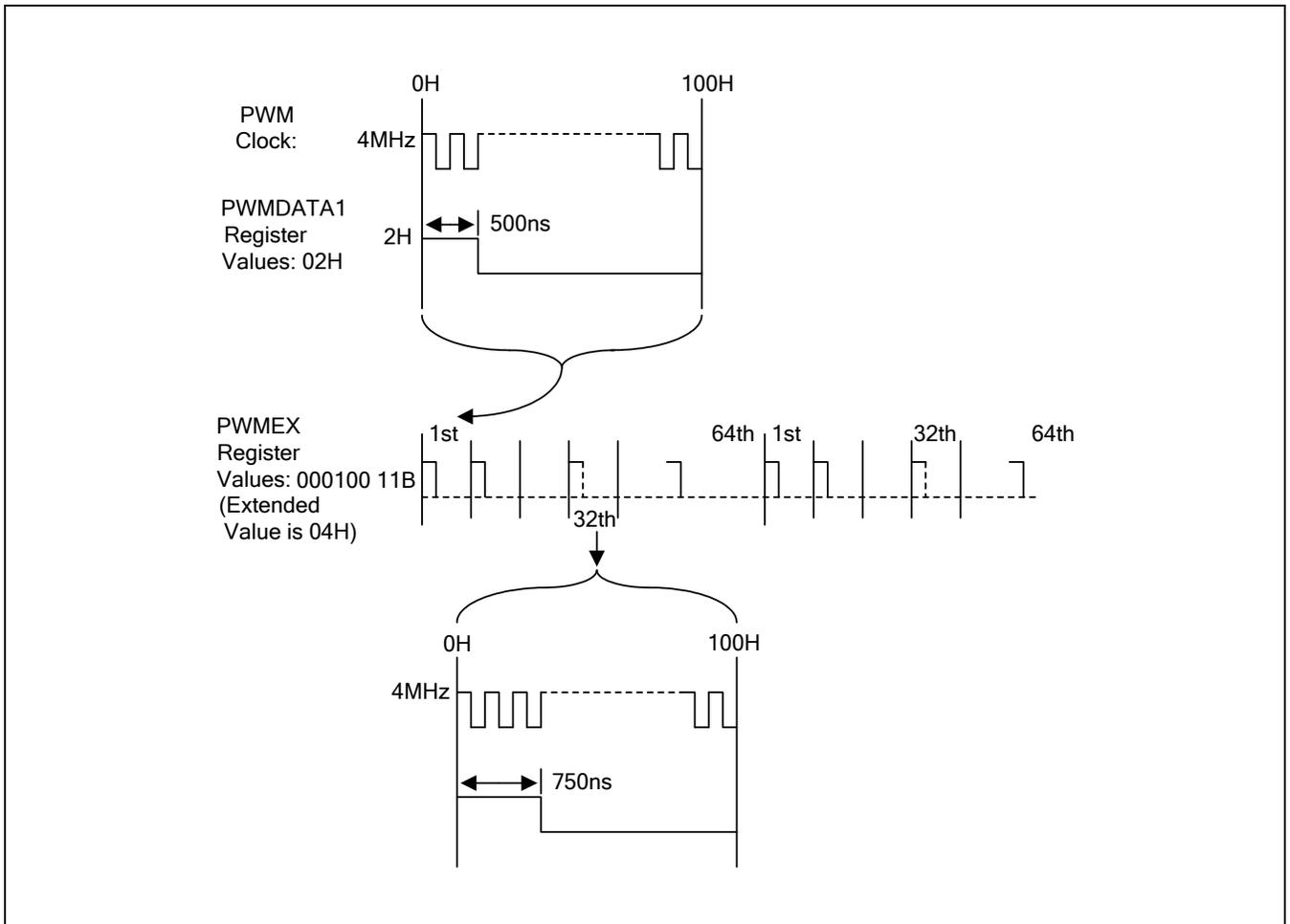**Figure 11-6. PWM Basic Waveform (8-bit base)**

Figure 11-7. PWM Basic Waveform (8-bit base + 6-bit extension)

11.2.7 PWM Control Register

The control register for the PWM module, PWMCON, is located at register address F3H. PWMCON is used for all three PWM resolutions. Bit settings in the PWMCON register control the following functions:

PWM counter clock selection

PWM data reload interval selection

PWM counter clear

PWM counter stop/start (or resume) operation

PWM counter overflow (upper counter overflow) interrupt control

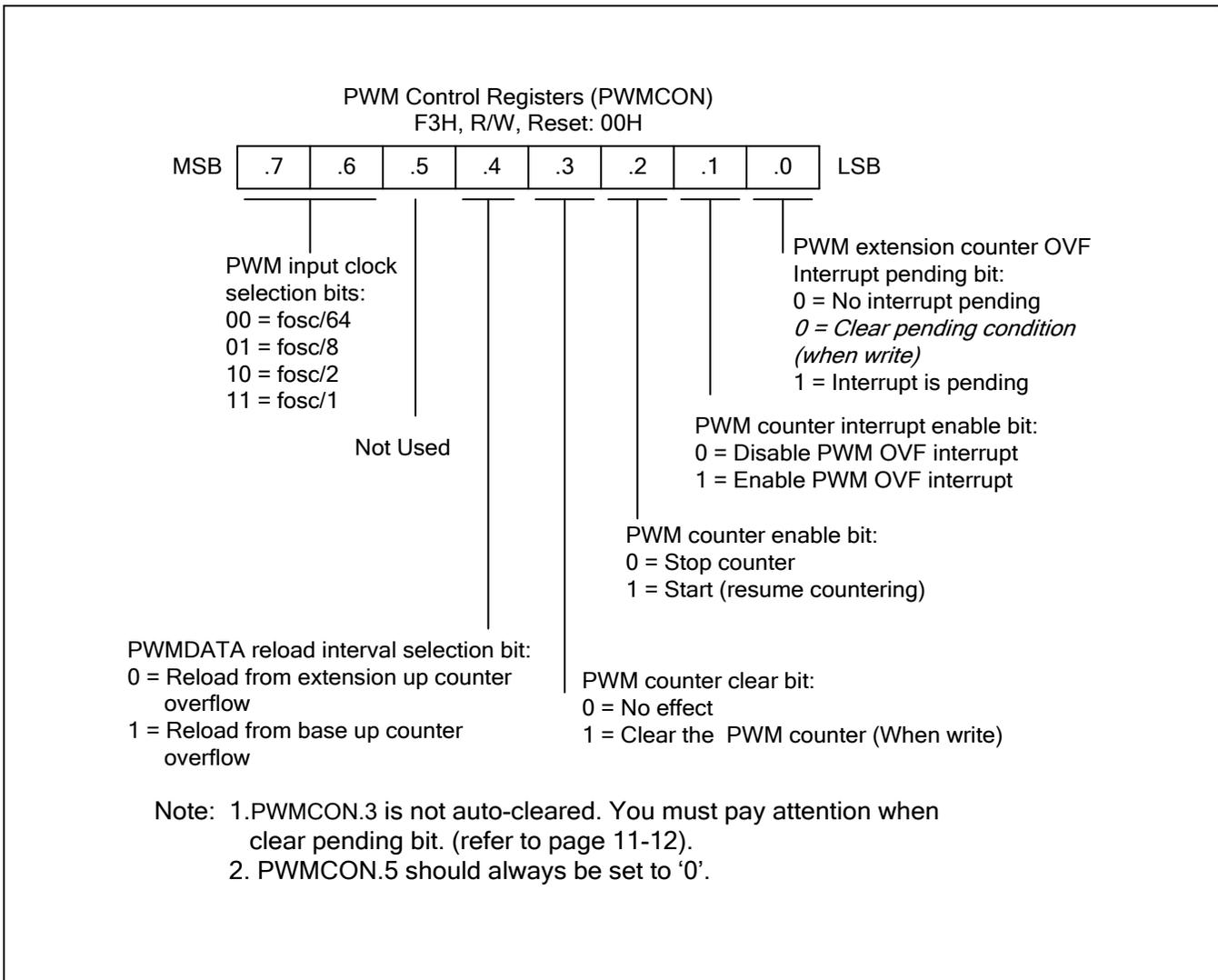A reset clears all PWMCON bits to logic zero, disabling the entire PWM module.

PWM Control Registers (PWMCON)
F3H, R/W, Reset: 00H

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

PWM extension counter OVF
Interrupt pending bit:
0 = No interrupt pending
*0 = Clear pending condition
(when write)*
1 = Interrupt is pending

PWM input clock
selection bits:
00 = fosc/64
01 = fosc/8
10 = fosc/2
11 = fosc/1

PWM counter interrupt enable bit:
0 = Disable PWM OVF interrupt
1 = Enable PWM OVF interrupt

Not Used

PWM counter enable bit:
0 = Stop counter
1 = Start (resume countering)

PWMDATA reload interval selection bit:
0 = Reload from extension up counter
       overflow
1 = Reload from base up counter
       overflow

PWM counter clear bit:
0 = No effect
1 = Clear the PWM counter (When write)

Note: 1.PWMCON.3 is not auto-cleared. You must pay attention when
            clear pending bit. (refer to page 11-12).
          2. PWMCON.5 should always be set to '0'.

Figure 11-8. PWM Control Register (PWMCON)

11.2.8 PWM Extension Register (PWMex)

The extension register for the PWM module, PWMEX, is located at register address F1H. PWMEX are used for resolution selection and extension bits of 6+6 and 8+6 resolution. Bit settings in the PWMEX register control the

following functions:

PWM Extension bits for 6+6 resolution and 8+6 resolution mode

PWM resolution selection

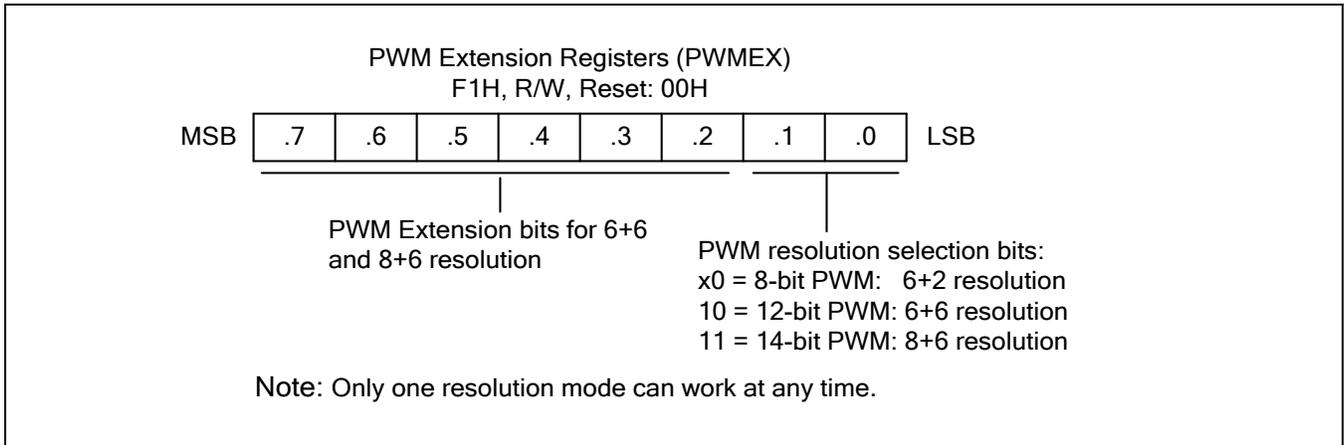A reset clears all PWMEX bits to logic zero, choose 6+2 as default resolution, no extension.
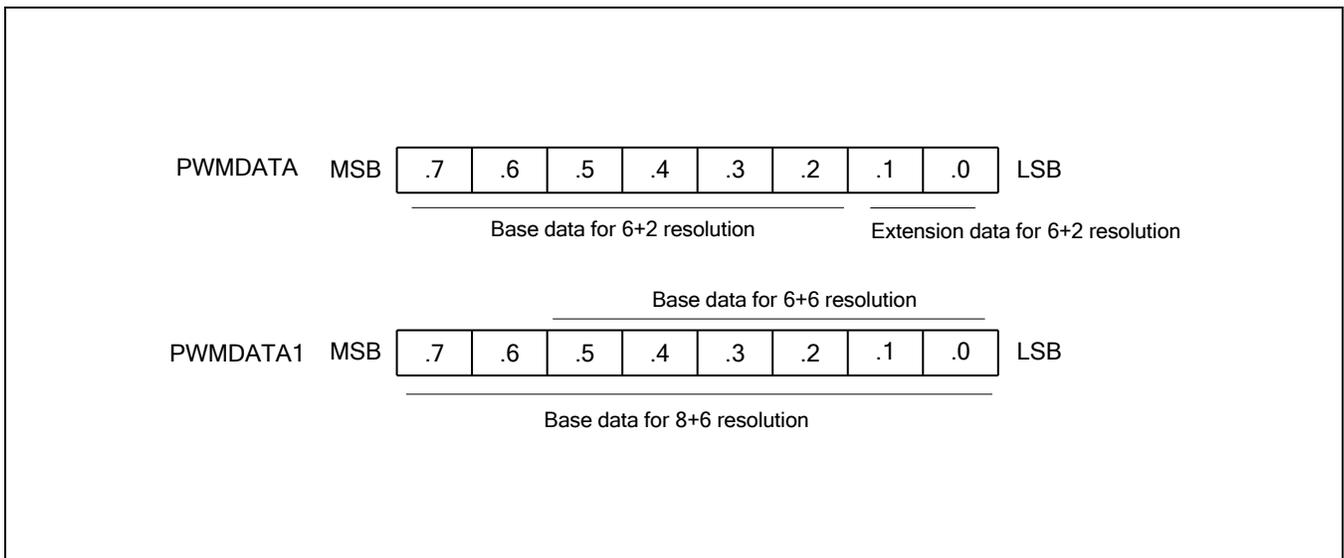

Figure 11-9. PWM Extension Register (PWMEX)

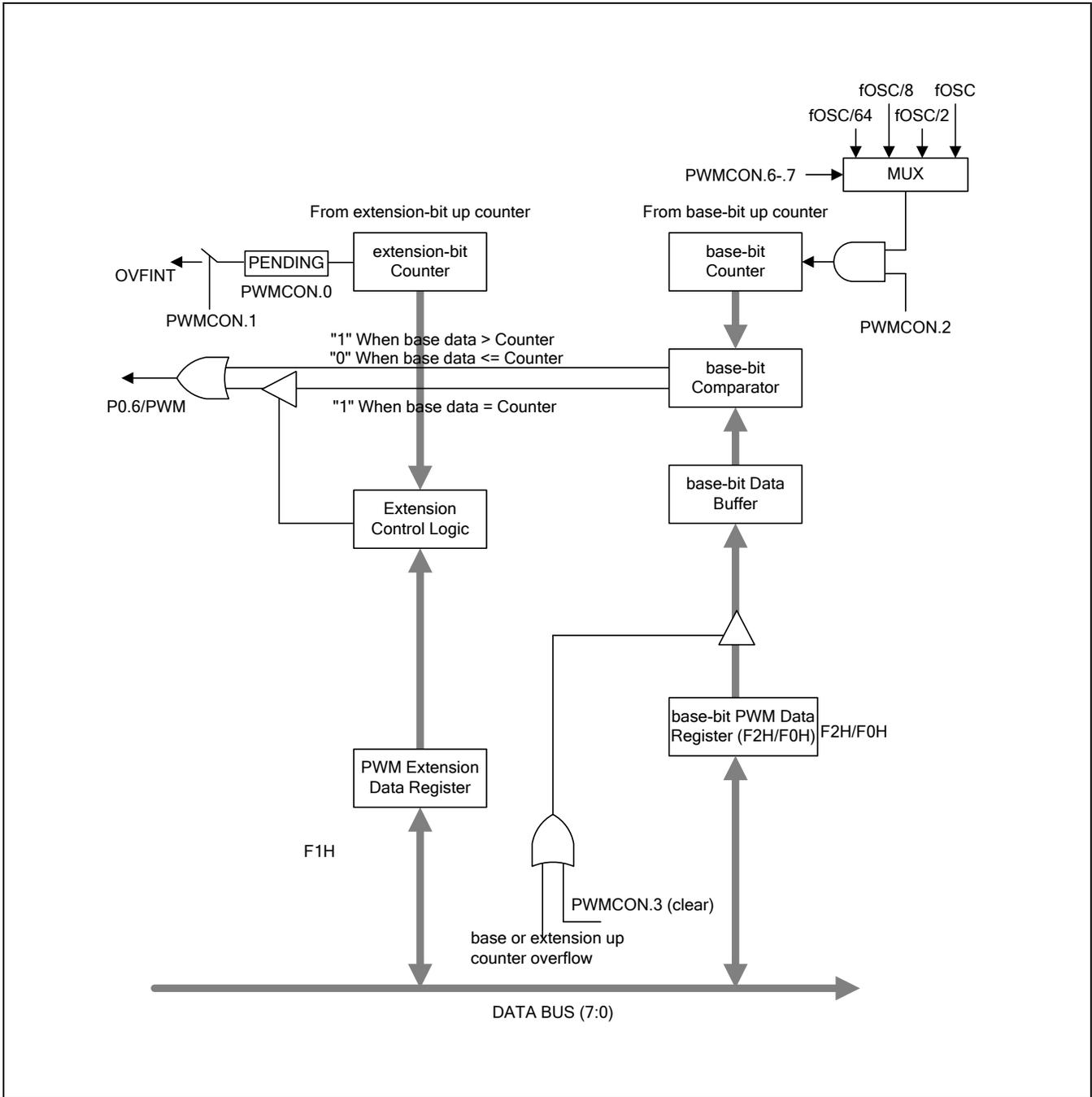**Figure 11-10. PWM Data Register (PWMDATA)**

Figure 11-12.   PWM Module Functional Block Diagram

☞   **PROGRAMMING TIP — Programming the PWM Module to Sample Specifications**

```
;--------------<< Interrupt Vector Address >>
                VECTOR      00H, INT_94C4              ;   S3F94C8/S3F94C4 interrupt vector


   ;-------------<< Smart Option >>
                ORG         003CH
                DB          000H                       ;   003CH, must be initialized to 1.
                DB          000H                       ;   003DH, must be initialized to 1.
                DB          0FFH                       ;   003EH, Enable LVR (2.3)
                DB          000H                       ;   003FH, External Crystal oscillator



;--------------<< Initialize System and Peripherals >>
                ORG         0100H
RESET:          DI                                     ;   disable interrupt
                LD          BTCON,#10100011B           ;   Watchdog disable
                •
                •


                LD          PWMEX,#00000000B           ;    Configure PWM as 6-bit base +2-bit extension
                LD          P0CONH,#10011010B          ;   Configure P0.6 PWM output
                LD          PWMCON,#00000110B          ;   f_OSC/64, counter/interrupt enable
                AND         PWMEX,#00000011B           ;    set extension bits as 00(basic output)
                   LD          PWMDATA,#80H     ;
                •
                •
                EI                                     ;   Enable interrupt

;--------------<< Main loop >>
MAIN:                                                  ;
                •                                      ;
                •                                      ;
                •                                      ;
                •                                      ;
                JR          t,MAIN                     ;

INT_94C4:                                              ;   94C4 interrupt service routine
                •
                •
                •
                AND         PWMCON,#11110110B          ;   pending bit clear
                IRET                                   ;
                •
                •
                END
```

12 A/D Converter

12.1 Overview

The 10-bit A/D converter (ADC) module uses successive approximation logic to convert analog levels entering at one of the nine input channels to equivalent 10-bit digital values. The analog input level must lie between the V_{DD} and V_{SS} values. The A/D converter has the following components:

Analog comparator with successive approximation logic

- D/A converter logic
- ADC Control Register (ADCON)
- Nine multiplexed analog data input pins (ADC0–ADC8)

10-bit A/D conversion data output register (ADDATAH/L)

To initiate an analog-to-digital conversion procedure, you write the channel selection data in the A/D Converter Control Register, ADCON, to select one of the nine analog input pins (ADCn, n = 0–8) and set the conversion start or enable bit, ADCON.0. The read-write ADCON register is located at address F7H.

During a normal conversion, ADC logic initially sets the successive approximation register to 200H (the approximate half-way point of a 10-bit register). This register is then updated automatically during each conversion step. The successive approximation block performs 10-bit conversions for one input channel at a time. You can dynamically select different channels by manipulating the channel selection bit value (ADCON.7–4) in the ADCON register. To start the A/D conversion, you should set a the enable bit, ADCON.0. When a conversion is completed, ACON.3, the end-of-conversion (EOC) bit is automatically set to 1 and the result is dumped into the ADDATA register where it can be read. The A/D converter then enters an idle state. Remember to read the contents of ADDATA before another conversion starts. Otherwise, the previous result will be overwritten by the next conversion result.

NOTE

Because the ADC does not use sample-and-hold circuitry, it is important that any fluctuations in the analog level at the ADC0–ADC8 input pins during a conversion procedure be kept to an absolute minimum. Any change in the input level, perhaps due to circuit noise, will invalidate the result.

12.1.1 Using A/D Pins for Standard digital Input

The ADC module's input pins are alternatively used as digital input in Port 0 and P2.6.

12.1.2 A/D Converter Control Register (ADCON)

The A/D Converter Control Register, ADCON, is located at address F7H. ADCON has four functions:

Bits 7–4 select an analog input pin (ADC0–ADC8)

Bit 3 indicates the status of the A/D conversion

Bits 2–1 select a conversion speed

Bit 0 starts the A/D conversion

Only one analog input channel can be selected at a time. You can dynamically select any one of the nine analog input pins (ADC0–ADC8) by manipulating the 4-bit value for ADCON.7–ADCON.4.
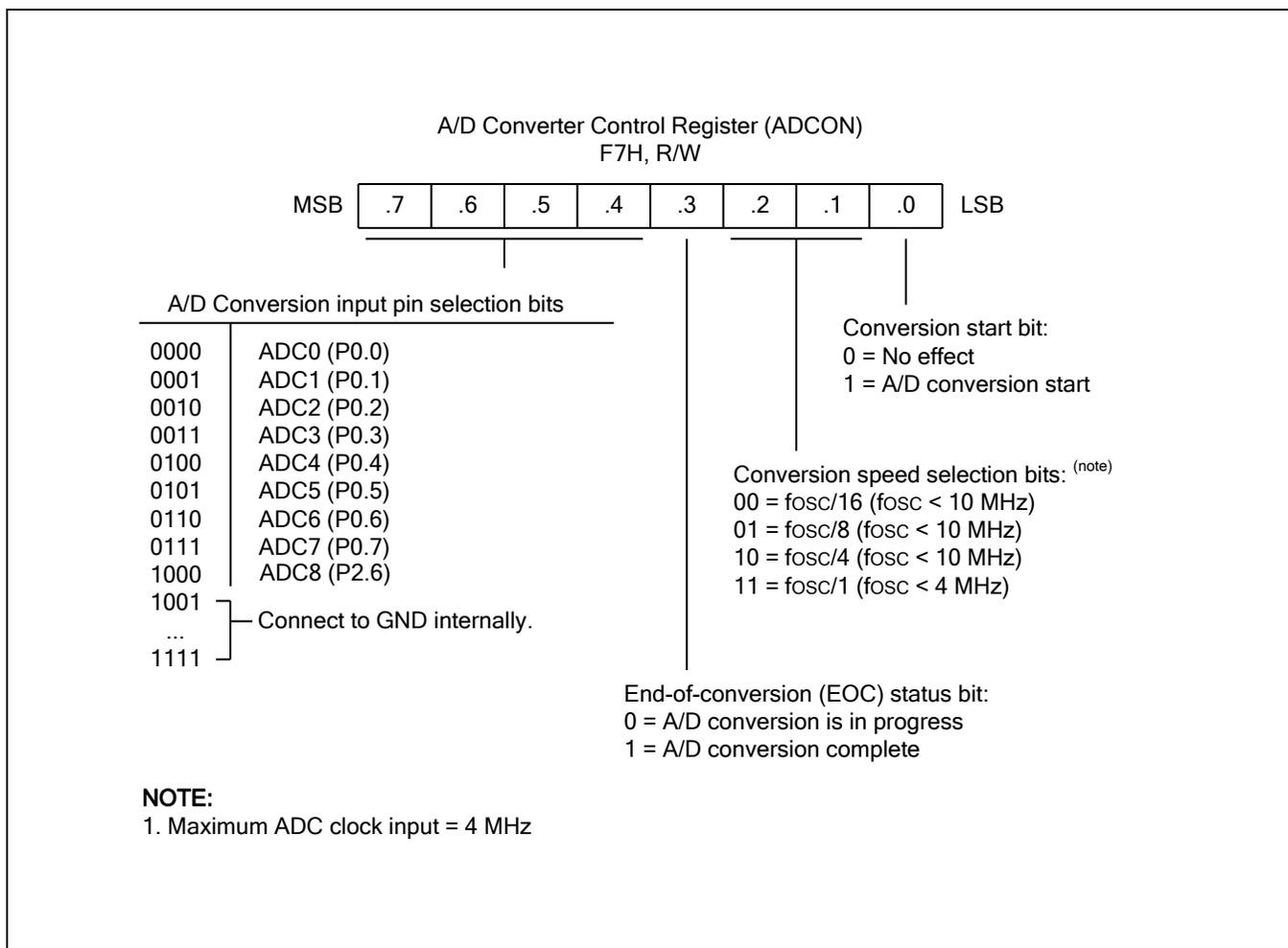
A/D Converter Control Register (ADCON)
F7H, R/W

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

A/D Conversion input pin selection bits

0000	ADC0 (P0.0)
0001	ADC1 (P0.1)
0010	ADC2 (P0.2)
0011	ADC3 (P0.3)
0100	ADC4 (P0.4)
0101	ADC5 (P0.5)
0110	ADC6 (P0.6)
0111	ADC7 (P0.7)
1000	ADC8 (P2.6)
1001	
...	Connect to GND internally.
1111	

Conversion start bit:
0 = No effect
1 = A/D conversion start

Conversion speed selection bits: [note]
00 = fosc/16 (fosc < 10 MHz)
01 = fosc/8 (fosc < 10 MHz)
10 = fosc/4 (fosc < 10 MHz)
11 = fosc/1 (fosc < 4 MHz)

End-of-conversion (EOC) status bit:
0 = A/D conversion is in progress
1 = A/D conversion complete

NOTE:
1. Maximum ADC clock input = 4 MHz

Figure 12-1. A/D Converter Control Register (ADCON)

12.1.3 Internal Reference Voltage Levels

In the ADC function block, the analog input voltage level is compared to the reference voltage. The analog input level must remain within the range V_{SS} to V_{DD}.

Different reference voltage levels are generated internally along the resistor tree during the analog conversion process for each conversion step. The reference voltage level for the first bit conversion is always 1/2 V_{DD}.
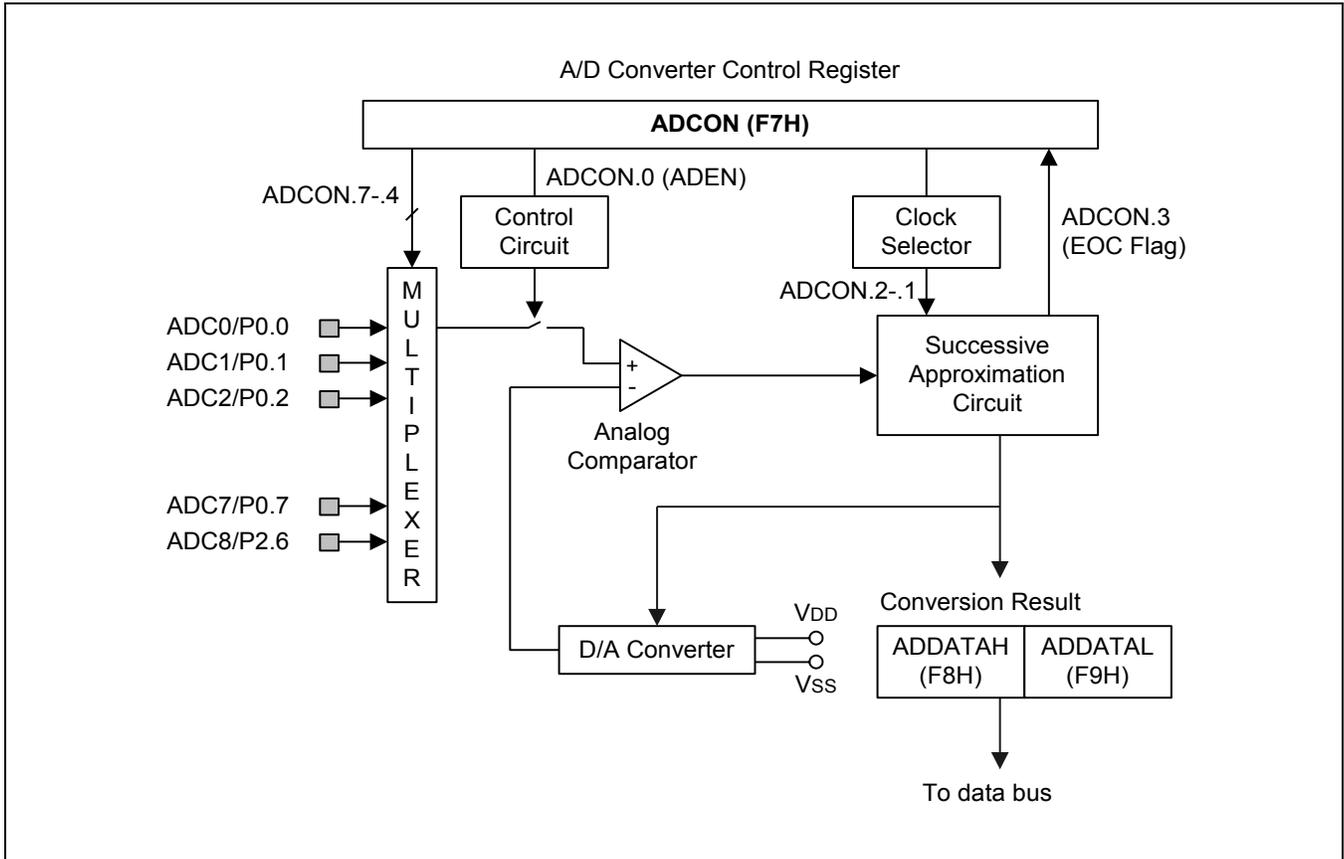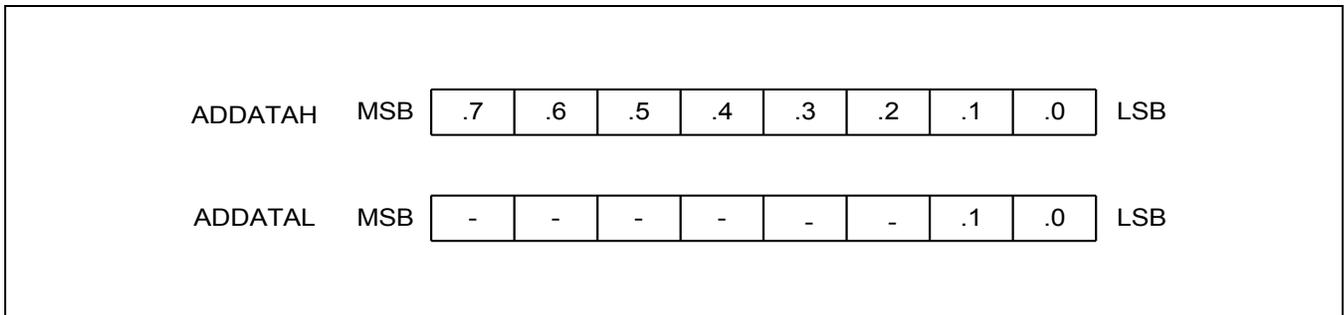

Figure 12-2. A/D Converter Circuit Diagram

Figure 12-3. A/D Converter Data Register (ADDATAH/L)
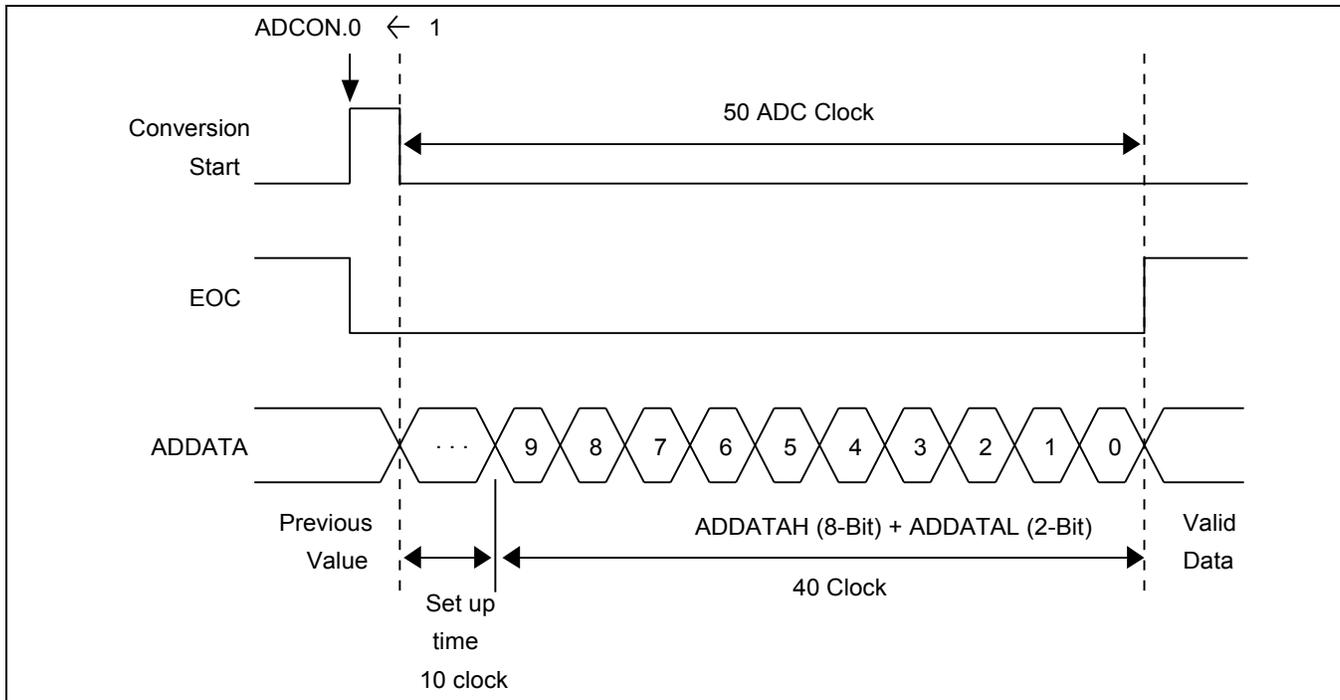
Figure 12-4. A/D Converter Timing Diagram

12.1.4 Conversion timing

The A/D conversion process requires 4 steps (4 clock edges) to convert each bit and 10 clocks to step-up A/D conversion. Therefore, total of 50 clocks is required to complete a 10-bit conversion: With a 10 MHz CPU clock frequency, one clock cycle is 400 ns (4/fxx). If each bit conversion requires 4 clocks, the conversion rate is calculated as follows:

4 clocks/bit x 10-bits + step-up time (10 clock) = 50 clocks
50 clock x 400 ns = 20 μs at 10 MHz, 1 clock time = 4/fxx (assuming ADCON.2–.1 = 10)

12.1.5 Internal A/D Conversion Procedure

1. Analog input must remain between the voltage range of V_{SS} and V_{DD}.

2. Configure the analog input pins to input mode by making the appropriate settings in P0CONH, P0CONL and P2CONH registers.

3. Before the conversion operation starts, you must first select one of the nine input pins (ADC0–ADC8) by writing the appropriate value to the ADCON register.

4. When conversion has been completed, (50 clocks have elapsed), the EOC flag is set to 1, so that a check can be made to verify that the conversion was successful.

5. The converted digital value is loaded to the output register, ADDATAH (8-bit) and ADDATAL (2-bit), and then the ADC module enters an idle state.

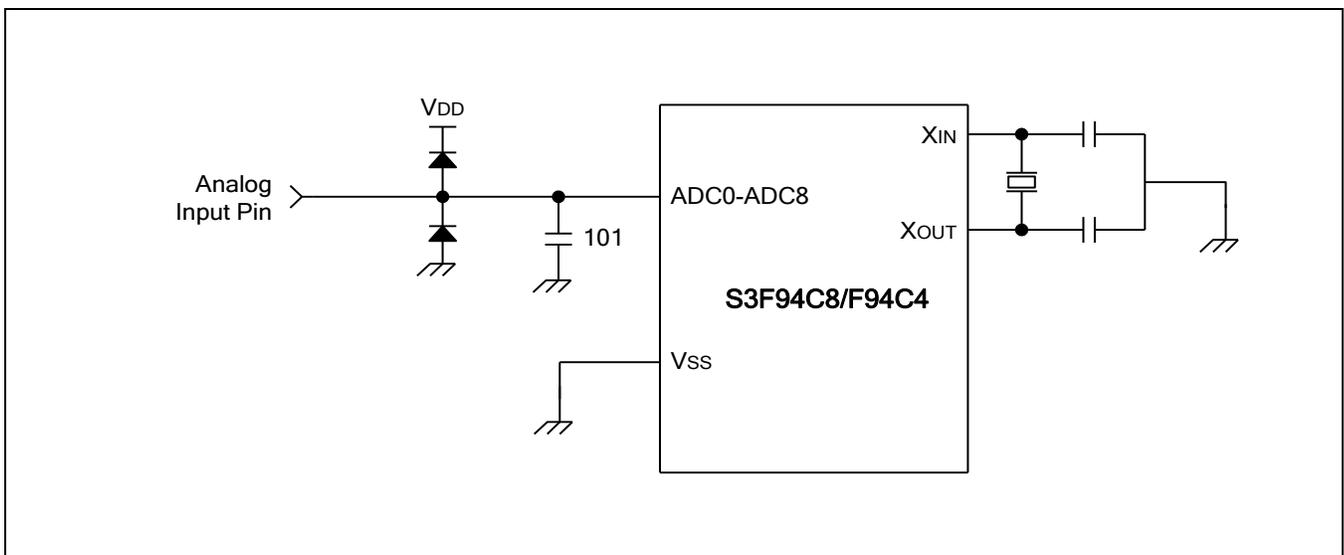6. The digital conversion result can now be read from the ADDATAH and ADDATAL registers.

Figure 12-5. Recommended A/D Converter Circuit for Highest Absolute Accuracy

☞   **PROGRAMMING TIP – Configuring A/D Converter**

```
;----------------<< Interrupt Vector Address >>
             VECTOR      00H, INT_TIMER 0         ;  S3F94C8/S3F94C4 has only one interrupt vector
;-------------<< Smart Option >>
             ORG      003CH
             DB       000H                        ;  003CH, must be initialized to 0
             DB       000H                        ;  003DH, must be initialized to 0
             DB       0FFH                        ;  003EH, enable LVR
             DB       003H                        ;  003FH, internal RC oscillator
             ORG      0100H
RESET:       DI                                   ;  disable interrupt
             LD       BTCON,#10100011B            ;  Watchdog disable
             •
             •

             •
             LD       P0CONH,#11111111B           ;  Configure P0.4–P0.7 AD input
             LD       P0CONL,#11111111B           ;  Configure P0.0–P0.3 AD input
             LD       P2CONH,#00100000B           ;  Configure P2.6 AD input
             EI                                   ;  Enable interrupt
;-------------<< Main loop >>
MAIN:        •
             •

             •
             CALL     AD_CONV                     ;  Subroutine for AD conversion
             •
             •

             •
             JR       t, MAIN                     ;

AD_CONV:     LD       ADCON, #00000001B           ;  Select analog input channel  →   P0.0
                                                  ;  select conversion speed    →   f_OSC/16
                                                  ;  set conversion start bit

             NOP
                                                  ;  If you select conversion speed to f_OSC/16
                                                  ;  at least one NOP must be included
```

☞ **PROGRAMMING TIP – Configuring A/D Converter (Continued)**

```
CONV_LOOP:  TM     ADCON,#00001000B        ;  Check EOC flag
            JR     Z,CONV_LOOP             ;  If EOC flag=0, jump to CONV_LOOP until EOC flag=1
            LD     R0,ADDATAH              ;  High 8 bits of conversion result are stored
                                           ;  to ADDATAH register
            LD     R1,ADDATAL              ;  Low 2 bits of conversion result are stored
                                           ;  to ADDATAL register
            LD     ADCON,#00010011B        ;  Select analog input channel   →  P0.1
                                           ;  Select conversion speed   →  fOSC/8
                                           ;  Set conversion start bit
CONV_LOOP2:TM     ADCON,#00001000B         ;  Check EOC flag
            JR     Z,CONV_LOOP2
            LD     R2,ADDATAH
            LD     R3,ADDATAL
            •
            •
            •
            RET                            ;

INT_TIMER 0:  •                            ;
              •                            ;
              •                            ;  Pending bit clear
              IRET                         ;
              •
              •
              END
```

13

Embedded Flash Memory Interface

13.1   Overview

The S3F94C8/S3F94C4 has an on-chip Flash memory internally instead of masked ROM. The Flash memory is accessed by instruction 'LDC'. This is a sector erasable and a byte programmable Flash. User can program the data in a Flash memory area any time you want. The S3F94C8/S3F94C4's embedded 8K/4K memory offers the following two operating features:

Tool Program Mode: Refer to the chapter 16. S3F94C8/S3F94C4 Flash MCU

User Program Mode

13.1.1 Flash ROM Configuration

The S3F94C8/S3F94C4 Flash memory consists of 64 sectors (S3F94C8) or 32sectors (S3F94C4). Each sector consists of 128bytes. Therefore, the total size of Flash memory is 64 x128 (8KB) or 32x128 bytes (4KB). User can erase the Flash memory by a sector unit at a time and write the data into the Flash memory by a byte unit at a time.

8K/4KB Internal Flash memory

Sector size: 128 bytes

10 years data retention

Fast programming Time:

■   Sector Erase: 8ms (min)

■   Byte Program: 25us (min)

Byte programmable

User programmable by 'LDC' instruction

Sector (128 bytes) erase available

Endurance: 10,000 Erase/Program cycles (min)

13.1.2 Tool Program Mode

This mode is for erasing and programming full area of Flash memory by external programming tools. The 5 pins of S3F94C8/S3F94C4 are connected to a programming tool and then internal Flash memory of S3F94C8/S3F94C4 can be programmed by Serial OTP/MTP Tools, SPW2 plus single programmer or GW-PRO2 gang programmer and so on. The other modules except Flash memory module are at a reset state. This mode doesn't support the sector erase but chip erase (all Flash memory erased at a time) and two protection modes (Hard lock protection/Read protection). The read protection mode is available only in tool program mode. So in order to make a chip into read protection, you need to select a read protection option when you write a program code to a chip in tool program mode by using a programming tool. After read protect, all data of Flash memory read "00". This protection is released by chip erase execution in the tool program mode.

Table 13-1. Descriptions of Pins Used to Read/Write the Flash in Tool Program Mode

Main Chip Pin Name	During Programming			
	Pin Name	Pin No.	I/O	Function
P0.1	SDAT	18 (20-pin) 14 (16-pin)	I/O	Serial data pin (output when reading, Input when writing) Input and push-pull output port can be assigned
P0.0	SCLK	19 (20-pin) 15 (16-pin)	I	Serial clock pin (input only pin)
RESET/P1.2	V$_{PP}$	4	I	Power supply pin for Tool mode entering (indicates that MTP enters into the Tool mode). When 11 V is applied, MTP is in Tool mode.
V$_{DD}$/V$_{SS}$	V$_{DD}$/V$_{SS}$	20 (20-pin), 16 (16-pin) 1 (20-pin), 1 (16-pin)	I	Logic power supply pin.

13.1.3 User Program Mode

This mode supports sector erase, byte programming, byte read and one protection mode (Hard Lock Protection). The S3F94C8/S3F94C4 has the internal pumping circuit to generate high voltage. To program a Flash memory in this mode several control registers will be used.

There are four kind functions in user program mode – programming, reading, sector erase, and one protection mode (Hard lock protection).

13.2   Flash Memory Control Registers, User Program Mode

13.2.1 Flash Memory Control Register

The 13.2.1 Flash Memory Control Register (FMCON) Register is available only in user program mode to select the Flash memory operation mode; sector erase, byte programming, and to make the Flash memory into a hard lock protection.
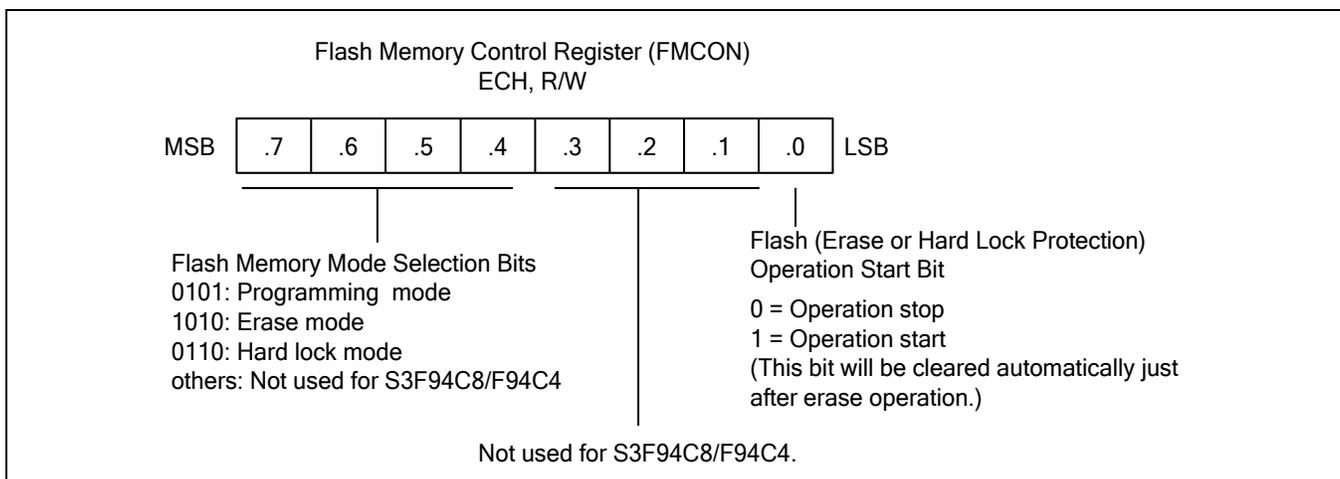

Figure 13-1. Flash Memory Control Register (FMCON)

The bit 0 of FMCON register (FMCON.0) is a bit for the operation start of Erase and Hard Lock Protection. Therefore, operation of Erase and Hard Lock Protection is activated when setting FMCON.0 to 1. If you write FMCON.0 to 1 for erasing, CPU is stopped automatically for erasing time (min.4ms). After erasing time, CPU is restarted automatically. When you read or program a byte data from or into Flash memory, this bit is not needed to manipulate.

13.2.2 Flash Memory User Programming Enable Register

The 13.2.2 Flash Memory User Programming Enable (FMUSR) Register is used for a safe operation of the Flash memory. This register will protect undesired erase or program operation from malfunctioning of CPU caused by an electrical noise. After reset, the user-programming mode is disabled, because the value of FMUSR is "00000000B" by reset operation. If necessary to operate the Flash memory, you can use the user programming mode by setting the value of FMUSR to "10100101B". The other value of "10100101B", user program mode is disabled.
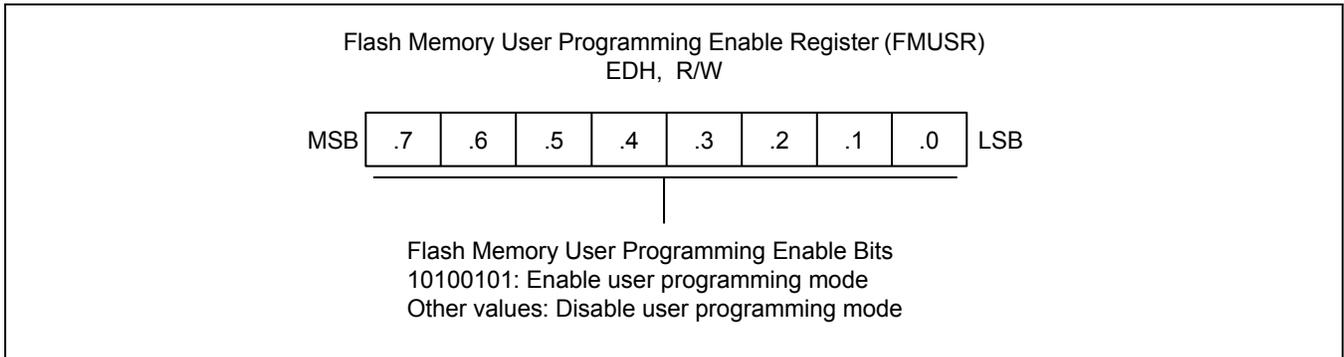
Flash Memory User Programming Enable Register (FMUSR)
EDH, R/W

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

Flash Memory User Programming Enable Bits
10100101: Enable user programming mode
Other values: Disable user programming mode

Figure 13-2. Flash Memory User Programming Enable Register (FMUSR)

13.2.3 Flash Memory Sector Address Registers

There are two sector address registers for the erase or programming Flash memory. The Flash Memory Sector Address Register Low Byte (FMSECL) Register indicates the low byte of the sector address, and the Flash Memory Address Sector Register High Byte (FMSECH) Register indicates the high byte of the sector address. The FMSECH is required for the S3F94C8/S3F94C4 MCUs because it has 64/32 sectors.

One sector consists of 128 bytes. Each sector's address starts XX00H or XX80H, that is, a base address of sector is XX00H or XX80H. So bit .6-.0 of FMSECL don't mean whether the value is 1 or 0. We recommend that it is the simplest way to load the sector base address into FMSECH and FMSECL register. When programming the Flash memory, user should program after loading a sector base address, which is located in the destination address to write data into FMSECH and FMSECL registers. If the next operation is also to write one byte data, user should check whether next destination address is located in the same sector or not. In case of other sectors, user should load sector address to FMSECH and FMSECL Register according to the sector. (Refer to page 13-12 PROGRAMMING TIP — Programming)
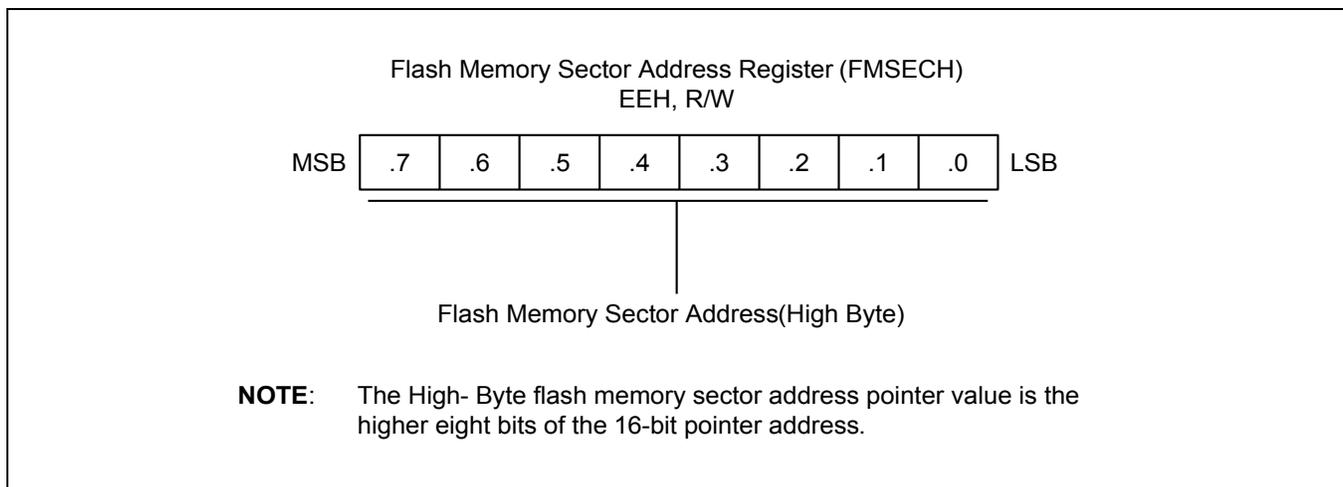
Flash Memory Sector Address Register (FMSECH)
EEH, R/W

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

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.

Figure 13-3. Flash Memory Sector Address Register (FMSECH)

Flash Memory Sector Address Register (FMSECL)
EFH, R/W

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

Don't Care

Flash Memory Sector Address(Low Byte)

NOTE: The Low- Byte flash memory sector address pointer value is the lower eight bits of the 16-bit pointer address.
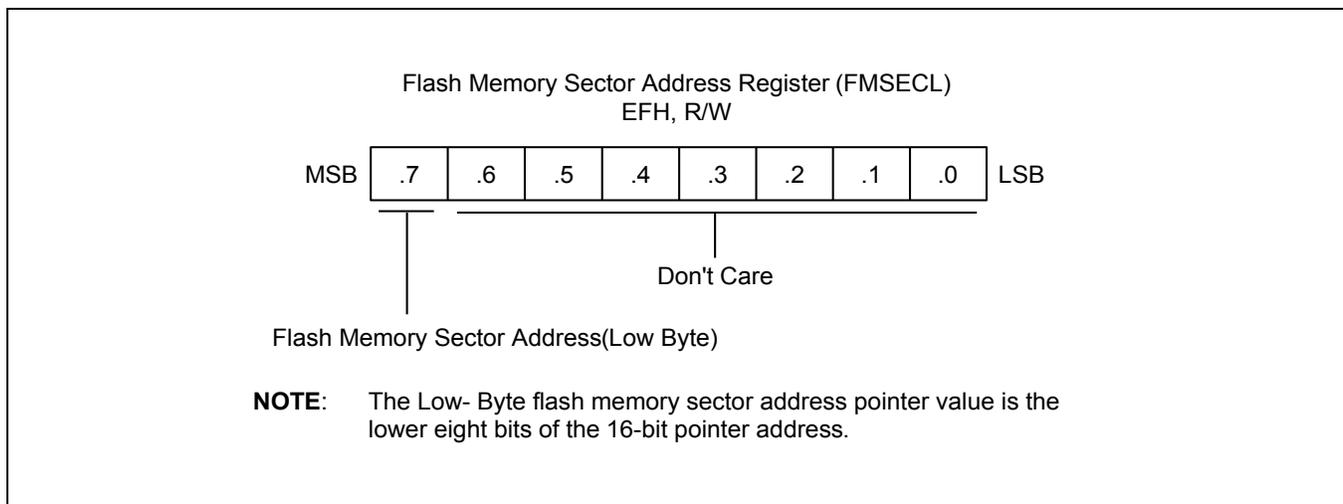
Figure 13-4. Flash Memory Sector Address Register (FMSECL)

13.3   Sector Erase

User can erase a Flash memory partially by using sector erase function only in user program mode. The only unit of Flash memory to be erased in the user program mode is a sector.

The program memory of S3F94C8/S3F94C4 8K/4KB Flash memory is divided into 64/32 sectors. Every sector has all 128-byte sizes. So the sector to be located destination address should be erased first to program a new data (one byte) into Flash memory. Minimum 4ms' 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 (MDS mode tool or programming tool).
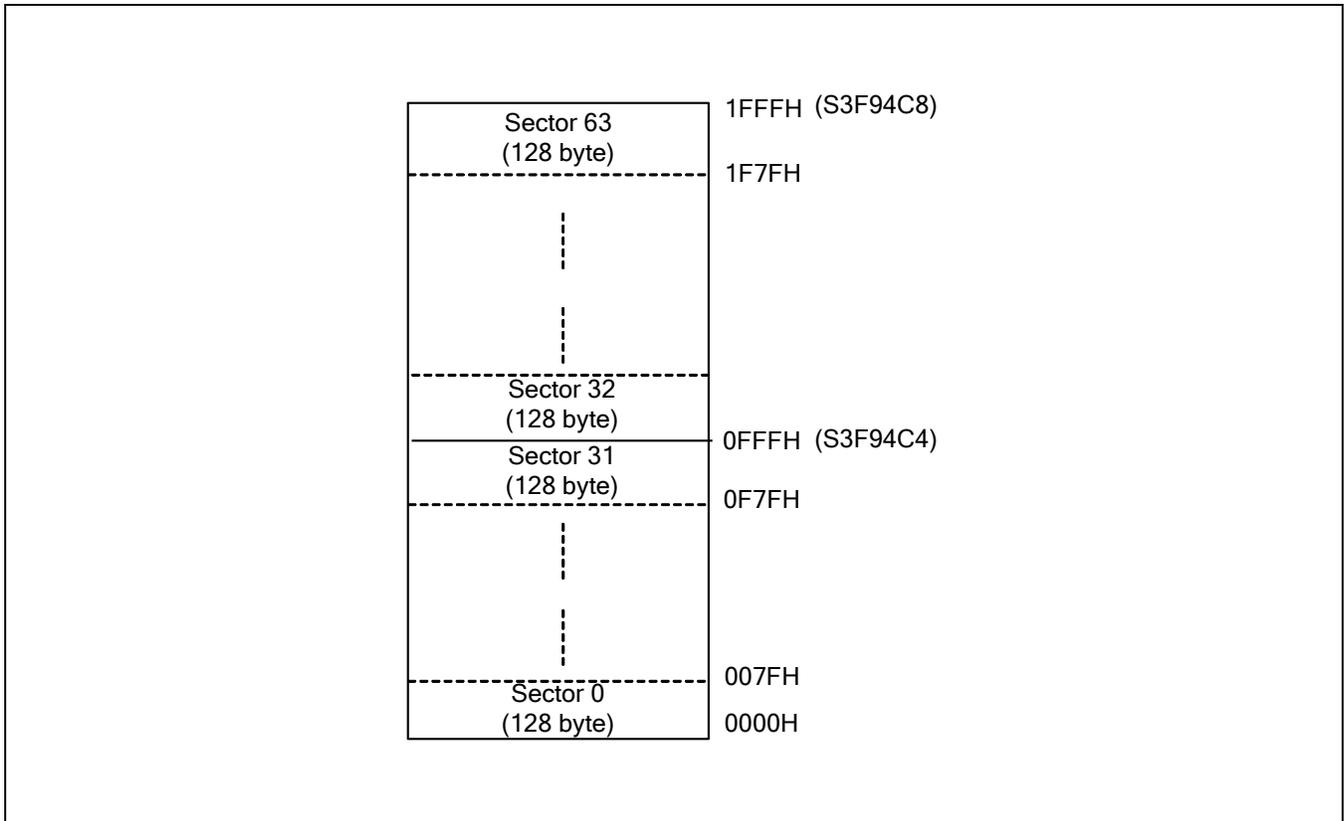

Figure 13-5. Sector Configurations in User Program Mode

The Sector Erase Procedure in User Program Mode

1.   Set Flash Memory User Programming Enable Register (FMUSR) to "10100101B".

2.   Set Flash Memory Sector Address Register (FMSECH and FMSECL).

3.   Set Flash Memory Control Register (FMCON) to "10100001B".

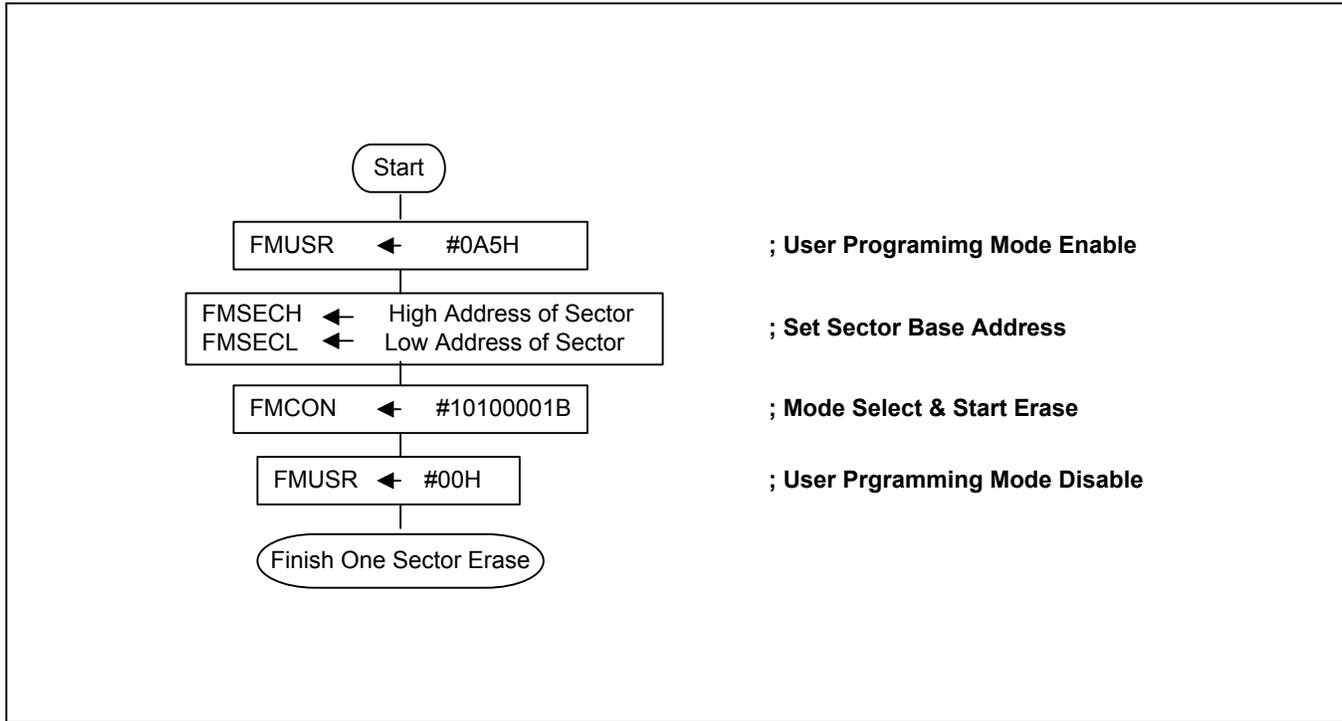4.   Set Flash Memory User Programming Enable Register (FMUSR) to "00000000B".

```
                    ┌─────────┐
                    │  Start  │
                    └─────────┘
                         │
        ┌─────────────────────────────┐
        │  FMUSR    ◄      #0A5H       │     ; User Programimg Mode Enable
        └─────────────────────────────┘
                         │
     ┌─────────────────────────────────────┐
     │  FMSECH  ◄   High Address of Sector  │  ; Set Sector Base Address
     │  FMSECL  ◄   Low Address of Sector   │
     └─────────────────────────────────────┘
                         │
         ┌─────────────────────────────┐
         │  FMCON   ◄     #10100001B    │    ; Mode Select & Start Erase
         └─────────────────────────────┘
                         │
           ┌─────────────────────────┐
           │  FMUSR  ◄  #00H          │      ; User Prgramming Mode Disable
           └─────────────────────────┘
                         │
          ╭───────────────────────────╮
          │  Finish One Sector Erase  │
          ╰───────────────────────────╯
```

Figure 13-6. Sector Erase Flowchart in User Program Mode

NOTES

If the user erases a sector selected by Flash Memory Sector Address Register FMSECH and FMSECL, FMUSR should be enabled just before starting sector erase operation. And to erase a sector, Flash Operation Start Bit of FMCON register is written from operation stop 0 to operation start 1. That bit will be cleared automatically just after the corresponding operation completed. In other words, when S3F94C8/S3F94C4 is in the condition that Flash memory user programming enable bits is enabled and executes start operation of sector erase, it will get the result of erasing selected sector as user's purpose and Flash Operation Start Bit of FMCON register is also clear automatically.

If the user executes sector erase operation with FMUSR disabled, FMCON.0 bit, Flash Operation Start Bit, remains 'high', which means start operation, and is not cleared even though next instruction is executed. So user should be careful to set FMUSR when executing sector erase, for no effect on other Flash sectors.

☞   **PROGRAMMING TIP — Sector Erase**

Case1. Erase one sector

-
-

ERASE_ONESECTOR:


```
    LD   FMUSR,#0A5H         ; User program mode enable
    LD        FMSECH,#04H    ; Set sector address 0400H, sector 8,
    LD   FMSECL,#00H         ; among sector 0~32
    LD   FMCON,#10100001B    ; Select erase mode enable & Start sector erase
```

ERASE_STOP:
```
    LD   FMUSR,#00H          ; User program mode disable
```

13.4   Programming

A Flash memory is programmed in one-byte unit after sector erase. The write operation of programming starts by 'LDC' instruction.

13.4.1 Program Procedure In User Program Mode

1. Must erase target sectors before programming.
2. Set Flash Memory User Programming Enable Register (FMUSR) to "10100101B".
3. Set Flash Memory Control Register (FMCON) to "0101000XB".
4. Set Flash Memory Sector Address Register (FMSECH and FMSECL) to the sector base address of destination address to write data.
5. Load a transmission data into a working register.
6. Load a Flash memory upper address into upper register of pair working register.
7. Load a Flash memory lower address into lower register of pair working register.
8. Load transmission data to Flash memory location area on 'LDC' instruction by indirectly addressing mode
9. Set Flash Memory User Programming Enable Register (FMUSR) to "00000000B".

NOTE

In programming mode, it doesn't care whether the value of FMCON.0 is 0 or 1.

```
                        ┌─────────┐
                        │  Start  │
                        └─────────┘
                             │
    ┌────────────────────────────────────────┐
    │ FMSECH  ◄──   High Address of Sector    │        ; Set Secotr Base Address
    │ FMSECL  ◄──   Low Address of Sector     │
    └────────────────────────────────────────┘
                             │
    ┌────────────────────────────────────────┐
    │ R(n)     ◄──   High Address to Write    │
    │ R(n+1)   ◄──   Low Address to Write     │        ; Set Address and Data
    │ R(data)  ◄──   8-bit Data               │
    └────────────────────────────────────────┘
                             │
       ┌───────────────────────────────┐
       │ FMUSR   ◄──    #0A5H           │               ; User Program Mode Enable
       └───────────────────────────────┘
                             │
       ┌───────────────────────────────┐
       │ FMCON   ◄──   #01010000B       │               ; Mode Select
       └───────────────────────────────┘
                             │
       ┌───────────────────────────────┐
       │ LDC    ◄──  @RR(n),R(data)     │               ; Write data at flash
       └───────────────────────────────┘
                             │
       ┌───────────────────────────────┐
       │ FMUSR   ◄──    #00H            │               ; User Program Mode Disable
       └───────────────────────────────┘
                             │
            ┌────────────────────────────┐
            │   Finish 1-BYTE Writing    │
            └────────────────────────────┘
```
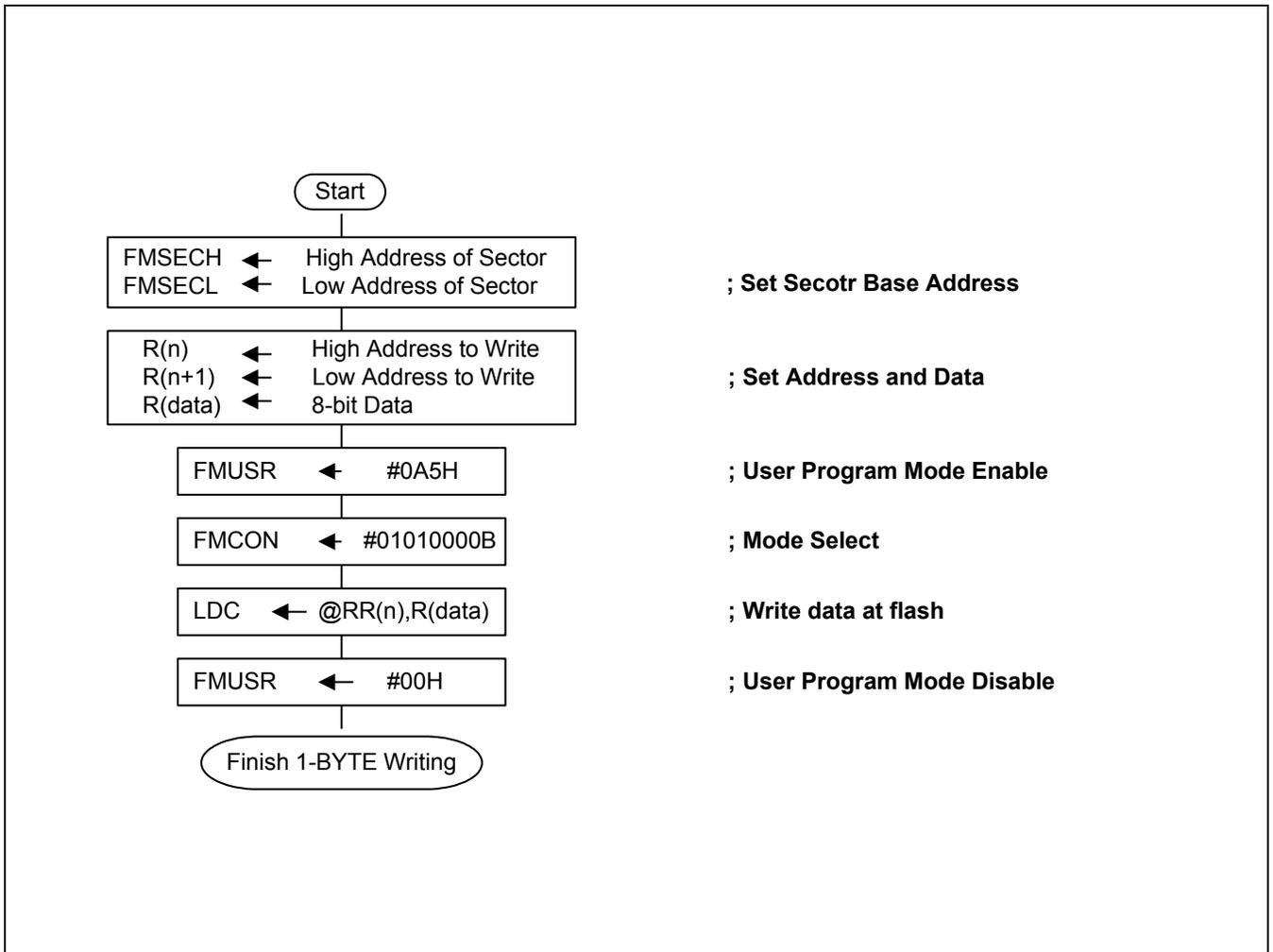
Figure 13-7. Byte Program Flowchart in a User Program Mode
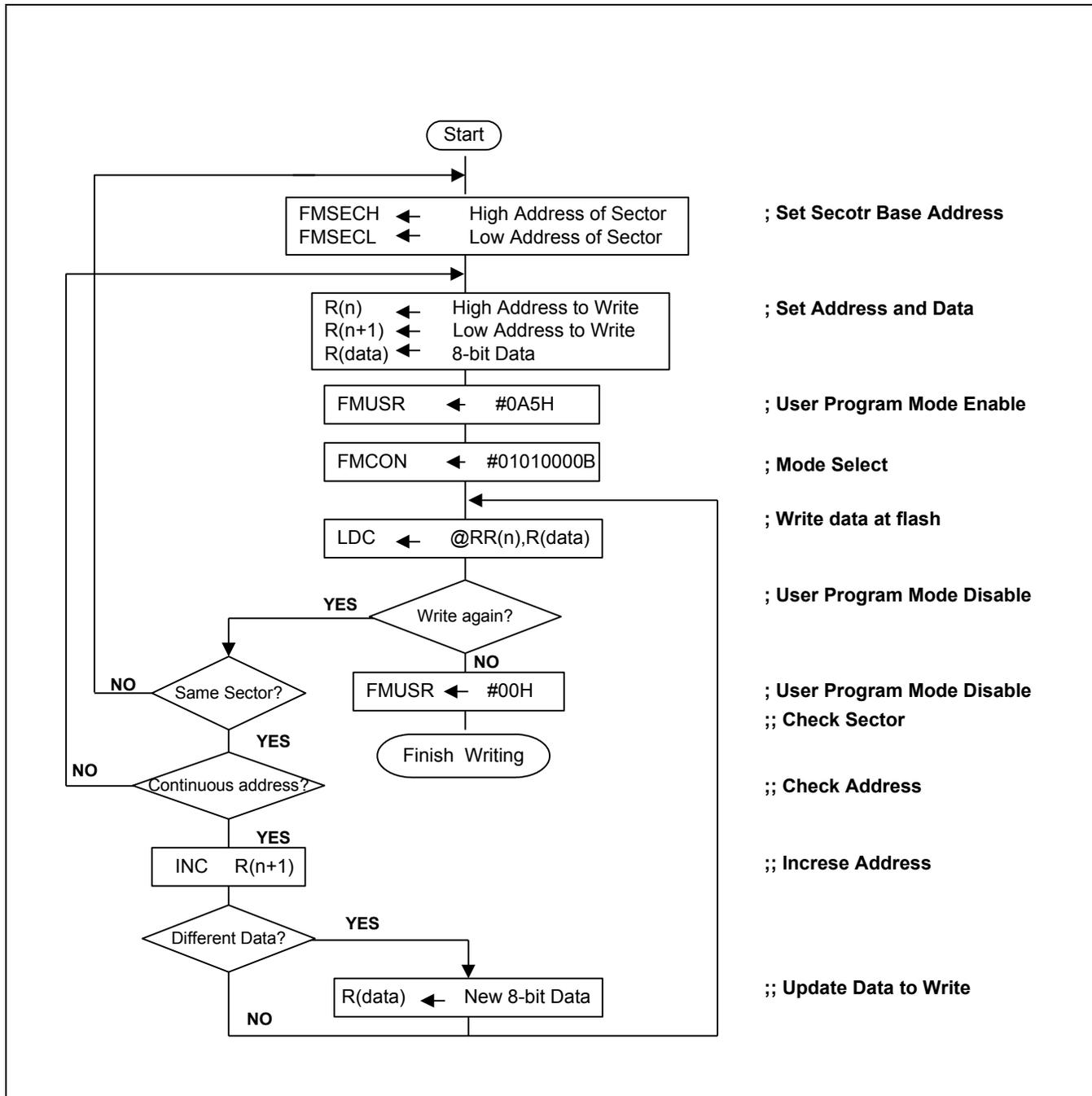
Figure 13-8. Program Flowchart in a User Program Mode

☞   **PROGRAMMING TIP — Programming**

Case1. 1-Byte Programming
•
•
```
WR_BYTE:                              ; Write data "AAH"   to destination address 0310H

    LD      FMUSR,#0A5H               ; User program mode enable
    LD      FMCON,#01010000B  ; Selection programming mode
    LD      FMSECH, #03H             ; Set the base address of sector (0300H)
    LD      FMSECL, #00H
    LD      R9,#0AAH                         ; Load data "AA" to write
    LD      R10,#03H         ; Load Flash memory upper address into upper register of pair working
                ; register
    LD      R11,#10H        ; Load Flash memory lower address into lower register of pair working
                ; register
    LDC     @RR10,R9        ; Write data 'AAH' at Flash memory location (0310H)

    LD      FMUSR,#00H              ; User program mode disable
```

Case2. Programming in the same sector
•
•
```
WR_INSECTOR:                          ; RR10-->Address copy   (R10 –high address,R11-low address)

        LD          R0,#40H



    LD          FMUSR,#0A5H          ; User program mode enable
    LD          FMCON,#01010000B       ; Selection programming mode and Start programming
    LD          FMSECH,#06H                ; Set the base address of sector located in target address to
write data
    LD          FMSECL,#00H                ; The sector 12's base address is 0600H.
    LD          R9,#33H                       ; Load data "33H" to write
    LD          R10,#06H          ; Load Flash memory upper address into upper register of pair working
                ; register
    LD      R11,#00H          ; Load Flash memory lower address into lower register of pair working
                ; register
WR_BYTE:
    LDC@RR10,R9             ; Write data '33H' at Flash memory location
    INC     R11                        ; Reset address in the same sector by INC instruction
      DEC             R0
    JP              NZ，WR_BYTE     ; Check whether the end address for programming reach 0640H or not.

    LD      FMUSR,#00H               ; User Program mode disable
```

Case3. Programming to the Flash memory space located in other sectors

•
•

```
WR_INSECTOR2:
    LD      R0,#40H
    LD      R1,#40H


    LD      FMUSR,#0A5H          ; User program mode enable
    LD      FMCON,#01010000B     ; Selection programming mode and Start programming
    LD      FMSECH,#01H          ; Set the base address of sector located in target address to write data
    LD      FMSECL,#00H          ; The sector 2's base address is 100H
    LD      R9,#0CCH             ; Load data "CCH" to write
    LD      R10,#01H             ; Load Flash memory upper address into upper register of pair working
                                 ; register
    LD      R11,#40H             ; Load Flash memory lower address into lower register of pair working
                                 ; register
    CALL    WR_BYTE


    LD      R0,#40H
WR_INSECTOR5:
    LD      FMSECH,#02H          ; Set the base address of sector located in target address to write data
LD       FMSECL,#80H          ; The sector 5's base address is 0280H
    LD      R9,# 55H             ; Load data "55H" to write
    LD      R10,#02H             ; Load   Flash memory upper address into upper register of pair working
                                 ; register
    LD      R11,#90H             ; Load Flash memory lower address into lower register of pair working
                                 ; register
    CALL    WR_BYTE


WR_INSECTOR12:
    LD      FMSECH,#06H          ; Set the base address of sector located in target address to write data
    LD      FMSECL,#00H          ; The sector 12's base address is 0600H
    LD      R9,#0A3H             ; Load data "A3H" to write
    LD      R10,#06H             ; Load   Flash memory upper address into upper register of pair working
                                 ; register
    LD      R11,#40H             ; Load Flash memory lower address into lower register of pair working
                                 ; register
WR_BYTE1:
    LDC @RR10,R9                 ; Write data 'A3H' at Flash memory location
    INC     R11
    DEC     R1
JP       NZ, WR_BYTE1
    LD      FMUSR,#00H           ; User Program mode disable
•
•
WR_BYTE:
    LDC     @RR10,R9             ; Write data written by R9   at Flash   memory location
    INC     R11
    DEC     R0
```

```
JP        NZ, WR_BYTE
    RET
```

13.5   Reading

The read operation starts by 'LDC' instruction.

13.5.1 Program Procedure In User Program Mode

1. Load a Flash memory upper address into upper register of pair working register.

2. Load a Flash memory lower address into lower register of pair working register.

3. Load receive data from Flash memory location area on 'LDC' instruction by indirectly addressing mode

☞   **PROGRAMMING TIP — Reading**

```
•
•
LD          R2,#03H        ; Load Flash memory's upper address
                           ; to upper register of pair working register
LD          R3,#00H        ; Load Flash memory's lower address
                           ; to lower register of pair working register

LOOP:
LDC         R0,@RR2        ; Read data from Flash memory location
                           ; (Between 300H and 3FFH)
INC         R3
CP          R3,#0FFH
JP          NZ,LOOP
•
•
•
•
```

13.6   Hard Lock Protection

User can set Hard Lock Protection by writing '0110B' in FMCON7-4. This function prevents the changes of data in a Flash memory area. If this function is enabled, the user cannot write or erase the data in a Flash memory area. This protection can be released by the chip erase execution in the tool program mode. In terms of user program mode, the procedure of setting Hard Lock Protection is following that. In Tool Mode, the manufacturer of serial tool writer could support Hardware Protection. Please refer to the manual of serial program writer tool provided by the manufacturer.

13.6.1 Program Procedure In User Program Mode

1. Set Flash Memory User Programming Enable Register (FMUSR) to "10100101B".

2. Set Flash Memory Control Register (FMCON) to "01100001B".

3. Set Flash Memory User Programming Enable Register (FMUSR) to "00000000B".

☞   **PROGRAMMING TIP — Hard Lock Protection**

-
-

```
LD        FMUSR,#0A5H              ; User program mode enable
LD        FMCON,#01100001B         ; Select Hard Lock Mode and Start protection
LD        FMUSR,#00H               ; User program mode disable
```

14 Electrical Data

14.1   Overview

In this chapter, the following S3F94C8/S3F94C4 electrical characteristics are presented in tables and graphs:

Absolute maximum ratings

D.C. electrical characteristics

A.C. electrical characteristics

Input timing measurement points

Oscillator characteristics

Oscillation stabilization time

Operating voltage range

Schmitt trigger input characteristics

Data retention supply voltage in Stop Mode

Stop Mode Release timing when initiated by a RESET

A/D converter electrical characteristics

LVR circuit characteristics

LVR reset timing

Full-Flash memory characteristics

ESD Characteristics

Table 14-1. Absolute Maximum Ratings

(T_A = 25 °C)

Parameter	Symbol	Conditions	Rating	Unit
Supply voltage	V_{DD}	–	– 0.3 to + 6.5	V
Input voltage	V_I	All ports	– 0.3 to V_{DD} + 0.3	V
Output voltage	V_O	All output ports	– 0.3 to V_{DD} + 0.3	V
Output current high	I_{OH}	One I/O pin active	– 25	mA
		All I/O pins active	– 80	
Output current low	I_{OL}	One I/O pin active	+ 30	mA
		All I/O pins active	+ 100	
Operating temperature	T_A	–	– 40 to + 85	°C
Storage temperature	T_{STG}	–	– 65 to + 150	°C

Table 14-2. DC Electrical Characteristics

$(T_A = -40\ °C$ to $+85\ °C$, $V_{DD} = 1.8\ V$ to $5.5\ V)$

Parameter	Symbol	Conditions		Min	Typ	Max	Unit
Operating Voltage	V_{DD}	f_{main}=0.4 – 4 MHz		1.8	–	5.5	V
		f_{main}=0.4 – 10 MHz		2.7	–	5.5	
Main crystal or ceramic frequency	f_{main}	V_{DD} = 2.7 V to 5.5V		0.4	–	10	MHz
		V_{DD} = 1.8 V to 2.7V		0.4	–	4	
Input high voltage	V_{IH1}	Ports 0,1, 2 and RESET	V_{DD} = 1.8 to 5.5 V	0.8 V_{DD}	–	V_{DD}	V
	V_{IH2}	X_{IN} and X_{OUT}		V_{DD} - 0.1			
Input low voltage	V_{IL1}	Ports 0, 1, 2 and RESET	V_{DD} = 1.8 to 5.5 V	–	–	0.2 V_{DD}	V
	V_{IL2}	X_{IN} and X_{OUT}				0.1	
Output high voltage	V_{OH}	I_{OH} = – 10 mA Ports 0,2,P1.0-P1.1	V_{DD} = 4.5 to 5.5 V	V_{DD} -1.5	V_{DD} - 0.4	–	V
Output low voltage	V_{OL}	I_{OL} = 25 mA Ports 0,2,P1.0-P1.1	V_{DD} = 4.5 to 5.5 V	–	0.4	2.0	V
Input high leakage current	I_{LIH1}	All input except $I_{LIH2,P1.2}[2]$	V_{IN} = V_{DD}	–	–	1	uA
	I_{LIH2}	X_{IN}	V_{IN} = V_{DD}			20	
Input low leakage current	I_{LIL1}	All input except I_{LIL2}	V_{IN} = 0 V	–	–	–1	uA
	I_{LIL2}	X_{IN}	V_{IN} = 0 V			–20	
Output high leakage current	I_{LOH}	All output pins	V_{OUT} = V_{DD}	–	–	2	uA
Output low leakage current	I_{LOL}	All output pins	V_{OUT} = 0 V	–	–	–2	uA
Pull-up resistors	R_{P1}	V_{IN} = 0 V, Ports 0, 2, P1.0-P1.1	V_{DD} = 5 V T_A=25°C	25	50	100	kΩ
Pull-down resistors	R_{P2}	V_{IN} = 0 V, P1.0-P1.1	V_{DD} = 5 V T_A=25°C	25	50	100	

Supply current[1]	I_{DD1}	Run mode 10 MHz CPU clock	V_{DD} = 4.5 to 5.5 V	–	2	5	mA
		3MHz CPU clock	V_{DD} = 2.0V		1	2	
	I_{DD2}	Idle Mode 10 MHz CPU clock	V_{DD} = 4.5 to 5.5 V	–	1.5	3.0	
		3MHz CPU clock	V_{DD} = 2.0V		0.5	1.5	
	I_{DD3}	Stop Mode	V_{DD} = 4.5 to 5.5 V (LVR disable) $T_A = 25\ °C$	–	0.3	2.0	uA
			V_{DD} = 4.5 to 5.5 V (LVR disable) T_A= – 40°C to +85°C	–	1	4.0	
			V_{DD} = 4.5 to 5.5 V (LVR enable) T_A = – 40°C to +85°C	–	40	80	
			V_{DD} = 2.6 V (LVR enable) T_A = – 40°C to +85°C	–	30	60	

NOTE: 1. Supply current does not include current drawn through internal pull-up resistors or external output
current loads and ADC module.

Table 14-3. AC Electrical Characteristics

($T_A = -40\ °C$ to $+85\ °C$, $V_{DD} = 1.8\ V$ to $5.5\ V$)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Interrupt input high, low width	t_{INTH} t_{INTL}	INT0, INT1 $V_{DD} = 5\ V \pm 10\ \%$	–	200	–	ns
RESET input low width	t_{RSL}	Input $V_{DD} = 5\ V \pm 10\ \%$	1	–	–	us
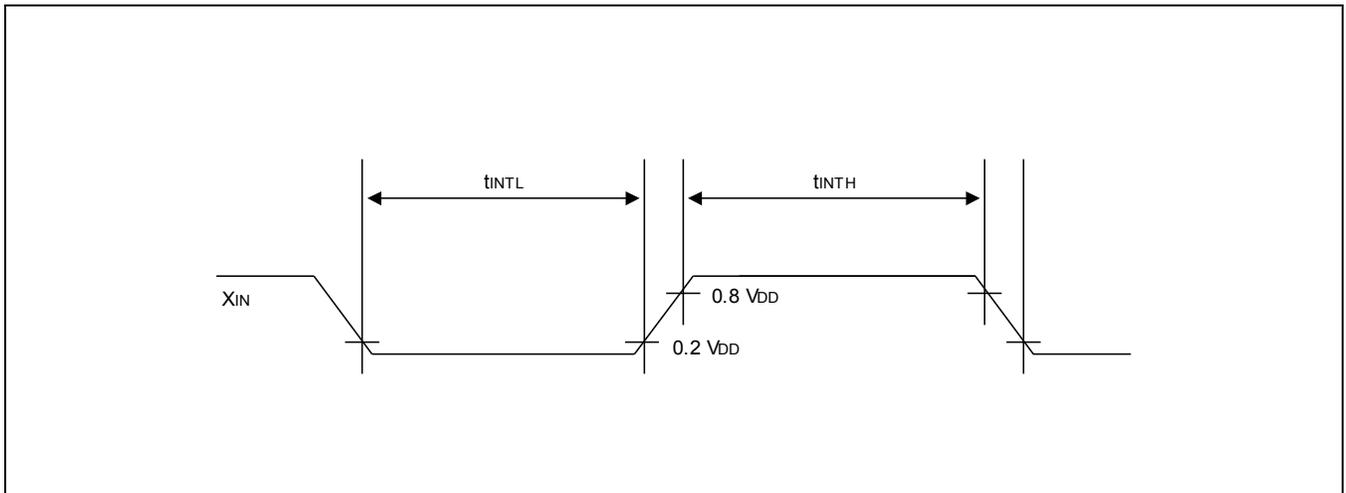
Figure 14-1. Input Timing Measurement Points

Table 14-4. Crystal or Ceramic Oscillator Characteristics

$(T_A = -40°C$ to $+85°C)$

Oscillator	Clock Circuit	Test Condition	Min	Typ	Max	Unit
Main crystal or ceramic	C1 X_{IN} C2 X_{OUT}	$V_{DD} = 2.7$ to 5.5 V	0.4	–	10	MHz
		$V_{DD}{}^1 = 1.8$ to 2.7 V	0.4	–	4	MHz
External clock (Main System)	X_{IN} X_{OUT}	$V_{DD} = 2.7$ to 5.5 V	0.4	–	10	MHz
		$V_{DD} = 1.8$ to 2.7 V	0.4	–	4	MHz
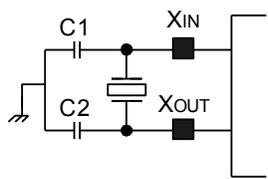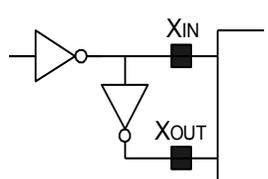
NOTE: 1. Please refer to the figure of Operating Voltage Range.

Table 14-5. Oscillation Stabilization Time

$(T_A = -40°C$ to $+85°C$, $V_{DD} = 1.8$ V to 5.5 V$)$

Oscillator	Test Condition	Min	Typ	Max	Unit
Main crystal	$f_{OSC} > 1.0$ MHz	–	–	20	ms
Main ceramic	Oscillation stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	10	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]	–	$2^{19}/f_{OSC}$	–	ms
	t_{WAIT} when released by an interrupt [2]	–	–	–	ms

NOTES:
1. f_{OSC} is the oscillator frequency.
2. The duration of the oscillator stabilization wait time, t_{WAIT}, when it is released by an interrupt is determined by the settings in the Basic Timer Control Register, BTCON.

Table 14-6. RC Oscillator Characteristics (S3F94C8EZZ/S3F94C4EZZ)

$(T_A = -25°C$ to $+85°C$, $V_{DD} = 1.8$ V to 5.5 V$)$

Oscillator	Clock Circuit	Test Condition	Min	Typ	Max	Unit

External RC oscillator	–	V_{DD} = 5 V	–	4	–	MHz
Internal RC oscillator	–	–	–	3.2	–	MHz
			–	500	–	KHz
Tolerance of Internal RC	–	V_{DD} = 5.0 V T_A = 25 °C	–	–	± 3	%
		V_{DD} = 5.0 V T_A = – 25°C to + 85 °C	–	–	± 5	%
		V_{DD} = 2.0 to 5.5 V T_A = – 25°C to + 85 °C	–	–	± 8	%

Table 14-7 RC Oscillator Characteristics (S3F94C8XZZ/S3F94C4XZZ)

(T_A = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Oscillator	Clock Circuit	Test Condition	Min	Typ	Max	Unit
External RC oscillator	–	V_{DD} = 5 V	–	4	–	MHz
Internal RC oscillator	–	–	–	3.2	–	MHz
			–	500	–	KHz
Tolerance of Internal RC	–	V_{DD} = 1.8 to 5.0 V TA = 25 °C	–	± 0.5	± 1	%
		V_{DD} = 1.8 to 5.5 V T_A = – 40°C to + 85 °C	–	–	± 3.5	%
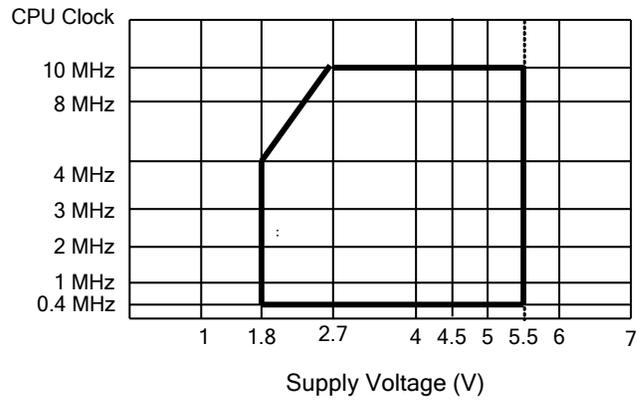
Figure 14-2. Operating Voltage Range
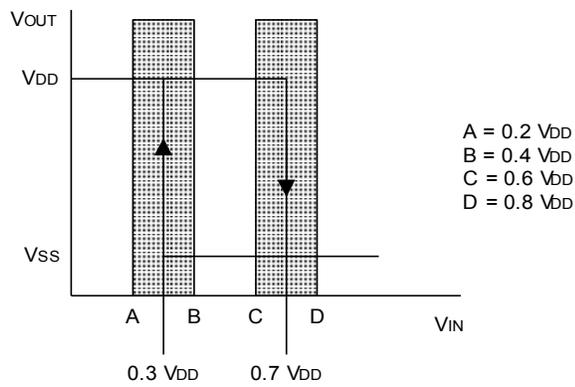
Figure 14-3. Schmitt Trigger Input Characteristics Diagram

Table 14-8. Data Retention Supply Voltage in Stop Mode

(T_A = – 40 °C   to   + 85 °C, V_{DD} = 1.8 V   to   5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data retention supply voltage	V_{DDDR}	Stop Mode	1.0	–	5.5	V
Data retention supply current	I_{DDDR}	Stop Mode; $V_{DDDR =}$ 2.0 V	–	–	1	uA

NOTE: Supply current does not include current drawn through internal pull-up resistors or external output current loads.
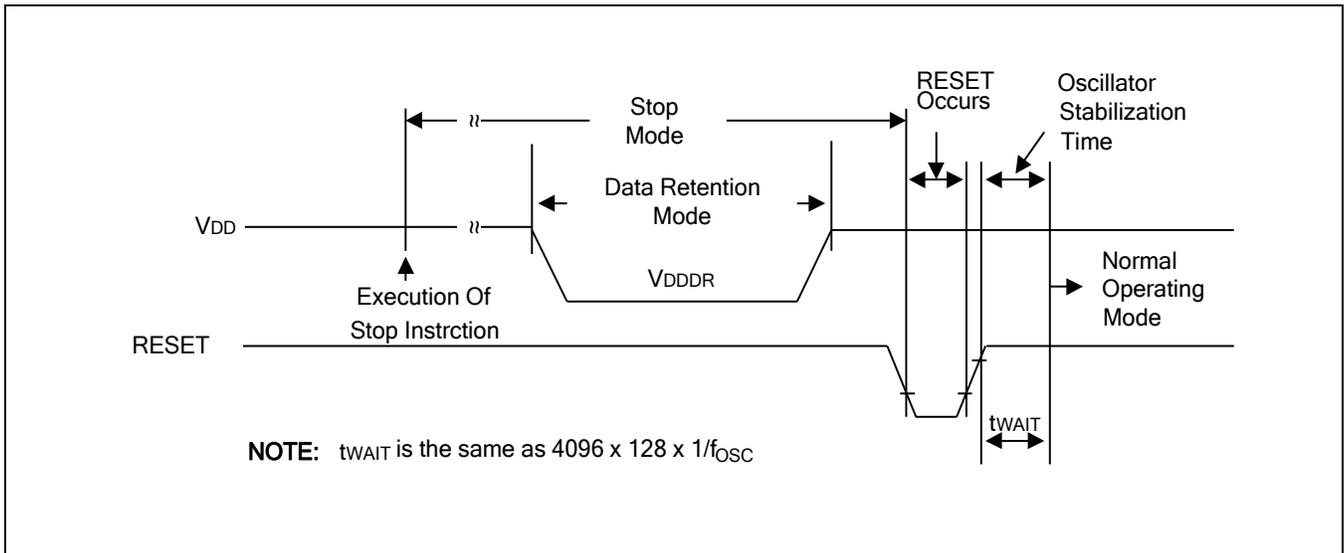
Figure 14-4. Stop Mode Release Timing When Initiated by a RESET

Table 14-9. A/D Converter Electrical Characteristics

$(T_A = -40\,°C$ to $+85\,°C$, $V_{DD} = 1.8$ V to 5.5 V, $V_{SS} = 0$ V)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Unit
Resolution			–	10	–	bit
Total accuracy		$V_{DD} = 5.12$ V CPU clock = 10 MHz $V_{SS} = 0$ V	–	–	± 3[1]	LSB
Integral linearity error	ILE	"	–	–	± 2	LSB
Differential linearity error	DLE	"	–	–	± 1	LSB
Offset error of top	EOT	"	–	± 1	± 3	LSB
Offset error of bottom	EOB	"	–	± 1	± 3	LSB
Conversion time [2]	t_{CON}	"	–	20	–	μs
Analog input voltage	V_{IAN}	–	V_{SS}	–	V_{DD}	V
Analog input impedance	R_{AN}	–	2	1000	–	$M\Omega$
Analog input current	I_{ADIN}	$V_{DD} = 5$ V	–	–	10	μA
Analog block current [3]	I_{ADC}	$V_{DD} = 5$ V	–	0.5	1.5	mA
		$V_{DD} = 3$ V		0.15	0.45	mA
		$V_{DD} = 5$ V power down mode		100	500	nA

NOTES:

1. The total accuracy is 3LSB(max.) at $V_{DD} = 2.7V – 5.5V$, It's for design guidance only and are not tested in production.

2. *Conversion time* is the time required from the moment a conversion operation starts until it ends.

3. I_{ADC} is operating current during A/D conversion.

Table 14-10. LVR Circuit Characteristics

(T_A = − 40 °C   to   + 85 °C, V_{DD} = 1.8 V   to   5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Low voltage reset	V_{LVR}	–	1.8 2.1 2.8 3.4 3.7	1.9 2.3 3.0 3.6 3.9	2.0 2.5 3.2 3.8 4.1	V
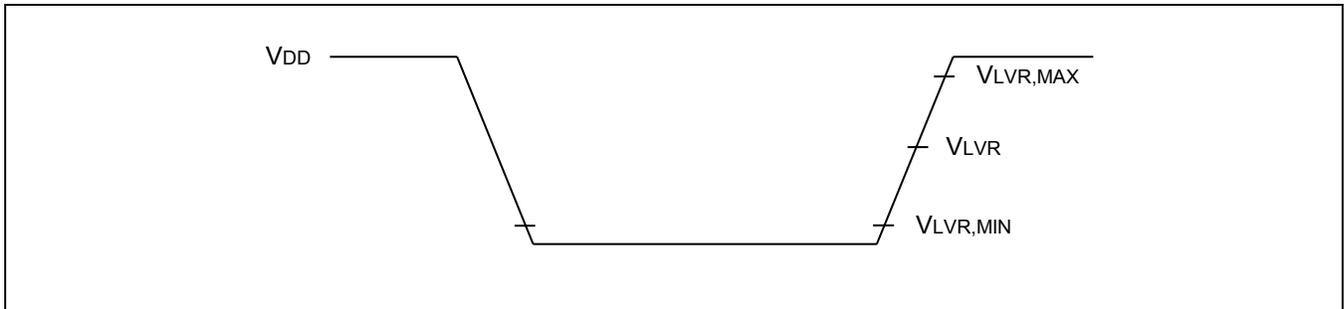
Figure 14-5. LVR Reset Timing

14.2   Table 14-11.   Flash Memory AC Electrical Characteristics

(T_A = − 40 °C to + 85 °C at V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Flash Erase/Write/Read Voltage	Fewrv	V_{DD}	1.8	5.0	5.5	V
Programming time(1)	Ftp		20	–	30	uS
Chip Erasing time (2)	Ftp1		32	–	70	mS
Sector Erasing time (3)	Ftp2		4	–	12	mS
Data Access Time	Ft_{RS}	V_{DD} = 2.0V	–	250	–	nS
Number of writing/erasing	FNwe	–	10,000	–	–	Times
Data Retention	Ftdr	–	10	–	–	Years

NOTES:

1. The programming time is the time during which one byte (8-bit) is programmed.
2. The Chip erasing time is the time during which entire program memory is erased.
3. The Sector erasing time is the time during which all 128byte block is erased.
4. The chip erasing is available in Tool Program Mode only.
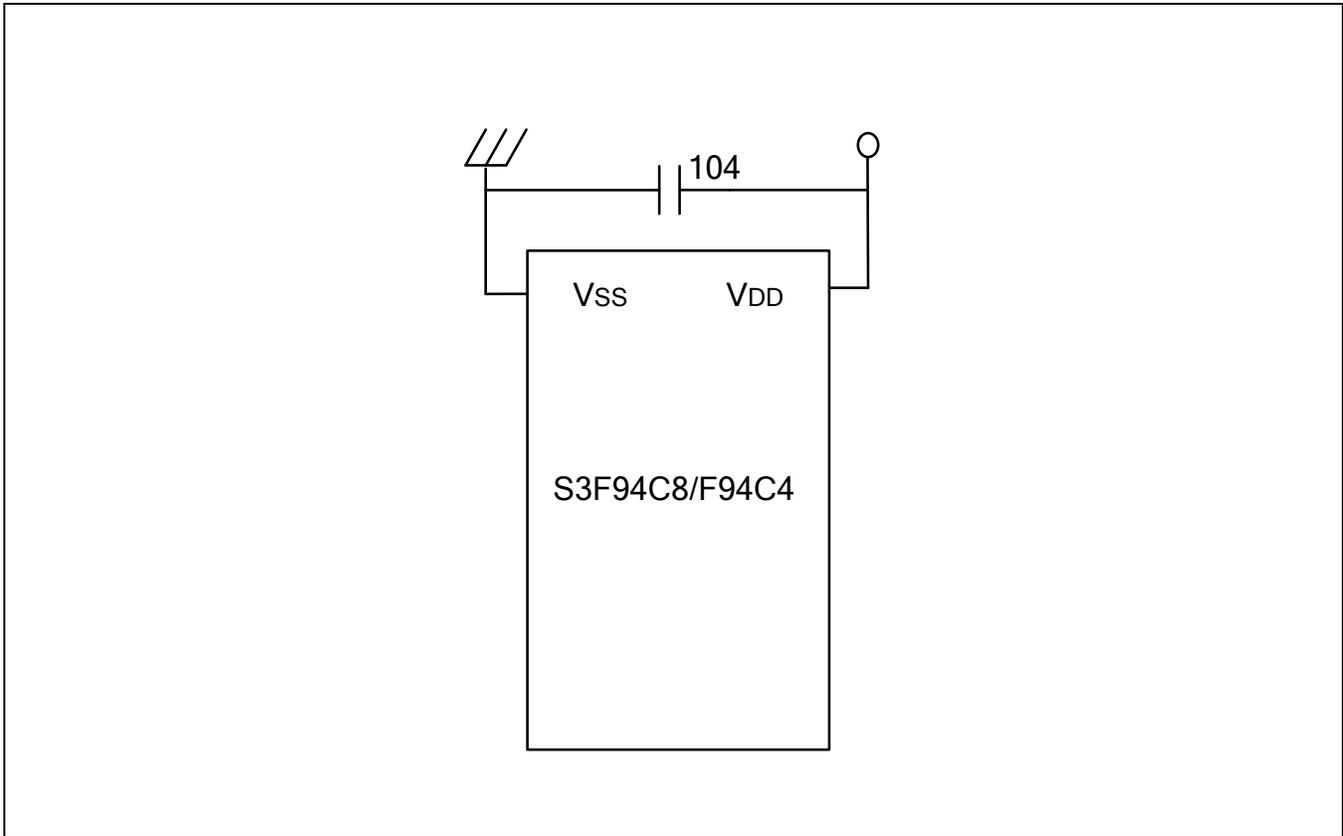
Figure 14-6. The Circuit Diagram to Improve EFT Characteristics

NOTE

To improve EFT characteristics, we recommend using power capacitor near S3F94C8/S3F94C4 like Figure 14-6.

Table 14-12. ESD Characteristics

Parameter	Symbol	Conditions	Min	Typ.	Max	Unit
Electrostatic discharge	V_{ESD}	HBM	2000	☐	☐	V
		MM	200	☐	☐	V
		CDM	500	☐	☐	V

15 Mechanical Data

The S3F94C8 and S3F94C4 MCUs are available in 20-pin DIP, 20-pin SOP, 20-pin SSOP, 16-pin SOP, and 16-pin TSSOP packages. Package dimensions are shown in Figures 15-1 through 15-6.

NOTE: Dimensions are in millimeters.
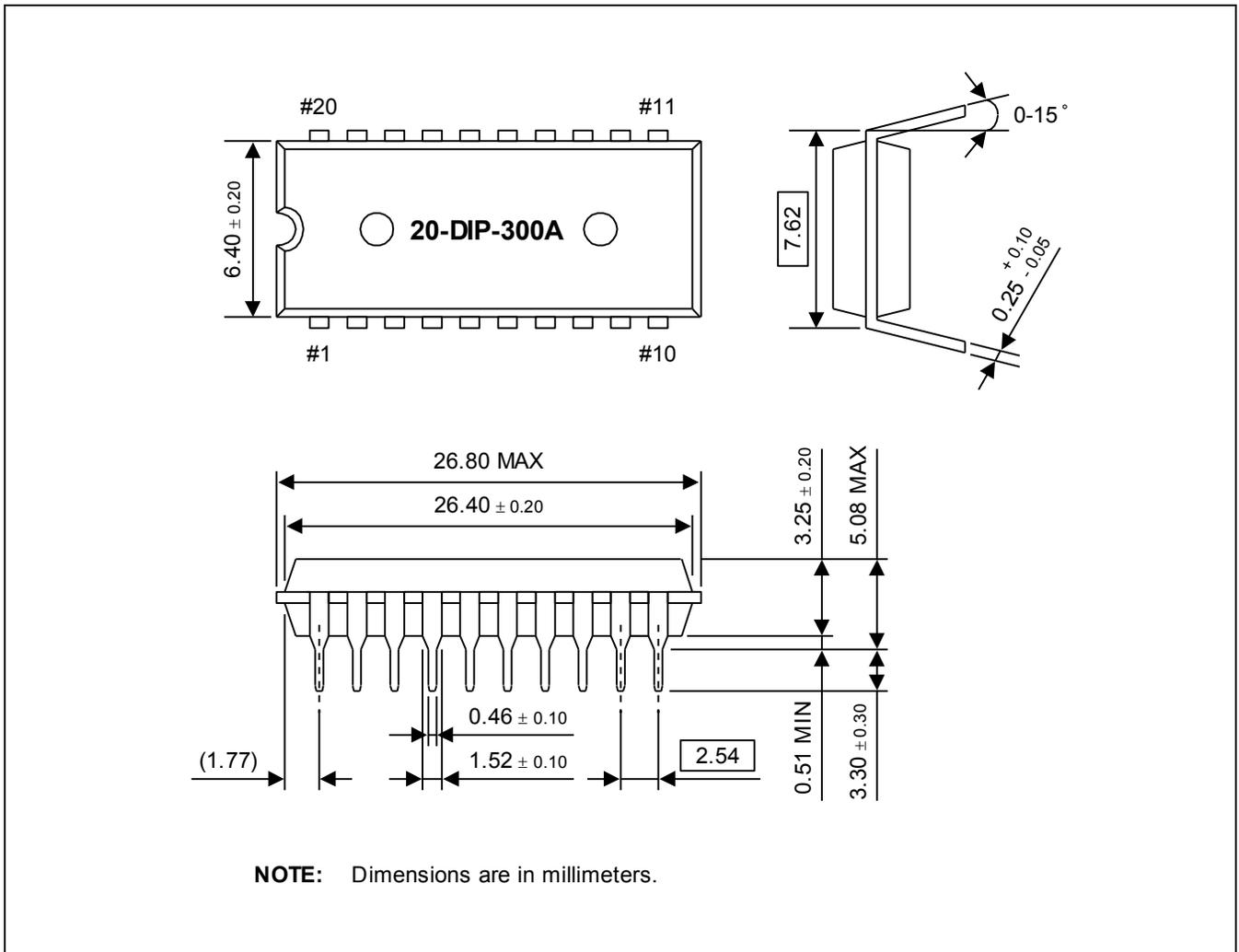
Figure 15-1. 20-DIP-300A Package Dimensions
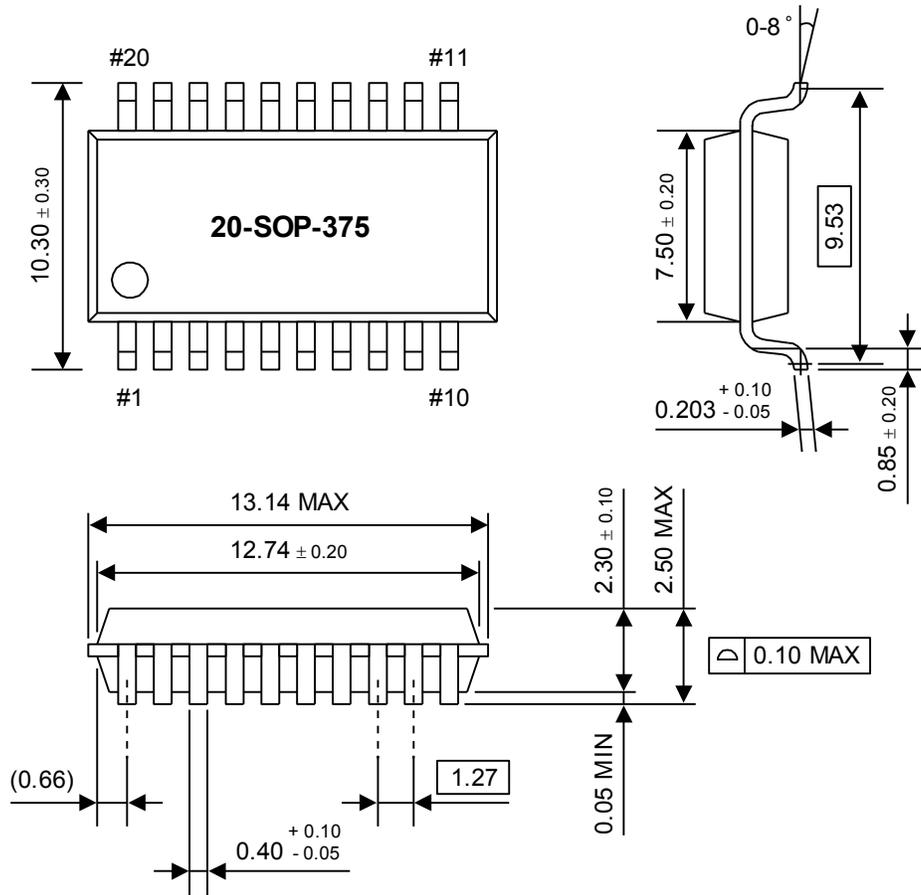
Figure 15-2. 20-SOP-375 Package Dimensions
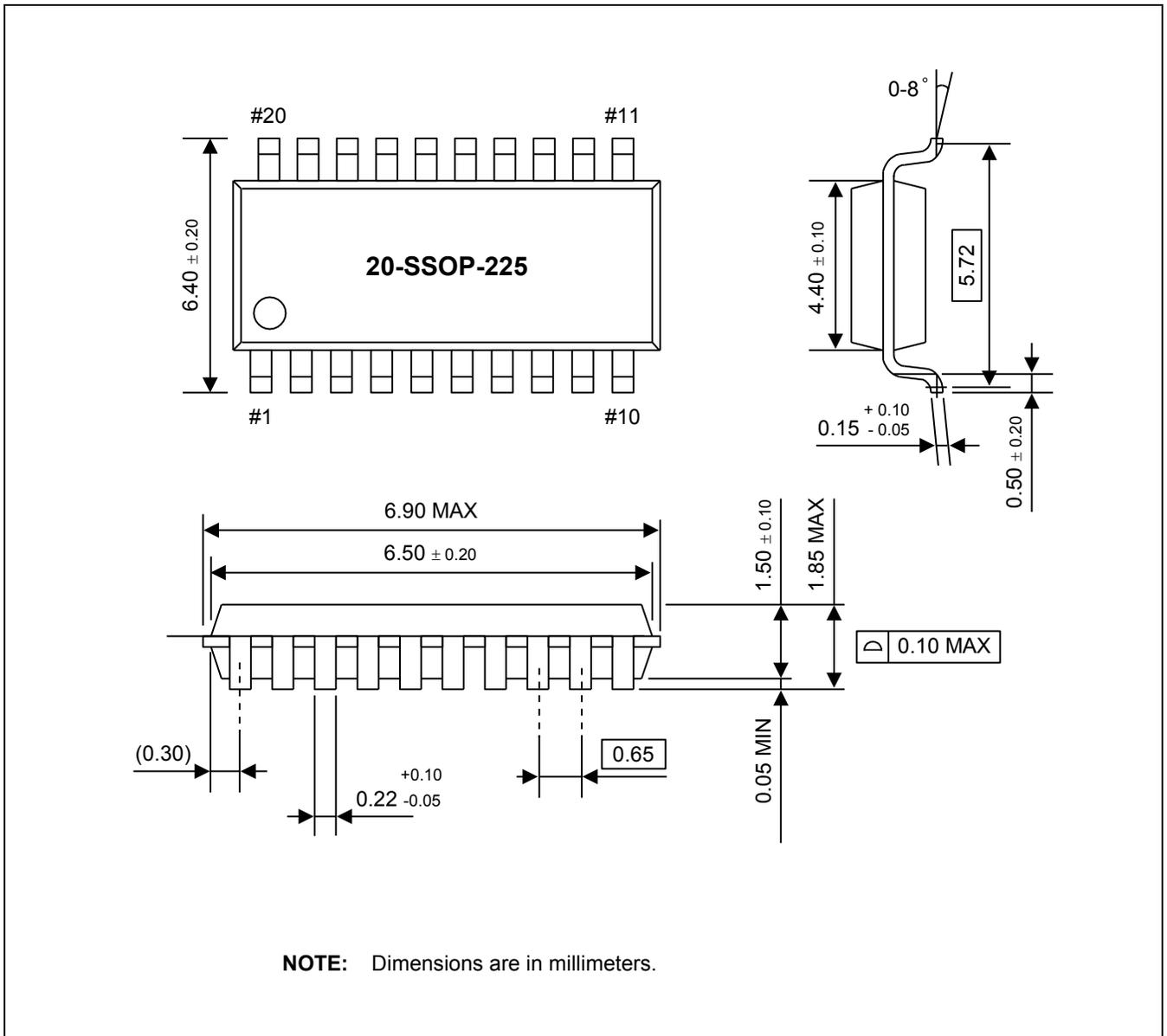
Figure 15-3. 20-SSOP-225 Package Dimensions

16-SOP-225

NOTE: Dimensions are in millimeters.
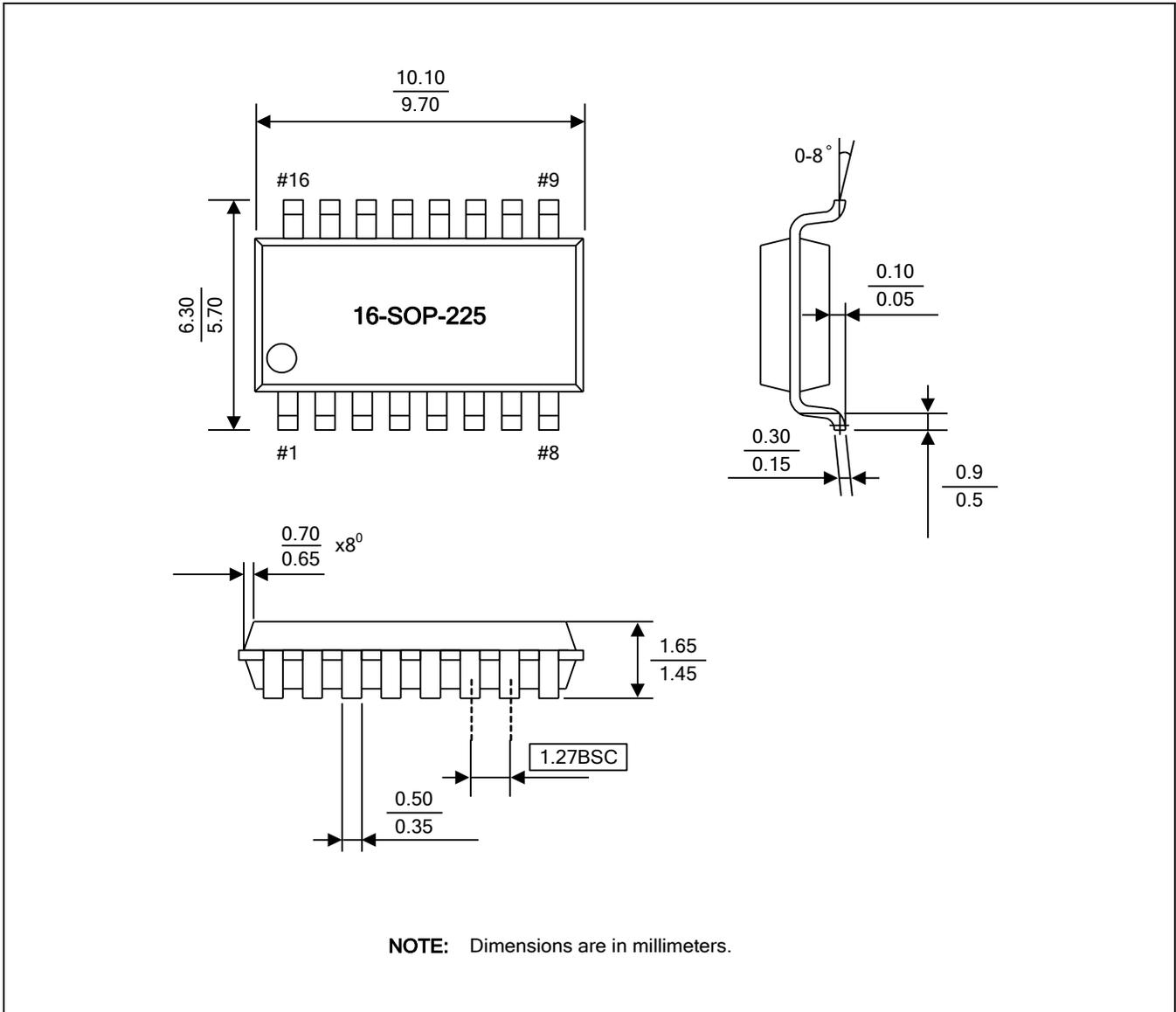
Figure 15-4. 16-SOP-225 Package Dimensions
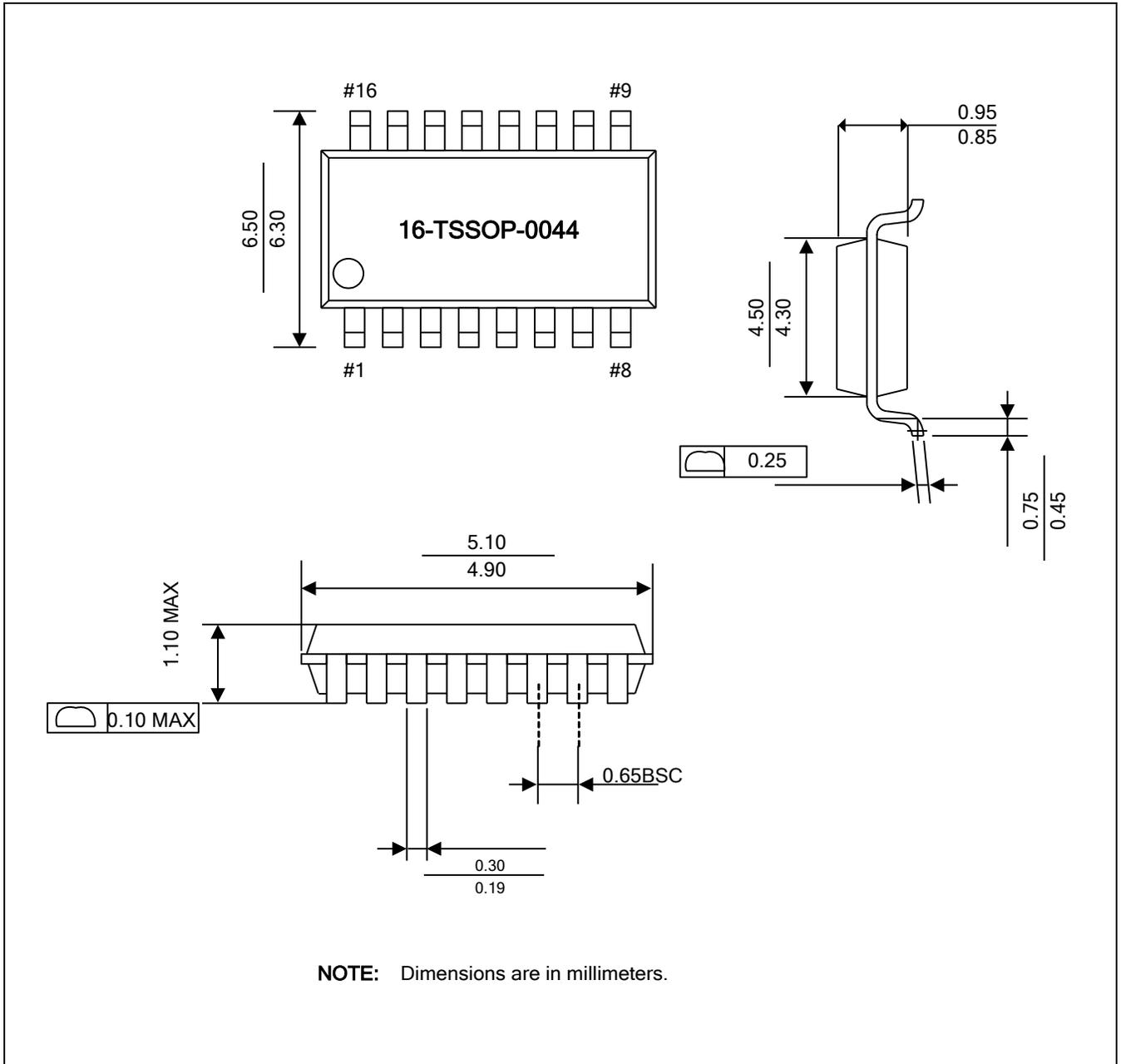
Figure 15-5. 16-TSSOP-0044 Package Dimensions

16 S3F94C8/S3F94C4 Flash MCUs

16.1   Overview

The S3F94C8/S3F94C4 single-chip CMOS microcontroller is the Flash MCU. It has an on-chip Flash MCU ROM of 8K/4K bytes. The Flash ROM is accessed by serial data format.

The serial data is transformed by two pins of the chip: SCLK and SDAT, SCLK is the synchronize signal, and the Flash Programmer Tool send data from the SDAT pin. The corresponding ports of SCLK and SDAT in S3F94C8/S3F94C4 are P0.0 and P1.1. And there also need power supply for chip to work and higher power for entering Flash Tool Mode. So the VDD, VSS of chip must be connected to power and ground. The higher power supply for the Flash operation is named as VPP port, the corresponding pin in S3F94C8/S3F94C4 is nRESET (P1.2) pin. The detail description of the pin functions are listed in the table 16-1.The pin assignments of the S3F94C8/S3F94C4 package types are shown in below figures.

NOTES

1. This chapter is about the Tool Program Mode of Flash MCU. If you want to know the User Program Mode, refer to the chapter 13. Embedded Flash Memory Interface.

2. In S3F94C8/S3F94C4, there only 5 pins are used as Flash operation pins, the nRESET pin is used as VPP input and without TEST pin that different with other Zilog MCU products.

NOTE: The bolds indicate MTP pin name.
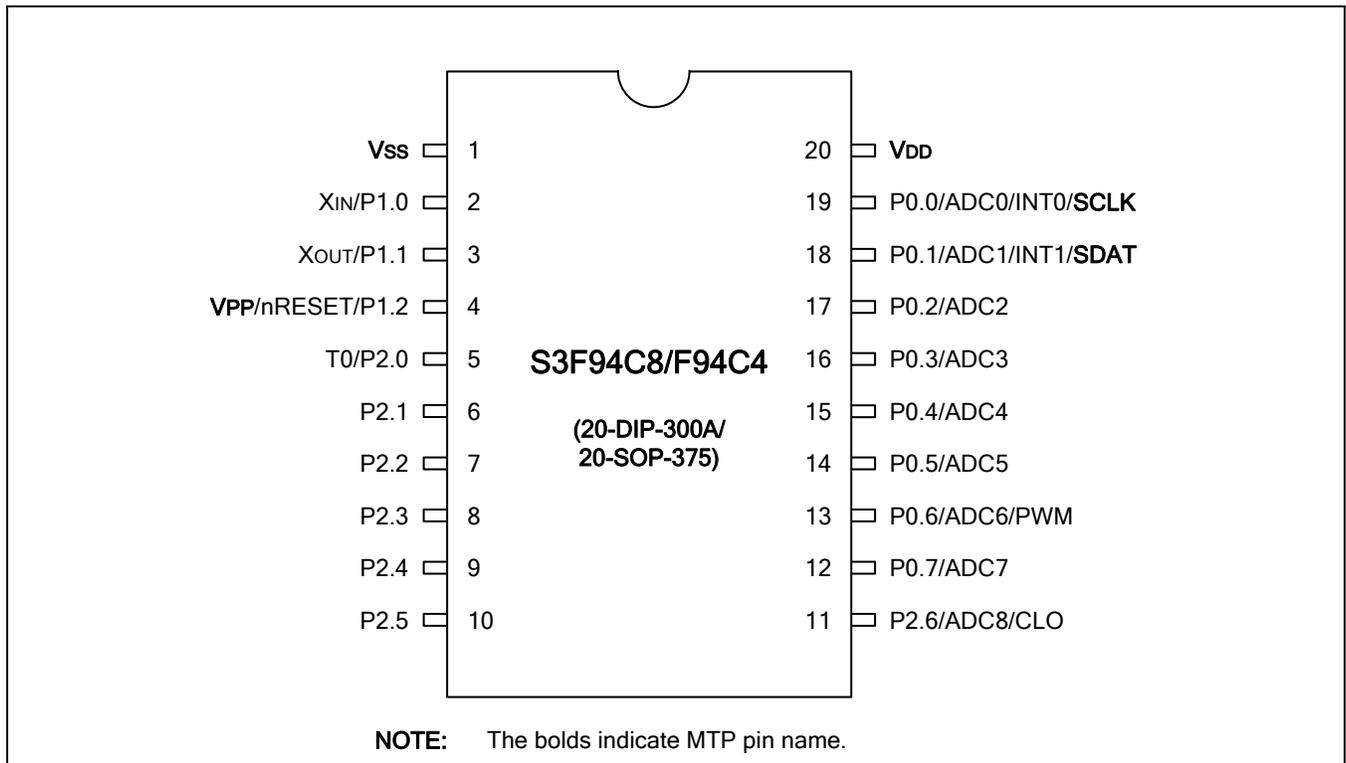
Figure 16-1. S3F94C8/S3F94C4 Pin Assignments (20-DIP/20SOP)
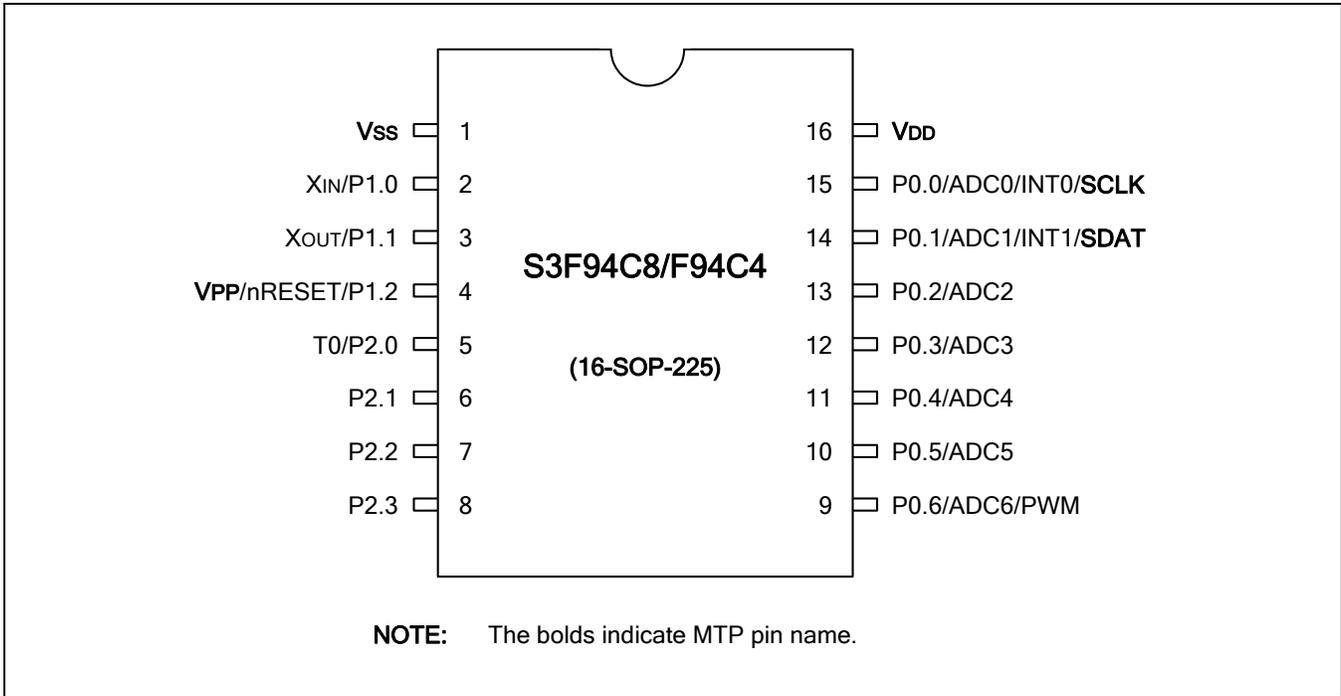
Figure 16-2. S3F94C8/S3F94C4 Pin Assignments (16SOP)

Table 16-1. Descriptions of Pins Used to Read/Write the EPROM

Main Chip Pin Name	During Programming			
	Pin Name	Pin No.	I/O	Function
P0.1	SDAT	18 (20-pin) 14 (16-pin)	I/O	Serial data pin (output when reading, Input when writing) Input and push-pull output port can be assigned
P0.0	SCLK	19 (20-pin) 15 (16-pin)	I	Serial clock pin (input only pin)
RESET/P1.2	V_{PP}	4	I	Power supply pin for Tool mode entering (indicates that MTP enters into the Tool mode). When 11 V is applied, MTP is in Tool mode.
V_{DD}/V_{SS}	V_{DD}/V_{SS}	20 (20-pin), 16 (16-pin) 1 (20-pin), 1 (16-pin)	I	Logic power supply pin.

NOTES: Parentheses indicate pin number for 20-DIP-300A package.

Table 16-2. Comparison of S3F94C8/S3F94C4 Features

Characteristic	S3F94C8/S3F94C4
Program memory	8K/4K Flash ROM
Operating voltage (V_{DD})	2.0 V to 5.5 V
Flash MCU programming mode	V_{DD} = 5.0 V, V_{PP} (nRESET) = 11 V
Pin configuration	20 DIP/20 SOP/20 SSOP/16SOP/16TSSOP
Programmability	User program multi time

16.2 On-Board Writing

The S3F94C8/S3F94C4 MCUs require only 5 signal lines including VDD and GND pins for writing internal Flash memory with serial protocol. Therefore the on-board writing is possible if the writing signal lines are considered when the PCB of application board is designed.

16.2.1 Circuit Design Guide

At the Flash writing, the writing tool needs 5 signal lines that are GND, VDD, VPP, SDAT and SCLK. When you design the PCB circuits, you should consider the usage of these signal lines for the on-board writing.

In case of VPP (nRESET) pin, for the purpose of increase the noise effect, a capacitor should be inserted between the VPP pin and GND.

Please be careful to design the related circuit of these signal pins because rising/falling timing of VPP, SCLK and SDAT is very important for proper programming.
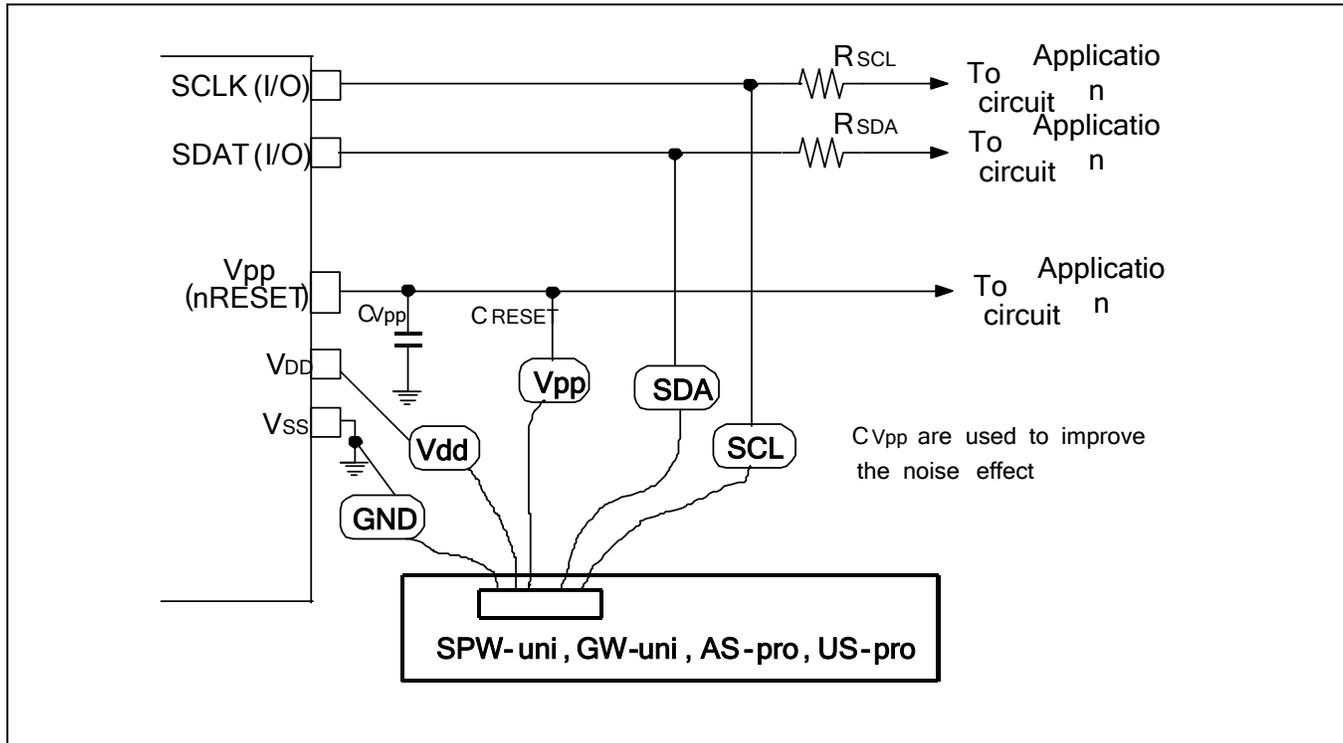
Figure 16-3. PCB Design Guide for Onboard Programming

Table 16-3. **Reference Table for Connection**

Pin Name	I/O mode in Applications	Resistor (need)	Required value
Vpp(nRESET)	Input	Yes	C_{Vpp} is 0.01uF ~ 0.02uF.
SDAT(I/O)	Input	Yes	R_{SDAT} is 2 Kohm ~ 5 Kohm.
	Output	No(Note)	-
SCLK(I/O)	Input	Yes	R_{SCLK} is 2 Kohm ~ 5 Kohm.
	Output	No(Note)	-

NOTES

1. In on-board writing mode, very high-speed signal will be provided to pin SCLK and SDAT. And it will cause some damages to the application circuits connected to SCLK or SDAT port if the application circuit is designed as high speed response such as relay control circuit. If possible, the I/O configuration of SDAT, SCLK pins had better be set to input mode.

2. The value of R, C in this table is recommended value. It varies with circuit of system.

16.3   Information Block

The S3F94C8/94C4 provides a special Flash area for storing chip ID or customer's information into it, called information block. This block is separated from the main Flash ROM, the Flash ROM memory erase/write/read/read protection operation take none affect to this block.   It can be erase/write/read by Flash Programmer Tools individually and is not available in User Mode.

The size of information block is 256Bytes. Since it is separated from Flash ROM, the programming operation (chip erase/write) will not erase/change the data in information block. User can write Chip ID into it, that different for each chip, to distinguish every chip. This is very useful for anti-imitation by storing production related information in this area.
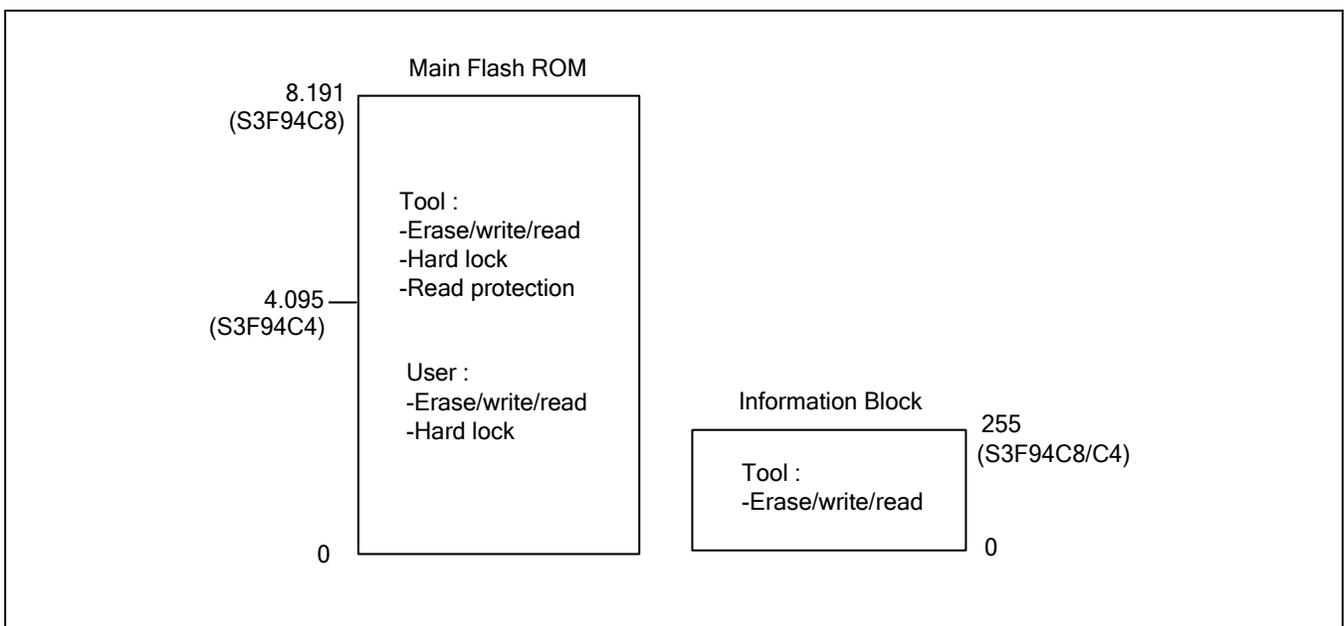

Figure 16-4. S3F94C8/S3F94C4 Flash Architecture.

Table 16-4. Operation Results Comparison of Main ROM and Information Block

Mode	Operation	Main Flash ROM	Information Block
Tool Mode	Erase MTP	Yes	No
	Program ROM/Read ROM	Yes	No
	Hard Lock/Read Protection	Yes	No
	Information Block Erase	No	Yes
	Information Block Write/Read	No	Yes
User Mode	Sector erase	Yes	No
	Write Byte/Read Byte	Yes	No
	Hard Lock	Yes	No

17 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 both in hardware and software: the powerful in-circuit emulator, OPENice-i500 and SK-1200, for the S3C7-, S3C9- and S3C8- microcontroller families. Zilog also offers supporting software that includes, debugger, an assembler, and a program for setting options.

17.1   Target Boards

Target boards are available for all the S3C9/S3F9-series microcontrollers. All the required target system cables and adapters are included on the device-specific target board. TB94C8/94C4 is a specific target board for the development of application systems using S3F94C8/S3F94C4.

17.2   Programming Socket Adapter

When you program S3F94C8/S3F94C4 MCU Flash memory by using an emulator or OTP/MTP writer, you will require a specific programming socket adapter for these two devices.
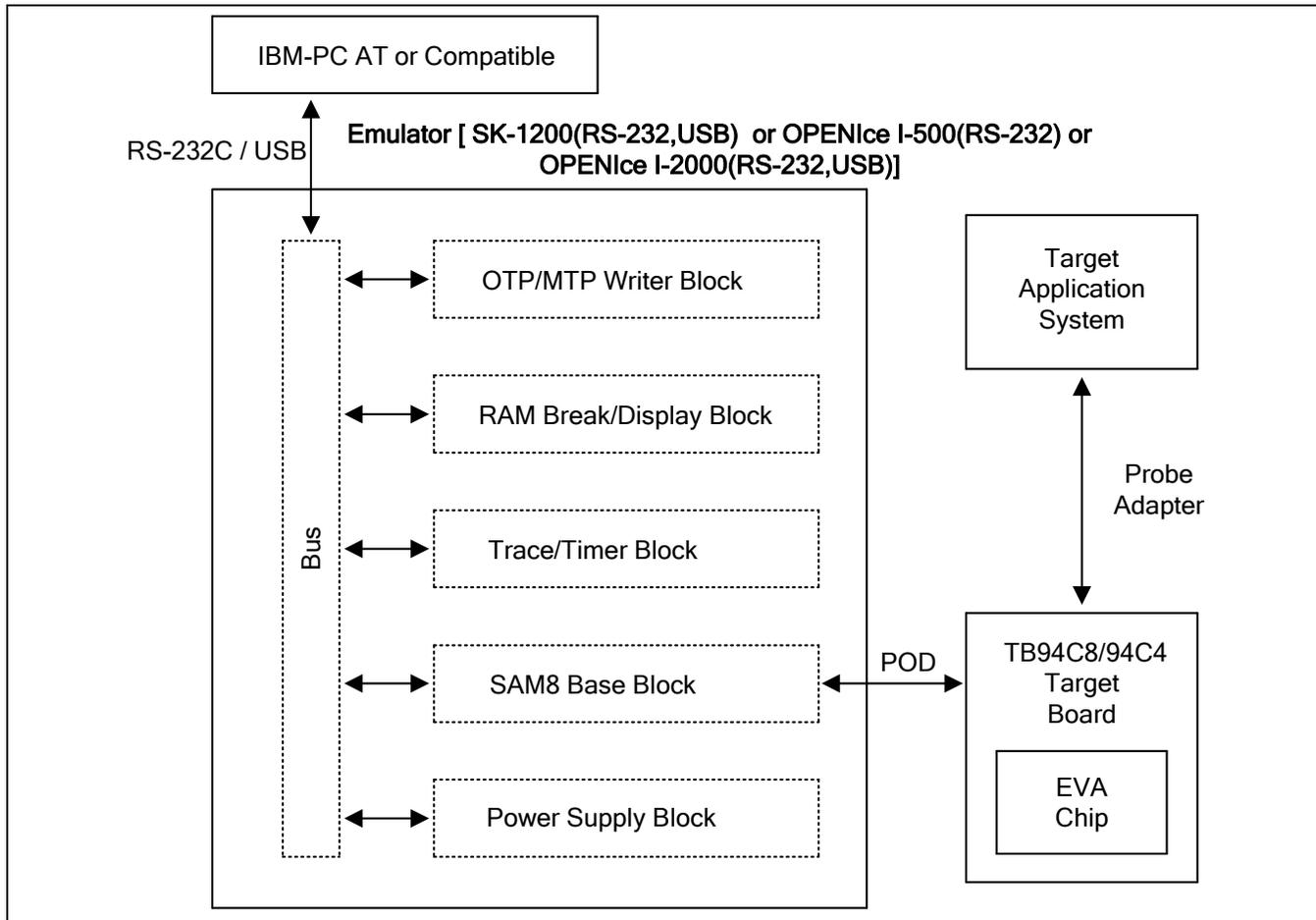
17.2.1 Development System Configuration

Figure 17-1. Development System Configuration

17.2.2 TB94C8/94C4 Target Board

The TB94C8/94C4 target board is used for the S3F94C8/S3F94C4 microcontrollers. The TB94C8/94C4 target board is operated as target CPU with Emulator (OPENIce I-500/2000, SK-1200).
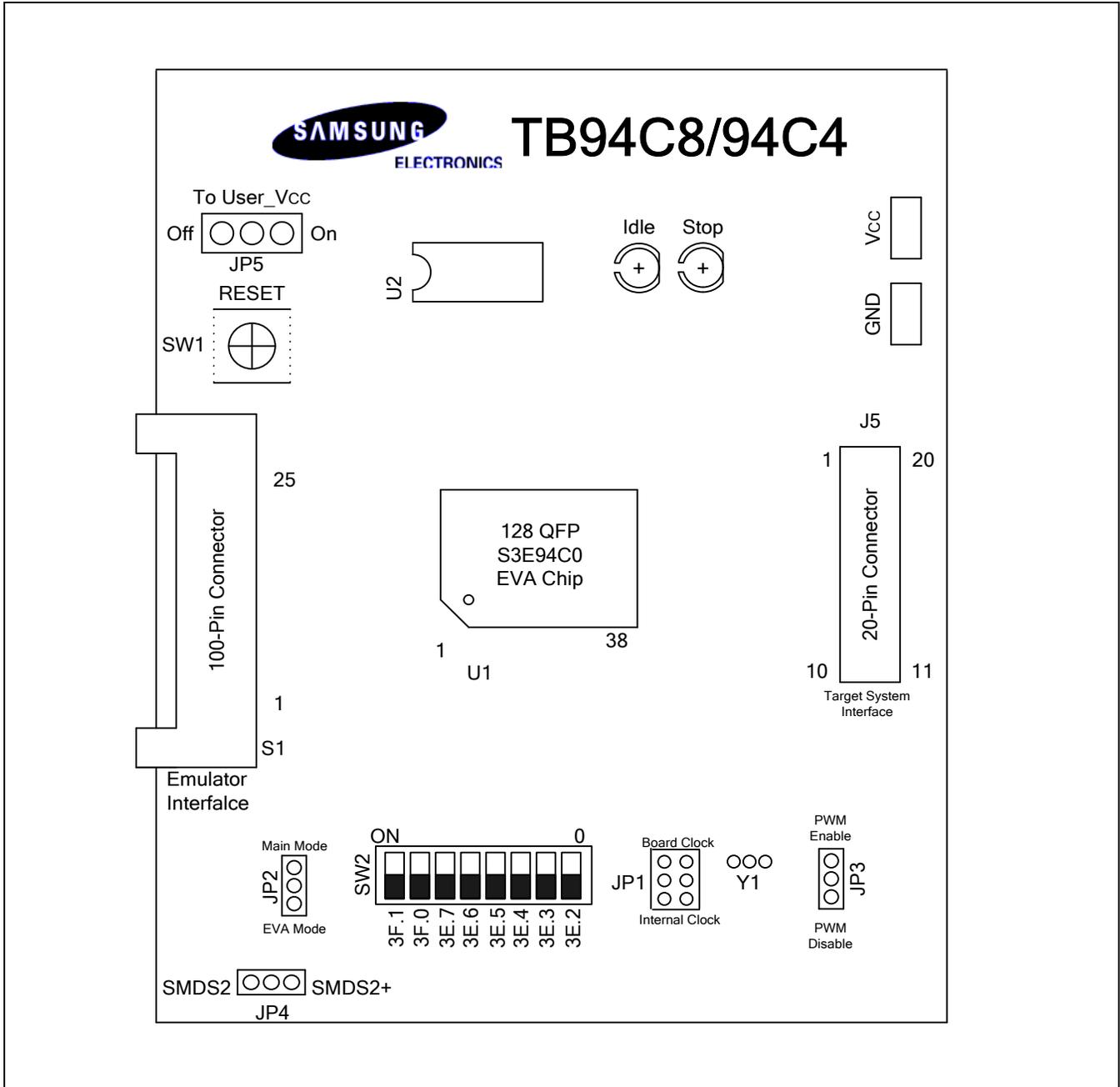
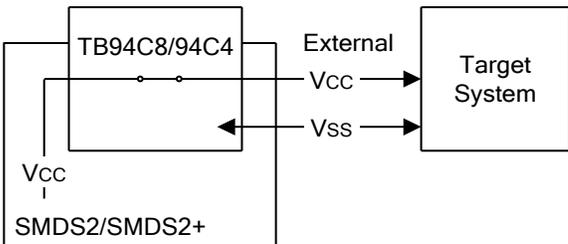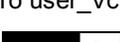

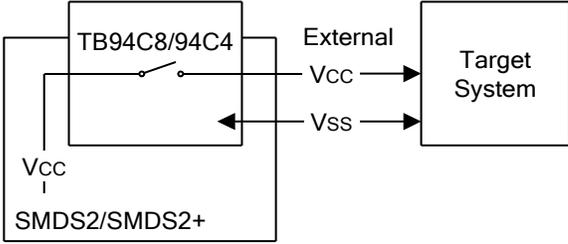**Figure 17-2. TB94C8/94C4 Target Board Configuration**

NOTE

TB94C8/94C4 should be supplied 5V normally. So the power supply from Emulator should be set 5V for the target board operation.

Table 17-1. Components of TB94C8/94C4

Symbols	Usage	Description
S1	100-pin connector	Connection between emulator and TB94C8/94C4 target board.
J5	20-pin connector	Connection between target board and user application system
SW2	8-pin switch	Smart Option setting for S3F94C8/94C4 EVA-chip
RESET	Push button	Generation low active reset signal to S3F94C8/94C4 EVA-chip
VCC, GND	POWER connector	External power connector for TB94C8/94C4
IDLE, STOP LED	STOP/IDLE Display	Indicate the status of STOP or IDLE of S3F94C8/94C4 EVA-chip on TB94C8/94C4 target board
JP1	Clock Source Selection	Selection of SMDS2/SMDS2+ internal/external clock
JP2	MODE Selection	Selection of Eva/Main-chip mode of S3F94C8/94C4 EVA-chip
JP3	PWM selection	Selection of PWM enable/disable
JP4	Emulator selection	Selection of SMDS2/SMDS2+
JP5	User's Power selection	Selection of Power to User.

Table 17-2. Power Selection Settings for TB94C8/94C4

"To User_Vcc" Settings	Operating Mode	Comments
To user_Vcc off ⟨○●●⟩ on		The SMDS2/SMDS2+ main board supplies V_{CC} to the target board (evaluation chip) and the target system.
To user_Vcc off ⟨●●○⟩ on		The SMDS2/SMDS2+ main board supplies V_{CC} only to the target board (evaluation chip). The target system must have its own power supply.

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

17.2.3 SMDS2+ Selection (SAM8)

In order to write data into program memory that is available in SMDS2+, the target board should be selected to be for SMDS2+ through a switch as follows. Otherwise, the program memory writing function is not available.
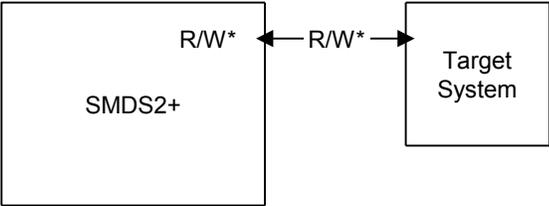
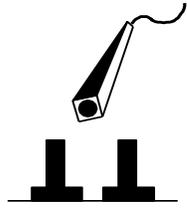**Table 17-3. The SMDS2+ Tool Selection Setting**

"JP4" Setting	Operating Mode
SMDS2 ●●○ SMDS2+	

Table 17-4. Using Single Header Pins to Select Clock Source/PWM/Operation Mode

Target Board Part	Comments
Board CLK JP1 Clock Source Inner CLK	Use SMDS2/SMDS2+ internal clock source as the system clock. <u>Default Setting</u>
Board CLK JP1 Clock Source Inner CLK	Use external crystal or ceramic oscillator as the system clock.
PWM Enable JP3 PWM Disable	PWM function is DISABLED.
PWM Enable JP3 PWM Disable	PWM function is ENABLED. <u>Default Setting</u>
Main Mode JP2 EVA Mode	The S3E94C0 run in main mode, just same as S3F94C8/S3F94C4. The debug interface is not available.
Main Mode JP2 EVA Mode	The S3E94C0 run in EVA mode, available. When debug program, please set the jumper in this mode. <u>Default Setting</u>

Table 17-5. Using Single Header Pins as the Input Path for External Trigger Sources

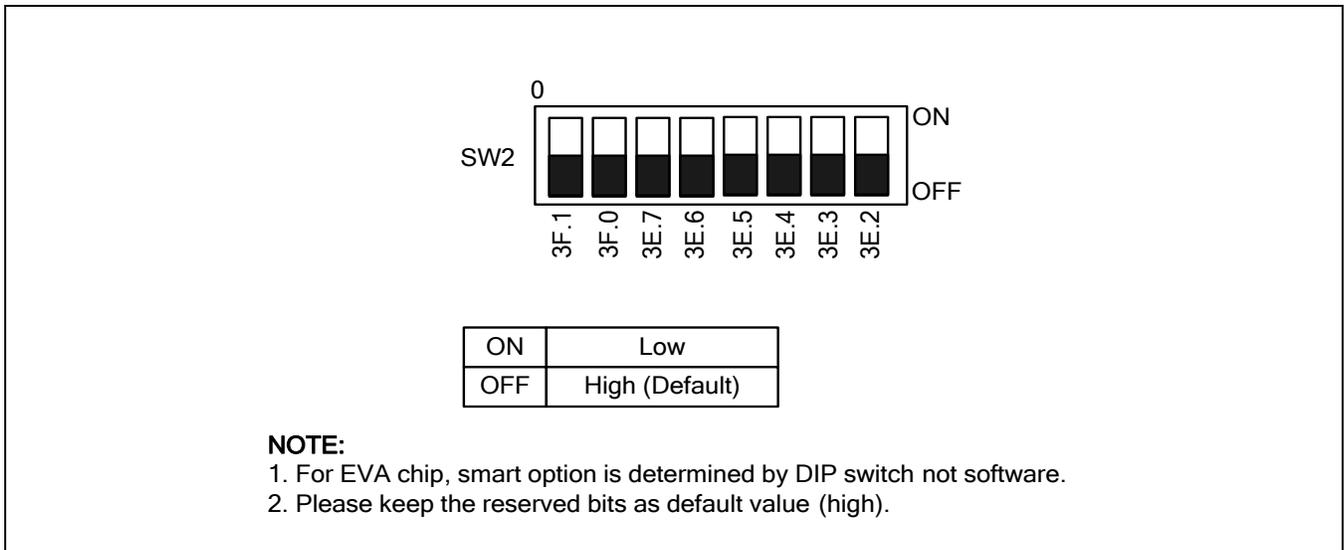| Target Board Part	Comments
External Triggers ◎ Ch1(TP3) ◎ Ch2(TP4)	Connector from External Trigger Sources of the Application System You can connect an external trigger source to one of the two external trigger channels (CH1 or CH2) for the SK-1000/SMDS2+ breakpoint and trace functions.

ON	Low
OFF	High (Default)

NOTE:
1. For EVA chip, smart option is determined by DIP switch not software.
2. Please keep the reserved bits as default value (high).

Figure 17-3. DIP Switch for Smart Option

Idle LED: This is LED is ON when the evaluation chip (S3E94C0) is in Idle Mode.

Stop LED: This LED is ON when the evaluation chip (S3E94C0) is in Stop Mode.
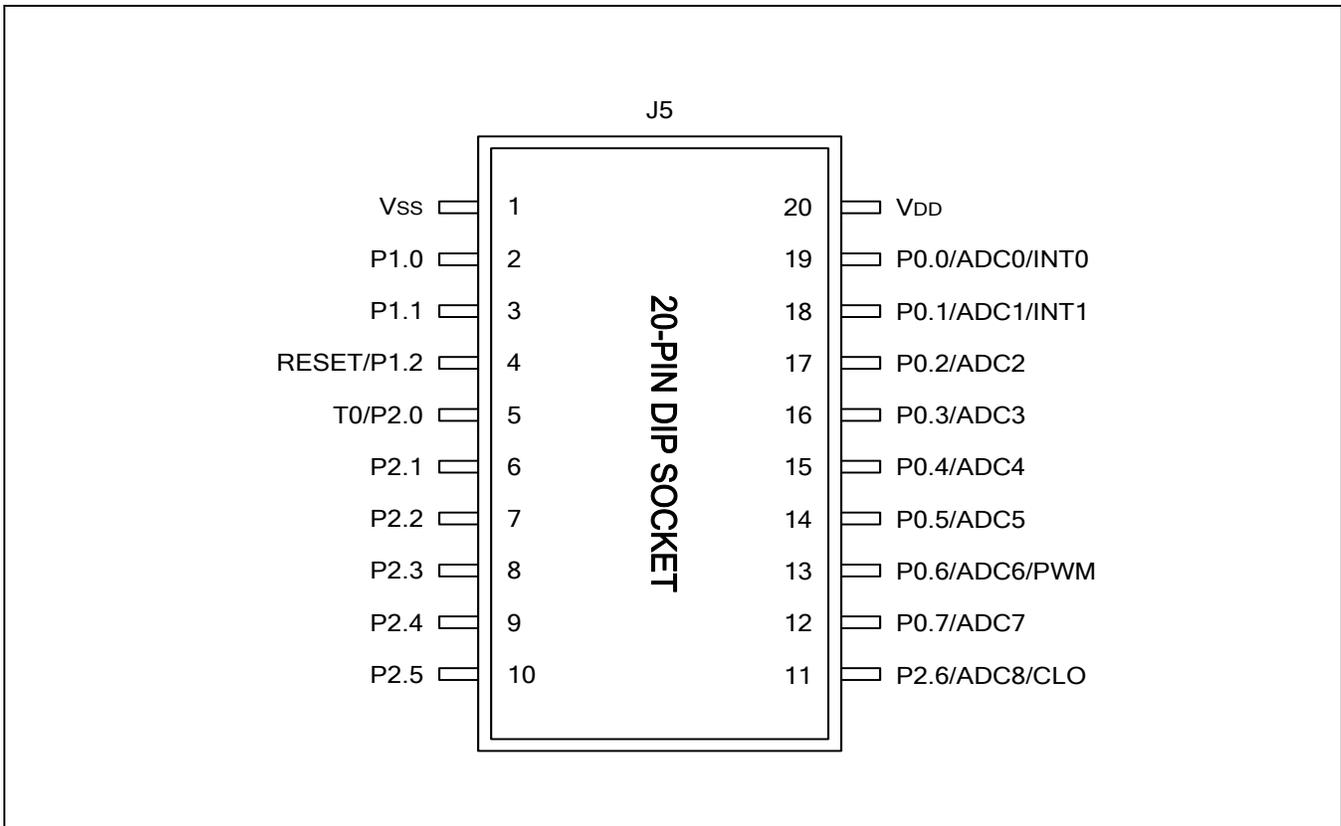
J5

Vss	1	20	VDD
P1.0	2	19	P0.0/ADC0/INT0
P1.1	3	18	P0.1/ADC1/INT1
RESET/P1.2	4	17	P0.2/ADC2
T0/P2.0	5	16	P0.3/ADC3
P2.1	6	15	P0.4/ADC4
P2.2	7	14	P0.5/ADC5
P2.3	8	13	P0.6/ADC6/PWM
P2.4	9	12	P0.7/ADC7
P2.5	10	11	P2.6/ADC8/CLO

20-PIN DIP SOCKET

Figure 17-4. 20-Pin Connector for TB94C8/94C4

Target Board

J101

20-Pin Connector

1   20

Target Cable for 20-Pin Connector

10   11

Target System

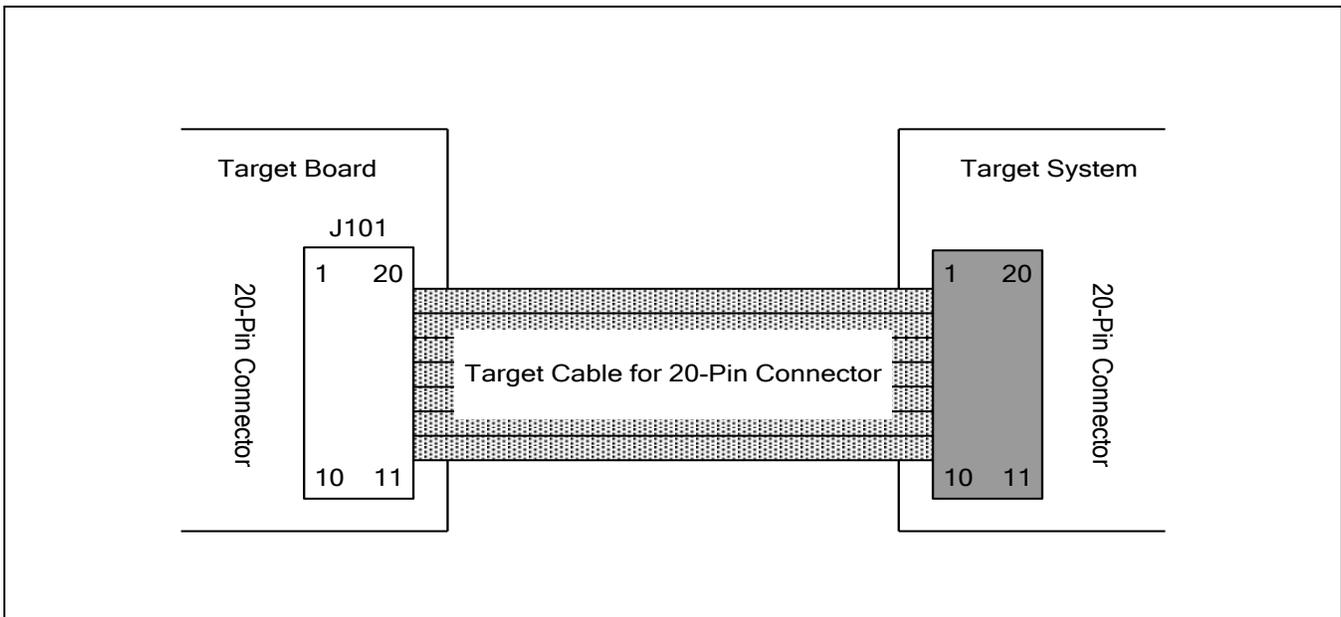20-Pin Connector

1   20

10   11

Figure 17-5. S3F94C8/S3F94C4 Probe Adapter for 20-DIP Package

17.2.4 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.

17.2.5 In-Circuit Emulators

- OPENice-i500/2000
- SK-1200 SmartKit

17.2.6 OTP/MTP Programmers

- GW-Uni2
- AS-Pro2
- Elnec programmers

To obtain the S3 Family development tools that will satisfy your S3F80QB development objectives, contact your local Zilog Sales Office, or visit Zilog's Third Party Tools page to review our list of third party tool suppliers.