zilog

Embedded in Life

An IXYS Company

S3 Family 8-Bit Microcontrollers

S3F8S45 MCU

Product Specification

PS032207-0418

PRELIMINARY

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

LIFE SUPPORT POLICY

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

As used herein

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

Document Disclaimer

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concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded.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

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

Revision History

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

Date	Revision Level	Description	Page
Apr 2018	07	Corrections to Port 0 Pull-up Resistor Enable Register; Updated Table 116 and added a note to Figure 138 for the XIN pin.	206,360, 361
Oct 2016	06	Corrected errors in Chapter 11; Added WDT features/register information; Replaced Figure 68; Added Watchdog Timer Counter Clock Source Selection section.	227, 228, 231
Jun 2015	05	Updated Figure 46 and Timer A, Timer C, and Timer D descriptions.	62, 235, 246, 251, 258
Jan 2015	04	Updated the Third Parties for Development Tools section; established chapter/section numbering for usability.	370
Mar 2014	03	Updated the Ordering Information table.	371
Feb 2014	02	Amended incorrect part numbers in Ordering Information table with respect to packaging containers for SDIP parts.	371
Nov 2013	01	Original Zilog issue.	All

Table of Contents

List of Figures

List of Tables

Chapter 1. Overview

Zilog's S3F8 Series of 8-bit single-chip microcontrollers offers a fast and efficient CPU, a wide range of integrated peripherals, and multiple Flash memory sizes. Important CPU features include:

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

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

1.1. S3F8S45 Microcontroller

The S3F8S45 MCU features 16KB of Flash ROM. Using a proven modular design approach, the S3F8S45 MCU was developed by integrating the following peripheral modules with the SAM88 core:

- Five programmable I/O ports, including three 8-bit ports and two 7-bit ports for a total of 38 pins
- Eight bit-programmable pins for external interrupts
- One 8-bit basic timer for oscillation stabilization and watchdog functions (system reset)
- Three 8-bit timer/counters and two 16-bit timer/counters with selectable operating modes
- Watch timer for real time
- LCD controller/driver
- A/D converter with 9 selectable input pins
- Synchronous SIO modules
- Two asynchronous UART modules
- Pattern generation module
- Battery level detector

The S3F8S45 MCU is currently available in 44-pin QFP, 42-pin SDIP, and 32-pin SDIP packages.

1.2. Features

The S3F8S45 MCU offers the following features:

- SAM88 RC CPU core
- Program memory (full Flash ROM):
  - 16K x 8 bits program memory
  - Internal Flash (program) memory
    - Sector size: 128 bytes
    - 10 years data retention
    - Fast programming time; user program and sector erase available
    - Endurance: 10,000 erase/program cycles
    - External serial programming support
    - Expandable On-Board Program (OBP) sector
- Data memory (RAM)
  - LCD display data memory
  - 528 x 8 bits data memory
- Instruction set
  - 78 instructions
  - IDLE and STOP instructions added for power-down modes
- 38 I/O pins
  - 8 pins shared with other signal pins
  - 30 pins shared with LCD signal outputs
- Interrupts
  - 8 interrupt levels and 22 interrupt sources
  - Fast interrupt processing feature
- 8-bit basic timer
  - Watchdog timer function
- 2 watchdog timer clock sources:
  - Basic timer
  - 32 kHz internal RC oscillator, selectable by the Smart Option
- 8-bit Timer/Counter A:
  - Programmable 8-bit internal timer
  - External event counter function
  - PWM and capture function

- 8-bit Timer/Counter B:
  - Programmable 8-bit internal timer
  - Carrier frequency generator
- 8-bit Timer/Counter C:
  - Programmable 8-bit internal timer
  - PWM function
- Two 16-bit Timer/Counters (D0/D1):
  - Programmable 16-bit internal timer
  - External event counter function
  - PWM and capture function
- Watch timer:
  - Interval times: 3.91 ms, 0.25 s, 0.5 s, and 1.0 s at 32.768 kHz
  - 0.5/1/2/4 kHz selectable buzzer output
- LCD controller/driver
  - 22 segments and 8 common terminals
  - 1/2, 1/3, 1/4, and 1/8 duty selectable
  - Internal resistor bias
  - 16-step contrast control possible
- Analog-to-digital converter
  - 9-channel analog input
  - 10-bit conversion resolution
  - 25 µs conversion time
- Two UART channels
  - Full-duplex serial I/O interface
  - Four programmable operating modes
  - Autogenerating parity bit
- 8-bit serial I/O interface
  - 8-bit transmit/receive mode
  - 8-bit receive mode
  - LSB-first or MSB-first transmission selectable
  - Internal or external clock source
- Pattern generation module
  - Triggered by timer match signal and software

- Low Voltage Reset (LVR)
  - Criteria voltage: 1.9V, 2.6V, 3.3V, 3.9V
  - Included power-on reset
  - Enable/disable by Smart Option (ROM address: 3Fh)
- Battery Level Detector (BLD)
  - Criteria voltage: 2.4V, 2.7V, 3.3V, 3.9V
- Oscillation sources
  - Crystal, ceramic, or RC for main clock
  - Main clock frequency: 0.4MHz–12.0MHz
  - 32.768kHz crystal oscillation circuit for subclock
- Two power-down modes
  - IDLE: only CPU clock stops
  - STOP: selected system clock and CPU clock stop
- Instruction execution times
  - 333ns at f_X = 12.0MHz (minimum)
  - 122.1µs at f_{XT} = 32.768kHz (minimum)
- Internal voltage converter (IVC) for 5V operations
- Smart Option
  - Watchdog timer clock source, P0.2/nRESET pin, LVR level, and LVR enable/disable selectable (ROM address 3Fh)
  - ISP related option selectable (ROM address 3Eh)
- Operating voltage range
  - 1.8V to 5.5V at 0.4–4.2MHz
  - 2.2V to 5.5V at 0.4–12.0MHz
- Operating temperature range: 40°C to +85°C
- Packages
  - 44-pin QFP-1010
  - 42-pin SDIP-600
  - 32-pin SDIP-400

1.3. Block Diagram

Figure 1 shows a block diagram for the S3F8S45 MCU, which is available in a 44-pin package.

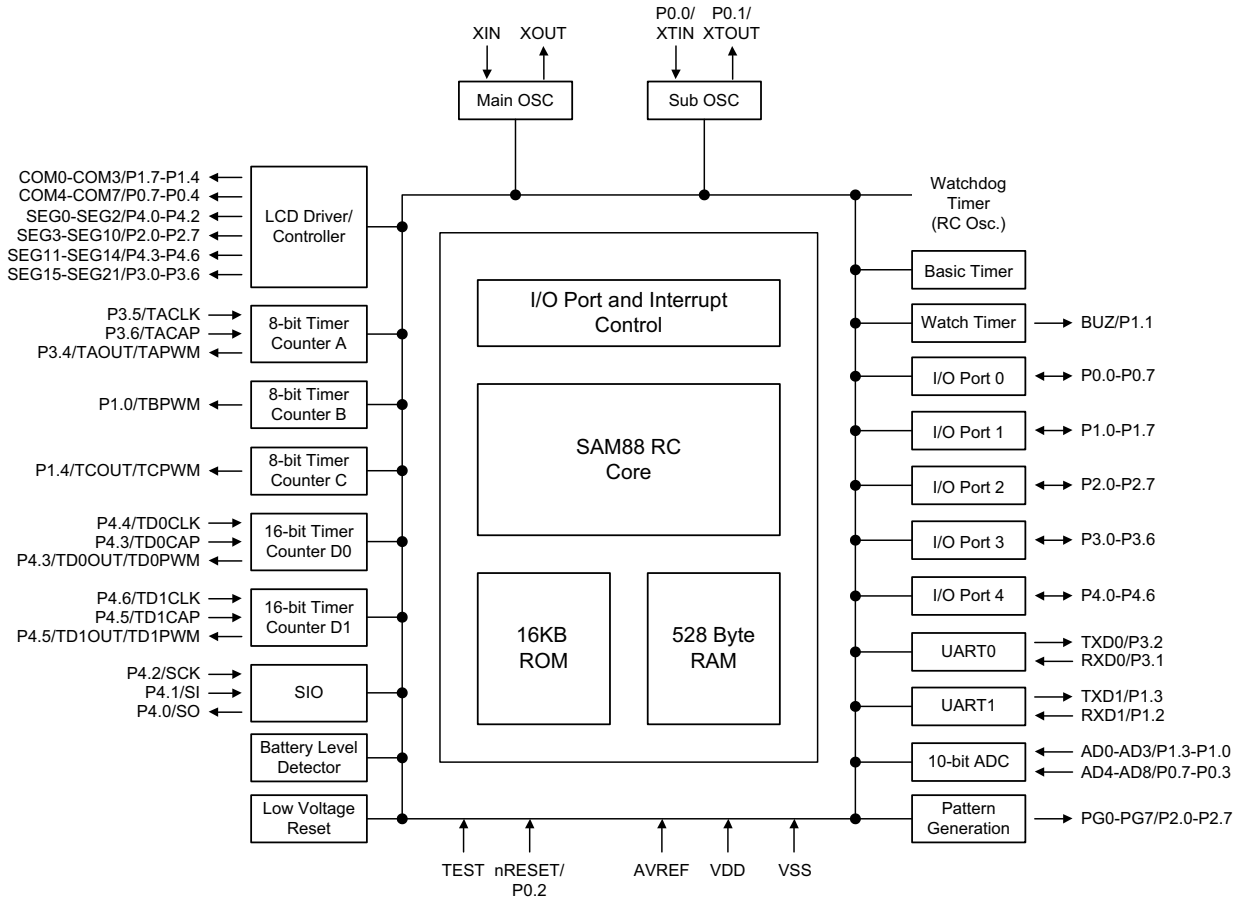


Figure 1. S3F8S45 MCU Block Diagram

1.4. Pin Assignments

Figure 2 shows the pin assignments for the 44-pin ELP package.

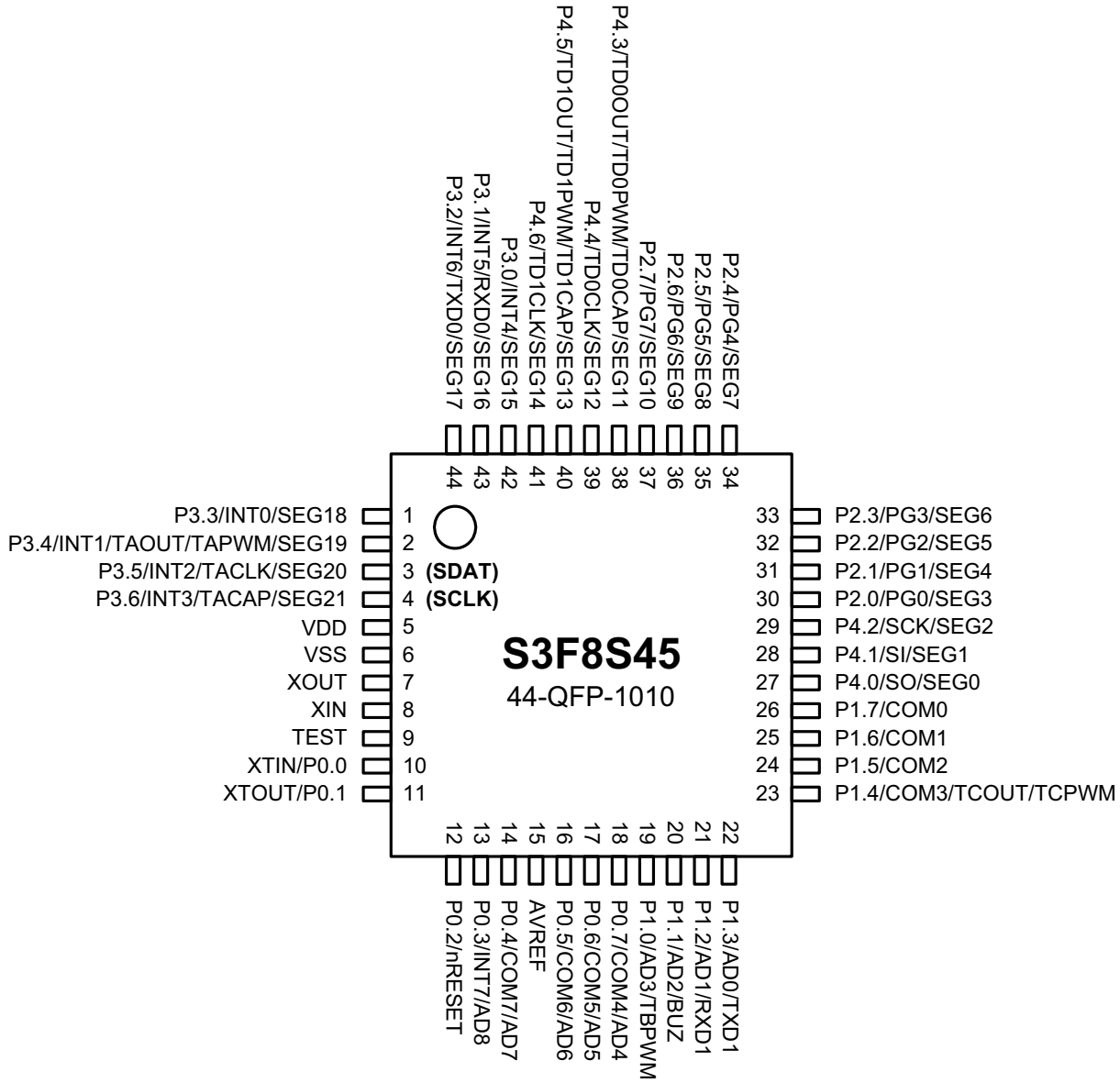


Figure 2. Pin Assignments, 44-Pin QFP Package

Figure 3 shows the pin assignments for the 42-pin SDIP package.

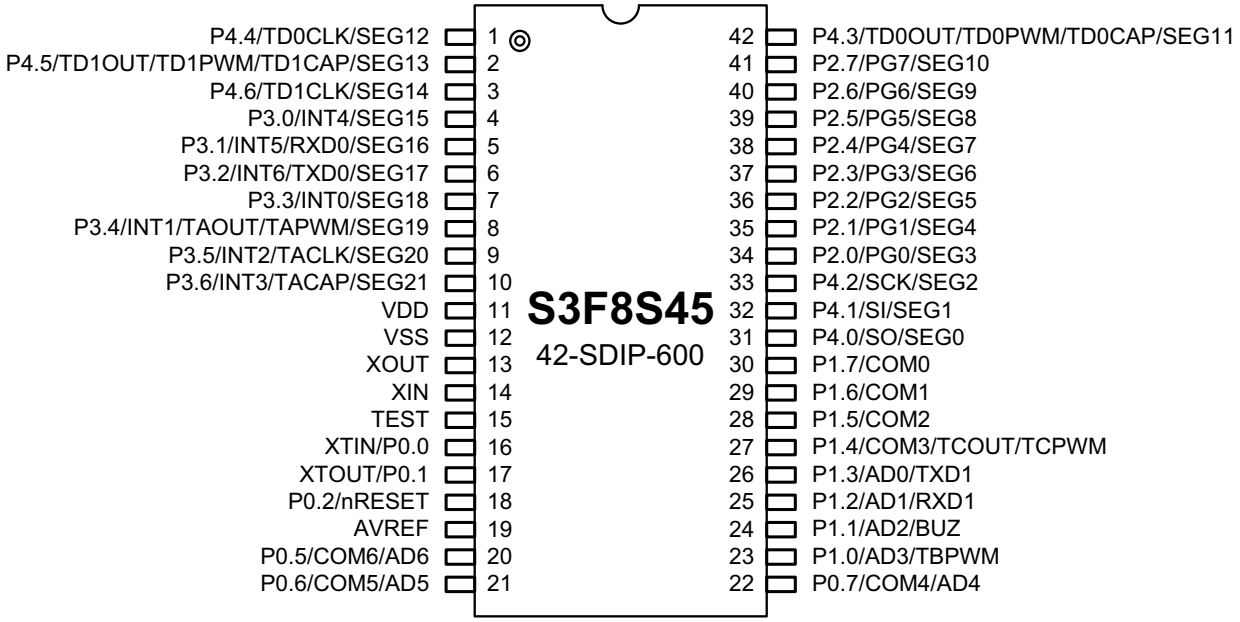


Figure 3. Pin Assignments, 42-Pin SDIP Package

Figure 4 shows the pin assignments for the 32-pin SDIP package.

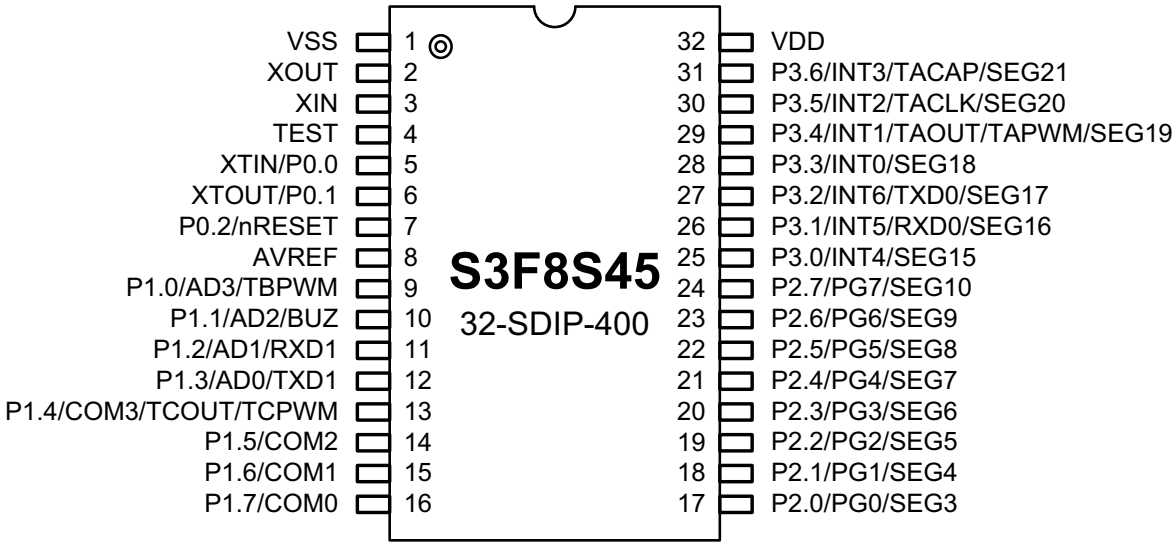


Figure 4. Pin Assignments, 32-Pin SDIP Package

Table 1 identifies each pin in the S3F8S45 MCU's 44-pin QFP package.

Table 1. S3F8S45 Pin Descriptions

Pin Names	Pin Description	Pin Type	Circuit Type	Pin Numbers 44/42-Pin	Pin Numbers 32-Pin	Shared Pins
P0.0 P0.1 P0.2 P0.3 P0.4 P0.5 P0.6 P0.7	I/O port with 1-bit-programmable pins; Schmitt trigger input or push-pull output and software assignable pull-ups. P0.3 is alternately used for external interrupt input (noise filters, interrupt enable and pending control). P0.3–P0.4 are not in the 42-pin package. P0.3–P0.7 are not in the 32-pin package.	I/O	D-2	10(16) 11(17)	5 6	X_{TIN} X_{TOUT}
			D-1	12(18)	7	nRESET
			F-1	13(–)	–	INT7/AD8
			H-2	14(–) 16(20) 17(21) 18(22)	– 	COM7/AD7 COM6/ AD6 COM5/AD5 COM4/AD4
P1.0 P1.1 P1.2 P1.3	I/O port with 1-bit-programmable pins; Schmitt trigger input or push-pull, open-drain output and software assignable pull-ups.	I/O	F-2	19(23) 20(24) 21(25) 22(26)	9 10 11 12	AD3/TBPWM AD2/ BUZ AD1/RXD1 AD0/TXD1
P1.4 P1.5 P1.6 P1.7			H-3	23(27) 24(28) 25(29) 26(30)	13 14 15 16	COM3/TCOUT/ TCPWM COM2 COM1 COM0
P2.0–P2.7	I/O port with 1-bit-programmable pins; Schmitt trigger input or push-pull output and software assignable pull-ups.	I/O	H-4	30–37 (34–41)	17–24	PG0–PG7 SEG3–SEG10
P3.0 P3.1 P3.2 P3.3 P3.4 P3.5 P3.6	I/O port with 1-bit-programmable pins; Schmitt trigger input or push-pull, open-drain output and software assignable pull-ups. Alternately used for external interrupt input (noise filters, interrupt enable and pending control).	I/O	H-3	42(4) 43(5) 44(6) 1(7) 2(8) 3(9) 4(10)	25 26 27 28 29 30 31	INT4/SEG15 INT5/RXD0/SEG16 INT6/TXD0/SEG17 INT0/SEG18 INT1/TAOUT/ TAPWM/SEG19 INT2/TACLK/ SEG20 INT3/TACAP/ SEG21

Note: Parentheses indicate pin numbers for the 42-pin SDIP package.

Table 1. S3F8S45 Pin Descriptions (Continued)

Pin Names	Pin Description	Pin Type	Circuit Type	Pin Numbers 44/42-Pin	Pin Numbers 32-Pin	Shared Pins
P4.0 P4.1 P4.2 P4.3 P4.4 P4.5 P4.6	I/O port with 1-bit-programmable pins; Schmitt trigger input or push-pull, open-drain output and software assignable pull-ups. The P4 is not in the 32-pin package.	I/O	H-3	27(31) 28(32) 29(33) 38(42) 39(1) 40(2) 41(3)	–	SO/SEG0 SI/SEG1 SCK/SEG2 TD0OUT/TD0PWM TD0CAP/SEG11 TD0CLK/SEG12 TD1OUT/TD1PWM TD1CAP/SEG13 TD1CLK/SEG14
COM0–COM2 COM3	LCD common signal output.	I/O	H-3	26–24 (30–28) 23(27)	16–14 13	P1.7–P1.5 P1.4/TCOUT/ TCPWM
COM4–COM6 COM7			H-2	18–16 (22–20) 14(–)	–	P0.7–P0.5/ AD4–AD6 P0.4/AD7
SEG0 SEG1 SEG2	LCD segment signal output.	I/O	H-3	27(31) 28(32) 29(33)	–	P4.0/SO P4.1/SI P4.2/SCK
SEG3–SEG10			H-4	30–37 (34–41)	17–24	P2.0–P2.7/ PG0–PG7
SEG11 SEG12 SEG13 SEG14 SEG15 SEG16 SEG17 SEG18 SEG19 SEG20 SEG21			H-3	38(42) 39(1) 40(2) 41(3) 42(4) 43(5) 44(6) 1(7) 2(8) 3(9) 4(10)	– – – – 25 26 27 28 29 30 31	P4.3/TD0OUT/ TD0PWM/TD0CAP P4.4/TD0CLK P4.5/TD1OUT/ TD1PWM/TD1CAP P4.6/TD1CLK P3.0/INT4 P3.1/INT5/RXD0 P3.2/INT6/TXD0 P3.3/INT0 P3.4/INT1/ TAOUT/TAPWM P3.5/INT2/TACLK P3.6/INT3/TACAP

Note: Parentheses indicate pin numbers for the 42-pin SDIP package.

Table 1. S3F8S45 Pin Descriptions (Continued)

Pin Names	Pin Description	Pin Type	Circuit Type	Pin Numbers 44/42-Pin	Pin Numbers 32-Pin	Shared Pins
AD0	LCD segment signal output.	I/O	F-2	22(26)	12	P1.3/TXD1
AD1			H-2	21(25)	11	P1.2/RXD1
AD2				20(24)	10	P1.1/BUZ
AD3				19(23)	9	P1.0/TBPWM
AD4–AD6				18–16 (22–20)	–	P0.7–P0.5/ COM4–COM6
AD7				14	–	P0.4/COM7
AD8				13	–	P0.3/INT7
SCK	Serial interface clock.	I/O	H-3	29(33)	–	P4.2/SEG2
SO	Serial interface data output.	I/O	H-3	27(31)	–	P4.0/SEG0
SI	Serial interface data input.	I/O	H-3	28(32)	–	P4.1/SEG1
BUZ	Output pin for buzzer signal.	I/O	F-2	20(24)	10	P1.1/AD2
TXD0	UART0 data output, input.	I/O	H-3	44(6)	27	P3.2/INT6/SEG17
RXD0				43(5)	26	P3.1/INT5/SEG16
TXD1 RXD1	UART1 data output, input.	I/O	F-2	22(26) 21(25)	12 11	P1.3/AD0 P1.2//AD1
PG0–PG7	Pattern generation output.	I/O	H-4	30–37 (34–41)	17–24	P2.0–P2.7/ SEG3–SEG10
TAOUT/ TAPWM	Timer A clock and PWM output.	I/O	H-3	2(8)	29	P3.4/INT1/SEG19
TACLK	Timer A external clock input.	I/O	H-3	3(9)	30	P3.5/INT2/SEG20
TACAP	Timer A capture input.	I/O	H-3	4(10)	31	P3.6/INT3/SEG21
TBPWM	Timer B PWM output.	I/O	F-2	19(23)	9	P1.0/AD3
TCOUT/ TCPWM	Timer C clock and PWM output.	I/O	H-3	23(27)	13	P1.4/COM3
TD0OUT/ TD0PWM	Timer D0 clock and PWM output.	I/O	H-3	38(42)	–	P4.3/TD0CAP/ SEG11
TD0CLK	Timer D0 external clock input.	I/O	H-3	39(1)	–	P4.4/SEG12
TD1CAP	Timer D1 capture input.	I/O	H-3	40(2)	–	P4.5/TD1OUT/ TD1PWM/SEG13

Note: Parentheses indicate pin numbers for the 42-pin SDIP package.

Table 1. S3F8S45 Pin Descriptions (Continued)

Pin Names	Pin Description	Pin Type	Circuit Type	Pin Numbers 44/42-Pin	Pin Numbers 32-Pin	Shared Pins
INT0	External interrupt input pins.	I/O	H-3	1(7)	28	P3.3/SEG18
INT1				2(8)	29	P3.4/TAOUT/ TAPWM/SEG19
INT2				3(9)	30	P3.5/TACLK/SEG20
INT3				4(10)	31	P3.6/TACAP/SEG21
INT4				42(4)	25	P3.0/SEG15
INT5				43(5)	26	P3.1/RXD0/SEG16
INT6				44(6)	27	P3.2/TXD0/SEG17
INT7			F-1	13	–	P0.3/AD8
AVREF	A/D converter reference voltage.	–	–	15(19)	8	–
nRESET	System reset pin with a pull-up resistor when it is selected as the nRESET by smart option.	I	B	12(18)	7	P0.2
XIN	Main oscillator pins.	–	–	8(14)	3	–
XOUT				7(13)	2	
X_{TIN}	Crystal oscillator pins for subclock	–	–	10(16)	5	P0.0
X_{TOUT}				11(17)	6	P0.1
TEST	Test pin: it must be connected to VSS.	I	–	9(15)	4	–
VDD	Power supply input pin.	–	–	5(11)	32	–
VSS	Ground pin.	–	–	6(12)	1	–

Note: Parentheses indicate pin numbers for the 42-pin SDIP package.

1.5. Pin Circuits

Figure 5 shows the Type A pin circuit for Port 0.

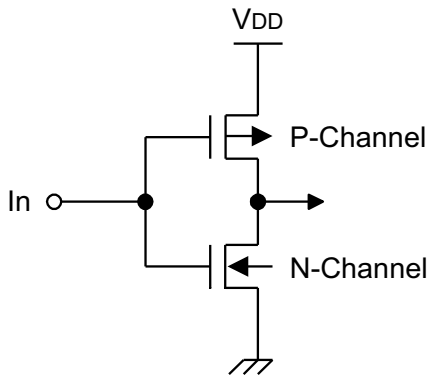


Figure 5. Pin Circuit, Type A

Figure 6 shows the Type B pin circuit.

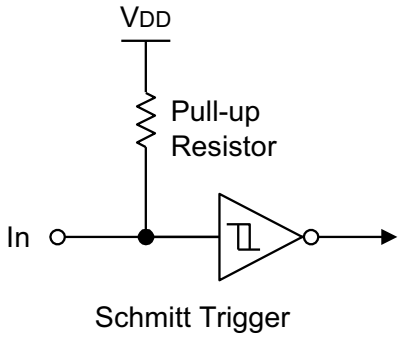


Figure 6. Pin Circuit, Type B

Figure 7 shows Type C pin circuit.

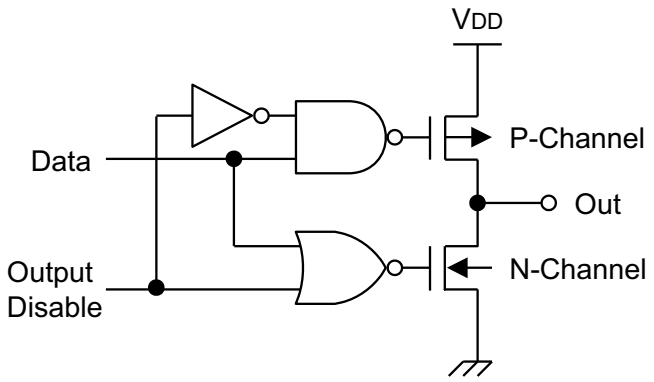


Figure 7. Pin Circuit, Type C

Figure 8 shows the Type D-1pin circuit for Port 0.2.

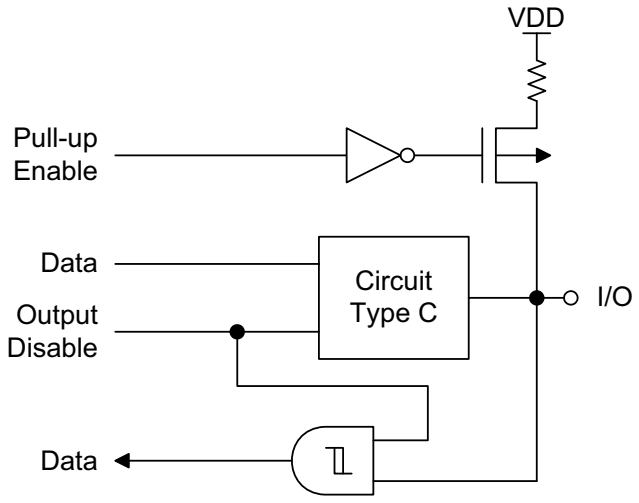


Figure 8. Pin Circuit, Type D-1

Figure 9 shows the Type D-2 pin circuit for P0.0–0.1.

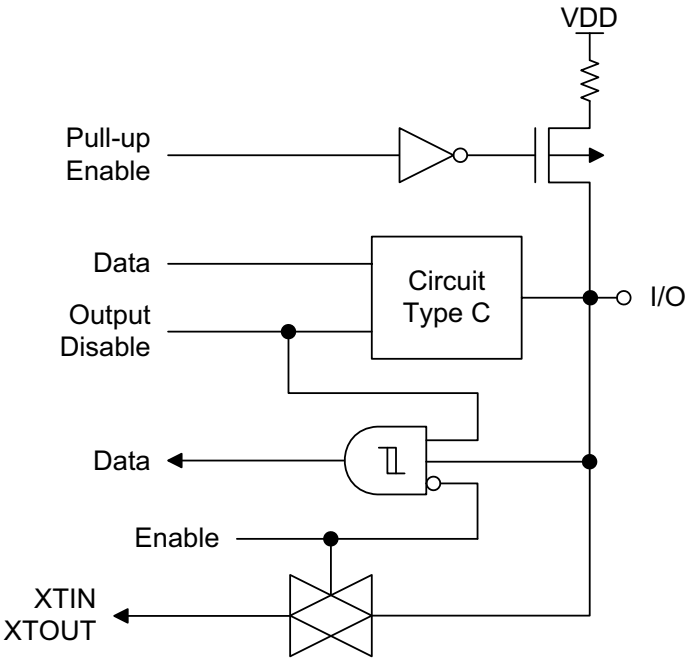


Figure 9. Pin Circuit Type D-2

Figure 10 shows the Type F-1 pin circuit for P0.3.

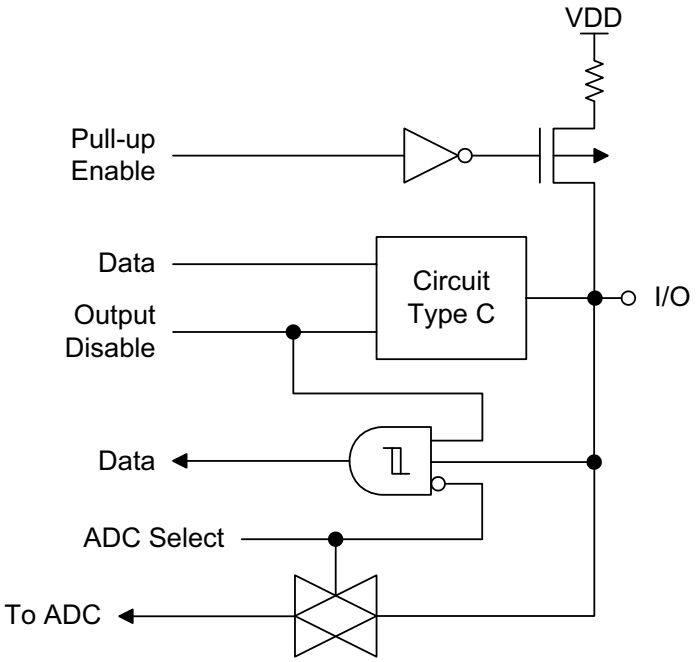


Figure 10. Pin Circuit Type F-1

Figure 11 shows the Type F-2 pin circuit for P1.0 and P1.3.

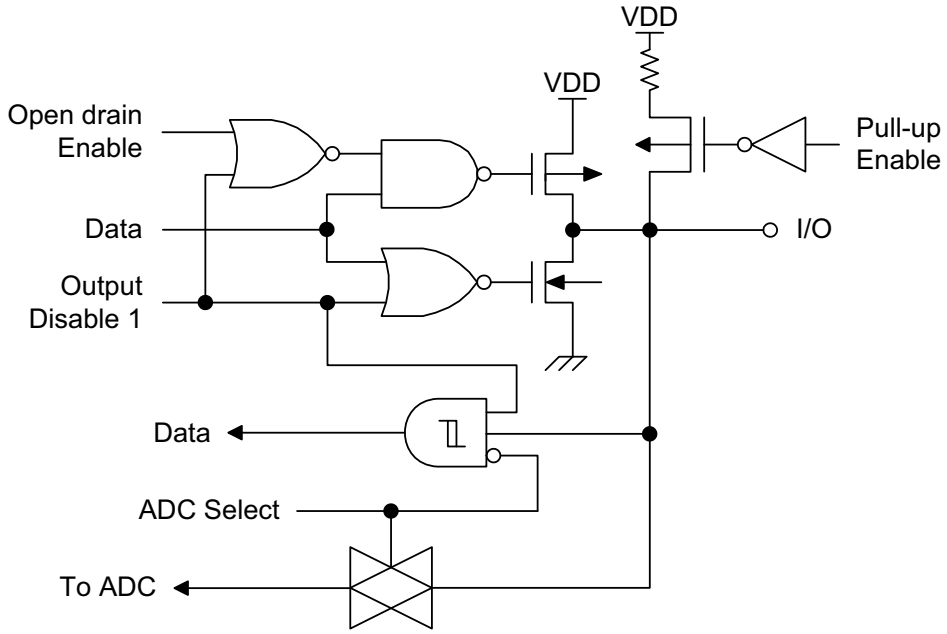


Figure 11. Pin Circuit Type F-2 (P1.0–P1.3)

Figure 12 shows the Type H-1 pin circuit.

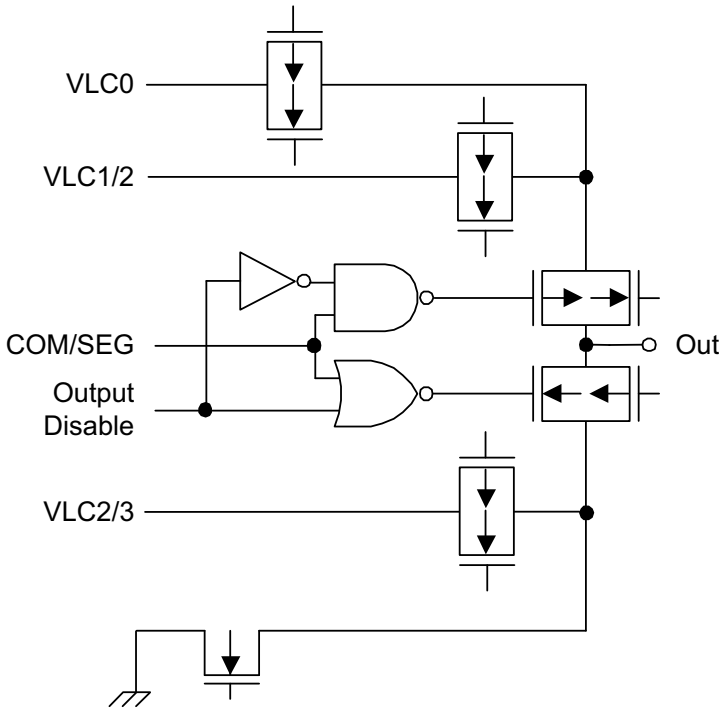


Figure 12. Pin Circuit Type H-1

Figure 13 shows the Type H-2 pin circuit for P0.4–P0.7.

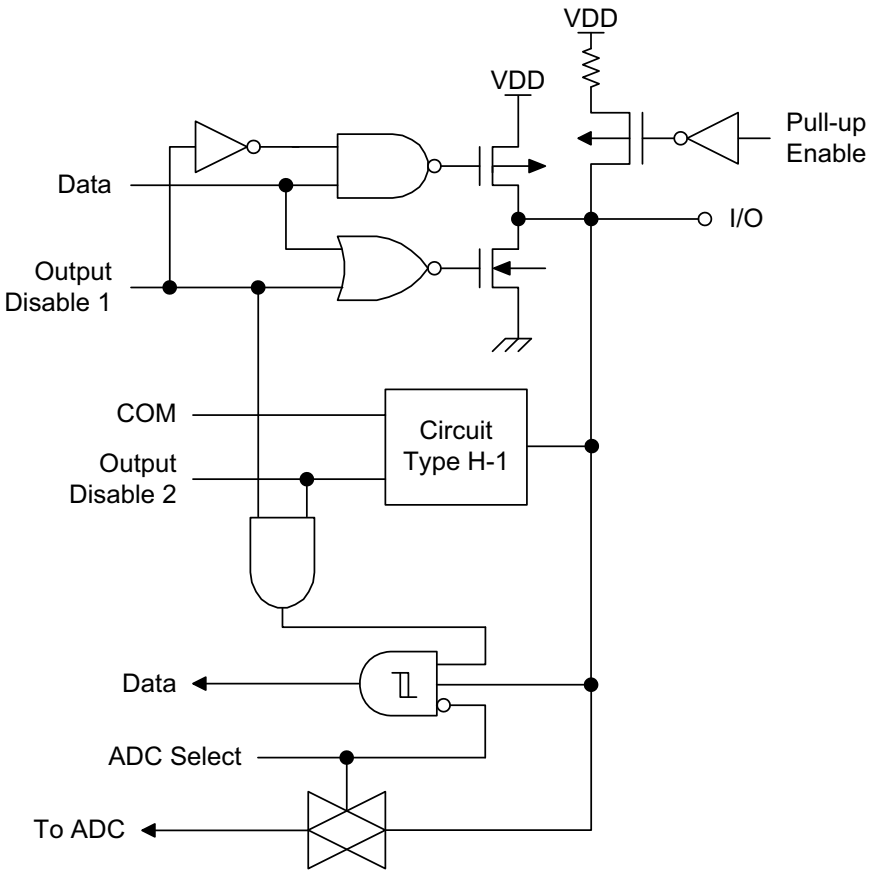


Figure 13. Pin Circuit Type H-2

Figure 14 shows the Type H-3 pin circuit for P1.4–P1.7, P3.0–P3.6, and P4.0–P4.6.

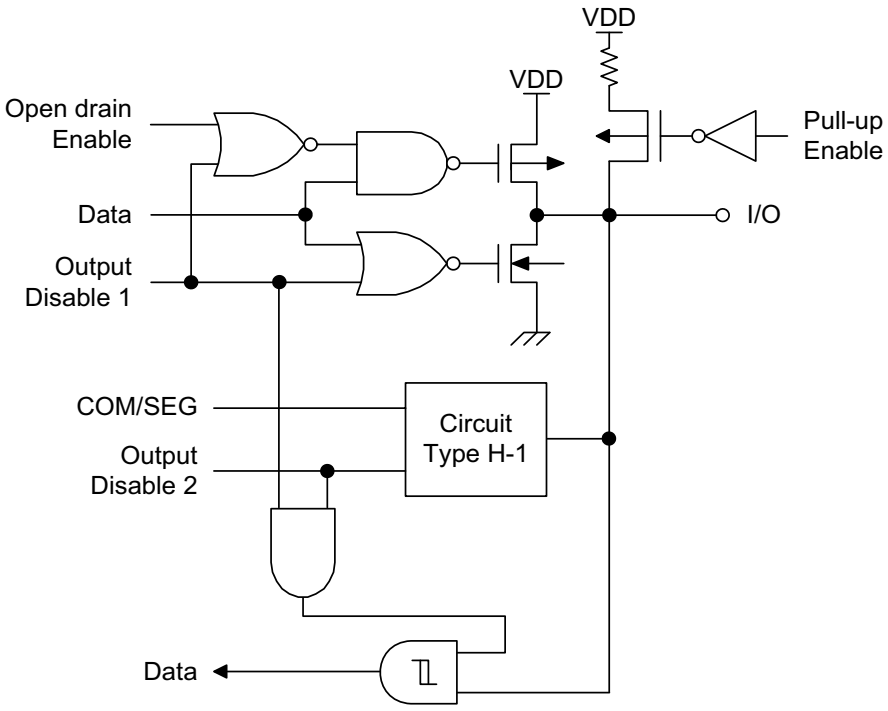


Figure 14. Pin Circuit Type H-3

Figure 15 shows the Type H-4 pin circuit for P2.0–P2.7.

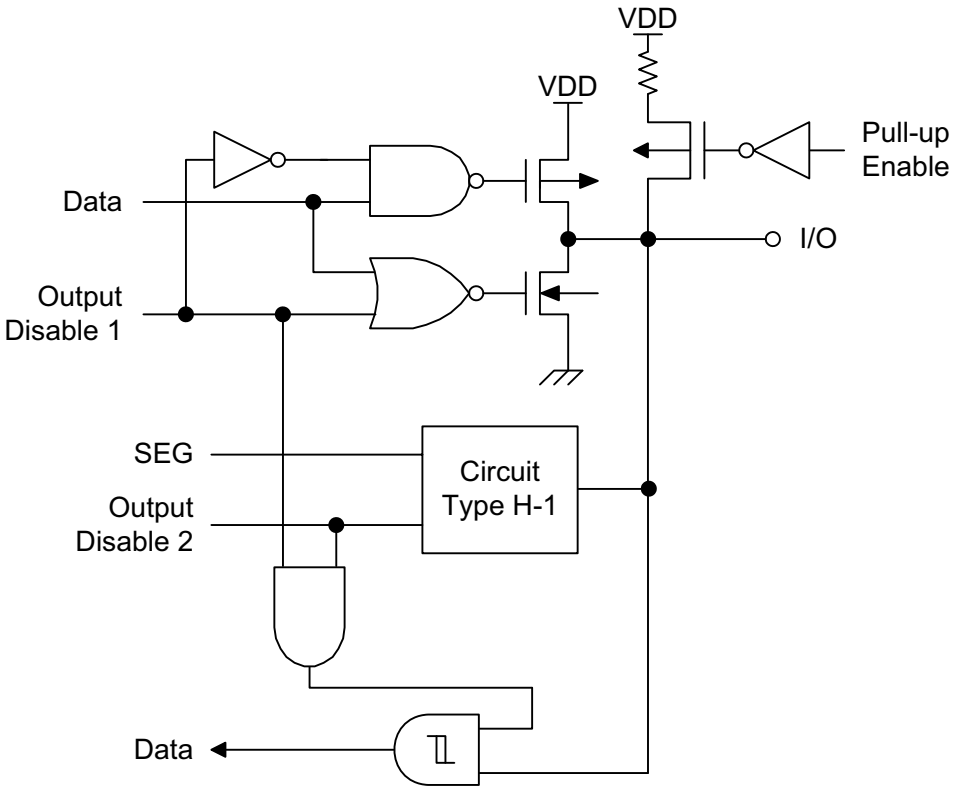


Figure 15. Pin Circuit Type H-4

Chapter 2. Address Space

The S3F8S45 microcontroller features two types of address space:

- Internal program ROM
- Internal register file

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

The S3F8S45 MCU features 16KB of internal Flash ROM. The 256-byte physical register space is expanded into an addressable area of 320 bytes using addressing modes.

A 22-byte LCD display register file is implemented.

2.1. Program Memory

Program memory (ROM) stores program codes or table data. The S3F8S45 MCU features 16KB internal Flash program memory.

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

The ROM address at which program execution starts after a reset is `0100h`.

The reset address of ROM can be changed by the smart option only in the S3F8S45 (full-Flash) device. Refer to the Embedded Flash Memory Interface chapter on page 327 to learn more.

The program memory address space is shown in Figure 16.

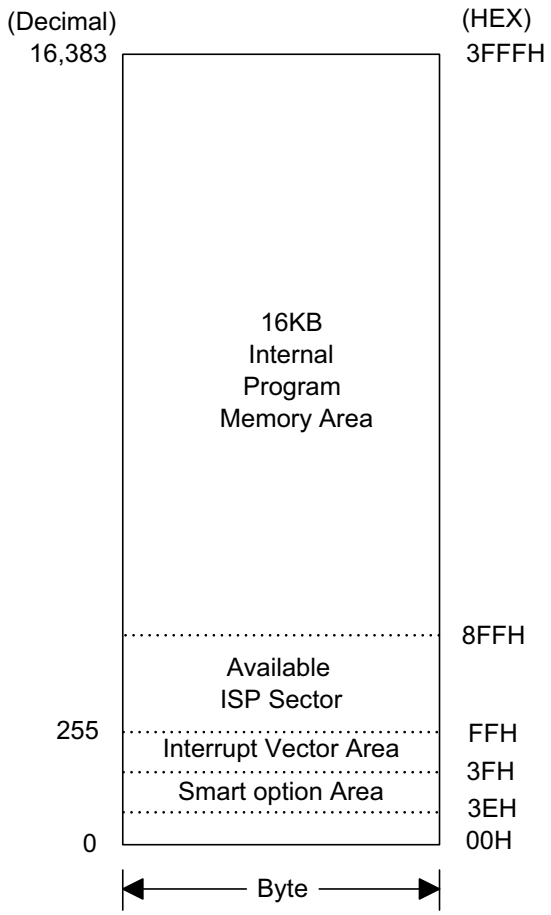


Figure 16. Program Memory Address Space

2.2. Smart Option

The *Smart Option,* diagrammed in Figure 17, is the ROM option for the start condition of the chip. The ROM address used for the smart option ranges from 003Ch to 003Fh. The S3F8S45 MCU uses only addresses in the range 003Eh to 003Fh.

When any values are written in the Smart Option area (i.e., 003Ch–003Fh) by an LDC instruction, the data in the area may be changed, but the Smart Option is not affected. Smart Option data should be written in this Smart Option area using an OTP/MTP programming tool.

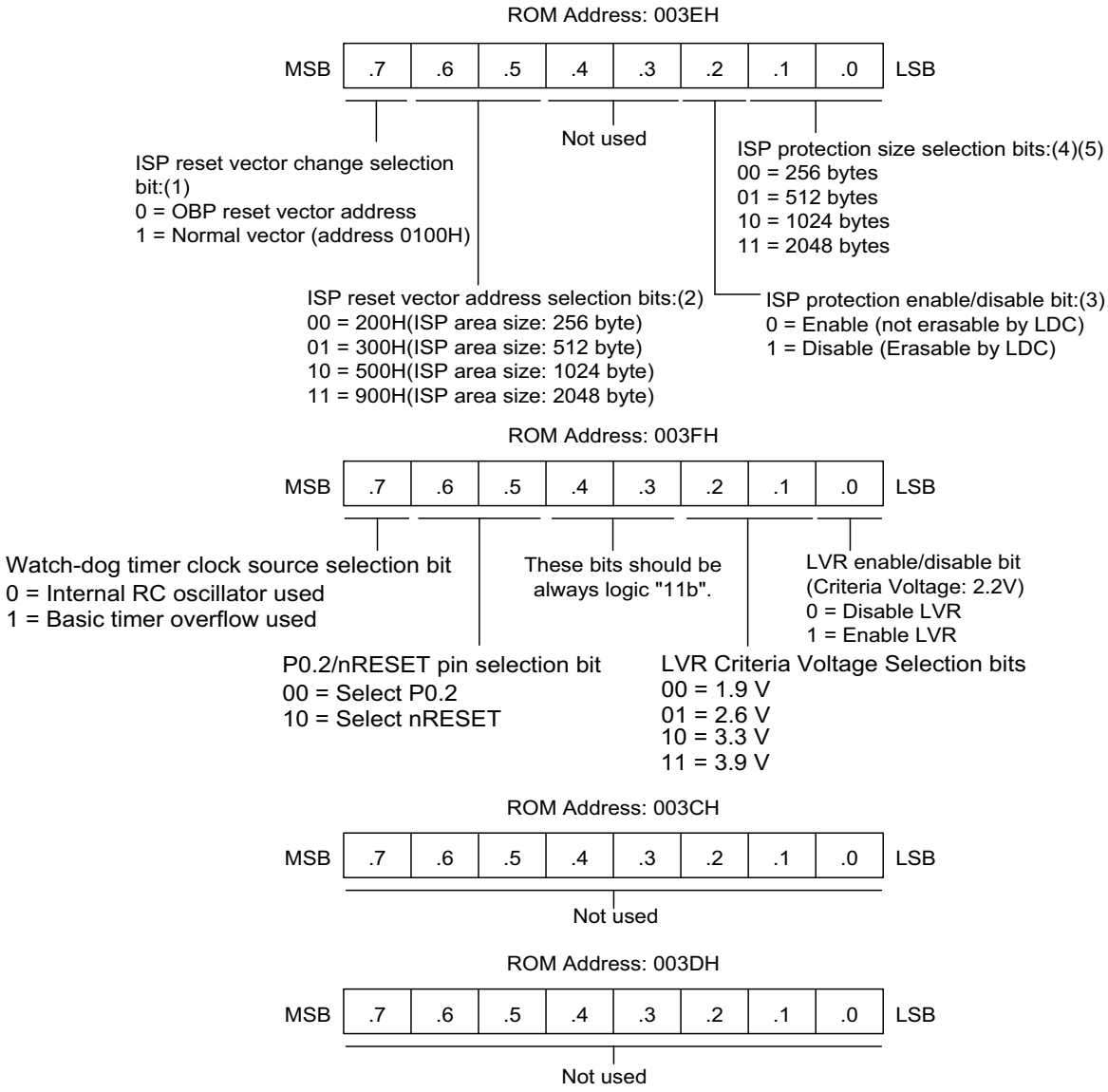


Figure 17. Smart Option

Notes: 1. In Figure 17, by setting ISP reset vector change selection bit (3E.7) to 0, the ISP area becomes available. If this bit is 1, 3Eh.6 and 3Eh.5 are rendered meaningless.

2. If the ISP reset vector change selection bit (3Eh.7) is 0, the user must change the ISP reset vector address from 0100h to an address for which the user wants to set a reset

address (i.e., 0200h, 0300h, 0500h, or 0900h). If the reset vector address is 0200h, the ISP area can be assigned from 0100h to 01FFh (an area of 256 bytes). If 0300h, the ISP area can be assigned from 0100h to 02FFh (512 bytes). If 0500h, the ISP area can be assigned from 0100h to 04FFh (1024 bytes). If 0900h, the ISP area can be assigned from 0100h to 08FFh (2048 bytes).

3. If the ISP protection enable/disable bit is 0, user cannot erase or program the ISP area selected by 3Eh.1 and 3Eh.0 in Flash memory.
4. The user can select a suitable ISP protection size using 3Eh.1 and 3Eh.0. If the ISP protection enable/disable bit (3Eh.2) is 1, 3Eh.1 and 3Eh.0 are rendered meaningless.
5. After selecting the ISP reset vector address when selecting the ISP protection size, do not select a value greater than the ISP area size.
6. If the P0.2/nRESET pin is selected as P0.2 by the Smart Option, then current consumption is added by the POR logic.

2.3. Register Architecture

In the S3F8S45 implementation, the upper 64-byte area of register files is expanded to two 64-byte areas, called *Set1* and *Set2*. The upper 32-byte area of Set1 is further expanded to two 32-byte register banks (Bank0 and Bank1), and the lower 32-byte area is a single 32-byte common area.

In the S3F8S45 MCU, the total number of addressable 8-bit registers is 617. Of these 617 registers, 13 bytes are designated for the CPU and system control registers, 76 bytes are designated for the peripheral control and data registers, 16 bytes are used as shared working registers, and 512 registers are designated for general-purpose use in page 0–page 1 (including 22 bytes for LCD display registers).

Set1 register locations can always be set, regardless of which of the ten register pages is currently selected. A Set1 location, however, can only be addressed using register addressing modes.

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

Specific register types and the area occupied in the S3F8S45 internal register space are summarized in Table 2.

Table 2. S3F8S45 Register Types

Register Type	Number of Bytes
General-purpose registers, including the 16-byte common working register area, the four 192-byte prime register areas (including LCD data registers), and four 64-byte Set 2 areas.	528
CPU and system control registers.	13
Mapped clock, peripheral, I/O control, and data registers.	76
Total addressable bytes.	617

Figure 18 shows the organization of the internal register file.

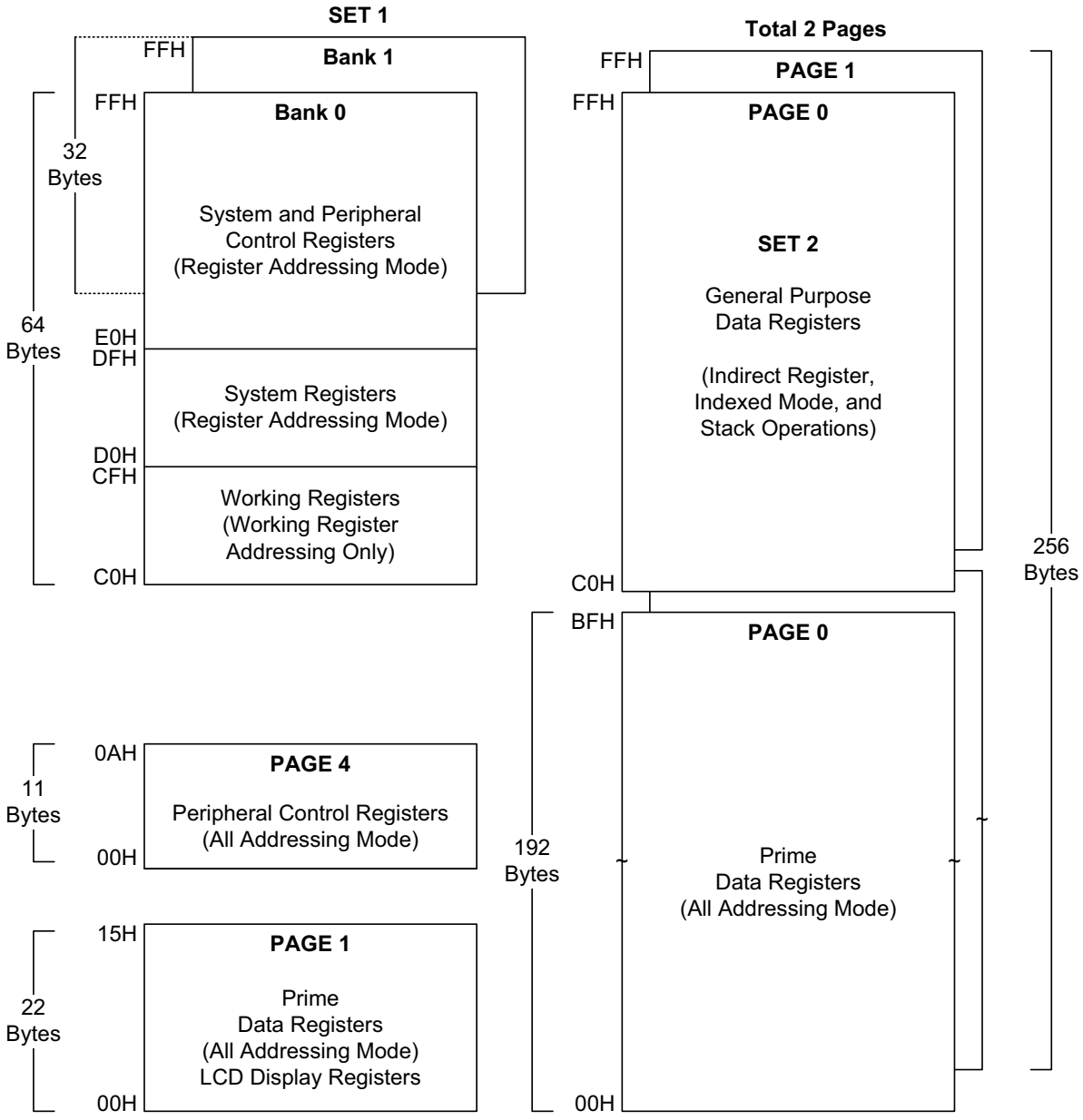


Figure 18. Internal Register File Organization

2.4. Register Page Pointer

The S3F8 Series architecture supports the logical expansion of the physical 256-byte internal register files (using an 8-bit data bus) into as many as 16 separately-addressable register pages. Page addressing is controlled by the Register Page Pointer (PP at address DFh). In the S3F8S45 microcontroller, a paged register file expansion is implemented for LCD data registers, and the register page pointer must be changed to address other pages.

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

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

Table 3. Register Page Pointer (PP; Set1, Bank0)

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

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

Bit	Description
[7:4]	**Destination Register Page Selection Bits**[1] 0000: Destination: Page 0. 0001: Destination: Page 1. 0100: Destination: Page 4. 0101–1111: Reserved.
[3:0]	**Source Register Page Selection Bits**[2] 0000: Source: Page 0. 0001: Source: Page 1. 0100: Source: Page 4. 0101–1111: Reserved.

Notes:
1. In the S3F8S45 microcontroller, the internal register file is configured as three pages (pages 0–1, 4).
2. Pages 0–1 are used for the general purpose register file; Page 1 addresses 00h–15h are also used for the LCD Data Register.

The following example indicates how to use the Page Pointer to clear the contents of RAM (i.e., Page 0, Page 1).

```
LD    PP,#00h   ; Destination ← 0, Source ← 0
SRP   #0C0h
```

```
            LD    R0,#0FFh  ; Page 0 RAM clear starts
RAMCL0      CLR   @R0
            DJNZ  R0,RAMCL0
            CLR   @R0       ; R0 = 00h LD PP,#10h ; Destination ←
                            ; 1, Source ← 0
            LD    R0,#0FFh  ; Page 1 RAM clear starts
RAMCL1      CLR   @R0
            DJNZ  R0,RAMCL1
            CLR   @R0       ; R0 = 00h
```

Note: To learn more about the DJNZ instruction, see page 125.

2.5. Register Set1

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

The upper 32-byte area of this 64-byte space (E0h–FFh) is divided into two 32-byte register banks, Bank0 and Bank1. The Set Register's SB0 or SB1 bank instructions are used to address one bank or the other. In the S3F8S45 microcontroller, Bank0 and Bank1 are implemented. A hardware reset operation always selects Bank0 addressing.

The upper two 32-byte areas of Set1 (E0h–FFh), Bank0 and Bank1, contain 68 mapped system and peripheral control registers. The lower 32-byte area contains 16 system registers (D0h–DFh) and a 16-byte common working register area (C0h–CFh). Use the common working register area as a *scratch* area for data operations being performed in other areas of the register file.

Registers in the Set1 location are directly accessible at all times using Register Addressing Mode. The 16-byte working register area can only be accessed using working register addressing. To learn more about working register addressing, see the Addressing Modes chapter on page 44.

2.6. Register Set2

The same 64-byte physical space that is used for Set1 location C0h–FFh is logically duplicated to add another 64 bytes of register space. This expanded area of the register file is called *Set2*. The Set2 locations (C0h–FFh) are accessible on pages 0–3 in the S3F8S45 MCU's register space.

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

2.7. Prime Register Space

The lower 192 bytes of the 256-byte physical internal register file (00h–BFh) are called the *prime register space* or, more simply, the *prime area*. Prime registers can be accessed using any of the seven addressing modes (to learn more, see the Addressing Modes chapter on page 44).

The prime register area on Page 0 is immediately addressable following a reset. To address the prime registers on pages 0 or 1, you must set the register page pointer (PP) to the appropriate source and destination values.

Figure 19 shows a map of the Set1, Set2, LCD Data, and prime register space.

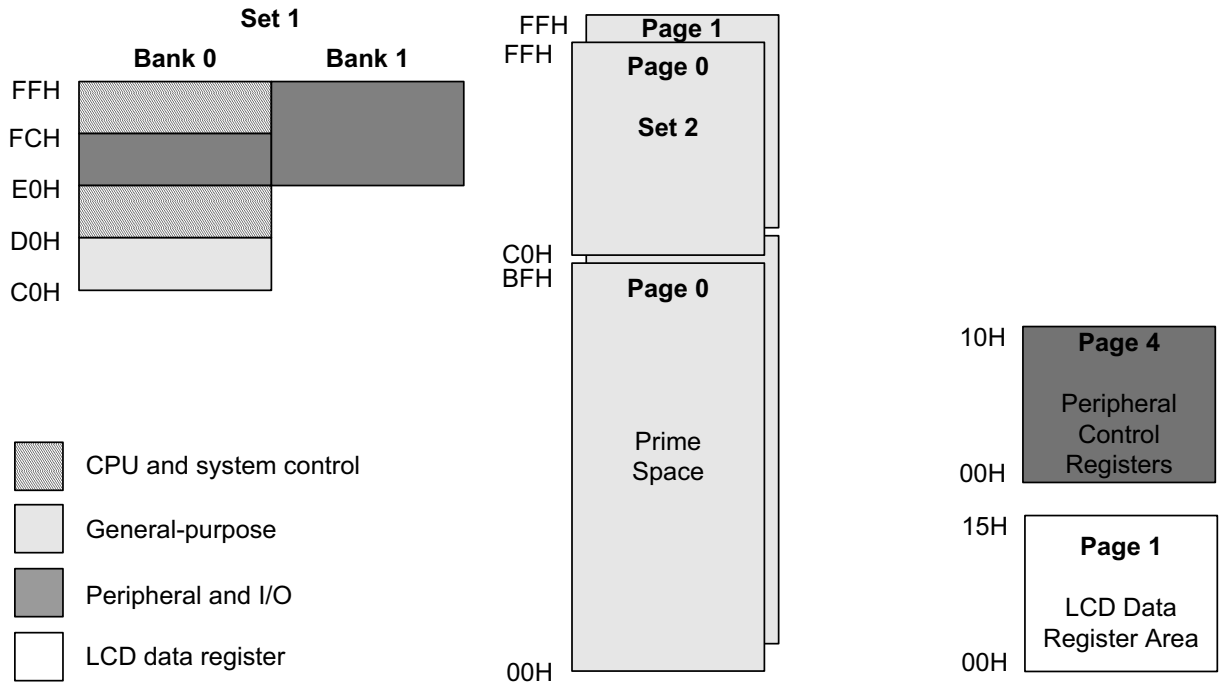


Figure 19. Set1, Set2, Prime Area Register and LCD Data Register Map

2.8. Working Registers

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

When using the two 8-bit register pointers, RP1 and RP0, two working register slices can be selected at any time to form a 16-byte working register block. When using these register pointers, you can move this 16-byte register block anywhere in the addressable register file, except for the Set2 area.

The terms *slice* and *block* are used in this document to help readers visualize the size and relative locations of selected working register spaces, as follows:

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

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

After a reset, RP0 and RP1 always point to the 16-byte common area in Set1 (C0h–CFh). Figure 20 illustrates the 8-byte working register areas (i.e., slices).

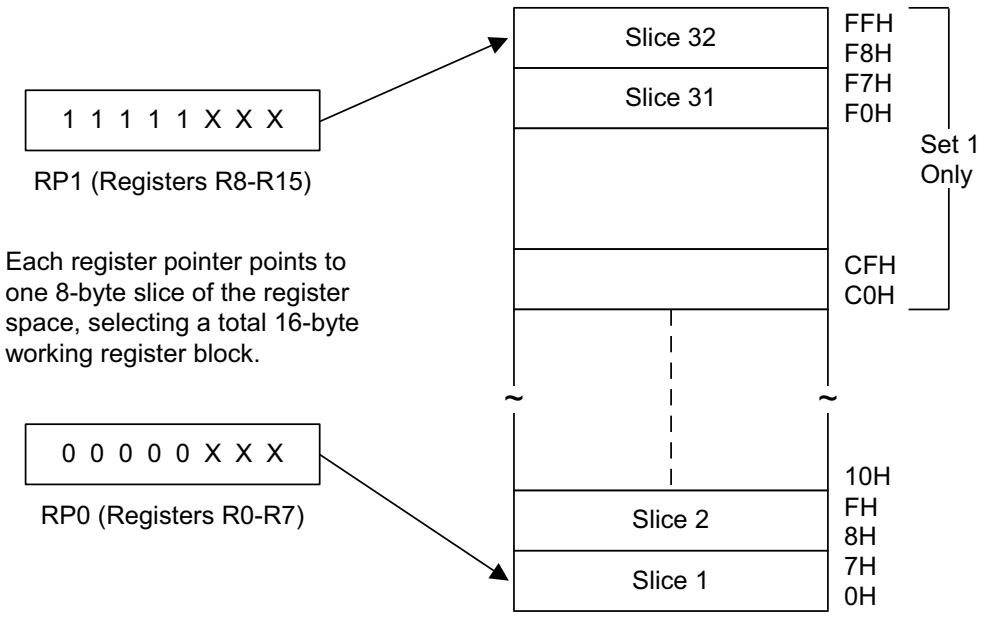


Figure 20. 8-Byte Working Register Areas

2.9. Using the Register Pointers

Register pointers RP0 and RP1, mapped to addresses D6h and D7h in Set1, are used to select two movable 8-byte working register slices in the register file. After a reset, they

point to the working register common area; RP0 points to addresses C0h–C7h, and RP1 points to addresses C8h–CFh.

To change a register pointer value, load a new value to RP0 and/or RP1 using an SRP or LD instruction; see Figures 21 and 22.

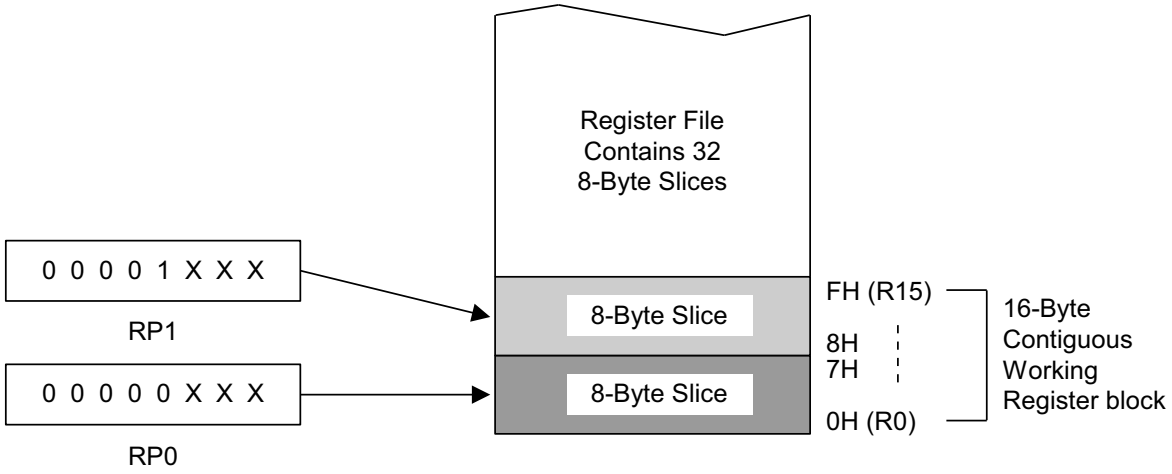


Figure 21. Contiguous 16-Byte Working Register Block

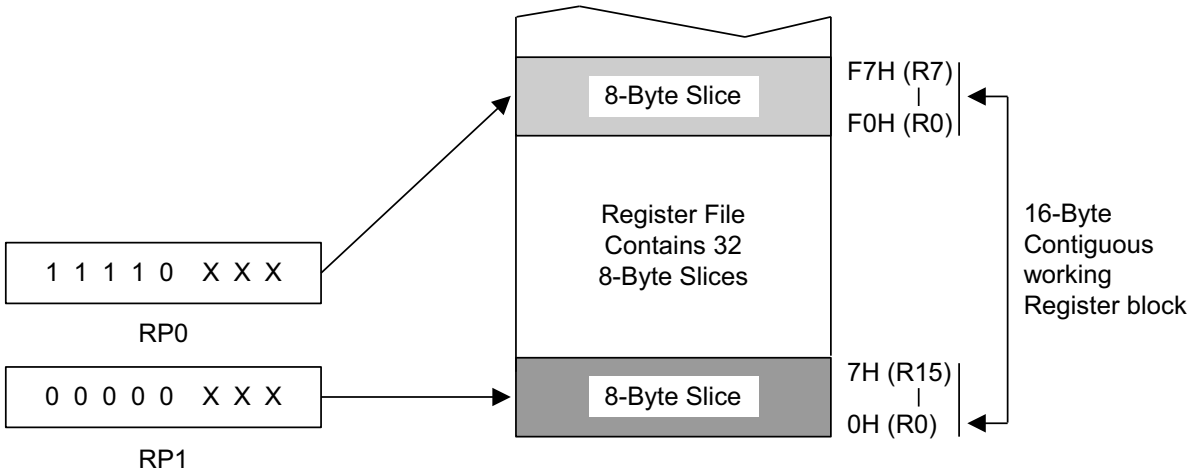


Figure 22. Noncontiguous 16-Byte Working Register Block

With working register addressing, you can only access those two 8-bit slices of the register file that are currently pointed to by RP0 and RP1. You cannot, however, use the register

pointers to select a working register space in Set2, C0h to FFh, because these locations can be accessed only using the Indirect Register or Indexed Addressing modes.

The selected 16-byte working register block usually consists of two contiguous 8-byte slices. As a general programming guideline, Zilog recommends that RP0 point to the lower slice and RP1 point to the upper slice; see Figure 20 on page 30. In some cases, it may be necessary to define working register areas in different (noncontiguous) areas of the register file. In Figure 21, RP0 points to the *upper* slice and RP1 to the *lower* slice.

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

Setting the Register Pointers

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

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

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

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

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

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

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

2.10. Register Pointer Registers

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

Table 4. Register Pointer 0 (RP0; Set1)

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

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

Bit	Description
[7:3]	**Register Pointer 0 Address Value** Register pointer 0 can independently point to one of the 256-byte working register areas in the register file. Using register pointers RP0 and RP1, select two 8-byte register slices at one time as the active working register space. After a reset, RP0 points to C0h in register Set1, selecting the 8-byte working register slice C0h–C7h.
[2:0]	**Reserved**

The contents of the Register Pointer 1 (RP1) Register are described in Table 5.

Table 5. Register Pointer 1 (RP1; Set1, Bank0)

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

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

Bit	Description
[7:3]	**Register Pointer 1 Address Value** Register pointer 1 can independently point to one of the 256-byte working register areas in the register file. Using register pointers RP0 and RP1, select two 8-byte register slices at one time as the active working register space. After a reset, RP1 points to C8h in register Set1, selecting the 8-byte working register slice C8h–CFh.
[2:0]	**Reserved**

2.11. Register Addressing

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

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

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

Working register addressing differs from register addressing because it uses a register pointer to identify a specific 8-byte working register space in the internal register file, and a specific 8-bit register within that space; see Figures 23 and 24.

Figure 23. 16-Bit Register Pair

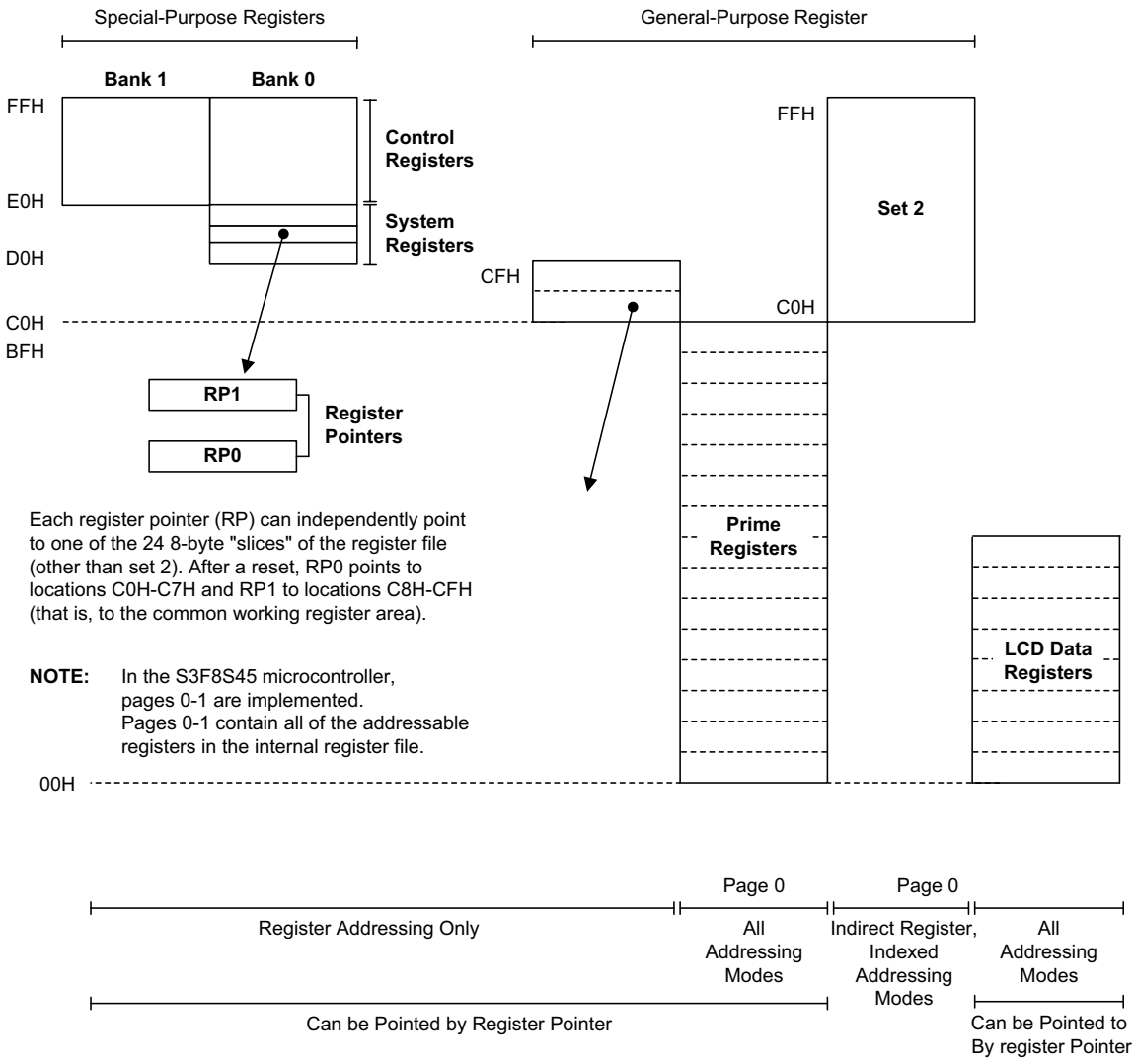


Figure 24. Register File Addressing

2.12. Common Working Register Area

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

- RP0 → `C0h–C7h`
- RP1 → `C8h–CFh`

This 16-byte address range is called the *common working area*. Essentially, locations in this area can be used as working registers by operations that address any location on any page in the register file. Typically, these working registers serve as temporary buffers for data operations between different pages. See Figure 25.

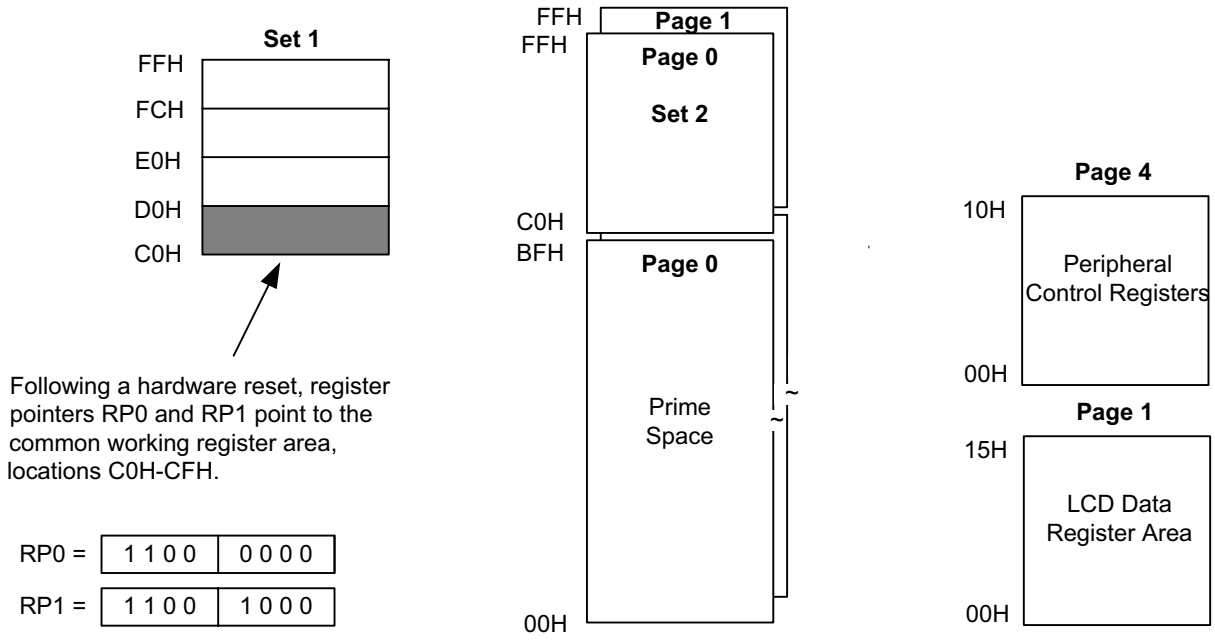


Figure 25. Common Working Register Area

Addressing the Common Working Register Area

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

Example 1

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

Use working register addressing instead:

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

Example 2

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

Use working register addressing instead:

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

2.13. 4-Bit Working Register Addressing

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

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

As shown in Figure 26, the result of this operation is that the five high-order bits from the register pointer are concatenated with the three low-order bits from the instruction address to form the complete address. As long as the address stored in the register pointer remains unchanged, the three bits from the address will always point to an address in the same 8-byte register slice.

Figure 27 shows a typical example of 4-bit working register addressing. The high-order bit of the INC-R6 instruction is 0, which selects RP0. The five high-order bits stored in RP0 (`01110b`) are concatenated with the three low-order bits of the instruction's 4-bit address (`110b`) to produce the register address `76h` (`01110110b`).

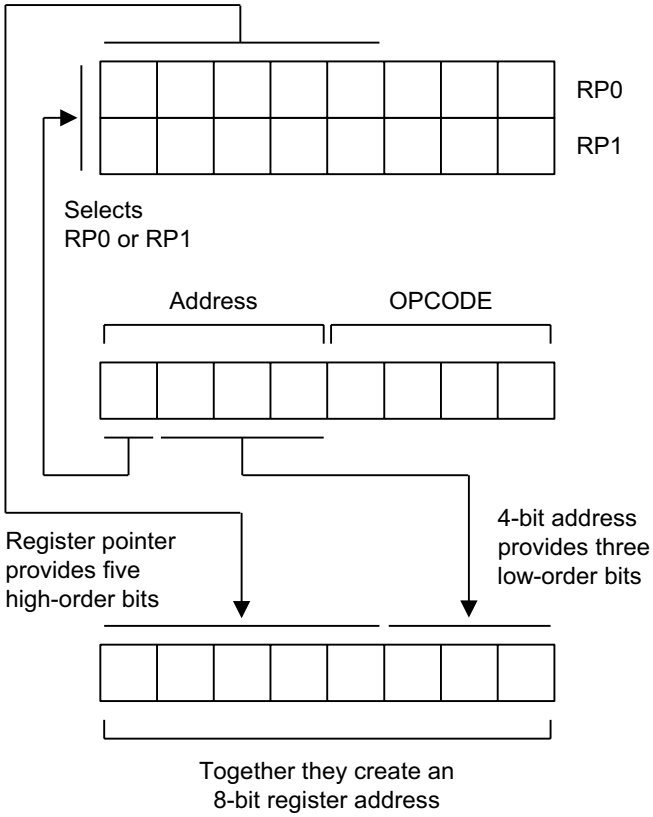


Figure 26. 4-Bit Working Register Addressing

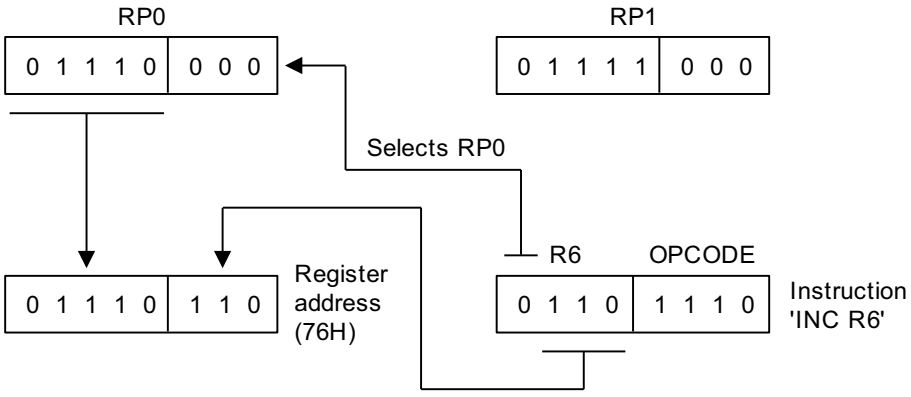


Figure 27. 4-Bit Working Register Addressing Example

2.14. 8-Bit Working Register Addressing

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

As shown in Figure 28, the lower nibble of the 8-bit address is concatenated in much the same way as for 4-bit addressing: bit 3 selects either RP0 or RP1, which then supplies the five high-order bits of the final address. The three low-order bits of the complete address are provided by the original instruction.

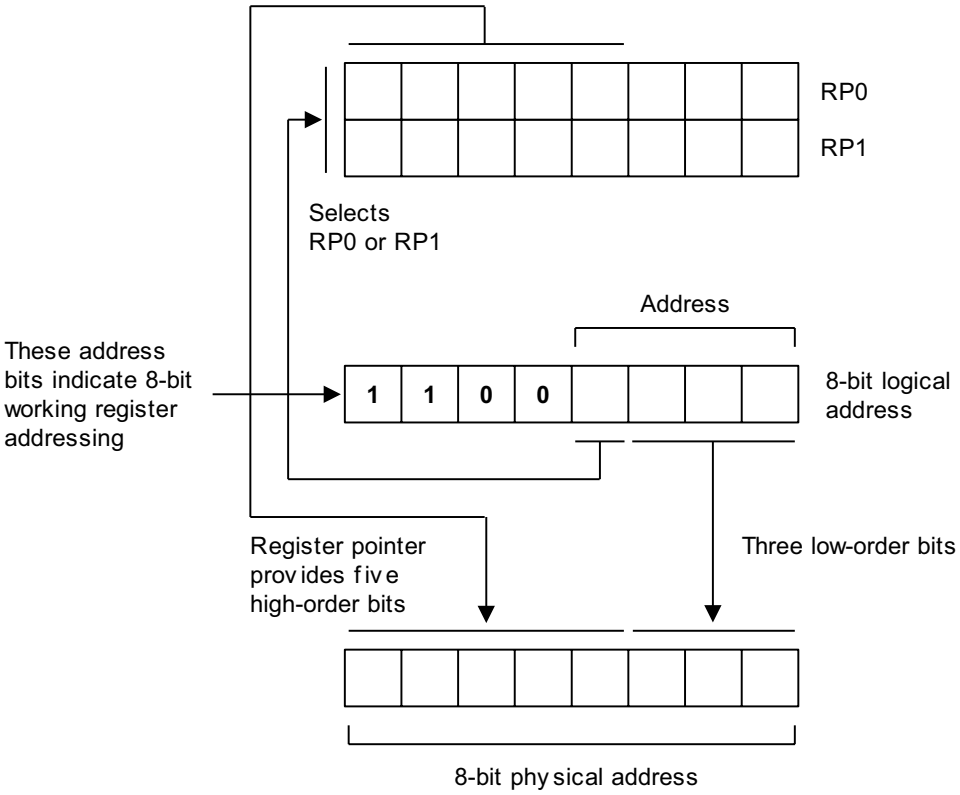


Figure 28. 8-Bit Working Register Addressing

Figure 29 shows an example of 8-bit working register addressing. The four high-order bits of the instruction address (1100b) specify 8-bit working register addressing. Bit 4 (1) selects RP1, and the five high-order bits in RP1 (10101b) become the five high-order bits of the register address. The three low-order bits of the register address (011) are provided by the three low-order bits of the 8-bit instruction address. The five-address bits from RP1

and the three address bits from the instruction are concatenated to form the complete register address, 0ABh (10101011b).

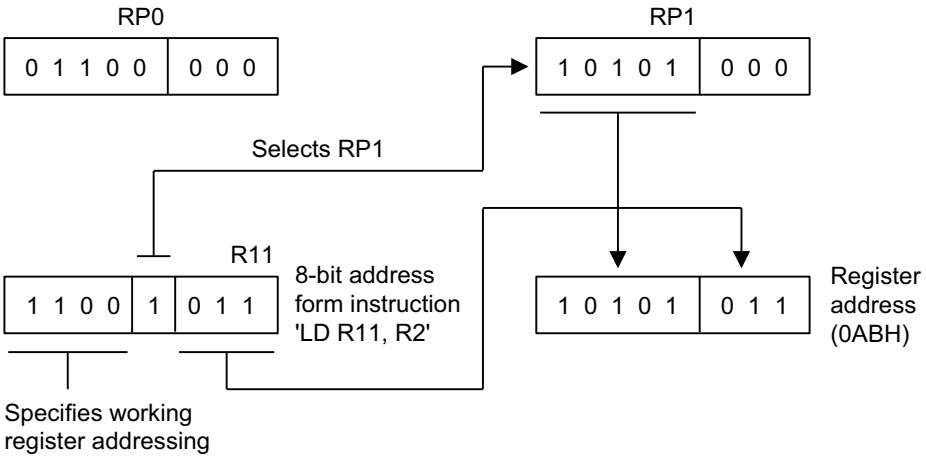


Figure 29. 8-Bit Working Register Addressing Example

2.15. System and User Stack

S3F8 Series microcontrollers use the system stack for subroutine calls and returns and to store data. The PUSH and POP instructions are used to control system stack operations. The S3F8S45 architecture supports stack operations in the internal register file.

2.16. Stack Operations

Return addresses for procedure calls, interrupts, and data are stored on the stack. The contents of the PC are saved to stack by a CALL instruction and restored by the RET instruction. When an interrupt occurs, the contents of the PC and the Flags registers are pushed to the stack. The IRET instruction then pops these values back to their original locations. The stack address value is always decreased by one before a push operation and increased by one after a pop operation. The stack pointer (SP) always points to the stack frame stored on the top of the stack, as shown in Figure 30.

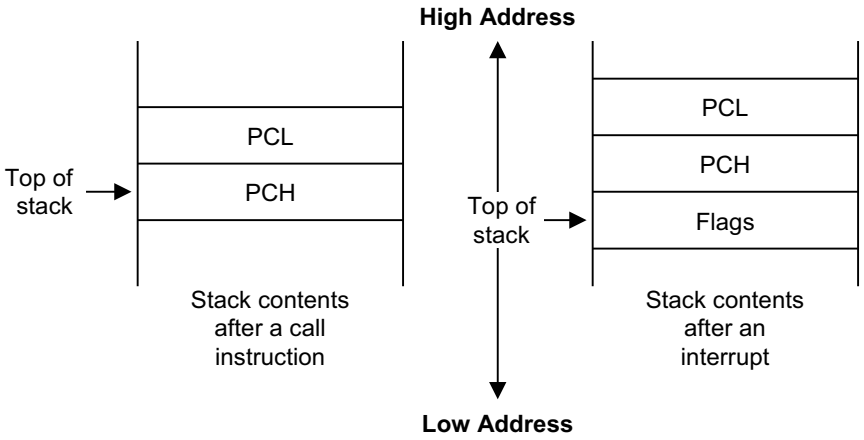


Figure 30. Stack Operations

2.17. User-Defined Stacks

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

2.18. Stack Pointers

Register locations D8h and D9h contain the 8-bit stack pointer (SPL) that is used for system stack operations. The most significant byte of the SP address, SP15–SP8, is stored in the SPH Register (D8h), and the least significant byte, SP7–SP0, is stored in the SPL Register (D9h). After a reset, the SP value is undetermined.

Because only internal memory space is implemented in the S3F8S45 MCU, the SPL must be initialized to an 8-bit value in the range 00h–FFh. The SPH Register is not required and can be used as a general-purpose register, if necessary.

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

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

Standard Stack Operations Using PUSH and POP

```
LD      SPL, #0FFh  ; SPL ← FFh; (Normally, the SPL is set to
                                ; 0FFh by the initialization
                                ; routine)
•
•
•
PUSH    PP          ; Stack address 0FEh ← PP
PUSH    RP0         ; Stack address 0FDh ← RP0
PUSH    RP1         ; Stack address 0FCh ← RP1
PUSH    R3          ; Stack address 0FBh ← R3
•
•
•
POP     R3          ; R3 ← Stack address 0FBh
POP     RP1         ; RP1 ← Stack address 0FCh
POP     RP0         ; RP0 ← Stack address 0FDh
POP     PP          ; PP ← Stack address 0FEh
```

The contents of the Stack Pointer High Byte (SPH) and Stack Pointer Low Byte (SPL) registers are described in Tables 6 and 7.

Table 6. Stack Pointer High Byte (SPH; Set1)

Bit	7	6	5	4	3	2	1	0
Reset	x	x	x	x	x	x	x	x
R/W	R/W							
Address	D8h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:0]	**Stack Pointer Address High Byte** The high-byte stack pointer value is the upper eight bits of the 16-bit stack pointer address (SP15–SP8). The lower byte of the stack pointer value is located in register SPL (D9H). The SP value is undefined following a reset.

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

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

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

Bit	Description
[7:0]	**Stack Pointer Address Low Byte** The low-byte stack pointer value is the lower eight bits of the 16-bit stack pointer address (SP7–SP0). The upper byte of the stack pointer value is located in register SPH (D8h). The SP value is undefined following a reset.

Chapter 3. Addressing Modes

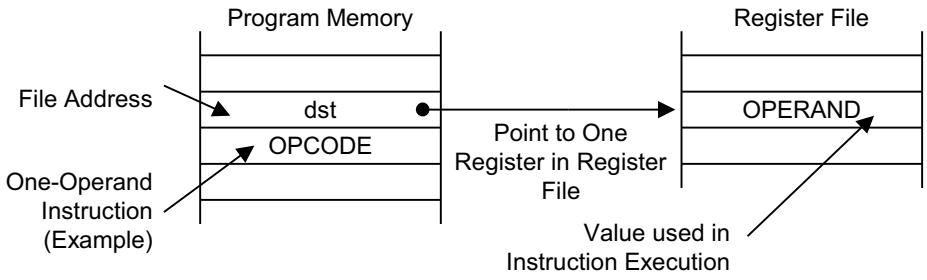
The program counter is used to fetch instructions that are stored in program memory for execution. Instructions indicate the operation to be performed and the data to be operated on. Addressing Mode is the method used to determine the location of the data operand. The operands specified in instructions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

The S3F8 Series instruction set supports the following seven explicit addressing modes; not all of these addressing modes are available for each instruction.

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

3.1. Register Addressing Mode

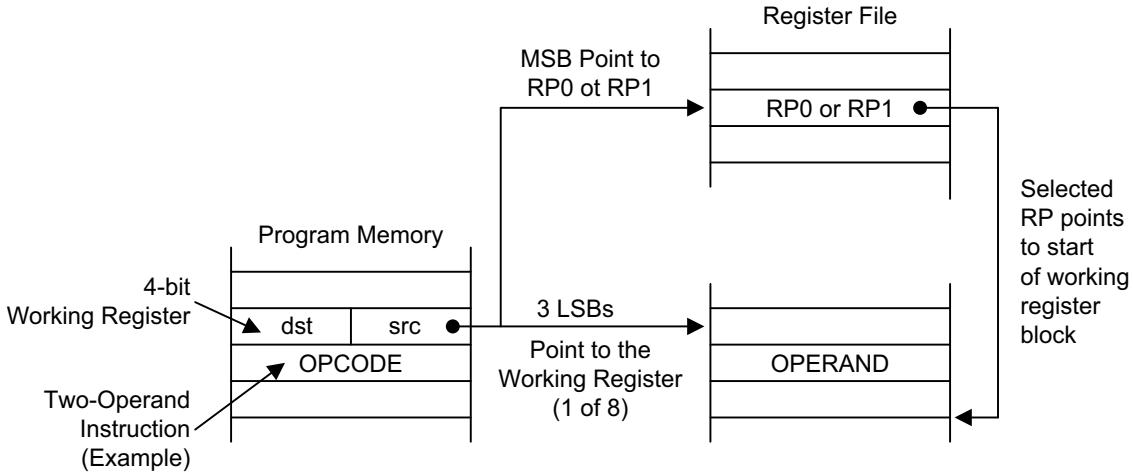
In Register Addressing Mode (R), the operand is the content of a specified register or register pair; see Figure 31. Working register addressing differs from register addressing because it uses a register pointer to specify an 8-byte working register space in the register file and an 8-bit register within that space; see Figure 32.



Sample Instruction:

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

Figure 31. Register Addressing



Sample Instruction:

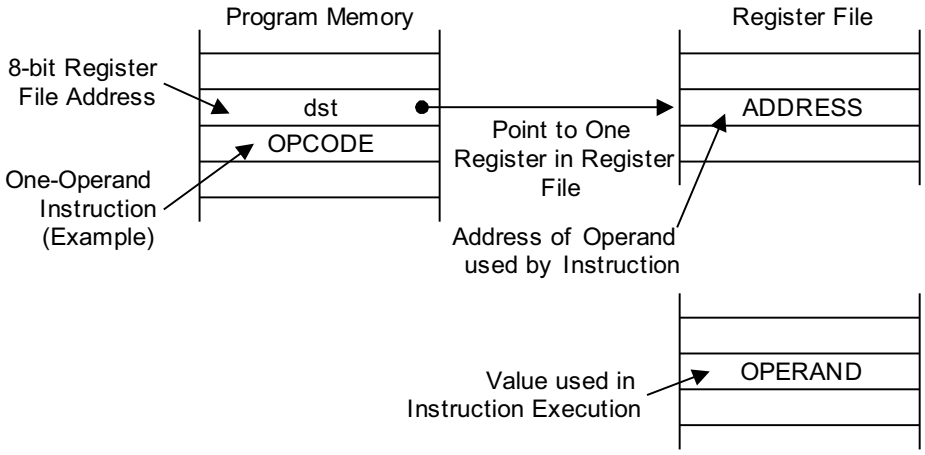
```
ADD     R1, R2          ;   Where R1 and R2 are registers in the currently
                            selected working register area.
```

Figure 32. Working Register Addressing

3.2. Indirect Register Addressing Mode

In Indirect Register (IR) Addressing Mode, the contents of the specified register or register pair represent the address of the operand. Depending on the instruction used, the actual address may point to a register in the register file, to program memory (ROM), or to an external memory space, if implemented; see Figures 33 through 36.

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



Sample Instruction:

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

Figure 33. Indirect Register Addressing to Register File

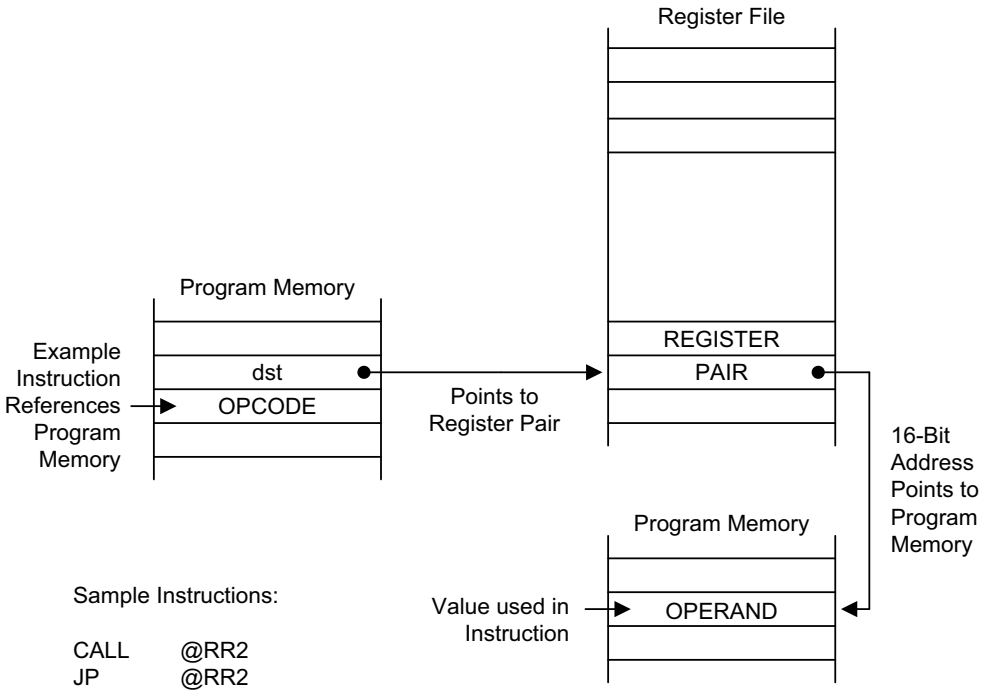


Figure 34. Indirect Register Addressing to Program Memory

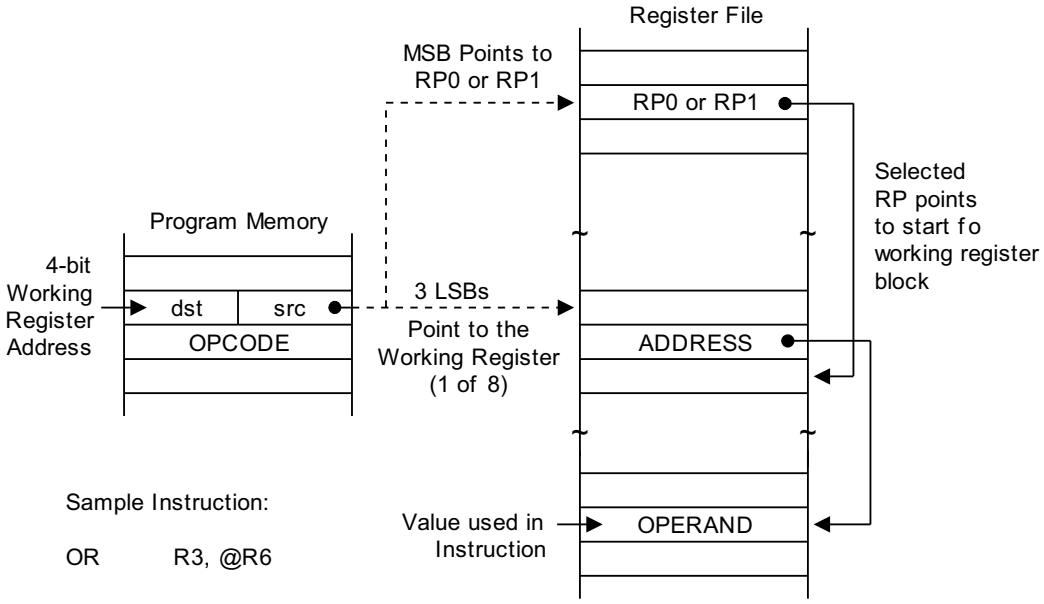
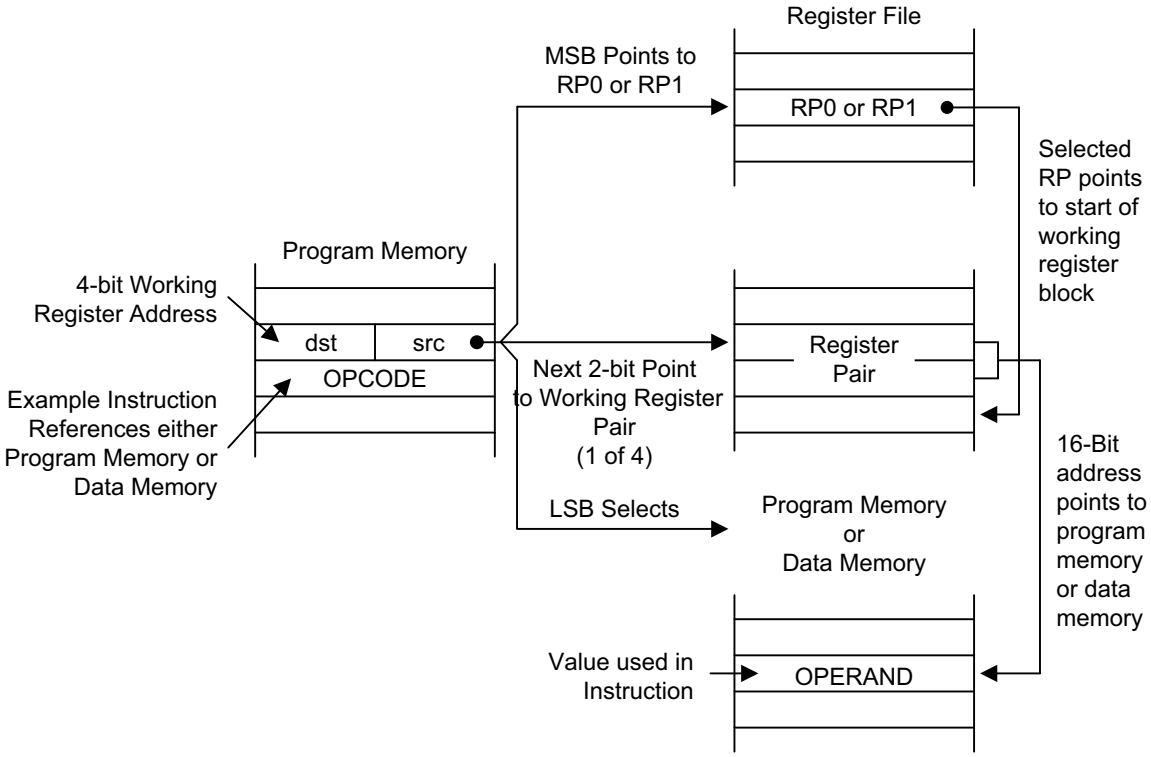


Figure 35. Indirect Working Register Addressing to Register File



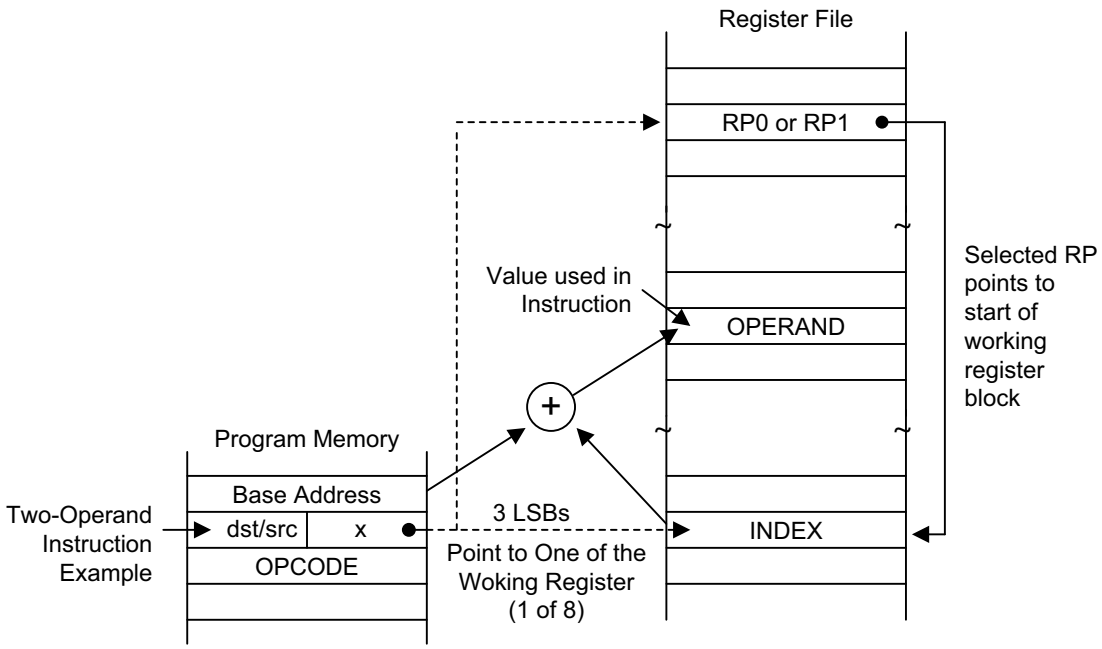
Sample Instructions:

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

Figure 36. Indirect Working Register Addressing to Program or Data Memory

3.3. Indexed Addressing Mode

Indexed (X) Addressing Mode adds an offset value to a base address during instruction execution to calculate the effective operand address; see Figure 37. Use Indexed Addressing Mode to access locations in the internal register file or in external memory. You cannot, however, access locations C0h–FFh in Set1 using indexed addressing.



Sample Instruction:

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

Figure 37. Indexed Addressing to Register File

In short offset Indexed Addressing Mode, the 8-bit displacement is treated as a signed integer in the range –128 to +127. This displacement applies to external memory accesses only; see Figure 38.

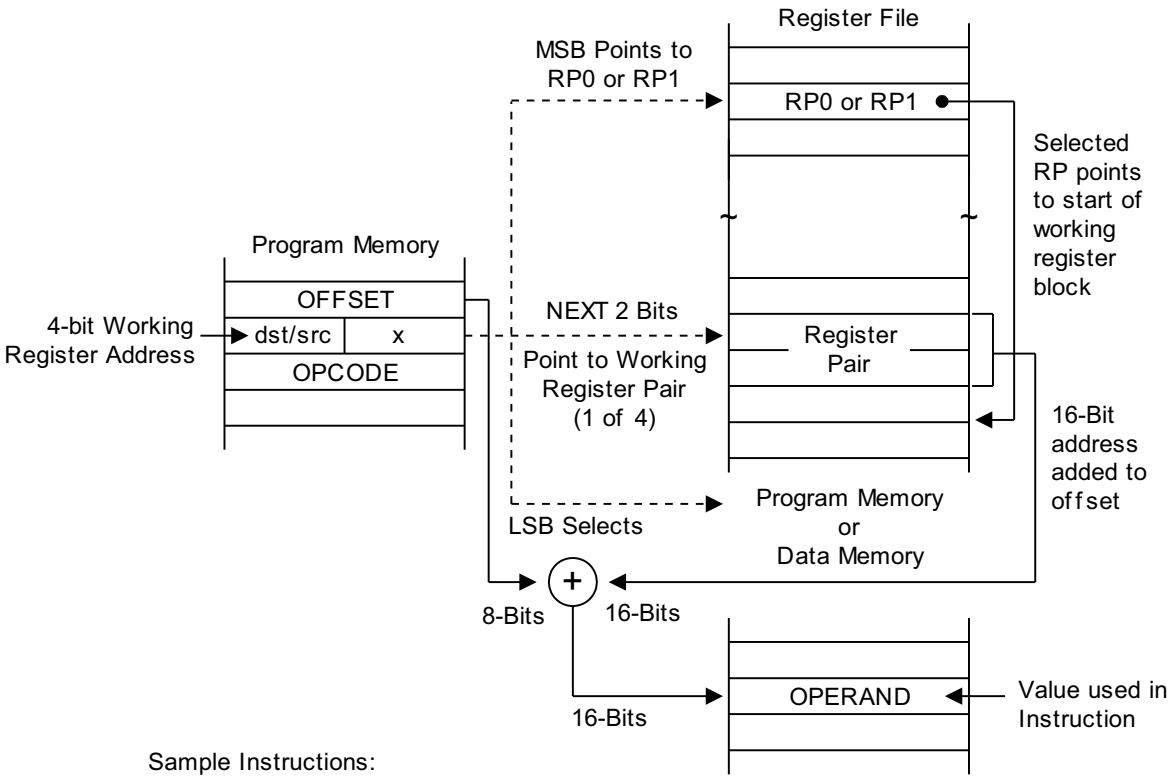
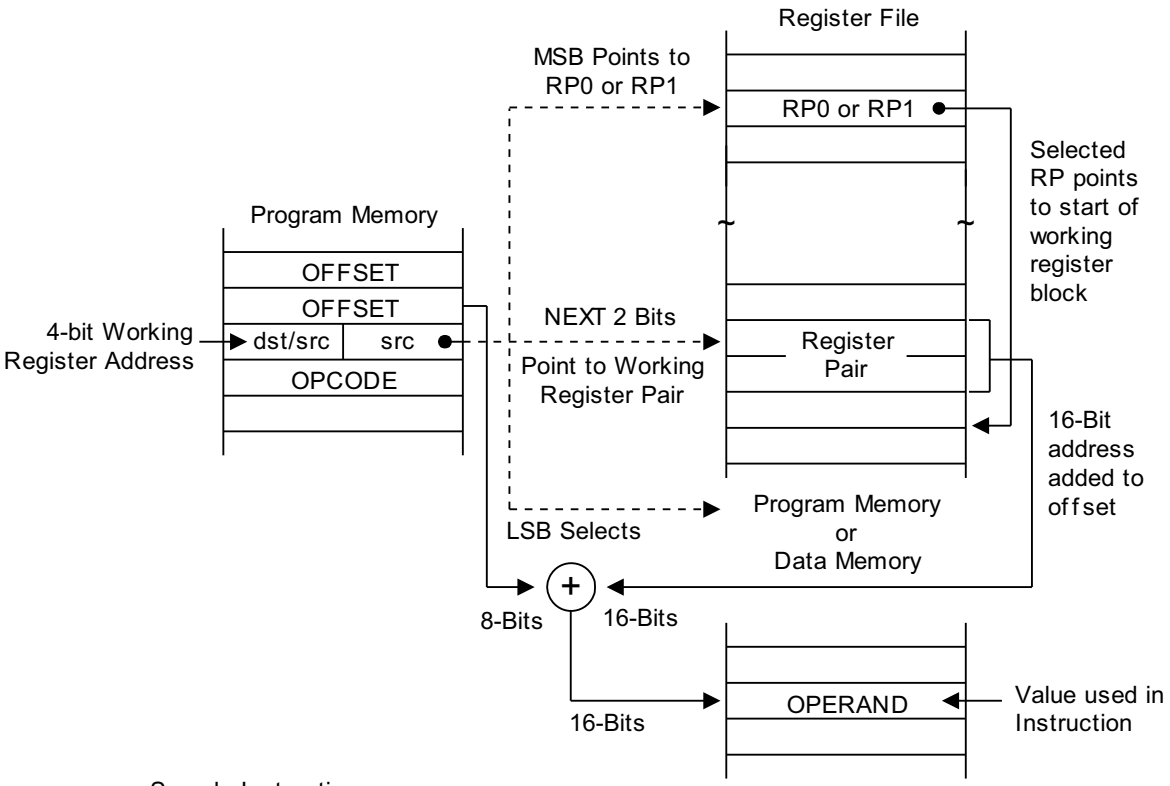


Figure 38. Indexed Addressing to Program or Data Memory with Short Offset

For register file addressing, an 8-bit base address provided by the instruction is added to an 8-bit offset contained in a working register. For external memory accesses, the base address is stored in the working register pair designated in the instruction. The 8-bit or 16-bit offset provided in the instruction is then added to the base address; see Figure 39.



Sample Instructions:

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

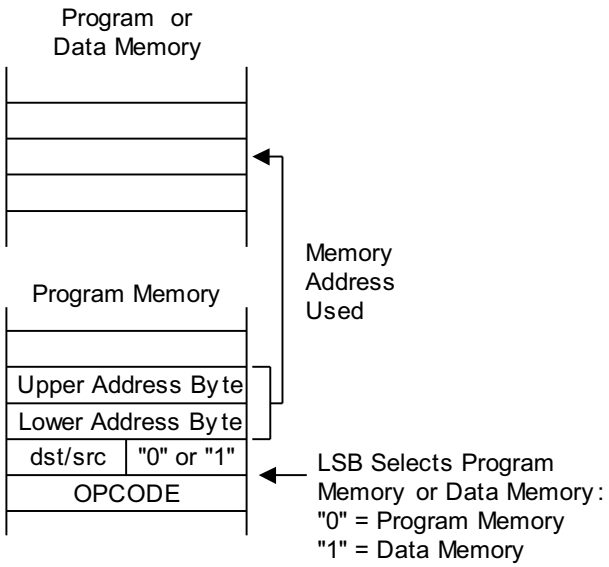
Figure 39. Indexed Addressing to Program or Data Memory

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

3.4. Direct Address Mode

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

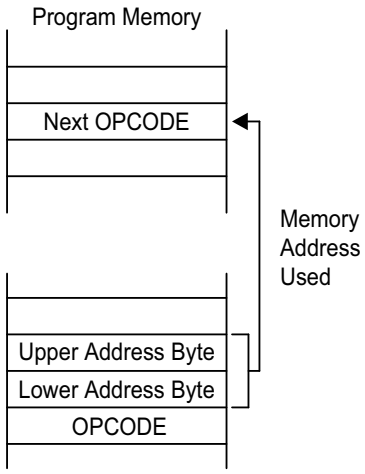
The LDC and LDE instructions can use Direct Address Mode to specify the source or destination address for Load operations to program memory (LDC) or to external data memory (LDE), if implemented. See Figures 40 and 41.



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 40. Direct Addressing for Load Instructions



Sample Instructions:

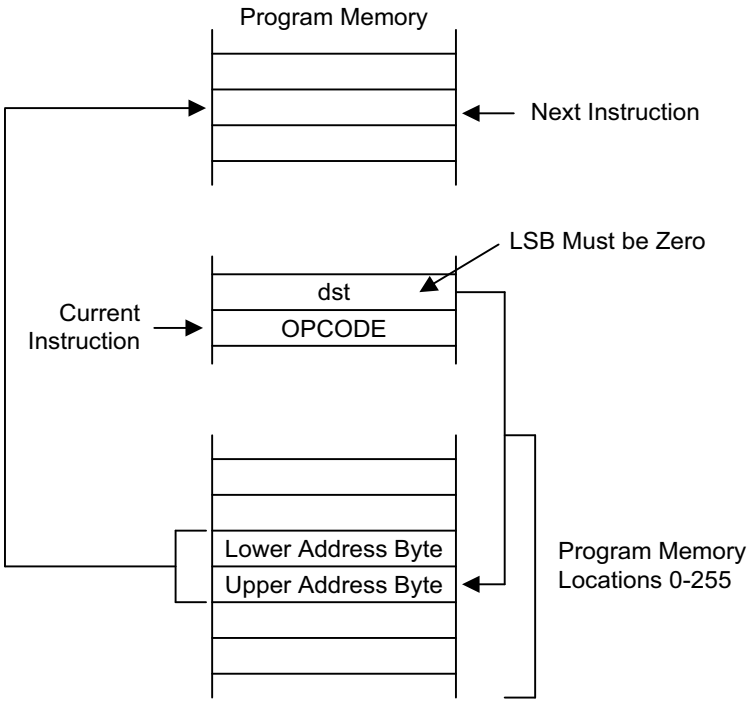
```
JP      C,JOB1          ;   Where JOB1 is a 16-bit immediate address
CALL    DISPLAY ;   Where DISPLAY is a 16-bit immediate address
```

Figure 41. Direct Addressing for Call and Jump Instructions

3.5. Indirect Address Mode

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

Because Indirect Address Mode assumes that the operand is located in the lowest 256 bytes of program memory, only an 8-bit address is supplied in the instruction; the upper bytes of the destination address are assumed to be all zeros. See Figure 42.



Sample Instruction:

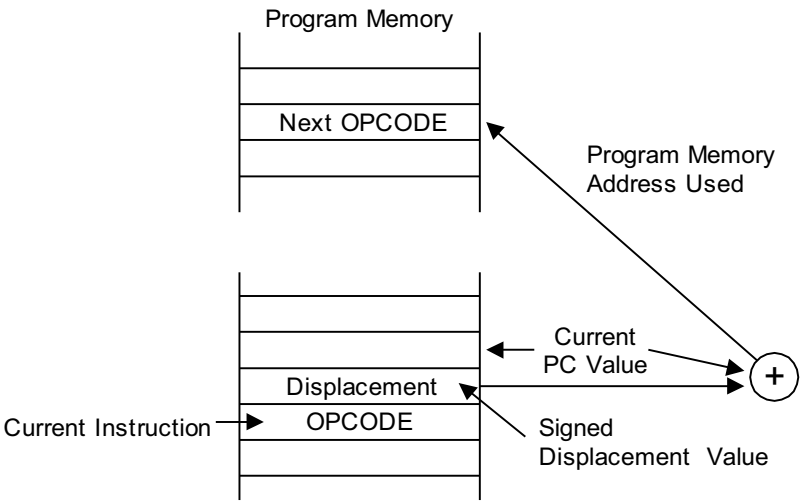
```
CALL    #40H        ; The 16-bit value in program memory addresses 40H
                      and 41H is the subroutine start address.
```

Figure 42. Indirect Addressing

3.6. Relative Address Mode

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

Several program control instructions use Relative Address Mode to perform conditional jumps. The instructions that support RA addressing are BTJRF, BTJRT, DJNZ, CPIJE, CPIJNE, and JR. See Figure 43.



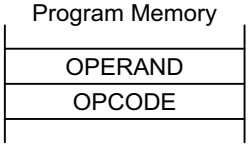
Sample Instructions:

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

Figure 43. Relative Addressing

3.7. Immediate Mode

In Immediate (IM) Addressing Mode, the operand value used in the instruction is the value supplied in the operand field itself. The operand may be one byte or one word in length, depending on the instruction used. Immediate Addressing Mode is useful for loading constant values into registers. See Figure 44.



(The Operand value is in the instruction)

Sample Instruction:

LD R0,#0AAH

Figure 44. Immediate Addressing

Chapter 4. Control Registers

This chapter describes the S3F8S45 MCU's control registers. Data and counter registers are not described in this chapter; information about all registers used by a specific peripheral is presented in corresponding chapters.

Table 8 identifies the names, mnemonics, decimal and hex equivalents, and read/write settings of the Set1, Bank0 mapped registers; click to the linked page to review the contents of each. The hardware reset value for each mapped register is described in the Reset and Power-Down chapter on page 194.

Table 8. Set1 Registers

Register Name	Mnemonic	Page #	Decimal	Hex	R/W
Basic Timer Control	BTCON	229	211	D3h	R/W
System Clock Control	CLKCON	190	212	D4h	R/W
Flags	FLAGS	86	213	D5h	R/W
Register Pointer 0	RP0	33	214	D6h	R/W
Register Pointer 1	RP1	33	215	D7h	R/W
Stack Pointer High Byte	SPH	42	216	D8h	R/W
Stack Pointer Low Byte	SPL	43	217	D9h	R/W
Instruction Pointer High Byte	IPH	79	218	DAh	R/W
Instruction Pointer Low Byte	IPL	80	219	DBh	R/W
Interrupt Request	IRQ	74	220	DCh	R
Interrupt Mask	IMR	76	221	DDh	R/W
System Mode	SYM	69	222	DEh	R/W
Register Page Pointer	PP	27	223	DFh	R/W

Table 9 identifies the names, mnemonics, decimal and hex equivalents, and read/write settings of the Page 4 registers.

Table 9. Page 4 Registers

Register Name	Mnemonic	Page #	Decimal	Hex	R/W
Reset Source Indicating	RESETID	199	0	00h	R/W
Timer D0 Control	TD0CON	250	1	01h	R/W
Timer D0 Counter High Byte	TD0CNTH	–	2	02h	R
Timer D0 Counter Low Byte	TD0CNTL	–	3	03h	R
Timer D0 Data High Byte	TD0DATAH	–	4	04h	R/W

Table 9. Page 4 Registers (Continued)

Register Name	Mnemonic	Page #	Decimal	Hex	R/W
Timer D0 Data Low Byte	TD0DATAL	–	5	05h	R/W
Timer D1 Control	TD1CON	257	10	0Ah	R/W
Timer D1 Counter High Byte	TD1CNTH	–	6	06h	R
Timer D1 Counter Low Byte	TD1CNTL	–	7	07h	R
Timer D1 Data High Byte	TD1DATAH	–	8	08h	R/W
Timer D1 Data Low Byte	TD1DATAL	–	9	09h	R/W

Table 10. Set1, Bank0 Registers

Register Name	Mnemonic	Page #	Decimal	Hex	R/W
A/D Converter Data High Byte	ADDATAH	281	208	D0h	R
A/D Converter Data Low Byte	ADDATAL	281	209	D1h	R
A/D Converter Control	ADCON	280	210	D2h	R/W
Timer A Counter	TACNT	–	224	E0h	R
Timer A Data	TADATA	–	225	E1h	R/W
Timer A Control	TACON	234	226	E2h	R/W
Timer B Control	TBCON	240	227	E3h	R/W
Timer B Data High Byte	TBDATAH	–	228	E4h	R/W
Timer B Data Low Byte	TBDATAL	–	229	E5h	R/W
Watch Timer Control	WTCON	264	230	E6h	R/W
SIO Control	SIOCON	285	231	E7h	R/W
SIO Data	SIODATA	–	232	E8h	R/W
SIO Prescaler	SIOPS	286	233	E9h	R/W
Timer C Counter	TCCNT	–	234	EAh	R
Timer C Data	TCDATA	–	235	EBh	R/W
Timer C Control	TCCON	246	236	ECh	R/W
STOP Control	STPCON	192	237	EDh	R/W
UART0 Control High Byte	UART0CONH	290	238	EEh	R/W
UART0 Control Low Byte	UART0CONL	292	239	EFh	R/W
UART0 Data	UDATA0	293	240	F0h	R/W
UART0 Baud Rate Data	BRDATA0	293	241	F1h	R/W
UART1 Control High Byte	UART1CONH	305	242	F2h	R/W
UART1 Control Low Byte	UART1CONL	307	243	F3h	R/W

Table 10. Set1, Bank0 Registers (Continued)

Register Name	Mnemonic	Page #	Decimal	Hex	R/W
UART1 Data	UDATA1	308	244	F4h	R/W
UART1 Baud Rate Data	BRDATA1	308	245	F5h	R/W
Flash Memory sector Address High Byte	FMSECH	330	246	F6h	R/W
Flash Memory sector Address Low Byte	FMSECL	331	247	F7h	R/W
Flash Memory User Programming Enable	FMUSR	329	248	F8h	R/W
Flash Memory Control	FMCON	328	249	F9h	R/W
Oscillator Control	OSCCON	191	250	FAh	R/W
Interrupt Pending	INTPND	76	251	FBh	R/W
Location FCh is not mapped.					
Basic Timer Counter	BTCNT	–	253	FDh	R
Location FEh is not mapped.					
Interrupt Priority Register	IPR	73	255	FFh	R/W

Table 11. Set1, Bank1 Registers

Register Name	Mnemonic	Page #	Decimal	Hex	R/W
Port 0 Control High Byte	P0CONH	205	208	D0h	R/W
Port 0 Control Low Byte	P0CONL	206	209	D1h	R/W
Port 0 Pull-up Resistor Enable	P0PUR	207	210	D2h	R/W
Port 1 Control High Byte	P1CONH	208	224	E0h	R/W
Port 1 Control Low Byte	P1CONL	209	225	E1h	R/W
Port 1 Pull-up Resistor Enable	P1PUR	210	226	E2h	R/W
Port 1 n-Channel Open-drain Mode	PNE1	211	227	E3h	R/W
Port 2 Control High Byte	P2CONH	212	228	E4h	R/W
Port 2 Control Low Byte	P2CONL	213	229	E5h	R/W
Port 2 Pull-up Resistor Enable	P2PUR	214	230	E6h	R/W
Port 0/3 Interrupt Pending	P03PND	217	231	E7h	R/W
Port 0/3 Interrupt Control High Byte	P03INTH	220	232	E8h	R/W
Port 0/3 Interrupt Control Low Byte	P03INTL	221	233	E9h	R/W
Port 3 Control High Byte	P3CONH	215	234	EAh	R/W
Port 3 Control Low Byte	P3CONL	216	235	EBh	R/W

Table 11. Set1, Bank1 Registers (Continued)

Register Name	Mnemonic	Page #	Decimal	Hex	R/W
Port 3 Pull-up Resistor Enable	P3PUR	218	236	ECh	R/W
Port 3 n-Channel Open-drain Mode	PNE3	219	237	EDh	R/W
Port 4 Control High Byte	P4CONH	222	238	EEh	R/W
Port 4 Control Low Byte	P4CONL	224	239	EFh	R/W
Port 0 Data	P0	–	240	F0h	R/W
Port 1 Data	P1	–	241	F1h	R/W
Port 2 Data	P2	–	242	F2h	R/W
Port 3 Data	P3	–	243	F3h	R/W
Port 4 Data	P4	–	244	F4h	R/W
Port 4 Pull-up Resistor Enable	P4PUR	225	245	F5h	R/W
Port 4 n-Channel Open-drain Mode	PNE4	226	246	F6h	R/W
Pattern Generation Control	PGCON	321	247	F7h	R/W
Pattern Generation Data	PGDATA	–	248	F8h	R/W
LCD Control	LCON	268	249	F9h	R/W
LCD Mode Control	LMOD	269	250	FAh	R/W
Battery Level Detection Control	BLDCON	325	251	FBh	R/W
Watch-Dog Timer Control	WDTCON	230	252	FCh	R/W
Locations FDh–FFh are not mapped.					

Chapter 5. Interrupt Structure

The S3F8 Series interrupt structure has three basic components: levels, vectors, and sources. The SAM88 RC CPU recognizes up to eight interrupt levels and supports up to 128 interrupt vectors. When a specific interrupt level has more than one vector address, the vector priorities are established in hardware. A vector address can be assigned to one or more sources.

5.1. Levels

Interrupt levels are the main unit for interrupt priority assignment and recognition. All peripherals and I/O blocks can issue interrupt requests. In other words, peripheral and I/O operations are interrupt-driven. There are eight possible interrupt levels: IRQ0–IRQ7, also called Level 0–Level 7. Each interrupt level directly corresponds to an interrupt request number (IRQn). The total number of interrupt levels used in the interrupt structure varies from device to device. The S3F8S45 MCU's interrupt structure recognizes eight interrupt levels.

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

5.2. Vectors

Each interrupt level can have one or more interrupt vectors, or it may have no vector address assigned at all. The maximum number of vectors that can be supported for a given level is 128. (The actual number of vectors used for S3F8 Series devices is always much smaller.) If an interrupt level has more than one vector address, the vector priorities are set in hardware. The S3F8S45 MCU uses twenty-two vectors.

5.3. Sources

A source is any peripheral that generates an interrupt. For example, a source can be an external pin or a counter overflow. Each vector can have several interrupt sources. In the S3F8S45 MCU's interrupt structure, there are 22 possible interrupt sources.

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

5.4. Interrupt Types

The three components of the S3F8 Series interrupt structure described previously – levels, vectors, and sources – are combined to determine the interrupt structure of an individual device and to make full use of its available interrupt logic. There are three possible combinations of interrupt structure components, called interrupt types 1, 2, and 3. These three types differ in the number of vectors and interrupt sources assigned to each level, as follows.

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

In the S3F8S45 microcontroller, all three interrupt types are implemented; see Figure 45.

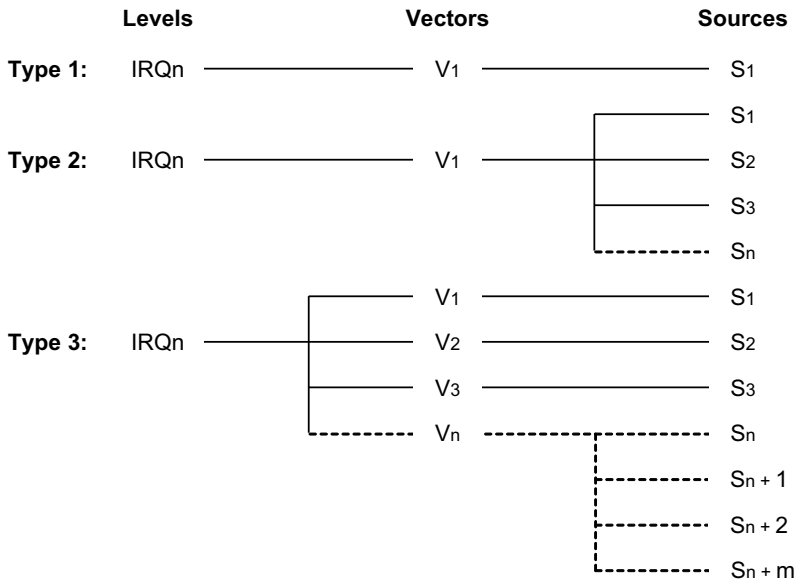


Figure 45. S3F8 Series Interrupt Types

Note: In Figure 45, the number of Sn and Vn values is expandable. In the S3F8S45 implementation, interrupt types 1 and 3 are used.

5.5. Interrupt Structure

The S3F8S45 microcontroller supports twenty-two interrupt sources, all of which have a corresponding interrupt vector address. Eight interrupt levels are recognized by the CPU in this device-specific interrupt structure, as shown in Figure 46.

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

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

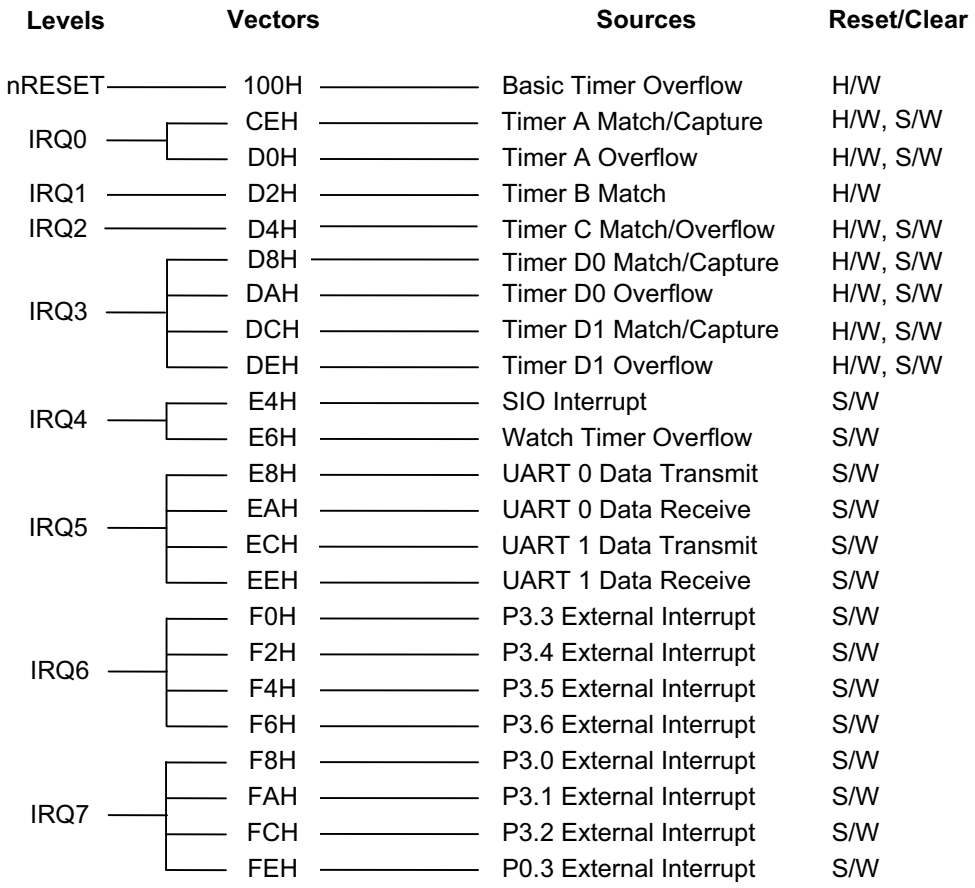


Figure 46. S3F8S45 Interrupt Structure

Notes: 1. In Figure 46, within a given interrupt level, the low vector address has high priority. For example, within the IRQ0 level, CEh has higher priority than D0h. Priorities within each level are set at the factory.

2. External interrupts are triggered by a rising or falling edge, depending on the corresponding control register setting.

5.6. Interrupt Vector Addresses

All interrupt vector addresses for the S3F8S45 interrupt structure are stored in the vector address area of the internal 16KB ROM, 0h–3FFFh; see Figure 47.

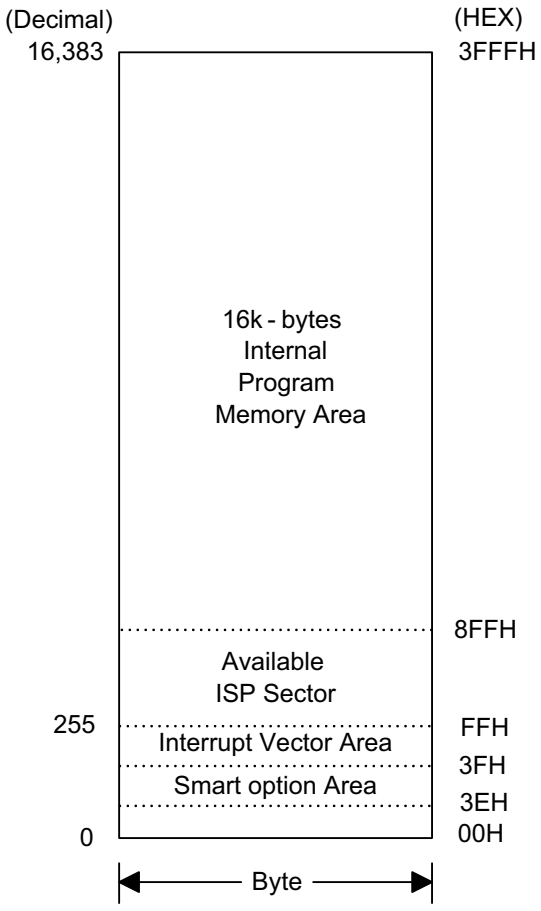


Figure 47. ROM Vector Address Area

Allocate unused locations in the vector address area as normal program memory. However, be careful not to overwrite any of the stored vector addresses, which are listed in Table 12.

Table 12. Interrupt Vectors

Vector Address			Request		Reset/Clear	
Decimal Value	Hex Value	Interrupt Source	Interrupt Level	Priority in Level	Hardware	Software
256	100h	Basic timer overflow	Reset	–	√	
206	CEh	Timer A match/capture	IRQ0	0	√	√
208	D0h	Timer A overflow		1	√	√
210	D2h	Timer B match	IRQ1	–	√	
212	D4h	Timer C match/overflow	IRQ2	0	√	√
216	D8h	Timer D0 match/capture	IRQ3	0	√	√
218	DAh	Timer D0 overflow		1	√	√
220	DCh	Timer D1 match/capture		2	√	√
222	DEh	Timer D1 overflow		3	√	√
228	E4h	SIO interrupt	IRQ4	0		√
230	E6h	Watch timer overflow		1		√
232	E8h	UART0 data transmit	IRQ5	0		√
234	EAh	UART0 data receive		1		√
236	ECh	UART1 data transmit		2		√
238	EEh	UART1data receive		3		√
240	F0h	P3.3 external interrupt	IRQ6	0		√
242	F2h	P3.4 external interrupt		1		√
244	F4h	P3.5 external interrupt		2		√
246	F6h	P3.6 external interrupt		3		√
248	F8h	P3.0 external interrupt	IRQ7	0		√
250	FAh	P3.1 external interrupt		1		√
252	FCh	P3.2 external interrupt		2		√
254	FEh	P0.3 external interrupt		3		√

Note: Interrupt priorities are identified in inverse order: 0 is the highest priority, 1 is the next highest priority, etc. If two or more interrupts within the same level contend, the interrupt with the lowest vector address usually has priority over an interrupt with a higher vector address. Priorities within a given level are fixed in hardware.

The program reset address in ROM is `0100h`. This reset address can be changed using the Smart Option; refer to the Embedded Flash Memory Interface chapter on page 327 to learn more.

5.7. Enable/Disable Interrupt Instructions

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

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

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

5.8. System-Level Interrupt Control Registers

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

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

A summary of these interrupt control registers is provided in Table 13.

Table 13. Interrupt Control Register Overview

Control Register	Mnemonic	R/W	Function Description
Interrupt Mask Register	IMR	R/W	Bit settings in the IMR Register enable or disable interrupt processing for each of the eight interrupt levels: IRQ0–IRQ7.
Interrupt Priority Register	IPR	R/W	Controls the relative processing priorities of the interrupt levels. The eight levels of the S3F8S45 MCU are organized into three groups: A, B, and C. Group A is IRQ0 and IRQ1, Group B is IRQ2, IRQ3, IRQ4, and Group C is IRQ5, IRQ6, and IRQ7.
Interrupt Request Register	IRQ	R	This register contains a request pending bit for each interrupt level.
System Mode Register	SYM	R/W	This register enables/disables fast interrupt processing, dynamic global interrupt processing, and external interface control (an external memory interface is implemented in the S3F8S45 microcontroller).

Note: Interrupts must be disabled prior to modifying the contents of the IMR Register. Zilog recommends using the DI instruction.

5.9. Interrupt Processing Control Points

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

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

Figure 48 diagrams the functions of these combined interrupt processes.

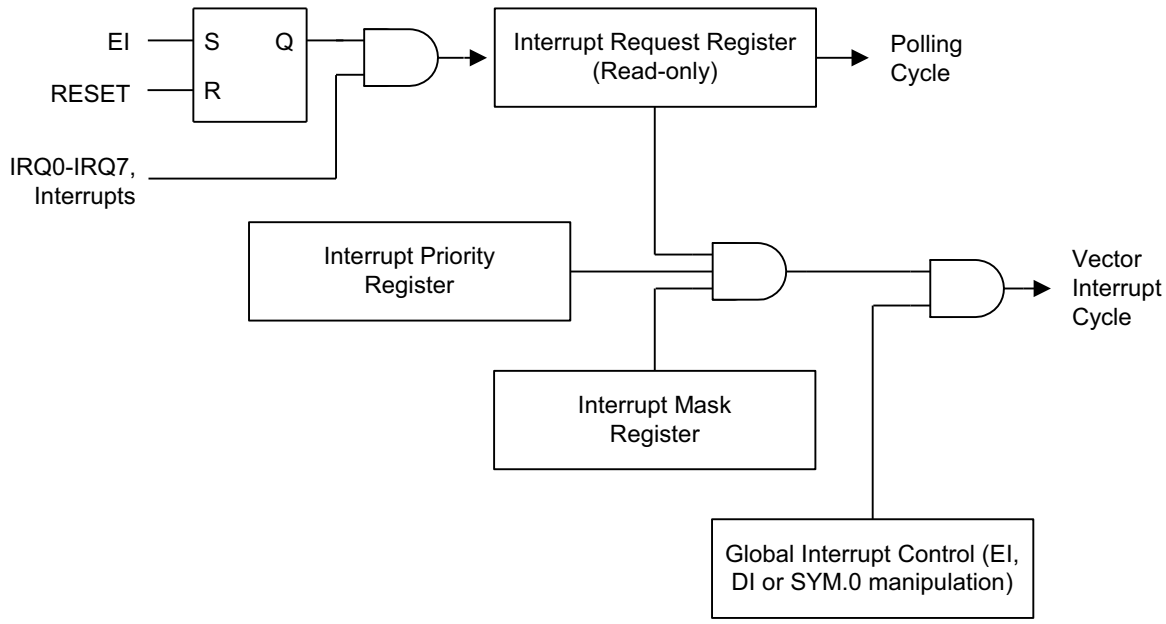


Figure 48. Interrupt Function Diagram

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

5.10. Peripheral Interrupt Control Registers

For each interrupt source, there are one or more corresponding peripheral control registers that let you control the interrupt generated by its corresponding peripheral; see Table 14.

Table 14. Interrupt Source Control and Data Registers

Interrupt Source	Interrupt Level	Register(s)	Location(s)
Timer A match/capture Timer A overflow	IRQ0	TACON TACNT TADATA	E2h, Set1, Bank0 E0h, Set1, Bank0 E1h, Set1, Bank0
Timer B match	IRQ1	TBCON TBDATAH TBDATAL	E3h, Set1, Bank0 E4h, Set1, Bank0 E5h, Set1, Bank0
Timer C match/overflow	IRQ2	TCCON TCCNT TCDATA	ECh, Set1, Bank0 EAh, Set1, Bank0 EBh, Set1, Bank0
Timer D0 match/capture Timer D0 overflow Timer D1 match/capture Timer D1 overflow	IRQ3	TD0CON TD0CNTH TD0CNTL TD0DATAH TD0DATAL TD1CON TD1CNTH TD1CNTL TD1DATAH TD1DATAL	01h, page 4 02h, page 4 03h, page 4 04h, page 4 05h, page 4 0Ah, page 4 06h, page 4 07h, page 4 08h, page 4 09h, page 4
SIO interrupt Watch timer overflow	IRQ4	SIOCON SIODATA SIOPS WTCON	E7h, Set1, Bank0 E8h, Set1, Bank0 E9h, Set1, Bank0 E6h, Set1, Bank0
UART0 data transmit UART0 data receive UART1 data transmit UART1 data receive	IRQ5	UART0CONH UART0CONL UDATA0 BRDATA0 UART1CONH UART1CONL UDATA1 BRDATA1	EEh, Set1, Bank0 EFh, Set1, Bank0 F0h, F1h, Set1, Bank0 F2h, Set1, Bank0 F3h, Set1, Bank0 F4h, F5h, Set1, Bank0

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

Table 14. Interrupt Source Control and Data Registers (Continued)

P3.3 external interrupt	IRQ6	P3CONH	EAh, Set1, Bank1
P3.4 external interrupt		P3CONL	EBh, Set1, Bank1
P3.5 external interrupt		P03INTL	E9h, Set1, Bank1
P3.6 external interrupt		P03PND	E7h, Set1, Bank1
P3.0 external interrupt	IRQ7	P3CONL	EBh, Set1, Bank1
P3.1 external interrupt		P0CONL	D1h, Set1, Bank1
P3.2 external interrupt		P03INTH	E8h, Set1, Bank1
P0.3 external interrupt		P03PND	E7h, Set1, Bank1

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

5.11. System Mode Register

The System Mode (SYM) Register, shown in Table 15, is used to globally enable and disable interrupt processing and to control fast interrupt processing.

Table 15. System Mode Register (SYM; Set1, Bank0)

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

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

Bit	**Description**
[7]	Reserved
[6:5]	**Reserved**

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

Bit	Description (Continued)
[4:2]	**Fast Interrupt Level Selection Bits**[1] 000: IRQ0. 001: IRQ1. 010: IRQ2. 011: IRQ3. 100: IRQ4. 101: IRQ5. 110: IRQ6. 111: IRQ7.
[1]	**Fast Interrupt Enable Bit**[2] 0: Disable fast interrupt processing. 1: Enable fast interrupt processing.
[0]	**Global Interrupt Enable Bit**[3] 0: Disable global interrupt processing. 1: Enable global interrupt processing.

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

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

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

5.12. Interrupt Mask Register

The Interrupt Mask (IMR) Register, shown in Table 16, is used to enable or disable interrupt processing for individual interrupt levels. After a reset, all IMR bit values are undetermined and must therefore be written to their required settings by the initialization routine.

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

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

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

Table 16. Interrupt Mask Register (IMR; Set1)

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

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

Bit	Description
[7]	**Interrupt Level 7 (IRQ7) Enable Bit; External Interrupts P3.0–P3.2, P0.3** 0: Disable (Mask). 1: Enable (Unmask).
[6]	**Interrupt Level 6 (IRQ6) Enable Bit; External Interrupts P3.3–P3.6** 0: Disable (Mask). 1: Enable (Unmask).
[5]	**Interrupt Level 5 (IRQ5) Enable Bit; UART0/1 Transmit, UART0/1 Receive** 0: Disable (Mask). 1: Enable (Unmask).
[4]	**Interrupt Level 4 (IRQ4) Enable Bit; Watch Timer, SIO** 0: Disable (Mask). 1: Enable (Unmask).
[3]	**Interrupt Level 3 (IRQ3) Enable Bit; Timer D0/1 Match/Capture or Overflow** 0: Disable (Mask). 1: Enable (Unmask).
[2]	**Interrupt Level 2 (IRQ2) Enable Bit; Timer C Match/Overflow** 0: Disable (Mask). 1: Enable (Unmask).
[1]	**Interrupt Level 1 (IRQ1) Enable Bit; Timer B Match** 0: Disable (Mask). 1: Enable (Unmask).
[0]	**Interrupt Level 0 (IRQ0) Enable Bit; Timer A Match/Capture or Overflow** 0: Disable (Mask). 1: Enable (Unmask).

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

5.13. Interrupt Priority Register

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

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

To support programming of the relative interrupt level priorities, interrupts are organized into groups and subgroups by the interrupt logic. These groups and subgroups are used only by IPR logic for the IPR Register priority definitions, as listed below and shown in Figure 49.

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

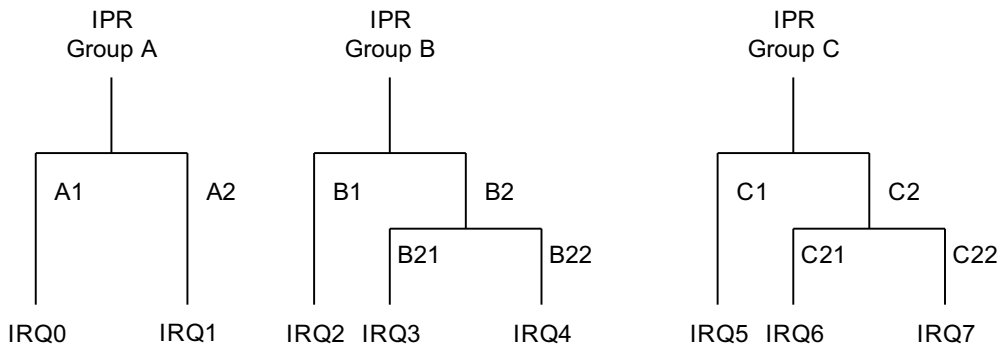


Figure 49. Interrupt Request Priority Groups

In the Interrupt Priority Register, shown in Table 17, IPR.7, IPR.4, and IPR.1 control the relative priority of interrupt groups A, B, and C. For example, a setting of 001b for these bits would select the group relationship B > C > A; a setting of 101b would select the relationship C > B > A.

Table 17. Interrupt Priority Register (IPR; Set1, Bank0)

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

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

Bit	Description
[7, 4, 1]	**Priority Control Bits for Interrupt Groups A, B, and C** 000: Group priority undefined 001: B > C > A. 010: A > B > C. 011: B > A > C. 100: C > A > B. 101: C > B > A. 110: A > C > B. 111: Group priority undefined.
[6]	**Interrupt Subgroup C Priority Control Bit** 0: IRQ6 > IRQ7. 1: IRQ7 > IRQ6.
[5]	**Interrupt Group C Priority Control Bit** 0: IRQ5 > (IRQ6, IRQ7). 1: (IRQ6, IRQ7) > IRQ5.
[3]	**Interrupt Subgroup B Priority Control Bit*** 0: IRQ3 > IRQ4. 1: IRQ4 > IRQ3.
[2]	**Interrupt Group B Priority Control Bit*** 0: IRQ2 > (IRQ3, IRQ4). 1: (IRQ3, IRQ4) > IRQ2.
[0]	**Interrupt Group A Priority Control Bit** 0: IRQ0 > IRQ1. 1: IRQ1 > IRQ0.

Note: Interrupt group A: IRQ0, IRQ1. Interrupt group B: IRQ2, IRQ3, IRQ4. Interrupt group C: IRQ5, IRQ6, IRQ7.

The functions of the other IPR bit settings are:

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

5.14. Interrupt Request Register

Bit values in the Interrupt Request (IRQ) Register, shown in Table 18, can be polled to monitor interrupt request status for all levels in the microcontroller's interrupt structure. Each bit corresponds to the interrupt level of the same number: bit 0 to IRQ0, bit 1 to IRQ1, etc. A 0 indicates that no interrupt request is currently being issued for that level; a 1 indicates that an interrupt request has been generated for that level.

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

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

Table 18. Interrupt Request Register (IPQ; Set1)

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

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

Bit	**Description**
[7]	**Level 7 (IRQ7) Request Pending Bit; External Interrupts P3.0–P3.2, P0.3** 0: Not pending. 1: Pending.
[6]	**Level 6 (IRQ6) Request Pending Bit; External Interrupts P3.3–P3.6** 0: Not pending. 1: Pending.
[5]	**Level 5 (IRQ5) Request Pending Bit; UART0/1 Transmit, UART0/1 Receive** 0: Not pending. 1: Pending.
[4]	**Level 4 (IRQ4) Request Pending Bit; Watch Timer, SIO** 0: Not pending. 1: Pending.

Bit	Description (Continued)
[3]	**Level 3 (IRQ3) Request Pending Bit; Timer D0/1 Match/Capture or Overflow** 0: Not pending. 1: Pending.
[2]	**Level 2 (IRQ2) Request Pending Bit; Timer C Match/Overflow** 0: Not pending. 1: Pending.
[1]	**Level 1 (IRQ1) Request Pending Bit; Timer B Match** 0: Not pending. 1: Pending.
[0]	**Level 0 (IRQ0) Request Pending Bit; Timer A Match/Capture or Overflow** 0: Not pending. 1: Pending.

5.15. Interrupt Pending Function Types

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

5.16. Pending Bits Cleared Automatically by Hardware

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

In the S3F8S45 MCU's interrupt structure, the Timer A overflow interrupt (IRQ0), the Timer B match interrupt (IRQ1), the Timer C overflow interrupt (IRQ2), and the Timer D0/D1 overflow interrupt (IRQ3) belong to this category of interrupts in which a pending condition is cleared automatically by hardware.

5.17. Pending Bits Cleared by the Service Routine

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

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

Table 19. Interrupt Mask Register (INTPND; Set1, Bank0)

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

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

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

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

Example 1

```
SB1
LD  P03PND, #11111011b      ; Clear P3.5's interrupt pending bit
•
•
```

```
•
IRET
```

Example 2

```
SB0
LD  INTPND, #11111101b      ; Clear Timer A match/capture
                            ; interrupt pending bit
•
•
•
IRET
```

5.18. Interrupt Source Polling Sequence

Interrupt request polling and servicing occurs via the following sequence:

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

5.19. Interrupt Service Routines

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

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

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

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

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

5.20. Generating Interrupt Vector Addresses

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

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

Note: A 16-bit vector address always begins at an even-numbered ROM address within the range 00h–FFh.

5.21. Nesting of Vectored Interrupts

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

1. Push the current 8-bit Interrupt Mask (IMR) Register value to the stack (PUSH IMR).
2. Load the IMR Register with a new mask value that enables only the higher-priority interrupt.
3. Execute an EI instruction to enable interrupt processing (a higher-priority interrupt will be processed if it occurs).

4. When the lower-priority interrupt service routine ends, restore the IMR to its original value by returning the previous mask value from the stack (POP IMR).
5. Execute an IRET.

Depending on the application, the above procedure can be simplified to some extent.

5.22. Instruction Pointer

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

The contents of the Instruction Pointer High Byte (IPH) and Instruction Pointer Low Byte (IPL) registers are described in Tables 20 and 21.

Table 20. Instruction Pointer High Byte (IPH; Set1)

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

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

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

Table 21. Instruction Pointer Low Byte (IPL; Set1)

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

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

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

5.23. Fast Interrupt Processing

This feature lets you specify that an interrupt within a given level be completed in approximately six clock cycles instead of the usual 22 clock cycles. To select a specific interrupt level for fast interrupt processing, you write the appropriate 3-bit value to SYM.4–SYM.2. Then, to enable fast interrupt processing for the selected level, you set SYM.1 to 1.

Two other system registers support fast interrupts processing:

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

Note: For the S3F8S45 microcontroller, the service routine for any one of the eight interrupt levels: IRQ0–IRQ7, can be selected for fast interrupt processing.

5.24. Procedure for Initiating Fast Interrupts

To initiate fast interrupt processing, follow these steps:

1. Load the start address of the service routine into the instruction pointer (IP).
2. Load the interrupt level number (IRQn) into the fast interrupt selection field (SYM.4–SYM.2).

3. Write a 1 to the fast interrupt enable bit in the SYM Register.

5.25. Fast Interrupt Service Routine

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

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

5.26. Relationship to Interrupt Pending Bit Types

As described previously, there are two types of interrupt pending bits; one is automatically cleared by hardware after the interrupt service routine is acknowledged and executed, and the other must be cleared by the application software's interrupt service routine. Fast interrupt processing can be selected for interrupts with either type of pending condition clear function – hardware or software.

5.27. Programming Guidelines

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

Chapter 6. Instruction Set

The SAM88 instruction set, which comprises 78 instructions, is specifically designed to support the large register files that are typical of most SAM88 microcontrollers.

The powerful data manipulation capabilities and features of the SAM88 instruction set include:

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

6.1. Data Types

The SAM88 CPU performs operations on bits, bytes, BCD digits, and two-byte words. Bits in the register file can be set, cleared, complemented and tested. Bits within a byte are numbered from 7 to 0, in which bit 0 is the least significant (right-most) bit.

6.2. Register Addressing

To access an individual register, an 8-bit address in the range 0–255 or the 4-bit address of a working register is specified. Paired registers can be used to construct 16-bit data or 16-bit program memory or data memory addresses. To learn more about register addressing, see the Address Space chapter on page 16.

6.3. Addressing Modes

There are seven explicit addressing modes: Register (R), Indirect Register (IR), Indexed (X), Direct (DA), Relative (RA), Immediate (IM) and Indirect (IA). To learn more about these addressing modes, see the Addressing Modes chapter on page 44.

6.4. Instruction Summary

All instructions are summarized by type, operand, and description in Table 22.

Table 22. Instruction Group Summary

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

Table 22. Instruction Group Summary (Continued)

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

Table 22. Instruction Group Summary (Continued)

Bit Manipulation Instructions (continued)		
BOR	dst,src	Bit OR
BXOR	dst,src	Bit XOR
TCM	dst,src	Test complement under mask
TM	dst,src	Test under mask
Rotate and Shift Instructions		
RL	dst	Rotate left
RLC	dst	Rotate left through carry
RR	dst	Rotate right
RRC	dst	Rotate right through carry
SRA	dst	Shift right arithmetic
SWAP	dst	Swap nibbles
CPU Control Instructions		
CCF		Complement carry flag
DI		Disable interrupts
EI		Enable interrupts
IDLE		Enter Idle mode
NOP		No operation
RCF		Reset carry flag
SB0		Set Bank0
SB1		Set Bank1
SCF		Set carry flag
SRP	src	Set register pointers
SRP0	src	Set register pointer 0
SRP1	src	Set register pointer 1
STOP		Enter Stop Mode

6.5. Flags Register

The Flags (FLAGS) Register, shown in Table 23, contains eight bits that describe the current status of CPU operations. Four of these bits, FLAGS.7–FLAGS.4, can be tested and used with conditional jump instructions; two others, FLAGS.3 and FLAGS.2, are used for BCD arithmetic.

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

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

Table 23. Flags Register (FLAGS; Set1)

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

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

Bit	Description
[7]	**Carry Flag Bit (C)** 0: Operation does not generate a carry or borrow condition. 1: Operation generates a carry-out or borrow into high-order bit 7.
[6]	**Zero Flag Bit (Z)** 0: Operation result is a nonzero value. 1: Operation result is zero.
[5]	**Sign Flag Bit (S)** 0: Operation generates a positive number (MSB = 0). 1: Operation generates a negative number (MSB = 1).
[4]	**Overflow Flag Bit (V)** 0: Operation result is ≤ +127 or –128. 1: Operation result is > +127 or < –128.
[3]	**Decimal Adjust Flag Bit (D)** 0: Add operation completed. 1: Subtraction operation completed.
[2]	**Half-Carry Flag Bit (H)** 0: No carry-out of bit 3 or no borrow into bit 3 by addition or subtraction. 1: Addition generated carry-out of bit 3 or subtraction generated borrow into bit 3.

Bit	Description (Continued)
[1]	**Fast Interrupt Status Flag Bit (FIS)** 0: Interrupt return (IRET) in progress when read. 1: Fast interrupt service routine in progress when read.
[0]	**Bank Address Selection Flag Bit (BA)** 0: Bank0 is selected. 1: Bank1 is selected.

Table 24 describes each of the flags managed by the Flags Register.

Table 24. Flag Descriptions

Flag	Description
C	**Carry Flag (FLAGS.7)** The C flag is set to 1 if the result from an arithmetic operation generates a carry-out from or a borrow to the bit 7 position (MSB). After rotate and shift operations, it contains the last value shifted out of the specified register. Program instructions can set, clear, or complement the carry flag.
Z	**Zero Flag (FLAGS.6)** For arithmetic and logic operations, the Z flag is set to 1 if the result of the operation is zero. For operations that test register bits, and for shift and rotate operations, the Z flag is set to 1 if the result is logic 0.
S	**Sign Flag (FLAGS.5)** Following the arithmetic, logic, rotate, or shift operations, the sign bit identifies the state of the MSB of the result. A logic 0 indicates a positive number and a logic 1 indicates a negative number.
V	**Overflow Flag (FLAGS.4)** The V flag is set to 1 when the result of a two's-complement operation is greater than +127 or less than –128. It is also cleared to 0 following logic operations.
D	**Decimal Adjust Flag (FLAGS.3)** The DA bit is used to specify what type of instruction is executed last during BCD operations, so that a subsequent decimal adjust operation can execute correctly. The DA bit is not usually accessed by programmers, and cannot be used as a test condition.
H	**Half-Carry Flag (FLAGS.2)** The H bit is set to 1 whenever an addition generates a carry-out of bit 3, or when a subtraction borrows out of bit 4. It is used by the Decimal Adjust (DA) instruction to convert the binary result of a previous addition or subtraction into the correct decimal (BCD) result. The H flag is seldom accessed directly by a program.

Table 24. Flag Descriptions (Continued)

Flag	Description
FIS	**Fast Interrupt Status Flag (FLAGS.1)** The FIS bit is set during a fast interrupt cycle and reset during the IRET following interrupt servicing. When set, it inhibits all interrupts and causes the fast interrupt return to be executed when the IRET instruction is executed.
BA	**Bank Address Flag (FLAGS.0)** The BA flag indicates which register bank in the Set1 area of the internal register file is currently selected, Bank0 or Bank1. The BA flag is cleared to 0 (select Bank0) when you execute the SB0 instruction and is set to 1 (select Bank1) when you execute the SB1 instruction.

6.6. Instruction Set Notation

Table 25 lists the conventions used for each of the flags managed by the Instruction Set. Symbols for the Instruction Set are listed in Table 26; conditions for these symbols are described in Table 27.

Table 25. Flag Notation Conventions

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

Table 26. Instruction Set Symbols

Symbol	Description
dst	Destination operand.
src	Source operand.
@	Indirect register address prefix.

Table 26. Instruction Set Symbols (Continued)

Symbol	Description
PC	Program counter.
IP	Instruction pointer.
FLAGS	Flags register (D5h).
RP	Register pointer.
#	Immediate operand or register address prefix.
H	Hexadecimal number suffix.
D	Decimal number suffix.
B	Binary number suffix.
opc	Op code.

Table 27. Instruction Notation Conventions

Notation	Description	Actual Operand Range
cc	Condition code	See list of condition codes in Table 30 on page 93
r	Working register only	Rn (n= 0–15)
rb	Bit (b) of working register	Rn.b (n = 0–15, b = 0–7)
r0	Bit 0 (LSB) of working register	Rn (n = 0–15)
rr	Working register pair	RRp (p = 0, 2, 4, .., 14)
R	Register or working register	reg or Rn (reg = 0–255, n = 0–15)
Rb	Bit (b) of register or working register	reg.b (reg = 0–255, b = 0–7)
RR	Register pair or working register pair	reg or RRp (reg = 0–254, even number only, in which p = 0, 2, .., 14)
IA	Indirect Addressing Mode	addr (addr = 0–254, even number only)
Ir	Indirect working register only	@Rn (n = 0–15)
IR	Indirect register or indirect working register	@Rn or @reg (reg = 0–255, n = 0–15)
Irr	Indirect working register pair only	@RRp (p = 0, 2, .., 14)
IRR	Indirect register pair or indirect working register pair	@RRp or @reg (reg = 0–254, even only, in which p = 0, 2, .., 14)
X	Indexed Addressing Mode	#reg [Rn] (reg = 0–255, n = 0–15)
XS	Indexed (Short Offset) Addressing Mode	#addr [RRp] (addr = range –128 to +127, in which p = 0, 2, .., 14)
xl	Indexed (Long Offset) Addressing Mode	#addr [RRp] (addr = range 0–65535, in which p = 0, 2, .., 14)

Table 27. Instruction Notation Conventions (Continued)

Notation	Description	Actual Operand Range
da	Direct Addressing Mode	addr (addr = range 0–65535)
ra	Relative Addressing Mode	addr (addr = number in the range +127 to –128 that is an offset relative to the address of the next instruction)
im	Immediate Addressing Mode	#data (data = 0–255)
iml	Immediate (Long) Addressing Mode	#data (data = range 0–65535)

Chapter 7. Op Code Maps

Tables 28 and 29 provide quick reference op code maps to addresses 0–7 and 8–F, respectively.

Table 28. Op Code Quick Reference (0–7)

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

Table 29. Op Code Quick Reference (8–F)

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

7.1. Condition Codes

The op code of a conditional jump always contains a 4-bit field called the *condition code* (cc) that specifies under which conditions it is to execute the jump. For example, a conditional jump with a condition code of *equal* after a compare operation only jumps if the two operands are equal. The carry (C), zero (Z), sign (S), and overflow (V) flags are used to control the operation of conditional jump instructions.

These condition codes are listed in Table 30.

Table 30. Condition Codes[1]

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

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

7.2. Instruction Descriptions

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

The following information is included in each instruction description:

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

Add with Carry

ADC dst, src

Operation dst ← dst + src + c

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

Flags

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

Format

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

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

ADC	R1, R2	→	R1 = 14h, R2 = 03h
ADC	R1, @R2	→	R1 = 1Bh, R2 = 03h
ADC	01h, 02h	→	Register 01h = 24h, register 02h = 03h
ADC	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 *ADC R1, R2* statement adds `03h` and the carry flag value (1) to the destination value `10h`, leaving `14h` in register R1.

Add

ADD dst, src

Operation dst ← dst + src

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

Flags

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

Format

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

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

ADC	R1, R2	→	R1 = 15h, R2 = 03h
ADC	R1, @R2	→	R1 = 1Ch, R2 = 03h
ADC	01h, 02h	→	Register 01h = 24h, register 02h = 03h
ADC	01h, @02h	→	Register 01h = 2Bh, register 02h = 03h
ADC	01h, #25h	→	Register 01h = 46h

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

Logical AND

AND dst, src

Operation dst ← dst AND src

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

Flags

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

Format

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

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

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

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

Bit AND

BAND dst, src.b

BAND dst.b, src

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

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

Flags

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

Format

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

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

Example Assume that R1 = 07h and register 01h = 05h.

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

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

Bit Compare

BCP dst, src.b

Operation dst(0)–src(b)

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

Flags

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

Format

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

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

Example Assume that R1 = 07h and register 01h = 01h.

BCP R1, 01h.1 → R1 = 07h, register 01h = 01h

If destination working register R1 contains the value 07h (00000111b) and the source register 01h contains the value 01h (00000001b), the *BCP R1, 01h.1* statement compares bit 1 of the source register (01h) and bit 0 of the destination register (R1). Because the bit values are not identical, the zero flag bit (Z) is cleared in the Flags Register (0D5h).

Bit Complement

BITC dst.b

Operation dst(b) ← NOT dst(b)

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

Flags

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

Format

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

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

Example Assume that R1 = 07h.

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

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

Bit Reset

BITR dst.b

Operation dst(b) ← 0

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

Flags No flags are affected.

Format

	Bytes	Cycles	Op Code (Hex)	Address Mode dst
opc | dst | b | 0	2	4	77	rb

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

Example Assume that R1 = 07h.

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

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

Bit Set

BITS dst.b

Operation dst(b) ← 1

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

Flags No flags are affected.

Format

		Bytes	Cycles	Op Code (Hex)	Address Mode dst
opc	dst | b | 1	2	4	77	rb

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

Example Assume that R1 = 07h.

```
BITR    R1.3        →    R1 = 0Fh
```

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

Bit OR

BOR dst, src.b

BOR dst.b, src

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

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

Flags

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

Format

			Bytes	Cycles	Op Code (Hex)	Address Mode dst	Address Mode src
opc	dst | b | 0	src	3	6	07	r0	Rb
opc	src | b | 1	dst	3	6	07	Rb	r0

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

Example Assume that R1 = 07h and register 01h = 03h.

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

In the first example, destination working register R1 contains the value 07h (00000111b) and source register 01h the value 03h (00000011b). The *BOR R1, 01h.1* statement logically ORs bit 1 of register 01h (source) with bit 0 of R1 (destination). This leaves the same value (07h) in working register R1.

In the second example, destination register 01h contains the value 03h (00000011b) and the source working register R1 the value 07h (00000111b).

The *BOR 01h.2, R1* statement logically ORs bit 2 of register `01h` (destination) with bit 0 of R1 (source). This leaves the value `07h` in register `01h`.

Bit Test, Jump Relative on False

BTJRF dst, src.b

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

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

Flags No flags are affected.

Format

			Bytes	Cycles	Op Code (Hex)	Address Mode dst	Address Mode src
opc	dst | b | 1	dst	3	10	37	RA	rb

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

Example Assume that R1 = 07h.

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

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

Bit Test, Jump Relative on True

BTJRT dst, src.b

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

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

Flags No flags are affected.

Format

			Bytes	Cycles	Op Code (Hex)	Address Mode dst	Address Mode src
opc	src | b | 1	dst	3	10	37	RA	rb

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

Example Assume that R1 = 07h.

```
BTJRT   SKIP, R1.1
```

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

Bit XOR

BXOR dst, src.b

BXOR dst.b, src

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

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

Flags

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

Format

			Bytes	Cycles	Op Code (Hex)	Address Mode dst	Address Mode src
opc	dst | b | 0	src	3	6	27	r0	Rb
opc	src |b | 1	dst	3	6	27	Rb	r0

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

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

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

In the first example, destination working register R1 has the value 07h (00000111b) and source register 01h has the value 03h (00000011b). The *BXOR R1, 01h.1* statement exclusive-ORs bit 1 of register 01h (source) with bit 0 of R1 (destination). The result bit value is stored in bit 0 of R1, changing its value from 07h to 06h. The value of source register 01h is unaffected.

Call Procedure

CALL dst

Operation

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

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

Flags No flags are affected.

Format

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

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

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

In the first example, if the program counter value is 1A47h and the stack pointer contains the value 0002h, the *CALL 3521h* statement pushes the current PC value onto the top of the stack. The stack pointer now points to mem-

ory location 0000h. The PC is then loaded with the value 3521h, the address of the first instruction in the program sequence to be executed.

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

Complement Carry Flag

CCF

Operation C ← NOT C

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

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

Format

	Bytes	Cycles	Op Code (Hex)
opc	1	4	EF

Example Assume that the carry flag = 0.

```
CCF
```

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

Clear

CCF dst

Operation dst ← 0

The destination location is cleared to 0.

Flags No flags are affected.

Format

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

Example Assume that Register 00h = 4Fh, register 01h = 02h, and register 02h = 5Eh.

```
CLR   00h     →   Register 00h = 00h
CLR   @01h    →   Register 01h = 02h, register 02h = 00h
```

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

Complement

COM dst

Operation dst ← NOT dst

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

Flags

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

Format

		Bytes	Cycles	Op Code (Hex)	Address Mode dst
opc	dst	2	4	60	R
			4	61	IR

Example Assume that R1 = 07h and register 07h = 0F1h.

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

In the first example, destination working register R1 contains the value 07h (00000111b). The *COM R1* statement complements all of the bits in R1: all logic 1s are changed to logic 0s, and vice-versa, leaving the value 0F8h (11111000b).

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

Compare

CP dst, src

Operation dst ← src

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

Flags

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

Format

			Bytes	Cycles	Op Code (Hex)	Address Mode dst	Address Mode src
opc	dst | src		2	4	A2	r	r
				6	A3	r	Ir
opc	src	dst	3	6	A4	R	R
				6	A5	R	IR
opc	dst	src	3	6	A6	R	IM

Example 1. Assume that R1 = `02h` and R2 = `03h`.

```
CP      R1, R2          →       Set the C and S flags
```

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

2. Assume that R1 = `05h` and R2 = `0Ah`

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

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

Compare, Increment, and Jump on Equal

CPIJE dst, src, RA

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

Ir ← Ir + 1

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

Flags No flags are affected.

Format

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

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

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

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

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

Compare, Increment, and Jump on NonEqual

CPIJE dst, src, RA

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

Ir ← Ir + 1

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

Flags No flags are affected.

Format

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

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

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

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

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

Decimal Adjust

DA dst

Operation dst ← DA dst

The destination operand is adjusted to form two 4-bit BCD digits following an addition or subtraction operation. For addition (ADD, ADC) or subtraction (SUB, SBC), Table 31 indicates the operations performed. These operations are undefined if the destination operand is not the result of a valid addition or subtraction of BCD digits.

Table 31. DA Instruction

Instruction	Carry Before DA	Bits 4–7 Value (Hex)	H Flag Before DA	Bits 0–3 Value (Hex)	Number Added to Byte	Carry After DA
ADD ADC	0	0–9	0	0–9	00	0
	0	0–8	0	A–F	06	0
	0	0–9	1	0–3	06	0
	0	A–F	0	0–9	60	1
	0	9–F	0	A–F	66	1
	0	A–F	1	0–3	66	1
	1	0–2	0	0–9	60	1
	1	0–2	0	A–F	66	1
	1	0–3	1	0–3	66	1
SUB SBC	0	0–9	0	0–9	00 =–00	0
	0	0–8	1	6–F	FA =–06	0
	1	7–F	0	0–9	A0 =–60	1
	1	6–F	1	6–F	9A =–66	1

Flags

C Set if there is a carry from the most significant bit; cleared otherwise (see Table 31).

Z Set if result is 0; cleared otherwise.

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

V Undefined.

D Unaffected.

H Unaffected.

Format

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

Example

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

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

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

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

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

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

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

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

Decrement

DEC dst

Operation dst ← dst – 1

The contents of the destination operand are decremented by one.

Flags

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

Format

opc	dst

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

Example Assume that R1 = 03h and register 03h = 10h.

DEC	R1	→	R1 = 02h
DEC	@R1	→	Register 03h = 0Fh

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

Decrement Word

DECW dst

Operation dst ← dst – 1

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

Flags

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

Format

opc	dst

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

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

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

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

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

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

Disable Interrupts

DI

Operation SYM (0) ← 0

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

Flags No flags are affected.

Format

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

Example Assume that SYM = `01h`.

```
DI
```

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

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

Divide (Unsigned)

DIV dst, src

Operation dst ÷ src

dst (UPPER) ← REMAINDER

dst (LOWER) ← QUOTIENT

The destination operand (16 bits) is divided by the source operand (8 bits). The quotient (8 bits) is stored in the lower half of the destination. The remainder (8 bits) is stored in the upper half of the destination. When the quotient is „d 28, the numbers stored in the upper and lower halves of the destination for quotient and remainder are incorrect. Both operands are treated as unsigned integers.

Flags

C Set if the V flag is set and quotient is between 2^8 and $2^9 - 1$; cleared otherwise.

Z Set if divisor or quotient = 0; cleared otherwise.

S Set if MSB of quotient = 1; cleared otherwise.

V Set if quotient is $\geq$ 28 or if divisor = 0; cleared otherwise.

D Unaffected.

H Unaffected.

Format

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

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

Example Assume that R0 = 10h, R1 = 03h, R2 = 40h, register 40h = 80h.

```
DIV   RR0, R2    →   R0 = 03h, R1 = 40h
DIV   RR0, @R2   →   R0 = 03h, R1 = 20h
DIV   RR0, #20h  →   R0 = 03h, R1 = 80h
```

In the first example, destination working register pair RR0 contains the values 10h (R0) and 03h (R1), and register R2 contains the value 40h. The *DIV RR0, R2* statement divides the 16-bit RR0 value by the 8-bit value of the R2 (source) register. After the DIV instruction, R0 contains the value 03h and R1 contains

40h. The 8-bit remainder is stored in the upper half of the destination register RR0 (R0) and the quotient in the lower half (R1).

Decrement and Jump if NonZero

DJNZ r, dst

Operation r ← r – 1

If r ≠ 0, PC ← PC + dst

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

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

Flags No flags are affected.

Format

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

Example Assume that R1 = `02h` and LOOP is the label of a relative address.

```
SRP     #0C0h
DJNZ    R1, LOOP
```

DJNZ is typically used to control a loop of instructions. In many cases, a label is used as the destination operand instead of a numeric relative address value. In the example, working register R1 contains the value `02h`, and LOOP is the label for a relative address.

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

Enable Interrupts

EI

Operation SYM (0) ← 1

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

Flags No flags are affected.

Format

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

Example Assume that SYM = 00h.

```
EI
```

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

Enter

ENTER

Operation

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

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

Flags No flags are affected.

Format

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

Example Figure 50 shows an example of how to use an *ENTER* statement.

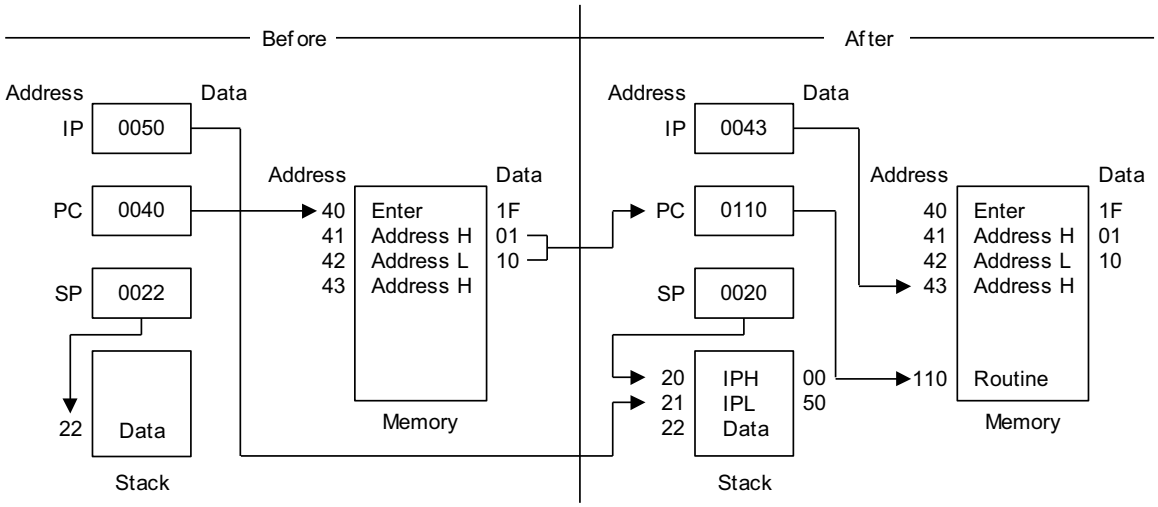


Figure 50. How to Use an ENTER Statement

Exit

EXIT

Operation

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

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

Flags No flags are affected.

Format

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

Example Figure 51 shows an example of how to use an *EXIT* statement.

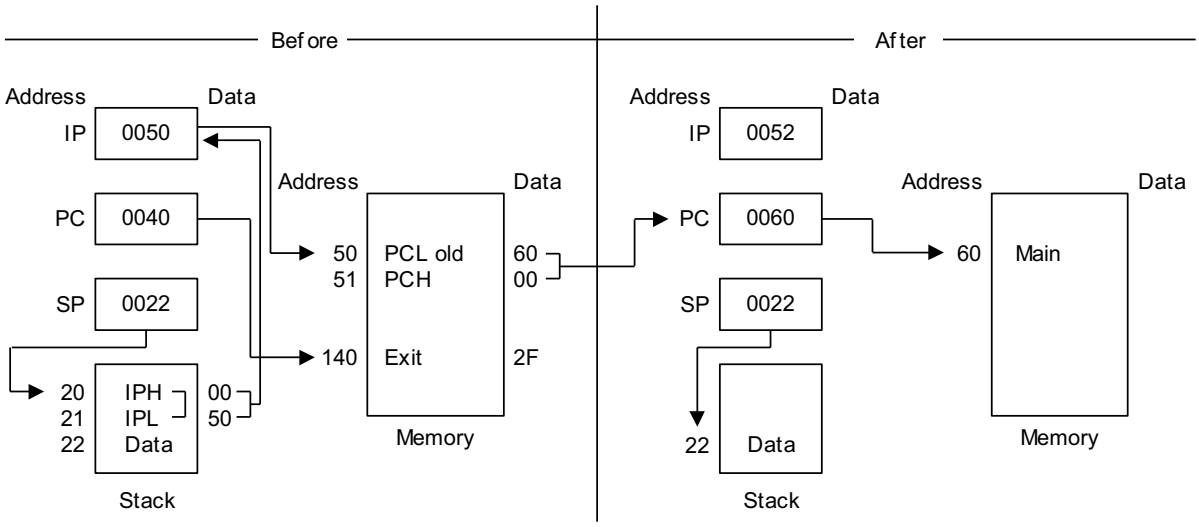


Figure 51. How to Use an EXIT Statement

Idle Operation

IDLE

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

Flags No flags are affected.

Format

	Bytes	Cycles	Op Code (Hex)
opc	1	4	6F

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

Increment

INC dst

Operation dst ← dst + 1

The contents of the destination operand are incremented by one.

Flags

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

Format

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

Example Assume that R0 = 1Bh, register 00h = 0Ch, and register 1Bh = 0Fh.

INC	R0	→	R0 = 1Ch
INC	00h	→	Register 00h = 0Dh
INC	@R0	→	R0 = 1Bh, register 01h = 10h

In the first example, if destination working register R0 contains the value 1Bh, the *INC R0* statement leaves the value 1Ch in that same register.

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

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

Increment Word

INCW dst

Operation dst ← dst + 1

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

Flags

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

Format

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

Example Assume that R0 = 1Ah, R1 = 02h, register 02h = 0Fh, and register 03h = 0FFh.

```
INCW    RR0     →    R0 = 1Ah, R1 = 03h
INCW    @R1     →    Register 02h = 10h, register 03h = 00h
```

In the first example, the working register pair RR0 contains the value 1Ah in register R0 and 02h in register R1. The *INCW RR0* statement increments the 16-bit destination by one, leaving the value 03h in register R1. In the second example, the *INCW @R1* statement uses Indirect Register (IR) Addressing Mode to increment the contents of general register 03h from 0FFh to 00h and register 02h from 0Fh to 10h.

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

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

Interrupt Return

IRET IRET (Normal), IRET (FAST)

Operation

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

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

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

Format

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

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

Example In the Figure 52, the instruction pointer is initially loaded with 100h in the main program before interrupts are enabled. When an interrupt occurs, the program counter and instruction pointer are swapped, causing the PC to jump to address 100h and the IP to keep the return address.

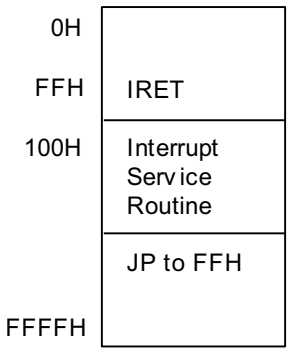


Figure 52. Instruction Pointer

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

The last instruction in the service routine normally is a jump to IRET at address FFh. This causes the instruction pointer to be loaded with 100h again and the program counter to jump back to the main program. Now, the next interrupt can occur and the IP is still correct at 100h.

Jump

JP cc, dst (Conditional)

JP dst (Unconditional)

Operation If cc is true, PC ← dst

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

Flags No flags are affected.

Format

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

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

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

```
JP    C, LABEL_W   →   LABEL_W = 1000h, PC = 1000h
JP    @00h         →   PC = 0120h
```

The first example shows a conditional JP. Assuming that the carry flag is set to 1, the *JP C, LABEL_W* statement replaces the contents of the PC with the value `1000h` and transfers control to that location. 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 *JP @00* statement replaces the contents of the PC with the contents of the register pair `00h` and `01h`, leaving the value `0120h`.

Jump Relative

JR cc, dst

Operation If cc is true, PC ← PC + dst

If the condition specified by the condition code (cc) is true, the relative address is added to the program counter and control passes to the statement for which the address is now in the program counter; otherwise, the instruction following the JR instruction is executed; see the list of condition codes in Table 30 on page 93.

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

Flags No flags are affected.

Format

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

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

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

```
JR      C, LABEL_X    →    PC = 1FF7h
```

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

Load

LD dst, src

Operation dst ← src

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

Flags No flags are affected.

Format

			Bytes	Cycles	Op Code (Hex)	Address Mode dst	Address Mode src
dst | src	src		2	4	rC	r	IM
				4	r8	r	R
src | opc	dst		2	4	r9 r = 0 to F	R	r
opc	dst | src		2	4	C7	r	Ir
				4	D7	Ir	r
opc	src	dst	3	6	E4	R	R
				6	E5	R	IR
opc	dst	src	3	6	E6	R	IM
				6	D6	IR	IM
opc	src	dst	3	6	F5	IR	R
opc	dst | src	x	3	6	87	r	x[r]
opc	src | dst	x	3	6	97	x[r]	r

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

```
LD   R0, #10h    →   R0 = 10h
LD   R0, 01h     →   R0 = 20h, register 01h = 20h
LD   01h, R0     →   Register 01h = 01h, R0 = 01h
```

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

Load Bit

LDB dst, src.b

LDB dst.b, src

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

The specified bit of the source is loaded into bit 0 (LSB) of the destination, or bit 0 of the source is loaded into the specified bit of the destination. No other bits of the destination are affected. The source is unaffected.

Flags No flags are affected.

Format

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

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

Example Assume that R0 = 06h and general register 00h = 05h.

```
LDB   R0, 00h.2   →   R0 = 07h, register 00h = 05h
LDB   00h.0, R0   →   R0 = 06h, register 00h = 04h
```

In the first example, destination working register R0 contains the value 06h and the source general register 00h the value 05h. The *LD R0, 00h.2* statement loads the bit 2 value of the 00h register into bit 0 of the R0 register, leaving the value 07h in register R0.

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

Load Memory

LDC/LDE dst, src

Operation dst ← src

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

Flags No flags are affected.

Format

	Bytes	Cycles	Op Code (Hex)	Address Mode dst	Address Mode src
opc | dst | src	2	10	C3	r	Irr
opc | src | dst	2	10	D3	Irr	r
opc | dst | src | XS	3	12	E7	r	XS[rr]
opc | src | dst | XS	3	12	F7	XS[rr]	r
opc | dst | src | XL_L | XL_H	4	14	A7	r	XL[rr]
opc | src | dst | XL_L | XL_H	4	14	B7	XL[rr]	r
opc | dst | 0000 | DA_L | DA_H	4	14	A7	r	DA
opc | src | 0000 | DA_L | DA_H	4	14	B7	DA	r
opc | dst | 0001 | DA_L | DA_H	4	14	A7	r	DA

				Bytes	Cycles	Op Code (Hex)	Address Mode dst	src
opc	src | 0001	DA_L	DA_H	4	14	B7	DA	r

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

Example

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

```
LDC     R0, @RR2            ; R0 ← contents of program memory location
                              0104h, R0 = Ah, R2 = 01h, R3 = 04h
LDE     R0, @RR2            ; R0 ← contents of external data memory
                              location 0104h, R0 = 2Ah, R2 = 01h, R3 = 04h
LDC*    @RR2, R0            ; 11h (contents of R0) is loaded into program
                              memory location 0104h (RR2), working
                              registers R0, R2, R3 → no change
LDE     @RR2, R0            ; 11h (contents of R0) is loaded into external
                              data memory location 0104h (RR2), working
                              registers R0, R2, R3 → no change
LDC     R0, #01h[RR2]       ; R0 ← contents of program memory location
                              0105h (01h + RR2), R0 = 6Dh, R2 = 01h, R3 =
                              04h
LDE     R0, #01h[RR2]       ; R0 ← contents of external data memory
                              location 0105h (01h + RR2), R0 = 7Dh, R2 =
                              01h, R3 = 04h
LDC*    #01h[RR2], R0       ; 11h (contents of R0) is loaded into program
                              memory location 0105h (01h + 0104h)
LDE     #01h[RR2], R0       ; 11h (contents of R0) is loaded into external
                              data memory location 0105h (01h + 0104h)
LDC     R0, #1000h[RR2]     ; R0 ← contents of program memory location
                              1104h (1000h + 0104h), R0 = 88h, R2 = 01h, R3
                              = 04h
```

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

```
LDE     R0, #1000h[RR2]     ; R0 ← contents of external data memory
                              location 1104h (1000h + 0104h), R0 = 98h, R2 =
                              01h, R3 = 04h
LDC     R0, 1104h           ; R0 ← contents of program memory location
                              1104h, R0 = 88h
LDE     R0, 1104h           ; 0 ← contents of external data memory
                              location 1104h, R0 = 98h
LDC*    1105h, R0           ; 11h (contents of R0) is loaded into program
                              memory location 1105h, (1105h) ← 11h
LDE     1105h, R0           ; 11h (contents of R0) is loaded into external
                              data memory location 1105h, (1105h) ← 11h
```

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

Load Memory and Decrement

LDCD/LDED dst, src

Operation dst ← src

rr ← rr – 1

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

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

Flags No flags are affected.

Format

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

Example Assume that R6 = 10h, R7 = 33h, R8 = 12h, program memory location 1033h = 0CDh, and external data memory location 1033h = 0DDh.

```
LDCD    R8, @RR6    ; 0CDh (contents of program memory location 1033h)
                    is loaded into R8 and RR6 is decremented by one, R8
                    = 0CDh, R6 = 10h, R7 = 32h (RR6 ← RR6 – 1)
LDED    R8, @RR6    ; 0DDh (contents of data memory location 1033h) is
                    loaded into R8 and RR6 is decremented by one (RR6
                    ← RR6 – 1), R8 = 0DDh, R6 = 10h, R7 = 32h
```

Load Memory and Increment

LDCI/LDEI dst, src

Operation dst ← src

rr ← rr + 1

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

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

Flags No flags are affected.

Format

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

Example Assume that R6 = 10h, R7 = 33h, R8 = 12h, program memory locations 1033h = 0CDh and 1034h = 0C5h, external data memory locations 1033h = 0DDh and 1034h = 0D5h.

```
LDCI    R8, @RR6    ; 0CDh (contents of program memory location 1033h)
                    ; is loaded into R8 and RR6 is incremented by one
                    ; (RR6 ← RR6 + 1), R8 = 0CDh, R6 = 10h, R7 = 34h
LDEI    R8, @RR6    ; 0DDh (contents of data memory location 1033h) is
                    ; loaded into R8 and RR6 is incremented by one (RR6
                    ; ← RR6 + 1), R8 = 0DDh, R6 = 10h, R7 = 34h
```

Load Memory with PreDecrement

LDCPD/ LDEPD dst, src

Operation rr ← rr – 1

dst ← src

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

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

Flags No flags are affected.

Format

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

Example Assume that R0 = 77h, R6 = 30h and R7 = 00h.

```
LDCPD   @RR6, R0    ; (RR6 ← RR6 – 1) 77h (contents of R0) is loaded into
                    program memory location 2FFFh (3000h – 1h), R0 =
                    77h, R6 = 2Fh, R7 = 0FFh
LDEPD   @RR6, R0    ; (RR6 ← RR6 – 1) 77h (contents of R0) is loaded into
                    external data memory location 2FFFh (3000h – 1h), R0
                    = 77h, R6 = 2Fh, R7 = 0FFh
```

Load Memory with PreIncrement

LDCPI/ LDEPI dst, src

Operation rr ← rr + 1

dst ← src

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

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

Flags No flags are affected.

Format

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

Example Assume that R0 = 7Fh, R6 = 21h and R7 = 0FFh.

```
LDCPI   @RR6, R0    ; (RR6 ← RR6 + 1) 7Fh (contents of R0) is loaded into
                    program memory location 2200h (21FFh + 1h), R0 =
                    7Fh, R6 = 22h, R7 = 00h
LDEPD   @RR6, R0    ; (RR6 ← RR6 + 1) 7Fh (contents of R0) is loaded into
                    external data memory location 2200h (21FFh + 1h), R0
                    = 7Fh, R6 = 22h, R7 = 00h
```

Load Word

LDW dst, src

Operation dst ← src

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

Flags No flags are affected.

Format

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

Example Assume that R4 = 06h, R5 = 1Ch, R6 = 05h, R7 = 02h, register 00h = Ah, register 01h = 02h, register 02h = 03h, and register 03h = 0Fh.

LDW	RR6, RR4	→	R6 = 06h, R7 = 1Ch, R4 = 06h, R5 = 1Ch
LDW	00h, 02h	→	Register 00h = 03h, register 01h = 0Fh, register 02h = 03h, register 03h = 0Fh
LDW	RR2, @R7	→	R2 = 03h, R3 = 0Fh
LDW	04h, @01h	→	Register 04h = 03h, register 05h = 0Fh
LDW	RR6, #1234h	→	R6 = 12h, R7 = 34h
LDW	02h, #0FEDh	→	Register 02h = 0Fh, register 03h = 0EDh

In the second example, the *LDW 00h, 02h* statement loads the contents of the source word 02h, 03h into the destination word 00h, 01h. This leaves the value 03h in general register 00h and the value 0Fh in register 01h.

The other examples show how to use the LDW instruction with multiple addressing modes and formats.

Multiply (Unsigned)

MULT dst, src

Operation dst ← dst × src

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

Flags

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

Format

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

Example Assume that Register 00h = 20h, register 01h = 03h, register 02h = 09h, register 03h = 06h.

MULT	00h, 02h	→	Register 00h = 01h, register 01h = 20h, register 02h = 09h
MULT	00h, @01h	→	Register 00h = 00h, register 01h = 0C0h
MULT	01h, @02h	→	Register 00h = 06h, register 01h = 00h

In the first example, the *MULT 00h, 02h* statement multiplies the 8-bit destination operand (in the register 00h of the register pair 00h, 01h) by the source register 02h operand (09h). The 16-bit product, 0120h, is stored in the register pair 00h, 01h.

Next

NEXT

Operation

PC ← @ IP

IP ← IP + 2

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

Flags

No flags are affected.

Format

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

Example

Figure 53 shows an example of how to use the NEXT instruction.

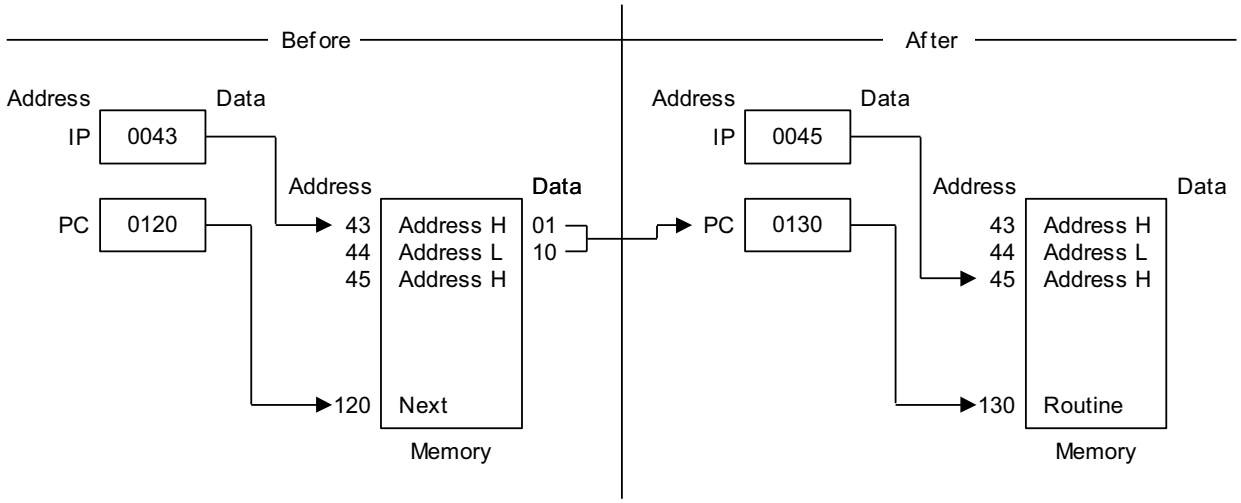


Figure 53. How to Use the NEXT Instruction

No Operation

NOP

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

Flags No flags are affected.

Format

	Bytes	Cycles	Op Code (Hex)
opc	1	4	FF

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

Logical OR

OR dst, src

Operation dst ← dst OR src

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

Flags

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

Format

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

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

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

In the first example, if working register R0 contains the value 15h and register R1 the value 2Ah, the *OR R0, R1* statement logical-ORs the R0 and R1 register contents and stores the result (3Fh) in destination register R0.

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

Pop from Stack

ADC dst

Operation dst ← @SP

SP ← SP + 1

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

Flags No flags are affected.

Format

opc	dst

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

Example Assume that Register 00h = 01h, register 01h = 1Bh, SPH (0D8h) = 00h, SPL (0D9h) = 0FBh and stack register 0FBh = 55h.

```
POP   00h    →   Register 00h = 55h, SP = 00FCh
POP   @00h   →   Register 00h = 01h, register 01h = 55h, SP = 00FCh
```

In the first example, general register 00h contains the value 01h. The *POP 00h* statement loads the contents of location 00FBh (55h) into destination register 00h and then increments the stack pointer by one. Register 00h then contains the value 55h and the SP points to location 00FCh.

Pop User Stack (Decrementing)

POPUD dst, src

Operation dst ← src

IR ← IR – 1

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

Flags No flags are affected.

Format

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

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

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

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

Pop User Stack (Incrementing)

POPUI dst, src

Operation dst ← src

IR ← IR + 1

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

Flags No flags are affected.

Format

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

Example Assume that Register 00h = 01h and register 01h = 70h.

POPUI 02h, @00h → Register 00h = 02h, register 01h = 70h, register 02h = 70h

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

Push to Stack

PUSH src

Operation SP ← SP – 1

@SP ← src

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

Flags No flags are affected.

Format

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

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

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

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

In the first example, if the stack pointer contains the value 0000h, and general register 40h the value 4Fh, the *PUSH 40h* statement decrements the stack pointer from 0000 to 0FFFFh. It then loads the contents of register 40h into location 0FFFFh and adds this new value to the top of the stack.

Push User Stack (Decrementing)

PUSHUD dst, src

Operation IR ← IR – 1

dst ← src

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

Flags No flags are affected.

Format

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

Example Assume that Register 00h = 03h, register 01h = 05h, and register 02h = 1Ah.

PUSHUD @00h, 01h → Register 00h = 02h, register 01h = 05h, register 02h = 05h

If the user stack pointer (register 00h, for example) contains the value 03h, the *PUSHUD @00h, 01h* statement decrements the user stack pointer by one, leaving the value 02h. The 01h register value, 05h, is then loaded into the register addressed by the decremented user stack pointer.

Push User Stack (Incrementing)

PUSHUI dst, src

Operation IR ← IR + 1

dst ← src

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

Flags No flags are affected.

Format

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

Example Assume that Register 00h = 03h, register 01h = 05h, and register 04h = 2Ah.

PUSHUI @00h, 01h → Register 00h = 04h, register 01h = 05h, register 04h = 05h

If the user stack pointer (register 00h, for example) contains the value 03h, the *PUSHUI @00h, 01h* statement increments the user stack pointer by one, leaving the value 04h. The 01h register value, 05h, is then loaded into the location addressed by the incremented user stack pointer.

Reset Carry Flag

RCF RCF

Operation C ← 0

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

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

Format

	Bytes	Cycles	Op Code (Hex)
opc	1	4	CF

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

Return

RET

Operation

PC ← @SP

SP ← SP + 2

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

Flags

No flags are affected.

Format

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

Example

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

RET → PC = 101Ah, SP = 00FEh

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

Rotate Left

RL dst

Operation C ← dst(7)

dst(0) ← dst(7)

dst(n + 1) ← dst(n), n = 0 – 6

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

Figure 54 shows how bits rotate left.

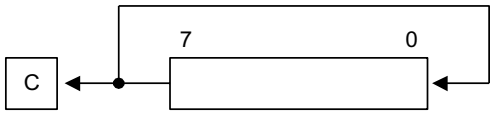


Figure 54. Rotate Left

Flags

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

Format

	Bytes	Cycles	Op Code (Hex)	Address Mode dst
opc dst	2	4	90	R
		4	91	IR

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

```
RL    00h    →    Register 00h = 55h, C = 1
RL    @01h   →    Register 01h = 02h, register 02h = 2Eh, C = 0
```

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

Rotate Left through Carry

RCL dst

Operation dst(0) ← C

C ← dst(7)

dst(n + 1) ← dst(n), n = 0 – 6

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

Figure 55 shows how bits rotate left through carry.

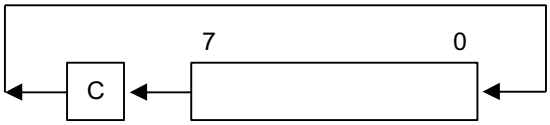


Figure 55. Rotate Left through Carry

Flags

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

Format

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

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

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

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

Rotate Right

RR dst

Operation C ← dst(0)

dst(7) ← dst(0)

dst(n) ← dst(n + 1), n = 0 – 6

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

Figure 56 shows how bits rotate right.

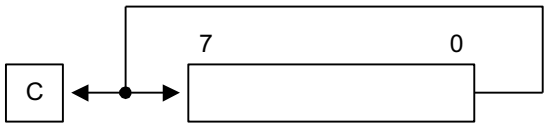


Figure 56. Rotate Right

Flags

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

Format

	Bytes	Cycles	Op Code (Hex)	Address Mode dst
opc dst	2	4	E0	R
		4	E1	IR

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

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

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

Rotate Right through Carry

RRC dst

Operation dst(7) ← C

C ← dst(0)

dst(n) ← dst(n + 1), n = 0 – 6

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

Figure 57 shows how bits rotate right through carry.

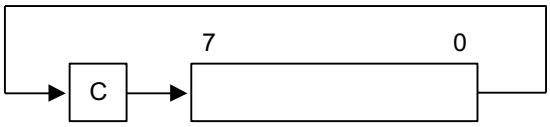


Figure 57. Rotate Right through Carry

Flags

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

Format

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

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

```
RRC   00h     Register 00h = 2Ah, C = 1
RRC   @01h    Register 01h = 02h, register 02h = 0Bh, C = 1
```

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

Select Bank0

SB0

Operation BANK ← 0

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

Flags No flags are affected.

Format

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

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

Select Bank1

SB1

Operation BANK ← 1

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

Flags No flags are affected.

Format

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

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

Subtract with Carry

SBC dst, src

Operation dst ← dst – src – c

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

Flags

C Set if a borrow occurred (src > dst); cleared otherwise.

Z Set if the result is 0; cleared otherwise.

S Set if the result is negative; cleared otherwise.

V Set if arithmetic overflow occurred, i.e., if the operands were of opposite sign and the sign of the result is the same as the sign of the source; cleared otherwise.

D Always set to 1.

H Cleared if there is a carry from the most significant bit of the low-order four bits of the result; set otherwise, indicating a borrow.

Format

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

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

SBC	R1, R2	R1 = 0Ch, R2 = 03h
SBC	R1, @R2	R1 = 05h, R2 = 03h, register 03h = 0Ah
SBC	01h, 02h	Register 01h = 1Ch, register 02h = 03h

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

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

Set Carry Flag

SCF

Operation C ← 1

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

Flags **C** Set to 1.

No other flags are affected.

Format

	Bytes	Cycles	Op Code (Hex)
opc	1	4	DF

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

Shift Right Arithmetic

SRA dst

Operation dst(7) ← dst(7)

C ← dst(0)

dst(n) ← dst(n + 1), n = 0 – 6

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

Figure 58 shows how bits shift right.

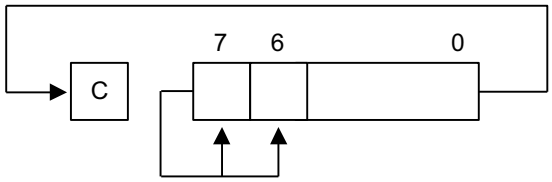


Figure 58. Shift Right

Flags

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

Format

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

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

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

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

Set Register Pointer

SRP src

SRP0 src

SRP1 src

Operation

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

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

Flags No flags are affected.

Format

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

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

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

Stop Operation

STOP

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

Flags No flags are affected.

Format

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

Example The *STOP* statement halts all microcontroller operations.

Subtract

SUB dst, src

Operation dst ← dst – src

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

Flags

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

Format

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

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

SUB	R1, R2	R1 = 0Fh, R2 = 03h
SUB	R1, @R2	R1 = 08h, R2 = 03h
SUB	01h, 02h	Register 01h = 1Eh, register 02h = 03h
SUB	01h, @02h	Register 01h = 17h, register 02h = 03h
SUB	01h, #90h	Register 01h = 91h; C, S, and V = 1
SUB	01h, #65h	Register 01h = 0BCh; C and S = 1, V = 0

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

Swap Nibbles

SRA dst

Operation dst(0-3) ↔ dst(4-7)

The contents of the lower four bits and upper four bits of the destination operand are swapped.

Figure 59 shows how to swap nibbles.

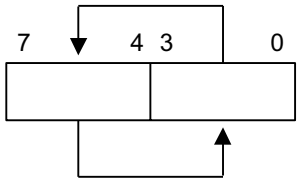


Figure 59. Swap Nibbles

Flags

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

Format

		Bytes	Cycles	Op Code (Hex)	Address Mode dst
opc	dst	2	4	F0	R
			4	F1	IR

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

```
SWAP  00h      Register 00h = 0E3h
SWAP  @02h     Register 02h = 03h, register 03h = 4Ah
```

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

Test Complement under Mask

TCM dst, src

Operation (NOT dst) AND src

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

Flags

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

Format

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

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

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

In the first example, if working register R0 contains the value 0C7h (11000111b) and register R1 the value 02h (00000010b), the *TCM R0, R1* statement tests bit 1 in the destination register for a 1 value. Because the mask value corresponds to the test bit, the Z flag is set to logic 1 and can be tested to determine the result of the TCM operation.

Test Under Mask

TM dst, src

Operation dst AND src

This instruction tests selected bits in the destination operand for a logic 0 value. The bits to be tested are specified by setting a 1 bit in the corresponding position of the source operand (mask), which is ANDed with the destination operand. The zero (Z) flag can then be checked to determine the result. The destination and source operands are unaffected.

Flags

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

Format

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

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

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

statement tests bit 1 in the destination register for a 0 value. Because the mask value does not match the test bit, the Z flag is cleared to logic 0 and can be tested to determine the result of the TM operation.

Wait for Interrupt

WFI

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

Flags No flags are affected.

Format

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

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

Example Figure 60 presents a sample program structure that depicts the sequence of operations that follow a *WFI* statement.

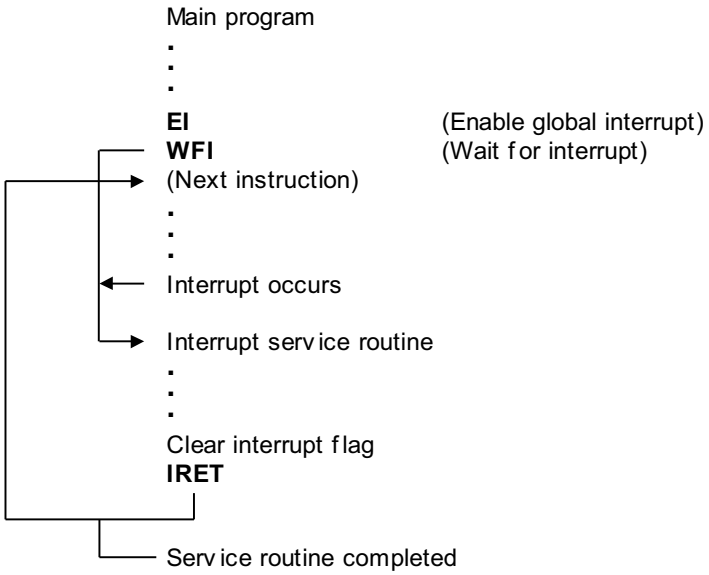


Figure 60. Sample Program Structure

Logical Exclusive OR

XOR dst, src

Operation dst ← dst XOR src

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

Flags

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

Format

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

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

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

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

Chapter 8. Clock Circuit

The S3F8S45 microcontroller features two oscillator circuits: a main clock and a subclock circuit. The CPU and peripheral hardware operate on the system clock frequency supplied through these circuits. The maximum CPU clock frequency of the S3F8S45 MCU is determined by the settings in the CLKCON Register. These main oscillator circuits are shown in Figures 61 through 65.

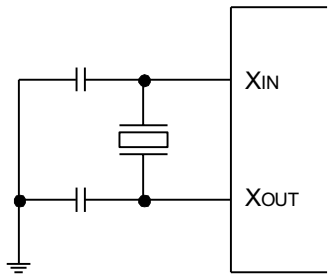


Figure 61. Crystal/Ceramic Oscillator (f_X)

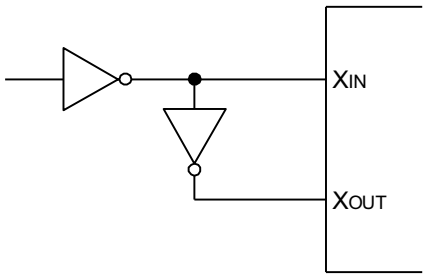


Figure 62. External Oscillator (f_X)

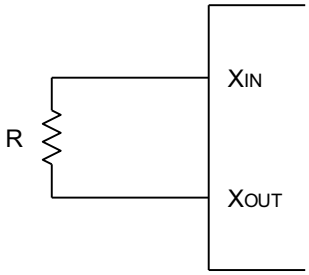


Figure 63. RC Oscillator (f_X)

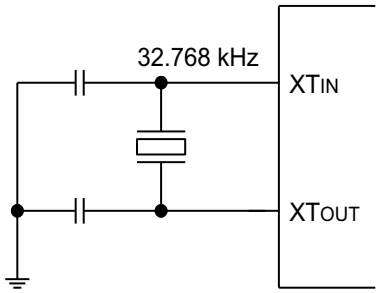


Figure 64. Crystal Oscillator (f_{XT})

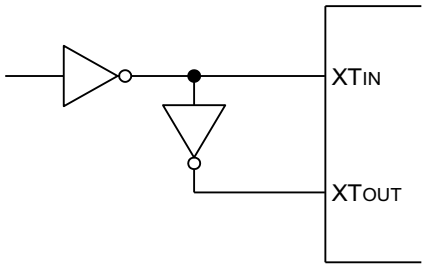


Figure 65. External Oscillator (f_{XT})

8.1. System Clock Circuit

The system clock circuit features the following components:

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

8.2. CPU Clock Notation

In this document, the following notations are used to describe the CPU clock:

- f_X = Main clock

- f_{XT} = Subclock
- f_{XX} = Selected system clock

8.3. Clock Status During Power-Down Modes

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

- In Stop Mode, the main oscillator is halted. Stop Mode is released, and the oscillator is started, by a reset operation or an external interrupt (with an RC delay noise filter), and can also be released by internal interrupt when the subsystem oscillator is running and the watch timer is operating with the subsystem clock.
- In Idle Mode, the internal clock signal is gated to the CPU, but not to the interrupt structure, timers, and timer/counters. Idle Mode is released by a reset or by an external or internal interrupt.

A block diagram of the system clock circuit is shown in Figure 66.

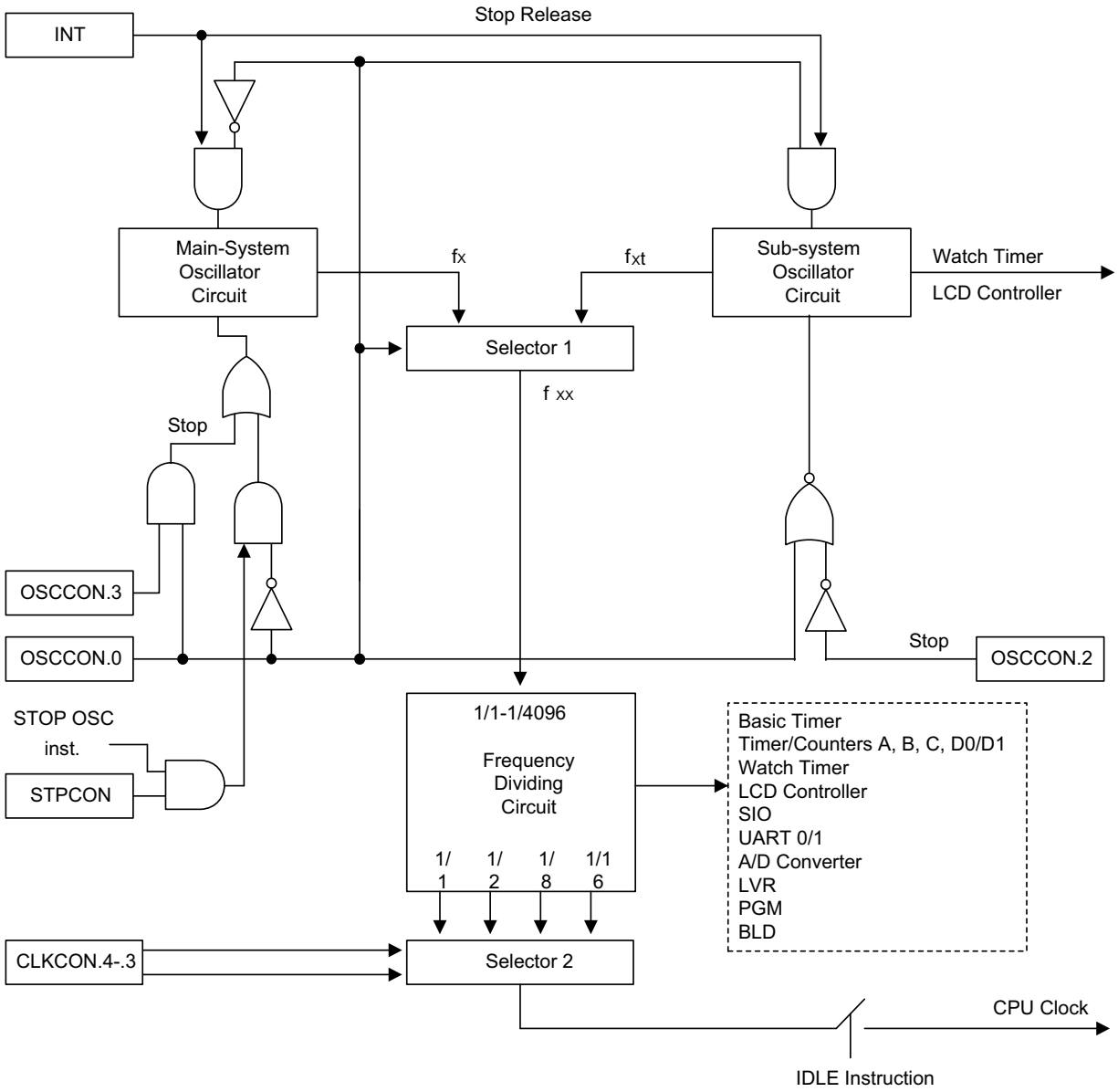


Figure 66. System Clock Circuit Diagram

8.4. System Clock Control Register

The System Clock Control (CLKCON) Register, shown in Table 32, is located at address D4h, Set1. This register is read/write-addressable and offers an oscillator frequency divide-by value.

After the main oscillator is activated, $f_{XX}/16$ (the slowest clock speed) is selected as the CPU clock. If necessary, the CPU clock speed can be increased to $f_{XX}/8$, $f_{XX}/2$, or $f_{XX}/1$.

Table 32. System Clock Control Register (CLKCON; Set1)

Bit	7	6	5	4	3	2	1	0
Reset	–	–	–	0	0	–	–	–
R/W	R/W	–	–	R/W	R/W	–	–	–
Address	D4h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7]	**Oscillator IRQ Wake-up Function Bit** 0: Enable IRQ for main wake-up in Power Down Mode. 1: Disable IRQ for main wake-up in Power Down Mode.
[6:5]	**Reserved**
[4:3]	**CPU Clock (System Clock) Selection Bits*** 00: $f_{XX}/16$. 01: $f_{XX}/8$. 10: $f_{XX}/2$. 11: $f_{XX}/1$.
[2:0]	**Reserved**

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

8.5. Oscillator Control Register

The Oscillator Control (OSCCON) Register, shown in Table 33, is located in Set1, Bank0, at address FAh. It is read/write addressable and offers the following functions:

- System clock selection
- Main oscillator control
- Suboscillator control

Table 33. Oscillator Control Register (OSCCON; Set1, Bank0)

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

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

Bit	Description
[7:4]	**Reserved**
[3]	**Main Oscillator Control Bit** 0: Main oscillator RUN. 1: Main oscillator STOP.
[2]	**Suboscillator Control Bit** 0: Suboscillator RUN. 1: Suboscillator STOP.
[1]	**Reserved**
[0]	**System Clock Selection Bit** 0: Select main oscillator for system clock. 1: Select suboscillator for system clock.

The OSCCON.0 register settings select the main clock or the subclock as the system clock. After a reset, the main clock is selected for the system clock because the reset value of OSCCON.0 is 0.

The main oscillator can be stopped or run by setting OSCCON.3. The suboscillator can be stopped or run by setting OSCCON.2.

8.6. Stop Control Register

The Stop Control (STPCON) Register, shown in Table 34, is located in Set1, Bank0, at address EDh. It is read/write addressable and offers an enable/disable STOP instruction. After a reset, this STOP instruction is disabled, because the value of STPCON is *other values*.

Table 34. Stop Control Register (STPCON; Set1, Bank0)

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

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

Bit	Description
[7]	**Stop Control Bits*** 10100101: Enable stop instruction. All other values: disable stop instruction.

Note: *Before executing a stop instruction, this STPCON Register must be set as 10100101b; otherwise, the stop instruction will not execute, and a reset will be generated.

If necessary, the STOP instruction can be used by setting the value of STPCON to 10100101b, as shown in the following example, which shows how to enter Stop Mode when a main clock is selected as the system clock.

Example

```
LD      STOPCON,#1010010b       ; Enable STOP instruction
STOP                            ; Enter Stop Mode
NOP
NOP
NOP                             ; Release STOP mode
LD      STOPCON,#00000000b      ; Disable STOP instruction
```

8.7. Switching the CPU Clock

Data loading in the Oscillator Control (OSCCON) Register determines whether a main clock or a subclock is selected as the CPU clock, and also how this frequency is to be

divided by setting CLKCON. As a result, it is possible to dynamically switch between main and subclocks and to modify operating frequencies.

OSCCON.0 selects the main clock (f_X) or the subclock (f_{XT}) for the CPU clock. OSCCON .3 starts or stops the main clock oscillation, and OSCCON.2 starts or stops the subclock oscillation. CLKCON.4–.3 controls the frequency divider circuit and divides the selected f_{XX} clock by 1, 2, 8, and 16. If the subclock (f_{XT}) is selected as the system clock, CLKCON. 4–.3 must be set to 11.

As an example, if you are using the default CPU clock (i.e., under normal operating mode and with a main clock of $f_X/16$) and you want to switch from the f_X clock to a subclock and to stop the main clock, you must set CLKCON.4–.3 to 11, OSCCON.0 to 1, and OSCCON.3 to 1 in sequence. As a result, the clock is switched from f_X to f_{XT}, and main clock oscillation is stopped.

The following example presents the steps that must be taken to switch from a subclock to the main clock. First, set OSCCON.3 to 0 to enable main clock oscillation. Next, after a certain number of machine cycles have elapsed, select the main clock by setting OSCCON.0 to 0.

Example 1. The following example shows how to change from the main clock to the subclock.

```
MA2SUB OR   CLKCON,#18h     ; Nondivided clock for system clock
LD          OSCCON,#01h     ; Switches to the subclock
CALL        DLY16           ; Delay 16 ms
OR          OSCCON,#08h     ; Stop the main clock oscillation RET
```

Example 2. This next example shows how to change from the subclock to the main clock.

```
SUB2MA      AND OSCCON,#07h ; Start the main clock oscillation
CALL        DLY16           ; Delay 16 ms
AND         OSCCON,#06h     ; Switch to the main clock
RET
DLY16       SRP #0C0h
LD          R0,#20h
DEL         NOP
DJNZ        R0,DEL
RET
```

Chapter 9. Reset and Power-Down

This chapter discusses system reset and power-down modes.

9.1. System Reset

During a power-on reset, the voltage at V_{DD} goes High and the nRESET pin is forced Low. The nRESET signal is input through a Schmitt trigger circuit, where it is synchronized with the CPU clock. This procedure brings the S3F8S45 MCU into a known operating status.

To allow time for internal CPU clock oscillation to stabilize, the nRESET pin must be held Low for a minimum time interval after the power supply comes within tolerance. The minimum required time of a reset operation for oscillation stabilization is one millisecond.

Whenever a reset occurs during normal operation (i.e., when both V_{DD} and nRESET are High), the nRESET pin is forced Low, and the reset operation starts. All system and peripheral control registers are then reset to their default hardware values. In summary, the following sequence of events occurs during a reset operation:

- All interrupts are disabled.
- The watchdog (basic timer) function is enabled.
- Ports 0–4 are set to Input Mode, and all pull-up resistors are disabled for the I/O port.
- Peripheral control and data register settings are disabled and reset to their default hardware values.
- The program counter (PC) is loaded with the `0100h` program reset address within ROM.
- When the programmed oscillation stabilization time interval has elapsed, the instruction stored at ROM location `0100h` (and at `0101h`) is fetched and executed in Normal Mode by the Smart Option.
- The reset address in ROM can be changed by the Smart Option in the S3F8S45 full-flash device. Refer to the Embedded Flash Memory Interface chapter on page 327 to learn more.

9.2. Normal Mode Reset Operation

In Normal Mode, the Test pin is tied to V_{SS}. A reset enables access to the 16KB on-chip ROM; the external interface is not automatically configured.

Note: To program the duration of the oscillation stabilization interval, make the appropriate settings to the Basic Timer Control (BTCON) Register before entering Stop Mode. Additionally, if not using the basic timer watchdog function (which causes a system reset if a basic timer counter overflow occurs), it can be disabled by writing `1010b` to the upper nibble of BTCON.

9.3. Hardware Reset Values

Tables 35 through 38 list the reset values for the CPU and system registers, the peripheral control registers, and the peripheral data registers following a reset operation. The following notation is used to represent these reset values:

- A 1 or a 0 shows the reset bit value as logic 1 or logic 0, respectively
- An x means that the bit value is undefined after a reset
- A dash (–) means that the bit is either not used or not mapped (however, a 0 is read from the bit position)

Table 35. Set1 Register Values After RESET

		Address		Bit Values After Reset							
Register Name	**Mnemonic**	**Dec**	**Hex**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Locations D0h–D2h are not mapped.											
Basic Timer Control	BTCON	211	D3h	0	0	0	0	0	0	0	0
System Clock Control	CLKCON	212	D4h	0	–	–	0	0	–	–	–
System Flags	FLAGS	213	D5h	x	x	x	x	x	x	0	0
Register Pointer 0	RP0	214	D6h	1	1	0	0	0	–	–	–
Register Pointer 1	RP1	215	D7h	1	1	0	0	1	–	–	–
Stack Pointer High Byte	SPH	216	D8h	x	x	x	x	x	x	x	x
Stack Pointer Low Byte	SPL	217	D9h	x	x	x	x	x	x	x	x
Instruction Pointer High Byte	IPH	218	DAh	x	x	x	x	x	x	x	x
Instruction Pointer Low Byte	IPL	219	DBh	x	x	x	x	x	x	x	x
Interrupt Request	IRQ	220	DCh	0	0	0	0	0	0	0	0
Interrupt Mask	IMR	221	DDh	x	x	x	x	x	x	x	x

Notes:
1. An *x* means that the bit value is undefined following reset.
2. A dash (–) means that the bit is neither used nor mapped, but the bit is read as 0.

Table 35. Set1 Register Values After RESET (Continued)

		Address		Bit Values After Reset							
Register Name	**Mnemonic**	**Dec**	**Hex**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
System Mode	SYM	222	DEh	0	–	–	x	x	x	0	0
Register Page	PP	223	DFh	0	0	0	0	0	0	0	0

Notes:
1. An *x* means that the bit value is undefined following reset.
2. A dash (–) means that the bit is neither used nor mapped, but the bit is read as 0.

Table 36. Set1, Bank0 Register Values After Reset

		Address		Bit Values After Reset							
Register Name	**Mnemonic**	**Dec**	**Hex**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
A/D Converter Data High Byte	ADDATAH	208	D0h	x	x	x	x	x	x	x	x
A/D Converter Data Low Byte	ADDATAL	209	D1h	–	–	–	–	–	–	x	x
A/D Converter Control	ADCON	210	D2h	–	0	0	0	0	0	0	0
Timer A Counter	TACNT	224	E0h	0	0	0	0	0	0	0	0
Timer A Data	TADATA	225	E1h	1	1	1	1	1	1	1	1
Timer A Control	TACON	226	E2h	0	0	0	0	0	0	0	0
Timer B Control	TBCON	227	E3h	–	–	0	0	0	0	0	0
Timer B Data High Byte	TBDATAH	228	E4h	1	1	1	1	1	1	1	1
Timer B Data Low Byte	TBDATAL	229	E5h	1	1	1	1	1	1	1	1
Watch Timer Control	WTCON	230	E6h	0	0	0	0	0	0	0	0
SIO Control	SIOCON	231	E7h	0	0	0	0	0	0	0	0
SIO Data	SIODATA	232	E8h	0	0	0	0	0	0	0	0
SIO Prescaler	SIOPS	233	E9h	0	0	0	0	0	0	0	0
Timer C Counter	TCCNT	234	EAh	0	0	0	0	0	0	0	0
Timer C Data	TCDATA	235	EBh	1	1	1	1	1	1	1	1
Timer C Control	TCCON	236	ECh	0	0	0	0	0	0	0	0
Stop Control	STPCON	237	EDh	0	0	0	0	0	0	0	0
UART0 Control High Byte	UART0CONH	238	EEh	0	0	0	0	0	0	0	0
UART0 Control Low Byte	UART0CONL	239	EFh	0	0	0	0	0	0	0	0
UART0 Data	UDATA0	240	F0h	x	x	x	x	x	x	x	x
UART0 Baud Rate Data	BRDATA0	241	F1h	1	1	1	1	1	1	1	1
UART1 Control High Byte	UART1CONH	242	F2h	0	0	0	0	0	0	0	0
UART1 Control Low Byte	UART1CONL	243	F3h	0	0	0	0	0	0	0	0

Table 36. Set1, Bank0 Register Values After Reset (Continued)

		Address		Bit Values After Reset							
Register Name	**Mnemonic**	**Dec**	**Hex**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UART1 Data	UDATA1	244	F4h	x	x	x	x	x	x	x	x
UART1 Baud Rate Data	BRDATA1	245	F5h	1	1	1	1	1	1	1	1
Flash Memory Sector Address High Byte	FMSECH	246	F6h	0	0	0	0	0	0	0	0
Flash Memory Sector Address Low Byte	FMSECL	247	F7h	0	0	0	0	0	0	0	0
Flash Memory User Programming Enable	FMUSR	248	F8h	0	0	0	0	0	0	0	0
Flash Memory Control	FMCON	249	F9h	0	0	0	0	0	–	–	0
Oscillator Control	OSCCON	250	FAh	–	–	–	–	0	0	–	0
Interrupt pending	INTPND	251	FBh	–	–	0	0	0	0	0	0
Basic Timer Counter	BTCNT	253	FDh	0	0	0	0	0	0	0	0
Location FEh is not mapped.											
Interrupt Priority	IPR	255	FFh	x	x	x	x	x	x	x	x

Table 37. Set1, Bank1 Register and Values After RESET

		Address		Bit Values After Reset							
Register Name	**Mnemonic**	**Dec**	**Hex**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Port 0 Control High Byte	P0CONH	208	D0h	0	0	0	0	0	0	0	0
Port 0 Control Low Byte	P0CONL	209	D1h	0	0	–	0	0	0	0	0
Port 0 Pull-up Resistor Enable	P0PUR	210	D2h	0	0	0	0	0	0	0	0
Port 1 Control High Byte	P1CONH	224	E0h	0	0	0	0	0	0	0	0
Port 1 Control Low Byte	P1CONL	225	E1h	0	0	0	0	0	0	0	0
Port 1 Pull-up Resistor Enable	P1PUR	226	E2h	0	0	0	0	0	0	0	0
Port 1 n-Channel Open-Drain Mode	PNE1	227	E3h	0	0	0	0	0	0	0	0
Port 2 Control High Byte	P2CONH	228	E4h	0	0	0	0	0	0	0	0
Port 2 Control Low Byte	P2CONL	229	E5h	0	0	0	0	0	0	0	0
Port 2 Pull-Up Resistor Enable	P2PUR	230	E6h	0	0	0	0	0	0	0	0
Port 0/3 Interrupt Pending	P03PND	231	E7h	0	0	0	0	0	0	0	0

Notes:
1. An *x* means that the bit value is undefined following reset.
2. A dash (–) means that the bit is neither used nor mapped, but the bit is read as 0.

Table 37. Set1, Bank1 Register and Values After RESET (Continued)

		Address		Bit Values After Reset							
Register Name	**Mnemonic**	**Dec**	**Hex**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Port 0/3 Interrupt Control High Byte	P03INTH	232	E8h	0	0	0	0	0	0	0	0
Port 0/3 Interrupt Control Low Byte	P03INTL	233	E9h	0	0	0	0	0	0	0	0
Port 3 Control High Byte	P3CONH	234	EAh	–	–	0	0	0	0	0	0
Port 3 Control Low Byte	P3CONL	235	EBh	0	0	0	0	0	0	0	0
Port 3 Pull-Up Resistor Enable	P3PUR	236	ECh	–	0	0	0	0	0	0	0
Port 3 n-Channel Open-Drain Mode	PNE3	237	EDh	–	–	0	0	0	0	0	0
Port 4 Control High Byte	P4CONH	238	EEh	–	–	–	0	0	0	0	0
Port 4 Control Low Byte	P4CONL	239	EFh	0	0	0	0	0	0	0	0
Port 0 Data	P0	240	F0h	0	0	0	0	0	0	0	0
Port 1 Data	P1	241	F1h	–	–	–	–	0	0	0	0
Port 2 Data	P2	242	F2h	0	0	0	0	0	0	0	0
Port 3 Data	P3	243	F3h	0	0	0	0	0	0	0	0
Port 4 Data	P4	244	F4h	0	0	0	0	0	0	0	0
Port 4 Pull-up Resistor Enable	P4PUR	245	F5h	0	0	0	0	0	–	–	0
Port 4 Interrupt Control	P4INT	246	F6h	0	0	0	0	0	0	0	0
Port 4 Interrupt Pending	P4PND	247	F7h	0	0	0	0	0	0	0	0
Pattern Generation Control	PGCON	248	F8h	1	1	1	1	1	1	1	1
Pattern Generation Data	PGDATA	249	F9h	1	1	1	1	1	1	1	1
LCD Control	LCON	250	FAh	0	0	0	0	0	0	0	0
PWM Control	PWMCON	251	FBh	0	0	0	0	0	0	0	0
PWM Data High Byte	PWMDATAH	252	FCh	0	0	0	0	0	0	0	0
PWM Data Low Byte	PWMDATAL	253	FDh	0	0	0	0	0	0	0	0
Watchdog Timer Control	WDTCON	254	FEh	1	1	1	1	1	1	1	1
Locations FFh is not mapped.											

Notes:
1. An *x* means that the bit value is undefined following reset.
2. A dash (–) means that the bit is neither used nor mapped, but the bit is read as 0.

Table 38. Page 4 Register Values After RESET

Register Name	Mnemonic	Address		Bit Values After Reset							
		Dec	Hex	7	6	5	4	3	2	1	0
Reset Source Indicating	RESETID	0		See Table 40 on page 200							
Timer D0 Control	TD0CON	1	01h	0	0	0	0	0	0	0	0
Timer D0 Counter High Byte	TD0CNTH	2	02h	0	0	0	0	0	0	0	0
Timer D0 Counter Low Byte	TD0CNTL	3	03h	0	0	0	0	0	0	0	0
Timer D0 Data High Byte	TD0DATAH	4	04h	1	1	1	1	1	1	1	1
Timer D0 Data Low Byte	TD0DATAL	5	05h	1	1	1	1	1	1	1	1
Timer D1 Counter High Byte	TD1CNTH	6	06h	0	0	0	0	0	0	0	0
Timer D1 Counter Low Byte	TD1CNTL	7	07h	0	0	0	0	0	0	0	0
Timer D1 Data High Byte	TD1DATAH	8	08h	1	1	1	1	1	1	1	1
Timer D1 Data Low Byte	TD1DATAL	9	09h	1	1	1	1	1	1	1	1
Timer D1 Control	TD1CON	10	0Ah	0	0	0	0	0	0	0	0

9.4. Reset Source Indicating Register

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

Table 39. Reset Source Indicating Register (RESETID; Page 4)

Bit	7	6	5	4	3	2	1	0
R/W	–	–	–	R/W	–	R/W	R/W	R/W
Address	00h							
Mode	Register Addressing Mode only							

Note: R = read only; R/W = read/write; a dash (–) means that the bit is neither used nor mapped, but the bit is read as 0.

Bit	Description
[7:5]	**Reserved**
[4]	**nRESET Pin Indicating Bit** 0: Reset is not generated by nRESET pin when read; cleared to =0 (when write). 1: Reset is generated by nRESET pin when read; no effect (when write).
[3]	**Reserved**
[2]	**WDT Reset Indicating Bit** 0: Reset is not generated by WDT (when read); cleared to 0 (when write). 1: Reset is generated by WDT (when read); no effect (when write).

Bit	Description (Continued)
[1]	**LVD Reset Indicating Bit** 0: Reset is not generated by LVR (when read); cleared to 0 (when write). 1: Reset is generated by LVR (when read); no effect (when write).
[0]	**POR Reset Indicating Bit** 0: Reset is not generated by POR (when read); cleared to 0 (when write). 1: Reset is generated by POR (when read); no effect (when write).

Table 40. State of RESETID

Bit	7	6	5	4	3	2	1	0
POR	–	–	–	0	–	0	2	1
LVD	–	–	–	0	–	0	1	3
WDT, nReset	–	–	–	4	–	4	3	3

Notes:
1. To clear an indicating register, write a 0 to the indicating flag bit. Writing a 1 to a reset-indicating flag (RESETID.1–.2 and .4) has no effect.
2. RESETID.1 will be set to a logic 1 if LVR is enabled by the Smart Option; otherwise 0.
3. Not affected by any other reset.
4. Bits corresponding to sources that are active at the time of reset will be set.

9.5. Power-Down Modes

This section describes the following power-down modes:

- Stop Mode
- Idle Mode

9.6. Stop Mode

Stop Mode is invoked by executing the STOP instruction after setting the Stop Control (STOPCON) Register. In Stop Mode, the operation of the CPU and all peripherals is halted. Essentially, the on-chip main oscillator stops and the current consumption can be reduced. All system functions stop when the clock freezes, but data stored in the internal register file is retained. However, the status of internal ring oscillator (ICLK, 15kHz) is configurable. Stop Mode can be released in one of two ways: by a system reset or by an external interrupt. After releasing from Stop Mode, the value of STOPCON is cleared automatically.

Note: Do not use Stop Mode if you are using an external clock source, because the X_{IN} or X_{TIN} inputs must be restricted internally to V_{SS} to reduce current leakage.

Using nRESET to Release Stop Mode

Stop Mode is released when the nRESET signal is released and returns to a high level; all system and peripheral control registers are reset to their default hardware values and the contents of all data registers are retained. A reset operation automatically selects a slow clock (e.g., f_{XX}/16) because CLKCON.3 and CLKCON.4 are cleared to `00b`. After the programmed oscillation stabilization interval has elapsed, the CPU starts the system initialization routine by fetching the program instruction stored in ROM location `0100h` (and `0101h`).

Using an External Interrupt to Release Stop Mode

External interrupts with an RC-delay noise filter circuit can be used to release Stop Mode. Which interrupt you can use to release Stop Mode in a given situation depends on the microcontroller's current internal operating mode. The external interrupts in the S3F8S45 MCU's interrupt structure that can be used to release Stop Mode are P3.0–P3.6/P0.3 (INT0─VINT7).

Please note the following conditions for Stop Mode release:

- If you release Stop Mode using an external interrupt, the current values in system and peripheral control registers are unchanged except STPCON register.
- If you use an internal or external interrupt for Stop Mode release, you can also program the duration of the oscillation stabilization interval by making the appropriate control and clock settings before entering Stop Mode.
- When Stop Mode is released by external interrupt, the CLKCON.4 and CLKCON.3 bit-pair setting remains unchanged and the currently selected clock value is used.
- The external interrupt is serviced when the Stop Mode release occurs. Following the IRET from the service routine, the instruction immediately following the one that initiated Stop Mode is executed.

Using an Internal Interrupt to Release Stop Mode

Activate any enabled interrupt to release Stop Mode; all other functions remain the same as if using an external interrupt.

How to Enter Stop Mode

The following example shows how to handle the STPCON Register then writing a STOP instruction, in the sequential order shown.

```
LD STPCON,#10100101b
STOP
NOP
NOP
NOP
```

9.7. Idle Mode

Idle Mode is invoked by the IDLE instruction (op code 6Fh). In Idle Mode, CPU operations are halted while some peripherals remain active. During Idle Mode, the internal clock signal is gated away from the CPU, but all peripheral timers remain active. Port pins retain the mode (input or output) they are operating in at the time Idle Mode is entered.

The following two methods can be used 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 the slow clock (f_{XX}/16) because CLKCON.4 and CLKCON.3 are cleared to 00b. If interrupts are masked, a reset is the only way to release Idle Mode.

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

Chapter 10. I/O Ports

The S3F8S45 microcontroller features a 44-pin QFP package type with a total of 38 I/O pins. Among these are five bit-programmable I/O ports, P0–P3 and P4. Four ports, P0–P2 and P4, are 8-bit ports; P3 is a 6-bit port.

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

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

Table 41 provides a general overview of the S3F8S45 MCU's I/O port functions.

Table 41. S3F8S45 Port Configuration Overview (44-Pin QFP)

Port	Configuration Options
Port 0	8-bit general-purpose I/O port; Input or push-pull output; external interrupt input on falling edges, rising edges, or both edges; all P0 pin circuits have noise filters and Interrupt Enable/ Disable Register (P0INT) and the Pending Control Register (P0PND); pull-up resistors can be assigned to individual P0 pins using the P0PUR Register settings. This port is dedicated for key input in IR controller applications.
Port 1	8-bit general-purpose I/O port; input with or without pull-ups, open-drain output, or push-pull output. This port is dedicated for key output in IR controller applications.
Port 2	8-bit general-purpose I/O port; input, open-drain output, or push-pull output. The P2 pins, P2.0- P2.7, can be used as external interrupt inputs and have noise filters. The P2INT Register is used to enable/disable interrupts and P2PND bits can be polled by software for interrupt pending control. Pull-up resistors can be assigned to individual P2 pins using P2PUR register settings. Additionally, P2.4 to P2.7 can be used for SPI function.
P3.0–P3.1	P3.0 is configured input functions (Input Mode, with or without pull-up, for normal input or T0CAP) or output functions (push-pull or Open-Drain Output Mode, for normal output or T0PWM). P3.1 is configured input functions (Input Mode, with or without pull-up, for normal input) or output functions (push-pull or Open-Drain Output Mode, for normal output or REM function). P3.1 is dedicated for IR drive pin and P3.0 can be used for indicator LED drive.
P3.2–P3.3	P3.2 is configured only input pin with pull-up resistor (for normal input or T0CK function). P3.3 is configured only input pin with pull-up resistor (for normal input, T1CAP function, or T2CAP function). P3.3 can be used for IR signal capture pin with T1CAP function or T2CAP function.
P3.4–P3.5	2-bit general-purpose I/O port; Input without or with pull-up, open-drain output, or push-pull output.
P3.7	P3.7 is not configured for I/O pin and it only used to control carrier signal on/off.
Port 4	8-bit general-purpose I/O port; Input without or with pull-up, open-drain output, or push-pull output. This port is dedicated for key output in IR controller application.

10.1. Port Data Registers

Table 42 provides an overview of the register locations of all four S3F8S45 I/O port data registers. Data registers for ports 0, 1, 2, 3, and 4 feature the general format shown in Figure 67.

Table 42. Port Data Register Summary

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

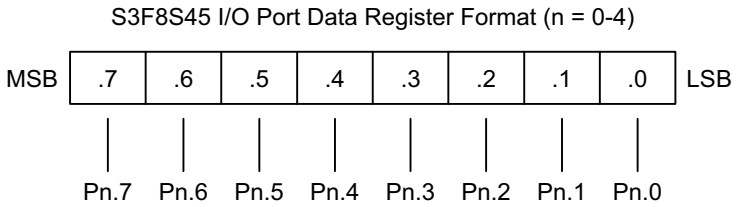


Figure 67. S3F8S45 I/O Port Data Register Format

10.2. Port 0

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

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

Port 0 Control Registers

Port 0 provides two 8-bit control registers, P0CONH for P0.4–P0.7 and P0CONL for P0.0–P0.3 (see Tables 43 and 44). A reset clears these P0CONH and P0CONL registers to 00h, configuring all pins to Input Mode. When P0.3 is in Input Mode, the following three selections are available:

- Schmitt-trigger input with interrupt generation on falling signal edges
- Schmitt-trigger input with interrupt generation on rising signal edges
- Schmitt-trigger input with interrupt generation on falling/rising signal edges

Table 43. Port 0 Control High Byte Register (P0CONH; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	D0h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:6]	**P0.7/COM4/AD4 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode; push-pull. 10: Alternative function (AD4). 11: Alternative function (COM4).
[5:4]	**P0.26/COM5/AD5 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode; push-pull. 10: Alternative function (AD5). 11: Alternative function (COM5).
[3:2]	**P0.5/COM6/AD6 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode; push-pull. 10: Alternative function (AD6). 11: Alternative function (COM6).
[1:0]	**P0.4/COM7/AD7 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode; push-pull. 10: Alternative function (AD7). 11: Alternative function (COM7).

Table 44. Port 0 Control Register Low Byte (P0CONL; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	D1h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:6]	**P0.3/INT7/AD8 Configuration Bits** 00: Schmitt Trigger Input Mode (INT7). 01: Output Mode; push-pull. 10: Alternative function (AD8). 11: Not available.
[5]	**Reserved**
[4]	**P0.2/nRESET Configuration Bit** 00: Schmitt Trigger Input Mode. 01: Output Mode; push-pull.
[3:2]	**P0.1/COM1/AD1 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode; push-pull. 10: Alternative function (X_{TOUT}). 11: Not available.
[1:0]	**P0.0/COM0/AD0 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode; push-pull. 10: Alternative function (X_{TIN}). 11: Not available.

Note: Refer to the Smart Option for configuring as P0.2/nRESET.

These control registers settings can be used to select Input Mode (with or without pull-ups) or to select Push-Pull Output Mode and enable the alternative functions.

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

Port 0 Pull-Up Resistor Enable Register

When using the Port 0 Pull-Up Resistor Enable (P0PUR) Register (D2h, Set1, Bank1; see Table 45), the pull-up resistors can be configured to individual Port 0 pins.

Table 45. Port 0 Pull-Up Resistor Enable Register (P0PUR; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	D2h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7]	**P0.7 Pull-up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[6]	**P0.6 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[5]	**P0.5 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[4]	**P0.4 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[3]	**P0.3 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[2]	**P0.2 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[1]	**P0.1 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[0]	**P0.0 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.

Note: A pull-up resistor of port 0 is automatically disabled only when the corresponding pin is selected as push-pull output or alternative function.

10.3. Port 1

Port 1 is an 8-bit I/O port with individually-configurable pins. Port 1 pins are accessed directly by writing or reading the Port 1 Data (P1) Register at location F1h in Set1, Bank1. P1.0–P1.7 can serve as inputs (with or without pull-ups) and outputs (push-pull or open-drain), can be configured to the following alternative functions:

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

Port 1 Control Registers

Port 1 has two 8-bit control registers: P1CONH for P1.4–P1.7 and P1CONL for P1.0–P1.3 (see Tables 46 and 47). A reset clears these P1CONH and P1CONL registers to 00h, configuring all pins to Input Mode. The control registers settings can be used to select Input Mode (with or without pull-ups) or Output Mode and enable the alternative functions.

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

Table 46. Port 1 Control Register High Byte (P1CONH; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	E0h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:6]	**P1.7/COM0 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode. 10: Not available. 11: Alternative function (COM0).
[5:4]	**P1.6/COM1 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode. 10: Not available. 11: Alternative function (COM1).

Bit	Description (Continued)
[3:2]	**P1.5/COM2 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode. 10: Not available. 11: Alternative function (COM2).
[1:0]	**P1.4/COM3/TCOUT/TCPWM Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode. 10: Alternative function (TCOUT/TCPWM). 11: Alternative function (COM3).

Table 47. Port 1 Control Register Low Byte (P1CONL; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	E1h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:6]	**P1.3/AD0/TXD1 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode. 10: Alternative function (AD0). 11: Alternative function (TXD1).
[5:4]	**P1.2/AD1/RXD1 Configuration Bits** 00: Schmitt Trigger Input Mode (RXD1 In). 01: Output Mode. 10: Alternative function (AD1). 11: Alternative function (RXD1 Out).
[3:2]	**P1.1/AD2/BUZ Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode. 10: Alternative function (AD2). 11: Alternative function (BUZ).
[1:0]	**P1.0/AD3/TBPWM Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode. 10: Alternative function (AD3). 11: Alternative function (TBPWM).

Port 1 Pull-Up Resistor Enable Register

When using the Port 1 Pull-Up Resistor Enable (P1PUR) Register (E2h, Set1, Bank1; see Table 48), the pull-up resistors can be configured to individual Port 1 pins.

Table 48. Port 1 Output Pull-Up Resistor Enable Register (P1PUR; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	E2h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7]	**P1.7 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[6]	**P1.6 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[5]	**P1.5 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[4]	**P1.4 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[3]	**P1.3 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[2]	**P1.2 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[1]	**P1.1 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[0]	**P1.0 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.

Note: Port 1 pull-up resistors are automatically disabled only when the corresponding pin is selected as a push-pull output or as an alternative function.

Port 1 n-Channel Open Drain Mode Register

When using the Port 1 n-Channel Open Drain Mode (PNE1) Register (E3h, Set1, Bank1; see Table 49), the Push-Pull or Open Drain Output Mode can be configured to individual Port 1 pins.

Table 49. Port 1 n-Channel Open Drain Mode Register (PNE1; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	E3h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7]	**P1.7 n-Channel Open Drain Mode Bit** 0: Output Mode, push-pull. 1: Output Mode, open-drain.
[6]	**P1.6 n-Channel Open Drain Mode Bit** 0: Output Mode, push-pull. 1: Output Mode, open-drain.
[5]	**P1.5 n-Channel Open Drain Mode Bit** 0: Output Mode, push-pull. 1: Output Mode, open-drain.
[4]	**P1.4 n-Channel Open Drain Mode Bit** 0: Output Mode, push-pull. 1: Output Mode, open-drain.
[3]	**P1.3 n-Channel Open Drain Mode Bit** 0: Output Mode, push-pull. 1: Output Mode, open-drain.
[2]	**P1.2 n-Channel Open Drain Mode Bit** 0: Output Mode, push-pull. 1: Output Mode, open-drain.
[1]	**P1.1 n-Channel Open Drain Mode Bit** 0: Output Mode, push-pull. 1: Output Mode, open-drain.
[0]	**P1.0 n-Channel Open Drain Mode Bit** 0: Output Mode, push-pull. 1: Output Mode, open-drain.

Note: Port 1 pull-up resistors are automatically disabled only when the corresponding pin is selected as a push-pull output or as an alternative function.

10.4. Port 2

Port 2 is an 8-bit I/O port with individually-configurable pins which are accessed directly by writing or reading the Port 2 Data (P2) Register at location F2h in Set1, Bank1. P2.0–P2.7 can serve as inputs (with or without pull-ups) and push-pull outputs, or can be configured to the following alternative functions:

- Low-byte pins (P2.0–P2.3): PG0/SEG3, PG1/SEG4, PG2/SEG5, PG3/SEG6
- High-byte pins (P2.4–P2.7): PG4/SEG7, PG5/SEG8, PG6/SEG9, PG7/SEG10

Port 2 Control Registers

Port 2 provides two 8-bit control registers, P2CONH for P2.4–P2.7 and P2CONL for P2.0–P2.3 (see Tables 50 and 51). A reset clears these P2CONH and P2CONL registers to 00h, configuring all pins to Input Mode. Use control register settings to select Input Mode (with or without pull-ups) or to select Push-Pull Output Mode and enable the alternative functions.

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

Table 50. Port 2 Control Register High Byte (P2CONH; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	E4h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:6]	**P2.7/PG7/SEG10 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode, push-pull. 10: Alternative function (PG7). 11: Alternative function (SEG10).
[5:4]	**P2.6/PG6/SEG9 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode, push-pull. 10: Alternative function (PG6). 11: Alternative function (SEG9).

Bit	Description (Continued)
[3:2]	**P2.5/PG5/SEG8 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode, push-pull. 10: Alternative function (PG5). 11: Alternative function (SEG8).
[1:0]	**P2.4/PG4/SEG7 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode, push-pull. 10: Alternative function (PG4). 11: Alternative function (SEG7).

Table 51. Port 2 Control Register Low Byte (P2CONL; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	E5h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:6]	**P2.3/PG3/SEG6 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode, push-pull. 10: Alternative function (PG3). 11: Alternative function (SEG6).
[5:4]	**P2.2/PG2/SEG5 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode, push-pull. 10: Alternative function (PG2). 11: Alternative function (SEG5).
[3:2]	**P2.1/PG1/SEG4 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode, push-pull. 10: Alternative function (PG1). 11: Alternative function (SEG4).
[1:0]	**P2.0/PG0/SEG3 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode, push-pull. 10: Alternative function (PG0). 11: Alternative function (SEG3).

Port 2 Pull-Up Resistor Enable Register

When using the Port 2 Pull-Up Resistor Enable (P2PUR) Register (E6h, Set1, Bank1; see Table 52), the pull-up resistors can be configured to individual Port 2 pins.

Table 52. Port 2 Pull-Up Resistor Enable Register (P2PUR; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	E6h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7]	**P2.7 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[6]	**P2.6 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[5]	**P2.5 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[4]	**P2.4 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[3]	**P2.3 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[2]	**P2.2 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[1]	**P2.1 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[0]	**P2.0 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.

Note: Port 2 pull-up resistors are automatically disabled only when the corresponding pin is selected as a push-pull output or as an alternative function.

10.5. Port 3

Port 3 is a 7-bit I/O port with individually-configurable pins which are accessed directly by writing or reading the Port 3 Data (P3) Register at location F3h in Set1, Bank1. P3.0–P3.6 can serve as inputs (with or without pull-ups) and outputs (push pull or open-drain), or you can configure the following alternative functions:

- Low-byte pins (P3.0–P3.3): INT4/SEG15, INT5/RXD0/SEG16, INT6/TXD0/SEG17, INT0/SEG18
- High-byte pins (P3.4–P3.6): INT1/TAOUT/TAPWM, INT2/TACLK/SEG20, INT3/TACAP/SEG21

Port 3 Control Registers

Port 3 provides two 8-bit control registers, P3CONH for P3.4–P3.6 and P3CONL for P3.0–P3.3 (see Tables 53 and 54). A reset clears the P3CONH and P3CONL registers to 00h, configuring all pins to Input Mode. When P3.0–P3.6 are in Input Mode, the following three selections are available:

- Schmitt-trigger input with interrupt generation on falling signal edges
- Schmitt-trigger input with interrupt generation on rising signal edges
- Schmitt-trigger input with interrupt generation on falling/rising signal edges

Table 53. Port 3 Control High Byte Register (P3CONH; Set1, Bank1)

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

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

Bit	Description
[7:6]	**Reserved**
[5:4]	**P3.6/INT3/TACAP/SEG21 Configuration Bits** 00: Schmitt Trigger Input Mode (INT3, TACAP). 01: Output Mode. 10: Not available. 11: Alternative function (SEG21).

Bit	Description (Continued)
[3:2]	**P3.5/INT2/TACLK/SEG20 Configuration Bits** 00: Schmitt Trigger Input Mode (INT2, TACLK). 01: Output Mode. 10: Not available. 11: Alternative function (SEG20).
[1:0]	**P3.4/INT1/TAOUT/TAPWM/SEG19 Configuration Bits** 00: Schmitt Trigger Input Mode (INT1). 01: Output Mode. 10: Alternative function (TAOUT/TAPWM). 11: Alternative function (SEG19).

Table 54. Port 3 Control Low Byte Register (P3CONL; Set1, Bank1)

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

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

Bit	Description
[7:6]	**P3.3/INT0/SEG18 Configuration Bits** 00: Schmitt Trigger Input Mode (INT0). 01: Output Mode. 10: Not available. 11: Alternative function (SEG18).
[5:4]	**P3.2/INT6/TXD0/SEG17 Configuration Bits** 00: Schmitt Trigger Input Mode (INT6). 01: Output Mode. 10: Alternative function (TXD0). 11: Alternative function (SEG17).
[3:2]	**P3.1/INT5/RXD0/SEG16 Configuration Bits** 00: Schmitt Trigger Input Mode (INT5, RXD In). 01: Output Mode. 10: Alternative function (RXD0 Out). 11: Alternative function (SEG16).
[1:0]	**P3.0/INT4/SEG15 Configuration Bits** 00: Schmitt Trigger Input Mode (INT4). 01: Output Mode. 10: Not available. 11: Alternative function (SEG15).

These control register's settings can be used to select Input Mode (with or without pull-ups) or Output Mode and enable the alternative functions.

When programming the port, note that any alternative peripheral I/O function you configure using the Port 3 Control registers must also be enabled in the associated peripheral module.

Port 0/3 Interrupt Enable and Pending Registers

To process external interrupts at the Port 0/3 pins, the following additional control registers are provided:

- Port 0/3 Interrupt Enable High (P03INTH) Register (E8h, Set1, Bank1)
- Port 0/3 Interrupt Enable Low (P03INTL) Register (E9h, Set1, Bank1)
- Port 0/3 Interrupt Pending (P03PND) Register (E7h, Set1, Bank1)

The Port 0/3 Interrupt Pending (P03PND) Register (see Table 55) lets you check for interrupt pending conditions and clear the pending condition when the interrupt service routine has been initiated. The application software detects interrupt requests by polling this P03PND Register at regular intervals.

When the interrupt enable bit of any Port 0/3 pin is 1; a rising or falling signal edge at this pin will generate an interrupt request. The corresponding P03PND bit is then automatically set to 1, and the IRQ level goes Low to signal the CPU that an interrupt request is waiting. When the CPU acknowledges the interrupt request, application software must clear the pending condition by writing a 0 to the corresponding P03PND bit.

Table 55. Port 3 Control Register (P03PND; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	E7h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7]	**P0.3 (INT7) External Interrupt Pending Bit** 0: Clear pending bit (when write). 1: Interrupt request is pending (when read).
[6]	**P3.2 (INT6) External Interrupt Pending Bit** 0: Clear pending bit (when write). 1: Interrupt request is pending (when read).

Bit	Description (Continued)
[5]	**P3.1 (INT5) External Interrupt Pending Bit** 0: Clear pending bit (when write). 1: Interrupt request is pending (when read).
[4]	**P3.0 (INT4) External Interrupt Pending Bit** 0: Clear pending bit (when write). 1: Interrupt request is pending (when read).
[3]	**P3.6 (INT3) External Interrupt Pending Bit** 0: Clear pending bit (when write). 1: Interrupt request is pending (when read).
[2]	**P3.5 (INT2) External Interrupt Pending Bit** 0: Clear pending bit (when write). 1: Interrupt request is pending (when read).
[1]	**P3.4 (INT1) External Interrupt Pending Bit** 0: Clear pending bit (when write). 1: Interrupt request is pending (when read).
[0]	**P3.3 (INT0) External Interrupt Pending Bit** 0: Clear pending bit (when write). 1: Interrupt request is pending (when read).

Port 3 Pull-Up Resistor Enable Register

When using the Port 3 Pull-Up Resistor Enable (P3PUR) Register (ECh, Set1, Bank1; see Table 56), the pull-up resistors can be configured to individual Port 3 pins.

Table 56. Port3[4:5] Control Register (P3PUR; Set1, Bank1)

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

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

Bit	Description
[7]	**Reserved**
[6]	**P3.6 Pull-Up Resistor Enable Bit** 0: Pull-up disable. 1: Pull-up enable.
[5]	**P3.5 Pull-Up Resistor Enable Bit** 0: Pull-up disable. 1: Pull-up enable.

Bit	Description (Continued)
[4]	**P3.4 Pull-Up Resistor Enable Bit** 0: Pull-up disable. 1: Pull-up enable.
[3]	**P3.3 Pull-Up Resistor Enable Bit** 0: Pull-up disable. 1: Pull-up enable.
[2]	**P3.2 Pull-Up Resistor Enable Bit** 0: Pull-up disable. 1: Pull-up enable.
[1]	**P3.1 Pull-Up Resistor Enable Bit** 0: Pull-up disable. 1: Pull-up enable.
[0]	**P3.0 Pull-Up Resistor Enable Bit** 0: Pull-up disable. 1: Pull-up enable.

Note: Port 3 pull-up resistors are automatically disabled only when the corresponding pin is selected as a push-pull output or as an alternative function.

Port 3 n-Channel Open Drain Mode Register

When using the Port 3 n-Channel Open Drain Mode (PNE3) Register (EDh, Set1, Bank1; see Table 57), the Push-Pull or Open Drain Output Mode can be configured to individual Port 3 pins.

Table 57. Port 3 Output Pull-Up Resistor Enable Register (PNE3; Set1, Bank1)

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

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

Bit	Description
[7]	**Reserved**
[6]	**P3.6 Output Mode Enable Bit** 0: Output Mode, push-pull. 1: Output Mode, open-drain.
[5]	**P3.5 Output Mode Enable Bit** 0: Output Mode, push-pull. 1: Output Mode, open-drain.

Bit	Description (Continued)
[4]	**P3.4 Output Mode Enable Bit** 0: Output Mode, push-pull. 1: Output Mode, open-drain.
[3]	**P3.3 Output Mode Enable Bit** 0: Output Mode, push-pull. 1: Output Mode, open-drain.
[2]	**P3.2 Output Mode Enable Bit** 0: Output Mode, push-pull. 1: Output Mode, open-drain.
[1]	**P3.1 Output Mode Enable Bit** 0: Output Mode, push-pull. 1: Output Mode, open-drain.
[0]	**P3.0 Output Mode Enable Bit** 0: Output Mode, push-pull. 1: Output Mode, open-drain.

Ports 0–3 Interrupt Control Registers

The contents of the Ports 0–3 Interrupt Control High Byte (P03INTH) and Ports 0–3 Interrupt Control Low Byte (P03INTL) registers are described in Tables 58 and 59.

Table 58. Ports 0–3 Interrupt Control High Byte Registers (P03INTH; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	E8h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:6]	**P0.3 (INT7) External Interrupt Configuration Bit** 00: Disable interrupt. 01: Enable interrupt by falling edge. 10: Enable interrupt by rising edge. 11: Enable interrupt by both falling and rising edge.
[5:4]	**P3.2 (INT6) External Interrupt Configuration Bit** 00: Disable interrupt. 01: Enable interrupt by falling edge. 10: Enable interrupt by rising edge. 11: Enable interrupt by both falling and rising edge.

Bit	Description
[3:2]	**P3.1 (INT5) External Interrupt Configuration Bit** 00: Disable interrupt. 01: Enable interrupt by falling edge. 10: Enable interrupt by rising edge. 11: Enable interrupt by both falling and rising edge.
[1:0]	**P3.0 (INT4) External Interrupt Configuration Bit** 00: Disable interrupt. 01: Enable interrupt by falling edge. 10: Enable interrupt by rising edge. 11: Enable interrupt by both falling and rising edge.

Table 59. Ports 0–3 Interrupt Control Low Byte Registers (P03INTL; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	E9h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:6]	**P3.6 (INT3) External Interrupt Configuration Bit** 00: Disable interrupt. 01: Enable interrupt by falling edge. 10: Enable interrupt by rising edge. 11: Enable interrupt by both falling and rising edge.
[5:4]	**P3.5 (INT2) External Interrupt Configuration Bit** 00: Disable interrupt. 01: Enable interrupt by falling edge. 10: Enable interrupt by rising edge. 11: Enable interrupt by both falling and rising edge.
[3:2]	**P3.4 (INT1) External Interrupt Configuration Bit** 00: Disable interrupt. 01: Enable interrupt by falling edge. 10: Enable interrupt by rising edge. 11: Enable interrupt by both falling and rising edge.
[1:0]	**P3.3 (INT0) External Interrupt Configuration Bit** 00: Disable interrupt. 01: Enable interrupt by falling edge. 10: Enable interrupt by rising edge. 11: Enable interrupt by both falling and rising edge.

10.6. Port 4

Port 4 is an 7-bit I/O port with individually-configurable pins which are accessed directly by writing or reading the Port 4 Data (P4) Register at location F4h in Set1, Bank1. P4.0–P4.6 can serve as inputs (with or without pull-ups), and outputs (push pull or open-drain), or can be configured to the following alternative functions:

- Low-byte pins (P4.0–P4.3): SO/SEG0, SI/SEG1, SCK/SEG2, TD0CAP/TD0OUT/TD0PWM/SEG11
- High-byte pins (P4.4–P4.6): TD0CLK/SEG12, TD1CAP/TD1OUT/TD1PWM/SEG13, TD1CLK/SEG14

Port 4 Control Register

Port 4 provides two 7-bit control registers, P4CONH for P4.4–P4.6 and P4CONL for P4.0–P4.3 (see Tables 60 and 61). A reset clears these P4CONH and P4CONL registers to 00h, configuring all pins to Input Mode. Use these control register settings to select Input Mode (with or without pull-ups) or to select Output Mode and enable the alternative functions.

When programming the port, note that any alternative peripheral I/O function you configure using the Port 4 Control registers must also be enabled in the associated peripheral module.

Table 60. Port 4 Control High Byte Register (P4CONH; Set1, Bank1)

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

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

Bit	Description
[7:6]	**Reserved**
[5:4]	**P4.6/TD1CLK/SEG14Configuration Bits** 00: Schmitt Trigger Input Mode (TD1CLK). 01: Output Mode. 10: Not available. 11: Alternative function (SEG14).

Bit	Description
[3:2]	**P4.5/TD1OUT/TD1PWM/TD1CAP/SEG13 Configuration Bits** 00: Schmitt Trigger Input Mode (TD1CAP). 01: Output Mode. 10: Alternative function (TD1OUT/TD1PWM). 11: Alternative function (SEG13).
[1:0]	**P4.4/TD0CLK/SEG12 Configuration Bits** 00: Schmitt Trigger Input Mode (TD0CLK). 01: Output Mode. 10: Not available. 11: Alternative function (SEG12).

Table 61. Port 4 Control Low Byte Register (P4CONL; Set1, Bank1)

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

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

Bit	Description
[7:6]	**P4.3/TD0OUT/TD0PWM/TD0CAP/SEG11 Configuration Bits** 00: Schmitt Trigger Input Mode (TD0CAP). 01: Output Mode. 10: Alternative function (TD0OUT/TD0PWM). 11: Alternative function (SEG11).
[5:4]	**P4.2/SCK/SEG2 Configuration Bits** 00: Schmitt Trigger Input Mode (SCK In). 01: Output Mode. 10: Alternative function (SCK Out). 11: Alternative function (SEG2).
[3:2]	**P4.1/SI/SEG1 Configuration Bits** 00: Schmitt Trigger Input Mode (SI). 01: Output Mode. 10: Not available. 11: Alternative function (SEG1).
[1:0]	**P4.0/SO/SEG0 Configuration Bits** 00: Schmitt Trigger Input Mode. 01: Output Mode. 10: Alternative function (SO). 11: Alternative function (SEG0).

Port 4 Pull-Up Resistor Enable Register

When using the Port 4 Pull-Up Resistor Enable (P4PUR) Register (F5h, Set1, Bank1; see Table 62), the pull-up resistors can be configured to individual Port 4 pins.

Table 62. Port 4 Pull-Up Resistor Enable Register (P4PUR; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	–	0	0	0	0	0	0	0
R/W	–	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F5h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7]	**Reserved**
[6]	**P4.6 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[5]	**P4.5 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[4]	**P4.4 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[3]	**P4.3 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[2]	**P4.2 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[1]	**P4.1 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[0]	**P4.0 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.

Note: Port 4 pull-up resistors are automatically disabled only when the corresponding pin is selected as a push-pull output or as an alternative function.

Port 4 n-Channel Open Drain Mode Register

When using the Port 4 n-Channel Open Drain Mode (PNE4) Register (F6h, Set1, Bank1; see Table 63), the Push-Pull or Open Drain Output modes can be configured to individual Port 4 pins.

Table 63. Port 4 n-Channel Open-Drain Mode Register (PNE4; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	–	0	0	0	0	0	0	0
R/W	–	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F6h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7]	**Reserved**
[6]	**P4.6 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[5]	**P4.5 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[4]	**P4.4 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[3]	**P4.3 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[2]	**P4.2 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[1]	**P4.1 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.
[0]	**P4.0 Pull-Up Resistor Enable Bit** 0: Disable pull-up resistor. 1: Enable pull-up resistor.

Chapter 11. Basic Timer

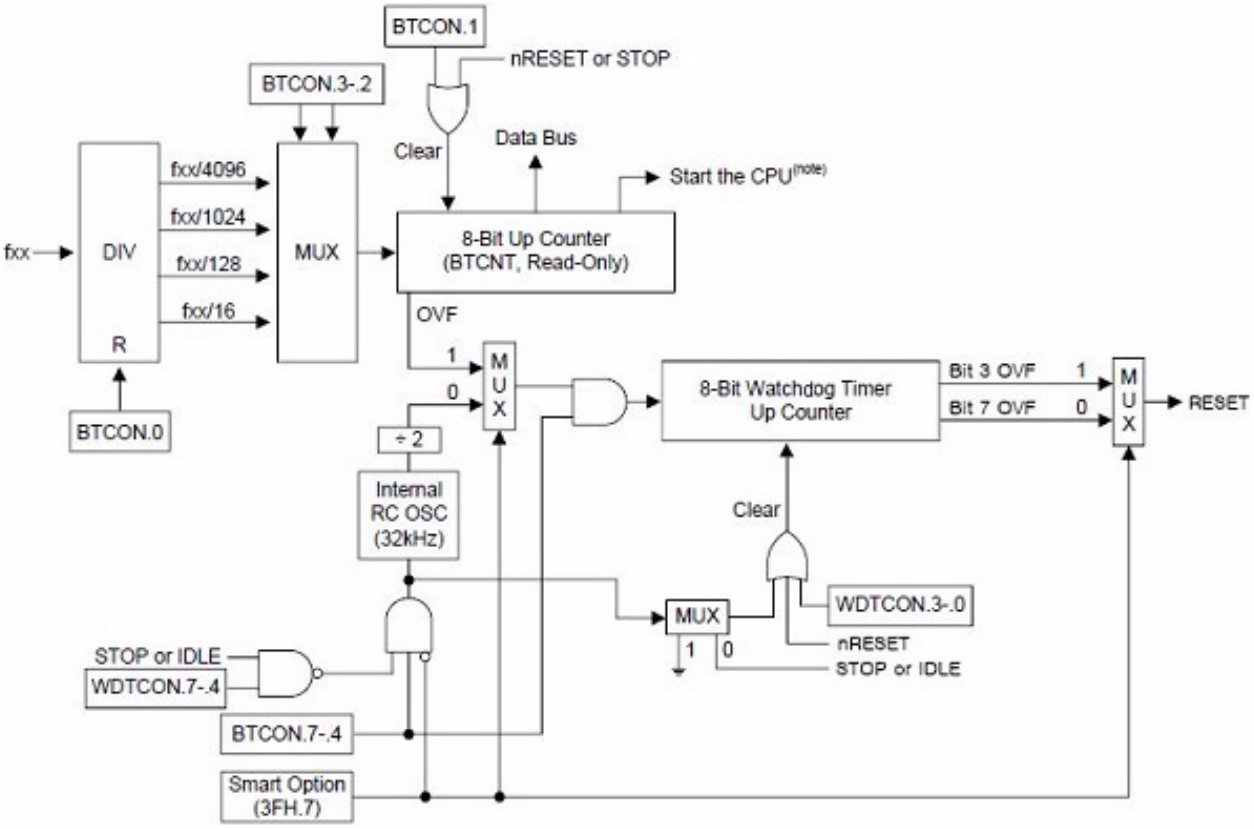
The S3F8S45 MCU features an 8-bit basic timer (BT) that can be used in the following two ways:

- 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_{XX} divided by 4096, 1024, 128, or 16) with multiplexer
- 8-bit Basic Timer Counter (BTCNT) Register (FDh, Set1, Bank0; read-only)
- Basic Timer Control (BTCON) Register (D3h, Set1; read/write)
- Internal RC oscillation or basic timer overflow signal
- Smart Option 3FH.7 selects counter clock source, 8-bit watchdog timer overflow condition (bit 7 OVF with internal ring oscillator or bit 3 OVF with basic timer overflow). Also, in Stop and Idle Modes with internal RC ring oscillator, the watchdog timer counter is not cleared by Smart Option
- Enable or Disable Internal RC OSC in Stop and Idle Mode
- 8-bit watchdog timer counter
- Basic Timer Control (WDTCON) Register (FCh, Set1, Bank1; read/write)

Figure 68 diagrams the Basic Timer.



NOTE: 1. During a power-on reset operation, the CPU is idle during the required oscillation stabilization interval (until bit 4 of the basic timer counter overflows).
2. The internal RC OSC can be stopped in STOP or IDLE mode by WDTCON.7-.4 = "1010B" when the clock is selected for watch-dog timer. At that time the 8-bit watchdog timer up counter will be cleared to '0' when the CPU goes to STOP or IDLE mode.

Figure 68. Basic Timer Block Diagram

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

11.1. Basic Timer Control Register

The Basic Timer Control (BTCON) Register, shown in Table 64, is used to select the input clock frequency, to clear the basic timer counter and frequency dividers, and to enable or disable the watchdog timer function. It is located in Set1, address D3h, and is read/write addressable using Register Addressing Mode.

Table 64. Basic Timer Control Register (BTCON; Set1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	D3h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:4]	**Watchdog Timer Enable Bits** 0000–1001: Reserved. 1010: Disable watchdog timer function. 1011–1111: Reserved.
[3:2]	**Basic Timer Input Clock Selection Bits**[1] 00: f_{XX}/4096. 01: f_{XX} /1024. 10: f_{XX} /128. 11: f_{XX} /16.
[1]	**Basic Timer Counter Clear Bit**[2] 0: No effect. 1: Clear the basic timer counter value.
[0]	**Clock Frequency Divider Clear Bit for Basic Timer and Timer/Counters**[3] 0: No effect. 1: Clear both clock frequency dividers.

Notes:
1. f_{XX} is the clock selected for the system (i.e., for the main oscillator. or the suboscillator).
2. When writing a 1 to BTCON.1, the basic timer counter value is cleared to 00h. Immediately following this write operation, the BTCON.1 value is automatically cleared to 0.
3. When writing a 1 to BTCON.0, the corresponding frequency divider is cleared to 00h. Immediately following this write operation, the BTCON.0 value is automatically cleared to 0.

A reset clears BTCON to 00h, enabling the watchdog function and selecting a basic timer clock frequency of f_{XX}/4096. To disable the watchdog function, write the signature code 1010b to the basic timer register control bits BTCON.7–BTCON.4.

The 8-bit Basic Timer Counter (BTCNT) Register (FDh, Set1, Bank0), can be cleared at any time during normal operation by writing a 1 to BTCON.1. To clear the frequency dividers, write a 1 to BTCON.0.

Watchdog Timer Function

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

A reset is generated whenever a watchdog timer counter overflow occurs (a bit 3 overflow or a bit 7 overflow of the 8-bit watchdog timer counter). During normal operation, the application program must prevent the overflow and the accompanying reset operation from occurring by clearing the BTCNT/watchdog counter value (i.e., by writing a 1 to BTCON.1 or by writing a 1010b to WDTCON.3–WDTCON.0) at regular intervals.

If a system malfunction occurs due to circuit noise or some other error condition, the watchdog counter clear operation will not be executed and a watchdog timer overflow will occur, initiating a reset. After this reset, the device will carry out nornal operation again. The contents of the Watchdog Timer Control (WDTCON) Register are described in Table 65.

Table 65. Watchdog Timer Control Register (WDTCON; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	F2h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:4]	**Internal RC Oscillator Enable Bits In Stop/Idle Mode**[1] 1010: Disable internal RC oscillator in Stop/Idle modes. All others: Enable internal RC oscillator in Stop/Idle modes.
[3:0]	**Watchdog Timer Counter Clear Bits**[2] 1010: Clear watchdog timer counter. All others: No effect.

Notes:
1. Only when the internal RC oscillator is selected for the watchdog timer clock source by the Smart Option.
2. The WDTCON.3–.0 value is automatically cleared to 0 after being cleared by the watchdog counter.

Watchdog Timer Counter Clock Source Selection

You can select either the basic timer overflow signal or the internal RC ring oscillator as the counter clock source. If you use basic timer overflow clock source, WDT overflow will occur when counter bit 3 overflows. If you use internal RC ring oscillator clock source, WDT overflow will occur when counter bit 7 overflows.

Oscillation Stabilization Interval Timer Function

The basic timer can also be used to program a specific oscillation stabilization interval after a reset, or when Stop Mode has been released by an external interrupt.

In Stop Mode, whenever a reset or an external interrupt occurs, the oscillator starts. The BTCNT value then begins increasing at the rate of $f_{XX}/4096$ (for a reset), or at the rate of the preset clock source (for an external interrupt). When BTCNT.3 overflows, a signal is generated to indicate that the stabilization interval has elapsed and to gate the clock signal off to the CPU so that it can resume normal operation.

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

1. During Stop Mode, a power-on reset or an external interrupt occurs to trigger the Stop Mode release, and oscillation begins.
2. If a power-on reset occurs, the basic timer counter will increase at the rate of $f_{XX}/4096$. If an interrupt is used to release Stop Mode, the BTCNT value increases at the rate of the preset clock source.
3. A clock oscillation stabilization interval begins and continues until the 4th bit of the basic timer counter (BTCNT.3) overflows.
4. When a BTCNT.3 overflow occurs, normal CPU operation resumes.

Chapter 12. 8-Bit Timer A/B

This chapter discusses the S3F8S45 MCU's two 8-bit timers, Timer A and Timer B.

12.1. 8-Bit Timer A

The 8-bit Timer A is an 8-bit general-purpose timer/counter featuring three operating modes, each of which can be selected using one of the following TACON settings:

- Interval Timer Mode (toggle output at TAOUT pin)
- Capture Input Mode with a rising or falling edge trigger at the TACAP pin
- PWM Mode (TAPWM)

Timer A provides the following functional components:

- Clock frequency divider (f_{XX} divided by 1024, 256, 64, 8, or 1) with multiplexer
- External clock input pin (TACLK)
- 8-bit counter (TACNT), 8-bit comparator, and 8-bit reference data register (TADATA)
- I/O pins for capture input (TACAP), PWM output (TAPWM), or match output (TAOUT)
- Timer A overflow interrupt (IRQ0 vector D0h) and match/capture interrupt (IRQ0 vector CEh) generation
- Timer A Control (TACON) Register (E2h, Set1, Bank0; read/write)

Figure 69 presents a block diagram of the Timer A function.

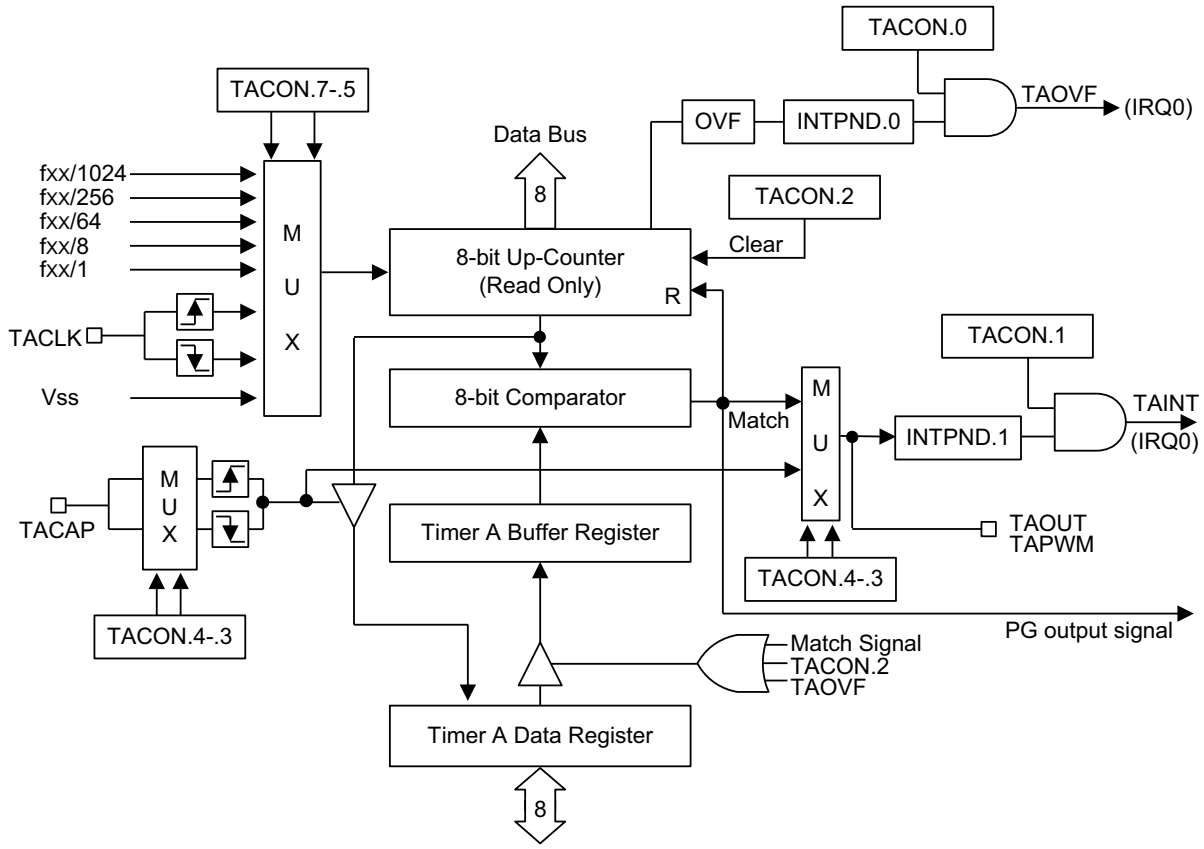


Figure 69. Timer A Functional Block Diagram

12.2. Timer A Control Register

The Timer A Control (TACON) Register, shown in Table 66, can be used to perform the following tasks:

- Select the Timer A operating mode (Interval Timer, Capture Mode, or PWM Mode)
- Select the Timer A input clock frequency
- Clear the Timer A counter, TACNT
- Enable the Timer A overflow interrupt or Timer A match/capture interrupt

Table 66. Timer A Control Register (TACON; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	E2h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:5]	**Timer A Input Clock Selection Bits** 000: f_{XX} /1024. 001: f_{XX} /256. 010: f_{XX} /64. 011: f_{XX} /8. 100: f_{XX} /1. 101: External clock (TACLK) falling edge. 110: External clock (TACLK) rising edge. 111: Counter stop.
[4:3]	**Timer A Operating Mode Selection Bits** 00: Interval Mode (TAOUT). 01: Capture Mode (Capture on rising edge, counter running, OVF can occur). 10: Capture Mode (Capture on falling edge, counter running, OVF can occur). 11: PWM Mode; OVF and match interrupt can occur.
[2]	**Timer A Counter Enable Bit** 0: No effect. 1: Clear the Timer A counter (when write).
[1]	**Timer A Match/Capture Interrupt Enable Bit** 0: Disable interrupt. 1: Enable interrupt.
[0]	**Timer A Overflow Interrupt Enable Bit** 0: Disable interrupt. 1: Enable interrupt.

Note: The TACON.2 value is automatically cleared to 0 after being cleared by the counter.

TACON is located in Set1, Bank0 at address E2h, and is read/write addressable using Register Addressing Mode.

A reset clears TACON to 00h, thereby setting Timer A to normal Interval Timer Mode and selecting an input clock frequency of f_{XX}/1024; all Timer A interrupts are disabled. The Timer A counter can be cleared at any time during normal operation by writing a 1 to TACON.2.

The IRQ0-level Timer A overflow interrupt (TAOVF) is at vector address D0h. When a Timer A overflow interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware or must be cleared by software.

To enable the Timer A match/capture interrupt (IRQ0, vector CEh), write a 1 to TACON.1. To detect a match/capture interrupt pending condition, the application software polls INTPND.1. When a 1 is detected, a Timer A match or capture interrupt is pending. When IMR.0 = 1 and TAINT is enabled, theIRQ0 and INTPND.1 are cleared by hardware when the interrupt request has been serviced.

When IMR.0 = 0 and TAINT is enabled, the IRQ0 and INTPND.1 are cleared by software by writing a 0 to the Timer A match/capture interrupt pending bit, INTPND.1

12.3. Timer A Function Description

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

12.4. Timer A Interrupts

Timer A can generate two interrupts: the Timer A overflow interrupt (TAOVF) and the Timer A match/capture interrupt (TAINT). TAOVF occurs at interrupt level IRQ0, vector D0h. TAINT also occurs at IRQ0, but is assigned the separate vector address CEh.

A Timer A overflow interrupt pending condition is automatically cleared by hardware when it has been serviced, or should be cleared by software in the interrupt service routine by writing a 0 to the INTPND.0 interrupt pending bit. When IMR.0 = 1 and TAINT is enabled, the IRQ0 and INTPND.1 are cleared by hardware when the interrupt request has been serviced.

When IMR.0 = 0 and TAINTis enabled, the IRQ0 and INTPND.1 are cleared by software by writing a 0 to the Timer A match/capture interrupt pending bit, INTPND.1

12.5. Timer A Interval Timer Mode

In Interval Timer Mode, a match signal is generated when the counter value is identical to the value written to the Timer A Reference Data (TADATA) Register. The match signal generates a Timer A match interrupt (TAINT, vector CEh) and clears the counter.

If, for example, you write the value 10h to TADATA, the counter will increment until it reaches 10h. At this point, the Timer A interrupt request is generated, the counter value is reset, and counting resumes. With each match, the level of the signal at the Timer A output pin is inverted; see Figure 70.

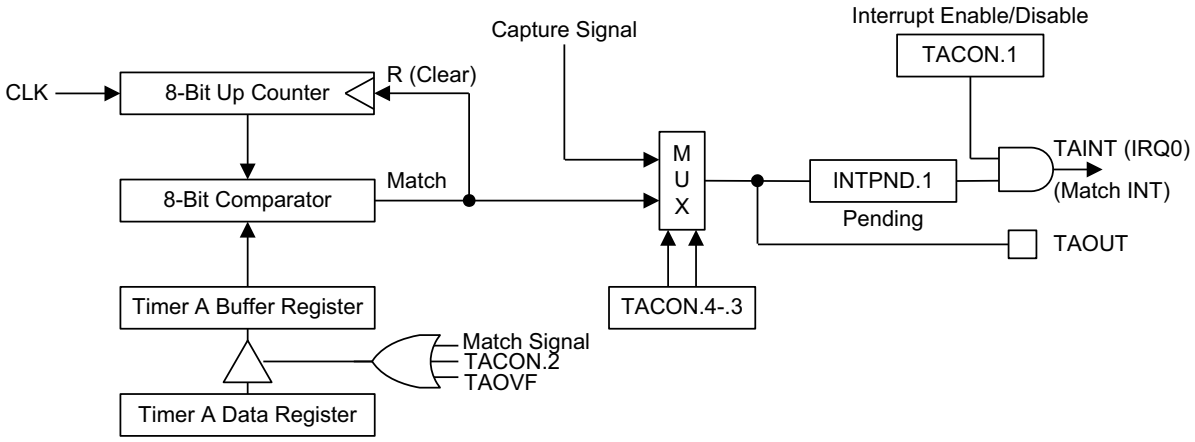


Figure 70. Simplified Timer A Function Diagram: Interval Timer Mode

12.6. Timer A Pulse Width Modulation Mode

Pulse Width Modulation (PWM) Mode lets you program the width (duration) of the pulse that is output at the TAPWM pin. As in Interval Timer Mode, a match signal is generated when the counter value is identical to the value written to the Timer A Data Register. In PWM Mode, however, the match signal does not clear the counter. Instead, it runs continuously, overflowing at FFh, then continues incrementing from 00h.

Although the match signal can be used to generate a Timer A overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the TAPWM pin is held Low as long as the reference data value is less than or equal to (≤) the counter value, the pulse is then held High for as long as the data value is greater than (>) the counter value. One pulse width is equal to t_{CLK} x 256; see Figure 71.

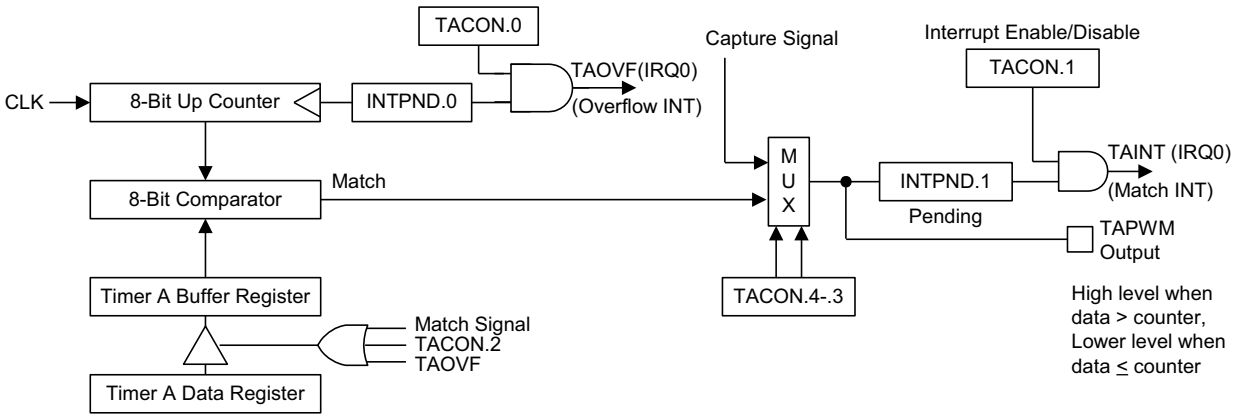


Figure 71. Simplified Timer A Function Diagram: PWM Mode

12.7. Timer A Capture Mode

In Capture Mode, a signal edge that is detected at the TACAP pin opens a gate and loads the current counter value into the Timer A Data Register. Rising or falling edges can be selected to trigger this operation.

Timer A also provides a capture input source: the signal edge at the TACAP pin. Select the capture input by setting the values of the Timer A capture input selection bits in the Port 1 Control Register, P1CONL.4, (E1h, Set1, Bank1). When P1CONL.4 is 0, the TACAP input is selected.

Both kinds of Timer A interrupts can be used in Capture Mode: the Timer A overflow interrupt is generated whenever a counter overflow occurs, and the Timer A match/capture interrupt is generated whenever the counter value is loaded into the Timer A Data Register.

By reading the captured data value in TADATA, and assuming a specific value for the Timer A clock frequency, you can calculate the pulse width (duration) of the signal that is being input at the TACAP pin; see Figure 72.

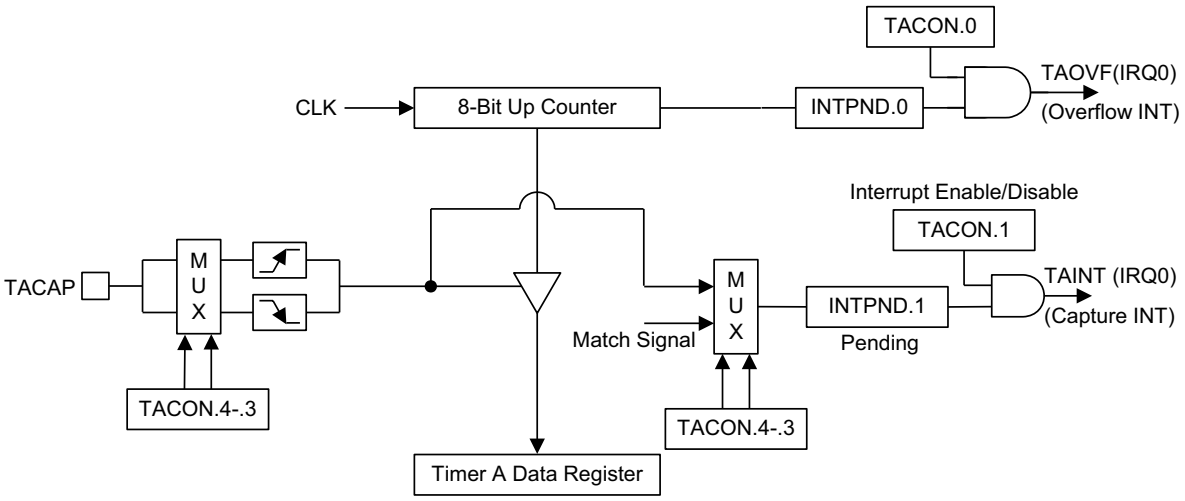


Figure 72. Simplified Timer A Function Diagram: Capture Mode

12.8. 8-Bit Timer B

The S3F8S45 microcontroller has an 8-bit counter called Timer B, which can be used to generate the carrier frequency of a remote controller signal. Timer B offers the following two functions:

- As a normal interval timer, generating a Timer B interrupt at programmed time intervals
- To supply a clock source to the 8-bit timer/counter module, Timer B, for generating the Timer B overflow interrupt

Figure 73 presents a block diagram of the Timer B function.

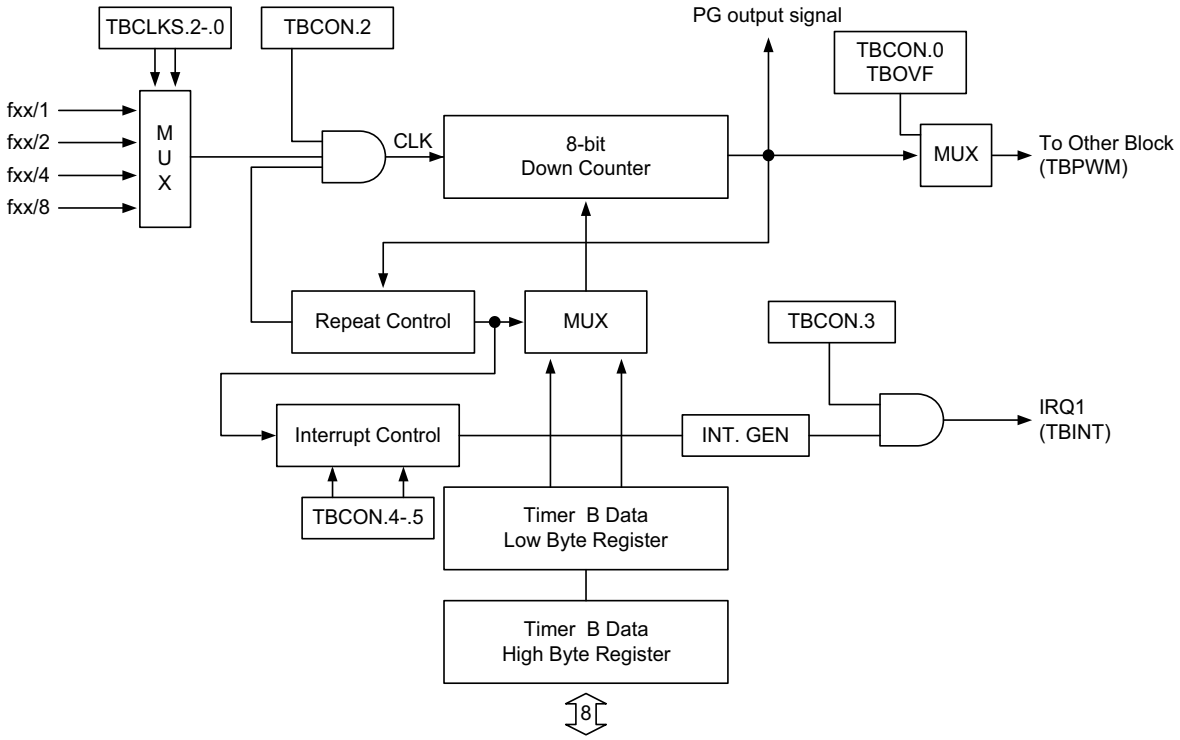


Figure 73. Timer B Functional Block Diagram

Note: The value of the TBDATAL Register is loaded into the 8-bit counter when the operation of Timer B starts. If A borrow occurs in the counter, the value of The TBDATAH Register is loaded into the 8-bit counter. However, on the next borrow, the value of the TBDATAL Register is loaded into the bit counter.

The Timer B Control (TBCON) Register is shown in Table 67.

Table 67. Timer B Control Register (TBCON; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	E3h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:6]	**Timer B Input Clock Selection Bits** 00: f_{OSC} /1. 01: f_{OSC} /2. 10: f_{OSC} /4. 11: f_{OSC} /8.
[5:4]	**Timer B Interrupt Time Selection Bits** 00: Generating after low data is borrowed. 01: Generating after high data is borrowed. 10: Generating after low and high data are borrowed. 11: Not available.
[3]	**Timer B Interrupt Enable Bit** 0: Disable interrupt. 1: Enable interrupt.
[2]	**Timer B Start/Stop Bit** 0: Stop Timer B. 1: Start Timer B.
[1]	**Timer B Mode Selection Bit** 0: One-Shot Mode. 1: Repeat Mode.
[0]	**Timer B Output Flip-Flop Control Bit** 0: TBOF is Low (TBPWM: low level for low data, high level for High data). 1: TBOF is High (TBPWM: high level for low data, low level for High data).

12.9. Timer B Pulse Width Calculations

Figure 74 presents an example waveform consisting of a low period time, t_{LOW}, and high period time, t_{HIGH}.

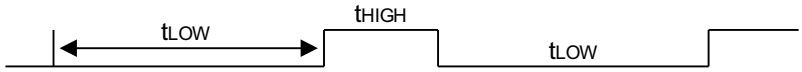


Figure 74. Timer B Waveform, Example #1 of 3

To generate the waveform in Figure 74, observe the following calculations.

When TBOF = 0:
t_{LOW} = (TBDATAL + 2) x $1/f_X$, 0h < TBDATAL < 100h, where f_X = the selected clock.
t_{HIGH} = (TBDATAH + 2) x $1/f_X$, 0h < TBDATAH < 100h, where f_X = the selected clock.

When TBOF = 1:
t_{LOW} = (TBDATAH + 2) x $1/f_X$, 0h < TBDATAH < 100h, where f_X = the selected clock.
t_{HIGH} = (TBDATAL + 2) x $1/f_X$, 0h < TBDATAL < 100h, where f_X = the selected clock.

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

When TBOF = 0:
t_{LOW} = 24 µs = (TBDATAL + 2) / f_X = (TBDATAL + 2) x 1 µs, TBDATAL = 22.
t_{HIGH} = 15 µs = (TBDATAH + 2) / f_X = (TBDATAH + 2) x 1 µs, TBDATAH = 13.

When TBOF = 1:
t_{HIGH} = 15 µs = (TBDATAL + 2) / f_X = (TBDATAL + 2) x 1 µs, TBDATAL = 13.
t_{LOW} = 24 µs = (TBDATAH + 2) / f_X = (TBDATAH + 2) x 1 µs, TBDATAH = 22.

Figure 75 presents the waveform output of the Timer B output in Repeat Mode.

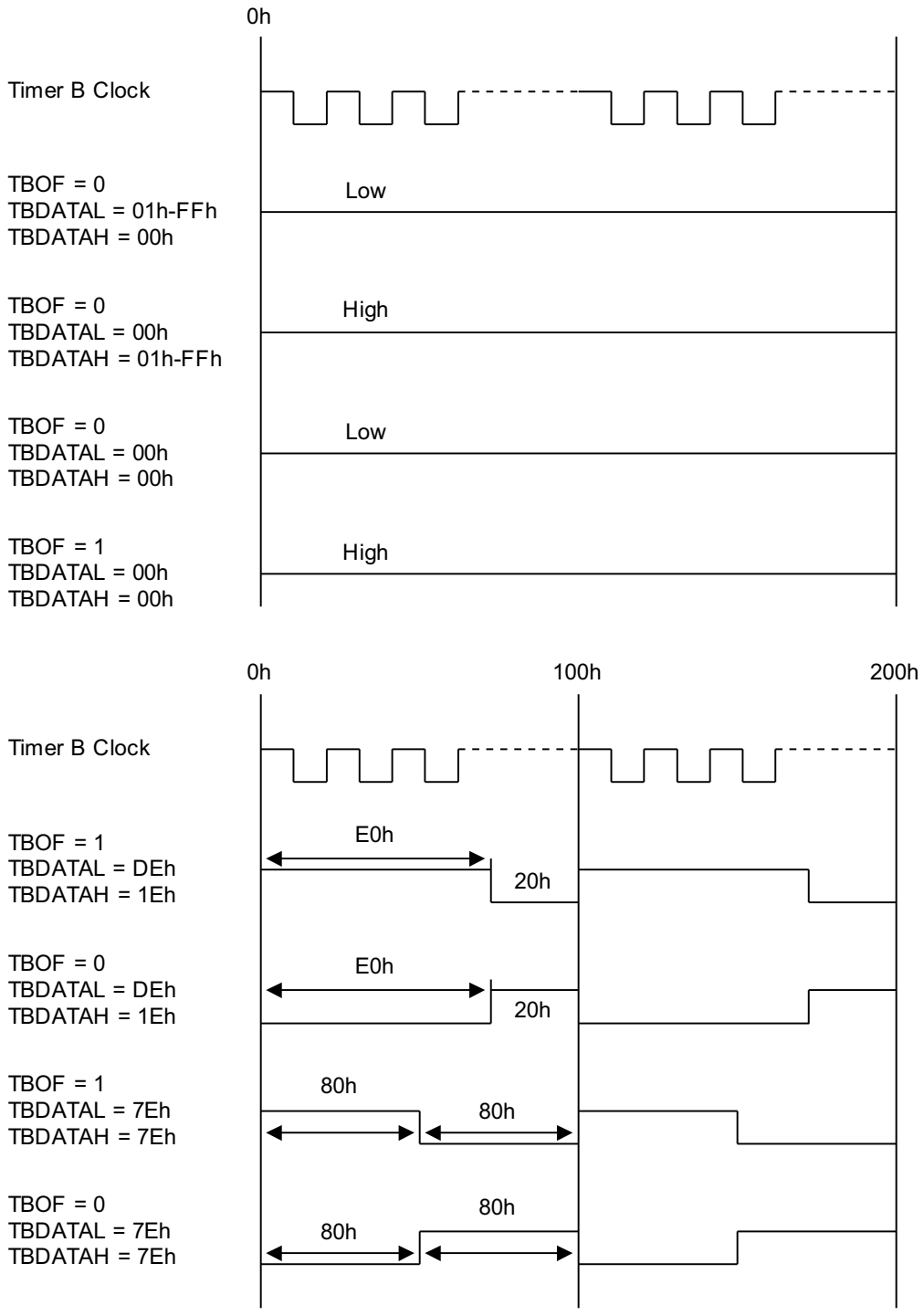


Figure 75. Timer B Output Flip-Flop Waveforms in Repeat Mode

Figure 76 presents an example waveform that sets Timer B to Repeat Mode, sets the oscillation frequency as the Timer B clock source, and causes TBDATAH and TBDATAL to establish a 38kHz, 1/3-duty carrier frequency.

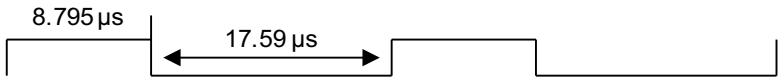


Figure 76. Timer B Waveform, Example #2 of 3

To generate the waveform in Figure 76, the following factors form the calculation, which follows.

- Timer B is used in Repeat Mode
- The oscillation frequency is 4MHz (1 clock = 0.25µs)
- TBDATAH = 8.795µs/0.25µs = 35.18, TBDATAL = 17.59µs/0.25µs = 70.36
- P1.0 is set to TBPWM Mode

```
      ORG 0100h              ;Reset address
START DI
      •
      •
      •
      LD  TBDATAL,#(70-2)    ;Set 17.5 µs
      LD  TBDATAH,#(35-2)    ;Set 8.75 µs
      LD  TBCON,#00000111b   ;Clock Source ← fxx
                             ;Disable Timer B interrupt.
                             ;Select Repeat Mode for Timer B.
                             ;Start Timer B operation.
                             ;Set Timer B Output flip-flop (TBOF)
                             ;High.
      OR  P1CONL,#00000011b  ;Set P1.0 to TBPWM Mode.
                             ;This command generates 38kHz, 1/3
                             ;duty pulse signal through P1.0.
      •
      •
      •
```

Figure 77 presents an example waveform that sets Timer B to One Shot Mode, sets the oscillation frequency as the Timer B clock source, and causes TBDATAH and TBDATAL to establish a 40µs-width pulse.

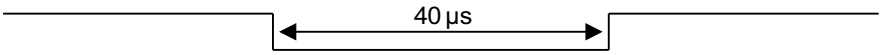


Figure 77. Timer B Waveform, Example #3 of 3

To generate the waveform in Figure 77, the following factors form the calculation, which follows.

- Timer B is used in One Shot Mode
- The oscillation frequency is 4 MHz (1 clock = 0.25 µs)
- TBDATAH = 40 µs/0.25 µs = 160, TBDATAL = 1
- P1.0 is set to TBPWM Mode

```
      ORG   0100h           ;Reset address
START DI
      •
      •
      •
      LD    TBDATAH,# (160-2);Set 40 µs
      LD    TBDATAL,# 1     ;Set any value except 00H
      LD    TBCON,#00000001b;Clock Source ← fOSC
                            ;Disable Timer B interrupt.
                            ;Select One Shot Mode for Timer B.
                            ;Stop Timer B operation.
                            ;Set Timer B output flip-flop (TBOF)
                            ;High
      OR P2CONL, #00000011b ;Set P1.0 to TBPWM Mode.
      •
      •
Pulse_out: LD TBCON,#00000101b ;Start Timer B operation
                            ;To make the pulse at this point.
                            ;After the instruction is executed,
      •                     ;0.75 µs is required
      •                     ;before the falling edge of the pulse
      •                     ;starts.
```

Chapter 13. 8-Bit Timer C

This chapter discusses the S3F8S45 MCU's 8-bit Timer C, an 8-bit general-purpose timer/counter featuring two operating modes, each of can be selected using one of the following two TCCON settings:

- Interval Timer Mode (toggle output at the TCOUT/TCPWM pins), in which only a match interrupt occurs
- PWM Mode (TCOUT/TCPWM pin), in which match and overflow interrupts can occur

Timer C also features the following functional components:

- Clock frequency divider with multiplexer
- 8-bit counter, 8-bit comparator, and 8-bit reference data register (TCDATA)
- PWM or match output (TCOUT/TCPWM)
- Timer C match/overflow interrupt (IRQ2, vector D4h) generation
- Timer C Control (TCCON) Register; Set1, Bank0, ECh, read/write

Figure 78 presents a block diagram of the Timer C function.

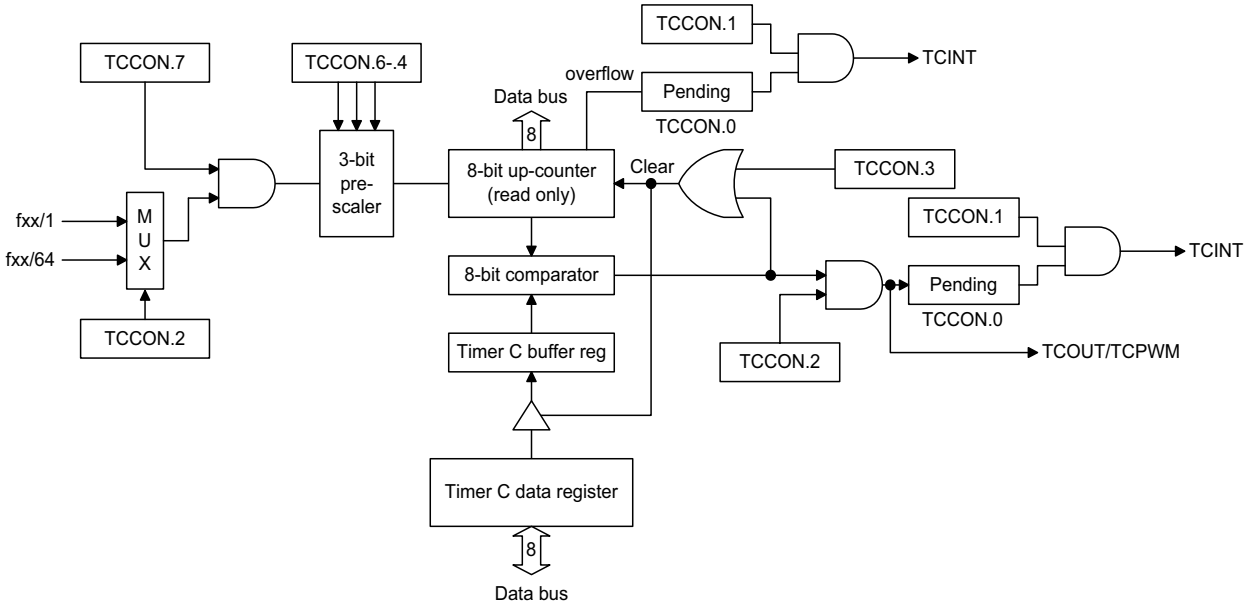


Figure 78. Timer C Functional Block Diagram

Note: When operating in PWM Mode, the match signal cannot clear the counter.

13.1. Timer C Control Register

The Timer C Control (TCCON) Register, shown in Table 68, can be used to perform the following tasks:

- Select the Timer C operating mode (f_{XX}/1 & PWM Mode or f_{XX}/64 & Interval Mode)
- Select the Timer C 3-bits prescaler
- Clear the Timer C counter, TCCNT
- Enable the Timer C match/overflow interrupt
- Start the Timer C

TCCON is located in Set1, Bank0 at address ECh, and is read/write addressable using Register Addressing Mode. At System Reset, TCCON is cleared, thereby disabling Timer C.

To enable the Timer C match/overflow interrupt (IRQ2, vector D4h), write a 1 to TCCON.7 and TCCON.1. To generate the exact time interval, write a 0 to TCCON.3 to clear the counter and the interrupt pending bit. To detect an interrupt pending condition when TCINT is disabled, the application software polls the pending bit, TCCN.0. When a 1 is detected, a Timer C match/overflow interrupt is pending. When IMR.2 = 1 and TCINT is enabled, the IRQ2 and TCCN.0 are cleared when the TCINT subroutine has been serviced. When IMR.2 = 0 and TCINT is enabled, the IRQ2 and TCCN.0 are cleared by software by writing a 0 to TCCN.0.

Table 68. Timer C Control Register (TCCON; Set1, Bank0)

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

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

Bit	Description
[7]	**Timer C Start/Stop Bit** 0: Stop Timer C. 1: Start Timer C.
[6:4]	**Timer C 3-Bit Prescaler Bits** 000: Not divided. 001: Divided by 2. 010: Divided by 3. 011: Divided by 4. 100: Divided by 5. 101: Divided by 6. 110: Divided by 7. 111: Divided by 8.
[3]	**Timer C Counter Clear Bit*** 0: No effect. 1: Clear the Timer C counter (when write).
[2]	**Timer C Mode Selection Bit** 0: f_{XX}/1 and PWM Mode. 1: f_{XX}/64 and Interval Mode.
[1]	**Timer C Interrupt Enable Bit** 0: Disable interrupt. 1: Enable interrupt.
[0]	**Timer C Interrupt Pending Bit** 0: No interrupt pending (when read), clear pending bit (when write). 1: Interrupt is pending (when read).

Note: *The TCCON.3 value is automatically cleared to 0 after being cleared by the counter.

Chapter 14. 16-Bit Timer D0/D1

This chapter discusses the S3F8S45 MCU's 16-bit Timer D0/D1, an 8-bit general-purpose timer. Timer D0 features the following three operating modes, each of which can be selected using the appropriate TD0CON setting:

- Interval Timer Mode (toggle output at the TD0OUT pin)
- Capture Input Mode with a rising or falling edge trigger at the TD0CAP pin
- PWM Mode (TD0PWM); the PWM output shares its output port with the TD0OUT pin

Timer D0 also features the following functional components:

- Clock frequency divider (f_{XX} divided by 1024, 256, 64, 8, 1) with multiplexer
- External clock input pin (TD0CLK)
- A 16-bit counter (TD0CNTH/TD0CNTHL), a 16-bit comparator, and two 16-bit reference data registers (TD0DATAH/TD0DATAL)
- I/O pins for capture input (TD0CAP) or match output (TD0OUT)
- Timer D0 overflow interrupt (IRQ3, vector `DAh`) and match/capture interrupt generation (IRQ3, vector `D8h`)
- Timer D0 Control (TD0CON) Register (Page 4, `01h`; read/write)

Figure 79 presents a block diagram of the Timer D0 function.

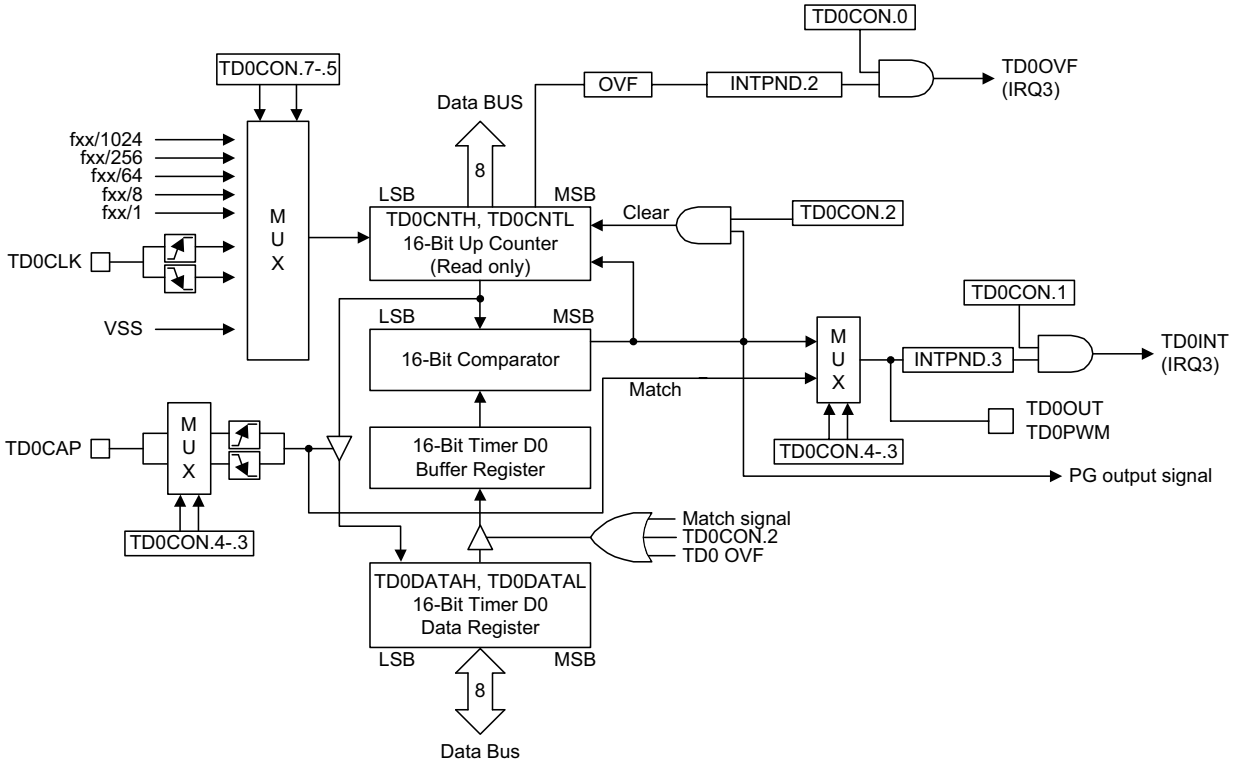


Figure 79. Timer D0 Functional Block Diagram

14.1. Timer D0 Control Register

Use the Timer D0 Control (TD0CON) Register, shown in Table 69, to perform the following tasks:

- Select the Timer D0 operating mode (Interval Timer, Capture Mode, or PWM Mode)
- Select the Timer D0 input clock frequency
- Clear the Timer D0 counter, TD0CNTH/TD0CNTL
- Enable the Timer D0 overflow interrupt or Timer D0 match/capture interrupt

TD0CON is located in Page 4 at address `01h`, and is read/write addressable using Register Addressing Mode.

A reset clears TD0CON to `00h`, thereby setting Timer D0 to normal Interval Timer Mode, and selecting an input clock frequency of $f_{XX}/1024$; all Timer D0 interrupts are disabled.

To disable counter operation, set TD0CON.7–.5 to 111b. Clear the Timer D0 counter at any time during normal operation by writing a 1 to TD0CON.2.

Table 69. Timer D0 Control Register (TD0CON; Page 4)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	01h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:5]	**Timer D0 Input Clock Selection Bits** 000: f_{XX}/1024. 001: f_{XX}/256. 010: f_{XX}/64. 011: f_{XX}/8. 100: f_{XX}/1. 101: External clock (TD0CLK) falling edge. 110: External clock (TD0CLK) rising edge. 111: Counter stop.
[4:3]	**Timer D0 Operating Mode Selection Bits** 00: Interval Mode (TD0OUT). 01: Capture Mode (Capture on rising edge, counter running, OVF can occur). 10: Capture Mode (Capture on falling edge, counter running, OVF can occur. 11: PWM Mode (OVF and match interrupt can occur).
[2]	**Timer D0 Counter Clear Bit*** 0: No effect. 1: Clear the Timer D0 counter (when write).
[1]	**Timer D0 Match/Capture Interrupt Enable Bit** 0: Disable interrupt. 1: Enable interrupt.
[0]	**Timer D0 Overflow Interrupt Enable Bit** 0: Disable overflow interrupt. 1: Enable overflow interrupt.

Note: *The TD0CON.2 value is automatically cleared to 0 after being cleared by the counter.

The IRQ3-level Timer D0 overflow interrupt (TD0OVF) is at vector address DAh. When a Timer D0 overflow interrupt occurs and is serviced by an interrupt (IRQ3, vector DAh), write a 1 to TD0CON.0. When a Timer D0 overflow interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware or must be cleared by software.

To enable the Timer D0 match/capture interrupt (IRQ3, vector D8h), write a 1 to TD0CON.1. To detect a match/capture interrupt pending condition, the application software polls INTPND.3. When a 1 is detected, a Timer D0 match or capture interrupt is pending. When IMR.3 = 1 and TD0INT is enabled, the IRQ3 and INTPND.3 are cleared by hardware when the interrupt request has been serviced.

When IMR.3 = 0 and TD0INT is enabled, the IRQ3 and INTPND.3 are cleared by software by writing a 0 to the Timer D0 match/capture interrupt pending bit, INTPND.3.

14.2. Timer D0 Function Description

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

14.3. Timer D0 Interrupts

Timer D0 can generate two interrupts: the Timer D0 overflow interrupt (TD0OVF) and the Timer D0 match/capture interrupt (TD0INT). TD0OVF is associated to interrupt level IRQ3 at vector DAh. TD0INT is also associated to interrupt level IRQ3, but is assigned the separate vector address, D8h.

A Timer D0 overflow interrupt pending condition is automatically cleared by hardware when it has been serviced or should be cleared by software in the interrupt service routine by writing a 0 to the INTPND.2 interrupt pending bit. When IMR.3 = 1 and TD0INT is enabled, the IRQ3 and INTPND.3 are cleared by hardware when the interrupt request has been serviced.

When IMR.3 = 0 and TD0INT is enabled, the IRQ3 and INTPND.3 are cleared by software by writing a 0 to the Timer D0 match/capture interrupt pending bit, INTPND.3

14.4. Timer D0 Interval Timer Mode

In Interval Timer Mode, a match signal is generated when the counter value is identical to the value written to the Timer D0 Reference Data Register, TD0DATAH/TD0DATAL. The match signal generates a Timer D0 match interrupt (TD0INT, vector D8h) and clears the counter.

If, for example, you write the value 1087h to TD0DATAH/TD0DATAL, the counter will increment until it reaches 1087h. At this point, the Timer D0 interrupt request is generated, the counter value is reset, and counting resumes. With each match, the level of the signal at the Timer D0 output pin is inverted; see Figure 80.

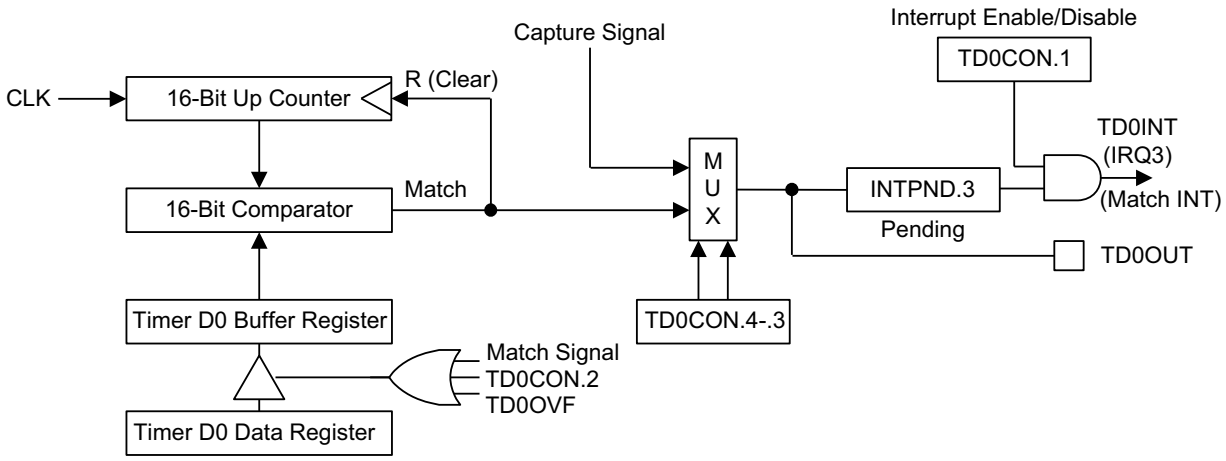


Figure 80. Simplified Timer D0 Function Diagram: Interval Timer Mode

14.5. Timer D0 Pulse Width Modulation Mode

Pulse Width Modulation (PWM) Mode lets you program the width (duration) of the pulse that is output at the TD0PWM pin. As in Interval Timer Mode, a match signal is generated when the counter value is identical to the value written to the Timer D0 Data Register. In PWM Mode, however, the match signal does not clear the counter. Instead, it runs continuously, overflowing at FFFFh, then continues incrementing from 0000h.

Although you can use the match signal to generate a Timer D0 overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the TD0PWM pin is held Low as long as the reference data value is less than or equal to (≤) the counter value; the pulse is then held High for as long as the data value is greater than (>) the counter value. One pulse width is equal to t_{CLK} x 65536; see Figure 81.

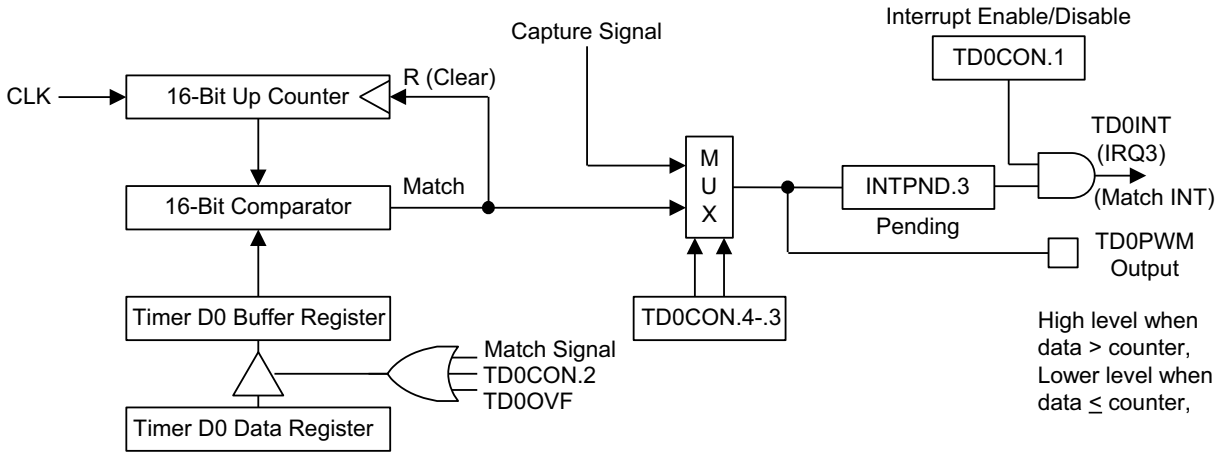


Figure 81. Simplified Timer D0 Function Diagram: PWM Mode

14.6. Timer D0 Capture Mode

In Capture Mode, a signal edge that is detected at the TD0CAP pin opens a gate and loads the current counter value into the Timer D0 Data Register. Rising or falling edges can be selected to trigger this operation.

Timer D0 also provides an capture input source: the signal edge at the TD0CAP pin. Select the capture input by setting the values of the Timer D0 capture input selection bits in the Port 2 Control Register, P2CONL.3–.2 (Set1, Bank1, `E7h`). When P2CONL.3–.2 is 00, the TD0CAP input is selected.

Both kinds of Timer D0 interrupts can be used in Capture Mode: the Timer D0 overflow interrupt is generated whenever a counter overflow occurs; the Timer D0 match/capture interrupt is generated whenever the counter value is loaded into the Timer D0 Data Register.

By reading the captured data value in TD0DATAH/TD0DATAL, and assuming a specific value for the Timer D0 clock frequency, you can calculate the pulse width (duration) of the signal that is being input at the TD0CAP pin; see Figure 82.

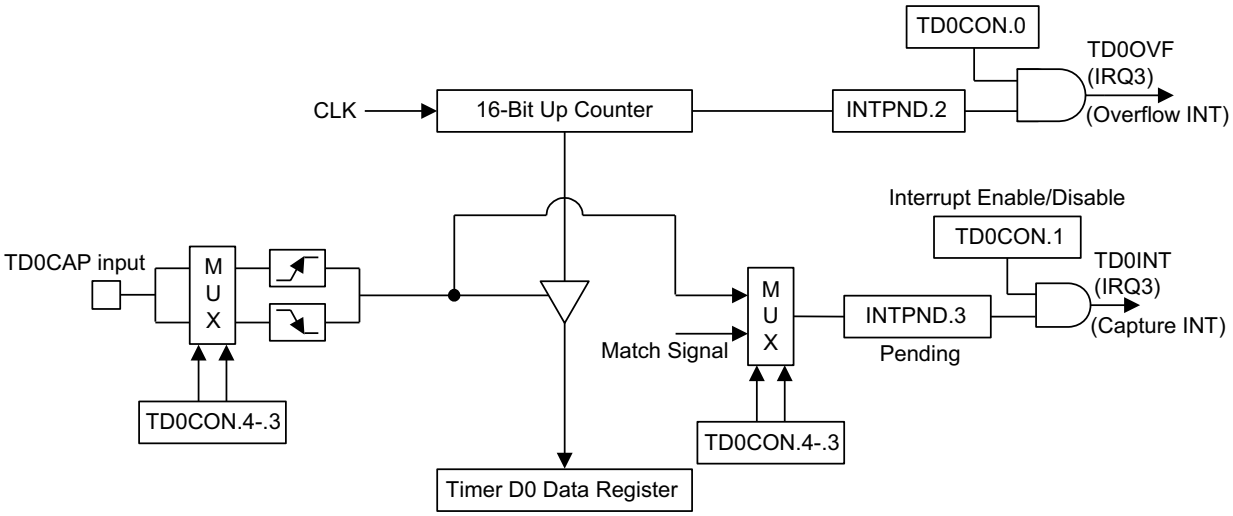


Figure 82. Simplified Timer D0 Function Diagram: Capture Mode

Chapter 15. 16-Bit Timer D1

The 16-bit Timer D1 is a 16-bit general-purpose timer featuring three operating modes, each of which can be selected using one of the following TD1CON settings:

- Interval Timer Mode (toggle output at the TD1OUT pin)
- Capture Input Mode with a rising or falling edge trigger at the TD1CAP pin
- PWM Mode (TD1PWM); PWM output shares its output port with the TD1OUT pin

Timer D1 also features the following functional components:

- Clock frequency divider (f_{XX} divided by 1024, 256, 64, 8, 1) with multiplexer
- External clock input pin (TD1CLK)
- A 16-bit counter (TD1CNTH/TD1CNTL), a 16-bit comparator, and two 16-bit reference data registers (TD1DATAH/TD1DATAL)
- I/O pins for capture input (TD1CAP), or match output (TD1OUT)
- Timer D1 overflow interrupt (IRQ3, vector DEh) and match/capture interrupt (IRQ3, vector DCh) generation
- Timer D1 Control (TD1CON) Register (Page 4, 0Ah; read/write)

Figure 83 presents a block diagram of the Timer D1 function.

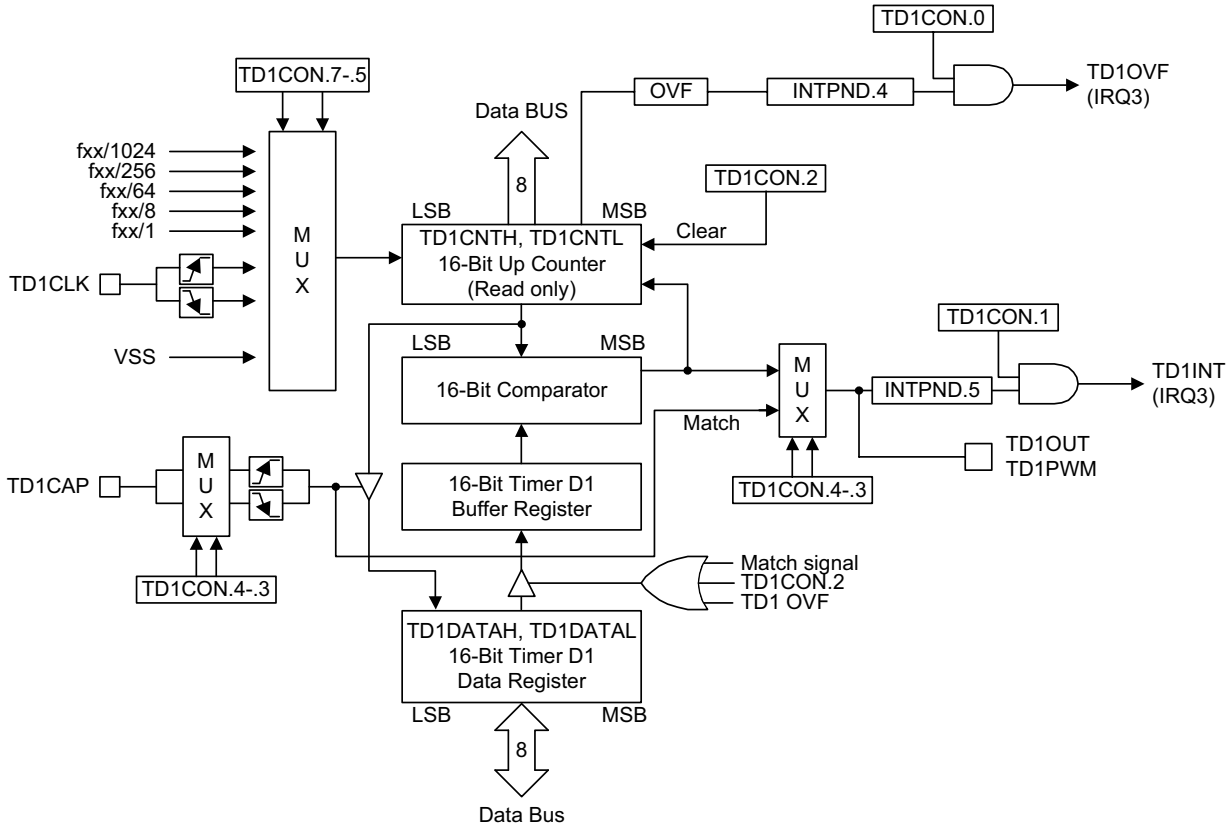


Figure 83. Timer D1 Functional Block Diagram

15.1. Timer D1 Control Register

Use the Timer D1 Control (TD1CON) Register, shown in Table 70, to perform the following tasks:

- Select the Timer D1 operating mode (Interval Timer, Capture Mode, or PWM Mode)
- Select the Timer D1 input clock frequency
- Clear the Timer D1 counter, TD1CNTH/T D1CNTL
- Enable the Timer D1 overflow interrupt or Timer D1 match/capture interrupt

TD1CON is located in Set1 and Bank1 at address FBh, and is read/write addressable using Register Addressing Mode.

A reset clears TD1CON to 00h, thereby setting Timer D1 to normal Interval Timer Mode, selecting an input clock frequency of f_{XX}/1024, and disabling all Timer D1 interrupts. To disable the counter operation, set TD1CON.7–.5 to 111b. The Timer D1 counter can be cleared at any time during normal operation by writing a 1 to TD1CON.2. The IRQ3-level Timer D1 overflow interrupt (TD1OVF) is at vector address DEh.

Table 70. Timer D1 Control Register (TD1CON; Page 4)

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

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

Bit	Description
[7:5]	**Timer D1 Input Clock Selection Bits** 000: f_{XX}/1024. 001: f_{XX}/256. 010: f_{XX}/64. 011: f_{XX}/8. 100: f_{XX}/1. 101: External clock (TD1CLK) falling edge. 110: External clock (TD1CLK) rising edge. 111: Counter stop.
[4:3]	**Timer D1 Operating Mode Selection Bits** 00: Interval Mode (TD1OUT). 01: Capture Mode (Capture on rising edge, counter running, OVF can occur). 10: Capture Mode (Capture on falling edge, counter running, OVF can occur. 11: PWM Mode (OVF and match interrupt can occur).
[2]	**Timer D1 Counter Clear Bit*** 0: No effect. 1: Clear the Timer D1 counter (when write).
[1]	**Timer D1 Match/Capture Interrupt Enable Bit** 0: Disable interrupt. 1: Enable interrupt.
[0]	**Timer D1 Overflow Interrupt Enable Bit** 0: Disable overflow interrupt. 1: Enable overflow interrupt.

Note: *The TD1CON.2 value is automatically cleared to 0 after being cleared by the counter.

When a Timer D1 overflow interrupt occurs and is serviced interrupt (IRQ3, vector DEh), write a 1 to TD1CON.0. When a Timer D1 overflow interrupt occurs and is serviced by

the CPU, the pending condition is cleared automatically by hardware or must be cleared by software.

To enable the Timer D1 match/capture interrupt (IRQ3, vector DCh), write a 1 to TD1CON.1. To detect a match/capture interrupt pending condition, the application software polls INTPND.3. When a 1 is detected, a Timer D1 match or capture interrupt is pending. When IMR.3 = 1 and TD1INT is enabled, the IRQ3 and INTPND.5 are cleared by hardware when the interrupt request has been serviced.

When IMR.3 = 0 and TD1INT is enabled, the IRQ3 and INTPND.5 are cleared by software by writing a 0 to the Timer D1 match/capture interrupt pending bit, INTPND.5.

15.2. Timer D1 Function Description

This section describes the features and functions of the Timer D1 interrupt.

15.3. Timer D1 Interrupts

The Timer D1 can generate two interrupts: the Timer D1 overflow interrupt (TD1OVF), and the Timer D1 match/capture interrupt (TD1INT). TD1OVF is associates to interrupt level IRQ3, vector DEh. TD1INT also associates to interrupt level IRQ3, but is assigned the separate vector address, DCh.

A Timer D1 overflow interrupt pending condition is automatically cleared by hardware when it has been serviced, or should be cleared by software in the interrupt service routine by writing a 0 to the INTPND.4 interrupt pending bit. When IMR.3 = 1 and TD1INT is enabled, the IRQ3 and INTPND.5 are cleared by hardware when the interrupt request has been serviced.

When IMR.3 = 0 and TD1INT is enabled, the IRQ3 and INTPND.5 are cleared by software by writing a 0 to the Timer D1 match/capture interrupt pending bit, INTPND.5.

15.4. Timer D1 Interval Timer Mode

In Interval Timer Mode, a match signal is generated when the counter value is identical to the value written to the Timer D1 Reference Data Register, TD1DATAH/TD1DATAL. The match signal generates a Timer D1 match interrupt (TD1INT, vector DCh) and clears the counter.

If, for example, you write the value 1087h to TD1DATAH/TD1DATAL, the counter will increment until it reaches 1087h. At this point, the Timer D1 interrupt request is generated, the counter value is reset, and counting resumes. With each match, the level of the signal at the Timer D1 output pin is inverted; see Figure 84.

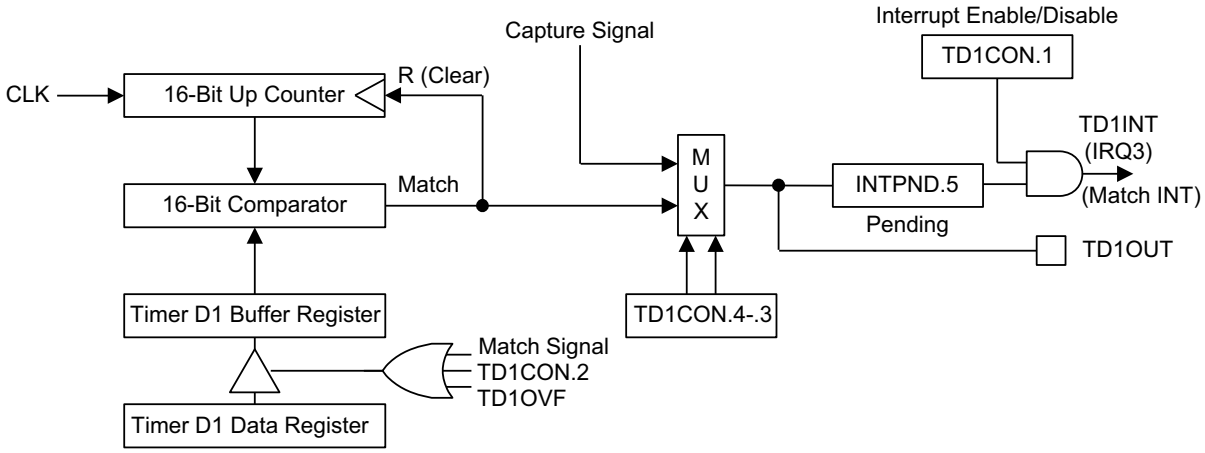


Figure 84. Simplified Timer D1 Function Diagram: Interval Timer Mode

15.5. Timer D1 Pulse Width Modulation Mode

Pulse Width Modulation (PWM) Mode lets you program the width (duration) of the pulse that is output at the TD1PWM pin. As in Interval Timer Mode, a match signal is generated when the counter value is identical to the value written to the Timer D1 Data Register. In PWM Mode, however, the match signal does not clear the counter. Instead, it runs continuously, overflowing at FFFFh, then continues incrementing from 0000h.

Although you can use the match signal to generate a Timer D1 overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the TD1PWM pin is held Low as long as the reference data value is less than or equal to (≤) the counter value; the pulse is then held High for as long as the data value is greater than (>) the counter value. One pulse width is equal to t_{CLK} x 65536; see Figure 85.

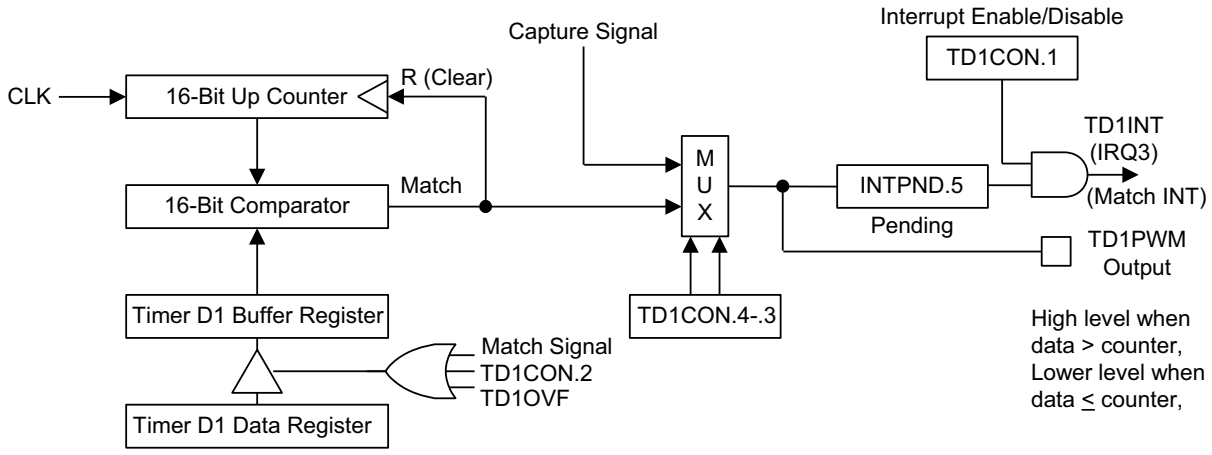


Figure 85. Simplified Timer D1 Function Diagram: PWM Mode

15.6. Timer D1 Capture Mode

In Capture Mode, a signal edge that is detected at the TD1CAP pin opens a gate and loads the current counter value into the Timer D1 Data Register. Select rising or falling edges to trigger this operation.

Timer D1 also provides a capture input source: the signal edge at the TD1CAP pin. Select the capture input by setting the values of the Timer D1 capture input selection bits in the Port 1 Control Register, P1CONH.3–.2 (Set1, Bank1, E0h). When P1CONH.3–.2 is 00, the TD1CAP input is selected.

Both kinds of Timer D1 interrupts can be used in Capture Mode: the Timer D1 overflow interrupt is generated whenever a counter overflow occurs, and the Timer D1 match/capture interrupt is generated whenever the counter value is loaded into the Timer D1 Data Register.

By reading the captured data value in TD1DATAH/TD1DATAL, and assuming a specific value for the Timer D1 clock frequency, you can calculate the pulse width (duration) of the signal that is being input at the TD1CAP pin; see Figure 86.

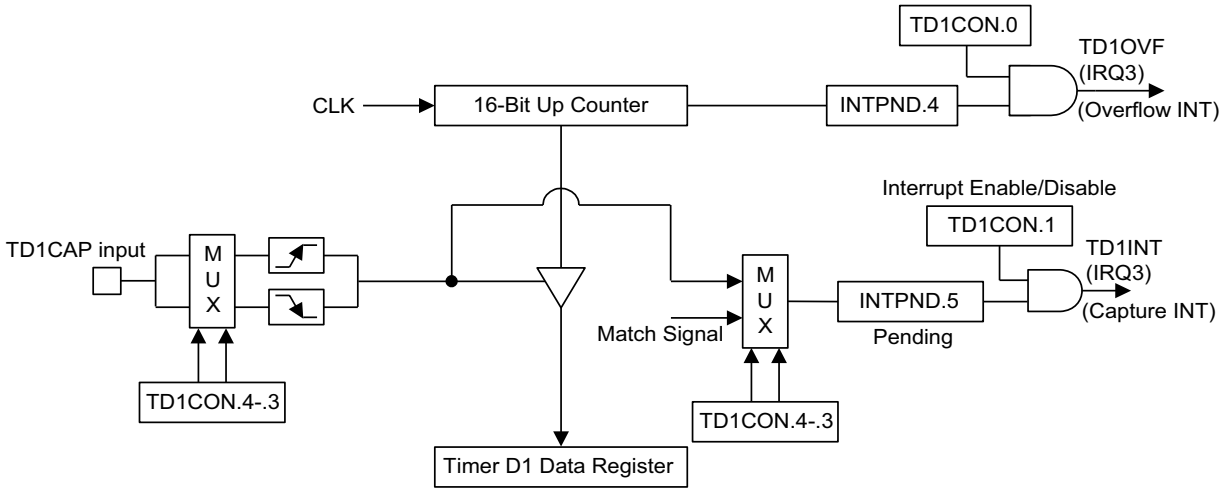


Figure 86. Simplified Timer D1 Function Diagram: Capture Mode

Chapter 16. Watch Timer

The watch timer functions include programmable periodic interrupts, programmable frequency buzzer output, and a clock source for the LCD controller. The time base input for the Watch Timer can be set to either the main clock divided by 128 or the 32.768kHz subclock.

To begin watch timer operation, set bit 1 of the Watch Timer Control Register, WTCON.1, to 1. To enable the watch timer overflow interrupt (IRQ4, vector E6h), set WTCON.6 to 1. After the watch timer starts and the time selected by WTCON .3–.2 elapses, the watch timer interrupt pending bit (WTCON.0) is automatically set to 1, and interrupt requests continue to occur in 3.91ms, 0.25s, 0.5s, and 1.0s intervals. The watch timer overflow interrupt pending condition, WTCON.0, must be cleared by software in the application's interrupt service routine by writing a 0 to the WTCON.0 interrupt pending bit.

The watch timer can also generate a programmable frequency for the BUZ output pin for a buzzer function. WTCON.3 and WTCON.2 must be set to 11b by selecting 3.91ms; the output frequency can then be selected from 0.5kHz, 1kHz, 2kHz, or 4kHz with WTCON.4 and .5.

The watch timer supplies the clock frequency for the LCD controller, f_{LCD}. Therefore, if the watch timer is disabled, the LCD controller does not operate.

The watch timer provides the following features:

- Periodic interrupt generation (3.91ms, 0.25s, 0.5s, or 1.0s)
- Selectable main clock or subclock for time base
- Provides clock source generation for the LCD controller (f_{LCD})
- I/O pin for buzzer output frequency generator (BUZ)
- Watch timer overflow interrupt (IRQ4, vector E6h) generation
- Watch timer control register, WTCON (Set1, Bank1, E6h, read/write)

Figure 87 presents a block diagram of the Watch Timer circuit.

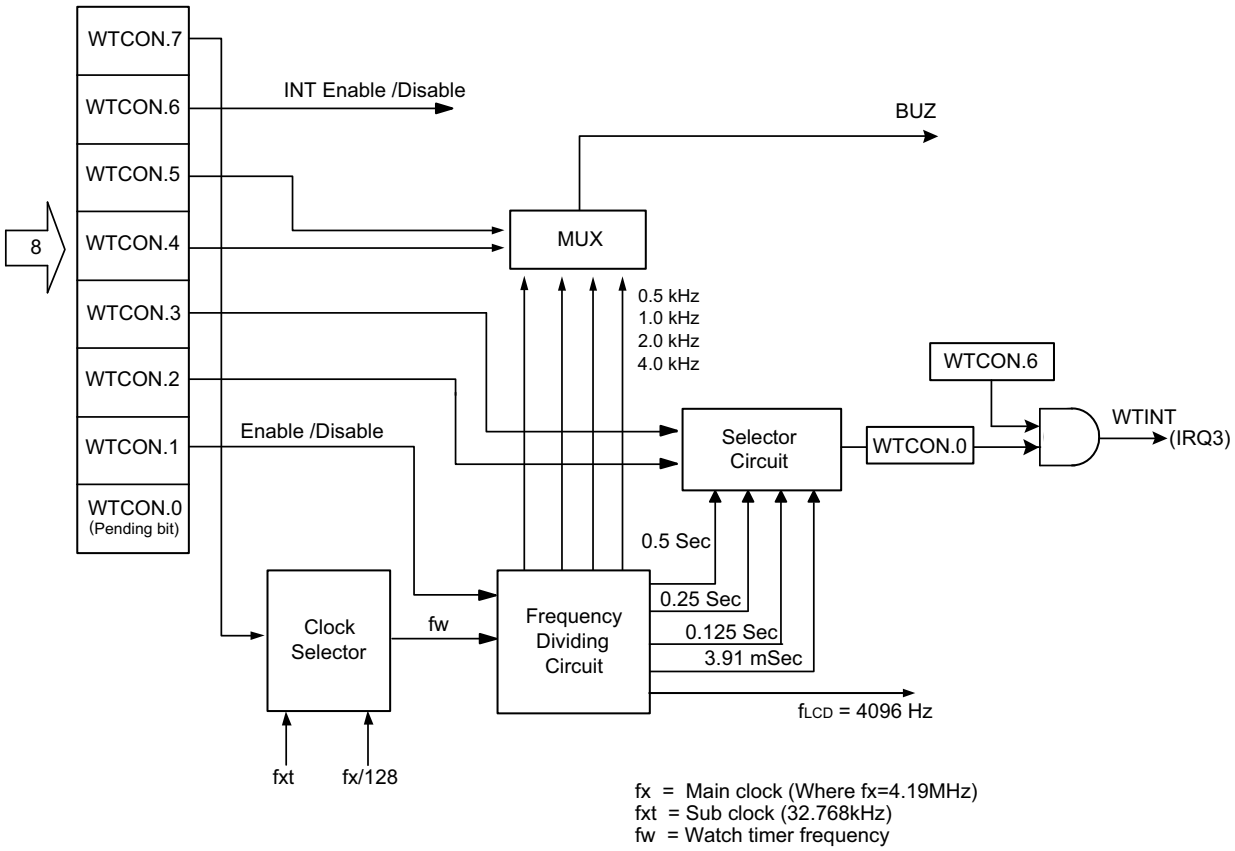


Figure 87. Watch Timer Circuit Block Diagram

16.1. Watch Timer Control Register

The Watch Timer Control (WTCON) Register, shown in Table 71, is used to select the watch timer interrupt time and buzzer signal to enable or disable the watch timer function. It is located in Set1, Bank1 at address FEh, and is read/write addressable using Register Addressing Mode.

Table 71. Watch Timer Control Register (WTCON; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	E6h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7]	**Watch Timer Clock Selection Bit** 0: Main system clock divided by 128 (f_X/128). 1: Subsystem clock (f_{XT}).
[6]	**Watch Timer Interrupt Enable Bit** 0: Disable watch timer interrupt. 1: Enable watch timer interrupt.
[5:4]	**Buzzer Signal Selection Bits** 00: 0.5kHz. 01: 1kHz. 10: 2kHz. 11: 4kHz.
[3:2]	**Watch Timer Speed Selection Bits** 00: Set watch timer interrupt to 1.0s. 01: Set watch timer interrupt to 0.5s. 10: Set watch timer interrupt to 0.25s. 11: Set watch timer interrupt to 3.91ms.
[1]	**Watch Timer Enable Bit** 0: Disable watch timer; clear frequency dividing circuits. 1: Enable watch timer.
[0]	**Watch Timer Interrupt Pending Bit** 0: No interrupt pending (when read), clear pending bit (when write). 1: Interrupt is pending (when read).

Note: Watch timer clock frequency (f_W) is assumed to be 32.768kHz.

A reset clears WTCON to `00h`, thereby disabling the watch timer. Therefore, to use the watch timer, write the appropriate value to WTCON.

Chapter 17. LCD Controller/Driver

The S3F8S45 microcontroller can directly drive a 22-segment x 8-common LCD panel up to 176 dots. Its LCD block features the following components:

- LCD controller/driver
- Display RAM (`00h–15h`) for storing display data in Page 1
- 22 segment output pins (SEG0–SEG21)
- 8 common output pins (COM0–COM7)
- LCD bias by voltage-dividing resistors

The LCD Control (LCON) Register is used to turn the LCD display on and off and to select the frame frequency, LCD duty, and bias. Data written to LCD display RAM can be automatically transferred to the segment signal pins without any program control. When a subsystem clock is selected as the LCD clock source, the LCD display is enabled even in the main clock stop or idle modes.

LCD data stored in display RAM locations are transferred to the segment signal pins automatically without program control.

Figure 88 presents a block diagram of the LCD function.

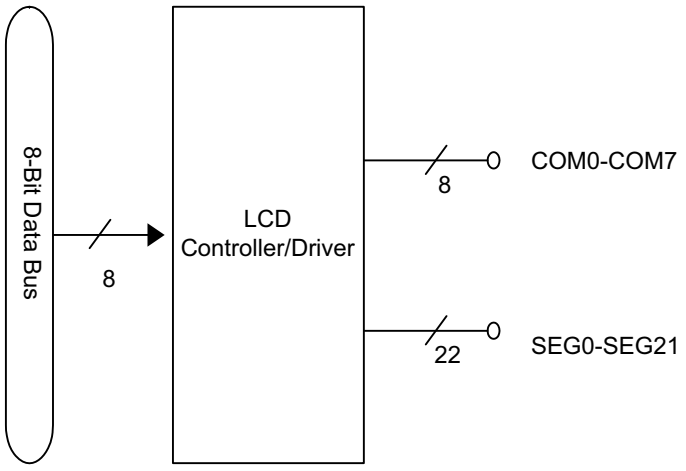


Figure 88. LCD Function Diagram

Figure 89 presents a diagram of the LCD circuit.

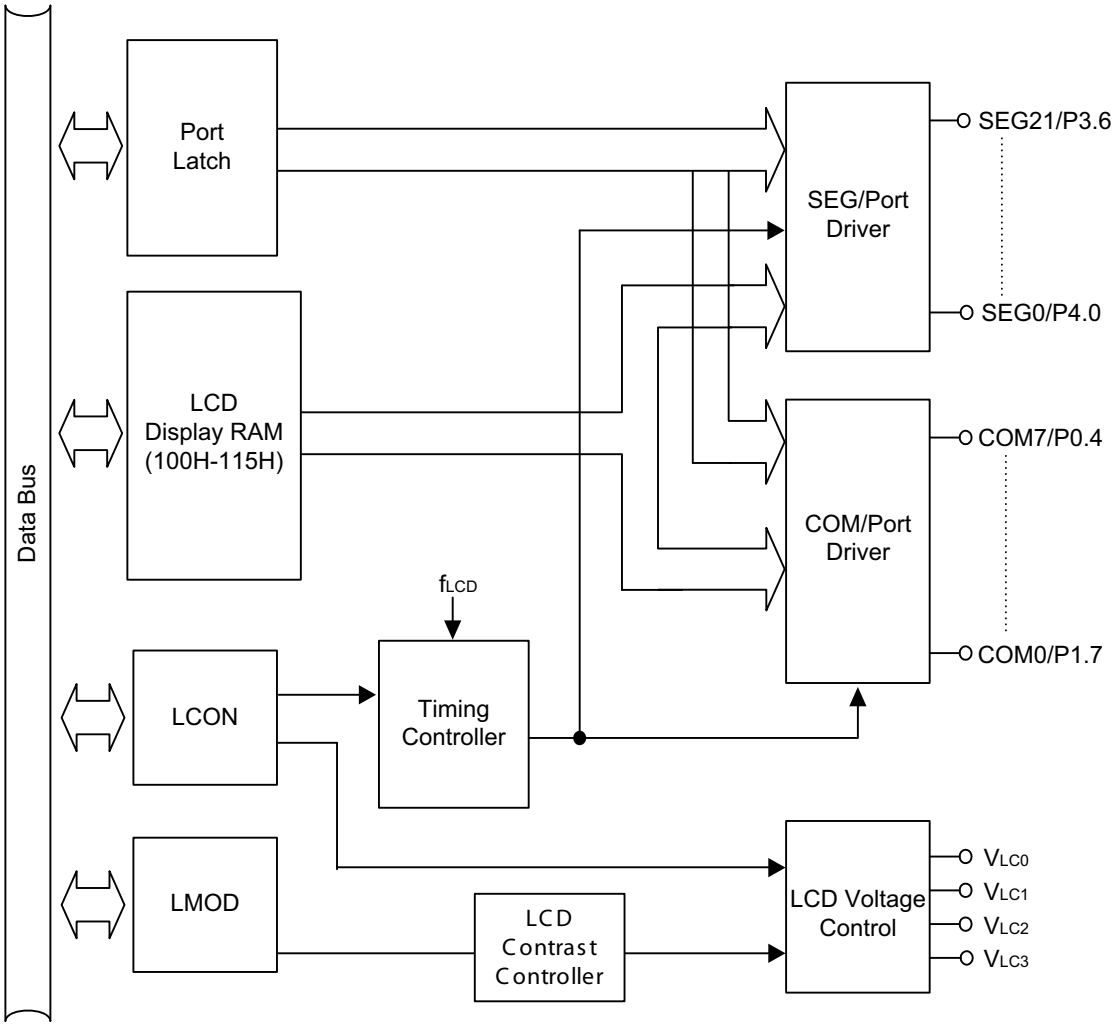


Figure 89. LCD Circuit

17.1. LCD RAM Address Area

RAM addresses of `00h–15h` in Page 1 are used as LCD data memory. These locations can be addressed by 1-bit or 8-bit instructions. When the bit value of a display segment is 1, the LCD display is turned on; when the bit value is 0, the display is turned off.

Display RAM data are sent out through the segment pins, SEG0–SEG21, using a direct memory access (DMA) method that is synchronized with the f_{LCD} signal. RAM addresses

in this location that are not used for LCD display can be allocated to general-purpose use; see Table 72.

Table 72. LCD Display Data RAM Organization

COM	Bit	SEG0	SEG1	SEG2	SEG3	SEG4	–	SEG20	SEG21
COM0	.0	100h	101h	102h	103h	104h	–	114h	115h
COM1	.1								
COM2	.2								
COM3	.3								
COM4	.4								
COM5	.5								
COM6	.6								
COM7	.7								

17.2. LCD Control Register

The LCD Control (LCON) Register, shown in Table 73, is located in Set1, Bank1 at address F9h, and is read/write addressable using Register Addressing Mode. This register features the following control functions:

- LCD duty and bias selection
- LCD clock selection
- LCD display control

The LCON Register is used to turn the LCD display on/off, to select duty and bias, to select the LCD clock. A reset clears the LCON Register to 00h, thereby turning off the LCD display, selecting 1/8 duty and 1/4 bias, and selecting 128Hz for the LCD clock.

The LCD clock signal determines the frequency of the COM signal scan of each segment output; also referred to as the *LCD frame frequency*. Because the LCD clock is generated by the watch timer clock (f_W), the watch timer should be enabled when the LCD display is turned on.

Note: The clock and duty for the LCD controller/driver is automatically initialized by hardware whenever the LCON Register data value is rewritten. Therefore, the LCON Register does not rewrite frequently.

Table 73. LCD Control Register (LCON; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F9h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:5]	**LCD Clock Selection Bits** 000: $f_W/2^8$ (128Hz). 001: $f_W/2^7$ (256Hz). 010: $f_W/2^6$ (512Hz). 011: $f_W/2^5$ (1024Hz). 100: $f_W/2^4$ (2048Hz). 101–111: Not available.
[4:2]	**LCD Duty and Bias Selection Bits** 000: 1/8 duty, 1/4 bias. 001: 1/4 duty, 1/3 bias. 010: 1/3 duty, 1/3 bias. 011: 1/3 duty, 1/2 bias. 1xx: 1/2 duty, 1/2 bias.
[1:0]	**LCD Display Control Bits** 00: Display off. 01: All dots off signal. 10: All dots on signal. 11: Turn display on.

Note: The clock and duty cycle for the LCD controller/driver are automatically initialized by hardware whenever the LCON Register data value is rewritten. As a result, the LCON Register does not rewrite frequently.

17.3. LCD Mode Control Register

The LCD Mode Control (LMOD) Register, shown in Table 74, is located in Set1, Bank1 at address FAh, and is read/write addressable using Register Addressing Mode. This register features the following control functions:

- LCD contrast level selection
- LCD contrast controller enable/disable
- LCD dividing resistors selection

The LMOD Register is used to select contrast level, to enable or disable the contrast controller, and to select dividing resistors. A reset clears the LMOD Register to 00h, thereby selecting a 1/16 step contrast level and disabling the contrast controller.

Table 74. LCD Mode Control Register (LMOD; Set1, Bank1)

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

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

Bit	Description
[7:4]	**LCD Contrast Level Control Bits (16 Steps)** 0000: 1/16 step (the dimmest level). 0001: 2/16 step. 0010: 3/16 step. 0011: 4/16 step. 0100: 5/16 step. 0101: 6/16 step. 0110: 7/16 step. 0111: 8/16 step. 1000: 9/16 step. 1001: 10/16 step. 1010: 11/16 step. 1011: 12/16 step. 1100: 13/16 step. 1101: 14/16 step. 1110: 15/16 step. 1111: 16/16 step (the brightest level).
[3:1]	**Reserved**
[0]	**Enable/Disable LCD Contrast Control Bit** 0: Disable LCD contrast control. 1: Enable LCD contrast control.

Note: $V_{LCD} = V_{DD}$ x (n+17)/32 with 1/4 bias and $V_{LCD} = V_{DD}$ x (n+13)/28 with 1/3 bias, where n = 0–15.

17.4. Internal Resistor Bias

Figure 90 shows the internal bias connections.

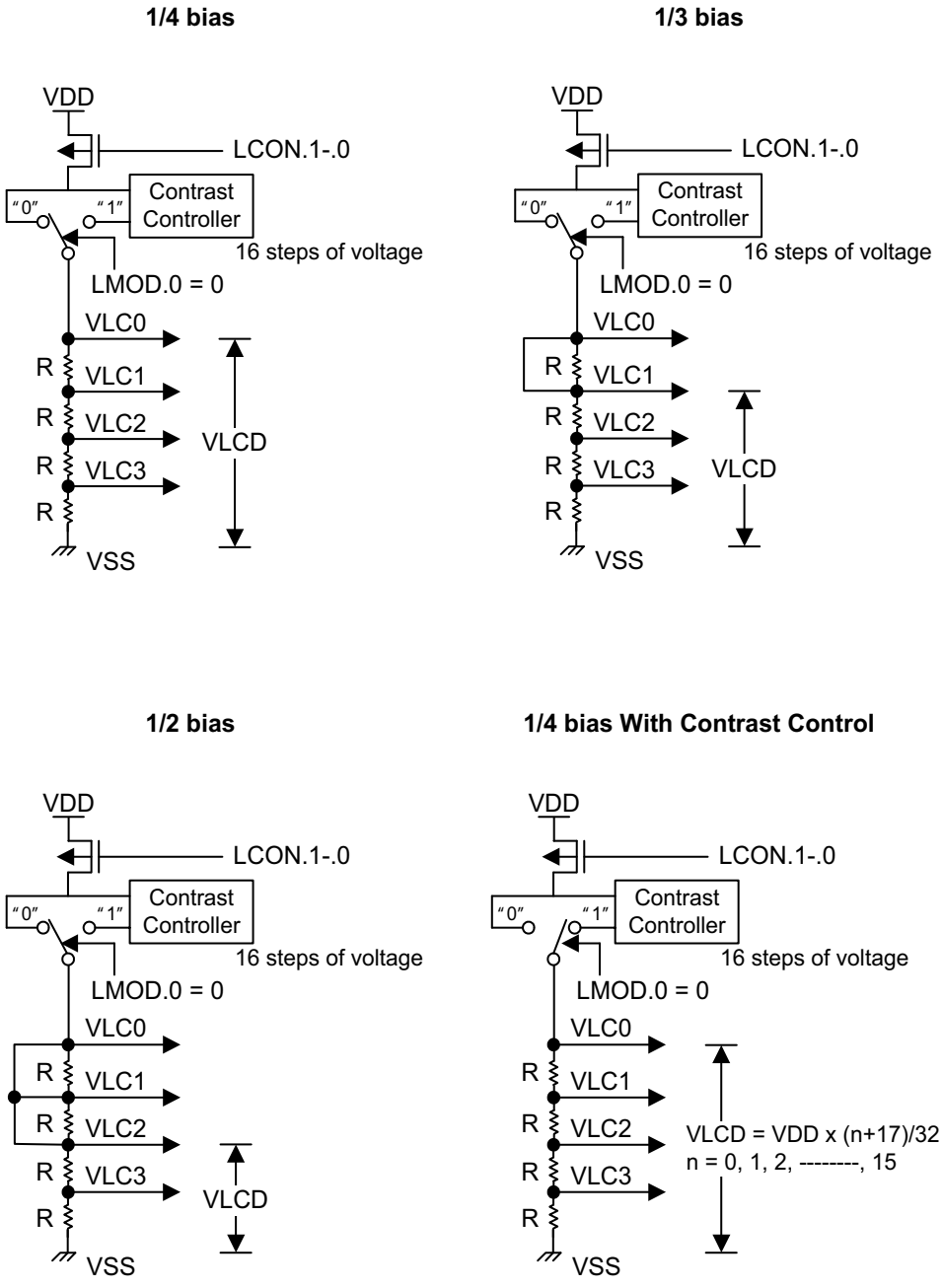


Figure 90. Internal Resistor Bias Pin Connections

17.5. Common Signals

The common signal (COM) output pin selection varies according to the selected duty cycle, as follows:

- In 1/8 Duty Mode, COM0–COM7 (SEG0–SEG21) pins are selected
- In 1/4 Duty Mode, COM0–COM3 (SEG0–SEG21) pins are selected
- In 1/3 Duty Mode, COM0–COM2 (SEG0–SEG21) pins are selected
- In 1/2 Duty Mode, COM0–COM1 (SEG0–SEG21) pins are selected

17.6. Segment Signals

The 22 LCD segment signal pins are connected to their corresponding display RAM locations at Page 3. The bits of the display RAM are synchronized with the common signal output pins.

When the bit value of a display RAM location is 1, a select signal is sent to the corresponding segment pin. When the display bit is 0, a *no-select signal* is sent to the corresponding segment pin. See Figures 91 and 92.

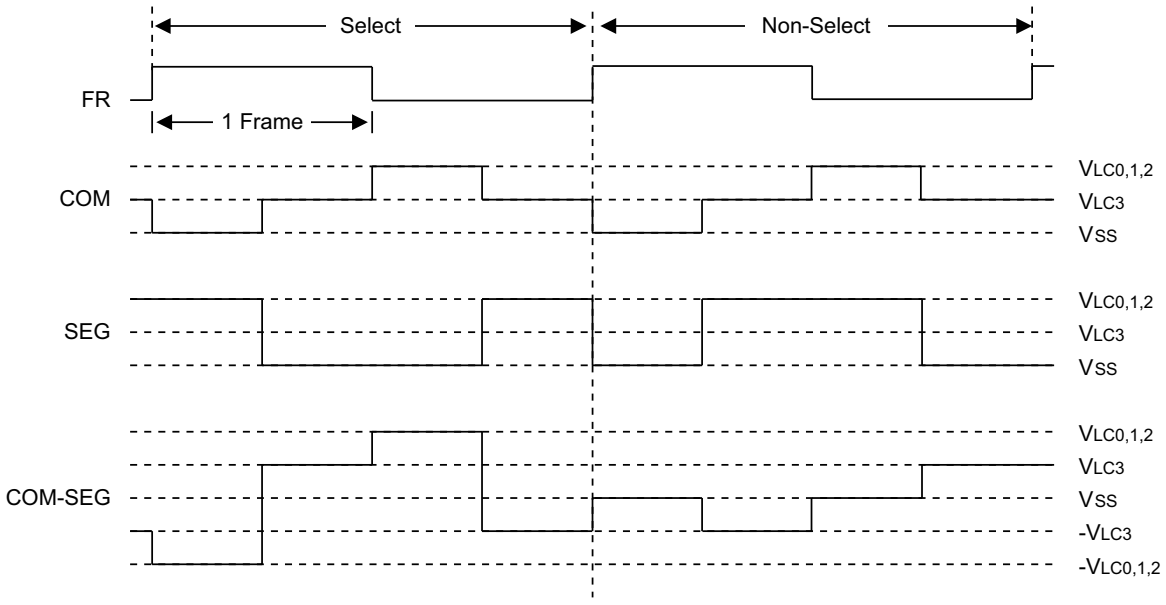


Figure 91. Select/No-Select Signal in 1/2 Duty, 1/2 Bias Display Mode

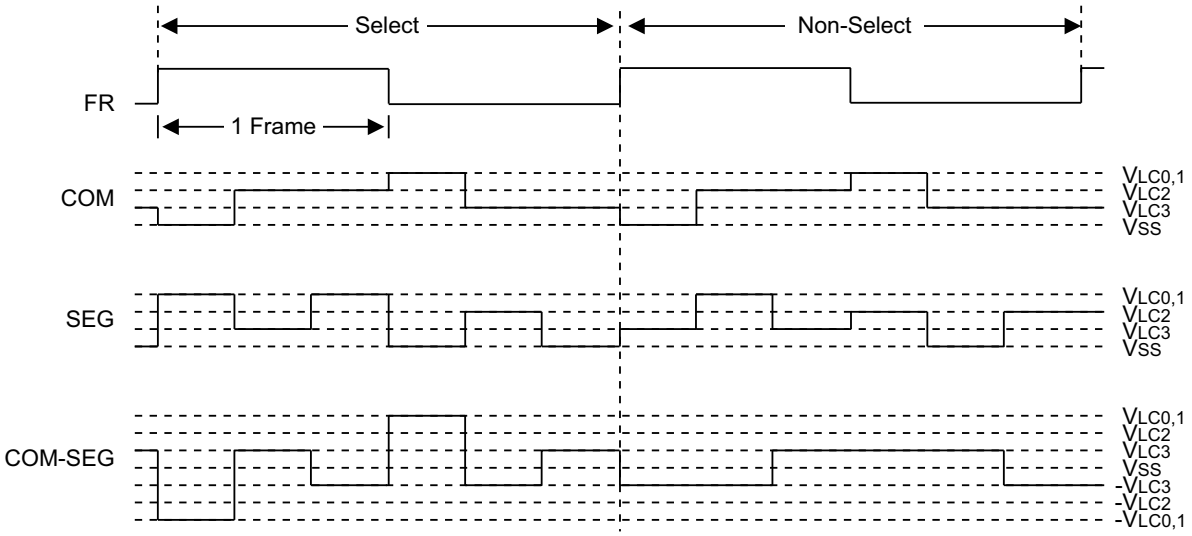


Figure 92. Select/No-Select Signal in 1/3 Duty, 1/3 Bias Display Mode

Figures 93 through 97 present waveforms of the LCD signal for differing duty and bias configurations.

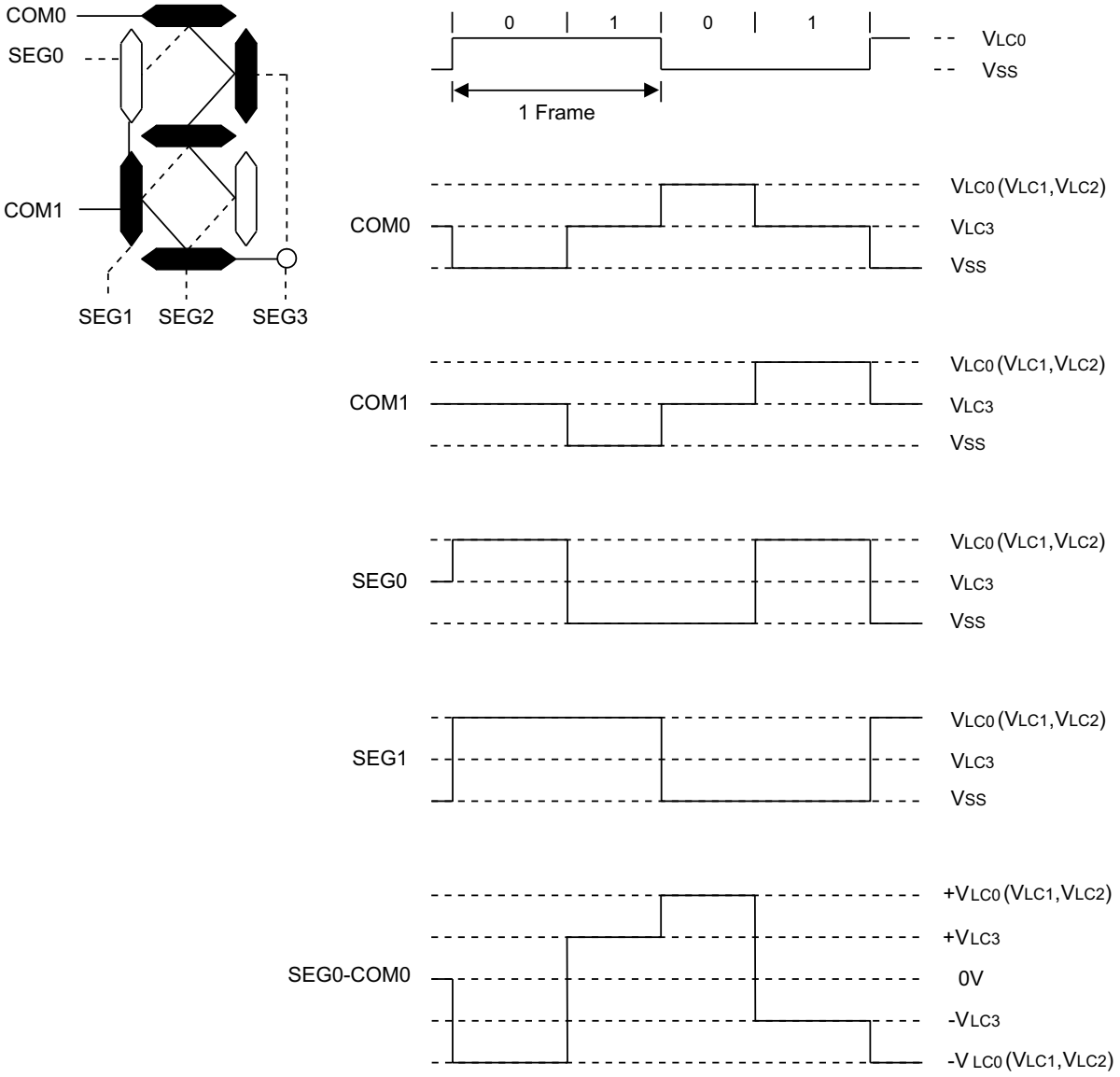


Figure 93. LCD Signal Waveforms: 1/2 Duty, 1/2 Bias

Note: In Figure 93, $V_{LC0} = V_{LC1}, V_{LC2}$.

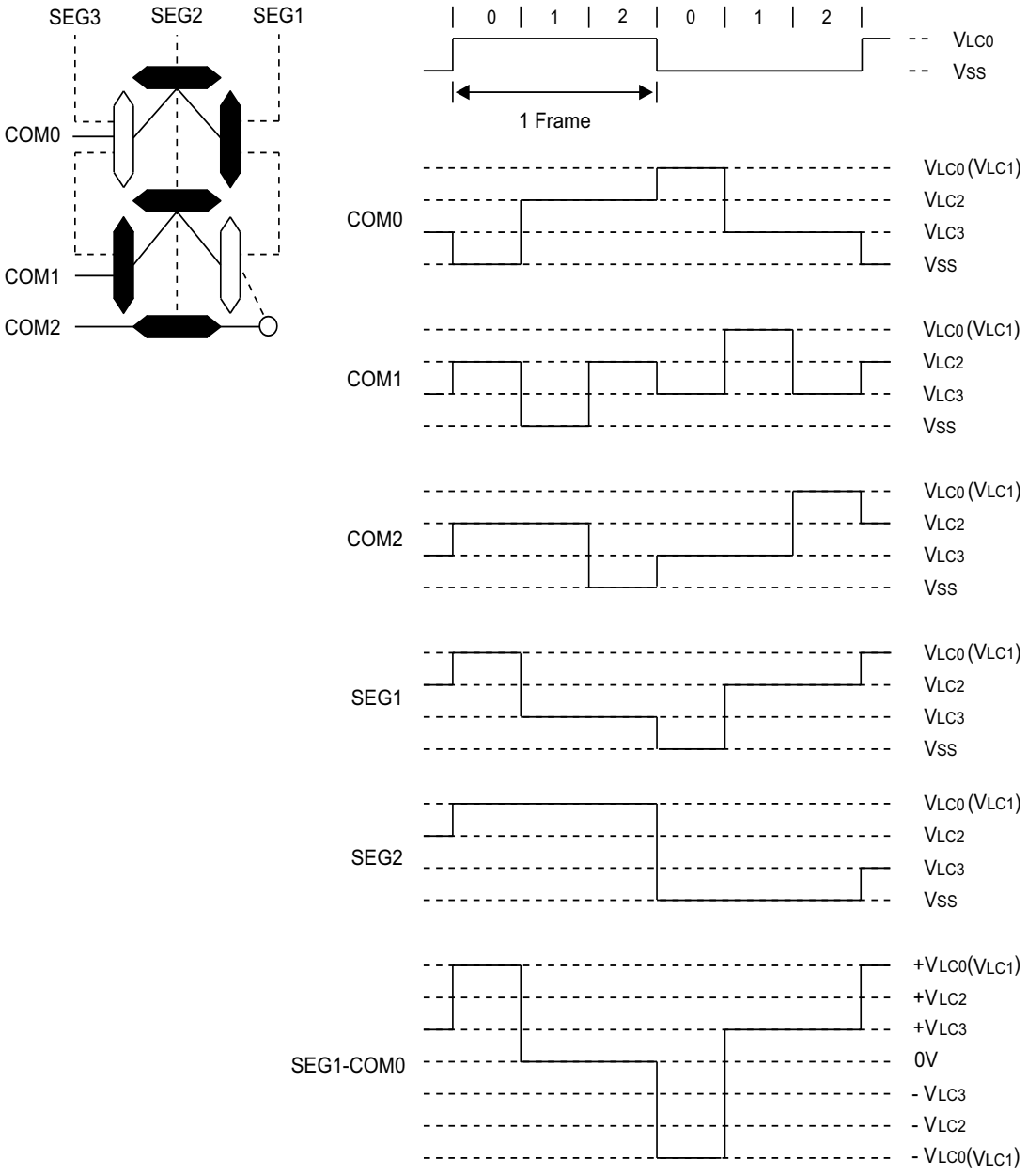


Figure 94. LCD Signal Waveforms: 1/3 Duty, 1/3 Bias

Note: In Figure 94, $V_{LC0} = V_{LC1}$.

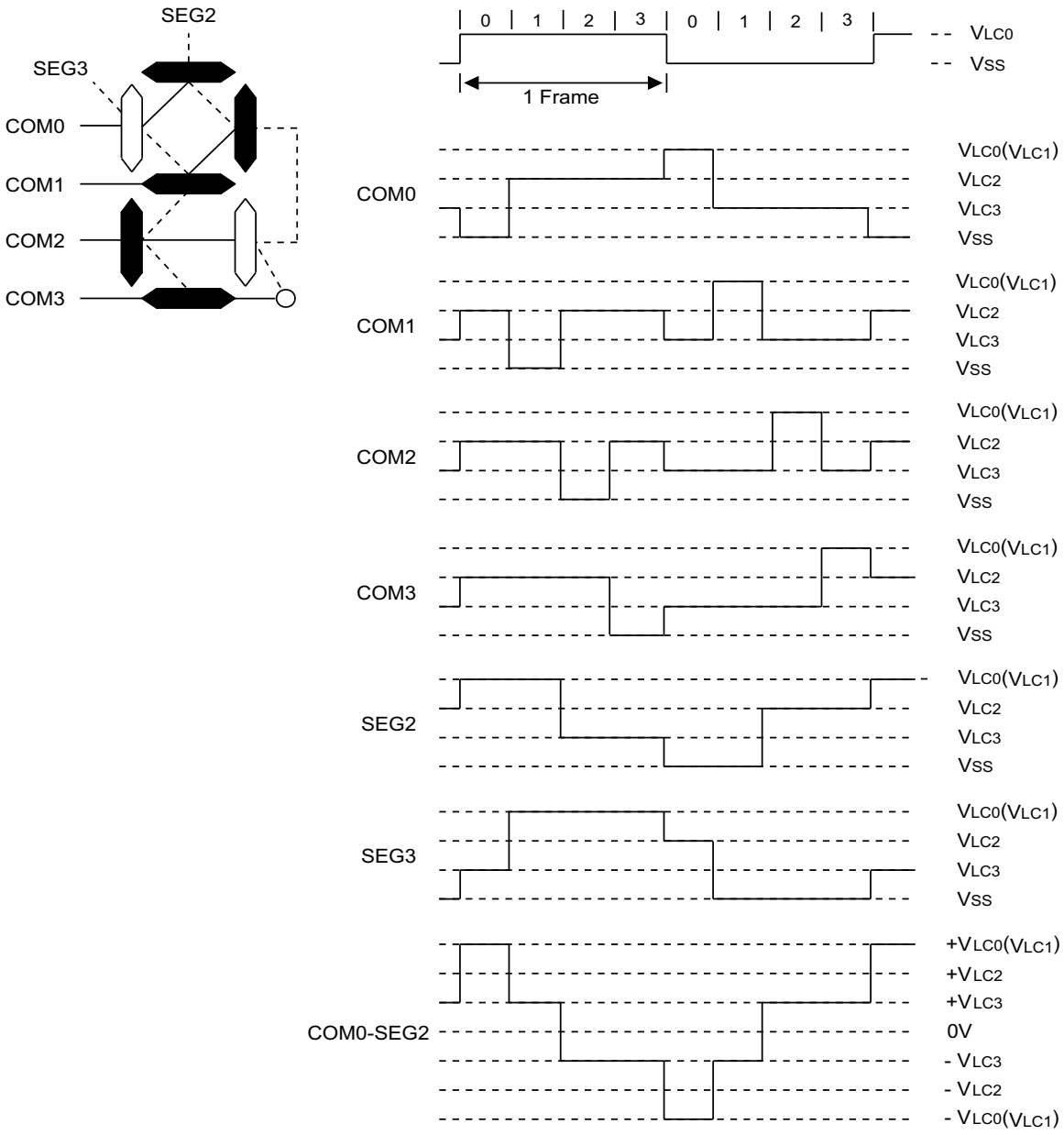


Figure 95. LCD Signal Waveforms: 1/4 Duty, 1/3 Bias

Note: In Figure 95, $V_{LC0} = V_{LC1}$.

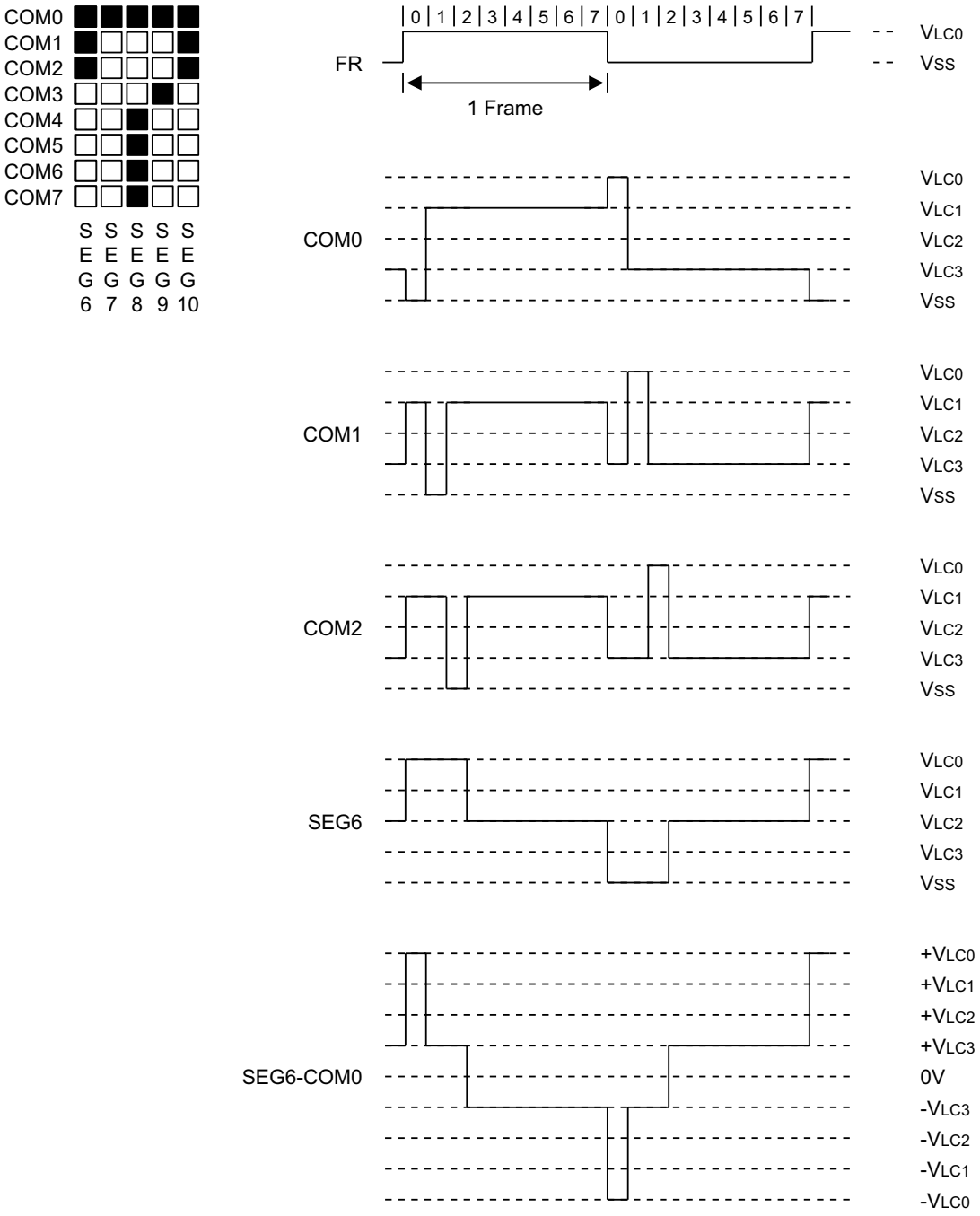


Figure 96. LCD Signal Waveforms: 1/8 Duty, 1/4 Bias, #1 of 2

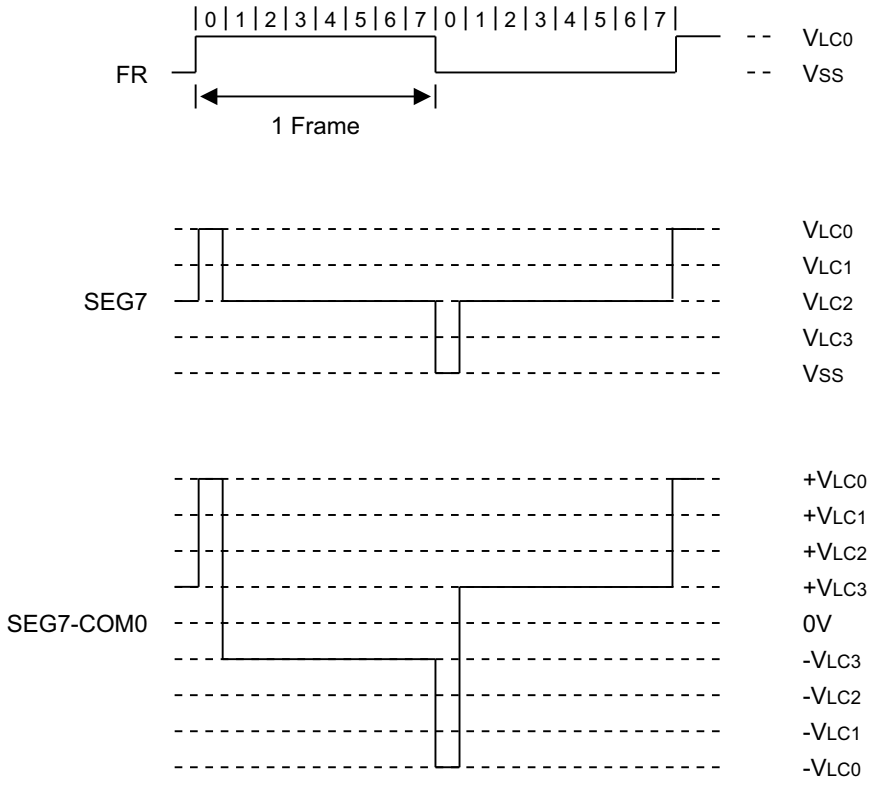


Figure 97. LCD Signal Waveforms: 1/8 Duty, 1/4 Bias, #2 of 2

Chapter 18. 10-Bit Analog-to-Digital Converter

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

The A/D converter features the following components:

- Analog comparator with successive approximation logic
- D/A converter logic (resistor string type)
- ADC Control (ADCON) Register
- Nine multiplexed analog data input pins (AD0–AD8)
- 10-bit A/D conversion data output register (ADDATAH/ADDATAL)
- 9-bit digital input port (alternatively, I/O port)
- AV_{REF} and V_{SS} pins

18.1. Function Description

To initiate an analog-to-digital conversion procedure, set the ADCEN signal for ADC input enable at Ports 0/1; the pin set with the alternative function can be used for ADC analog input. Next, write the channel selection data in ADCON.4–.7 to select one of the eight analog input pins (AD0–AD8), then set the conversion start or disable bit, ADCON.0. The read/write ADCON Register (see Table 75 on page 280) is located in Set1, Bank0 at address D2h. The pins that are not used for ADC can be used for normal I/O.

During a normal conversion, ADC logic initially sets the successive approximation register to 200h (the approximate halfway 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. Different channels can be dynamically selected by manipulating the channel selection bit value (ADCON.7–.4) in the ADCON Register. To start the A/D conversion, set the start bit, ADCON.0. When a conversion is completed, ADCON.3, the end-of-conversion (EOC) bit, is automatically set to 1, and the result is dumped into the ADDATAH/ADDATAL registers (see Tables 76 and 77), where it can be read. The A/D converter then enters an idle state. Remember to read the contents of ADDATAH/ADDATAL before another conversion starts; otherwise, the previous result will be overwritten by the next conversion result.

Note: Because the A/D converter has no sample-and-hold circuitry, it is very important that fluctuation in the analog level at the AD0–AD8 input pins during a conversion procedure be kept to an absolute minimum. Any change in the input level, perhaps due to noise, will invalidate the result. If the chip enters the STOP or IDLE modes in this conversion process, there will be a leakage current path in the A/D block. Therefore, use the STOP or IDLE modes after ADC operation is finished.

18.2. Conversion Timing

The A/D conversion process requires four steps (i.e., 4 clock edges) to convert each bit and 10 clocks to set up an A/D conversion. Therefore, total of 50 clocks are required to complete a 10-bit conversion: When $f_{XX}/8$ is selected for a conversion clock with an 8 MHz f_{XX} clock frequency, one clock cycle is 1 µs.

Each bit conversion requires 4 clocks; the conversion rate is calculated as follows:

4 clocks/bit x 10 bits + set-up time = 50 clocks

50 clocks x 1 µs = 50 µs at 1 MHz

18.3. A/D Converter Control Register

The A/D Converter Control (ADCON) Register, shown in Table 75, is located at address D2h in Set1, Bank0, and provides the following functions:

- Analog input pin selection (ADCON.7–.4)
- End-of-conversion status detection (ADCON.3)
- ADC clock selection (ADCON.2–.1)
- A/D operation start or disable (ADCON.0)

After a reset, the start bit is turned off; only one analog input channel at a time can be selected. Other analog input pins (AD0–AD8) can be selected dynamically by manipulating the ADCON.4–.7 bits; analog input pins that remain unused can be used for normal I/O functions.

Table 75. A/D Converter Control Register (ADCON; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
Address	D2h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:4]	**A/D Input Pin Selection Bits** 0000: AD0. 0001: AD1. 0010: AD2. 0011: AD3. 0100: AD4. 0101: AD5. 0110: AD6. 0111: AD7. 1000: AD8. 1001–1111: Reserved.
[3]	**End-of-Conversion Bit (Read Only)** 0: Conversion not complete. 1: Conversion complete.
[2:1]	**Clock Source Selection Bits** 00: $f_{XX}/16$. 01: $f_{XX}/8$. 10: $f_{XX}/4$. 11: $f_{XX}/1$.
[0]	**Start or Enable Bit** 0: Disable operation. 1: Start operation.

18.4. A/D Converter Data Registers

The contents of the A/D Converter Data High and Low Byte (ADDATAH/ADDATAL) registers are shown in Tables 76 and 77.

Table 76. A/D Converter Data High Byte Register (ADDATAH; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	D0h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Table 77. A/D Converter Control Register (ADDATAL; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	D1h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

18.5. Internal Reference Voltage Levels

In the ADC function block, the analog input voltage level is compared to the reference voltage. This voltage level must remain within the V_{SS}-to-AV_{REF} range; usually, $AV_{REF} \leq V_{DD}$.

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

Figure 98 presents a block diagram of the A/D Converter function.

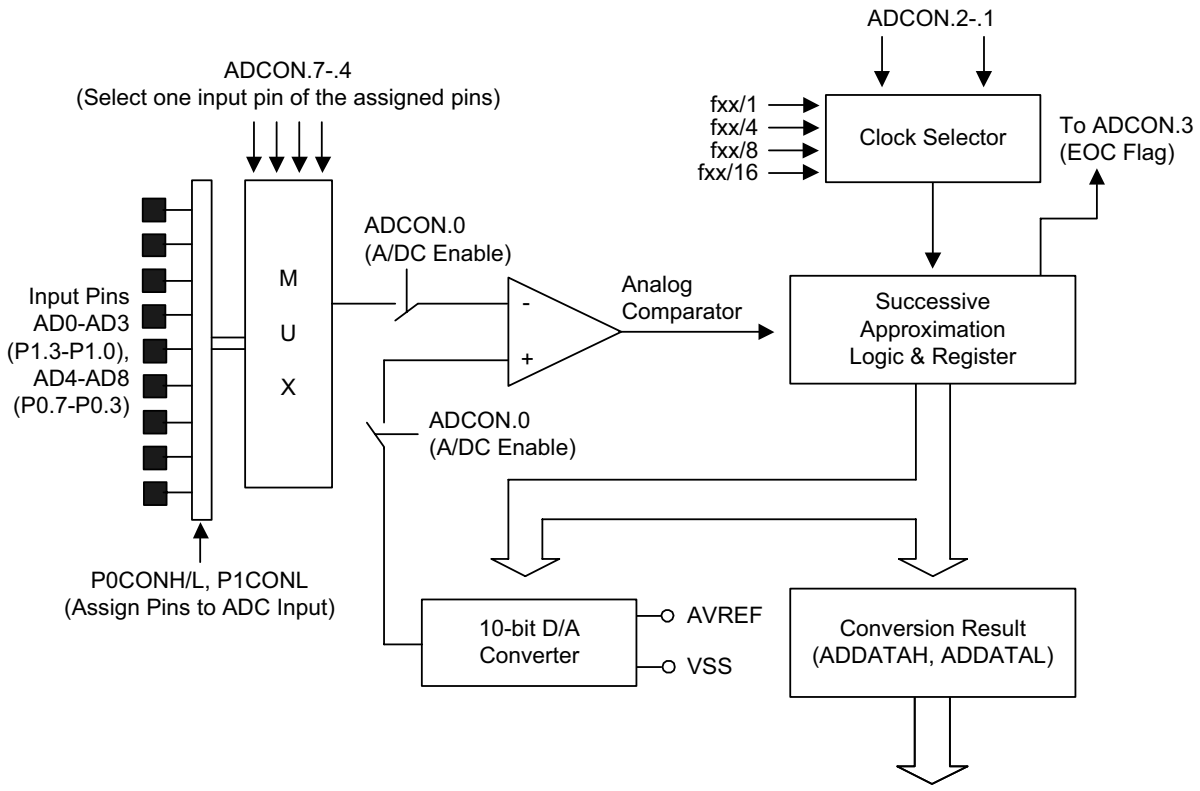


Figure 98. A/D Converter Functional Block Diagram

Figure 99 presents a diagram of the recommended A/D Converter circuit.

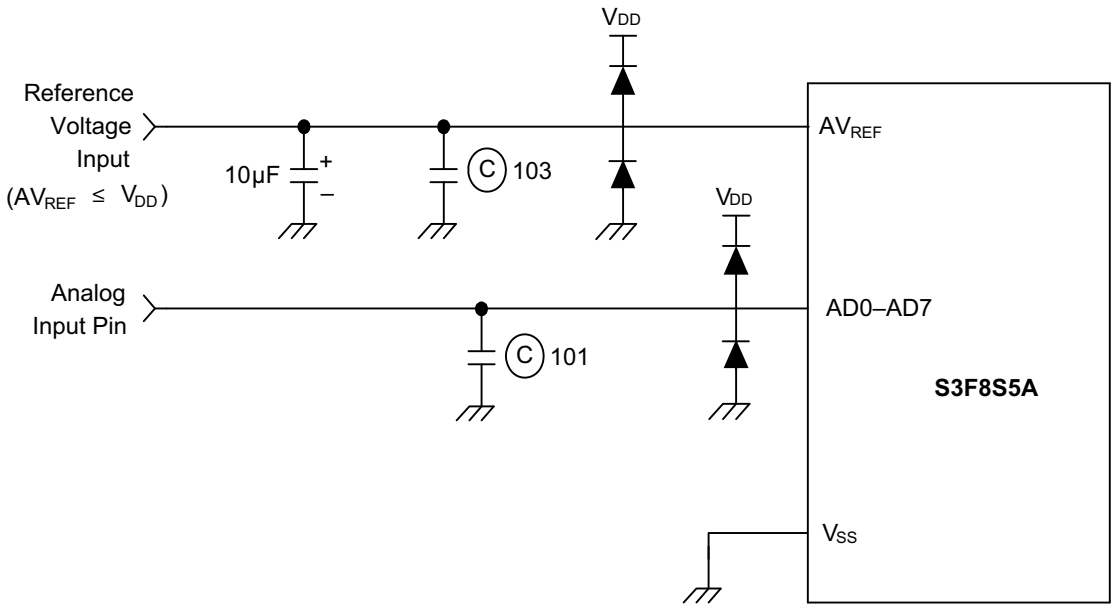


Figure 99. Recommended A/D Converter Circuit for Highest Absolute Accuracy

Chapter 19. Serial I/O Interface

The serial I/O (SIO) module can interface with various types of external devices that require serial data transfers. The components of each SIO function block are:

- 8-bit control register (SIOCON)
- Clock selector logic
- 8-bit data buffer (SIODATA)
- 8-bit prescaler (SIOPS)
- 3-bit serial clock counter
- Serial data I/O pins (SI, SO)
- Serial clock input/output pins (SCK)

The SIO module can transmit or receive 8-bit serial data at a frequency determined by its corresponding control register settings. To ensure flexible data transmission rates, you can select an internal or external clock source.

19.1. Programming Procedure

To program the SIO modules, observe the following basic steps:

1. Configure the I/O pins at port (SO, SCK, SI) by loading the appropriate value to the P4CONL Register, if necessary.
2. Load an 8-bit value to the SIOCON Control Register to properly configure the serial I/O module. In this operation, SIOCON.2 must be set to 1 to enable the data shifter.
3. For interrupt generation, set the serial I/O interrupt enable bit (SIOCON.1) to 1.
4. When transmitting data to the serial buffer, write data to SIODATA and set SIOCON.3 to 1; the shift operation starts.
5. When the shift operation (transmit/receive) is completed, the SIO pending bit (SIOCON.0) is set to 1 and an SIO interrupt request is generated.

19.2. SIO Control Register

The control register for the serial I/O interface module, SIOCON, shown in Table 78, is located at E7h in Set1, Bank0. This register provides the following control settings for the SIO module:

- Clock source selection (internal or external) for shift clock
- Interrupt enable
- Edge selection for shift operation
- Clear 3-bit counter and start shift operation
- Shift operation (transmit) enable
- Mode selection (transmit/receive or receive-only)
- Data direction selection (MSB-first or LSB-first)

Table 78. SIO Control Register (SIOCON; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	E7h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7]	**SIO Shift Clock Selection Bit** 0: Internal clock (P.S clock). 1: External clock (SCK).
[6]	**Data Direction Control Bit** 0: MSB-First Mode. 1: LSB-First Mode.
[5]	**SIO Mode Selection Bit** 0: Receive-Only Mode. 1: Transmit/Receive Mode.
[4]	**Shift Clock Edge Selection Bit** 0: Tx at falling edges; Rx at rising edges. 1: Tx at rising edges, Rx at falling edges.
[3]	**SIO Counter Clear and Shift Start Bit** 0: No action. 1: Clear 3-bit counter and start shifting.
[2]	**SIO Shift Operation Enable Bit** 0: Disable shifter and clock counter. 1: Enable shifter and clock counter.

Bit	Description (Continued)
[1]	**SIO Interrupt Enable Bit** 0: Disable SIO interrupt. 1: Enable SIO interrupt.
[0]	**SIO Interrupt Pending Bit** 0: No interrupt is pending when read; clear pending condition when write. 1: Interrupt is pending.

A reset clears the SIOCON value to 00h, thereby configuring the corresponding module with an internal clock source at the SCK, selecting a receive-only operating mode, and clearing the 3-bit counter. The data shift operation and the interrupt are disabled. The selected data direction is MSB-first.

19.3. SIO Prescaler Register

The control register for the serial I/O interface module, SIOPS, is located at E9h in Set1, Bank0. The value stored in this SIOPS Register lets you determine the SIO clock rate (baud rate) as follows:

Baud rate = Input clock (f_{XX}/4)/(Prescaler value + 1), or SCK input clock, in which the input clock is f_{XX}/4

The contents of the SIO Prescaler (SIOPS) Register are shown in Table 79.

Table 79. SIO Prescaler Register (SIOPS; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	E9h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Bit	Description
[7:0]	**SIO Prescaler Configuration Bits** Baud rate = (f_{XX} /4)/(SIOPS+1).

Figure 100 presents a block diagram of the SIO function.

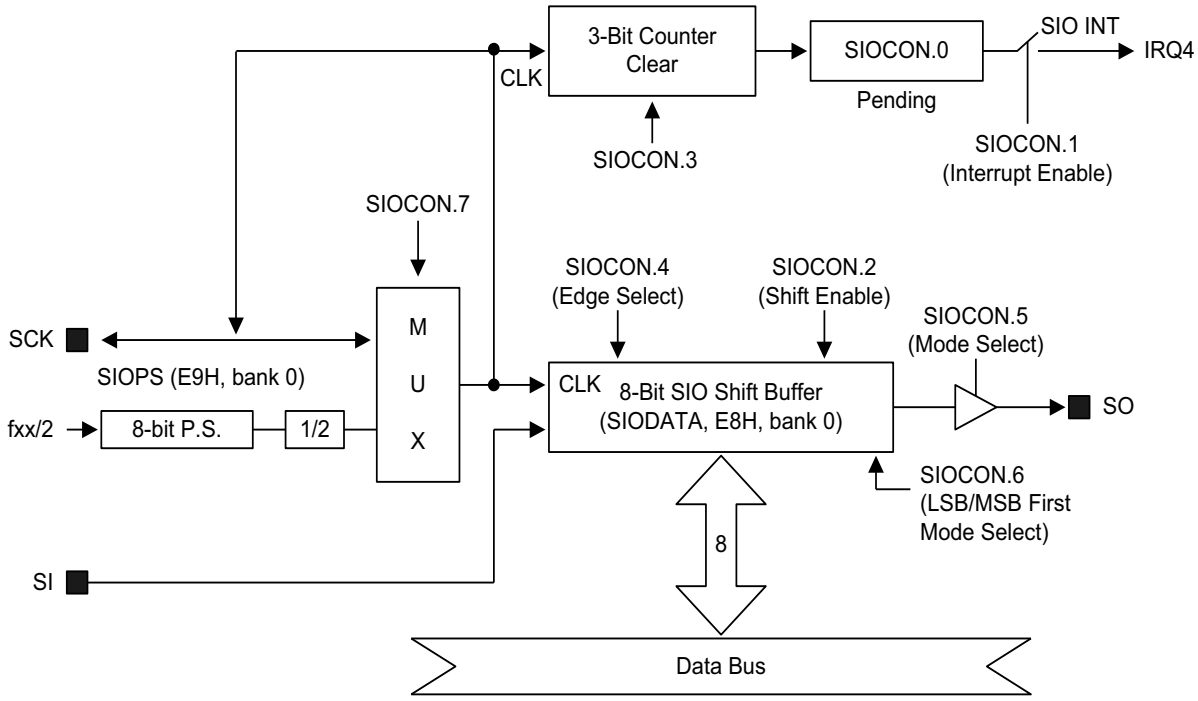


Figure 100. SIO Functional Block Diagram

19.4. SIO Serial Timing

Figure 101 presents a timing diagrams for Transmit/Receive Mode operation at the falling edge.

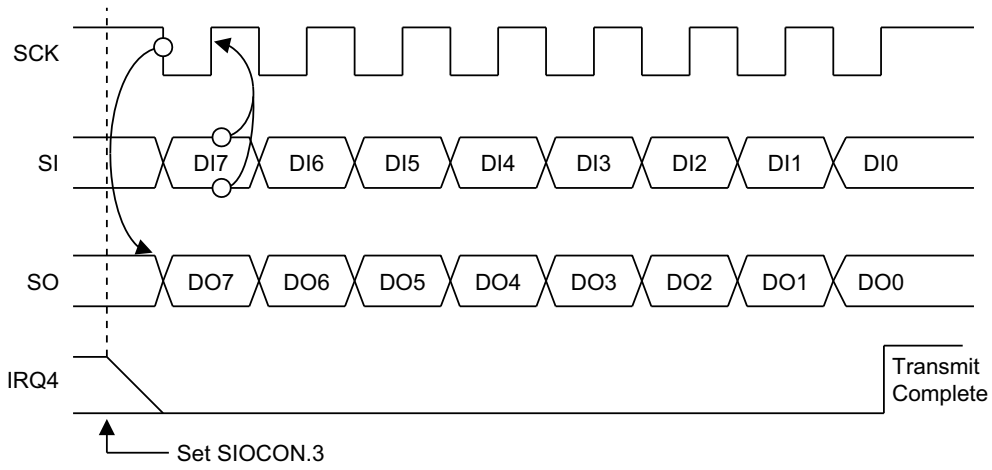


Figure 101. Serial I/O Timing in Transmit/Receive Mode (T_X at Falling Edge; SIOCON.4 = 0)

Figure 102 presents a timing diagrams for Transmit/Receive Mode operation at the falling edge.

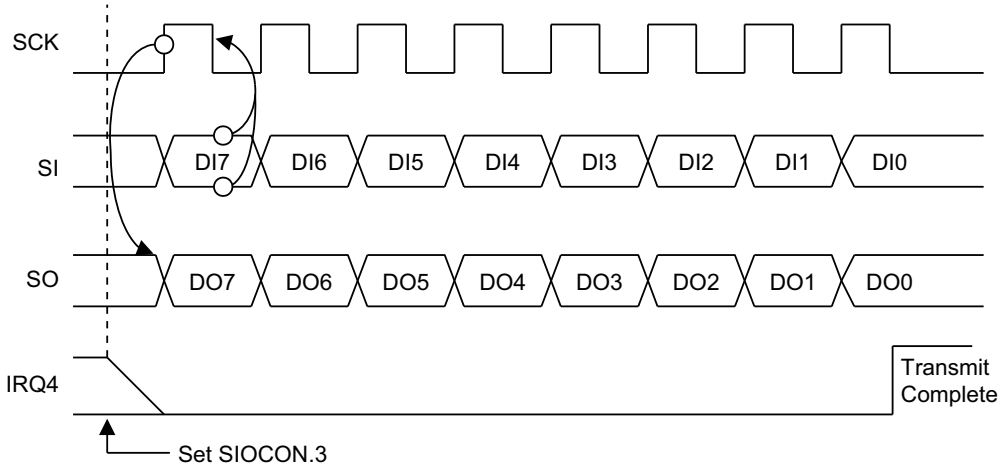


Figure 102. Serial I/O Timing in Transmit/Receive Mode (T_X at at Rising Edge; SIOCON.4 = 1)

Chapter 20. UART0

The UART0 block features a full-duplex serial port with the following four programmable operating modes – one synchronous mode and three Universal Asynchronous Receiver/ Transmitter (UART) modes:

- Serial I/O with baud rate of f_U/(16 x (BRDATA0+1))
- 8-bit UART Mode; variable baud rate
- 9-bit UART Mode; f_U/16
- 9-bit UART Mode, variable baud rate

The UART0 receive and transmit buffers are both accessed via the UDATA0 Data Register, and are located in Set1, Bank0 at address F0h. Writing to the UART data register loads the transmit buffer; reading the UART0 data register accesses a physically separate receive buffer.

When accessing a receive data buffer (i.e., the shift register), reception of the next byte can begin before the previously received byte has been read from the receive register. However, if the first byte has not been read by the time the next byte has been completely received, one of the bytes will be lost.

In all operating modes, transmission begins when any instruction (usually a write operation) uses the UDATA0 Register as its destination address. In Mode 0, serial data reception starts when the receive interrupt pending bit (UART0CONH.0) is 0 and the receive enable bit (UART0CONH.4) is 1. In modes 1, 2, and 3, reception starts whenever an incoming start bit (0) is received and the receive enable bit (UART0CONH.4) is set to 1.

20.1. Programming Procedure

To program the UART0 modules, observe the following basic steps:

1. Configure P3.1 and P3.2 to alternative function (RxD0 (P3.1), TxD0 (P3.2)) for the UART0 module by setting the P3CONL Register to the appropriate value.
2. Load an 8-bit value to the UART0CONH/UART0CONL control registers to properly configure the UART0 I/O module.
3. For interrupt generation, set the UART0 I/O interrupt enable bit (UART0CONH.1 or UART0CONL.1) to 1.
4. When you transmit data to the UART0 buffer, the data is written to UDATA0, and the shift operation starts.

5. When the shift operation (receive/transmit) is completed, the UART0 pending bit (UART0CONH.0 or UART0CONL.0) is set to 1 and a UART0 interrupt request is generated.

20.2. UART0 High-Byte Control Register

The high-byte control register for UART0 is called UART0CONH, and is shown in Table 80. The UART0CONH Register is located at Set1, Bank0 at address EEh, and provides the following control functions:

- Operating mode and baud rate selection
- Multiprocessor communication and interrupt control
- Serial receive enable/disable control
- 9th data bit location for transmit and receive operations (modes 2 and 3 only)
- UART0 receive interrupt control

A reset clears the UART0CONH value to 00h. Therefore, to use the UART0 module, write the appropriate value to UART0CONH.

Table 80. UART0 Control High Byte Register (UART0CONH; Set1, Bank0)

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

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

Bit	Description
[7:6] MS1:MS0	**UART0 Mode Selection Bits**[1] 00: Mode 0: Shift Register (f_U/(16 x (BRDATA0 + 1))). 01: Mode 1: 8-bit UART (f_U/(16 x (BRDATA0 + 1))). 10: Mode 2: 9-bit UART (f_U/16). 11: Mode 3: 9-bit UART (f_U/(16 x (BRDATA0 + 1))).

Notes:
1. The descriptions for the 8-bit and 9-bit UART modes do not include start and stop bits for the serial data receive and transmit operations.
2. In modes 2 and 3, if the MCE bit is set to 1, then the receive interrupt will not be activated if the received 9th data bit is 0. In mode 1, if MCE = 1, then the receive interrupt will not be activated if a valid stop bit was not received. In Mode 0, the MCE bit should be 0.
3. If UART0CONL.7 = 1, this bit is a *don't care*.

Bit	Description (Continued)
[5]	**Multiprocessor Communication Enable Bit (modes 2 and 3 only)[2]** 0: Disable. 1: Enable.
[4]	**Serial Data Receive Enable Bit** 0: Disable. 1: Enable.
[3]	**TB8 (only when UART0CONL.7 = 0)[3]** Location of the 9th data bit to be transmitted in UART0 Mode 2 or 3 (i.e., 0 or 1).
[2]	**RB8 (only when UART0CONL.7 = 0)[3]** Location of the 9th data bit to be received in UART0 Mode 2 or 3 (i.e., 0 or 1).
[1]	**UART0 Receive Interrupt Enable Bit** 0: Disable Rx interrupt. 1: Enable Rx interrupt.
[0]	**UART0 Receive Interrupt Pending Bit** 0: No interrupt pending (when read); clear pending bit (when write). 1: Interrupt is pending (when read).

Notes:
1. The descriptions for the 8-bit and 9-bit UART modes do not include start and stop bits for the serial data receive and transmit operations.
2. In modes 2 and 3, if the MCE bit is set to 1, then the receive interrupt will not be activated if the received 9th data bit is 0. In mode 1, if MCE = 1, then the receive interrupt will not be activated if a valid stop bit was not received. In Mode 0, the MCE bit should be 0.
3. If UART0CONL.7 = 1, this bit is a *don't care*.

20.3. UART0 Low-Byte Control Register

The low-byte control register for UART0 is called UART0CONL, and is shown in Table 81. The UART0CONL Register is located at Set1, Bank0 at address EFh, and provides the following control functions:

- UART0 transmit and receive parity-bit selection
- UART0 clock selection
- UART0 transmit interrupt control

A reset clears the UART0CONL value to 00h. Therefore, to use UART0 module, write the appropriate value to UART0CONL.

Table 81. UART0 Control Low Byte Register (UART0CONL; Set1, Bank0)

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

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

Bit	Description
[7]	**UART0 Transmit Parity Autogeneration Enable Bit (modes 2 and 3 only)** 0: Disable parity bit autogeneration. 1: Enable parity bit autogeneration.
[6]	**UART0 Transmit Parity Selection Bit (modes 2 and 3 only)*** 0: Even parity bit. 1: Odd parity bit.
[5]	**UART0 Receive Parity Selection Bit (modes 2 and 3 only)*** 0: Even parity bit check. 1: Odd parity bit check.
[4]	**UART0 Receive Parity Error Status Bit (modes 2 and 3 only)*** 0: No parity bit error. 1: Parity bit error.
[3:2]	**UART0 Clock Selection Bits** 00: $f_{XX}/8$. 01: $f_{XX}/4$. 10: $f_{XX}/2$. 11: $f_{XX}/1$.
[1]	**UART0 Transmit Interrupt Enable Bit** 0: Disable Tx interrupt. 1: Enable Tx interrupt.
[0]	**UART0 Transmit Interrupt Pending Bit** 0: No interrupt pending (when read); clear pending bit (when write). 1: Interrupt is pending (when read).

Note: *If UART0CONL.7 = 0, this bit is a *don't care*.

20.4. UART0 Interrupt Pending bits

In Mode 0, the receive interrupt pending bit UART0CONH.0 is set to 1 when the 8th receive data bit has been shifted. In mode 1, the UART0CONH.0 bit is set to 1 at the half-way point of the stop bit's shift time. In modes 2 or 3, the UART0CONH.0 bit is set to 1 at the halfway point of the RB8 bit's shift time. When the CPU has acknowledged the

receive interrupt pending condition, the UART0CONH.0 bit must then be cleared by software in the interrupt service routine.

In Mode 0, the transmit interrupt pending bit UART0CONL.0 is set to 1 when the 8th transmit data bit has been shifted. In modes 1, 2, or 3, the UART0CONL.0 bit is set at the start of the stop bit. When the CPU has acknowledged the transmit interrupt pending condition, the UART0CONL.0 bit must then be cleared by software in the interrupt service routine.

20.5. UART0 Data Register

The UART0 Data (UDATA0) Register is shown in Table 82.

Table 82. UART0 Data Register (UDATA0; Set1, Bank0)

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

Bit	Description
[7:0]	**UART0 Data Register Configuration Bits** Transmit or Receive data.

20.6. UART0 Baud Rate Data Register

The value stored in the UART0 Baud Rate Data (BRDATA0) Register, shown in Table 83, lets you determine the UART0 clock (baud) rate.

Table 83. UART0 Baud Rate Data Register (BRDATA0; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	F1h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Bit	Description
[7:0]	**UART0 Data Register Configuration Bits** Transmit or Receive data.

20.7. UART0 Baud Rate Calculations

Baud rate calculations for UART0 modes 0 to 3 are described in this section.

20.8. Mode 0 Baud Rate Calculation

In Mode 0, the baud rate is determined by the UART0 Baud Rate Data (BRDATA0) Register, which is located at Set1, Bank0 at address F1h. The calculation for this baud rate is:

Mode 0 baud rate = $f_U/(16 \times (BRDATA0 + 1))$

20.9. Mode 2 Baud Rate Calculation

The baud rate in Mode 2 is fixed at the f_U clock frequency divided by 16, as follows:

Mode 2 baud rate = $f_U/16$

20.10. Modes 1 and 3 Baud Rate Calculation

In modes 1 and 3, the baud rate is determined by the UART0 Baud Rate Data (BRDATA0) Register, which is located at Set1, Bank0 at address F1h. The calculation for this baud rate is:

Mode 1 and 3 baud rate = $f_U/(16 \times (BRDATA0 + 1))$

Table 84 presents a matrix of commonly used baud rates generated by BRDATA0.

Table 84. Commonly Used Baud Rates Generated by BRDATA0

			BRDATA0	
Mode	**Baud Rate**	**UART Clock (f_U)**	**Decimal**	**Hexadecimal**
Mode 2	0.5 MHz	8 MHz	x	x
Mode 0 Mode 1 Mode 3	230,400 Hz	11.0592 MHz	02	02h
	115,200 Hz	11.0592 MHz	05	05h
	57,600 Hz	11.0592 MHz	11	0Bh
	38,400 Hz	11.0592 MHz	17	11h
	19,200 Hz	11.0592 MHz	35	23h
	9,600 Hz	11.0592 MHz	71	47h
	4,800 Hz	11.0592 MHz	143	8Fh
	62,500 Hz	10 MHz	09	09h
	9,615 Hz	10 MHz	64	40h
	38,461 Hz	8 MHz	12	0Ch
	12,500 Hz	8 MHz	39	27h
	19,230 Hz	4 MHz	12	0Ch
	9,615 Hz	4 MHz	25	19h

Figure 103 presents a block diagram of the UART0 function.

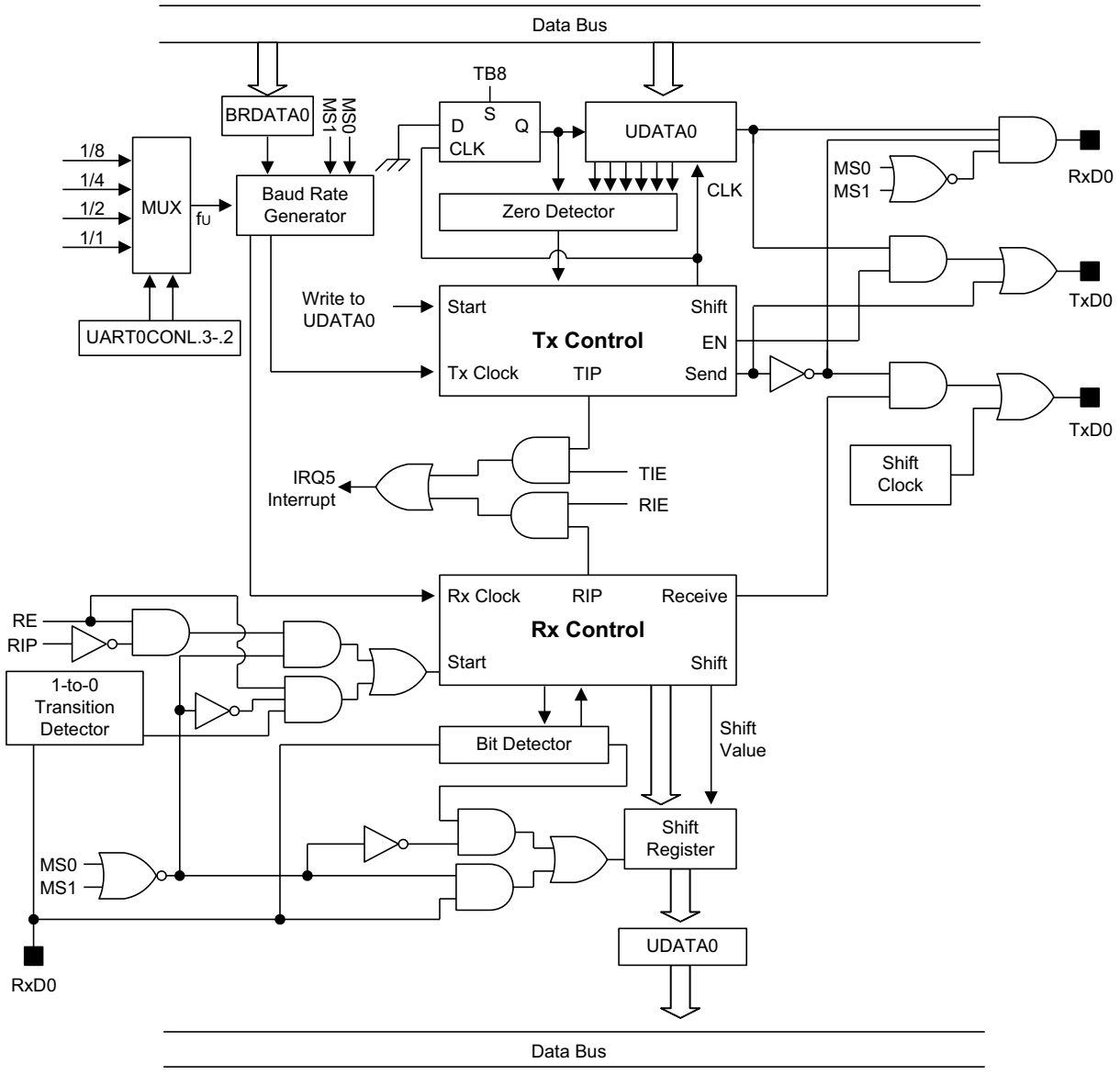


Figure 103. UART0 Functional Block Diagram

20.11. UART0 Mode 0 Function Description

In Mode 0, UART0 is input and output through the RxD0 (P3.1) pin, and the TxD0 (P3.2) pin outputs the shift clock. Data is transmitted or received in 8-bit units only. The LSB of the 8-bit value is transmitted (or received) first.

20.12. Mode 0 Transmit Procedure

Observe the following procedure to write transmission data via UART0 in Mode 0.

1. Select the UART0 clock, UART0CONL.3 and .2.
2. Clear the UART0 transmit parity-bit autogeneration enable bit, UART0CONL.7.
3. Select Mode 0 by setting UART0CONH.7 and .6 to `00b`.
4. Write the transmission data to the UDATA0 Shift Register (`F0h`, Set1, Bank0) to start the transmission operation.

20.13. Mode 0 Receive Procedure

Observe the following procedure to read receive data via UART0 in Mode 0.

1. Select the UART0 clock, UART0CONL.3 and .2.
2. Clear the UART0 transmit parity-bit autogeneration enable bit, UART0CONL.7.
3. Select Mode 0 by setting UART0CONH.7 and .6 to `00b`.
4. Clear the receive interrupt pending bit, UART0CONH.0, by writing a 0 to UART0CONH.0.
5. Set the UART0 receive enable bit, UART0CONH.4, to 1.
6. The shift clock will now be output to the TxD0 (P3.2) pin and will read the data at the RxD0 (P3.1) pin. A UART0 receive interrupt occurs when UART0CONH.1 is set to 1.

Figure 104 shows the timing of the Serial Port Mode 0 operation.

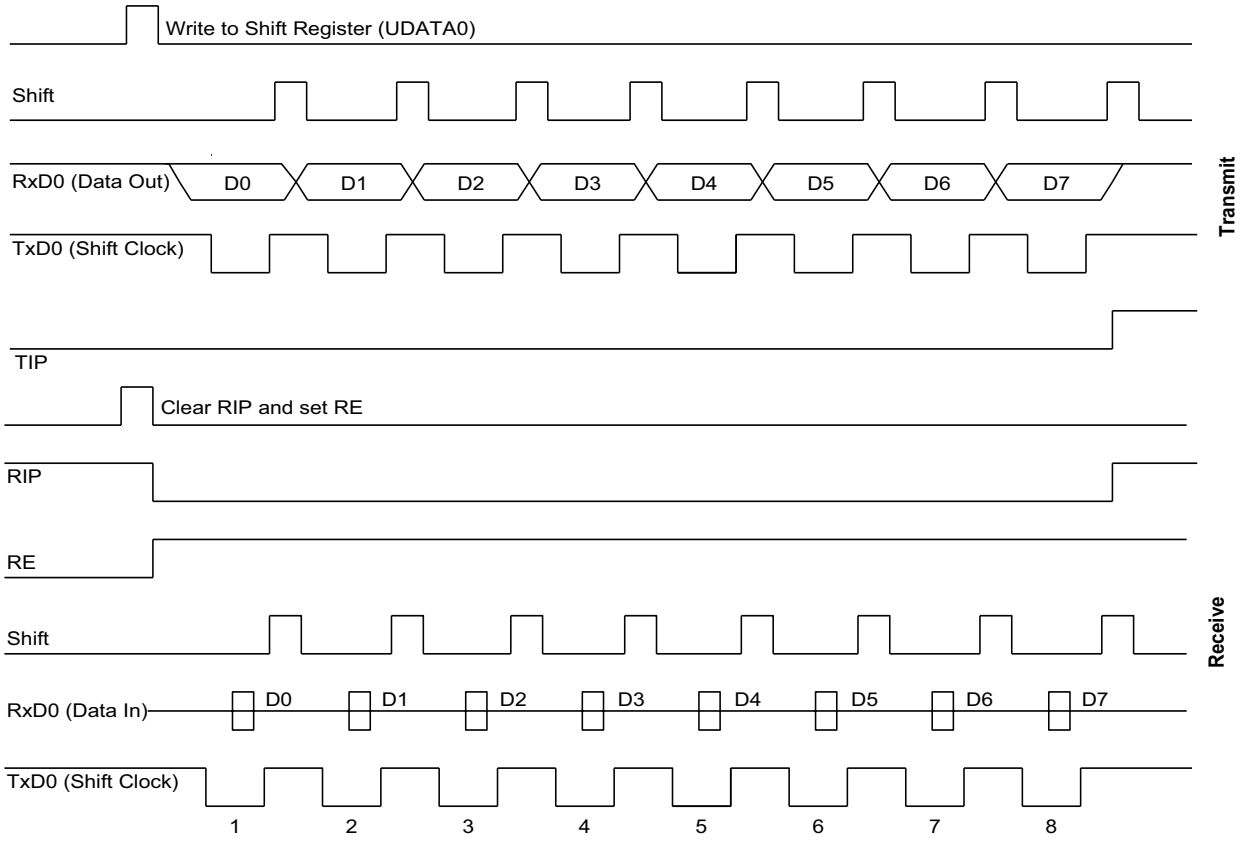


Figure 104. UART0 Serial Port Mode 0 Timing

20.14. Serial Port Mode 2 Function Description

In Mode 2, 11 bits are transmitted through the TxD0 (P3.2) pin or received through the RxD0 (P3.1) pin. Each data frame has the following four components:

- Start bit (0)
- 8 data bits (LSB first)
- Programmable 9th data bit
- Stop bit (1)

The 9th data bit to be transmitted can be assigned a value of 0 or 1 by writing the TB8 bit, UART0CONH.3. When receiving, the 9th data bit that is received is written to the RB8

bit, UART0CONH.2, while the stop bit is ignored. The baud rate for mode 2 is $f_U/16$ clock frequency.

20.15. Mode 2 Transmit Procedure

Observe the following procedure to write transmission data via UART0 in Mode 2.

1. Select the UART0 clock, UART0CONL.3 and .2.
2. Select the UART0 transmit parity-bit autogeneration enable or disable (UART0CONL.7).
3. Select Mode 2 (9-bit UART) by setting UART0CONH bits 7 and 6 to `10b`. Additionally, select the 9th data bit to be transmitted by writing TB8 to 0 or 1.
4. Write the transmission data to the UDATA0 Shift Register (`F0h`, Set1, Bank0) to start the transmit operation.

20.16. Mode 2 Receive Procedure

Observe the following procedure to read receive data via UART0 in Mode 2.

1. Select the UART0 clock, UART0CONL.3 and .2.
2. Select the UART0 transmit parity-bit autogeneration enable or disable (UART0CONL.7).
3. Select Mode 2 and set the receive enable bit (RE) in the UART0CONH Register to 1.
4. The receive operation starts when the signal at the RxD0 (P3.1) pin goes Low.

Figure 105 shows the timing of the Serial Port Mode 2 operation.

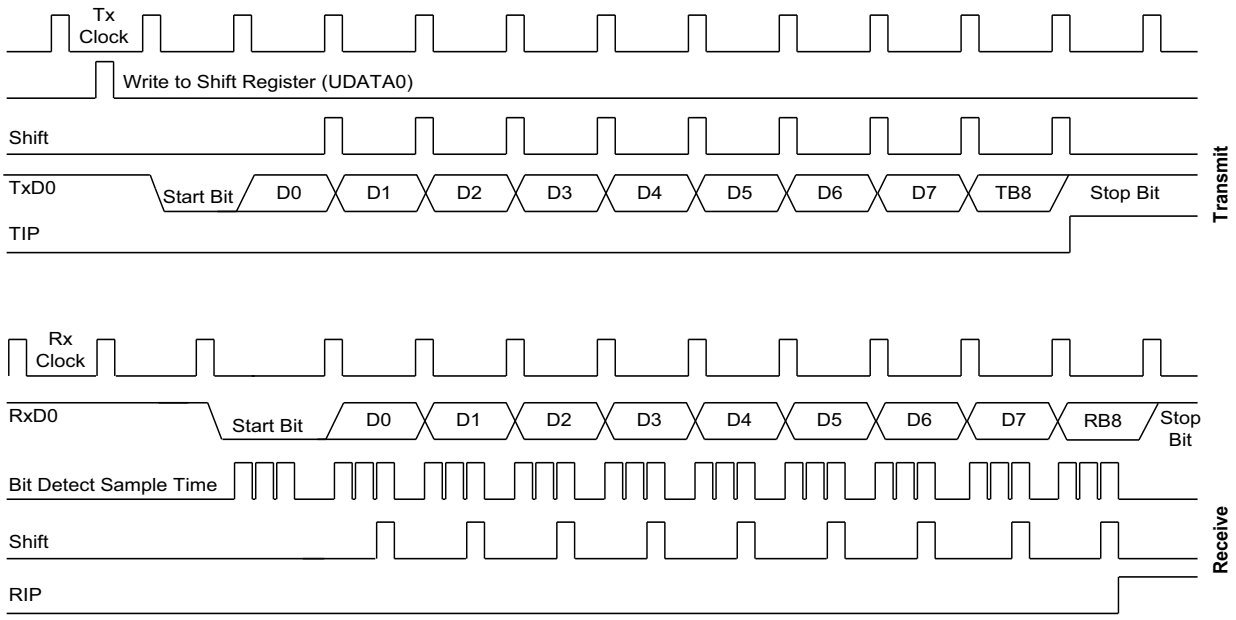


Figure 105. UART0 Serial Port Mode 2 Timing

20.17. Serial Port Mode 3 Function Description

In Mode 3, 11 bits are transmitted through the TxD0 (P3.2) pin or received through the RxD0 (P3.1) pin. Mode 3 is identical to Mode 2 except for baud rate, which is variable.

Each data frame has four components:

- Start bit (0)
- 8 data bits (LSB first)
- Programmable 9th data bit
- Stop bit (1)

20.18. Mode 3 Transmit Procedure

Observe the following procedure to write transmission data via UART0 in Mode 3.

1. Select the UART0 clock, UART0CONL.3 and .2.
2. Select the UART0 transmit parity-bit autogeneration enable or disable (UART0CONL.7).

3. Select Mode 3 operation (9-bit UART) by setting UART0CONH bits 7 and 6 to 11b. Additionally, select the 9th data bit to be transmitted by writing UART0CONH.3 (TB8) to 0 or 1.
4. Write transmission data to the UDATA0 Shift Register (F0h, Set1, Bank0), to start the transmit operation.

20.19. Mode 3 Receive Procedure

Observe the following procedure to read receive data via UART0 in Mode 3.

1. Select the UART0 clock, UART0CONL.3 and .2.
2. Select the UART0 transmit parity-bit autogeneration enable or disable (UART0CONL.7).
3. Select Mode 3 and set the Receive Enable (RE) bit in the UART0CONH Register to 1.
4. The receive operation will be started when the signal at the RxD0 (P3.1) pin goes Low.

Figure 106 shows the timing of the Serial Port Mode 3 operation.

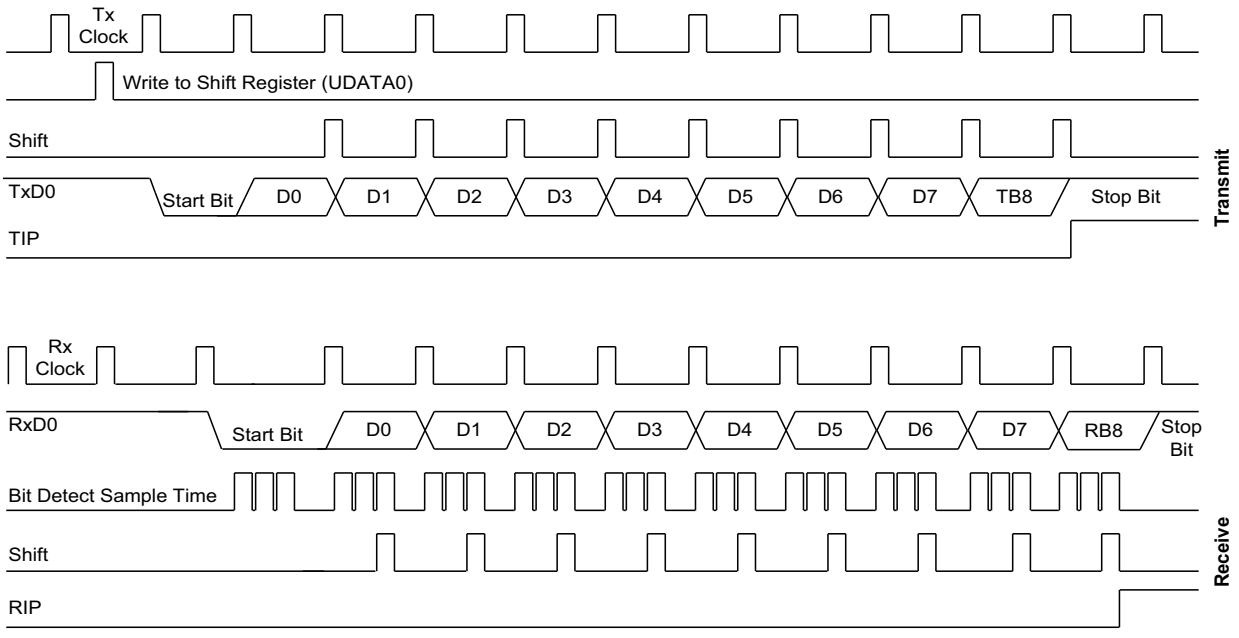


Figure 106. UART0 Serial Port Mode 3 Timing

20.20. Serial Communication for Multiprocessor Configurations

The S3F8 Series multiprocessor communication features lets a *master* S3F8S45 MCU send a multiple-frame serial message to a *slave* device in a multi- S3F8S45 configuration without interrupting other slave devices that may be on the same serial line.

This feature can be used only in UART modes 2 or 3. In these two modes, 9 data bits are received. The 9th bit value is written to RB8 (UART0CONH.2). The data receive operation is concluded with a stop bit. You can program this function so that when the stop bit is received, the serial interrupt will be generated only if RB8 = 1.

To enable this feature, set the MCE bit in the UART0CONH Register. When the MCE bit is 1, serial data frames that are received with the 9th bit = 0 do not generate an interrupt. In this case, the 9th bit simply separates the address from the serial data.

20.21. Sample Protocol for Master/Slave Interaction

When the master device wants to transmit a block of data to one of several slaves on a serial line, it first sends out an address byte to identify the target slave. Note that in this case, an address byte differs from a data byte; in an address byte, the 9th bit is 1, and in a data byte, it is 0.

The address byte interrupts all slaves so that each slave can examine the received byte and see if it is being addressed. The addressed slave then clears its MCE bit and prepares to receive incoming data bytes.

The MCE bits of slaves that were not addressed remain set, and they continue operating normally while ignoring the incoming data bytes.

While the MCE bit setting has no effect in Mode 0, it can be used in Mode 1 to check the validity of the stop bit. For Mode 1 reception, if MCE is 1, the receive interrupt will be issue unless a valid stop bit is received.

20.22. Setup Procedure for Multiprocessor Communications

Observe the following steps to configure multiprocessor communications.

1. Set all S3F8S45 devices (masters and slaves) to UART0 Mode 2 or 3.
2. Write the MCE bit of all the slave devices to 1.
3. The master device's transmission protocol is:
   - First byte: the address identifying the target slave device (9th bit = 1)

- Next bytes: data (9th bit = 0)

4. When the target slave receives the first byte, all of the slaves are interrupted because the 9th data bit is 1. The targeted slave compares the address byte to its own address and then clears its MCE bit to receive incoming data. The other slaves continue operating normally.

Figure 107 shows an example of multiprocessor serial data communications.

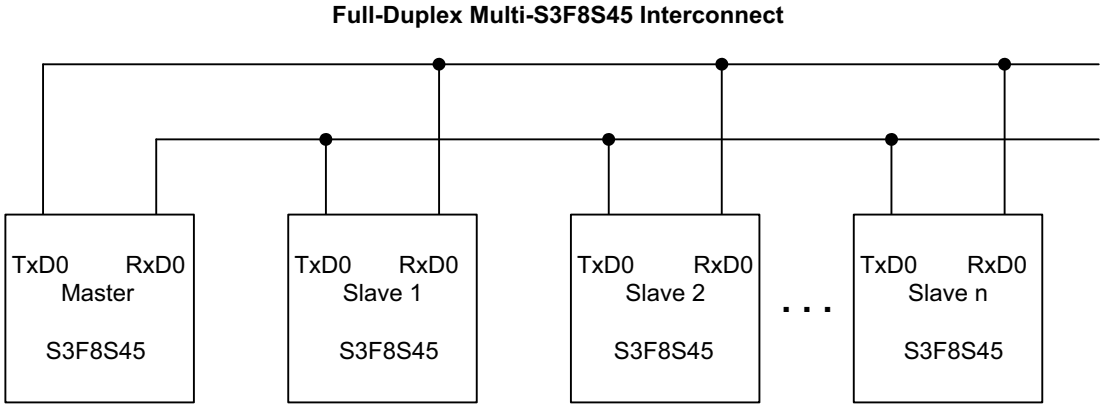


Figure 107. UART0 Multiprocessor Serial Data Communications Example

Chapter 21. UART1

The UART1 block features a full-duplex serial port with the following four programmable operating modes – one synchronous mode and three Universal Asynchronous Receiver/ Transmitter (UART) modes:

- Serial I/O with baud rate of f_U/(16 x (BRDATA1+1))
- 8-bit UART Mode; variable baud rate
- 9-bit UART Mode; f_U/16
- 9-bit UART Mode, variable baud rate

The UART1 receive and transmit buffers are both accessed via the UDATA1 Data Register, and are located in Set1, Bank0 at address F4h. Writing to the UART1 Data Register loads the transmit buffer; reading the UART1 Data Register accesses a physically separate receive buffer.

When accessing a receive data buffer (i.e., the shift register), reception of the next byte can begin before the previously received byte has been read from the receive register. However, if the first byte has not been read by the time the next byte has been completely received, one of the bytes will be lost.

In all operating modes, transmission is started when any instruction (usually a write operation) uses the UDATA1 Register as its destination address. In Mode 0, serial data reception starts when the receive interrupt pending bit (UART1CONH.0) is 0 and the receive enable bit (UART1CONH.4) is 1. In modes 1, 2, and 3, reception starts whenever an incoming start bit (0) is received and the receive enable bit (UART1CONH.4) is set to 1.

21.1. Programming Procedure

To program the UART1 modules, observe the following basic steps.

1. Configure P1.2 and P1.3 to alternative function (RxD1 (P1.2), TxD1 (P1.3)) for the UART1 module by setting the P3CONH Register to an appropriate value.
2. Load an 8-bit value to the UART1CONH/UART1CONL control registers to properly configure the UART1 I/O module.
3. For interrupt generation, set the UART1 I/O interrupt enable bit (UART1CONH.1 or UART1CONL.1) to 1.
4. When transmitting data to the UART1 buffer, write the data to UDATA1; the shift operation starts.

5. When the shift operation (receive/transmit) is completed, the UART1 pending bit (UART1CONH.0 or UART1CONL.0) is set to 1 and an UART1 interrupt request is generated.

21.2. UART1 High-Byte Control Register

The control register for UART1 is called UART1CONH, and is shown in Table 85. The UART1CONH Register is located in Set1, Bank0 at address F2h, and provides the following control functions:

- Operating mode and baud rate selection
- Multiprocessor communication and interrupt control
- Serial receive enable/disable control
- 9th data bit location for transmit and receive operations (modes 2 and 3 only)
- UART1 receive interrupt control

A reset clears the UART1CONH value to 00h. Therefore, to use the UART1 module, write the appropriate value to UART1CONH.

Table 85. UART1 Control High Byte Register (UART1CONH; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	F2h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:6]	**UART1 Mode Selection Bits**[1] 00: Mode 0: Shift Register (f_U/(16 x (BRDATA1 + 1))). 01: Mode 1: 8-bit UART (f_U/(16 x (BRDATA1 + 1))). 10: Mode 2: 9-bit UART (f_U/16). 11: Mode 3: 9-bit UART (f_U/(16 x (BRDATA1 + 1))).
[5]	**Multiprocessor Communication Enable Bit (modes 2 and 3 only)**[2] 0: Disable. 1: Enable.
[4]	**Serial Data Receive Enable Bit** 0: Disable. 1: Enable.

Bit	Description (Continued)
[3]	**TB8 (only when UART1CONL.7 = 0)[3]** Location of the 9th data bit to be transmitted in UART1 Mode 2 or 3 (i.e., 0 or 1).
[2]	**RB8 (only when UART1CONL.7 = 0)[3]** Location of the 9th data bit to be received in UART1 Mode 2 or 3 (i.e., 0 or 1).
[1]	**UART1 Receive Interrupt Enable Bit** 0: Disable Rx interrupt. 1: Enable Rx interrupt.
[0]	**UART1 Receive Interrupt Pending Bit** 0: No interrupt pending (when read); clear pending bit (when write). 1: Interrupt is pending (when read).

Notes:
1. The descriptions for the 8-bit and 9-bit UART modes do not include start and stop bits for the serial data receive and transmit operations.
2. In modes 2 and 3, if the MCE bit is set to 1, then the receive interrupt will not be activated if the received 9th data bit is 0. In mode 1, if MCE = 1, then the receive interrupt will not be activated if a valid stop bit was not received. In Mode 0, the MCE bit should be 0.
3. If UART1CONL.7 = 1, this bit is a *don't care*.

21.3. UART1 Low-Byte Control Register

The control register for the UART1 is called UART1CONL, shown in Table 86. The UART1CONL Register is located in Set1, Bank0 at address F3h, and provides the following control functions:

- UART1 transmit and receive parity-bit selection
- UART1 clock selection
- UART1 transmit interrupt control

A reset clears the UART1CONL value to 00h. Therefore, to use the UART1 module, write the appropriate value to UART1CONL.

Table 86. UART1 Control Low Byte Register (UART1CONL; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	F3h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7]	**UART1 Transmit Parity Autogeneration Enable Bit (modes 2 and 3 only)** 0: Disable parity bit autogeneration. 1: Enable parity bit autogeneration.
[6]	**UART1 Transmit Parity Selection Bit (modes 2 and 3 only)*** 0: Even parity bit. 1: Odd parity bit.
[5]	**UART1 Receive Parity Selection Bit (modes 2 and 3 only)*** 0: Even parity bit check. 1: Odd parity bit check.
[4]	**UART1 Receive Parity Error Status Bit (modes 2 and 3 only)*** 0: No parity bit error. 1: Parity bit error.
[3:2]	**UART1 Clock Selection Bits** 00: $f_{XX}/8$. 01: $f_{XX}/4$. 10: $f_{XX}/2$. 11: $f_{XX}/1$.
[1]	**UART1 Transmit Interrupt Enable Bit** 0: Disable Tx interrupt. 1: Enable Tx interrupt.
[0]	**UART1 Transmit Interrupt Pending Bit** 0: No interrupt pending (when read); clear pending bit (when write). 1: Interrupt is pending (when read).

Note: *If UART1CONL.7 = 0, this bit is a *don't care*.

21.4. UART1 Interrupt Pending Bits

In Mode 0, the receive interrupt pending bit, UART0CONH.0, is set to 1 when the 8th receive data bit has been shifted. In Mode 1, the UART0CONH.0 bit is set to 1 at the half-way point of the stop bit's shift time. In modes 2 or 3, the UART0CONH.0 bit is set to 1 at the halfway point of the RB8 bit's shift time. When the CPU has acknowledged the

receive interrupt pending condition, the UART0CONH.0 bit must then be cleared by software in the interrupt service routine.

In Mode 0, the transmit interrupt pending bit, UART1CONL.0, is set to 1 when the 8th transmit data bit has been shifted. In modes 1, 2, or 3, the UART1CONL.0 bit is set at the start of the stop bit. When the CPU has acknowledged the transmit interrupt pending condition, the UART1CONL.0 bit must then be cleared by software in the interrupt service routine.

21.5. UART1 Data Register

The UART1 Data (UDATA1) Register is shown in Table 87.

Table 87. UART1 Data Register (UDATA1; Set1, Bank0)

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

Bit	Description
[7:0]	**UART1 Data Register Configuration Bits** Transmit or Receive data.

21.6. UART1 Baud Rate Data Register

The value stored in the UART1 Baud Rate Data (BRDATA1) Register, shown in Table 88, lets you determine the UART1 clock (baud) rate.

Table 88. UART1 Baud Rate Data Register (BRDATA1; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	F5h							
Mode	Register Addressing Mode only							
Note: R = read only; R/W = read/write.								

Bit	Description
[7:0]	**UART1 Baud Rate Data Register Configuration Bits** Transmit or Receive data.

21.7. UART1 Baud Rate Calculations

Baud rate calculations for UART1 modes 0 to 3 are described below.

21.8. Mode 0 Baud Rate Calculation

In Mode 0, the baud rate is determined by the UART1 Baud Rate Data (BRDATA1) Register, which is located at Set1, Bank0 at address F5h. The calculation for this baud rate is:

Mode 0 baud rate = $f_U/(16 \times (BRDATA1 + 1))$

21.9. Mode 2 Baud Rate Calculation

The baud rate in Mode 2 is fixed at the f_U clock frequency divided by 16, as follows:

Mode 2 baud rate = $f_U/16$

21.10. Modes 1 and 3 Baud Rate Calculation

In modes 1 and 3, the baud rate is determined by the UART1 Baud Rate Data (BRDATA1) Register, which is located at Set1, Bank0 at address F5h. The calculation for this baud rate is:

Mode 1 and 3 baud rate = $f_U/(16 \times (BRDATA1 + 1))$

Table 89 presents a matrix of commonly used baud rates generated by BRDATA1.

Table 89. Commonly Used Baud Rates Generated by BRDATA1

			BRDATA1	
Mode	**Baud Rate**	**UART Clock (f_U)**	**Decimal**	**Hexadecimal**
Mode 2	0.5 MHz	8 MHz	x	x
Mode 0 Mode 1 Mode 3	230,400 Hz	11.0592 MHz	02	02h
	115,200 Hz	11.0592 MHz	05	05h
	57,600 Hz	11.0592 MHz	11	0Bh
	38,400 Hz	11.0592 MHz	17	11h
	19,200 Hz	11.0592 MHz	35	23h
	9,600 Hz	11.0592 MHz	71	47h
	4,800 Hz	11.0592 MHz	143	8Fh
	62,500 Hz	10 MHz	09	09h
	9,615 Hz	10 MHz	64	40h
	38,461 Hz	8 MHz	12	0Ch
	12,500 Hz	8 MHz	39	27h
	19,230 Hz	4 MHz	12	0Ch
	9,615 Hz	4 MHz	25	19h

Figure 108 presents a block diagram of the UART1 function.

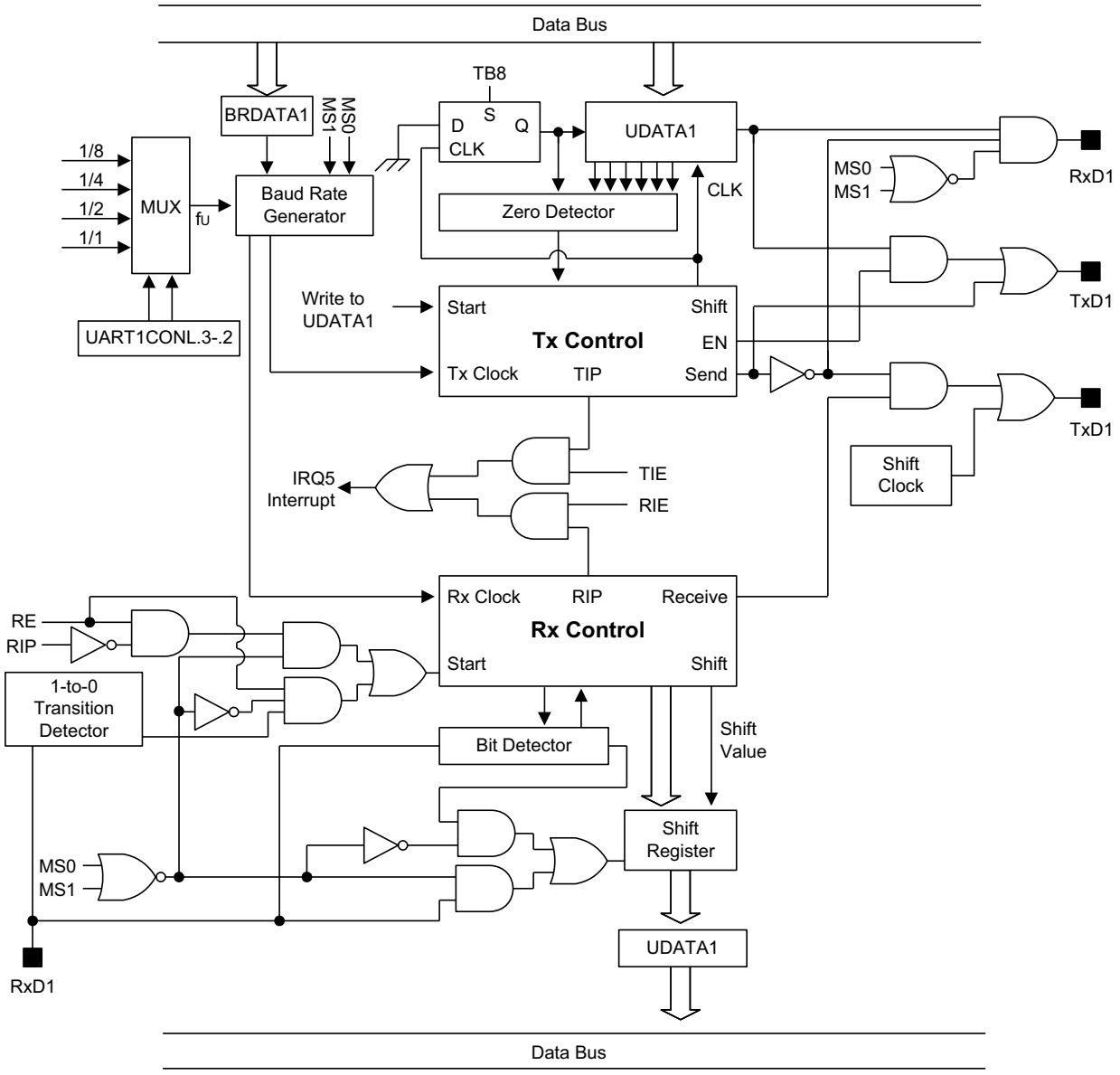


Figure 108. UART1 Functional Block Diagram

21.11. UART1 Mode 0 Function Description

In Mode 0, UART1 is input and output through the RxD1 (P1.2) pin, and TxD1 (P1.3) pin outputs the shift clock. Data is transmitted or received in 8-bit units only. The LSB of the 8-bit value is transmitted (or received) first.

21.12. Mode 0 Transmit Procedure

Observe the following procedure to write transmission data via UART1 in Mode 0.

1. Select the UART1 clock, UART1CONL.3 and .2.
2. Clear the UART1 transmit parity-bit autogeneration enable bit, UART1CONL.7.
3. Select Mode 0 by setting UART1CONH.7 and .6 to `00b`.
4. Write the transmission data to the UDATA1 Shift Register (`F4h`, Set1, Bank0) to start the transmission operation.

21.13. Mode 0 Receive Procedure

Observe the following procedure to read receive data via UART1 in Mode 0.

1. Select the UART1 clock, UART1CONL.3 and .2.
2. Clear the UART1 transmit parity-bit autogeneration enable bit, UART1CONL.7.
3. Select Mode 0 by setting UART1CONH.7 and .6 to `00b`.
4. Clear the receive interrupt pending bit, UART1CONH.0, by writing a 0 to UART1CONH.0.
5. Set the UART1 receive enable bit, UART1CONH.4, to 1.
6. The shift clock will now be output to the TxD1 (P1.3) pin and will read the data at the RxD1 (P1.2) pin. A UART1 receive interrupt occurs when UART1CONH.1 is set to 1.

Figure 109 shows the timing of the UART1 Serial Port Mode 0 operation.

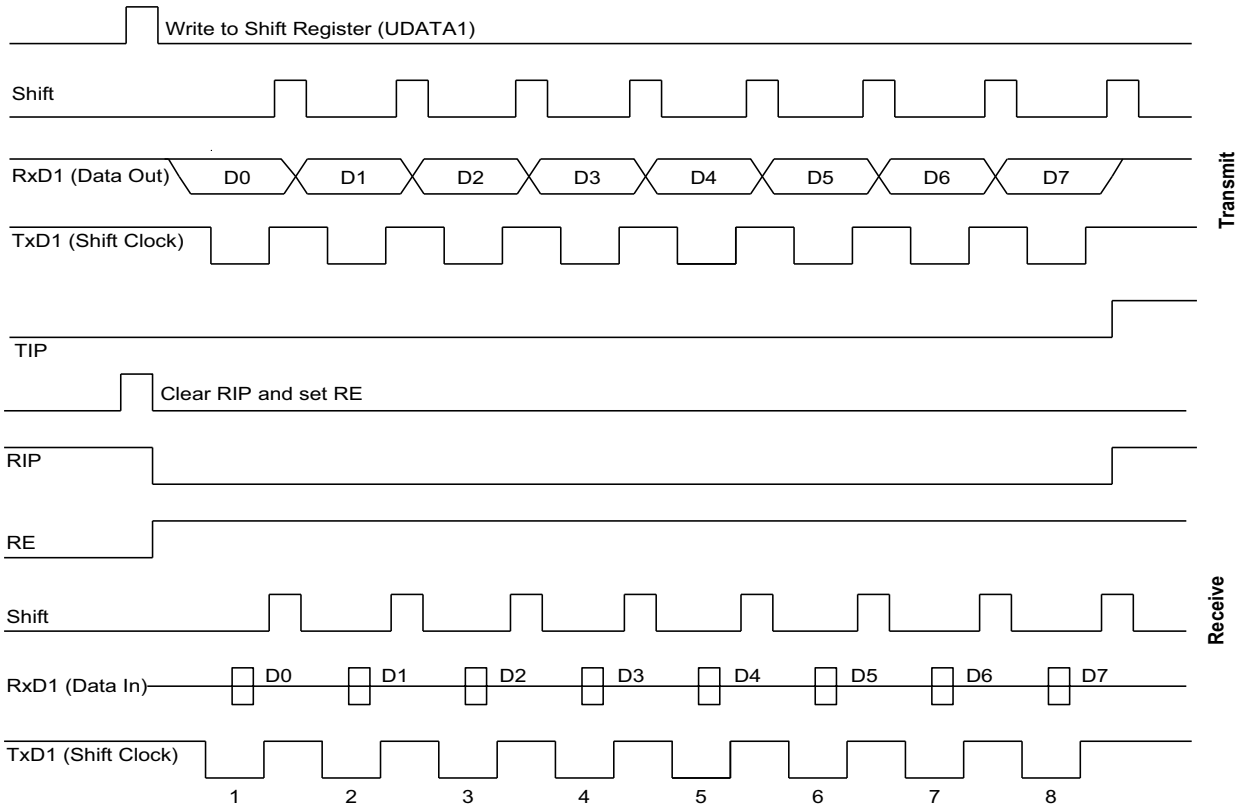


Figure 109. UART1 Serial Port Mode 0 Timing

21.14. Serial Port Mode 1 Function Description

In Mode 1, 10-bits are transmitted through the TxD1 (P1.3) pin or received through the RxD1 (P1.2) pin. Each data frame features the following three components:

- Start bit (0)
- 8 data bits (LSB first)
- Stop bit (1)

The baud rate for Mode 1 is variable.

21.15. Mode 1 Transmit Procedure

Observe the following procedure to write transmission data via UART1 in Mode 1.

1. Select the UART1 clock, UART1CONL.3 and .2.
2. Clear the UART1 transmit parity-bit autogeneration enable bit (UART1CONL.7).
3. Select the baud rate to be generated by BRDATA1.
4. Select Mode 1 (8-bit UART) by setting UART1CONH bits 7 and 6 to `01b`.
5. Write transmission data to the UDATA1 Shift Register (`F4h`, Set1, Bank0). The start and stop bits are generated automatically by hardware.

21.16. Mode 1 Receive Procedure

Observe the following procedure to read receive data via UART1 in Mode 1.

1. Select the UART1 clock, UART1CONL.3 and .2.
2. Clear the UART1 transmit parity-bit autogeneration enable bit (UART1CONL.7).
3. Select the baud rate to be generated by BRDATA1.
4. Select Mode 1 and set the Receive Enable (RE) bit in the UART1CONH Register to 1.
5. The start bit low (0) condition at the RxD1 (P1.2) pin will cause the UART1 module to start the serial data receive operation.

Figure 110 shows the timing of the UART1 Serial Port Mode 1operation.

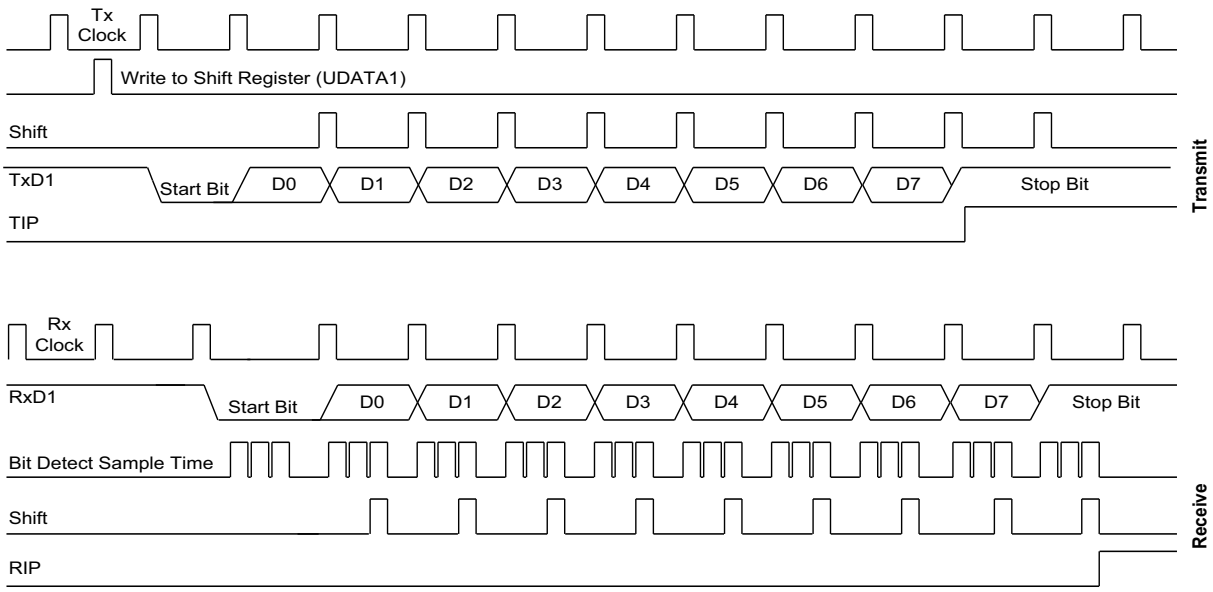


Figure 110. UART1 Serial Port Mode 1Timing

21.17. Serial Port Mode 2 Function Description

In Mode 2, 11 bits are transmitted through the TxD1 (P1.3) pin. Each data frame features the following three components:

- Start bit (0)
- 8 data bits (LSB first)
- Programmable 9th data bit
- Stop bit (1)

The 9th data bit to be transmitted can be assigned a value of 0 or 1 by writing the TB8 bit, UART1CONH.3. When receiving, the 9th data bit that is received is written to the RB8 bit, UART1CONH.2, while the stop bit is ignored. The baud rate for mode 2 is $f_U/16$ clock frequency.

21.18. Mode 2 Transmit Procedure

Observe the following procedure to write transmission data via UART1 in Mode 2.

1. Select the UART1 clock, UART0CONL.3 and .2.

2. Select the UART1 transmit parity-bit autogeneration enable or disable (UART0CONL.7).
3. Select Mode 2 (9-bit UART) by setting UART1CONH bits 7 and 6 to `10b`. Additionally, select the 9th data bit to be transmitted by writing TB8 to 0 or 1.
4. Write the transmission data to the UDATA1 Shift Register (`F4h`, Set1, Bank0) to start the transmit operation.

21.19. Mode 2 Receive Procedure

Observe the following procedure to read receive data via UART1 in Mode 2.

1. Select the UART1 clock, UART0CONL.3 and .2.
2. Select the UART1 transmit parity-bit autogeneration enable or disable (UART0CONL.7).
3. Select Mode 2 and set the receive enable bit (RE) in the UART1CONH Register to 1.
4. The receive operation starts when the signal at the RxD1 (P1.2) pin goes Low.

Figure 105 shows the timing of the Serial Port Mode 2 operation.

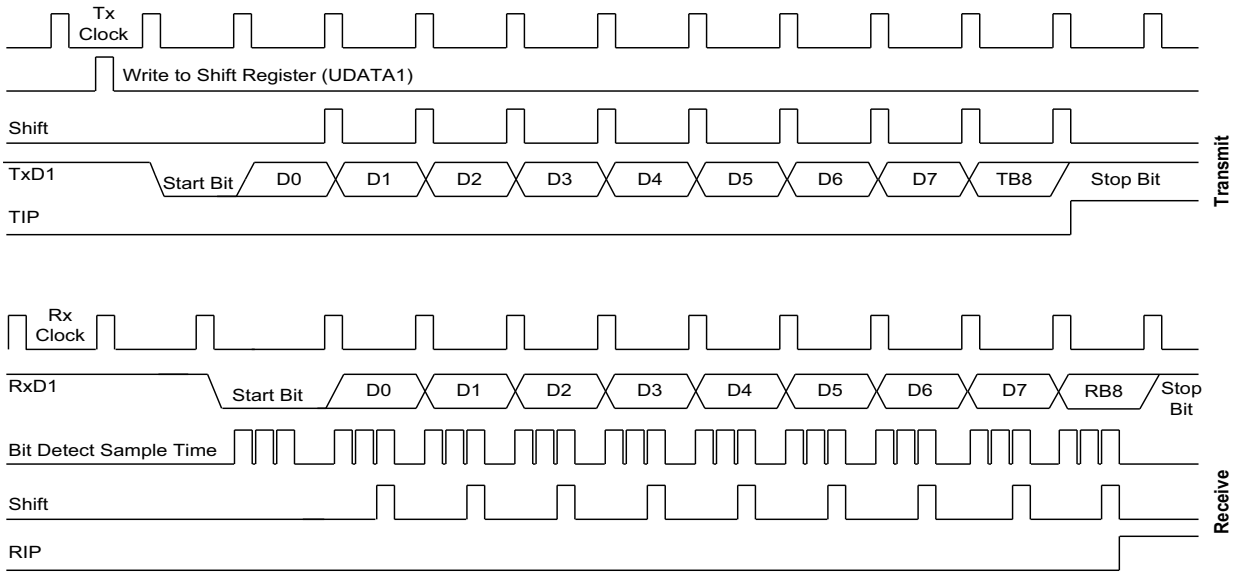


Figure 111. UART1 Serial Port Mode 2 Timing

21.20. Serial Port Mode 3 Function Description

In Mode 3, 11 bits are transmitted through the TxD1 (P1.3) pin or received through the RxD0 (P1.2) pin. Mode 3 is identical to Mode 2 except for the baud rate, which is variable. Each data frame features the following four components:

- Start bit (0)
- 8 data bits (LSB first)
- Programmable 9th data bit
- Stop bit (1)

21.21. Mode 3 Transmit Procedure

Observe the following procedure to write transmission data via UART1 in Mode 3.

1. Select the UART1 clock, UART1CONL.3 and .2.
2. Select the UART1 transmit parity-bit autogeneration enable or disable (UART1CONL.7).
3. Select Mode 3 operation (9-bit UART) by setting UART1CONH bits 7 and 6 to `11b`. Additionally, select the 9th data bit to be transmitted by writing UART1CONH.3 (TB8) to 0 or 1.
4. Write transmission data to the UDATA1 Shift Register (`F4h`, Set1, Bank0), to start the transmit operation.

21.22. Mode 3 Receive Procedure

Observe the following procedure to read receive data via UART1 in Mode 3.

1. Select the UART1 clock, UART1CONL.3 and .2.
2. Select the UART1 transmit parity-bit autogeneration enable or disable (UART1CONL.7).
3. Select Mode 3 and set the Receive Enable (RE) bit in the UART1CONH Register to 1.
4. The receive operation will be started when the signal at the RxD1 (P1.2) pin goes Low.

Figure 112 shows the timing of the Serial Port Mode 3 operation.

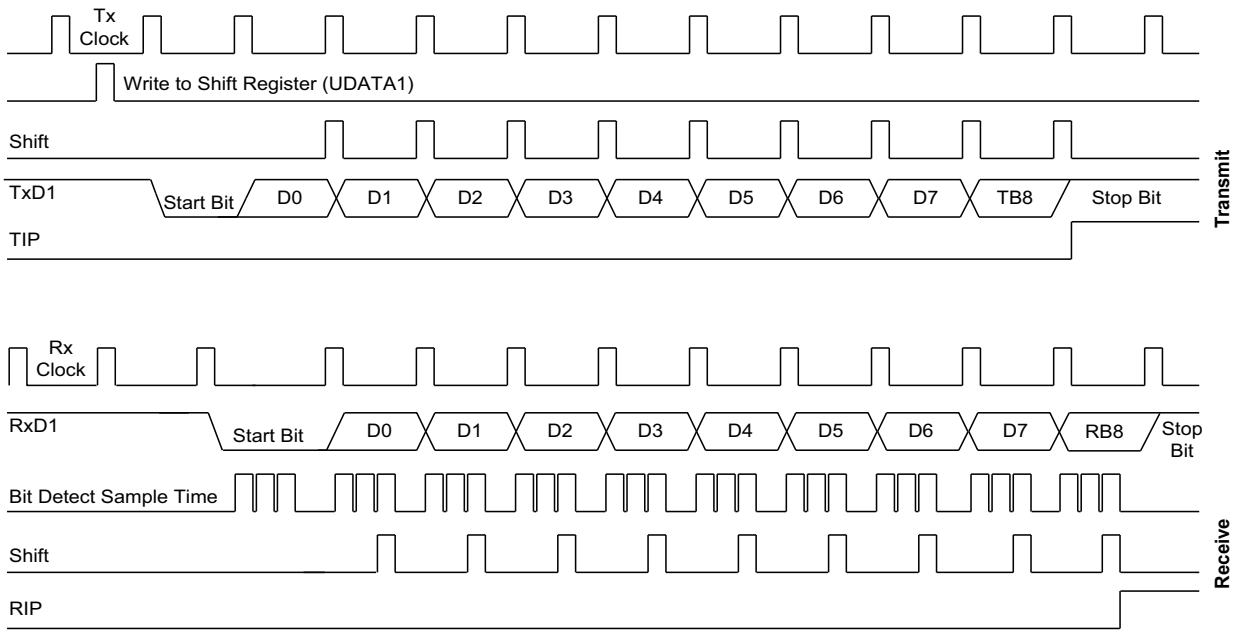


Figure 112. UART1 Serial Port Mode 3 Timing

21.23. Serial Communication for Multiprocessor Configurations

The S3F8 Series multiprocessor communication features lets a *master* S3F8S45 MCU send a multiple-frame serial message to a *slave* device in a multi-S3F8S45 configuration without interrupting other slave devices that may be on the same serial line. This feature can be used only in UART modes 2 or 3. In these two modes, 9 data bits are received. The 9th bit value is written to RB8 (UART1CONH.2). The data receive operation is concluded with a stop bit. This function can be programmed such that when the stop bit is received, the serial interrupt will be generated only if RB8 = 1.

To enable this feature, set the MCE bit in the UART1CONH Register. When the MCE bit is 1, serial data frames that are received with the 9th bit = 0 do not generate an interrupt. In this case, the 9th bit simply separates the address from the serial data.

21.24. Sample Protocol for Master/Slave Interaction

When the master device device transmits a block of data to one of several slaves on a serial line, it first sends out an address byte to identify the target slave. Note that in this

case, an address byte differs from a data byte; in an address byte, the 9th bit is 1, and in a data byte, it is 0.

The address byte interrupts all slaves so that each slave can examine the received byte and determine if it is being addressed. The addressed slave then clears its MCE bit and prepares to receive incoming data bytes.

The MCE bits of slaves that were not addressed remain set, and they continue operating normally while ignoring the incoming data bytes.

While the MCE bit setting has no effect in Mode 0, it can be used in Mode 1 to check the validity of the stop bit. For Mode 1 reception, if MCE is 1, the receive interrupt will be issue unless a valid stop bit is received.

21.25. Setup Procedure for Multiprocessor Communications

Observe the following steps to configure multiprocessor communications.

1. Set all S3F8S45 devices (masters and slaves) to UART1 Mode 2 or 3.
2. Write the MCE bit of all the slave devices to 1.
3. The master device's transmission protocol is:
   - First byte: the address identifying the target slave device (9th bit = 1)
   - Next bytes: data (9th bit = 0)
4. When the target slave receives the first byte, all of the slaves are interrupted because the 9th data bit is 1. The targeted slave compares the address byte to its own address and then clears its MCE bit to receive incoming data. The other slaves continue operating normally.

Figure 113 shows an example of multiprocessor serial data communications.

Full-Duplex Multi-S3F8S45 Interconnect

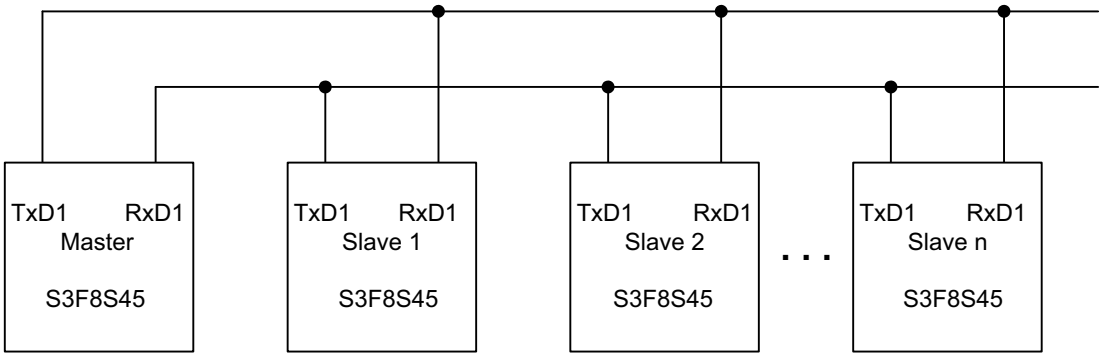


Figure 113. UART1 Multiprocessor Serial Data Communications Example

Chapter 22. Pattern Generation Module

Up to 8 bits can be output through P2.0–P2.7 by tracing the sequence shown in Figure 114. First, PGDATA must be changed into a desired output.

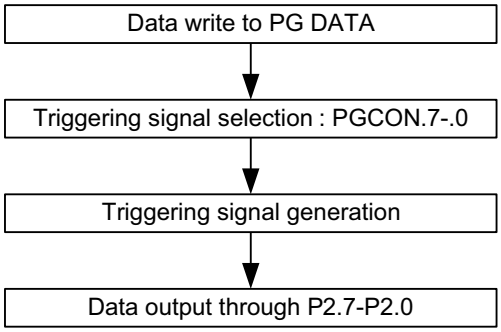


Figure 114. Pattern Generation Flow

Next, the PGCON Register, shown in Table 90, must be set to enable the pattern generation module and select the triggering signal. At this point, the PGDATA bits are in the range P2.0–P2.7 whenever the selected triggering signal occurs.

Table 90. Pattern Generation Module Control Register (PGCON; Set1, Bank1)

Bit	7	6	5	4	3	2	1	0
Reset	–	–	–	–	0	0	0	0
R/W	–	–	–	–	R/W	R/W	R/W	R/W
Address	F7h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:4]	**Reserved**
[3]	**Software Trigger Start Bit** 0: No effect. 1: Software trigger start (auto clear).

Bit	Description (Continued)
[2]	**Pattern Generation Operation Disable/Enable Selection Bit** 0: Pattern generation disable. 1: Pattern generation enable.
[1:0]	**Detection Voltage Selection Bits** 00: Timer A match signal triggering. 01: Timer B overflow signal triggering. 10: Timer D0 match signal triggering. 11: Software triggering.

Figure 115 presents a diagram of the pattern generation circuit.

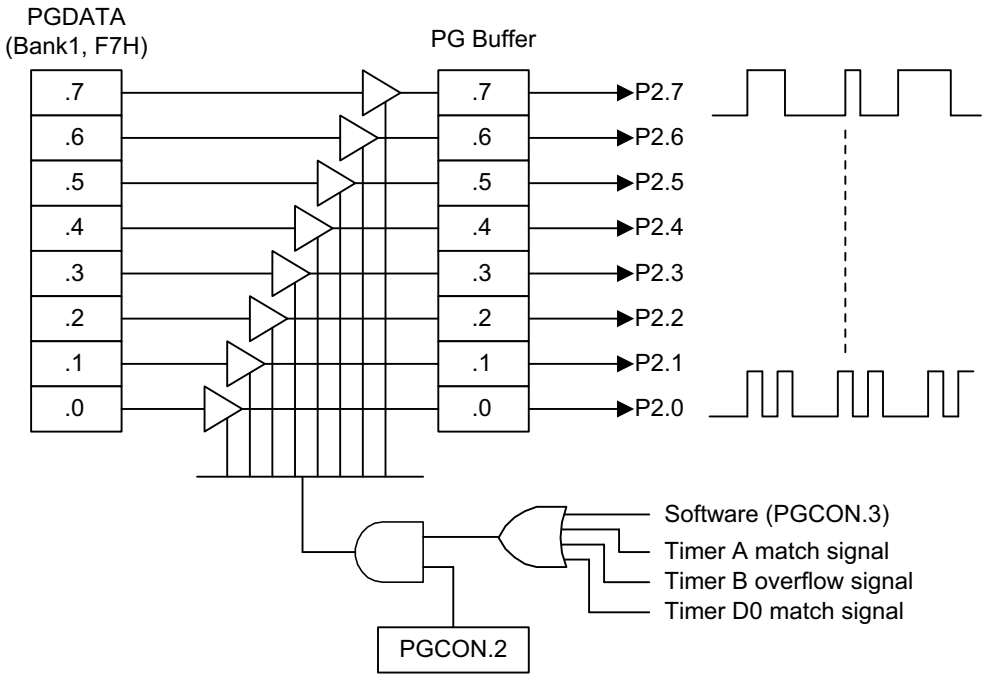


Figure 115. Pattern Generation Circuit Diagram

The following routine presents an example of pattern generation.

```
  ORG     0000h
  ORG     0100h
INITIAL:
  SB0
  LD      SYM, #00h          ; Disable Global/Fast interrupt → SYM
```

```
  LD      IMR, #01h          ; Enable IRQ0 interrupt
  LD      SPH, #0h           ; High byte of stack pointer → SPH
  LD      SPL, #0FFh         ; Low byte of stack pointer → SPL
  LD      BTCON, #10100011b ; Disable Watchdog
  LD      CLKCON, #00011000b; Non-divided (fxx)
  SB1
  LD      P2CONH,#10101010b ; Enable PG output
  LD      P2CONL,#10101010b ; Enable PG output SB0
  EI
MAIN:
  NOP
  NOP
  SB1
  LD      PGDATA, #10101010b; PG data setting
  OR      PGCON, #00000100b ; Triggering by Timer A match then
                            ; pattern data are output
  SB0
  NOP
  NOP
  JR      T,MAIN
  .END
```

Chapter 23. Battery Level Detector

The S3F8S45 microcontroller features a built-in Battery Level Detector (BLD) circuit which allows detection of a power voltage drop. Turning the BLD operation on and off can be controlled by software. Because the IC consumes a large amount of current during BLD operation, Zilog recommends that BLD operation should remain OFF unless deemed necessary.

Additionally, BLD voltage criteria can be set by software. This criteria can be set by matching to one of the following four voltage values:

- 2.4V, 2.7V, 3.3V or 3.9V (V_{DD} reference voltage)

The BLD block works only when BLDCON.0 is set. If the V_{DD} level is lower than the reference voltage selected with BLDCON.3–.2, BLDCON.1 will be set. If the V_{DD} level is higher, BLDCON.1 will be cleared. When minimizing current consumption, do not operate the BLD block.

Figure 116 presents a block diagram of the Battery Level Detect function.

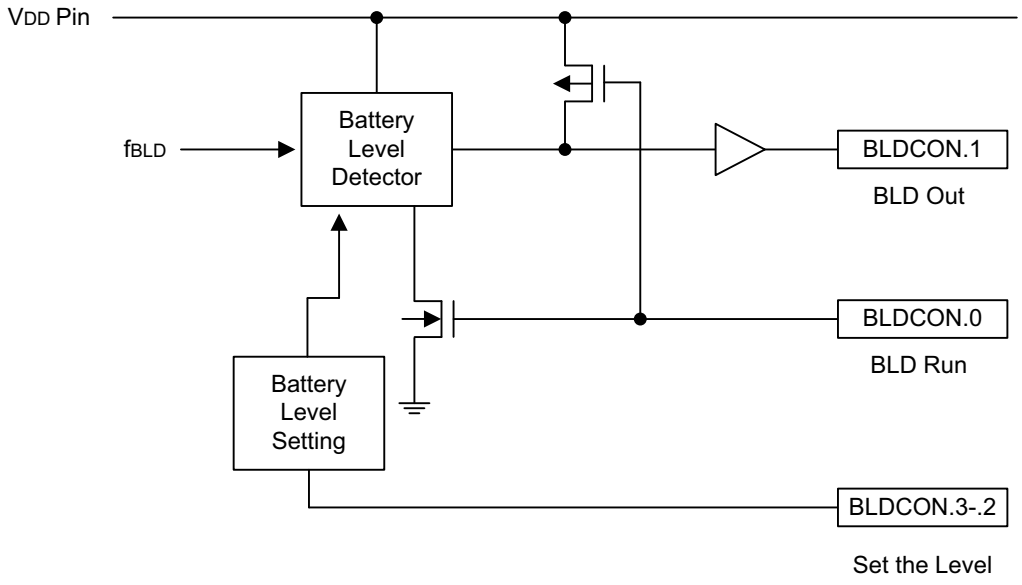


Figure 116. Battery Level Detect Block Diagram

23.1. Battery Level Detector Control Register

Bit 0 of the Battery Level Detector Control (BLDCON) Register allows control of whether to run or disable the operation of battery level detection. This V_{BLD} is essentially set to 3.3 V by a system reset, and it can be changed to 4 different voltage levels via BLDCON. When you write 2-bit data value to BLDCON, an established resistor string is selected and the V_{BLD} is fixed in accordance with this resistor. The Battery Level Detector Control (BLDCON) Register, shown in Table 91, shows these four VBLD levels.

Table 91. Battery Level Detector Control Register (BLDCON; Set1, Bank1)

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

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

Bit	Description
[7:4]	**Reserved**
[3:2]	**Detection Voltage Selection Bits** 00: V_{BLD} = 2.4 V. 01: V_{BLD} = 2.7 V. 10: V_{BLD} = 3.3 V. 11: V_{BLD} = 3.9 V
[1]	**Battery Level Detector Output Bit (Read Only)** 0: $V_{DD} > V_{REF}$ (when BLD is enabled). 1: $V_{DD} < V_{REF}$ (when BLD is enabled).
[0]	**Battery Level Detector Enable/Disable Bit** 0: Disable BLD. 1: Enable BLD.

Figure 117 presents a diagram of the battery level detector circuit.

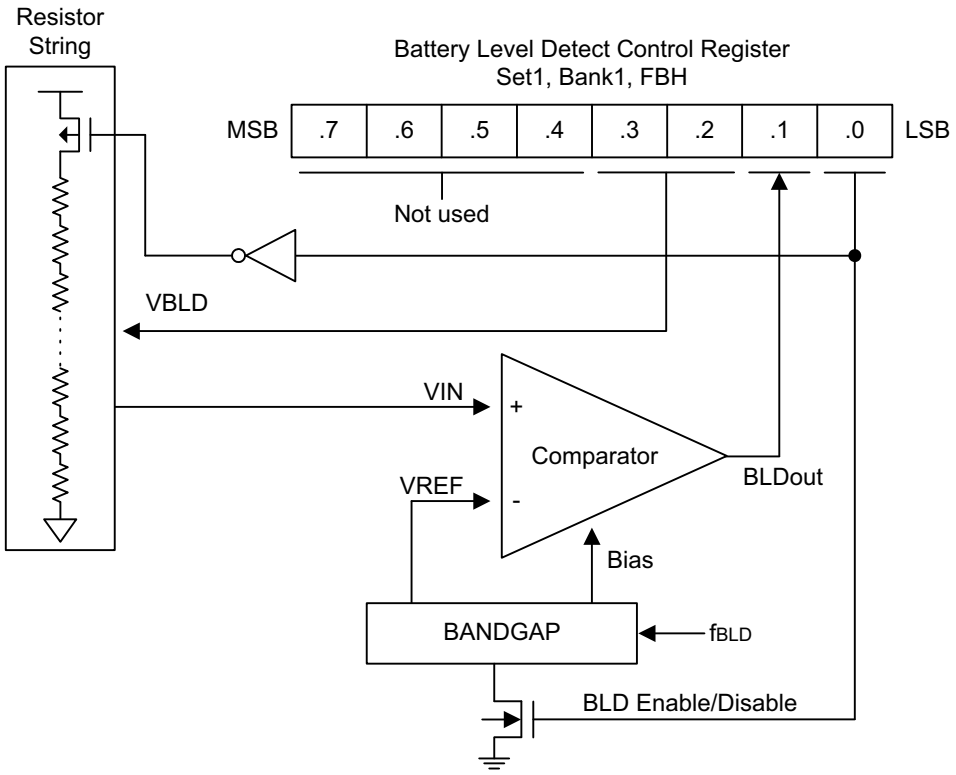


Figure 117. Battery Level Detector Circuit Diagram

Table 92 indicates voltage detection levels for differing BLDCON values.

Table 92. BLDCON Values and Detection Levels

BLDCON .2–.0	V_{BLD}
0 0	2.4V
0 1	2.7V
1 0	3.3V
1 1	3.9V

Chapter 24. Embedded Flash Memory Interface

The S3F8S45 MCU features internal on-chip Flash memory instead of masked ROM. This Flash memory is accessed by an LDC instruction. With sector erase and byte-programmable Flash, data can be programmed into a Flash memory space at any time. The S3F8S45 MCU's embedded 48 KB of memory offers the following two operating features:

- User Program Mode
- Tool Program Mode – see the S3F8S45 Flash MCU chapter on page 358

24.1. User Program Mode

User Program Mode supports sector erase, byte programming, byte read, and one protection mode, Hard Lock protection; read protection is available only in Tool Program Mode. To read-protect the chip, select a read protection option when you initially program your code in Tool Program Mode by using a programming tool.

The S3F8S45 MCU also features an internal pumping circuit; therefore, 12.5 V into a V_{PP} (test) pin is not required. To program Flash memory in this mode, several control registers are used. There are four functions: programming, reading, sector erase, and hard lock protection.

Note:
1. User Program Mode cannot be used when the CPU operates with the subsystem clock.
2. Be sure to execute the DI instruction before starting User Program Mode, which checks the Interrupt Request Register (IRQ). If an interrupt request is generated, User Program Mode is stopped.
3. User Program Mode is also stopped by an interrupt request that is masked even in the DI status. To prevent this situation, disable the interrupt by using the each peripheral interrupt enable bit.

24.2. Flash Memory Control Registers

This section describes the use of the Flash Memory Control (FMCON), Flash Memory User Programming Enable, and Flash Memory Sector Address registers when operating in User Program Mode.

24.3. Flash Memory Control Register

The Flash Memory Control (FMCON) Register, shown in Table 93, is available only in User Program Mode to select the operational mode of Flash memory, as well as the sector erase and byte programming functions, and to protect the Flash memory space with the Hard Lock function.

Table 93. Flash Memory Control Register (FMCON; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	–	–	–	0
R/W	R/W	R/W	R/W	R/W	–	–	–	R/W
Address	F9h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:4]	**Flash Memory Mode Selection Bits** 0000–0100: Reserved. 0101: Programming Mode. 1010: Sector Erase Mode. 0110: Hard Lock Mode. 0111–1111: Reserved.
[3]	**Sector Erase Status Bit (Read Only)** 0: Sector erase success. 1: Sector erase failure.
[2:1]	**Reserved**
[0]	**Flash Operation Start Bit*** 0: Operation stop. 1: Operation start.

Note: *FMCON.0 will be cleared automatically immediately after the corresponding operation has completed.

Bit 0 of the FMCON Register, FMCON.0, is a start bit for both the Erase and Hard Lock operation modes. Therefore, operation of the Erase and Hard Lock modes is activated when setting FMCON.0 to 1. Additionally, a waiting period for the Erase (sector erase) or Hard Lock modes to complete their operations must occur before performing a byte programming or byte read operation of the same sector area by with the LDC instruction. When reading or programming a byte data from or into Flash memory, this bit is not required to be manipulated.

The sector erase status bit is read only. Even if the IMR bits are 0, the interrupt is serviced during the operation of a sector erase, when each peripheral interrupt enable bit is set to 1, and when the interrupt pending bit is set to 1. If an interrupt is requested during a sector

erase operation, the operation of this sector erase is discontinued, and the interrupt is serviced by the CPU. Therefore, the sector erase status bit should be checked after executing a sector erase. The sector erase operation is successful if the bit is logic 0, and is a failure if the bit is logic 1.

Caution: When the A5h ID code is written to the FMUSR Register, it is possible that the sector erase, user program, and hard lock modes may be executed; therefore caution is necessary.

24.4. Flash Memory User Programming Enable Register

The Flash Memory User Programming Enable (FMUSR) Register, shown in Table 94, manages the safe operation of Flash memory. This register will protect undesired erase or program operations from CPU malfunctions caused by electrical noise. After reset, User Program Mode is disabled because the value of FMUSR is 00000000b as a result of the reset operation. If it is necessary to operate Flash memory, enable User Program Mode by setting the value of FMUSR to 10100101b. Any value written to FMUSR other than 10100101b disables User Program Mode.

Table 94. Flash Memory User Programming Enable Register (FMUSR; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	F8h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:0]	**Flash Memory User Programming Enable Bits** 00000001–10100100: Disable User Program Mode. 10100101: Enable User Program Mode. 10100110-11111111: Disable User Program Mode.

24.5. Flash Memory Sector Address Registers

There are two sector address registers used to select a particular sector. The Flash Memory Address Sector High Byte (FMSECH) Register and the Flash Memory Sector Address Low Byte (FMSECL) Register are combined to generate the most significant bits of the 16-bit base address for the sector to be selected. FMSECH represents the bits 8 through 15

of the 16-bit address, and FMSECL represents bit 7 of the 16-bit base address. The S3F8S45 MCU provides 384 sectors, and each sector is 128 bytes, in effect meaning that the lower 7 bits of FMSECL have no effect; see the example in Table 95.

Table 95. Flash Memory Sector Addressing

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

16-Bit Address															
.15	.14	.13	.12	.11	.10	.9	.8	.7	X	X	X	X	X	X	X

Note: Zilog recommends that the lower 7 bits of FMSECL be set to 0.

When programming Flash memory, write the data after loading the sector base address located in the target address into the FMSECH and FMSECL registers. If the next operation is also a data write operation, check to determine if the next address is located in the same sector. In the case of other sectors, you must load the sector address to the FMSECH and FMSECL registers according to the sector.

The contents of these Flash Memory Sector Address registers are described in Tables 96 and 97.

Table 96. Flash Memory Sector Address Register High Byte (FMSECH; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	F6h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7:0]	**Flash Memory Sector Address 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 upper eight bits of the 16-bit pointer address.

Table 97. Flash Memory Sector Address Register Low Byte (FMSEL; Set1, Bank0)

Bit	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0
R/W	R/W							
Address	F7h							
Mode	Register Addressing Mode only							

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

Bit	Description
[7]	**Flash Memory Sector Address Low Byte** The 7th bit to select a sector of Flash ROM.
[6:0]	**Don't Care**

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

24.6. ISP™ Onboard Programming Sector

ISP™ sectors located in the program memory space can store onboard program software (i.e., boot program code for upgrading application code by interfacing with an I/O port pin). These ISP™ sectors cannot be erased or programmed by an LDC instruction for the safety of onboard program software.

ISP sectors are available only when the ISP enable/disable bit is set to 0, i.e., ISP is enabled using the Smart Option. If you prefer not to use the ISP sector method, this area can be used as normal program memory (i.e., it can be erased or programmed using the LDC instruction) by setting the ISP disable bit (1) with the Smart Option. Even if the ISP sector is selected, the ISP sector can be erased or programmed in Tool Program Mode using a serial programming tool.

The size of the ISP sector can be adjusted using Smart Option settings; see Figure 118. Additionally, refer to Table 99 on page 333 to choose an appropriate ISP sector size according to the size of the onboard program software.

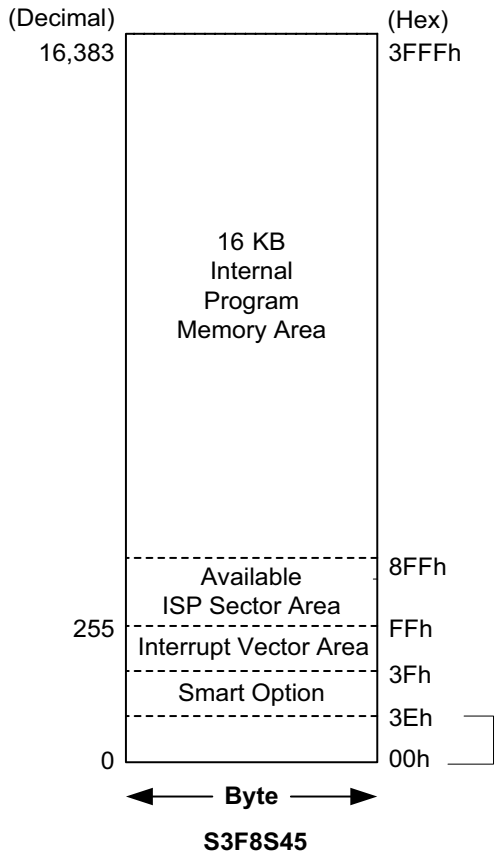


Figure 118. Program Memory Address Space

24.7. ISP Reset Vector and ISP Sector Size

If you use ISP sectors by setting the ISP enable/disable bit to 0 and the reset vector selection bit to 0 at the Smart Option (see Figure 17 on page 23), you can choose the reset vector address of the CPU as shown in Table 98 by setting the ISP reset vector address selection bits; also see the list of ISP sector sizes in Table 99.

Table 98. Reset Vector Address

Smart Option (003Eh) ISP Reset Vector Address Selection Bit			Reset Vector Address after POR	Usable Area for ISP Sector	ISP Sector Size
Bit 7	Bit 6	Bit 5			
1	x	x	0100h	0	0
0	0	0	0200h	100h–1FFh	256 bytes
0	0	1	0300h	100h–2FFh	512 bytes
0	1	0	0400h	100h–4FFh	1024 bytes
0	1	1	0500h	100h–8FFh	2048 bytes

Note: The selection of the ISP reset vector address by Smart Option (003Eh.7–003Eh.5) is not dependent of the selection of ISP sector size by Smart Option (003Eh.2–003Eh.0).

Table 99. ISP Sector Size

Smart Option (003Eh) ISP Size Selection Bit				
Bit 2	Bit 1	Bit 0	Area of ISP Sector	ISP Sector Size
1	x	x	0	0
0	0	0	100h–1FFh (256 bytes)	256 bytes
0	0	1	100h–2FFh (512 bytes)	512 bytes
0	1	0	100h–4FFh (1024 bytes)	1024 bytes
0	1	1	100h–8FFh (2048 bytes)	2048 bytes

Note: The area of the ISP sector selected by Smart Option bits (003Eh.2–003Eh.0) cannot be erased and programmed by the LDC instruction in User Program Mode.

24.8. Sector Erase Operations

Flash memory can be partially erased by using the sector erase functions in User Program Mode only. Sectors are the only units of Flash memory that can be erased in User Program Mode.

Program memory on the S3F8S45 MCU is divided into 384 sectors for erase and program operations; every sector is 128 bytes. Each sector should be first be erased prior to programming a new data byte into Flash memory. A minimum of 10 ms delay time is required prior to an erase and after setting the sector address and triggering the erase start bit (FMCON.0). The sector erase function is not supported in Tool Program modes (i.e., when using an MDS mode too or programming tool).

Figure 119 portrays how sectors are mapped in User Program Mode.

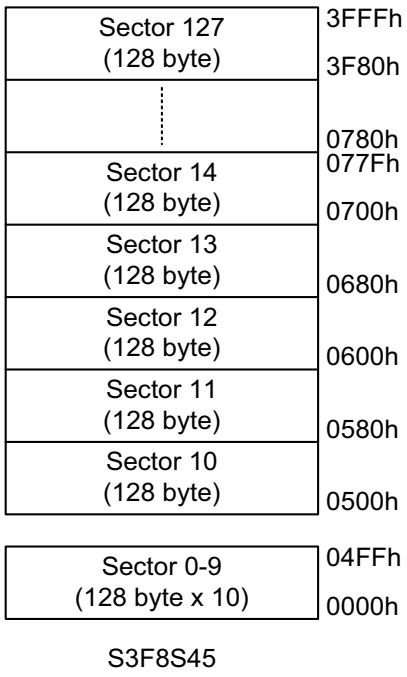


Figure 119. Sector Configurations in User Program Mode

24.9. The Sector Erase Procedure in User Program Mode

Observe the following procedure to perform a sector erase in User Program Mode.

1. If the sector erase procedure must be stopped by any interrupt, set the appropriate bit in the Interrupt Mask Enable Register (IMR) and the appropriate peripheral interrupt enable bit. Otherwise, clear all bits in the Interrupt Mask Enable Register (IMR) and all peripheral interrupt enable bits.
2. Set the Flash Memory User Programming Enable (FMUSR) Register to `10100101b`.
3. Set the Flash Memory Sector Address (FMSECH and FMSECL) registers.
4. Check the user's ID code (written by user).

5. Set the Flash Memory Control (FMCON) Register to 10100001b.
6. Set the Flash Memory User Programming Enable (FMUSR) Register to 00000000b.
7. Check the sector erase status bit to determine if the sector erase is successful.

The following routine presents an example of a successful sector erase.

```
  •
  •
  SB0
reErase: LD FMUSR,Temp0      ; User Program mode enable
                             ; Temp0 = #0A5H
                             ; Temp0 variable is must be setting
                             ; another routine
  LD      FMSECH,#10h
  LD      FMSECL,#00h        ; Set sector address (1000h-107Fh)
CP UserID_Code,#User_value   ; Check user's ID code (written by
                             ; user)
                             ; User_value is any value by user
  JR      NE,Not_ID_Code     ; If not equal, jump to Not_ID_Code
  LD      FMCON,Temp1        ; Start sector erase
                             ; Temp1 = #0A1h
                             ; Temp1 variable is must be setting
                             ; another routine
  NOP                        ; Dummy instruction - required
  NOP                        ; Dummy instruction - required
  LD      FMUSR,#0           ; User Program Mode disable
  TM FMCON,#00001000b        ; Check "sector erase status bit"
  JR      NZ,reErase         ; Jump to reErase if fail
  •
  •
  •
  •
Not_ID_Code:
  SB0
  LD      FMUSR,#0           ; User Program Mode disable
  LD      FMCON,#0           ; Sector Erase Mode disable
  •
  •
  •
  •
```

Note: In the case of Flash User Mode, the Temp0 to Temp1 data values must set another routine. Temp0 to Temp(n) variables should be defined by the user.

24.10. Program Operations

After a sector erase, Flash memory is programmed in one-byte units. For the sake of programming safety, FMSECH and FMSECL must each be set to the Flash memory sector value.

Programming in User Program Mode

Observe the following procedure to program Flash memory in User Program Mode.

1. Erase all target sectors before programming.
2. Set the Flash Memory User Programming Enable (FMUSR) Register to `10100101b`.
3. Set the Flash Memory Sector Address registers (FMSECH and FMSECL) to the sector base address of the destination address to write data.
4. Load the Flash memory upper address into the upper register of the working register pair.
5. Load the Flash memory lower address into the lower register of the working register pair.
6. Load transmission data into a working register.
7. Check the user's ID code (written by user).
8. Set the Flash Memory Control Register (FMCON) to `01010001b`.
9. Load transmission data to a Flash memory location using the LDC instruction via Indirect Addressing Mode.
10. Set the Flash Memory User Programming Enable (FMUSR) Register to `00000000b`.

The following routine presents an example of a successful programming operation.

```
•
•
SB0
LD      FMUSR,Temp0         ; User Program Mode enable
                            ; Temp0 = #0A5H
                            ; Temp0 variable is must be setting
                            ; another routine
```

```
  LD      FMSECH,#17h
  LD      FMSECL,#80h         ; Set sector address (1780h-17FFh)
  LD      R2,#17h             ; Set a ROM address in the same sector
                              ; 1780h-17FFh
  LD      R3,#84h
  LD      R4,#78h             ; Temporary data
  CP      UserID_Code,#User_value ; Check user's ID code (written
                              ; by user)
                              ; User_value is any value by user
  JR      NE,Not_ID_Code      ; If not equal, jump to Not_ID_Code
  LD      FMCON,Temp1         ; Start program
                              ; Temp1 = #51H
                              ; Temp1 variable is must be setting
                              ; another routine
  LDC     @RR2,R4             ; Write the data to a address of same
                              ; sector(1784h)
  NOP                         ; Dummy instruction - required
  LD      FMUSR,#0            ; User Program Mode disable
  •
  •
  •
  •
Not_ID_Code:
  SB0
  LD      FMUSR,#0            ; User Program Mode disable
  LD      FMCON,#0            ; Programming Mode disable
```

Note: In the case of Flash User Mode, the Temp0 to Temp1 data values must set another routine. Temp0 to Temp(n) variables should be defined by the user.

24.11. Read Operations

The read operation is initiated by the LDC instruction. Observe the following procedure to program read operations in User Program Mode.

1. Load an upper Flash memory address into the upper register of the working register pair.
2. Load a lower Flash memory address into the lower register of the working register pair.

3. Load the receive data from the Flash memory space using the LDC instruction via Indirect Addressing Mode.

The following example shows how to perform read programming.

```
  •
  •
  LD      R2,#3h            ; Load Flash memory upper address
                            ; to upper of pair working register
  LD      R3,#0             ; Load Flash memory lower address
                            ; to lower pair working register
LOOP: LDC R0,@RR2           ; Read data from Flash memory location
                            ; (Between 300h and 3FFh)
  INC     R3
  CP      R3,#0h
  JP      NZ,LOOP
  •
  •
  •
  •
```

24.12. Hard Lock Protection

The Hard Lock Protection function prevents changes to data in Flash memory. It can be set by writing 0110b to FMCON.7–.4. If this function is enabled, the user cannot write or erase the data within Flash memory. This protection can be released by executing a chip erase in Tool Program Mode.

Hard Lock Protection can be enabled by the application software when User Program Mode is enabled, or with a Serial Programmer while in Tool Program Mode. Refer to the documentation that accompanies the Serial Programmer you are using to enable Hard Lock Protection in Tool Program Mode.

To enable Hard Lock protection with application software, observe the following procedure.

1. Set the Flash Memory User Programming Enable (FMUSR) Register to 10100101b.
2. Check the user's ID code (written by user).
3. Set the Flash Memory Control (FMCON) Register to 01100001b.
4. Set the Flash Memory User Programming Enable (FMUSR) Register to 00000000b.

The following example shows how to set Hard Lock protection.

```
  SB0
```

```
  LD      FMUSR,Temp0         ; User Program Mode enable
                              ; Temp0 = #0A5h
                              ; Temp0 variable is must be setting
                              ; another routine
  CP      UserID_Code,#User_value ; Check user's ID code (written
                              ; by user)
                              ; User_value is any value by user
  JR      NE,Not_ID_Code      ; If not equal, jump to Not_ID_Code
  LD      FMCON,Temp1         ; Hard Lock Mode set & start
                              ; Temp1 = #61H
                              ; Temp1 variable is must be setting
                              ; another routine
  NOP                         ; Dummy Instruction - required
  LD      FMUSR,#0            ; User Program Mode disable
  •
  •
  •
  •
Not_ID_Code:
  SB0
  LD      FMUSR,#0            ; User Program Mode disable
  LD      FMCON,#0            ; Hard Lock Protection Mode disable
  •
  •
  •
  •
```

Note: In the case of Flash User Mode, the Temp0 to Temp1 data values must set another routine. Temp0 to Temp(n) variables should be defined by the user.

Chapter 25. Electrical Characteristics

In this chapter, the S3F8S45 MCU's electrical characteristics are presented in tables and charts. This information is arranged in the following order:

1. Absolute Maximum Ratings – see Table 100
2. DC Electrical Characteristics – see Table 101
3. AC Electrical Characteristics – see Table 102 on page 343
4. Input Timing for External Interrupts, Ports 0 and 2 – see Figure 120 on page 344
5. Input Timing for Reset (nRESET Pin) – see Figure 121 on page 344
6. Input/Output Capacitance – see Table 103 on page 344
7. Data Retention Supply Voltage – see Table 104 on page 344
8. Stop Mode Release Timing Initiated by nRESET – see Table 104 on page 344
9. Stop Mode Release Timing Initiated by Interrupts – see Table 104 on page 344
10. A/D Converter Electrical Characteristics – see Table 105 on page 346
11. Low Voltage Reset Electrical Characteristics – see Table 106 on page 346
12. Low Voltage Reset Timing – see Figure 124 on page 347
13. Synchronous SIO Electrical Characteristics – see Table 107 on page 347
14. LCD Contrast Controller Electrical Characteristics – see Table 108 on page 348
15. Internal Watchdog Timer RC Electrical Characteristics – see Table 109 on page 348
16. Serial Data Transfer Timing – see Figure 125 on page 349
17. UART Timing Characteristics in Mode 0 – see Table 110 on page 349
18. Waveform for UART Timing Characteristics – see Figure 126 on page 350
19. Timing Waveform for the UART Module – see Figure 127 on page 350
20. Main Oscillator Characteristics – see Table 111 on page 351
21. Suboscillation Characteristics – see Table 112 on page 351
22. Main Oscillation Stabilization Time – see Table 113 on page 352
23. Clock Timing Measurement at X_{IN} – see Figure 128 on page 352
24. Suboscillation Stabilization Time – see Table 114 on page 352
25. Clock Timing Measurement at X_{TIN} – see Figure 129 on page 353
26. Operating Voltage Range – see Figure 130 on page 353
27. Internal Flash ROM Electrical Characteristics – see Figure 115 on page 354

Table 100. Absolute Maximum Ratings (T_A = 25°C)

Parameter	Symbol	Conditions	Rating TBD	Unit
Supply Voltage	V_{DD}	–	–0.3 to +6.5	V
Input Voltage	V_I	Ports 0–4	–0.3 to V_{DD} +0.3	V
Output Voltage	V_O	–	–0.3 to V_{DD} +0.3	V
Output Current High	I_{OH}	One I/O pin active	–5	mA
		All I/O pins active	–60	
Output Current Low	I_{OL}	One I/O pin active	+30 (peak value)	mA
		Total pin current for ports	+100 (peak value)	
Operating Temperature	T_A	–	–40 to +85	°C
Storage Temperature	T_{STG}	–	–65 to +150	°C

Table 101. DC Electrical Characteristics (T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V)[1]

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Operating Voltage	V_{DD}	f_X = 0.4–4.2MHz	1.8	–	5.5	V
		f_X = 0.4–12.0MHz	2.2	–	5.5	V
Input High Voltage	V_{IH1}	All ports except for V_{IH2}	0.8V_{DD}	–	V_{DD}	V
	V_{IH2}	X_{IN}, X_{OUT}, X_{TIN}, X_{TOUT}	V_{DD}–0.1	–	V_{DD}	V
Input Low Voltage	V_{IL1}	All ports except V_{IL2}	–	–	0.2 V_{DD}	V
	V_{IL2}	X_{IN}, X_{OUT}, X_{TIN}, X_{TOUT}	–	–	0.1	–
Output High Voltage	V_{OH1}	V_{DD} = 2.4V, P1.0–P1.1, P3.4–P3.6; I_{OH} = –1mA	V_{DD}–0.7	V_{DD}–0.7	–	V
	V_{OH2}	V_{DD} = 5V; P2; I_{OH} = –4mA	V_{DD}–1.0	V_{DD}–1.0	–	V
	V_{OH3}	V_{DD} = 5V; the other ports; I_{OH} = –1mA	V_{DD}–1.0	V_{DD}–1.0	–	V
Output Low Voltage	V_{OL1}	V_{DD} = 2.4V; P1.0–P1.1, P3.4–P3.6; I_{OL} = 12mA	–	0.3	0.5	V
	V_{OL2}	V_{DD} = 5V; P2; I_{OL} = 15mA	–	0.4	2.0	V
	V_{OL3}	V_{DD} = 5V; the other ports; IOL = 4mA	–	0.4	2.0	V

Notes:
1. Every value in this table is measured when bits 4–3 of the System Clock Control Register (CLKCON.4–.3) is set to 11b.
2. Supply current does not include current drawn through internal pull-up resistors, LCD voltage-dividing resistors, the LVR block, and external output current loads.
3. I_{DD1} and I_{DD2} include a power consumption of subclock oscillation.
4. I_{DD3} and I_{DD4} are the current when the main clock oscillation stops and the subclock is used.
5. I_{DD5} is the current when the main and subclock oscillation stops.

Table 101. DC Electrical Characteristics (T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V)[1]

Parameter	Symbol	Conditions		Min.	Typ.	Max.	Unit
Input High Leakage Current	I_{LIH1}	$V_{IN} = V_{DD}$; all input pins except for I_{LIH2}		–	–	3	µA
	I_{LIH2}	$V_{IN} = V_{DD}$; X_{IN}, X_{OUT}, X_{TIN}, X_{TOUT}		–	–	20	µA
Input Low Leakage Current	I_{LIL1}	V_{IN} = 0V; all input pins except for nRESET, I_{LIL2}		–	–	–3	µA
	I_{LIL2}	V_{IN} = 0V; X_{IN}, X_{OUT}, X_{TIN}, X_{TOUT}		–	–	–20	
Output High Leakage Current	I_{LOH}	$V_{OUT} = V_{DD}$; all output pins		–	–	3	µA
Output Low Leakage Current	I_{LOL}	V_{OUT} = 0V; all output pins		–	–	–3	µA
Pull-Up Resistors	R_{L1}	VI = 0V; T_A = 25°C, Ports 0–4	V_{DD} = 5V	25	50	100	kΩ
			V_{DD} = 3V	50	100	150	kΩ
	R_{L2}	V_{IN} = 0V, T_A = 25°C, nRESET	V_{DD} = 5V	150	250	400	kΩ
			V_{DD} = 3V	300	500	700	kΩ
Oscillator Feedback Resistors	R_{OSC1}	V_{DD} = 5V, T_A = 25°C, $X_{IN} = V_{DD}$, X_{OUT} = 0V		420	850	1700	kΩ
	R_{OSC2}	V_{DD} = 5V, T_A = 25°C, $X_{IN} = V_{DD}$, $X_{TIN} = V_{DD}$, X_{OUT} = 0V		2200	4500	9000	kΩ
LCD Voltage Dividing Resistor	R_{LCD}	T_A = 25°C		45	75	100	
VLCD-COMi Voltage Drop (i = 0 to 7)	V_{DC}	–15µA per common pin		–	–	120	
V_{LCD}–SEGx Voltage Drop (x = 0–18)	V_{DS}	–15µA per segment pin		–	–	120	mV

Notes:
1. Every value in this table is measured when bits 4–3 of the System Clock Control Register (CLKCON.4–.3) is set to 11b.
2. Supply current does not include current drawn through internal pull-up resistors, LCD voltage-dividing resistors, the LVR block, and external output current loads.
3. I_{DD1} and I_{DD2} include a power consumption of subclock oscillation.
4. I_{DD3} and I_{DD4} are the current when the main clock oscillation stops and the subclock is used.
5. I_{DD5} is the current when the main and subclock oscillation stops.

Table 101. DC Electrical Characteristics (T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V)[1]

Parameter	Symbol	Conditions		Min.	Typ.	Max.	Unit
Middle Output Voltage	V_{LC1}	V_{DD} = 2.7V to 5.5V, LCD clock = 0Hz, V_{LC0} = V_{DD}, ¼ Bias		0.75 V_{DD}–0.2	0.75 V_{DD}	0.75 V_{DD}+0.2	V
	V_{LC2}			0.5 V_{DD}–0.2	0.5 V_{DD}	0.5 V_{DD}+0.2	V
	V_{LC3}			0.25 V_{DD}–0.2	0.25 V_{DD}	0.25 V_{DD}+0.2	V
Supply Current[2]	I_{DD1}[3]	Run Mode; V_{DD} = 5.0V; crystal oscillator C1 = C2 = 22pF	4.2MHz	–	1.2	2.0	mA
			12.0MHz	–	2.2	4.0	mA
		V_{DD} = 3.0V	4.2MHz	–	0.8	1.5	mA
	I_{DD2}[3]	Idle Mode; V_{DD} = 5.0V; crystal oscillator C1 = C2 = 22pF	4.2MHz	–	0.8	1.5	mA
			12.0MHz	–	1.3	2.3	mA
		V_{DD} = 3.0V	4.2MHz	–	0.4	0.8	mA
	I_{DD3}[4]	Suboperating Mode; 32,768Hz crystal oscillator, V_{DD} = 3.0V, T_A = 25°C		–	80.0	120.0	µA
	I_{DD4}[4]	Subidle Mode; 32,768Hz crystal oscillator, V_{DD} = 3.0V, T_A = 25°C		–	6.0	15.0	µA
	I_{DD5}[5]	Stop Mode; V_{DD} = 5.0V		–	0.3	6.0	µA

Notes:
1. Every value in this table is measured when bits 4–3 of the System Clock Control Register (CLKCON.4–.3) is set to 11b.
2. Supply current does not include current drawn through internal pull-up resistors, LCD voltage-dividing resistors, the LVR block, and external output current loads.
3. I_{DD1} and I_{DD2} include a power consumption of subclock oscillation.
4. I_{DD3} and I_{DD4} are the current when the main clock oscillation stops and the subclock is used.
5. I_{DD5} is the current when the main and subclock oscillation stops.

Table 102. AC Electrical Characteristics (T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V)

Parameter	Symbol	Conditions	Min.*	Typ.	Max.	Unit
Interrupt input high, low width (P3.0–P3.7)	t_{INTH}, t_{INTL}	All interrupt, V_{DD} = 5V	500	–	–	ns
nRESET input low width	t_{RSL}	Input, V_{DD} = 5V	10	–	–	ns

Note: If the width of the interrupt or reset pulse is greater than the minimum value, the pulse is always recognized as a valid pulse.

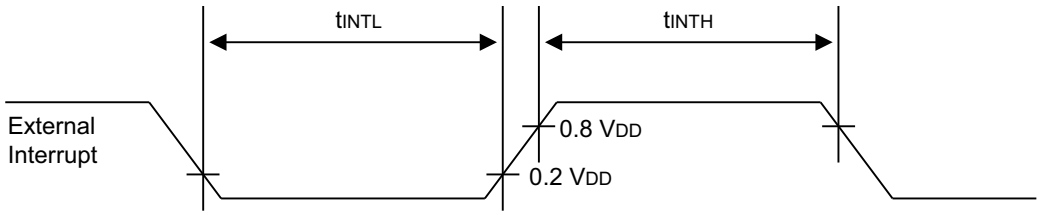


Figure 120. Input Timing for External Interrupts

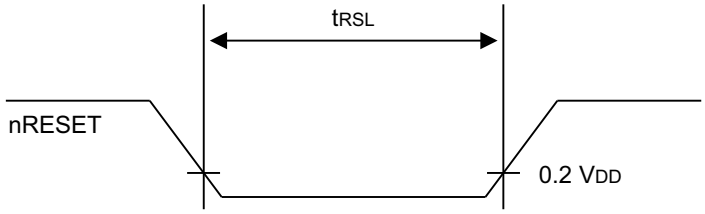


Figure 121. Input Timing for nRESET

Table 103. Input/Output Capacitance (T_A = –40°C to +85°C, V_{DD} = 0V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Input capacitance	C_{IN}	f = 1MHz; unmeasured pins are returned to V_{SS}	–	–	10	pF
Output capacitance	C_{OUT}	–	–	–	–	–
I/O capacitance	C_{IO}	–	–	–	–	–

Table 104. Data Retention Supply Voltage in Stop Mode (T_A = –40°C to +85°C)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Data retention supply voltage	V_{DDDR}	–	1.8	–	5.5	V
Data retention supply current	I_{DDDR}	Stop Mode, T_A = 25°C, V_{DDDR} = 1.8V	–	–	1	µA

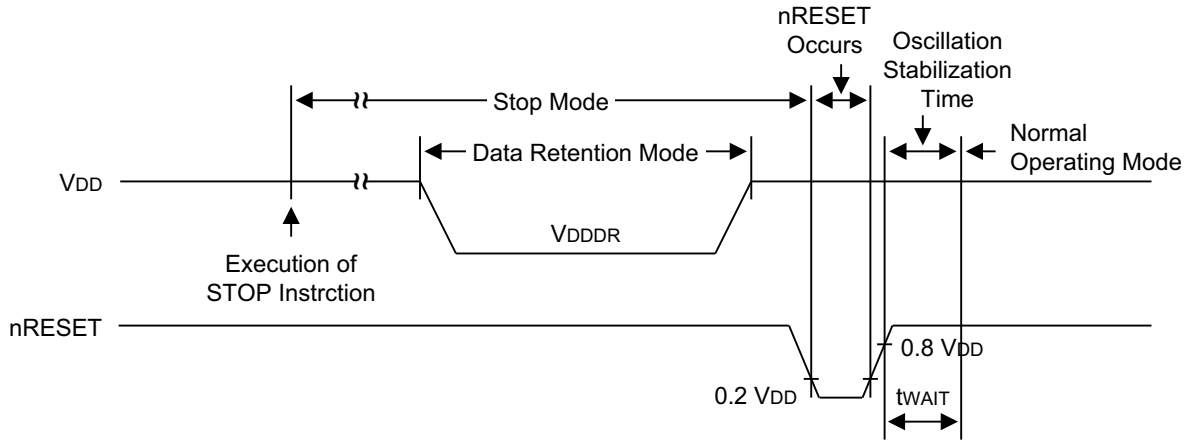


Figure 122. Stop Mode Release Timing Initiated by nRESET

Note: In Figure 122, t_{WAIT} is the same as 4096 x 16 x 1 ÷ f_{XX}.

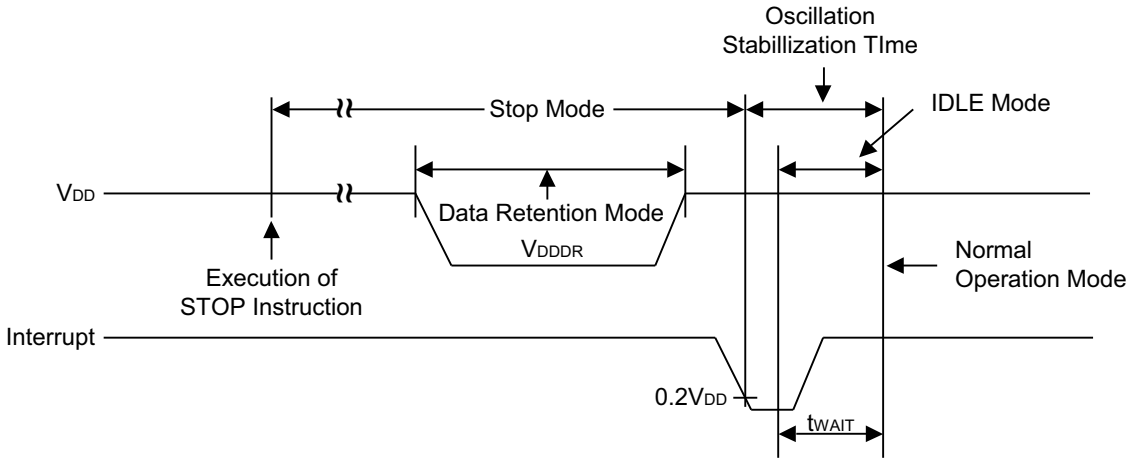


Figure 123. Stop Mode Release Timing Initiated by Interrupts

Note: In Figure 123, t_{WAIT} is the same as 16 x 1 / f_{BT}; f_{BT} = the basic timer clock selection.

Table 105. A/D Converter Electrical Characteristics
(T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V, V_{SS} = 0V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Resolution	–	–	–	10	–	bit
Total accuracy	–	–	–	–	±3	LSB
Integral linearity error	ILE	V_{DD} = 5.120V V_{SS} = 0V CPU clock = 12.0MHz	–		±2	LSB
Differential linearity error	DLE		–		±1	LSB
Offset error of top	EOT		–		±3	LSB
Offset error of bottom	EOB				±3	LSB
Conversion time[1]	T_{CON}	–	25	–	–	µs
Analog input voltage	V_{IAN}	–	V_{SS}	–	AV_{REF}	V
Analog input impedance	R_{AN}	–	2	1000	–	MΩ
Analog reference voltage	AV_{REF}	–	1.8	–	V_{DD}	V
Analog input current	I_{ADIN}	V_{DD} = 5.0V	–	–	10	µA
Analog block current[2]	I_{ADC}	V_{DD} = 5.0V	–	0.5	1.5	mA
		V_{DD} = 5.0V when in Power Down Mode	–	100	500	nA

Notes:
1. *Conversion time* is the time required from the moment a conversion operation starts until it ends.
2. I_{ADC} is an operating current during A/D converter.

Table 106. Low Voltage Reset Electrical Characteristics
(T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V)*

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Voltage of LVR	V_{LVR}	T_A = 25°C	1.8	1.9	2.0	V
			2.4	2.6	2.8	V
			3.1	3.3	3.5	V
			3.7	3.9	4.1	V
V_{DD} voltage rising time	t_R	–	10	–	–	µs
V_{DD} voltage off time	t_{OFF}	–	0.5	–	–	s

Note: *The current of the LVR circuit is consumed when LVR is enabled by the Smart Option.

Table 106. Low Voltage Reset Electrical Characteristics (T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V)*

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Hysteresis	ΔV	–	–	50	150	mV
Current consumption of LVR	I_{LVR}	V_{DD} = 3.0V	–	30	60	μA

Note: *The current of the LVR circuit is consumed when LVR is enabled by the Smart Option.

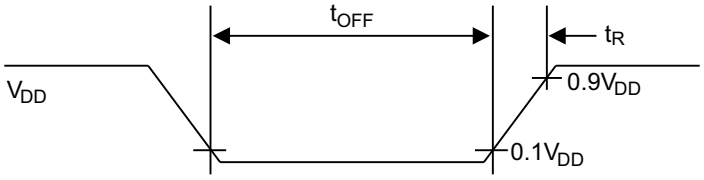


Figure 124. Low Voltage Reset Timing

Table 107. Synchronous SIO Electrical Characteristics (T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Voltage of BLD	V_{BLD}	T_A = 25°C	2.2	2.4	2.6	V
			2.5	2.7	2.9	V
			3.1	3.3	3.5	V
			3.7	3.9	4.1	V
Hysteresis	ΔV	–	–	10	100	mV
Current consumption of BLD	I_{BLD}	V_{DD} = 3.0V	–	30	60	μA
Response time	T_{BLD}	f_W = 32.768kHz	–	–	1	ms
SCK cycle time	t_{KCY}	External SCK source	1000	–	–	ns
		Internal SCK source	1000	–	–	ns
SCK high, low width	t_{KH}, t_{KL}	External SCK source	500	–	–	ns
		Internal SCK source	t_{KCY}/2–50	–	–	ns
SI setup time to SCK high	t_{SIK}	External SCK source	250	–	–	ns
		Internal SCK source	250	–	–	ns

Table 107. Synchronous SIO Electrical Characteristics (T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
SI hold time to SCK high	t_{KSI}	External SCK source	400	–	–	ns
		Internal SCK source	400	–	–	ns
Output delay for SCK to SO	t_{KSO}	External SCK source	–	–	–	ns
		Internal SCK source	–	–	–	ns

Table 108. LCD Contrast Controller Electrical Characteristics (T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Resolution	–	T_A = 25°C	–	–	4	bit
Linearity	R_{LIN}	V_{DD} = 5.0V, ¼ bias	–	–	±1.0	LSB
Max output voltage	V_{LPP}	V_{LC0} = V_{DD} = 5.0V, the brightest level	4.9	–	V_{LC0}	V

Table 109. Internal Watchdog Timer RC Oscillator Electrical Characteristics (T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V)

Parameter	Symbol	Conditions		Min.	Typ.	Max.	Unit
Frequency	f_{WDTRC}	Normal/ Idle Mode	T_A = 25°C	16	32	48	kHz
		Stop Mode		9	18	27	
Stabilization time	T_{WDTS}	–		–	–	500	µs
Current consumption	I_{WDTRC}	Enable		–	1	3	µA
		Disable		–	–	0.1	

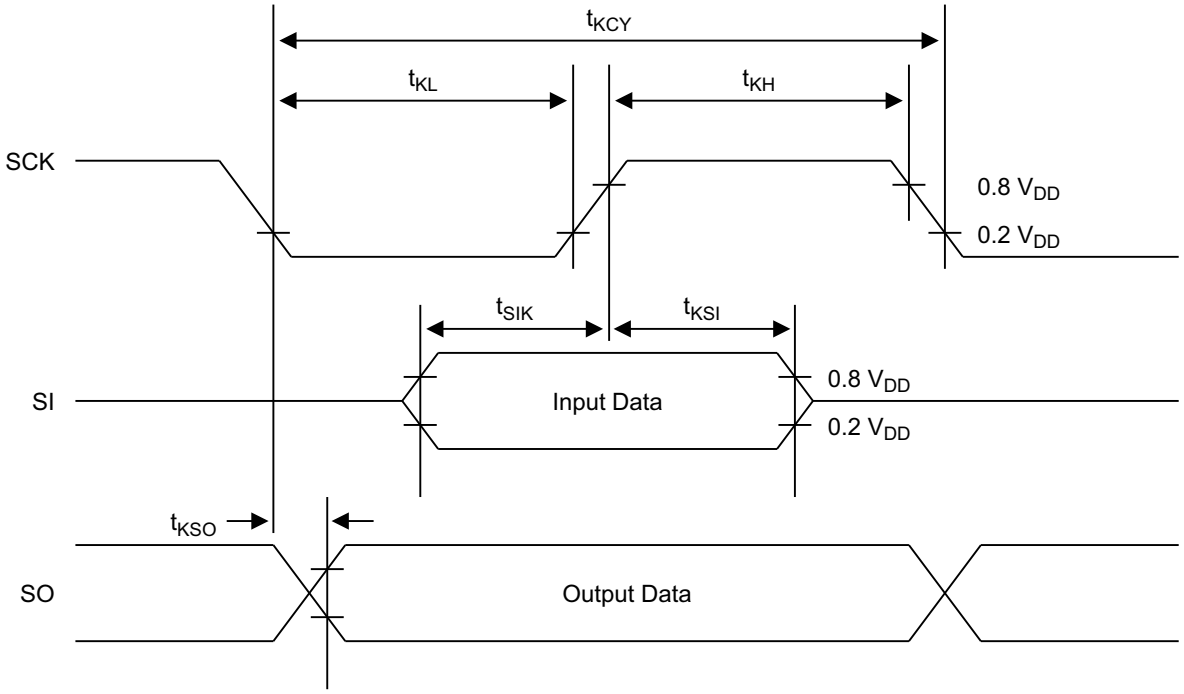


Figure 125. Serial Data Transfer Timing

Table 110. UART Timing Characteristics in Mode 0[1]
(12.0 MHz; T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V; Load Capacitance = 80pF)

Parameter	Symbol	Min.	Typ.	Max.	Unit
Serial port clock cycle time[2]	t_{SCK}	1160	t_{CPU} x 16	1500	ns
Output data setup to clock rising edge	t_{S1}	500	t_{CPU} x 13	–	ns
Clock rising edge to input data valid	t_{S2}	–	–	500	ns
Output data hold after clock rising edge[2]	t_{H1}	t_{CPU}–50	t_{CPU}	–	ns
Input data hold after clock rising edge	t_{H2}	0	–	–	ns
Serial port clock High, Low level width[2]	t_{HIGH}, t_{LOW}	450	t_{CPU} x 8	890	ns

Notes:
1. All timings are in nanoseconds (ns) and assume a 12.0MHz CPU clock frequency.
2. t_{CPU} = 1 UART clock period.

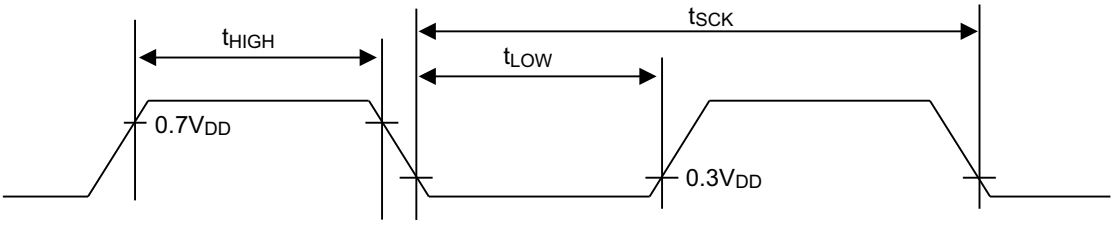


Figure 126. Waveform for UART Timing Characteristics

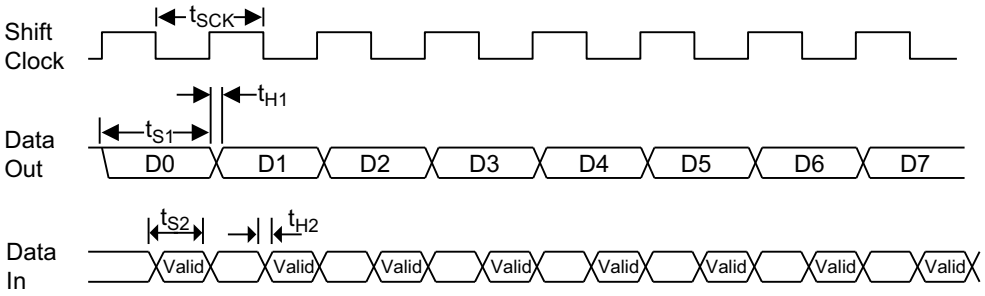


Figure 127. Timing Waveform for the UART Module

Notes: The symbols shown in Figure 127 are defined as:

- f_{SCK} = Serial port clock cycle time.
- t_{S1} = Output data setup to clock rising edge.
- t_{S2} = Clock rising edge to input data valid.01
- t_{H1} = Output data hold after clock rising edge.
- t_{H2} = Input data hold after clock rising edge.

Table 111. Main Oscillator Characteristics (T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V)

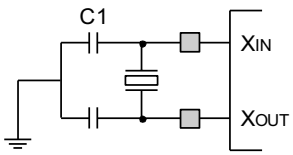
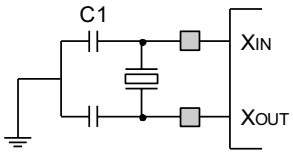
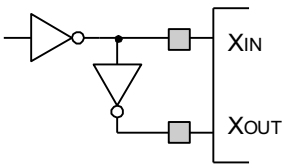
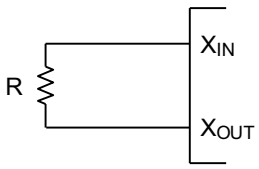
Oscillator	Clock Configuration	Parameter	Test Condition	Min.	Typ.	Max.	Unit
Crystal		Main oscillation frequency	2.2V–5.5V	0.4	–	12.0	MHz
			1.8V–5.5V	0.4	–	4.2	MHz
Ceramic oscillator		Main oscillation frequency	2.2V–5.5V	0.4	–	12.0	MHz
			1.8V–5.5V	0.4	–	4.2	MHz
External clock		X_{IN} input frequency	2.2V–5.5V	0.4	–	12.0	MHz
			1.8V–5.5V	0.4	–	4.2	MHz
RC oscillator		Frequency	5.0V	0.4	–	2.0	MHz
			3.0V	0.4	–	1.0	MHz

Table 112. Suboscillation Characteristics (T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V)

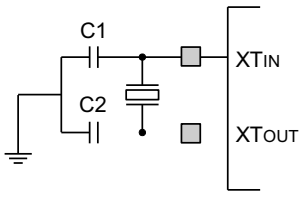
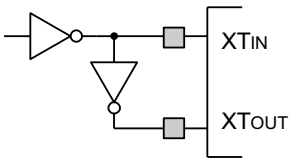
Oscillator	Clock Configuration	Parameter	Test Condition	Min.	Typ.	Max.	Unit
Crystal		Suboscillation frequency	1.8V–5.5V	–	32.768	–	kHz
External clock		X_{TIN} frequency	1.8V–5.5V	32	–	100	kHz

Table 113. Main Oscillation Stabilization Time (T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V)

Parameter	Test Condition	Min.	Typ.	Max.	Unit
Crystal	f_X > 1MHz; oscillation stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	40	ms
Ceramic		–	–	10	ms
External clock	X_{IN} input high and low width (t_{XH}, t_{XL})	62.5	–	1250	ns

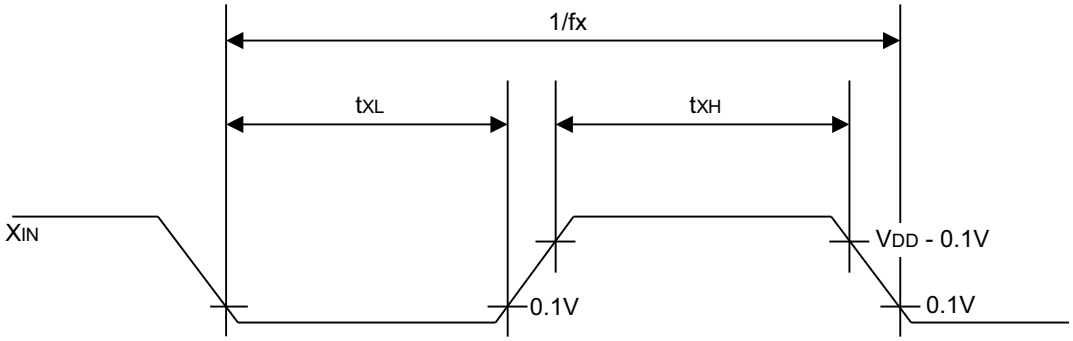


Figure 128. Clock Timing Measurement at X_{IN}

Table 114. Suboscillation Stabilization Time (T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V)

Parameter	Test Condition	Min.	Typ.	Max.	Unit
Crystal	–	–	–	10	s
External clock	X_{TIN} input high and low width (t_{XTH}, t_{XTL})	5	–	15	µs

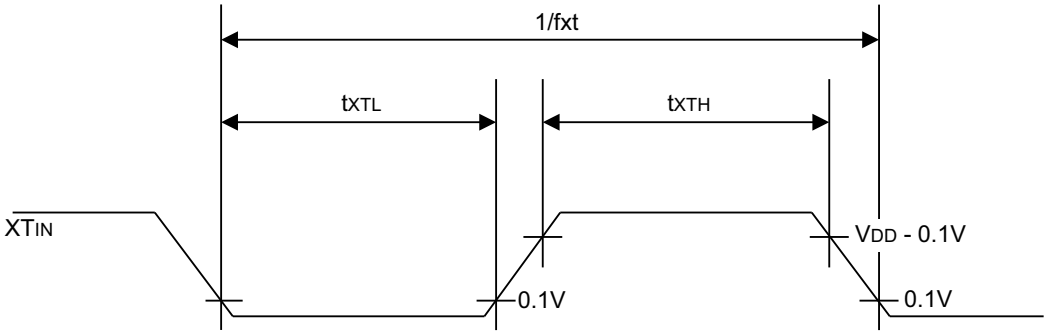
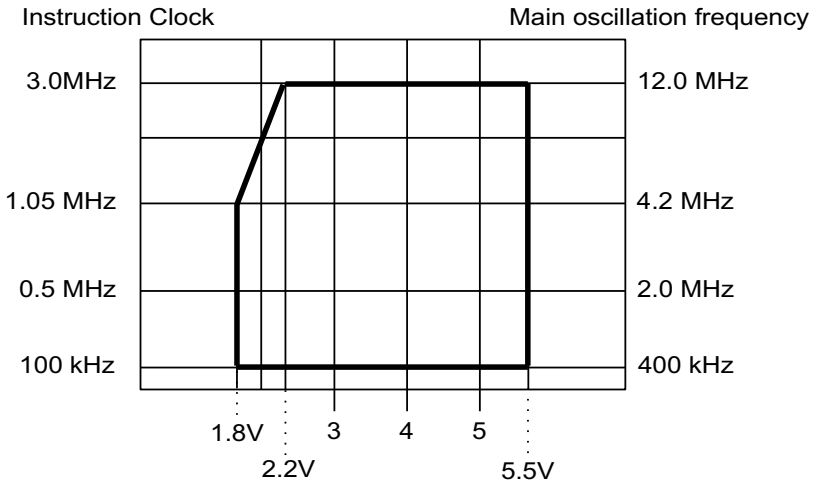


Figure 129. Clock Timing Measurement at X_{TIN}



CPU Clock = 1/4n x oscillator frequency (n = 1, 2, 8, 16)

Figure 130. Operating Voltage Range

Table 115. Internal Flash ROM Electrical Characteristics (T_A = –40°C to +85°C, V_{DD} = 1.8V to 5.5V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Programming time[1]	Ftp	–	20	25	30	µs
Chip erasing time[2]	Ftp1	–	32	50	70	ms
Sector erasing time[3]	Ftp2	–	4	8	12	ms
Number of writes/erases	FN_{WE}	–	–	–	10000[4]	times

Notes:
1. Programming time = the time during which one byte (8-bit) is programmed.
2. Chip erasing time = the time during which the entire 64KB block is erased.
3. Sector erasing time = the time during which the entire 128-byte block is erased.
4. Chip erasing is available in Tool Program Mode only.

Chapter 26. Mechanical Data

The S3F8S45 microcontroller is currently available in 44-pin QFP, 42-pin SDIP, and 32-pin SDIP packages. A mechanical drawing of the 44-pin QFP package is shown in Figure 131.

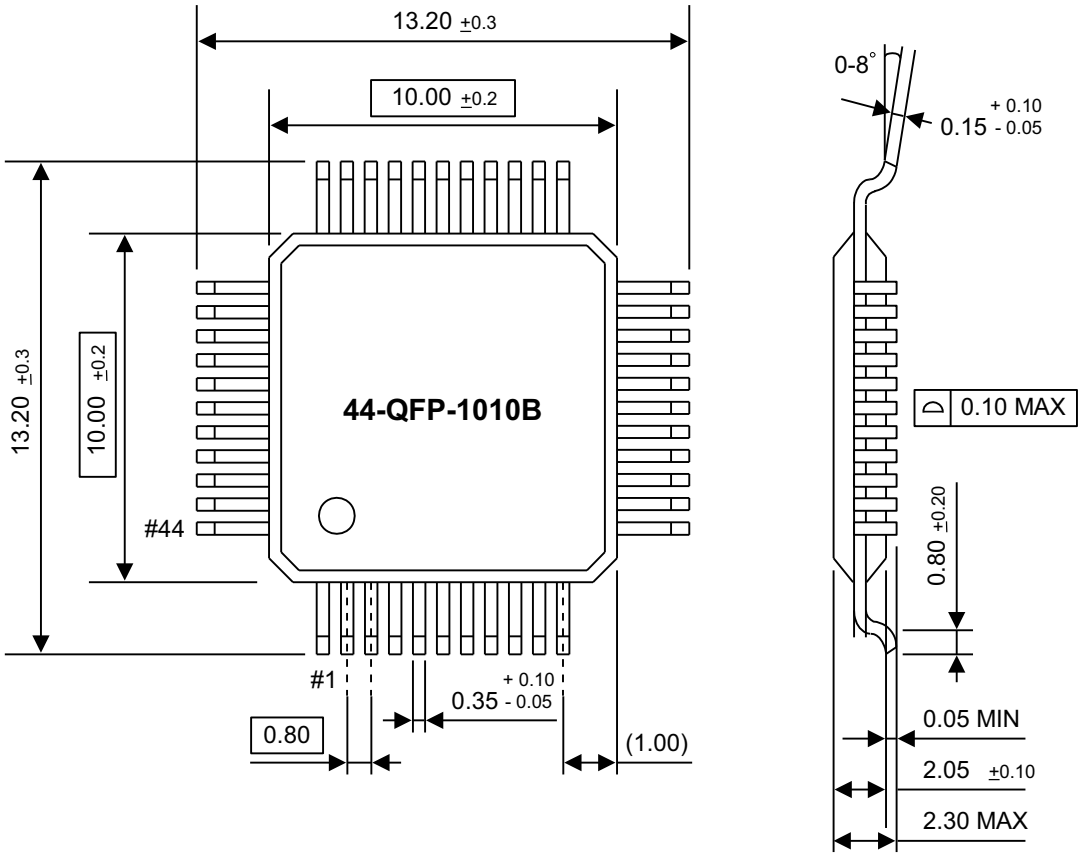


Figure 131. Package Dimensions, 44-Pin QFP Package

Note: Dimensions in Figure 131 are expressed in millimeters.

A mechanical drawing of the 42-pin SDIP package is shown in Figure 132.

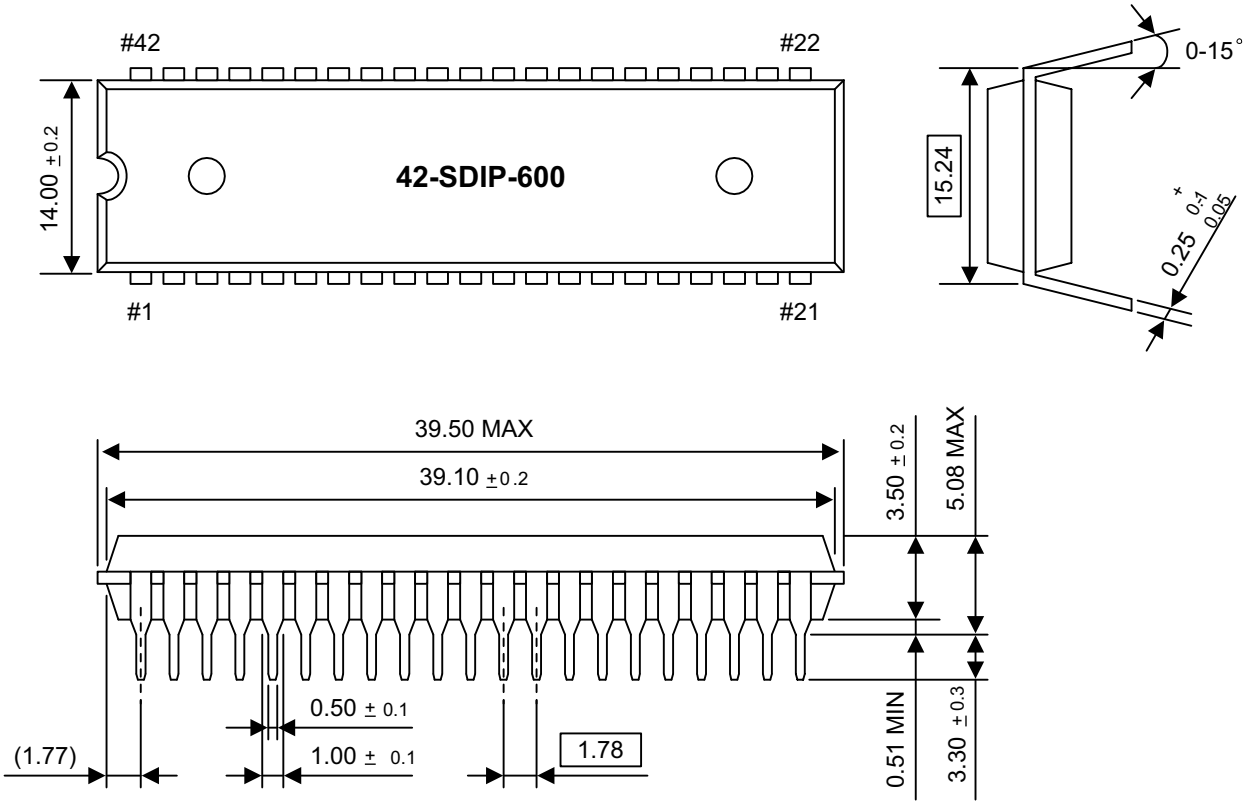


Figure 132. Package Dimensions, 42-pin SDIP Package

Note: Dimensions in Figure 132 are expressed in millimeters.

A mechanical drawing of the 32-pin SDIP package is shown in Figure 133.

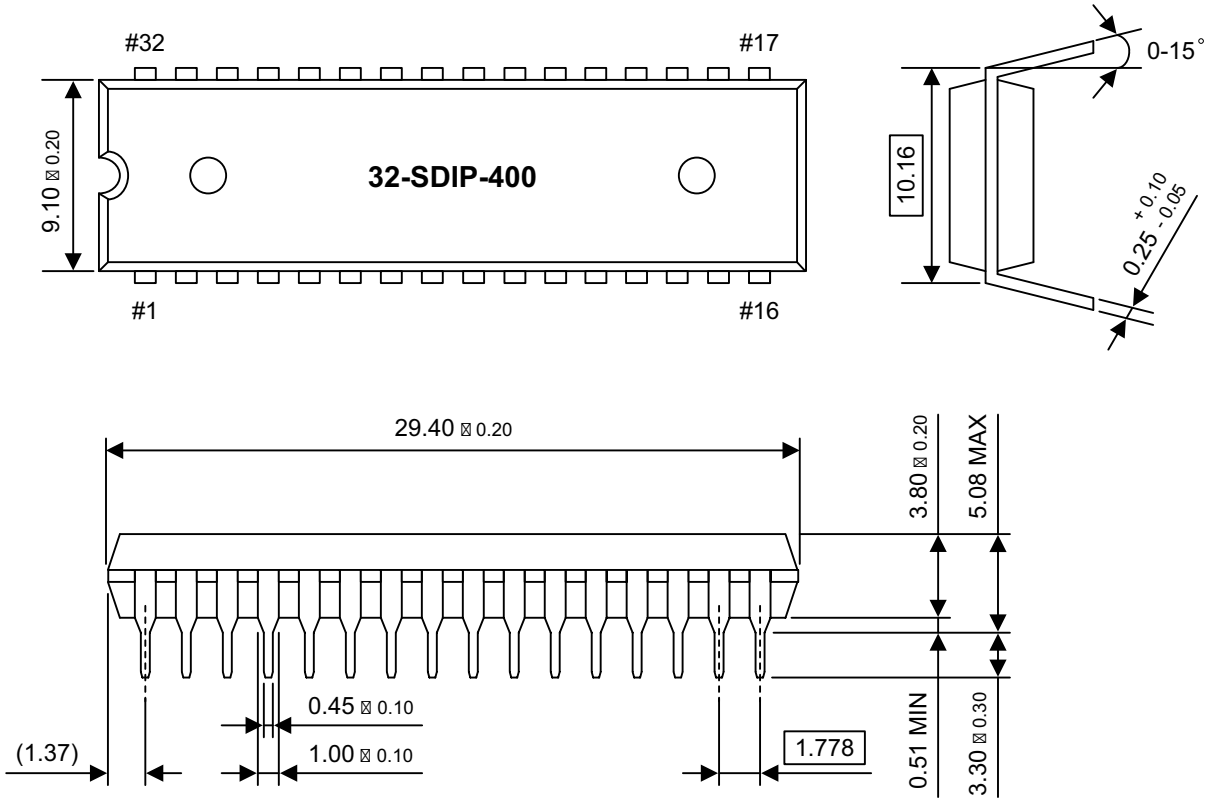


Figure 133. Package Dimensions, 32-Pin SDIP Package

Note: Dimensions in Figure 133 are expressed in millimeters.

Chapter 27. S3F8S45 Flash MCU

This chapter describes Tool Program Mode operation for the S3F8S45 Flash MCU. To learn more about User Program Mode operation, refer to the Embedded Flash Memory Interface chapter on page 327.

The S3F8S45 microcontroller features an on-chip Flash MCU ROM which is accessed by serial data format.

Pin assignments for the 44-pin QFP package are shown in Figure 134.

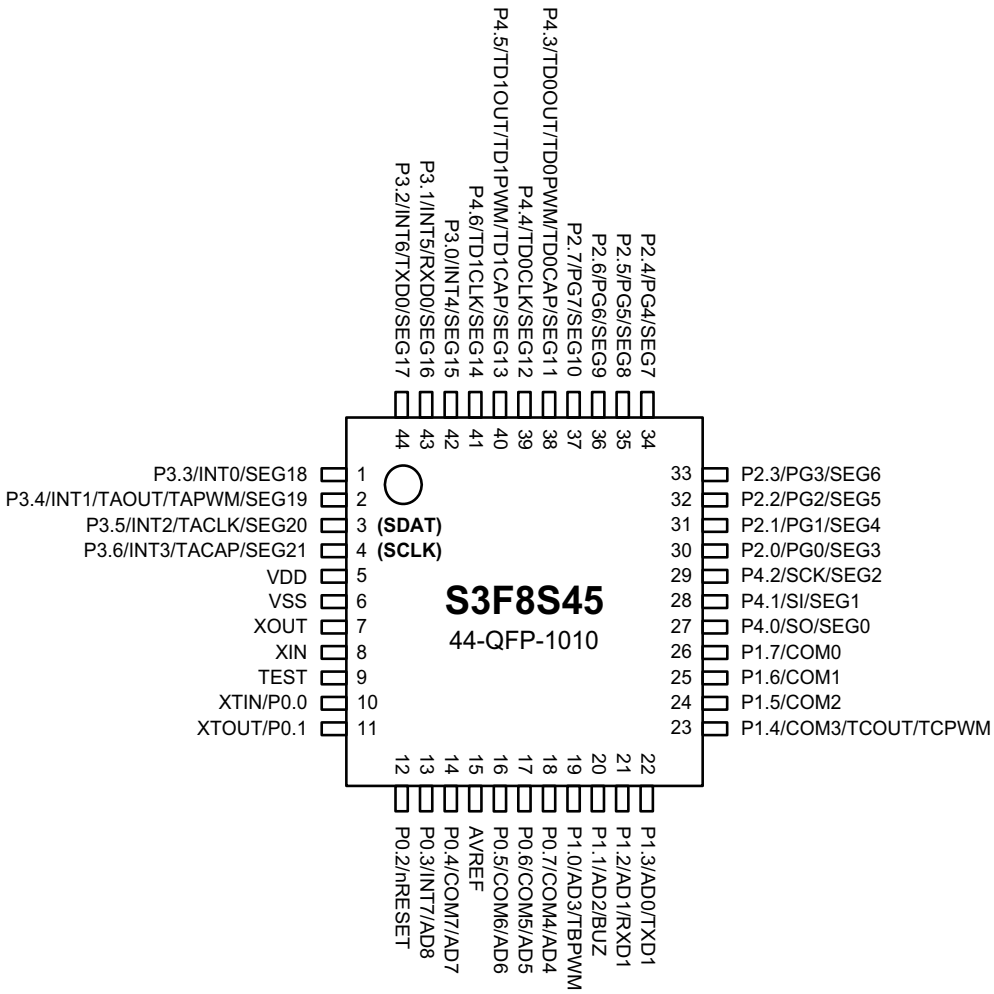


Figure 134. S3F8S45 Pin Assignments, 44-QFP Package

Pin assignments for the 42-pin SDIP package are shown in Figure 135.

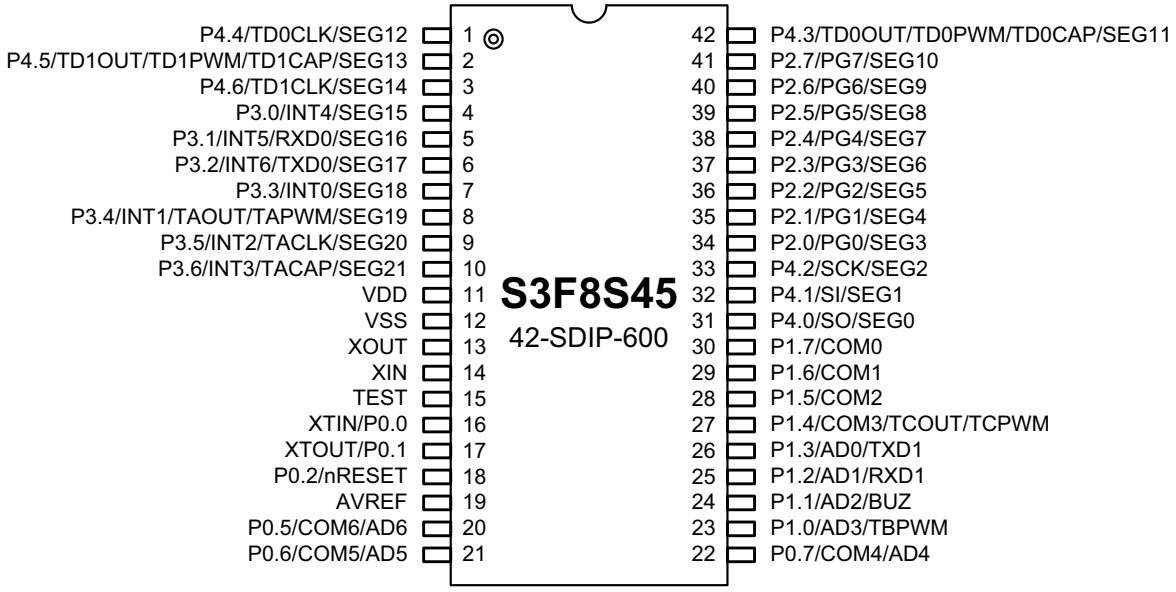


Figure 135. S3F8S45 Pin Assignments, 42-SDIP Package

Pin assignments for the 32-pin SDIP package are shown in Figure 136.

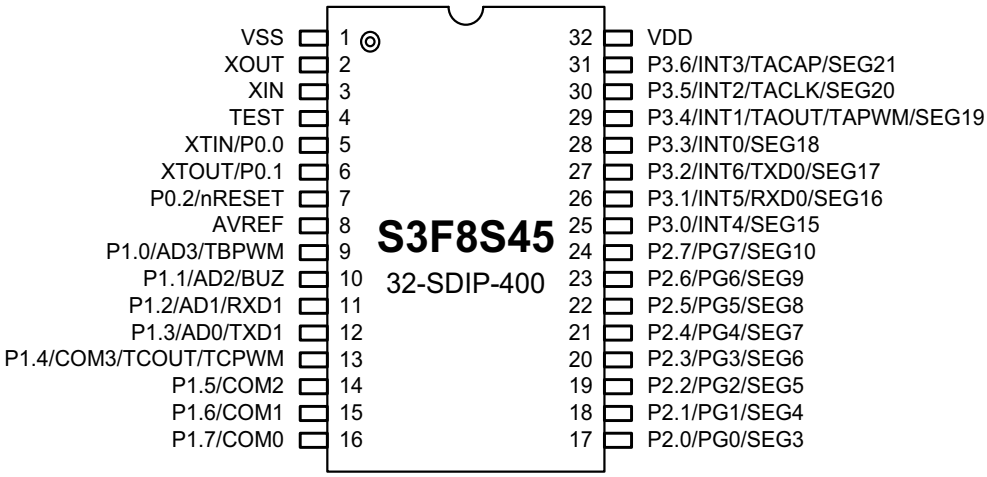


Figure 136. S3F8S45 Pin Assignments, 32-SDIP Package

Table 116. Pins Used to Read/Write the Flash ROM

Main Chip		During Programming				
		Pins				
Pin Name	Pin Name	44-Pin	42-Pin	32-Pin	I/O	Function
P3.5	SDAT	3	9	30	I/O	Serial data pin; output port when reading and input port when writing. Can be assigned as an input/push-pull output port.
P3.6	SCLK	4	10	31	I/O	Serial clock pin; input only.
TEST	V_{PP}	9	15	4	I	Tool Mode selection when TEST/ V_{PP} pin sets logic value 1. If using Flash writer in Tool Mode (e.g., spw2+, etc.), connect the TEST/V_{PP} pin to V_{DD}. The S3F8S45 MCU supplies 12.5V via the internal high-voltage generation circuit.
nRESET	nRESET	12	18	7	I	Chip initialization.
X_{IN}	X_{IN}	8	14	3	–	When P0.2/nRESET is configured as P0.2 by Smart Options, this pin should be connected to V_{SS} during programming.
V_{DD}	V_{DD}	5	11	31	–	Power supply pin for logic circuit. V_{DD} should be tied to 5V during programming.
V_{SS}	V_{SS}	6	12	1	–	

27.1. Test Pin Voltage

The TEST pin on the socket board for the MTP Writer must be connected to V_{DD} (5.0V) with RC delay as shown in Figure 137 (only when SPW 2+ and GW-pro2 are used). The TEST pin on this socket board must not be connected to V_{PP} (12.5V), which is generated from the MTP Writer. Therefore, the specific socket board for the S3F8S45 MCU must be used when writing or erasing using the MTP Writer.

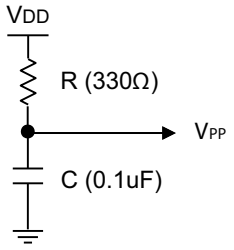


Figure 137. RC Delay Circuit

27.2. Onboard Writing

The S3F8S45 requires only six signal lines, including the V_{DD} and V_{SS} pins, for writing internal Flash memory with the serial protocol. Therefore, onboard writing is possible if the writing signal lines are considered when the application board is designed.

27.3. Circuit Design Guide

As Flash memory is being written to, the writing tool requires the following six signal lines: V_{SS}, V_{DD}, nRESET, TEST, SDAT, and SCLK. When designing the PCB circuits, consider how these signal lines will be used for onboard writing operations. The TEST pin is normally connected to V_{SS}; however, in writing mode, a resistor should be inserted between the TEST pin and V_{SS}. The nRESET, SDAT, and SCLK lines should be treated with the same consideration.

when designing your application board, be careful that you design the circuitry related to these signal pins, because the rise and fall timing of the V_{PP}, SCLK, and SDAT lines will be very important for proper programming. See Figure 138.

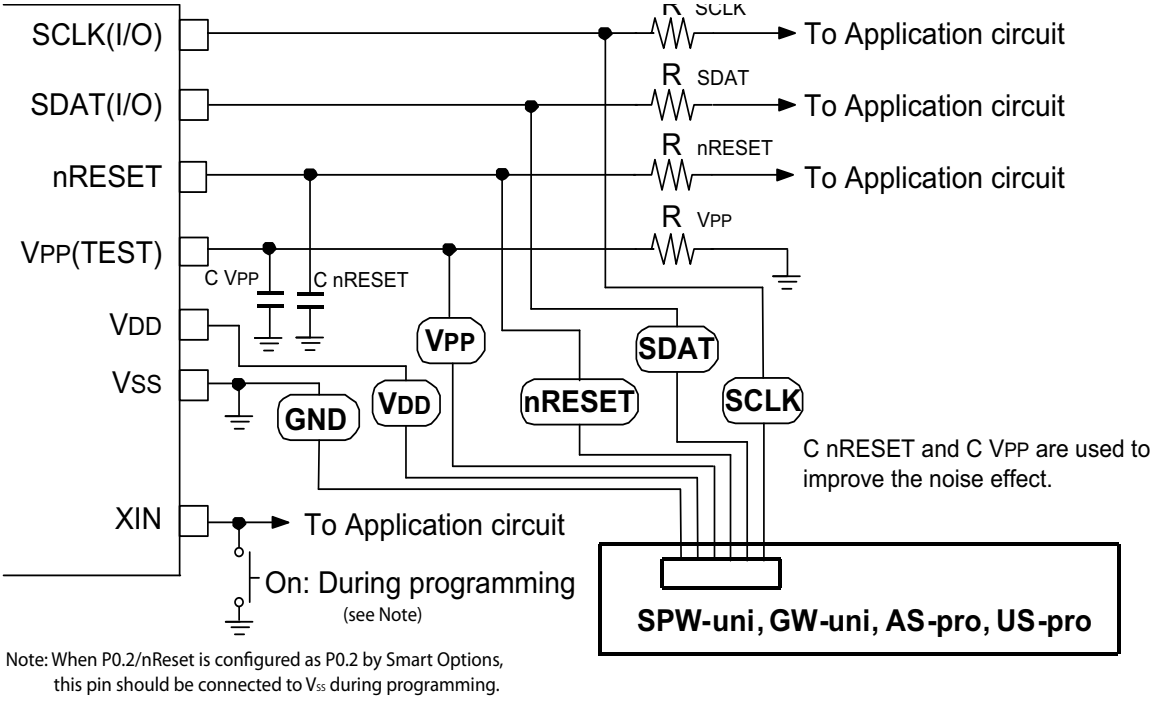


Figure 138. PCB Design Guide for on Board Programming

Note: If the writer tool you are using is the SPW 2+ or the GW-pro2, refer to Figure 137 on page 360.

Table 117. Circuit Connections

Pin Name	I/O Mode in Application	Resistor Required?	Required Value
V_{PP} (TEST)	Input	Yes	RV_{PP} is 10kΩ to 50kΩ. CV_{PP} is 0.01µF to 0.02µF.
nRESET	Input	Yes	RnRESET is 2kΩ to 5kΩ. CnRESET is 0.01µF to 0.02µF.
SDAT(I/O)	Input	Yes	RSDAT is 2kΩ to 5kΩ.
	Output	No	–
SCLK(I/O)	Input	Yes	RSDAT is 2kΩ to 5kΩ.
	Output	No	–

Note: In Onboard Writing Mode, a very high-speed signal will be provided to the SCLK and SDAT pins that can cause damage to the application circuits connected to the SCLK or SDAT ports if the application circuit is designed for high-speed response, such as a relay control circuit. If possible, the I/O configuration of the SDAT and SCLK pins must set to Input Mode. The value of R, C in this table is the recommended value; this value varies with the system circuitry.

Chapter 28. Development Tools

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

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

28.1. Development System Configuration

Figure 139 shows the basic configuration of the development system.

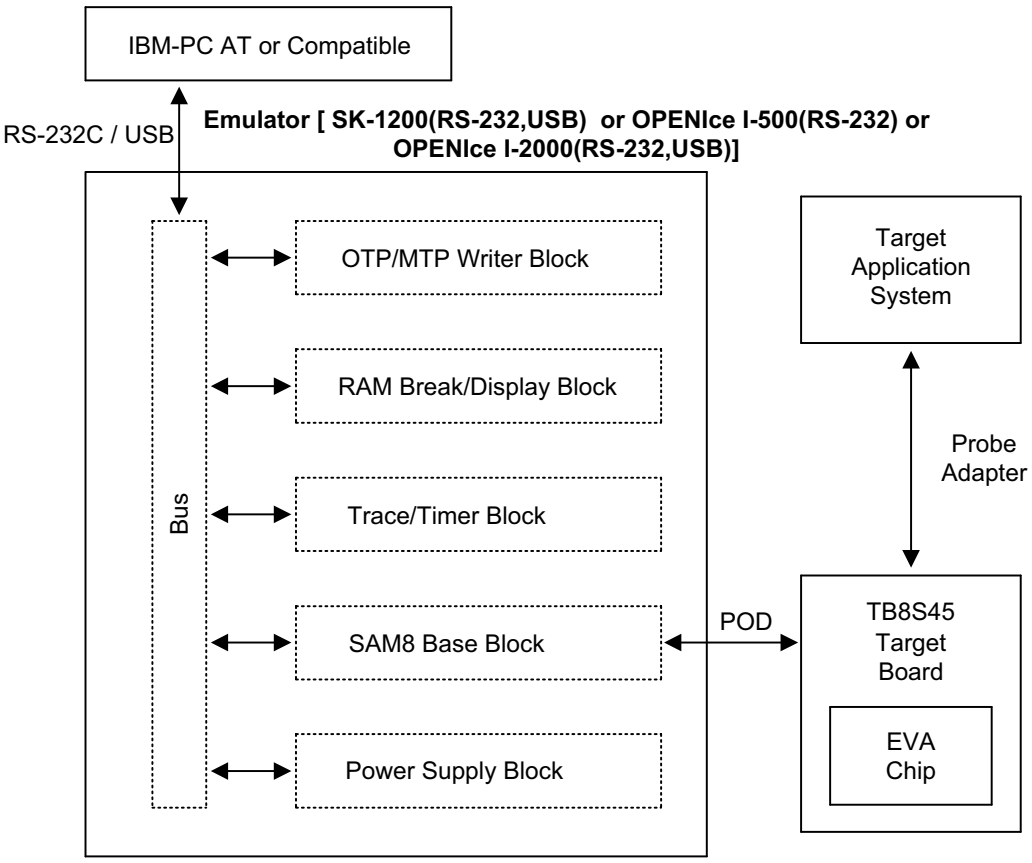


Figure 139. Development System Configuration

28.2. Target Board

The TB8S45 Target Board is specific to the S3F8S45 MCU, and ships complete with all target system cables and adapters. This target board is operated as a target CPU with an emulator (OPENIce I-500/2000, SK-1200).

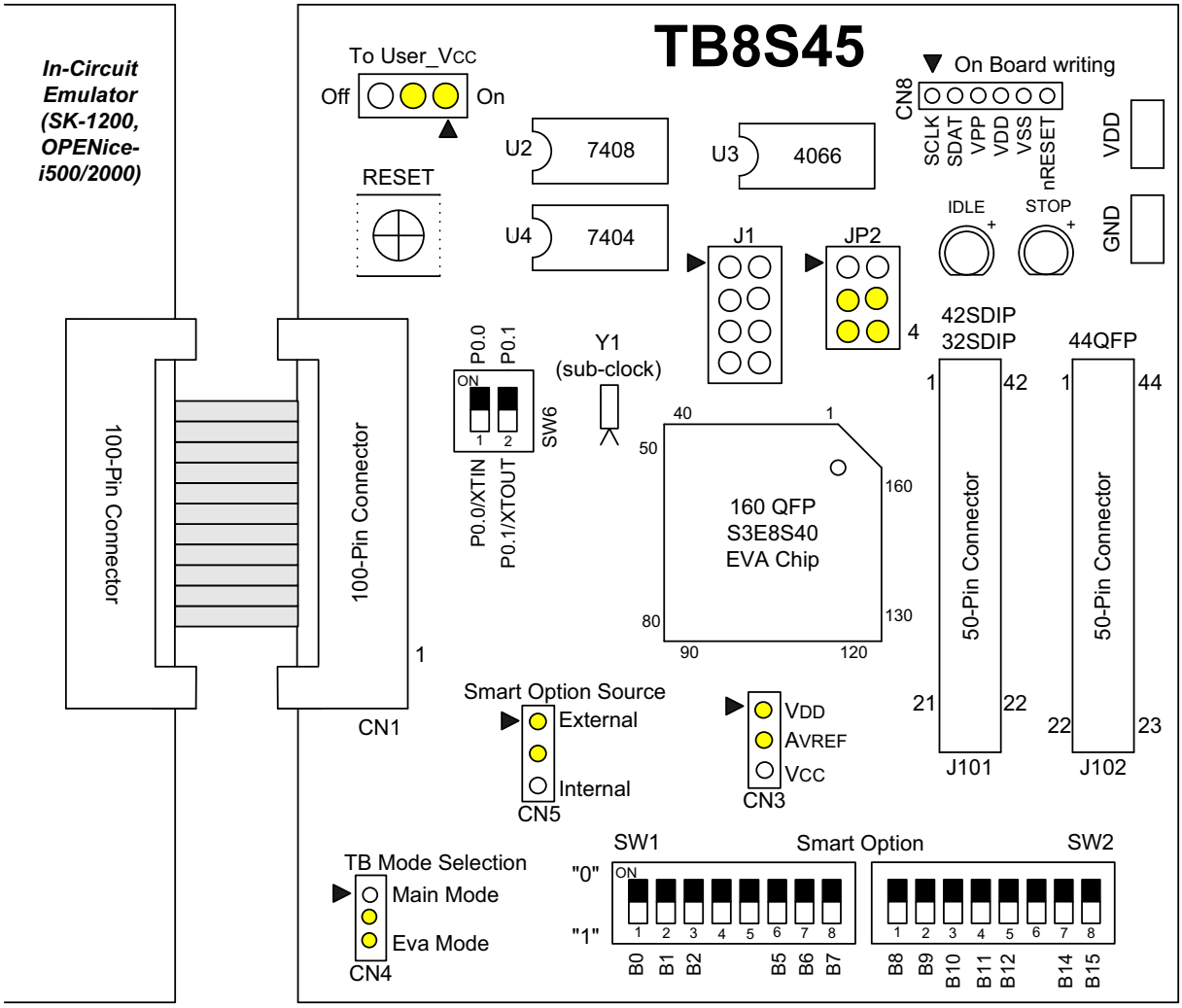


Figure 140. TB8S45 Target Board Configuration

Notes: **IDLE LED.** This LED is ON when the S3E8S40 evaluation chip is in Idle Mode.

STOP LED. This LED is ON when the S3E8S40 evaluation chip is in Stop Mode.

Table 118 lists the power selection settings for the TB8S45 target board.

Table 118. TB8S45 Target Board Components

Symbol	Usage	Description
CN1	100-pin connector	Connection between emulator and TB8S45 target board.
J101, J102	50-pin connector	Connection between target board and user application system.
RESET	Push button	Generation low active reset signal to S3F8S45 EVA-chip.
V_{DD}, GND	POWER connector	External power connector for the TB8S45 target board.
STOP, IDLE LED	STOP/IDLE Display	Indicates the STOP or IDLE status of the S3F8S45 EVA chip on the TB8S45 target board.
CN8	Flash serial programming	Signal points for programming Flash ROM by the external programmer. Do not use when in User Mode.
SW6	P0.0/X_{IN} and P0.1/X_{OUT} selection	When used by P0.0/P0.1, turn this switch ON. When used by X_{IN}/X_{OUT}, turn this switch OFF.

Table 119 lists the power selection settings for the TB8S45 target board.

Table 119. TB8S45 Target Board Jumper Settings

JP#	Description	1-2 Connection	2-3 Connection	Default Setting
JP2	Clock source selection.	When using the internal clock source which is generated from the emulator, join connector 2-3 and the 4-5 pin. To use the external clock source as a crystal, change the jumper setting from 1-2 to 5-6 and connect J1 to an external clock source.		Emulator 2-3 4-5
J1	External clock source.	Connecting points for the external clock source.		
CN5	Smart Option source selection.	The Smart Option is selected with an external Smart Option switch (SW1 & SW2).	The Smart Option is selected by an internal Smart Option area (i.e., the 003Eh–003Fh address range of ROM). However, this selection is not available.	Join 1-2

Table 119. TB8S45 Target Board Jumper Settings (Continued)

JP#	Description	1-2 Connection	2-3 Connection	Default Setting
SW1, SW2	Smart Option selection.	The Smart Option can be selected by this switch when the Smart Option source is selected externally. B7–B0 are comparable to 003Eh.7–.0. B15–B8 are comparable to 003Fh.7–.0. To learn more, see the Smart Option section on page 22.		
CN4	Target board mode selection.	Main Mode	EVA Mode	Join 2-3
To User_ V_{CC}	Target system is supplied V_{DD}.	V_{DD} to the Target Board is not supplied from the user's system.	V_{DD} to the Target Board is supplied from the user's system.	Join 2-3
CN3	AV_{REF} power source.	V_{DD}	User power	Join 1-2

Figure 141 shows the pin assignments for one of the TB8S45 Target Board's two 50-pin connector, J102.

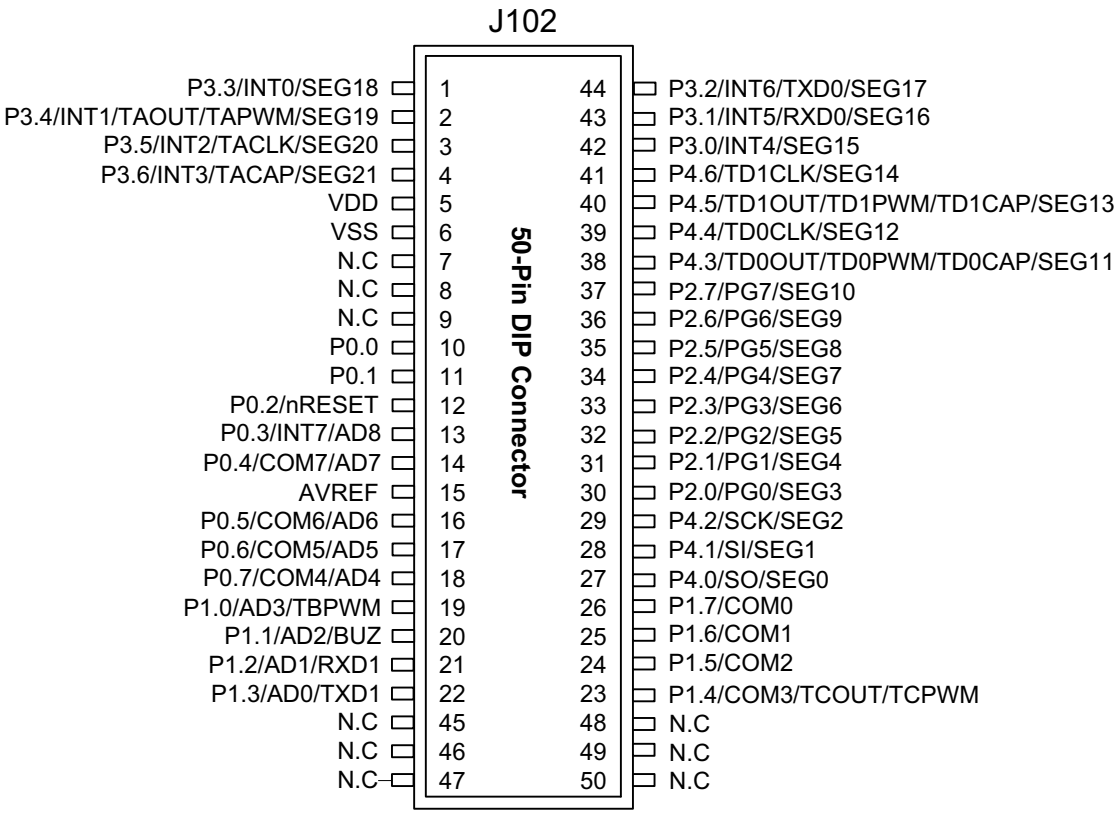


Figure 141. TB8S45 50-Pin J102 Connector

Figure 142 shows the pin assignments for the second of the TB8S45 Target Board's two 50-pin connector, J101.

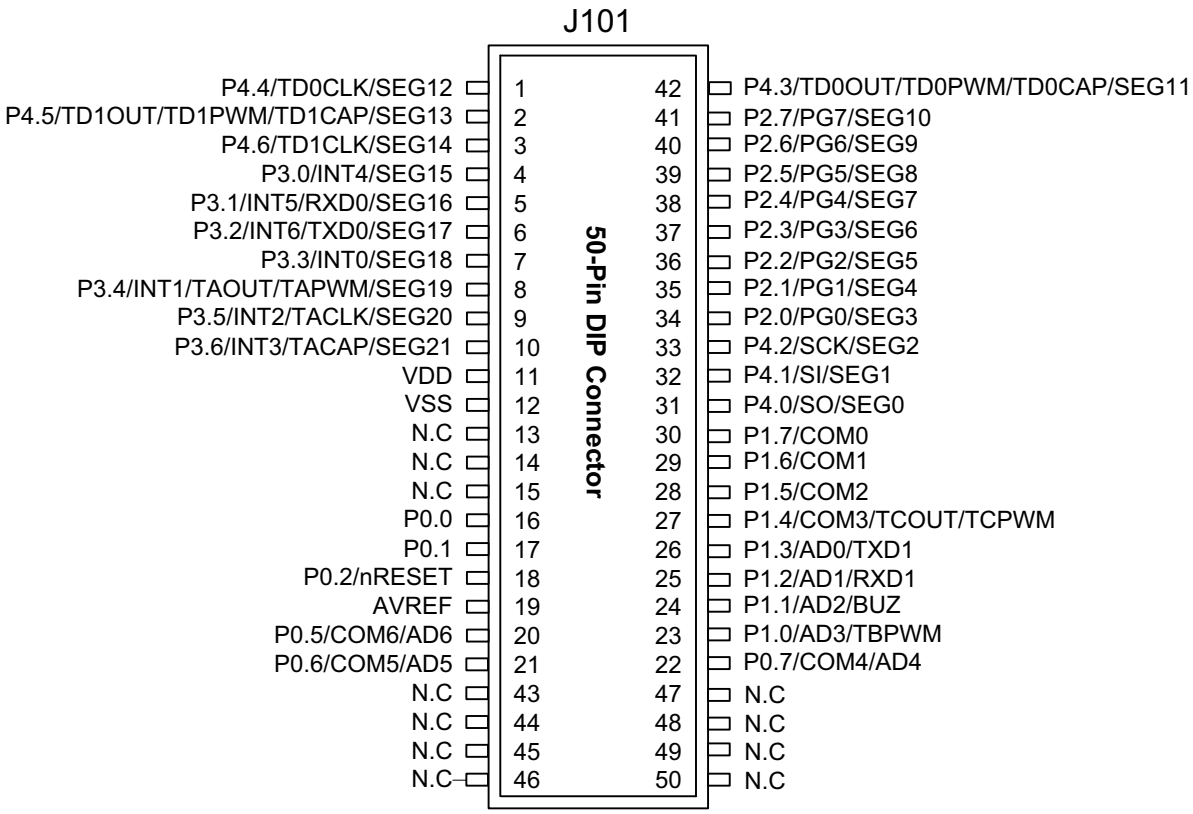


Figure 142. TB8S45 50-Pin J101 Connector

28.3. Programming Socket Adapter

When programming Flash memory with an emulator or OTP/MTP writer, a specific programming socket adapter is required for the S3F8S45 MCU.

28.4. Probe Adapter

Figure 143 shows the probe adapter for the S3F8S45 MCU's 44-pin QFP package.

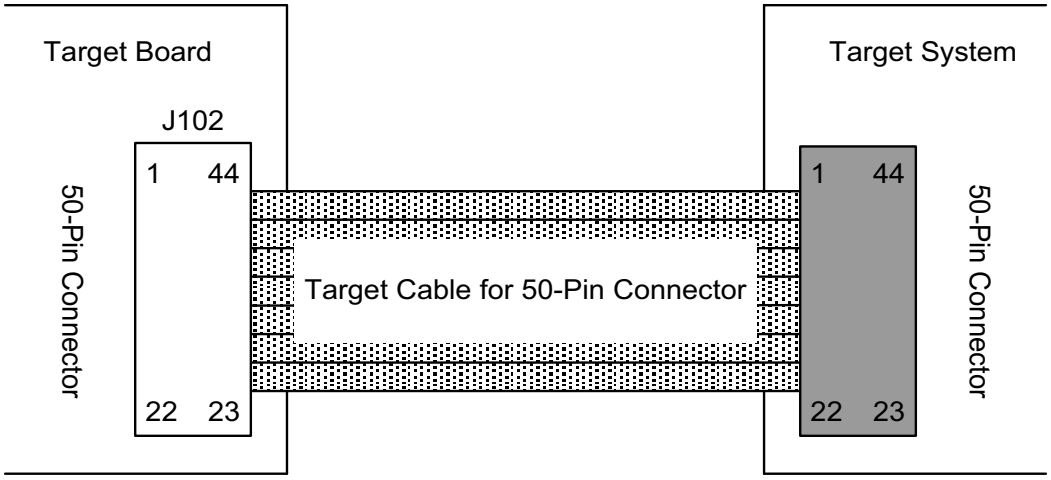


Figure 143. S3F8S45 Probe Adapter for the 44-Pin QFP Package

Figure 144 shows the probe adapter for the S3F8S45 MCU's 42-pin SDIP package.

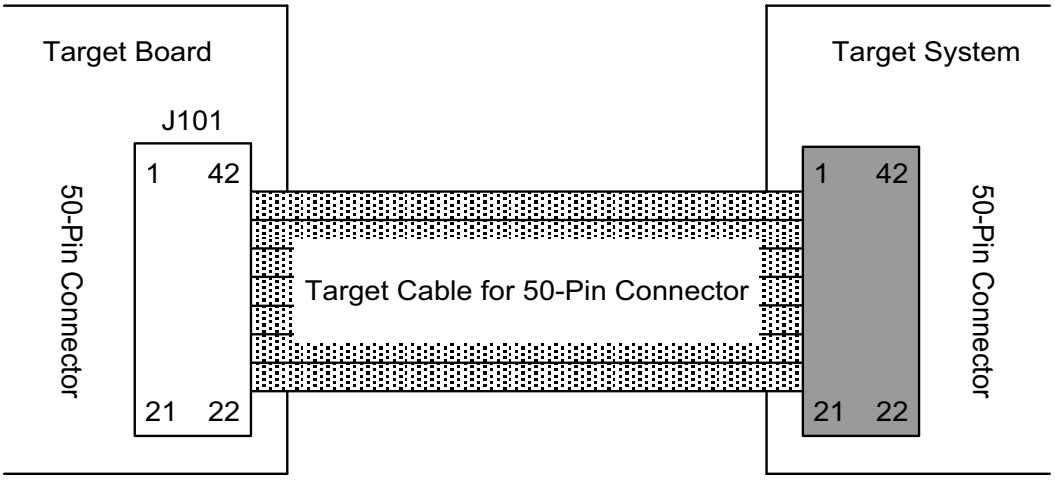


Figure 144. S3F8S45 Probe Adapter for the 42-Pin SDIP Package

Figure 145 shows the probe adapter for the S3F8S45 MCU's 32-pin SDIP package.

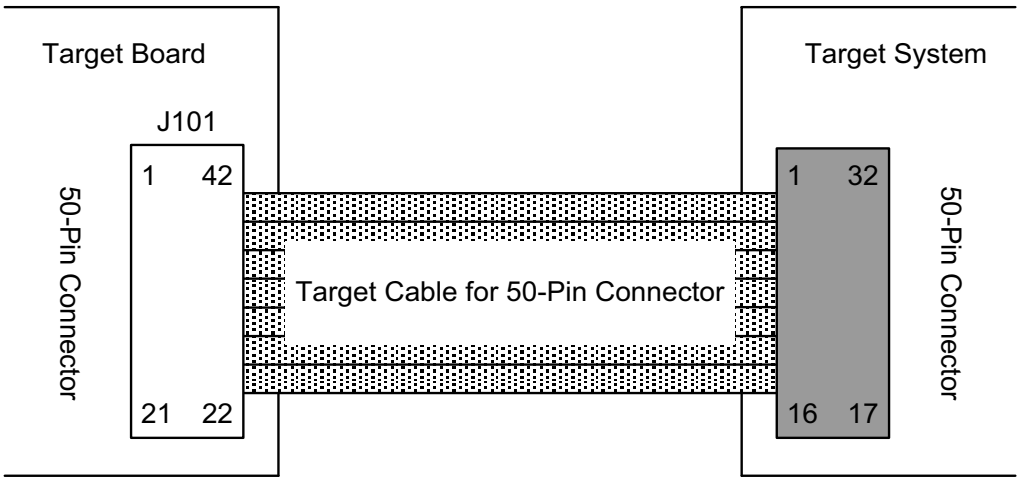


Figure 145. S3F8S45 Probe Adapter for the 32-Pin SDIP Package

28.5. Third Parties for Development Tools

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

In-circuit emulators:

- OPENice-i500/2000
- SK-1200 SmartKit

OTP/MTP Programmers:

- GW-Uni2
- AS-Pro2
- Elnec programmers

To obtain the S3 Family development tools that will satisfy your S3F8S45 development objectives, contact your local Zilog Sales Office, or visit Zilog's Third Party Tools page to review our list of third party tool suppliers.

Chapter 29. Ordering Information

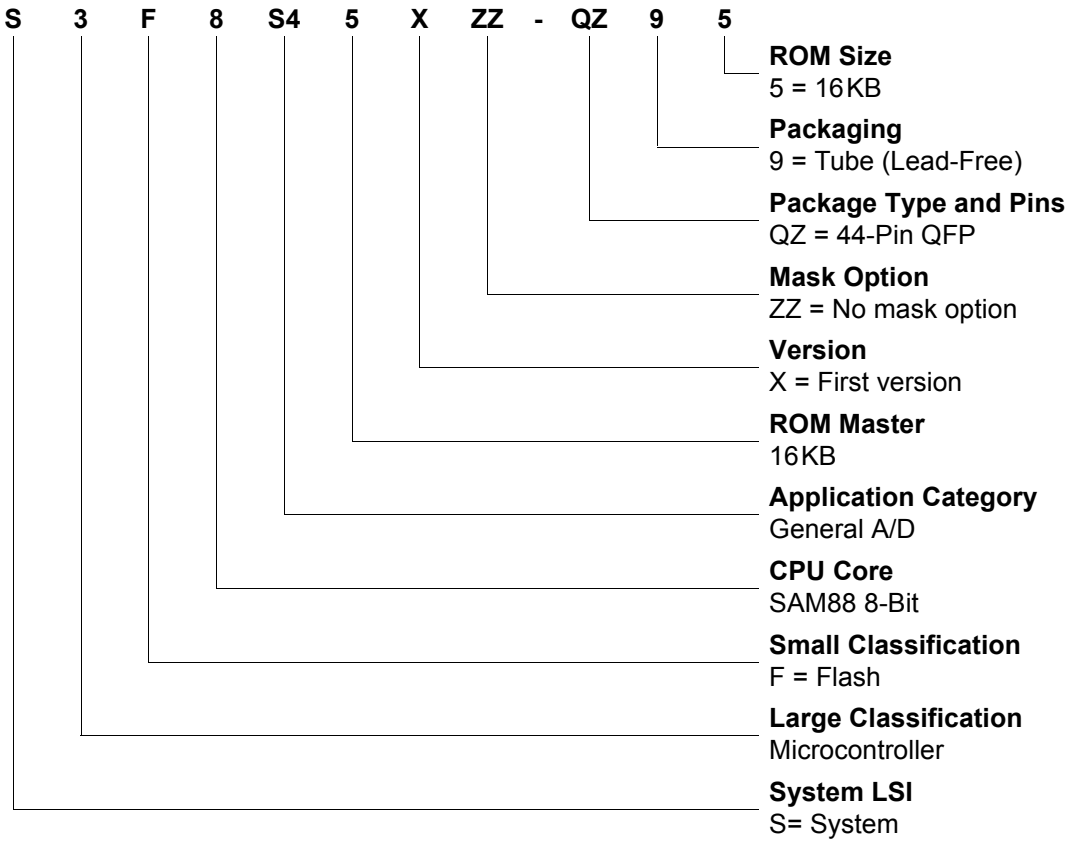
Table 120 identifies the basic features and package styles available for the S3F8S45 MCU.

Table 120. Ordering Information for the S3F8S45 MCU

Device	Flash Size	RAM Size	LCD	GPIO	ADC	SIO	Package
S3F8S45XZZ-QZ85	16KB	550 Bytes	19 seg. x 8 com.	38	10 bits x 9 channels	UART x2, SIO	44-Pin QFP
S3F8S45XZZ-AQ95	16KB	550 Bytes	19 seg. x 4 com.	36	10 bits x 7 channels	UART x2, SIO	32-Pin SDIP
S3F8S45XZZ-AO95	16KB	550 Bytes	12 seg. x 4 com.	26	10 bits x 4 channels	UART x2	32-Pin SDIP
S3F8S45XZZ-C0C5	16KB	550 Bytes	19 seg. x 8 com.	38	10 bits x 9 channels	UART x2, SIO	Pellet (Die)

29.1. Part Number Suffix Designations

Zilog part numbers consist of a number of components. For example, part number S3F8S45XZZ-QZ85 is an unmasked 8-bit MCU with 16KB of Flash memory in a 44-pin QFP package and built using lead-free solder.


S 3 F 8 S4 5 X ZZ - QZ 9 5
ROM Size
5 = 16KB
Packaging
9 = Tube (Lead-Free)
Package Type and Pins
QZ = 44-Pin QFP
Mask Option
ZZ = No mask option
Version
X = First version
ROM Master
16KB
Application Category
General A/D
CPU Core
SAM88 8-Bit
Small Classification
F = Flash
Large Classification
Microcontroller
System LSI
S= System

Chapter 30. Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb or consider participating in the Zilog Forum at http://zilog.com/forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.