SX48BD/SX52BD

High-Performance 8-Bit Microcontroller with EE/Flash Program Memory and In-System Programming Capability

1.0 PRODUCT OVERVIEW

1.1 Introduction

The Scenix SX48BD/SX52BD are members of the SX family of high-performance 8-bit microcontrollers fabricated in an advanced CMOS process technology. The advanced process, combined with a RISC-based architecture, allows high-speed computation, flexible I/O control, and efficient data manipulation. Throughput is enhanced by operating the device at frequencies up to 50 MHz and by optimizing the instruction set to include mostly single-cycle instructions.

On-chip functions include two 16-bit timers with 8-bit prescalers supporting different operating modes (PWM, simultaneous PWM/capture, and external event counter), a general-purpose 8-bit timer with prescaler, an analog comparator, a brown-out detector, a watchdog timer, a power-save mode with multi-source wakeup capability, an internal R/C oscillator, user-selectable clock modes, and high-current outputs.

The SX48BD and SX52BD are functionally the same, except for the package type and pinout. The SX48BD has four fewer pins and has only four rather than eight I/O pins for Port A.

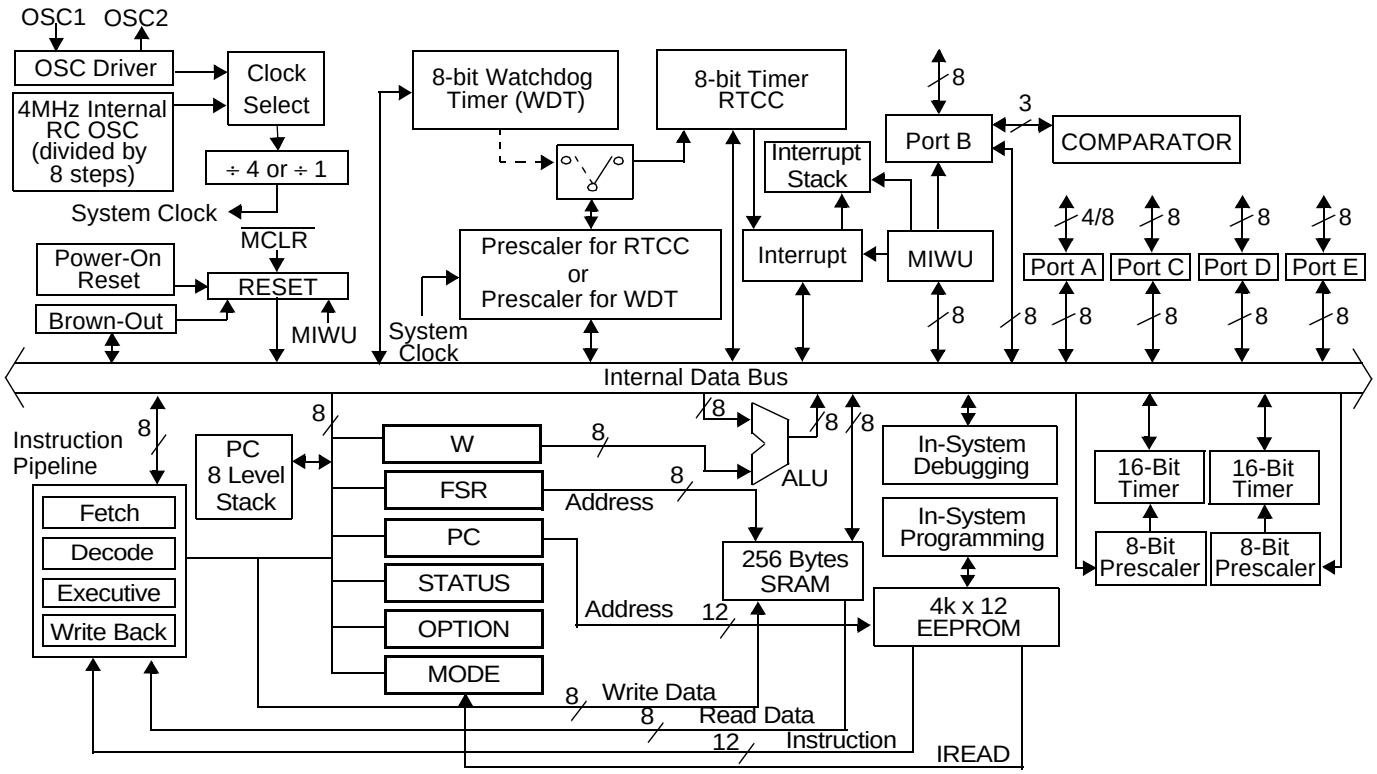


Figure 1-1. Block Diagram

1.2 Key Features

- 50 MIPS performance at 50 MHz oscillator frequency
- 4096 x 12 bits EE/Flash program memory rated for 10,000 rewrite cycles
- 256 x 8 bits SRAM
- Two 16-bit timers with 8-bit prescalers supporting:
  - Software Timer mode
  - PWM mode
  - Simultaneous PWM/Capture mode
  - External Event mode
- In-system programming capability through OSC pins
- Internal RC oscillator with configurable rate from 31.25 kHz to 4 MHz, ±8% accuracy
- User selectable clock modes:
  - Internal RC oscillator
  - External oscillator
  - Crystal/resonator options
  - External RC oscillator
- Analog comparator
- Brown-out detector
- Multi-Input Wakeup (MIWU) on eight pins
- Fast lookup capability through run-time readable code
- Complete development tool support

1.2.1 CPU Features

- Fully static design – DC to 50 MHz operation
- 20 ns instruction cycle time
- Compact instruction set
- Mostly single-cycle instructions
- 8-level deep hardware subroutine stack
- Fixed interrupt response time: 60 ns internal, 100 ns external at 50 MHz (Turbo Mode)
- Hardware context save/restore for interrupt

1.2.2 I/O Features

- Software-selectable I/O configuration
  - Each pin programmable as an input or output
  - TTL or CMOS level selection on inputs
  - Internal weak pull-up selection on inputs (~20 kΩ to V_{dd})
- Schmitt trigger inputs on ports B, C, D, and E
- All outputs capable of sinking/sourcing 30 mA
- Symmetrical drive on Port A outputs
- 48-pin TQFP and 48-pin QFP packages

1.3 Architecture

The SX devices use a modified Harvard architecture. This architecture uses two separate memories with separate address buses, one for the program and one for data, while allowing transfer of data from program memory to SRAM. This ability allows accessing data tables from program memory. The advantage of this architecture is that instruction fetch and memory transfers can be overlapped with a multi-stage pipeline, which means the next instruction can be fetched from program memory while the current instruction is being executed using data from the data memory.

The SX family implements a four-stage pipeline (fetch, decode, execute, and write back), which results in execution of one instruction per clock cycle. At the maximum operating frequency of 50 MHz, instructions are executed at the rate of one per 20-ns clock cycle.

1.4 Programming and Debugging Support

The SX devices are currently supported by the SX-Key™ offered by Parallax, Inc. This tool provides an integrated development environment including editor, macro assembler, debugger, and programmer.

1.5 Applications

Emerging applications and advances in existing ones require higher performance while maintaining low cost and fast time-to-market.

The device provides solutions for many familiar applications such as process controllers, electronic appliances/tools, security/monitoring systems, consumer automotive, sound generation, motor control, and personal communication devices. In addition, the device is suitable for applications that require DSP-like capabilities, such as closed-loop servo control (digital filters), digital answering machines, voice notation, interactive toys, and magnetic-stripe readers.

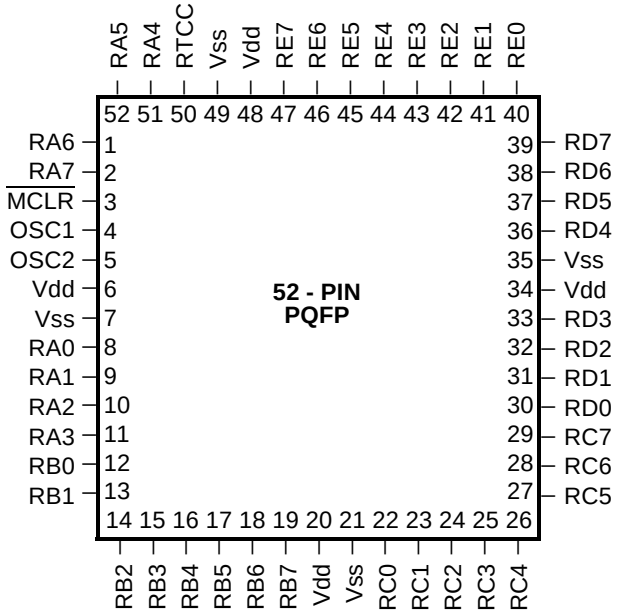
The enhanced throughput of the device allows efficient development of software modules called Virtual Peripherals to replace on-chip hardware peripherals. The concept of Virtual Peripherals provides benefits such as using a more simple device, reduced component count, fast time to market, increased flexibility in design, and ultimately overall system cost reduction.

Some examples of Virtual Peripheral modules are:

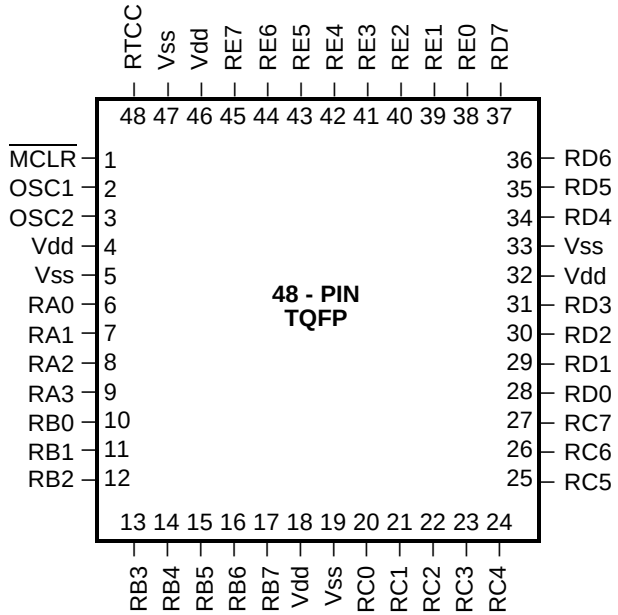
- Serial, Parallel, I^2C™, Microwire™ (µ-Wire), Dallas µ-Wire, SPI, DMX-512, X-10, IR transceivers
- Frequency generation and measurement
- Spectrum analysis
- Multi-tasking, interrupts, and networking
- Resonance loops
- DRAM drivers
- Music and voice synthesis
- PPM/PWM output
- Delta/Sigma ADC
- DTMF generation/detection
- PSK/FSK generation/detection
- Quadrature encoder/decoder
- Peripheral Interface Device (PID) and servo control
- Video controller

2.0 CONNECTION DIAGRAMS

2.1 Pin Assignments



Top View



Top View

2.2 Pin Descriptions

Name	Pin Type	Input Levels	Description
RA0	I/O	TTL/CMOS	Bidirectional I/O Pin; symmetrical source / sink capability
RA1	I/O	TTL/CMOS	Bidirectional I/O Pin; symmetrical source / sink capability
RA2	I/O	TTL/CMOS	Bidirectional I/O Pin; symmetrical source / sink capability
RA3	I/O	TTL/CMOS	Bidirectional I/O Pin; symmetrical source / sink capability
RA4	I/O	TTL/CMOS	Bidirectional I/O Pin; symmetrical source / sink capability (52-pin pkg. only)
RA5	I/O	TTL/CMOS	Bidirectional I/O Pin; symmetrical source / sink capability (52-pin pkg. only)
RA6	I/O	TTL/CMOS	Bidirectional I/O Pin; symmetrical source / sink capability (52-pin pkg. only)
RA7	I/O	TTL/CMOS	Bidirectional I/O Pin; symmetrical source / sink capability (52-pin pkg. only)
RB0	I/O	TTL/CMOS/ST	Bidirectional I/O Pin; comparator output; MIWU input
RB1	I/O	TTL/CMOS/ST	Bidirectional I/O Pin; comparator negative input; MIWU input
RB2	I/O	TTL/CMOS/ST	Bidirectional I/O Pin; comparator positive input; MIWU input
RB3	I/O	TTL/CMOS/ST	Bidirectional I/O Pin; MIWU input
RB4	I/O	TTL/CMOS/ST	Bidirectional I/O Pin; MIWU input, Timer T1 Capture Input 1
RB5	I/O	TTL/CMOS/ST	Bidirectional I/O Pin; MIWU input, Timer T1 Capture Input 2
RB6	I/O	TTL/CMOS/ST	Bidirectional I/O Pin; MIWU input, Timer T1 PWM/Compare Output
RB7	I/O	TTL/CMOS/ST	Bidirectional I/O Pin; MIWU input, Timer T1 External Event Clock Source
RC0	I/O	TTL/CMOS/ST	Bidirectional I/O pin, Timer T2 Capture Input 1
RC1	I/O	TTL/CMOS/ST	Bidirectional I/O pin, Timer T2 Capture Input 2
RC2	I/O	TTL/CMOS/ST	Bidirectional I/O pin, Timer T2 PWM/Compare Output
RC3	I/O	TTL/CMOS/ST	Bidirectional I/O pin, Timer T2 External Event Clock Source
RC4	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RC5	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RC6	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RC7	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RD0	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RD1	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RD2	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RD3	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RD4	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RD5	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RD6	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RD7	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RE0	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RE1	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RE2	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RE3	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RE4	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RE5	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RE6	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RE7	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RTCC	I	ST	Input to Real-Time Clock/Counter
$\overline{\text{MCLR}}$	I	ST	Master Clear reset input – active low
OSC1/In/Vpp	I	ST	Crystal oscillator input – external clock source input
OSC2/Out	O	CMOS	Crystal oscillator output – in R/C mode, internally pulled to V_{dd} through weak pull-up
V_{dd}	P	–	Positive supply pins (a total of four positive supply pins, one on each side of the device)
Vss	P	–	Ground pins (a total of four ground pins, one on each side of the device)
Note:I = input, O = output, I/O = Input/Output, P = Power, TTL = TTL input, CMOS = CMOS input, ST = Schmitt Trigger input, MIWU = Multi-Input Wakeup input			

2.3 Part Numbering g

Table 2-1. Ordering Information

Device	Pins	I/O	EE/Flash (Words)	RAM (Bytes)
SX52BD/PQ	52	40	4K	256
SX48BD/TQ	48	40	4K	256

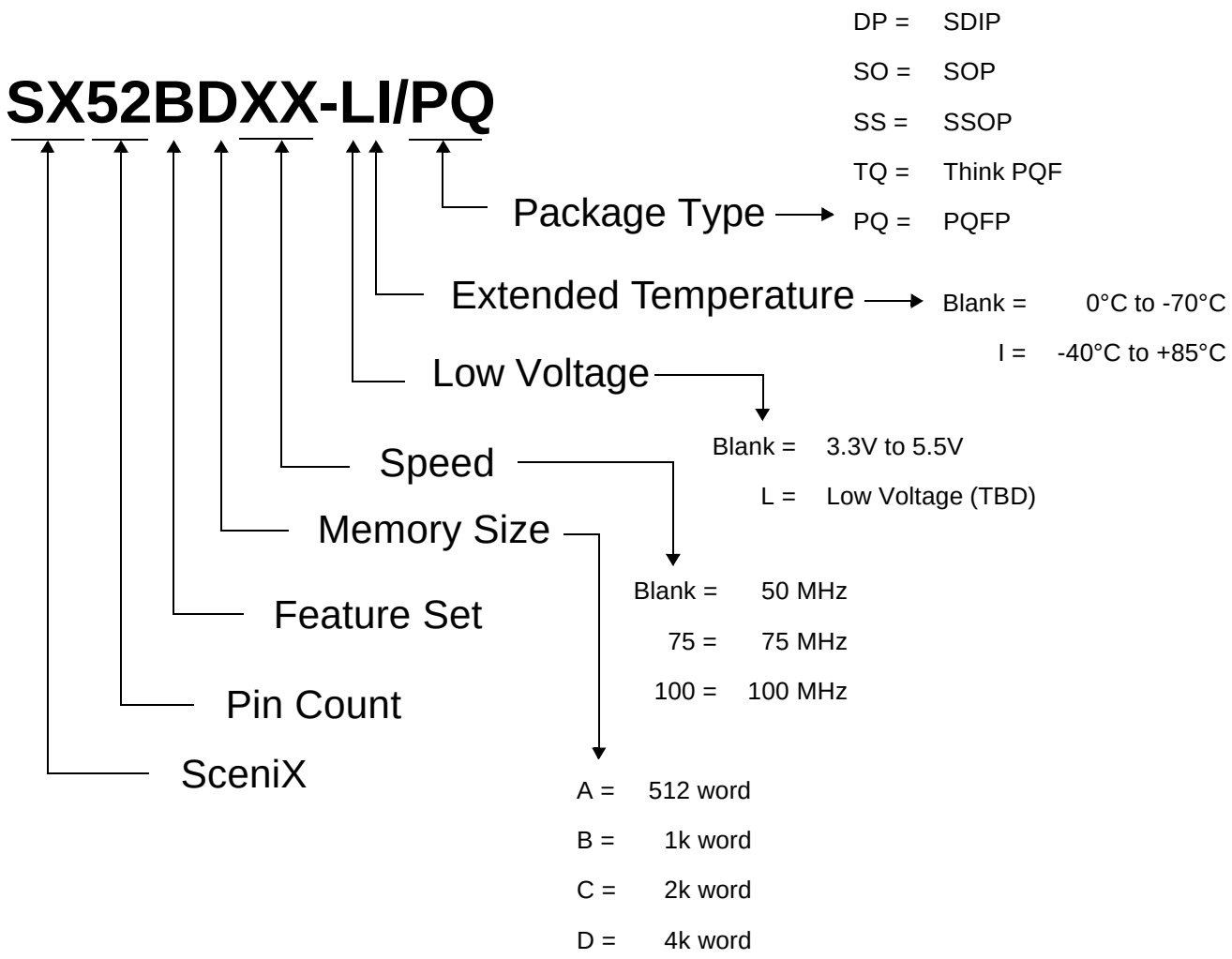


Figure 2-1. Part Number Reference Guide

3.0 PORT DESCRIPTIONS

The device contains five 8-bit I/O ports (Port A through Port E). Port A provides symmetrical drive capability. In the 48-pin version of the device, Port A has only four pins rather than eight. The unavailable pins are not terminated i.e., they are floating. A read operation for these unterminated pins will return unpredictable values. The user must ensure that the software takes this into account by either masking or restricting accesses to bit operations. The user can program the unavailable pins with internal weak pullups.

Each port has three associated 8-bit registers (Direction, Data, TTL/CMOS Select, and Pull-Up Enable) to configure each port pin as Hi-Z input or output, to select TTL or CMOS voltage levels, and to enable/disable the weak pull-up resistor. The least significant bit of the registers corresponds to the least significant port pin. To access these configuration registers, an appropriate value must be written into the MODE register.

Upon power-up, all bits in these registers are initialized to "1".

The associated registers allow for each port bit to be individually configured under software control as shown below:

Table 3-1. Port Configuration

Data Direction Registers: RA, RB, RC		TTL/CMOS Select Registers: LVL_A, LVL_B, LVL_C		Pullup Enable Registers: PLP_A, PLP_B, PLP_C	
0	1	0	1	0	1
Output	Hi-Z Input	CMOS	TTL	Enable	Disable

Ports B, C, D, and E have additional associated registers (Schmitt-Trigger Enable Registers ST_B and ST_C) to enable or disable the Schmitt Trigger function on each individual port pin as indicated in table below.

Table 3-2. Schmitt Trigger Select

Schmitt Trigger Enable Registers: ST_B, ST_C	
0	1
Enable	Disable

Port B also supports the on-chip differential comparator. Ports RB1 and RB2 are the comparator negative and positive inputs, respectively, while Port RB0 is the comparator output pin. Port B also supports the Multi-Input Wakeup feature on all eight pins.

Port C and Port D also support the multi-function timers T1 and T2. RB4 and RB5 and the T1 capture inputs, RB6 is the T1 PWM output, and RB7 is the T1 external event counter input. Similarly, RC0 and RC1 are the T2 capture inputs, RC2 is the T2 PWM output, and RC3 is the T2 external event counter input.

Figure 3-1 shows the internal hardware structure and configuration registers for each pin of Port A. Figure 3-2 shows the same for each pin of Port B, C, D, or E.

3.1 Reading and Writing the Ports

The five ports are memory-mapped into the data memory address space. To the CPU, the five ports are available as the RA, RB, RC, RD, and RE file registers at data memory addresses 05h through 09h, respectively. Writing to a port data register sets the voltage levels of the corresponding port pins that have been configured to operate as outputs. Reading from a data register reads either the voltage levels of the corresponding port pins or the data contained in the port data register depending on the status PORTRD bit contained in the T2CNTB register.

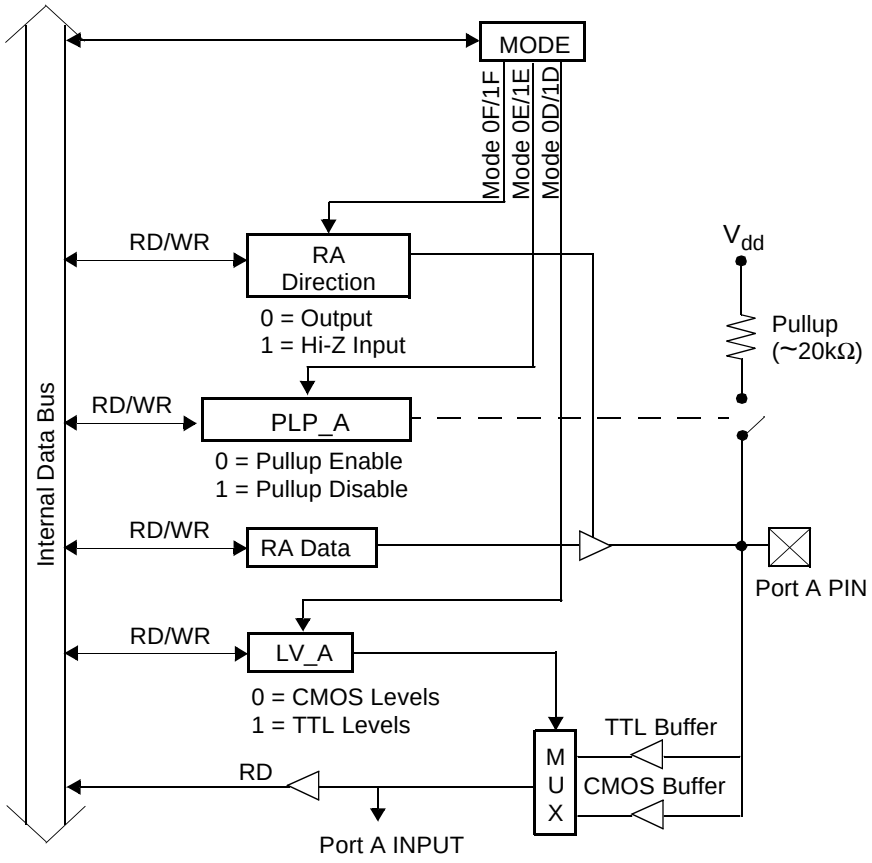


Figure 3-1. Port A Configuration

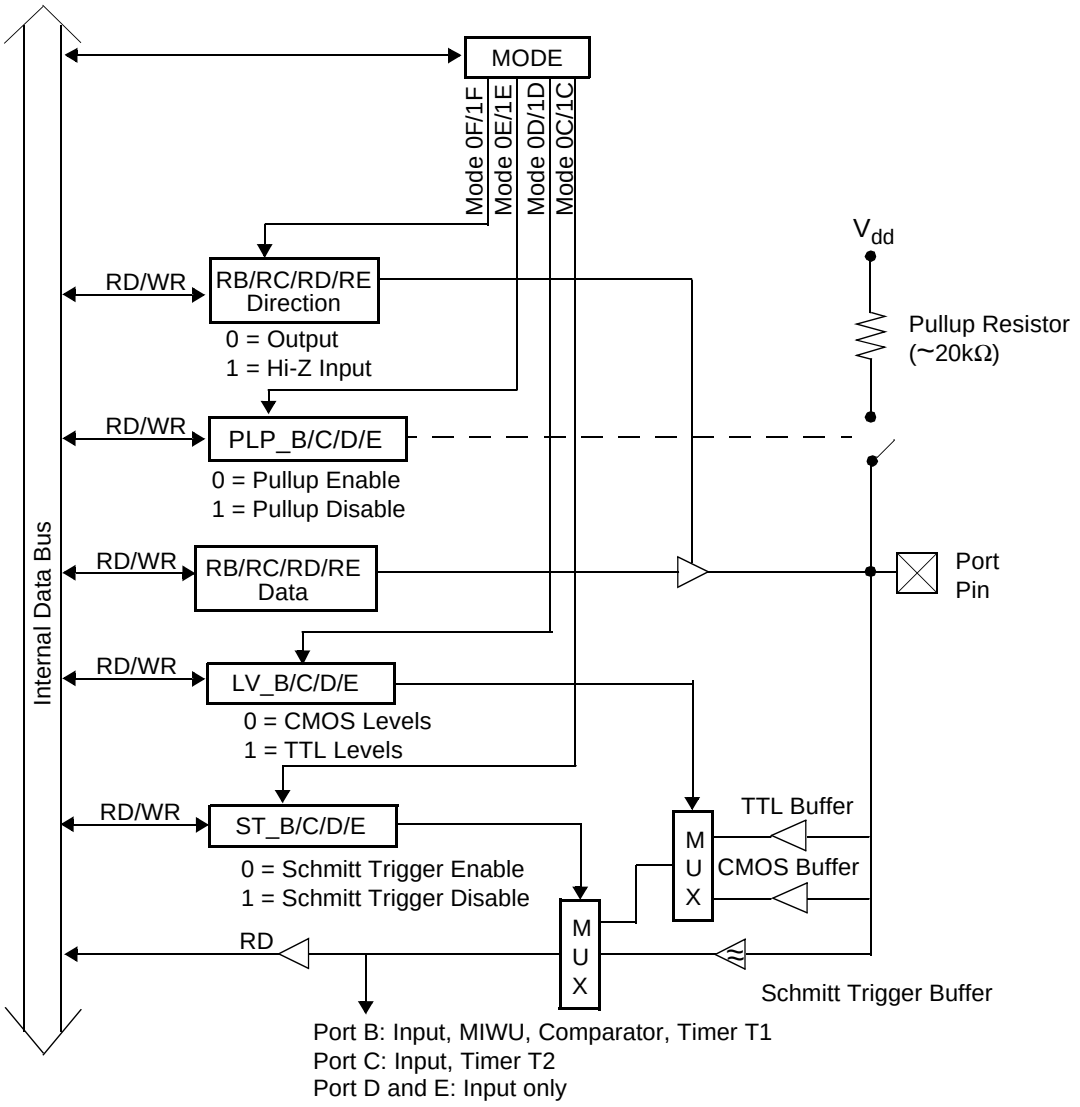


Figure 3-2. Port B, Port C Configuration

For example, suppose all four Port A pins are configured as outputs. To make RA0 and RA1 high and the remaining Port A pins low, you could use the following code:

```
mov   W,#$03    ;load W with the value 03h
                ;(bits 0 and 1 high)
mov   $05,W     ;write 03h to Port A data
                ;register
```

The second “mov” instruction in this example writes the Port A data register (RA), which controls the output levels of the Port A pins, RA0 through RA7. Note that Port A has only four I/O pins in the 48-pin version of the device, in which case only the four least significant bits of this register are used.

When a write is performed to a port bit position that has been configured as an input, a write to the port data register is still performed, but it has no immediate effect on the pin. If later that pin is configured to operate as an output, it will reflect the value that has been written to the data register.

In the default device configuration, when a read is performed from a port bit position, the operation is actually reading the voltage level on the pin itself, not necessarily the bit value stored in the port data register. This is true whether the pin is configured to operate as an input or an output. Therefore, with the pin configured to operate as an input, the data register contents have no effect on the value that you read. With the pin configured to operate as an output, what is read generally matches what has been written to the register. PORTRD of the T2CNT2 register determines how the device reads data from its I/O ports (Port A through Port E). Set this bit to 1 to have the device read data from the port I/O pins directly. Clear this bit to 0 to have the device read data from the port data registers. Under normal conditions, it should not matter which method you use to read the port data. However, if a port pin is configured as an output and an external circuit forces the pin to the opposite value, the value read from the port will depend on the reading mode used. Note that this control bit is not related to multi-function timers T1 and T2.

When two successive read-modify-write instructions are used on the same I/O port with a very high clock rate, the “write” part of one instruction might not occur soon enough before the “read” part of the very next instruction, resulting in getting “old” data for the second instruction. To ensure predictable results, avoid using two successive read-modify-write instructions that access the same port data register if the clock rate is high.

3.2 Port Configuration

Each port pin offers the following configuration options:

- data direction
- input voltage levels (TTL or CMOS)
- pullup type (pullup resistor or open collector)
- Schmitt trigger input (except for Port A)

Port B offers the additional option to use the port pins for the Multi-Input Wakeup/Interrupt function, the analog comparator function, or Timer T1 I/O. Port C offers the additional option to use the port pins for Timer T2 I/O.

Port configuration is preformed by writing to a set of control registers associated with the port. A special-purpose instruction is used to write these control registers:

- mov !RA,W (move W to/from Port A control register)
- mov !RB,W (move W to/from Port B control register)
- mov !RC,W (move W to/from Port C control register)
- mov !RD,W (move W to/from Port D control register)
- mov !RE,W (move W to/from Port E control register)

Each one of these instructions reads or writes a port control register for Port A, B, C, D, or E. There are multiple control registers for each port. To specify which one you want to access, you use another register called the MODE register.

3.2.1 MODE Register

The MODE register controls access to the port configuration registers and Timer T1/T2 control registers. Because the MODE register is not memory-mapped, it is accessed by the following special-purpose instructions:

- mov M, #lit (move literal to MODE register)
- mov M,W (move W to MODE register)
- mov W,M (move MODE register to W)

The value contained in the MODE register determines which port control register is accessed by the “mov !rx,W” instruction as indicated in Table 3-3. (The table also shows the timer control registers accessed according to the MODE register setting.) MODE register values not defined in the table are reserved for future expansion and should not be used. Upon power-up, the MODE register is initialized to 1Fh, which enables write access to the port direction control registers.

When bit 4 of the MODE register is 0 (the top half of Table 3-3), a “mov !rx,W” instruction moves the contents of the applicable control register into W. When bit 4 of the MODE register is 1 (the bottom half of Table 3-3), a “mov !rx,W” instruction moves the contents of W into the applicable control register. However, there are some exceptions to this. For the CMP_B and WKPND_B registers, the CPU does an exchange of data between W and the control register, regardless of the state of bit 4 in the MODE register. For the WKED_B and WKEN_B registers, the CPU moves the data from W to the control register, regardless of the state of bit 4 in the MODE register.

After a value is written to the MODE register, that setting remains in effect until it is changed by writing to the MODE register again. For example, you can write the value 1Eh to the MODE register just once, and then write to each of the three pullup configuration registers using the three “mov !rx,W” instructions.

Table 3-3. Mode Register Settings

MODE Reg.	mov !RA,W	mov !RB,W	mov !RC,W	mov !RD,W	mov !RE,W
00h		Read T1CPL	Read T2CPL		
01h		Read T1CPH	Read T2CPH		
02h		Read T1R2CML	Read T2R2CML		
03h		Read T1R2CMH	Read T2R2CMH		
04h		Read T1R1CML	Read T2R1CML		
05h		Read T1R1CMH	Read T2R1CMH		
06h		Read T1CNTB	Read T2CNTB		
07h		Read T1CNTA	Read T2CNTA		
08h		Exchange CMP_B			
09h		Exchange WKPND_B			
0Ah		Write WKED_B			
0Bh		Write WKEN_B			
0Ch		Read ST_B	Read ST_C	Read ST_D	Read ST_E
0Dh	Read LVL_A	Read LVL_B	Read LVL_C	Read LVL_D	Read LVL_E
0Eh	Read PLP_A	Read PLP_B	Read PLP_C	Read PLP_D	Read PLP_E
0Fh	Read RA Direction	Read RB Direction	Read RC Direction	Read RD Direction	Read RE Direction
10h		Clear Timer T1	Clear Timer T2		
11h					
12h		Write T1R2CML	Write T2R2CML		
13h		Write T1R2CMH	Write T2R2CMH		
14h		Write T1R1CML	Write T2R1CML		
15h		Write T1R1CMH	Write T2R1CMH		
16h		Write T1CNTB	Write T2CNTB		
17h		Write T1CNTA	Write T2CNTA		
18h		Exchange CMP_B			
19h		Exchange WKPND_B			
1Ah		Write WKED_B			
1Bh		Write WKEN_B			
1Ch		Write ST_B	Write ST_C	Write ST_D	Write ST_E
1Dh	Write VL_A	Write LVL_B	Write LVL_C	Write LVL_D	Write LVL_E
1Eh	Write PLP_A	Write PLP_B	Write PLP_C	Write PLP_D	Write PLP_E
1Fh	Write RA Direction	Write RB Direction	Write RC Direction	Write RD Direction	Write RE Direction

The following code example shows how to program the pullup control registers.

```
mov  W,#$1E   ;MODE=1Eh to write port pullup
mov  M, W     ;registers

mov  W,#$03   ;W = 0000 0011
mov  !RA,W    ;disable pullups for A0 and A1

mov  W,#$FF   ;W = 1111 1111
mov  !RB,W    ;disable all pullups for B0-B7

mov  W,#$00   ;W = 0000 0000
mov  !RC,W    ;enable all pullups for C0-C7
```

First the MODE register is loaded with 1Eh to select write access to the pullup control registers (PLP_A, PLP_B, and so on). Then the MOV !rx,W instructions are used to specify which port pins are to be connected to the internal pullup resistors. Setting a bit to 1 disconnects the corresponding pullup resistor, and clearing a bit to 0 connects the corresponding pullup resistor.

3.2.2 Port Configuration Registers

The port configuration registers that you control with the MOV !rx,W instruction operate as described below.

RA through RE Data Direction Registers (MODE=1Fh)

Each register bit sets the data direction for one port pin. Set the bit to 1 to make the pin operate as a high-impedance input. Clear the bit to 0 to make the pin operate as an output.

PLP_A through PLP_E: Pullup Enable Registers (MODE=1Eh)

Each register bit determines whether an internal pullup resistor is connected to the pin. Set the bit to 1 to disconnect the pullup resistor or clear the bit to 0 to connect the pullup resistor.

LVL_A through LVL_E: Input Level Registers (MODE=1Dh)

Each register bit determines the voltage levels sensed on the input port, either TTL or CMOS, when the Schmitt trigger option is disabled. Program each bit according to the type of device that is driving the port input pin. Set the bit to 1 for TTL or clear the bit to 0 for CMOS.

ST_B through ST_E: Schmitt Trigger Enable Registers (MODE=1Ch)

Each register bit determines whether the port input pin operates with a Schmitt trigger. Set the bit to 1 to disable Schmitt trigger operation and sense either TTL or CMOS voltage levels; or clear the bit to 0 to enable Schmitt trigger operation.

WKEN_B: Wakeup Enable Register (MODE=1Bh)

Each register bit enables or disables the Multi-Input Wakeup/Interrupt (MIWU) function for the corresponding Port B input pin. Clear the bit to 0 to enable MIWU operation or set the bit to 1 to disable MIWU operation. For more information on using the Multi-Input Wakeup/Interrupt function, see Section 7.1.

WKED_B: Wakeup Edge Register (MODE=1Ah)

Each register bit selects the edge sensitivity of the Port B input pin for MIWU operation. Clear the bit to 0 to sense rising (low-to-high) edges. Set the bit to 1 to sense falling (high-to-low) edges.

WKPND_B: Wakeup Pending Flag Register (MODE=19h)

When you access the WKPND_B register using MOV !RB,W, the CPU does an exchange between the contents of W and WKPND_B. Each bit read from the WKPND_B register indicates the status of the corresponding MIWU pin. A bit set to 1 indicates that a valid edge has occurred on the corresponding MIWU pin, triggering a wakeup or interrupt. A bit set to 0 indicates that no valid edge has occurred on the MIWU pin.

CMP_B: Comparator Register (MODE=08h)

When you access the CMP_B register using MOV !RB,W, the CPU does an exchange between the contents of W and CMP_B. This feature lets you read the CMP_B register contents while writing a new value to the register. Clear bit 7 to enable operation of the comparator. Clear bit 6 to place the comparator result on the RB0 pin. Bit 0 is a result flag that is set to 1 when the voltage on RB2 is greater than RB1, or cleared to 0 otherwise. (For more information using the comparator, see Section 11.0.)

3.2.3 Port Configuration Upon Power-Up

Upon power-up, all the port control registers are initialized to FFh. Thus, each port pin is configured to operate as a high-impedance input that senses TTL voltage levels, with no internal pullup resistor connected. The MODE register is initialized to 1Fh, which allows immediate write access to the data direction registers using the “MOV !rx,W” instruction.

4.0 SPECIAL-FUNCTION REGISTERS

The CPU uses a set of special-function registers to control operation of the device.

The CPU registers include an 8-bit working register (W), which serves as a pseudo accumulator. It holds the second operand of an instruction, receives the literal in immediate type instructions, and also can be program-selected as the destination register.

A set of 31 file registers serves as the primary accumulator. One of these registers holds the first operand of an instruction and another can be program-selected as the destination register. The first 10 file registers include the Real-Time Clock/Counter register (RTCC), the lower eight bits of the 12-bit Program Counter (PC), the 8-bit STATUS register, five port control registers for Ports A through E, and the 8-bit File Select Register (FSR).

The five low-order bits of the FSR register select one of the 31 file registers in the indirect addressing mode. Calling for the file register located at address 00h (INDF) in any of the file-oriented instructions selects indirect addressing, which uses the FSR register. It should be noted that the file register at address 00h is not a physically implemented register. The CPU also contains an 8-level, 12-bit hardware push/pop stack for subroutine linkage.

Table 4-1. Special-Function Registers

Addr	Name	Function
00h	INDF	Used for indirect addressing
01h	RTCC	Real Time Clock/Counter
02h	PC	Program Counter (low byte)
03h	STATUS	Holds Status bits of ALU
04h	FSR	File Select Register
05h	RA	Port RA data register
06h	RB	Port RB data register
07h	RC	Port RC data register
08h	RD	Port RD data register
09h	RE	Port RE data register

4.1 PC Register (02h)

The PC register holds the lower eight bits of the program counter. It is accessible at run time to perform branch operations.

4.2 STATUS Register (03h)

The STATUS register holds the arithmetic status of the ALU, the page select bits, and the reset state. The STATUS register is accessible during run time, except that bits PD and TO are read-only. It is recommended that only SETB and CLRB instructions be used on this register. Care should be exercised when writing to the STATUS register as the ALU status bits are updated upon completion of the write operation, possibly leaving the STATUS register with a result that is different than intended.

PA2	PA1	PA0	TO	PD	Z	DC	C
Bit 7							**Bit 0**

Bit 7-5: Page select bits PA2:PA0

000 = Page 0 (000h – 01FFh)

001 = Page 1 (200h – 03FFh)

. . .

111 = Page 7 (E00h – FFFh)

Bit 4: Time Out bit, TO

1 = Set to 1 after power up and upon execution of CLRWDT or SLEEP instructions

0 = A watchdog time-out occurred

Bit 3: Power Down bit, PD

1= Set to a 1 after power up and upon execution of the CLRWDT instruction

0 = Cleared to a ‘0’ upon execution of SLEEP instruction

Bit 2: Zero bit, Z

1 = Result of math operation is zero

0 = Result of math operation is non-zero

Bit 1: Digit Carry bit, DC

After Addition:

1 = A carry from bit 3 occurred

0 = No carry from bit 3 occurred

After Subtraction:

1 = No borrow from bit 3 occurred

0 = A borrow from bit 3 occurred

Bit 0: Carry bit, C

1 = A carry or borrow from the MSB of the result occurred. For rotate (RR and RL) instructions, this bit is loaded with the low or high order bit, respectively.

0 = No carry from the MSB as a result of Add operation, or borrow as a result of Subtract operation.

4.3 OPTION Register

RTW	RTE_IE	RTS	RTE_ES	PSA	PS2	PS1	PS0
Bit 7							**Bit 0**

When the OPTIONX bit in the FUSE word is cleared, bits 7 and 6 of the OPTION register are implemented.

When the OPTIONX bit is set, bits 7 and 6 of the OPTION register read as 1s.

RTW RTCC/W register selection:
0 = Register 01h addresses W
1 = Register 01h addresses RTCC

RTE_IE RTCC edge interrupt enable:
0 = RTCC roll-over interrupt is enabled
1 = RTCC roll-over interrupt is disabled

RTS RTCC increment select:
0 = RTCC increments on internal instruction cycle
1 = RTCC increments upon transition on RTCC pin

RTE_ES RTCC edge select:
0 = RTCC increments on low-to-high transitions
1 = RTCC increments on high-to-low transitions

PSA Prescaler Assignment:
0 = Prescaler is assigned to RTCC, with divide rate determined by PS0-PS2 bits
1 = Prescaler is assigned to WDT, and divide rate on RTCC is 1:1

PS2-PS0 Prescaler divider (see Table 4-2)

Table 4-2. Prescaler Divider Ratios

PS2, PS1, PS0	RTCC Divide Rate	Watchdog Timer Divide Rate
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

Upon reset, all bits in the OPTION register are set to 1.

4.4 DEVICE CONFIGURATION REGISTERS

The SX device has three registers (FUSE, FUSEX, DEVICE) that control functions such as operating the device in Turbo mode, extended (8-level deep) stack operation, and speed selection for the internal RC oscillator. These registers are not programmable “on the fly” during normal device operation. Instead, the FUSE and FUSEX registers can only be accessed when the SX device is being programmed. The DEVICE register is a read-only, hard-wired register, programmed during the manufacturing process.

4.5 FUSE Word (Read/Program at 1FFFh in main memory map)

$\overline{\text{TURBO}}$	$\overline{\text{SYNC}}$	$\overline{\text{OPTIONX}}$	$\overline{\text{STACKX}}$	$\overline{\text{IRC}}$	DIV2	DIV1	DIV0	$\overline{\text{CP}}$	WDTE	FOSC1	FOSC0
11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

$\overline{\text{TURBO}}$ Turbo mode enable:
- 0 = turbo mode (instruction clock = osc/1)
- 1 = compatible mode (instruction clock = osc/4)

$\overline{\text{SYNC}}$ Synchronous input enable (for turbo mode):
- 0 = enabled
- 1 = disabled

$\overline{\text{OPTIONX}}$ OPTION register extension enable:
- 0 = OPTION register increased from six to eight bits for RTW and RTW_IE
- 1 = OPTION register is six bits (two most significant bits forced to 1)

$\overline{\text{STACKX}}$ Stack extension enable:
- 0 = 8 levels (stack extension enabled)
- 1 = 2 levels (stack extension disabled)

$\overline{\text{IRC}}$ Internal RC oscillator enable:
- 0 = enabled - OSC1 weakly pulled low, OSC2 weakly pulled high
- 1 = disabled - OSC1 and OSC2 behave according to FOSC1: FOSC0

DIV2: DIV0 Internal RC oscillator divider:
- 000b = 4 MHz
- 001b = 2 MHz
- 010b = 1 MHz
- 011b = 500 KHz
- 100b = 250 KHz
- 101b = 125 KHz
- 110b = 62.5 KHz
- 111b = 31.25 KHz

$\overline{\text{CP}}$ Code protect enable:
- 0 = enabled (FUSE, code, and ID memories read back as garbled data)
- 1 = disabled (FUSE, code, and ID memories can be read normally)

WDTE Watchdog timer enable:
- 0 = disabled
- 1 = enabled

FOSC1: FOSC0 External oscillator configuration (valid when IRC = 1):
- 00b = LP – low power crystal
- 10b = HS – high speed crystal
- 01b = XT – normal crystal
- 11b = RC network - OSC2 is pulled high by a weak pullup(no CLKOUT output)

4.6 FUSEX Word (Read/Program via Programming Command)

SLEEPCLK	WDRT2 : WDRT0	$\overline{CF}$	IRCTRIM2 : IRCTRIM0	Unused	BOR0:BOR0
11	**10 9 8**	**7**	**6 5 4**	**3 2**	**1 0**

SLEEPCLLK — Sleep Clock Disable. Clear this bit to 0 to enable operation of the clock during sleep mode (to allow fast start-up). Set this bit to 1 to disable clock operation during sleep mode (to reduce power consumption).

WDRT2: WDRT0 — Delay Reset Timer (DRT) timeout period. This 3-bit field can be used to specify the DRT timeout period that results in an automatic wake-up from the sleep mode:

100 = 0 msec (no delay)
101 = 0.06 msec
110 = 7.68 msec
111 = 18.4 msec (default)
000 = 60 msec
001 = 480 msec
010 = 960 msec
011 = 1920 msec

For fast start-up from the sleep mode, clear the SLEEPCLK bit and set the WDRT2:WDRT0 field to 100. This will keep the clock operating during the sleep mode and allow a zero start-up delay.

$\overline{CF}$ — active low – makes the carry flag an input to ADD and SUB instructions.

IRCTRIM2: IRCTRIM0 — Internal RC Oscillator Trim. This 3-bit field adjusts the operation of the internal RC oscillator to make it operate within the target frequency range of 4.0 MHz plus or minus 8%. Parts are shipped from the factory untrimmed. The device relies on the programming tool to provide trimming.

BOR1: BOR0 — Brown-Out Reset; factory preset values. Bits should not be changed unless brown-out feature is to be disabled. Set bits to “11b” to disable.

4.7 DEVICE Word (Hard-Wired Read-Only)- Part ID Code

0	0	0	0	0.	0	0	0	0	0	0	1
11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

5.0 MEMORY ORGANIZATION

5.1 Program Memory

The program memory is organized as 4K, 12-bit wide words. The program memory words are addressed sequentially by a binary program counter. The program counter starts at zero. If there is no branch operation, it will increment to the maximum value possible for the device and roll over and begin again.

Internally, the program memory has a semi-transparent page structure. A page is composed of 512 contiguous program memory words. The lower nine bits of the program counter are zeros at the first address of a page and ones at the last address of a page. This page structure has no effect on the program counter. The program counter will freely increment through the page boundaries.

5.1.1 Program Counter

The program counter contains the 12-bit address of the instruction to be executed. The lower eight bits of the program counter are contained in the PC register (02h). The three upper bits are specified by the STATUS register (PA0, PA1, PA2). Changing the STATUS bits is necessary to cause jumps and subroutine calls *across* program memory page boundaries. Prior to the execution of a branch operation, the user program must initialize the upper bits of the STATUS register to cause a branch to the desired page. An alternative method is to use the PAGE instruction, which automatically causes branch to the desired page, based on the value specified in the operand field. Upon reset, the program counter is initialized with 0FFFh.

5.1.2 Subroutine Stack

The subroutine stack consists of eight 12-bit save registers. A physical transfer of register contents from the program counter to the stack or vice versa, and within the stack, occurs on all operations affecting the stack, primarily calls and returns. The stack is physically and logically separate from data RAM. The program cannot read or write the stack.

5.2 Data Memory

The data memory is a RAM-based register set consisting of 262 general-purpose registers and nine special-purpose registers. All of these registers are eight bits wide.

The data memory is organized into 16 banks, designated Bank 0 through Bank F, each containing 16 registers, plus an additional bank of 15 “global” registers. Because the registers are organized into banks or “files,” these memory-mapped registers are called “file registers.”

5.2.1 Addressing Modes

Each SX instruction that accesses a data memory register contains a 5-bit field in the instruction opcode that specifies the register to be accessed. The abbreviation “fr” represents the 5-bit register address designator. For example, the instruction description “mov fr,W” means that a 5-bit value or label must be substituted for “fr” in the instruction, such as “mov $0F,W” (to move the contents of the working register W into file register 0Fh).

There are three different addressing modes, called the indirect, direct, and semi-direct modes. The addressing mode used for register access depends on the 5-bit “fr” value used in the instruction:

- indirect mode: fr = 00h
- direct mode: fr = 01h through 0Fh
- semi-direct mode: fr = 10h through 1Fh

For indirect addressing (fr=00), the File Select Register (FSR) specifies the register to be accessed. FSR is an 8-bit, memory-mapped register (at address 04h) which serves as an 8-bit pointer into data memory for indirect addressing. In this mode, the global register bank and Bank 1 through Bank F are accessible. Bank 0 is not accessible.

For direct addressing (fr=01-0F), the value of “fr” itself specifies the register to be accessed, and the FSR register is ignored. For this addressing mode, only the global register bank is accessible. To gain access to any other bank, you must use either indirect or semi-direct addressing.

For semi-direct addressing (fr=10-1F), the bank number is selected by the four high-order bits of FSR, and the register within that bank is selected by the four low-order bits of “fr.” In other words, the register address is obtained by combining the four high-order bits of FSR with the four low-order bits of “fr”. In this addressing mode, the low-order bits of FSR are ignored. Bank 0 through Bank F are accessible, but the global register bank is not accessible.

Figure 5-1 shows how register addressing works in the indirect, direct, and semi-direct modes. The 15 global registers are always accessible by direct addressing, regardless of what is contained in the FSR register. The global registers are also accessible with indirect addressing , but they are not accessible with semi-direct addressing. Of the 15 global registers, the first nine are special-purpose registers (RTCC, PC, STATUS, and so on), and the next six are general-purpose registers. All of the registers in Bank 0 though Bank F are general-purpose registers.

To change the contents of the FSR register, the program can either write an eight-bit value to the FSR register or use the “bank” instruction. The “bank” instruction writes the three high-order bits in the FSR register without affecting the other bits in the register. Thus, the “bank” instruction lets you quickly change from one even-numbered bank to another, or from one odd-numbered bank to another.

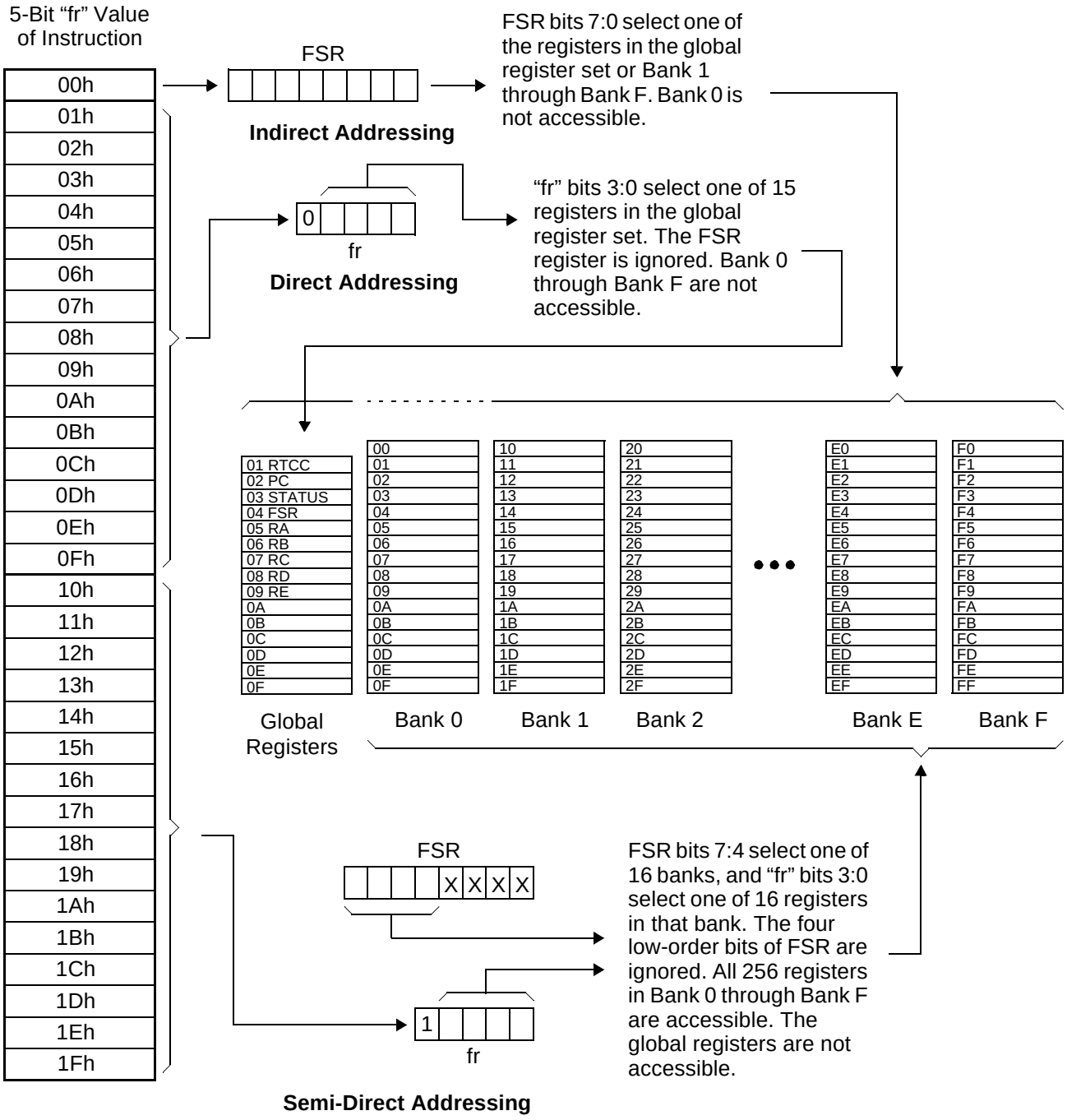


Figure 5-1. Register Access Modes

5.2.2 Register Access Examples

Here is an example of an instruction that uses direct addressing:

```
inc $0F   ;increment file register 0Fh
```

This instruction increments the contents of file register 0Fh in the global register bank. It does not matter what is contained in the FSR register.

To gain access to any register outside of the global register bank, it is necessary to use semi-direct or indirect addressing. In that case, you need to make sure that the FSR register contains the correct value for accessing the desired bank.

Here is an example that uses semi-direct addressing:

```
mov W,#$F0  ;load W with F0h
mov FSR,W   ;set upper 4 bits of FSR (Bank F)
inc $1F     ;increment file register FFh
```

In this example, “FSR” is a label that represents the value 04h, which is the address of the FSR register in the global register bank. Note that the FSR register is itself a memory-mapped global register, which is always accessible using direct addressing.

To use the “bank” instruction, in the syntax of the assembly language, you specify an 8-bit value that corresponds to the desired bank number. The assembler encodes the three high-order bits of the specified value into the instruction opcode and ignores the low-order bits. In addition, the “bank” instruction forces a zero to bit 4 of the FSR register. For example, to increment file register 2Fh, you could use the following instructions:

```
bank $20   ;select Bank 2 in FSR
inc $1F    ;increment register 2F
```

Note that the “bank” instruction only modifies the upper three (not four) bits in the FSR register. Therefore, to change from an even-numbered to odd-numbered bank (or from an odd-numbered to even-numbered bank), the “bank” instruction will not work. Instead, you need to write the whole FSR register using code such as the following:

```
mov W,#$30  ;load W with 30h
mov FSR,W   ;select Bank 3 in FSR
```

Another approach to set bit 4 of the FSR register individually after the “bank” instruction to address an odd-numbered bank.

With indirect addressing, you specify the full 8-bit address of the register using FSR as a pointer. This addressing mode provides the flexibility to access different registers or multiple registers using the same instruction in the program.

You invoke indirect addressing by using fr=00h. For example:

```
mov W,#$F5  ;load W with F5h
mov $04,W   ;move value F5h into FSR
mov W,#$01  ;load W with 01h
mov $00,W   ;move value 01h into register F5h
```

In the second “mov” instruction, FSR is loaded with the desired 8-bit register address. In the fourth “mov” instruction, fr = 00, so the device looks at FSR and moves the result to the register addressed by FSR, which is the register at F5h (Bank F, register number 5).

A practical example that uses indirect addressing is the following program, which clears the upper eight registers in the global register bank and all banks from Bank 1 through Bank F:

```
 clr FSR     ;clear FSR to 00h (at address 04h)
:loop setb FSR.3   ;set FSR bit 3
 clr $00     ;clear register pointed to by FSR
 incsz FSR   ;increment FSR and test
             ;skip jmp if 00h
 jmp :loop   ;jump back and clear next reg.
```

This program initially clears FSR to 00h. At the beginning of the loop, it sets bit 3 of FSR so that it starts at 08h. The “clr $00” instruction clears the register pointed to by FSR (initially, the file register at 08h in the global register bank). Then the program increments FSR and clears consecutive file registers, always in the upper half of each bank: (08h, 09h, 0Ah ... 0Fh, 18h, 19h ... FFh). The loop ends when FSR wraps back to 00h.

For addresses from 01h through 0Fh, the global register bank is accessed. For higher addresses, Bank 1 through Bank F are accessed. This program does not affect Bank 0, which is not accessible in the indirect addressing mode. Bank 0 can be accessed only using the semi-direct mode.

6.0 POWER DOWN MODE

The power down mode is entered by executing the SLEEP instruction.

In power down mode, only the Watchdog Timer (WDT) is active. If the Watchdog Timer is enabled, upon execution of the SLEEP instruction, the Watchdog Timer is cleared, the TO (time out) bit is set in the STATUS register, and the PD (power down) bit is cleared in the STATUS register.

There are three different ways to exit from the power down mode: a timer overflow signal from the Watchdog Timer (WDT), a valid transition on any of the Multi-Input Wakeup pins (Port B pins), or through an external reset input on the MCLR pin.

To achieve the lowest possible power consumption, the Watchdog Timer should be disabled and the device should exit the power down mode through the (Multi-Input Wakeup) MIWU pins or an external reset.

Bit 11 of the FUSEX can be used to enable (clear bit to 0) the clock operation during the sleep mode (to allow fast clock start-up upon existing the power down mode).

6.1 Multi-Input Wakeup

Multi-Input Wakeup is one way of causing the device to exit the power down mode. Port B is used to support this feature. The WKEN_B register (Wakeup Enable Register) allows any Port B pin or combination of pins to cause the wakeup. Clearing a bit in the WKEN_B register enables the wakeup on the corresponding Port B pin. If multi-input wakeup is selected to cause a wakeup, the trigger condition on the selected pin can be either rising edge (low to high) or falling edge (high to low). The WKED_B register (Wakeup Edge Select) selects the desired transition edge. Setting a bit in the WKED_B register selects the falling edge on the corresponding Port B. Resetting the bit selects the rising edge. The WKEN_B and WKED_B registers are set to FFh upon reset.

Once a valid transition occurs on the selected pin, the WKPND_B register (Wakeup Pending Register) latches the transition in the corresponding bit position. A logic '1' indicates the occurrence of the selected trigger edge on the corresponding Port B pin. The WKPND_B comes up with undefined value upon reset. The user program must clear the WKPND_B register prior to enabling the interrupt.

Upon exiting the power down mode, the Multi-Input Wakeup logic causes program counter to branch to the maximum program memory address (same as reset).

Figure 6-1 shows the Multi-Input Wakeup block diagram.

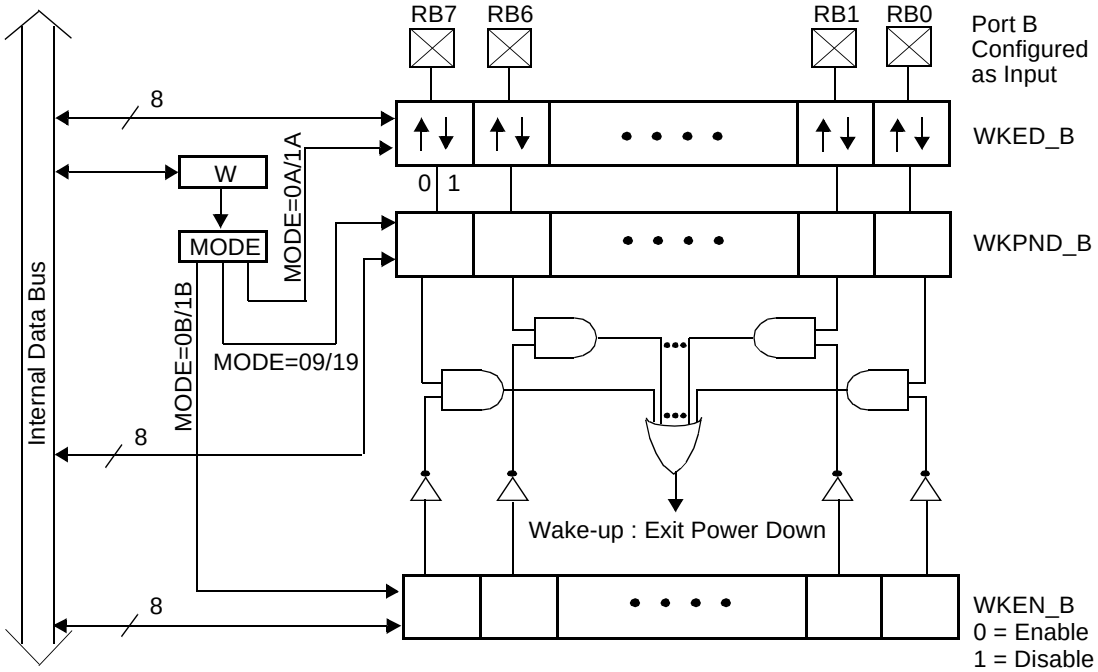


Figure 6-1. Multi-Input Wakeup Block Diagram

6.2 Port B MIWU/Interrupt Configuration

The WKPND_B register comes up with a random value upon reset. The user program must clear the register prior to enabling the wake-up condition or interrupts. The proper initialization sequence is:

1. Select the desired edge (through WKED_B register)
2. Clear the WKPND_B register
3. Enable the Wakeup condition (through WKEN_B register)

Below is an example of how to read the WKPND_B register to determine which Port B pin caused the wakeup or interrupt, and to clear the WKPND_B register:

```
mov   W, #$19
mov   M, W
clr   W
mov   !RB,W   ;W contains WKPND_B
              ;contents of W exchanged
              ;with contents of WKPND_B
```

The final “mov” instruction in this example performs an exchange of data between the working register (W) and the WKPND_B register. This exchange occurs only with accesses to the WKPND_B and CMP_B registers. Otherwise, the “mov” instruction does not perform an exchange, but only moves data from the source to the destination.

Here is an example of a program segment that configures the RB0, RB1, and RB2 pins to operate as Multi-Input Wakeup/Interrupt pins, sensitive to falling edges:

```
mov W,#$1F  ;prepare to write port data
mov M,W     ;direction registers
mov W,#$07  ;load W with the value 07h
mov !RB,W   ;configure RB0-RB2 to be inputs

mov W,#$1A  ;prepare to write WKED_B
mov M,W     ;(edge) register
            ;W contains the value 07h
mov !RB,W   ;configure RB0-RB2 to sense
            ;falling edges

mov W,#$19  prepare to access WKPND_B
mov M,W     (pending) register
mov W,#$00  clear W
mov !RB,W   clear all wakeup pending flags

mov W,#$1B  ;prepare to write WKEN_B (enable)
mov M,W     ;register
mov W,#$F8h ;load W with the value F8h
mov !RB,W   ;enable RB0-RB2 to operate as
            ;wakeup inputs
```

To prevent false interrupts, the enabling step (clearing bits in WKEN_B) should be done as the last step in a sequence of Port B configuration steps.

After this program segment is executed, the device can receive interrupts on the RB0, RB1, and RB2 pins. If the device is put into the power down mode (by executing a “sleep” instruction), the device can then receive wakeup signals on those same pins.

7.0 INTERRUPT SUPPORT

The device supports both internal and external maskable interrupts. The internal interrupt is generated as a result of the RTCC rolling over from FFh to 00h. This interrupt source has an associated enable bit located in the OPTION register and pending flag bit in the Timer T1 Control B register. In addition, timers T1 and T2 each has three interrupt sources associated with counter overflow, compare match, and input capture.

Port B provides the source for eight external software selectable, edge sensitive interrupts, when the device is not in the SLEEP mode. These interrupt sources share logic with the Multi-Input Wakeup circuitry. The WKEN_B register allows interrupt from Port B to be individually enabled or disabled. Clearing a bit in the WKEN_B register enables the interrupt on the corresponding Port B pin. The WKED_B selects the transition edge to be either positive or negative. The WKEN_B and WKED_B registers are set to FFh upon reset. Setting a bit in the WKED_B register selects the falling edge while resetting the bit selects the rising edge on the corresponding Port B pin.

The WKPND_B register serves as the external interrupt pending register.

The WKPND_B register comes up a with random value upon reset. The user program must clear the WKPND_B register prior to enabling the interrupt. .

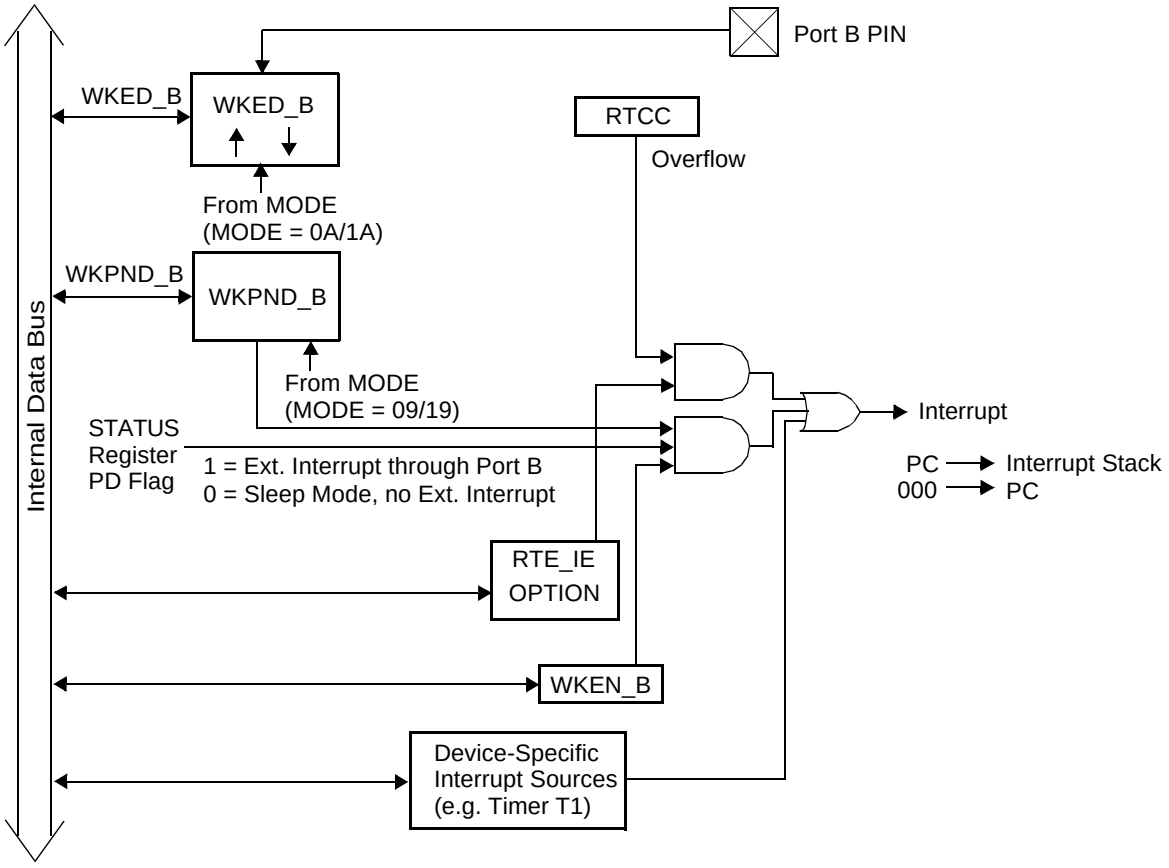


Figure 7-1. Interrupt Structure

All interrupts are global in nature; that is, no interrupt has priority over another. Interrupts are handled sequentially. Figure 7-2 shows the interrupt processing sequence. Once an interrupt is acknowledged, all subsequent global interrupts are disabled until return from servicing the current interrupt. The PC is pushed onto the single level interrupt stack, and the contents of the FSR, STATUS, and W registers are saved in their corresponding shadow registers. Bits PA0, PA1, and PA2 of the STATUS register are cleared after the STATUS register has been saved in its corresponding shadow register. The interrupt logic has its own single-level stack and is not part of the CALL subroutine stack. The vector for the interrupt service routines is address 0.

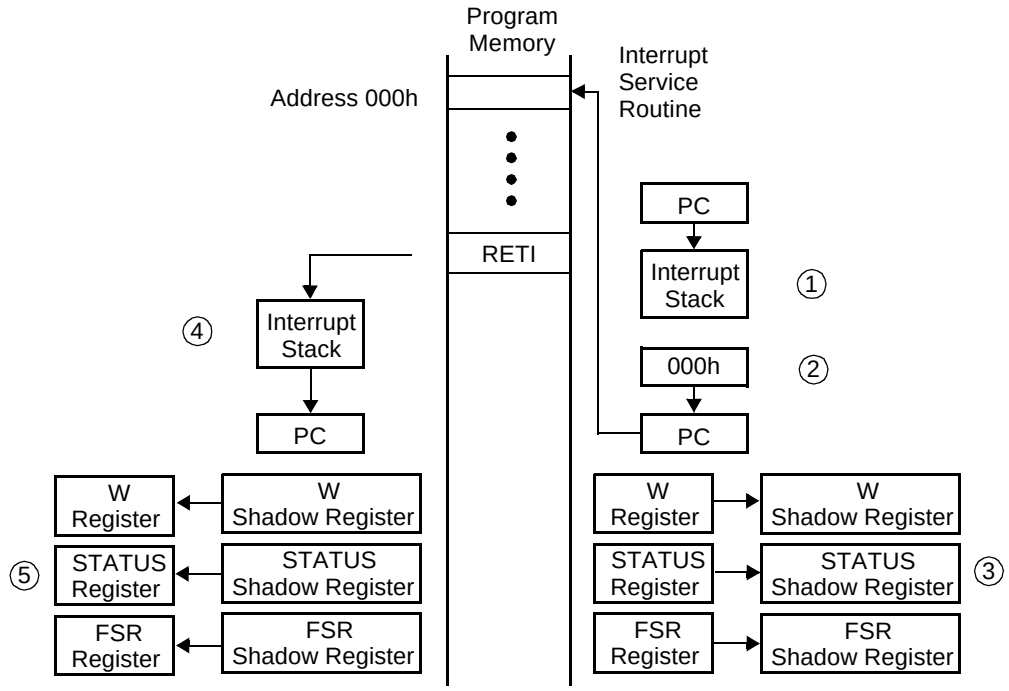
Once in the interrupt service routine, the user program must poll all interrupt pending bits to determine the source of the interrupt. The interrupt service routine should clear the corresponding interrupt pending flag.

Normally it is a requirement for the user program to process every interrupt without missing any. To ensure this, the longest path through the interrupt routine must take less time than the shortest possible delay between interrupts.

Upon return from the interrupt service routine, the contents of PC, FSR, STATUS, and W registers are restored from their corresponding shadow registers. The interrupt service routine should end with instructions such as RETI

or RETIW. RETI pops the interrupt stack and the special shadow registers used for storing W, STATUS, and FSR (preserved during interrupt handling). RETIW behaves like RETI but also adds a literal to RTCC. The interrupt return instruction enables the global interrupts.

If a MIWU interrupt occurs during a pre-existing interrupt service routine, the MIWU interrupt flag is set immediately, and the MIWU interrupt is serviced upon completion of the pre-existing interrupt service routine.



Note:The interrupt logic has its own single-level stack and is not part of the CALL subroutine stack.

Figure 7-2. Interrupt Processing

8.0 OSCILLATOR CIRCUITS

The device supports several user-selectable oscillator modes. The oscillator modes are selected by programming the appropriate values into the FUSE Word register. These are the different oscillator modes offered:

LP: Low Power Crystal
XT: Crystal/Resonator
HS: High Speed Crystal/Resonator
RC: External Resistor/Capacitor
Internal Resistor/Capacitor

8.1 XT, LP or HS modes

In XT, LP or HS, modes, you can use either an external resonator network or an external clock signal as the device clock.

To use an external resonator network, you connect a crystal or ceramic resonator to the OSC1/CLKIN and OSC2/CLKOUT pins according to the circuit configuration shown in Figure 8-1. Table 8-1 shows the recommended capacitor values to be used with ceramic resonators. Table 8-2 shows the recommended component values to be used at different crystal frequencies. A parallel resonant crystal type is recommended. Use of a series resonant crystal may result in a frequency that is outside the crystal manufacturer specifications.

If the XT, LP, or HS mode is selected, the OSC1/CLKIN pin can driven by an external clock source rather than a resonator network, as long as the clock signal meets the specified duty cycle, rise and fall times, and input levels (Figure 8-2). In this case, the OSC2/CLKOUT pin should be left open.

Note:Series resistor (R_S) is not required for frequencies higher than 1 MHz; use a direct connection instead.

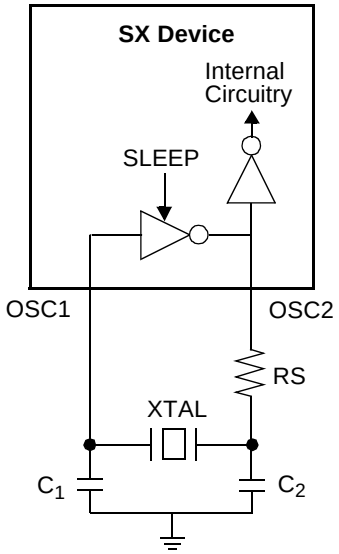


Figure 8-1. Crystal Operation (or Ceramic Resonator) (HS, XT or LP OSC Configuration)

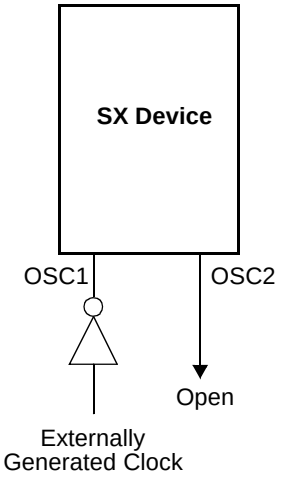


Figure 8-2. External Clock Input Operation (HS, XT or LP OSC Configuration)

Table 8-1. Capacitor Selection for Ceramic Resonators

Clock Mode	Resonator Frequency	C1	C2	R_S
XT	455 kHz	220 pF	220 pF	6.8 kΩ
XT	1 MHz	100 pF	100 pF	6.8 kΩ
XT	2 MHz	100 pF	100 pF	680 Ω
HS	4 MHz	100 pF	100 pF	0
HS	4 MHz	Internal (47 pF)	Internal (47 pF)	470 Ω
HS	8 MHz	30 pF	30 pF	0
HS	8 MHz	Internal (47 pF)	Internal (47 pF)	470 Ω
HS	12 MHz	30 pF	30 pF	0
HS	12 MHz	Internal (22 pF)	Internal (22 pF)	0
HS	16 MHz	15 pF	15 pF	0
HS	16 MHz	Internal (15 pF)	Internal (15 pF)	0
HS	20 MHz	10 pF	10 pF	0
HS	33 MHz	10 pF	10 pF	0
HS	50 MHz*	Internal (5 pF)	Internal (5 pF)	0

* Note: Resonator with built-in capacitors.

Table 8-2. Component Selection for Crystal Oscillator

Osc Type	Resonator Frequency	C1 (pF)	C2 (pF)
XT	4	20	47
HS	8	20	47
HS	12	20	47
HS	16	15	30
HS	20	15	30
HS	25	5	20
HS	30	5	20
HS	36	5	15
HS	40	5	15

8.2 External RC Mode

The external RC oscillator mode provides a cost-effective approach for applications that do not require a precise operating frequency. In this mode, the RC oscillator frequency is a function of the supply voltage, the resistor (R) and capacitor (C) values, and the operating temperature. In addition, the oscillator frequency will vary from unit to unit due to normal manufacturing process variations. Furthermore, the difference in lead frame capacitance between package types also affects the oscillation frequency, especially for low C values. The external R and C component tolerances contribute to oscillator frequency variation as well.

Figure 8-3 shows the external RC connection diagram. The recommended R value is from 3kΩ to 100kΩ. For R values below 2.2kΩ, the oscillator may become unstable, or may stop completely. For very high R values (such as 1 MΩ), the oscillator becomes sensitive to noise, humidity, and leakage.

Although the oscillator will operate with no external capacitor (C = 0pF), it is recommended that you use values above 20 pF for noise immunity and stability. With no or small external capacitance, the oscillation frequency can vary significantly due to variation in PCB trace or package lead frame capacitances.

In the external RC mode, the OSC2/CLKOUT pin provides an output frequency, which the input frequency divided by four.

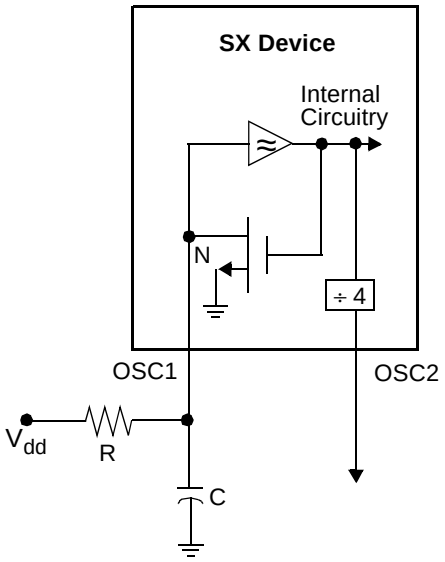


Figure 8-3. RC Oscillator Mode

8.3 Internal RC Mode

The internal RC mode uses an internal oscillator, so the device does not need any external components. At 4 MHz, the internal oscillator provides +/–8% accuracy over the allowed temperature range. The internal clock frequency can be divided down to provide one of eight lower-frequency choices by selecting the desired value in the FUSE Word register. The frequency range is from 31.25 kHz to 4 MHz.The default operating frequency of the internal RC oscillator may not be 4 MHz. This is due to the fact that the SX device requires trimming to obtain 4 MHz operation. The parts shipped out of the factory are not trimmed. The device relies on the programming tool provided by the third party vendors to support trimming.

9.0 REAL TIME CLOCK (RTCC)/WATCHDOG TIMER

The device contains an 8-bit Real Time Clock/Counter (RTCC) and an 8-bit Watchdog Timer (WDT). An 8-bit programmable prescaler extends the RTCC to 16 bits. If the prescaler is not used for the RTCC, it can serve as a postscaler for the Watchdog Timer. Figure 9-1 shows the RTCC and WDT block diagram.

9.1 RTCC

RTCC is an 8-bit real-time timer that is incremented once each instruction cycle or from a transition on the RTCC pin. The on-board prescaler can be used to extend the RTCC counter to 16 bits.

The RTCC counter can be clocked by the internal instruction cycle clock or by an external clock source presented at the RTCC pin.

To select the internal clock source, bit 5 of the OPTION register should be cleared. In this mode, RTCC is incremented at each instruction cycle unless the prescaler is selected to increment the counter.

To select the external clock source, bit 5 of the OPTION register must be set. In this mode, the RTCC counter is incremented with each valid signal transition at the RTTC pin. By using bit 4 of the OPTION register, the transition can be programmed to be either a falling edge or rising edge. Setting the control bit selects the falling edge to increment the counter. Resetting the bit selects the rising edge.

The RTCC generates an interrupt as a result of an RTCC rollover from FFh to 00h. Bit 7 of the Timer T1 Control B register is an interrupt pending flag associated with this event. The program should read this flag to determine any overflow occurrence. Writing to the RTCC clears the prescaler if it is assigned to the RTCC (PSA bit in the OPTION register is 0).

9.2 Watchdog Timer

The watchdog logic consists of a Watchdog Timer which shares the same 8-bit programmable prescaler with the RTCC. The prescaler actually serves as a postscaler if used in conjunction with the WDT, in contrast to its use as a prescaler with the RTCC.

9.3 The Prescaler

The 8-bit prescaler may be assigned to either the RTCC or the WDT through the PSA bit (bit 3 of the OPTION register). Setting the PSA bit assigns the prescaler to the WDT. If assigned to the WDT, the WDT clocks the prescaler and the prescaler divide rate is selected by the PS0, PS1, and PS2 bits located in the OPTION register. Clearing the PSA bit assigns the prescaler to the RTCC. Once assigned to the RTCC, the prescaler clocks the RTCC and the divide rate is selected by the PS0, PS1, and PS2 bits in the OPTION register. The prescaler is not mapped into the data memory, so run-time access is not possible.

The prescaler cannot be assigned to both the RTCC and WDT simultaneously.

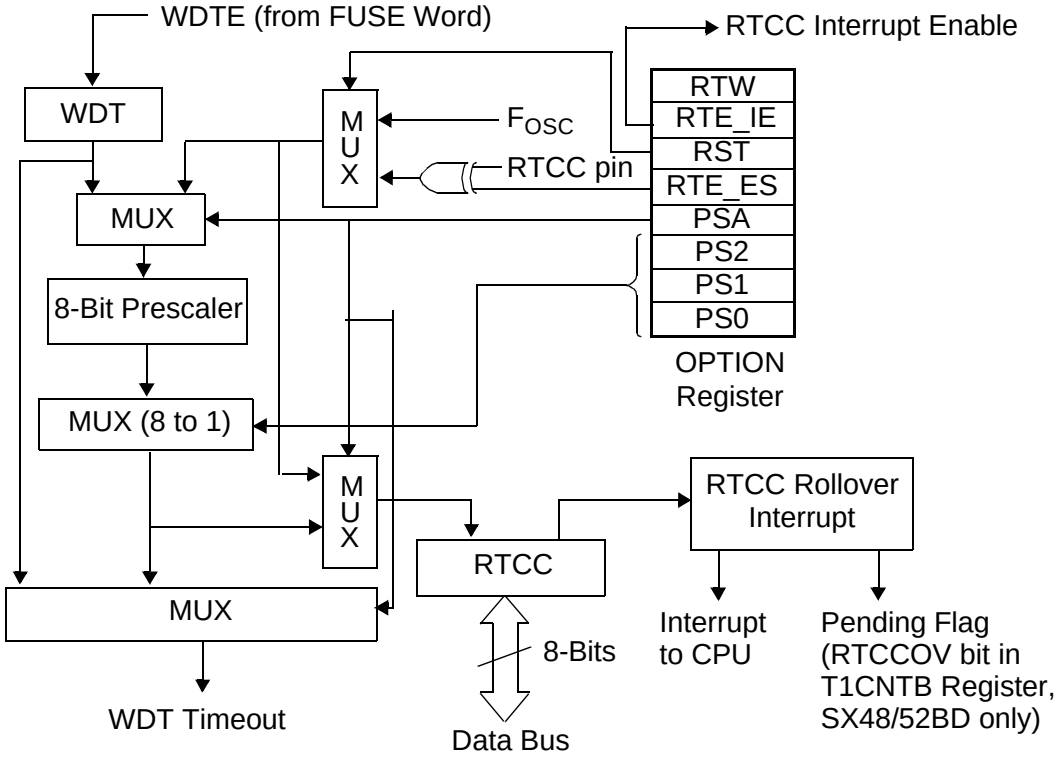


Figure 9-1. RTCC and WDT Block Diagram

10.0 MULTI-FUNCTION TIMERS

The device contains two independent 16-bit multi-function timers, designated T1 and T2. These versatile, programmable timers reduce the software burden on the CPU in real-time control applications such as PWM generation, motor control, triac control, variable-brightness display control, sine wave generation, and data acquisition.

Each timer consists of a 16-bit counter register supported by a 16-bit capture register and a 16-bit comparison register. Each timer uses up to four I/O pins: one clocking input, two capture inputs, and one timer output. The timer I/O pins are alternate functions of Port B pins for timer T1 and Port C pins for Timer T2.

Figure 10-1 is a block diagram showing the registers and I/O pins of one timer. The 16-bit free-running timer/counter register is initialized to 0000h upon reset and counts upward continuously. It is clocked either by an external signal provided on an I/O pin or by the on-chip system clock divided by a programmable 8-bit prescaler register.

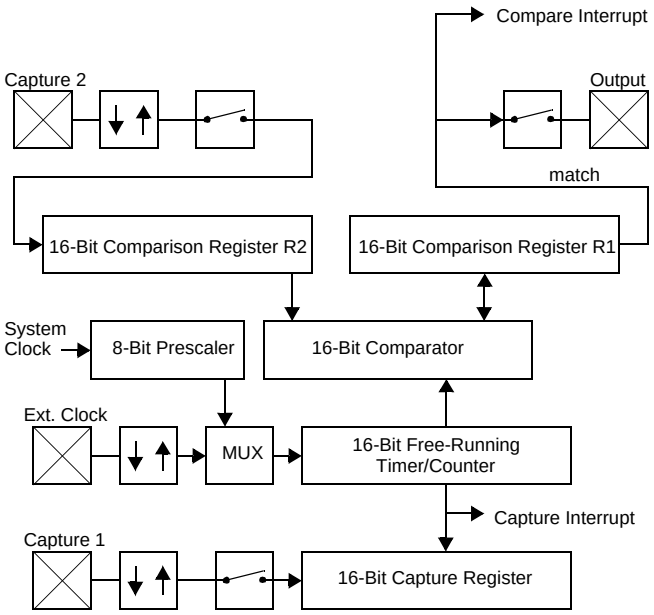


Figure 10-1. Multi-Function Timer Block Diagram

The CPU can access the R1, R2, and Capture registers by using the “mov !RB,W” instruction for T1 or the “mov !RC,W” instruction for T2. The other timer registers are not directly accessible.

You can configure the timer to generate an interrupt upon overflow from FFFFh to 0000h, upon a match between the counter value and a programmed comparison value, or upon the occurrence of a valid capture signal on either of two capture inputs.

10.1 Timer Operating Modes

Each timer can be configured to operate in one of the following modes:

- Pulse Width Modulation (PWM) mode
- Software Timer mode
- External Event mode
- Capture/Compare mode

10.1.1 PWM Mode

In the Pulse Width Modulation (PWM) mode, the timer generates an output signal having a programmable frequency and duty cycle. To use this mode, you load two 16-bit comparison registers, R1 and R2, with the number of timer clock cycles that you want the output signal to be high and low.

The timer starts from zero and counts up until it reaches the value in R1. At that point, it generates an interrupt (if enabled), toggles the output signal, and starts counting from zero again. The second time, it counts up until it reaches the value in R2. At that point, it again generates an interrupt (if enabled), toggles the output signal, and starts counting from zero again. This process is repeated continuously, alternating between R1 and R2 to obtain the value at which to toggle the output signal and return the counter to zero. The values of R1 and R2 establish the duty cycle and frequency of the output signal. If R1 and R2 contain the same value, the resulting output signal is a square wave.

In the PWM mode, the timer is clocked by the on-chip system clock divided by an 8-bit prescaler value. The divide-by factor can be set to any power-of-2 from 1 to 256. Thus, the period of the timer clock can be set from 1 to 256 times the system clock period.

10.1.2 Software Timer Mode

The Software Timer mode is the same as the PWM mode, except that the timer does not toggle the output signal. Instead, the application program takes action in response to the interrupts generated upon each match between the counter and the contents of the active comparison value in either R1 or R2. The software can determine the cause of each interrupt by checking the timer interrupt pending flags. There are different flag bits associated with each type of event (R1 match, R2 match, and overflow).

10.1.3 External Event Mode

The External Event mode is the same as the PWM mode, except that the counter register is clocked by an external signal provided on an input pin rather than by the system clock. This mode can be used to count the occurrences of external events. The input pin can be configured to sense either rising or falling edges.

10.1.4 Capture/Compare Mode

In the Capture/Compare mode, the counter counts upward continuously without interruption. A valid transition received on either of two input pins causes the current value of the counter to be captured in an associated

capture register. This capture feature can be used to keep track of the elapsed time between successive external events. In addition, the timer continuously compares the counter value against the value programmed into the R1 register. Each time a match occurs, it toggles the timer output pin, and also generates an interrupt (if enabled). The timer continues to count upward after a match occurs (unlike the PWM mode, which resets the counter to zero when a match occurs).

In the Capture/Compare mode, the timer is clocked by the on-chip system clock divided by an 8-bit prescaler value. The divide-by factor can be set to any power-of-2 from 1 to 256.

The two input capture pins are designated Capture 1 and Capture 2. They can be configured to sense either rising or falling edges. The Capture 1 pin captures the counter value in a dedicated 16-bit capture register, a read-only register. The Capture 2 pin captures the counter value in the R2 register. The occurrence of a capture event also generates an interrupt (if enabled) and sets an associated interrupt pending flag.

Overflow of the counter from FFFFh to 0000h also generates an interrupt (if enabled) and sets an associated interrupt pending flag. Because the counter is free-running, an overflow can occur at any time. In cases where the time between successive capture events might exceed 65,536 counts of the timer, the software should keep track of the number of overflows between successive events in order to determine the true amount of time between such events.

An occurrence of a match between the counter value and the programmed value in the R1 register toggles the timer output pin. It also generates an interrupt (if enabled) and sets an associated interrupt pending flag. The timer continues to run freely without being reset to zero. Therefore, if you want to have a fixed timeout period for this interrupt, the interrupt service routine must write a new value into R1 each time a match occurs.

10.2 Timer Pin Assignments

The following table lists the I/O port pins associated with the Timer T1 and Timer T2 I/O functions.

Table 10-1. Timer T1/T2 Pin Assignments

I/O Pin	Timer T1/T2 Function
RB4	Timer T1 Capture Input 1
RB5	Timer T1 Capture Input 2
RB6	Timer T1 PWM/Compare Output
RB7	Timer T1 External Event Clock Source
RC0	Timer T2 Capture Input 1
RC1	Timer T2 Capture Input 2
RC2	Timer T2 PWM/Compare Output
RC3	Timer T2 External Event Clock Source

10.3 Timer Control Registers

There are two 8-bit control registers associated with each timer, called the Control A and Control B registers. The Control A register contains the interrupt enable bits and interrupt flag bits associated with the timer. (Interrupts are caused by comparison, capture, and overflow events.) The Control B register contains bits for setting the timer operating mode, the clock prescaler divide-by factor, and the input signal edge sensitivity. Each Control B register also contains one device configuration bit not related to operation of the multi-function timers.

The register formats are shown in the following diagrams.

Timer T1 Control A Register (T1CNTA)

T1CPF2	T1CPF1	T1CPIE	T1CMF2	T1CMF1	T1CMIE	T1OVF	T1OVIE
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**

T1CPF2 Timer T1 Capture Flag 2. In Capture/Compare mode, this flag is automatically set to 1 when a capture event occurs on the Capture 2 pin of Timer T1 (pin RB5). It stays set until cleared by the software.

T1CPF1 Timer T1 Capture Flag 1. In Capture/Compare mode, this flag is automatically set to 1 when a capture event occurs on the Capture 1 pin of Timer T1 (pin RB4). It stays set until cleared by the software.

T1CPIE Timer T1 Capture Interrupt Enable. Set this bit to 1 to enable capture interrupts for Timer T1 in Capture/Compare mode. In that case, an interrupt will occur each time a valid edge is received on the Capture 1 or Capture 2 pin of Timer T1. Clear this bit to 0 to disable capture interrupts.

T1CMF2 Timer T1 Comparison Flag 2. This flag is automatically set to 1 when the contents of the timer counter match the contents of R2, when R2 is the active comparison register. The flag stays set until it is cleared by the software.

T1CMF1 Timer T1 Comparison Flag 1. This flag is automatically set to 1 when the contents of the timer counter match the contents of R1, when R1 is the active comparison register. The flag stays set until it is cleared by the software.

T1CMIE Timer T1 Comparison Interrupt Enable. Set this bit to 1 to enable comparison interrupts for Timer T1. In that case, an interrupt will occur each time the contents of the timer counter match the contents of the active comparison register (R1 or R2) of Timer T1. Clear this bit to 0 to disable comparison interrupts.

T1OVF — Timer T1 Overflow Flag. This flag is automatically set to 1 when the timer counter overflows from FFFFh to 0000h. The flag stays set until it is cleared by the software.

T1OVIE — Timer T1 Overflow Interrupt Enable. Set this bit to 1 to enable overflow interrupts for Timer T1. In that case, an interrupt will occur each time Timer T1 overflows. Clear this bit to 0 to disable overflow interrupts.

Timer T1 Control B Register (T1CNTB)

RTCCOV	T1CPEDG	T1EXEDG	T1PS2-T1PS0			T1MC1-T1MC0	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**

RTCCOV — RTCC Overflow Flag. This flag is automatically set to 1 when the Real-Time Clock/Counter (RTCC) overflows from FFh to 00h. This flag stays set until it is cleared by the software. Note that this flag is not related to multi-function timers T1 and T2.

T1CPEDG — Timer T1 Capture Edge. This bit sets the edge sensitivity of the Timer T1 input capture pins, Capture 1 and Capture 2 (RB4 and RB5). Set this bit to 1 to sense positive-going (low-to-high) edges. Clear this bit to 0 to sense negative-going (high-to-low) edges.

T1EXEDG — Timer T1 External Event Clock Edge. This bit sets the edge sensitivity of the Timer T1 input used to count external events (RB7). Set this bit to 1 to sense positive-going (low-to-high) edges. Clear this bit to 0 to sense negative-going (high-to-low) edges.

T1PS2-T1PS0 — Timer T1 Prescaler Divider field. This 3-bit field specifies the divide-by factor for generating the timer clock from the on-chip system clock:

000 = divide by 1
001 = divide by 2
010 = divide by 4
011 = divide by 8
100 = divide by 16
101= divide by 32
110 = divide by 64
111 = divide by 128

For example, setting this field to 010 sets the divide-by factor to 4, which means that the T1 counter register is incremented once every four system clock cycles.

T1MC1-T1MC0 — Timer T1 Mode Control field. This 2-bit field specifies the Timer T1 operating mode as follows:

00 = Software Timer mode
01 = PWM mode
10 = Capture/Compare mode
11 = External Event mode

Timer T2 Control A Register (T2CNTA)

T2CPF2	T2CPF1	T2CPIE	T2CMF2	T2CMF1	T2CMIE	T2OVF	T2OVIE
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**

T2CPF2 — Timer T2 Capture Flag 2. In Capture/Compare mode, this flag is automatically set to 1 when a capture event occurs on the Capture 2 pin of Timer T2 (pin RC1). It stays set until cleared by the software.

T2CPF1 — Timer T2 Capture Flag 1. In Capture/Compare mode, this flag is automatically set to 1 when a capture event occurs on the Capture 1 pin of Timer T2 (pin RC1). It stays set until cleared by the software.

T2CPIE — Timer T2 Capture Interrupt Enable. Set this bit to 1 to enable capture interrupts for Timer T2 in Capture/Compare mode. In that case, an interrupt will occur each time a valid edge is received on the Capture 1 or Capture 2 pin of Timer T2. Clear this bit to 0 to disable capture interrupts.

T2CMF2 — Timer T2 Comparison Flag 2. This flag is automatically set to 1 when the contents of the timer counter match the contents of R2, when R2 is the active comparison register. The flag stays set until it is cleared by the software.

T2CMF1 Timer T2 Comparison Flag 1. This flag is automatically set to 1 when the contents of the timer counter match the contents of R1, when R1 is the active comparison register. The flag stays set until it is cleared by the software.

T2CMIE Timer T2 Comparison Interrupt Enable. Set this bit to 1 to enable comparison interrupts for Timer T2. In that case, an interrupt will occur each time the contents of the timer counter match the contents of the active comparison register (R1 or R2) of Timer T2. Clear this bit to 0 to disable comparison interrupts.

T2OVF Timer T2 Overflow Flag. This flag is automatically set to 1 when the timer counter overflows from FFFFh to 0000h. The flag stays set until it is cleared by the software.

T2OVIE Timer T2 Overflow Interrupt Enable. Set this bit to 1 to enable overflow interrupts for Timer T2. In that case, an interrupt will occur each time Timer T2 overflows. Clear this bit to 0 to disable overflow interrupts.

Timer T2 Control B Register (T2CNTB)

PORTRD	T2CPEDG	T2EXEDG	T2PS2-T2PS0			T2MC1-T2MC0	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**

PORTRD Port Read mode. This bit determines how the device reads data from its I/O ports (Port A through Port E). Set this bit to 1 to have the device read data from the port I/O pins directly. Clear this bit to 0 to have the device read data from the port data registers. Under normal conditions, it should not matter which method you use to read the port data. However, if a port pin is configured as an output and an external circuit forces the pin to the wrong value, the value read from the port will depend on the reading mode used. Note that this control bit is not related to multi-function timers T1 and T2.

T2CPEDG Timer T2 Capture Edge. This bit sets the edge sensitivity of the Timer T2 input capture pins, Capture 1 and Capture 2 (RC0 and RC1). Set this bit to 1 to sense positive-going (low-to-high) edges. Clear this bit to 0 to sense negative-going (high-to-low) edges.

T2EXEDG Timer T2 External Event Clock Edge. This bit sets the edge sensitivity of the Timer T2 input used to count external events (RC3). Set this bit to 1 to sense positive-going (low-to-high) edges. Clear this bit to 0 to sense negative-going (high-to-low) edges.

T2PS2-T2PS0 Timer T2 Prescaler Divider field. This 3-bit field specifies the divide-by factor for generating the timer clock from the on-chip system clock:

000 = divide by 1
001 = divide by 2
010 = divide by 4
011 = divide by 8
100 = divide by 16
101 = divide by 32
110 = divide by 64
111 = divide by 128

For example, setting this field to 010 sets the divide-by factor to 4, which means that the T2 counter register is incremented once every four system clock cycles.

T2MC1-T2MC0 Timer T2 Mode Control field. This 2-bit field specifies the Timer T1 operating mode as follows:

00 = Software Timer mode
01 = PWM mode
10 = Capture/Compare mode
11 = External Event mode

11.0 COMPARATOR

The device contains an on-chip differential comparator. Ports RB0-RB2 support the comparator. Ports RB1 and RB2 are the comparator negative and positive inputs, respectively, while Port RB0 serves as the comparator output pin. To use these pins in conjunction with the comparator, the user program must configure Ports RB1 and RB2 as inputs and Port RB0 as an output. The CMP_B register is used to enable the comparator, to read the output of the comparator internally, and to enable the output of the comparator to the comparator output pin.

The comparator enable bits are set to “1” upon reset, thus disabling the comparator. To avoid drawing additional current during the SLEEP mode, the comparator should be disabled before entering the SLEEP mode. Here is an example of how to set up the comparator and read the CMP_B register.

```
mov  W,#$18   ;set MODE register to access
mov  M,W      ;CMP_B

mov  W,#$00   ;clear W

mov  !RB,W    ;enable comparator and its
              ;output

...           ;delay after enabling
              ;comparator for response

mov  W,#$18   ;set MODE register to access
mov  M,W      ;CMP_B

mov  W,#$00   ;clear W

mov  !RB,W    ;enable comparator and its
              ;output and also read CMP_B
              ;(exchange W and CMB_B)

and  W,#$01   ;set/clear Z flag based on
              ;comparator result

snb  $03.2    ;test Z flag in STATUS reg
              ;(0 => RB2<RB1)

jmp  rb2_hi   ;jump only if RB2>RB1

...

snb  $03.2    ;test Z flag in STATUS reg
              ;(0 => RB2<RB1)

jmp  rb2_hi   ;jump only if RB2>RB1

...
```

The final “mov” instruction in this example performs an exchange of data between the working register (W) and the CMP_B register. This exchange occurs only with accesses to CMP_B and WKPEND_B. Otherwise, the “mov” instruction does not perform an exchange, but only moves data from the source to the destination.

The following figure shows the format of the CMP_B register.

CMP_B - Comparator Enable/Status Register

$\overline{\text{CMP\_EN}}$	$\overline{\text{CMP\_OE}}$	Reserved	CMP_RES
Bit 7	Bit 6	Bits 5–1	Bit 0

CMP_RES	Comparator result: 1 for RB2>RB1 or 0 for RB2<RB1. Comparator must be enabled (CMP_EN = 0) to read the result. The result can be read whether or not the CMP_OE bit is cleared.
$\overline{\text{CMP\_OE}}$	When cleared to 0, enables the comparator output to the RB0 pin.
$\overline{\text{CMP\_EN}}$	When cleared to 0, enables the comparator.

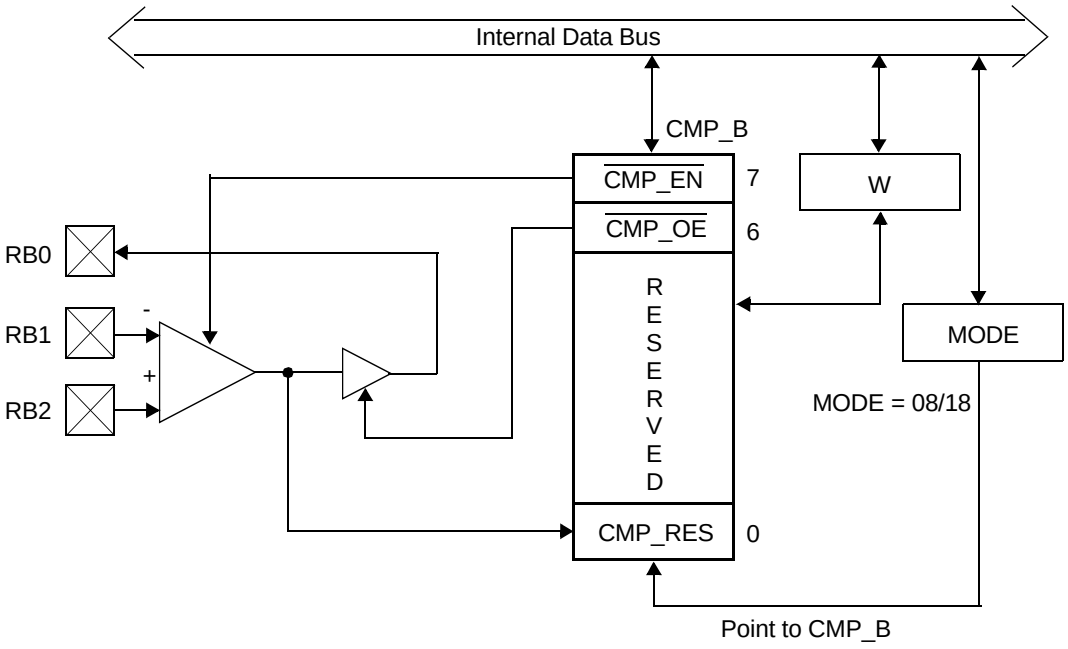
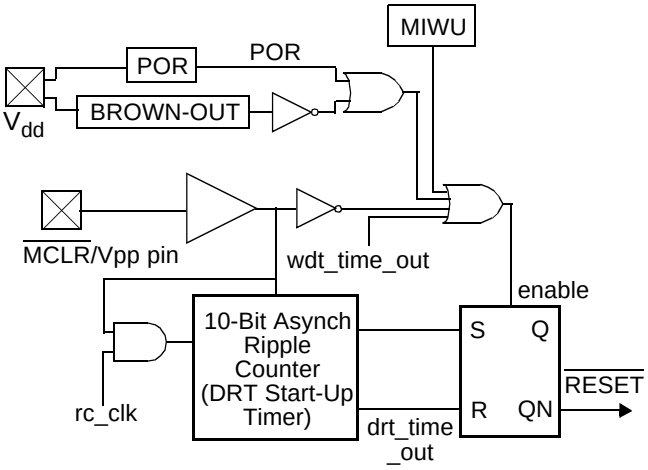


Figure 11-1. Comparator Block Diagram

12.0 RESET

Power-On-Reset, Brown-Out reset, watchdog reset, wakeup reset, or external reset initializes the device. Each one of these reset conditions causes the program counter to branch to the top of the program memory (FFFh).

The device incorporates an on-chip Power-On Reset (POR) circuit that generates an internal reset as V_{dd} rises during power-up. Figure 12-1 shows the block diagram of the circuit. The circuit contains an 10-bit Delay Reset Timer (DRT) and a reset latch. The DRT controls the reset timeout delay. The reset latch controls the internal reset signal. Upon power-up, the reset latch is set (device held in reset), and the DRT starts counting once it detects a valid logic high signal at the MCLR pin. Once DRT reaches the end of the timeout period (typically 72 msec), the reset latch is cleared, releasing the device from reset state.



Note:Ripple counter is 10 bits for Power on Reset (POR) only.

Figure 12-1. Block Diagram of On-Chip Reset Circuit

Figure 12-2 shows a power-up sequence where MCLR is not tied to the V_{dd} pin and V_{dd} signal is allowed to rise and stabilize before MCLR pin is brought high. The device will actually come out of reset T_{drt} msec after MCLR goes high.

The brown-out circuitry resets the chip when device power (V_{dd}) dips below its minimum allowed value, but not to zero, and then recovers to the normal value.

Figure 12-3 shows the on-chip Power-On Reset sequence where the MCLR and V_{dd} pins are tied together. The V_{dd} signal is stable before the DRT time-out period expires. In this case, the device will receive a proper reset. However, Figure 12-4 depicts a situation where V_{dd} rises too slowly. In this scenario, the DRT will time-out prior to V_{dd} reaching a valid operating voltage level (V_{dd} min). This means the device will come out of reset and start operating with the supply voltage not at a valid level. In this situation, it is recommended that you use the external RC circuit shown in Figure 12-5. The RC delay should exceed the time period it takes V_{dd} to reach a valid operating voltage.

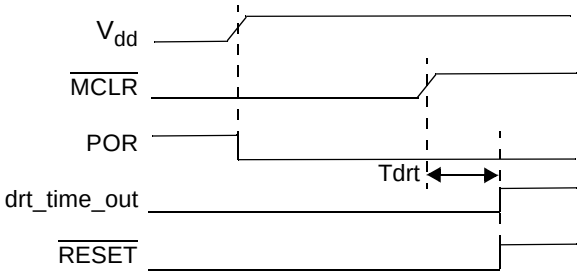


Figure 12-2. Time-Out Sequence on Power-Up (MCLR not tied to V_{dd})

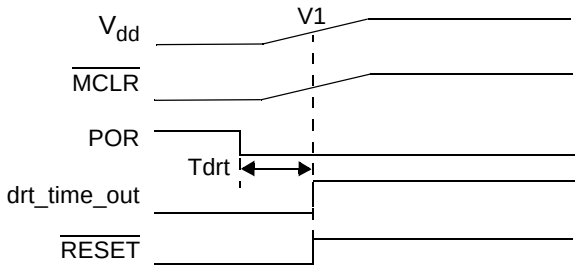


Figure 12-3. Time-out Sequence on Power-up (MCLR tied to V_{dd}): Slow Rise Time

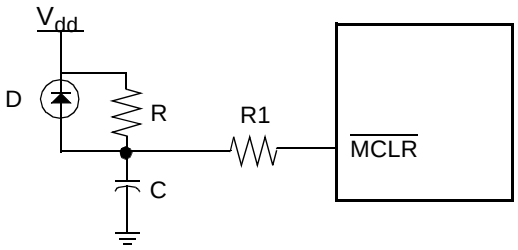


Figure 12-4. External Power-On Reset Circuit (For Slow V_{dd} Power-up)

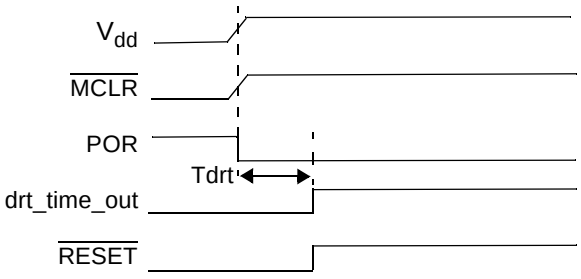


Figure 12-5. Time-out Sequence on Power-up (MCLR tied to V_{dd}): Fast V_{dd} Rise Time

A 3-bit field in the FUSX register can be used to specify the Delay Reset Timer (DRT) timeout period that results in an automatic wake-up from the sleep mode:

100 = 0 msec (no delay)
101 = 0.06 msec
110 = 7.68 msec
111 = 18.4 msec (default)
000 = 60 msec
001 = 480 msec
010 = 960 msec
011 = 1920 msec

For fast start-up from the sleep mode, clear the SLEEPCLK bit and set the WDRT2:WDRT0 field to 100. This will keep the clock operating during the sleep mode and allow a zero start-up delay.

Note 1: The external Power-On Reset circuit is required only if V_{dd} power-up is too slow. The diode D helps discharge the capacitor quickly when V_{dd} powers down.

Note 2: R < 40 kΩ is recommended to make sure that voltage drop across R does not violate the device electrical specifications.

R1 = 100Ω to 1kΩ will limit any current flowing into $\overline{\text{MCLR}}$ from external capacitor C. This helps prevent $\overline{\text{MCLR}}$ pin breakdown due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS).

13.0 REGISTER STATES UPON RESET

The effect of a reset operation on a register depends on the register and the type of reset operation. Some registers are initialized to specific values, some are left unchanged (for wakeup and brown-out resets), and some are initialized to an unknown value. A register that starts with an unknown value should be initialized by the software to a known value; you cannot simply test the initial state and rely on it starting in that state consistently.

Table 13-1 lists the SX registers and shows the state of each register upon reset, with a different column for each type of reset.

Table 13-1. Register States Upon Different Resets

Register	Power-On	Wakeup	Brown-out	Watchdog Timeout	MCLR
W	Undefined	Unchanged	Undefined	Unchanged	Unchanged
OPTION	FFh	FFh	FFh	FFh	FFh
MODE	1Fh	1Fh	1Fh	1Fh	1Fh
RTCC (01h)	Undefined	Unchanged	Undefined	Unchanged	Unchanged
PC (02h)	FFh	FFh	FFh	FFh	FFh
STATUS (03h)	Bits 0-2: Undefined Bits 3-4: 11 Bits 5-7: 000	Bits 0-2: Undefined Bits 3-4: Unch. Bits 5-7: 000	Bits 0-4: Undefined Bits 5-7: 000	Bits 0-2: Undefined Bits 3-4: (Note 1) Bits 5-7: 000	Bits 0-2: Undefined Bits 3-4: (Note 2) Bits 5-7:000
FSR (04h)	Undefined	Bits 0-6: Undefined Bit 7: 1	Bits 0-6: Undefined Bit 7: 1	Bits 0-6: Undefined Bit 7: 1	Bits 0-6: Undefined Bit 7: 1
RA through RE Direction	FFh	FFh	FFh	FFh	FFh
RA through RE Data	Undefined	Unchanged	Undefined	Unchanged	Unchanged
Other File Registers - SRAM	Undefined	Unchanged	Undefined	Unchanged	Unchanged
CMP_B	Bits 0, 6-7: 1 Bits 1-5: Undefined	Bits 0, 6-7: 1 Bits 1-5: Undefined	Bits 0, 6-7: 1 Bits 1-5: Undefined	Bits 0, 6-7: 1 Bits 1-5: Undefined	Bits 0, 6-7: 1 Bits 1-5: Undefined
WKPND_B	Undefined	Unchanged	Undefined	Unchanged	Unchanged
WKED_B	FFh	FFh	FFh	FFh	FFh
WKEN_B	FFh	FFh	FFh	FFh	FFh
ST_B through ST_E	FFh	FFh	FFh	FFh	FFh
LVL_A through LVL_E	FFh	FFh	FFh	FFh	FFh
PLP_A through PLP_E	FFh	FFh	FFh	FFh	FFh
Watchdog Counter	Undefined	Unchanged	Undefined	Unchanged	Unchanged

NOTE: 1. Watchdog reset during power down mode: 00 (bits TO, PD)
Watchdog reset during Active mode: 01 (bits TO, PD)

NOTE: 2. External reset during power down mode: 10 (bits TO, PD)
External reset during Active mode: Unchanged (bits TO, PD)

14.0 INSTRUCTION SET

As mentioned earlier, the SX family of devices uses a modified Harvard architecture with memory-mapped input/output. The device also has a RISC type architecture in that there are 43 single-word basic instructions. The instruction set contains byte-oriented file register, bit-oriented file register, and literal/control instructions.

Working register W is one of the CPU registers, which serves as a pseudo accumulator. It is a pseudo accumulator in a sense that it holds the second operand, receives the literal in the immediate type instructions, and also can be program-selected as the destination register. The bank of 31 file registers can also serve as the primary accumulators, but they represent the first operand and may be program-selected as the destination registers.

14.1 Instruction Set Features

1. All single-word (12-bit) instructions for compact code efficiency.
2. All instructions are single cycle except the jump type instructions (JMP, CALL) and failed test instructions (DECSZ fr, INCSZ fr, SB bit, SNB bit), which are two-cycle.
3. A set of file registers can be addressed directly or indirectly, and serve as accumulators to provide first operand; W register provides the second operand.
4. Many instructions include a destination bit which selects either the register file or the accumulator as the destination for the result.
5. Bit manipulation instructions (Set, Clear, Test and Skip if Set, Test and Skip if Clear).
6. STATUS Word register memory-mapped as a register file, allowing testing of status bits (carry, digit carry, zero, power down, and timeout).
7. Program Counter (PC) memory-mapped as register file allows W to be used as offset register for indirect addressing of program memory.
8. Indirect addressing data pointer FSR (file select register) memory-mapped as a register file.
9. IREAD instruction allows reading the instruction from the program memory addressed by W and upper four bits of MODE register.
10. Eight-level, 12-bit push/pop hardware stack for subroutine linkage using the Call and Return instructions.
11. Six addressing modes provide great flexibility.

14.2 Instruction Execution

An instruction goes through a four-stage pipeline to be executed (Figure 14-1). The first instruction is fetched from the program memory on the first clock cycle. On the second clock cycle, the first instruction is decoded and the second instruction is fetched. On the third clock cycle, the first instruction is executed, the second instruction is decoded, and the third instruction is fetched. On the fourth clock cycle, the first instruction's results are written to its destination, the second instruction is executed, the third instruction is decoded, and the fourth instruction is fetched. Once the pipeline is full, instructions are executed at the rate of one per clock cycle.

Instructions that directly affect the contents of the program counter (such as jumps and calls) require that the pipeline be cleared and subsequently refilled. Therefore, these instructions take more than one clock cycle.

The instruction execution time is derived by dividing the oscillator frequency by either one (turbo mode) or four (non-turbo mode). The divide-by factor is selected through the FUSE Word register.

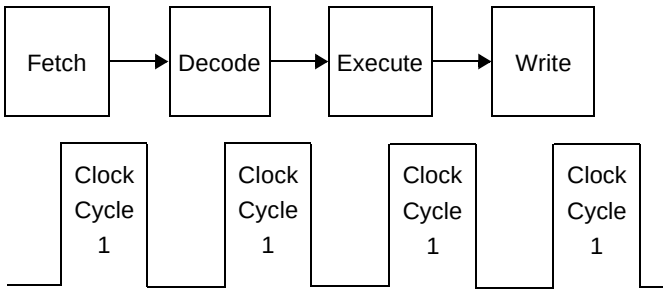


Figure 14-1. Pipeline and Clock Scheme

14.3 Addressing Modes

The device support the following addressing modes:

Data Direct
Data Indirect
Immediate
Program Direct
Program Indirect
Relative

Both direct and indirect addressing modes are available. The INDF register, though physically not implemented, is used in conjunction with the indirect data pointer (FSR) to perform indirect addressing. An instruction using INDF as its operand field actually performs the operation on the register pointed by the contents of the FSR. Consequently, processing two multiple-byte operands requires alternate loading of the operand addresses into the FSR pointer as the multiple byte data fields are processed.

Examples:

Direct addressing:

```
mov   RA,#$01          ;move "1" to RA
```

Indirect Addressing:

```
mov   FSR,#RA          ;FSR = address of RA
mov   INDF,#$01        ;move "1" to RA
```

14.4 The Bank Instruction

Often it is desirable to set the bank select bits of the FSR register in one instruction cycle. The Bank instruction provides this capability. This instruction sets the upper bits of the FSR to point to a specific RAM bank without affecting the other FSR bits, in preparation for using direct addressing.

Example:

```
bank  $F0       ;Select Bank 7 in FSR
inc   $1F       ;increment file register
                ;1Fh in Bank 7
```

14.5 Bit Manipulation

The instruction set contains instructions to set, reset, and test individual bits in data memory. The device is capable of bit addressing anywhere in data memory.

14.6 Input/Output Operation

The device contains three registers associated with each I/O port. The first register (Data Direction Register), configures each port pin as a Hi-Z input or output. The second register (TTL/CMOS Register), selects the desired input level for the input. The third register (Pull-Up Register), enables a weak pull-up resistor on the pin configured as a input. To read or write these registers, you must first write an appropriate value into the MODE register to select the desired register set, and then use the “mov !rx,W” instruction to read or write the register.

14.6.1 Read-Modify-Write Considerations

Caution must be exercised when performing successive SETB or CLRB operations on I/O port pin. Input data used for an instruction must be valid *during* the time the instruction is executed, and the output result from an instruction is valid only *after* that instruction completes its operation. Unexpected results from successive read-modify-write operations on I/O pins can occur when the device is running at extremely high speeds. Although the device has an internal write-back section to prevent such conditions, it is still recommended that the user program include a NOP instruction as a buffer between successive read-modify-write instructions performed on I/O pins of the same port.

In the default device configuration, when a read is performed from a port bit position, the operation is actually reading the voltage level on the pin itself, not necessarily the bit value stored in the port data register. This is true whether the pin is configured to operate as an input or an output. Therefore, with the pin configured to operate as an input, the data register contents have no effect on the value that you read. With the pin configured to operate as an output, what is read generally matches what has been written to the register. PORTRD of the T2CNT2 register determines how the device reads data from its I/O ports (Port A through Port E). Set this bit to 1 to have the device read data from the port I/O pins directly. Clear this bit to 0 to have the device read data from the port data registers. Under normal conditions, it should not matter which method you use to read the port data. However, if a port pin is configured as an output and an external circuit forces the pin to the opposite value, the value read from the port will depend on the reading mode used. Note that this control bit is not related to multi-function timers T1 and T2.

14.7 Increment/Decrement

The current selected bank of 31 registers serves as a set of accumulators. The instruction set contains instructions to increment and decrement the register file. The device also includes both INCSZ fr (increment file register and skip if zero) and DECSZ fr (decrement file register and skip if zero) instructions.

14.8 Loop Counting and Data Pointing Testing

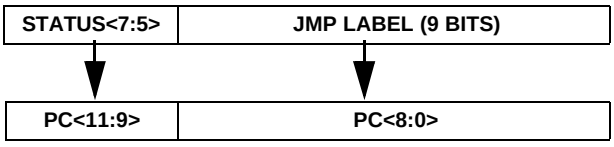
The device has specific instructions to facilitate loop counting. The DECSZ fr (decrement file register and skip if zero) tests any one of the file registers and skips the next instruction (which can be a branch back to loop) if the result is zero.

14.9 Branch and Loop Call Instructions

The device contains an 8-level hardware stack where the return address is stored with a subroutine call. Multiple stack levels allow subroutine nesting. The instruction set supports absolute address branching.

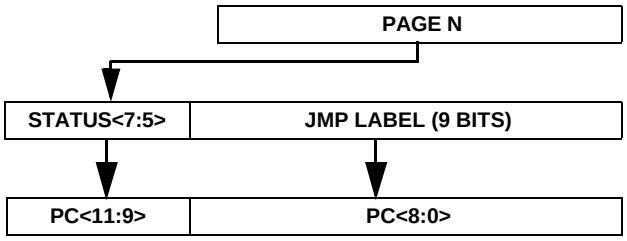
14.9.1 Jump Operation

When a JMP instruction is executed, the lower nine bits of the program counter are loaded with the address of the specified label. The upper three bits of the program counter are loaded with the page select bits, PA2:PA0, contained in the STATUS register. Therefore, care must be exercised to ensure the page select bits are pointing to the correct page *before* the jump occurs.



14.9.2 Page Jump Operation

When a JMP instruction is executed and the intended destination is on a different page, the page select bits must be initialized with appropriate values to point to the desired page before the jump occurs. This can be done easily with SETB and CLRB instructions or by writing a value to the STATUS register. The device also has the PAGE instruction, which automatically selects the page in a single-cycle execution.



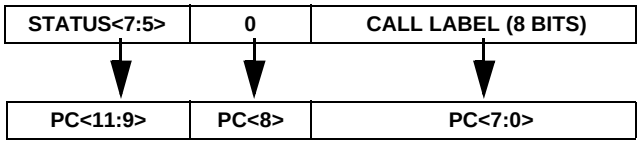
Note:“N” must be 0, 1, 2, or 3.

14.9.3 Call Operation

The following happens when a CALL instruction is executed:

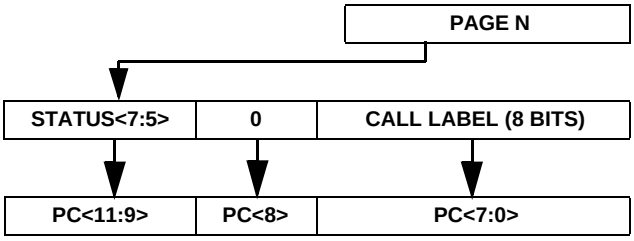
- The current value of the program counter is incremented and pushed onto the top of the stack.
- The lower eight bits of the label address are copied into the lower eight bits of the program counter.
- The ninth bit of the Program Counter is cleared to zero.
- The page select bits (in STATUS register) are copied into the upper three bits of the 12-bit program counter.

This means that the call destination must *start* in the lower half of any page. For example, 00h-0FFh, 200h-2FFh, 400h-4FFh, etc.



14.9.4 Page Call Operation

When a subroutine that resides on a different page is called, the page select bits must contain the proper values to point to the desired page before the call instruction is executed. This can be done easily using SETB and CLRB instructions or writing a value to the STATUS register. The device also has the PAGE instruction, which automatically selects the page in a single-cycle execution.



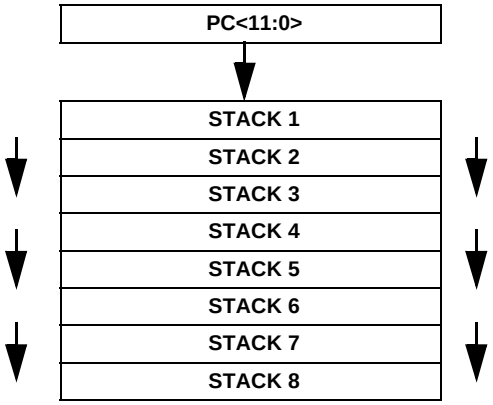
14.10 Return Instructions

The device has several instructions for returning from subroutines and interrupt service routines. The return from subroutine instructions are RET (return without affecting W), RETP (same as RET but affects PA2:PA0), RETI (return from interrupt), RETIW (return that affects W), and RETW #literal (return and place literal in W). The literal serves as an immediate data value from memory. This instruction can be used for table lookup operations. To do table lookup, the table must contain a string of RETW #literal instructions. The first instruction just in front of the table calculates the offset into the table. The table can be used as a result of a CALL.

14.11 Subroutine Operation

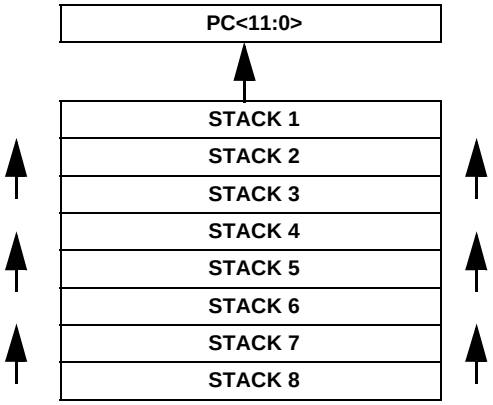
14.11.1 Push Operation

When a subroutine is called, the return address is pushed onto the subroutine stack. Specifically, each address in the stack is moved to the next lower level in order to make room for the new address to be stored. Stack 1 receives the contents of the program counter. Stack 8 is overwritten with what was in Stack 7. The contents of stack 8 are lost.



14.11.2 Pop Operation

When a return instruction is executed the subroutine stack is popped. Specifically, the contents of Stack 1 are copied into the program counter and the contents of each stack level are moved to the next higher level. For example, Stack 1 receives the contents of Stack 2, etc., until Stack 7 is overwritten with the contents of Stack 8. Stack 8 is left unchanged, so the contents of Stack 8 are duplicated in Stack 7.



14.12 Comparison and Conditional Branch Instructions

The instruction set includes instructions such as DECSZ fr (decrement file register and skip if zero), INCSZ fr (increment file register and skip if zero), SNB bit (bit test file register and skip if bit clear), and SB bit (bit test file register and skip if bit set). These instructions will cause the next instruction to be skipped if the tested condition is true. If a skip instruction is immediately followed by a PAGE or BANK instruction (and the tested condition is true) then two instructions are skipped and the operation consumes three cycles. This is useful for conditional branching to another page where a PAGE instruction precedes a JMP. If several PAGE and BANK instructions immediately follow a skip instruction then they are all skipped plus the next instruction and a cycle is consumed for each.

14.13 Logical Instruction

The instruction set contain a full complement of the logical instructions (AND, OR, Exclusive OR), with the W register and a selected memory location (using either direct or indirect addressing) serving as the two operands.

14.14 Shift and Rotate Instructions

The instruction set includes instructions for left or right rotate-through-carry.

14.15 Complement and SWAP

The device can perform one's complement operation on the file register (fr) and W register. The MOV W,<>fr instruction performs nibble-swap on the fr and puts the value into the W register.

14.16 Key to Abbreviations and Symbols

Symbol	Description
W	Working register
fr	File register (memory-mapped register in the range of 00h to FFh)
PC	Lower eight bits of program counter (file register 02h)
STATUS	STATUS register (file register 03h)
FSR	File Select Register (file register 04h)
C	Carry flag in STATUS register (bit 0)
DC	Digit Carry flag in STATUS register (bit 1)
Z	Zero flag in STATUS register (bit 2
PD	Power Down flag in STATUS register (bit 3)
TO	Watchdog Timeout flag in STATUS register (bit 4)
PA2:PA0	Page select bits in STATUS register (bits 7:5)
OPTION	OPTION register (not memory-mapped)
WDT	Watchdog Timer register (not memory-mapped)
MODE	MODE register (not memory-mapped)
rx	Port control register pointer (RA, RB, or RC)
!	Non-memory-mapped register designator
f	File register address bit in opcode
k	Constant value bit in opcode
n	Numerical value bit in opcode
b	Bit position selector bit in opcode
.	File register / bit selector separator in assembly language instruction
#	Immediate literal designator in assembly language instruction
lit	Literal value in assembly language instruction
addr8	8-bit address in assembly language instruction
addr9	9-bit address in assembly language instruction
addr12	12-bit address in assembly language instruction
/	Logical 1's complement
|	Logical OR
^	Logical exclusive OR
&	Logical AND
<>	Swap high and low nibbles (4-bit segments)
<<	Rotate left through carry flag
>>	Rotate right through carry flag
- -	Decrement file register
++	Increment file register

15.0 INSTRUCTION SET SUMMARY TABLE

Table 15-1 lists all of the instructions, organized by category. For each instruction, the table shows the instruction mnemonic (as written in assembly language), a brief description of what the instruction does, the number of instruction cycles required for execution, the binary opcode, and the status flags affected by the instruction.

The “Cycles” column typically shows a value of 1, which means that the overall throughput for the instruction is one per clock cycle. In some cases, the exact number of cycles depends on the outcome of the instruction (such as the test-and-skip instructions) or the clocking mode (Compatible or Turbo). In those cases, all possible numbers of cycles are shown in the table.

The instruction execution time is derived by dividing the oscillator frequency by either one (Turbo mode) or four (Compatible mode). The divide-by factor is selected through the FUSE Word register.

Table 15-1. The SX Instruction Set

Mnemonic, Operands	Description	Cycles (Compatible)	Cycles (Turbo)	Opcode	Flags Affected
Logical Operations					
AND fr, W	AND of fr and W into fr (fr = fr & W)	1	1	0001 011f ffff	Z
AND W, fr	AND of W and fr into W (W = W & fr)	1	1	0001 010f ffff	Z
AND W,#lit	AND of W and Literal into W (W = W & lit)	1	1	1110 kkkk kkkk	Z
NOT fr	Complement of fr into fr (fr = fr ^ FFh)	1	1	0010 011f ffff	Z
OR fr,W	OR of fr and W into fr (fr = fr | W)	1	1	0001 001f ffff	Z
OR W,fr	OR of W and fr into fr (W = W | fr)	1	1	0001 000f ffff	Z
OR W,#lit	OR of W and Literal into W (W = W | lit)	1	1	1101 kkkk kkkk	Z
XOR fr,W	XOR of fr and W into fr (fr = fr ^ W)	1	1	0001 101f ffff	Z
XOR W,fr	XOR of W and fr into W (W = W ^ fr)	1	1	0001 100f ffff	Z
XOR W,#lit	XOR of W and Literal into W (W = W ^ lit)	1	1	1111 kkkk kkkk	Z
Arithmetic and Shift Operations					
ADD fr,W	Add W to fr (fr = fr + W); carry flag is added if $\overline{CF}$ bit in FUSEX register is cleared to 0	1	1	0001 111f ffff	C, DC, Z
ADD W,fr	Add fr to W (W = W + fr); carry flag is added if $\overline{CF}$ bit in FUSEX register is cleared to 0	1	1	0001 110f ffff	C, DC, Z
CLR fr	Clear fr (fr = 0)	1	1	0000 011f ffff	Z
CLR W	Clear W (W = 0)	1	1	0000 0100 0000	Z
CLR !WDT	Clear Watchdog Timer, clear prescaler if assigned to the Watchdog (TO = 1, PD = 1)	1	1	0000 0000 0100	TO, PD
DEC fr	Decrement fr (fr = fr - 1)	1	1	0000 111f ffff	Z
DECSZ fr	Decrement fr and Skip if Zero (fr = fr - 1 and skip next instruction if result is zero)	1 or 2 (skip)	1 or 2 (skip)	0010 111f ffff	none
INC fr	Increment fr (fr = fr + 1)	1	1	0010 101f ffff	Z
INCSZ fr	Increment fr and Skip if Zero (fr = fr + 1 and skip next instruction if result is zero)	1 or 2 (skip)	1 or 2 (skip)	0011 111f ffff	none
RL fr	Rotate fr Left through Carry (fr = << fr)	1	1	0011 011f ffff	C
RR fr	Rotate fr Right through Carry (fr = >> fr)	1	1	0011 001f ffff	C
SUB fr,W	Subtract W from fr (fr = fr - W); complement of the carry flag is subtracted if $\overline{CF}$ bit in FUSEX register is cleared to 0	1	1	0000 101f ffff	C, DC, Z
SWAP fr	Swap High/Low Nibbles of fr (fr = <> fr)	1	1	0011 101f ffff	none

Table 15-1. The SX Instruction Set (Continued)

Mnemonic, Operands	Description	Cycles (Compatible)	Cycles (Turbo)	Opcode	Flags Affected
Bitwise Operations					
CLRB fr.bit	Clear Bit in fr (fr.bit = 0)	1	1	0100 bbbf ffff	none
SB fr.bit	Test Bit in fr and Skip if Set (test fr.bit and skip next instruction if bit is 1)	1 or 2 (skip)	1 or 2 (skip)	0111 bbbf ffff	none
SETB fr.bit	Set Bit in fr (fr.bit = 1)	1	1	0101 bbbf ffff	none
SNB fr.bit	Test Bit in fr and Skip if Clear (test fr.bit and skip next instruction if bit is 0)	1 or 2 (skip)	1 or 2 (skip)	0110 bbbf ffff	none
Data Movement Instructions					
MOV fr,W	Move W to fr (fr = W)	1	1	0000 001f ffff	none
MOV W,fr	Move fr to W (W = fr)	1	1	0010 000f ffff	Z
MOV W,fr-W	Move (fr-W) to W (W = fr - W); complement of carry flag is subtracted if CF bit in FUSEX register is cleared to 0	1	1	0000 100f ffff	C, DC, Z
MOV W,#lit	Move Literal to W (W = lit)	1	1	1100 kkkk kkkk	none
MOV W,/fr	Move Complement of fr to W (W = fr ^ FFh)	1	1	0010 010f ffff	Z
MOV W,--fr	Move (fr-1) to W (W = fr - 1)	1	1	0000 110f ffff	Z
MOV W,++fr	Move (fr+1) to W (W = fr + 1)	1	1	0010 100f ffff	Z
MOV W,<<fr	Rotate fr Left through Carry and Move to W (W = << fr)	1	1	0011 010f ffff	C
MOV W,>>fr	Rotate fr Right through Carry and Move to W (W = >> fr)	1	1	0011 000f ffff	C
MOV W,<>fr	Swap High/Low Nibbles of fr and move to W (W = <> fr)	1	1	0011 100f ffff	none
MOV W,M	Move MODE Register to W (W = MODE), high nibble cleared	1	1	0000 0100 0010	none
MOVSZ W,--fr	Move (fr-1) to W and Skip if Zero (W = fr -1 and skip next instruction if result is zero)	1 or 2 (skip)	1 2 (skip)	0010 110f ffff	none
MOVSZ W,++fr	Move (fr+1) to W and Skip if Zero (W = fr + 1 and skip next instruction if result is zero)	1 or 2 (skip)	1 2 (skip)	0011 110f ffff	none
MOV M,W	Move W to MODE Register (MODE = W)	1	1	0000 0100 0011	none
MOV M,#lit	Move Literal to MODE Register (MODE = lit)	1	1	0000 0101 kkkk	none
MOV !rx,W	Move Data Between W and Control Register: rx = W (move W to rx) *or* W = rx (move rx to W) *or* rx <=> W (exchange W and rx)	1	1	0000 0000 0fff	none
MOV !OPTION, W	Move W to OPTION Register (OPTION = W)	1	1	0000 0000 0010	none
TEST fr	Test fr for Zero (fr = fr to set or clear Z flag)	1	1	0010 001f ffff	Z

Table 15-1. The SX Instruction Set (Continued)

Mnemonic, Operands	Description	Cycles (Compatible)	Cycles (Turbo)	Opcode	Flags Affected
Program Control instruction					
CALL addr8	Call Subroutine: top-of-stack = program counter + 1 PC(7:0) = addr8 program counter (8) = 0 program counter (11:9) = PA2:PA0	2	3	1001 kkkk kkkk	none
JMP addr9	Jump to Address: PC(7:0) = addr9(7:0) program counter (8) = addr9(8) program counter (11:9) = PA2:PA0	2	3	101k kkkk kkkk	none
NOP	No Operation	1	1	0000 0000 0000	none
RET	Return from Subroutine (program counter = top-of-stack)	2	3	0000 0000 1100	none
RETP	Return from Subroutine Across Page Boundary (PA2:PA0 = top-of-stack (11:9) and program counter = top-of-stack)	2	3	0000 0000 1101	PA1, PA0
RETI	Return from Interrupt (restore W, STATUS, FSR, and program counter from shadow registers)	2	3	0000 0000 1110	all STATUS
RETIW	Return from Interrupt and Adjust RTCC with W (restore W, STATUS, FSR, and program counter from shadow registers; and add W to the RTCC register)	2	3	0000 0000 1111	all STATUS
RETW lit	Return from Subroutine with Literal in W (W = lit and program counter = top-of-stack)	2	3	1000 kkkk kkkk	none
System Control Instructions					
BANK addr8	Load Bank Number into FSR(7:5) FSR(7:5) = addr8(7:5)	1	1	0000 0001 1nnn	none
IREAD	Read Word from Instruction Memory MODE:W = data at (MODE:W)	1	4	0000 0100 0001	none
PAGE addr12	Load Page Number into STATUS(7:5) STATUS(7:5) = addr12(11:9)	1	1	0000 0001 0nnn	PA2, PA1, PA0
SLEEP	Power Down Mode WDT = 00h, TO = 1, PD = 1, stop oscillator (PD = 0, clears prescaler if assigned)	1	1	0000 0000 0011	TO, PD

15.1 Equivalent Assembler Mnemonics

Some assemblers support additional instruction mnemonics that are special cases of existing instructions or alternative mnemonics for standard ones. For example, an assembler might support the mnemonic “CLC” (clear carry), which is interpreted the same as the instruction “clrb $03.0” (clear bit 0 in the STATUS register). Some of the commonly supported equivalent assembler mnemonics are described in Table 15-2.

Table 15-2. Equivalent Assembler Mnemonics

Syntax	Description	Equivalent	Cycles
CLC	Clear Carry Flag	CLRB $03.0	1
CLZ	Clear Zero Flag	CLRB $03.2	1
JMP W	Jump Indirect W	MOV $02,W	4 or 3 (note 1)
JMP PC+W	Jump Indirect W Relative	ADD $02,W	4 or 3 (note 1)
MODE imm4	Move Immediate to MODE Register	MOV M,#lit	1
NOT W	Complement W	XOR W,#$FF	1
SC	Skip if Carry Flag Set	SB $03.0	1 or 2 (note 2)
SKIP	Skip Next Instruction	SNB $02.0 or SB $02.0	4 or 2 (note 3)

Note 1: The JMP W or JMP PC+W instruction takes 4 cycles in the “compatible” clocking mode or 3 cycles in the “turbo” clocking mode.

Note 2: The SC instruction takes 1 cycle if the tested condition is false or 2 cycles if the tested condition is true.

Note 3: The assembler converts the SKIP instruction into a SNB or SB instruction that tests the least significant bit of the program counter, choosing SNB or SB so that the tested condition is always true. The instruction takes 4 cycles in the “compatible” clocking mode or 2 cycles in the “turbo” clocking mode.

16.0 ELECTRICAL CHARACTERISTICS

16.1 Absolute Maximum Ratings

Ambient temperature under bias	-40° C to +85° C (to +125° C at 4V to 6.25V)
Storage temperature	-65° C to +150° C
Voltage on V_{dd} with respect to V_{ss}	0 V to +7.5V
Voltage on OSC1 with respect to V_{ss}	0 V to +12.5V
Voltage on $\overline{MCLR}$ with respect to V_{ss}	0 V to +14V
Voltage on all other pins with respect to V_{ss}	0.6 V to (V_{dd} + 0.6V)V
Total power dissipation	TBD mW
Max. current out of V_{ss} pin	100mA
Max. current into V_{dd} pin	100mA
Max. DC current into an input pin (with internal protection diode forward biased)	±500µA
Input clamp current, I_{ik} (V_i < 0 or V_i > V_{dd})	±20mA
Output clamp current, I_{ok} (V_O < 0 or V_O > V_{dd})	±20mA
Max. allowable sink current per I/O pin	45mA
Max. allowable source current per I/O pin	45 mA
Max. allowable sink current per I/O port (PORT A, B, or C)	TBD
Max. allowable source current per I/O port (PORT A, B, or C)	TBD

16.2 DC Characteristics

Operating Temperature 0° C <= Ta <= +70° C (Commercial)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{dd}	Supply Voltage	RC XT HS LP	3.3 3.3 3.3 3.3	–	5.5 5.5 5.5 5.5	V V V V
V_{por}	V_{dd} start voltage to ensure Power-On Reset	–	V_{ss}	–	–	V
S_{Vdd}	V_{dd} rise rate	–	0.05	–	–	V/ms
I_{dd}	Supply Current, active	V_{dd} = 5.0V, F_{osc} = 50 MHz V_{dd} = 5.0V, F_{osc} = 4 MHz internal V_{dd} = 3.3V, F_{osc} = 20 MHz	- 	80 TBD 18	–	mA mA mA
I_{pd}	Supply Current, power down	V_{dd} = 4.5V, WDT enabled V_{dd} = 4.5V, WDT disabled	-	TBD 1	–	µA µA
V_{ih}, V_{il}	Input Levels $\overline{MCLR}$, OSC1, RTCC Logic High Logic Low All Other Inputs CMOS Logic High Logic Low TTL Logic High Logic Low		 $0.8V_{dd}$ V_{ss} $0.7V_{dd}$ V_{ss} 2.0 V_{ss}		 V_{dd} $0.2V_{dd}$ V_{dd} $0.3V_{dd}$ V_{dd} 0.8	 V V V V V V
I_{il}	Input Leakage Current	V_{in} = V_{dd} or V_{ss}	-1.0		+1.0	µA
I_{pup}	Weak Pullup Current	V_{dd} = 5.5V, V_{in} = 0V V_{dd} = 3.3V, V_{in} = 0V			400 180	µA µA
V_{oh}	Output High Voltage OSC2, Ports B, C Port A	 Ioh = 20mA, Vdd = 4.5V Ioh = 12mA, Vdd = 3.3V Ioh = 30mA, Vdd = 4.5V Ioh = 20mA, Vdd = 3.3V	 Vdd-0.7 Vdd-0.7 Vdd-0.7 Vdd-0.7			 V V V V
V_{ol}	Output Low Voltage OSC2, All Ports	I_{ol} = 30mA, Vdd = 4.5V Iol = 20mA, Vdd = 3.3V			0.6 0.6	V V

16.3 AC Characteristics

Operating Temperature 0° C <= Ta <= +70° C (Commercial)

Symbol	Parameter	Min	Typ	Max	Units	Conditions
F_{osc}	External CLKIN Frequency	DC	–	4.0	MHz	RC
				4.0	MHz	XT
				50	MHz	HS
				200	KHz	LP
	Oscillator Frequency	DC	–	4.0	MHz	RC
		0.1		4.0	MHz	XT
		4		50	MHz	HS
		5		200	KHz	LP
T_{osc}	External CLKIN Period	250	–	–	ns	RC
		250			ns	XT
		20			ns	HS
		5.0			µs	LP
	Oscillator Period	250	–	–	ns	RC
		250		10,000	ns	XT
		20		250	ns	HS
		5.0		200	µs	LP
T_{osL}, T_{osH}	Clock in (OSC1) Low or High Time	50	–	–	ns	XT
		8.0			ns	HS
		2.0			µs	LP
T_{osR}, T_{osF}	Clock in (OSC1) Rise or Fall Time	–	–	25	ns	XT
				25	ns	HS
				50	µs	LP

Note:Data in the Typical (“TYP”) column is at 5V, 25° C unless otherwise stated.

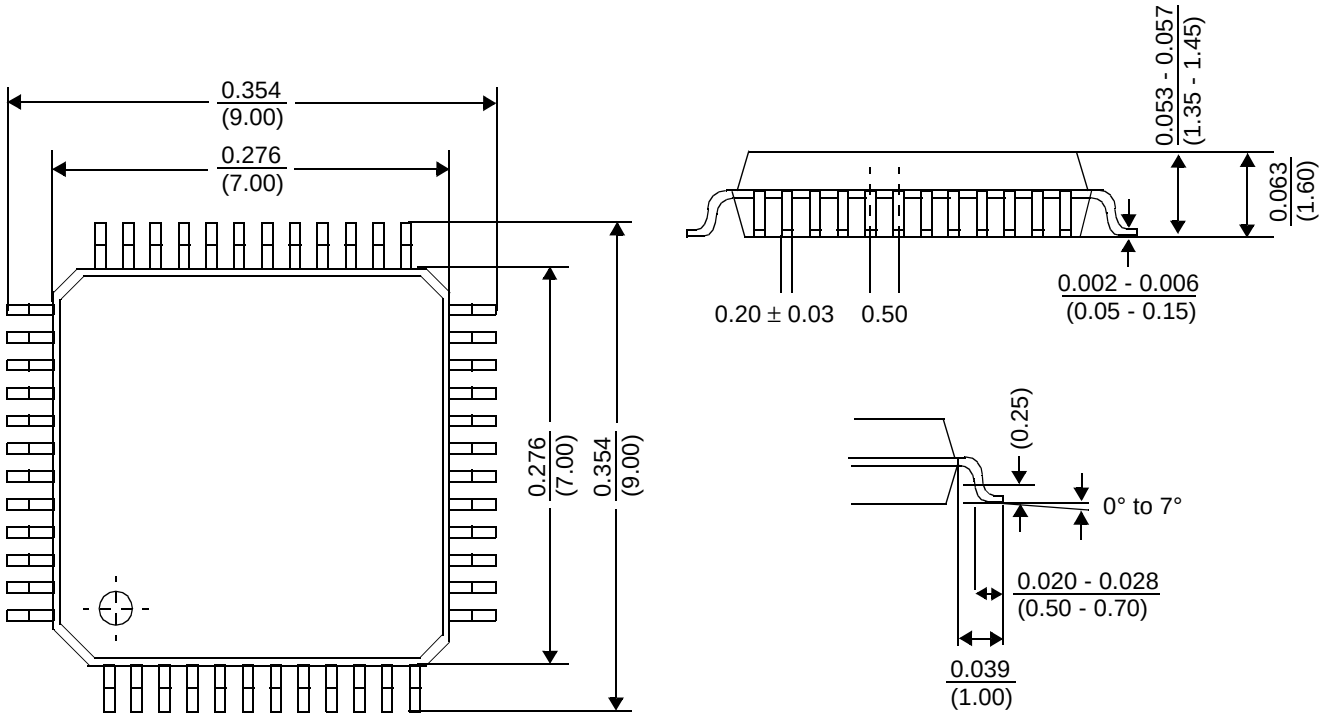
Internal RC Oscillator AC specs are still being characterized. Specification is 4MHz ± 8% over commercial temp (0° C-70° C) range.

16.4 Comparator DC and AC Specifications

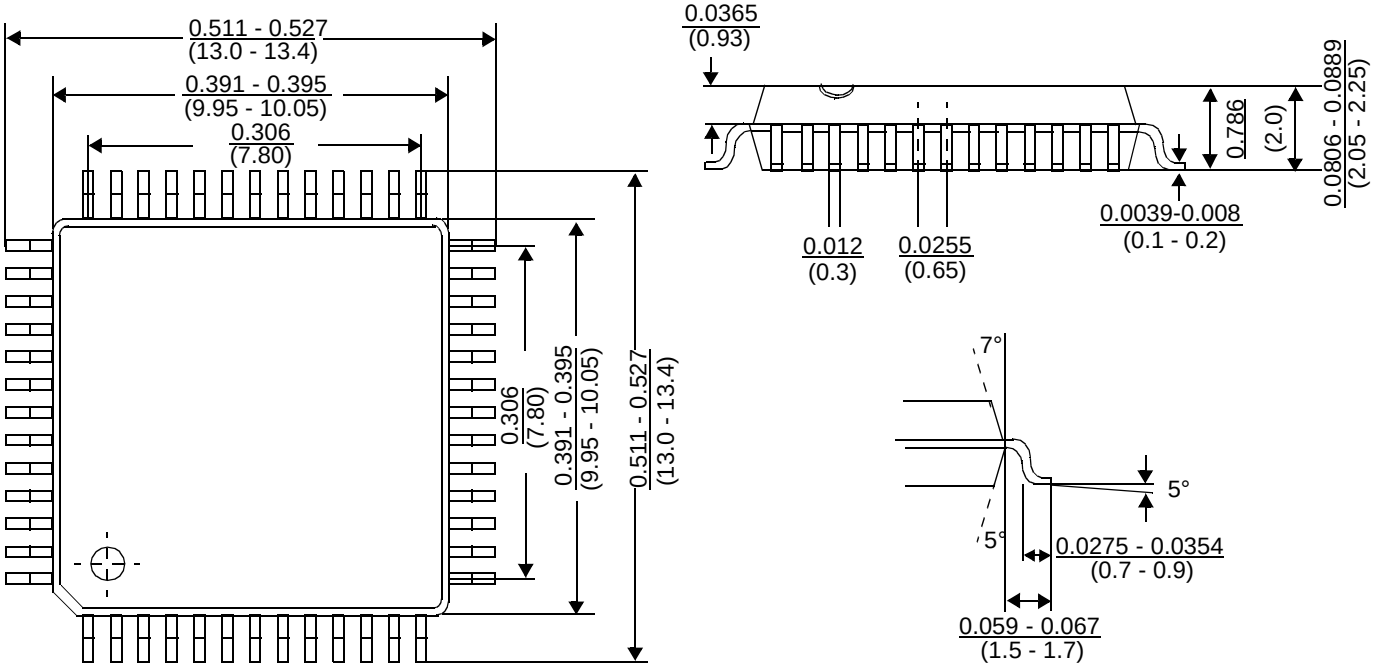
Parameter	Conditions	Min	Typ	Max	Units
Input Offset Voltage	0.4V < Vin < Vdd – 1.5V		+/- 10	+/- 25	mV
Input Common Mode Voltage Range		0.4		Vdd – 1.3	V
Voltage Gain			300k		V/V
DC Supply Current (enabled)	Vdd = 5.5V			120	µA
Response Time	$V_{overdrive}$ = 25mV			250	ns

17.0 PACKAGE DIMENSIONS [DIMENSIONS ARE IN INCHES/(MILLIMETERS)]

SX48TQFP



SX52PQFP



Sales and Tech Support Contact Information

For the latest contact and support information on SX devices, please visit the Scenix Semiconductor website at www.scenix.com. The site contains technical literature, local sales contacts, tech support and many other features.

Scenix Semiconductor, Inc.

3160 De La Cruz Blvd., Suite #200,
Santa Clara, CA 95054
Contact: sales@scenix.com
http://www.scenix.com
Tel.: (408) 327-8888
Fax: (408) 327-8880