Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

March, 2002

SX20AC/SX28AC

Configurable Communications Controllers with EE/Flash Program Memory, In-System Programming Capability and On-Chip Debug

1.0 PRODUCT OVERVIEW

1.1 Introduction

The Ubicom SX family of configurable communications controllers are 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 75 MHz and by optimizing the instruction set to include mostly single-cycle instructions. In addition, the SX architecture is deterministic and totally reprogramable. The unique combination of these characteristics enables the device to implement hard real-time functions as software modules (Virtual Peripheral™) to replace traditional hardware functions.

On-chip functions include 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.

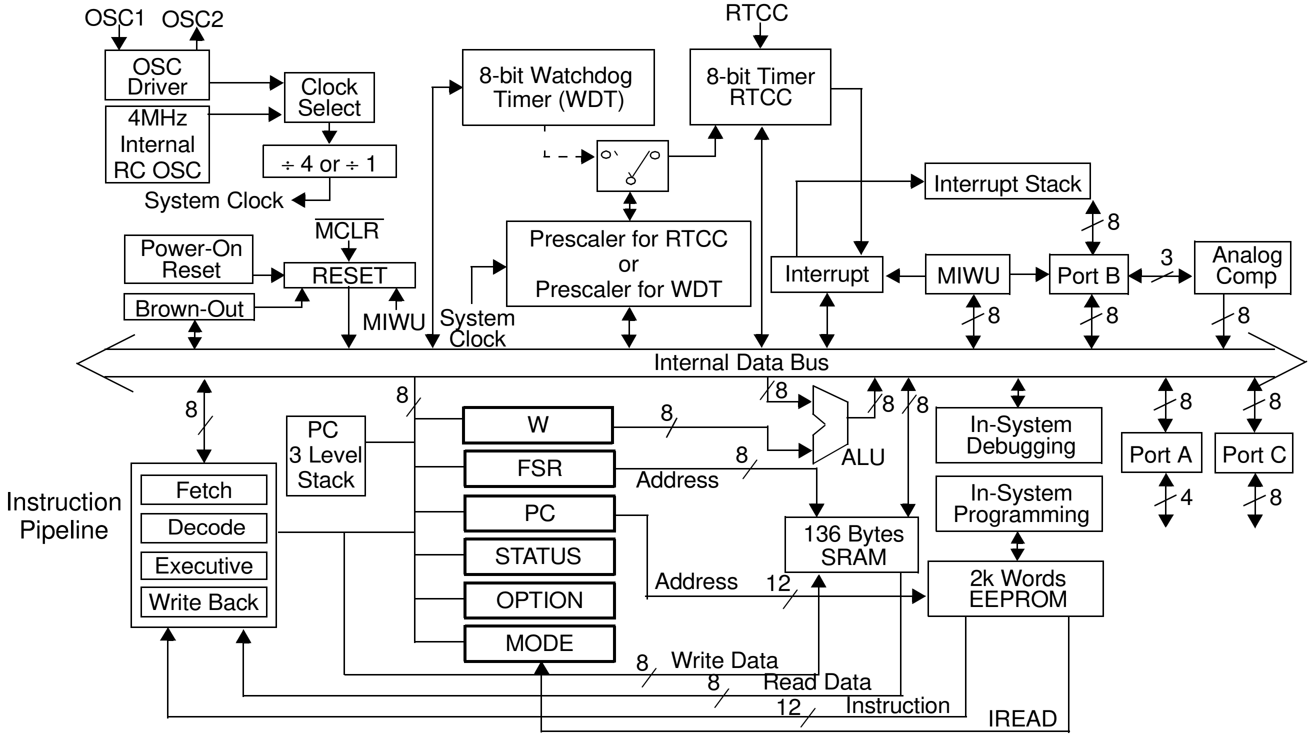


Figure 1-1. Block Diagram

Table of Contents

1.2 Key Features

50 MIPS Performance

- SX20AC/SX28AC: DC - 75 MHz
- SX20AC/SX28AC: 13.3 ns instruction cycle, 39.9 ns internal interrupt response
- 1 instruction per clock (branches 3)

EE/FLASH Program Memory and SRAM Data Memory

- Access time of < 13.3 ns provides single cycle access
- EE/Flash rated for > 10,000 rewrite cycles
- 2048 Words EE/Flash program memory
- 136x8 bits SRAM data memory

CPU Features

- Compact instruction set
- All instructions are single cycle except branch
- Eight-level push/pop hardware stack for subroutine linkage
- Fast table lookup capability through run-time readable code (IREAD instruction)
- Totally predictable program execution flow for hard real-time applications

Fast and Deterministic Interrupt

- Jitter-free 3-cycle internal interrupt response
- Hardware context save/restore of key resources such as PC, W, STATUS, and FSR within the 3-cycle interrupt response time
- External wakeup/interrupt capability on Port B (8 pins)

Flexible I/O

- All pins individually programmable as I/O
- Inputs are TTL or CMOS level selectable
- All pins have selectable internal pull-ups
- Selectable Schmitt Trigger inputs on Ports B, and C
- All outputs capable of sourcing/sinking 30 mA
- Port A outputs have symmetrical drive
- Analog comparator support on Port B (RB0 OUT, RB1 IN-, RB2 IN+)
- Selectable I/O operation synchronous to the oscillator clock

Hardware Peripheral Features

- One 8-bit Real Time Clock/Counter (RTCC) with programable 8-bit prescaler
- Watchdog Timer (shares the RTCC prescaler)
- Analog comparator
- Brown-out detector
- Multi-Input Wakeup logic on 8 pins
- Internal RC oscillator with configurable rate from 31.25 kHz to 4 MHz
- Power-On-Reset

Packages

- 20-pin SSOP, 28-pin DIP/SSOP

Programming and Debugging Support

- On- chip in-system programming support with serial and parallel interfaces
- In-system serial programming via oscillator pins
- On-chip in-System debugging support logic
- Real-time emulation, full program debug, and integrated development environment offered by third party tool vendors

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.

Ubicom has developed a revolutionary RISC-based architecture and memory design techniques that is 20 times faster than conventional MCUs, deterministic, jitter free, and totally reprogramable.

The SX family implements a four-stage pipeline (fetch, decode, execute, and write back), which results in execution of one instruction per clock cycle. For example, at the maximum operating frequency of 75 MHz, instructions are executed at the rate of one per 13.3ns clock cycle.

1.3.1 The Virtual Peripheral Concept

Virtual Peripheral concept enables the “software system on a chip” approach. Virtual Peripheral, a software module that replaces a traditional hardware peripheral, takes advantage of the Ubicom architecture’s high performance and deterministic nature to produce same results as the hardware peripheral with much greater flexibility.

The speed and flexibility of the Ubicom architecture complemented with the availability of the Virtual Peripheral library, simultaneously address a wide range of engineering and product development concerns. They decrease the product development cycle dramatically, shortening time to production to as little as a few days.

Ubicom’s time-saving Virtual Peripheral library gives the system designers a choice of ready-made solutions, or a head start on developing their own peripherals. So, with Virtual Peripheral modules handling established functions, design engineers can concentrate on adding value to other areas of the application.

The concept of Virtual Peripheral combined with in-system re-programmability provides a power development platform ideal for the communications industry because of the numerous and rapidly evolving standards and protocols.

Overall, the concept of Virtual Peripheral provides benefits such as using a more simple device, reduced component count, fast time to market, increased flexibility in design, customization to your application, and ultimately overall system cost reduction.

Some examples of Virtual Peripheral modules are:

- Communication interfaces such as I^{2}C™, Microwire™ (µ-Wire), SPI, IrDA Stack, UART, and Modem functions
- Frequency generation and measurement
- PPM/PWM output
- Delta/Sigma ADC
- DTMF generation/detection
- PSK/FSK generation/detection
- FFT/DFT based algorithms

1.3.2 The Communications Controller

The combination of the Ubicom hardware architecture and the Virtual Peripheral concept create a powerful, creative platform for the communications design communities: SX communications controller. Its high processing power, recofigurability, cost-effectiveness, and overall design freedom give the designer the power to build products for the future with the confidence of knowing that they can keep up with innovation in standards and other areas.

1.4 Programming and Debugging Support

The SX devices are currently supported by third party tool vendors. On-chip in-system debug capabilities have been added, allowing tools to provide an integrated development environment including editor, macro assembler, debugger, and programmer. Un-obtrusive in-system programming is provided through the OSC pins. There is no need for a bon-out chip, so the user does not have to worry about the potential variations in electrical characteristics of a bond-out chip and the actual chip used in the target applications. the user can test and revise the fully debugged code in the actual SX, in the actual application, and get to production much faster.

1.5 Applications

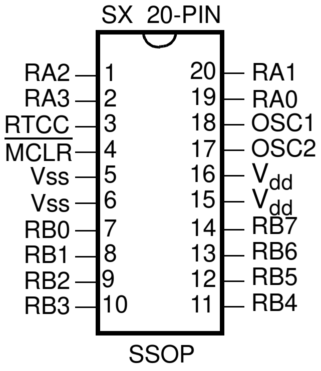
Emerging applications and advances in existing ones require higher performance while maintaining low cost and fast time-to-production.

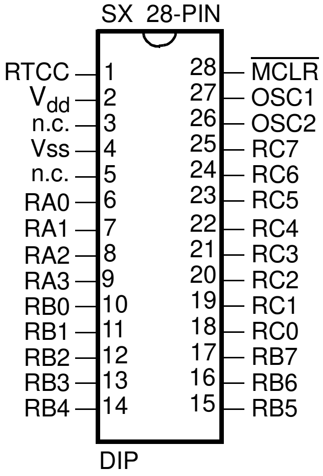
The device provides solutions for many familiar applications such as process controllers, electronic appliances/tools, security/monitoring systems, consumer automotive, sound generation, motor control, and personal communication devices. In addition, the device is suitable for applications that require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

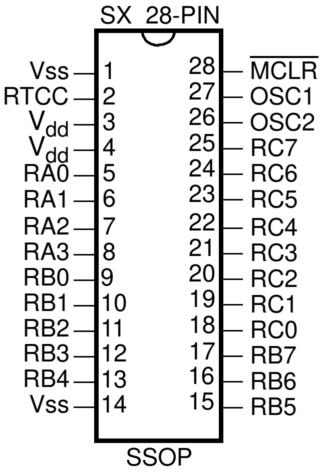
Furthermore, the growing Virtual Peripheral library features new components, such as the Internet Protocol stack, and communication interfaces, that allow design engineers to embed Internet connectivity into all of their products at extremely low cost and very little effort.

2.0 CONNECTION DIAGRAMS

2.1 Pin Assignments







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
RB0	I/O	TTL/CMOS/ST	Bidirectional I/O Pin; comparator output; MIWU/Interrupt input
RB1	I/O	TTL/CMOS/ST	Bidirectional I/O Pin; comparator negative input; MIWU/Interrupt input
RB2	I/O	TTL/CMOS/ST	Bidirectional I/O Pin; comparator positive input; MIWU/Interrupt input
RB3	I/O	TTL/CMOS/ST	Bidirectional I/O Pin; MIWU/Interrupt input
RB4	I/O	TTL/CMOS/ST	Bidirectional I/O Pin; MIWU/Interrupt input
RB5	I/O	TTL/CMOS/ST	Bidirectional I/O Pin; MIWU/Interrupt input
RB6	I/O	TTL/CMOS/ST	Bidirectional I/O Pin; MIWU/Interrupt input
RB7	I/O	TTL/CMOS/ST	Bidirectional I/O Pin; MIWU/Interrupt input
RC0	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RC1	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RC2	I/O	TTL/CMOS/ST	Bidirectional I/O pin
RC3	I/O	TTL/CMOS/ST	Bidirectional I/O pin
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
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 pin
Vss	P	–	Ground pin
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

Table 2-1. Ordering Information

Device	Pins	I/O	Operating Frequency (MHz)	EE/Flash (Words)	RAM (Bytes)	Operating Temp. (°C)
SX20AC/SS SX20AC/SS	20 20	12 12	50 75	2K 2K	136 136	-40°C to +85°C 0°C to +70°C
SX28AC/DP SX28AC/DP	28 28	20 20	50 75	2K 2K	136 136	-40°C to +85°C 0°C to +70°C
SX28AC/SS SX28AC/SS	28 28	20 20	50 75	2K 2K	136 136	-40°C to +85°C 0°C to +70°C

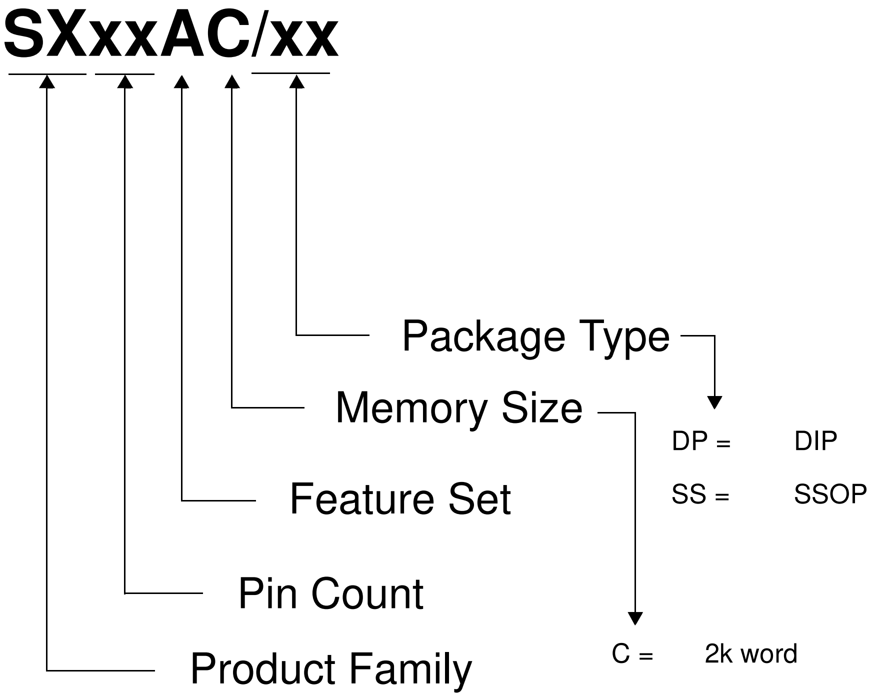


Figure 2-1. Part Number Reference Guide

3.0 PORT DESCRIPTIONS

The device contains a 4-bit I/O port (Port A) and two 8-bit I/O ports (Port B, Port C). Port A provides symmetrical drive capability. 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 upper four bits of the registers associated with Port A are not used. The least significant bit of the registers corresponds to the least significant port pin. To access these 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

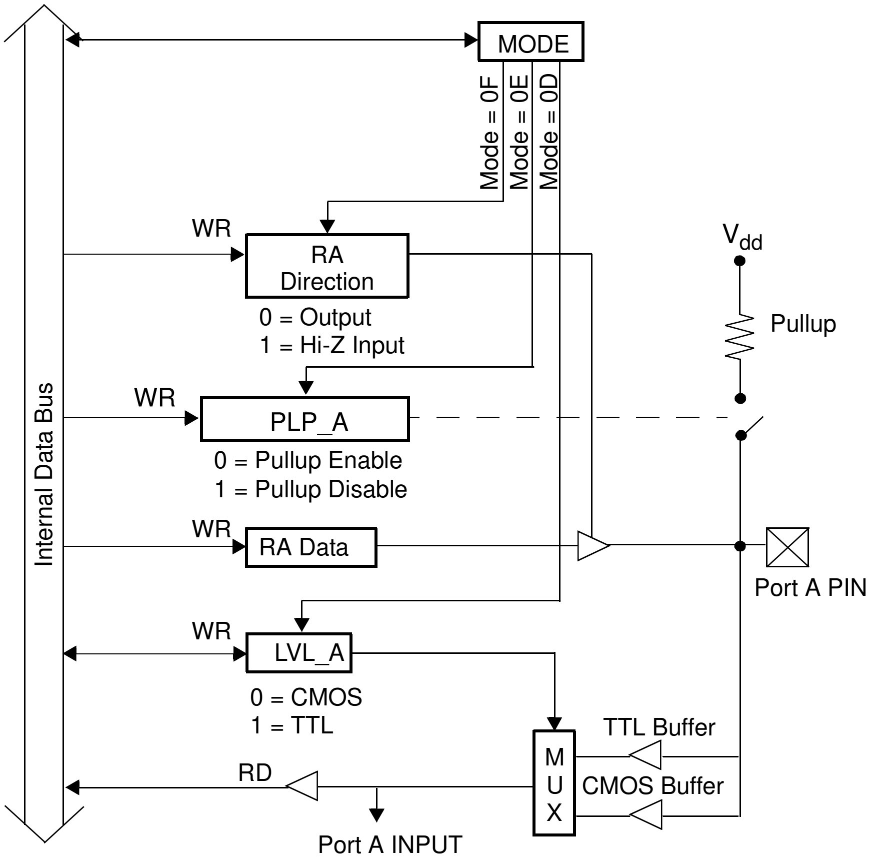


Figure 3-1. Port A Configuration

3.1 Reading and Writing the Ports

The three ports are memory-mapped into the data memory address space. To the CPU, the three ports are available as the RA, RB, and RC file registers at data memory addresses 05h, 06h, and 07h, 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 register reads the voltage levels of the corresponding port pins that have been configured as inputs.

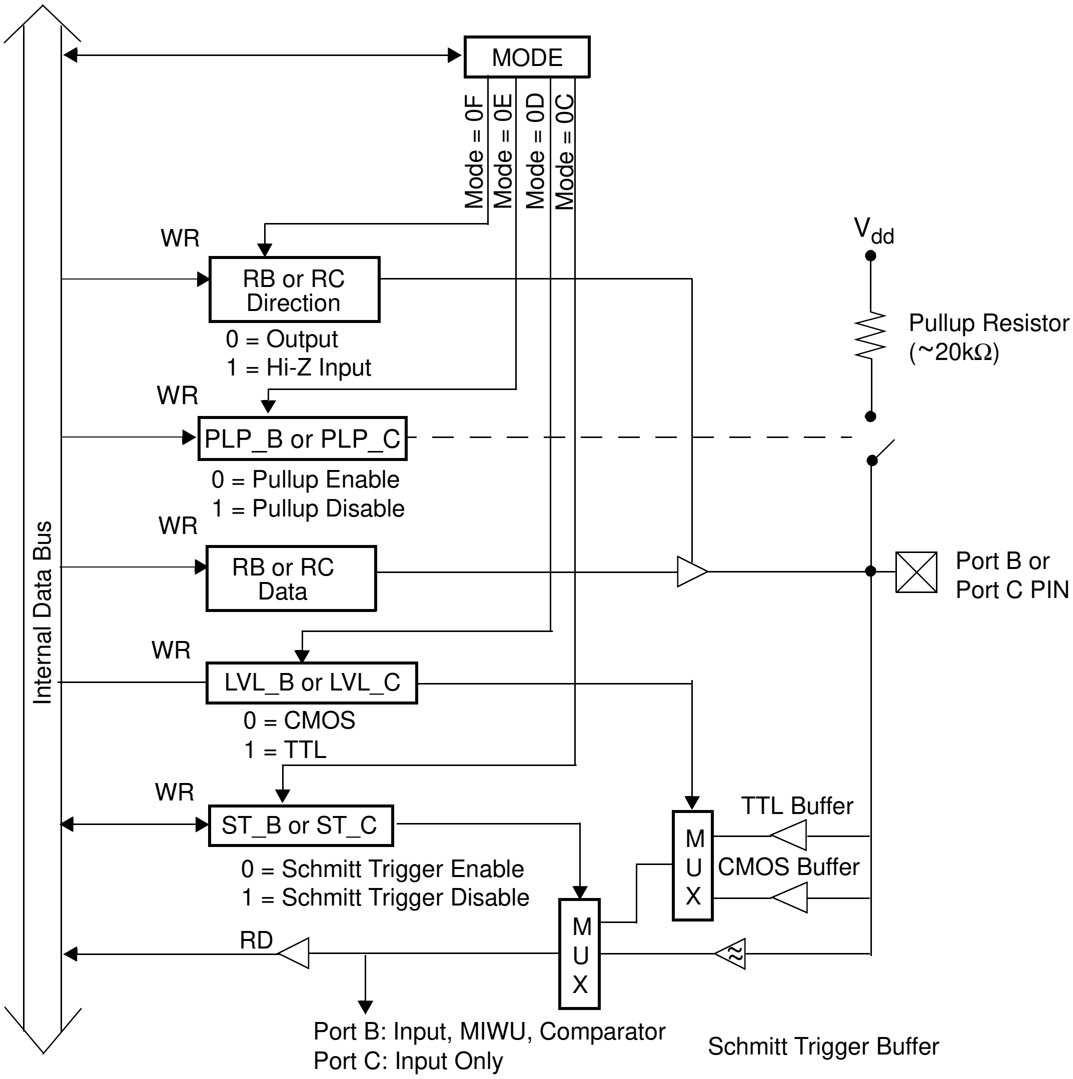


Figure 3-2. Port B, Port C Configuration

For example, suppose all four Port A pins are configured as outputs and with RA0 and RA1 to be high, and RA2 and RA3 to be low:

```
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 four Port A pins, RA0 through RA3. Because Port A has only four I/O pins, only the four least significant bits of this register are used. The four high-order register bits are “don’t care” bits. Port B and Port C are both eight bits wide, so the full widths of the RB and RC registers are used.

When a write is performed to a bit position for a port 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.

When a read is performed from a bit position for a port, 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.

3.1.1 Read-Modify-Write Considerations

Caution must be exercised when performing two successive read-modify-write instructions (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 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.

Also note that reading an I/O port is actually reading the pins, not the output data latches. That is, if the pin output driver is enabled and driven high while the pin is held low externally, the port pin will read low.

3.2 Port Configuration

Each port pin offers the following configuration options:

- data direction
- input voltage levels (TTL or CMOS)
- pullup type (pullup resistor enable or disable)
- Schmitt trigger input (for Port B and Port C only)

Port B offers the additional option to use the port pins for the Multi-Input Wakeup/Interrupt function and/or the analog comparator function.

Port configuration is performed 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 Port A control register)
- mov !RB,W (move W to Port B control register)
- mov !RC,W (move W to Port C control register)

Each one of these instructions writes a port control register for Port A, Port B, or Port C. 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. 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. MODE register values not listed in the table are reserved for future expansion and should not be used. Therefore, the MODE register should always contain a value from 08h to 0Fh. Upon reset, the MODE register is initialized to 0Fh, which enables access to the port direction registers.

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 0Eh 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 and Port Control Register Access

MODE Reg.	mov !RA,W	mov !RB,W	mov !RC,W
08h	not used	CMP_B	not used
09h	not used	WKPND_B	not used
0Ah	not used	WKED_B	not used
0Bh	not used	WKEN_B	not used
0Ch	not used	ST_B	ST_C
0Dh	LVL_A	LVL_B	LVL_C
0Eh	PLP_A	PLP_B	PLP_C
0Fh	RA Direction	RB Direction	RC Direction

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

```
mov  M,#$0E   ;MODE=0Eh to access port pullup
              ;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 0Eh to select access to the pullup control registers (PLP_A, PLP_B, and PLP_C). 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, RB, and RC Data Direction Registers (MODE=0Fh)

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, PLP_B, and PLP_C: Pullup Enable Registers (MODE=0Eh)

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, LVL_B, and LVL_C: Input Level Registers (MODE=0Dh)

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 and ST_C: Schmitt Trigger Enable Registers (MODE=0Ch)

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=0Bh)

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 7.0.

WKED_B: Wakeup Edge Register (MODE=0Ah)

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 Bit Register (MODE=09h)

When you access the WKPND_B register using MOV !RB,W, the CPU does an exchange between the contents of W and WKPND_B. This feature lets you read the WKPND_B register contents. Each bit 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. 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 bit 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 Reset

Upon reset, all the port control registers are initialized to FFh. Thus, each 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 0Fh, which allows immediate 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 the 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 eight file registers include the Real-Time Clock/Counter register (RTCC), the lower eight bits of the 11-bit Program Counter (PC), the 8-bit STATUS register, three port control registers for Port A, Port B, Port C, the 8-bit File Select Register (FSR), and INDF used for indirect addressing.

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, 11-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 Control register
06h	RB	Port RB Control register
07h	RC*	Port RC Control register

*In the SX18 package, Port C is not used, and address 07h is available as a general-purpose RAM location.

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)

010 = Page 2 (400h – 05FFh)

011 = Page 3 (600h – 07FFh)

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

After Addition:

1 = A carry from bit 7 of the result occurred

0 = No carry from bit 7 of the result occurred

After Subtraction:

1 = No borrow from bit 7 of the result occurred

0 = A borrow from bit 7 of the result occurred

Rotate (RR or RL) Instructions:

The carry bit is loaded with the low or high order bit, respectively. When $\overline{\text{CF}}$ bit is cleared, Carry bit works as input for ADD and SUB instructions.

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 function as described below.

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

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.

5.0 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.

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

$\overline{\text{TURBO}}$	$\overline{\text{SYNC}}$	Reserved	Reserved	$\overline{\text{IRC}}$	DIV1/ $\overline{\text{IFBD}}$	DIV0/ FOSC2	Re- served	$\overline{\text{CP}}$	WDTE	FOSC1	FOSC0

Bit 11 **Bit 0**

$\overline{\text{TURBO}}$ Turbo mode enable:

0 = turbo (instruction clock = osc/1)

1 = instr clock = osc/4

$\overline{\text{SYNC}}$ Synchronous input enable (for turbo mode): This bit synchronizes the signal presented at the input pin to the internal clock through two internal flip-flops.

0 = enabled

1 = disabled

$\overline{\text{IRC}}$ Internal RC oscillator enable:

0 = enabled - OSC1 pulled low by weak pullup, OSC2 pulled high by weak pullup

1 = disabled - OSC1 and OSC2 behave according to FOSC2: FOSC0

DIV1: DIV0 Internal RC oscillator divider:

00b	=	4 MHz
01b	=	1 MHz
10	=	128 KHz
11b	=	32 KHz

$\overline{\text{IFBD}}$ Internal crystal/resonator oscillator feedback resistor (1 MΩ):

0= disabled Internal feedback resistor disable (external feedback required)

1= enabled Internal feedback resistor enabled (valid when $\overline{\text{IRC}}$ = 1)

$\overline{\text{CP}}$ 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

FOSC2: FOSC0 External oscillator configuration (valid when $\overline{\text{IRC}}$ = 1):

000b = LP1 – low power crystal (32KHz)

001b = LP2 – low power crystal/resonator (32 KHz to 1 MHz)

010b = XT1 – normal crystal/resonator (32 KHz to 10 MHz)

011b = XT2 – normal crystal/resonator (1MHz to 24 MHz)

100b = HS1 – high speed crystal/resonator/external crystal oscillator (1MHz to 50 MHz)

101b = HS2 – high speed crystal/resonator/external crystal oscillator (1 MHz to 50 MHz)

110b = HS3 – high speed crystal/resonator/external crystal oscillator (1 MHz to 75 MHz)

111b = RC network - OSC2 is pulled high with a weak pullup (no CLKOUT output)

Note: The frequencies are target values.

5.2 FUSEX Word (Read/Program via Programming Command)

IRCTRIM2	PINS	IRCTRIM1	IRCTRIM0	$\overline{\text{OPTIONX}}$/ $\overline{\text{STACKX}}$	$\overline{\text{CF}}$	BOR1	BOR0	BORTRIM1	BORTRIM0	BP1	BP0

Bit 11 **Bit 0**

IRCTRIM2: IRCTRIM0 Internal RC oscillator trim bits. This 3-bit field adjusts the operation of the internal RC oscillator to make it operate within the target frequency range 4 MHz plus or minus 8%. Parts are shipped from the factory untrimmed. The device relies on the programming toll to provide the trimming function.

000b = minimum frequency

111b = maximum frequency

each step about 3%

PINS Selects the number of pins.

0 = 18/20 pins

1 = 28 pins

$\overline{\text{OPTIONX}}$/ $\overline{\text{STACKX}}$ OPTION Register Extension and Stack Extension. Set to 1 to disable the programmability of bit 6 and bit 7 in the OPTION register, the RTW and RTE_IE bits (in other words, to force these two bits to 1) and to limit the program stack size to two locations. Clear to 0 to enable programming of the RTW and RTE_IE bits in the OPTION register, and to extend the stack size to eight locations.

$\overline{\text{CF}}$ active low – makes carry bit input to ADD and SUB instructions.

BOR1: BOR0 Brown-Out Reset;These bits enable or disable the brown-out reset function and set the brown-out threshold voltage as follows:

00b = 4.2V

01b = 2.6V

10b = 2.2V

11b = Brown-Out disabled

BORTRIM1: BORTRIM0 Brown-Out trim bits (parts are shipped out of factory untrimmed).

BP1:BP0 Configure Memory Size:

00b =	1 page, 1 bank	
01b =	2 pages, 1 bank	
10b =	4 pages, 4 banks	
11b =	4 pages, 8 banks	(default configuration)

5.3 DEVICE Word (Hard-Wired Read-Only)

1	1	1	1	1	1	0	0	1	1	1	0

Bit 11 **Bit 0**

6.0 MEMORY ORGANIZATION

6.1 Program Memory

The program memory is organized as 2K, 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.

6.1.1 Program Counter

The program counter contains the 11-bit address of the instruction to be executed. The lower eight bits of the program counter are contained in the PC register (02h) while the upper bits come from the upper three bits of the STATUS register (PA0, PA1, PA2). This 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 07FFh.

6.1.2 Subroutine Stack

The subroutine stack consists of eight 11-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.

6.2 Data Memory

The data memory consists of 136 bytes of RAM, organized as eight banks of 16 registers plus eight registers which are not banked. Both banked and non-banked memory locations can be addressed directly or indirectly using the FSR (File Select Register). The special-function registers are mapped into the data memory.

6.2.1 File Select Register (04h)

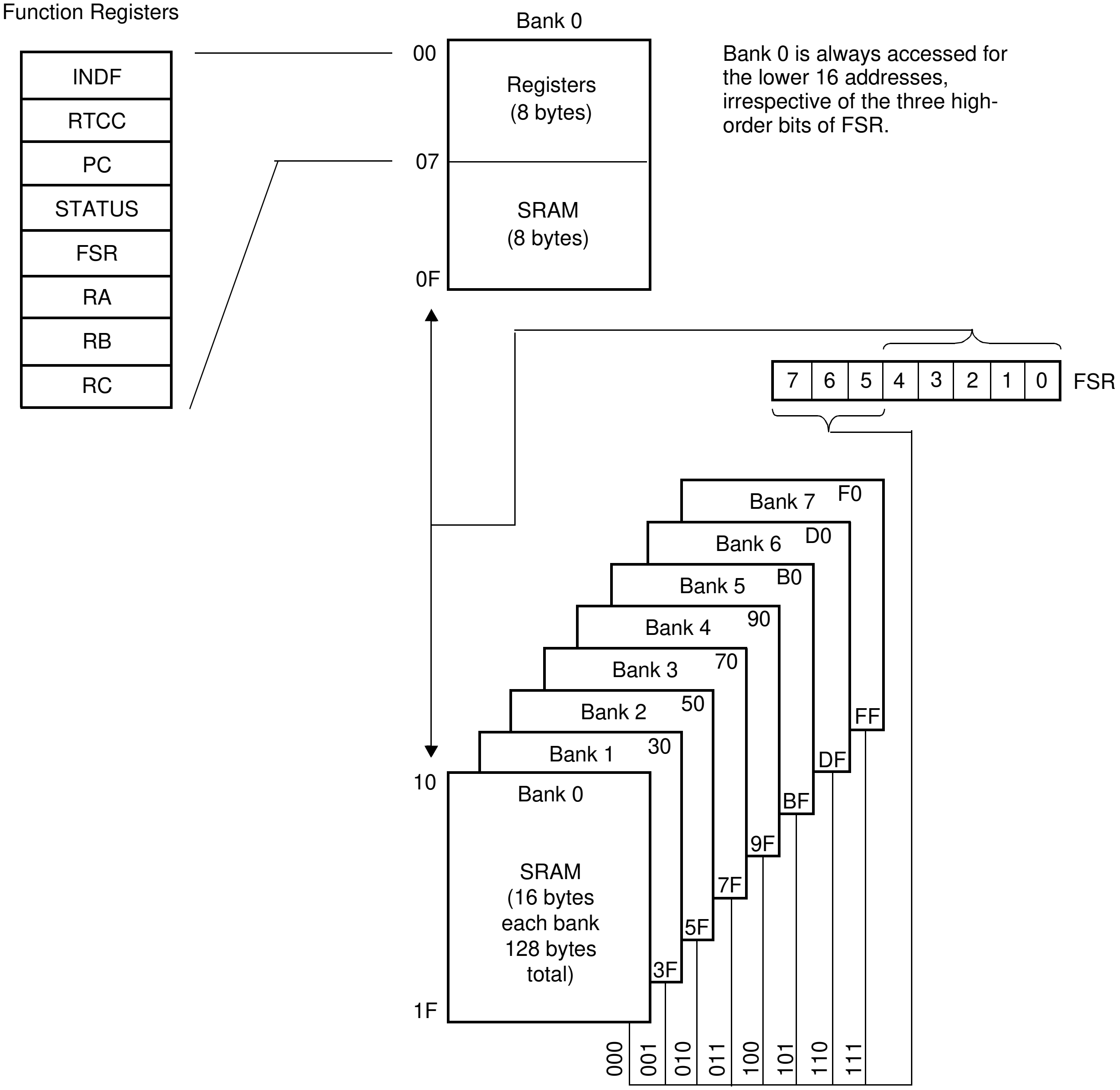
Instructions that specify a register as the operand can only express five bits of register address. This means that only registers 00h to 1Fh can be accessed. The File Select Register (FSR) provides the ability to access registers beyond 1Fh.

Figure 6-1 shows how FSR can be used to address RAM locations. The three high-order bits of FSR select one of eight SRAM banks to be accessed. The five low-order bits select one of 32 SRAM locations within the selected bank. For the lower 16 addresses, Bank 0 is always accessed, irrespective of the three high-order bits. Thus, RAM register addresses 00h through 0Fh are “global” in that they can always be accessed, regardless of the contents of the FSR.

The entire data memory (including the dedicated-function registers) consists of the lower 16 bytes of Bank 0 and the upper 16 bytes of Bank 0 through Bank 7, for a total of (1+8)*16 = 144 bytes. Eight of these bytes are for the function registers, leaving 136 general-purpose memory locations. In the 18-pin SX packages, register RC is not used, which makes address 07h available as an additional general-purpose memory location.

Below is an example of how to write to register 10h in Bank 4:

```
mov    FSR,#$90    ;Select Bank 4 by
                   ;setting FSR<7:5>
mov    $10,#$64    ;load register 10h with
                   ;the literal 64h
```



Bank 0 is always accessed for the lower 16 addresses, irrespective of the three high-order bits of FSR.

Figure 6-1. Data Memory Organization

7.0 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.

7.1 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. Clearing 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.

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 7-1 shows the Multi-Input Wakeup block diagram.

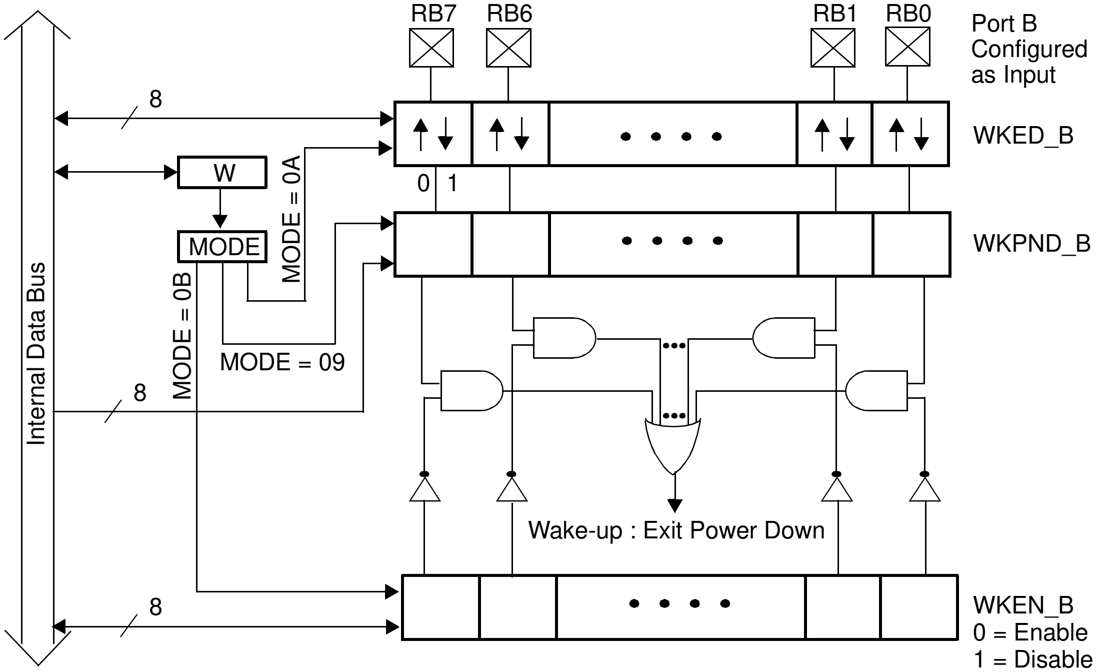


Figure 7-1. Multi-Input Wakeup Block Diagram

7.2 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   M,#$09
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 Port B accesses. 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 M,#$0F  ;prepare to write port data
            ;direction registers
mov W,#$07  ;load W with the value 07h
mov !RB,W   ;configure RB0-RB2 to be inputs

mov M,#$0A  ;prepare to write WKED_B
            ;(edge) register
            ;W contains the value 07h
mov !RB,W   ;configure RB0-RB2 to sense
            ;falling edges
mov M,#$09  ;prepare to access WKPND_B
            ;(pending) register
mov W,#$00  ;clear W
mov !RB,W   ;clear all wakeup pending bits

mov M,#$0B  ;prepare to write WKEN_B (enable)
            ;register
mov W,#$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 the SLEEP instruction), the device can then receive wakeup signals on those same pins.

8.0 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 0FFh to 00h. This interrupt source has an associated enable bit located in the OPTION register. There is no pending bit associated with this interrupt.

Port B provides the source for eight external software selectable, edge sensitive interrupts. 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 clearing 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. The proper sequence is described in Section 7.2 .

Figure 8-1 shows the structure of the interrupt logic.

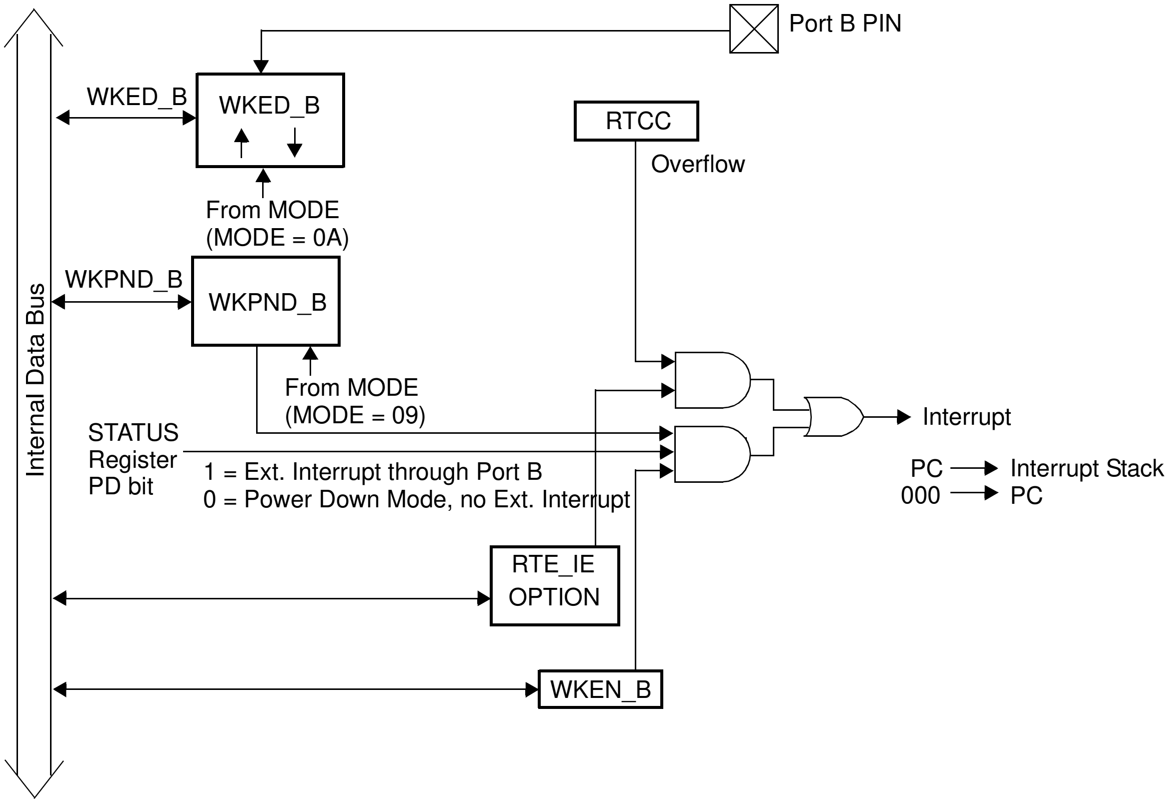


Figure 8-1. Interrupt Structure

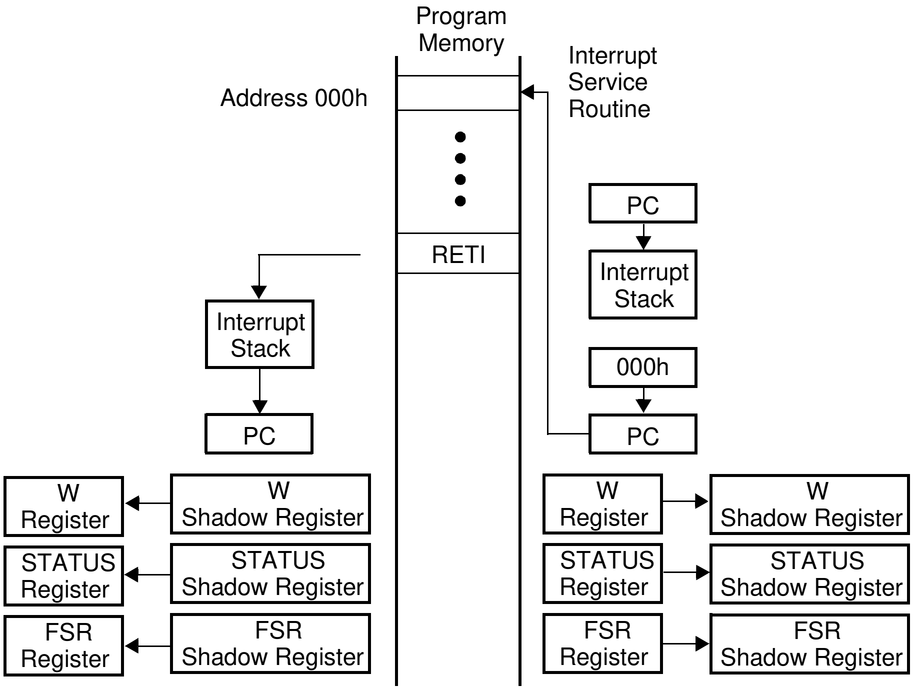
All interrupts are global in nature; that is, no interrupt has priority over another. Interrupts are handled sequentially. Figure 8-2 shows the interrupt processing sequence. Once an interrupt is acknowledged, all subsequent global interrupts are disabled until return from servicing the current interrupt. 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. The status bits PA0, PA1, and PA2 bits are cleared after the STATUS register has been saved in its 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 routine is address 0.

Once in the interrupt service routine, the user program must check all external interrupt pending bits (contained in the WKPND_B register) to determine the source of the interrupt. The interrupt service routine should clear the corresponding interrupt pending bit. If both internal and external interrupts are enabled, the user program may also need to read the contents of RTCC to determine any recent RTCC rollover. This is needed since there is no interrupt pending bit associated with the RTCC rollover.

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.

If an external interrupt occurs during the interrupt routine, the pending register will be updated but the trigger will be ignored unless interrupts are disabled at the beginning of the interrupt routine and enabled again at the end. This also requires that the new interrupt does not occur before interrupts are disabled in the interrupt routine. If there is a possibility of additional interrupts occurring before they can be disabled, the device will miss those interrupt triggers. In other words, using more than one interrupt, such as multiple external interrupts or both RTCC and external interrupts, can result in missed or, at best, jittery interrupt handling should one occur during the processing of another. When handling external interrupts, the interrupt routine should clear at least one pending register bit. The bit that is cleared should represent the interrupt being handled in order for the next interrupt to trigger.

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 and 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 W to RTCC. The interrupt return instruction enables the global interrupts.



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

Figure 8-2. Interrupt Processing

9.0 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

9.1 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 9-1. 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. Table 9-1 shows the recommended external components associated with a crystal-based oscillator. Table 9-2 shows the recommended external component values for a resonator-based oscillator.

Bits 0, 1 and 5 of the FUSE register (FOSC1:FOSC2) are used to configure the different external resonator/crystal oscillator modes. These bits allow the selection of the appropriate gain setting for the internal driver to match the desired operating frequency. If the XT, LP, or HS mode is selected, the OSC1/CLKIN pin can be 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 9-2). In this case, the OSC2/CLKOUT pin should be left open.

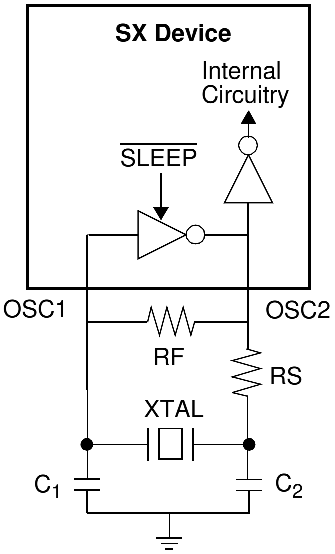


Figure 9-1. Crystal Operation (or Ceramic Resonator) (HS, XT or LP OSC Configuration)

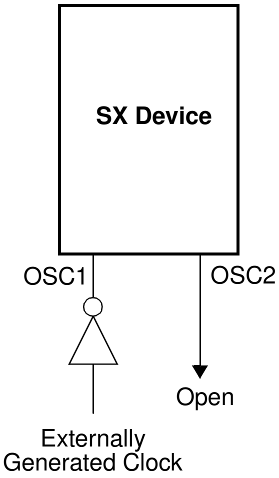


Figure 9-2. External Clock Input Operation (HS, XT or LP OSC Configuration)

Table 9-1. External Component Selection for Crystal Oscillator (Vdd=5.0V), Rs = 0 Ω

FOSC2:FOSC0	Crystal Frequency	C1	C2	R_F
010	4 MHz	15 pF	22 pF	1 MΩ
011	8 MHz	56 pF	33 pF	1 MΩ
011	20 MHz	33 pF	22 pF	1 MΩ
011	32 MHz	15 pF	22 pF	1 MΩ
100	50* MHz	15 pF	15 pF	1 MΩ

* 50 MHz fundamental crystal

Table 9-2. External Component Selection for Murata Ceramic Resonators (Vdd=5.0V)

FOSC2:FOSC0	Resonator Frequency	Resonator Part Number	C1	C2	R_F	R_S
011	4 MHz	CSA4.00MG	30 pF	30 pF	1MΩ	0 Ω
011	4 MHz	CST4.00MGW	Internal (30 pF)	Internal (30 pF)	1 MΩ	0 Ω
011	4 MHz	CSTCC4.00G0H6	Internal (47 pF)	Internal (47 pF)	1 MΩ	0 Ω
011	8 MHz	CSA8.00MTZ	30 pF	30 pF	1 MΩ	0 Ω
011	8 MHz	CST8.00MTW	Internal (30 pF)	Internal 30 pF)	1 MΩ	0 Ω
011	8 MHz	CSTCC8.00MG0H6	Internal (47 pF)	Internal 47pF)	1 MΩ	0 Ω
011	20 MHz	CSA20.00MXZ040	5 pF	5 pF	1 MΩ	0 Ω
011	20 MHz	CST20.00MXW0H1	Internal (5 pF)	Internal (5 pF)	1 MΩ	0 Ω
011	20 MHz	CSACV20.00MXJ040	5 pF	5 pF	22 kΩ	0 Ω
011	20 MHz	CSTCV20.00MXJ0H1	Internal (5 pF)	Internal (5 pF)	22 kΩ	0 Ω
100	33 MHz	CSA33.00MXJ040	5 pF	5 pF	1 MΩ	0 Ω
100	33 MHz	CST33.00MXW040	Internal (5 pF)	Internal (5 pF)	1 MΩ	0 Ω
100	33 MHz	CSACV33.00MXJ040	5 pF	5 pF	1 MΩ	0 Ω
100	33 MHz	CSTCV33.00MXJ040	Internal (5 pF)	Internal (5 pF)	1 MΩ	0 Ω
101	50 MHz	CSA50.00MXZ040	15 pF	15 pF	10 kΩ	0 Ω
101	50 MHz	CST50.00MXW0H3	Internal (15 pF)	Internal (15 pF)	10 kΩ	0Ω
101	50 MHz	CSACV50.00MXJ040	15 pF	15 pF	10 kΩ	0 Ω
101	50 MHz	CSTCV50.00MXJ0H3	Internal (15 pF)	Internal (15 pF)	10 kΩ	0 Ω

9.2 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 9-3 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.

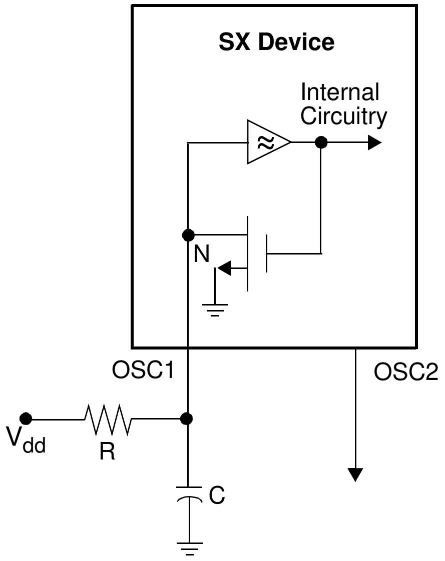


Figure 9-3. RC Oscillator Mode

9.3 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 typically +/–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.

10.0 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 10-1 shows the RTCC and WDT block diagram.

10.1 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. Clearing the bit selects the rising edge.

The RTCC generates an interrupt as a result of an RTCC rollover from 0FF to 000. There is no interrupt pending bit to indicate the overflow occurrence. The RTCC register must be sampled by the program to determine any overflow occurrence.

10.2 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.

10.3 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.

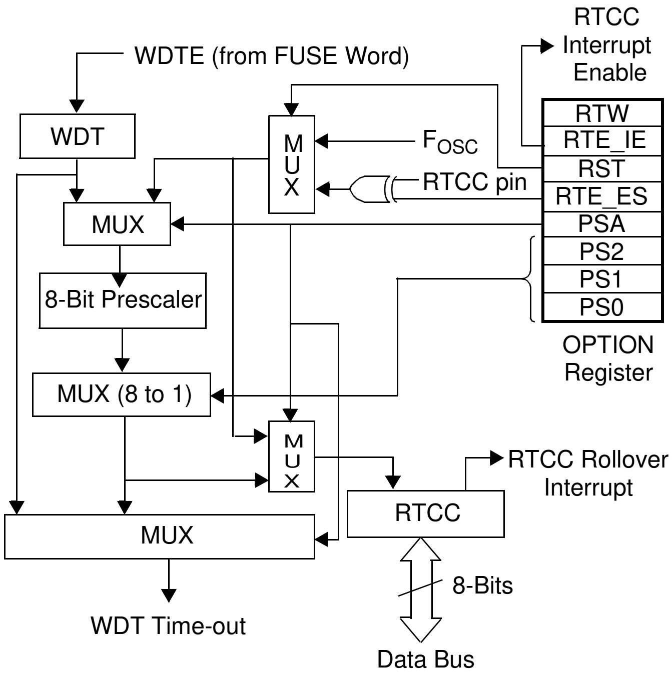


Figure 10-1. RTCC and WDT Block Diagram

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 power down mode, the comparator should be disabled before entering the power down mode. Here is an example of how to setup the comparator and read the CMP_B register.

```
mov  M,#$08   ;set MODE register to access
              ;CMP_B

mov  W,#$00   ;clear W

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

...           ;delay after enabling
              ;comparator for response

mov  M,#$08   ;set MODE register to access
              ;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 bit based on
              ;comparator result

snb  $03.2    ;test Z bit 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 Port B accesses. Otherwise, the “mov” instruction does not perform an exchange, but only moves data from the source to the destination.

Figure 11-1 shows the comparator block diagram.

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.