Texas Instruments

MSC120x Precision ADC and DACs with 8051 Microcontroller and Flash Memory

User's Guide

April 2006

Data Acquisition

SBAU112

Contents

List of Figures

List of Tables

About This Manual

This user's guide describes the function and operation of the MSC120x family of precision ADC and DACs with 8051 microcontroller and flash memory.

This document applies to the following MSC devices:

- MSC1200
- MSC1201
- MSC1202

For convenience, the abbreviation MSC120x is used to indicate the MSC1200, MSC1201, and MSC1202.

The abbreviation MSC12xx is used to indicate entire MSC family, including the MSC120x devices and also the MSC1210, MSC1211, MSC1212, MSC1213, and MSC1214.

Related Documentation and Tools From Texas Instruments

Data Sheets	Literature Number
MSC1200	SBAS317
MSC1201	SBAS317
MSC1202	SBAS317

User's Guides	Literature Number
MSC1200EVM User's Guide	SBAU098
MSC1201/02EVM User's Guide	SBAU105

For a complete list of application notes and related documentation, see the MSC web site at www.ti.com/msc.

Trademarks

I²C is a trademark of Koninklijke Philips Electronics N.V.

SPI is a trademark of Motorola.

All other trademarks are the property of their respective owners.

Introduction

This chapter provides a functional overview of the MSC120x precision analog-to-digital converter (ADC) and digital-to-analog converters (DACs) with 8051 microcontroller and flash memory.

Topic **Page**

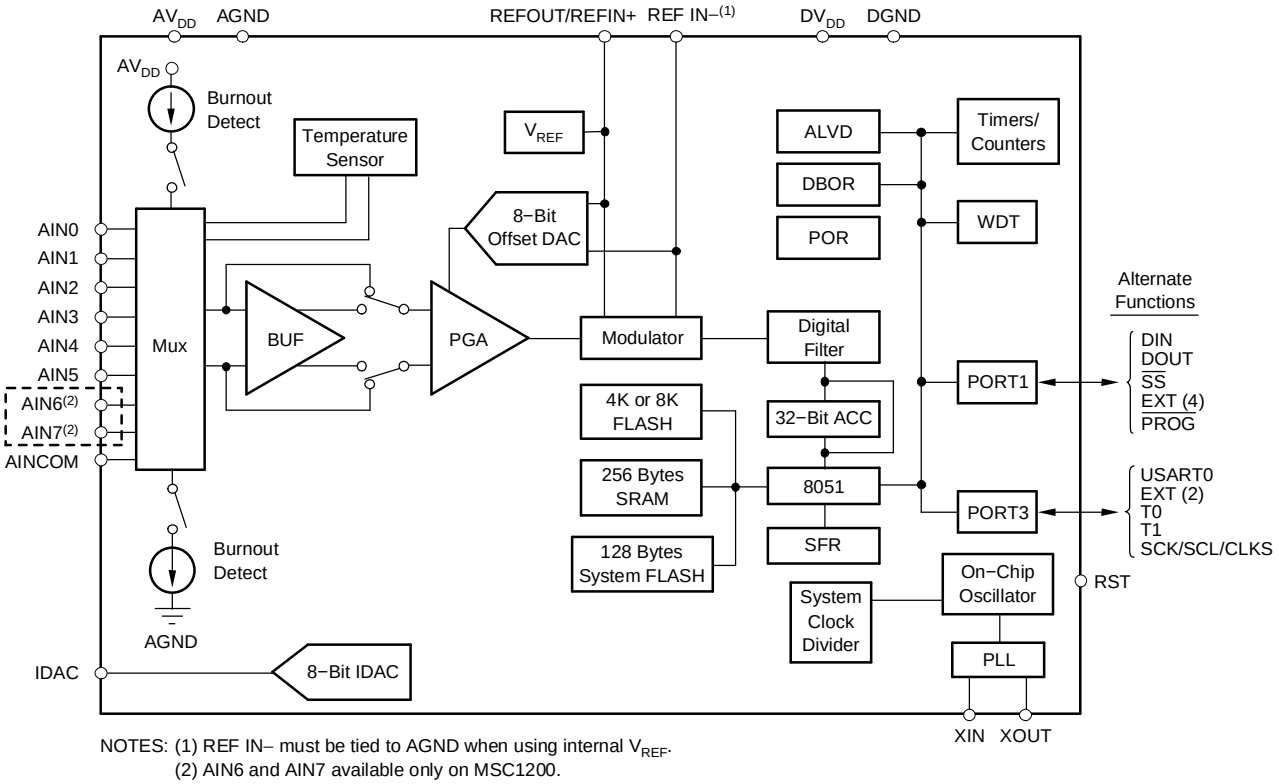
1.1 MSC120x Description

The MicroSystem family of devices is designed for high-resolution measurement applications in smart transmitters, industrial process control, weigh scales, chromatography, and portable instrumentation. They provide cost-effective, space-saving, high-performance, mixed-signal solutions. The MicroSystem family not only includes high-performance analog features and digital processing capability, but also integrates many digital peripherals to offer a unique and effective system solution.

The main components of a MicroSystem product include:

- High-performance analog functions
- Low-power enhanced 8051 microcontroller core
- RAM and Flash memory
- High-performance digital peripherals

The enhanced 8051 microcontroller includes dual data pointers and executes most instructions up to three times faster than a standard 8051 core. This increased execution speed provides greater flexibility in applications requiring a trade-off among speed, power and noise. A block diagram of the MSC120x is shown in Figure 1-1.



NOTES: (1) REF IN– must be tied to AGND when using internal V_{REF}.
(2) AIN6 and AIN7 available only on MSC1200.

Figure 1-1. MSC120x Block Diagram

For some designers, the MSC120x is viewed as a microcontroller with integrated analog functions, while to others it is a high-performance analog-to-digital converter (ADC) with an integrated microcontroller. The MSC120x provides unparalleled analog and digital integration for all designers concerned with embedded instrumentation and control.

Complementing the high-resolution ADC are a precision voltage reference, programmable gain amplifier (PGA), and analog multiplexer (mux), as well as a temperature sensor and low voltage detectors.

Apart from two bit-wise programmable digital ports, there is one full-duplex USART, two timer/counters, a programmable watchdog timer, a basic serial (SPI™) bus, and a basic I^2C bus. Up to 8k of FLASH memory and 256 bytes of RAM are included as well.

Taken together, the MSC120x features blend analog and digital functions to significantly simplify overall system design, which reduces the design time and board space, as well as the need for external components.

Table 1-1 compares the basic features and functionality of the MSC120x family.

Table 1-1. MSC120x Product Family Matrix

	MSC1200	MSC1201	MSC1202
Maximum Clock Frequency (MHz)	33	33	33
Flash Memory (kB)	4/8	4/8	4/8
SRAM (Bytes)	256	256	256
ADC (Channel x Resolution)	8 × 24	6 x 24	6 x 16
DAC (Channel x Resolution)	1 x 8 (IDAC)	1 x 8 (IDAC)	1 x 8 (IDAC)
I/Os	2 x 8	2 x 8	2 x 8
USART	1	1	1
SPI / I^2C	1	1	1
Digital Brownout Reset	1	1	1
Analog Low-Voltage Detect	1	1	1
Package	TQFP-48	QFN-36	QFN-36

1.2 MSC120x Pinout

The names and functions of the MSC120x pins are similar to those found on most 8051-compatible devices, but with extensions that are specific to the MSC120x. The pin configuration for the MSC1200 (TQFP) is shown in Figure 1-2, and the pin configuration for the MSC1201 and MSC1202 (QFN) is shown in Figure 1-3. The pin descriptions are listed in Table 1-2.

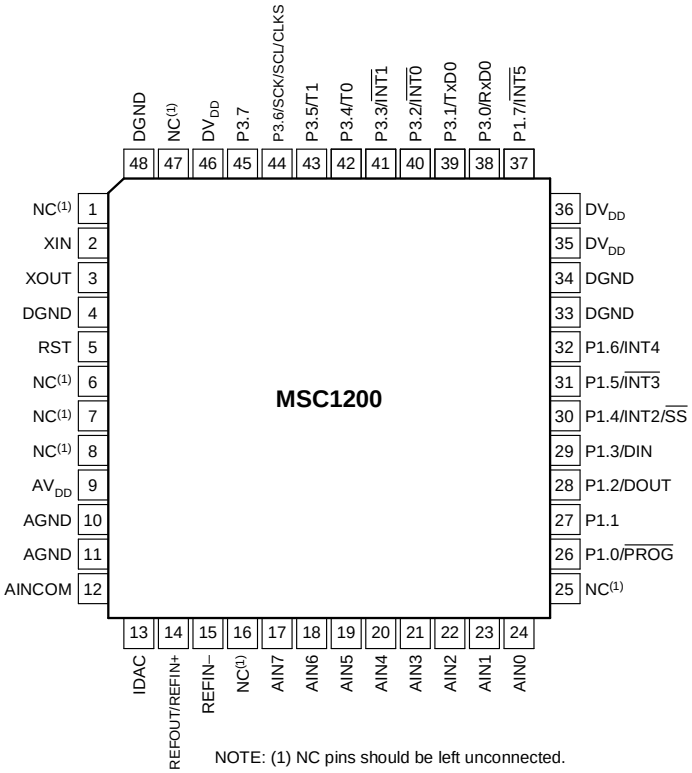


Figure 1-2. MSC1200 (TQFP-48) Pin Configuration

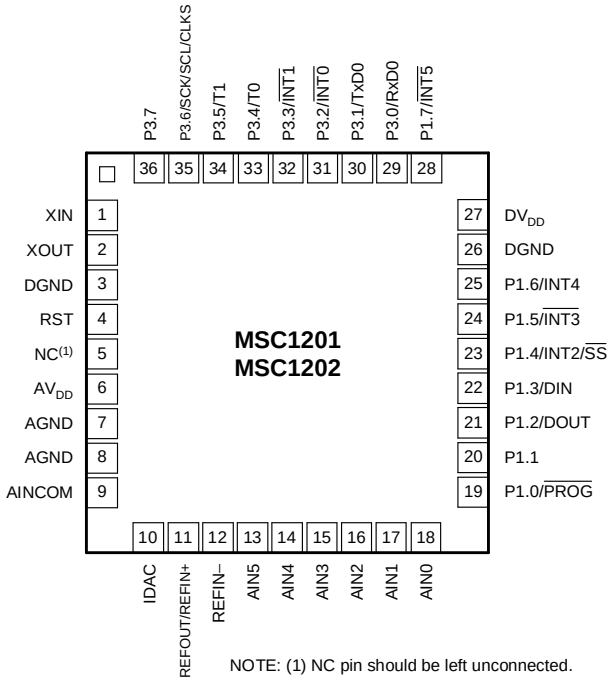


Figure 1-3. MSC1201 and MSC1202 (QFN-36) Pin Configuration

Table 1-2. Pin Descriptions

Name	MSC1200 Pin #	MSC1201/1202 Pin #	Description		
NC	1, 6, 7, 8, 16, 25, 47	5	No Connection. Leave unconnected.		
XIN	2	1	The crystal oscillator pin XIN supports parallel resonant AT-cut fundamental frequency crystals and ceramic resonators. XIN can also be an input if there is an external clock source instead of a crystal. XIN must not be left floating.		
XOUT	3	2	The crystal oscillator pin XOUT supports parallel resonant AT-cut fundamental frequency crystals and ceramic resonators. XOUT serves as the output of the crystal amplifier. Note that it should not be used as a clock source.		
DGND	4, 33, 34, 48	3, 26	Digital ground		
RST	5	4	Holding the reset input high for two t_{OSC} periods will reset the device.		
AV_{DD}	9	6	Analog power supply		
AGND	10, 11	7, 8	Analog ground		
AINCOM	12	9	Analog input (can be analog common for single-ended inputs or analog input for differential inputs)		
IDAC	13	10	IDAC output		
REFOUT/REF IN+	14	11	Internal voltage reference output/voltage reference positive input (required C_{REF} = 0.1µF)		
REF IN–	15	12	Voltage reference negative input (tie to AGND for internal voltage reference)		
AIN7	17	—	Analog input channel 7		
AIN6	18	—	Analog input channel 6		
AIN5	19	13	Analog input channel 5		
AIN4	20	14	Analog input channel 4		
AIN3	21	15	Analog input channel 3		
AIN2	22	16	Analog input channel 2		
AIN1	23	17	Analog input channel 1		
AIN0	24	18	Analog input channel 0		
P1.0-P1.7	26-32, 37	19-25, 28	Port 1 is a bidirectional I/O port (refer to P1DDRL, SFR AEh, and P1DDRH, SFR AFh, for port pin configuration control). The alternate functions for Port 1 are listed below.		
			Port 1.x	**Alternate Name(s)**	**Alternate Use**
			P1.0	$\overline{\text{PROG}}$	Serial programming mode (must be LOW on reset)
			P1.1	N/A	
			P1.2	DOUT	Serial data out
			P1.3	DIN	Serial data in
			P1.4	INT2/$\overline{\text{SS}}$	External interrupt 2 / SPI slave select
			P1.5	$\overline{\text{INT3}}$	External interrupt 3
			P1.6	INT4	External interrupt 4
			P1.7	$\overline{\text{INT5}}$	External interrupt 5
DV_{DD}	35, 36, 46	27	Digital Power Supply		
P3.0-P3.7	38-45	29-36	Port 3 is a bidirectional I/O port (refer to P3DDRL, SFR B3h, and P3DDRH, SFR B4h, for port pin configuration control). The alternate functions for Port 3 are listed below.		
			Port 3.x	**Alternate Name(s)**	**Alternate Use**
			P3.0	RxD0	Serial port 0 input
			P3.1	TxD0	Serial port 0 output
			P3.2	$\overline{\text{INT0}}$	External Interrupt 0
			P3.3	$\overline{\text{INT1}}$	External interrupt 1
			P3.4	T0	Timer 0 external input
			P3.5	T1	Timer 1 external input
			P3.6	SCK/SCL/CLKS	SCK / SCL / various clocks (refer to PASEL, SFR F2h in the data sheet)
			P3.7	N/A	

1.2.1 Input/Output (I/O) Ports – P1 and P3

In principle, each port consists of eight bits. Each bit may be placed low, high, or read by accessing the corresponding bit in the appropriate special function register (SFR). However, when alternate functions are used, the corresponding bit of the port SFR will reflect the alternate function.

Every I/O port bit has an optional pull-up resistor that is enabled when the bit is in standard 8051 mode (default after reset), as configured by the P*x*DDRH and P*x*DDRL SFRs, where *x* = 1 or 3. The pull-up resistor is disabled when the port bit is configured as either a CMOS output, or an open-drain output or input, as shown in Figure 1-4 through Figure 1-7.

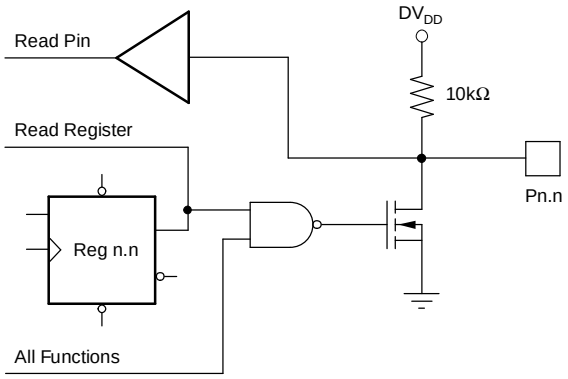


Figure 1-4. Standard 8051 I/O Pin Structure

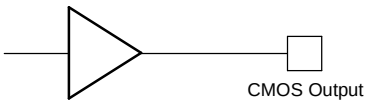


Figure 1-5. CMOS Output Pin Structure

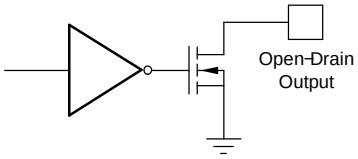


Figure 1-6. Open-Drain Output Pin Structure

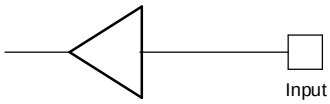


Figure 1-7. Input Pin Structure

Note that:

- When a port pin is to act as an input to an alternate function, it is essential that the pin is not configured as an output.
- To make use of the alternate functions associated with Ports 1 and 3, the corresponding port output latches should be high, with the data direction bits defined in a manner appropriate to the alternate function.
- A special case exists for the 8051 mode, which has a weak pull-up resistor and offers bidirectional capability.

1.2.1.1 Port 1 – P1

Port 1 is a bidirectional I/O port, which provides not only eight independently-programmable bits, but also alternate functions, as shown in Table 1-3.

Table 1-3. Port 1 Alternate Functions

Port 1 Bit Name	Alternate Function
P1.0 ($\overline{\text{PROG}}$)	Serial programming mode (must be LOW on reset)
P1.1	N/A
P1.2 (DOUT)	Serial data out
P1.3 (DIN)	Serial data in
P1.4 (INT2/$\overline{\text{SS}}$)	Positive-edge triggered external interrupt 2 or active-low slave-select input during SPI operation
P1.5 ($\overline{\text{INT3}}$)	Negative-edge triggered external interrupt 3
P1.6 (INT4)	Positive-edge triggered external interrupt 4
P1.7 ($\overline{\text{INT5}}$/SCK/SCL)	Negative-edge triggered external interrupt 5 or clock signal during SPI / I^2C operation.

1.2.1.2 Port 3 – P3

Port 3 is a bidirectional I/O port, which provides not only eight independently programmable bits, but also alternate functions, as shown in Table 1-4.

Table 1-4. Port 3 Alternate Functions

Port 3 Bit Name	Alternate Function
P3.0 (RxD0)	Serial input to USART0. An external receiver is needed for RS232 signals.
P3.1 (TxD0)	Serial output from USART0. An external driver is needed for RS232 signals.
P3.2 ($\overline{\text{INT0}}$)	Active-low or negative-edge triggered external interrupt 0
P3.3 ($\overline{\text{INT1}}$)	Active-low or negative-edge triggered external interrupt 1
P3.4 (T0)	Clock source for Timer/Counter 0 if TMOD.1 is set. See Chapter 9, Timer/Counters, for gated conditions.
P3.5 (T1)	Clock source for Timer/Counter 1 if TMOD.2 is set. See Chapter 9, Timer/Counters, for gated conditions.
P3.6 (SCK/SCL/CLKS)	Multiple functions (refer to PASEL, SFR F2h in the data sheet)
P3.7	N/A

1.2.2 Oscillator – XOUT and XIN

In many applications, a quartz crystal or ceramic resonator is connected between XOUT and XIN to provide a reference clock that is between 1MHz and approximately 30MHz. For the MSC120x, a commonly-used crystal for exact baud rates is 11.0592MHz. The static design of the MSC120x allows a digital clock that is between 0MHz and 30MHz to be applied to XIN .

The MSC120x provides an internal oscillator and phase lock loop (PLL), which means the MSC120x can work without an external crystal or oscillator.The onboard PLL works with a 32kHz external crystal.

Note: The load capacitors for the crystal must be verified to work over the operating conditions of the application. Due to the design of the oscillator circuit, it is generally better to use lower value load capacitors than those recommended by the crystal manufacturer.

1.2.3 Reset Line – RST

RST is the master reset line. When it is brought high for two or more clock cycles, the MSC120x is reset. All SFRs are placed at their default values and the program counter is reset to 0000h. The contents of internal SRAM are not affected by a reset. Instruction execution begins (2^{17}– 1) clock cycles after RST is brought low and when $\overline{\text{PROG}}$ is high. If $\overline{\text{PROG}}$ is low when RST is brought low, the MSC120x enters Flash Programming mode.

The RST pin has a CMOS Schmitt-trigger input that permits the use of a simple RC network to achieve reset when power is first applied. There is no internal pull-down resistor.

1.3 Enhanced 8051 Core

The entire MSC family of mixed-signal microcontrollers (MSC12xx) uses a core that is instruction-set-compatible with the industry-standard 8051. All instruction codes have the same binary patterns and produce exactly the same logical changes. However, the MSC12xx is approximately three times faster in execution for the same clock frequency; instead of using 12 clocks per instruction cycle, the MSC12xx uses four, as shown in Figure 1-8.

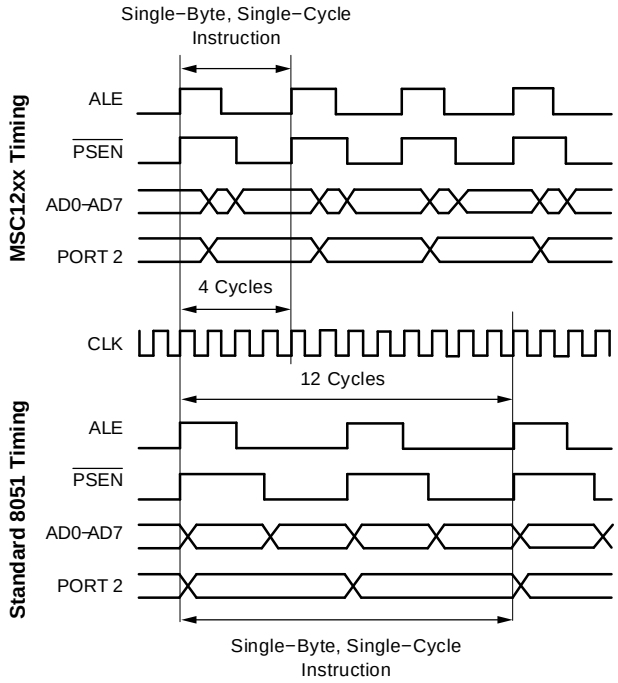


Figure 1-8. Comparison of MSC12xx Timing to Standard 8051 Timing

The designer can either make use of the increased execution speed, or operate at the same speed, but at a lower clock frequency. A lower clock speed results in less system noise and lower power dissipation.

When porting existing 8051 code to the MSC12xx, the designer/programmer may need to consider the change in performance associated with all software timing loops and make adjustments where necessary. By default, hardware timers are still clocked every 12 clock cycles, but can be changed to every four cycles, if required.

Software development tools for the 8051/8052 can be used directly to develop programs for the MSC12xx.

1.4 Family Compatibility

The MSC12xx family allows the most cost-effective part to be used for each application and ensures a migration path towards larger memories when required. Code written for the 4K byte part runs unaltered on an 8K byte part.

1.5 Flash Memory

The MSC12xx parts feature flexible Flash memory that can be partitioned into program and data areas that are best suited for each application. They may be programmed over the entire operating voltage range and temperature range using serial and self-programming methodologies.

1.6 Internal SRAM

The MSC120x contains a total of 256 bytes of static random access memory (RAM). 128 bytes are directly addressable using instructions that incorporate the address. An additional 128 bytes are indirectly addressable via instructions using a register as a pointer.

1.7 High-Performance Analog Functions

The analog functionality of the MSC120x is state-of-the-art. The ADC is extremely low-noise, and meets the most stringent requirements for analog instrumentation. The integrated programmable gain amplifier (PGA) further improves the ADC performance, which then achieves nanovolt resolution.

The integrated low-drift, high-accuracy voltage reference complements the performance of the ADC and usually eliminates the need for an external reference. However, ratiometric measurements are still possible and easily implemented.

Also present are a programmable filter, an analog multiplexer for single-ended and differential signals, a temperature sensor, burnout current sources, an analog input buffer, an offset DAC, and an 8-bit current DAC.

1.8 High-Performance Peripherals

Additional digital peripherals are included, which offload CPU processing and control functions from the core to improve the overall efficiency further. In particular, there is a 32-bit accumulator closely associated with the ADC, one USART, one serial port (SPI-compatible or basic I^2C), power-on reset, brownout reset, low-voltage detection, two digital ports with configurable I/O, a watchdog timer, two timer/counters, PLL, on-chip oscillator, and external/internal interrupts.

The I^2C and SPI interfaces allow synchronous serial communications with minimal CPU overhead.

The 32-bit accumulator significantly reduces the processing overhead associated with multi-byte data. It allows automatic 32-bit additions from the ADC, and shifts without using CPU registers. 32-bit addition is supported with minimal program interaction.

MSC120x Addressable Resources

This chapter provides a detailed description of the MSC120x addressable resources.

Topic **Page**

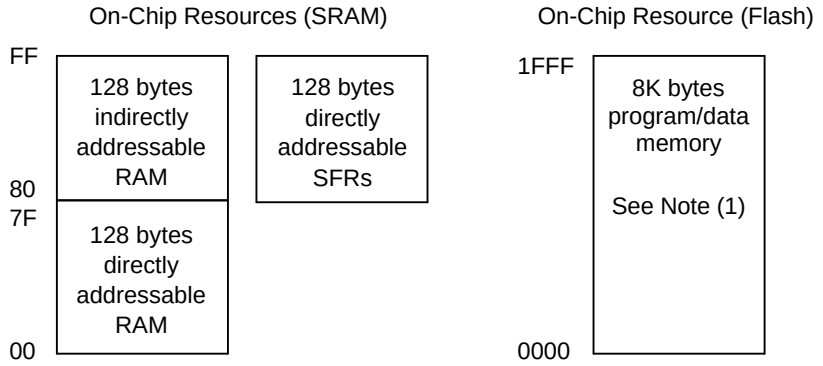
2.1 Introduction

Some microprocessors have a single unified address space that is used for program code, data values and input/output ports. However, most 8051 cores (and thus the MSC120x) have several distinct addressable spaces that serve different purposes, as shown in Figure 2-1. The MSC120x implements all address spaces found in the 8051.

Direct and indirect 8-bit addresses access up to 384 bytes of on-chip resources, comprised of 256 bytes of static random access memory (SRAM) and up to 128 special function registers (SFRs). 16-bit pointers (PC and DPTR) allow up to 8K bytes of program memory to be accessed.

Memory for data may be allocated in different places, depending upon the size of the data, how frequently it is altered, and how efficiently it is accessed. The resources available on the MSC120x are:

- 256 bytes of on-chip SRAM for working registers, bit-wide variables, byte and multi-byte variables, and a stack. This memory is accessed by the majority of data-processing instructions.
- A configurable number of kilobytes of on-chip FLASH memory that is accessed only with MOVX instructions, even though it is on-chip. Typically, data here consists of lookup tables or constant data.



NOTE: (1) 8K bytes of program/data space can be partitioned to program memory and/or data memory; however, data space can be partitioned only up to 4k bytes.

Figure 2-1. On-Chip Memory Resources

2.2 Program Memory and Data Memory

Table 2-1 shows the addresses available associated with program memory and data memory for the Y2 and Y3 versions of the MSC120x.

Table 2-1. Flash Memory Partitioning Addresses

HCR0	MSC120xY2		MSC120xY3	
DFSEL	**Program Memory**	**Data Memory**	**Program Memory**	**Data Memory**
00	0000-07FF (2kB)	0400-0BFF (2kB)	0000-0FFF (4kB)	0400-13FF (4kB)
01	0000-07FF (2kB)	0400-0BFF (2kB)	0000-17FF (6kB)	0400-0BFF (2kB)
10	0000-0BFF (3kB)	0400-07FF (1kB)	0000-1BFF (7kB)	0400-07FF (1kB)
11 (default)	0000-0FFF (4kB)	0000 (0kB)	0000-1FFF (8kB)	0000 (0kB)

In typical 8051 architecture, program memory is read-only. However, in the MSC120x, Flash memory that is allocated to code space can be modified when an instruction such as MOVX @DPTR,A is executed with bit 0 of MWS (SFR 8Fh) set to 1. For more details, see the Program Memory Lock and Reset Sector Lock bits in HCR0. Although modifying code in this way can provide flexibility of design, it is not intended to support repetitive use of self-modifying coding techniques. In addition, a entire page of 64 bytes must be erased before a byte can be change in Flash memory.

The Boot ROM (1K bytes) provides functions to manipulate the Flash memory.

The on-chip Flash memory may be partitioned so that it is shared between code and data spaces. This is done via the three least-significant bits (DFSEL) in HCR0 when the MSC120x is programmed.

1kB of on-chip Boot ROM is used during serial programming modes when it is temporarily mapped to 0000h to 03FFh. During normal program execution, it may be mapped into addresses F800h to FBFFh to provide access to useful routines (for example, serial I/O). This occurs by default via bit 4 of HCR0.

Program memory is accessed in an implicit manner as a program is executed, or by explicit use of the assembly-level MOVC instruction.

Data memory is always accessed via the assembly-level MOVX instruction. Even though this mnemonic stands for MOV eXternal, the memory is internal.

2.3 Program Data Memory and Special Function Registers

The MSC120x has 256 bytes of on-chip SRAM that are closely associated with the core processor, as well as over 80 SFRs, as shown in Table 2-2.

As instructions are executed, the address of SRAM or SFRs is either explicit or implicit, as shown by the examples in Example 2-1.

Table 2-2. On-Chip 8051 Memory

SFR Base (Hex)		**C0**	**C8**	**D0**	**D8**	**E0**	**E8**	**F0**	**F8**	
Bit-Addressable Bit #(1)		*C0-C7*	*C8-CF*	*D0-D7*	*D8-DF*	*E0-E7*	*E8-EF*	*F0-F7*	*F8-FF*	
SFR Base (Hex)		**80**	**88**	**90**	**98**	**A0**	**A8**	**B0**	**B8**	
Bit-Addressable Bit #(1)		*80-87*	*88-8F*	*90-97*	*98-9F*	*A0-A7*	*A8-AF*	*B0-B7*	*B8-BF*	
Designation	**Start Address (Hex)**	**Contect (Hex)**								**End Address (Hex)**
SFRs	80	128 byte space for SFRs; only directly addressable								FF
SRAM	80	128 bytes of SRAM; only indirectly addressable								FF
SRAM	30	80 bytes of SRAM; directly and indirectly addressable								7F
Bit #(1)	28	40-47	48-4F	50-57	58-5F	60-67	68-6F	70-77	78-7F	2F
Bit #(1)	20	00-07	08-0F	10-17	18-1F	20-27	28-2F	30-37	38-3F	27
Register Bank 3(2)	18	R0	R1	R2	R3	R4	R5	R6	R7	1F
Register Bank 2(2)	10	R0	R1	R2	R3	R4	R5	R6	R7	10
Register Bank 1(2)	08	R0	R1	R2	R3	R4	R5	R6	R7	0F
Register Bank 0(2)	00	R0	R1	R2	R3	R4	R5	R6	R7	07

(1) Bit variables numbered 00h to 7Fh are mapped to SRAM bytes 20h to 2Fh. Bit variables numbered 80h to FFh are mapped to SFRs with an address of the form 1xxxx000b. This means that bits within 16 of the 128 possible SFRs may be manipulated by the bit-addressing instructions.

(2) Only R0 or R1 may be used as 8-bit indirect pointers to on-chip SRAM between 00h and FFh.

Example 2-1. Instructions

Instructions	**Condition or Comment**	**Net Effect on SRAM or SFR Location**
MOV R1,4AH	Register bank 1 is active (R1 = 09h)	Contents of RAM at 4Ah is copied to RAM at 09h
MOV @R1,#F4H	Register bank 2 is active and RAM at address 11h contains 8Ah	Immediate code data of F4h is copied to RAM at 8Ah
SETB sync	sync = 5Eh	Bit 6 of RAM at 2Bh is set
PUSH 34H	Stack Pointer (SP) is 9Bh, but is pre-incremented to 9Ch	Contents of RAM at 34h is copied to RAM at 9Ch
POP P0	P0 = 80h, which is the SFR for physical Port 0; Stack Pointer (SP) is 80h and is post decremented to 7Fh	Contents of RAM at 80h is copied to the SFR at 80h
INC P1	P1 = 90h, which is the SFR for physical Port 1	SFR at 90h is incremented
DEC R6	Register bank 0 is active	Contents of RAM at 06h is decremented
CLC C	C = carry = bit 7 of the Program Status Word at D0h	Bit 7 of SFR at D0h is cleared
MUL AB	Accumulator = 12h, Register B = 3Bh	The accumulator = SFR at E0h, becomes the low part of the product (= 26h), and reg B = SFR at F0h becomes the high part (= 04h)
CPL TF1	TF1 = 8Fh; the bit address of timer 1 overflow flag	Complement bit 7 of SFR TCON at 88h
CLC A		The accumulator at SFR E0h is cleared

Special Function Registers

This chapter describes the special function registers of the MSC120x.

Topic **Page**

3.1 Introduction

Special Function Registers (SFRs) are addressable resources within the MSC120x architecture. They can be accessed by a program in several ways:

1. Via instructions with an 8-bit direct address between 80h and FFh. For example, *CLR 90H* clears all bits in port 1 to 0.
2. Bit-addressing instructions with bits in the range 80h and FFh. For example, *SETB 0A8H* enables interrupts from external interrupt 0.
3. By instructions with implicit access. For example, *PUSH 13H* increments the Stack Pointer at SFR address 81h before using it as a pointer to save the content of address 13h, which is Register 3 in bank 2.

The 8-bit addresses of all SFRs are shown in Table 3-1 with respect to a base group at addresses of the form 1xxxx000b. The SFRs in this group are byte-addressable. Shaded SFRs are bit-addressable.

Reading an unassigned SFR will give 00h, while any values written will be ignored. All SFRs are read and written by the processor one byte at a time, even when they are part of a multi-byte value.

Table 3-1. Special Function Register Map(1)(2)

Base	0 (8)	1 (9)	2 (A)	3 (B)	4 (C)	5 (D)	6 (E)	7 (F)
(F8)	EIP	SECINT	MSINT	USEC	MSECL	MSECH	HMSEC	WDTCON
F0	B	PDCON	PASEL		PLLL	PLLH	ACLK	SRST
(E8)	EIE	HWPC0	HWPC1	HDWVER			FMCON	FTCON
E0	ACC	SSCON	SUMR0	SUMR1	SUMR2	SUMR3	ODAC	LVDCON
(D8)	EICON	ADRESL(3)	ADRESM(3)	ADRESH(3)	ADCON0	ADCON1	ADCON2	ADCON3
D0	PSW	OCL	OCM	OCH	GCL	GCM	GCH	ADMUX
(C8)								
C0							EWU	SYSCLK
(B8)	IP							
B0	P3			P3DDRL	P3DDRH	IDAC		
(A8)	IE						P1DDRL	P1DDRH
A0					AIPOL	PAI	AIE	AISTAT
(98)	SCON0	SBUF0	SPICON I2CCON	SPIDATA I2CDATA				
90	P1	EXIF		CADDR	CDATA			
(88)	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	MWS
80		SP	DPL0	DPH0	DPL1	DPH1	DPS	PCON

(1) In general, the low part of multi-byte SFRs (such as the 16-bit pointer comprised of DPL0 and DPH0) reside at adjacent addresses, but this is not always the case; see TL0 (at 8Ah) and TH0 (at 8Ch).

(2) The least significant part of a 16-bit variable is usually at an even address, but this is not always the case; see P3DDRL (at B3h) and P2DDRH (at B4h).

(3) For the MSC1200/01, the ADC result is contained in ADRESH, ADRESM, and ADRESL. For the MSC1202, the ADC result is contained in ADRESM and ADRESL (that is, shifted right one byte) and the MSB is sign-extended (Bipolar mode) or zero-padded (Unipolar mode) in ADRESH. Therefore, when migrating between the MSC1200/01 and MSC1202, the ADC result calculation must be adjusted accordingly. For all devices, the ADC interrupt is cleared by reading ADRESL.

3.2 Referencing SFRs in Assembly and C Languages

When writing programs in assembly language, an SFR can be referenced by its absolute address or by a symbol associated with its address. In C language, a variable must first be declared, as shown in Example 3-1.

Example 3-1. Assembly Code and C Code Comparison

Purpose	Assembly Code[1]			C Code (Compiler-Dependent Directives)[2][3]
Output the character 'A' to serial port 0	SBUF0	DATA	99H	at 0x99 sfr SBUF0;
	MOV	99H,#41H		
	MOV	SBUF0, #41H		SBUF0=0x41;
Enable interrupt for Timer 1	IE	DATA	0A8H	at 0xA8 sfr IE;
	ET1	BIT	IE.3	sbit ET1=IE^3;
	SETB	ET1		ET1=1;
	or			or
	ET1	BIT	0ABH	at 0xAB sbit ET1;
	SETB	ET1		ET1=1;
Set the decimation ratio for the ADC to 3E8h	decimation	DATA	0DEH	at 0xDE sfr16 decimation
	MOV	decimation,#0E8H		decimation=1000;
	MOV	decimation+1,#3		

(1) Indicating a hexadecimal number in assembly language requires a trailing "H" and leading "0" if the first character would otherwise be a letter; for example, 99H or 099H but 0A8H instead of A8H.
(2) In C, a hexadecimal number always starts with 0x; for example, 0x99 and 0xA8.
(3) The keyword "sfr16" cannot be used with TH0:TL0 as Timer0 because the addresses are not adjacent. This is also true for TH1:TL1.

For assembly language programs, declarations that associate common symbols with values are usually grouped in an included file with the name **.inc* that is referenced in the source code. Similarly, for C language, declarations appear in a file with the name **.h*. However, there are some assembly compilers that use **.h* as a file name as well.

3.3 SFR Types

The SFRs belong to functional groups that relate to different aspects of the operation of the MSC120x.

- Port input/output with bit manipulation
- Interrupts
- Integrated peripherals (for example: ADC, SPI, USARTs, or Counter/Timers)
- System functions (for example: power-down, clock generators, and breakpoint registers)
- The core processor architecture (for example: Stack Pointer, Accumulator, and Program Status Word)
- Extensions to the architecture (for example: auxiliary data pointer)

Table 3-2 shows the SFR cross-reference.

Table 3-2. SFR Cross-Reference

SFR	Address	Functions	CPU	IRQ	Ports	Serial Comm.	Power and Clocks	Timer/ Counters	Flash Memory	ADC/ DACs
SP	81h	Stack Pointer	X							
DPL0	82h	Data Pointer Low 0	X							
DPH0	83h	Data Pointer High 0	X							
DPL1	84h	Data Pointer Low 1	X							
DPH1	85h	Data Pointer High 1	X							
DPS	86h	Data Pointer Select	X							
PCON	87h	Power Control					X			
TCON	88h	Timer/Counter Control				X		X		
TMOD	89h	Timer Mode Control				X		X		
TL0	8Ah	Timer0 LSB						X		
TL1	8Bh	Timer1 LSB						X		
TH0	8Ch	Timer0 MSB						X		
TH1	8Dh	Timer1 MSB						X		
CKCON	8Eh	Clock Control				X	X	X		
MWS	8Fh	Memory Write Select							X	
P1	90h	Port 1			X					
EXIF	91h	External Interrupt Flag		X						
CADDR	93h	Configuration Address							X	
CDATA	94h	Configuration Data							X	
SCON0	98h	Serial Port 0 Control				X				
SBUF0	99h	Serial Data Buffer 0				X				
SPICON	9Ah	SPI Control				X				
I2CCON		I^2C Control				X				
SPIDATA	9Bh	SPI Data				X				
I2CDATA		I^2C Data				X				
AIPOL	A4h	Auxiliary Interrupt Poll		X		X	X	X		X
PAI	A5h	Pending Auxiliary Interrupt		X		X	X	X		X
AIE	A6h	Auxiliary Interrupt Enable		X		X	X	X		X
AISTAT	A7h	Auxiliary Interrupt Status		X		X	X	X		X
IE	A8h	Interrupt Enable		X						
P1DDRL	AEh	Port 1 Data Direction Low			X					
P1DDRH	AFh	Port 1 Data Direction High			X					
P3	B0h	Port 3			X					
P3DDRL	B3h	Port 3 Data Direction Low			X					
P3DDRH	B4h	Port 3 Data Direction High			X					
IDAC	B5h	Current DAC								X
IP	B8h	Interrupt Priority		X						
EWU	C6h	Enable Wake Up		X			X			
SYSCLK	C7h	System Clock Divider				X	X	X	X	X
PSW	D0h	Program Status Word	X							
OCL	D1h	ADC Offset Calibration Low Byte								X
OCM	D2h	ADC Offset Calibration Mid Byte								X
OCH	D3h	ADC Offset Calibration High Byte								X
GCL	D4h	ADC Gain Calibration Low Byte								X
GCM	D5h	ADC Gain Calibration Mid Byte								X
GCH	D6h	ADC Gain Calibration High Byte								X
ADMUX	D7h	ADC Input Multiplexer								X
EICON	D8h	Enable Interrupt Control		X		X	X			X
ADRESL	D9h	ADC Results Low Byte								X

Table 3-2. SFR Cross-Reference (continued)

SFR	Address	Functions	CPU	IRQ	Ports	Serial Comm.	Power and Clocks	Timer/ Counters	Flash Memory	ADC/ DACs
ADRESM	DAh	ADC Results Middle Byte								X
ADRESH	DBh	ADC Results High Byte								X
ADCON0	DCh	ADC Control 0								X
ADCON1	DDh	ADC Control 1								X
ADCON2	DEh	ADC Control 2								X
ADCON3	DFh	ADC Control 3								X
ACC	E0h	Accumulator	X							
SSCON	E1h	Summation/Shifter Control	X							X
SUMR0	E2h	Summation 0	X							X
SUMR1	E3h	Summation 1	X							X
SUMR2	E4h	Summation 2	X							X
SUMR3	E5h	Summation 3	X							X
ODAC	E6h	Offset DAC								X
LVDCON	E7h	Low Voltage Detect Control					X			
EIE	E8h	Extended Interrupt Enable		X						
HWPC0	E9h	Hardware Product Code 0	X							
HWPC1	EAh	Hardware Product Code 1	X							
HWVER	EBh	Hardware Version	X							
FMCON	EEh	Flash Memory Control							X	
FTCON	EFh	Flash Memory Timing Control							X	
B	F0h	Second Accumulator	X							
PDCON	F1h	Power Down Control				X	X	X		X
PASEL	F2h	$\overline{\text{PSEN}}$/ALE Select			X		X			
PLLL	F4h	Phase Lock Loop Low					X			
PLLH	F5h	Phase Lock Loop High					X			
ACLK	F6h	Analog Clock					X			
SRST	F7h	System Reset	X				X			
EIP	F8h	Extended Interrupt Priority		X						
SECINT	F9h	Seconds Interrupt		X			X			
MSINT	FAh	Milliseconds Interrupt		X			X			
USEC	FBh	One Microsecond					X		X	
MSECL	FCh	One Millisecond Low					X		X	
MSECH	FDh	One Millisecond High					X		X	
HMSEC	FEh	One Hundred Millisecond					X			
WDTCON	FFh	Watchdog Timer Control	X				X			
Hardware Configuration Regsiters										
HCR0	3Fh	Hardware Configuration Reg. 0							X	
HCR1	3Eh	Hardware Configuration Reg. 1					X			
HCR2	3Fh	Hardware Configuration Reg. 2					X			

3.4 SFR Overview

Table 3-3 lists the SFRs, with addresses and descriptions. Shaded SFR addresses in the table are bit-addressable.

Table 3-3. SFR Overview

Name	Address (Hex)	Description
SP	81h	Stack Pointer SP acts as an 8-bit pointer to core RAM. It creates a last-in/first-out data structure that is used by the instructions PUSH, POP, ACALL, LCALL, RET, RETI, and interrupt calls. The stack is placed in low memory and grows upwards. SP is pre-incremented and post-decremented, and therefore points to the most recent entry on the stack. The default value is 07h, but this is often increased so that additional register banks may be accessed.
DPL0 DPH0	82h 83h	Data Pointer 0 Low (least significant byte) Data Pointer 0 High (most significant byte) DPL0 and DPH0 are read and written independently (except for the instruction MOV DPTR,#data16), but are used together by instructions that reference the 16-bit data pointer called DPTR. DPTR is used to address code and external data by the MOVC and MOVX instructions, respectively. See Data Pointer Select (DPS) at 86h.
DPL1 DPH1	84h 85h	Data Pointer 1 Low (least significant byte) Data Pointer 1 High (most significant byte) DPL1 and DPH1 are read and written independently (except for the instruction MOV DPTR, #data16) but are used together by instructions that reference the16-bit data pointer called DPTR. DPTR is used to address code and `external' data by the MOVC and MOVX instructions, respectively. See Data Pointer Select (DPS) at 86h.
DPS	86h	Data Pointer Select The original 8051 architecture has one DPTR but the MSC120x has two. If bit 0 of DPS is low, DPTR is formed from DPH0:DPL0; otherwise, it is formed by DPH1:DPL1.
PCON	87h	Power Control The core processor may be placed in low power modes by setting the STOP and IDLE bits of this SFR. It also contains two general-purpose flags. There is also a bit called SMOD, which may be used to double the baud rate for serial port 0.This bit is not to be confused with PDCON at F1h, which is used to turn various subsystems on and off.
TCON	88h	Timer Control Bits within TCON control the response to interrupts from Timer/Counters 0 and 1, and external inputs $\overline{INT0}$ and $\overline{INT1}$. Timer/Counters 0 and 1 may also be halted or allowed to run.
TMOD	89h	Timer Mode Configures the modes of operation for Timer/Counters 0 and 1 (for example, whether clocks are internal or external, the number of bits and the reload options). All 8051 Timer/Counters, except for system timers, increment (count up) when they are clocked.
TL0 TH0	8Ah 8Ch	Timer 0 Low Timer 0 High Depending on the mode of operation defined by TMOD at 89h, these SFRs may be considered as independent 8-bit entities, or together as a 13- or 16-bit register. NOTE: These SFRs do not have adjacent addresses and cannot be referenced using the C compiler keyword "sfr16."
TL1 TH1	8Bh 8Dh	Timer 1 Low Timer 1 High Depending on the mode of operation defined by TMOD at 89h, these SFRs may be considered as independent 8-bit entities, or together as a 13- or 16-bit register. NOTE: These SFRs do not have adjacent addresses and cannot be referenced using the C compiler keyword "sfr16."
CKCON	8Eh	Clock Control The original 8051 required 12 system clock pulses per instruction cycle, and each timer had a divide-by-12 prescaler. Since the MSC120x uses only four clocks, three bits within CKCON selectively allow the prescalers of Timers 0, 1, or 2 to be divide-by-12 (default) or divide-by-4. Three other bits determine the number of wait states introduced into the timing of read ($\overline{RD}$ = P3.7) and write ($\overline{WR}$ = P3.6) strobes when the MOVX instruction is used to access off-chip memory.
MWS	8Fh	Memory Write Select When bit 0 is clear (default), any writes to Flash memory via MOVX instructions are written to data space; otherwise, writes are directed to code space. Writing to Flash memory may be inhibited via RSL (bit 5) and PML (bit 6) in HCR0.
P1	90h	Port 1 Controls the byte-wide, bit-programmable input/output called Port 1. Each bit in the SFR corresponds to a pin on the actual part. Individual bits may be configured as bidirectional, CMOS output, open drain output, or input via the Data Direction SFRs for Port 1. See P1DDRL at AEh, and P1DDRH at AFh. Each pin may also be used to provide an alternate function and this may require particular values in P1 and the corresponding data direction bits. For example, P1.4 may be either a general-purpose I/O bit, an input for interrupt INT2 or an active-low Slave Select (input or output) for the SPI interface.

Table 3-3. SFR Overview (continued)

Name	Address (Hex)	Description
EXIF	91h	External Interrupt Flag Four bits represent interrupt flags for interrupts INT2, $\overline{INT3}$, INT4 and $\overline{INT5}$ that must be cleared manually by software. INT2 and INT4 are triggered by a rising edge, while $\overline{INT3}$ and $\overline{INT5}$ respond to a negative edge. If a bit is set in software, an interrupt will occur if the corresponding interrupt is enabled by IE (SFR A8h) and EIE (SFR E8h). See Extended Interrupt Enable (EIE) at E8h, and Extended Interrupt Priority (EIP) at FBh
CADDR	93h	Configuration Address Register The MSC120x contains 128 bytes of Flash memory that may represent hardware configuration data, such as the date of manufacture or any other identification data. This memory is distinct from all other memory addressed by the MSC120x during normal execution of instructions. To access this configuration data, a 7-bit address must first be written to CADDR. See CDATA at 94h.
CDATA	94h	Configuration Data Register Data in the 128 bytes of Flash hardware configuration memory is accessed via this read-only register. The 7-bit address must first be written to CADDR at 093h. NOTE: The instruction reading CDATA must not be in Flash memory itself; otherwise, the data read will be invalid. Typically, instructions will be executed from the internal boot ROM, SRAM that is mapped to code space, or off-chip program memory when reading CDATA.
SCON0	98h	Serial Control 0 Contains six bits that determine the format of data on serial port 0 as well as two bits for transmit and receive interrupt flags. It is used in conjunction with TCON at 88h, TMOD at 89h and various timer data registers.
SBUF0	99h	Serial Buffer 0 When written, SFUF0 provides data for the transmitter associated with serial port 0. When read, data is provided by the receive register. Serial data is output on pin TxD0 and received on pin RxD0.
SPICON I2CCON	9Ah 9Ah	SPI Control If the Serial Peripheral Interface (SPI) is enabled (see bit 0 of PDCON at F1h), SPICON configures SPI communication characteristics such as data rate, clock polarity, and whether the MSC120x is a master or slave. Writing to SPICON resets the counters and pointers used by the SPI interface in FIFO mode. I^2C Control If the I^2C interface is enabled (see bit 5 of PDCON at F1h), I2CCON configures I^2C communication characteristics, such as START, STOP, ACK, clock stretching, and whether the MSC120x is a master or slave. Writing to I2CCON does not reset the I^2C interface.
SPIDATA I2CDATA	9Bh 9Bh	SPI Data If the SPI is enabled (see bit 0 of PDCON at F1h), data written to SPIDATA causes it to be transmitted via the SPI interface, while received data is obtained by reading SPIDATA. I^2C DATA If the I^2C Interface is enabled (see bit 5 of PDCON at F1h), data written to I2CDATA causes it to be transmitted via the I^2C interface, while received data is obtained by reading I2CDATA.
AIPOL	A4h	Auxiliary Interrupt Poll Interrupt status before masking.
PAI	A5h	Pending Auxiliary Interrupt Provides a 4-bit number that corresponds with the hardware priority of the highest pending auxiliary interrupt. All auxiliary interrupts transfer control to location 0033h.
AIE	A6h	Auxiliary Interrupt Enable Bits written determine if a particular auxiliary interrupt is enabled (not masked). In this group are interrupts from the Seconds timer, ADC Summation, ADC, Milliseconds timer. SPI Transmit, SPI Receive/I2C Status, Analog Low-Voltage Detect, and Digital Low Voltage Detect. Bits read indicate the status of each auxiliary interrupt before masking (refer to AIPOL at A4h). EIA, bit 5, of EICON at D8h is a common enable for all auxiliary interrupts. See AISTAT at A7h.
AISTAT	A7h	Auxiliary Interrupt Status When read, AISTAT indicates the status of each auxiliary interrupt after masking. A `1' indicates that an interrupt is pending, while a `0' indicates there is either no interrupt or that it is masked. See AIE at A6h.
IE	A8h	Interrupt Enable Bits written determine if a particular interrupt is enabled (not masked). In this group are enables for Serial port 1, Timer 2, Serial port 0, Timer 1, external $\overline{INT1}$, Timer 0, and external $\overline{INT0}$. Bit 7 is a Global Enable for this group of interrupts. Bits read indicate the status of each enable bit (that is, returns what was previously written).
P1DDRL P1DDRH	AEh AFh	Port 1 Data Direction Low (configures bits 3, 2, 1, and 0 in Port 1) Port 1 Data Direction High (configures bits 7, 6, 5, and 4 in Port 1) Adjacent bits in P1DDRL and P1DDRH control the configuration for pins of Port 1. Standard 8051 bidirectional with weak pull-up is 00, CMOS output is 01, Open drain output is 10 and input only is 11.

Table 3-3. SFR Overview (continued)

Name	Address (Hex)	Description
P3	B0h	Port 3 Controls the byte-wide, bit-programmable input/output called Port 3. Each bit in the SFR corresponds to a pin on the actual part. Individual bits may be configured as bidirectional, CMOS output, open drain output, or input via the Data Direction SFRs for Port 3. See P3DDRL, at B3h, and P1DDRH, at B4h). Each pin can also be used to provide an alternate function and this may require particular values in P3 and the corresponding data direction bits. For example, P3.0 may be either a general-purpose I/O bit, or an input for serial port 0. If EGP23 (bit 0) or EGP0 (bit 1) of HCR1 are 0, or $\overline{EA}$ is 0 when the RST pin is released, P3.6 is an active-low write strobe, and P3.7 an active-low read strobe. These are used in conjunction with ALE and $\overline{PSEN}$ to coordinate access to off-chip memory.
P3DDRL P3DDRH	B3h B4h	Port 3 Data Direction Low (configures bits 3, 2, 1, and 0 in Port 3) Port 3 Data Direction High (configures bits 7, 6, 5, and 4 in Port 3) Adjacent bits in P3DDRL and P3DDRH control the configuration for pins of Port 3. Standard 8051 bidirectional with weak pull-up is 00, CMOS output is 01, Open drain output is 10 and input only is 11. If EGP23 (bit 0) or EGP0 (bit 1) of HCR1 are 0, or $\overline{EA}$ is 0 when the RST pin is released, P3.6 is an active-low write strobe, and P3.7 an active-low read strobe. They are CMOS outputs and P3DDRH bits 4 to 7 have no effect.
IDAC	B5h	Current DAC IDAC represents the 8-bit data value for the current DAC present in the MSC120x. $IDAC_{OUT}$ = IDAC × 3.9µA
IP	B8h	Interrupt Priority Bits within IP correspond in position with those enables in IE at A8h. Each determines if the corresponding interrupt has a low or high priority, using 0 or 1 respectively.
EWU	C6h	Enable Wake-up When the processor has been placed in the IDLE condition by writing a 1 to bit 0 of PCON at 87h, it may be returned to normal operation by an interrupt from either the Watchdog timer, $\overline{INT1}$ or $\overline{INT0}$. Bits 2, 1, and 0 correspond, in order, with these interrupt sources and act as selective enables when set. An auxiliary interrupt can also restore normal operation; this configuration is enabled with EAI, bit 5, of EICON at D8h.
SYSCLK	C7h	System Clock Divider By default, the crystal oscillator is used as the system clock (that is, $f_{CLK} = f_{OSC}$). SYSCLK allows f_{CLK} to be f_{OSC} divided by 1, 2, 4, 8, 16, 32, 1024, 2048, or 4096 and for the change in the divider to be immediate or synchronized with the milliseconds interrupt. The speed of the processor and all other timers that use f_{CLK} will be affected. When f_{CLK} is decreased, the power consumption is reduced.
PSW	D0h	Program Status Word The processor Program Status Word is accessed via PSW. Bits 7 to 0 represent, in order, Carry, Auxiliary Carry, User Flag 0, Register Bank Select 1 and 0, Overflow Flag, User Flag 1, and Parity Flag. PSW is not saved on the stack automatically at the start of an interrupt service routine (ISR) and it is common for each ISR to begin with the instruction PUSH PSW.
OCL OCM OCH	D1h D2h D3h	(ADC) Offset Calibration Low (least significant byte) (ADC) Offset Calibration Middle (ADC) Offset Calibration High (most significant byte) OCH:OCM:OCL represents a 24-bit value that compensates for the offsets within the ADC or system. Usually, values are provided by the ADC subsystem when the ADC is instructed to perform a calibration cycle, but for some applications, the user may provide other values.
GCL GCM GCH	D4h D5h D6h	(ADC) Gain Low (least significant byte) (ADC) Gain Middle (ADC) Gain High (most significant byte) GCH:GCM:GCL represents a 24-bit value that sets the gain of the ADC or system. Usually values are provided by the ADC subsystem when the ADC is instructed to perform a calibration cycle, but for some applications the user may provide other values.
ADMUX	D7h	ADC Multiplexer Register Selects the sources for the positive and negative inputs of the differential delta-sigma ADC. This includes nine pins on the MSC120x and an internal temperature-related source.
EICON	D8h	Enable Interrupt Control EICON contains the enable for the auxiliary interrupts (EAI), the auxiliary interrupt flag (AI), the watchdog timer interrupt flag (WDTI) and a mode bit for serial port 1 (SMOD1), which doubles the baud rate when set.
ADRESL ADRESM ADRESH	D9h DAh DBh	ADC Conversion Results Low (least significant byte) ADC Conversion Results Medium ADC Conversion Results Low High (most significant byte) ADRESH:ADRESM:ADRESL represents the 24-bit, read-only value of the latest ADC conversion. These registers are not updated on an ADC conversion unless ADRESL has been read from the previous result.
ADCON0	DCh	ADC Control 0 Sets the Burnout Detect, Internal/External voltage reference, 1.25V or 2.5V internal reference, V_{REF} clock source, analog buffer, and programmable gain amplifier for the delta-sigma ADC.

Table 3-3. SFR Overview (continued)

Name	Address (Hex)	Description
ADCON1	DDh	ADC Control 1 Sets the polarity, filter type, and conversion mode for the delta-sigma ADC. It also indicates if an overflow or underflow of the summation register has occurred.
ADCON2 ADCON3	DEh DFh	ADC Control 2 ADC Control 3 ADCON3: ADCON2 represent an 11-bit value for the Decimation Ratio of the delta-sigma ADC. The ADC conversion rate is (ACLK+1)/64/Decimation Ratio. See ACLK at F6h.
ACC	E0h	Accumulator The Accumulator is the implicit destination of many operations. Instructions that reference the Accumulator implicitly are always shorter and faster than similar instructions that reference it as an SFR. However, as an SFR it may be used to advantage by instructions such as: JB ACC.2, label, or PUSH ACC.
SSCON	E1h	Summation and Shift Control The result of an ADC conversion is placed in ADRES (D9h to DBh), but may be automatically added to a 32-bit sum represented by SUMR (E2h to E5h). The operation of the summation register is controlled by SSCON, which includes the number of times ADC conversions are added to the sum, and the number of bits that the sum is shifted to the right. There is also a mode where 32-bit values provided by the CPU may be added to, or subtracted from, the summation register when SUMR0 is written. If 00h is written to SSCON, the 32-bit summation register is cleared.
SUMR0 SUMR1 SUMR2 SUMR3	E2h E3h E4h E5h	Summation Register 0 (least significant byte) Summation Register 1 Summation Register 2 Summation Register 3 (most significant byte) SUMR3:SUMR2:SUMR1:SUMR0 represent the 32-bit sum (and optional shift) of a number of ADC conversions. See SSCON at E1h.
ODAC	E6h	(ADC) Offset DAC Register An analog voltage of up to half the range of the ADC is set with an 8-bit DAC and used to offset the input voltage to the ADC. The ODAC register cannot be used to extend the analog inputs beyond their specified input range.
LVDCON	E7h	Low-Voltage Detection Control The voltages present on the analog supply pins may be enabled to generate interrupts if they fall below preset limits. The limits are 2.9V, 3.1V, 3.3V, 3.6V, 3.8V, 4.2V, and 4.6V.
EIE	E8h	Extended Interrupt Enable Provides selective enables for the watchdog timer, $\overline{\text{INT5}}$, INT4, $\overline{\text{INT3}}$, and INT2. See WDTI, bit 3 in EICON at D8h. See External Interrupt Flags, EXIF at 91h.
HWPC0	E9h	Hardware Product Code 0 Refer to the respective data sheet for the code that indicates the type of device.
HWPC1	EAh	Hardware Product Code 1
HDWVER	EBh	Hardware Version Number
FMCON	EEh	Flash Memory Control Three bits to provide control of the Flash memory; specifically, page mode erase, frequency control mode, and busy.
FTCON	EFh	Flash Memory Timing Control The upper four bits (FER) determine the Flash erase time while the lower four bits (FWR) determine the Flash write time, according to the following equations: Erase time = (1 + FER) × (MSECH:MSECL + 1) × t_{CLK} 5 ms (11ms) for commercial (industrial) temperatures Write time = 5 × (1 + FWR) × (USEC + 1) × t_{CLK} The write time should be between 30ms and 40ms. See MSECH and MSECL at FDh and FCh, respectively, and USEC at FBh.
B	F0h	B Register The B register is used only by the instructions MUL AB and DIV AB. If these instructions are not used, B is available to store a byte-wide variable or eight single-bit variables. Caution is needed when programming in C because run-time libraries may use this register.
PDCON	F1h	Power-Down Control Active high bits within PDCON provide selective power-down of IDAC, I^2C, PWM, ADC, Watchdog, System Timer, and SPI subsystems .
PASEL	F2h	$\overline{\text{PSEN}}$/ALE Select If PSEN0 (bit 3 of PASEL) is '0', this pin acts as a general I/O. PSEN1, PSEN2, and PSEN3 (bits 4, 5, and 6 of PASEL) are used to select the output signal for the $\overline{\text{PSEN}}$/ALE pin. Refer to the MSC120x data sheet for more information about the PASEL SFR. If PSEN4 = '0', the output signal is the original signal selected by PSEN1, PSEN2, and PSEN3. If PSEN4 = '1', the output signal is half the frequency of the original signal selected by PSEN1, PSEN2, and PSEN3, and has a 50% duty cycle.

Table 3-3. SFR Overview (continued)

Name	Address (Hex)	Description
PLLL	F4h	Phase Lock Loop Low
PLLH	F5h	Phase Lock Loop High. PLL clock status and PLL lock status.
ACLK	F6h	Analog Clock. The frequency of the delta-sigma ADC modulator is given by: $f_{CLK}/(ACLK + 1) \times 64$ where f_{CLK} is the frequency of the system clock.
SRST	F7h	System Reset Register If bit 0 is set high then low, the MSC120x will be reset. This reset causes exactly the same behavior as if a system reset had been initiated by the RST pin.
EIP	F8h	Extended Interrupt Priority Five bits determine the priority for interrupts: Watchdog, $\overline{INT5}$, INT4, $\overline{INT3}$, and INT2. See Extended Interrupt Flags, EXIF, at 91h and Extended Interrupt Enable, EIE, at E8h.
SECINT	F9h	Seconds Timer Interrupt The seconds interrupt, if enabled, occurs at an interval given by: $(SECINT + 1) \times (HMSEC + 1) \times (MSEC + 1) \times t_{CLK}$ If bit 7 is set when SECINT is written, the SECINT value will be loaded into the counter immediately; otherwise, it will be delayed until the current count expires. When the associated down-counter reaches zero, it is reloaded with the value in SECINT. See ESEC, bit 7 of AIE at A6h.
MSINT	FAh	Milliseconds Interrupt The milliseconds interrupt, if enabled, occurs with an interval given by: $(MSINT + 1) \times (MSEC + 1) \times t_{CLK}$ If bit 7 is set when MSINT is written, the MSINT value will be loaded into the counter immediately; otherwise, it will be delayed until the current count expires. When the associated down-counter reaches zero, it is reloaded with the value in MSINT. See EMSEC, bit 4 of AIE at A6h.
USEC	FBh	Microsecond Register The internal microseconds clock has a period given by: $(USEC + 1) \times t_{CLK} = (USEC + 1)/f_{CLK}$ When the associated down-counter reaches zero, it is reloaded with the value in USEC. See FTCON at EFh.
MSECL MSECH	FCh FDh	Millisecond Low Millisecond High MSECH:MSECL together represent MSEC, which determines the time between millisecond interrupts given by: $(MSECH \times 256 + MSECL + 1) \times t_{CLK}$ When the associated down-counter reaches zero, it is reloaded with the value in MSECH:MSECL.
HMSEC	FEh	Hundred Millisecond Clock The hundred milliseconds counter has a period given by: $(HMSEC + 1) \times (MSECH \times 256 + MSECL + 1) \times t_{CLK}$ When the associated down-counter reaches zero, it is reloaded with the value in HMSEC.
WDTCON	FFh	Watchdog Timer Control Once enabled, the watchdog timer expires after a delay of: $(WDCNT + 1) \times HMSEC$ to $(WDCNT + 2) \times HMSEC$ Writing a 1, then 0 sequence to bit 7, bit 6, or bit 5 enables, disables, or restarts the watchdog timer, respectively. If the associated down-counter reaches zero, a watchdog timeout occurs. By default, this timeout causes a reset, but EWDR (bit 3) in HCR0 may disable the reset and trigger an interrupt instead. When the watchdog is enabled, the counter must be repeatedly restarted before it reaches zero.

Programmer's Model and Instruction Set

This chapter describes the programmer's model and instruction set for the MSC120x.

Topic **Page**

4.1 Introduction

The MSC120x incorporates a microcontroller with the same instruction set as the industry-standard 8051; however, for a given external clock, it executes up to three times more quickly. This increased rate is because the MSC120x is based on a machine cycle of four clocks rather than the original 12.

Although the most frequently used instructions are three times faster, the aggregate speed improvement for programs is about 2.5 times faster.

4.2 Registers

The MSC120x manipulates data via a single 8-bit accumulator (A), together with eight 8-bit registers (R0 to R7). The result of all arithmetic and logical operations is placed in the accumulator, which can then be copied to R*n*, on-chip memory, or off-chip memory, by various MOV instructions.

To the programmer, the MSC120x may be modelled as shown in Table 4-1.

Table 4-1. 8051 Working Registers

Name	Bit															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSW									CY	AC	F0	RS1	RS0	OV	F1	P
B									Register B							
R*n*									Register *n* (where *n* = 0 to 7)							
A									Accumulator							
DPL									Data Pointer Low							
DPH									Data Pointer High							
SP									Stack Pointer							
DPTR	Data Pointer High								Data Pointer Low							
PC	Program Counter															

The Data Pointer (DPTR) is composed of two 8-bit SFRs accessed as separate bytes. However, it is used implicitly by some instructions as a 16-bit pointer to either program or data memory. Two different data pointers can be selected for use by these instructions.

The Stack Pointer (SP) is used to support a first-in/first-out data structure within data memory. When referenced implicitly as an 8-bit pointer, it is pre-incremented and post-decremented, which means that the stack grows upwards and SP always points to the most recent entry. The stack can store either data values or addresses.

The default value for SP is 7, so it starts to grow just above memory associated with address bank 0 at data memory locations 00h to 07h. Since address bank 1 occupies locations 08h to 0Fh, care must be taken to redefine the initial value of SP whenever register bank 1 (or 2 or 3) is to be used. The selection of the active register bank is determined by bits 4 and 3 of the Program Status Word (PSW). For example, when RS1 = 1 and RS0 = 1, bank 3 is active and R2 corresponds with data memory location 1Ah.

PSW also contains the Carry flag (CY), Auxiliary Carry (AC), General-purpose flags F1 and F0, as well as the Overflow flag (OV) and the Parity flag (P). Some instructions change the flags, but the majority do not.

Register B is sometimes useful to store a byte-wide variable or 8 bit-wide variables, especially for applications written entirely in assembly language. Care is needed because it is used by MUL and DIV instructions that may be called by C run-time libraries.

The 16-bit program counter (PC) is incremented as sequential instructions are executed. For jumps, it is loaded with a new value, and for CALLs and interrupts, it is stored to the stack for recovery during RETurns and RETI. It always points to a byte in program memory and has a reset value of 0000h.

4.3 Instruction Types and Addressing Modes

MSC120x instruction types are shown in Example 4-1.

For each type of instruction, there may be more than one mode of addressing. For example, there are four different modes associated with the ADD instruction, as shown in Example 4-2.

The MOV instruction has the greatest number of combinations of addressing modes, with special variants such as MOVX and MOVC.

Table 4-3 shows the MSC120x op-codes.

Table 4-4 shows all instructions with their mnemonic, description, flags, cycles, clocks, and op-code. If the exact operation is unclear, the reader is referred to any of the numerous data sheets and books for the 8051 that are generally available.

Example 4-1. Instruction Types

Type	Examples		
Simple data movement	MOV A,R5	MOV R4,#0A3H	MOV P1,A
Data movement	MOV A,@R1	MOVX A,@DPTR	PUSH PSW
Data processing	DEC R3	ADDC A,10H	ORL P2,#5
Bit operations	CLR C	SETB 084H	ANL C,/F0
Program flow	LCALL *16-bit address*	SJMP *relative address*	CJNE A,#4, *address*
Miscellaneous	DJNZ R4, *relative address*	MUL	DA

Example 4-2. Instruction Addressing Modes

Assembly Level Instruction	Addressing Mode	Action	Hex Operation Code(s)
ADD A,#0C3H	Immediate	The code byte at PC + 1 (that is, C3h) is added to A.	24 C3h
ADD A,@R1	Indirect	The contents of Register 1 provide the 8-bit address of the data in memory that is added to A.	27h
ADD A,P0	Direct	The code byte at PC + 1 provides the 8-bit address of the data in memory that is added to A. In this case, SFR P1 at 90h.	25 90h
ADD A,R4	Register	The contents of Register 4 is added to the accumulator. The memory location corresponding to register 4 is either 04h, 0Ch, 14h, or 1Ch depending on the register bank select bits in PSW.	2Ch

Table 4-2. Symbol Descriptions for Instruction List of Table 4-3

Symbol	Description
A	Accumulator
R*n*	Register R0-R7 of the current register bank.
direct	Internal core address. RAM (00h-7Fh) or SFR (80h-FFh).
@R*i*	R0 or R1 act as an 8-bit pointer to internal core RAM (00h-FFh), except MOVX references external data space.
rel	Two's complement offset byte (-128 to +127) relative to the start address of the next sequential instruction.
bit	Direct bit address. Bits 00h-7Fh map to RAM while 80h-FFh map to SFRs.
#data	8-bit immediate constant.
#data16	16-bit immediate constant.
addr16	16-bit destination address anywhere within program memory address space.
addr11	11-bit destination address anywhere within the current 2K page of program memory.

Table 4-3. MSC120x Op-Codes

Table Cell Contents:		op code / bytes/cycles / instruction / operand(s)		Operand Definitions: addr11: 11-bit address addr16: 16-bit address bit: addressable bit		dir: direct address #d8: 8-bit immediate data #d16: 16-bit immediate data rel8: 8-bit relative address	
00 1/1 NOP	01 2/3 AJMP addr11	02 3/4 LJMP addr16	03 1/1 RR A	04 1/1 INC A	05 2/2 INCdir	06 1/1 INC @R0	07 1/1 INC @R1
10 3/4 JBC bit,rel8	11 2/3 ACALL addr11	12 3/4 LCALL addr16	13 1/1 RRC A	14 1/1 DEC A	15 2/2 DEC dir	16 1/1 DEC @R0	17 1/1 DEC @R1
20 3/4 JB bit,rel8	21 2/3 AJMP addr11	22 1/4 RET	23 1/1 RL A	24 2/2 ADD A,#d8	25 2/2 ADD A,dir	26 1/1 ADD A.@R0	27 1/1 ADD A,@R1
30 3/4 JNB bit,rel8	31 2/3 ACALL addr11	32 1/4 RETI	33 1/1 RLC A	34 2/2 ADDC A,#d8	35 2/2 ADDC A,dir	36 1/1 ADDC A.@R0	37 1/1 ADDC A,@R1
40 2/3 JC rel8	41 2/3 AJMP addr11	42 2/2 ORL dir,A	43 3/3 ORL dir,#d8	44 2/2 ORL A,#d8	45 2/2 ORL A,dir	46 1/1 ORL A.@R0	47 1/1 ORL A,@R1
50 2/3 JNC rel8	51 2/3 ACALL addr11	52 2/2 ANL dir,A	53 3/3 ANL dir,#d8	54 2/2 ANL A,#d8	55 2/2 ANL A,dir	56 1/1 ANL A.@R0	57 1/1 ANL A,@R1
60 2/3 JZ rel8	61 2/3 AJMP addr11	62 2/2 XRL dir,A	63 3/3 XRL dir,#d8	64 2/2 XRL A,#d8	65 2/2 XRL A,dir	66 1/1 XRL A,@R0	67 1/1 XRL A,@R1
70 2/3 JNZ rel8	71 2/3 ACALL addr11	72 2/2 ORL C,bit	73 1/3 JMP @A+DPTR	74 2/2 MOV A,#d8	75 3/3 MOV dir,#d8	76 2/2 MOV @R0,#d8	77 2/2 MOV @R1,#d8
80 2/3 SJMP rel8	81 2/3 AJMP addr11	82 2/2 ANL C,bit	83 1/3 MOVC A,@A+PC	84 1/5 DIV AB	85 3/3 MOV dir,dir	86 2/2 MOV dir,@R0	87 2/2 MOV dir,@R1
90 3/3 MOV DPTR,#d16	91 2/3 ACALL addr11	92 2/2 MOV bit,C	93 1/3 MOVC A,@A+DPTR	94 2/2 SUBB A,#d8	95 2/2 SUBB A,dir	96 1/1 SUBB A,@R0	97 1/1 SUBB A,@R1
A0 2/2 ORL C,/bit	A1 2/3 AJMP addr11	A2 2/2 MOV C,bit	A3 1/3 INC DPTR	A4 1/5 NUL AB	A5 1/1 Reserved	A6 2/2 MOV @R0,dir	A7 2/2 MOV @R1,dir
B0 2/2 ANL C,/bit	B1 2/3 ACALL addr11	B2 2/2 CPL bit	B3 1/1 CPL C	B4 3/4 CJNE A,#d8,rel8	B5 3/4 CJNE A,dir,rel8	B6 3/4 CJNE @R0,#d8,rel8	B7 3/4 CJNE @R1,#d8,rel8
C0 2/2 PUSH dir	C1 2/3 AJMP addr11	C2 2/2 CLR bit	C3 1/1 CLR C	C4 1/1 SWAP A	C5 2/2 XCH A,dir	C6 1/1 XCH A,@R0	C7 1/1 XCH A,@R1
D0 2/2 POP dir	D1 2/3 ACALL addr11	D2 2/2 SETB bit	D3 1/1 SETB C	D4 1/1 DA A	D5 3/4 DJNZ dir,rel8	D6 1/1 XCHD A,@R0	D7 1/1 XCHD A,@R1
E0 1/2-9(1) MOVX A,@DPTR	E1 2/3 AJMP addr11	E2 1/2-9(1) MOVX A,@R0	E3 1/2-9(1) MOVX A,@R1	E4 1/1 CLR A	E5 2/2 MOV A,diR	E6 1/1 MOV A,@R0	E7 1/1 MOV A,@R1
F0 1/2-9(1) MOVX @DPTR,A	F1 2/3 ACALL addr11	F2 1/2-9(1) MOVX @R0,A	F3 1/2-9(1) MOVX @R1,A	F4 1/1 CPL A	F5 2/2 MOV dir,A	F6 1/1 MOV @R0,A	F7 1/1 MOV @R1,A

(1) Number of cycles is user-selectable; see SFR CKCON at 8Eh.

Table 4-3. MSC120x Op-Codes (continued)

Table Cell Contents: op code / bytes/cycles / instruction / operand(s)							
Operand Definitions: **addr11: 11-bit address** **addr16: 16-bit address** **bit: addressable bit**				**dir: direct address** **#d8: 8-bit immediate data** **#d16: 16-bit immediate data** **rel8: 8-bit relative address**			
08 1/1 INC R0	09 1/1 INC R1	0A 1/1 INC R2	0B 1/1 INC R3	0C 1/1 INC R4	0D 1/1 INC R5	0E 1/1 INC R6	0F 1/1 INC R7
18 1/1 DEC R0	19 1/1 DEC R1	1A 1/1 DEC R2	1B 1/1 DEC R3	1C 1/1 DEC R4	1D 1/1 DEC R5	1E 1/1 DEC R6	1F 1/1 DEC R7
28 1/1 ADD A,R0	29 1/1 ADD A,R1	2A 1/1 ADD A,R2	2B 1/1 ADD A,R3	2C 1/1 ADD A,R4	2D 1/1 ADD A,R5	2E 1/1 ADD A,R6	2F 1/1 ADD A,R7
38 1/1 ADDC A,R0	39 1/1 ADDC A,R1	3A 1/1 ADDC A,R2	3B 1/1 ADDC A,R3	3C 1/1 ADDC A,R4	3D 1/1 ADDC A,R5	3E 1/1 ADDC A,R6	3F 1/1 ADDC A,R7
48 1/1 ORL A,R0	49 1/1 ORL A,R1	4A 1/1 ORL A,R2	4B 1/1 ORL A,R3	4C 1/1 ORL A,R4	4D 1/1 ORL A,R5	4E 1/1 ORL A,R6	4F 1/1 ORL A,R7
58 1/1 ANL A,R0	59 1/1 ANL A,R1	5A 1/1 ANL A,R2	5B 1/1 ANL A,R3	5C 1/1 ANL A,R4	5D 1/1 ANL A,R5	5E 1/1 ANL A,R6	5F 1/1 ANL A,R7
68 1/1 XRL A,R0	69 1/1 XRL A,R1	6A 1/1 XRL A,R2	6B 1/1 XRL A,R3	6C 1/1 XRL A,R4	6D 1/1 XRL A,R5	6E 1/1 XRL A,R6	6F 1/1 XRL A,R7
78 2/2 MOV R0,#d8	79 2/2 MOV R1,#d8	7A 2/2 MOV R2,#d8	7B 2/2 MOV R3,#d8	7C 2/2 MOV R4,#d8	7D 2/2 MOV R5,#d8	7E 2/2 MOV R6,#d8	7F 2/2 MOV R7,#d8
88 2/2 MOV dir,R0	89 2/2 MOV dir,R1	8A 2/2 MOV dir,R2	8B 2/2 MOV dir,R3	8C 2/2 MOV dir,R4	8D 2/2 MOV dir,R5	8E 2/2 MOV dir,R6	8F 2/2 MOV dir,R7
98 1/1 SUBB A,R0	99 1/1 SUBB A,R1	9A 1/1 SUBB A,R2	9B 1/1 SUBB A,R3	9C 1/1 SUBB A,R4	9D 1/1 SUBB A,R5	9E 1/1 SUBB A,R6	9F 1/1 SUBB A,R7
A8 2/2 MOV R0,dir	A9 2/2 MOV R1,dir	AA 2/2 MOV R2,dir	AB 2/2 MOV R3,dir	AC 2/2 MOV R4,dir	AD 2/2 MOV R5,dir	AE 2/2 MOV R6,dir	AF 2/2 MOV R7,dir
B8 3/4 CJNE R0,#d8,rel8	B9 3/4 CJNE R1,#d8,rel8	BA 3/4 CJNE R2,#d8,rel8	BB 3/4 CJNE R3,#d8,rel8	BC 3/4 CJNE R4,#d8,rel8	BD 3/4 CJNE R5,#d8,rel8	BE 3/4 CJNE R6,#d8,rel8	BF 3/4 CJNE R7,#d8,rel8
C8 1/1 XCH A,R0	C9 1/1 XCH A,R1	CA 1/1 XCH A,R2	CB 1/1 XCH A,R3	CC 1/1 XCH A,R4	CD 1/1 XCH A,R5	CE 1/1 XCH A,R6	CF 1/1 XCH A,R7
D8 2/3 DJNZ R0,rel8	D9 2/3 DJNZ R1,rel8	DA 2/3 DJNZ R2,rel8	DB 2/3 DJNZ R3,rel8	DC 2/3 DJNZ R4,rel8	DD 2/3 DJNZ R5,rel8	DE 2/3 DJNZ R6,rel8	DF 2/3 DJNZ R7,rel8
E8 1/1 MOV A,R0	E9 1/1 MOV A,R1	EA 1/1 MOV A,R2	EB 1/1 MOV A,R3	EC 1/1 MOV A,R4	ED 1/1 MOV A,R5	EE 1/1 MOV A,R6	EF 1/1 MOV A,R7
F8 1/1 MOV R0,A	F9 1/1 MOV R1,A	FA 1/1 MOV R2,A	FB 1/1 MOV R3,A	FC 1/1 MOV R4,A	FD 1/1 MOV R5,A	FE 1/1 MOV R6,A	FF 1/1 MOV R7,A

Table 4-4. Instruction List

Mnemonic	Description	Flags[1]			Bytes	MSC120x Cycles	MSC120x Clocks	8051 Clocks	Code (Hex)
		CY	AC	OV					
Arithmetic									
ADD A,R*n*	Add register to A	X	X	X	1	1	4	12	28-2F
ADD A,direct	Add direct byte to A	X	X	X	2	2	8	12	25
ADD A,@R*i*	Add indirect data memory to A	X	X	X	1	1	4	12	26-27
ADD A,#data	Add immediate data to A	X	X	X	2	2	8	12	24
ADDC A,R*n*	Add register to A with carry	X	X	X	1	1	4	12	38-3F
ADDC A,direct	Add direct byte to A with carry	X	X	X	2	2	8	12	35
ADDC A,@R*i*	Add indirect data memory to A with carry	X	X	X	1	1	4	12	36-37
ADDC A,#data	Add immediate data to A with carry	X	X	X	2	2	8	12	34
SUBB A,R*n*	Subtract register from A with borrow	X	X	X	1	1	4	12	98-9F
SUBB A,direct	Subtract direct byte from A with borrow	X	X	X	2	2	8	12	95
SUBB A,@R*i*	Subtract indirect data memory from A with borrow	X	X	X	1	1	4	12	96-97
SUBB A,#data	Subtract immediate data from A with borrow	X	X	X	2	2	8	12	94
INC A	Increment A	-	-	-	1	1	4	12	04
INC R*n*	Increment register	-	-	-	1	1	4	12	08-0F
INC direct	Increment direct byte	-	-	-	2	2	8	12	05
INC @R*i*	Increment indirect data memory	-	-	-	1	1	4	12	06-07
DEC A	Decrement A	-	-	-	1	1	4	12	14
DEC R*n*	Decrement register	-	-	-	1	1	4	12	18-1F
DEC direct	Decrement direct byte	-	-	-	2	2	8	12	15
DEC @R*i*	Decrement indirect data memory	-	-	-	1	1	4	12	16-17
INC DPTR	Increment 16-bit data pointer	-	-	-	1	3	12	24	A3
MUL AB	Multiply A by B	0	-	X	1	5	20	48	A4
DIV AB	Divide A by B	0	-	X	1	5	20	48	84
DA A	Decimal adjust A to give 2 BCD nibbles. Used after ADD or ADDC.	X	-	-	1	1	4	12	D4
Logical									
ANL A,R*n*	AND register to A	-	-	-	1	1	4	12	58-5F
ANL A,direct	AND direct byte to A	-	-	-	2	2	8	12	55
ANL A,@R*i*	AND indirect data memory to A	-	-	-	1	1	4	12	56-57
ANL A,#data	AND immediate data to A	-	-	-	2	2	8	12	54
ANL direct,A	AND A to direct byte	-	-	-	2	2	8	12	52
ANL direct,#data	AND immediate data to direct byte	-	-	-	3	3	12	24	53
ORL A,R*n*	OR register to A	-	-	-	1	1	4	12	48-4F
ORL A,direct	OR direct byte to A	-	-	-	2	2	8	12	45
ORL A,@R*i*	OR indirect data memory to A	-	-	-	1	1	4	12	46-47
ORL A,#data	OR immediate data to A	-	-	-	2	2	8	12	44
ORL direct,A	OR A to direct byte	-	-	-	2	2	8	12	42
ORL direct,#data	OR immediate data to direct byte	-	-	-	3	3	12	24	43
XRL A,R*n*	Exclusive OR register to A	-	-	-	1	1	4	12	68-6F
XRL A,direct	Exclusive OR direct byte to A	-	-	-	2	2	8	12	65
XRL A,@R*i*	Exclusive OR indirect data memory to A	-	-	-	1	1	4	12	66-67
XRL A,#data	Exclusive OR immediate data to A	-	-	-	2	2	8	12	64
XRL direct,A	Exclusive OR A to direct byte	-	-	-	2	2	8	12	62
XRL direct,#data	Exclusive OR immediate data to direct byte	-	-	-	3	3	12	24	63
CLR A	Clear A	-	-	-	1	1	4	12	E4

(1) Flags CY, AC, and OV may also be changed by explicit writes to corresponding bits in the PSW.

Table 4-4. Instruction List (continued)

Mnemonic	Description	Flags[1]			Bytes	MSC120x Cycles	MSC120x Clocks	8051 Clocks	Code (Hex)
		CY	AC	OV					
CPL A	Complement A	-	-	-	1	1	4	12	F4
RL A	Rotate A left	-	-	-	1	1	4	12	23
RLC A	Rotate A left through carry	X	-	-	1	1	4	12	33
RR A	Rotate A right	-	-	-	1	1	4	12	03
RRC A	Rotate A right through carry	X	-	-	1	1	4	12	13
SWAP A	Swap nibbles of A	-	-	-	1	1	4	12	C4
Data Movement									
MOV A,R*n*	Move register to A	-	-	-	1	1	4	12	E8-EF
MOV A,direct	Move direct byte to A	-	-	-	2	2	8	12	E5
MOV A,@R*i*	Move indirect data memory to A	-	-	-	1	1	4	12	E6-E7
MOV A,#data	Move immediate data to A	-	-	-	2	2	8	12	74
MOV R*n*,A	Move A to register	-	-	-	1	1	4	12	F8-FF
MOV R*n*,direct	Move direct byte to register	-	-	-	2	2	8	24	A8-AF
MOV R*n*,#data	Move immediate data to register	-	-	-	2	2	8	12	78-7F
MOV direct,A	Move A to direct byte	-	-	-	2	2	8	12	F5
MOV direct,R*n*	Move register to direct byte	-	-	-	2	2	8	24	88-8F
MOV direct,direct	Move direct byte to direct byte	-	-	-	3	3	12	24	85
MOV direct,@R*i*	Move indirect data memory to direct byte	-	-	-	2	2	12	24	86-87
MOV direct,#data	Move immediate data to direct byte	-	-	-	3	3	12	24	75
MOV @R*i*,A	MOV A to indirect data memory	-	-	-	1	1	4	12	F6-F7
MOV @R*i*,direct	Move direct byte to indirect data memory	-	-	-	2	2	8	24	A6-A7
MOV @R*i*,#data	Move immediate data to indirect data memory	-	-	-	2	2	8	12	76-77
MOV DPTR,#data	Move 2 bytes of immediate data to data pointer	-	-	-	3	3	12	24	90
MOVC A,@A+DPTR	Move a byte in code space A (unsigned) after DPTR to A	-	-	-	1	3	12	24	93
MOVC A,@A+PC	Move a byte in code space A (unsigned) after from the address of the next instruction to A	-	-	-	1	3	12	24	83
MOVX A,@R*i*	Move external data (A8) to A	-	-	-	1	2-9[2]	8	12	E2-E3
MOVX A,@DPTR	Move external data (A16) to A	-	-	-	1	2-9[2]	8	24	E0
MOVX @R*i*,A	Move A to external data. Upper 8-bit address comes from MPAGE SFR.	-	-	-	1	2-9[2]	8	24	F2-F3
MOVX @DPTR,A	Move A to external data memory	-	-	-	1	2-9[2]	8	24	F0
PUSH direct	Push direct byte onto stack	-	-	-	2	2	8	24	C0
POP direct	Pop direct byte from stack	-	-	-	2	2	8	24	D0
XCH A,R*n*	Exchange A and register	-	-	-	1	1	4	12	C8-CF
XCH A,direct	Exchange A and direct byte	-	-	-	2	2	8	12	C5
XCH A,@R*i*	Exchange A and indirect data memory	-	-	-	1	1	4	12	C6-C7
XCHD A,@R*i*	Exchange A and indirect data memory nibble in bits 3-0	-	-	-	1	1	4	12	D6-D7
Boolean									
CLR C	Clear carry	0	-	-	1	1	4	12	C3
CLR bit	Clear direct bit	-	-	-	2	2	8	12	C2
SETB C	Set carry	1	-	-	1	1	4	12	D3
SETB bit	Set direct bit	-	-	-	2	2	8	12	D2
CPL C	Complement carry	X	-	-	1	1	4	12	B3
CPL bit	Complement direct bit	-	-	-	2	2	8	12	B2
ANL C,bit	AND direct bit to carry	X	-	-	2	2	8	24	82
ANL C,/bit	AND inverse of direct bit to carry	X	-	-	2	2	8	24	B0

(2) Number of cycles is user-selectable; see SFR CKCON at 8Eh.

Table 4-4. Instruction List (continued)

Mnemonic	Description	Flags[1]			Bytes	MSC120x Cycles	MSC120x Clocks	8051 Clocks	Code (Hex)
		CY	AC	OV					
ORL C,bit	OR direct bit to carry	X	-	-	2	2	8	24	72
ORL C,/bit	OR inverse of direct bit to carry	X	-	-	2	2	8	24	A0
MOV C,bit	Move direct bit to carry	X	-	-	2	2	8	12	A2
MOV bit,C	Move carry to direct bit	-	-	-	2	2	8	24	92
Branching									
ACALL addr11	Absolute call to subroutine within current page.	-	-	-	2	3	12	24	11-F1
LCALL addr16	Long call to subroutine; PC becomes addr16	-	-	-	3	4	16	24	12
RET	Return from subroutine	-	-	-	1	4	16	24	22
RETI	Return from interrupt	-	-	-	1	4	16	24	32
AJMP addr11	Absolute unconditional jump within current page.	-	-	-	2	3	12	24	01-E1
LJMP addr16	Long jump; PC becomes addr16	-	-	-	3	4	16	24	02
SJMP rel	Unconditional relative jump	-	-	-	2	3	12	24	80
JC rel	Jump relative if carry is 1	-	-	-	2	3	12	24	40
JNC rel	Jump relative if carry is 0	-	-	-	2	3	12	24	50
JB bit,rel	Jump relative if direct bit is 1	-	-	-	3	4	16	24	20
JNB bit,rel	Jump relative if direct bit is 0	-	-	-	3	4	16	24	30
JBC bit,rel	Jump relative if direct bit is 1 and clear the bit	-	-	-	3	4	16	24	10
JMP @A+DPTR	Jump indirect. Program counter becomes DPTR plus A.	-	-	-	1	3	12	24	73
JZ rel	Jump relative if accumulator is 00h.	-	-	-	2	3	12	24	60
JNZ rel	Jump relative if accumulator is not 00h.	-	-	-	2	3	12	24	70
CJNE A,direct,rel	Compare A with direct data and jump relative if not equal.	X	-	-	3	4	16	24	B5
CJNE A,#data,rel	Compare A with immediate data and jump relative if not equal	X	-	-	3	4	16	24	B4
CJNE R*n*,#data,rel	Compare register with immediate data and jump relative if not equal.	x	-	-	3	4	16	24	B8-BF
CJNE @R*i*,#data,rel	Compare indirect with immediate data and jump relative if not equal.	x	-	-	3	4	16	24	B6-B7
DJNZ Rn,rel	Decrement register and jump relative if not 0.	-	-	-	2	3	12	24	D8-DF
DJNZ direct,rel	Decrement direct byte and jump relative if not 0.	-	-	-	3	4	16	24	D5
Miscellaneous									
NOP	No operation	-	-	-	1	1	4	12	00
Reserved	No operation	-	-	-	1	1	4	12	A5

4.4 Examples of MSC120x Instructions

For a particular application, suppose it is required to compute the logical function:

Q = (W & X) + Y + not(Z)

given a byte where:

Q is bit 7 of port 1,

and W is bit 0, X is bit 1, Y is bit 2, and Z is bit 3, all of the accumulator.

The assembly code listed below shows how this can be achieved in a number of different ways and allows the reader to see the application of many different types of instructions.

Example 4-3. Assembly Code

```
; Assembly Language Example
$include (reg1200.inc)
W bit ACC.0
X bit ACC.1
Y bit ACC.2
Z bit ACC.3
Q bit P1.7
CSEG AT 0100H
main:      mov R7,#0 ;initial value
main_1:    lcall fun1             ; decision tree
           lcall fun2             ; 'better' tree ?
           lcall fun3             ; boolean operations
           lcall fun4             ; look-up table
           lcall fun5             ; faster
           lcall fun6             ; fastest
           inc R7
           cjne R7,#10H,main_1    ; try values 00 to 0F
           sjmp main
;Number of clock cycles:(MSC120x:120) (8051:204) Ratio=1.7
fun1:      mov A,R7               ; get input values W, X, Y, Z
           anl A,#8h              ; select' Z
           cjne A,#0,fun1_1       ; test for Z = 0
           sjmp fun1_setQ         ; set Q=1 because Z=0
fun1_1:    mov A,R7               ; recover input values
           anl A,#4               ; select Y
           cjne A,#4,fun1_2       ; test for Y = 1
           sjmp fun1_setQ         ; set Q=1 because Y = 1
fun1_2:    mov A,R7               ; recover input values
           anl A,#3h              ; select W, X
           cjne A,#3,fun1_clrQ    ; test for W = X = 1
           sjmp fun1_setQ
fun1_clrQ: clr  Q                 ; clear Q
           sjmp fun1_z
fun1_setQ: setb Q                 ; set Q
fun1_z:    ret

;Number of clock cycles:(MSC120x:114) (8051:252) Ratio=2.2
fun2:      mov A,R7               ; get input values Z,Y,X,W
           anl A,#8               ; select Z
           jz fun2_setQ           ; set Q because Z = 0
           mov A,R7               ; recover inputs
           anl A,#4               ; select Y
           jnz fun2_setQ          ; set Q because Y = 1
           mov A,R7               ; recover inputs
           rrc A                  ; W into carry
           mov R0,A               ; X in bit #0
           rlc A                  ; recover W
           anl A,R0               ; AND with X
           anl A,#1               ; get just W&X
           jnz fun1_setQ          ; set Q because W&X = 1
           clr  Q                 ; Q=0
```

Example 4-3. Assembly Code (continued)

```
           sjmp fun2_z
fun2_setQ: setb Q                  ; Q=1
fun2_z:    ret

;Number of clock cycles:(MSC120x:60) (8051:144) Ratio=2.4
fun3:      mov A,R7                ; get input values Z,Y,X,W
           mov C,W                 ; Carry = W
           anl C,X                 ; Carry = W&X
           orl C,Y                 ; Carry = W&X + Y
           orl C,/Z                ; Carry = W&X + Y + /Z
           mov Q,C                 ; Output new Q value
           ret

;Number of clock cycles:(MSC120x:64) (8051:120) Ratio=1.9
fun4:      mov A,R7                ; get input values Z,Y,X,W
           anl A,#0FH              ; ensure just 4 bits
           add A,#(fun4_t-fun4_1) ; offset for instructions
           movc A,@A+PC            ; get table entry
fun4_1:    mov C,ACC.0             ; lsb into carry
           mov Q,C                 ; and hence Q
           ret
fun4_t:    db 1,1,1,1,1,1,1,1      ; table represents easy way
           db 0,0,0,1,1,1,1,1      ; to implement any function
;Number of clock cycles:(MSC120x:52) (8051:108) Ratio=2.1
fun5:      mov A,R7                ; get input values Z,Y,X,W
           xrl a,#08H              ; complement Z
           clr C                   ; clear carry
           subb A,#3               ; test for boundary
           cpl C                   ; correct polarity
           mov Q,C                 ; and output to Q
           ret

;Number of clock cycles:(MSC120x:44) (8051:84) Ratio=1.9
fun6:      mov A,R7                ; get input values Z,Y,X,W
           xrl A,#08H              ; complement Z
           add A,#0FDH             ; identify boundary
           mov Q,C                 ; and output to Q
           ret

           end
```

System Clocks, Timers, and Functions

This chapter describes the system clocks, timers, and functions of the MSC120x.

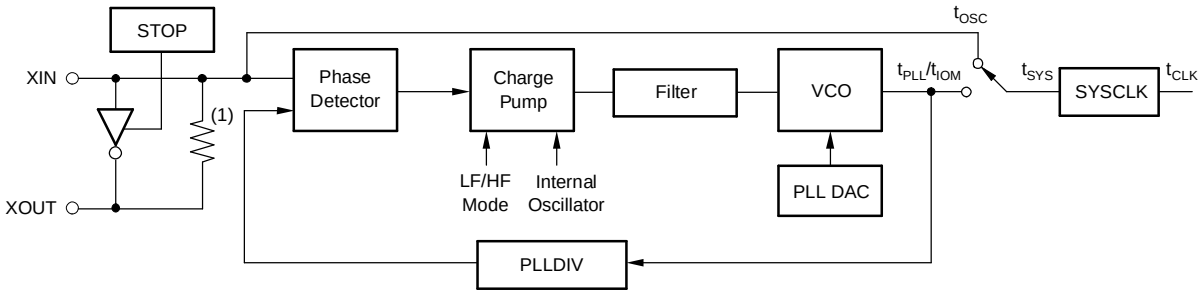
Topic **Page**

5.1 System Clocks

The MSC120x can operate in three different clock modes:

- Internal oscillator mode (IOM)
- External clock/oscillator mode (ECM)
- Phase Lock Loop mode (PLL)

A block diagram is shown in Figure 5-1



NOTE: (1) Disabled in PLL mode; therefore, an external resistor between XIN and XOUT is required.

Figure 5-1. Clock Block Diagram

During User Application mode (UAM), the clock mode of MSC120x is selected via the CLKSEL bits in Hardware Configuration Register 2 (HCR2). Please refer to Table 5-1 for a list of active clock modes. The default active clock mode is Internal Oscillator Low Frequency (IO LF) mode (HCR2, bit 2:0 = 111).

In Serial Flash Programming mode (SFPM), IO LF is the only clock mode. The HCR2 CLKSEL bits have no effect.

Table 5-1. Active Clock Modes

Selected Clock Mode		HCR2, CLKSEL2:0	Startup Condition[1]	Active Clock Mode (f_{SYS})
External Clock/Oscillator mode (ECM)		010	Active clock present at XIN	External Clock mode
			No clock present at XIN	IO LF mode
Internal Oscillator mode (IOM)[2]	IO LF mode	111	N/A	IO LF mode (default)
	IO HF mode	110	N/A	IO HF mode
Phase Lock Loop mode (PLL)[3]	PLL LF mode	101	Active 32.768kHz clock at XIN	PLL LF mode
			No clock present at XIN	Nominal 50% of IO LF mode[4]
	PLL HF mode	100	Active 32.768kHz clock at XIN	PLL HF mode
			No clock present at XIN	Nominal 50% of IO HF mode[4]

(1) Clock detection is only done at startup; refer to the MSC120x data sheet, Figure 2 - Serial Flash Programming Timing parameter, t_{RFD}.
(2) XIN must not be left floating; it must be tied high or low or parasitic oscillation may occur.
(3) PLL operation requires that both AV_{DD} and DV_{DD} are within specified ranges.
(4) This is not a proper working mode, when a PLL mode is selected, a 32.769KHz crystal is required

5.1.1 Internal Oscillator Mode

In IOM (internal oscillator mode), the CPU executes either in LF mode (if HCR2, CLKSEL = 111) or HF mode (if HCR2, CLKSEL = 110).

f_{SYS} = 14.75MHz in IO LF Mode (at T_A = +25°C and V_{DD} = 5V)

f_{SYS} = 29.5MHz in IO HF mode (at T_A = +25°C and V_{DD} = 5V)

5.1.2 External Clock/Oscillator Mode

In ECM (external clock/oscillator mode) (HCR2, CLKSEL = 011), the CPU uses an external crystal, external ceramic resonator, external clock, or external oscillator. If an external clock is detected at startup, then the CPU begins execution in ECM after startup. If an external clock is not detected at startup, then the device reverts to Internal Oscillator Low frequency Mode (IO LF).

5.1.3 PLL Mode

In Phase Lock Loop (PLL) mode (HCR2, CLKSEL = 101 or 100), the CPU can execute from an external 32.768kHz crystal. This mode enables the use of a PLL circuit that synthesizes the selected clock frequencies (PLL LF mode or PLL HF mode). If an external clock is detected at startup, then the CPU will begin execution in PLL mode after startup. If an external clock is not detected at startup, then the device will revert to the mode shown in Table 5-1.

PLL9:0 (PLLH and PLLL, SFR at F4h and F5h) is preloaded with default trimmed values. However, the PLL frequency can be fine-tuned by writing to PLL9:0. The equation for the PLL frequency is:

PLL Frequency = (PLL9:0 + 1) • f_{OSC}

where f_{OSC} = 32.768kHz

The default frequencies for the PLL are:

PLL LF mode 14.7456 MHz (at T_A = +25°C, V_{DD} = 5V)

PLL HF mode 29.4912 MHz (at T_A = +25°C, V_{DD} = 5V)

The status of the PLL can be determined by first writing the PLLLOCK bit (enable) and then reading the PLLLOCK status bit in the PLLH SFR.

5.2 Timing Chain and Clock Controls

Along with Timer/Counters 0 and 1 found in the 8051/8052 architecture, the MSC120x has numerous additional system timers and clock generators. Figure 5-2 shows the MSC120x timing chain and clock control. The main oscillator provides the system clock at frequency f_{CLK} either directly, or via a programmable system clock divider. (t_{CLK} = 1/f_{CLK}).

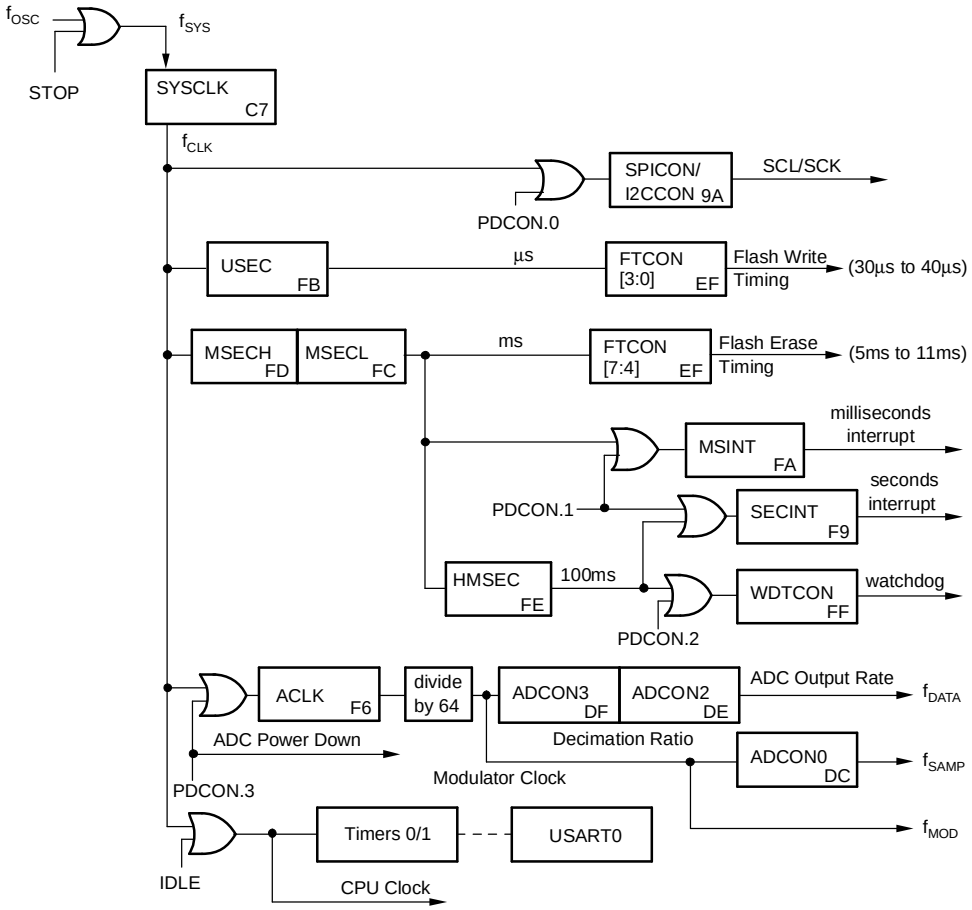


Figure 5-2. MSC120x Timing Chain and Clock Control

At power-on reset, the signal from the oscillator is not allowed to propagate until after (2^{17}– 1) periods. This period of time allows the power rails and crystal oscillator to stabilize. Thereafter, the CPU will begin to execute code starting at location 0000h. While operating, the CPU may set bit 1 of PCON at 87h to assert a stop condition that can only be exited by a hardware reset. During this stop condition, all dynamic activity ceases, but the port I/O pins retain their levels. To pause the CPU and core peripherals temporarily, bit 0 of PCON may be set. This will invoke an IDLE state that is terminated by an auxiliary interrupt associated with AIE at A6h, a wake-up via EWU at C6h, or a reset. See Chapter 11 for further detail on interrupts and their sources.

Subsystems are enabled/disabled by bits in PDCON at F1h in any combination, except that SPI and I^2C subsystems must not be simultaneously active. When a bit is high, the associated subsystems are inactive and power is reduced to a minimum static level.

With a system clock of *f* MHz, the program must write (*f*– 1) to the USEC register at FBh in order to provide a clock period as close to 1µs as possible. This clock provides the start and stop timing for the I^2C interface, and is used in conjunction with FTCON [3:0] at EFh to define the Flash memory write cycle timing. The least significant four bits of FTCON are referred to as FWR, and should be set so that (1 + FWR) × (USEC + 1) × 5 × t_{CLK} is between 30µs and 40µs. The designer should consider the relative trade-offs between crystal frequency and accuracy of baud rate generation versus accuracy of other real-time counters.

Just as USEC is programmed to provide a 1µs reference, MSECH at FDh and MSECL at FCh are used together to provide a signal with a period of 1ms to clock other counters. The period is:

(256 × MSECH + MSECL + 1) × t_{CLK}

which may not be an integer number of milliseconds. For example, with a 11.0592MHz crystal and MSECH:MSECL set to 11058, the period will be 1.000018ms. The default value for MSECH:MSECL is 3999 and assumes a 4MHz oscillator. Note that if the system divider, defined by SYSCLK at C7h, is present and active, an extra division factor may be present as well.

The output of MSECH:MSECL clocks three different counters with reload limits set by FTCON [7:4] at EFh, MSINT [6:0] at FAh, and HMSEC [7:0] at FEh. The values in these registers define the Flash memory erase timing between 5ms and 11ms, the number of counts for the milliseconds interrupt and the hundreds of millisecond interrupt, respectively. Each counter repeats in *N* + 1 clocks, where *N* is the value written to the bits in each SFR. If bit 7 of MSINT is set, the associated counter will be reloaded as the SFR is written; otherwise, the new value will be loaded next time the count expires.

The interrupt associated with SECINT [6:0] at F9h can be set between 1 and 128 counts of the hundred millisecond counter. If bit 7 of SECINT is set, the associated counter will be reloaded as the SFR is written; otherwise, the new value will be loaded next time the count expires.

The frequency of the ADC modulator is given by:

$$f_{MOD} = \frac{f_{CLK}}{(ACLK + 1) \times 64}$$

where ACLK is the SFR at F6h.

The conversion data rate is given by:

$$\text{ADC Update Rate} = \frac{f_{MOD}}{\text{Decimation Ratio}}$$

The decimation ratio is ADCON3[2:0] at DFh concatenated with ADCON2[7:0] at DEh plus 1, and the ADC output data rate is f_{MOD}/(decimation ratio).

5.2.1 Nonstandard Timing

If flash programming is not needed, and the timing does not require 1ms resolution, the value for the MSEC divider can be increased to allow longer timeouts and interrupts. For example, with an 11.0592MHz crystal frequency, the millisecond clock could extend to almost 6ms and the hundered millisecond clock could then extend to 1.5 seconds. This extension will allow a second interrupt of up to 194 seconds, or over three minutes. With the clock divider or slower crystals, much longer times can be achieved.

5.3 System Clock Divider

In order to reduce the average operating power of the microcontroller, a programmable system clock divider may lower the frequency of the internal clocks.

The system clock (f_{SYS}) divided by SYSCLK[2:0] (SFR C7h) equals f_{CLK}. The default value is $f_{CLK} = f_{SYS}$.

Table 5-2. SYSCLK—System Clock Divider Register

SYSCLK		SFR C7h	Reset Value = 00h
Bit #	**Name**	**Action or interpretation**	
7-6	0	Always 0	
5-4	DIVMOD	Clock Divide Mode	
		Write:	
		00: Normal mode (default, no divide)	
		01: Immediate mode: start divide immediately; return to Normal mode on an IDLE wakeup condition or direct write to SFR.	
		10: Delay mode: same as Immediate mode, except that the mode changes with the millisecond interrupt (MSINT). If MSINT is enabled, the divide will start on the next MSINT and return to normal mode on the following MSINT. If MSINT is not enabled, the divide will start on the next MSINT condition (even if masked) but will not leave the divide mode until the MSINT counter overflows, which follows a wakeup condition. Can exit by directly writing to SFR.	
		11: Manual mode: same as Immediate mode but cannot return to Normal mode on IDLE wakeup condition. Must write directly to SFR.	
		Read: Status	
		00: No divide	
		01: Divider is in Immediate mode	
		10: Divider is in Delay mode	
		11: Manual mode	
3	0	Always 0	
2-0	DIV	Divide Mode ($f_{CLK} = f_{OSC}$/Divisor)000:	
		000: divide-by-2 (default)	
		001: divide-by-4	
		010: divide-by-8	
		011: divide-by-16	
		100: divide-by-32	
		101: divide-by-1024	
		110: divide-by-2048	
		111: divide-by-4096	

5.3.1 Behavior in Delay Mode 10_2

Changes in the divisor are synchronized with the timeout of the milliseconds system timer, MSINT at FAh, which must be powered up (that is, bit 1 of PDCON at F1h must be 0). Once a new divisor is written to SYSCLK with this mode, it will take effect at the next MSINT timeout. During this time, bit 0 of PCON at 87h can be set to place the CPU in the IDLE state and reduce the power still further.

When the divisor is active and the milliseconds interrupt is enabled via EMSEC (bit 4 of AIE at A6h), the timeout causes immediate removal of the divisor. This event is likely to occur when a real-time (elapsed) clock is supported in software by maintaining a record of the accumulated number of millisecond interrupts. The program must compensate for the increase in time caused by the divisor.

In effect, if the milliseconds interrupt is enabled via EMSEC when the divider mode is changed to 10_2, the divisor will become active on the next MSINT interrupt, and return to divide-by-1 on the following MSINT interrupt. However, if the milliseconds interrupt is masked, the divisor will still become active on the next MSINT interrupt, but will not return to divide-by-1 until the milliseconds interrupt after a wake-up condition. If the wake-up condition is caused by an enabled seconds interrupt that is synchronous with a millisecond interrupt, the divider immediately returns to divide-by-1.

5.4 Watchdog Timer

WDTCON [4:0] at FFh defines the number of 100ms intervals (+1) before the watchdog timer expires, assuming that the watchdog restart sequence is not performed. The watchdog is enabled (or disabled) by writing a 1,0 sequence to bit 7 (or bit 6) of WDTCON. Writing 1,0 to bit 5 restarts the time out.

When the watchdog is enabled and expires, it generates either an interrupt or a reset (default), as determined by bit 3 of HCR0.

WDTI must be cleared within the interrupt service routine (ISR). Setting WDTI in software generates a watchdog timer interrupt, if enabled.

Table 5-3. Watchdog Control Bits

Watchdog Interrupt has priority 12 (Low) and jumps to address 63h				
Bit Name	**Abbreviation**	**Name of Related SFR**	**Abbreviation**	**Address (Hex)**
Global Interrupt Enable	EA	Interrupt Enable	IE.7	A8
Enable Watchdog Interrupt	EWDI	Extended Interrupt Enable	EIE.4	E8
Watchdog Timer Interrupt flag	WDTI	Enable Interrupt Control	EICON.3	D8
Watchdog Interrupt Priority	PWDI	Extended Interrupt Priority	EIP.4	F8

5.4.1 Watchdog Timer Example Program

When the program is run, it first requires a carriage return (CR) character to be received so that the baud rate can be determined. Thereafter, a CR code must be repeatedly received within three seconds; otherwise, the MSC120x is reset and the autobaud routine is restarted.

In Example 5-1, EWDR, bit 3 of HCR0, must be 1 (default) for a reset to occur. In another application, the programmer may clear EWDR so that when the timer expires, an interrupt is requested via WDTI, bit 3 of EICON at D8h.

Example 5-1. Watchdog Timer Program

```
// File WDT.c - Watch Dog Timer
// MSC1200 EVM
//Jumper setting:  factory default setting

#include <Reg1200.h>
#include <stdio.h>

#define xtal 1105920
sbit RedLed = P3^4;                // RED LED on EVM
sbit YellowLed = P3^5;             // Yellow LED on EVM

code at 0xfbfa void autobaud(void);

data unsigned char i='A';

void main(void)
{   PDCON&=~0x04;                  // power up WatchDog,
    MSEC=xtal/1000-1;              // 1ms tick
    HMSEC=100-1;                   // 100ms tick
    RedLed=0;                      // Turn Red LED on
    CKCON = 0x10;                  // UART autobaud
    TCON = 0x00;                   // UART autobaud
    autobaud();                    // Requires CR
    printf("\nMSC1200 Watchdog Test");
    printf("\nRepeatedly press CR/Enter within 3 seconds\n");
    RI_0 = 0;                      // clear received flag in USART
    WDTCON=0x80;                   // start watchdog and define
    WDTCON=30;                     // 30 * 100ms timeout
    RedLed=1;                      // Turn Red LED off
    while(1){
        while(!RI_0);              // wait for key press
        YellowLed=!YellowLed;      // Toggle Yellow LED
        putchar(i);
        i=(i+1) & 0x5F;            // 32 character sequence
        if((SBUF0 & 0x7F)==0x0D) { // Test for CR
            WDTCON|=0x20;          // restart Watchdog timer
            WDTCON&=~0x20;         // with 1-0 sequence in bit #5
        }
        RI_0 = 0;                  // clear received flag in USART
    }
}
```

5.5 Analog Low-Voltage Detection

The MSC120x family has an analog low-voltage detection function. An interrupt is generated when the analog power supply drops below a user-defined level (set via LVDCON, E7h), and if the Analog Low-voltage Detect interrupt is enabled; that is, EAI (EICON.5, SFR D8h) and EALV (AIE.1, SFR A6h) are set to 1.

To clear the interrupt, the AI flag (EICON.4, SFR D8h) must be cleared before exiting from the ISR.

For the threshold of analog low-voltage detection, please refer to Table 5-4.

Table 5-4. LVDCON—Low-Voltage Detect Control

LVDCON				SFR E7h Reset Value = 00h
ALVD3 Bit 3	ALVD2 Bit 2	ALVD1 Bit 1	ALVD0 Bit 0	Analog Threshold of AV_{DD}
1	x	x	x	Reserved
0	0	0	0	4.6V
0	0	0	1	4.2V
0	0	1	0	3.8V
0	0	1	1	3.6V
0	1	0	0	3.3V
0	1	0	1	3.1V
0	1	1	0	2.9V
0	1	1	1	2.7V

5.6 Digital Brownout Reset

The Digital Brownout Reset (DBOR) is enabled via bit 2 of Hardware Configuration Register 1 (HCR1.2). The system goes into a hold state (internal reset signal goes low) when the digital power supply goes below the user-defined level (set via bit 7-4 of HCR1). When the digital power supply rises above the threshold level for 2^{11} clock cycles, an internal reset is then generated and the user program starts from the beginning.

Please note that when program execution begins, the device current consumption may increase; this current consumption can result in a power-supply voltage drop, which may then initiate another brownout condition. Additionally, the DBOR comparison is done against an analog reference; therefore, AV_{DD} must be within its valid operating range for DBOR to function.

Table 5-5. HCR1—Hardware Control Register 1

HCR1				SFR E7h Reset Value = 00h
DBSEL3 Bit 7	DBSEL2 Bit 6	DBSEL1 Bit 5	DBSEL0 Bit 4	Digital Threshold of DV_{DD}
1	x	x	x	Reserved (except 1000)
0	0	0	0	4.6V
0	0	0	1	4.2V
0	0	1	0	3.8V
0	0	1	1	3.6V
0	1	0	0	3.3V
0	1	0	1	3.1V
0	1	1	0	2.9V
0	1	1	1	2.7V
1	0	0	0	2.6V

5.7 Hardware Configuration

There are three hardware configuration registers in the MSC120x:

- HCR0 at CADDR 3Fh
- HCR1 at CADDR 3Eh
- HCR2 at CADDR 3Dh

These three registers can be read during User Application mode (UAM), but can be written to only during Serial Flash Programming mode (SFPM).

Table 5-6. HCR0—Hardware Configuration Register 0 (1)

HCR0		Non-SFR address 3Fh accessed indirectly via SFR CADDR at 93h; Erased Value = FFh
Bit #	**Name**	**Action or Interpretation**
7	EPMA	Enable Programming Memory Access (security bit). 0: After a reset in programming modes, Flash memory can only be accessed in UAM until a mass erase is done. 1: Fully Accessible (default)
6	PML	Program Memory Lock (PML has priority over RSL). 0: Enable read and write for program memory in UAM. 1: Enable read-only mode for program memory in UAM (default).
5	RSL	Reset Sector Lock (Reset Sector is first 4K bytes of flash memory). Used to provide another method of Flash Memory programming, which allows Program Memory updates without changing the jumpers for in-circuit code updates or program development. The code in this boot sector would then provide the monitor and programming routines with the ability to jump into the main Flash code when programming is finished. 0: Enable reset sector writing 1: Enable read-only mode for reset sector (4kB) (default). Same effect as PML for the MSC120xY2.
4	EBR	Enable Boot Rom. Boot ROM is 1kB of code located in ROM at address F800h during UAM, not to be confused with the 4kB boot sector located in Flash memory at address F800h during UAM. 0: Disable internal boot ROM 1: Enable internal boot ROM (default)
3	EWDR	Enable Watchdog Reset 0: Disable watchdog reset 1: Enable watchdog reset (default)
1	DFSEL1	Data Flash Memory Size. On-chip Flash memory can be partitioned between data memory and program memory. The total memory available depends on the Y version of the device. See Section 2.2 for a complete description of memory partitioning.
0	DFSEL0	00: 4kB data Flash memory (MSC120xY3 only) 01: 2kB data Flash memory 10: 1kB data Flash memory 11: no data Flash memory (default)

(1) HCR0 is programmable only in SFPM, but can be read in UAM using the **faddr_data_read** boot ROM routine.

Table 5-7. HCR1—Hardware Configuration Register 1[(1)]

HCR1		Non-SFR address 3Eh accessed indirectly via SFR CADDR at 93h; Erased Value = FFh
Bit #	**Name**	**Action or Interpretation**
7	DBSEL3	Digital Supply Brownout Level Select. The values listed are nominal. The actual value will vary depending on device clock frequency and supply voltage. For high clock frequencies, the variation could be on the order of 10% below the nominal value. The digital brownout level is loaded after POR; therefore, a proper POR must occur for digital brownout levels to be properly loaded. 0000: 4.6V 0001: 4.2V 0010: 3.8V 0011: 3.6V 0100: 3.3V 0101: 3.1V 0110: 2.9V 0111: 2.7V 1000: 2.6V 1001: Reserved 1010: Reserved 1011: Reserved 1100: Reserved 1101: Reserved 1110: Reserved 1111: Reserved
6	DBSEL2	
5	DBSEL1	
4	DBSEL0	
2	DDB	Disable Digital Brownout Detection 0: Enable digital brownout detection 1: Disable digital brownout detection (default)

(1) HCR1 is programmable only in SFPM, but can be read in UAM using the **faddr_data_read** boot ROM routine.

Table 5-8. HCR2—Hardware Configuration Register 2[(1)]

HCR2		Non-SFR address 3Dh accessed indirectly via SFR CADDR at 93h; Erased Value = FFh
Bit #	**Name**	**Action or Interpretation**
2	CLKSEL2	Clock Select. NOTE: Clock status can be verified by reading PLLH in UAM. 000: Reserved 001: Reserved 010: Reserved 011: External clock mode 100: PLL HF mode 101: PLL LF mode 110: IO HF mode 111: IO LF mode
1	DBSEL1	
0	DBSEL0	

(1) HCR2 is programmable only in SFPM, but can be read in UAM using the **faddr_data_read** boot ROM routine.

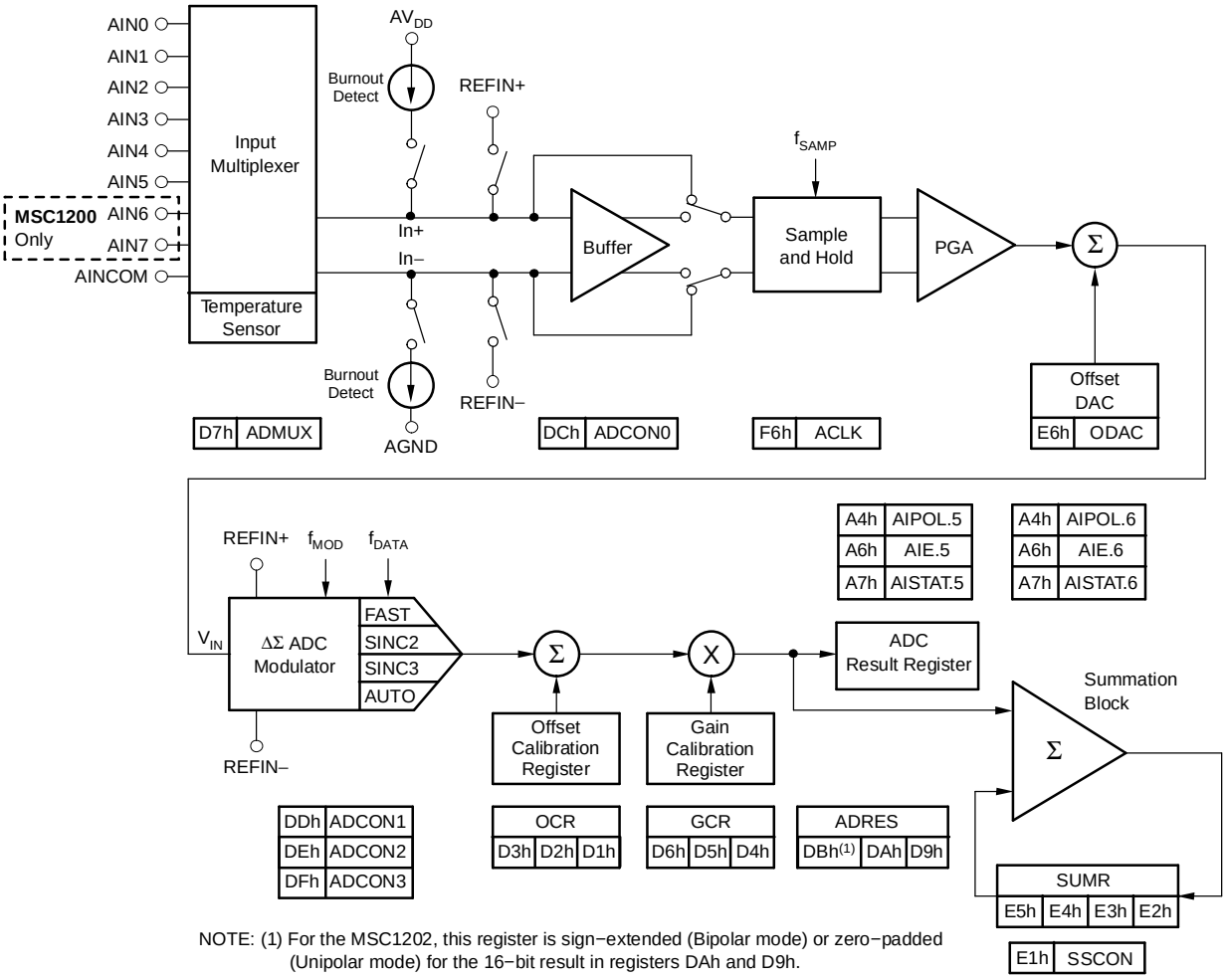
Analog-To-Digital and Digital-to-Analog Converters

This chapter describes the analog-to-digital converters (ADCs) and the digital-to-analog converter (DAC) of the MSC120x.

Topic **Page**

6.1 ADC Functional Blocks

A key feature of the MSC120x that differentiates it from other mixed-signal microcontrollers is a high-precision analog-to-digital subsystem, with performance that is usually found only in embedded systems with a separate ADC and microprocessor. The major elements of the ADC subsystem are shown in Figure 6-1.



NOTE: (1) For the MSC1202, this register is sign−extended (Bipolar mode) or zero−padded (Unipolar mode) for the 16−bit result in registers DAh and D9h.

Figure 6-1. ADC Subsystem Elements

6.2 ADC Signal Flow and General Description

Analog signals from pins AIN0 to AIN5 (AIN0 to AIN7 for MSC1200), AINCOM, and internal temperature-sensitive diodes are selected independently by two analog multiplexers to provide a differential signal to the programmable gain amplifier (PGA), which may optionally be preceded by a high-impedance buffer. An analog offset of up to ±50% of the full range may be injected into the PGA by the Offset DAC.

The delta-sigma ($\Delta\Sigma$) ADC can be configured for sampling rate and decimation ratio as well as filter type before its output is passed to digital offset and gain calibration stages to give a 24-bit unipolar or bipolar result.

ADC conversions can be automatically added to a 32-bit summation register (SUMR3 to SUMR0), which is considerably more efficient than using machine-code instructions. A defined number of conversions may also trigger an automatic right shift to produce an averaged value. The CPU can control the 32-bit hardware accumulator directly, as long as the ADC subsystem is powered up. All MSC120x family parts also support 32-bit subtraction.

6.3 Analog Input Stage

Special function register ADMUX at D7h provides two groups of four bits each that specify the analog source channels for the noninverting (positive) and inverting (negative) inputs to the buffer and/or the PGA.

The upper four bits control the noninverting input while the lower four bits control the inverting input. Codes 0000_2 to 0111_2 represent channels AIN0 to AIN7, respectively. Code 1000_2 selects AINCOM, and if both codes are 1111_2, two temperature-sensitive diodes are selected.

When Burnout Detection is enabled, current sources cause the inputs to be pulled to either AV_{DD} or AGND if the selected channel is open circuit, as may happen when a resistive sensor is broken.

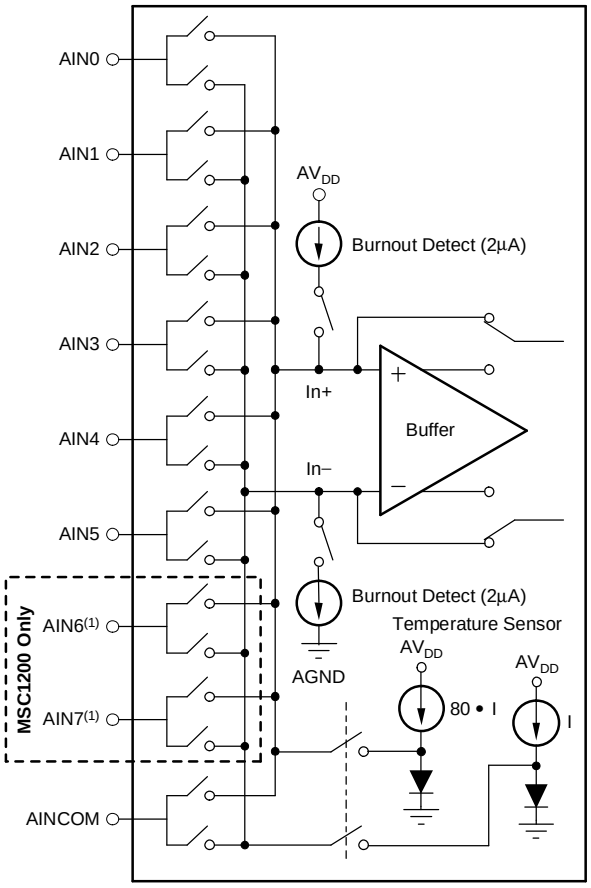
The internal diodes are used to provide a temperature-sensitive differential voltage of approximately:

$$V = \frac{nk\ \ln(80)}{q}(T_C + 273.16) = \alpha T_C + \beta$$

For the MSC1200, α = 375µV/°C, and β = 0.115 – 25°C × (375µV/°C).

For the MSC1201 and MSC1202, α = 345µV/°C, and β = 0.115 – 25°C × (345µV/°C).

For further information about accuracy and calibration, see Texas Instruments application report SBAA100, *Using the MSC120x as a High-Precision Intelligent Temperature Sensor*, available for download at www.ti.com.



NOTE: (1) For MSC1201/MSC1202, AIN6 and AIN7 are tied to REFIN–.

Figure 6-2. Analog Input Structure without Buffer

Table 6-1. ADMUX—ADC Multiplexer

ADMUX				SFR D7h Reset Value = 01h
INP3 Bit 7	INP2 Bit 6	INP1 Bit 5	INP0 Bit 4	Positive input selection
INN3 Bit 3	INN2 Bit 2	INN1 Bit 1	INN0 Bit 0	Negative input selection
0	0	0	0	AIN0 (default positive input)
0	0	0	1	AIN1 (default negative input)
0	0	1	0	AIN2
0	0	1	1	AIN3
0	1	0	0	AIN4
0	1	0	1	AIN5
0	1	1	0	AIN6 (MSC1200 only)
0	1	1	1	AIN7 (MSC1200 only)
1	0	0	0	AINCOM
1	1	1	1	Temperature sensor. Requires ADMUX = FFh.

6.4 Input Impedance, PGA, and Voltage References

When the buffer is enabled, the input current is typically 0.5nA (impedance is > 1GΩ) and the common-mode range is from AGND + 50mV to AV_{DD} – 1.5V. The buffer should be enabled whenever burnout detection is used.

However, when the buffer is not enabled, each analog input is presented with a dynamic load such that the mean differential impedance is (7MΩ/*G* at f_{ACLK} = 1MHz); where *G* is defined in Table 6-2. The input impedance is lowered and varies with gain. The input range is from AGND – 0.1 V to AV_{DD} + 0.1 V.

Table 6-2. Impedance Divisor (G) for a Given PGA

PGA	1	2	4	8	16	32	64	128
G	1	2	4	8	16	32	64	64

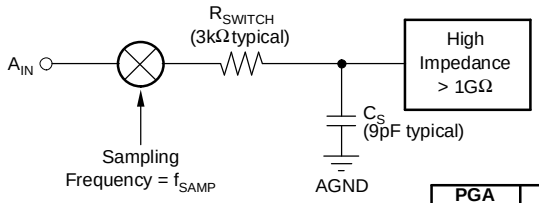
When the buffer is not selected, the input impedance of the analog input changes with ACLK clock frequency (ACLK, SFR F6h) and gain (PGA). The relationship is:

$$A_{IN}\ \text{Impedance}\ (\Omega) = \left(\frac{1\text{MHz}}{\text{ACLK Frequency}}\right) \cdot \left(\frac{7\text{M}\Omega}{\text{G}}\right)$$

where:

$$\text{ACLK frequency}\ (f_{ACLK}) = \frac{f_{CLK}}{\text{ACLK} + 1}$$

Figure 6-3 shows the basic input structure of the MSC120x.



PGA	C_S
1	9pF
2	18pF
4 to 128	36pF

PGA	BIPOLAR MODE FULL-SCALE RANGE	UNIPOLAR MODE FULL-SCALE RANGE	f_{SAMP}
1	$\pm V_{REF}$	$+V_{REF}$	f_{MOD}
2	$\pm V_{REF}/2$	$+V_{REF}/2$	f_{MOD}
4	$\pm V_{REF}/4$	$+V_{REF}/4$	f_{MOD}
8	$\pm V_{REF}/8$	$+V_{REF}/8$	$f_{MOD} \cdot 2$
16	$\pm V_{REF}/16$	$+V_{REF}/16$	$f_{MOD} \cdot 4$
32	$\pm V_{REF}/32$	$+V_{REF}/32$	$f_{MOD} \cdot 8$
64	$\pm V_{REF}/64$	$+V_{REF}/64$	$f_{MOD} \cdot 16$
128	$\pm V_{REF}/128$	$+V_{REF}/128$	$f_{MOD} \cdot 16$
NOTE: f_{MOD} = ACLK frequency/64			

Figure 6-3. Input Multiplexer Configuration

Table 6-3. ADCON0—ADC Control Register 0

ADCON0		SFR DCh	Reset Value = 30h
Bit #	**Name**	**Action or Interpretation**	
6	BOD	Burnout Detect When enabled, a 2µA current source is connected from AV_{DD} to the positive input, while a 2µA current sink is connected from the negative input to ground. Write: 0: Burnout Current Sources Off (default) 1: Burnout Current Sources On	
5	EVREF	Enable Internal Voltage Reference Write: 0: Internal Voltage Reference Off 1: Internal Voltage Reference On (default). If the internal voltage reference is not used, it should be turned off to save power and reduce noise	
4	VREFH	Voltage Reference High Select Write: 0: REFOUT is 1.25V (AV_{DD} = 2.7V to 5.25V) 1: REFOUT is 2.5V (default) (AV_{DD} = 3.3V to 5.25V)	
3	EBUF	Enable Buffer Write: Buffer disabled (default) Buffer enabled; results in increased power and impedance but reduced range	
2-0	PGA	Programmable Gain Amplifier Write: 000 to 111: Gives a gain G = 2^{PGA} or 1 (default) to 128	

Bits of ADCON0 determine various parameters according to Table 6-4.

Table 6-4. ADCON0 Bit Parameters

PGA Bits [2:0]	Gain	Full-Scale Range	Sampling Frequency	Effective Number of Bits at 10Hz Rate	RMS Resolution for V_{REF} = 2.5V (nV)
000	1	$\pm V_{REF}$	f_{MOD}	21.7	1468
001	2	$\pm V_{REF}/2$	f_{MOD}	21.5	843
010	4	$\pm V_{REF}/4$	f_{MOD}	21.4	452
011	8	$\pm V_{REF}/8$	2 f_{MOD}	21.2	259
100	16	$\pm V_{REF}/16$	4 f_{MOD}	20.8	171
101	32	$\pm V_{REF}/32$	8 f_{MOD}	20.4	113
110	64	$\pm V_{REF}/64$	16 f_{MOD}	20	74.5
111	128	$\pm V_{REF}/128$	16 f_{MOD}	19	74.5

By default, the internal voltage reference is turned on at 2.5V, when the ADC subsystem is powered up. Therefore, if an external reference is provided, the internal reference should be disabled via EVREF before bit 3 of PDCON at F1h is cleared.

If the internal voltage reference is to be used, the default level of 2.5V is allowed only if AV_{DD} is between 3.3V and 5.25V. The internal 1.25V V_{REF} can be used over the entire analog supply range (AV_{DD} = 2.7V to 5.25V).

When the internal voltage reference is disabled, an external differential reference is represented by the voltage between REF IN+ and REF IN–. This permits ratiometric measurements, but the absolute voltage on either input must be between AGND and AV_{DD}.

In both cases, the REF IN+ pin should have a 0.1µF capacitor to AGND.

6.5 Offset DAC

The PGA input range may be offset by up to ±50% via the offset DAC. This 8-bit DAC is controlled by ODAC at E6h with a coding scheme such that the most significant bit (bit 7) represents the sign of the offset, while bits 6-0 represent the magnitude. When the magnitude is zero, the ODAC is disabled and the voltage into the PGA is not offset.

$$\text{Offset} = \frac{V_{REF}}{2 \times \text{PGA}}\left(\frac{\text{ODAC}[6:0]}{127}\right)(-1)^{[\text{ODAC7}]}$$

where PGA is the gain of the programmable gain amplifier.

Here, V_{REF} is the voltage on the REF IN+ pin with respect to REF IN– and should not be confused with the internal voltage reference that is with respect to AGND.

The gain error of the 8-bit ODAC is typically about ±1.5% of its range, which means its absolute accuracy can be significant in some applications. However, it is monotonic with an integral nonlinearity of less than 0.25 bits, and has a temperature coefficient of typically 1ppm/°C. It may be used in a predictive manner with due regard to its range, resolution, stability, and accuracy, or it may be calibrated using the ADC.

6.6 ADC Data Rate, Filters, and Calibration

The data rate for ADC conversions is determined by the frequency of the modulator clock, f_{MOD}, and the decimation ratio, which is a right-justified, 11-bit field in ADCON3 at DFh (high) concatenated with ADCON2 at DEh (low). Its default value is 1563.

$$\text{ADC Output Data Rate} = f_{DATA} = \frac{f_{MOD}}{\text{Decimation Ratio}} = \frac{1}{t_{DATA}}$$

$$\text{where } f_{MOD} = \frac{f_{CLK}}{(\text{ACLK} + 1) \times 64}$$

When the decimation ratio, PGA, AV_{DD}, buffer, voltage reference, or temperature are changed, the ADC must be recalibrated.

The mode of operation of the ADC is controlled by ADCON1 at DDh, which determines whether the inputs are interpreted as unipolar or bipolar, the type of digital filter, and the type of calibration.

Table 6-5. ADCON1—ADC Control Register 1

ADCON1		SFR DDh Reset Value = 30h
Bit #	**Name**	**Action or Interpretation**
7	OF_UF	Summation Invalid If this bit is set, the data in the summation register is invalid; either an overflow or underflow occurred. The bit is cleared by writing a '0' to it.
6	POL	Polarity Write: 0: Bipolar such that –FSR = 0x800000, zero = 0x000000 and +FSR = 0x7FFFFF 1: Unipolar such that –FSR = 0x000000, zero = 0x000000 and +FSR = 0xFFFFFF
5	SM1	Settling Mode Write: 00: Auto 01: Fast 10: $Sinc^2$ 11: $Sinc^3$
4	SM0	
3	—	Not used
2	CAL2	Calibration Control Write: 000: No Calibration (default) 001: Self Calibration for Offset and Gain (14 t_{DATA} periods to complete) 010: Self Calibration for Offset only (7 t_{DATA} periods to complete) 011: Self Calibration for Gain only (7 t_{DATA} periods to complete) 100: System Calibration for Offset only (7 t_{DATA} periods to complete, requires external signal) 101: System Calibration for Gain only (7 t_{DATA} periods to complete, requires external signal) 110: Reserved 111: Reserved Read: 000_2
1	CAL1	
0	CAL0	

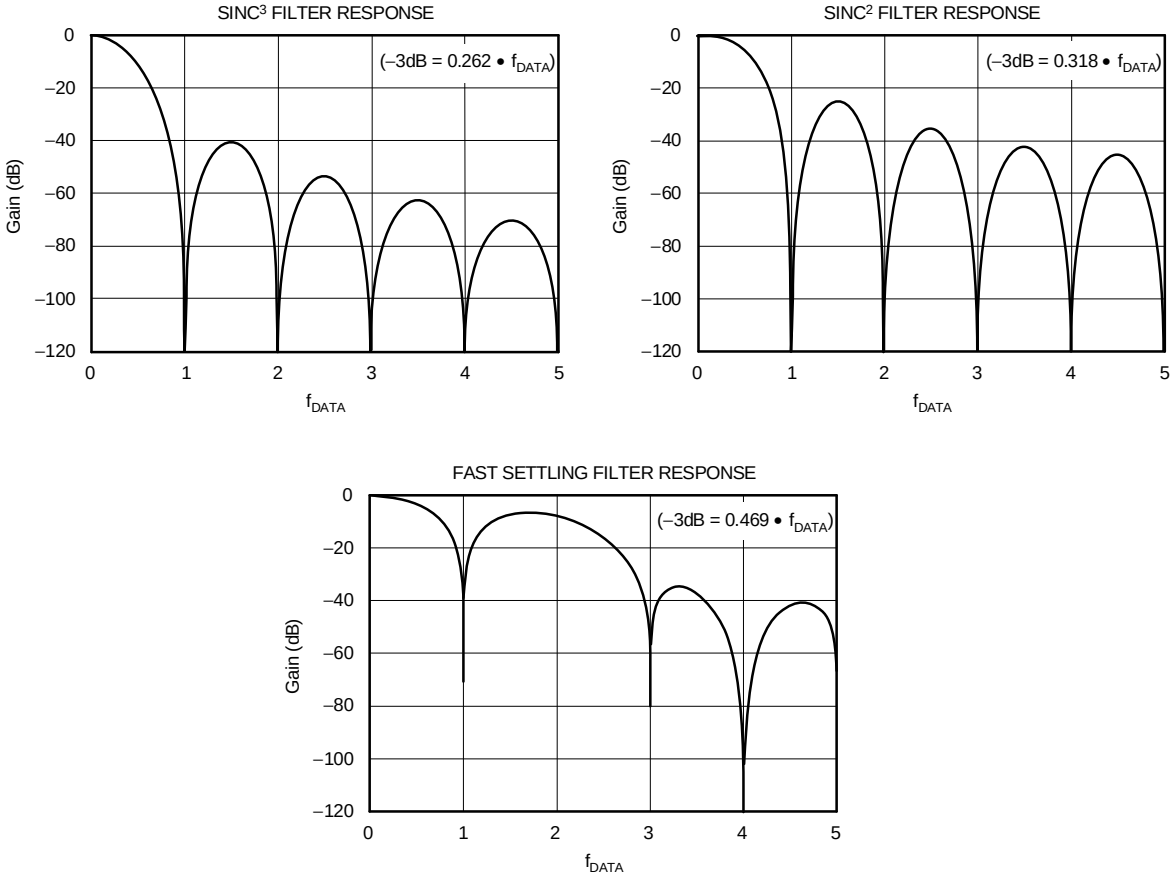
When the voltage presented to the ADC changes, the time it takes to receive valid data depends upon the type of filter that is selected, as well as the conversion time, t_{DATA}. Higher-order filters provide better noise immunity but take longer to settle, and the user must make considered judgments as to system performance based on resolution, settling time, and notch frequency.

In Auto mode, the type of filter that is used changes whenever the input multiplexer, ADMUX, or PGA are altered. The ADC first makes two conversions using the Fast filter, then one with $Sinc^2$, and then one with $Sinc^3$.

In the graphs shown in Figure 6-4, f_{DATA} = Data Output Rate = 1/ t_{DATA}.

The ADC performs conversions at a regular rate of f_{DATA}, as shown in the following equation:

$$f_{DATA} = \left(\frac{f_{CLK}}{64 \times (ACLK + 1) \times \text{ Decimation Ratio}} \right)$$



NOTE: f_{DATA} = Data Output Rate = $1/t_{DATA}$

Figure 6-4. Filter Frequency Responses

In applications where more than one analog input is measured, the program should write different values to ADMUX in a way that is synchronized with conversions to get the best throughput rate. Ideally, ADMUX should be updated as soon as the ADC interrupt flag is set, but there will always be a software delay. Assuming the delay is less than 20 × t_{MOD} and the decimation ratio is large (over 1000), any error introduced is less than intrinsic noise.

The 24-bit result is held in the logically concatenated registers ADRESH (high), ADRESM, and ADRESL (low), at SFR addresses DBh, DAh, and D9h, respectively. These registers are loaded when a conversion is completed, as long as ADRESL has been read since the last value was written.

In the MSC120x family, there are three registers that handle the interrupt functions:

- AIE (SFR A6h)
- AISTAT (SFR A7H)
- AIPOL (SFR A4h)

AIE enables or disables the corresponding auxiliary interrupt. AISTAT is the auxiliary interrupt status flag; it will be set valid only when the corresponding interrupt has been enabled in AIE and the interrupt occured. AIPOL is the auxiliary interrupt poll. It will always be set when an interrupt occurs. Table 6-6 shows the corresponding bits for ADC interrupt.

Table 6-6. ADC Interrupt Controls

Bit 5 of AIE (A6h) Enable/Disable ADC Interrupt	Bit 5 of AISTAT (A7h) ADC Interrupt Status Flag	Bit 5 of AIPOL (A4h) ADC Interrupt Poll
Read or Write: 0: Interrupt disabled (masked) 1: Interrupt enabled	Read: 0: Interrupt inactive or disabled 1: Interrupt active Write: No effect	Read: 0: Interrupt inactive 1: Interrupt active Write: No effect

When the MSC120x is reset, default values are loaded into the digital offset and digital gain calibration registers associated with the ADC. Specifically, for offset OCH:OCM:OCL = 00000000h and for gain GCH:GCM:GCL = 5FEC5Ah. (See Application Note SBAA099, Calibration Routines and Register Value Generation for the ADS121x Series, for additional information.) Although the ADC will then produce an output that varies linearly with the differential input voltage, it will not have the correct scale. A program is able to write any desired value to these calibration SFRs, but it is best to set the CAL bits in ADCON0 to force an internal calibration for offset and gain (CAL = 001). A differential input of V_{REF} = (REF IN+) – (REF IN–) will then map to a full-scale digital output. Alternatively, the overall system can be placed into a defined zero state and then calibrated for offset (CAL = 100) followed by a full-scale condition and calibrated for gain (CAL = 101). Each type of calibration takes seven t_{DATA} periods, as summarized in Table 6-3. CAL = 001 takes 14 t_{DATA} periods. For best results, calibration should be performed with the $Sinc^3$ or Auto filter selected.

6.7 32-Bit Summation Register

To use the 32-bit summation register, either under the control of the CPU and/or the ADC, bit 3 of PDCON at F1h must be '0'. Operations are controlled by SSCON at E1h, with data accessed via SUMR3:SUMR0.

Table 6-7. Summation Register

Register Name	Address (Hex)	Read Summation Register	Write Temporary Regiter
SUMR3	E5	Bits 31 to 24 (most significant)	Bits 31 to 24 (most significant)
SUMR2	E4	Bits 23 to 16	Bits 23 to 16
SUMR1	E3	Bits 15 to 8	Bits 15 to 8
SUMR0	E2	Bits 7 to 0 (least significant)	Bits 7 to 0 (least significant)

Table 6-8. SSCON—Summation/Shift Control

SSCON								SFR E1h Reset Value = 00h
SSCON1	SSCON0	SCNT2	SCNT1	SCNT0	SHF2	SHF1	SHF0	Description
0	0	0	0	0	0	0	0	Clear summation register
0	0	0	1	0	0	0	0	CPU summation on write to SUMR0 (sum count/shift ignored)
0	0	1	0	0	0	0	0	CPU subtraction on write to SUMR0 (sum count/shift ignored)
1	0	x	x	x	SHF			CPU shift according to SHF value
0	1	SCNT			x	x	x	ADC summation according to SCNT value
1	1	SCNT			SHF			ADC summation completes, then CPU shift according to SHF value

Immediately after a CPU instruction writes data to SUMR0, it may trigger an addition, subtraction, or shift operation, depending on the value of SSCON. Addition and subtraction take a single cycle, t_{CLK}. Shifting is performed either 1 or 2 bits per cycle, and takes up to four t_{CLK} periods to complete.

Table 6-9. Summation Interrupt Controls

Bit 6 of AIE (A6h) Enable Summation Interrupt	Bit 6 of AISTAT (A7h) Summation Interrupt Status Flag	Bit 6 of AIPOL (A4h) Summation Interrupt Poll
Read or Write: 0: Masked 1: Enabled	Read: 0: Inactive or masked 1: Active While active, no new data will be written to SUMR. Cleared by reading SUMR0 at E2h.	Read: Summation interrupt flag before masking.

6.8 Accessing the ADC Multi-Byte Conversion in C

ADRESH:ADRESM:ADRESL represent a 24-bit ADC conversion result register, while SUMR3:SUMR2:SUMR1:SUMR0 represent a 32-bit summation register. It is often useful to map both of these to long integers in C, but care should be taken. For example, assuming that the variable *sum* has been declared to be of type "signed long int," it is tempting to write:

```
sum = SUMR3 << 24 + SUMR2 << 16 + SUMR1 << 8 + SUMR0;
```

However, this produces a pattern-dependent incorrect value because of the (ANSI-defined) 16-bit integer promotion rules within most compilers for the 8051 family.

Changing to:

```
sum = ((unsigned long)SUMR3 << 24) + ((unsigned long)SUMR2 << 16)
    +  ((unsigned long)SUMR1 << 8)  +  (unsigned long)SUMR0;
```

will produce the expected value, but may take between approximately 800 and 1200 machine cycles, as compilers call run-time libraries to achieve multi-bit shifts. Since the order of additions is not defined in C, it is possible that SUMR0 is accessed first and the ADC interrupt flag is cleared. If other interrupts are present and their service routines take more time to complete than the next conversion, SUMR3, 2, 1 may be overwritten before being used to complete the evaluation of the expression.

Another approach is to define a union to overlay byte-wide variables with a 4-byte long integer.

```
typedef union {
   unsigned long v;
   char va[4];
   struct {char v3,v2,v1,v0;} vs;
   } type_sumv;
type_sumv data s;  //variable s is placed in on-chip data space
```

Then use:

```
s.vs.v3=SUMR3;
s.vs.v2=SUMR2;
s.vs.v1=SUMR1;
s.vs.v0=SUMR0;    // SUMR0 is accessed last
reading = f(s.v); // some function of the 4-byte variable v.s
```

Alternatively, array elements may be used, but the order of subscripts is reversed.

```
s.va[0]=SUMR3;
s.va[1]=SUMR2;
s.va[2]=SUMR1;
s.va[3]=SUMR0;
```

Although the code needed to access the union may appear clumsy, it maps to simple inline assembly-level MOV instructions that take 3 × 4 = 12 machine cycles to execute. In other words, it is approximately 100 times faster than using multiple shifts.

In the next example, the ADC results register is read using an assembly-level program, which makes expressions in C more intuitive. This technique may also be used to read the summation register.

6.9 ADC Example Program

Example 6-1 shows how the ADC may be used in a polled environment with a foreground activity that produces a pseudo-random binary data stream. The number of characters output per line equals the temperature of the MSC120x in degrees Celsius (°C). The main program is written in C and calls the boot ROM to determine the baud rate, and an assembly language function to read the ADC conversion. It is intended for use directly with Texas Instruments MSC1200-DAQ-EVM or full EVMs with an appropriate value for ACLK.

Example 6-1. ADC Program

```
// Polledadc.c - Pseudo Random Binary Sequence generator with Polled ADC
// MSC1200 EVM
// jumper setting as factory default

#include <Reg1200.h>
#include <stdio.h>

sbit RedLed = P3^4;    // RED LED on EVM
sbit YellowLed = P3^5; // Yellow LED on EVM
code at 0xFBFA void autobaud(void);

extern signed long bipolar(void); // reads ADC value

void main(void)
{
    data char mask=0x8E, r=1,n,j,x, temp=50, count=255;
    data signed long reading;
    data int iy;
    data float y;

    PDCON &=~0x08;      // turns on adc and leaves other subsystems unchanged
    ACLK = 17;          // = 11.0592MHz/(17+1) = 0.6144MHz
    DECIMATION = 1920;  // => 200ms per conversion
    ADCON0 = 0x20;      // BOD off, Vref on, 1.25V, Buff off, PGA 1
    CKCON = 0x10;       // for UART autobaud in MSC1200
    TCON = 0x00;        // for UART autobaud in MSC1200
    RedLed = 0;         // turn on red LED
    autobaud();         // requires a CR
    printf("MSC120x Random bit generator with polled ADC\n");
    printf("Readings begin in (14+3)*200ms = 3.4 seconds \n");
    ADMUX = 0xff;       // Select Temperature diodes
    ADCON1 = 0x01;      // bipolar, auto mode, self calibration - offset and gain
    for (j=0;j<3;j++) {
        while (!(AIPOL & 0x20)) {}
        reading=bipolar();                    // discard 3 conversions after calibration
    }
    RI_0 = 0;                                 // Clear received flag in USART
    while (!(AIPOL & 0x20)) {}                // wait for conversion
    while(1){
        while(!RI_0) {
            if (AIPOL & 0x20) {               // wait for conversion completed
                reading = bipolar();          // get reading and clear flag
                y=(reading-704509)/2595.1;    // convert to Degrees C (empirical 2)
                iy=y+0.5;                     // nearest integer
                if ((iy>0) && (iy<50)) temp=iy; // clamp range
            }
            if (count>=temp) {                // if line length >= temperature
                printf("\n%3d",temp);         // output new line and temperature
                YellowLed=RedLed;
                count=0;
            }
            n=r & mask;                       // PRBS generator with 4-bit feedback
            j=0;                              // j will become the sum of 1's in n
            while (n)
```

Example 6-1. ADC Program (continued)

```
                {n&=(n-1); j++;}
                r=(r+r)+(j&1);                  // shift r left with lsb of sum
                if (r&1) putchar('*');          // Note: putchar takes 28 machine cycles
                else printf(".");               // but printf takes 354 machine cycles
                count++;                        // increment character count
            }
            RI_0 = 0;
            while(!RI_0);                       // wait for character
            RI_0 = 0;
    }                                           // continue
}
```

From TI file *Utilities.A51*

```
;File name: utilities.a51
;
; Copyright 2003 Texas Instruments Inc as an unpublished work.
; All Rights Reserved.
;
; Revision History
; Version 1.1
;
; Assembler Version (Keil V2.38), (Raisonance V6.10.13)
;
; Module Description:
; ADC routines to read 24-bit ADC and return the value as a long integer.
;*********************************************************************
;$include (legal.a51) ; Texas Instruments, Inc. copyright and liability
$include (reg1200.inc)
;*********************************************************************
PUBLIC unipolar, bipolar, read_sum_regs
adc_sub SEGMENT CODE
RSEG adc_sub
;;;;;;;;;;;;;;;;;;;;
; unsigned long unipolar(void)
; return the 3 byte adres to R4567 (MSB~LSB)
; unsigned long int with R4=0
unipolar:
    mov r4,#0
    mov r5,adresh
    mov r6,adresm
    mov r7,adresl
    ret
;;;;;;;;;;;;;;;;;;;;
; signed long bipolar(void)
; return the 3 byte adres to R4567 (MSB~LSB)
; return signed long int with sign extension on R4
bipolar:
    mov r4,#0
    mov a,adresh
    mov r5,a
    mov r6,adresm
    mov r7,adresl
    jnb acc.7,positive
    mov r4,#0ffh
positive:
    ret
```

6.10 Digital-to-Analog Converter

The MSC120x has one 8-bit current DAC (IDAC) that provides a programmable current source. The IDAC operates on its own voltage reference, which is independent of the ADC voltage reference.

The output current is determined by IDAC (SFR B5h). The full scale output current of this IDAC is approximately 1mA. The equation for the output current is:

$I_{OUT} \approx$ IDAC x 3.9µA

Where IDAC is the value stored in IDAC (SFR B5h).

The output voltage of the IDAC cannot exceed the compliance voltage of AV_{DD} – 1.5V. That means when the current flows through an impedance, the impedance must be low enough so that the voltage does not exceed (AV_{DD} - 1.5V). When AV_{DD} = 5V, the impedance of the load should be less than 3.5kΩ. Setting bit 6 of PDCON (SFR F1h) will shut down the IDAC function, and the IDAC pin will float.

Inter-IC (I2C) Subsystem

This chapter describes the Inter-IC (I^2C) subsystem of the MSC120x.

Topic **Page**

7.1 Introduction to the I²C Bus

The I²C protocol was defined to permit multiple 8-bit transfers between multiple integrated circuits on the same 2-wire bus. At any one time, a bus master coordinates transfers from one slave or to multiple slaves.

For a detailed description of the I²C bus, refer to the I²C bus specification by Philips.

7.2 I²C Terminology

For many systems where the MSC120x is the only microcontroller, it will be the master, and coordinate the transfer of data between itself and slave ICs. If active, it can transmit data onto the I²C bus or receive data from the bus. In either case, it generates the synchronizing clock.

Similarly, where the MSC120x is considered a slave to another microcontroller, it is able to transmit and receive data synchronized by this master.

More than one MSC120x can share a single I²C bus where each acts as a master at different times. The active master can be determined by software or result from bus arbitration in the event of asynchronous contention.

Table 7-1 describes selected I²C terms.

Table 7-1. I²C Terminology

Name	Description
Transmitter	The IC that sends data to the bus
Receiver	The IC that receives data from the bus
Master	The IC that initiates a transfer, generates clock signals, and terminates a transfer
Slave	The IC addressed by a master
Multi-master	More than one master can attempt to control the bus at the same time without corrupting the message
Arbitration	Procedure to ensure that if more than one master simultaneously tries to control the bus, only one is allowed to do so and the message is not corrupted
Synchronization	Procedure to synchronize the clock signals of two or more ICs

7.3 I^2C Bus Lines and Basic Timing

The I^2C bus uses two bidirectional data lines. One is the data line (SDA), and the other is the clock line (SCL). Each is connected to a positive supply voltage via a pull-up resistor, and when the bus is free, both lines are high.

The output stages of I^2C interfaces connected to the bus must have an open drain or open collector to perform the wired-AND function. The original specification for the I^2C bus allowed the data transfer rate to be up to 100kbits/s; however, this has been extended to 400kbits/s in fast mode, which is supported by the MSC120x. In either mode, the maximum rate is determined by the value of the pull-up resistors and the capacitance to ground.

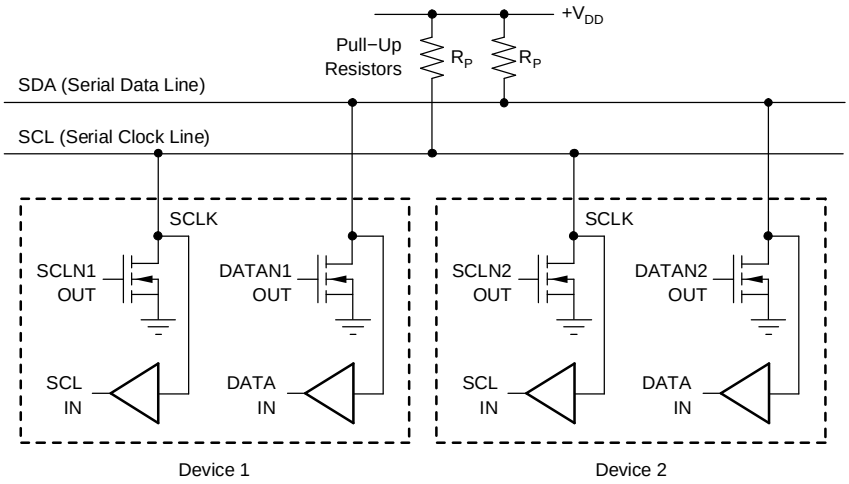


Figure 7-1. I^2C Bus Connection of Standard and Fast Mode Devices

Unique start and stop conditions are identified when SCL is high and SDA changes. If SDA changes from 1 to 0, a start condition is created; if SDA changes from 0 to 1, a stop condition is created. All ICs connected to the bus, including the MSC120x, recognize and respond to start and stop conditions. For a data-bit transfer, SCL is pulsed high while SDA is stable.

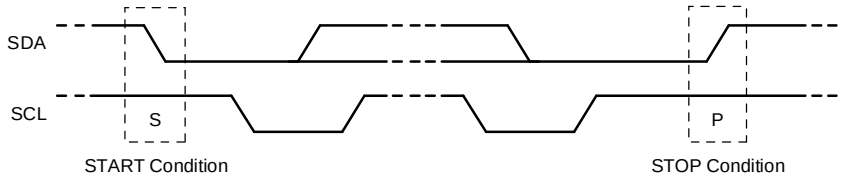


Figure 7-2. Start and Stop Conditions

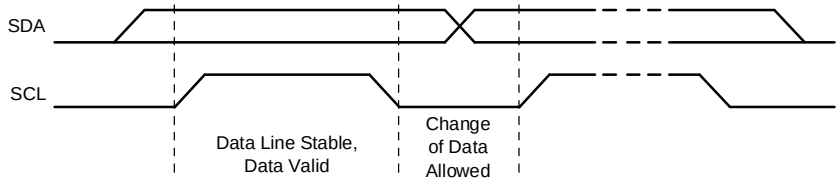


Figure 7-3. I^2C-Bus Bit Transfer

7.4 I²C Data Transfers and the Acknowledge Bit

Once a master asserts a start condition, the bus is no longer free. The master then transmits eight bits comprising of the 7-bit address of the slave followed by a read/write (R/$\overline{W}$) bit. In a system with multiple asynchronous masters, there may be a period of bus contention and arbitration before the address of the slave is transmitted.

If the slave is to receive data, the R/$\overline{W}$ bit must be 0; otherwise, it will prepare to transmit data (since the R/$\overline{W}$ bit is 1). For some I²C devices, such as memories, it is necessary to first write an internal address to the slave and then read or write data bytes. In this case, a start condition can be re-asserted.

When a master has generated eight SCL pulses, it places its own SDA output high and generates a ninth clock pulse. If the addressed slave has responded, it will have pulled the SDA line low; this represents an acknowledgement (ACK). However, if the addressed slave leaves the SDA line high, the master recognizes that the slave has not acknowledged (NACK) the transfer.

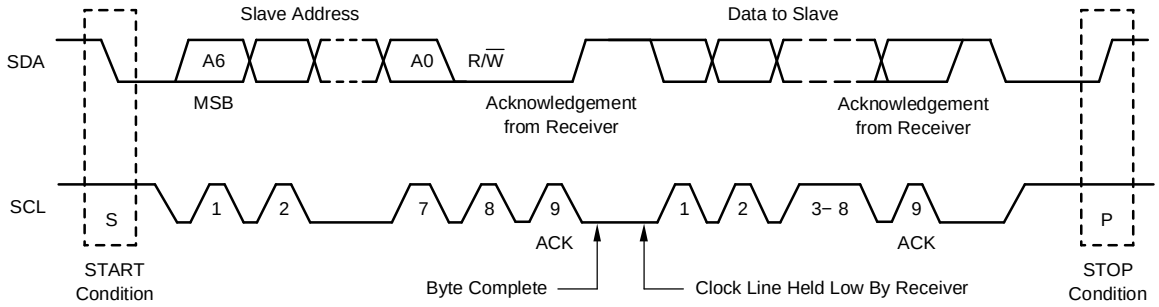


Figure 7-4. I²C-Bus Data Transfer

Once addressed, a multi-byte data transfer can be terminated when a slave generates a NACK rather than the usual ACK. In addition, after the acknowledge bit, a slave may pull the SCL line low while it performs local processing; this often occurs when the slave is a microcontroller that executes a time-consuming interrupt service routine (ISR). While the SCL line is held low, the master will wait.

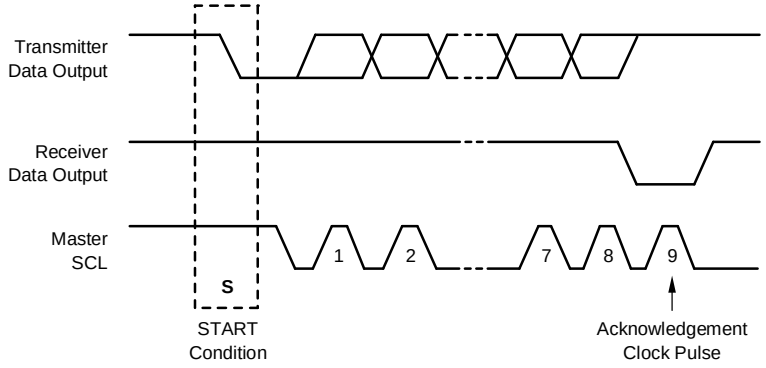


Figure 7-5. I²C Acknowledge

If a master issues a slave address with a R/$\overline{W}$ bit that is 1, it will become a master receiver when the slave responds with an ACK. Thereafter, the slave provides data bytes to the master, but releases the SDA line every ninth clock pulse and samples the acknowledgement that is provided by the master. Typically, the master will generate ACKs for as long as it expects more data, and then generate a NACK to inform the slave on the last byte.

7.5 I^2C Configuration in the MSC120x

In the MSC120x, basic hardware is provided to implement the I^2C protocol. A block diagram of the I^2C interface is shown in Figure 7-6. This hardware is shared by both the SPI™ and I^2C interfaces. Therefore, in the MSC120x, only I^2C or SPI can function at any given time. If both are activated, SPI will be selected.

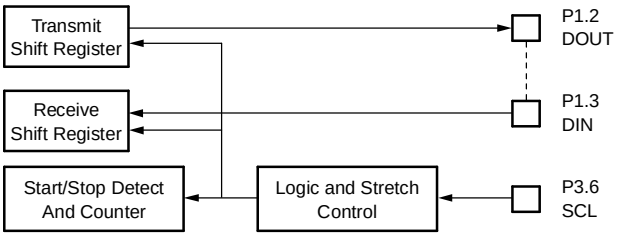


Figure 7-6. I^2C Interface Block Diagram

This basic hardware together with software supports Standard and Fast modes of I^2C data transfer, along with master mode, slave mode, 7-bit addressing, 10-bit addressing, clock stretching, and general call.

For I^2C operation, pins P1.3 (DIN) and P1.2 (DOUT) should be tied to each other externally, and be used as the I^2C data line SDA.

In I^2C mode, there are only two lines: SCL and SDA. The I^2C protocol defines that MSB is always transmitted first, and the first edge is the latching edge.

7.5.1 SCL

SCL is the clock signal that is used for the transfer of bits from one device to another. Pin P3.6 is used as the SCL line. When the MSC120x is used as an I^2C master, the software is responsible for generating the clock on this line. It can use either the port function of that pin or one of the toggle clocks. See Section 7.7, *Clock Generation*, for more information.

7.5.2 SDA

SDA is the data line. Pins P1.3 and P1.2 must be tied together to function as SDA. Pin P1.3 should be configured as an input, and pin P1.2 should be configured in the 8051 or open drain mode.

7.6 I²C Registers

There are two Special Function Registers (SFRs) associated with the MSC120x I²C interface. These SFRs are I2CCON (SFR 9Ah) and I2CDATA (SFR 9Bh).

7.6.1 I2CCON—I ²C Control Register

The I²C Control register (I2CCON) is described in Table 7-2.

Table 7-2. I2CCON—I²C Control Register

I2CCON		SFR 9Ah		Reset Value = 00h
Bit #	**Name**	**Action or Interpretation**		
7-4	SBIT3-0	Serial Bit Count (gray code, read-only). In I²C mode, the count is reset when a start or stop condition is detected, or when data are written to I2CDATA. The serial bit count indicates how many bits have been shifted in or out.		
		SBIT3:0	**Count**	
		0x00	0	
		0x01	1	
		0x03	2	
		0x02	3	
		0x06	4	
		0x07	5	
		0x05	6	
		0x04	7	
		0x0C	8	
3	STOP	STOP Bit Status (read-only) The STOP bit is set to '1' when a stop condition occurs on the I²C bus. It is cleared by writing to the I2CDATA register.		
2	START	START Bit Status (read-only) The START bit is set to '1' when a start or repeated start condition occurs on the I²C bus. It is cleared by writing to the I2CDATA register.		
1	DCS	Disable Clock Stretch. When DCS is set to '0', the clock stretch function is enabled. When DCS is set to '1', the clock on the SCL line is not stretched. DCS is automatically cleared when a start or repeated start condition occurs on the I²C bus.		
0	CNTSEL	Count Select When CNTSEL is '1', CNTIP goes high when the counter value is equal to 1. When CNTSEL is '0', CNTIP goes high when the counter value is equal to 8.		

7.6.2 I²C Control Bits

The I²C Control Bits are DCS, CNTSEL, START, and STOP.

7.6.2.1 Disable Clock Stretch (DCS)

DCS is bit 1 of I2CCON (SFR, 9Ah) and is used for disabling the clock stretch. DCS is automatically cleared on reset, or when a start or a repeated start condition is seen on the bus. The clock is actually stretched during the CNT interrupt signal.

When DCS is set to '1', the clock on the SCL line is not stretched. Also, the device stops counting the clocks and shifting the data, and does not interrupt the CPU for CNT interrupts.

7.6.2.2 Count Select (CNTSEL)

CNTSEL is bit 0 of I2CCON and is used to selects whether the CNT interrupt occurs after eight counts or one count. When this bit is '1', the CNT interrupt (refer to Section 7.6.4, *Other I ²C Registers*) goes high when the counter value is '1'; otherwise, the interrupt will go high when the counter value is equal to '8'.

7.6.2.3 START

The START bit is set whenever a start or a repeated start condition occurs on the I²C bus. This bit is read-only and is cleared by writing to the I2CDATA register. The I²C start/stop interrupt is received when either a start condition or a stop condition is seen on the bus. If the interrupt is enabled (EI2C, bit 3 of AIE, SFR A6h), the I²C start/stop interrupt will be generated.

7.6.2.4 STOP

The STOP bit is set whenever a stop condition occurs on the I²C bus. This bit is read-only, and is cleared by writing to the I2CDATA register. If the interrupt is enabled (EI2C, bit 3 of AIE, SFR A6h), the I²C start/stop interrupt will be generated.

7.6.3 I2CDATA—I²C Data Registers

Table 7-3 shows the I2CDATA SFR. All I²C bytes are written to, or read from, I2CDATA, which consists of two shift registers: transmit shift and receive shift. Both shift registers are addressed by the same address.

Table 7-3. I2CDATA SFR

I2CDATA SFR 9Bh	7	6	5	4	3	2	1	0	Reset Value
Data	MSB							LSB	00h
Address	MSB						LSB	R/$\overline{W}$	00h

Transmit Shift Register—Once the data are written to the transmit shift register, the data are shifted out on the falling edge of SCL. When the I²C subsystem is receiving data, the transmit shift register must be written as 0xFF in order to make sure that SDA is not pulled low.

Receive Shift Register—Data are shifted into the receive shift register on the rising edge of SCL. The data that has been received into the shift register can be read by reading the I2CDATA register.

7.6.4 Other I²C Registers

The I²C interface shares pins and registers with the Serial Peripheral Interface (SPI); both interfaces should not be enabled at the same time via bit 5 (PDI2C) and bit 0 (PDSPI) of Power-Down Control (PDCON) SFR at F1h. If both interfaces are enabled, SPI will be activated and I²C will be disabled. When I²C is powered down (disabled), the I²C finctions on P1.2, P1.3, and P3.6 are disabled.

Table 7-4. PDCON of I²C and SPI

PDCON at F1h		I²C	SPI
Bit 5 = PDI2C	Bit 0 = PDSPI		
0	0	Undefined	Enabled
0	1	Enabled	Disabled
1	0	Disabled	Enabled
1	1	Disabled	Disabled

The I²C interface uses bit 2 (EI2C) of the Auxiliary Interrupt Enable (AIE) SFR at A6h to enable interrupts, as well as bit 2 (I2CSI) of the Auxiliary Interrupt Status Register (AISTAT) SFR at A7h and bit 4 (AI) of the Enable Interrupt Control (EICON) SFR at D8h. Table 7-5 shows a summary of interrupt-replated I²C registers.

Table 7-5. Interrupt Control Registers for I²C

SFR Name	SFR Address	Bit Number	Bit Name	Action or Interpretation
AIPOL	A4h	2	CNTIP	Serial Bit Count Interrupt (before masking) 0: Serial bit count interrput inactive 1: Serial bit count interrupt active
		3	I2CIP	I²C Start/Stop Interrupt (before masking) 0: I²C start/stop interrupt inactive 1: I²C start/stop interrupt active
PAI	A5h	3, 2, 1, 0	PAI3-PAI0	Pending Auxiliary Interrupt Register Read: 0011_2: Indicates I²C interrupt and possible lower priority interrupt pending. 0100_2: Indicates serial bit count interrupt and possible lower priority interrput pending.
AIE	A6h	2	ECNT	Enable Serial Bit Count Interrupt 0: Masked 1: Enabled
		3	EI2C	Enable I²C Start/Stop Interrupt 0: Masked 1: Enabled (shared vector to address 0033h)
AISTAT	A7h	2	CNT	Serial Bit Count Interrupt Status (after masking) 0: Interrupt inactive or masked 1: Interrupt active
		3	I2C	I²C Start/Stop Interrupt (after masking) 0: I²C start/stop interrupt inactive or masked 1: I²C start/stop interrupt active
EICON	D8h	4	AI	Auxiliary Interrupt Flag When PAI indicates that there are no pending auxiliary interrupts (that is, all auxiliary interrupts have been serviced), AI must be cleared by software before exiting the ISR; otherwise, the interrupt will occur again. Setting AI in software generates an auxiliary interrupt, if enabled. 0: No Auxiliary Interrupt detected (default) 1: Auxiliary Interrupt detected
		5	EAI	Enable Auxiliary Interrupt The Auxiliary Interrupt accesses nine different interrupts that are masked by AIE (SFR A6h) and identified by AISTAT (SFR A7h) and PAI (SFR A5h). 0: Auxiliary Interrupt disabled (default) 1: Auxiliary Interrupt enabled

I²C generates two interrupts: the CNT interrupt (AIE.2) and the I²C interrupt (AIE.3).

If CNTSEL is set to '0', the CNT interrupt goes high whenever the counter value is equal to eight (SBIT=0xCh). If CNTSEL is set to '1', the CNT interrupt goes high whenever the counter value is equal to one (SBIT=0x1h). By using these two modes, an interrupt on a count of nine can be achieved. The CNT interrupt is cleared whenever I2CDATA is written to or CNTSEL is changed. Generation of CNT interrupts can be disabled by setting the DCS bit.

The I²C interrupt is generated whenever the hardware detects a start or a stop condition on the bus. The I²C interrupt can be cleared by writing to I2CDATA.

7.7 Clock Generation

When MSC120x is configured as an I^{2}C Master, a clock must be generated for shifting the data in and out. The clock is not generated by the hardware in MSC120x. However, there are many ways of generating the clock by software in Master mode.

Listed below are the most feasible ways of clock generation, including those requiring PASEL register settings (Refer to PASEL register in MSC120x data sheet for more details).

- Toggle P3.6 by setting and clearing the port pin.
- Toggle Memory Write Pulse: In this mode, the clock toggles whenever an external write command is issued.
- T0_Out signal can be used as a clock. A pulse is generated whenever timer0 expires.
- Toggle T0_Out: In this mode, the clock toggles to the next state whenever timer0 expires.
- T1_Out signal can be used as a clock. A pulse is generated whenever timer0 expires.
- Toggle T1_Out: In this mode, the clock toggles to the next state whenever timer1 expires.

NOTE: In I^{2}C protocol, the clock idle value is defined as low. If the toggle clock is in the wrong state (that is not low state), an extra clock must be generated to put the clock into the correct state

7.8 Clock Stretching

The clock line (SCL) is stretched by the slave when a count interrupt occurs. The slave stretches the clock line to give the CPU time to process and load the next data. The stretch can be released only by writing to the I2CDATA register or changing CNTSEL. If additional clock stretch time is needed, the SCL signal will have to be driven low.

Slave address matching must be done in software. When the address does not match that of the MSC1200 slave, then the CPU can set the DCS bit to disable further clock stretching. This also stops the generation of CNT interrupts. The DCS bit is automatically cleared on a start or a repeated start condition. The clock stretching and CNT interrupts will resume after the DCS bit is cleared.

7.9 Start and Stop Conditions

The start and stop conditions can be generated by the master by manipulating the SDA and SCL signals using the port function. To send the start condition, first the P1.2 pin is cleared, and then the P3.6 pin is cleared. A stop condition can also be generated using a similar method. Detection of start and stop conditions in slave mode is done by using the I^{2}C interrupt. Reading the START and STOP bits in I2CCON tells which of the two conditions occurred on the bus.

7.10 Application Flow

The following steps explain the typical application usage flow of I^2C in master and slave modes.

7.10.1 Application Flow for I^2C Master Receiving Data

1. Send the start condition.
2. Load I2CDATA with the slave address: (7 bits) + read/write control.
3. Issue eight clock cycles to send the slave address out.
4. Load I2CDATA with 0xFFh to clear the interrupt flags and also to make sure that the SDA line is not pulled down during the ACK cycle.
5. Issue one more clock cycle for the ACK cycle.
6. Bit 0 of I2CDATA will have the value of ACK cycle. If I2CDATA.0 is '0', then an ACK was received; otherwise, a NACK was received.
7. Load the I2CDATA with 0xFFh so that the master drives the SDA line in standard I/O mode.
8. Issue eight clock cycles to receive the data.
9. Read I2CDATA to get the received data.
10. Write bit 7 of I2CDATA with ‘1’ to transmit NACK, and ‘0’ to transmit ACK.
11. Issue one more clock cycle for the ACK cycle.
12. Repeat steps 7 to 11 until finished.

7.10.2 Application Flow for I^2C Slave Transmitting Data

1. Detect start condition with the I^2C interrupt. Remove the interrupt by writing 0xFF to I2CDATA.
2. Wait for the CNT interrupt (with CNTSEL=0) and then read the data.
3. Compare the addresses to see if the MSC120x is addressed. If the addresses do not match, then load I2CDATA with 0xFF and set DCS.
4. If the addresses match, then change CNTSEL to ‘1’ and load I2CDATA with bit 7 equal to ‘1’ for NACK and ‘0’ for ACK.
5. On the next CNT interrupt, change CNTSEL = 0, and load I2CDATA with the byte to be transmitted.
6. On the next CNTIRQ, change CNTSEL = 1, and load I2CDATA with bit 7 equal to ‘1’ for NACK and ‘0’ for ACK.
7. Repeat steps 5 and 6 until finished.

7.11 I²C Synchronization and Arbitration

Byte-level synchronization is achieved when an MSC120x, acting as a slave, holds SCL low after the ninth bit of any byte transferred. However, bit-level synchronization is also supported when an MSC120x is configured as a master, and a slave pulls SCL low after each bit. In effect, it will pause if it senses a low level on SCL when it should be high. More generally, the SCL clock has a low period determined by the device with the longest clock low period, and a high period determined by the device with the shortest high period.

In a system with multiple masters, there is a chance that two or more masters will attempt to place data on SDA at the same time. When one master attempts to transmit a high level while another is already transmitting a low level, it will disable its output stage and relinquish control of SDA. The winning master is left to control the bus and pass error-free data.

7.12 I²C 10-Bit Addressing

The MSC120x supports both 7-bit and 10-bit addressing defined by I²C protocol. Table 7-6 shows I²C protocol 7-bit and 10-bit addressing.

Table 7-6. Address Allocation

Most Significant Seven bits	R/$\overline{W}$	Standard Meaning	Extended Meaning, Where Different
0000 000	0	General call	
0000 000	1	Start byte for slow devices	
0000 001	x	Address for CBUS protocol	
0000 010	?	Address reserved for different protocol	
0000 011 to 0000 111	x	To be defined	
0001 000 to 1110 111	R/$\overline{W}$	I²C device addresses	
1111 0aa	R/$\overline{W}$	Reserved	Most significant two bits of 10-bit address
1111 1xx	x	Reserved	

To increase the number of addressable devices, the I²C standard was extended to accommodate an additional 10-bit address space. The most significant two bits are contained within addresses that were originally reserved, while the remaining eight bits are provided in the following byte. The MSC120x neither generates nor accepts 10-bit addresses automatically. However, the 10-bit addressing protocol may be replicated under software control to implement either a master transmitter or a slave receiver. A master receiver or slave transmitter cannot be implemented because the interpretation of the R/$\overline{W}$ flag precludes transmission or reception (respectively) of the low part of the address as a data byte.

Serial Peripheral Interface (SPI)

This chapter describes the serial peripheral interface (SPI) of the MSC120x.

Topic **Page**

8.1 Description

The Serial Peripheral Interface, or SPI, is a synchronous, serial, full-duplex communications standard that simultaneously transfers eight bits of data from a master to a slave, and another eight bits of data from the slave to the master. The MSC120x has implemeted a basic SPI hardware that executes almost all of the standard SPI features. It supports full-duplex mode. The clock phase, polarities, and the order of transfer are programmable. The MSC120x can be programmed to behave as a master or a slave and uses four signals to coordinate transfers. These signals are:

1. DOUT—Serial Data Out (shared with P1.2)
2. DIN—Serial Data In (shared with P1.3)
3. SCK—SPI Clock (shared with P3.6/SCK/SCL/CLKS)
4. $\overline{SS}$—Slave Select (shared with P1.4/INT2)

8.1.1 DOUT

DOUT (master out/slave out) is the serial output data line. DOUT is made to float when the SPI is powered down or the slave is not selected.

8.1.2 DIN

DIN (master in/slave in) is the serial input data line that receives the data.

8.1.3 SCK

SCK (serial clock) is generated by the master device and synchronizes data movement in and out of the device through the DIN and DOUT lines. Master and slave devices are capable of exchanging a byte of information during a sequence of eight clock cycles. When the device is configured as master, the software has to generate the clock on the SCK pin (P3.6).

8.1.4 $\overline{SS}$

$\overline{SS}$ (slave select) is controlled by the master to select the slave. It is a low active signal. $\overline{SS}$ must be low before data transactions and must stay low for the duration of the transaction. When $\overline{SS}$ is high, the data is neither shifted into the shift register nor the counter increments. $\overline{SS}$ also controls the DOUT pin (P1.2). When $\overline{SS}$ is low, DOUT is driven according to the transmit shift register; when $\overline{SS}$ is high, the DOUT pin is put to high impedance.

8.2 SPI Configuration

The SPI function shares hardware with the I^2C function. Therefore, SPI and I^2C cannot be enabled at the same time. If SPI and I^2C are both enabled, SPI is activated and I^2C is disabled. SPI is enabled by PDSPI (bit 0 of PDCON).

The SPI is configured by SPICON at 9Ah, according to Table 8-1.

Table 8-1. SPICON—SPI Control

SPICON		SFR 9Ah		Reset Value = 00h
Bit #	**Name**	**Action or Interpretation**		
7-4	SBIT3-0	Serial Bit Count (gray code, read-only). In SPI mode, these bits are cleared when reading or writing data to SPIDATA (SFR 9Bh).		
		SBIT3:0	**Count**	
		0x00	0	
		0x01	1	
		0x03	2	
		0x02	3	
		0x06	4	
		0x07	5	
		0x05	6	
		0x04	7	
		0x0C	8	
3	ORDER	Sets bit order for transmit and receive. 0: Most significant bit first 1: Least significant bit first		
2	CHPA	Serial Clock Phase Control 0: Valid data starting from half SCK period before the first edge of SCK 1: Valid data starting from the first edge of SCK		
1	ESS	Enable Slave Select 0: SS is configured as general-purpose I/O 1: $\overline{SS}$ (P1.4) is configured as slave select for SPI mode. DOUT (P1.2) drives when $\overline{SS}$ is low. DOUT is high-impedance when $\overline{SS}$ is high.		
0	CPOL	CPOL serial clock polarity 0: SCK idle at logic LOW 1: SCK idle at logic HIGH		

8.2.1 Serial Bit Count (SBIT3-0)

SBIT 3-0 is a 4-bit value that indicates the number of bits shifted in/out so far. A gray code counter is used to avoid glitches. During SPI mode, the counter is reset when any read or write is performed to the SPIDATA register. During I^2C mode, the counter is reset whenever a start condition is detected or data is written to the SPIDATA register. The interpretation of the bit count is given in Table 8-1.

8.2.2 ORDER

Software can also set the transmission order of the byte (either LSB first or MSB first) by writing to the ORDER bit in the SPICON register. This gives the device the flexibility to talk to devices that implement both transfer orders. Setting the ORDER bit transmits the LSB first; otherwise, it will be MSB first.

8.2.3 Clock Phase (CPHA) and Clock Polarity (CPOL)

Software can select one of four combinations of clock phase and clock polarity using two bits (CPHA and CPOL) in the SPI control register (SPICON). The clock polarity is specified by the CPOL control bit, which selects an active high or active low clock, and has no significant effect on the transfer format. The clock phase (CPHA) control bit selects one of two different transfer formats. Figure 8-1 shows the SPI clock/data timing diagram for different values of CPHA and CPOL.

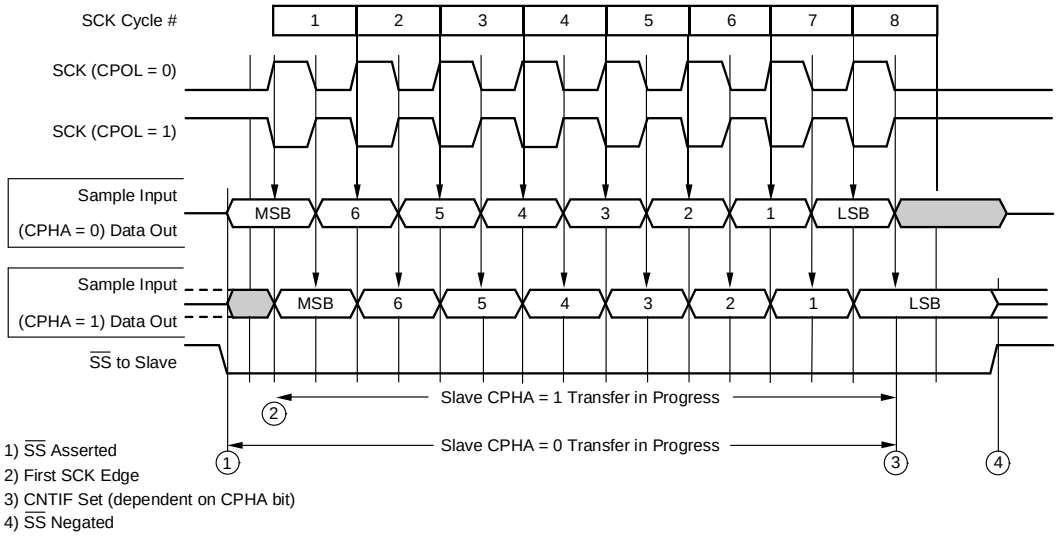


Figure 8-1. SPI Clock/Data Timing

8.2.4 Enable Slave Select (ESS)

ESS is bit 1 of SPICON and is used to enable the slave select feature. When ESS is set to '0', the slave select feature is disabled, SS (Pin 1.4) and DOUT are configured as general I/O. When ESS is set to '1', DOUT value depends on SS pin (P1.4). When SS is high, DOUT is in high impedance and when SS is low, DOUT is driven according the transmit shift register output.

8.3 Shift Registers and Clock Generation

The hardware includes two shift registers: transmit and receive. The transmit shift and receive shift registers share the same address (SPIDATA, SFR9Bh).

8.3.1 Transmit Shift Register

Writing to the SPIDATA register puts the data into the transmit shift register. The transmit shift register is an 8-bit register. The shifting direction is determined by the ORDER bit in the SPICON register, which defines whether the data is shifted out from bit 7 or bit 0.

8.3.2 Receive Shift Register

The data on the DIN pin is shifted into the receive shift register, and is read by reading the SPIDATA register.

Table 8-2. SPIDATA—SPI Data Register

SPIDATA		SFR 9Bh	Reset Value = 00h
Bit #	**Name**	**Action or Interpretation**	
7-0	SPIDATA	Write/Read: Data to be transmitted by, or received from, the Serial Peripheral Interface.	

8.4 SPI Clock Generation

When configured as master, the SPI must generate the clock for the shifting of data in and out. The clock is not generated by hardware in the MSC120x; it must be generated by the software. There are many ways of generating the clock in master mode. Listed are some ways for clock generation. Some of these require PASEL register settings (Refer to the PASEL register in the MSC120x data sheet for more details).

- Toggle P3.6 by setting and clearing the port pin. This is the easiest way and can be used for both the clock polarities, but can tie up the processor unless it is interrupt driven.
- Memory Write Pulse ($\overline{WR}$), which is idle high. Whenever an external memory write command is issued, a pulse is seen on P3.6. This can be used only when CPOL is set to '1'.
- Toggle Memory Write Pulse: In this mode, the clock toggles whenever an external write command is issued. This can be used for all clock modes since an additional toggle can be sent if the level is in the wrong state.
- T0_Out signal can be used as a clock. A pulse is generated whenever timer0 expires. The idle state of the signal is low. This can be used only if CPOL to '0'.
- Toggle T0_Out: In this mode, the clock toggles to the next state whenever timer0 expires. This can be used for all clock modes since an additional toggle can be sent if the level is in the wrong state.
- T1_Out signal can be used as a clock. A pulse is generated whenever timer0 expires. The idle state of the signal is low. This can be used only if CPOL to '0'.
- Toggle T1_Out: In this mode, the clock toggles to the next state whenever timer1 expires. This can be used for all clock modes since an additional toggle can be sent if the level is in the wrong state.

To select any of the three sources, the PASEL (SFR, F2h) must be set accordingly. Refer to the MSC120x data sheet for more information.

8.5 SPI Interrupts

When an SPI interrupt is enabled and active, the MSC120x CPU jumps to location 0033h. The interrupt service routine (ISR) can read the Pending Auxiliary Interrupt Register (PAI at A5h) or the Auxiliary Interrupt Status (AISTAT, SFR A7h) to establish the source of the interrupt.

The SPI generates the CNT interrupt (AIE.2) to indicate that the transfer/reception of the byte is done. The interrupt goes high whenever the counter value is equal to eight (indicating that the eight bits have been transferred). The interrupt is cleared on reading or writing to the SPIDATA register.

AI (bit 4 of EICON, SFR D8h), must be cleared within the ISR when no further auxiliary interrupts are pending. Setting AI in software generates an Auxiliary Interrupt, if enabled, but if there are no pending interrupts the Pending Auxiliary Interrupt vector in PAI at A5h will read as 0.

A byte is only received when one is transmitted because of the bit-synchronous nature of the SPI. Consequently, if a master expects a reply that is dependent upon the byte it sent to a slave, it must ignore the first byte returned and transmit dummy bytes to receive subsequent reply bytes.

Note: If SPIDATA is not read before the next SPI data byte is received or transmitted, the CNT interrupt will be cleared and the previous data will be lost.

Table 8-3. SPI Interrupt Registers

Bit Name	Abbreviation	Name of Related SFR	Abbreviation	Address
Enable Auxiliary Interrupt	EAI	Enable Interrupt Control	EICON.5	D8h
Auxiliary Interrupt Flag	AI	Enable Interrupt Control	EICON.4	D8h
Enable Serial Bit Count Interrupt	ECNT	Auxiliary Interrupt Enable	AIE.2	A6h
CNT Interrupt Status Flag	CNT	Auxiliary Interrupt Status	AISTAT.2	A7h
Serial Bit Count Interrupt Poll	CNTIP	Auxiliary Interrupt Poll	AIPOL.2	A4h

Table 8-4. PAI—Pending Auxiliary Interrupt Register

PAI		SFR A5h Reset Value = 00h
Bit #	**Name**	**Interpretation When Read**
7-4	—	Return 0
3	PAI3	Auxiliary Interrupt Status 0000: No Pending Auxiliary Interrupt 0001: Reserved 0010: Analog Low-Voltage Detect Interrupt and Possible Lower Priority Interrupt Pending 0011: I^2C Interrupt and Possible Lower Priority Interrupt Pending 0100: Serial Bit count Interrupt and Possible Lower Priority Interrupt Pending 0101: One Millisecond System Timer Interrupt and Possible Lower Priority Interrupt Pending 0110: ADC Interrupt and Possible Lower Priority Interrupt Pending 0111: Summation Interrupt and Possible Lower Priority Interrupt Pending 1000: One Second System Timer Interrupt
2	PAI2	
1	PAI1	
0	PAI0	

8.6 Application Flow

The following steps explain the typical application usage flow of SPI in master and slave modes.

8.6.1 Master Mode Application Flow

1. Configure the ports and SPI.
2. Assert for the desired slave, optional.
3. Write data to SPIDATA.
4. Issue eight clocks.
5. Read the data from SPIDATA.
6. Repeat steps 3 to 6 until finished.

8.6.2 Slave Mode Application Flow

1. Configure the ports and SPI.
2. Write data to SPIDATA.
3. Wait for the eight clocks completion interrupt.
4. Read the data from SPIDATA.
5. Repeate steps 2 to 5 until finished.

Timers and Counters

This chapter describes the MSC120x timers and counters.

Topic **Page**

9.1 Description

The MSC120x includes two Timer/Counter modules, 0 and 1, that behave in the same way as those found in a standard 8051/8052 microcintroller. When a module is clocked from the system clock, it changes at a known rate; this mode is referred to as a timer. However, when clocked from an external source, it may be considered as an event counter or timer. There are numerous modes of operation, which include but are not limited to:

- 13-bit timer/counter (mode 0)
- 16-bit timer/counter (mode 1)
- 8-bit timer/counter with auto-reload (mode 2)
- Two 8-bit counters (mode 3, Timer 0 only)
- Baud rate generator

Note: Not all modes are available within each module, but a combination of modes satisfies many application environments.

9.2 Timer/Counters 0 and 1

Bits in TMOD at 89h and TCON at 88h configure the operation of Timer/Counter 0 and Timer/Counter 1. They have identical relative behavior in modes 0, 1, and 2, but differ in mode 3, as expressed in the following tables and figures.

Table 9-1. TMOD—Timer Mode Control

TMOD		SFR 89h — Reset Value = 00h
Bit #	**Name**	**Action or Interpretation**
7	GATE	Timer/Counter 1 Gate Control Write: 0: Operation of Timer/Counter 1 does not depend upon pin P3.3/$\overline{INT1}$ 1: Pin P3.3/$\overline{INT1}$ has to be '1' to enable clocking. See TR1 (bit 6 of TCON at 88h)
6	C/$\overline{T}$	Timer/Counter 1 Select Write: 0: Timer/Counter is clocked at f_{CLK}/12 (default) or f_{CLK}/4. See CKCON.4 at 8Eh 1: Timer/Counter is clocked from pin P3.5/T1. See also TR1 (bit 6 of TCON at 88h)
5	M1	Timer/Counter 1 Mode Select 00 (Mode 0): 13-bit counter 01 (Mode 1): 16-bit counter 10 (Mode 2): 8-bit counter with auto reload 11 (Mode 3): Timer/Counter 1 is halted, but holds its count. Same effect as clearing TR1.
4	M0	
3	GATE	Timer/Counter 0 Gate Control Write: 0: Operation of Timer 1 does not depend upon pin P3.2/$\overline{INT0}$ 1: Pin P3.2/$\overline{INT0}$ has to be '1' to enable clocking. See TR0 (bit 4 of TCON at 88h).
2	C/$\overline{T}$	Timer/Counter 0 Select Write: 0: Timer/Counter is clocked at f_{CLK}/12 (default) or f_{CLK}/4. See CKCON.3 at 8Eh. 1: Timer/Counter is clocked from pin P3.4/T0. See TR0 (bit 4 of TCON at 88h).
1	M1	Timer/Counter 0 Mode Select 00 (Mode 0): 13-bit counter 01 (Mode 1): 16-bit counter 10 (Mode 2): 8-bit counter with auto reload 11 (Mode 3): Timer/Counter 0 acts as two independent 8-bit Timer/Counters.
0	M0	

Table 9-2. TCON—Timer/Counter Control

TCON		SFR 88h Reset Value = 00h
Bit #	**Name**	**Action or Interpretation**
7	TF1	Timer 1 (Interrupt) Overflow Flag Read: 0: No Overflow 1: Timer 1 reached the maximum count and changed to 0 Write: 0: Clear flag 1: Set flag and generate interrupt request if unmasked Cleared in software by writing 0, or cleared automatically as the processor jumps to the ISR at 001Bh
6	TR1	Timer 1 Run Control Write: 0: Timer 1 cannot be clocked 1: Timer 1 may be clocked
5	TF0	Timer 0 (Interrupt) Overflow Flag Read: 0: No Overflow 1: Timer 0 reached maximum count and changed to 0 Write: 0: Clear flag 1: Set flag and generate interrupt request if unmasked Cleared in software by writing 0, or cleared automatically as the processor jumps to the ISR at 000Bh
4	TR0	Timer 0 Run Control Write: 0: Timer 0 cannot be clocked 1: Timer 0 may be clocked
3 (1)	IE1	Interrupt 1 Edge Detect If $\overline{INT1}$ is edge-sensitive because IT1 = 1, IE1 is set when a negative edge is detected. It is cleared when the CPU jumps to the ISR at 0013h or by writing 0 in software. If IT1 = 0, IE1 is set when the $\overline{INT1}$ pin is low, and cleared when the $\overline{INT1}$ pin is high.
2	IT1	Interrupt 1 type select Write: 0: $\overline{INT1}$ is sensitive to a low level 1: $\overline{INT1}$ is sensitive to a negative (falling) edge
1	IE0	Interrupt 0 edge select If $\overline{INT0}$ is edge-sensitive because IT0 = 1, IE0 is set when a negative edge is detected. It is cleared when the CPU jumps to the ISR at 0013h or by writing 0 in software. If IT0 = 0, IE1 is set when the $\overline{INT0}$ pin is low, and cleared when the $\overline{INT0}$ pin is high.
0 (1)	IT0	Interrupt 0 type select Write: 0: $\overline{INT0}$ is sensitive to a low level 1: $\overline{INT0}$ is sensitive to a negative (falling) edge

(1) Bit 0 to bit 3 of TCON are not associated with the operation of any Timer/Counter.

9.2.1 Modes 0 and 1

The description that follows is with respect to Timer/Counter 0. but also applies to Timer/Counter 1 with the appropriate re-allocation of control bits. However, *only* the overflow condition of Timer 1 is able to act as a reference clock for the serial ports.

TH0:TL0 represents a 13-bit, negative-edge triggered up counter that can be clocked from a variety of sources. When C/$\overline{T}$ is 0, it behaves as a gated timer running at either $f_{CLK}/12$ (default) or $f_{CLK}/4$. However, when C/$\overline{T}$ is 1, it behaves as a gated event counter, where appropriate transitions on pin T0, TR0, GATE, or pin $\overline{INT0}$ cause it to increment.

In mode 0, the upper three bits of TL0 are undefined and should not be used.

When TH0 overflows from FFh to 00h, the interrupt flag (TF0) is set. It is cleared automatically as the CPU jumps to the interrupt service routine (ISR) at 000Bh, or cleared manually by writing a '0' to it in software.

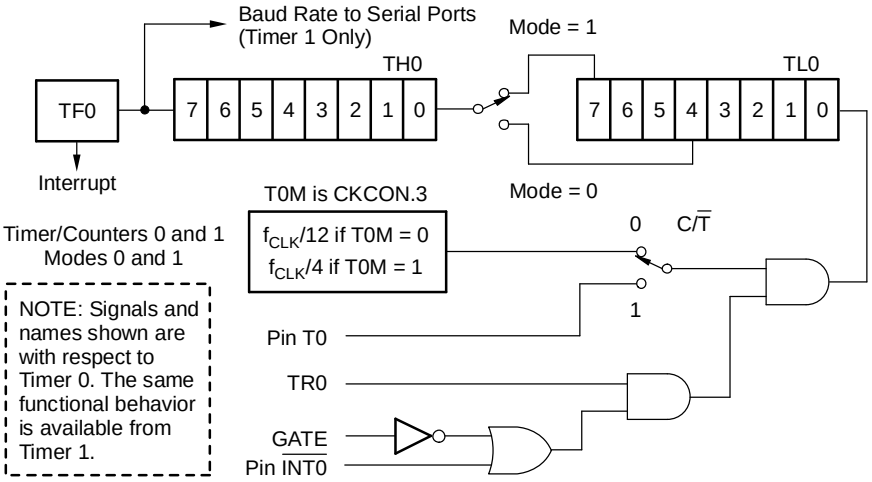


Figure 9-1. Timer 0/1—Modes 0 and 1

Table 9-3. Modes 0 and 1 Operation[(1)]

C/$\overline{T}$	T0M	Pin T0	TR0	GATE	Pin $\overline{INT0}$	CLOCK
0	0	x	1	0	x	$f_{CLK}/12$
0	0	x	1	1	1	$f_{CLK}/12$
0	1	x	1	0	x	$f_{CLK}/4$
0	1	x	1	1	1	$f_{CLK}/4$
1	x	1 to 0	1	0	x	Increment
1	x	1 to 0	1	1	1	Increment
1	x	1	1 to 0	0	x	Increment
1	x	1	1 to 0	1	1	Increment
1	x	1	1	0 to 1	0	Increment
1	x	1	1	1	1 to 0	Increment

(1) For all other combinations of control bits and pins, TH0:TL0 is unchanged.

9.2.2 Mode 2

The description that follows is with respect to Timer/Counter 0, but applies to Timer/Counter 1 with appropriate re-allocation of control bits. However, *only* the overflow condition of Timer 1 is able to act as a clock for the serial ports.

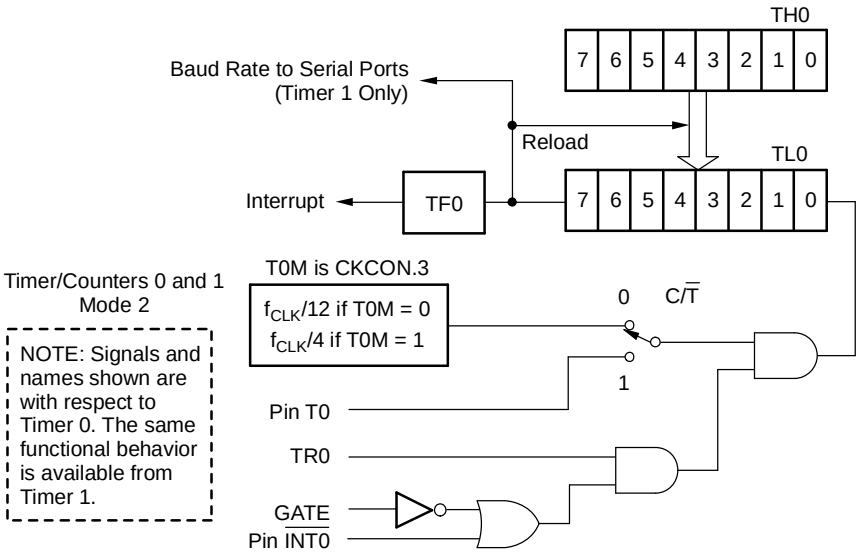


Figure 9-2. Timer 0/1—Mode 2

TL0 represents an 8-bit, negative-edge triggered counter that is reloaded from TH0 as it overflows. It may be from clocked from a variety of sources as described for modes 0 and 1.

9.2.3 Mode 3

The behavior of Timer/Counter 0 in mode 3 is not the same as that of Timer/Counter 1 because the interrupt flag usually associated with Timer 1 is controlled by TH0.

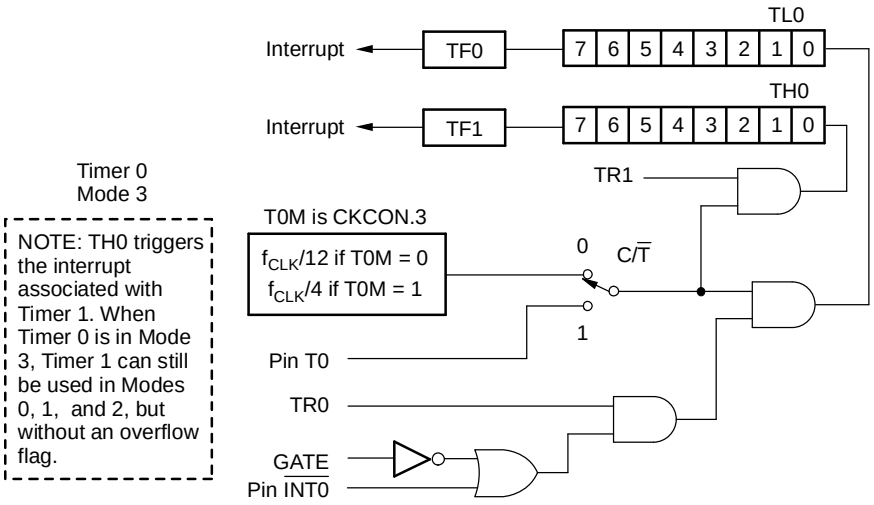


Figure 9-3. Timer 0—Mode 3

TL0 is an 8-bit timer or counter that is clocked and gated in a manner similar to Mode 0. TH0 must be clocked from the same source but is gated only by control bit TR1. Without TR1 and TF1, Timer 1 can still be used for baud rate generation.

9.2.4 Summary of Control Bits and SFRs for Timer/Counters 0 and 1

Table 9-4. Control Bit and SFR Summary for Timer/Counters 0 and 1

Signal, Control, or Data		Timer 1			Timer 0		
		Name or Bit	SFR Address	SFR Bit Address	Name or Bit	SFR Address	SFR Bit Address
Timer overflow flag	TFx	TCON.7	88h	8Fh	TCON.5	88h	8Dh
Count high byte		TH1	8Dh		TH0	8Ch	
Count low byte		TL1	8Bh		TL0	8Ah	
Timer/Counter select	C/$\overline{T}$	TMOD.6	89h		TMOD.2	89h	
Mode bit 1	M1	TMOD.5	89h		TMOD.1	89h	
Mode bit 0	M0	TMOD.4	89h		TMOD.0	89h	
Divide by 4 or 12 select	TxM	CKCON.4	8Eh		CKCON.3	8Eh	
External clock input	Tx	P3.5/T1	B0h	B5h	P3.4/T0	B0h	B4h
Timer run control	TRx	TCON.6	88h	8Eh	TCON.4	88h	8Ch
Internal timer gate	GATE	TMOD.7	89h		TMOD.3	89h	
External timer gate	$\overline{INTx}$	P3.3/$\overline{INT1}$	B0h	B3h	P3.2/$\overline{INT0}$	B0h	B2h
Enable interrupt	ETx	IE.3	A8h	ABh	IE.1	A8h	A9h
Interrupt priority	PTx	IP.3	B8h	BBh	IP.1	B8h	B9h

9.3 Baud Rate Generator

Timer 1 can be used to generate a baud rate for serial port 0. Figure 9-4 shows a baud-rate generation diagram. The best method is to use Timer 1, mode 2 (8-bit counter with auto-reload), although any counter mode can be used. The baud rate is determined by the equation:

$$\text{Baud Rate (K bit)} = \frac{3^{T1M} \times 2^{SMOD}}{384} \times \frac{f_{CLK}}{256 - TH1}$$

Where:

T1M = bit 4 of CKCON (SFR 8Eh)

SMOD = bit 7 of PCON (SFR 87h)

$$TH1 = 256 - \frac{3^{T1M} \times 2^{SMOD} \times f_{CLK}}{384 \times \text{Baud Rate (K bit)}}$$

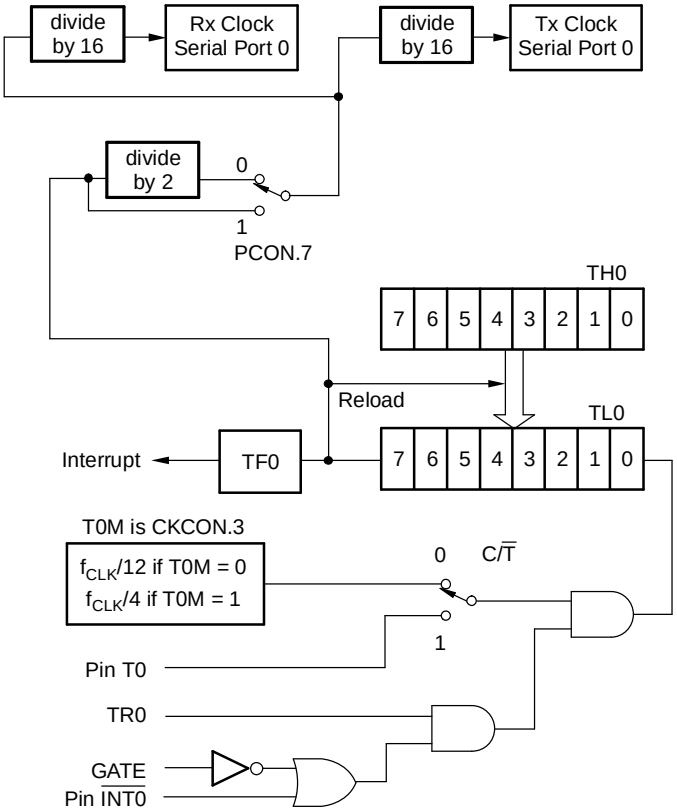


Figure 9-4. Baud Rate Generation

9.4 Example Program Using Timers 0 and 1

In Example 9-1, the yellow LED on the MSC1200 EVM is associated with overflow interrupts from Timer 0.

f_{CLK} is the same frequency as the external (crystal) oscillator, unless the System Clock Divider SFR (SYSCLK at C7h) is present and active. The default value of SYSCLK in the MSC120x will cause no division of the external clock.

Table 9-5. Timer Modes

Timer	Mode	Type	Clock	Overflow or Baud Rate (22.1184MHz Clock)
0	0	13-bit	$f_{CLK}/12$	$\frac{1}{12} \times \frac{f_{CLK}}{8192} = \frac{22118400}{98304} = 225$
1	2	8-bit reload	$f_{CLK}/12$	$\frac{1}{2} \times \frac{1}{16} \times \frac{f_{CLK}}{12 \times (256 - TH1)} = \frac{22118400}{2304} = 9600$ Baud Where TH1 = 250

Example 9-1. Program Using Timers 0 and 1

```
// File Timer01.c
// Timer 0 in 13-bit mode,and Timer 1 in 8-bit reload
// MSC1200 EVM jumper setting as factory default
#include <Reg1200.h>
#include <stdio.h>

#define fclk 22118400
#define BAUD 9600
#define limit 225

sbit RedLed = P3^4;           // RED LED on EVM
sbit YellowLed = P3^5;        // Yellow LED on EVM
data unsigned int i=1, j=1;

void Timer0Int(void) interrupt 1 using 1
{   if(!--i)
    {   i=limit;
        YellowLed=!YellowLed;
    }
}
void main(void)
{   CKCON = 0x00;               // Timers 0,1 at fclk/12
    PCON = 0x30;                // SMOD = 0
    TMOD = 0x20;                // Timer 1 Auto reload; Timer 0 13-bit
    TCON = 0x50;                // TR1 and TR0 are 1
    TH1 = 256 - fclk/32/12/BAUD; // Timer 1 reload value
    SCON0 = 0x52;               // Asynchronous, TI_0=1, RI_0=0
    while(!RI_0);               // wait for key press
    printf("\nMSC1200 Timer 0 in mode 0, Timer 1 in mode 2\n");
    IE =0x82;                   // EA, ET1 interrupt enable
    while(1){
        while(!RI_0);           // wait for key press
        SBUF0=SBUF0;            // echo
        RI_0=0;
    }
}
hrc.a51
    CSEG AT 0803DH
    DB   0FBH ; 3DH PLLCON{2:0] 011-For External Crystal
    DB   00BH ; 3EH DBOD[7:4],2-BODEn
    DB   0FFH ; 3FH 7-Security,6-PML0CK,5-RSTLOCK,4-ENBR,3-EWDR,,2-NU1-0:DFM
    END
```

Serial Port (USART0)

This chapter describes the serial port of the MSC120x.

Topic **Page**

10.1 Description

The MSC120x has one serial port. Serial port 0 (USART0) can be configured in synchronous or asynchronous modes and clocked via f_{CLK}, the overflow from Timer/Counter 1. USART stands for *Universal Synchronous/Asynchronous Receiver/Transmitter*.

The serial port has a control register and a data register, referenced as SCON0 at 98h and SBUF0 at 99h.

10.2 Control Bits in SCON0

Table 10-1. SCON0—Serial Port 0 Control Register

SCON0		SFR 98h						Reset Value = 00h
		Action or Interpretation						
Bit #	Name	SM0	SM1	SM2	Function	Mode	Length	Rate(1)
7	SM0_0	0	0	0	Synchronous	0	8 Bits	f_{CLK} /12
		0	0	1	Synchronous	0	8 Bits	f_{CLK} /4
		0	1	0	Asynchronous	1	10 Bits	Timer 1
				1(2)	Asynchronous		10 Bits	
6	SM1_0	1	0	0	Asynchronous	2	11 Bits	$\frac{2^{SMOD}}{64} \times f_{CLK}$
				(3)1(4)	Asynchronous (Multiprocessor)		11 Bits	
5	SM2_0	1	1	0	Asynchronous	3	11 Bits	Timer 1
		1	1	1(4)	Asynchronous (Multiprocessor)	3	11 Bits	
4	REN_0	Receive Enable Write: 0: receive shift register is disabled 1: receive shift register is enabled (for mode 0, RI = 0 is also required)						
3	TB8_0	Ninth Transmission Bit State The state of the ninth bit to be transmitted in modes 2 and 3						
2	RB8_0	Ninth Received Bit State The state of the ninth bit received in modes 2 and 3. In mode 1, when SM2 = 0, RB8_0 is the state of the stop bit. RB8_0 is not used in mode 0.						
1	TI_0	TI_0 Transmitter Interrupt Flag This bit is set when the transmit buffer has been completely shifted out. In mode 0, this occurs at the end of the eighth data bit, while in all other modes it is set at the beginning of the STOP bit. This flag must be manually cleared by software and can be set in software to cause an interrupt.						
0	RI_0	RI_0 Receiver Interrupt Flag This bit indicates that a byte has been received in the input shift register. In mode 0, it is set at the end of the eighth data bit; in mode 1, after the last sample of the incoming stop bit; and in modes 2 and 3, after the last sample of the ninth data bit. This bit must be manually cleared by software and can be set in software to cause an interrupt.						

(1) If IDLE (bit 0 of PCON at 87h) is set, the CPU, Timer/Counters 0 and 1, and the serial port will freeze until there is an auxiliary interrupt or external wake-up (see AIE at A6h, EICON at D8h and EWU at C6h).
(2) In mode 1 with SM2 = 1, RI is activated only if a valid stop bit is received.
(3) SMOD0 is bit 7 of PCON at 87h.
(4) In modes 2 and 3 with SM2 = 1, RI is activated only if the ninth received data bit is 1.

10.3 Pin and Interrupt Assignments

Table 10-2. USART Pin and Interrupt Assignments

Function	USART 0 : SBUF0 at 99h		
	Rx Pin	**Tx Pin**	**Clock**
Mode 0[1] Transmit triggered by write to SBUF	—	P3.0	P3.1
Mode 0[1] Receive triggered by REN = 1 and RI = 0	P3.0	—	P3.1
Modes 1, 2, and 3 Transmit triggered by write to SBUF with TI = 1 Received data read via SBUF when RI = 1 and REN = 1.	P3.0	P3.1	—
	Name	**SFR Address**	**SFR Bit Address**
Interrupt Enable	IE.4	A8h	ACh
Interrupt Priority	IP.4	B8h	BCh

(1) In mode 0, the Rx pin is used to receive and transmit synchronous data. Consequently, the corresponding data direction bits should be defined as input, output, or bidirectional, as appropriate.

10.4 Timer/Counter 1 Baud-Rate Generation

In asynchronous modes 1 and 3, the overflow rate of Timer/Counter 1 can determine the receive or transmit baud rate for serial port 0.

The overflow of all Timer/Counters passes through a fixed divide-by-16 counter (see Figure 9-4, *Baud Rate Generation*) that is reset when a start condition is identified. By default, the overflow output of Timer/Counter 1 is also divided by 2, but this may be avoided if SMOD = 1 (bit 7 of PCON at 87h).

See Table 10-3 for For Timer/Counter 1 baud-rate generation.

Table 10-3. Timer/Counter 1 Baud Rate Generation

Mode of Timer/Counter 1 Determined by TMOD at 89h		Timer/Counter 1 Overflow Rate When Clocked Internally TMOD.6 = 0		Timer/Counter 1 Overflow Rate When Clocked Externally TMOD.6 = 1
M1 (bit 5)	M0 (bit 4)	CKCON.4 = T1M = 0	CKCON.4 = T1M = 1	CKCON.4 = T1M = 0 or 1
0	0	$f_{CLK}/(12 \times 8192)$	$f_{CLK}/(4 \times 8192)$	$f_{T1}/8192$[1]
0	1	$f_{CLK}/(12 \times 65536)$	$f_{CLK}/(12 \times 655362)$	$f_{T1}/65536$[1]
1	0	$f_{CLK}/(12 \times [256 - TH1])$	$f_{CLK}/(4 \times [256 - TH1])$	$f_{T1}/(256 - TH1)$[1]
1	1	Stopped	Stopped	Stopped

(1) f_{T1} is the frequency of the signal at pin P3.5/T1.

Table 10-4. USART Baud Rate Generation

Serial Port #	Timer #	Conditions	Baud Rate
0	1	T2CON = $xx00xxxx_2$; PCON.7 = SMOD0 = 1; TCON = 01000000_2; TMOD = $0100xxxx_2$; f_{T1} = 19.660800MHz.	$= \frac{1}{16} \times \frac{f_{T1}}{8192} = \frac{19660800}{131072} = 150$
0	1	T2CON = $xx00xxxx_2$; PCON.7 = SMOD0 = 0; CKCON.4 = T1M = 0; TCON = 01000000_2; TMOD = $0010xxxx_2$; TH1 = 253; f_{CLK} = 11.059200MHz	$= \frac{1}{2} \times \frac{1}{16} \times \frac{f_{CLK}}{12 \times (256 - TH1)} = \frac{11059200}{384 \times 3} = 9600$

10.5 Mode 0 (8-Bit Synchronous)

In mode 0, serial data is either received or transmitted eight bits at a time in a synchronous fashion with respect to a shared input/output pin and a common clock output. Reception is triggered when REN = 1 and RI = 0, while transmission is triggered by a write to SBUF. If SM2 is 0, the clock runs at $f_{CLK}/12$; otherwise, it runs at $f_{CLK}/4$, as shown in Figure 10-1 and Figure 10-2. There are no start or stop bits in this mode.

RI is set three f_{CLK} cycles after the eighth bit has been received, and TI is set three f_{CLK} cycles after the eighth bit has been transmitted. This is true when SM2 = 0, but the delays change to four f_{CLK} cycles when SM2 = 1.

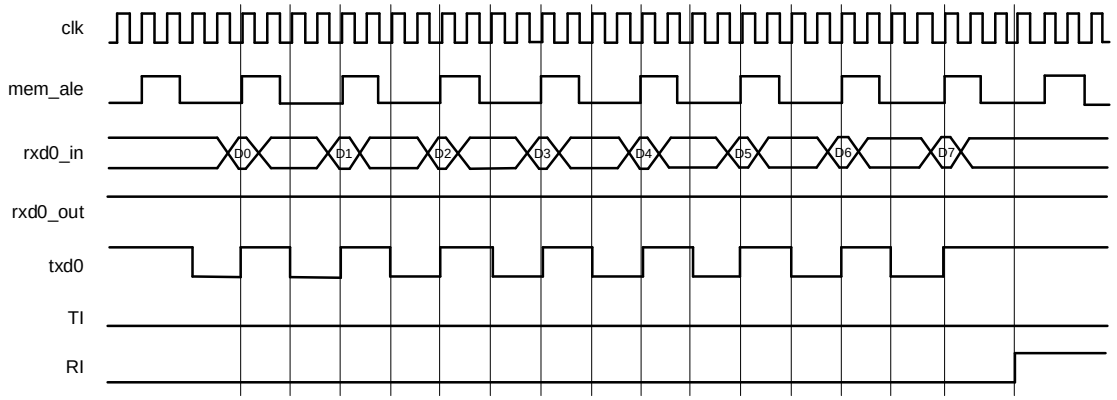


Figure 10-1. Synchronous Receive at $f_{CLK}/4$

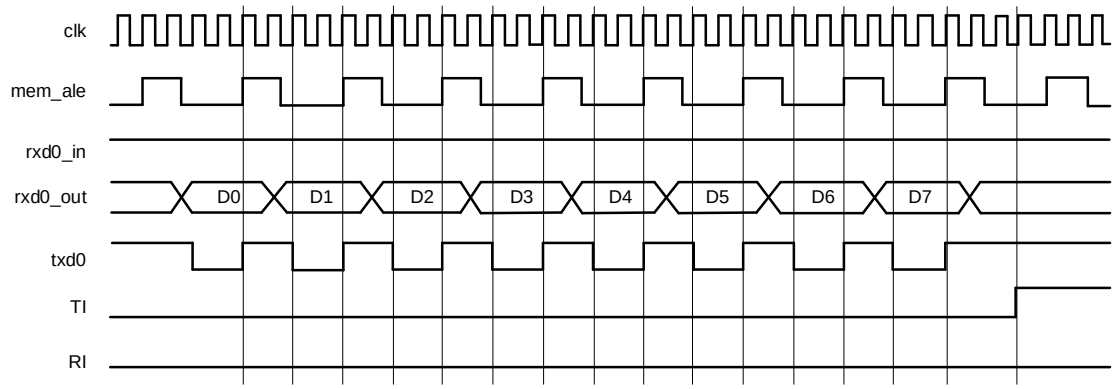


Figure 10-2. Synchronous Transmit at $f_{CLK}/4$

10.6 Mode 1 (10-Bit Asynchronous)

In mode 1, serial data is received or transmitted eight bits at a time, in an asynchronous fashion with respect to independent input and output pins. The baud rate is determined by the overflow rate of Timer/Counter 1 for serial port 0. Reception of a byte begins when REN is 1 and a start bit is recognized. This occurs after a high-to-low transition on the receive pin, followed by a low level on two of three consecutive samples made at 7/16th, 8/16th, and 9/16th of the bit time. In this way, short-lived pulses are not regarded as a valid start bit. During reception, eight bits are shifted into an input shift register, which is then loaded into the received SBUF register if:

1. RI is 0, and
2. SM2 is 1 and the stop bit is 1, or SM2 is 0 (that is, the state of the stop bit does not matter).

If these conditions are not met, the received data is lost and RI is not set. If SBUF is loaded, the state of the stop bit is copied into RB8.

Transmission is triggered by a write to SBUF and results in a 10-bit frame consisting of a low-level start bit, eight data bits, and a high-level stop bit. The start bit begins at the next rollover of the local divide-by-16 counter, and TI is set at the beginning of the stop bit.

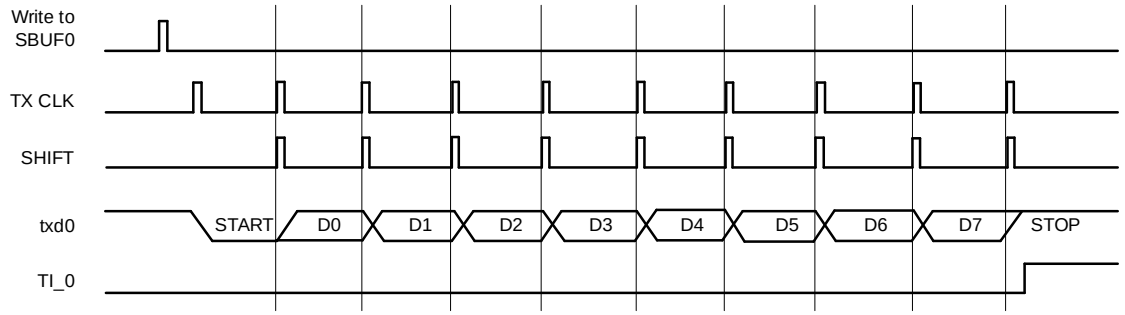


Figure 10-3. Asynchronous 10-Bit Receive Timing

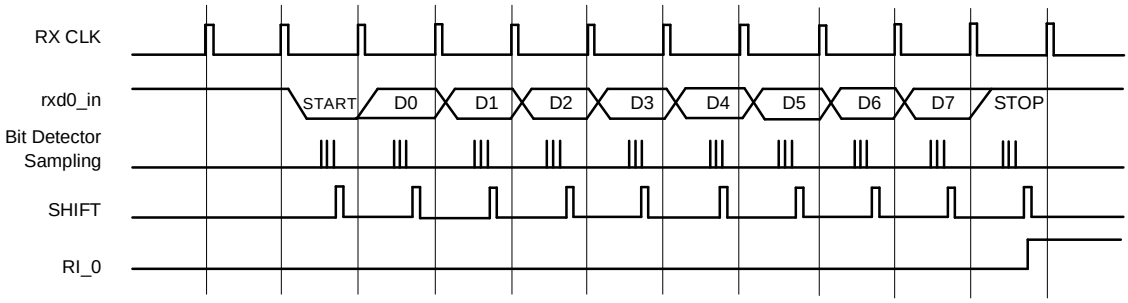


Figure 10-4. Asynchronous 10-Bit Transmit Timing

10.7 Modes 2 and 3—11-Bit Asynchronous

Modes 2 and 3 are similar in principle to mode 1, except the data field is extended to nine bits. During reception, nine bits are shifted into an input shift register, of which eight bits are then loaded into the received SBUF register if:

1. RI is 0, and
2. SM2 is 1 and the ninth bit is 1, or SM2 is 0 (that is, the state of the ninth bit does not matter).

If the conditions are not met, the received data is lost, RB8 is not loaded, and RI is not set. If SBUF is loaded, the state of the ninth data bit is copied into RB8 at SCON.2, and RI is set.

Transmission is triggered by a write to SBUF and results in an 11-bit frame consisting of a low-level start bit, eight data bits from SBUF, TB8 from SCON.3, and a high-level stop bit. The start bit begins at the next rollover of the local divide-by-16 counter, and TI is set at the beginning of the stop bit.

For mode 2, the baud rate is $f_{CLK}/64$ if SMOD is 0 (default), or $f_{CLK}/32$ if SMOD is 1. SMOD is bit 7 of PCON at 87h.

For mode 3, the baud rate is determined by the overflow rate of Timer/Counter 1 for serial port 0.

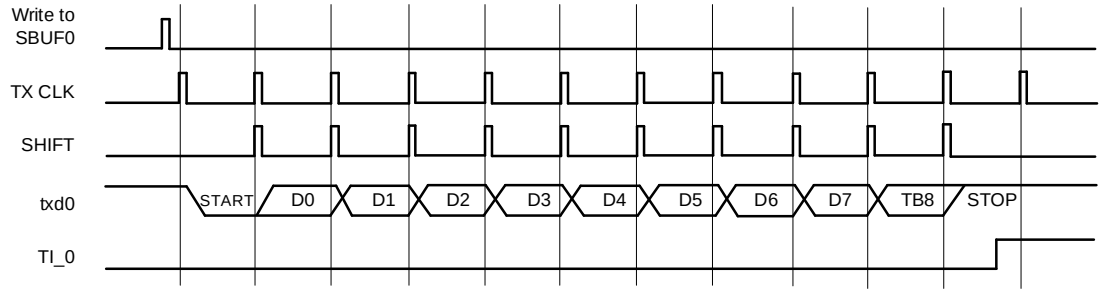


Figure 10-5. Asynchronous 11-Bit Receive

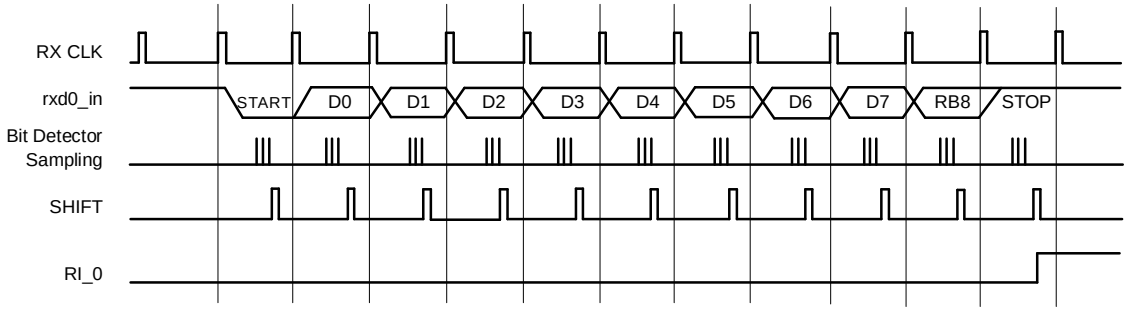


Figure 10-6. Asynchronous 11-Bit Transmit

10.8 Multiprocessor Communications

For serial ports operating in modes 2 or 3 with control bit SM2 = 1, the RI flag will only be set if the ninth bit of a received data field is 1. In this way, a byte may cause an interrupt only when the ninth data bit is 1.

In a multiprocessor system, when a master chooses to send a block of data to one of several slaves, it first transmits an address with the ninth data bit (from SCON.3) at 1. Assuming all slaves initially have SM2 set, then each will be interrupted because RI is set, but only the one matching the address will change its SM2 bit to 0. Thereafter, data bytes with the ninth data bit at 0 will be ignored by unaddressed slaves, but cause an interrupt in the addressed slave.

10.9 Example Program

Refer to Example 9-1 for an example program using Timer 0.

Interrupts

This chapter describes the MSC120x interrupts.

Topic **Page**

11.1 Description

The MSC120x extends the interrupt sources provided by the 8051 architecture in two ways. First, the MSC120x has more interrupts, which have programmable priorities of low or high. Second, the MSC120x has a new group of auxiliary interrupts of highest priority.

When an interrupt occurs, the normal execution of machine-level codes is altered by the forced insertion of an LCALL instruction to an address that depends upon the source of the interrupt. The interrupt itself is generated when the following conditions are present:

1. An asynchronous event sets an interrupt flag.
2. The corresponding interrupt enable bit is set.
3. The group enable bit is set.
4. An interrupt of equal or higher priority has not already occurred.

Normal subroutines are entered via LCALL (or ACALL) instructions, which automatically pushes the Program Counter (PC) onto the stack, and are terminated by the RET instruction, which recovers the PC from the stack.

Interrupt service routines (ISRs) are similar but must be terminated by the RETI instruction, which not only recovers the PC from the stack but also restores the interrupt level. Typically, at the start of an ISR, the Program Status Word (PSW) and Accumulator are PUSHed onto the stack and POPed off just before the RETI instruction. Once an RETI instruction has returned control to an interrupted environment and restored the interrupt level, at least one instruction will be executed before another interrupt is acknowledged.

When application code is written in C, protection of the operating context is usually managed by the compiler.

11.2 Standard and Extended Interrupts

Table 11-1 shows the standard and extended interrupts with low or high group priorities and high or low relative priorities. Global Enable = EA (IE.7), where IE is at SFR A8h. Note that in the first column of Table 11-1, the normal text describes the event, while the *italic text* describes how to clear it.

By default, all standard and extended interrupts are grouped with a low priority. However, individual interrupts may be changed to have a high priority by writing '1' to the appropriate bit within either the IP or EIP registers. All interrupts in a high priority group are serviced before those of a low priority group, and those within a group are serviced in the order of relative priority shown in Table 11-1. For example, if Timer 0 and Serial Port 0 are both low priority, then the timer will be serviced before the port. However, if bit 4 of IP at B8h is set to '1', the interrupt from Serial Port 0 will have a higher priority and be serviced before Timer 0.

Any interrupt flag associated with a low or high priority interrupt may be set in software to cause an interrupt, if enabled.

Table 11-1. Standard and Extended Interrupts

Event *Cleared by*	Flag		Enable		ISR Addr	Priority 0 = Low; 1 = High		Relative Priority
	Name	Bit(1)	Name	Bit(1)	00XXh	Name	Bit(1)	
External Interrupt 0 *Notes (2) and (3)*	IE0	TCON.1	EX0	IE.0	03	PX0	IP.0	High 1
Timer 0 Overflow *Cleared automatically*	TF0	TCON.5	ET0	IE.1	0B	PT0	IP.1	2
External Interrupt 1 *Notes (2) and (3)*	IE1	TCON.3	EX1	IE.2	13	PX1	IP.2	3
Timer 1 Overflow *Cleared automatically*	TF1	TCON.7	ET1	IE.3	1B	PT1	IP.3	4
Serial Port 0 *Clear RI_0*	RI_0	SCON0.0	ES0	IE.4	23	PS0	IP.4	5
Serial Port 0 *Clear TI_0*	TI_0	SCON0.1	ES0	IE.4	23	PS0	IP.4	5
External Interrupt 2 Positive Edge *Clear IE2*	IE2	EXIF.4	EX2	EIE.0	43	PX2	EIP.0	6
External Interrupt 3 Negative Edge *Clear IE3*	IE3	EXIF.5	EX3	EIE.1	4B	PX3	EIP.1	7
External Interrupt 4 Positive Edge *Clear IE4*	IE4	EXIF.6	EX4	EIE.2	53	PX4	EIP.2	8
External Interrupt 5 Negative Edge *Clear IE5*	IE5	EXIF.7	EX5	EIE.3	5B	PX5	EIP.3	9
Watchdog(3)(4) *Clear WDTI*	WDTI	EICON.3	EWDI	EIE.4	63	PWDI	EIP.4	10 Low

(1) Interrupt Enable (IE) is at A8h; Interrupt Priority (IP) is at B8h. Extended Interrupt Enable (EIE) is at E8h; Extended Interrupt Priority (EIP) is at F8h; External Interrupt Flag (EXIF) is at 91h.
(2) If the interrupt was edge-triggered, the flag is cleared automatically as the ISR is entered; otherwise, the flag follows the state of the pin.
(3) May also cause a wakeup from idle, if enabled.
(4) For the Watchdog Timer to generate an interrupt, bit 3 of HCR0 must be cleared; otherwise, a reset (default) will occur.

11.3 Auxiliary Interrupt Sources

Table 11-2 shows the auxiliary interrupts with highest group priority. Global Enable = EAI (EICON.5), where EICON is at D8h, Auxiliary Interrupt Enable (AIE) is at A6h. All these interrupts set the AI flag (EICON.4), which must be cleared in software in addition to the individual interrupt flags. Setting AI in software generates an auxiliary interrupt, if enabled. Note that in the first column of Table 11-2, the normal text describes the event, while the *italic text* describes how to clear it.

When multiple auxiliary interrupts are enabled, the ISR at 0033h can read the Pending Auxiliary Interrupt (PAI) at A5h to identify the interrupt of greatest relative priority. If PAI returns 0, there is no pending auxiliary interrupt.

Pending (active and enabled) interrupts can be identified by testing the corresponding bits in AISTAT at A7h. This allows the programmer to service auxiliary interrupts with arbitrary and even dynamic relative priorities.

Unlike flags in the low and high priority groups, no interrupt flag in the highest priority group may be set in software to cause an interrupt. However, AI (EICON.4) can be set to trigger an auxiliary interrupt, but a user-specific mechanism must be used to recognize this as a separate source.

For a particular interrupt flag to be set, the corresponding subsystem must be powered up as determined by bits in PDCON at F1h. For example, PDCON.3 must be 0 for an ADC interrupt to occur.

Table 11-2. Auxiliary Interrupts with Highest Group Priority

Event ***Cleared By***	Flag		Enable		ISR Addr	Priority	Relative Priority and Value from PAI
	Name	**Bit**	**Name**	**Bit**	**00XXh**		
AV_{DD} Low Voltage *Voltage is restored*	ALVDIP	AIPOL.1	EALV	AIE.1	33	Highest	2
I^2C Start/Stop	I2CIP	AIPOL.3	ESPIT	AIE.3	33	Highest	3
SPI (or I^2C) Count	CNTIP	AIPOL.2	ESPIR	AIE.2	33	Highest	4
Milliseconds Timer *Read MSINT at FAh*	MSECIP	AIPOL.4	EMSEC	AIE.4	33	Highest	5
ADC Conversion *Read ADRESL at D9h*	ADCIP	AIPOL.5	EADC	AIE.5	33	Highest	6
Summation Register *Read SUMR0 at E2h*	SUMIP	AIPOL.6	ESUM	AIE.6	33	Highest	7
Seconds timer *Read SECINT at F9h*	SECIP	AIPOL.7	ESEC	AIE.7	33	Highest	8

11.4 Multiple Interrupts

When there is just one interrupt, the ISR is most often relatively easy to write and the model of the timing is simple. However, with multiple sources of different priorities, the complexity, in terms of timing and exact behavior, grows quickly. Since there are three groups of priority (classed as low, high, and highest), it is possible to have three nested levels of interrupts. For example, the main program may be interrupted by an event of low priority, but the ISR may be interrupted by an event of high priority, which in turn could be interrupted by an event of highest priority.

It is essential that there is no unintentional interaction between different interrupts, and that the operating environment or context is restored prior to termination of an ISR. For all but the simplest of ISRs, it is necessary to save and restore the primary context (PSW and Accumulator) to and from the stack. Similarly, working registers R0 to R7 may need to be PUSHed and POPed, but this is time consuming and can be avoided by register bank switching.

Once the primary context has been PUSHed onto the stack, the value of bits 4 and 3 in the PSW may be changed to select a different bank of 8-bit working registers. In this way, the values in the previous bank are not changed by instructions that reference registers relative to the new bank. It is practical to allocate bank 0 to the main program, bank 1 to low interrupts, bank 2 to high, and bank 3 to highest. Since working registers are also mapped to memory locations, it is possible to modify (and corrupt) any register by writing to an explicit location. For example, R4 of bank 2 is at data address 14h. Care may be needed in this regard when using multiple interrupts.

11.5 Example of Multiple and Nested Interrupts

The example shows ISRs for interrupts of low, high, and highest priority. Based on a clock of 11.0592MHz, the program does the following:

1. Toggles signal sync3 (P1.3) as frequently as possible, subject to servicing interrupts. Assuming the main program is implemented as the instruction CPL P1.3, sync3 will toggle every 0.723µs.
2. Transmits a digit between 0 and 3 at 9600 baud on serial port 0 every 20ms, as triggered by the milliseconds system timer.
3. Uses the interrupt from Timer 0 to toggle sync0 (P1.0) every 278ms.
4. Receives characters from serial port 0 via an interrupt and toggles sync1 (P1.1).
5. Shows the level of interrupt nesting by the value of the digit transmitted.

If the time to execute the ISR associated with Timer 0 is short, then most interrupts will occur with respect to the main program. However, every application with multiple interrupts should cater to the least likely combination of events. In this case, it is possible that the main program is interrupted by an overflow from Timer 0, which is then interrupted due to a character received on serial port 0, which itself is interrupted by the milliseconds timer. The variable called *level* will then be 3 and cause a '3' to be transmitted because the MsecInt ISR forces a transmit interrupt for serial port 0 by setting TI_0.

The characters most often transmitted are '1' and `2'. However, a '3' may be seen occasionally, depending upon the relative timing of the received character with respect to the other interrupts. The probability is affected considerably by the rate at which characters are received, the value of LIMIT, and the efficiency of the code produced by the compiler.

Example 11-1. Multiple and Nested Interrupts

```
// Example: Multiple and Nested Interrupts
// File Interrupts_4.c

// MSC1200 EVM jumper setting as factory default

#include <Reg1200.h>
#include <stdio.h>

#define xtal 22118400
#define BAUD 9600
#define LIMIT 150
#define RATE 100

sbit RedLed = P3^4;           // RED LED on EVM
sbit YellowLed = P3^5;        // Yellow LED on EVM
sbit sync0 = P1^0;            // Port 1 bit 0
sbit sync1 = P1^1;            // Port 1 bit 1
sbit sync2 = P1^2;            // Port 1 bit 2
sbit sync3 = P1^3;            // Port 1 bit 3

data unsigned int j; char level=0, send;

void process(void)
{   data char i;              // simulate additional execution time
    for(i=0;i<LIMIT;i++);
}

void MsecInt(void) interrupt 6 using 3
{      data char temp;
    level++;
    temp=MSINT;               // read MSINT to remove interrupt
    AI=0;                     // remove auxiliary flag
    send='0'+level;           // characters '1' to '3'
    TI_0=1;                   // trigger serial output
    level--;
}

void Serial0Int(void) interrupt 4 using 2
{      level++;
    RedLed=YellowLed;         // monitor
    if(RI_0) {
        sync1=!sync1;         // monitor
        process();            // simulate additional execution time
        RI_0=0;               // remove Rx flag
    }
    if(TI_0) {                // test transmit interrupt flag
        TI_0=0;               // remove Tx flag
        if(send) {SBUF0=send; send=0;}
    }
    level--;
}

void Timer0Int(void) interrupt 1 using 1
{   level++;
    sync0=!sync0;             // monitor
    YellowLed=0;
    process();                // simulate additional execution time
    YellowLed=1;
    level--;
}

void main(void)
{   PDCON=0x6D;               // System Timer enabled
```

Example 11-1. Multiple and Nested Interrupts (continued)

```
    SCON0=0x50;               // Serial 0 enable; RI_0 cleared
    CKCON=0x00;               // Timers 0 and 1 at fclk/12
    PCON =0x30;               // SMOD = 0 => normal Baud rate eqution
    TMOD =0x22;               // Timers 1 and 0 Auto reload
    TCON =0x50;               // TR1 and TR0 are 1
    TH1 =256-xtal/32/12/BAUD; // Timer 1 reload value
    TH0 =0;                   // Overflows every 256 * 12 * tclk
    MSEC=xtal/1000 - 1;       // 1 ms reference
    MSINT=20 - 1;             // 20 ms interrupt interval
    IP =0x90;                 // Priorities: Serial0 as'high',Timer0,Timer1 as natural priority
    IE =0x92;                 // EA, ES0 and ET0 enabled
    AIE =0x10;                // EMSEC enabled
    EICON=0x60;               // Auxiliary interrupts enabled
    while(1){
        sync3=!sync3;         // foreground program
    }
}
```

Interrupts from Timers 0 and 1 are both in the low priority group, and are therefore mutually exclusive and share register bank 1. The priority of Serial Port 0 is raised to high by writing a '1' to bit 4 of register IP, and therefore uses a different register bank. Similarly, since the milliseconds interrupt is in the highest group, the ISR is allocated its own register bank.

If an interrupt from Timer 0 is pending at the same moment, Timer 0 will be serviced first because it has a higher relative priority within the low group.

In this particular example, individual ISRs may not use registers, depending upon the efficiency and optimization level of the compiler. However, the allocation of register banks ensures mutually exclusive contexts and is the usual practice.

The interrupt number used in C is given by (ISR Address - 3) divided by 8.

11.6 Example of Wake Up from Idle

In order to reduce operating power, the MSC120x can be placed into an idle state by writing '1' to bit 0 of PCON at 87h. In this state, the CPU, Timers 0, 1, and 2, and the USARTs are not clocked, although other peripherals remain active (unless previously powered-down via bits in PDCON at F1h). Once in the idle state, normal operation is resumed by an enabled auxiliary interrupt or an enabled wake-up condition.

Three wake-up conditions are enabled by bits in the Enable Wake Up (EWU) SFR at C6h, as shown in Table 11-3.

Table 11-3. EWU—Enable Wake Up

EWU		SFR C6h	Reset Value = 00h
Bit #	**Name**	**Action or Interpretation**	
7-3		Undefined	
2	EWUWDT	Enable wake up on watchdog timer 0: Disable wake up on watchdog timer interrupt 1: Enable wake up on watchdog timer interrupt	
1	EWUEX1	Enable wake up on external 1 0: Disable wake up on external interrupt source 1 1: Enable wake up on external interrupt source 1	
0	EWUEX0	Enable wake up on external 0 0: Disable wake up on external interrupt source 0 1: Enable wake up on external interrupt source 0	

It is possible to synchronize the activity of a program to an external input by repeatedly reading its level; however, that requires more power than configuring an interrupt and placing the MSC120x into an idle state.

Example 11-2. Wake Up From Idle

```
// File Idle.c
// MSC120x EVM
#include <Reg1200.h>
sbit sync0 = P1^0;                      // Port 1 bit 0
sbit sync1 = P1^1;                      // Port 1 bit 1
sbit sync2 = P1^2;                      // Port 1 bit 2

data unsigned char j;

void INT0Int(void) interrupt 0 using 1  // No action
{
sync1=!sync1;                           // monitor for interest
}

void main(void)
{   IE   =0x81;                         // EA EX0
    EWU  =0x01;                         // Enable Wakeup
    IT0  =1;                            // Falling edge on INT0
    while(1){
        while(!INT0);                   // wait for INT0=1
        sync2=0;
        PCON|=1;                        // IDLE
        sync2=1;
        for(j=0;j<30;j++) sync0=!sync0;
    }
}
```

If an external interrupt is configured for falling-edge detection, the IDLE bit must be set when the input is high. Similarly, for rising-edge detection, IDLE must be set when the input is low.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265