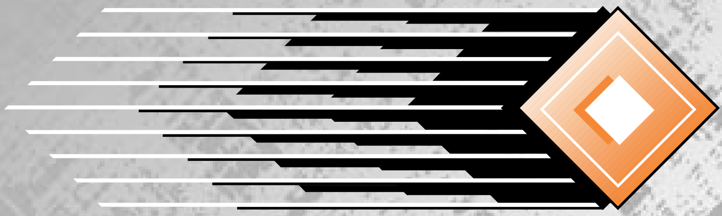
8XC251SB
Embedded Microcontroller
User's Manual



8XC251SB Embedded Microcontroller User's Manual

February 1995

Information in this document is provided solely to enable use of Intel products. Intel assumes no liability whatsoever, including infringement of any patent or copyright, for sale and use of Intel products except as provided in Intel's Terms and Conditions of Sale for such products.

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local Intel sales office or your distributor to obtain the latest specifications before placing your product order.

MDS is an ordering code only and is not used as a product name or trademark of Intel Corporation.

Intel Corporation and Intel's FASTPATH are not affiliated with Kinetics, a division of Excelan, Inc. or its FASTPATH trademark or products.

*Other brands and names are the property of their respective owners.

Additional copies of this document or other Intel literature may be obtained from:

Intel Corporation
Literature Sales
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

© INTEL CORPORATION, 2/26/96

CONTENTS

CHAPTER 1
GUIDE TO THIS MANUAL

1.1 MANUAL CONTENTS 1-1
1.2 NOTATIONAL CONVENTIONS AND TERMINOLOGY 1-3
1.3 RELATED DOCUMENTS 1-5
  1.3.1 Data Sheet 1-6
  1.3.2 Application Notes 1-6
1.4 CUSTOMER SERVICE 1-7
  1.4.1 How to Use Intel's FaxBack Service 1-7
  1.4.2 How to Use Intel's Application BBS 1-8
  1.4.3 How to Find the Latest *Ap*BUILDER Files and Hypertext Manuals and Data Sheets on the BBS 1-9

CHAPTER 2
ARCHITECTURAL OVERVIEW

2.1 8XC251SB CORE 2-4
  2.1.1 CPU 2-4
  2.1.2 Clock and Reset Unit 2-5
  2.1.3 Interrupt Handler 2-6
  2.1.4 On-chip Code Memory 2-6
  2.1.5 On-chip RAM 2-7
2.2 ON-CHIP PERIPHERALS 2-7
  2.2.1 Timer/Counters and Watchdog Timer 2-7
  2.2.2 Programmable Counter Array (PCA) 2-7
  2.2.3 Serial I/O Port 2-8

CHAPTER 3
ADDRESS SPACES

3.1 ADDRESS SPACES FOR MCS® 251 MICROCONTROLLERS 3-1
  3.1.1 Compatibility with the MCS® 51 Architecture 3-2
3.2 THE 8XC251SB MEMORY SPACE 3-5
  3.2.1 On-chip General-purpose Data RAM 3-6
  3.2.2 On-chip Code Memory (87C251SB/83C251SB) 3-6
    3.2.2.1 Accessing On-chip Code Memory in Region 00: 3-6
  3.2.3 External Memory 3-8
3.3 THE 8XC251SB REGISTER FILE 3-8
  3.3.1 Byte, Word, and Dword Registers 3-8
  3.3.2 Dedicated Registers 3-10

3.3.2.1 Accumulator and B Register 3-10
3.3.2.2 Extended Data Pointer, DPX 3-10
3.3.2.3 Extended Stack Pointer, SPX 3-11
3.4 SPECIAL FUNCTION REGISTERS (SFRS) 3-12

CHAPTER 4
PROGRAMMING

4.1 BINARY MODE AND SOURCE MODE CONFIGURATIONS 4-1
4.1.1 Selecting Binary Mode or Source Mode 4-2
4.2 PROGRAMMING FEATURES OF THE MCS® 251 ARCHITECTURE 4-4
4.2.1 Data Types 4-4
4.2.2 Register Notation 4-4
4.2.3 Address Notation 4-5
4.2.4 Addressing Modes 4-5
4.3 DATA INSTRUCTIONS 4-6
4.3.1 Data Addressing Modes 4-6
4.3.1.1 Register Addressing 4-8
4.3.1.2 Immediate 4-8
4.3.1.3 Direct 4-8
4.3.1.4 Indirect 4-9
4.3.1.5 Displacement 4-9
4.3.2 Arithmetic Instructions 4-10
4.3.3 Logical Instructions 4-11
4.3.4 Data Transfer Instructions 4-11
4.4 BIT INSTRUCTIONS 4-12
4.4.1 Bit Addressing 4-12
4.5 CONTROL INSTRUCTIONS 4-14
4.5.1 Addressing Modes for Control Instructions 4-14
4.5.2 Conditional Jumps 4-15
4.5.3 Unconditional Jumps 4-16
4.5.4 Calls and Returns 4-16
4.6 PROGRAM STATUS WORDS 4-17

CHAPTER 5
INTERRUPT SYSTEM

5.1 OVERVIEW 5-1
5.2 8XC251SB INTERRUPT SOURCES.................... 5-3
5.2.1 External Interrupts 5-3
5.2.2 Timer Interrupts 5-4
5.3 PROGRAMMABLE COUNTER ARRAY (PCA) INTERRUPT.................... 5-5
5.4 SERIAL PORT INTERRUPT.................... 5-5
5.5 INTERRUPT ENABLE 5-5
5.6 INTERRUPT PRIORITIES.................... 5-6

5.7 INTERRUPT PROCESSING 5-9
5.7.1 Minimum Fixed Interrupt Time 5-10
5.7.2 Variable Interrupt Parameters 5-10
5.7.2.1 Response Time Variables 5-10
5.7.2.2 Computation of Worst-case Latency With Variables 5-12
5.7.2.3 Latency Calculations 5-13
5.7.2.4 Blocking Conditions 5-14
5.7.2.5 Interrupt Vector Cycle 5-14
5.7.3 ISRs in Process 5-15

CHAPTER 6
INPUT/OUTPUT PORTS

6.1 INPUT/OUTPUT PORT OVERVIEW 6-1
6.2 I/O CONFIGURATIONS 6-2
6.3 PORT 1 AND PORT 3 6-2
6.4 PORT 0 AND PORT 2 6-2
6.5 READ-MODIFY-WRITE INSTRUCTIONS 6-5
6.6 QUASI-BIDIRECTIONAL PORT OPERATION 6-5
6.7 PORT LOADING 6-7
6.8 EXTERNAL MEMORY ACCESS 6-7

CHAPTER 7
TIMER/COUNTERS AND WATCHDOG TIMER

7.1 TIMER/COUNTER OVERVIEW 7-1
7.2 TIMER/COUNTER OPERATION 7-1
7.3 TIMER 0 7-4
7.3.1 Mode 0 (13-bit Timer) 7-4
7.3.2 Mode 1 (16-bit Timer) 7-5
7.3.3 Mode 2 (8-bit Timer With Auto-reload) 7-5
7.3.4 Mode 3 (Two 8-bit Timers) 7-5
7.4 TIMER 1 7-6
7.4.1 Mode 0 (13-bit Timer) 7-9
7.4.2 Mode 1 (16-bit Timer) 7-9
7.4.3 Mode 2 (8-bit Timer with Auto-reload) 7-9
7.4.4 Mode 3 (Halt) 7-9
7.5 TIMER 0/1 APPLICATIONS 7-9
7.5.1 Auto-load Setup Example 7-9
7.5.2 Pulse Width Measurements 7-10
7.6 TIMER 2 7-10
7.6.1 Capture Mode 7-11
7.6.2 Auto-reload Mode 7-12
7.6.2.1 Up Counter Operation 7-12
7.6.2.2 Up/Down Counter Operation 7-13

7.6.3 Baud Rate Generator Mode 7-14
7.6.4 Clock-out Mode 7-14
7.7 WATCHDOG TIMER 7-16
7.7.1 Description 7-16
7.7.2 Using the WDT 7-18
7.7.3 WDT During Idle Mode 7-18
7.7.4 WDT During PowerDown 7-18

CHAPTER 8
PROGRAMMABLE COUNTER ARRAY

8.1 PCA DESCRIPTION.................... 8-1
8.2 PCA TIMER/COUNTER.................... 8-2
8.3 PCA COMPARE/CAPTURE MODULES 8-5
8.3.1 16-bit Capture Mode 8-5
8.3.2 Compare Modes 8-7
8.3.3 16-bit Software Timer Mode 8-7
8.3.4 High-speed Output Mode 8-8
8.3.5 PCA Watchdog Timer Mode 8-9
8.3.6 Pulse Width Modulation Mode 8-11

CHAPTER 9
SERIAL I/O PORT

9.1 OVERVIEW 9-1
9.2 MODES OF OPERATION.................... 9-4
9.2.1 Synchronous Mode (Mode 0) 9-4
9.2.1.1 Transmission (Mode 0) 9-4
9.2.1.2 Reception (Mode 0) 9-5
9.2.2 Asynchronous Modes (Modes 1, 2, and 3) 9-6
9.2.2.1 Transmission (Modes 1, 2, 3) 9-6
9.2.2.2 Reception (Modes 1, 2, 3) 9-6
9.3 FRAMING BIT ERROR DETECTION (MODES 1, 2, AND 3).................... 9-7
9.4 MULTIPROCESSOR COMMUNICATION (MODES 2 AND 3).................... 9-7
9.5 AUTOMATIC ADDRESS RECOGNITION.................... 9-7
9.5.1 Given Address 9-8
9.5.2 Broadcast Address 9-9
9.5.3 Reset Addresses 9-10
9.6 BAUD RATES.................... 9-10
9.6.1 Baud Rate for Mode 0 9-10
9.6.2 Baud Rates for Mode 2 9-10
9.6.3 Baud Rates for Modes 1 and 3 9-10
9.6.3.1 Timer 1 Generated Baud Rates (Modes 1 and 3) 9-11
9.6.3.2 Selecting Timer 1 as the Baud Rate Generator 9-11
9.6.3.3 Timer 2 Generated Baud Rates (Modes 1 and 3) 9-12
9.6.3.4 Selecting Timer 2 as the Baud Rate Generator 9-12

CHAPTER 10
MINIMUM HARDWARE SETUP

10.1 MINIMUM HARDWARE SETUP 10-1
10.2 ELECTRICAL ENVIRONMENT 10-2
  10.2.1 Power and Ground Pins 10-2
  10.2.2 Unused Pins 10-2
  10.2.3 Noise Considerations 10-2
10.3 CLOCK SOURCES 10-3
  10.3.1 On-chip Oscillator (Crystal) 10-3
  10.3.2 On-chip Oscillator (Ceramic Resonator) 10-4
  10.3.3 External Clock 10-4
10.4 RESET 10-5
  10.4.1 Externally Initiated Resets 10-6
  10.4.2 WDT Initiated Resets 10-6
  10.4.3 Reset Operation 10-6
  10.4.4 Power-on Reset 10-7

CHAPTER 11
SPECIAL OPERATING MODES

11.1 GENERAL 11-1
11.2 POWER CONTROL REGISTER 11-1
  11.2.1 Serial I/O Control Bits 11-1
  11.2.2 Power Off Flag 11-1
11.3 IDLE MODE 11-4
  11.3.1 Entering Idle Mode 11-4
  11.3.2 Exiting Idle Mode 11-5
11.4 POWERDOWN MODE 11-5
  11.4.1 Entering Powerdown Mode 11-6
  11.4.2 Exiting Powerdown Mode 11-6
11.5 ON-CIRCUIT EMULATION (ONCE) MODE 11-7
  11.5.1 Entering ONCE Mode 11-7
  11.5.2 Exiting ONCE Mode 11-7

CHAPTER 12
EXTERNAL MEMORY INTERFACE

12.1 EXTERNAL MEMORY INTERFACE SIGNALS 12-1
12.2 CONFIGURING THE EXTERNAL MEMORY INTERFACE 12-2
  12.2.1 Page Mode and Nonpage Mode (PAGE Bit) 12-3
  12.2.2 RD#, PSEN#, and the Number of External Address Pins (Bits RD1:0) 12-3
    12.2.2.1 Sixteen External Address Bits and a Single Read Signal (RD1 = 1, RD0 = 0) 12-4
    12.2.2.2 Seventeen External Address Bits and a Single Read Signal (RD1 = 0, RD0 = 1) 12-4

12.2.2.3 Sixteen External Address Bits and Two Read Signals (RD1 = 1, RD0 = 1) 12-5
12.2.3 Wait States (WSA, WSB, XALE) 12-6
12.2.4 Mapping On-chip Code Memory to Data Memory (87C251SB/83C251SB) 12-7
12.3 EXTERNAL BUS CYCLES 12-7
12.3.1 Inactive External Bus 12-7
12.3.2 Bus Cycle Definitions 12-8
12.3.3 Nonpage Mode Bus Cycles 12-8
12.3.4 Page Mode Bus Cycles 12-10
12.4 WAIT STATES 12-13
12.4.1 Extending PSEN#/RD#/WR# 12-13
12.4.2 Extending ALE 12-14
12.5 PORT 0 AND PORT 2 STATUS 12-15
12.5.1 Port 0 and Port 2 Pin Status in Nonpage Mode 12-15
12.5.2 Port 0 and Port 2 Pin Status in Page Mode 12-16
12.6 EXTERNAL MEMORY DESIGN EXAMPLES.... 12-16
12.6.1 Nonpage Mode, 64 Kbytes External EPROM, 64 Kbytes External RAM 12-16
12.6.1.1 An Application Requiring Fast Access to the Stack 12-16
12.6.1.2 An Application Requiring Fast Access to Data 12-17
12.6.2 Nonpage Mode, 128 Kbytes External RAM 12-19
12.6.3 Page Mode, 128 Kbytes External Flash 12-21
12.6.4 Page Mode, 64 Kbytes External EPROM, 64 Kbytes External RAM 12-21
12.6.5 Page Mode, 64 Kbytes External Flash, 32 Kbytes External RAM 12-22
12.7 EXTERNAL BUS AC TIMING SPECIFICATIONS 12-24
12.7.1 Explanation of AC Symbols 12-28
12.7.2 AC Timing Definitions 12-28

CHAPTER 13
PROGRAMMING AND VERIFYING NONVOLATILE MEMORY

13.1 GENERAL.... 13-1
13.2 PROGRAMMING AND VERIFYING MODES.... 13-2
13.3 GENERAL SETUP.... 13-3
13.4 OTPROM PROGRAMMING ALGORITHM.... 13-4
13.5 VERIFY ALGORITHM.... 13-5
13.6 PROGRAMMABLE FUNCTIONS 13-5
13.6.1 On-chip Code Memory 13-5
13.6.2 Configuration Bytes 13-6
13.6.3 Lock Bit System 13-9
13.6.4 Encryption Array 13-10
13.6.5 Signature Bytes 13-10
13.7 VERIFYING THE 83C251SB (ROM) 13-10
13.8 VERIFYING THE 80C251SB (ROMLESS) 13-11

APPENDIX A
INSTRUCTION SET REFERENCE

A.1 NOTATION FOR INSTRUCTION OPERANDS A-2
A.2 OPCODE MAP AND SUPPORTING TABLES A-4
A.3 INSTRUCTION SET SUMMARY A-11
A.3.1 Execution Times for Instructions that Access the Port SFRs A-11
A.3.2 Instruction Summaries A-14
A.4 INSTRUCTION DESCRIPTIONS A-26

APPENDIX B
SIGNAL DESCRIPTIONS

APPENDIX C
REGISTERS

GLOSSARY

INDEX

FIGURES

Figure **Page**
2-1 Functional Block Diagram of the 8XC251SB.....2-2
2-2 The CPU.....2-5
2-3 8XC251SB Timing.....2-6
3-1 Address Spaces for MCS® 251 Microcontrollers.....3-1
3-2 Address Spaces for the MCS® 51 Architecture.....3-3
3-3 Address Space Mappings MCS® 51 Architecture to MCS® 251 Architecture.....3-4
3-4 8XC251SB Memory Space.....3-7
3-5 The Register File.....3-9
3-6 Dedicated Registers in the Register File and their Corresponding SFRs.....3-11
4-1 Binary Mode Opcode Map.....4-3
4-2 Source Mode Opcode Map.....4-3
4-3 Program Status Word Register.....4-19
4-4 Program Status Word 1 Register.....4-20
5-1 Interrupt Control System.....5-2
5-2 Interrupt Enable Register.....5-6
5-3 Interrupt Priority High Register.....5-8
5-4 Interrupt Priority Low Register.....5-8
5-5 The Interrupt Process.....5-9
5-6 Response Time Example #1.....5-11
5-7 Response Time Example #2.....5-12
6-1 Port 1 and Port 3 Structure.....6-3
6-2 Port 0 Structure.....6-3
6-3 Port 2 Structure.....6-4
6-4 Internal Pullup Configurations.....6-6
7-1 Basic Logic of the Timer/Counters.....7-2
7-2 Timer 0/1 in Mode 0 and Mode 1.....7-4
7-3 Timer 0/1 in Mode 2, Auto-Reload.....7-5
7-4 Timer 0 in Mode 3, Two 8-bit Timers.....7-6
7-5 TMOD: Timer/Counter Mode Control Register.....7-7
7-6 TCON: Timer/Counter Control Register.....7-8
7-7 Timer 2: Capture Mode.....7-11
7-8 Timer 2: Auto Reload Mode (DCEN = 0).....7-12
7-9 Timer 2: Auto Reload Mode (DCEN = 1).....7-13
7-10 Timer 2: Clock Out Mode.....7-15
7-11 T2MOD: Timer 2 Mode Control Register.....7-16
7-12 T2CON: Timer 2 Control Register.....7-17
8-1 Programmable Counter Array.....8-3
8-2 PCA 16-bit Capture Mode.....8-6
8-3 PCA Software Timer and High-speed Output Modes.....8-8
8-4 PCA Watchdog Timer Mode.....8-10
8-5 PCA 8-bit PWM Mode.....8-11
8-6 PWM Variable Duty Cycle.....8-12
8-7 CMOD: PCA Timer/Counter Mode Register.....8-13
8-8 CCON: PCA Timer/Counter Control Register.....8-14

FIGURES

Figure **Page**
8-9 CCAPMx: PCA Compare/Capture Module Mode Registers......................................8-16
9-1 Serial Port Block Diagram 9-2
9-2 Serial Port Special Function Register..9-3
9-3 Mode 0 Timing...9-5
9-4 Data Frame (Modes 1, 2, and 3) 9-6
9-5 Timer 2 in Baud Rate Generator Mode 9-13
10-1 Minimum Setup 10-1
10-2 CHMOS On-chip Oscillator..10-3
10-3 External Clock Connection 10-4
10-4 External Clock Drive Waveforms..10-5
10-5 Reset Timing Sequence 10-8
11-1 Power Control (PCON) Register...11-2
11-2 Idle and Powerdown Clock Control 11-3
12-1 Internal and External Memory Spaces for RD1 = 1, RD0 = 0...................................12-4
12-2 Internal and External Memory Spaces for RD1 = 0, RD0 = 1...................................12-5
12-3 Internal and External Memory Spaces for RD1 = 1, RD0 = 1...................................12-6
12-4 External Code Fetch or Data Read Bus Cycle (Nonpage Mode) 12-9
12-5 External Write Bus Cycle (Nonpage Mode)...12-9
12-6 Bus Structure in Nonpage Mode and Page Mode 12-10
12-7 External Code Fetch Bus Cycle (Page Mode)...12-11
12-8 External Data Read Bus Cycle (Page Mode) 12-12
12-9 External Write Bus Cycle (Page Mode)...12-12
12-10 External Code Fetch or Data Read Bus Cycle with One PSEN#/RD#
Wait State (Nonpage Mode)...12-13
12-11 External Write Bus Cycle with One WR# Wait State (Nonpage Mode)....................12-14
12-12 External Code Fetch or Data Read Bus Cycle with One ALE Wait State
(Nonpage Mode) 12-14
12-13 80C251SB in Nonpage Mode with External EPROM and RAM 12-17
12-14 The Memory Space for the Systems of Figure 12-13 and Figure 12-18 12-18
12-15 87C251SB/83C251SB in Nonpage Mode with 128 Kbytes of External RAM...........12-19
12-16 The Memory Space for the System of Figure 12-15..12-20
12-17 80C251SB in Page Mode with External Flash...12-21
12-18 80C251SB in Page Mode with External EPROM and RAM 12-22
12-19 80C251SB in Page Mode with External Flash and RAM...12-23
12-20 The Memory Space for the System of Figure 12-19..12-24
12-21 External Bus Cycles for Data/Instruction Read and Data Write in
Nonpage Mode...12-25
12-22 External Bus Cycles for Data Read and Data Write in Page Mode.........................12-26
12-23 External Bus Cycles for Instruction Read in Page Mode...12-27
13-1 Setup for Programming and Verifying 13-3
13-2 OTPROM Programming Waveforms..13-4
13-3 Configuration Byte 0..13-7
13-4 Configuration Byte 1..13-8
13-5 OTPROM Timing...13-11

TABLES

Table Page

2-1 Summary of 8XC251SB Features 2-4
3-1 Address Mappings 3-4
3-2 Register Bank Selection 3-8
3-3 Dedicated Registers in the Register File and their Corresponding SFRs....................3-12
3-4 8XC251SB SFR Map and Reset Values 3-13
3-5 Core SFRs....................3-14
3-6 I/O Port SFRs 3-14
3-7 Serial I/O SFRs 3-15
3-8 Timer/Counter and Watchdog Timer SFRs 3-15
3-9 Programmable Counter Array (PCA) SFRs....................3-15
4-1 Examples of Opcodes in Binary and Source Modes 4-2
4-2 Data Types 4-4
4-3 Notation for Byte Registers, Word Registers, and Dword Registers 4-5
4-4 Addressing Modes for Data Instructions in the MCS® 51 Architecture 4-6
4-5 Addressing Modes for Data Instructions in the MCS® 251 Architecture 4-7
4-6 Bit-addressable Locations 4-13
4-7 Addressing Two Sample Bits....................4-13
4-8 Addressing Modes for Bit Instructions....................4-14
4-9 Addressing Modes for Control Instructions....................4-15
4-10 Compare-conditional Jump Instructions 4-16
4-11 The Effects of Instructions on the PSW and PSW1 Flags....................4-18
5-1 Interrupt System Pin Signals 5-1
5-2 Interrupt System Special Function Registers 5-3
5-3 Interrupt Control Matrix....................5-4
5-4 Level of Priority....................5-7
5-5 Interrupt Priority Within Level 5-7
5-6 Interrupt Latency Variables 5-13
5-7 Actual vs. Predicted Latency Calculations....................5-13
6-1 Input/Output Port Pin Descriptions 6-1
6-2 Instructions for External Data Moves....................6-8
7-1 Timer/Counter and Watchdog Timer SFRs 7-2
7-2 External Signals 7-3
7-3 Timer 2 Modes of Operation....................7-15
8-1 PCA Special Function Registers (SFRs) 8-4
8-2 External Signals 8-4
8-3 PCA Module Modes 8-15
9-1 Serial Port Signals....................9-1
9-2 Serial Port Special Function Registers....................9-2
9-3 Summary of Baud Rates 9-10
9-4 Timer 1 Generated Baud Rates for Serial I/O Modes 1 and 3....................9-12
9-5 Selecting the Baud Rate Generator(s) 9-13
9-6 Timer 2 Generated Baud Rates 9-14
11-1 Pin Conditions in Various Modes....................11-3
12-1 External Memory Interface Signals....................12-1

TABLES

Table **Page**

12-2 Configuration Bits RD1:0 12-3
12-3 Wait State Selection 12-6
12-4 Bus Cycle Definitions (No Wait States) 12-8
12-5 Port 0 and Port 2 Pin Status In Normal Operating Mode 12-15
12-6 AC Timing Symbol Definitions 12-28
12-7 AC Timing Definitions for Specifications on the 8XC251SB 12-29
12-8 AC Timing Definitions for Specifications on the Memory System 12-30
13-1 Programming and Verifying Modes 13-2
13-2 Configuration Byte Values for 80C251SB and 80C251SB-16 13-9
13-3 Lock Bit Function 13-9
13-4 Contents of the Signature Bytes 13-10
13-5 OTPROM Timing Definitions 13-12
A-1 Notation for Register Operands A-2
A-2 Notation for Direct Addresses A-3
A-3 Notation for Immediate Addressing A-3
A-4 Notation for Bit Addressing A-3
A-5 Notation for Destinations in Control Instructions A-3
A-6 Instructions for MCS® 51 Microcontrollers A-4
A-7 New Instructions for the MCS® 251 Architecture A-5
A-8 Data Instructions A-6
A-9 High Nibble, Byte 0 of Data Instructions A-6
A-10 Bit Instructions A-7
A-11 Byte 1 (High Nibble) for Bit Instructions A-7
A-12 PUSH/POP Instructions A-8
A-13 Control Instructions A-8
A-14 Displacement/Extended MOVs A-9
A-15 INC/DEC A-10
A-16 Encoding for INC/DEC A-10
A-17 Shifts A-10
A-18 State Times to Access the Port SFRs A-12
A-19 Summary of Add and Subtract Instructions A-14
A-20 Summary of Compare Instructions A-15
A-21 Summary of Increment and Decrement Instructions A-16
A-22 Summary of Multiply, Divide, and Decimal-adjust Instructions A-16
A-23 Summary of Logical Instructions A-17
A-24 Summary of Move Instructions A-19
A-25 Summary of Exchange, Push, and Pop Instructions A-22
A-26 Summary of Bit Instructions A-23
A-27 Summary of Control Instructions A-24
A-28 Flag Symbols A-26
B-1 Signals Arranged by Functional Categories B-1
B-2 Description of Columns of Table B-3 .... B-2
B-3 Signal Descriptions B-2
C-1 8XC251SB Special Function Registers (SFRs) C-1

TABLES

Table **Page**

1

Guide to This Manual

CHAPTER 1
GUIDE TO THIS MANUAL

This manual describes the 8XC251SB embedded microcontroller which is the first member of the MCS® 251 microcontroller family. It is intended for use by both software and hardware designers familiar with the principles of microcontrollers.

1.1 MANUAL CONTENTS

This manual contains 13 chapters and 3 appendixes. This chapter, Chapter 1, provides an overview of the manual. This section summarizes the contents of the remaining chapters and appendixes. The remainder of this chapter describes notational conventions and terminology used throughout the manual and provides references to related documentation.

Chapter 2 — Architectural Overview — provides an overview of device hardware. It covers core functions (pipelined CPU, clock and reset unit, and on-chip memory) and on-chip peripherals (timer/counters, watchdog timer, programmable counter array, and serial I/O port.)

Chapter 3 — Address Spaces — describes the three address spaces of the MCS 251 microcontroller: memory address space, special function register (SFR) space, and the register file. It also provides a map of the SFR space showing the location of the SFRs and their reset values and explains the mapping of the address spaces of the MCS® 51 architecture into the address spaces of the MCS 251 architecture.

Chapter 4 — Programming — provides an overview of the instruction set. It describes each instruction type (control, arithmetic, and logical, etc.) and lists the instructions in tabular form. This chapter also discusses the binary mode and source mode configurations, addressing modes, bit instructions, and the program status words. For additional information about the instruction set, see Appendix A.

Chapter 5 — Interrupts — describes the 8XC251SB interrupt circuitry which provides a TRAP instruction interrupt and seven maskable interrupts: two external interrupts, three timer interrupts, a PCA interrupt, and a serial port interrupt. This chapter also discusses the interrupt priority scheme, interrupt enable, interrupt processing, and interrupt response time.

Chapter 6— Input/Output Ports — describes the four 8-bit I/O ports (ports 0–3) and explains how to configure them for general-purpose I/O and alternate special functions. It also describes the use of ports 2 and 4 as the external address/data bus.

Chapter 7— Timer/Counters and WDT — describes the three on-chip timer/counters and discusses their application. This chapter also provides instructions for using the hardware watchdog timer (WDT) and describes the operation of the WDT during the idle and powerdown modes.

Chapter 8 — Programmable Counter Array (PCA) — describes the PCA on-chip peripheral and explains how to configure it for general-purpose applications (timers and counters) and special applications (programmable WDT and pulse-width modulator).

Chapter 9 — Serial I/O Port — describes the full-duplex serial I/O port and explains how to program it to communicate with external peripherals. This chapter also discusses baud rate generation, framing error detection, multiprocessor communications, and automatic address recognition.

Chapter 10 — Minimum Hardware Considerations — describes the basic requirements for operating the 8XC251SB in a system. It also discusses on-chip and external clock sources and describes device resets, including power-on reset.

Chapter 11 — Special Operating Modes — provides an overview of the idle, powerdown, and on-circuit emulation (ONCE) modes and describes how to enter and exit each mode. This chapter also describes the (PCON) register and lists the status of the device pins during the special modes and reset (Table 11-1).

Chapter 12 — External Memory Interface — discusses the options available for configuring the external memory interface for a variety of applications. These options include page mode (for accelerated external code fetches), the number of external address bits (16 or 17), the number of external wait states, the regions of memory for strobing PSEN# and RD#, and making a portion of the on-chip code memory accessible as data. This chapter also discusses external memory signals, control registers, and external bus cycles and their timing, and provides several examples of external memory designs.

Chapter 13 — Programming and Verifying Nonvolatile Memory — provides instructions for programming and verifying on-chip code memory, configuration bytes, signature bytes, lock bits and the encryption array. This chapter provides the bit definitions of the configuration bytes.

Appendix A — Instruction Set Reference — provides reference information for the instruction set. It describes each instruction; defines the bits in the program status word registers (PSW, PSW1); shows the relationships between instructions and PSW flags; and lists hexadecimal opcodes, instruction lengths, and execution times. For additional information about the instruction set, see Chapter 4, “Programming.”

Appendix B — Signal Descriptions — describes the function(s) of each device pin. Descriptions are listed alphabetically by signal name. This appendix also provides a list of the signals grouped by functional category.

Appendix C — Registers — provides for convenient reference a copy of the register definition figures that appear throughout the manual.

1.2 NOTATIONAL CONVENTIONS AND TERMINOLOGY

The following notations and terminology are used in this manual. The Glossary defines other terms with special meanings.

#	The pound symbol (#) has either of two meanings, depending on the context. When used with a signal name, the symbol means that the signal is active low. When used in an instruction, the symbol prefixes an immediate value in immediate addressing mode.
italics	Italics identify variables and introduce new terminology. The context in which italics are used distinguishes between the two possible meanings. Variables in registers and signal names are commonly represented by *x* and *y*, where *x* represents the first variable and *y* represents the second variable. For example, in register P*x.y*, *x* represents the variable [1–4] that identifies the specific port, and *y* represents the register bit variable [7:0]. Variables must be replaced with the correct values when configuring or programming registers or identifying signals.
XXXX	Uppercase X (no italics) represents an unknown value or a “don’t care” state or condition. The value may be either binary or hexadecimal, depending on the context. For example, 2XAFH (hex) indicates that bits 11:8 are unknown; 10XX in binary context indicates that the two LSBs are unknown.
Assert and Deassert	The terms *assert* and *deassert* refer to the act of making a signal active (enabled) and inactive (disabled), respectively. The active polarity (high/low) is defined by the signal name. Active-low signals are designated by a pound symbol (#) suffix; active-high signals have no suffix. To assert RD# is to drive it low; to assert ALE is to drive it high; to deassert RD# is to drive it high; to deassert ALE is to drive it low.
Instructions	Instruction mnemonics are shown in upper case to avoid confusion. You may use either upper case or lower case.
Logic 0 (Low)	An input voltage level equal to or less than the maximum value of V_{IL} or an output voltage level equal to or less than the maximum value of V_{OL}. See data sheet for values.
Logic 1 (High)	An input voltage level equal to or greater than the minimum value of V_{IH} or an output voltage level equal to or greater than the minimum value of V_{OH} . See data sheet for values.

Numbers

Hexadecimal numbers are represented by a string of hexadecimal digits followed by the character *H*. Decimal and binary numbers are represented by their customary notations. (That is, 255 is a decimal number and 1111 1111 is a binary number. In some cases, the letter *B* is added for clarity.)

Register Bits

Bit locations are indexed by 7:0 for byte registers, 15:0 for word registers, ands 31:0 for double-word (dword) registers, where bit 0 is the least-significant bit and 7, 15, or 31 is the most-significant bit. An individual bit is represented by the register name, followed by a period and the bit number. For example, PCON.4 is bit 4 of the power control register. In some discussions, bit names are used. For example, the name of PCON.4 is POF, the power off flag.

Register Names

Register names are shown in upper case. For example, PCON is the power control register. If a register name contains a lowercase character, it represents more than one register. For example, CCAPM*x* represents the five registers: CCAPM0 through CCAPM4.

Reserved Bits

Some registers contain *reserved* bits. These bits are not used in this device, but they may be used in future implementations. Do not write a “1” to a reserved bit. The value read from a reserved bit is indeterminate.

Set and Clear

The terms *set* and *clear* refer to the value of a bit or the act of giving it a value. If a bit is *set,* its value is “1”; *setting* a bit gives it a “1” value. If a bit is *clear,* its value is “0”; *clearing* a bit gives it a “0” value.

Signal Names

Signal names are shown in upper case. When several signals share a common name, an individual signal is represented by the signal name followed by a number. Port pins are represented by the port abbreviation, a period, and the pin number (e.g., P0.0, P0.1). A pound symbol (#) appended to a signal name identifies an active-low signal.

Units of Measure

The following abbreviations are used to represent units of measure:

A	amps, amperes
DCV	direct current volts
Kbyte	kilobytes
KΩ	kilo-ohms
mA	milliamps, milliamperes
Mbyte	megabytes
MHz	megahertz

ms	milliseconds
mW	milliwatts
ns	nanoseconds
pF	picofarads
W	watts
V	volts
µA	microamps, microamperes
µF	microfarads
µs	microseconds
µW	microwatts

1.3 RELATED DOCUMENTS

The following documents contain additional information that is useful in designing systems that incorporate the 8XC251SB microcontroller. To order documents, please call Intel Literature Fulfillment (1-800-548-4725 in the U.S. and Canada; +44(0) 793-431155 in Europe).

Embedded Microcontrollers	Order Number 270646
Embedded Processors	Order Number 272396
Embedded Applications	Order Number 270648
Packaging	Order Number 240800

1.3.1 Data Sheet

The data sheet is included in *Embedded Microcontrollers* and is also available individually.

8XC251SB CHMOS Single-Chip 8-bit Microcontroller (Commercial/Express)	Order Number 272459

1.3.2 Application Notes

The following application notes apply to the MCS 251 microcontroller.

AP-125, Designing Microcontroller Systems for Electrically Noisy Environments	Order Number 210313
AP-155, Oscillators for Microcontrollers	Order Number 230659
AP-709, Maximizing Performance Using MCS 251 Microcontroller -Programming the 8XC251SB	Order Number 272671

The following MCS 51 microcontroller application notes also apply to the MCS 251 microcontroller.

AP70, Using the Intel MCS 51 Boolean Processing Capabilities	Order Number 203830
AP-223, *8051 Based CRT Terminal Controller*	Order Number 270032
AP-252, *Designing With the 80C51BH*	Order Number 270068
AP-425, *Small DC Motor Control*	Order Number 270622
AP-410, *Enhanced Serial Port on the 83C51FA*	Order Number 270490
AP-415, *83C51FA/FB PCA Cookbook*	Order Number 270609
AP-476, How to Implement I^{2}C Serial Communication Using Intel MCS 51 Microcontrollers	Order Number 272319

1.4 CUSTOMER SERVICE

This section provides telephone numbers and describes various customer services.

- Customer Support (U.S. and Canada) 800-628-8686
- Customer Training (U.S. and Canada) 800-234-8806
- Literature Fulfillment
  - 800-468-8118 (U.S. and Canada)
  - +44(0)793-431155 (Europe)
- FaxBack* Service
  - 800-628-2283 (U.S. and Canada)
  - +44(0)793-496646 (Europe)
  - 916-356-3105 (worldwide)
- Application Bulletin Board System
  - 800-897-2536 (U.S. and Canada)
  - 916-356-3600 (worldwide, up to 14.4-Kbaud line)
  - 916-356-7209 (worldwide, dedicated 2400-baud line)
  - +44(0)793-496340 (Europe)

Intel provides 24-hour automated technical support through our FaxBack service and our centralized Intel Application Bulletin Board System (BBS). The FaxBack service is a simple-to-use information system that lets you order technical documents by phone for immediate delivery to your fax machine. The BBS is a centralized computer bulletin board system that provides updated application-specific information about Intel products.

1.4.1 How to Use Intel's FaxBack Service

Think of the FaxBack service as a library of technical documents that you can access with your phone. Just dial the telephone number (see page 1-7) and respond to the system prompts. After you select a document, the system sends a copy to your fax machine.

Each document is assigned an order number and is listed in a subject catalog. First-time users should order the appropriate subject catalogs to get a complete listing of document order numbers.

The following catalogs and information packets are available:

1. Microcontroller, Flash, and iPLD catalog
2. Development Tools Handbook
3. System catalog
4. DVI and multimedia catalog
5. BBS catalog
6. Microprocessor and peripheral catalog
7. Quality and reliability catalog
8. Technical questionnaire

1.4.2 How to Use Intel's Application BBS

The Application Bulletin Board System (BBS) provides centralized access to information, software drivers, firmware upgrades, and revised software. Any user with a modem and computer can access the BBS. Use the following modem settings.

- 14400, N, 8, 1

If your modem does not support 14.4K baud, the system provides auto configuration support for 1200- through 14.4K-baud modems.

To access the BBS, just dial the telephone number (see page 1-7) and respond to the system prompts. During your first session, the system asks you to register with the system operator by entering your name and location. The system operator will then set up your access account within 24 hours. At that time, you can access the files on the BBS. For a listing of files, call the FaxBack service and order catalog #6 (the BBS catalog).

If you encounter any difficulty accessing our high-speed modem, try our dedicated 2400-baud modem (see page 1-7). Use the following modem settings.

- 2400 baud, N, 8, 1

1.4.3 How to Find the Latest *Ap*BUILDER Files and Hypertext Manuals and Data Sheets on the BBS

The latest *Ap*BUILDER files and hypertext manuals and data sheets are available first from the BBS. To access the files:

1. Select [F] from the BBS Main menu.
2. Select [L] from the Intel Apps Files menu.
3. The BBS displays the list of all area levels and prompts for the area number.
4. Select [25] to choose the *Ap*BUILDER / Hypertext area.
5. Area level 25 has four sublevels: (1) General, (2) 196 Files, (3) 186 Files, and (4) 8051 Files.
6. Select [1] to find the latest *Ap*BUILDER files or the number of the appropriate product-family sublevel to find the hypertext manuals and data sheets.
7. Enter the file number to tag the files you wish to download. The BBS displays the approximate download time for tagged files.

2

Architectural Overview

CHAPTER 2
ARCHITECTURAL OVERVIEW

The 8XC251SB is the first microcontroller in Intel's family of MCS® 251 microcontrollers. This family of 8-bit microcontrollers extends the features and performance of the widely-used MCS 51 microcontrollers, while providing binary-code compatibility. Pin compatible with the 8XC51FX, the 8XC251SB provides a high-performance upgrade with minimal impact on existing hardware and software. Typical control applications for the 8XC251SB include copiers, scanners, and CD ROM and tape drives. It is also well suited for communications applications, such as phone terminals, business/feature phones, and phone switching and transmission systems.

All MCS 251 microcontrollers share a set of common features:

- 24-bit linear addressing and up to 16 Mbytes of memory
- a register-based CPU with registers accessible as bytes, words, and double words.
- a page mode for accelerating external instruction fetches
- an instruction pipeline
- an enriched instruction set, including 16-bit arithmetic and logic instructions
- a 64-Kbyte extended stack space
- a minimum instruction-execution time of two clocks (vs. 12 clocks for MCS 51 microcontrollers)
- binary-code compatibility with MCS 51 microcontrollers

Several benefits are derived from these features:

- preservation of code written for MCS 51 microcontrollers
- a significant increase in core execution speed in comparison with MCS 51 microcontrollers at the same clock rate
- support for larger programs and more data
- increased efficiency for code written in C

Figure 2-1 is a functional block diagram of the 8XC251SB. The core, which is common to all MCS 251 microcontrollers, is described in "8XC251SB Core" on page 2-4. A specific microcontroller in the family has its own on-chip peripherals, I/O ports, external system bus, size of on-chip RAM, and type and size of on-chip program memory.

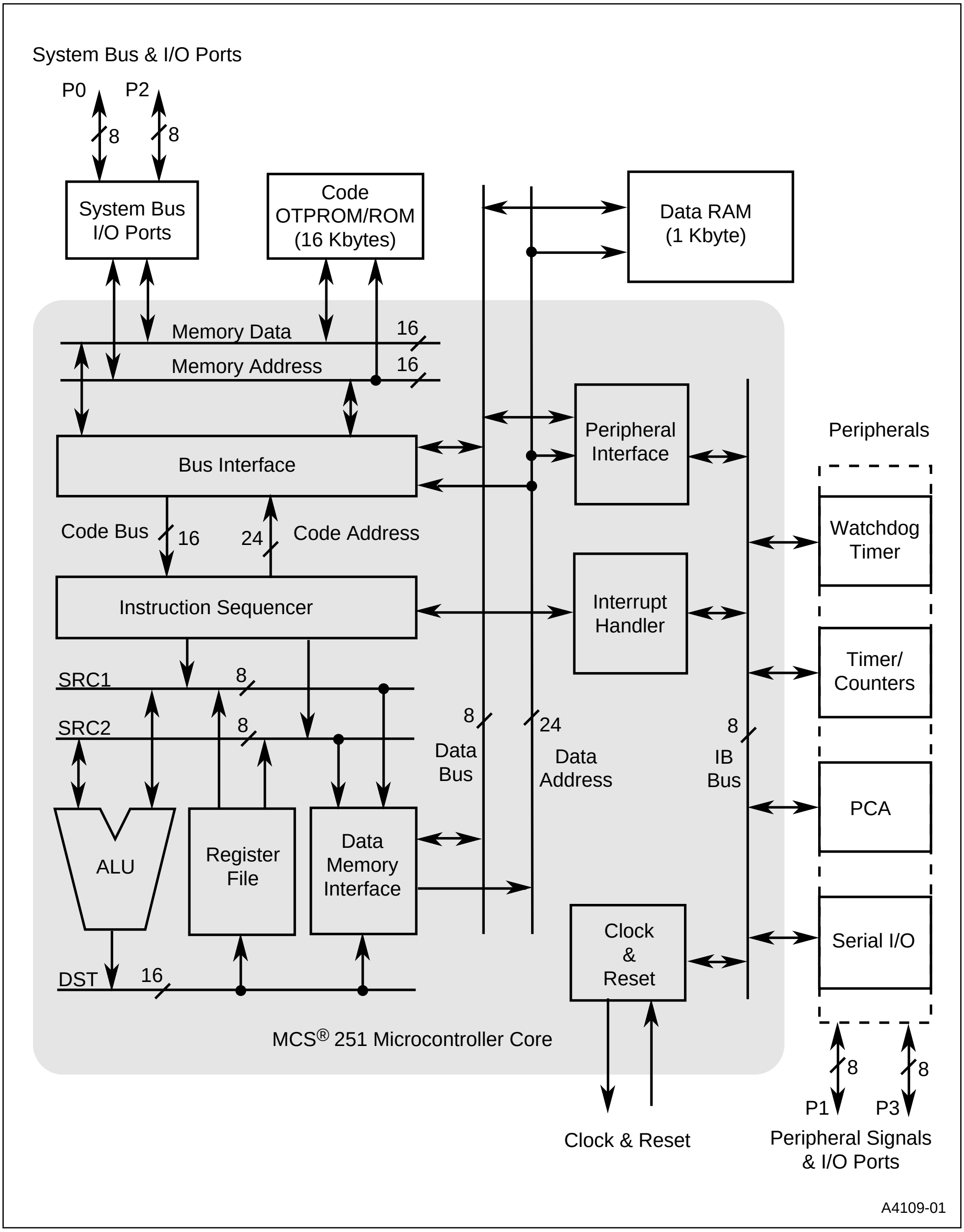

System Bus & I/O Ports
P0
P2
8
8
System Bus I/O Ports
Code OTPROM/ROM (16 Kbytes)
Data RAM (1 Kbyte)
Memory Data
16
Memory Address
16
Peripheral Interface
Peripherals
Bus Interface
Code Bus
16
24
Code Address
Watchdog Timer
Instruction Sequencer
Interrupt Handler
Timer/ Counters
SRC1
8
SRC2
8
8
Data Bus
24
Data Address
8
IB Bus
PCA
ALU
Register File
Data Memory Interface
Serial I/O
Clock & Reset
DST
16
MCS® 251 Microcontroller Core
8
8
P1
P3
Clock & Reset
Peripheral Signals & I/O Ports
A4109-01

Figure 2-1. Functional Block Diagram of the 8XC251SB

The 8XC251SB peripherals include a dedicated watchdog timer, a timer/counter unit, a programmable counter array (PCA), and a serial I/O unit. The 8XC251SB has four 8-bit I/O ports, P0–P4. Each port pin can be individually programmed as a general I/O signal or a special-function signal that supports the external bus or one of the on-chip peripherals. Ports P0 and P2 comprise the external bus, which has 16 lines that are multiplexed for a 16-bit address and 8-bit data. (You can also configure the 8XC251SB to have a 17th external address bit. See Chapter 12, "External Memory Interface.") Ports P1 and P3 comprise bus-control and peripheral signals.

The 8XC251SB has two power-saving modes. In idle mode, the CPU clock is stopped, while clocks to the peripherals continue to run. In powerdown mode, the on-chip oscillator is stopped, and the chip enters a static state. An enabled interrupt or a hardware reset can bring the chip back to its normal operating mode from idle or powerdown. See Chapter 11, "Special Operating Modes" for details on the power-saving modes.

MCS 251 microcontrollers use an instruction set that has been expanded to include new operations, addressing modes, and operands. Many instructions can operate on 8-, 16-, or 32-bit operands, providing easier and more efficient programming in high-level languages such as C. Additional new features include the TRAP instruction, a new displacement addressing mode, and several conditional jump instructions. Chapter 4, "Programming," describes the instruction set and compares it with the instruction set for MCS 51 microcontrollers.

You can configure the 8XC251SB to run in *binary mode* or *source mode*. In either mode, the 8XC251SB can execute all instructions in the MCS 51 architecture and the MCS 251 architecture. However, source mode is more efficient for MCS 251 architecture instructions, and binary mode is more efficient for MCS 51 architecture instructions. In binary mode, object code for an MCS 51 microcontroller can run on the 8XC251SB without recompiling.

If a system was originally developed using an MCS 51 microcontroller, and if the new 8XC251SB-based system will run code written for the MCS 51 microcontroller, performance will be better with the 8XC251SB running in binary mode. Object code written for the MCS 51 microcontroller runs faster on the 8XC251SB.

However, if most of the code is rewritten using the new instruction set, performance will be better with the 8XC251SB running in source mode. In this case the 8XC251SB can run significantly faster than the MCS 51 microcontroller. See Chapter 4, "Programming" for a discussion of binary mode and source mode.

MCS 251 microcontrollers store both code and data in a single, linear 16-Mbyte memory space. The 8XC251SB can address up to 128 Kbytes of external memory. The special function registers (SFRs) and the register file have separate address spaces. See Chapter 3, "Address Spaces" for a description of the address spaces.

Table 2-1 summarizes some features of the 8XC251SB.

Table 2-1. Summary of 8XC251SB Features

Address Space	Register File	Code Memory	Data RAM	I/O Lines	External Bus	Interrupt Sources
256 Kbytes	40 bytes	83C251SB: 16 Kbytes ROM 87C251SB: 16 Kbytes OTPROM 80C251SB: 0 Kbytes	1 Kbyte	32	Multiplexed: 16/17 Address Bits 8 Data Bits	11

2.1 8XC251SB CORE

The 8XC251SB core architecture contains the clock and reset unit, the interrupt handler, the bus interface, the peripheral interface, and the CPU. The CPU contains the instruction sequencer, ALU, register file, and data memory interface.

2.1.1 CPU

Figure 2-2 is a functional block diagram of the CPU (central processor unit). The 8XC251SB fetches instructions from on-chip code memory two bytes at a time or from external memory in single bytes. The instructions are sent over the 16-bit code bus to the execution unit. You can configure the 8XC251SB to operate in *page mode* for accelerated instruction fetches from external memory. In page mode, if an instruction fetch is to the same 256-byte “page” as the previous fetch, the fetch requires one state (two clocks) rather than two states (four clocks).

The 8XC251SB register file has forty registers, which can be accessed as bytes, words, and double words. As in the MCS 51 architecture, registers 0–7 consist of four banks of eight registers each, where the active bank is selected by the program status word (PSW) for fast context switches.

The 8XC251SB is a single-pipeline machine. When the pipeline is full and code is executing from on-chip code memory, an instruction is completed every state time. When the pipeline is full and code is executing from external memory (with no wait states and no extension of the ALE signal) an instruction is completed every two state times.

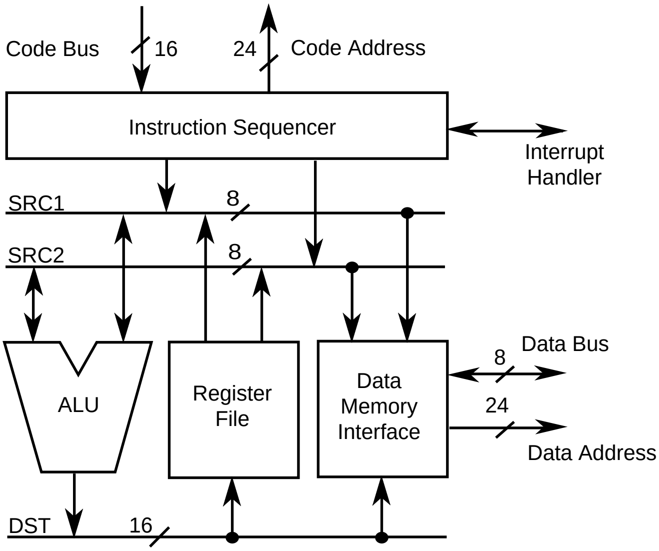

Code Bus
16
24
Code Address
Instruction Sequencer
Interrupt Handler
SRC1
8
SRC2
8
ALU
Register File
Data Memory Interface
8
Data Bus
24
Data Address
DST
16

Figure 2-2. The CPU

2.1.2 Clock and Reset Unit

The timing source for the 8XC251SB can be an external oscillator or an internal oscillator with an external crystal/resonator (see Chapter 10, “Minimum Hardware Setup”). The basic unit of time in MCS 251 microcontrollers is the *state time* (or *state*), which is two oscillator periods (see Figure 2-3). The state time is divided into *phase 1* and *phase 2*.

The 8XC251SB peripherals operate on a *peripheral cycle*, which is six state times. (This peripheral cycle is particular to the 8XC251SB and not a characteristic of the MCS 251 architecture.) A one-clock interval in a peripheral cycle is denoted by its state and phase. For example, the PCA timer is incremented once each peripheral cycle in phase 2 of state 5 (denoted as S5P2).

The reset unit places the 8XC251SB into a known state. A chip reset is initiated by asserting the RST pin or allowing the watchdog timer to time out (see Chapter 10, “Minimum Hardware Setup”).

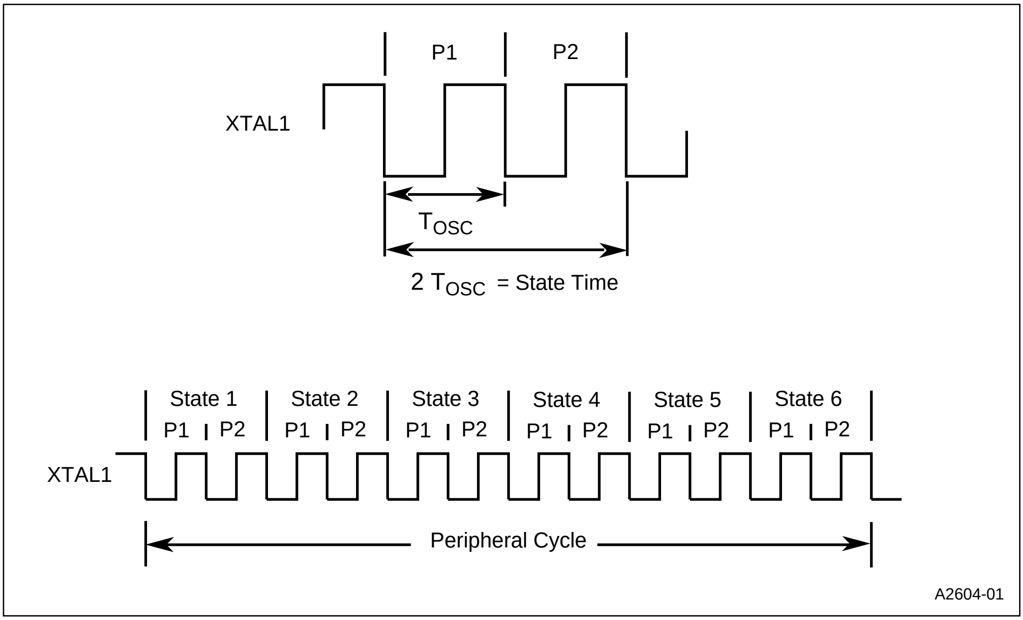

P1
P2
XTAL1
T_{OSC}
2 T_{OSC} = State Time
State 1
State 2
State 3
State 4
State 5
State 6
P1
P2
XTAL1
Peripheral Cycle
A2604-01

Figure 2-3. 8XC251SB Timing

2.1.3 Interrupt Handler

The interrupt handler can receive interrupt requests from eleven sources: seven maskable sources and the TRAP instruction. When the interrupt handler grants an interrupt request, the CPU discontinues the normal flow of instructions and branches to a routine that services the source that requested the interrupt. You can enable or disable the interrupts individually (except for TRAP) and you can assign one of four priority levels to each interrupt. See Chapter 5, “Interrupt System” for a detailed description.

2.1.4 On-chip Code Memory

For the 83C251SB and the 87C251SB, memory locations FF:0000H–FF:3FFFH are implemented with 16-Kbytes of on-chip code memory (ROM in the 83C251SB and EPROM in the 87C251SB). Following a reset, the first instruction is fetched from location FF:0000H. For the 80C251SB location FF:0000H is always in external memory.

2.1.5 On-chip RAM

The 8XC251SB has 1-Kbyte of on-chip data RAM (locations 20H–41FH) which can be accessed with direct, indirect, and displacement addressing. Ninety-six of these locations (20H–7FH) are bit addressable. An additional 32 bytes of on-chip RAM (00H–1FH) provide storage for the four banks of registers R0–R7.

2.2 ON-CHIP PERIPHERALS

The on-chip peripherals, which lie outside the core, perform specialized functions. Software accesses the peripherals via their special function registers (SFRs). The 8XC251SB has four peripherals: the watchdog timer, the timer/counters, the programmable counter array (PCA), and the serial I/O port.

2.2.1 Timer/Counters and Watchdog Timer

The timer/counter unit has three timer/counters, which can be clocked by the oscillator (for timer operation) or by an external input (for counter operation). You can set up an 8-bit, 13-bit, or 16-bit timer/counter, and you can program them for special applications, such as capturing the time of an event on an external pin, outputting a programmable clock signal on an external pin, or generating a baud rate for the serial I/O port. Timer/counter events can generate interrupt requests.

The watchdog timer is a circuit that automatically resets the 8XC251SB in the event of a hardware or software upset. When enabled by software, the watchdog timer begins running, and unless software intervenes, the timer reaches a maximum count and initiates a chip reset. In normal operation, software periodically clears the timer register to prevent the reset. If an upset occurs and software fails to clear the timer, the resulting chip reset disables the timer and returns the system to a known state. The watchdog and the timer/counters are described in Chapter 7, "Timer/Counters and WatchDog Timer."

2.2.2 Programmable Counter Array (PCA)

The programmable counter array (PCA) has its own timer and five capture/compare modules that perform several functions: capturing (storing) the timer value in response to a transition on an input pin; generating an interrupt request when the timer matches a stored value; toggling an output pin when the timer matches a stored value; generating a programmable PWM (pulse width modulator) signal on an output pin; and serving as a software watchdog timer. Chapter 8, "Programmable Counter Array" describes this peripheral in detail.

2.2.3 Serial I/O Port

The serial I/O port provides one synchronous and three asynchronous communication modes. The synchronous mode (mode 0) is half-duplex: the serial port outputs a clock signal on one pin and transmits or receives data on another pin.

The asynchronous modes (modes 1–3) are full-duplex (i.e., the port can send and receive simultaneously). Mode 1 uses a serial frame of 10 bits: a start bit, 8 data bits, and a stop bit. The baud rate is generated by overflow of timer 1 or timer 2. Modes 2 and 3 use a serial frame of 11 bits: a start bit, eight data bits, a programmable ninth data bit, and a stop bit. The ninth bit can be used for parity checking or to specify that the frame contains an address and data. In mode 2, you can use a baud rate of 1/32 or 1/64 of the oscillator frequency. In mode 3, you can use the overflow from timer 1 or timer 2 to determine the baud rate.

In its synchronous modes (modes 1–3) the serial port can operate as a slave in an environment where multiple slaves share a single serial line. It can accept a message intended for itself or a message that is being broadcast to all of the slaves, and it can ignore a message sent to another slave.

3

Address Spaces

CHAPTER 3
ADDRESS SPACES

MCS® 251 microcontrollers have three address spaces: a memory space, a special function register (SFR) space, and a register file. This chapter describes these address spaces as they apply to all MCS 251 microcontrollers and to the 8XC251SB in particular. It also discusses the compatibility of the MCS 251 architecture and the MCS 51 architecture in terms of their address spaces.

3.1 ADDRESS SPACES FOR MCS® 251 MICROCONTROLLERS

Figure 3-1 shows the memory space, the SFR space, and the register file for MCS 251 microcontrollers. (The address spaces are depicted as being eight bytes wide with addresses increasing from left to right.)

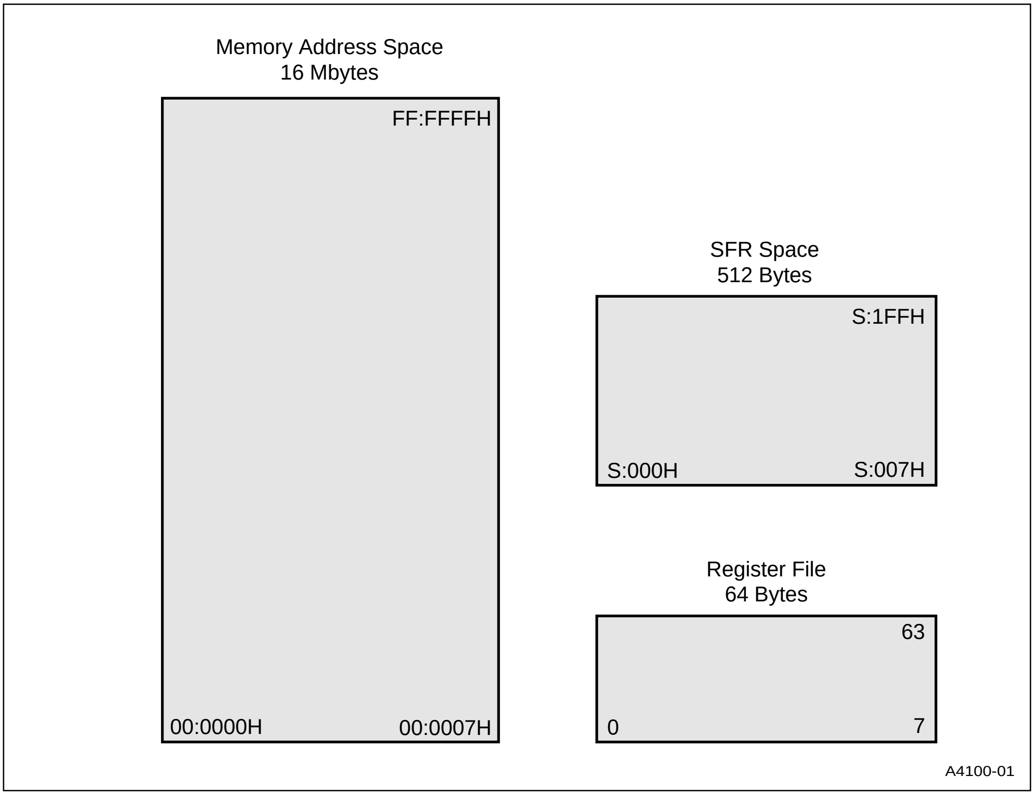

Memory Address Space
16 Mbytes
FF:FFFFH
00:0000H
00:0007H
SFR Space
512 Bytes
S:1FFH
S:000H
S:007H
Register File
64 Bytes
63
0
7
A4100-01

Figure 3-1. Address Spaces for MCS® 251 Microcontrollers

It is convenient to view the unsegmented, 16-Mbyte memory space as consisting of 256 64-Kbyte regions, numbered 00: to FF:.

NOTE

> The memory space in the MCS 251 architecture is unsegmented. The 64-Kbyte "regions" 00:, 01:, ..., FF: are introduced only as a convenience for discussions. Addressing in the MCS 251 architecture is linear; there are **no** segment registers.

MCS 251 microcontrollers can have up to 64 Kbytes of on-chip code memory in region FF:. On-chip data RAM begins at location 00:0000H. The first 32 bytes (00:0000H–00:001FH) provide storage for a part of the register file. On-chip, general-purpose data RAM begins at 00:0020H. The sizes of the on-chip code memory and on-chip RAM depend on the particular device.

The register file has its own address space (Figure 3-1). The 64 locations in the register file are numbered decimally from 0 to 63. Locations 0–7 represent one of four, switchable register banks, each having 8 registers (see "The 8XC251SB Register File" on page 3-8). The 32 bytes required for these banks occupy locations 00:0000H–00:001FH in the memory space. Register file locations 8–63 do not appear in the memory space.

The SFR space can accommodate up to 512 8-bit special function registers with addresses S:000H–S:1FFH. Some of these locations may be unimplemented in a particular device. In the MCS 251 architecture, the prefix "S:" is used with SFR addresses to distinguish them addresses from the memory space addresses 00:0000H–00:01FFH.

3.1.1 Compatibility with the MCS® 51 Architecture

The address spaces in the MCS 51 architecture are mapped into the address spaces in the MCS 251 architecture. This mapping allows code written for MCS 51 microcontrollers to run on MCS 251 microcontrollers. (Chapter 4, "Programming," discusses the compatibility of the two instruction sets.)

Figure 3-2 shows the address spaces for the MCS 51 architecture[†]. Internal data memory locations 00H–7FH can be addressed directly and indirectly. Internal data locations 80H–FFH can only be addressed indirectly. Directly addressing these locations accesses the Special Function Registers (SFRs).

The register file (registers R0–R7) comprises four, switchable register banks, each having 8 registers. The 32 bytes required for the four banks occupy locations 00H–1FH in the on-chip data memory.

[†] *MCS®51 Microcontroller Family User's Manual* (Order Number: 272383)

The 64-Kbyte code memory has a separate memory space. Data in the code memory can be accessed only with the MOVC instruction. Similarly, the 64-Kbyte external data memory can be accessed only with the MOVX instruction.

Figure 3-3 shows how the address spaces in the MCS 51 architecture map into the address spaces in the MCS 251 architecture; details are listed in Table 3-1.

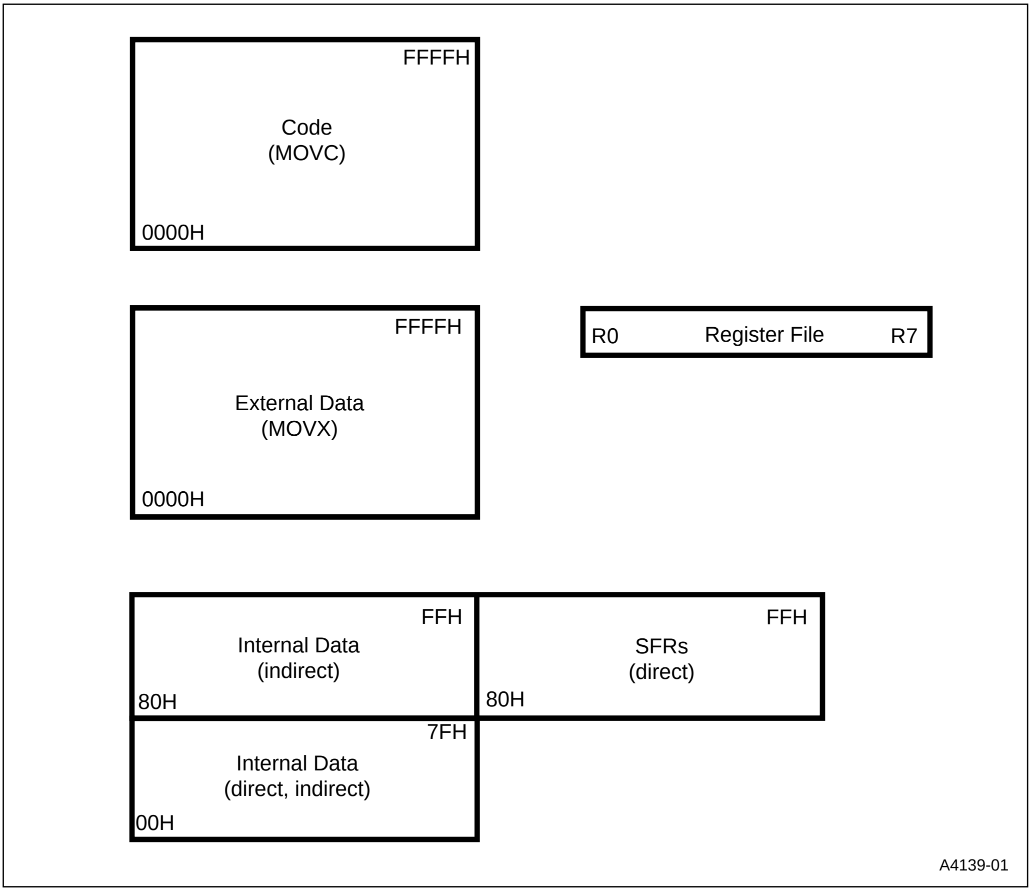

FFFFH
Code
(MOVC)
0000H
FFFFH
External Data
(MOVX)
0000H
R0
Register File
R7
FFH
Internal Data
(indirect)
80H
FFH
SFRs
(direct)
80H
7FH
Internal Data
(direct, indirect)
00H
A4139-01

Figure 3-2. Address Spaces for the MCS® 51 Architecture

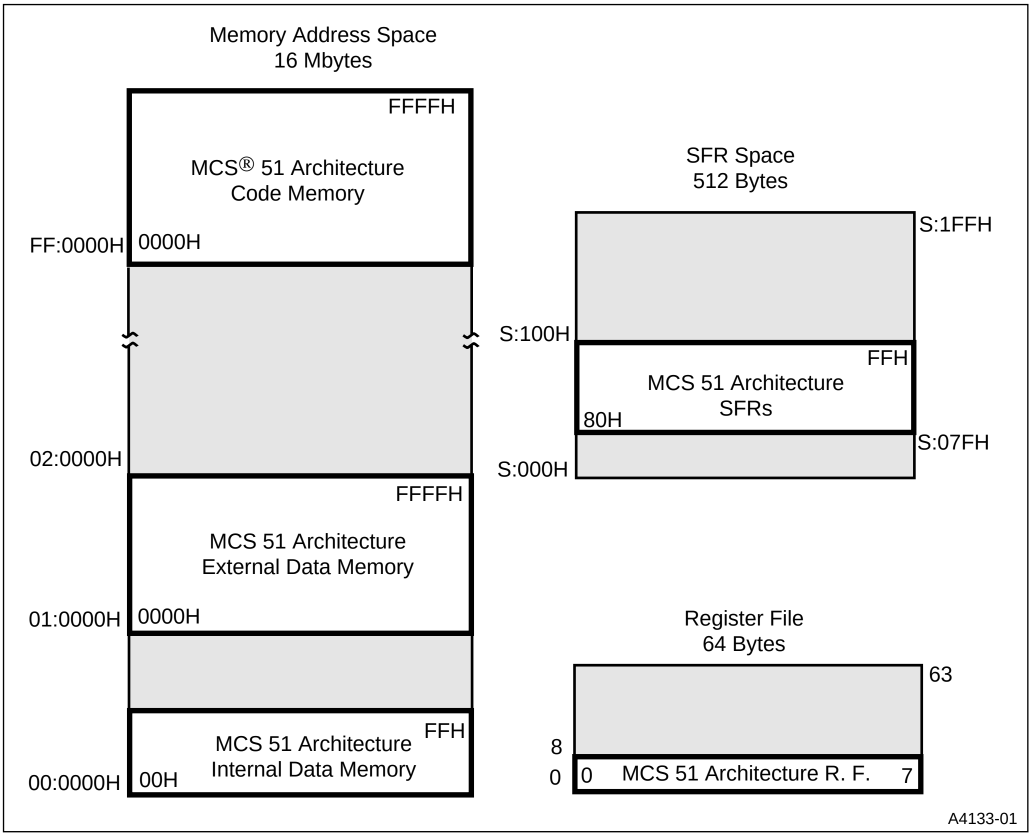

Memory Address Space
16 Mbytes
FFFFH
MCS® 51 Architecture
Code Memory
FF:0000H
0000H
02:0000H
FFFFH
MCS 51 Architecture
External Data Memory
01:0000H
0000H
MCS 51 Architecture
Internal Data Memory
FFH
00:0000H
00H
SFR Space
512 Bytes
S:1FFH
S:100H
FFH
MCS 51 Architecture
SFRs
80H
S:07FH
S:000H
Register File
64 Bytes
63
8
0
0
MCS 51 Architecture R. F.
7
A4133-01

Figure 3-3. Address Space Mappings MCS® 51 Architecture to MCS® 251 Architecture

Table 3-1. Address Mappings

Memory Type	MCS® 51 Architecture			MCS® 251 Architecture
	Size	Location	Data Addressing	Location
Code	64 Kbytes	0000H–FFFFH	Indirect using MOVC instr.	FF:0000H–FF:FFFFH
External Data	64 Kbytes	0000H–FFFFH	Indirect using MOVX instr.	01:0000H–01:FFFFH
Internal Data	128 bytes	00H–7FH	Direct, Indirect	00:0000H–00:007FH
	128 bytes	80H–FFH	Indirect	00:0080H–00:00FFH
SFRs	128 bytes	S:80H–S:FFH	Direct	S:080H–S:0FFH
Register File	8 bytes	R0–R7	Register	R0–R7

The 64-Kbyte code memory for MCS 51 microcontrollers maps into region FF: of the memory space for MCS 251 microcontrollers. Assemblers for MCS 251 microcontrollers assemble code for MCS 51 microcontrollers into region FF:, and data accesses to code memory are directed to this region. The assembler also maps the interrupt vectors to region FF:. This mapping is transparent to the user; code executes just as before without modification.

The 64-Kbyte external data memory for MCS 51 microcontrollers is mapped into the memory region specified by bits 16–23 of the data pointer DPX, i.e., DPXL, which is accessible as register file location 57 and also as the SFR at S:084H (see “Dedicated Registers” on page 3-10). The reset value of DPXL is 01H, which maps the external memory to region 01: as shown in Figure 3-3. You can change this mapping by writing a different value to DPXL. A mapping of the MCS 51 microcontroller external data memory into any 64-Kbyte memory region in the MCS 251 architecture provides complete run-time compatibility because the lower 16 address bits are identical in the two address spaces.

The on-chip data memory for MCS 51 microcontrollers is mapped to region 00: to ensure complete run-time compatibility. From location 00H to 7FH, the internal data memory is the same in the two architectures. In the MCS 251 architecture, the data memory extends beyond these 128 bytes to allow enhanced data and stack access using new instructions.

The 128-byte SFR space for MCS 51 microcontrollers is mapped into the 512-byte SFR space of the MCS 251 architecture starting at address S:080H, as shown in Figure 3-3. This provides complete compatibility with direct addressing of MCS 51 microcontroller SFRs (including bit addressing). The SFR addresses are unchanged in the new architecture. In the MCS 251 architecture, SFRs A, B, DPL, DPH, and SP (as well as the new SFRs DPXL and SPH) reside in the register file for high performance. However, to maintain compatibility, they are also mapped into the SFR space at the same addresses as in the MCS 51 architecture.

3.2 THE 8XC251SB MEMORY SPACE

The logical memory space for the 8XC251SB microcontroller is shown in Figure 3-4. The arrows on the left side indicate the addressing modes that apply to the partitions of the memory space. (Chapter 4, “Programming,” discusses addressing modes.) The right side of the figure shows the hardware implementation of the different areas of the memory space. For the 8XC251SB, the usable memory space consists of four 64-Kbyte regions: 00:, 01:, FE:, and FF:. Code can execute from all four regions. Regions 02:–FD: are reserved. Reading a location in the reserved area returns an unspecified value. Software can execute a write to the reserved area, but nothing is actually written.

3.2.1 On-chip General-purpose Data RAM

Memory locations 00:0020H–00:041FH are implemented as 1 Kbyte of on-chip RAM, which can be used for general data storage. Instructions cannot execute from on-chip data RAM. The data is accessible by direct, indirect, and displacement addressing. Locations 00:0020H–00:007FH are also bit addressable.

3.2.2 On-chip Code Memory (87C251SB/83C251SB)

The 87C251SB/83C251SB has 16-Kbytes of on-chip OTPROM/ROM at locations FF:0000H–FF:3FFFH. This memory is intended primarily for code storage, although its contents can also be read as data with the indirect and displacement addressing modes. Following a chip reset, program execution begins at FF:0000H. Chapter 13, "Programming and Verifying Nonvolatile Memory," describes programming and verification of the OTPROM/ROM.

NOTE

> Beware of executing code from the upper eight bytes of the on-chip OTPROM/ROM (FF:3FFF8H–FF:3FFFFH). The 8XC251SB may attempt to prefetch code from external memory (at an address above FF:3FFFH) and thereby disrupt I/O ports 0 and 2. Fetching code constants from these eight bytes does not affect ports 0 and 2.

A code fetch in the range FF:0000H–FF:3FFFH accesses the on-chip OTPROM/ROM only if EA# = 1. For EA# = 0, a code fetch in this address range accesses external memory. The value of EA# is latched when the chip leaves the reset state.

3.2.2.1 Accessing On-chip Code Memory in Region 00:

The 87C251SB/83C251SB can be configured so that the upper 8 Kbytes of the on-chip code memory can be read as data in region 00: (see "Configuration Bytes" on page 13-6). This is useful for accessing code constants stored in OTPROM/ROM. Specifically, the upper 8 Kbytes of code memory are mapped to locations 00:E000H–00:FFFFH (as well as to locations FF:E000H–FF:FFFFH) if the following three conditions hold:

- The 87C251SB/83C251SB is configured with EMAP = 0 in the CONFIG1 register (Chapter 13, "Programming and Verifying Nonvolatile Memory").
- EA# = 1.
- The access is a data read, not a code fetch.

If one or more of these conditions do not hold, accesses to locations 00:E000H–00:FFFFH are referred to external memory.

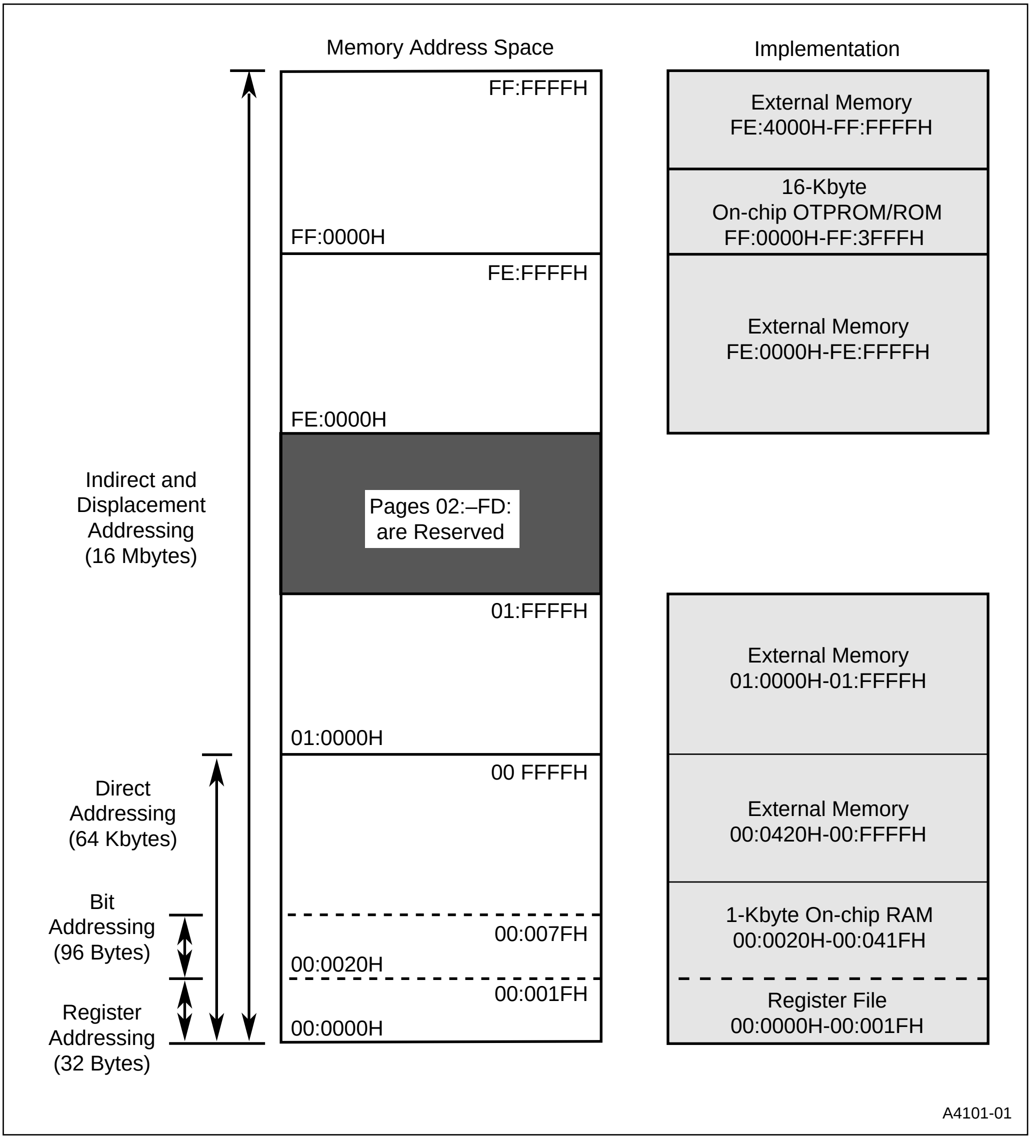

Memory Address Space
Implementation
FF:FFFFH
FF:0000H
FE:FFFFH
FE:0000H
Pages 02:–FD: are Reserved
01:FFFFH
01:0000H
00 FFFFH
00:007FH
00:0020H
00:001FH
00:0000H
External Memory FE:4000H-FF:FFFFH
16-Kbyte On-chip OTPROM/ROM FF:0000H-FF:3FFFH
External Memory FE:0000H-FE:FFFFH
External Memory 01:0000H-01:FFFFH
External Memory 00:0420H-00:FFFFH
1-Kbyte On-chip RAM 00:0020H-00:041FH
Register File 00:0000H-00:001FH
Indirect and Displacement Addressing (16 Mbytes)
Direct Addressing (64 Kbytes)
Bit Addressing (96 Bytes)
Register Addressing (32 Bytes)
A4101-01

Figure 3-4. 8XC251SB Memory Space

3.2.3 External Memory

Regions 01: and FE: and portions of regions 00: and FF: of the memory space are implemented as external memory (Figure 3-4). External memory is described in Chapter 12, “External Memory Interface.”

3.3 THE 8XC251SB REGISTER FILE

The 8XC251SB register file consists of 40 locations: 0–31 and 56–63, as shown in Figure 3-5. Locations 0–7 are in the on-chip RAM. The other locations are in the CPU.

Registers 0–7 actually consist of four switchable banks of eight registers each. These 32 bytes are stored in locations 00:0000H–00:001FH in the memory space and are implemented in the on-chip RAM. However, because these locations are dedicated to the register file, they are not considered a part of the general-purpose, 1-Kbyte on-chip RAM (locations 00:0020H–00:041FH).

Bits RS1 and RS0 in the PSW register select one of the four register banks to be active, i.e., to currently serve as register file locations 0–7, as shown in Table 3-2. (The PSW is described in “Program Status Words” on page 4-17.) This bank selection can be used for fast context switches. The inactive banks are inaccessible via the register file; however, registers in both the active and inactive banks can be addressed as locations in the memory space.

Register file locations 32–55 are reserved and cannot be accessed.

Table 3-2. Register Bank Selection

Bank	Address Range	PSW Selection Bits	
		RS1	RS0
Bank 0	00H–07H	0	0
Bank 1	08H–0FH	0	1
Bank 2	10H-17H	1	0
Bank 3	18H-1FH	1	1

3.3.1 Byte, Word, and Dword Registers

Depending on its location in the register file, a register is addressable as a byte, a word, and/or a dword, as shown in the right side of Figure 3-5. A register is named for its least-significant byte. For example:

R4 is the byte register consisting of location 4.

WR4 is the word register consisting of registers 4 and 5.

DR4 is the dword register consisting of registers 4–7.

Locations R0–R15 are addressable as bytes, words, or dwords. Locations 16–31 are addressable only as words or dwords. Locations 56–63 are addressable only as dwords. Registers are addressed only by the names shown in Figure 3-5 — except for the 32 registers that comprise the four banks of registers R0–R7, which can also be accessed as locations 00:0000H–00:001FH in the memory space.

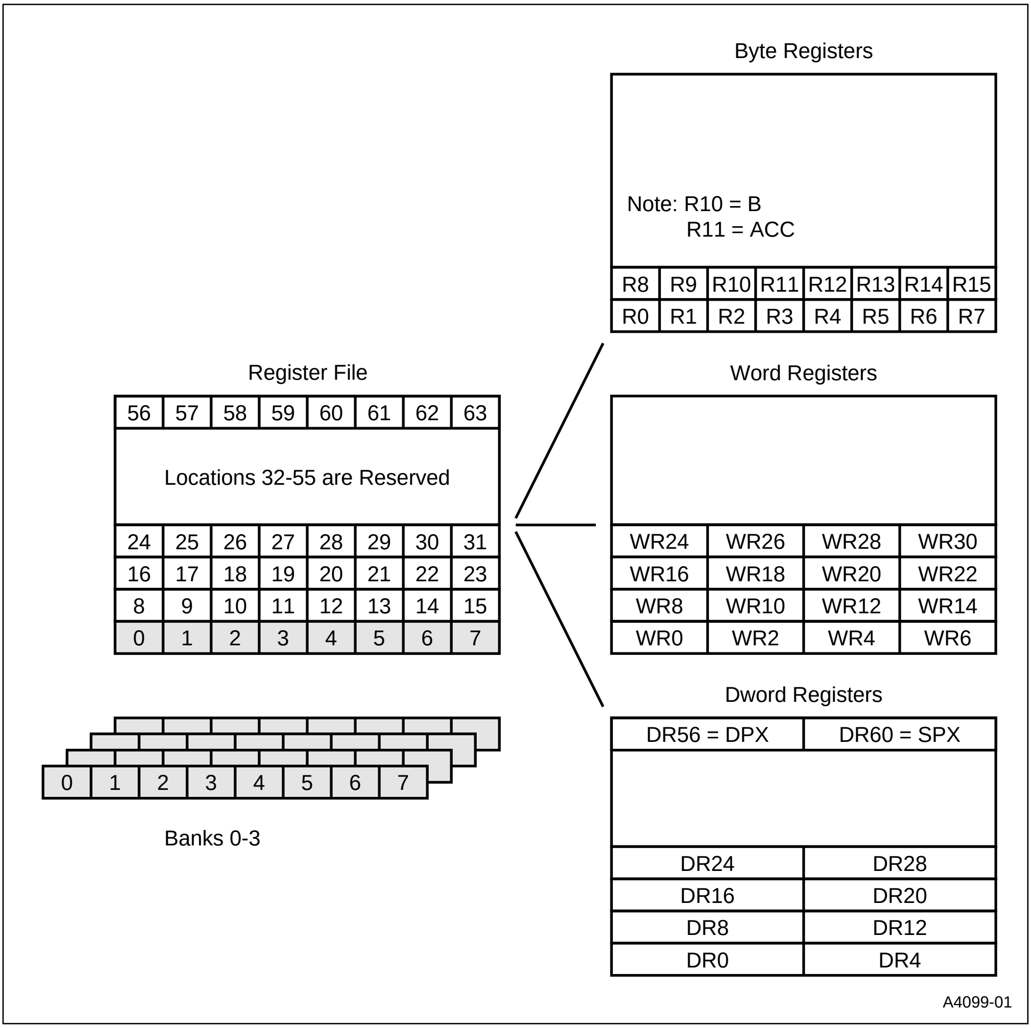

Byte Registers
Note: R10 = B
R11 = ACC
R8 R9 R10 R11 R12 R13 R14 R15
R0 R1 R2 R3 R4 R5 R6 R7
Register File
56 57 58 59 60 61 62 63
Locations 32-55 are Reserved
24 25 26 27 28 29 30 31
16 17 18 19 20 21 22 23
8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7
Word Registers
WR24 WR26 WR28 WR30
WR16 WR18 WR20 WR22
WR8 WR10 WR12 WR14
WR0 WR2 WR4 WR6
Dword Registers
DR56 = DPX
DR60 = SPX
DR24 DR28
DR16 DR20
DR8 DR12
DR0 DR4
0 1 2 3 4 5 6 7
Banks 0-3
A4099-01

Figure 3-5. The Register File

3.3.2 Dedicated Registers

The register file has four dedicated registers:

- R10 is the B-register
- R11 is the accumulator (ACC)
- DR56 is the extended data pointer, DPX
- DR60 is the extended stack pointer, SPX

These registers are located in the register file; however, R10, R11, and some bytes of DR56 and DR60 are also accessible as SFRs. The bytes of DPX and SPX can be accessed in the register file only by addressing the dword registers. The dedicated registers in the register file and their corresponding SFRs are illustrated in Figure 3-6 and listed in Table 3-3 on page 3-12.

3.3.2.1 Accumulator and B Register

The 8-bit *accumulator* (ACC) is byte register R11, which is also accessible in the SFR space as ACC at S:0E0H (Figure 3-6). The *B register*, used in multiplies and divides, is register R10, which is also accessible in the SFR space as B at S:0F0H. Accessing ACC or B as a register is one state faster than accessing them as SFRs.

Instructions in the MCS 51 architecture use the accumulator as the primary register for data moves and calculations. However, in the MCS 251 architecture, any of registers R1–R15 can serve for these tasks[†]. As a result, the accumulator does not play the central role that it has in MCS 51 microcontrollers.

3.3.2.2 Extended Data Pointer, DPX

Dword register DR56 is the *extended data pointer*, DPX (Figure 3-6). The lower three bytes of DPX (DPL, DPH, and DPXL) are accessible as SFRs. DPL and DPH comprise the 16-bit *data pointer* DPTR. While instructions in the MCS 51 architecture always use DPTR as the data pointer, instructions in the MCS 251 architecture can use any word or dword register as a data pointer.

DPXL, the byte in location 58, specifies the region of memory (00:–FF:) that maps into the 64-Kbyte external data memory space in the MCS 51 architecture. In other words, the MOVX instruction addresses the region specified by DPXL when it moves data to and from external memory. The reset value of DPXL is 01H.

[†] Bits in the PSW and PSW1 registers reflect the status of the accumulator. There are no equivalent status indicators for the other registers.

3.3.2.3 Extended Stack Pointer, SPX

Dword register DR60 is the *stack pointer*, SPX (Figure 3-6). The low byte (location 60) is the 8-bit stack pointer, SP, in the MCS 51 architecture. The byte at location 61 is the *stack pointer high*, SPH. The two bytes allow the stack to extend to the top of memory region 00:. SP and SPH can be accessed as SFRs.

Two instructions, PUSH and POP directly address the stack pointer. Subroutine calls (ACALL, ECALL, LCALL) and returns (ERET, RET, RETI) also use the stack pointer. To preserve the stack, do not use DR60 as a general-purpose register.

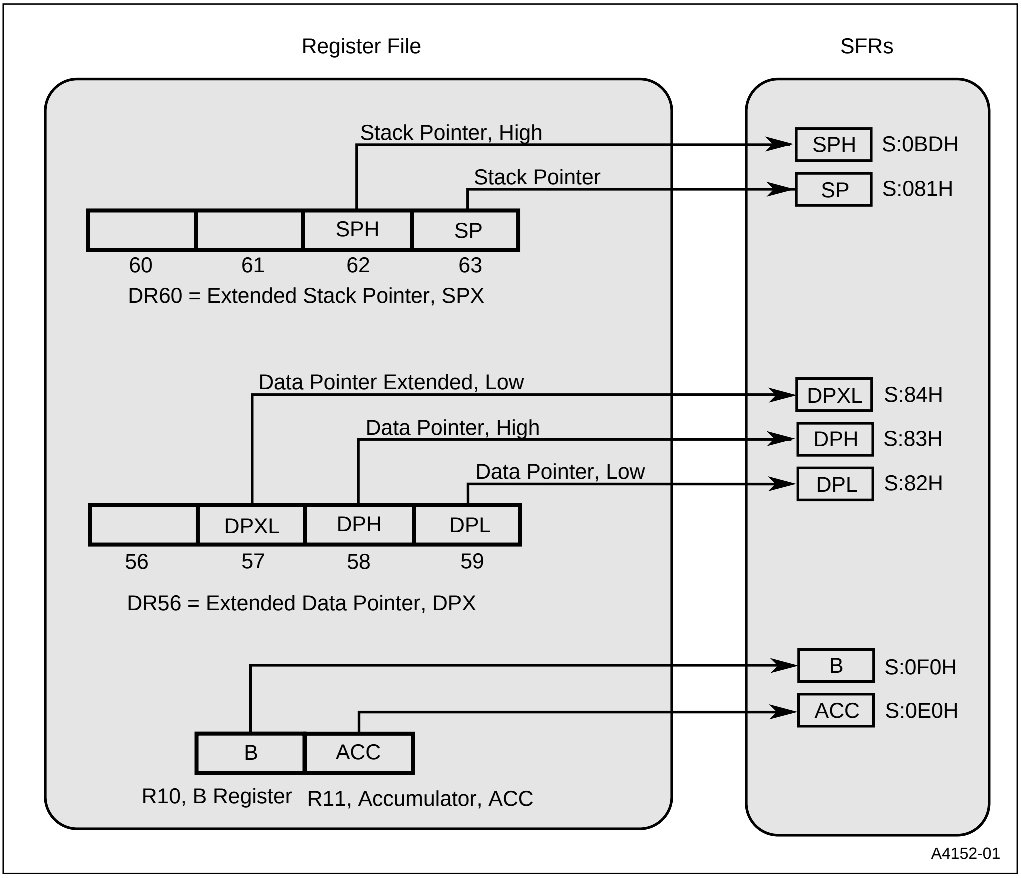

Register File
SFRs
Stack Pointer, High
Stack Pointer
SPH
SP
60
61
62
63
DR60 = Extended Stack Pointer, SPX
SPH S:0BDH
SP S:081H
Data Pointer Extended, Low
Data Pointer, High
Data Pointer, Low
DPXL
DPH
DPL
56
57
58
59
DR56 = Extended Data Pointer, DPX
DPXL S:84H
DPH S:83H
DPL S:82H
B
ACC
R10, B Register
R11, Accumulator, ACC
B S:0F0H
ACC S:0E0H
A4152-01

Figure 3-6. Dedicated Registers in the Register File and their Corresponding SFRs

Table 3-3. Dedicated Registers in the Register File and their Corresponding SFRs

Register File						SFRs	
Name			Mnemonic	Reg.	Location	Mnemonic	Address
Stack Pointer (SPX)	—		—	DR60	60	—	—
	—		—		61	—	—
	Stack Pointer, High		SPH		62	SPH	S:BDH
	Stack Pointer, Low		SP		63	SP	S:81H
Data Pointer (DPX)	Data Pointer, Extended High		—	DR56	56	—	—
	Data Pointer, Extended Low		DPXL		57	DPXL	S:84H
	DPTR	Data Pointer, High	DPH		58	DPH	S:83H
		Data Pointer, Low	DPL		59	DPL	S:82H
Accumulator (A Register)			A	R11	11	ACC	S:E0H
B Register			B	R10	10	B	S:F0H

3.4 SPECIAL FUNCTION REGISTERS (SFRS)

The special function registers (SFRs) reside in the their associated on-chip peripherals or in the core. Table 3-4 shows the SFR address space with the SFR mnemonics and reset values. SFR addresses are preceded by “S:” to differentiate them from addresses in the memory space. Unoccupied locations in the SFR space (the shaded locations in Table 3-4) are unimplemented, i.e., no register exists. If an instruction attempts to write to an unimplemented SFR location, the instruction executes, but nothing is actually written. If an unimplemented SFR location is read, it returns an unspecified value.

NOTE

SFRs may be accessed only as bytes; they may not be accessed as words or dwords.

Table 3-4. 8XC251SB SFR Map and Reset Values

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
F8		CH 00000000	CCAP0H xxxxxxxx	CCAP1H xxxxxxxx	CCAP2H xxxxxxxx	CCAP3H xxxxxxxx	CCAP4H xxxxxxxx		FF
F0	B 00000000								F7
E8		CL 00000000	CCAP0L xxxxxxxx	CCAP1L xxxxxxxx	CCAP2L xxxxxxxx	CCAP3L xxxxxxxx	CCAP4L xxxxxxxx		EF
E0	ACC 00000000								E7
D8	CCON 00x00000	CMOD 00xxx000	CCAPM0 x0000000	CCAPM1 x0000000	CCAPM2 x0000000	CCAPM3 x0000000	CCAPM4 x0000000		DF
D0	PSW 00000000	PSW1 00000000							D7
C8	T2CON 00000000	T2MOD xxxxxx00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			CF
C0									C7
B8	IPL0 x0000000	SADEN 00000000				SPH 00000000			BF
B0	P3 11111111							IPH0 x0000000	B7
A8	IE0 00000000	SADDR 00000000							AF
A0	P2 11111111						WDTRST xxxxxxxx		A7
98	SCON 00000000	SBUF xxxxxxxx							9F
90	P1 11111111								97
88	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000			8F
80	P0 11111111	SP 00000111	DPL 00000000	DPH 00000000	DPXL 00000001			PCON 00xx0000	87
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

NOTE: Shaded areas represent unimplemented SFR locations. Locations S:000H–S:07FH and S:100H–S:1FFH are also unimplemented.

The following tables list the mnemonics, names, and addresses of the SFRs:

Table 3-5 on page 3-14 — Core SFRs
Table 3-6 on page 3-14 — I/O Port SFRs
Table 3-7 on page 3-15 — Serial I/O SFRs
Table 3-8 on page 3-15 — Timer/Counter and Watchdog SFRs
Table 3-9 on page 3-15 — Programmable Counter Array (PCA) SFRs

Table 3-5. Core SFRs

Mnemonic	Name	Address
ACC†	Accumulator	S:E0H
B†	B register	S:F0H
PSW	Program Status Word	S:D0H
PSW†	Program Status Word 1	S:D1H
SP†	Stack Pointer – LSB of SPX	S:81H
SPH†	Stack Pointer High – MSB of SPX	S:BDH
DPTR†	Data Pointer (2 bytes)	—
DPL†	Low Byte of DPTR	S:82H
DPH†	High Byte of DPTR	S:83H
DPXL†	Data Pointer, Extended Low	S:84H
PCON	Power Control	S:87H
IE0	Interrupt Enable Control 0	S:A8H
IPH0	Interrupt Priority Control High 0	S:B7H
IPL0	Interrupt Priority Control Low 0	S:B8H

†These SFRs can also be accessed by their corresponding registers in the register file (see Table 3-3).

Table 3-6. I/O Port SFRs

Mnemonic	Name	Address
P0	Port 0	S:80H
P1	Port 1	S:90H
P2	Port 2	S:A0H
P3	Port 3	S:B0H

Table 3-7. Serial I/O SFRs

Mnemonic	Name	Address
SCON	Serial Control	S:98H
SBUF	Serial Data Buffer	S:99H
SADEN	Slave Address Mask	S:B9H
SADDR	Slave Address	S:A9H

Table 3-8. Timer/Counter and Watchdog Timer SFRs

Mnemonic	Name	Address
TL0	Timer/Counter 0 Low Byte	S:8AH
TH0	Timer/Counter 0 High Byte	S:8CH
TL1	Timer/Counter 1 Low Byte	S:8BH
TH1	Timer/Counter 1 High Byte	S:8DH
TL2	Timer/Counter 2 Low Byte	S:CCH
TH2	Timer/Counter 2 High Byte	S:CDH
TCON	Timer/Counter 0 and 1 Control	S:88H
TMOD	Timer/Counter 0 and 1 Mode Control	S:89H
T2CON	Timer/Counter 2 Control	S:C8H
T2MOD	Timer/Counter 2 Mode Control	S:C9H
RCAP2L	Timer 2 Reload/Capture Low Byte	S:CAH
RCAP2H	Timer 2 Reload/Capture High Byte	S:CBH
WDTRST	WatchDog Timer Reset	S:A6H

Table 3-9. Programmable Counter Array (PCA) SFRs

Mnemonic	Name	Address
CCON	PCA Timer/Counter Control	S:D8H
CMOD	PCA Timer/Counter Mode	S:D9H
CCAPM0	PCA Timer/Counter Mode 0	S:DAH
CCAPM1	PCA Timer/Counter Mode 1	S:DBH
CCAPM2	PCA Timer/Counter Mode 2	S:DCH
CCAPM3	PCA Timer/Counter Mode 3	S:DDH
CCAPM4	PCA Timer/Counter Mode 4	S:DEH

Table 3-9. Programmable Counter Array (PCA) SFRs (Continued)

Mnemonic	Name	Address
CL	PCA Timer/Counter Low Byte	S:E9H
CH	PCA Timer/Counter High Byte	S:F9H
CCAP0L	PCA Compare/Capture Module 0 Low Byte	S:EAH
CCAP1L	PCA Compare/Capture Module 1 Low Byte	S:EBH
CCAP2L	PCA Compare/Capture Module 2 Low Byte	S:ECH
CCAP3L	PCA Compare/Capture Module 3 Low Byte	S:EDH
CCAP4L	PCA Compare/Capture Module 4 Low Byte	S:EEH
CCAP0H	PCA Compare/Capture Module 0 High Byte	S:FAH
CCAP1H	PCA Compare/Capture Module 1 High Byte	S:FBH
CCAP2H	PCA Compare/Capture Module 2 High Byte	S:FCH
CCAP3H	PCA Compare/Capture Module 3 High Byte	S:FDH
CCAP4H	PCA Compare/Capture Module 4 High Byte	S:FEH

4

Programming

CHAPTER 4
PROGRAMMING

The instruction set for the MCS® 251 architecture is a superset of the instruction set for the MCS 51 architecture. This chapter describes the addressing modes and summarizes the instruction set, which is divided into data instructions, bit instructions, and control instructions. (Appendix A, “Instruction Set Reference” contains an opcode map and a detailed description of each instruction.) The program status words PSW and PSW1 are also described (page 4-17). The chapter begins with a discussion of the binary-mode and source-mode encodings of the instruction set.

NOTE

> The instruction execution times given in Appendix A are for code executing from on-chip code memory and for data that is read from and written to on-chip RAM. Execution times are increased by executing code from external memory, accessing peripheral SFRs, accessing data in external memory, using a wait state, or extending the ALE pulse.
>
> For some instructions, accessing the port SFRs, P*x*, *x* = 1–3, increases the execution time. These cases are noted individually in the tables in Appendix A.

4.1 BINARY MODE AND SOURCE MODE CONFIGURATIONS

Binary mode and *source mode* refer to two ways of assigning opcodes to the instruction set for the MCS 251 architecture. One of these modes must be selected when the chip is configured. Depending on the application, binary mode or source mode may produce more efficient code. This section describes the binary and source modes and provides some guidelines for selecting the mode for your application.

The MCS 251 architecture has two types of instructions:

- instructions that originate in the MCS 51 architecture
- instructions that are unique to the MCS 251 architecture

Figure 4-1 shows the opcode map for binary mode. On the left (areas I and II) is the opcode map for the instructions that originate in the MCS 51 architecture. Every opcode (00H–FFH), is used for an instruction except A5H, which is reserved. On the right (area III) is the opcode map for the instructions that are unique to the MCS 251 architecture. (Some of these opcodes are reserved for future instructions.) Note that the opcode values for areas II and III are identical (06H–FFH). To distinguish between the two areas, the opcodes in area III are given the prefix A5H. The area III opcodes are then A506H–A5FFH.

Figure 4-2 shows the opcode map for source mode. Areas II and III have switched places (compare Figure 4-1). The instructions that are unique to the MCS 251 architecture now have opcodes without the A5H prefix. The instructions from area II of the MCS 51 architecture use the escape prefix A5H.

To illustrate the difference between the binary-mode and source-mode opcodes, Table 4-1 shows the opcode assignments for three sample instructions.

Table 4-1. Examples of Opcodes in Binary and Source Modes

Instruction	Opcode	
	Binary Mode	Source Mode
DEC A	14H	14H
SUBB A,R4	9CH	A59CH
SUB R4,R4	A59CH	9CH

4.1.1 Selecting Binary Mode or Source Mode

If you have code that was written for an MCS 51 microcontroller and you want to run it unmodified on an MCS 51 microcontroller, choose binary mode. You can use the object code without reassembling the source code. You can also assemble the source code with an assembler for the MCS 251 architecture and have it produce object code that is binary-compatible with MCS 51 microcontrollers. The remainder of this section discusses the selection of binary mode or source mode for code that may contain instructions from both architectures.

An instruction with a prefixed opcode requires one more byte for code storage, and if an additional fetch is required for the extra byte, the execution time is increased by one state. This means that using fewer prefixed opcodes produces more efficient code.

If a program uses only instructions from the MCS 51 architecture, the binary-mode code is more efficient because it uses no prefixes. On the other hand, if a program uses many more new instructions than instructions from the MCS 51 architecture, source mode is likely to produce more efficient code. For a program where the choice is not clear, the better mode can be found by experimenting with a simulator.

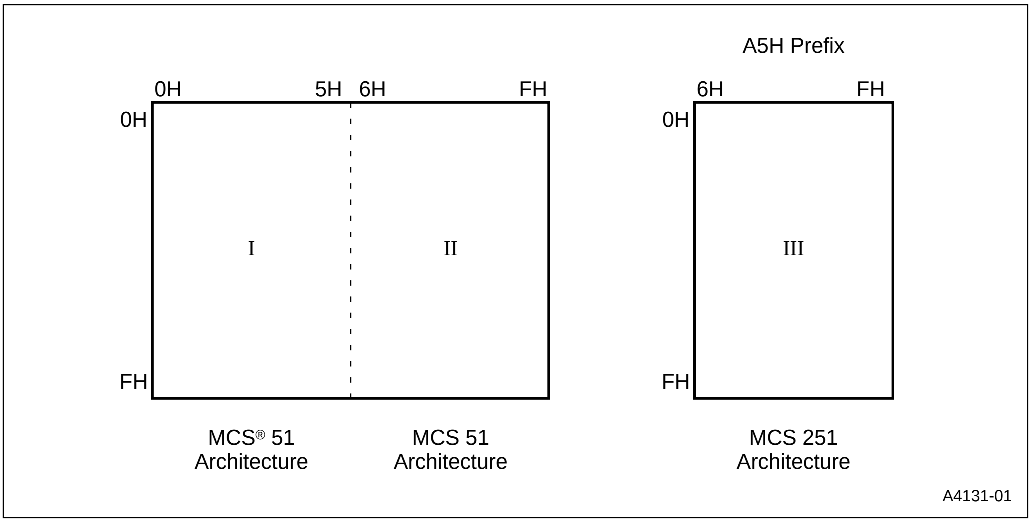

A5H Prefix
0H
5H
6H
FH
0H
FH
I
II
6H
FH
0H
FH
III
MCS® 51 Architecture
MCS 51 Architecture
MCS 251 Architecture
A4131-01

Figure 4-1. Binary Mode Opcode Map

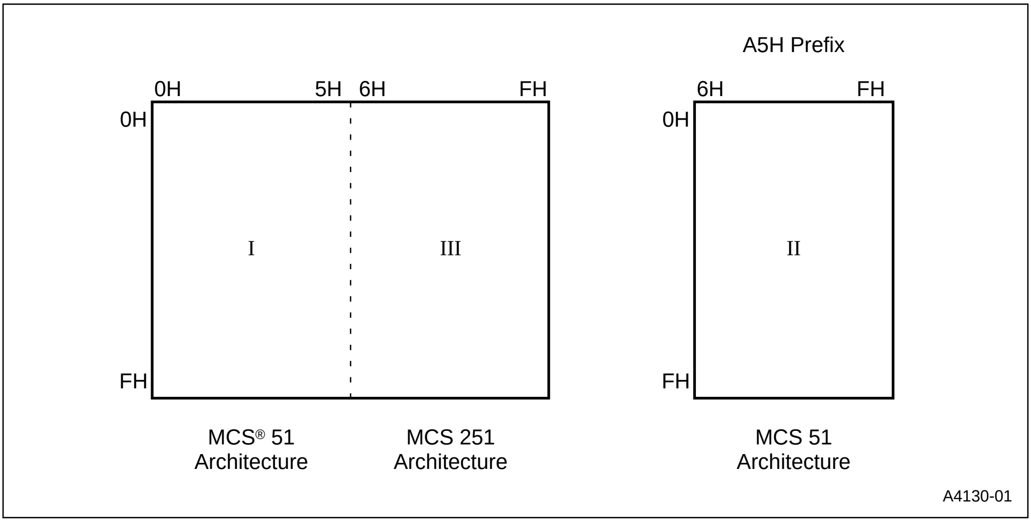

A5H Prefix
0H
5H
6H
FH
0H
FH
I
III
6H
FH
0H
FH
II
MCS® 51 Architecture
MCS 251 Architecture
MCS 51 Architecture
A4130-01

Figure 4-2. Source Mode Opcode Map

4.2 PROGRAMMING FEATURES OF THE MCS® 251 ARCHITECTURE

The instruction set for MCS 251 microcontrollers provides the user with new instructions that exploit the features of the architecture while maintaining compatibility with the instruction set for MCS 51 microcontrollers. Many of the new instructions can operate on either 8-bit, 16-bit, or 32-bit operands. (In comparison with 8-bit and 16-bit operands, 32-bit operands are accessed with fewer addressing modes.) This capability increases the ease and efficiency of programming MCS 251 microcontrollers in a high-level language such as C.

The instruction set is divided into “Data Instructions”(page 4-6), “Bit Instructions” (page 4-12), and “Control Instructions” (page 4-14). Data instructions process 8-bit, 16-bit, and 32-bit data; bit instructions manipulate bits; and control instructions manage program flow.

4.2.1 Data Types

Table 4-2 lists the data types that are addressed by the instruction set. A word or dword (double word) in memory can have its least significant byte at any address; alignment on two-byte or four-byte boundaries is not required.

Table 4-2. Data Types

Data Type	Number of Bits
Bit	1
Byte	8
Word	16
Dword (Double Word)	32

4.2.2 Register Notation

In register-addressing instructions , specific indices denote the registers that can be used in that instruction. For example, the instruction ADD A,Rn uses “Rn” to denote any one of R0, R1, ..., R7; i.e., the range of n is 0–7. The instruction ADD Rm,#data uses “Rm” to denote R0, R1, ..., R15; i.e., the range of m is 0–15. Table 4-3 summarizes the notation used for the register indices. When an instruction contains two registers of the same type (e.g., MOV Rmd,Rms) the first index “d” denotes “destination” and the second index “s” denotes “source.”

Table 4-3. Notation for Byte Registers, Word Registers, and Dword Registers

Register Type	Register Symbol	Destination Register	Source Register	Register Range
Byte	Ri	—	—	R0, R1
	Rn	—	—	R0–R7
	Rm	Rmd	Rms	R0–R15
Word	WRj	WRjd	WRjs	WR0, WR2, WR4, ..., WR30
Dword	DRk	DRkd	DRks	DR0, DR4, DR8, ..., DR28, DR56, DR60

4.2.3 Address Notation

In the MCS 251 architecture, memory addresses include a region number (00:, 01:, ..., FF:) (Figure 3-1 on page 3-1). SFR addresses have a prefix “S:” (S:000H–S:1FFH). The distinction between memory addresses and SFR addresses is necessary, because memory locations 00:0000H–00:01FFH and SFR locations S:000H–S:1FFH can both be directly addressed in an instruction.

Instructions in the MCS 51 architecture use 80H–FFH as addresses for both memory locations and SFRs, because memory locations are addressed only indirectly and SFR locations are addressed only directly. For compatibility, software tools for MCS 251 controllers recognize this notation for instructions in the MCS 51 architecture. No change is necessary in any code written for MCS 51 controllers.

For new instructions in the MCS 251 architecture, the memory region prefixes (00:, 01, ..., FF:) and the SFR prefix (S:) are required. Also, software tools for the MCS 251 architecture permit 00: to be used for memory addresses 00H–FFH and permit the prefix S: to be used for SFR addresses in instructions in the MCS 51 architecture.

4.2.4 Addressing Modes

The MCS 251 architecture supports the following addressing modes:

- register addressing: The instruction specifies the register that contains the operand.
- immediate addressing: The instruction contains the operand.
- direct addressing: The instruction contains the operand address.
- indirect addressing: The instruction specifies the register that contains the operand address.
- displacement addressing: The instruction specifies a register and an offset. The operand address is the sum of the register contents (the base address) and the offset.

- relative addressing: The instruction contains the signed offset from the next instruction to the target address (the address for transfer of control, e.g., the jump address).
- bit addressing: The instruction contains the bit address.

More detailed descriptions of the addressing modes are given in “Data Addressing Modes” on page 4-6, “Bit Addressing” on page 4-12, and “Addressing Modes for Control Instructions” on page 4-14.

4.3 DATA INSTRUCTIONS

Data instructions consist of arithmetic, logical, and data-transfer instructions for 8-bit, 16-bit, and 32-bit data. This section describes the data addressing modes and the set of data instructions.

4.3.1 Data Addressing Modes

This section describes the data-addressing modes, which are summarized in two tables: Table 4-5 for the instructions that are native to the MCS 51 architecture, and Table 4-5 for the new data instructions in the MCS 251 architecture.

NOTE

References to registers R0–R7, WR0–WR6, DR0, and DR2 always refer to the register bank that is currently selected by the PSW and PSW1 registers (see “Program Status Words” on page 4-17). Registers in all banks (active and inactive) can be accessed as memory locations in the range 00H–1FH.

Table 4-4. Addressing Modes for Data Instructions in the MCS® 51 Architecture

Mode	Address Range of Operand	Assembly Language Reference	Comments
Register	00H–1FH	R0–R7 (Bank selected by PSW)	
Immediate	Operand in Instruction	#data = #00H–#FFH	
Direct	00H–7FH	dir8 = 00H–7FH	On-chip RAM
	SFRs	dir8 = 80H–FFH or SFR mnemonic.	SFR address
Indirect	00H–FFH	@R0, @R1	Accesses on-chip RAM or the lowest 256 bytes of external data memory (MOVX).
	0000H–FFFFH	@DPTR, @A+DPTR	Accesses external data memory (MOVX).
	0000H–FFFFH	@A+DPTR, @A+PC	Accesses region FF: of code memory (MOVC).

Table 4-5. Addressing Modes for Data Instructions in the MCS® 251 Architecture

Mode	Address Range of Operand	Assembly Language Notation	Comments
Register	00:0000H–00:001FH (R0–R7, WR0–WR3, DR0, DR2) (1)	R0–R15, WR0–WR30, DR0–DR28, DR56, DR60	R0–R7, WR0–WR6, DR0, and DR2 are in the register bank currently selected by the PSW and PSW1.
Immediate, 2 bits	N.A. (Operand is in the instruction)	#short = 1, 2, or 4	Used only in increment and decrement instructions.
Immediate, 8 bits	N.A. (Operand is in the instruction)	#data8 = #00H–#FFH	
Immediate, 16 bits	N.A. (Operand is in the instruction)	#data16 = #0000H–#FFFFH	
Direct, 8 address bits	00:0000H–00:007FH	dir8 = 00:0000H–00:007FH	On-chip RAM
	SFRs	dir8 = S:080H–S:1FFH (2) or SFR mnemonic	SFR address
Direct, 16 address bits	00:0000H–00:FFFFH	dir16 = 00:0000H–00:FFFFH	
Indirect, 16 address bits	00:0000H–00:FFFFH	@WR0–@WR30	
Indirect, 24 address bits	00:0000H–FF:FFFFH	@DR0–@DR30, @DR56, @DR60	Upper 8 bits of DRk must be 00H.
Displacement, 16 address bits	00:0000H–00:FFFFH	@WRj + dis16 = @WR0 + 0H through @WR30 + FFFFH	Offset is signed; address wraps around in region 00:.
Displacement, 24 address bits	00:0000H–FF:FFFFH	@DRk + dis24 = @DR0 + 0H–through @DR28 + FFFFH, @DR56 + (0H–FFFFH), @DR60 + (0H–FFFFH)	Offset is signed, upper 8 bits of DRk must be 00H.

NOTES:

1. These registers are accessible in the memory space as well as in the register file (see “The 8XC251SB Register File” on page 3-8).
2. The MCS 251 architecture supports SFRs in locations S:000H–S:1FFH; however, in the 8XC251SB, all SFRs are in the range S:080H–S:0FFH.

NOTE

Instructions from the MCS 51 architecture access external memory through the region of memory specified by byte DPXL in the extended data pointer register, DPX (DR56). Following reset, DPXL contains 01H, which maps the external memory to region 01:. You can specify a different region by writing to DR56 or the DPXL SFR. (See “Dedicated Registers” on page 3-10.).

4.3.1.1 Register Addressing

Both architectures address registers directly.

- MCS 251 architecture. In the register addressing mode, the operand(s) in a data instruction are in byte registers (R0–R15), word registers (WR0, WR2, ..., WR30), or dword registers (DR0, DR4, ..., DR28, DR56, DR60).
- MCS 51 architecture. Instructions address registers R0–R7 only.

4.3.1.2 Immediate

Both architectures use immediate addressing.

- MCS 251 architecture. In the immediate addressing mode, the instruction contains the data operand itself. Byte operations use 8-bit immediate data (#data); word operations use 16-bit immediate data (#data16). Dword operations use 16-bit immediate data in the lower word and either zeros in the upper word (denoted by #0data16) or ones in the upper word (denoted by #1data16). MOV instructions that place 16-bit immediate data into a dword register (DRk), place the data either into the upper word while leaving the lower word unchanged, or into the lower word with a sign extension or a zero extension.

  The increment and decrement instructions contain immediate data (#short = 1, 2, or 4), which specifies the amount of the increment/decrement.
- MCS 51 architecture. Instructions use only 8-bit immediate data (#data).

4.3.1.3 Direct

- MCS 251 architecture. In the direct addressing mode, the instruction contains the address of the data operand. The 8-bit direct mode addresses on-chip RAM (dir8 = 00:0000H–00:007FH) as both bytes and words, and addresses the SFRs (dir8 = S:080H–S:1FFH) as bytes only. (See the note below Table 4-5 on page 4-7 regarding SFRs in the MCS 251 architecture.) The 16-bit direct mode addresses both bytes and words in memory (dir16 = 00:0000H–00:FFFFH).
- MCS 51 architecture. The 8-bit direct mode addresses 256 bytes of on-chip RAM (dir8 = 00H–7FH) as bytes only and the SFRs (dir8 = 80H–FFH) as bytes only.

4.3.1.4 Indirect

In arithmetic and logical instructions that use indirect addressing, the source operand is always a byte, and the destination is either the accumulator or a byte register (R0–R15). The source address is a byte, word, or dword. The two architectures do indirect addressing via different registers:

- MCS 251 architecture. Memory is indirectly addressed via word and dword registers:
  - Word register (@WRj, j = 0, 2, 4, ..., 30). The 16-bit address in WRj can access locations 00:0000H–00:FFFFH.
  - Dword register (@DRk, k = 0, 4, 8, ..., 28, 56, and 60). The 24 least significant bits can access the entire 16-Mbyte address space. The upper eight bits of DRk must be 0. (If you use DR60 as a general data pointer, be aware that DR60 is the extended stack pointer register DPX.)
- MCS 51 architecture. Instructions use indirect addressing to access on-chip RAM, code memory, and external data RAM. (See the Note on page 4-7 regarding the region of external data RAM that is addressed by instructions in the MCS 51 architecture.)
  - Byte register (@Ri, i = 1, 2). Registers R0 and R1 indirectly address on-chip memory locations 00H–FFH and the lowest 256 bytes of external data RAM.
  - 16-bit data pointer (@DPTR or @A+DPTR). The MOVC and MOVX instructions use these indirect modes to access code memory and external data RAM.
  - 16-bit program counter (@A+PC). The MOVC instruction uses this indirect mode to access code memory.

4.3.1.5 Displacement

Several move instructions use displacement addressing to move bytes or words from a source to a destination. Sixteen-bit displacement addressing (@WRj+dis16) accesses indirectly the lowest 64 Kbytes in memory. The base address can be in any word register WRj. The instruction contains a 16-bit signed offset which is added to the base address. Only the lowest 16 bits of the sum are used to compute the operand address. If the sum of the base address and a positive offset exceeds FFFFH, the computed address wraps around within region 00: (e.g. F000H + 2005H becomes 1005H). Similarly, if the sum of the base address and a negative offset is less than zero, the computed address wraps around the top of region 00: (e.g., 2005H + F000H becomes 1005H).

Twenty-four-bit displacement addressing (@DRk+dis24) accesses indirectly the entire 16-Mbyte address space. The base address must be in DR0, DR4, ..., DR24, DR28, DR56, or DR60. The upper byte in the dword register must be zero. The instruction contains a 16-bit signed offset which is added to the base address.

4.3.2 Arithmetic Instructions

The set of arithmetic instructions is greatly expanded in the MCS 251 architecture. The ADD and SUB instructions (Table A-19 on page A-14) operate on byte and word data that is accessed in several ways:

- as the contents of the accumulator, a byte register (Rn), or a word register (WRj)
- in the instruction itself (immediate data)
- in memory via direct or indirect addressing

The ADDC and SUBB instructions (Table A-19 on page A-14) are the same as those for MCS 51 microcontrollers.

The CMP (compare) instruction (Table A-20 on page A-15) calculates the difference of two bytes or words and then writes to flags CY, OV, AC, N, and Z in the PSW and PSW1 registers. The difference is not stored. The operands can be addressed in a variety of modes. The most frequent use of CMP is to compare data or addresses preceding a conditional jump instruction.

Table A-21 on page A-16 lists the INC (increment) and DEC (decrement) instructions. The instructions for MCS 51 microcontrollers are supplemented by instructions that can address byte, word, and dword registers and increment or decrement them by 1, 2, or 4 (denoted by #short). These instructions are supplied primarily for register-based address pointers and loop counters.

The MCS 251 architecture provides the MUL (multiply) and DIV (divide) instructions for unsigned 8-bit and 16-bit data (Table A-22 on page A-16). Signed multiply and divide are left for the user to manage through a conversion process. The following operations are implemented:

- eight-bit multiplication: 8 bits × 8 bits → 16 bits
- sixteen-bit multiplication: 16 bits × 16 bits → 32 bits
- eight-bit division: 8 bits ÷ 8 bits → 16 bits (8-bit quotient, 8-bit remainder)
- sixteen-bit division: 16 bits ÷ 16 bits → 32 bits (16-bit quotient, 16-bit remainder)

These instructions operate on pairs of byte registers (Rmd,Rms), word registers (WRjd,WRjs), or the accumulator and B register (A,B). For 8-bit register multiplies, the result is stored in the word register that contains the first operand register. For example, the product from an instruction MUL R3,R8 is stored in WR2. Similarly, for 16-bit multiplies, the result is stored in the dword register that contains the first operand register. For example, the product from the instruction MUL WR6,WR18 is stored in DR4.

For 8-bit divides, the operands are byte registers. The result is stored in the word register that contains the first operand register. The quotient is stored in the lower byte, and the remainder is stored in the higher byte. A 16-bit divide is similar. The first operand is a word register, and the result is stored in the double word register that contains that word register. If the second operand (the divisor) is zero, the overflow flag (OV) is set and the other bits in PSW and PSW1 are meaningless.

4.3.3 Logical Instructions

The MCS 251 architecture provides a set of instructions that perform logical operations. The ANL, ORL, and XRL (logical AND, logical OR, and logical exclusive OR) instructions operate on bytes and words that are accessed via several addressing modes (Table A-23 on page A-17). A byte register, word register, or the accumulator can be logically combined with a register, immediate data, or data that is addressed directly or indirectly. These instructions affect the Z and N flags.

In addition to the CLR (clear), CPL (complement), SWAP (swap), and four rotate instructions that operate on the accumulator, MCS 251 microcontrollers have three shift commands for byte and word registers:

- SLL (Shift Left Logical) shifts the register one bit left and replaces the LSB with 0.
- SRL (Shift Right Logical) shifts the register one bit right and replaces the MSB with 0.
- SRA (Shift Right Arithmetic) shifts the register one bit right; the MSB is unchanged.

4.3.4 Data Transfer Instructions

Data transfer instructions copy data from one register or memory location to another. These instructions include the move instructions (Table A-24 on page A-19) and the exchange, push, and pop instructions (Table A-24 on page A-19). Instructions that move only a single bit are listed with the other bit instructions in Table A-26 on page A-23.

MOV (Move) is the most versatile instruction, and its addressing modes are expanded in the MCS 251 architecture. MOV can transfer a byte, word, or dword between any two registers or between a register and any location in the address space.

The MOVX (Move External) instruction moves a byte from external memory to the accumulator or from the accumulator to memory. The external memory is in the region specified by DPXL, whose reset value is 01H. (See “Dedicated Registers” on page 3-10.)

The MOVC (Move Code) instruction moves a byte from code memory (region FF:) to the accumulator.

MOVS (Move with Sign Extension) and MOVZ (Move with Zero Extension) move the contents of an 8-bit register to the lower byte of a 16-bit register. The upper byte is filled with the sign bit (MOVS) or zeros (MOVZ). The MOVH (Move to High Word) instruction places 16-bit immediate data into the high word of a dword register.

The XCH (Exchange) instruction interchanges the contents of the accumulator with a register or memory location. The XCHD (Exchange Digit) instruction interchanges the lower nibble of the accumulator with the lower nibble of a byte in on-chip RAM. XCHD is useful for BCD (binary coded decimal) operations.

The PUSH and POP instructions facilitate storing information (PUSH) and then retrieving it (POP) in reverse order. Push can push a byte, a word, or a dword onto the stack, using the immediate, direct, or register addressing modes. POP can pop a byte or a word from the stack to a register or to memory.

4.4 BIT INSTRUCTIONS

A bit instruction addresses a specific bit in a memory location or SFR. There are four categories of bit instructions:

- SETB (Set Bit), CLR (Clear Bit), CPL (Complement Bit). These instructions can set, clear or complement any addressable bit.
- ANL (And Logical), ANL/ (And Logical Complement), ORL (OR Logical), ORL/ (Or Logical Complement). These instructions allow ANDing and ORing of any addressable bit or its complement with the CY flag.
- MOV (Move) instructions transfer any addressable bit to the carry (CY) bit or vice versa.
- Bit-conditional jump instructions execute a jump if the bit has a specified state. The bit-conditional jump instructions are classified with the control instructions and are described in “Conditional Jumps” on page 4-15.

4.4.1 Bit Addressing

The bits that can be individually addressed are in the on-chip RAM and the SFRs (Table 4-6). The bit instructions that are unique to the MCS 251 architecture can address a wider range of bits than the instructions from the MCS 51 architecture.

Table 4-6. Bit-addressable Locations

Architecture	Bit-addressable Locations	
	On-chip RAM	SFRs
MCS 251® Architecture	20H–7FH	All defined SFRs
MCS 51 Architecture	20H–2FH	SFRs with addresses ending in 0H or 8H: 80H, 88H, 90H, 98H, ..., F8H

There are some differences in the way the instructions from the two architectures address bits. In the MCS 51 architecture, a bit (denoted by bit51) can be specified in terms of its location within a certain register, or it can be specified by a bit address in the range 00H–7FH. The MCS 251 architecture does not have bit addresses as such. A bit can be addressed by name or by its location within a certain register, but not by a bit address.

Table 4-7 illustrates bit addressing in the two architectures by using two sample bits:

- RAMBIT is bit 5 in RAMREG, which is location 23H. (“RAMBIT” and “RAMREG” are assumed to be defined in user code.)
- IT1 is bit 2 in TCON, which is an SFR at location 88H.

Table 4-7. Addressing Two Sample Bits

Location	Addressing Mode	MCS® 51 Architecture	MCS 251 Architecture
On-chip RAM	Register Name	RAMREG.5	RAMREG.5
	Register Address	23H.5	23H.5
	Bit Name	RAMBIT	RAMBIT
	Bit Address	1DH	NA
SFR	Register Name	TCON.2	TCON.2
	Register Address	88.2H	S:88.2H
	Bit Name	IT1	IT1
	Bit Address	8A	NA

Table 4-8 lists the addressing modes for bit instructions, and Table A-26 on page A-23 summarizes the bit instructions. “bit” denotes a bit that is addressed by a new instruction in the MCS 251 architecture, and “bit51” denotes a bit that is addressed by an instruction in the MCS 51 architecture.

Table 4-8. Addressing Modes for Bit Instructions

Architecture	Variants	Bit Address	Memory/SFR Address	Comments
MCS® 251 Architecture (bit)	Memory	NA	20H.0–7FH.7	
	SFR	NA	All defined SFRs	
MCS 51 Architecture (bit51)	Memory	00H–7FH	20H.0–7FH.7	
	SFR	80H–F8H	XXH.0–XXH.7, where XX = 80, 88, 90, 98, ..., F0, F8.	SFRs are not defined at all bit-addressable locations.

4.5 CONTROL INSTRUCTIONS

Control instructions—instructions that change program flow—include calls, returns, and conditional and unconditional jumps (see Table A-27 on page A-24). Instead of executing the next instruction in the queue, the processor executes a target instruction.

4.5.1 Addressing Modes for Control Instructions

A control instruction provides the address of a target instruction. The instruction can specify the target address implicitly, as in a return from a subroutine, or explicitly, in the form of a relative, direct, or indirect address:

- Relative addressing: The control instruction provides the target address as an 8-bit signed offset (rel) from the address of the next instruction.
- Direct addressing: The control instruction provides a target address, which can have 11 bits (addr11), 16 bits (addr16), or 24 bits (addr24). The target address is written to the PC.
  - addr11: Only the lower 11 bits of the PC are changed; i.e., the target address must be in the current 2-Kbyte block (the 2-Kbyte block that includes the first byte of the next instruction).
  - addr16: Only the lower 16 bits of the PC are changed; i.e., the target address must be in the current 64-Kbyte region (the 64-Kbyte region that includes the first byte of the next instruction).
  - addr24: The target address can be anywhere in the 16-Mbyte address space.
- Indirect addressing: There are two types of indirect addressing for control instructions:
  - For the instructions LCALL @WRj and LJMP @WRj, the target address is in the current 64-Kbyte region. The 16-bit address in WRj is placed in the lower 16 bits of the PC. The upper eight bits of the PC remain unchanged from the address of the next instruction.
  - For the instruction JMP @A+DPTR, the sum of the accumulator and DPTR is placed in the lower 16 bits of the PC, and the upper eight bits of the PC are FF:, which restricts the target address to the code memory space of the MCS 51 architecture.

Table 4-9 lists the addressing modes for the control instructions.

Table 4-9. Addressing Modes for Control Instructions

Description	Address Bits Provided	Address Range
Relative, 8-bit relative address (rel)	8	-128 to +127 from first byte of next instruction
Direct, 11-bit target address (addr11)	11	Current 2 Kbytes
Direct, 16-bit target address (addr16)	16	Current 64 Kbytes
Direct, 24-bit target address (addr24)†	24	00:0000H–FF:FFFFH
Indirect (@WRj)†	16	Current 64 Kbytes
Indirect (@A+DPTR)	16	64-Kbyte region specified by DPXL (reset value = 01H)

†These modes are not used by instructions in the MCS® 51 architecture.

4.5.2 Conditional Jumps

The MCS 251 architecture supports bit-conditional jumps, compare-conditional jumps, and jumps based on the value of the accumulator. A bit-conditional jump is based on the state of a bit. In a compare-conditional jump, the jump is based on a comparison of two operands. All conditional jumps are relative, and the target address (rel) must be in the current 256-byte block of code.

The instruction set includes three kinds of bit-conditional jumps:

- JB (Jump on Bit): Jump if the bit is set.
- JNB (Jump on Not Bit): Jump if the bit is clear.
- JBC (Jump on Bit then Clear it): Jump if the bit is set; then clear it.

“Bit Addressing” on page 4-12 describes the bit addressing used in these instructions.

Compare-conditional jumps test a condition resulting from a compare (CMP) instruction that is assumed to precede the jump instruction. The jump instruction examines the PSW and PSW1 registers and interprets their flags as though they were set or cleared by a compare (CMP) instruction. Actually, the state of each flag is determined by the last instruction that could have affected that flag.

The condition flags are used to test one of the following six relations between the operands:

- equal (=), not equal (≠)
- greater than (>), less than (<)
- greater than or equal (≥), less than or equal (≤)

For each relation there are two instructions, one for signed operands and one for unsigned operands (Table 4-10).

Table 4-10. Compare-conditional Jump Instructions

Operand Type	Relation					
	=	≠	>	<	≥	≤
Unsigned	JE	JNE	JG	JL	JGE	JLE
Signed			JSG	JSL	JSGE	JSLE

4.5.3 Unconditional Jumps

There are five unconditional jumps. NOP and SJMP jump to addresses relative to the program counter. AJMP, LJMP, and EJMP jump to direct or indirect addresses.

- NOP (No Operation) is an unconditional jump to the next instruction.
- SJMP (Short Jump) jumps to any instruction within -128 to 127 of the next instruction.
- AJMP (Absolute Jump) changes the lowest 11 bits of the PC to jump anywhere within the current 2-Kbyte block of memory. The address can be direct or indirect.
- LJMP (Long Jump) changes the lowest 16 bits of the PC to jump anywhere within the current 64-Kbyte region.
- EJMP (Extended Jump) changes all 24 bits of the PC to jump anywhere in the 16-Mbyte address space. The address can be direct or indirect.

4.5.4 Calls and Returns

The MCS 251 architecture provides relative, direct, and indirect calls and returns.

ACALL (Absolute Call) pushes the lower 16 bits of the next instruction address onto the stack and then changes the lower 11 bits of the PC to the 11-bit address specified by the instruction. The call is to an address that is in the same 2-Kbyte block of memory as the address of the next instruction.

LCALL (Long Call) pushes the lower 16 bits of the next-instruction address onto the stack and then changes the lower 16 bits of the PC to the 16-bit address specified by the instruction. The call is to an address in the same 64-Kbyte block of memory as the address of the next instruction.

ECALL (Extended Call) pushes the 24 bits of the next instruction address onto the stack and then changes the 24 bits of the PC to the 24-bit address specified by the instruction. The call is to an address anywhere in the 16-Mbyte memory space.

RET (Return) pops the top two bytes from the stack to return to the instruction following a subroutine call. The return address must be in the same 64-Kbyte region.

ERET (Extended Return) pops the top three bytes from the stack to return to the address following a subroutine call. The return address can be anywhere in the 16-Mbyte address space.

RETI (Return from Interrupt) provides a return from an interrupt service routine. The operation of RETI depends on the INTR configuration bit in the CONFIG1 register:

- For INTR = 0, an interrupt causes the two lower bytes of the PC to be pushed onto the stack. The RETI instruction pops these two bytes and uses them as the 16-bit return address in region FF:. RETI also restores the interrupt logic to accept additional interrupts at the same priority level as the one just processed.
- For INTR = 1, an interrupt causes four bytes to be pushed onto the stack: the three bytes of the PC plus the PSW1 register. The RETI instruction pops these four bytes and then returns to the specified 24-bit address, which can be anywhere in the 16-Mbyte address space. RETI also clears the interrupt request line.

The TRAP instruction is useful for the development of emulations of an MCS 251 microcontroller.

4.6 PROGRAM STATUS WORDS

The Program Status Word (PSW) register and the Program Status Word 1 (PSW1) register contain four types of bits (Figure 4-3 on page 4-19 and Figure 4-4 on page 4-20):

- CY, AC, OV, N, and Z are flags set by hardware to indicate the result of an operation.
- The P bit indicates the parity of the accumulator.
- Bits RS0 and RS1 are programmed by software to select the active register bank for registers R0–R7.
- F0 and UD are available to the user as general-purpose flags.

The PSW and PSW1 registers are read/write registers; however, the parity bit in the PSW is not affected by a write. Individual bits can be addressed with the bit instructions (“Bit Instructions” on page 4-12). The PSW and PSW1 bits are used implicitly in the conditional jump instructions (“Conditional Jumps” on page 4-15).

The PSW register is identical to the PSW register in MCS 51 microcontrollers. The PSW1 register exists only in MCS 251 microcontrollers. Bits CY, AC, RS0, RS1, and OV in PSW1 are identical to the corresponding bits in PSW, i.e., the same bit can be accessed in either register. Table 4-11 lists the instructions that affect the CY, AC, OV, N, and Z bits.

Table 4-11. The Effects of Instructions on the PSW and PSW1 Flags

Instruction Type	Instruction	Flags Affected (1)				
		CY	OV	AC (2)	N	Z
Arithmetic	ADD, ADDC, SUB, SUBB, CMP	X	X	X	X	X
	INC, DEC				X	X
	MUL, DIV (3)	0	X		X	X
	DA	X			X	X
Logical	ANL, ORL, XRL, CLR A, CPL A, RL, RR, SWAP				X	X
	RLC, RRC, SRL, SLL, SRA (4)	X			X	X
Program Control	CJNE	X			X	X
	DJNE				X	X

NOTES:

1. X = the flag can be affected by the instruction.
   0 = the flag is cleared by the instruction.
2. The AC flag is affected only by operations on 8-bit operands.
3. If the divisor is zero, the OV flag is set, and the other bits are meaningless.
4. For SRL, SLL, and SRA instructions, the last bit shifted out is stored in the CY bit.

PSW

Address: S:D0H
Reset State: 0000 0000B

7							0
CY	AC	F0	RS1	RS0	OV	UD	P

Bit Number	Bit Mnemonic	Function
7	CY	Carry Flag: The carry flag is set by an addition instruction (ADD, ADDC) if there is a carry out of the MSB. It is set by a subtraction (SUB, SUBB) or compare (CMP) if a borrow is needed for the MSB. The carry flag is also affected by some rotate and shift instructions, logical bit instructions and bit move instructions, and the multiply (MUL) and decimal adjust (DA) instructions (see Table 4-11 on page 4-18).
6	AC	Auxiliary Carry Flag: The auxiliary carry flag is affected only by instructions that address 8-bit operands. The AC flag is set if an arithmetic instruction with an 8-bit operand produces a carry out of bit 3 (from addition) or a borrow into bit 3 (from subtraction). Otherwise it is cleared. This flag is useful for BCD arithmetic (see Table 4-11 on page 4-18).
5	F0	Flag 0: This general-purpose flag is available to the user.
4:3	RS1:0	Register Bank Select Bits 1 and 0: These bits select the memory locations that comprise the active bank of the register file (registers R0–R7). **RS1 RS0 Bank Address** 0 0 0 00H–07H 0 1 1 08H–0FH 1 0 2 10H–17H 1 1 3 18H–1FH
2	OV	Overflow Flag: This bit is set if an addition or subtraction of signed variables results in an overflow error (i.e., if the magnitude of the sum or difference is too great for the seven LSBs in 2's-complement representation). The overflow flag is also set if a multiplication product overflows one byte or if a division by zero is attempted.
1	UD	User-definable Flag: This general-purpose flag is available to the user.
0	P	Parity Bit: This bit indicates the parity of the accumulator. It is set if an odd number of bits in the accumulator are set. Otherwise, it is cleared. Not all instructions update the parity bit.

Figure 4-3. Program Status Word Register

PSW1

Address: S:D1H
Reset State: 0000 0000B

7							0
CY	AC	N	RS1	RS0	OV	Z	—

Bit Number	Bit Mnemonic	Function
7	CY	Carry Flag: Identical to the CY bit in the PSW register (Figure 4-3 on page 4-19.)
6	AC	Auxiliary Carry Flag: Identical to the AC bit in the PSW register (Figure 4-3 on page 4-19.)
5	N	Negative Flag: This bit is set if the result of the last logical or arithmetic operation was negative. Otherwise it is cleared.
4–3	RS1:0	Register Bank Select Bits 0 and 1: Identical to the RS1:0 bits in the PSW register (Figure 4-3 on page 4-19).
2	OV	Overflow Flag: Identical to the OV bit in the PSW register (Figure 4-3 on page 4-19.)
1	Z	Zero Flag: This flag is set if the result of the last logical or arithmetic operation is zero. Otherwise it is cleared.
0	—	Reserved: The value read from this bit is indeterminate. Do not write a “1” to this bit.

Figure 4-4. Program Status Word 1 Register

5

Interrupt System

CHAPTER 5
INTERRUPT SYSTEM

5.1 OVERVIEW

The 8XC251SB, like other control-oriented computer architectures, employs a program interrupt method. This operation branches to a subroutine and performs some service in response to the interrupt. When the subroutine completes, execution resumes at the point where the interrupt occurred. Interrupts may occur as a result of internal 8XC251SB activity (e.g., timer overflow) or at the initiation of electrical signals external to the microcontroller (e.g., serial port communication). In all cases, interrupt operation is programmed by the system designer, who determines priority of interrupt service relative to normal code execution and other interrupt service routines. Seven of the eight interrupts are enabled or disabled by the system designer and may be manipulated dynamically.

A typical interrupt event chain occurs as follows. An internal or external device initiates an interrupt-request signal. This signal, connected to an input pin (see Table 5-1, Interrupt System Pin Signals) and periodically sampled by the 8XC251SB, latches the event into a flag buffer. The priority of the flag (see Table 5-2, Interrupt System Special Function Registers) is compared to the priority of other interrupts by the interrupt handler. A high priority causes the handler to set an interrupt flag. This signals the instruction execution unit to execute a context switch. This context switch breaks the current flow of instruction sequences. The execution unit completes the current instruction prior to a save of the program counter (PC) and reloads the PC with the start address of a software service routine. The software service routine executes assigned tasks and as a final activity performs a RETI (return from interrupt) instruction. This instruction signals completion of the interrupt, resets the interrupt-in-progress priority, and reloads the program counter. Program operation then continues from the original point of interruption.

Table 5-1. Interrupt System Pin Signals

Signal Name	Type	Description	Multiplexed With
INT1:0#	I	**External Interrupts 0 and 1**. These inputs set bits IE1:0 in the TCON register. If bits IT1:0 in the TCON register are set, bits IE1:0 are controlled by a negative-edge trigger on INT1#/INT0#. If bits INT1:0# are clear, bits IE1:0 are controlled by a low level trigger on INT1:0#.	P3.3:2

NOTE: Other pin signals are defined in their respective chapters and in Appendix B, “Signal Descriptions.”

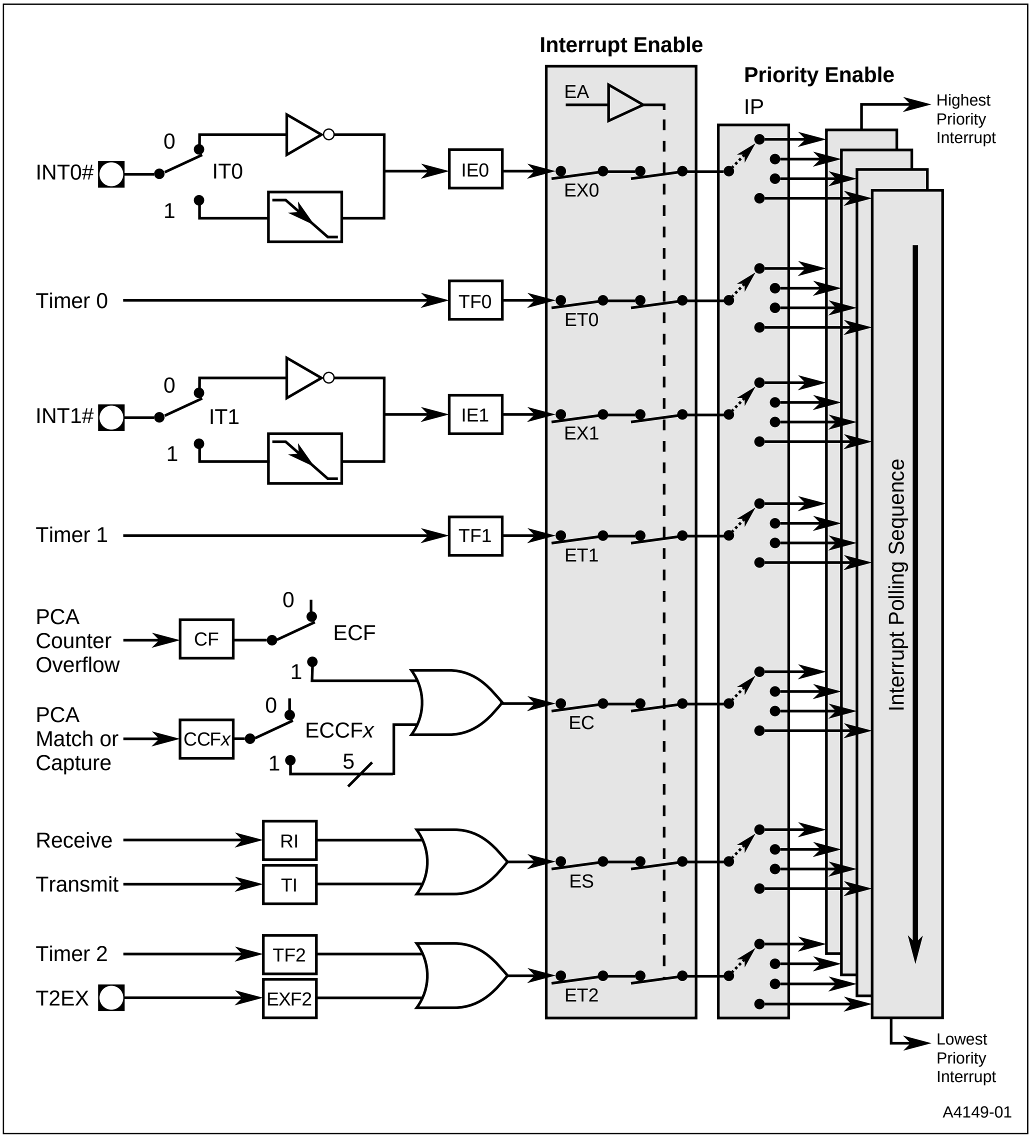

Interrupt Enable
Priority Enable
IP
EA
Highest
Priority
Interrupt
INT0#
0
IT0
1
IE0
EX0
Timer 0
TF0
ET0
INT1#
0
IT1
1
IE1
EX1
Timer 1
TF1
ET1
PCA
Counter
Overflow
CF
0
ECF
1
PCA
Match or
Capture
CCFx
0
ECCFx
1
5
EC
Receive
RI
Transmit
TI
ES
Timer 2
TF2
T2EX
EXF2
ET2
Interrupt Polling Sequence
Lowest
Priority
Interrupt
A4149-01

Figure 5-1. Interrupt Control System

Table 5-2. Interrupt System Special Function Registers

Mnemonic	Description	Address
IE0	**Interrupt Enable Register**. Used to enable and disable programmable interrupts. The reset value of this register is zero (interrupts disabled).	S:A8H
IPL0	**Interrupt Priority Low Register**. Establishes relative four-level priority for programmable interrupts. Used in conjunction with IPH0.	S:B8H
IPH0	**Interrupt Priority High Register**. Establishes relative four-level priority for programmable interrupts. Used in conjunction with IPL0.	S:B7H

NOTE: Other Special Function Registers are described in their respective chapters.

5.2 8XC251SB INTERRUPT SOURCES

Figure 5-1 on page 5-2 illustrates the interrupt control system. The 8XC251SB has eight interrupt sources; seven maskable sources and the TRAP instruction (always enabled). The maskable sources include two external interrupts (INT0# and INT1#), three timer interrupts (timers 0, 1, and 2), one programmable counter array (PCA) interrupt, and one serial port interrupt. Each interrupt (except TRAP) has an interrupt request flag, which can be set by software as well as by hardware (see Table 5-3 on page 5-4). For some interrupts, hardware clears the request flag when it grants an interrupt. Software can clear any request flag to cancel an impending interrupt.

5.2.1 External Interrupts

External interrupts INT0# and INT1# (INT*x*#) pins may each be programmed to be level-triggered or edge-triggered, dependent upon bits IT0 and IT1 in the TCON register (see Figure 7-6 on page 7-8). If IT*x* = 0, INT*x#* is triggered by a detected low at the pin. If IT*x* = 1, INT*x#* is negative-edge triggered. External interrupts are enabled with bits EX0 and EX1 (EX*x*) in the IE0 register (see Figure 5-2 on page 5-6). Events on the external interrupt pins set the interrupt request flags IE*x* in TCON. These request bits are cleared by hardware vectors to service routines only if the interrupt is negative-edge triggered. If the interrupt is level-triggered, the interrupt service routine must clear the request bit. External hardware must deassert INTx# before the service routine completes, or an additional interrupt is requested. External interrupt pins must be deasserted for at least four state times prior to a request.

External interrupt pins are sampled once every four state times (a frame length of 666.4 ns at 12 MHz). A level-triggered interrupt pin held low or high for any five-state time period guarantees detection. Edge-triggered external interrupts must hold the request pin low for at least five state times. This ensures edge recognition and sets interrupt request bit EX*x*. The CPU clears EX*x* automatically during service routine fetch cycles for edge-triggered interrupts.

Table 5-3. Interrupt Control Matrix

Interrupt Name	Global Enable	PCA	Timer 2	Serial Port	Timer 1	INT1#	Timer 0	INT0#
Bit Name in Interrupt Enable Register @S:A8H	EA	EC	ET2	ES	ET1	EX1	ET0	EX0
Interrupt Priority-Within-Level (7 = Low Priority, 1 = High Priority)	NA	7	6	5	4	3	2	1
Bit Name in Interrupt Priority Low @S:B8H	Reserved	IPL0.6	IPL0.5	IPL0.4	IPL0.3	IPL0.2	IPL0.1	IPL0.0
Bit Name in Interrupt Priority High @S:B7H	Reserved	IPH0.6	IPH0.5	IPH0.4	IPH0.3	IPH0.2	IPH0.1	IPH0.0
Programmable for Negative-edge Triggered or Level-triggered Detect?	NA	Edge	No	No	No	Yes	No	Yes
Request Flag	NA	CF, CCF*X*	TF2, EXF2	RI, TI	TF1	IE1	TF0	IE0
Request Flag Cleared by Hardware?	No	No	No	No	Yes	Edge Yes, Level No	YES	Edge Yes, Level No
ISR Vector Address	NA	FF: 0033H	FF: 002BH	FF: 0023H	FF: 001BH	FF: 0013H	FF: 000BH	FF: 0003H

5.2.2 Timer Interrupts

Two timer-interrupt request bits TF0 and TF1 (see TCON register, Figure 7-6 on page 7-8) are set by timer overflow (the exception is Timer 0 in Mode 3, see Figure 7-4 on page 7-6). When a timer interrupt is generated, the bit is cleared by an on-chip-hardware vector to an interrupt service routine. Timer interrupts are enabled by bits ET0, ET1, and ET2 in the IE0 register (see Figure 5-2 on page 5-6).

Timer 2 interrupts are generated by a logical OR of bits TF2 and EXF2 in register T2CON (see Figure 7-12 on page 7-17). Neither flag is cleared by a hardware vector to a service routine. In fact, the interrupt service routine must determine if TF2 or EXF2 generated the interrupt, and then clear the bit. Timer 2 interrupt is enabled by ET2 in register IE0.

5.3 PROGRAMMABLE COUNTER ARRAY (PCA) INTERRUPT

The programmable counter array (PCA) interrupt is generated by logical OR of five event flags (CCF*x*) and the PCA timer overflow flag (CF) in the CCON register (see Figure 8-8 on page 8-14). All PCA interrupts share a common interrupt vector. Bits are not cleared by hardware vectors to service routines. Normally, interrupt service routines resolve interrupt requests and clear flag bits. This allows the user to define the relative priorities of the five PCA interrupts.

The PCA interrupt is enabled by bit EC in the IE0 register (see Figure 5-1 on page 5-2. In addition, the CF flag and each of the CCF*x* flags must also be individually enabled by bits ECF and ECCF*x* in registers CMOD and CCAPM*x* respectively for the flag to generate an interrupt (see Figure 8-8 on page 8-14 and Figure 8-9 on page 8-16).

NOTE

> CCF*x* refers to 5 separate bits, one for each PCA module (CCF0, CCF1, CCF2, CCF3, CCF4).
>
> CCAPM*x* refers to 5 separate registers, one for each PCA module (CCAPM0, CCAPM1, CCAPM2, CCAPM3, CCAPM4).

5.4 SERIAL PORT INTERRUPT

Serial port interrupts are generated by the logical OR of bits RI and TI in the SCON register (see Figure 9-2 on page 9-3). Neither flag is cleared by a hardware vector to the service routine. The service routine resolves RI or TI interrupt generation and clears the serial port request flag. The serial port interrupt is enabled by bit ES in the IE0 register (see Figure 5-2 on page 5-6).

5.5 INTERRUPT ENABLE

Each interrupt source (with the exception of TRAP) may be individually enabled or disabled by the appropriate interrupt enable bit in the IE0 register at S:A8H (see Figure 5-2 on page 5-6). Note IE0 also contains a global disable bit (EA). If EA is set, interrupts are individually enabled or disabled by bits in IE0. If EA is clear, all interrupts are disabled.

IE0

Address: S:A8H
Reset State: 0000 0000B

7							0
EA	EC	ET2	ES	ET1	EX1	ET0	EXO

Bit Number	Bit Mnemonic	Function
7	EA	Global Interrupt Enable: Setting this bit enables all interrupts that are individually enabled by bits 0–6. Clearing this bit disables all interrupts, except the TRAP interrupt, which is always enabled.
6	EC	PCA Interrupt Enable: Setting this bit enables the PCA interrupt.
5	ET2	Timer 2 Overflow Interrupt Enable: Setting this bit enables the timer 2 overflow interrupt.
4	ES	Serial I/O Port Interrupt Enable: Setting this bit enables the serial I/O port interrupt.
3	ET1	Timer 1 Overflow Interrupt Enable: Setting this bit enables the timer 1 overflow interrupt.
2	EX1	External Interrupt 1 Enable: Setting this bit enables external interrupt 1.
1	ET0	Timer 0 Overflow Interrupt Enable: Setting this bit enables the timer 0 overflow interrupt.
0	EX0	External Interrupt 0 Enable: Setting this bit enables external interrupt 0.

Figure 5-2. Interrupt Enable Register

5.6 INTERRUPT PRIORITIES

Each of the seven interrupt sources on the 8XC251SB may be individually programmed to one of four priority levels. This is accomplished by a bit in the interrupt priority low and high registers (IPH0.*x*/IPL0.*x*, see Figure 5-3 and Figure 5-4 on page 5-8). The IPH0 register has the same bit map as the IPL0 register. This gives each interrupt source two priority-level select bits (see Table 5-4). The MSB of the priority select bits is in the IPH0 register, and the LSB is in the IPL0 register.

Table 5-4. Level of Priority

IPH0.*X* (MSB)	IPL0.*X* (LSB)	Priority Level
0	0	0 Lowest Priority
0	1	1
1	0	2
1	1	3 Highest Priority

A low-priority interrupt is always interrupted by a higher priority interrupt but not by another interrupt of equal or lower priority. The highest priority interrupt is not interrupted by any other interrupt source. Higher priority interrupts are serviced before lower priority interrupts. The response to simultaneous occurrence of equal priority interrupts (i.e., sampled within the same four state interrupt cycle) is determined by a hardware priority-within-level resolver (see Table 5-5).

Table 5-5. Interrupt Priority Within Level

Priority Number	Interrupt Name
1(Highest Priority)	INT0#
2	Timer 0
3	INT1#
4	Timer 1
5	Serial Port
6	Timer 2
7(Lowest Priority)	PCA

NOTE

The 8XC251SB interrupt priority-within-level table (Table 5-5) differs from MCS® 51 microcontrollers. Other MCS 251 microcontrollers may have unique interrupt priority-within-level tables.

IPH0

Address: S:B7H
Reset State: 0000 0000B

7							0
—	IPH0.6	IPH0.5	IPH0.4	IPH0.3	IPH0.2	IPH0.1	IPH0.0

Bit Number	Bit Mnemonic	Function
7	—	Reserved. The value read from this bit is indeterminate. Do not write a “1” to this bit.
6	IPH0.6	PCA Interrupt Priority Bit High
5	IPH0.5	Timer 2 Overflow Interrupt Priority Bit High
4	IPH0.4	Serial I/O Port Interrupt Priority Bit High
3	IPH0.3	Timer 1 Overflow Interrupt Priority Bit High
2	IPH0.2	External Interrupt 1 Priority Bit High
1	IPH0.1	Timer 0 Overflow Interrupt Priority Bit High
0	IPH0.0	External Interrupt 0 Priority Bit High

Figure 5-3. Interrupt Priority High Register

IPL0

Address: S:B8H
Reset State: 0000 0000B

7							0
—	IPL0.6	IPL0.5	IPL0.4	IPL0.3	IPL0.2	IPL0.1	IPL0.0

Bit Number	Bit Mnemonic	Function
7	—	Reserved. The value read from this bit is indeterminate. Do not write a “1” to this bit.
6	IPL0.6	PCA Interrupt Priority Bit Low
5	IPL0.5	Timer 2 Overflow Interrupt Priority Bit Low
4	IPL0.4	Serial I/O Port Interrupt Priority Bit Low
3	IPL0.3	Timer 1 Overflow Interrupt Priority Bit Low
2	IPL0.2	External Interrupt Priority Bit Low
1	IPL0.1	Timer 0 Overflow Interrupt Priority Bit Low
0	IPL0.0	External Interrupt 0 Priority Bit Low

Figure 5-4. Interrupt Priority Low Register

5.7 INTERRUPT PROCESSING

Interrupt processing is a dynamic operation that begins when a source requests an interrupt and lasts until the execution of the first instruction in the interrupt service routine (see Figure 5-5). *Response time* is the amount of time between the interrupt request and the resulting break in the current instruction stream. *Latency* is the amount of time between the interrupt request and the execution of the first instruction in the interrupt service routine. These periods are dynamic due to the presence of both fixed-time sequences and several variable conditions. These conditions contribute to total elapsed time.

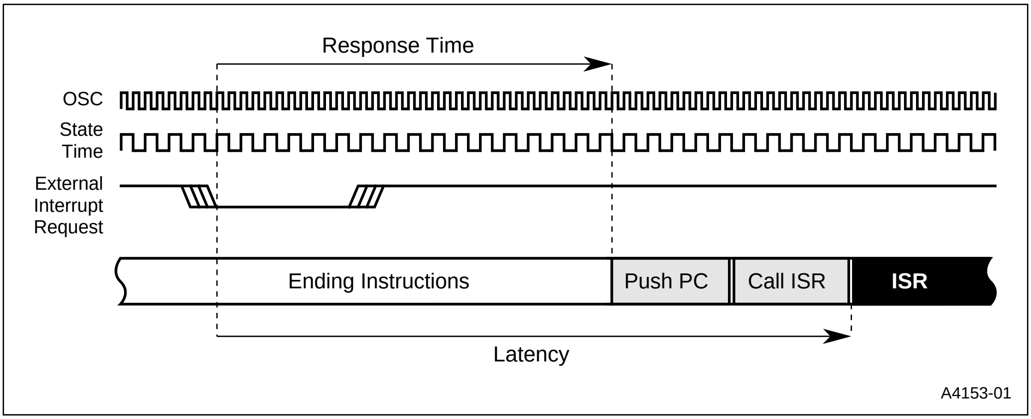


Figure 5-5. The Interrupt Process

Both response time and latency begin with the request. The subsequent minimum fixed sequence comprises the interrupt sample, poll, and request operations. The variables consist of (but are not limited to): specific instructions in use at request time, internal versus external interrupt source requests, internal versus external program operation, stack location, presence of wait states, page-mode operation, and branch pointer length.

NOTE

In the following discussion external interrupt request pins are assumed to be inactive for at least four state times prior to assertion. In this chapter all external hardware signals maintain some setup period (i.e., less than one state time). Signals must meet V_{IH} and V_{IL} specifications prior to any state time under discussion. This setup state time is not included in examples or calculations for either response or latency.

5.7.1 Minimum Fixed Interrupt Time

All interrupts are sampled or polled every four state times (see Figure 5-5 on page 5-9). Two of eight interrupts are latched and polled per state time within any given four-state time window. One additional state time is required for a context switch request. For code branches to jump locations in the current 64-Kbyte memory region (compatible with MCS 51 microcontrollers), the context switch time is 11 states. Therefore, the minimum fixed poll and request time is 16 states (4 poll states + 1 request state + 11 states for the context switch = 16 state times).

Therefore, this minimum fixed period rests upon four assumptions:

- The source request is an internal interrupt with high enough priority to take precedence over other potential interrupts,
- The request is coincident with internal execution and needs no instruction completion time,
- The program uses an internal stack location, and
- The ISR is in on-chip OTPROM/ROM.

5.7.2 Variable Interrupt Parameters

Both response time and latency calculations contain fixed and variable components. By definition, it is often difficult to predict exact timing calculations for real-time requests. One large variable is the completion time of an instruction cycle coincident with the occurrence of an interrupt request. Worst-case predictions typically use the longest-executing instruction in an architecture's code set. In the case of the 8XC251SB, the longest-executing instruction is a 16-bit divide (DIV). However, even this 21- state instruction may have only 1 or 2 remaining states to complete before the interrupt system injects a context switch. This uncertainty affects both response time and latency.

5.7.2.1 Response Time Variables

Response time is defined as the start of a dynamic time period when a source requests an interrupt and lasts until a break in the current instruction execution stream occurs (see Figure 5-5 on page 5-9). Response time (and therefore latency) is affected by two primary factors: the incidence of the request relative to the four-state-time sample window and the completion time of instructions in the response period (i.e., shorter instructions complete earlier than longer instructions).

NOTE

External interrupt signals require one additional state time in comparison to internal interrupts. This is necessary to sample and latch the pin value prior to a poll of interrupts. The sample occurs in the first half of the state time and the poll/request occurs in the second half of the next state time. Therefore, this sample and poll/request portion of the minimum fixed response and latency time is five states for internal interrupts and six states for external interrupts. External interrupts must remain active for at least five state times to guarantee interrupt recognition when the request occurs immediately after a sample has been taken (i.e., requested in the second half of a sample state time).

If the external interrupt goes active one state after the sample state, the pin is not resampled for another three states. After the second sample is taken and the interrupt request is recognized, the interrupt controller requests the context switch. The programmer must also consider the time to complete the instruction at the moment the context switch request is sent to the execution unit. If 9 states of a 10-state instruction have completed when the context switch is requested, the total response time is 6 states, with a context switch immediately after the final state of the 10-state instruction (see Figure 5-6).

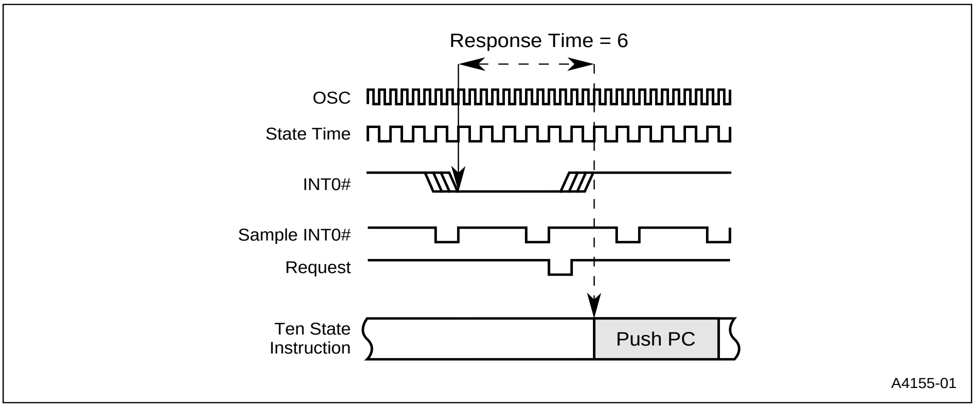


Figure 5-6. Response Time Example #1

Conversely, if the external interrupt requests service in the state just prior to the next sample, response is much quicker. One state asserts the request, one state samples, and one state requests the context switch. If at that point the same instruction conditions exist, one additional state time is needed to complete the 10-state instruction prior to the context switch (see Figure 5-7 on page 5-12). The total response time in this case is four state times. The programmer must evaluate all pertinent conditions for accurate predictability.

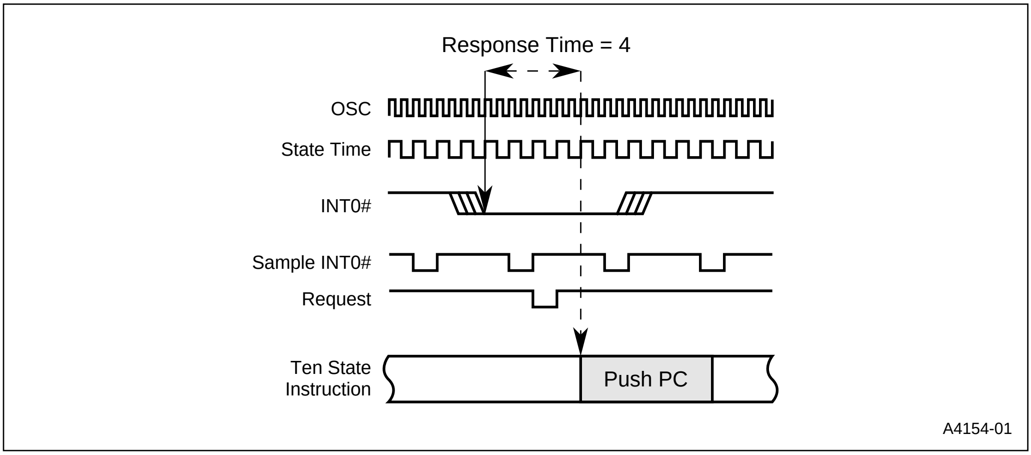

Response Time = 4
OSC
State Time
INT0#
Sample INT0#
Request
Ten State Instruction
Push PC
A4154-01

Figure 5-7. Response Time Example #2

5.7.2.2 Computation of Worst-case Latency With Variables

Worst-case latency calculations assume that the longest 8XC251SB instruction used in the program must fully execute prior to a context switch. The instruction execution time is reduced by one state with the assumption the instruction state overlaps the request state (therefore, 16-bit DIV is 21 state times - 1 = 20 states for latency calculations). The calculations add fixed and variable interrupt times (see Table 5-6 on page 5-13) to this instruction time to predict latency. The worst-case latency (both fixed and variable times included) is expressed by a pseudo-formula:

FIXED_TIME + VARIABLES + LONGEST_INSTRUCTION = MAXIMUM LATENCY PREDICTION

Table 5-6. Interrupt Latency Variables

Variable	INT0#, INT1#, T2EX	External Execution	Page Mode	>64K Jump to ISR (1)	External Memory Wait State	External Stack <64K (1)	External Stack >64K (1)	External Stack Wait State
Number of States Added	1	2	1	8	1 per bus cycle	4	8	1 per bus cycle

NOTES:

1. <64K/>64K means inside/outside the 64-Kbyte memory region where code is executing.
2. Base-case fixed time is 16 states and assumes:
   - — A 2-byte instruction is the first ISR byte.
   - — <64K jump to ISR
   - — Internal peripheral interrupt
   - — Internal execution
   - — Internal stack

5.7.2.3 Latency Calculations

Assume the use of a zero-wait-state external memory where current instructions, the ISR, and the stack are located within the same 64-Kbyte memory region (compatible with memory maps for MCS 51 microcontrollers.) Further, assume there are 3 states yet to complete in the current 21-state DIV instruction when INT0# requests service. Also assume INT0# has made the request one state prior to the sample state (as in Figure 5-7 on page 5-12). Unlike in Figure 5-7, the response time for this assumption is three state times as the current instruction completes in time for the branch to occur. Latency calculations begin with the minimum fixed latency of 16 states. From Table 5-6, one state is added for an INT0# request from external hardware; two states are added for external execution; and four states for an external stack in the current 64-Kbyte region. Finally, three states are added for the current instruction to complete. The actual latency is 26 states. Worst-case latency calculations predict 43 states for this example due to inclusion of total DIV instruction time (less one state).

Table 5-7. Actual vs. Predicted Latency Calculations

Latency Factors	Actual	Predicted
Base Case Minimum Fixed Time	16	16
INT0# External Request	1	1
External Execution	2	2
<64K Byte Stack Location	4	4
Execution Time for Current DIV Instruction	3	20
TOTAL	26	43

5.7.2.4 Blocking Conditions

If all enable and priority requirements have been met, a single prioritized interrupt request at a time generates a vector cycle to an interrupt service routine (see CALL instructions, Appendix A, "Instruction Set Reference"). There are three causes of blocking conditions with hardware-generated vectors:

1. An interrupt of equal or higher priority level is already in progress (defined as any point after the flag has been set and the RETI of the ISR has not executed).
2. The current polling cycle is not the final cycle of the instruction in progress.
3. The instruction in progress is RETI or any write to the IE0, IPH0, or IPL0 registers.

Any of these conditions blocks calls to interrupt service routines. Condition two ensures the instruction in progress completes before the system vectors to the ISR. Condition three ensures at least one more instruction executes before the system vectors to additional interrupts if the instruction in progress is a RETI or any write to IE0, IPH0, or IPL0. The complete polling cycle is repeated each four state times.

5.7.2.5 Interrupt Vector Cycle

When an interrupt vector cycle is initiated, the CPU breaks the instruction stream sequence, resolves all instruction pipeline decisions, and pushes multiple program counter (PC) bytes onto the stack. The CPU then reloads the PC with a start address for the appropriate ISR. The number of bytes pushed to the stack depends upon the INTR bit in the CONFIG1 configuration register (see Figure 13-4 on page 13-8). The complete sample, poll, request and context switch vector sequence is illustrated in the interrupt latency timing diagram (see Figure 5-5 on page 5-9).

NOTE

If the interrupt flag for a level-triggered external interrupt is set but denied for one of the above conditions and is clear when the blocking condition is removed, then the denied interrupt is ignored. In other words, blocked interrupt requests are not buffered for retention.

5.7.3 ISRs in Process

ISR execution proceeds until the RETI instruction is encountered. The RETI instruction informs the processor the interrupt routine is completed. The RETI instruction in the ISR pops PC address bytes off the stack (as well as PSW1 for INTR = 1), and execution resumes at the suspended instruction stream.

NOTE

A simple RET instruction also returns execution to the interrupted program. In previous implementations this inappropriately allowed the system to operate as though an interrupt service routine is still in progress. The 8XC251SB allows use of both RETI and RET instructions for interrupt completion. However, for code expected to run properly on both MCS 51 microcontrollers and 8XC251SB products, only the execution of a RETI instruction is considered proper completion of the interrupt operation.

With the exception of TRAP, the start addresses of consecutive interrupt service routines are eight bytes apart. If consecutive interrupts are used (IE0 and TF0, for example, or TF0 and IE1), the first interrupt routine (if more than seven bytes long) must execute a jump to some other memory location. This prevents overlap of the start address of the following interrupt routine.

6

Input/Output Ports

CHAPTER 6
INPUT/OUTPUT PORTS

6.1 INPUT/OUTPUT PORT OVERVIEW

The 8XC251SB uses input/output (I/O) ports to exchange data with external devices. In addition to performing general-purpose I/O, some ports are capable of external memory operations (see Chapter 12, “External Memory Interface”); others allow for alternate functions. All four 8XC251SB I/O ports are bidirectional. Each port contains a latch, an output driver, and an input buffer. Port 0 and port 2 output drivers and input buffers facilitate external memory operations. Port 0 drives the lower address byte onto the parallel address bus, and port 2 drives the upper address byte (16 or 17) onto the bus. In nonpage mode, the data is multiplexed with the lower address byte on port 0. In page mode, the data is multiplexed with the upper address byte on port 2. All port 1 and port 3 pins serve for both general-purpose I/O and alternate functions (see Table 6-1).

Table 6-1. Input/Output Port Pin Descriptions

Pin Name	Type	Alternate Pin Name	Alternate Description	Alternate Type
P0.7:0	I/O	AD7:0	Address/Data Lines (Nonpage Mode), Address Lines (Page Mode)	I/O
P1.0	I/O	T2	Timer 2 Clock Input/Output	I/O
P1.1	I/O	T2EX	Timer 2 External Input	I
P1.2	I/O	ECI	PCA External Clock Input	I
P1.3	I/O	CEX0	PCA Module 0 I/O	I/O
P1.4	I/O	CEX1	PCA Module 1 I/O	I/O
P1.5	I/O	CEX2	PCA Module 2 I/O	I/O
P1.6	I/O	CEX3	PCA Module 3 I/O	I/O
P1.7	I/O	CEX4	PCA Module 4 I/O	I/O
P2.7:0	I/O	A15:8	Address Lines (Nonpage Mode), Address/Data Lines (Page Mode)	I/O
P3.0	I/O	RXD	Serial Port Receive Data Input	I (I/O)
P3.1	I/O	TXD	Serial Port Transmit Data Output	O (O)
P3.2	I/O	INT0#	External Interrupt 0	I
P3.3	I/O	INT1#	External Interrupt 1	I
P3.4	I/O	T0	Timer 0 Input	I
P3.5	I/O	T1	Timer 1 Input	I
P3.6	I/O	WR#	Write Signal to External Memory	O
P3.7	I/O	RD#/A16	Read Signal to External Memory or 17th Address Bit	O

6.2 I/O CONFIGURATIONS

Each port SFR operates via type-D latches, as illustrated in Figure 6-1 for ports 1 and 3. A CPU “write to latch” signal initiates transfer of internal bus data into the type-D latch. A CPU “read latch” signal transfers the latched Q output onto the internal bus. Similarly, a “read pin” signal transfers the logical level of the port pin. Some port data instructions activate the “read latch” signal while others activate the “read pin” signal. Latch instructions are referred to as read-modify-write instructions (see “Read-Modify-Write Instructions” on page 6-5). Each I/O line may be independently programmed as input or output.

6.3 PORT 1 AND PORT 3

Figure 6-1 shows the structure of ports 1 and 3, which have internal pullups. An external source can pull the pin low. Each port pin can be configured either for general-purpose I/O or for its alternate input or output function (Table 6-1).

To use a pin for general-purpose output, set or clear the corresponding bit in the P*x* register (*x* = 1, 3). To use a pin for general-purpose input, set the bit in the P*x* register. This turns off the output driver FET.

To configure a pin for its alternate function, set the bit in the P*x* register. When the latch is set, the “alternate output function” signal controls the output level (Figure 6-1). The operation of ports 1 and 3 is discussed further in “Quasi-bidirectional Port Operation” on page 6-5.

6.4 PORT 0 AND PORT 2

Ports 0 and 2 are used for general-purpose I/O or as the external address/data bus. Port 0, shown in Figure 6-2, differs from the other ports in not having internal pullups. Figure 6-3 on page 6-4 shows the structure of port 2. An external source can pull a port 2 pin low.

To use a pin for general-purpose output, set or clear the corresponding bit in the P*x* register (*x* = 0, 2). To use a pin for general-purpose input set the bit in the P*x* register to turn off the output driver FET.

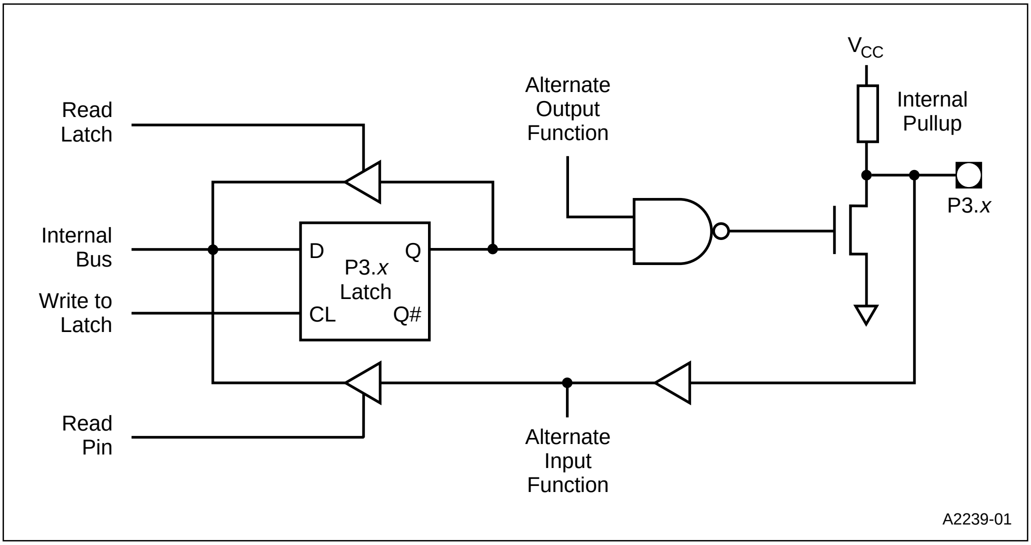

VCC
Read Latch
Alternate Output Function
Internal Pullup
P3.x
Internal Bus
D
Q
P3.x Latch
Write to Latch
CL
Q#
Read Pin
Alternate Input Function
A2239-01

Figure 6-1. Port 1 and Port 3 Structure

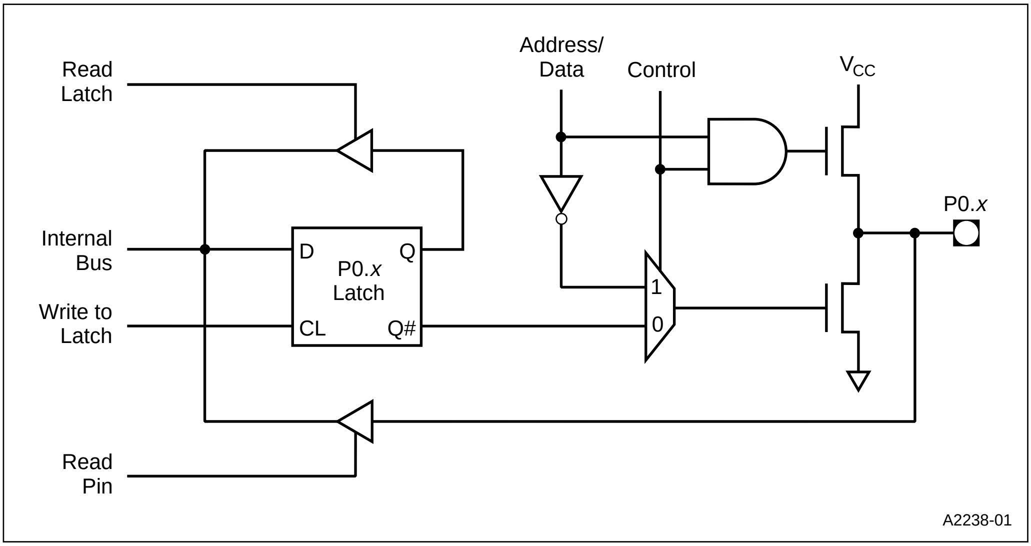

Read Latch
Address/ Data
Control
VCC
P0.x
Internal Bus
D
Q
P0.x Latch
Write to Latch
CL
Q#
1
0
Read Pin
A2238-01

Figure 6-2. Port 0 Structure

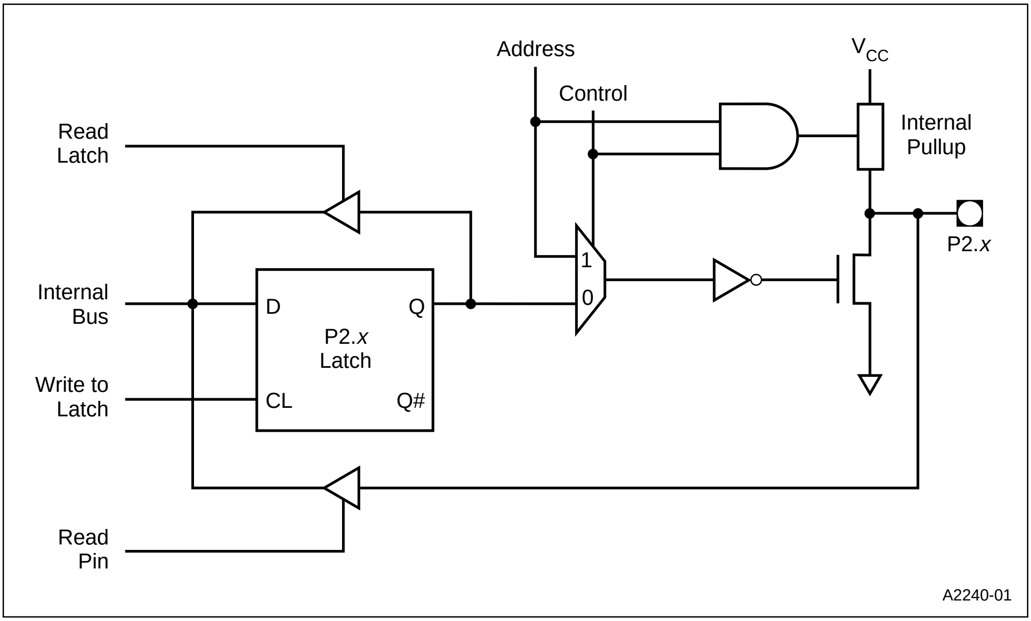

Address
VCC
Control
Read Latch
Internal Pullup
P2.x
1
0
Internal Bus
D
Q
P2.x Latch
Write to Latch
CL
Q#
Read Pin
A2240-01

Figure 6-3. Port 2 Structure

When port 0 and port 2 are used for an external memory cycle, an internal control signal switches the output-driver input from the latch output to the internal address/data line. “External Memory Access” on page 6-7 discusses the operation of port 0 and port 2 as the external address/data bus.

NOTE

Port 0 and port 2 are precluded from use as general purpose I/O ports when used as address/data bus drivers.

Port 0 internal pullups assist the logic-one output for memory bus cycles only. Except for these bus cycles, the pullup FET is off. All other port 0 outputs are open drain.

6.5 READ-MODIFY-WRITE INSTRUCTIONS

Some instructions read the latch data rather than the pin data. The latch based instructions read the data, modify the data, and then rewrite the latch. These are called “read-modify-write” instructions. Below is a complete list of these special instructions. When the destination operand is a port, or a port bit, these instructions read the latch rather than the pin:

`ANL`	`(logical AND, e.g., ANL P1, A)`
`ORL`	`(logical OR, e.g., ORL P2, A)`
`XRL`	`(logical EX-OR, e.g., XRL P3, A)`
`JBC`	`(jump if bit = 1 and clear bit, e.g., JBC P1.1, LABEL)`
`CPL`	`(complement bit, e.g., CPL P3.0)`
`INC`	`(increment, e.g., INC P2)`
`DEC`	`(decrement, e.g., DEC P2)`
`DJNZ`	`(decrement and jump if not zero, e.g., DJNZ P3, LABEL)`
`MOV PX.Y, C`	`(move carry bit to bit Y of port X)`
`CLR PX.Y`	`(clear bit Y of port X)`
`SETB PX.Y`	`(set bit Y of port x)`

It is not obvious the last three instructions in this list are read-modify-write instructions. These instructions read the port (all 8 bits), modify the specifically addressed bit, and write the new byte back to the latch. These read-modify-write instructions are directed to the latch rather than the pin in order to avoid possible misinterpretation of voltage (and therefore, logic) levels at the pin. For example, a port bit used to drive the base of an external transistor appears to provide incorrect information. When logic one is written to the bit, the external base-emitter transistor junction saturates. Due to Kirchoff’s Law of Series Circuits and the characteristics of transistor base-emitter saturation, the voltage measurement on the transistor base is low (below V_{IL}). If the CPU attempts to read the port at the pin, the base voltage of the external transistor is incorrectly interpreted as logic zero. A read of the latch rather than the pin returns the correct logic-one value.

6.6 QUASI-BIDIRECTIONAL PORT OPERATION

Port 1, port 2, and port 3 have fixed internal pullups and are referred to as “quasi-bidirectional” ports. When configured as an input, the pin impedance appears as logic one and sources current (see 8XC251SB datasheet) in response to an external logic-zero condition. Port 0 is a “true bidirectional” pin. The pin floats when configured as input. Resets write logical one to all port latches. If logical zero is subsequently written to a port latch, it can be returned to input conditions by a logical one written to the latch. For additional electrical information, refer to the current 8XC251SB datasheet.

NOTE

Port latch values change near the end of read-modify-write instruction cycles. Output buffers (and therefore the pin state) update early in the instruction after the read-modify-write instruction cycle.

Logical zero-to-one transitions in port 1, port 2, and port 3 utilize an additional pullup to aid this logic transition (see Figure 6-4). This increases switch speed. The extra pullup briefly sources 100 times normal internal circuit current. The internal pullups are field-effect transistors rather than linear resistors. Pullups consist of three p-channel FET (pFET) devices. A pFET is on when the gate senses logical zero and off when the gate senses logical one. pFET #1 is turned on for two oscillator periods immediately after a zero-to-one transition in the port latch. A logic one at the port pin turns on pFET #3 (a weak pullup) through the inverter. This inverter and pFET pair form a latch to drive logic one. pFET #2 is a very weak pullup switched on whenever the associated nFET is switched off. This is traditional CMOS switch convention. Current strengths are 1/10 that of pFET #3.

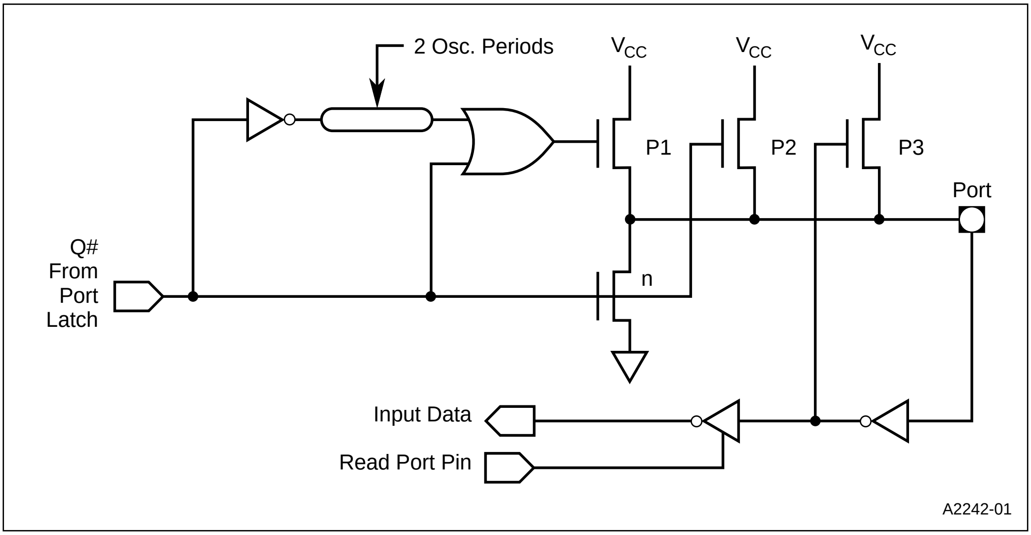


Figure 6-4. Internal Pullup Configurations

6.7 PORT LOADING

Output buffers of port 1, port 2, and port 3 can each sink 1.6 mA at logic zero (see V_{OL} specifications in the 8XC251SB data sheet). These port pins can be driven by open-collector and open-drain devices. Logic zero-to-one transitions occur slowly as limited current pulls the pin to a logic-one condition (Figure 6-4 on page 6-6). A logic-zero input turns off pFET #3. This leaves only pFET #2 weakly in support of the transition. In external bus mode, port 0 output buffers each sink 3.2 mA at logic zero (see V_{OL1} in the 8XC251SB data sheet). However, the port 0 pins require external pullups to drive external gate inputs. See the latest revision of the 8XC251SB datasheet for complete electrical design information. External circuits must be designed to limit current requirements to these conditions.

6.8 EXTERNAL MEMORY ACCESS

The external bus structure is different for page mode and nonpage mode. In nonpage mode (used by MCS 51 microcontrollers), port 2 outputs the upper address byte; the lower address byte and the data are multiplexed on port 0. In page mode, the upper address byte and the data are multiplexed on port 2, while port 0 outputs the lower address byte.

The 8XC251SB CPU writes FFH to the P0 register for all external memory bus cycles. This overwrites previous information in P0. In contrast, the P2 register is unmodified for external bus cycles. When address bits or data bits are not on the port 2 pins, the bit values in P2 appear on the port 2 pins.

In nonpage mode, port 0 uses a strong internal pullup FET to output ones or a strong internal pulldown FET to output zeros for the lower address byte and the data. Port 0 is in a high-impedance state for data input. In page mode, port 0 uses a strong internal pullup FET to output ones or a strong internal pulldown FET to output zeros for the lower address byte or a strong internal pulldown FET to output zeros for the upper address byte.

In nonpage mode, port 2 uses a strong internal pullup FET to output ones or a strong internal pulldown FET to output zeros for the upper address byte. In page mode, port 2 uses a strong internal pullup FET to output ones or a strong internal pulldown FET to output zeros for the upper address byte and data. Port 2 is in a high-impedance state for data input.

NOTE

In external bus mode port 0 outputs do not require external pullups.

There are two types of external memory accesses: external program memory and external data memory (see Chapter 12, “External Memory Interface”). External program memories utilize signal PSEN# as a read strobe. MCS 51 microcontrollers use RD# (read) or WR# (write) to strobe memory for data accesses. Depending on its RD1:0 configuration bits, the 8XC251SB uses PSEN# or RD# for data reads (“RD#, PSEN#, and the Number of External Address Pins (Bits RD1:0)” on page 12-3).

During instruction fetches, external program memory can transfer instructions with 16-bit addresses for binary-compatible code or with 17-bit addresses for extended memory operations.

External data memory transfers use an 8-, 16-, or 17-bit address bus, depending on the instruction. Table 6-2 lists the instructions that can be used for the three bus widths.

Table 6-2. Instructions for External Data Moves

Bus Width	Instructions
8	MOVX @Ri; MOV @Rm; MOV dir8
16	MOVX @DPTR; MOV @WRj; MOV @WRj+dis; MOV dir16
17	MOV @DRk; MOV @DRk+dis

NOTE

Avoid MOV P0 instructions for external memory accesses. These instructions can corrupt input code bytes at port 0.

External signal ALE (address latch enable) facilitates external address latch capture. The address byte is valid after the ALE pin drives V_{OL}. For write cycles, valid data is written to port 0 just prior to the write (WR#) pin asserting V_{OL}. Data remains valid until WR# is undriven. For read cycles, data returned from external memory must appear at port 0 before the read (RD#) pin is undriven (refer to the 8XC251SB datasheet for exact specifications). Wait states, by definition, affect bus-timing.

7

Timer/Counters and Watchdog Timer

CHAPTER 7
TIMER/COUNTERS AND WATCHDOG TIMER

This chapter describes the timer/counters and the watchdog timer (WDT) included as peripherals on the 8XC251SB. When operating as a timer, a timer/counter runs for a programmed length of time, then issues an interrupt request. When operating as a counter, a timer/counter counts negative transitions on an external pin. After a preset number of counts, the counter issues an interrupt request. Timer/counters are covered in sections 7.1 through 7.6.

The watch dog timer provides a way to monitor system operation. It causes a system reset if a software malfunction allows it to expire. The watchdog timer is covered in “Watchdog Timer” on page 7-16.

7.1 TIMER/COUNTER OVERVIEW

The 8XC251SB contains three general-purpose, 16-bit timer/counters. Although they are identified as timer 0, timer 1, and timer 2, you can independently configure each to operate in a variety of modes as a timer or as an event counter. Each timer employs two 8-bit timer registers, used separately or in cascade, to maintain the count. The timer registers and associated control and capture registers are implemented as addressable special function registers (SFRs). Table 7-1 briefly describes the SFRs referred to in this chapter. Four of the SFRs provide programmable control of the timers as follows:

- Timer/counter mode control register (TMOD) and timer/counter control register (TCON) control timer 0 and timer 1.
- Timer/counter 2 mode control register (T2MOD) and timer/counter 2 control register (T2CON) control timer 2

For a map of the SFR address space, see Table 3-4 on page 3-13. Table 7-2 describes the external signals referred to in this chapter.

7.2 TIMER/COUNTER OPERATION

The block diagram in Figure 7-1 depicts the basic logic of the timers. Here timer registers TH*x* and TL*x* (*x* = 0, 1, and 2) connect in cascade to form a 16-bit timer. Setting the run control bit (TR*x)* turns the timer on by allowing the selected input to increment TL*x*. When TL*x* overflows it increments TH*x*; when TH*x* overflows it sets the timer overflow flag (TF*x*) in the TCON or T2CON register. Setting the run control bit does not clear the TH*x* and TL*x* timer registers. The timer registers can be accessed to obtain the current count or to enter preset values. Timer 0 and timer 1 can also be controlled by external pin INT*x*# to facilitate pulse width measurements.

Table 7-1. Timer/Counter and Watchdog Timer SFRs

Mnemonic	Description	Address
TL0 TH0	**Timer 0 Timer Registers.** Used separately as 8-bit counters or in cascade as a 16-bit counter. Counts an internal clock signal with frequency $F_{OSC}/12$ (timer operation) or an external input (event counter operation)	S:8AH S:8CH
TL1 TH1	**Timer 1 Timer Registers.** Used separately as 8-bit counters or in cascade as a 16-bit counter. Counts an internal clock signal with frequency $F_{OSC}/12$ (timer operation) or an external input (event counter operation)	S:8BH S:8DH
TL2 TH2	**Timer 2 Timer Registers.** TL2 and TH2 connect in cascade to provide a 16-bit counter. Counts an internal clock signal with frequency $F_{OSC}/12$ (timer operation) or an external input (event counter operation)	S:CCH S:CDH
TCON	**Timer 0/1 Control Register.** Contains the run control bits, overflow flags, interrupt flags, and interrupt-type control bits for timer 0 and timer 1.	S:88H
TMOD	**Timer 0/1 Mode Control Register.** Contains the mode select bits, counter/timer select bits, and external control gate bits for timer 0 and timer 1.	S:89H
T2CON	**Timer 2 Control Register.** Contains the receive clock, transmit clock, and capture/reload bits used to configure timer 2. Also contains the run control bit, counter/timer select bit, overflow flag, external flag, and external enable for timer 2.	S:C8H
T2MOD	**Timer 2 Mode Control Register.** Contains the timer 2 output enable and down count enable bits.	S:C9H
RCAP2L RCAP2H	**Timer 2 Reload/Capture Registers (RCAP2L, RCAP2H)**. Provide values to and receive values from the timer registers (TL2,TH2.)	S:CAH S:CBH
WDTRST	**Watchdog Timer Reset Register (WDTRST).** Used to reset and enable the WDT.	S:A6H

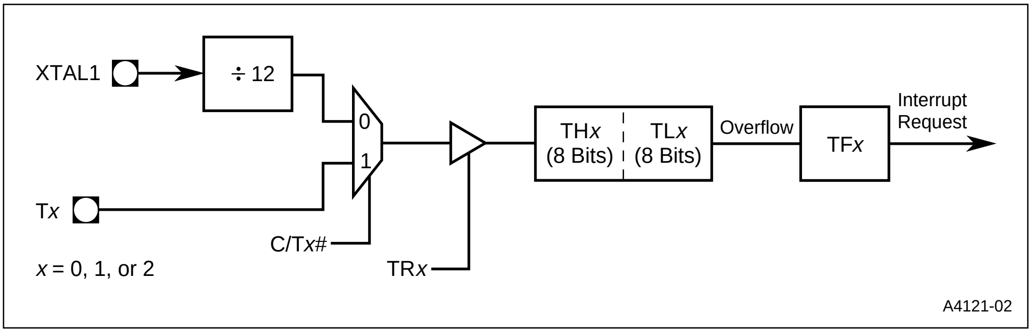


Figure 7-1. Basic Logic of the Timer/Counters

The C\T*x*# control bit selects timer operation or counter operation by selecting the divided-down system clock or external pin T*x* as the source for the counted signal.

For timer operation (C/T*x*# = 0), the timer register counts the divided-down system clock. The timer register is incremented once every peripheral cycle, i.e. once every six states (see "Clock and Reset Unit" on page 2-5). Since six states equals 12 clock cycles, the timer clock rate is $F_{OSC}/12$. Exceptions are the timer 2 baud rate and clock-out modes, where the timer register is incremented by the system clock divided by two.

For counter operation (C/T*x*# = 1), the timer register counts the negative transitions on the T*x* external input pin. The external input is sampled during every S5P2 state. ("Clock and Reset Unit" on page 2-5 describes the notation for the states in a peripheral cycle.) When the sample is high in one cycle and low in the next, the counter is incremented. The new count value appears in the register during the next S3P1 state after the transition was detected. Since it takes 12 states (24 oscillator periods) to recognize a negative transition, the maximum count rate is 1/24 of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full peripheral cycle.

Table 7-2. External Signals

Signal Name	Type	Description	Multiplexed With
T2	I/O	**Timer 2 Clock Input/Output**. This signal is the external clock input for the timer 2 capture mode; and it is the timer 2 clock-output for the clock-out mode.	P1.0
T2EX	I	**Timer 2 External Input**. In timer 2 capture mode, a falling edge initiates a capture of the timer 2 registers. In auto-reload mode, a falling edge causes the timer 2 registers to be reloaded. In the up-down counter mode, this signal determines the count direction: high = up, low = down.	P1.1
INT1:0#	I	**External Interrupts 1:0**. These inputs set the IE1:0 interrupt flags in the TCON register. TCON bits IT1:0 select the triggering method: IT1:0 = 1 selects edge-triggered (high-to-low); IT1:0 = 0 selects level-triggered (active low). INT1:0# also serves as external run control for timer 1:0, when selected by TCON bits GATE1:0#.	P3.3:2
T1:0	I	**Timer 1:0 External Clock Inputs**. When timer 1:0 operates as a counter, a falling edge on the T1:0 pin increments the count.	P3.5:4

7.3 TIMER 0

Timer 0 functions as either a timer or event counter in four modes of operation. Figures 7-2, 7-3, and 7-4 show the logical configuration of each mode.

Timer 0 is controlled by the four low-order bits of the TMOD register (Figure 7-5) and bits 5, 4, 1, and 0 of the TCON register (Figure 7-6). The TMOD register selects the method of timer gating (GATE0), timer or counter operation (T/C0#), and mode of operation (M10 and M00). The TCON register provides timer 0 control functions: overflow flag (TF0), run control (TR0), interrupt flag (IE0), and interrupt type control (IT0).

For normal timer operation (GATE0 = 0), setting TR0 allows TL0 to be incremented by the selected input. Setting GATE0 and TR0 allows external pin INT0# to control timer operation. This setup can be used to make pulse width measurements. See “Pulse Width Measurements” on page 7-10.

Timer 0 overflow (count rolls over from all 1s to all 0s) sets the TF0 flag generating an interrupt request.

7.3.1 Mode 0 (13-bit Timer)

Mode 0 configures timer 0 as an 13-bit timer which is set up as an 8-bit timer (TH0 register) with a modulo 32 prescaler implemented with the lower five bits of the TL0 register (Figure 7-2). The upper three bits of the TL0 register are ignored. Prescaler overflow increments the TH0 register.

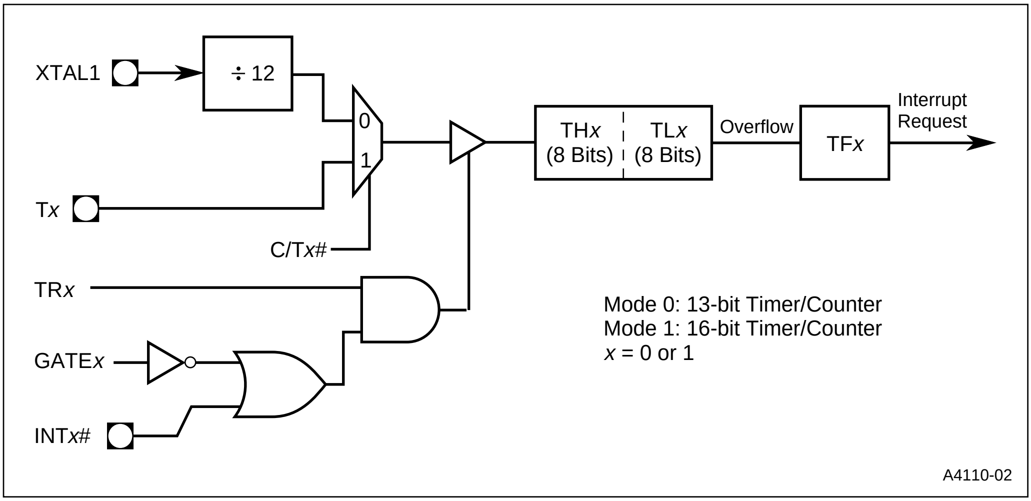

XTAL1
÷ 12
0
1
Tx
C/Tx#
THx (8 Bits)
TLx (8 Bits)
Overflow
TFx
Interrupt Request
TRx
GATEx
INTx#
Mode 0: 13-bit Timer/Counter
Mode 1: 16-bit Timer/Counter
x = 0 or 1
A4110-02

Figure 7-2. Timer 0/1 in Mode 0 and Mode 1

7.3.2 Mode 1 (16-bit Timer)

Mode 1 configures timer 0 as a 16-bit timer with TH0 and TL0 connected in cascade (Figure 7-2). The selected input increments TL0.

7.3.3 Mode 2 (8-bit Timer With Auto-reload)

Mode 2 configures timer 0 as an 8-bit timer (TL0 register) that automatically reloads from the TH0 register (Figure 7-3). TL0 overflow sets the timer overflow flag (TF0) in the TCON register and reloads TL0 with the contents of TH0, which is preset by software. When the interrupt request is serviced, hardware clears TF0. The reload leaves TH0 unchanged. See “Auto-load Setup Example” on page 7-9

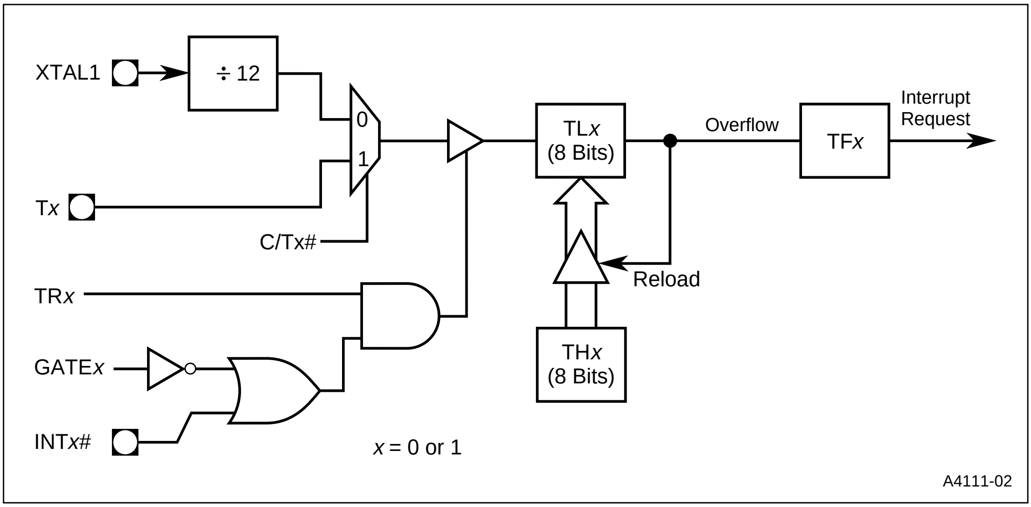


Figure 7-3. Timer 0/1 in Mode 2, Auto-Reload

7.3.4 Mode 3 (Two 8-bit Timers)

Mode 3 configures timer 0 such that registers TL0 and TH0 operate as separate 8-bit timers (Figure 7-4). This mode is provided for applications requiring an additional 8-bit timer or counter. TL0 uses the timer 0 control bits C/T0# and GATE0 in TMOD, and TR0 and TF0 in TCON in the normal manner. TH0 is locked into a timer function (counting F_{OSC} /12) and takes over use of the timer 1 interrupt (TF1) and run control (TR1) bits. Thus, operation of timer 1 is restricted when timer 0 is in mode 3. See “Timer 1” on page 7-6 and “Mode 3 (Halt)” on page 7-9.

7.4 TIMER 1

Timer 1 functions as either a timer or event counter in three modes of operation. Figures 7-2 and 7-3 show the logical configuration for modes 0, 1, and 2. Timer 1's mode 3 is a hold count mode.

Timer 1 is controlled by the four high-order bits of the TMOD register (Figure 7-5) and bits 7, 6, 3, and 2 of the TCON register (Figure 7-6). The TMOD register selects the method of timer gating (GATE1), timer or counter operation (T/C1#), and mode of operation (M11 and M01). The TCON register provides timer 1 control functions: overflow flag (TF1), run control (TR1), interrupt flag (IE1), and interrupt type control (IT1).

Timer 1 operation in modes 0, 1, and 2 is identical to timer 0. Timer 1 can serve as the baud rate generator for the serial port. Mode 2 is best suited for this purpose.

For normal timer operation (GATE1 = 0), setting TR1 allows timer register TL1 to be incremented by the selected input. Setting GATE1 and TR1 allows external pin INT1# to control timer operation. This setup can be used to make pulse width measurements. See "Pulse Width Measurements" on page 7-10.

Timer 1 overflow (count rolls over from all 1s to all 0s) sets the TF1 flag generating an interrupt request.

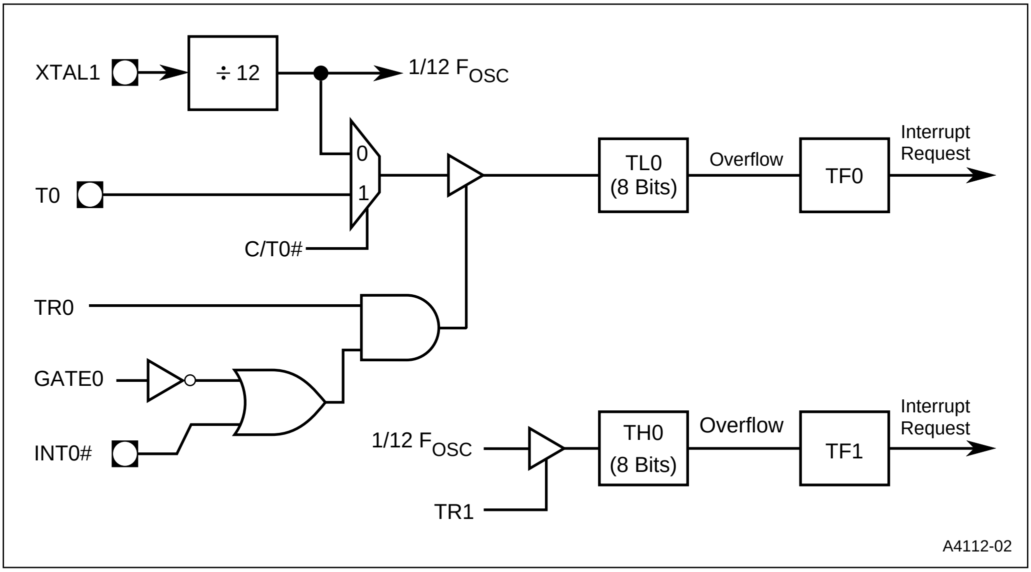

XTAL1
÷ 12
1/12 F_{OSC}
T0
0
1
C/T0#
TL0
(8 Bits)
Overflow
TF0
Interrupt
Request
TR0
GATE0
INT0#
1/12 F_{OSC}
TH0
(8 Bits)
Overflow
TF1
Interrupt
Request
TR1
A4112-02

Figure 7-4. Timer 0 in Mode 3, Two 8-bit Timers

TMOD

Address: S:89H
Reset State: 0000 0000B

7							0
GATE1	C/T1#	M11	M01	GATE0	C/T0#	M10	M00

Bit Number	Bit Mnemonic	Function
7	GATE1	Timer 1 Gate: When GATE1 = 0, run control bit TR1 gates the input signal to the timer register. When GATE1 = 1 and TR1 = 1, external signal INT1 gates the timer input.
6	C/T1#	Timer 1 Counter/Timer Select: C/T1# = 0 selects timer operation: timer 1 counts the divided-down system clock. C/T1# = 1 selects counter operation: timer 1 counts negative transitions on external pin T1.
5, 4	M11, M01	Timer 1 Mode Select: **M11 M01** 0 0 Mode 0: 8-bit timer/counter (TH1) with 5-bit prescaler (TL1) 0 1 Mode 1: 16-bit timer/counter 1 0 Mode 2: 8-bit auto-reload timer/counter (TL1). Reloaded from TH1 at overflow. 1 1 Mode 3: Timer 1 halted. Retains count.
3	GATE0	Timer 0 Gate: When GATE0 = 0, run control bit TR0 gates the input signal to the timer register. When GATE0 = 1 and TR0 = 1, external signal INT0 gates the timer input.
2	C/T0#	Timer 0 Counter/Timer Select: C/T0# = 0 selects timer operation: timer 0 counts the divided-down system clock. C/T0# = 1 selects counter operation: timer 0 counts negative transitions on external pin T0.
1, 0	M10, M00	Timer 0 Mode Select: **M10 M00** 0 0 Mode 0: 8-bit timer/counter (TH0) with 5-bit prescaler (TL0) 0 1 Mode 1: 16-bit timer/counter 1 0 Mode 2: 8-bit auto-reload timer/counter (TL0). Reloaded from TH0 at overflow 1 1 Mode 3: TL0 is 8-bit timer/counter. TH0 is 8-bit timer only using timer 1 TR1 and TF1 bits.

Figure 7-5. TMOD: Timer/Counter Mode Control Register

TCON

Address: S:88H
Reset State: 0000 0000B

7							0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Bit Number	Bit Mnemonic	Function
7	TF1	Timer 1 Overflow Flag: Set by hardware when the timer 1 register overflows. Cleared by hardware when the processor vectors to the interrupt routine.
6	TR1	Timer 1 Run Control Bit: Set/cleared by software to turn timer 1 on/off.
5	TF0	Timer 0 Overflow Flag: Set by hardware when the timer 0 register overflows. Cleared by hardware when the processor vectors to the interrupt routine.
4	TR0	Timer 1 Run Control Bit: Set/cleared by software to turn timer 1 on/off.
3	IE1	Interrupt 1 Flag: Set by hardware when an external interrupt is detected on the INT1# pin. Edge- or level- triggered (see IT1). Cleared when interrupt is processed if edge-triggered.
2	IT1	Interrupt 1 Type Control Bit: Set this bit to select edge-triggered (high-to-low) for external interrupt 1. Clear this bit to select level-triggered (active-low).
1	IE0	Interrupt 1 Flag: Set by hardware when an external interrupt is detected on the INT0# pin. Edge- or level- triggered (see IT0). Cleared when interrupt is processed if edge-triggered.
0	IT0	Interrupt 0 Type Control Bit: Set this bit to select edge-triggered (high-to-low) for external interrupt 0. Clear this bit to select level-triggered (active-low).

Figure 7-6. TCON: Timer/Counter Control Register

When timer 0 is in mode 3, it uses timer 1’s overflow flag (TF1) and run control bit (TR1). For this situation, use timer 1 only for applications that do not require an interrupt (such as a baud rate generator for the serial interface port) and switch timer 1 in and out of mode 3 to turn it off and on.

7.4.1 Mode 0 (13-bit Timer)

Mode 0 configures timer 0 as a 13-bit timer, which is set up as an 8-bit timer (TH1 register) with a modulo-32 prescaler implemented with the lower 5 bits of the TL1 register (Figure 7-2). The upper 3 bits of the TL1 register are ignored. Prescaler overflow increments the TH1 register.

7.4.2 Mode 1 (16-bit Timer)

Mode 1 configures timer 1 as a 16-bit timer with TH1 and TL1 connected in cascade (Figure 7-2). The selected input increments TL1.

7.4.3 Mode 2 (8-bit Timer with Auto-reload)

Mode 2 configures timer 1 as an 8-bit timer (TL1 register) with automatic reload from the TH1 register on overflow (Figure 7-3). Overflow from TL1 sets overflow flag TF1 in the TCON register and reloads TL1 with the contents of TH1, which is preset by software. The reload leaves TH1 unchanged. See “Auto-load Setup Example” on page 7-9

7.4.4 Mode 3 (Halt)

Placing timer 1 in mode 3 causes it to halt and hold its count. This can be used to halt timer 1 when the TR1 run control bit is not available, i.e. when timer 0 is in mode 3. See the final paragraph of “Timer 1” on page 7-6.

7.5 TIMER 0/1 APPLICATIONS

Timer 0 and timer 1 are general purpose timers that can be used in a variety of ways. The timer applications presented in this section are intended to demonstrate timer setup, and do not represent the only arrangement nor necessarily the best arrangement for a given task. These examples employ timer 0, but timer 1 can be set up in the same manner using the appropriate registers.

7.5.1 Auto-load Setup Example

Timer 0 can be configured as an eight-bit timer (TL0) with automatic reload as follows:

1. Program the four low-order bits of the TMOD register (Figure 7-5) to specify: mode 2 for timer 0, C/T0# = 0 to select $F_{OSC}/12$ as the timer input, and GATE0 = 0 to select TR0 as the timer run control.

2. Enter an eight-bit initial value (n_0) in timer register TL0, so that the timer overflows after the desired number of peripheral cycles.
3. Enter an eight-bit reload value (n_R) in register TH0. This can be the same as n_0 or different, depending on the application.
4. Set the TR0 bit in the TCON register (Figure 7-6) to start the timer. Timer overflow occurs after FFH + 1 - n_0 peripheral cycles, setting the TF0 flag and loading n_R into TL0 from TH0. When the interrupt is serviced, hardware clears TF0.
5. The timer continues to overflow and generate interrupt requests every FFH + 1 - n_R peripheral cycles.
6. To halt the timer, clear the TR0 bit.

7.5.2 Pulse Width Measurements

For timer 0 and timer 1, setting GATE*x* and TR*x* allows an external waveform at pin INT*x*# to turn the timer on and off. This setup can be used to measure the width of a positive-going pulse present at pin INT*x*#. Pulse width measurements using timer 0 in mode 1 can be made as follows:

1. Program the four low-order bits of the TMOD register (Figure 7-5) to specify: mode 1 for timer 0, C/T0# = 0 to select $F_{OSC}/12$ as the timer input, and GATE0 = 1 to select INT0 as timer run control.
2. Enter an initial value of all zeros in the 16-bit timer register TH0/TL0, or read and store the current contents of the register.
3. Set the TR0 bit in the TCON register (Figure 7-6) to enable INT0.
4. Apply the pulse to be measured to pin INT0. The timer runs when the waveform is high.
5. Clear the TR0 bit to disable INT0.
6. Read timer register TH0/TL0 to obtain the new value.
7. Calculate pulse width = 12 T_{OSC} × (new value - initial value).
8. Example: F_{OSC} = 16 MHz and $12T_{OSC}$ = 750 ns. If the new value = $10{,}000_{10}$ and the initial value = 0, the pulse width = 750 ns × 10,000 = 7.5 ms.

7.6 TIMER 2

Timer 2 is a 16-bit timer/counter. The count is maintained by two eight-bit timer registers, TH2 and TL2, connected in cascade. The timer/counter 2 mode control register (T2MOD) (Figure 7-11 on page 7-16) and the timer/counter 2 control register (T2CON) (Figure 7-12 on page 7-17) control the operation of timer 2.

Timer 2 provides the following operating modes: capture mode, auto-reload mode, baud rate generator mode, and programmable clock-out mode. Select the operating mode with T2MOD and TCON register bits as shown in Table 7-3 on page 7-15. Auto-reload is the default mode. Setting RCLK and/or TCLK selects the baud rate generator mode.

Timer 2 operation is similar to timer 0 and timer 1. C/T2# selects F_{OSC} /12 (timer operation) or external pin T2 (counter operation) as the timer register input. Setting TF2 allows TL2 to be incremented by the selected input.

The operating modes are described in the following paragraphs. Block diagrams in Figure 7-7 through Figure 7-10 show the timer 2 configuration for each mode.

7.6.1 Capture Mode

In the capture mode, timer 2 functions as a 16-bit timer or counter (Figure 7-7). An overflow condition sets bit TF2, which you can use to request an interrupt. Setting the external enable bit EXEN2 allows the RCAP2H and RCAP2L registers to capture the current value in timer registers TH2 and TL2 in response to a 1-to-0 transition at external input T2EX. The transition at T2EX also sets bit EXF2 in T2CON. The EXF2 bit, like TF2, can generate an interrupt.

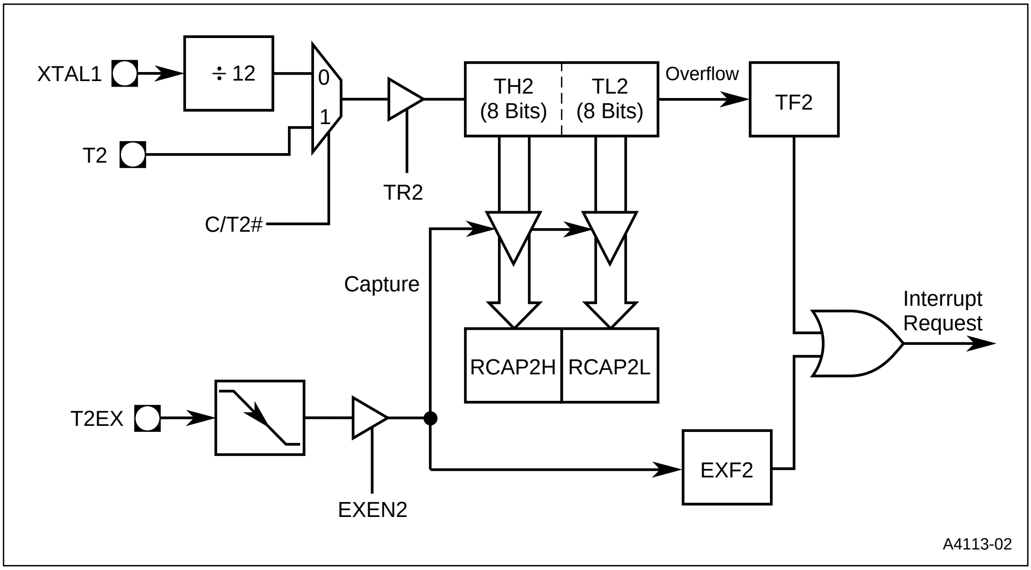


Figure 7-7. Timer 2: Capture Mode

7.6.2 Auto-reload Mode

The auto-reload mode configures timer 2 as a 16-bit timer or event counter with automatic reload. The timer operates an as an up counter or as an up/down counter, as determined by the down counter enable bit (DCEN). At device reset, DCEN is cleared, so in the auto-reload mode, timer 2 defaults to operation as an up counter.

7.6.2.1 Up Counter Operation

When DCEN = 0, timer 2 operates as an up counter (Figure 7-8). The external enable bit EXEN2 in the T2CON register provides two options (Figure 7-12). If EXEN2 = 0, timer 2 counts up to FFFFH and sets the TF2 overflow flag. The overflow condition loads the 16-bit value in the reload/capture registers (RCAP2H, RCAP2L) into the timer registers (TH2, TL2). The values in RCAP2H and RCAP2L are preset by software.

If EXEN2 = 1, the timer registers are reloaded by either a timer overflow or a high-to- low transition at external input T2EX. This transition also sets the EXF2 bit in the T2CON register. Either TF2 or EXF2 bit can generate a timer 2 interrupt request.

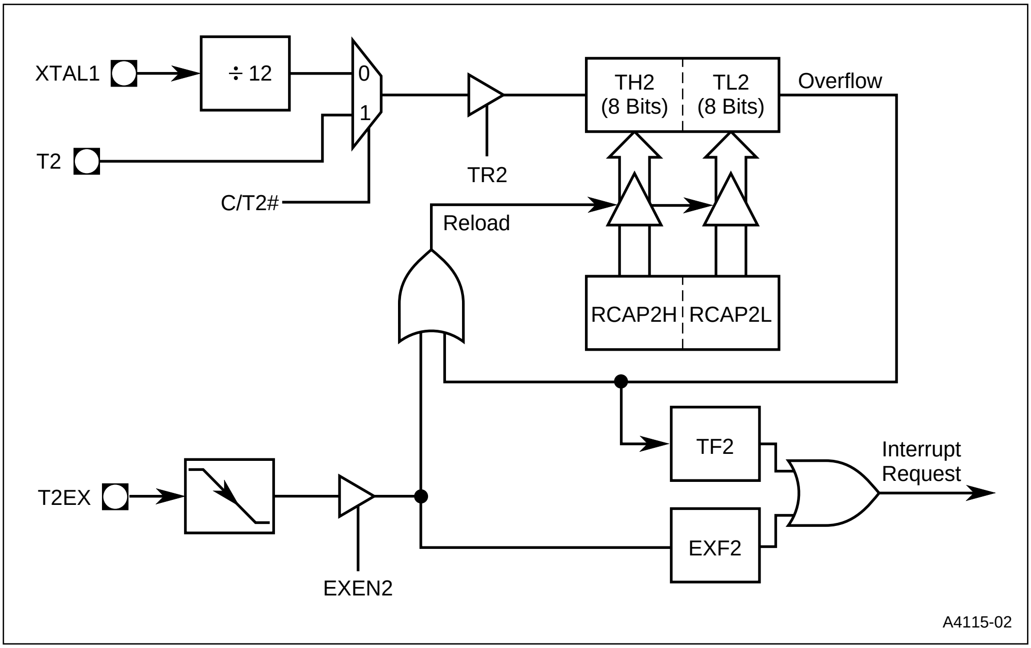

XTAL1
÷ 12
0
1
T2
C/T2#
TR2
TH2 (8 Bits)
TL2 (8 Bits)
Overflow
Reload
RCAP2H
RCAP2L
TF2
Interrupt Request
T2EX
EXF2
EXEN2
A4115-02

Figure 7-8. Timer 2: Auto Reload Mode (DCEN = 0)

7.6.2.2 Up/Down Counter Operation

When DCEN = 1, timer 2 operates as an up/down counter (Figure 7-9). External pin T2EX controls the direction of the count (Table 7-2 on page 7-3). When T2EX is high, timer 2 counts up. The timer overflow occurs at FFFFH which sets the timer 2 overflow flag (TF2) and generates an interrupt request. The overflow also causes the 16-bit value in RCAP2H and RCAP2L to be loaded into the timer registers TH2 and TL2.

When T2EX is low, timer 2 counts down. Timer underflow occurs when the count in the timer registers (TH2, TL2) equals the value stored in RCAP2H and RCAP2L. The underflow sets the TF2 bit and reloads FFFFH into the timer registers.

The EXF2 bit toggles when timer 2 overflows or underflows changing the direction of the count. When timer 2 operates as an up/down counter, EXF2 does not generate an interrupt. This bit can be used to provide 17-bit resolution.

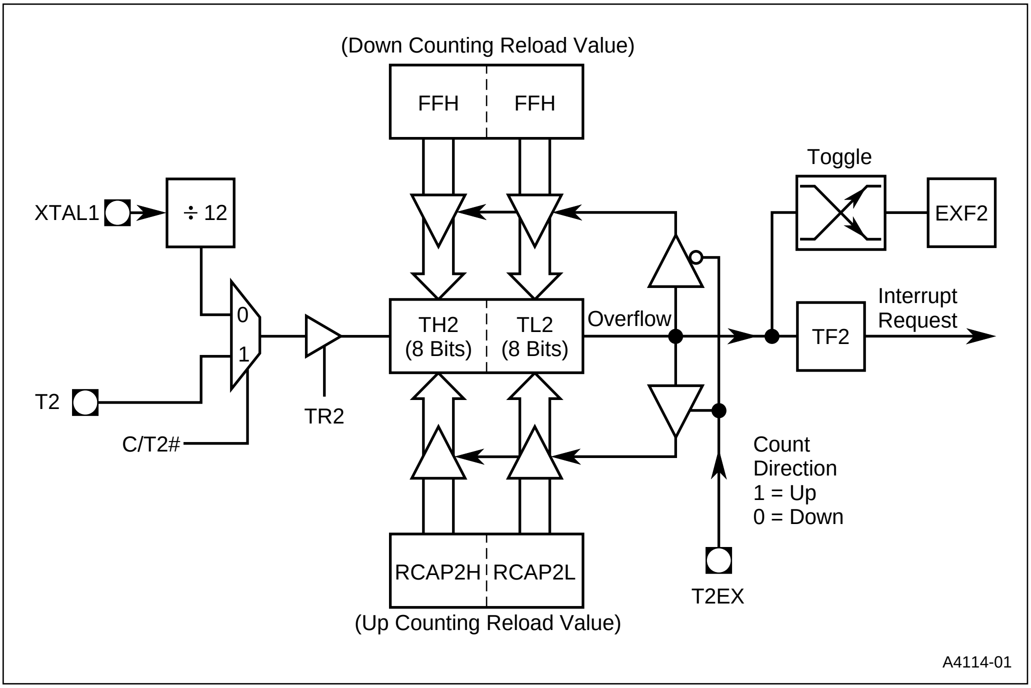

(Down Counting Reload Value)
FFH
FFH
Toggle
XTAL1
÷ 12
EXF2
0
1
TH2 (8 Bits)
TL2 (8 Bits)
Overflow
TF2
Interrupt Request
T2
TR2
C/T2#
Count Direction
1 = Up
0 = Down
RCAP2H
RCAP2L
T2EX
(Up Counting Reload Value)
A4114-01

Figure 7-9. Timer 2: Auto Reload Mode (DCEN = 1)

7.6.3 Baud Rate Generator Mode

This mode configures timer 2 as a baud rate generator for use with the serial port. Select this mode by setting the RCLK and/or TCLK bits in T2CON. See Table 7-3 on page 7-15. For details regarding this mode of operation, refer to “Baud Rates” on page 9-10.

7.6.4 Clock-out Mode

In the clock-out mode, timer 2 functions as a 50%-duty-cycle, variable-frequency clock (Figure 7-10). The input clock increments TL0 at frequency $F_{OSC}/2$. The timer repeatedly counts to overflow from a preloaded value. At overflow, the contents of the RCAP2H and RCAP2L registers are loaded into TH2/TL2. In this mode, timer 2 overflows do not generate interrupts. The formula gives the clock-out frequency as a function of the system oscillator frequency and the value in the RCAP2H and RCAP2L registers:

$$\text{Clock-out Frequency} = \frac{F_{OSC}}{4 \times (65535 - \text{RCAP2H, RCAP2L})}$$

For a 16 MHz system clock, timer 2 has a programmable frequency range of 61 Hz to 4 MHz. The generated clock signal is brought out to the T2 pin.

Timer 2 is programmed for the clock-out mode as follows:

1. Set the T2OE bit in T2MOD. This gates the timer register overflow to the ÷2 counter.
2. Clear the C/T2# bit in T2CON to select $F_{OSC}/2$ as the timer input signal. This also gates the output of the ÷2 counter to pin T2.
3. Determine the 16-bit reload value from formula and enter in the RCAP2H/RCAP2L registers.
4. Enter a 16-bit initial value in timer register TH2/TL2. This can be the same as the reload value or different depending on the application.
5. To start the timer, set the TR2 run control bit in T2CON.

Operation is similar to timer 2 operation as a baud rate generator. It is possible to use timer 2 as a baud rate generator and a clock generator simultaneously. For this configuration, the baud rates and clock frequencies are not independent since both functions use the values in the RCAP2H and RCAP2L registers.

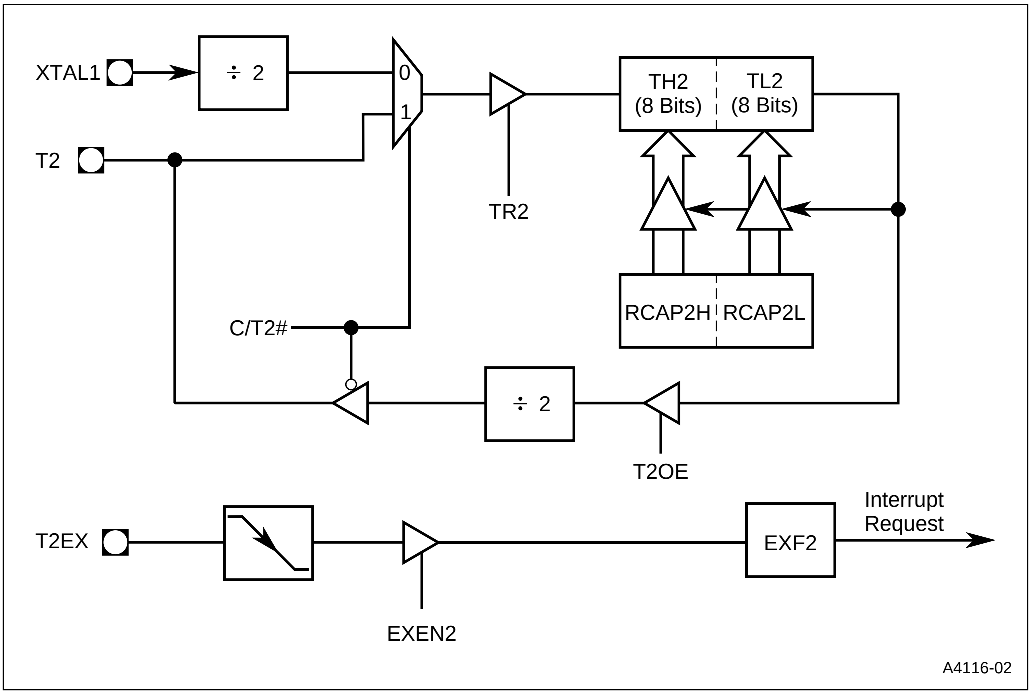

XTAL1
÷ 2
0
1
T2
TR2
TH2
(8 Bits)
TL2
(8 Bits)
RCAP2H
RCAP2L
C/T2#
÷ 2
T2OE
T2EX
EXEN2
EXF2
Interrupt
Request
A4116-02

Figure 7-10. Timer 2: Clock Out Mode

.

Table 7-3. Timer 2 Modes of Operation

Mode	RCLK OR TCLK (in T2CON)	CP/RL2# (in T2CON)	T2OE (in T2MOD)
Auto-reload Mode	0	0	0
Capture Mode	0	1	0
Baud Rate Generator Mode	1	X	X
Programmable Clock-Out	X	0	1

T2MOD

Address: S:C9H
Reset State: XXXX XX00B

7							0
—	—	—	—	—	—	T2OE	DCEN

Bit Number	Bit Mnemonic	Function
7:2	—	Reserved: The values read from these bits are indeterminate. Do not write a “1” to these bits.
1	T2OE	Timer 2 Output Enable Bit: In the timer 2 clock-out mode, connects the programmable clock output to external pin T2.
0	DCEN	Down Count Enable Bit: Configures timer 2 as an up/down counter.

Figure 7-11. T2MOD: Timer 2 Mode Control Register

7.7 WATCHDOG TIMER

The peripheral section of the 8XC251SB contains a dedicated, hardware watchdog timer (WDT) that automatically resets the chip if it is allowed to time out. The WDT provides a means of recovering from routines that do not complete successfully due to software malfunctions. The WDT described in this section is not associated with the PCA watchdog timer, which is implemented in software.

7.7.1 Description

The WDT is a 14-bit counter that counts peripheral cycles, i.e. the system clock divided by twelve ($F_{OSC}/12$). The WDTRST special function register at address S:A6H provides control access to the WDT. Two operations control the WDT:

- Device reset clears and disables the WDT (see “Reset” on page 10-5).
- Writing a specific two-byte sequence to the WDTRST register clears and enables the WDT.

If it is not cleared, the WDT overflows on count 3FFFH + 1. With F_{OSC} = 16 MHz, a peripheral cycle is 750 ns and the WDT overflows in 750 × 16384 = 12.288 ms. The WDTRST is a write-only register. Attempts to read it return FFH. The WDT itself is not read or write accessible. The WDT does **not** drive the external RESET pin.

T2CON

Address: S:C8H
Reset State: 0000 0000B

7							0
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2#	CP/RL2#

Bit Number	Bit Mnemonic	Function
7	TF2	Timer 2 Overflow Flag: Set by timer 2 overflow. Must be cleared by software. TF2 is not set if RCLK = 1 or TCLK = 1.
6	EXF2	Timer 2 External Flag: If EXEN2 = 1, capture or reload caused by a negative transition on T2EX sets EFX2. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1)
5	RCLK	Receive Clock Bit: Selects timer 2 overflow pulses (RCLK = 1) or timer 1 overflow pulses (RCLK = 0) as the baud rate generator for serial port modes 1 and 3.
4	TCLK	Transmit Clock Bit: Selects timer 2 overflow pulses (TCLK = 1) or timer 1 overflow pulses (TCLK = 0) as the baud rate generator for serial port modes 1 and 3.
3	EXEN2	Timer 2 External Enable Bit: Setting EXEN2 causes a capture or reload to occur as a result of a negative transition on T2EX unless timer 2 is being used as the baud rate generator for the serial port. Clearing EXEN2 causes timer 2 to ignore events at T2EX.
2	TR2	Timer 2 Run Control Bit: Setting this bit starts the timer.
1	C/T2#	Timer 2 Counter/Timer Select: C/T2# = 0 selects timer operation: timer 2 counts the divided-down system clock. C/T2# = 1 selects counter operation: timer 2 counts negative transitions on external pin T2.
0	CP/RL2#	Capture/Reload Bit: When set, captures occur on negative transitions at T2EX if EXEN2 = 1. When cleared, auto-reloads occur on timer 2 overflows or negative transitions at T2EX if EXEN2 = 1. The CP/RL2# bit is ignored and timer 2 forced to auto-reload on timer 2 overflow, if RCLK = 1 or TCLK = 1.

Figure 7-12. T2CON: Timer 2 Control Register

7.7.2 Using the WDT

To use the WDT to recover from software malfunctions, the user program should control the WDT as follows:

1. Following device reset, write the two-byte sequence 1EH-E1H to the WDTRST register to enable the WDT. The WDT begins counting from 0.
2. Repeatedly for the duration of program execution, write the two-byte sequence 1EH-E1H to the WDTRST register to clear and enable the WDT before it overflows. The WDT starts over at 0.

If the WDT overflows, it initiates a device reset (see “Reset” on page 10-5). Device reset clears the WDT and disables it.

7.7.3 WDT During Idle Mode

Operation of the WDT during the power reduction modes deserves special attention. The WDT continues to count while the microcontroller is in idle mode. This means the user must service the WDT during idle. One approach is to use a peripheral timer to generate an interrupt request when the timer overflows. The interrupt service routine then clears the WDT, reloads the peripheral timer for the next service period, and puts the microcontroller back into idle.

7.7.4 WDT During PowerDown

The powerdown mode stops all phase clocks. This causes the WDT to stop counting and to hold its count. The WDT resumes counting from where it left off if the powerdown mode is terminated by INT0/INT1. To ensure that the WDT does not overflow shortly after exiting the powerdown mode, clear the WDT just before entering powerdown. The WDT is cleared and disabled if the powerdown mode is terminated by a reset.

8

Programmable Counter Array

CHAPTER 8
PROGRAMMABLE COUNTER ARRAY

This chapter describes the programmable counter array (PCA), an on-chip peripheral of the 8XC251SB that performs a variety of timing and counting operations, including pulse width modulation (PWM). The PCA provides the capability for a software watchdog timer (WDT).

8.1 PCA DESCRIPTION

The programmable counter array (PCA) consists of a 16-bit timer/counter and five 16-bit compare/capture modules. The timer/counter serves as a common time base and event counter for the compare/capture modules, distributing the current count to the modules by means of a 16-bit bus. A special function register (SFR) pair, CH/CL, maintains the count in the timer/counter, while five SFR pairs, CCAP*x*H/CCAP*x*L, store values for the modules (see Figure 8-1). Additional SFRs provide control and mode select functions as follows:

- The PCA timer/counter mode register (CMOD) and the PCA timer/counter control register (CCON) control the operation of the timer/counter. See Figures 8-7 and 8-8 beginning on page 8-13.
- Five PCA module mode registers (CCAPM*x*) specify the operating modes of the compare/capture modules. See Figure 8-9 on page 8-16.

For a list of SFRs associated with the PCA, see Table 8-1. For an SFR address map, see Table 3-4 on page 3-13. Port 1 provides external I/O for the PCA on a shared basis with other functions. Table 8-2 identifies the port pins associated with the timer/counter and compare/capture modules. When not used for PCA I/O, these pins can be used for standard I/O functions.

The operating modes of the five compare/capture modules determine the functions performed by the PCA. Each module can be independently programmed to provide input capture, output compare, or pulse width modulation. Module 4 only also has a watchdog-timer mode.

The PCA timer/counter and the five compare/capture modules share a single interrupt vector. The EC bit in the IE special function register is a global interrupt enable for the PCA. Capture events, compare events in some modes, and PCA timer/counter overflow set flags in the CCON register. Setting the overflow flag (CF) generates a PCA interrupt request if the PCA timer/counter interrupt enable bit (ECF) in the CMOD register is set (Figure 8-1). Setting a compare/capture flag (CCF*x*) generates a PCA interrupt request if the ECCF*x* interrupt enable bit in the corresponding CCAPM*x* register is set (Figures 8-2 and 8-3). For a description of the 8XC251SB interrupt system see Chapter 5, “Interrupt System.”

8.2 PCA TIMER/COUNTER

Figure 8-1 depicts the basic logic of the timer/counter portion of the PCA. The CH/CL special function register pair operates as a 16-bit timer/counter. The selected input increments the CL (low byte) register. When CL overflows, the CH (high byte) register increments after two oscillator periods; when CH overflows it sets the PCA overflow flag (CF in the CCON register) generating a PCA interrupt request if the ECF bit in the CMOD register is set.

The CPS1 and CPS0 bits in the CMOD register select one of four signals as the input to the timer/counter (Figure 8-7 on page 8-13).

- $F_{OSC}/12$. Provides an clock pulse at S5P2 of every peripheral cycle. With F_{OSC} = 16 MHz, the time/counter increments every 750 nanoseconds.
- $F_{OSC}/4$. Provides clock pulses at S1P2, S3P2, and S5P2 of every peripheral cycle. With F_{OSC} = 16 MHz, the time/counter increments every 250 nanoseconds.
- Timer 0 overflow. The CL register is incremented at S5P2 of the peripheral cycle when timer 0 overflows. This selection provides the PCA with a programmable frequency input.
- External signal on P1.2/ECI. The CPU samples the ECI pin at S1P2, S3P2, and S5P2 of every peripheral cycle. The first clock pulse (S1P2, S3P2, or S5P2) that occurs following a high-to-low transition at the ECI pin increments the CL register. The maximum input frequency for this input selection is $F_{OSC}/8$.

For a description of peripheral cycle timing, see “Clock and Reset Unit” on page 2-5.

Setting the run control bit (CR in the CCON register) turns the PCA timer/counter on, if the output of the NAND gate (Figure 8-1) equals logic 1. The PCA timer/counter continues to operate during idle mode unless the CIDL bit of the CMOD register is set. The CPU can read the contents of the CH and CL registers at any time. However, writing to them is inhibited while they are counting i.e., when the CR bit is set.

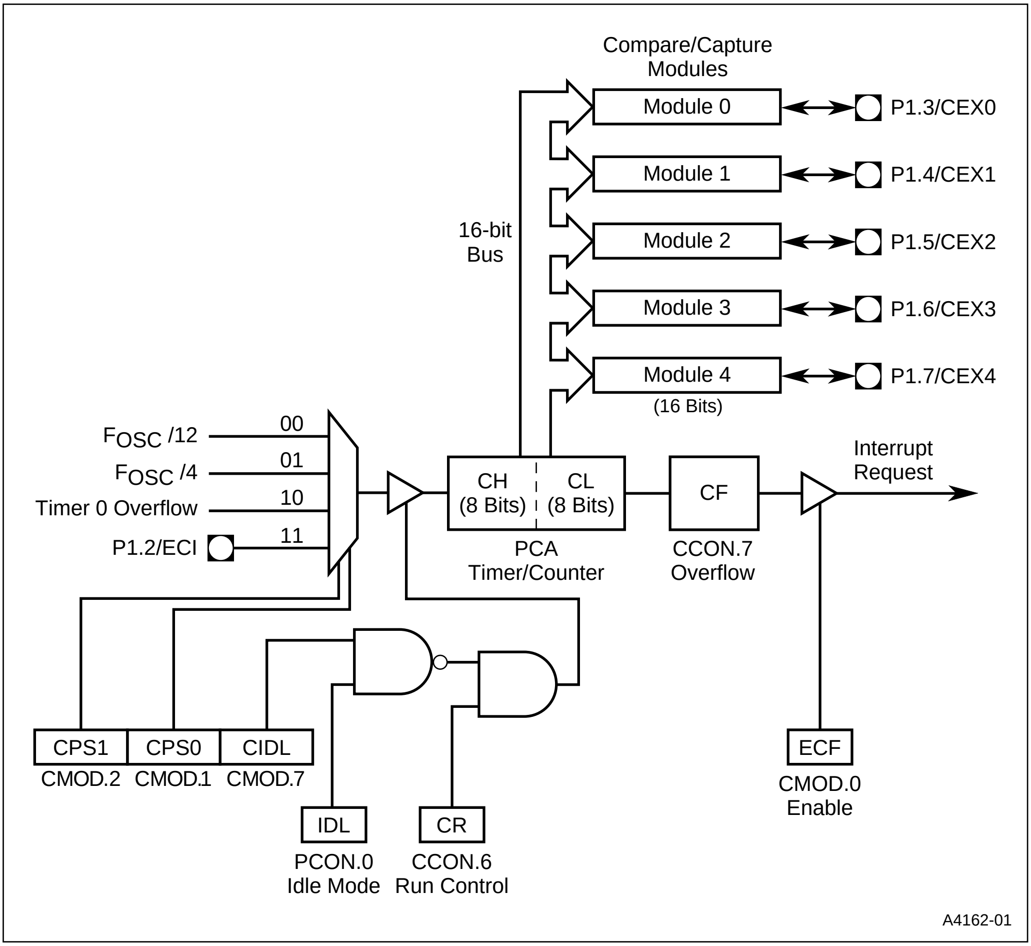

Compare/Capture Modules
Module 0
Module 1
Module 2
Module 3
Module 4
(16 Bits)
P1.3/CEX0
P1.4/CEX1
P1.5/CEX2
P1.6/CEX3
P1.7/CEX4
16-bit Bus
FOSC /12
FOSC /4
Timer 0 Overflow
P1.2/ECI
00
01
10
11
CH (8 Bits)
CL (8 Bits)
PCA Timer/Counter
CF
CCON.7 Overflow
Interrupt Request
CPS1
CPS0
CIDL
CMOD.2
CMOD.1
CMOD.7
IDL
PCON.0 Idle Mode
CR
CCON.6 Run Control
ECF
CMOD.0 Enable
A4162-01

Figure 8-1. Programmable Counter Array

Table 8-1. PCA Special Function Registers (SFRs)

Mnemonic	Description	Address
CL CH	**PCA Timer/Counter.** These registers serve as a common 16-bit timer or event counter for the five compare/capture modules. Counts $F_{OSC}/12$, $F_{OSC}/4$, timer 0 overflow, or the external signal on P1.2/ECI, as selected by CMOD. In PWM mode CL operates as an 8-bit timer.	S:E9H S:F9H
CCON	**PCA Timer/Counter Control Register.** Contains the run control bit and the overflow flag for the PCA timer/counter, and interrupt flags for the five compare/capture modules.	S:D8H
CMOD	**PCA Timer/Counter Mode Register.** Contains bits for disabling the PCA timer/counter during idle mode, enabling the PCA watchdog timer (module 4), selecting the timer/counter input, and enabling the PCA timer/counter overflow interrupt.	S:D9H
CCAP0H CCAP0L	**PCA Module 0 Compare/Capture Registers**. This register pair stores the comparison value or the captured value. In the PWM mode, the low-byte register controls the duty cycle of the output waveform.	S:FAH S:EAH
CCAP1H CCAP1L	**PCA Module 1 Compare/Capture Registers**. This register pair stores the comparison value or the captured value. In the PWM mode, the low-byte register controls the duty cycle of the output waveform.	S:FBH S:EBH
CCAP2H CCAP2L	**PCA Module 2 Compare/Capture Registers**. This register pair stores the comparison value or the captured value. In the PWM mode, the low-byte register controls the duty cycle of the output waveform.	S:FCH S:ECH
CCAP3H CCAP3L	**PCA Module 3 Compare/Capture Registers**. This register pair stores the comparison value or the captured value. In the PWM mode, the low-byte register controls the duty cycle of the output waveform.	S:FDH S:EDH
CCAP4H CCAP4L	**PCA Module 4 Compare/Capture Registers**. This register pair stores the comparison value or the captured value. In the PWM mode, the low-byte register controls the duty cycle of the output waveform.	S:FEH S:EEH
CCAPM0 CCAPM1 CCAPM2 CCAPM3 CCAPM4	**PCA Compare/Capture Module Mode Registers.** Contain bits for selecting the operating mode of the compare/capture modules and enabling the compare/capture flag. See Table 8-3 on page 8-15 for mode select bit combinations.	S:DAH S:DBH S:DCH S:DDH S:DEH

Table 8-2. External Signals

Signal Name	Type	Description	Multiplexed With
ECI	I	**PCA Timer/counter External Input**. This signal is the external clock input for the PCA timer/counter.	P1.2
CEX0 CEX1 CEX2 CEX3 CEX4	I/O	**Compare/Capture Module External I/O.** Each compare/capture module connects to a Port 1 pin for external I/O. When not used by the PCA, these pins can handle standard I/O.	P1.3 P1.4 P1.5 P1.6 P1.7

8.3 PCA COMPARE/CAPTURE MODULES

Each compare/capture module is made up of a compare/capture register pair (CCAP*x*H/CCAPxL), a 16-bit comparator, and various logic gates and signal transition selectors. The registers store the time or count at which an external event occurred (capture) or at which an action should occur (comparison). In the PWM mode, the low-byte register controls the duty cycle of the output waveform.

The logical configuration of a compare/capture module depends on its mode of operation (Figures 8-2 through 8-5). Each module can be independently programmed for operation in any of the following modes:

- 16-bit capture mode with triggering on the positive edge, negative edge, or either edge.
- Compare modes: 16-bit software timer, 16-bit high-speed output, 16-bit WDT (module 4 only), or 8-bit pulse width modulation.
- No operation.

Bit combinations programmed into a compare/capture module's mode register (CCAPM*x*) determine the operating mode. Figure 8-9 on page 8-16 provides bit definitions and Table 8-3 on page 8-15 lists the bit combinations of the available modes. Other bit combinations are invalid and produce undefined results.

The compare/capture modules perform their programmed functions when their common time base, the PCA timer/counter, runs. The timer/counter is turned on and off with the CR bit in the CCON register. To disable any given module, program it for the no operation mode. The occurrence of a capture, software timer, or high-speed output event in a compare/capture module sets the module's compare/capture flag (CCF*x*) in the CCON register and generates a PCA interrupt request if the corresponding enable bit in the CCAPM*x* register is set.

The CPU can read or write the CCAP*x*H and CCAP*x*L registers at any time.

8.3.1 16-bit Capture Mode

The capture mode (Figure 8-2) provides the PCA with the ability to measure periods, pulse widths, duty cycles, and phase differences at up to five separate inputs. External I/O pins CEX0 through CEX4 are sampled for signal transitions (positive and/or negative as specified). When a compare/capture module programmed for the capture mode detects the specified transition, it captures the PCA timer/counter value. This records the time at which an external event is detected, with a resolution equal to the timer/counter clock period.

To program a compare/capture module for the 16-bit capture mode, program the CAPP*x* and CAPN*x* bits in the module's CCAPM*x* register as follows:

- To trigger the capture on a positive transition, set CAPP*x* and clear CAPN*x*.
- To trigger the capture on a negative transition, set CAPN*x* and clear CAPP*x*.
- To trigger the capture on a positive or negative transition, set both CAPPx and CAPNx.

Table 8-3 on page 8-15 lists the bit combinations for selecting module modes. For modules in the capture mode, detection of a valid signal transition at the I/O pin (CEX*x*) causes hardware to load the current PCA timer/counter value into the compare/capture registers (CCAP*x*H/CCAP*x*L) and to set the module's compare/capture flag (CCF*x*) in the CCON register. If the corresponding interrupt enable bit (ECCF*x*) in the CCAPM*x* register is set (Figure 8-9 on page 8-16), a the PCA sends an interrupt request to the interrupt handler.

Since hardware does not clear the event flag when the interrupt is processed, the user must clear the flag in software. A subsequent capture by the same module overwrites the existing captured value. To preserve a captured value, save it in RAM with the interrupt service routine before the next capture event occurs.

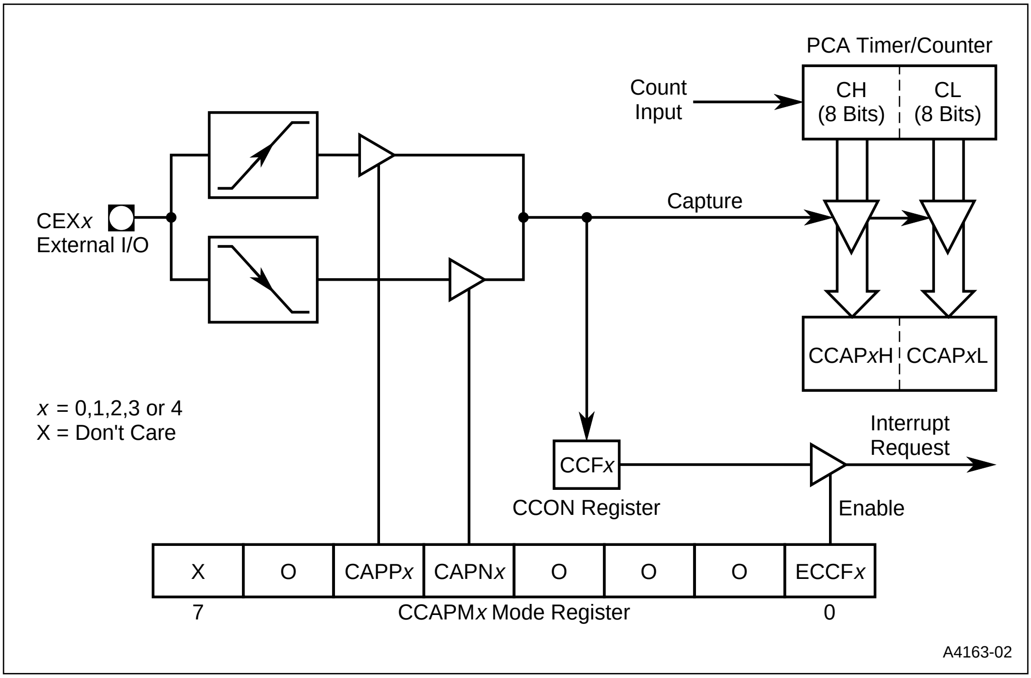

PCA Timer/Counter
Count Input
CH (8 Bits)
CL (8 Bits)
Capture
CEXx
External I/O
CCAPxH
CCAPxL
x = 0,1,2,3 or 4
X = Don't Care
Interrupt Request
CCFx
CCON Register
Enable
X
O
CAPPx
CAPNx
O
O
O
ECCFx
7
CCAPMx Mode Register
0
A4163-02

Figure 8-2. PCA 16-bit Capture Mode

8.3.2 Compare Modes

The compare function provides the capability for operating the five modules as timers, event counters, or pulse width modulators. Four modes employ the compare function: 16-bit software timer mode, high-speed output mode, WDT mode, and PWM mode. In the first three of these, the compare/capture module continuously compares the 16-bit PCA timer/counter value with the 16-bit value pre-loaded into the module's CCAPxH/CCAPxL register pair. In the PWM mode, the module continuously compares the value in the low-byte PCA timer/counter register (CL) with an 8-bit value in the CCAPxL module register. Comparisons are made three times per peripheral cycle to match the fastest PCA timer/counter clocking rate ($F_{OSC}/4$). For a description of peripheral cycle timing, see "Clock and Reset Unit" on page 2-5.

Setting the ECOM*x* bit in a module's mode register (CCAPMx) selects the compare function for that module (Figure 8-9 on page 8-16). To use the modules in the compare modes, observe the following general procedure:

1. Select the module's mode of operation.
2. Select the input signal for the PCA timer/counter.
3. Load the comparison value into the module's compare/capture register pair.
4. Set the PCA timer/counter run control bit.
5. After a match causes an interrupt, clear the module's compare/capture flag.

8.3.3 16-bit Software Timer Mode

To program a compare/capture module for the 16-bit software timer mode (Figure 8-3), set the ECOM*x* and MAT*x* bits in the module's CCAPM*x* register. Table 8-3 on page 8-15 lists the bit combinations for selecting module modes.

A match between the PCA timer/counter and the compare/capture registers (CCAP*x*H/CCAP*x*L) sets the module's compare/capture flag (CCF*x* in the CCON register). This generates an interrupt request if the corresponding interrupt enable bit (ECCF*x* in the CCAPM*x* register) is set. Since hardware does not clear the compare/capture flag when the interrupt is processed, the user must clear the flag in software. During the interrupt routine, a new 16-bit compare value can be written to the compare/capture registers (CCAP*x*H/CCAP*x*L).

NOTE

To prevent an invalid match while updating these registers, user software should write to CCAPxL first, then CCAPxH. A write to CCAP*x*L clears the ECOM*x* bit disabling the compare function, while a write to CCAP*x*H sets the ECOM*x* bit re-enabling the compare function.

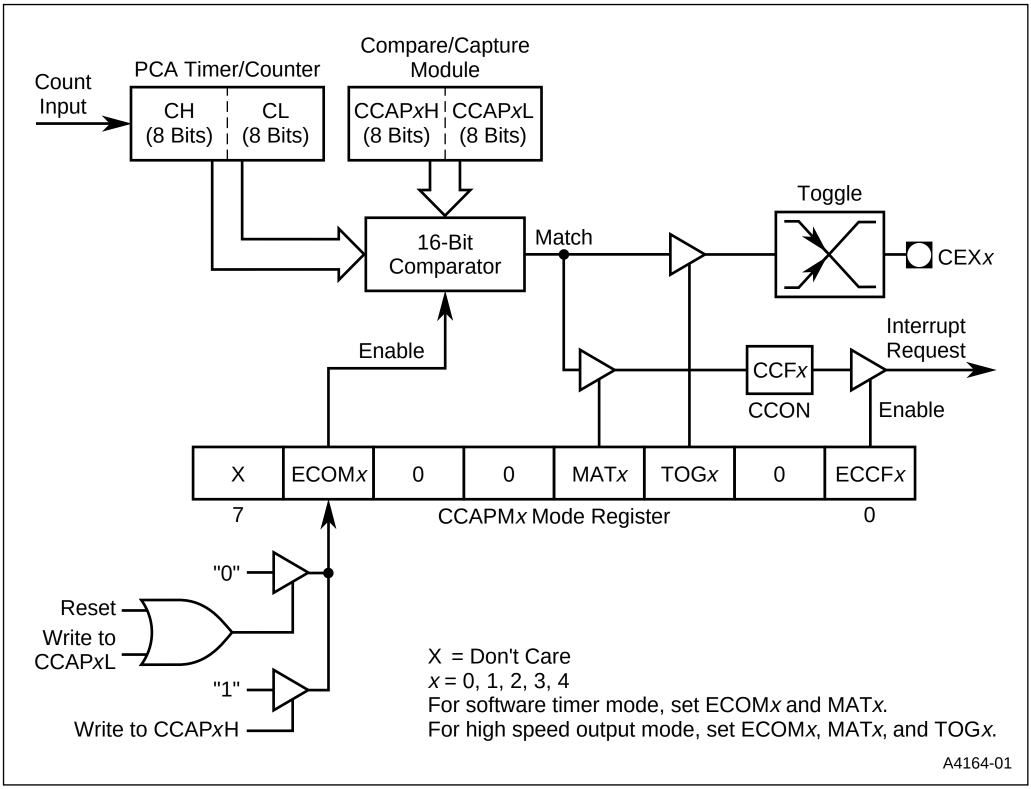

Count Input
PCA Timer/Counter
CH (8 Bits)
CL (8 Bits)
Compare/Capture Module
CCAPxH (8 Bits)
CCAPxL (8 Bits)
16-Bit Comparator
Match
Toggle
CEXx
Interrupt Request
CCFx
CCON
Enable
Enable
X
ECOMx
0
0
MATx
TOGx
0
ECCFx
7
CCAPMx Mode Register
0
"0"
Reset
Write to CCAPxL
"1"
Write to CCAPxH
X = Don't Care
x = 0, 1, 2, 3, 4
For software timer mode, set ECOMx and MATx.
For high speed output mode, set ECOMx, MATx, and TOGx.
A4164-01

Figure 8-3. PCA Software Timer and High-speed Output Modes

8.3.4 High-speed Output Mode

The high-speed output mode (Figure 8-3) generates an output signal by toggling the module's I/O pin (CEX*x*) when a match occurs. This provides greater accuracy than toggling pins in software because the toggle occurs *before* the interrupt request is serviced. Thus, interrupt response time does not affect the accuracy of the output.

To program a compare/capture module for the high-speed output mode, set the ECOM*x*, MAT*x*, TOG*x* bits in the module's CCAPM*x* register. Table 8-3 on page 8-15 lists the bit combinations for selecting module modes. A match between the PCA timer/counter and the compare/capture registers (CCAP*x*H/CCAP*x*L) toggles the CEX*x* pin and sets the module's compare/capture flag (CCF*x* in the CCON register). By setting or clearing the CEX*x* pin in software, the user selects whether the match toggles the pin from low to high or vice versa.

The user also has the option of generating an interrupt request when the match occurs by setting the corresponding interrupt enable bit (ECCF*x* in the CCAPM*x* register). Since hardware does not clear the compare/capture flag when the interrupt is processed, the user must clear the flag in software.

If the user does not change the compare/capture registers in the interrupt routine, the next toggle occurs after the PCA timer/counter rolls over and the count again matches the comparison value. During the interrupt routine, a new 16-bit compare value can be written to the compare/capture registers (CCAP*x*H/CCAP*x*L).

NOTE

To prevent an invalid match while updating these registers, user software should write to CCAPxL first, then CCAPxH. A write to CCAP*x*L clears the ECOM*x* bit disabling the compare function, while a write to CCAP*x*H sets the ECOM*x* bit re-enabling the compare function.

8.3.5 PCA Watchdog Timer Mode

A watchdog timer (WDT) provides the means to recover from routines that do not complete successfully. A WDT automatically invokes a device reset if it does not regularly receive hold-off signals. WDTs are used in applications that are subject to electrical noise, power glitches, electrostatic discharges, etc., or where high reliability is required.

In addition to the 8XC251SB’s 14-bit hardware WDT, the PCA provides a programmable-frequency 16-bit WDT as a mode option on compare/capture module 4. This mode generates a device reset when the count in the PCA timer/counter matches the value stored in the module 4 compare/capture registers. A PCA WDT reset has the same effect as an external reset. Module 4 is the only PCA module that has the WDT mode. When not programmed as a WDT, it can be used in the other modes.

To program module 4 for the PCA WDT mode (Figure 8-4), set the ECOM4 and MAT4 bits in the CCAPM4 register and the WDTE bit in the CMOD register. Table 8-3 on page 8-15 lists the bit combinations for selecting module modes. Also select the desired input for the PCA timer/counter by programming the CPS0 and CPS1 bits in the CMOD register (see Figure 8-7 on page 8-13). Enter a 16-bit comparison value in the compare/capture registers (CCAP4H/CCAP4L). Enter a 16-bit initial value in the PCA timer/counter (CH/CL) or use the reset value (0000H). The difference between these values multiplied by the PCA input pulse rate determines the running time to “expiration.” Set the timer/counter run control bit (CR in the CCON register) to start the PCA WDT.

The PCA WDT generates a reset signal each time a match occurs. To hold off a PCA WDT reset, the user has three options:

- periodically change the comparison value in CCAP4H/CCAP4L so a match never occurs
- periodically change the PCA timer/counter value so a match never occurs
- disable the module 4 reset output signal by clearing the WDTE bit before a match occurs, then later re-enable it

The first two options are more reliable because the WDT is not disabled as in the third option. The second option is not recommended if other PCA modules are in use, since the five modules share a common time base. Thus, in most applications the first option is the best one.

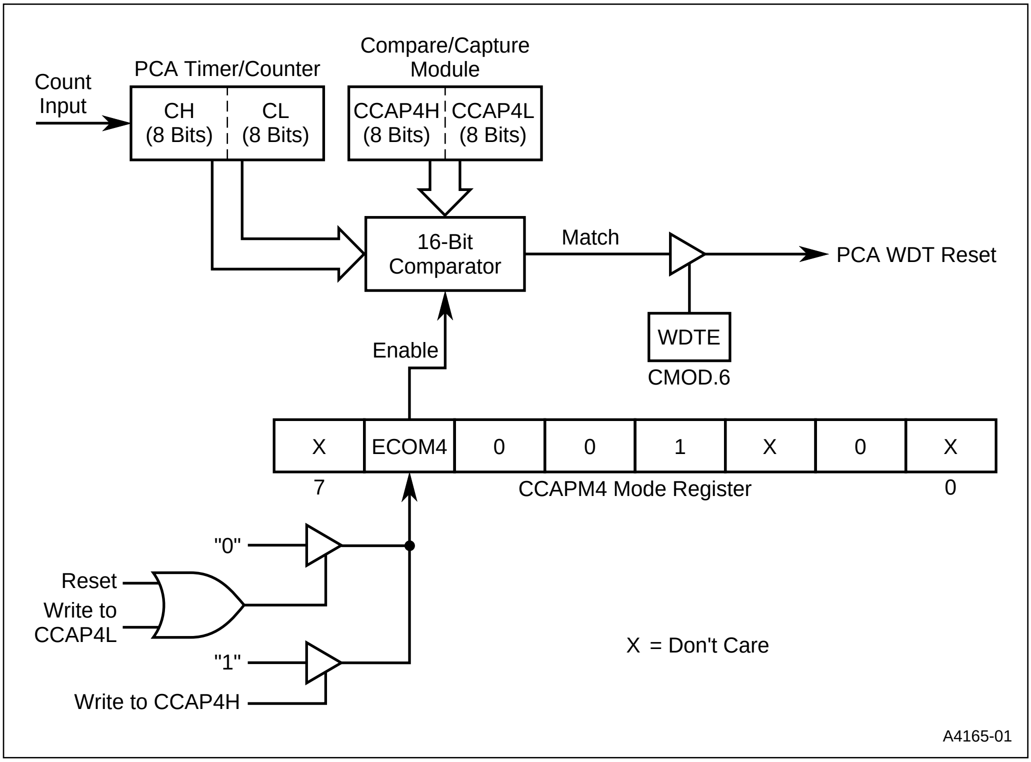

Count Input
PCA Timer/Counter
CH (8 Bits)
CL (8 Bits)
Compare/Capture Module
CCAP4H (8 Bits)
CCAP4L (8 Bits)
16-Bit Comparator
Match
PCA WDT Reset
WDTE
CMOD.6
Enable
X
ECOM4
0
0
1
X
0
X
7
CCAPM4 Mode Register
0
"0"
Reset
Write to CCAP4L
"1"
Write to CCAP4H
X = Don't Care
A4165-01

Figure 8-4. PCA Watchdog Timer Mode

8.3.6 Pulse Width Modulation Mode

The five PCA comparator/capture modules can be independently programmed to function as pulse width modulators (Figure 8-5). The modulated output, which has a pulse width resolution of eight bits, is available at the CEX*x* pin. The PWM output can be used to convert digital data to an analog signal with simple external circuitry.

In this mode the value in the low byte of the PCA timer/counter (CL) is continuously compared with the value in the low byte of the compare/capture register (CCAP*x*L). When CL < CCAP*x*L, the output waveform (Figure 8-6) is low. When a match occurs (CL = CCAP*x*L), the output waveform goes high and remains high until CL rolls over from FFH to 00H, ending the period. At rollover the output returns to a low, the value in CCAP*x*H is loaded into CCAP*x*L, and a new period begins.

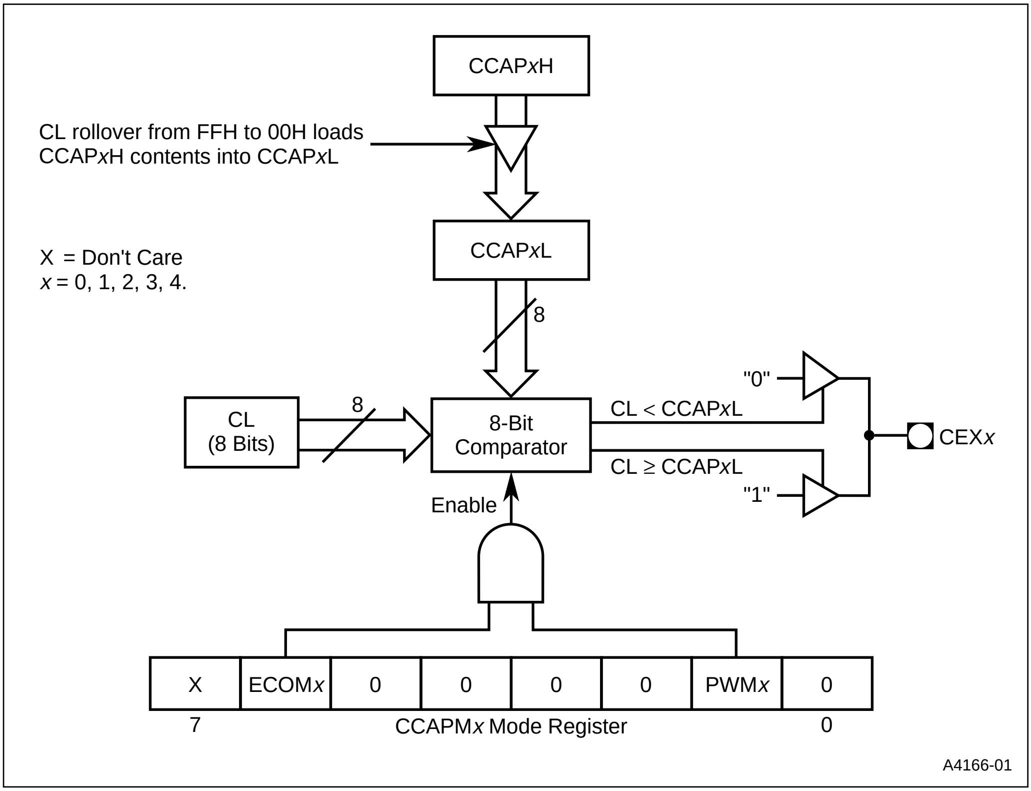

CCAPxH
CL rollover from FFH to 00H loads CCAPxH contents into CCAPxL
X = Don't Care
x = 0, 1, 2, 3, 4.
CCAPxL
8
"0"
CL
(8 Bits)
8
8-Bit Comparator
CL < CCAPxL
CEXx
CL ≥ CCAPxL
"1"
Enable
X
ECOMx
0
0
0
0
PWMx
0
7
CCAPMx Mode Register
0
A4166-01

Figure 8-5. PCA 8-bit PWM Mode

The value in CCAP*x*L determines the duty cycle of the current period. The value in CCAP*x*H determines the duty cycle of the following period. Changing the value in CCAP*x*L over time modulates the pulse width. As depicted in Figure 8-6, the 8-bit value in CCAP*x*L can vary from 0 (100% duty cycle) to 255 (0.4% duty cycle).

NOTE

> To change the value in CCAP*x*L without glitches, write the new value to the high byte register (CCAP*x*H). This value is shifted by hardware into CCAP*x*L when CL rolls over from FFH to 00H.

The frequency of the PWM output equals the frequency of the PCA timer/counter input signal divided by 256. The highest frequency occurs when the $F_{OSC}/4$ input is selected for the PCA timer/counter. For F_{OSC} = 16 MHz, this is 15.6 KHz.

To program a compare/capture module for the PWM mode, set the ECOM*x* and PWM*x* bits in the module's CCAPM*x* register. Table 8-3 on page 8-15 lists the bit combinations for selecting module modes. Also select the desired input for the PCA timer/counter by programming the CPS0 and CPS1 bits in the CMOD register (see Figure 8-7 on page 8-13). Enter an 8-bit value in CCAPxL to specify the duty cycle of the first period of the PWM output waveform. Enter an 8-bit value in CCAP*x*H to specify the duty cycle of the second period. Set the timer/counter run control bit (CR in the CCON register) to start the PCA timer/counter.

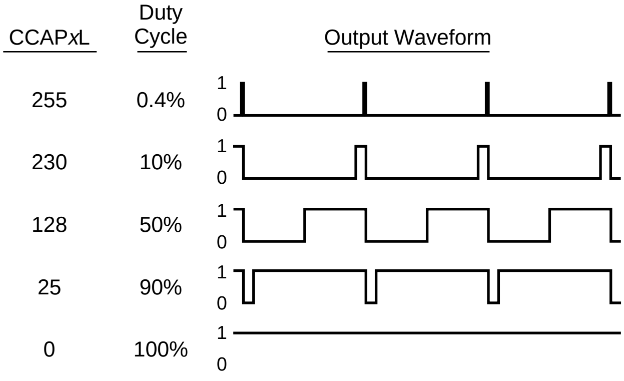


Figure 8-6. PWM Variable Duty Cycle

CMOD

Address: S:D9H
Reset State: 00XX X000B

7							0
CIDL	WDTE	—	—	—	CPS1	CPS0	ECF

Bit Number	Bit Mnemonic	Function
7	CIDL	Counter Idle Control: CIDL = 1 disables the PCA timer/counter during idle mode. CIDL = 0 allows the PCA timer/counter to run during idle mode.
6	WDTE	Watchdog Timer Enable: WDTE = 1 enables the watchdog timer output on PCA module 4. WDTE = 0 disables the PCA watchdog timer output.
5:3	—	Reserved: The values read from these bits are indeterminate. Do not write “1”s to these bits.
2:1	CPS1:0	PCA Timer/Counter Input Select: **CPS1 CPS0** 0 0 $F_{OSC}/12$ 0 1 $F_{OSC}/4$ 1 0 Timer 0 overflow 1 1 External clock at ECI pin (maximum rate = $F_{OSC}/8$)
0	ECF	PCA Timer/Counter Interrupt Enable: ECF = 1 enables the CF bit in the CCON register to generate an interrupt request.

Figure 8-7. CMOD: PCA Timer/Counter Mode Register

CCON

Address: S:D8H
Reset State: 00X0 0000B

7							0
CF	CR	—	CCF4	CCF3	CCF2	CCF1	CCF0

Bit Number	Bit Mnemonic	Function
7	CF	PCA Timer/Counter Overflow Flag: Set by hardware when the PCA timer/counter rolls over. This generates an interrupt request if the ECF interrupt enable bit in CMOD is set. CF can be set by hardware or software but can be cleared only by software.
6	CR	PCA Timer/Counter Run Control Bit: Set and cleared by software to turn the PCA timer/counter on and off.
5	—	Reserved: The value read from this bit is indeterminate. Do not write a “1” to this bit.
4:0	CCF4 CCF3 CCF2 CCF1 CCF0	PCA Module Compare/Capture Flags: Set by hardware when a match or capture occurs. This generates a PCA interrupt request if the ECCF*x* interrupt enable bit in the corresponding CCAPM*x* register is set. Must be cleared by software.

Figure 8-8. CCON: PCA Timer/Counter Control Register

Table 8-3. PCA Module Modes

ECOM*x*	**CAPP*x***	**CAPN*x***	**MAT*x***	**TOG*x***	**PWM*x***	**ECCF*x***	**Module Mode**
0	0	0	0	0	0	0	No operation
X	1	0	0	0	0	X	16-bit capture on positive-edge trigger at CEX*x*
X	0	1	0	0	0	X	16-bit capture on negative-edge trigger at CEX*x*
X	1	1	0	0	0	X	16-bit capture on positive- or negative-edge trigger at CEX*x*
1	0	0	1	0	0	X	Compare: software timer
1	0	0	1	1	0	X	Compare: high-speed output
1	0	0	0	0	1	0	Compare: 8-bit PWM
1	0	0	1	X	0	X	Compare: PCA WDT (CCAPM4 only) (Note 3)

NOTES:

1. This table shows the CCAPM*x* register bit combinations for selecting the operating modes of the PCA compare/capture modules. Other bit combinations are invalid. See Figure 8-9 for bit definitions.
2. *x* = 0–4, X = Don’t care.
3. For PCA WDT mode, also set the WDTE bit in the CMOD register to enable the reset output signal.

CCAPMx (x = 0–4)

Address: CCAPM0 S:DAH
CCAPM1 S:DBH
CCAPM2 S:DCH
CCAPM3 S:DDH
CCAPM4 S:DEH

Reset State: X000 0000B

7							0
—	ECOM*x*	CAPP*x*	CAPN*x*	MAT*x*	TOG*x*	PWM*x*	ECCF*x*

Bit Number	Bit Mnemonic	Function
7	—	Reserved: The value read from this bit is indeterminate. Do not write a “1” to this bit.
6	ECOM*x*	Compare Modes: ECOMx = 1 enables the module comparator function. The comparator is used to implement the software timer, high-speed output, pulse width modulation, and watchdog timer modes.
5	CAPP*x*	Capture Mode (Positive): CAPPx = 1 enables the capture function with capture triggered by a positive edge on pin CEX*x*.
4	CAPN*x*	Capture Mode (Negative): CAPNx = 1 enables the capture function with capture triggered by a negative edge on pin CEX*x*.
3	MAT*x*	Match: Set ECOM*x* and MAT*x* to implement the software timer mode. When MAT*x* = 1, a match of the PCA timer/counter with the compare/capture register sets the CCF*x* bit in the CCON register, flagging an interrupt.
2	TOG*x*	Toggle: Set ECOM*x*, MAT*x*, and TOG*x* to implement the high-speed output mode. When TOG*x* = 1, a match of the PCA timer/counter with the compare/capture register toggles the CEX*x* pin.
1	PWM*x*	Pulse Width Modulation Mode: PWMx = 1 configures the module for operation as an 8-bit pulse width modulator with output waveform on the CEX*x* pin.
0	ECCF*x*	Enable CCFx Interrupt: Enables compare/capture flag CCFx in the CCON register to generate an interrupt request.

Figure 8-9. CCAPM*x*: PCA Compare/Capture Module Mode Registers

9

Serial I/O Port

CHAPTER 9
SERIAL I/O PORT

The serial input/output port supports communication with modems and other external peripheral devices. This chapter provides instructions on programming the serial port and generating the serial I/O baud rates with timer 1 and timer 2.

9.1 OVERVIEW

The serial I/O port provides both synchronous and asynchronous communication modes. It operates as a universal asynchronous receiver and transmitter (UART) in three full-duplex modes (modes 1, 2, and 3). Asynchronous transmission and reception can occur simultaneously and at different baud rates. The UART supports framing-bit error detection, multiprocessor communication, and automatic address recognition. The serial port also operates in a single synchronous mode (mode 0).

The synchronous mode (mode 0) operates at a single baud rate. Mode 2 operates at two baud rates. Modes 1 and 3 operate over a wide range of baud rates, which are generated by timer 1 and timer 2. Baud rates are detailed in .“Baud Rates” on page 9-10.

The serial port signals are defined in Table 9-1, and the serial port special function registers are described in Table 9-2. Figure 9-1 is a block diagram of the serial port.

For the three asynchronous modes, the UART transmits on the TXD pin and receives on the RXD pin. For the synchronous mode (mode 0), the UART outputs a clock signal on the TXD pin and sends and receives messages on the RXD pin (Figure 9-1). The SBUF register, which holds received bytes and bytes to be transmitted, actually consists of two physically different registers. To send, software writes a byte to SBUF; to receive, software reads SBUF. The receive shift register allows reception of a second byte before the first byte has been read from SBUF. However, if software has not read the first byte by the time the second byte is received, the second byte will overwrite the first. The UART sets interrupt bits TI and RI on transmission and reception, respectively. These two bits share a single interrupt request and interrupt vector.

Table 9-1. Serial Port Signals

Function Name	Type	Description	Multiplexed With
TXD	O	**Transmit Data.** In mode 0, TXD transmits the clock signal. In modes 1, 2, and 3, TXD transmits serial data.	P3.1
RXD	I/O	**Receive Data.** In mode 0, RXD transmits and receives serial data. In modes 1, 2, and 3, RXD receives serial data.	P3.0

Table 9-2. Serial Port Special Function Registers

Mnemonic	Description	Address
SBUF	**Serial Buffer.** Two separate registers comprise the SBUF register. Writing to SBUF loads the transmit buffer; reading SBUF accesses the receive buffer.	99H
SCON	**Serial Port Control**. Selects the serial port operating mode. SCON enables and disables the receiver, framing bit error detection, multiprocessor communication, automatic address recognition, and the serial port interrupt bits.	98H
SADDR	**Serial Address.** Defines the individual address for a slave device.	A8H
SADEN	**Serial Address Enable.** Specifies the mask byte that is used to define the given address for a slave device.	B8H

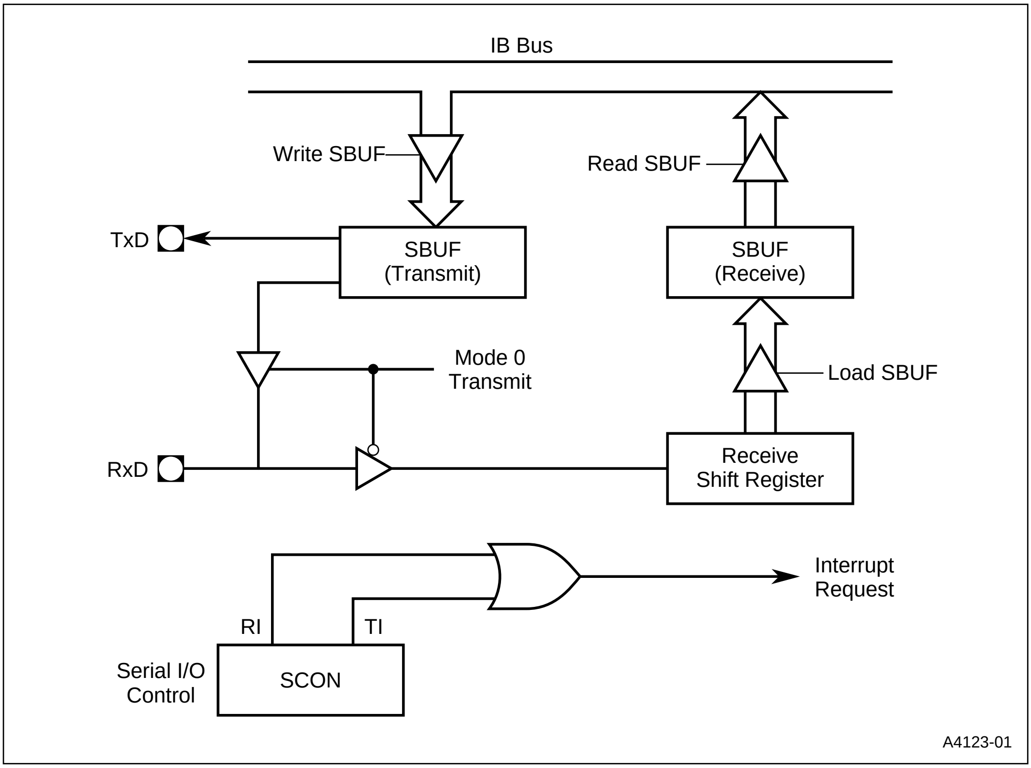

IB Bus
Write SBUF
Read SBUF
TxD
SBUF
(Transmit)
SBUF
(Receive)
Mode 0
Transmit
Load SBUF
RxD
Receive
Shift Register
Interrupt
Request
RI
TI
Serial I/O
Control
SCON
A4123-01

Figure 9-1. Serial Port Block Diagram

The serial port control (SCON) register (Figure 9-2) configures and controls the serial port.

SCON

Address: 98H
Reset State: 0000 0000B

7							0
FE/SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Bit Number	Bit Mnemonic	Function
7	FE SM0	Framing Error Bit: To select this function, set the SMOD0 bit in the PCON register. Set by hardware to indicate an invalid stop bit. Cleared by software. Serial Port Mode Bit 0: To select this function, clear the SMOD0 bit in the PCON register. Software writes to bits SM0 and SM1 to select the serial port operating mode. Refer to the SM1 bit for the mode selections.
6	SM1	Serial Port Mode Bit 1: Software writes to bits SM1 and SM0 (above) to select the serial port operating mode. **SM0 SM1 Mode Description Baud Rate** 0 0 0 Shift register $F_{OSC}/12$ 0 1 1 8-bit UART Variable 1 0 2 9-bit UART $F_{OSC}/32$† or $F_{OSC}/64$† 1 1 3 9-bit UART Variable †Select by programming the SMOD bit in the PCON register (see “Baud Rates” on page 9-10).
5	SM2	Serial Port Mode Bit 2: Software writes to bit SM2 to enable and disable the multiprocessor communication and automatic address recognition features. This allows the serial port to differentiate between data and command frames and to recognize slave and broadcast addresses.
4	REN	Receiver Enable Bit: To enable reception, set this bit. To enable transmission, clear this bit.
3	TB8	Transmit Bit 8: In modes 2 and 3, software writes the 9th data bit to be transmitted to TB8. Not used in modes 0 and 1.
2	RB8	Receiver Bit 8: Mode 0: Not used. Mode 1 (SM2 clear): Set or cleared by hardware to reflect the stop bit received. Modes 2 and 3 (SM2 set): Set or cleared by hardware to reflect the 9th bit received.

Figure 9-2. Serial Port Special Function Register

1	TI	Transmit Interrupt Flag Bit: Set by the transmitter after the last data bit is transmitted. Cleared by software.
0	RI	Receive Interrupt Flag Bit: Set by the receiver after the last data bit of a frame has been received. Cleared by software.

Figure 9-2. Serial Port Special Function Register (Continued)

9.2 MODES OF OPERATION

The serial I/O port can operate in one synchronous and three asynchronous modes.

9.2.1 Synchronous Mode (Mode 0)

Mode 0 is a half-duplex, synchronous mode, which is commonly used to expand the I/O capabilities of a device with shift registers. The transmit data (TXD) pin outputs a set of eight clock pulses while the receive data (RXD) pin transmits or receives a byte of data. The eight data bits are transmitted and received least-significant bit (LSB) first. Shifts occur in the last phase (S6P2) of every peripheral cycle, which corresponds to a baud rate of $F_{OSC}/12$. Figure 9-3 shows the timing for transmission and reception in mode 0.

9.2.1.1 Transmission (Mode 0)

Follow these steps to begin a transmission:

1. Write to the SCON register, clearing bits SM0, SM1, and REN.
2. Write the byte to be transmitted to the SBUF register. This write starts the transmission.

Hardware executes the write to SBUF in the last phase (S6P2) of a peripheral cycle. At S6P2 of the following cycle, hardware shifts the LSB (D0) onto the RXD pin. At S3P1 of the next cycle, the TXD pin goes low for the first clock-signal pulse. Shifts continue every peripheral cycle. In the ninth cycle after the write to SBUF, the MSB (D7) is on the RXD pin. At the beginning of the tenth cycle, hardware drives the RXD pin high and asserts RI to indicate the end of the transmission.

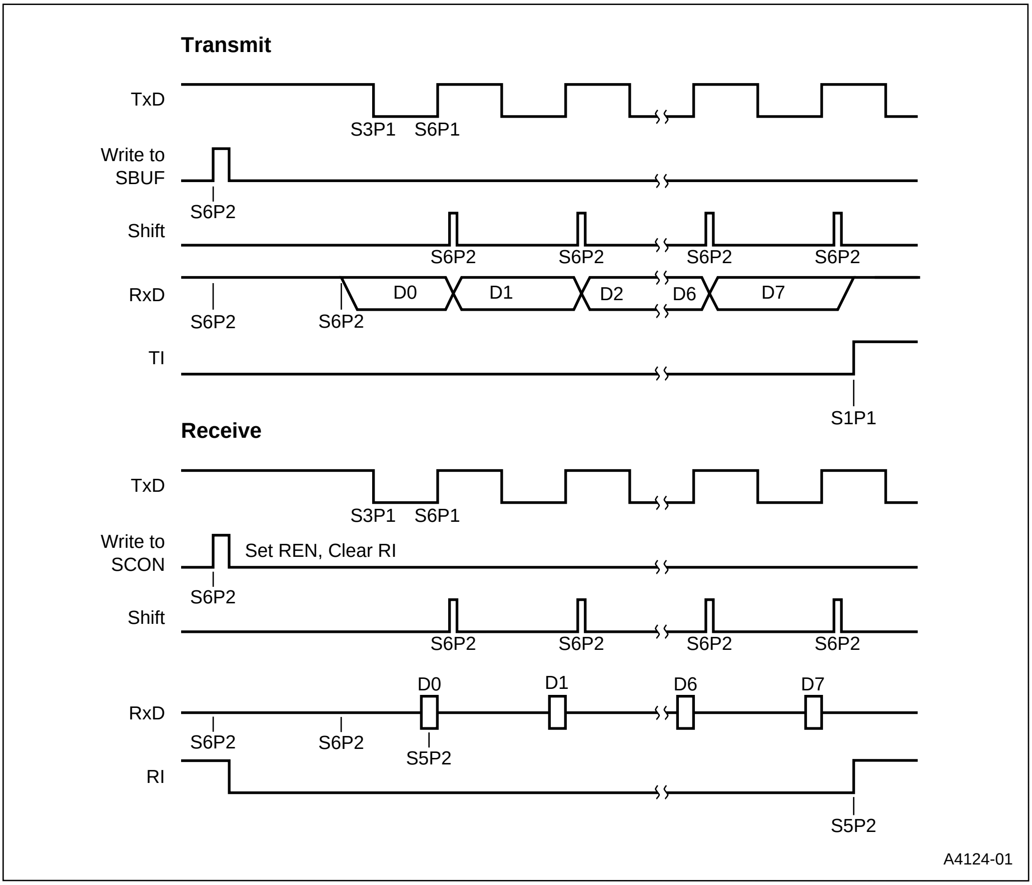

Transmit
TxD
S3P1
S6P1
Write to SBUF
S6P2
Shift
S6P2
S6P2
S6P2
S6P2
RxD
D0
D1
D2
D6
D7
S6P2
S6P2
TI
S1P1
Receive
TxD
S3P1
S6P1
Write to SCON
Set REN, Clear RI
S6P2
Shift
S6P2
S6P2
S6P2
S6P2
D0
D1
D6
D7
RxD
S6P2
S6P2
S5P2
RI
S5P2
A4124-01

Figure 9-3. Mode 0 Timing

9.2.1.2 Reception (Mode 0)

To start a reception in mode 0, write to the SCON register. Clear bits SM0, SM1, and RI and set the REN bit.

Hardware executes the write to SCON in the last phase (S6P2) of a peripheral cycle (Figure 9-3). In the second peripheral cycle following the write to SCON, TXD goes low at S3P1 for the first clock-signal pulse, and the LSB (D0) is sampled on the RXD pin at S5P2. The D0 bit is then shifted into the shift register. After eight shifts at S6P2 of every peripheral cycle, the LSB (D7) is shifted into the shift register, and hardware asserts RI to indicate a completed reception. Software can then read the received byte from SBUF.

9.2.2 Asynchronous Modes (Modes 1, 2, and 3)

The serial port has three asynchronous modes of operation.

- **Mode 1.** Mode 1 is a full-duplex, asynchronous mode. The data frame (Figure 9-4) consists of 10 bits: one start bit, eight data bits, and one stop bit. Serial data is transmitted on the TXD pin and received on the RXD pin. When a message is received, the stop bit is read in the RB8 bit in the SCON register. The baud rate is generated by overflow of timer 1 or timer 2 (see "Baud Rates" on page 9-10).
- **Modes 2 and 3.** Modes 2 and 3 are full-duplex, asynchronous modes. The data frame (Figure 9-4) consists of 11 bits: one start bit, eight data bits (transmitted and received LSB first), one programmable ninth data bit, and one stop bit. Serial data is transmitted on the TXD pin and received on the RXD pin. On receive, the ninth bit is read from the RB8 bit in the SCON register. On transmit, the ninth data bit is written to the TB8 bit in the SCON register. (Alternatively, you can use the ninth bit as a command/data flag.)
  - — In mode 2, the baud rate is programmable to 1/32 or 1/64 of the oscillator frequency.
  - — In mode 3, the baud rate is generated by overflow of timer 1 or timer 2.

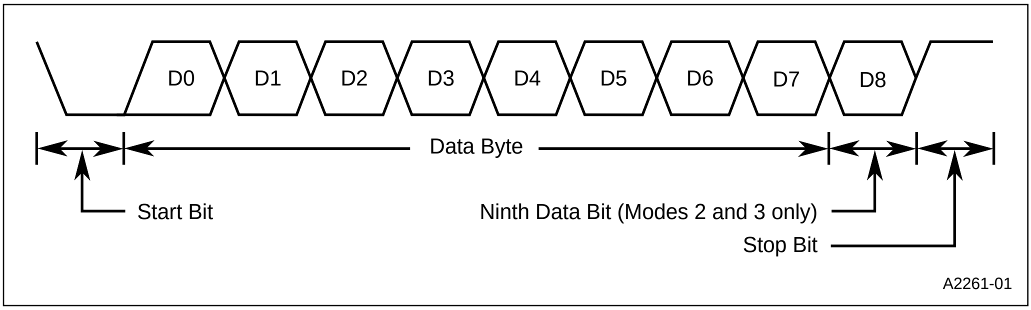


Figure 9-4. Data Frame (Modes 1, 2, and 3)

9.2.2.1 Transmission (Modes 1, 2, 3)

Follow these steps to initiate a transmission:

1. Write to the SCON register. Select the mode with the SM0 and SM1 bits, and clear the REN bit. For modes 2 and 3, also write the ninth bit to the TB8 bit.
2. Write the byte to be transmitted to the SBUF register. This write starts the transmission.

9.2.2.2 Reception (Modes 1, 2, 3)

To prepare for a reception, set the REN bit in the SCON register. The actual reception is then initiated by a detected high-to-low transition on the RXD pin.

9.3 FRAMING BIT ERROR DETECTION (MODES 1, 2, AND 3)

Framing bit error detection is provided for the three asynchronous modes. To enable the framing bit error detection feature, set the SMOD0 bit in the PCON register (see Figure 11-1 on page 11-2). When this feature is enabled, the receiver checks each incoming data frame for a valid stop bit. An invalid stop bit may result from noise on the serial lines or from simultaneous transmission by two CPUs. If a valid stop bit is not found, the software sets the FE bit in the SCON register (see Figure 9-2 on page 9-3).

Software may examine the FE bit after each reception to check for data errors. Once set, only software or a reset can clear the FE bit. Subsequently received frames with valid stop bits cannot clear the FE bit.

9.4 MULTIPROCESSOR COMMUNICATION (MODES 2 AND 3)

Modes 2 and 3 provide a ninth-bit mode to facilitate multiprocessor communication. To enable this feature, set the SM2 bit in the SCON register (see Figure 9-2 on page 9-3). When the multiprocessor communication feature is enabled, the serial port can differentiate between data frames (ninth bit clear) and address frames (ninth bit set). This allows the microcontroller to function as a slave processor in an environment where multiple slave processors share a single serial line.

When the multiprocessor communication feature is enabled, the receiver ignores frames with the ninth bit clear. The receiver examines frames with the ninth bit set for an address match. If the received address matches the slave's address, the receiver hardware sets the RB8 bit and the RI bit in the SCON register, generating an interrupt.

NOTE

> The ES bit must be set in the IE register to allow the RI bit to generate an interrupt. The IE register is described in Chapter 8, Interrupts.

The addressed slave's software then clears the SM2 bit in the SCON register and prepares to receive the data bytes. The other slaves are unaffected by these data bytes because they are waiting to respond to their own addresses.

9.5 AUTOMATIC ADDRESS RECOGNITION

The automatic address recognition feature is enabled when the multiprocessor communication feature is enabled (the SM2 bit is set in the SCON register).

Implemented in hardware, automatic address recognition enhances the multiprocessor communication feature by allowing the serial port to examine the address of each incoming command frame. Only when the serial port recognizes its own address does the receiver set the RI bit in the SCON register to generate an interrupt. This ensures that the CPU is not interrupted by command frames addressed to other devices.

If desired, you may enable the automatic address recognition feature in mode 1. In this configuration, the stop bit takes the place of the ninth data bit. The RI bit is set only when the received command frame address matches the device's address and is terminated by a valid stop bit.

NOTE

> The multiprocessor communication and automatic address recognition features cannot be enabled in mode 0 (i.e., setting the SM2 bit in the SCON register in mode 0 has no effect).

To support automatic address recognition, a device is identified by a *given* address and a *broadcast* address.

9.5.1 Given Address

Each device has an *individual* address that is specified in the SADDR register; the SADEN register is a mask byte that contains don't-care bits (defined by zeros) to form the device's *given* address. These don't-care bits provide the flexibility to address one or more slaves at a time. The following example illustrates how a given address is formed. (To address a device by its individual address, the SADEN mask byte must be 1111 1111.)

SADDR = 0101 0110
SADEN = 1111 1100
Given = 0101 01XX

The following is an example of how to use given addresses to address different slaves:

Slave A: SADDR = 1111 0001
SADEN = 1111 1010
Given = 1111 0X0X

Slave B: SADDR = 1111 0011
SADEN = 1111 1001
Given = 1111 0XX1

Slave C: SADDR = 1111 0010
SADEN = 1111 1101
Given = 1111 00X1

The SADEN byte is selected so that each slave may be addressed separately. For Slave A, bit 0 (the LSB) is a don't-care bit; for Slaves B and C, bit 0 is a 1. To communicate with Slave A only, the master must send an address where bit 0 is clear (e.g., 1111 0000).

For Slave A, bit 1 is a 0; for Slaves B and C, bit 1 is a don’t care bit. To communicate with Slaves B and C, but not Slave A, the master must send an address with bits 0 and 1 both set (e.g., 1111 0011).

For Slaves A and B, bit 2 is a don’t care bit; for Slave C, bit 2 is a 0. To communicate with Slaves A and B, but not Slave C, the master must send an address with bit 0 set, bit 1 clear, and bit 2 set (e.g., 1111 0101).

To communicate with Slaves A, B, and C, the master must send an address with bit 0 set, bit 1 clear, and bit 2 clear (e.g., 1111 0001).

9.5.2 Broadcast Address

A *broadcast* address is formed from the logical OR of the SADDR and SADEN registers with zeros defined as don't-care bits, e.g.:

SADDR = 0101 0110
SADEN = 1111 1100
(SADDR) OR (SADEN) = 1111 111X

The use of don't-care bits provides flexibility in defining the broadcast address, however, in most applications, a broadcast address is 0FFH.

The following is an example of using broadcast addresses:

Slave A: SADDR = 1111 0001
SADEN = 1111 1010
Broadcast = 1111 1X11

Slave B: SADDR = 1111 0011
SADEN = 1111 1001
Broadcast = 1111 1X11

Slave C: SADDR = 1111 0010
SADEN = 1111 1101
Broadcast = 1111 1111

For Slaves A and B, bit 2 is a don’t care bit; for Slave C, bit 2 is set. To communicate with all of the slaves, the master must send an address FFH.

To communicate with Slaves A and B, but not Slave C, the master can send an address FBH.

9.5.3 Reset Addresses

On reset, the SADDR and SADEN registers are initialized to 00H, i.e., the given and broadcast addresses are XXXX XXXX (all don't-care bits). This ensures that the serial port is backwards compatible with MCS® 51 microcontrollers that do not support automatic address recognition.

9.6 BAUD RATES

You must select the baud rate for the serial port transmitter and receiver when operating in modes 1, 2, and 3. (The baud rate is preset for mode 0.) In its asynchronous modes, the serial port can transmit and receive simultaneously. Depending on the mode, the transmission and reception rates can be the same or different. Table 9-3 summarizes the baud rates that can be used for the four serial I/O modes.

Table 9-3. Summary of Baud Rates

Mode	No. of Baud Rates	Send and Receive at the Same Rate	Send and Receive at Different Rates
0	1	N/A	N/A
1	Many†	Yes	Yes
2	2	Yes	
3	Many†	Yes	Yes

† Baud rates are determined by overflow of timer 1 and/or timer 2.

9.6.1 Baud Rate for Mode 0

The baud rate for mode 0 is fixed at $F_{OSC}/12$.

9.6.2 Baud Rates for Mode 2

Mode 2 has two baud rates, which are selected by the SMOD1 bit in the PCON register (Figure 11-1 on page 11-2). The following expression defines the baud rate:

$$\text{Serial I/O Mode 2 Baud Rate} = 2^{\text{SMOD1}} \times \frac{F_{\text{OSC}}}{64}$$

9.6.3 Baud Rates for Modes 1 and 3

In modes 1 and 3, the baud rate is generated by overflow of timer 1 (default) and/or timer 2. You may select either or both timer(s) to generate the baud rate(s) for the transmitter and/or the receiver.

9.6.3.1 Timer 1 Generated Baud Rates (Modes 1 and 3)

Timer 1 is the default baud rate generator for the transmitter and the receiver in modes 1 and 3. The baud rate is determined by the timer 1 overflow rate and the value of SMOD, as shown in the following formula:

$$\text{Serial I/O Modes 1 and 3 Baud Rate} = 2^{\text{SMOD1}} \times \frac{\text{Timer 1 Overflow Rate}}{32}$$

9.6.3.2 Selecting Timer 1 as the Baud Rate Generator

To select timer 1 as the baud rate generator:

- Disable the timer interrupt by clearing the ETI bit in the IE0 register (Figure 5-2 on page 5-6).
- Configure timer 1 as a timer or an event counter (set or clear the C/T# bit in the TMOD register). The TMOD register is described in Chapter 7, Timers/Counters.
- Select timer mode 0–3 by programming the M1, M0 bits in the TMOD register.

In most applications, timer 1 is configured as a timer in auto-reload mode (high nibble of TMOD = 0010B). The resulting baud rate is defined by the following expression:

$$\text{Serial I/O Modes 1 and 3 Baud Rate} = 2^{\text{SMOD1}} \times \frac{F_{\text{OSC}}}{32 \times 12 \times [256 - (\text{TH1})]}$$

Timer 1 can generate very low baud rates with the following setup:

- Enable the timer 1 interrupt by setting the ET1 bit in the IE register.
- Configure timer 1 to run as a 16-bit timer (high nibble of TMOD = 0001B).
- Use the timer 1 interrupt to initiate a 16-bit software reload.

Table 9-4 lists commonly used baud rates and shows how they are generated by timer 1.

Table 9-4. Timer 1 Generated Baud Rates for Serial I/O Modes 1 and 3

Baud Rate	Oscillator Frequency (F_{OSC})	SMOD1	Timer 1		
			C/T#	Mode	Reload Value
62.5 Kbaud (Max)	12.0 MHz	1	0	2	FFH
19.2 Kbaud	11.059 MHz	1	0	2	FDH
9.6 Kbaud	11.059 MHz	0	0	2	FDH
4.8 Kbaud	11.059 MHz	0	0	2	FAH
2.4 Kbaud	11.059 MHz	0	0	2	F4H
1.2 Kbaud	11.059 MHz	0	0	2	E8H
137.5 Baud	11.986 MHz	0	0	2	1DH
110.0 Baud	6.0 MHz	0	0	2	72H
110.0 Baud	12.0 MHz	0	0	1	FEEBH

9.6.3.3 Timer 2 Generated Baud Rates (Modes 1 and 3)

Timer 2 may be selected as the baud rate generator for the transmitter and/or receiver (Figure 9-5 on page 9-13). The timer 2 baud rate generator mode is similar to the auto-reload mode. A roll-over in the TH2 register reloads registers TH2 and TL2 with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

The timer 2 baud rate is expressed by the following formula:

$$\text{Serial I/O Modes 1 and 3 Baud Rate} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

9.6.3.4 Selecting Timer 2 as the Baud Rate Generator

NOTE

Turn the timer off (clear the TR2 bit in the T2CON register) before accessing registers TH2, TL2, RCAP2H, and RCAP2L.

To select timer 2 as the baud rate generator for the transmitter and/or receiver, program the RCLCK and TCLCK bits in the T2CON register as shown in Table 9-5. (You may select different baud rates for the transmitter and receiver.) Setting RCLK and/or TCLK puts timer 2 into its baud rate generator mode(Figure 9-5). In this mode, a rollover in the TH2 register does not set the TF2 bit in the T2CON register. Also, a high-to-low transition at the T2EX pin sets the EXF2 bit in the T2CON register but does not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). You can use the T2EX pin as an additional external interrupt by setting the EXEN2 bit in T2CON.

You may configure timer 2 as a timer or a counter. In most applications, it is configured for timer operation (i.e., the C/T2# bit is clear in the T2CON register).

Table 9-5. Selecting the Baud Rate Generator(s)

RCLCK Bit	TCLCK Bit	Receiver Baud Rate Generator	Transmitter Baud Rate Generator
0	0	Timer 1	Timer 1
0	1	Timer 1	Timer 2
1	0	Timer 2	Timer 1
1	1	Timer 2	Timer 2

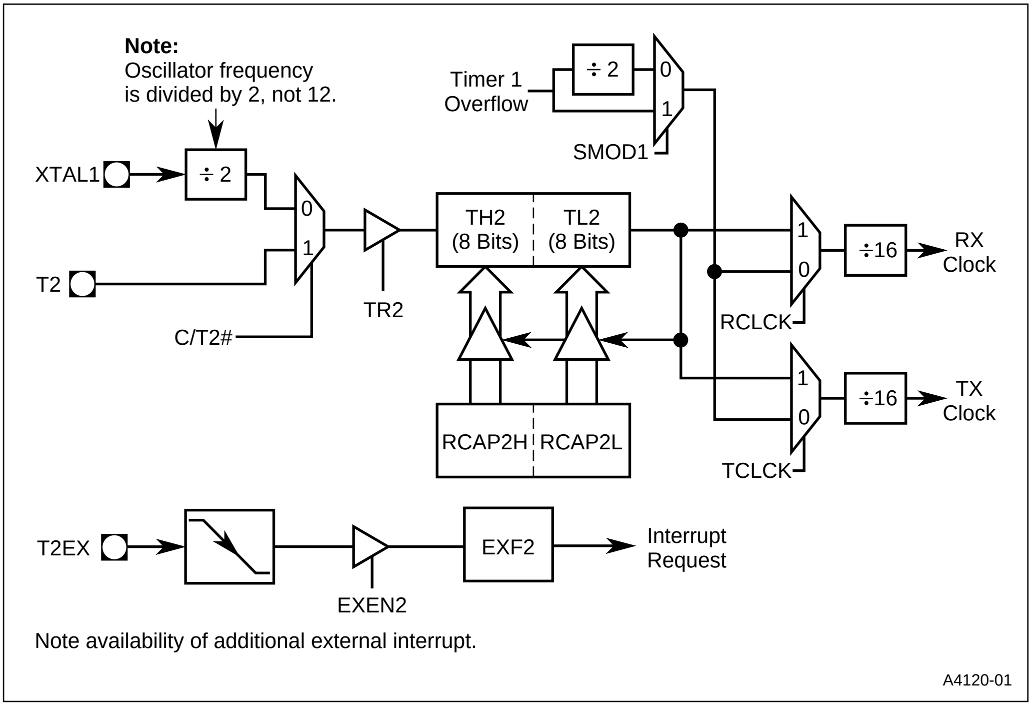

Note:
Oscillator frequency
is divided by 2, not 12.
XTAL1
÷ 2
T2
C/T2#
0
1
TR2
Timer 1
Overflow
÷ 2
0
1
SMOD1
TH2
(8 Bits)
TL2
(8 Bits)
RCAP2H
RCAP2L
1
0
RCLCK
÷16
RX
Clock
1
0
TCLCK
÷16
TX
Clock
T2EX
EXEN2
EXF2
Interrupt
Request
Note availability of additional external interrupt.
A4120-01

Figure 9-5. Timer 2 in Baud Rate Generator Mode

Note that timer 2 increments every state time ($2T_{OSC}$) when it is in the baud rate generator mode. In the baud rate formula that follows, “RCAP2H, RCAP2L” denotes the contents of RCAP2H and RCAP2L taken as a 16-bit unsigned integer:

$$\text{Serial I/O Modes 1 and 3 Baud Rates} = \frac{F_{OSC}}{32 \times [553 - (\text{RCAP2H}, \text{RCAP2L})]}$$

NOTE

> When timer 2 is configured as a timer and is in baud rate generator mode, do not read or write the TH2 or TL2 registers. The timer is being incremented every state time, and the results of a read or write may not be accurate. In addition, you may read, but not write to, the RCAP2 registers; a write may overlap a reload and cause write and/or reload errors.

Table 9-6 lists commonly used baud rates and shows how they are generated by timer 2.

Table 9-6. Timer 2 Generated Baud Rates

Baud Rate	Oscillator Frequency (F_{OSC})	RCAP2H	RCAP2L
375.0 Kbaud	12 MHz	FFH	FFH
9.6 Kbaud	12 MHz	FFH	D9H
4.8 Kbaud	12 MHz	FFH	B2H
2.4 Kbaud	12 MHz	FFH	64H
1.2 Kbaud	12 MHz	FEH	C8H
300.0 baud	12 MHz	FBH	1EH
110.0 baud	12 MHz	F2H	AFH
300.0 baud	6 MHz	FDH	8FH
110.0 baud	6 MHz	F9H	57H

10

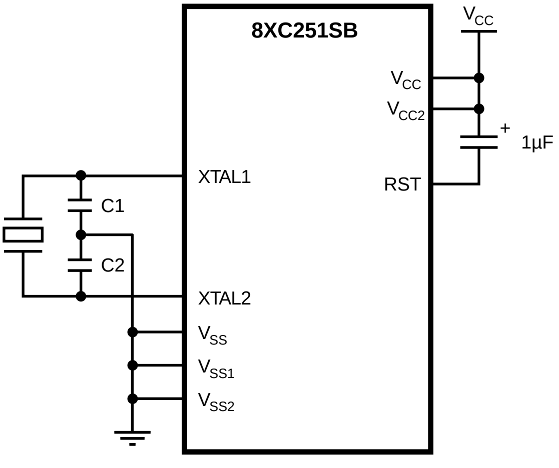
Minimum Hardware Setup

CHAPTER 10
MINIMUM HARDWARE SETUP

This chapter discusses the basic operating requirements of the MCS® 251 microcontroller and describes a minimum hardware setup. Topics covered include power, ground, clock source, and device reset. For parameter values, refer to the device data sheet.

10.1 MINIMUM HARDWARE SETUP

Figure 10-1 shows a minimum hardware setup that employs the on-chip oscillator for the system clock and provides power-on reset. Control signals and Ports 0, 1, 2, and 3 are not shown. See "Clock Sources" on page 10-3 and "Power-on Reset" on page 10-7.



Note:
V_{CC2} is a secondary power pin that reduces power supply noise. V_{SS1} and V_{SS2} are secondary ground pins that reduce ground bounce and improve power supply by-passing. Connections to these pins are not required for proper device operation.

A4141-01

Figure 10-1. Minimum Setup

10.2 ELECTRICAL ENVIRONMENT

The 8XC251SB is a high-speed CHMOS device. To achieve satisfactory performance, its operating environment should accommodate the device signal waveforms without introducing distortion or noise. Design considerations relating to device performance are discussed in this section. See the device data sheet for voltage and current requirements, operating frequency, and waveform timing.

10.2.1 Power and Ground Pins

Power the 8XC251SB from a well-regulated power supply designed for high-speed digital loads. Use short, low impedance connections to the power (V_{CC} and V_{CC2}) and ground (V_{SS}, V_{SS1}, and V_{SS2}) pins.

V_{CC2} is a secondary power pin that reduces power supply noise. V_{SS1} and V_{SS2} are secondary ground pins that reduce ground bounce and improve power supply bypassing. The secondary power and ground pins are not substitutes for V_{CC} and V_{SS}. They are not required for proper device operation; thus, the 8XC251SB is compatible with designs that do not provide connections to these pins.

10.2.2 Unused Pins

To provide stable, predictable performance, connect unused input pins to V_{SS} or V_{CC}. Unterminated input pins can float to a mid-voltage level and draw excessive current. Unterminated interrupt inputs may generate spurious interrupts.

10.2.3 Noise Considerations

The fast rise and fall times of high-speed CHMOS logic may produce noise spikes on the power supply lines and signal outputs. To minimize noise and waveform distortion follow good board layout techniques. Use sufficient decoupling capacitors and transient absorbers to keep noise within acceptable limits. Connect 0.01 µF bypass capacitors between V_{CC} and each V_{SS} pin. Place the capacitors close to the device to minimize path lengths.

Multilayer printed circuit boards with separate V_{CC} and ground planes help minimize noise. For additional information on noise reduction, see Application Note AP-125, “Designing Microcontroller Systems for Noisy Environments.”

10.3 CLOCK SOURCES

The 8XC251SB can obtain the system clock signal from an external clock source (Figure 10-3) or it can generate the clock signal using the on-chip oscillator amplifier and external capacitors and resonator (Figure 10-2).

10.3.1 On-chip Oscillator (Crystal)

This clock source uses an external quartz crystal connected from XTAL1 to XTAL2 as the frequency-determining element (Figure 10-2). The crystal operates in its fundamental mode as an inductive reactance in parallel resonance with capacitance external to the crystal. Oscillator design considerations include crystal specifications, operating temperature range, and parasitic board capacitance. Consult the crystal manufacturer's data sheet for parameter values. With high quality components, C1 = C2 = 30 pF is adequate for this application.

Pins XTAL1 and XTAL2 are protected by on-chip electrostatic discharge (ESD) devices, D1 and D2, which are diodes parasitic to the R_F FETs. They serve as clamps to V_{CC} and V_{SS}. Feedback resistor R_F in the inverter circuit, formed from paralleled n- and p- channel FETs, permits the PD bit in the PCON register (Figure 11-1 on page 11-2) to disable the clock during powerdown.

Noise spikes at XTAL1 and XTAL2 can disrupt microcontroller timing. To minimize coupling between other digital circuits and the oscillator, locate the crystal and the capacitors near the chip and connect to XTAL1, XTAL2, and V_{SS} with short, direct traces. To further reduce the effects of noise, place guard rings around the oscillator circuitry and ground the metal crystal case.

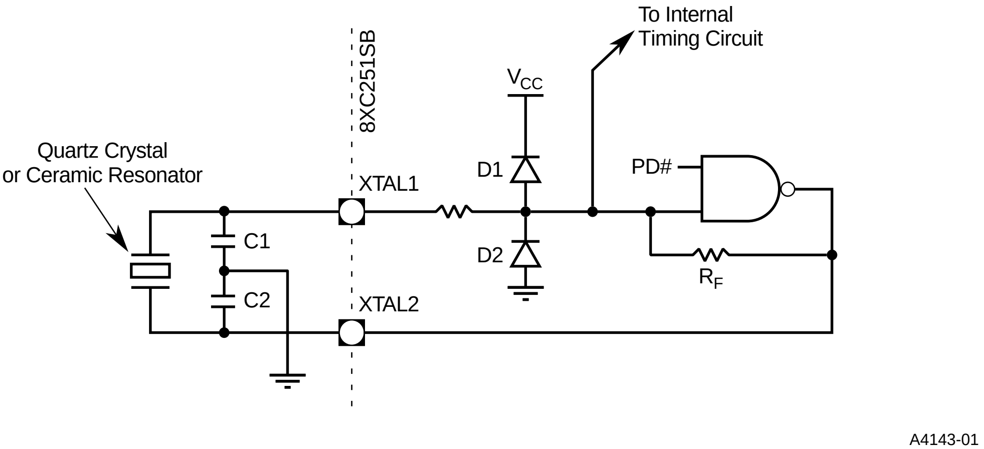


Figure 10-2. CHMOS On-chip Oscillator

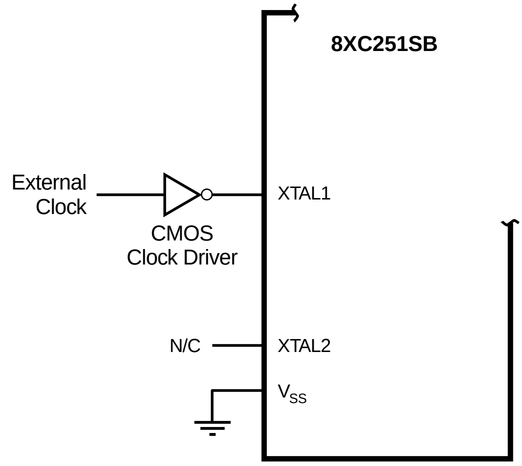
For a more in-depth discussion of crystal specifications, ceramic resonators, and the selection of C1 and C2 see Applications Note AP-155, “Oscillators for Microcontrollers” in the Embedded Applications handbook.

10.3.2 On-chip Oscillator (Ceramic Resonator)

In cost-sensitive applications, you may choose a ceramic resonator instead of a crystal. Ceramic resonator applications may require slightly different capacitor values and circuit configuration. Consult the manufacturer’s data sheet for specific information.

10.3.3 External Clock

To operate the CHMOS 8XC251SB from an external clock, connect the clock source to the XTAL1 pin as shown in Figure 10-3. Leave the XTAL2 pin floating. The external clock driver can be a CMOS gate. If the clock driver is a TTL device, its output must be connected to V_{CC} through a 4.7 kΩ pullup resister.



Note: If TTL clock driver is used, connect a 4.7kΩ pull up resistor from driver output to $V_{CC.}$

A4142-01

Figure 10-3. External Clock Connection

For external clock drive requirements, see the device data sheet. Figure 10-4 shows the clock drive waveform. The external clock source must meet the minimum high and low times (T_{CHCX} and T_{CLCX}) and the maximum rise and fall times (T_{CLCH} and T_{CHCL}) to minimize the effect of external noise on the clock generator circuit. Long rise and fall times increase the chance that external noise will affect the clock circuitry and cause unreliable operation.

The external clock driver may encounter increased capacitance loading at XTAL1 due to the Miller effect of the internal inverter as the clock waveform builds up in amplitude following power on. Once the input waveform requirements are met, the input capacitance remains under 20 pF.

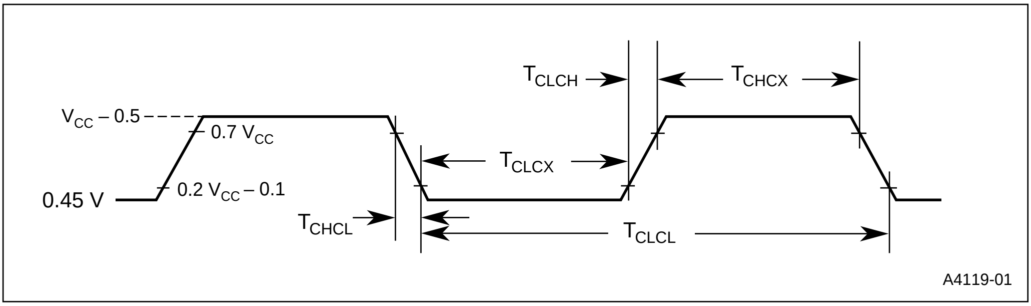


Figure 10-4. External Clock Drive Waveforms

10.4 RESET

A device reset initializes the 8XC251SB and vectors the CPU to address FF:0000H. A reset is required after applying power at turn-on. A reset is a means of exiting the idle and powerdown modes or recovering from software malfunctions.

To achieve a valid reset, V_{CC} must be within its normal operating range (see device data sheet) and the reset signal must be maintained for 64 clock cycles ($64T_{OSC}$) after the oscillator has stabilized.

Device reset is initiated in two ways:

- externally, by asserting the RST pin
- internally, if the hardware WDT or the PCA WDT expires

The power off flag (POF) in the PCON register indicates whether a reset is a warm start or a cold start. A cold start reset (POF = 1) is a reset that occurs after power has been off or V_{CC} has fallen below 3 V, so the contents of volatile memory are indeterminate. POF is set by hardware when V_{CC} rises from less than 3V to its normal operating level. See “Power Off Flag” on page 11-1. A warm start reset (POF = 0) is a reset that occurs while the chip is at operating voltage, for example, a reset initiated by a WDT overflow or an external reset used to terminate the idle or powerdown modes.

10.4.1 Externally Initiated Resets

To reset the 8XC251SB, hold the RST pin at a logic high for at least 64 clock cycles ($64T_{OSC}$) while the oscillator is running. Reset can be accomplished automatically at the time power is applied by capacitively coupling RST to V_{CC} (see Figure 10-1 and “Power-on Reset” on page 10-7). The RST pin has a Schmitt trigger input and a pulldown resistor.

10.4.2 WDT Initiated Resets

Expiration of the hardware WDT (overflow) or the PCA WDT (comparison match) generates a reset signal. WDT initiated resets have the same effect as an external reset. See “Watchdog Timer” on page 7-16 and “PCA Watchdog Timer Mode” on page 8-9.

10.4.3 Reset Operation

When a reset is initiated, whether externally or by a WDT, the port pins are immediately forced to their reset condition as a fail-safe precaution, whether the clock is running or not.

The external reset signal and the WDT initiated reset signals are combined internally. For an external reset the voltage on the RST pin must be held high for $64T_{OSC}$. For WDT initiated resets, a 5-bit counter in the reset logic maintains the signal for the required $64T_{OSC}$.

The CPU checks for the presence of the combined reset signal every $2T_{OSC}$. When a reset is detected, the CPU responds by triggering the internal reset routine. The reset routine loads the SFR’s with their reset values (see Table 3-4 on page 3-13). Reset does not affect on-chip data RAM or the register file. (However following a cold start reset, these are indeterminate because V_{CC} has fallen too low or has been off.) Following a synchronizing operation and the configuration fetch, the CPU vectors to address FF:0000. Figure 10-5 shows the reset timing sequence.

While the RST pin is high ALE, PSEN#, and the port pins are weakly pulled high. The first ALE occurs $32T_{OSC}$ after the reset signal goes low. For this reason, other devices can not be synchronized to the internal timings of the 8XC251SB.

NOTE

> Externally driving the ALE and/or PSEN# pins to 0 during the reset routine may cause the device to go into an indeterminate state.
>
> Powering up the 8XC251SB without a reset may improperly initialize the program counter and SFRs and cause the CPU to execute instructions from an undetermined memory location.

10.4.4 Power-on Reset

To automatically generate a reset on power up, connect the RST pin to the V_{CC} pin through a 1-µF capacitor as shown in Figure 10-1.

When V_{CC} is applied, the RST pin rises to V_{CC}, then decays exponentially as the capacitor charges. The time constant must be such that RST remains high (above the turn-off threshold of the Schmitt trigger) long enough for the oscillator to start and stabilize, plus $64T_{OSC}$. At power up, V_{CC} should rise within approximately 10 ms. Oscillator start-up time is a function the crystal frequency; typical start-up times are 1 ms for a 10 MHz crystal and 10 ms for a 1 Mhz crystal.

During power up, the port pins are in a random state until forced to their reset state by the asynchronous logic.

Reducing V_{CC} quickly to 0 causes the RST pin voltage to momentarily fall below 0 V. This voltage is internally limited and does not harm the device.

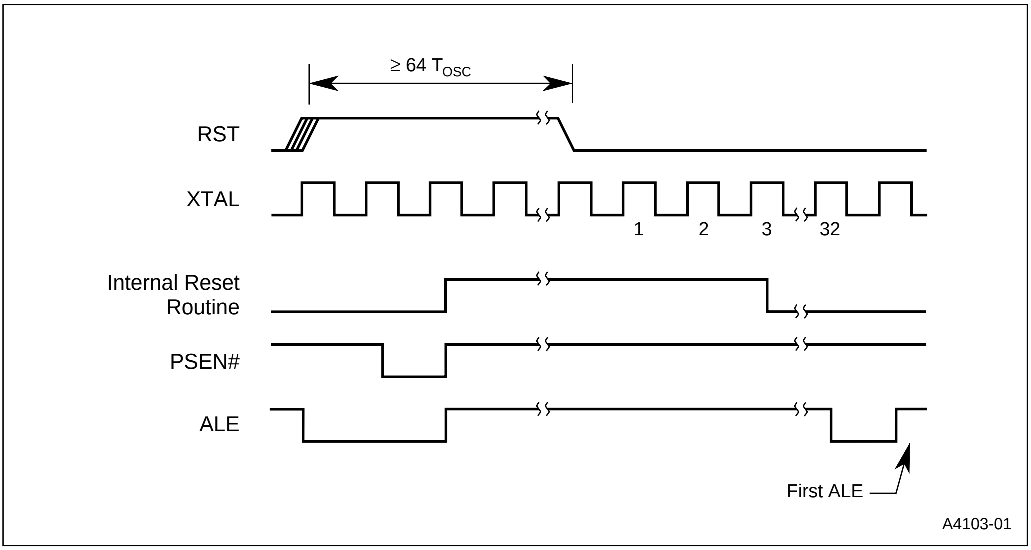

≥ 64 T_{OSC}
RST
XTAL
1
2
3
32
Internal Reset Routine
PSEN#
ALE
First ALE
A4103-01

Figure 10-5. Reset Timing Sequence

11

Special Operating Modes

CHAPTER 11
SPECIAL OPERATING MODES

This chapter describes the power control (PCON) register and three special operating modes: idle, powerdown, and on-circuit emulation (ONCE).

11.1 GENERAL

The idle and powerdown modes are power reduction modes for use in applications where power consumption is a concern. User instructions activate these modes by setting bits in the PCON register. Program execution halts, but resumes when the mode is exited by an interrupt. While in idle or power-down, the V_{CC} pin is the input for backup power.

ONCE is a test mode that electrically isolates the 8XC251SB from the system in which it operates.

11.2 POWER CONTROL REGISTER

The PCON special function register (Figure 11-1) provides two control bits for the serial I/O function, bits for selecting the idle and powerdown modes, the power off flag, and two general purpose flags.

11.2.1 Serial I/O Control Bits

The SMOD1 bit in the PCON register is a factor in determining the serial I/O baud rate. See Figure 11-1 and “Baud Rates” on page 9-10.

The SMOD0 bit in the PCON register determines whether bit 7 of the SCON register provides read/write access to the framing error (FE) bit (SMOD0 = 1) or to SM0, a serial I/O mode select bit (SMOD0 = 0). See Figure 11-1 and Figure 9-2, “Serial Port Special Function Register” on page 9-3.

11.2.2 Power Off Flag

Hardware sets the Power Off Flag (POF) in PCON when V_{CC} rises from < 3 V to > 3 V to indicate that on-chip volatile memory is indeterminate, e.g., at power on. The POF can be set or cleared by software. In general after a reset, check the status of this bit to determine whether a cold start reset or a warm start reset occurred (see “Reset” on page 10-5). After a cold start, user software should clear the POF. If POF = 1 is detected at other times, do a reset to reinitialize the chip, since for $V_{CC} < 3$ V data may have been lost or some logic may have malfunctioned.

PCON

Address: S:87H
Reset State: 00xx 0000B

7							0
SMOD1	SMOD0	—	POF	GF1	GF0	PD	IDL

Bit Number	Bit Mnemonic	Function
7	SMOD1	Double Baud Rate Bit: When set, doubles the baud rate when timer 1 is used and mode 1, 2, or 3 is selected in the SCON register. See “Baud Rates” on page 9-10.
6	SMOD0	SCON.7 Select: When set, read/write accesses to SCON.7 are to the FE bit. When clear, read/write accesses to SCON.7 are to the SM0 bit. See Figure 9-2 on page 9-3.
5	—	Reserved: The value read from this bit is indeterminate. Do not write a “1” to this bit.
4	POF	Power Off Flag: Set by hardware as V_{CC} rises above 3 V to indicate that power has been off or V_{CC} had fallen below 3 V and that on-chip volatile memory is indeterminate. Set or cleared by software.
3	GF1	General Purpose Flag: Set or cleared by software. One use is to indicate whether an interrupt occurred during normal operation or during idle mode.
2	GF0	General Purpose Flag: Set or cleared by software. One use is to indicate whether an interrupt occurred during normal operation or during idle mode.
1	PD	Powerdown Mode Bit: When set, activates powerdown mode. Cleared by hardware when an interrupt or reset occurs.
0	IDL	Idle Mode Bit: When set, activates idle mode. Cleared by hardware when an interrupt or reset occurs. If IDL and PD are both set, PD takes precedence.

Figure 11-1. Power Control (PCON) Register

Table 11-1. Pin Conditions in Various Modes

Mode	Program Memory	ALE Pin	PSEN# Pin	Port 0 Pins	Port 1 Pins	Port 2 Pins	Port 3 Pins
Reset	Don't Care	Weak High	Weak High	Floating	Weak High	Weak High	Weak High
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Floating	Data	Data	Data
Powerdown	Internal	0	0	Data	Data	Data	Data
Powerdown	External	0	0	Floating	Data	Data	Data
ONCE	Don't Care	Floating	Floating	Floating	Weak High	Weak High	Weak High

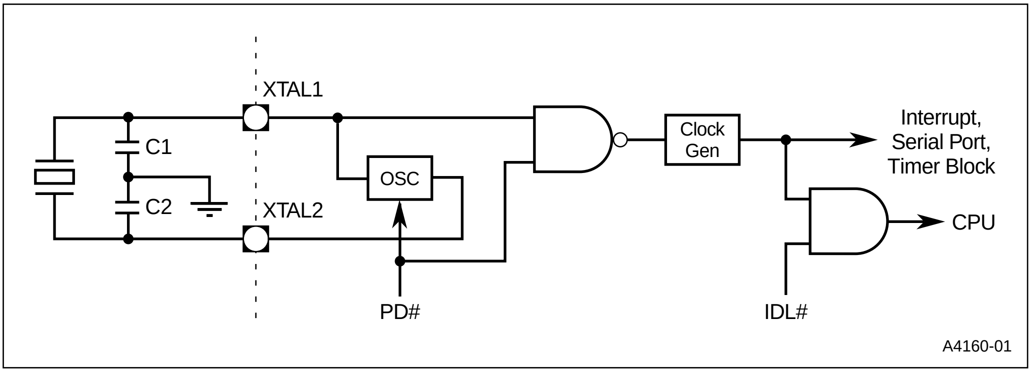

XTAL1
C1
C2
XTAL2
OSC
PD#
Clock Gen
Interrupt, Serial Port, Timer Block
CPU
IDL#
A4160-01

Figure 11-2. Idle and Powerdown Clock Control

11.3 IDLE MODE

Idle mode is a power reduction mode that reduces power consumption to about 40% of normal. In this mode, program execution halts. Idle mode freezes the clocks to the CPU at known states while the peripherals continue to be clocked (Figure 11-2). The CPU status before entering idle mode is preserved, i.e., the program counter, program status word register, and register file retain their data for the duration of idle mode. The contents of the SFRs and RAM are also retained. The status of the port pins depends upon the location of the program memory:

- Internal program memory: the ALE and PSEN# pins are pulled high and the ports 0, 1, 2, and 3 pins are reading data (Table 11-1).
- External program memory: the ALE and PSEN# pins are pulled high; the port 0 pins are floating; and the pins of ports 1, 2, and 3 are reading data (Table 11-1).

NOTE

If desired, the PCA may be instructed to pause during idle mode by setting the CIDL bit in the CMOD register (Figure 8-7 on page 8-13).

11.3.1 Entering Idle Mode

To enter idle mode, set the PCON register IDL bit. The 8XC251SB enters idle mode upon execution of the instruction that sets the IDL bit. The instruction that sets the IDL bit is the last instruction executed.

CAUTION

If the IDL bit and the PD bit are set simultaneously, the 8XC251SB enters powerdown mode

11.3.2 Exiting Idle Mode

There are two ways to exit idle mode:

- Generate an enabled interrupt. Hardware clears the PCON register IDL bit which restores the clocks to the CPU. Execution resumes with the interrupt service routine. Upon completion of the interrupt service routine, program execution resumes with the instruction immediately following the instruction that activated idle mode. The general purpose flags (GF1 and GF0 in the PCON register) may be used to indicate whether an interrupt occurred during normal operation or during idle mode. When idle mode is exited by an interrupt, the interrupt service routine may examine GF1 and GF0.
- Reset the chip. See “Reset” on page 10-5. A logic high on the RST pin clears the IDL bit in the PCON register directly and asynchronously. This restores the clocks to the CPU. Program execution momentarily resumes with the instruction immediately following the instruction that activated the idle mode and may continue for a number of clock cycles before the internal reset algorithm takes control. Reset initializes the 8XC251SB and vectors the CPU to address FF:0000H.

NOTE

During the time that execution resumes, the internal RAM cannot be accessed; however, it is possible for the port pins to be accessed. To avoid unexpected outputs at the port pins, the instruction immediately following the instruction that activated idle mode should not write to a port pin or to the external RAM.

11.4 POWERDOWN MODE

The powerdown mode places the 8XC251SB in a very low power state. Powerdown mode stops the oscillator and freezes all clocks at known states (Figure 11-2). The CPU status prior to entering powerdown mode is preserved, i.e., the program counter, program status word register, and register file retain their data for the duration of powerdown mode. In addition, the SFRs and RAM contents are preserved. The status of the port pins depends on the location of the program memory:

- Internal program memory: the ALE and PSEN# pins are pulled low and the ports 0, 1, 2, and 3 pins are reading data (Table 11-1).
- External program memory: the ALE and PSEN# pins are pulled low; the port 0 pins are floating; and the pins of ports 1, 2, and 3 are reading data (Table 11-1).

NOTE

V_{CC} may be reduced to as low as 2 V during powerdown to further reduce power dissipation. Take care, however, that V_{CC} is not reduced until powerdown is invoked.

11.4.1 Entering Powerdown Mode

To enter powerdown mode, set the PCON register PD bit. The 8XC251SB enters the power-down mode upon execution of the instruction that sets the PD bit. The instruction that sets the PD bit is the last instruction executed.

11.4.2 Exiting Powerdown Mode

CAUTION

> If V_{CC} was reduced during the powerdown mode, do not exit powerdown until V_{CC} is restored to the normal operating level.

There are two ways to exit the powerdown mode:

- Generate an enabled external interrupt. Hardware clears the PD bit in the PCON register which starts the oscillator and restores the clocks to the CPU and peripherals. Execution resumes with the interrupt service routine. Upon completion of the interrupt service routine, program execution resumes with the instruction immediately following the instruction that activated powerdown mode.

NOTE

> To enable an external interrupt, set the IE register EX0 and/or EX1 bit[s]. The external interrupt used to exit powerdown mode must be configured as level sensitive and must be assigned the highest priority. In addition, the duration of the interrupt must be of sufficient length to allow the oscillator to stabilize.

- Generate a reset. See “Reset” on page 10-5. A logic high on the RST pin clears the PD bit in the PCON register directly and asynchronously. This starts the oscillator and restores the clocks to the CPU and peripherals. Program execution momentarily resumes with the instruction immediately following the instruction that activated powerdown and may continue for a number of clock cycles before the internal reset algorithm takes control. Reset initializes the 8XC251SB and vectors the CPU to address FF:0000H.

NOTE

> During the time that execution resumes, the internal RAM cannot be accessed; however, it is possible for the port pins to be accessed. To avoid unexpected outputs at the port pins, the instruction immediately following the instruction that activated the powerdown mode should not write to a port pin or to the external RAM.

11.5 ON-CIRCUIT EMULATION (ONCE) MODE

The on-circuit emulation (ONCE) mode permits external testers to test and debug 8XC251SB-based systems without removing the chip from the circuit board. A clamp-on emulator or test CPU is used in place of the 8XC251SB which is electrically isolated from the system.

11.5.1 Entering ONCE Mode

To enter the ONCE mode:

1. Assert RST to initiate a device reset. See “Externally Initiated Resets” on page 10-6 and the reset waveforms in Figure 10-5 on page 10-8.
2. While holding RST asserted, apply and hold logic levels to I/O pins as follows: PSEN# = low, P0.7:5 = low, P0.4 = high, P0.3:0 = low (i.e., port 0 = 10H).
3. Deassert RST, then remove the logic levels from PSEN# and port 0.

These actions cause the 8XC251SB to enter the ONCE mode. Port 1, 2, and 3 pins are weakly pulled high and port 0, ALE, and PSEN# pins are floating (Table 11-1). Thus the device is electrically isolated from the remainder of the system which can then be tested by an emulator or test CPU. Note that in the ONCE mode the device oscillator remains active.

11.5.2 Exiting ONCE Mode

To exit ONCE mode, reset the device.

12

External Memory Interface

CHAPTER 12
EXTERNAL MEMORY INTERFACE

The external memory interface comprises the external bus (ports 0 and 2) and the bus control signals. Chip configuration bytes determine several interface options: page mode or nonpage mode for external code fetches, the number of external address bits (16 or 17), the address ranges for PSEN# and RD#, and external wait states. You can use these options to tailor the interface to your application. This chapter describes the external memory interface, its configuration, and the external bus cycles. Examples illustrate several types of external memory designs.

12.1 EXTERNAL MEMORY INTERFACE SIGNALS

Table 12-1 describes the external memory interface signals. The address and data signals (AD7:0 on port 0 and A15:8 on port 2) are defined for nonpage mode. Address bits A7:0 are multiplexed with the data (D7:0) on port 0, and address bits A15:8 are on port 2. In page mode, address bits A7:0 are on port 0, and address bits A15:8 are multiplexed with the data (D7:0) on port 2 (see "Page Mode Bus Cycles" on page 12-10).

Table 12-1. External Memory Interface Signals

Signal Name	Type	Description	Multiplexed With
A16	O	**Address Line 16**. See RD#.	N.A.
A15:8†	O	**Address Lines**. Upper address lines for the external bus.	P2.7:0
AD7:0†	I/O	**Address/Data Lines**. Multiplexed lower address lines and data lines for the external bus.	P0.7:0
ALE	O	**Address Latch Enable**. ALE signals the start of an external bus cycle and indicates that valid address information is available on lines A15:8 and AD7:0. An external latch can use ALE to demultiplex the address from the address/data bus.	PROG#
EA#	I	**External Access**. Directs program memory accesses to on-chip or off-chip code memory. For EA# strapped to ground, all program memory accesses are off-chip. For EA# = strapped to V_{CC}, an access is to on-chip OTPROM/ROM if the address is within the range of the on-chip OTPROM/ROM; otherwise the access is off-chip. The value of EA# is latched at reset. For a ROMless part, EA# must be strapped to ground.	V_{PP}

†The descriptions of A15:8/P2.7:0 and AD7:0/P0.7:0 are for the nonpage-mode chip configuration (compatible with 44-pin PLCC MCS® 51 microcontrollers). If the chip is configured for page-mode operation, port 0 carries the lower address bits (A7:0), and port 2 carries the upper address bits (A15:8) and the data (D7:0).

Table 12-1. External Memory Interface Signals (Continued)

Signal Name	Type	Description	Multiplexed With
PSEN#	O	**Program Store Enable**. Read signal output. This output is asserted for a memory address range that depends on bits RD0 and RD1 in configuration byte CONFIG1 (see also RD#): **RD1 RD0 Address Range for Assertion** 0 0 Reserved 0 1 All addresses 1 0 All addresses 1 1 All addresses ≥ 80:0000H	—
RD#	O	**Read or 17th Address Bit (A16)**. Read signal output to external data memory or 17th external address bit (A16), depending on the values of bits RD0 and RD1 in configuration byte CONFIG1. (See also PSEN#): **RD1 RD0 Function** 0 0 Reserved 0 1 The pin functions as A16 only. 1 0 The pin functions as P3.7 only. 1 1 RD#: asserted for reads at all addresses £ 7F:FFFFH	P3.7
WR#	O	**Write**. Write signal output to external memory. For configuration bits RD1 = RD0 = 1, WR# is strobed only for writes to locations 00 0000H–01 FFFFH. For other values of RD1 and RD0, WR# is strobed for writes to all memory locations.	P3.6

†The descriptions of A15:8/P2.7:0 and AD7:0/P0.7:0 are for the nonpage-mode chip configuration (compatible with 44-pin PLCC MCS® 51 microcontrollers). If the chip is configured for page-mode operation, port 0 carries the lower address bits (A7:0), and port 2 carries the upper address bits (A15:8) and the data (D7:0).

12.2 CONFIGURING THE EXTERNAL MEMORY INTERFACE

This section describes the configuration options that affect the external memory interface. ("Configuration Bytes" on page 13-6 describes the configuration bytes.) The configuration bits described here determine the following interface features:

- page mode or nonpage mode
- the number of external address pins (16 or 17)
- the memory regions assigned to the read signals RD# and PSEN#
- the external wait states
- mapping a portion of on-chip code memory to data memory

12.2.1 Page Mode and Nonpage Mode (PAGE Bit)

The PAGE bit (bit 1 in CONFIG0) selects page-mode or nonpage-mode code fetches and determines the structure of the external bus. See “Page Mode Bus Cycles” on page 12-10 for a description of page mode and the bus structure.

- PAGE = 1. The 8XC251SB operates in nonpage mode. The bus structure is the same as for the MCS 51 architecture, and external code fetches require two state times ($4T_{OSC}$).
- PAGE = 0. The 8XC251SB operates in page mode. The bus structure is different from the bus structure in MCS 51 controllers, and under certain conditions, external code fetches require only one state time ($2T_{OSC}$).

12.2.2 RD#, PSEN#, and the Number of External Address Pins (Bits RD1:0)

The RD1:0 configuration bits (bits 2 and 3 in CONFIG0) determine the number of external address lines and the address ranges for strobing the read signals PSEN# and RD#. These selections offer different ways of addressing external memory.

A key to using the memory interface is the relationship between internal memory addresses and external memory addresses. While the 8XC251SB has 24 internal address bits, it has only 16 external address pins, A15:0 on ports 0 and 2. Therefore, internal addresses that differ only in their upper eight bits are indistinguishable at the external address pins. For example, if you write to location 00:6000H and location 01:6000H, the same address (6000H) appears at the external address pins. The 16 pins can address only 64 Kbytes of external memory. The options provided by bits RD1:0, offer ways to expand the external memory space beyond 64 Kbytes.

Table 12-2 describes how RD# and PSEN# function for the values of RD1:0. RD# can function as a read signal, as a general-purpose I/O signal, or as the seventeenth external address bit A16. PSEN# always functions as a read signal, and in two cases PSEN# is a read strobe for data memory as well as code memory. For a design that is compatible with MCS 51 microcontrollers, select RD1 = 1 and RD0 = 1.

Table 12-2. Configuration Bits RD1:0

RD1	RD0	External Address Bits	RD#	PSEN#
0	0	—	Reserved	
0	1	17	RD# is the 17th address bit (A16).	PSEN# is strobed for all addresses.
1	0	16	RD# is a general-purpose I/O signal (P3.7).	PSEN# is strobed for all addresses.
1	1	16	RD# is strobed for locations 00:0000H–7F:FFFFH.	PSEN# is strobed for locations 80:0000H–FF:FFFFH.

12.2.2.1 Sixteen External Address Bits and a Single Read Signal (RD1 = 1, RD0 = 0)

For RD1 = 1 and RD0 = 0, PSEN# is strobed for all external reads, and pin RD#/P3.7 is devoted exclusively to general-purpose I/O, i.e., it does not function as RD#. With this configuration you can address the minimum amount of external memory (64 Kbytes), but you gain an extra I/O channel (P3.7). Figure 12-1 illustrates the difference between the internal and external memory spaces for these values of RD1:0. Regions 00:, 01:, FE:, and FF: of internal memory are mapped into a single 64-Kbyte region of external memory. This selection of RD1:0 can be used, for example, in a design where the 87C251SB/83C251SB executes from on-chip code memory and accesses 64 Kbytes of external RAM.

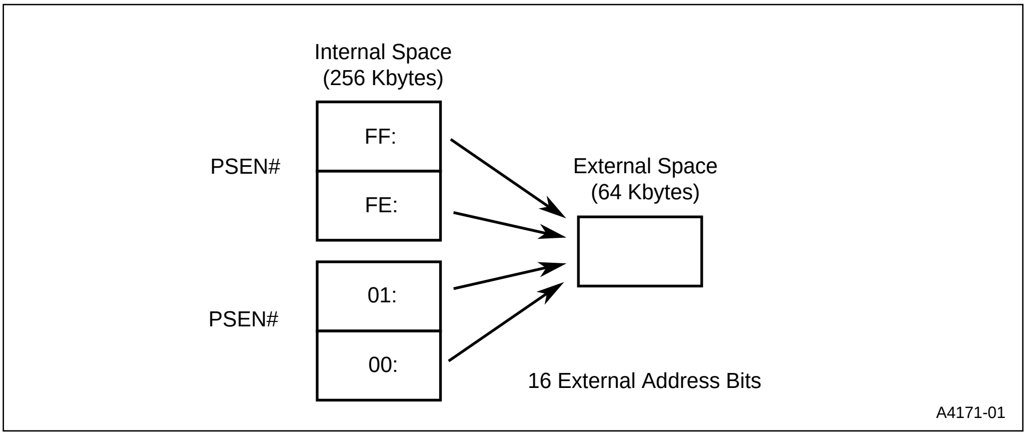


Figure 12-1. Internal and External Memory Spaces for RD1 = 1, RD0 = 0

12.2.2.2 Seventeen External Address Bits and a Single Read Signal (RD1 = 0, RD0 = 1)

For RD1 = 0 and RD0 = 1, the RD# signal becomes the seventeenth external address bit (A16) and PSEN# is strobed for all external reads. The 17 external address bits can address 128 Kbytes of external memory. As illustrated in Figure 12-2, internal memory regions 00: and FE: are mapped into external memory region 0, and internal memory regions 01: and FF: are mapped into external memory region 1. This option provides supports three basic designs:

- 128 Kbytes of external code memory (addressed as regions FE: and FF:)
- 128 Kbytes of external data memory (addressed as regions 00: and 01:)
- 64 Kbytes of external code memory (addressed as region FF:) and 64Kbytes of external data memory (addressed as region 00:).

Sections 12.6.2 and 12.6.5 show examples of memory designs with this option.

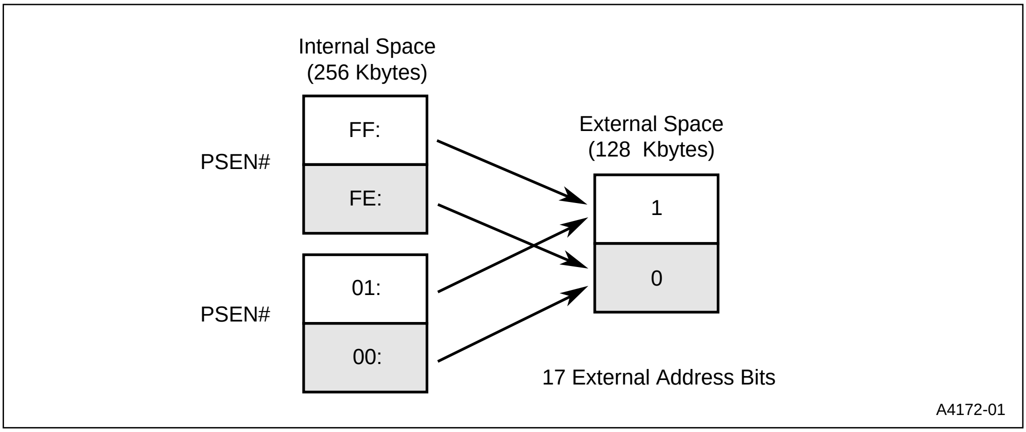

Internal Space
(256 Kbytes)
PSEN#
FF:
FE:
PSEN#
01:
00:
External Space
(128 Kbytes)
1
0
17 External Address Bits
A4172-01

Figure 12-2. Internal and External Memory Spaces for RD1 = 0, RD0 = 1

12.2.2.3 Sixteen External Address Bits and Two Read Signals (RD1 = 1, RD0 = 1)

For RD1 = 1 and RD0 = 1, there are 16 external address bits; however, RD# is strobed for regions 00: and 01:, and PSEN# is strobed for regions FE: and FF:. As illustrated in Figure 12-3, regions 00: and 01: are mapped into 64 Kbytes of data memory (strobed by RD#), and regions FE: and FF: are mapped into 64 Kbytes of code memory (strobed by PSEN#). This selection is compatible with MCS 51 microcontrollers and supports designs that use both external code memory and external data memory.

For this selection of RD1:0, WR# is strobed for writes to regions 00: and 01: but is **not** strobed for writes to regions FE: and FF:. This is compatible with MCS 51 microcontrollers, which cannot write to external code memory. Sections 12.6.1 and 12.6.4 show examples of memory designs with this option.

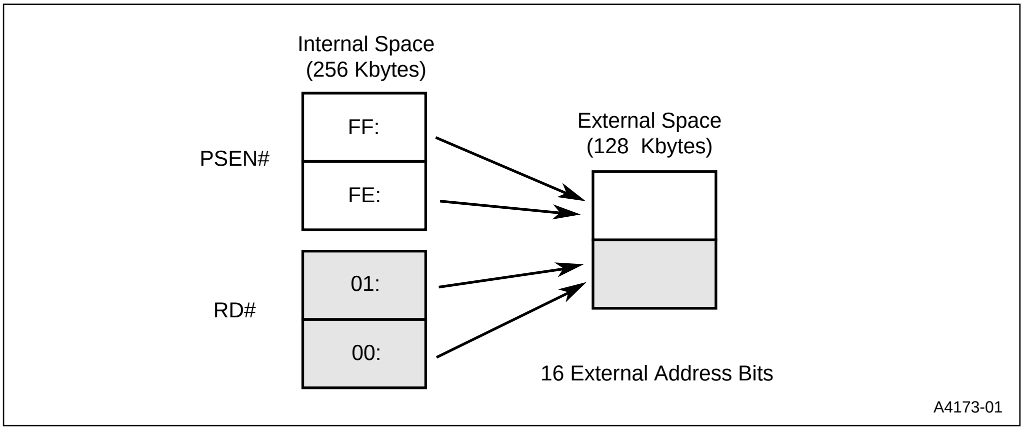


Figure 12-3. Internal and External Memory Spaces for RD1 = 1, RD0 = 1

12.2.3 Wait States (WSA, WSB, XALE)

You can add wait states to external bus cycles by extending the PSEN#/RD#/WR# pulse and/or extending the ALE pulse:

- The WSA bit (bit 5 in CONFIG0) and the WSB bit (bit 3 in CONFIG1) specify the wait states (0 or 1) added by extending the time that PSEN#/RD#/WR# is asserted from T_{OSC} to $3T_{OSC}$. This wait state accommodates slower external devices and allows the 8XC251SB to directly replace the 8XC51FB in a system design. The combinations of WSA and WSB select the memory regions to be accessed with one wait state (Table 12-3). The option of a wait state for region 01: is for accessing a slow external device addressed in region 01: without slowing down accesses to other external devices. “Extending PSEN#/RD#/WR#” on page 12-13 shows bus cycles with PSEN#/RD# extended and WR# extended.

Table 12-3. Wait State Selection

WSB	WSA	Memory Regions with 1 Wait State
0	0	All regions (00:, 01:, FE:, FF:)
0	1	Region 01:
1	0	Regions 00:, FE:, FF:
1	1	None

- Clearing XALE (bit 4 in CONFIG0) extends the time ALE is asserted from T_{OSC} to $3T_{OSC}$. This accommodates an address latch that is too slow for the normal ALE signal. “Extending ALE” on page 12-14 shows a bus cycle with ALE extended.

You can add two wait states by extending both ALE and the read/write signals (PSEN#, RD#, WR#).

12.2.4 Mapping On-chip Code Memory to Data Memory (87C251SB/83C251SB)

For the 87C251SB/83C251SB, the EMAP bit (bit 0 in CONFIG1) provides the option of accessing the upper 8 Kbytes of on-chip code memory as data memory.

EMAP = 0. The upper 8 Kbytes of the on-chip code memory (FF:2000H–FF:3FFFH) are mapped to locations 00:E000H–00:FFFFH (in addition to locations FF:2000H–FF:3FFFH). This allows code constants to be accessed as data in region 00:. See “On-chip Code Memory (87C251SB/83C251SB)” on page 3-6 for the exact conditions required for this mapping to be effective.

EMAP = 1. Locations FF:2000H–FF:3FFFH are not mapped to region 00:. Locations 00:E000H–00:FFFFH are implemented by external RAM.

12.3 EXTERNAL BUS CYCLES

The 8XC251SB executes external bus cycles to fetch code, read data, and write data in external memory. This section uses bus waveforms with idealized timings to describe the external bus cycles in nonpage mode and page mode. The bus cycles in this section have no wait states. (For bus cycles with wait states, see “Wait States” on page 12-13.) Timing parameters for the bus cycles are given in “External Bus AC Timing Specifications” on page 12-24.

“Inactive External Bus” describes the situations where the bus is not executing external bus cycles.

12.3.1 Inactive External Bus

The external bus is inactive (not executing external bus cycles) under any of these three conditions:

- The chip is in normal operating mode but no external read or write cycles are executing (the *bus-idle* condition).
- The chip is in idle mode.
- The chip is in powerdown mode.

12.3.2 Bus Cycle Definitions

Table 12-4 summarizes the activity on the bus for bus cycles in nonpage mode and page mode with no wait states. Nonpage mode has only two types of bus cycles: a code/data read cycle and a write cycle. Page mode has four types of bus cycles: a code-read cycle for a page miss, a code-read cycle for a page hit, a data-read cycle, and a write cycle. The data-read and write cycles are the same for page mode and nonpage mode (except for the different signals on ports 0 and 2).

Table 12-4. Bus Cycle Definitions (No Wait States)

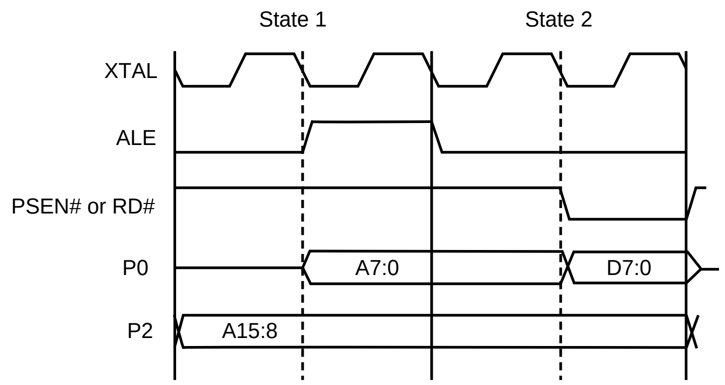
Mode	Bus Cycle	Bus Activity		
		State 1	State 2	State 3
Nonpage Mode	Code/Data Read	ALE Strobe	PSEN#/RD# Strobe	(3)
	Write	ALE Strobe	WR# Strobe	WR# High
Page Mode	Code Read (Page Miss)	ALE Strobe	PSEN#/RD# Strobe	(3)
	Code Read (Page Hit)	PSEN# Strobe	(4)	
	Data Read (1)	ALE Strobe	PSEN#/RD# Strobe	
	Write (2)	ALE Strobe	WR# Strobe	WR# high

NOTES:

1. The code/data read cycle in nonpage mode and the data-read cycle in page mode are the same, except for the different signals on ports 0 and 2.
2. The write cycle is the same in page mode and nonpage mode, except for the difference in bus structure.
3. Only write cycles have a third state.
4. A page hit requires only one state.

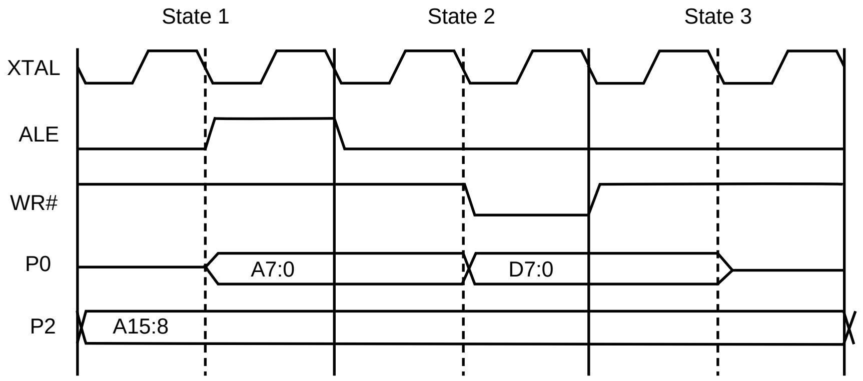
12.3.3 Nonpage Mode Bus Cycles

In nonpage mode, the external bus structure is the same as for MCS 51 microcontrollers. The upper address bits (A15:8) are on port 2, and the lower address bits (A7:0) are multiplexed with the data (D7:0) on port 0. External code fetches and data reads use the two-state bus cycle shown in Figure 12-4. For the write cycle (Figure 12-5), a third state is appended to provide recovery time for the bus. Note that the write signal WR# is strobed for all memory regions, except for the case of RD1 = 1 and RD0 = 1, where WR# is strobed for regions 00: and 01: but **not** for regions FE: and FF:.


State 1
State 2
XTAL
ALE
PSEN# or RD#
P0
A7:0
D7:0
P2
A15:8

A2807-02

Figure 12-4. External Code Fetch or Data Read Bus Cycle (Nonpage Mode)


State 1
State 2
State 3
XTAL
ALE
WR#
P0
A7:0
D7:0
P2
A15:8

A2808-02

Figure 12-5. External Write Bus Cycle (Nonpage Mode)

12.3.4 Page Mode Bus Cycles

Page mode increases performance by reducing the time for external code fetches. Under certain conditions the controller fetches an instruction from external memory in one state time instead of two. Page mode does not affect internal code fetches.

The first code fetch to a 256-byte “page” of memory always uses a two-state bus cycle. Subsequent successive code fetches to the same page (*page hits*) require only a one-state bus cycle. When a subsequent fetch is to a different page (a *page miss*) it again requires a two-state bus cycle. The following external code fetches are always page-miss cycles:

- the first external code fetch after a page rollover†
- the first external code fetch after an external data bus cycle
- the first external code fetch after powerdown or idle mode
- the first external code fetch after a branch, return, interrupt, etc.

In page mode, the 8XC251SB bus structure is different from the bus structure in MCS 51 controllers (Figure 12-6). The upper address bits A15:8 are multiplexed with the data D7:0 on port 2, and the lower address bits (A7:0) are on port 0.

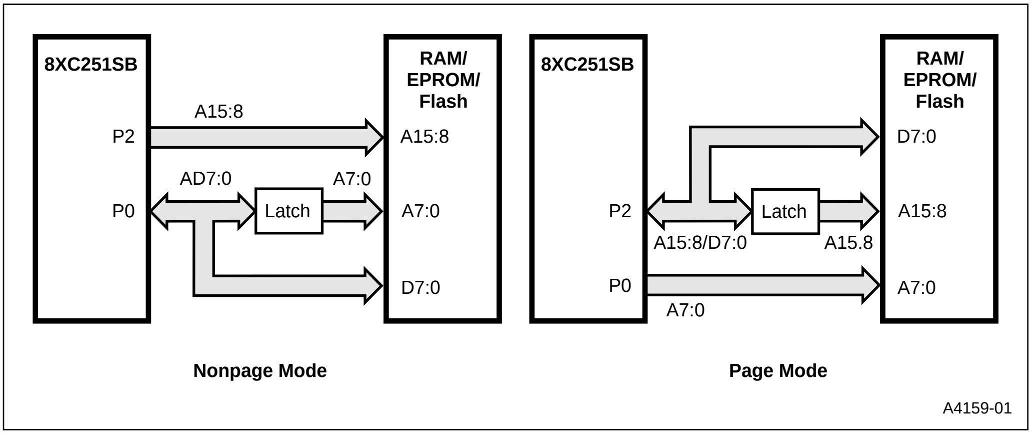

8XC251SB
P2
P0
A15:8
AD7:0
Latch
A7:0
RAM/
EPROM/
Flash
A15:8
A7:0
D7:0
Nonpage Mode
8XC251SB
P2
P0
A15:8/D7:0
Latch
A15.8
A7:0
RAM/
EPROM/
Flash
D7:0
A15:8
A7:0
Page Mode
A4159-01

Figure 12-6. Bus Structure in Nonpage Mode and Page Mode

---

† A page rollover occurs when the address increments from the top of one 256-byte page to the bottom of the next (e.g., from FF:FAFFH to FF:FB00H).

Figure 12-7 shows the two types of external bus cycles for code fetches in page mode. The *page-miss* cycle is the same as a code fetch cycle in nonpage mode (except for the different signals on ports 0 and 2). For the *page-hit* cycle, the upper eight address bits are the same as for the preceding cycle. Therefore, ALE is not strobed, and the values of A15:8 are retained in the address latches. In a single state, the new values of A7:0 are placed on port 0, and memory places the instruction byte on port 2. Notice that a page hit reduces the available address access time by one state. Therefore, faster memories may be required to support page mode.

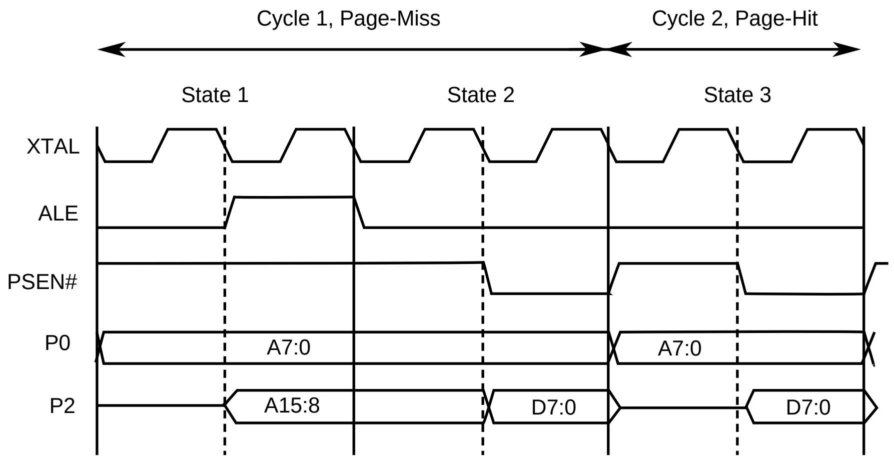

Cycle 1, Page-Miss
Cycle 2, Page-Hit
State 1
State 2
State 3
XTAL
ALE
PSEN#
P0
P2
A7:0
A7:0
A15:8
D7:0
D7:0
A2809-02

Figure 12-7. External Code Fetch Bus Cycle (Page Mode)

Figure 12-8 and Figure 12-9 show the bus cycles for data reads and writes in page mode. These cycles are identical to those for nonpage mode, except for the different signals on ports 0 and 2.

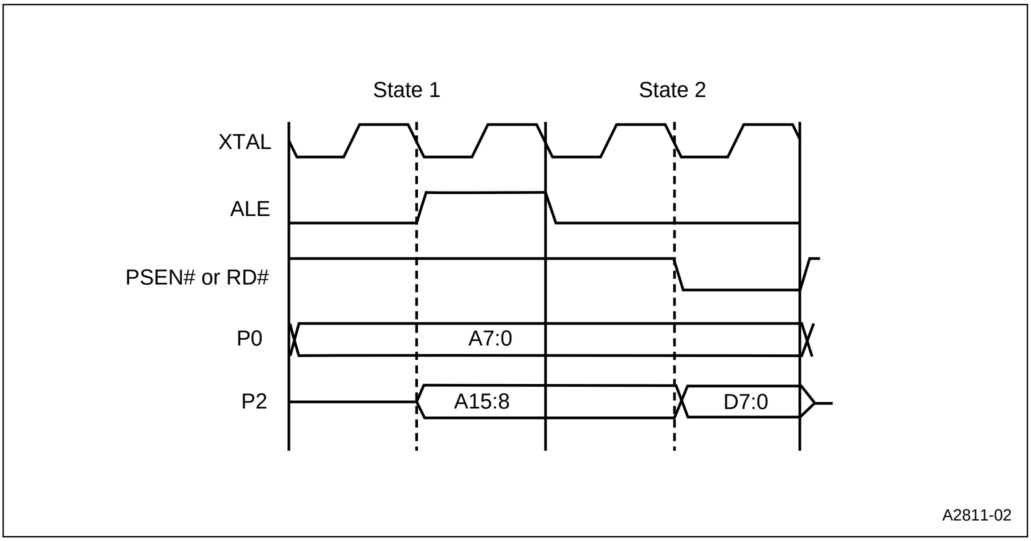

State 1
State 2
XTAL
ALE
PSEN# or RD#
P0
A7:0
P2
A15:8
D7:0
A2811-02

Figure 12-8. External Data Read Bus Cycle (Page Mode)

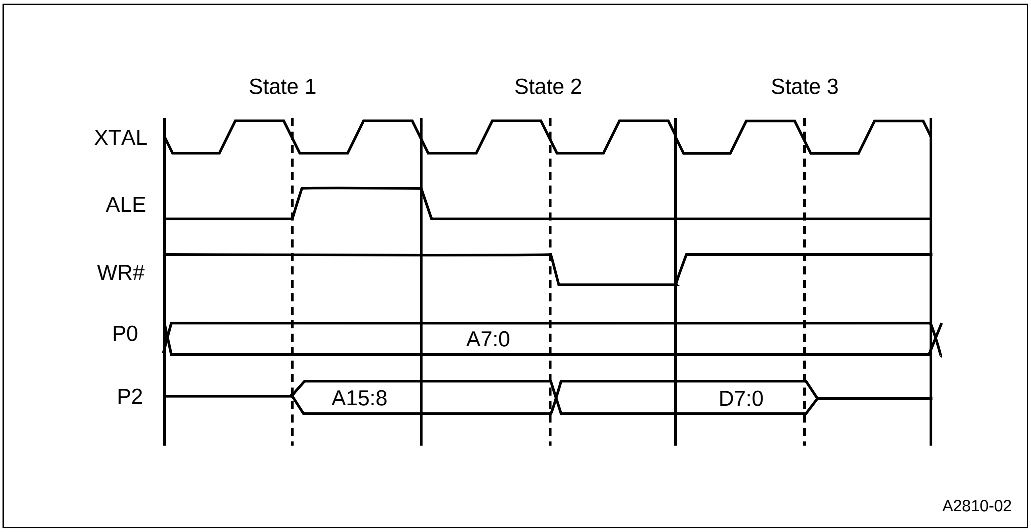

State 1
State 2
State 3
XTAL
ALE
WR#
P0
A7:0
P2
A15:8
D7:0
A2810-02

Figure 12-9. External Write Bus Cycle (Page Mode)

12.4 WAIT STATES

The 8XC251SB can be configured to add an external wait state by extending the RD#/PSEN#/WR# pulses or by extending the ALE pulse (see “Wait States (WSA, WSB, XALE)” on page 12-6). You can also configure the chip to use both types of wait states for a total of two external wait states. Accesses to on-chip code and data memory always use zero wait states.

12.4.1 Extending PSEN#/RD#/WR#

Figures 12-10 and 12-11 show bus cycles with an extended RD#/PSEN# wait state and an extended WR# wait state.

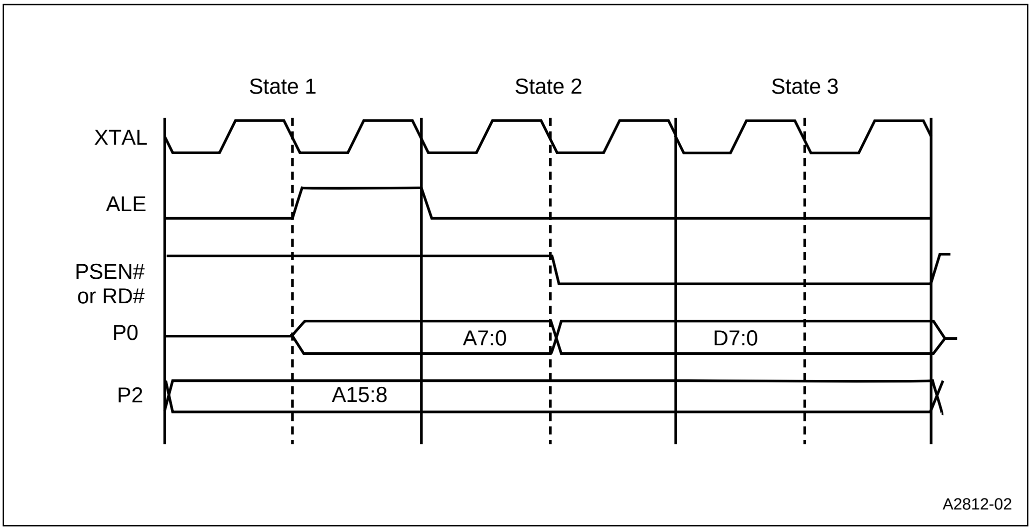


Figure 12-10. External Code Fetch or Data Read Bus Cycle with One PSEN#/RD# Wait State (Nonpage Mode)

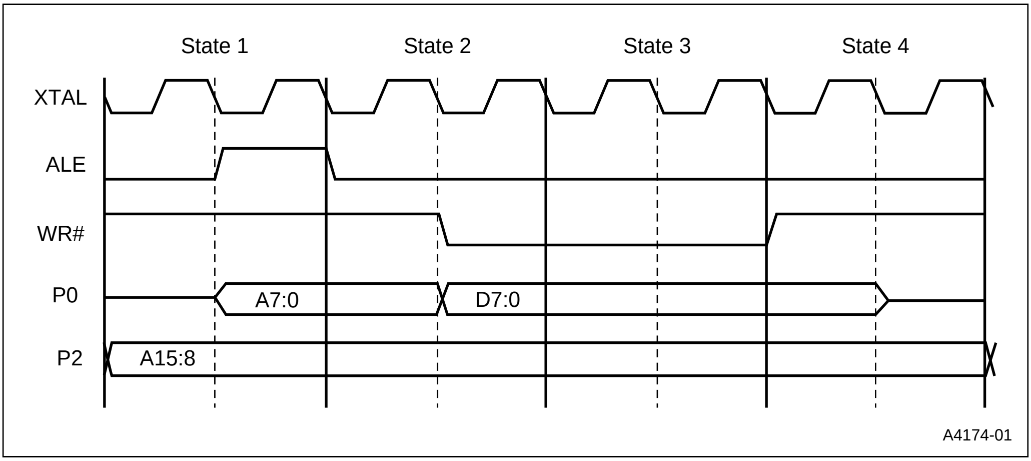

State 1
State 2
State 3
State 4
XTAL
ALE
WR#
P0
A7:0
D7:0
P2
A15:8
A4174-01

Figure 12-11. External Write Bus Cycle with One WR# Wait State (Nonpage Mode)

12.4.2 Extending ALE

Figure 12-12 shows a bus cycle for a code-fetch or a data-read with an extended ALE wait state. The wait state extends the bus cycle from two states to three. For an external write, the extended ALE extends the bus cycle from three states to four.

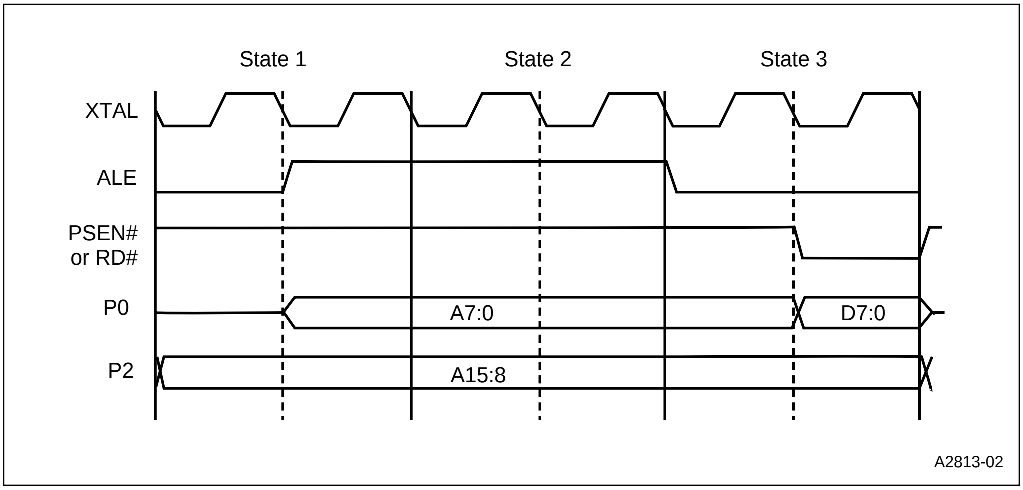

State 1
State 2
State 3
XTAL
ALE
PSEN# or RD#
P0
A7:0
D7:0
P2
A15:8
A2813-02

Figure 12-12. External Code Fetch or Data Read Bus Cycle with One ALE Wait State (Nonpage Mode)

12.5 PORT 0 AND PORT 2 STATUS

This section summarizes the status of the port 0 and port 2 pins when these ports are used as the external bus. A more comprehensive description of the ports and their use is given in Chapter 6, “Input/Output Ports.”

When port 0 and port 2 are used as the external memory bus, the signals on the port pins can originate from three sources:

- the 8XC251SB CPU (address bits, data bits)
- the port SFRs: P0 and P2 (logic levels)
- an external device (data bits)

The port 0 pins (but not the port 2 pins) can also be held in a high-impedance state. Table 12-5 lists the status of the port 0 and port 2 pins when the chip in is the normal operating mode and the external bus is idle or executing a bus cycle.

Table 12-5. Port 0 and Port 2 Pin Status In Normal Operating Mode

Port	8-bit/16-bit Addressing	Nonpage Mode		Page Mode	
		Bus Cycle	Bus Idle	Bus Cycle	Bus Idle
Port 0	8 or 16	AD7:0 (1)	High Impedance	A7:0 (1)	High Impedance
Port 2	8	P2 (2)	P2	P2/D7:0 (2)	High Impedance
	16	A15:8	P2	A15:8/D7:0	High Impedance

NOTES:

1. During external memory accesses, the CPU writes FFH to the P0 register and the register contents are lost.
2. The P2 register can be used to select 256-byte pages in external memory.

12.5.1 Port 0 and Port 2 Pin Status in Nonpage Mode

In nonpage mode the port pins have the same signals as those on the 8XC51FX. For an external memory instruction using a 16-bit address, the port pins carry address and data bits during the bus cycle. However, if the instruction uses an 8-bit address (e.g., MOVX @Ri), the contents of P2 are driven onto the pins. These pin signals can be used to select 256-bit pages in external memory.

During a bus cycle, the CPU always writes FFH to P0, and the former contents of P0 are lost. A bus cycle does not change the contents of P2. When the bus is idle, the port 0 pins are held at high impedance, and the contents of P2 are driven onto the P2 pins.

12.5.2 Port 0 and Port 2 Pin Status in Page Mode

In a page-mode bus cycle, the data is multiplexed with the upper address byte on port 2. However, if the instruction uses an 8-bit address (e.g., MOVX @Ri), the contents of P2 are driven onto the pins when data is not on the pins. These logic levels can be used to select 256-bit pages in external memory. During bus idle, the port 0 and port 2 pins are held at high impedance.

(For port pin status when the chip in is idle mode, powerdown mode, or reset, see Chapter 11, “Special Operating Modes.”)

12.6 EXTERNAL MEMORY DESIGN EXAMPLES

This section shows five examples of external memory designs for 8XC251SB systems. The examples illustrate the design flexibility provided by the configuration options, especially for the PSEN# and RD# signals. Many other designs are possible.

12.6.1 Nonpage Mode, 64 Kbytes External EPROM, 64 Kbytes External RAM

Figure 12-13 shows an 80C251SB in nonpage mode with 64 Kbytes of external EPROM and 64 Kbytes of external RAM. The 80C251SB is configured so that RD# strobes for addresses ≤ 7F:FFFFH and PSEN# strobes for addresses ≥ 80:0000H (RD1 = 1 and RD0 = 1). Figure 12-14 shows two ways to address the external memory in the internal memory space.

The lower 1056 bytes of the external RAM must be addressed in region 01:. Addressing the other external RAM locations in either region 00: or region 01: produces the same address at the external bus pins. However, if the external EPROM and the external RAM require different numbers of wait states, the external RAM must be addressed entirely in region 01:. (Recall that regions 00:, FE:, and FF: always have the same number of wait states. See “Wait States (WSA, WSB, XALE)” on page 12-6.)

The examples that follow illustrate two possibilities for addressing the external RAM.

12.6.1.1 An Application Requiring Fast Access to the Stack

If an application requires fast access to the stack, the stack can reside in the fast on-chip data RAM (00:0020H–00:041FH) and, when necessary, roll out into the slower external RAM. In this case, the external RAM can have a wait state only if the EPROM has a wait state. Otherwise, if the stack rolls out above location 00:041FH, the external RAM would be accessed with no wait state. Regions 00: and 01: on the left side of Figure 12-14 apply to this example.

12.6.1.2 An Application Requiring Fast Access to Data

If fast access to a block of data is more important than fast access to the stack, the data can be stored in the on-chip data RAM, and the stack can be located entirely in external memory. If the external RAM has a wait state and the EPROM has no wait state, the external RAM must be addressed entirely in region 01:. Regions 00: and 01: on the right side of Figure 12-14 apply to this example.

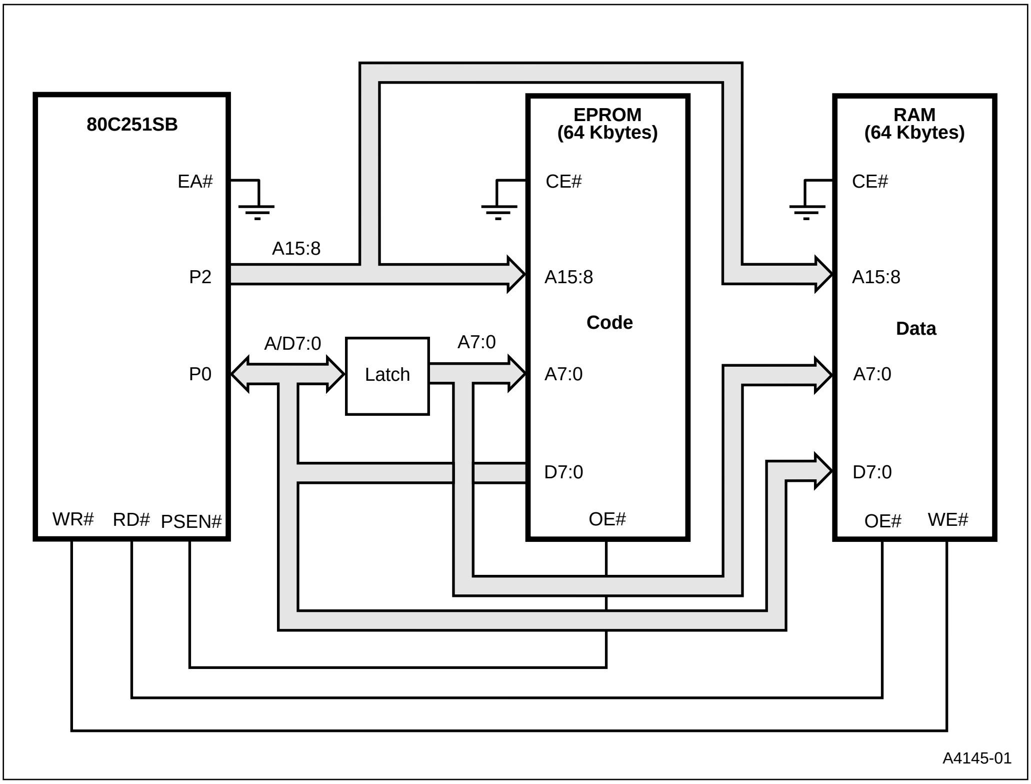

80C251SB
EA#
P2
P0
WR#
RD#
PSEN#
A15:8
A/D7:0
Latch
A7:0
EPROM
(64 Kbytes)
CE#
A15:8
Code
A7:0
D7:0
OE#
RAM
(64 Kbytes)
CE#
A15:8
Data
A7:0
D7:0
OE#
WE#
A4145-01

Figure 12-13. 80C251SB in Nonpage Mode with External EPROM and RAM

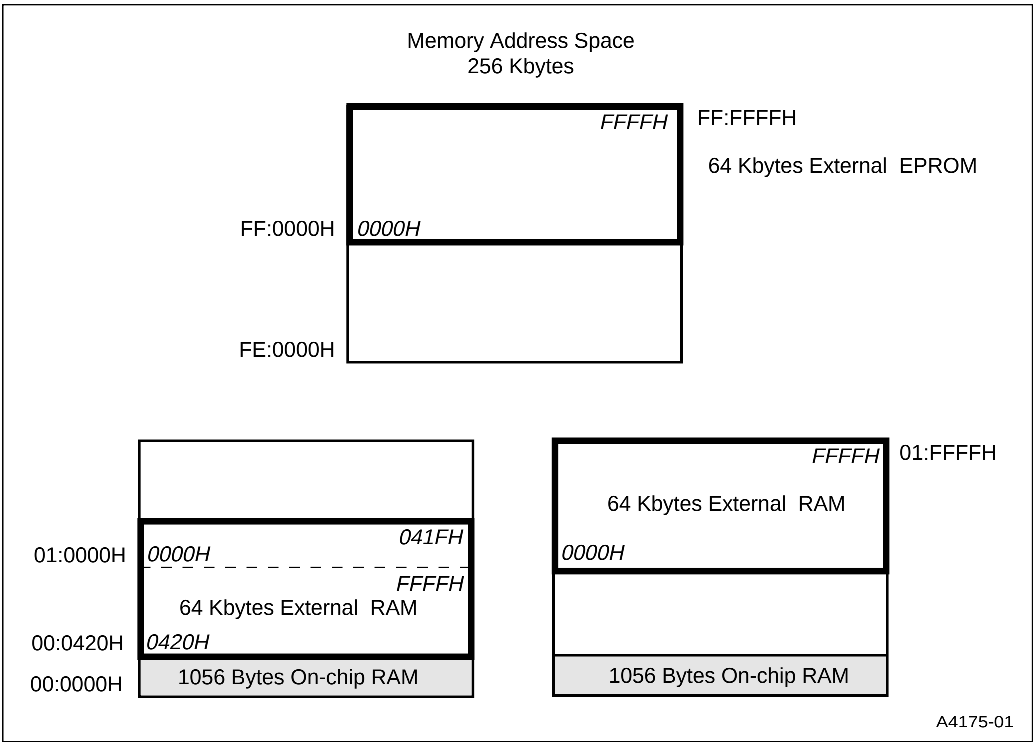

Memory Address Space
256 Kbytes
FFFFH
FF:FFFFH
64 Kbytes External EPROM
FF:0000H
0000H
FE:0000H
041FH
01:0000H
0000H
FFFFH
64 Kbytes External RAM
00:0420H
0420H
00:0000H
1056 Bytes On-chip RAM
FFFFH
01:FFFFH
64 Kbytes External RAM
0000H
1056 Bytes On-chip RAM
A4175-01

Figure 12-14. The Memory Space for the Systems of Figure 12-13 and Figure 12-18

12.6.2 Nonpage Mode, 128 Kbytes External RAM

Figure 12-15 shows an 87C251SB/83C251SB in nonpage mode with 128 Kbytes of external RAM. The 87C251SB/83C251SB is configured so that RD# functions as A16, and PSEN# is strobed for all addresses (RD1 = 0, RD0 = 1). Figure 12-16 shows how the external RAM is addressed in the internal memory space. The lower 1056 bytes of external RAM are unavailable because accesses to the lower 1056 bytes in region 00: are directed to on-chip RAM.

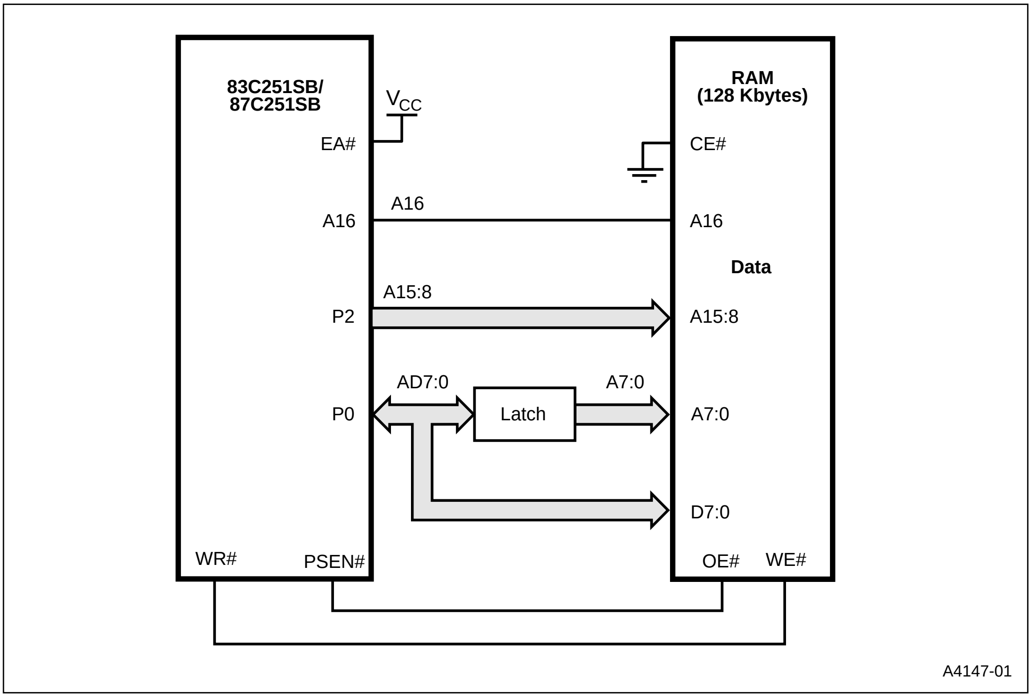

83C251SB/
87C251SB
V_{CC}
EA#
A16
A16
P2
A15:8
P0
AD7:0
Latch
A7:0
WR#
PSEN#
RAM
(128 Kbytes)
CE#
A16
Data
A15:8
A7:0
D7:0
OE#
WE#
A4147-01

Figure 12-15. 87C251SB/83C251SB in Nonpage Mode with 128 Kbytes of External RAM

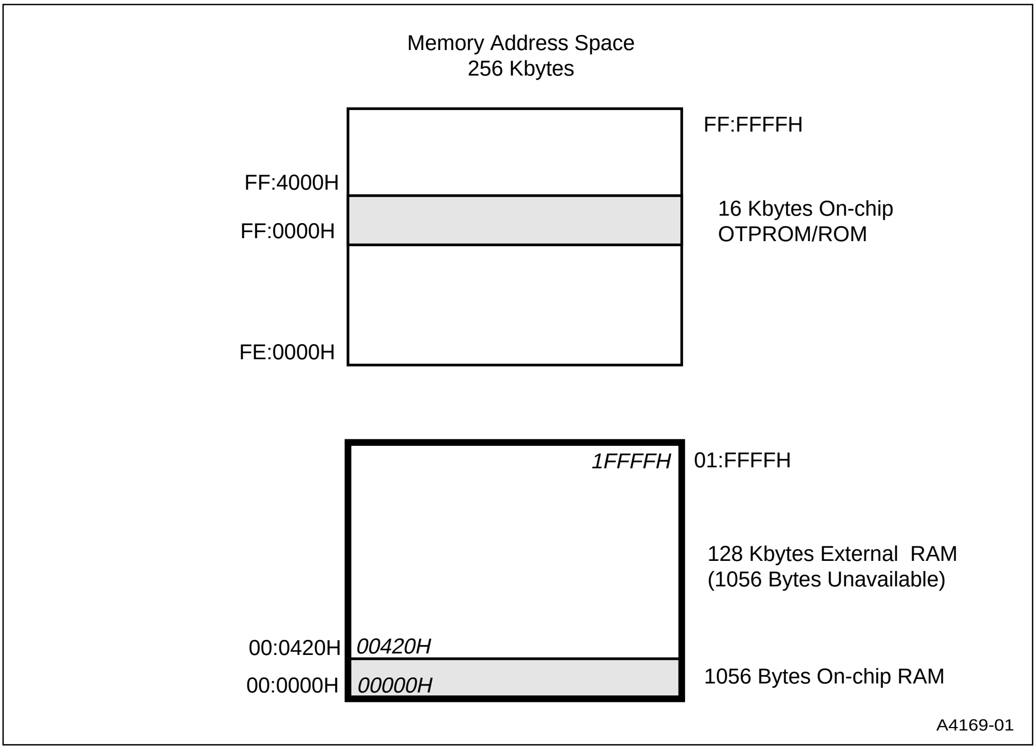

Memory Address Space
256 Kbytes
FF:FFFFH
FF:4000H
FF:0000H
16 Kbytes On-chip
OTPROM/ROM
FE:0000H
1FFFFH
01:FFFFH
128 Kbytes External RAM
(1056 Bytes Unavailable)
00:0420H
00420H
00:0000H
00000H
1056 Bytes On-chip RAM
A4169-01

Figure 12-16. The Memory Space for the System of Figure 12-15

12.6.3 Page Mode, 128 Kbytes External Flash

Figure 12-17 shows the 80C251SB in page mode with 128 Kbytes of external flash. Note that port 2 carries both the upper address bits (A15:0) and the data (D7:0), while port 0 carries only the lower address bits (A7:0). The 80C251SB is configured for 17 external address bits and a single read signal (PSEN#). The 128 Kbytes of external flash are accessed in pages FE: and FF: in the internal memory space.

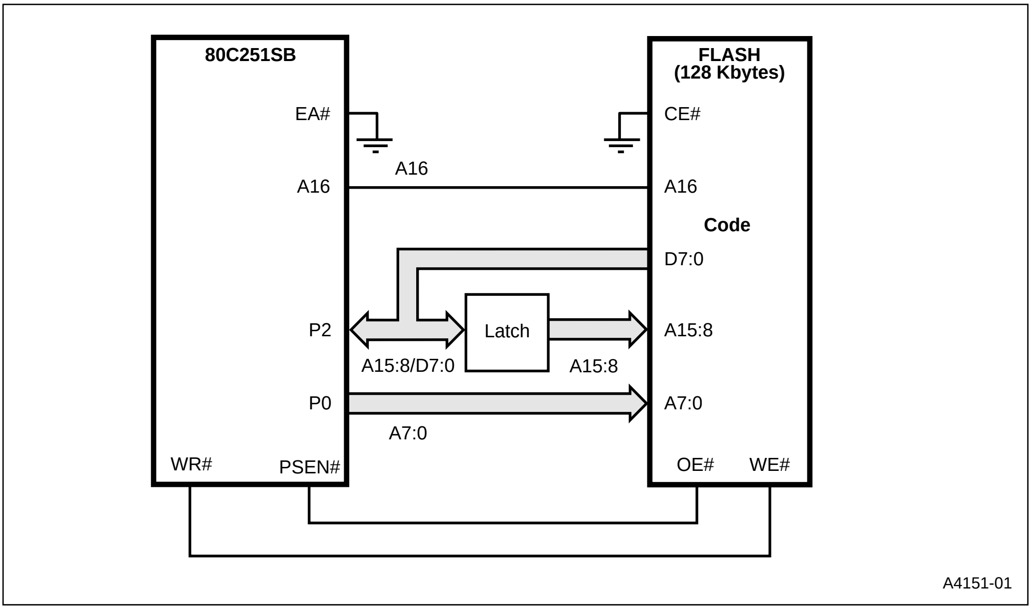


Figure 12-17. 80C251SB in Page Mode with External Flash

12.6.4 Page Mode, 64 Kbytes External EPROM, 64 Kbytes External RAM

Figure 12-18 shows an 80C251SB in page mode with 64 Kbytes of external EPROM and 64 Kbytes of external RAM. The 80C251SB is configured so that RD# strobes for addresses ≤ 7F:FFFFH, and PSEN# strobes for addresses ≥ 80:0000H (RD1 = 1 and RD0 = 1).

This system is the same as the system in Figure 12-13 on page 12-17, except that this design operates in page mode. Accordingly, the two systems have the same memory map (Figure 12-14 on page 12-18), and the comments on addressing external RAM apply here also.

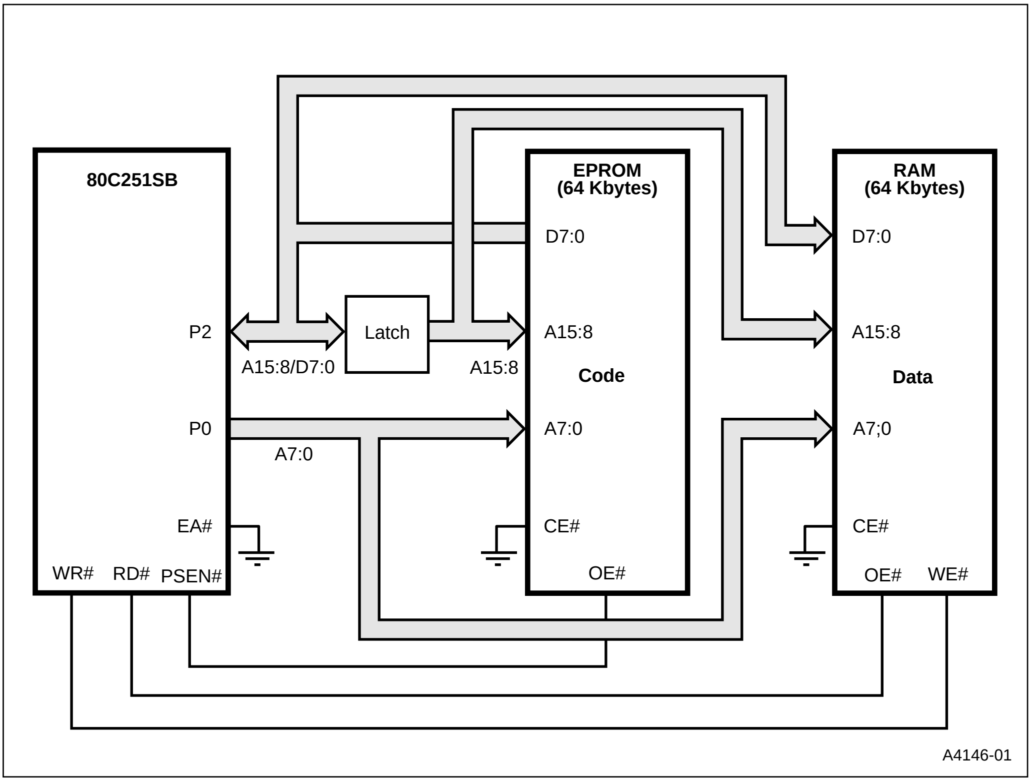

80C251SB
P2
P0
EA#
WR#
RD#
PSEN#
A15:8/D7:0
Latch
A15:8
A7:0
EPROM
(64 Kbytes)
D7:0
A15:8
Code
A7:0
CE#
OE#
RAM
(64 Kbytes)
D7:0
A15:8
Data
A7;0
CE#
OE#
WE#
A4146-01

Figure 12-18. 80C251SB in Page Mode with External EPROM and RAM

12.6.5 Page Mode, 64 Kbytes External Flash, 32 Kbytes External RAM

Figure 12-19 shows an 80C251SB in page mode with 64 Kbytes of external flash memory for code storage and 32 Kbytes of external RAM. The 80C251SB is configured so that PSEN# is strobed for all reads, and RD# functions as A16 (RD1 = 0, RD0 = 1). Figure 12-20 shows how the external flash and RAM are addressed in the internal memory space. The external RAM is accessed for internal addresses 00:0420H–00:7FFFH. The first 1056 bytes of external RAM are unused because accesses to locations 00:0000H–00:041FH are directed to on-chip RAM.

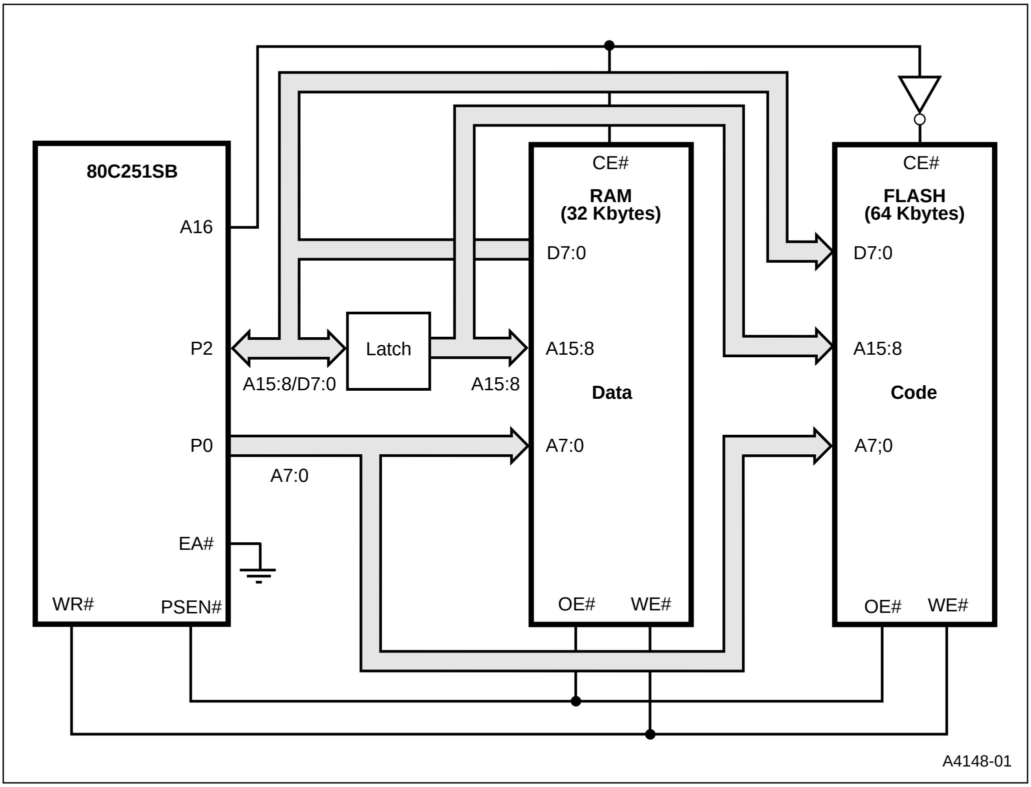

80C251SB
A16
P2
P0
EA#
WR#
PSEN#
A15:8/D7:0
Latch
A15:8
A7:0
CE#
RAM
(32 Kbytes)
D7:0
A15:8
Data
A7:0
OE#
WE#
CE#
FLASH
(64 Kbytes)
D7:0
A15:8
Code
A7;0
OE#
WE#
A4148-01

Figure 12-19. 80C251SB in Page Mode with External Flash and RAM

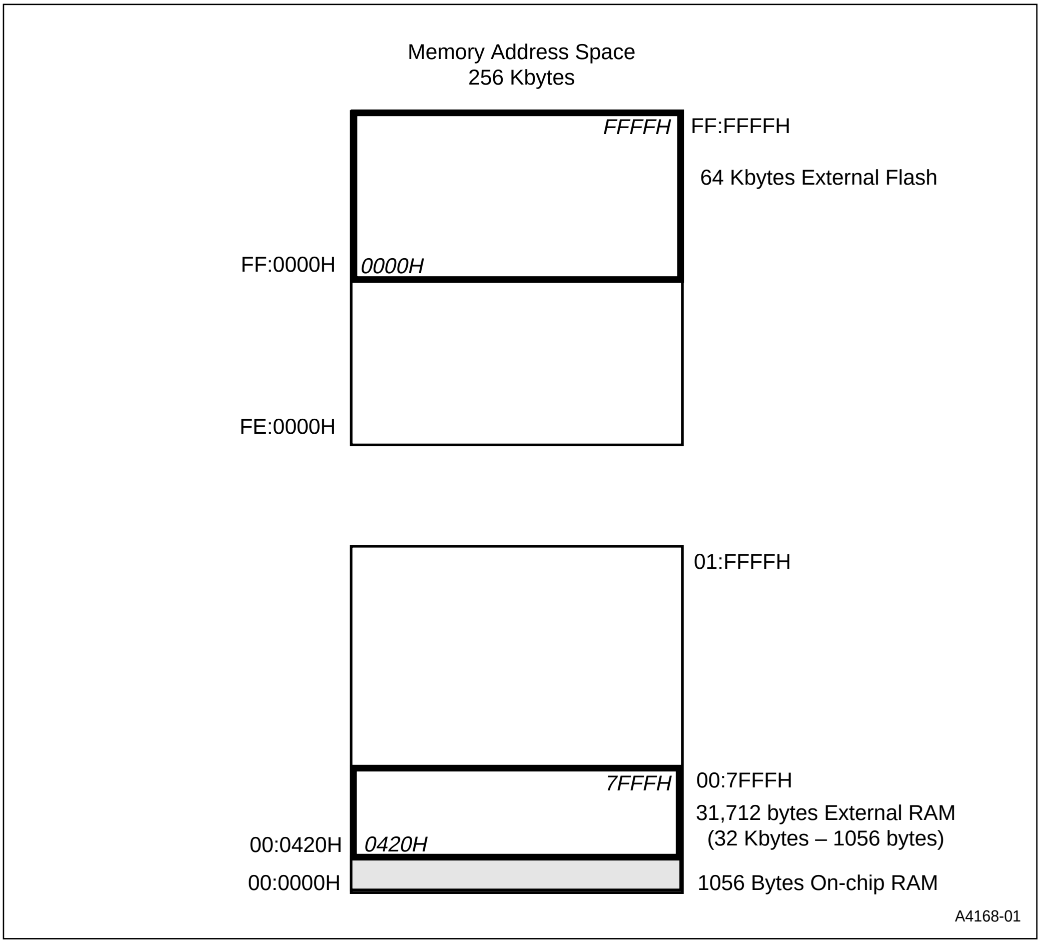

Memory Address Space
256 Kbytes
FFFFH
FF:FFFFH
64 Kbytes External Flash
FF:0000H
0000H
FE:0000H
01:FFFFH
7FFFH
00:7FFFH
31,712 bytes External RAM
(32 Kbytes – 1056 bytes)
00:0420H
0420H
00:0000H
1056 Bytes On-chip RAM
A4168-01

Figure 12-20. The Memory Space for the System of Figure 12-19

12.7 EXTERNAL BUS AC TIMING SPECIFICATIONS

This section defines the AC timing specifications for the external bus. Refer to the latest data sheet to be sure that your system meets specifications. Figure 12-21 shows the bus waveforms for instruction or data reads and data writes in nonpage mode. Figure 12-22 shows the bus waveforms for data reads and data writes in page mode, and Figure 12-23 shows the bus waveforms for instruction fetches in page mode. Table 12-6 on page 12-28 defines the symbols used in the timing diagrams. Tables 12-8 and 12-7 define the timing parameters.

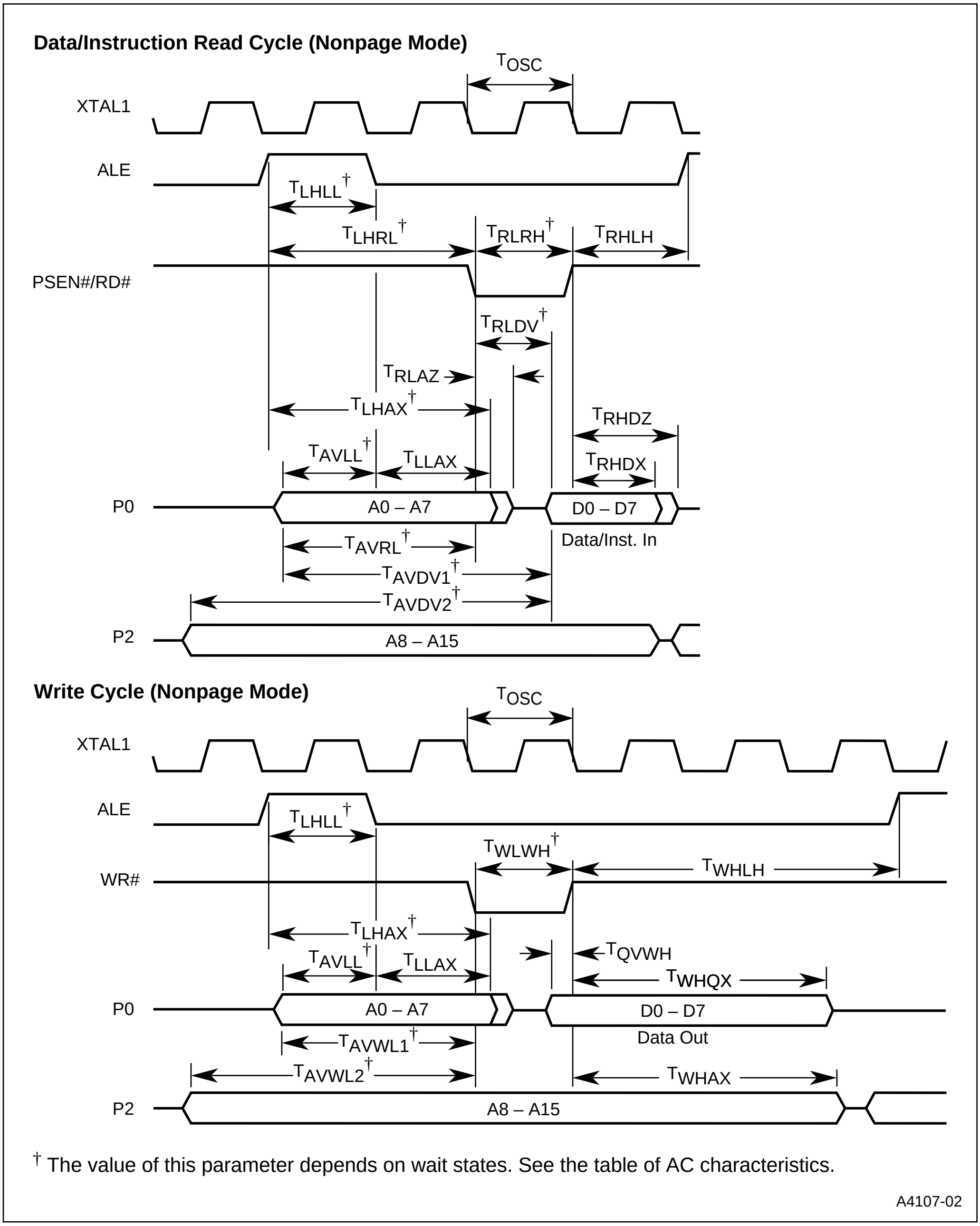

Data/Instruction Read Cycle (Nonpage Mode)
TOSC
XTAL1
ALE
TLHLL†
TLHRL†
TRLRH†
TRHLH
PSEN#/RD#
TRLDV†
TRLAZ
TLHAX†
TRHDZ
TAVLL†
TLLAX
TRHDX
P0
A0 – A7
D0 – D7
Data/Inst. In
TAVRL†
TAVDV1†
TAVDV2†
P2
A8 – A15
Write Cycle (Nonpage Mode)
TOSC
XTAL1
ALE
TLHLL†
TWLWH†
WR#
TWHLH
TLHAX†
TAVLL†
TLLAX
TQVWH
TWHQX
P0
A0 – A7
D0 – D7
Data Out
TAVWL1†
TAVWL2†
TWHAX
P2
A8 – A15
† The value of this parameter depends on wait states. See the table of AC characteristics.
A4107-02

Figure 12-21. External Bus Cycles for Data/Instruction Read and Data Write in Nonpage Mode

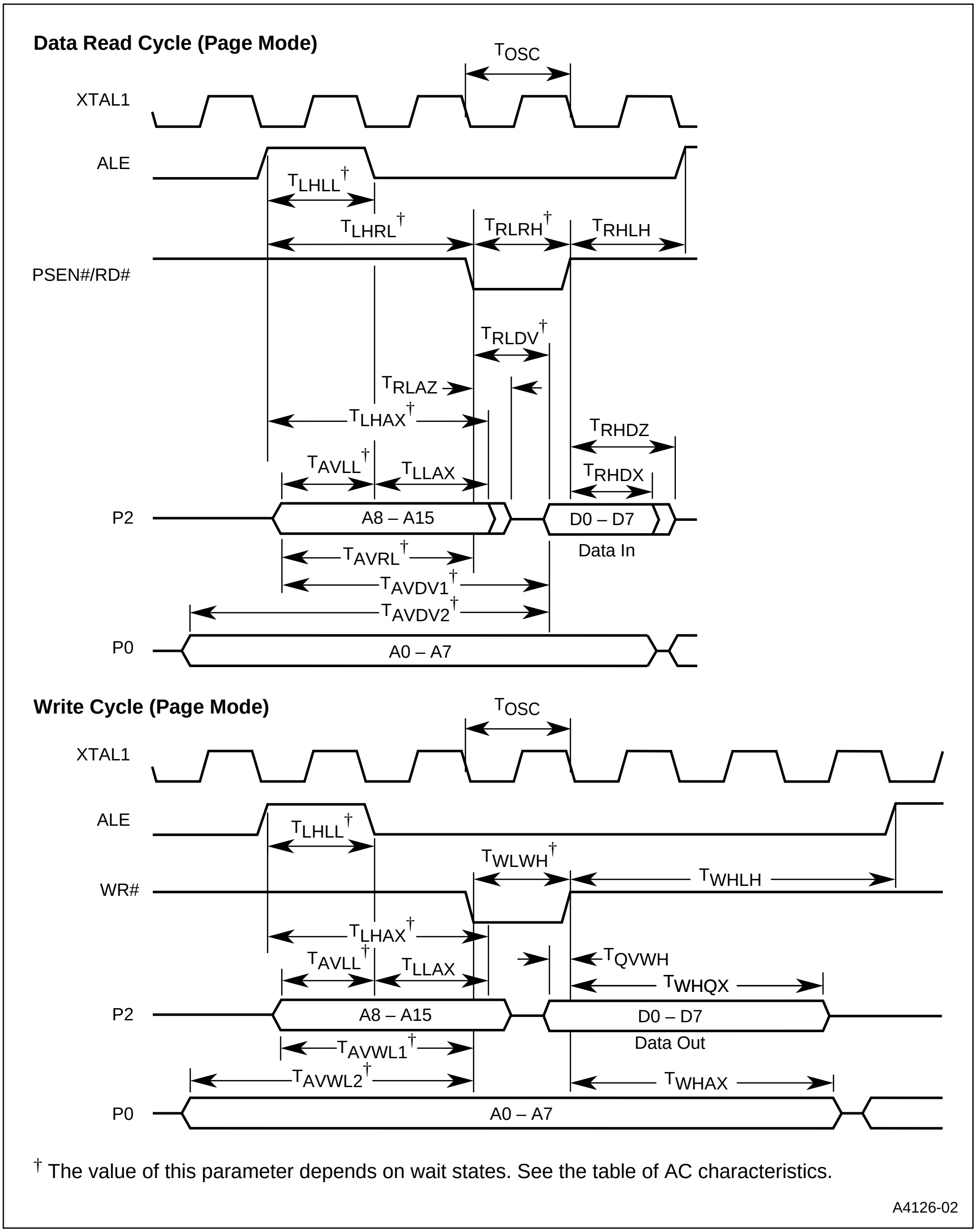
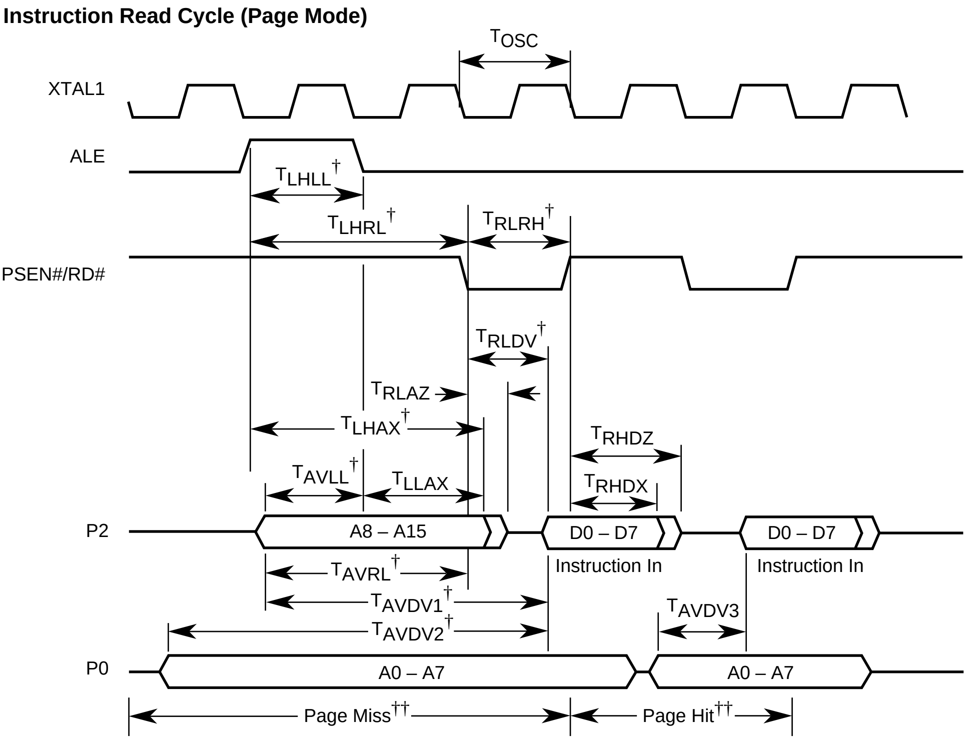

Data Read Cycle (Page Mode)
TOSC
XTAL1
ALE
TLHLL†
TLHRL†
TRLRH†
TRHLH
PSEN#/RD#
TRLDV†
TRLAZ
TLHAX†
TRHDZ
TAVLL†
TLLAX
TRHDX
P2
A8 – A15
D0 – D7
Data In
TAVRL†
TAVDV1†
TAVDV2†
P0
A0 – A7
Write Cycle (Page Mode)
TOSC
XTAL1
ALE
TLHLL†
TWLWH†
WR#
TWHLH
TLHAX†
TAVLL†
TLLAX
TQVWH
TWHQX
P2
A8 – A15
D0 – D7
Data Out
TAVWL1†
TAVWL2†
TWHAX
P0
A0 – A7
† The value of this parameter depends on wait states. See the table of AC characteristics.
A4126-02

Figure 12-22. External Bus Cycles for Data Read and Data Write in Page Mode


Instruction Read Cycle (Page Mode)
TOSC
XTAL1
ALE
TLHLL†
TLHRL†
TRLRH†
PSEN#/RD#
TRLDV†
TRLAZ
TLHAX†
TRHDZ
TAVLL†
TLLAX
TRHDX
P2
A8 – A15
D0 – D7
D0 – D7
Instruction In
Instruction In
TAVRL†
TAVDV1†
TAVDV3
TAVDV2†
P0
A0 – A7
A0 – A7
Page Miss††
Page Hit††

† The value of this parameter depends on wait states. See the table of AC characteristics.

†† A page hit (i.e., a code fetch to the same 256-byte "page" as the previous code fetch) requires one state ($2T_{OSC}$); a page miss requires two states ($4T_{OSC}$).

A4127-02

Figure 12-23. External Bus Cycles for Instruction Read in Page Mode

12.7.1 Explanation of AC Symbols

Each symbol consists of two pairs of letters prefixed by “T” (for time). The characters in a pair indicate a signal and its condition, respectively. Symbols represent the time between the two signal/condition points. For example, T_{LHRL} is the time between signal L (ALE) condition H (high) and R (RD#) condition L (Low). Table 12-6 defines the signal and condition codes

Table 12-6. AC Timing Symbol Definitions

Signals	
A	Address
D	DATA
L	ALE
Q	Data Out
R	RD#/PSEN#
W	WR#

Conditions	
H	High
L	Low
V	Valid
X	No Longer Valid
Z	Floating

12.7.2 AC Timing Definitions

This section defines the timing parameters shown in Figures 12-21, 12-22, and 12-23. Tables 12-8 and 12-7 list the definitions of timing specifications on the memory system and the 8XC251SB.

Table 12-7. AC Timing Definitions for Specifications on the 8XC251SB

THE 8XC251SB MEETS THESE SPECIFICATIONS		
Symbol	**Definition**	**Notes**
F_{OSC}	Frequency on XTAL: Frequency of the signal input on the XTAL1 input.	
T_{OSC}	$1/F_{OSC}$: Period of the signal on XTAL1/XTAL2: AC Timings are referenced to T_{OSC}.	
T_{LHLL}	ALE Pulse Width: Length of time ALE is asserted.	(2)
T_{LHRL}	ALE High to RD# or PSEN# Low: Time after ALE goes high until RD# or PSEN# goes low.	(1)
T_{RLRH}	RD# or PSEN# Pulse Width: Length of time RD# or PSEN# is asserted.	(3)
T_{RHLH}	RD# High to ALE Asserted: Time after RD# goes high until the next ALE pulse goes high.	(1)
T_{RLAZ}	RD# Low to Address Float: Time after RD# goes low until the 8XC251SB stops driving the address on the bus.	
T_{AVLL}	Address Valid to ALE Low. Length of time the lower byte of the address is valid (on port 0) before ALE goes low.	(2)
T_{LHAX}	ALE High to Address Hold. Length of time the 8XC251SB holds the lower byte of the address on the bus (port 0) after ALE goes high.	(2)
T_{LLAX}	Address Hold after ALE Low: Length of time the 8XC251SB holds the lower byte of the address on the bus (port 0) after ALE goes low.	
T_{AVRL}	Address Valid to RD# or PSEN# Low: Length of time the lower byte of the address is valid on the bus (port 0) before RD# or PSEN# goes low.	(1,2)
T_{WLWH}	WR# Pulse Width: Length of time WR# is asserted.	(3)
T_{WHLH}	WR# High to ALE High: Time after WR# goes high until the next ALE pulse is goes high.	
T_{AVWL1}	Address (port 0) Valid to WR# Low: Length of time that the 8XC251SB drives the address onto the bus (port 0) before WR# goes low.	(2)
T_{AVWL2}	Address (port 2) Valid to WR# Low: Length of time that the 8XC251SB drives the address onto the bus (port 2) before WR# goes low.	(2)
T_{WHAX}	Address Hold after WR# High: Time the 8XC251SB holds the upper byte of the address on the bus (port 2) after WR# goes high.	

NOTES:

1. Specifications for PSEN# are identical to those for RD#.
2. If a wait state is added by extending ALE, this time increases by $2T_{OSC}$.
3. If a wait state is added by extending RD#/PSEN#/WR#, this time increases by $2T_{OSC}$.
4. If wait states are added as described in both Note 2 and Note 3, this time increases by a total of $4T_{OSC}$.

Table 12-8. AC Timing Definitions for Specifications on the Memory System

THE EXTERNAL MEMORY SYSTEM MUST MEET THESE SPECIFICATIONS		
Symbol	**Definition**	**Notes**
T_{RHDZ}	Data/Instruction Float After RD# or PSEN# High: Time after RD# or PSEN# goes high until memory system must float the bus. If this timing is not met, bus contention occurs.	(1)
T_{RHDX}	Data/Instruction Hold After RD#/ PSEN# High: Length of time the memory system must hold data on the bus after RD# or PSEN# goes high.	(1)
T_{RLDV}	RD# Low to Input Data Valid: Time after RD# goes low until the memory system must output valid data/instruction.	(1,3)
T_{QVWH}	Data Valid to WR# High: Length of time the memory system must output valid data before WR# goes high.	
T_{WHQX}	Data Hold after WR# High: Length of time the memory system must hold data on the bus after WR# goes high.	
T_{AVDV1}	Address (port 0) valid to Valid Data/Instruction In: Time after the 8XC251SB places a valid address on the bus (port 0) until the memory system must place valid data on the bus (port 0).	(2,3,4)
T_{AVDV2}	Address (port 2) Valid to Valid Data/Instruction In: Time after the 8XC251SB places a valid address on the bus (port 2) until the memory system must place valid data/instruction on the bus (port 0). If the bus cycle is an instruction fetch, this applies to a page miss.	(2,3,4)
T_{AVDV3}	Address (port 2) Valid to Valid Instruction In: Time after the 8XC251SB places a valid address on the bus (port 2) until the memory system must place a valid instruction on the bus (port 0). This applies to a page hit.	

NOTES:

1. Specifications for PSEN# are identical to those for RD#.
2. If a wait state is added by extending ALE, this time increases by $2T_{OSC}$.
3. If a wait state is added by extending RD#/PSEN#/WR#, this time increases by $2T_{OSC}$.
4. If wait states are added as described in both Note 2 and Note 3, this time increases by a total of $4T_{OSC}$

13

Programming and Verifying Nonvolatile Memory

CHAPTER 13
PROGRAMMING AND VERIFYING NONVOLATILE MEMORY

This chapter provides instructions for programming and verifying on-chip nonvolatile memory on the 8XC251SB. The programming instructions cover the entry of program code into on-chip code memory and other categories of information into nonvolatile memory outside the memory address space. The verify instructions permit reading these memory locations to verify their contents. The operations covered in this chapter are:

• programming and verifying the on-chip code memory	(16 Kbytes)
• programming and verifying the configuration bytes	(4 bytes)
• programming and verifying the lock bits	(3 bits)
• programming the encryption array	(128 bytes)
• verifying the signature bytes	(3 bytes)

The programming instructions apply to the one-time-programmable 87C251SB (OTPROM). The verify instructions apply to the 87C251SB, the 83C251SB (ROM), and the configuration bytes on the 80C251SB (ROMless). In the unprogrammed state, OTPROM contains all 1s.

13.1 GENERAL

The 87C251SB OTPROM device is programmed and verified in the same manner as the 87C51FX, using the same quick-pulse programming algorithm, which programs at V_{PP} = 12.75 V using a series of five 100 µs PROG# pulses per byte. This results in a programming time of approximately 16 seconds for the 16-Kbyte on-chip code memory.

Programming and verifying operations differ from normal controller operation. Memory accesses are made one byte at a time, input/output ports are used in a different manner, and some pins (EA#/V_{PP} and ALE/PROG#) assume their alternative (programming) functions. For a complete list of signal descriptions, see Appendix B.

In some microcontroller applications, it is desirable that user program code be secure from unauthorized access. The 8XC251SB offers two types of protection for program code stored in the on-chip array.

- Program code in the on-chip code memory is encrypted when read out for verification if the encryption array is programmed.
- A three-level lock bit system restricts external access to the on-chip code memory.

It is recommended that user program code be located starting at address FF:0100H. Since the first instruction following device reset is fetched from FF:0000H, use a jump instruction to FF:0100H to begin execution of the user program. For information on address spaces, see Chapter 3.

CAUTION

> Execution of user code located in the top eight bytes of the on-chip user memory (i.e., FF:3FF8H–FF:3FFFH) may cause prefetches from the next higher addresses, which are in external memory. External memory fetches make use of port 0 and port 3 and may disrupt program execution if the program uses ports 0 or 3 for a different purpose.

13.2 PROGRAMMING AND VERIFYING MODES

Table 13-1 defines the programming and verifying modes and provides details about the setup. The modes correspond to the nonvolatile memory functions, i.e. on-chip code memory, encryption array, configuration bytes, etc. The configuration bytes, signature bytes, encryption array, and lock bits reside in nonvolatile memory outside the memory address space. The value applied to port 0 (see Table 13-1) specifies program or verify and provides the base address for the function. Addresses in the Address column are with respect to the base address.

Table 13-1. Programming and Verifying Modes

Mode	RST	PSEN#	V_{PP}	PROG#	Port 0	Port 2	Address Port 1 (high) Port 3 (low)	Notes
Program - On-chip Code Memory	High	Low	5 V, 12.75 V	5 Pulses	68H	data	0000H-3FFFH	1
Verify - On-chip Code Memory	High	Low	5 V	High	28H	data	0000H-3FFFH	
Program - Configuration Bytes	High	Low	5 V, 12.75 V	5 Pulses	69H	data	0080H-0083H	1
Verify - Configuration Bytes	High	Low	5 V	High	29H	data	0080H-0083H	
Program - Lock Bits	High	Low	5 V, 12.75 V	25 Pulses	6BH	data	0001H-0003H	1, 2
Verify - Lock bits	High	Low	5 V	High	2BH	data	0000H	3
Program - Encryption Array	High	Low	5 V, 12.75 V	25 Pulses	6CH	data	0000H-007FH	1
Verify - Signature Bytes	High	Low	5 V	High	29H	data	0030H, 0031H, 0060H	

NOTES:

1. To program, raise V_{PP} to 12.75 V and pulse the PROG# pin. See Figure 13-2 for waveforms.
2. No data input. Identify the lock bits with the address lines as follows: LB3 - 0003H, LB2 - 0002H, LB1 - 0001H
3. The three lock bits are verified in a single operation. The states of the lock bits appear simultaneously at port 2 as follows: LB3 - P2.3, LB2 - P2.2. LB1 - P2.1. High = programmed.

13.3 GENERAL SETUP

Figure 13-1 shows the general setup for programming and verifying the OTPROM areas on the 87C251SB. The figure also applies to verifying the 83C251SB and reading the configuration bytes on the 80C251SB.

The controller must be running with an oscillator frequency of 4 MHz to 6 MHz. To program, set up the controller as shown in Table 13-1 with the mode of operation (program/verify and memory area) specified on port 0, the address with respect to the starting address of the memory area applied to ports 1 and 3, and the data on port 2. Apply a logic high to the RST pin and V_{CC} to EA#/V_{PP}. ALE/PSEN#, normally an output pin, must be held low externally.

To perform the write operation, raise V_{PP} to 12.75 V and pulse the PROG# pin per Table 13-1. Then return V_{PP} to 5 V. Waveforms are shown in Figure 13-2.

CAUTION

> The V_{PP} source must be well regulated and free of glitches. The voltage on the V_{PP} pin must not exceed the specified maximum, even under transient conditions. See latest data sheet.

Verification is performed in a similar manner but without increasing V_{PP} and without pulsing PROG#. Figure 13-2 shows the OTPROM programming and verifying waveforms. For waveform timing information, refer to Figure 13-5 and Table 13-5 at the end of this section.

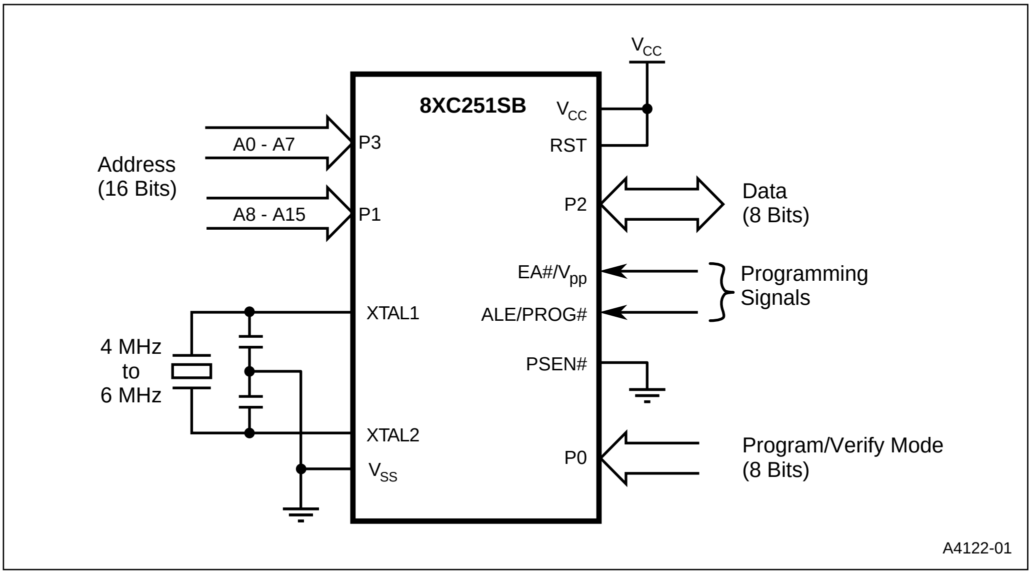

VCC
8XC251SB
VCC
RST
A0 - A7
P3
Address
(16 Bits)
A8 - A15
P1
P2
Data
(8 Bits)
EA#/Vpp
ALE/PROG#
Programming
Signals
XTAL1
4 MHz
to
6 MHz
PSEN#
XTAL2
VSS
P0
Program/Verify Mode
(8 Bits)
A4122-01

Figure 13-1. Setup for Programming and Verifying

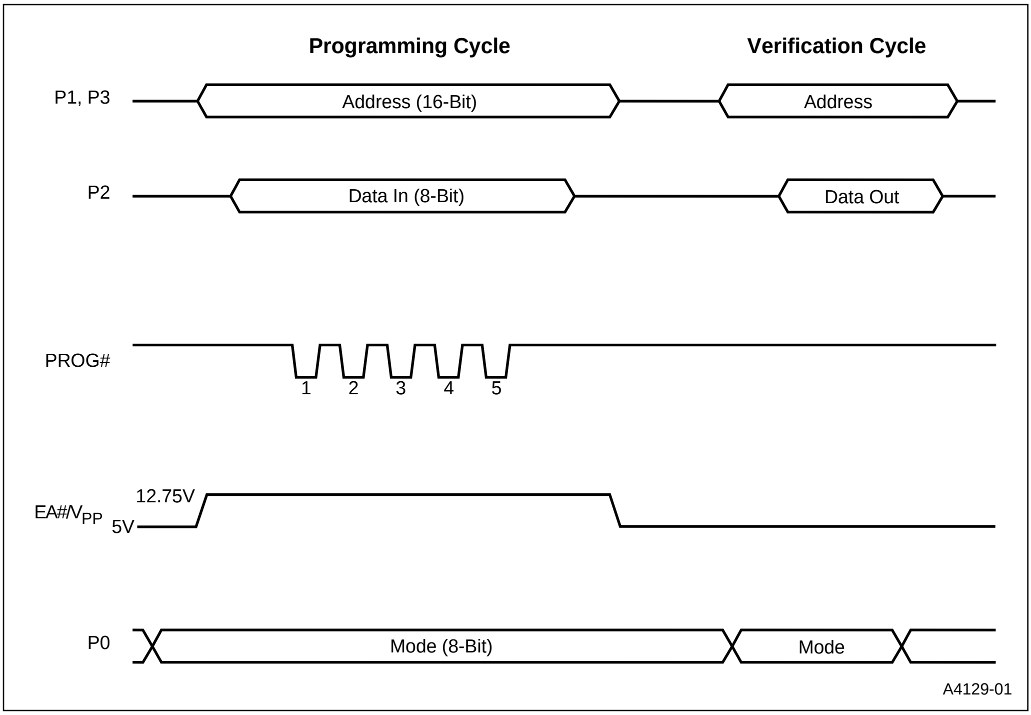

Programming Cycle
Verification Cycle
P1, P3
Address (16-Bit)
Address
P2
Data In (8-Bit)
Data Out
PROG#
1
2
3
4
5
12.75V
EA#/VPP
5V
P0
Mode (8-Bit)
Mode
A4129-01

Figure 13-2. OTPROM Programming Waveforms

13.4 OTPROM PROGRAMMING ALGORITHM

The procedure for programming the 87C251SB is as follows:

1. Set up the controller for operation in the appropriate mode according to Table 13-1.
2. Input the 16-bit address on ports 1 and 3.
3. Input the data byte on port 2.
4. Raise the voltage on the V_{PP} pin from 5 V to 12.75 V.
5. Pulse the PROG# pin 5 times for the on-chip code memory and the configuration bytes, and 25 times for the encryption array and the lock bits.
6. Reduce the voltage on the V_{PP} pin to 5 V.

7. If the procedure is program/immediate-verify, go to “Verify Algorithm” on page 13-5 and perform steps 1 through 4 to verify the currently addressed byte. Make sure the voltage on the EA#/V_{PP} pin has been lowered to 5 V before performing the verifying procedure.
8. Repeat steps 1 through 7 until all memory locations are programmed.

13.5 VERIFY ALGORITHM

Use this procedure to verify user program code, signature bytes, configuration bytes and lock bits stored in nonvolatile memory on the 8XC251SB. To preserve the secrecy of the encryption key byte sequence, the encryption array can not be verified. Verification can be performed on bytes as they are programmed, or on a block of bytes that have been previously programmed. The procedure for verifying the 8XC251SB is as follows:

1. Set up the controller for operation in the appropriate mode according to Table 13-1.
2. Input the 16-bit address on ports P1 and P3.
3. Wait for the data on port P2 to become valid (T_{AVQV} = 48 clock cycles, Figure 13-5), then compare the data with the expected value.
4. If the procedure is program/immediate-verify, return to step 8 of “OTPROM Programming Algorithm” on page 13-4 to program the next byte.
5. Repeat steps 1 through 5 until all memory locations are verified.

13.6 PROGRAMMABLE FUNCTIONS

This section discusses factors related to programming and verifying the various nonvolatile memory functions.

13.6.1 On-chip Code Memory

The 16-Kbyte on-chip code memory is located in the top region of the memory space starting at address FF:0000H. At reset, the 87C251SB and 83C251SB devices vector to this address. See Chapter 3 for detailed information on the 8XC251SB memory space.

To enter user program code and data in the on-chip code memory, perform the procedure described in “OTPROM Programming Algorithm” on page 13-4 using the program on-chip code memory mode (Table 13-1).

To verify that the on-chip code memory is correctly programmed, perform the procedure described in “Verify Algorithm” on page 13-5 using the verify on-chip code memory mode (Table 13-1).

13.6.2 Configuration Bytes

The MCS® 251 microcontroller contains four configuration bytes, CONFIG0 through CONFIG3, implemented in OTPROM. CONFIG0 through CONFIG3 correspond to addresses 0080H through 0083H in Table 13-1. The configuration bytes are located in nonvolatile memory outside the memory address space and are inaccessible by user code. CONFIG0 and CONFIG1 specify the following:

- WSA,WSB. Wait states
- RD0, RD1. This two-bit code determines the address ranges for RD# and PSEN# and selects a 16-bit or 17-bit external bus. RD# as 17th address bit (A16), P3.7 as general purpose pin.
- XALE. Extends ALE pulse.
- SRC. Source code/ binary code
- EMAP. Maps upper 8 Kbytes of on-chip code memory to region 00H.
- PAGE. Page mode select, external bus structure
- INTR. Return from interrupt

CONFIG2 and CONFIG3 are reserved for future use. See Figure 13-3 and Figure 13-4 for CONFIG0 and CONFIG1 bit assignments and definitions. These figures also give the configuration values for making the 8XC251SB pin compatible with the 8XC51FB and 8XC54. Table 13-2 lists the CONFIG0 and CONFIG1 values for the 80C251SB.

To program the configuration bytes, perform the procedure described in "OTPROM Programming Algorithm" on page 13-4 using the program configuration byte mode (Table 13-1).

To verify that the configuration bytes are correctly programmed, perform the procedure described in "Verify Algorithm" on page 13-5 using the verify configuration byte mode (Table 13-1).

CONFIG0

7							0
—	—	WSA	XALE	RD1	RD0	PAGE	SRC

Bit Number	Bit Mnemonic	Function
7:6	—	Reserved: Set these bits when writing to CONFIG0.
5	WSA	Wait State A: Clear this bit to generate one external wait state for memory regions 00:, FE:, and FF:. Set this bit for no wait states for these regions.
4	XALE	Extend ALE: If this bit is set, the time of the ALE pulse is T_{OSC}. Clearing this bit extends the time of the ALE pulse from T_{OSC} to $3T_{OSC}$, which adds one external wait state.
3:2	RD1, RD0	RD# and PSEN# Function Select: **RD1 · RD0 · RD# Range · PSEN# Range · Features** 0 · 0 · Reserved · Reserved · Reserved 0 · 1 · RD# = A16 · All addresses · 128-Kbyte External Address Space 1 · 0 · P3.7 only · All addresses · One additional port pin 1 · 1 · ≤ 7F:FFFFH · ≥ 80:0000H · Compatible with MCS 51 microcontrollers
1	PAGE	Page Mode Select: Clear this bit for page-mode (A15:8/D7:0 on P2, and A7:0 on P0). Set this bit for nonpage-mode (A15:8 on P2, and A7:0/D7:0 on P0 (compatible with 44-pin PLCC MCS 51 microcontrollers)).
0	SRC	Source Mode/Binary Mode Select: Set this bit for source mode. Clear this bit for binary mode (binary-code compatible with MCS 51 microcontrollers).

NOTE: To make the 8XC251SB pin compatible with 44-pin PLCC MCS 51 microcontrollers, use the following bit values in CONFIG0: 1101 1110B.

Figure 13-3. Configuration Byte 0

CONFIG1

7							0
—	—	—	INTR	WSB	—	—	EMAP

Bit Number	Bit Mnemonic	Function
7:5	—	Reserved: Set these bits when writing to CONFIG1.
4	INTR	Interrupt Mode: If this bit is set, interrupts push 4 bytes onto the stack (the 3 bytes of the PC register and the PSW1 register). If this byte is clear, interrupts push 2 bytes onto the stack (the 2 lower bytes of the PC register).
3	WSB	Wait State B: Clear this bit to generate one external wait state for memory region 01:. Set this bit for no wait states for region 01:.
2:1		Reserved: Set these bits when writing to CONFIG1.
0	EMAP	EPROM MAP: Clearing this bit maps the upper 8 Kbytes of on-chip code memory (FF:2000H–FF:3FFFH) to 00:E000H–00:FFFFH. If this bit is set, the upper 8 Kbytes of on-chip code memory are mapped only to FF:2000H–FF:3FFFH.

NOTE: To make the 8XC251SB pin compatible with 44-pin PLCC MCS 51 microcontrollers, use the following bit values in CONFIG1: 1110 0111B.

Figure 13-4. Configuration Byte 1

Table 13-2. Configuration Byte Values for 80C251SB and 80C251SB-16

Bit Number	CONFIG0 (1)		CONFIG1 (1)	
	Bit Mnemonic	Value	Bit Mnemonic	Value
7	Reserved	1	Reserved	1
6	Reserved	1	Reserved	1
5	WSA	(2)	Reserved	1
4	XALE	1	INTR	0
3	RD1	1	WSB	(2)
2	RD0	1	Reserved	1
1	PAGE	1	Reserved	1
0	SRC	0	EMAP	1

NOTE:

1. In addition to the configuration given in the table, the 80C251SB and 80C251SB-16 are available in user-defined configurations.
2. The 80C251SB is available with no wait states (WSA = WSB = 1).
   The 80C251SB-16 is available with one wait state (WSA = WSB = 0).

13.6.3 Lock Bit System

The 87C251SB provides a three-level lock system for protecting user program code stored in the on-chip code memory from unauthorized access. On the 83C251SB, only LB1 protection is available. Table 13-3 describes the levels of protection.

To program the lock bits, perform the procedure described in “OTPROM Programming Algorithm” on page 13-4 using the program lock bits mode (Table 13-1).

To verify that the lock bits are correctly programmed, perform the procedure described in “Verify Algorithm” on page 13-5 using the verify lock bits mode (Table 13-1).

Table 13-3. Lock Bit Function

	Lock Bits Programmed			Protection Type
	LB3	LB2	LB1	
Level 1	U	U	U	No program lock features are enabled. On-chip user code is encrypted when verified, if encryption array is programmed.
Level 2	U	U	P	External code is prevented from fetching code bytes from on-chip code memory. Further programming of the on-chip OTPROM is disabled.
Level 3	U	P	P	Same as level 2, plus on-chip code memory verify is disabled.
Level 4	P	P	P	Same as level 3, plus external memory execution is disabled.

NOTE: Other combinations of the lock bits are not defined.

13.6.4 Encryption Array

The 87C251SB and 83C251SB controllers include a 128-byte encryption array located in nonvolatile memory outside the memory address space. During verification of the on-chip code memory, the seven low-order address bits also address the encryption array. As the byte of the code memory is read, it is exclusive-NOR'ed (XNOR) with the key byte from the encryption array. If the encryption array is not programmed (still all 1s), the user program code is placed on the data bus in its original, unencrypted form. If the encryption array is programmed with key bytes, the user program code is encrypted and can't be used without know the key byte sequence.

CAUTION

> If the encryption feature is implemented, the portion of the on-chip code memory that does not contain program code should be filled with "random" byte values other than FFH to prevent the encryption key sequence from being revealed.

To program the encryption array, perform the procedure described in "OTPROM Programming Algorithm" on page 13-4 using the program encryption array mode (Table 13-1).

To verify that the configuration bytes are correctly programmed, perform the procedure described in "Verify Algorithm" on page 13-5 using the verify encryption array mode (Table 13-1).

13.6.5 Signature Bytes

The 87C251SB and 83C251SB contain factory-programmed signature bytes. These bytes are located at 30H, 31H, and 60H in nonvolatile memory outside the memory address space. To read the signature bytes, perform the procedure described in "Verify Algorithm" on page 13-5 using the verify signature mode (Table 13-1). Signature byte values are listed in Table 13-4.

Table 13-4. Contents of the Signature Bytes

Device	Address		
	30H	**31H**	**60H**
83C251SB	89H	40H	7BH
87C251SB	89H	40H	FBH

13.7 VERIFYING THE 83C251SB (ROM)

Nonvolatile memory on the 83C251SB controller is factory programmed. The verification procedure for the 83C251SB is exactly the same as for the 87C251SB OTPROM version. The setup shown in Figure 13-1 applies as do the waveform and timing diagrams. Like the 87C251SB, the 83C251SB contains a 16-Kbyte on-chip code memory and a 128-byte encryption array.

For information on verifying the contents of nonvolatile memory on the 83C251SB, see “Programmable Functions” on page 13-5 for each function desired. Or more directly, perform the verification procedure described in “Verify Algorithm” on page 13-5 using the appropriate verify mode (Table 13-1).

13.8 VERIFYING THE 80C251SB (ROMLESS)

The configuration bytes stored in nonvolatile memory on the 80C251SB can be read using the verify procedure presented in this chapter. For information regarding the configuration bytes see “Configuration Bytes” on page 13-6.

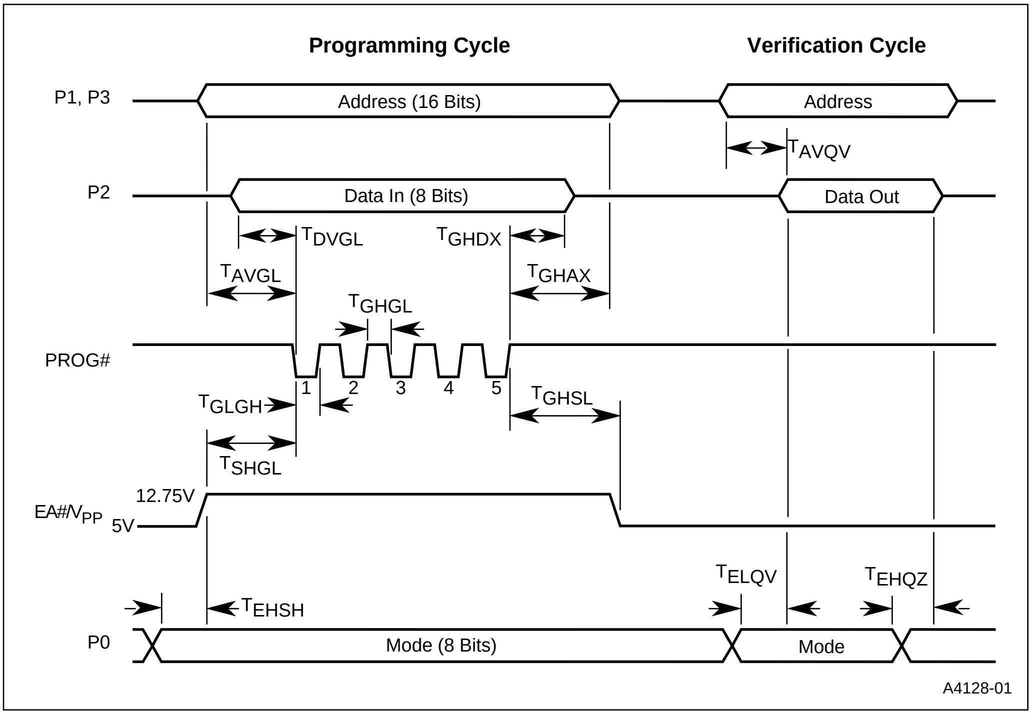

Programming Cycle
Verification Cycle
P1, P3
Address (16 Bits)
Address
TAVQV
P2
Data In (8 Bits)
Data Out
TDVGL
TGHDX
TAVGL
TGHAX
TGHGL
PROG#
1
2
3
4
5
TGLGH
TGHSL
TSHGL
12.75V
EA#/VPP
5V
TELQV
TEHQZ
TEHSH
P0
Mode (8 Bits)
Mode
A4128-01

Figure 13-5. OTPROM Timing

Table 13-5. OTPROM Timing Definitions

Symbol	Definition	Symbol	Definition
$1/T_{CLCL}$	Oscillator Frequency	T_{GHAX}	Address Hold after PROG#:
T_{AVGL}	Address Setup to PROG# Low	T_{GHDX}	Data Hold after PROG#
T_{AVQV}	Address to Data Valid	T_{GHSL}	V_{PP} Hold after PROG#
T_{DVGL}	Data Setup to PROG# Low	T_{GHGL}	PROG# High to PROG# Low
T_{EHSH}	ENABLE High to V_{PP}	T_{GLGH}	PROG# Width
T_{EHQZ}	Data Float after ENABLE	T_{SHGL}	V_{PP} Setup to PROG# Low:
T_{ELQV}	ENABLE Low to Data Valid		

NOTE: A = Address, D = Data, E = Enable, G = PROG#, H = High, L = Low, Q = Data out, S = Supply (V_{PP}), V = Valid, X = No longer valid, Z = Floating.

A

Instruction Set Reference

APPENDIX A
INSTRUCTION SET REFERENCE

This appendix contains reference material for the instructions in the MCS® 251 architecture. It includes an opcode map, a summary of the instructions — with instruction lengths and execution times — and a detailed description of each instruction. It contains the following tables:

- Tables A-1 through A-4 describe the notation used for the instruction operands.
- Table A-6 on page A-4 and Table A-7 on page A-5 comprise the opcode map for the instruction set.
- Table A-8 on page A-6 through Table A-17 on page A-10 contain supporting material for the opcode map.
- Table A-18 on page A-12 lists execution times for a group of instructions that access the port SFRs.
- The following tables list the instructions with their lengths in bytes and their execution times:
  - Add and Subtract Instructions, Table A-19 on page A-14
  - Compare Instructions, Table A-20 on page A-15
  - Increment and Decrement Instructions, Table A-21 on page A-16
  - Multiply, Divide, and Decimal-adjust Instructions, Table A-22 on page A-16
  - Logical Instructions, Table A-23 on page A-17
  - Move Instructions, Table A-24 on page A-19
  - Exchange, Push, and Pop Instructions, Table A-24 on page A-19
  - Bit Instructions, Table A-26 on page A-23
  - Control Instructions, Table A-27 on page A-24

“Instruction Descriptions” on page A-26 contains a detailed description of each instruction.

NOTE

> The instruction execution times given in this appendix are for code executing from on-chip code memory and for data that is read from and written to on-chip RAM. Execution times are increased by executing code from external memory, accessing peripheral SFRs, accessing data in external memory, using a wait state, or extending the ALE pulse.
>
> For some instructions, accessing the port SFRs, P*x*, *x* = 0–3, increases the execution time. These cases are listed in Table A-18 on page A-12 and are noted in the instruction summary tables and the instruction descriptions.

A.1 NOTATION FOR INSTRUCTION OPERANDS

Table A-1. Notation for Register Operands

Register Notation		MCS® 251 Arch.	MCS 51 Arch.
@Ri	A memory location (00H–FFH) addressed indirectly via byte register R0 or R1		✔
Rn	Byte register R0–R7 of the currently selected register bank		✔
n	Byte register index: n = 0–7		
r r r	Binary representation of n		
Rm	Byte register R0–R15 of the currently selected register file	✔	
Rmd	Destination register		
Rms	Source register		
m, md, ms	Byte register index: m, md, ms = 0–15		
s s s s	Binary representation of m or md		
S S S S	Binary representation of ms		
WRj	Word register WR0, WR2, ..., WR30 of the currently selected register file	✔	
WRjd	Destination register		
WRjs	Source register		
@WRj	A memory location (00:0000H–00:FFFFH) addressed indirectly through word register WR0–WR30		
@WRj +dis16	Data RAM location (00:0000H–00:FFFFH) addressed indirectly through a word register (WR0–WR30) + displacement value		
j, jd, js	Word register index: j, jd, js = 0–30		
t t t t	Binary representation of j or jd		
T T T T	Binary representation of js		
DRk	Dword register DR0, DR4, ..., DR28, DR56, DR60 of the currently selected register file	✔	
DRkd	Destination Register		
DRks	Source Register		
@DRk	A memory location (00:0000H–FF:FFFFH) addressed Indirectly through dword register DR0–DR28, DR56, DR60		
@DRk +dis24	Data RAM location (00:0000H–FF:FFFFH) addressed indirectly through a dword register (DR0–DR28, DR56, DR60) + displacement value		
k, kd, ks	Dword register index: k, kd, ks = 0, 4, 8, ..., 28, 56, 60		
u u u u	Binary representation of k or kd		
U U U U	Binary representation of ks		

Table A-2. Notation for Direct Addresses

Direct Address.	Description	MCS® 251 Arch.	MCS 51 Arch.
dir8	An 8-bit direct address. This can be a memory address (00:0000H–00:00FFH) or an SFR address (S:00H - S:FFH).	✔	✔
dir16	A 16-bit memory address (00:0000H–00:FFFFH) used in direct addressing.	✔	

Table A-3. Notation for Immediate Addressing

Immediate Data	Description	MCS® 251 Arch.	MCS 51 Arch.
#data	An 8-bit constant that is immediately addressed in an instruction.	✔	✔
#data16	A 16-bit constant that is immediately addressed in an instruction.	✔	
#0data16 #1data16	A 32-bit constant that is immediately addressed in an instruction. The upper word is filled with zeros (#0data16) or ones (#1data16).	✔	
#short v v	A constant, equal to 1, 2, or 4, that is immediately addressed in an instruction. Binary representation of #short.	✔	

Table A-4. Notation for Bit Addressing

Bit Address	Description	MCS® 251 Arch.	MCS 51 Arch.
bit y y y	A directly addressed bit in memory locations 00:0020H–00:007FH or in any defined SFR. A binary representation of the bit number (0–7) within a byte.	✔	
bit51	A directly addressed bit (bit number = 00H–FFH) in memory or an SFR. Bits 00H–7FH are the 128 bits in byte locations 20H–2FH in the on-chip RAM. Bits 80H–FFH are the 128 bits in the 16 SFR's with addresses that end in 0H or 8H: S:80H, S:88H, S:90H, . . . , S:F0H, S:F8H.		✔

Table A-5. Notation for Destinations in Control Instructions

Destination Address	Description	MCS® 251 Arch.	MCS 51 Arch.
rel	A signed (two's complement) 8-bit relative address. The destination is -128 to +127 bytes relative to first byte of the next instruction.	✔	✔
addr11	An 11-bit destination address. The destination is in the same 2-Kbyte block of memory as the first byte of the next instruction.	✔	✔
addr16	A 16-bit destination address. A destination can be anywhere within the same 64-Kbyte region as the first byte of the next instruction.	✔	✔
addr24	A 24-bit destination address. A destination can be anywhere within the 16-Mbyte address space.	✔	

A.2 OPCODE MAP AND SUPPORTING TABLES

Table A-6. Instructions for MCS® 51 Microcontrollers

Bin.	**0**	**1**	**2**	**3**	**4**	**5**	**6-7**	**8-F**
Src.	**0**	**1**	**2**	**3**	**4**	**5**	**A5*x*6–A5*x*7**	**A5*x*8–A5*x*F**
0	NOP	AJMP addr11	LJMP addr16	RR A	INC A	INC dir8	INC @Ri	INC Rn
1	JBC bit,rel	ACALL addr11	LCALL addr16	RRC A	DEC A	DEC dir8	DEC @Ri	DEC Rn
2	JB bit,rel	AJMP addr11	RET	RLA	ADD A,#data	ADD A,dir8	ADD A,@Ri	ADD A,Rn
3	JNB bit,rel	ACALL addr11	RETI	RLCA	ADDC A,#data	ADDC A,dir8	ADDC A,@Ri	ADDC A,Rn
4	JC rel	AJMP addr11	ORL dir8,A	ORL dir8,#data	ORL A,#data	ORL A,dir8	ORL A,@Ri	ORL A,Rn
5	JNC rel	ACALL addr11	ANL dir8,A	ANL dir8,#data	ANL A,#data	ANL A,dir8	ANL A,@Ri	ANL A,Rn
6	JZ rel	AJMP addr11	XRL dir8,A	XRL dir8,#data	XRL A,#data	XRL A,dir8	XRL A,@Ri	XRL A,Rn
7	JNZ rel	ACALL addr11	ORL CY,bit	JMP @A+DPTR	MOV A,#data	MOV dir8,#data	MOV @Ri,#data	MOV Rn,#data
8	SJMP rel	AJMP addr11	ANL CY,bit	MOVC A,@A+PC	DIV AB	MOV dir8,dir8	MOV dir8,@Ri	MOV dir8,Rn
9	MOV DPTR,#data16	ACALL addr11	MOV bit,CY	MOVC A,@A+DPTR	SUBB A,#data	SUBB A,dir8	SUBB A,@Ri	SUBB A,Rn
A	ORL CY,bit	AJMP addr11	MOV CY,bit	INC DPTR	MUL AB	ESC	MOV @Ri,dir8	MOV Rn,dir8
B	ANL CY,bit	ACALL addr11	CPL bit	CPL CY	CJNE A,#data,rel	CJNE A,dir8,rel	CJNE @Ri,#data,rel	CJNE Rn,#data,rel
C	PUSH dir8	AJMP addr11	CLR bit	CLR CY	SWAP A	XCH A,dir8	XCH A,@Ri	XCH A,Rn
D	POP dir8	ACALL addr11	SETB bit	SETB CY	DA A	DJNZ dir8,rel	XCHD A,@Ri	DJNZ Rn,rel
E	MOVX A,@DPTR	AJMP addr11	MOVX A,@Ri		CLR A	MOV A,dir8	MOV A,@Ri	MOV A,Rn
F	MOV @DPT,A	ACALL addr11	MOVX @Ri,A		CPL A	MOV dir8,A	MOV @Ri,A	MOV Rn,A

Table A-7. New Instructions for the MCS® 251 Architecture

Bin.	A5*x*8	A5*x*9	A5*x*A	A5*x*B	A5*x*C	A5*x*D	A5*x*E	A5*x*F
Src.	***x*8**	***x*9**	***x*A**	***x*B**	***x*C**	***x*D**	***x*E**	***x*F**
0	JSLE rel	MOV Rm,@WRj+dis	MOVZ WRj,Rm	INC R,#short (1) MOV reg,ind			SRA reg	
1	JSG rel	MOV @WRj+dis,Rm	MOVS WRj,Rm	DEC R,#short (1) MOV ind,reg			SRL reg	
2	JLE rel	MOV Rm,@DRk+dis			ADD Rm,Rm	ADD WRj,WRj	ADD reg,op2 (2)	ADD DRk,DRk
3	JG rel	MOV @DRk+dis,Rm					SLL reg	
4	JSL rel	MOV WRj,@WRj+dis			ORL Rm,Rm	ORL WRj,WRj	ORL reg,op2 (2)	
5	JSGE rel	MOV @WRj+dis,WRj			ANL Rm,Rm	ANL WRj,WRj	ANL reg,op2 (2)	
6	JE rel	MOV WRj,@DRk+dis			XRL Rm,Rm	XRL WRj,WRj	XRL reg,op2 (2)	
7	JNE rel	MOV @DRk+dis,WRj	MOV op1,reg (2)		MOV Rm,Rm	MOV WRj,WRj	MOV reg,op2 (2)	MOV DRk,DRk
8		LJMP @WRj EJMP @DRk	EJMP addr24		DIV Rm,Rm	DIV WRj,WRj		
9		LCALL@WRj ECALL @DRk	ECALL addr24		SUB Rm,Rm	SUB WRj,WRj	SUB reg,op2 (2)	SUB DRk,DRk
A		Bit Instructions (3)	ERET		MUL Rm,Rm	MUL WRj,WRj		
B		TRAP			CMP Rm,Rm	CMP WRj,WRj	CMP reg,op2 (2)	CMP DRk,DRk
C			PUSH op1 (4) MOV DRk,PC					
D			POP op1 (4)					
E								
F								

NOTES:

1. R = Rm/WRj/DRk.
2. op1, op2 are defined in Table A-8 on page A-6.
3. See Tables A-10 and A-11 on page A-7.
4. See Table A-12 on page A-8.

Table A-8. Data Instructions

Instruction	Byte 0		Byte 1		Byte 2		Byte 3
Oper Rmd,Rms	x	C	md	ms			
Oper WRjd,WRjs	x	D	jd/2	js/2			
Oper DRkd,DRks	x	F	kd/4	ks/4			
Oper Rm,#data	x	E	m	0000	#data		
Oper WRj,#data16	x	E	j/2	0100	#data (high)		#data (low)
Oper DRk,#data16	x	E	k/4	1100	#data (high)		#data (low)
MOV DRk(h),#data16	7	A	k/4	1000	#data (high)		#data (low)
MOV DRk,#1data16	7	E					
CMP DRk,#1data16	B	E					
Oper Rm,dir8	x	E	m	0001	dir8 addr		
Oper WRj,dir8	x	E	j/2	0101	dir8 addr		
Oper DRk,dir8	x	E	k/4	1101	dir8 addr		
Oper Rm,dir16	x	E	m	0011	dir16 addr (high)		dir16 addr (low)
Oper WRj,dir16	x	E	j/2	0111	dir16 addr (high)		dir16 addr (low)
Oper DRk,dir16	x	E	k/4	1111	dir16 addr (high)		dir16 addr (low)
Oper Rm,@WRj	x	E	j/2	1001	m	00	
Oper Rm,@DRk	x	E	k/4	1011	m	00	

Table A-9. High Nibble, Byte 0 of Data Instructions

x	Operation	Notes
2	ADD reg,op2	All addressing modes are supported.
9	SUB reg,op2	
B	CMP reg,op2	
4	ORL reg,op2	
5	ANL reg,op2	
6	XRL reg,op2	
7	MOV reg,op2	
8	DIV reg,op2	Two modes only: reg,op2 = Rmd,Rms reg,op2 = Wjd,Wjs
A	MUL reg,op2	

All of the bit instructions in the MCS 251 architecture (Table A-7) have opcode A9, which serves as an escape byte (similar to A5). The high nibble of byte 1 specifies the bit instruction, as given in Table A-10.

Table A-10. Bit Instructions

	Instruction	**Byte 0(*x*)**		**Byte 1**			**Byte 2**	**Byte 3**
1	Bit Instr (dir8)	A	9	*xxxx*	0	bit	dir8 addr	

Table A-11. Byte 1 (High Nibble) for Bit Instructions

xxxx	**Bit Instruction**
0001	JBC bit
0010	JB bit
0011	JNB bit
0111	ORL CY,bit
1000	ANL CY,bit
1001	MOV bit,CY
1010	MOV CY,bit
1011	CPL bit
1100	CLR bit
1101	SETB bit
1110	ORL CY, /bit
1111	ANL CY, /bit

Table A-12. PUSH/POP Instructions

Instruction	Byte 0(x)		Byte 1		Byte 2	Byte 3
PUSH #data	C	A	0000	0010	#data	
PUSH #data16	C	A	0000	0110	#data16 (high)	#data16 (low)
PUSH Rm	C	A	m	1000		
PUSH WRj	C	A	j/2	1001		
PUSH DRk	C	A	k/4	1011		
MOV DRk,PC	C	A	k/4	0001		
POP Rm	D	A	m	1000		
POP WRj	D	A	j/2	1001		
POP DRk	D	A	k/4	1011		

Table A-13. Control Instructions

Instruction	Byte 0(*x*)		Byte 1		Byte 2	Byte 3
EJMP addr24	8	A	addr[23:16]		addr[15:8]	addr[7:0]
ECALL addr24	9	A	addr[23:16]		addr[15:8]	addr[7:0]
LJMP @WRj	8	9	j/2	0100		
LCALL @WRj	9	9	j/2	0100		
EJMP @DRk	8	9	k/4	1000		
ECALL @DRk	8	9	k/4	1000		
ERET	A	A				
JE rel	8	8	rel			
JNE rel	7	8	rel			
JLE rel	2	8	rel			
JG rel	3	8	rel			
JSL rel	4	8	rel			
JSGE rel	5	8	rel			
JSLE rel	0	8	rel			
JSG rel	1	8	rel			
TRAP	B	9				

Table A-14. Displacement/Extended MOVs

Instruction	Byte 0		Byte 1		Byte 2		Byte 3
MOV Rm,@WRj+dis	0	9	m	j/2	dis[15:8]		dis[7:0]
MOV WRk,@WRj+dis	4	9	j/2	k2	dis[15:8]		dis[7:0]
MOV Rm,@DRk+dis	2	9	m	k/4	dis[15:8]		dis[7:0]
MOV WRj,@DRk+dis	6	9	j/2	k/4	dis[15:8]		dis[7:0]
MOV @WRj+dis,Rm	1	9	m	j/2	dis[15:8]		dis[7:0]
MOV @WRj+dis,WRk	5	9	j/2	k2	dis[15:8]		dis[7:0]
MOV @DRk+dis,Rm	3	9	m	k/4	dis[15:8]		dis[7:0]
MOV @DRk+dis,WRj	7	9	j/2	k/4	dis[15:8]		dis[7:0]
MOVS WRj,Rm	1	A	j/2	m			
MOVZ WRj,Rm	0	A	j/2	m			
MOV WRj,@WRj	0	B	j/2	1000	j/2	0000	
MOV WRj,@DRk	0	B	k/4	1010	j/2	0000	
MOV @WRj,WRj	1	B	j/2	1000	j/2	0000	
MOV @DRk,WRj	1	B	k/4	1010	j/2	0000	
MOV dir8,Rm	7	A	m	0001	dir8 addr		
MOV dir8,WRj	7	A	j/2	0101	dir8 addr		
MOV dir8,DRk	7	A	k/4	1101	dir8 addr		
MOV dir16,Rm	7	A	m	0011	dir16 addr (high)		dir16 addr (low)
MOV dir16,WRj	7	A	j/2	0111	dir16 addr (high)		dir16 addr (low)
MOV dir16,DRk	7	A	k/4	1111	dir16 addr (high)		dir16 addr (low)
MOV @WRj,Rm	7	A	j/2	1001	m	0000	
MOV @DRk,Rm	7	A	k/4	1011	m	0000	

Table A-15. INC/DEC

	Instruction	Byte 0		Byte 1		
1	INC Rm,#short	0	B	m	00	ss
2	INC WRj,#short	0	B	j/2	01	ss
3	INC DRk,#short	0	B	k/4	11	ss
4	DEC Rm,#short	1	B	m	00	ss
5	DEC WRj,#short	1	B	j/2	01	ss
6	DEC DRk,#short	1	B	k/4	11	ss

Table A-16. Encoding for INC/DEC

ss	#short
00	1
01	2
10	4

Table A-17. Shifts

	Instruction	Byte 0		Byte 1	
1	SRA Rm	0	E	m	0000
2	SRA WRj	0	E	j/2	0100
3	SRL Rm	1	E	m	0000
4	SRL WRj	1	E	j/2	0100
5	SLL Rm	3	E	m	0000
6	SLL WRj	3	E	j/2	0100

A.3 INSTRUCTION SET SUMMARY

This section contains tables that summarize the instruction set. For each instruction there is a short description, its length in bytes, and its execution time in states.

NOTE

> The instruction execution times given in the tables are for code executing from on-chip code memory and for data that is read from and written to on-chip RAM. Execution times are increased by executing code from external memory, accessing peripheral SFRs, accessing data in external memory, using a wait state, or extending the ALE pulse.
>
> For some instructions, accessing the port SFRs, P*x*, *x* = 0–3, increases the execution time. These cases are noted individually in the tables.

A.3.1 Execution Times for Instructions that Access the Port SFRs

The execution times for some instructions increase when the instruction accesses a port SFR (P*x*, *x* = 0–3) as opposed to any other SFR. Table A-18 lists these instructions and the execution times for Case 0:

- Case 0. Code executes from on-chip OTPROM/ROM and accesses locations in on-chip data RAM. The port SFRs are not accessed.

In Cases 1–4, the instructions access a port SFR:

- Case 1. Code executes from on-chip OTPROM/ROM and accesses a port SFR.
- Case 2. Code executes from external memory with no wait state and a short ALE (not extended) and accesses a port SFR.
- Case 3. Code executes from external memory with one wait state and a short ALE (not extended) and accesses a port SFR.
- Case 4. Code executes from external memory with one wait state and an extended ALE, and accesses a port SFR.

The times for Cases 1 through 4 are expressed as the number of state times to add to the state times for given for Case 0.

Table A-18. State Times to Access the Port SFRs

Instruction	Case 0 Execution Times		Additional State Times			
	Binary	Source	Case 1	Case 2	Case 3	Case 4
ADD A,dir8	1	1	1	2	3	4
ADD Rm,dir8	3	2	1	2	3	4
ADDC A,dir8	1	1	1	2	3	4
ANL A,dir8	1	1	1	2	3	4
ANL CY,bit	3	2	1	2	3	4
ANL CY,bit51	1	1	1	2	3	4
ANL CY,/bit	3	2	1	2	3	4
ANL CY,/bit51	1	1	1	2	3	4
ANL dir8,#data	3	3	2	4	6	8
ANL dir8,A	2	2	2	4	6	8
ANL Rm,dir8	3	2	1	2	3	4
CLR bit	4	3	2	4	6	8
CLR bit51	2	2	2	4	6	8
CMP Rm,dir8	3	2	1	2	3	4
CPL bit	4	3	2	4	6	8
CPL bit51	2	2	2	4	6	8
DEC dir8	2	2	2	4	6	8
INC dir8	2	2	2	4	6	8
MOV A,dir8	1	1	1	2	3	4
MOV bit,CY	4	3	2	4	6	8
MOV bit51,CY	2	2	2	4	6	8
MOV CY,bit	3	2	1	2	3	4
MOV CY,bit51	1	1	1	2	3	4
MOV dir8,#data	3	3	1	2	3	4
MOV dir8,A	2	2	1	2	3	4
MOV dir8,Rm	4	3	1	2	3	4
MOV dir8,Rn	2	3	1	2	3	4
MOV Rm,dir8	3	2	1	2	3	4
MOV Rn,dir8	1	2	1	2	3	4
ORL A,dir8	1	1	1	2	3	4
ORL CY,bit	3	2	1	2	3	4
ORL CY,bit51	1	1	1	2	3	4
ORL CY,/bit	3	2	1	2	3	4

Table A-18. State Times to Access the Port SFRs (Continued)

Instruction	Case 0 Execution Times		Additional State Times			
	Binary	Source	Case 1	Case 2	Case 3	Case 4
ORL CY,/bit51	1	1	1	2	3	4
ORL dir8,#data	3	3	1	2	3	4
ORL dir8,A	2	2	2	4	6	8
ORL Rm,dir8	3	2	1	2	3	4
SETB bit	4	3	2	4	6	8
SETB bit51	2	2	2	4	6	8
SUB Rm,dir8	3	2	1	2	3	4
SUBB A,dir8	1	1	1	2	3	4
XCH A,dir8	3	3	2	4	6	8
XRL A,dir8	1	1	1	2	3	4
XRL dir8,#data	3	3	2	4	6	8
XRL dir8,A	2	2	2	4	6	8
XRL Rm,dir8	3	2	1	2	3	4

A.3.2 Instruction Summaries

Table A-19. Summary of Add and Subtract Instructions

Add	**ADD <dest>,<src>**	**dest opnd ← dest opnd + src opnd**
Subtract	**SUB <dest>,<src>**	**dest opnd ← dest opnd - src opnd**
Add with Carry	**ADDC <dest>,<src>**	**(A) ← (A) + src opnd + carry bit**
Subtract with Borrow	**SUBB <dest>,<src>**	**(A) ← (A) - src opnd - carry bit**

Mnemonic	**<dest>,<src>**	**Notes**	**Binary Mode**		**Source Mode**	
			Bytes	**States**	**Bytes**	**States**
ADD	A,Rn	Reg to acc	1	1	2	2
	A,dir8	Dir byte to acc	2	1 (2)	2	1 (2)
	A,@Ri	Indir addr to acc	1	2	2	3
	A,#data	Immediate data to acc	2	1	2	1
ADD; SUB	Rmd,Rms	Byte reg to/from byte reg	3	2	2	1
	WRjd,WRjs	Word reg to/from word reg	3	3	2	2
	DRkd,DRks	Dword reg to/from dword reg	3	5	2	4
	Rm,#data	Immediate 8-bit data to/from byte reg	4	3	3	2
	WRj,#data16	Immediate 16-bit data to/from word reg	5	4	4	3
	DRk,#0data16	16-bit unsigned immediate data to/from dword reg	5	6	4	5
	Rm,dir8	Dir addr to/from byte reg	4	3 (2)	3	2 (2)
	WRj,dir8	Dir addr to/from word reg	4	4	3	3
	Rm,dir16	Dir addr (64K) to/from byte reg	5	3	4	2
	WRj,dir16	Dir addr (64K) to/from word reg	5	4	4	3
	Rm,@WRj	Indir addr (64K) to/from byte reg	4	3	3	2
	Rm,@DRk	Indir addr (16M) to/from byte reg	4	4	3	3
ADDC; SUBB	A,Rn	Reg to/from acc with carry	1	1	2	2
	A,dir8	Dir byte to/from acc with carry	2	1 (2)	2	1 (2)
	A,@Ri	Indir RAM to/from acc with carry	1	2	2	3
	A,#data	Immediate data to/from acc with carry	2	1	2	1

NOTES:

1. A shaded cell denotes an instruction in the MCS® 51 architecture.
2. If this instruction addresses an I/O port (P*x*, *x* = 0–3), add 1 to the number of states.

Table A-20. Summary of Compare Instructions

Compare	**CMP <dest>,<src>**	**dest opnd – src opnd**				
Mnemonic	**<dest>,<src>**	**Notes**	**Binary Mode**		**Source Mode**	
			Bytes	**States**	**Bytes**	**States**
CMP	Rmd,Rms	Reg with reg	3	2	2	1
	WRjd,WRjs	Word reg with word reg	3	3	2	2
	DRkd,DRks	Dword reg with dword reg	3	5	2	4
	Rm,#data	Reg with immediate data	4	3	3	2
	WRj,#data16	Word reg with immediate 16-bit data	5	4	4	3
	DRk,#0data16	Dword reg with zero-extended 16-bit immediate data	5	6	4	5
	DRk,#1data16	Dword reg with one-extended 16-bit immediate data	5	6	4	5
	Rm,dir8	Dir addr from byte reg	4	3†	3	2†
	WRj,dir8	Dir addr from word reg	4	4	3	3
	Rm,dir16	Dir addr (64K) from byte reg	5	3	4	2
	WRj,dir16	Dir addr (64K) from word reg	5	4	4	3
	Rm,@WRj	Indir addr (64K) from byte reg	4	3	3	2
	Rm,@DRk	Indir addr (16M) from byte reg	4	4	3	3

†If this instruction addresses an I/O port (P*x*, *x* = 0–3), add 1 to the number of states.

Table A-21. Summary of Increment and Decrement Instructions

Increment	**INC DPTR**	**(DPTR) ← (DPTR) + 1**
Increment	**INC byte**	**byte ← byte + 1**
Increment	**INC <dest>,<src>**	**dest opnd ← dest opnd + src opnd**
Decrement	**DEC byte**	**byte ← byte – 1**
Decrement	**DEC <dest>,<src>**	**dest opnd ← dest opnd - src opnd**

Mnemonic	**<dest>,<src>**	**Notes**	**Binary Mode**		**Source Mode**	
			Bytes	**States**	**Bytes**	**States**
INC; DEC	A	acc	1	1	1	1
	Rn	Reg	1	1	2	2
	dir8	Dir byte	2	2 (2)	2	2 (2)
	@Ri	Indir RAM	1	3	2	4
	Rm,#short	Byte reg by 1, 2, or 4	3	2	2	1
	WRj,#short	Word reg by 1, 2, or 4	3	2	2	1
	DRk,#short	Double word reg by 1, 2, or 4	3	4	2	3
INC	DPTR	Data pointer	1	1	1	1

NOTES:

1. A shaded cell denotes an instruction in the MCS® 51 architecture.
2. If this instruction addresses an I/O port (P*x*, *x* = 0–3), add 2 to the number of states.

Table A-22. Summary of Multiply, Divide, and Decimal-adjust Instructions

Multiply	**MUL <reg1,reg2>**	**(2)**
	MUL AB	**(B:A) = A x B**
Divide	**DIV <reg1>,<reg2>**	**(2)**
	DIV AB	**(A) = Quotient; (B) =Remainder**
Decimal-adjust ACC for Addition (BCD)	**DA A**	**(2)**

Mnemonic	**<dest>,<src>**	**Notes**	**Binary Mode**		**Source Mode**	
			Bytes	**States**	**Bytes**	**States**
MUL	AB	Multiply A and B	1	5	1	5
	Rmd,Rms	Multiply byte reg and byte reg	3	6	2	5
	WRjd,WRjs	Multiply word reg and word reg	3	12	2	11
DIV	AB	Divide A by B	1	10	1	10
	Rmd,Rms	Divide byte reg by byte reg	3	11	2	10
	WRjd,WRjs	Divide word reg by word reg	3	21	2	20
DA	A	Decimal adjust acc	1	1	1	1

NOTES:

1. A shaded cell denotes an instruction in the MCS® 51 architecture.
2. See “Instruction Descriptions” on page A-26

Table A-23. Summary of Logical Instructions

Logical AND	**ANL <dest>,<src>**	**dest opnd ←dest opnd Λ src opnd**
Logical OR	**ORL <dest>,<src>**	**dest opnd ← dest opnd V src opnd**
Logical Exclusive OR	**XRL <dest>,<src>**	**dest opnd ← dest opnd ∀ src opnd**
Clear	**CLR A**	**(A) ← 0**
Complement	**CPL A**	**(Ai) ← Ø(Ai)**
Rotate	**RXX A**	**(1)**
Shift	**SXX Rm or Wj**	**(1)**
SWAP	**A**	**A3:0 ↔ A7:4**

Mnemonic	**<dest>,<src>**	**Notes**	**Binary Mode**		**Source Mode**	
			Bytes	**States**	**Bytes**	**States**
ANL; ORL; XRL;	A,Rn	Reg to acc	1	1	2	2
	A,dir8	Dir byte to acc	2	1 (3)	2	1 (3)
	A,@Ri	Indir addr to acc	1	2	2	3
	A,#data	Immediate data to acc	2	1	2	1
	dir8,A	Acc to dir byte	2	2 (4)	2	2 (4)
	dir8,#data	Immediate data to dir byte	3	3 (4)	3	3 (4)
	Rmd,Rms	Byte reg to byte reg	3	2	2	1
	WRjd,WRjs	Word reg to word reg	3	3	2	2
	Rm,#data	8-bit data to byte reg	4	3	3	2
	WRj,#data16	16-bit data to word reg	5	4	4	3
	Rm,dir8	Dir addr to byte reg	4	3 (3)	3	2 (3)
	WRj,dir8	Dir addr to word reg	4	4	3	3
	Rm,dir16	Dir addr (64K) to byte reg	5	3	4	2
	WRj,dir16	Dir addr (64K) to word reg	5	4	4	3
	Rm,@WRj	Indir addr (64K) to byte reg	4	3	3	2
	Rm,@DRk	Indir addr (16M) to byte reg	4	4	3	3
CLR	A	Clear acc	1	1	1	1
CPL	A	Complement acc	1	1	1	1
RL	A	Rotate acc left	1	1	1	1
RLC	A	Rotate acc left through the carry	1	1	1	1
RR	A	Rotate acc right	1	1	1	1
RRC	A	Rotate acc right through the carry	1	1	1	1
SLL	Rm	Shift byte reg left	3	2	2	1
	WRj	Shift word reg left	3	2	2	1

NOTES:

1. See “Instruction Descriptions” on page A-26.
2. A shaded cell denotes an instruction in the MCS® 51 architecture.
3. If this instruction addresses an I/O port (P*x*, *x* = 0–3), add 1 to the number of states.
4. If this instruction addresses an I/O port (P*x*, *x* = 0–3), add 2 to the number of states.

Table A-23. Summary of Logical Instructions (Continued)

Logical AND	**ANL <dest>,<src>**	**dest opnd ←dest opnd Λ src opnd**
Logical OR	**ORL <dest>,<src>**	**dest opnd ← dest opnd V src opnd**
Logical Exclusive OR	**XRL <dest>,<src>**	**dest opnd ← dest opnd ∀ src opnd**
Clear	**CLR A**	**(A) ← 0**
Complement	**CPL A**	**(Ai) ← Ø(Ai)**
Rotate	**RXX A**	**(1)**
Shift	**SXX Rm or Wj**	**(1)**
SWAP	**A**	**A3:0 ↔ A7:4**

Mnemonic	**<dest>,<src>**	**Notes**	**Binary Mode**		**Source Mode**	
			Bytes	**States**	**Bytes**	**States**
SRA	Rm	Shift byte reg right through the MSB	3	2	2	1
	WRj	Shift word reg right through the MSB	3	2	2	1
SRL	Rm	Shift byte reg right	3	2	2	1
	WRj	Shift word reg right	3	2	2	1
SWAP	A	Swap nibbles within the acc	1	2	1	2

NOTES:
1. See “Instruction Descriptions” on page A-26.
2. A shaded cell denotes an instruction in the MCS® 51 architecture.
3. If this instruction addresses an I/O port (P*x*, *x* = 0–3), add 1 to the number of states.
4. If this instruction addresses an I/O port (P*x*, *x* = 0–3), add 2 to the number of states.

Table A-24. Summary of Move Instructions

Move (2)	**MOV <dest>,<src>**	**destination ← src opnd**
Move with Sign Extension	**MOVS <dest>,<src>**	**destination ← src opnd with sign extend**
Move with Zero Extension	**MOVZ <dest>,<src>**	**destination ← src opnd with zero extend**
Move Code Byte	**MOVC <dest>,<src>**	**A ← code byte**
Move to External Mem	**MOVX <dest>,<src>**	**external mem ← (A)**
Move from External Mem	**MOVX <dest>,<src>**	**A ← source opnd in external mem**

Mnemonic	**<dest>,<src>**	**Notes**	**Binary Mode**		**Source Mode**	
			Bytes	**States**	**Bytes**	**States**
MOV	A,Rn	Reg to acc	1	1	2	2
	A,dir8	Dir byte to acc	2	1 (3)	2	1 (3)
	A,@Ri	Indir RAM to acc	1	2	2	3
	A,#data	Immediate data to acc	2	1	2	1
	Rn,A	Acc to reg	1	1	2	2
	Rn,dir8	Dir byte to reg	2	1 (3)	3	2 (3)
	Rn,#data	Immediate data to reg	2	1	3	2
	dir8,A	Acc to dir byte	2	2 (3)	2	2 (3)
	dir8,Rn	Reg to dir byte	2	2 (3)	3	3 (3)
	dir8,dir8	Dir byte to dir byte	3	3	3	3
	dir8,@Ri	Indir RAM to dir byte	2	3	3	4
	dir8,#data	Immediate data to dir byte	3	3 (3)	3	3 (3)
	@Ri,A	Acc to indir RAM	1	3	2	4
	@Ri,dir8	Dir byte to indir RAM	2	3	3	4
	@Ri,#data	Immediate data to indir RAM	2	3	3	4
	DPTR,#data16	Load Data Pointer with a 16-bit const	3	2	3	2
	Rmd,Rms	Byte reg to byte reg	3	2	2	1
	WRjd,WRjs	Word reg to word reg	3	2	2	1
	DRkd,DRks	Dword reg to dword reg	3	3	2	2
	Rm,#data	8-bit immediate data to byte reg	4	3	3	2
	WRj,#data16	16-bit immediate data to word reg	5	3	4	2
	DRk,#0data16	zero-extended 16-bit immediate data to dword reg	5	5	4	4
	DRk,#1data16	one-extended 16-bit immediate data to dword reg	5	5	4	4

NOTES:

1. A shaded cell denotes an instruction in the MCS® 51 architecture.
2. Instructions that move bits are in Table A-26 on page A-23.
3. If this instruction addresses an I/O port (P*x*, *x* = 0–3), add 1 to the number of states.
4. External memory addressed by instructions in the MCS 51 architecture is in the region specified by DPXL (reset value = 01H). See “Compatibility with the MCS® 51 Architecture” on page 3-2.

Table A-24. Summary of Move Instructions (Continued)

Move (2)	**MOV <dest>,<src>**	**destination ← src opnd**
Move with Sign Extension	**MOVS <dest>,<src>**	**destination ← src opnd with sign extend**
Move with Zero Extension	**MOVZ <dest>,<src>**	**destination ← src opnd with zero extend**
Move Code Byte	**MOVC <dest>,<src>**	**A ← code byte**
Move to External Mem	**MOVX <dest>,<src>**	**external mem ← (A)**
Move from External Mem	**MOVX <dest>,<src>**	**A ← source opnd in external mem**

Mnemonic	**<dest>,<src>**	**Notes**	**Binary Mode**		**Source Mode**	
			Bytes	**States**	**Bytes**	**States**
MOV	DRk,dir8	Dir addr to dword reg	4	6	3	5
	DRk,dir16	Dir addr (64K) to dword reg	5	6	4	5
	Rm,dir8	Dir addr to byte reg	4	3 (3)	3	2 (3)
	WRj,dir8	Dir addr to word reg	4	4	3	3
	Rm,dir16	Dir addr (64K) to byte reg	5	3	4	2
	WRj,dir16	Dir addr (64K) to word reg	5	4	4	3
	Rm,@WRj	Indir addr (64K) to byte reg	4	2	3	2
	Rm,@DRk	Indir addr (16M) to byte reg	4	4	3	3
	WRjd,@WRjs	Indir addr(64K) to word reg	4	4	3	3
	WRj,@DRk	Indir addr(16M) to word reg	4	5	3	4
	dir8,Rm	Byte reg to dir addr	4	4 (3)	3	3 (3)
	dir8,WRj	Word reg to dir addr	4	5	3	4
	dir16,Rm	Byte reg to dir addr (64K)	5	4	4	3
	dir16,WRj	Word reg to dir addr (64K)	5	5	4	4
	@WRj,Rm	Byte reg to indir addr (64K)	4	4	3	3
	@DRk,Rm	Byte reg to indir addr (16M)	4	5	3	4
	@WRjd,WRjs	Word reg to indir addr (64K)	4	5	3	4
	@DRk,WRj	Word reg to indir addr (16M)	4	6	3	5
	dir8,DRk	Dword reg to dir addr	4	7	3	6
	dir16,DRk	Dword reg to dir addr (64K)	5	7	4	6
	Rm,@WRj+dis16	Indir addr with disp (64K) to byte reg	5	6	4	5
	WRj,@WRj+dis16	Indir addr with disp (64K) to word reg	5	7	4	6
	Rm,@DRk+dis24	Indir addr with disp (16M) to byte reg	5	7	4	6
	WRj,@DRk+dis24	Indir addr with disp (16M) to word reg	5	8	4	7
	@WRj+dis16,Rm	Byte reg to Indir addr with disp (64K)	5	6	4	5

NOTES:

1. A shaded cell denotes an instruction in the MCS® 51 architecture.
2. Instructions that move bits are in Table A-26 on page A-23.
3. If this instruction addresses an I/O port (P*x*, *x* = 0–3), add 1 to the number of states.
4. External memory addressed by instructions in the MCS 51 architecture is in the region specified by DPXL (reset value = 01H). See “Compatibility with the MCS® 51 Architecture” on page 3-2.

Table A-24. Summary of Move Instructions (Continued)

Move (2)	**MOV <dest>,<src>**	**destination ← src opnd**
Move with Sign Extension	**MOVS <dest>,<src>**	**destination ← src opnd with sign extend**
Move with Zero Extension	**MOVZ <dest>,<src>**	**destination ← src opnd with zero extend**
Move Code Byte	**MOVC <dest>,<src>**	**A ← code byte**
Move to External Mem	**MOVX <dest>,<src>**	**external mem ← (A)**
Move from External Mem	**MOVX <dest>,<src>**	**A ← source opnd in external mem**

Mnemonic	**<dest>,<src>**	**Notes**	**Binary Mode**		**Source Mode**	
			Bytes	**States**	**Bytes**	**States**
MOV	@WRj+dis16,WRj	Word reg to Indir addr with disp (64K)	5	7	4	6
	@DRk+dis24,Rm	Byte reg to Indir addr with disp (16M)	5	7	4	6
	@DRk+dis24,WRj	Word reg to Indir addr with disp (16M)	5	8	4	7
MOVH	DRk(hi), #data16	16-bit immediate data into upper word of dword reg	5	3	4	2
MOVS	WRj,Rm	Byte reg to word reg with sign extension	3	2	2	1
MOVZ	WRj,Rm	Byte reg to word reg with zeros extension	3	2	2	1
MOVC	A,@A+DPTR	Code byte relative to DPTR to acc	1	6	1	6
	A,@A+PC	Code byte relative to PC to acc	1	6	1	6
MOVX	A,@Ri	External mem (8-bit addr) to acc (4)	1	4	2	5
	A,@DPTR	External mem (16-bit addr) to acc (4)	1	5	1	5
	@Ri,A	Acc to external mem (8-bit addr) (4)	1	4	1	4
	@DPTR,A	Acc to external mem (16-bit addr) (4)	1	5	1	5

NOTES:

1. A shaded cell denotes an instruction in the MCS® 51 architecture.
2. Instructions that move bits are in Table A-26 on page A-23.
3. If this instruction addresses an I/O port (P*x*, *x* = 0–3), add 1 to the number of states.
4. External memory addressed by instructions in the MCS 51 architecture is in the region specified by DPXL (reset value = 01H). See “Compatibility with the MCS® 51 Architecture” on page 3-2.

Table A-25. Summary of Exchange, Push, and Pop Instructions

Exchange Contents	**XCH <dest>,<src>**	**A ↔ src opnd**
Exchange Digit	**XCHD <dest>,<src>**	**A3:0 ↔ on-chip RAM bits 3:0**
Push	**PUSH <src>**	**SP ← SP + 1; (SP) ← src**
Pop	**POP <dest>**	**dest ← (SP); SP ← SP – 1**

Mnemonic	**<dest>,<src>**	**Notes**	**Binary Mode**		**Source Mode**	
			Bytes	**States**	**Bytes**	**States**
XCH	A,Rn	Acc and reg	2	3	2	3
	A,dir8	Acc and dir addr	2	3 (2)	2	3 (2)
	A,@Ri	Acc and on-chip RAM (8-bit addr)	1	4	1	4
XCHD	A,@Ri	Acc and low nibble in on-chip RAM (8-bit addr)	1	4	1	4
PUSH	dir8	Push dir byte onto stack	2	2	2	2
	#data	Push immediate data onto stack	4	4	3	3
	#data16	Push 16-bit immediate data onto stack	5	5	4	5
	Rm	Push byte reg onto stack	3	4	2	3
	WRj	Push word reg onto stack	3	6	2	5
	DRk	Push double word reg onto stack	3	10	2	9
POP	Dir	Pop dir byte from stack	2	3/3	2	3/3
	Rm	Pop byte reg from stack	3	3	2	2
	WRj	Pop word reg from stack	3	5	2	4
	DRk	Pop double word reg from stack	3	9	2	8

NOTES:
1. A shaded cell denotes an instruction in the MCS® 51 architecture.
2. If this instruction addresses an I/O port (P*x*, *x* = 0–3), add 2 to the number of states.

Table A-26. Summary of Bit Instructions

Clear Bit	**CLR bit**	**bit ← 0**
Set Bit	**SETB bit**	**bit ← 1**
Complement Bit	**CPL bit**	**bit← Øbit**
AND Carry with Bit	**ANL CY,bit**	**CY ← CY Λ bit**
AND Carry with Complement of Bit	**ANL CY,/bit**	**CY ← CY Λ Øbit**
OR Carry with Bit	**ORL CY,bit**	**CY ← CY V bit**
ORL Carry with Complement of Bit	**ORL CY,/bit**	**CY ← CY V Øbit**
Move Bit to Carry	**MOV CY,bit**	**CY ← bit**
Move Bit from Carry	**MOV bit,CY**	**bit ← CY**

Mnemonic	<src>,<dest>	Notes	Binary Mode		Source Mode	
			Bytes	States	Bytes	States
CLR	CY	Clear carry	1	1	1	1
	bit51	Clear dir bit	2	2 (2)	2	2 (2)
	bit	Clear dir bit	4	4	3	3
SETB	CY	Set carry	1	1	1	1
	bit51	Set dir bit	2	2 (2)	2	2 (2)
	bit	Set dir bit	4	4 (2)	3	3 (2)
CPL	CY	Complement carry	1	1	1	1
	bit51	Complement dir bit	2	2 (2)	2	2 (2)
	bit	Complement dir bit	4	4 (2)	3	3 (2)
ANL	CY,bit51	AND dir bit to carry	2	1 (3)	2	1 (3)
	CY,bit	AND dir bit to carry	4	3 (3)	3	2 (3)
ANL/	CY,/bit51	AND complemented dir bit to carry	2	1 (3)	2	1 (3)
	CY,/bit	AND complemented dir bit to carry	4	3 (3)	3	2 (3)
ORL	CY,bit51	OR dir bit to carry	2	1 (3)	2	1 (3)
	CY,bit	OR dir bit to carry	4	3 (3)	3	2 (3)
ORL/	CY,/bit51	OR complemented dir bit to carry	2	1 (3)	2	1 (3)
	CY,/bit	OR complemented dir bit to carry	4	3 (3)	3	2 (3)
MOV	CY,bit51	Move dir bit to carry	2	1 (3)	2	1 (3)
	CY,bit	Move dir bit to carry	4	3 (3)	3	2 (3)
	bit51,CY	Move carry to dir bit	2	2 (2)	2	2 (2)
	bit,CY	Move carry to dir bit	4	4 (2)	3	3 (2)

NOTES:

1. A shaded cell denotes an instruction in the MCS® 51 architecture.
2. If this instruction addresses an I/O port (P*x*, *x* = 0–3), add 2 to the number of states.
3. If this instruction addresses an I/O port (P*x*, *x* = 0–3), add 1 to the number of states.

Table A-27. Summary of Control Instructions

Mnemonic	<dest>,<src>	Notes	Binary Mode		Source Mode	
			Bytes	States (2)	Bytes	States (2)
ACALL	addr11	Absolute subroutine call	2	9	2	9
ECALL	@DRk	Extended subroutine call, indirect	3	12	2	11
	addr24	Extended subroutine call	5	14	4	13
LCALL	@WRj	Long subroutine call, indirect	3	9	2	8
	addr16	Long subroutine call	3	9	3	9
RET		Return from subroutine	1	6	1	6
ERET		Extended subroutine return	3	10	2	9
RETI		Return from interrupt	1	6	1	6
AJMP	addr11	Absolute jump	2	3	2	3
EJMP	addr24	Extended jump	5	6	4	5
	@DRk	Extended jump, indirect	3	7	2	6
LJMP	@WRj	Long jump, indirect	3	6	2	5
	addr16	Long jump	3	4	3	4
SJMP	rel	Short jump (relative addr)	2	3	2	3
JMP	@A+DPTR	Jump indir relative to the DPTR	1	5	1	5
JC	rel	Jump if carry is set	2	1/4	2	1/4
JNC	rel	Jump if carry not set	2	1/4	2	1/4
JB	bit51,rel	Jump if dir bit is set	3	2/5	3	2/5
	bit,rel	Jump if dir bit of 8-bit addr location is set	5	4/7	4	3/6
JNB	bit51,rel	Jump if dir bit is not set	3	2/5	3	2/5
	bit,rel	Jump if dir bit of 8-bit addr location is not set	5	4/7	4	3/6
JBC	bit51,rel	Jump if dir bit is set & clear bit	3	4/7	3	4/7
	bit,rel	Jump if dir bit of 8-bit addr location is set and clear bit	5	7/10	4	6/9
JZ	rel	Jump if acc is zero	2	2/5	2	2/5
JNZ	rel	Jump if acc is not zero	2	2/5	2	2/5
JE	rel	Jump if equal	3	2/5	2	1/4
JNE	rel	Jump if not equal	3	2/5	2	1/4
JG	rel	Jump if greater than	3	2/5	2	1/4
JLE	rel	Jump if less than or equal	3	2/5	2	1/4

NOTES:

1. A shaded cell denotes an instruction in the MCS® 51 architecture.
2. For conditional jumps, times are given as not-taken/taken.

Table A-27. Summary of Control Instructions (Continued)

Mnemonic	<dest>,<src>	Notes	Binary Mode		Source Mode	
			Bytes	States (2)	Bytes	States (2)
JSL	rel	Jump if less than (signed)	3	2/5	2	1/4
JSLE	rel	Jump if less than or equal (signed)	3	2/5	2	1/4
JSG	rel	Jump if greater than (signed)	3	2/5	2	1/4
JSGE	rel	Jump if greater than or equal (signed)	3	2/5	2	1/4
CJNE	A,dir8,rel	Compare dir byte to acc and jump if not equal	3	2/5	3	2/5
	A,#data,rel	Compare immediate to acc and jump if not equal	3	2/5	3	2/5
	Rn,#data,rel	Compare immediate to reg and jump if not equal	3	2/5	4	3/6
	@Ri,#data,rel	Compare immediate to indir and jump if not equal	3	3/6	4	4/7
DJNZ	Rn,rel	Decrement reg and jump if not zero	3	2/5	3	3/6
	dir8,rel	Decrement dir byte and jump if not zero	3	3/6	3	3/6
TRAP	—	Jump to the trap interrupt vector	2	10	1	9
NOP	—	No operation	1	1	1	1

NOTES:

1. A shaded cell denotes an instruction in the MCS® 51 architecture.
2. For conditional jumps, times are given as not-taken/taken.

A.4 INSTRUCTION DESCRIPTIONS

This section describes each instruction in the MCS 251 architecture. See the note on page A-11 regarding execution times.

Table A-28 defines the symbols (—, ✓, 1, 0, ?) used to indicate the effect of the instruction on the flags in the PSW and PSW1 registers. For a conditional jump instruction, "!" indicates that a flag influences the decision to jump.

Table A-28. Flag Symbols

Symbol	Description
—	The instruction does not modify the flag.
✓	The instruction sets or clears the flag, as appropriate.
1	The instruction sets the flag.
0	The instruction clears the flag.
?	The instruction leaves the flag in an indeterminate state.
!	For a conditional jump instruction: The state of the flag before the instruction executes influences the decision to jump or not jump.

ACALL <addr11>

Function: Absolute call

Description: Unconditionally calls a subroutine at the specified address. The instruction increments the 3-byte PC twice to obtain the address of the following instruction, then pushes bytes 0 and 1 of the result onto the stack (byte 0 first) and increments the stack pointer twice. The destination address is obtained by successively concatenating bits 15–11 of the incremented PC, opcode bits 7–5, and the second byte of the instruction. The subroutine called must therefore start within the same 2-Kbyte "page" of the program memory as the first byte of the instruction following ACALL.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: The stack pointer (SP) contains 07H and the label "SUBRTN" is at program memory location 0345H. After executing the instruction

ACALL SUBRTN

at location 0123H, SP contains 09H; on-chip RAM locations 08H and 09H contain 01H and 25H, respectively; and the PC contains 0345H.

	Binary Mode	Source Mode
Bytes:	2	2
States:	9	9

[Encoding]

a10 a9 a8 1	0 0 0 1	a7 a6 a5 a4	a3 a2 a1 a0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ACALL
(PC) ← (PC) + 2
(SP) ← (SP) + 1
((SP)) ← (PC.7:0)
(SP) ← (SP) + 1
((SP)) ← (PC.15:8)
(PC.10:0) ← page address

ADD <dest>,<src>

Function: Add

Description: Adds the source operand to the destination operand, which can be a register or the accumulator, leaving the result in the register or accumulator. If there is a carry out of bit 7 (CY), the CY flag is set. If byte variables are added, and if there is a carry out of bit 3 (AC), the AC flag is set. For addition of unsigned integers, the CY flag indicates that an overflow occurred.

If there is a carry out of bit 6 but not out of bit 7, or a carry out of bit 7 but not bit 6, the OV flag is set. When adding signed integers, the OV flag indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, and immediate.

Flags:

CY	AC	OV	N	Z
✓	✓	✓	✓	✓

Example: Register 1 contains 0C3H (11000011B) and register 0 contains 0AAH (10101010B). After executing the instruction

ADD R1,R0

register 1 contains 6DH (01101101B), the AC flag is clear, and the CY and OV flags are set.

Variations

ADD A,#data

	Binary Mode	Source Mode
Bytes:	2	2
States:	1	1

[Encoding]

0 0 1 0	0 1 0 0	immed. data

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ADD
(A) ← (A) + #data

ADD A,dir8

	Binary Mode	Source Mode
Bytes:	2	2
States:	1†	1†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

0 0 1 0	0 1 0 1	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ADD
(A) ← (A) + (dir8)

ADD A,@Ri

	Binary Mode	Source Mode
Bytes:	1	2
States:	2	3

[Encoding]

0 0 1 0	0 1 1 i

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: ADD
(A) ← (A) + ((Ri))

ADD A,Rn

	Binary Mode	Source Mode
Bytes:	1	2
States:	1	2

[Encoding]

0 0 1 0	1 r r r

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: ADD
(A) ← (A) + (Rn)

ADD Rmd,Rms

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]

0 0 1 0	1 1 0 0	s s s s	S S S S

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ADD
(Rmd) ← (Rmd) + (Rms)

ADD WRjd,WRjs

	Binary Mode	Source Mode
Bytes:	3	2
States:	3	2

[Encoding]

0 0 1 0	1 1 0 1	t t t t	T T T T

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ADD
(WRjd) ← (WRjd) + (WRjs)

ADD DRkd,DRks

	Binary Mode	Source Mode
Bytes:	3	2
States:	5	4

[Encoding]

0 0 1 0	1 1 1 1	u u u u	U U U U

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ADD
(DRkd) ← (DRkd) + (DRks)

ADD Rm,#data

	Binary Mode	Source Mode
Bytes:	4	3
States:	3	2

[Encoding]

0 0 1 0	1 1 1 0	s s s s	0 0 0 0	#data

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ADD
(Rm) ← (Rm) + #data

ADD WRj,#data16

	Binary Mode	**Source Mode**
Bytes:	5	4
States:	4	3

[Encoding]

0 0 1 0	1 1 1 0	t t t t	0 1 0 0	#data hi	#data low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ADD
(WRj) ← (WRj) + #data16

ADD DRk,#0data16

	Binary Mode	**Source Mode**
Bytes:	5	4
States:	6	5

[Encoding]

0 0 1 0	1 1 1 0	u u u u	1 0 0 0	#data hi	#data low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ADD
(DRk) ← (DRk) + #data16

ADD Rm,dir8

	Binary Mode	**Source Mode**
Bytes:	4	3
States:	3†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

0 0 1 0	1 1 1 0	s s s s	0 0 0 1	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ADD
(Rm) ← (Rm) + (dir8)

ADD WRj,dir8

	Binary Mode	Source Mode
Bytes:	4	3
States:	4	3

[Encoding]

0010	1110	tttt	0101	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding

Operation: ADD
(WRj) ← (WRj) + (dir8)

ADD Rm,dir16

	Binary Mode	Source Mode
Bytes:	5	4
States:	3	2

[Encoding]

0010	1110	ssss	0011	direct addr	direct add

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ADD
(Rm) ← (Rm) + (dir16)

ADD WRj,dir16

	Binary Mode	Source Mode
Bytes:	5	4
States:	4	3

[Encoding]

0010	1110	tttt	0111	direct addr	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ADD
(WRj) ← (WRj) + (dir16)

ADD Rm,@WRj

	Binary Mode	Source Mode
Bytes:	4	3
States:	3	2

[Encoding]

0010	1110	tttt	1001	ssss	0000

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ADD
(Rm) ← (Rm) + ((WRj))

ADD Rm,@DRk

	Binary Mode	Source Mode
Bytes:	4	3
States:	4	3

[Encoding]

0010	1110	uuuu	1011	ssss	0000

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ADD
(Rm) ← (Rm) + ((DRk))

ADDC A,<src>

Function: Add with carry

Description: Simultaneously adds the specified byte variable, the CY flag, and the accumulator contents, leaving the result in the accumulator. If there is a carry out of bit 7 (CY), the CY flag is set; if there is a carry out of bit 3 (AC), the AC flag is set. When adding unsigned integers, the CY flag indicates that an overflow occurred.

If there is a carry out of bit 6 but not out of bit 7, or a carry out of bit 7 but not bit 6, the OV flag is set. When adding signed integers, the OV flag indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, and immediate.

Flags:

CY	AC	OV	N	Z
✓	✓	✓	✓	✓

Example: The accumulator contains 0C3H (11000011B), register 0 contains 0AAH (10101010B), and the CY flag is set. After executing the instruction

ADDC A,R0

the accumulator contains 6EH (01101110B), the AC flag is clear, and the CY and OV flags are set.

Variations

ADDC A,#data

	Binary Mode	Source Mode
Bytes:	2	2
States:	1	1

[Encoding]	0 0 1 1	0 1 0 0	immed. data

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ADDC
(A) ← (A) + (CY) + #data

ADDC A,dir8

	Binary Mode	Source Mode
Bytes:	2	2
States:	1†	1†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]	0 0 1 1	0 1 0 1	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ADDC
(A) ← (A) + (CY) + (dir8)

ADDC A,@Ri

	Binary Mode	Source Mode
Bytes:	1	2
States:	2	3

[Encoding]	0 0 1 1	0 1 1 i

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: ADDC
(A) ← (A) + (CY) + ((Ri))

ADDC A,Rn

	Binary Mode	Source Mode
Bytes:	1	2
States:	1	2

[Encoding]

0 0 1 1	1 r r r

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: ADDC
(A) ← (A) + (CY) + (Rn)

AJMP addr11

Function: Absolute jump

Description: Transfers program execution to the specified address, which is formed at run time by concatenating the upper five bits of the PC (after incrementing the PC twice), opcode bits 7–5, and the second byte of the instruction. The destination must therefore be within the same 2-Kbyte "page" of program memory as the first byte of the instruction following AJMP.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: The label "JMPADR" is at program memory location 0123H. After executing the instruction

AJMP JMPADR

at location 0345H, the PC contains 0123H.

	Binary Mode	Source Mode
Bytes:	2	2
States:	3	3

[Encoding]

a10 a9 a8 0	0 0 0 1	a7 a6 a5 a4	a3 a2 a1 a0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: AJMP
(PC) ← (PC) + 2
(PC.10:0) ← page address

ANL <dest>,<src>

Function: Logical-AND

Description: Performs the bitwise logical-AND (Λ) operation between the specified variables and stores the results in the destination variable.

The two operands allow 10 addressing mode combinations. When the destination is the register or accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data is read from the output data latch, not the input pins.

Flags:

CY	AC	OV	N	Z
—	—	—	✓	✓

Example: Register 1 contains 0C3H (11000011B) and register 0 contains 55H (01010101B). After executing the instruction

ANL R1,R0

register 1 contains 41H (01000001B).

When the destination is a directly addressed byte, this instruction clears combinations of bits in any RAM location or hardware register. The mask byte determining the pattern of bits to be cleared would either be an immediate constant contained in the instruction or a value computed in the register or accumulator at run time. The instruction

ANL P1,#01110011B

clears bits 7, 3, and 2 of output port 1.

Variations

ANL dir8,A

	Binary Mode	Source Mode
Bytes:	2	2
States:	2†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 2 states.

[Encoding]			
	0 1 0 1	0 0 1 0	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ANL
(dir8) ← (dir8) Λ (A)

ANL dir8,#data

	Binary Mode	Source Mode
Bytes:	3	3
States:	3†	3†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

0 1 0 1	0 0 1 1	direct addr	immed. data

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ANL
(dir8) ← (dir8) Λ #data

ANL A,#data

	Binary Mode	Source Mode
Bytes:	2	2
States:	1	1

[Encoding]

0 1 0 1	0 1 0 0	immed. data

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ANL
(A) ← (A) Λ #data

ANL A,dir8

	Binary Mode	Source Mode
Bytes:	2	2
States:	1†	1†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

0 1 0 1	0 1 0 1	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ANL
(A) ← (A) Λ (dir8)

ANL A,@Ri

	Binary Mode	Source Mode
Bytes:	1	2
States:	2	3

[Encoding]

0 1 0 1	0 1 1 i

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: ANL
(A) ← (A) Λ ((Ri))

ANL A,Rn

	Binary Mode	Source Mode
Bytes:	1	2
States:	1	2

[Encoding]

0 1 0 1	1 r r r

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: ANL
(A) ← (A) Λ (Rn)

ANL Rmd,Rms

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]

0 1 0 1	1 1 0 0	s s s s	S S S S

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ANL
(Rmd) ← (Rmd) Λ (Rms)

ANL WRjd,WRjs

	Binary Mode	Source Mode
Bytes:	3	2
States:	3	2

[Encoding]

0 1 0 1	1 1 0 1	t t t t	T T T T

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ANL
(WRjd) ← (WRjd) Λ (WRjs)

ANL Rm,#data

	Binary Mode	Source Mode
Bytes:	4	3
States:	3	2

[Encoding]

0 1 0 1	1 1 1 0	s s s s	0000	#data

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ANL
(Rm) ← (Rm) Λ #data

ANL WRj,#data16

	Binary Mode	Source Mode
Bytes:	5	4
States:	4	3

[Encoding]

0 1 0 1	1 1 1 0	t t t t	0 1 0 0	#data hi	#data low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ANL
(WRj) ← (WRj) Λ #data16

ANL Rm,dir8

	Binary Mode	Source Mode
Bytes:	4	3
States:	3†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

0 1 0 1	1 1 1 0	s s s s	0 0 0 1	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ANL
(Rm) ← (Rm) Λ (dir8)

ANL WRj,dir8

	Binary Mode	Source Mode
Bytes:	4	3
States:	4	3

[Encoding]

0 1 0 1	1 1 10	t t t t	0 1 0 1	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ANL
(WRj) ← (WRj) Λ (dir8)

ANL Rm,dir16

	Binary Mode	Source Mode
Bytes:	5	4
States:	3	2

[Encoding]

0 1 0 1	1 1 1 0	s s s s	0 0 1 1	direct	direct

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ANL
(Rm) ← (Rm) Λ (dir16)

ANL WRj,dir16

	Binary Mode	Source Mode
Bytes:	5	4
States:	4	3

[Encoding]

0 1 0 1	1 1 1 0	t t t t	0 1 1 1	direct	direct

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ANL
(WRj) ← (WRj) Λ (dir16)

ANL Rm,@WRj

	Binary Mode	Source Mode
Bytes:	4	3
States:	3	2

[Encoding]

0101	1110	tttt	1001	ssss	0000

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ANL
(Rm) ← (Rm) Λ ((WRj))

ANL Rm,@DRk

	Binary Mode	Source Mode
Bytes:	4	3
States:	4	3

[Encoding]

0101	1110	uuuu	1011	ssss	0000

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ANL
(Rm) ← (Rm) Λ ((DRk))

ANL CY,<src–bit>

Function: Logical-AND for bit variables

Description: If the Boolean value of the source bit is a logical 0, clear the CY flag; otherwise leave the CY flag in its current state. A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is not affected.

Only direct addressing is allowed for the source operand.

Flags:

CY	AC	OV	N	Z
✓	—	—	—	—

Example: Set the CY flag if, and only if, P1.0 = 1, ACC. 7 = 1, and OV = 0:

```
MOV CY,P1.0    ;Load carry with input pin state
ANL CY,ACC.7  ;AND carry with accumulator bit 7
ANL CY,/OV       ;AND with inverse of overflow flag
```

ANL CY,bit51

	Binary Mode	Source Mode
Bytes:	2	2
States:	1†	1†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

1 0 0 0	0 0 1 0	bit addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ANL
(CY) ← (CY) Λ (bit51)

ANL CY,/bit51

	Binary Mode	Source Mode
Bytes:	2	2
States:	1†	1†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

1 0 1 1	0 0 0 0	bit addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ANL
(CY) ← (CY) Λ Ø (bit51)

ANL CY,bit

	Binary Mode	Source Mode
Bytes:	4	3
States:	3†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

1 0 1 0	1 0 0 1	1 0 0 0	0	y y y	dir addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ANL
(CY) ← (CY) Λ (bit)

ANL CY,/bit

	Binary Mode	Source Mode
Bytes:	4	3
States:	3†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

1010	1001	1111	0	y y y	dir addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ANL
(CY) ← (CY) Λ Ø (bit)

CJNE <dest>,<src>,rel

Function: Compare and jump if not equal.

Description: Compares the magnitudes of the first two operands and branches if their values are not equal. The branch destination is computed by adding the signed relative displacement in the last instruction byte to the PC, after incrementing the PC to the start of the next instruction. If the unsigned integer value of <dest-byte> is less than the unsigned integer value of <src-byte>, the CY flag is set. Neither operand is affected.

The first two operands allow four addressing mode combinations: the accumulator may be compared with any directly addressed byte or immediate data, and any indirect RAM location or working register can be compared with an immediate constant.

Flags:

CY	AC	OV	N	Z
✓	—	—	✓	✓

Example: The accumulator contains 34H and R7 contains 56H. After executing the first instruction in the sequence

```
                CJNE        R7,#60H,NOT_EQ
;               . . .       . . .               ;R7 = 60H
NOT_EQ:         JC          REQ_LOW             ; IF R7 < 60H
;               . . .       . . .               ;R7 > 60H
```

the CY flag is set and program execution continues at label NOT_EQ. By testing the CY flag, this instruction determines whether R7 is greater or less than 60H.

If the data being presented to Port 1 is also 34H, then executing the instruction,

```
WAIT: CJNE A,P1,WAIT
```

clears the CY flag and continues with the next instruction in the sequence, since the accumulator does equal the data read from P1. (If some other value was being input on P1, the program loops at this point until the P1 data changes to 34H.)

Variations

CJNE A,#data,rel

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	3	3
States:	2	5	2	5

[Encoding]

1 0 1 1	0 1 0 0	immed. data	rel. addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation:

```
(PC) ← (PC) + 3
IF (A) ≠ #data
THEN
        (PC) ← (PC) + relative offset
IF (A) < #data
THEN
        (CY) ← 1
ELSE
        (CY) ← 0
```

CJNE A,dir8,rel

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	3	3
States:	3	6	3	6

[Encoding]

1 0 1 1	0 1 0 1	direct addr	rel. addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation:

```
(PC) ← (PC) + 3
IF (A) ≠ dir8
THEN
        (PC) ← (PC) + relative offset
IF (A) < dir8
THEN
        (CY) ← 1
ELSE
        (CY) ← 0
```

CJNE @Ri,#data,rel

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	4	4
States:	3	6	4	7

[Encoding]	1 0 1 1	0 1 1 i	immed. data	rel. addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation:

```
(PC) ← (PC) + 3
IF ((Ri)) ≠ #data
THEN
        (PC) ← (PC) + relative offset
IF ((Ri)) < #data
THEN
        (CY) ← 1
ELSE
        (CY) ← 0
```

CJNE Rn,#data,rel

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	4	4
States:	2	5	3	6

[Encoding]	1 01 1	1 r r r	immed. data	rel. addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation:

```
(PC) ← (PC) + 3
IF (Rn) ≠ #data
THEN
        (PC) ← (PC) + relative offset
IF (Rn) < #data
THEN
        (CY) ← 1
ELSE
        (CY) ← 0
```

CLR A

Function: Clear accumulator

Description: Clears the accumulator (i.e., resets all bits to zero).

Flags:

CY	AC	OV	N	Z
—	—	—	✓	✓

Example: The accumulator contains 5CH (01011100B). The instruction

CLR A

clears the accumulator to 00H (00000000B).

	Binary Mode	**Source Mode**
Bytes:	1	1
States:	1	1

[Encoding]

1 1 1 0	0 1 0 0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: CLR
(A) ← 0

CLR bit

Function: Clear bit

Description: Clears the specified bit. CLR can operate on the CY flag or any directly addressable bit.

Flags: Only for instructions with CY as the operand.

CY	AC	OV	N	Z
✓	—	—	—	—

Example: Port 1 contains 5DH (01011101B). After executing the instruction

CLR P1.2

port 1 contains 59H (01011001B).

Variations

CLR bit51

	Binary Mode	Source Mode
Bytes:	4	3
States:	2†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 2 states.

[Encoding]

1 1 0 0	0 0 1 0	Bit addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: CLR
(bit51) ← 0

CLR CY

	Binary Mode	Source Mode
Bytes:	1	1
States:	1	1

[Encoding]

1 1 0 0	0 0 1 1

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: CLR
(CY) ← 0

CLR bit

	Binary Mode	Source Mode
Bytes:	4	4
States:	4†	3†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 2 states.

[Encoding]

1 0 1 0	1 0 0 1	1 1 0 0	0	y y y	dir addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: CLR
(bit) ← 0

CMP <dest>,<src>

Function: Compare

Description: Subtracts the source operand from the destination operand. The result is not stored in the destination operand. If a borrow is needed for bit 7, the CY (borrow) flag is set; otherwise it is clear.

When subtracting signed integers, the OV flag indicates a negative result when a negative value is subtracted from a positive value, or a positive result when a positive value is subtracted from a negative value.

The source operand allows four addressing modes: register, direct, immediate and indirect.

Flags:

CY	AC	OV	N	Z
✓	✓	✓	✓	✓

Example: Register 1 contains 0C9H (11001001B) and register 0 contains 54H (01010100B). The instruction

CMP R1,R0

clears the CY and AC flags and sets the OV flag.

Variations

CMP Rmd,Rms

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]

1 0 1 1	1 1 0 0	s s s s	S S S S

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: CMP
(Rmd) – (Rms)

CMP WRjd,WRjs

	Binary Mode	Source Mode
Bytes:	3	2
States:	3	2

[Encoding]

1 0 1 1	1 1 1 0	t t t t	T T T T

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: CMP
(WRjd) – (WRjs)

CMP DRkd,DRks

	Binary Mode	Source Mode
Bytes:	3	2
States:	5	4

[Encoding]

1011	1111	uuuu	UUUU

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: CMP
(DRkd) – (DRks)

CMP Rm,#data

	Binary Mode	Source Mode
Bytes:	4	3
States:	3	2

[Encoding]

1011	1110	ssss	0000	# data

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: CMP
(Rm) – #data

CMP WRj,#data16

	Binary Mode	Source Mode
Bytes:	5	4
States:	4	3

[Encoding]

1011	1110	tttt	0100	#data hi	#data low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: CMP
(WRj) – #data16

CMP DRk,#0data16

	Binary Mode	Source Mode
Bytes:	5	4
States:	6	5

[Encoding]

1011	1110	uuuu	1000	#data hi	#data hi

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: CMP
(DRk) – #0data16

CMP DRk,#1data16

	Binary Mode	Source Mode
Bytes:	5	4
States:	6	5

[Encoding]

1 0 1 1	1 1 1 0	u u u u	1 1 0 0	#data hi	#data hi

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: CMP
(DRk) – #1data16

CMP Rm,dir8

	Binary Mode	Source Mode
Bytes:	4	3
States:	3†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

1 0 1 1	1 1 1 0	s s s s	0001	dir addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: CMP
(Rm) – (dir8)

CMP WRj,dir8

	Binary Mode	Source Mode
Bytes:	4	3
States:	4	3

[Encoding]

1 0 1 1	1 1 10	t t t t	0 1 0 1	dir addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: CMP
(WRj) – (dir8)

CMP Rm,dir16

	Binary Mode	Source Mode
Bytes:	5	4
States:	3	2

[Encoding]

1 0 1 1	1 1 1 0	s s s s	0 0 1 1	dir addr	dir addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: CMP
(Rm) – (dir16)

CMP WRj,dir16

	Binary Mode	Source Mode
Bytes:	5	4
States:	4	3

[Encoding]

1 0 1 1	1 1 1 0	t t t t	0 1 1 1	dir addr	dir addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: CMP
(WRj) – (dir16)

CMP Rm,@WRj

	Binary Mode	Source Mode
Bytes:	4	3
States:	3	2

[Encoding]

1 0 1 1	1 1 1 0	t t t t	1 0 0 1	s s s s	0 0 0 0

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: CMP
(Rm) – ((WRj))

CMP Rm,@DRk

	Binary Mode	Source Mode
Bytes:	4	3
States:	4	3

[Encoding]

1011	1110	uuuu	1011	ssss	0000

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: CMP
(Rm) – ((DRk))

CPL A

Function: Complement accumulator

Description: Logically complements (Ø) each bit of the accumulator (one's complement). Clear bits are set and set bits are cleared.

Flags:

CY	AC	OV	N	Z
—	—	—	✓	✓

Example: The accumulator contains 5CH (01011100B). After executing the instruction

CPL A

the accumulator contains 0A3H (10100011B).

	Binary Mode	Source Mode
Bytes:	1	1
States:	1	1

[Encoding]

1111	0100

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: CPL
(A) ← Ø(A)

CPL bit

Function: Complement bit

Description: Complements (Ø) the specified bit variable. A clear bit is set, and a set bit is cleared. CPL can operate on the CY or any directly addressable bit.

Note: When this instruction is used to modify an output pin, the value used as the original data is read from the output data latch, not the input pin.

Flags: Only for instructions with CY as the operand.

CY	AC	OV	N	Z
✓	—	—	—	—

Example: Port 1 contains 5BH (01011101B). After executing the instruction sequence

CPL P1.1
CPL P1.2

port 1 contains 5BH (01011011B).

Variations

CPL bit51

	Binary Mode	Source Mode
Bytes:	2	2
States:	2†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 2 states.

[Encoding]

1 0 1 1	0 0 1 0		bit addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: CPL
(bit51) ← Ø(bit51)

CPL CY

	Binary Mode	Source Mode
Bytes:	1	1
States:	1	1

[Encoding]

1 0 1 1	0 0 1 1

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: CPL
(CY) ← Ø(CY)

CPL bit

	Binary Mode	Source Mode
Bytes:	4	3
States:	4†	3†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 2 states.

[Encoding]

1 0 1 0	1 0 0 1		1 0 1 1	0	y y y		dir addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: CPL
(bit) ← Ø(bit)

DA A

Function: Decimal-adjust accumulator for addition

Description: Adjusts the 8-bit value in the accumulator that resulted from the earlier addition of two variables (each in packed-BCD format), producing two 4-bit digits. Any ADD or ADDC instruction may have been used to perform the addition.

If accumulator bits 3:0 are greater than nine (XXXX1010–XXXX1111), or if the AC flag is set, six is added to the accumulator, producing the proper BCD digit in the low nibble. This internal addition sets the CY flag if a carry out of the lowest 4 bits propagated through all higher bits, but it does not clear the CY flag otherwise.

If the CY flag is now set, or if the upper four bits now exceed nine (1010XXXX–1111XXXX), these four bits are incremented by six, producing the proper BCD digit in the high nibble. Again, this sets the CY flag if there was a carry out of the upper four bits, but does not clear the carry. The CY flag thus indicates if the sum of the original two BCD variables is greater than 100, allowing multiple-precision decimal addition. The OV flag is not affected.

All of this occurs during one instruction cycle. Essentially, this instruction performs the decimal conversion by adding 00H, 06H, 60H, or 66H to the accumulator, depending on initial accumulator and PSW conditions.

Note: DA A cannot simply convert a hexadecimal number in the accumulator to BCD notation, nor does DA A apply to decimal subtraction.

Flags:

CY	AC	OV	N	Z
✓	—	—	✓	✓

Example: The accumulator contains 56H (01010110B), which represents the packed BCD digits of the decimal number 56. Register 3 contains 67H (01100111B), which represents the packed BCD digits of the decimal number 67. The CY flag is set. After executing the instruction sequence

```
ADDC A,R3
DA A
```

the accumulator contains 0BEH (10111110) and the CY and AC flags are clear.
The Decimal Adjust instruction then alters the accumulator to the value 24H (00100100B), indicating the packed BCD digits of the decimal number 24, the lower two digits of the decimal sum of 56, 67, and the carry-in. The CY flag is set by the Decimal Adjust instruction, indicating that a decimal overflow occurred. The true sum of 56, 67, and 1 is 124.

BCD variables can be incremented or decremented by adding 01H or 99H. If the accumulator contains 30H (representing the digits of 30 decimal), then the instruction sequence,

```
ADD A,#99H
DA A
```

leaves the CY flag set and 29H in the accumulator, since 30 + 99 = 129. The low byte of the sum can be interpreted to mean 30 – 1 = 29.

	Binary Mode	Source Mode
Bytes:	1	1
States:	1	1

[Encoding]

1 1 0 1	0 1 0 0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation:

```
DA
(Contents of accumulator are BCD)
IF      [[(A.3:0) > 9] V [(AC) = 1]]
          THEN (A.3:0) ← (A.3:0) + 6
                 AND
IF      [[(A.7:4) > 9] V [(CY) = 1]]
        THEN (A.7:4) ← (A.7:4) + 6
```

DEC byte

Function: Decrement

Description: Decrements the specified byte variable by 1. An original value of 00H underflows to 0FFH. Four operands addressing modes are allowed: accumulator, register, direct, or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data is read from the output data latch, not the input pins.

Flags:

CY	AC	OV	N	Z
—	—	—	✓	✓

Example: Register 0 contains 7FH (01111111B). On-chip RAM locations 7EH and 7FH contain 00H and 40H, respectively. After executing the instruction sequence

```
DEC @R0
DEC R0
DEC @R0
```

register 0 contains 7EH and on-chip RAM locations 7EH and 7FH are set to 0FFH and 3FH, respectively.

Variations

DEC A

	Binary Mode	Source Mode
Bytes:	1	1
States:	1	1

[Encoding]

0 0 0 1	0 1 0 0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: DEC
(A) ← (A) – 1

DEC dir8

	Binary Mode	Source Mode
Bytes:	2	2
States:	2†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 2 states.

[Encoding]	0 0 0 1	0 1 0 1	dir addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: DEC
(dir8) ← (dir8) – 1

DEC @Ri

	Binary Mode	Source Mode
Bytes:	1	2
States:	3	4

[Encoding]	0 0 0 1	0 1 1 i

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: DEC
((Ri)) ← ((Ri)) – 1

DEC Rn

	Binary Mode	Source Mode
Bytes:	1	2
States:	1	2

[Encoding]	0 0 0 1	1 r r r

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: DEC
(Rn) ← (Rn) – 1

DEC <dest>,<src>

Function: Decrement

Description: Decrements the specified variable at the destination operand by 1, 2, or 4. An original value of 00H underflows to 0FFH.

Flags:

CY	AC	OV	N	Z
—	—	—	✓	✓

Example: Register 0 contains 7FH (01111111B). After executing the instruction sequence

DEC R0,#1

register 0 contains 7EH.

Variations

DEC Rm,#short

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]

0 0 0 1	1 0 1 1	s s s s	0 1	v v

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: DEC
(Rm) ← (Rm) – #short

DEC WRj,#short

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]

0 0 0 1	1 0 1 1	t t t t	0 1	v v

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: DEC
(WRj) ← (WRj) – #short

DEC DRk,#short

	Binary Mode	Source Mode
Bytes:	3	2
States:	5	4

[Encoding]

0 0 0 1	1 0 1 1	u u u u	1 1	v v

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: DEC
(DRk) ← (DRk) – #short

DIV <dest>,<src>

Function: Divide

Description: Divides the unsigned integer in the register by the unsigned integer operand in register addressing mode and clears the CY and OV flags.

For byte operands (<dest>,<src> = Rmd,Rms) the result is 16 bits. The 8-bit quotient is in R(md+1), and the 8-bit remainder is in Rmd. For example: Register 1 contains 251 (0FBH or 11111011B) and register 5 contains 18 (12H or 00010010B). After executing the instruction

DIV R1,R5

register 0 contains 13 (0DH or 00001101B); register 1 contains 17 (11H or 00010001B), since 251 = (13 X 18) + 17; and the CY and OV bits are clear (see Flags).

Flags: The CY flag is cleared. The N flag is set if the MSB of the quotient is set. The Z flag is set if the quotient is zero.**:**

CY	AC	OV	N	Z
0	✓	✓	✓	✓

Exception: if <src> contains 00H, the values returned in both operands are undefined; the CY flag is cleared, OV flag is set, and the rest of the flags are undefined.:

CY	AC	OV	N	Z
0	?	1	?	?

Variations

DIV Rmd Rms

	Binary Mode	Source Mode
Bytes:	3	2
States:	11	10

[Encoding]

1 0 0 0	1 1 0 0	s s s s	S S S S

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: DIV (8-bit operands)
(Rmd) ← quotient (Rmd) / (Rms) if <dest> md = 0,2,4,..,14
(Rmd+1) ← remainder (Rmd) / (Rms)

(Rmd–1) ← quotient (Rmd) / (Rms) if <dest> md = 1,3,5,..,15
(Rmd) ← remainder (Rmd) / (Rms)

DIV WRjd,WRjs

	Binary Mode	Source Mode
Bytes:	3	2
States:	22	21

[Encoding]

1 0 0 0	1 1 0 1	t t t t	T T T T

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: DIV (16-bit operands)
(WRjd) ← quotient (WRjd) / (WRjs) if <dest> jd = 0, 4, 8, ... 28
(WRjd+2) ← remainder (WRjd) / (WRjs)

(WRjd–2) ← quotient (WRjd) / (WRjs) if <dest> jd = 2, 6, 10, ... 30
(WRjd) ← remainder (WRjd) / (WRjs)

For word operands (<dest>,<src> = WRjd,WRjs) The 16-bit quotient is in WR(jd+2), and the 16-bit remainder is in WRjd. For example, for a destination register WR4, assume the quotient is 1122H and the remainder is 3344H. Then, the results are stored in these register file locations:

Location	4	5	6	7
Contents	33H	44H	11H	22H

DIV AB

Function: Divide

Description: Divides the unsigned 8-bit integer in the accumulator by the unsigned 8-bit integer in register B. The accumulator receives the integer part of the quotient; register B receives the integer remainder. The CY and OV flags are cleared.

Exception: if register B contains 00H, the values returned in the accumulator and register B are undefined; the CY flag is cleared and the OV flag is set.

Flags:

CY	AC	OV	N	Z
0	✓	✓	✓	✓

For division by zero**:**

CY	AC	OV	N	Z
0	?	1	?	?

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Example: The accumulator contains 251 (0FBH or 11111011B) and register B contains 18 (12H or 00010010B). After executing the instruction

DIV AB

the accumulator contains 13 (0DH or 00001101B); register B contains 17 (11H or 00010001B), since 251 = (13 X 18) + 17; and the CY and OV flags are clear.

	Binary Mode	Source Mode
Bytes:	1	1
States:	10	10

[Encoding]

1 0 0 0	0 1 0 0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: DIV
(A) ← quotient (A)/(B)
(B) ← remainder (A)/(B)

DJNZ <byte>,<rel–addr>

Function: Decrement and jump if not zero

Description: Decrements the specified location by 1 and branches to the address specified by the second operand if the resulting value is not zero. An original value of 00H underflows to 0FFH. The branch destination is computed by adding the signed relative-displacement value in the last instruction byte to the PC, after incrementing the PC to the first byte of the following instruction.

The location decremented may be a register or directly addressed byte.

Note: When this instruction is used to modify an output port, the value used as the original port data is read from the output data latch, not the input pins.

Flags:

CY	AC	OV	N	Z
—	—	—	✓	✓

Example: The on-chip RAM locations 40H, 50H, and 60H contain 01H, 70H, and 15H, respectively. After executing the instruction sequence

```
DJNZ 40H,LABEL1
DJNZ 50H,LABEL2
DJNZ 60H,LABEL
```

on-chip RAM locations 40H, 50H, and 60H contain 00H, 6FH, and 14H, respectively, and program execution continues at label LABEL2. (The first jump was not taken because the result was zero.)

This instruction provides a simple way of executing a program loop a given number of times, or for adding a moderate time delay (from 2 to 512 machine cycles) with a single instruction.

The instruction sequence,

```
        MOV R2,#8
TOGGLE: CPL P1.7
        DJNZ R2,TOGGLE
```

toggles P1.7 eight times, causing four output pulses to appear at bit 7 of output Port 1. Each pulse lasts three states: two for DJNZ and one to alter the pin.

Variations

DJNZ dir8,rel

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	3	3
States:	3	6	3	6

[Encoding]	1 1 0 1	0 1 0 1	direct addr	rel. addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation:

```
DJNZ
(PC) ← (PC) + 2
(dir8) ← (dir8) – 1
IF (dir8 > 0 or (dir8) < 0
        THEN
            (PC) ← (PC) + rel
```

DJNZ Rn,rel

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	3	3
States:	2	5	3	6

[Encoding]	1 1 0 1	1 r r r	rel. addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation:

```
DJNZ
(PC) ← (PC) + 2
(Rn) ← (Rn) – 1
IF (Rn) > 0 or (Rn) < 0
        THEN
          (PC) ← (PC) + rel
```

ECALL <dest>

Function: Extended call

Description: Calls a subroutine located at the specified address. The instruction adds four to the program counter to generate the address of the next instruction and then pushes the 24-bit result onto the stack (high byte first), incrementing the stack pointer by three. The 8 bits of the high word and the 16 bits of the low word of the PC are then loaded, respectively, with the second, third and fourth bytes of the ECALL instruction. Program execution continues with the instruction at this address. The subroutine may therefore begin anywhere in the full 16-Mbyte memory space.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: The stack pointer contains 07H and the label "SUBRTN" is assigned to program memory location 123456H. After executing the instruction

ECALL SUBRTN

at location 012345H, SP contains 09H; on-chip RAM locations 08H, 09H and 0AH contain 01H, 23H and 45H, respectively; and the PC contains 123456H.

Variations

ECALL addr24

	Binary Mode	Source Mode
Bytes:	5	4
States:	14	13

[Encoding]

1 0 0 1	1 0 1 0	addr23–addr16	addr15–addr8	addr7–addr0

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation:

```
ECALL
(PC) ← (PC) + 4
(SP) ← (SP) + 1
((SP)) ← (PC.23:16)
(SP) ← (SP) + 1
((SP)) ← (PC.15:8)
(SP) ← (SP) + 1
((SP)) ← (PC.7:0)
(PC) ← (addr.23:0)
```

ECALL @DRk

	Binary Mode	Source Mode
Bytes:	3	2
States:	12	11

[Encoding]

1 0 0 1	1 0 0 1	u u u u

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation:

```
ECALL
(PC) ← (PC) + 4
(SP) ← (SP) + 1
((SP)) ← (PC.23:16)
(SP) ← (SP) + 1
((SP)) ← (PC.15:8)
(SP) ← (SP) + 1
((SP)) ← (PC.7:0)
(PC) ← ((DRk))
```

EJMP <dest>

Function: Extended jump

Description: Causes an unconditional branch to the specified address by loading the 8 bits of the high order and 16 bits of the low order words of the PC with the second, third, and fourth instruction bytes. The destination may be therefore be anywhere in the full 16-Mbyte memory space.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: The label "JMPADR" is assigned to the instruction at program memory location 123456H. The instruction is

EJMP JMPADR

Variations

EJMP addr24

	Binary Mode	Source Mode
Bytes:	5	4
States:	6	5

[Encoding]					
	1 0 0 0	1 0 1 0	addr23–addr16	addr15–addr8	addr7–addr0

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: EJMP
(PC) ← (addr.23:0)

EJMP @DRk

	Binary Mode	Source Mode
Bytes:	3	2
States:	7	6

[Encoding]			
	1 0 0 0	1 0 0 1	u u u u

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: EJMP
(PC) ← ((DRk))

ERET

Function: Extended return

Description: Pops byte 2, byte 1, and byte 0 of the 3-byte PC successively from the stack and decrements the stack pointer by 3. Program execution continues at the resulting address, which normally is the instruction immediately following ECALL.

Flags: No flags are affected.

Example: The stack pointer contains 0BH. On-chip RAM locations 08H, 09H and 0AH contain 01H, 23H and 49H, respectively. After executing the instruction

ERET

the stack pointer contains 07H and program execution continues at location 012349H.

	Binary Mode	Source Mode
Bytes:	3	2
States:	10	9

[Encoding]		
	1 0 1 0	1 0 1 0

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ERET
(PC.7:0) ← ((SP))
(SP) ← (SP) – 1
(PC.15:8) ← ((SP))
(SP) ← (SP) – 1
(PC.23:16) ← ((SP))
(SP) ← (SP) – 1

INC <Byte>

Function: Increment

Description: Increments the specified byte variable by 1. An original value of FFH overflows to 00H. Three addressing modes are allowed for 8-bit operands: register, direct, or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data is read from the output data latch, not the input pins.

Flags:

CY	AC	OV	N	Z
—	—	—	✓	✓

Example: Register 0 contains 7EH (011111110B) and on-chip RAM locations 7EH and 7FH contain 0FFH and 40H, respectively. After executing the instruction sequence

INC @R0
INC R0
INC @R0

register 0 contains 7FH and on-chip RAM locations 7EH and 7FH contain 00H and 41H, respectively.

Variations

INC A

	Binary Mode	Source Mode
Bytes:	1	1
States:	1	1

[Encoding]		
	0 0 0 0	0 1 0 0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: INC
(A) ← (A) + 1

INC dir8

	Binary Mode	Source Mode
Bytes:	2	2
States:	2†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 2 states.

[Encoding]			
	0 0 0 0	0 1 0 1	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: INC
(dir8) ← (dir8) + 1

INC @Ri

	Binary Mode	Source Mode
Bytes:	1	2
States:	3	4

[Encoding]		
	0 0 0 0	0 1 1 i

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: INC
((Ri) ← ((Ri)) + 1

INC Rn

	Binary Mode	Source Mode
Bytes:	1	2
States:	1	2

[Encoding]		
	0 0 0 0	1 r r r

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: INC
(Rn) ← (Rn) + 1

INC <dest>,<src>

Function: Increment

Description: Increments the specified variable by 1, 2, or 4. An original value of 0FFH overflows to 00H.

Flags:

CY	AC	OV	N	Z
—	—	—	✓	✓

Example: Register 0 contains 7EH (011111110B). After executing the instruction

INC R0,#1

register 0 contains 7FH.

Variations

INC Rm,#short

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]

0 0 0 0	1 0 1 1	s s s s	00	v v

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: INC
(Rm) ← (Rm) + #short

INC WRj,#short

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]

0 0 0 0	1 0 1 1	t t t t	01	v v

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: INC
(WRj) ← (WRj) + #short

INC DRk,#short

	Binary Mode	Source Mode
Bytes:	3	2
States:	4	3

[Encoding]

0 0 0 0	1 0 1 1	u u u u	11	v v

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: INC
(DRk) ← (DRk) + #shortdata pointer

INC DPTR

Function: Increment data pointer

Description: Increments the 16-bit data pointer by one. A 16-bit increment (modulo 2^{16}) is performed; an overflow of the low byte of the data pointer (DPL) from 0FFH to 00H increments the high byte of the data pointer (DPH) by one. An overflow of the high byte (DPH) does not increment the high word of the extended data pointer (DPX = DR56).

Flags:

CY	AC	OV	N	Z
—	—	—	✓	✓

Example: Registers DPH and DPL contain 12H and 0FEH, respectively. After the instruction sequence

INC DPTR
INC DPTR
INC DPTR

DPH and DPL contain 13H and 01H, respectively.

	Binary Mode	**Source Mode**
Bytes:	1	1
States:	1	1

[Encoding]	1 0 1 0	0 0 1 1

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: INC
(DPTR) ← (DPTR) + 1

JB bit51,rel
JB bit,rel

Function: Jump if bit set

Description: If the specified bit is a one, jump to the address specified; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. The bit tested is not modified.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: Input port 1 contains 11001010B and the accumulator contains 56 (01010110B). After the instruction sequence

JB P1.2,LABEL1
JB ACC.2,LABEL2

program execution continues at label LABEL2.

Variations

JB bit51,rel

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	3	3
States:	2	5	2	5

[Encoding]

0 0 1 0	0 0 0 0	bit addr	rel. addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation:

```
JB
(PC) ← (PC) + 3
IF (bit51) = 1
        THEN
           (PC) ← (PC) + rel
```

JB bit,rel

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	5	5	4	4
States:	4	7	3	6

[Encoding]

1 0 1 0	1 0 0 1	0 0 1 0	0	y y	direct addr	rel. addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation:

```
JB
(PC) ← (PC) + 3
IF (bit) = 1
        THEN
           (PC) ← (PC) + rel
```

JBC bit51,rel
JBC bit,rel

Function: Jump if bit is set and clear bit

Description: If the specified bit is one, branch to the specified address; otherwise proceed with the next instruction. The bit is not cleared if it is already a zero. The branch destination is computed by adding the signed relative displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction.

Note: When this instruction is used to test an output pin, the value used as the original data is read from the output data latch, not the input pin.

Flags:

CY	AC	OV	N	Z
!	—	—	—	—

Example: The accumulator contains 56H (01010110B). After the instruction sequence

JBC ACC.3,LABEL1
JBC ACC.2,LABEL2

the accumulator contains 52H (01010010B) and program execution continues at label LABEL2.

Variations

JBC bit51,rel

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	3	3
States:	4	7	4	7

[Encoding]				
	0 0 0 1	0 0 0 0	bit addr	rel. addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation:

```
JBC
(PC) ← (PC) + 3
IF (bit51) = 1
        THEN
           (bit51) ← 0
          (PC) ← (PC) + rel
```

JBC bit,rel

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	5	5	4	4
States:	4	7	3	6

[Encoding]

1 0 1 0	1 0 0 1	0 0 0 1	0	y y y	direct addr	rel. addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation:

```
JBC
(PC) ← (PC) + 3
IF (bit51) = 1
THEN
(bit51) ← 0
(PC) ← (PC) + rel
```

JC rel

Function: Jump if carry is set

Description: If the CY flag is set, branch to the address specified; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative displacement in the second instruction byte to the PC, after incrementing the PC twice.

Flags:

CY	AC	OV	N	Z
!	—	—	—	—

Example: The CY flag is clear. After the instruction sequence

```
JC              LABEL1
CPL CY
JC LABEL 2
```

the CY flag is set and program execution continues at label LABEL2.

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	2	2	2	2
States:	1	4	1	4

[Encoding]

0 1 0 0	0 0 0 0	rel. addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation:

```
JC
(PC) ← (PC) + 2
IF (CY) = 1
    THEN
        (PC) ← (PC) + rel
```

JE rel

Function: Jump if equal

Description: If the Z flag is set, branch to the address specified; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative displacement in the second instruction byte to the PC, after incrementing the PC twice.

Flags:

CY	AC	OV	N	Z
—	—	—	—	!

Example: The Z flag is set. After executing the instruction

JE LABEL1

program execution continues at label LABEL1.

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	2	2
States:	2	5	1	4

[Encoding]			
	1 0 1 0	1 0 0 0	rel. addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation:

```
JE
(PC) ← (PC) + 2
IF (Z) = 1
    THEN (PC) ← (PC) + rel
```

JG rel

Function: Jump if greater than

Description: If the Z flag and the CY flag are both clear, branch to the address specified; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative displacement in the second instruction byte to the PC, after incrementing the PC twice.

Flags:

CY	AC	OV	N	Z
—	—	—	!	—

Example: The instruction

JG LABEL1

causes program execution to continue at label LABEL1 if the Z flag and the CY flag are both clear.

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	2	2
States:	2	5	1	4

[Encoding]			
	0 0 1 1	1 0 0 0	rel. addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation:

```
JG
(PC) ← (PC) + 2
IF (Z) = 0 AND (CY) = 0
        THEN (PC) ← (PC) + rel
```

JLE rel

Function: Jump if less than or equal

Description: If the Z flag or the CY flag is set, branch to the address specified; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative displacement in the second instruction byte to the PC, after incrementing the PC twice.

Flags:

CY	AC	OV	N	Z
—	—	—	!	!

Example: The instruction

JLE LABEL1

causes program execution to continue at LABEL1 if the Z flag or the CY flag is set.

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	2	2
States:	2	5	1	4

[Encoding]			
	0 0 1 0	1 0 0 0	rel. addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: JLE
(PC) ← (PC) + 2
IF (Z) = 1 OR (CY) = 1
THEN (PC) ← (PC) + rel

JMP @A+DPTR

Function: Jump indirect

Description: Add the 8-bit unsigned contents of the accumulator with the 16-bit data pointer and load the resulting sum into the lower 16 bits of the program counter. Load FFH into bits 16–23 of the program counter. This is the address for subsequent instruction fetches. The contents of the accumulator and the data pointer are not affected.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: The accumulator contains an even number from 0 to 6. The following sequence of instructions branch to one of four AJMP instructions in a jump table starting at JMP_TBL:

```
            MOV     DPTR,#JMP_TBL
            JMP     @A+DPTR
            AJMP    LABEL0
JMP_TBL:    AJMP    LABEL1
            AJMP    LABEL2
            AJMP    LABEL3
```

If the accumulator contains 04H at the start this sequence, execution jumps to LABEL2. Remember that AJMP is a two-byte instruction, so the jump instructions start at every other address.

	Binary Mode	Source Mode
Bytes:	1	1
States:	5	5

[Encoding]

0 1 1 1	0 0 1 1

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: JMP
(PC.15:0) ← (A) + (DPTR)
(PC.23:16) ← FFH

JNB bit51,rel
JNB bit,rel

Function: Jump if bit not set

Description: If the specified bit is clear, branch to the specified address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. The bit tested is not modified.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: Input port 1 contains 11001010B and the accumulator contains 56H (01010110B). After executing the instruction sequence

JNB P1.3,LABEL1
JNB ACC.3,LABEL2

program execution continues at label LABEL2.

Variations

JNB bit51,rel

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	3	3
States:	2	5	2	5

[Encoding]

0 0 1 1	0 0 0 0	bit addr	rel. addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation:

```
JNB
(PC) ← (PC) + 3
IF (bit51) = 0
    THEN (PC) ← (PC) + rel
```

JNB bit,rel

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	5	5	4	4
States:	4	7	3	6

[Encoding]

1 0 1 0	1 0 0 1	0 0 1 1	0	y y	direct addr	rel. addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation:

```
JNB
(PC) ← (PC) + 3
IF (bit) = 0
        THEN
            (PC) ← (PC) + rel
```

JNC rel

Function: Jump if carry not set

Description: If the CY flag is clear, branch to the address specified; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative displacement in the second instruction byte to the PC, after incrementing the PC twice to point to the next instruction. The CY flag is not modified.

Flags:

CY	AC	OV	N	Z
!	—	—	—	—

Example: The CY flag is set. The instruction sequence

JNC LABEL1
CPL CY
JNC LABEL2

clears the CY flag and causes program execution to continue at label LABEL2.

	Binary Mode		Source Mode	
	Not Taken	Taken	Not Taken	Taken
Bytes:	2	2	2	2
States:	1	4	1	4

[Encoding]

0 1 0 1	0 0 0 0	rel. addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation:

```
JNC
(PC) ← (PC) + 2
IF (CY) = 0
    THEN (PC) ← (PC) + rel
```

JNE rel

Function: Jump if not equal

Description: If the Z flag is clear, branch to the address specified; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative displacement in the second instruction byte to the PC, after incrementing the PC twice.

Flags:

CY	AC	OV	N	Z
—	—	—	—	!

Example: The instruction

JNE LABEL1

causes program execution to continue at LABEL1 if the Z flag is clear.

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	2	2
States:	2	5	1	4

[Encoding]

0 1 1 1	1 0 0 0	rel. addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation:

```
JNE
(PC) ← (PC) + 2
IF (Z) = 0
    THEN (PC) ← (PC) + rel
```

JNZ rel

Function: Jump if accumulator not zero

Description: If any bit of the accumulator is set, branch to the specified address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative displacement in the second instruction byte to the PC, after incrementing the PC twice. The accumulator is not modified.

Flags:

CY	AC	OV	N	Z
—	—	—	—	!

Example: The accumulator contains 00H. After executing the instruction sequence

JNZ LABEL1
INC A
JNZ LABEL2

the accumulator contains 01H and program execution continues at label LABEL2.

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	2	2	2	2
States:	2	5	2	5

[Encoding]	0 1 1 1	0 0 0 0	rel. addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation:

```
JNZ
(PC) ← (PC) + 2
IF (A) ≠ 0
    THEN (PC) ← (PC) + rel
```

JSG rel

Function: Jump if greater than (signed)

Description: If the Z flag is clear AND the N flag and the OV flag have the same value, branch to the address specified; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative displacement in the second instruction byte to the PC, after incrementing the PC twice.

Flags:

CY	AC	OV	N	Z
—	—	!	!	!

Example: The instruction

JSG LABEL1

causes program execution to continue at LABEL1 if the Z flag is clear AND the N flag and the OV flag have the same value.

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	2	2
States:	2	5	1	4

[Encoding]	0 0 0 1	1 0 0 0	rel. addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation:

```
JSG
(PC) ← (PC) + 2
IF [(N) = 0 AND (N) = (OV)]
    THEN (PC) ← (PC) + rel
```

JSGE rel

Function: Jump if greater than or equal (signed)

Description: If the N flag and the OV flag have the same value, branch to the address specified; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative displacement in the second instruction byte to the PC, after incrementing the PC twice.

Flags:

CY	AC	OV	N	Z
—	—	!	!	!

Example: The instruction

JSGE LABEL1

causes program execution to continue at LABEL1 if the N flag and the OV flag have the same value.

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	2	2
States:	2	5	1	4

[Encoding]

0 1 0 1	1 0 0 0	rel. addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation:

```
JSGE
(PC) ← (PC) + 2
IF [(N) = (OV)]
        THEN (PC) ← (PC) + rel
```

JSL rel

Function: Jump if less than (signed)

Description: If the N flag and the OV flag have different values, branch to the address specified; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative displacement in the second instruction byte to the PC, after incrementing the PC twice.

Flags:

CY	AC	OV	N	Z
—	—	!	!	!

Example: The instruction

JSL LABEL1

causes program execution to continue at LABEL1 if the N flag and the OV flag have different values.

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	2	2
States:	2	5	1	4

[Encoding]

0 1 0 0	1 0 0 0		rel. addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation:

```
JSL
(PC) ← (PC) + 2
IF (N) ≠ (OV)
     THEN (PC) ← (PC) + rel
```

JSLE rel

Function: Jump if less than or equal (signed)

Description: If the Z flag is set OR if the the N flag and the OV flag have different values, branch to the address specified; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative displacement in the second instruction byte to the PC, after incrementing the PC twice.

Flags:

CY	AC	OV	N	Z
—	—	!	!	!

Example: The instruction

JSLE LABEL1

causes program execution to continue at LABEL1 if the Z flag is set OR if the the N flag and the OV flag have different values.

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	3	3	2	2
States:	2	5	1	4

[Encoding]

0 0 0 0	1 0 0 0		rel. addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation:

```
JSLE
(PC) ← (PC) + 2
IF {(Z) = 1 OR [(N) ≠ (OV)]}
        THEN (PC) ← (PC) + rel
```

JZ rel

Function: Jump if accumulator zero

Description: If all bits of the accumulator are clear (zero), branch to the address specified; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative displacement in the second instruction byte to the PC, after incrementing the PC twice. The accumulator is not modified.

Flags:

CY	AC	OV	N	Z
—	—	—	—	!

Example: The accumulator contains 01H. After executing the instruction sequence

```
JZ LABEL1
DEC A
JZ LABEL2
```

the accumulator contains 00H and program execution continues at label LABEL2.

	Binary Mode		Source Mode	
	Not Taken	**Taken**	**Not Taken**	**Taken**
Bytes:	2	2	2	2
States:	2	5	2	5

[Encoding]	0 1 1 0	0 0 0 0	rel. addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation:

```
JZ
(PC) ← (PC) + 2
IF (A) = 0
    THEN (PC) ← (PC) + rel
```

LCALL <dest>

Function: Long call

Description: Calls a subroutine located at the specified address. The instruction adds three to the program counter to generate the address of the next instruction and then pushes the 16-bit result onto the stack (low byte first). The stack pointer is incremented by two. The high and low bytes of the PC are then loaded, respectively, with the second and third bytes of the LCALL instruction. Program execution continues with the instruction at this address. The subroutine may therefore begin anywhere in the 64-Kbyte region of memory where the next instruction is located.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: The stack pointer contains 07H and the label "SUBRTN" is assigned to program memory location 1234H. After executing the instruction

LCALL SUBRTN

at location 0123H, the stack pointer contains 09H, on-chip RAM locations 08H and 09H contain 01H and 26H, and the PC contains 1234H.

LCALL addr16

	Binary Mode	Source Mode
Bytes:	3	3
States:	9	9

[Encoding]	0 0 0 1	0 0 1 0	addr15–addr8	addr7–addr0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: LCALL
(PC) ← (PC) + 3
(SP) ← (SP) + 1
((SP)) ← (PC.7:0)
(SP) ← (SP) + 1
((SP)) ← (PC.15:8)
(PC) ← (addr.15:0)

LCALL @WRj

	Binary Mode	Source Mode
Bytes:	3	2
States:	9	8

[Encoding]	1 0 0 1	1 0 0 1	t t t t	0 1 0 0

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: LCALL
(PC) ← (PC) + 3
(SP) ← (SP) + 1
((SP)) ← (PC.7:0)
(SP) ← (SP) + 1
((SP)) ← (PC.15:8)
(PC) ← ((WRj))

LJMP <dest>

Function: Long Jump

Description: Causes an unconditional branch to the specified address, by loading the high and low bytes of the PC (respectively) with the second and third instruction bytes. The destination may therefore be anywhere in the 64-Kbyte memory region where the next instruction is located.

Flags:

CY	AC	OV	N	Z
—	—	—	—	——

Example: The label "JMPADR" is assigned to the instruction at program memory location 1234H. After executing the instruction

LJMP JMPADR

at location 0123H, the program counter contains 1234H.

LJMP addr16

	Binary Mode	Source Mode
Bytes:	3	3
States:	5	5

[Encoding]				
	0 0 0 0	0 0 1 0	addr15–addr8	addr7–addr0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: LJMP
(PC) ← (addr.15:0)

LJMP @WRj

	Binary Mode	Source Mode
Bytes:	3	2
States:	6	5

[Encoding]				
	1 0 0 0	1 0 0 1	t t t t	0 1 0 0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: LJMP
(PC) ← ((WRj))

MOV <dest>,<src>

Function: Move byte variable

Description: Copies the byte variable specified by the second operand into the location specified by the first operand. The source byte is not affected.

This is by far the most flexible operation. Twenty-four combinations of source and destination addressing modes are allowed.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: On-chip RAM location 30H contains 40H, on-chip RAM location 40H contains 10H, and input port 1 contains 11001010B (0CAH). After executing the instruction sequence

```
MOV     R0,#30H       ;R0 < = 30H
MOV     A,@R0         ;A < = 40H
MOV     R1,A          ;R1 < = 40H
MOV     B,@R1         ;B < = 10H
MOV     @R1,P1        ;RAM (40H) < = 0CAH
MOV     P2,P1         ;P2 #0CAH
```

register 0 contains 30H, the accumulator and register 1 contain 40H, register B contains 10H, and on-chip RAM location 40H and output port 2 contain 0CAH (11001010B).

Variations

MOV A,#data

	Binary Mode	Source Mode
Bytes:	2	2
States:	1	1

[Encoding]

0 1 1 1	0 1 0 0	immed. data

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: MOV
(A) ← #data

MOV dir8,#data

	Binary Mode	Source Mode
Bytes:	3	3
States:	3†	3†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

0 1 1 1	0 1 0 1	direct addr	immed. data

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: MOV
(dir8) ← #data

MOV @Ri,#data

	Binary Mode	Source Mode
Bytes:	2	3
States:	3	4

[Encoding]	0 1 1 1	0 1 1 i	immed. data

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: MOV
((Ri)) ← #data

MOV Rn,#data

	Binary Mode	Source Mode
Bytes:	2	3
States:	1	2

[Encoding]	0 1 1 1	1 r r r r	immed. data

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: MOV
(Rn) ← #data

MOV dir8,dir8

	Binary Mode	Source Mode
Bytes:	3	3
States:	3	3

[Encoding]	1 0 0 0	0 1 0 1	direct addr	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: MOV
(dir8) ← (dir8)

MOV dir8,@Ri

	Binary Mode	Source Mode
Bytes:	2	3
States:	3	4

[Encoding]	1 0 0 0	0 1 1 i	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: MOV
(dir8) ← ((Ri))

MOV dir8,Rn

	Binary Mode	Source Mode
Bytes:	2	3
States:	2†	3†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]	1 0 0 0	1 r r r	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: MOV
(dir8) ← (Rn)

MOV @Ri,dir8

	Binary Mode	Source Mode
Bytes:	2	3
States:	3	4

[Encoding]	1 0 1 0	0 1 1 i	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: MOV
((Ri)) ← (dir8)

MOV Rn,dir8

	Binary Mode	Source Mode
Bytes:	2	3
States:	1†	2†

††If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]	1 0 1 0	1 r r r	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: MOV
(Rn) ← (dir8)

MOV A,dir8

	Binary Mode	Source Mode
Bytes:	2	2
States:	1†	1

††If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

1 1 1 0	0 1 0 1	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: MOV
(A) ← (dir8)

MOV A,@Ri

	Binary Mode	Source Mode
Bytes:	1	2
States:	2	3

[Encoding]

1 1 1 0	0 1 1 i

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: MOV
(A) ← ((Ri))

MOV A,Rn

	Binary Mode	Source Mode
Bytes:	1	2
States:	1	2

[Encoding]

1 1 1 0	1 r r r

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: MOV
(A) ← (Rn)

MOV dir8,A

	Binary Mode	Source Mode
Bytes:	2	2
States:	2†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

1 1 1 1	0 1 0 1	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: MOV
(dir8) ← (A)

MOV @Ri,A

	Binary Mode	Source Mode
Bytes:	1	2
States:	3	4

[Encoding]

1 1 1 1	0 1 1 i

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: MOV
((Ri)) ← (A)

MOV Rn,A

	Binary Mode	Source Mode
Bytes:	1	2
States:	1	2

[Encoding]

1 1 1 1	1 1 1 r

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: MOV
(Rn) ← (A)

MOV Rmd,Rms

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]

0 1 1 1	1 1 0 0	s s s s	S S S S

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(Rmd) ← (Rms)

MOV WRjd,WRjs

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]

0 1 1 1	1 1 0 1	t t t t	T T T T

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(WRjd) ← (WRjs)

MOV DRkd,DRks

	Binary Mode	Source Mode
Bytes:	3	2
States:	3	2

[Encoding]

0 1 1 1	1 1 1 1	u u u u	UUUU

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(DRkd) ← (DRks)

MOV Rm,#data

	Binary Mode	Source Mode
Bytes:	4	3
States:	3	2

[Encoding]

0 1 1 1	1 1 1 0	s s s s	0 0 0 0	#data

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(Rm) ← #data

MOV WRj,#data16

	Binary Mode	Source Mode
Bytes:	5	4
States:	3	2

[Encoding]

0 1 1 1	1 1 1 0	t t t t	0 1 0 0	#data hi	#data low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(WRj) ← #data16

MOV DRk,#0data16

	Binary Mode	Source Mode
Bytes:	5	4
States:	5	4

[Encoding]

0 1 1 1	1 1 1 0	u u u u	1 0 0 0	#data hi	#data low

Hex Code in: Binary Mode = [A5][Encoding]
Source Mode = [Encoding]

Operation: MOV
(DRk) ← #0data16

MOV DRk,#1data16

	Binary Mode	Source Mode
Bytes:	5	4
States:	5	4

[Encoding]

0 1 1 1	1 1 1 0	u u u u	1 1 0 0	#data hi	#data low

Hex Code in: Binary Mode = [A5][Encoding]
Source Mode = [Encoding]

Operation: MOV
(DRk) ← #1data16

MOV Rm,dir8

	Binary Mode	Source Mode
Bytes:	4	3
States:	3†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]	0 1 1 1	1 1 1 0	s s s s	0 0 0 1	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(Rm) ← (dir8)

MOV WRj,dir8

	Binary Mode	Source Mode
Bytes:	4	3
States:	4	3

[Encoding]	0 1 1 1	1 1 1 0	t t t t	0 1 0 1	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(WRj) ← (dir8)

MOV DRk,dir8

	Binary Mode	Source Mode
Bytes:	4	3
States:	6	5

[Encoding]	0 1 1 1	1 1 1 0	u u u u	1 1 0 1	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(DRk) ← (dir8)

MOV Rm,dir16

	Binary Mode	Source Mode
Bytes:	5	4
States:	3	2

[Encoding]

0 1 1 1	1 1 1 0	s s s s	0 0 1 1	direct addr	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(Rm) ← (dir16)

MOV WRj,dir16

	Binary Mode	Source Mode
Bytes:	5	4
States:	4	3

[Encoding]

0 1 1 1	1 1 1 0	t t t t	0 1 1 1	direct addr	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(WRj) ← (dir16)

MOV DRk,dir16

	Binary Mode	Source Mode
Bytes:	5	4
States:	6	5

[Encoding]

0 1 1 1	1 1 1 0	u u u u	1 1 1 1	direct addr	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(DRk) ← (dir16)

MOV Rm,@WRj

	Binary Mode	Source Mode
Bytes:	4	3
States:	2	2

[Encoding]

0 1 1 1	1 1 1 0	t t t t	1 0 0 1	s s s s	0 0 0 0

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(Rm) ← ((WRj))

MOV Rm,@DRk

	Binary Mode	Source Mode
Bytes:	4	3
States:	4	3

[Encoding]

0111	1110	uuuu	1011	ssss	0000

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(Rm) ← ((DRk))

MOV WRjd,@WRjs

	Binary Mode	Source Mode
Bytes:	4	3
States:	4	3

[Encoding]

0000	1011	TTTT	1000	tttt	0000

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(WRjd) ← ((WRjs))

MOV WRj,@DRk

	Binary Mode	Source Mode
Bytes:	4	3
States:	5	4

[Encoding]

0000	1011	uuuu	1010	tttt	0000

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(WRj) ← ((DRk))

MOV dir8,Rm

	Binary Mode	Source Mode
Bytes:	4	3
States:	4†	3†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

0 1 1 1	1 0 1 0	s s s s	0 0 1 1	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(dir8) ← (Rm)

MOV dir8,WRj

	Binary Mode	Source Mode
Bytes:	4	3
States:	5	4

[Encoding]

0 1 1 1	1 0 1 0	t t t t	0 1 0 1	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(dir8) ← (WRj)

MOV dir8,DRk

	Binary Mode	Source Mode
Bytes:	4	3
States:	7	6

[Encoding]

0 1 1 1	1 0 1 0	u u u u	1 1 0 1	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(dir8) ← (DRk)

MOV dir16,Rm

	Binary Mode	Source Mode
Bytes:	5	4
States:	4	3

[Encoding]

0 1 1 1	1 0 1 0	s s s s	0 0 1 1	direct addr	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(dir16) ← (Rm)

MOV dir16,WRj

	Binary Mode	Source Mode
Bytes:	5	4
States:	5	4

[Encoding]

0 1 1 1	1 0 1 0	t t t t	0 1 1 1	direct addr	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(dir16) ← (WRj)

MOV dir16,DRk

	Binary Mode	Source Mode
Bytes:	5	4
States:	7	6

[Encoding]

0 1 1 1	1 0 1 0	u u u u	1 1 1 1	direct addr	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(dir16) ← (DRk)

MOV @WRj,Rm

	Binary Mode	Source Mode
Bytes:	4	3
States:	4	3

[Encoding]

0 1 1 1	1 0 1 0	t t t t	1 0 0 1	s s s s	0 0 0 0

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
((WRj)) ← (Rm)

MOV @DRk,Rm

	Binary Mode	Source Mode
Bytes:	4	3
States:	5	4

[Encoding]

0111	1010	uuuu	1011	ssss	0000

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
((DRk)) ← (Rm)

MOV @WRjd,WRjs

	Binary Mode	Source Mode
Bytes:	4	3
States:	5	4

[Encoding]

0001	1011	tttt	1000	TTTT	0000

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
((WRjd)) ← (WRjs)

MOV @DRk,WRj

	Binary Mode	Source Mode
Bytes:	4	3
States:	6	5

[Encoding]

0001	1011	uuuu	1010	tttt	0000

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
((DRk)) ← (WRj)

MOV Rm,@WRj + dis16

	Binary Mode	Source Mode
Bytes:	5	4
States:	6	5

[Encoding]

0 0 0 0	1 0 0 1	s s s s	t t t t	dis hi	dis low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(Rm) ← ((WRj)) + (dis)

MOV WRj,@WRj + dis16

	Binary Mode	Source Mode
Bytes:	5	4
States:	7	6

[Encoding]

0 1 0 0	1 0 0 1	t t t t	T T T T	dis hi	dis low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(WRj) ← ((WRj)) + (dis)

MOV Rm,@DRk + dis24

	Binary Mode	Source Mode
Bytes:	5	4
States:	7	6

[Encoding]

0 0 1 0	1 0 0 1	s s s s	u u u u	dis hi	dis low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(Rm) ← ((DRk)) + (dis)

MOV WRj,@DRk + dis24

	Binary Mode	Source Mode
Bytes:	5	4
States:	8	7

[Encoding]

0 1 1 0	1 0 0 1	t t t t	u u u u	dis hi	dis low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(WRj) ← ((DRk)) + (dis)

MOV @WRj + dis16,Rm

	Binary Mode	Source Mode
Bytes:	5	4
States:	6	5

[Encoding]

0 0 0 1	1 0 0 1	t t t t	s s s s	dis hi	dis low

Operation: MOV
((WRj)) + (dis) ← (Rm)

MOV @WRj + dis16,WRj

	Binary Mode	Source Mode
Bytes:	5	4
States:	7	6

[Encoding]

0 1 0 1	1 0 0 1	t t t t	T T T T	dis hi	dis low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
((WRj)) + (dis) ← (WRj)

MOV @DRk + dis24,Rm

	Binary Mode	Source Mode
Bytes:	5	4
States:	7	6

[Encoding]

0 0 1 1	1 0 0 1	u u u u	s s s s	dis hi	dis low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
((DRk)) + (dis) ← (Rm)

MOV @DRk + dis24,WRj

	Binary Mode	Source Mode
Bytes:	5	4
States:	8	7

[Encoding]

0 1 1 1	1 0 0 1	u u u u	t t t t	dis hi	dis low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
((DRk)) + (dis) ← (WRj)

MOV <dest–bit>,<src–bit>

Function: Move bit data

Description: Copies the Boolean variable specified by the second operand into the location specified by the first operand. One of the operands must be the CY flag; the other may be any directly addressable bit. Does not affect any other register.

Flags:

CY	AC	OV	N	Z
✓	—	—	—	—

Example: The CY flag is set, input Port 3 contains 11000101B, and output Port 1 contains 35H (00110101B). After executing the instruction sequence

MOV P1.3,CY
MOV CY,P3.3
MOV P1.2,CY

the CY flag is clear and Port 1 contains 39H (00111001B).

Variations

MOV bit51,CY

	Binary Mode	Source Mode
Bytes:	2	2
States:	2†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 2 states.

[Encoding]

1 0 0 1	0 0 1 0	bit addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: MOV
(bit51) ← (CY)

MOV CY,bit51

	Binary Mode	Source Mode
Bytes:	2	2
States:	1†	1†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

1 0 1 0	0 0 1 0		bit addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: MOV
(CY) ← (bit51)

MOV bit,CY

	Binary Mode	Source Mode
Bytes:	4	3
States:	4†	3†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 2 states.

[Encoding]

1 0 1 0	1 0 0 1		1 0 0 1	0	y y y		direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(bit) ← (CY)

MOV CY,bit

	Binary Mode	Source Mode
Bytes:	4	3
States:	3†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

1 0 1 0	1 0 0 1		1 0 1 0	0	y y y		direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOV
(CY) ← (bit)

MOV DPTR,#data16

Function: Load data pointer with a 16-bit constant

Description: Loads the 16-bit data pointer (DPTR) with the specified 16-bit constant. The high byte of the constant is loaded into the high byte of the data pointer (DPH). The low byte of the constant is loaded into the low byte of the data pointer (DPL).

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: After executing the instruction

MOV DPTR,#1234H

DPTR contains 1234H (DPH contains 12H and DPL contains 34H).

	Binary Mode	Source Mode
Bytes:	3	3
States:	2	2

[Encoding]

1 0 0 1	0 0 0 0	data hi	data low

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: MOV
(DPTR) ← #data16

MOVC A,@A+<base–reg>

Function: Move code byte

Description: Loads the accumulator with a code byte or constant from program memory. The address of the byte fetched is the sum of the original unsigned 8-bit accumulator contents and the contents of a 16-bit base register, which may be the 16 LSBs of the data pointer or PC. In the latter case, the PC is incremented to the address of the following instruction before being added with the accumulator; otherwise the base register is not altered. Sixteen-bit addition is performed.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: The accumulator contains a number between 0 and 3. The following instruction sequence translates the value in the accumulator to one of four values defined by the DB (define byte) directive.

```
RELPC:    INC     A
          MOVC    A,@A+PC
          RET
          DB      66H
          DB      77H
          DB      88H
          DB      99H
```

If the subroutine is called with the accumulator equal to 01H, it returns with 77H in the accumulator. The INC A before the MOVC instruction is needed to "get around" the RET instruction above the table. If several bytes of code separated the MOVC from the table, the corresponding number would be added to the accumulator instead.

Variations

MOVC A,@A+PC

	Binary Mode	**Source Mode**
Bytes:	1	1
States:	6	6

[Encoding]	1 0 0 0	0 0 1 1

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: MOVC
(PC) ← (PC) + 1
(A) ← ((A) + (PC))

MOVC A,@A+DPTR

	Binary Mode	**Source Mode**
Bytes:	1	1
States:	6	6

[Encoding]	1 0 0 1	0 0 1 1

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: MOVC
(A) ← ((A) + (DPTR))

MOVH DRk,#data16

Function: Move immediate 16-bit data to the high word of a dword (double-word) register.

Description: Moves 16-bit immediate data to the high word of a dword (32-bit) register. The low word of the dword register is unchanged.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: The dword register DRk contains 5566 7788H. After the instruction

MOVH DRk,#1122H

executes, DRk contains 1122 7788H.

Variations

MOVH DRk,#data16

	Binary Mode	Source Mode
Bytes:	5	4
States:	3	2

[Encoding]

0 1 1 1	1 0 1 0	u u u u	1 1 0 0	#data hi	#data low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOVH
(DRk).31:16 ← #data16

MOVS WRj,Rm

Function: Move 8-bit register to 16-bit register with sign extension

Description: Moves the contents of an 8-bit register to the low byte of a 16-bit register. The high byte of the 16-bit register is filled with the sign extension, which is obtained from the MSB of the 8-bit source register.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: Eight-bit register Rm contains 055H (01010101B) and the 16-bit register WRj contains 0FFFFH (11111111 11111111B). The instruction

MOVSE WRj,Rm

moves the contents of register Rm (01010101B) to register WRj (i.e., WRj contains 00000000 01010101B).

Variations

MOVS WRj,Rm

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]

0001	1010	tttt	ssss

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOVS
(WRj).7–0 ← (Rm).7–0
(WRj).15–8 ← MSB

MOVX <dest>,<src>

Function: Move external

Description: Transfers data between the accumulator and a byte in external data RAM. There are two types of instructions. One provides an 8-bit indirect address to external data RAM; the second provides a 16-bit indirect address to external data RAM.

In the first type of MOVX instruction, the contents of R0 or R1 in the current register bank provides an 8-bit address on port 0. Eight bits are sufficient for external I/O expansion decoding or for a relatively small RAM array. For larger arrays, any port pins can be used to output higher address bits. These pins would be controlled by an output instruction preceding the MOVX.

In the second type of MOVX instruction, the data pointer generates a 16-bit address. Port 2 outputs the upper eight address bits (from DPH) while port 0 outputs the lower eight address bits (from DPL).

For both types of moves in nonpage mode, the data is multiplexed with the lower address bits on port 0. In page mode, the data is multiplexed with the contents of P2 on port 2 (8-bit address) or with the upper address bits on port 2 (16-bit address).

It is possible in some situations to mix the two MOVX types. A large RAM array with its upper address lines driven by P2 can be addressed via the data pointer, or with code to output upper address bits to P2 followed by a MOVX instruction using R0 or R1.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: The MCS 251 controller is operating in nonpage mode. An external 256-byte RAM using multiplexed address/data lines (e.g., an Intel 8155 RAM/I/O/Timer) is connected to port 0. Port 3 provides control lines for the external RAM. ports 1 and 2 are used for normal I/O. R0 and R1 contain 12H and 34H. Location 34H of the external RAM contains 56H. After executing the instruction sequence

MOVX A,@R1
MOVX @R0,A

the accumulator and external RAM location 12H contain 56H.

Variations

MOVX A,@DPTR

	Binary Mode	Source Mode
Bytes:	1	1
States:	4	4

[Encoding]

1 1 1 0	0 0 0 0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: MOVX
(A) ← ((DPTR))

MOVX A,@Ri

	Binary Mode	Source Mode
Bytes:	1	1
States:	3	3

[Encoding]

1 1 1 0	0 0 1 i

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: MOVX
(A) ← ((Ri))

MOVX @DPTR,A

	Binary Mode	Source Mode
Bytes:	1	1
States:	5	5

[Encoding]

1 1 1 1	0 0 0 0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: MOVX
((DPTR)) ← (A)

MOVX @Ri,A

	Binary Mode	Source Mode
Bytes:	1	1
States:	4	4

[Encoding]

1 1 1 1	0 0 1 i

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: MOVX
((Ri)) ← (A)

MOVZ WRj,Rm

Function: Move 8-bit register to 16-bit register with zero extension

Description: Moves the contents of an 8-bit register to the low byte of a 16-bit register. The upper byte of the 16-bit register is filled with zeros.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: Eight-bit register Rm contains 055H (01010101B) and 16-bit register WRj contains 0FFFFH (11111111 11111111B). The instruction

MOVZ WRj,Rm

moves the contents of register Rm (01010101B) to register WRj. At the end of the operation, WRj contains 00000000 01010101B.

Variations

MOVZ WRj,Rm

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]

0 0 0 0	1 0 1 0	t t t t	s s s s

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: MOVZ
(WRj)7–0 ← (Rm)7–0
(WRj)15–8 ← 0

MUL <dest>,<src>

Function: Multiply

Description: Multiplies the unsigned integer in the register with the other unsigned integer operand. Only register addressing mode is allowed. For 8-bit operands, the result is 16 bits with the low byte stored in low byte of the destination register and high byte of the result stored in the following byte register. The OV flag is set if the product is greater than 255 (0FFH), otherwise it is cleared. If both operands are 16 bit, the result is 32 bit with the low word stored in the low word of the destination register and high word of the result stored in the following word register. In this operation, the OV flag is set if the product is greater than 0FFFFH, otherwise it is cleared. The CY flag is always cleared. The N flag is set when the MSB of the result is set. The Z flag is set when the result is zero.

Flags:

CY	AC	OV	N	Z
0	—	✓	✓	✓

Example: Register 1 contains 80 (50H or 10010000B) and register 0 contains 160 (0A0H or 10010000B). After executing the instruction

MUL R1,R0

which gives the product 12,800 (3200H), register 1 contains 32H (00110010B), register 0 contains 00H, the OV flag is set, and the CY flag is clear.

MUL Rmd,Rms

	Binary Mode	Source Mode
Bytes:	3	2
States:	6	5

[Encoding]

1 0 1 0	1 1 0 0	s s s s	S S S S

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation:

```
MUL (8-bit operands)
if <dest> md = 0, 2, 4, .., 14
Rmd ← low byte of the Rmd X Rms
Rmd+1 ← high byte of the Rmd X Rms
if <dest> md = 1, 3, 5, .., 15
Rmd–1 ← low byte of the Rmd X Rms
Rmd ← high byte of the Rmd X Rms
```

MUL WRjd,WRjs

	Binary Mode	Source Mode
Bytes:	3	2
States:	12	11

[Encoding]

1 0 1 0	1 1 0 1	t t t t	t t t t

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation:
MUL (16-bit operands)
if <dest> jd = 0, 4, 8, .., 28
WRjd ← low byte of the WRjd X WRjs
WRjd+2 ← high byte of the WRjd X WRjs
if <dest> jd = 2, 6, 10, .., 30
WRjd–2 ← low byte of the WRjd X WRjs
WRjd ← high byte of the WRjd X WRjs

MUL AB

Function: Multiply

Description: Multiplies the unsigned 8-bit integers in the accumulator and register B. The low byte of the 16-bit product is left in the accumulator, and the high byte is left in register in B. If the product is greater than 255 (0FFH) the OV flag is set; otherwise it is clear. The CY flag is always clear.

Flags:

CY	AC	OV	N	Z
0	—	✓	✓	✓

Example: The accumulator contains 80 (50H) and register B contains 160 (0A0H). After executing the instruction

MUL AB

which gives the product 12,800 (3200H), register B contains 32H (00110010B), the accumulator contains 00H, the OV flag is set, and the CY flag is clear.

	Binary Mode	**Source Mode**
Bytes:	1	1
States:	5	5

[Encoding]

1 0 1 0	0 1 0 0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation:
MUL
(A) ← low byte of (A) X (B)
(B) ← high byte of (A) X (B)

NOP

Function: No operation

Description: Execution continues at the following instruction. Affects the PC register only.

Flags:

CY	AC	OV	N	Z

—	—	—	—	—

Example: You want to produce a low-going output pulse on bit 7 of Port 2 that lasts exactly 11 states. A simple CLR-SETB sequence generates an eight-state pulse. (Each instruction requires four states to write to a port SFR.) You can insert three additional states (if no interrupts are enabled) with the following instruction sequence:

```
CLR   P2.7
NOP
NOP
NOP
SETB  P2.7
```

	Binary Mode	Source Mode
Bytes:	1	1
States:	1	1

[Encoding]

0 0 0 0	0 0 0 0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: NOP
(PC) ← (PC) + 1

ORL <dest> <src>

Function: Logical-OR for byte variables

Description: Performs the bitwise logical-OR operation (V) between the specified variables, storing the results in the destination operand.

The destination operand can be a register, an accumulator or direct address.

The two operands allow twelve addressing mode combinations. When the destination is the accumulator, the source can be register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the accumulator or immediate data. When the destination is register the source can be register, immediate, direct and indirect addressing.

Note: When this instruction is used to modify an output port, the value used as the original port data is read from the output data latch, not the input pins.

Flags:

CY	AC	OV	N	Z
—	—	—	✓	✓

Example: The accumulator contains 0C3H (11000011B) and R0 contains 55H (01010101B). After executing the instruction,

ORL A,R0

the accumulator contains 0D7H (11010111B).

When the destination is a directly addressed byte, the instruction can set combinations of bits in any RAM location or hardware register. The pattern of bits to be set is determined by a mask byte, which may be a constant data value in the instruction or a variable computed in the accumulator at run time. After executing the instruction

ORL P1,#00110010B

sets bits 5, 4, and 1 of output Port 1.

Variations

ORL dir8,A

	Binary Mode	Source Mode
Bytes:	2	2
States:	2†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 2 states.

[Encoding]			
	0 1 0 0	0 0 1 0	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ORL
(dir8) ← (dir8) V (A)

ORL dir8,#data

	Binary Mode	Source Mode
Bytes:	3	3
States:	3†	3†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]				
	0 1 0 0	0 0 1 1	direct addr	immed. data

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ORL
(dir8) ← (dir8) V #data

ORL A,#data

	Binary Mode	Source Mode
Bytes:	2	2
States:	1	1

[Encoding]			
	0 1 0 0	0 1 0 0	immed. data

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ORL
(A) ← (A) V #data

ORL A,dir8

	Binary Mode	Source Mode
Bytes:	2	2
States:	1†	1†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

0 1 0 0	0 1 0 1	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ORL
(A) ← (A) V (dir8)

ORL A,@Ri

	Binary Mode	Source Mode
Bytes:	1	2
States:	2	3

[Encoding]

0 1 0 0	0 1 1 i

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: ORL
(A) ← (A) V ((Ri))

ORL A,Rn

	Binary Mode	Source Mode
Bytes:	1	2
States:	1	2

[Encoding]

0 1 0 0	1 r r r

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: ORL
(A) ← (A) V (Rn)

ORL Rmd,Rms

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]				
	0 1 0 0	1 1 0 0	s s s s	S S S S

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ORL
(Rmd) ← (Rmd) V (Rms)

ORL WRjd,WRjs

	Binary Mode	Source Mode
Bytes:	3	2
States:	3	2

[Encoding]				
	0 1 0 0	1 1 0 1	t t t t	T T T T

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ORL
(WRjd)←(WRjd) V (WRjs)

ORL Rm,#data

	Binary Mode	Source Mode
Bytes:	4	3
States:	3	2

[Encoding]					
	0 1 0 0	1 1 1 0	s s s s	0 0 0 0	#data

Hex Code in **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ORL
(Rm) ← (Rm) V #data

ORL WRj,#data16

	Binary Mode	Source Mode
Bytes:	5	4
States:	4	3

[Encoding]

0 1 0 0	1 1 1 0	t t t t	0 1 0 0	#data hi	#data low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ORL
(WRj) ← (WRj) V #data16

ORL Rm,dir8

	Binary Mode	Source Mode
Bytes:	4	3
States:	3†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

0 1 0 0	1 1 1 0	s s s s	0 0 0 1	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ORL
(Rm) ← (Rm) V (dir8)

ORL WRj,dir8

	Binary Mode	Source Mode
Bytes:	4	3
States:	4	3

[Encoding]

0 1 0 0	1 1 1 1	t t t t	0101	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ORL
(WRj) ← (WRj) V (dir8)

ORL Rm,dir16

	Binary Mode	Source Mode
Bytes:	5	4
States:	3	2

[Encoding]

0 1 0 0	1 1 1 0	s s s s	0 0 1 1	direct addr	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ORL
(Rm) ← (Rm) V (dir16)

ORL WRj,dir16

	Binary Mode	Source Mode
Bytes:	5	4
States:	4	3

[Encoding]

0100	1110	tttt	0111	direct addr	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ORL
(WRj) ← (WRj) V (dir16)

ORL Rm,@WRj

	Binary Mode	Source Mode
Bytes:	4	3
States:	3	2

[Encoding]

0100	1110	tttt	1001	ssss	0000

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ORL
(Rm) ← (Rm) V ((WRj))

ORL Rm,@DRk

	Binary Mode	Source Mode
Bytes:	4	3
States:	4	3

[Encoding]

0100	1110	uuuu	1011	ssss	0000

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ORL
(Rm) ← (Rm) V ((DRk))

ORL CY,<src–bit>

Function: Logical-OR for bit variables

Description: Sets the CY flag if the Boolean value is a logical 1; leaves the CY flag in its current state otherwise . A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is not affected.

Flags:

CY	AC	OV	N	Z
✓	—	—	—	—

Example: Set the CY flag if and only if P1.0 = 1, ACC. 7 = 1, or OV = 0:

```
MOV CY,P1.0            ;LOAD CARRY WITH INPUT PIN P10
ORL CY,ACC.7 ;OR CARRY WITH THE ACC. BIT 7
ORL CY,/OV             ;OR CARRY WITH THE INVERSE OF OV.
```

Variations

ORL CY,bit51

	Binary Mode	Source Mode
Bytes:	2	2
States:	1†	1†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]			
	0 1 1 1	0 0 1 0	bit addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ORL
(CY) ← (CY) V (bit51)

ORL CY,/bit51

	Binary Mode	Source Mode
Bytes:	2	2
States:	1†	1†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]			
	1 0 1 0	0 0 0 0	bit addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: ORL
(CY) ← (CY) V¬ (bit51)

ORL CY,bit

	Binary Mode	Source Mode
Bytes:	4	3
States:	3†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

1 0 1 0	1 0 0 1	0 1 1 1	0	y y y	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ORL
(CY) ← (CY) V (bit)

ORL CY,/bit

	Binary Mode	Source Mode
Bytes:	4	3
States:	3†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

1 0 1 0	1 0 0 1	1 1 1 0	0	y y y	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: ORL
(CY) ← (CY) V ¬ (bit)

POP <src>

Function: Pop from stack.

Description: Reads the contents of the on-chip RAM location addressed by the stack pointer, then decrements the stack pointer by one. The value read at the original RAM location is transferred to the newly addressed location, which can be 8-bit or 16-bit.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: The stack pointer contains 32H and on-chip RAM locations 30H through 32H contain 01H, 23H, and 20H, respectively. After executing the instruction sequence

POP DPH
POP DPL

the stack pointer contains 30H and the data pointer contains 0123H. After executing the instruction

POP SP

the stack pointer contains 20H. Note that in this special case the stack pointer was decremented to 2FH before it was loaded with the value popped (20H).

Variations

POP dir8

	Binary Mode	Source Mode
Bytes:	2	2
States:	3	3

[Encoding]	1 1 0 1	0 0 0 0	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: POP
(dir8) ← ((SP))
(SP) ← (SP) – 1

POP Rm

	Binary Mode	Source Mode
Bytes:	3	2
States:	3	2

[Encoding]	1 1 0 1	1 0 1 0	s s s s	1 0 0 0

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: POP
(Rm) ← ((SP))
(SP) ← (SP) – 1

POP WRj

	Binary Mode	Source Mode
Bytes:	3	2
States:	5	4

[Encoding]	1 1 0 1	1 0 1 0	t t t t	1 0 0 1

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: POP
(WRj) ← ((SP))
(SP) ← (SP) – 2

POP DRk

	Binary Mode	Source Mode
Bytes:	3	2
States:	10	9

[Encoding]

1101	1010	uuuu	1101

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: POP
(DRk) ← ((SP))
(SP) ← (SP) – 2

PUSH <dest>

Function: Push onto stack

Description: Increments the stack pointer by one. The contents of the specified variable are then copied into the on-chip RAM location addressed by the stack pointer.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: On entering an interrupt routine, the stack pointer contains 09H and the data pointer contains 0123H. After executing the instruction sequence

PUSH DPL
PUSH DPH

the stack pointer contains 0BH and on-chip RAM locations 0AH and 0BH contain 01H and 23H, respectively.

Variations

PUSH dir8

	Binary Mode	Source Mode
Bytes:	2	2
States:	4	4

[Encoding]

1100	0000	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: PUSH
(SP) ← (SP) + 1
((SP)) ← (dir8)

PUSH #data

	Binary Mode	Source Mode
Bytes:	4	3
States:	4	3

[Encoding]

1 1 0 0	1 0 1 0	0 0 0 0	0 0 1 0	#data

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: PUSH
(SP) ← (SP) + 1
((SP)) ← #data

PUSH #data16

	Binary Mode	Source Mode
Bytes:	5	4
States:	6	5

[Encoding]

1 1 0 0	1 0 1 0	0 0 0 0	0 1 1 0	#data hi	#data lo

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: PUSH
(SP) ← (SP) + 2
((SP)) ← #data16

PUSH Rm

	Binary Mode	Source Mode
Bytes:	3	2
States:	4	3

[Encoding]

1 1 0 0	1 0 1 0	s s s s	1 0 0 0

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: PUSH
(SP) ← (SP) + 1
((SP)) ← (Rm)

PUSH WRj

	Binary Mode	Source Mode
Bytes:	3	2
States:	5	4

[Encoding]

1 1 0 0	1 0 1 0	t t t t	1 0 0 1

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: PUSH
(SP) ← (SP) + 2
((SP)) ← (WRj)

PUSH DRk

	Binary Mode	Source Mode
Bytes:	3	2
States:	9	8

[Encoding]

1 1 0 0	1 0 1 0	u u u u	1 1 0 1

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: PUSH
(SP) ← (SP) + 4
((SP)) ← (DRk)

RET

Function: Return from subroutine

Description: Pops the high and low bytes of the PC successively from the stack, decrementing the stack pointer by two. Program execution continues at the resulting address, which normally is the instruction immediately following ACALL or LCALL.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: The stack pointer contains 0BH and on-chip RAM locations 0AH and 0BH contain 01H and 23H, respectively. After executing the instruction,

RET

the stack pointer contains 09H and program execution continues at location 0123H.

	Binary Mode	Source Mode
Bytes:	1	1
States:	7	7

[Encoding]

0 0 1 0	0 0 1 0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: RET
(PC).15–8 ← ((SP))
(SP) ← (SP) – 1
(PC).7–8 ← ((SP))
(SP) ← (SP) – 1

RETI

Function: Return from interrupt

Description: This instruction pops two or four bytes from the stack, depending on the INTR bit in the CONFIG1 register .

If INTR = 0, RETI pops the high and low bytes of the PC successively from the stack and uses them as the 16-bit return address in region FF:.The stack pointer is decremented by two. No other registers are affected, and neither PSW nor PSW1 is automatically restored to its pre-interrupt status.

If INTR = 1, RETI pops four bytes from the stack: PSW1 and the three bytes of the PC. The three bytes of the PC are the return address, which can be anywhere in the 16-Mbyte memory space. The stack pointer is decremented by four. PSW1 is restored to its pre-interrupt status, but PSW is **not** restored to its pre-interrupt status. No other registers are affected.

For either value of INTR1, hardware restores the interrupt logic to accept additional interrupts at the same priority level as the one just processed. Program execution continues at the return address, which normally is the instruction immediately after the point at which the interrupt request was detected. If an interrupt of the same or lower priority is pending when the RETI instruction is executed, that one instruction is executed before the pending interrupt is processed.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: INTR1 = 0. The stack pointer contains 0BH. An interrupt was detected during the instruction ending at location 0122H. On-chip RAM locations 0AH and 0BH contain 01H and 23H, respectively. After executing the instruction,

RETI

the stack pointer contains 09H and program execution continues at location 0123H.

	Binary Mode	Source Mode
Bytes:	1	1
States (INTR = 0):	9	9
States (INTR = 1):	12	12

[Encoding]

0 0 1 1	0 0 1 0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation for for INTR1 = 0:

RETI
(PC).7:0 ← ((SP))
(SP) ← (SP) – 1
(PC).15:8 ← ((SP))
(SP) ← (SP) – 1

Operation for INTR1 = 1:

RETI
X ← ((SP))
(SP) ← (SP) – 1
X ← ((SP))
(SP) ← (SP) – 1
X ← ((SP))
(SP) ← (SP) – 1
X ← ((SP))
(SP) ← (SP) – 1

RL A

Function: Rotate accumulator left

Description: Rotates the eight bits in the accumulator one bit to the left. Bit 7 is rotated into the bit 0 position.

Flags:

CY	AC	OV	N	Z
—	—	—	✓	✓

Example: The accumulator contains 0C5H (11000101B). After executing the instruction,

RL A

the accumulator contains 8BH (10001011B); the CY flag is unaffected.

	Binary Mode	Source Mode
Bytes:	1	1
States:	1	1

[Encoding]

0 0 1 0	0 0 1 1

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: RL
(A).a+1 ← (A).a
(A).0 ← (A).7

RLC A

Function: Rotate accumulator left through the carry flag

Description: Rotates the eight bits in the accumulator and the CY flag one bit to the left. Bit 7 moves into the CY flag position and the original state of the CY flag moves into bit 0 position.

Flags:

CY	AC	OV	N	Z
✓	—	—	✓	✓

Example: The accumulator contains 0C5H (11000101B) and the CY flag is clear. After executing the instruction

RLC A

the accumulator contains 8AH (10001010B) and the CY flag is set.

	Binary Mode	Source Mode
Bytes:	1	1
States:	1	1

[Encoding]

0 0 1 1	0 0 1 1

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: RLC
(A).a+1 ← (A).a
(A).0 ← (CY)
(CY) ← (A) .7

RR A

Function: Rotate accumulator right

Description: Rotates the 8 or 16 bits in the accumulator one bit to the right. Bit 0 is moved into the bit 7 or 15 position.

Flags:

CY	AC	OV	N	Z
—	—	—	✓	✓

Example: The accumulator contains 0C5H (11000101B). After executing the instruction,

RR A

the accumulator contains 0E2H (11100010B) and the CY flag is unaffected.

	Binary Mode	Source Mode
Bytes:	1	1
States:	1	1

[Encoding]

0 0 0 0	0 0 1 1

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: RR
(A).a ← (A).a+1
(A).7 ← (A) .0

RRC A

Function: Rotate accumulator right through carry flag

Description: Rotates the eight bits in the accumulator and the CY flag one bit to the right. Bit 0 moves into the CY flag position; the original value of the CY flag moves into the bit 7 position.

Flags:

CY	AC	OV	N	Z
✓	—	—	✓	✓

Example: The accumulator contains 0C5H (11000101B) and the CY flag is clear. After executing the instruction

RRC A

the accumulator contains 62 (01100010B) and the CY flag is set.

	Binary Mode	Source Mode
Bytes:	1	1
States:	1	1

[Encoding]

0 0 0 1	0 0 1 1

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: RRC
(A).a ← (A).a+1
(A).7 ← (CY)
(CY) ← (A).0

SETB <bit>

Function: Set bit

Description: Sets the specified bit to one. SETB can operate on the CY flag or any directly addressable bit.

Flags: No flags are affected except the CY flag for instruction with CY as the operand.

CY	AC	OV	N	Z
✓	—	—	—	—

Example: The CY flag is clear and output Port 1 contains 34H (00110100B). After executing the instruction sequence,

SETB CY
SETB P1.0

the CY flag is set and output Port 1 contains 35H (00110101B).

SETB bit51

	Binary Mode	Source Mode
Bytes:	2	2
States:	2†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 2 states.

[Encoding]

1 1 0 1	0 0 1 0	bit addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: SETB
(bit51) ← 1

SETB CY

	Binary Mode	Source Mode
Bytes:	1	1
States:	1	1

[Encoding]

1 1 0 1	0 0 1 1

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: SETB
(CY) ← 1

SETB bit

	Binary Mode	Source Mode
Bytes:	4	3
States:	4†	3†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 2 states.

[Encoding]

1 0 1 0	1 0 0 1	1 1 0 1	0	y y y	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SETB
(bit) ← 1

SJMP rel

Function: Short jump

Description: Program control branches unconditionally to the specified address. The branch destination is computed by adding the signed displacement in the second instruction byte to the PC, after incrementing the PC twice. Therefore, the range of destinations allowed is from 128 bytes preceding this instruction to 127 bytes following it.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: The label "RELADR" is assigned to an instruction at program memory location 0123H. The instruction

SJMP RELADR

assembles into location 0100H. After executing the instruction, the PC contains 0123H.

(Note: In the above example, the instruction following SJMP is located at 102H. Therefore, the displacement byte of the instruction is the relative offset (0123H–0102H) = 21H. Put another way, an SJMP with a displacement of 0FEH would be a one-instruction infinite loop.)

	Binary Mode	Source Mode
Bytes:	2	2
States:	4	4

[Encoding]

1 0 0 0	0 0 0 0	rel. addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: SJMP
(PC) ← (PC) + 2
(PC) ← (PC) + rel

SLL <src>

Function: Shift logical left by 1 bit

Description: Shifts the specified variable to the left by 1 bit, replacing the LSB with zero.

Flags:

CY	AC	OV	N	Z
✓	—	—	✓	✓

Example: Register 1 contains 0C5H (11000101B). After executing the instruction

SLL register 1

Register 1 contains 8AH (10001010B).

Variations

SLL Rm

	Binary Mode	**Source Mode**
Bytes:	3	2
States:	2	1

[Encoding]

0011	1110	ssss	0000

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SLL
(Rm).a+1 ← (Rm).a
(Rm).0 ← 0

SLL WRj

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]

0011	1110	tttt	0100

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SLL
(WRj).b+1 ← (WRj).b
(WRj).0 ← 0

SRA <src>

Function: Shift arithmetic right by 1 bit

Description: Shifts the specified variable to the arithmetic right by 1 bit. The MSB is unchanged.

Flags:

CY	AC	OV	N	Z
✓	—	—	✓	✓

Example: Register 1 contains 0C5H (11000101B). After executing the instruction

SRA register 1

Register 1 contains 0E2H (11100010B).

Variations

SRA Rm

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]	0 0 0 0	1 1 1 0	s s s s	0 0 0 0

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SRA
(Rm).7 ← (Rm).7
(Rm).a ← (Rm).a+1

SRA WRj

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]	0 0 0 0	1 1 1 0	t t t t	0 1 0 0

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SRA
(WRj).15 ← (WRj).15
(WRj).b ← (WRj).b+1

SRL <src>

Function: Shift logical right by 1 bit

Description: SRL shifts the specified variable to the right by 1 bit, replacing the MSB with a zero.

Flags:

CY	AC	OV	N	Z
✓	—	—	✓	✓

Example: Register 1 contains 0C5H (11000101B). After executing the instruction

SRL register 1

Register 1 contains 62H (01100010B).

Variations

SRL Rm

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]

0 0 0 1	1 1 1 0		s s s s	0 0 0 0

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SRL
(Rm).7 ← 0
(Rm).a ← (Rm).a+1

SRL WRj

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]

0 0 0 1	1 1 1 0		t t t t	0 1 0 0

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SRL
(WRj).15 ← 0
(WRj).b ← (WRj).b+1

SUB <dest>,<src>

Function: Subtract

Description: Subtracts the specified variable from the destination operand, leaving the result in the destination operand. SUB sets the CY (borrow) flag if a borrow is needed for bit 7. Otherwise, CY is clear.

When subtracting signed integers, the OV flag indicates a negative number produced when a negative value is subtracted from a positive value, or a positive result when a positive number is subtracted from a negative number.

The source operand allows four addressing modes: immediate, indirect, register and direct.

Flags:

CY	AC	OV	N	Z
✓	✓†	✓	✓	✓

†For word and dword subtractions, AC is not affected.

Example: Register 1 contains 0C9H (11001001B) and register 0 contains 54H (01010100B). After executing the instruction

SUB R1,R0

register 1 contains 75H (01110101B), the CY and AC flags are clear, and the OV flag is set.

Variations

SUB Rmd,Rms

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]

1 0 0 1	1 1 0 0	s s s s	S S S S

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SUB
(Rmd) ← (Rmd) – (Rms)

SUB WRjd,WRjs

	Binary Mode	Source Mode
Bytes:	3	2
States:	3	2

[Encoding]

1 0 0 1	1 1 0 1	t t t t	T T T T

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SUB
(WRjd) ← (WRjd) – (WRjs)

SUB DRkd,DRks

	Binary Mode	Source Mode
Bytes:	3	2
States:	5	4

[Encoding]

1 0 0 1	1 1 1 1	u u u u	U U U U

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SUB
(DRkd) ← (DRkd) – (DRks)

SUB Rm,#data

	Binary Mode	Source Mode
Bytes:	4	3
States:	3	2

[Encoding]

1 0 0 1	1 1 1 0	s s s s	0 0 0 0	#data

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SUB
(Rm) ← (Rm) – #data

SUB WRj,#data16

	Binary Mode	Source Mode
Bytes:	5	4
States:	4	3

[Encoding]

1 0 0 1	1 1 1 0	t t t t	0 1 0 0	#data hi	#data low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SUB
(WRj) ← (WRj) – #data16

SUB DRk,#data16

	Binary Mode	**Source Mode**
Bytes:	5	4
States:	6	5

[Encoding]

1001	1110	u u u u	1000	#data hi	#data low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SUB
(DRk) ← (DRk) – #data16

SUB Rm,dir8

	Binary Mode	**Source Mode**
Bytes:	4	3
States:	3†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

1001	1110	s s s s	0001	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SUB
(Rm) ← (Rm) – (dir8)

SUB WRj,dir8

	Binary Mode	**Source Mode**
Bytes:	4	3
States:	4	3

[Encoding]

1001	1110	t t t t	0101	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SUB
(WRj) ← (WRj) – (dir8)

SUB Rm,dir16

	Binary Mode	Source Mode
Bytes:	5	4
States:	3	2

[Encoding]

1001	1110	ssss	0011	direct addr	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SUB
(Rm) ← (Rm) – (dir16)

SUB WRj,dir16

	Binary Mode	Source Mode
Bytes:	5	4
States:	4	3

[Encoding]

1001	1110	tttt	0111	direct addr	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SUB
(WRj) ← (WRj) – (dir16)

SUB Rm,@WRj

	Binary Mode	Source Mode
Bytes:	4	3
States:	3	2

[Encoding]

1001	1110	tttt	1001	ssss	0000

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SUB
(Rm) ← (Rm) – ((WRj))

SUB Rm,@DRk

	Binary Mode	Source Mode
Bytes:	4	3
States:	4	3

[Encoding]

1001	1110	uuuu	1011	ssss	0000

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: SUB
(Rm) ← (Rm) – ((DRk))

SUBB A,<src–byte>

Function: Subtract with borrow

Description: SUBB subtracts the specified variable and the CY flag together from the accumulator, leaving the result in the accumulator. SUBB sets the CY (borrow) flag if a borrow is needed for bit 7, and clears CY otherwise. (If CY was set before executing a SUBB instruction, this indicates that a borrow was needed for the previous step in a multiple precision subtraction, so the CY flag is subtracted from the accumulator along with the source operand.) AC is set if a borrow is needed for bit 3, and cleared otherwise. OV is set if a borrow is needed into bit 6, but not into bit 7, or into bit 7, but not bit 6.
When subtracting signed integers the OV flag indicates a negative number produced when a negative value is subtracted from a positive value, or a positive result when a positive number is subtracted from a negative number.

The source operand allows four addressing modes: register, direct, register-indirect, or immediate.

Flags:

CY	AC	OV	N	Z
✓	✓	✓	✓	✓

Example: The accumulator contains 0C9H (11001001B), register 2 contains 54H (01010100B), and the CY flag is set. After executing the instruction

SUBB A,R2

the accumulator contains 74H (01110100B), the CY and AC flags are clear, and the OV flag is set.

Notice that 0C9H minus 54H is 75H. The difference between this and the above result is due to the CY (borrow) flag being set before the operation. If the state of the carry is not known before starting a single or multiple-precision subtraction, it should be explicitly cleared by a CLR CY instruction.

Variations

SUBB A,#data

	Binary Mode	Source Mode
Bytes:	2	2
States:	1	1

[Encoding]

1 0 0 1	0 1 0 0	immed. data

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: SUBB
(A) ← (A) – (CY) – #data

SUBB A,dir8

	Binary Mode	Source Mode
Bytes:	2	2
States:	1†	1†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

1 0 0 1	0 1 0 1		direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: SUBB
(A) ← (A) – (CY) – (dir8)

SUBB A,@Ri

	Binary Mode	Source Mode
Bytes:	1	2
States:	2	3

[Encoding]

1 0 0 1	0 1 1 i

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: SUBB
(A) ← (A) – (CY) – ((Ri))

SUBB A,Rn

	Binary Mode	Source Mode
Bytes:	1	2
States:	1	2

[Encoding]

1 0 0 1	1 r r r

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: SUBB
(A) ← (A) – (CY) – (Rn)

SWAP A

Function: Swap nibbles within the accumulator

Description: Interchanges the low and high nibbles (4-bit fields) of the accumulator (bits 3–0 and bits 7–4). This operation can also be thought of as a 4-bit rotate instruction.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: The accumulator contains 0C5H (11000101B). After executing the instruction

SWAP A

the accumulator contains 5CH (01011100B).

	Binary Mode	Source Mode
Bytes:	1	1
States:	2	2

[Encoding]

1 1 0 0	0 1 0 0

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: SWAP
$(A).3:0 \rightarrow \leftarrow (A).7:4$

TRAP

Function: Causes interrupt call

Description: Causes an interrupt call that is vectored through location 0FF007BH. The operation of this instruction is not affected by the state of the interrupt enable flag in PSW0 and PSW1. Interrupt calls can not occur immediately following this instruction. This instruction is intended for use by Intel-provided development tools. These tools do not support user application of this instruction.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: The instruction

TRAP

causes an interrupt call to location 0FF007BH during normal operation.

	Binary Mode	Source Mode
Bytes:	2	1
States (2 bytes):	11	10
States (4 bytes):	16	15

[Encoding]

1 0 1 1	1 0 0 1

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: TRAP
SP ← SP – 2
(SP) ← PC
PC ← (0FF007BH)

XCH A,<byte>

Function: Exchange accumulator with byte variable

Description: Loads the accumulator with the contents of the specified variable, at the same time writing the original accumulator contents to the specified variable. The source/destination operand can use register, direct, or register-indirect addressing.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: R0 contains the address 20H, the accumulator contains 3FH (00111111B) and on-chip RAM location 20H contains 75H (01110101B). After executing the instruction

XCH A,@R0

RAM location 20H contains 3FH (00111111B) and the accumulator contains 75H (01110101B).

Variations

XCH A,dir8

	Binary Mode	Source Mode
Bytes:	2	2
States:	3†	3†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 2 states.

[Encoding]

1 1 0 0	0 1 0 1	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: XCH
(A) → ← (dir8)

XCH A,@Ri

	Binary Mode	**Source Mode**
Bytes:	1	2
States:	4	5

[Encoding]

1 1 0 0	0 1 1 i

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: XCH
(A) → ← ((Ri))

XCH A,Rn

	Binary Mode	Source Mode
Bytes:	1	2
States:	3	4

[Encoding]

1 1 0 0	1 r r r

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: XCH
(A) → ← (Rn)

Variations

XCHD A,@Ri

Function: Exchange digit

Description: Exchanges the low nibble of the accumulator (bits 3-0), generally representing a hexadecimal or BCD digit, with that of the on-chip RAM location indirectly addressed by the specified register. Does not affect the high nibble (bits 7-4) of either register.

Flags:

CY	AC	OV	N	Z
—	—	—	—	—

Example: R0 contains the address 20H, the accumulator contains 36H (00110110B), and on-chip RAM location 20H contains 75H (01110101B). After executing the instruction,

XCHD A,@R0

on-chip RAM location 20H contains 76H (01110110B) and 35H (00110101B) in the accumulator.

	Binary Mode	Source Mode
Bytes:	1	2
States:	4	5

[Encoding]

1 1 0 1	0 1 1 i

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: XCHD
(A).3:0 → ← ((Ri)).3:0

XRL <dest>,<src>

Function: Logical Exclusive-OR for byte variables

Description: Performs the bitwise logical Exclusive-OR operation (∀) between the specified variables, storing the results in the destination. The destination operand can be the accumulator, a register, or a direct address.

The two operands allow 12 addressing mode combinations. When the destination is the accumulator or a register, the source addressing can be register, direct, register-indirect, or immediate; when the destination is a direct address, the source can be the accumulator or immediate data.

(Note: When this instruction is used to modify an output port, the value used as the original port data is read from the output data latch, not the input pins.)

Flags:

CY	AC	OV	N	Z
—	—	—	✓	✓

Example: The accumulator contains 0C3H (11000011B) and R0 contains 0AAH (10101010B). After executing the instruction,

XRL A,R0

the accumulator contains 69H (01101001B).

When the destination is a directly addressed byte, this instruction can complement combinations of bits in any RAM location or hardware register. The pattern of bits to be complemented is then determined by a mask byte, either a constant contained in the instruction or a variable computed in the accumulator at run time. The instruction

XRL P1,#00110001B

complements bits 5, 4, and 0 of output Port 1.

Variations

XRL dir8,A

	Binary Mode	Source Mode
Bytes:	2	2
States:	2†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 2 states.

[Encoding]	0 1 1 0	0 0 1 0	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: XRL
(dir8) ← (dir8) ∀ (A)

XRL dir8,#data

	Binary Mode	Source Mode
Bytes:	3	3
States:	3†	3†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]	0 1 1 0	0 0 1 1	direct addr	immed. data

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: XRL
(dir8) ← (dir8) ∀ #data

XRL A,#data

	Binary Mode	Source Mode
Bytes:	2	2
States:	1	1

[Encoding]	0 1 1 0	0 1 0 0	immed. data

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: XRL
(A) ← (A) ∀ #data

XRL A,dir8

	Binary Mode	**Source Mode**
Bytes:	2	2
States:	1†	1†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

0 1 1 0	0 1 0 1	direct addr

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [Encoding]

Operation: XRL
(A) ← (A) ∀ (dir8)

XRL A,@Ri

	Binary Mode	**Source Mode**
Bytes:	1	2
States:	2	3

[Encoding]

0 1 1 0	0 1 1 i

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: XRL
(A) ← (A) ∀ ((Ri))

XRL A,Rn

	Binary Mode	Source Mode
Bytes:	1	2
States:	1	2

[Encoding]

0 1 1 0	1 r r r

Hex Code in: **Binary Mode = [Encoding]**
Source Mode = [A5][Encoding]

Operation: XRL
(A) ← (A) ∀ (Rn)

XRL Rmd,Rms

	Binary Mode	Source Mode
Bytes:	3	2
States:	2	1

[Encoding]

0 1 1 0	1 1 0 0	s s s s	S S S S

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: XRL
(Rmd) ← (Rmd) ∀ (Rms)

XRL WRjd,WRjs

	Binary Mode	Source Mode
Bytes:	3	2
States:	3	2

[Encoding]

0 1 1 0	1 1 0 1	t t t t	T T T T

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: XRL
(WRds) ← (WRjd) ∀ (WRjs)

XRL Rm,#data

	Binary Mode	Source Mode
Bytes:	4	3
States:	3	2

[Encoding]

0 1 1 0	1 1 1 0	s s s s	0 0 0 0	#data

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: XRL
(Rm) ← (Rm) ∀ #data

XRL WRj,#data16

	Binary Mode	Source Mode
Bytes:	5	4
States:	4	3

[Encoding]

0 1 1 0	1 1 1 0	t t t t	0 1 0 0	#data hi	#data low

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: XRL
(WRj) ← (WRj) ∀ #data16

XRL Rm,dir8

	Binary Mode	Source Mode
Bytes:	4	3
States:	3†	2†

†If this instruction addresses a port (P*x*, *x* = 0–3), add 1 state.

[Encoding]

0 1 1 0	1 1 1 0	s s s s	0 0 0 1	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: XRL
(Rm) ← (Rm) ∀ (dir8)

XRL WRj,dir8

	Binary Mode	Source Mode
Bytes:	4	3
States:	4	3

[Encoding]

0 1 1 0	1 1 1 0	t t t t	0 1 0 1	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: XRL
(WRj) ← (WRj) ∀ (dir8)

XRL Rm,dir16

	Binary Mode	**Source Mode**
Bytes:	5	4
States:	3	2

[Encoding]

0 1 1 0	1 1 1 0	s s s s	0 0 1 1	direct addr	dir8 addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: XRL
(Rm) ← (Rm) ∀ (dir16)

XRL WRj,dir16

	Binary Mode	Source Mode
Bytes:	5	4
States:	4	3

[Encoding]

0 1 1 0	1 1 1 0	t t t t	0 1 1 1	direct addr	direct addr

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: XRL
(WRj) ← (WRj) ∀ (dir16)

XRL Rm,@Wrj

	Binary Mode	Source Mode
Bytes:	4	3
States:	3	2

[Encoding]

0 1 1 0	1 1 1 0	t t t t	1 0 0 1	s s s s	0 0 0 0

Hex Code in: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: XRL
(Rm) ← (Rm) ∀ ((WRj))

XRL Rm,@Drk

	Binary Mode	**Source Mode**
Bytes:	4	3
States:	4	3

[Encoding]

0 1 1 0	1 1 1 0	u u u u	1 0 1 1	s s s s	0 0 0 0

Hex Code In: **Binary Mode = [A5][Encoding]**
Source Mode = [Encoding]

Operation: XRL
(Rm) ← (Rm) ∀ ((DRk))

B

Signal Descriptions

APPENDIX B
SIGNAL DESCRIPTIONS

This appendix provides reference information for the pin functions of the 8XC251SB. Table B-1 lists the signals, grouped by function. Table B-2 defines the columns used in Table B-3, which describes the signals.

Table B-1. Signals Arranged by Functional Categories

Address & Data
Name
AD0/P0.0
AD1/P0.1
AD2/P0.2
AD3/P0.3
AD4/P0.4
AD5/P0.5
AD6/P0.6
AD7/P0.7
A8/P2.0
A9/P2.1
A10/P2.2
A11/P2.3
A12/P2.4
A13/P2.5
A14/P2.6
A15/P2.7

Input/Output
Name
T2/P1.0
T2EX/P1.1
ECI/P1.2
CEX0/P1.3
CEX1/P1.4
CEX2/P1.5
CEX3/P1.6
CEX4/P1.7
RXD/P3.0
TXD/P3.1
T0/P3.4
T1/P3.5

Bus Control & Status
Name
WR#/P3.6
RD#/P3.7
ALE/PROG#
PSEN#

Processor Control
Name
INT0#/P3.2
INT1#/P3.3
EA#/V_{PP}
RST
XTAL1
XTAL2

Power & Ground
Name
V_{CC}
V_{CC2}
V_{SS}
V_{SS1}
V_{SS2}

Table B-2. Description of Columns of Table B-3

Column Heading	Description
Signal Name	Lists the signals, arranged alphabetically. Many pins have two functions, so there are more entries in this column than there are pins. Every signal is listed in this column; for each signal, the alternate function that shares the pin is listed in the *Multiplexed With* column.
Type	Identifies the pin function listed in the *Signal Name* column as an input (I), output (O), bidirectional (I/O), power (PWR), or ground (GND). Note that all inputs except RESET# are *sampled inputs*. RESET# is a level-sensitive input. During powerdown mode, the powerdown circuitry uses EXTINT*x* as a level-sensitive input.
Description	Briefly describes the function of the pin for the specific signal listed in the *Signal Name* column.
Multiplexed With	Lists the multiplexed signal name for the alternate function that the pin provides (if applicable).

Table B-3. Signal Descriptions

Signal Name	Type	Description	Multiplexed With
A16	O	**Address Line 16**. See RD#.	N.A.
A15:8†	O	**Address Lines**. Upper address lines for the external bus.	P2.7:0
AD7:0†	I/O	**Address/Data Lines**. Multiplexed lower address lines and data lines for external memory.	P0.7:0
ALE	O	**Address Latch Enable**. ALE signals the start of an external bus cycle and indicates that valid address information is available on lines A15:8 and AD7:0. An external latch can use ALE to demultiplex the address from the address/data bus.	PROG#
CEX4:0	I/O	**Programmable Counter Array (PCA) Input/Output Pins**. These are input signals for the PCA capture mode and output signals for the PCA compare mode and PCA PWM mode.	P1.7:3
EA#	I	**External Access**. Directs program memory accesses to on-chip or off-chip code memory. For EA# = 0, all program memory accesses are off-chip. For EA# = 1, an access is to on-chip OTPROM/ROM if the address is within the range of the on-chip OTPROM/ROM; otherwise the access is off-chip. The value of EA# is latched at reset. For a ROMless part, EA# must be strapped to ground.	V_{PP}
ECI	I	**PCA External Clock Input**. External clock input to the 16-bit PCA timer.	P1.2
INT1:0#	I	**External Interrupts 0 and 1**. These inputs set bits IE1:0 in the TCON register. If bits IT1:0 in the TCON register are set, bits IE1:0 are set by a falling edge on INT1#/INT0#. If bits INT1:0 are clear, bits IE1:0 are set by a low level on INT1:0#.	P3.3:2

† The descriptions of A15:8/P2.7:0 and AD7:0/P0.7:0 are for the nonpage-mode chip configuration (compatible with PLCC MCS 51 microcontrollers). If the chip is configured for page-mode operation, port 0 carries the lower address bits (A0–A7), and port 2 carries the upper address bits (A8–A15) and the data (D0–D7).

Table B-3. Signal Descriptions (Continued)

Signal Name	Type	Description	Multiplexed With
P0.7:0	I/O	**Port 0**. This is an 8-bit, open-drain, bidirectional I/O port.	AD7:0
P1.0 P1.1 P1.2 P1.7:3	I/O	**Port 1**. This is an 8-bit, bidirectional I/O port with internal pullups.	T2 T2EX ECI CEX4:0
P2.7:0	I/O	**Port 2**. This is an 8-bit, bidirectional I/O port with internal pullups.	A15:8
P3.0 P3.1 P3.3:2 P3.5:4 P3.6 P3.7	I/O	**Port 3**. This is an 8-bit, bidirectional I/O port with internal pullups.	RXD TXD INT1:0# T1:0 WR# RD#
PROG#	I	**Programming Pulse**. The programming pulse is applied to this pin for programming the on-chip OTPROM.	ALE
PSEN#	O	**Program Store Enable**. Read signal output. This output is asserted for a memory address range that depends on bits RD0 and RD1 in configuration byte CONFIG1 (see also RD#): **RD1RD0Address Range for Assertion** 00Reserved 01All addresses 10All addresses 11All addresses ≥ 80:0000H	—
RD#	O	**Read or 17th Address Bit (A16).** Read signal output to external data memory or 17th external address bit (A16), depending on the values of bits RD0 and RD1 in configuration byte CONFIG1. (See also PSEN#): **RD1RD0Function** 00Reserved 01The pin functions as A16 only. 10The pin functions as P3.7 only. 11RD#: asserted for reads at all addresses ≤ 7F:FFFFH	P3.7
RST	I	**Reset**. Reset input to the chip. Holding this pin high for 64 oscillator periods while the oscillator is running resets the device. The port pins are driven to their reset conditions when a voltage greater than V_{IH1} is applied, whether or not the oscillator is running. This pin has an internal pulldown resistor, which allows the device to be reset by connecting a capacitor between this pin and V_{CC}. Asserting RST when the chip is in idle mode or powerdown mode returns the chip to normal operation.	—
RXD	I/O	**Receive Serial Data**. RXD sends and receives data in serial I/O mode 0 and receives data in serial I/O modes 1, 2, and 3.	P3.0
T1:0	I	**Timer 1:0 External Clock Inputs**. When timer 1:0 operates as a counter, a falling edge on the T1:0 pin increments the count.	P3.5:4

† The descriptions of A15:8/P2.7:0 and AD7:0/P0.7:0 are for the nonpage-mode chip configuration (compatible with PLCC MCS 51 microcontrollers). If the chip is configured for page-mode operation, port 0 carries the lower address bits (A0–A7), and port 2 carries the upper address bits (A8–A15) and the data (D0–D7).

Table B-3. Signal Descriptions (Continued)

Signal Name	Type	Description	Multiplexed With
T2	I/O	**Timer 2 Clock Input/Output**. For the timer 2 capture mode, this signal is the external clock input. For the clock-out mode, it is the timer 2 clock output.	P1.0
T2EX	I	**Timer 2 External Input**. In timer 2 capture mode, a falling edge initiates a capture of the timer 2 registers. In auto-reload mode, a falling edge causes the timer 2 registers to be reloaded. In the up-down counter mode, this signal determines the count direction: 1 = up, 0 = down.	P1.1
TXD	O	**Transmit Serial Data**. TXD outputs the shift clock in serial I/O mode 0 and transmits serial data in serial I/O modes 1, 2, and 3.	P3.1
V_{CC}	PWR	**Supply Voltage**. Connect this pin to the +5V supply voltage.	—
V_{CC2}	PWR	**Secondary Supply Voltage 2.** This supply voltage connection is provided to reduce power supply noise. Connection of this pin to the +5V supply voltage is recommended. However, when using the 8XC251SB as a pin-for-pin replacement for the 8XC51FX, V_{SS2} can be unconnected without loss of compatibility.	—
V_{PP}	I	**Programming Supply Voltage**. The programming supply voltage is applied to this pin for programming the on-chip OTPROM.	EA#
V_{SS}	GND	**Circuit Ground**. Connect this pin to ground.	—
V_{SS1}	GND	**Secondary Ground**. This ground is provided to reduce ground bounce and improve power supply bypassing. Connection of this pin to ground is recommended. However, when using the 8XC251SB as a pin-for-pin replacement for the 8XC51BH, V_{SS1} can be unconnected without loss of compatibility.	—
V_{SS2}	GND	**Secondary Ground 2**. This ground is provided to reduce ground bounce and improve power supply bypassing. Connection of this pin to ground is recommended. However, when using the 8XC251SB as a pin-for-pin replacement for the 8XC51FX, V_{SS2} can be unconnected without loss of compatibility.	—
WR#	O	**Write**. Write signal output to external memory. For configuration bits RD1 = RD0 = 1, WR# is strobed only for writes to locations 00 0000H–01 FFFFH. For other values of RD1 and RD0, WR# is strobed for writes to all memory locations.	P3.6
XTAL1	I	**Input to the On-chip, Inverting, Oscillator Amplifier**. To use the internal oscillator, a crystal/resonator circuit is connected to this pin. If an external oscillator is used, its output is connected to this pin. XTAL1 is the clock source for internal timing.	—
XTAL2	O	**Output of the On-chip, Inverting, Oscillator Amplifier**. To use the internal oscillator, a crystal/resonator circuit is connected to this pin. If an external oscillator is used, leave XTAL2 unconnected.	—

† The descriptions of A15:8/P2.7:0 and AD7:0/P0.7:0 are for the nonpage-mode chip configuration (compatible with PLCC MCS 51 microcontrollers). If the chip is configured for page-mode operation, port 0 carries the lower address bits (A0–A7), and port 2 carries the upper address bits (A8–A15) and the data (D0–D7).

C

Registers

APPENDIX C
REGISTERS

Table C-1. 8XC251SB Special Function Registers (SFRs)

SFR Mnemonic	SFR Name	Hex Address	Binary Reset Value	
			High	Low
ACC[†]	Accumulator	S:E0H	0000	0000
B[†]	B Register	S:F0H	0000	0000
CCAP0H	PCA Module 0 Compare/Capture Register High Byte	S:FAH	XXXX	XXXX
CCAP0L	PCA Module 0 Compare/Capture Register Low Byte	S:EAH	XXXX	XXXX
CCAP1H	PCA Module 1 Compare/Capture Register High Byte	S:FBH	XXXX	XXXX
CCAP1L	PCA Module 1 Compare/Capture Register Low Byte	S:EBH	XXXX	XXXX
CCAP2H	PCA Module 2 Compare/Capture Register High Byte	S:FCH	XXXX	XXXX
CCAP2L	PCA Module 2 Compare/Capture Register Low Byte	S:ECH	XXXX	XXXX
CCAP3H	PCA Module 3 Compare/Capture Register High Byte	S:FDH	XXXX	XXXX
CCAP3L	PCA Module 3 Compare/Capture Register Low Byte	S:EDH	XXXX	XXXX
CCAP4H	PCA Module 4 Compare/Capture Register High Byte	S:FEH	XXXX	XXXX
CCAP4L	PCA Module 4 Compare/Capture Register Low Byte	S:EEH	XXXX	XXXX
CCAPM0	PCA Compare/Capture Module 0 Mode Register	S:DAH	X000	0000
CCAPM1	PCA Compare/Capture Module 1 Mode Register	S:DBH	X000	0000
CCAPM2	PCA Compare/Capture Module 2 Mode Register	S:DCH	X000	0000
CCAPM3	PCA Compare/Capture Module 3 Mode Register	S:DDH	X000	0000
CCAPM4	PCA Compare/Capture Module 4 Mode Register	S:DEH	X000	0000
CCON	PCA Timer/Counter Control Register	S:D8H	00X0	0000

[†] This register resides in the register file. It can also be accessed as an SFR.

Table C-1. 8XC251SB Special Function Registers (SFRs)

SFR Mnemonic	SFR Name	Hex Address	Binary Reset Value	
			High	Low
CH	PCA Timer/Counter High Byte	S:F9H	0000	0000
CL	PCA Timer/Counter Low Byte	S:E9H	0000	0000
CMOD	PCA Timer/Counter Mode Register	S:D9H	00XX	X000
DPH[†]	Data Pointer High	S:83H	0000	0000
DPL[†]	Data Pointer Low	S:82H	0000	0000
DPXL[†]	Data Pointer Extended Low	S:84H	0000	0001
IE0	Interrupt Enable Control Register 0	S:A8H	0000	0000
IPH0	Interrupt Priority High Control Register 0	S:B7H	X000	0000
IPL0	Interrupt Priority Low Control Register 0	S:B8H	X000	0000
P0	Port 0	S:80H	1111	1111
P1	Port 1	S:90H	1111	1111
P2	Port 2	S:A0H	1111	1111
P3	Port 3	S:B0H	1111	1111
PCON	Power Control Register	S:87H	00XX	0000
PSW	Program Status Word	S:D0H	0000	0000
PSW1	Program Status Word 1	S:D1H	0000	0000
RCAP2H	Timer 2 Reload/Capture Register High Byte	S:CBH	0000	0000
RCAP2L	Timer 2 Reload/Capture Register Low Byte	S:CAH	0000	0000
SADDR	Slave Individual Address Register	S:A9H	0000	0000
SADEN	Mask Byte Register	S:B9H	0000	0000
SBUF	Serial Data Buffer	S:99H	XXXX	XXXX
SCON	Serial Control Register	S:98H	0000	0000
SP[†]	Stack Pointer - LS byte of SPX	S:81H	0000	0111
SPH[†]	Stack Pointer High - MSB of SPX	S:BDH	0000	0000
T2CON	Timer 2 Control Register	S:C8H	0000	0000
T2MOD	Timer 2 Mode Control Register	S:C9H	XXXX	XX00
TCON	Timer 0/1 Control Register	S:88H	0000	0000
TMOD	Timer 0/1 Mode Control Register	S:89H	0000	0000
TH0	Timer 0 Timer Register High Byte	S:8CH	0000	0000
TL0	Timer 0 Timer Register Low Byte	S:8AH	0000	0000

[†] This register resides in the register file. It can also be accessed as an SFR.

Table C-1. 8XC251SB Special Function Registers (SFRs)

SFR Mnemonic	SFR Name	Hex Address	Binary Reset Value	
			High	Low
TH1	Timer 1 Timer Register High Byte	S:8DH	0000	0000
TL1	Timer 1 Timer Register Low Byte	S:8BH	0000	0000
TH2	Timer 2 Timer Register High Byte	S:CDH	0000	0000
TL2	Timer 2 Timer Register Low Byte	S:CCH	0000	0000
WDTRST	Watchdog Timer Reset Register	S:A6H	XXXX	XXXX

[†] This register resides in the register file. It can also be accessed as an SFR.

ACC

Address: E0H
Reset State: 0000 0000B

Accumulator. ACC provides SFR access to the accumulator, which resides in the register file as byte register R11 (also named ACC). Instructions in the MCS® 51 architecture use the accumulator as both source and destination for calculations and moves. Instructions in the MCS 251 architecture assign no special significance to R11. These instructions can use byte registers Rm (m = 0–15) interchangeably.

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

Bit Number	Bit Mnemonic	Function
7:0	ACC.7:0	Accumulator.

B

Address: F0H
Reset State: 0000 0000B

B Register. The B register provides SFR access to byte register R10 (also named B) in the register file. The B register is used as both a source and destination in multiply and divide operations. For all other operations, the B register is available for use as one of the byte registers Rm, m = 0–15.

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

Bit Number	**Bit Mnemonic**	**Function**
7:0	B.7:0	B Register.

CCAPxH, CCAPxL (x = 0–4)

Address: CCAP0H,L S:FAH, S:EAH
CCAP1H,L S:FBH, S:EBH
CCAP2H,L S:FCH, S:ECH
CCAP3H,L S:FDH, S:EDH
CCAP4H,L S:FEH, S:EEH

Reset State: XXXX XXXXB

PCA Module Compare/Capture Registers. These five register pairs store the 16-bit comparison value or captured value for the corresponding compare/capture modules. In the PWM mode, the low-byte register controls the duty cycle of the output waveform.

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

Bit Number	**Bit Mnemonic**	**Function**
7:0	CCAP*x*H.7:0 CCAP*x*L.7:0	High byte of PCA comparison or capture values. Low byte of PCA comparison or capture values.

CCAPM*x* (*x* = 0–4)

Address: CCAPM0 S:DAH
CCAPM1 S:DBH
CCAPM2 S:DCH
CCAPM3 S:DDH
CCAPM4 S:DEH

Reset State: X000 0000B

PCA Compare/Capture Module Mode Registers. These five registers select the operating mode of the corresponding compare/capture module. Each register also contains an enable interrupt bit (ECCFx) for generating an interrupt request when the module's compare/capture flag (CCF*x* in the CCON register) is set. See Table 8-3 on page 8-15 for mode select bit combinations.

7							**0**
—	ECOM*x*	CAPP*x*	CAPN*x*	MAT*x*	TOG*x*	PWM*x*	ECCF*x*

Bit Number	**Bit Mnemonic**	**Function**
7	—	Reserved: The value read from this bit is indeterminate. Do not write a "1" to this bit.
6	ECOM*x*	Compare Modes: ECOM*x* = 1 enables the module comparator function. The comparator is used to implement the software timer, high-speed output, pulse width modulation, and watchdog timer modes.
5	CAPP*x*	Capture Mode (Positive): CAPP*x* = 1 enables the capture function with capture triggered by a positive edge on pin CEX*x*.
4	CAPN*x*	Capture Mode (Negative): CAPN*x* = 1 enables the capture function with capture triggered by a negative edge on pin CEX*x*.
3	MAT*x*	Match: Set ECOM*x* and MAT*x* to implement the software timer mode. When MAT*x* = 1, a match of the PCA timer/counter with the compare/capture register sets the CCF*x* bit in the CCON register, flagging an interrupt.
2	TOG*x*	Toggle: Set ECOM*x*, MAT*x*, and TOG*x* to implement the high-speed output mode. When TOG*x* = 1, a match of the PCA timer/counter with the compare/capture register toggles the CEX*x* pin.
1	PWM*x*	Pulse Width Modulation Mode: PWM*x* = 1 configures the module for operation as an 8-bit pulse width modulator with output waveform on the CEX*x* pin.
0	ECCF*x*	Enable CCF*x* Interrupt: Enables compare/capture flag CCF*x* in the CCON register to generate an interrupt request.

CCON

Address: S:D8H
Reset State: 00X0 0000B

PCA Timer/Counter Control Register. Contains the run control bit and overflow flag for the PCA timer/counter, and the compare/capture flags for the five PCA compare/capture modules.

7							0
CF	CR	—	CCF4	CCF3	CCF2	CCF1	CCF0

Bit Number	Bit Mnemonic	Function
7	CF	PCA Timer/Counter Overflow Flag: Set by hardware when the PCA timer/counter rolls over. This generates an interrupt request if the ECF interrupt enable bit in CMOD is set. CF can be set by hardware or software but can be cleared only by software.
6	CR	PCA Timer/Counter Run Control Bit: Set and cleared by software to turn the PCA timer/counter on and off.
5	—	Reserved: The value read from this bit is indeterminate. Do not write a “1” to this bit.
4:0	CCF4 CCF3 CCF2 CCF1 CCF0	PCA Module Compare/Capture Flags: Set by hardware when a match or capture occurs. This generates a PCA interrupt request if the ECCF*x* interrupt enable bit in the corresponding CCAPM*x* register is set. Must be cleared by software.

CH, CL

Address: S:F9H
S:E9H

Reset State: 0000 0000B

CH, CL Registers. These registers operate in cascade to form the 16-bit PCA timer/counter.

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

Bit Number	Bit Mnemonic	Function
7:0	CH.7:0 CL.7:0	High byte of the PCA timer/counter Low byte of the PCA timer/counter

CMOD

Address: S:D9H
Reset State: 00XX X000B

PCA Timer/Counter Mode Register. Contains bits for selecting the PCA timer/counter input, disabling the PCA timer/counter during idle mode, enabling the PCA WDT reset output (module 4 only), and enabling the PCA timer/counter overflow interrupt.

7							**0**
CIDL	WDTE	—	—	—	CPS1	CPS0	ECF

Bit Number	Bit Mnemonic	Function
7	CIDL	PCA Timer/Counter Idle Control: CIDL = 1 disables the PCA timer/counter during idle mode. CIDL = 0 allows the PCA timer/counter to run during idle mode.
6	WDTE	Watchdog Timer Enable: WDTE = 1 enables the watchdog timer output on PCA module 4. WDTE = 0 disables the PCA watchdog timer output.
5:3	—	Reserved: The values read from these bits are indeterminate. Do not write a “1” to these bits.
2:1	CPS1:0	PCA Timer/Counter Input Select: **CPS1 CPS0** 0 0 F_{OSC} /12 0 1 F_{OSC} /4 1 0 Timer 0 overflow 1 1 External clock at ECI pin (maximum rate = F_{OSC} /8)
0	ECF	PCA Timer/Counter Interrupt Enable: ECF = 1 enables the CF bit in the CCON register to generate an interrupt request.

DPH

Address: S:83H
Reset State: 0000 0000B

Data Pointer High. DPH provides SFR access to register file location 58 (also named DPH). DPH is the upper byte of the 16-bit data pointer, DPTR. Instructions in the MCS® 51 architecture use DPTR for data moves, code moves, and for a jump instruction (JMP @A+DPTR). See also DPL and DPXL.

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

Bit Number	**Bit Mnemonic**	**Function**
7:0	DPH.7:0	Data Pointer High: Bits 8–15 of the extended data pointer, DPX (DR56).

DPL

Address: S:82H
Reset State: 0000 0000B

Data Pointer Low. DPL provides SFR access to register file location 59 (also named DPL). DPL is the low byte of the 16-bit data pointer, DPTR. Instructions in the MCS® 51 architecture use the 16-bit data pointer for data moves, code moves, and for a jump instruction (JMP @A+DPTR). See also DPH and DPXL.

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

Bit Number	Bit Mnemonic	Function
7:0	DPL.7:0	Data Pointer Low: Bits 0–7 of the extended data pointer, DPX (DR56).

DPXL

Address: S:84H
Reset State: 0000 0001B

Data Pointer Extended Low. DPXL provides SFR access to register file location 57 (also named DPXL). Location 57 is the lower byte of the upper word of the extended data pointer, DPX = DR56, whose lower word is the 16-bit data pointer, DPTR. See also DPH and DPL.

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

Bit Number	Bit Mnemonic	Function
7:0	DPL.7:0	Data Pointer Extended Low: Bits 16–23 of the extended data pointer, DPX (DR56).

IE0

Address: S:A8H
Reset State: 0000 0000B

Interrupt Enable Register 0. IE0 contains two types of interrupt enable bits. The global enable bit (EA) enables/disables all of the interrupts, except the TRAP interrupt, which is always enabled. The remaining bits enable/disable the other individual interrupts.

7							0
EA	EC	ET2	ES	ET1	EX1	ET0	EXO

Bit Number	Bit Mnemonic	Function
7	EA	Global Interrupt Enable: Setting this bit enables all interrupts that are individually enabled by bits 0–6. Clearing this bit disables all interrupts, except the TRAP interrupt, which is always enabled.
6	EC	PCA Interrupt Enable: Setting this bit enables the PCA interrupt.
5	ET2	Timer 2 Overflow Interrupt Enable: Setting this bit enables the timer 2 overflow interrupt.
4	ES	Serial I/O Port Interrupt Enable: Setting this bit enables the serial I/O port interrupt.
3	ET1	Timer 1 Overflow Interrupt Enable: Setting this bit enables the timer 1 overflow interrupt.
2	EX1	External Interrupt 1 Enable: Setting this bit enables external interrupt 1.
1	ET0	Timer 0 Overflow Interrupt Enable: Setting this bit enables the timer 0 overflow interrupt.
0	EX0	External Interrupt 0 Enable: Setting this bit enables external interrupt 0.

IPH0

Address: S:B7H
Reset State: X000 0000B

Interrupt Priority High Control Register 0. IPH0, together with IPL0, assigns each interrupt a priority level from 0 (lowest) to 3 (highest):

IPH0.*x*	**IPL0.*x***	**Priority Level**
0	0	0 (lowest priority)
0	1	1
1	0	2
1	1	3 (highest priority)

7							**0**
—	IPH0.6	IPH0.5	IPH0.4	IPH0.3	IPH0.2	IPH0.1	IPH0.0

Bit Number	Bit Mnemonic	Function
7	—	Reserved. The value read from this bit is indeterminate. Do not write a “1” to this bit.
6	IPH0.6	PCA Interrupt Priority Bit High
5	IPH0.5	Timer 2 Overflow Interrupt Priority Bit High
4	IPH0.4	Serial I/O Port Interrupt Priority Bit High
3	IPH0.3	Timer 1 Overflow Interrupt Priority Bit High
2	IPH0.2	External Interrupt Priority Bit High
1	IPH0.1	Timer 0 Overflow Interrupt Priority Bit High
0	IPH0.0	External Interrupt 0 Priority Bit High

IPL0

Address: S:B8H
Reset State: X000 0000B

Interrupt Priority Low Control Register 0. IPL0, together with IPH0, assigns each interrupt a priority level from 0 (lowest) to 3 (highest):

IPH0.*x*	**IPL0.*x***	**Priority Level**
0	0	0 (lowest priority)
0	1	1
1	0	2
1	1	3 (highest priority)

7							**0**
—	IPL0.6	IPL0.5	IPL0.4	IPL0.3	IPL0.2	IPL0.1	IPL0.0

Bit Number	**Bit Mnemonic**	**Function**
7	—	Reserved. The value read from this bit is indeterminate. Do not write a “1” to this bit.
6	IPL0.6	PCA Interrupt Priority Bit Low
5	IPL0.5	Timer 2 Overflow Interrupt Priority Bit Low
4	IPL0.4	Serial I/O Port Interrupt Priority Bit Low
3	IPL0.3	Timer 1 Overflow Interrupt Priority Bit Low
2	IPL0.2	External Interrupt Priority Bit Low
1	IPL0.1	Timer 0 Overflow Interrupt Priority Bit Low
0	IPL0.0	External Interrupt 0 Priority Bit Low

P0

Address: S:80H
Reset State: 1111 1111B

Port 0. P0 is the SFR that contains data to be driven out from the port 0 pins. Read-modify-write instructions that read port 0 read this register. The other instructions that read port 0 read the port 0 pins. When port 0 is used for an external bus cycle, the CPU always writes FFH to P0, and the former contents of P0 are lost.

7 **0**

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

Bit Number	Bit Mnemonic	Function
7:0	P0.7:0	Port 0 Register: Write data to be driven onto the port 0 pins to these bits.

P1

Address: S:90H
Reset State: 1111 1111B

Port 1. P1 is the SFR that contains data to be driven out from the port 1 pins. Read-write-modify instructions that read port 1 read this register. Other instructions that read port 1 read the port 1 pins.

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

Bit Number	Bit Mnemonic	Function
7:0	P1.7:0	Port 1 Register: Write data to be driven onto the port 1 pins to these bits.

P2

Address: S:A0H
Reset State: 1111 1111B

Port 2. P2 is the SFR that contains data to be driven out from the port 2 pins. Read-modify-write instructions that read port 2 read this register. Other instructions that read port 2 read the port 2 pins.

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

Bit Number	Bit Mnemonic	Function
7:0	P2.7:0	Port 2 Register: Write data to be driven onto the port 2 pins to these bits.

P3

Address: S:B0H
Reset State: 1111 1111B

Port 3. P3 is the SFR that contains data to be driven out from the port 3 pins. Read-modify-write instructions that read port 3 read this register. Other instructions that read port 3 read the port 3 pins.

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

Bit Number	Bit Mnemonic	Function
7:0	P3.7:0	Port 3 Register: Write data to be driven onto the port 3 pins to these bits.

PCON

Address: S:87H
Reset State: 00XX 0000B

Power Control Register. Contains the power off flag (POF) and bits for enabling the idle and powerdown modes. Also contains two general-purpose flags and two bits that control serial I/O functions—the double baud rate bit and a bit that selects whether accesses to SCON.7 are to the FE bit or the SM0 bit.

7							**0**
SMOD1	SMOD0	—	POF	GF1	GF0	PD	IDL

Bit Number	**Bit Mnemonic**	**Function**
7	SMOD1	Double Baud Rate Bit: When set, doubles the baud rate when timer 1 is used and mode 1, 2, or 3 is selected in the SCON register. See “Baud Rates” on page 9-10.
6	SMOD0	SCON.7 Select: When set, read/write accesses to SCON.7 are to the FE bit. When clear, read/write accesses to SCON.7 are to the SM0 bit. See Figure 9-2 on page 9-3.
5	—	Reserved: The value read from this bit is indeterminate. Do not write a “1” to this bit.
4	POF	Power Off Flag: Set by hardware as V_{CC} rises above 3 V to indicate that power has been off or V_{CC} had fallen below 3 V and that on-chip volatile memory is indeterminate. Set or cleared by software.
3	GF1	General Purpose Flag: Set or cleared by software. One use is to indicate whether an interrupt occurred during normal operation or during idle mode.
2	GF0	General Purpose Flag: Set or cleared by software. One use is to indicate whether an interrupt occurred during normal operation or during idle mode.
1	PD	Powerdown Mode Bit: When set, activates powerdown mode. Cleared by hardware when an interrupt or reset occurs.
0	IDL	Idle Mode Bit: When set, activates idle mode. Cleared by hardware when an interrupt or reset occurs. If IDL and PD are both set, PD takes precedence.

PSW

Address: S:D0H
Reset State: 0000 0000B

Program Status Word. PSW contains bits that reflect the results of operations, bits that select the register bank for registers R0–R7, and two general-purpose flags that are available to the user.

7							0
CY	AC	F0	RS1	RS0	OV	UD	P

Bit Number	Bit Mnemonic	Function
7	CY	Carry Flag: The carry flag is set by an addition instruction (ADD, ADDC) if there is a carry out of the MSB. It is set by a subtraction (SUB, SUBB) or compare (CMP) if a borrow is needed for the MSB. The carry flag is also affected by some rotate and shift instructions, logical bit instructions, bit move instructions, and the multiply (MUL) and decimal adjust (DA) instructions (see Table 4-11 on page 4-18).
6	AC	Auxiliary Carry Flag: The auxiliary carry flag is affected only by instructions that address 8-bit operands. The AC flag is set if an arithmetic instruction with an 8-bit operand produces a carry out of bit 3 (from addition) or a borrow into bit 3 (from subtraction). Otherwise it is cleared. This flag is useful for BCD arithmetic (see Table 4-11 on page 4-18).
5	F0	Flag 0: This general-purpose flag is available to the user.
4:3	RS1:0	Register Bank Select Bits 1 and 0: These bits select the memory locations that comprise the active bank of the register file (registers R0–R7). **RS1 RS0 Bank Address** 0 0 0 00H–07H 0 1 1 08H–0FH 1 0 2 10H–17H 1 1 3 18H–1FH
2	OV	Overflow Flag: This bit is set if an addition or subtraction of signed variables results in an overflow error (i.e., if the magnitude of the sum or difference is too great for the seven LSBs in 2's-complement representation). The overflow flag is also set if a multiplication product overflows one byte or if a division by zero is attempted.
1	UD	User-definable Flag: This general-purpose flag is available to the user.
0	P	Parity Bit: This bit indicates the parity of the accumulator. It is set if an odd number of bits in the accumulator are set. Otherwise, it is cleared. Not all instructions update the parity bit.

PSW1

Address: S:D1H
Reset State: 0000 0000B

Program Status Word 1. PSW1 contains bits that reflect the results of operations and bits that select the register bank for registers R0–R7.

7							0
CY	AC	N	RS1	RS0	OV	Z	—

Bit Number	Bit Mnemonic	Function
7	CY	Carry Flag: Identical to the CY bit in the PSW register on page C-22.
6	AC	Auxiliary Carry Flag: Identical to the AC bit in the PSW register on page C-22.
5	N	Negative Flag: This bit is set if the result of the last logical or arithmetic operation was negative. Otherwise it is cleared.
4:3	RS1:0	Register Bank Select Bits 0 and 1: Identical to the RS1:0 bits in the PSW register on page C-22.
2	OV	Overflow Flag: Identical to the OV bit in the PSW register page C-22.
1	Z	Zero Flag: This flag is set if the result of the last logical or arithmetic operation is zero. Otherwise it is cleared.
0	—	Reserved: The value read from this bit is indeterminate. Do not write a “1” to this bit.

RCAP2H, RCAP2L

Address: RCAP2H S:CBH
RCAP2L S:CAH

Reset State: 0000 0000B

Timer 2 Reload/Capture Registers. This register pair stores 16-bit values to be loaded into or captured from the timer register (TH2/TL2) in timer 2.

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

Bit Number	Bit Mnemonic	Function
7:0	RCAP2H.7:0 RCAP2L.7:0	High byte of the timer 2 reload/recapture register Low byte of the timer 2 reload/recapture register

SADDR

Address: S:A9H
Reset State: 0000 0000B

Slave Individual Address Register. SADDR contains the device's individual address for multiprocessor communication.

7 **0**

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

Bit Number	Bit Mnemonic	Function
7:0	SADDR.7:0	

SADEN

Address: S:B9H
Reset State: 0000 0000B

Mask Byte Register. This register masks bits in the SADDR register to form the device's given address for multiprocessor communication.

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

Bit Number	Bit Mnemonic	Function
7:0	SADEN.7:0	

SBUF

Address: S:99H
Reset State: XXXX XXXXB

Serial Data Buffer. Writing to SBUF loads the transmit buffer of the serial I/O port. Reading SBUF reads the receive buffer of the serial I/O port.

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

Bit Number	Bit Mnemonic	Function
7:0	SBUF.7:0	

SCON

Address: 98H
Reset State: 0000 0000B

Serial Port Control Register. SCON contains serial I/O control and status bits, including the mode select bits and the interrupt flag bits.

7							0
FE/SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Bit Number	Bit Mnemonic	Function
7	FE	Framing Error Bit: To select this function, set the SMOD0 bit in the PCON register. Set by hardware to indicate an invalid stop bit. Cleared by software, not by valid frames.
	SM0	Serial Port Mode Bit 0: To select this function, clear the SMOD0 bit in the PCON register. Software writes to bits SM0 and SM1 to select the serial port operating mode. Refer to the SM1 bit for the mode selections.
6	SM1	Serial Port Mode Bit 1: Software writes to bits SM1 and SM0 (above) to select the serial port operating mode. **SM0 SM1 Mode Description Baud Rate** 0 0 0 Shift register $F_{OSC}/12$ 0 1 1 8-bit UART Variable 1 0 2 9-bit UART $F_{OSC}/32^{\dagger}$ or $F_{OSC}/64^{\dagger}$ 1 1 3 9-bit UART Variable †Select by programming the SMOD bit in the PCON register (see “Baud Rates” on page 9-10).
5	SM2	Serial Port Mode Bit 2: Software writes to bit SM2 to enable and disable the multiprocessor communication and automatic address recognition features. This allows the serial port to differentiate between data and command frames and to recognize slave and broadcast addresses.
4	REN	Receiver Enable Bit: To enable reception, set this bit. To enable transmission, clear this bit.
3	TB8	Transmit Bit 8: In modes 2 and 3, software writes the ninth data bit to be transmitted to TB8. Not used in modes 0 and 1.
2	RB8	Receiver Bit 8: Mode 0: Not used. Mode 1 (SM2 clear): Set or cleared by hardware to reflect the stop bit received. Modes 2 and 3 (SM2 set): Set or cleared by hardware to reflect the ninth data bit received.

SCON

Address: 98H
Reset State: 0000 0000B

Serial Port Control Register. SCON contains serial I/O control and status bits, including the mode select bits and the interrupt flag bits.

7							0
FE/SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Bit Number	Bit Mnemonic	Function
1	TI	Transmit Interrupt Flag Bit: Set by the transmitter after the last data bit is transmitted. Cleared by software.
0	RI	Receive Interrupt Flag Bit: Set by the receiver after the last data bit of a frame has been received. Cleared by software.

SP

Address: S:81H
Reset State: 0000 0111B

Stack Pointer. SP provides SFR access to location 63 in the register file (also named SP). SP is the lowest byte of the extended stack pointer (SPX = DR60). The extended stack pointer points to the current top of stack. When a byte is saved (PUSHed) on the stack, SPX is incremented, and then the byte is written to the top of stack. When a byte is retrieved (POPped) from the stack, it is copied from the top of stack, and then SPX is decremented.

7 **0**

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

Bit Number	Bit Mnemonic	Function
7:0	SP.7:0	Stack Pointer: Bits 0–7 of the extended stack pointer, SPX (DR60).

SPH

Address: S:BDH
Reset State: 0000 0000B

Stack Pointer High. SPH provides SFR access to location 62 in the register file (also named SPH). SPH is the upper byte of the lower word of DR60, the extended stack pointer (SPX). The extended stack pointer points to the current top of stack. When a byte is saved (PUSHed) on the stack, SPX is incremented, and then the byte is written to the top of stack. When a byte is retrieved (POPped) from the stack, it is copied from the top of stack, and then SPX is decremented.

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

Bit Number	Bit Mnemonic	Function
7:0	SPH.7:0	Stack Pointer High: Bits 8–15 of the extended stack pointer, SPX (DR(60)).

T2CON

Address: S:C8H
Reset State: 0000 0000B

Timer 2 Control Register. Contains the receive clock, transmit clock, and capture/reload bits used to configure timer 2. Also contains the run control bit, counter/timer select bit, overflow flag, external flag, and external enable for timer 2.

7							0
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2#	CP/RL2#

Bit Number	Bit Mnemonic	Function
7	TF2	Timer 2 Overflow Flag: Set by timer 2 overflow. Must be cleared by software. TF2 is not set if RCLK = 1 or TCLK = 1.
6	EXF2	Timer 2 External Flag: If EXEN2 = 1, capture or reload caused by a negative transition on T2EX sets EFX2. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1)
5	RCLK	Receive Clock Bit: Selects timer 2 overflow pulses (RCLK = 1) or timer 1 overflow pulses (RCLK = 0) as the baud rate generator for serial port modes 1 and 3.
4	TCLK	Transmit Clock Bit: Selects timer 2 overflow pulses (TCLK = 1) or timer 1 overflow pulses (TCLK = 0) as the baud rate generator for serial port modes 1 and 3.
3	EXEN2	Timer 2 External Enable Bit: Setting EXEN2 causes a capture or reload to occur as a result of a negative transition on T2EX unless timer 2 is being used as the baud rate generator for the serial port. Clearing EXEN2 causes timer 2 to ignore events at T2EX.
2	TR2	Timer 2 Run Control Bit: Setting this bit starts the timer.
1	C/T2#	Timer 2 Counter/Timer Select: C/T2# = 0 selects timer operation: timer 2 counts the divided-down system clock. C/T2# = 1 selects counter operation: timer 2 counts negative transitions on external pin T2.
0	CP/RL2#	Capture/Reload Bit: When set, captures occur on negative transitions at T2EX if EXEN2 = 1. When cleared, auto-reloads occur on timer 2 overflows or negative transitions at T2EX if EXEN2 = 1. The CP/RL2# bit is ignored and timer 2 forced to auto-reload on timer 2 overflow, if RCLK = 1 or TCLK = 1.

T2MOD

Address: S:C9H
Reset State: XXXX XX00B

Timer 2 Mode Control Register. Contains the timer 2 down count enable and clock-out enable bits for timer 2 .

7							0
—	—	—	—	—	—	T2OE	DCEN

Bit Number	Bit Mnemonic	Function
7:2	—	Reserved: The values read from these bits are indeterminate. Do not write a “1” to these bits.
1	T2OE	Timer 2 Output Enable Bit: In the timer 2 clock-out mode, connects the programmable clock output to external pin T2.
0	DCEN	Down Count Enable Bit: Configures timer 2 as an up/down counter.

TCON

Address: S:88H
Reset State: 0000 0000B

Timer/Counter Control Register. Contains the overflow and external interrupt flags and the run control and interrupt transition select bits for timer 0 and timer 1.

7							0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Bit Number	Bit Mnemonic	Function
7	TF1	Timer 1 Overflow Flag: Set by hardware when the timer 1 register overflows. Cleared by hardware when the processor vectors to the interrupt routine.
6	TR1	Timer 1 Run Control Bit: Set/cleared by software to turn timer 1 on/off.
5	TF0	Timer 0 Overflow Flag: Set by hardware when the timer 0 register overflows. Cleared by hardware when the processor vectors to the interrupt routine.
4	TR0	Timer 1 Run Control Bit: Set/cleared by software to turn timer 1 on/off.
3	IE1	Interrupt 1 Flag: Set by hardware when an external interrupt is detected on the INT1# pin. Edge- or level- triggered (see IT1). Cleared when interrupt is processed if edge-triggered.
2	IT1	Interrupt 1 Type Control Bit: Set this bit to select edge-triggered (high-to-low) for external interrupt 1. Clear this bit to select level-triggered (active low).
1	IE0	Interrupt 1 Flag: Set by hardware when an external interrupt is detected on the INT0# pin. Edge- or level- triggered (see IT0). Cleared when interrupt is processed if edge-triggered.
0	IT0	Interrupt 0 Type Control Bit: Set this bit to select edge-triggered (high-to-low) for external interrupt 0. Clear this bit to select level-triggered (active low).

TMOD

Address: S:89H
Reset State: 0000 0000B

Timer/Counter Mode Control Register. Contains mode select, run control select, and counter/timer select bits for controlling timer 0 and timer 1.

7							0
GATE1	C/T1#	M11	M01	GATE0	C/T0#	M01	M00

Bit Number	Bit Mnemonic	Function
7	GATE1	Timer 1 Gate: When GATE1 = 0, run control bit TR1 gates the input signal to the timer register. When GATE1 = 1 and TR1 = 1, external signal INT1 gates the timer input.
6	C/T1#	Timer 1 Counter/Timer Select: C/T1# = 0 selects timer operation: timer 1 counts the divided-down system clock. C/T1# = 1 selects counter operation: timer 1 counts negative transitions on external pin T1.
5, 4	M11, M01	Timer 1 Mode Select: **M11 M01** 0 0 Mode 0: 8-bit timer/counter (TH1) with 5-bit prescalar (TL1) 0 1 Mode 1: 16-bit timer/counter 1 0 Mode 2: 8-bit auto-reload timer/counter (TL1). Reloaded from TH1 at overflow. 1 1 Mode 3: Timer 1 halted. Retains count.
3	GATE0	Timer 0 Gate: When GATE0 = 0, run control bit TR0 gates the input signal to the timer register. When GATE0 = 1 and TR0 = 1, external signal INT0 gates the timer input.
2	C/T0#	Timer 0 Counter/Timer Select: C/T0# = 0 selects timer operation: timer 0 counts the divided-down system clock. C/T0# = 1 selects counter operation: timer 0 counts negative transitions on external pin T0.
1, 0	M01, M00	Timer 0 Mode Select: **M10 M00** 0 0 Mode 0: 8-bit timer/counter (T0) with 5-bit prescaler (TL0) 0 1 Mode 1: 16-bit timer/counter 1 0 Mode 2: 8-bit auto-reload timer/counter (TL0). Reloaded from TH0 at overflow 1 1 Mode 3: TL0 is 8-bit timer/counter. TH0 is 8-bit timer only using timer 1 TR1 and TF1 bits.

TH0, TL0

Address: TH0 S:8CH
TL0 S:8AH

Reset State: 0000 0000B

TH0, TL0 Timer Registers. These registers operate in cascade to form the 16-bit timer register in timer 0 or separately as 8-bit timer/counters.

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

Bit Number	**Bit Mnemonic**	**Function**
7:0	TH0.7:0 TL0.7:0	High byte of the timer 0 timer register. Low byte of the timer 0 timer register.

TH1, TL1

Address: TH1 S:8DH
TL1 S:8BH

Reset State: 0000 0000B

TH1, TL1 Timer Registers. These registers operate in cascade to form the 16-bit timer register in timer 1 or separately as 8-bit timer/counters.

7 **0**

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

Bit Number	**Bit Mnemonic**	**Function**
7:0	TH1.7:0 TL1.7:0	High byte of the timer 1 timer register. Low byte of the timer 1 timer register.

TH2, TL2

Address: TH2 S:CDH
TL2 S:CCH

Reset State: 0000 0000B

TH2, TL2 Timer Registers. These registers operate in cascade to form the 16-bit timer register in timer 2.

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

Bit Number	Bit Mnemonic	Function
7:0	TH2.7:0 TL2.7:0	High byte of the timer 2 timer register. Low byte of the timer 2 timer register.

WDTRST

Address: S:A6H
Reset State: XXXX XXXXB

Watchdog Timer Reset Register. Writing the two-byte sequence 1EH-E1H to the WDTRST register clears and enables the hardware WDT. The WDTRST register is a write-only register. Attempts to read it return FFH. The WDT itself is not read or write accessible. See “Watchdog Timer” on page 7-16.

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

Bit Number	Bit Mnemonic	Function
7:0	WDTRST.7:0	Provides user control of the hardware WDT.

Glossary

GLOSSARY

This glossary defines acronyms, abbreviations, and terms that have special meaning in this manual. (Chapter 1, “Guide to this Manual,” discusses notational conventions and general terminology.)

#0data16	A 32-bit constant that is immediately addressed in an instruction. The upper word is filled with zeros.
#1data16	A 32-bit constant that is immediately addressed in an instruction. The upper word is filled with ones.
#data	An 8-bit constant that is immediately addressed in an instruction.
#data16	A 16-bit constant that is immediately addressed in an instruction.
#short	A constant, equal to 1, 2, or 4, that is immediately addressed in an instruction.
accumulator	A register or storage location that forms the result of an arithmetic or logical operation.
addr11	An 11-bit destination address. The destination can be anywhere in the same 2-Kbyte block of memory as the first byte of the next instruction.
addr16	A 16-bit destination address. The destination can be anywhere within the same 64-Kbyte region as the first byte of the next instruction.
addr24	A 24-bit destination address. The destination can be anywhere within the 16-Mbyte address space.
ALU	Arithmetic-logic unit. The part of the CPU that processes arithmetic and logical operations.
assert	The term *assert* refers to the act of making a signal active (enabled). The polarity (high/low) is defined by the signal name. Active-low signals are designated by a pound symbol (#) suffix; active-high signals have no suffix. To *assert* RD# is to drive it low; to *assert* ALE is to drive it high.

binary-code compatibility	The ability of an MCS® 251 microcontroller to execute, without modification, binary code written for an MCS 51 microcontroller.
binary mode	An operating mode, selected by a configuration bit, that enables an MCS 251 microcontroller to execute, without modification, binary code written for an MCS 51 microcontroller.
bit	A binary digit.
bit (operand)	An addressable bit in the MCS 251 architecture.
bit51	An addressable bit in the MCS 51 architecture.
byte	Any 8-bit unit of data.
clear	The term *clear* refers to the value of a bit or the act of giving it a value. If a bit is *clear,* its value is “0”; *clearing* a bit gives it a “0” value.
code memory	See *program memory.*
configuration bytes	Bytes, residing in on-chip OTPROM/ROM, that determine a set of operating parameters for the 8XC251SB.
dir8	An 8-bit direct address. This can be a memory address or an SFR address.
dir16	A 16-bit memory address (00:0000H–00:FFFFH) used in direct addressing.
DPTR	The 16-bit data pointer. In MCS 251 microcontrollers, DPTR is the lower 16 bits of the 24-bit extended data pointer, DPX.
DPX	The 24-bit extended data pointer in MCS 251 microcontrollers. See also *DPTR*.
deassert	The term *deassert* refers to the act of making a signal inactive (disabled). The polarity (high/low) is defined by the signal name. Active-low signals are designated by a pound symbol (#) suffix; active-high signals have no suffix. To *deassert* RD# is to drive it high; to *deassert* ALE is to drive it low.

doping	The process of introducing a periodic table Group III or Group V element into a Group IV element (e.g., silicon). A Group III impurity (e.g., indium or gallium) results in a *p-type* material. A Group V impurity (e.g., arsenic or antimony) results in an *n-type* material.
double word	A 32-bit unit of data. In memory, a double word comprises four contiguous bytes.
dword	See *double word*.
edge-triggered	The mode in which a device or component recognizes a falling edge (high-to-low transition), a rising edge (low-to-high transition), or a rising or falling edge of an input signal as the assertion of that signal. See also *level-triggered*.
encryption array	An array of key bytes used to encrypt user code in the on-chip code memory as that code is read; protects against unauthorized access to user's code.
external address	A 16-bit or 17-bit address presented on the device pins. The address decoded by an external device depends on how many of these address bits the external system uses. See also *internal address*.
FET	Field-effect transistor.
idle mode	The power conservation mode that freezes the core clocks but leaves the peripheral clocks running.
input leakage	Current leakage from an input pin to power or ground.
integer	Any member of the set consisting of the positive and negative whole numbers and zero.
internal address	The 24-bit address that the device generates. See also *external address*.
interrupt handler	The module responsible for handling interrupts that are to be serviced by user-written interrupt service routines.
interrupt latency	The delay between an interrupt request and the time when the first instruction in the interrupt service routine begins execution.
interrupt response time	The time delay between an interrupt request and the resulting break in the current instruction stream.

interrupt service routine (ISR)	The software routine that services an interrupt.
level-triggered	The mode in which a device or component recognizes a high level (logic one) or a low level (logic zero) of an input signal as the assertion of that signal. See also *edge-triggered.*
LSB	Least-significant bit of a byte or least-significant byte of a word.
maskable interrupt	An interrupt that can be disabled (masked) by its individual mask bit in an interrupt enable register. All 8XC251SB interrupts, except the software trap (TRAP), are maskable.
MSB	Most-significant bit of a byte or most-significant byte of a word.
multiplexed bus	A bus on which the data is time-multiplexed with (some of) the address bits.
***n*-channel FET**	A field-effect transistor with an *n*-type conducting path (channel).
***n*-type material**	Semiconductor material with introduced impurities (*doping*) causing it to have an excess of negatively charged carriers.
nonmaskable interrupt	An interrupt that cannot be disabled (masked). The software trap (TRAP) is the 8XC251SB's only nonmaskable interrupt.
***npn* transistor**	A transistor consisting of one part *p*-type material and two parts *n*-type material.
OTPROM	One-time-programmable read-only memory, a version of EPROM.
***p*-channel FET**	A field-effect transistor with a *p*-type conducting path.
***p*-type material**	Semiconductor material with introduced impurities (*doping*) causing it to have an excess of positively charged carriers.
PC	Program counter
program memory	A part of memory where instructions can be stored for fetching and execution.

powerdown mode	The power conservation mode that freezes both the core clocks and the peripheral clocks.
PWM	Pulse-width modulated (outputs).
rel	A signed (two's complement) 8-bit, relative destination address. The destination is -128 to +127 bytes relative to the first byte of the next instruction.
reserved bits	Register bits that are not used in this device but may be used in future implementations. Avoid any software dependence on these bits. In the 8XC251SB, the value read from a reserved bit is indeterminate; do not write a “1” to a reserved bit.
set	The term *set* refers to the value of a bit or the act of giving it a value. If a bit is *set*, its value is “1”; *setting* a bit gives it a “1” value.
SFR	Special-function register.
sign extension	A method for converting data to a larger format by filling the extra bit positions with the value of the sign. This conversion preserves the positive or negative value of signed integers.
sink current	Current flowing **into** a device to ground. Always a positive value.
source-code compatibility	The ability of an MCS 251 microcontroller to execute recompiled source code written for an MCS 51 microcontroller.
source current	Current flowing **out of** a device from V_{CC}. Always a negative value.
source mode	An operating mode that is selected by a configuration bit. In source mode, an MCS 251 microcontroller can execute recompiled source code written for an MCS 51 microcontroller. In source mode, the MCS 251 microcontroller cannot execute unmodified binary code written for an MCS 51 microcontroller. See *binary mode*.
SP	Stack pointer.
SPX	Extended stack pointer.

state time (or state)	The basic time unit of the device; the combined period of the two internal timing signals, PH1 and PH2. (The internal clock generator produces PH1 and PH2 by halving the frequency of the signal on XTAL1.) With a 16-MHz crystal, one *state time* equals 125 ns. Because the device can operate at many frequencies, this manual defines time requirements in terms of *state times* rather than in specific units of time.
UART	Universal asynchronous receiver and transmitter. A part of the serial I/O port.
WDT	Watchdog timer, an internal timer that resets the device if the software fails to operate properly.
word	A 16-bit unit of data. In memory, a word comprises two contiguous bytes.
wraparound	The result of interpreting an address whose hexadecimal expression uses more bits than the number of available address lines. Wraparound ignores the upper address bits and directs access to the value expressed by the lower bits.

Index

#0data16, A-3
#1data16, A-3
#data
    definition, A-3
#data16, A-3
#short, A-3
80C251SB, 13-1
    configuration byte values, 13-9
83C251SB, 13-1
    See also ROM
87C251SB, 13-1
    See also OTPROM
8XC251SB, 2-1
    applications, 2-1
    block diagram, 2-2
    features, 2-4
    on-chip peripherals, 2-1, 2-3
8XC51FX, 2-1

A

A15:8, 6-1, B-2
    description, 12-1
A16, B-2
    configuring for, 13-6
    description, 12-1
AC flag, 4-19, 4-20
ACALL instruction, 4-16, A-24, A-26
ACC, 3-10, 3-13, 3-14, C-4
Accumulator, 3-12
    in register file, 3-10
    See also ACC
AD7:0, 6-1, B-2
    description, 12-1
ADD instruction, 4-10, A-14
ADDC instruction, 4-10, A-14
addr11, 4-14, A-3
addr16, 4-14, A-3
addr24, 4-14, A-3
Address spaces *See Memory space, SFRs, Register file, External memory, Compatibility*
Addresses
    internal vs external, 12-3
Addressing modes, 3-5, 4-5
    See also Data instructions, Bit instructions, Control instructions
AJMP instruction, 4-16, A-24
ALE, B-2
    caution, 10-6
    description, 12-1
    extended, 12-6
    following reset, 10-6
    idle mode, 11-4
    programming for extension, 13-6
    programming on-chip OTPROM, 13-3
ANL instruction, 4-11, 4-12
    for bits, A-23
ANL/ instruction, 4-12
    for bits, A-23
Arithmetic instructions, 4-10, 4-11
    table of, A-14, A-15, A-16

B

B register, 3-12, C-5
    as SFR, 3-13, 3-14
    in register file, 3-10
Base address, 4-5
Baud rate *See Serial I/O port, Timer 1, Timer 2*
Binary and source modes, 2-3, 4-1–4-3
    opcode maps, 4-1
    selection guidelines, 2-3, 4-2
Bit address
    addressing modes, 4-14
    definition, A-3
    examples, 4-13
Bit instructions, 4-4, 4-12–4-14
    addressing modes, 4-6, 4-12
bit51, 4-13, A-3
Broadcast address *See Serial I/O port*

C

Call instructions, 4-16
Capacitors
    bypass, 10-2
CCAP1H–CCAP4H, CCAP1L–CCAP4L, 3-13, 3-16, C-6
CCAPM1–4, 3-13, 3-15, C-7
    interrupts, 5-5
CCON, 3-13, 3-15, C-8
Ceramic resonator, 10-4
CEX4:0, 6-1, B-2
CH, CL, 3-13, 3-16, C-9
CJNE instruction, A-25
Clock, 2-5
    external, 10-4

  - external source, 10-3
  - idle and powerdown modes, 11-5
  - idle mode, 11-4
  - powerdown mode, 11-5, 11-6
  - sources, 10-3
- CLR instruction, 4-11, 4-12, A-17, A-23
- CMOD, 3-13, 3-15, C-10
  - interrupts, 5-5
- CMP instruction, 4-10, 4-15, A-15
- Code constants, 12-7
- Code fetches
  - external, 12-10
  - internal, 12-10
  - page hit and page miss, 12-11
  - page mode, 12-11
- Code memory
  - MCS 51 architecture, 3-5
  - *See also On-chip code memory, External code memory*
- Compatibility (MCS 251 and MCS 51 architectures), 2-1, 3-2–3-5
  - address spaces, 3-2, 3-4
  - external memory, 3-5
  - memory configuration for, 12-5
  - on-chip RAM, 3-5
  - SFR space, 3-5
  - *See also Binary and source modes*
- CONFIG0
  - bit definitions, 13-7
- CONFIG1
  - bit definitions, 13-8
- Configuration bytes
  - programming, 13-1
  - programming and verifying, 13-6
  - setup for programming and verifying, 13-2–13-3
- Control instructions, 4-4, 4-14–4-17
  - addressing modes, 4-14, 4-15
  - table of, A-24
- Core, 2-4
  - SFRs, 3-14
- CPL instruction, 4-11, 4-12, A-17, A-23
- CPU, 2-4
  - block diagram, 2-5
- Crystal
  - for on-chip oscillator, 10-3
- CY flag, 4-19, 4-20

D

- DA instruction, A-16
- Data instructions, 4-4, 4-6–4-12
  - addressing modes, 4-6
- Data pointer *See DPH, DPL, DPTR, DPX, DPXL*
- Data transfer instructions, 4-11–4-12
  - table of, A-22
  - *See also Move instructions*
- Data types, 4-4
- DEC instruction, 4-10, A-16
- Destination register, 4-5
- dir16, A-3
- dir8, A-3
- Direct addressing, 4-5
  - in control instructions, 4-14
- Displacement addressing, 4-5, 4-9
- DIV instruction, 4-10, A-16
- Division, 4-10
- DJNZ instruction, A-25
- Documents, related, 1-5
- DPH, DPL, 3-12, C-11, C-12
  - as SFRs, 3-13, 3-14
- DPTR, 3-12
  - in jump instruction, 4-14
- DPX, 3-5, 3-10, 3-12, 4-7
- DPXL, 3-12, C-13
  - as SFR, 3-13, 3-14
  - external data memory mapping, 3-5, 4-7, 4-11
  - reset value, 3-5

E

- EA#, 3-6, B-2
  - description, 12-1
- ECALL instruction, 4-16, A-24
- ECI, 6-1, B-2
- EJMP instruction, 4-16, A-24
- EMAP bit, 3-6, 12-7
- Encryption, 13-1
- Encryption array
  - key bytes, 13-10
  - programming and verifying, 13-1, 13-10
  - setup for programming, 13-2–13-3
- ERET instruction, 4-17, A-24
- Escape prefix (A5H), 4-2
- Extended stack pointer *See SPX*
- External address lines

  number of, 12-3
  See also External bus
External bus
  AC timing definitions, 12-28
  AC timing specifications, 12-24–12-27
  bus-idle condition, 12-7
  inactive, 12-7
  pin status, 12-15, 12-16
  structure in page mode, nonpage mode, 12-10
External bus cycles, 12-7
  definitions, 12-8
  extended ALE wait state, 12-14
  extended PSEN#/RD#/WR# wait state, 12-13
  nonpage mode, 12-8, 12-9
  page hit vs page miss, 12-10
  page mode, 12-10–12-12
External code memory, 12-4, 12-5
  example, 12-16, 12-21, 12-22
  idle mode, 11-4
  powerdown mode, 11-5
External memory, 3-8
  design examples, 12-16–12-24
  MCS 51 architecture, 3-3, 3-4, 3-5
External memory interface, 12-1–12-30
  configuring, 12-2–12-7
  signals, 12-1
External RAM, 12-4, 12-5
  example, 12-16, 12-19, 12-21, 12-22
  exiting idle mode, 11-5

F

F0 flag, 4-19
Flash memory
  example, 12-21, 12-22

G

Given address *See Serial I/O port*
Ground bounce, 10-2

H

Hardware
  application notes, 1-6

I

I/O ports, 6-1–6-8
  external memory access, 6-7, 6-8
  latches, 6-2
  loading, 6-7
  pullups, 6-6
  quasi-bidirectional, 6-5
  SFRs, 3-14
  See also Ports 0–3
Idle mode, 2-3, 11-1, 11-4–11-5
  entering, 11-4
  exiting, 10-5, 11-5
  external bus, 12-7
IE, 5-3, 5-5
IE0, 3-13, 3-14, 5-14, 9-11, C-14
Immediate addressing, 4-5
INC instruction, 4-10, A-16
Indirect addressing, 4-5
  in control instructions, 4-14
  in data instructions, 4-9
Input pins
  level-sensitive, B-2
  sampled, B-2
INT1:0#, 5-1, 6-1, 7-1, 7-3, B-2
  pulse width measurements, 7-10
Interrupt request, 5-1
  cleared by hardware, 5-4
Interrupt service routine
  exiting idle mode, 11-5
  exiting powerdown mode, 11-6
Interrupts, 5-1–5-15
  blocking conditions, 5-14
  detection, 5-3
  edge-triggered, 5-4
  enable/disable, 5-5
  exiting idle mode, 11-5
  exiting powerdown mode, 11-6
  external, 5-3, 5-11
  global enable, 5-5
  instruction completion time, 5-10
  latency, 5-9–5-13
  level-triggered, 5-4
  PCA, 5-5
  polling, 5-9, 5-10
  priority, 5-1, 5-3, 5-4, 5-6–5-8
  priority within level, 5-7
  processing, 5-9–5-15
  request *See Interrupt request*
  response time, 5-9, 5-10
  sampling, 5-3, 5-10
  serial port, 5-5

  service routine (ISR), 5-4, 5-9, 5-14, 5-15
  sources, 5-3
  timer/counters, 5-4
  vector cycle, 5-14
  vectors, 3-5, 5-4
INTR bit
  and RETI instruction, 4-17
IPH0, 3-13, 3-14, 5-3, 5-6, 5-14, C-15
  bit definitions, 5-7
IPL0, 3-13, 3-14, 5-3, 5-6, 5-14, C-16
  bit definitions, 5-7
ISR *See Interrupts, service routine*

J

JB instruction, 4-15, A-24
JBC instruction, 4-15, A-24
JC instruction, A-24
JE instruction, A-24
JG instruction, A-24
JLE instruction, A-24
JMP instruction, A-24
JNB instruction, 4-15, A-24
JNC instruction, A-24
JNE instruction, A-24
JNZ instruction, A-24
JSG instruction, A-25
JSGE instruction, A-25
JSL instruction, A-25
JSLE instruction, A-25
Jump instructions
  bit-conditional, 4-15
  compare-conditional, 4-15, 4-16
  unconditional, 4-16
JZ instruction, A-24

K

Key bytes *See Encryption array*

L

LCALL instruction, 4-16, A-24
Level-sensitive input, B-2
LJMP instruction, 4-16, A-24
Lock bits
  programming and verifying, 13-1, 13-9
  protection types, 13-9
  setup for programming and verifying, 13-2–13-3
Logical instructions, 4-11
  table of, A-17

M

MCS 251 microcontroller, 2-1
  features, 2-1
MCS 51 microcontroller, 2-1
Memory space, 2-3, 3-1, 3-5–3-8
  compatibility *See Compatibility (MCS 251 and MCS 51 architectures)*
  hardware implementation, 3-5
  internal vs external, 12-4–12-6
  regions, 3-2, 3-5
  reserved locations, 3-5
Miller effect, 10-4
MOV instruction, 4-11, A-19, A-20, A-21
  for bits, 4-12, A-23
MOVC instruction, 3-3, 4-11, A-21
Move instructions
  table of, A-19
MOVH instruction, 4-12, A-21
MOVS instruction, 4-12, A-21
MOVX instruction, 3-3, 4-11, A-21
MOVZ instruction, 4-12, A-21
MUL instruction, 4-10
Multiplication, 4-10

N

N flag, 4-11, 4-20
Noise reduction, 10-2, 10-3, 10-4
Nonpage mode
  bus cycles *See External bus cycles, Nonpage mode*
  bus structure, 12-1
  configuring for, 12-3
  design example, 12-16, 12-19
  port pin status, 12-15
Nonvolatile memory
  programming and verifying, 13-1–13-12
  See also On-chip code memory, Configuration bytes, Lock bits, Encryption array, Signature bytes
NOP instruction, 4-16, A-25

O

ONCE mode, 11-1, 11-7
  entering, 11-7

- exiting, 11-7

On-chip code memory, 3-2, 12-4, 12-13
- accessing in data memory, 12-7
- accessing in region 00:, 3-6
- idle mode, 11-4
- powerdown mode, 11-5
- programming and verifying, 13-1, 13-5
- remapping, 13-6
- setup for programming and verifying, 13-2–13-3
- starting address, 3-6, 13-1, 13-2
- top eight bytes, 3-6, 13-2
- *See also OTPROM, ROM*

On-chip oscillator
- hardware setup, 10-1

On-chip RAM, 3-2, 3-6
- bit addressable, 3-6, 4-13
- bit addressable in MCS 51 architecture, 4-13
- idle mode, 11-4
- MCS 51 architecture, 3-2, 3-4
- reset, 10-6

Opcodes
- for binary and source modes, 4-1
- map, A-4
- *See also Binary and source modes*

ORL instruction, 4-11, 4-12
- for bits, A-23

ORL/ instruction, 4-12
- for bits, A-23

Oscillator, 2-5
- at startup, 10-7
- during reset, 10-5
- ONCE mode, 11-7
- on-chip, 10-3
- powerdown mode, 11-5, 11-6
- programming and verifying on-chip OTPROM/ROM, 13-3

OTPROM (on-chip), 13-1
- programming algorithm, 13-4
- programming and verifying, 13-1–13-12
- programming waveforms, 13-4
- timing for programming and verifying, 13-11
- verify algorithm, 13-5
- *See also On-chip code memory, Configuration bytes, Lock bits, Encryption array, Signature bytes*

OV bit, 4-19, 4-20

Overflow *See OV bit*

P

P bit, 4-19

P0, 3-13, 3-14, 6-2, C-17

P1, 3-13, 3-14, 6-2, C-18

P2, 3-13, 3-14, 6-2, C-19

P3, 3-13, 3-14, 6-2, C-20

PAGE bit, 12-3

Page mode, 2-4
- address access time, 12-11
- bus cycles *See External bus cycles, page mode*
- configuring for, 12-3, 13-6
- design example, 12-21, 12-22
- port pin status, 12-16

Parity *See P bit*

PCA
- idle mode, 11-4
- SFRs, 3-15

PCON, 3-13, 3-14, 9-7, 11-1, 11-2, 11-5, C-21
- idle mode, 11-4
- powerdown mode, 11-6
- reset, 10-5

Peripheral cycle, 2-5

Phase 1 and phase 2, 2-5

Pin conditions, 11-3

Pins
- unused inputs, 10-2

Pipeline, 2-4

POP instruction, 3-11, 4-12, A-22

Port 0, 6-2, B-3
- and top of on-chip code memory, 13-2
- pullups, 6-7
- structure, 6-3
- *See also External bus*

Port 1, 6-2, B-3
- structure, 6-3

Port 2, 6-2, B-3
- and top of on-chip code memory, 13-2
- structure, 6-4
- *See also External bus*

Port 3, 6-2, B-3
- structure, 6-3

Ports
- at power on, 10-7
- exiting idle mode, 11-5
- exiting powerdown mode, 11-5
- extended execution times, 4-1, A-1, A-11

- programming and verifying on-chip OTPROM/ROM, 13-3, 13-4, 13-5

Power supply, 10-2
Powerdown mode, 2-3, 11-1, 11-5–11-6
- accidental entry, 11-4
- entering, 11-6
- exiting, 10-5, 11-6
- external bus, 12-7

PROG#, 13-1, B-3
Program status word *See PSW, PSW1*
PSEN#, 13-6, B-3
- caution, 10-6
- description, 12-2
- idle mode, 11-4
- programming on-chip OTPROM, 13-3
- regions for strobe, 12-3

PSW, A-26
PSW, PSW1, 3-13, 3-14, 4-17–4-18, C-22, C-23
- conditional jumps, 4-15
- effects of instructions on flags, 4-18

PSW1, A-26
Pullups, 6-7
- ports 1, 2, 3, 6-5

Pulse width measurements, 7-10
PUSH instruction, 3-11, 4-12, A-22

Q

Quick-pulse algorithm, 13-1

R

RCAP2H, RCAP2L, 3-13, 3-15, 7-2, 9-12, C-24
RD#, 6-1, 13-6, B-3
- as 17th address bit, 12-3, 12-4
- described, 12-2
- regions for strobe, 12-3

RD1:0 configuration bits, 12-3–12-6
- table, 12-3

Read-modify-write instructions, 6-2, 6-5
Register addressing, 4-5, 4-8
Register banks, 3-2, 3-8
- accessing in memory address space, 4-6
- implementation, 3-8, 3-9
- MCS 51 architecture, 3-2
- selection bits (RS1:0), 4-19, 4-20

Register file, 2-4, 3-1, 3-5, 3-8–3-12
- address space, 3-2
- addressing locations in, 3-9
- and reset, 10-6
- MCS 51 architecture, 3-4
- naming registers, 3-8
- register types, 3-8

Registers *See Register addressing, Register banks, Register file*
rel, A-3
Relative addressing, 4-6, 4-14
Reset, 10-5–10-7
- cold start, 10-5, 11-1
- entering ONCE mode, 11-7
- exiting idle mode, 11-5
- exiting powerdown mode, 11-6
- externally initiated, 10-5
- need for, 10-6
- operation, 10-6
- power on, 10-6
- power-on setup, 10-1
- timing sequence, 10-6, 10-7
- warm start, 10-5, 11-1

RET instruction, 4-17, A-24
RETI instruction, 5-1, 5-14, 5-15, A-24
Return instructions, 4-16
RL instruction, A-17
RLC instruction, A-17
ROM (on-chip), 13-1
- verifying, 13-1–13-12
- *See also On-chip code memory, Configuration bytes, Lock bits, Encryption array, Signature bytes*

Rotate instructions, 4-11
RR instruction, A-17
RRC instruction, A-17
RST, 10-5, 10-6, B-3
- exiting idle mode, 11-5
- exiting powerdown mode, 11-6
- ONCE mode, 11-7
- power-on reset, 10-6
- programming and verifying on-chip OTPROM/ROM, 13-3

RXD, 6-1, 9-1, B-3
- mode 0, 9-4
- modes 1, 2, 3, 9-6

S

SADDR, 3-13, 3-15, 9-2, 9-8, 9-9, 9-10, C-25
SADEN, 3-13, 3-15, 9-2, 9-8, 9-9, 9-10, C-26

Sampled input, B-2
SBUF, 3-13, 3-15, 9-2, 9-4, 9-5, C-27
SCON, 3-13, 3-15, 9-2, 9-4, 9-5, 9-6, 9-7, C-28, C-29
    bit definitions, 9-3
    interrupts, 5-5
Security, 13-1
Serial I/O port, 9-1–9-14
    asynchronous modes, 9-6
    automatic address recognition, 9-7–9-10
    baud rate generator, 7-9
    baud rate, mode 0, 9-4, 9-10
    baud rate, modes 1, 2, 3, 9-6, 9-10–9-14
    broadcast address, 9-9
    data frame, modes 1, 2, 3, 9-6
    framing bit error detection, 9-7
    full-duplex, 9-6
    given address, 9-8
    half-duplex, 9-4
    interrupts, 9-1, 9-8
    mode 0, 9-4–9-5
    modes 1, 2, 3, 9-6
    multiprocessor communication, 9-7
    SFRs, 3-15, 9-1, 9-2
    synchronous mode, 9-4
    timer 1 baud rate, 9-11, 9-12
    timer 2 baud rate, 9-12–9-14
    timing, mode 0, 9-5
SETB instruction, 4-12, A-23
SFRs
    accessing, 3-12
    address space, 3-1, 3-2
    idle mode, 11-4
    map, 3-13
    MCS 51 architecture, 3-4
    powerdown mode, 11-5
    reset initialization, 10-6
    reset values, 3-12
    tables of, 3-14
    unimplemented, 3-2, 3-12
Shift instruction, 4-11
Signal descriptions, 8-4
Signature bytes
    setup for verifying, 13-2–13-3
    values, 13-10
    verifying, 13-1, 13-10
SJMP instruction, 4-16, A-24
SLL instruction, 4-11, A-17
Software
    application notes, 1-6
Source register, 4-5
SP, 3-11, 3-12, 3-13, 3-14, C-30
Special function registers *See SFRs*
SPH, 3-11, 3-12, 3-13, 3-14, C-31
SPX, 3-10, 3-11, 3-12
SRA instruction, 4-11, A-18
SRL instruction, 4-11, A-18
State time, 2-5
SUB instruction, 4-10, A-14
SUBB instruction, 4-10, A-14
SWAP instruction, 4-11, A-18

T

T1:0, 6-1, 7-3, B-3
T2, 6-1, 7-3, B-4
T2CON, 3-13, 3-15, 7-1, 7-2, 7-10, 9-13, C-32
    baud rate generator, 9-12
    bit definitions, 7-17
T2EX, 6-1, 7-3, 7-11, 9-12, B-4
T2MOD, 3-13, 3-15, 7-1, 7-2, 7-10, C-33
    bit definitions, 7-16
Target address, 4-6
TCON, 3-13, 3-15, 7-1, 7-2, 7-4, 7-6, C-34
    bit definitions, 7-8
    interrupts, 5-1
TH2, TL2
    baud rate generator, 9-14
    baud-rate generator, 9-12
THx, TLx (x = 0, 1, 2), 3-13, 3-15, 7-2, C-36, C-37, C-38
Timer 0, 7-4–7-8
    applications, 7-9
    auto-reload, 7-5
    counter/timer select, 7-7
    interrupt, 7-4, 7-8
    mode 0, 7-4
    mode 1, 7-5
    mode 2, 7-5
    mode 3, 7-5
    mode selection, 7-7
    pulse width measurements, 7-10
Timer 1
    applications, 7-9
    auto-reload, 7-9
    baud rate generator, 7-6

- counter/timer select, 7-7
- interrupt, 7-6, 7-8
- mode 0, 7-6
- mode 1, 7-9
- mode 2, 7-9
- mode 3, 7-9
- mode selection, 7-7
- pulse width measurements, 7-10

Timer 2, 7-10–7-17
- auto-reload mode, 7-12
- baud rate generator, 7-14
- capture mode, 7-11
- clock out mode, 7-14
- interrupt, 7-11
- mode select, 7-15

Timer/counters, 7-1–7-17
- external input sampling, 7-3
- internal clock, 7-3
- interrupts, 7-1
- overview, 7-1–7-3
- registers, 7-2
- SFRs, 3-15
- signal descriptions, 7-3
- *See also Timer 0, Timer 1, Timer 2*

Timing
- symbol definitions, 12-28

TMOD, 3-13, 3-15, 7-1, 7-2, 7-4, 7-6, 9-11, C-35
- bit definitions, 7-7

Tosc, 2-5, 2-6
- *See also Oscillator*

TRAP instruction, 4-17, 5-3, 5-5, 5-15, A-25

TXD, 6-1, 9-1, B-4
- mode 0, 9-4
- modes 1, 2, 3, 9-6

U

UART, 9-1

UD flag, 4-19

V

Vcc, 10-2, B-4
- during reset, 10-5
- power off flag, 11-1
- powerdown mode, 11-5, 11-6
- power-on reset, 10-7
- *See also Power supply*

Vcc2, 10-2, B-4

Vpp, 13-1, B-4
- requirements, 13-3

Vss, B-4

Vss1, 10-2, B-4

Vss2, 10-2, B-4

W

Wait state, 12-6
- configuring for, 13-6
- extended ALE, 12-6
- PSEN#/RD#/WR#, 4-1, 12-6, A-1, A-11

Watchdog timer
- SFRs, 3-15

Watchdog timer (hardware), 7-16–7-18
- enabling, disabling, 7-16
- in idle mode, 7-18
- in powerdown mode, 7-18
- overflow, 7-16

WDT
- initiating reset, 10-5

WDTRST, 3-13, 3-15, 7-2, 7-16, C-39

WR#, 6-1, B-4
- described, 12-2

WSA, WSB bits, 12-6

X

XALE bit, 12-6

XCH instruction, 4-12, A-22

XCHD instruction, 4-12, A-22

XRL instruction, 4-11

XTAL1, B-4

XTAL1, XTAL2, 10-3
- capacitance loading, 10-4

XTAL2, B-4

Z

Z flag, 4-11, 4-20