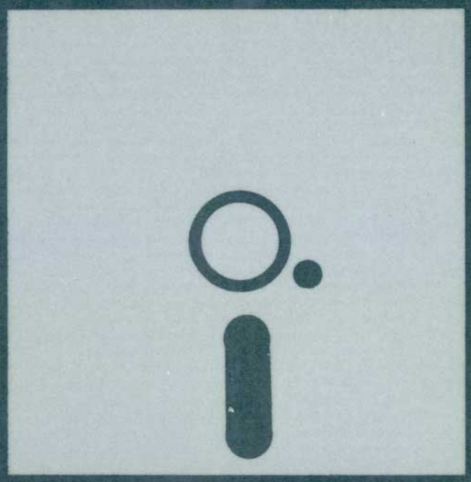
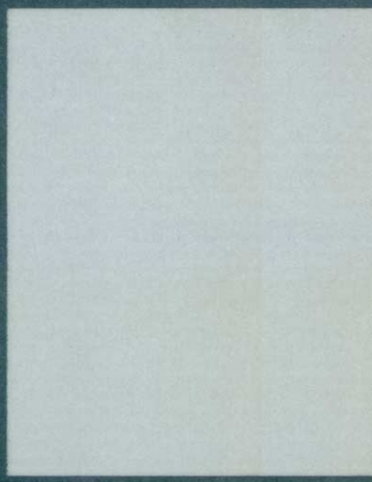
INTRODUCTION TO THE Z80 MICROCOMPUTER

ADI J. KHAMBATA





2e

INTRODUCTION TO THE Z80 MICROCOMPUTER

Second Edition

ADI J. KHAMBATA

St. Paul Technical-Vocational Institute
St. Paul, Minnesota

JOHN WILEY & SONS

New York Chichester Brisbane Toronto Singapore

Copyright © 1987, 1982 by John Wiley & Sons, Inc.

All rights reserved.

Reproduction or translation of any part of this work beyond that permitted by Section 107 or 108 of the 1976 United States Copyright Act without the permission of the copyright owner is unlawful. Requests for permission or further information should be addressed to the Permissions Department, John Wiley & Sons, Inc.

ISBN 0-471-85287-2
Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

To my wife Ruth and daughter Pixie

PREFACE

This book is intended as a supplement to the textbook *Microprocessors/Microcomputers: Architecture, Software and Systems,* 2nd edition (written by this author and published by John Wiley & Sons, Inc., New York, 1987). It may also be used as a supplement for other basic texts or as a brief stand-alone introduction to the Z80.

During the planning and writing of *Microprocessors/Microcomputers,* we encountered the following problem. In addition to offering theoretical descriptions and discussions, we wanted to describe at least one real, existing, 8-bit microcomputer system that is widely used in education institutions. However, including a description of any one product in that text would immediately have dated the publication and made it unacceptable to other users who might be committed to another product. This dilemma was resolved by adopting a unique approach. We selected three most popular and widely used 8-bit microcomputer products and present them in three separate, softcover supplements. The three products that are included in this series are the Zilog Z80, the Intel 8080A, and hte Intel 8085A. This approach allows users of the main textbook to purchase only the supplements that are of direct use for their specific needs. In addition to the paperback supplements, separate laboratory manuals—on some of the most popular microcomputers—are also planned.

This book is based primarily on the microprocessor/microcomputer courses that I have designed and taught at St. Paul Technical-Vocational Institute, St. Paul, Minnesota, and in local industries over the past several years. As a result of this experience, I have introduced a number of features in this book. First, each chapter starts with a brief listing of chapter objectives in order to give students a quick overview of the specific items that will be covered in each chapter. On completing a study of the chapter, students will be able to review the objectives, thereby (it is hoped) gaining further understanding of the material covered in that chapter.

Second, since this book may be used as a companion to my hardcover text (2nd edition), references to that book are included for the principal topics covered in each chapter. The references are made by both chapters and sections within the chapters. I feel that this simple convenience will save students a lot of time and effort when they wish to refer to the main textbook for a quick review of any particular topic.

Third, at the end of each chapter, numerous review questions are included. To aid both students and instructors, the questions are structured in formats such as true/false, multiple-choice, and fill-in. For most questions several possible answers are provided in parentheses at the end of each question. Students have only to select the appropriate answer. However, selection of the correct answer requires an understanding of the subject matter of the chapter. Furthermore, the earlier questions in any chapter are simple, but they become progressively more complex and more challenging.

Finally, two distinct styles of presentation are introduced in each chapter, namely, a list style and a descriptive prose style. The descriptive prose style is certainly easier to read and, in many cases, easier to understand, since more explanation is provided. However, classroom experience has indicated to me that the prose style does very little to aid students in remembering specific technical points or events. For instance, if a certain operation involves six tasks to be performed in a specified sequence, it is much easier for students to remember an enumeration of those tasks in six discrete steps than to remember them from a prose description.

The Z80 CPU chip and its principal support chips are covered in this book. Additionally, chapters are included for initializing, or programming, three of the most commonly used support chips—the PIO, the CTC, and the DMA.

Traditionally, a brief synopsis of each chapter is included in the preface. I find this somewhat redundant since a very detailed table of contents is included in the book. However, it is appropriate to point out some of the new features in this edition that were not present in the first edition. In the instruction set tables the number of bytes in each instruction is deleted, since it is apparent from each instruction how many bytes are involved. In its place the number of M cycles and the number of T states for each instruction are presented.

A completely new chapter on the DART (dual asynchronous receiver/transmitter) chip (Chapter 13) is included. This gives the instructor a three-way choice of including either Chapter 12 (the SIO chip) or Chapter 13 (the DART chip), or both, in a particular course. It is quite possible that the limited number of hours available for the course may restrict teaching both the SIO and the DART chips.

Two appendices are also included in this edition. Timing Diagram Conventions are shown in Appendix A and the timing diagrams for all the chips in this edition are modified to conform to these widely used conventions. The Z80 assembly language codes by mnemonics are

given in Appendix B. For each set of codes the corresponding Instruction number in the table (where the particular instruction is described) is also given. I find that inclusion of this appendix saves students a lot of time in looking up and constructing each individual instruction, particularly in the programming part of the course.

I would like to thank the many people who helped me during the writing and publication of this book. At Wiley, Hank Stewart, editor, and Joe Keenan, administrative assistant, were very helpful during the entire writing and publication process. Dr. George Richter, Technical Division manager at St. Paul Technical-Vocational Institute, constantly encouraged and supported me. Several of my students (past and present) and former engineering colleagues at Sperry Univac reviewed parts of the manuscript and offered many valuable comments and suggestions. Special contributions were made by Frank Kline, Jr., Kenneth Jarosch, Steve O'Gara, Mark Dreyer, and Gary Dunning.

As usual, this book is a Khambata family project. My wife Ruth and daughter Pixie assisted in the preparation of the index; my son Jim and his wife Shelly assisted in the corrections and proofreading; and all diagrams and timing charts were drawn by my son Danny and his wife Renee. I thank all of them.

Adi J. Khambata

St. Paul, Minnesota
November 12, 1986

ACKNOWLEDGMENTS

The author gratefully acknowledges and thanks Zilog, Inc., Cupertino, California, for granting permission to use and reproduce in this book block diagrams and timing charts, as well as the instruction set, of the Z80 Microcomputer from ZILOG published Technical Manuals. These manuals and the material used from them are identified in the following list.

Since this product supplement is primarily intended for student use in the classroom, minor modifications and additions are made in the block diagrams and timing charts to conform to explanatory material in the text and to aid the student in learning and understanding the subject matter.

ZILOG and Z80 (whichever applicable) are/is a trademark(s) of Zilog, Inc., with whom the author or publisher is not connected.

Reproduced by permission © 1977 & 1981 by Zilog, Inc. This material shall not be reproduced without written consent of Zilog, Inc.

From the Z80 CPU & Z80A-CPU TECHNICAL MANUAL © 1977

Page 4	Fig. 2.0-2	Z80 CPU Register Configuration
Page 3	Fig. 2.0-1	Z80 CPU Block Diagram
Page 39		Flag Register Format
Page 44	Table 6.0-1	Summary of Flag Operation (Note: Some of the notation symbols have been changed to accommodate the types available on my typewriter.)
Pages 43–54	Tables 7.0-1 thru 7.0-11	Summary of OP codes and execution times (Note: Flag symbols have been changed to accommodate available types on my typewriter. The columns for M cycles and number of T cycles are not included. Finally word descriptions of each word are added.)
Page 11	Fig. 4.0-0	Basic CPU Timing example.
Page 12	Fig. 4.0-1	Instruction OP code Fetch
Page 13	Fig. 4.0-1A	Instruction OP code Fetch with WAIT states
Page 13	Fig. 4.0-2	Memory Read or Write Cycles
Page 14	Fig. 4.0-2A	Memory Read or Write Cycles with WAIT states
Page 15	Fig. 4.0-3	Input or Output Cycles
Page 15	Fig. 4.0-3A	Input or Output Cycles with WAIT states
Page 16	Fig. 4.0-4	Bus Request/Acknowledge Cycle
Page 16	Fig. 4.0-5	Interrupt Request/Acknowledge Cycle
Page 18	Fig. 4.0-6	Nonmaskable Interrupt Request/Operation
Page 18	Fig. 4.0-7	Halt Exit

From the Z80-PIO & Z80A-PIO TECHNICAL MANUAL © 1977

Page 3	Fig. 2.0-1	PIO Block Diagram
Page 3	Fig. 2.0-2	Port I/O Block Diagram
Page 13	Fig. 5.0-1	Mode 0 (Output) Timing
Page 13	Fig. 5.0-2	Mode 1 (Input) Timing
Page 14	Fig. 5.0-3	Port A, Mode 2 (Bidirectional) Timing
Page 14	Fig. 5.0-4	Mode 3, Bit Control Mode Timing
Page 15	Fig. 6.0-1	Interrupt Acknowledge Timing
Page 17	Fig.7.0-1	A method of extending the interrupt priority daisy chain

From the Z80-CTC & Z80A-CTC TECHNICAL MANUAL © 1977

Page 2	Fig. 2.0-1	CTC Block Diagram
Page 3	Fig. 2.0-2	Channel Block Diagram

Page 16		CTC Write Cycle
Page 17		CTC Read Cycle
Page 18		CTC Counting and Timing Diagram
Page 19		Interrupt Acknowledge Timing Diagram
Page 20		Return from Interrupt Timing Diagram

From the Z80-SIO TECHNICAL MANUAL © 1977

Page 1		Z80-SIO Block Diagram
Page 6	Fig. 4	Transmit and Receive Data Path
Page 13	Fig. 7	Synchronous Formats
Page 21	Fig. 8	Transmit/Receive SDLC/HDLC Message Format
Page 41	Fig. 14d	Return from Interrupt Cycle
Page 41	Fig. 14c	Interrupt Acknowledge Cycle
Page 41	Fig. 14b	Write Cycle
Page 41	Fig. 14a	Read Cycle

From the Z80-DMA TECHNICAL MANUAL (January 1981) © 1981

Page 2–3	Fig. 5	Basic Functions of Z80 DMA
Page 8-7	Fig. 62	Variable-Cycle and Edge Timing
Page 8-7	Fig. 63	WAIT line Sampling in Variable-Cycle Timing
Page 8-1	Fig. 48	CPU-to-DMA Write Cycle Requirements
Page 8-1	Fig. 49	CPU-to-DMA Read Cycle Requirements
Page 8-2	Fig. 50	Sequential Memory-to-I-O Transfer Standard Timing
Page 8-2	Fig. 51	Sequential I/O-to-Memory Transfer Standard Timing
Page 8-4	Fig. 54	Bus Request and Acceptance Timing
Page 8-4	Fig. 56	Bus Release on End-of-Block (Burst & Continuous Modes)
Page 8-4	Fig. 58	Bus Release on Not-Ready (Burst Mode)
Page 8-4	Fig. 57	Bus Release on Match (Burst and Continuous Modes)
Page 8-4	Fig. 55	Bus Release in Byte Mode

CONTENTS

1 THE Z80 CPU ARCHITECTURE

CHAPTER OBJECTIVES *1*
TEXTBOOK REFERENCES *1*
1-1 INTRODUCTION *1*
1-2 THE Z80 SYSTEM *1*
1-2.1 Principal Hardware Components *1*
1-2.2 The Development System *2*
1-3 PRINCIPAL FEATURES OF THE Z80 CPU *2*
1-4 THE CPU ARCHITECTURE *2*
1-4.1.1 *GP Registers, Accumulators and Status Flag Registers* *3*
1-4.1.2 *Special-Purpose Registers* *3*
1-4.2 The Arithmetic Logic Unit (ALU) *4*
1-4.3 Instruction Register, Decode, and Control *4*
1-4.4 Program Status Word (F and F') *5*
1-5 ADDRESSING MODES *6*
1-5.1 Implied Addressing *6*
1-5.2 Register Addressing *6*
1-5.3 Register Indirect Addressing *6*
1-5.4 Bit Addressing *6*
1-5.5 Immediate Addressing *6*
1-5.6 Immediate Extended Addressing *6*
1-5.7 Extended Addressing (Direct Addressing Mode) *6*
1-5.8 Indexed Addressing *6*
1-5.9 Relative Addressing *6*
1-5.10 Modified Page Zero Addressing *7*
1-6 PACKAGE PIN ASSIGNMENTS AND FUNCTIONS *7*
1-7 REVIEW QUESTIONS *11*

2 THE Z80 INSTRUCTION SET (PART ONE)

CHAPTER OBJECTIVES *15*
TEXTBOOK REFERENCES *15*
2-1 INTRODUCTION *15*
2-2 CATEGORIES OF INSTRUCTIONS *16*
2-2.1 The Load and Exchange Group *16*
2-2.2 The Block Transfer and Search Group *16*
2-2.3 The Arithmetic and Logic Group *16*
2-2.4 The Rotate and Shift Group *16*
2-2.5 The Bit Manipulation Group (Set, Reset, and Test) *16*
2-2.6 The JUMP, CALL, and RETURN Group *16*

2-2.7 The Input/Output Group *16*
2-2.8 The Basic CPU Instructions Group *16*
2-3 INSTRUCTION FORMAT AND SYMBOLS *16*
2-3.1 The Generalized Instruction Format *16*
2-3.2 The OP Code *17*
2-3.3 The Mnemonic Structure *17*
2-3.4 Symbolic Expressions of Operations and Functions *19*
2-3.5 The Status Register Symbols *19*
2-3.6 The Register Pair Codes *20*
2-4 THE INSTRUCTION SET *20*
2-5 CONSTRUCTING COMPLEX INSTRUCTION CODES *34*
2-6 REVIEW QUESTIONS *35*

3 THE Z80 INSTRUCTION SET (PART TWO)
CHAPTER OBJECTIVES *37*
TEXTBOOK REFERENCES *37*
3-1 INTRODUCTION *37*
3-2 SHIFT OPERATIONS *37*
3-2.1 Arithmetic Shifts *37*
3-2.1.1 *Shift Left Arithmetic (SLA)* *37*
3-2.1.2 *Shift Right Arithmetic (SRA)* *38*
3-2.1.3 *Shift Right Logic (SRL)* *38*
3-2.2 Register/Memory Symbols for Rotate/Shift *38*
3-2.3 The Mnemonic Convention *38*
3-2.4 OP Code Construction *39*
3-3 ROTATE AND SHIFT GROUP *40*
3-4 BIT MANIPULATION GROUP *40*
3-5 THE JUMP INSTRUCTIONS GROUP *40*
3-6 THE CALL AND RETURN GROUP *40*
3-7 REVIEW QUESTIONS *49*

4 THE Z80 INSTRUCTION SET AND INTERRUPT RESPONSE
CHAPTER OBJECTIVES *51*
TEXTBOOK REFERENCES *51*
4-1 INTRODUCTION *51*
4-2 THE I/O INSTRUCTION GROUP *51*
4-3 THE BASIC CPU INSTRUCTION GROUP *57*
4-4 THE INTERRUPT SYSTEM *57*
4-4.1 The Objectives *57*
4-4.2 The Nonmaskable Interrupt (NMI) *57*
4-4.3 The Maskable Interrupt (INT) *57*
4-4.4 The INT Enable-Disable Flip-Flop *57*
4-4.5 CPU Response to Nonmaskable Interrupts (NMI) *57*
4-4.6 CPU Response to Maskable Interrupts (INT) *60*
4-4.6.1 *Mode 0* *60*
4-4.6.2 *Mode 1* *60*
4-4.6.3 *Mode 2* *60*
4-5 REVIEW QUESTIONS *61*

5 THE Z80 CPU TIMING AND CONTROL

CHAPTER OBJECTIVES *63*
TEXTBOOK REFERENCES *63*
5-1 THE FUNDAMENTAL INSTRUCTION CYCLE *63*
5-2 THE OP CODE FETCH CYCLE *64*
5-2.1 Without WAIT States *64*
5-2.2 With WAIT States *65*
5-3 MEMORY READ AND WRITE CYCLES *65*
5-3.1 Without WAIT States *65*
5-3.2 With WAIT States *66*
5-4 I/O READ AND WRITE CYCLES *66*
5-4.1 Without WAIT States *66*
5-4.2 With WAIT States *68*
5-5 BUS REQUEST/ACKNOWLEDGE CYCLES *69*
5-6 MASKABLE INTERRUPT REQUEST/ACKNOWLEDGE CYCLE *69*
5-7 NONMASKABLE INTERRUPT REQUEST/ACKNOWLEDGE CYCLE *70*
5-8 EXIT FROM HALT INSTRUCTION CYCLE *71*
5-9 REVIEW QUESTIONS *71*

6 Z80 PARALLEL INPUT/OUTPUT CHIP

CHAPTER OBJECTIVES *74*
TEXTBOOK REFERENCES *74*
6-1 INTRODUCTION *74*
6-2 PRINCIPAL FEATURES OF THE PIO *74*
6-3 THE PIO ARCHITECTURE *75*
6-3.1 The Generalized PIO Block Diagram *75*
6-3.2 The Port Logic Block Diagram *75*
6-3.3 The Interrupt Control Protocol *76*
6-4 PACKAGE PIN ASSIGNMENTS AND FUNCTIONS *77*
6-5 THE PIO INTERRUPT TIMING *77*
6-5.1 Mode 0--Output Byte Mode *77*
6-5.2 Mode 1—Input Byte Mode *80*
6-5.3 Mode 2—Bidirectional Mode *81*
6-5.4 Mode 3—Bit Control Mode *81*
6-5.5 Interrupt Acknowledge and Servicing *82*
6-5.6 Extending the Daisy Chain *83*
6-6 REVIEW QUESTIONS *84*

7 PROGRAMMING THE PIO CHIP

CHAPTER OBJECTIVES *86*
TEXTBOOK REFERENCES *86*
7-1 INTRODUCTION *86*
7-2 THE PROGRAMMING WORDS *87*
7-2.1 Word for Mode Set *87*
7-2.2 The Interrupt Vector *87*
7-2.3 Set Interrupt Control Byte *87*
7-2.4 Interrupt Enable Flip-Flop Set *88*
7-3 PROGRAMMING THE PIO FOR A CONTROL APPLICATION *88*
7-4 PROGRAMMING THE PIO FOR A BYTE TRANSFER APPLICATION *89*
7-5 REVIEW QUESTIONS *91*

8 Z80 COUNTER/TIMER CIRCUIT CHIP

CHAPTER OBJECTIVES *93*
TEXTBOOK REFERENCES *93*
8-1 INTRODUCTION *93*
8-2 PRINCIPAL FEATURES OF THE CTC *94*
8-3 THE CTC ARCHITECTURE *94*
8-3.1 Generalized CTC Block Diagram *94*
8-3.2 The Channel Logic Block Diagram *94*
8-3.3 Interrupt Control Logic and Protocol *96*
8-4 PACKAGE PIN ASSIGNMENTS AND FUNCTIONS *96*
8-5 THE OPERATING MODES *97*
8-5.1 Setting Up the CTC Chip *97*
8-5.2 Operating in the Counter Mode *97*
8-5.3 Operating in the Timer Mode *100*
8-6 THE CTC TIMING *100*
8-6.1 The CTC Write Cycle *100*
8-6.2 The CTC Read Cycle *101*
8-6.3 The CTC Counter Mode *101*
8-6.4 The CTC Timer Mode *102*
8-6.5 Interrupt Acknowledge Cycle Timing *102*
8-6.6 Return from Interrupt Timing *103*
8-7 REVIEW QUESTIONS *103*

9 PROGRAMMING THE CTC CHIP

CHAPTER OBJECTIVES *106*
TEXTBOOK REFERENCES *106*
9-1 INTRODUCTION *106*
9-2 THE PROGRAMMING WORDS *107*
9-2.1 The Interrupt Vector *107*
9-2.2 The Channel Control Word *107*
9-2.3 The Time Constant Word *108*
9-3 PROGRAMMING FOR COUNTER APPLICATIONS *108*
9-4 PROGRAMMING FOR TIMER APPLICATIONS *109*
9-5 REVIEW QUESTIONS *11*

10 Z80 DIRECT MEMORY ACCESS CHIP

CHAPTER OBJECTIVES *113*
TEXTBOOK REFERENCES *114*
10-1 INTRODUCTION *114*
10-2 PRINCIPAL FEATURES OF THE DMA *114*
10-2.1 General and Electrical Features *114*
10-2.2 Operational Features *114*
10-3 THE DMA ARCHITECTURE *115*
10-4 PACKAGE PIN ASSIGNMENTS AND FUNCTIONS *116*
10-5 FUNCTIONAL OPERATION OF THE DMA *117*
10-5.1 The Classes of Operations *120*
10-5.2 Modes of Operation *120*
10-5.3 The Interrupt System *121*
10-5.4 Pulse Generation *121*
10-5.5 Command and Control Bytes *121*
10-5.6 Reading from Internal Registers *121*

10-5.7 Variable Cycle Length Time *122*
10-5.8 Port Addressing *122*
10-5.9 Auto Restart *122*
10-6 THE DMA TIMING *122*
10-6.1 The Variable Cycle Timing *122*
10-6.2 DMA Inactive State Timing *122*
10-6.2.1 *Command/Control Byte Write Cycle* *122*
10-6.2.2 *DMA Register Read Cycle* *123*
10-6.3 DMA Active State Timing *123*
10-6.3.1 *Memory-to-I/O Transfers* *123*
10-6.3.2 *Memory-to-Memory Transfers* *124*
10-6.3.3 *I/O-to-Memory Transfers* *124*
10-6.3.4 *I/O-to-I/O Transfers* *124*
10-6.4 Bus Request and Acceptance Timing *124*
10-6.5 Bus Release Timing for Burst and Continuous Modes *125*
10-6.5.1 *At End of Block* *125*
10-6.5.2 *On "Not Ready"* *125*
10-6.5.3 *On "Match"* *125*
10-6.6 Bus Release Timing for Byte-at-a-Time Mode *126*
10-7 REVIEW QUESTIONS *127*

11 PROGRAMMING THE DMA CHIP

CHAPTER OBJECTIVES *131*
TEXTBOOK REFERENCES *131*
11-1 INTRODUCTION *131*
11-2 COMMAND WORDS FOR DMA SET-UP *132*
11-2.1 The WR0 Group *132*
11-2.2 The WR1 Group *132*
11-2.3 The WR2 Group *133*
11-2.4 The WR3 Group *133*
11-2.5 The WR4 Group *134*
11-2.6 The WR5 Group *135*
11-2.7 The WR6 Group *135*
11-2.8 The Status Register *136*
11-3 PROGRAMMING THE DMA FOR TRANSFER-ONLY OPERATIONS *136*
11-3.1 I/O-to-Memory Transfer *137*
11-3.2 Memory-to-Memory Transfer *137*
11-4 PROGRAMMING THE DMA FOR SEARCH OPERATIONS *138*
11-5 PROGRAMMING THE DMA FOR SEARCH-TRANSFER OPERATIONS *140*
11-6 REVIEW QUESTIONS *140*

12 Z80 SERIAL INPUT/OUTPUT CHIP

CHAPTER OBJECTIVES *142*
TEXTBOOK REFERENCES *142*
12-1 INTRODUCTION *142*
12-2 PRINCIPAL FEATURES OF THE SIO *143*
12-3 THE SIO ARCHITECTURE *143*
12-3.1 The Generalized SIO Block Diagram *143*
12-3.2 The Receive Data Path *144*
12-3.2.1 *The Asynchronous Mode* *144*
12-3.2.2 *The Synchronous Mode* *145*

A. BYTE-ORIENTED PROTOCOLS *145*
B. BIT-ORIENTED (SDLC MODE) *146*
12-3.3 The Transmit Data Path *146*
12-3.3.1 *The Asynchronous Mode* *146*
12-3.3.2 *The Synchronous Mode* *147*
12-4 PACKAGE PIN ASSIGNMENTS AND FUNCTIONS *147*
12-4.1 The CPU Side Pins *148*
12-4.2 The I/O Side Pins *149*
12-5 THE SIO TIMING *151*
12-5.1 The Read Cycle *152*
12-5.2 The Write Cycle *152*
12-5.3 The Interrupt Acknowledge Cycle *152*
12-5.4 The Return from Interrupt Cycle *153*
12-6 NOTES ON CRC OPERATION *153*
12-6.1 Introduction *153*
12-6.2 Principle of Operation *153*
12-6.3 The CRC Generator in the Transmitter *154*
12-6.4 The CRC Generator in the Receiver *155*
12-6.5 The Comparison Process in the Receiver *155*
12-7 REVIEW QUESTIONS *156*

13 Z80 DUAL ASYNCHRONOUS RECEIVER/TRANSMITTER CHIP

CHAPTER OBJECTIVES *159*
TEXTBOOK REFERENCES *159*
13-1 INTRODUCTION *159*
13-2 PRINCIPAL FEATURES OF THE DART *160*
13-3 THE DART ARCHITECTURE *160*
13-3.1 The Generalized DART Block Diagram *160*
13-3.2 The Receive Data Path *161*
13-3.3 The Transmit Data Path *162*
13-4 PACKAGE PIN ASSIGNMENTS AND FUNCTIONS *162*
13-4.1 The CPU Side Pins *162*
13-4.2 The I/O Side Pins *162*
13-5 THE DART TIMING *166*
13-5.1 The Read Cycle *166*
13-5.2 The Write Cycle *166*
13-5.3 The Interrupt Acknowledge Cycle *167*
13-5.4 The Return from Interrupt Cycle *167*
13-6 COMMAND WORDS FOR DART INITIALIZATION *167*
13-6.1 Write Command Words *167*
13-6.1.1 *The WR0 Register* *167*
13-6.1.2 *The WR1 Register* *168*
13-6.1.3 *The WR2 Register* *169*
13-6.1.4 *The WR3 Register* *169*
13-6.1.5 *The WR4 Register* *169*
13-6.1.6 *The WR5 Register* *170*
13-6.2 The Read Command Words *170*
13-6.2.1 *The RR0 Register* *170*
13-6.2.2 *The RR1 Register* *171*
13-6.2.3 *The RR2 Register* *171*
13-7 REVIEW QUESTIONS *171*

APPENDIX A TIMING DIAGRAM CONVENTIONS *173*
APPENDIX B Z80 ASSEMBLY LANGUAGE CODE BY MNEMONICS *178*

ANSWERS TO SELECTED REVIEW QUESTIONS *187*
INDEX *191*

1 THE Z80 CPU ARCHITECTURE

CHAPTER OBJECTIVES

The objectives of this chapter are as follows:

1. To introduce students to the Z80 μC system and its principal hardware components.
2. To describe the hardware architecture of the Z80 CPU.
3. To describe the general-purpose (GP) and the special-purpose (SP) registers and their respective functions.
4. To present the functions performed by the arithmetic logic unit (ALU).
5. To discuss the program status word (PSW) and present its format.
6. To describe and discuss the addressing modes used in the Z80 CPU.
7. To discuss the functions of the CPU package pins and present them in convenient tabular form.

TEXTBOOK REFERENCES

For the convenience of students who wish to review the relevant material of this chapter in the textbook, the following sections, and/or pertinent chapters are suggested:

1. For general review of CPU registers and counters	Sec. 1-3.2
2. For index register	Secs. 5-6 and 5-7
3. For program status word	Sec. 3-5.3
4. For addressing modes	Chapter 5
5. For interrupt page address register	Secs. 9-5 and 9-7

1-1 INTRODUCTION

The Z80 microcomputer system is designed and manufactured by Zilog, Inc., of Cupertino, California. At the present time, Mostek, Inc., of Carrollton, Texas, is the official second source for the Z80 products.

The Z80 devices are designed by personnel who were formerly employed by Intel Corporation, and had designed the Intel 8080A, which is an enhanced version of the Intel 8008 μC. Thus, the Z80 naturally tends to have many similarities with the 8080A and can be considered as an enhancement of the 8080A. Intel's enhanced version of the 8080A is the 8085A.

1-2 THE Z80 SYSTEM

1-2.1 Principal Hardware Components

The Z80 μC system consists of five principal hardware components:

1. *The Z80 CPU.* This is the central processing unit which contains the arithmetic logic unit (ALU) and the other logical components required for performing the usual CPU functions.
2. *The Z80 PIO.* This is a programmable, two-port, parallel I/O interface designed for transfer of 8-bit parallel data between the Z80 CPU and a variety of peripherals. The interfaces are TTL compatible and both ports are bidirectional.

3. *The Z80 CTC.* This is a programmable counter timer circuit with four independent channels, capable of providing counting and timing control functions for real-time events. The inputs and the outputs on this chip are also TTL compatible.
4. *The Z80 SIO.* This is a programmable, dual-channel, serial I/O interface designed for communication with a variety of serial data peripherals. It is primarily a serial-to-parallel and parallel-to-serial converter/controller using a number of different protocols.
5. *The Z80 DMA.* This is a programmable direct memory access controller which can directly transfer data between the memory and the previously mentioned PIO and SIO interfaces in a cycle-stealing mode of operation.

Note: All the preceding interfaces and controllers monitor the status of the peripherals. CPU polling is not involved. They also include priority interrupt systems which can be implemented by using the daisy chain logic technique. Additional external logic is not required for this.

1-2.2 The Development System

Zilog, Inc., also provides a hardware/software development system for the Z80 product line. Briefly, they include the following features:

1. The program development is RAM-based, not ROM-based.
2. The system is configured around a stand-alone, floppy disk.
3. The system provides real-time debugging capability.
4. The offline support software includes:
   a. Assembler.
   b. Compiler.
   c. Simulator.
   d. Test pattern generation capability.
5. The resident software includes:
   a. RAM-based assembler.
   b. RAM-based text editor.
   c. The BASIC compiler.
   d. Real-time debugging program.
   e. Disk operating system (DOS) with file maintenance.
   f. ROM-based executive firmware.
6. Diagnostic capability is also available. It includes:
   a. Rapid fault detection by means of system diagnostic software.
   b. Continuous memory diagnostics performed in a background operation mode.
7. It is possible to replace RAM chips by ROM or PROM chips.

1-3 PRINCIPAL FEATURES OF THE Z80 CPU

The main features of the CPU chip follow in summarized form:

1. The Z80 is a third-generation μP which uses the N-channel, depletion-mode, silicon-gate technology.
2. A single 5-volt ±5% power supply is used in the system.
3. A single-phase TTL-level clock is used.
4. The Z80 uses a 2.5-MHz clock. This gives an instruction execution time of 1.5 μs.
5. For a higher throughput rate, the Z80 uses a 4.0-MHz clock which gives an instruction execution time of 1.0 μms.
6. The Z80 CPU chip is available in a standard 40-pin, dual-in-line (DIP) package.
7. The Z80 has an instruction set of 157 instructions which is software compatible with the 78 instructions of the Intel 8080A μP.
8. The Z80 instruction set includes instructions for
   a. Bit manipulation.
   b. Byte manipulation.
   c. Character string operations.
   d. Block transfers of data.
9. Two 16-bit index registers are available. They enhance the processing of look-up tables and arrays.
10. The Z80 includes a duplicate set of general-purpose (GP) registers and status registers. The CPU has a total of 16 such registers.
11. The Z80 CPU generates all the control signals for standard memory chips. Thus,
    a. Using only an external address decoder, static memory chips can be readily interfaced with the CPU.
    b. All controls for refreshing dynamic MOS memory chips are provided.
    c. The control bus signals are capatible with the commonly used 18-pin and 22-pin 4K RAMs.
12. Expanded 16-bit arithmetic operations, as well as BCD operations, are available in the Z80 CPU.
13. The Z80 chips are designed to operate in temperature environments ranging from 0° to 55°C.

1-4 THE CPU ARCHITECTURE

In the Z80 CPU there is a total of 22 registers/accumulators which are divided into two groups, namely, the general-purpose (GP) group and the special-purpose (SP) group, which contains registers dedicated to performing

only certain specific functions. Four of these registers are 16-bit units and the rest are 8-bit registers. The CPU has two accumulators. Additionally, the CPU has a 16-bit instruction register, with its associated instruction decoder, and the timing and control. All the previously mentioned registers are programmable, i.e., they can be accessed by the programmer.

1-4.1.1 GP Registers, Accumulators, and Status Flag Registers

1. There is a total of 16 8-bit GP registers. Two of these are accumulators and two are status flag registers, assocated with each of the two accumulators.
2. These 16 registers are divided into two sets, the main set and the alternate set, as shown in Fig. 1-1.
3. The main register set units are labeled A (for the accumulator) and F (for its corresponding flag register). The rest are labeled B, C, D, E, H, and L. Since registers A and F perform specific functions, it is more appropriate to say that the true GP registers are B, C, D, E, H, and L registers only.
4. The second set is called the alternate register set and is labeled in exactly the same manner but with a prime attached to each letter symbol. Once again B′, C′, D′, H′, and L′ are the true GP register, while A′ and F′ are the accumulator and the status flag registers, respectively.
5. Accumulator A is the primary accumulator. Whenever a PUSH or a POP operation is performed involving the accumulator and the flag register, both registers are always transferred.
6. The remaining six registers, in both the main and the alternate set, either can be used individually as 8-bit units or can be paired and used as 16-bit units as shown below:

Main Set	*Alternate Set*
B with C	B′ with C′
D with E	D′ with E′
H with L	H′ with L′

7. The register pairs BC, DE, HL, and B′C′, D′E′, H′L′ can be used as secondary registers and/or data counters.
8. Under program control, either the main set or the alternate set can be selected and operated by means of a single instruction. Only one set can be used at any one time, not both.

1-4.1.2 Special-Purpose Registers

1. The Z80 CPU has six special-purpose registers (SPR).
2. There are two 8-bit registers and four 16-bit registers, as shown in Fig. 1-2.
3. A brief explanation of each SPR register follows:
   a. *Program counter (PC).* At any time, the PC holds the 16-bit address of the current instruction fetched from the program memory.
   After the contents of the PC (i.e., the address of the instruction) are placed on the address bus, the PC is automatically incremented.
   b. When a JUMP or a BRANCH is executed, the new address is automatically inserted in the PC.
   Stack pointer (SP) The Z80 uses a pointer-type stack to provide return addresses from BRANCH operations.

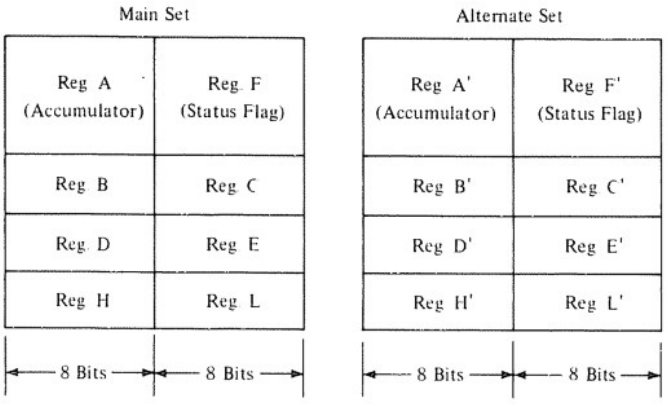


Figure 1-1 General-Purpose Registers in the Z80 CPU

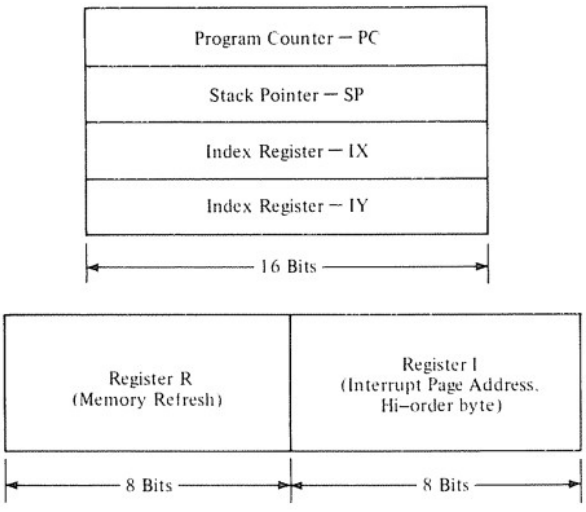


Figure 1-2 Special-Purpose Registers in the Z80 CPU

A previously designated portion of the external RAM memory is used for the stack.

The stack is organized on a last-in, first-out (LIFO) basis.

By means of PUSH/POP instructions, data can be pushed onto the stack from any specified CPU register. Likewise, data can be popped from the stack into any specified CPU register.

The stack provides unlimited nesting capability.

Capability for multiple-level interrupts is also provided.

c. *Index registers (IX and IV).* These are two completely independent index registers.

Their primary function is to provide a 16-bit base address in the indexed addressing mode.

They are extensively used in table look-up operations when locations in the memory are to be accessed.

For indexed addressing, a displacement byte is included in the instruction. The displacement is a signed twos complement integer and it is added to the base address in the index register to obtain the true or effective address.

d. *Memory refresh register (R).* The Z80 is one of the μPs which provides for the use of both static and dynamic MOS RAM chips in its system.

Dynamic memory chips are relatively inexpensive but can hold the information in them for only a very short time.

Thus, at millisecond intervals, the contents of each memory location are read out and rewritten in the same memory location.

The preceding process is called dynamic memory refreshing.

A counter is needed to track and update each memory address as each location is refreshed.

In the Z80 CPU, the R register, which is a counter, performs the address incrementing function.

The memory refreshing operation is completely transparent to the programmer or the user.

Although the R register is not used for operations other than refreshing, it is programmable, i.e., the programmer can externally load the register for testing purposes.

e. *Interrupt Page Address Register (I).* In response to an interrupt, the Z80 provides the capability to branch to a service subroutine by means of an indirect addressing mode as an optional addressing mode.

Indirect addressing requires that the memory page address be provided as well as the local address in that particular page.

That I register is used for storing the high-order byte (i.e., the page address) of the 16-bit starting address of the subroutine.

The local address (i.e., the low-order byte) is provided by the interrupting device itself.

The I register is also called the interrupt vector register.

1-4.2 The Arithmetic Logic Unit (ALU)

1. The ALU normally performs arithmetic and logic operations on 8-bit operands. Additionally, the ALU can also perform additions and subtractions on 16-bit operands located in the 16-bit registers.
2. The ALU communicates with the other internal registers of the CPU by means of the 8-bit internal data bus.
3. The ALU performs the following arithmetic functions:
   - **a.** Add.
   - **b.** Subtract.
   - **c.** Increment.
   - **d.** Decrement.
4. The ALU performs the following logical functions:
   - **a.** AND.
   - **b.** OR.
   - **c.** Exclusive-or.
   - **d.** Compare.
   - **e.** Right rotate.
   - **f.** Left rotate.
   - **g.** Right shift logical.
   - **h.** Left shift logical.
   - **i.** Right shift Arithmetic.
   - **j.** Left shift Arithmetic.
5. The ALU performs the following additional functions:
   - **a.** Test or sample bit.
   - **b.** Set bit (to a binary one).
   - **c.** Reset bit (to a binary zero).

1-4.3 Instruction Register, Decode, and Control

1. Figure 1–3 shows a generalized block diagram of the Z80 CPU.
2. When the instruction is fetched from the program memory, it is loaded into the 16-bit instruction register.
3. The instruction is then decoded and various control signals are sent to the CPU and external devices of the system as commanded by the particular instruction.

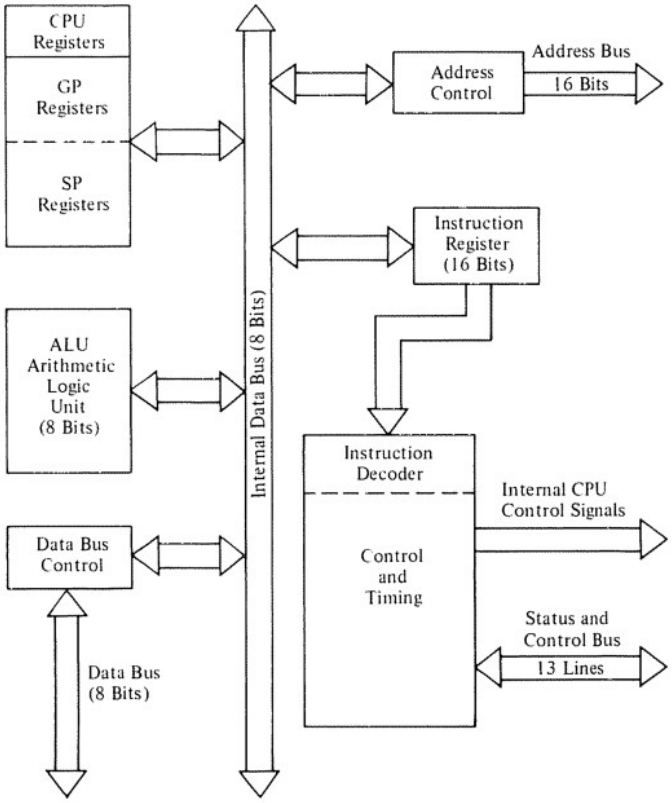


Figure 1-3 Generalized Block Diagram of the Z80 CPU

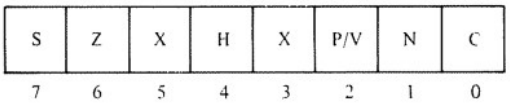


Figure 1-4 Program Status Word Format

1-4.4 Program Status Word (F and F′)

From Fig. 1-1 we see that each accumulator, A and A′, has an associated status flag register, F and F′. The function of each of these two registers is to flag out and indicate the current status of certain conditions in its associated accumulator resulting from either an arithmetic or logical operation. Although both registers, F and F′, are 8 bits long, only six of these bits are used in the Z80. The format of these two registers is shown in Fig. 1-4. The two unused bit positions are shown as X. Also remember that these two registers are made up of flip-flops which can be set to the 1 side or reset to the 0 side. A brief explanation of each status flag follows:

1. *Sign bit (S).* The sign bit flag is used in conjunction with arithmetic operations that use signed numbers (a 1 for negative numbers and a 0 for positive numbers). The most-significant-bit (MSB) of the number is the sign bit. This means that the S flag bit indicates the status of the least significant 7 bits in the accumulator associated with the status flag register. The S flag is set if the MSB of the accumulator is a 1 and reset if it is a 0.
2. *Zero bit (Z).* The zero flag checks all the bits in the accumulator which results from either an arithmetic or a logical operation. If all the bits in the associated accumulator are 0, the Z flag is set to the 1 side; otherwise, it is reset to the 0 side.
3. *Half carry bit (H).* Binary-coded decimal (BCD) arithmetic can be used in the Z80. Because of the nature of BCD additions and subtractions, a BCD carry or borrow could result from arithmetic operations on the least significant 4 bits of the operands. Consequently a decimal arithmetic adjust (DAA) operation must be performed to produce a correct BCD answer. The H flag is set when such a DAA operation is required. Note that the H flag is also referred to as the auxiliary carry flag and is indicated by the symbol A_C. The H flag is never tested by conditional JUMP or BRANCH instructions.
4. *Parity/overflow bit (P/V).* This bit is used for two separate functions in the Z80 CPU. When used as a parity bit, it is set if the result of a logical operation (such as AND, OR, XOR) results in an even number of ones in the accumulator. Also, during one specific input operation, i.e., IN r, (C), the parity of the incoming data byte is indicated by this flag. Other normal input operations do not affect this flag. As an overflow bit, this flag is used when signed twos complement arithmetic is performed. Since the MSB of the accumulator is the sign bit, this flag is set when the resulting quantity in the accumulator exceeds the maximum possible number +127 or is less than the minimum number −128.
5. *Subtract bit (N).* The DAA operations, required in BCD arithmetic, are performed by two different algorithms, one for the addition and the other for the subtraction process. The N flag indicates which instruction was executed that required the DAA operation, addition or subtraction, so that the appropriate algorithm may be selected and executed. The flag is set to the 1 side if a subtraction is involved.
6. *Carry Bit (C).* If a carry is generated from the MSB of the accumulator, the C flag is set. This flag is set if a carry is generated as a result of an addition process and also if a borrow is generated because of a subtraction process. Also, in shift and rotate operations involving the accumulator, this flip-flop can be included as a link flip-flop under program control.
7. *X Bits.* Bits in positions 3 and 5 of the status word, which are indicated by X in Fig. 1-4, are ignored. They are not used in any operations.

1-5 ADDRESSING MODES

The Z80 system offers a wide variety of addressing modes, most of which have been covered in considerable detail in Chapter 5 of the textbook. Consequently, only a brief explanation of each, as used in the Z80 CPU, is given here.

1-5.1 Implied Addressing

This is a very simple mode of addressing where the OP code within the instruction itself implies one or more of the CPU registers as either the source or the destination of the operand(s). A typical example of this addressing mode is that of arithmetic operations in which the destination of the result is always automatically implied as one of the two accumulators.

1-5.2 Register Addressing

The Z80 CPU architecture provides the capability to address specific registers in the CPU which can be used for certain operations. The register(s) concerned is (are) identified by certain bits of information in the OP codes themselves. An example of this mode of addressing is an instruction which would load (i.e., transfer) the contents of register D into register E.

1-5.3 Register Indirect Addressing

In this addressing mode, a 16-bit pointer, which is the ultimate address of the memory location to be accessed, is stored in one of the register pairs in the CPU, usually the HL pair. The specific register pair of the CPU is addressed by certain bits in the OP code of the instruction which could be either one or two bytes. This addressing mode is very powerful. Very often this mode is used to access 16-bit operands in the memory which are generally stored as 2 bytes in two successive memory locations. In such a situation the CPU register pair (such as the HL pair) points to the location of the low-order byte of the desired operand. The high-order byte is then accessed simply by incrementing the contents of the register pair, which would be the HL pair in this case.

1-5.4 Bit Addressing

Several instructions in the Z80 enable the programmer to set, reset, and test or sample any bits of words located in any of the CPU registers or any locations in the memory. These bit operations are performed by specifying the location of the word by means of one of the three addressing modes, namely, register addressing (Sec. 1-5.2), register indirect addressing (Sec. 1-5.3), and indexed addressing, which is covered in Sec. 1-5.8. The particular bit to be manipulated in the specified word is identified by a 3-bit field in the OP code of instruction.

1-5.5 Immediate Addressing

In this addressing mode, the operand is not located in any CPU register or memory location. Instead, it is contained directly within the instruction itself. The instruction could be either a 2-byte or a 3-byte instruction with either a single-byte or a 2-byte OP code. The 8-bit operand is contained in the following byte.

1-5.6 Immediate Extended Addressing

This mode is similar to the immediate addressing mode described in Sec. 1-5.5. The only difference is that the OP code (either 1 or 2 bytes) is followed by a 2-byte operand. The first byte of the operand is the low-order byte and the second byte is the high-order byte. Thus, the instruction could be either 3 or 4 bytes long.

1-5.7 Extended Addressing (Direct Addressing Mode)

This addressing mode involves the standard direct addressing mode. The OP code can be either 1 or 2 bytes long. The address field always contains two bytes. In this addressing mode, the two address bytes of the instruction contain the true or effective memory address where the operand is located or is to be stored. By means of the extended addressing mode, all locations in the memory can be addressed directly.

1-5.8 Indexed Addressing

This is standard indexed addressing mode whose operation is explained in detail in Sec. 5-7 of the textbook. The OP code of the instruction consists of 2 bytes. The address field is a single byte and it contains the displacement which, when added to the contents of the previously loaded index register, gives the true or effective address of the memory location to be accessed. In some cases the instruction could have a fourth byte if an immediate operand is involved in the operation specified by the instruction.

1-5.9 Relative Addressing

A 2-byte instruction is used for the program counter (PC) relative addressing mode. The first byte contains the OP code and the second byte an 8-bit signed 2s complement displacement. This displacement is added to the address of the next instruction's OP code, which is located in the program counter. Since the instruction using the relative addressing mode itself is 2 bytes long, the displacement is added to the contents of the PC incremented twice. Recall from previous discussion of the relative addressing mode in the textbook (Sec. 5-5) that this method normally allows

access to an address range from +127 to −128 relative to the address of the instruction contained in the program counter. However, in the Z80 CPU, this is not the case. Because the signed displacement is added to PC + 2, the accessible memory address ranges from +129 to −126.

1-5.10 Modified Page 0 Addressing

A modified version of the base page or page 0 addressing mode, described in Sec. 5-4.2 and Sec. 5-4.3 of the hardcover textbook, is available in the Z80. This mode is initiated by a single-byte CALL instruction (often referred to as a restart instruction) which enables access to eight locations in page 0 of the memory. These eight locations contain 16-bit pointers which are addresses in the memory. This addressing mode is used to BRANCH to frequently used subroutines having 16-bit starting addresses by means of a single-byte instruction. It operates as follows:

1. The instruction contains a 3-bit code which specifies the address of one of the eight reserved locations in page 0 of the memory.
2. The eight locations in page 0 are previously loaded with the 16-bit starting addresses of the frequently used subroutines.
3. The incremented contents of the program counter are pushed onto the stack.
4. The 16-bit address in the accessed location of page 0 is then loaded into the program counter (at the programmer's option).

The previously-mentioned application of these reserved locations is frequently used but is not the only option available to the programmer. These locations can also be used for other purposes. For instance, they may contain the actual subroutine if it is a short one. Note that this mode of addressing is really a combination of the page 0 relative and the extended indirect addressing modes.

1-6 PACKAGE PIN ASSIGNMENTS AND FUNCTIONS

The Z80 CPU chip is available in a standard 40-pin, dual-in-line-package (DIP). The package pins and their respective functions, together with their symbolic designations, are presented in convenient tabular form in Tables 1-1 through 1-6. Functionally, the pins can be divided into five groups as shown in Fig. 1-5. Note that some lines are shown as unidirectional and others as bidirectional. In the latter group some of the lines may indicate signal flow in one direction and others in the other direction. The tables clarify this with the following arrow symbols:

CPU → Indicates signals flowing from the CPU.
CPU ← Indicates signals flowing into the CPU.
CPU ← Indicates bidirectional signal flow on the same line.

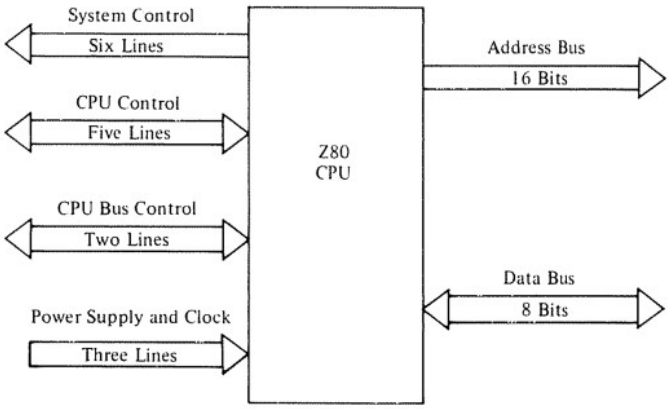


Figure 1-5 Diagram of CPU Package Pin Groups

Table 1-1 ADDRESS AND DATA BUSES GROUPS

Pin Designations	*Signal Flow Directions*	*Functions*
A0–A15	CPU ⟶	16-bit, tristate, address bus, is active high. Provides direct addressing capability for 64K byte memory and I/O devices for data transfers. Up to 256 input and 256 output ports may be selected by the lower 8 bits. A0 is LSB and A15 is MSB.
D0–D7	CPU ⟷	8-bit, tristate, data I/O bus is active high. It is used for data transfers between the CPU and the memory as well as between the CPU and the I/O devices.

Table 1-2 SYSTEM CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
$\overline{M1}$	CPU ⟶	Output signal indicating that the current machine cycle involves OP code fetch subcycle. If the OP code contains 2 bytes, this signal is generated as each byte is fetched. When this signal is active with $\overline{IORQ}$ signal, it indicates an interrupt acknowledge cycle. It is active low.
$\overline{MREQ}$	CPU ⟶	Tristate, memory output request signal indicating that the bits on the address bus are valid memory address. It is active low.
$\overline{IORQ}$	CPU ⟶	Tristate, I/O request signal indicates that the low byte of the address bus has a valid I/O address for I/O data transfers. When interrupt is acknowledged, this signal is simultaneously generated, with $\overline{M1}$ indicating that an interrupt vector can be placed on the data bus. It is active low.
$\overline{RD}$	CPU ⟶	Tristate signal indicating that the CPU is ready to read data output by either the memory or the I/O device. On receiving this signal, the memory or the I/O device gates the data onto the data bus. It is active low.
$\overline{WR}$	CPU ⟶	Tristate signal to the memory or I/O device indicating that the CPU has placed valid data on the data bus for acceptance by the memory or the I/O device.
$\overline{RFSH}$	CPU ⟶	Indicates that lower 7 bits of the address bus contain a refresh address for dynamic memories. Thus, the current $\overline{MREQ}$ signal executes a refresh read for all such memory chips.

Table 1-3 CPU CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
$\overline{\text{HALT}}$	CPU ⟶	Indicates that the CPU has executed a HALT instruction. CPU then resumes execution only on receipt of an interrupt (either maskable or nonmaskable). During HALT, CPU executes NOOP instructions so that information in dynamic memories can be maintained by refreshing.
$\overline{\text{WAIT}}$	CPU ⟵	An active low signal from either the memory or the I/O device indicating that it is not ready for a data read or write operation. The CPU idles until the memory or I/O indicates that it is ready for a transfer.
$\overline{\text{INT}}$	CPU ⟵	Interrupt request signal, generated by the I/O devices. CPU completes current instruction and honors this signal only if the software-controlled interrupt F/F is enabled (IFF) and the $\overline{\text{BUSRQ}}$ signal is inactive. CPU sends out an acknowledge signal $\overline{\text{IORQ}}$ at the beginning of the next instruction cycle.
$\overline{\text{NMI}}$	CPU ⟵	This is a negative-going, edge-triggered, nonmaskable interrupt request which has a higher priority than $\overline{\text{INT}}$ and can override the IFF, which is software controlled. The contents of the program counter are saved in the stack for return to the original point in the interrupted program.
$\overline{\text{RESET}}$	CPU ⟵	This signal forces the program counter to reset to 0. It initializes the CPU and disables the interrupt enable F/F. Also, all control output signals are deactivated and the address and data busses go into the high-impedance states.

Table 1-4 CPU BUS CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
$\overline{\text{BUSRQ}}$	CPU ←—	This signal puts the tristate address and data buses, as well as the tristate output control signal lines, into the high-impedance state. The external devices then control these buses and lines. The CPU, of course, completes its current machine cycle and then complies with this request.
$\overline{\text{BUSAK}}$	CPU —→	The CPU sends this acknowledge signal to the requesting external device that the address and data buses as well as the tristate output control lines are in the high-impedance state and are under control of the external device.

Table 1-6 shows the actual pin numbers assigned to each of the functional designation covered in Tables 1-1 through 1-6.

Table 1-5 POWER SUPPLY AND CLOCK GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
ø	CPU ←—	This is the single-phase, TTL-level clock input. A 330-ohm pull-up resistor to 5 volts is required.
5 V	CPU ←—	This is the single power supply of 5 volt ±5 % that is needed for the Z-80
GND	CPU ←—	This is the power supply ground.

Table 1-6 PIN ASSIGNMENT NUMBERS

PIN DESIGNATIONS	*PIN NUMBERS*	*PIN DESIGNATIONS*	*PIN NUMBERS*
Address Bus		*System Control*	
A0	30	$\overline{M1}$	27
A1	31	$\overline{MREQ}$	19
A2	32	$\overline{IORQ}$	20
A3	33	$\overline{RD}$	21
A4	34	$\overline{WR}$	22
A5	35	$\overline{RFSH}$	28
A6	36		
A7	37	*CPU Control*	
A8	38	$\overline{HALT}$	18
A9	39	$\overline{WAIT}$	24
A10	40	$\overline{INT}$	16
A11	1	$\overline{NMI}$	17
A12	2	$\overline{RESET}$	26
A13	3		
A14	4	*CPU Bus Control*	
A15	5	$\overline{BUSRQ}$	25
Data Bus		$\overline{BUSAK}$	23
D0	14		
D1	15	*Power Supply and Clock*	
D2	12		
D3	8	ø	6
D4	7	+5V	11
D5	9	GND	29
D6	10		
D7	13		

1-7 REVIEW QUESTIONS

1-1 The Z80 μP is similar to some of the currently available μPs on the market in many respects. Circle the correct one(s).

a. The Z-8000 by Zilog
b. The 8086 by Intel
c. The 8080A by Intel
d. The 6800 by Motorola

1-2 The following statements apply to the Z80 μC system. Indicate true or false.

a. ____ The Z80 CPU is a third-generation μP.
b. ____ The Z80 uses a ± 5% power supply.
c. ____ An instruction execution time of 1.0 μs is obtained with a 2.5-MHz clock.
d. ____ The Z80 CPU is available in a 28-pin DIP package.

1-3 The following statements relate to the instructions available for the Z80. Indicate true or false.

a. ____ Byte manipulation is possible.
b. ____ Bit manipulation capability is *not* available.
c. ____ Block of data can be transferred.
d. ____ Character string operations are available.

1-4 Fill in the blanks.

a. Index registers are primarily used for _____ _____ and _____ operations. (array; direct addressing; table look-ups; JUMP; BRANCH)
b. There is (are) _____ index registers in the Z80. (1; 2; 4; 8; 16)
c. The index register(s) is (are) _____ bits long. (4; 8; 12; 16)

1-5 Name one technique that can be used for priority interrupts in the Z80 (at the user's option).

1-6 The Z80 uses a duplicate set of GP registers. This is done to provide (circle all that apply):

a. Capability to handle both 8- and 16-bit operands.
b. Faster memory data transfers.
c. Faster interrupt processing capability.
d. Faster I/O data transfers.

1-7 Indicate true or false for the Z-80.
- **a.** ____ Operates in a temperature range of −55° to +55°C.
- **b.** ____ BCD arithmetic operations are available.
- **c.** ____ Both static and dynamic memory chips can be used.
- **d.** ____ A total of eight GP registers are available.

1-8 The alternate register set in the Z80 includes _____ registers, each _____ bits long. (4; 6; 8; 12; 16; 32).

1-9 _____ How many bits are used in the status flag register of the main register set? (2; 4; 6; 8; 16)

1-10 The following questions relate to the GP registers in the Z80. Answer them yes or no.
- **a.** ____ Is it possible to use register A′ and F′ as a 16-bit unit?
- **b.** ____ Is it possible to pair register D and D′ as a 16-bit unit?
- **c.** ____ Is it possible to use register C as a status register for register B?

1-11 Assume that the main register set is used during a certain program and an interrupt signal comes in. Indicate true or false for the following statements for the time during which a fast interrupt subroutine is serviced.
- **a.** ____ The contents of all the alternate set registers are automatically transferred into the registers of the main set.
- **b.** ____ The alternate set's registers are used by the interrupt service subroutine.
- **c.** ____ The interrupt service routine uses the registers from both the main and the alternate sets.
- **d.** ____ Status saving may be done in the alternate set.

1-12 The Z80 CPU has _____ SP registers. (4; 6; 8; 10; 12; 16)

1-13 In the Z80 CPU the program counter is incremented (circle the correct statement)
- **a.** Before the contents of the PC are on the address bus
- **b.** After the contents of the PC are on the address bus.

1-14 Indicate true or false for the Z80 CPU SP registers.
- **a.** ____ A LIFO cascade stack is used.
- **b.** ____ The stack provides unlimited nesting capability.
- **c.** ____ Multiple-level interrupts are possible.
- **d.** ____ Registers IX and IY are completely independent of each other.

1-15 Which of the following SP registers is used for keeping track of the addresses during the refreshing of dynamic RAM chips? (Circle the correct one.)
- **a.** Stack pointer
- **b.** Program counter
- **c.** Register I
- **d.** Register R
- **e.** Register IX
- **f.** Register IY

1-16 The following statements apply to interrupt processing and register I. Indicate true or false.
- **a.** ____ Register I is used for directly branching to the interrupt service subroutine.
- **b.** ____ Register I stores the *entire* 16-bit starting address of the interrupt service subroutine.
- **c.** ____ The local address of the interrupt service subroutine is provided by the interrupting device itself.
- **d.** ____ The page address of the interrupt service subroutine's starting address is stored in the I register.

1-17 Circle the statement(s) which apply to the Z (zero) flag bit in the F and F′ registers.
- **a.** The Z flag checks all the GP registers for 0s.
- **b.** The Z flag checks all the bits of its corresponding accumulator.
- **c.** If all the bits of the accumulator are 0, the Z F/F is set to the 1 side.

1-18 When is the sign bit flag (S) in the status register set? Circle the correct answer(s).
- **a.** When the MSB of the accumulator indicates a positive number.
- **b.** When any of the GP register of that particular set is a negative quantity.
- **c.** When the MSB of the corresponding accumulator is a 1.
- **d.** When the LSB of the corresponding accumulator is a 1.

1-19 The statements below apply to the carry bit (C) in the F and F′ registers. Indicate true or false.
- **a.** ____ C is set if there is no carry from an arithmetic operation.
- **b.** ____ C is set if a borrow results from a subtraction operation.
- **c.** ____ C is set if a carry is generated from the MSB position of the accumulator.

1-20 Fill in the blanks. The half carry bit (H) is _____ to indicate the necessary correction for the carry/borrow resulting in the least significant 4 bits from an

_____ arithmetic operation. (BCD; set; reset; hex; binary; octal)

1-21 The following statements apply to the DAA operations and the Subtract Bit flag (N). Indicate true or false.

a. ___ In DAA operations the same algorithm is used for addition and subtraction processes.

b. ___ The N flag is set if the prior instruction executed is a subtraction instruction.

1-22 The following statements apply to the P/V flag bits. Indicate true or false.

a. ___ The P flag is set if the result in the accumulator contains an even number of 1 bits.

b. ___ The overflow flag, V, is used when a logical function is performed and the result is stored in the accumulator.

c. ___ The V flag is set when the resulting quantity in the accumulator exceeds +127.

d. ___ The V flag is set when the resulting quantity in the accumulator is less than −128.

1-23 The questions below apply to the Z80 addressing modes. Answer them.

a. _____ Is it always necessary to add second byte after the OP code if register B is to be accessed?

b. _____ Refer to (a) above. Which addressing mode will be used in this situation?

c. _____ Which addressing mode contains bits in the OP code to identify the accumulator as the destination?

1-24 Which addressing mode in the Z80 CPU will allow a 2-byte operand in the memory to be accessed by means of a single-byte address in the instruction.

_____

1-25 Refer to 1-24. Where is the pointer for the 2-byte memory operand stored?

_____

1-26 Refer to 1-24 and 1-25. Do we need a second instruction to access the second byte in the memory? If not, why not?

_____

_____

_____

1-27 We want to reset a certain bit in a memory location. Indicate true or false for the situation.

a. ___ The bit is identified by a separate byte in the OP code.

b. ___ The bit is identified by a 3-bit field in the OP code.

c. ___ Register indirect addressing cannot be used.

d. ___ Indexed addressing can be used.

e. ___ The bit in the memory location can also be set.

1-28 The following statements relate to immediate and immediate extended addressing modes. Indicate true or false.

a. ___ The operand *is not* located in a CPU register.

b. ___ The operand is located in the memory.

c. ___ The operand is located within the instruction itself.

d. ___ The instruction is either 3 or 4 bytes long when the immediate extended addressing mode is used.

1-29 The following statements relate to the extended direct/indirect addressing modes. Fill in the blanks.

a. The OP code can be either _____ or _____ bytes long. (1; 2; 3)

b. The address field always contains _____ byte(s) (1; 2; 3)

c. In direct addressing, the _____ byte(s) contains(s) the true or effective address of the memory location. (1; 2; 3)

d. In direct addressing, the _____ order byte follows the OP code. (high-; low-)

e. In indirect addressing, the address field contains the _____ bit pointer. (8-; 16-)

1-30 The following statements relate to the relative addressing mode in the Z80 CPU. Indicate true or false.

a. ___ A single-byte instruction is used for this mode.

b. ___ The second byte of the instruction contains a signed 2s complement displacement.

c. ___ The displacement is added to the address of the next instruction's OP code in the program counter.

d. ___ The displacement is added to the contents of the PC which is incremented *once*.

e. ___ The accessible memory addresses range from +129 to −126 relative to the address of the instruction.

1-31 When the indexed addressed mode is used, the OP code of the instruction contains _____ byte(s) and the address field contains _____ byte(s). (1; 2; 3)

1-32 Indicate true or false for the modified page zero addressing mode.

a. ___ Pointers from memory addresses are stored in the first eight locations of page 0.

b. ___ The pointers are 16 bits long.

c. ___ 16-bit starting addresses of subroutines can be reached by means of a single byte.

d. ___ The page 0 locations are identified by a 4-bit code in the OP code field.

e. ___ It is a combination of the page 0 relative and the extended indirect addressing modes.

2

THE Z80 INSTRUCTION SET (PART ONE)

CHAPTER OBJECTIVES

The following are the objectives for this chapter:

1. To describe briefly the eight fundamental categories of instructions in the instruction set.
2. To present the generalized instruction formats of the single-, 2-, 3-, and 4-byte instructions.
3. To describe the mnemonic structure of the instructions, including the fixed and the variable parts, and the situation with implied operands.
4. To present and explain the symbolic expressions of operations and functions.
5. To present and explain the status register symbols.
6. To present, in convenient tabular form, the mnemonics, symbolic operations, status flags, OP codes, number of bytes, and operation descriptions of the following groups of instructions:
   a. 8-bit and 16-bit LOAD instructions.
   b. 16-bit STACK PUSH/POP instructions.
   c. EXCHANGE instructions.
   d. BLOCK TRANSFER and SEARCH instructions.
   e. 8-bit arithmetic and logic instructions.
   f. 16-bit arithmetic instructions.

TEXTBOOK REFERENCES

For reviewing material in the textbook, relevant to the topics covered in this chapter, the following chapter and/or pertinent sections are suggested:

1. For a general review of instructions	Chapter 4
2. For categorization of instructions	Sec. 4-2.1
3. For data transfer instructions	Sec. 4-2.2.1
4. For stack operations and instructions	Sec. 3-5.2.2
5. For arithmetic instructions	Sec. 4-2.2.2
6. For logic instructions	Sec. 4-2.2.3
7. For symbolic expressions and mnemonics	Sec. 4-3 and Sec. 4-4.

2-1 INTRODUCTION

The Intel 8080A μP has a set of 78 instructions. Because the Z80 is an enhancement of the 8080A, the Z80 instruction set includes all the instructions of the 8080A. An additional 79 instructions make up a total of 157 instructions. For an 8-bit μP, the Z80 has a very powerful instruction set. The instructions are divided into eight convenient categories which are briefly described here.

2-2 CATEGORIES OF INSTRUCTIONS

2-2.1 The Load and Exchange Group

This group of instructions primarily involves transfers of 8-bit and 16-bit words between either the internal registers of the CPU or the CPU registers and the external memory. The instruction specifies both the source of the word and the destination involved in each transfer. Note that such transfers *do not* destroy the contents of the source location. The exchange instructions swap the contents of the two specified locations. This group also includes *load immediate* instructions which load the operand, contained in the instructions themselves, into one of the specified CPU registers or the specified location in the external memory.

2-2.2 The Block Transfer and Search Group

This group of instructions makes it possible to transfer a block of data (containing any number of words) from one location in memory to another location in memory by means of just one instruction. The block search group of instructions enables the user to search an entire block of data (of any length) in memory for an 8-bit word or character with just one instruction. When the desired word is found or when the end of the block is reached, the instruction and the search are automatically terminated.

2-2.3 The Arithmetic and Logic Group

This group of instructions performs various logical and arithmetic operations on operands. The words to be operated on can be located either in external memory, the accumulator or the GP registers of the CPU. In the Z80, the result of any of these operations is always placed in the accumulator. Depending on the result of the operation, the appropriate flag is then set in the appropriate Status Register, F or F'. Also included in this group are instructions capable of performing 16-bit additions and subtractions on operands located in 16-bit registers.

2-2.4 The Rotate and Shift Group

This group of instructions make it possible to perform right and left shifts as well as right and left rotates on any operands located anywhere in the external memory or in the CPU registers. Right and left rotations can be made either with or without including the carry flip-flop. Both logical and arithmetic shifts can be performed. In logical shifts all bits in the register are given the same treatment. In arithmetic shifts, the sign bit in the most-significant-bit position is preserved and not altered during the shifting operation.

2-2.5 The Bit Manipulation Group (Set, Reset, and Test)

This group of instructions makes it possible to set, reset, or test any bit in any external memory location, the accumulator, or any CPU register with a single instruction. These instructions are particularly useful in control applications.

2-2.6 The JUMP, CALL, and RETURN Group

This group contains instructions for performing JUMPS, CALLS (i.e., BRANCHES), and RETURNS (from BRANCHES). Basically, these instructions are used for transferring control of the CPU from one part of the program to another (JUMPS) or between the main program and subroutines (BRANCHES and RETURNS).

2-2.7 The Input/Output Group

This group of instructions allows the user to perform a variety of different data transfers between the I/O devices and the CPU registers or between the I/O devices and external memory. With just one instruction, it is possible to transfer a single block of data (maximum 256 bytes) directly between an I/O device and the memory. Some instructions in this group will affect some of the flags in the status register (depending on the conditions involved), so that additional operations are not required to determine the status of the transferred data.

2-2.8 The Basic CPU Instructions Group

This group contains instructions which are used to define and control various modes and options available in the CPU. These include setting the mode interrupt response, setting or resetting the interrupt flip-flop, decimal arithmetic adjust (DAA), setting the carry flip-flop, the NOP instruction, and other similar operations.

2-3 INSTRUCTION FORMAT AND SYMBOLS

2-3.1 The Generalized Instruction Format

A Z80 instruction could be made up of 1, 2, 3, or 4 bytes. The OP code portion of the instruction may consist of either 1 or 2 bytes'. Depending on the instruction involved, the OP code is followed by either a 1- or 2-byte address. If the instruction happens to be an immediate

type of instruction, then the OP code is followed by a 1- or 2-byte operand. All the bytes of any multibyte instruction are stored in consecutive memory locations. Figure 2-1 shows some possible combinations of multibyte instructions where the starting address of the OP code is identified as A.

2-3.2 The OP Code

As shown in Fig. 2-1, the OP code of the Z80 instruction can be either 1 or 2 bytes. Using the implied addressing mode, data words can be transferred between the internal registers of the CPU. This means that the instruction itself must contain the codes, identifying addresses, for both the source and the destination registers.

In the OP code byte, the destination register is designated as r_1 and the source register as r_2. The 3-bit identifying codes or addresses, assigned to the 7 CP registers are shown below in Table 2-1. The relative placement of these codes in the OP code determines which register is the source and which is the destination. The instruction shown in Fig. 2-2 explains this.

Suppose we wish to transfer the contents of register D of the main set into register E of the main set. The single-byte instruction is then formatted as shown in Fig. 2-2. The 01⁻ combination in bit positions 7 and 6, respectively, is the register-to-register transfer OP code. Bit combination 010 in bit positions 2, 1, and 0, respectively, identify the source register, as determined by the Table 2-1, and 011 combination in bit positions 5, 4, and 3, respectively, identify the destination register in the main set.

Table 2-1 ADDRESS CODES OF THE INTERNAL GP REGISTER

Register	*Code*
A	111
B	000
C	001
D	010
E	011
H	100
L	101

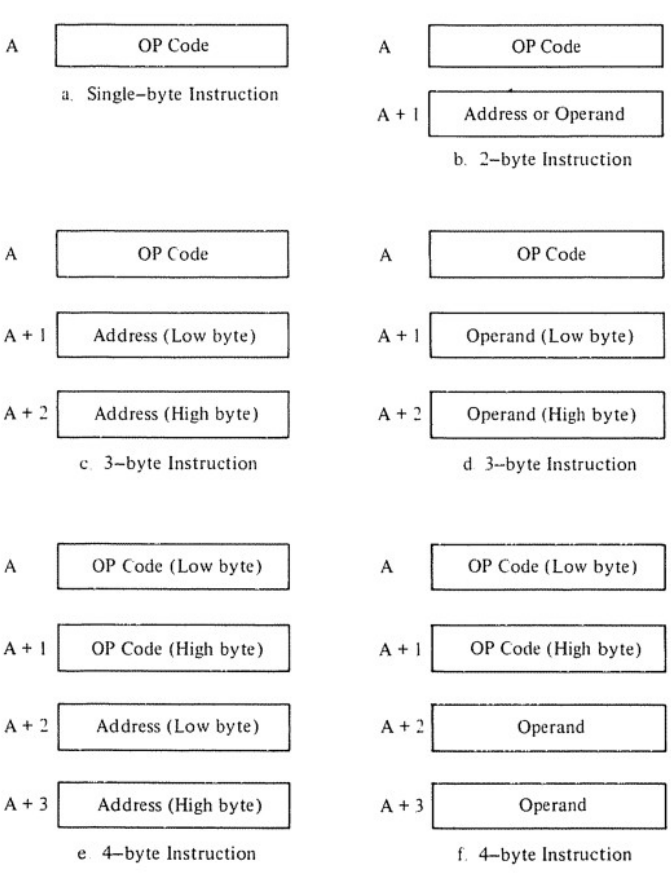


Figure 2-1 Generalized Z80 Instruction Formats

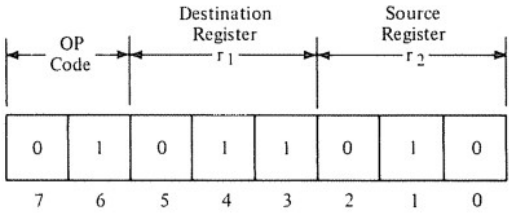


Figure 2-2 Instruction Format for Internal CPU Register Transfer

2-3.3 The Mnemonic Structure

1. The assembly language instructions for the Z80 are given in convenient mnemonic forms for simple understanding and use. Basically, the instruction mnemonics are structured in two parts as explained here:
   a. *The fixed part.* This is shown in capitals or upper-case letters. The fixed part generally defines the function of the operation that the instruction performs. The letters in this part are arranged such that they convey the operation performed very easily. For example, INC is used for increment, DEC for decrement, SUB for subtraction, and so on.
   b. *The variable part.* The variable part can consist of one or two operands or one or two addresses or displacements. The mnemonics for the variable

Table 2-2 INSTRUCTION SYMBOLS

Symbols	*Functions*
r	Denotes any register in the CPU GP main register set.
( )	Denotes that the contents of the register(s) are used as pointer to a memory location.
←	Denotes that the contents from the source are transferred to a destination. The tail of the arrow indicates the source and the head indicates the destination.
n	Represents an 8-bit integer whose decimal value ranges from 0 to 255, which is commonly expressed as ⟨0, 255⟩
nn	Represents a 16-bit integer whose decimal value ranges from 0 to 65, 535 commonly shown as ⟨0, 65535⟩
dd	Represents any register pairs BC, DE, HL, SP.
qq	Represents any register pairs AF, BC, DE, HL.
$(PAIR)_H$	Represents the high-order byte of the register pair indicated in the parentheses, e.g., BC_H.
b	When used as subscript denotes the bit number (000–111) to be set, reset, or tested in the indicated register.
$(PAIR)_L$	Represents the low-order byte of the register pair indicated in the parentheses, e.g., DE_L.
	Note: H and L when used as subscripts mean high- and low-order bytes of any register pair. They *should not* be confused with H and L registers in the CPU.
↔	A two-headed arrow indicates that the contents of the two registers are exchanged.
+	Means simple arithmetic add.
−	Means simple arithmetic subtract.
+ 1	Means increment (i.e., add 1).
− 1	Means decrement (i.e., subtract 1).
=	Means equal to.
≠	Means not equal to.
s	Any 8-bit location for all addressing modes allowed for that particular instruction.
ss	Any 16-bit location for all addressing modes allowed for that particular instruction.
∧	Indicates logical AND operation.
∨	Indicates logical OR operation.
⊻	Indicates logical Exclusive-OR operation.
CY	Indicates carry flip-flop.
$\overline{CY}$	Indicates complement carry flip-flop.
← →	Indicates a register shift or rotate operations. Arrow indicates the direction of shift or rotate.
CY ← □ Or □ → CY	Indicates register shift or rotate with carry flip-flop. Arrow indicates direction of shift or rotate.

part can be expressed by either the upper- or the lower-case letters. Generally, upper-case letters are used to indicate the internal CPU registers and lower-case letters are used to identify other locations, such as I/O devices or external memories. If two operands or displacements are involved in the variable part, then they are separated by a comma.

c. *Implied operand.* If the instructions are such that

they contain implied operands, then the mnemonics for such instructions do not have variable parts.

2. If the operand is enclosed by parentheses, it means that the contents of that location (usually a GP register) are used as a pointer to a memory location where the desired operand is stored or is to be stored.
3. The mnemonics are structured such that the fixed part of the instruction is followed by the destination (if a transfer is involved) which is followed by a comma. The source of the operand is the last item in the mnemonic structure.
4. Following the guidelines in paragraph 3, the assembly language instruction for the example mentioned in Sec. 2-3.2 would be written as follows:

   LD r_1, r_2

   where LD is the mnemonic for the load operation, r_1 represents the destination register (register E in this case), and r_2 represents the source (register D in this case). Of course when these mnemonics are translated into machine language by the assembler, they will appear as shown in Fig. 2-2.
5. Suppose we want to store the contents of accumulator A in a memory location whose address pointer is located in register pair BC. The assembly language instruction for this operation would be as follows:

   LD (BC), A

Table 2-3 STATUS FLAG SYMBOLS

Symbols	*Effect on Flags*
–	Flag unchanged by the operation.
*	Flag affected according to the result of the operation.
1	Flag is set by the operation.
0	Flag is reset by the operation.
X	Flag is in a "don't care" status.
P	Flag is set according to parity of the result of the operation.
V	Flag is set according to the overflow resulting from the operation.
/	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag.
$	P/V flag is 0 if the result of BC − 1 = 0. Otherwise, P/V = 1.
¢	Z flag is 1 if A = (HL). Otherwise, Z = 0.
?	Flag is unknown.
@	If B − 1 = 0, Z flag is set. Otherwise it is reset.

2-3.4 Symbolic Expressions of Operations and Functions

All the instructions in the Z80 set can be symbolically expressed. Before we describe and explain the actual instruction set, the symbols used are presented in Table 2-2.

2-3.5 The Status Register Symbols

Many instructions, when executed, give results whose status is stored in the appropriate flag or flip-flop in the status register F or F′. In our descriptions of the various instructions we include an indication of which status flags are affected, if any, and how they are affected. The symbols used to indicate the effect on each of the six status flags are given in Table 2-3.

Table 2-4 THE r, r′ CODES

Register	*r or r′ Binary Codes*
A	111
B	000
C	001
D	010
E	011
H	100
L	101

Table 2-5 THE qq CODES

Register Pair	*qq Binary Codes*
BC	00
DE	01
HL	10
AF	11

Table 2-6 THE dd CODES

Register Pair	*dd Binary Codes*
BC	00
DE	01
HL	10
SP	11

Table 2-7 THE ss CODES

Register Pair	*ss Binary Codes*
BC	00
DE	01
HL	10
SP	11

Table 2-8 THE pp CODES

Register Pair	*pp Binary Codes*
BC	00
DE	01
IX	10
SP	11

Table 2-9 THE rr CODES

Register Pair	*rr Binary Codes*
BC	00
DE	01
IY	10
SP	11

Table 2-10 BIT IDENTIFICATION CODES

Bit No.	*b Binary Codes*
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

2-3.6 The Register Pair Codes

In the Z80 instructions which are given in tabular form in the rest of this chapter (as well as in Chapters 3 and 4), the registers in the CPU that are involved in the particular instructions are identified in the OP codes by lower-case letters(s). The binary codes that identify the register or the register pairs are given in the following tables.

Additionally, in the bit set, rest, and test group of instructions, the b code identifies the bit number that is involved. Table 2-10 gives the binary codes for the bits involved in the execution of these instructions.

2-4 THE INSTRUCTION SET

See Tables 2-11 through 2-17.

Table 2-11 8-BIT LOAD INSTRUCTIONS GROUP

No.	Mnemonics	Symbolic Operations	Status Flags						OP Code			Cycles/ States		Operation Description
			C	Z	P/V	S	N	H	76	543	210	M	T	
1.	LD r_1, r_2	$r_1 \longleftarrow r_2$	—	—	—	—	—	—	01	r_1	r_2	1	4	Contents of r_2 loaded into r_1.
2.	LD r, n	r ← n	—	—	—	—	—	—	00 ←	r n	110 →	2	7	8-bit integer is loaded into register r.
3.	LD r, (HL)	r ←(HL)	—	—	—	—	—	—	01	r	110	2	7	8-bit contents of memory location HL are loaded into register r.
4.	LD r, (IX + d)	r ← (IX + d)	—	—	—	—	—	—	11 01 ←	011 r d	101 110 →	5	19	Contents of memory address, formed by sum of IX and displacement d, are loaded into register r.
5.	LD r, (IY + d)	r ← (IY + d)	—	—	—	—	—	—	11 01 ←	111 r d	101 110 →	5	19	Contents of memory address, formed by sum of IY and displacement d, are loaded into register r.
6.	LD (HL), r	(HL) ← r	—	—	—	—	—	—	01	110	r	2	7	Contents of r are loaded into memory location specified by the HL register pair.
7.	LD (IX + d), r	(IX + d) ← r	—	—	—	—	—	—	11 01 ←	011 110 d	101 r →	5	19	Contents of r are loaded into memory location obtained by adding index register IX and displacement d.
8.	LD (IY + d), r	(IY + d) ← r	—	—	—	—	—	—	11 01 ←	111 110 d	101 r →	5	19	Contents of r are loaded into memory location obtained by adding index register IY and displacement d.
9.	LD (HL) n	(HL) ← n	—	—	—	—	—	—	00 ←	110 n	110 →	3	10	Integer n is loaded into memory location specified by register pair HL.
10.	LD (IX + d), n	(IX + d) ← n	—	—	—	—	—	—	11 00 ← ←	011 110 d n	101 110 → →	5	19	Operand n is loaded into memory location specified by the sum of IX and the displacement d.
11.	LD (IY + d), n	(IY + d) ← n	—	—	—	—	—	—	11 00 ← ←	111 110 d n	101 110 → →	5	19	Operand n is loaded into memory location specified by the sum of IY and the displacement d.

Table 2-11 (*continued*)

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
12.	LD A, (BC)	A ⟵ (BC)	—	—	—	—	—	—	00 001 011	2	7	Contents of memory location specified by the contents of BC registers are loaded into the accumulator.
13.	LD A, (DE)	A ⟵ (DE)	—	—	—	—	—	—	00 011 010	2	7	Contents of memory location specified by the contents of DE registers are loaded into the accumulator.
14.	LD A, (nn)	A ⟵ (nn)	—	—	—	—	—	—	00 111 010 ← n → ← n →	4	13	Contents of memory location specified by the contents of operands nn are loaded into the accumulator.
15.	LD (BC), A	(BC) ⟵ A	—	—	—	—	—	—	00 000 010	2	7	Contents of the accumulator are loaded into memory location specified by the contents of register pair BC.
16.	LD (DE), A	(DE) ⟵ A	—	—	—	—	—	—	00 010 010	2	7	Contents of accumulator are loaded into memory location specified by contents of register pair DE.
17.	LD (nn), A	(nn) ⟵ A	—	—	—	—	—	—	00 110 010 ← n → ← n →	4	13	Contents of accumulator are loaded into memory location specified by the contents of operand nn.
18.	LD A, I	A ⟵ I	—	*	/	—	—	—	11 101 101 01 010 111	2	9	Contents of interrupt vector register I are loaded into the accumulator.
19.	LD A, R	A ⟵ R	—	*	/	—	—	—	11 101 101 01 011 111	2	9	Contents of memory refresh register R are loaded into the accumulator.
20.	LD I, A	I ⟵ A	—	—	—	—	—	—	11 101 101 01 000 111	2	9	Contents of the accumulator are loaded into the interrupt vector register I.
21.	LD R, A	R ⟵ A	—	—	—	—	—	—	11 101 101 01 001 111	2	9	Contents of the accumulator are loaded into memory refresh register R.

Table 2-12 16-BIT LOAD INSTRUCTIONS GROUP

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
22.	LD dd, nn	dd ⟵ nn	—	—	—	—	—	—	00 dd0 001 ← n → ← n →	3	10	2-byte integer nn is loaded into dd register in the CPU.
23.	LD IX, nn	IX ⟵ nn	—	—	—	—	—	—	11 011 101 00 100 001 ← n → ← n →	4	14	2-byte integer nn is loaded into the index register IX.
24.	LD IY, nn	IY ⟵ nn	—	—	—	—	—	—	11 111 101 00 100 001 ← n → ← n →	4	14	2-byte integer nn is loaded into the index register IY.
25.	LD HL, (nn)	H ← (nn + 1) L ← (nn)	—	—	—	—	—	—	00 101 010 ← n → ← n →	5	16	Contents of memory location (nn) are loaded into L register and contents of next higher memory location (nn + 1) are loaded into the H register.
26.	LD dd, (nn)	dd_H ← (nn + 1) dd_L ← (nn)	—	—	—	—	—	—	11 101 101 01 dd1 011 ← n → ← n →	6	20	Contents of memory location (nn) are loaded into the low-order position of dd and the contents of next higher location (nn + 1) are loaded into the higher-order position of the dd register pair.
27.	LD IX, (nn)	IX_H ← (nn + 1) IX_L ← (nn)	—	—	—	—	—	—	11 011 101 00 101 010 ← n → ← n →	6	20	Contents of memory location (nn) are loaded into low-order byte index register IX, and contents of next higher location (nn + 1) are loaded into the high-order byte of IX register pair.

Table 2-12 (*continued*)

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
28.	LD IY, (nn)	$IY_H \leftarrow (nn + 1)$ $IY_L \leftarrow (nn)$	—	—	—	—	—	—	11 111 101 00 101 010 ← n → ← n →	6	20	Contents of memory location (nn) are loaded into low-order byte of index register IY; contents of higher location (nn + 1) are loaded into high-order byte of IY.
29.	LD (nn), HL	$(nn + 1) \leftarrow H$ $(nn) \longleftarrow L$	—	—	—	—	—	—	00 100 010 ← n → ← n →	5	16	Contents of low byte of pair HL are loaded into memory location (nn); contents of high bite of H are loaded into next higher location (nn + 1).
30.	LD (nn), dd	$(nn + 1) \leftarrow dd_H$ $(nn) \longleftarrow dd_L$	—	—	—	—	—	—	11 101 101 01 dd0 011 ← n → ← n →	6	20	Low byte of register pair dd is loaded into memory location (nn); high bite of dd is loaded into (nn + 1).
31.	LD (nn), IX	$(nn + 1) \leftarrow IX_H$ $(nn) \longleftarrow IX_L$	—	—	—	—	—	—	11 011 101 00 100 010 ← n → ← n →	6	20	Low byte of index register IX is loaded into memory location (nn); high byte of IX is loaded into (nn + 1).
32.	LD (nn), IY	$(nn + 1) \leftarrow IY_H$ $(nn) \longleftarrow IY_L$	—	—	—	—	—	—	11 111 101 00 100 010 ← n → ← n →	6	20	Low byte of index register IY is loaded into memory location (nn); high byte of IY is loaded into (nn + 1).
33.	LD SP, HL	$SP \longleftarrow HL$	—	—	—	—	—	—	11 111 001	1	6	Contents of register pair HL are loaded into the stack pointer SP.
34.	LD SP, IX	$SP \longleftarrow IX$	—	—	—	—	—	—	11 011 101 11 111 001	2	10	Contents of index register IX are loaded into the stack pointer SP.
35.	LD SP, IY	$SP \longleftarrow IY$	—	—	—	—	—	—	11 111 101 11 111 001	2	10	Contents of index register IY are loaded into the stack pointer SP.

Table 2-13 STACK PUSH/POP, 16-BIT, INSTRUCTIONS GROUP

No.	*Mnemonics*	*Symbolic Operations*	*Status Flags*						*OP Code*		*Cycles/ States*		*Operation Description*
			C	*Z*	*P/V*	*S*	*N*	*H*	*76*	*543 210*	*M*	*T*	
36.	PUSH qq	$(SP - 1) \leftarrow qq_H$ $(SP - 2) \leftarrow qq_L$	—	—	—	—	—	—	11	qq0 101	3	11	Stack pointer is first decremented and high byte of registers qq is pushed into this location. SP is again decremented and low byte is pushed into this location.
37.	PUSH IX	$(SP - 1) \leftarrow IX_H$ $(SP - 2) \leftarrow IX_L$	—	—	—	—	—	—	11 11	011 101 100 101	4	15	Stack pointer is decremented and high byte of IX pair is pushed into this location. SP is decremented again and low byte of IX is pushed into this location
38.	PUSH IY	$(SP - 1) \leftarrow IY_H$ $(SP - 2) \leftarrow IY_L$	—	—	—	—	—	—	11 11	111 101 100 101	4	15	Stack pointer is decremented and high byte of IY pair is pushed into this location. SP is decremented again and low byte of IY is pushed into this location.
39.	POP qq	$qq_L \longleftarrow (SP)$ $qq_H \leftarrow (SP + 1)$	—	—	—	—	—	—	11	qq0 001	3	10	Contents of the stack (low byte) are loaded into register pair qq low byte. The SP is then incremented and contents of this location are loaded into qq high byte. The SP is incremented once more.
40.	POP IX	$IX_L \longleftarrow (SP)$ $IX_H \leftarrow (SP + 1)$	—	—	—	—	—	—	11 11	011 101 000 001	4	14	Contents of stack are loaded in low byte of IX. SP is then incremented and contents of this location are loaded in the high byte of IX. The SP is incremented once more.
41.	POP IY	$IY_L \longleftarrow (SP)$ $IY_H \leftarrow (SP + 1)$	—	—	—	—	—	—	11 11	111 101 100 001	4	14	Contents of stack are loaded in low byte of IY. SP is then incremented and contents of this location are loaded in the high byte of IY. The SP is incremented once more.

Table 2-14 EXCHANGE INSTRUCTIONS GROUP

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
42.	EX DE, HL	DE ⟷ HL	—	—	—	—	—	—	11 101 011	1	4	Contents of register pairs DE and HL are exchanged.
43.	EX AF, A′F′	AF ⟷ A′F′	—	—	—	—	—	—	00 001 000	1	4	Contents of register pairs AF and A′F′ are exchanged.
44.	EXX	BC ⟷ B′C′ DE ⟷ D′E′ HL ⟷ H′L′	—	—	—	—	—	—	11 011 001	1	4	Contents of BC and B′C′ are exchanged, contents of DE and D′E′ are exchanged, and contents of HL and H′L′ are exchanged.
45.	EX (SP), HL	L ⟷ (SP) H ↔ (SP + 1)	—	—	—	—	—	—	11 100 011	5	19	Low byte of HL pairs is exchanged with the location specified by SP. SP is then incremented and the high byte of HL is exchanged with the memory location specified by SP + 1.
46.	EX (SP), IX	IX_L ⟷ (SP) IX_H ↔ (SP + 1)	—	—	—	—	—	—	11 011 101 11 100 011	6	23	Low byte of index register IX is exchanged with the contents of memory location specified by SP. IX is then exchanged with the location specified by SP + 1.
47.	EX (SP), IY	IY_L ⟷ (SP) IY_H ↔ (SP + 1)	—	—	—	—	—	—	11 111 101 11 100 011	6	23	Low byte of index register IY is exchanged with the contents of memory location specified by SP. SP is then incremented. High byte of IY is then exchanged with the location specified by SP + 1.

Table 2-15 BLOCK TRANSFER AND SEARCH INSTRUCTIONS GROUP

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
48.	LDI	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1	—	—	* $	—	0	0	11 101 101 10 100 000	4	16	Contents of memory locations specified by HL are transferred to memory location specified by DE. Both HL and DE are incremented and BC (byte counter) is decremented.
49.	LDIR	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1 Repeat until BC = 0.	—	—	0	—	0	0	11 101 101 10 110 000	5 (if BC ≠ 0) 4 (if BC = 0)	21 16	The LDI instruction is repeated. If BC = 0 then the instruction is terminated. If BC ≠ 0, then PC is decremented by 2 and the instruction is repeated.
50.	LDD	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1	—	—	* $	—	0	0	11 101 101 10 101 000	4	16	Contents of memory locations specified by HL are transferred to memory location specified by DE. The HL, DE, and BC are decremented.
51.	LDDR	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1 Repeat until BC = 0.	—	—	0	—	0	0	11 101 101 10 111 000	5 (if BC ≠ 0) 4 (if BC = 0)	21 16	The LDD instruction above is repeated. If BC = 0, then the instruction is terminated. If BC ≠ 0, PC is decremented by 2 and the instruction is repeated.

Table 2-15 (*continued*)

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
52.	CPI	A − (HL) HL ← HL + 1 BC ← BC + 1	—	* ¢	* $	*	*	*	11 101 101 10 100 001	4	16	Contents of memory location specified by HL are compared with accumulator. If A = HL, Z is set, HL is incremented, and BC is decremented.
53.	CPIR	A − (HL) HL ← HL + 1 BC ← BC − 1 Repeat until A = HL or BC = 0	—	* ¢	* $	*	*	*	11 101 101 10 110 001	5 (if BC ≠ 0 & (HL) ≠ A) 4 (if BC = 0 or A = (HL)	21 16	The CPI instruction is repeated. If A = (HL), i.e., a match is found, the instruction is terminated. If BC ≠ 0, then PC is decremented by 2 and the instruction is repeated.
54.	CPD	A − (HL) HL ← HL − 1 BC ← BC − 1	—	* ¢	* $	*	*	*	11 101 101 10 101 001	4	16	Contents of memory location specified by HL are compared with the contents of the accumulator. If A = HL, the Z flag is set. Then both HL and BC are decremented.
55.	CPDR	A − (HL) HL ← HL − 1 BC ← BC − 1 Repeat until A = (HL) or BC = 0.	—	* ¢	* $	*	*	*	11 101 101 10 111 001	5 (If BC ≠ 0 & A ≠ (HL) 4 (If BC = 0 or A = (HL)	21 16	The CPD instruction above is repeated. If A = (HL), i.e., a match is found, the instruction is terminated. If both BC ≠ 0 and A ≠ HL, then the PC is decremented by 2 and the instruction is repeated.

Table 2-16 8-BIT ARITHMETIC AND LOGIC INSTRUCTIONS GROUP

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
56.	ADD A, r	A ⟵ A + r	*	*	V	*	0	*	10 000 r	1	4	Contents of accumulator and register r are added and the result is stored in the accumulator.
57.	ADD A, n	A ⟵ A + n	*	*	V	*	0	*	11 000 110 ← n →	2	7	Integer n is added to the accumulator and the result is stored in the accumulator.
58.	ADD A, (HL)	A ⟵ A + (HL)	*	*	V	*	0	*	10 000 110	2	7	Contents of memory location specified by the contents of HL are added to the accumulator and the result is stored in the accumulator.
59.	ADD A, (IX+d)	A ← A + (IX+d)	*	*	V	*	0	*	10 011 101 10 000 110 ← d →	5	19	Addition of displacement d and the contents of index register IX point to memory location whose contents are added to the accumulator. Result is stored in the accumulator.
60.	ADD A, (IY+d)	A ← A + (IY+d)	*	*	V	*	0	*	11 111 101 10 000 110 ← d →	5	19	Addition of displacement d and the contents of index register IY point to memory location whose contents are added to the accumulator. Result is stored in the accumulator.
61.	ADC A, s The M cycles and T states for each case are shown below: ADC A, r ---------- ADC A, n ---------- ADC A, (HL) ---- ADC A, (IX + d) ---------- ADC A, (IY + d) ----------	A ← A + s + CY	*	*	V	*	0	*	001 ---------- ---------- ---------- ---------- ----------	 1 2 2 5 5	 4 7 7 19 19	Operand s includes r, n, (HL), (IX + d), and (IY + d). s is added to the accumulator with carry and result is stored in the accumulator. (*Note:* Bits 5, 4, and 3, as shown here, are substituted for corresponding bits in the OP codes for instructions 56–60 for corresponding operations. In instruction 57 it is substituted in byte 1 but in instructions 59 and 60 it is substituted in byte 2.)

Table 2-16 (*continued*)

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
												Students should note that the explanation in instruction 61 above regarding bits 5, 4, and 3 in the op codes also applies to instructions 62–67.
62.	SUB s The M cycles and T states for each case are shown below:	A ⟵ A − s	*	*	V	*	1	*	010			Operand s (same as instruction 61) and the carry are subtracted from the accumulator and the result is stored in the accumulator.
	SUB r									1	4	
	SUB n									2	7	
	SUB (HL)									2	7	
	SUB (IX + d)									5	19	
	SUB (IY + d)									5	19	
63.	SBC A, s The M cycles and T states for each case are shown below:	A ← A − s − CY	*	*	V	*	1	*	011			Operand s (same as instruction 61) and the carry are subtracted from the accumulator and the result is stored in the accumulator.
	SBC A, r									1	4	
	SBC A, n									2	7	
	SBC A, (HL)									2	7	
	SBC A, (IX + d)									5	19	
	SBC A, (IY + d)									5	19	
												For instructions 61–63, construct the OP code as shown in instructions 56–60 but with appropriate changes in bits 5, 4, and 3 as shown above. For index registers changes are only made in the second byte. *SEE SEC. 2-5 FOR EXAMPLES OF HOW THESE INSTRUCTIONS ARE CONSTRUCTED.*

Table 2-16 (*continued*)

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
64.	AND s	A ← A∧ s	0	*	P	*	0	1	100			Operand s (same as instruction 61) is ANDed with the accumulator and the result is stored in the accumulator.
	The M cycles and T states for each case are shown below:											
	AND r									1	4	
	AND n									2	7	
	AND (HL)									2	7	
	AND (IX + d)									5	19	
	AND (IY + d)									5	19	
65.	OR s	A ← A ∨ s	0	*	P	*	0	1	110			Operand s (same as instruction 61)is ORed with the accumulator and the result is stored in the accumulator.
	The M cycles and T states for each case are shown below:											
	OR r									1	4	
	OR n									2	7	
	OR (HL)									2	7	
	OR (IX + d)									5	19	
	OR (IY + d)									5	19	
66.	XOR s	A ← A ∀ s	0	*	P	*	0	1	101			Operand s (same as instruction 61) is exclusive-ored with the accumulator and the result is stored in the accumulator.
	The M cycles and T states for each case are shown below:											
	XOR r									1	4	
	XOR n									2	7	
	XOR (HL)									2	7	
	XOR (IX + d)									5	19	
	XOR (IY + d)									5	19	
67.	CP s	A − s	*	*	V	*	1	*	111			Operand s (same as instruction 61) is compared with the accumulator. If A = s, then the Z flag is set. The accumulator contents are not altered.
	The M cycles and T states for each case are shown below:											
	CP r									1	4	
	CP n									2	7	
	CP (HL)									2	7	
	CP (IX + d)									5	19	
	CP (IY + d)									5	19	

Table 2-16 (*continued*)

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
68.	INC r	r ⟵ r + 1	—	*	V	*	0	*	00 r 100	1	4	Contents of r are incremented.
69.	INC (HL)	(HL) ← (HL) + 1	—	*	V	*	0	*	00 110 100	3	11	Contents of memory location addressed by HL are incremented.
70.	INC (IX + d)	(IX + d) ⟵ (IX + d) + 1	—	*	V	*	0	*	11 011 101 00 110 100 ← d →	6	23	Displacement d is added to the contents of the index register IX. The contents of the memory addressed by this sum are incremented.
71.	INC (IY + d)	(IY + d) ⟵ (IY + d) + 1	—	*	V	*	0	*	11 111 101 00 110 100 ← d →	6	23	Displacement d is added to the contents of the index register IY. The contents of the memory addressed by this sum are incremented.
72.	DEC m	m ⟵ m − 1	—	*	V	*	1	*	101	1	4	Operand m includes r, (HL), (IX + d), and (IY + d). The contents of operand m are decremented. (*Note:* Bits 2, 1, and 0, as shown here, are substituted for corresponding bits in the op codes of instructions 68–71 for corresponding operations. In instructions 68–69, it is substituted in byte 1, but in instructions 70–71, it is substituted in byte 2.

Table 2-17 16-BIT ARITHMETIC INSTRUCTIONS GROUP

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
73.	ADD HL, ss	HL ⟵ HL + ss	*	—	—	—	0	?	00 ss1 001	3	11	Contents of register pair ss are added to HL and the result is stored in HL.
74.	ADC HL, ss	HL←HL+ss+CY	*	*	V	*	0	?	11 101 101 01 ss1 010	4	15	Contents of register pair ss and Carry are added to HL and the result is stored in HL.
75.	SBC HL, ss	HL←HL−ss−CY	*	*	V	*	0	?	11 101 101 01 ss0 010	4	15	Contents of register pair ss and Carry are subtracted from HL and the result is stored in HL.
76.	ADD IX, pp	IX ⟵ IX + pp	*	—	—	—	0	?	11 011 101 00 pp1 001	4	15	Contents of register pair pp are added to the index register IX and the result is stored in IX.
77.	ADD IY, rr	IY ← IY + rr	*	—	—	—	0	?	11 111 101 00 rr1 001	4	15	Contents of register pair rr are added to the index register IY and the result is stored in IY.
78.	INC ss	ss ← ss + 1	—	—	—	—	—	—	00 ss0 011	1	6	Contents of register pair ss are incremented.
79.	INC IX	IX ← IX + 1	—	—	—	—	—	—	11 011 101 00 100 011	2	10	Contents of index register IX are incremented.
80.	INC IY	IY ← IY + 1	—	—	—	—	—	—	11 111 101 00 100 011	2	10	Contents of index register IY are incremented.
81.	DEC ss	ss ⟵ ss − 1	—	—	—	—	—	—	00 ss1 011	1	6	Contents of pair ss are decremented.
82.	DEC IX	IX ⟵ IX − 1	—	—	—	—	—	—	11 011 101 00 101 011	2	10	Contents of index register IX are decremented.
83.	DEC IY	IY ⟵ IY − 1	—	—	—	—	—	—	11 111 101 00 101 011	2	10	Contents of index register IY are decremented.

2-5 CONSTRUCTING COMPLEX INSTRUCTION CODES

The following examples are intended to demonstrate how specific mnemonics and tables are used to construct the OP Codes for instructions 61-81, previously shown in Tables 2-16 and 2-17.

Example 2-1

The contents of register D are to be added to the contents of register A and the result stored in register A. Show how you would construct the mnemonics and the OP code for this operation.

Solution

From Table 2-16 we find that we can use instruction 56, whose generalized mnemonics (or source code) is ADD A, r, and the OP code is

76 543 210
10 000 r

From Table 2-9 for the r r'codes, we find that the binary code for register D is 010. We substitute this in the general formula above and get

76 543 210
10 000 010

The complete source code, and the OP code (both in binary and hex), are as follows:

Mnemonics (Source Code)	*OP Code (Binary)*	*OP Code (Hex)*
ADD A, D	10 000 010	82

Example 2-2

The contents of memory location, whose address is contained in register pair HL, are to be added to the contents of the accumulator with carry. Show how the source and the OP code for this operation are constructed.

Solution

Instruction 61 of Table 2-16 is used here. The generalized mnemonic code is ADC A, s. Here (HL) is substituted for s, and the code is ADC A, (HL). Following the operation description of instruction 61, we select instruction 58, since (HL) is involved. However, we will substitute code 001 for bits 5, 4, and 3 of instruction 58.

Instruction 58 10 *000* 110
Instruction 61 10 *001* 110

Mnemonics (Source Code)	*OP Code (Binary)*	*OP Code (Hex)*
ADC A, (HL)	10 001 110	8E

Example 2-3

The contents of memory location, whose effective address is specified by the sum of the contents of index register Y and the displacement d, are to be subtracted from the contents of register A with carry. Show how the code(s) is (are) constructed for this operation if the displacement d = 14 hex.

Solution

Instruction 63, SBC A, s is used in this case. However, since index register Y and displacement are involved, the OP codes for instruction 60 are used. Bits 5, 4, and 3 of byte 2 of this instruction are replaced by bits 011. Therefore, the following codes are obtained for this operation:

Mnemonics (Source Code)	*OP Code (Binary)*	*OP Code (Hex)*
SBC A, (IY + d)	11 111 101	FD
	10 011 110	9E
	00 010 100	14

Example 2-4

The contents of register L are EXCLUSIVE-ORed with the contents of register A and the result is loaded in register A. Write the codes for this operation.

Solution

Instruction 66 is used for this operation. From Table 2-4, the code for register L is 101. However, instruction 66 gives us only the 5, 4, and 3 bits of the OP code. We notice that the rest of the bits for the OP code can come from instruction 56, which is 10 000 r. Using this we construct the OP code as follows:

Mnemonics (Source Code)	*OP Code (Binary)*	*OP Code (Hex)*
XOR L	10 101 101	AD

Example 2-5

The contents of the memory location, whose effective address is given by the sum of IX + d, are decremented. Construct the OP codes for this operation.

Solution

Instruction 72 is used here. We will substitute 101 in bits 2, 1, and 0 of byte 2 of instruction 70 to obtain the following:

Mnemonics (Source Code)	*OP Code (Binary)*	*OP Code (Hex)*
DEC (IX + d)	11 011 101	DD
	00 110 101	35
	← d →	?? ■

2-6 REVIEW QUESTIONS

2-1 Indicate true or false for the Load and Exchange instructions group.

a. ____ Involves both 8-bit and 16-bit data transfers.

b. ____ Involves data transfers between internal registers of the CPU.

c. ____ Does not involve data transfers between the CPU and the external memory.

d. ____ The transfers destroy the data in the source locations.

e. ____ Instructions specify both the source and destination.

2-2 The load instructions include _____ instructions which allow the operand contained in the instruction to be transferred. (load immediate; implied mode; indexed mode; relative mode)

2-3 The following statements relate to the block transfer and search instructions group. Indicate true or false.

a. ____ Block of data can be transferred from one memory location to another.

b. ____ Block transfers require as many instructions as there are bytes in the block.

c. ____ An entire block of data of any length can be searched by just one instruction.

d. ____ When the desired word is found, the search is automatically terminated.

e. ____ If the desired word is not found, the search is continued in the next block.

2-4 Operands for arithmetic and logic operations are located in (circle all that apply):

a. The accumulator in the CPU.

b. The GP registers of the CPU.

c. The external I/O devices.

d. The external memory.

2-5 The results of the arithmetic/logic operations are placed in the _____. (index registers; GP registers; accumulator; instruction register)

2-6 The arithmetic/logic instructions of the Z80 are capable of performing (circle all that apply):

a. 8-bit subtractions.

b. 16-bit logical operations.

c. 16-bit additions.

d. 8-bit additions.

2-7 Indicate true or false for the Bit Manipulation group of instructions.

a. ____ Any bit in the accumulator can be set.

b. ____ Any bit in the CPU registers can be set or reset.

c. ____ Bits in the external memory can be set and reset.

d. ____ Bits in the external memory can be tested but not set or reset.

2-8 The following statements apply to the I/O group of instructions. Indicate true or false.

a. ____ Data can be transferred between the I/O devices and the external memory.

b. ____ A single instruction can transfer a block of data of a maximum of 512 bytes.

c. ____ Parity errors in serial transmissions *cannot* be detected.

d. ____ Additional operations (i.e., instructions) are needed to determine the status of the incoming data.

2-9 From the following statements pick out those that apply to the generalized instruction formats of the Z80.

a. The instruction could consist of 1, 2, 3, or 4 bytes.

b. If the OP code contains 2 bytes, then the following byte must always be an operand.

c. If the OP code contains 2 bytes, then the following byte must be an address byte.

d. In a 2-byte operand, the first byte is the high-order byte.

e. In a 2-byte address, the second byte is always the high-order byte.

f. The second byte of a 2-byte instruction could be either an operand or an address.

2-10 The contents of register E in the main set are to be loaded into register C of the same register set. Circle the correct OP code for this operation.

a. 00 001 011

b. 01 011 001

c. 11 000 101

d. 01 001 011

e. 10 011 001

f. 10 001 011

2-11 Refer to question 2-10 above. The correct assembly language mnemonic for this instruction is
- **a.** LD E, C
- **b.** LD C, E
- **c.** LD r_2, r_1
- **d.** LD r_1, r_2
- **e.** LD p, p
- **f.** LD q, q

2-12 The following statements relate to the Z80 mnemonic structure. Indicate true or false.
- **a.** ___ The fixed part of the mnemonic defines the function to be performed.
- **b.** ___ The fixed part is shown in capitals.
- **c.** ___ The variable part is generally shown in lower-case letters.
- **d.** ___ Two operands or displacements in the variable part are separated by a comma.
- **e.** ___ If the instruction contains an implied operand, then the mnemonic does not have a variable part.

2-13 The lower-case letter *n* represents _____ bit integer whose decimal value ranges from 0 to _____. (8; 16; 255; 256; 65,535; 65,536)

2-14 Any CPU register pairs BC, DE, HL, SP are represented by _____. (rr'; pp; nn; dd; qq)

2-15 The following statements relate to the mnemonic LD r, (HL). Indicate true or false.
- **a.** ___ Contents of HL are transferred to any GP register r.
- **b.** ___ Contents of HL are a memory address.
- **c.** ___ Contents of register r are transferred into HL pair.
- **d.** ___ Contents of memory addressed by HL are transferred to CPU register r.

2-16 We wish to load an integer *n* into the memory location whose address is obtained by adding a displacement to the contents of the index register IX. Pick out the proper symbolic representation for this operation.
- **a.** n ← IX + d
- **b.** IX + d ← n
- **c.** (IX + d) ← n
- **d.** d + IX ← n

2-17 The contents of the index register IY are pushed into the Stack. The correct symbolic representation for this operation is:
- **a.** (SP − 1) ← IY_H
  (SP−2)←IY_L
- **b.** (SP − 1) ← IY_L
  (SP − 2) ← IY_H
- **c.** SP ⟵ IY
- **d.** IY ⟵ SP

2-18 We want to exchange the contents of HL with the contents of HL'. The correct mnemonic for this instruction is
- **a.** LD HL, HL'
- **b.** LD HL', HL
- **c.** EXX
- **d.** EX HL', HL
- **e.** EX HL, HL'

2-19 We want to transfer the contents of HL to DE and then decrement both DE and HL. What are the applicable mnemonics for this operation?
- **a.** LDD
- **b.** LDH
- **c.** LDHD
- **d.** LDDH

2-20 Write down the symbolic representation for the compare mnemonic CPD.

2-21 We want to add (IX + d) to the accumulator and store the result in the accumulator. The correct mnemonic for this is:
- **a.** LD A, (IX + d)
- **b.** ADD A, (IX + d)
- **c.** LD (IX + d), A
- **d.** ADD (IX + d), A

2-22 The symbolic representation A ← A $\vee$ S means that
- **a.** Operand s is ORed with the accumulator.
- **b.** The result of the OR operation is stored in the accumulator.
- **c.** s is ANDed with the accumulator.

3

THE Z80 INSTRUCTION SET (PART TWO)

CHAPTER OBJECTIVES

The objectives of this chapter are as follows:

1. To present the ROTATE and SHIFT group of instructions.
2. To present the BIT MANIPULATION group of instructions which include the set, reset, and test operations.
3. To present the JUMP group of instructions which includes both conditional and unconditional JUMPS.
4. To present the CALL and RETURN group of instructions which includes both conditional and unconditional BRANCHES.
5. To explain arithmetic and logic shifts (right and left) as used in the Z80 CPU.

TEXTBOOK REFERENCES

For reviewing material in the textbook, relevant to the topics covered in this chapter, the following chapters and/or sections are suggested.

1. For ROTATE and SHIFT instructions	Sec. 4-2.5
2. For JUMP instructions and operations	Sec. 4-2.3.2
3. For CALL and RETURN instructions and operations	Sec. 4-2.3.3

3-1 INTRODUCTION

This chapter is a continuation of the Z80 instruction set which was introduced in Chapter 2. Four groups of instructions are covered in this chapter. The rest of the instructions are covered in Chapter 4. The instructions are presented in the same convenient tabular format that was used in Chapter 2. All the status flag symbols that are presented in Table 2-3 also apply here. Likewise, the instruction symbols presented in Table 2-2 also apply to the instructions in this chapter.

3-2 SHIFT OPERATIONS

3-2.1 Arithmetic Shifts

3-2.1.1 Shift Left Arithmetic (SLA) In any computer, including μCs, operands involved in arithmetic operations must include a proper sign bit to identify the operand as either a positive or a negative quantity. The most significant bit (MSB) of the word is the sign bit. The convention for the sign bit is:

0 indicates a positive quantity
1 indicates a negative quantity

Arithmetic operations, by their very nature, require that the sign bit be preserved regardless of the shifting process. In the Z80 system when a left shift arithmetic is performed, either on a CPU register or on a word in the memory, the MSB of the word is shifted and saved in the Carry flag, i.e., the CY flip-flop. During such a shift, a 0 is automatically inserted in the vacated LSB position of the word. This operation is graphically shown in Fig. 3-01,

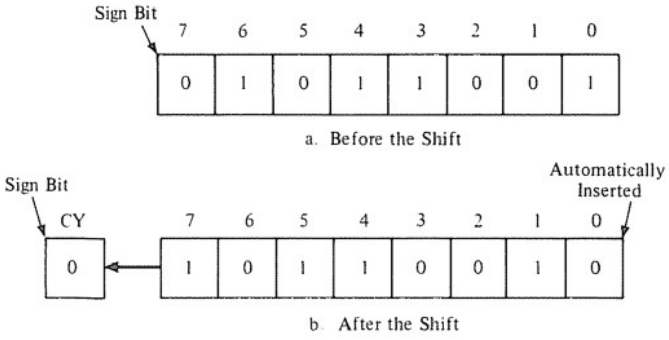


Figure 3-01 Shift Left Arithmetic (SLA)

which includes the contents of the register before and after the shift.

3-2.1.2 Shift Right Arithmetic (SRA) In this operation the sign bit is saved in the original bit 7 position and replicated in the next lower position of the register, i.e., bit 6 position. The LSB of the register (i.e., bit 0 position) is transferred into the CY flip-flop and saved there. Figure 3-02 shows the contents of the register (or memory location) before and after the shifting operation.

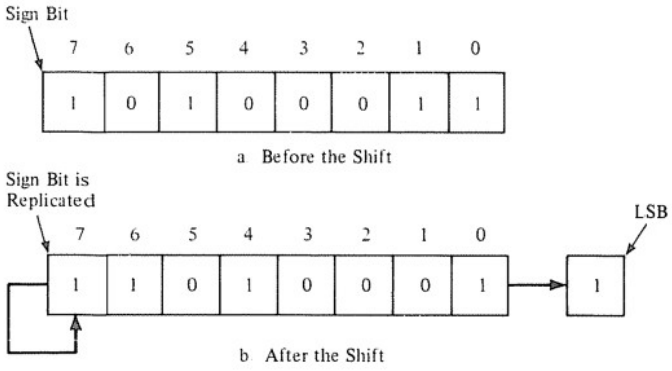


Figure 3-02 Shift Right Arithmetic (SRA)

3-2.1.3 Shift Right Logic (SRL) A word or an operand that is to be used in an arithmetic operation must have a sign bit associated with it. However, in a word to be used for a logical operation the sign bit is meaningless and so is not used. In a shift right logic operation in the Z80 system, a 0 is automatically inserted in the vacated MSB position (i.e., the bit 7 position) of the register. The LSB is shifted into the CY flip-flop and saved there. A typical example of this operation, before and after the shift, is shown in Fig. 3-03.

Note: In the Z80 system there is no instruction for shift left logic.

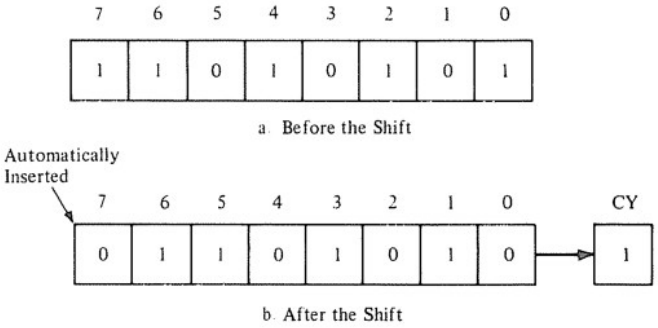


Figure 3-03 Shift Right Logic (SRL)

3-2.2 Register/Memory Symbols for Rotate/Shift

In describing the Rotate and Shift instructions, descriptive figures for the various registers and memory locations, involved in these operations and the carry flip-flop, are used. It is inconvenient to include these figures in the symbolic operations columns of the instructions in the tables that follow. We get around this problem by first giving all the figures in this section and assigning a figure number to each. We will then insert only the figure number in the appropriate column of the tables and we will include the identity of the register involved in the operation, such as A for the accumulator. In these tables note the following:

r denotes any register in the CPU main register set.
m denotes r, (HL), (IX + d), and (IY + d).
A denotes the accumulator.

3-2.3 The Mnemonic Convention

The following explanation is intended to help students with the proper interpretation of the mnemonics used in Table 3-1.

1. In instruction 84 to 94, the first letter R, means *rotate* or *recirculate.*
2. The second letter, L, means *left* and the second letter, R, means *right.*
3. The third letter, C, whenever it appears, means that the rotation takes place on the *contents* of the register (or the memory location) only. However, the end bit of the word, either the LSB or the MSB, is shifted into the CY flip-flop.
4. The third or the fourth letter, A, refers to the *accumulator* while letters r and m refer to the registers or memory locations which were defined in Section 3-2.2.

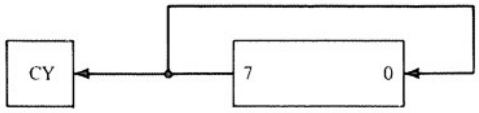


Figure 3-1

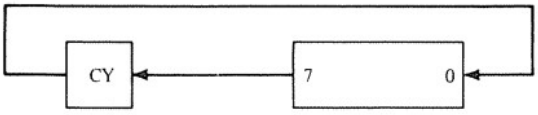


Figure 3-2

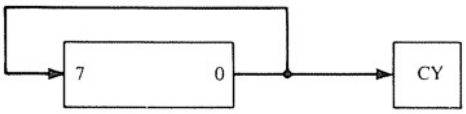


Figure 3-3

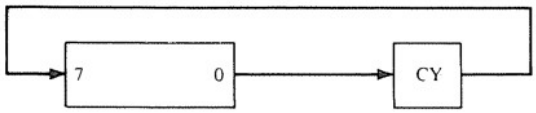


Figure 3-4

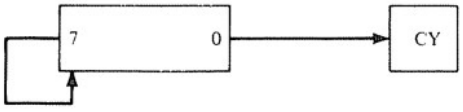


Figure 3-5

Symbols for Shift/Rotate Operations

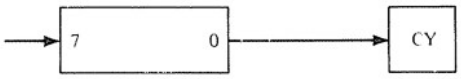


Figure 3-6

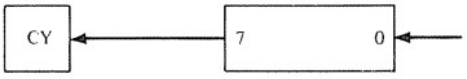


Figure 3-7

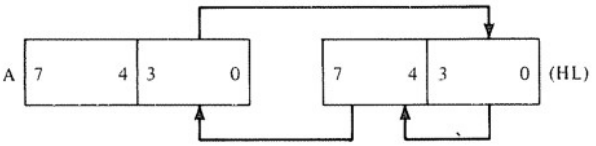


Figure 3-8

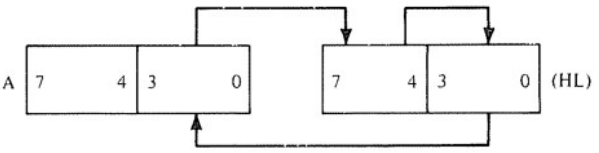


Figure 3-9

5. For instructions 95, 96, and 97, the interpretation of the mnemonics for the three capital letters is as follows:
   a SLA means *shift left arithmetic.*
   b. SRA means *shift right arithmetic.*
   c. SRL means *shift right logic.*
6. For instructions 98 and 99 operations descriptions, given in Table 3-1, give the proper interpretations.

3-2.4 OP Code Construction

In Table 3-1 notice that, for instructions 92–97, only 3 bits are given for the OP code. These 3 bit codes are intended to be substituted for the corresponding 3 bits (bits 5, 4, and 3) in the last byte of the related RLC instructions, i.e., instructions 88–91. The rest of the op code is the same. The following examples clarify how this is done.

Example 3-1

Write down the complete OP code for the RRC m instruction (instruction 93) for the case where the operand is stored in a location whose address is contained in the HL register pair of the CPU.

Solution

The final mnemonics for this instruction is

RRC (HL)

To get the OP code, we go to RLC (HL) whose OP code (from instruction 89) is

Byte 1	11 001 011
Byte 2	00 000 110

To get the appropriate op code for RRC (HL), we get the 3 bits 001 from instruction 93 in Table 3-1 and substitute them for bits 5, 4, and 3 in byte 2 above.

Thus, the correct OP code for RRC (HL) is

Byte 1	11 001 011
Byte 2	00 001 110

■

Example 3-2

Write down the complete OP code for the RR m instruction (instruction 94) for the situation where the operand is stored in a memory location whose address is obtained by adding the contents of the index register Y to the displacement in the instruction.

Solution

The final mnemonic for this instruction is RR (IY + d). To get the OP code, we go to RL_c(IY + d) (instruction 91) whose OP code is

Byte 1	11 111 101
Byte 2	11 001 011
Byte 3	← d →
Byte 4	00 000 110

To get the appropriate OP code for RR (IY + d), we get the 3 bits 011 from instruction 94 in Table 3-1 and substitute them for bits 5, 4, and 3 in byte 4 above. Thus, the correct OP code for RR (IY + d) is

Byte 1	11 111 101
Byte 2	11 001 011
Byte 3	← d →
Byte 4	00 011 110

■

3-3 ROTATE AND SHIFT GROUP

See Table 3.1.

3-4 BIT MANIPULATION GROUP

See Tables 3-2 and 3-4.

3-5 THE JUMP INSTRUCTIONS GROUP

Conditional JUMPS and conditional BRANCHES are performed by testing or sampling the conditions of the various flags in the Status Register. The conditions of the various flags are included in the instruction itself, and in the Z80 they are referred to as the cc codes. Table 3-3 below gives these codes and the conditions as well as the flags in the Status Register that are affected.

In the tables that follow Table 3-3, the byte denoted by e-2 is the displacement that is added to the program counter. In the mnemonics, it is labeled as just e. Here e ranges from −126 to +129.

3-6 THE CALL AND RETURN GROUP

The CALL and RETURN instructions are given in Table 3-6. Table 3-5 below gives the starting addresses for special subroutines which are stored in Page 0 of the memory. The 3-bit t code is inserted in the RES p instruction (instruction 126) of Table 3-6. In Table 3-5, the H in the P column indicates that the address is given in the hexadecimal code.

Table 3-3 CONDITIONAL JUMP/BRANCH CONDITIONS AND CODES

cc Code	*Status Flag*	*Condition*
000	Z	NZ Nonzero
001	Z	Z Zero
010	C	NC No carry
011	C	C Carry
100	P/V	PO Parity odd
101	P/V	PE Parity even
110	S	P Sign positive
111	S	M Sign negative

Table 3-5 STARTING ADDRESSES FOR SPECIAL SUBROUTINES

t Code	*p (Address in Page 0)*
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

Table 3-1 ROTATE AND SHIFT INSTRUCTIONS GROUP

No.	*Mnemonics*	*Symbolic Operations*		*Status Flags*						*OP Code*			*Cycles/ States*		*Operation Description*
				C	*Z*	*P/V*	*S*	*N*	*H*	*76*	*543*	*210*	*M*	*T*	
84.	RLCA	Fig. 3-1	A	*	–	–	–	0	0	00	000	111	1	4	Contents of accumulator are rotated left 1 bit. Sign bit is also transferred in CY flag.
85.	RLA	Fig. 3-2	A	*	–	–	–	0	0	00	010	111	1	4	Contents of accumulator, including the CY flag, are rotated left 1 bit.
86.	RRCA	Fig. 3-3	A	*	–	–	–	0	0	00	001	111	1	4	Contents of accumulator are rotated right one bit. Bit 0 is transferred into the CY flag.
87.	RRA	Fig. 3-4	A	*	–	–	–	0	0	00	011	111	1	4	Contents of accumulator, including CY flag, are rotated right 1 bit.
88.	RLC r	Fig. 3-1	r	*	*	P	*	0	0	11 00	001 000	011 r	2	8	Contents of register R are rotated left 1 bit. Bit 7 is also transferred to the CY flag.
89.	RLC (HL)	Fig. 3-1	(HL)	*	*	P	*	0	0	11 00	001 000	011 110	4	15	Contents of memory location addressed by HL are rotated left 1 bit. Bit 7 is also transferred into the CY flag.
90.	RLC (IX + d)	Fig. 3-1 (IX + d)		*	*	P	*	0	0	11 11 ← 00	011 001 d 000	101 011 → 110	6	23	Contents of memory location specified by the sum of index register IX and displacement d are rotated left 1 bit. Bit 7 is also transferred into the CY flag.

Table 3-1 (*continued*)

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
91.	RLC (IY + d)	Fig. 3-1	*	*	P	*	0	0	11 111 101 11 001 011 ← d → 00 000 110	6	23	Contents of memory location specified by sum of index register IY and displacement d are rotated left 1 bit. Bit 7 is also transferred to CY flag.
92.	RL m The M cycles and T states for each case are shown below:	Fig. 3-2 m	*	*	P	*	0	0	010			The m operand (as defined for RLC codes) is rotated left 1 bit, including the CY flag.
	RL r									2	8	
	RL (HL)									4	15	
	RL (IX + d)									6	23	
	RL (IY + d)									6	23	
93.	RRC m	Fig. 3-3 m	*	*	P	*	0	0	001			The m operands (as defined for RLC codes) is rotated right 1 bit. Bit 0 is also transferred to the CY flag.
	RRC r									2	8	
	RRC (HL)									4	15	
	RRC (IX + d)									6	23	
	RRC (IY + d)									6	23	
94.	RR m	Fig. 3-4 m	*	*	P	*	0	0	011			The operand m (as defined for RLC codes) is rotated right 1 bit, including the CY flag.
	RR r									2	8	
	RR (HL)									4	15	
	RR (IX + d)									6	23	
	RR (IY + d)									6	23	
95.	SLA m	Fig. 3-7 m	*	*	P	*	0	0	100			The m operand (as defined for RLC codes above) is shifted left 1 bit, including the CY flag.
	SLA r									2	8	
	SLA (HL)									4	15	
	SLA (IX + d)									6	23	
	SLA (IY + d)									6	23	

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
96.	SRA m	Fig. 3-5 m	*	*	P	*	0	0	101			Operand m (as defined for RLC codes above) is shifted right
	SRA r									2	8	1 bit. LSB is shifted into
	SRA (HL)									4	15	CY and MSB is replicated.
	SRA (IX + d)									6	23	
	SRA (IY + d)									6	23	
97.	SRL m	Fig. 3-6 m	*	*	P	*	0	0	111			Operand m (as defined for RLC codes above) is shifted right
	SRL r									2	8	1 bit. Bit 0 is shifted into
	SRL (HL)									4	15	the CY flag.
	SRL (IX + d)									6	23	
	SRL (IY + d)									6	23	
98.	RLD	Fig. 3-8 A, (HL)	–	*	P	*	0	0	11 101 101 01 101 111	5	18	Contents of low nibble of memory location specified by HL are loaded into the high nibble of the same location. High nibble of this memory location is loaded into the low nibble of the accumulator. Low nibble of the accumulator is loaded into the low nibble of the memory byte.
99.	RRD	Fig. 3-9 A, (HL)	–	*	P	*	0	0	11 101 101 01 100 111	5	18	Low nibble of the accumulator is loaded into the high nibble of the memory location specified by HL. High nibble of this memory location is shifted into the low nibble of the same location, and the low nibble of this location is shifted into the low nibble of the accumulator.

Table 3-2 BIT SET, RESET AND TEST INSTRUCTIONS GROUP

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76	543	210	Cycles/States M	T	Operation Description
100.	BIT b, r	$Z \longleftarrow \bar{r}_b$	–	*	?	?	0	1	11 01	001 b	011 r	2	8	The complement of bit b of register r is loaded into the Z flag.
101.	BIT b, (HL)	$Z \leftarrow \overline{(HL)_b}$	—	*	?	?	0	1	11 01	001 b	011 110	3	12	The complement of bit b of memory location specified by HL is loaded into the Z flag.
102.	BIT b, (IX + d)	$Z \leftarrow \overline{(IX+d)_b}$	—	*	?	?	0	1	11 11 ← 01	011 001 d b	101 011 → 110	5	20	The complement of bit b of memory location specified by (IX + d) is loaded into Z flag.
103.	BIT b, (IY + d)	$Z \leftarrow \overline{(IY+d)_b}$	—	*	?	?	0	1	11 11 ← 01	111 001 d b	101 011 → 110	5	20	The complement of bit b of memory location specified by (IY + d) is loaded into Z flag.
104.	SET b, r	$r_b \longleftarrow 1$	—	—	—	—	—	—	11 11	001 b	011 r	2	8	Bit b of operand register r is set.
105.	SET b, (HL)	$(HL)_b \leftarrow 1$	—	—	—	—	—	—	11 11	001 b	011 110	4	15	Bit b of memory location specified by HL is set.
106.	SET b, (IX + d)	$(IX + d)_b \leftarrow 1$	—	—	—	—	—	—	11 11 ← 11	011 001 d b	101 011 → 110	6	23	Bit b of memory location specified by (IX + d) is set.
107.	SET b, (IY + d)	$(IY + d)_b \leftarrow 1$	—	—	—	—	—	—	11 11 ← 11	111 001 d b	101 011 → 110	6	23	Bit b of memory location specified by (IY + d) is set.
108.	RES b, m	$s_b \longleftarrow 0$	—	—	—	—	—	—	10					Bit b of operand m (as defined by SET b, m above, instructions 104–107) is reset. (*Note:* The code in bits 7 and 6 replaces the corresponding bits 7 and 6 in the last byte of instructions 104–107.)
	RES r	----------							----------			4	8	
	RES (HL)	---------							----------			4	15	
	RES (IX + d)	----------							----------			6	23	
	RES (IY + d)	----------							----------			6	23	

NOTE: The codes for bit b are given in Table 2-10

Table 3-4 JUMP INSTRUCTIONS GROUP

No.	*Mnemonics*	*Symbolic Operations*	*Status Flags* C	Z	P/V	S	N	H	*OP Code* 76	543	210	*Cycles/ States* M	T	*Operation Description*
109.	JP nn	PC ⟵ nn	—	—	—	—	—	—	11	000	011	3	10	Operand nn, given in bytes 2 and 3 of the instruction, is loaded into the PC. This is an unconditional JUMP.
									←	n	→			
									←	n	→			
110.	JP cc, nn	If cc true, then PC ⟵ nn	—	—	—	—	—	—	11	cc	010	3	10	If condition cc is true, operand nn is loaded into PC. If not, PC is incremented.
									←	n	→			
									←	n	→			
111.	JR e	PC ← PC + e	—	—	—	—	—	—	00	011	000	3	12	Displacement e-2 is added to PC. This is an unconditional JUMP.
									←	e-2	→			
112.	JR C, e	If C = 0 then continue. If C = 1, then PC ← PC + e	—	—	—	—	—	—	00	111	000	2	7	If C flag is 0, then the program continues. If not, then the displacement e-2 is added to the PC. This is a conditional JUMP.
									←	e-2	→	3	12	
113.	JR NC, e	If C = 1, then continue. If C = 0, then PC ← PC + e	—	—	—	—	—	—	00	110	000	2	7	If C flag is 1, then the program continues. If not, then the displacement e-2 is added to the PC. This is a conditional JUMP.
									←	e-2	→	3	12	
114.	JR Z, e	If Z = 0, then continue. If Z = 1, then PC ← PC + e	—	—	—	—	—	—	00	101	000	2	7	If Z flag is 0, then the program continues. If not, then the displacement e-2 is aded to the PC. This is a conditional JUMP.
									←	e-2	→	3	12	

Table 3-4 (*continued*)

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
115.	JR NZ, e	If Z = 1, then continue. If Z = 0, then PC ← PC + e	—	—	—	—	—	—	00 100 000 ← e-2 →	2	7	If Z flag is 1, then the program continues. If not, then the displacement e-2 is added to the PC. This is a conditional JUMP.
116.	JP (HL)	PC ⟵ HL	—	—	—	—	—	—	11 101 001	1	4	Contents of register pair HL are loaded into the PC. This is an unconditional JUMP.
117.	JP (IX)	PC ⟵ IX	—	—	—	—	—	—	11 011 101 11 101 001	2	8	Contents of index register IX are loaded into the PC. This is an unconditional JUMP.
118.	JP (IY)	PC ⟵ IY	—	—	—	—	—	—	11 111 101 11 101 001	2	8	Contents of index register IY are loaded into the PC. This is an unconditional JUMP.
119.	DJNZ e	B ⟵ B − 1 If B = 0, then continue. If B ≠ 0, then PC ⟵ PC + e	—	—	—	—	—	—	00 010 000 ← e-2 →	2 3	8 13	Register B is first decremented. If B is now 0, then the program continues. If B is not 0, then displacement e-2 is added to the PC. This is a conditional JUMP.

Table 3-6 CALL AND RETURN INSTRUCTIONS GROUP

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76	543	210	Cycles/States M	T	Operation Description
120.	CALL nn	$(SP - 1) \leftarrow PC_H$ $(SP - 2) \leftarrow PC_L$ $PC \longleftarrow nn$	—	—	—	—	—	—	11 ← n → ← n →	001	101	5	17	Contents of PC are pushed into the stack, first high and then the low byte. Operands nn are then loaded into the PC. This is an unconditional BRANCH.
121.	CALL cc, nn	If cc true, then $(SP - 1) \leftarrow PC_H$ $(SP - 2) \leftarrow PC_L$ If cc is false, continues program	—	—	—	—	—	—	11 ← n → ← n →	cc	100	3 3	10 17	If cc condition is true, contents of PC are pushed into stack and operands nn are loaded into the stack. If cc is false, PC is incremented and the program continues. This is a conditional BRANCH.
122.	RET	$PC_L \longleftarrow (SP)$ $PC_H \leftarrow (SP + 1)$	—	—	—	—	—	—	11	001	001	3	10	Control is returned to the main program upon completion of a subroutine. Contents of (SP) are loaded into low byte of PC. SP is then incremented and contents of (SP + 1) are loaded into high byte of PC. The SP is again incremented.
123.	RET cc	If cc is false, continue program. If cc is true, $PC_L \longleftarrow (SP)$ $PC_H \leftarrow (SP + 1)$	—	—	—	—	—	—	11	cc	000	1 3	5 11	If cc condition is false, subroutine continues. If cc is true, contents of the stack are popped into the PC and main program is resumed. This is a conditional RETURN. The SP is again incremented.

Table 3-6 (*continued*)

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
124.	RETI	Return from interrupt	—	—	—	—	—	—	11 101 101 01 001 101	4	14	Instruction used at the end of a maskable interrupt service subroutine. Contents of stack are popped into PC (same as RET instruction). CPU signals the interrupting I/O device that the service subroutine is completed.
125.	RETN	Return from nonmaskable interrupt	—	—	—	—	—	—	11 101 101 01 000 101	4	14	Instruction used at the end of a nonmaskable interrupt subroutine. Contents of stack are popped into PC (same as RET instruction). State of IFF2 is loaded into IFF1 so that the maskable interrupts are fully enabled after this instruction.
126.	RST p	$(SP - 1) \leftarrow PC_H$ $(SP - 2) \leftarrow PC_L$ $PC_H \longleftarrow O$ $PC_L \longleftarrow p$	—	—	—	—	—	—	11 t 111	3	11	Contents of PC are pushed into the stack. All 0s are loaded into the high byte of PC and address byte p (from Table 3-5) is loaded into the low byte of PC. Now PC addresses page 0 of memory. This restart instruction allows control of program to be given to one of eight subroutine starting addresses as shown in Table 3-5.

3-7 REVIEW QUESTIONS

3-1 Refer to Fig. 3-2 and indicate true or false.
- **a.** _____ This is a left logical shift.
- **b.** _____ This is a left rotate with carry.
- **c.** _____ This is a right rotate with carry.
- **d.** _____ The MSB is shifted into the carry F/F.

3-2 Pick out the statements that correctly apply to Fig. 3-5.
- **a.** The MSB is shifted into the CY E/F.
- **b.** Right-shift arithmetic is performed.
- **c.** The MSB is replicated.
- **d.** After every shift, the bit in the CY F/F is lost.

3-3 Indicate true or false for Fig. 3-8.
- **a.** ____ The high-order nibble of the accumulator is unaltered.
- **b.** ____ The low-order nibble of the accumulator is loaded into the low-order nibble of the HL pair.
- **c.** ____ The low-order nibble of the accumulator is loaded into the high-order nibble of the accumulator.
- **d.** ____ If this instruction is repeated three times, the contents of both the accumulator and the HL pair remain unchanged.

3-4 Refer to Fig. 3-3 and fill in the blanks.
- **a.** The _____ of the register is loaded into the _____ position (MSB; LSB; CY)
- **b.** The _____ of the register is loaded into the CY F/F. (MSB; LSB)

3-5 Figure 3-6 represents a _____ operation. (SLA m; SRA m; SRL m)

3-6 Pick out the correct mnemonic for the instruction where the contents of the accumulator are rotated right 1 bit and bit 0 is transferred into the CY/F/F.
- **a.** RRCA
- **b.** RLCA
- **c.** RRA
- **d.** RLA

3-7 Contents of the memory location specified by the sum of the index register IX and the displacement d are rotated left 1 bit and bit 7 is transferred into the CY F/F.
- **a.** The mnemonic for this instruction is _____.
- **b.** This instruction requires _____ bytes.

3-8 Fill in the blanks for the RRD instruction.
- **a.** The _____ nibble of accumulator is loaded into the _____ nibble of memory location specified by HL. (low; high)
- **b.** The high _____ of this memory location is shifted into the _____ nibble of the same location. (low; high; byte; nibble)
- **c.** The _____ nibble of this memory location is shifted into the _____ nibble of the accumulator. (low; high)

3-9 Explain in your own words what the symbolic operation $Z \leftarrow \overline{r_b}$ means.

_____

_____

_____

3-10 Indicate true or false for the symbolic operation

$$Z \leftarrow \overline{(IY + d)_b}$$

- **a.** ____ The complement of register b is transferred into the Z flag.
- **b.** ____ IY + d represents a memory address.
- **c.** ____ The displacement d is provided in the instruction.
- **d.** ____ This is a 2-byte instruction.
- **e.** ____ Bit b of the memory location is inverted and loaded into the Z flag.

3-11 Write down the mnemonic and the symbolic operation for the instruction. Bit b of operand register r is set.
- **a.** The mnemonic is _____
- **b.** The symbolic operation is _____

3-12 What is the symbolic operation for RES b, m ?

_____

3-13 Fill in the blanks for the mnemonic JP cc, nn.
If condition cc is _____, operand nn is loaded into the _____.
If not, PC is _____. (true; false; PC; HL; IX; IY; incremented, decremented)

3-14 The instruction JR NC, e is a _____ JUMP. (conditional; unconditional)

3-15 The instruction JP nn is _____ JUMP. (conditional; unconditional)

3-16 In the instruction JR NC, e if the flag is _____, then the program continues. (1; 0)

3-17 In the instruction JP (IX) what happens to the contents of the index register IX?
- **a.** It is loaded into the low byte of the PC.
- **b.** Nothing.
- **c.** It is added to the displacement d.
- **d.** It is loaded into the PC (high and low byte)
- **e.** It is incremented.
- **f.** It is decremented.

3-18 Indicate true or false for the instruction CALL nn.

a. ___ Contents of PC are pushed into the stack.
b. ___ Stack pointer is decremented twice.
c. ___ Operands nn are loaded into the PC.
d. ___ This is an unconditional BRANCH.

3-19 Indicate true or false for the RET cc instruction.

a. ___ If cc is true, the subroutine continues.
b. ___ If cc is false, contents of stack are popped into the PC.
c. ___ The main program is then terminated.
d. ___ This is a conditional RETURN.

3-20 The instruction RETI is used at the _____ of a _____ interrupt service subroutine. (beginning; end; maskable; nonmaskable)

4

THE Z80 INSTRUCTION SET AND INTERRUPT RESPONSE

CHAPTER OBJECTIVES

The objectives of this chapter are as follows:

1. To present the Input group of instructions.
2. To present the Output group of instructions.
3. To present the various instructions involved in the operation of the Z80 CPU.
4. To present the instructions associated with the maskable interrupt feature.
5. To present the instructions associated with the interrupt modes available to the Z80.
6. To describe the Z80 interrupt system, including the operation and function of the interrupt enable-disable flip-flop.
7. To present and discuss the CPU responses to both the maskable and the nonmaskable interrupts.

TEXTBOOK REFERENCES

For reviewing material in the main hardcover textbook, relevant to the topics covered in this chapter, the following chapters and/or sections are suggested:

1.	For Input/Output instructions	Sec. 4-2.6
2.	For HALT instruction	Sec. 4-2.3.1
3.	For complement and set instructions	Sec. 4-2.5
4.	For DAA instruction	Sec. 2-4.5 and 2-4.6
5.	For interrupt system operation	Chapter 9

4-1 INTRODUCTION

This chapter is a continuation of Chapters 2 and 3 and it completes the Z80 instruction set. In this chapter we will cover the following groups of instructions:

1. The Input/Output group.
2. The basic CPU instructions group.

In this chapter we also give a fairly extensive discussion of the interrupt processes as used in the Z80 system.

4-2 THE I/O INSTRUCTION GROUP

See Tables 4-1 and 4-2.

Table 4-1 THE INPUT INSTRUCTIONS GROUP

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76	543	210	Cycles/States M	T	Operation Description
127.	IN A, (n)	A ⟵ (n)	—	—	—	—	—	—	11	011	011	3	11	Contents of an addressed I/O device are loaded into register A of the CPU. Following takes place: 1. Operand n → (A0–A7) lines. (*n* is the address of one of the 256 possible I/O ports.) 2. Also, the contents of the accumulator appear on the upper half of the address bus ACC ⟶ (A8–A15) 3. As a result, 1 byte from the I/O is placed on the data bus I/O → (D0–D7) → ACC
									←	n	→			
128.	IN r, (C)	r ⟵ (C)	—	*	P	*	0	*	11	101	101	3	12	Contents of an addressed I/O device are loaded into register r of the CPU. Following takes place: 1. Contents of register C ⟶ (A0–A7) lines. 2. Contents of register B ⟶ (A8–A15) lines. 3. I/O → (D0–D7) → r (*Note:* If *r* = 110, only the flag bits are affected.)
									01	r	000			
129.	INI	(HL) ⟵ (C) B ⟵ B − 1 HL ⟵ HL + 1 *Note:* Memory addresses are incremented.	?	* @	?	?	1	?	11	101	101	4	16	Instruction is used in DMA operations for transferring one word from an I/O device to the memory. Following takes place: 1. Contents of register C ⟶ (A0–A7) lines. 2. Register B is used as a byte counter. B ⟶ (A8–A15). 3. I/O → (D0–D7) ⟶ CPU 4. HL ⟶ (A0–A15) lines. 5. Word input into the CPU from the I/O device is loaded into the memory location. CPU ⟶ (HL) 6. Finally, B is decremented and HL is incremented.
									10	100	010			

Table 4-1 (*continued*)

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
130.	INIR	(HL) ⟵ (C) B ⟵ B − 1 HL ⟵ HL + 1 Repeat until B = 0 *Note:* Memory addresses are incremented.	?	*	?	?	1	?	11 101 101 10 110 010 If B ≠ 0 --- If B = 0 ---	 5 4	 21 16	Used for block transfer in DMA operations. Instruction 129 is repeated until byte counter is 0. 1. C ⟶ (A0–A7) lines. 2. B ⟶ (A8–A15) lines. 3. I/O → (D0–D7) ⟶ CPU. 4. HL ⟶ (A0–A15) lines. 5. CPU ⟶ HL. 6. HL is incremented and B is decremented. 7. If B = 0, instruction is terminated. 8. If B ≠ 0, PC − 2 ⟶ PC and instruction is repeated.
131.	IND	(HL) ⟵ (C) B ⟵ B − 1 HL ⟵ HL − 1 *Note:* Memory addresses are decremented.	?	* @	?	?	1	?	11 101 101 10 101 010	4	16	Used for single-word transfer in DMA operations with decrementing memory addresses. Transfer is from I/O device to memory. 1. C ⟶ (A0–A7) lines. 2. Register B is byte counter, B ⟶ (A8–A15) lines. 3. I/O → (D0–D7) → CPU. 4. HL ⟶ (A0–A15) lines. 5. CPU → (HL). 6. B and HL are decremented.
132.	INDR	(HL) ⟵ (C) B ⟵ B − 1 HL ⟵ HL − 1 Repeat til B = 0.	?	*	?	?	1	?	11 101 101 10 111 010 If B ≠ 0 --- If B = 0 ---	 5 4	 21 16	Used for block transfer in DMA operations. Instruction 131 is repeated until byte counter is 0. 1. C ⟶ (A0–A7) lines. 2. Register B is byte counter, B ⟶ (A8–A15) lines. 3. I/O → (D0–D7) CPU. 4. HL ⟶ (A0–A15) lines. 5. CPU ⟶ (HL) 6. B and HL are decremented. 7. If B = 0, instruction is terminated. 8. If B ≠ 0, PC − 2 ⟶ PC and instruction is repeated.

Table 4-2 THE OUTPUT INSTRUCTIONS GROUP

No.	*Mnemonics*	*Symbolic Operations*	*Status Flags* C	Z	P/V	S	N	H	*OP Code* 76 543 210	*Cycles/ States* M	T	*Operation Description*
133.	OUT (n), A	(n) ⟵A	—	—	—	—	—	—	11 010 011 n	3	11	Contents of register A in the CPU are loaded into addressed I/O device. 1. Operand n → (A0–A7) lines. (n is the address of one of the 256 possible I/O ports.) 2. Also, the contents of the accumulator appear on upper half of the address bus. ACC ⟶ (A8–A15) lines. 3. As a result, 1 byte from the accumulator is placed on the data bus and transferred to I/O. ACC → (D0–D7) → I/O
134.	OUT (C), r	(C) ⟵ r	—	—	—	—	—	—	11 101 101 01 r 001	3	12	Contents of register r in the CPU are loaded into the addressed I/O device. 1. C ⟶ (A0–A7) lines. 2. B ⟶ (A8–A15) lines. 3. r → (D0–D7) → I/O

Table 4-2 (*continued*)

No.	*Mnemonics*	*Symbolic Operations*	*Status Flags* C	Z	P/V	S	N	H	*OP Code* 76 543 210	*Cycles/ States* M	T	*Operation Description*
135.	OUTI	(C) ⟵ (HL) B ⟵ B − 1 HL ⟵ HL + 1 *Note:* Memory addresses are incremented.	?	* @	?	?	1	?	11 101 101 10 100 011	4	16	Used in DMA operations for transferring one word of data from the memory to the I/O device 1. HL ⟶ (A0–A7) lines. 2. HL selects the byte in the addressed memory location. 3. Memory byte ⟶ CPU (temporarily stored) 4. B − 1 ⟶ B 5. (C) ⟶ (A0–A7) lines. 6. (B − 1) → (A8–A15) lines. 7. CPU (output byte) ⟶ (D0–D7) ⟶ I/O device. 8. HL + 1 ⟶ HL
136.	OTIR	(C) ⟵ (HL) B ⟵ B − 1 HL ⟵ HL + 1 Repeat until B = 0 *Note:* Memory addresses are incremented.	?	1	?	?	1	?	11 101 101 If B ≠ 0 --- If B = 0 ---	 5 4	 21 16	Used for block transfer in DMA operations. Instruction 135 is repeated until byte counter is 0. 1. HL ⟶ (A0–A7) lines. 2. HL selects the byte in the addressed memory location. 3. Memory byte ⟶ CPU. (temporarily stored) 4. B − 1 ⟶ B 5. (C) ⟶ (A0–A7) lines. 6. (B − 1) → (A8–A15) lines. 7. CPU (output byte) ⟶ (D0–D7) ⟶ I/O device. 8. HL + 1 ⟶ HL 9. If B ≠ 0, PC − 2 → PC and instruction is repeated 10. If B = 0, instruction terminated.

Table 4-2 (*continued*)

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
137.	OUTD	(C) ← (HL) B ← B − 1 HL ← HL − 1 *Note:* Memory addresses are decremented.	?	* @	?	?	1	?	11 101 101 10 101 011	4	16	Used in DMA operations for transferring one word of data from memory to I/O device. 1. HL ⟶ (A0–A15) lines. 2. HL selects the byte in the addressed memory location. 3. Memory byte ⟶ CPU. (temporarily stored) 4. B − 1 ⟶ B 5. (C) → (A0–A7) lines. 6. (B − 1) → (A8–A15) lines. 7. CPU (output byte) ⟶ (D0–D7) ⟶ I/O device. 8. HL − 1 ⟶ HL.
138.	OTDR	(C) ← (HL) B ← B − 1 HL ← HL − 1 Repeat until B = 0.	?	*	?	?	1	?	11 101 101 10 111 011			Used for block transfers in DMA operations. Instruction 137 is repeated until byte counter is 0. 1. HL ⟶ (A0–A15) lines. 2. HL selects the byte in the addressed memory location. 3. Memory byte ⟶ CPU. (temporarily stored) 4. B − 1 ⟶ B 5. (C) ⟶ (A0–A7) lines. 6. (B − 1) → (A8–A15) lines. 7. CPU (output byte) ⟶ (D0–D7) ⟶ I/O device. 8. HL − 1 ⟶ HL 9. If B ≠ 0, PC − 2 ⟶ PC and instruction is repeated. 10. If B = 0, instruction is terminated.

4-3 THE BASIC CPU INSTRUCTION GROUP

Note: Table 4-3 completes the Z-80 instruction set with instruction IM 2, which brings it to a total of 150 instructions. In Section 2-1 it was mentioned that the Z80 has a total of 157 instructions. Students may wonder what happened to the remaining eight instructions.

This discrepancy is readily explained by looking at the RST p instruction (instruction 126 in Table 3-6). Notice that the OP code of this instruction contains the 3-bit t code which defines the starting addresses for special subroutines. (See Table 3-5.) Since eight such starting addresses are possible, instruction 126 in Table 3-6 is the general format for eight such instructions, which brings the total available instructions in the Z80 set to 157.

4-4 THE INTERRUPT SYSTEM

4-4.1 The Objectives

1. Interrupts are initiated by external I/O peripherals or other devices to demand servicing of the particular service subroutine.
2. The CPU current instruction is completed and the CPU operation is suspended in an orderly fashion.
3. Service subroutines normally involve transfer of data between peripherals and CPU or status and control information between the two.
4. On completion of the service subroutine, CPU resumes execution of the main program.

4-4.2 The Nonmaskable Interrupt (NMI)

1. This type of interrupt is designed into the hardware of the CPU logic and is not under control of the programmer.
2. The programmer cannot selectively disable or enable this interrupt.
3. The NMI is accepted, and serviced, by the CPU whenever it arrives. It has the highest priority and can override all other operations.
4. The NMI is reserved for very critical functions, such as an impending power failure, whose demand for service cannot be delayed.

4-4.3 The Maskable Interrupt (INT)

1. The INT can be selectively deactivated by the programmer.
2. When the INT is deactivated, any incoming interrupts will be ignored by the CPU.
3. This feature is generally used by the programmer when certain critical instructions in the program are being executed. Usually such instructions involve critical timing restraints which cannot be interrupted.
4. When execution of these critical instructions is completed, the INT feature is reactivated.

4-4.4 The INT Enable-Disable Flip-Flop

1. The Z80 CPU has a flip-flop, IFF1, which is used to control the acceptance or rejection of the maskable interrupt (INT).
2. The DI instruction (see Table 4-3) resets IFF1 to the O side, and this forces the CPU to ignore all incoming INT requests.
3. The EI instruction (see Table 4-3) sets IFF1 to the 1 side. Now the CPU will recognize and honor INT requests from the I/O devices.
4. Notice one peculiarity of the EI instruction. When it is being executed, if any INT comes along, it will only be accepted after completion of the instruction following the EI instruction. This one instruction delay is particularly useful if the instruction after the EI instruction is a return from subroutine (RET).
5. In the Z80 CPU there are really two enable/disable flip-flops. The second one is labeled IFF2. Both these flip-flops work in unison for the maskable interrupt situations.
6. The function of IFF2 is to save the status of IFF1 when a nonmaskable interrupt (NMI) happens. Here is how it works:
   a. When an NMI is recognized, IFF1 is immediately reset.
   b. This locks out any other INT from coming in the CPU while the NMI is being serviced.
   c. During the NMI servicing, IFF2 is not disturbed and retains the prior status of IFF1 (either a 0 or a 1).
7. The last instruction in the NMI service subroutine is a return from nonmaskable interrupt (RETN) instruction (see Table 3-6).
8. The RETN instruction copies the status of IFF2 into IFF1, thereby returning IFF1 to its pre-NMI status automatically.

4-4.5 CPU Response to Nonmaskable Interrupts (NMI)

1. The CPU accepts the NMI at all times.
2. The CPU ignores the next instruction it has fetched

Table 4-3 BASIC CPU INSTRUCTIONS GROUP

No.	Mnemonics	Symbolic Operations	Status Flags C	Z	P/V	S	N	H	OP Code 76 543 210	Cycles/States M	T	Operation Description
139.	DAA	ACC contents are converted into packed BCD for BCD add and subtract.	*	*	P	*	—	*	00 100 111	1	4	Conditionally adjusts ACC for BCD addition and subtraction. Additions involve ADD, ADC, and INC instructions. Subtractions involve SUB, SBC, DEC, and NEG instructions.
140.	CPL	$A \longleftarrow \overline{A}$	—	—	—	—	1	1	00 101 111	1	4	Contents of the accumulator are complemented.
141.	NEG	$A \longleftarrow 0 - A$	*	*	V	*	1	1	11 101 101 01 000 100	2	8	Accumulator contents are negated by subtracting them from 0.
142.	CCF	$CY \longleftarrow \overline{CY}$	*	—	—	—	0	?	00 111 111	1	4	The CY F/F in register F is complemented.
143.	SCF	$CY \longleftarrow 1$	1	—	—	—	0	0	00 110 111	1	4	The CY F/F in register F is set to 1.
144.	NOP	No operation	—	—	—	—	—	—	00 000 000	1	4	No operation is performed by CPU during this machine cycle, but dynamic memories are refreshed.
145.	HALT	CPU IS HALTED	—	—	—	—	—	—	01 110 110	1	4	CPU operation is suspended until an interrupt or reset comes along. CPU executes NOP instructions, which keeps on refreshing dynamic memory chips.

Table 4-3 (*continued*)

No.	*Mnemonics*	*Symbolic Operations*	*Status Flags*						*OP Code*			*Cycles/ States*		*Operation Description*
			C	*Z*	*P/V*	*S*	*N*	*H*	*76*	*543*	*210*	*M*	*T*	
146.	DI	IFF ⟵ 0	—	—	—	—	—	—	11	110	011	1	4	Maskable interrupt feature is disabled by resetting IFF1 and IFF2 flip-flops.
147.	EI	IFF ⟵ 1	—	—	—	—	—	—	11	111	011	1	4	Maskable interrupt feature is enabled by setting IFF1 and IFF2 flip-flops to the 1 side.
148.	IM 0	Set interrupt Mode 0	—	—	—	—	—	—	11 01	101 000	101 110	2	8	Mode 0 interrupt is activated. The interrupting device sends the instruction to be executed to the CPU via the data lines (D0 D7), 1 byte at a time.
149.	IM 1	Set interrupt Mode 1	—	—	—	—	—	—	11 01	101 010	101 110	2	8	Mode 1 interrupt is activated. CPU responds to interrupt by executing a restart to location 0038H in memory.
150.	IM 2	Set interrupt Mode 2	—	—	—	—	—	—	11 01	101 011	101 110	2	8	Mode 2 interrupt is activated. Interrupting I/O suplies 8-bit vector for low byte of indirect address pointer. Register I supplies the high-order byte of the pointer. This mode provides an indirect CALL to any memory location for the starting address of the service subroutine.

and instead goes to memory location 0066H and performs an operation similar to a restart operation.

3. Notice that the memory location 0066H is not included in the eight software-controlled restart addresses located in Page 0 of the memory. (See Table 3-5.)

4-4.6 CPU Response to Maskable Interrupts (INT)

The Z80 has three different operational modes for responding to INT requests. The modes can be changed by means of program instructions. (See Table 4-3.) Brief descriptions of each of these modes follow:

4-4.6.1 Mode 0

1. In this mode the interrupting device sends an 8-bit instruction to the CPU on the data bus lines (D0–D7).
2. The CPU does not execute the next instruction fetched from the memory but instead executes the instruction sent on the data bus.
3. This instruction could be a 3-byte call (i.e., a branch) instruction to some other location in the memory.
4. In executing this instruction, the CPU automatically inserts two wait states in the special M cycle. This will be explained later in Sec. 5-6 with Fig. 5-9. This allows sufficient time for the daisy chain to respond and the interrupting device to identify itself.

4-4.6.2 Mode 1

1. In this mode the CPU responds to an INT by automatically branching to memory location 0038H, which is the starting address of the subroutine.
2. This response is identical to the NMI response, described in Sec. 4-4.5, except that the subroutine starting address is 0038H instead of 0066H.
3. One difference is that, unlike the NMI response, the CPU automatically inserts two wait states, for reasons that were previously discussed. See and compare the two timing charts in Fig. 5-9 and Fig. 5-10.

4-4.6.3 Mode 2

1. This is the most powerful and versatile interrupt response. Its primary objective is to give the user the ability to branch to any location within the memory that can be addressed essentially with a 15-bit address for the starting address of the subroutine.
2. An indirect addressing mode is used, and the address is formed as explained below.
3. The programmer constructs a 16-bit address table and prestores it in the memory. The 2-byte addresses are the starting addresses of the various service subroutines.
4. The interrupt vector register I is previously stored with a desired byte, which forms the high-order byte of the concatenated 16-bit address.
5. The low order of the concatenated 16-bit address is supplied by the interrupting device.

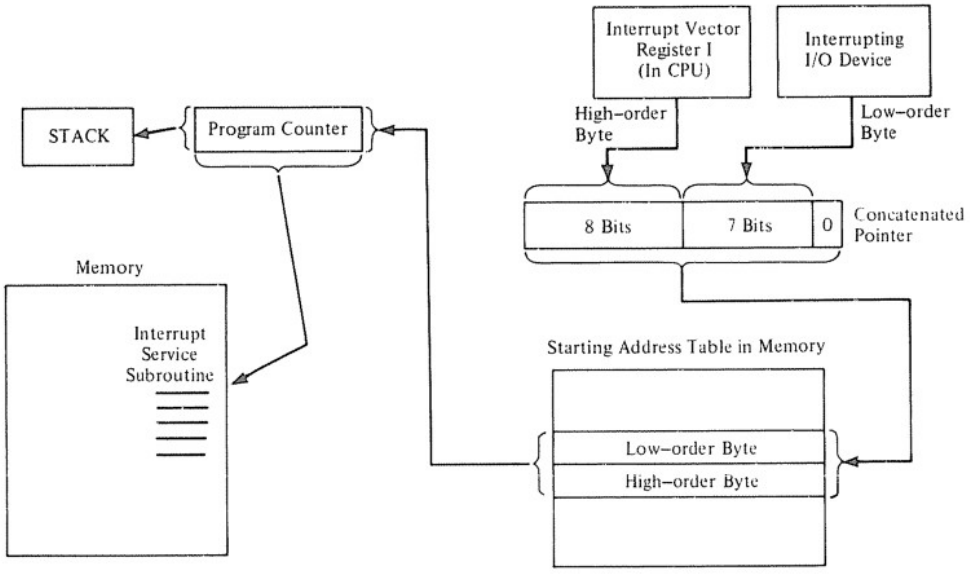


Figure 4–1 Graphical Representation of the Mode 2 Interrupt Response

6. Since the concatenated address is a 2-byte address, this fact is indicated by bit 0 of the low-order byte, which is always a 0.
7. The 16-bit address thus formed is used as a pointer that points to the previously stored table of starting addresses. This table contains the true or effective starting addresses of the desired service subroutines.
8. This is a branch operation which requires 19 clock periods as follows:

   7—Fetch low-order byte from interrupting device.
   6—Push PC contents into the stack.
   6—Fetch the effective starting address from the table.

9. Figure 4-1 is a graphical representation of Mode 2 interrupt operation.

4-5 REVIEW QUESTIONS

4-1 _____ In the Z80 system how many I/O ports is it possible to address? (8; 16; 32; 64; 256; 512; 1024)

4-2 Refer to Table 4-1 and indicate true or false for the instruction

IN A, (*n*)

- **a.** ____ Operand *n* is the address of one of the ports
- **b.** ____ The CPU places *n* on the D0–D7 lines.
- **c.** ____ The accumulator places its contents on the lower half of the address bus.
- **d.** ____ One byte from the I/O is loaded into the accumulator via the data bus.

4-3 The following statements apply to the IN r, (C) instruction. Fill in the blanks.

- **a.** The contents of register _____ are put on the A8–A15 lines. (A; B; C; D; E; H; L)
- **b.** The contents of register _____ are put on the A0–A7 lines. (A; B; C; D; E; H; L)
- **c.** The I/O device loads its 8-bit word into the _____. (accumulator; B register; C register)

4-4 In the INI instructions _____ is used as a byte counter. (register A; register B; register C; r register)

4-5 In the INI instruction _____ is decremented and register pair _____ is incremented. (A; B; C; D; I; BC; DE; HL)

4-6 Indicate true or false for the INIR instruction.

- **a.** ____ The instruction is repeated until the contents of BC are 0.
- **b.** ____ The contents of the byte counter are put on bus lines A8–A15.
- **c.** ____ The contents of the HL pair are put on bus lines A0–A7.
- **d.** ____ If B $\neq$ 0, the instruction is terminated.
- **e.** ____ If B $\neq$ 0, the contents of PC is decremented once.

4-7 In the IND instruction, the contents of both register B and the register pair HL are _____. (incremented; decremented)

4-8 Indicate true or false for the INDR instruction.

- **a.** ____ Register B is the byte counter.
- **b.** ____ B and HL are both incremented.
- **c.** ____ If B = 0, the instruction is terminated.
- **d.** ____ Instruction is repeated until B = 0.

4-9 The instruction OUT (C), r outputs the contents of register _____ to the I/O on the _____ lines. (m; n; r; B; C; A0–A7; A8–A15; D0–D7)

4-10 The following statements relate to the OUTI instruction. Indicate true or false.

- **a.** ____ HL selects a byte in the addressed memory location.
- **b.** ____ The memory byte is temporarily stored in the CPU.
- **c.** ____ Register B is incremented.
- **d.** ____ Contents of register B are put on the A8–A15 lines.
- **e.** ____ Contents of register C are put on the A0–A7 lines.

4-11 Fill in the blanks for the OUTIR instruction.

- **a.** If _____, the instruction is terminated. (B $\neq$ 0; B = 0)
- **b.** If _____, the instruction is repeated until _____. (B = 0; B $\neq$ 0; PC = 0; PC $\neq$ 0).
- **c.** The program counter is _____ twice. (incremented; decremented)

4-12 In the OUTD instruction, the word to be output to the I/O device is originally located in the _____. (accumulator; B register; memory; HL register)

4-13 The DAA instruction applies to the _____ arithmetic operations. (BCD; hexadecimal; binary)

4-14 Which of the following mnemonics is used to complement the contents of the accumulator?

a. CPL
b. CCF
c. SCF
d. DI
e. None of the above

4-15 The SCF instruction _____ the carry F/F. (sets; resets; complements)

4-16 Indicate true or false for the HALT instruction.
a. ___ Dynamic memory chips are refreshed.
b. ___ NOP instructions are executed.
c. ___ HALT state continues until an interrupt comes in.

4-17 The maskable interrupt feature is disabled by the _____ instruction, which _____ the IFF1 and the IFF2 flip-flops. (EI; DI; resets; sets)

4-18 Identify the interrupt mode in which the interrupting I/O device sends the instruction to be executed to the CPU.
a. Mode 0
b. Mode 1
c. Mode 2

4-19 Identify the interrupt mode which provides an indirect branch to a memory location for the starting address of the service subroutine.
a. Mode 0
b. Mode 1
c. Mode 2

4-20 Which type of interrupt mode can be selectively deactivated?
a. NMI
b. INT

5

THE Z80 CPU TIMING AND CONTROL

CHAPTER OBJECTIVES

The objectives of this chapter are as follows:

1. To explain the fundamental Z80 instruction cycle and the concept of M cycles.
2. To present and analyze the OP code fetch cycle with and without WAIT states.
3. To present and discuss the memory read and write cycles with and without WAIT states.
4. To present and examine the I/O read and write cycles with and without WAIT states.
5. To examine bus request and acknowledge cycles and their respective timings.
6. To present and discuss the interrupt request and acknowledge cycles for both the maskable and the nonmaskable interrupts.
7. To present the timing involved in an exit from HALT cycle.

TEXTBOOK REFERENCES

For reviewing material in the main textbook, relevant to the topics covered in this chapter, the following chapters and/or sections are suggested:

1. For general discussion on machine cycles	Secs. 3-2.1 and 3-2.2
2. For fixed instruction cycle	Sec. 3-2.3.1
3. For variable instruction cycle	Sec. 3-2.3.2
4. For general review of interrupt I/O	Chapter 9
5. For maskable interrupts	Sec. 9-3

5-1 THE FUNDAMENTAL INSTRUCTION CYCLE

As previously explained in Sec. 1-3 (paragraphs 3, 4, and 5) and Table 1-5 in Chapter 1, the Z80 uses a single-phase clocking system for controlling and synchronizing all its operations. The Z80 instructions execute the following five basic operations:

1. Memory read.
2. Memory write.
3. External device (I/O) read.
4. External devices (I/O) write.
5. Interrupt acknowledge.

In the Z80 CPU the clock period is defined as the time interval between the leading edge of one positive-going edge of the clock pulse and the leading edge of the next positive-going edge of the clock pulse. It is referred to as the T cycle, as shown in Fig. 5-1. Each of the previously mentioned five basic operations normally takes from three to six clock periods to complete. However, the Z80 CPU is designed to be flexible in its operations, so that the time required to perform these operations could be lengthened beyond the six clock periods in order to synchronize the CPU with the slower peripherals. As shown in Fig. 5-1, the instruction cycle is broken down into subcycles,

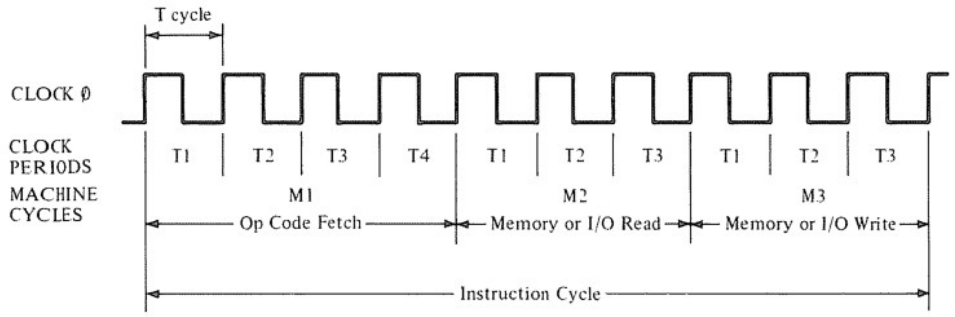


Figure 5-1 Z80 CPU Basic Instruction Cycle Timing

referred to as the M cycles. Each M cycle consists of several different clock periods, or T cycles. Thus, each instruction cycle will consist of a series of M cycles and each M cycle will consist of a different number of T cycles. Figure 5-1 is merely an example of a typical instruction. The M1 subcycle fetches the OP code of the next instruction to be executed. Normally, M1 consists of four to six T cycles. However, M1 could be lengthened by a WAIT signal from an external device (see Table 1-3). M2 and M3 (as shown in Fig. 5-1) are machine cycles that merely transfer data between the CPU and the memory or I/O devices. M2 and M3 are normally three to five T cycles long, but they could also be lengthened by the WAIT signals to accommodate slower external devices.

Figure 5-1 only shows a fundamental timing diagram. The specific timing for each operation is of course different. In this chapter we will describe and discuss the timing charts for the following seven specific operations. The diagrams are intended to show operations with and without the WAIT states. It is recommended that students refer to Tables 1-1 through 1-4, where the functions of the various CPU signals are described. The seven specific operations are:

1. The OP code fetch cycle (M1).
2. Memory read and write cycles.
3. I/O read and write cycles.
4. Bus request/acknowledge cycles.
5. Maskable interrupt request/acknowledge cycles.
6. Nonmaskable interrupt request/acknowledge cycles.
7. Exit from a HALT instruction cycle.

5-2 THE OP CODE FETCH CYCLE

5-2.1 Without Wait States

1. Figure 5-2 shows the timing of the various CPU signals involved in fetching the OP code of the instruction. Notice that in Fig. 5-1 the clock pulses are shown as perfect square waves for simplicity. Of course the leading and trailing edges of each pulse have associated with them corresponding rise and fall times. In the rest of this chapter, the clock pulses are shown with their rise and fall times. Also, all the other signals are referenced to the 50% point of the rise and fall times.
2. Figure 5-2 shows the M1 cycle and the status of each signal during the four clock periods, T1–T4. The first step in fetching the OP code is to transfer the contents of the PC, i.e., the instruction address, to the memory. In Fig. 5-2 this is shown. The address bus A0–A15 is activated during T1 and T2 clock periods. Note that the two waveforms as shown indicate that some of the address lines will be high and others low, depending on the combinations of 1s and 0s in the address.
3. During the T3 and T4 clock periods, the lower seven lines of the address bus are used to send the memory refresh address.
4. Since an OP code fetch is a memory access operation, the CPU informs the memory that the bits on the address bus are valid address bits by activating the $\overline{\text{MREQ}}$ signal. Notice that this line goes low during the second half of T1, thus giving the address bus ample time to stabilize during the first half of T1. The $\overline{\text{MREQ}}$ signal stays active for the entire duration of T2.
5. Since the OP code fetch is a memory read operation, the CPU so informs the memory by activating the $\overline{\text{RD}}$ signal, essentially during the same time that $\overline{\text{MREQ}}$ is active.
6. On receiving $\overline{\text{RD}}$, the memory activates its output gates and the OP code byte is placed on the data bus, DB0–DB7. This happens during the second half of T2. The symbol shown in Fig. 5-2 indicates that the various lines could be either high or low.
7. Output signal $\overline{\text{M1}}$ is activated during T1 and T2. It indicates that the current machine cycle involves

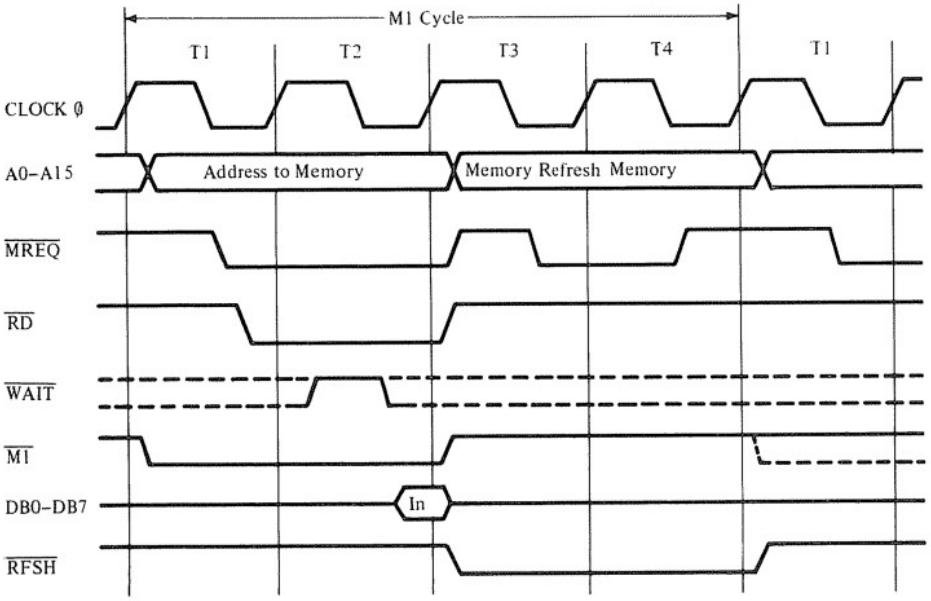


Figure 5-2 CPU Timing for Instruction OP Code Fetch without WAIT

fetching the OP code. Notice that $\overline{M1}$ is shown only once in Fig. 5-2, since it is assumed that the OP code contains only 1 byte. The dotted lines show that $\overline{M1}$ would be generated during T1 of M2 if the OP code involved has 2 bytes.

8. During T3 and T4, the $\overline{RFSH}$ is activated by the CPU, indicating that the refresh address for dynamic memories is put on the lower seven lines of the address bus.
9. Notice that the $\overline{WAIT}$ signal is high during the M1 cycle. The pulse shown in T2 is only intended to indicate the high and low levels of this signal from the memory.
10. Notice that the $\overline{MREQ}$ signal, which is active during the second half of T3 and the first half of T4, is intended to perform a refresh operation for all the elements of dynamic memories. The refresh address is stable on the address bus only when $\overline{MREQ}$ is active. Therefore, the $\overline{RFSH}$ signal by itself does not perform the refresh operation.

5-2.2 With WAIT States

1. The CPU timing with WAIT states is shown in Fig. 5-3. The basic operation of the various signals is identical to that described in Sec. 5-2.1, and so will not be repeated here. There are differences, and these are pointed out.
2. During T2 the CPU samples the signal on the $\overline{WAIT}$ line. If it is active (i.e., if it is low), the CPU goes into a wait state for the next T period of the machine cycle. This is identified as TW1 in Fig. 5-3.
3. As a result of the activation of $\overline{WAIT}$, all the CPU signals are maintained in the positions that they were in at the end of T2.
4. During TW1 the CPU again samples the $\overline{WAIT}$ line. If this signal is active, the CPU goes into another wait cycle, identified as TW2 in Fig. 5-3.
5. Every subsequent wait cycle, TW, the line is checked for its status. When the $\overline{WAIT}$ is sampled and found to be high, clock periods T3 and T4 are resumed and the normal functions are executed.
6. This technique of status checking allows the Z80 CPU to work effectively with memory chips requiring much longer access times.

5-3 MEMORY READ AND WRITE CYCLES

5-3.1 Without WAIT States

1. Because an OP code fetch operation is a memory access operation, the memory read/write cycles are very similar to the previously described fetch operation

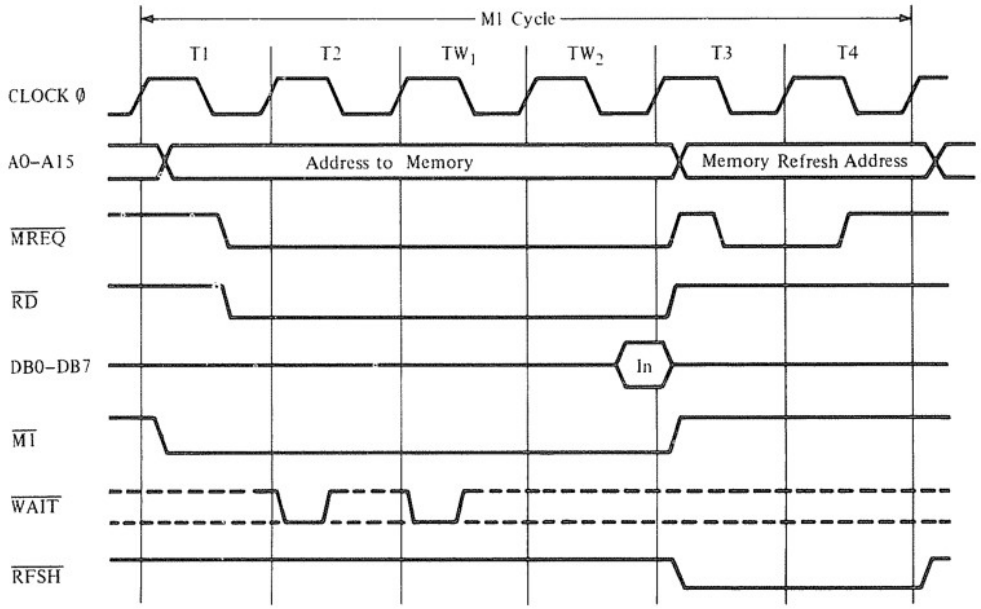


Figure 5-3 CPU Timing for Instruction OP Code Fetch with WAIT

for the OP code (Sec. 5-2) and so will not be repeated here.

2. Both the memory read and write cycles each require three clock periods. They are shown in Fig. 5-4 as two separate M cycles.
3. The functions and operations of both the MREQ and RD signals are identical to those for the op code fetch operation.
4. Of course, the address of the memory location to be accessed is first placed on the address bus A0–A15.
5. During both the read and the write operations, the MREQ line is active during the second half of T1, all of T2, and the first half of T3. This allows the address bus to stabilize. This is especially important during the write cycle. This then allows the MREQ to be directly used as a chip enable signal in dynamic memory chip.
6. During the memory write cycle, note that the WR signal is active during the second half of T2 and the first half of T3. The WR signal is issued by the CPU to inform the memory that the CPU has placed valid data on the data bus and that it should be accepted by the memory.
7. Because of the time span during which $\overline{WR}$ is active (see Fig. 5-4), it is possible to use it as $\overline{R/W}$ signal and so can be used for almost any semiconductor memory chips.
8. Also notice that WR goes high one-half clock period (i.e., during the second half of T3) before the contents of either the address bus or the data bus are changed. This makes the Z80 flexible enough to meet the overlap limitations and makes it suitable for use with many different types of memory chips.

5-3.2 With WAIT States

1. Figure 5-5 shows two WAIT state cycles, TW1 and TW2, introduced by two WAIT pulses during T2 and TW1, respectively. The operation due to these two pulses is identical to that previously described for the op code fetch cycle.
2. In Fig. 5-5 we have shown separate data bus situations for the read and write cycles using the same WAIT signals. Of course it is impossible to have a read and a write operation simultaneously. Figure 5-5 is drawn this way for convenience only.
3. Notice that in the read operation, data are placed on the data bus D0–D7 only during the T3 clock period. For the write operation, the CPU maintains the data bits on the data bus from the second half of T1 and T2, through the wait states, and through T3. This gives the memory ample time to write the data into the accessed locations.

5-4 I/O READ AND WRITE CYCLES

5-4.1 Without WAIT States

1. The I/O read and write operations are similar to the memory read/write operations in most respects, but there are also differences, as explained below.

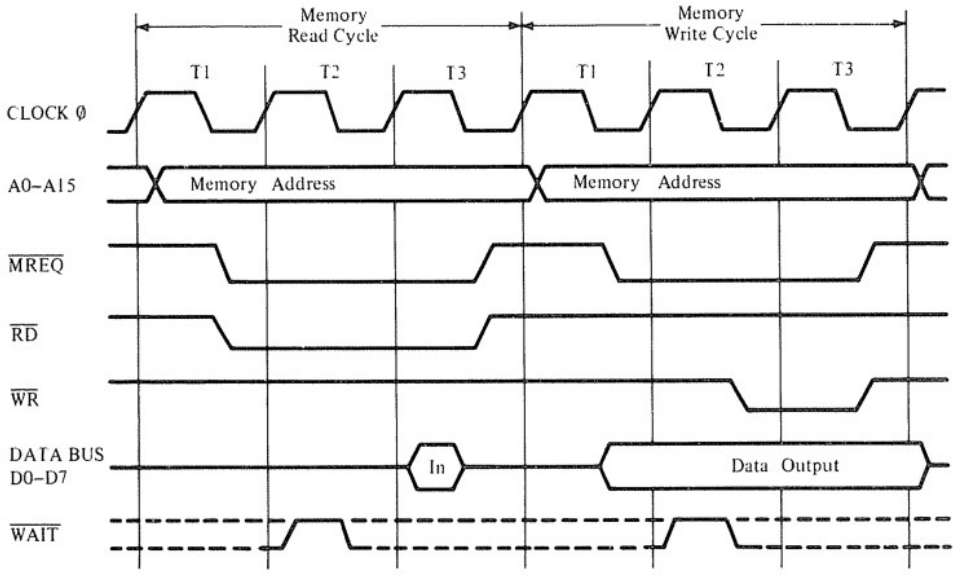


Figure 5-4 Memory Read/Write Cycle Timing without WAIT

2. Figure 5-6 shows the I/O read and write operations cycles. Note that the IORQ is now used instead of the MREQ signal, as in the previous case.
3. The first obvious difference is that Fig. 5-6 includes a wait state identified as the TW0 cycle. This state is intentionally designed and inserted into the system for the following reasons:
   a. The IORQ signal is activated during the T2 cycle. If the CPU were to use the same WAIT line during T2, then there would be very little time available between these two signals being activated.
   b. Consequently, this *very short* period of time would not be sufficient for the I/O port to decode its address and activate the WAIT line if a genuine wait were required.
   c. I/O devices using the MOS technology are inherently slower than the CPU. The short time between these two signals would only aggravate the speed incompatibility problem even more.

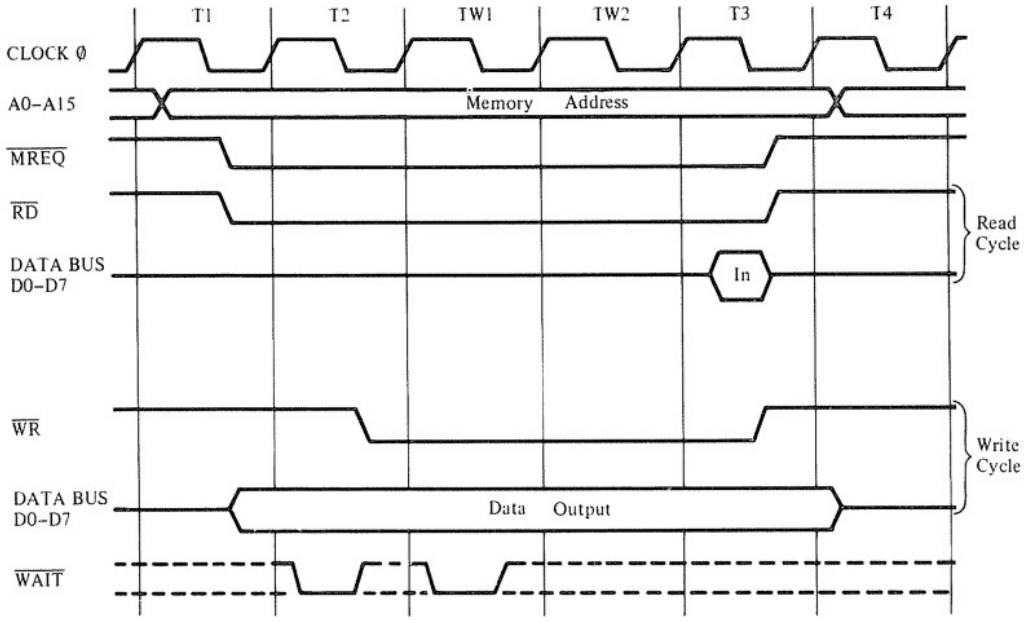


Figure 5-5 Memory Read/Write Cycle Timing with WAIT

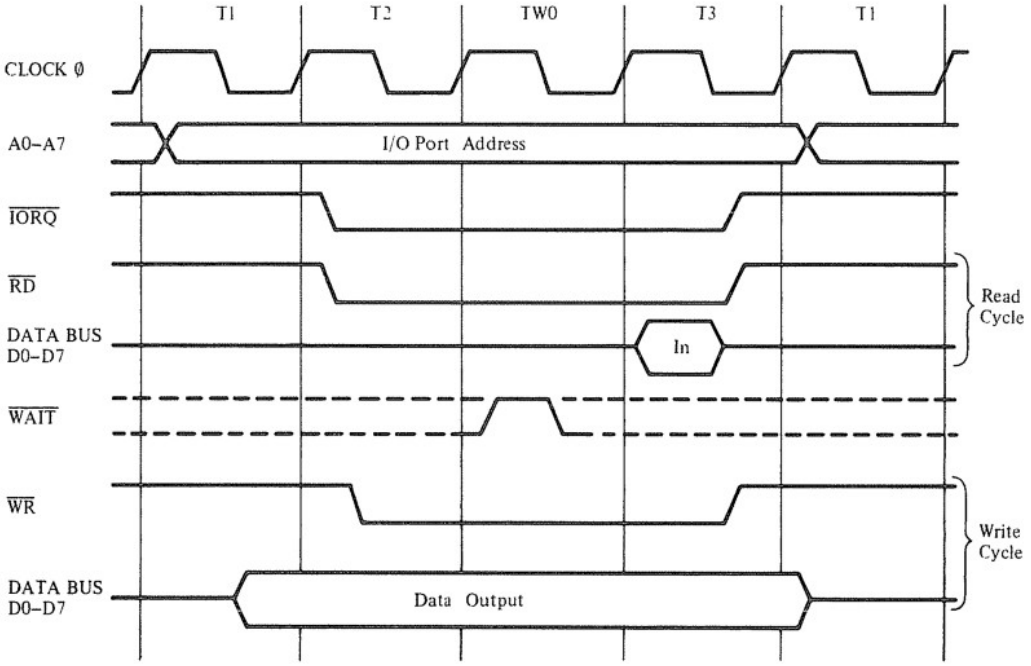


Figure 5-6 I/O Read and Write Cycle Timing without WAIT States

4. During TW0, the CPU samples the $\overline{\text{WAIT}}$ line, as shown in Fig. 5-6.
5. During the read operation, the $\overline{\text{RD}}$ signal is used to gate the contents of the particular port that was addressed onto the data bus lines D0–D7.
6. During the write operation, the $\overline{\text{WR}}$ signal is used as a clock signal to the addressed I/O port. It is used to clock the available bits on the data bus lines to be input into the I/O device.

5-4.2 With WAIT States

1. Figure 5-7 shows the I/O read/write operations with the wait states. All the previous discussions of Sec. 5-3.2 apply here.

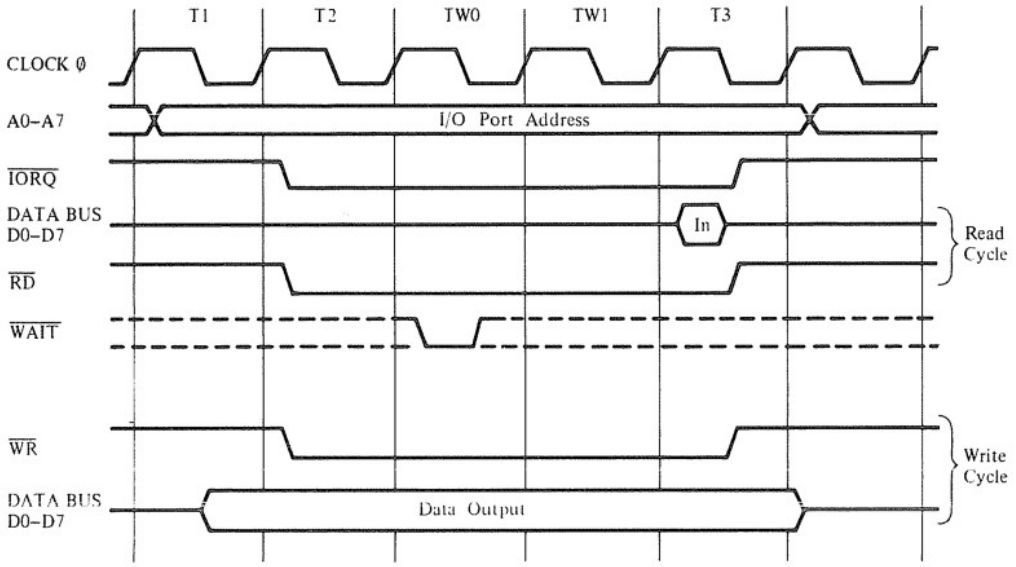


Figure 5-7 I/O Read and Write Cycle Timing with WAIT States

2. Note that in Fig. 5-7, the $\overline{\text{WAIT}}$ signal is introduced as an active signal during TW0. The reason for this is explained in Sec. 5-4.1, paragraphs 3a, b, and c.

Note: During both the memory and the I/O data transfers, the $\overline{\text{WAIT}}$ line must be low during the *trailing edge* of the clock pulse for the CPU to go into the wait state during the next clock period.

5-5 BUS REQUEST/ACKNOWLEDGE CYCLES

1. For certain operations, the external I/O devices need to have control of the three buses from the CPU. These buses are the address bus (A0–A15), the data bus (D0–D7), and the control bus lines. In the Z80, this mode of operation is mainly used in the DMA transfers of blocks of data between the I/O device and the memory on a cycle-stealing basis.
2. The external I/O device initiates the control of the buses by sending a signal on the $\overline{\text{BUSRQ}}$ line. The Z80 CPU is designed so that it samples this line at the leading edge rise time of the clock pulse during the last clock period T of every machine cycle, M.
3. If the $\overline{\text{BUSRQ}}$ is active (i.e., in the low state), then at the leading edge of the next clock pulse, the CPU relinquishes control of these three buses by putting them in the high-impedance states. This is shown in Fig. 5-8. The external device then gets control of these buses and can retain this control for as many clock pulses as necessary.
4. If dynamic memory chips are used in the system and if the DMA transfers involve large blocks of data, requiring very long DMA cycles, then the external DMA controller or chip must perform the memory refresh function, since the CPU does not have any direct control over the memories during DMA cycles.
5. Also note that during the bus request cycles, both the $\overline{\text{INT}}$ (interrupt request signal) and the $\overline{\text{NMI}}$ (nonmaskable interrupt request) are deactivated. Neither of these two signals will be able to interrupt the CPU during bus request cycles. (See Table 1-3.)
6. The I/O device terminates this cycle by deactivating the $\overline{\text{BUSRQ}}$ signal (i.e., returning it to the high state). At the leading edge of the next clock pulse, control of these busses is returned to the CPU.

5-6 MASKABLE INTERRUPT REQUEST/ACKNOWLEDGE CYCLE

1. Figure 5-9 shows the timing involved in the maskable interrupt cycle. A maskable interrupt refers to the situation where the interrupt can be masked or overriden by means of CPU software. The timing chart of Fig. 5-9 assumes that this feature is not activated under program control and so the CPU will honor this incoming interrupt.
2. At the leading edge of the last clock pulse of all instructions, the CPU samples the $\overline{\text{INT}}$ line. If it is low, then the CPU goes into a special M1 cycle, starting with the leading edge of the next clock pulse.

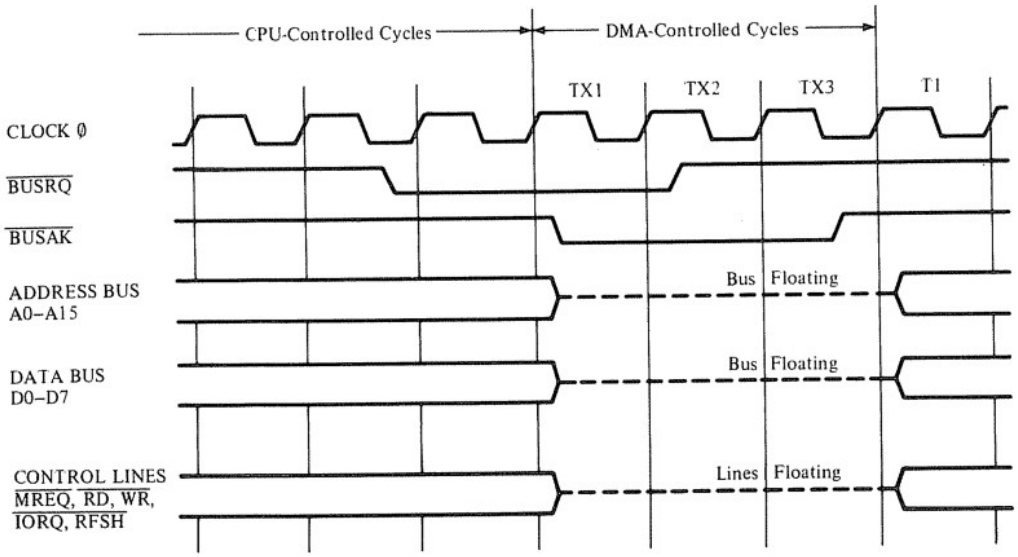


Figure 5-8 Bus Request/Acknowledge Cycle

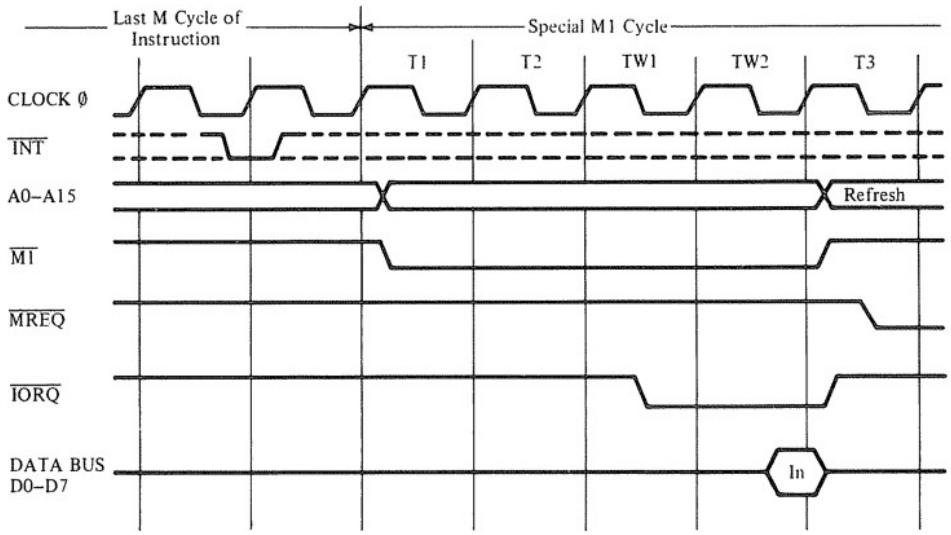


Figure 5-9 Maskable Interrupt Request/Acknowledge Cycle

3. As shown in Fig. 5-9, the M1 cycle consists of five T states with two special wait states automatically inserted between T2 and T3. As mentioned in Sec. 1-3, a daisy chain priority interrupt scheme is available in the Z80 CPU. Since a daisy chain involves rippling a signal through the logic hardware associated with each I/O device, to identify the interrupting I/O, this signal requires time. This additional time is provided by TW1 and TW2. This allows ample time for the interrupting device to place its unique 8-bit vector or address on the data bus.
4. Note that the $\overline{M1}$ line is active during M1 cycle, and when $\overline{IORQ}$ is simultaneously active (during the second half of TW1 and TW2), it indicates that the interrupt is acknowledged and that the interrupting device's identifying vector can be placed on the data bus.
5. During the M1 cycle, the $\overline{MREQ}$ signal is inactive.

5-7 NONMASKABLE INTERRUPT REQUEST/ACKNOWLEDGE CYCLE

1. This interupt has a higher priority over the $\overline{INT}$ signal and can override it. It cannot be disabled under software control. The principal use of this interrupt is

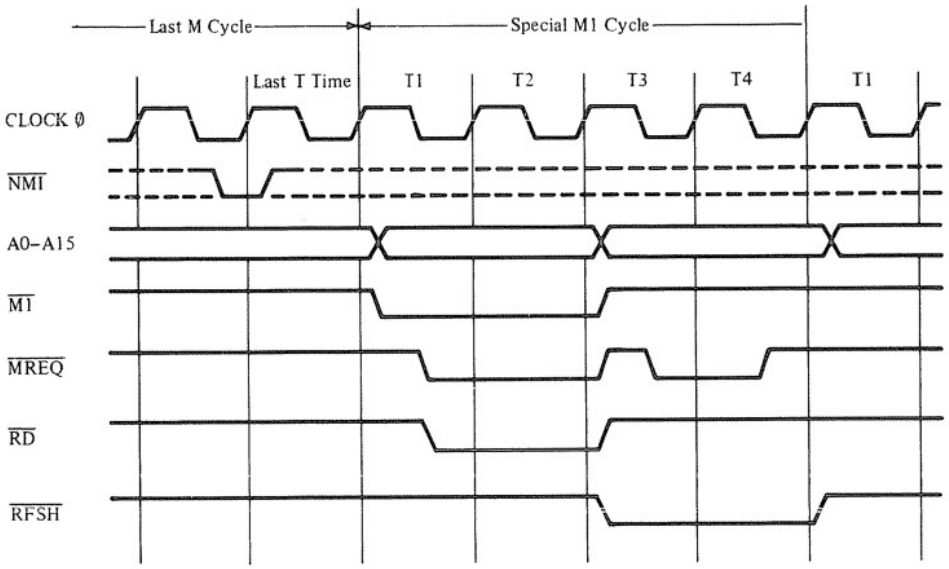


Figure 5-10 Nonmaskable Interrupt/Request Cycle

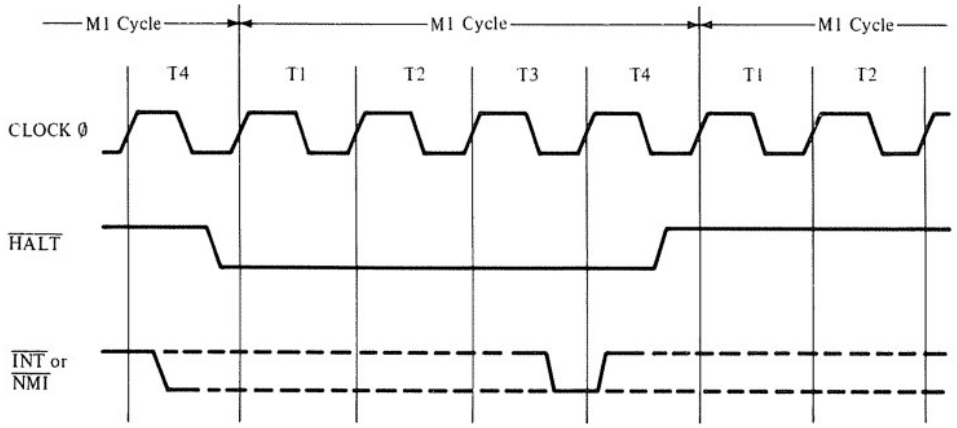


Figure 5-11 Exit from HALT Cycle

to indicate an impending power failure to the CPU, which immediately responds to it by taking the following action:

   a. The response is very similar to a normal memory read operation, except that the CPU ignores the information on the data bus.
   b. The contents of the PC are automatically pushed into the stack.
   c. The program automatically branches to a service subroutine whose starting address is 0066_H.

2. The $\overline{NMI}$ line is sampled by the CPU at the leading edge of the last clock pulse of the last M cycle, as shown in Fig. 5-10. If it is low, then the CPU starts a special M1 cycle at the leading edge of the next clock pulse.
3. This cycle consists of four T states. The operations of the various signals are similar to those which were described in previous sections and so will not be repeated here.

5-8 EXIT FROM HALT INSTRUCTION CYCLE

1. When the CPU encounters a HALT instruction in the program, it starts to execute a series of NOP instructions which in effect idle the CPU.
2. The CPU continues to idle until it receives an interrupt from an external I/O device. The interrupt could be either a maskable interrupt, $\overline{INT}$, or a nonmaskable interrupt, $\overline{NMI}$. If an $\overline{INT}$ signal comes along, then the software interrupt disable flip-flop must be set; otherwise the CPU will not honor the $\overline{INT}$.
3. Both the $\overline{INT}$ and the $\overline{NMI}$ lines are sampled by the CPU at the leading edge of the T4 clock pulse, as shown in Fig. 5-11. If either $\overline{NMI}$ or $\overline{INT}$ with the IFF enabled is received, the CPU will exit from or terminate the HALT state and resume operation at the leading edge of the next clock pulse T1. The following cycle will then be an interrupt acknowledge cycle, depending on whether the $\overline{INT}$ or the $\overline{NMI}$ signal was received. (See Table 1-3.)
4. If both $\overline{INT}$ and $\overline{NMI}$ are simultaneously active, as shown in Fig. 5-11, then only the $\overline{NMI}$ will be acknowledged, since it has a higher priority than $\overline{INT}$.
5. During the HALT state, NOP instructions are continuously executed. This is done so that the memory refresh signals are kept active for the dynamic memories while the CPU is idling. Otherwise the data in the memory chips will be destroyed.
6. Also note that the HALT cycle is executed during the M1 cycle, which is normally the OP code fetch cycle. However, during the HALT cycle, the CPU ignores any data placed by the memory on the data bus and the CPU is forced to execute NOP instructions.

5-9 REVIEW QUESTIONS

5-1 In the Z80 CPU the clock period is defined as:
   a. The clock frequency.
   b. The T cycle.
   c. The M cycle.
   d. The clock ø.

5-2 In the Z80 CPU the clock period is the time interval between
   a. The leading edge of one negative-going pulse to the leading edge of the next negative-going pulse.
   b. The trailing edge of one positive-going pulse to the trailing edge of the next positive-going pulse.
   c. The trailing edge of one positive-going pulse to

the leading edge of the next negative-going pulse.

d. The leading edge of one positive-going pulse to the leading edge of the next positive-going pulse.

5-3 Indicate true or false for the Z80 CPU.

a. ___ Each basic operation normally takes three to six clock periods to complete.

b. ___ The time required to perform the basic operations cannot be extended beyond six clock periods.

c. ___ More clock periods are required to synchronize the slower peripherals with the faster CPU.

d. ___ The instruction cycle is broken down into subcycles called the M cycles.

5-4 Answer yes or no for the Z80 instruction cycle.

a. ___ Do all instructions contain the same number of M cycles?

b. ___ Can the M cycles contain different numbers of T cycles?

c. ___ Is it possible to lengthen the M1 cycle?

5-5 Indicate true or false for the normal instruction cycle.

a. ___ Normally, M1 consists of three to five T cycles.

b. ___ M1 can be lengthened by a wait signal.

c. ___ M2 and M3 are normally four to six T cycles long.

d. ___ M2 and M3 *cannot* be lengthened by a WAIT signal.

e. ___ M2 and M3 transfer data between I/O and CPU.

5-6 Fill in the blanks for the OP code fetch cycle.

a. The M1 cycle contains _____ T cycles. (3; 4; 5; 6)

b. The first step transfers the contents of the _____ to the memory. (ALU; CPU; PC; I/O)

c. The address is activated during the _____ _____ clock periods. (T1 and T2; T2 and T3; T1 and T3; T3 and T4)

d. During T3 and T4, the _____ lines of the address bus send the memory refresh address. (upper 8; lower 7; lower 8; upper 7)

5-7 During an OP code fetch cycle, which signal is sent by the CPU to the memory to inform that the bits on the address bus are valid address bits?

a. IORQ
b. RD
c. MREQ
d. RFSH

5-8 Which signal from the CPU informs the memory that the OP code fetch is a memory read operation?

a. RFSH
b. RD
c. WAIT
d. MREQ

5-9 Refer to Fig. 5-2. The MREQ signal is active (i.e., goes low) during the second half of T1 because

a. of an error in the logic design of the CPU.
b. the M1 signal is activated during the first half of T1.
c. the RD signal is activated during the second half of T1.
d. it gives the address bus ample time to stabilize during the first half of T1.

5-10 Refer to Fig. 5-2. Indicate true or false for the M1 signal.

a. ___ It deactivates the address bus A0–A15.

b. ___ It activates the data bus D0–D7.

c. ___ It *does not* signal a write operation to the memory chips.

d. ___ It *indicates* that the current cycle is an OP code fetch cycle.

5-11 Refer to Fig. 5-2 and answer yes or no.

a. ___ Is it possible to generate M1 more than once?

b. ___ Is it possible to have more than 1 byte in the OP code?

c. ___ Is it possible to place the memory refresh address on the address bus, A0–A15, during M1 of the OP code fetch cycle?

5-12 During the OP code fetch cycle, the refresh address is stable on the address bus when

a. MREQ is active
b. MREQ is inactive
c. RD is active
d. M1 is active

5-13 Indicate true or false for the OP code fetch operation with WAIT states. (Fig. 5-3)

a. ___ During T2 the CPU samples WAIT line.

b. ___ If the WAIT line is high, the CPU then goes into a wait state for the next T period.

c. ___ When WAIT is low, all CPU signals are maintained in their respective positions as at the end of T1.

d. ___ During every subsequent TW cycle, the CPU samples the WAIT line.

e. ___ T3 and T4 are resumed only when WAIT is high during the last TW period.

5-14 The memory read cycle requires _____ clock periods and the memory write cycle requires

_____ clock periods. (two; two or three; three; three to five; two to four).

5-15 The following statements relate to the memory read/write cycles without $\overline{\text{WAIT}}$ states. Indicate true or false.

a. ___ $\overline{\text{MREQ}}$ is active during T1 and T2 but inactive during T3.

b. ___ $\overline{\text{MREQ}}$ can be directly used as a chip enable signal in dynamic memory chips.

c. ___ The $\overline{\text{WR}}$ signal informs the memory that the CPU has placed valid data on the data bus.

d. ___ It is possible to use $\overline{\text{WR}}$ as $\overline{\text{R/W}}$ signal for almost any semiconductor chip.

5-16 Refer to Fig. 5-5. During the read operation, data are placed on the data bus only during T3. For the write operation, data are placed on the data bus during the second half of T1, T2, through the wait states and T3. The reason for this is that

a. The data bus is not busy with other things during the write operations.

b. Address bus is tied up with the OP code's address during the read memory operation.

c. It assures sufficient time for the memory to write into the addressed location.

d. The $\overline{\text{WR}}$ line is high during this time frame.

5-17 The following statements relate to the I/O read/write cycles without WAIT states. (Fig. 5-6). Indicate true or false.

a. ___ The $\overline{\text{IORQ}}$ signals that the bits on the address bus are valid address bits.

b. ___ The $\overline{\text{IORQ}}$ indicates valid I/O address on the lower 8-bit lines of the address bus.

c. ___ The TW0 cycle is intentionally introduced in the I/O read/write cycle.

d. ___ The TW0 cycle helps with the speed incompatibility problem when slower speed MOS technology is used for I/O devices.

5-18 During both the memory and the I/O data transfers, the $\overline{\text{WAIT}}$ line must go low during the _____ edge of the clock pulse for the CPU to go into the wait state during the next clock period. (leading; trailing)

5–19 The following statements relate to the bus request/acknowledge cycles. Indicate true or false.

a. ___ The $\overline{\text{BUSRQ}}$ signal is sent by the CPU.

b. ___ The CPU samples the $\overline{\text{BUSRQ}}$ line at the leading edge of rise time of the clock pulse during the last T period of every M cycle.

c. ___ If $\overline{\text{BUSRQ}}$ is active, the CPU relinquishes control of the buses immediately in the same T cycle.

d. ___ During bus request cycles, both $\overline{\text{INT}}$ and $\overline{\text{NMI}}$ are deactivated.

e. ___ The bus request cycle is terminated by the I/O device by putting the $\overline{\text{BUSRQ}}$ signal in the high state.

5-20 A maskable interrupt refers to the condition where the interrupt _____ be overriden by means of the CPU software. (can; cannot)

5-21 Refer to Fig. 5-9 for the maskable interrupt request/acknowledge cycle and indicate true or false.

a. ___ The CPU samples the $\overline{\text{INT}}$ line at the leading edge of the last clock pulse of all instructions.

b. ___ If $\overline{\text{INT}}$ is low the CPU goes into the special M1 cycle, starting with the trailing edge of the next clock pulse.

c. ___ Wait states TW1 and TW2 are automatically inserted during interrupt request cycles.

d. ___ TW1 and TW2 ensures adequate time for the interrupting device to place its 8-bit vector on the data bus.

5-22 The _____ signal indicates that the interrupt is acknowledged and that the interrupting device can place its identifying vector on the data bus. ($\overline{\text{IORQ}}$; $\overline{\text{MREQ}}$; $\overline{\text{INT}}$)

6

Z80 PARALLEL INPUT/OUTPUT CHIP

CHAPTER OBJECTIVES

1. To introduce students to the programmable parallel I/O chip.
2. To present the principal features of the Z80 parallel Input/Output (PIO) chip.
3. To present and describe the generalized architecture of the PIO.
4. To present the various registers involved in the logic of each I/O port and describe the functions performed by each logic block.
5. To present the interrupt control protocol involved in PIO operations.
6. To present the package pin assignments and functions in convenient tabular form.
7. To describe and discuss the interrupt timing associated with all three operational modes.
8. To present the timing involved in interrupt acknowledge and interrupt servicing.
9. To show how the daisy chain can be extended without using additional external logic.

TEXTBOOK REFERENCES

For reviewing material in the main textbook, relevant to the topics covered in this chapter, the following chapters and/or sections are suggested:

1. For general discussion of interrupt I/O	Chapter 9
2. For daisy chain logic	Sec. 9-7
3. For vectored-interrupt sequence of events	Sec. 9-5.2
4. For vectored-interrupt logic/operation	Sec. 9-5.3
5. For general description of programmable I/O interfaces	Chapter 12
6. For features and capabilities of programmable parallel interfaces	Secs. 12-3.1 and 12-3.2

6-1 INTRODUCTION

The Parallel Input-Output (PIO) chip provides a programmable, two-port interface between parallel peripheral I/O devices and the Z80 CPU. Under program control the CPU is capable of reconfiguring the PIO so that it can interface with a wide variety of peripherals such as paper tape readers, most keyboards, printers, etc., without requiring additional external logic. Data transfers are accomplished by interrupt-controlled techniques which allow the external I/O devices to initiate such transfers. An interrupt priority system is provided using the daisy chain technique.

6-2 PRINCIPAL FEATURES OF THE PIO

1. The PIO is a third-generation chip which uses the N-channel, depletion-mode, silicon-gate technology.
2. A single, 5-volt ±5%, power supply is required.
3. A single-phase TTL level clock is used.
4. The PIO chip operates in a temperature environment of 0° to 70°C.

5. On the peripheral side, all inputs as well as the outputs are TTL compatible.
6. Two I/O ports are provided and they are completely independent of each other.
7. There are four distinct and separate modes of port operations. Both ports, A and B, have the following three modes (a, b, and c):
   a. Byte output—unidirectional (mode 0).
   b. Byte input—unidirectional (mode 1).
   c. Individual bits control—input or output (mode 3).
   d. Additionally, port A has a bidirectional byte control mode (mode 2).
8. Both ports have interrupt-driven handshake capability (i.e., status control signals).

6-3 THE PIO ARCHITECTURE

6-3.1 The Generalized PIO Block Diagram

1. Figure 6-1 shows a generalized block diagram of the PIO chip.
2. On the computer side, the chip provides I/O interface logic for transferring data on an 8-bit data bus D0–D7, and a 6-bit control bus is also used.
3. The port A I/O and the port B I/O logic each communicates with the external I/O device via an 8-bit data bus, respectively, A0–A7 and B0–B7. These lines could be either unidirectional or bidirectional, depending on the selected operational mode.
4. Each port also has two handshake lines to their respective I/O devices.
5. Also on the computer side, the PIO communicates with the CPU via three interrupt control lines, all of which are unidirectional.
6. The primary function of the internal control logic block is to synchronize the CPU data bus to the interfaces involved in each I/O port. In other words, synchronize the data flow between the CPU and the I/O devices.
7. The interrupt control logic primarily deals with interrupt operations and the daisy chain priorities.

6-3.2 The Port Logic Block Diagram

1. Figure 6-2 is a block diagram of the block labeled port A or port B in the generalized PIO block diagram of Fig. 6-1.
2. Figure 6-2 shows six registers and the logic block for the handshake control.
3. The mode control register is first loaded by the CPU with a 2-bit code which selects one of the four operating modes mentioned in Sec. 6-2, paragraph 7. This code sets up the logic configuration for the desired PIO operating mode.
4. The 8-bit data output register receives the byte from the CPU and sends it out of the PIO chip to the I/O device via the 8-bit data/control bus.
5. Likewise, the 8-bit data input register receives the incoming byte from the I/O device via the 8-bit data/control bus and transmits it to the CPU via the internal bus.
6. The actual data transfers, between the CPU and the I/O devices, are controlled by the two status control lines via the handshake control logic. The exact functions of these two lines are explained later.
7. The 8-bit I/O select logic is used only in the bit control mode (mode 3). This register is first loaded by the

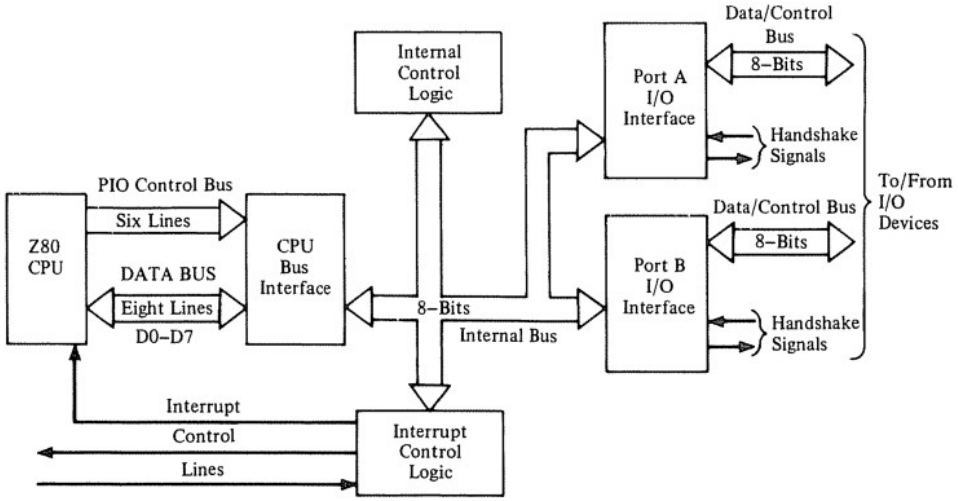


Figure 6-1 Generalized PIO Block Diagram

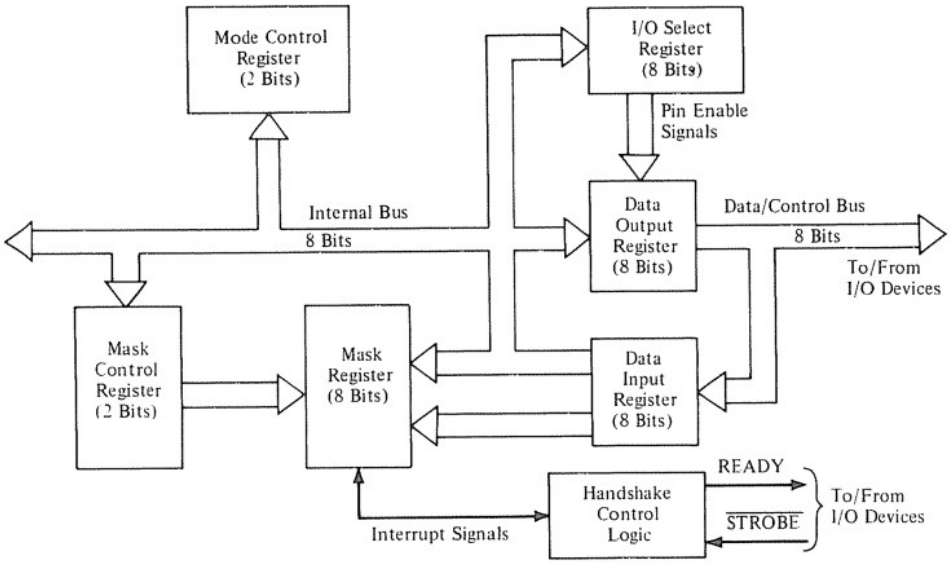


Figure 6-2 Port Logic Block Diagram

CPU with a control word. This word then assigns the eight lines of the I/O data/control bus individually as either an input or an output line. The bus then becomes a combination of designated input and output lines.

8. The 8-bit mask register is, likewise, used only in the bit control mode (mode 3). It is used for a very special interrupt feature of the Z80 system. Again, this register is preloaded with a word from the CPU. The word in this register effectively masks out the selected I/O pins of the I/O data control bus. Then, when any or all of the unmasked pins reach a prespecified logical state (either high or low), an interrupt is generated.
9. The 2-bit mask control register, which is also preloaded by the CPU, defines the desired active states (either high or low) for the unmasked bits of the mask register. It also performs another function. It defines whether the interrupt should be generated when all the unmasked bits of the bus are in the defined state (the AND function) or when any one or more of the unmasked bits is in the active state (the OR function).

6-3.3 The Interrupt Control Protocol

1. Since the daisy chain technique is used, the priority of each I/O device is established by its position in the chain relative to the CPU. The device closest to the CPU has the highest priority.
2. In the PIO, port A has a higher priority than port B.
3. When the PIO is operated in either mode 0, 1, or 2, an interrupt request is sent to the PIO when the I/O device is ready for a data transfer.
4. When the PIO is operated in mode 3 (bit control mode), an interrupt can be generated only when the state of the I/O device matches a programmed value.

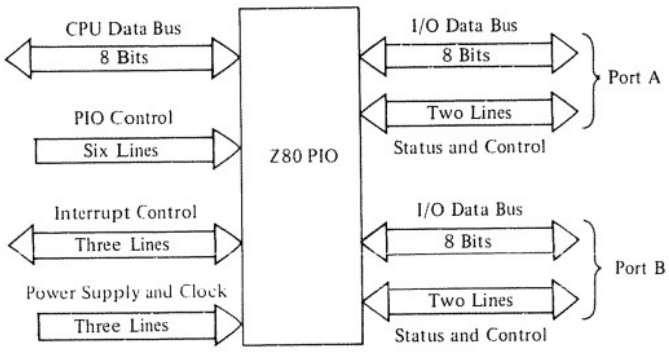


Figure 6-3 Block diagram of CPU Package Pins Groups

5. In the PIO, interrupts can be nested (i.e., multilevel priority interrupt capability is available). An I/O device with a lower priority on the daisy chain cannot interrupt while the service subroutine of a device with a higher priority is being executed. However, a device with a higher priority can interrupt during the subroutine execution of an I/O device having a lower priority on the daisy chain.
6. When operating in mode 2 (bidirectional byte transfer mode), the interrupting I/O sends an 8-bit vector (i.e., its own unique address) to the CPU. This vector forms a pointer. This points to a location in the memory where the starting address of the interrupt service subroutine is stored. (See Sec. 5-4.3.)

6-4 PACKAGE PIN ASSIGNMENTS AND FUNCTIONS

The PIO is contained in a standard 40-pin, dual-in-line package (DIP). The various pins and their respective functions, along with their designations, are presented in convenient tabular form. Functionally, the pins can be divided into six groups, as shown in Fig. 6-3. Note that some lines are shown as unidirectional and others as bidirectional. In the latter group, some of the lines may indicate signal flow in one direction, and others may indicate signal flow in the other direction. The function tables, Tables 6-1 through 6-5, clarify these with the following arrow symbols:

PIO → Indicates signals flowing *from the PIO*
PIO ← Indicates signals flowing *into the PIO*
PIO ↔ Indicates *bidirectional* flow on the same line(s)

6-5 THE PIO INTERRUPT TIMING

In this section we discuss the timing involved in the four interrupt modes. In the following timing charts, notice that the clock phases are shown as perfect square waves. This is done only for convenience. Actually each clock pulse has its own rise and fall times. The rise and fall times of the other signals are shown as they should be. The clock pulses are used in these charts simply for reference purposes only and so they are shown as square or rectangular pulses.

6-5.1 Mode 0—Output Byte Mode

1. The timing associated with this mode is shown in Fig. 6-4. A byte output cycle is started when, as a result of an interrupt, the CPU executes an output instruction.
2. During this CPU I/O write operation, the PIO generates a $\overline{WR}$ signal. This is a composite signal obtained by mixing (i.e., ANDing) four signals, as follows:

$$\text{(Write data) } \overline{WR} = RD \times \overline{CE} \times \overline{C/D} \times \overline{IORQ}$$

3. The $\overline{WR}$ pulse latches the data output by the CPU on the data bus into the output register of the selected port (A or B).
4. The positive-going trailing edge of the $\overline{WR}$ pulse then raises the Ready flag after the trailing edge of the next clock pulse (i.e., pulse 4 in Fig. 6-4). This flag then signals to the I/O device that the data word is now available to it.
5. The Ready signal remains high or active until one of the following two events takes place:
   a. A positive-going trailing edge of the $\overline{STROBE}$ signal indicates that the I/O device has accepted and obtained the data.
   b. If the READY is already active, then, the Ready signal will be forced into the low state one and a half clock pulses after the leading edge of the $\overline{IORQ}$ signal if the word is written into the selected port's output register.
6. The READY signal will go high on the first trailing edge of the clock pulse after the trailing edge of $\overline{IORQ}$. This then assures that the READY signal is low when the data in the selected port is in the process of changing.

Table 6-1 PIO DATA BUSES

Pin Designations	*Signal Flow Directions*	*Functions*
D0–D7	PIO ⟷	8-bit, tristate, CPU data bus, active high. It is used for data and commands transfers between the PIO and the CPU.
A0–A7	PIO ⟷	8-bit, tristate bus for transfers of data and/or status and control signals between the I/O device and port A.
B0–B7	PIO ⟷	8-bit, tristate bus for transfer of data and/or status and control signals between the I/O device and port B of PIO.

Table 6-2 PIO CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
B/$\overline{A}$ Sel	PIO ←	Port select signal defines the port to be selected during a data transfer operation. A low signal selects port A and a high selects port B. (Note: Sometimes *address bit* A0 from the CPU is used for this selection.)
C/$\overline{D}$ Sel	PIO ←	Control or data select signal defines whether data or control information is transferred between the CPU and the port selected by the B/$\overline{A}$ Sel signal. A low signal indicates that data bits are transferred between the CPU and the PIO. A high during CPU write indicates that a command or control word is sent to the port selected by the B/$\overline{A}$ Sel signal. (*Note:* Sometimes *address bit* A1 from the CPU is used for this function.)
$\overline{CE}$	PIO ←	An active low signal that informs the PIO to transmit the data word to the CPU during the CPU read cycle. During the CPU write cycle, this signal informs PIO to accept either the control or the data word from the CPU as defined by the C/$\overline{D}$ Sel signal.
$\overline{M1}$	PIO ←	Machine cycle 1 signal, active low, is a SYNC pulse from the CPU which synchronizes the PIO interrupt logic. When $\overline{M1}$ and $\overline{RD}$ are simultaneously active, the CPU is fetching an instruction from the memory. When $\overline{M1}$ and $\overline{IORQ}$ are simultaneously active, the CPU is acknowledging an interrupt. When $\overline{M1}$ is active without the other two being active, the PIO goes into a reset state.
$\overline{IORQ}$	PIO ←	I/O request signal, active low, used for transferring commands and data between the CPU and the PIO. When $\overline{CE}$, $\overline{RD}$ and $\overline{IORQ}$ are active, the addressed port sends data to the CPU. When $\overline{CE}$ and $\overline{IORQ}$ are active and $\overline{RD}$ is not active, the CPU writes either a data or a command word into the addressed port of the PIO. If $\overline{IORQ}$ and $\overline{M1}$ are active, the CPU acknowledges an interrupt and the port involved will place its interrupt vector on the CPU data bus, depending on its priority in the daisy chain.
$\overline{RD}$	PIO ←	Read cycle status, active low, signal indicates that the CPU is reading a word from either the I/O or the memory. When used in conjunction with the $\overline{CE}$, $\overline{IORQ}$, B/$\overline{A}$ sel, and C/$\overline{D}$ sel signals, the $\overline{RD}$ transfers data from the PIO to the CPU.

Table 6-3 INTERRUPT CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
$\overline{INT}$	PIO ⟶	Interrupt request signal, active low. The PIO requests an interrupt from the CPU.
$\overline{IEI}$	PIO ⟵	Interrupt enable in, active high. Used in the priority interrupt daisy chain. A high or active signal indicates that no other I/O device with higher priority is being currently serviced.
$\overline{IEO}$	PIO ⟶	Interrupt enable out, active high. Also used in the priority interrupt daisy chain. It is active only when IEI is active and the CPU *is not* servicing an interrupt from the PIO. When active, it blocks interrupt signals from lower-priority I/O devices from interrupting while it is being serviced.

Table 6-4 PORT STATUS AND CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
$\overline{\text{A STB}}$	PIO ⟵	Port A strobe signal, active low, input to the PIO from the I/O device. Operation depends on selected mode as follows: 1. Mode 0—output byte mode—a positive-going, trailing edge of this pulse indicates that the I/O has received the data sent by the PIO. 2. Mode 1—input byte mode—the I/O issues this signal, indicating that the data are sent to the input register of the selected PIO port. 3. Mode 2—bidirectional mode—when active indicates that the data from the output register of port A are put on their own bidirectional data bus. Then the positive-going, trailing edge of this pulse indicates that the data are received by the I/O. 4. Mode 3—control mode—this pulse is internally inhibited by the PIO.
A RDY	PIO ⟶	Register A ready signal, active high. Operation depends on selected mode as follows: 1. Mode 0—output byte mode—when active indicates that the port A output register contains the data word, the data bus has stabilized ready for transfer to the I/O device. 2. Mode 1-input byte mode—when active indicates that the input register of port A is empty and ready to accept the next word from the I/O device. 3. Mode 2—bidirectional mode—when active indicates that the output register of port A has the data word for transfer to the I/O device. However, data are not output to the data bus unless A STB is active. 4. Mode 3—control mode—this signal is internally inhibited by the PIO.
$\overline{\text{B STB}}$	PIO ⟵	Port B strobe signal, active low. Its operation is similar to that of the $\overline{\text{A STB}}$ signal except that in the port A, mode 2 operation, this signal strobes data from the I/O device into the input register of port A.
B RDY	PIO ⟶	Register B ready signal, active high. Operation is similar to that of the A RDY signal except that in the port A mode 2 operation, this signal is active when the input register of port A is empty and ready to accept the next incoming data word from the I/O device.

Table 6-5 POWER SUPPLY AND CLOCK GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
ø	PIO ⟵	This is the single-phase, TTL level clock input.
+5 V	PIO ⟵	This is the single power supply of 5 volt ± 5% that is needed for the PIO.
GND	PIO ⟵	This is the power supply ground.

7. Notice that the READY signal will not go into the low or inactive state until the negative-going edge of the clock pulse happens. This delaying of the READY signal allows a simple scheme for generating the STROBE signal. (This is accomplished by simply connecting the READY line to the STROBE line without requiring any additional external logic.)

8. The positive-going trailing edge of the STROBE signal automatically generates an $\overline{\text{INT}}$ request provided the interrupt enable flip-flop has been set and the interrupting I/O device has the highest priority.

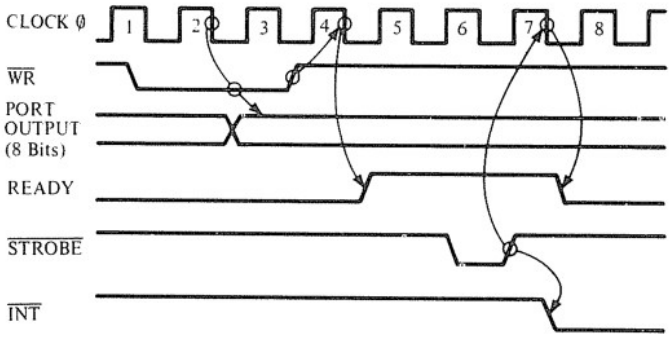


Figure 6-4 Mode 0, Byte Output Timing

Table 6-6 PIN NUMBER ASSIGNMENTS

Pin Designations	*Pin Numbers*	*Pin Designations*	*Pin Numbers*
CPU Data Bus		*PIO Control*	
D0	19	B/$\overline{\text{A}}$ Sel	6
D1	20	C/$\overline{\text{D}}$ Sel	5
D2	1	$\overline{\text{CE}}$	4
D3	40	$\overline{\text{M1}}$	37
D4	39	$\overline{\text{IORQ}}$	36
D5	38	$\overline{\text{RD}}$	35
D6	3		
D7	2		
Port A I/O		*Interrupt Control*	
A0	15		
A1	14	$\overline{\text{INT}}$	23
A2	13	IEI	24
A3	12	IEO	22
A4	10		
A5	9		
A6	8		
A7	7		
A RDY	18		
$\overline{\text{A STB}}$	16		
Port B I/O		*Power Supply and Clock*	
B0	27		165
B1	28	ø	25
B2	29	+5 V	26
B3	30	GND	11
B4	31		
B5	32		
B6	33		
B7	34		
B RDY	21		
$\overline{\text{B STB}}$	17		

6-5.2 Mode 1—Input Byte Mode

1. The timing for this mode is shown in Fig. 6-5.
2. The input cycle is initiated by the I/O device by means of a negative-going pulse on the $\overline{\text{STROBE}}$ line. This pulse loads the data word into the input register of the selected port.
3. The positive-going trailing edge of the $\overline{\text{STROBE}}$ pulse activates the $\overline{\text{INT}}$ signal provided the interrupt enable flip-flop is set and the interrupting I/O device has the highest priority.
4. The trailing edge of the next clock pulse (pulse 3 in Fig. 6-5) puts the READY line in the low or inactive state. This signifies that the input register in the PIO contains a data word and that further loading of this register must be prevented.
5. The CPU executes the service subroutine and reads in the data from the selected I/O port in the PIO.
6. On completion of this operation, the positive-going, trailing edge of the $\overline{\text{RD}}$ signal (from the CPU) raises the READY signal to the high level after the negative-going transition of the next clock pulse (pulse 11 in Fig. 6-5). This indicates to the I/O device that the next word can be loaded into the input register of the PIO.
7. If the READY line is already active, then this line will be forced into the low state one and a half clock periods following the leading edge of the $\overline{\text{IORQ}}$ signal.

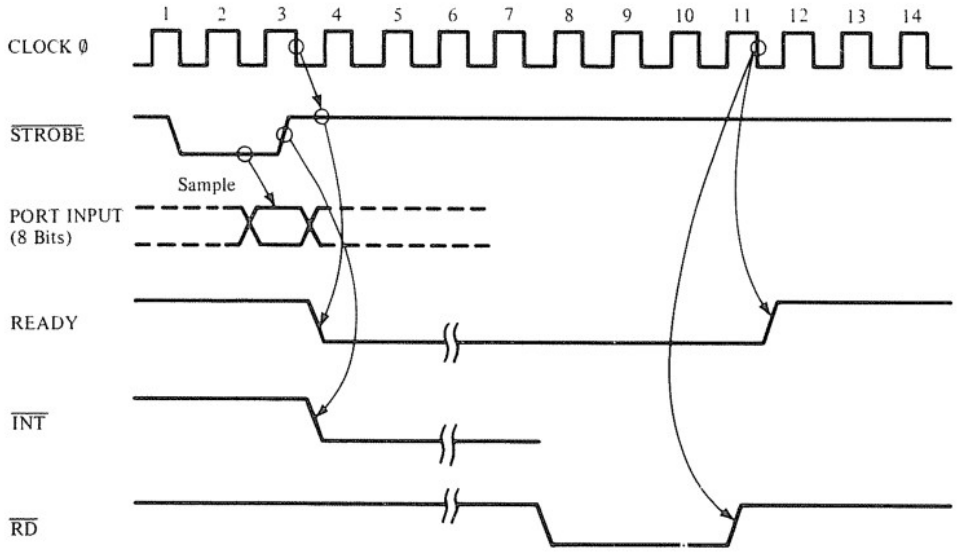


Figure 6-5 Mode 1, Byte Input Timing

8. Note that the $\overline{RD}$ is a composite signal obtained by mixing the following four signals:

$$\text{(Read data) } \overline{RD} = \overline{RD} \times \overline{CE} \times \overline{C/D} \times \overline{IORQ}$$

6-5.3 Mode 2—Bidirectional Mode

1. Figure 6-6 shows the timing of this mode, which is a combination of modes 0 and 1. It utilizes all four handshake lines and is available for only port A. When port A is operated in mode 2, port B must be set to mode 3.
2. The timing for mode 2 is almost identical to the timing for mode 0 (Fig. 6-4) and mode 1 (Fig. 6-5). In mode 2 operation, the handshake lines of port A are used for data output control and port B handshake control lines are used for input control.
3. In mode 2 operation, the data word is gated out to the bus only when the $\overline{A\ STB}$ is active (i.e., low). The positive-going, trailing edge of this signal latches the data word into the I/O device.
4. In mode 2 the data input operates in an identical manner to the mode 1 operation. Also, in mode 2, both ports A and B must have their interrupt enables set so that bidirectional transfers are truly interrupt-driven.
5. The data input into the PIO are latched into the input register when the $\overline{B\ STB}$ is active or low. This avoids the possibility of the data being gated onto the bus by the I/O device when the A RDY is active or high.
6. The $\overline{WR}$ is a composite of the following signals:

$$\overline{WR} = RD \times \overline{CE} \times \overline{C/D} \times \overline{IORQ}$$

6-5.4 Mode 3—Bit Control Mode

1. In mode 3 the handshake signals are not used, so it is possible to execute a normal port read or a normal port write any time.
2. When writing into the port, the data word is latched into the output register of the selected port using the same timing as that for mode 0, byte output operation as shown in Fig. 6-4. Mode 3 timing is shown in Fig. 6-7.
3. When port A is operated in mode 3, the A RDY line is forced into a low state. When port B is operated in mode 3, the B RDY line is held low except when port A is operated in mode 2. In that case B RDY is not affected at all.
4. During the PIO read operation in mode 3, the data transmitted to the CPU by the PIO will be composed of the following two segments:
   a. Bits from the output register whose lines are designated as output lines.
   b. Bits from the input register whose lines are designated as input lines.
5. Note that the input register will contain the word which was in it immediately before the negative-going edge of the $\overline{RD}$ signal.
6. Remember that an interrupt will be generated only if
   a. The interrupts from the port are enabled.
   b. The word on the port data line conforms to the logical conditions that were preloaded into the 8-bit mask register and the 2-bit mask control register of the PIO.
7. Another interrupt signal will not be generated by the PIO until after a change in the preceding logical condition occurs.

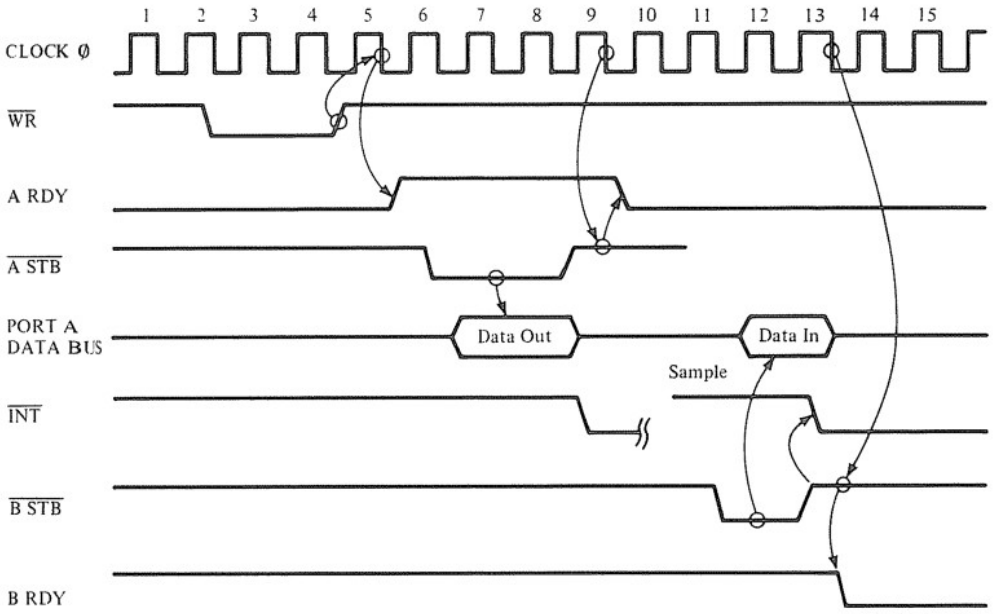


Figure 6-6 Mode 2, Port A, Bidirectional Byte Transfer Timing

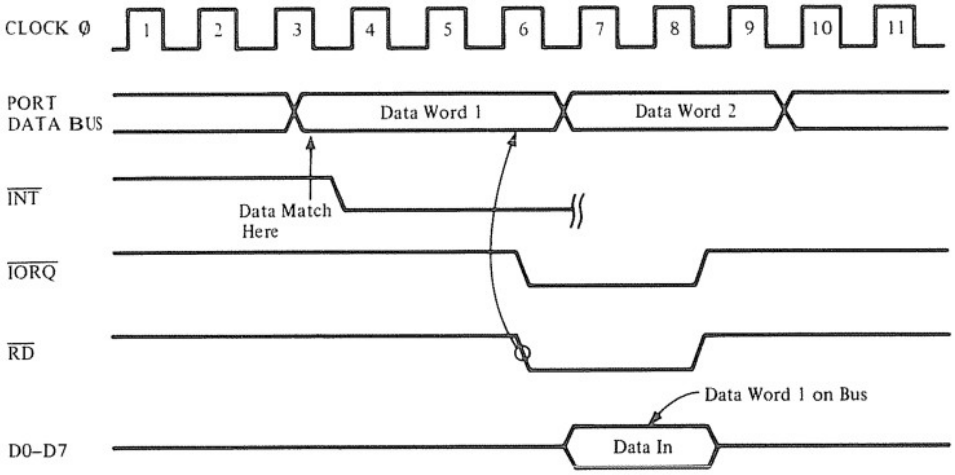


Figure 6-7 Mode 3, Bit Control Mode Timing

6-5.5 Interrupt Acknowledge and Servicing

In the Z80 system it is normally possible to form a daisy chain by connecting up to four PIO chips in tandem or in series without the need for any additional external logic. Since two I/O devices can be connected to each PIO chip, this means that up to eight I/O devices can be normally used on the daisy chain. Later we will see how the daisy chain can be extended even further. However, in order to understand the limitations and how to overcome them, we will first see how the interrupt acknowledge works and how an interrupt is serviced.

1. Figure 6-8 shows the timing involved in the interrupt acknowledge signal, $\overline{\text{INTA}}$, which is a composite of two signals, as shown below:

$$\overline{\text{INTA}} = \overline{\text{IORQ}} \times \overline{\text{M1}}$$

2. The $\overline{\text{INT}}$ line is sampled by the CPU during the negative-going trailing edge of the last clock pulse of

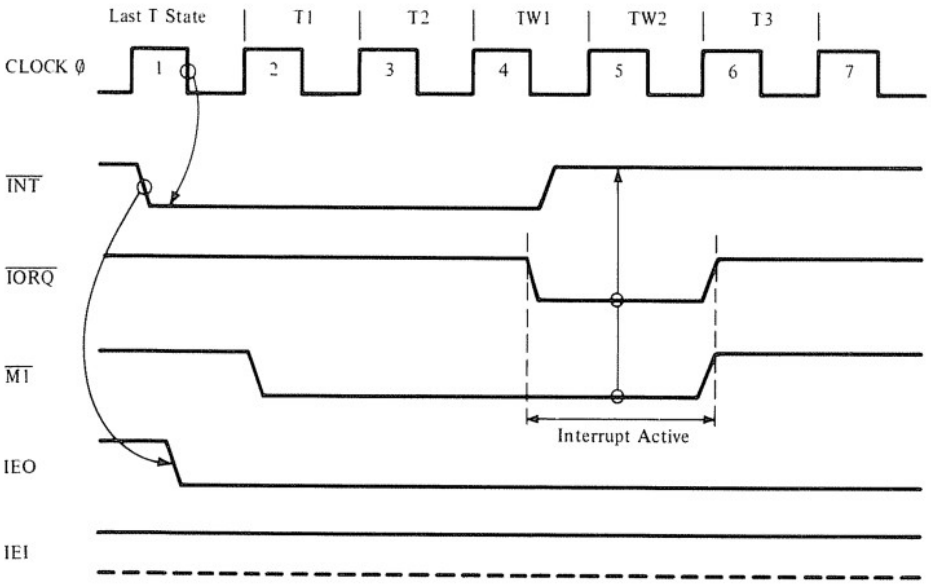


Figure 6-8 Interrupt Acknowledge Timing

the instruction. If it is active (i.e., low), it goes into an interrupt cycle, as shown in Fig. 6-8.

3. At the leading edge of T1, $\overline{M1}$ goes low or active. During the trailing edge of the first wait period clock pulse, TW1, $\overline{IORQ}$ goes low. Both these signals remain active or low for one and a half clock periods. They go high at the leading edge of the T3 clock pulse. Thus, during this one and a half clock period, $\overline{INTA}$ is active.
4. During $\overline{INTA}$ time, the PIO checks the highest priority port which is requesting the interrupt by determining which device has its IEI high and its IEO low.
5. The highest priority I/O device that is requesting an interrupt service places the word contained in its interrupt vector register on the CPU data bus, thereby identifying itself.
6. During the time when $\overline{M1}$ is active, no new interrupt requests are generated, thus giving the $\overline{INTA}$ composite signal ample time to ripple through up to four PIO daisy chain configurations.
7. Figure 6-9 shows how up to four chips can be connected to form the daisy chain.

6-5.6 Extending the Daisy Chain

1. Figure 6-9 shows that up to four PIO chips can be connected without any additional external logic.
2. This limitation is designed so that the IEO will have adequate time to ripple through the entire chain in the time available between the start of the $\overline{M1}$ signal and the leading edge of the $\overline{IORQ}$ signal during the $\overline{INTA}$ cycle. (See Fig. 6-8.)
3. In many applications it is desirable to extend the chain beyond the simple four shown in Fig. 6-9. It is possible to extend this to 30 PIO chips or more, using standard TTL logic, by means of the look-ahead logic structure shown in Fig. 6-10, which requires two-input and four-input AND gates connected as shown.

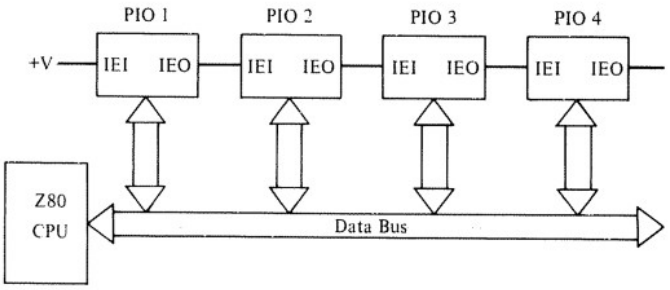


Figure 6-9 Maximum PIO Daisy Chain without Additional Logic

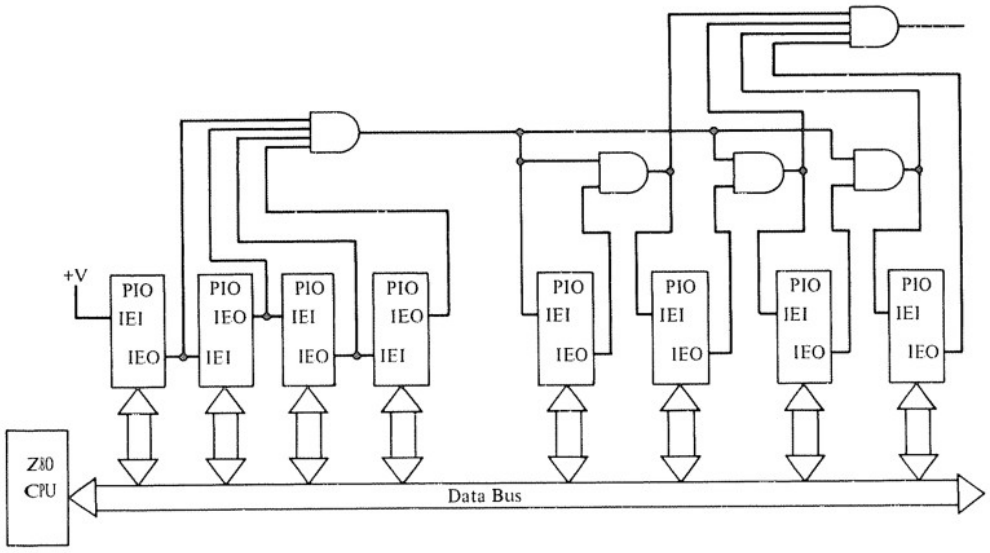


Figure 6-10 Extending the Interrupt Priority Daisy Chain with Standard TTL Logic

6-6 REVIEW QUESTIONS

6-1 In the Z80 system, data transfers between the CPU and the I/O devices are initiated by
- **a.** The CPU.
- **b.** The I/O devices.
- **c.** The PIO.

6-2 Indicate true or false for the PIO.
- **a.** ___ The PIO can be reconfigured under program control.
- **b.** ___ The daisy chain technique is used to provide an interrupt priority system.
- **c.** ___ Both ports A and B have a bidirectional byte transfer mode (mode 2).
- **d.** ___ Both ports A and B have handshake capabilities.

6-3 Refer to the generalized block diagram of Fig. 6-1 and indicate true or false.
- **a.** ___ Control information is transferred from the CPU to the PIO on an 8-bit control bus.
- **b.** ___ Lines A0–A7 are bidirectional byte lines and B0–B7 are unidirectional lines.
- **c.** ___ The PIO communicates with the CPU via three unidirectional interrupt control lines.
- **d.** ___ The internal control logic block synchronizes the data flow between the CPU and the I/O devices.

6-4 Refer to the port logic block diagram of Fig. 6-2. The *operating mode* of the PIO is (circle all that apply)
- **a.** Determined by the type of I/O used by that port.
- **b.** Determined by the CPU.
- **c.** Established by preloading a control word into the 8-bit I/O select register.
- **d.** Established by means of a 2-bit code sent by the CPU.

6-5 Again refer to Fig. 6-2. The 8-bit I/O select logic is used in the
- **a.** Bit control mode (mode 3).
- **b.** Output byte mode (mode 0).
- **c.** Bidirectional byte mode (mode 2).
- **d.** Input byte mode (mode 1).

6-6 The following statements relate to the mask control register of Fig. 6-2. Fill in the blanks.
- **a.** It is preloaded with a 2-bit code from the _____. (CPU; I/O device)
- **b.** It defines the desired active states for the _____ bits of the mask register. (masked; unmasked)
- **c.** It defines that the interrupt should be generated by the AND function, that is, when _____ of the unmasked bits are in the _____ state. (all; some; active; inactive; defined).

6-7 The following statements relate to the interrupt control protocol in the PIO. Indicate true or false.
- **a.** ___ Port A has a higher priority than port B.
- **b.** ___ The device closest to the CPU has the lowest priority.
- **c.** ___ In the bit control mode, an interrupt can be generated only when the I/O device matches a programmed value.
- **d.** ___ An I/O device can *never* interrupt during the subroutine execution of *any* other device on the daisy chain.

6-8 The tristate 8-bit data bus _____ is active high and is used for data and commands transfers between the CPU and the PIO. (A0–A7; B0–B7; D0–D7)

6-9 The $\overline{CE}$ signal is (circle all that apply)
- **a.** Active high and informs the PIO to transmit the data word to the CPU.
- **b.** Active low and informs the CPU to transmit the command word to the PIO.
- **c.** Active low and informs the PIO to transmit the data word to the CPU.
- **d.** Active high and informs the CPU to transmit the data word to the PIO.

6-10 The following statements apply to the $\overline{M1}$ signal. Indicate true or false.
- **a.** ____ It synchronizes the PIO interrupt cycle.
- **b.** ____ It is a SYNC pulse from the PIO to the CPU.
- **c.** ____ The CPU fetches an instruction from the memory when $\overline{M1}$ and $\overline{RD}$ are simultaneously active.
- **d.** ____ The CPU acknowledges an interrupt when $\overline{M1}$ and $\overline{IORQ}$ are simultaneously active.
- **e.** ____ When $\overline{M1}$ is active by *itself,* the PIO goes into the *set* state.

6-11 The following statements relate to the $\overline{A\ STB}$ signal when the PIO is operated in mode 2, bidirectional mode. Fill in the blanks.
- **a.** Indicates that the _____ word from the output register of port _____ is put on its own bidirectional data bus. (command; data; A; B)
- **b.** The _____ going, _____ edge of this pulse indicates that the data is received by the I/O device. (positive; negative; leading; trailing)

6-12 Refer to the mode 0, byte output, timing chart of Fig. 6-4. The WR is a composite of
- **a.** $RD \times \overline{CE} \times \overline{C/D} \times \overline{IORQ}$
- **b.** $\overline{RD} \times \overline{CE} \times \overline{C/D} \times IORQ$
- **c.** $RD \times \overline{CE} \times C/D \times IORQ$
- **d.** $\overline{RD} \times CE \times C/E \times \overline{IORQ}$

6-13 Again refer to Fig. 6-4. Indicate true or false for the READY signal.
- **a.** ____ The signal goes low on the first trailing edge of the clock pulse after the trailing edge of IORQ.
- **b.** ____ This signal is low when the data in the port are changing.
- **c.** ____ This signal will not go low until the negative-going edge of the clock pulse happens.
- **d.** ____ Positive-going, trailing edge of $\overline{STROBE}$ signal indicates that the I/O has obtained the data.

6-14 The following statements relate to the mode 1 timing chart of Fig. 6-5. Fill in the blanks.
- **a.** The _____ cycle is initiated by the negative-going pulse on the _____ line. (input; output; $\overline{INT}$; $\overline{STROBE}$)
- **b.** The positive-going, trailing edge of the $\overline{STROBE}$ signal activates the _____ signal. (READY; $\overline{INT}$; $\overline{RD}$)
- **c.** If the READY line is already active, it is forced into the low state one and a half clock periods following the leading edge of the _____ signal. ($\overline{INT}$; $\overline{IORQ}$; $\overline{STROBE}$; $\overline{RD}$)

6-15 Refer to the bidirectional byte timing chart of Fig. 6-6 and indicate true or false.
- **a.** ____ The positive-going, trailing edge of the $\overline{A\ STB}$ signal latches the data word in the I/O device.
- **b.** ____ The data into the PIO are latched into the input register when the $\overline{B\ STB}$ signal is high.
- **c.** ____ The $\overline{WR}$ signal is a composite of $RD \times \overline{CE} \times \overline{C/D} \times \overline{IORQ}$.
- **d.** ____ Mode 2 utilizes all four handshake lines and is available only for port A of the PIO.

6-16 The following questions relate to the timing chart of the bit control mode (Fig. 6-7). Answer yes or no.
- **a.** ____ Are the four handshake signals used in this mode?
- **b.** ____ Is the A $\overline{RDY}$ line forced into a high state?
- **c.** ____ Is an interrupt generated if the words on the port data lines do not conform to the preloaded logical conditions?

6-17 The statements below refer to the interrupt acknowledge and servicing in the PIO. (See Fig. 6-8.) Indicate true or false.
- **a.** ____ The $\overline{INTA}$ signal is composed of $\overline{IORQ}$ and $\overline{M1}$.
- **b.** ____ The $\overline{INT}$ line is sampled by the CPU during the negative-going trailing edge of the last clock pulse of the instruction.
- **c.** ____ During the trailing edge of the first WAIT period clock pulse, TW1, $\overline{IORQ}$ goes low.
- **d.** ____ During the $\overline{INTA}$ time, PIO determines priority by checking for I/O device with high IEO and low IEI conditions.
- **e.** ____ During the time when $\overline{M1}$ is active, no new interrupt requests are generated.

6-18 Extending the daisy chain requires the following additional gates (circle the correct answer):
- **a.** Four-input AND gates only.
- **b.** Two-input AND and four-input OR gates.
- **c.** Two-input and four-input OR gates.
- **d.** Two-input and four-input AND gates.
- **e.** Two-input OR and four-input AND gates.

7

PROGRAMMING THE PIO CHIP

CHAPTER OBJECTIVES

The objectives of this chapter are as follows:

1. To introduce the concept of customizing the PIO chip to specific applications.
2. To explain the format and functions of the mode set byte.
3. To explain the function and use of the I/O select register control word for the mode 3 operation.
4. To introduce the usage of the interrupt vector.
5. To discuss the function and format of the interrupt control byte.
6. To explain the function of the mask control byte.
7. To introduce the format of the interrupt enable flip-flop control byte.
8. To demonstrate how the PIO can be initialized for an industrial control application by means of a typical example.
9. To set up and program the PIO chip for a typical byte transfer application by means of a typical example.

TEXTBOOK REFERENCES

1. For general discussion of programmable parallel interfaces	Sec. 12-3
2. For operational modes control words	Secs. 12-3.3.2 and 12-3.3.3
3. For bit set/reset control word	Sec. 12-3.3.4
4. For discussion on vectored interrupt system	Sec. 9-5

7-1 INTRODUCTION

In Chapter 6 we discussed the PIO chip and how it operates, including the various modes of operation. The PIO is a programmable interface, which means that under program control, the user can select and use the various features of the chip to customize the interface to the exact requirements of a particular application. This is of course true within certain specified limits of the design.

The process of customizing the PIO chip requires that the chip first be set up by means of an initialization program which can include up to 6 bytes. Once the PIO is set up by the initialization program, it remains in that condition until it is altered by another similar initialization program. After power is turned off, the chip must be set up again with an initialization program before it can be used.

In this chapter we describe the initialization process and the bytes or words involved in such a program. Also, examples are included to show how the chip can be programmed for some typical applications.

7-2 THE PROGRAMMING WORDS

In this section we describe and discuss the formats of the various bytes (or words) involved in the initialization program. The programming bytes are presented in the sequence in which they would be included in the initialization program.

7-2.1 Word for Mode Set

As we discussed in Chapter 6, the PIO can be operated in any one of the four possible modes. The initialization program must identify which of the four modes is selected for a specific application. The first word in the set-up program does this. The format of this byte is shown in Fig. 7-1. In Fig. 7-1 the bits D0–D3 are always 1. The bits in positions D4 and D5 are not relevant. They could be anything. An X in each position means "Don't care." The code for the selected mode is inserted in bit positions D7 and D6. The codes follow,

M1	*M0*	*Mode Selected*
0	0	Output byte
0	1	Input byte
1	0	Bidirectional byte
1	1	Bit control

When mode 3 (i.e., the bit control mode) is used, the next byte must be used to set up the I/O select register. (See Fig. 6-2) The format of this byte is shown in Fig. 7-2. The function of this word is to assign each line to the I/O data/control bus individually as either an input or an output line. A 1 in any bit position sets the corresponding I/O line as an input line. A 0 sets the line as an output line.

7-2.2 The Interrupt Vector

Since the Z80 system uses the vectored interrupt scheme for identifying the interrupting device, in conjunction with the daisy chain for prioritization, it is necessary that a vector or a unique identifying code (i.e., an address) be assigned to each I/O chip. A 7-bit vector can be assigned as shown in Fig. 7-3. Notice that the D0 bit in this byte contains a 0. The selection of the code or the bit combination is left entirely to the discretion of the user, but it must be loaded only in bit positions D1–D7.

D7	D6	D5	D4	D3	D2	D1	D0
M1	M0	X	X	1	1	1	1

Figure 7-1 Format of the Mode Set Byte

D7	D6	D5	D4	D3	D2	D1	D0
I/O_7	I/O_6	I/O_5	I/O_4	I/O_3	I/O_2	I/O_1	I/O_0

Figure 7-2 Format of the I/O Select Register Control Word for the Mode 3 Operation

7-2.3 Set Interrupt Control Byte

The next byte in the program sets up the control word for the interrupt control logic block shown in Fig. 6-1. The format of this word is shown in Fig. 7-4. In this word, bits D3–D0 are always 0111. A 1 in the D7 position sets the interrupt enable flip-flop of the selected port. This means that the particular port could generate an interrupt. A 0 in this position resets this flip-flop and so interrupts cannot be generated by this port.

Bits D6, D5, and D4 are used only if the PIO is initialized for the mode 3 operation. Bit D6 enables the programmer to define the specific conditions that must exist on the I/O lines for that port to generate an interrupt. A 1 in the D6 position indicates an AND function, which means that all I/O lines must go to the specified logical states for an interrupt to be generated. A 0 in D6 specifies an OR condition, which means that an interrupt will be generated if any one or more of the I/O lines go to the specified logical state.

Bit position D5 specifies the high or low state of the I/O which will be monitored for generating interrupts. A 1 in D5 will monitor the I/O lines of that port for the high logical state and a 0 will monitor them for the low state.

At the option of the programmer, it is possible to monitor some I/O lines and mask out the rest. Bit position D4 allows us to do this. A 1 in this position indicates that the next control byte, input into the PIO, is the mask word whose format is shown in Fig. 7-5. A 0 in any bit position of the mask word will monitor the corresponding I/O line while a 1 will mask out the corresponding line.

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2	V1	0

Figure 7-3 Format of the Interrupt Vector

D7	D6	D5	D4	D3	D2	D1	D0
Interrupt Enable	AND/OR	Low/High	Mask Follows	0	1	1	1
	Only for Mode 3			Identifies Interrupt Control Word			

Figure 7-4 Format of the Interrupt Control Byte

D7	D6	D5	D4	D3	D2	D1	D0
Interrupt Enable	X	X	X	0	0	1	1

Figure 7-5 Format of the Mask Control Byte

D7	D6	D5	D4	D3	D2	D1	D0
MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0

Figure 7-6 Format of the Interrupt Enable Flip-Flop Control byte

7-2.4 Interrupt Enable Flip-Flop Set

The interrupt enable flip-flop can be set or reset by means of the byte whose format is shown in Fig. 7-6. In this case a 1 sets the flip-flop and a 0 resets it. Bit positions D3–D0 are always 0011, and this code identifies the byte as the interrupt enable control word. Bits D6, D5, and D4 can be any combination of 1s and 0s. We will now show some examples of how the PIO can be initialized for specific applications.

7-3 PROGRAMMING THE PIO FOR A CONTROL APPLICATION

Example 7-1

In an industrial control application, the RPM of an electric motor must be maintained within the upper RPM limit, R_{max}, and the lower limit, R_{min}. In other words,

$$R_{max} > RPM > R_{min}$$

An RPM value lower than R_{min} is indicated by a digital 1 on signal line S_1 and RPM value higher than R_{max} is indicated by a 1 on the signal line S_2. If the motor stalls, because of a sudden surge in the load, the stalled condition is indicated by a digital 1 on the S_3 signal line. Finally, an equipment failure in the motor controller is indicated by a 1 on the S_4 line. From the computer side, the following signals are to be sent to the motor controller. A high state is used for generating interrupts.

1. Special test signal, ST, to initiate the test cycle.
2. Power turn on signal, P_{ON}.
3. Increase motor RPM signal, R_I.
4. Decrease motor RPM signal, R_D.

The unique address assigned to the PIO chip is 11101010. Draw a system configuration for this application and write a PIO initialization program explaining how each program byte is constructed. Write the program in both machine language and hexadecimal codes.

Solution

From the problem statement it is clear that we will have to use mode 3 (bit control mode) in this particular application. The system configuration for this problem is shown in Fig. 7-7.

For this particular application the status and control word, to be transferred between the CPU and the PIO on the D0–D7 bus, is formatted as shown in Fig. 7-8. Note that the status and control bits are arbitrarily assigned the various bit positions in this word. The following procedure is then used for generating the initialization program.

1. We use port A of the PIO for the mode 3 operation. Mode 3 is selected by loading the mode select word from the CPU into the PIO, as discussed in Sec. 7-2.1. The format of this word is shown in Fig. 7-9.
2. Because mode 3 is selected, the next word in the program to be sent by the CPU is the I/O select word, which was discussed in Sec. 7-2.1, Fig. 7-2. This word assigns the various PIO-to-motor controller lines (A0–A7) as either input or output lines. In this case, the direction of signal flow is defined in Fig. 7-7, so the word shown in Fig. 7-10 is loaded into the I/O select register of Fig. 6-2.
3. The interrupt vector of the PIO is now loaded. The format of this byte, which was previously discussed in Sec. 7-2.2, is shown in Fig. 7-11 for this problem.
4. The interrupt control byte, whose format was discussed in Sec. 7-2.3 (Fig. 7-4), is now constructed as shown in Fig. 7-12, where
   a. Bit D7 is a 1 because the interrupt enable flip-flop is to be set.
   b. D6 is a 0 because any one of the incoming signals, S_1–S_4, must be able to generate an interrupt. An OR function is used.
   c. D5 is a 1 because the incoming signals S_1–S_4 are active high according to the problem statement.
   d. D4 is a 1 because the next byte in the initialization program is a mask byte.
5. The next word to be loaded is the mask word, which was discussed in Sec. 7-2.3, Fig. 7-5. Since lines A3, A2, A1, and A0 are to be monitored, the format of the word is shown in Fig. 7-13. Note that bits D0–D3 are 0s, so lines A0–A3 will be monitored and the rest masked out.

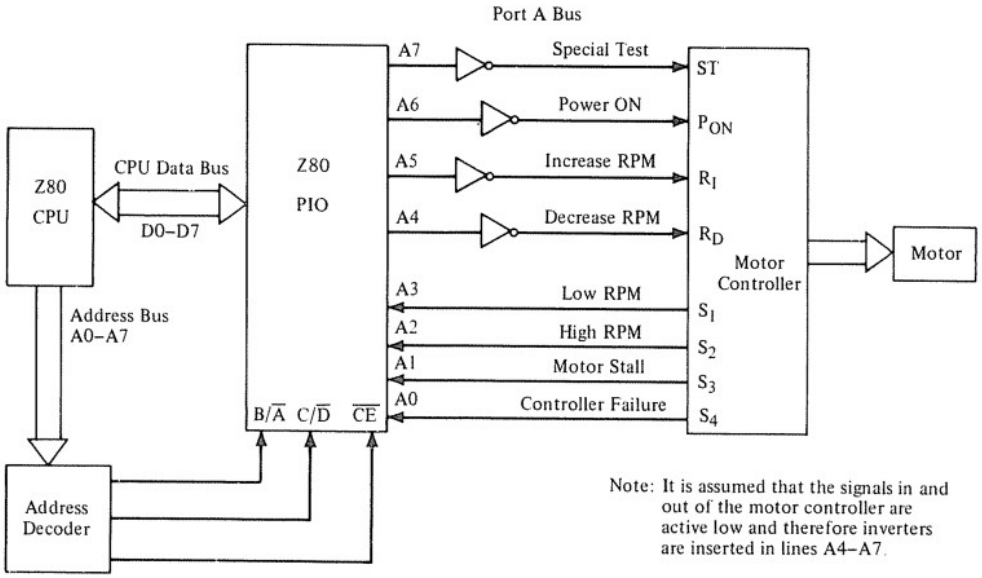


Figure 7-7 System Configuration for Example 7-1 (Mode 3—Bit Control Mode)

D7	D6	D5	D4	D3	D2	D1	D0
Special Test (ST)	Power On (P_{ON})	Increase RPM (R_I)	Decrease RPM (R_D)	Low RPM (S_1)	High RPM (S_2)	Motor Stall (S_3)	Controller Failure (S_4)

Figure 7-8 Format of the Status and Control Word for Example 7-1

D7	D6	D5	D4	D3	D2	D1	D0
1	1	X	X	1	1	1	1

Figure 7-9 Mode Set Byte for Example 7-1

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	1	1	1

Figure 7-10 I/O Select Word for Example 7-1

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	0	1	0	1	0

Figure 7-11 The Interrupt Vector for Example 7-1

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	1	0	1	1	1

Figure 7-12 Set Interrupt Control Byte for Example 7-1

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	0	0	0	0

Figure 7-13 The Mask Word for Example 7-1

6. This completes the words required to configure the PIO chip for our specific problem. Now if one of the sensors from the motor controller puts a 1 signal on any of the four lines A0–A3, an interrupt will be generated and a status word will be sent by the PIO to the CPU. The CPU will then send back an appropriate control word to the motor controller via the PIO chip. The program for initializing the PIO for this problem is shown in Table 7-1. Both machine codes and hexadecimal codes are included. ■

7-4 PROGRAMMING THE PIO FOR A BYTE TRANSFER APPLICATION

In this section we will show, by means of an example, how the PIO can be programmed for an application which requires the transfer of data bytes between a bidirectional I/O peripheral and the PIO. This means that the PIO must be operated in mode 2, bidirectional mode. Recall from Chapter 6 that only port A can be used in mode 2 because bidirectional data transfers utilize all four handshake

Table 7-1 INITIALIZATION PROGRAM FOR PROBLEM 7-1

Byte Function	*Hexa decimal Code*	*Machine Language Code*							
		D7	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*
Mode set byte	C F	1	1	0	0	1	1	1	1
I/O select word	O F	0	0	0	0	1	1	1	1
Interrupt vector	E A	1	1	1	0	1	0	1	0
Set interrupt control	B 7	1	0	1	1	0	1	1	1
Mask word	F O	1	1	1	1	0	0	0	0

lines. The PIO chip is designed such that in mode 2 operation, the port A handshake signals (i.e., A RDY and $\overline{\text{A STB}}$) are used for output control and the port B handshake signals, i.e., B RDY and $\overline{\text{B STB}}$, are used for input control. Example 7-2 shows how the PIO can be programmed for this mode and how the system is configured for such an application.

Example 7-2

A bidirectional, 8-bit parallel I/O peripheral whose interrupt vector is 9C in hexadecimal code is connected to port A of the PIO chip. Draw a sketch of the system configuration. Write the program for initializing the PIO chip and explain how it is constructed. Show both machine language and hexadecimal codes for this problem.

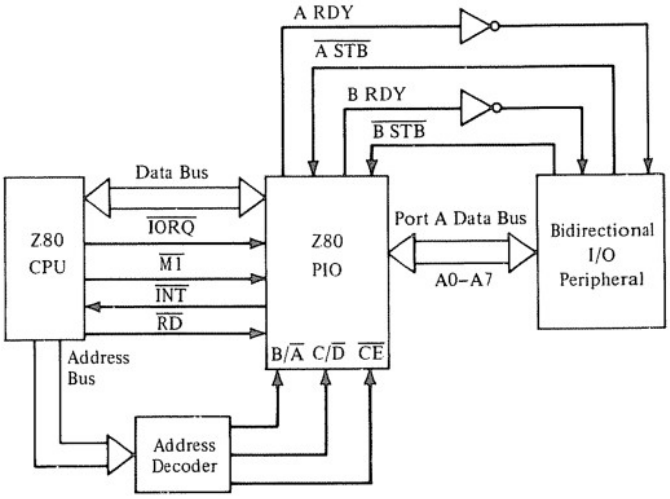


Figure 7-14 System Configuration for Example 7-2

Solution

The system diagram for this problem is shown in Fig. 7-14. Notice that the port A handshake signals are used for output control signals and port B handshake signals for the input control signals.

In Fig. 7-14 two inverters are inserted in the A RDY and the B RDY lines. This is because it is assumed that all handshake signals in and out of the I/O peripheral are active low.

To initialize the PIO chip, the mode set word (Fig. 7-1) is loaded first. This is shown in Fig. 7-15. The Interrupt Vect is the next word to be loaded into the PIO. The problem statement gives this word to be 9C and so this word is shown in Fig. 7-16. Finally, the interrupt enable flip-flop control byte, whose format is shown in Fig. 7-6, must be loaded. This is shown in Fig. 7-17. The program for initializing the PIO for this problem is shown in convenient tabular form in Table 7-2. Both machine language codes and hexadecimal codes are included. ■

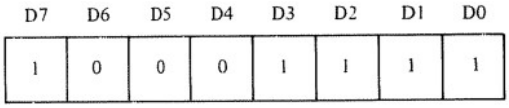


Figure 7-15 Mode Set Word for Example 7-2

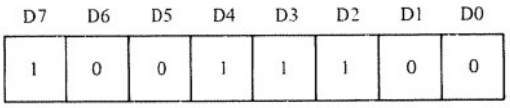


Figure 7-16 Interrupt Vector Word for Example 7-2

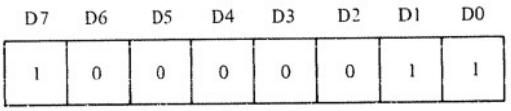


Figure 7-17 Interrupt Enable Flip-Flop Control Byte for Example 7-2

Table 7-2 INITIALIZATION PROGRAM FOR EXAMPLE 7-2

BYTE FUNCTION	*Hexa-decimal CODE*	*MACHINE LANGUAGE CODE*							
		D7	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*
Mode set byte	8 F	1	0	0	0	1	1	1	1
Interrupt Vector	9 C	1	0	0	1	1	1	0	0
Interrupt enable flip-flop control byte	8 3	1	0	0	0	0	0	1	1

7-5 REVIEW QUESTIONS

7-1 The following statements relate to the programmable features of the PIO chip. Indicate true or false.

a. ___ The PIO can be customized to the specific requirements of a particular application.

b. ___ The initialization program always contains 6 bytes.

c. ___ After being initialized, the PIO remains in that configuration until reinitialized by a different program.

7-2 From the following statements pick out the one(s) applicable to the initialization program byte 1 1 X X 1 1 1 1.

a. This is the I/O select register control byte.

b. This is the mode set byte.

c. This byte defines the bidirectional byte transfer mode.

d. This byte must be followed by the I/O select register control word.

7-3 In a PIO initialization program using mode 3 operation, the second byte in the program is 1 0 1 0 1 0 1 0. Indicate true or false for the following statements which apply to this byte.

a. ___ It is the interrupt vector.

b. ___ Lines D0, D2, and D4 are input lines.

c. ___ Lines D1 and D7 are input lines.

d. ___ Line D6 is an output line and D3 is an input line.

7-4 From the following interrupt vectors (given in hexadecimal codes) pick out those than can be legitimately used in initializing the PIO chip.

a. 8 8

b. D 0

c. 9 B

d. E E

e. F F

f. A 6

7-5 The interrupt control byte of a particular PIO initialization program is 1 0 0 0 0 1 1 1 for mode 3 operation. Indicate true or false for the following statements which relate to this byte.

a. ___ A low state of the I/O lines will be monitored for generating interrupts.

b. ___ The interrupt enable flip-flop is reset.

c. ___ The next word input into the PIO is mask word.

d. ___ An interrupt will be generated if any one or more of the I/O lines go to a low state.

7-6 Construct the interrupt control byte for the initialization program that will satisfy the following conditions:

a. Monitor the high state of the I/O lines.

b. Set the interrupt enable flip-flop.

c. Define only specific I/O lines to be monitored.

d. Monitor the high state of all the specified I/O lines.

Bit positions	D7	D6	D5	D4	D3	D2	D1	D0
Byte code								

7-7 Refer to question 7-6. Construct the byte which will follow the byte resulting from question 7-6 and which will monitor I/O lines D0, D1, D3, and D6.

Bit positions	D7	D6	D5	D4	D3	D2	D1	D0
Byte code								

7-8 Refer to Example 7-1 (Fig. 7-7). Inverters are inserted in PIO lines A4–A7 because

a. Signals A4–A7 are output by the PIO chip.

b. Signals A3–A0 do not have inverters external to the PIO chip.

c. Signals A4–A7 are unidirectional.

d. Signals into the motor controller are active low.

e. Signals into the motor controller are active high.

7-9 Refer to the status and control word format for Example 7-1 and indicate true or false.

a. ___ A single bit for indicating motor RPM (such as 1 for high and 0 for low RPM) could have been used instead of two signals.

b. ___ Signal S_4 from the motor controller is not

really necessary because the controller would fail only if there were a power failure.

7-10 Refer to Fig. 7-8, which is the status and control byte for Example 7-1. From the following statements pick out those that apply to this byte.

a. The status signals must always be placed on the lower half of the bus for port A, i.e., on lines A0–A3.
b. The control signals must always be output on the upper half of the port A bus, i.e., on lines A4–A7.
c. The assignment of the status and control lines of port A is arbitrary.
d. No more than four lines may be used for incoming status signals.
e. No more than four lines may be used for the outgoing control signals.

7-11 Bit D6 of the interrupt control byte for Example 7-1 is a 0 because

a. The low state of the I/O lines is monitored.
b. Any incoming status signal is required to generate an interrupt.
c. The high state of the I/O lines is monitored.
d. A mask byte follows this byte.

7-12 Indicate true or false for the mask byte for Example 7-1.

a. ___ This word is included in the initialization program because bit D4 of the previous interrupt control byte is a 1.
b. ___ If bit D4 of the previous interrupt control byte had been a 0, then the mask word of Fig. 7-13 will contain all 0s.
c. ___ Bits D4–D7 are all 1s because lines A4–A7 of the PIO port A are output lines.

7-13 The following statements relate to the system configuration (Fig. 7-14) for Example 7-2. Indicate true or false.

a. ___ Lines A7–A4 are used for control signals.
b. ___ Lines A3–A0 are used for status signals.
c. ___ Signals A RDY and $\overline{\text{A STB}}$ are used for input control.
d. ___ Signals B RDY and $\overline{\text{B STB}}$ are used for output control.

7-14 Refer to Fig. 7-14. Inverters are inserted in the A RDY and the B RDY lines because

a. They are input control lines.
b. They are output control lines.
c. They indicate the status of the Z80 CPU.
d. All handshake signals in and out of the peripheral are assumed to be active low.

7-15 Indicate true or false for Example 7-2.

a. ___ The PIO could have used mode 3 for the operation.
b. ___ Either mode 0 or mode 2 could have been selected.
c. ___ Only mode 2 could have used in this application.
d. ___ Either mode 1 or mode 2 could have been selected.

7-16 The initialization program for Example 7-2 contains only 3 bytes because

a. Mode 2 requires only the mode set byte, the interrupt vector, and the interrupt enable flip-flop control byte.
b. Only 1 byte of data is transferred at any one time.
c. Only 4 handshake signals are required for Example 7-2.
d. Bits D4 and D5 of the mode set byte are both 0s.

8

Z80 COUNTER/TIMER CIRCUIT CHIP

CHAPTER OBJECTIVES

1. To introduce students to programmable timing and counting functions.
2. To present the four independent, programmable channels for timing and counting in the Z80 μC system.
3. To describe the operation of the counter mode and the decrement logic triggering by an external signal.
4. To discuss the operation of the timer mode as well as the prescaler and the down counter.
5. To show how the counter/timer circuit chip fits into and operates in the prioritized interrupt daisy chain of the Z80 μC system.
6. To present the interrupt control logic and the protocol involved in the counter/timer operation.

TEXTBOOK REFERENCES

For reviewing material in the main textbook, relevant to the topics in this chapter, the following chapters and/or sections are sugggested:

1.	For general discussion of counters and timers	Chapter 14
2.	For time constant register	Sec. 14-2.2
3.	For the prescalar	Secs. 14-2.3 and 14-6.2
4.	For the interrupt feature	Sec. 14-2.4
5.	For timer mode operation	Secs. 14-2.5 and 14-7.2
6.	For counter mode operation	Sec. 14-2.6
7.	For counter-timer chip architecture	Sec. 14-3
8.	For chip initialization words	Sec. 14-3.2
9.	For channel control logic	Sec. 14-3.3
10.	For prescale factor selection	Sec. 14-6
11.	For advanced features and capabilities	Sec. 14-9

8-1 INTRODUCTION

The counter/timer circuit chip (CTC) provides four independent, programmable channels in the Z80-based μC system for counting and timing functions. Under program control, the CTC can be reconfigured to operate in several different conditions and modes so that it can be interfaced with a wide variety of I/O devices. When the CTC is used in the counter mode, an external signal is used to trigger the decrement logic. It is possible to have the CTC generate an interrupt when a certain number of external events have been counted out. In the timer mode, the CTC generates time intervals by dividing the system clock by means of a prescaler that decrements a preset down counter. Time intervals as short as 6.4 μs (for the Z80 system) or 4.0 μs (for the Z80A system) can be generated. The time interval is an integer multiple of the clock period.

The four CTC channels have priorities established so that they can readily fit into the standard Z80 interrupt daisy chain, previously described in Chapter 6. Channel 0 has the highest priority, and channel 3 has the lowest.

8-2 PRINCIPAL FEATURES OF THE CTC

1. The CTC is a third-generation chip which uses the N-channel, depletion mode, silicon-gate technology.
2. A single, 5 volt-±5% power supply is required.
3. A single-phase 5-volt clock is used.
4. The chip operates in a temperature environment of 0° to 70°C.
5. All inputs and outputs are fully TTL compatible.
6. The CTC can be interfaced directly with the Z80 CPU.
7. The CTC can be directly interfaced with the Z80 SIO chip (to be described in Chapter 12) for generating the desired baud rate.
8. When used in either the counter or the timer mode, a down counter is provided to indicate to the CPU the number of counts still to go before zero.
9. A time constant register makes it possible to reload the down counter when it reaches 0. This can be done in either the counter or the timer mode.
10. Either a positive or a negative trigger can be selected to initiate timer operation in the timer mode or to monitor for the events count in the counter mode.

8-3 THE CTC ARCHITECTURE

8-3.1 Generalized CTC Block Diagram

1. Figure 8-1 shows a generalized block diagram of the CTC.
2. On the computer side, the CTC chip provides I/O interface logic for transferring data on an 8-bit bus, D0–D7. A 7-bit control bus is also present.
3. Four independent channels are numbered 0 through 3. Channel 0 has the highest priority and channel 3 the lowest. These four channels can be connected into the 4 consecutive slots of the standard Z80 daisy chain which was previously discussed in Chapter 6.
4. A unique interrupt vector can be generated for each channel. This vector is used to determine the starting address for that particular channel's service subroutine.
5. Also on the computer side, the CTC communicates via three interrupt vector control lines. Each of these lines is an unidirectional line.
6. The internal control logic block primarily deals with the overall operation of the CTC chip and includes functions such as the read/write operation, chip enable and reset functions.
7. The interrupt control logic works in a manner very similar to that described in Chapter 6 for this block. Fundamentally, it performs the following functions:
   a. If the IEI is high, the CTC has the priority.
   b. During execution of the interrupt service subroutine, the IEO signal is held low. This inhibits interrupt requests from I/O devices with lower priorities.

8-3.2 The Channel Logic Block Diagram

1. Figure 8-2 is a block diagram of the block labeled *channel logic* in Fig. 8-1. It consists of two counters, two registers, and the associated control logic.

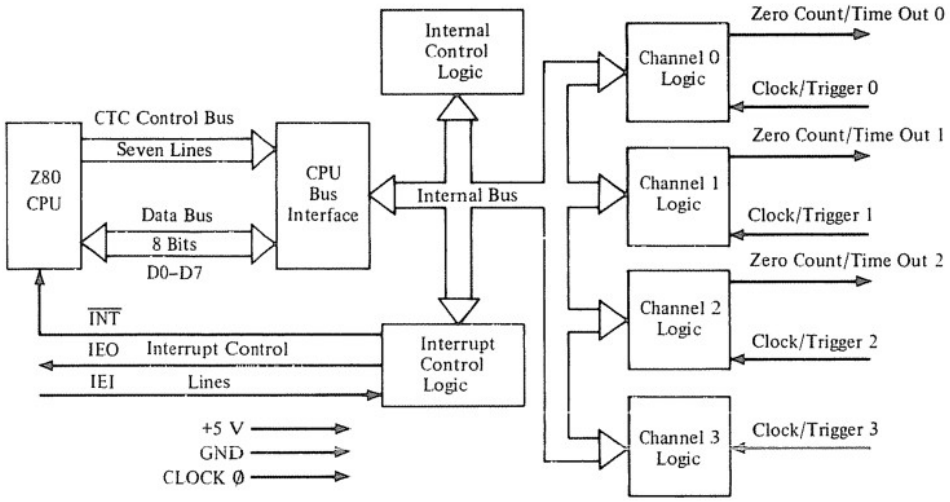


Figure 8-1 Generalized CTC Block Diagram

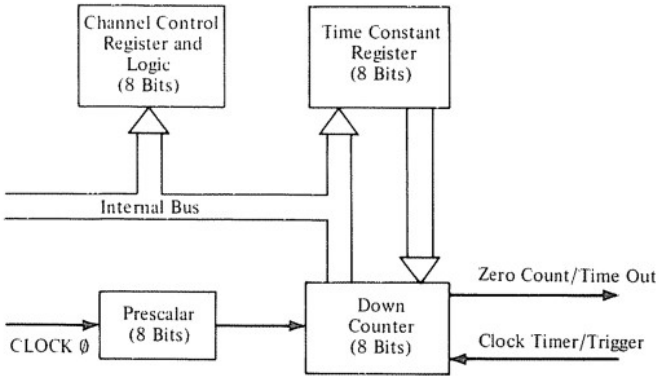
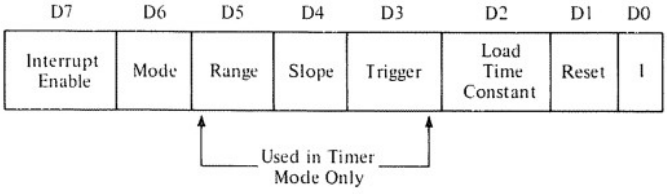


Figure 8-2 Channel Logic Block Diagram

2. The channel control register and logic receives an 8-bit control word from the CPU when that particular channel is programmed or selected.
3. The selection of one of the four channels is made by a 2-bit code supplied by the CPU and input to the CTC chip on two pins (CS0 and CS1) which are connected to the A0 and A1 lines of the address bus from the CPU. The channel codes are shown below in table form.
4. The format of the control word sent by the CPU, and loaded into the channel control register, follows:



The bits of the channel control word have the following meanings:

D0 1 indicates control word
0 indicates vector
D1 1 software reset
0 continued operation
D2 1 time constant follows
0 no time constant follows
D3 1 clock/trigger pulse starts the timer
0 automatic trigger when time constant is loaded
D4 1 selects positive-going edge of clock/trigger pulse
0 selects negative-going edge of clock/trigger pulse
D5 1 indicates prescaled value of 256
0 indicates prescaled value of 16
D6 1 indicates counter mode
0 indicates timer mode
D7 1 enables interrupt
0 disables interrupt

5. The prescaler, which is used only in the timer mode, is an 8-bit register which is programmed by the CPU (through the channel control register).
6. The prescaler divides the input clock frequency (i.e., the system clock ø) by either 16 or 256. The divided output is used as the clock input to the down counter.
7. The time constant register is an 8-bit register which is used both in the counter and the timer operation modes. As noted in paragraph 4, a 1 in D2 in the channel control word indicates that the next word sent to the selected channel by the CPU is a time constant word, and it is preloaded in the time constant register. This word is an integer time constant whose value ranges from 0 to 255, since it is 8 bits long. The 0 is interpreted as 256.
8. When the CTC chip is first initialized, this time constant value is loaded into the down counter. After the down counter is decremented to 0, the value in the time constant register is automatically reloaded into the down counter.
9. If a new integer is loaded into the time constant register while that particular channel's down counter is in the process of counting down, the count down is not interrupted. The new time constant is loaded into the down counter only after it reaches 0.
10. The down counter is an 8-bit register/counter which is always decremented when the CTC is used in either the counter or the timer mode. It receives its clock ø from the divided value output by the 8-bit prescaler. A high signal is output on the zero count/time out line every time the count reaches zero in this counter.
11. Bit D3 of the channel control word determines how the down counter is triggered to start counting, i.e., decrementing. If D3 is a 0, the counter is triggered automatically and starts decrementing. On reaching a count of 0 the down counter is reloaded with the starting value (i.e., the time constant) from the time constant register. Once started, this process continues until it is stopped by a reset.

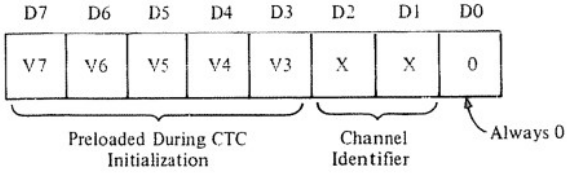
Table 8-1 CHANNEL SELECT CODES

	CS1	*CS0*
Channel 0	0	0
Channel 1	0	1
Channel 2	1	0
Channel 3	1	1

12. If bit D3 of the channel control word is a 1, the decrementing process of the down counter is started only when the active edge of the pulse appears on the clock timer/trigger line as shown in Fig. 8-2. Again, once it is started, it will continue running until it is stopped by a reset.
13. In the generalized block diagram of the CTC chip, note that the zero count/time out signal is available from channels 0, 1, and 2 but not from channel 3. This is due to the package pin limitations. Thus, channel 3 can only be used in those applications which do not require this signal output from the CTC.
14. An unusual feature of the CTC is that the CPU can obtain, or read, the contents of the down counter any time by performing an I/O operation of the selected CTC channel. This feature enables the CPU to determine how many count downs are still to go before the down counter reaches 0.

8-3.3 Interrupt Control Logic and Protocol

1. The primary function of the interrupt control logic in the CTC chip (see Fig. 8-1) is to ensure that the interrupt protocol of the Z80 system is followed by the CTC.
2. Again, because of the daisy chain technique, the priority of any device depends on its relative position in this chain. The device closest to the CPU has the highest priority.
3. Within the CTC chip itself, channel 0 has the highest priority and channel 3 the lowest. Note that in accordance with the Z80 interrupt protocol, higher priority channels can override and interrupt lower priority channels but not vice versa.
4. Any channel can be programmed to issue an interrupt request when its down counter reaches a 0 count. However, this requires that the CPU be programmed for the interrupt mode 2.
5. On receiving the interrupt acknowledge from the CPU, the CTC logic determines which of the highest priority channels is interrupting and will be serviced. If the IEI signal is active, thereby indicating that the CTC has priority within the daisy chain, the CTC will place its 8-bit interrupt vector on the data bus.



6. The format of the CTC interrupt vector is shown here. Note that this word is identified by bit D0 being always 0, as opposed to the previously discussed channel control word whose D0 bit is always 1.
7. Bits D3–D7 of the interrupt vector are preloaded during the initialization of the CTC chip. Bits D2 and D1 represent a 2-bit code identifying the interrupting channel. These two bits are generated and inserted into the vector by the Interrupt Control Logic of the CTC. The table below shows the codes for bits D2 and D1.
8. The interrupt vector is utilized by the CPU as the least significant 8 bits of a 16-bit pointer which provides the starting address of interrupt service subroutine for the particular channel. The upper 8 bits of this pointer are provided by the contents of the I register in the CPU.
9. Students are referred to Fig. 4-1, which graphically represents how the pointer is concatenated and the service subroutine is called out.
10. A system convention requires that all the 16-bit addresses, which are prestored in the interrupt service subroutine table in the memory, should have their low-order byte placed in an even location in the memory. The high-order byte is then placed in the next highest location, which will always be an odd-numbered address. Thus, the LSB of any interrupt vector will always be even as indicated by a 0 in the D0 position.
11. The last instruction in all interrupt service subroutines is RETI, which initializes the daisy chain enable line IEO. This ensures proper control for handling priorities in the interrupt system. The CTC monitors the CPU data bus, decodes this instruction, and executes it.

8-4 PACKAGE PIN ASSIGNMENTS AND FUNCTIONS

The CTC is contained in a standard 28-pin DIP package. The various pins and their respective functions, along with

Table 8-2 INTERRUPTING CHANNEL IDENTIFICATION CODES

	D2	*D1*
Channel 0	0	0
Channel 1	0	1
Channel 2	1	0
Channel 3	1	1

their designations, are presented in convenient tabular form. Functionally, the pins can be divided into five groups, as shown in Fig. 8-3. Note that some of the lines are shown as unidirectional and others as bidirectional. In the latter group, some of the lines may indicate signal flow in one direction and others in the other direction. The function tables clarify these with the following arrow symbols:

CTC → Indicates signals flowing *from the CTC*
CTC ← Indicates signals flowing *into the CTC*
CTC ↔ Indicates *bidirectional* flow on the same line(s)

8-5 THE OPERATING MODES

8-5.1 Setting Up the CTC Chip

1. When power is turned on, the CTC chip goes into an undefined and unpredictable state. Recall that the $\overline{\text{RESET}}$ signal from the CPU puts the CTC into a certain state, which accomplishes the following:
   a. The interrupt-generating capabilities of the CTC are disabled.
   b. The zero count/time out and the $\overline{\text{INT}}$ output lines are put into the inactive state.
   c. The D0–D7 bus lines go into the high-impedance states.
   d. The IEO line is set to the same state as the IEI line.
2. However, the $\overline{\text{RESET}}$ signal does not program the CTC chip. Before the desired channel can perform any timing or counting function, the CTC must be programmed with the following three words, in the given order:
   a. *The interrupt vector.* This 8-bit word is loaded into the interrupt control logic of the CTC. The function, format and operation of this word is explained in Sec. 8-3.3.
   b. *The channel control word.* This 8-bit word is loaded into the channel control register. The format of this word is explained in Sec. 8-3.2.
   c. *The time constant word.* This 8-bit word is loaded into the time constant register. It contains an integer whose value ranges from 1 to 256. When all 8 bits are zero, it is automatically interpreted as 256.
3. We will now explain the operation of each mode in more detail.

8-5.2 Operating in the Counter Mode

1. When set up in this mode, the function of the CTC chip is to count the number of signals or pulses input to the preselected CTC channel on the CLK/TRG line.

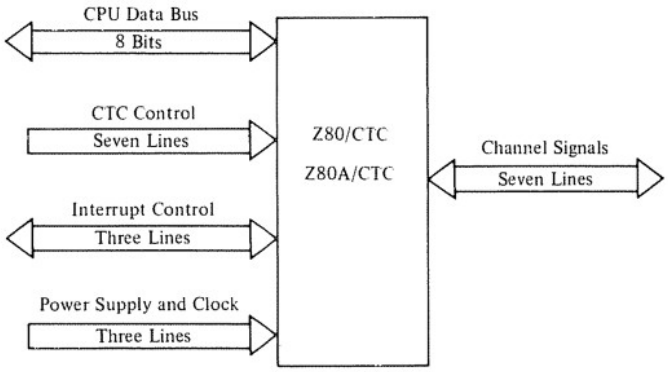


Figure 8-3 Block Diagram of CTC Package Pin Groups

Table 8-3 DATA TRANSFER BUS GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
D0–D7	CTC ↔	8-bit, tristate, CPU data bus, is active high. It is used for data and commands transfer between the CTC and the CPU.

Table 8-4 POWER SUPPLY AND CLOCK GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
ø	CTC ←	Single-phase TTL level clock input.
+5 V	CTC ←	This is the single power supply of 5 volt-±5% that is required.
GND	CTC ←	This is the power supply ground.

Table 8-5 CTC CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
CS0–CS1	CTC ⟵	2-bit binary code for selecting one of the four channels on the CTC chip during I/O operations. Generally connected to lines A0–A1 of the CPU address bus.
$\overline{M1}$	CTC ⟵	Machine cycle 1 signal, active low, is a SYNC pulse from the CPU. When $\overline{M1}$ and $\overline{RD}$ are simultaneously active, the CPU is fetching an instruction from the memory. When $\overline{M1}$ and $\overline{IORQ}$ are simultaneously active, the CPU is acknowledging an interrupt. The CTC then places its interrupt vector on the CPU data bus, provided it has the highest priority and an interrupt has been requested by a CTC channel.
$\overline{IORQ}$	CTC ⟵	I/O request from CPU signal, active low, is used for transferring data between CPU and CTC as well as for sending channel control words to the CTC. The CTC write cycle is initated by the active status of $\overline{CE}$ and $\overline{IORQ}$ signals and inactive of $\overline{RD}$. (*Note:* CTC does not receive a specific external write signal but generates its own by inverting the $\overline{RD}$ signal.) A CTC read cycle is initiated by simultaneous activation of $\overline{CE}$, $\overline{IORQ}$, and $\overline{RD}$ signals and the contents of the down counter are transferred to the CPU on the data bus. When $\overline{IORQ}$ and $\overline{M1}$ are simultaneously active, the CPU is acknowledging an interrupt and so the highest-priority channel sends its vector to the CPU via the data bus.
$\overline{RD}$	CTC ⟵	Read cycle status, active low, signal when used simultaneously in conjunction with $\overline{CE}$ and $\overline{IORQ}$, it signals the CTC to transfer contents of the down counter to the CPU. During the write cycle, it is inactive while the other two are active. The CTC then accepts channel control word from the CPU or data transfers between CTC and the CPU.
$\overline{CE}$	CTC ⟵	Chip enable, an active low signal that informs the CTC to transmit the contents of the down counter to the CPU during the read cycle. During the write cycle, it informs the CTC to accept either control words, time constant words, or interrupt vectors from the CPU.
$\overline{RESET}$	CTC ⟵	Reset signal, active low. Stops the down counting process of all channels in the CTC and resets the interrupt enable bits in all control registers. This kills all interrupts generated by the CTC. Also, the zero count/time out and the $\overline{INT}$ output lines are put into the inactive states. The D0–D7 bus lines go into the high-impedance states, and the IEO line is set to the same state as the IEI line.

Table 8-6 THE INTERRUPT CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
$\overline{INT}$	CTC ⟶	Interrupt request. Is active when the down counter of any channel in the CTC that is programmed to enable interrupt reaches the 0 count.
IEI	CTC ⟵	Interrupt enable in, active high. Used in the system-wise priority interrupt daisy chain. A high or active signal indicates that no other I/O device with higher priority is being currently serviced.
IEO	CTC ⟶	Interrupt enable out, active high. Also used in the priority interrupt daisy chain. It is active only when IEI is active and the CPU is not servicing an interrupt from any other CTC channel. When low, it blocks interrupt signals from lower-priority I/O devices from interrupting while it is being serviced.

Table 8-7 CHANNEL SIGNALS GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
CLK/TRG_0–CLK/TRG_3	CTC ⟵	External clock timer trigger, active either high or low as selected by the user. Four such lines are available for each independent channel. When used in the counter mode, the signal transition (i.e., either from high to low or low to high, as predefined) decrements the down counter. When used in the timer mode, the signal transition initiates the timing function.
ZC/TO_0–ZC/TO_2	CTC ⟶	This is the zero count time out signal, from channels 0, 1, and 2 only, which is active high. When the down counter goes to 0, in either the counter or the timer mode, this signal goes high.

Table 8-8 PIN NUMBER ASSIGNMENTS

Pin Designations	*Pin Numbers*	*Pin Designations*	*Pin Numbers*
CPU Data Bus		*Channel Signals*	
D0	25	CLK/TRG 0	23
D1	26	ZC/TO 0	7
D2	27	CLK/TRG 1	22
D3	28	ZC/TO 1	8
D4	1	CLK/TRG 2	21
D5	2	ZC/TO 2	9
D6	3	CLK/TRG 3	20
D7	4		
CTC Control		*Interrupt Control*	
CS0	18		
CS1	19	$\overline{INT}$	12
$\overline{CE}$	16	IEI	13
$\overline{M1}$	14	IEO	11
$\overline{IORQ}$	10		
$\overline{RD}$	6		
$\overline{RESET}$	17		
		Power Supply and Control	
		5 volts	24
		GND	5
		ø	15

2. Each input signal then decrements the contents of the counter. Notice that the time constant word had previously been loaded into the down counter.
3. When the down counter reaches 0, a pulse is output on the ZC/T0 line of that particular channel. As previously explained, this output is available only for channels 0, 1, and 2 (because of pin limitations on the package).
4. When the down counter reaches 0, an interrupt request signal ($\overline{INT}$) is generated and sent to the CPU, provided the channel has been programmed to do so by bit 7 of the channel control word.
5. Notice that the incoming signal on the CKL/TRG line can be programmed (by means of bit D4 of the channel control word) to respond to either the positive-going or the negative-going edge of this pulse.
6. The CLK/TRG pulse inputs do not have to come into the CTC at regular timer intervals. They can, and do, come randomly. However, the down counter will not

decrement until the next clock ø following the CLK/TRG input.

7. When the down counter reaches 0, the time constant register automatically reloads this counter with the 8-bit time constant word.
8. If a new time constant word is written into the time constant register while the counter is in the decrementing process, the current count will be completed. The new time constant will be loaded only after the down counter has been decremented to zero.

8-5.3 Operating in the Timer Mode

1. When programmed in the timer mode, the function of the selected CTC channel is to provide a signal on the ZC/TO line when a certain preestablished time has elapsed.
2. The elapsed time can be programmed by the user and is the product of three factors. Two of these three factors are variable and directly controllable by the programmer. The elapsed time is given by the equation

$$T_e = t_c \times \text{PF} \times \text{TC}$$

where T_e is the elapsed time,
t_c is the system clock period,
PF is the prescaler factor, and
TC is the 8-bit integer time constant.

3. The operation of this mode is relatively simple. The system clock ø is passed through two counters which divide the clock frequency at each stage. First, the prescaler issues an output once every 16 or 256 input pulses of the clock ø. This is then used as input clock for the down counter, which is preloaded with a time constant.
4. If D3 of the channel control word is reset to 0, the CTC will start the timing operation automatically at the start of the CPU cycle following the I/O write cycle that loads the TC data word in the selected channel.
5. If D3 is set to 1, then the timing operation starts on the second succeeding positive-going edge of ø after it receives the trigger active edge on the CLK/TRG line of that channel.
6. If D2 of the channel control word is a 0, indicating that no time constant word follows, then the CTC starts the timing operation on the second succeeding positive-going edge of the clock ø after the timer trigger edge following the control word inserted in the CTC.
7. When D7 of the channel control word is set to 1, the interrupt-generating capabilities of the CTC are enabled and an interrupt request signal (INT) is sent to the CPU every time the down counter reaches a 0.
8. The Z80 system now has the following three clock speeds available:

Z-80	2.5 MHz
Z-80A	4.0 MHz
Z-80B	6.0 MHz

Using the TC range of 1 –256 and the prescaler factor of 16 or 256, the elapsed time range available for each of three systems is calculated from the equation in paragraph 2 above and given as follows:

For Z-80,	6.4 μs	to 26.214 ms
For Z-80A,	4.0 μs	to 16.834 ms
For Z-80B,	2.65 μs	to 10.88 ms

8-6 THE CTC TIMING

In this section we discuss the timing involved in the write cycle, the read cycle, and the counting and timing modes as well as the interrupt acknowledge and the return from the interrupt cycle. In the following timing charts, notice that the clock phases are shown as perfect rectangular waves. This is done only for convenience. Actually each clock pulse has its own rise and fall times. The rise and fall times of the other signals are shown as appropriate. The clock pulses are used in these charts simply for reference purposes and therefore they are shown as rectangular pulses.

8-6.1 The CTC Write Cycle

1. The write operation in the CTC refers to the operation involved in initialization of the chip. In other words, this relates to the acceptance of the channel control word, the time constant word, and the interrupt vector sent by the CPU.
2. Note that the CTC does not have a separate write signal input. A write signal is internally generated when the $\overline{\text{RD}}$ signal goes high.
3. The timing chart for the write operation is shown in Fig. 8-4. During or before clock period T1, the $\overline{\text{RD}}$ goes high. This prepares the CTC chip for the write cycle.
4. During clock period T2 the CPU initiates the write cycle by putting the chip enable, $\overline{\text{CE}}$, and the $\overline{\text{IORQ}}$

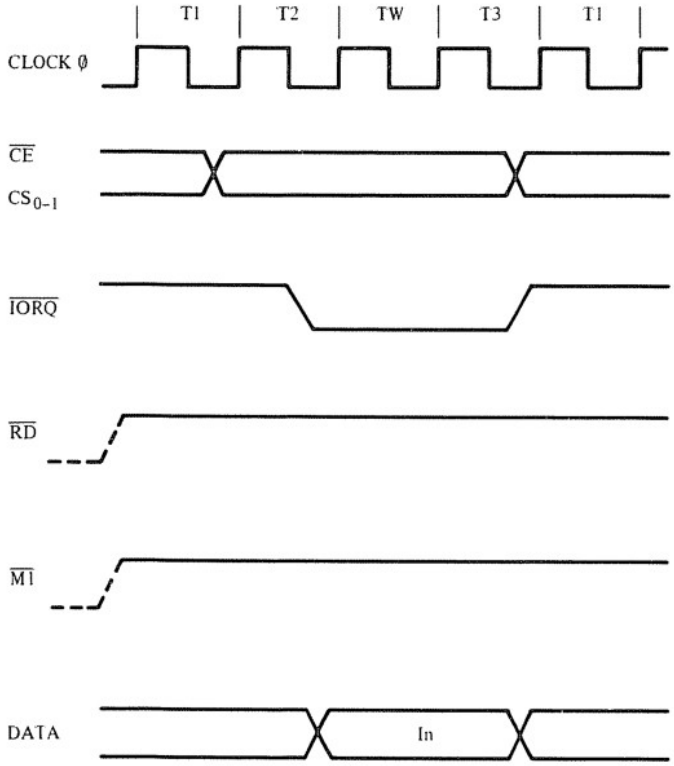


Figure 8-4 CTC Write Cycle Timing

signals in the active or low state. Notice that during this time $\overline{M1}$ must be inactive—that is, in the high state—otherwise it will be misinterpreted as an interrupt acknowledge cycle.

5. Also, during the T2 clock period, the two-input channel select code appears on lines CS0 and CS1. This, of course, specifies which of the four available CTC channels is selected for that particular operation.
6. Then, during T2, the CPU puts the word to be written into the CTC chip on the CPU data bus lines D0–D7.
7. the TW wait state is automatically inserted by the CPU. At the leading edge of clock ø T3 the word output by the CPU is latched into the appropriate internal register of the CTC.

8-6.2 The CTC Read Cycle

1. The function of the read cycle involves reading of the current contents of the down counter by the CPU. This operation does not destroy or alter the contents of the down counter.
2. The CPU initiates the read cycle during the T2 clock period by both the $\overline{RD}$ and the $\overline{IORQ}$ signals going to the active or low states. Note that the $\overline{CE}$ is also active during this time.
3. At the same time, the 2-bit binary code CS0 and CS1 also appears and selects the appropriate channel.
4. Once again note that the $\overline{M1}$ signal is inactive (i.e., high) during this time, thereby ensuring that this cycle is not misinterpreted as an interrupt acknowledge cycle.
5. The contents of the down counter are then placed on the CPU data bus during the rising edge of the clock ø T3 as shown in Fig. 8-5. Notice that this word is available on the data bus only during the T3 period.
6. The wait state TW is automatically inserted by the CPU.

8-6.3 The CTC Counter Mode

1. Figure 8-6 shows the timing of the CTC operation in the counter mode. The function of the CTC in this mode is to count the number of incoming signals on the CLK/TRG input and then output a signal on the ZC/TO line when the preestablished number of incoming signals has been counted by the down counter going to 0.

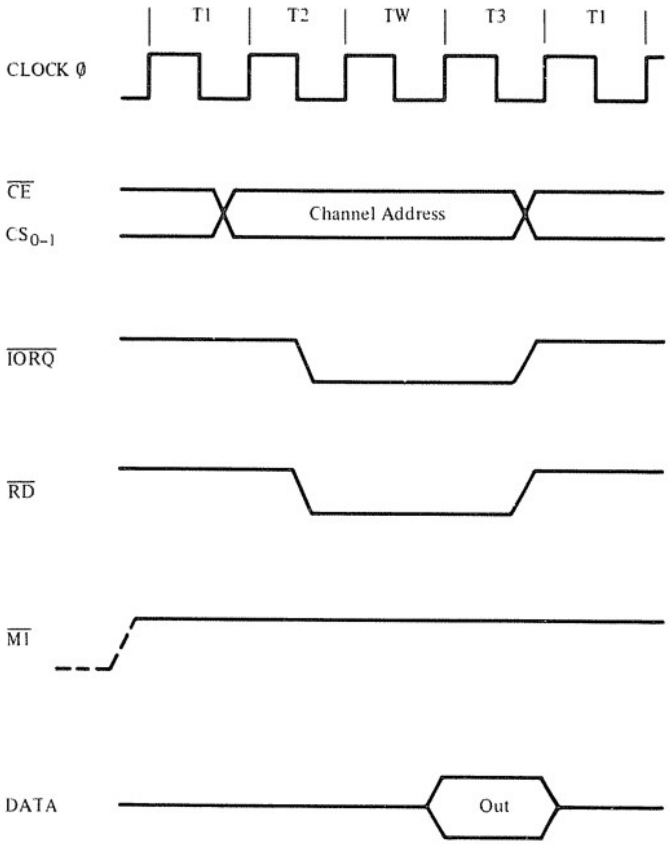


Figure 8-5 CTC Read Cycle Timing

2. In the example shown in Fig. 8-6 the CTC is programmed to respond to the positive-going input signals on the CLK/TRG line. (*Note:* This is accomplished by a 1 bit in the D4 position of the channel control word.)
3. Each incoming signal on the CLK/TRG line decrements the down counter. This is of course synchronized with the clock ø.
4. In Fig. 8-6 notice that the CLK/TRG pulse, identified as 1, is the last input that makes the down counter go from a 1 to a 0. At the rising edge of the next clock ø, T2, the down counter goes to 0. Almost immediately a pulse is sent out on the ZC/TO line.

8-6.4 The CTC Timer Mode

1. Figure 8-7 shows the timing of the CTC operation in the timer mode. The function of the CTC in this mode is to provide a signal on the ZC/TO line when a certain, preprogrammed time interval has elapsed after the timing operation has been started.
2. In this mode a trigger pulse at the CLK/TRG input starts the process. The CTC could be triggered by either the positive-going or the negative-going edge, depending on how the D4 bit is programmed in the channel control word.
3. As seen in Fig. 8-7, the internal timing process starts on the leading edge of the next clock ø (i.e., T2 in this case), since the trigger input can arrive at the CTC any time. In other words it is an asynchronous input.
4. When the programmed time has elapsed, as indicated by the down counter reaching 0, a pulse is sent out on its ZC/T0 line, such as the one shown in Fig. 8-6 for the counter mode.
5. The pulse on the ZC/T0 line is not shown in Fig. 8-7, since the elapsed time is a variable quantity which is different in each application.

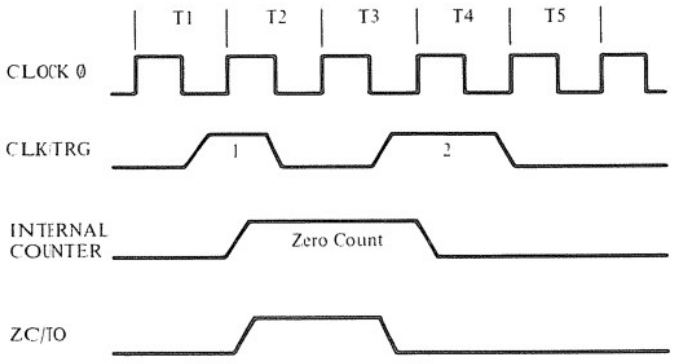


Figure 8-6 CTC Counter Mode Timing

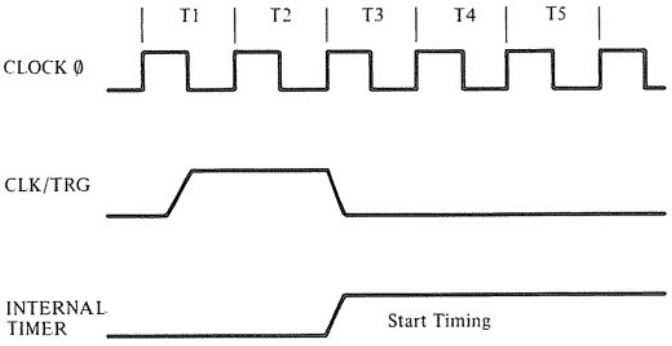


Figure 8-7 CTC Timer Mode Timing

8-6.5 Interrupt Acknowledge Cycle Timing

1. As we have previously discussed, the CTC has the capability of being programmed for the generation of an interrupt request signal $\overline{INT}$ every time the down counter reaches zero.
2. Figure 8-8 shows the timing involved in the interrupt acknowledge cycle.
3. Sometime after receiving an $\overline{INT}$ signal, the CPU acknowledges the request by activating the $\overline{M1}$ and $\overline{IORQ}$ lines.
4. During this time the $\overline{RD}$ line is deactivated (i.e., is high); otherwise it might be interpreted as an instruction fetch cycle.
5. In Fig. 8-8 notice that the $\overline{IORQ}$ signal is activated about two clock periods after the $\overline{M1}$ is activated.

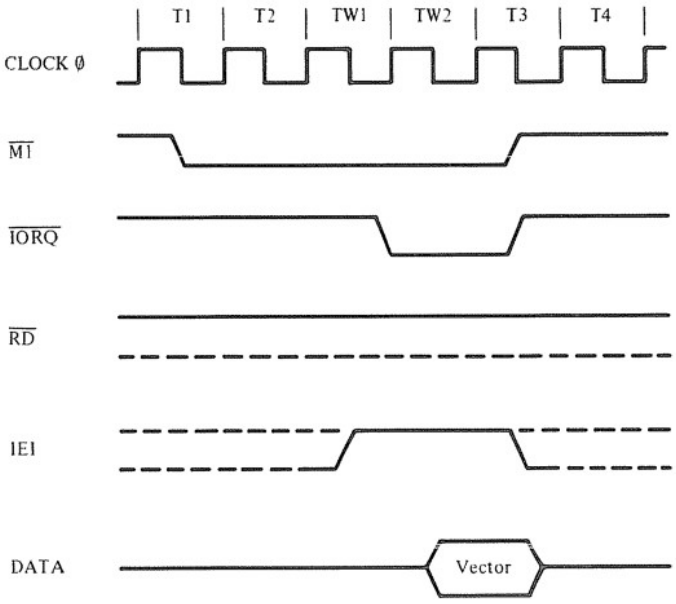


Figure 8-8 Interrupt Acknowledge Cycle Timing

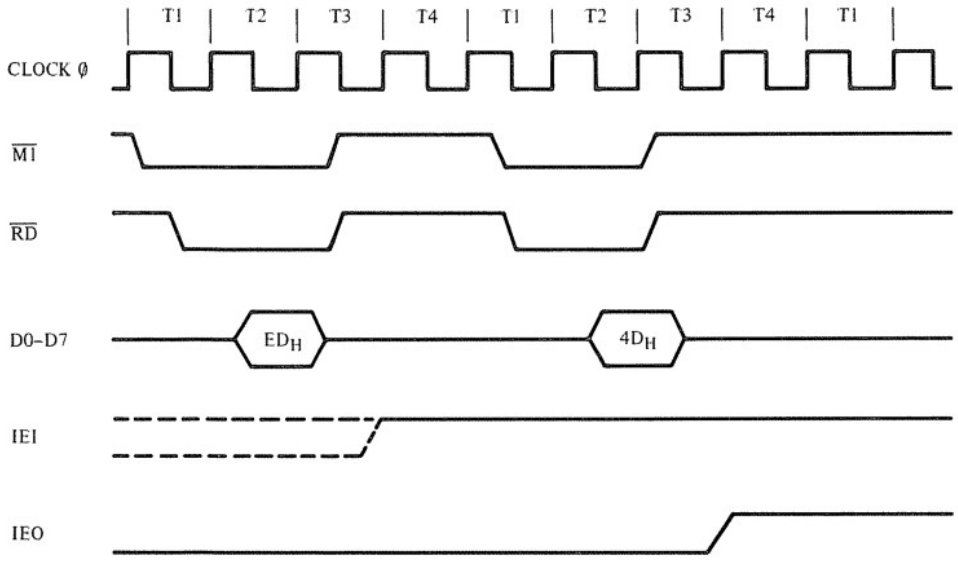


Figure 8-9 Return from Interrupt Timing

When $\overline{M1}$ is active, the channels cannot change their interrupt status, thus ensuring that the daisy chain lines have ample time to stabilize.

6. During this time interval the interrupt logic in the CTC chip will determine the highest priority channel requesting the interrupt.
7. The two wait states, TW1 and TW2, are automatically inserted by the CPU to enable the daisy chain lines to stabilize.
8. Then, during the TW1 clock period, if the CTC IEI input is put in the high state by the external device (thereby indicating that it has the highest priority in the daisy chain), the CTC places the appropriate interrupt vector on the CPU data bus since both $\overline{IORQ}$ and IEI are active.

8-6.6 Return from Interrupt Timing

1. At the end of each interrupt service subroutine, the RETI instruction is used. This instruction sets up the daisy chain for proper control of the interrupt priority handling logic for the daisy chain.
2. Figure 8-9 shows the timing involved with the execution of the RETI instruction. The 2-byte RETI code is internally decoded by the CTC.
3. The EDH OP code is placed on the data bus when $\overline{M1}$ and $\overline{RD}$ are simultaneously active. The IEI line is then activated, and if the following op code is 4DH, then the I/O device which is being serviced will be reinitialized. After transfer of the 4DH OP code, the IEO line is activated.

8-7 REVIEW QUESTIONS

8-1 The following statements refer to the CTC chip. Indicate true or false.

a. ____ The four channels on the CTC are independent of each other.

b. ____ When used in the counter mode, the CTC counts the number of pulses input to it.

c. ____ When used in the timer mode, the CTC generates time intervals by dividing the system clock frequency.

d. ____ When used in either counter or timer mode, channel 3 has the highest priority and channel 0 the lowest.

e. ____ The CTC can be directly interfaced with the Z80 SIO for generating the required baud rate.

8-2 Answer yes or no for the CTC chip.

a. ____ When used in the timer mode, can the CPU read the number of counts still to go before 0?

b. ____ In either counter or timer mode, when the down counter reaches 0, can it be reloaded automatically?

c. ____ Is it possible to initiate timer operation by means of a negative-going trigger?

8-3 Refer to the generalized CTC block diagram of Fig. 8-1 and indicate ture or false.

a. ___ The four channels can be connected into four consecutive slots of the Z80 daisy chain.

b. ___ A single interrupt vector is used as a common service subroutine starting address for all four channels.

c. ___ The interrupt control lines between the CTC and the CPU are all *bidirectional* lines.

d. ___ The read/write, chip enable and reset functions are handled by the block labeled *interrupt* control logic.

e. ___ During the interrupt service subroutine execution, the IEO signal is held high.

f. ___ During the interrupt service subroutine execution, the IEO signal is used to inhibit interrupt requests from I/O devices with lower priorities.

8-4 The following statements relate to the channel logic block diagram of Fig. 8-2. Fill in the blanks.

a. The channel control register receives _____ bit control word from the _____. (6; 8; 16; CPU; I/O devices; $channel_{0-3}$).

b. One of the four channels is selected by a _____ bit code from the CPU on the _____ pins. (2; 4; 6; 8; CS_{0-1}; ZC/T0; CLK/TRG)

c. The CTC pins CS_0 and CS_1 are usually connected to the _____ and _____ lines from the CPU. (D0; D1; A0; A1; ZC/TO_0; ZC/TO_1)

d. Bit _____ of the channel control word selects the prescaler value of _____ or _____. (D3; D5; D7; 8; 16; 64; 128; 256)

8-5 Indicate true of false for the channel logic block diagram of Fig. 8-2.

a. ___ The prescaler is used only in the counter mode.

b. ___ The divided output of the prescaler is used as the clock input to the down counter.

c. ___ The time constant register is used in both the counter and the timer operation modes.

d. ___ The time constant is an integer whose value ranges from 0 to 255.

e. ___ The time constant is loaded into the down counter when the CTC is first initialized.

8-6 Fill in the blanks for the down counter operation.

a. The down counter is _____ when used in the counter mode. (incremented; decremented)

b. The down counter is _____ when used in the timer mode. (incremented; decremented)

c. If _____ of the channel control word is a _____, the down counter is triggered automatically. (D3; D6; 0; 1)

d. The down counter is triggered by the active edge of the CLK/TRG pulse only when bit _____ of the channel control word is a _____. (D1; D2; D3; D4; 0; 1)

8-7 The following statements relate to the interrupt control logic and protocol. Indicate true or false.

a. ___ For the down counter to generate an interrupt request, the CPU must be programmed for mode 2 operation.

b. ___ Higher-priority channels can override the lower-priority channels.

c. ___ If the IEI signal is active, the CTC places its interrupt vector on the data bus.

d. ___ Bits D3–D7 of the interrupt vector are preloaded during the CTC initialization.

e. ___ The interrupt vector is utilized by the CPU as the least significant 8 bits of a 16-bit pointer.

8-8 The following statements apply to the 16-bit pointer which provides the starting address of the interrupt service subroutine. Indicate true or false.

a. ___ The upper 8 bits of the pointer are provided by the R register of the Z80 CPU.

b. ___ 16-bit starting addresses of the interrupt service subroutines are prestored in a table in the memory.

c. ___ The low-order byte of the address is placed in an even location in the memory.

d. ___ The LSB of the interrupt vector will always be a 0.

e. ___ The first instruction in all interrupt service subroutines is always the RETI instruction.

8-9 Fill in the blanks in the following statements.

a. The machine cycle 1 signal is active _____. (high; low)

b. The CPU is fetching an instruction from the memory when M1 and _____ are simultaneously active. ($\overline{RD}$; $\overline{IORQ}$; $\overline{CE}$)

c. The CPU is acknowledging an interrupt when $\overline{M1}$ and _____ are simultaneously active. ($\overline{CE}$; $\overline{IORQ}$; $\overline{IEO}$; $\overline{IEI}$; $\overline{RESET}$)

d. Assuming that it has the highest priority, the CTC places its interrupt vector on the CPU data bus after _____ and _____ are simultaneously active. ($\overline{RD}$; $\overline{CE}$; $\overline{RESET}$; $\overline{IORQ}$; $\overline{IEI}$; $\overline{M1}$)

8-10 _____ Name the signal that the CPU sends to the CTC for sending the channel control word to the CTC as well as for transferring data between the CPU and the CTC.

8-11 The CTC write cycle is initiated by the active status of _____ and _____ signals and the inactive status of the signal. ($\overline{CE}$; $\overline{RD}$; $\overline{M1}$; $\overline{IORQ}$)

8-12 To transfer the contents of the down counter to the CPU, the following signals must be simultaneously active (circle all that apply):
- **a.** $\overline{CE}$
- **b.** $\overline{RD}$
- **c.** $\overline{M1}$
- **d.** $\overline{IORQ}$
- **e.** IEO
- **f.** IEI

8-13 Circle those signals that are active during the CTC write cycle.
- **a.** $\overline{CE}$
- **b.** $\overline{RD}$
- **c.** $\overline{M1}$
- **d.** $\overline{IORQ}$
- **e.** IEO
- **f.** IEI

8-14 Which of the following signal(s) will stop the down counting process of all channels in the CTC?
- **a.** $\overline{RESET}$
- **b.** $\overline{RD}$
- **c.** $\overline{IORQ}$
- **d.** $\overline{M1}$

8-15 Which of the following signal(s) will reset the interrupt enable bits in all control registers? (Circle the one(s) that apply.)
- **a.** $\overline{RD}$
- **b.** $\overline{M1}$
- **c.** $\overline{RESET}$
- **d.** $\overline{IORQ}$

8-16 Channel 1 is programmed to enable interrupt. Which of the following signal(s) will be issued by the CTC when the down counter reaches 0?
- **a.** $\overline{RESET}$
- **b.** $\overline{IEI}$
- **c.** $\overline{INT}$
- **d.** $\overline{IORQ}$

8-17 The following statements apply to the signals that are communicated between the I/O devices and the various CTC channels. Indicate true or false.
- **a.** ___ The CLK/TRG signal can be selected to be active either high or low.
- **b.** ___ When used in the timer mode, each predefined signal on the CLK/TRG pin decrements the down counter.
- **c.** ___ When used in the counter mode, each predefined signal on the CLK/TRG pin initiates the timing function.
- **d.** ___ When the down counter goes to 0, the ZC/TO output signal goes high.

8-18 Does the reset signal program the CTC chip? Check one.
- **a.** ___ Yes.
- **b.** ___ No.

8-19 Indicate true or false for the CTC counter mode operation.
- **a.** ___ The CLK/TRG inputs *do not* have to come to the CTC at regular time intervals.
- **b.** ___ The new time constant will be loaded only after the down counter has decremented to 0.
- **c.** ___ When the down counter of channel 3 reaches 0, a pulse is output on the ZC/TO_3 line.
- **d.** ___ When the down counter reaches 0, the time constant register reloads the down counter with the time constant word.

8-20 The following statements relate to the CTC timer mode operation. Fill in the blanks.
- **a.** The elapsed time is given by the _____ of the system clock period, the prescaler factor, and the time constant. (sum; product)
- **b.** The system clock ø is passed through two _____ which _____ the frequency at each stage. (register; counters; adds; divides; multiplies)
- **c.** The output of the _____ is used as an input clock for the _____. (prescaler; clock ø; down counter; time constant register)

9
PROGRAMMING THE CTC CHIP

CHAPTER OBJECTIVES

The objectives of this chapter are as follows:

1. To introduce the concept of customizing the CTC chip for specific applications involving counting and timing functions.
2. To describe the function and format of the channel control word.
3. To discuss the function of the time constant register and the format of the time constant data word.
4. To present and discuss the formation of the 16-bit pointer for the starting address of the interrupt service subroutine when the CTC is programmed for mode 2 interrupt response.
5. To set up and program the CTC chip for a typical operation in the counter mode by means of an example.
6. To demonstrate how the CTC can be programmed for the timing operation by means of a typical example.

TEXTBOOK REFERENCES

For reviewing material in the textbook, relevant to the topics covered in this chapter, the following chapters and/or sections are suggested:

1. For general discussion of counters and timers	Chapter 14
2. For the time constant register	Sec. 14-2.2
3. For the prescalar	Secs. 14-2.3 and 14-6.2
4. For the interrupt feature	Sec. 14-2.4
5. For timer mode operation	Secs. 14-2.5 and 14-7.2
6. For counter mode operation	Sec. 14-2.6
7. For counter-timer chip architecture	Sec. 14-3
8. For chip initialization words	Sec. 14-3.2
9. For channel control logic	Sec. 14-3.3
10. For prescale factor selection	Sec. 14-6
11. For counter applications	Sec. 14-8.1
12. For timer applications	Sec. 14-8.2

9-1 INTRODUCTION

The CTC is a programmable interface chip which, under program control, can be customized to perform functions for specific applications within certain limits. Customizing the CTC requires that the chip be set up first by means of an initialization program, which in this case consists of only three words. In this chapter we describe the initialization process and the words involved in such a program. Also, examples are included to show how this chip can be programmed for some typical counting and timing applications.

9-2 THE PROGRAMMING WORDS

In this section we describe and discuss the formats of the various initialization words. These words are presented in the sequence in which they would be included in the initialization program.

9-2.1 The Interrupt Vector

For interrupt operations with the CTC, the Z80 CPU must be programmed for mode 2 interrupt responses. This mode, which is the most versatile and powerful interrupt response, was described in Chapter 4. Students are referred to Sec. 4-4.6.3 and Fig. 4-1 for a review of this mode.

In mode 2 interrupt response, the CPU receives an interrupt request from one of the CTC channels and then responds with an interrupt acknowledge. The true or effective starting address of the appropriate interrupt service subroutine is stored in a starting address table in the memory. This address is called out by a 16-bit address pointer, using the indirect addressing mode. This 16-bit pointer is formed by concatenating the prestored contents of the I register in the CPU, which forms the upper 8 bits, and the 8-bit vector provided by the CTC, which forms the lower 8 bits. Figure 9-1 shows the formation of this pointer. Notice that bit 0 of this vector is always 0. The 0 in bit 0 position of the vector identifies it as the interrupt vector. As we shall see in Sec. 9-2.2, a 1 bit in bit 0 position will identify it as a channel control word. The 2-byte effective starting address of the subroutine is always stored in the table in the memory with the low-order byte first and the high-order byte in the next addressed location. Since bit 0 of the interrupt vector is always a 0, this means that the 16-bit pointer will always point to an even address in the starting address table.

During initialization of the CTC, the interrupt vector is loaded first. The format of this 8-bit word was previously presented in Sec. 8-3.3. For convenience it is repeated in Fig. 9-2. In this word the upper 5 bits, D3–D7, are supplied by the programmer. Bit D0 is always a 0 (as previously explained). The 2 bits for positions D1 and D2 are automatically supplied by the interrupting channel

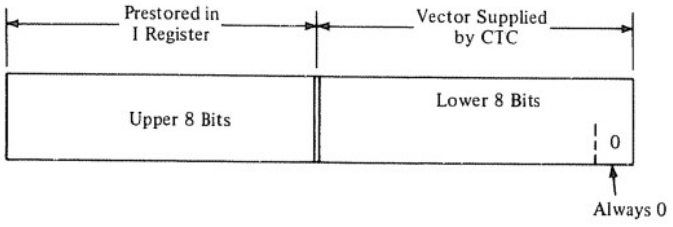


Figure 9-1 Pointer for Interrupt Subroutine Starting Address

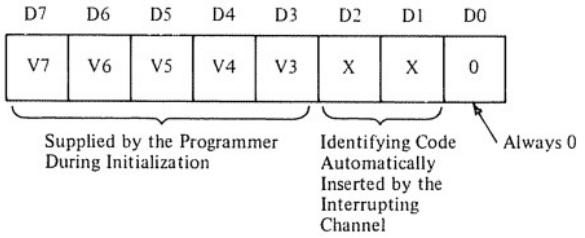


Figure 9-2 Interrupt Vector Format

according to the codes in Table 8-2. These codes are also repeated for convenience in Table 9-1.

During initialization the CPU writes the interrupt vector into the CTC chip by writing into the I/O port address corresponding to channel 0 of the CTC. This means that bits D2, D1, and D0 must all be 0 during the writing in process. Also remember that channel 0 has the highest priority and channel 3 the lowest.

9-2.2 The Channel Control Word

The next word in the initialization program is the channel control word which is loaded in the 8-bit channel control register. (See Fig. 8-2.) The format of the channel control word and the function of each bit in that word were described in Sec. 8-3.2 in reasonable detail and will not be discussed here. However, for convenience, the format of the channel control word is repeated in Fig. 9-3 and the meanings of each of the bits are also given following Fig. 9-3. Note that bit position D0 is always a 1 for the channel control word.

D0 Word identifier
Always 1 for channel control word
D1 Reset
1 Software reset
0 Continued operation

Table 9-1 INTERRUPT CHANNEL IDENTIFICATION CODES

	D2	*D1*
Channel 0	0	0
Channel 1	0	1
Channel 2	1	0
Channel 3	1	1

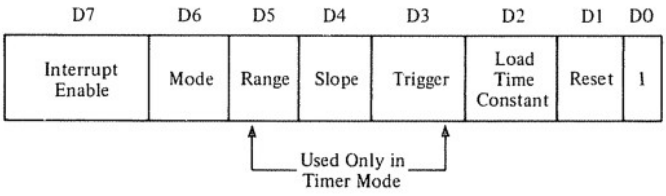


Figure 9-3 Channel Control Word Format

D2 Load time constant
- 1 Time constant follows
- 0 No time constant follows

D3 Trigger
- 1 Clock/trigger pulse starts the timer
- 0 Automatic trigger when time constant is loaded

D4 Slope
- 1 Selects positive-going edge of clock/trigger pulse
- 0 Selects negative-going edge of clock/trigger pulse

D5 Range
- 1 Indicates prescaled value of 256
- 0 Indicates prescaled value of 16

D6 Mode
- 1 Indicates counter mode
- 0 Indicates timer mode

D7 Interrupt enable
- 1 Enables interrupt
- 0 Disables interrupt

The following additional explanations are intended to aid students in understanding and composing the channel control word for the initialization program.

1. When bit D1 is set (i.e., it is a 1), the channel involved stops the timing or the counting process. In other words, the current operation of the channel is terminated. However, if both bits D1 and D2 are concurrently 1, the channel will resume operation when a time constant word is loaded. If D1 is a 0, the channel continues with the current operation.
2. A 1 in the D2 bit position informs the logic in the selected CTC channel that the next word is a time constant word which should be loaded into the time constant register (Fig. 8-2). A 0 in this bit position indicates that no time constant is to follow and that this channel control word is written to update the status of the channel that is currently in operation.
3. Bit position D3 is active only when CTC is operating in the timer mode. When this bit is set, an external trigger pulse starts the timing operation after the leading edge of T2 of the machine cycle following the one that loads the time constant word as shown in the timing chart of Fig. 8-7.
4. A 0 in the D7 position disables the interrupt-generating capability of the selected CTC channel. On the other hand, a 1 in this position results in the generation of an interrupt signal on the $\overline{INT}$ line every time the down counter reaches the 0 condition. This feature is applicable in both the timer and the counter modes. It is of course possible to write an updated channel control word while the particular channel is already in operation. In such a case, if the previous channel control word had a 0 in the D7 position, then a 1 in the updated word will not preempt the previously defined condition and generate an interrupt when the down counter reaches the 0 count condition. The conditions specified by the updated channel control word become operative only after completion of the process defined by the previous channel control word.

9-2.3 The Time Constant Word

The format of the time control word, which is loaded into the time constant register of Fig. 8-2, is shown in Fig. 9-4. It is an 8-bit word whose integer value ranges from 1 to 256. If all 8 bits are 1s, its value is 255. If all 8 bits are 0s, its value is interpreted as 256.

The logic of the CTC is designed such that a channel will not operate, in either the counter or the timer mode, unless a time constant word is previously loaded in the time constant register of that channel. This means that in the initialization program, bit position D2 of the channel control word must contain a 1. If it is a 0, it only means that the channel is already in operation and that the new channel control word is only an update. Note that the time constant word can be written into the time constant register at any time without interfering with the operation of the down counter, which may currently be in the process of counting down. The down counter will continue decrementing to 0 and the new time constant word will be loaded into it only after the count has reached 0.

9-3 PROGRAMMING FOR COUNTER APPLICATIONS

Example 9-1

The interrupt page address register in the CPU (the I register) is preloaded with $1D_H$ and the CPU is preprogrammed for mode 2 interrupt response. The starting address table is stored from memory location $1D18_H$ to $1D1F_H$. Channel 0 is operated in the counter mode and interrupt logic is enabled. The channel is operated with the initial time constant of $3E_H$ and the down counter is decremented on the positive-going edge of the input trigger pulse. Channels 1, 2, and 3 are inactive. Write the program words required to initialize

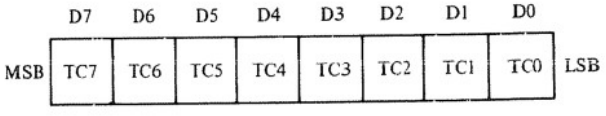


Figure 9-4 Time Constant Word Format

the CTC for this problem in both hexadecimal and machine codes and explain what you did and why.

Solution

The first initialization word that must be loaded in the CTC chip is the interrupt vector. Since the address of the starting address table, stored in the memory, starts with $1D18_H$, the pointer for the channel 0 interrupt service subroutine is $1D18_H$. Since $1D_H$ is preloaded into the I register of the CPU, the interrupt vector from the CTC must supply the lower 8 bits of the pointer as shown in Fig. 9-1. Therefore, the interrupt vector for channel 0 must be 18_H. The machine code for this is shown below.

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	1	1	0	0	0

As explained in Sec. 9-2.1, bit D0) is a 0. The five high-order bits (i.e., D7–D3) are supplied by the programmer. Although bits D2 and D1 are given as 0 and 0, respectively, the interrupting channel will automatically insert its proper identifying code in these two bit positions, as shown in Table 9-1. The channel control word is the next word to be loaded.

1. Since the interrupt is enabled, bit D7 is a 1.
2. Bit D6 is a 1 because we are operating in the counter mode.
3. Since bits D5 and D3 are both applicable only in the timer mode, the bits in these two positions are not relevant in this case. We will, therefore, arbitrarily insert two 0s here.
4. Bit D4 is a 1 because the problem statement calls out for the down counter to be triggered on the positive-going trigger input pulse.
5. Bit D2 is a 1 because a time constant word follows the channel control word.
6. We put a 1 in bit D1 position. This terminates the current operation of channel 0, but since D2 is also a 1, channel 0 will resume operation when the time constant word is loaded.
7. Bit D0 is always a 1 and it identifies this word as the channel control word.

The complete channel control word is shown below:

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	1	0	1	1	1

The next word to be loaded is the time constant word which is given in the problem statement as $3E_H$. The machine code for this is

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	1	1	1	1	0

The three CTC initialization words for this problem are shown in Table 9-2 with both the machine codes and the hexadecimal codes. ■

Example 9-2

Refer to the problem statement of Example 9-1. Using the same data, construct the three CTC initialization words for channel 1 of the CTC but with the following two exceptions.

1. The down counter must be triggered on the negative-going edge of the input trigger pulse.
2. Channel 1 must generate an interrupt request when 254 input trigger pulses have been received.

Additional explanations need not be given. Give both the machine and the hexadecimal codes in tabular form similar to Table 9-2.

Solution ■

9-4 PROGRAMMING FOR TIMER APPLICATIONS

Example 9-3

Design the three initialization words for operating the CTC in the timer mode for the following conditions:

1. Channels 0 and 1 are previously programmed for operation in the counter mode.

Table 9-2 CTC INITIALIZATION WORDS FOR EXAMPLE 9-1

	Hex Codes	*D7*	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*
Interrupt vector	18	0	0	0	1	1	0	0	0
Channel control word	D7	1	1	0	1	0	1	1	1
Time constant word	3E	0	0	1	1	1	1	1	0

Table 9-3 CTC INITIALIZATION WORDS FOR EXAMPLE 9-2

	Hex Codes	*D7*	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*
Interrupt vector	18	0	0	0	1	1	0	0	0
Channel control word	C7	1	1	0	0	0	1	1	1
Time constant word	FE	1	1	1	1	1	1	1	0

2. Channel 2 is to be programmed for the timer mode and channel 3 is not used.
3. The I register in the Z80 CPU has been preloaded with $A4_H$.
4. Memory locations $A408_H$ through $A40F_H$ are reversed for storing the interrupt service subroutine starting addresses.
5. A prescaled value of 16 is used.
6. The time constant to be used is 23_H.
7. The interrupt vector is 08_H.
8. The interrupt is enabled.
9. The down counter is triggered on the positive-going edge of the clock timer input pulse.

Give both the hexadecimal and the machine codes for each word in tabular form and explain the procedure used as appropriate.

Solution

1. We program the interrupt vector for channel 2. When channel 2 generates an interrupt, it will automatically insert the appropriate channel identifier code (Table 9-1) in bit positions D2 and D1.
2. Thus, the interrupt vector is 08_H = 00001000.
3. It is assumed that the software reset is set and so bit position D1 is a 1.
4. Since a prescaled value of 16 is indicated, a 0 is used in bit position D5.
5. A 1 is used in D4 because a positive-going edge of the input pulse is used for triggering the timer.
6. A 1 is used for D3 because a trigger pulse is used for starting the timer.
7. Since a time constant word follows, a 1 is used in the D2 bit position.
8. The three set-up words are shown in Table 9-4. ■

Example 9-4

Design an initialization program for the CTC to operate in the timer mode for the following conditions and present both the hexidecimal and the machine codes in tabular form. Explanations may be inserted as appropriate.

1. The interrupt is enabled.
2. The clock is triggered on the negative-going edge of the input pulse.
3. Software reset is set.
4. A 2.5-MHz clock is used.
5. The interrupt vector is EO_H.
6. Channel 1 is to be programmed for this operation.
7. A time-out pulse must be output by the timer every 1.536 ms.
8. The I register in the CPU is preloaded with 44_H.
9. The interrupt subroutine starting address table is stored in memory locations $44EO_H$ to $44E7_H$.
10. A prescaler value of either 16 or 256 may be used.
11. A time constant of either 08_H or $0F_H$ may be used.

Solution

In this problem there are two variables which are unknown, namely, the prescaler factor (PF) and the time constant integer (TC). Since the elapsed time (T_e)

Table 9-4 CTC INITIALIZATION WORDS FOR EXAMPLE 9-3.

	Hex Codes	*D7*	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*
Interrupt vector	08	0	0	0	0	1	0	0	0
Channel control word	9F	1	0	0	1	1	1	1	1
Time constant word	23	0	0	1	0	0	0	1	1

Table 9-5 CTC INITIALIZATION WORDS FOR EXAMPLE 9-4.

	Hex Codes	*D7*	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*
Interrupt vector	EO	1	1	1	0	0	0	0	0
Channel control word	AF	1	0	1	0	1	1	1	1
Time constant word	OF	0	0	0	0	1	1	1	1

is 1.536 ms, we use the formula given in Sec. 8-5.3 to arrive at the values for PF and TC. By trial and error the values are found to be,

$$\text{PF} = 256 \text{ and } \text{TC} = 0\text{F}_\text{H}.$$

Since the Z80 system uses a 2.5-MHz clock, the system clock period is $1/2.5 \times 10^{-6} = 0.4 \times 10^{-6}$

Then the elapsed time between the time-out signals is given by

$$\begin{aligned} T_e &= t_c \times \text{PF} \times \text{TC} \\ &= 0.4 \times 10^{-6} \times 256 \times 15 \\ &= 1536 \times 10^{-6} \\ &= 1.536 \times 10^{-3} = 1.536 \text{ ms} \end{aligned}$$

The initialization program is given in Table 9-5. ■

9-5 REVIEW QUESTIONS

9-1 For interrupt operations with the CTC, the Z80 CPU must be programmed for interrupt responses in
- **a.** Mode 0.
- **b.** Mode 2.
- **c.** Mode 1.

9-2 The starting addresses of the interrupt service subroutine is called out by using the (circle all that apply):
- **a.** Register addressing mode.
- **b.** Extended addressing mode.
- **c.** Indirect addressing mode.
- **d.** Implied addressing mode.

9-3 The following statements relate to the starting address of the interrupt service subroutine. Indicate true or false.
- **a.** ___ The effective subroutine starting addresses are stored in a table in the memory.
- **b.** ___ An 8-bit address pointer is used for calling out the starting address.
- **c.** ___ The upper 8 bits of the address pointer are contained in the R register in the CPU.
- **d.** ___ The lower 8 bits of the address pointer are supplied by the CTC.

9-4 Bit 0 of the CTC interrupt vector is always a
- **a.** 0.
- **b.** 1.

9-5 The 16-bit effective starting address of the interrupt service subroutine is always stored in the memory table with the _____ order byte first, followed by the _____ order byte in the next addressed location. (low-; high-)

9-6 The 16-bit address pointer always points to an _____ address in the starting address table. (odd; even)

9-7 In the interrupt vector, the code for the following bit position(s) is (are) automatically supplied by the interrupting channel:
- **a.** D0.
- **b.** D0 and D1.
- **c.** D1 and D2.
- **d.** D0 and D2.
- **e.** D0 and D3–D7.
- **f.** D3–D7.

9-8 The channel with the highest priority in the CTC is:
- **a.** Channel 2.
- **b.** Channel 3.
- **c.** Channel 1.
- **d.** Channel 0.

9-9 The following statements apply to the bits in the channel control word. Indicate true or false.
- **a.** ___ A 1 in the D2 position indicates that a time constant follows:
- **b.** ___ A 1 in the D1 bit position indicates that software reset is enabled.
- **c.** ___ The timer is started by the clock/trigger pulse if there is a 1 in the D3 position.
- **d.** ___ A 0 in the D5 bit position indicates a prescaled value of 16.

9-10 The I register in the CPU is preloaded with 3D_H and channel 2 is operated in the counter mode. The 16-bit address pointer for the starting address of the interrupt service subroutine is:
- **a.** 3DO1_H.
- **b.** 3DO4_H.

c. $3DAB_H$.
d. $3DD9_H$.

9-11 Channels 0 and 1 are operated in the counter mode and channels 2 and 3 are inoperative. The preloaded time constant word for both channels is 29_H. The channel control word for channel 0 is CF_H, and for channel 1 it is $D7_H$. Indicate true or false.
a. ___ In channel 0 the down counter is triggered automatically when the time constant word is loaded.
b. ___ In channel 1 the negative-going edge of the clock/trigger pulse starts the down counter.
c. ___ Software reset is set for channel 1.
d. ___ In channel 0 the positive-going edge of the clock/trigger pulse starts the down counter.
e. ___ In channel 0 a clock/trigger pulse starts the down counter.

9-12 Is it possible to write an updated channel control word while the particular channel is already in operation?
a. Yes
b. No

9-13 When there is a 1 in the D7 position of the channel control word, every time the down counter reaches the 0 condition, a signal is generated in one or more of the following signal line(s):
a. IEI.
b. IEO.
c. $\overline{INT}$.
d. Clock/trigger pulse line.

9-14 The word statements relate to the situation where the previous channel control word had a 0 in the D7 bit position and the current channel control word has a 1 in the D7 bit position. Indicate true or false.
a. ___ A 0 in the D2 position indicates that the current channel control word updates the status of the channel currently in operation.
b. ___ A 0 in the D2 position indicates that a time constant word does not follow the channel control word.
c. ___ The current channel control word preempts the previously defined condition and generates an interrupt when the down counter reaches 0.
d. ___ The conditions specified by the current channel control word becomes operative as soon as this word is loaded.

9-15 Channels 0 and 1 are operated in the timer mode and channels 2 and 3 are inactive. The I register is preloaded with 10100011. The CTC interrupt vector is 00111010. The following statements relate to the address pointer for the starting address table stored in the memory. Indicate true or false.
a. ___ The address pointer for channel 0 is $A33A_H$.
b. ___ The address pointer for channel 1 is $A33A_H$.
c. ___ The address pointer for channel 0 is $A338_H$.
d. ___ The address pointer for channel 1 is $A3A3_H$.
e. ___ The address pointer for channel 1 is $A33A_H$.
f. ___ The address pointer for channel 1 is $A338_H$.

9-16 Channel 2 of the CTC is operated in the timer mode. Which of the following pair of hexadecimal codes is appropriate for this operation? The first code in each pair is the address pointer and the second code is the channel control word.
a. BB_H and 35_H
b. $8B_H$ and $D6_H$
c. 79_H and $D2_H$
d. $3C_H$ and 35_H
e. $2E_H$ and $D6_H$

9-17 The CTC is operated in the timer mode with a prescaler factor, PF = 16 and a system clock of 2.5 MHz. The preloaded time constant, TC = 09_H. The elapsed time between the time-out signals is
a. 5.76 μs.
b. 57.6 μs.
c. 5.76 ms.
d. 576.0 ms.

9-18 Refer to question 9-17. If the prescaler factor was 256 instead of 16, the elapsed time between the time-out signals would be
a. 9.216 μs.
b. 92.16 μs.
c. 921.6 μs.
d. 9216.0 μs.

10

Z80 DIRECT MEMORY ACCESS CHIP

CHAPTER OBJECTIVES

1. To introduce students to the programmable DMA chip.
2. To present the principal features of the three classes of operation in the Z80 DMA, namely,
   a. Transfer only.
   b. Search only.
   c. The combined search-transfer operation.
3. To introduce the modes of operation, namely,
   a. Byte-at-a-time mode.
   b. Burst mode.
   c. Continuous mode.
   d. Transparent mode.
4. To present the generalized architecture of the DMA chip and describe the function of each entity in the block diagram.
5. To present the package pin assignments and functions in convenient tabular form.
6. To examine the use and operation of the DMA in the daisy chain interrupt scheme of the Z80 system.
7. To discuss the pulse generation scheme used to keep track of the number of bytes transferred in the DMA operation.
8. To present the bytes used for initialization and the bytes used for control purposes.
9. To discuss the concept of the variable cycle length time and auto restart.
10. To present and discuss the timing involved in various DMA operations such as
    a. The variable cycle.
    b. Command/control byte write.
    c. Register read.
    d. Memory-to-I/O transfers.
    e. Memory-to-memory transfers.
    f. I/O-to-memory transfers.
    g. I/O-to-I/O transfers.
    h. Bus request and acceptance.
    i. Bus release timing.

TEXTBOOK REFERENCES

For reviewing material in the textbook relevant to the topics in this chapter, the following chapter(s) and/or sections are suggested:

1. For general discussion of DMA	Chapter 10
2. For transparent mode	Sec. 10-2.1
3. For cycle stealing mode	Sec. 10-2.2
4. For byte-at-a-time DMA transfers	
a. Using single memory	Sec. 10-3.1
b. Using multiple memories	Sec. 10-3.2
5. For continuous mode single-block transfers	Sec. 10-4
6. For transfer of multiple data blocks	Sec. 10-5
7. For CPU responses to DMA request	Sec. 10-6.2
8. For operation of the DMA interface chip	Sec. 10-7
9. For byte and block count	Sec. 10-7.3
10. For byte count update	Sec. 10-7.7
11. For block count update	Sec. 10-7.8

10-1 INTRODUCTION

The Z80 DMA chip is a programmable device that provides the capability to transfer blocks of data between two ports, independent of the CPU. Data block transfers can take place either between the main memory and an I/O device, between two memory devices, or between two I/O devices. The DMA chip can also search a block of data for a specified byte (bit maskable). During such a search, a simultaneous transfer can also be accomplished, if desired.

Signals for source and destination addresses, control, and timing are provided on the chip. The DMA chip is fully compatible with the rest of the Z80 system with respect to operations involving bus requests and prioritized interrupt requests of the daisy chain. It is capable of being directly interfaced to the Z80 system without requiring any additional external logic.

10-2 PRINCIPAL FEATURES OF THE DMA

10-2.1 General and Electrical Features

1. The DMA is a third-generation chip which uses the N-channel, depletion-mode, silicon-gate technology.
2. A single 5-volt ±5% power supply is required.
3. A single phase TTL level clock is used.
4. The chip operates in a temperature environment of 0° to 70°C.
5. All inputs and outputs are TTL compatible.
6. A 40-pin DIP package is used.

10-2.2 Operational Features

1. The DMA device can operate in any of the following three classes of operation:
   a. *Transfer only.* Data block is transferred from one port to another, 1 byte at a time.
   b. *Search only.* Data blocks are not transferred. Each byte in a data block is read and compared with contents of two registers, one containing the full match byte and the other an optional mask byte if only certain desired bits are to be compared.
   c. *Search-transfer combined.* Block of data is transferred until a match is found. Then the transfer operation could be suspended and/or an interrupt request generated.
2. During a transfer, addresses for both the read and the write ports are generated.
3. The timing can be programmed to accommodate the speed of any port.
4. Both the block length and the address registers are double buffered, thereby enabling the values of the next operation to be loaded in the buffer without interfering with the value in the in-process operation.
5. Interrupts can be generated on either match found, ready, or end-of-block under program control.
6. Under program control, the DMA channel can be enabled, disabled, or reset.
7. The starting addresses that are programmed for each port (i.e., read and write port), for data transfer, or for search operations can be automatically incremented or decremented. These addresses can also remain fixed, if so desired.
8. Complete status of the DMA channel can be determined when requested by the CPU.
9. Without interfering with the data transfer operations, or stopping the transfers, the DMA can signal the CPU when a previously specified number of bytes have been transferred.
10. Any operation may be made to repeat, in its entirety, either automatically (auto restart) or on command (load).
11. The DMA can be operated at rates up to 1.25 megabytes per second for either the search or the transfer operations.
12. The DMA system can be operated in one of the following three modes, under program control, in either the search or the transfer class of operations:

a. *Byte-at-a-time.* On request, only 1 byte is transferred and then control of the system busses is returned to the CPU. A new DMA request is made for each subsequent byte to be transferred.
b. *Burst mode.* Search or transfer operations continue until such time as the port is not ready and indicates this by deactivating the corresponding ready line.
c. *Continuous mode.* The search or transfer operation continues until the end of the predetermined data block is reached. If the port is not ready for the operation before end-of-block is reached, the DMA operation pauses until the port is ready and indicates this by reactivating the ready line.

10-3 THE DMA ARCHITECTURE

1. The generalized block diagram of the DMA chip is shown in Fig. 10-1.
2. The two ports are labeled port A and port B and each port can be addressed individually. The address counter of each port is double buffered. The starting address of the data block to be searched, or transferred, is first loaded into the port start address register of the appropriate port. It is then transferred into its respective counter.
3. Each port counter then provides the required address to either the memory or the I/O device on the 16-bit address bus (A0–A15), via a multiplexer, for the byte

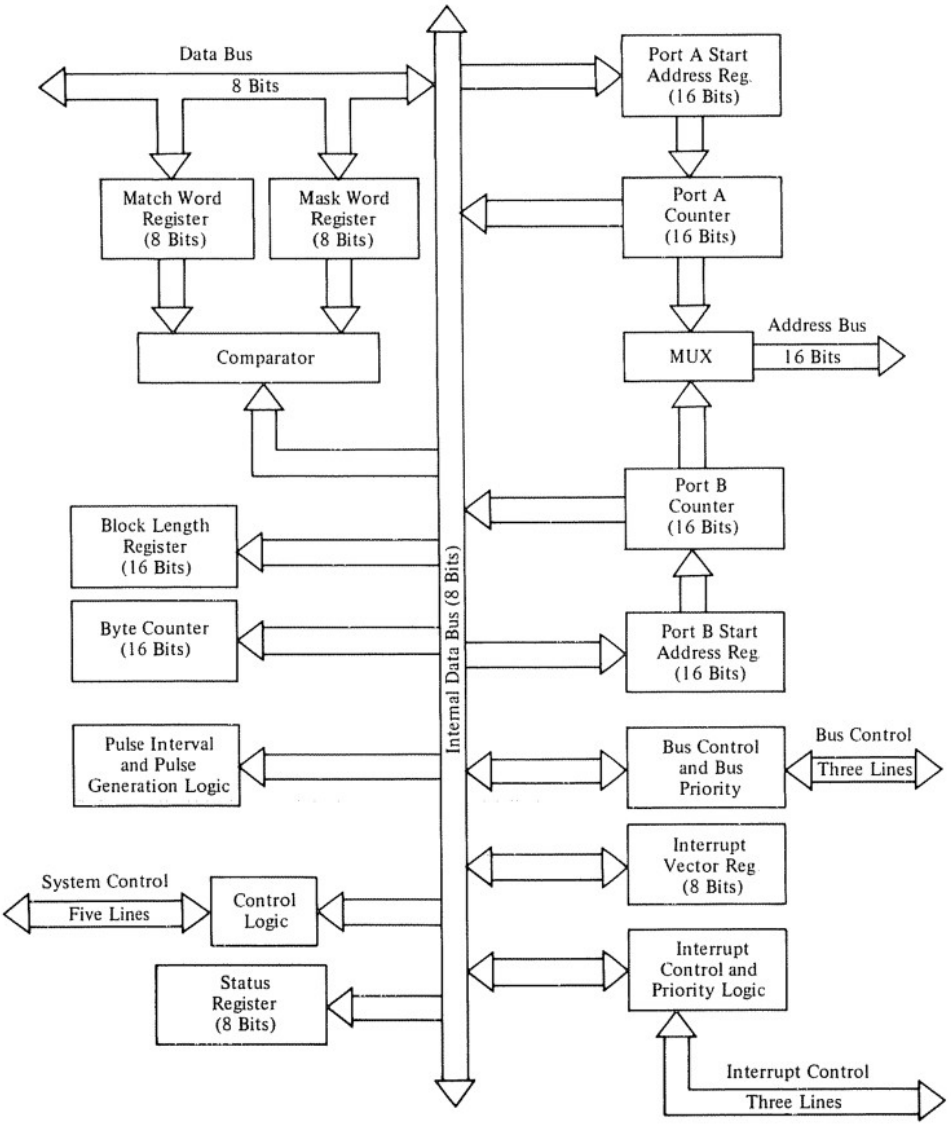


Figure 10-1 Generalized Block Diagram of the DMA Chip

to be transferred or searched. Under program control the port counters can be either incremented or decremented, or their contents can remain unaltered.

4. For data transfer operations, a 16-bit starting address is involved for both ports. The starting addresses involving memories require 2-byte addresses. The starting addresses for I/O ports generally require only 1 byte, and so only the lower byte is used. The 8-bit address for the I/O ports, contained in the address register, is usually a fixed address.
5. The 8-bit status register contains flags which indicate the current status of the DMA operation. This is a read-only register which indicates status or conditions such as end-of-block, byte match, interrupt pending, ready active, and write address bits valid.
6. The bus control and bus priority logic are concerned with three signals which are involved with requests for control of the CPU address bus, the data bus, and the status and control bus (system control bus). This logic also indicates the condition when the system buses have been released for DMA operations as well as conformance to and with the daisy chain priority system.
7. The interrupt vector register holds the 8-bit vector assigned to that particular chip and which is preloaded into this register during the initialization of the chip. During the interrupt acknowledge cycle, the contents of this register are output onto the data bus and sent to the CPU, provided that the DMA chip is the highest-priority device in the daisy chain requesting the interrupt. The vector is then utilized by the CPU for calling up the appropriate service subroutine.
8. The interrupt control and priority contains logic which is involved in establishing and/or determining the priority of the device in the daisy chain, and if it is the highest-priority device, then it blocks the lower-priority devices from interrupting during execution of the service subroutine.
9. The control logic contains control registers which are loaded with specific control bytes for performing the various DMA functions. They are loaded during the chip initialization process. The information contained in these registers include such items as the mode and class of operations and when to initiate an interrupt or other pulse.
10. The pulse interval and pulse generation logic contain an 8-bit pulse control register. During initialization this register is loaded with information on the length of the data block (a data block is defined as a group of data words which is considered as a unit and which are stored in consecutive memory locations) expressed in the number of bytes. When the entire data block is processed, the DMA emits a signal pulse on the interrupt line. This signal is not interpreted by the CPU as an interrupt signal but instead is used to communicate with a peripheral device.
11. The byte counter is a 16-bit counter which is cleared at the start of the DMA operation. Then when the processing of each byte (i.e., either a transfer or a search operation) is completed, the counter is incremented. When the count in the counter matches the contents of the block length register, the end-of-block flag in the status register is set. At this point, depending on how the DMA has been preprogrammed, the operation may be suspended and/or an interrupt signal generated.
12. The block length register is a 16-bit register which is preloaded during initialization with a number indicating the length (in bytes) of the data block to be processed during the DMA operation (i.e., either the transfer or the search operation).
13. The match word register is an 8-bit register which contains the byte for which a match is to be found during the search operation. When a match is obtained, the byte-match flag is set in the status register. Note that in this case a bit-by-bit match must be obtained for all 8 bits for the flag in the status register to be set.
14. In some cases it is desired that a match be obtained not for all the 8 bits in the match byte but for only specific bits in the word. In other words, some bits of the match word are required to be masked out. The word containing the bits to be masked is preloaded into the 8-bit mask word register. Then during the comparison process, only the unmasked bits are examined and compared for a match.
15. The comparator compares each byte to be processed (i.e. searched or transferred) with the contents of the match word register, taking into consideration the bits to be masked as indicated by the mask word register.

10-4 PACKAGE PIN ASSIGNMENTS AND FUNCTIONS

The DMA chip is contained in a standard 40-pin DIP package. The various pins and their respective functions, along with their designations, are presented in convenient tabular form. Functionally, the pins can be divided into seven groups, as shown in Fig. 10-2. Note that some lines are shown as unidirectional and others as bidirectional. In the latter group, some of the lines may indicate signal flow in one direction and others in the other direction. The function tables clarify these with the following arrow signals.

DMA ⟶ indicates signals flowing *from* the DMA.
DMA ⟵ Indicates signals flowing *into* the DMA.
DMA ↔ Indicates bidirectional flow on the same line(s).

10-5 FUNCTIONAL OPERATION OF THE DMA

The Z80 DMA is capable of performing three different functions referred to as the three classes of operations. Additionally, it can be programmed to operate in three

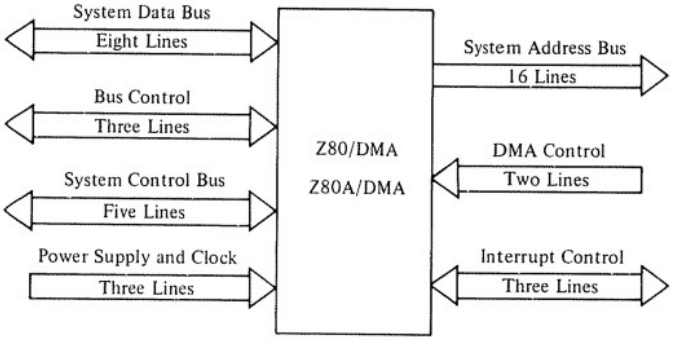


Figure 10-2 Diagram of DMA Package Pin Groups

Table 10-1 DATA TRANSFER BUS GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
D0–D7	DMA ↔	8-bit, tristate system data bus, active high. Used for transferring command words from the CPU, DMA status to the CPU and data transfer to and from memories and/or peripheral devices.

Table 10-2 POWER SUPPLY AND CLOCK GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
Ø	DMA ⟵	This is the single-phase TTL level clock input.
5 V	DMA ⟵	This is the single power supply of 5-volt ±5% that is needed for DMA.
GND	DMA ⟵	This is the power supply ground.

Table 10-3 SYSTEM ADDRESS BUS GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
A0–A15	DMA ⟶	16-bit, tristate system address bus, active high. Provides direct addressing capability for 64K byte memory and up to 256 I/O ports for DMA, transfer and search of data blocks.

Table 10-4 INTERRUPT CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
INT/PULSE	DMA ⟶	Interrupt request signal, active low. The DMA requests interrupt service from the CPU.
IEI	DMA ⟵	Interrupt enable in, active high. Used in the priority interrupt daisy chain. A high or active signal indicates that no other I/O device with higher priority is currently being serviced.
IEO	DMA ⟶	Interrupt enable out, active high. Also used in the priority interrupt daisy chain. It is active only when IEI is active and the CPU *is not* servicing an interrupt from the DMA. When active, it blocks interrupt signals from lower-priority I/O devices from interrupting while it is being serviced.

Table 10-5 BUS CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
$\overline{\text{BUSRQ}}$	DMA ⟷	Bus request, active low. As output signal it requests the CPU to relinquish control to the system data bus, the system address bus, and the control bus. When more than one DMA chip is used in the daisy chain, this signal is an input signal which prevents this DMA from making bus requests to CPU until the DMA operation of the other DMA is completed.
$\overline{\text{BAI}}$	DMA ⟵	Bus acknowledge in, active low. It is an input from the CPU informing the DMA chip that the CPU has released the buses. If more than one DMA chip is connected on the daisy chain, then the acknowledge pin of the CPU is connected to the $\overline{\text{BAI}}$ pin of only the highest-priority DMA chip. The $\overline{\text{BAI}}$ pins of the lower-priority DMA chips are connected to the $\overline{\text{BAO}}$ pin of the previous higher-priority DMA chip.
$\overline{\text{BAO}}$	DMA ⟶	Bus acknowledge out, active low. It is used when more than one DMA chip is connected in the daisy chain. When active, it signals that no other higher-priority DMA chip has currently requested control of the system buses. $\overline{\text{BAO}}$ pin receives its input from the $\overline{\text{BAI}}$ pin of the next-higher-priority DMA chip in the daisy chain.

Table 10-6 SYSTEM CONTROL BUS GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
$\overline{\text{M1}}$	DMA ⟵	Machine cycle 1 signal, active low. When $\overline{\text{M1}}$ and $\overline{\text{RD}}$ are simultaneously active, the CPU is fetching an instruction from the memory. When $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ are active simultaneously, the CPU is acknowledging an interrupt. When $\overline{\text{M1}}$ is active without the other two being active, it is used by the DMA to decode the 2-byte return-from-interrupt (RETI) instruction, whose hex codes are ED and 4D.
$\overline{\text{IORQ}}$	DMA ⟷	Tristate, I/O request signal, active low. When $\overline{\text{IORQ}}$ and $\overline{\text{M1}}$ are simultaneously active, the CPU is acknowledging an interrupt. When active by itself as an input, it informs the DMA that the lower 8 bits of the address bus are currently holding a valid I/O port address and that either the transfer of the status word from the I/O port to the CPU or the transfer of a control byte from the CPU to the port may take place. Also as an input, it is simultaneously active with $\overline{\text{CE}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ signals; it informs the DMA that this particular chip is the addressed port. After the DMA chip has acquired control of the system buses, it outputs a signal on the $\overline{\text{IORQ}}$ pin, indicating that the lower 8 bits of the address bus contain a valid address for the I/O device that is involved in the transfer of data.
$\overline{\text{MREQ}}$	DMA ⟷	Memory request, active low, tristate signal. For a memory read or write operation, it indicates that a valid memory address is available on the system address bus. After the DMA chip has obtained control of the system buses, this signal requests a data transfer either to or from the memory.
$\overline{\text{RD}}$	DMA ⟷	Tristate, active low, read signal. An input from the CPU on this pin results in a readout of the status words from the DMA register. The DMA outputs a signal on this pin, after it has obtained control of the system buses, to request a readout from either the addressed memory location or the addressed I/O port.
$\overline{\text{WR}}$	DMA ⟷	Tristate, active low, write signal. An input signal on this pin from the CPU informs the DMA that the CPU wants to write control or command words into the appropriate registers of the DMA chip. After the DMA chip has obtained control of the system buses, it outputs a signal of this pin to indicate a DMA write operation on either the addressed memory location or the addressed I/O port.

Table 10-7 DMA CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
RDY	DMA ⟵	Ready signal input which, under program control, can be made either active high or active low. The DMA chip monitors this input to establish whether a peripheral connected to a DMA port is ready for a transfer operation (read or write). Also, depending on the mode of DMA operation, this signal indirectly controls the DMA activity by activating the $\overline{\text{BUSRQ}}$ line, i.e., go either low or high.
$\overline{\text{CE/WAIT}}$	DMA ⟵	Chip enable, active low, signal from the CPU. It is active when $\overline{\text{WR}}$ and $\overline{\text{IORQ}}$ are simultaneously active. The address on the system address bus is the address of the DMA. This allows a command or control word to be transferred from the CPU to the DMA chip. The same pin can be programmed as a $\overline{\text{WAIT}}$ signal input from the memory or the peripheral device. As a $\overline{\text{WAIT}}$ signal, it inserts wait states in the DMA cycles after the DMA has received an acknowledge from the CPU in response to its bus request. The insertion of the wait states slows the DMA and establishes speed compatibility with slower memory and I/O devices.

Table 10-8 PIN NUMBER ASSIGNMENTS

Pin Designations	*Pin Numbers*	*Pin Designations*	*Pin Numbers*
System Address Bus		*System Data Bus*	
A0	6	D0	35
A1	5	D1	34
A2	4	D2	33
A3	3	D3	32
A4	2	D4	31
A5	1	D5	29
A6	40	D6	28
A7	39	D7	27
A8	24		
A9	23	*System Control Bus*	
A10	22	$\overline{\text{M1}}$	26
A11	21	$\overline{\text{IORQ}}$	10
A12	20	$\overline{\text{MREQ}}$	12
A13	19	$\overline{\text{RD}}$	9
A14	18	$\overline{\text{WR}}$	8
A15	17		
Bus Control		*Interrupt Control*	
$\overline{\text{BUSRQ}}$	15	$\overline{\text{INT}}$/PULSE	37
$\overline{\text{BAI}}$	14	IEI	38
$\overline{\text{BAO}}$	13	IEO	36
Power Supply and Clock		*DMA Control*	
Ø	7	RDY	25
5 V	11	$\overline{\text{CE/WAIT}}$	16
GND	30		

different modes. The classes and modes of operations are briefly described in this section.

10-5.1 The Classes of Operations

Basically, the DMA can be programmed to perform any of the three functional operations which involve transfer of data between two ports and/or search for a specific byte (or certain bits of a byte). The three classes of operations can be explained by referring to Fig. 10-3.

1. *Data transfers between ports.* Blocks of data can be transferred from a designated source port to a designated destination port. In Fig. 10-3 the transfers indicated are as follows:
   a. Memory-to-memory transfers (lines 1).
   b. I/O-to-I/O transfers (lines 2).
   c. I/O-to-memory (lines 5).
   d. Memory-to-I/O transfers (lines 6).

   In the DMA chip the two ports are labeled port A, and port B. Both ports are addressable and data are transferred from the source to the destination port, 1 byte at a time. The memory-to-memory transfer capability makes it possible to relocate blocks of data from one part of the main memory to another.
2. *Search operations.* Blocks of data, located either in the memory (line 3) or in the peripheral device (line 4), can be called out, 1 byte at a time, and compared with the contents of two previously loaded internal registers in the DMA chip. The first register contains a match byte and the second register contains a mask byte which, if desired, can mask out certain bits in the match byte. This enables the user to conduct a block search only on certain desired bits instead of the whole byte. Using the 2.5-MHz Z-80 DMA, search rates of up to 1.25 M bytes per second can be obtained. With the 4.0-MHz Z-80 DMA, search rates of up to 2 M bytes per second are obtained.
3. *Combined transfer-search operations.* This class of operation combines the transfer and search operations, described in paragraphs 1 and 2 above. A block of data is transferred from one port to another, as mentioned in paragraph 1. As each byte is transferred, it is simultaneously compared with the match byte (which may contain selectively masked bits, if so desired). When a match is obtained, the transfer is suspended and/or an interrupt generated, depending on how the DMA chip is programmed.

10-5.2 Modes of Operation

Under program control the DMA can be made to operate in one of the following four modes.

1. *Byte-at-a-time mode.* In this mode, the DMA operation is performed on 1 byte of the data block at a time. On completion of the DMA operation on a single byte, the control of system buses is returned to the CPU. For each of the succeeding DMA operations, a new request for the buses is made.
2. *Continuous mode.* In this mode once the DMA operation is started it continues the processing of the entire programmed data block. If the particular port's ready line goes inactive before the end of the block is reached, the DMA does not release the system buses, but just pauses until the ready line goes into the active state again.
3. *Burst mode.* In this mode the DMA operation continues as long as the port's ready line is in the active status. If it goes into the inactive status before the end of the programmed data block, the DMA operation stops and control of the system buses is returned to the CPU after completing the operation on the current byte.
4. *Transparent mode.* In this mode the DMA operation takes place during the time frames when the memory refresh operation takes place. Effectively, the DMA operations are interleaved with CPU operations, and consequently, the CPU does not lose any time because of the DMA operations.

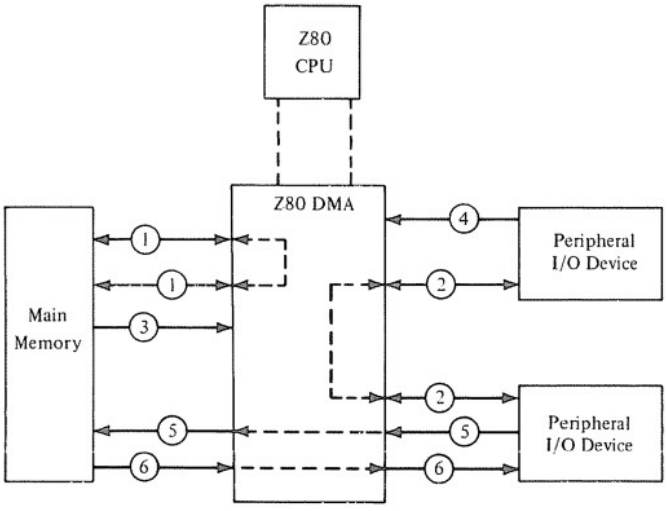


Figure 10-3 The Three Classes of Operations

5. *Block lengths.* In the Z80 DMA, high-speed buffering is used to read the data bytes from a port. The design of the data read system is such that the DMA operations on the first byte in are not completed until the following byte is read in. This has the following two implications. First, the data block on which a DMA operation is to be performed must be 2 or more bytes long. Second, the block length loaded into the 16-bit block length register (see Fig. 10-1) must be one less than the actual length of the block.

Note: In all the various modes, the DMA operation on the current byte (i.e., the byte that is read in) is always completed regardless of the status of the port's ready line or the status of other control signals involved in the operation.

10-5.3 The Interrupt System

The DMA chip is also used in the daisy chain interrupt system of the Z80. Under program control, the DMA can interrupt the CPU under any of the following conditions:

1. Generate an interrupt on ready, before requesting control of the busses.
2. Generate an interrupt when a match is obtained when operating in the search mode or combined transfer-search mode.
3. Generate an interrupt when the end of the data block is reached.

Note that because of the peculiarity in buffering the incoming data bytes (discussed in paragraph 5 of Sec. 10-5.2), an interrupt generated by a match at the end of the data block will be generated if a match is obtained on the byte just prior to the very last byte of the block. The interrupt then is generated just before the last byte in the block.

Another feature of the Z80 DMA interrupt system is that when the interrupt is generated, it sets an interrupt-pending bit which can, under program control, alter the interrupt vector assigned to that DMA chip.

10-5.4 Pulse Generation

In some applications it is desirable for the peripheral I/O device to keep track of how many bytes have been transferred in a DMA operation. In the Z80 DMA a pulse is output at every 256-byte transfer interval. This signal is output on the interrupt line but is not interpreted as an interrupt by the CPU because it appears when both the bus request ($\overline{\text{BUSRQ}}$) and the bus acknowledge ($\overline{\text{BAI}}$ or $\overline{\text{BAO}}$) lines are active. (See Table 10-5.)

10-5.5 Command and Control Bytes

The DMA chip contains several registers into which specific words can be loaded or written. Bytes can be loaded into these registers only when the DMA is not controlling the system buses. During the process of loading the write registers, the DMA is disabled and remains in that condition until it is enabled by means of a specific command. These registers fall into the following two categories:

1. *Initialization bytes.* These bytes are used for initially setting up the DMA chip. They include the following commands:
   - **a.** Class of operation.
   - **b.** Mode of operation.
   - **c.** Starting address of data block.
   - **d.** Length of data block.
   - **e.** The match byte.
   - **f.** The mask byte.
   - **g.** Interrupt vector.
   - **h.** Status conditions affecting interrupt vectors.
   - **i.** Pulse counting for bytes transferred.
   - **j.** Interrupt conditions.
   - **k.** Rules for ready line and wait line.
2. *Control bytes.* These bytes are loaded into write registers and they initiate immediate actions by the DMA chip. They include the following:
   - **a.** Load starting address buffers.
   - **b.** Clear counters.
   - **c.** Clear status bits.
   - **d.** Enable.
   - **e.** Disable.
   - **f.** Reset.

10-5.6 Reading from Internal Registers

1. The Z80 DMA has seven internal registers whose contents can be read out. The contents of these registers can be read out in the following sequence:

a. The 8-bit status register.
b. Block length register, low byte.
c. Block length register, high byte.
d. Port A address register, low byte.
e. Port A address register, high byte.
f. Port B address register, low byte.
g. Port B address register, high byte.

2. The sequential readout of these seven registers is accomplished by means of an internal pointer which points to each register successively after each read operation.
3. It is possible to skip the readout of any desired register by inserting a 0 in the appropriate bit position in the read byte.

10-5.7 Variable Cycle Length Time

To accommodate the different operating speeds of various memories and peripherals used in the DMA operations, the Z80 DMA chip has unique features which allow the user to vary the cycle time length. The cycle length can be varied from one to four T cycles. (Recall that a T cycle is defined as one clock period.) For situations where additional time is desired, additional $\overline{\text{WAIT}}$ states can be inserted. This feature has two principal advantages. First, it allows the designer to adapt the DMA operating speeds compatible with the speed of the various other system components. Second, it eliminates the need for additional external logic for conditioning the associated control signals. The various timing relationships are shown later in this chapter.

10-5.8 Port Addressing

For every DMA transfer, two 16-bit addresses are generated by the DMA chip; one address is for the source port and the other for the destination port. Each address can be either fixed or variable. The variable address scheme allows the port address to be sequentially incremented or decremented from a starting address. The length of the block of data to be transferred, expressed in number of bytes, is loaded into the block length register. Block lengths of up to 64 kilobytes can be used. During a DMA transfer, a port address is first generated for the source port and then another is created for the destination port. These addresses are time-multiplexed onto the system address bus.

10-5.9 Auto Restart

This feature automatically reloads the starting address for either of the two ports on completion of a block transfer. When the addresses are reloaded, the byte counter is cleared. This feature is activated by an auto restart control bit in one of the command words. This feature reduces the software burden on the CPU in applications which require repetitive operations. Also, if byte-at-a-time or the burst mode is selected for data transfers, the CPU has access to the system buses for certain periods of time. During such time periods, it is possible to write new or different block starting addresses in the DMA chip. Thus, an auto restart will start a block transfer or search operation at a new starting address.

10-6 THE DMA TIMING

10-6.1 The Variable Cycle Timing

Previously, in Sec. 10-5.7, we had briefly described the variable cycle timing of the Z80 DMA. In addition to allowing the user to vary the cycle length time, a unique feature allows the four control signals associated with each port (namely $\overline{\text{IORQ}}$, $\overline{\text{MREQ}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$) to have their trailing, positive-going edges terminated one half T cycle early if so desired.

In Fig. 10-4 the variable timing waveforms are shown as programmed for four T cycles. The earlier termination of the four control signals is shown in dotted lines for each signal. This feature gives the user an added dimension of flexibility. The shorter duration of the read and write control signals allows them to be terminated before the transferred data begin to change. Also shown in Fig. 10-4 are the points at which $\overline{\text{WAIT}}$ signals from either the memory or the I/O device will come in during the lengthened timing cycle.

10-6.2 DMA Inactive State Timing

10-6.2.1 Command/Control Byte Write Cycle Figure 10-5 shows the timing waveforms associated with the process of writing the command or control byte into one of the registers in the DMA chip. This is a CPU-to-DMA transfer of 1 byte which involves activation of three control signals, namely $\overline{\text{CE}}$, $\overline{\text{IORQ}}$, and $\overline{\text{WR}}$, which are activated simultaneously. Because of this and to simplify the timing waveforms, only one waveform is shown in Fig. 10-5 for all three control signals. The command or control byte is then output by the Z80 CPU approximately one T cycle later. Note that during this operation the DMA is in an inactive or disabled state and so the

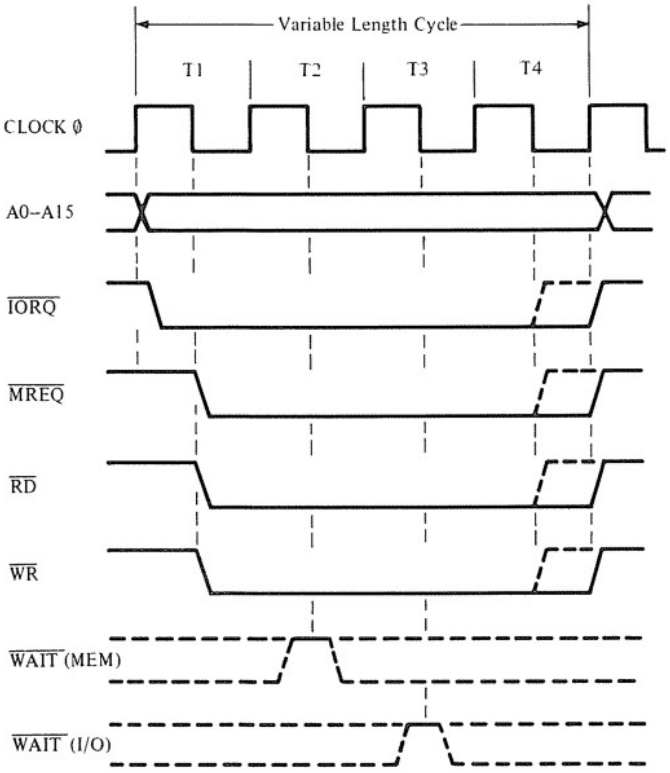


Figure 10-4 Variable Cycle Length Timing (Programmed for Four T Cycles)

CPU addresses it and writes into it as if it were an I/O device.

10-6.2.2 DMA Register Read Cycle The timing waveforms shown in Fig. 10-6 are for the operating DMA-to-CPU readouts from the DMA status register, the port address, and the byte counters. The three control signals involved in this operation are $\overline{CE}$, $\overline{IORQ}$, and $\overline{RD}$, and since they are activated simultaneously, they are shown as only one waveform in Fig. 10-6 for simplicity. The data appear on the bus approximately one T cycle after the control signals are activated.

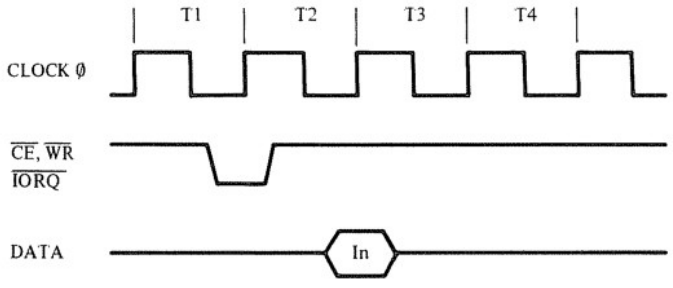


Figure 10-5 CPU-to-DMA Command Byte Write Timing

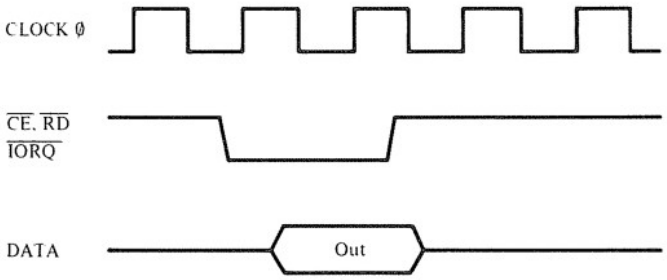


Figure 10-6 Readout Timing (DMA-to-CPU)

10-6.3 DMA Active State Timing

10-6.3.1 Memory-to-I/O Transfers The memory-to-I/O transfers involve reading data bytes from the memory port, transferring them internally to the other port, and then writing them into the I/O device or peripheral. After resetting the DMA chip, the read and write timing, in the DMA operation are identical to the CPU read and write timings for the memory and the I/O devices. There is one exception, clearly shown in Fig. 10-7, which is the timing waveform chart for the memory-to-I/O data transfers. During the memory read cycle, the data bytes are latched onto the D0–D7 bus during the negative-going trailing edge of T3 of this cycle. These bytes are then retained on the system data bus until the end of the I/O write cycle, i.e., until after completion of the following T3 cycle.

Notice that the memory read cycle consists of only three T cycles, whereas an extra wait state is inserted between T2 and T3 of the I/O write cycle. This is done primarily to reconcile the timing differences between faster memories and slower I/O devices. To further accommodate slower peripherals, the following procedure is used:

1. The $\overline{CE/WAIT}$ is programmed as a $\overline{WAIT}$ line when the DMA chip is operating in the active state.
2. The $\overline{CE/WAIT}$ line is sampled during the negative-going trailing edge of T2 in the memory read cycle and the negative-going trailing edge of TW in the I/O write cycle.
3. If the $\overline{CE/WAIT}$ line is low when it is sampled, it indicates that the I/O or the memory is not ready for a byte transfer.
4. The DMA then automatically inserts a TW cycle either after T2 of the memory read cycle, or after the TW cycle of the I/O write cycle, or after both, as appropriate.
5. The $\overline{CE/WAIT}$ line is then sampled again on the negative-going, trailing edge of the newly inserted TW cycle. If the $\overline{CE/WAIT}$ is still low, another TW cycle is inserted. The process is repeated until the I/O or the memory is ready for the transfer and indicates this by a high signal on the $\overline{CE/WAIT}$.

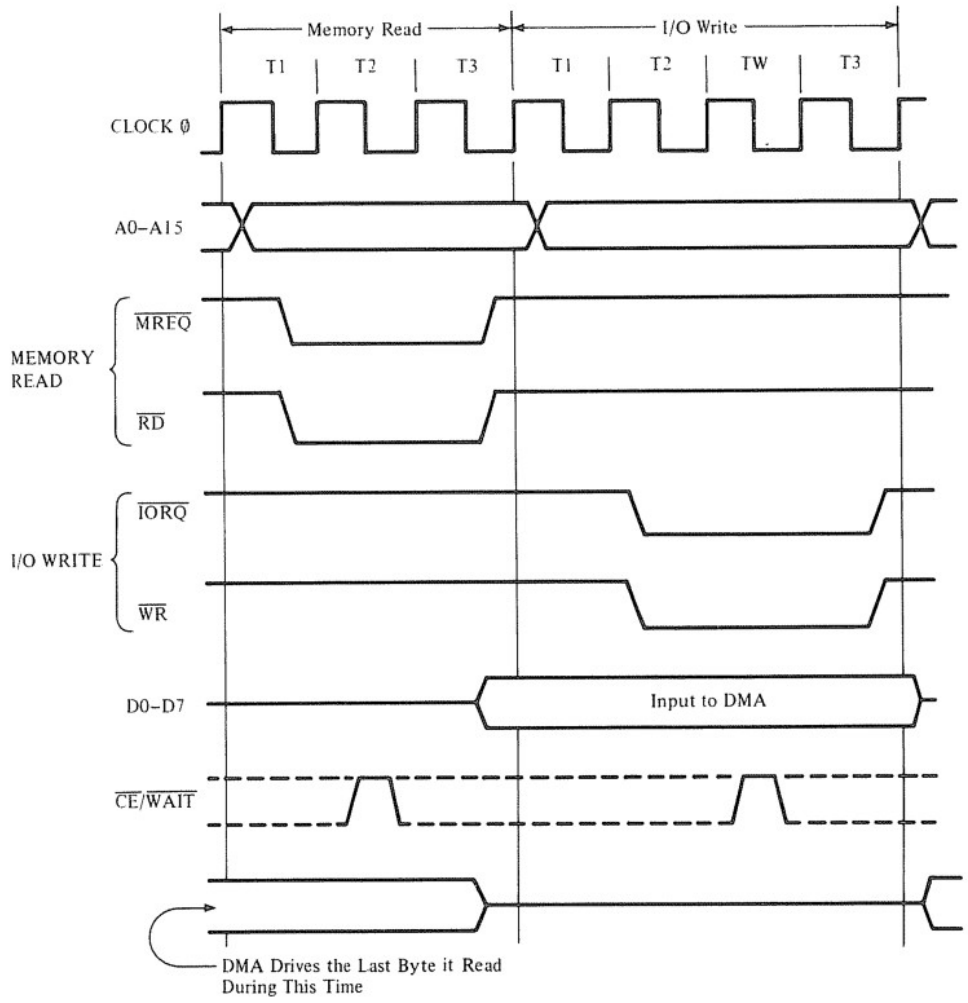


Figure 10-7 Memory-to-I/O Transfer Timing

10-6.3.2 Memory-to-Memory Transfers The timing involved in memory-to-memory DMA transfers is shown in Fig. 10-8. Notice that both the memory read and the memory write cycles each have three T cycles. Also, an active $\overline{\text{MREQ}}$ signal is involved in both the read and the write cycles. The $\overline{\text{RD}}$ and the $\overline{\text{WR}}$ signals identify the read and the write cycles, respectively. As in the previous case of memory-to-I/O DMA transfers, wait states can be inserted between the T2 and T3 of either read or write cycle, or both, if the $\overline{\text{CE/WAIT}}$ line is low during the middle of T2. If the $\overline{\text{CE/WAIT}}$ line were not high, as shown in Fig. 10-8, then a wait state would have been inserted between T2 and T3.

10-6.3.3 I/O-to-Memory Transfers Figure 10-9 shows the timing involved in the I/O-to-memory DMA transfers. Notice that a wait state is automatically inserted in the I/O read cycle but not in the memory write cycle. Of course additional wait states can be inserted in either cycle if the $\overline{\text{CE/WAIT}}$ is low at the appropriate time.

10-6.3.4 I/O-to-I/O Transfers The timing involved in I/O-to-I/O DMA transfers is shown in Fig. 10-10. In this case, both the I/O read and write cycles have one wait state automatically inserted between T2 and T3. Of course, additional wait states can be inserted in either or both cycles when the $\overline{\text{CE/WAIT}}$ line is low.

10-6.4 Bus Request and Acceptance Timing

The DMA bus request and acceptance timing for the byte-at-a-time, burst, and continuous modes are shown in

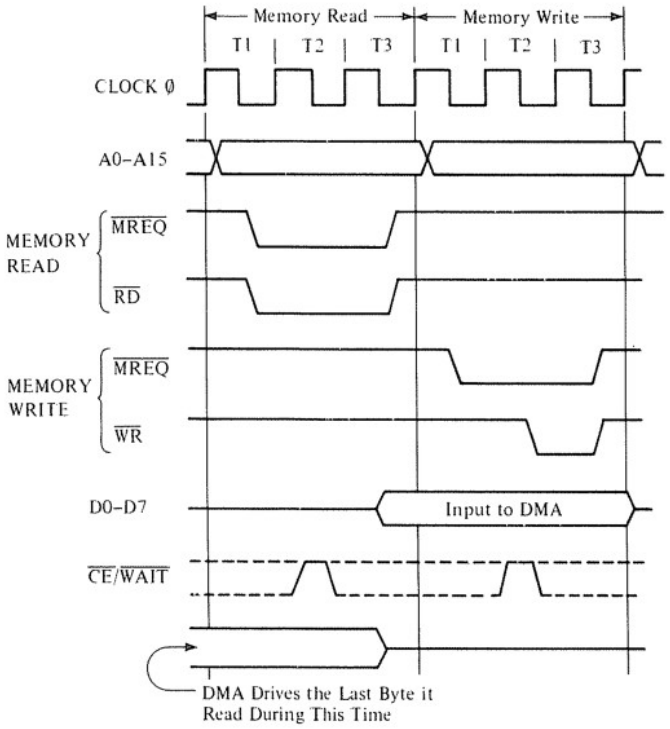


Figure 10-8 Memory-to-Memory Transfer Timing

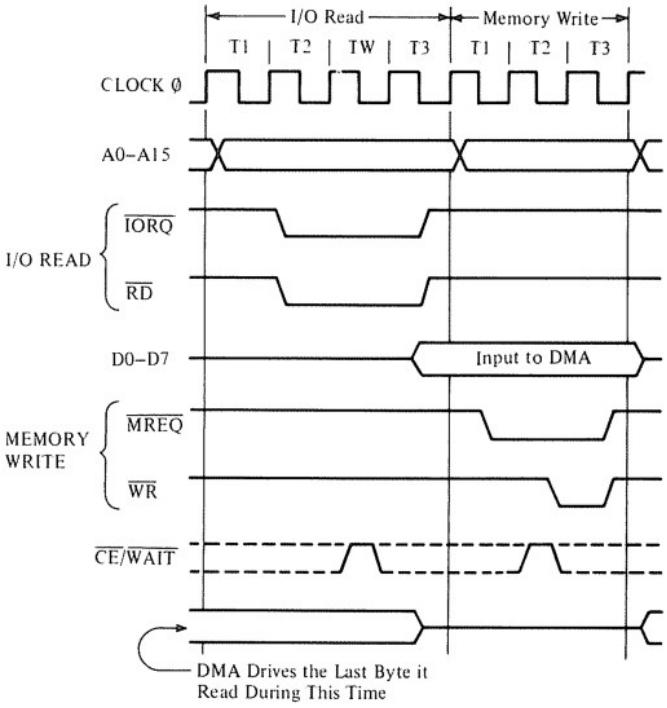


Figure 10-9 I/O-to-Memory Transfer Timing

Fig. 10-11. The DMA chip samples the ready input line (RDY), which can be programmed active high or low, on the negative-to-positive edge of every clock ø to determine if the peripheral associated with a DMA port is ready for a transfer. If $\overline{\text{RDY}}$ is active, a $\overline{\text{BUSRQ}}$ signal is generated and sent to the CPU on the rising edge of the next clock ø. The CPU then responds with a bus acknowledge signal to the $\overline{\text{BAI}}$ line to the DMA either directly or through the daisy chain. The $\overline{\text{BAI}}$ line is sampled on the leading edge of every clock ø. When a $\overline{\text{BAI}}$ active state is detected for two consecutive positive-going edges of the clock, the DMA goes into the active state on the next leading edge of ø, as in Fig. 10-11.

10-6.5 Bus Release Timing for Burst and Continuous Modes

10-6.5.1 At End of Block When operating in either the burst or the continuous mode, the $\overline{\text{BUSRQ}}$ line goes high when the end of the data block is reached. Usually, this happens on the same leading edge of the clock ø at which the DMA completes the transfer of the data block. The DMA then goes into an inactive or disabled state, as shown in Fig. 10-12. Notice that the transfer of the last byte in the data block is completed regardless of whether the RDY is active or inactive during this transfer.

10-6.5.2 On "Not Ready" When operating in the burst mode, if the RDY signal becomes inactive, the DMA completes its current operation on the in-process byte and then causes the $\overline{\text{BUSRQ}}$ line to go high (i.e., inactive) on the next leading edge of the clock ø. This is shown in Fig. 10-13.

When operating in the continuous mode, the $\overline{\text{BUSRQ}}$ line is held in the active state or low when the RDY line becomes inactive. In this case, the DMA just idles after completing the current operation until such time as RDY again is in the active state.

10-6.5.3 On "Match" When operating in either the burst or the continuous mode, the DMA can be programmed to stop when a match is obtained by comparing the data byte with the match byte. When a match is found, the $\overline{\text{BUSRQ}}$ goes high or inactive, but only after completing the DMA operation on the next byte, which could be either a read in a search process or a write in a transfer process. Recall that because of the pipelining process (i.e., double buffering) used in the DMA chip, matches are performed while the next DMA read or write operation is being executed. This is shown in Fig. 10-14. Note that the RDY line can become inactive any time after the match operation without affecting bus release.

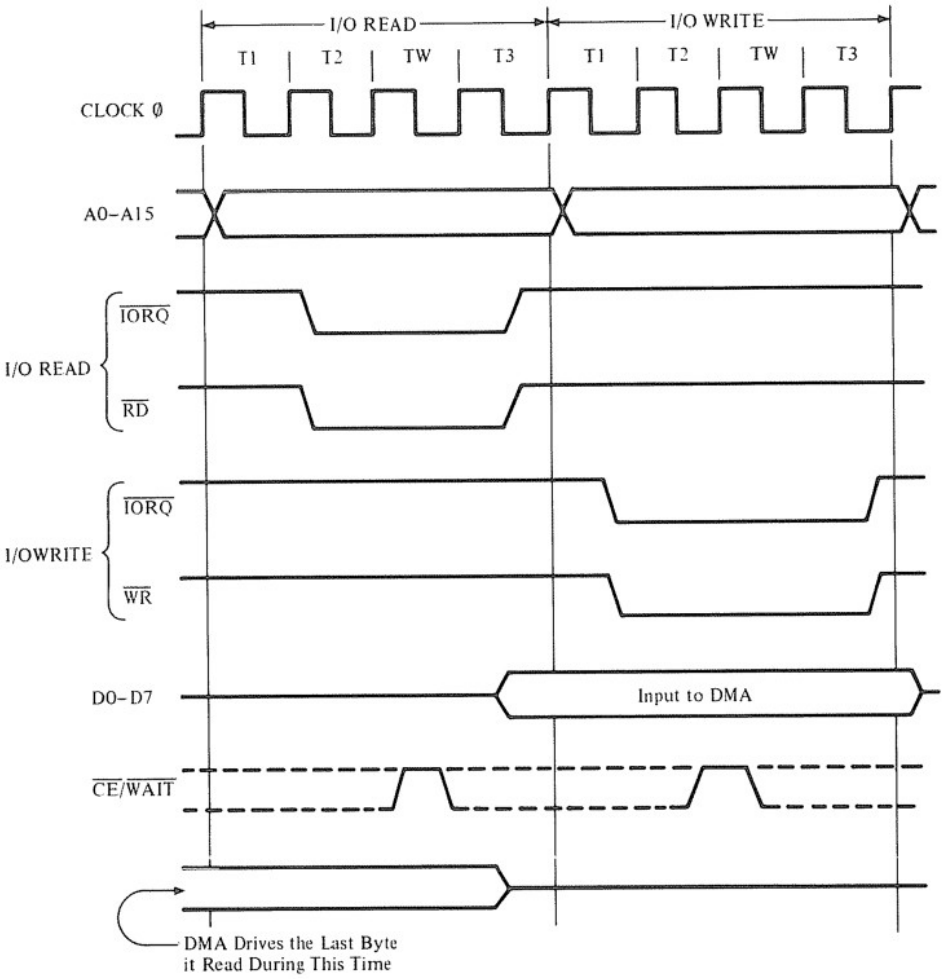


Figure 10-10 I/O-to-I/O Transfer Timing

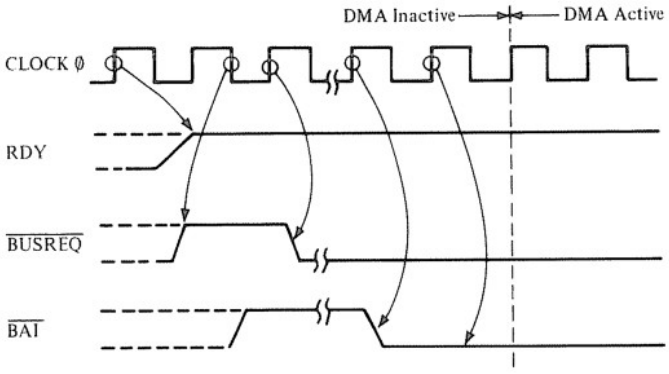


Figure 10-11 Bus Request and Acceptance for All Modes

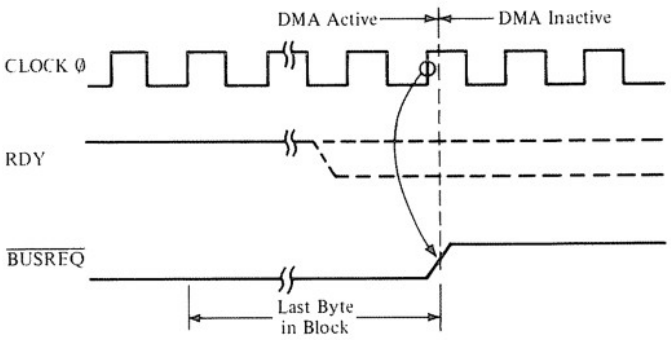


Figure 10-12 Bus Release at End-of-Block Timing for Burst and Continuous Modes

10-6.6 Bus Release Timing for Byte-at-a-Time Mode

When operating in the byte-at-a-time mode, the $\overline{\text{BUSRQ}}$ signal goes high on the leading edge of the clock ø prior to the end of each cycle if operating in the search-only class. If the DMA is operated in the transfer or the combined search-transfer class, the $\overline{\text{BUSRQ}}$ line goes high on the leading edge of the clock ø prior to the end of each write

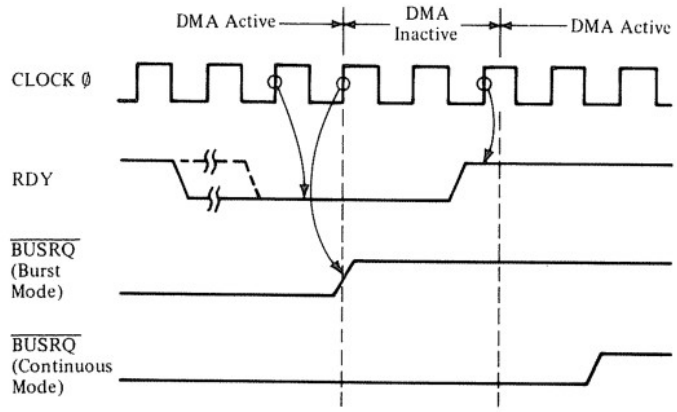


Figure 10-13 Bus Release Timing when "Not Ready" for Burst and Continous Mode

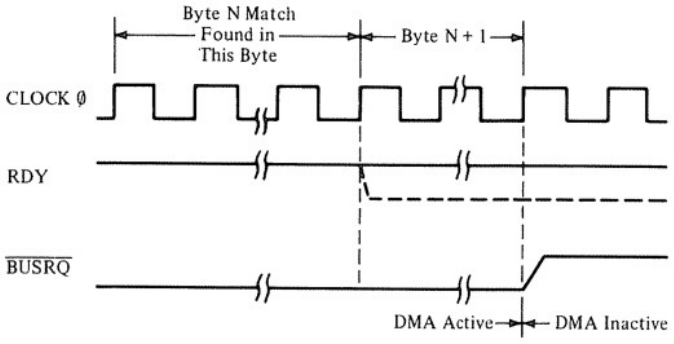


Figure 10-14 Bus Release Timing on "Match" for Burst and Continuous Modes

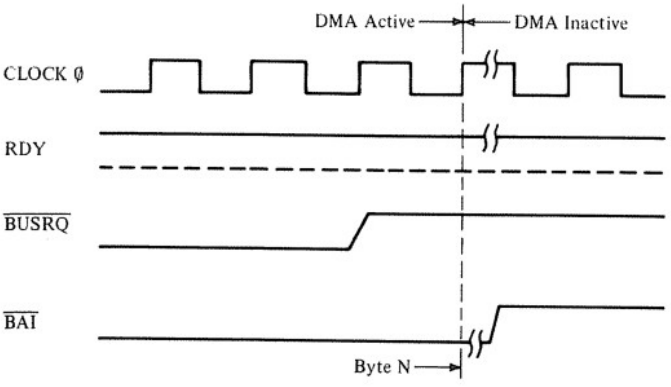


Figure 10-15 Bus Release Timing in Byte-at-a-Time Mode

cycle. This is shown in Fig. 10-15. Note that $\overline{\text{BUSRQ}}$ goes high regardless of the status of the RDY line.

10-7 REVIEW QUESTIONS

10-1 The following statements apply to the Z80 DMA chip. Indicate true or false.

a. ___ Blocks of data can be transferred between two ports independent of the CPU.

b. ___ Transfer of data blocks can take place only between an I/O device and the main memory.

c. ___ A block of data can be searched to match a specific match byte.

d. ___ During a search operation it is possible to transfer a block of data simultaneously.

e. ___ It is not possible to use the DMA chip in the priority daisy chain of the Z-80 system.

10-2 The following statements relate to the starting address of the DMA port(s). Circle the correct ones.

a. The starting addresses can be automatically incremented but not automatically decremented.

b. The starting address can remain fixed, if so desired.

c. The starting addresses can be automatically decremented but not automatically incremented.

d. The starting addresses can be automatically incremented as well as automatically decremented.

10-3 Fill in the blanks.

a. The address registers are _____ buffered. (not; double; triple; quadruple)

b. The block length register is _____ buffered. (not; double; triple)

10-4 Interrupts can be generated on (circle the correct answers)

a. Ready.

b. Match not found.

c. End-of-block.

d. Match found.

e. Address found.

10-5 Answer yes or no to the following questions.

a. ___ Can the DMA signal the CPU when a previously specified number of bytes has been transferred?

b. ___ Does the transfer operation stop when the DMA signals the CPU that a previously specified number of bytes has been transferred?

c. ___ Is it possible for the CPU to read out the status of any DMA channel on request?

10-6 From the statements below select the one(s) that will make the following statement true.

"Under program control the DMA channel can be _____"

a. enabled but not disabled or reset.
b. enabled, disabled, and reset.
c. reset and enabled but not disabled.
d. enabled and disabled but not reset.

10-7 Indicate true or false for the rates of DMA operations.

a. ____ Can be operated up to rates of 2.2 megabytes per second for the transfer operations only.
b. ____ Can be operated up to 1.25 megabytes per second for only the search operation.
c. ____ Can be operated up to 2.2 megabytes per second for both search and transfer operations.
d. ____ Can be operated up to 1.25 megabytes for both search and transfer operations.

10-8 Under which of the following conditions can the entire DMA operation be made to repeat itself? Circle all that apply.

a. On command (load).
b. When the port address is fixed.
c. Automatically (auto restart).
d. On end-of-block.

10-9 The following statements apply to the byte-at-a-time mode of operation. Indicate true or false.

a. ____ Only 1 byte in a data block can be transferred at any time when requested, under program control.
b. ____ The DMA retains control of the system buses after the byte transfer is completed.
c. ____ A new DMA request is made after each byte is transferred.
d. ____ Only a transfer operation can be performed in the byte-at-a-time mode.

10-10 When operating in the continuous mode, which of the following apply if the port is not ready for the operation before end-of-block is reached? (Circle all that apply.)

a. The DMA operation continues until the end-of-block is reached.
b. The DMA operation is terminated.
c. The DMA operation pauses until the port is ready.
d. When the port is ready it indicates this by reactivating the ready line.

10-11 Refer to the generalized block diagram of the DMA, Fig. 10-1, and indicate true or false for the port-addressing scheme.

a. ____ For data transfer operations an 8-bit starting address is involved for ports A and B.
b. ____ The upper byte is used for addressing an I/O port.
c. ____ The single-byte address for the I/O ports is usually a fixed address.
d. ____ A 2-byte starting address is involved for both ports in memory-to-memory transfers.

10-12 Refer to Fig. 10-1, the generalized block diagram of the DMA. From the following items pick out those whose condition is indicated by the 8-bit status register.

a. Interrupt pending.
b. Write address bits valid.
c. End-of-block.
d. Byte match found.
e. Ready active.

10-13 The bus control and the bus priority logic in Fig. 10-1 involve three request signals for control of certain Z80 system facilities. Name these facilities.

a. _____
b. _____
c. _____

10-14 The interrupt control and priority logic block in Fig. 10-1 involve three control lines. Name the signals involved in these three lines.

a. _____
b. _____
c. _____

10-15 The following statements relate to the 8-bit interrupt vector register of Fig. 10-1. Indicate true or false.

a. ____ It holds the vector assigned to that particular DMA chip.
b. ____ The vector is preloaded during initialization of the chip.
c. ____ The contents of this register are output onto the data bus during the interrupt request cycle.
d. ____ The vector is sent to the CPU only if the interrupting DMA has the highest priority in the daisy chain.
e. ____ The CPU utilizes the vector for calling up the pertinent service subroutine.

10-16 Refer to Fig. 10-1 and indicate the logic block(s) in which the specific control bytes for performing the various DMA operations are loaded during chip initialization.

a. Status register.
b. Match word register.

c. Control logic.
d. Byte counter.
e. Port start address register.

10-17 Refer to the byte counter in Fig. 10-1 and indicate true or false.
a. ___ It is an 8-bit counter.
b. ___ It is cleared at the start of the DMA operation.
c. ___ It can be either incremented or decremented.
d. ___ When the DMA processing of each byte is completed, the byte counter is incremented.
e. ___ The count in the byte counter is compared with the contents of the match word register.
f. ___ The end-of-block flag is set in the status register only when a match is found between the contents of the block length register and the byte counter.

10-18 Fill in the blanks in the following statements which refer to the match word register in Fig. 10-1.
a. It contains a word for which a match is to be found during a _____ operation. (transfer; search)
b. Contents of the match word register are compared with the byte from the _____. (data block; address register; vector interrupt register)
c. When a match is found, the _____ flag in the status register is set. (interrupt-pending; end-of-block; ready active; byte-match)

10-19 Refer to the pulse interval and pulse generation logic block in Fig. 10-1 and indicate true or false.
a. ___ The 16-bit pulse control register is loaded during DMA chip initialization.
b. ___ The length of the data block, expressed in number of bytes, is loaded into the pulse control register.
c ___ The DMA emits a signal only when the entire block is processed.
d. ___ The signal emitted by the DMA is interpreted as an interrupt signal by the CPU.
e. ___ The signal emitted by the DMA is used to communicate with a peripheral device.

10-20 The following statements apply to the mask word register in Fig. 10-1. Fill in the blanks.
a. This register is loaded during _____. (initialization; start of the data block; end of the data block)
b. The word in the register masks out certain bits loaded in the _____ register. (block length; byte counter; start address; interrupt vector; match word; pulse control)

10-21 Which of the following signals requests the CPU to release the system buses?
a. $\overline{\text{BAI}}$
b. $\overline{\text{BUSRQ}}$
c. $\overline{\text{BAO}}$
d. $\overline{\text{M1}}$

10-22 After the DMA has obtained control of the system busses, it outputs a signal on the _____ pin. ($\overline{\text{MREQ}}$; $\overline{\text{RD}}$; $\overline{\text{IORQ}}$; $\overline{\text{WR}}$; $\overline{\text{M1}}$)

10-23 Which of the following signals can be used to slow down the DMA if the I/O devices and/or memory are slower than the DMA chip?
a. $\overline{\text{RDY}}$
b. $\overline{\text{CE/WAIT}}$
c. $\overline{\text{MREQ}}$
d. $\overline{\text{IORQ}}$
e. $\overline{\text{BUSRQ}}$

10-24 The following statements apply to the search operations. Indicate true or false.
a. ___ With 2.5-MHz Z80 DMA, search rates of up to 2.0 M bytes per second are possible.
b. ___ With 4.0-MHz Z80 DMA, search rates of up to 2.0 M bytes per second are possible.
c. ___ With 2.5 MHz Z80 DMA, search rates of up to 4.0 M bytes per second are possible.
d. ___ With 4.0-MHz Z80 DMA, search rates of up to 4.0 M bytes per second are possible.

10-25 Which of the following transfers are possible when the DMA is programmed in the transfer class of operations?
a. CPU-to-memory transfers.
b. CPU-to-I/O transfers.
c. Memory-to-I/O transfers.
d. I/O-to-CPU transfers.
e. Memory-to-memory transfers.

10-26 The following statements apply to the auto restart feature of the Z80 DMA. Indicate true or false.
a. ___ The byte counter is not affected when the addresses are loaded.
b. ___ The starting addresses for either port are reloaded when an interrupt comes along.
c. ___ A control bit in one of the command words activates auto restart.
d. ___ An auto restart can start a block transfer or search operation at a new starting address.

10-27 Refer to Fig. 10-4, variable cycle length timing. Which of the following control signals can be terminated one half T cycle early?

a. $\overline{\text{BUSRQ}}$
b. $\overline{\text{MREQ}}$
c. $\overline{\text{INT}}$
d. $\overline{\text{CE/WAIT}}$
e. $\overline{\text{IORQ}}$
f. $\overline{\text{BAI}}$
g. $\overline{\text{BAO}}$

10-28 Which of the following signals must be simultaneously active for readout of the DMA status register?
a. $\overline{\text{WR}}$
b. $\overline{\text{BAO}}$
c. $\overline{\text{IORQ}}$
d. $\overline{\text{RD}}$
e. $\overline{\text{CE}}$
f. $\overline{\text{BAI}}$

10-29 Refer to the memory-to-I/O transfer timing of Fig. 10-7. Answer the following questions. (Yes or no)
a. ____ Is it possible to include a TW state in the memory read cycle?
b. ____ Is it possible to include a second TW state in the I/O write cycle?
c. ____ Is it possible to insert a TW state between T1 and T2 of the I/O write cycle?
d. ____ Is the TW state in the I/O write cycle inserted automatically?

10-30 For proper control of the I/O-to-I/O transfers (Fig. 10-10). Certain control signals must be simultaneously active.
a. _____ Which pair must be simultaneously active for the I/O write operations? ($\overline{\text{RD}}$; $\overline{\text{WR}}$; $\overline{\text{BAI}}$; $\overline{\text{MREQ}}$; $\overline{\text{IORQ}}$; $\overline{\text{CE/WAIT}}$)
b. _____ Which signals must be simultaneously active for the I/O read operation? ($\overline{\text{RD}}$; $\overline{\text{WR}}$; $\overline{\text{BAI}}$; $\overline{\text{MREQ}}$; $\overline{\text{IORQ}}$; $\overline{\text{CE/WAIT}}$)

10-31 For all three modes of operation, for the DMA to be active it must have control of the system buses. Which signals must be simultaneously active for bus acceptance and release?
a. $\overline{\text{BAI}}$, $\overline{\text{RD}}$, and $\overline{\text{IORQ}}$
b. RDY, $\overline{\text{IORQ}}$, and $\overline{\text{BUSRQ}}$
c. $\overline{\text{BAI}}$, RDY, and $\overline{\text{BUSRQ}}$
d. $\overline{\text{BAO}}$, $\overline{\text{BUSRQ}}$, $\overline{\text{IORQ}}$, and RDY

11
PROGRAMMING THE DMA CHIP

CHAPTER OBJECTIVES

The objectives of this chapter are as follows:

1. To introduce the concept of customizing the DMA chip for specific applications involving transfer, search, and combined transfer-search operations.
2. To present the command bytes for the basic setup of DMA port A, together with the four associated registers.
3. To present the variable timing byte for port A.
4. To describe the basic setup for port B together with the variable timing byte.
5. To examine the format of the command byte for the search class of operations, including the match and mask bytes.
6. To present the command byte for the pulse generation, interrupt generation and the interrupt vector features of the DMA, along with its five associated registers.
7. To present the command byte for miscellaneous signals associated with control of the DMA operation.
8. To describe the command byte for several other functions such as resetting, enabling, and disabling several features of the DMA chip.
9. To present the features of the read mask register and the status register.

TEXTBOOK REFERENCES

For reviewing material in the textbook relevant to the topics in this chapter, the following chapter(s) and/or sections are suggested:

1. For general discussion of DMA	Chapter 10
2. For transparent mode	Sec. 10-2.1
3. For cycle stealing mode	Sec. 10-2.2
4. For byte-at-a-time DMA transfers.	
a. Using single memory	Sec. 10-3.1
b. Using multiple memories	Sec. 10-3.2
5. For continuous mode single-block transfers	Sec. 10-4
6. For transfer of multiple data blocks	Sec. 10-5
7. For CPU response to DMA request	Sec. 10-6.2
8. For operation of the DMA interface chip	Sec. 10-7
9. For byte and block count	Sec. 10-7.3
10. For byte count update	Sec. 10-7.7
11. For block count update	Sec. 10-7.8

11-1 INTRODUCTION

The Z80 DMA is a programmable interface chip which, under program control, can be initialized to perform

functions for specific applications within certain limits. Customizing the DMA requires that the chip be first set up by writing programming or command bytes into some or all of the seven groups of write registers. The DMA chip has two ports and either a memory or an I/O device can be connected to either of the two ports. Transfers in either direction are possible between the two ports. It is also possible to search a data block for a match with a preestablished match byte whose bits can be selectively masked out, if so desired. In this chapter we describe the initialization process and the command bytes involved in a set-up program. Also, examples are included that show how the DMA is initialized to perform transfer, search, and combined transfer-search operations.

11-2 COMMAND WORDS FOR DMA SET-UP

In this section we discuss the formats of the various command words which are required to initialize the DMA chip. The words or bytes define the different classes and modes of the DMA operation. The DMA chip has seven groups of write registers. Each group consists of a base register. Some of these groups have one or more associated write registers. The seven groups are labeled WR0–WR6 and there is a total of 14 associated registers. The procedure of writing into any of the register groups which has one or more associated register(s) consists of the following steps.

1. The desired word is first written into the base register.
2. Appropriate pointer bits are included in the byte written into the base register.
3. The pointer then sets up the address for each of the sequentially associated registers and the bytes are then successively written into them.

Each of the seven write register groups and their associated registers are briefly described next. Remember that the DMA chip has two fundamental states which can be programmed. In the *enabled* state, the DMA obtains control of the system buses and thereby controls the transfer of data between the addressed ports. In the *disabled* state, no requests for buses are made and so no data are transferred between ports. When power is first applied to the DMA chip it automatically goes into the disabled state. It also goes into the disabled state when the chip is reset by any means. It is possible to write command bytes into the write registers, in either of the two states, but the writing process immediately puts the chip into the disabled state. It remains in that state until an enable command is sent by the CPU.

11-2.1 The WR0 Group

The WR0 group, which contains the basic set-up information for the DMA chip consists of one base register WR0-0 and four associated registers labeled WR0-1 through WR0-4. Figure 11-1 shows these registers and what is contained in each. The arrows show the bit in the base register which identifies or affects the associated registers.

Bits D0 and D1 in the base register identify the class of operation as in the following codes:

C_1	C_2	*Operation Class*
0	0	Not used
0	1	Data block transfer only
1	0	Search only
1	1	Combined search/transfer

Bit position D2 in the base register identifies which port is the source port. Selecting one port automatically makes the other port the destination port for transfer operations. If the search only mode is selected, then D2 identifies only the source port. No destination port is addressed or selected.

11-2.2 The WR1 Group

The WR1 group also contains the basic setup information for port A of the DMA chip. It consists of the base

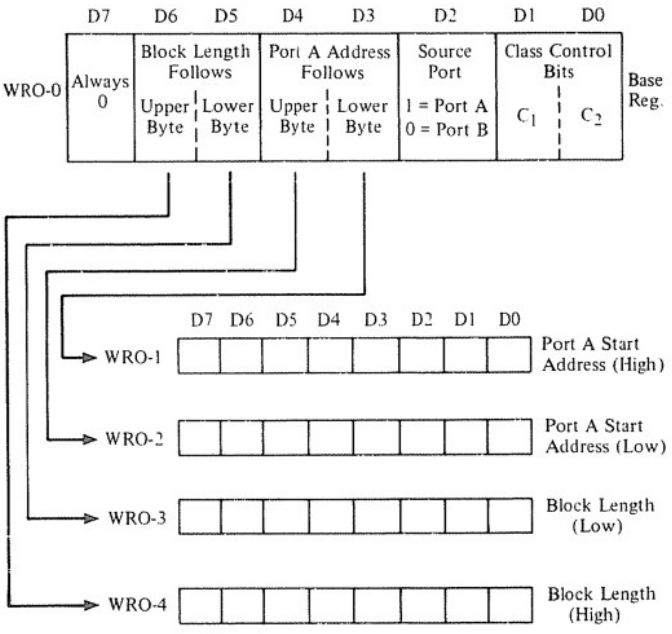


Figure 11-1 Write Registers Group WR0

register WR1-0 and one associated register WR1-1, as shown in Fig. 11-2.

For the Base Register WR1-0

1. A 1 in bit D5 indicates a fixed address for port A, and so the bit in D4 has no effect on this information. It can either be a 0 or a 1.
2. A 0 in bit D5 indicates that the port A address can either be incremented or decremented after each byte transfer, as indicated by the bit in D4 position.

For the Variable Timing Register

1. Bits D4 and D5 are always 0.
2. The variable cycle length time is indicated by bits D0 and D1 as follows:

D1	*D0*	*No. of T Cycles*
0	0	4
0	1	3
1	0	2
1	1	Not used

3. A 0 in the remaining bit positions indicates the following timing changes. (Note that these timing variations are explained in Chapter 10.)

0 in Bit Position	*Ends One Half Cycle Early*
D2	$\overline{\text{IORQ}}$
D3	$\overline{\text{MREQ}}$
D6	$\overline{\text{RD}}$
D7	$\overline{\text{WR}}$

11-2.3 The WR2 Group

The WR2 group contains the basic setup information for port B of the DMA chip. It consists of the base register WR2-0 and one associated register WR2-1, as shown in Fig. 11-3. The register formats are identical to that for port A, as shown in Sec. 11-2.2.

For the Base Register WR2-0

1. A 1 in the bit D5 position indicates a fixed address for port B; consequently, the bit in D4 has no effect on the operation. It can be either a 0 or a 1.
2. A 0 in bit D5 position indicates that the port B address can either be incremented or decremented after each byte transfer, as indicated by the bit in D4 position.

For the Variable Timing Register WR2-1

1. Bits D4 and D5 are always 0.
2. The variable cycle length time is indicated by bits D0 and D1 as follows:

D1	*D0*	*No. of T Cycles*
0	0	4
0	1	3
1	0	2
1	1	Not used

3. A 0 in the remaining bit positions indicates the following timing changes. (Note that these timing variations are explained in Chapter 10.)

0 in Bit Positions	*End One Half Cycle Early*
D2	$\overline{\text{IORQ}}$
D3	$\overline{\text{MREQ}}$
D6	$\overline{\text{RD}}$
D7	$\overline{\text{WR}}$

11-2.4 The WR3 Group

The WR3 group contains information for the search class of operations. It consists of the base register WR3-0 and two associated registers WR3-1 and WR3-2 for the mask

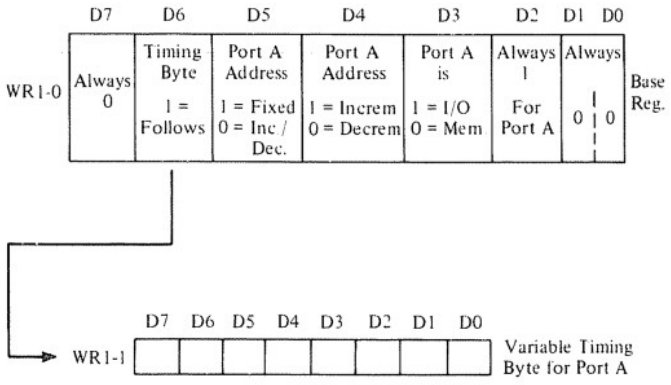


Figure 11-2 Write Registers Group WR1

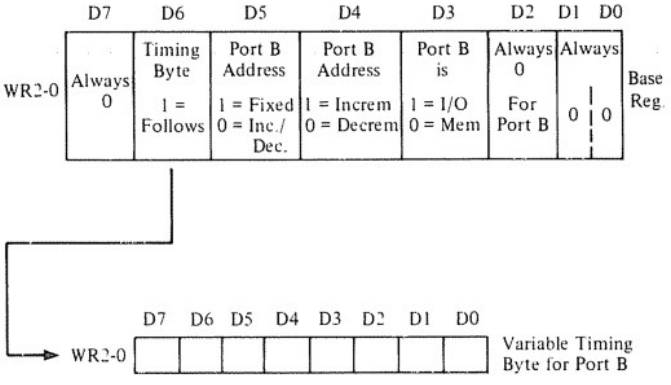


Figure 11-3 Write Registers Group WR2

and the match bytes, respectively. The register formats are shown in Fig. 11-4.

In the mask byte, WR3-1, a 0 in any bit position permits the bit in the corresponding bit position in the match byte (WR3-2) to be compared. A 1 in any bit position of WR3-1 masks out the corresponding bit in WR3-2.

A 1 in D6 of WR3-0 enables the DMA chip to request the bus. This bit duplicates the function 87 hex in WR6-0. It is used as the last control byte written into the DMA chip before the DMA gets control of the bus from the CPU.

11-2.5 The WR4 Group

The WR4 group contains one base register, WR4-0, and five associated registers, WR4-1 through WR4-5. This group of commands is concerned with the pulse generation, interrupt generation, and the interrupt vector features of the Z80 DMA. The register formats are shown in Fig. 11-5.

For the Base Register WR4-0

1. The base register provides pointers in positions D2 and D3 which indicate that the following 2 bytes are the block starting address for port B. D2 points to the low byte (WR4-1) and D3 to the high point (WR4-1). This is done by means of a 1 in D2 and D3.
2. The mode of operation is defined by bits in positions D5 and D6 as follows:

M1	*M0*	*Mode of Operation*
0	0	Byte-at-a-time
0	1	Continuous
1	0	Burst
1	1	Transparent

For the Interrupt Control Byte WR4-3

1. Bit D7 is always a 0.
2. A 1 in the following bit positions defines the conditions under which an interrupt is generated:

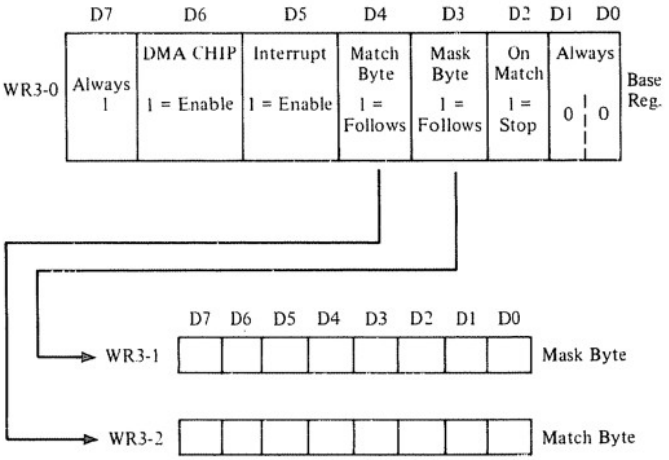


Figure 11-4 Write Registers Group WR3

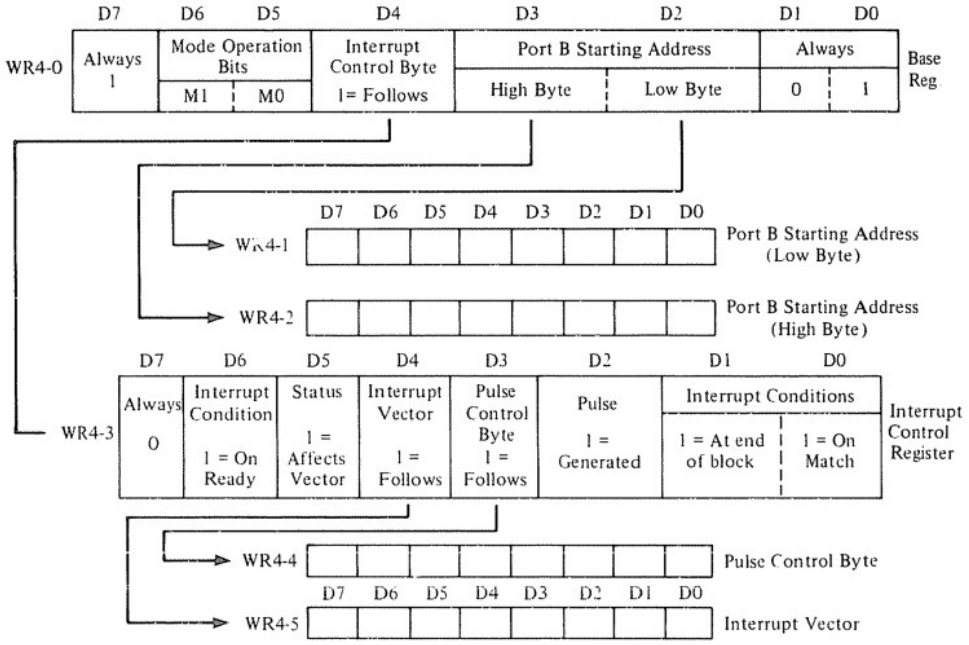


Figure 11-5 Write Registers Group WR4

Bit Positions	*Interrupt-Generating Condition*
D0	On match
D1	At the end of the data block
D6	On ready

3. A 1 in D2 generates a pulse on the $\overline{INT}$ terminal line when the entire data block is processed. The length of the block, in bytes, is loaded in the pulse control byte register WR4-4.
4. A 1 in D3 indicates that a pulse control byte follows.
5. A 1 in D4 indicates that the interrupt vector follows and is to be loaded in register WR4-5.
6. A 1 in D5 indicates that the status register's contents will determine and modify the interrupt vector byte (WR4-5), as shown below. (Note that the format of the status byte is described later in Sec. 11-2.8.)

For the Pulse Control Byte WR4-4

1. An 8-bit control word is loaded into WR4-4. This is usually the desired number of bytes in the data block.
2. On completion of each DMA byte operation, either a transfer or a search, the contents of the register are compared with the contents of the low-order byte of the byte counter.
3. When the 2 bytes compare (i.e., they are the same), a pulse is generated on the $\overline{INT}$ line. However, a legitimate interrupt is not generated.

For the Interrupt Vector Byte WR4-5

1. The contents of this register identify the particular DMA chip in the Z80 system.
2. The byte in WR4-5 is transferred to the CPU during the interrupt acknowledge, provided the interrupting DMA is the highest-priority device on the daisy chain.
3. When bit D5 of the interrupt control byte (WR4-3) is a 1 and the DMA chip is programmed to generate an interrupt when a specific status condition exists, the vector in WR4-5 will be changed, as shown below by modifying bits D1 and D2.

D2	*D1*	*Status Condition*
0	0	Interrupt on ready---RDY
0	1	Interrupt on match
1	0	Interrupt on end-of-block
1	1	Interrupt on match and end-of-block

11-2.6 The WR5 Group

The WR5 group contains only the base register WR5-0 and has no associated registers. It contains miscellaneous signals associated with control of the DMA operation. The format of the byte is shown in Fig. 11-6 and the functions performed by each bit are presented below.

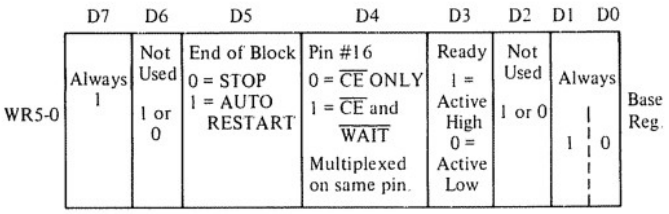


Figure 11-6 Write Register Group WR5

1. Bit positions D2 and D6 are not used, so they can be either a 0 or a 1.
2. Bit position D3 indicates whether the input on the RDY pin (pin 25) is to be intepreted as active when it is high or low.
3. Bit position D4 specifies whether the CE/WAIT input pin (pin 16) is to be interpreted as a chip enable or as a wait during the time when the bus acknowledge signal input is active.
4. Bit position D5 indicates the action to be taken when the end of block is reached. A 0 in D5 stops the DMA operation and a 1 automatically repeats the entire operation.

11-2.7 The WR6 Group

This group contains the base register WR6-0 and one associated register, WR6-1. This group performs a number of miscellaneous functions including resetting, as well as enabling and disabling various features. The formats of the two registers are shown in Fig. 11-7.

For the Base Register WR6-0

1. With the exception of bit positions D0, D1, and D7 which always contains a 1, positions D2–D6 are coded F0–F4. They contain the applicable binary codes.
2. The F0–F4 codes and the functions they perform are given in Table 11-1. For programming convenience the corresponding hex codes are also included.

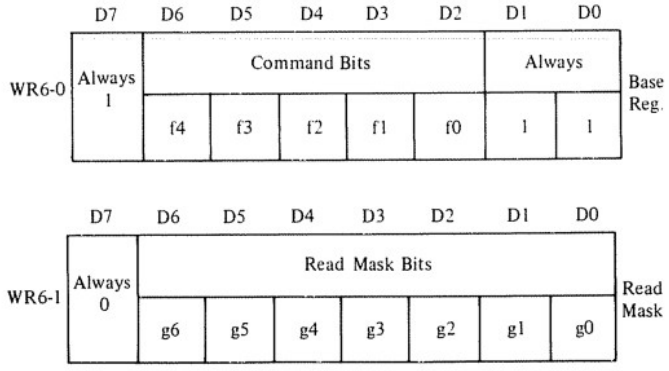


Figure 11-7 Write Register Group WR6

Table 11-1 COMMAND CODES FOR REGISTER WR6-0

Hex Code	Command Bits f4	f3	f2	f1	f0	Command Description
C3	1	0	0	0	0	Resets interrupt circuitry, disables interrupt and bus request logic, removes internal ready condition, disables MUXCE, and stops auto repeat.
C7	1	0	0	0	1	Resets port A timing to standard Z-80 timing.
CB	1	0	0	1	0	Resets port B timing to standard Z-80 timing.
CF	1	0	0	1	1	Loads starting address for ports A and B and clears the byte counter.
D3	1	0	1	0	0	Continues address from present location and clears the byte counter.
AB	0	1	0	1	0	Enables interrupts.
AF	0	1	0	1	1	Disables interrupts.
A3	0	1	0	0	0	Resets and disables interrupt circuitry like (RETI) and removes/delete the internal ready condition.
87	0	0	0	0	1	Enables DMA.*
83	0	0	0	0	0	Disables DMA.†
A7	0	1	0	0	1	Initiates read sequence to the first register designated as readable by the read mask register WR6-1.
BF	0	1	1	1	1	Sets read status so that the next readout is from the status register.
B3	0	1	1	0	0	Forces an internal ready condition regardless of the input on the RDY pin.†
8B	0	0	0	1	0	Clears match and end-of-block status bits.
B7	0	1	1	0	1	Enable after RETI instruction so that the DMA requests bus only after it receives a RETI. An enable DMA command must follow this code.
BB	0	1	1	1	0	A read mask byte follows.

*Both codes 87 and 83, enable and disable DMA, respectively, affect all operations except interrupts. They do not reset any functions.
†This command (B3) is used for memory-to-memory operations where a RDY signal is not required. This command does not function in a byte-at-a-time mode.

For the Read Mask Register WR6-1

1. If the BB code is used in the base register WR6-0, then WR6-1 must follow.
2. The bit in the D7 position is always a 0.
3. A 1 in bit position D0–D6 enables the readout of the various register bytes, as follows:

D0 Status register (RR0).
D1 Byte counter, low byte (RR1).
D2 Byte counter, high byte (RR2).
D3 Port A address, low byte (RR3).
D4 Port A address, high byte (RR4).
D5 Port B address, low byte (RR5).
D6 Port B address, high byte (RR6).

11-2.8 The Status Register

The seven registers, identified as RR0–RR6, are the read registers. When the bit position D0 of the mask register WR6-1 contains a 1, the contents of the status register are read out. The format of the status register is shown below in Fig. 11-8. Note that the status bits are active low.

11-3 PROGRAMMING THE DMA FOR TRANSFER-ONLY OPERATIONS

In this section we show, by means of typical examples, how the DMA chip can be programmed for the transfer-only class of operations, which allows us to transfer blocks of data between the source and the destination ports. (See Sec. 10-5.1 for review of this class of operation.)

D7	D6	D5	D4	D3	D2	D1	D0
Not Used		End-Of-Block	Match Found	Interrupt Pending	Not Used	Ready Active	DMA Operation Has Occurred
0 or 1	0 or 1	0	0	0	0 or 1	0	1

Figure 11-8 Status Register

11-3.1 I/O-to-Memory Transfer

Example 11-1

A peripheral device whose starting address is 36_H is connected to Port A and a memory is connected to port B. A block of data whose length is 0435_H bytes is to be transferred from the peripheral to the memory at starting address 1820_H using the Continuous Mode of transfer. An interrupt must be generated on the Ready condition and the $\overline{INT}$ line is to be pulsed every time byte 87_H of the block is transferred. The interrupt vector of the peripheral is $4E_H$. The ready signal is active high and the $\overline{CE}$ pin is interpreted only as a chip enable.

Design the DMA initialization program for this problem giving each byte in both machine language code and the hexadecimal code and identify each write register (WR) used, along with a brief explanation of each byte.

Solution

The solution for this problem is given in convenient tabular form in Table 11-2. In step 13 of this table note that an X and a 0 are indicated in bit positions D2 and D6. The X means that these bit positions do not affect the operation of that particular register. However, for purposes of giving the proper hexadecimal code, either a 1 or a 0 must be used in these two positions. Arbitrarily we have chosen to use a 0 in both these bit positions. ■

11-3.2 Memory-to-Memory Transfers

Example 11-2

One block of data, consisting of 128_{10} bytes, is to be transferred from memory Y (connected to port A) to memory Z (connected to port B). The starting address of the block in memory Y is $3F94_H$ and each subsequent address is incremented. The starting address of the block in memory Z is $28AA_H$ and each subsequent address is decremented. The burst mode is used. On completion of the block transfer, an interrupt is to be generated on the $\overline{INT}$ line. The $\overline{CE}$ is used only as a chip enable. The interrupt vector of memory Y (port A) is $7A_H$. Ready is indicated as active low and the DMA activity is terminated at the end of the block. To make the two memories speed compatible, memory Z requires two additional T cycles.

Design the DMA initialization program for this problem, identifying each write register (WR) along with a

Table 11-2 DMA INITIALIZATION BYTE FOR EXAMPLE 11-1

Byte	*Write Registers*			*Machine Language Code*								*Hex*
No.	*Base*	*Associated*	*Functions and Comments*	*D7*	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*	*Code*
1	WR0-0	–	Setup byte—direction of transfer, port A address, and block length follow.	0	1	1	0	1	1	0	1	6D
2	–	WR0-1	Port A start address—low byte	0	0	1	1	0	1	1	0	36
3	–	WR0-3	Block length—low byte	0	0	1	1	0	1	0	1	35
4	–	WR0-4	Block length—high byte	0	0	0	0	0	1	0	0	04
5	WR1-0	–	Port A is peripheral with fixed address. No variable timing byte follows.	0	0	1	1	1	1	0	0	3C
6	WR2-0	–	Port B setup byte. Port B is memory. Starting address is incremented. No variable timing byte follows.	0	0	0	1	0	0	0	0	10
7	WR4-0	–	Defines continuous mode of operation. Port B starting address bytes and interrupt control byte follow.	1	0	1	1	1	1	0	1	BD
8	–	WR4-1	Port B start address—low byte	0	0	1	0	0	0	0	0	20
9	–	WR4-2	Port B start address—high byte	0	0	0	1	1	0	0	0	18
10	–	WR4-3	Interrupt control byte—INT is generated on ready condition.	0	1	0	1	1	1	0	0	5C
11	–	WR4-4	Pulse control byte—compared with byte counter—generates pulse on $\overline{INT}$ line when the two compare.	1	0	0	0	0	1	1	1	87
12	–	WR4-5	Interrupt vector of peripheral.	0	1	0	0	1	1	1	0	4E
13	WR5-0	–	Control byte—ready active high, stop end-of-block, and $\overline{CE}$ is chip enable only.	1	0 X	0	0	1	0 X	1	0	8A
14	WR6-0	–	Load starting addresses and reset byte counter.	1	1	0	0	1	1	1	1	CF
15	WR6-0	–	Enable DMA.	1	0	0	0	0	1	1	1	87

brief explanation of each byte. Give both the machine language and the hexadecimal codes for each byte.

Solution

The solution for this problem is given in convenient tabular form in Table 11-3. In step 13 of this table note that an X and a 0 are indicated in bit positions D2 and D6. The X means that these bit positions do not affect the operation of that particular register. However, for purposes of giving the proper hexadecimal code, either a 1 or a 0 must be used in these two positions. Arbitrarily, we have chosen to use a 0 in both these two positions. ■

11-4 PROGRAMMING THE DMA FOR SEARCH OPERATIONS

Example 11-3

A block of data, 96_{10} bytes long, is stored in memory starting at address $2A2A_H$. The block addresses increment from this address. The memory is connected to port A. The data block is to be searched for a match on $8D_H$. The match must be obtained on all bits of this byte. The DMA operation must stop when a match is obtained and an interrupt must be generated. The operation must then branch to a subroutine whose starting address low byte is BC_H. The byte-at-a-time mode of operation is used. Ready is active high and the $\overline{CE}$ pin is used for chip enable only.

Design the DMA initialization program for this problem, identifying each write register (WR) along with a brief explanation of each byte. Give both the machine language and the hexadecimal code for each byte.

Solution

The solution for this problem is given in convenient tabular form in Table 11-4. In step 10 of this table note that an X and a 0 are indicated in bit positions D2 and D6. The X means that these bit positions do not affect the operation of that particular hexadecimal code; either a 1 or a 0 must be used in these two positions. Arbitrarily, we have chosen to use a 0 in both these bit positions.

Table 11-3 DMA INITIALIZATION BYTES FOR EXAMPLE 11-2

Byte	*Write Registers*			*Machine Language Code*								*Hex*
No.	*Base*	*Associated*	*Functions and Comments*	*D7*	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*	*Code*
1	WR0-0	–	Setup byte—port A address, direction of transfer, and block length follow.	0	0	1	1	1	1	0	1	3D
2	–	WR0-1	Port A start address—low byte	1	0	0	1	0	1	0	0	94
3	–	WR0-2	Port A start address—high byte	0	0	1	1	1	1	1	1	3F
4	–	WR0-3	Block length—low byte	1	0	0	0	0	0	0	0	80
5	WR1-0	–	Port A is memory—addresses are not fixed. Port A address increments. Variable timing byte does not follow.	0	0	0	1	0	1	0	0	14
6	WR2-0	–	Port B setup byte—port B is memory, starting address is decremented. Variable timing byte follows.	0	1	0	0	0	0	0	0	40
7	–	WR2-1	Variable cycle length time—two additional T cycles—IORQ, MREQ, RD, and RW do not end one half cycle early.	1	1	0	0	1	1	1	0	CE
8	WR4-0	–	Defines burst mode of operation. Port B starting address bytes and interrupt control byte follow.	1	1	0	1	1	1	0	1	DD
9	–	WR4-1	Port B start address—low byte	1	0	1	0	1	0	1	0	AA
10	–	WR4-2	Port B start address—high byte	0	0	1	0	1	0	0	0	28
11	–	WR4-3	Interrupt control byte—interrupt is generated on ready and end-of-block. Pulse control byte does not follow	0	1	1	1	0	0	1	0	72
12	–	WR4-5	Interrupt vector of memory Y (port A).	0	1	1	1	1	0	1	0	7A
13	WR5-0	–	Control byte—ready active low, CE is chip enable only, and stop DMA at end-of-block.	1	0 X	0	0	0	0 X	1	0	82
14	WR6-0	–	Load starting addresses and reset byte counter.	1	1	0	0	1	1	1	1	CF
15	WR6-0	–	Enable DMA.	1	0	0	0	0	1	1	1	87

■

Table 11-4 DMA INITIALIZATION BYTES FOR EXAMPLE 11-3

Byte	Write Registers			Machine Language Code								Hex
No.	*Base*	*Associated*	*Functions and Comments*	*D7*	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*	*Code*
1	WR0-0	–	Setup byte, port A address, block length follows	0	0	1	1	1	1	1	0	3E
2	–	WR0-1	Port A start address, low byte	0	0	1	0	1	0	1	0	2A
3	–	WR0-2	Port A start address, high byte	0	0	1	0	1	0	1	0	2A
4	–	WR0-3	Block length, low byte	0	1	1	0	0	0	0	0	60
5	WR1-0	–	Port A is memory. Address is not fixed. Address increments. No variable timing byte follows.	0	0	0	1	0	1	0	0	14
6	WR3-0	–	Search class of operation. Match byte follows. Mask byte does not follow. Stop DMA on match and generate interrupt.	1	1	1	1	0	1	0	0	F4
7	–	WR3-2	Match byte (all bits).	1	0	0	0	1	1	0	1	8D
8	WR4-0	–	Defines byte-at-a-time mode. No port B address. Interrupt control byte follows.	1	0	0	1	0	0	0	1	91
9	–	WR4-3	Interrupt control byte. Interrupt generated on match. Interrupt vector follows (loaded in WR4-5).	0	0	0	1	0	0	0	1	11
10	–	WR4-5	Interrupt vector of memory (port A)	1	0	1	1	1	1	0	0	BC
11	WR5-0	–	Control byte. Ready active high. $\overline{CE}$ is chip enable only.	1	0	0	0	1	0	1	0	8A
12	WR6-0	–	Enable DMA.	1	0	0	0	0	1	1	1	87

Table 11-5 DMA INITIALIZATION BYTES FOR EXAMPLE 11-4

Byte	Write Registers			Machine Language Code								Hex
No.	*Base*	*Associated*	*Functions and Comments*	*D7*	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*	*Code*
1	WR0-0	–	Setup byte. Port A address. Defines port B as source Block length follows.	0	0	1	1	1	0	1	1	3B
2	–	WR0-1	Port A start address, low byte	1	1	0	1	1	1	0	1	DD
3	–	WR0-2	Port A start address, high byte	0	0	1	0	0	1	0	0	24
4	–	WR0-3	Block length, low byte	1	0	1	1	0	0	1	0	B2
5	WR1-0	–	Port A is memory. Address is not fixed. Address decrements Variable timing byte does not follow.	0	0	0	0	0	1	0	0	04
6	WR2-0	–	Setup byte for port B. Address decrements. Port B is memory. Variable timing byte does not follow.	0	0	0	0	0	0	0	0	00
7	WR3-0	–	Defines search class of operation. Match and mask bytes follow. Stop on match and generate interrupt.	1	1	1	1	1	1	0	0	FC
8	–	WR3-1	Mask byte. Bits D1, D2, D5 and D6 are masked.	0	1	1	0	0	1	1	0	66
9	–	WR3-2	Match byte	0	1	1	1	0	1	1	1	77
10	WR4-0	–	Defines byte-at-a-time mode. Port B address and interrupt control bytes follow.	1	0	0	1	1	1	0	1	9D
11	–	WR4-1	Port B start address, low byte	0	0	1	0	0	0	1	1	23
12	–	WR4-2	Port B start address, high byte	1	1	1	0	1	0	0	1	E9
13	–	WR4-3	Interrupt control byte. Interrupt generated on match. Interrupt vector follows.	0	0	0	1	0	0	0	1	11
14	–	WR4-5	Interrupt vector.	1	0	0	1	1	0	0	0	98
15	WR5-0	–	Control byte. Pin 16 used for $\overline{CE}$.	1	0	0	0	0	0	1	0	82
16	WR6-0	–	Enable DMA.	1	0	0	0	0	1	1	1	87

11-5 PROGRAMMING THE DMA FOR SEARCH-TRANSFER OPERATIONS

Example 11-4

Memory A is connected to port A of the DMA chip and memory B to port B. A block of data, $B2_H$ in length is to be transferred from memory B to memory A. The starting address in memory A is $24DD_H$ and $E923_H$ in memory B. Both addresses are decremented. Simultaneously, a match is to be performed on bits D0, D3, D4, and D7 only of the bit 77_H. When a match is found, an interrupt must be generated and a service subroutine, whose starting address low byte is 98_H, is to be executed. If a match is not found, then the DMA operation must be terminated on completion of the block transfer. The $\overline{CE}$ pin is used for chip enable only. Byte-at-a-time mode of operation is used.

Design the DMA initialization program for this problem, identifying each write register (WR) along with a brief explanation of each byte. Give both the machine language and the hexadecimal codes for each byte.

Solution

The solution for this problem is given in convenient tabular form in Table 11-5. ■

11-6 REVIEW QUESTIONS

11-1 The following statements apply to the write registers used for initialization of the DMA chip. Fill in the blanks.

a. There are _____ base registers and a total of _____ associated registers. (2; 4; 6; 7; 8; 12; 14; 16)

b. Pointer bits are written into _____ registers. (base; associated)

c. Control bytes are successively written into the _____ registers. (base; associated)

11-2 The base register of the WR0 group is loaded with the following byte:

0 0 1 1 1 1 1 0

Answer the following:

a. ___ Which port is the source port? (A; B)

b. ___ How many bytes are involved in the source port address? (1; 2)

c. ___ Which class of operation is the DMA programmed for? (transfer only; search only; combined search/transfer)

d. ___ How many bytes does the number specifying the block length contain? (1; 2)

11-3 The following statements apply to the WR1 group whose registers are loaded as shown below. Indicate true or false.

WR1-0 ⟵ 0 1 0 0 0 1 0 0
WR1-1 ⟵ 1 1 0 0 0 1 0 1

a. ___ Port A is an I/O port.

b. ___ Port A address is decremented after each byte is transferred.

c. ___ A timing byte follows the base register byte.

11-4 Refer to question 11-3 and indicate true or false.

a. ___ WR1-1 specifies that variable cycle length timing is used.

b. ___ The number of T cycles used is two.

c. ___ The $\overline{IORQ}$ signal is terminated one half cycle early.

d. ___ The $\overline{WR}$ signal is terminated one half cycle early.

3. ___ The $\overline{MREQ}$ signal is terminated one half cycle early.

f. ___ The $\overline{RD}$ signal is terminated one half cycle early.

11-5 Answer the following questions (yes or no) for the WR2 group whose base register is loaded with the following byte.

WR2-0 ⟵ 0 1 1 1 0 0 0

a. ___ Is the variable timing byte loaded after WR2-0?

b. ___ Will the port address be incremented after each byte is transferred?

c. ___ Will the port address be decremented after each byte is transferred?

d. ___ Is the port B address fixed?

e. ___ Is the port B an I/O port?

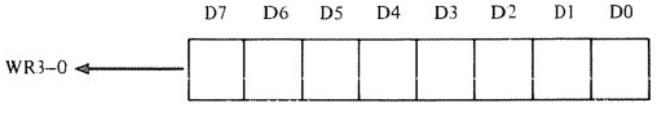
11-6 The WR3 group is concerned with the mask and match bytes. Construct the WR3-0 byte for the following conditions:

a. A match byte follows.

b A mask byte follows:

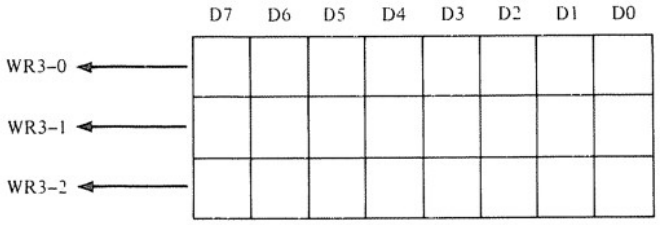
c. Stop on match.

d. Disable interrupt.



11-7 The WR3 group of the Z80 DMA is to be programmed such that the match byte consists of

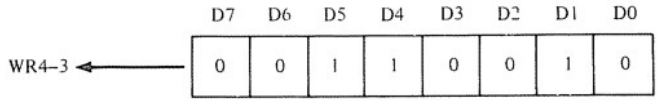
all ones. However, the first and the last bits of this byte are to be ignored. When a match is obtained, the search must be stopped but an interrupt is not generated. Fill in the bits for the 3 bytes in the following chart.



11-8 The following statements relate to writing the WR4 group of bytes. The DMA is operated in the continuous mode and no interrupt control byte follows. The hex code for the port B starting address is $3B8D_H$. Indicate true or false.

a. ____ Bit D0 and D1 of WR4-09 are 1 and 0, respectively.
b. ____ Bits D6 and D5 of WR4-0 are 1 and 0, respectively.
c. ____ Bit D4 of WR4-0 is a 1.
d. ____ WR4-2 contains 0 0 1 1 1 0 1 1.
e. ____ WR4-1 contains 1 0 0 0 1 1 0 1.
f. ____ Bit D7 of WR4-0 is a 0.

11-9 The interrupt control register, WR4-3, contains the following byte.

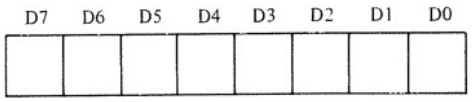


In this case the D2 and D1 bits in WR4-5 will be, respectively:

a. 1 1
b. 0 0
c. 1 0
d. 0 1

11-10 Construct the byte for the base register of group WR5 for the following conditions:

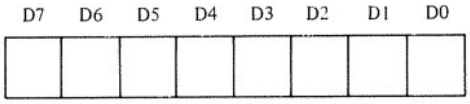
a. Auto restart on end-of-block
b. Ready active low
c. CE/WAIT line signal is wait when the bus acknowledge signal input is active.



11-11 The following statements apply to the base register of the WR6 group. Indicate true or false for the command which results in the readout from the status register.

a. ____ The hex code for the command is $B8_H$.
b. ____ The f4 bit of the command is 0.
c. ____ The f1 bit of the command is a 1.
d. ____ The f0, f2, and f3 bits of the command are all 1s.

11-12 The hex code BB is used in the WR6-0 register. Construct the byte for the WR6-1 register which will enable us to read out the low byte of port A address.



13 The following statements refer to the DMA status register. Indicate true or false.

a. ____ A 1 in D1 indicates ready active status.
b. ____ The end-of-block condition is indicated by a 0 in bit position D5.
c. ____ Bit position D2 must *always* be a 0.
d. ____ A match found condition is indicated by a 0 in bit D4 position.

12

Z80 SERIAL INPUT/OUTPUT CHIP

CHAPTER OBJECTIVES

1. To introduce students to the programmable serial I/O chip.
2. To present the principal features of the Z80 serial input-output (SIO) chip.
3. To present and describe the generalized architecture of the SIO receive data path for both the asynchronous and the synchronous modes of operation.
4. To describe byte-oriented, synchronous protocols, namely, the monosync, bisync, and external methods of character synchronization.
5. To describe the operation and use of the cyclic redundancy check (CRC) character.
6. To examine the bit-oriented protocol (SDLC mode).
7. To present and describe the generalized architecture and operation of the SIO transmit data path for both the asynchronous and the synchronous modes of operation.
8. To present the package pin assignments and functions in convenient tabular form.
9. To discuss the SIO timing involved in the read, the write, the interrupt acknowledge, and the return from interrupt cycles.

TEXTBOOK REFERENCES

For reviewing material in the textbook relevant to the topics in this chapter, the following chapters and/or sections are suggested:

1. For general discussion of serial I/O transfers	Chapter 11
2. For general discussion of programmable I/O transfers	Chapter 12
3. For conversion and synchronization problems in serial I/O transfers	Sec. 11-2
4. For description of the UART	Sec. 12-2
5. For multiple buffering of data	Sec. 12-2-1.4
6. For UART error indications	Sec. 12-2-1.5
7. For UART initialization	Sec. 12-2-1.6
8. For general UART operation and block diagram	Sec. 12-2-1.8

12-1 INTRODUCTION

The serial input-output (SIO) chip provides programmable, full duplex, two independent channels which have separate control and status lines for interfacing the Z80 CPU with modems and other devices which require serial digital communication capabilities. Basically, the SIO functions as a serial-to-parallel and parallel-to-serial converter and controller. Under program control, the SIO can be reconfigured to operate under several different conditions and modes so that it can be interfaced with I/O devices having different word lengths and a wide range of operating speeds.

The SIO handles several protocols. They include both synchronous and asynchronous protocols as well as the bit- and byte-oriented functions, usually performed by UARTS and USARTS. All incoming and outgoing data are fully buffered. The receiver data are quadruple buffered and the transmitter data are double buffered. In

the Z80 μC system, the SIO can be interfaced with either the CPU or the DMA chip. Also, it can be used in the interrupt daisy chain and has the capability of providing interrupt vectors without any additional logic.

12-2 PRINCIPAL FEATURES OF THE SIO

1. The SIO is a third-generation chip that uses the N-channel, depletion-mode, silicon-gate technology.
2. A single 5 volt-±5% power supply is required.
3. A single-phase 5-volt clock is used.
4. All inputs and outputs are fully TTL compatible.
5. The chip operates in a temperature environment of 0°C to +70°C.
6. In the synchronous operating mode, the following data rates can be used:
   a. With 2.5-MHz system clock - 0–550 K bits/second
   b. With 4.0-MHz system clock - 0–880 K bits/second
7. The SIO can operate with 5, 6, 7, or 8 bits per character and odd or even parity or no parity at all.
8. Parity, framing, and overrun errors can be detected and the SIO can operate in four clock modes, ×1, ×16, ×32, and ×64.

12-3 THE SIO ARCHITECTURE

12-3.1 The Generalized SIO Block Diagram

1. Figure 12-1 shows the generalized block diagram of the SIO. On the CPU side, the chip uses the 8-bit data bus for transfer of data between the CPU and the SIO as well as the transfer of command words from the CPU to the SIO. Control signals are sent to the SIO on a 6-bit control bus.
2. The internal control logic and the interrupt control logic blocks essentially perform the same functions that were previously described in Chapter 6 for the PIO chip and in Chapter 8 for the CTC chip.
3. Each channel has is own group of read/write registers, which include five 8-bit control registers, two sync-character registers, and two status registers. Each channel also has discrete control logic and status logic, which makes it possible for the SIO to communicate with modems or other I/O devices.
4. For both channels the write registers are designated WR0–WR7, and the read registers are designated RR0–RR2. The function performed by each register follows.

Write Registers

WR0: Contains registers pointers and initialization commands for the different modes, including CRC initialization.

WR1: Defines receive/transmit interrupts and the data transfer mode.

WR2: Contains the channel B interrupt vector (not channel A).

WR3: Contains receiver parameters and controls.

WR4: Contains various miscellaneous parameters for different receive/transmit modes.

WR5: Contains parameters and controls for the transmit operation.

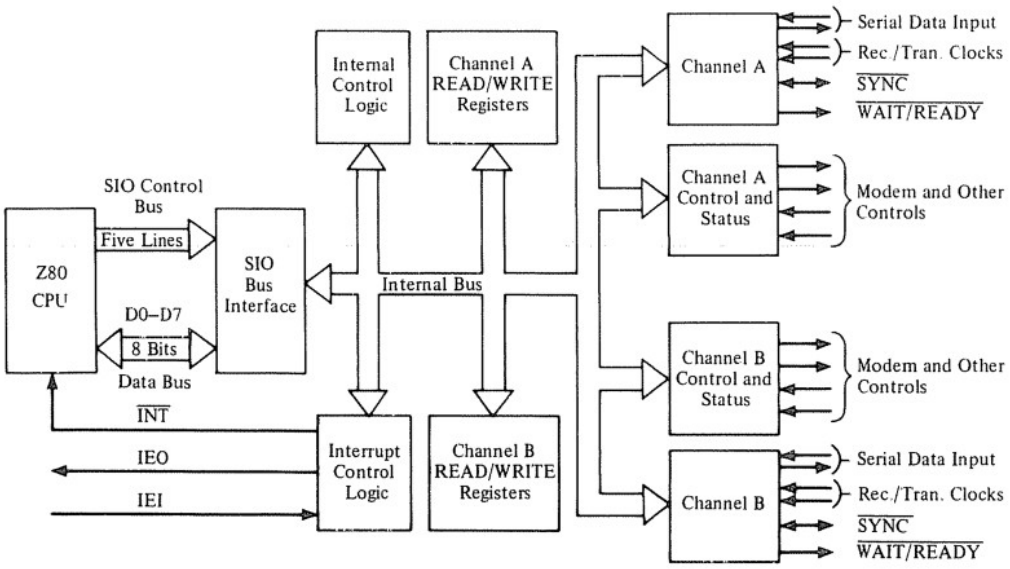


Figure 12-1 Generalized SIO Block Diagram

WR6: Contains the sync character or the SDLC (synchronous data link control) address field.

WR7: Contains the sync character or the SDLC flag.

Read Registers

RR0: Holds the buffer status, interrupt status, and external status words for the receive/transmit operations.

RR1: Contains the status for the special receive condition.

RR2: Contains the channel B modified interrupt vector (not channel A).

5. In both channels the associated logic formats synchronize and validate the data transferred in and out of that channel.
6. The modem control signals, for both channels, are general-purpose and can be used for functions other than controlling the modems.
7. The interrupt control logic provides the automatic interrupt vectoring capabilities by determining which channel, and which I/O device within that channel, has the highest priority.
8. In the SIO, channel A has a higher priority over channel B. Also, the receive status, the transmit, and the external status interrupts are assigned priorities, in that order, within each channel.

12-3.2 The Receive Data Path

1. Figure 12-2 is a simplified block diagram of the path taken by the incoming data in the SIO.
2. The incoming data are a serial bit stream in which the word length could be from 5 to 8 bits. Also, the serial data could be transmitted in either the synchronous or the asynchronous mode. (See Sec. 11-3.3 in the textbook for a brief description of these modes.) We will follow the data path in Fig. 12-2 for both these data transmission modes.

12-3.2.1 The Asynchronous Mode

1. The serial data stream enters the SIO chip via the RxDA line and goes through a 1-bit time delay.
2. From this point the data bit stream can proceed in one of the two possible paths, depending on the word length. If the character is 5 or 6 bits long, it enters the 3-bit buffer and is then loaded into the 8-bit receive shift register.
3. If the incoming character is 7 or 8 bits long, it bypasses the 3-bit buffer and is directly loaded into the 8-bit receive shift register.
4. The reason for the preceding two procedures is that the receive shift register is an 8-bit serial-in/parallel-out shift register. The output of this register is always an 8-bit parallel word regardless of the number of bits in

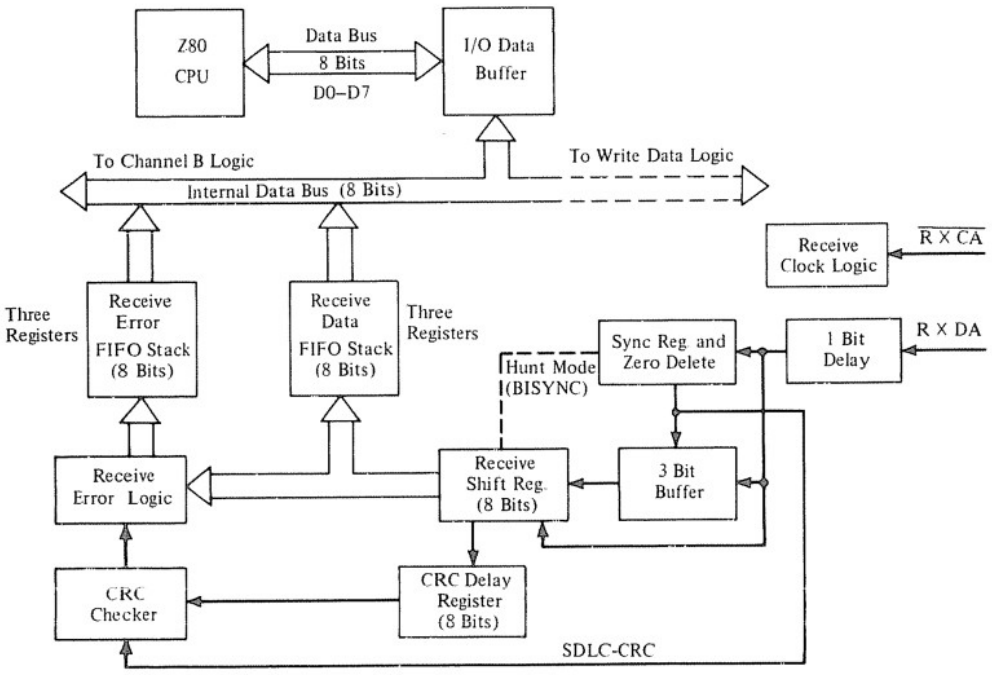


Figure 12-2 Block Diagram of Receive Data Path

the incoming word. To accomplish this goal, the SIO is designed to operate as follows:

a. The receiver logic automatically inserts 1s when a character length of less than 8 bits is used in the incoming serial bit stream.

b. If the parity is enabled and the incoming word length is 8 bits, then the parity bit is stripped from the word and only the 8 data bits are converted into a parallel format.

c. If the incoming word is 7 bits long plus parity, then the parity is not stripped but carried along.

d. If the incoming word is 5 or 6 bits long, then the 3-bit buffer inserts 1, 2, or 3 bits (binary 1s), depending on whether the word is 5 or 6 bits and whether a parity bit is included.

5. The artificially inserted 1 bits are always in the higher significant bit positions of the parallel formatted word. For example, if the incoming word were 5 bits long and had a parity bit, then the assembled word in the receive shift register would look as shown in Fig. 12-3.

6. The 8-bit word is then sent to the three-register receiver data stack, which operates on a first-in, first-out (FIFO) basis. From this stack the 8-bit word is transferred to the Z-80 CPU via the internal data bus, the I/O data buffer, and the CPU data bus (D0–D7).

7. Notice that the 8-bit parallel data word is quadruple buffered, since it passes through the 3 FIFO registers and the receive shift register. This time delay is provided, so that the CPU has ample time to service any interrupts that may come up while the SIO keeps accepting and formatting the incoming data words.

8. The formatted 8-bit word is also sent to the receive error logic. This logic checks for parity, framing, and overrun errors and generates error status word. This word is then sent to the 3-register, 8-bit, receive error stack. This is also a FIFO stack.

9. Thus, each data word is quadruple buffered and its corresponding error status word is likewise quadruple buffered, and for the same reason. Since both stacks read out to the internal data bus, correlation between the two is maintained by first reading out the error status word and then the corresponding data word. A special interrupt vector is generated if an error is indicated by the status word.

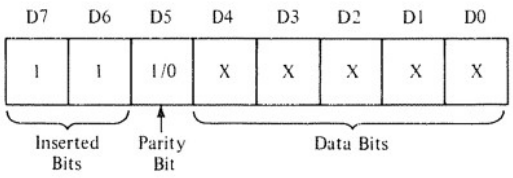


Figure 12-3 Artificially Inserted Bits in Serial Bit Stream

12-3.2.2 The Synchronous Mode

A. *Byte-Oriented Protocols*

The synchronous mode is a little more complex than the asynchronous mode. In the SIO modes of data transmissions we use some terms which are explained as follows:

1. *Monosync synchronization of character.* This is the situation where the actual character or data word is preceded by a preset 8-bit SYNC pattern which is inserted by the transmitter. The SIO has the same pattern of bits prestored in register WR7. When a match between the incoming data stream and this pattern in WR7 is made, synchronization is achieved.

2. *Bisync synchronization of character.* This is very similar to monosync synchronization except that the same SYNC bit pattern is prestored in registers WR6 and WR7. This essentially implies a 16-bit SYNC pattern. Synchronization is achieved only when the desired data character is preceded by two contiguous SYNC bit patterns. The data character is captured and synchronized only when a match is achieved between the two SYNC patterns and the contents of WR6 and WR7. Note that in both the monosync and the bisync modes, the pin labelled $\overline{\text{SYNC}}$ of the SIO chip is used as an output. It is active during the part of the receiver clock that detects the SYNC character or pattern of bits.

3. *External synchronization.* Character synchronization is achieved externally in this mode. The $\overline{\text{SYNC}}$ pin is used as an input that indicates that synchronization is achieved. After the sync pattern is detected by comparing the incoming bit stream with the contents of WR7, the logic in the I/O must wait for two cycles of the receive clock before the $\overline{\text{SYNC}}$ input line is activated. The assembly of the data character begins on the positive-going edge of the $\overline{\text{RxC}}$ receive clock that precedes the falling edge of the $\overline{\text{SYNC}}$ signal.

4. *The hunt phase.* In all three preceding modes of character synchronization, the SIO first goes through a hunt phase, during which it looks for character synchronization. The SIO goes into the hunt phase after it is reset and only if the receiver is enabled. The data transfer begins only when proper synchronization is achieved.

5. *Cyclic redundancy check (CRC) character.* This is a 16-bit error checking code that is appended to each record that is transmitted and subsequently received. A record is formed by several characters of groups of words. The CRC is generated by counting the total number of bits (both 1s and 0s) involved in the record and a mathematical formula in the transmitter. In the receiver the bits in the receiver are again checked using the CRC code appended to the record.

The message formats for the monosync, bisync, and external sync modes are shown in Fig. 12-4.

6. *The receive operation.* To understand the receive operation in the synchronous mode, refer to Fig. 12-2. The path taken by the incoming data through the receiver depends on the character length and the transmission mode used.
   a. The receive operation begins with the hunt phase. The incoming data bit stream is monitored for the SYNC pattern bits. If the SIO is programmed for the monosync mode, a match between the contents of WR7 and the incoming bits results in synchronization, and the succeeding words are then captured and handled in exactly the same manner as previously described for the asynchronous mode. In this mode the comparison is made with the incoming bits assembled in the sync register and the WR7.
   b. In the bisync mode the first 8 bits of the incoming data stream are checked in the sync register and then transferred into the receive shift register. The next 8 bits are then shifted into the sync register. Synchronization is established when the contents of the receive/sync register are matched with the contents of the WR6/WR7 register. The data/record characters are then handled as previously explained.

B. *Bit Oriented Protocol (SDLC Mode)*

1. In addition to the monosync and the bisync synchronous modes, which are byte-oriented protocols, the SIO is also designed to handle a bit-oriented protocol, called the IBM SDLC (synchronous data link control).
2. Operation in this mode also starts with the hunt phase. The incoming data bit stream first goes through the receive SYNC register. When five contiguous 1 bits are received, the sixth bit is monitored. If it is a 0 it is deleted from the bit stream. If it is a 1, then the seventh bit is checked. If the seventh bit is a 0, then it is interpreted that a flag sequence has been received. If the seventh bit is a 1, then that is interpreted that an abort sequence has been received.
3. The reformatted word is then transferred to the 3-bit register and from there to the receive shift register. The contents of this register are then compared with the prestored contents of the WR7 register, called the flag register. Once the flag character match is established, subsequent data or record are transmitted and handled as explained before.
4. The CRC characters are also appended to the data or record field in the SDLC mode operation.
5. In both the synchronous and the SDLC modes the CRC character is checked for errors, and so is passed through the CRC checker. (See Fig. 12-2.) However, in the bisync mode the SYNC-CRC is passed through an 8-bit delay register to allow the CPU ample time to make the necessary decision and to include the data character in the CRC. The SDLC-CRC is not delayed because the SIO contains logic, in this mode, that determines the bytes by means of which the CRC is checked.

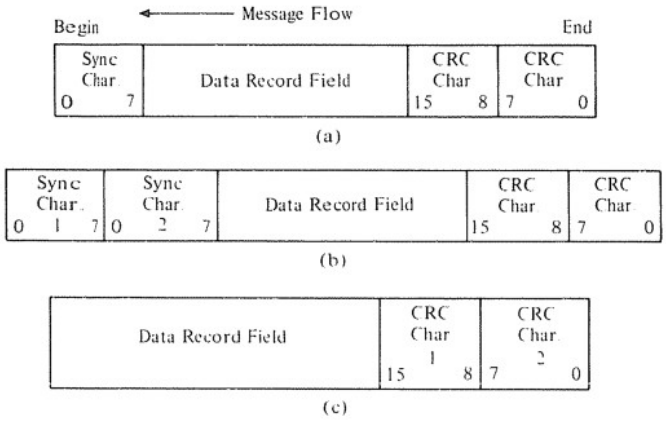


Figure 12-4 Synchronous Message Transmission Formats. (a) Monosync, Internal Sync Detect, Format. (b) Bisync, Internal Sync Detect Format. (c) External Sync Detect Format

12.3.3 The Transmit Data Path

1. Figure 12-5 is a simplified block diagram of the path taken by the serial data going out from the SIO.
2. The 8-bit parallel output data word from the Z80 CPU comes to the SIO on the D0–D7 data bus and is then loaded into the I/O data buffer.
3. From there it is loaded into the 8-bit transmit data register via the internal data bus. The data word is then transmitted out of the SIO, as explained later, depending on the transmission mode used.

12-3.3.1 The Asynchronous Mode

1. In the asynchronous mode the serial data words are shifted out at the TxDA terminal at a clock rate equal to 1, 1/16, 1/32, or 1/64 of the clock rate input to the transmit clock logic at the $\overline{\text{TxCA}}$ terminal.
2. The data word is automatically formatted by the logic in the SIO, which adds the start bit, the parity bit (odd, even, or no parity bit, depending on how it is programmed), and the appropriate number of stop bits, also depending on the program.
3. If the character word to be transmitted is 6 or 7 bits long, the SIO automatically ignores the unused bits.

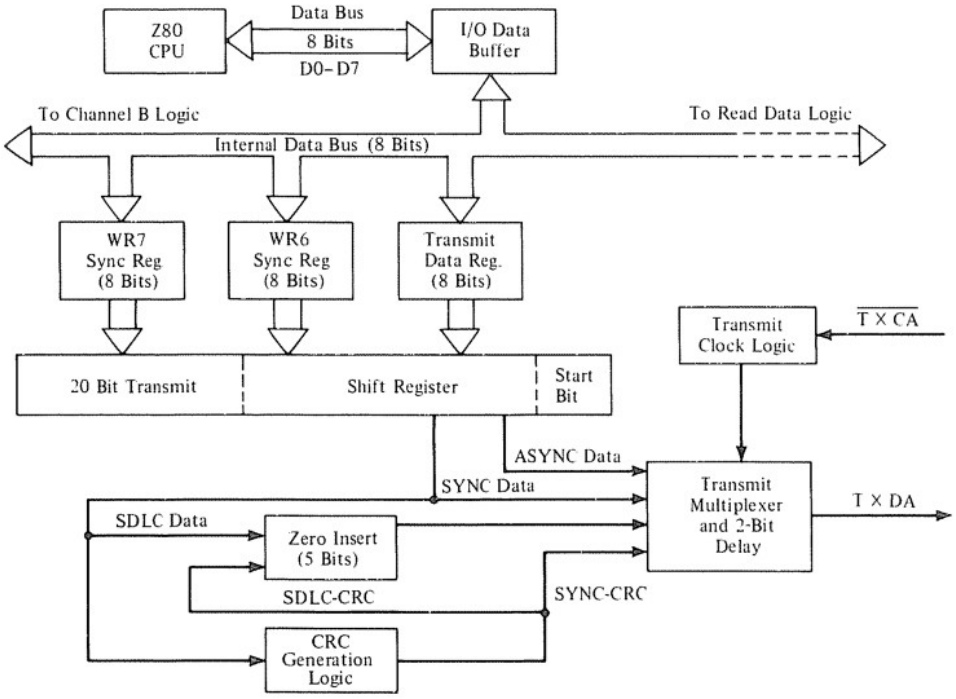


Figure 12-5 Block Diagram of Transmit Data Path

4. If the character word to be transmitted is 5 bits or less, then the transmitted word is formatted according to a preestablished convention which is described in the programming section of the Zilog Z80 SIO Technical Manual (© 1977) in tabular form for the WR5 register format.

12-3.3.2 The Synchronous Mode

1. The transmit multiplexer receives its input from four possible data paths. It then formats the word appropriately and transmits it out serially on the TxDA output terminal.
2. WR6 and WR7 contain the SYNC characters which are transferred into the 20-bit shift register at the beginning of the message. They are also used as time fillers in the middle of a message if a transmit underrun condition happens.
3. The SYNC pattern word(s) is (are) followed by the actual data word or character. The two CRC characters are then generated and appended as shown in Fig. 12-4.
4. As mentioned before, the SDLC message is called the frame whose format is shown in Fig. 12-6.
5. The SDLC message opens and closes with the 8-bit flag, which is 01111110. It also contains an 8-bit address field. This is called the secondary station address. It is used in the address search mode, which utilizes this address to either accept or reject the frame.
6. From the 20-bit transmit shift register, the SDLC data word is shifted into the zero insert logic block. When the closing and opening flags are transmitted, this block is disabled. But, when the other fields are transmitted, a 0 is inserted after five contiguous 1s in the data stream.
7. In the SDLC mode, the data generated by the CRC generator logic are also routed through the zero insert logic block.

12-4 PACKAGE PIN ASSIGNMENTS AND FUNCTIONS

The SIO is packaged in a standard 40-pin DIP package. The various pins and their respective functions, along with

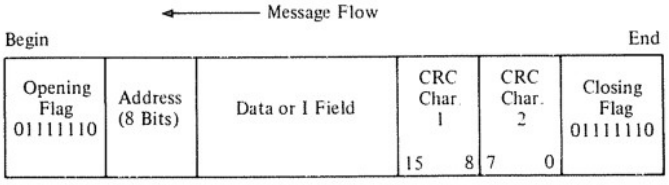


Figure 12-6 The SDLC Message Format

their designations, are presented in convenient tabular form. Functionally, the pins can be divided into nine groups. Because of package pin limitations, a peculiar problem arises on the SIO on the I/O side of the chip. Three lead bonding configurations are possible. The group on the CPU side remains constant and is shown in Fig. 12-7. The three configurations on the I/O side will be covered after we have presented the various pin functions first.

12-4.1 The CPU Side Pins

1. Figure 12-7 shows the five groups of pins on the CPU side of the SIO chip. These pins and their groupings remain constant regardless of the three different configurations on the channel side of the chip.
2. The pin functions and their designations are shown in Tables 12-1 through 12-5. Some of the lines in these

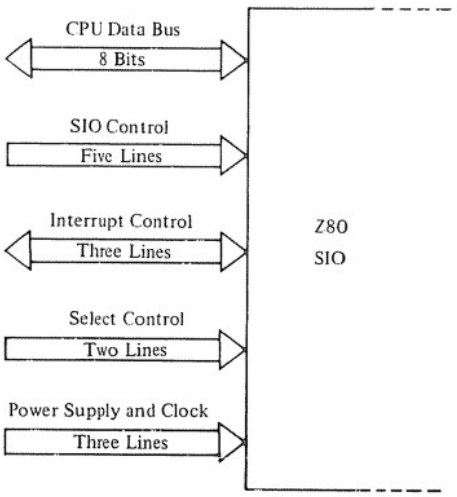


Figure 12-7 Block Diagram of the SIO Package Pin Groups on the CPU Side

Table 12-1 DATA TRANSFER BUS GROUP

Pin Designations	*Signal flow Directions*	*Functions*
D0–D7	SIO ⟷	8-bit, tristate, CPU data bus, is active high. It is used for data and commands transfers between the SIO and the CPU.

Table 12-2 SIO CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
$\overline{CE}$	SIO ⟵	Chip enable, active low signal that commands the SIO chip to transmit data during the read cycle. During the write cycle, the SIO accepts data or command words from the CPU.
$\overline{RESET}$	SIO ⟵	Reset signal, active low. It does the following: (1) Disables transmitters and receivers, (2) disables all interrupts, (3) forces the modem controls to the high-impedance state, (4) Forces TxDA and TxDB marking. After the SIO is reset, the control registers must be rewritten before data are received or transmitted.
$\overline{M1}$	SIO ⟵	Machine cycle 1 signal, active low. When $\overline{M1}$ and $\overline{RD}$ are simultaneously active the CPU is fetching an instruction from the memory. When $\overline{M1}$ and $\overline{IORQ}$ are simultaneously active, the CPU is acknowledging an interrupt. The SIO then places its interrupt vector on the CPU data bus, provided it has the highest priority and it has requested an interrupt.
$\overline{IORQ}$	SIO ⟵	I/O request from CPU signal, active low, is used for transferring data between CPU and SIO as well as for sending commands to the SIO. The SIO write cycle is initiated by the active status of $\overline{CE}$ and $\overline{IORQ}$ and inactive status of $\overline{RD}$. During the write cycle the CPU writes either the data word or the control word into the selected channel as specified by the C/$\overline{D}$ signal. The SIO read cycle is initiated by simultaneous activation of the $\overline{CE}$, $\overline{IORQ}$, and $\overline{RD}$ signals. During the read cycle, the SIO channel selected by the B/$\overline{A}$ signal transfers the data word to the CPU via the data bus.
$\overline{RD}$	SIO ⟵	Read cycle status, active low. When used simultaneously in conjunction with $\overline{CE}$ and $\overline{IORQ}$, it signals the SIO to transfer data to the CPU. During the write cycle, $\overline{RD}$ is inactive while $\overline{CE}$ and $\overline{IORQ}$ are active.

Table 12-3 INTERRUPT CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
$\overline{INT}$	SIO ⟶	Interrupt request. The $\overline{INT}$ is pulled low by the SIO chip when it is requesting an interrupt.
IEI	SIO ⟵	Interrupt enable in, active high. Used in the system-wide priority interrupt daisy chain. A high or active signal indicates that no other device with higher priority is currently being serviced.
IEO	SIO ⟶	Interrupt enable out, active high. Also used in the interrupt priority daisy chain. It is active only when IEI is active and the CPU *is not* servicing an interrupt from the SIO. When inactive, it blocks interrupt signals from lower-priority I/O devices while the CPU is servicing an I/O device having a higher priority.

Table 12-4 SELECT CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
C/$\overline{D}$	SIO ⟵	Control or data select signal. In the high state it informs the SIO that the word output by the CPU on the data bus must be accepted by the selected channel as a command or a control word. When in the low state, the selected channel will interpret and accept the word on the data bus as a data word.
B/$\overline{A}$	SIO ⟵	Channel select signal. When high, channel B is selected and when low channel A is accessed. Transfers are then made between the CPU and the selected channel. This signal is often transmitted to the SIO on the A0 line of the address bus from the CPU.

Table 12-5 POWER SUPPLY AND CLOCK GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
ø	SIO ⟵	Single-phase TTL-level clock input.
+5V	SIO ⟵	This is the single power supply of 5 volt-±5% that is required.
GND	SIO ⟵	This is the power supply ground.

tables may indicate signal flow in one direction and others in the other direction. The function tables clarify these with the following arrow symbols:

SIO → Indicates signals flowing *from the SIO.*
SIO ← Indicates signals flowing *into the SIO.*
SIO ↔ Indicates bidirectional flow on the same line(s).

12-4.2 The I/O Side Pins

1. From the presentation in Section 12-4.1 and Table 12-6 we find that 21 of the 40 package pins have been used by the CPU side, leaving 19 pins for the I/O side.
2. The demands of the I/O functions are such that these 19 pins are just not adequate for all I/O situations. Some compromises have to be made. In the SIO, these compromises are made such that in one case two signals from the chip are internally bonded together to one package pin. In two other instances one output from the chip is not brought out to a package pin at all. We will look at each of these three output pin configurations after we have discussed all the functions on the I/O side of the chip.
3. Figure 12-8 is the block diagram of the SIO package pin groups on the I/O side. Notice that the number of lines for channel A are fixed. The previously mentioned pin limitation compromises are made only in

Table 12-6 CPU SIDE PIN NUMBER ASSIGNMENTS

Pin Designations	*Pin Numbers*	*Pin Designations*	*Pin Numbers*
CPU Data Bus		*Interrupt Control*	
D0	40	$\overline{INT}$	5
D1	1	IEI	6
D2	39	IEO	7
D3	2		
D4	38	*Select Control*	
D5	3		
D6	37	C/$\overline{D}$	33
D7	4	B/$\overline{A}$	34
SIO Control		*Power Supply and Clock*	
$\overline{CE}$	35		
$\overline{RESET}$	21	ø	20
$\overline{M1}$	8	+5V	9
$\overline{IORQ}$	36	GND	31
$\overline{RD}$	32		

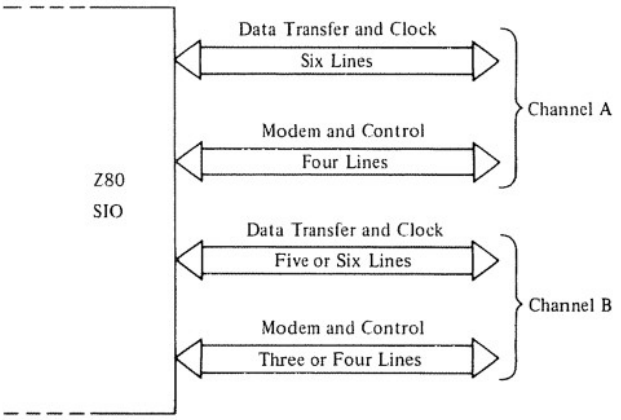


Figure 12-8 Block Diagram of the SIO Package Pin Groups on the I/O Side

Table 12-7 DATA TRANSFER AND CLOCK GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
RxDA, RxDB	SIO ⟵	Receive data, active high. Serial data, TTL compatible, signals are input to the SIO.
$\overline{RxCA}$, $\overline{RxCB}$	SIO ⟵	Receive clocks, active low. These clock inputs may be driven by the Z80 CTC timer, discussed in Chapter 8, for the programmable generation of the baud rate. In asynchronous mode of operation the receiver clocks could be 1, 16, 32, or 64 times the incoming data rate. Incoming data are sampled on the positive-going edge of the $\overline{RxC}$ signal. Both inputs are buffered by Schmitt triggers.
TxDA, TxDB	SIO ⟶	Transmit data, active high. Serial data, TTL compatible, signals are output by the SIO.
$\overline{TxCA}$, $\overline{TxCB}$	SIO ⟵	Transmitter clocks, active low. These clock inputs may be driven by the Z80 CTC timer, discussed in Chapter 8, for the programmable generation of the baud rate. In asynchronous mode of operation, the transmitter clocks could be 1, 16, 32, or 64 times the outgoing data rate, but the clock multiple factor for the receiver and the transmitter must be the same. The transmitted data (TxD) changes states at the negative-going edge of the $\overline{TxC}$ pulse. Both inputs are buffered by Schmitt triggers.
$\overline{SYNCA}$, $\overline{SYNCB}$	SIO ⟷	Synchronization, bidirectional dignals, active low. In the asynchronous mode, change of signals (high-to-low or low-to-high) affect the SYNC/HUNT status bits in RRO. When externally synchronized, $\overline{SYNC}$ must be driven low on the second positive-going edge of the receiver clock ($\overline{RxC}$) after the first rising edge of ($\overline{RxC}$) on which the last character of the SYNC character was received. This means that after the sync pattern is detected, the $\overline{SYNC}$ input must be activated only after waiting for two full $\overline{RC}$ cycles. When used in the internal synchronization mode (bisync or monosync), these pins are active as outputs during the portion of the $\overline{RxC}$ cycle in which SYNC characters are recognized.
$\overline{W/RDYA}$, $\overline{W/RDYB}$	SIO ⟶	Wait/ready signal lines. The lines are open (drain open) when they are programmed for the wait function. They can be either in the high or the low state when programmed for the ready function. As a wait function they are used to synchronize the CPU to the SIO data transfer rate. When used with a DMA controller, they can be used for the ready function.

Table 12-8 MODEM AND OTHER CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
$\overline{\text{RTSA}}$, $\overline{\text{RTSB}}$	SIO ⟶	Ready to send, active low. This signal goes low when the RTS bit is set. In the asynchronous mode, when the RTS bit is reset, the output line goes high after the transmitter is empty. When used in the synchronous mode, the $\overline{\text{RTS}}$ lines only follow the state of the RTS bit.
$\overline{\text{CTSA}}$, $\overline{\text{CTSB}}$	SIO ⟵	Clear to send, active low. If programmed as auto enables, low signal input on these lines enables the corresponding transmitter. Otherwise, they can be programmed and used as general-purpose inputs. Logic level transitions in either direction on these lines are detected by the SIO, which then sends an interrupt request to the CPU. Both lines are buffered with Schmitt triggers.
$\overline{\text{DTRA}}$, $\overline{\text{DTRB}}$	SIO ⟶	Data terminal ready, active low. These lines follow the DTR bit state as determined by the program. They can also be used as general-purpose outputs.
$\overline{\text{DCDA}}$, $\overline{\text{DCDB}}$	SIO ⟵	Data carrier detect, active low. If the SIO is programmed for auto enables, these lines function as receive enables. Otherwise, they can be used as general-purpose inputs. Logic level transitions in either direction in these lines are detected by the SIO, which then sends an interrupt request to the CPU. Both lines are buffered with Schmitt triggers.

channel B, and that is why alternate numbers are given for the lines involved.

4. The package pin assignments for channel A remain the same, as shown in Table 12-9, regardless of the three configurations for channel B.
5. Because of the previously mentioned limitations imposed by the availability of only 40 pins on the package, some sacrifices are made in channel B signals, namely, receive clock, transmit clock, data terminal ready, and SYNC signals. Two of the preceding four signals are internally bonded to a single package pin in some cases.
6. In the SIO/O configuration all four previously-mentioned signals are available, but $\overline{\text{TxCB}}$ and $\overline{\text{RxCB}}$ are internally bonded together. The resulting pin number assignment is shown in Table 12-10.
7. In the SIO/1 configuration, the DTRB signal is sacrificed and not connected. However, the $\overline{\text{TxCB}}$, the $\overline{\text{RxCB}}$ and the $\overline{\text{SYNCB}}$ signals are available. The pin number assignments are shown in Table 12-11.
8. In the SIO/2 configuration, the $\overline{\text{SYNC}}$ is sacrificed and not connected. However, the $\overline{\text{TxCB}}$, the $\overline{\text{RxCB}}$ and the $\overline{\text{DTRB}}$ signals are available. The pin number assignments are shown in Table 12-12.

12-5 THE SIO TIMING

In this section we briefly describe and discuss the SIO timing charts involved in the read, write, interrupt acknowledge, and the return from interrupt cycles. The frequency and phase relationship of the SIO clock must be the same as that of the CPU clock.

Table 12-9 Channel A Pin Number Assignments

Pin Designations	*Pin Numbers*	*Pin Designations*	*Pin Numbers*
Data Transfer and Clock		*Modem and Control*	
RxDA	12	$\overline{\text{RTSA}}$	17
$\overline{\text{RxCA}}$	13	$\overline{\text{CTSA}}$	18
$\overline{\text{TxDA}}$	15	$\overline{\text{DTRA}}$	16
$\overline{\text{TxCA}}$	14	$\overline{\text{DCDA}}$	19
$\overline{\text{SYNCA}}$	11		
$\overline{\text{W/RDYA}}$	10		

Table 12-10 The SIO/0, Channel B Pin Number Assignments

Pin Designations	*Pin Numbers*	*Pin Designations*	*Pin Numbers*
Data Transfer and Clock		*Modem and Control*	
RxDB	28	$\overline{\text{RTSB}}$	24
$\overline{\text{RxCB}}$, $\overline{\text{TxCB}}$	27	$\overline{\text{CTSB}}$	23
TxDB	26	$\overline{\text{DTRB}}$	25
$\overline{\text{SYNCB}}$	29	$\overline{\text{DCDB}}$	22
$\overline{\text{W/RDYB}}$	30		

Table 12-11 The SIO/1, Channel B Pin Number Assignments

Pin Designations	*Pin Numbers*	*Pin Designations*	*Pin Numbers*
Data Transfer and Clock		*Modem and Control*	
RxDB	28	$\overline{\text{RTSB}}$	24
$\overline{\text{RxCB}}$	27	$\overline{\text{CTSB}}$	23
TxDB	25	$\overline{\text{DCDB}}$	22
$\overline{\text{TxCB}}$	26		
$\overline{\text{SYNCB}}$	29		
$\overline{\text{W/RDYB}}$	30		

Table 12-12 The SIO/2, Channel B Pin Number Assignments

Pin Designations	*Pin Numbers*	*Pin Designations*	*Pin Numbers*
Data Transfer and Clock		*Modem and Control*	
$\overline{\text{RxDB}}$	29	$\overline{\text{RTSB}}$	24
$\overline{\text{RxCB}}$	28	$\overline{\text{CTSB}}$	23
TxDB	26	$\overline{\text{DTRB}}$	25
$\overline{\text{TxCB}}$	27	$\overline{\text{DCDB}}$	22
$\overline{\text{W/RDYB}}$	30		

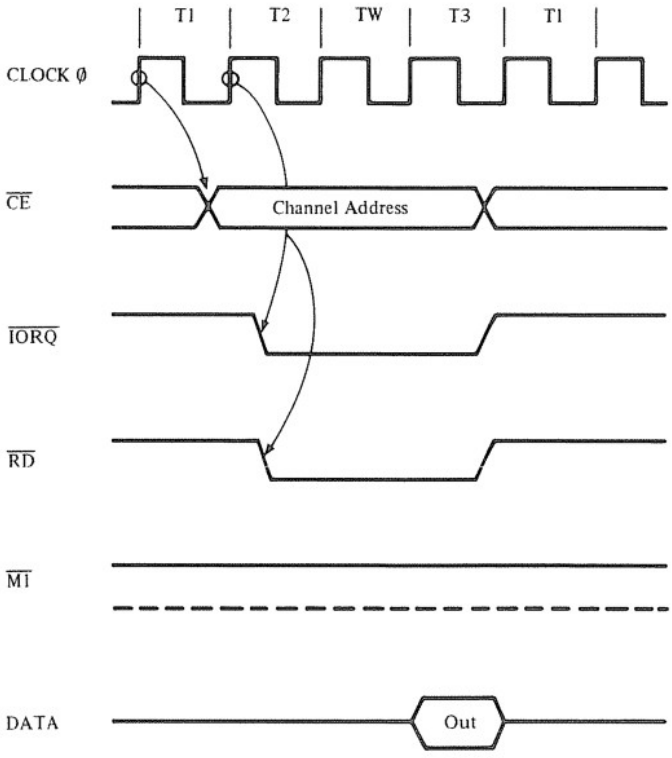


Figure 12-9 SIO Read Cycle Timing

12-5.1 The Read Cycle

Figure 12-9 shows the timing relationships during a CPU input instruction when the CPU requests a data or status word from the SIO. Note that the read cycle is initiated by the active status of $\overline{\text{IORQ}}$ and $\overline{\text{RD}}$ and the inactive status of $\overline{\text{M1}}$. The wait state TW is automatically inserted.

12-5.2 The Write Cycle

Figure 12-10 shows the timing relationships during a CPU output instruction when the CPU writes a data word or a control byte in the SIO chip. Notice that the write cycle is initiated by the active status of $\overline{\text{IORQ}}$ and the inactive status of $\overline{\text{RD}}$ and $\overline{\text{M1}}$. The wait state is automatically inserted.

12-5.3 The Interrupt Acknowledge Cycle

1. An interrupt request to the CPU is indicated by the $\overline{\text{INT}}$ line going low.
2. Sometime thereafter the CPU acknowledges the interrupt by the $\overline{\text{M1}}$ line going low and a few clock periods later, the $\overline{\text{IORQ}}$ going low.

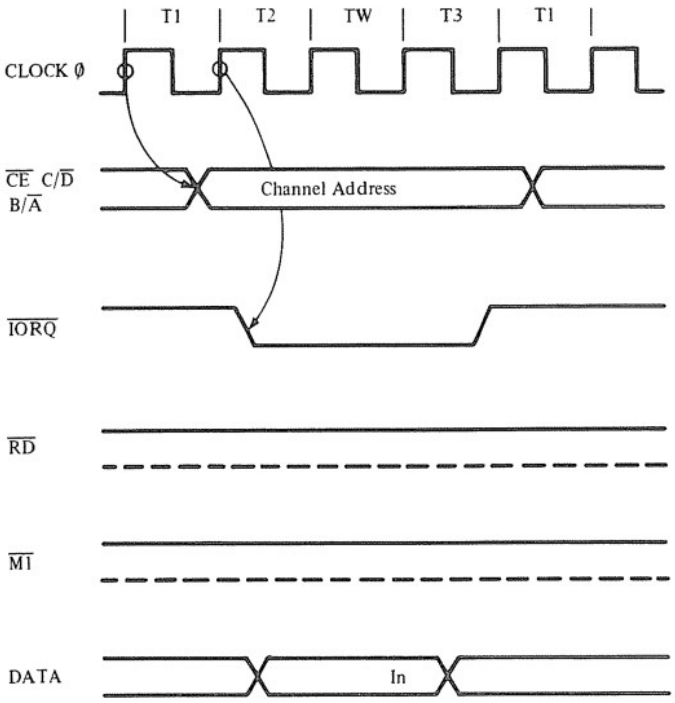


Figure 12-10 SIO Write Cycle Timing

3. The circuits in the daisy chain then determine the interrupt priority rating of the requesting device. The IEI line of the highest priority device is high.
4. The IEO of any device whose interrupt request is being serviced, or is pending, is in the low state.
5. The IEO of devices with no pending interrupts or not currently being serviced have IEO = IEI.
6. All interrupt signals are prevented from changing while $\overline{M1}$ is low. This ensures stability in the daisy chain.
7. When $\overline{IORQ}$ is active, the highest-priority device requesting interrupt service places its interrupt vector on the data bus and latches in its interrupt request.
8. The timing relationships are shown in Fig. 12-11.

12-5.4 The Return from Interrupt Cycle

1. A return from interrupt instruction (RETI) is inserted at the end of every interrupt service subroutine.
2. RETI is a 2-byte op code (ED-4D). It resets the interrupt-under-service flip-flop, thereby ending the interrupt that has just been processed.
3. Then the first byte of the RETI (i.e., ED) is decoded and the daisy chain forces the IEO high of any interrupt that has not been acknowledged. Thus, the daisy chain asserts that the device that is currently being serviced is the only one with a high IEI and a low IEO.
4. The timing relationships for this cycle are shown in Fig. 12-12.

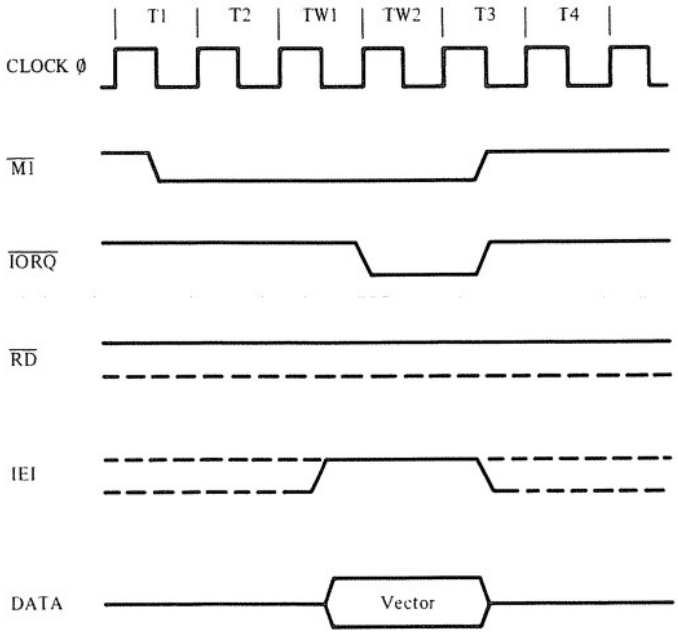


Figure 12-11 SIO Interrupt Acknowledge Cycle Timing

12-6 NOTES ON CRC OPERATION

12-6.1 Introduction

A brief explanation of how the CRC logic operates, in both the transmitter and the receiver equipment, is given in this section without resorting to any mathematical analysis. The CRC operation is based on a serial shift register and exclusive-OR circuits. Students will recall that an exclusive-OR circuit, such as the one shown in Fig. 12.13a, has outputs as shown in the truth table of Fig. 12-13b, for various combinations of input signals.

12-6.2 Principle of Operation

Explanations of the basic steps involved in the CRC operations follow:

1. At the transmitting end, two 8-bit CRC characters are generated and appended to the data/record field, as shown in Fig. 12-4.

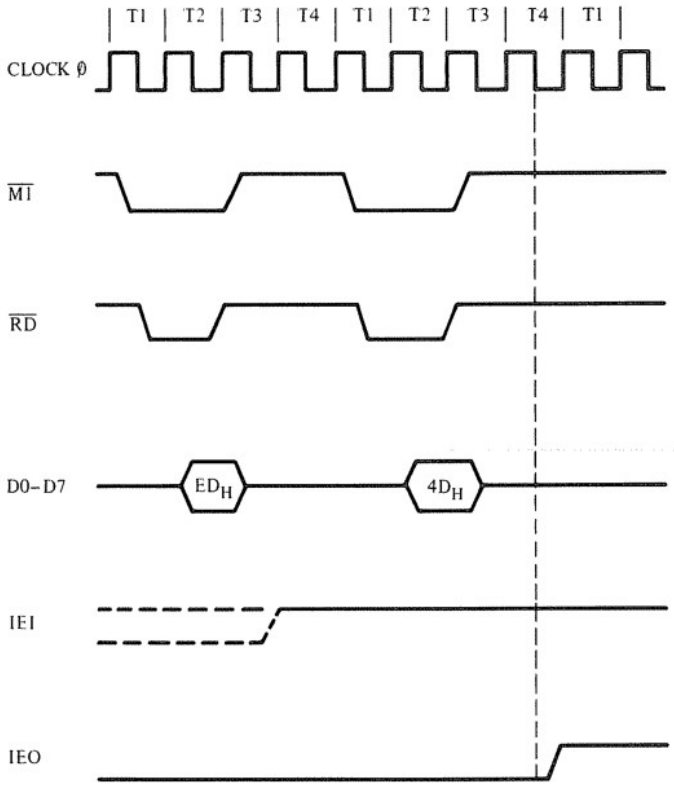
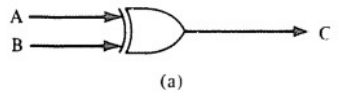


Figure 12-12 SIO Return from Interrupt Cycle Timing



(a)

Inputs		Output	Rules
A	B	C	No.
0	0	0	1
1	0	1	2
0	1	1	2
1	1	0	1

(b)

Figure 12-13 The Exclusive-OR. (a) Circuit Symbol. (b) The Truth Table

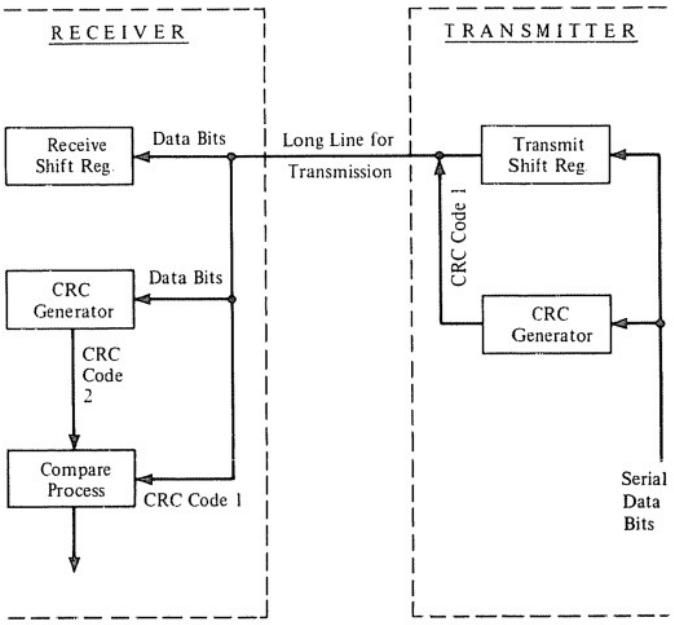


Figure 12-14 CRC Code in Transmission and Reception

2. The compositions of this 16-bit CRC code are determined by the combination of 1s and 0s in the data/record field which is to be serially transmitted.
3. The receiver also has a CRC code generator, the same as in the transmitter.
4. As the serial bits of the data/record field are transmitted over the long line, they are also simultaneously fed into the CRC generator, 1 bit at a time. The CRC code then follows the data/record field, as shown in Fig. 12-4.
5. At the receiving end, the bits in the data/record field are handled as explained in Sec. 12-3.2 (see Fig. 12-2), and another CRC code (16 bits) is generated as the incoming bits are received.
6. When the CRC code that is generated by the transmitter is received, it is compared with the CRC code that is generated by the receiver.
7. If there is an error involved in the transmission, then the CRC codes (one generated by the transmitter and the other by the receiver) are not identical and an error is indicated. A match between the two CRC codes indicates an error-free transmission. This process is shown in a generalized block diagram form in Fig. 12-14. The CRC code generator is explained in Sec. 12-6.3.

12-6.3 The CRC Generator in the Transmitter

1. The CRC generator consists of a multisectional, serial shift register whose output is fed into an exclusive-OR gate. This is X01 gate in Fig. 12-15.
2. Additional exclusive-OR gates are inserted between the various section of this shift register. These are the X02 and X03 gates shown in Fig. 12-15.
3. The output from X01 is fed back into one of the inputs each of X02 and X03 and into the MSB position of the third segment of the serial shift register, i.e., bit position 15.
4. The number of exclusive-OR gates and their placements in the segmented shift register varies, depending on the code used. This is determined mathematically.
5. Before transmission of the data begins, the serial shift register is cleared to 0s.
6. The first serial bit of the data stream is then applied to one of the inputs of X01 at point *M*. The output of X01 is fed back into X02 and X03 and bit position 15 of the register.
7. Simultaneously, a shift pulse is applied to the shift register. Thus, if the first bit of the data stream were a 1, a digital 1 would be loaded into bit positions 4, 10, and 15 of the register.
8. As subsequent bits of the data stream are fed at point *M* and shift pulses applied, a 1 or a 0 will be output by each of the exclusive-OR gates, depending on the input at *M* and the inputs from previous bit positions in the register.
9. When 16 bits are input at *M*, a 16-bit CRC code is generated in the register, depending on the combination of 1s and 0s input at *M*.

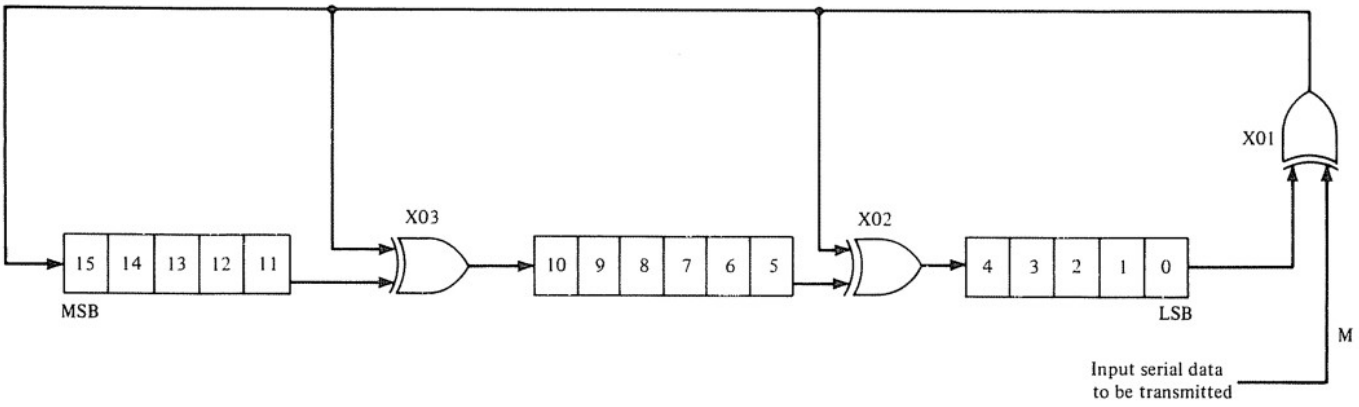


Figure 12-15 The CRC Generator

10. After the original data word is transmitted, the contents of the register are serially shifted out and put on the communications line.

12-6.4 The CRC Generator in the Receiver

1. The CRC generator in the receiver is exactly the same as the one in the transmitter.
2. The incoming data bits on the communications line are fed into the receive shift register and simultaneously into the CRC generator.
3. The CRC code is then created by the generator in exactly the same manner as was described in Sec. 12-6.3 for the transmitter.
4. When the transmitter-generated CRC code arrives at the receiver end, it is compared with the receiver-generated CRC code. If the transmission is error-free, then the two CRC codes (one generated by the transmitter and the other by the receiver) must match exactly. Any decrepancy between the two will indicate an error.

12-6.5 The Comparison Process in the Receiver

1. The actual comparison of the two CRC codes in the receiver can be done by feeding both the CRC codes into a comparator, as shown in Fig. 12-14. The discrepancy between the two CRC codes, if any, is then detected by the comparator.
2. In reality, a more sophisticated method for comparison is used, without the use of a comparator. However, before we describe this method, let us take a look at some of the properties of the exclusive-OR gate, as shown in Fig. 12-16.
3. The truth table of the exclusive-OR (Fig. 12-13b) shows two rules which are defined below:
   Rule 1 If both inputs to the exclusive-OR are the same, then the output is always a 0.
   Rule 2 If one of the inputs to the exclusive-OR is a 0, the output is the same as the other input.
4. What Rule 1 says is that when both inputs to the exclusive-OR are the same, this gate essentially acts like a two-input AND gate, with one input being inverted and the two inputs tied together as shown in Fig. 12-16a. It is readily seen that the output is always a 0, regardless of the input.
5. What rule 2 says is that if one of the two inputs is always a 0, then the exclusive-OR gate again acts like a 2-input AND gate with one inverted input, as shown in Fig. 12-16b. The inverted input is always held at 0. The output then reflects the digit 1 input at the other terminal of the AND gate.
6. The implementation of the preceding two rules enables us to perform a comparison of the two CRC codes without the use of a comparator.

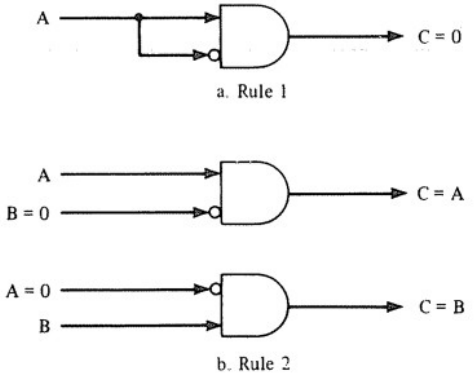


Figure 12-16 Rules 1 and 2 for the Exclusive-OR

7. This is accomplished by feeding the incoming, transmitter-generated CRC code directly into the CRC generator of the receiver.
8. When the first bit of the transmitter-generated CRC code is applied at point M of the generator (Fig. 12-15), gate X01 follows rule 1 and outputs a 0, provided the first bit of the two CRC codes match.
9. The rest of the exclusive-OR gates in the generator follow rule 2. Note that the 0 output of X01 is now shifted into the MSB position of the register.
10. Subsequent incoming bits at point *M* follow the same procedure and the output of X01 continues to be a 0 as long as the corresponding bits of the two CRC codes match. Thus after all 16 bits are compared, the shift register contains all 0s, thereby indicating a perfect match between the two CRC codes.

12-7 REVIEW QUESTIONS

12-1 Indicate true or false for the SIO chip.
- **a.** ___ The SIO contains two fully independent simplex communications channels.
- **b.** ___ The SIO interfaces the Z80 CPU with modems and other I/O devices which require parallel digital communications capabilities.
- **c.** ___ The SIO functions as a serial-to-parallel and parallel-to-serial converter.
- **d.** ___ The SIO handles both synchronous and asynchronous protocols.
- **e.** ___ The SIO handles only the byte-oriented protocols.

12-2 In the SIO, receiver data are _____ buffered and the transmitter data are _____ buffered. (single; double; triple; quadruple)

12-3 Indicate the data rates at which the SIO can operate in the synchronous mode.
- **a.** _____ With 2.5-MHz system clock.
- **b.** _____ With 4.0-MHz system clock. (0–880; 150–880; 0–150; 0–550; 50–550K bits per second)

12-4 Circle the number of bits per character that the SIO can handle.
- **a.** 6
- **b.** 10
- **c.** 12
- **d.** 8
- **e.** 16
- **f.** 5

12-5 Answer yes or no for the capabilities of the SIO chip.
- **a.** ___ Can it work with two stop bits?
- **b.** ___ Is it possible to detect framing and overrun errors?
- **c.** ___ Can it operate at eight times the clock rate?
- **d.** ___ Can it operate at 64 times the clock rate?
- **e.** ___ Is it possible to have more than three lead bonding configurations?

12-6 Indicate true or false for the SIO package pin functions.
- **a.** ___ The D0–D7 bus is used for data and command transfers between the CPU and the I/O devices.
- **b.** ___ During the write cycle the SIO accepts data or command words from the CPU.
- **c.** ___ When $\overline{\text{RESET}}$ is active, it enables transmitter and receiver.
- **d.** ___ When $\overline{\text{RESET}}$ is active, it disables all interrupts.

12-7 Circle the statements that are applicable to the $\overline{\text{IORQ}}$ signal.
- **a.** Initiates the write cycle when $\overline{\text{CE}}$ is inactive.
- **b.** Initiates the write cycle when $\overline{\text{CE}}$ is active.
- **c.** Initiates the read cycle when $\overline{\text{CE}}$ and $\overline{\text{RD}}$ are active.
- **d.** During the write cycle, $\overline{\text{IORQ}}$ and $\overline{\text{RD}}$ are active while $\overline{\text{CE}}$ is inactive.

12-8 Which of the following signals is pulled low by the SIO chip when it is requesting an interrupt?
- **a.** IEI
- **b.** $\overline{\text{INT}}$
- **c.** IEO
- **d.** $\overline{\text{IORQ}}$

12-9 The following statements relate to the IEI and the IEO signals. Indicate true or false.
- **a.** ___ IEI is used in the system-wide priority interrupt daisy chain.
- **b.** ___ When IEI is high it indicates that a device with higher priority is currently being serviced.
- **c.** ___ IEO is inactive when IEI is active.
- **d.** ___ When IEO is inactive, it blocks interrupt signals from lower-priority I/O devices.

12-10 Which of the following package pins will be active during transmission of serial data via channel B?
- **a.** TxDA
- **b.** $\overline{\text{RxCB}}$
- **c.** TxDB
- **d.** $\overline{\text{RxCA}}$

12-11 The incoming data into the SIO is sampled on the _____ going edge of the _____ signal. (Positive; negative; TxD; $\overline{\text{RxC}}$; $\overline{\text{RxD}}$)

12-12 Which of the following signals are buffered by Schmitt triggers?
- **a.** RxDA and RxDB
- **b.** $\overline{\text{RxCA}}$ and $\overline{\text{RxCB}}$
- **c.** TxDA and TxDB
- **d.** $\overline{\text{TxCA}}$ and $\overline{\text{TxCB}}$

12-13 The following statements apply to the $\overline{\text{SYNCA}}$ and $\overline{\text{SYNCB}}$ signals. Indicate true or false.
- **a.** ___ In the synchronous mode, transition of this signal affects the SYNC/HUNT status bits in the RRO.
- **b.** ___ After the sync pattern is detected, the $\overline{\text{SYNC}}$ input must be activated after waiting for two full $\overline{\text{RxC}}$ cycles.
- **c.** ___ In the bisync mode these pins are active as outputs during the portion of the $\overline{\text{RxC}}$ cycle in which sync characters are recognized.

12-14 The following statements apply to the $\overline{\text{RTSA}}$, $\overline{\text{RTSB}}$, $\overline{\text{CTSA}}$, and $\overline{\text{CTSB}}$ signals. Indicate true or false.
- **a.** ___ When used in the asynchronous mode, $\overline{\text{RTSA}}$ and $\overline{\text{RTSB}}$ follow the state of the RTS bit.
- **b.** ___ When the RTS bit is set, $\overline{\text{RTSA}}$ and $\overline{\text{RTSB}}$ go low.
- **c.** ___ Low signal on $\overline{\text{CTSA}}$ or $\overline{\text{CTSB}}$ enables the corresponding transmitter.
- **d.** ___ The SIO sends an interrupt request to the CPU when the levels of either $\overline{\text{CTSA}}$ or $\overline{\text{CTSB}}$ change.

12-15 _____ Is it possible to use the $\overline{\text{DTRA}}$ pin as a general-purpose output pin. (yes or no)

12-16 Indicate those lines that are buffered by Schmitt triggers.
- **a.** $\overline{\text{DCDA}}$
- **b.** $\overline{\text{DCDB}}$
- **c.** $\overline{\text{DTRA}}$
- **d.** $\overline{\text{DTRB}}$

12-17 Refer to the generalized SIO block diagram of Fig. 12-1 and indicate true or false.
- **a.** ___ Modem control signals are general-purpose and can be used for functions other than controlling modems.
- **b.** ___ The interrupt control logic provides automatic interrupt vectoring capabilities.
- **c.** ___ Channels A and B have equal priorities in the SIO.
- **d.** ___ The external status has higher priority than the transmit status.
- **e.** ___ The transmit status has higher priority than the receive status.
- **f.** ___ The receive status has the highest priority.

12-18 Refer to the read cycle timing in Fig. 12-9. The read cycle is initiated by
- **a.** The inactive status of $\overline{\text{M1}}$ and the $\overline{\text{RD}}$ and the active status of $\overline{\text{IORQ}}$.
- **b.** The active states of $\overline{\text{M1}}$ and the inactive states of $\overline{\text{RD}}$ and $\overline{\text{IORQ}}$.
- **c.** The inactive status of $\overline{\text{IORQ}}$ and the active status of $\overline{\text{CE}}$ and $\overline{\text{M1}}$.
- **d.** The inactive status of $\overline{\text{M1}}$ and the active status of $\overline{\text{IORQ}}$ and $\overline{\text{RD}}$.

12-19 Refer to Fig. 12-11 and indicate true or false for the SIO interrupt acknowledge cycle.
- **a.** ___ The IEO of any device whose interrupt request is being serviced, or pending, is in the high state.
- **b.** ___ All interrupt signals are prevented from changing while $\overline{\text{M1}}$ is low.
- **c.** ___ When $\overline{\text{IORQ}}$ is low, the highest-priority device requesting interrupt service places its interrupt vector on the data bus.

12-20 Refer to the receive data path block diagram of Fig. 12-2. If the incoming character is 5 or 6 bits long and is transmitted in the asynchronous mode, then
- **a.** It enters the 3-bit buffer and is loaded into the 8-bit receive shift register.
- **b.** It enters the 3-bit buffer, moves into the sync register zero delete and is then loaded into the receive shift register.

12-21 Refer to Fig. 12-2 and indicate true or false for the asynchronous transmission mode.
- **a.** ___ If the incoming word is 7 bits long plus parity, the parity bit is stripped.
- **b.** ___ The receive logic automatically inserts 0s when a character of less than 8 bits is received.
- **c.** ___ If the incoming character is 7 bits long, it bypasses the 3-bit buffer.
- **d.** ___ The output of the receive shift register is always an 8-bit parallel word.

12-22 From the statements below pick out those that apply to the synchronous receive mode.
- **a.** The sync pattern is stored only in WR6 for the bisync synchronization character.
- **b.** The data character is captured and synchron-

ized only when a match is achieved between the two SYNC patterns and the contents of WR6 and WR7.

c. The SIO first goes through a HUNT phase regardless of whether the monosync or the bisync mode is used for character synchronization.

12-23 Indicate true or false for the CRC logic shift register.

a. ____ It is a serial-input/parallel-output shift register.

b. ____ It is a parallel-input/parallel-output shift register.

c. ____ It is a serial-input/serial-output shift register.

d. ____ It is a parallel-input/serial-output shift register.

12-24 Fill in the blanks for the CRC codes.

a. In error-free serial transmissions _____ CRC codes are generated in the receiver and the transmitter. (same; different)

b. When a match between the two codes is obtained, the receiver serial shift register will contain _____. (transmitter-generated CRC code; receiver-generated CRC code; all 0s)

12-25 Refer to the CRC generator of Fig. 12-15 and indicate true or false.

a. ____ The register may contain more than 16 bit positions.

b. ____ The MSB segment and the LSB segment must contain the same number of bit positions.

c. ____ It is possible to insert more than two exclusive-OR gates in the register.

d. ____ The completed CRC code is always shifted out of the register (for serial transmission) from the LSB end.

13

Z80 DUAL ASYNCHRONOUS RECEIVER/TRANSMITTER CHIP

CHAPTER OBJECTIVES

The objectives of this chapter are as follows:

1. To introduce students to the programmable serial, dual-channel, asynchronous receiver/transmitter chip (DART).
2. To present the principal features of the DART chip.
3. To present and describe the generalized architecture of the receive and transmit data paths for asynchronous operations.
4. To present the package pin assignments and functions in convenient tabular form.
5. To discuss the DART timing involved in the read, write, interrupt acknowledge, and return from interrupt cycles.
6. To present and explain the formats and functions of the write registers involved in the initialization of the chip.
7. To present and explain the formats of the three read registers in this chip.

TEXTBOOK REFERENCES

For reviewing the material in the textbook relevant to the topics covered in this chapter, the following chapters and/or sections are suggested:

1. For general discussion of serial I/O transfers — Chapter 11
2. For general discussion of programmable I/O transfers — Chapter 12
3. For description of the serial, asynchronous channels — Sec. 12-2.1.1
4. For multiple buffering of data — Sec. 12-2.1.4
5. For error indications in serial, asynchronous channels — Sec. 12-2.1.5
6. For initialization of the serial, asynchronous channels — Sec. 12-2.1.6
7. For general operation and block diagram of the serial, asynchronous channels — Sec. 12-2.1.8

13-1 INTRODUCTION

The dual asynchronous receiver/transmitter (DART) chip provides two programmable, full-duplex independent channels which have separate control and status lines for interfacing the Z80 CPU with modems and other devices which require serial digital communication capabilities. Fundamentally, the DART functions as a serial-to-parallel and parallel-to-serial converter and controller. Under program control, the DART can be reconfigured to

operate under different conditions and modes so that it can be interfaced with I/O devices having different word lengths and a variety of operating speeds.

This chip can monitor modem status. All incoming words are quadruple buffered, and the transmitter data are double buffered. In the Z80 μC system the DART can be interfaced with either the CPU or the DMA chip. Also, it can be used in the interrupt daisy chain and can provide interrupt vectors without any additional logic.

13-2 PRINCIPAL FEATURES OF THE DART

1. The DART is a third-generation chip which uses the N-channel depletion-mode silicon-gate technology.
2. A single 5-volt ±5% power supply is required.
3. A single-phase, 5-volt clock is used.
4. All inputs and outputs are fully TTL compatible.
5. The chip operates in a temperature environment of 0 to 70°C.
6. The DART can operate with 5, 6, 7, or 8 bits per character and odd parity, even parity, or no parity at all.
7. Parity, framing, and overrun errors can be detected and the DART can operate in four clock modes, ×1, ×16, ×32, and ×64.
8. With a 2.5-MHz clock the data rates are 0–500K bits/second in the ×1 clock mode. With a 4.0-MHz clock the data rates are 0 to 800K bits/second, also in the ×1 clock mode.

13-3 THE DART ARCHITECTURE

13-3.1 The Generalized DART Block Diagram

1. Figure 13-1 shows the generalized block diagram of the DART. On the CPU side, the chip uses the 8-bit data bus for transfer of data between the CPU and the DART as well as the transfer of command words from the CPU to the DART. Control signals are sent to the DART on a 7-bit control bus.
2. The internal control logic and the interrupt control logic blocks essentially perform the same functions that were previously described for the PIO chip in Chapter 6 and the CTC chip in Chapter 8.
3. Each channel has its own group of read/write registers which include six 8-bit write registers and three 8-bit read registers. Each channel also has discrete control logic and status logic which makes it possible for the DART to communicate with modems or other I/O devices.
4. For both channels the write registers are designated WR0–WR5 and the read registers are designated RR0–RR1. The function performed by each register follows:

Write Registers

WR0 Contains register pointers and initialization commands for the different modes.

WR1 Defines receive/transmit interrupts and the data transfer mode.

WR2 Contains the channel B interrupt vector (not channel A)

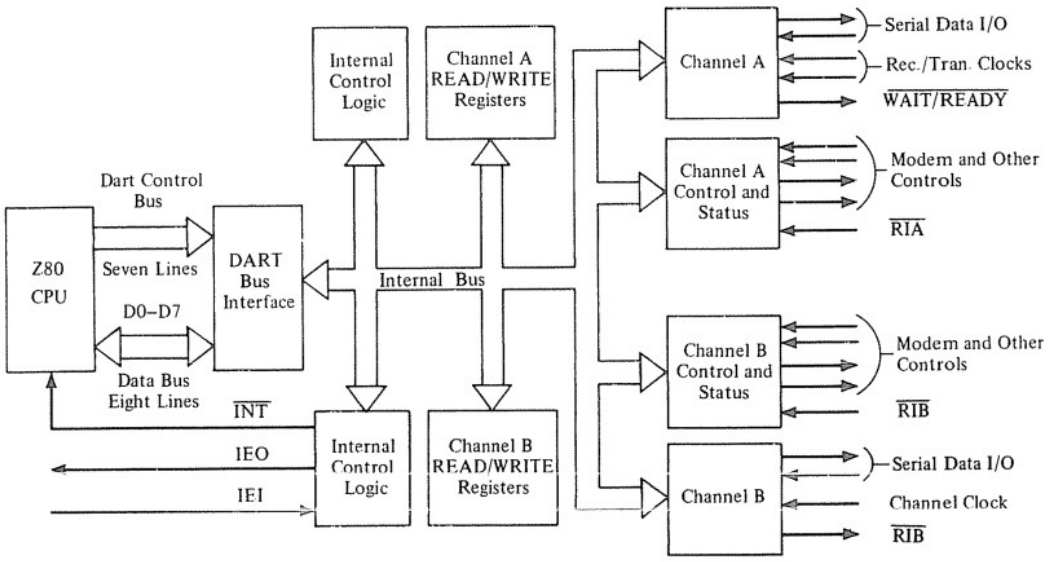


Figure 13-1 Generalized DART Block Diagram

WR3 Contains receiver parameters and controls.
WR4 Contains various miscellaneous parameters for different receive/transmit modes.
WR5 Contains parameters and controls for the transmit operation.

Read Registers

RR0 Holds the buffer status, interrupt status, and external status words for the receive/transmit operations.
RR1 Contains the status for the special receive condition.
RR2 Contains the channel B modified interrupt vector (not channel A).

5. In both channels the associated logic formats synchronize and validate the data transferred in and out of the channel.
6. The modem control signals, for both channels, are general-purpose and can be used for functions other than controlling the modems.
7 The interrupt control logic provides the automatic interrupt vectoring capabilities by determining which channel, or which I/O device within that channel, has the highest priority.
8. In the DART, channel A has a higher priority over channel B. Also, the receive status, the transmit, and the external status interrupts are assigned priorities in that order, within each channel.

13-3.2 The Receive Data Path

1. Figure 13-2 is a simplified block diagram of the path taken by the incoming data in the DART.
2. The incoming data word is an asynchronously transmitted serial bit stream in which the word length could be from 5 to 8 bits. The following description shows the data path followed by this bit stream.
3. The serial data stream enters the DART chip via the RxDA line and goes through a 1-bit time delay.
4. From this point, the data bit stream can proceed in one of two possible paths, depending on the word length. If the character is 5 or 6 bits long, it enters the 3-bit buffer and is then loaded into the 8-bit receive shift register.
5. If the incoming character is 7 or 8 bits long, it bypasses the 3-bit buffer and is directly loaded into the 8-bit receive shift register.
6. The reason for the preceding two procedures is that the receive shift register is an 8-bit serial-in/parallel-out shift register. The output of this register is always an 8-bit parallel word regardless of the number of bits

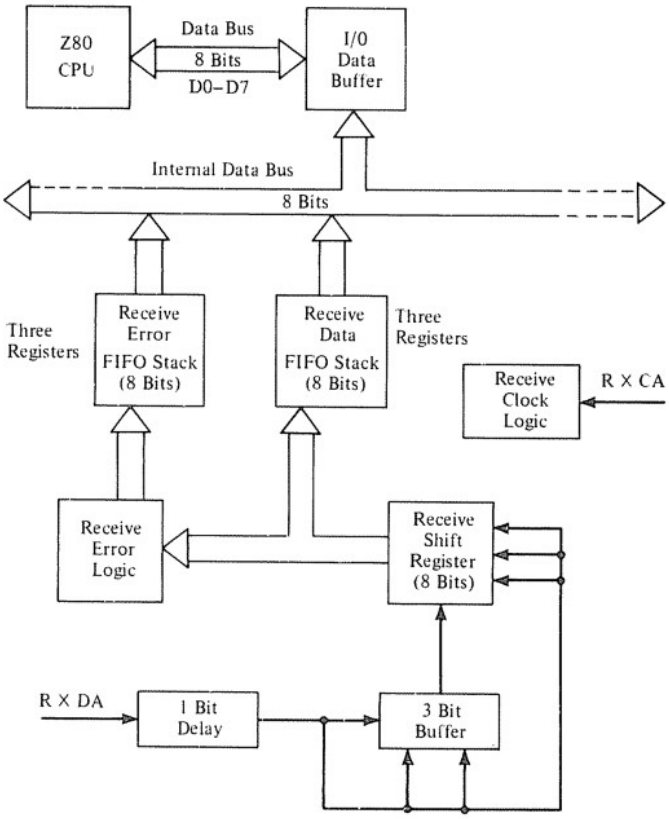


Figure 13-2 Block Diagram of the DART Receive Data Path

in the incoming word. To accomplish this goal, the DART is designed to operate as follows:

a. The receiver logic automatically inserts 1s when a character length of less than 8 bits is used in the incoming serial bit stream.
b. If the parity is enabled and the incoming word length is 8 bits, then the parity bit is stripped from the word and only the 8 data bits are converted into a parallel format.
c. If the incoming word is 7 bits long plus parity, then the parity bit is not stripped but carried along.
d. If the incoming word is 5 or 6 bits long, then the 3-bit buffer inserts 1, 2, or 3 bits (binary 1s), depending on whether the word is 5 or 6 bits and whether a parity bit is included or not.

7. The artificially inserted 1 bits are always in the higher significant bit positions of the parallel formatted word. For example, if the incoming word were 5 bits long and had a parity bit, then the assembled word in the receive shift register would look as shown in Fig. 13-3.
8. The 8-bit word is then sent to the three-register receiver data stack which operates on a first-in, first-out (FIFO) basis. From this stack the 8-bit word is

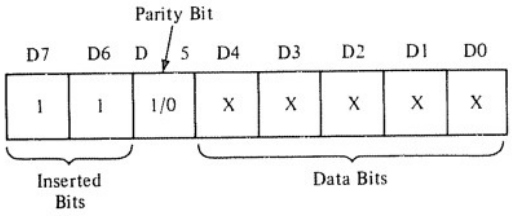


Figure 13-3 Artificially Inserted Bits in Parallel Bit Stream

transferred to the Z80 CPU via the internal data bus, the I/O data buffer, and the CPU data bus (D0–D7).

9. Notice that the 8-bit parallel data word is quadruple buffered, since it passes through the three FIFO registers and the receive shift register. This time delay is provided so that the CPU has ample time to service any interrupts that may come up while the DART keeps accepting and formatting the incoming data words.
10. The formatted 8-bit word is also sent to the receive error logic. This logic checks for parity, framing, and overrun errors and generates error status word. This word is then sent to the three-register 8-bit receive error stack. This is also a FIFO stack.
11. Thus, each data word is quadruple buffered and its corresponding error status word is likewise quadruple buffered, and for the same reason. Since both stacks read out to the internal data bus, correlation between the two is maintained by first reading out the error status word and then the corresponding data word. A special interrupt vector is generated if an error is indicated by the status word.

13-3.3 The Transmit Data Path

1. Figure 13-4 is a simplified block diagram of the path taken by the serial data going out from the DART.
2. The 8-bit parallel output word from the Z80 CPU comes to the DART on the D0–D7 data bus and is then loaded into the I/O data buffer.
3. From there it is loaded into the 8-bit transmit data register via the internal data bus. The data word is then transmitted out of the DART, as explained below.
4. The serial data words are shifted out at the TxDA terminal at a clock rate equal to 1, 1/16, 1/32, and 1/64 of the clock rate input to the transmit clock logic at the $\overline{\text{TxCA}}$ terminal.
5. The word is automatically formatted by the logic in the DART which adds the start bit, the parity bit (odd, even, or no parity bit, depending on how it is programmed) and the appropriate number of stop bits, also depending on the program.

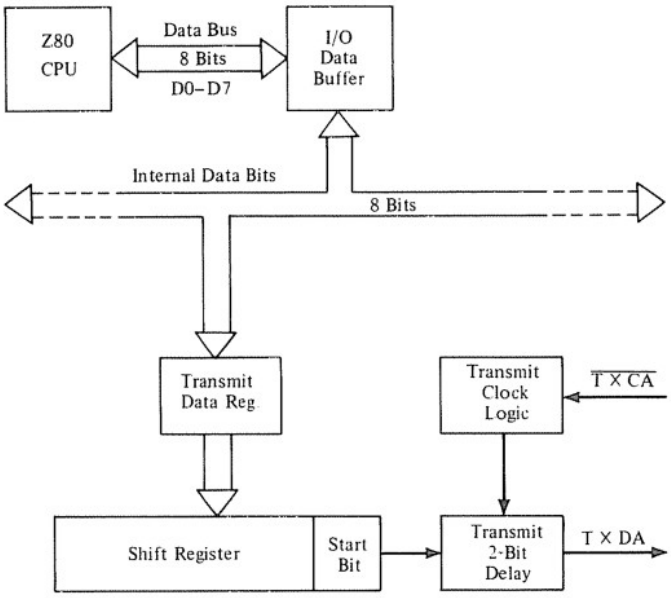


Figure 13-4 Block Diagram of the DART Transmit Data Path

13-4 PACKAGE PIN ASSIGNMENTS AND FUNCTIONS

The DART is packaged in a standard 40-pin DIP package. The various pins and their respective functions, along with their designations, are presented in convenient tabular form. Functionally, the pins can be divided into nine groups.

13-4.1 The CPU Side Pins

1. Figure 13-5 shows the five groups of pins on the CPU side of the DART chip.
2. The pin functions and their designations are shown in Tables 13-1 through 13-5. Some of the lines in these tables may indicate signal flow in one direction and others in the other direction. The function tables clarify these with the following arrow symbols:

   DART → Indicates signals flowing *from the DART*
   DART ← Indicates signals flowing *into the DART*
   DART ↔ Indicates bidirectional flow on the same line(s).

13-4.2 The I/O Side Pins

1. From the presentation in Section 13-4.1 and Table 12-6 we find that 21 of the 40 package pins have been used by the CPU side, leaving 19 pins for the I/O side.
2. The demands of the I/O functions are such that these 19 pins are just not adequate for all I/O situations.

Table 13-1 DATA TRANSFER BUS GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
D0–D7	DART ⟷	8-bit tristate CPU data bus, active high. Used for data and command transfers between the DART and the CPU.

Table 13-2 DART CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
$\overline{CE}$	DART ⟵	Chip enable, active low signal that commands the DART to transmit data during the read cycle. During the write cycle the DART accepts data or command words from the CPU.
$\overline{RESET}$	DART ⟵	Reset signal, active low. It does the following: (1) disables transmitters and receivers, (2) disables all interrupts, (3) forces the modem controls to the high-impedance state, (4) forces TxDA and TxDB marking. After the DART is reset, the control registers must be rewritten before data are received or transmitted.
$\overline{M1}$	DART ⟵	Machine cycle 1 signal, active low. When $\overline{M1}$ and $\overline{RD}$ are simultaneously active the CPU is fetching an instruction from the memory. When $\overline{M1}$ and $\overline{IORQ}$ are simultaneously active, the CPU is acknowledging an interrupt. The DART then places its interrupt vector on the CPU data bus, provided it has the highest priority and it has requested an interrupt.
$\overline{IORQ}$	DART ⟵	I/O request from CPU signal, active low, used for transferring data between CPU and DART as well as for sending commands to the DART. The DART write cycle is initiated by the active status of $\overline{CE}$ and $\overline{IORQ}$ and inactive status of $\overline{RD}$. During the write cycle the CPU writes either data word or the control word into the selected channel as specified by the C/$\overline{D}$ signal. The DART read cycle is initiated by simultaneous activation of the $\overline{CE}$, $\overline{IORQ}$, and $\overline{RD}$ signals. During the read cycle, the DART channel selected by the B/$\overline{A}$ signal transfers the data word to the CPU via the data bus.
$\overline{RD}$	DART ⟵	Read cycle status, active low. When used simultaneously in conjunction with $\overline{CE}$ and $\overline{IORQ}$, it signals the DART to transfer data to the CPU. During the write cycle, $\overline{RD}$ is inactive while $\overline{CE}$ and $\overline{IORQ}$ are active.

Table 13-3 INTERRUPT CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
$\overline{INT}$	DART ⟶	Interrupt request. The $\overline{INT}$ is pulled low by the DART chip when it is requesting an interrupt.
IEI	DART ⟵	Interrupt enable in, active high. Used in the system-wide priority interrupt daisy chain. A high or active signal indicates that no other device with higher priority is currently being serviced.
IEO	DART ⟶	Interrupt enable out, active high. Also used in the interrupt priority daisy chain. It is active only when the IEI is active and the CPU is not servicing an interrupt from the DART. When inactive, it blocks interrupt signals from lower-priority I/O devices while the CPU is servicing an I/O device having a higher priority.

Table 13-4 SELECT CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
C/$\overline{D}$	DART ⟵	Control data select signal. In the high state it informs the DART that the word output by the CPU on the data bus must be accepted by the selected channel as a command or control word. When in the low state, the selected channel will interpret and accept the word on the data bus as a data word.
B/$\overline{A}$	DART ⟵	Channel select signal. When high, channel B is selected and when low channel A is accessed. Transfers are then made between the CPU and the selected channel. This signal is often transmitted to the SIO on the A0 line of the address bus from the CPU.

Table 13-5 POWER SUPPLY AND CLOCK GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
0	DART ⟵	Single-phase TTL-level clock input
+5V	DART ⟵	This is the required single power supply of 5 volts ±5%.
GND	DART ⟵	This is the power supply ground.

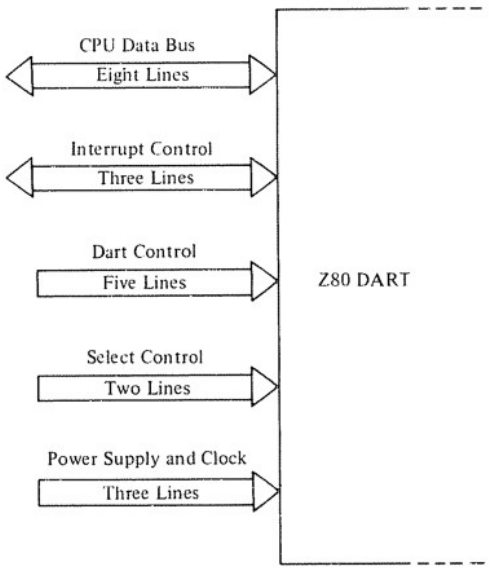


Figure 13-5 Block Diagram of the DART Package Pin Groups on the CPU Side

Table 13-6 CPU SIDE PIN NUMBER ASSIGNMENTS

Pin Designations	*Pin Numbers*	*Pin Designations*	*Pin Numbers*
CPU Data Bus		*Interrupt Control*	
D0	40	$\overline{INT}$	5
D1	1	IEI	6
D2	39	IEO	7
D3	2		
D4	38	*Select Control*	
D5	3		
D6	37	C/$\overline{D}$	33
D7	4	B/$\overline{A}$	34
SIO Control		*Power Supply and Clock*	
$\overline{CE}$	35		
$\overline{RESET}$	21	ø	20
$\overline{M1}$	8	+5V	9
$\overline{IORQ}$	36	GND	31
$\overline{RD}$	32		

Some compromises have to be made. In the DART, these compromises are made such that two signals from the chip are internally bonded together to one package pin.

3. Figure 13-6 is the block diagram of the DART package pin groups on the I/O side. Notice that the data transfer and clock group has five lines for channel A, whereas the same group for channel B has only four lines. This is where the compromise mentioned in paragraph 2 comes in. Pin 27 has two signals bonded on it internally from the DART chip.
4. The package pin assignments for channel A remain the same, as shown in Table 13-9, regardless of which configuration is used for channel B.
5. Because of the previously mentioned limitations imposed by the availability of only 40 pins on the package, a sacrifice is made in channel B signals, namely, receive clock and transmit clock signals. These two signals are internally bonded to pin 327 of the package. This means that channel B can only be used as either a receiver or a transmitter at any one time, not as a full duplex channel. The pin configurations are shown in Tables 13-10 and 13-11.

Table 13-7 DATA TRANSFER AND CLOCK GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
RxDA, RxDB	DART ⟵	Receive data, active high. Serial data, TTL compatible, signals are input to the DART.
$\overline{\text{RxCA}}$, $\overline{\text{RxCB}}$	DART ⟵	Receive clocks, active low. These clock inputs may be driven by the Z80 CTC timer, discussed in Chapter 8, for the programmable generation of the baud rate. The receiver clocks could be 1, 16, 32, or 64 times the incoming data rate. Incoming data are sampled on the positive-going edge of the $\overline{\text{RxC}}$ signal. Both inputs are buffered by Schmitt triggers.
TxDA, TxDB	DART ⟶	Transmit data, active high. Serial data TTL compatible signals are output by the DART.
$\overline{\text{TxCA}}$, $\overline{\text{TxCB}}$	DART ⟵	Transmitter clocks, active low. These clock inputs may be driven by the Z80 CTC timer, discussed in Chapter 8, for the programmable generation of the baud rate. The transmitter clocks could be 1, 16, 32, or 64 times the outgoing data rate, but the clock multiple factor for the receiver and the transmitter must be the same. The transmitted data (TxD) changes states at the negative-going edge of the TxC pulse. Both inputs are buffered by Schmitt triggers.
$\overline{\text{W/RDYA}}$, $\overline{\text{W/RDYB}}$	DART ⟶	Wait/ready signal lines. The lines are open (drain open) when they are programmed for the WAIT function. They can be either in the high or the low state when programmed for the READY function. As a WAIT function they are used to synchronize the CPU to the DART data transfer rate. When used with a DMA controller, they can be used for the READY function.

Table 13-8 MODEM CONTROL GROUP

Pin Designations	*Signal Flow Directions*	*Functions*
$\overline{\text{RTSA}}$, $\overline{\text{RTSB}}$	DART ⟶	Ready to send, active low. This signal goes low when the RTS bit is set. When the RTS bit is reset, the output line goes high after the transmitter is empty.
$\overline{\text{CTSA}}$, $\overline{\text{CTSB}}$	DART ⟵	Clear to send, active low. If programmed as Auto Enables, low signal inputs on these lines enables the corresponding transmitter. Otherwise, they can be programmed and used as general-purpose inputs. Logic level transitions in either direction on these lines are detected by the DART which then sends an interrupt request to the CPU. Both lines are buffered with Schmitt triggers.
$\overline{\text{DTRA}}$, $\overline{\text{DTRB}}$	DART ⟶	Data terminal ready, active low. These lines follow the DTR bit state as determined by the program. They can also be used as general-purpose outputs.
$\overline{\text{DCDA}}$, $\overline{\text{DCDB}}$	DART ⟵	Data carrier detect, active low. If the DART is programmed for auto enables, these lines function as receive enables. Otherwise, they can be used as general-purpose inputs. Logic level transitions in either direction in these lines are detected by the DART, which then sends an interrupt request to the CPU. Both lines are buffered with Schmitt triggers.
$\overline{\text{RIA}}$, $\overline{\text{RIB}}$	DART ⟵	Ring Indicator inputs, active low. The two inputs are similar to the clear to send (CTS) and the data carrier detect (DCD) inputs. Both logic level transitions are detected by the DART and interrupts sent to the Z80 CPU. These inputs can also be used as general-purpose inputs.

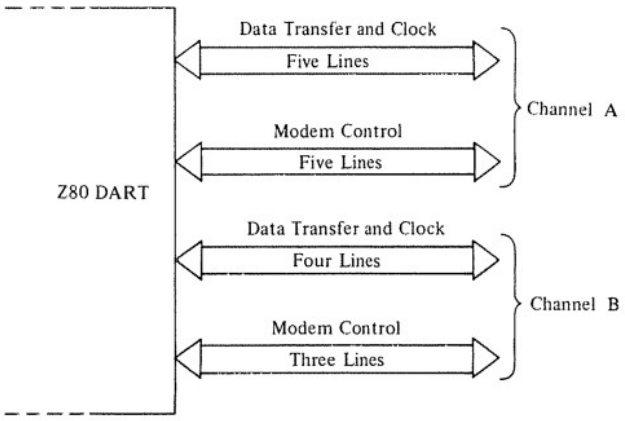


Figure 13-6 Block Diagram of the DART Package Pin Groups on the I/O Side

Table 13-9 CHANNEL A PIN NUMBER ASSIGNMENTS

Pin Designations	*Pin Numbers*	*Pin Designations*	*Pin Numbers*
Data Transfer and Clock		*Modem Control*	
RxDA	12	$\overline{\text{RTSA}}$	17
$\overline{\text{RxCA}}$	13	$\overline{\text{CTSA}}$	18
RxDA	15	$\overline{\text{DTRA}}$	16
$\overline{\text{TxCA}}$	14	$\overline{\text{DCDA}}$	19
$\overline{\text{W/RDYA}}$	10	$\overline{\text{RIA}}$	11

Table 13-10 CHANNEL B PIN NUMBER ASSIGNMENTS FOR RECEIVER OPERATIONS

Pin Designations	*Pin Numbers*	*Pin Designations*	*Pin Numbers*
Data Transfer and Clock		*Modem Control*	
RxDB	28	$\overline{\text{RTSB}}$	24
$\overline{\text{RxCB}}$	27	$\overline{\text{CTSB}}$	23
TxDB	26	$\overline{\text{DTRB}}$	25
$\overline{\text{W/RDYB}}$	30	$\overline{\text{DCDB}}$	22
		$\overline{\text{RIB}}$	29

13-5 THE DART TIMING

In this section we briefly describe and discuss the DART timing charts involved in the read, write, interrupt acknowledge, and the return from interrupt cycles. The frequency and phase relationship of the DART clock must be the same as that of the CPU clock.

Table 13-11 CHANNEL B PIN NUMBER ASSIGNMENTS FOR TRANSMITTER OPERATIONS

Pin Designations	*Pin Numbers*	*Pin Designations*	*Pin Numbers*
Data Transfer and Clock		*Modem Control*	
RxDB	28	$\overline{\text{RTSB}}$	24
$\overline{\text{TxCB}}$	27	$\overline{\text{CTSB}}$	23
TxDB	26	$\overline{\text{DTRB}}$	25
$\overline{\text{W/RDYB}}$	30	$\overline{\text{DCDB}}$	22
		$\overline{\text{RIB}}$	29

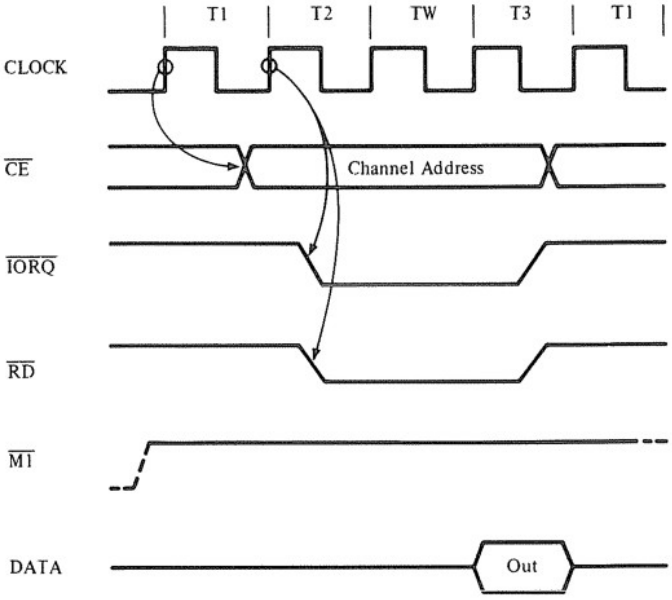


Figure 13-7 DART Read Cycle Timing

13-5.1 The Read Cycle

Figure 13-7 shows the timing relationships during a CPU input instruction when the CPU requests a data or status word from the DART. Note that the read cycle is initiated by the active status of $\overline{\text{IORQ}}$ and $\overline{\text{RD}}$ and the inactive status of $\overline{\text{M1}}$. The wait state TW is automatically inserted.

13-5.2 The Write Cycle

Figure 13-8 shows the timing relationships during a CPU output instruction when the CPU writes a data word or a control byte in the DART chip. Notice that the write cycle is initiated by the active status of $\overline{\text{IORQ}}$ and the inactive status of $\overline{\text{RD}}$ and $\overline{\text{M1}}$. The wait state TW is automatically inserted.

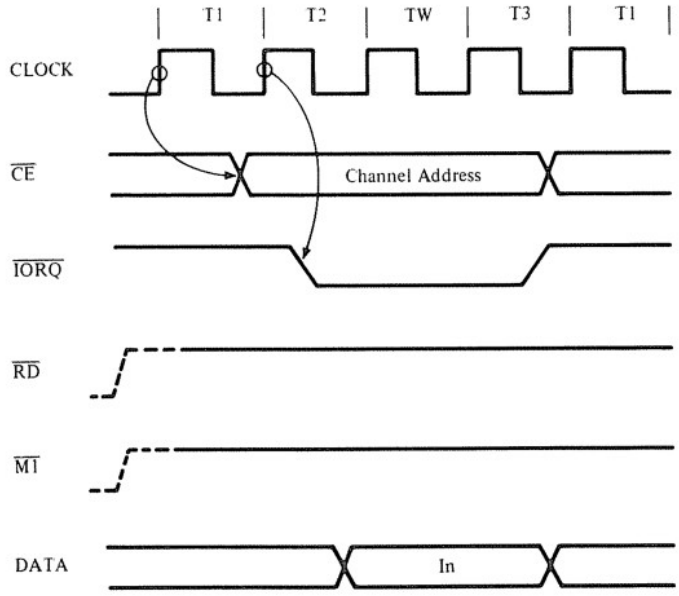


Figure 13-8 DART Write Cycle Timing

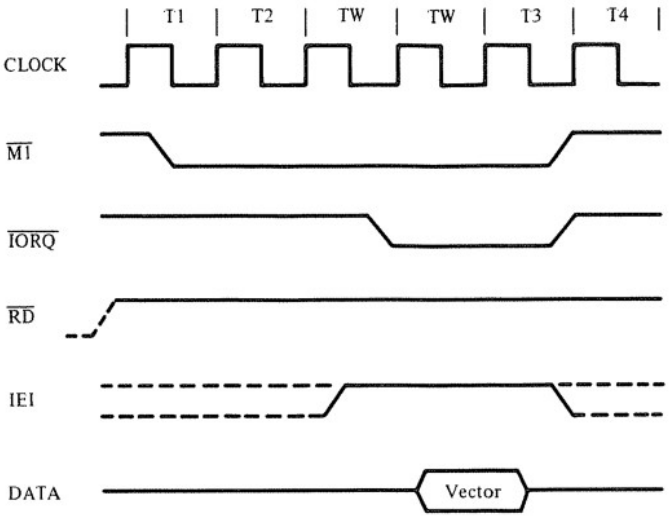


Figure 13-9 DART Interrupt Acknowledge Cycle

13-5.3 The Interrupt Acknowledge Cycle

1. An interrupt request to the CPU is indicated by the $\overline{INT}$ line going low.
2. Sometime thereafter the CPU acknowledges the interrupt by the $\overline{M1}$ line going low and, a few clock periods later, by the $\overline{IORQ}$ going low.
3. The circuits in the daisy chain then determine the interrupt priority of the requesting device. The IEI line of the highest-priority device is high.
4. The IEO of any device whose interrupt request is being serviced, or is pending, is in the low state.
5. The IEO of devices with no pending interrupts, or not currently being serviced, have IEO = IEI.
6. All interrupt signals are prevented from changing while $\overline{M1}$ is low. This ensures stability in the daisy chain.
7. When $\overline{IORQ}$ is active, the highest-priority device requesting interrupt service places its interrupt vector on the data bus and latches in its interrupt request.
8. The timing relationships are shown in Fig. 13-9.

13-5.4 The Return from Interrupt Cycle

1. A return from interrupt instruction (RETI) is inserted at the end of every interrupt service subroutine.
2. RETI is a 2-byte op code (ED-4D). It resets the interrupt-under-service flip-flop, thereby ending the interrupt that has just been processed.
3. Then the first byte of the RETI (i.e., ED) is decoded and the daisy chain forces the IEO high of any interrupt that has not been acknowledged. Thus, the daisy chain asserts that the device that is currently being serviced is the only one with a high IEI and a low IEO.
4. The timing relationships for this cycle are shown in Fig. 13-10.

13-6 COMMAND WORDS FOR DART INITIALIZATION

In this section we describe and discuss the formats of the various command bytes that are used for initializing the DART chip. The words or bytes define the different modes and functions of the DART operations. The chip has six write registers, labeled WR0–WR5. For proper initialization it is necessary that each command word first be preceded by the pointer word, as described below. Additionally, there are three read registers that basically contain status words or a vector. They are labeled RR0–RR2

13-6.1 Write Command Words

13-6.1.1 The WRO Register The bit format for this register is shown in Fig. 13-11. This word has three fields. Bits D7 and D6 are not used and they can be any bits. Bits D0–D2 are pointers for the other five registers. Since only one register identification code can be inserted in these three positions, this word must be used first to

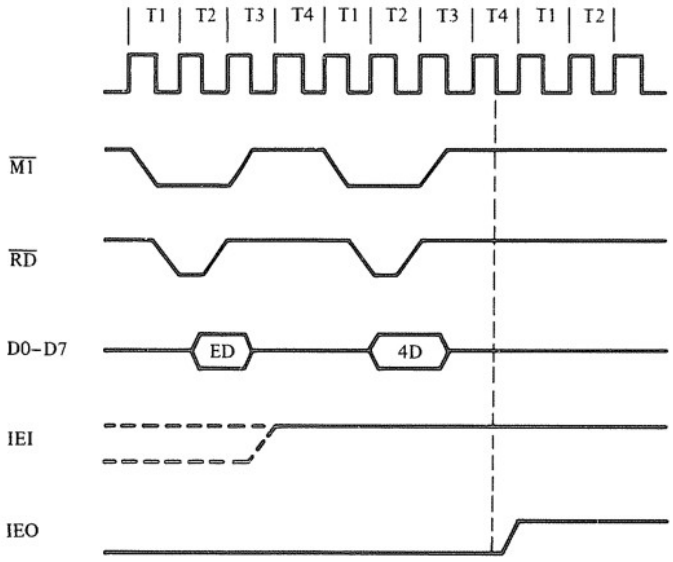


Figure 13-10 DART Return from Interrupt Cycle

Not Used		Basic Command Bits			Register Pointer Bits		
D7	D6	D5	D4	D3	D2	D1	D0

Figure 13-11 Write Register WR0 Format (Pointers/Commands/Modes)

identify each individual register separately. The codes for these registers follow.

D2	*D1*	*D0*	
0	0	0	Register 0
0	0	1	Register 1
0	1	0	Register 2
0	1	1	Register 3
1	0	0	Register 4
1	0	1	Register 5
1	1	0	Invalid
1	1	1	Invalid

Bits D3–D5 contain the codes for the basic commands. These codes and their commands are shown below in Table 13-12.

13-6.1.2 The WR1 Register

The bit format for this register is shown in Fig. 13-12. This word contains the control bits for the various interrupt and wait/ready modes. The codes for the receive interrupt are defined in bits D3 and D4 according to the following:

D4	*D3*	
0	0	Receiver interrupts disabled.
0	1	Receiver interrupts only on first character.
1	0	Interrupts on all character received (Note: parity error is a condition that is treated as a special receive).
1	1	Interrupts on all characters received (Note: parity error is not treated as a special receive condition).

Bit D0 *External interrupt enable.* Honors externally generated interrupts.

Bit D1 *Transmitter interrupt enable.* Interrupts are generated when the transmitter buffer is empty.

Bit D2 *Status affects vector.* Active only in channel B. When set to 1, the fixed vector is returned when interrupt is acknowledged. When reset to 0, the vector returned when interrupt is acknowledged is variable according to the following Table 13-13.

Bit 7 *WAIT/READY enable.* The WAIT/READY function is enabled by setting this bit to a 1.

Table 13-12 BASIC COMMAND CODES FOR WR0

D5	*D4*	*D3*	*Explanation*
0	0	0	*Null command*—has no effect on the chip.
0	0	1	Not used in this chip.
0	1	0	*Reset external/status interrupts.* The status bits of RR0 are latched after an interrupt. This code resets them and so interrupts are honored again.
0	1	1	*Channels reset.* This code resets a single channel, after an external interrupt, including prioritization logic.
1	0	0	*Enable interrupt on next receive character.* When the Interrupt on First Receive Character is used, this mode is reactivated after each complete message.
1	0	1	*Reset transmitter interrupt pending.* When no characters are to be transmitted this code prevents additional transmitter interrupts.
1	1	0	*Error reset.* This code resets error latches, including Parity and Overrun errors latched in Read Register RR1.
1	1	1	*Return from interrupt (channel A only).* This code resets the interrupt-under-service latch of the highest-priority internal device.

D7	D6	D5	D4	D3	D2	D1	D0
Wait/ Ready Enable	Wait / Ready Function	Wait/ Ready on Receive/ Transmit	Interrupt Mode Definition		Status Affects Vector	Transmit Interrupt Enable	External Interrupt Enable

Figure 13-12 Write Register WR1 Format (Interrupt/Data Transfers)

Table 13-13 VARIABLE VECTOR CODES

V3	*V2*	*V1*	
0	0	0	Channel B transmit buffer empty.
0	0	1	Channel B external status change.
0	1	0	Channel B receive character available.
0	1	1	Channel B special receive condition. (parity, overrun, framing errors)
1	0	0	Channel A transmit buffer empty.
1	0	1	Channel A external/status change.
1	1	0	Channel A receive character available.
1	1	1	Channel A special receive condition. (parity, overrun, framing errors)

Bit 6 *WAIT/READY function.* This function can only be selected when bit D7 is set to a 1. When D6 = 0, the READY function is selected and the WAIT function is selected when D6 = 1. When the WAIT function is selected, the WAIT/READY output switches from high to low.

13-6.1.3 The WR2 Register The bit format of the WR2 register is shown in Fig. 13-13. This word is the interrupt vector for channel B only. In case of an interrupt, bits V0 and bits V4–V7 are always returned to the CPU unchanged. But bits V1–V3 may be altered, depending on whether bit D2 in WR1 is a 1 or a 0. If D2 = 1 in WR1, then bits V1–V3 in WR 3 are modified as previously explained in the WR1 description.

13-6.1.4 The WR3 Register This word contains bits for the parameters and logic control the receiver section of the DART. The format of this register is shown in Fig. 13-14.

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	D3	D2	D1	D0

Figure 13-13 Write Register WR2 Format (Channel B Vector)

D7	D6	D5	D4	D3	D2	D1	D0
Receiver Bits Per Character		Auto Enable	Not Used				Receiver Enable
			X	X	X	X	

Figure 13-14 Write Register WR3 Format (Receive Parameters/Control)

Bit D0 *Receiver enable.* A1 in this bit position permits the receive operation to start.

Bits D1–D4 *Not used.*

Bit D5 *Auto enables.* When D5 = 1, lines $\overline{\text{DCDA}}$, $\overline{\text{DCDB}}$, $\overline{\text{CTSA}}$, and $\overline{\text{CTSB}}$ become receiver and transmitter enables, respectively.

Bits D6 and D7 *Receiver bits per character.* These 2 bits define the number of bits per character received as follows:

D7	*D6*	*Bits per Character*
0	0	5
0	1	7
1	0	6
1	1	8

13-6.1.5 The WR4 Register The bit format of the WR4 register is shown in Fig. 13-15. The bits in this word affect both the receiver and the transmitter operations.

Bit D0 *Parity enable.* A 1 in this position enables the parity feature in the DART.

Bit D1 *Parity.* When D1 = 0, odd parity is set. When D1 = 1, the even parity is set.

Bits D3 and D2 *Stop bits per character.* These 2 bits define the number of stop bits per character as follows:

D3	*D2*	*Stop Bits/Character*
0	0	1
0	1	1
1	0	2
1	1	Invalid

Bits D5 and D4 *Not used.*

Bits D7 and D6 *Receive/transmit clock rates.* These two bits specify the multiply factor between the clock and the data rates ($\overline{\text{TxC}}$ and $\overline{\text{RxC}}$).

D7 D6	D5 D4	D3 D2	D1	D0
Receive/ Transmit Clock Rates	Not Used	Number of Stop Bits Per Character	Parity	Parity Enable
0 0 = X 1 0 1 = X 16 1 0 = X 32 1 1 = X 64	X X	0 1 = 1 bit 1 0 = 1½ bits 1 1 = 2 bits 0 0 = Invalid	0 = Odd 1 = Even	1 = Enable 0 = Disable

Figure 13-15 Write Register WR4 Format (Miscellaneous Parameters/Modes)

D7	D6 D5	D4	D3	D2	D1	D0
Data Terminal Ready (DTR)	Transmit Bits Per Character 0 0 = 5 or less 0 1 = 7 1 0 = 6 1 1 = 8	Send Break	Transmit Enable	Not Used X	Request To Send	Not Used X

Figure 13-16 Write Register WR5 Format (Transmit Parameters/Control)

13-6.1.6 The WR5 Register This word controls the operation of the transmitter. The bit format is shown in Fig. 13-16.

Bits D0 and D2 *Not used.*

Bit D1 *Request to send (RTS).* When D1 = 1, $\overline{\text{RTS}}$ goes low.

Bit D3 *Transmit enable.* Data word is transmit only when D3 = 1.

Bit D4 *Send break.* When D4 = 1, it forces the Transmit Data to the spacing condition. When D = 0 $\overline{\text{TxD}}$ returns to marking.

Bit D6 and D5 *Transmit bits per character.* These two bits define the number of bits in the character to be transmitted as follows:

D6	D5	Transmit Bits/Character
0	0	5 or less
0	1	7
1	0	6
1	1	8

Bit D7 *Data terminal ready.* When D7 = 1, the $\overline{\text{DTR}}$ line goes low, and when D7 = 0, it goes high.

13-6.2 The Read Command Words

The DART chip contains three read registers which are used for reading out the status of the two channels. Registers RR0 and RR1 apply to both the channels, whereas register RR2 applies only to channel B. The information readout from these three registers contains error conditions, interrupt vectors, and various usual communications interface handshake signals.

To read out from any of these registers, except RR0, the correct register must be first identified by using the WR0 pointer register followed by an input instruction which will then read out the contents of the addressed register to the CPU.

13-6.2.1 The RR0 Register The RR0 register contains the status of the transmit/receive buffers, the interrupt status, and the DCD and CTS inputs. The bit format of this register is shown in Fig. 13-17.

Bit D0 *Receive character available.* When at least one character is available in the receive FIFO stack, this bit is set. When it is completely empty it is reset to 0.

Bit D1 *Interrupt pending for channel A.* An interrupt condition in channel A causes this bit to be set. It is always 0 in channel B. Used primarily in applications that do not have vectored interrupts.

Bit D2 *Transmit buffer empty.* This bit is set to 1 whenever the transmit buffer becomes empty.

Bit D3 *Data carrier detect.* The transition of the DCD input causes this bit to be latched and causes an external/status interrupt.

Bit D4 *Ring indicator.* The DART detects both the high-to-low and the low-to-high transitions and then interrupts the CPU. This bit indicates these conditions by a 1.

Bit D5 *Clear to send.* This bit is very similar to the Data Carrier Detect (DSD) bit. However, it shows the inverted status of the signal at the CTS pin.

Bit D6 *Not used.*

Bit D7 *Break.* D7 = 1 when a break sequence, i.e., the null character pulls the framing error, is detected in the incoming data stream.

D7	D6	D5	D4	D3	D2	D1	D0
Break	Not Used X	Clear To Send	Ring Indicator	Data Carrier Detect (DCD)	Transmit Buffer Empty	Interrupt Pending for Channel A	Receive Character Available

Figure 13-17 Read Register RRO Format (Buffer/Interrupt/External status)

D7	D6	D5	D4	D3	D2	D1	D0
Not Used	Framing Error	Overrun Error	Parity Error 1 = Error 0 = No error	Not Used			All Sent
X				X	X	X	

Figure 13-18 Read Register RR1 Format (Special Receive Condition Status).

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2	V1	V0

Figure 13-19 Read Register RR2 Format (Modified Interrupt Vector for Channel B).

13-6.2.2 The RR1 Register The special receive condition status bits are included in this register, whose format is shown in Fig. 13-18.

Bit D0 *All sent.* D0 = 1 when all the characters in the transmit buffer have been completely sent from the transmitter.

Bits D1–D3 *Not used.*

Bit D4 *Parity error.* This bit indicates if a parity error is detected in the received character. A 1 by this bit indicates an error.

Bit D5 *Overrun error.* This bit is set if an Overrun Error is detected. This really means that more than three characters have been received without a corresponding read instruction from the CPU.

Bit D6 *Framing error.* This bit is set if a framing error in the received character is detected. To avoid interpreting the framing error as a new start bit, the DART adds an additional one-half of a bit time to the character time.

Bit D7 *Not used.*

13-6.2.3 The RR2 Register The format of the RR2 register, which is used for only register B, is shown in Fig. 13-19. Recall that bits V1–V3 are variable if the status affect vector (bit D2 in register WR1) is set.

13-7 REVIEW QUESTIONS

13-1 Indicate true or false for the DART chip.

a. ___ The DART contains two fully independent simplex communications channels.

b. ___ The DART interfaces the Z80 CPU with modems and other I/O devices which require parallel digital communications capabilities.

c. ___ The DART functions as a serial-to-parallel and parallel-to-serial converter.

d. ___ The DART handles both synchronous and asynchronous protocols.

13-2 In the DART, receiver data are _____ buffered and the transmitter data are _____ buffered. (single; double; triple; quadruple)

13-3 Indicate the data rates at which the DART can operate asynchronously in the ×1 clock mode.

a. _____ With 2.50-MHz system clock.

b. _____ With 4.00-MHz system clock.

(0–880; 0–500; 100–500; 100–800; 0–800; 500–800 kilobits per second)

13-4 Circle the number of bits per character that the DART can handle.

a. 6;
b. 10;
c. 12;
d. 8;
e. 16;
f. 5.

13-5 Answer yes or no for the capabilities of the DART chip.

a. ___ Can it work with two stop bits per character?

b. ___ Is it possible to detect framing and overrun errors?

c. ___ Can it operate at ×8 the clock rate?

d. ___ Can it operate at ×64 the clock rate?

13-6 Indicate true or false for the DART package pin functions.

a. ___ The D0–D7 bus is used for data and command transfers between the CPU and the I/O devices.

b. ___ During the write cycle, the DART accepts data or command words from the CPU.

c. ___ When $\overline{\text{RESET}}$ is active, it enables the transmitter and the receiver.

d. ___ When $\overline{\text{RESET}}$ is active, it disables all interrupts.

e. ___ When $\overline{\text{RESET}}$ is active, it forces the modem controls to the high-impedance state.

13-7 Circle the statements that are applicable to the $\overline{\text{IORQ}}$ signal.

a. Initiates the write cycle when $\overline{\text{CE}}$ is inactive.

b. Initiates the write cycle when $\overline{\text{CE}}$ is active.

c. Initiates the read cycle when $\overline{CE}$ and $\overline{RD}$ are active.
d. During the write cycle, $\overline{IORQ}$ and $\overline{RD}$ are active, while $\overline{CE}$ is inactive.

13-8 Which of the following signals is pulled low by the DART chip when it is requesting an interrupt?
a. IEI
b. $\overline{INT}$
c. IEO
d. $\overline{IORQ}$

13-9 The following statements relate to the IEI and the IEO signals. Indicate true or false.
a. ____ IEI is used in the system-wide priority interrupt daisy chain.
b. ____ When IEI is high, it indicates that a device with higher priority is currently being serviced.
c. ____ IEO is inactive when IEI is active.
d. ____ When IEO is inactive, it blocks interrupt signals from lower-priority I/O devices.

13-10 Which of the following package pins will be active during transmission of several data via channel B?
a. TxDA
b. $\overline{RxCB}$
c. TxDB
d. $\overline{RxCA}$

13-11 The incoming data word into the DART is sampled on the _____ going edge of the signal. (positive; negative; $\overline{RxC}$; RxD)

13-12 Which of the following signals are buffered by Schmitt triggers?
a. RxDA and RxDB
b. $\overline{RxCA}$ and $\overline{RxCB}$
c. TxDA and TxDB
d. $\overline{TxCA}$ and $\overline{TxCB}$

13-13 The following statements apply to the $\overline{RTSA}$, $\overline{RTSB}$, $\overline{CTSA}$, and $\overline{CTSB}$ signals. Indicate true or false.
a. ____ $\overline{RTSA}$ and $\overline{RTSB}$ follow the state of the RTS bit.
b. ____ When the RTS bit is set, $\overline{RTSA}$ and $\overline{RTSB}$ go low.
c. ____ Low signal on $\overline{CTSA}$ or $\overline{CTSB}$ enables the corresponding transmitter.
d. ____ The DART sends an interrupt request to the CPU when the levels of either $\overline{CTSA}$ or $\overline{CTSB}$ change.

13-14 _____ Is it possible to use the DTRA pin as a general-purpose pin? (Yes or no)

13-15 Indicate those lines that are buffered by Schmitt triggers.
a. $\overline{DCDA}$
b. $\overline{DCDB}$
c. $\overline{DTRA}$
d. $\overline{DTRB}$

13-16 Refer to the generalized block diagram of Fig. 13-1 and indicate true or false.
a. ____ Modem control signals are general-purpose and can be used for functions other than controlling modems.
b. ____ The interrupt control logic provides automatic interrupt vectoring capabilities.
c. ____ Channels A and B have equal priorities in the DART.

13-17 Refer to the read cycle timing of Fig. 13-7. The read cycle is initiated by:
a. The inactive status of $\overline{M1}$ and $\overline{RD}$ are the active status of $\overline{IORQ}$.
b. The active states of $\overline{M1}$ and the inactive states of $\overline{RD}$ and $\overline{IORQ}$.
c. The inactive status of $\overline{IORQ}$ and the active status of $\overline{CE}$ and $\overline{M1}$.
d. The inactive status of $\overline{M1}$ and the active status of $\overline{IORQ}$ and $\overline{RD}$.

13-18 Refer to Fig. 13-9 and indicate true or false for the DART interrupt acknowledge cycle.
a. ____ The IEO of any device whose interrupt request is being serviced, or pending, is in the high state.
b. ____ All interrupt signals are prevented from changing while M1 is low.
c. ____ When $\overline{IORQ}$ is low, the highest-priority device requesting interrupt service places its interrupt vector on the data bus.

13-19 Refer to the receive data path block diagram of Fig. 13-2. If the incoming character is 5 or 6 bits long, then
a. It enters the 3-bit buffer and is then loaded into the 8-bit receive shift register.
b. It bypasses the 3-bit buffer and is loaded directly in the 8-bit receive shift register.

13-20 Refer to Fig. 13-2 and indicate true or false for the incoming data characters.
a. ____ If the incoming word is 7 bits long plus parity, the parity bit is stripped.
b. ____ The receive logic automatically inserts 0s when a character of less than 8 bits is received.
c. ____ If the incoming character is 7 bits long, it bypasses the 3-bit buffer.
d. ____ The output of the receive shift register is always an 8-bit parallel word.

APPENDIX A
TIMING DIAGRAM CONVENTIONS

A timing diagram is a pictoral representation of a digital signal on a single line, or a group of lines (a bus), that displays changes in logic states as a function of time. These diagrams represent logic states and transitions between states. They also show time relationships between transitions of several different, individual digital signals. The purpose of this appendix is to give students an understanding of the various timing charts that are commonly used in the digital computer field. The diagrams, their symbology, and explanations of these follow.

A-1 SIGNAL SYMBOLS IN TIMING CHARTS

Refer to Fig. A-1 for explanation of the following symbols.

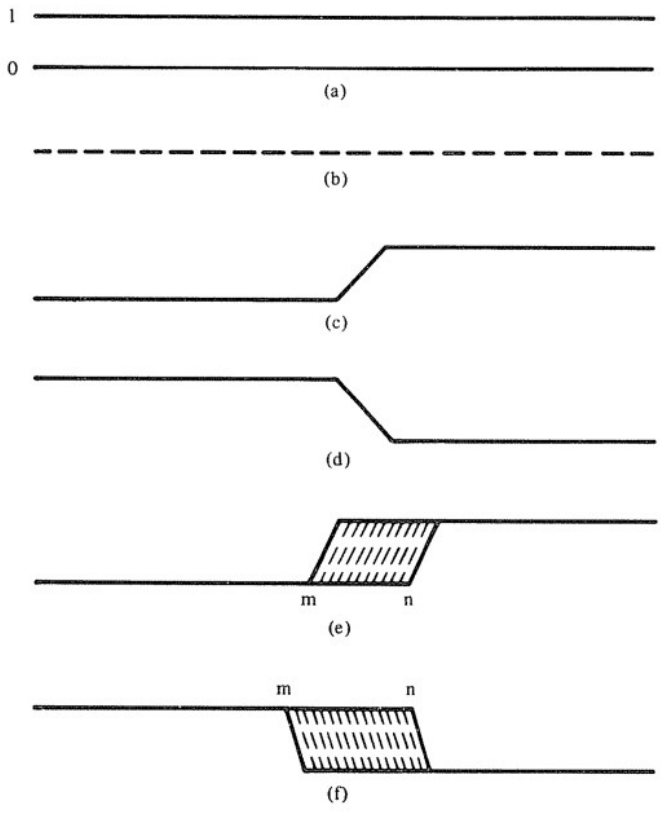


Figure A-1 Timing Chart Symbols

a. A single solid line with either a 1 or a 0 in front of it indicates a steady logical 1 or 0 state.
b. A dotted line indicates a floating signal line. This condition is prevalent when a three-state line is placed in the high impedance state.
c. Indicates a signal transition from a low to a high state (logical 0 to logical 1).
d. Indicates a signal transition from a high to a low state (logical 1 to logical 0).
e. Indicates that a low-to-high transition can take place any time between *m* and *n*.
f. Indicates that a high-to-low transition can take place any time between *n* and *p*.

A-2 BUS CONVENTIONS

Refer to Fig. A-2 for diagrams relating to bus conventions.

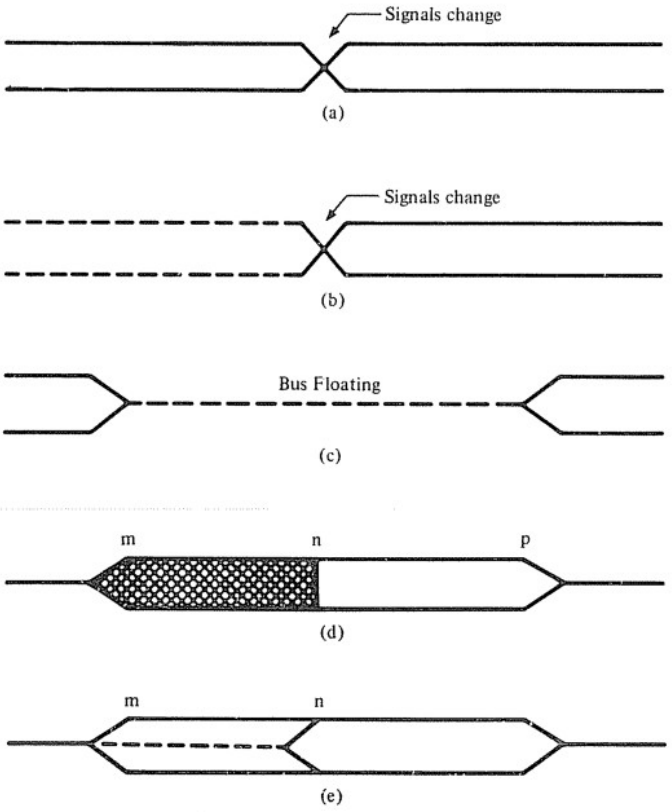


Figure A-2 Bus Conventions

a. Indicates that the signals on the bus lines change at the indicated crossover point.
b. The dotted lines indicate that the signals on the bus lines are not important or relevant until after they change at the crossover point.
c. During the time indicated by the dotted line, the three-state bus is floating, i.e., is in the high-impedance state.
d. The cross-hatched portion of the bus indicates that the signals on the lines are not stable from point m to n but that they are stable between points *n* and *p*.
e. Indicates that the three-state buffers may become active any time between points *m* and *n*.

A-3 TRANSITIONS CAUSED BY TRANSITIONS

The following explanations refer to Fig. A-3.

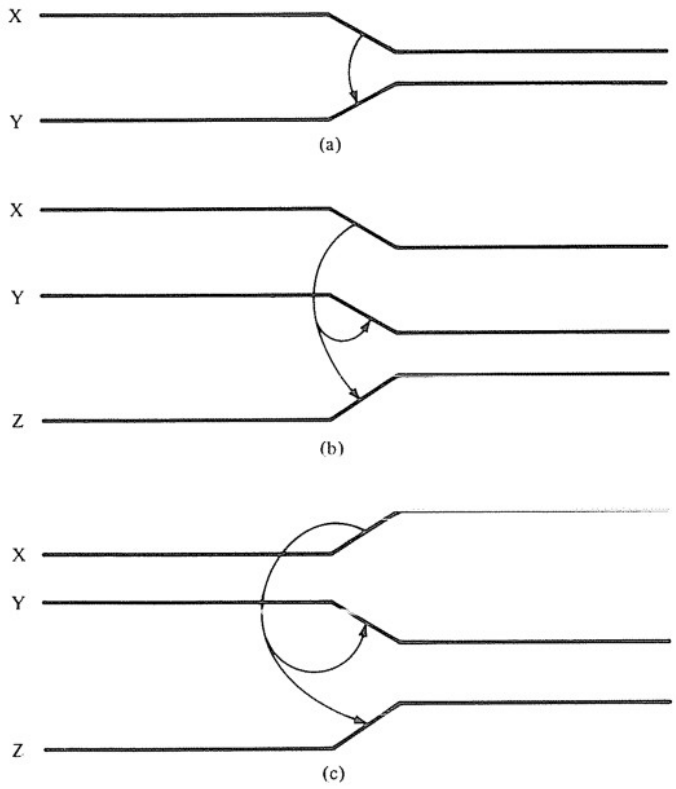


Figure A-3 Transitions Caused by Transitions

a. Indicates that the high-to-low transition in the signal on line *X* causes a low-to-high transition in the signal on line *Y*. Note that the transitions on both lines could be in directions opposite to that shown here.

b. Indicates that the high-to-low transition in line *X* causes simultaneous transitions in lines *Y* and *Z*.
c. Indicates that the low-to-high transition in line *X* causes simultaneous transitions in lines *Y* and *Z*.

A-4 LEVEL CHANGES CAUSED BY SIGNAL TRANSITIONS

Refer to Fig. A-4 for the following explanations.

a. Indicates that the high-to-low transition on line *X* causes a low level on line *Y*.
b. Indicates that the high-to-low transition on line *X* results in a high level on line *Y*.
c. Indicates that low-to-high transition on line *X* causes a low level on line *Y*.
d. Indicates that a low-to-high transition on line *X* causes a high level on line *Y*.

A-5 SIGNAL TRANSITIONS CAUSED BY LEVELS

Refer to Fig. A-5 for explanation.

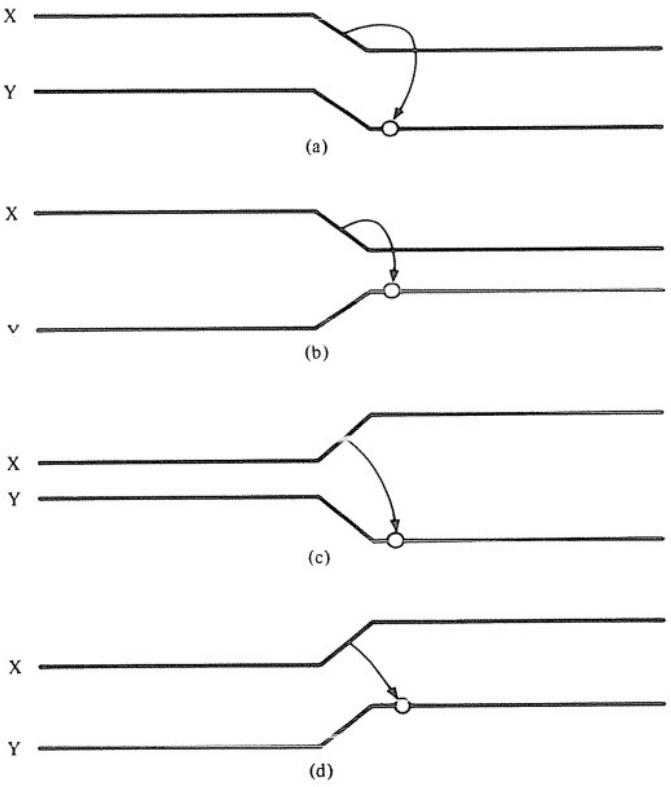


Figure A-4 Levels Caused by Transitions

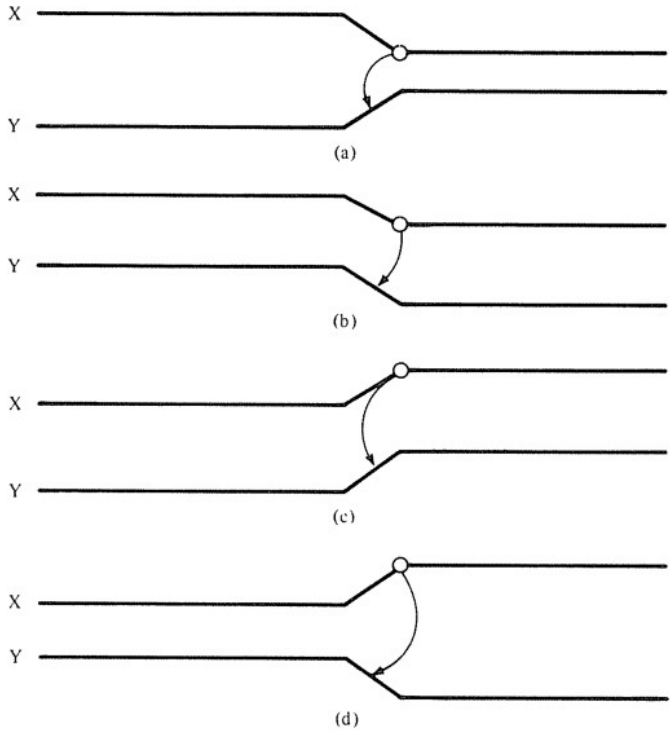


Figure A-5 Signal Transitions Caused by Levels

a. A low level on line *X* causes a low-to-high transition on line *Y*.
b. A low level on line *X* causes a high-to-low transition on line *Y*.
c. A high level on line *X* causes a low-to-high transition on line *Y*.
d. A high level on line *X* causes a high-to-low transition on line *Y*.

A-6 BUS SIGNAL CHANGES CAUSED BY LEVEL CHANGES

Refer to Fig. A-6 for the following explanation.

a. A transition to a low level on line *X* causes the signals on the bus lines to change.
b. A transition to a high level on line *X* causes the signals on the bus lines to change.

A-7 LEVEL CHANGES CAUSED BY LEVEL CHANGES

Refer to Fig. A-7 for the explanation given below.

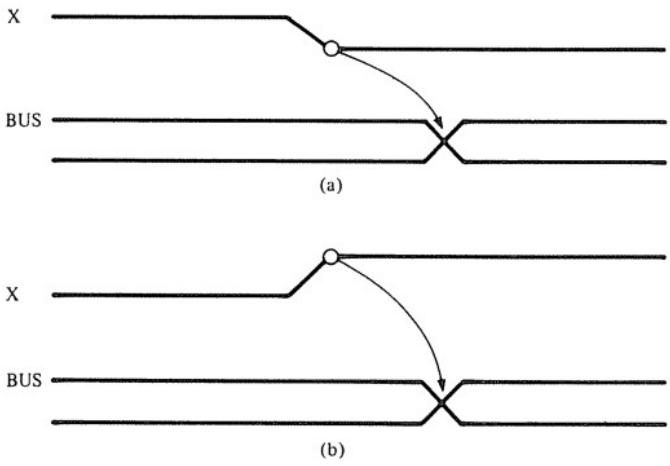


Figure A-6 Changes in Bus Signals Caused by Changes in Levels

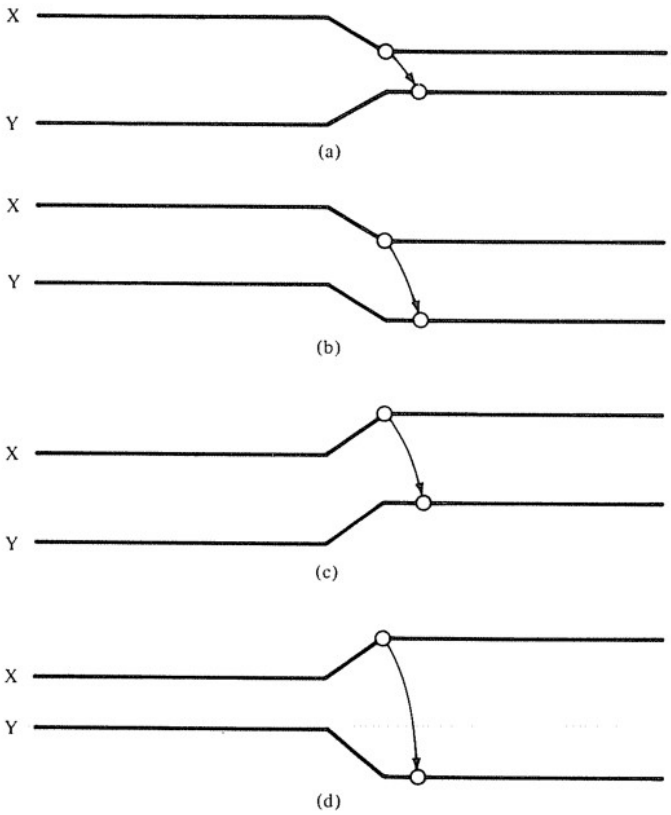


Figure A-7 Level Changes Caused by Levels

a. Transition to a low state on line *X* causes a transition to a high state on line *Y*.
b. Transition to a low state on line *X* causes a transition to a low state on line *Y*.
c. Transition to a high state on line *X* causes a transition to a high state on line *Y*.

d. Transition to a high state on line *X* causes a transition to a low state on line *Y*.

A-8 SINGLE/MULTIPLE TRANSITION CAUSED BY SINGLE/MULTIPLE LEVELS

a. A simultaneous transition from high-to-low level on line *X* and low-to-high level on line *Y* causes a low-to-high transition on line *Z*.
b. A simultaneous transition from high-to-low level on line *X* and low-to-high level on line *Y* causes high-to-low transition on line *Z*.
c. A high-to-low transition on line *X* causes simultaneous transitions, high-to-low on line *Y* and low-to-high on line *Z*.
d. A low-to-high transition on line *X* causes simultaneous transitions, high-to-low on line *Y* and low-to-high on line *Z*.

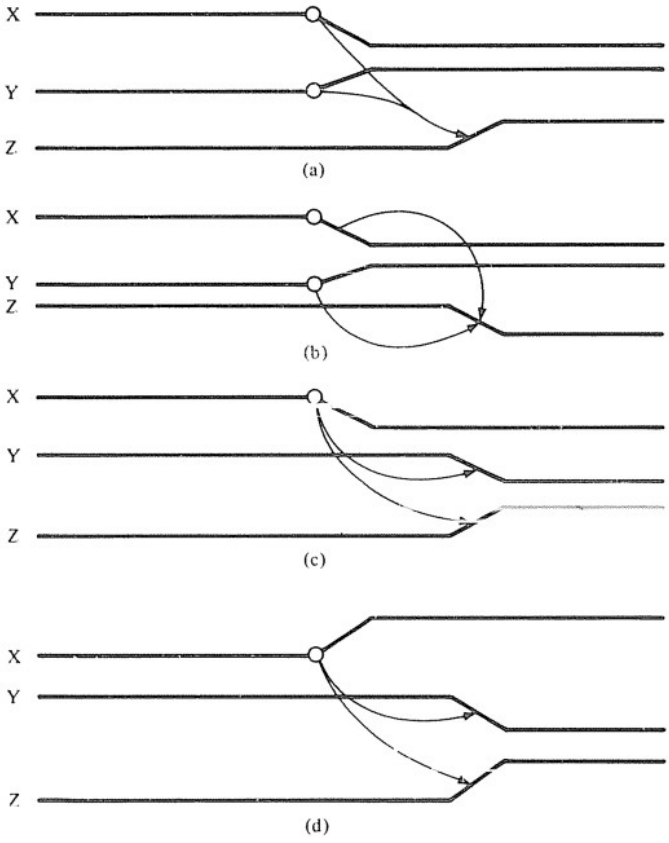


Figure A-8 Single/Multiple Transitions Caused by Single/Multiple Levels

A-9 TRANSITIONS CAUSED BY LEVEL/TRANSITION COMBINATIONS

Refer to Fig. A-9 for the following explanations.

a. A low logic level on line *X* and a low-to-high transition on line *Y* causes a transition in a third, unspecified signal.

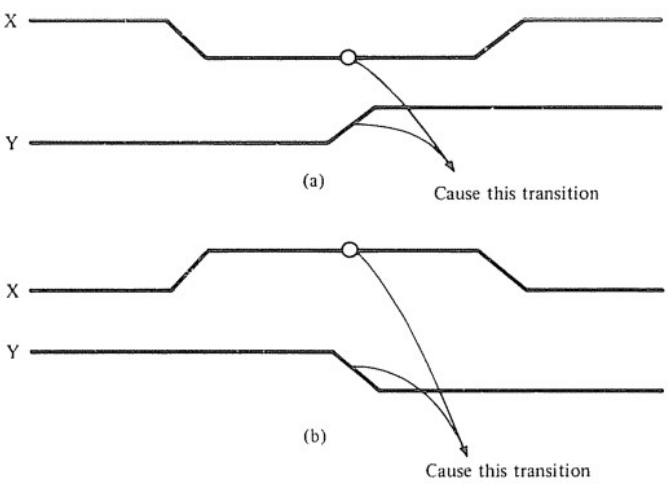


Figure A-9 Transitions Caused by Level/Transition Combinations

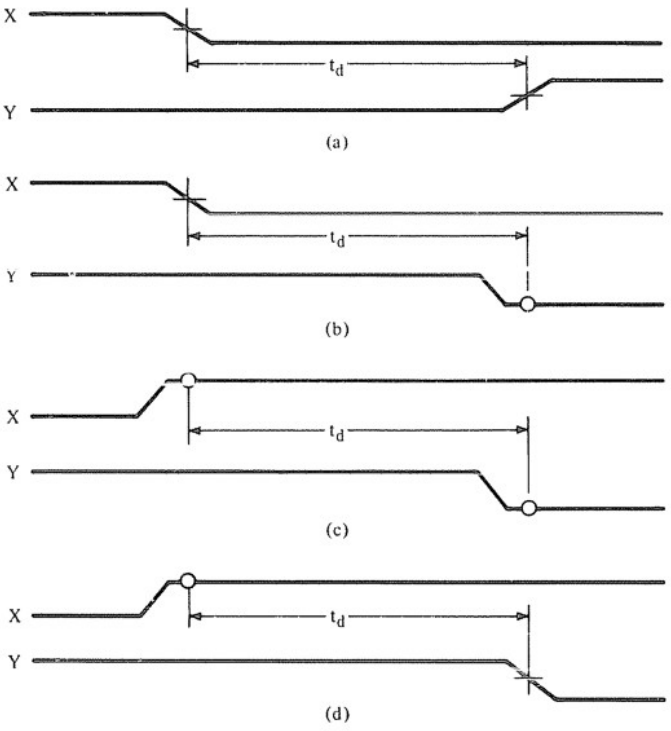


Figure A-10 Delay Time Conventions

b. A high logic level on line X and a high-to-low transition on line Y causes a transition in a third, unspecified signal.

A-10 DELAY TIME CONVENTIONS

Refer to Fig. A-10 for the explanation below. Time delay between two or more signals can be expressed in a number of different ways. Time delays can be measured between the rise and fall times of signals or between signals reaching specific logic levels and stabilizing on the lines. They can also be measured between the rise or fall times and signal levels. In the examples shown in Fig. A-10 we have arbitrarily measured delays from the 50% points in the rise and fall times. The 10% and 90% points are also frequently used in time delay measurements.

a. Indicates time delay measured from the 50% fall time of signal on line X to the 50% rise time of the signal on line Y.
b. Indicates time delay between the 50% fall time of the signal on line X and the resulting stabilized low logic level on line Y.
c. Indicates the resulting time delay between the stabilized high signal on line X and the stabilized low signal on line Y.
d. Indicates the stabilized high level on line X and the 50% fall time transition on line Y.

APPENDIX B

Z80 ASSEMBLY LANGUAGE CODES BY MNEMONICS

Table 2-11 B-BIT LOAD INSTRUCTIONS GROUP

Ins #	*Source Statement*	*Object Code*
1	LD A, A	7F
	LD A, B	78
	LD A, C	79
	LD A, D	7A
	LD A, E	7B
	LD A, H	7C
	LD A, L	7D
	LD B, A	47
	LD B, B	40
	LD B, C	41
	LD B, D	42
	LD B, E	43
	LD B, H	44
	LD B, L	45
	LD C, A	4F
	LD C, B	48
	LD C, C	49
	LD C, D	4A
	LD C, E	4B
	LD C, H	4C
	LD C, L	4D
	LD D, A	57
	LD D, B	50
	LD D, C	51
	LD D, D	52
	LD D, E	53
	LD D, H	54
	LD D, L	55
	LD E. A	5F
	LD E, B	58
	LD E, C	59
	LD E, D	5A
	LD E, E	5B
	LD E, H	5C
	LD E, L	5D
	LD H, A	67
	LD H, B	60
	LD H, C	61
	LD H, D	62
	LD H, E	63
	LD H, H	64
	LD H, L	65
	LD L, A	6F
	LD L, B	68
	LD L, C	69
	LD L, D	6A
1	LD L, E	6B
	LD L, H	6C
	LD L, L	6D
2	LD A, n	3E n
	LD B, n	06 n
	LD C, n	0E n
	LD D, n	16 n
	LD E, n	1E n
	LD H, n	26 n
	LD L, n	2E n
3	LD A, (HL)	7E
	LD B, (HL)	46
	LD C, (HL)	4E
	LD D, (HL)	56
	LD E, (HL)	5E
	LD H, (HL)	66
	LD L, (HL)	6E
4	LD A, (IX+d)	DD7Ed
	LD B, (IX+d)	DD46d
	LD C, (IX+d)	DD4Ed
	LD D, (IX+d)	DD56d
	LD E, (IX+d)	DD5Ed
	LD H, (IX+d)	DD66d
	LD L, (IX+d)	DD6Ed
5	LD A, (IY+d)	FD7Ed
	LD B, (IY+d)	FD46d
	LD C, (IY+d)	FD4Ed
	LD D, (IY+d)	FD65d
	LD E, (IY+d)	FD5Ed
	LD H, (IY+d)	FD66d
	LD L, (IY+d)	FD5Ed
6	LD (HL), A	77
	LD (HL), B	70
	LD (HL), C	71
	LD (HL), D	72
	LD (HL), E	73
	LD (HL), H	74
	LD (HL), L	75
7	LD (IX+d), A	DD77d
	LD (IX+d), B	DD70d
	LD (IX+d), C	DD71d
	LD (IX+d), D	DD72d

Ins #	*Source Statement*	*Object Code*
	LD (IX+d), E	DD73d
	LD (IX+d), H	DD74d
	LD (IX+d), L	DD75d
8	LD (IY+d), A	FD77d
	LD (IY+d), B	FD70d
	LD (IY+d), C	FD71d
	LD (IY+d), D	FD72d
	LD (IY+d), E	FD73d
	LD (IY+d), H	FD74d
	LD (IY+d), L	FD75d
9	LD (HL) n	36n
10	LD (IX+d), n	DD36dn
11	LD (IY+d), n	FD36dn
12	LD A, (BC)	0A
13	LD A, (DE)	1A
14	LD A, (nn)	3Ann
15	LD (BC), A	02
16	LD (DE), A	12
17	LD (nn), A	32 nn
18	LD A, I	ED57
19	LD A, R	ED5F
20	LD I, A	ED47
21	LD R, A	ED4F

Table 2-12 16-BIT LOAD INSTRUCTIONS GROUP

Ins #	*Source Statement*	*Object Code*
22	LD BC, nn	01nn
	LD DE, nn	11nn
	LD HL, nn	21nn
	LD SP, nn	31nn
23	LD IX, nn	DD21nn
24	LD IY, nn	FD21nn
25	LD HL, (nn)	2Ann
26	LD BC, (nn)	ED4Bnn
	LD DE, (nn)	ED5Bnn
	LD HL, (nn)	ED6Bnn
	LD SP, (nn)	ED7Bnn
27	LD IX, (nn)	DD2Ann
28	LD IY, (nn)	FD2Ann
29	LD (nn), HL	22nn
30	LD (nn), BC	ED43nn
	LD (nn), DE	ED53nn
	LD (nn), HL	ED63nn
	LD (nn), SP	ED73nn
31	LD (nn), IX	DD22nn
32	LD (nn), IY	FD22nn
33	LD SP, HL	F9
34	LD SP, IX	DDF9
35	LD SP, IY	FDF9

Table 2-13 STACK PUSH/POP 16-BIT INSTRUCTIONS GROUP

Ins #	*Source Statement*	*Object Code*
36	PUSH BC	C5
	PUSH DE	D5
	PUSH HL	E5
	PUSH AF	F5
37	PUSH IX	DDE5
38	PUSH IY	FDE5
39	POP BC	C1
	POP DE	D1
	POP HL	E1
	POP AF	F1
40	POP IX	DDE1
41	POP IY	FDE1

Table 2-14 EXCHANGE INSTRUCTIONS GROUP

Ins #	*Source Statement*	*Object Code*
42	EX DE, HL	EB
43	EX AF, A'F'	08
44	EXX	D9
45	EX (SP), HL	E3
46	EX (SP), IX	DDE3
47	EX (SP), IY	FDE3

Table 2-15 BLOCK TRANSFER AND SEARCH INSTRUCTIONS GROUP

Ins #	*Source Statement*	*Object Code*
48	LDI	EDA0
49	LDIR	EDB0
50	LDD	EDAB
51	LDDR	EDBB
52	CPI	EDA1
53	CPIR	EDB1
54	CPD	EDA9
55	CPDR	EDB9

Table 2-16 8-BIT ARITHMETIC AND LOGIC INSTRUCTIONS GROUP

Ins #	*Source Statement*	*Object Code*
56	ADD A, A	87
	ADD A, B	80
	ADD A, C	81
	ADD A, D	82
	ADD A, E	83
	ADD A, H	84
	ADD A, L	85
57	ADD A, n	C6n
58	ADD A, (HL)	86
59	ADD A, (IX+d)	DD86d
60	ADD A, (IY+d)	FD86d
61	ADC A, A	8F
	ADC A, B	88
	ADC A, C	89
	ADC A, D	8A
	ADC A, E	8B
	ADC A, H	8C
	ADC A, L	8D
	ADC A, n	CEn
	ADC A, (HL)	8E
	ADC A, (IX+d)	DD8Ed
	ADC A, (IY+d)	FD8Ed
62	SUB A	97
	SUB B	90
	SUB C	91
	SUB D	92
	SUB E	93
	SUB H	94
	SUB L	95
	SUB n	D6n
	SUB (HL)	96
	SUB (IX+d)	DD96d
	SUB (IY+d)	FD96d
63	SBC A, A	9E
	SBC A, B	98
	SBC A, C	99
	SBC A, D	9A
	SBC A, E	9B
	SBC A, H	9C
	SBC A, L	9D
	SBC A, n	DEn
	SBC A, (HL)	9E
	SBC (IX+d)	DD9Ed
	SBC (IY+d)	FD9Ed
64	AND A	A7
	AND B	A0
	AND C	A1
	AND D	A2
	AND E	A3
	AND H	A4
	AND L	A5
	AND n	E6n
	AND (HL)	A6
	AND (IX+d)	DDA6d
	AND (IY+d)	FDA6d
65	OR A	B7
	OR B	B0
	OR C	B1
	OR D	B2
	OR E	B3
	OR H	B4
	OR L	B5
	OR n	F6d
	OR (HL)	B6
	OR (IX+d)	DDB6d
	OR (IY+d)	FDB6d
66	XOR A	AF
	XOR B	A8
	XOR C	A9
	XOR D	AA
	XOR E	AB
	XOR H	AC
	XOR L	AD
	XOR n	EEn
	XOR (HL)	AE
	XOR (IX+d)	DDAE
	XOR (IY+d)	FDAE
67	CP A	BF
	CP B	B8
	CP C	B9
	CP D	BA
	CP E	BB
	CP H	BC
	CP L	BD
	CP (HL)	BE
	CP (IX+ d)	DDBEd
	CP (HL+d)	FDBEd
	CP n	FE n
68	INC A	3C
	INC B	04
	INC C	0C
	INC D	14
	INC E	1C
	INC H	24
	INC L	2C
69	INC (HL)	34
70	INC (IX+d)	DD34d
71	INC (IY+d)	FD34d
72	DEC A	3D
	DEC B	05
	DEC C	0D
	DEC D	15
	DEC E	1D
	DEC H	25
	DEC L	2D
	DEC (HL)	35
	DEC (IX+d)	DD35d
	DEC (IY+d)	FD35d

Table 2-17 16-BIT ARITHMETIC INSTRUCTIONS GROUP

Ins #	*Source Statement*	*Object Code*
73	ADD HL, BC	09
	ADD HL, DE	19
	ADD HL, HL	29
	ADD HL, SP	39
74	ADC HL, BC	ED4A
	ADC HL, DE	ED5A
	ADC HL, HL	ED6A
	ADC HL, SP	ED7A
75	SBC HL, BC	ED42
	SBC HL, DE	ED52
	SBC HL, HL	ED62
	SBC HL, SP	ED72
76	ADD IX, BC	DD09
	ADD IX, DE	DD19
	ADD IX, IX	DD29
	ADD IX, SP	DD39
77	ADD IY, BC	FD09
	ADD IY, DE	FD19
	ADD IY, IY	FD29
	ADD IY, SP	FD39
78	INC BC	03
	INC DE	13
	INC HL	23
	INC SP	33
79	INC IX	DD23
80	INC IY	FD23
81	DEC BC	0B
	DEC DE	1B
	DEC HL	2B
	DEC SP	3B
82	DEC IX	DD2B
83	DEC IY	FD 2B

Table 3-1 ROTATE AND SHIFT INSTRUCTIONS GROUP

Ins #	*Source Statement*	*Object Code*
84	RLCA	07
85	RLA	17
86	RRCA	0F
87	RRA	1F
88	RLC A	CB07
	RLC B	CB00
	RLC C	CB01
	RLC D	CB02
	RLC E	CB03
	RLC H	CB04
	RLC L	CB05
89	RLC (HL)	CB06
90	RLC (IX+d)	DDCBd06
91	RLC (IY+d)	FDCBd06
92	RL A	CB17
	RL B	CB10
	RL C	CB11
	RL D	CB12
	RL E	CB13
	RL H	CB14
	RL L	CB15
	RL (HL)	CB16
	RL (IX+d)	DDCBd16
	RL (IY+d)	FDCBd16
93	RRC A	CB0F
	RRC B	CB08
	RRC C	CB09
	RRC D	CB0A
	RRC E	CB0B
	RRC H	CB0C
	RRC L	CB0D
	RRC (HL)	CB0E
	RRC (IX+d)	DDCBd0E
	RRC (IY+d)	FDCBd0E
94	RR A	CB1F
	RR B	CB18
	RR C	CB19
	RR D	CB1A
	RR E	CB1B
	RR H	CB1C
	RR L	CB1D
	RR (HL)	CB1E
	RR (IX+d)	DDCBd1E
	RR (IY+d)	FDCBd1E

Ins #	*Source Statement*	*Object Code*
95	SLA A	CB27
	SLA B	CB20
	SLA C	CB21
	SLA D	CB22
	SLA E	CB23
	SLA H	CB24
	SLA L	CB25
	SLA (HL)	CB26
	SLA (IX+d)	DDCBd26
	SLA (IY+d)	FDCBd26
96	SRA A	CB2F
	SRA B	CB28
	SRA C	CB29
	SRA D	CB2A
	SRA E	CB2B
	SRA H	CB2C
	SRA L	CB2D
	SRA (HL)	CB3E
	SRA (IX+d)	DDCBd3E
	SRA (IY+d)	FDCBd3E
97	SRL A	CB3F
	SRL B	CB38
	SRL C	CB39
	SRL D	CB3A
	SRL E	CB3B
	SRL H	CB3C
	SRL L	CB3D
	SRL (HL)	CB3E
	SRL (IX+d)	DDCBd3E
	SRL (IY+d)	FDCBd3E
98	RLD	ED6F
99	RRD	ED67

Table 3-2 BIT SET, RESET, AND TEST INSTRUCTIONS GROUP

Ins #	*Source Statement*	*Object Code*
100	BIT 0, A	CB47
	BIT 0, B	CB40
	BIT 0, C	CB41
	BIT 0, D	CB42
	BIT 0, E	CB43
	BIT 0, H	CB44
	BIT 0, L	CB45
	BIT 1, A	CB4F
	BIT 1, B	CB48
	BIT 1, C	CB49
	BIT 1, D	CB4A
	BIT 1, E	CB4B
	BIT 1, H	CB4C
	BIT 1, L	CB4D
	BIT 2, A	CB57
	BIT 2, B	CB50
	BIT 2, C	CB51
	BIT 2, D	CB52
	BIT 2, E	CB53
	BIT 2, H	CB54
	BIT 2, L	CB55
	BIT 3, A	CB5F
	BIT 3, B	CB58
	BIT 3, C	CB59
	BIT 3, D	CB5A
	BIT 3, E	CB5B
	BIT 3, H	CB5C
	BIT 3, L	CB5D
	BIT 4, A	CB67
	BIT 4, B	CB60
	BIT 4, C	CB61
	BIT 4, D	CB62
	BIT 4, E	CB63
	BIT 4, H	CB64
	BIT 4, L	CB65
	BIT 5, A	CB6F
	BIT 5, B	CB68
	BIT 5, C	CB69
	BIT 5, D	CB6A
	BIT 5, E	CB6B
	BIT 5, H	CB6C
	BIT 5, L	CB6D
	BIT 6, A	CB77
	BIT 6, B	CB70
	BIT 6, C	CB71
	BIT 6, D	CB72
	BIT 6, E	CB73

Ins #	*Source Statement*	*Object Code*
100	BIT 6, H	CB74
	BIT 6, L	CB75
	BIT 7, A	CB7F
	BIT 7, B	CB78
	BIT 7, C	CB79
	BIT 7, D	CB7A
	BIT 7, E	CB7B
	BIT 7, H	CB7C
	BIT 7, L	CB7D
101	BIT 0, (HL)	CB46
	BIT 1, (HL)	CB4E
	BIT 2, (HL)	CB56
	BIT 3, (HL)	CB5E
	BIT 4, (HL)	CB66
	BIT 5, (HL)	CB6E
	BIT 6, (HL)	CB76
	BIT 7, (HL)	CB7E
102	BIT 0, (IX+d)	DDCBd46
	BIT 1, (IX+d)	DDCBd4E
	BIT 2, (IX+d)	DDCBd56
	BIT 3, (IX+d)	DDCBd5E
	BIT 4, (IX+d)	DDCBd66
	BIT 5, (IX+d)	DDCBd6E
	BIT 6, (IX+d)	DDCBd76
	BIT 7, (IX+d)	DDCBd7E
103	BIT 0, (IY+d)	FDCBd46
	BIT 1, (IY+d)	FDCBd4E
	BIT 2, (IY+d)	FDCBd56
	BIT 3, (IY+d)	FDCBd5E
	BIT 4, (IY+d)	FDCBd66
	BIT 5, (IY+d)	FDCBd6E
	BIT 6, (IY+d)	FDCBd76
	BIT 7, (IY+d)	FDCBd7E
104	SET 0, A	CBC7
	SET 0, B	CBC0
	SET 0, C	CBC1
	SET 0, D	CBC2
	SET 0, E	CBC3
	SET 0, H	CBC4
	SET 0, L	CBC5
	SET 1, A	CBCF
	SET 1, B	CBC8
	SET 1, C	CBC9
	SET 1, D	CBCA
	SET 1, E	CBCB
	SET 1, H	CBCC
	SET 1, L	CBCD
	SET 2, A	CBD7
	SET 2, B	CBD0
	SET 2, C	CBD1
	SET 2, D	CBD2
104	SET 2, E	CBD3
	SET 2, H	CBD4
	SET 2, L	CBD5
	SET 3, A	CBDF
	SET 3, B	CBD8
	SET 3, C	CBD9
	SET 3, D	CBDA
	SET 3, E	CBDB
	SET 3, H	CBDC
	SET 3, L	CBDD
	SET 4, A	CBE7
	SET 4, B	CBE0
	SET 4, C	CBE1
	SET 4, D	CBE2
	SET 4, E	CBE3
	SET 4, H	CBE4
	SET 4, L	CBE5
	SET 5, A	CBEF
	SET 5, B	CBE8
	SET 5, C	CBE9
	SET 5, D	CBEA
	SET 5, E	CBEB
	SET 5, H	CBEC
	SET 5, L	CBED
	SET 6, A	CBF7
	SET 6, B	CBF0
	SET 6, C	CBF1
	SET 6, D	CBF2
	SET 6, E	CBF3
	SET 6, H	CBF4
	SET 6, L	CBF5
	SET 7, A	CBFF
	SET 7, B	CBF8
	SET 7, C	CBF9
	SET 7, D	CBFA
	SET 7, E	CBFB
	SET 7, H	CBFC
	SET 7, L	CBFD
105	SET 0, (HL)	CBC6
	SET 1, (HL)	CBCE
	SET 2, (HL)	CBD6
	SET 3, (HL)	CBDE
	SET 4, (HL)	CBE6
	SET 5, (HL)	CBEE
	SET 6, (HL)	CBF6
	SET 7, (HL)	CBFE
106	SET 0, (IX+d)	DDCBdC6
	SET 1, (IX+d)	DDCBdCE

Ins #	*Source Statement*	*Object Code*
106	SET 2, (IX+d)	DDCBdD6
	SET 3, (IX+d)	DDCBdDE
	SET 4, (IX+d)	DDCBdE6
	SET 5, (IX+d)	DDCBdEE
	SET 6, (IX+d)	DDCBdF6
	SET 7, (IX+d)	DDCBdFE
107	SET 0, (IY+d)	FDCBdC6
	SET 1, (IY+d)	FDCBdCE
	SET 2, (IY+d)	FDCBdD6
	SET 3, (IY+d)	FDCBdDE
	SET 4, (IY+d)	FDCBdE6
	SET 5, (IY+d)	FDCBdEE
	SET 6, (IY+d)	FDCBdF6
	SET 7, (IY+d)	FDCBdFE
108	RES 0, A	CB87
	RES 0, B	CB80
	RES 0, C	CB81
	RES 0, D	CB82
	RES 0, E	CB83
	RES 0, H	CB84
	RES 0, L	CB85
	RES 1, A	CB8F
	RES 1, B	CB88
	RES 1, C	CB89
	RES 1, D	CB8A
	RES 1, E	CB8B
	RES 1, H	CB8C
	RES 1, L	CB8D
	RES 2, A	CB97
	RES 2, B	CB90
	RES 2, C	CB91
	RES 2, D	CB92
	RES 2, E	CB93
	RES 2, H	CB94
	RES 2, L	CB95
	RES 3, A	CB9F
	RES 3, B	CB98
	RES 3, C	CB99
	RES 3, D	CB9A
	RES 3, E	CB9B
	RES 3, H	CB9C
	RES 3, L	CB9D
	RES 4, A	CBA7
	RES 4, B	CBA0
	RES 4, C	CBA1
	RES 4, D	CBA2
	RES 4, E	CBA3
	RES 4, H	CBA4
	RES 4, L	CBA5

Ins #	*Source Statement*	*Object Code*
108	RES 5, A	CBAF
	RES 5, B	CBA8
	RES 5, C	CBA9
	RES 5, D	CBAA
	RES 5, E	CBAB
	RES 5, H	CBAC
	RES 5, L	CBAD
	RES 6, A	CBB7
	RES 6, B	CBB0
	RES 6, C	CBB1
	RES 6, D	CBB2
	RES 6, E	CBB3
	RES 6, H	CBB4
	RES 6, L	CBB5
	RES 7, A	CBBF
	RES 7, B	CBB8
	RES 7, C	CBB9
	RES 7, D	CBBA
	RES 7, E	CBBB
	RES 7, H	CBBC
	RES 7, L	CBBD
	RES 0 (HL)	CB86
	RES 1 (HL)	CB8E
	RES 2 (HL)	CB96
	RES 3 (HL)	CB9E
	RES 4 (HL)	CBA6
	RES 5 (HL)	CBAE
	RES 6 (HL)	CBB6
	RES 7 (HL)	CBBE
	RES 0 (IX+d)	DDCBd86
	RES 1 (IX+d)	DDCBd8E
	RES 2 (IX+d)	DDCBd96
	RES 3 (IX+d)	DDCBd9E
	RES 4 (IX+d)	DDCBdA6
	RES 5 (IX+d)	DDCBdAE
	RES 6 (IX+d)	DDCBdB6
	RES 7 (IX+d)	DDCBdBE
	RES 0, (IY+d)	FDCBd86
	RES 1, (IY+d)	FDCBd8E
	RES 2, (IY+d)	FDCBd96
	RES 3, (IY+d)	FDCBd9E
	RES 4, (IY+d)	FDCBdA6
	RES 5, (IY+d)	FDCBdAE
	RES 6, (IY+d)	FDCBdB6
	RES 7, (IY+d)	FDCBdBE

Table 3-4 JUMP INSTRUCTIONS GROUP

Ins #	*Source Statement*	*Object Code*
109	JP nn	C3nn
110	JP C, nn	DAnn
	JP NC, nn	D2nn
	JP M, nn	FA
	JP Z, nn	CAnn
	JP NZ, nn	C2nn
	JP P, nn	F2nn
	JP PE, nn	EAnn
	JP PO, nn	E2nn
111	JR DIS	18 dis
112	JR C, DIS	38 dis
113	JR NC, DIS	30 dis
114	JR Z, DIS	28 dis
115	JR NZ, DIS	20 dis
116	JP (HL)	E9
117	JP (IX)	DDE9
118	JP (IY)	FDE9
119	DJNZ	10 dis

Table 3-6 CALL AND RERUN INSTRUCTIONS GROUP

Ins #	*Source Statement*	*Object Code*
120	CALL nn	CD nn
121	CALL C, nn	DC nn
	CALL NC, nn	D4 nn
	CALL Z, nn	CC nn
	CALL NZ, nn	C4 nn
	CALL M, nn	FC nn
	CALL P, nn	F4 nn
	CALL PE, nn	EC nn
	CALL PO, nn	E4 nn
122	RET	C9
123	RET C	D8
	RET NC	D0
	RET Z	C8
	RET NZ	C0
	RET M	F8
	RET P	F0
	RET PE	E8
	RET PO	E0
124	RETI	ED4D
125	RETN	ED45
126	RST 0	C7
	RST 10H	D7
	RST 18H	DF
	RST 20H	E7
	RST 28H	EF
	RST 30H	F7
	RST 38H	FF
	RST 8	FC

Table 4-1 INPUT INSTRUCTIONS GROUP

Ins #	*Source Statement*	*Object Code*
127	IN A (n)	DB n
128	IN A (C)	ED78
	IN B (C)	ED40
	IN C (C)	ED48
	IN D (C)	ED50
	IN E (C)	ED58
	IN H (C)	ED60
	IN L (C)	ED68
129	INI	EDA2
130	INIR	EDB2
131	IND	EDAA
132	INDR	EDBA

Table 4-2 THE OUTPUT INSTRUCTIONS GROUP

Ins #	*Source Statement*	*Object Code*
133	OUT (n), A	D3 n
134	OUT (C), A	ED79
	OUT (C), B	ED41
	OUT (C), C	ED49
	OUT (C), D	ED51
	OUT (C), E	ED59
	OUT (C), H	ED61
	OUT (C), L	ED69
135	OUTI	EDA3
136	OTIR	EDB3
137	OUTD	EDAB
138	OUTDR	EDBB

Table 4-3 BASIC CPU INSTRUCTIONS GROUP

Ins #	*Source Statement*	*Object Code*
139	DAA	27
140	CPL	2F
141	NEG	ED44
142	CCF	3F
143	SCF	37
144	NOP	00
145	HALT	76
146	DI	F3
147	EI	FB
148	IM 0	ED46
149	IM 1	ED56
150	IM 2	ED5E

ANSWERS TO SELECTED REVIEW QUESTIONS

CHAPTER 1 THE Z80 CPU HARDWARE ARCHITECTURE

1-3 **a.** True **c.** True
b. False **d.** True

1-5 Daisy chain logic

1-8 8; 8

1-11 **a.** False **c.** False
b. False **d.** True

1-15 **d.** Register R

1-18 **c.** When the MSB of the corresponding accumulator is a 1.

1-21 **a.** False **b.** True

1-24 Register indirect addressing mode.

1-27 **a.** False **d.** True
b. True **e.** True
c. False

1-30 **a.** False **b.** True

CHAPTER 2 THE Z80 INSTRUCTION SET (PART ONE)

2-2 Load immediate

2-5 Accumulator

2-8 **a.** True **c.** False
b. False **d.** False

2-11 **d.** LD r_1, r_2

2-14 dd

2-17 **a.** $(SP - 1) \longleftarrow IY_H$
$(SP - 2) \longleftarrow IY_L$

2-20 A − (HL)
HL ⟵ HL −1
BC ⟵ BC − 1

CHAPTER 3 THE Z80 INSTRUCTION SET (PART TWO)

3-1 **a.** False **c.** False
b. True **d.** True

3-4 **a.** LSB; MSB **b.** LSB

3-7 **a.** RLC (IX + d); **b.** 4

3-10 **a.** False **c.** True
b. True **d.** False
e. True

3-13 True; PC; incremented.

3-16 1

3-19 **a.** False **c.** False
b. False **d.** True

CHAPTER 4 THE Z80 INSTRUCTION SET AND INTERRUPT RESPONSE

4-2 **a.** True **c.** False
b. False **d.** True

4-5 B; HL

4-8 **a.** True **c.** True
b. False **d.** True

4-11 **a.** B = 0 **c.** Decremented
b. B ≠ 0; B = 0

4-14 **a.** CPL

4-17 DI; resets

4-20 **b.** INT

CHAPTER 5 THE Z80 CPU TIMING AND CONTROL

5-1 **b.** The T cycle

5-4 **a.** No Yes
b. Yes

5-7 **c.** MREQ

5-10 **a.** False **c.** True
b. False **d.** True

5-13 **a.** True **d.** True
b. False **e.** True
c. True

5-16 **c.** It assures sufficient time for the memory to write into the addressed location.

5-19 **a.** False **c.** False
b. True **d.** True
e. True

5-22 IORQ

5-25 Leading; last; last.

CHAPTER 6 Z80 PARALLEL INPUT/OUTPUT CHIP

6-2 **a.** True **c.** False
b. True **d.** True

6-5 **a.** Bit control mode (mode 3)

6-8 D0–D7

6-11 **a.** Data; A **b.** Positive; trailing.

6-14 **a.** Input; STROBE **c.** IORQ
b. INT

6-17 **a.** True **c.** True
b. True **d.** False
e. True

CHAPTER 7 PROGRAMMING THE PIO CHIP

7-1 **a.** True **c.** True
b. False

7-4 **a.** 8 8 **d.** E E
b. D 0 **f.** A 6

7-7

Bit positions	D7	D6	D5	D4	D3	D2	D1	D0
Byte code	1	0	0	1	0	1	0	0

7-10 **c.** The assignment of the status and control lines of port A is arbitrary.

7-13 **a.** False **c.** False
b. False **d.** False

7-16 **a.** Mode 2 requires only the mode set byte, interrupt vector, and interrupt enable flip-flop control byte.

CHAPTER 8 Z80 COUNTER/TIMER CIRCUIT CHIP

8-2 **a.** Yes **c.** Yes
b. Yes

8-5 **a.** False **c.** True
b. True **d.** True
e. True

8-8 **a.** False **c.** True
b. True **d.** True
e. False

8-11 CE; IORQ; RD

8-14 RESET

8-17 **a.** True **c.** False
b. False **d.** True

8-20 **a.** Product **c.** Prescaler; down counter
b. Counters; divides

CHAPTER 9 PROGRAMMING THE CTC CHIP

9-1 **b.** Mode 2

9-4 **a.** 0

9-7 **b.** D0 and D1 **c.** D1 and D2

9-10 **b.** $3D04_H$

9-13 **b.** IEO **c.** INT

9-16 **d.** $3C_H$ and 35_H

9-18 **c.** 921.6 μs

CHAPTER 10 Z80 DIRECT MEMORY ACCESS CHIP

10-2 **b.** The starting address can remain fixed. if so desired.
d. The starting address can be automatically incremented.

10-5 **a.** Yes Yes
b. No

10-8 **a.** On command (load)
c. Automatically (auto restart)

10-11 **a.** False **c.** True
b. False **d.** True

10-14 **a.** INT **c.** IEO
b. IEI

10-17 **a.** False **d.** True
b. True **e.** False
c. False **f.** True

10-20 **a.** Initialization
b. Match word

10-23 **a.** RDY **b.** CE/WAIT

10-26 **a.** False **c.** True
b. False **d.** True

10-29 **a.** Yes **c.** No
b. Yes **d.** Yes

CHAPTER 11 PROGRAMMING THE DMA CHIP

11-1 **a.** 7; 14 **c.** Associated
b. Base

11-4 **a.** True **d.** False
b. False **e.** True
c. False **f.** False

11-7

WR3-0	1 1 0 1 1 1 0 0
WR3-1	1 0 0 0 0 0 0 1
WR3-2	1 1 1 1 1 1 1 1

11-10 1 1/0 1 1 0 1/0 1 0

11-13 **a.** False **c.** False
b. True **d.** True

CHAPTER 12 Z80 SERIAL INPUT/OUTPUT CHIP

12-2 Quadruple; double

12-5 **a.** Yes **c.** No
b. Yes **d.** No

12-8 INT

12-11 Positive; RxC

12-14 **a.** False **c.** True
b. True **d.** True

12-17 **a.** True **d.** False
b. True **e.** False
c. False **f.** True

12-20 **a.** It enters the 3-bit buffer and is then loaded into the 8-bit receive shift register.

12-23 **a.** False **c.** True
b. False **d.** False

CHAPTER 13 Z80 DUAL ASYNCHRONOUS RECEIVER/TRANSMITTER CHIP

13-1 **a.** False **c.** True
b. False **d.** False

13-4 **a.** 6
b. 8
c. 5

13-7 **b.** Initiates the write cycle when CE is active.
c. Initiates the read cycle when CE and RD are active.

13-11 Positive; RxCA

13-13 **a.** False
b. True
c. True
d. True

13-17 The inactive status of M1 and the active status of IORQ and RD.

13-20 **a.** True
b. False
c. True
d. True

INDEX

ADD 4
Addressing Mode
  bit 6
  extended (direct) 6
  immediate 6, 16
  immediate extended 6
  implied 6, 17
  indexed 4, 6
  indirect 4, 107
  modified page zero 7
  register 6
  register indirect 6
  relative 6
Algorithm 5
Arithmetic Logic Unit (ALU) 1, 4
Arrays 2
Assembler 2
Auto Restart 122

Binary-coded-decimal (BCD) 2, 5
Bit
  Carry 5
  Half Carry 5
  Overflow 5
  Parity 5
  Reset 4
  Sample 4
  Set 4
  Sign 5
  Subtract 5
  Test 4
  Zero 5
Bit control, individual 75
Bit control mode 75
Bit manipulation 2
Block lengths 121
Block transfers of data 2
BRANCH 3
Buffering
  double 142
  quadruple 142
Bus acknowledge 121
Bus control 116
Bus Priority logic 116
Bus request 121
Byte counter 116
Byte input, unidirectional 75
Byte manipulation 2
Byte, set interrupt control 87
Byte transfer application 89
Byte output unidirectional 75
Bytes, command and control 121

Carry flip-flop 16
Character
  Bisync 145
  CRC 145
  Monosync 145
Character string 2
Clock period 63
Codes, register pair 18
Comparator 116
Compiler 2
Control word 76
Converter
  Parallel-to-serial 142
  Serial-to-parallel 142
Counter/Timer Circuit (CTC) 2, 93
Counter, down 95
CRC
  generator 154
  initilization 143
  operation 153
CTC counter mode 101
CTC Read cycle 101
CTC Timer mode 102
CTC Write cycle 100
Cycle
  Bus request/acknowledge 69
  Exit from HALT instruction 71
  Instruction 63
  I/O read 66
  I/O write 66
  M 64
  Maskable interrupt request/acknowledge 69
  Memory Read 65
  Memory Write 65
  Non-maskable interrupt request/acknowledge 70
  OP code fetch 64
  Stealing 2
  T 63

Daisy chain 74
Daisy chain, extending the 83
DART (Dual Asynchronous Receiver/Transmitter) 159
DART Timing 166
Data path
  Receive 161
  Transmit 162
Debugging, real-time 2
Decimal Arithmetic Adjust (DAA) 5
Decrement 4
Depletion Mode 2

Development system (hardware/software) 2
Direct Memory Access (DMA) 2
Disk, floppy 2
Disk Operating System (DOS) 2
Displacement 6
Displacement byte 4
DMA mode
  Burst 115, 120
  Byte-at-a-time 115, 120
  Continuous 115, 120
  Transparent 120
DMA operation
  Search only 114, 120
  Search-Transfer combined 114, 120
  Transfer only 114
DMA Transfers
  I/O-to-I/O 120, 124
  I/O-to-memory 120, 124
  Memory-to-I/O 120, 123
  Memory-to-memory 120, 124
Dual-in-line (DIP) packages 2, 7, 77
Dynamic memory chips 2

Effective memory address 6
End-of-block 114, 125
Error
  Framing 143
  Overrun 143
  Parity 143
Exclusive-OR 153
Executive firmware 2
Executive time, instruction 2

File maintenance 2
FIFO (First-in, first-out) 145
Fixed part of mnemonic 17
Flag
  Auxiliary carry 5
  End-of-block 116
  Ready 77
Full Duplex 142
Functions
  bit-oriented 142
  byte-oriented 142

Handshake capability 75
Hunt phase 145

Increment 4
Industrial control application 88
Instruction
  HALT 9
  Restart 7

Instructions
  arithmetic, group of 16
  basic CPU group 16, 58
  bit manipulation group 16
  block transfer, group of 16
  CALL group of 16
  EXCHANGE, group of 16
  Input/Output, group of 16, 52, 54
  JUMP, group of 16, 43
  Load, group of 16
  Logic, group of 16
  RETURN, group of 16
  ROTATE, group of 16
  Search, group of 16
  Shift, group of 16
INTEL CORPORATION 1, 15
INTEL 8008 uC 1
INTEL 8080A uC 1, 15
INTEL 8085A uC 1
Internal Data Bus 4
Interrupt acknowledge 82
Interrupt acknowledge cycle timing 102
Interrupt control and priority logic 116
Interrupt enable/disable flip-flop 57, 88
Interrupt
  maskable (INT) 57
  multi-level priority 77
  non-maskable (NMI) 57
  pending 116
  request ($\overline{INT}$) 9
  servicing 82
  system 57
Interrupts, multiple-level 4

JUMP 3

Last-in, first-out (LIFO) 4
Logic
  internal control 92, 143
  interrupt control 94, 98
Logical functions 4
Look-up tables 9

Machine cycle one ($\overline{M1}$) 8
Match found 114
Mnemonic conversion 38
Mnemonic structure 17
Mode
  asynchronous 144, 146
  bidirectional 81
  bit control 75, 81, 87
  counter 93, 97
  input byte 80
  synchronous 145, 147
  timer 93, 100
  output byte 77
Mode control register 75
Mode set word 87
MOS RAM chips, dynamic 4
MOS RAM chips, static 4
MOSTEK, INC. 1

N-channel 2
Nesting 4
Non-maskable interrupt ($\overline{NMI}$) 9
Not ready signal 125

OP code 6, 16
OP code construction 39
Operand, implied 18
ON Match 125

Package pin assignments 7
Page, base 7
Page, zero 7
Parallel input/output (PIO) chip 1, 74
Pattern generation, test 2
Pointer 6
Pointer, address 77, 107
POP operation 3
Port addressing 122
Prescalar 95
Program counter 3
Program status word 5
Protocol
  bit-oriented 146
  interrupt control 76, 96
  SDLC (IBM) 146
Pulse generation logic 116, 121
Pulse interval 116
PUSH operation 3

Refresh address 8
Register
  alternate set 3
  block length 116
  channel control 95
  general-purpose (GP) 2, 3
  index 2,
  instruction 4
  interrupt page address 4
  interrupt vector 4, 116
  main set 3
  mask 76, 116
  mask control 76
  match work 116
  memory refresh 4
  special-purpose (SP) 2, 3
  status 116, 136, 143
  sync-character 143
  time constant 95, 106
Return from interrupt timing 103

SDLC message format 147
Serial input-output (SIO) chip 2
Shifts
  arithmetic 16, 37
  logical 16, 38
Silicon-gate 2
Simulator 2
Software
  diagnostics 2
  support 2
Stack Pointer 3
Static memory chips 2
STROBE signal 7
Subtract 4
Subroutine, service 4
Symbolic expressions 19
Symbols
  register/memory (for rotation/shift) 38
  status register 18
Synchronization, external 145

Table look-up 4
Text editor 2
Timing
  bus release (burst/continuous mode, DMA) 125
  bus release (byte-at-a-time mode) 126
  bus request and acceptance (DMA) 124
  DMA active state 123
  DMA inactive state 122
  variable cycle 122
Throughput rate 2
Trigger pulse 102
Tristate busses 7
True memory address 6
Twos complement 4, 6

UART 142
USART 142

Variable cycle length time 122
Variable part of mnemonic 17
Vector, interrupt 77, 87, 94, 95, 97, 107

WAIT states 64
Word, channel control 95, 97, 106
Words, command for DMA set-up 87
Word, time constant 97, 106, 108

ZILOG, INC. 1

ISBN 0-471-85287-2