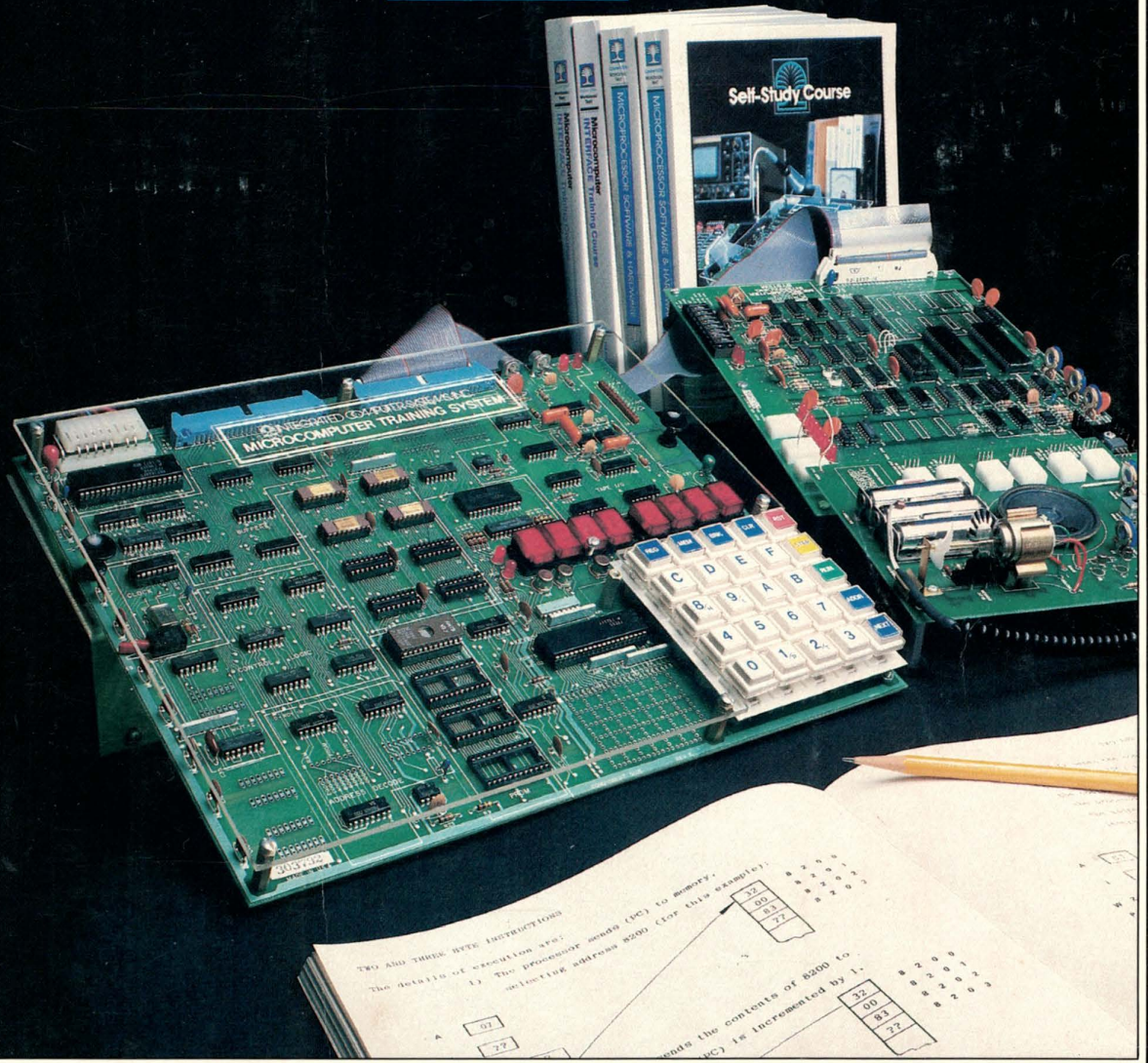


Self-Study Course



MICROPROCESSOR REAL-TIME INTERFACING

Workbook/Text

Volume 1



Self-Study Course

Course 536A:
**MICROPROCESSOR
REAL-TIME INTERFACING**

Workbook/Text

Volume 1

DEVELOPED & PUBLISHED BY:
INTEGRATED COMPUTER SYSTEMS
Course Development Division
© Copyright 1980

SENIOR AUTHOR:
Edward Dillingham, M.E., M.S.E.E.

ASSISTED BY:
Dr. Daniel M. Forsyth
Dr. Rudolf Hirschmann
Ms. Ruth H. Savoie
Dr. David C. Collins

EDUCATION IS OUR BUSINESS™

All materials © copyright 1980 by Integrated Computer Systems.
Not to be reproduced without prior written consent.

D/9/80

© Copyright 1980 by INTEGRATED COMPUTER SYSTEMS.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, or translated into any language, without the prior written permission of the publisher.

MICROPROCESSOR REAL-TIME INTERFACING

Two Volumes
ISBN 0-89438-003-6
Volume I
ISBN 0-89438-004-4
Volume II
ISBN 0-89438-005-2

VOLUME ITABLE OF CONTENTS

TABLE OF CONTENTS		i
LIST OF ILLUSTRATIONS		vii
1	HARDWARE INTERFACING AND REAL TIME PROGRAMMING	
1.1	INTRODUCTION	1-1
1.2	PURPOSE AND CONTENT OF THE COURSE	1-1
1.3	CABLES AND CONNECTIONS	1-3
1.3.1	Power Connections	1-3
1.3.2	Singal Terminals	1-5
1.4	INTERFACE HARDWARE AND REFERENCES	1-7
1.4.1	MTS Interface	1-7
1.4.2	Added Memory	1-7
1.4.3	Chip Selects and Resets	1-11
1.4.4	Port 1 and A/D - D/A Converter	1-15
1.4.5	Interrupt System	1-17
1.4.6	Optical Couplers and Power Driver	1-19
1.4.7	Serial Interface Circuit	1-19
1.4.8	Tape Cassette Modem	1-23
1.4.9	Tape Cassette Library	1-23
1.5	USE OF PMTL OR INTEGRATED EXPERIMENT ASSEMBLY	1-24
2	INPUT/OUTPUT AND INTERRUPTS	
2.1	PORT ASSIGNMENTS AND ADDRESSES	2-1
2.2	PROGRAMMING AND USING THE 8255	2-4
2.3	PORT 1A LED's AND DRIVERS	2-10
2.4	MTS DISPLAY	2-12
2.5	INPUT/OUTPUT CONNECTIONS	2-15
2.6	EXTERNAL INPUTS 4 AND 5	2-16
2.7	INTERRUPT FLIP-FLOPS AND ENABLES	2-19
2.7.1	Interrupt Sources	2-19
2.7.2	Interrupt Flip-Flops	2-21
2.7.3	Interrupt Status and Enables	2-22
2.7.4	Clearing Interrupts	2-25

TABLE OF CONTENTS

2.8	RESTART INSTRUCTIONS	2-31
2.8.1	RST Dispatch	2-31
2.8.2	RST Generation	2-33
2.9	INTERRUPT SERVICE FOR EXT 4 AND EXT 5	2-37
2.10	STANDARD PROGRAMMING FOR 8255's	2-43
3	INTERVAL TIMERS	
3.1	INTEL 8253 INTERVAL TIMER	3-1
3.2	CLOCK, GATE, AND OUTPUT	3-5
3.3	TIMER MODES	3-9
3.4	MODE 0 - INTERRUPT ON TERMINAL COUNT	3-13
3.5	RESTARTING A COUNTER IN MODE 0	3-21
3.6	READING A TIMER	3-25
3.6.1	Measuring a Pulse Duration	3-25
3.6.2	Additional Exercises	3-34
3.6.3	Reading While Counting	3-36
3.7	MODE 2 - RATE GENERATOR	3-39
3.7.1	Use of Mode 2	3-39
3.7.2	Real Time Clock	3-43
3.8	CASCADED TIMERS	3-51
3.9	MODE 3 - SQUARE WAVE GENERATOR	3-61
3.9.1	Observing the Output	3-61
3.9.2	Observing the Counting	3-62
3.10	TIMER MODE DESCRIPTIONS	3-65
4	DIGITAL TO ANALOG OUTPUT	
4.1	METHODS OF D/A OUTPUT	4-3
4.2	PULSE WIDTH MODULATION	4-5
4.2.1	PWM Output Program	4-6
4.2.2	Variable Cycle Time	4-23
4.3	FREQUENCY CONTROL	4-25
4.3.1	Audio Tone Generator	4-27
4.3.2	Frequency Modulation Program	4-29
4.3.3	Recorded Music Player	4-33
4.3.4	Music Recording Program	4-48

TABLE OF CONTENTS

4.4	MULTI-BIT OUTPUT	4-51
4.5	ANALOG VOLTAGE GENERATION	4-55
4.5.1	Binary Summing Circuit	4-55
4.5.2	R-2R Ladder Network	4-61
4.6	FERRANTI D/A CONVERTER	4-67
4.6.1	D/A Circuit Input and Output	4-69
4.6.2	D/A Circuit Control Signals	4-69
4.6.3	Generating an Analog Voltage	4-71
4.7	FUNCTION GENERATOR	4-73
4.7.1	Voltage Ramps	4-75
4.7.2	Keyboard Controlled Function Generators	4-80
4.7.3	Exponential Function	4-110
5	ANALOG TO DIGITAL INPUT	
5.1	PULSE INTERVAL MEASUREMENT	5-3
5.1.1	Measuring a Steady Signal	5-3
5.1.2	Measuring a Multi-Valued Interval	5-6
5.1.3	Measuring Received Pulse Intervals	5-24
5.2	FREQUENCY MEASUREMENT	5-25
5.2.1	Logic Level Frequency Measurements	5-25
5.2.2	AC Input Signal	5-30
5.3	A/D INPUT - VOLTAGE	5-41
5.3.1	Output, Input and Display Subroutine	5-43
5.3.2	Ramping Voltmeter	5-53
5.3.3	Tracking Voltmeter	5-63
5.3.4	Successive Approximation Voltmeter	5-66
5.4	AUTOMATIC A/D INPUT	5-73
5.4.1	Reading A/D Input	5-77
5.4.2	A/D Input with Interrupt	5-83
5.5	DIGITAL NOISE FILTER	5-88
5.5.1	Filter Program Algorithm	5-89
5.5.2	Program Definitions	5-90
5.5.3	Filter Response	5-103

TABLE OF CONTENTS

5.6	TEMPERATURE MEASUREMENT	5-104
5.6.1	Thermistor Characteristics	5-104
5.6.2	Thermistor Operation	5-109
5.6.3	Thermistor Input Adjustment	5-111
5.6.4	Table Lookup and Interpolation	5-112
5.6.5	Voltage to Temperature Conversion	5-115
5.6.6	Thermometer Program	5-126
5.6.7	Data Logging	5-139
5.6.8	Thermistor Self Heating	5-149
5.6.9	Other Temperature Logging Experiments	5-156
5.6.10	Abbreviated Temperature Lookup	5-156
5.6.11	Thermistor Resistance Matching	5-161

TABLE OF CONTENTS

VOLUME II

6	CLOSED LOOP CONTROL	
6.1	ON-OFF CONTROL	6-3
6.1.1	On-Off Control Without Deadband	6-6
6.1.2	On-Off Control with Deadband	6-21
6.1.3	Thermostat with Alarm Limits	6-26
6.1.4	Two-Way Control	6-29
6.2	PROPORTIONAL vs INTEGRAL CONTROL	6-33
6.2.1	Voltage Control Circuit	6-41
6.2.2	Voltage Control by PWM	6-49
6.2.3	Observing Response Time	6-82
6.2.4	Closing the Loop	6-85
6.4.5	Closed Loop Operation	6-104
6.2.6	Closed Loop Response	6-111
6.3	PROPORTIONAL PLUS INTEGRAL CONTROL	6-121
6.3.1	Applying Gain to Error Signal	6-122
6.3.2	Subroutine CLOSL Version 2	6-125
6.3.3	Revised Program	6-129
6.3.4	Experiments with PI Control	6-138
6.3.5	Full Scale Control and Overflow	6-148
6.4	PROPORTIONAL - INTEGRAL - DIFFERENTIAL CONTROL	6-164
6.5	SUMMARY	6-166
7	MOTOR CONTROL	
7.1	OPTICAL DISC AND SLOT SENSOR	7-3
7.1.1	Motor, Sensor and Disc Mounting	7-5
7.1.2	Slot Sensor Connection and Adjustment	7-8
7.1.3	Motor Connection	7-13
7.1.4	Motor Characteristics	7-17
7.2	CONTROL SYSTEM DEVELOPMENT	7-21
7.2.1	Speed Measurement	7-24
7.2.2	Interrupt Service	7-27
7.2.3	Initialization	7-34
7.2.4	Main Program Loop	7-37
7.2.5	Keyboard Input Subroutine KYTIM	7-41

TABLE OF CONTENTS

7.2.6	Subroutine DECBI	7-46
7.2.7	Subroutines SMULT, SCUML	7-47
7.2.8	Open Loop Operation	7-48
7.2.9	False Speed Indications	7-48
7.3	CLOSED LOOP MOTOR CONTROL	7-51
7.3.1	Subroutine SPEED	7-53
7.3.2	Subroutine WIDTH	7-57
7.3.3	Subroutine DIVID	7-61
7.3.4	Summary of Subroutines	7-65
7.3.5	Speed Logging	7-66
7.3.6	Motor Control Program Operation	7-81
7.4	MOTOR CONTROL BY VARIABLE VOLTAGE	7-83
7.5	POWER TRANSISTOR DISSIPATION	7-88
7.6	REVIEW	7-91

APPENDIX A	REFERENCE FIGURES	A-1
APPENDIX B	CASSETTE INTERFACE INSTRUCTIONS AND PROGRAM CASSETTE LIBRARY	B-1
APPENDIX C	RS 232c INTERFACE SYSTEM	C-1
APPENDIX D	TELETYPE INTERFACE SYSTEM	D-1

LIST OF ILLUSTRATIONS

VOLUME ILIST OF ILLUSTRATIONS

<u>FIGURE</u>	<u>TITLE</u>	<u>PAGE</u>
1-1	Interface Training System	1-4
1-2	Microcomputer Interfacing System	1-6
1-3	List of Interface Signals to MTS	1-8
1-4	List of Interface Signals to MTS	1-9
1-5	I/O Chip Selects and Interrupt Flip-Flop Reset	1-10
1-6	Truth Table for Chip Selects and Resets	1-13
1-7	Port 1 and A/D - D/A Converter	1-14
1-8	Vectored Priority Interrupt System	1-16
1-9	Optical Couplers and Power Driver	1-18
1-10	Serial Interface Circuit	1-20
1-11	Tape Cassette Modem	1-22
2-1	8255 I/O Port Assignments	2-2
2-2	Port Addresses and Assignments	2-3
2-3	Programming the 8255's	2-7
2-4	MTS Keyboard Configuration and Port Assignments	2-9
2-5	Input/Output Connections	2-14
2-6	Interrupt System - Partial Diagram	2-18
2-7	EXT 4 and EXT 5 Connections and Signals	2-20
2-8	Clearing Interrupts (Flowchart)	2-26
2-9	Clearing Interrupt Flip-Flops	2-28
2-10	Interrupt Dispatching	2-30
2-11	Generation of RST Instructions	2-32
2-12	Interrupt Service - RST 5, RST 6 (Flowchart)	2-36
2-13	Status and Command Bytes	2-39
2-14	EXT 4 and EXT 5 Service	2-40
2-15	Standard Programming for 8255's	2-44

LIST OF ILLUSTRATIONS

<u>FIGURE</u>	<u>TITLE</u>	<u>PAGE</u>
3-1	Intel 8253 Interval Timer	3-3
3-2	Timer Clocks, Gates, and Outputs	3-4
3-3	Timer Control Byte Structure	3-10
3-4	Timer Control Bytes	3-12
3-5	Compare Timing Loop with Interval Timer (Flowchart)	3-14
3-6	Compare Timing Loop with Interval Timer (Program)	3-17
3-7	GETKY Flow Diagram	3-20
3-8	GETKY with Timer	3-22
3-9	GETKY Using Interval Timer	3-24
3-10	Twos and Tens Complement Counting	3-27
3-11	Time Diagram for Pulse Width Measurements	3-29
3-12	Pulse Width Measurement (Flowchart)	3-30
3-13	Pulse Width Measurement (Program)	3-31
3-14	Timer and Flip-Flop Operation - Mode 2 Rate Generator	3-40
3-15	Time of Day Clock	3-42
3-16	RST 5 Interrupt Service	3-45
3-17	Time of Day	3-46
3-18	Cascaded Timers	3-50
3-19	Cascaded Timers with Main Gate Input	3-52
3-20	Time Delay Program - Main (Flowcharts)	3-55
3-21	Time Delay Program	3-57
3-22	Square Wave Generator - Mode 3	3-60
3-23	Reading the Timer Contents	3-64
3-24	8253 Timer Modes	3-67
3-25	Timing Relationships	3-68
4-1	A/D and D/A Conversion	4-2
4-2	Output Connections for PWM	4-6
4-3	PWM Interrupt Service	4-9
4-4	PWM Test Program	4-11
4-5	PWM Main Program	4-12
4-6	Conversion of Binary Count to Time	4-15

LIST OF ILLUSTRATIONS

<u>FIGURE</u>	<u>TITLE</u>	<u>PAGE</u>
4-7	Pulse Width Modulation Program	4-18
4-8	Audio Output Program and Circuit	4-26
4-9	List of Concert Pitch Musical Tones	4-28
4-10	Tone Generator - Main Program	4-30
4-11	Tone Generator - Interrupt Service	4-31
4-12	Codes for Musical Notes	4-32
4-13	Tune - Main Program	4-34
4-14	Tune Interrupt Service	4-36
4-15	Tune Program	4-39
4-16	Tone Table for Chromatic Scale	4-43
4-17	Home on the Range, and the Drunken Sailor	4-46
4-18	Music Recording Program, Hex Key Chart	4-50
4-19	Patch to Display Tone	4-54
4-20	Binary Summing Circuit	4-56
4-21	Numerical Values for Circuit of 4-20	4-57
4-22	Binary Summing Circuit with Op Amp	4-59
4-23	R-2R Ladder Network	4-60
4-24	R-2R Ladder Impedance	4-63
4-25	Equivalent Circuits for Single Bit = 1	4-64
4-26	D/A Converter Output Circuit	4-66
4-27	Ferranti D/A Converter	4-68
4-28	Keyboard to Voltage Program Flow and Circuit Connection	4-70
4-29	Voltage Ramp Generators	4-74
4-30	Triangular Function Generator	4-79
4-31	Keyboard Controlled Function Generator	4-81
4-32	Keyboard Controlled Function Generator	4-85
4-33	Ramp - Dispatch	4-87
4-34	Function - Key Input Processing	4-91
4-35	Timer 0 Interrupt Service	4-92
4-36	Triangular Wave Function Subroutine	4-96
4-37	Function Generator	4-99
4-38	Exponential Function	4-111

LIST OF ILLUSTRATIONS

<u>FIGURE</u>	<u>TITLE</u>	<u>PAGE</u>
4-39	Successive Charge/Discharge Cycles	4-114
4-40	Key Selection of Waveform	4-117
4-41	EXPV	4-121
4-42	Subroutine BMULT	4-126
4-43	Test Program for EXPV	4-131
4-44	Function Generator	4-132
4-45	Multiplication - Subroutine BMULT	4-140
4-46	Function Subroutine EXPV	4-141
5-1	Pulse Interval Measurement (Flowchart)	5-2
5-2	Pulse Interval Measurement (Program)	5-4
5-3	Multi-Valued Interval (Flowchart)	5-8
5-4	Multi-Valued Interval (Program)	5-17
5-5	Frequency Measurement - Interrupt	5-26
5-6	Frequency Measurement (Program)	5-27
5-7	Protection Circuits for AC Signals	5-31
5-8	AC Frequency Measurement (Flowcharts)	5-33
5-9	AC Frequency Measurement (Program)	5-36
5-10	Connections for Voltmeter Experiments	5-40
5-11	Output, Input, and Display Subroutine	5-44
5-12	Test Program for OIDSF	5-46
5-13	OIDSF - Program	5-49
5-14	Voltage Ramp Generator (Flowchart)	5-52
5-15	D/A Outputs and Inputs	5-53
5-16	Voltage Ramp Generator (Program)	5-54
5-17	Ramping Voltmeter (Flowchart)	5-56
5-18	Ramping Voltmeter (Program)	5-58
5-19	Tracking Voltmeter (Flowchart)	5-62
5-20	Tracking Voltmeter (Program)	5-65
5-21	Successive Approximation Signals	5-66
5-22	Successive Approximation Voltmeter (Flowchart)	5-68

LIST OF ILLUSTRATIONS

<u>FIGURE</u>	<u>TITLE</u>	<u>PAGE</u>
5-23	Successive Approximation Voltmeter (Program)	5-70
5-24	Ferranti A/D Logic	5-72
5-25	Automatic A/D Input (Flowchart)	5-76
5-26	Automatic A/D Input (Program)	5-78
5-27	A/D Input with Interrupt (Flowchart)	5-82
5-28	A/D Input with Interrupt (Program)	5-84
5-29	Subroutine FILTR (Flowchart)	5-92
5-30	A/D Input with FILTR - Program	5-97
5-31	Filter Response for Various N	5-102
5-32	Thermistor Resistance	5-105
5-33	Thermistor Connection and Voltage Plot	5-108
5-34	Expected Voltage at Room Temperature	5-110
5-35	Temperature Conversion by Integration	5-114
5-36	Thermistor Calibration Data	5-116
5-37	Temperature Lookup by Integration	5-118
5-38	Test Program for Temperature Lookup	5-120
5-39	Thermometer (Flowcharts)	5-128
5-40	Thermometer (Program)	5-130
5-41	Logging Thermometer - Main	5-140
5-42	Logging Thermometer - Review Data	5-142
5-43	Logging Thermometer - Replay	5-143
5-44	Logging Thermometer - Timing Constants	5-144
5-45	Logging Thermometer (Program)	5-145
5-46	Thermistor Connection and Calibration for Self-Heating Experiment	5-150
5-47	Thermistor Self-Heating (Program)	5-153
5-48	Abbreviated Temperature Lookup	5-157
5-49	Thermistor Resistor Matching	5-160
5-50	Thermistor Resistor Matching Flow	5-163

LIST OF ILLUSTRATIONS

VOLUME II

<u>FIGURE</u>	<u>TITLE</u>	<u>PAGE</u>
6-1	Connections for Thermostat Exercise	6-2
6-2	Connections for On-Off Voltage Control	6-4
6-3	On-Off Control, No Deadband	6-7
6-4	Thermostat (Program)	6-9
6-5	Thermostat with Deadband - RST 6	6-20
6-6	Thermostat with Deadband (Program)	6-22
6-7	Circuit Connections for Simulation	6-28
6-8	Heating and Cooling Simulation	6-29
6-9	Heating and Cooling Limits	6-30
6-10	Connections for PWM Experiment	6-40
6-11	PWM Voltage - Fixed Period	6-44
6-12	PWM Voltage Control - Main Loop	6-48
6-13	PWM Voltage - Subroutine KYTIM	6-50
6-14	PWM KYTIM - Set Pulse Widths	6-52
6-15	Logging Voltmeter	6-58
6-16	PWM Timer Operation	6-63
6-17	PWM Interrupt Service	6-65
6-18	PWM Voltage Control (Program)	6-71
6-19	PWM - Open Loop Response	6-83
6-20	PWM Subroutine CLOSL	6-91
6-21	PWM Subroutine INTEG	6-93
6-22	REG Module of KYTIM	6-95
6-23	PWM Voltage Control (Program)	6-97
6-24	Open and Closed Loop Waveforms	6-112
6-25	Closed Loop Response Waveform	6-114
6-26	Effect of Total Period	6-116
6-27	Subroutines CLOSL, INTEG, and PROPG	6-124
6-28	PWM Voltage Control (Program)	6-130
6-29	Response with Proportional Control	6-140
6-30	Response Versus Integral Gain	6-142
6-31	Proportional Plus Integral Response	6-144

LIST OF ILLUSTRATIONS

<u>FIGURE</u>	<u>TITLE</u>	<u>PAGE</u>
6-32	Response to Voltage Request	6-146
6-33	LDT1 with Full Scale Control	6-150
6-34	PWM - Subroutine LDT1, Version 2	6-152
6-35	Full Scale Response to Voltage Request	6-156
6-36	Subroutine ADTOV - Double Precision Add and Test for Overflow	6-161
6-37	PWM Subroutine INTEG, Version 3 (Program)	6-162
7-1	Motor and Slot Sensor	7-2
7-2	Motor, Sensor and Disc Mounting	7-4
7-3	Optical Slot Sensor	7-6
7-4	Test for Slot Sensor	7-10
7-5	Motor Connections	7-12
7-6	Motor Connections with External Power	7-14
7-7	Motor Speed vs Voltage	7-16
7-8	Motor Speed vs Duty Cycle Open Loop	7-18
7-9	Motor Control Program Structure	7-20
7-10	Motor Control Interrupt Manager	7-26
7-11	EXT 4 Interrupt Service	7-28
7-12	Timer 0 Service	7-32
7-13	Motor Control	7-36
7-14	KYTIM - Input and Dispatch	7-40
7-15	Load Timer 1 Modules	7-44
7-16	Motion Detection with Dual Sensors	7-50
7-17	Subroutine SPEED	7-52
7-18	Subroutine WIDTH	7-56
7-19	Subroutine DIVID	7-60
7-20	Subroutine Register Usage	7-64
7-21	Motor Control (Program)	7-67
7-22	Motor Control (Program)	7-75
7-23	Patches for Variable Voltage Control	7-85
7-24	Patches to Motor Control (Program)	7-86
7-25	More Patches to Motor Control (Program)	7-89

This page intentionally left blank.

MICROCOMPUTER INTERFACING WORKBOOK

CHAPTER 1

HARDWARE INTERFACING AND REAL TIME PROGRAMMING

This page intentionally left blank.

1.1 INTRODUCTION

All computer applications involve the connection of the computer to external hardware for input and output. In computational systems these external devices are slaves to the computer, and they exist only to serve the computer. In control applications, however, the computer (or microcomputer) exists to serve the process, and the computer design and programming must be adapted to the process. In this course we will be concerned with control applications: how a microprocessor is connected to equipment it controls, and how it is programmed to meet process requirements.

In most control applications the computer must receive input data, process the data and generate control outputs in a timely fashion in order to achieve its intended goals. Failure to react in the allowed time will result in loss of data and possibly improper control. Real time programming deals with these requirements. Most of the exercises in this course are real time programs.

1.2 PURPOSE AND CONTENT OF THE COURSE

The text and exercises of this course teach the use of programmed, timed, and interrupt driven input and output. These are applied to open and closed loop control problems, with various forms of discrete and analog input signals. Sensor calibration is used to convert a thermistor signal to temperature, and the speed of a motor is measured using an optical sensor. Triangular and logarithmic output signals are generated. A digital noise filter is developed. The student will measure the response time of a closed loop control

HARDWARE INTERFACING AND REAL TIME PROGRAMMING

system, after observing the difference in behavior between open loop and closed loop control. Although no attempt is made to teach servomechanism and feedback theory, the basic ideas of proportional and integral feedback are presented and used in exercises.

The interface circuit board includes an interrupt system with priorities and vectors. These are explained and used in many exercises having multiple interrupts.

The manufacturers of microprocessors are introducing new LSI chips to make real time control systems easier to design and cheaper to build. The background provided through this course will make such devices comprehensible to the engineer and programmer. Two such devices, the INTEL 8255 Peripheral Interface Adaptor and the INTEL 8253 interval timer, are included in the course hardware and extensively treated.

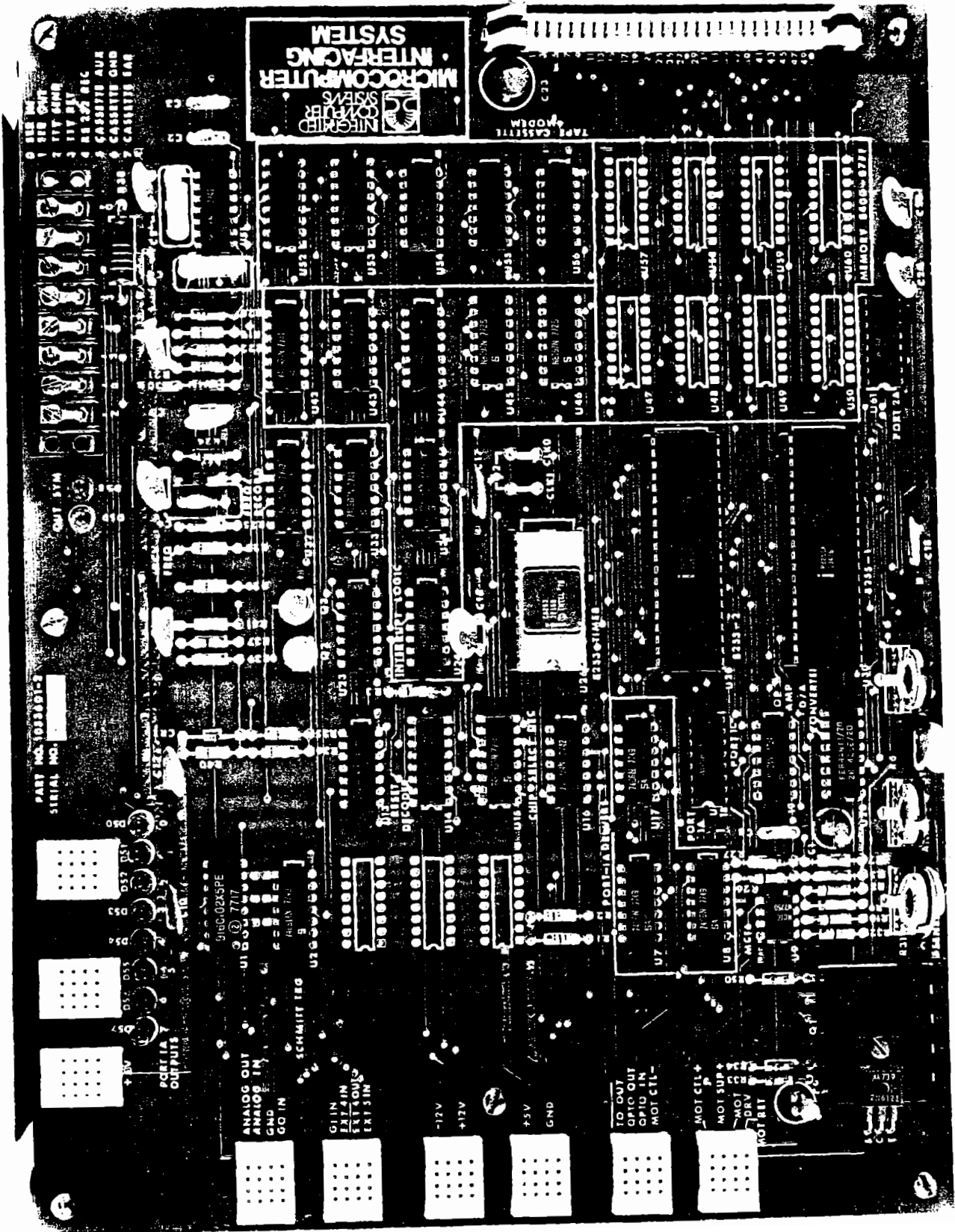
The remainder of this chapter gives an introduction to the hardware of the interface circuit board. Complete schematics are included here, but details of how various parts of the hardware operate are covered along with exercises in later chapters.

1.3 CABLES AND CONNECTIONS

The interface circuit board is connected to the Microcomputer Training System through a ribbon cable. One end plugs into a connector at the upper right edge of the MTS circuit board; the other end into a similar connector at the right edge of the Interface Training System. Be careful to align the cable so that Pin 1 on the AMTS (the right hand end) is connected to Pin 1 of the ITS (toward the top of the circuit board). Misconnection is likely to damage the circuits. To aid in aligning the cable correctly both connectors are keyed, and one end of the ribbon cable has a colored stripe. Power should be turned off while the connection is being made.

1.3.1 Power Connections

The required +5 volt and +12 volt power is supplied to the ITS through the ribbon cable from the MTS. These voltages are made available at tie blocks on the ITS for use in experiments. There is no negative supply required for any of the experiments described in this course. One transistor amplifier, suitable for driving a teletype or RS232 interface, does require a negative 12 volt supply to be connected.



Interface Training System

Figure 1-1

1.3.2 Signal Terminals

Signals used to connect the interface board to external devices, or to connect various functions together for experiments, are made through tie blocks at the left and top edges. The white plastic tie blocks each have four different signals, labelled next to the block. Each row in a block is a common line, making it easy to tie several signals together. Wires or component leads can be inserted directly into these tie blocks. One block, at the upper left corner, has +5 volts at all points to facilitate insertion of pullup resistors.

A row of screw terminals at the upper right provides for connections to serial ports. The tie blocks and screw terminals can be seen in Figure 1-1.

If you have purchased the Integrated Experiment Assembly Board, it should be plugged into the tie blocks at the left edge of the Interface Training Board. All of the tie block connections remain available, but the necessary parts and connections for most experiments in the course are preassembled on the Integrated Experiment Assembly.

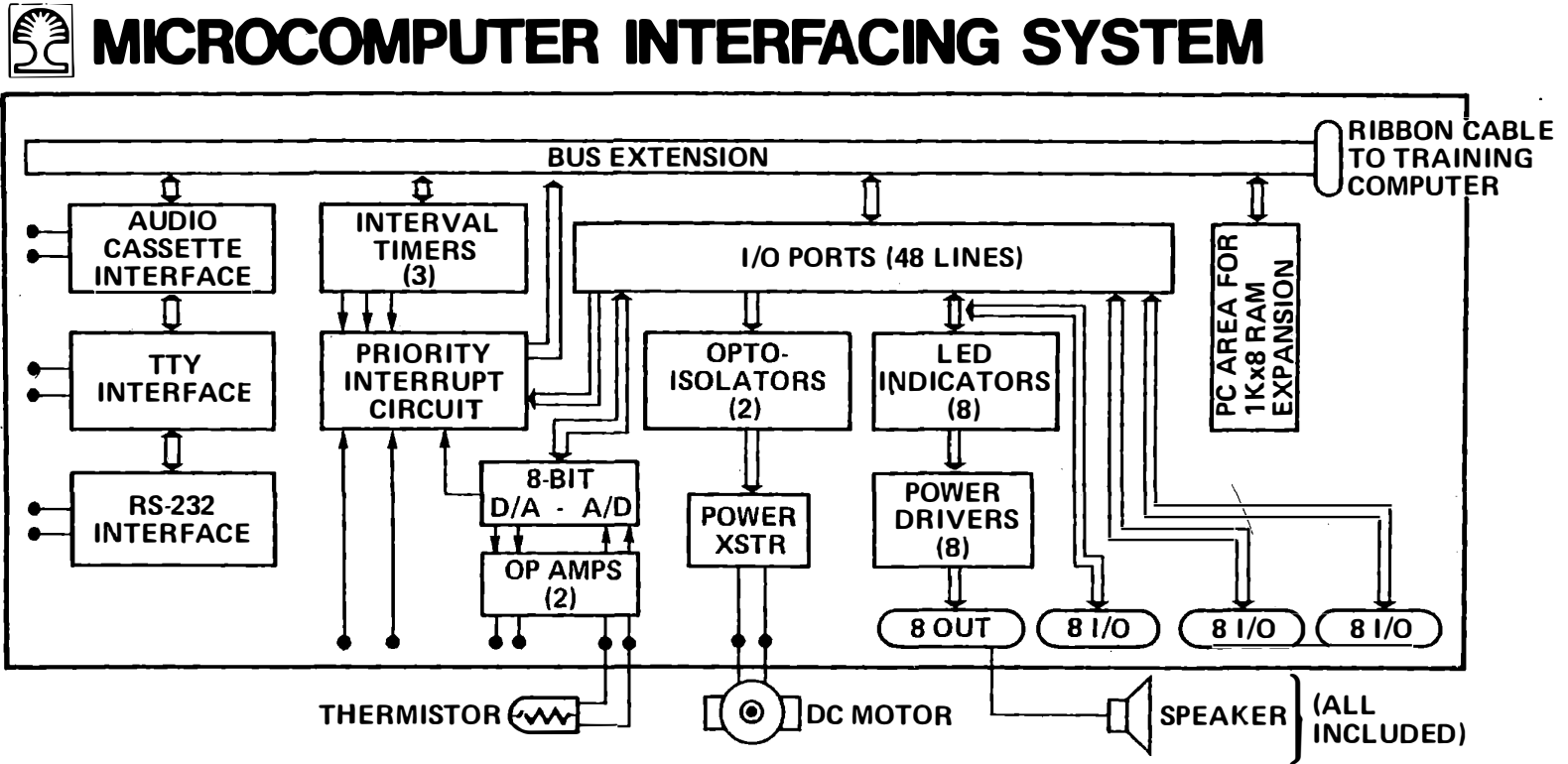


Figure 1-2

1.4 INTERFACE HARDWARE AND REFERENCES

An overall block diagram of the interface circuit board is shown in Figure 1-2. Various sections are shown in separate schematic diagrams and described in the chapters referred to below.

1.4.1 MTS Interface

Figures 1-3 and 1-4 list the signals that are brought out to the MTS via the 50 pin connector with their pin assignments in the connector head.

1.4.2 Added Memory

The interface circuit board provides space for memory expansion when the ITS is used with an early version of the MTS. THIS MEMORY SHOULD NOT BE USED WITH YOUR MICROCOMPUTER TRAINING SYSTEM. It occupies addresses that are filled with memory already supplied on the later version of the MTS.

HARDWARE INTERFACING AND REAL TIME PROGRAMMING

CONNECTOR PIN	SIGNAL NAME	CONNECTIONS ON INTERFACE BOARD
1	GND	
2	GND	
3	GND	
4	GND	
5	Vcc (+5 Volts)	
6	Vcc (+5 Volts)	
7	GND	
8	+12 Volts	
9	+12 Volts	
10	GND	
11	GND	
12	CLK \emptyset 2	(U26-18) (CLK1 Tiepoint) (CLK2 Tiepoint)
13	GND	
14	AB15	(U44-11)
15	AB7	(MEM-16)
16	AB6	(MEM-1)
17	AB5	(MEM-2)
18	AB4	(MEM-7) (U15-13)
19	AB3	(MEM-6) (U15-14)
20	AB10	(U44-10)
21	AB2	(MEM-5) (U16-9)
22	AB9	(MEM-14)
23	AB1	(MEM-4) (U26-20) (U28-8) (U30-8) (U13-5)
24	AB8	(MEM-15)
25	AB0	(MEM-8) (U26-19) (U28-9) (U30-9) (U13-4)

List of Interface Signals to MTS

Figure 1-3

HARDWARE INTERFACING AND REAL TIME PROGRAMMING

CONNECTOR PIN	SIGNAL NAME	CONNECTIONS ON INTERFACE BOARD
26	$\overline{\text{MEMR}}$	(U42-10)
27	$\overline{\text{RESET}}$	(U32-9) (U13-2)
28	POB0	(U54-6)
29	POC0	(U32-5)
30	$\overline{\text{MEMW}}$	(MEM-3) (U42-9)
31	$\overline{\text{INTA}}$	(U46-1) (U45-1)
32	DB7	(U45-9) (U15-2) (U60-11,12) (U26-1) (U28-7) (U30-7)
33	$\overline{\text{IOR}}$	(U28-5) (U30-5) (U26-22)
34	DB6	(U59-11,12) (U26-2) (U28-28) (U30-28) (U45-7)
35	DB5	(U58-11,12) (U26-3) (U28-29) (U30-29) (U45-5)
36	DB4	(U57-11,12) (U26-4) (U28-30) (U30-30) (U45-3)
37	DB3	(U50-11,12) (U26-5) (U28-31) (U30-31) (U46-9) (U14-3)
38	$\overline{\text{INTR}}$	(U33-3)
39	DB2	(U49-11,12) (U26-6) (U28-32) (U30-32) (U46-3) (U14-2)
40	DB1	(U48-11,12) (U26-7) (U28-33) (U30-33) (U46-5) (U14-1)
41	INTC	GND
42	DB0	(U47-11,12) (U26-8) (U28-34) (U30-34) (U46-7)
43	$\overline{\text{IOW}}$	(U28-36) (U30-36) (U26-23) (U15-3)
45, 46	-12 Volts	No Connections

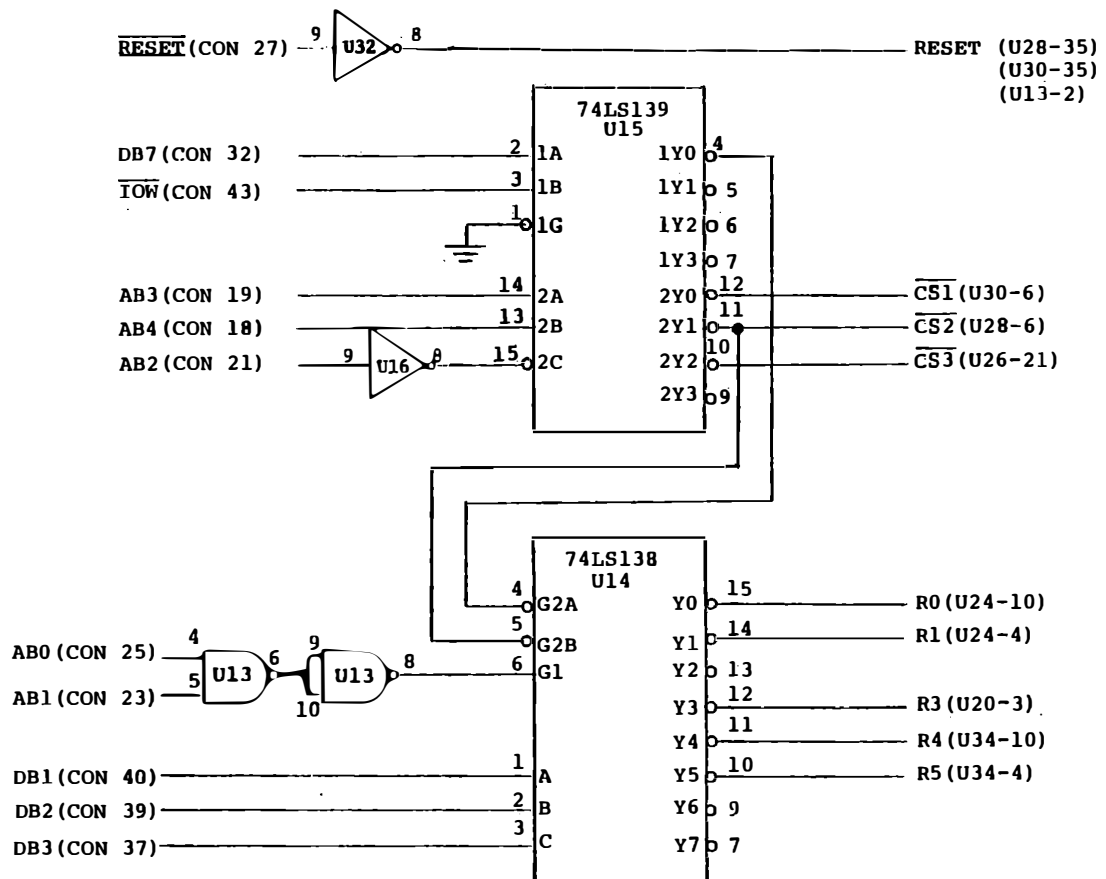
NOTE: (MEM -) refers to corresponding pins on all memory chips: U47, U48, U49, U50, U57, U58, U59, U60.

List of Interface Signals to MTS

1-9

Figure 1-4

HARDWARE INTERFACING AND REAL TIME PROGRAMMING



I/O Chip Selects and Interrupt Flip-Flop Reset

Figure 1-5

1.4.3 Chip Selects and Resets

Several bits of the address bus, control bus, and data bus are decoded to generate various chip select and reset signals needed on the interface board. The circuits are shown in Figure 1-5. Figure 1-6 gives a "truth table" listing the results of this decoding circuitry. The addressing of the interface board ports is described in Section 2-1. The purpose and use of the various reset signals are described in Section 2.7.

Figure 1-5 exemplifies the notation used in other schematics. For example, consider several signals at the upper left of Figure 1-5. $\overline{\text{RESET}}$ is received from the MTS through connector pin 27 (CON 27). It is inverted by one of the six inverters in a 7404, in chip U-32 on the circuit board, with input at pin 9 and output at pin 8. The output is high while the reset key is pressed so it is designated RESET, and goes to chips U-28 pin 35, U-30 pin 35, and U-13 pin 2. Chip and pin designations are used in debugging and circuit tracing and are of no interest in this course.

DB7 means Data Bus Line 7. $\overline{\text{IOW}}$ is low during an input/output write cycle. AB3 is Address Bus line 3.

HARDWARE INTERFACING AND REAL TIME PROGRAMMING

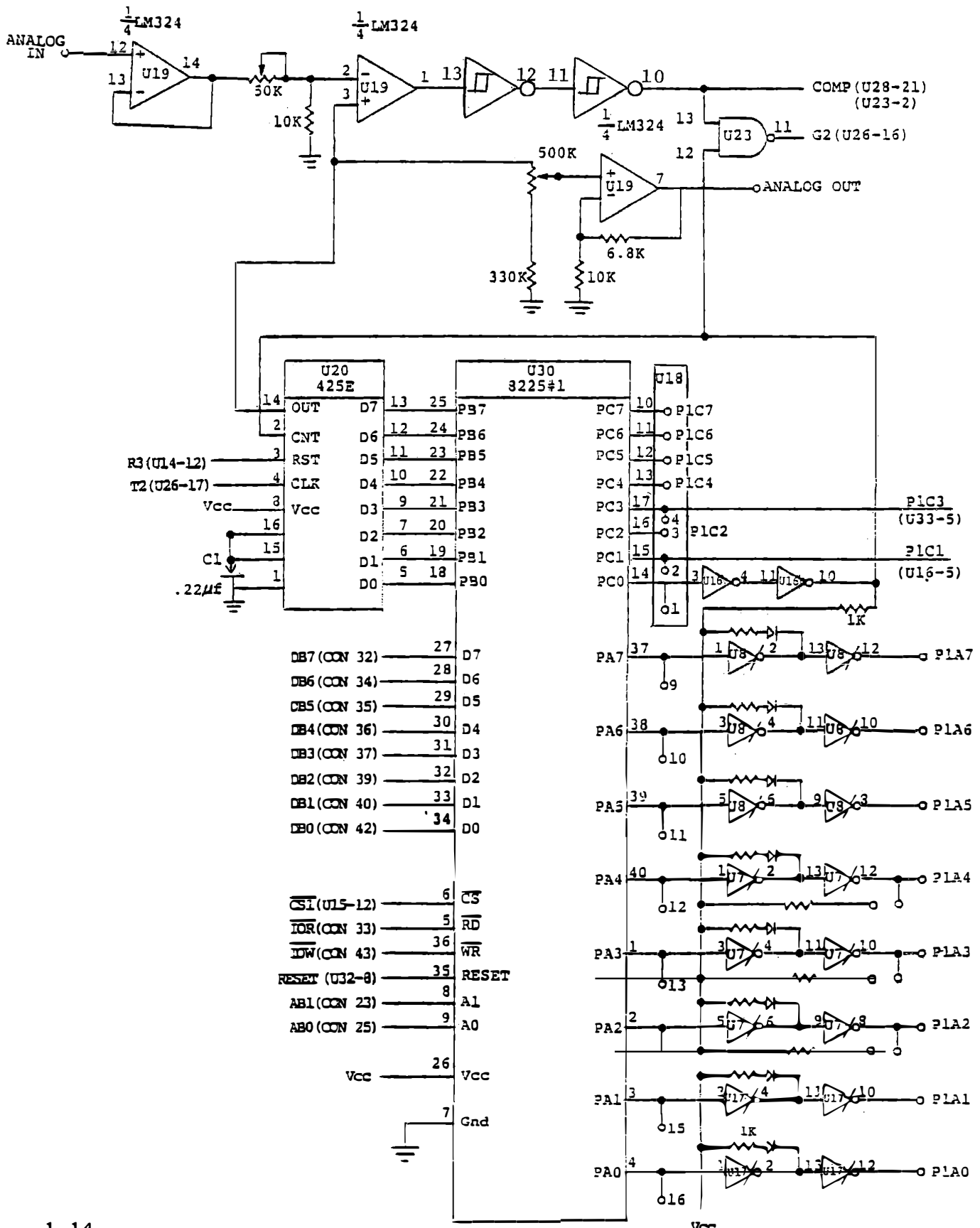
This page intentionally left blank.

$\overline{\text{IOW}}$	Low Address Bus								Data Bus								Chip Select			Resets					
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	3	2	1	5	4	3	1	0	
X	X	X	X	X	X	0	X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	1	1	No effect
X	X	X	X	1	0	1	X	X	X	X	X	X	X	X	X	0	1	1	1	1	1	1	1	1	Select Timer
X	X	X	X	0	0	1	X	X	X	X	X	X	X	X	X	1	1	0	1	1	1	1	1	1	Select Port 1
1	X	X	X	0	1	1	X	X	X	X	X	X	X	X	X	1	0	1	1	1	1	1	1	1	Select Port 2
0	X	X	X	0	1	1	0	0	X	X	X	X	X	X	X	1	0	1	1	1	1	1	1	1	Write Port 2A
0	X	X	X	0	1	1	0	1	X	X	X	X	X	X	X	1	0	1	1	1	1	1	1	1	Write Port 2B
0	X	X	X	0	1	1	1	0	X	X	X	X	X	X	X	1	0	1	1	1	1	1	1	1	Write Port 2C
0	X	X	X	0	1	1	1	1	1	X	X	X	X	X	X	1	0	1	1	1	1	1	1	1	Program Port 2
0	X	X	X	0	1	1	1	1	0	X	X	X	0	0	0	1	0	1	1	1	1	1	0	1	Bit Set/Reset 2C0
0	X	X	X	0	1	1	1	1	0	X	X	X	0	0	1	1	1	1	1	1	1	0	1	1	Bit Set/Reset 2C1
0	X	X	X	0	1	1	1	1	0	X	X	X	0	1	0	1	1	1	1	1	1	1	1	1	Bit Set/Reset 2C2
0	X	X	X	0	1	1	1	1	0	X	X	X	0	1	1	1	1	1	1	1	1	0	1	1	Bit Set/Reset 2C3
0	X	X	X	0	1	1	1	1	0	X	X	X	1	0	0	1	1	1	1	1	1	1	1	1	Bit Set/Reset 2C4
0	X	X	X	0	1	1	1	1	0	X	X	X	1	0	1	1	1	1	1	1	1	1	1	1	Bit Set/Reset 2C5
0	X	X	X	0	1	1	1	1	0	X	X	X	1	1	0	1	1	1	1	1	1	1	1	1	Bit Set/Reset 2C6
0	X	X	X	0	1	1	1	1	0	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	Bit Set/Reset 2C7

Truth Table for Chip Selects and Resets

Figure 1-6

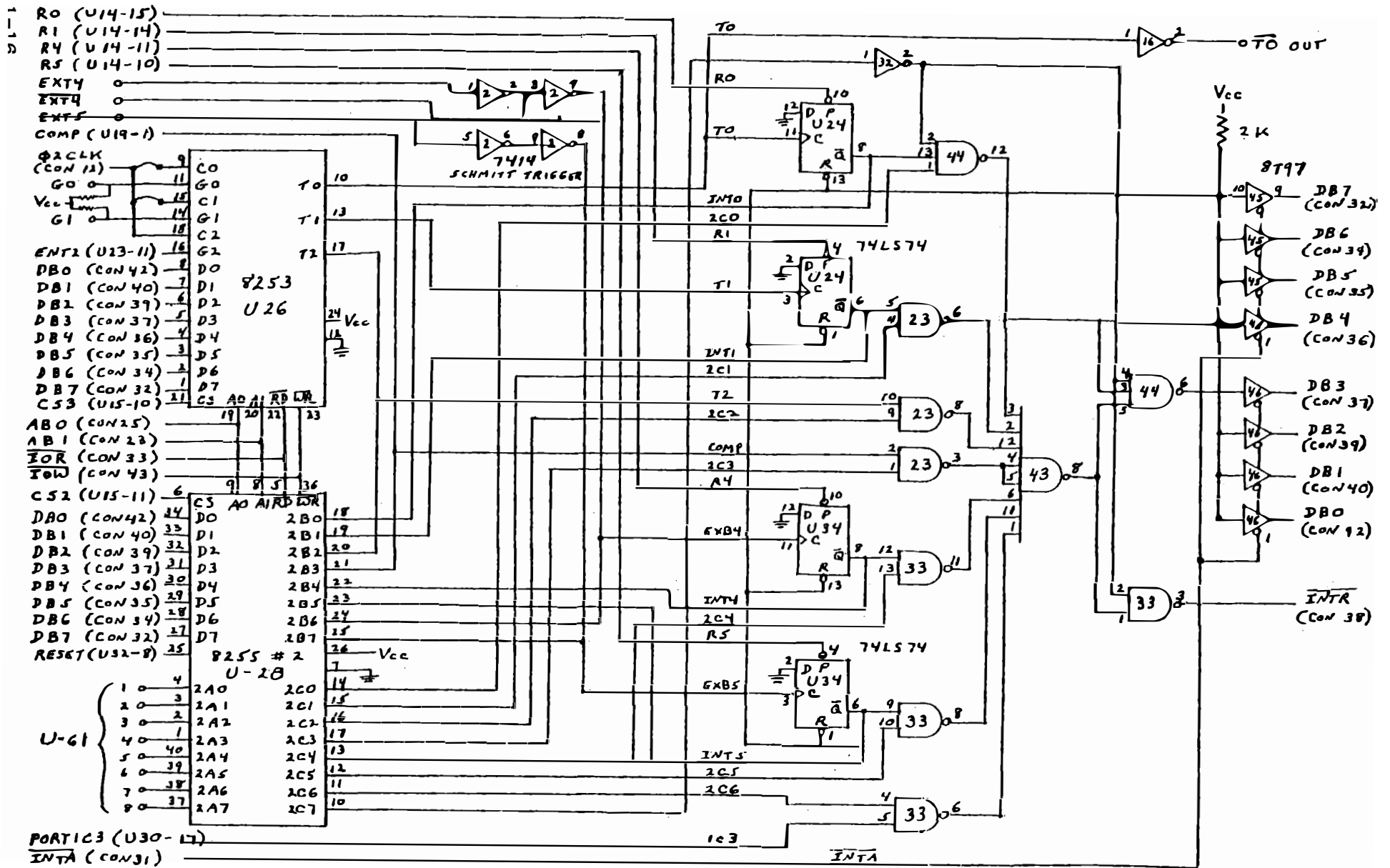
HARDWARE INTERFACING AND REAL TIME PROGRAMMING



Port 1 and A/D-D/A Converter

1.4.4 Port 1 and A/D - D/A Converter

The interface board includes two 8255 I/O devices, with three I/O Ports each (A, B & C). One of these, designated device (or Port) 1, drives the LED indicators at the top left of the interface board via Port 1A and provides connections for the digital to analog converter via Port 1B. It is shown in Figure 1-7. The discrete outputs are described in Sections 2.1 through 2.3. The digital to analog converter is described in Sections 4.5 and 4.6. The circuitry that makes it function as an analog to digital converter for input is the subject of Sections 5.3 and 5.4. Ports 1A and 1C are brought to a DIP socket (U-18) for general I/O applications.



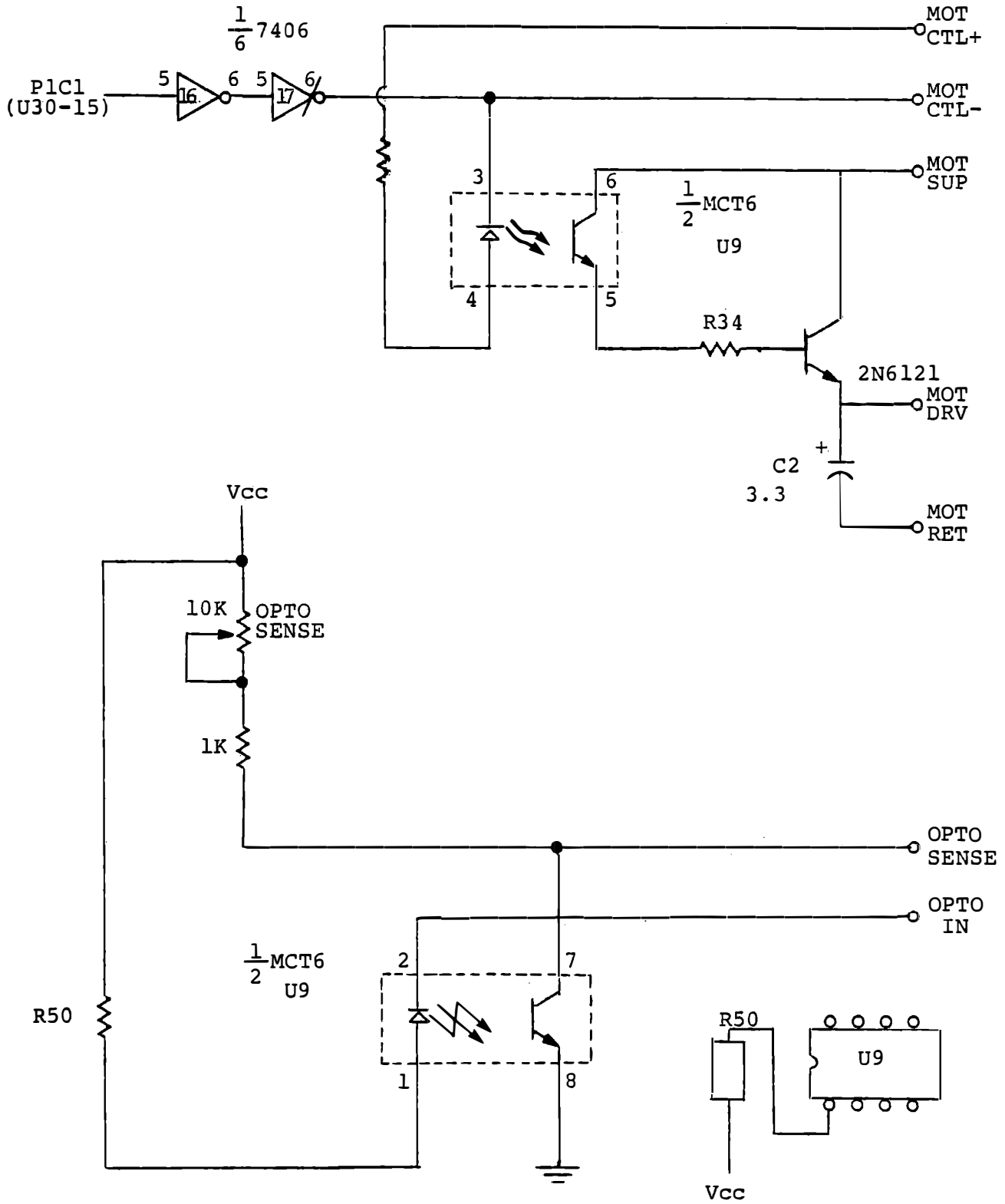
Vectored Priority Interrupt System

Figure 1-8

1.4.5 Interrupt System

The second 8255 I/O port is primarily devoted to the interrupt system shown in Figure 1-8 and described in Sections 2.7 through 2.9. The 8253 interval timer is closely tied to the interrupt system, so it is shown in the same schematic, but this device is so important (and so complex) that all of Chapter 3 is devoted to it.

HARDWARE INTERFACING AND REAL TIME PROGRAMMING



Optical Couplers and Power Driver

Figure 1-9

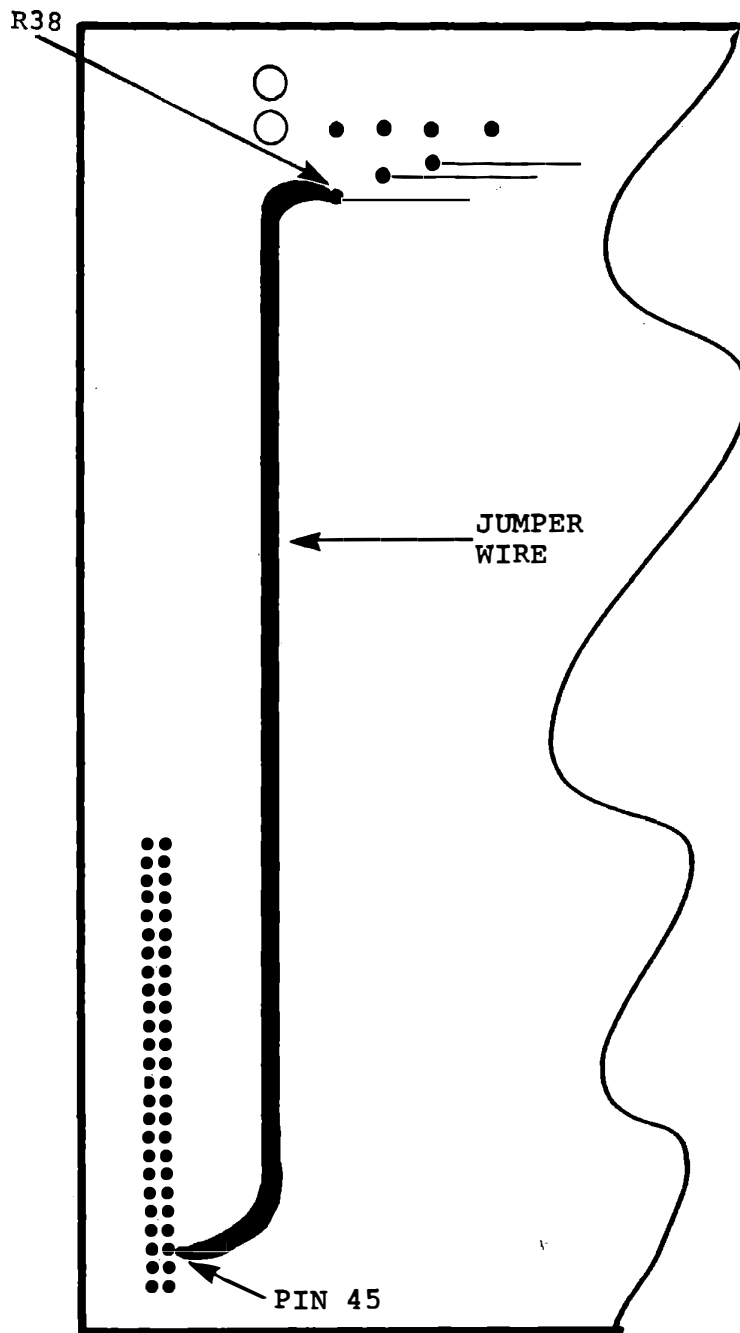
1.4.6 Optical Couplers and Power Driver

It is often necessary to provide electrical isolation between the computer and external equipment. Optical couplers use infrared light as a coupling medium for information while giving complete electrical isolation. An optical coupler (Monsanto MCT6) is provided on the circuit board. One coupler is used for output, driving a power transistor mounted on a heat sink. The other coupler is used for input. Figure 1-9 shows these circuits.

1.4.7 Serial Interface Circuit

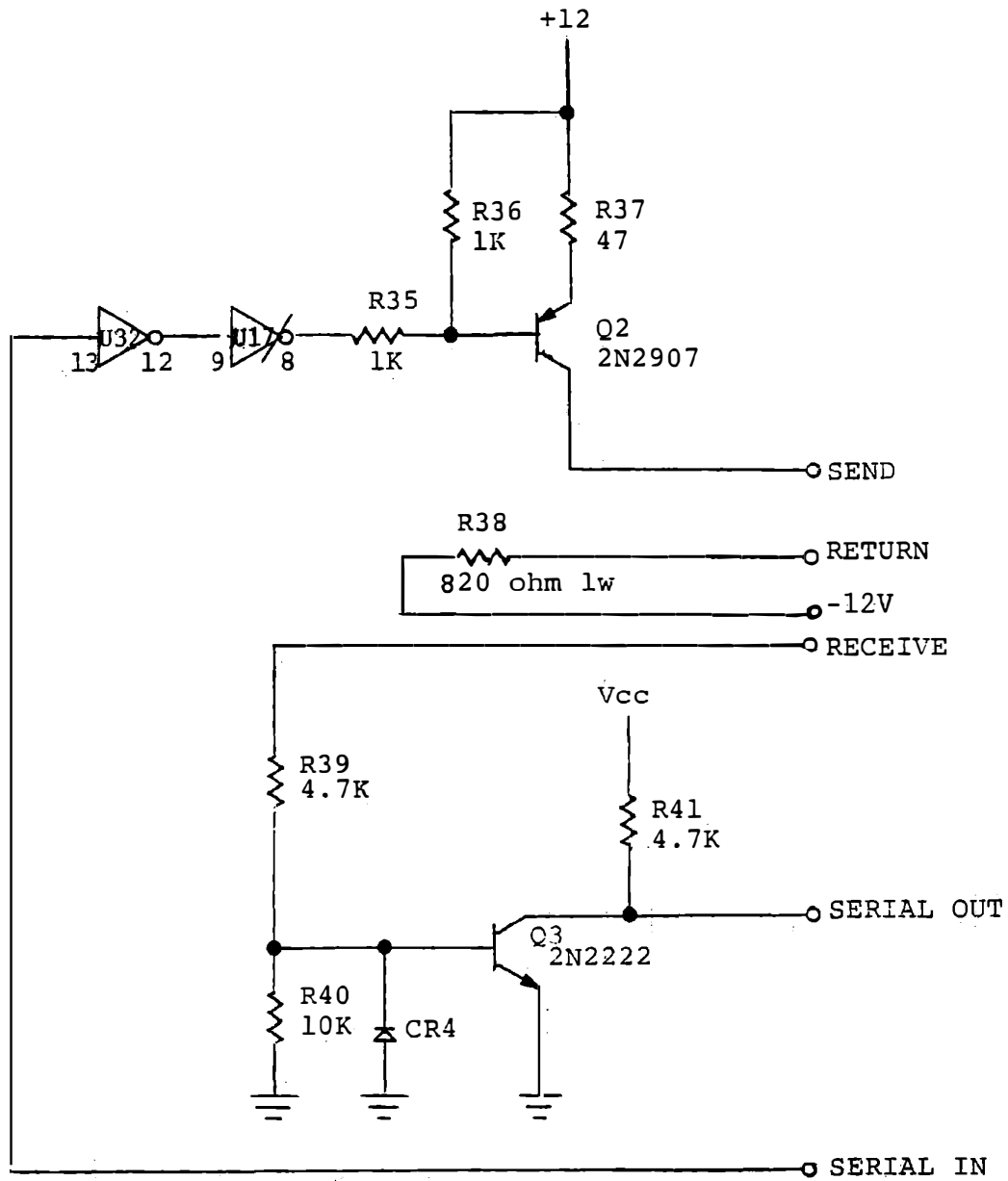
The circuits shown in Figure 1-10 can be used to connect the MTS to a teletype or a terminal such as a CRT that uses RS232 signal levels. The RS232 system's software and board connections are described in Appendix C.

Note that R38 is connected to the -12 volt tie block but not to the system -12 volt supply. To use this circuit, add a wire from R38 to pin 45 of the ribbon cable connector, which carries -12 volts.



Serial Interface Circuit
(Connecting R38 to -12 Volt Supply)

Figure 1-10a



Serial Interface Circuit

Figure 1-10b

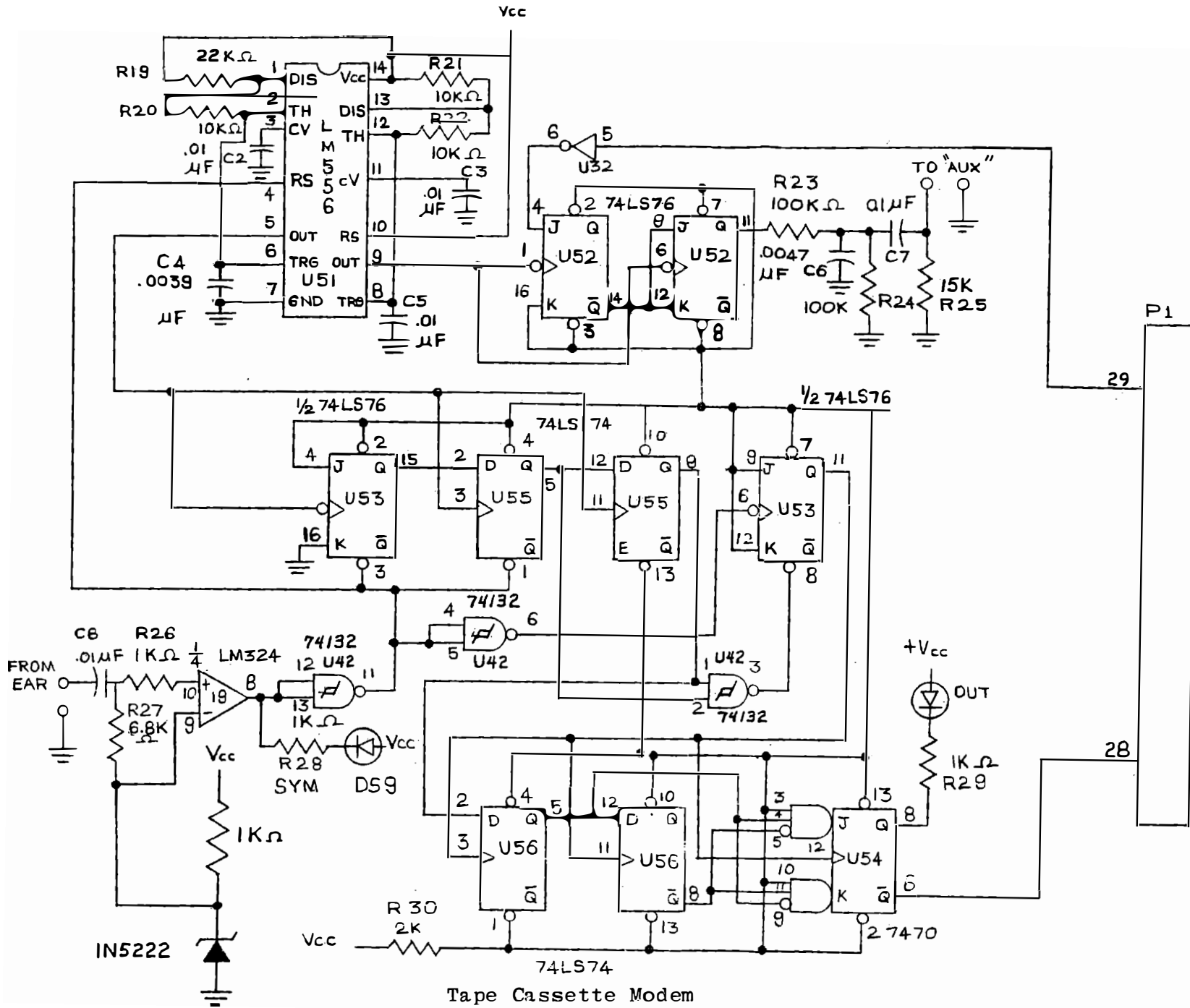
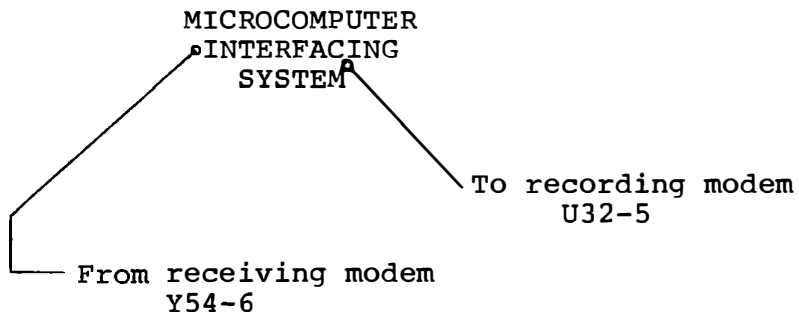


Figure 1-11

1.4.8 Tape Cassette Modem

The interface circuit board contains a duplicate of the tape cassette modem provided on the MTS, which was not available on the previous version of the MTS. The ITS modem will not ordinarily be used. It is shown in Figure 1-11. The digital input and output are carried through the ribbon cable but have no connections on the MTS circuit board. If the user wants to operate with two separate cassette recorders the signals should be picked up by soldering leads into feed-through holes on the ITS circuit board. Convenient locations are in the area with the silkscreened label "Microcomputer Interfacing System", as indicated below:



1.4.9 Tape Cassette Library

A cassette tape is provided with most of the programming exercise solutions and a few additional programs. It is described in Appendix B.

HARDWARE INTERFACING AND REAL TIME PROGRAMMING

1.5 USE OF PMTL OR INTEGRATED EXPERIMENT ASSEMBLY

If you have a Portable Microprocessor Training Laboratory, or if you have installed the Integrated Experiment Assembly on the Interface Training System, the necessary connections for the more complex experiments described in this course can be made by setting the slide switches appropriately. The early experiments all require that the slide switches be UP. Whenever the directions in this book indicate the need for an experimental setup, refer to the "Portable Microprocessor Training Lab - Selected Experiments" Manual, or to the "Experiment Assembly and Real-Time Firmware" Manual. There you will find instructions for all of the necessary switch settings.

MICROCOMPUTER INTERFACING WORKBOOK

CHAPTER 2

INPUT/OUTPUT AND INTERRUPTS

This page intentionally left blank.

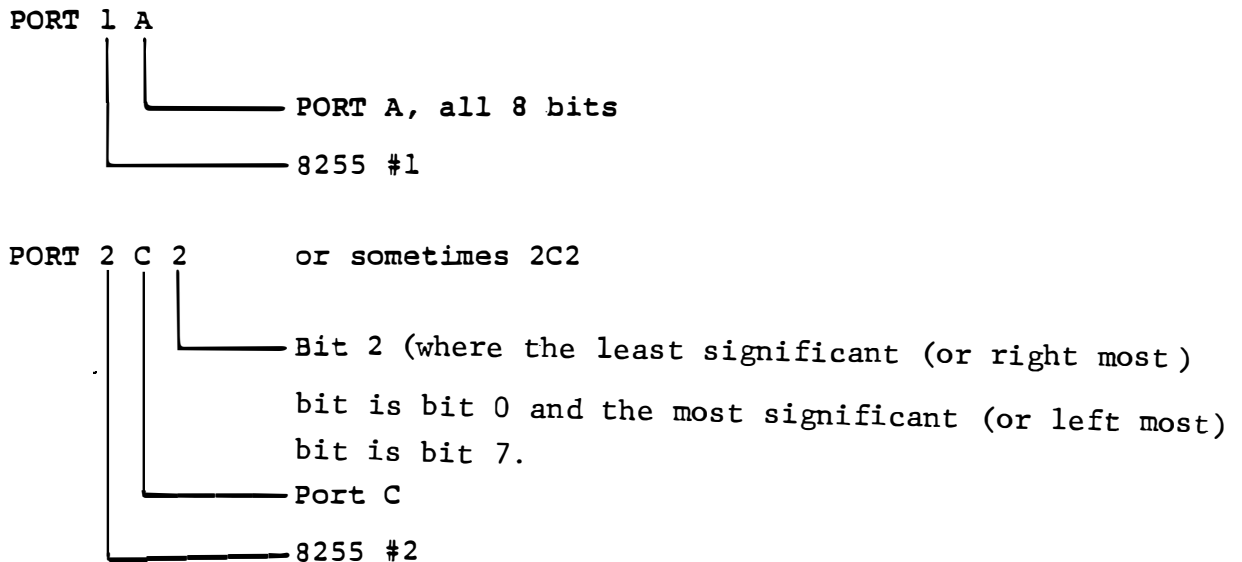
2. INPUT/OUTPUT AND INTERRUPTS

This chapter discusses the provisions made on the interface board for digital logic level inputs and outputs to the microprocessor. Interval timers, analog signals and optically isolated inputs are discussed in later chapters.

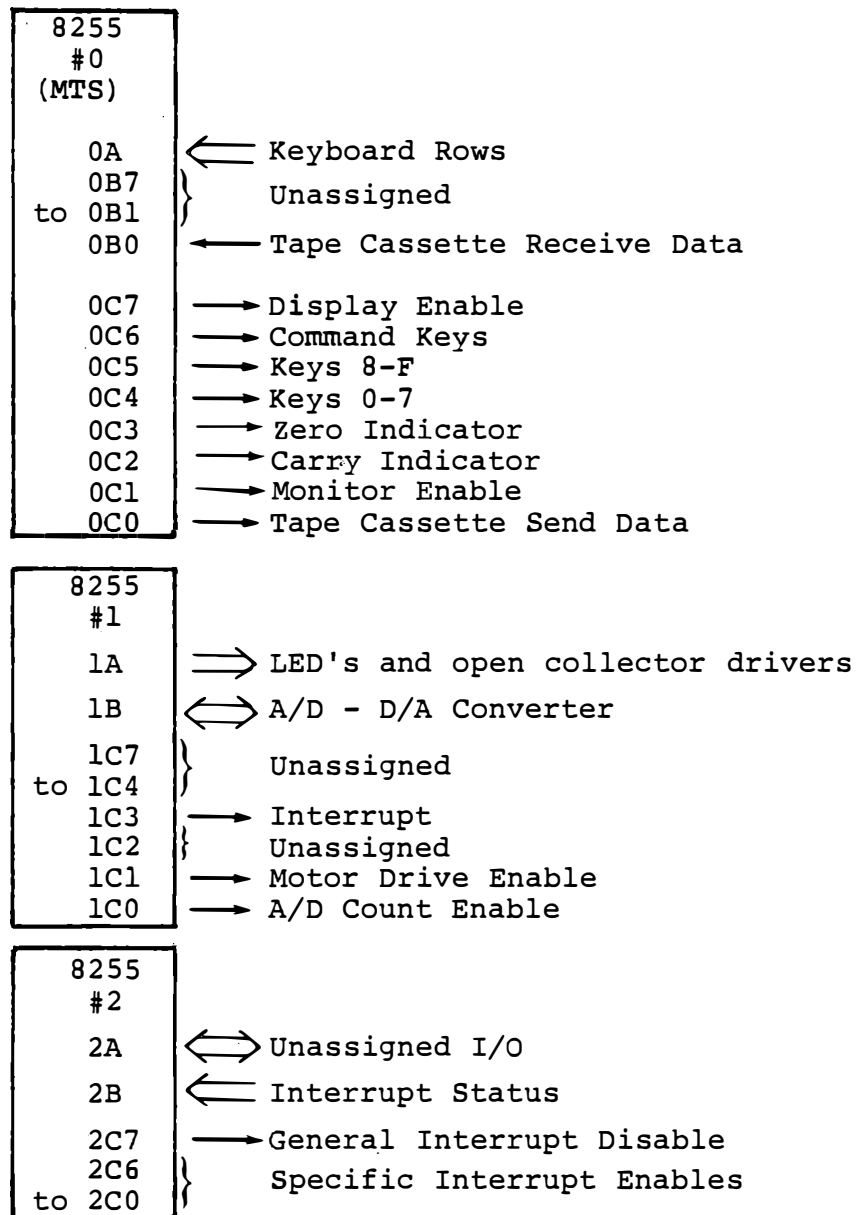
2.1 PORT ASSIGNMENTS AND ADDRESSES

The interface board includes two 8255 Programmable Peripheral Interface devices. Including the 8255 on the MTS board, a total of 72 bits of input/output is accessible to the 8080 microprocessor. In addition there is an Intel 8253 Interval Timer (see Chapter 3) which is addressed and programmed in much the same way as the 8255 ports. Figure 2-1 shows the port assignments. Figure 2-2 lists the port addresses and assignments and gives a list of programming control bytes suitable for each of the 8255's.

In this table and throughout the course we will refer to input ports by device number, port letter, and sometimes a bit number:



INPUT/OUTPUT AND INTERRUPTS



8255 I/O Port Assignments

Figure 2-1

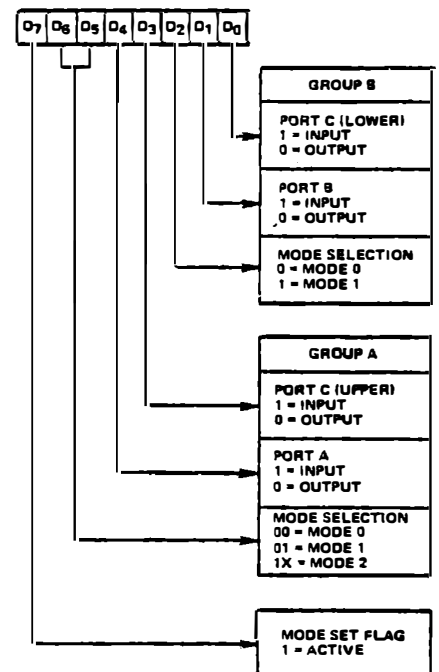
INPUT/OUTPUT AND INTERRUPTS

PORT ADDRESSES AND ASSIGNMENTS

ADDRESS	PORT NAME	FUNCTION	SPECIAL ASSIGNMENTS FOR 0C AND 1C		
00	PORT 0A	MTS Keyboard Input	0C7	Display Control	(1 = On)
01	PORT 0B	Unassigned except 0B0	0C6	Enable Command Keys	(0 = On)
02	PORT 0C	See column at right	0C5	Enable Keys 8-F	(0 = On)
03	CNT 0	Control Port for MTS 8255	0C4	Enable Keys 0-7	(0 = On)
04	PORT 1A	LED and Driver Outputs	0C3	Zero Indicator	(1 = On)
05	PORT 1B	D/A Output or A/D Input	0C2	Carry Indicator	(1 = On)
06	PORT 1C	See column at right	0C1	Monitor Enable	(1 = On)
07	CNT 1	Control Port for 8255 # 1	0C0	Cassette Modem Out	
0C	PORT 2A	Unassigned	0B0	Cassette Modem In	
0D	PORT 2B	Interrupt Status Input	1C7-4	Unassigned	
0E	PORT 2C	Interrupt Enable Output	1C3	Interrupt (If Enabled by 2C6)	
0F	CNT 2	Control Port for 8255 # 2	1C2	Unassigned	
14	TIM 0	Timer 0	1C1	Motor Drive Buffer	(0 = On)
15	TIM 1	Timer 1	1C0	D/A Control (1 = Automatic A/D)	
16	TIM 2	Timer 2			
17	TIM CT	Control Port for 8253			

8255 PROGRAMMING CONTROL BYTES (WRITE TO 8255 CONTROL PORT)

CONTROL BYTE	PORT A	PORT B	PORT C0-C3	PORT C4-C7	USE WITH 8255 #		
					0	1	2
80	Out	Out	Out	Out		D/A	
81	Out	Out	In	Out		●	
82	Out	In	Out	Out		A/D	*
83	Out	In	In	Out		A/D	
88	Out	Out	Out	In		D/A	
89	Out	Out	In	In		●	
8A	Out	In	Out	In		A/D	
8B	Out	In	In	In		A/D	
90	In	Out	Out	Out	*	D/A	
91	In	Out	In	Out	*	●	
92	In	In	Out	Out	*	A/D	*
93	In	In	In	Out	*	A/D	
98	In	Out	Out	In		D/A	
99	In	Out	In	In		●	
9A	In	In	Out	In		A/D	
9B	In	In	In	In		A/D	



* Generally only these control bytes should be used for normal operation.
 ● Forbidden configurations

Mode Definition Format

Figure 2-2

INPUT/OUTPUT AND INTERRUPTS

2.2 PROGRAMMING AND USING THE 8255

At system reset all ports of all 8255's are automatically set to input Mode 0. They can be used this way or programmed to other configurations by writing a control byte to the control port of the desired 8255. The monitor program automatically re-configures the 8255 on the MTS board such that ports 0A and 0B are input and 0C is output. It accomplishes this by writing 92 to the control register.

```
3E      MVI A,92
92                                0: A = IN; B = IN; C = OUT
D3      OUT CNT0                  Output A to 8255 #0 Control Register
03                                (Address 03)
```

The interface board 8255's must be set to the desired modes by your program. The first programs we will develop require output in all three ports of 8255 1. Figure 2-2 gives 80 as the required control byte:

```
3E      MVI A,80                  Load A with control byte to
80                                make device 1: A = OUT, B =
                                OUT, C = OUT
D3      OUT CNT1                  Set 8255 #1 Control Register
07                                (Address 07)
```

INPUT/OUTPUT AND INTERRUPTS

Because 8255 #2 is largely committed to the interrupt system it usually is programmed for input at port 2B and output at port 2C. Port 2A may be input or output but must be in Mode 0, the normal direct I/O mode. Most programs developed in this course use the interrupt system, so in general programs should contain (again from Figure 2-2):

```
3E      MVI A,92      Device 2: A out, B in, C out
                               (MVI A,82 may also be used)
92
D3      OUT CNT2
0F
```

With the 8255 ports programmed, data can be read from or written to the ports by IN or OUT instructions, for example:

```
DB      IN  PORT0A      (A) <--- Port 0A
00                               (Keyboard Input)
D3      OUT  PORT1A      Port 1A <--- (A)
04                               (Interface Board's LED's)
```

INPUT/OUTPUT AND INTERRUPTS

EXERCISE

If you write a program containing all of the instructions listed so far, terminated with a jump back to the input instruction as in Figure 2-3, the LED indicators on the interface board will show the keyboard input data.

Figure 2-4 shows the keyboard connections to ports OC and OA. Programming a port to output mode automatically sets its outputs low. Therefore, (from Figure 2-4), if a key is pressed the corresponding bit in port OA is made low. For example, if the MEM key is depressed, then port OA0 (which was pulled high by the resistor to Vcc) is now pulled low through the MEM key and port OC6. (The monitor scans the keyboard by alternately making ports OC4, OC5 and OC6 low or 0, thus identifying the key pressed).

With the program shown, port OC is programmed for output, so all keys are enabled (OC4, OC5 and OC6 all low). Therefore 0, 8 and MEM will show the same output. The input to a bit of port OA is high if no key in that column is pressed; if a key is pressed and the bit of port OC for that row is low, then the input is low. (Review Course 525, Section 8.1 if a more detailed description is needed).

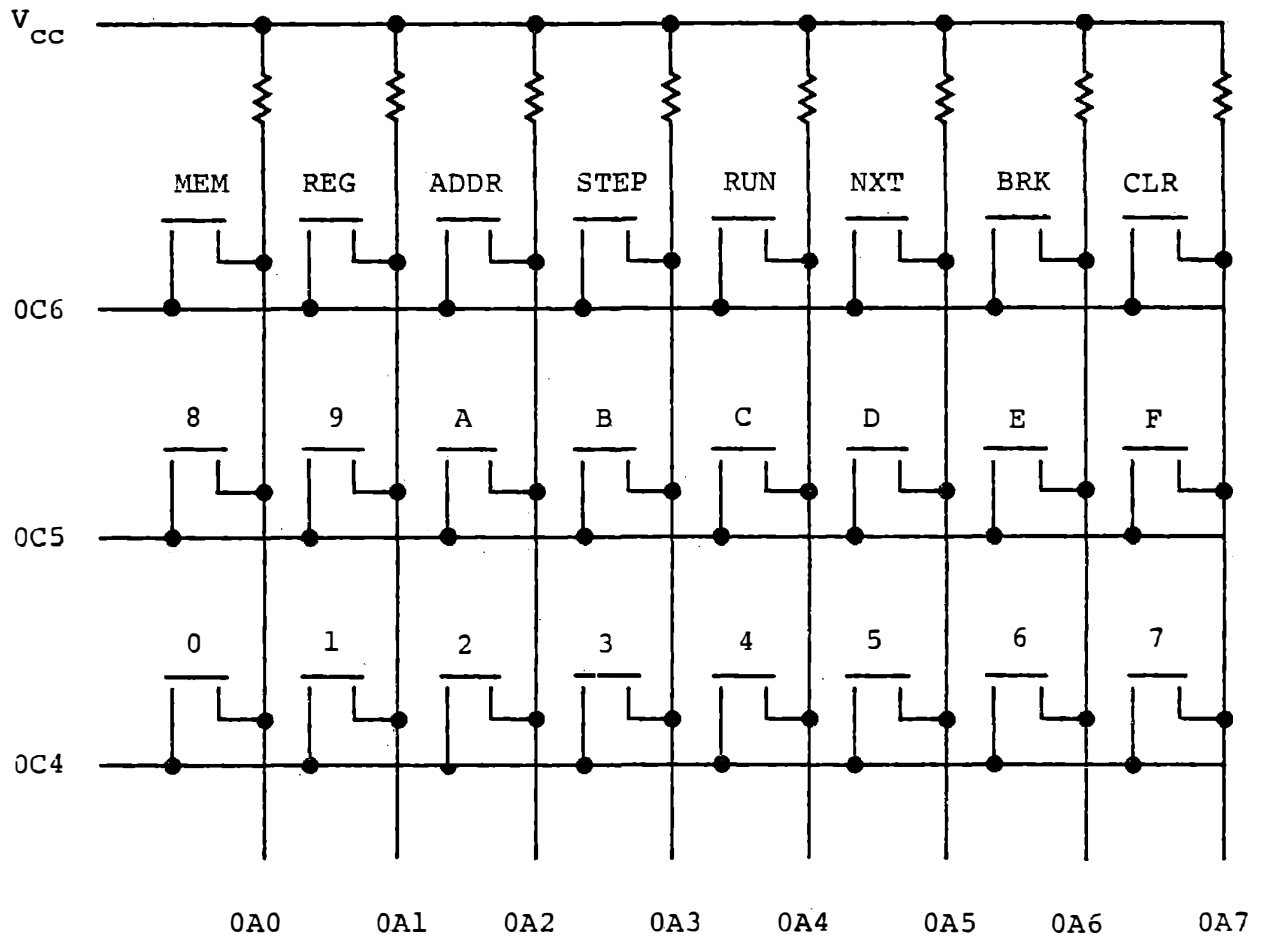
PROGRAMMING THE 8255's

A D D R		CODE							
CODING SHEET	8	20	0	3E	MVI	A,	92	Program MTS 8255	
			1	92				A in Bin Cout	
			2	D3	OUT	CNT	D		
			3	03					
			4	3E	MVI	A,	80	Program 8255 #1	
			5	80				A out Bout Cout	
			6	D3	OUT	CNT	1		
			7	07					
			8	3E	MVI	A,	92	Program 8255 #2	
			9	92				A in Bin Cout	
MICROCOMPUTER TRAINING SYSTEM		A	D3	OUT	CNT	2			
		B	0F						
		820	C	DB	IN	PORT	0A	Read Keyboard	
			D	00					
			E	D3	OUT	PORT	1A	Display in LED's	
			F	04					
	INTEGRATED COMPUTER SYSTEMS	8	21	0	C3	TMP	820	C	Back to input
				1	0C				
				2	82				
				3					
			4						
			5						
			6						
			7						
			8						
			9						
		A							
		B							
		C							
		D							
		E							
		F							
	8	0							
		1							
		2							
		3							
		4							
		5							
		6							
		7							
		8							

Figure 2-3

INPUT/OUTPUT AND INTERRUPTS

This page intentionally left blank.



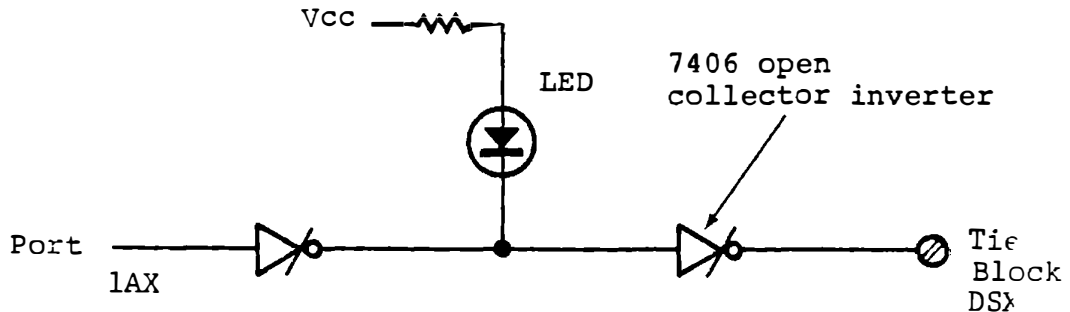
MTS Keyboard Configuration
and Port Assignment

Figure 2-4

INPUT/OUTPUT AND INTERRUPTS

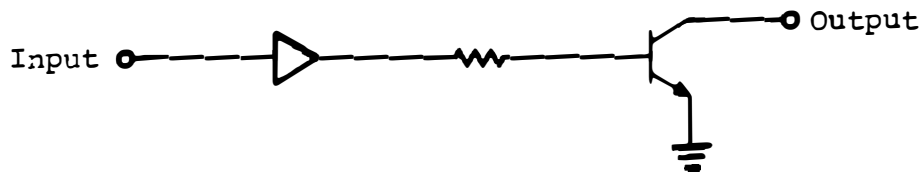
2.3 PORT 1A LED'S AND DRIVERS

Port 1A (address 04) drives eight sets of open collector inverters and LED's

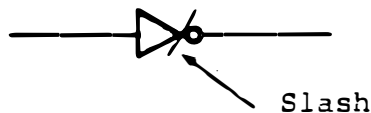


Each bit of the port drives an identical circuit. The LED indicates the state of the output, i.e. illuminated if a one is output. The terminal block output follows the port. The state is low if a zero is output and open if one is output.

An open collector buffer is a TTL amplifier whose output comes from a transistor with no internal connection to its collector. It is approximately equivalent to the circuit below. When the input is a logic 1 (greater than 2.4V) current flows into the transistor, thus turning it on and effectively connecting the output to ground. However, when the input is logic 0 (0V), no current flows into the transistor's base. Therefore, it is off and the output is "floating" (i.e., connected to neither ground nor +5V).

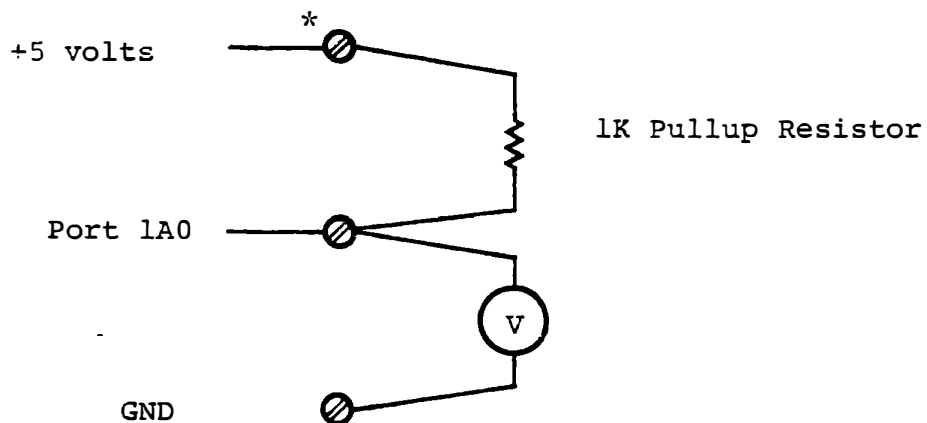


An open collector inverter is shown on a schematic diagram by:




INPUT/OUTPUT AND INTERRUPTS

The slash indicates an open collector. Note that the open collector output gives a signal only if it is pulled up through some load or pullup resistor to a positive voltage, which may be as high as 30 volts. The output is capable of sinking 40 ma to 0.7 volts. Connect a voltmeter from one of the port 1A output drivers to ground. It will show 0 volts whether the LED is on or off. Now connect a pullup resistor to +5 volts, and the voltmeter will display either 0V or 5V depending on the state of the Port 1A0 output bit:



For output bits 1A2, 1A3, and 1A4 (marked DS2, DS3, and DS4 on the ITS board) pullup resistors (in U1) are available on the circuit board, but not connected. They can be connected by soldering jumpers between two pads in front of the LED's for those three bits.

*NOTE: Throughout this text the illustrations represent ITS board Tie Block connect points with the symbol . These refer to one of the labelled rows within the white Tie blocks.

INPUT/OUTPUT AND INTERRUPTS

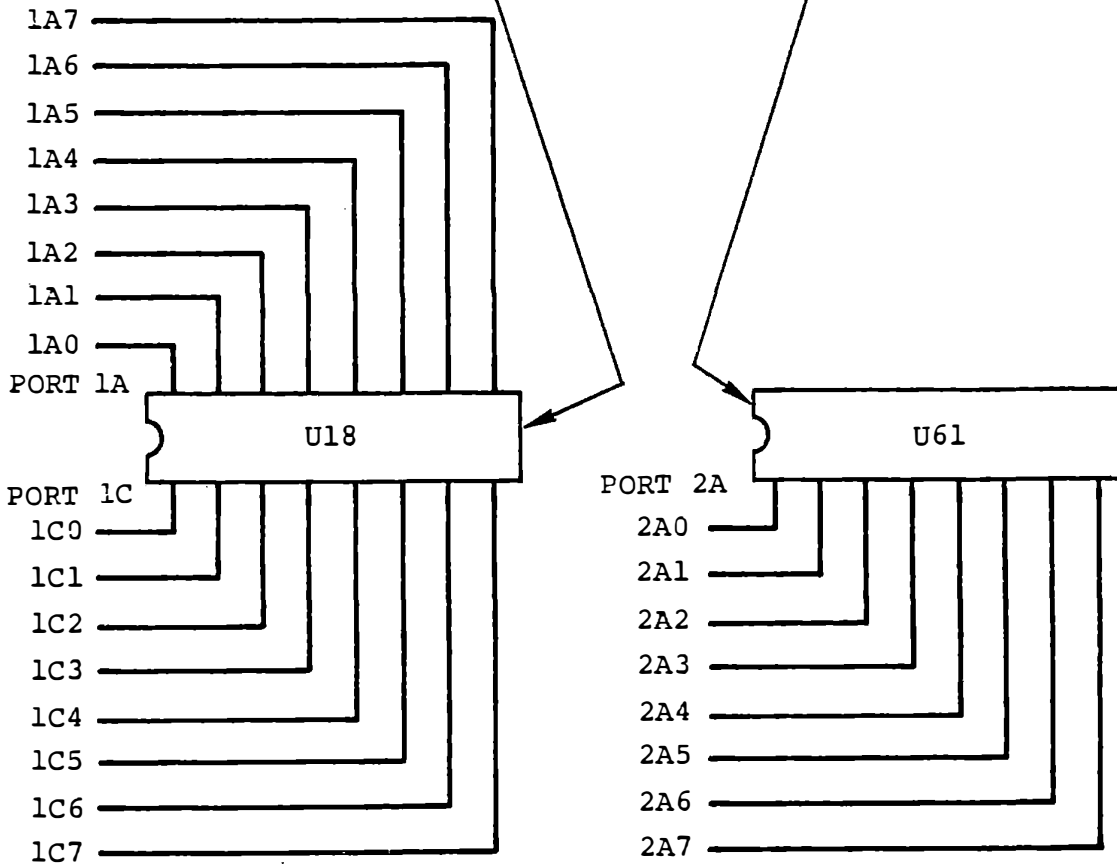
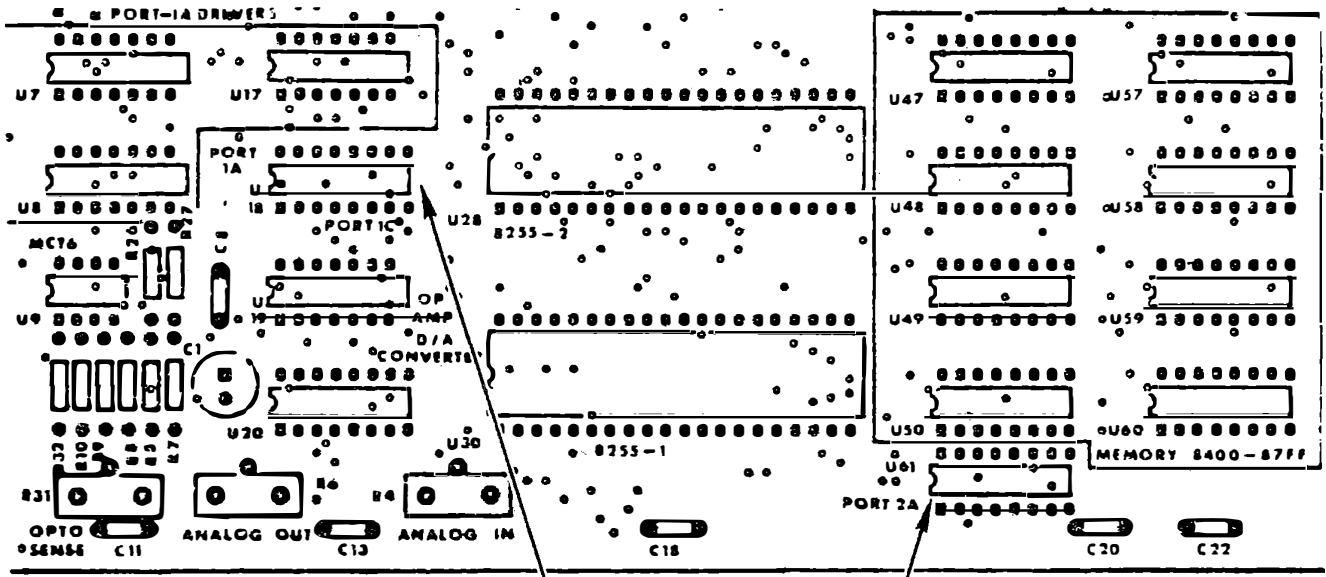
2.4 MTS DISPLAY

The seven segment displays on the MTS are operated by a direct memory access system. Whatever data are written to memory locations 83F8 through 83FF are automatically displayed in the eight digits. (Review Course 525, Section 8.3, for more detail.) The DMA channel must be enabled by a high output at port 0C7. Since programming a port to output automatically sets all bits low, the display was disabled when you programmed the MTS 8255 #0. Prove this by adding STA 83F8 before the jump instruction in your program. Even though you have written the same data to the DMA display area of the MTS memory as you wrote to the LED's, the display will remain blank.

A good way to turn the display on is by use of the bit set/reset function of the 8255. This allows a single bit of port C to be changed without affecting any other bit. Enter this at the end of your program (after OUT PORT1A):

```
32      STA      83F8      Write keyboard data to display
F8
83
3E      MVI      A,0F      Set bit 7 in port 0C
0F
D3      OUT      CNT0
03
C3      JMP      820C      Jump back to input instruction
0C
82
```


INPUT/OUTPUT AND INTERRUPTS



Input/Output Connections

Figure 2-5

2.5 INPUT/OUTPUT CONNECTIONS

In addition to the terminal block outputs driven by the buffers, port 1A, port 1C, and port 2A are directly connected to empty DIP sockets, shown in Figure 2-5. A cable that plugs into this socket can be obtained (available from Augat as part number 7P16-3T24-1). This allows connection of these ports to any suitable device for input or output. None of the experiments described here require these connections, but when you develop interfaces to other equipment they may be needed.

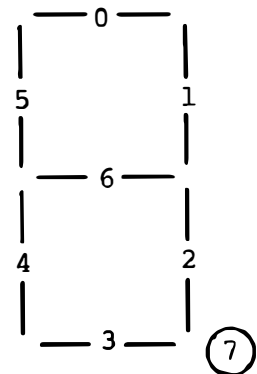
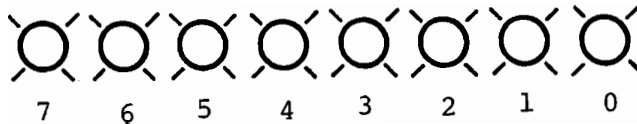
INPUT/OUTPUT AND INTERRUPTS

2.6 EXTERNAL INPUTS 4 AND 5

Two terminal block connections labelled EXT 4 and EXT 5 are provided for the external inputs needed in many of the experiments in this course. These inputs are part of the interrupt system, which will be described in Section 2.7. They can also be read as single input bits. They are connected to port 2B, bits 6 and 7 respectively (see Figure 2-7).

EXERCISE:

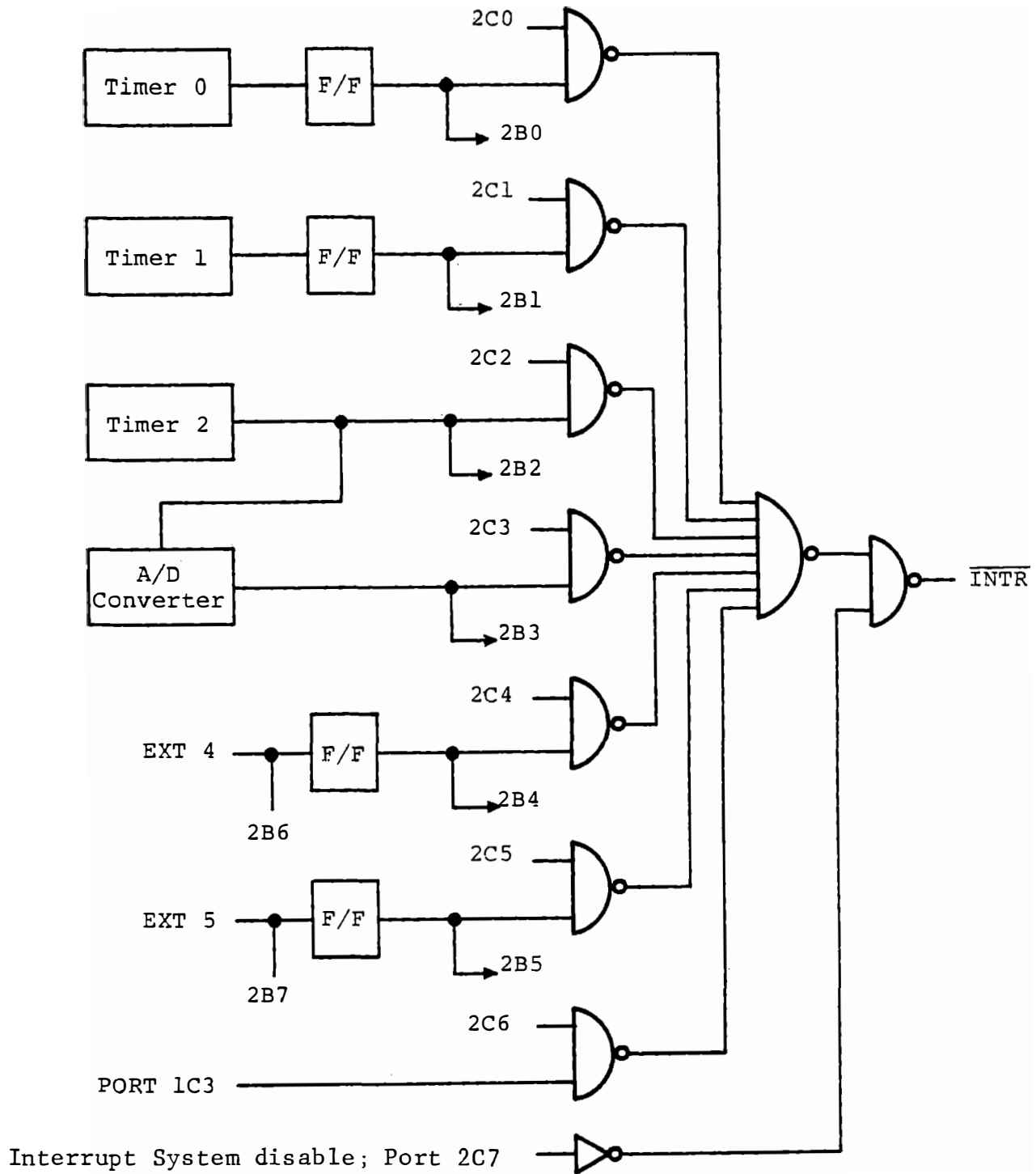
Read the external input bits and display them with the LED's. Change the program of Section 2.4 to read Port 2B instead of from Port 0A. (Refer to Figure 2-2 to find the address). Since the EXT 4 and EXT 5 inputs are connected to Port 2B bits 6 and 7, the modified program will repeatedly input these bits and display them (along with random data in the other bit positions) in both the LED's and in the left digit of the MTS display.



INPUT/OUTPUT AND INTERRUPTS

Connect a clip lead to the EXT 4 input and touch it to ground to see bit 6 change in the display. TTL circuits demand sharp rising and falling edges. To ensure suitable signals and to prevent spurious noise from causing interrupts, EXT 4 and EXT 5 are brought into Schmitt Trigger inverters (in chip U2, a 7414, see Figure 2-7). The signals are then inverted again and used to clock flip-flops in the interrupt system, discussed in the next section. The inverted signal, EXT 4, is available at a terminal labelled EXT 4 OUT. Connect this to EXT 5 IN, so that EXT 5 will be inverted from EXT 4. Now when you connect EXT 4 input to ground both signals will change. They will be displayed (using the program of Section 2.4) in bits 6 and 7 of the LED's.

INPUT/OUTPUT AND INTERRUPTS



Inerrupt System - Partial Diagram

Figure 2-6

2.7 INTERRUPT FLIP-FLOPS AND ENABLES

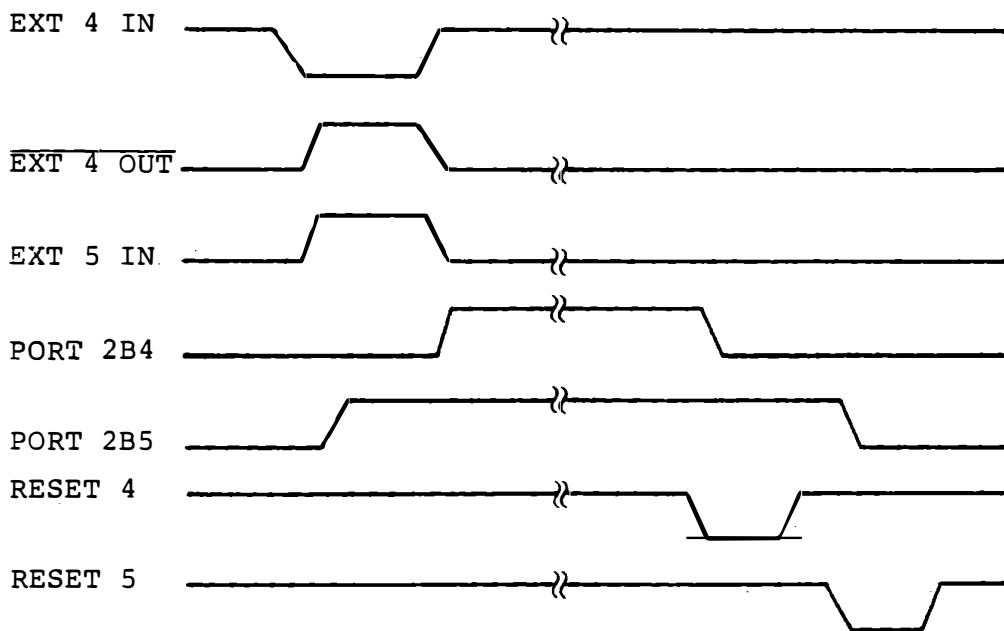
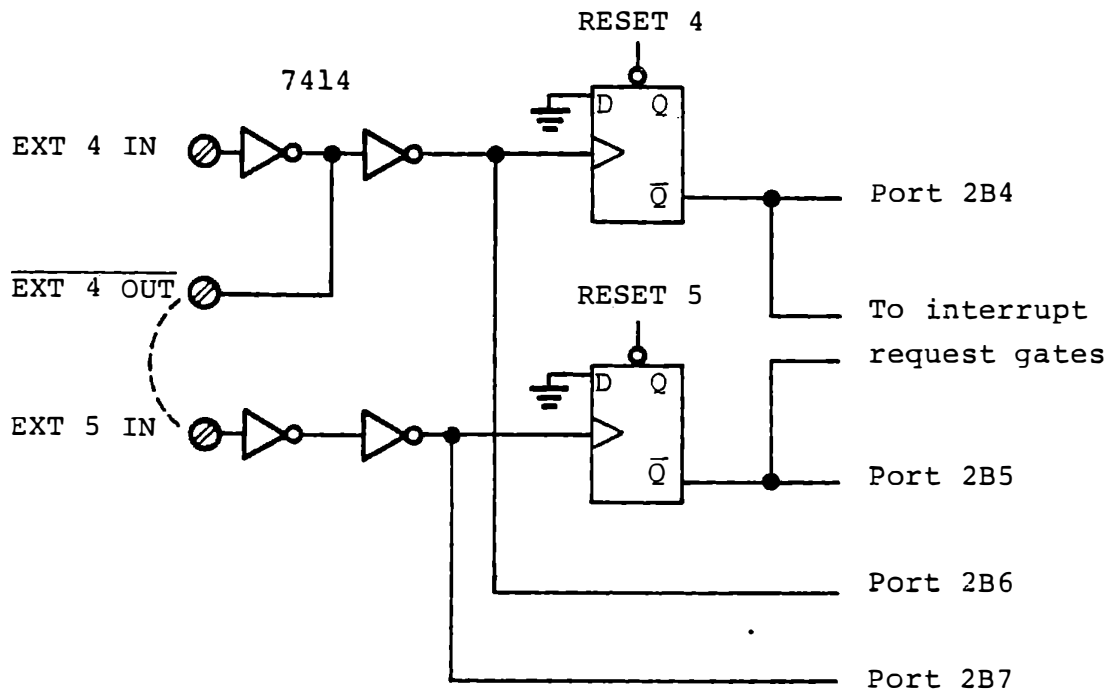
Most of the experiments in this course use the interrupt capability of the 8080. The student should be familiar with Section 8.4 through 8.6 in Course 525 . A review of those sections may be advisable at this point.

2.7.1 Interrupt Sources

The MTS will accept repeated interrupts generated by its own hardware when the AUTO/STEP toggle switch is set to STEP. We will also refer to these as "monitor interrupts" since their purpose is to invoke monitor functions such as single stepping or breakpoints. In addition, and independent of the AUTO/STEP switch, the MTS will accept interrupts generated by the interface board. (Once an interrupt has occurred, or if a DI instruction is executed in the program, no other interrupt can occur until an EI instruction and one following instruction have been executed).

Figure 2-6 is a partial diagram of the interrupt logic of the interface board. Interrupts can occur in response to signals generated by the interval timers (Chapter 3), by strobed input or output using port 1A, by the EXT 4 and EXT 5 input ports, and by the Analog to Digital converter.

INPUT/OUTPUT AND INTERRUPTS



EXT 4 and EXT 5 Connections and Signals

Figure 2-7

2.7.2 Interrupt Flip-Flops

The purpose of an interrupt system is to provide for processing of an occasional and perhaps fleeting event while allowing the computer to carry on other tasks that can be put aside temporarily when an interrupt occurs. To permit the processor to recognize a possibly very brief signal, the rising edges of the EXT 4 and EXT 5 inputs set flip-flops, which actually provide the interrupt data to the processor. Each flip-flop is reset only by a specific command from the processor under program control. This insures that each interrupt signal is retained until it has been processed.

Figure 2-7 shows the connections of these flip-flops in detail. To save components the interface board uses negative logic here. The flip-flop is actually set to 0 by the input signal and preset to 1 by the reset command. Since the inverted output of the flip-flop is used, the signal becomes true (=1) at the input rising edge and false (=0) at reset. These "reset" signals are software generated as described below.

Similar flip-flops are connected to the outputs of timer 0 and timer 1, to allow interrupts on narrow pulses from these timers. The interrupts from the A/D converter and port 1C3 are latched by their sources so flip-flops are not needed. Only timer 2 output is unlatched. It is used primarily in connection with the A/D converter or in a timing mode in which it latches its own input.

INPUT/OUTPUT AND INTERRUPTS

2.7.3 Interrupt Status and Enables

All of the interrupt sources, except port 1C3, are taken to port 2B as inputs. After an interrupt has occurred, the program can read this port to determine the source of the interrupt. We refer to the content of this port as the interrupt status byte. The program can, of course, read this port at any time. The data are not dependent on whether an interrupt was enabled. Refer again to Figure 2-6 to see these connections.

Although the hardware is designed to permit interrupts from many different sources, most programs will be concerned with only one or a few of these. To prevent any reaction to an undesired interrupt, each interrupt source is gated with an output bit from port 2C. The processor will be interrupted only if an event occurs and its enable bit at port 2C is set high. These gates also are shown in Figure 2-6.

The interrupt sources, their positions in the interrupt status byte at port 2B, and their enable bits from port 2C are listed on the next page.

INPUT/OUTPUT AND INTERRUPTS

Source	Interrupt Enable Bit	Interrupt Status Bit
	(Active High)	
Timer 0 Flip-Flop	2C0	2B0
Timer 1 Flip-Flop	2C1	2B1
Timer 2 (no Flip-Flop)	2C2	2B2
A/D Comparator	2C3	2B3
EXT 4 Flip-Flop	2C4	2B4
EXT 5 Flip-Flop	2C5	2B5
Port 1C3	2C6	1C3
General Disable	2C7	
EXT 4 Direct (no interrupt)		2B6
EXT 5 Direct (no interrupt)		2B7

Note that Timer 0, Timer 1, EXT 4, and EXT 5 generate interrupts only through their flip-flops, which are set by rising edges. In processing the interrupt, the flip-flop will be reset (see section 2.7.4), so it will not generate new interrupts until another rising edge occurs.

INPUT/OUTPUT AND INTERRUPTS

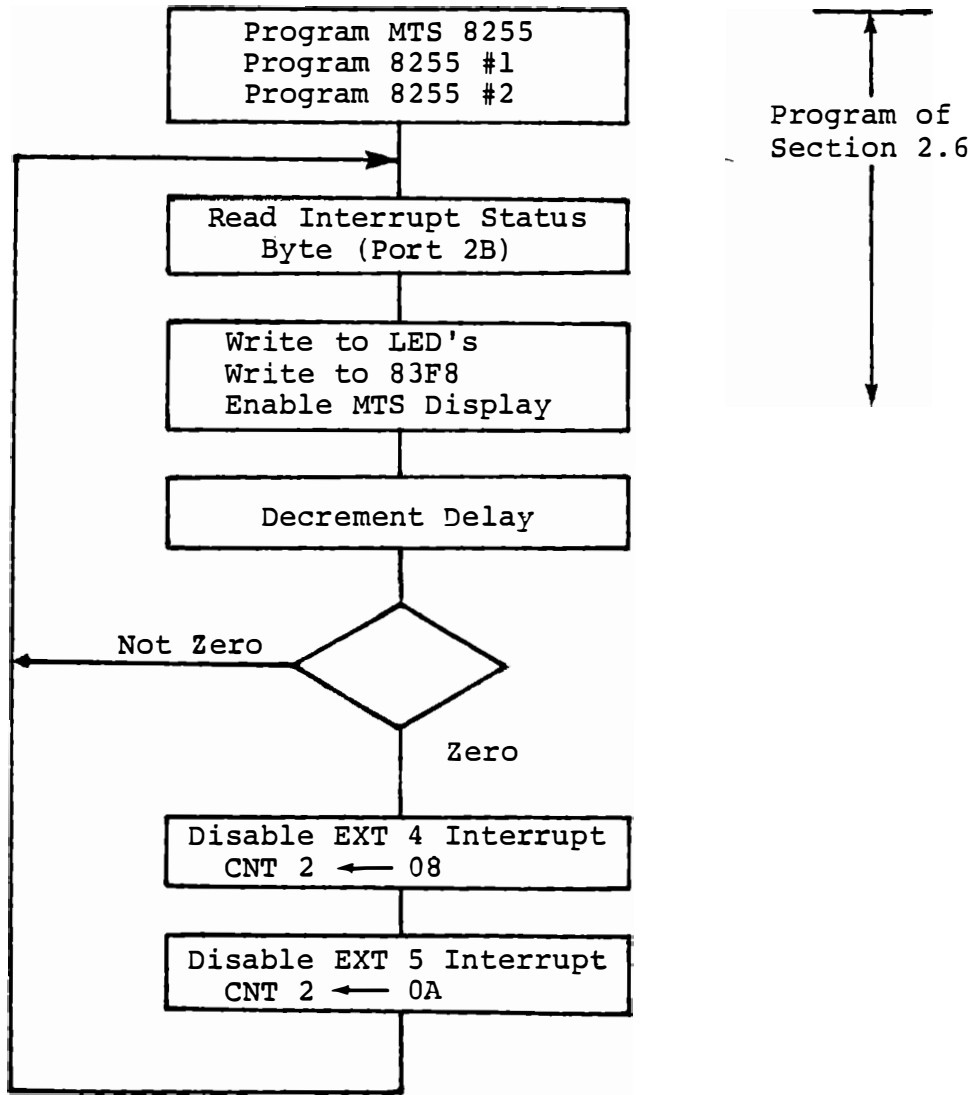
Port 2C7 is a general disable for all external interrupts. When it is high, only monitor interrupts can occur. At system reset all ports are forced to input mode thus floating the signal lines. To the logic this appears as a high signal so port 2C7 automatically inhibits external interrupts. Whenever port 2C is programmed for output, all bits are automatically set low. Now, bits 2C0 through 2C6 inhibit the individual interrupt sources. No external interrupt can occur until 8255 #2 has been programmed and specific bits of 2C have been set high.

2.7.4 Clearing Interrupts

The interrupt flip-flops for Timer 0, Timer 1, EXT 4, and EXT 5 are reset by the act of either setting or resetting the corresponding interrupt enable bit. This must be done by the bit set/reset command written to CNT2 (OF). Writing a byte to port 2C does not affect the interrupt flip-flop (see Figure 1-5). This logic design has three purposes:

- a) After an interrupt from one source has been processed, its flip-flop can be cleared without affecting any other source which may have received a signal while the previous interrupt was being serviced.
- b) A previously disabled source can be enabled and its flip-flop cleared so that only future events will generate interrupts.
- c) A previously disabled source can be enabled without clearing its flip-flop (by writing to Port 2C instead of CNT 2) so that a previous event can generate an interrupt.

INPUT/OUTPUT AND INTERRUPTS



Clearing Interrupts

Figure 2-8

EXERCISE:

We will demonstrate the setting and clearing of the interrupt flip-flops for EXT 4 and EXT 5, using the connection already set up (Figure 2-7), and extending the program of Section 2.6. The program will display EXT 4 and EXT 5 flip-flops as well as the direct inputs and demonstrate clearing the flip-flops. The routine will provide a delay period during which the inputs can be controlled manually and will be displayed. At the end of the delay, it will clear the flip-flop by a disable command. LED's DS6 and DS7 will display EXT 4 and EXT 5 direct inputs, while LED's DS4 and DS5 will display the state of EXT 4 and EXT 5 flip flops, respectively. Figure 2-8 shows the program.

During a delay period the interrupt status byte is repeatedly read and displayed. At the end of the delay, the EXT 4 and EXT 5 flip-flops are cleared by disabling their control bits in port 2C. Now if you ground the EXT 4 input during the delay period, its inverted output ($\overline{\text{EXT 4 OUT}}$) will become high and set the EXT 5 flip-flop. These signals will be displayed. If you remove the ground, the EXT 4 flip-flop will be set. If both flip-flops are set (both LED's on), it is due to "bouncing" the jumper contact more than once during a delay. In fact it is very difficult to make or break the connection so cleanly that you do not set both flip-flops, but it is possible. This phenomenon of seeing both rising and falling edges when a contact is opened or closed is called "contact bounce", and must be considered when connecting to switches.

CLEARING INTERRUPT FLIP-FLOPS

		A	D	D	R	CODE						
CODING SHEET	8	20	0	3E		MVI	A,	92		Program MTS 8255		
			1	92								
			2	D3		OUT		CNT0				
			3	03								
			4	3E		MVI	A,	80		Program 8255 #1		
			5	80								
			6	D3		OUT		CNT1				
			7	07								
			8	3E		MVI	A,	92		Program 8255 #2		
			9	92								
MICROCOMPUTER TRAINING SYSTEM		A	D3		OUT			CNT2				
			B	0F								
		820	C	DB		IN		PORT 2B		Read interrupt status byte		
			D	0D								
			E	D3		OUT		PORT 1A		Output to LED's		
			F	04								
	INTEGRATED COMPUTER SYSTEMS	8	21	0	32		STA		83F8		Output to MTS Display	
				1	F8							
				2	83							
				3	3E		MVI	A,	0F		Enable MTS Display	
			4	0F								
			5	D3		OUT		CNT0				
			6	03								
			7	2B		DCX		H		Delay with two byte countdown		
			8	7C		MOV	A,	H		in (HL)		
			9	B5		ORA		L		Repeat input and display		
		A	C2		JNZ		820C					
		B	0C									
		C	82									
		D	00		NO P							
		E	00		NO P							
		F	00		NO P							
	8		0									
			1									
			2									
			3									
			4									
			5									
			6									
			7									
			8									

Figure 2-9a

CLEARING INTERRUPTS (continued)

		A	D	D	R	CODE					
CODING SHEET	8	22	0	3E		MVI	A,	08			Disable and clear
			1	08							EXT 4 Interrupt
			2	D3		OUT	CNT	2			
			3	0F							
			4	3E		MVI	A,	0A			Disable and clear
			5	0A							EXT 5 Interrupt
			6	D3		OUT	CNT	2			
			7	0F							
			8	C3		JMP	820C				Back to input
			9	0C							
MICROCOMPUTER TRAINING SYSTEM	A	82									
	B										
	C										
	D										
	E										
	F										
	8	0									
		1									
		2									
		3									
		4									
		5									
		6									
		7									
		8									
	INTEGRATED COMPUTER SYSTEMS	8	0								
		1									
		2									
		3									
		4									
		5									
		6									
		7									
	8										

Figure 2-9b

INPUT/OUTPUT AND INTERRUPTS

INTERRUPT SERVICE

INSTRUCTION	ADDRESS STORED AT	COMMENT
RST 1	83F4,F5	Not used with ITS
RST 2	83F2,F3	Not used with ITS
RST 3	83F0,F1	Not used with ITS
RST 4	83EE,EF	Generally used for programmed call to monitor.
RST 5	83EC,ED	Generated by Timer 0 Default dispatch to 8228
RST 6	83EA,EB	Generated by other ITS interrupts Default dispatch to 8230
RST 7	83E8,E9	Monitor interrupt for STEP and breakpoint

Interrupt Dispatching

Figure 2-10

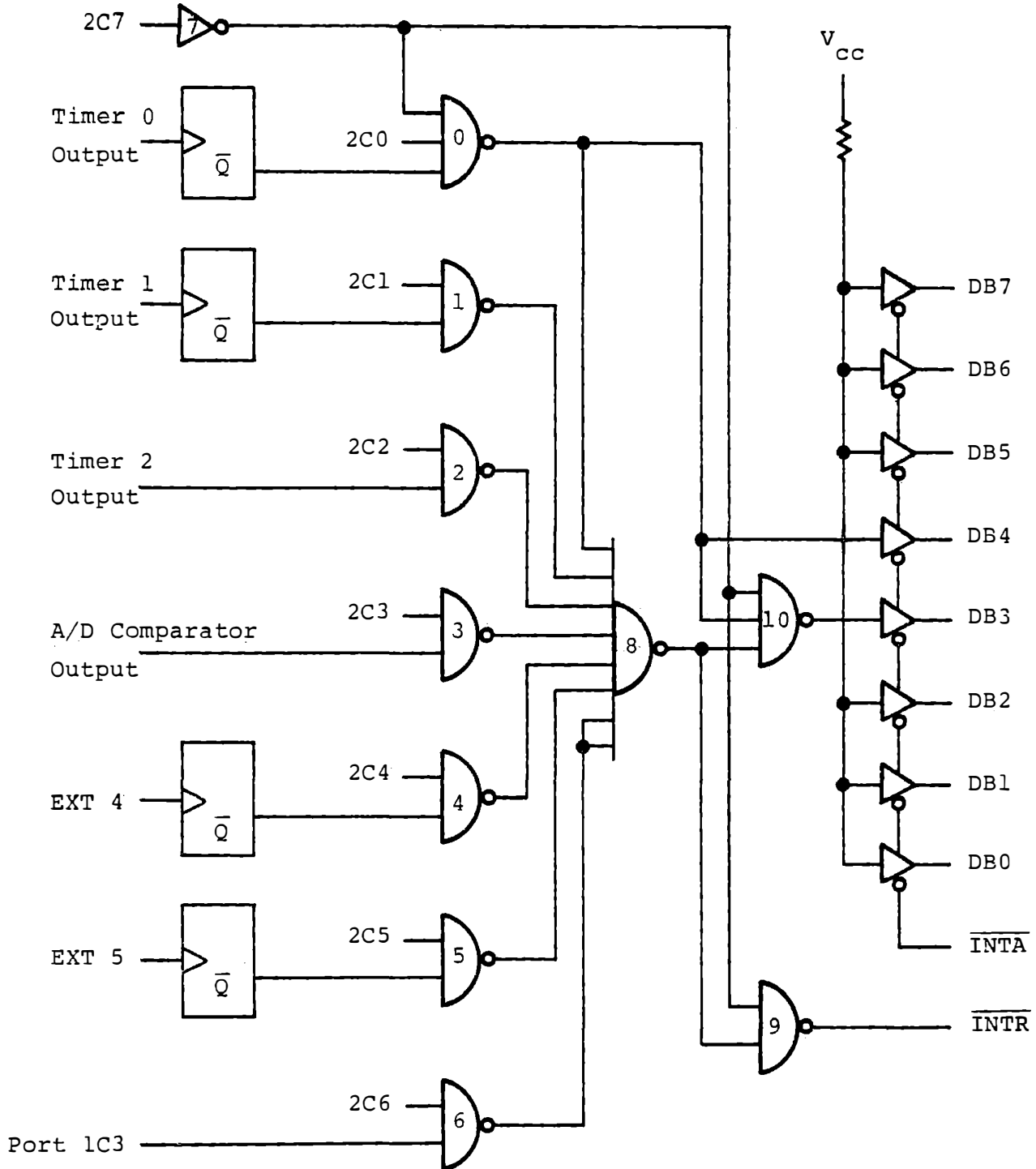
2.8 RESTART INSTRUCTIONS

The 8080 provides for eight "restart" instructions, RST0 - - RST7. (See Course 525, Chapter 8). The MTS hardware generates RST7 in response to an interrupt by resistive pullups on the data bus. The ITS interrupt system generates RST5 or RST6 in response to the various interrupt sources it provides.

2.8.1 RST Dispatch

The MTS monitor dispatches to an interrupt service routine in response to any of the RST instructions except RST0, which corresponds to RESET. The address of the interrupt service routine must be stored in RAM, according to the list in Figure 2-10. The monitor preloads addresses to dispatch RST5 and RST6 into the user's program area. Most of the program solutions in this course use these default addresses. Thus a Timer 0 interrupt service routine will be located at 8228, and service routines for all other ITS interrupts start at 8230.

INPUT/OUTPUT AND INTERRUPTS



(Numbers in gates are for reference to text, and do not indicate chip numbers.)

Generation of RST Instructions

2.8.2 RST Generation

The logic for generating interrupt request and restart instructions is shown in Figure 2-11. Port 2C7 is the general disable for interface board interrupts. When it is high (or floating), none of the interface board sources can generate an interrupt request. If the monitor hardware generates an interrupt request, all data bus bits will be high, giving an RST 7 interrupt.

When port 2C7 is low, the interface board interrupt sources can be enabled by the other bits of port 2C. If any interrupt source is high and its corresponding enable bit in port 2C is high, the NAND gate (0, 1, 2, - - , 6) output becomes low, forcing the output of gate 8 high. Now gate 9 generates the interrupt request, which is OR gated on the MTS board with the monitor interrupt request, so in STEP mode an interrupt occurs on every user instruction, but in AUTO mode an interrupt occurs only if 2C7 is low and one of the NAND gates 0-6 is low.

When INTA is output by the system controller in response to the interrupt request, the tri-state buffers are enabled to drive the data bus (DB0-DB7). Six of these are always high; DB3 and DB4 are controlled by the gates. The following possible combinations exist:

* 2C7 high. The interrupt request was generated by the MTS hardware. Gate 0 and gate 10 outputs are both high, giving 11111111 on the data bus. This is an RST 7 instruction.

INPUT/OUTPUT AND INTERRUPTS

This page intentionally left blank.

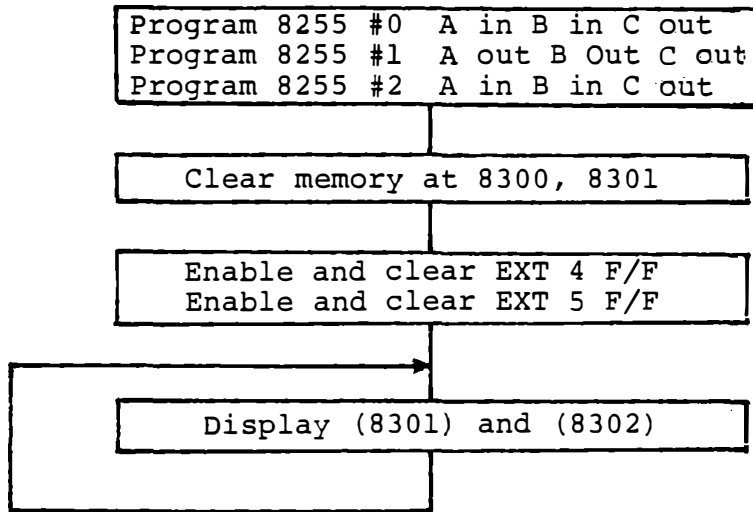
INPUT/OUTPUT AND INTERRUPTS

- * 2C7 low, timer 0 flip-flop and 2C0 high. Gate 0 output is low, forcing gate 10 output high. This gives 11101111 on the data bus, an RST 5 instruction.

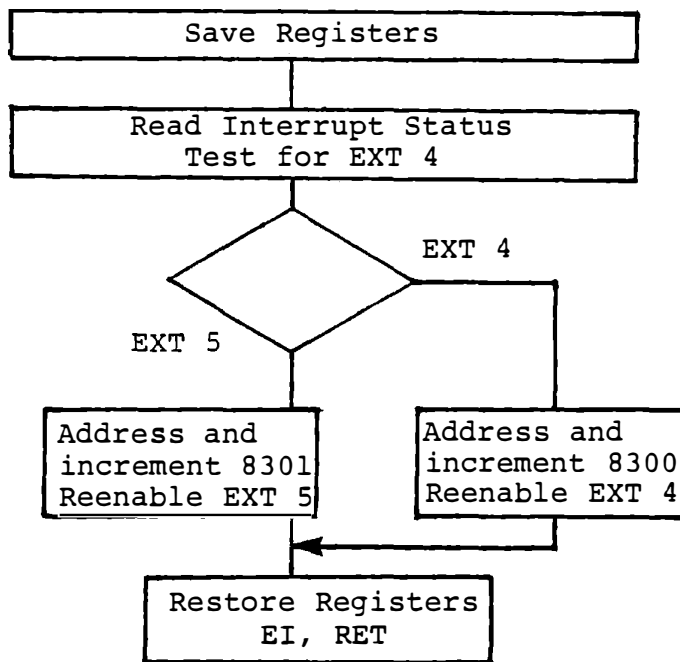
- * 2C7 low, timer 0 flip-flop or 2C0 low. Gate 0 output is high. Now if any other interrupt source and its enable bit are high, gate 10 is low, giving 11110111, RST 6, on the data bus.

- * 2C7 low but no enabled source high. Again the interrupt request has come from the monitor; gate 0 and gate 10 are high, and RST 7 is generated.

INPUT/OUTPUT AND INTERRUPTS



INTERRUPT SERVICE FOR RST 6



Interrupt Service - RST 5, RST 6

Figure 2-12

2.9 INTERRUPT SERVICE FOR EXT 4 AND EXT 5

EXERCISE:

Develop a program to count the number of times the EXT 4 input is connected to ground and released. Program the 8255's as in the preceding sections. Clear two bytes of variable memory at 8300 and 8301. Enable EXT 4 and EXT 5 interrupts (using the bit set command). Write a main program with a repetitive loop that loads and displays the two bytes from variable memory: high order byte for EXT 4, low order byte for EXT 5.

Write an interrupt service routine at 8230 to distinguish EXT 4 from EXT 5. Set the interrupt enable bit (to clear the flip-flop) and increment a count of number of interrupts. Use location 8300 for EXT 4 and 8301 for EXT 5. A flow diagram appears in Figure 2-12. Figure 2-13 lists the status bytes resulting from the various interrupt sources and the command bytes to disable or re-enable the interrupts. A solution to the programming problem is given in Figure 2-14a and 2-14b.

Note: The EXT4 and EXT5 inputs may react to noise, with the result that when one of them is triggered the other may also be triggered. Protect against this by connecting each of them through a 10K resistor to +5 volts.

INPUT/OUTPUT AND INTERRUPTS

In 2-14c an alternate interrupt service routine is shown to demonstrate two programming tricks. It is only necessary to save registers that will be used. Here only H, L, A and flags are use. When a conditional jump is to be made based on a yes-no decision, it is often more efficient to assume one result before making the jump. Here we can replace a three byte LXI 8301 (at 8246) with a single byte INX H, and we can omit the JMP 824E (at 8243). Such tricks are often powerful, but should be introduced only after a successful program has been written. Patching the program would be difficult in this situation.

STATUS AND COMMAND BYTES

INTERRUPT SOURCE	STATUS BYTE OBTAINED BY IN PORT 2B (see Note 2)									COMMAND BYTE WRITTEN BY OUT CNT 2 (see Note 1)		
	BINARY									HEX	DISABLE	ENABLE
Timer 0	0	X	X	X	X	X	X	1		01	00	01
Timer 1	0	X	X	X	X	X	1	X		02	02	03
Timer 2	0	X	X	X	X	1	X	X		04	04	05
A/D Comparator	0	X	X	X	1	X	X	X		08	06	07
EXT 4	0	X	X	1	X	X	X	X		10	08	09
EXT 5	0	X	1	X	X	X	X	X		20	0A	0B
Port 1C3	(see Note 3)										0C	0D

Note 1: Disable or enable command byte must be output to CNT 2 to clear the interrupt flip flop for Timer 0, Timer 1, EXT 4, or EXT 5. Disable or enable for A/D Comparator clears the interrupt in automatic A/D mode only.

Note 2: The hex values shown assume all other bits are 0. ANI (hex value) will give zero if the interrupt is not present.

Note 3: Port 1C3 does not appear in the status byte. It is read as XXXX1XXX by IN PORT1C. It is cleared by reading PORT1A in strobed input mode (mode 1 or mode 2) or by writing to PORT1A in strobed output mode (mode 1 or mode 2). Otherwise it can be cleared or set by writing 06 or 07 to CNT1. The interrupt enable for Port 1C3 is cleared or set by writing 0C or 0D to CNT2, but this does not change the data at Port 1C3.

Status and Command Bytes

Figure 2-13

		A	D	D	R	CODE	MAIN FOR EXT 4 AND EXT 5 SERVICE							
CODING SHEET	8	20	0	3	E		M	V	I	A	92	Program MTS 8255		
		1				92								
		2			D	3		O	U	T	C	N	T	0
		3			0	3								
		4			3	E		M	V	I	A	80	Program 8255 #1	
		5			8	0								
		6			D	3		O	U	T	C	N	T	1
		7			0	7								
		8			3	E		M	V	I	A	92	Programs 8255 #2	
		9			9	2							(sets all enable	
MICROCOMPUTER TRAINING SYSTEM		A			D	3		O	U	T	C	N	T	2
		B			0	F								
		C			2	1		L	X	I	H	0000	Clear two bytes	
		D			0	0							of memory at	
		E			0	0							8300 and 8301	
		F			2	2		S	H	L	D	8300		
		8	21	0	0	0								
		1			8	3								
		2			3	E		M	V	I	A	09	Enable EXT4	
		3			0	9							interrupt and	
INTEGRATED COMPUTER SYSTEMS		4			D	3		O	U	T	C	N	T	2
		5			0	F							clear its Flip Flop	
		6			3	E		M	V	I	A	0B	Enable EXT5	
		7			0	B							interrupt and	
		8			D	3		O	U	T	C	N	T	2
		9			0	F							clear its Flip Flop	
		821	A		2	A		L	H	L	D	8300		
		B			0	0								
		C			8	3								
		D			C	D		C	A	L	L	D	W	O
	E			D	1									
	F			0	2									
	8	22	0	C	3		J	M	P	821A				
	1			1	A									
	2			8	2									
	3													
	4													
	5													
	6													
	7													
	8													

Figure 2-14a

		A	D	D	R	CODE	INTERRUPT SERVICE FOR EXT 4 AND EXT 5				
CODING SHEET	8 23	0	F5			PUSH		PSW			Save registers
		1	E5			PUSH	H				
		2	D5			PUSH	D				
		3	C5			PUSH	B				
		4	DB			IN		PORT 2B			Read interrupt status byte
		5	0D								
		6	E6			ANI		10			Test for EXT 4
		7	10								
		8	CA			JZ		8246			Jump if not EXT 4
		9	46								
	A	82									
MICROCOMPUTER TRAINING SYSTEM	B	21				LXI	H,	8300			Address and increment counter for EXT 4
	C	00									
	D	83									
	E	34				INR	M				
	F	3E				MVI	A,	09			Reenable EXT 4 and clear flip flop
	8 24	0	09								
		1	D3			OUT		CNT 2			
		2	0F								
		3	C3			JMP		824E			Jump to exit
		4	4E								
	5	82									
	824	6	21			LXI	H,	8301			Address and increment counter for EXT 5
	7	01									
	8	83									
	9	34				INR	M				
	A	3E				MVI	A,	0B			Reenable EXT 5 and clear flip flop
	B	0B									
	C	D3				OUT		CNT 2			
	D	0F									
	824	E	C1			POP	B				
	F	D1				POP	D				
INTEGRATED COMPUTER SYSTEMS	8 25	0	E1			POP	H				
		1	F1			POP	PSW				
		2	FB			EI					
		3	C9			RET					
		4									
		5									
		6									
		7									
		8									

Figure 2-14b

SHORTER INTERRUPT SERVICE FOR EXT 4 AND EXT 5

		A	D	U	R	C	D	E									
		CODE															
CODING SHEET	8 23	0	F	5		P	U	S	H	P	S	W	<i>Use only registers that will be used</i>				
		1	E	5		P	U	S	H	H							
		2	D	B		I	N			P	O	R		T	2B	<i>Read interrupt status bytes</i>	
		3	O	D													
		4	E	6		A	N	I		I	O				<i>Test for EXT 4</i>		
		5	I	O													
		6	2	1		L	X	I		H	,	8		3	0	0	<i>Address counter for EXT 4</i>
		7	O	O													
		8	8	3													
		9	3	E		M	V	I		A	,	0		9		<i>Remainder bytes for EXT 4</i>	
MICROCOMPUTER TRAINING SYSTEM	A	O	9														
	B	C	2		J	N	Z		8	2	4	1		<i>Jump if interrupt was EXT 4</i>			
	C	4	1														
	D	8	2														
	E	2	3		I	N	X		H					<i>Address EXT 5 Count</i>			
	F	3	E		M	V	I		A	,	0	B		<i>Remainder bytes for EXT 5</i>			
	8 24	0	O	B													
	8 24	1	D	3		O	U	T		C	N	T	2				
		2	O	F													
		3	3	4		I	N	R		M							
	4	E	1		P	O	P		H								
	5	F	1		P	O	P		P	S	W						
	6	F	B		E	I											
	7	C	9		R	E	T										
INTEGRATED COMPUTER SYSTEMS	8																
		9															
		A															
		B															
		C															
		D															
		E															
		F															
		8	0														
			1														
		2															
		3															
		4															
		5															
		6															
		7															
		8															

Figure 2-14c

2.10 STANDARD PROGRAMMING FOR 8255'S

In Figure 2-12 we programmed the 8255's as follows:

```

8255  # 0   A in   B in   C out
8255  # 1   A out  B out  C out
8255  # 2   A in   B in   C out

```

Almost all of the exercises in this course will use either that programming, or the same except for port 1B:

```

8255  # 1   A out  B in   C out

```

In most program flow diagrams hereafter, we will show either:

```

Program 8255's - 1B out
Program 8255's - 1B in

```

This is to imply the programming above, with the assumption that the user program need not program 8255 #0 since the monitor sets it in the required condition. Figure 2-15 shows the program steps. You may want to post it in a convenient place.

INPUT/OUTPUT AND INTERRUPTS

Program 8255's - 1B out

3E	MVI	A,80
80		
D3	OUT	CNT1
07		
3E	MVI	A,92
92		
D3	OUT	CNT2
0F		

Program 8255's - 1B in

3E	MVI	A,82
82		
D3	OUT	CNT1
07		
3E	MVI	A,92
92		
D3	OUT	CNT2
0F		

Standard Programming for 8255's

Figure 2-15

MICROCOMPUTER INTERFACING WORKBOOK

CHAPTER 3

INTERVAL TIMERS

This page intentionally left blank.

3. INTERVAL TIMERS

Timing functions are extremely common in computers used in real time applications and communications. Timing can be achieved by program loops but with two major limitations. The precision of the timing is limited to the length of the loop, (commonly of the order of four or more instructions) and the computer can do nothing else while it is timing. Hardware timers overcome these limitations at moderate cost.

3.1 INTEL 8253 INTERVAL TIMER

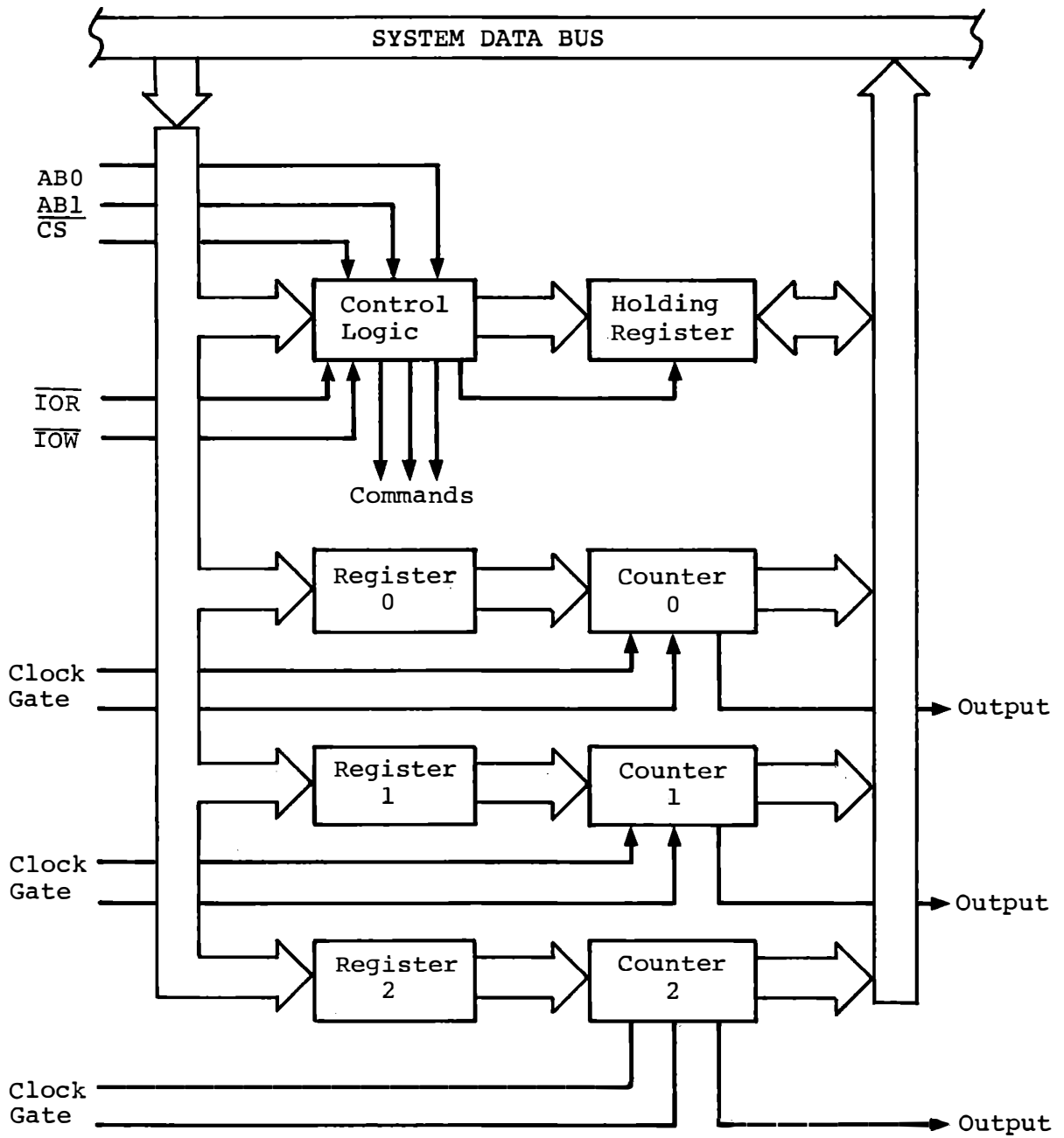
The 8253 provides three identical, independent 16 bit timers. Each timer comprises (Figure 3-1) a 16 bit counter and a 16 bit storage register (accessible by IN and OUT instructions), control logic and flip flops, a clock input, gate input, and an output. Each of the timers occupies one I/O port address for reading the counter and loading the register. A fourth I/O port provides for controlling all of the counters. Various operation modes exist which may be selected by writing a control byte to the control port.

Initiating a timer's operation always involves two steps. First the timer "mode" must be specified to the control register. Second, the timer count-down value is initialized. Typically this requires the following sequence.

INTERVAL TIMERS

MVI A, control byte	Write control byte to
OUT TIMCT	timer control port, to set mode.
MVI A, low data byte	
OUT TIMER	Load low time data byte
MVI A , high data byte	
OUT TIMER	Load high time data byte

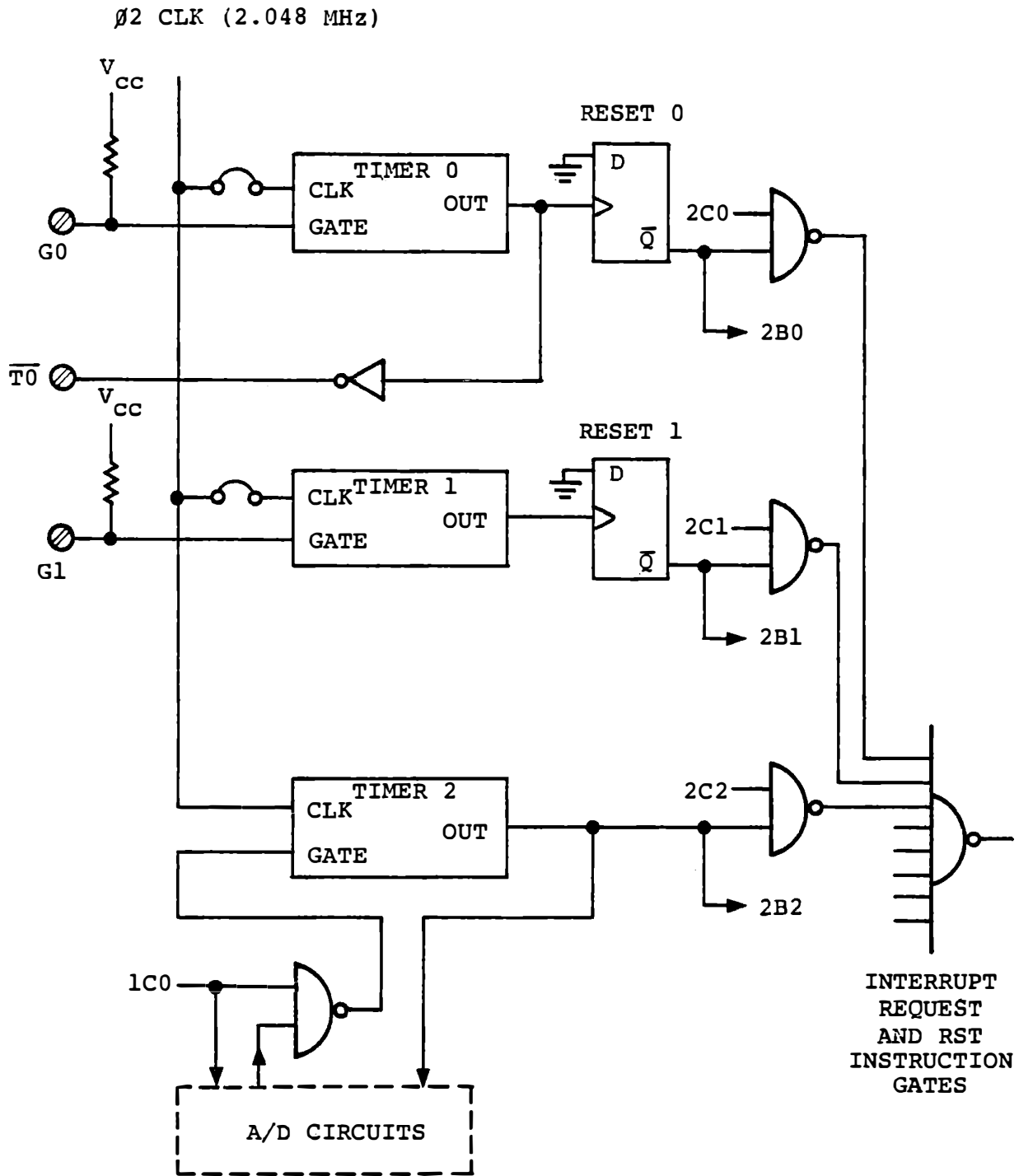
After the count value has been initialized the timer will run under control of its clock and gate inputs, and will generate a particular output signal depending on which timer mode was pre-specified. The output waveform could be used as a timed interrupt to the microprocessor, a low speed clock for an external circuit, or a variety of other applications in a microcomputer system, as we will see throughout the course.



Intel 8253 Interval Timer

Figure 3-1

INTERVAL TIMERS



Timer Clocks, Gates and Outputs

Figure 3-2

3.2 CLOCK, GATE, AND OUTPUT

Each timer receives a clock input and decrements the content of its counter at the falling edge of the clock. On the interface board all clock inputs are normally connected to the system 2.048 MHz clock, but this connection can be altered to permit use of an external clock input. (See Figure 3-2.)

The gate input to each timer starts, enables or disables its counting, depending on the selected mode. On the interface board these inputs for timer 0 and timer 1 are pulled high by resistors so that counting is normally enabled, but these gate inputs are also accessible at terminals for external control. The gate input to timer 2 is connected to the analog to digital converter circuitry because timer 2 is often used in A/D operations (as discussed in Chapter 5). To enable timer 2 for other functions its gate input must be forced high by setting port 1C0 low.

The output of a timer goes high to indicate the end of a time interval. The time at which it goes low depends on the mode selected. On the interface board the outputs of timer 0 and timer 1 set flip-flops in the interrupt system, exactly like the EXT 4 and EXT 5 flip-flops. Timer 2 output has no flip-flop. It is directly gated with an interrupt enable bit into the interrupt system, and it is also used to drive a counter in the A/D converter.

INTERVAL TIMERS

The output of timer 0, as well as setting an interrupt flip-flop, also drives an inverter whose output is available at a terminal for use by external hardware.

Because the system clock is nominally 2.048 MHz it is very easy to relate binary counts to decimal times. The following table lists some useful values.

INTERVAL TIMERS

Binary Count (Hexadecimal)	Decimal Count	Time (milliseconds)
0100	256	0.125
0200	512	0.250
0400	1024	0.500
0800	2048	1.000
1000	4096	2
1800	6144	3
2000	8192	4
2800	(10240)*	5
3000	(12288)	6
3800	(14336)	7
4000	(16384)	8
4800	(18432)	9
5000	(20480)	10
A000	(40960)	20
F000	(61440)	30
0000	(65536)	32
		Time (seconds)
1F40	8000	1/256
0FA0	4000	1/512
07D0	2000	1/1024

*The timer cannot be loaded with decimal values greater than 9999, so binary counting must be used.

INTERVAL TIMERS

This Page Intentionally Left Blank

3.3 TIMER MODES

Any of the three interval timers can operate in any of six modes (0-5). Most of the experiments here use mode 0 or mode 2. The other modes are intended principally for interfacing the timer directly with external hardware rather than through the program. The modes are listed below, and defined in subsequent sections along with experiments. A summary of the modes is given in section 3-10.

Mode 0 Interrupt on Terminal Count

Mode 1 Programmable One Shot

Mode 2 Rate Generator

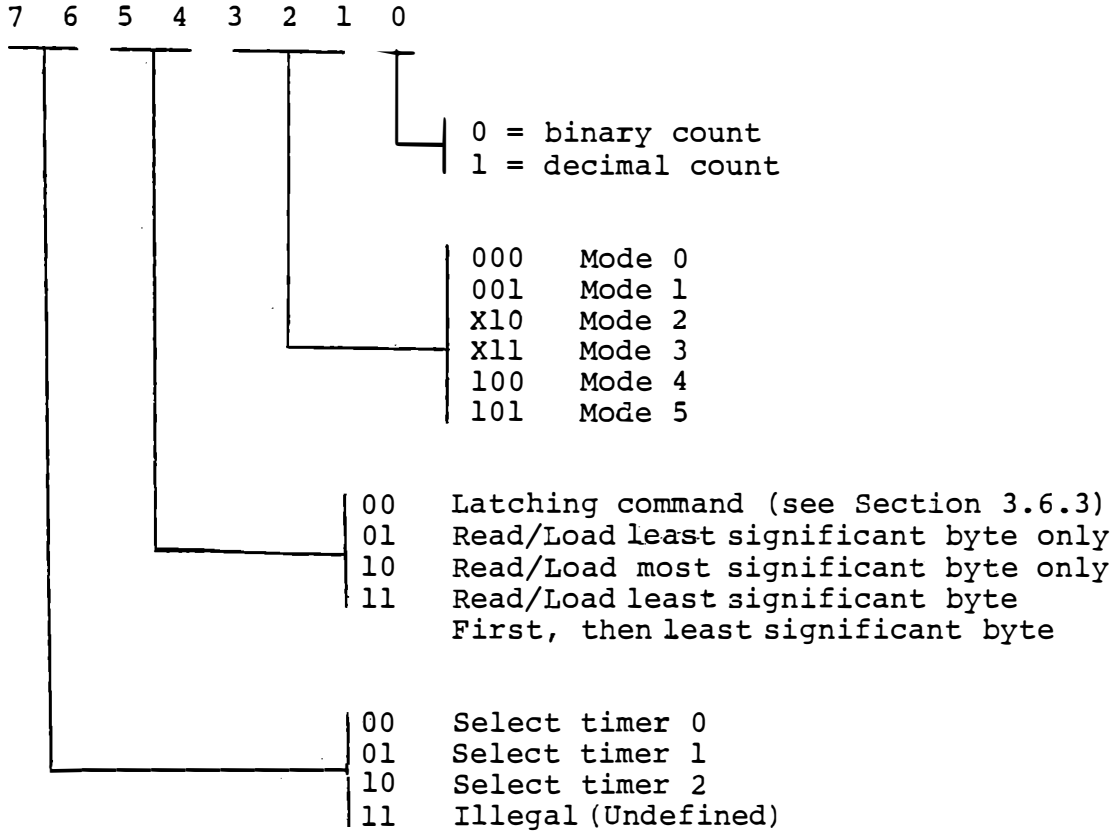
Mode 3 Square Wave Generator

Mode 4 Software Triggered Strobe

Mode 5 Hardware Triggered Strobe

Within each of these modes the user has some additional options. The counters are 16 bits long, and can be loaded with two bytes of data, less significant byte first. Two other options (which must be selected when the mode is programmed) are to load only the less significant byte or to load only the more significant byte. In either of these cases, the other byte is set to 00, and counting proceeds on both bytes.

INTERVAL TIMERS



Timer Control Byte Structure

Figure 3-3

INTERVAL TIMERS

The timers can count in binary or decimal, as selected when the mode is programmed.

The mode and options are selected for any one of three timers by writing a byte to the control port of the 8253.

```
3E      MVI A,CONTROL BYTE
XX
D3      OUT TIMCT
17
```

Figure 3-3 shows the bit structure of the control byte. Figure 3-4 lists the most commonly used control bytes for each of the three timers.

INTERVAL TIMERS

	0	1	2	3	4	5
Latch	00	00	00	00	00	00
Read/Load LSB	10	12	14	16	18	1A
Read/Load MSB	20	22	24	26	28	2A
Read/Load Both (LSB first)	30	32	34	36	38	3A

Timer 1	Mode					
	0	1	2	3	4	5
Latch	40	40	40	40	40	40
Read/Load LSB	50	52	54	56	58	5A
Read/Load MSB	60	62	64	66	68	6A
Read/Load Both (LSB first)	70	72	74	76	78	7A

Timer 2	Mode					
	0	1	2	3	4	5
Latch	80	80	80	80	80	80
Read/Load LSB	90	92	94	96	98	9A
Read/Load MSB	A0	A2	A4	A6	A8	AA
Read/Load Both (LSB first)	B0	B2	B4	B6	B8	BA

Control Bytes shown set binary counting

Add 1 for decimal counting

Write control byte to TIMCT, Port 17

Latching control byte does not affect mode

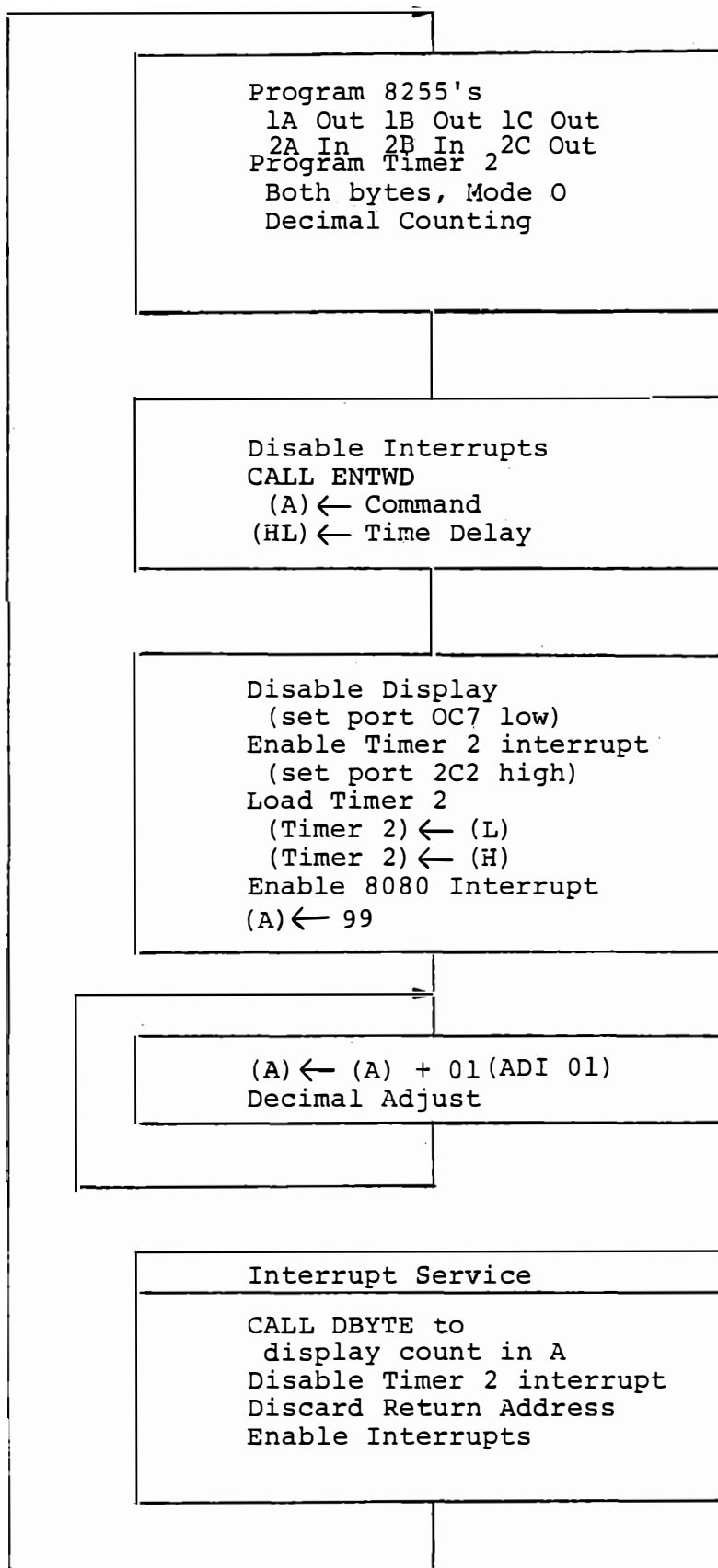
Timer Control Bytes

Figure 3-4

3.4 MODE 0 - INTERRUPT ON TERMINAL COUNT

When a timer is set to mode 0, its output goes low. When it has been loaded (with one or two bytes as required by the mode select option), and its gate input is high, it will decrement the count at each falling edge of the clock. When the count reaches zero, the output goes high. Mode 0 is intended to generate a time delay whose duration and starting time are set by the program. In the following exercise we compare a programmed timing loop with an interval timer. Figure 3-5 shows the program flow diagram.

INTERVAL TIMERS



Compare Timing Loop with Interval Timer

Figure 3-5

EXERCISE

This program accepts a time delay value from the keyboard, and starts timer 2 in mode 0 with this value. After enabling interrupts it enters a counting loop. At the interrupt generated by timer 2 it displays the value reached by the counting loop. The interrupt service discards the return address (by POP H) and jumps to start since the function of the main program is finished when the interrupt occurs.

Note that timer 2, which has no external flip-flop in the interrupt system, is appropriately used here because in mode 0 its output goes low when it is programmed to mode 0 or when it is loaded, goes high and stays high at the end of the interval.

The addresses, programming control bytes, and interrupt enable/disable bytes are found in Figures 2-2, 2-13, and 3-4. Duplicate copies of these are found in Appendix A. You may want to post them for ready reference.

```
The delay loop should be:  ADI  01    7 clocks
                           DAA          4  "
                           JMP          10  "
                               21 clocks
```

The interval timer will count 21 times as fast as the programmed timing loop. When you run the program, find the smallest delay value you can enter that results in a zero in (A). This represents the time taken to reach the DAA instruction after the second byte is

INTERVAL TIMERS

loaded to the timer. (Run the program in AUTO mode to make the time measurements.)

You should be able to add 21 to that value and get a count of 01. (We programmed the timer and the loop for decimal counting to make the arithmetic easier.) Each added value of 21 in the delay should result in one added count in the result. At some intermediate values you will see hex values in the display because the interrupt occurred after ADI 01 but before DAA. At 2088 you should obtain a count of 99. With delays from 2109 to 2115 the count will be 9A, and at 2116 it will be 00.

If the display is not disabled during counting the programmed timing loop will be slower because of the hold states introduced by the DMA channel for the display.

Try running the program in STEP mode (but with the RUN key). Now you can measure the time taken by the monitor. Insert some breakpoints that will never be reached and observe the effect.

		A	D	D	R	CODE	COMPARE	TIMING	LOOP	WITH	INTERVAL	TIMER		
CODING SHEET	8	20	0	3E		MVI	A,	80					Program 8255 #1	
			1	80									About Bout Count	
			2	D3		OUT	CNT	1						
			3	07										
			4	3E		MVI	A,	92					Program 8255 #2	
			5	92									A in Bin Count	
			6	D3		OUT	CNT	2						
			7	0F										
			8	3E		MVI	A,	B1					Program Timer 2	
			9	B1									Both Buttons	
MICROCOMPUTER TRAINING SYSTEM		A	D3		OUT	TIM	CT						Mode 0	
		B	17										Decimal	
		820	C	F3		DI							For step mode	
			D	CD		CALL	ENT	WD						
			E	46										
			F	03										
		8	21	0	3E		MVI	A,	0E					Set Port OC7 low
				1	0E									to disable display
				2	D3		OUT	CNT	0					
				3	03									
INTEGRATED COMPUTER SYSTEMS			4	3E		MVI	A,	05					Set Port 2C2 high	
				5	05								to enable Timer 2	
				6	D3		OUT	CNT	2				interrupt (but	
				7	0F								8080 interrupts	
				8	7D		MOV	A,	L				still disabled)	
				9	D3		OUT	TIM	2					
			A	16										
			B	7C		MOV	A,	H					Load time delay	
			C	D3		OUT	TIM	2					to Timer 2	
			D	16										
		E	FB		EI									
		F	3E		MVI	A,	99							
	8	22	0	99										
			1	C6		ADI	01							
			2	01										
			3	27		DAA								
			4	C3		JMP	8221							
			5	21										
			6	82										
			7											
			8											

Figure 3-6a

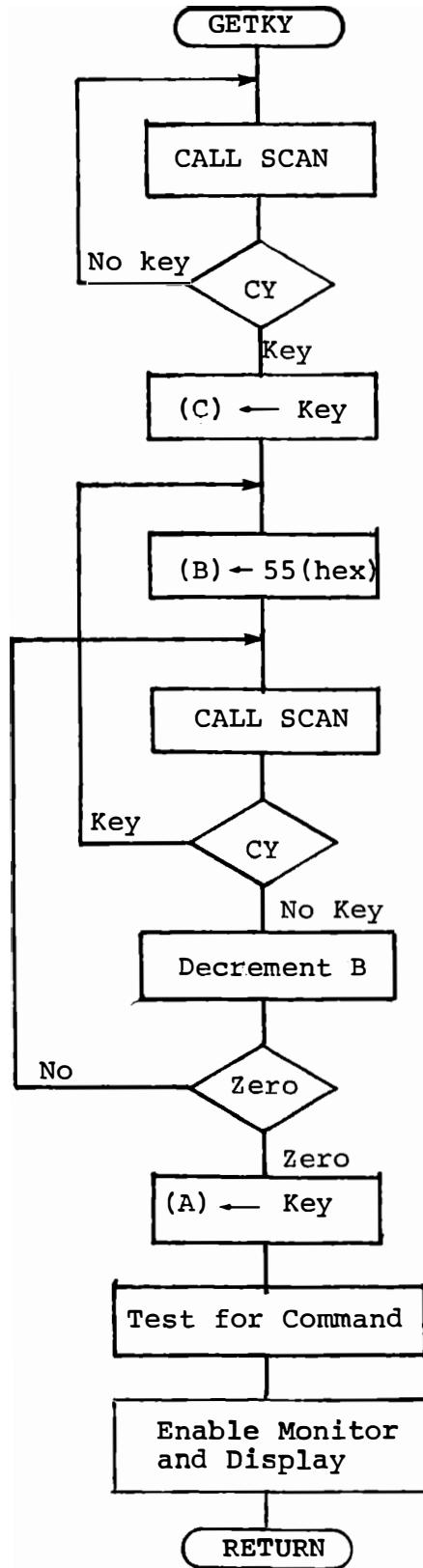
INTERRUPT SERVICE FOR TIMING COMPARISON

		A	D	D	R	CODE												
CODING SHEET	8		0															
			1															
			2															
			3															
			4															
			5															
			6															
			7															
MICROCOMPUTER TRAINING SYSTEM	8	22	8	C	3	J	M	P	8	2	3	0						
			9		3	0												
		A		8	0													
		B																
		C																
		D																
		E																
		F																
		8	23	0	C	D	C	A	L	D	B	Y	T	E	Display	Count		
				1	9	5												
				2	0	2												
				3	3	E	M	V	I	A	,	0	4	Disable	Timer 2			
				4	0	4								interrupt				
				5	D	3	O	U	T	C	N	T	2					
				6	0	F												
				7	E	I	P	O	P	H				Discard	return			
			8	F	B	E	I						address					
			9	C	3	J	M	P	8	2	0	0	Jump	to	start			
INTEGRATED COMPUTER SYSTEMS	A		0	0														
	B		8	2														
	C																	
	D																	
	E																	
	F																	
	8		0															
			1															

Figure 3-6b

This page intentionally left blank.

INTERVAL TIMERS



GETKY Flow Diagram

Figure 3-7

3.5 RESTARTING A COUNTER IN MODE 0.

When a counter is running in mode 0 it can be stopped and restarted by loading a new time count. The output will remain low while this is done, and go high only when the most recently loaded count reaches zero.

EXERCISE

The monitor subroutine GETKY is used to get a single keyboard entry. After a key has been pressed and read, it repeatedly reads the keyboard until the key has been released for 20 milliseconds to protect against contact bounce, which might otherwise cause a single key operation to be read as two or more operations. Figure 3-7 is a flow diagram for GETKY. SCAN is the subroutine that actually reads the keyboard. If a key is pressed SCAN returns the hex value in (A) and carry set. If no key is pressed SCAN returns carry cleared. When this has occurred for 20 milliseconds (85 repetitions of SCAN) we are sure that the key has been released. If a key contact is sensed during that time the delay is started again.

INTERVAL TIMERS

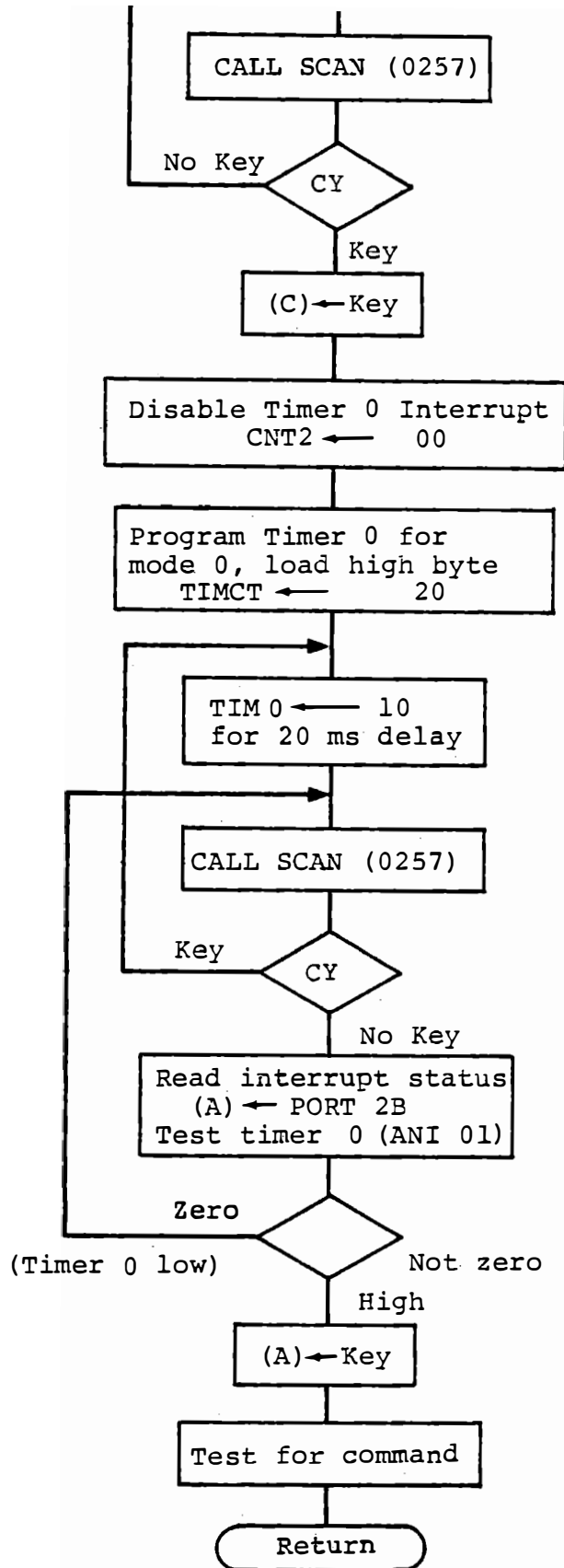
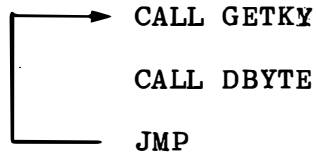


Figure 3-8

We will develop a substitute for GETKY that uses timer 0 instead of a delay loop. Figure 3-8 is a flow diagram for this program. The diagram is generally the same as figure 3-7 except for the delay functions. Write this subroutine and call it instead of GETKY in the program below.



Note that we restart timer 0 each time SCAN finds the key still present, but let it run when the key is released. Timer 0 output never goes high until the timer is decremented to zero.

We have disabled the timer 0 interrupt because this program tests timer 0 itself and does not want an interrupt to occur. Other interrupts are allowed. It is necessary to disable the timer interrupt (using the bit reset function) to clear the flip-flop, because it is the flip flop output that is read in port 2B, not the direct output from the timer. Timer 1 can be used in the same way. Make that substitution and see that the program still works. It can also work with timer 2, but port 1C must be programmed for output and bit 1C0 set low, otherwise timer 2 may be inhibited from counting by the A/D circuitry.

GETKY USING INTERVAL TIMER

	A	D	D	R	CODE													
CODING SHEET	8	24	0		CD		CALL		SCAN								Read Keyboard	
			1		57													
			2		02													
			3		D2		JNC		8240								Repeat until key	
			4		40													is pressed
			5		82													
			6		4F		MOV		C, A									(C) ← key
			7		3E		MVI		A, 00									Disable Timer 0
			8		00													interrupt
			9		D3		OUT		CNT2									
MICROCOMPUTER TRAINING SYSTEM		A			0F													
			B			3E		MVI		A, 20								Program Timer 0
			C			20												load high byte
			D			D3		OUT		TIMCT								mode 0, binary
			E			17												
			824	F				MVT		A, 10								TIM0 ← 10
			825	0														for 20 ms delay
				1				OUT		TIM0								
				2														
				825	3				CALL		SCAN							
INTEGRATED COMPUTER SYSTEMS				4														
				5														
				6														
				7														
				8														
				9														
				A														
				B														
				C														
				D														
			E															
			F															
			8															
			0															
			1															
			2															
			3															
			4															
			5															
			6															
			7															
			8															

Figure 3-9

3.6 READING A TIMER

A timer can be read as well as loaded. The exercises of this section make use of that facility.

3.6.1 Measuring a Pulse Duration

EXERCISE

In mode 0 (also modes 2, 3 and 4) counting continues only while the gate input is high. We can use this to measure the width of a pulse. A useful signal source is the MTS cassette modem output. Use a clip lead to connect the test point labelled AUDIO OUT at the upper right edge of the MTS to the gate input (G1 IN) for timer 1. Use a short jumper to connect this also to the EXT 4 input. The modem output is nominally 1200 Hz if port 0C0 output is low, and twice that when port 0C0 is high. We will write a program to select the frequency by keyboard input, measure the width of the high portion of the output signal, and display that width. The width is displayed in decimal clock pulse units. Divide the clock count by the clock frequency (2.048 MHz) to determine the input pulse width. Alternately, since the signal we are measuring is a square wave, obtain the frequency by $1024000/\text{count}$.

INTERVAL TIMERS

To measure the pulse width we will initially load timer 1 with zero, while the input signal is low. After the signal has gone high and then returned to low we will read the counter.

```

XRA  A           Enter zero to
OUT  TIM1        both bytes of
OUT  TIM1        the timer

           '      Wait for input
           '      to go high and
           '      then low

IN    TIM1       Read the timer
MOV  L,A        content into
IN    TIM1       registers H,L
MOV  H, A
```

Although we could clear the timer to zero with a single byte load, if it were so programmed, we would then be restricted to a single byte read.

The process above reads and stores the content of the timer, but since it counts down this result is the twos or tens complement of the actual time, as shown in Figure 3-10.

INTERVAL TIMERS

Binary Counting		Decimal Counting	
Positive Count	Timer Data	Positive Count	Timer Data
0000	0000	0000	0000
0001	FFFF	0001	9999
0002	FFFE	0002	9998
0003	FFFD	0003	9997
0004	FFFC	0004	9996
0005	FFFB	0005	9995
0006	FFFA	0006	9994
0007	FFF9	0007	9993
0008	FFF8	0008	9992
0009	FFF7	0009	9991
000A	FFF6	0010	9990
000B	FFF5	0011	9989
000C	FFF4	0012	9988
000D	FFF3		
000E	FFF2		
000F	FFF1		
0010	FFF0		
0011	FFEF		
00FF	FF01	0099	9901
0100	FF00	0100	9900
0101	FEFF	0101	9899
0FFF	F001	0999	9001
1000	F000	1000	9000
1001	EFFF	1001	8999
FFFF	0001	9999	0001
0000	0000	0000	0000

Twos and Tens Complement Counting

Figure 3-10

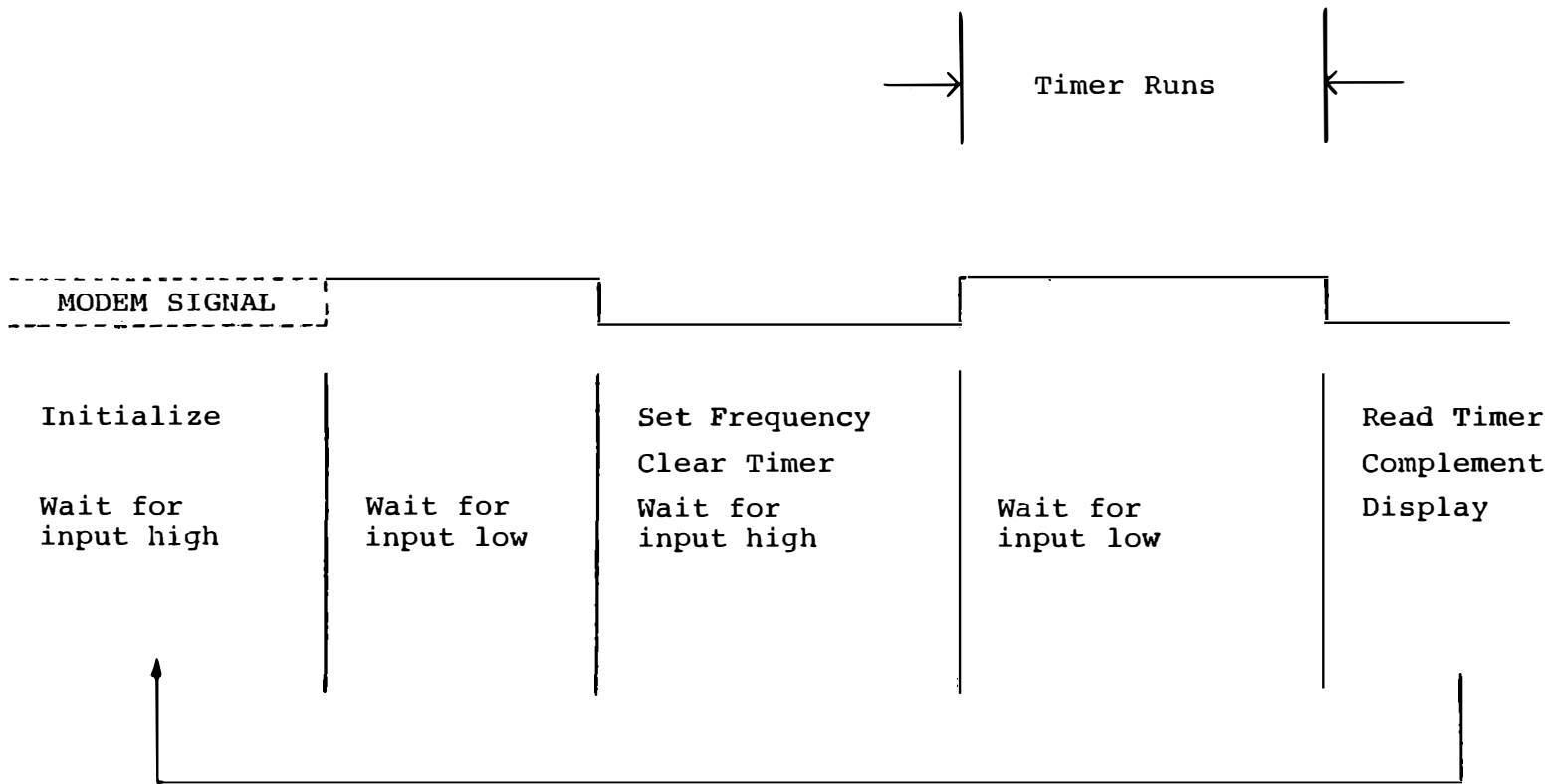
INTERVAL TIMERS

The twos complement can most easily be converted by complementing the byte as it is read and then adding one to the two byte result.

```
IN    TIM1
CMA
MOV   L,A
IN    TIM1
CMA
MOV   H,A
INX   H
```

The tens complement is needed if we use decimal counting. In Course 525 (Chapter 10) we developed a subroutine to convert a two byte decimal value to its hundreds complement. This subroutine is shown in Figure 3-13c.

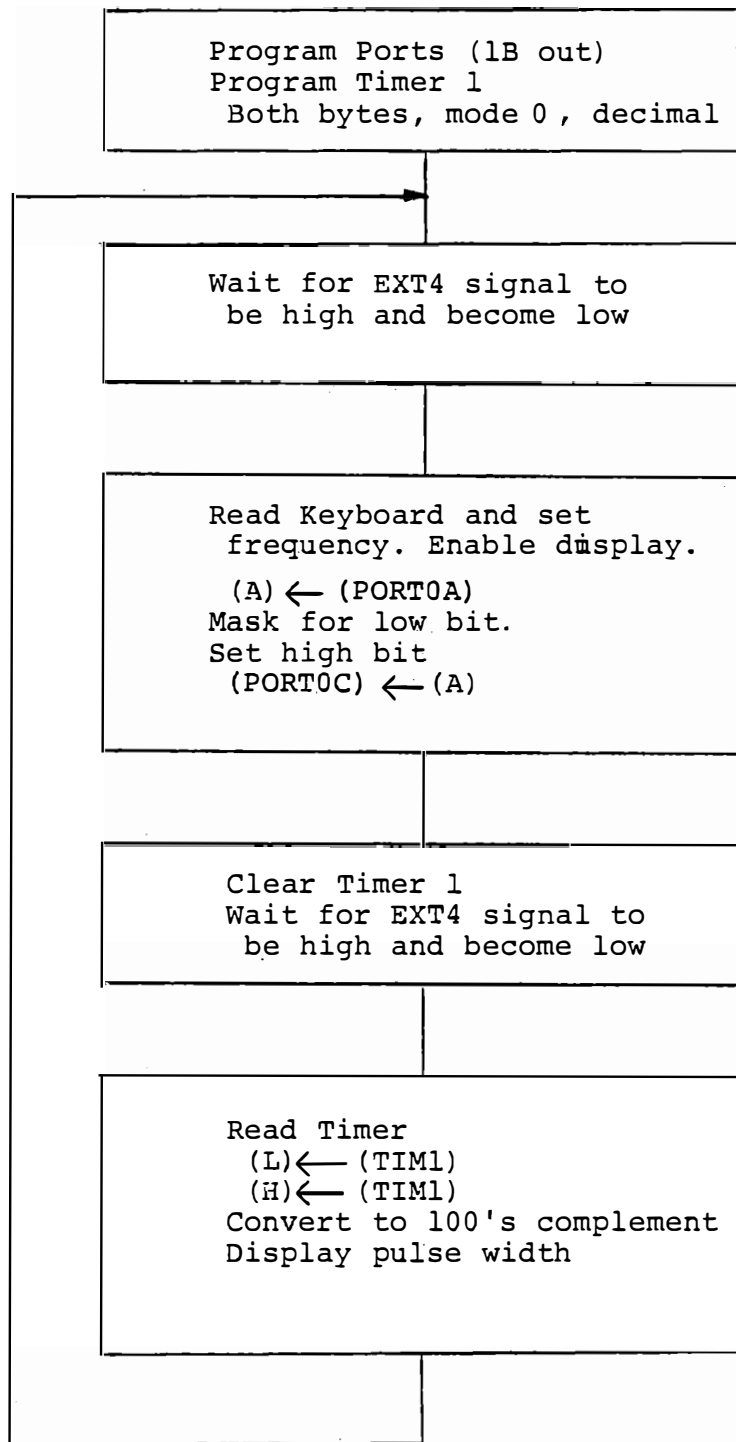
Although the counter in mode 0 will only run while its gate input is high, it gives no direct indication to the program when it stops. Figure 3-11 shows how the computer will react to the input signal. Figure 3-12 is a flow diagram for the program. A program solution is given in Figure 3-13 for decimal counting.



Time Diagram for Pulse Width Measurement

Figure 3-11

INTERVAL TIMERS



Pulse Width Measurement

Figure 3-12

PULSE WIDTH MEASUREMENT

	A	D	D	R	CODE														
CODING SHEET	8	20	0		3E		MVI	A,	80									Program 8255 #1	
			1		80													A out Bout Count	
			2		D3		OUT		CNT1										
			3		07														
			4		3E		MVI	A,	92									Program 8255 #2	
			5		92														
			6		D3		OUT		CNT2										
			7		0F														
			8		3E		MVI	A,	71										Program Timer 1
			9		71														Both lysters
MICROCOMPUTER TRAINING SYSTEM		A			D3		OUT		TIMCT									Mode 0	
			B		17													Decimal	
		820	C		CD		CALL		WTHL									Wait until input	
			D		30														has been high
			E		82														and becomes low
			F		DB		IN		PORTOA										Read keyboard
		821	0		00														
			1		E6		ANI		01										Mask low low bit
			2		01														and set high bit
			3		F6		ORI		80										to set frequency
		4		80														and enable display	
		5		D3		OUT		PORTOC											
		6		02															
		7		AF		XRA		A										Clear Timer	
		8		D3		OUT		TIM1											
		9		15															
		A		D3		OUT		TIM1											
		B		15															
		C		CD		CALL		WTHL										Wait through high	
		D		30														part of pulse.	
		E		82														Timer will read.	
		F		00		NOP													
INTEGRATED COMPUTER SYSTEMS	3	0																	
			1																
			2																
			3																
			4																
			5																
			6																
			7																
		8																Figure 3-13a	

PULSE WIDTH (continued)

		A	D	D	R	CODE													
CODING SHEET	8	22	0	D	B			T	N			T	I	M	1			Read Times	
			1			1	5											It is stopped	
			2		6	F			M	O	V		L	,	A			became gate	
			3		D	B			I	N			T	I	M	1			input is low
			4				1	5											
			5		6	7			M	O	V		H	,	A				
			6		C	D			C	A	L	L		H	U	N	C	P	Take hundreds
			7				4	0											complement for
			8				8	2											decimal time
			9		C	D			C	A	L	L		D	W	O	R	D	Display Times
MICROCOMPUTER TRAINING SYSTEM	A			D	1														
	B			0	2														
	C			C	3			J	M	P		8	2	0	C			Loop	
	D			0	C														
	E			8	2														
	F			0	0			N	O	P									
	8	23	0	D	B			I	N			P	O	R	T	2	B	WTHL	
			1		0	D													Load interrupt
			2		E	6			A	N	I		4	0					status byte and
			3				4	0											test EXT 4 input
		4		C	A			J	Z			8	2	3	0			at bit 6. Wait	
		5				3	0											until EXT 4 high.	
		6				8	2												
INTEGRATED COMPUTER SYSTEMS	8	23	7	D	B			I	N			P	O	R	T	2	B	Load interrupt	
			8		0	D													status byte
			9		E	6			A	N	I		4	0					and wait until
			A				4	0											EXT 4 is low.
			B		C	2			J	N	Z		8	2	3	7			
			C				3	7											
			D				8	2											
			E		C	9			P	E	T								
			F				0	0											
		3	0																
		1																	
		2																	
		3																	
		4																	
		5																	
		6																	
		7																	
		8																	

Figure 3-13b

		A	D	D	R	CODE	HUNDREDS COMPLEMENT - TWO BYTES						
CODING SHEET	8 24	0	3	E		M	V	I	A	,	9	A	Represents 100
		1	9	A									
		2	9	5		S	U	B	L				Subtract LSD
		3	C	6		A	D	I	0	0			Add 0 to make
		4	0	0									DAA valid
		5	2	7		D	A	A					
		6	6	F		M	O	V	L	,	A		
		7	3	E		M	V	I	A	,	9	9	
		8	9	9									Subtract higher
		9	C	E		A	C	I	0	0			digits from 99
MICROCOMPUTER TRAINING SYSTEM	A	0	0										but add in CY
	B	9	4		S	U	B	H					from LSD
	C	C	6		A	D	I	0	0				Add 0 to make
	D	0	0										DAA valid
	E	2	7		D	A	A						
	F	6	7		M	O	V	H	,	A			
	8 25	0	C	9		R	E	T					
		1											
		2											
		3											
	4												
	5												
	6												
	7												
	8												
	9												
INTEGRATED COMPUTER SYSTEMS	A												REGISTERS B, C, D, E
	B												PRESERVED
	C												
	D												
	E												
	F												
	8	0											
		1											
		2											
		3											
	4												
	5												
	6												
	7												
	8												

Figure 3-13c

INTERVAL TIMERS

3.6.2 Additional Exercises

Two other ways of recognizing the state of the input signal are possible. One method uses EXT 4 and EXT 5 interrupts but is merely a simple extension of the preceding program. The other reads the timer to determine whether it is running. You should develop the program of 3.6.2.2 yourself. Exercise 3.6.2.1 is optional.

3.6.2.1 Awaiting an Interrupt

EXERCISE

The program in Figure 3-13 can be modified to use the EXT 5 interrupt in place of the WTHL wait subroutine at 8230H.

Connect an additional jumper from EXT4 OUT to EXT5 IN, and enable the EXT5 interrupt which will occur at the falling edge of the signal. We will load Timer 1 the first time this interrupt occurs, and then wait for a second interrupt. When that occurs we will read the timer, which will have counted down during the time that the signal pulse was high.

The processor can be forced to stop operations by the HLT (76H) instruction. When this is encountered in a program the processor enters a wait state, and does not execute any further instructions until an interrupt occurs. At this time the interrupt service routine is executed and then control returns to the next instruction following the HLT.

Replace the WTHL subroutine with an interrupt service routine that does the following:

```
Save PSW
Reenable and clear the interrupt
Restore the PSW
EI
Return
```

To enable the interrupt initially, call this service routine instead of WTHL. Follow the call by HLT to wait for the first falling edge. Since RST6 is exactly equivalent to CALL 8230, you can insert both of these instructions and a NOP in place of CALL WTHL.

Now replace the second CALL WTHL by HLT, NOP, NOP. This will cause the processor to wait for the second falling edge. The remainder of the main program is unchanged.

INTERVAL TIMERS

3.6.2.2 Reading an Active Timer

EXERCISE

You can re-code the original program (Figure 3-13) to read the timer while it is running.

Use the original WTHL subroutine to detect the first falling edge (The timer interrupt should not be enabled). In place of the second call to WTHL, call a new subroutine that does the following:

- Read Timer 1 (both bytes)
- Save the result
- Read Timer 1 again (both bytes)
- Compare with previous result
- Repeat until the result changes, indicating that the timer is running
- Repeat until the result no longer changes, indicating that the timer has stopped

NOTE: use of an internal subroutine may shorten this program to less than 30D bytes.

3.6.3 Reading While Counting

In the exercise of 3.6.1, the timer is read only when counting has been inhibited by a low input at the gate. In 3.6.2.2, the counter is read while it is counting. The final measurement which is displayed was taken after counting stopped.

The IOR signal that places input data on the data bus extends across at least one phase 2 clock cycle. Since the timer runs from the phase

2 clock, it is guaranteed that the lowest bit will change during the time that the counter outputs are driving the bus, and possible that all 16 bits will change. The data thus received by the 8080 while the data bits are changing must be considered garbage. The 8253 provides a facility for accurately reading a timer while it is counting. There is one 16 bit register which can be synchronously loaded with the content of any one counter, upon command from the processor. A subsequent IN (or two IN's for two bytes) addressed to the same counter will access the latching register rather than the counter itself. The latching control bytes were included (though not defined) in Figures 3-3 and 3-4.

Timer Control Byte Digits

Control Byte				
Binary	Hex	Timer	Operation	
0000 XXXX	00	0	Copy timer into latching	
0100 XXXX	40	1	register before reading	
1000 XXXX	80	2		

INTERVAL TIMERS

Like the mode set control bytes, these are sent by OUT TIMCT. To read a running timer that is programmed for two byte read and load the following sequence is used.

3E	MVI	A, 40	Latch control byte for timer 1
40			
D3	OUT	TIMCT	Write to timer control
17			
DB	IN	TIM1	Read latched data from timer 1
15			
6F	MOV	L,A	
DB	IN	TIM1	Store in (HL)
15			
67	MOV	H,A	

Note that the IN instructions are still addressed to timer 1, and two reads are still required if the timer is programmed for two byte load and read.

EXERCISE

Develop a program to demonstrate that invalid data may be read from a running counter if the latching operation is not used.

3.7 MODE 2 - RATE GENERATOR

Probably the most common use of the interval timer is generation of a signal or an interrupt at precisely repeated intervals. This is useful for:

- * Generating a slower clock for an external device that cannot use the 2.048 MHz system clock.
- * Measuring times or generating timing functions too great for the 32 millisecond capacity of a 16 bit counter.
- * Servicing inputs or outputs on a schedule rather than by interrupts.
- * Keeping track of real time.

3.7.1 Use of Mode 2

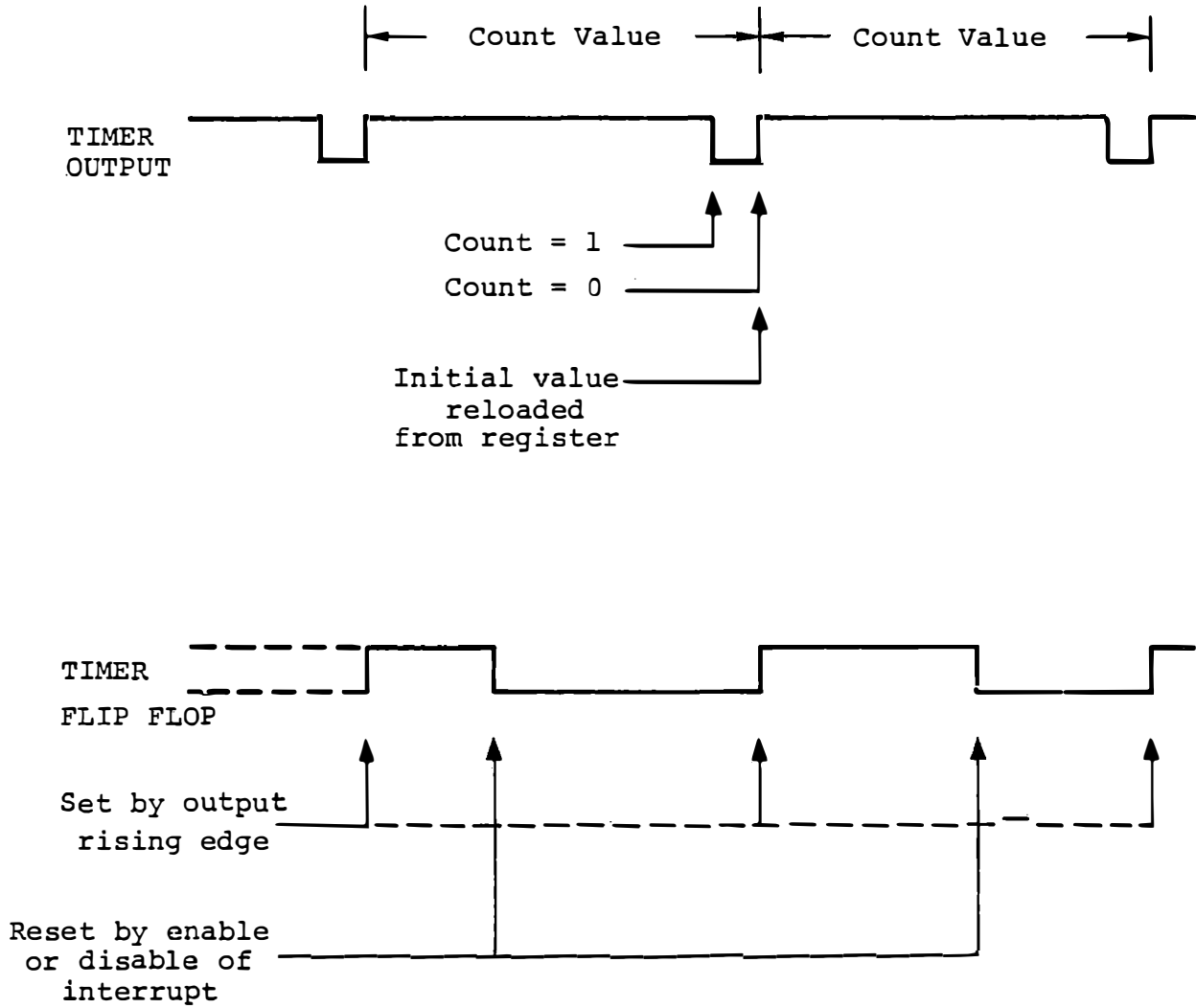
Mode 2 is programmed by writing a control byte to the timer control port in accordance with Figure 3-3. For instance, to program timer 1 for a two byte load, mode 2, binary:

```

3E MVI A, 74
74
D3 OUT TIMCT
17

```

INTERVAL TIMERS



Timer and Flip Flop Operation

Mode 2 - Rate Generator

Figure 3-14

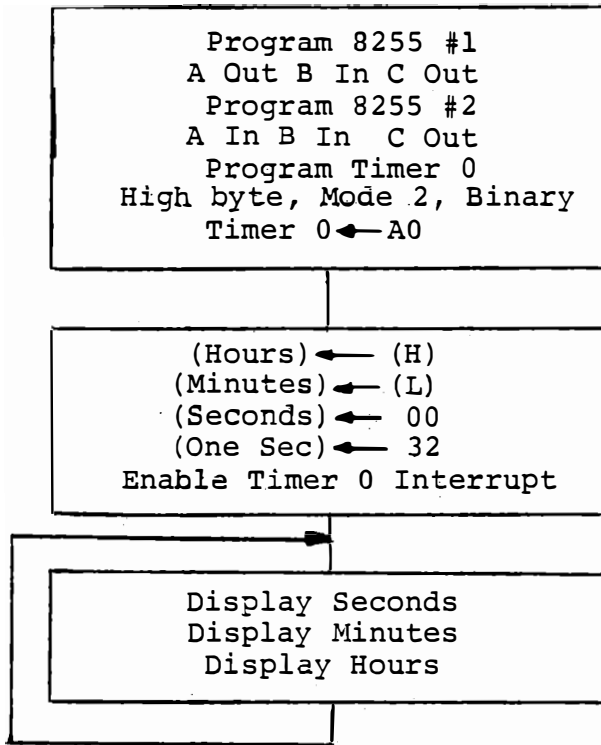
This immediately sets the output high if it was not high. Counting starts when a count value is loaded, provided the gate input is high. Counting is inhibited if the gate input is low.

The output remains high while counting, until the count value reaches 0001, when the output goes low. At the next falling edge of the clock, the output goes high and the initial value is reloaded from the count register into the counter. Thus, a half microsecond pulse is output once for each counting cycle. The timer need not be reloaded, and it will give the pulse at a precisely repeated interval even if the interrupt service is delayed.

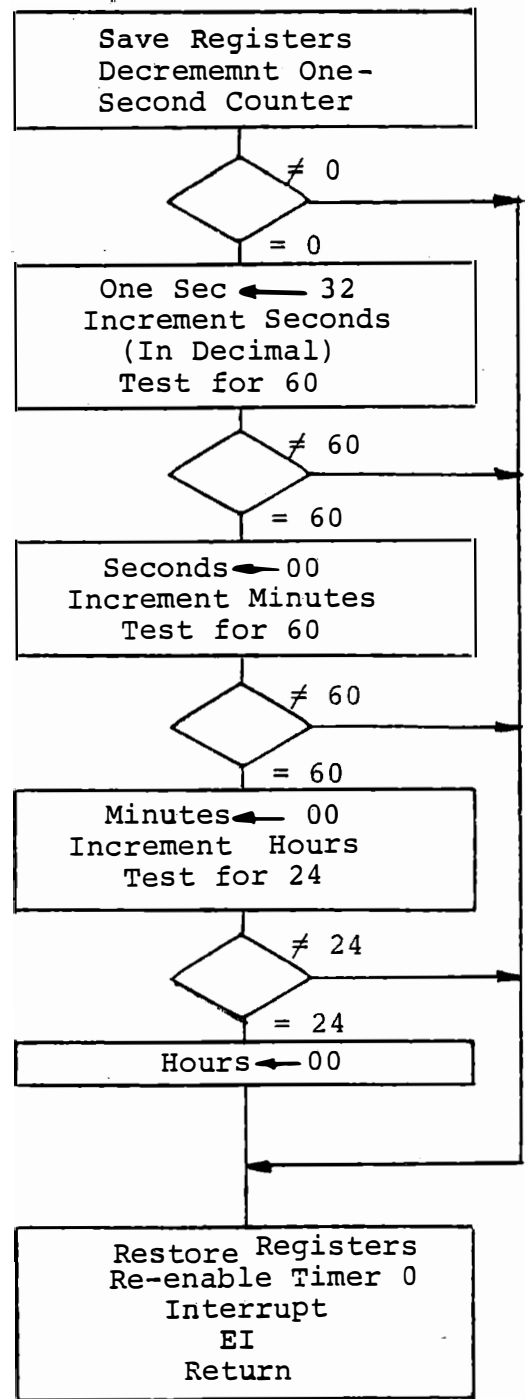
Figure 3-14 shows the relationship between the timer output and its flip flop. Note that the half microsecond pulse from the timer, if it were directly connected to interrupt request, might or might not generate an interrupt, depending on the state of the 8080 at that moment. Therefore, the interrupt must be taken from the flip flop of timer 0 or timer 1. The ITS does not provide a flip flop for Timer 2, so this timer cannot reliably generate an interrupt in mode 2 (nor in modes 4, 5 or 6, for the same reason).

After the flip flop has generated an interrupt it must be reset by setting or resetting the corresponding enable bit at port 2C0 or 2C1, before an EI instruction is given. Otherwise repeated interrupts will be generated, and the main program can never be executed.

INTERVAL TIMERS



MAIN TIME DISPLAY PROGRAM



RST 5 INTERRUPT SERVICE

Time of Day Clock

Figure 3-15

3.7.2 Real Time Clock

EXERCISE

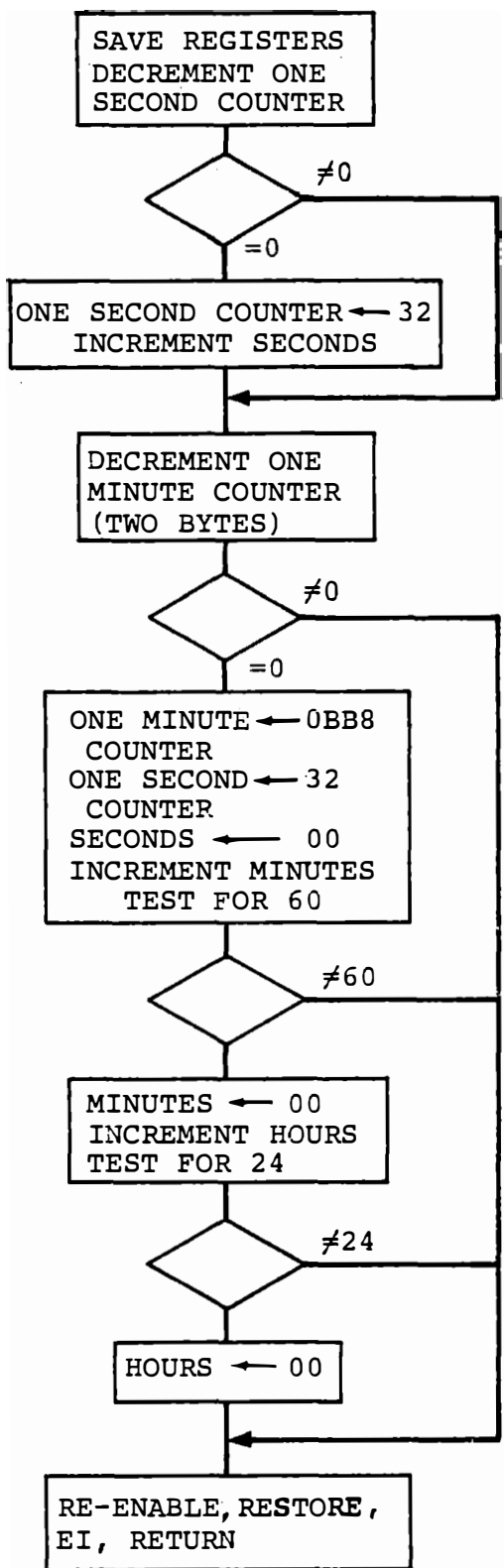
Develop a program that will keep and display the time of day. The flow diagram of Figure 3-15 displays hours, minutes and seconds. Timer 0 generates an interrupt every 20 milliseconds and a software counter is decremented from 32 (=50 decimal). At zero a seconds counter is incremented (in decimal). At 60 seconds a minutes counter is incremented and at 60 minutes an hours counter is incremented. The display function is handled by the main program. This would permit another program to operate in conjunction with the time of day. It can use the keyboard and display, and when nothing else is going on the time can be displayed.

In this program the starting time (in hours and minutes) is loaded from the content of H and L. Use the monitor to place the time in those registers and press RUN when the second hand of your watch reaches zero. Test the timekeeping. You will probably find this clock to be quite inaccurate because the crystal of the MTS is only accurate to 0.1%. This gives an error of 86 seconds a day.

INTERVAL TIMERS

The clock can be made somewhat better by using a separate software counter for one minute, with an initial value of about 0BB8 (= 3000). This can be adjusted for crystal frequency error; allowing the clock error to be less than 30 seconds a day. Figure 3-16 shows a flow diagram for this clock, with coding provided in Figure 3-17. A further improvement can be made with a one hour counter, with a nominal initial value of 02BF20 (180000 decimal). This permits an adjustment to less than half a second per day if the crystal is sufficiently stable.

You may want to elaborate the clock program for the fine adjustment, or to load time of day by keyboard entry, or to keep date as well as time with adjustments for 28, 29, 30, or 31 days. (Remember that leap year is omitted every 100 years but included every 400 years, so 29 February 2000 will exist).



RST 5 Interrupt Service

Figure 3-16

		A	D	D	R	CODE	TIME OF DAY - INITIALIZE				
CODING SHEET	8	20	0	3E		MVI	A,	82			Program 8255 #1
			1	82							About Bin Count
			2	D3		OUT	CNT	1			
			3	07							
			4	3E		MVI	A,	92			Program 8255 #2
			5	92							Bin Bin Count
			6	D3		OUT	CNT	2			
			7	0F							
			8	3E		MVI	A,	24			Program Timer 0
			9	24							Load high byte
MICROCOMPUTER TRAINING SYSTEM			A	D3		OUT	TIM	CT			Mode 2, Binary
				B	17						
				C	3E		MVI	A,	A0		Timer 0 ← A000
				D	A0						for 20 milliseconds
				E	D3		OUT	TIM	0		interrupt
				F	14						
		8	21	0	EB		XCHG				(DE) ← initial time
				1	21		LXI	H,	8300		Address software
				2	00						counters
				3	83						
			4	36		MVI	M,	32		(8300) ← 32 for	
			5	32						one second counter	
			6	23		INX	H			(8301) ← 00	
			7	36		MVI	M,	00		for 60 second	
			8	00						counter	
INTEGRATED COMPUTER SYSTEMS				9	23		INX	H			(8302) ← (E)
				A	73		MOV	M,	E		for minutes
				B	23		INX	H			(8303) ← (D)
				C	72		MOV	M,	D		for hours
				D	EF		RST	5			Programmed call
				E	C3		JMP	8260			to interrupt
				F	60						service to enable
		8	22	0	82						the interrupt
				1							
				2							
			3								
			4								
			5								
			6								
			7								
			8								

Figure 3-17a

		A	D	D	R	CODE	TIME OF DAY - PROGRAM LOOP				
CODING SHEET	8	26	0	2	1	LXI	H,	8301			Address 60
			1	0	1						second counter
			2	8	3						
			3	C	D	CALL	D	MEM			Display seconds
			4	9	4						
			5	0	2						
			6	2	3	INX	H				Address minutes
			7	7	E	MOV	A,	M			
			8	C	D	CALL	D	BY 2			Display
			9	9	8						
MICROCOMPUTER TRAINING SYSTEM	A		0	2							
			B	2	3	INX	H				Address hours
			C	7	E	MOV	A,	M			
			D	C	D	CALL	D	BY 2			Display
			E	9	8						
			F	0	2						
		8	27	0	C	3	JMP	8260			Loop
			7	1	6	0					
			7	2	8	2					
	INTEGRATED COMPUTER SYSTEMS			3							
			4								
			5								
			6								
			7								
			8								
			A								
			B								
			C								
			D								
		E									
		F									
	8		0								
			1								
			2								
			3								
			4								
			5								
			6								
			7								
			8								

Figure 3-17b

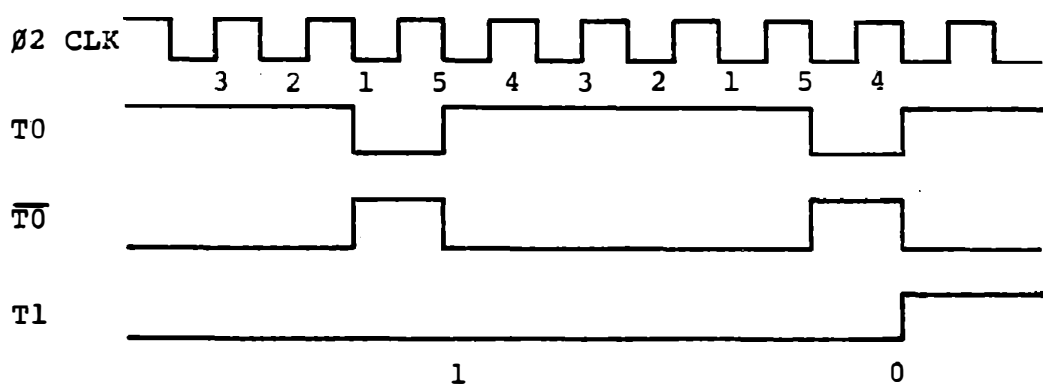
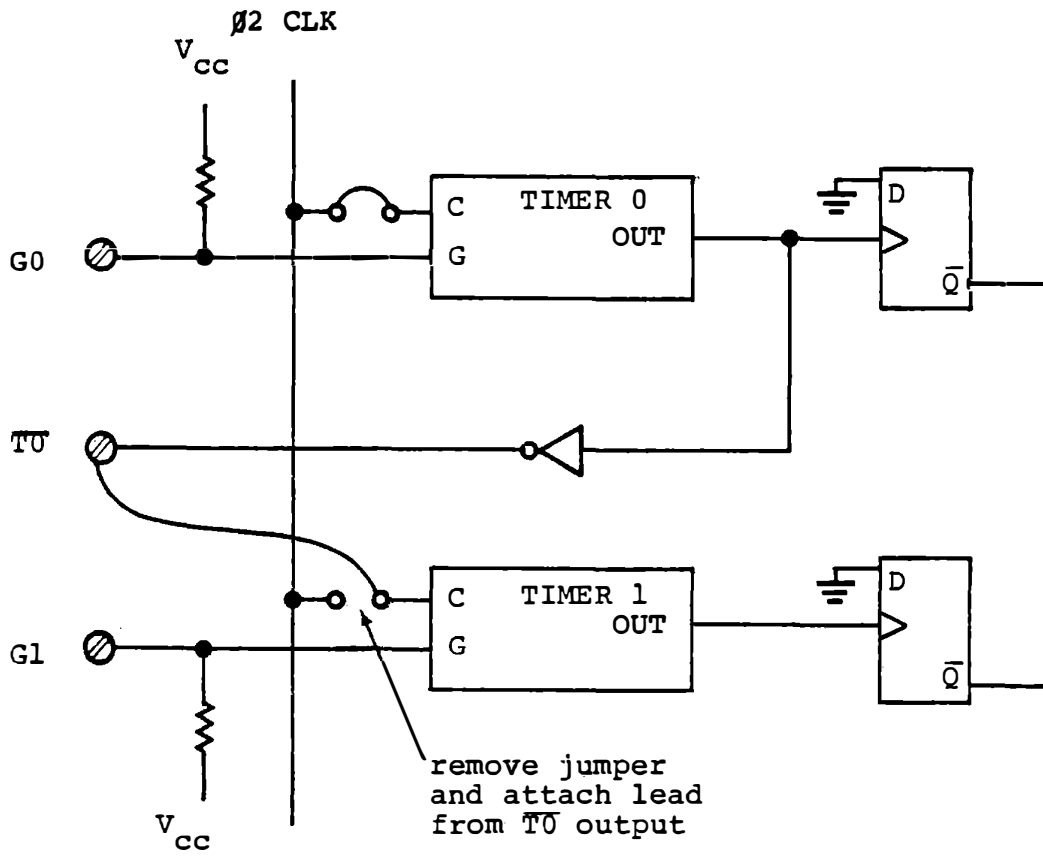
CODING SHEET		MICROCOMPUTER TRAINING SYSTEM										INTEGRATED COMPUTER SYSTEMS									
8	0																				
	1																				
	2																				
	3																				
	4																				
	5																				
	6																				
	7																				
822	8	F5		PUSH	PSW						save registers										
	9	E5		PUSH	H																
	A	21		LXI	H, 8300						Address one)										
	B	00									second counter										
	C	83																			
	D	C3		JMP	8233						Jump past RST6										
	E	33									interrupt location										
	F	82																			
823	0	E7		RST	4						No RST6 interrupt										
	1	FB		EI							should occur.										
	2	C9		RET							Leave space for JMP										
823	3	35		DCR	M						Decrement one										
	4	C2		JNZ	8248						second counter										
	5	48																			
	6	82																			
	7	36		MVI	M, 32						Restart one second										
	8	32									counter										
	9	3E		MVI	A, 59						Max value for										
	A	59									60 second counter										
	B	CD		CALL	8250						Subroutine increments										
	C	50									or clears counter										
	D	82																			
	E	3E		MVI	A, 59						Max value for										
	F	59									60 minute counter										
8	0																				
	1																				
	2																				
	3																				
	4																				
	5																				
	6																				
	7																				
	8																				

Figure 3-17c

		A	D	D	R	CODE	TIME OF DAY	-	RST	INT	SVC	AND	SUBR	
CODING SHEET	8 24	0	C	C		C	Z		8	2	5	0		Call if 60 second counter was 59
		1	5	0										to increment 60
		2	8	2										minute counter
		3	3	E		M	V	I	A	,	2	3		flag value for
		4	2	3										24 hour counter
		5	C	C		C	Z		8	2	5	0		Call if 60 minute
		6	5	0										counter was 59
		7	8	2										Renable timer 0
MICROCOMPUTER TRAINING SYSTEM	8 24	8	3	E		M	V	I	A	,	0	1		interrupt to
		9	0	1										close flip flop
		A	D	3		O	U	T	C	N	T	2		
		B	0	F										
		C	E	1		P	O	P	H					
		D	F	1		P	O	P	P	S	W			
		E	F	B		E	I							
		F	C	9		R	E	T						
INTEGRATED COMPUTER SYSTEMS	8 25	0	2	3		I	N	X	H					Subroutine
		1	9	6		S	U	B	M					Address next
		2	C	A		J	Z		8	2	5	9		counter and test
		3	5	9										for maximum value
		4	8	2										
		5	7	E		M	O	V	A	,	M			If not maximum
		6	C	6		A	D	I	0	1				increment in
		7	0	1										decimal
	8	2	7		D	A	A							
	8 25	9	7	7		M	O	V	M	,	A			
	A	C	9		R	E	T							
	B													
	C													
	D													
	E													
	F													
	8	0												
	1													
	2													
	3													
	4													
	5													
	6													
	7													
	8													

Figure 3-17d

INTERVAL TIMERS



Cascaded Timers

Figure 3-18

3.8 CASCADED TIMERS

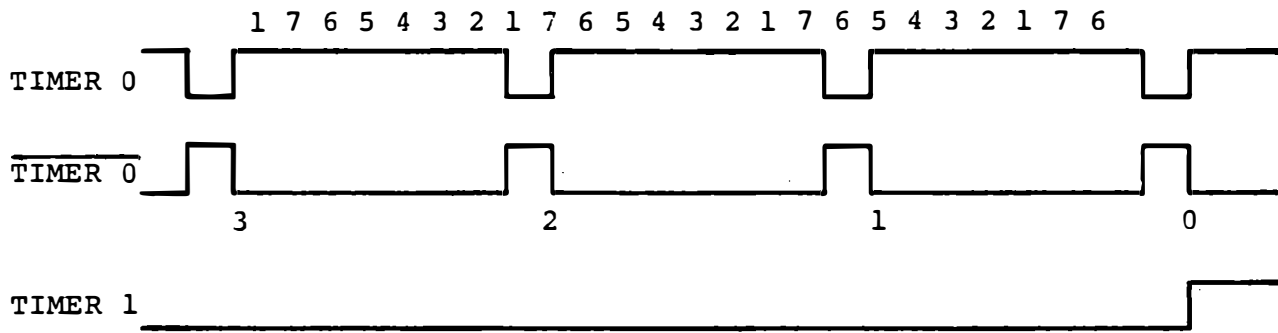
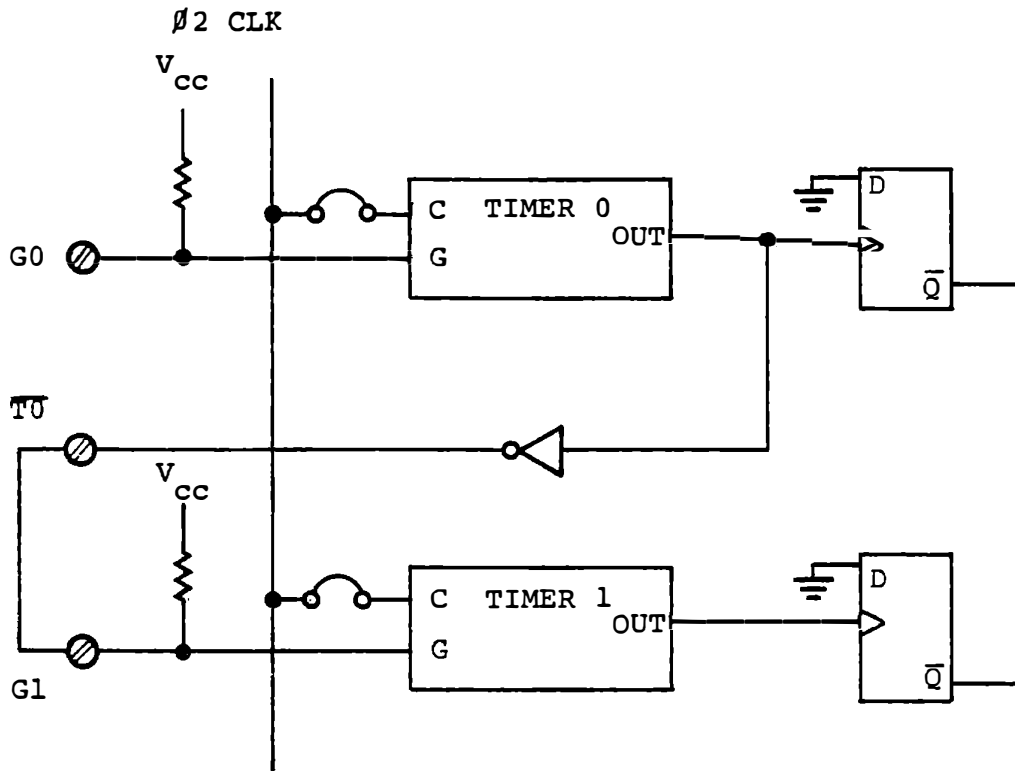
It is possible to use the output of one timer to control another, in either of two ways. One output can provide a clock to another timer, but on the experiment board this requires disconnecting the system clock from the second timer as shown in Figure 3-18. With this connection two timers in mode 2 can be cascaded to generate a long time interval:

Capacity	32 bits
Maximum Count	4,294,967,295 (decimal)
Time	2,097.152 seconds
	= 34 minutes, 57.152 seconds

A simpler connection, but with more restricted use, is to use the first timer output as a gate input to the second timer, as shown in Figure 3-19. Now if timer 0 is programmed to mode 2 its inverted output will enable the gate of timer 1 for exactly one clock pulse in each full count cycle of timer 0.

This is effective only if timer 0 is in mode 2, giving one pulse each count cycle, and timer 1 is in mode 0 or mode 4. In all other modes, the gate input rising edge restarts the counter by reloading it with the initial value from its storage register, so cascading can only be done with the clock input.

INTERVAL TIMERS



TIMER 0 GATING TIMER 1
 TIMER 0 IN MODE 2
 TIMER 1 IN MODE 0

Cascading Timers with Gate Input

Figure 3-19

EXERCISE

We will use the simpler connection from T0 OUT (at left of ITS board) to G1 IN to gate the second timer. In the program of Figures 3-20 and 3-21, we accept keyboard data for a time delay to be loaded to timer 1, which is in mode 0 and gated by timer 0. At the interrupt from timer 1 we shift a bit in the LED display as a visual indication.

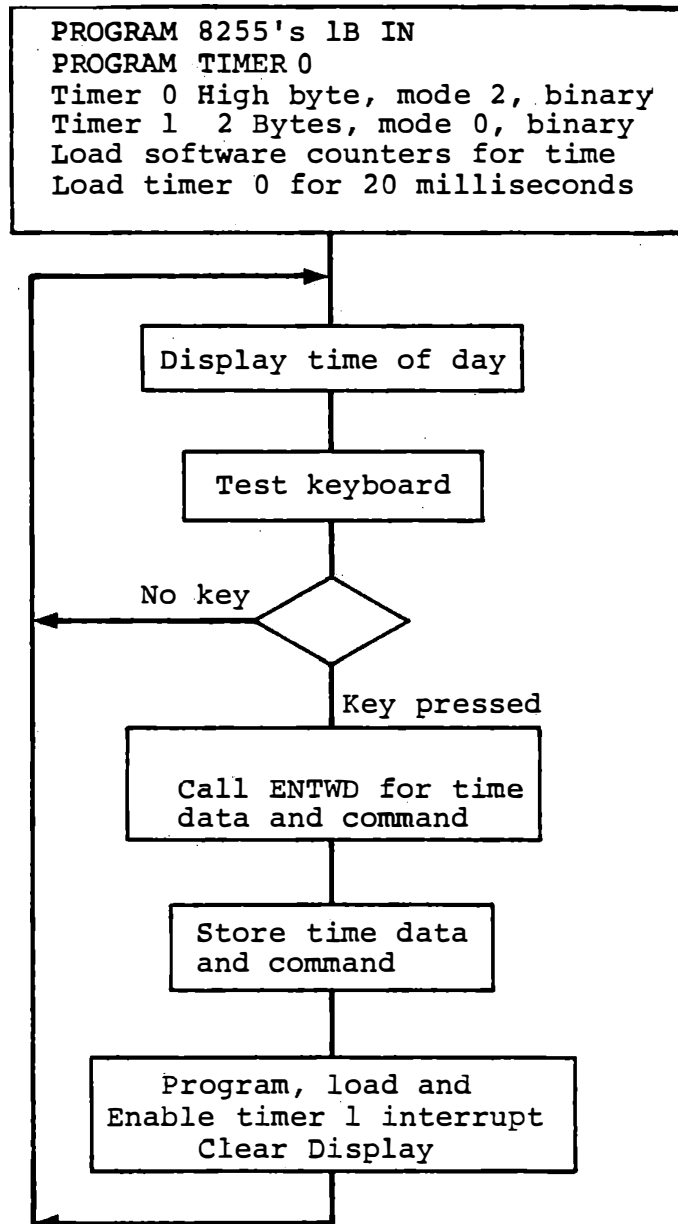
If the STEP key is pressed following the numeric data, the interrupt service routine disables the timer 1 interrupt, which is not restarted until a new keyboard entry is given. If the RUN key is pressed following the numeric data, then interrupt service reloads timer 1 and reenables the interrupt.

This program is designed to work concurrently with the time of day display of Figure 3-15. When no keyboard entry is made, the time of day display is shown. While ENTWD is accepting keyboard data, it controls the display

The effect of STEP and RUN commands here is analagous to mode 0 and mode 2 in the timers. With STEP timer 1 is decremented to zero and interrupts only once, like mode 0. With RUN it is reloaded and restarted each time it reaches zero.

INTERVAL TIMERS

This program can be instructive in other ways. Note that in the solution given we load timer 1 and then enable (or disable) its interrupt. If the time delay loaded is 0002, the RST6 interrupts will occur frequently. If the time loaded is 0000, the interrupts will occur very infrequently (once every 1310 seconds). If a value of 0001 is entered, no interrupts will occur at all. With this initial value the interval timer will not function correctly!



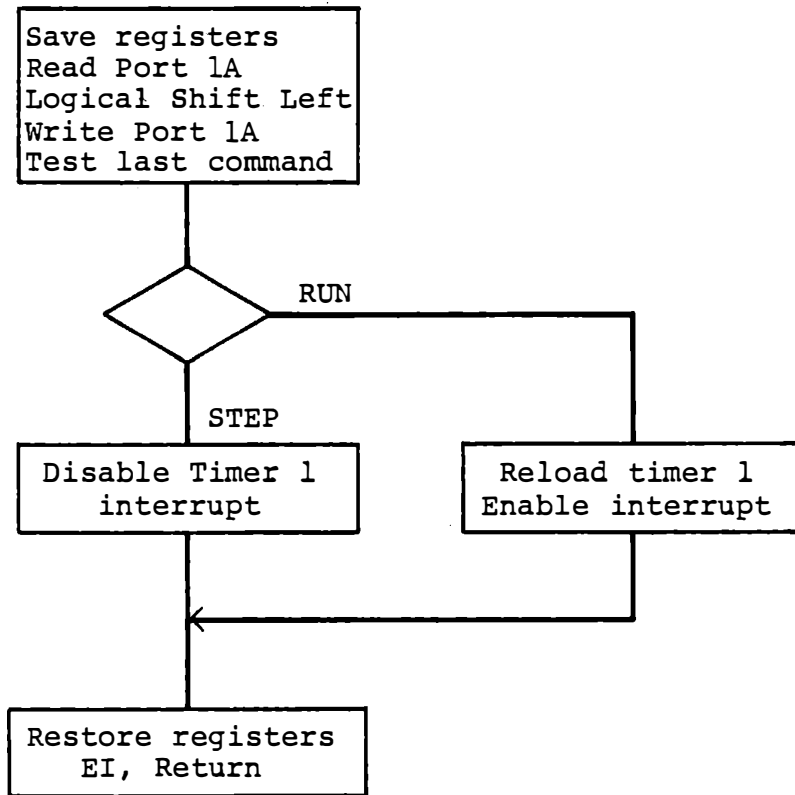
Time Delay Program - Main

Figure 3-20a

INTERVAL TIMERS

INTERRUPT SERVICE FOR TIMER 1

Figure 3-20b



Interrupt Service for Timer 1

Figure 3-20b

REVISED TIME CLOCK PLUS CASCADED TIMERS

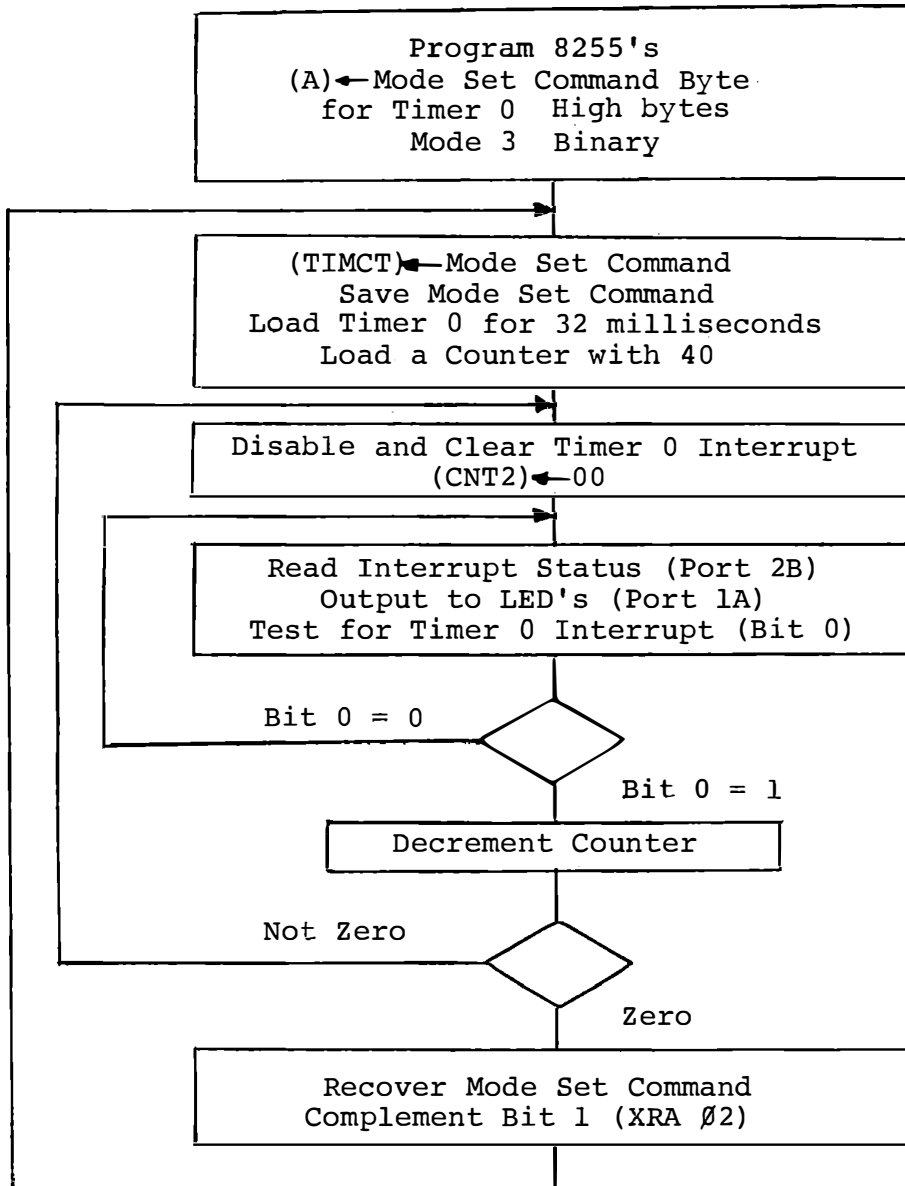
	A	D	D	R	CODE						
CODING SHEET	8	26	0	21		LXI	A,	8301			
			1	01							
			2	83							
			3	CD		CALL	DMEM				
			4	94							
			5	02							<i>same as</i>
			6	23		INX	H				<i>Figure 3-17d</i>
			7	7E		MOV	A, M				
			8	CD		CALL	DBY2				
			9	98							
MICROCOMPUTER TRAINING SYSTEM	A			02							
	B			23		INX	H				
	C			7E		MOV	A, M				
	D			CD		CALL	DBY2				
	E			98							
	F			02							
	8	27	0	DB		IN	PORTDA				<i>Test - hardware</i>
			1	00							<i>(A) ← FF^h in number</i>
			2	3C		INR	A				<i>Set zero bit in bus</i>
			3	CA		JZ	8260				<i>If no bus display</i>
		4	60							<i>'times of day' 1</i>	
		5	82							<i>0 p</i>	
		6	CD		CALL	ENTWD					
		7	46							<i>(A) ← command</i>	
		8	03							<i>(HL) ← time data</i>	
		9	32		STA	8304				<i>(8304) ← command</i>	
INTEGRATED COMPUTER SYSTEMS	A			04							
	B			83							
	C			22		SHLD	8305				<i>(8305.06) ← time</i>
	D			05							
	E			83							
	F			00		NOP					
	8		0								
			1								
			2								
			3								
		4									
		5									
		6									
		7									
		8									

Figure 3-21a

		A	D	D	R	CODE	REVISED CLOCK (continued)				
CODING SHEET	8	28	0	3E		MVI	A,	01			Set right LED on
			1	01							All others off.
			2	D3		OUT	PORT1A				"
			3	04							
			4	37		STC					Flag to enable
			5	CD		CALL	8290				Subr. program,
			6	90							loads, and enables
			7	82							Timer 1
			8	CD		CALL	CLEAR				Monitor subroutines
			9	87							clears display
MICROCOMPUTER TRAINING SYSTEM	A			02							
	B			C3		JMP	8260				Back to time display
	C			60							
	D			82							
	E										
	F										
	8	29	0	3E		MVI	A,	70			Subroutine to
			1	70							program, load and
			2	D3		OUT	TIMCT				enable Timer 1
			3	17							Both bytes
		4	7D		MOV	A,	L			Mode 0	
		5	D3		OUT	TIMI				Binary	
		6	15								
		7	7C		MOV	A,	H				
		8	D3		OUT	TIMI					
		9	15								
INTEGRATED COMPUTER SYSTEMS	A			3E		MVI	A,	01			(A) ← 03 to enable
	B			01							if CY set
	C			17		RAL					(A) ← 02 to disable
	D			D3		OUT	CNT2				if CY clear
	E			0F							
	F			C9		RET					
	8		0								
			1								
		2									
		3									
		4									
		5									
		6									
		7									
		8									Figure 3-21b

A D D R		CODE	8080 INTERRUPT SERVICE				
CODING SHEET	8	23 0	C3	JMP	82A0	Patch to	
		1	A0			Figure 3-17b	
		2	82				
		3					
		4					
		5					
		6					
		7					
		8					
		9					
MICROCOMPUTER TRAINING SYSTEM		A					
		B					
		C					
		D					
		E					
		F					
		8	2A 0	F5	PUSH	PSW	
		1	E5	PUSH	H		
		2	DB	IN	PORTIA		
		3	04				
	4	07	RLC				
	5	DB	OUT	PORTIA			
	6	04					
	7	2A	LHLD	8305	(HL) ← time delay		
	8	05					
	9	83					
INTEGRATED COMPUTER SYSTEMS		A	3A	LDA	8304	(A) ← last command	
		B	04				
		C	83				
		D	C6	ADI	EC	Set carry if RUN	
		E	EC			clear carry if STEP	
		F	CD	CALL	8290	To enable timer!	
		8	2B 0	90		if RUN, disable if	
		1	82			STEP.	
		2	E1	POP	H		
		3	F1	POP	PSW		
	4	FB	EI				
	5	C9	RET				
	6						
	7						
	8				Figure 3-21c		

INTERVAL TIMERS



Square Wave Generator - Mode 3

Figure 3-22

3.9 MODE 3 SQUARE WAVE GENERATOR

When programmed in mode 3, a timer repeatedly counts down from its initial value, starting with its output high. Halfway through the count, the output goes low. At zero, the output goes high and the initial value is reloaded from the count register. Thus a square wave is generated. If the initial value is an odd number, the first half of the count will be one bit time longer than the second half.

The gate input disables counting when it is low. At a rising edge of the gate input, the initial value is reloaded from the count register into the counter. The output becomes high and a new complete cycle starts.

If the count register is reloaded while the timer is running in this mode, the current half period of counting will be completed with the old value. The new value will become effective when the output changes in either direction, or at a rising edge of the gate input.

3.9.1 Observing the Output

EXERCISE

Write a program that will change the mode of timer 0 every few seconds, alternating between mode 2 and mode 3 (Figure 3-22 shows a flow diagram). Observe the inverters in one of these ways:

- a) With an oscilloscope
- b) With a voltmeter
- c) By connecting TO OUT to EXT4 IN, reading Port 2B and displaying its data in the LEDs.

INTERVAL TIMERS

An oscilloscope permits direct observation of the inverted square wave at T0 OUT and additional experiments. The voltmeter across T0 OUT and GND will show a low output (0.4 volts) when the timer is running in mode 2, but about 2 volts in mode 3. With the jumper connected (as in 'c' above), LED DS6 will be visibly illuminated in mode 3 but not in mode 2.

3.9.2 Observing the Counting

The square wave generator conceivably could operate in any one of three ways:

- (a) Divide initial value by 2 before loading the counter from the count register. Toggle the output and reload at zero.
- (b) Load the counter with the initial value, and decrement by 2 at each clock. Toggle the output and reload at zero.
- (c) Compare the counter content with half of the initial value, and set the output low at equal. Set the output high at zero.

The following exercise permits you to determine which of these is actually used.

EXERCISE

Program a timer for mode 3 operation, low byte only (Figure 3-23). Load it with 7E. In a loop, repeatedly latch and read the timer while it is counting. Display the byte in the LEDs of port 1A. Determine from this how the timer really operates in mode 3.

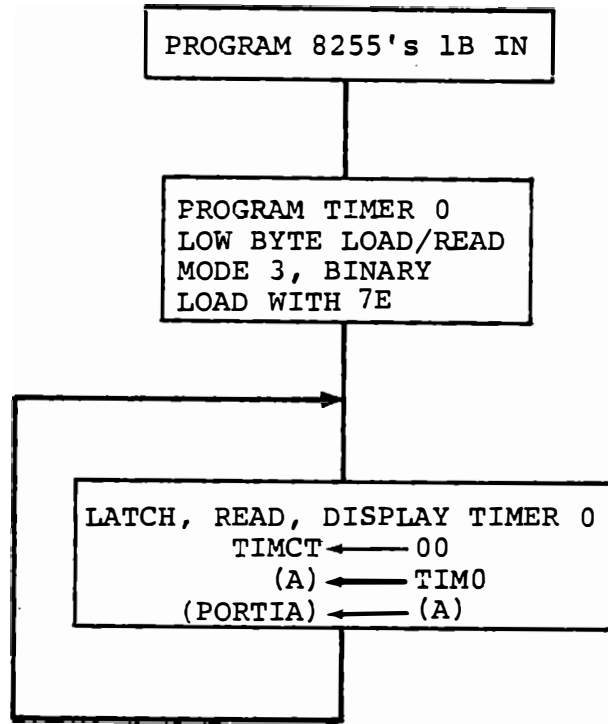
If (a) is true, the LEDs will never show a value greater than half of the initial value.

If (b) is true, the least significant bit will never change.

If (c) is true, the full value will be shown and the least significant bit will count.

This experiment is suggested because the manufacturer's literature does not state how the function is performed. It is sometimes necessary to know a detail that the manufacturer did not consider important.

INTERVAL TIMERS



Reading the Timer Contents

Figure 3-23

3.10 TIMER MODE DESCRIPTIONS

This section defines all six modes of the timer, including modes 0, 2 and 3 which have previously been discussed as well as the three modes that have been neglected.

The modes differ principally in the effect of the gate input and the behavior of the output.

<u>Modes</u>	<u>Gate Input</u>		
	<u>Low</u>	<u>Rising Edge</u>	<u>High</u>
0,4	Disables Counting		Enables Counting
2,3	Disables Counting	Reloads counter with initial value and initiates counting.	Enables Counting
1,5		Reloads counter with initial value and initiates counting	
<u>Modes</u>	<u>Output Signal</u>		
0,1	Low while counting High at count = 0		
2,4,5	High while counting Low for one clock period		
3	High during first half cycle Low during second half cycle		

INTERVAL TIMERS

<u>Modes</u>	<u>After Terminal Count</u>
0,4	Counting continues but output remains high
1,5	Counting stops until a new gate rising edge occurs
2,3	Counting starts again from the initial value and output pattern repeats for each full count cycle

Figure 3-24 shows more detail of the gate effect and output timing, and the following sections define each mode in detail. Figure 3-25 indicates the timing relationships. Note that mode 0 and mode 4 are similar except for the output state during counting, but for a given count loaded to the timer, mode 4 will generate an interrupt one clock time later than mode 0. The same relationship is true of modes 1 and 5

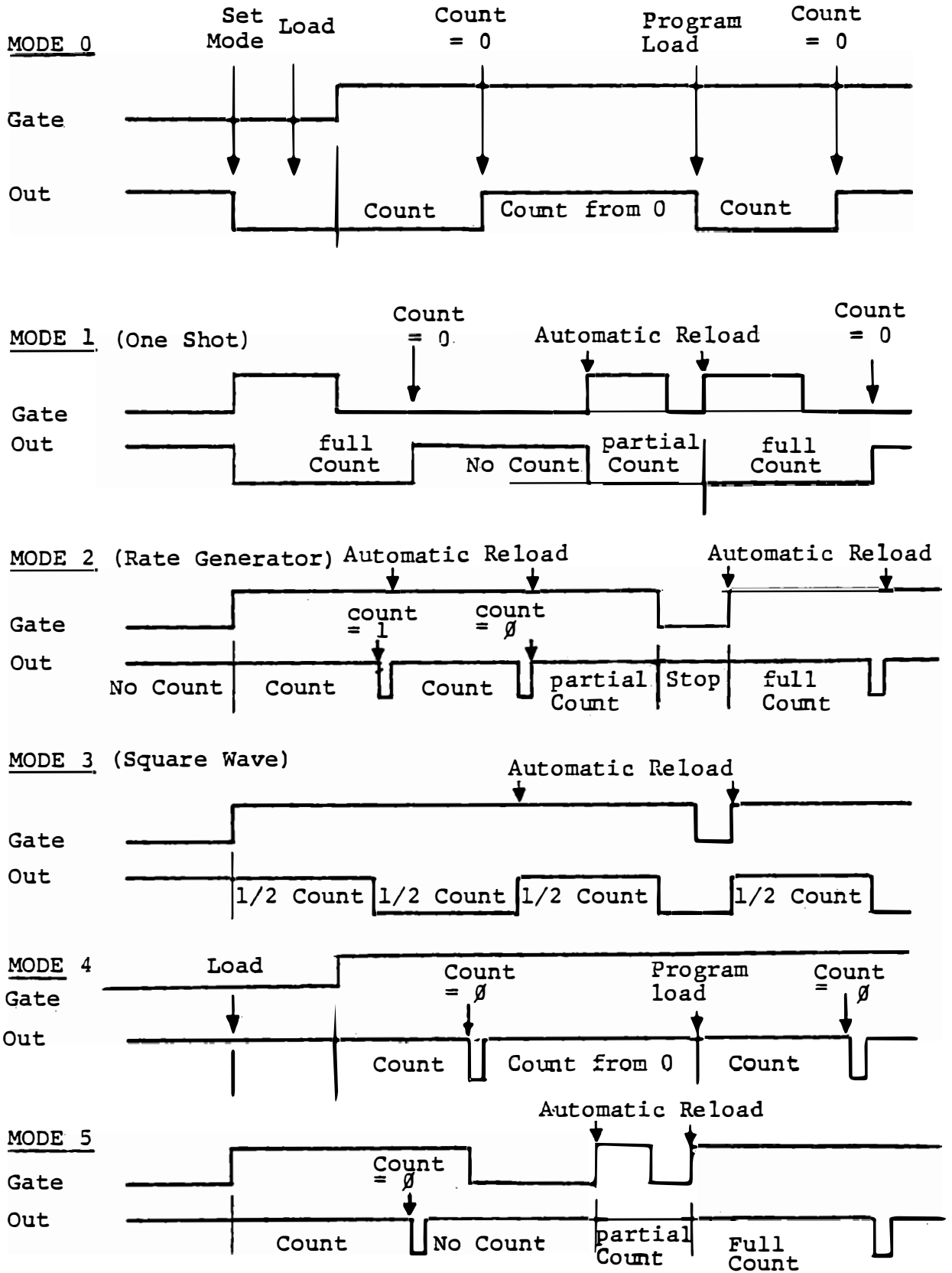
Mode	Output after mode set	Starts counting	Output goes low	Output goes high	Count restarted	Comments
0 Interrupt	Low	When final byte loaded	At mode set	At zero	By reloading	Output is set low by setting mode or by reloading.
1 One Shot	High	After gate rising edge	After gate rising edge	At zero	By gate rising edge	Can be preloaded during counting. Present period not affected. New value effective for next period.
2 Rate Generator	High	When final byte loaded	At count=1	At zero	At zero or by gate rising edge	
3 Square Wave	High	When final byte loaded	At $n/2$ or $(n + 1)/2$	At zero	At zero or by gate rising edge	If loaded while counting new period is effective for next half of total period.
4 Software Strobe	High	When final byte loaded	At zero	At next clock after zero	By reloading	
5 Hardware Strobe	High	After gate rising edge	At zero	At next clock after zero	By gate rising edge	If loaded while counting new period is effective after next gate rising edge.

INTERVAL TIMERS

8253 Timer Modes

Figure 3-24

INTERVAL TIMERS



Timing Relationships

Figure 3-25

Mode 0 Interrupt on Terminal Count

The timer counts down from the initial value and continues from zero. The output goes low when mode 0 is set or when new data is loaded. The output goes high when the count reaches zero. Counting starts when the final byte of the initial value is loaded. If a new value is loaded during counting, loading the first byte stops the count and sets the output low. Mode 0 is useful for generating a single time delay function or for measuring time from a programmed or external event, providing that the time is less than the 32 millisecond capability of the 16 bit counter. It can be used to measure the duration of an external signal, since counting is enabled only when the gate input is high.

Mode 1 Programmable One Shot

Starts counting down from the initial value after a rising edge of the gate input. The output goes low at the first count after the gate rising edge, high at zero. Counting starts again from the initial value each time a rising edge occurs at the gate input. Mode 1 is useful for generating a time delay or measuring time from an external event, especially if the external event is a narrow pulse.

Mode 2 Rate Generator

The output goes high when the mode is set. After the count has been loaded, the timer will repetitively count down from the initial value to zero. The output goes low when the count reaches one and high when it is zero, so a 0.5 microsecond pulse is generated. Mode 2 is especially useful for timing functions where software counters are to be used n times greater than the 32 millisecond capacity of the timers. Counting restarts from the initial value immediately after zero is reached, so a delay before the program services the counter does not introduce any uncertainty in the timing. If the counter register is reloaded during counting, the present period is not affected, but the new value is effective for subsequent periods. The gate input inhibits counting when it is low. A rising edge restarts the counter from the initial value.

INTERVAL TIMERS

Mode 3 Square Wave Rate Generator

The output goes high when the mode is set. After the count has loaded, the timer will repetitively count down from the initial value to zero. The output will go low when the count reaches half the initial value and high when the count reaches zero, so a square wave is generated. If the initial value is odd, the output will be high for $(n+1)/2$ counts and low for $(n-1)/2$ counts. If the counter register is reloaded during counting, the present half cycle is not affected, but the new value is effective for the next half cycle and subsequent periods. The gate input inhibits counting when it is low. A rising edge restarts the counter from the initial value.

Mode 4 Software Triggered Strobe

The timer counts down from the initial value. The output goes high when the mode is set, low when the count reaches zero, then high at the next clock pulse after zero. If the counter is reloaded while it is running, the new count is effective immediately after the final byte has been loaded. The gate input inhibits counting when it is low.

Mode 5 Hardware Triggered Strobe

Starts counting down from the initial value after a rising edge of the gate input. The output goes high when the mode is set, low when the count reaches zero, then high at the next clock pulse after zero. Counting starts again from the initial value each time a rising edge occurs at the gate input. If the count register is reloaded during counting, The present period is not affected. The new count is effective when the next gate rising edge occurs.

MICROCOMPUTER INTERFACING WORKBOOK

CHAPTER 4

DIGITAL TO ANALOG OUTPUT

This page intentionally left blank.

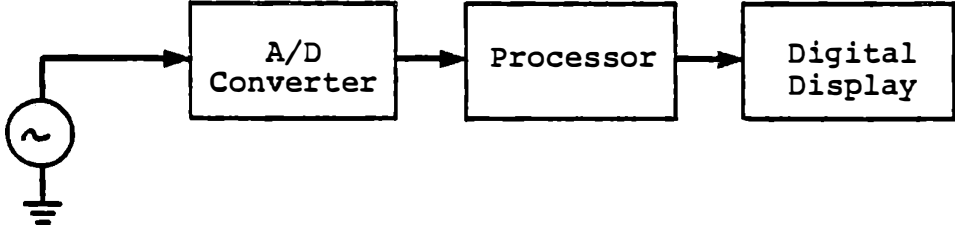
4. DIGITAL TO ANALOG OUTPUT

Very commonly a computer or microprocessor in a control system must receive or generate an analog signal - a signal which represents some value in a continuous range of values, rather than a binary 0 or 1. An analog signal may be a variable voltage which is the output of a measuring instrument or the input to a control driver: the voltage is like the rate of flow, the temperature, the position, that is being measured or controlled; it is an analog of the real variable.

In this chapter we will experiment with several means by which the computer can generate an analog signal. In chapter 5, we will investigate the opposite task, of converting an analog signal into a digital form that the computer can process. Instruments usually have a one-way conversion, but control systems very often need both, as suggested in Figure 4-1.

DIGITAL TO ANALOG OUTPUT

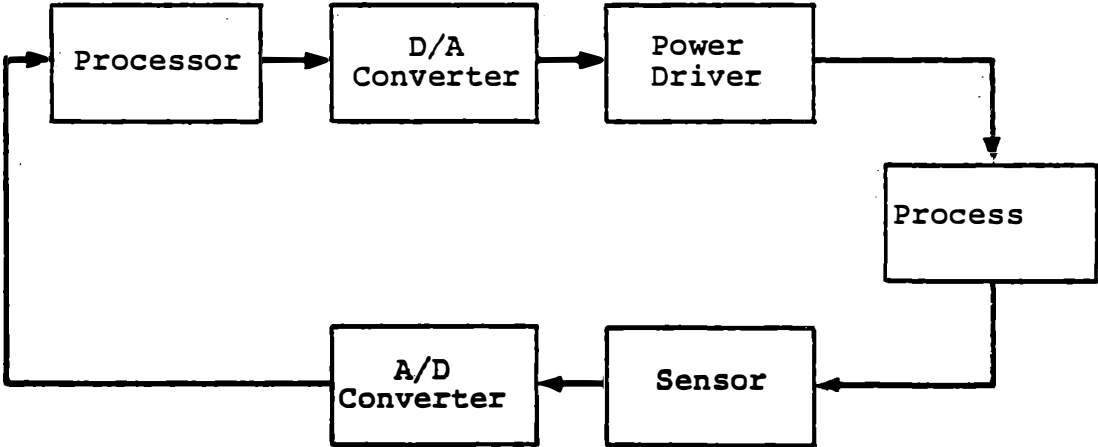
DIGITAL VOLTMETER



Function Generator



Closed Loop Process Control



A/D and D/A Conversion

Figure 4-1

4.1 METHODS OF D/A OUTPUT

Although a variable voltage is one of the most common analogs, there are many others, each having particular advantages in appropriate circumstances. We will discuss some of these in the next chapter, which deals with analog input; here we introduce the few that are especially suited for analog output from the computer.

It is not always clear where analog conversion ends. A computer might output a set of binary data which is converted to a voltage input to an op-amp, which drives a power transistor, whose output current controls a hysteresis motor that generates a torque to precess a gyro. The ultimate conversion was from digital data to a new position for the guidance gyro: enroute, we have had a voltage, a current, magnetic flux, magnetic force, mechanical force and a precessing torque; each of these was an analog of the desired motion.

There are two basic approaches that can be used for output from the digital processor to give a variable value: the output may be several binary signals representing a number that is converted to a voltage or current; or the output may be a single bit with time as the continuous variable, so that either frequency or average power output is the analog. In the latter case some external device, often the load which is being driven, must integrate the signal.

DIGITAL TO ANALOG OUTPUT

For example, the binary signal may turn a heater on and off to maintain a desired average temperature; a bimetal thermostat controlling a household furnace does exactly that, and the building integrates the binary (on and off) condition of the furnace.

In this chapter, we will consider four digital to analog conversions:

- Pulse width modulation

- Frequency modulation

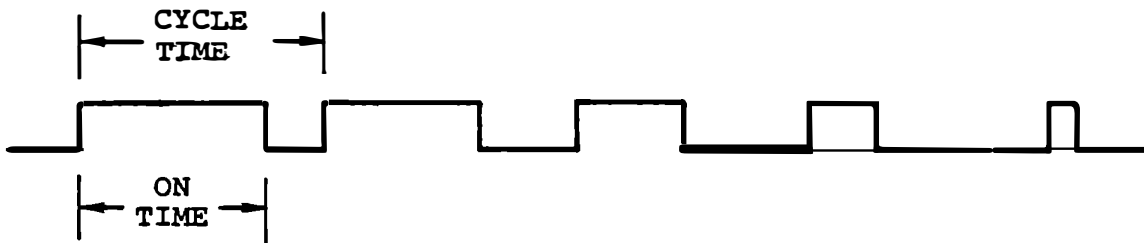
- Direct multi-bit output

- Ladder network digital to voltage conversion

4.2 PULSE WIDTH MODULATION

Pulse Width Modulation

Pulse width modulation (pulse duration modulation or pulse-time modulation) is a technique to vary average power output over time by varying the ratio of power source on-time versus cycle-time (one on-time plus off-time cycle). The implementation we shall discuss is the switching of a binary output on and off with varying duration to generate an average power output.



The figure shows a signal whose average power is decreasing over time; there is a constant cycle-time, but a varying on-time whose duration is decreasing. Although the constant cycle-time is not necessary, it simplifies the computation. Some loads that might be driven may limit the minimum or maximum on-time, in which case the cycle-time must be varied in order to vary the on-time/cycle-time ratio (duty cycle).

Pulse width modulation has three great advantages for analog output: it requires only a single bit from the processor to switch from the on state to the off state; it allows the load power to be controlled by a switching device such as a relay or SCR rather than a power

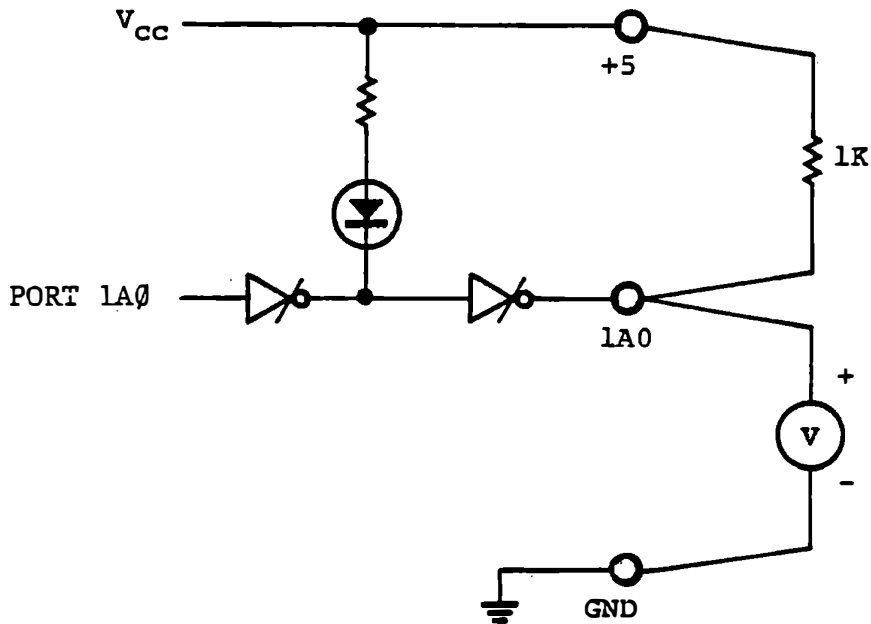
DIGITAL TO ANALOG OUTPUT

amplifier; and it minimizes power dissipation (heat loss) in the control device.

4.2.1 PWM Output Program

EXERCISE

We will develop a program to generate a pulse width modulated output signal, with keyboard entries to set the cycle time and the duty cycle. (Duty cycle = on-time/cycle-time). We will drive one of the port 1A outputs with this signal and observe the result with a voltmeter. It will also be visible to some degree in the brightness of the LED. Figure 4-2 shows the connections required.



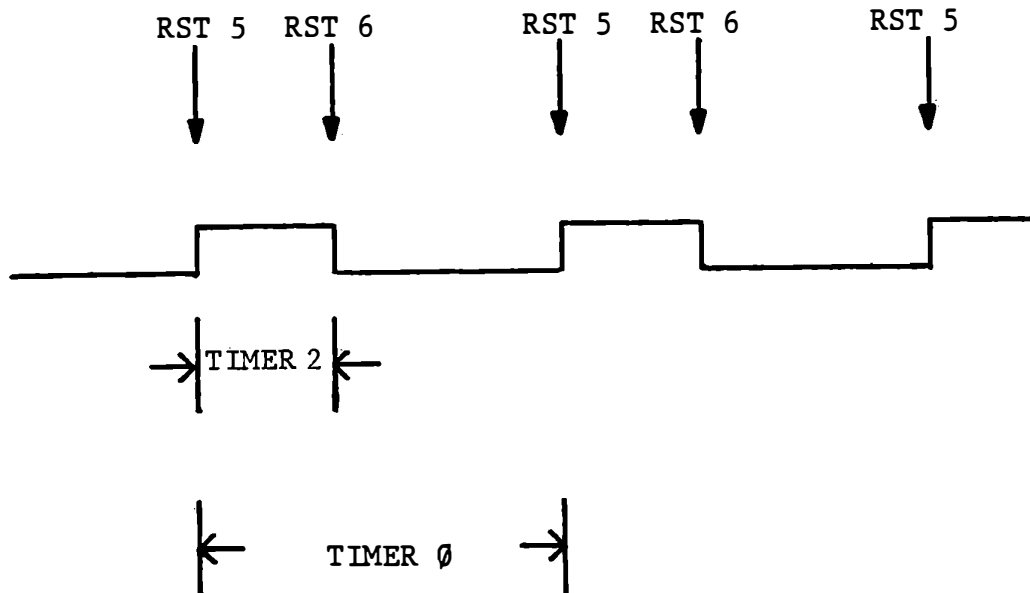
Output Connections for PWM

Figure 4-2

4.2.1.1 PWM Program Operation

We will use two timers and two interrupt service routines to control the PWM output signal. Timer 0, operating in mode 2, will control the uniform cycle time. It will repetitively count down from its initial value, generating a RST 5 interrupt when it reaches zero. The RST 5 service routine will turn the output signal on, load Timer 2 with the on-time and enable the Timer 2 interrupt. It will also reenale the Timer 0 interrupt to clear the latch.

When Timer 2 counts down to zero, an RST 6 interrupt will occur. The RST 6 interrupt service routine will turn the output signal off, and disable Timer 2 interrupt:



DIGITAL TO ANALOG OUTPUT

Since timer 0 generates RST 5, then only timer 2 will be enabled for RST 6; the program will distinguish the two interrupts by their RST instructions. For the moment, let us ignore the main program and examine the interrupt service routines in Figure 4-3.

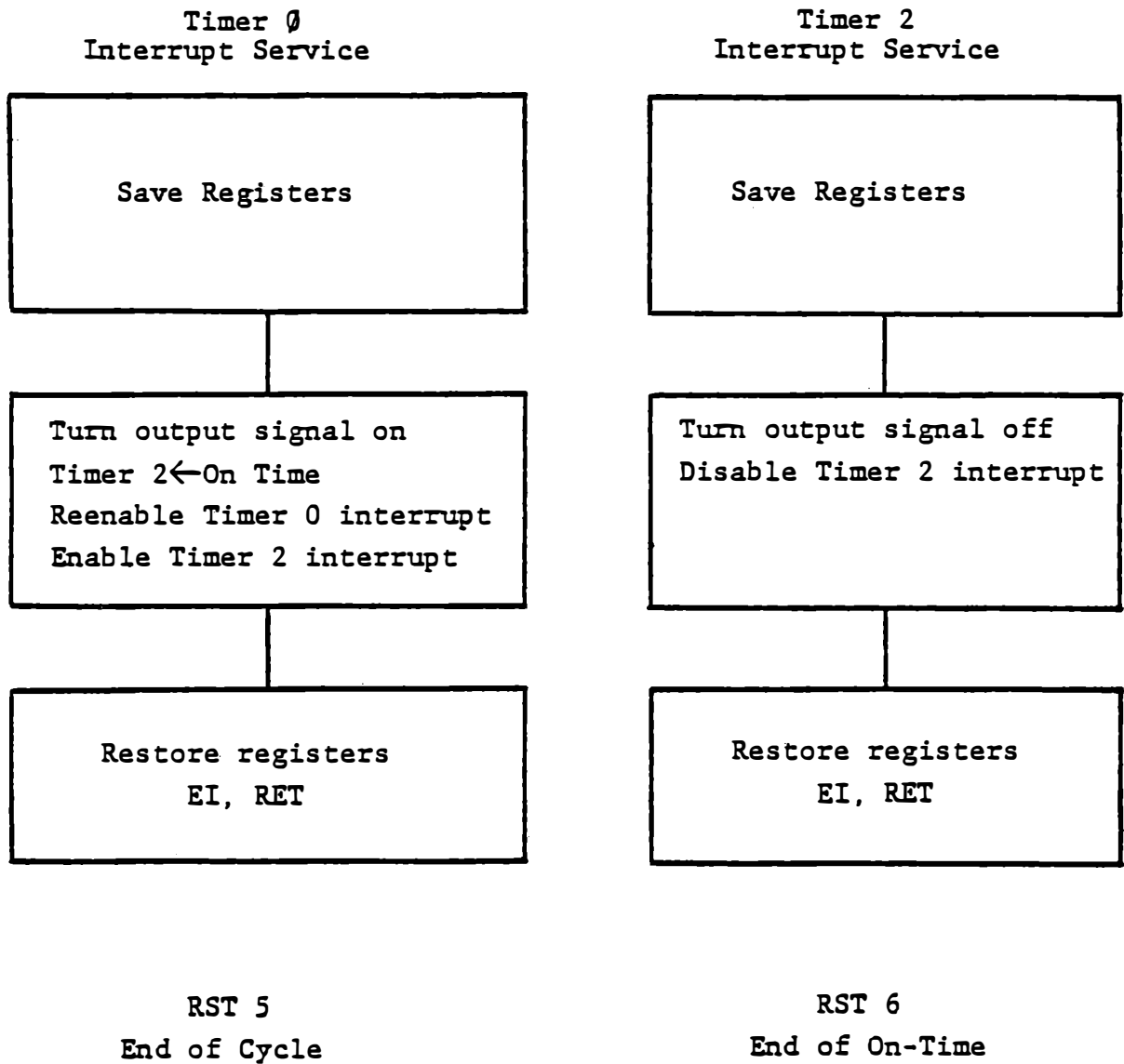
4.2.1.2 PWM Interrupt Service

At the end of a cycle, timer 0 generates an RST 5 interrupt. After saving the registers, we turn the output signal on. The 8255 does not have individual bit control for its port A, but we can achieve it by:

```
IN  PORT1A
ORI 01
OUT PORT1A
```

Even when a port is programmed for output, its content can be read by the program. This allows restoration of all bits in port 1A except the one we want to change. Since we are not using the other bits, this procedure is not really needed, but in some other programs it is a useful technique.

Now the RST 5 routine loads timer 2, which is operating in mode 0; enables both interrupts, and exits. The output signal has been turned on and will stay on until a timer 2 interrupt occurs.

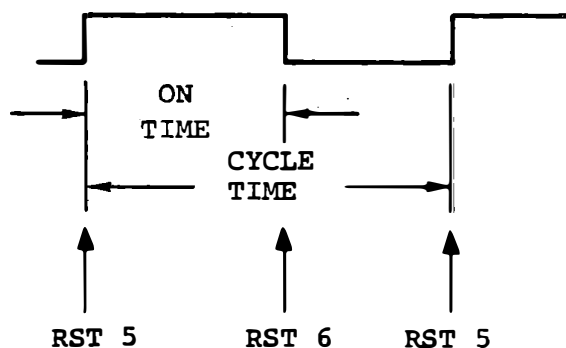


PWM Interrupt Service

Figure 4-3

DIGITAL TO ANALOG OUTPUT

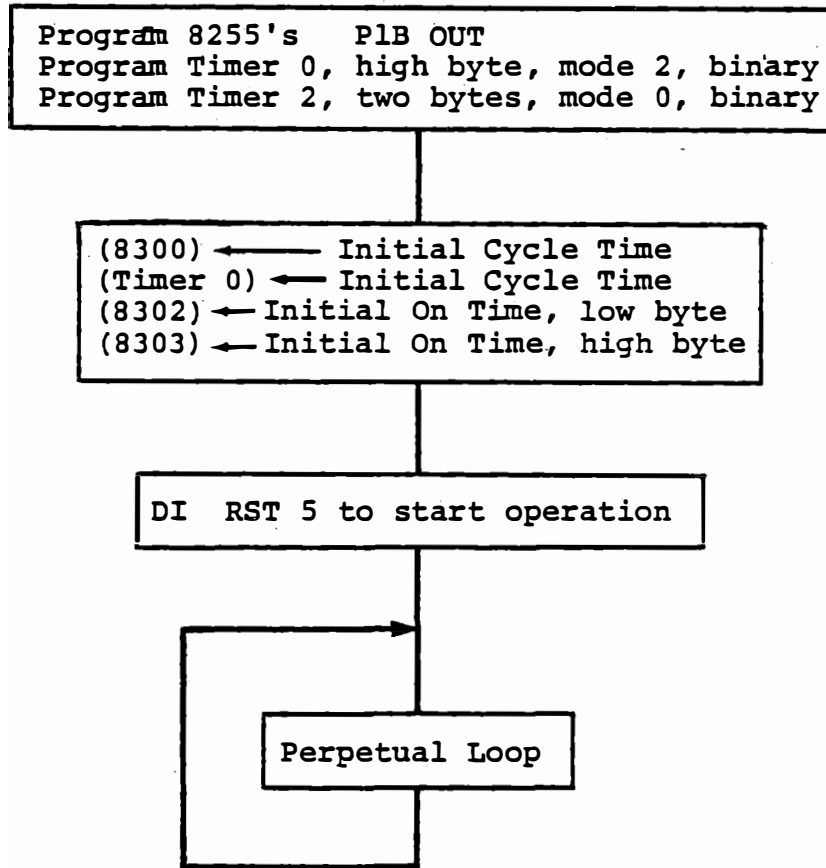
The RST 6 interrupt service routine is invoked by timer 2. It turns the output signal off; disables its own interrupt, and exits. Now the output will stay off until the timer 0 interrupt service turns it on again.



The time loaded to timer 0 sets the cycle time; the time loaded to timer 2 sets the on-time.

4.2.1.3 PWM Test Program

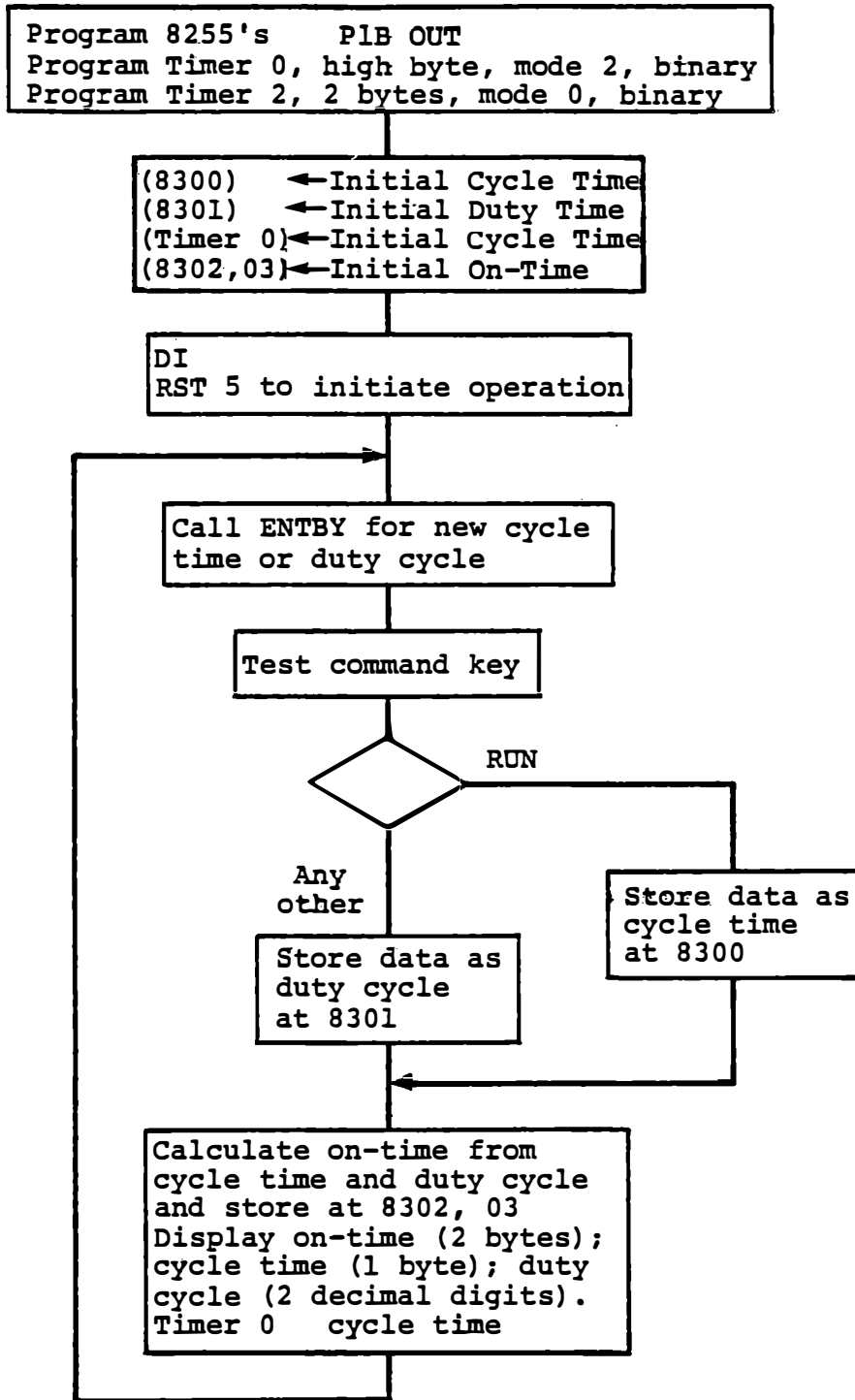
Write the interrupt service routines according to Figure 4-3, and a trivial main program to program the ports and timers as shown in Figure 4-4. This will prove the hardware interface and the interrupt service routines. The average power can be changed by entering different initial values for cycle time and on-time.



PWM Test Program

Figure 4-4

DIGITAL TO ANALOG OUTPUT



PWM Main Program

Figure 4-5

4.2.1.4 PWM Main Program

After testing the hardware and interrupt service routines, we will develop a more interesting program to allow keyboard control of cycle time and duty cycle. (Duty Cycle = on-time/cycle time.) The program of Figure 4-5 calls the monitor subroutine ENTBY to obtain a one byte value and a command key:

```
CD  CALL ENTBY
    36
    03
```

ENTBY accepts numeric keys, always returning the last two keys entered as a byte in register L, and returns when a command key has been pressed and released, with the command key value in registers A and C. All registers except E are used. (See Course 525, Section 6.10.3) During operation most of the time will be spent scanning the keyboard, waiting for keyboard entries. Although the monitor program, as a whole cannot be interrupted (since it disables the interrupt), any of its subroutines can be, so the PWM interrupt service routines will control the output.

DIGITAL TO ANALOG OUTPUT

When keyboard data are returned by ENTBY, we will test the command key (register A): if it is RUN (=14), the data byte (register L) will be stored as a new cycle time (and subsequently written to timer 0); other wise, the data byte will be taken as a decimal duty cycle.

We use a mixed number system in this program. Cycle time and on-time are kept as binary numbers, but duty cycle is accepted, stored and displayed as a decimal fraction. A subroutine (at 8290) multiplies the binary cycle time by the decimal fraction duty cycle to obtain a two byte binary on-time. This unorthodox procedure is used because it is easy to choose a cycle time from the table given in Figure 4-6, but binary fractions expressed in hexadecimal are awkward to handle mentally.

DIGITAL TO ANALOG OUTPUT

Binary Count	Decimal Count	Time (msecs)
0100	256	0.125
0200	512	0.250
0400	1024	0.500
0800	2048	1.000
1000	4096	2
1800	6144	3
2000	8192	4
2800	10240	5
3000	12288	6
3800	14336	7
4000	16384	8
4800	18432	9
5000	20480	10
A000	40960	20
F000	61440	30
0000	65536	32

Conversion of Binary Count to Time

Figure 4-6

DIGITAL TO ANALOG OUTPUT

4.2.1.5 Use of the PWM program

Write your complete program in accordance with Figures 4-3 and 4-5. You can test and debug the data entry, display and multiplication sections without enabling the interrupts by omitting the DI and RST 5 instructions. The solution given in Figure 4-7 is subdivided as follows:

4-7a	8200-8223	Initialization
4-7b	8228-823F	RST 5 entry and RST 6 processing
4-7c	8240-8257	RST 5 Processing
4-7d	8260-8288	Main Program Loop
4-7e	8290-82AA	Subroutine BVXDF

To run the given program, depress RST, then RUN. The voltmeter should show about 2-1/2 volts, due to an initial cycle-time of 50H(10ms) and an initial duty cycle of 50%. Keying in a decimal duty cycle (1-99), followed by the NEXT key (any command key except RUN) will change the duty cycle accordingly and display it in two right hand digits. The four left digits will contain the binary count on-time (in hexadecimal clock pulses, see Figure 4-7). Keying in a hexadecimal value followed by the RUN key, will change the cycle timer to the new value and display it in the remaining two display digits.

DIGITAL TO ANALOG OUTPUT

When the entire program is operating the voltmeter (connected as shown in Figure 4-2) will display a value proportional to the duty cycle. This depends on the mechanical inertia of the voltmeter to integrate the signal; a digital voltmeter will be confused by the PWM signal unless it has an averaging or true RMS capability.

The average output voltage is proportional to the duty cycle, and independent of the cycle time. With a maximum cycle time, you may be able to see some slight motion of the voltmeter needle; or if you make the cycle time = 16.625 milliseconds, you may see it with the stroboscopic effect of fluorescent lighting.

		A	D	D	R	CODE	PULSE WIDTH MODULATION OUTPUT							
CODING SHEET	8 20	0	3	E		M	V	I	A	, 80		Program 82554 PIB Out		
		1	8	0										
		2	D	3		O	U	T	C	N	T 1			
		3	0	7										
		4	3	E		M	V	I	A	, 92				
		5	9	2										
		6	D	3		O	U	T	C	N	T 2			
		7	0	F										
		8	3	E		M	V	I	A	, 24			Program 2 high level mode 2, program 0	
		9	2	4										
	A	D	3		O	U	T	T	I	M	C	T		
	B	1	7											
MICROCOMPUTER TRAINING SYSTEM		C	3	E		M	V	I	A	, B0		Program 3 2 nd burst mode 0, program 0		
		D	B	0										
		E	D	3		O	U	T	T	I	M		C	T
		F	1	7										
		8 21	0	2	1		L	X	I	H	, 5050			
		1	5	0									(L) ← Cycle time 10ms (H) ← Duty cycle 50%	
		2	5	0										
		3	2	2		S	H	L	D	8	3			0
	4	0	0											
	5	8	3											
	6	7	D		M	O	V	A	, L					
	7	D	3		O	U	T	T	I	M	0			
	8	1	4											
INTEGRATED COMPUTER SYSTEMS		9	2	1		L	X	I	H	, 2800		(HL) ← Initial On time = 5 milliseconds		
		A	0	0										
		B	2	8										
		C	2	2		S	H	L	D	8	3		0	2
		D	0	2										
		E	8	3										
		F	F	3		D	I							
		8 22	0	E	F		R	S	T	5				
		1	C	3		J	M	P	8	2	6		0	
		2	6	0										
	3	8	2											
	4	0	0											
	5	0	0											
	6	0	0											
	7	0	0											
	8													

Figure 4-7a

INTERRUPT SERVICE FOR PWM

		A	D	D	R	CODE												
CODING SHEET	8	0																
		1																
		2																
		3																
		4																
		5																
		6																
		7																RST 5 - Timer 0
MICROCOMPUTER TRAINING SYSTEM	8	22	8	F5		PUSH		PSW										
		9	E5		PUSH		H											
		A	C3		JMP		8240											
		B	40															
		C	82															
		D	00		NO P													
		E	00															
		F	00															
MICROCOMPUTER TRAINING SYSTEM	8	23	0	F5		PUSH		PSW										RST 6 - Timer 2
		1	E5		PUSH		H											
		2	DB		IN		PORT 1 A											Turn output
		3	04															signal at 1A0
		4	E6		ANI		EE											out. Do not
		5	FE															change other bits
		6	D3		OUT		PORT 1 A											
		7	04															
INTEGRATED COMPUTER SYSTEMS		8	3E		MVI		A, 04											Disable timer 2,
		9	04															interrupt
		A	D3		OUT		CNT 2											
		B	0F															
		C	E1		POP		H											
		D	F1		POP		PSW											
		E	FB		EI													
		F	C9		PET													
INTEGRATED COMPUTER SYSTEMS	8	0																
		1																
		2																
		3																
		4																
		5																
		6																
		7																
	8																	Figure 4-7b

		A	D	D	R	CODE	INTERRUPT SERVICE FOR TIMER 0 (continued)				
CODING SHEET	8	24	0	DB		IN		PORT	1A		Read Port 1A
			1	04							existing data
			2	F6		ORI		01			set bit 1 high
			3	01							
			4	D3		OUT		PORT	1A		Output data
			5	04							
			6	2A		LHLD		8302			(HL) ← on time
			7	02							
			8	83							
			9	7D		MOV		A,	L		Timer 2 ← On time
MICROCOMPUTER TRAINING SYSTEM	A			D3		OUT		TIM	2		
	B			16							
	C			7C		MOV		A,	H		
	D			D3		OUT		TIM	2		
	E			16							
	F			3E		MVI		A,	01	Reenable timer 0	
	8	25	0	01						interrupt	
			1	D3		OUT		CNT	2		
			2	0F							
			3	3E		MVI		A,	05	To enable timer 2	
		4	05						interrupt		
		5	C3		JMP		823A		Jump to enable		
		6	3A						and return		
		7	82								
INTEGRATED COMPUTER SYSTEMS	8										

Figure 4-7c

PWM OUTPUT - PROGRAM LOOP

		A	D	D	R	CODE										
CODING SHEET	8 26	0	C	D		C	A	L	L	E	N	T	B	Y		
		1		3	6											
		2		0	3											
		3		F	E		C	P	I		R	U	N		Test for command key = RUN	
		4		1	4											
		5		7	D		M	O	V	A,	L				(A) ← keyed bytes	
		6		2	1		L	X	I	H,	8	3	0	0	Address for cycle time	
		7		0	0											
		8		8	3											
		9		C	A		J	Z			8	2	6	D	Jump if RUN key	
MICROCOMPUTER TRAINING SYSTEM	A		6	D												
	B		8	2												
	C		2	3		I	N	X		H				Address duty cycle		
	8 26	D	7	7		M	O	V	M,	A				Store cycle time		
		E	2	A		L	H	L	D		8	3	0	0	or duty cycle	
		F		0	0										(L) ← cycle time	
	8 27	0	8	3											(H) ← duty cycle	
		1		E	5		P	U	S	H	H				Save for display	
		2		C	D		C	A	L	L	B	V	X	D	F	Multiply
		3		9	0											Binary Value
	4		8	2											X Decimal Fraction	
	5		2	2		S	H	L	D		8	3	0	2	Store on time	
	6		0	2											at 8302, 03	
	7		8	3												
	8		C	D		C	A	L	L	D	W	O	R	D	Display on time	
	9		D	1											at left (2 bytes)	
INTEGRATED COMPUTER SYSTEMS	A		0	2												
	B		E	1		P	O	P		H					(L) ← cycle time	
	C		7	C		M	O	V	A,	H					(H) ← duty cycle	
	D		C	D		C	A	L	L	D	B	Y	T	E	Display duty	
	E		9	5											cycle at right	
	F		0	2												
	8 28	0	7	D		M	O	V	A,	L					(A) ← cycle time	
		1		D	3		O	U	T	T	I	M	O			TIM0 ← cycle time
		2		1	4											
		3		C	D		C	A	L	L	D	B	Y	2		Display cycle time
	4		9	P											between on time	
	5		0	2											and duty cycle	
	6		C	3		J	M	P		8	2	6	0		Loop	
	7		6	0												
	8		8	2												

Figure 4-7d

SUBR-MULTIPLY BINARY VALUE X DECIMAL FRACTION

		A	D	D	R	CODE										
CODING SHEET	8	29	0	4C		M	O	V	C	,	H				(C) ← decimal fraction	
			1	55		M	O	V	D	,	L				(D) ← binary value	
			2	21		L	X	I	H	,	0	0	0	0	Clear product	
			3	00											in (HL) and	
			4	00											low byte of	
			5	5D		M	O	V	E	,	L				binary value in (E)	
		829	6	7A		M	O	V	A	,	D				LOOP - Test binary	
			7	B3		O	R	A	E						value and return	
			8	C8		R	Z								when zero	
			9	7A		M	O	V	A	,	D				Shift two byte	
MICROCOMPUTER TRAINING SYSTEM	A			1F		R	A	R						binary value		
	B			57		M	O	V	D	,	A				right one bit	
	C			7B		M	O	V	A	,	E				for half of	
	D			1F		R	A	R							preceding value	
	E			5F		M	O	V	E	,	A					
	F			79		M	O	V	A	,	C				(A) ← decimal fraction	
	8	2A	0	87		A	D	D	A						Double decimal	
			1	27		D	A	A							fraction	
			2	4F		M	O	V	C	,	A				(C) ← remainder	
			3	D2		J	N	C	8	2	9	6				Do not add binary
		4	96											value to product		
		5	82											If no carry		
		6	19		D	A	D	D						Add binary value to		
		7	C2		J	N	Z	8	2	9	6				product and loop	
		8	96											unless decimal		
		9	82											remainder zero		
INTEGRATED COMPUTER SYSTEMS	A			C9		R	E	T							Returns if decimal	
	B														remainder zero	
	C															
	D					E	N	T	E	R	W	I	T	H		
	E					(L)	=	B	I	N	A	R	Y	V	A	L
	F					(H)	=	D	E	C	I	M	A	L	F	
	8	0														
			1				R	E	T	U	R	N	W	I	T	H
			2				(H	L)	=	T	W	O	B	Y	T	E
			3													
		4				U	S	E	S	A	L	L	R	E	G	
		5				E	X	C	E	P	T	B				
		6														
		7														
		8														

Figure 4-7e

4.2.2 Variable Cycle Time

OPTIONAL EXERCISE

Postulate an open loop control system for the heating of a chemical reactor, in which the heater duty cycle is set according to the volume of material being cooked. As in the preceding exercise, the required duty cycle is an input to the computer. Because a gas fired heater is used, and ignition is not instantaneous, the minimum useful on-time is 20 seconds; fuel efficiency is enhanced by longer on-times. When the batch is small, however, a heater off-time exceeding 180 seconds may result in excessive cooling between heat cycles, and shorter off-time is preferable. The chemical engineer has provided this table of on time vs. duty cycle; interpolation is to be used between these values.

Duty Cycle	On Time	Off Time	Cycle Time
.10	20	180	200
.20	20	80	100
.30	30	70	100
.40	40	60	100
.50	60	60	120
.60	90	60	150
.70	140	60	200
.80	160	40	200
.90	180	20	200
1.00	200	0	200

DIGITAL TO ANALOG OUTPUT

Design a program that will vary the on-time and off-time according to this table, with any reasonable interpolation scheme. To make a convenient display during program debugging, divide the times by 10.

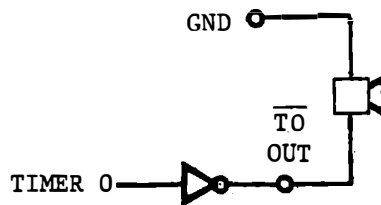
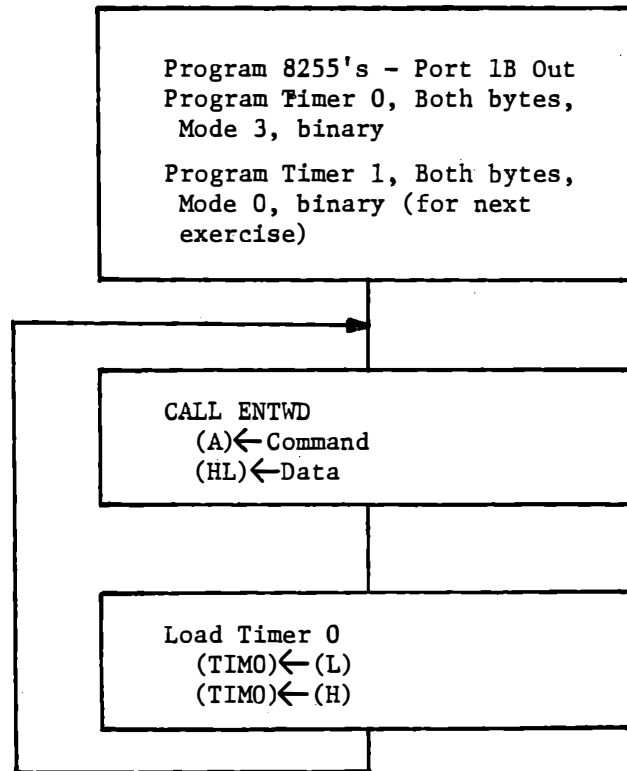
4.3 FREQUENCY CONTROL

Frequency is another analog that requires only a single output bit with time as the continuous variable. Varying the frequency of an output signal can control an induction or synchronous AC motor, can control the delivery or flow of a fluid, or can test frequency dependent hardware.

Generation of a fixed frequency signal is automatically accomplished by the 8253 in mode 3, the square wave generator, as we demonstrated in Section 3.9.1. The frequency can be varied by loading different time intervals to the timer. Some applications of variable frequency control cannot tolerate the high harmonic content of a square wave, and some form of multi-bit output is required to create a more nearly sinusoidal signal. We will experiment with this in Section 4.7.

The following exercise creates audio tones with the square wave generator thus demonstrating a technique for frequency generation. The most common use of tone generation is for frequency modulated data communication and recording, which is used in the tape cassette interface included on the computer. In the next exercise we will modify the previous program to vary the frequency and to demonstrate frequency modulation. The final exercise in this section will create music from the square wave generator. This is more of a toy than part of a useful system, but it uses important programming techniques such as bit manipulation and table look-up, as well as frequency modulation.

DIGITAL TO ANALOG OUTPUT



Audio Output Program and Circuit

Figure 4-8

4.3.1

Audio Tone Generator

Write a program to load timer 0 with data entered through the keyboard. Connect the loudspeaker as shown in Figure 4-8 to output the tone generated by timer 0. The timer is to operate in square wave mode.

Monitor subroutine ENTWD (0346) will accept two data bytes and a command, returning the data in register pair HL and the command in register A. Load the data (less significant byte first) into timer 0. If you enter data from the list in Figure 4-9 the tone will be a defined musical note. If you have perfect pitch or a standard for comparison such as a tuning fork or a high quality audio oscillator you may detect that the tone is imperfect. This stems from error in the computer's crystal clock and from rounding error in the calculation of $\text{period} = 1/\text{frequency}$. (The latter is only significant at the very high frequencies). Try keying in the data for various notes and listen to the tone.

DIGITAL TO ANALOG OUTPUT

Musical Note	Frequency (Hertz)	Timer Period (Hex Count)
C (below Middle C)	130.81	3D28
C#	138.59	39B9
D	146.83	367C
D#	155.56	336D
E	164.81	308A
F	174.61	2DD1
F#	185.00	2B3E
G	196.00	28D1
G#	207.65	2687
A	220.00	245D
A#	233.08	2253
B	246.94	2065
C (Middle C)	261.63	1E94
C#	277.18	1CDD
D	293.66	1B3E
D#	311.13	19B7
E	329.63	1845
F	349.23	16E8
F#	369.99	159F
G	392.00	1468
G#	415.30	1343
A A440	440.00	122F
A#	466.16	1129
B	493.88	1033
C (above Middle C)	523.25	0F4A
C#	554.37	0E6E
D	587.33	0D9F
D#	622.25	0CDB
E	659.26	0C23
F	698.46	0B74
F#	739.99	0AD0
G	783.99	0A34
G#	830.61	09A2
A	880.00	0917
A#	932.33	0895
B	987.77	0819
C C''	1046.50	07A5
C#	1108.73	0737
D	1174.66	06CF
D#	1244.51	066E
E	1318.51	0611
F	1396.91	05BA
F#	1479.98	0568
G	1567.98	051A
G#	1661.22	04D1
A	1760.00	048C
A#	1864.66	044A
B	1975.53	040D

List of Concert Pitch Musical Tones

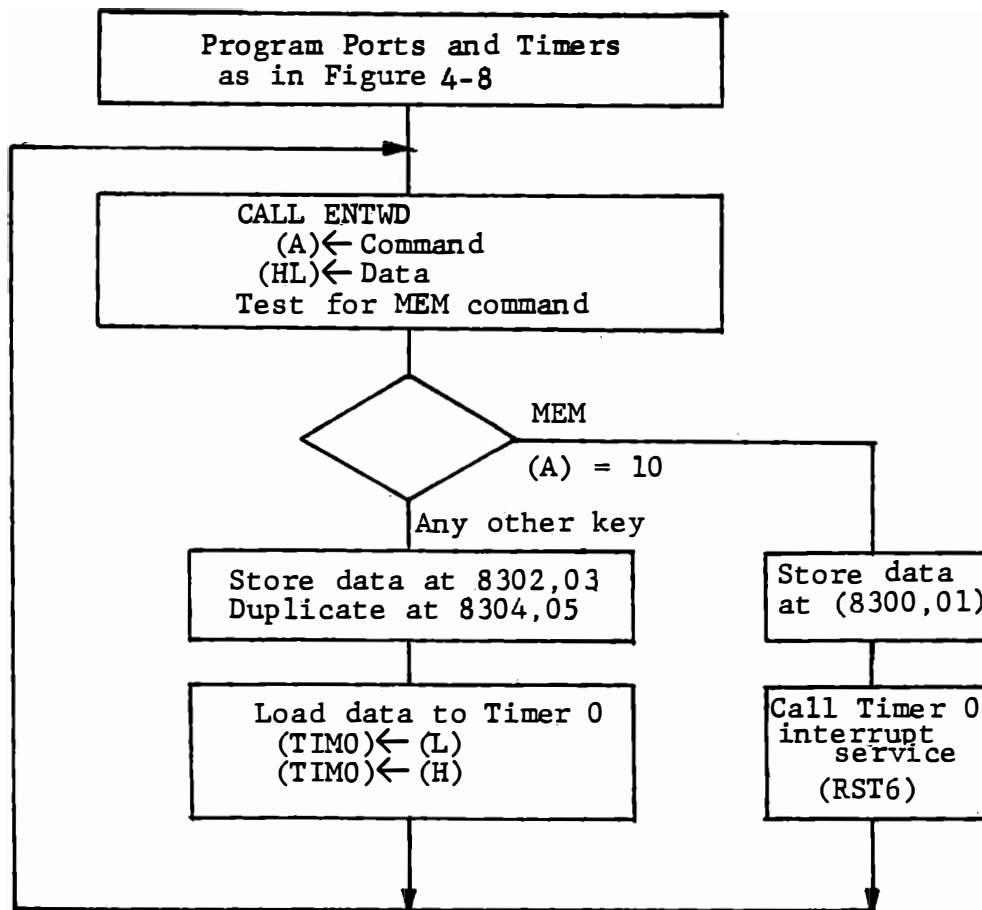
Figure 4-9

4.3.2 Frequency Modulation Program

Modify the tone generator program to generate a tone whose frequency increases with time. Data loaded with the MEM command will control the rate of change by setting an interrupt period in timer 1. Data loaded with any other command key will provide the initial frequency. At each interrupt the period loaded to timer 0 will be reduced by one count to increase its frequency. After the frequency of 8000 HZ (period = 0100 hex) has been generated the original frequency will be reloaded. Figure 4-10 shows the main program and Figure 4-11 shows timer 1 interrupt service.

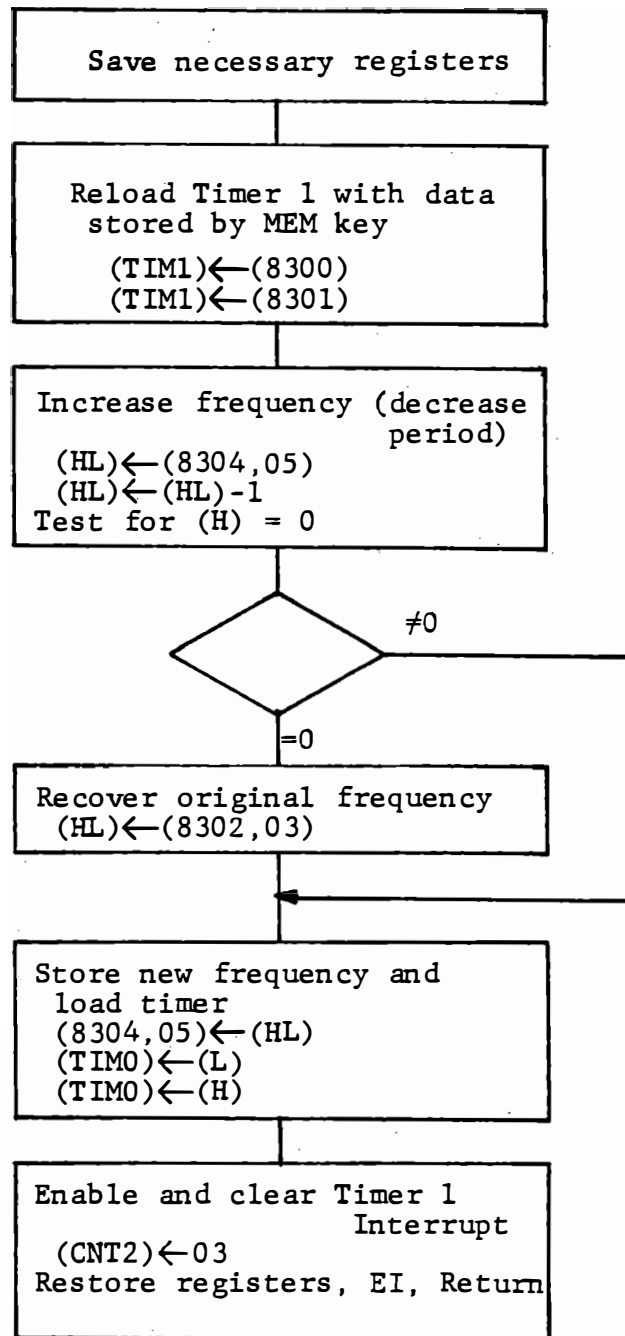
To use the program enter a period from the tone table of Figure 4-9 using any command except MEM. The tone will be steady at first, since timer 1 is not running. Now enter an interval to timer 1, using the MEM key. The tone will slide from the initial frequency up to 8000 HZ, and then repeat.

DIGITAL TO ANALOG OUTPUT



Tone Generator - Main Program





Figure 4-10



Tone Generator Interrupt Service

Figure 4-11

DIGITAL TO ANALOG OUTPUT

				
C (Below Middle C)	00	80	40	20
C# (D♭)	01	81	41	21
D	02	82	42	22
D# (E♭)	03	83	43	23
E	04	84	44	24
F	05	85	45	25
F# (G♭)	06	86	46	26
G	07	87	47	27
G# (A♭)	08	88	48	28
A	09	89	49	29
A# (B♭)	0A	8A	4A	2A
B	0B	8B	4B	2B
C (Middle C)	0C	8C	4C	2C
C# (D♭)	0D	8D	4D	2D
D	0E	8E	4E	2E
D# (E♭)	0F	8F	4F	2F
E	10	90	50	30
F	11	91	51	31
F# (G♭)	12	92	52	32
G	13	93	53	33
G# (A♭)	14	94	54	34
A	15	95	55	35
A# (B♭)	16	96	56	36
B	17	97	57	37
C (Above Middle C)	18	98	58	38
C# (D♭)	19	99	59	39
D	1A	9A	5A	3A
D# (E♭)	1B	9B	5B	3B
E	1C	9C	5C	3C
F	1D	9D	5D	3D
F# (G♭)	1E	9E	5E	3E
Rest	1F	9F	5F	3F

Codes for Musical Notes

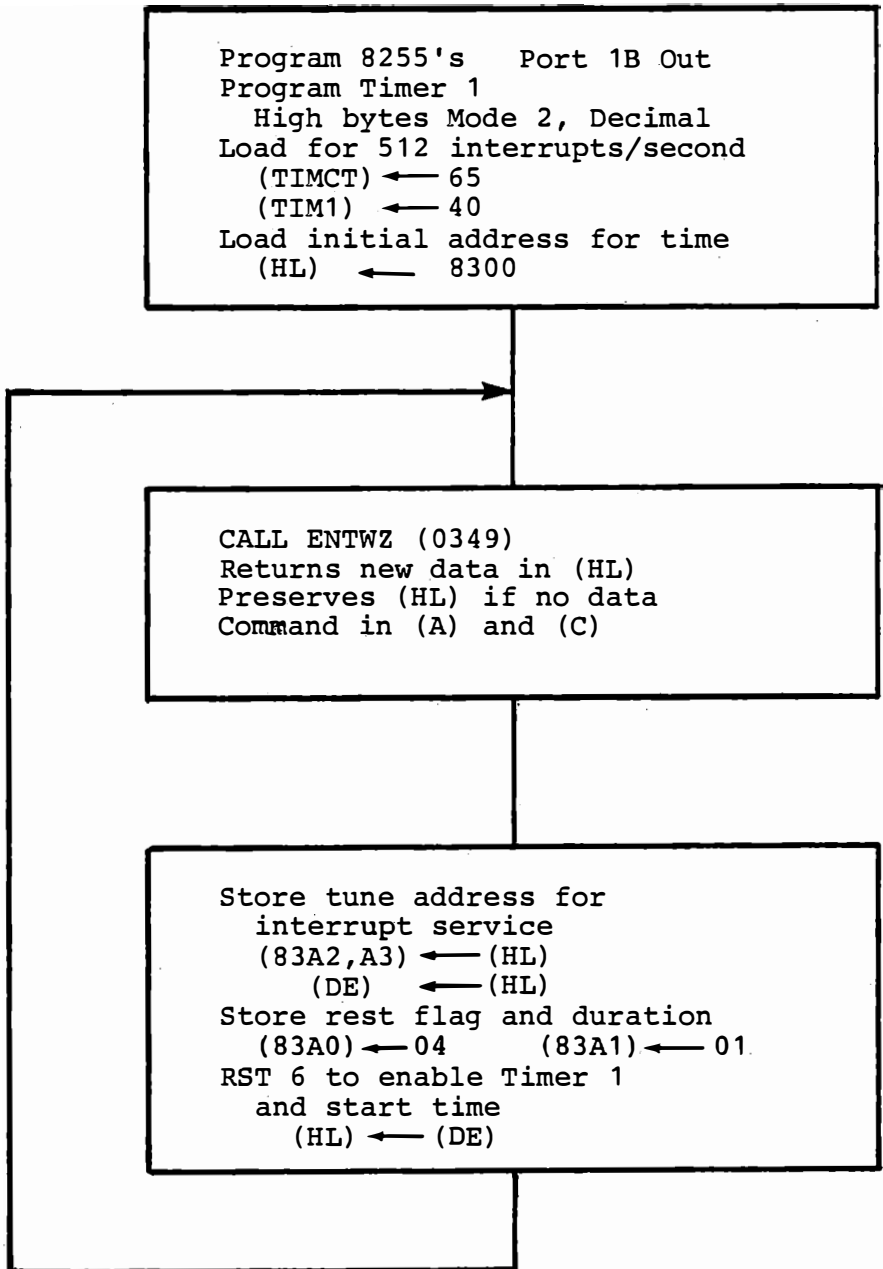
Figure 4-12

4.3.3 Recorded Music Player

This program module reads music - in the form of notes in a list. Each note in the tune includes three bits to indicate duration of the note (eighth, quarter, half and whole notes) and five bits to select one of 30 tones starting at C below middle C and covering two and one-half octaves. (See Figure 4-12). For each note it performs a table lookup on the five low order bits to find a time interval corresponding to the tone frequency, and outputs that time to timer 0 operating as a square wave generator. The inverted output of timer 0 drives a loudspeaker, as shown in Figure 4-8.

Timer 0 runs in mode 3; its interrupt is disabled, and its only function is to generate the square wave. Timer 1 runs in mode 2 to give a repetitive timing interrupt which is used to count down a software counter, loaded from the high three bits of the note, to set the note duration.

DIGITAL TO ANALOG OUTPUT



TUNE - MAIN PROGRAM

FIGURE 4-13

The main program (Figure 4-13) initializes the ports and Timer 1, and loads a fixed address (8300) where a tune is stored. It calls ENTWZ to permit entry of different addresses where other tunes can be loaded. This monitor subroutine preserves (HL) if no hex keys are entered. This address and a flag and counter are stored for use by the interrupt service routine.

Timer 1 is programmed during initialization to mode 2, decimal, and loaded with 80 in its high byte to interrupt 256 times per second. This value makes a whole note of one second. Faster tempo can be obtained by loading smaller values to timer 1. (You may want to elaborate the program to accept a value from the keyboard.)

DIGITAL TO ANALOG OUTPUT

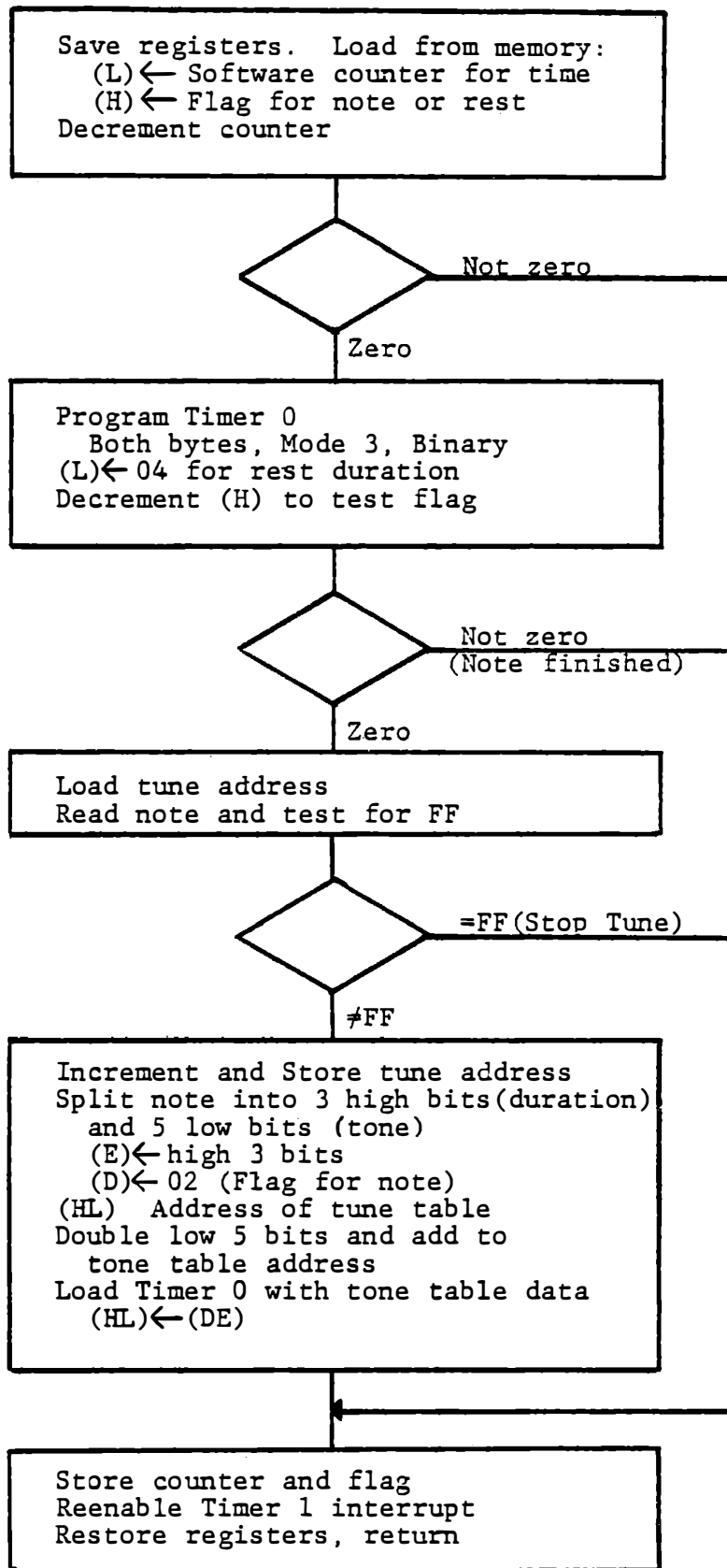


FIGURE 4-14

Interrupt service for timer 1 (figure 4-14) decrements a software counter (83A0) and exits if not zero.

At zero it reprograms timer 0 (to mode 3, binary) which stops counting until the timer is reloaded. A flag at 83A1 is decremented, and if it goes from 02 to 01 a rest of 1/64 note (4 counts) is generated to separate one musical note from the next. The next time that the software counter (83A0) reaches zero the flag will count down from 01 to 00, and a new note is played. The tune address is loaded from memory (83A2, A3) and the next note is read. If this note is FF the tune is finished and an exit is made without loading timer 0. For any other note the tune address is incremented and stored, and the note is split into three high bits for duration and five low bits for tone. The high bits are entered to the software counter, and the flag is set to 02 to indicate a note is being played. Now the five low bits are tested for code 1F which indicates a rest in the music. For any other code, 00 to 1E, the tone table is addressed to find the time interval for the corresponding note. The tone table data are copied to timer 0, which now generates a square wave of the required frequency.

DIGITAL TO ANALOG OUTPUT

Figure 4-15 is a complete program with a tone look-up table (Figure 4-16) and a few tunes (Figure 4-17). The tone table covers four octaves, more than can be addressed by the five bits allotted for selecting a note. You can change the table address to obtain a different set of notes. This also permits transposing a tune from one major key to another, simply by entering a different address for the table. You may want to provide keyboard entry for this, also.

TUNE - MAIN PROGRAM

A D D R		CODE				
CODING SHEET	8	20	0	3E	MVI A, 80	Program Ports
			1	80		
			2	D3	OUT CNT1	
			3	07		
			4	3E	MVI A, 92	
			5	92		
			6	D3	OUT CNT2	
			7	0F		
MICROCOMPUTER TRAINING SYSTEM	8			3E	MVI A, 65	Program Timer 1
			9	65		High byte
			A	D3	OUT TIMCT	Mode 2
			B	17		Decimal
			C	3E	MVI A, 80	256 interrupts
			D	80		per second
			E	D3	OUT TIM1	
			F	15		
INTEGRATED COMPUTER SYSTEMS	8	21	0	21	LXI H, 8300	Initial address
			1	00		for Tune
			2	83		
			3	CD	CALL ENTWZ	Preserve HL
			4	49		if no data
			5	03		entered
			6	22	SHLD 83A2	store tune address
			7	A2		for interrupt service
INTEGRATED COMPUTER SYSTEMS	8			83		
			9	EB	XCHG	
			A	21	LXI H, 0104	
			B	04		(L) ← 1/4 note
			C	01		(H) ← flag for rest
			D	22	SHLD 83AD	store
			E	A0		
			F	83		
INTEGRATED COMPUTER SYSTEMS	8	22	0	EB	XCHG	(HL) ← starting address
			1	F7	RST 6	start interrupts
			2	C3	JMP 8213	back to address
			3	13		input
			4	82		
			5			
			6			
			7			
		8				

Figure 4-15a

TUNE - INTERRUPT SERVICE

A D D R		CODE							
CODING SHEET	8	23	0	F5		PUSH	H	PSW	
			1	E5		PUSH	H		
			2	D5		PUSH	H	D	
			3	C5		PUSH	H	B	
			4	2A		LHLD		83A0	
			5	A0					(L) ← counter
			6	83					(H) ← flag
			7	2D		DCR	L		Decrement counter
			8	C2		JNZ		EXIT	Exit if not zero
			9	70					
MICROCOMPUTER TRAINING SYSTEM	A			82					
	B			3E		MVI	A, 36		Program timer 0
	C			36					Both bytes
	D			D3		OUT	TIMCT		Mode 3
	E			17					Binary
	F			2E		MVI	L, 04		(L) ← 1/4 note
	8	24	0	04					to count for rest
			1	25		DCR	H		If flag was 02
			2	C2		JNZ		EXIT	go to exit for rest
			3	70					
INTEGRATED COMPUTER SYSTEMS			4	82					
			5	2A		LHLD		83A2	(HL) ← tune address
			6	A2					
			7	83					
			8	7E		MOV	A, M		(A) ← note
			9	FE		CPI	FF		Test for end of
		A		FF					tune
		B		CA		JZ		EXIT	Jump to rest if end
		C		70					
		D		82					
	E		E6		ANI	E0		(A) ← 3 high bytes	
	F		E0						
	8		0						
			1						
			2						
			3						
			4						
			5						
			6						
			7						
			8						

Figure 4-15b

TUNE - INTERRUPT SERVICE (continued)

A D D R		CODE							
CODING SHEET	8	25	0	5F		M O V	E, A		(E) ← high 3 bits
			1	AE		X R A	M		(A) ← low 5 bits
			2	23		I N X	H		Next tune address
			3	22		S H L D		83A2	Store address
			4	A2					
			5	83					
			6	16		M V I	D, 02		Set flag for note
			7	02					
			8	FE		C P I	1F		Test for rest
			9	1F					in tune
MICROCOMPUTER TRAINING SYSTEM		A	CA		J Z		826A		Do not load timer
		B	6A						if rest code
		C	82						
		D	21		L X I	H, 82A0			Address tone table
		E	A0						
		F	82						
		8	26	0	87		A D D	A	Double low 5 bits
				1	85		A D D	L	Add to table
				2	6F		M O V	L, A	address
				3	7E		M O V	A, M)
			4	D3		O U T	T I M 0	{(Timer 0) ← low	
			5	14				} byte at interval	
			6	23		I N X	H)	
			7	7E		M O V	A, M	{(Timer 0) ← high	
			8	D3		O U T	T I M 0	} bytes of interval	
			9	14					
INTEGRATED COMPUTER SYSTEMS		A	EB		X C H G				(H) ← flag
		B	00		N O P				(L) ← count
		C	00						
		D	00						
		E	00						
		F	00						
		8	0						
			1						
			2						
			3						
		4							
		5							
		6							
		7							
		8							

Figure 4-15c

TUNE - INTERRUPT SERVICE EXIT

		A	D	D	R	CODE													
CODING SHEET	8	27	0	22		S	A	L	D	8	3	A	0					Store counter (L) and flag (H)	
			1	A0															
			2	83															
			3	3E			M	V	I		A	0	3						
			4	03															
			5	D3			O	U	T		C	N	T	2					
			6	0F															
			7	C1			P	O	P		B								
			8	D1			P	O	P		D								
			9	E1			P	O	P		H								
		A	F1				P	O	P		P	S	W						
		B	F B				E	I											
		C	C9				R	E	T										
	D																		
	E																		
	F																		
MICROCOMPUTER TRAINING SYSTEM	8	0																	
		1																	
		2																	
		3																	
		4																	
		5																	
		6																	
		7																	
		8																	
		9																	
		A																	
		B																	
		C																	
		D																	
		E																	
		F																	
INTEGRATED COMPUTER SYSTEMS	8	0																	
		1																	
		2																	
		3																	
		4																	
		5																	
		6																	
		7																	
	8																		

Figure 4-15d

		CLOCKS			FREQUENCY							
A D D R		CODE										
CODING SHEET	8	2C	0	45	E	4	=	329	.	63		
			1	18								
			2	E8	F	4	=	349	.	23		
			3	16								
			4	9F	F#	4	=	369	.	99		
			5	15								
			6	68	G	4	=	392	.	00		
			7	14								
			8	43	G#	4	=	415	.	30		
			9	13								
MICROCOMPUTER TRAINING SYSTEM		A	2F	A	4	=	440	.	00			
			B	12								
			C	29	A#	4	=	466	.	16		
			D	11								
			E	33	B	4	=	493	.	88		
			F	10								
		8	2D	0	4A	C	5	=	523	.	25	C ABOVE MIDDLE C
				1	0F							
				2	6E	C#	5	=	554	.	37	
				3	0E							
INTEGRATED COMPUTER SYSTEMS			4	9F	D	5	=	587	.	33		
			5	0D								
			6	DB	D#	5	=	622	.	25		
			7	0C								
			8	23	E	5	=	659	.	26		
			9	0C								
			A	74	F	5	=	698	.	46		
			B	0B								
			C	DO	F#	5	=	739	.	99		
			D	0A								
		E	34	G	5	=	783	.	99			
		F	0A									
	8	0			(CONTINUED)							
		1										
		2										
		3										
		4										
		5										
		6										
		7										
		8										

Figure 4-16b

		CLOCKS		FREQUENCY				
A	D	D	R	CODE				
CODING SHEET	8	2E	0	A	2	G # 5 =	830.61	
			1	0	9			
			2	1	7	A 5 =	880.00	
			3	0	9			
			4	9	5	A # 5 =	932.33	
			5	0	8			
			6	1	9	B 5 =	987.77	
			7	0	8			
			8	A	5	C 6 =	1046.50	
			9	0	7			
MICROCOMPUTER TRAINING SYSTEM	A			3	7	C # 6 =	1108.73	
	B			0	7			
	C			C	F	D 6 =	1174.66	
	D			0	6			
	E			6	E	D # 6 =	1244.51	
	F			0	6			
	8	2F	0	1	1	E 6 =	1318.51	
				1	0	6		
				2	B	A	F 6 =	1396.91
				3	0	5		
INTEGRATED COMPUTER SYSTEMS				6	8	F # 6 =	1479.98	
				5	0	5		
				6	1	A	G 6 =	1567.98
				7	0	5		
				8	D	1	G # 6 =	1661.22
				9	0	4		
				A	8	C	A 7 =	1760.00
				B	0	4		
				C	4	A	A # 7 =	1864.66
				D	0	4		
			E	0	D	B 7 =	1975.53	
			F	0	4			
	8		0					
			1					
			2					
			3					
			4					
			5					
			6					
			7					
			8					

Figure 4-16c

HOME ON THE RANGE

		A	D	D	R	CODE										TONE	DURATION			
CODING SHEET	8	3	0	0	4	C												C	1/4	0
				1	4	C												C	1/4	GIVE
				2	5	1												F	1/4	ME
				3	5	3												G	1/4	A
				4	9	5												A	1/2	HOME
				5	3	1												F	1/8	WHERE
				6	3	0												E	1/8	THE
				7	6	E												D	3/8	RUFF-
				8	3	6												Bb	1/8	A-
				9	5	6												Bb	1/4	LD
MICROCOMPUTER TRAINING SYSTEM	A				9	6												Bb	1/2	ROAM
	B				3	5												A	1/8	WHERE
	C				3	6												Bb	1/8	THE
	D				9	8												C	1/2	DEFER
	E				3	1												F	1/8	AND
	F				3	1												F	1/8	THE
	8	0			5	1												F	1/4	ANT-
		1			5	0												E	1/4	E-
		2			5	1												F	1/4	LOPF
		3			1	3												G	whole	PLAY
INTEGRATED COMPUTER SYSTEMS				4	4	C												C	1/4	WHERE
				5	4	C												C	1/4	SEL-
				6	5	1												F	1/4	DOM
				7	5	3												G	1/4	IS
				8	9	5												A	1/2	HEARD
				9	3	1												F	1/8	A
		A			3	0												E	1/8	DIS-
		B			6	E												D	3/8	COUR-
		C			3	6												Bb	1/8	AG-
		D			5	6												Bb	1/4	ING
				E	9	6												Bb	1/2	WORD
				F	3	6												Bb	1/8	AND
	8	0			3	6												Bb	1/8	THE
		1			7	5												A	3/8	SKIES
		2			3	3												G	1/8	ARE
		3			5	1												F	1/4	NOT
		4			7	0												E	3/8	CLOUD-
		5			3	1												F	1/8	Y
	6			5	3												G	1/4	ALL	
	7			1	1												F	whole	DAY	
	8			F	F													END OF TUNE		

Figure 4-17a

THE DRUNKEN SAILOR

	A	D	D	R	CODE																
CODING SHEET	8	33	0		55													A	1/4	WHAT	
			1		35													A	1/2	SHALL	
			2		35													A	1/8	WE	
			3		55													A	1/4	DO	
			4		35													A	1/8	WITH	
			5		35													A	1/2	A	
			6		55													A	1/4	DRUNK	
			7		4E													D	1/4	EN	
			8		51													F	1/4	SAIL-	
			9		55													A	1/4	OR	
MICROCOMPUTER TRAINING SYSTEM	A				53													G	1/4	WHAT	
	B				33													G	1/2	SHALL	
	C				33													G	1/8	WE	
	D				53													G	1/4	DO	
	E				33													G	1/2	WITH	
	F				33													G	1/8	A	
	8		0		53													G	1/4	DRUNK-	
			1		4C													C	1/4	EN	
			2		50													E	1/4	SAIL-	
			3		53													G	1/4	OR	
INTEGRATED COMPUTER SYSTEMS			4		55													A	1/4	WHAT	
			5		35													A	1/8	SHALL	
			6		35													A	1/8	WE	
			7		55													A	1/4	DO	
			8		35													A	1/8	WITH	
			9		35													A	1/2	A	
		A			55													A	1/4	DRUNK-	
		B			57													B	1/4	EN	
		C			58													C	1/4	SAIL-	
		D			5A													D	1/4	OR	
		E			58													C	1/4	EARL-	
		F			55													A	1/4	Y	
	8		0		53													G	1/4	IN	
			1		50													E	1/4	THE	
			2		8E													D	1/2	MORN-	
			3		8E													D	1/2	ING	
			4		FF																END OF TUNE
			5																		
		6																			
		7																			
		8																			

Figure 4-17b

DIGITAL TO ANALOG OUTPUT

4.3.4 Music Recording Program

OPTIONAL EXERCISE

Develop a program that will play notes entered from the keyboard, recording the tune as it is played. Define the hexadecimal keys to represent sixteen notes in the key of C.

C	D	E	F
F	G	A	B
B	C	D	E
E	F	G	A

Define the command keys for playing and editing the music:

ADDR	Load a starting address (optionally) followed by a four digit address. Otherwise use the last address entered.
BRK	Set a musical key (to be followed by a note, or by sharp or flat and a note).
MEM	Sharp (to precede a note)
REG	Flat (to precede a note)
CLR	Delete (from the recorded tune) the last note played, replacing it with the stop code (FF).
RUN	Play the tune from the beginning to the end, and wait for a new note to be added to the tune.
STEP	Play the next note recorded (if any) and wait for a new note to be added.
NEXT	Enter a rest in the tune.

To record a tune the musician will enter:

ADDR	XXXX	to locate the tune
REG	D	(for example) to set the key of D flat.
CLR		to delete any note already recorded at that location, and prepare to replace it.

Now enter the successive notes. The program must transpose the note of the selected musical key into a note in the chromatic scale, play that note while the hexadecimal key is held down, and measure the duration of the note. When the key is released, generate and store the code for the tune and duration.

To end the recording enter any note or a rest (NEXT) and press CLR to replace it with the stop code.

To play the tune back, press ADDR, RUN.

To edit the tune, press ADDR, STEP, STEP, STEP etc. to play one note at a time. Replace any desired note by CLR and the desired note.

Development of this program is left as an exercise for the student. The relationship between the musical keys (C, D flat, etc.) and the meaning of the hex keys is shown in Figure 4-18. The hex code given for each note is the whole note code shown in Figure 4-12.

DIGITAL TO ANALOG OUTPUT

Hex Key	C	C#	D	D#	E	F
0	E 04	D# 03	E 04	D# 03	E 04	E 04
1	F 05	F 05	F# 06	F 05	F# 06	F 05
2	G 07	F# 06	G 07	G 07	G# 08	G 07
3	A 09	G# 08	A 09	G# 08	A 09	A 09
4	B 0B	A# 0A	B 0B	A# 0A	B 0B	A# 0A
5	C 0C	C 0C	C 0C	C 0C	C# 0D	C 0C
6	D 0E	C# 0D	D 0E	D 0E	D# 0F	D 0E
7	E 10	D# 0F	E 10	D# 0F	E 10	E 10
8	F 11	F 11	F# 12	F 11	F# 12	F 11
9	G 13	F# 12	G 13	G 13	G# 14	G 13
A	A 15	G# 14	A 15	G# 14	A 15	A 15
B	B 17	A# 16	B 17	A# 16	B 17	A# 16
C	C 18	C 18	C 18	C 18	C# 19	C 18
D	D 1A	C# 19	D 1A	D 1A	D# 1B	D 1A
E	E 1C	D# 1B	E 1C	D# 1B	E 1C	C 1C
F	F 1D	F 1D	F# 1E	F 1D	F# 1E	F 1D

Hex Key	F#	G	G#	A	A#	B
0	F 05	E 04	E 04	E 04	D# 03	E 04
1	F# 06	F# 06	F 05	F# 06	F 05	F# 06
2	G# 08	G 07	G 07	G# 08	G 07	G# 08
3	A# 0A	A 09	G# 08	A 09	A 09	A# 0A
4	B 0B	B 0B	A# 0A	B 0B	A# 0A	B 0B
5	C# 0D	C 0C	C 0C	C# 0D	C 0C	C# 0D
6	D# 0F	D 0E	C# 0D	D 0E	D 0E	D# 0F
7	F 11	E 10	D# 0F	E 10	D# 0F	E 10
8	F# 12	F# 12	F 11	F# 12	F 11	F# 12
9	G# 14	G 13	G 13	G# 14	G 13	G# 14
A	A# 16	A 15	G# 14	A 15	A 15	A# 16
B	B 17	B 17	A# 16	B 17	A# 16	B 17
C	C# 19	C 18	C 18	C# 19	C 18	C# 19
D	D# 1B	D 1A	C# 19	D 1A	D 1A	D# 1B
E	F 1D	E 1C	D# 1B	E 1C	D# 1B	E 1C
F	F# 1E	F# 1E	F 1D	F# 1E	F 1D	F 1D

Music Recording Program, Hex Key Chart

4.4 MULTI-BIT OUTPUT

A multi-bit output can represent a continuous variable to any desired precision. The output is usually in the form of a binary number with each bit having a weighted value (e.g. 1,2,4,8,16---); this must be converted to a voltage or current by external hardware. Section 4.5 deals with this procedure. Another possibility, occasionally used as a display device, is to illuminate an LED as a pointer. A prototype automobile speedometer has been shown with an LED at each mile per hour position; here all of the lower values are illuminated, up to and including the actual speed. We will modify the tune program of Section 4.3.3 to display the tone in this fashion, using the LED's of port 1A.

In the program of Figure 4-15 the interrupt service obtains a note to be played and makes a conditional jump if the note is a rest. This is a good place to insert a patch to display the tone. At 825A replace JZ 826A by JMP 8280, where we will place the patch.

DIGITAL TO ANALOG OUTPUT

We will display the notes as follows:

NOTES	CODES	DISPLAY
C,C#,D,D#	00,01,02,03	00000001
E,F,F#,G	04,05,06,07	00000011
G#,A,A#,B	08,09,0A,0B	00000111

Octave of middle C

C,C#,D,D#	0C,0D,0E,0F	00001111
E,F,F#,G	10,11,12,13	00011111
G#,A,A#,B	14,15,16,17	00111111

Octave above middle C

C,C#,D,D#	18,19,1A,1B	01111111
E,F,F#	1C,1D,1E	11111111
Rest		00000000

DIGITAL TO ANALOG OUTPUT

The patch must save the note and the flag, and it must include the JZ 826A instruction that was replaced. We can obtain the desired display by masking unwanted bits and shifting, so that the codes 00,01,02,03 are transformed to 00, and 04,05,06,07 are transformed to 01, etc. Then increment the result so that the values range from 01, to 08. Now this procedure will shift from one to eight 1's into register H:

```
                LXI    H,00FF
LOOP           DAD    H
                DCR    A
                JNZ   LOOP
```

Register H can be displayed in port 1A. Figure 4-19 shows the patch. For many tunes it may be more interesting to mask for the three low bits and omit the shifting, so each note within a small range will be displayed differently.

PATCH TO DISPLAY TONE

		A	D	D	R	CODE				
CODING SHEET	8 28	0	F5			PUSH	PSW			Save note
		1	21			LXI	H, 00FF			Clear H
		2	FF							Load L with 1's
		3	00							to shift into H
		4	CA			JZ		8 291		Zero flag is set
		5	91							for rest -
		6	82							jump to display 00
		7	E6			ANI	1C			(Mask) for
		8	1C							bits 2, 3, 4
		9	1F			RAR				Shift to low bits
MICROCOMPUTER TRAINING SYSTEM	A		1F			RAR				
	B		3C			INR	A			(A) from 01 to 08
	828	C	29			DAD	H			Shift 1 into (H)
		D	3D			DCR	A			until (A) = 0
		E	C2			JNZ		8 28C		
		F	8C							
	8 29	0	82							
	829	1	7C			MOV	A, H			Display shifted 1's
		2	D3			OUT	PORT 1A			in LED's
		3	04							
INTEGRATED COMPUTER SYSTEMS		4	F1			POP	PSW			
		5	CA			JZ		8 26A		Exit if Rest
		6	6A							(replace patched
		7	82							instruction)
		8	C3							
		9	5D							
		A	82							
		B								
		C								
		D								
	E									
	F									
						PATCH	AT	8 25A		
	825	A	C3			JMP	8 280			Replaced
		B	80							JZ 826A
		C	82							

Figure 4-19

4.5 ANALOG VOLTAGE GENERATION

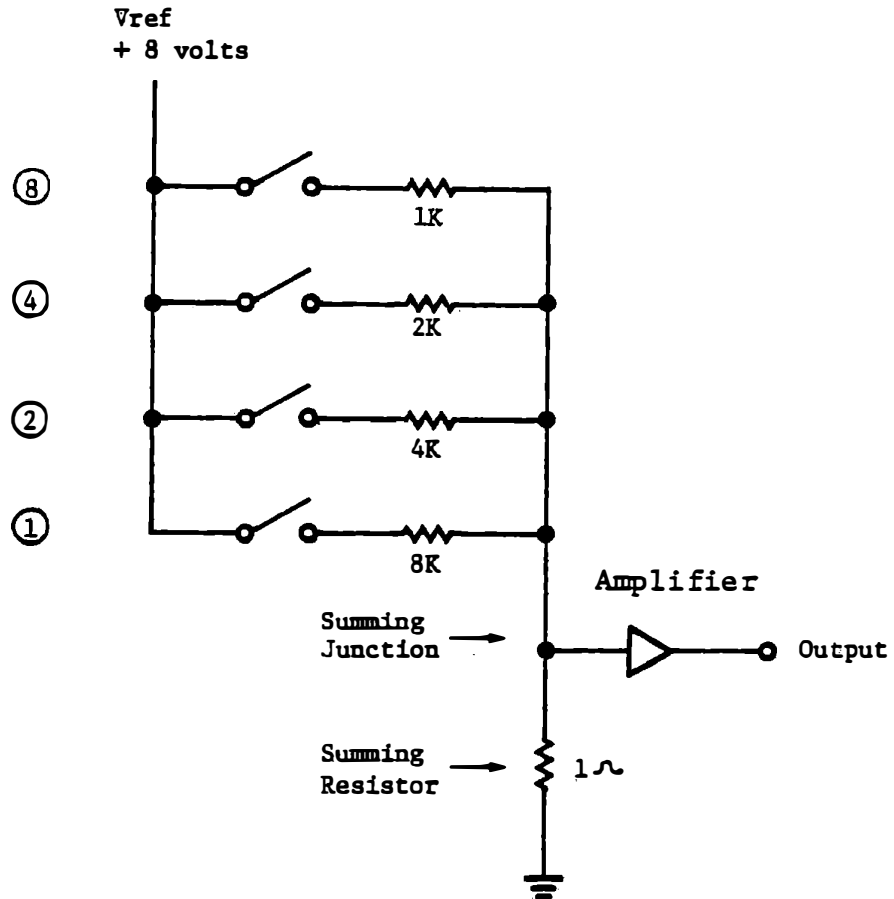
Probably the most common analog signal is a variable voltage. The remainder of this chapter and most of Chapter 5 are concerned with variable voltage signals and their interface with the computer.

Clearly a variable voltage can be generated by a pulse width modulated signal integrated by resistors and capacitors; we will use such a generator in Chapter 5. Other schemes involve multi-bit output.

4.5.1 Binary Summing Circuit

Consider the network shown in Figure 4-20.

DIGITAL TO ANALOG OUTPUT



Note: Resistance and reference voltage shown are selected for convenience of discussion; they are not typical values.

Binary Summing Circuit

Figure 4-20

DIGITAL TO ANALOG OUTPUT

Each of the switches, labelled 1,2,4, and 8, represents a contact closure or a transistor switch operated by one bit of the computer output. If switch 8 is closed, a current of 8 milliamperes flows through the 1K resistor and generates an 8 millivolt signal across the 1 ohm summing resistor. If switch 4 is also closed, a current of 4 milliamperes flows through the 2K resistor; the two currents are summed to generate 12 millivolts at the summing junction. Thus any combination of the four switches generates an analog voltage proportional to the binary output, as shown in Figure 4-21. The output amplifier generates a more useful signal level.

Bit Value	Resistor	Current	
1	8K	1 ma	
2	4K	2 ma	
4	2K	4 ma	
8	1K	8 ma	

Binary Value	Parallel Resistance	Current	Voltage
0000		0	0.0
0001	8000	1	0.001
0010	4000	2	0.002
0011	2667	3	0.003
0100	2000	4	0.004
0101	1600	5	0.005
0110	1333	6	0.006
0111	1143	7	0.007
1000	1000	8	0.008
1001	889	9	0.009
1010	800	10	0.010
1011	727	11	0.011
1100	667	12	0.012
1101	585	13	0.013
1110	571	14	0.014
1111	533	15	0.015

Numerical Values for Circuit of 4-12

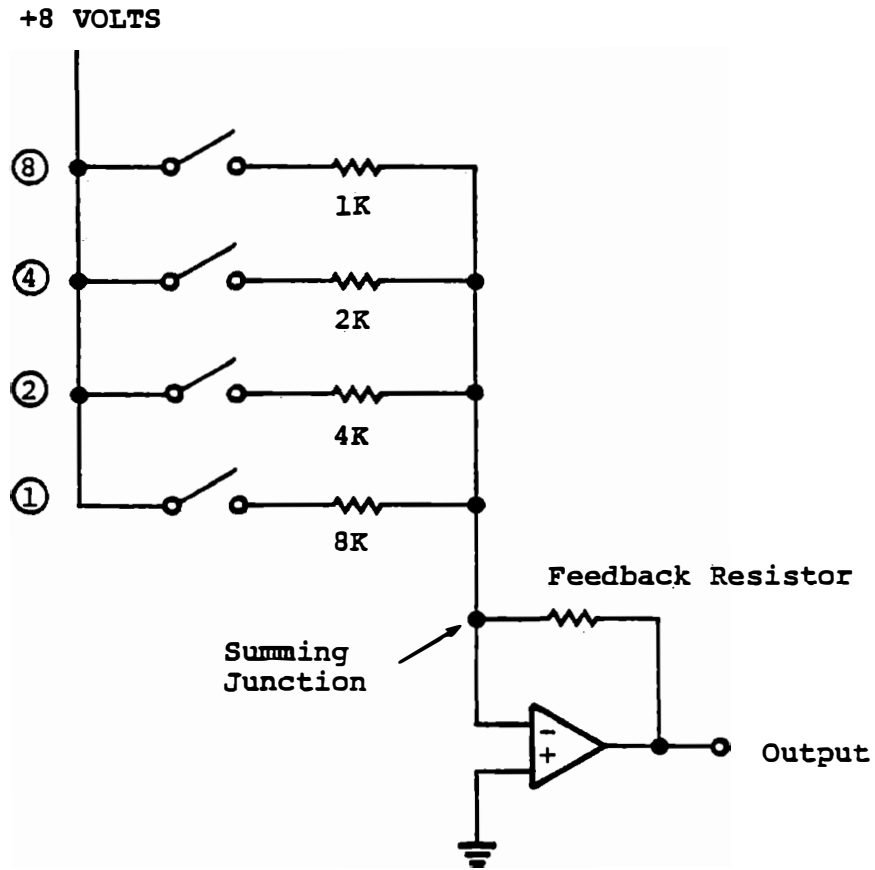
Figure 4-21

DIGITAL TO ANALOG OUTPUT

The accuracy of this device is limited by the influence of the voltage at the summing resistor; as more bits are used in the conversion, this becomes more significant, but is largely overcome by the use of an operational amplifier, as shown in Figure 4-22. With this connection, the op-amp output is inverted from the input signal; the current from the resistor network actually flows to the op-amp output through the feedback resistor, and the voltage at the summing junction is held very close to ground, so that the crosstalk between bits (i.e. the influence of one bit on the signal generated by another) is very small. For a detailed discussion of operational amplifiers, the student is referred to Wait, Huelsman and Vorn, "Introduction to Operational Amplifier Theory and Applications", McGraw-Hill, 1975. This particular subject is discussed in Chapter 1, page 11.

Even with the op-amp summing circuit, the binary weighted network suffers from the wide range of precision resistor values required. For a modest number of bits (up to 8 or even 12) these can be obtained with discrete resistors, but a range from 1k to 128K (for an eight bit converter) is impractical for monolithic construction. The R-2R ladder network overcomes this problem.

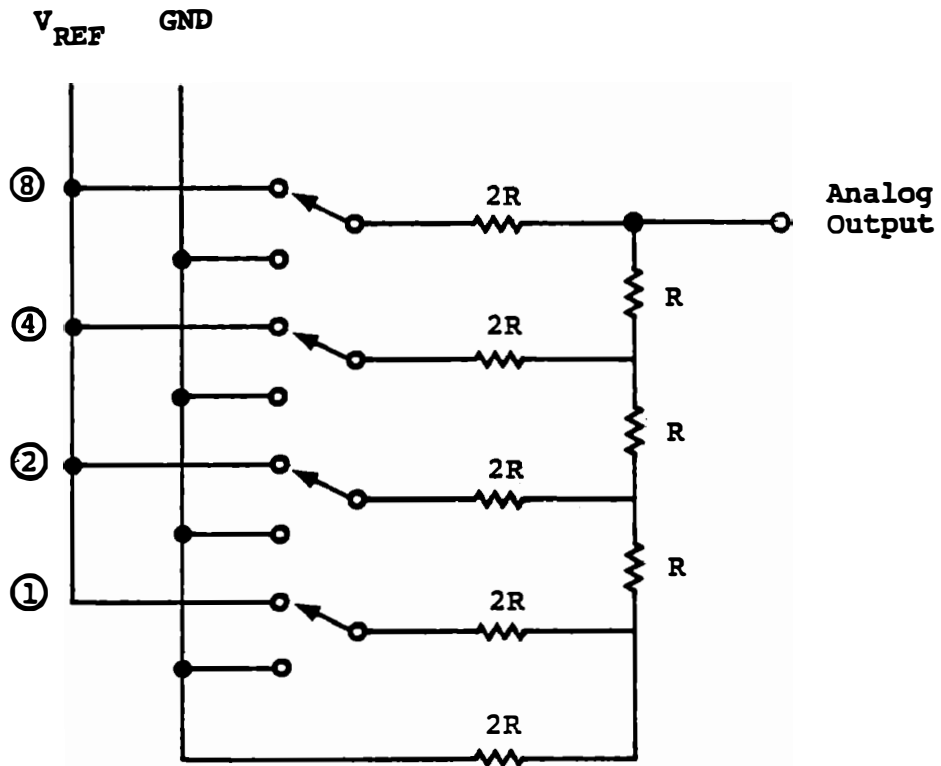
DIGITAL TO ANALOG OUTPUT



Binary Summing Circuit with OP AMP

Figure 4-22

DIGITAL TO ANALOG OUTPUT



R-2R Ladder Network

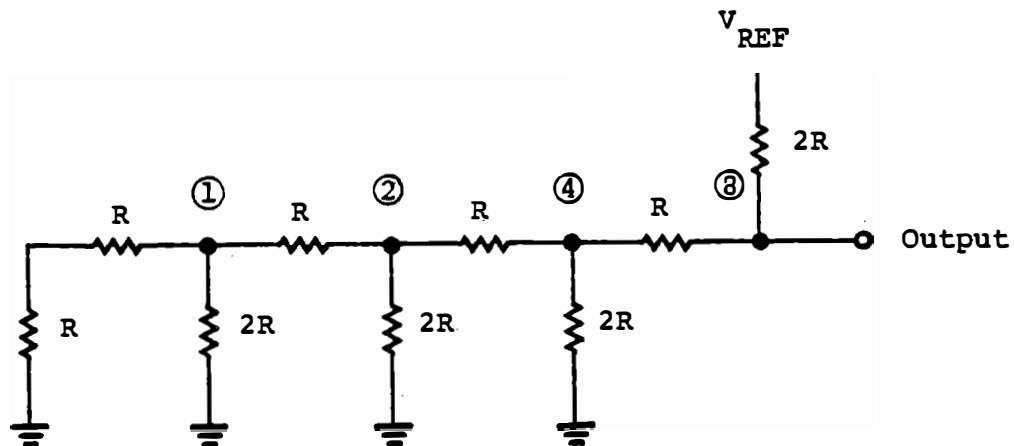
Figure 4-23

4.5.2 R-2R Ladder Network

Figure 4-23 shows an R-2R Ladder Network for digital to analog conversion. In this circuit bipolar (i.e. double throw) switches are required, so that for each bit a resistor is connected either to the reference voltage or to ground. If all bits are 0, then all connections go to ground and the output is 0 volts; if any bit is 1, its resistor is connected to the reference voltage and injects current into the network to develop a positive output voltage.

DIGITAL TO ANALOG OUTPUT

The R-2R network has two major advantages: only two resistor values are used, and they differ only by a factor of 2, so it can readily be constructed as a monolithic circuit; and it has a constant impedance independent of the binary input. The figure below shows an equivalent circuit for the case where the most significant bit is a 1 and the remaining bits are 0.



Looking to the left along the circuit from any node, with all of the less significant bits 0, one always sees an impedance of $2R$ to ground as depicted in Figures 4-24a through 4-24c. Now, if the $2R$ resistor for the most significant bit is connected to the positive reference voltage, a simple voltage divider is formed, giving an output signal equal to half the reference voltage as shown in Figure 4-24d.

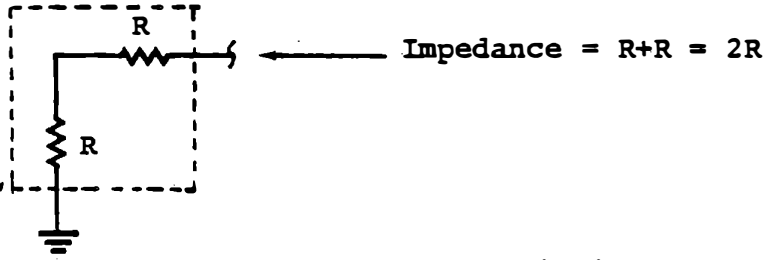


FIGURE 4-24a

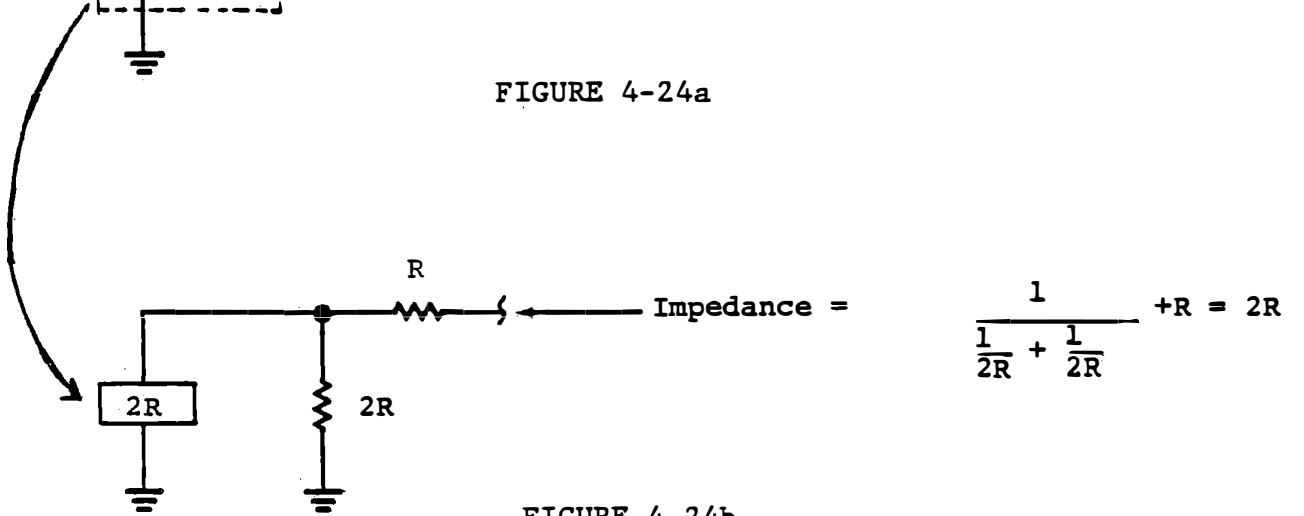


FIGURE 4-24b

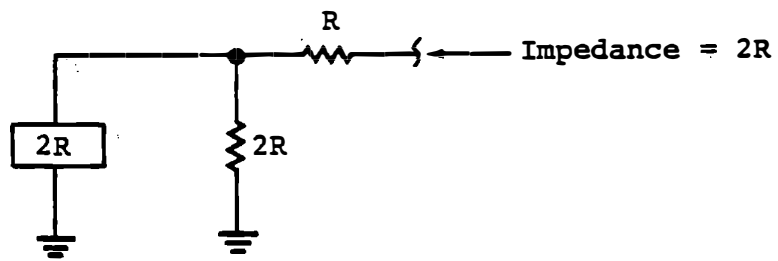


FIGURE 4-24c

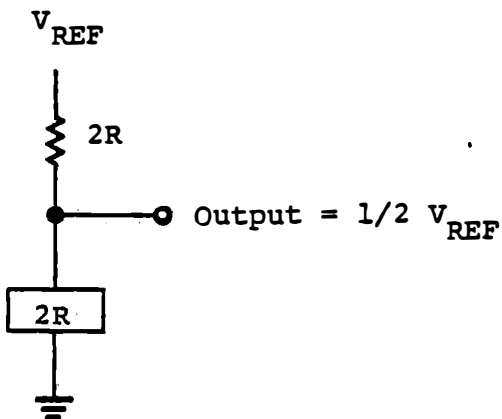
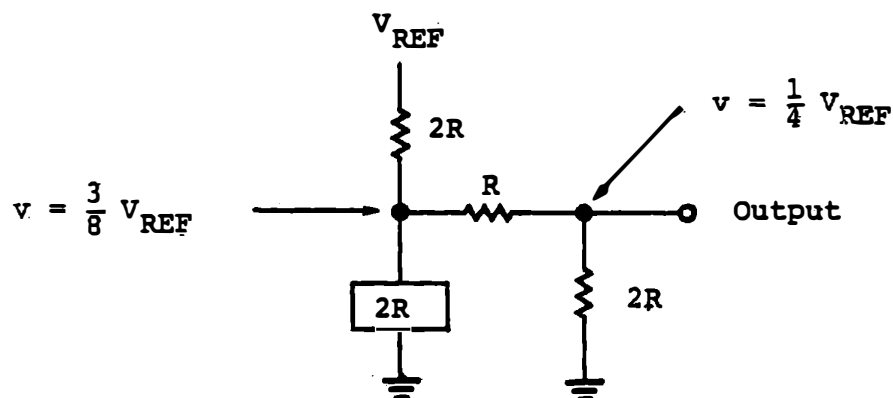
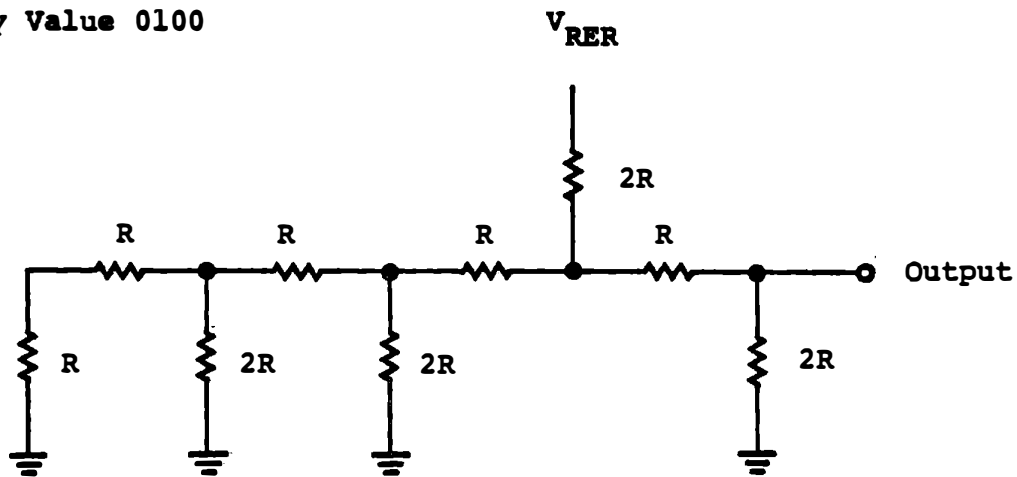


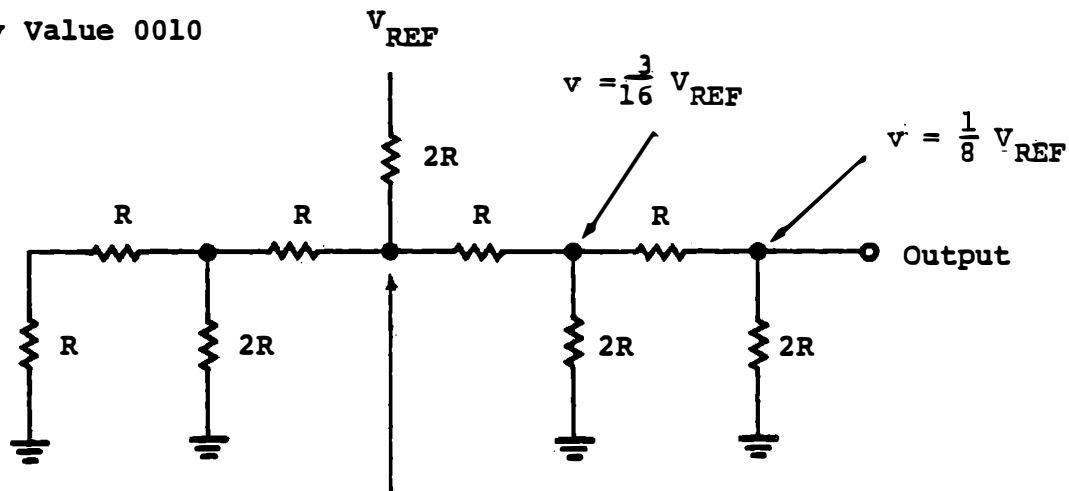
FIGURE 4-24d

DIGITAL TO ANALOG OUTPUT

Binary Value 0100



Binary Value 0010



$$v = \frac{11}{32} V_{REF}$$

Equivalent Circuits for Single Bit = 1

Figure 4-25

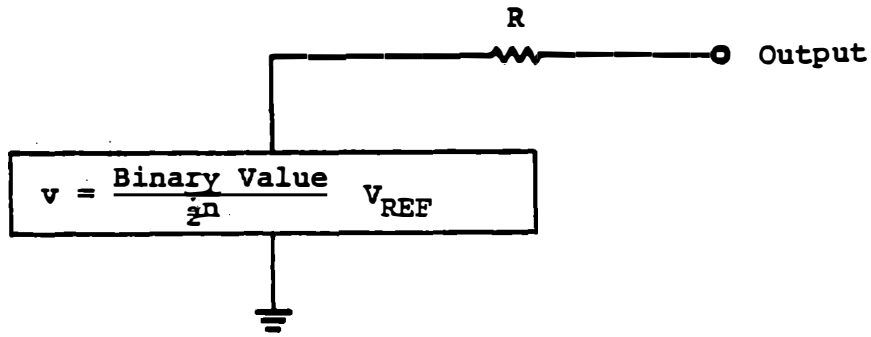
Figure 4-25 shows the voltages for two other cases where a single bit is 1. When multiple bits are 1's, their voltages add, to give an output proportional to the binary input and the reference voltage.

$$V_{\text{out}} = \frac{\text{Binary Value}}{2^n} V_{\text{ref}}$$

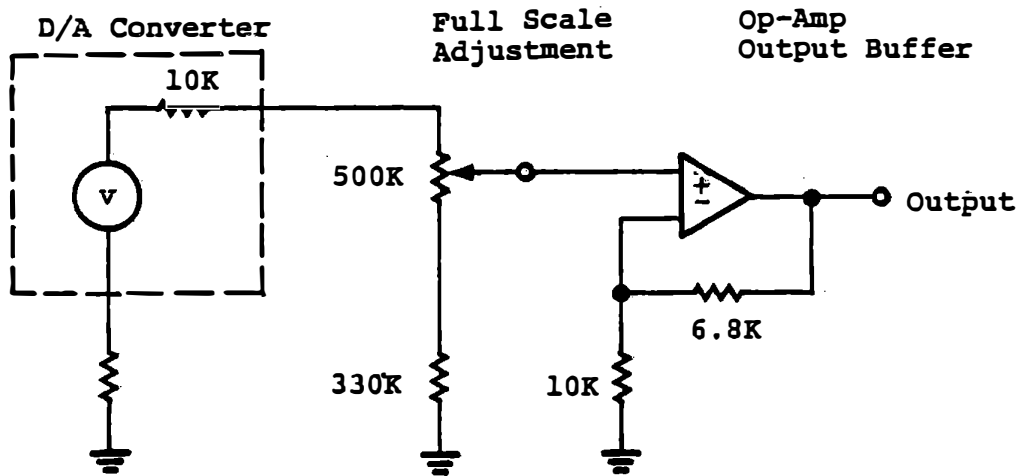
The output circuit sees a source impedance equal to R, from the parallel combination of the 2R resistor for the high bit and the ladder network to the left. Thus, the Thevenin equivalent circuit for the ladder network is the voltage given above with a series resistance equal to R, as shown in Figure 4-26.

The R-2R Ladder has been discussed in detail because it is used in the Ferranti D/A Converter included on the interface board. The operation of this device is discussed in the next section. It drives an operational amplifier, also shown in Figure 4-26, to isolate the load from the converter. A pot provides for adjustment of the full scale output, to compensate for error in the reference voltage and the value of R. In a system designed for a single purpose much less adjustment range would generally be provided, but in the interface board it was considered desirable to have a wide range to allow for various experiments.

DIGITAL TO ANALOG OUTPUT



R - 2R LADDER EQUIVALENT CIRCUIT



D/A Converter Output Circuit

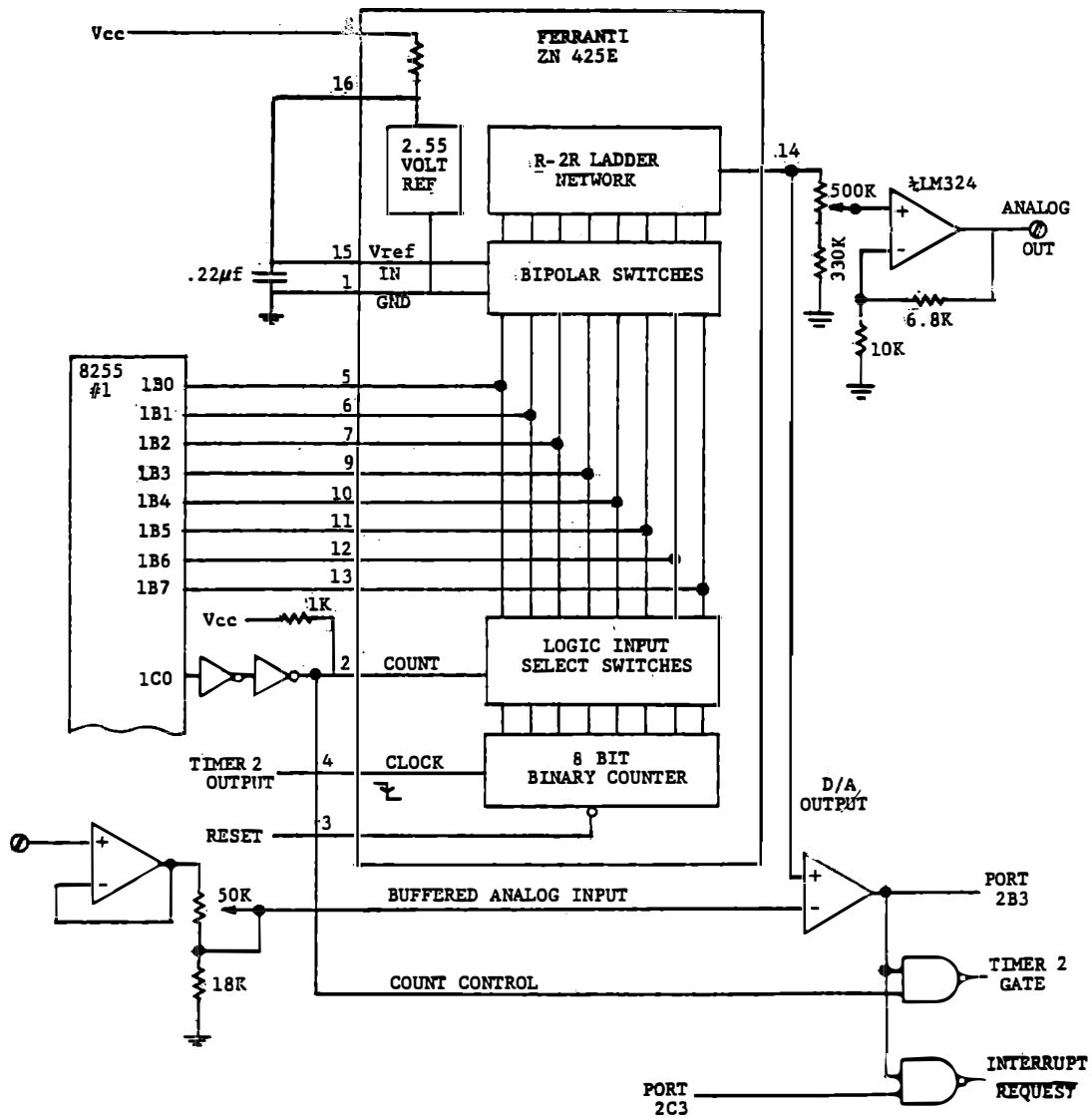
Figure 4-26

4.6 FERRANTI D/A CONVERTER

In this section we will describe the Ferranti ZN 425E Digital to Analog/ Analog to Digital Converter, and experiment with its D/A mode. Chapter 5 deals with analog to digital input using the 425.

The device is a monolithic 8 bit D/A converter using an R-2R ladder network. It contains an internal voltage reference source and a binary counter used for A/D conversion. Figure 4-27 is a block diagram of the device, with its inputs and outputs.

DIGITAL TO ANALOG OUTPUT



Ferranti D/A Converter

Figure 4-27

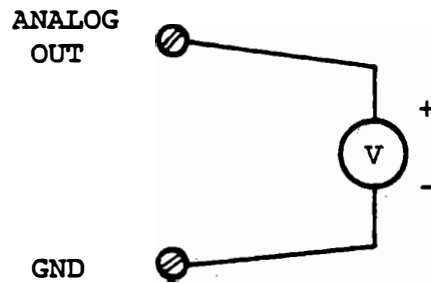
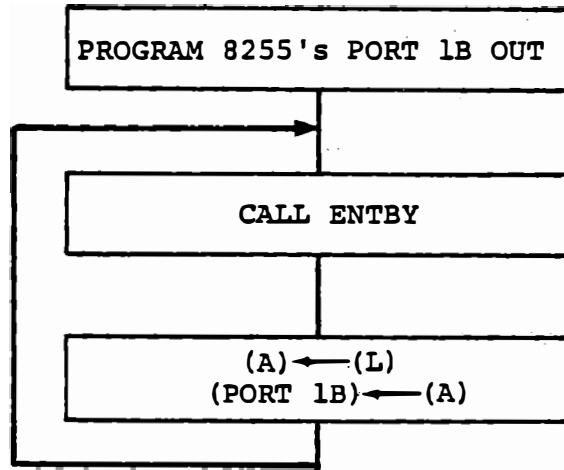
4.6.1 D/A Circuit Input and Output

The 425 has an eight bit port for digital data, connected to port 1B of 8255 1. These 8 bits control the bipolar switches for the R-2R ladder network. An internal circuit generates a 2.55 volt reference voltage for the ladder, although an external source can be connected. The analog output voltage appears at pin 14 and is connected to two op-amps. One of these (at the upper right of Figure 4-23) has a pot for full scale output adjustment as discussed in Section 4.5, and the op-amp generates a buffered analog output signal available at a tie block. This is the output signal to be used in the experiments of the following sections of Chapter 4.

4.6.2 D/A Circuit Control Signals

A count control signal at pin 2 of the 425 determines whether the eight bit digital data port is to be input to the 425 or output from the 425. When this signal from port 1C0 is low, the 425 accepts digital data from port 1B and converts the binary data to an analog voltage. This is the mode we will use in the remainder of Chapter 4. This signal also forces a NAND gate output high to give an enabling signal to Timer 2 gate input; in this mode, Timer 2 is independent of the D/A circuit and can be used for other purposes. The A/D interrupt control (port 2C3) should be low to inhibit any interrupt from the A/D comparator.

DIGITAL TO ANALOG OUTPUT



Keyboard to Voltage Program Flow
and Circuit Connection

Figure 4-28

We will discuss the remaining signals shown in Figure 4-27 in the next chapter, since they are concerned only with A/D input. To operate the 425 in D/A output mode, the following procedure should be used:

```
MVI  A,80      Program 8255 #1
OUT  CNT 1     A out B out C out
MVI  A,92      Program 8255 #2
OUT  CNT 2     A in B in C out
```

This sets count control (1C0) and interrupt control (2C3) low, as well as programming port 1B for output to allow writing data to the D/A converter.

4.6.3 Generating an Analog Voltage

EXERCISE

Write a program to accept data from the keyboard and write the data to the D/A converter. Observe this voltage at the ANALOG OUT tie block with a voltmeter. Adjust the full scale output to make the least significant bit of the data byte correspond to 10 millivolts. Figure 4-28 shows the program flow and the voltmeter connection.

DIGITAL TO ANALOG OUTPUT

For a 10 millivolt least count, full scale output should be 2.55 volts when the digital value is FF (=255). It is easier to read 2.50 volts on the meter, so key in FA (=250). Remember, you press a command key following the hex value. Adjust the output pot. Now key in various hexadecimal values and see that the output correctly follows the keyed value.

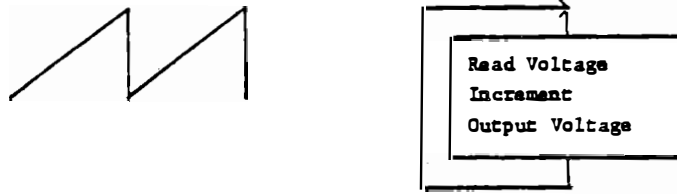
4.7 FUNCTION GENERATOR

The microprocessor, with the D/A converter, can be used to generate an analog signal that varies over time. If the variation of the signal repeats itself in a predictable manner, the result is a wave. The procedure of repeating a sequence of analog signals over time is called waveform synthesis or function generation. In this section we will experiment with several such functions, including sawtooth and triangular.

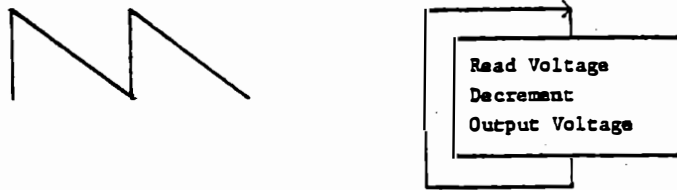
Unfortunately, the microprocessor is too slow to generate signals at useful frequencies for most purposes, typically being limited to less than one Hertz. However, some control applications do want very low frequency signals. Another possible use for waveform synthesis is examination of complex waveforms resulting from harmonics or the combination of non-harmonic frequencies, when real-time operation is not required.

DIGITAL TO ANALOG OUTPUT

Positive Sawtooth



Negative Sawtooth



Triangular Function

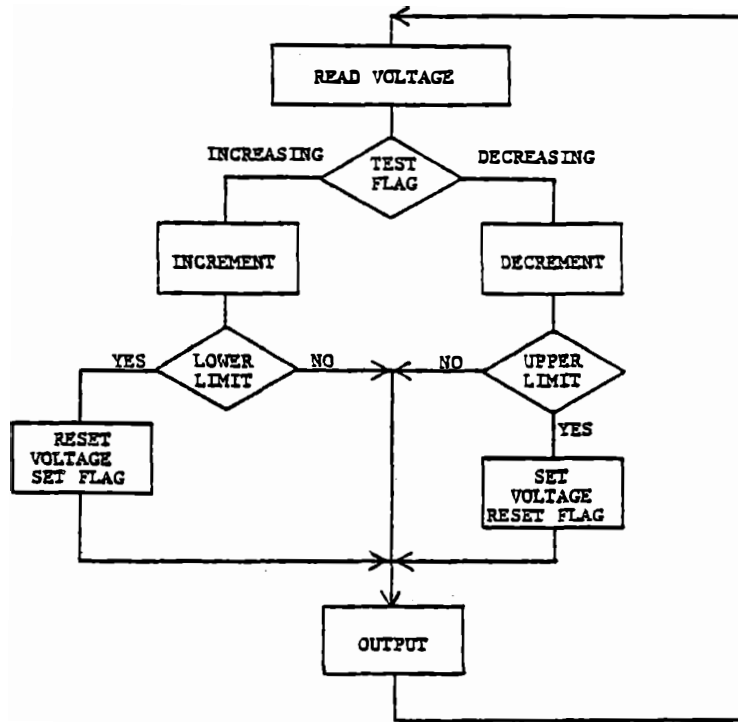
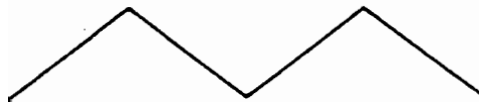


Figure 4-29

4.7.1 Voltage Ramps

One of the commonly needed control functions is a voltage ramp - an output voltage that increases or decreases linearly with time. Figure 4-29 shows positive and negative sawtooth functions and a triangular wave generator. The flow diagrams shown could be simple loops in a main program with some delay built in, but more probably each would be in an interrupt service routine invoked by a timer. The rate of increase or decrease is set by the time interval loaded to the timer.

If a full scale sawtooth is to be generated the service routine can read the present output voltage from the output port, increment the value, and output the new voltage. (Remember that a port programmed for output can be read). The service routine can be as simple as this:

PUSH	PSW	Save A and F
MVI	A,01	Re-enable timer 0
OUT	CNT2	Interrupt
IN	PORT1B	Read voltage
INR	A	Increment
OUT	PORT1B	Output voltage
POP	PSW	Restore A,F
EI		
RET		

Change INR A to DCR A for a negative sawtooth.

DIGITAL TO ANALOG OUTPUT

This page intentionally left blank

4.7.1.1 Voltage Ramp Program

EXERCISE:

Write a program to generate a positive sawtooth waveform using the above interrupt service routine. The main program must initialize the ports and timers. Then it has no further function, so it can end with an instruction that jumps to itself. The signal will appear at the Analog Out tie block. Connect your voltmeter across Analog Out to GND and observe the voltage increase gradually from zero to full scale (about 2.55 volts) and drop back to zero, in cycles of about 8 seconds.

You can generate a negative sawtooth function by changing the INR A instruction in the Interrupt Service Routine to DCR A.

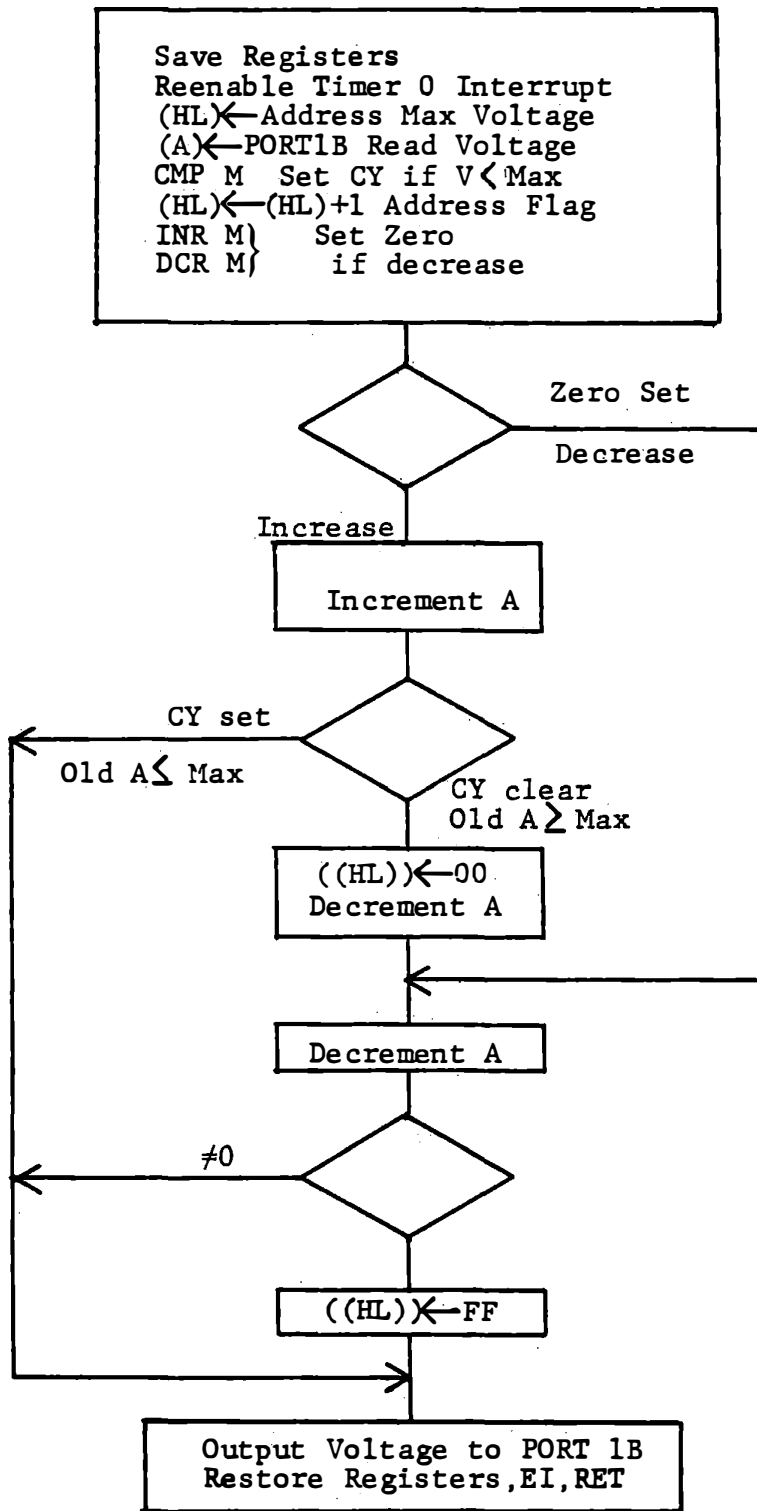
DIGITAL TO ANALOG OUTPUT

4.7.1.2 Triangular Wave Program

EXERCISE:

Write a program to generate a triangular waveform. The program must store a flag to indicate whether the voltage is increasing or decreasing. It will also need maximum amplitude data if the output is to be less than full scale. Figure 4-30 is a flow diagram of a service routine to generate a triangular function.

Rewrite the service routine of the previous program to compare the voltage with a maximum amplitude (stored at 8391) and to test an increase (FF) or decrease (00) flag stored at 8392. At the maximum voltage or at zero, reverse the flag. Use fixed values for the maximum amplitude and timer interval, or call for keyboard input of these if you wish. The next major exercise involves keyboard entry of data for a function generator.



Triangular Function Generator

Figure 4-30

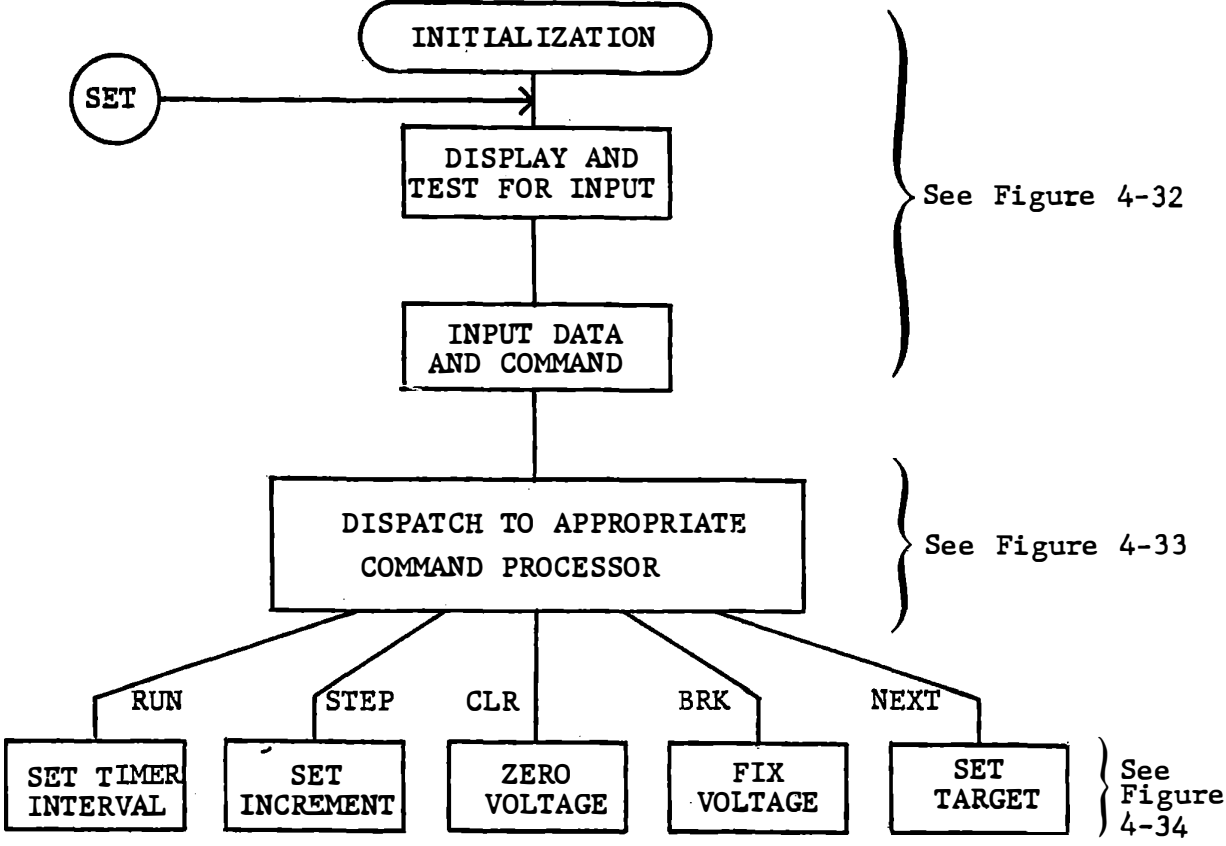
DIGITAL TO ANALOG OUTPUT

4.7.2 Keyboard Controlled Function Generators

In this and following exercises we shall develop a program to generate different waveforms selected by keyboard commands. The first exercise will again generate the triangular wave; later an exponential will be generated by numerical integration. Generation of a sine wave could be added. This exercise reviews some important programming techniques: interrupt service, keyboard input, dispatch tables, and using the stack for addresses. It also introduces methods of passing arguments to subroutines, and a variable subroutine call.

EXERCISE:

Write a program that repetitively increases the output voltage toward a target voltage and then gradually decreases it toward the complement of the original target. Accept keyboard data to set the rate of increase or decrease, and the target voltage. The rate can be adjusted either by adding a variable value to the output data at fixed time intervals (or subtracting the value for the decreasing ramp), or by incrementing (or decrementing) the output at a variable time interval. The latter approach gives a smoother ramp. The program shown in Figure 4-31 and following figures provides both approaches and also permits increasing or decreasing the rate by command key input. Timer 0 provides interval timing; it is used as a rate generator (mode 2) and interrupts the main program to add or subtract the voltage increment to the existing output data.



Keyboard Controlled Function Generator

Figure 4-31

DIGITAL TO ANALOG OUTPUT

When the output voltage is increased beyond the upper limit by adding the voltage increment, the voltage is set equal to the target and the mode is changed to decrease, and similarly when the output is decreased below the lower limit. This leads to a triangular output wave centered on half scale output. Section 4.7.2.5 describes the process in detail.

This page intentionally left blank.

DIGITAL TO ANALOG OUTPUT

4.7.2.1 Main Loop

In the main program (Figure 4-32) the display is controlled to show:

Time interval being used for interrupt

Voltage increment

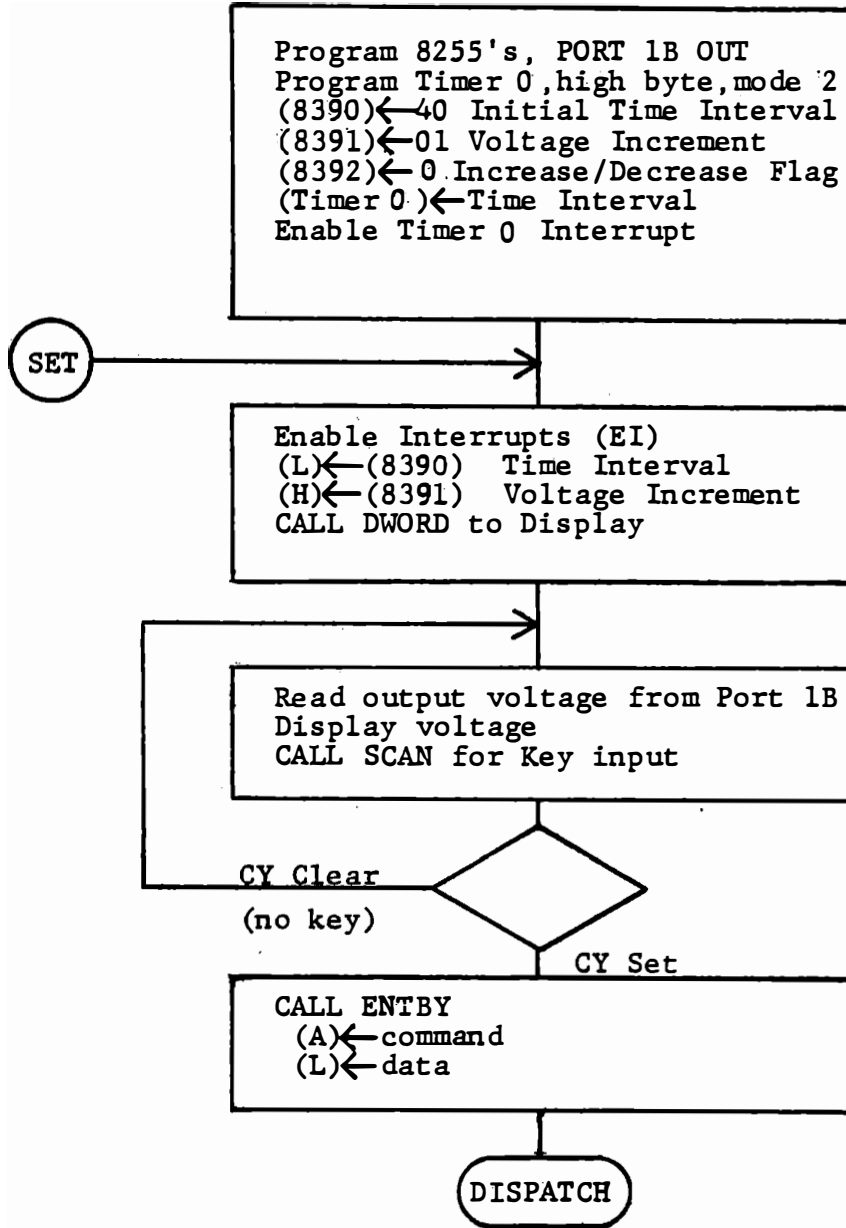
Present output voltage

Keyboard inputs are accepted to alter the time interval, the voltage increment, or the target voltage. Numeric entry is optional; a command key is required, and processed as shown in the table below:

RUN	Set time interval
STEP	Set voltage increment
CLR	Start ramp at zero
BRK	Set voltage and clear increment
NEXT	Store or complement target

Figure 4-32 shows the main program. After initialization and display of the initial values it repeatedly reads and displays the output voltage and tests the keyboard. When a key is pressed it calls ENTBY for new data and a command. The command is used to address an appropriate processing module.

RAMP GENERATOR
INITIALIZE, DISPLAY, ACCEPT INPUT, DISPATCH



(to Figure 4-33)

Keyboard Controlled Function Generator

Figure 4-32

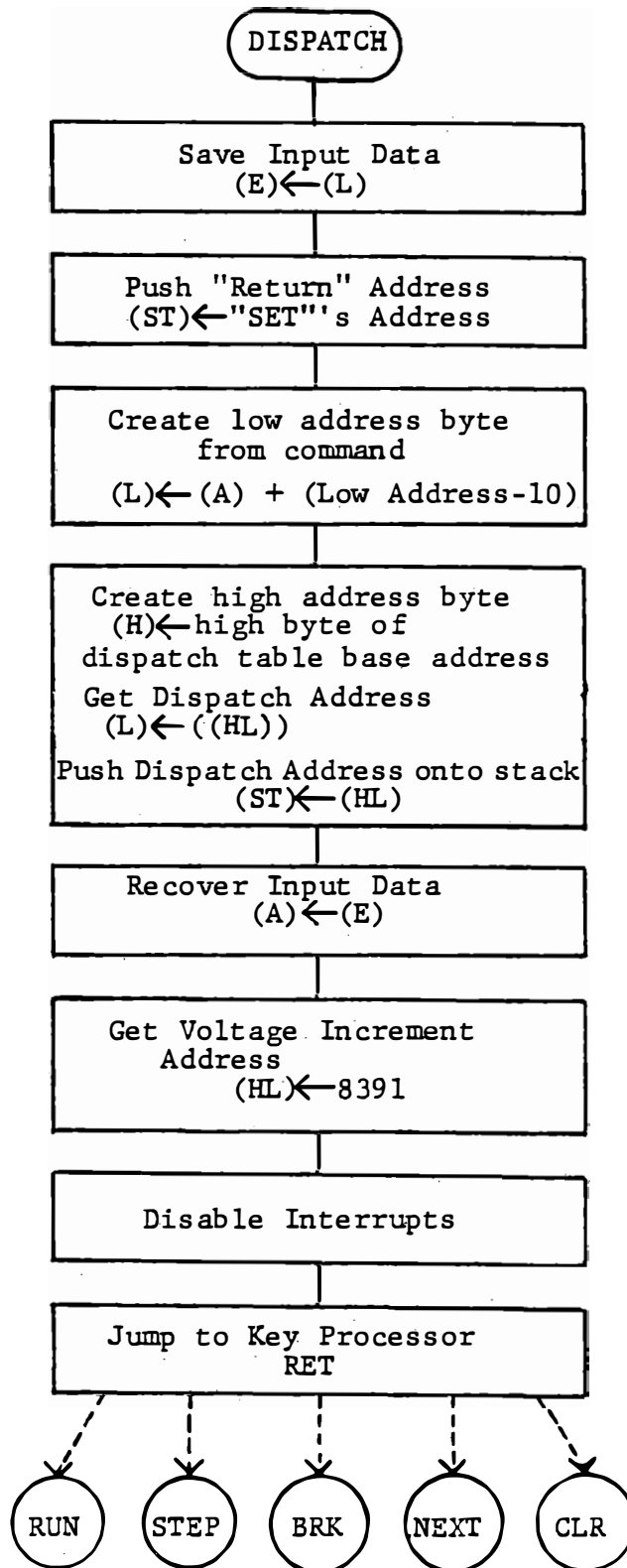
DIGITAL TO ANALOG OUTPUT

4.7.2.2 Dispatch for Keyboard Input

Dispatch is shown in Figure 4-33. We use the technique of dispatch tables and pushing addresses onto the stack: if you have not been using these techniques a review of Course 525, Section 7.5, "FLOW CONTROL TECHNIQUES".

Since all of the processes must return to the display of time interval and voltage increment, we will push that address onto the stack. The various processing modules can end with the return instruction instead of a three byte jump.

Then we add to the command key value (10H to 17H) the low byte of the dispatch table address minus 10H, so that MEM (10H) will direct us to the first location in the dispatch table. The dispatch low address byte is entered to L. Register H has already been loaded with 82. This dispatch address is pushed onto the stack. To reduce processing in the individual modules we will move the input data into register A and load HL with the address of the voltage increment (8391). Now a return will go to the appropriate module and the return address to the display function will be back on the top of the stack.



(To Figure 4-34)

Ramp - Dispatch

Figure 4-33

DIGITAL TO ANALOG OUTPUT

Some of the key input processing manipulates data that can also be changed by interrupt service. To prevent confusion the interrupts should be disabled while such processing is done. It is convenient to disable the interrupts by DI just before the "return" to the processing module, and enable just after the return from the processing module. Therefore, we have an EI instruction at the start of the main loop.

Since the monitor requires interrupts to operate in debug mode (STEP and BRK) it is desirable to modify the instructions that affect interrupts as you debug various parts of the program. Initially omit the instruction that enables the Timer 0 interrupt. Replace the DI instruction (before the "return" in dispatch) with RST4, so that the monitor will be called immediately before dispatching to the key processing module. Now you can either STEP or RUN through the main loop, using breakpoints as needed, but will always enter the monitor before dispatching. Since the timer interrupt is not enabled you can debug this part of the program even before writing the interrupt service routine.

After the main loop has been checked out replace the EI instruction at the beginning of the loop with a DI. Now the main loop, DWORD and ENTBY will operate without interruptions by the monitor, and you can try the various command entries very easily, always entering the monitor just before dispatch.

This page intentionally left blank.

DIGITAL TO ANALOG OUTPUT

4.7.2.3 Key Processing

Figure 4-34 shows the key processing modules. We enter the appropriate module with a return address (to the display function) in the stack, the key input in register A, and the address (8391) of the voltage increment in HL.

RUN sets the time interval. The input data byte is loaded to Timer 0 and stored at 8390 for display.

STEP stores the input byte at 8391 to set a new voltage increment. Since processing for RUN ends with MOV M,A; RET; we can simply dispatch to the MOV M,A instruction rather than duplicating it.

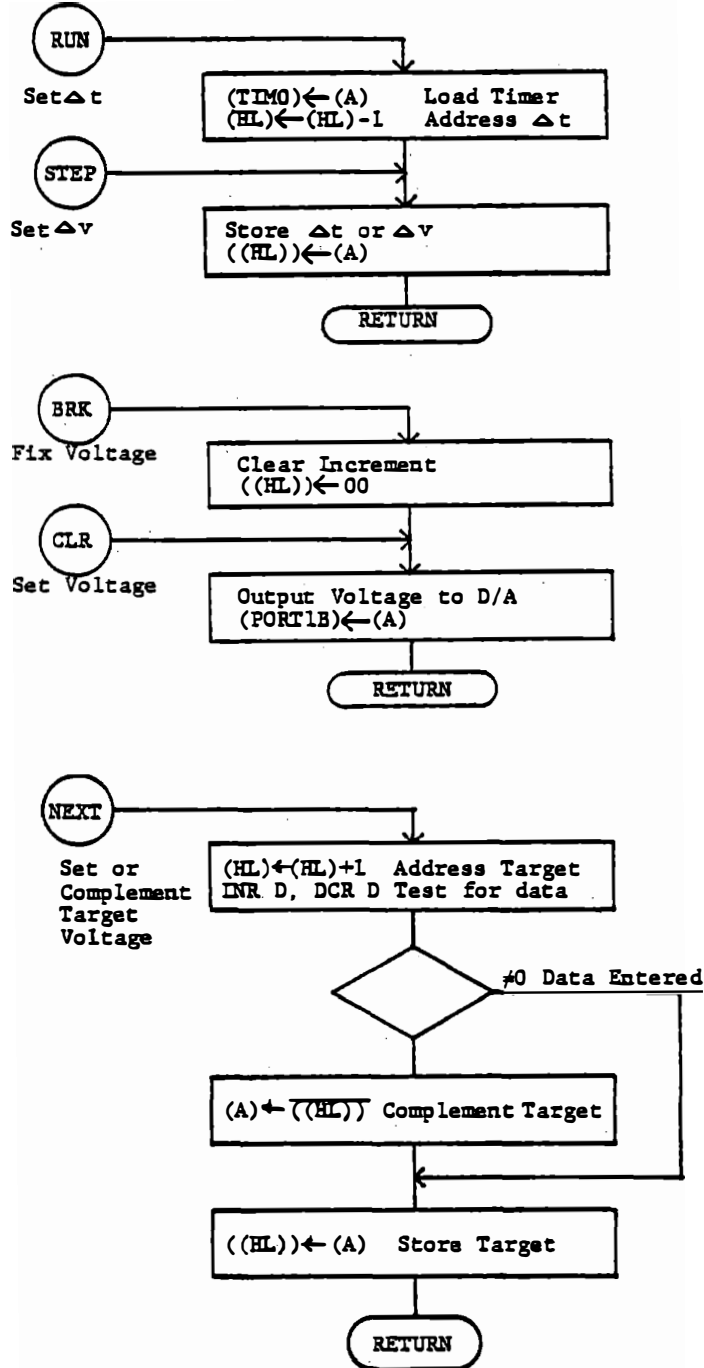
BRK stops the ramp and sets a fixed voltage. The ramp is stopped by setting the voltage increment (at 8391) to zero. Then the input data byte is written to PORT1B for the D/A output.

CLR writes the input data byte to the D/A output (00 if no data entered).

NEXT either sets a new target voltage (if a data byte was keyed in) or complements the old target voltage. Address the target voltage (8392) by INX H. ENTBY returns in register D a count of the number of hex keys entered. If this is zero, recover and complement the old target voltage; otherwise store the new data.

Enter key processing module from Dispatch with:

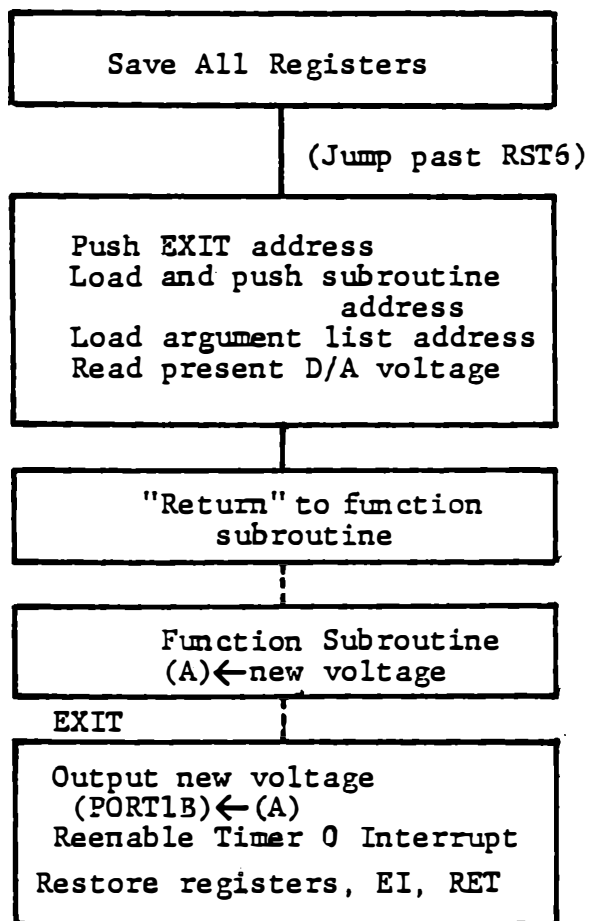
(A) = numeric data keyed (00 if none)
 (HL) = 8391 (address for Δv)



Function - Key Input Processing

Figure 4-34

DIGITAL TO ANALOG OUTPUT



Timer 0 Interrupt Service

Figure 4-35

4.7.2.4 Interrupt Service Routine

Interrupt service for timer 0 is shown in Figure 4-35. It performs the normal housekeeping duties of any interrupt service routine. At entry it saves the registers; at exit, it clears and re-enables the timer 0 interrupt flip-flop, restores the registers, enables interrupts and returns.

To provide for later exercises where other functions will be generated, the interrupt service routine enters a separate subroutine to perform the ramp calculation. Different subroutine calls are allowed (although at present only one will be used) by loading the subroutine's entry address from data memory. The procedure is:

LXI H,EXIT	Push an address for return
PUSH H	from the subroutine
LHLD FUNC	Push the stored entry
PUSH H	address for the subroutine
LXI H,8391	Load a data address
RET	Dispatch to the subroutine

The first function subroutine to be developed (TRIWV) is located at 8250. This address will be stored in memory locations 8398,99 to be loaded by interrupt service. Later we will make this a variable.

TRIWV needs as input data the present voltage, voltage increment, and increase/decrease flag. It returns the new voltage in register A.

We could require the function subroutine to read the voltage from PORT1B, load the other data from the defined addresses in memory, and

DIGITAL TO ANALOG OUTPUT

output the result to PORT1B. There are three advantages to the method chosen here:

- * The subroutine is "global". Another program could call the subroutine to operate on different data.
- * A different subroutine that needs the same data can be substituted for this one.
- * The function subroutine can be debugged by a temporary calling program.

The data needed by the function subroutine (arguments) can be passed in any of three ways:

- * Loaded into registers
- * Stored in memory locations reserved for this subroutine and its calling program
- * Stored in memory locations assigned to these particular data, with the address or addresses loaded into registers or into reserved memory locations.

Results can be returned in the same ways. In this program we will combine the first and third methods. The present voltage will be read from PORT1B by interrupt service and passed to the function subroutine in register A. The voltage increment and increase/decrease flag are in their assigned memory locations 8391 and 8392. Address 8391 is loaded into register pair HL and passed to the function. TRIWV obtains the increment from ((HL)) and the flag

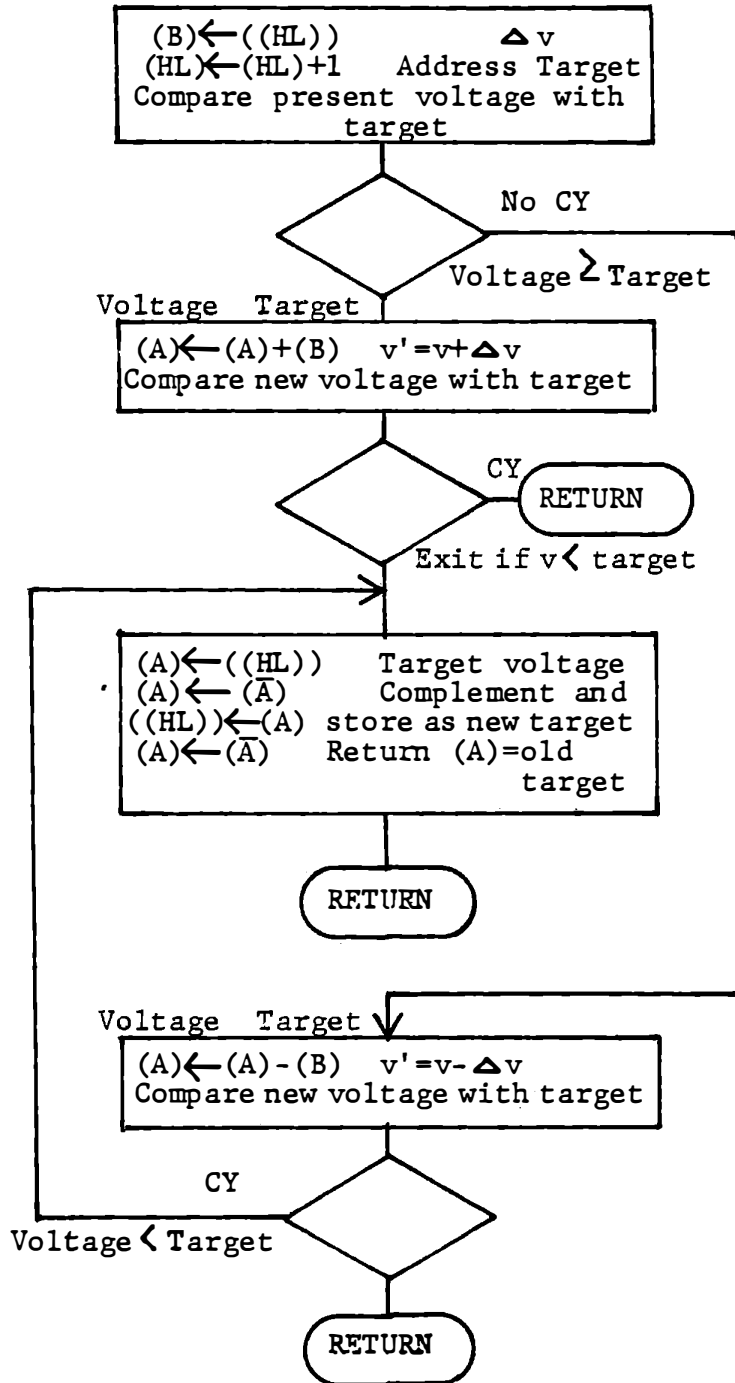
from ((HL)+1). The function subroutine would work equally well if some other calling program passed it a different memory address, with different data stored there.

The subroutine returns the new voltage in register A. This is output to the D/A converter by the interrupt service routine as the first step of its exit procedure.

DIGITAL TO ANALOG OUTPUT

TRIWV

Enter with (A) = present voltage
 ((HL)) = increment
 ((HL)+1) = target voltage



Triangular Wave Function Subroutine

Figure 4-36

4.7.2.5 Function Subroutine TRIWV

TRIWV calculates a new voltage by either adding or subtracting the voltage increment to the present voltage. It receives the following arguments (input data) from the calling program:

Location	Assignment
(A)	Present voltage
((HL))	Voltage increment
((HL)+1)	Target Voltage

The subroutine is shown in Figure 4-36. The increment is copied to register B and the present voltage is compared to the target to decide whether to add or subtract the increment. The new voltage is calculated and again compared with the target. If it was and still is less than the target, or if it was and still is greater, return with the new voltage in register A. If the voltage has passed the target, however, we will now complement the target voltage to be ready for the next entry, and return with the previous target voltage in register A to be output.

The result is a triangular waveform increasing until it reaches the target voltage and then decreasing to the complement of the voltage. The waveform is centered on 1.275 volts, half of full scale.

Note that TRIWV looks like any normal subroutine. It is not affected by the variable calling procedure, which could equally well be CALL TRIWV.

DIGITAL TO ANALOG OUTPUT

4.7.2.6 Instructions

Write the complete program in accordance with the flow charts presented. The solution shown in Figure 4-37 (a-h) follows the flow charts precisely and can be referred to if help is needed.

For display of the voltage in the main loop you can CALL DBYTE. For the sake of amusement, the program given in Figure 4-37 calls a subroutine to show the voltage in the LED'S as well as in the seven segment display. This subroutine has not been documented here; you can copy it, or figure it out, or just use DBYTE.

Memory assignments in the solution given are:

8200-8227	Initialize
8228-824F	Interrupt service
8250-827F	TRIWV function subroutine
8280-82A4	Main loop
82A5-82AC	Dispatch table
82AD-82DF	Key processing
82EB-82FF	Display subroutine
8390	Time interval
8391	Voltage increment
8392	Target Voltage
8398,99	Entry address for subroutine

FUNCTION GENERATOR - INITIALIZED

		A	D	D	R	CODE				
CODING SHEET	8	20	0	3E		MVI	A,	80		Program Ports Port 1B Out
			1	80						
			2	D3		OUT	CNT	1		
			3	07						
			4	3E		MVI	A,	92		
			5	92						
			6	D3		OUT	CNT	2		
			7	0F						
MICROCOMPUTER TRAINING SYSTEM	8			3E		MVI	A,	24		Program Timer 0 High byte Mode 2 Binary
			9	24						
		A	D3		OUT	TIM	CT			
			B	17						
			C	21		LXI	H,	0140		
			D	40						
			E	01						Initial Values
			F	22		PHLD		8390		
		8	21	0	90					(L) ← Δt
				1	83					(8390) ← Δt
			2	AF		XRA	A		(8391) ← Δt	
			3	32		STA		8392	(8392) ← target	
			4	92					(will be changed by interrupt)	
			5	83						
			6	D3		OUT	PORT	1B	(DA) ← 00	
			7	05						
			8	7D		MOV	A,	L		
			9	D3		OUT	TIM	0	(TIM) ← Δt	
			A	14						
INTEGRATED COMPUTER SYSTEMS			B	21		LXI	H,	8250		Store address of TRIWV subroutine for interrupt service
			C	50						
			D	82						
				E	22		SHLD		8398	
				F	98					
		8	22	0	83					
				1	3E		MVI	A,	01	Enable Timer 0 interrupt
				2	01					
				3	D3		OUT	CNT	2	
				4	0F					
			5	C3		JMP		8280	Jump to main loop	
			6	80						
			7	82						
			8							

Figure 4-37a

FUNCTION GENERATOR - INTERRUPT SERVICE

A D D R		CODE										
CODING SHEET	8	0										
		1										
		2										
		3										
		4										
		5										
		6										
		7										
MICROCOMPUTER TRAINING SYSTEM	8 22	8	F5	PUSH	PSW						Timer D	
		9	E5	PUSH	H						save registers	
		A	D5	PUSH	D							
		B	C5	PUSH	B							
		C	C3	JMP	8233							Jump past RST6
		D	33									
		E	82									
		F	00	NO P								
		8 23	0	E7	RST4							If RST6 occurs
			1	FB	EI							enter monitor
			2	C9	RET							
		8 23	3	21	LXI	H, EXIT						Push EXIT
			4	44								address for
			5	82								returns from
			6	E5	PUSH	H						function module
			7	2A	LHLD	FUNC						Push function
		8	98								address for	
		9	83								dispatch to	
		A	E5	PUSH	H						function module	
		B	21	LXI	H, 8391						(HL) ← address	
		C	91								of Δv for	
		D	83								function module	
		E	DB	IN	PORT1B						(A) ← present	
		F	05								voltage	
INTEGRATED COMPUTER SYSTEMS	8 24	0	C9	RET							Dispatch to	
		1	00	NO P							function module	
		2	00	NO P								
		3	00	NO P								
		4										
		5										
		6										
		7										
	8											

Figure 4-37b

FUNCTION GENERATOR - DATA FROM EXPERIMENT -

	A	D	D	R	CODE										
CODING SHEET	8	0													
		1													
		2													
		3													
	824	4	D3			OUT	PORT	1B			EXIT				
		5	05												
		6	3E			MVI	A,	01							
		7	01												
		8	D3			OUT	CNT	2							
		9	0F												
MICROCOMPUTER TRAINING SYSTEM	A	C1			POP	B									
	B	D1			POP	D									
	C	E1			POP	H									
	D	F1			POP	PSW									
	E	FB			ET										
	F	C9			PET										
	8	0													
INTEGRATED COMPUTER SYSTEMS		1													
		2													
		3													
		4													
		5													
		6													
		7													
		8													
		9													
		A													
		B													
		C													
		D													
		E													
		F													
		8	0												
	1														
	2														
	3														
	4														
	5														
	6														
	7														
	8														

Figure 4-37c

FUNCTION TRIWV

A D D R		CODE					
CODING SHEET	8 25	0 46	MOV	B, M			$(B) \leftarrow \Delta v$
		1 23	INX	H			Address target
		2 BE	CMP	M			Compare voltage
		3 D2	JNC	8 260			If voltage \geq target, go to subtract
		4 60					
		5 82					
		6 80	ADD	B			$(A) \leftarrow v' = v + \Delta v$
		7 DA	JC	8 266			If beyond FF go to reverse
		8 66					
MICROCOMPUTER TRAINING SYSTEM		9 82					
	A	BE	CMP	M			Compare new voltage to target
	B	D2	JNC	8 266			If past target go to reverse
	C	66					
	D	82					
	E	C9	RET				Exit if not past
	F	00	NOP				
	8 26	0 90	SUB	B			$(A) \leftarrow v' = v - \Delta v$
		1 DA	JC	8 266			If below 00 go to reverse
	2 66						
	3 82					Compare new voltage to target	
	4 BE	CMP	M			Exit if not past	
	5 D0	RNC				Reverse	
INTEGRATED COMPUTER SYSTEMS	8 26	6 7E	MOV	A, M			Reverse
		7 2F	CMA				Complement target voltage
		8 77	MOV	M, A			Return with $v = \text{old target}$
		9 2F	CMA				
	A	C9	RET				
	B						
	C						
	D			ENTER WITH			
	E			(A) = PRES			SENT VOLTAGE
F			((HL)) = INC			CREMENT, Δv	
8	0		((HL) + 1) =			TARGET VOLTAGE	
	1						
	2		RETURN				
	3		(A) = NEW			VOLTAGE	
	4		(HL) ADDRESSING			TARGET	
	5		AT OVERFLOW			RETURN	
	6		(A) = OLD			TARGET	
	7		TARGET			COMPLEMENTED	
	8					Figur 4-37d	

FUNCTION - MAIN LOOP, DISPATCH

		A	D	D	R	CODE					
CODING SHEET	8 28	0	FB		EI						Enable external bus processing
		1	2A		LHLD				8390		(L) ← ΔE
		2	90								(H) ← ΔV
		3	83								
		4	CD		CALL				DWORD		
		5	D1								
		6	02								
MICROCOMPUTER TRAINING SYSTEM	828	7	DB		IN				PORT1B		Read output voltage
		8	05								Display voltage
		9	CD		CALL				DISPL		
		A	EB								
		B	82								
		C	CD		CALL				SCAN		Test keyboard
		D	57								
		E	02								
		F	D2		JNC				8287		Loop if no key
	INTEGRATED COMPUTER SYSTEMS	8 29	0	87							
		1	82								
		2	CD		CALL				ENTBY		(A) ← command
		3	36								(L) ← data
		4	03								(D) ← key count
		5	5D		MOL				E, L		(E) ← data
		6	21		LXI				H, 8280		Push address of display function
		7	80								in return from process module
		8	82								
		9	E5		PUSH				H		
	A	C6		ADI				95		Add dispatch table address -10 to command	
	B	95									
	C	6F		MOV				L, A		(ST) ← dispatch address	
	D	6E		MOV				L, M			
	E	E5		PUSH				H			
	F	7B		MOV				A, E		(A) ← data counter	
	8 2A	0	21		LXI			H, 8391		Address of voltage increment	
		1	91								
		2	83								
		3	F3		DI						
		4	C9		RET						
		5									
		6									
		7									
		8									

Figure 4-37e

A D D R CODE FUNCTION - KEY PROCESSING

	A	D	D	R	CODE	FUNCTION	KEY	PROCESSING			
CODING SHEET	8				0						
					1						
					2						
					3						
					4						
		82A				5	A4	MEM			Undefined ↓
						6	A4	REG			
						7	A4	ADDR			
						8	B0	STEP			Set Δv
						9	AD	RUN			Set Δt
MICROCOMPUTER TRAINING SYSTEM					A	CO	NEXT			Set or reverse target	
					B	B2	BRK			Set fixed voltage	
					C	B4	CLR			Set voltage	
		82A				D	D3	OUT	TIMD	RUN Set Δt	
						E	14			Load timer D	
						F	2B	DCX	H	Address Δt	
		82B				0	77	MOV	M, A	STEP Set Δv	
						1	C9	RET			
		82B				2	36	MVI	M, 00		BRK Set fixed v
						3	00				Clear Δv
INTEGRATED COMPUTER SYSTEMS					82B	4	D3	OUT	PORT/B	CLR Set v	
						5	05				
						6	C9	RET			
						7					
						8					
						A					
						B					
						C					
						D					
						E					
					F						
	8				0						
					1						
					2						
					3						
					4						
					5						
					6						
					7						
					8						

Figure 4-37f

A D D R CODE FUNCTION - KEY PROCESSING (continued)

CODING SHEET		CODE		FUNCTION		KEY		PROCESSING		(continued)	
8	2C	0	23	I	N	X	H				NEXT set target
		1	14	I	N	R	D				Test for data entered
		2	15	D	C	R	D				
		3	C2	J	N	Z	8	2	C	8	If data entered
		4	C8								jump to store as
		5	82								target voltage
		6	7E	M	O	V	A	,	M		Else complement
		7	2F	C	M	A					old target
8	2C	8	77	M	O	V	M	,	A		store target
		9	C9	R	E	T					
		A									
		B									
		C									
		D									
		E									
		F									
		8	0								
		1									
		2									
		3									
		4									
		5									
		6									
		7									
		8									
		9									
		A									
		B									
		C									
		D									
		E									
		F									
		8	0								
		1									
		2									
		3									
		4									
		5									
		6									
		7									
		8									

Figure 4-37g

DISPLAY VOLTAGE SUBROUTINE

		A	D	D	R	CODE			
CODING SHEET	8	0							
		1							
		2							
		3							
		4							
		5							
		6							
		7							
		8							
		9							
MICROCOMPUTER TRAINING SYSTEM	A								
	82E	B	CD			CALL	DBYTE		Display voltage
		C	95						
		D	02						
		E	79			MOV	A, C		(A) ← voltage
		F	21			LXI	H, 00FF		To shift ones into (H)
	82F	0	FF						
		1	00						Divides by 29 to divide voltage range into 9 segments
	82F	2	D6			SUI	1D		When remainder less than 0 go to display
		3	1D						Shift in ones
	4	DA			JC	82FB		Loop	
	5	FB							
	6	82							
	7	29			DAD	H			
	8	C3			JMP	82F2			
	9	F2							
INTEGRATED COMPUTER SYSTEMS	A	82							
	82F	B	7C			MOV	A, H		Display result in LED's
		C	D3			OUT	PORT 1A		
		D	04						
		E	C9			RET			
		F	00			NOP			
	8	0							
		1							
		2							
		3							
	4								
	5								
	6								
	7								
	8								

Figure 4-37h

4.7.2.7 Debugging

Debugging techniques for the main program were suggested in Section 4.7.2.2. These are repeated here, referring to addresses in the given solution.

- a) Omit OUT CNT2 at 8223 to avoid enabling timer interrupt.
- b) To debug dispatch loop omit EI at 8280 and DI at 82A3.
- c) To debug key processing modules enter DI at 8280 and RST4 at 82A3.
- d) To debug interrupt service enter RST5 at 8280 and EI at 82A3. Replace EI at end of interrupt service with DI at 824E. Now interrupt service will be called, with monitor interrupts enabled, after each key entry has been processed. The monitor will be disabled during key input and dispatch but enabled during key processing and interrupt service.
- e) To run the debugged program restore all of the modified instructions:

8223	OUT	CNT2
824E	EI	
8280	EI	
82A3	DI	

DIGITAL TO ANALOG OUTPUT

As you develop your own program keep debugging in mind and provide for similar techniques. Keep a separate list of modified instructions to be sure you restore them all.

4.7.2.8 Program Operation

With the initial value of 40 for the time interval and 01 for the voltage increment, the output voltage will increase and decrease slowly enough to be observed in the display and on a voltmeter. Using NEXT will reverse the direction part way up or down. With an oscilloscope you can observe the triangular output at higher frequencies. Try entering smaller values of both time interval and voltage increment, keeping a constant ratio so that the total period is the same (see list below). Now observe the effect the on the waveform.

Voltage Increment (STEP)	Time Interval (RUN)	Total Period (Seconds)
01	40	4.096
02	80	4.096
03	00	4.096
04	00	4.096
01	01	0.064
02	02	0.064
04	04	0.064
08	08	0.064
10	10	0.064
20	20	0.064
40	40	0.064

DIGITAL TO ANALOG OUTPUT

The CLR key starts the triangle from zero or from any desired value keyed in. This will be of more use in later exercises. BRK stops the function and sets the voltage to any value keyed in. This is convenient for calibrating the A/D output. Request a voltage and adjust the analog out pot to make the output agree with the voltmeter. Operate at a slower rate (press 0, RUN)

Enter 1.55 volts for the target by 9B,NEXT. Press CLR to start a ramp at zero. Observe the voltage climb to 1.55 and then drop to 1.00, (64 hex) and climb again. Press NEXT while it is rising and see it fall.

While the voltage is falling, press CLR to start at zero. Now the voltage will rise toward the 1.0 volt target, complement the target to give 1.55 volts, and continue to climb. Then it will resume the steady rise and fall between 1.0 and 1.55.

DIGITAL TO ANALOG OUTPUT

4.7.3 Exponential Function

The charging of a capacitor leads to a voltage which increases with time at a slowing rate as the capacitor voltage approaches the source voltage. (See Figure 4-38). The capacitor voltage is given by:

$$(a) \quad v = Q/C$$

where Q is the accumulated charge and C the capacitance. The charge is accumulated as current flows into the capacitor and is calculated from the time integral of the current.

$$(b) \quad Q = \int_0^t i dt$$

The current through the resistor is proportional to the voltage across the resistor: the source voltage V_s minus the capacitor voltage at that instant.

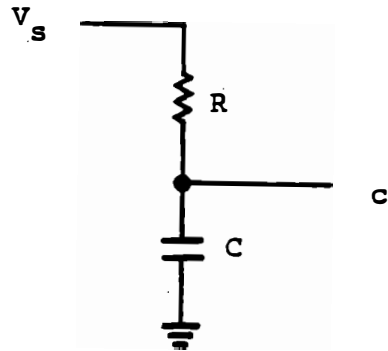
$$(c) \quad i = \frac{1}{R} (V_s - v)$$

then

$$(d) \quad v' = \frac{1}{RC} \int_0^t (V_s - v) dt$$

which can be solved to give:

$$(e) \quad v' = V_s(1 - e^{-t/RC})$$



$$v_c = \frac{1}{RC} \int_0^t (V_s - v_c) dt$$

$$v_c = V_s (1 - e^{-t/RC})$$

Exponential Function

Figure 4-38

DIGITAL TO ANALOG OUTPUT

As t increases without limit the exponential term approaches zero and the capacitor voltage approaches the source voltage. With a digital computer we can solve the integral equation (d) numerically without resort to the explicit solution (e); doing so can generate the exponential function.

For numerical integration there is no infinitesimal time, so equation (d) is rewritten as:

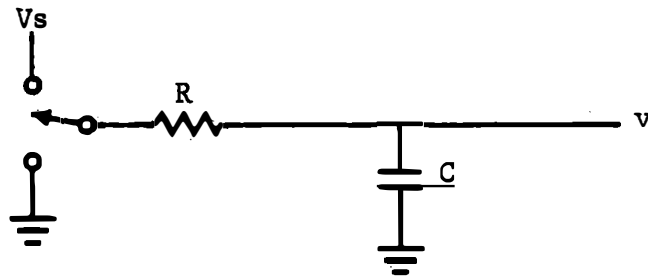
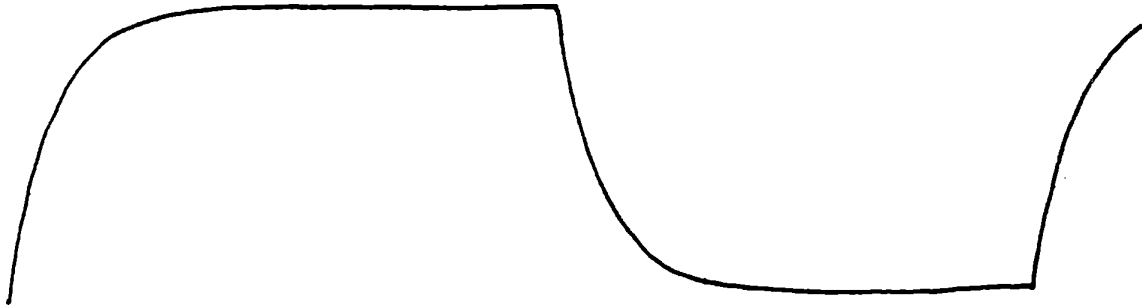
$$(f) \quad V' = V + (V_s - v)\Delta t/RC$$

In this section we shall create a function subroutine to evaluate equation (f), to be called in place of TRIWV in our function generator program.

As the output voltage v approaches the source voltage V_s , it changes very slowly. To obtain a good representation of the exponential we must use two byte precision for the calculation. We will store the less significant byte of v in memory, and the more significant byte will be read from and output to PORT 1B.

This page intentionally left blank

DIGITAL TO ANALOG OUTPUT



Successive Charge/Discharge Cycles

Figure 4-39

Eventually v will stop changing, even though it is not yet quite equal to V_s . At this point we will start a discharge period, in which the source voltage is zero. Then:

$$(g) \quad V' = V + (0 - V)\Delta t/RC$$

Again, after some time, v will stop changing and we will switch back to the charging function. Successive charge/discharge cycles are shown in figure 4-39.

We will store two variables to control the charge or discharge. The source voltage V_s will be stored as hex data entered with MEM. A "switch variable" will represent the state of the switch shown in figure 4-39. The "target voltage" stored or complemented by NEXT (at 8392) will be used as the switch variable: if its most significant bit is one the voltage will increase toward V_s ; if the most significant bit is zero the voltage will decrease toward ground. Thus the function of the NEXT key is preserved.

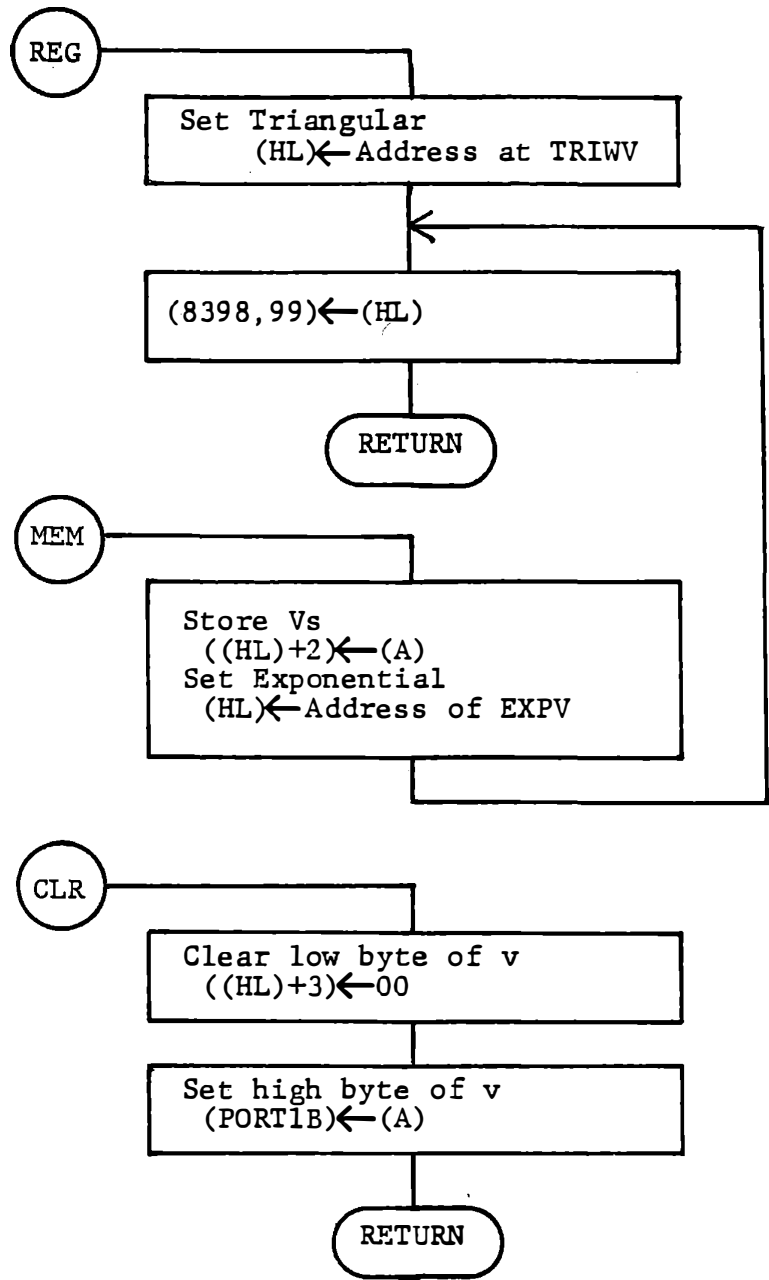
DIGITAL TO ANALOG OUTPUT

4.7.3.1 Exponential Waveform Generator

EXERCISE:

Modify and add to the program of Section 4.7.2 to generate either a triangular wave or an exponential. Accept keyboard input of source voltage (V_s), time interval Δt , and the ratio $\Delta t/RC$.

Data will be entered as one byte values with command keys, as before. Key input processing is shown in Figures 4-34 and 4-40, and described in the following subsections.



Key Selection of Waveform

Figure 4-40

DIGITAL TO ANALOG OUTPUT

4.7.3.2 Selecting the Waveform

Since the waveform is generated by interrupt service, this module must behave differently for the two different waveforms. The distinction is handled by storing a jump address in memory (at 8398, 99) according to the REG (for triangular) or MEM (for exponential) command. The interrupt service routine of the function generator program (Section 4.7.2) provides for the use of this variable subroutine address by:

```
LXI H, EXIT      PUSH exit address
PUSH H
LHLD 8398        PUSH subroutine address
PUSH H
LXI H, 8391      Load data address
RET              Dispatch to selected subroutine
```

4.7.3.3 Data Entry and Storage

As indicated in the table on the next page, there are two new variable data bytes to be stored in addition to the function address. The less significant byte of *v* is calculated and stored by EXPV. It is also to be cleared by the BRK and CLR keys. This is necessary in order to obtain a consistent result each time CLR is used.

The source voltage V_s is to be entered by MEM. For two byte precision in the calculation we will use this value as the high byte, with zero for the low byte. (If no value is entered with MEM, the source voltage will be set to zero.)

MEM also selects the exponential vaveform by storing the address of EXPV at 8398, 99. REG is to select the triangular waveform by storing the address of TRIWV at 8398, 99.

RUN, STEP and NEXT are unchanged:

RUN loads timer 0 and stores Δt at 8390.

STEP stores Δv or $\Delta t/RC$ at 8391.

NEXT stores the switch variable or target voltage at 8392. If no data were entered NEXT complements the old value.

Memory assignments are:

8390	Δt
8391	Δv or $\Delta t/RC$
8392	Target voltage or switch variable
8393	V_s
8394	v (low byte)
8398,99	Function address
8250	Entry to TRIWV
8100	Entry to EXPV

DIGITAL TO ANALOG OUTPUT

4.7.3.4 Calculating Exponential Voltage

Subroutine EXPV, shown in Figure 4-41a calculates:

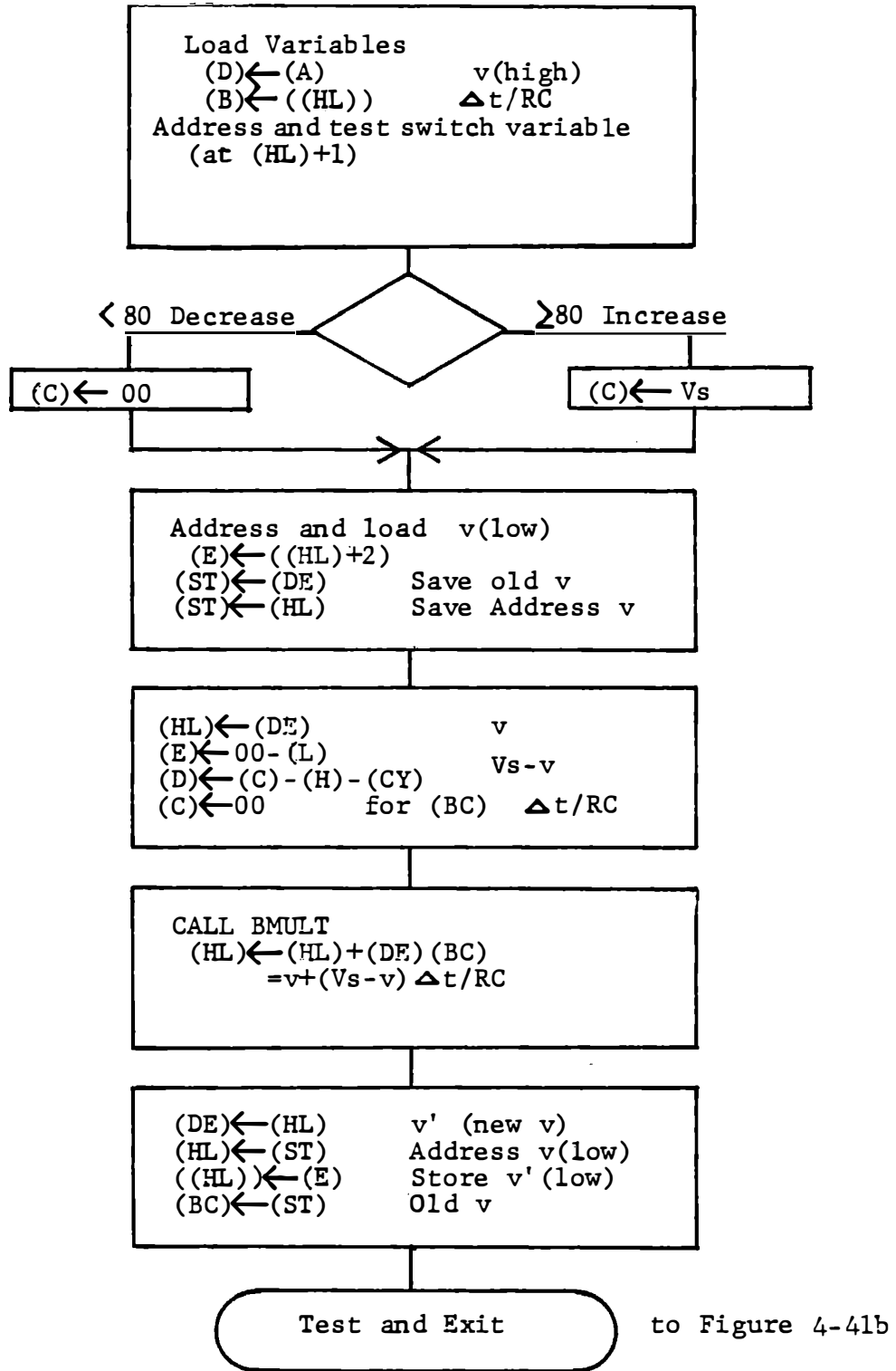
$$v' = v + (V_s - v) (\Delta t / RC)$$

At entry, register A contains the high byte of v , and register pair HL contains the address of the memory location for t/RC , a single byte value. The other variables are contained in successive locations:

Offset Address	Nominal Address	Variable Address
(HL)	8391	$\Delta t / RC$
(HL)+1	8392	switch variable
(HL)+2	8393	V_s (high byte)
(HL)+3	8394	v (low byte)

Although specific memory addresses are listed above, the subroutine will use only offset addressing, so that it could be called from another program module with different data in different storage locations.

The subroutine returns with the contents of the memory locations updated, and the new two byte value of v in (DE).



EXPV - Calculate Voltage

Figure 4-41a

DIGITAL TO ANALOG OUTPUT

The subroutine moves the input voltage into register D and loads the other variables, incrementing (HL) to access successive bytes. It tests the switch variable and either loads V_s into (C) from memory if the voltage is to increase or clears C if the voltage is to decrease. Then the low byte of v is loaded to register E, so that (DE) contains the two byte value of v . Both v and its address are saved in the stack, and the calculation begins.

$$V' = V + (V_s - V)(\Delta t/RC)$$

is to be evaluated. We shall develop a subroutine (BMULT) to evaluate this expression with the following entry data, all as two byte variables.

$$\begin{aligned} (BC) &= \Delta t/RC \\ (DE) &= V_s - v \\ (HL) &= v \end{aligned}$$

Before calling BMULT, the value of $V_s - v$ is calculated by EXPV as follows: Move v into (HL), and subtract its low byte from zero, placing the result in (E). Subtract the high byte of v from V_s (or zero if the voltage is decreasing) using the SBB instruction (subtract with borrow) and place the result in (D). Clear the low byte of $\Delta t/RC$ (register C) and call BMULT.

The new value of v is returned in register pair HL. It is moved to DE, the address for v (low) is popped and its new value is stored. The old value of v is popped into (BC) for comparison.

This page intentionally left blank

DIGITAL TO ANALOG OUTPUT

Figure 4-41b shows the procedure for changing the switch variable.

The new value of v is compared with the old value. If it has changed, the process will continue in the same direction. When v no longer changes, the switch variable is complemented.

4.7.3.5 Subroutine BMULT

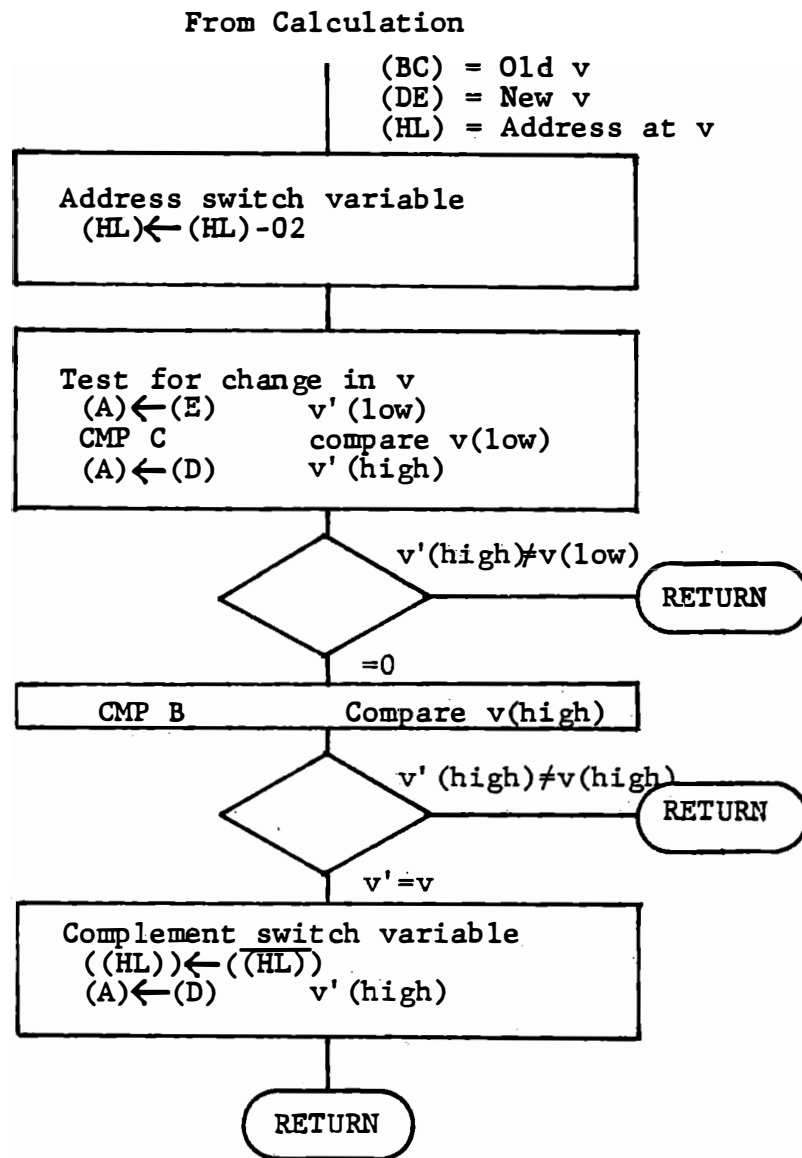
Both V_s and v are positive values that can range from 0000 to FFFF. $V_s - v$ can range from -FFFF to FFFF. The two byte subtract gives these values with CY set if the result is negative.

For example:

V_s	v	(CY)	(DE)	$V_s - v$
FFFF	0000	0	FFFF	+FFFF
FFFF	FFFE	0	0001	+0001
0000	0001	1	FFFF	-0001
0000	FFFF	1	0001	-FFFF

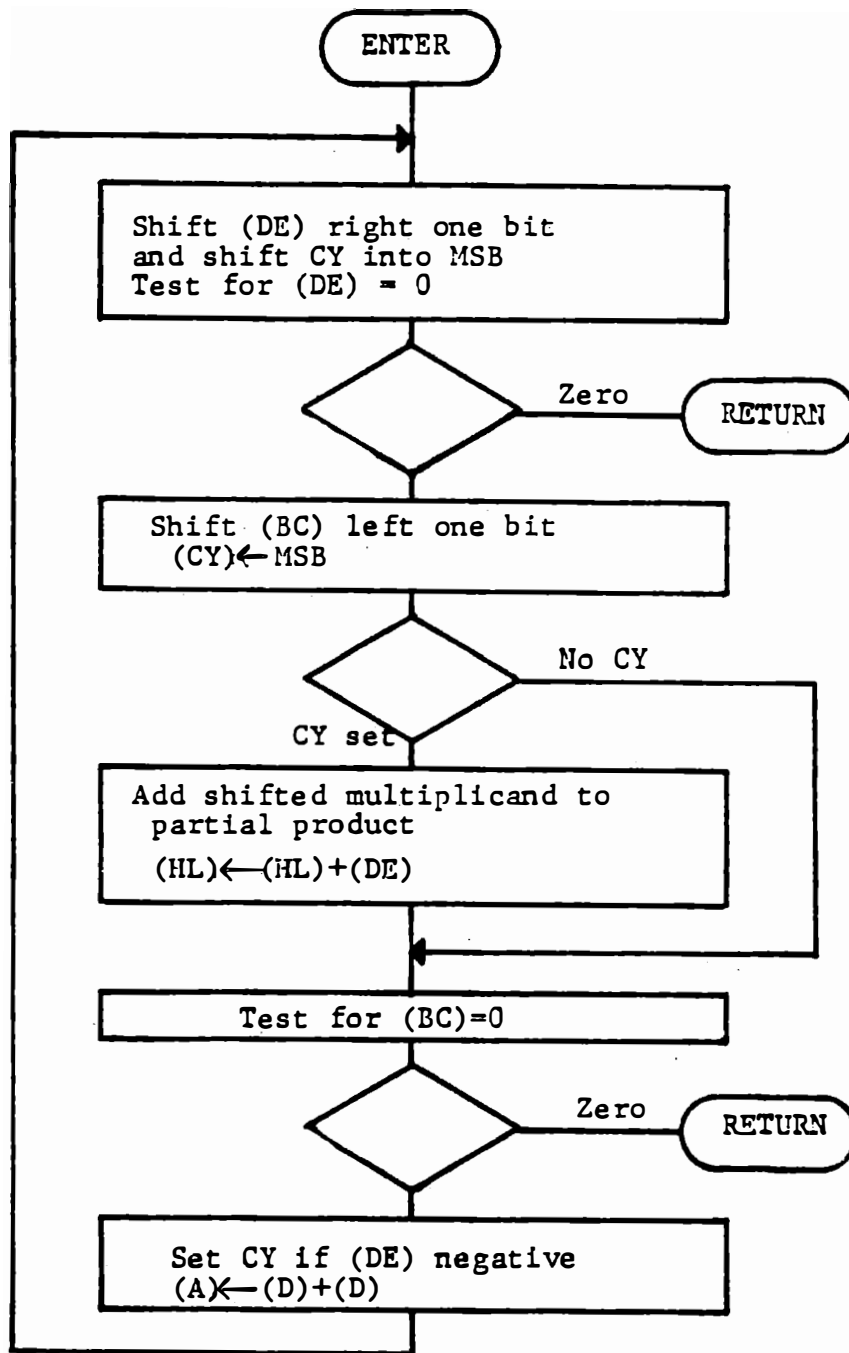
Thus (CY) and (DE) represent a 17 bit twos complement value.

We will design BMULT (Figure 4-42) to accept the data in this form and perform a correct multiplication with a positive or negative value in DE as marked by the CY flag. The resulting value of v' (in HL) will always be positive, since V_s , V , and t/RC are all positive. The value of v' will be greater or less than v , depending on the sign of $V_s - v$.



EXPV - Test for Change in Voltage

Figure 4-41b



Returns $(HL) \leftarrow (HL) + (DE)(BC)$

Subroutine BMULT

Figure 4-42

DIGITAL TO ANALOG OUTPUT

The multiplication will be done by repeatedly shifting the multiplicand (DE) right and the multiplier (BC) left. If the bit shifted out of the multiplier is a one, add the shifted multiplicand to the partial product (HL).

The first shift of the multiplicand will enter the carry bit into the high bit of the multiplicand, so that the shifted value will retain a proper twos complement form. At each loop the carry must be restored to its original state by copying the high bit from (D) into the carry. The table below shows successive values of the multiplicand for the two cases where it was initially +4000 or -4000 (represented as C000 with carry set).

CY Clear (+)	CY Set (-)
4000	C000
2000	E000
1000	F000
0800	F800
0400	FC00
0200	FE00
0100	FF00
0080	FF80

There is no rounding of the value as it is shifted. It is not necessary for the precision required, and any simple rounding procedure will lead to erroneous results.

DIGITAL TO ANALOG OUTPUT

4.7.3.6 Implementing the Program

Much of the exponential program is a modification of the previous program. The memory locations suggested in Section 4.7.2.6 allow room for these added functions, with BMULT and EXPV located at 8100 and 8120 respectively.

To debug EXPV and BMULT it is probably easiest to avoid the main loop and interrupt service altogether. Instead of running from 8200, use the debugging program shown as the first pages of Figure 4-43.

You can enter a value for v through the keyboard. With no data entered the previous value is recovered. Then you can step through the subroutine. When you are satisfied that program flow is correct and stack usage is balanced, repeatedly press NEXT, and successive values will be generated and displayed. Breakpoints can also be used in BMULT or EXPV. Note that the multiplier t/RC has been set to 50 (01010000); this is a nice value for testing BMULT because the multiplication process can be observed during the first four bits, but will terminate quickly. It reaches the switch point fairly quickly (32 iterations) which is also convenient.

When you are satisfied with your subroutines restore the operating program, but with the debugging modifications suggested in Section 4.7.2.7. Check the program flow. Remove the debugging changes and run the full program in AUTO mode.

4.7.3.7 Program Operation

Initialization sets $\Delta t = 40$ and $\Delta t/RC = 01$. No value is entered for V_s , but your debugging may have left the value FF in V_s .

The numeric display, the LED's, and the voltmeter will show the exponential rising quickly at first and slowing until it seems to stop at FD. Now only the less significant byte is changing. Suddenly the voltage will drop, and then approach zero very slowly.

With the ratio $\Delta t/RC$ fixed, an increase in the time interval Δt (entered with RUN) will also represent an increase in the time constant, so charging will take the same number of steps, but more time. An increase in $\Delta t/RC$ (entered with STEP) will result in larger and fewer steps to reach the source voltage, and therefore less time. If Δt and $\Delta t/RC$ are both increased proportionately, the time constant will remain constant and a coarser approximation of the same waveform will be obtained. If an oscilloscope is available this will be interesting to observe. The total cycle time will not be constant for corresponding values of Δt and $\Delta t/RC$ because the slow final approach to the source voltage is not calculated precisely.

DIGITAL TO ANALOG OUTPUT

Restore the original values for Δt and $\Delta t/RC$ and enter a lower value for V_s .

40, RUN

01, STEP

80, MEM

Now the exponential will rise only to half scale, taking the same time as before, and then reverse.

TEST PROGRAM FOR EXPV

	A	D	D	R	CODE													
CODING SHEET	8	00	0		21		L	X	I	H,	FF	50						
			1		50													
			2		FF													
			3		22		S	H	L	D		8391					(8391) ← 50 low Δt/RC	
			4		91													(8392) ← FF low switches
			5		83													
			6		21		L	X	I	H,	00	FF						
			7		FF													
			8		00													(8393) ← FF low 1/2
			9		22		S	H	L	D		8393						(8394) ← 00 low n low
MICROCOMPUTER TRAINING SYSTEM		A			93													
			B		83													
			C		6C		M	O	V	L,	H						(HL) ← 0000 low n	
		800	D		CD		C	A	L	L	D	W	O	R	D		LOOP	
			E		D1												Display result	
			F		02													
		8	01	0		CD		C	A	L	L	E	N	T	W	2		Alternate entry to
			1			49												ENTWD preserved
			2			3												(HL) if no entries
			3			22		S	H	L	D		8394					store n low (L)
		4			94													
		5			83													
		6			7C		M	O	V	A,	H						(A) ← n high	
		7			21		L	X	I	H,	8391						Address Δt/RC	
		8			91													
		9			83													
INTEGRATED COMPUTER SYSTEMS		A			CD		C	A	L	L	E	X	P	V			Return (DE) - n	
			B		20													
			C			81												
			D			EB		X	C	H	G							
			E			C3		J	M	P		800D						
			F			0D												
		8	02	0		80												
			1															
			2															
			3															
		4																
		5																
		6																
		7																
		8																

Figure 4-43

FUNCTION GENERATOR - INITIALIZE

		A	D	D	R	CODE				
CODING SHEET	8	20	0	3E		MVI	A,	80		Program Ports Port B Out
			1	80						
			2	D3		OUT	CNT1			
			3	07						
			4	3E		MVI	A,	92		
			5	92						
			6	D3		OUT	CNT2			
			7	0F						
MICROCOMPUTER TRAINING SYSTEM	8			3E		MVI	A,	24		Program timer 0 Light output Mode 2 Binary
			9	24						
		A	D3		OUT	TIMCT				
			B	17						
			C	21		LXI	H,	0140		
			D	40						
			E	01						
			F	22		SHLD		8390		Initial values
		8	21	0	90					(L) ← Δt
				1	83					(8390) ← Δt
			2	AF		XRA	A		(8391) ← Δt	
			3	32		STA		8392	(8392) ← target	
			4	92					(will be changed by interrupt	
			5	83						
			6	D3		OUT	PORT	B	(D/A) ← 00	
			7	05						
			8	7D		MOV	A,	L		
			9	D3		OUT	TIM0		(TIM1) ← Δt	
			A	14						
INTEGRATED COMPUTER SYSTEMS			B	21	*	LXI	H,	8120		
			C	20	*					
			D	81	*					
			E	22		SHLD		8398		Store address of EXPV subroutine for interrupt service
			F	98						
		8	22	0	83					
			1	3E		MVI	A,	01	Enable timer 0 interrupt	
			2	01						
			3	D3		OUT	CNT2			
			4	0F						
			5	C3		JMP		8280	Jump to main loop	
			6	80						
			7	82						
			8							

Figure 4-44a

		A	D	D	R	CODE															
CODING SHEET	8	0																			
		1																			
		2																			
		3																			
		4																			
		5																			
		6																			
		7																			
MICROCOMPUTER TRAINING SYSTEM	8 22	8	F5			PUSH	PSW													TIMER 0	
		9	E5			PUSH	H													save registers	
		A	D5			PUSH	D														
		B	C5			PUSH	B														
		C	C3			JMP	8233														Jump past RST6
		D	33																		
		E	82																		
		F	00			NO P															
		8 23	0	E7			RST 4														If RST6 occurs
			1	FB			EI														enter monitor
INTEGRATED COMPUTER SYSTEMS		2	C9			RET															
	8 23	3	21			LXI	H, EXIT														Push EXIT
		4	44																		address for
		5	82																		return from
		6	E5			PUSH	H														function module
		7	2A			LHL	FUNC														Push function
		8	98																		address for
		9	83																		dispatch to
		A	E5			PUSH	H														function module
		B	21			LXI	H, 8391														(HL) ← address
INTEGRATED COMPUTER SYSTEMS		C	91																		of ΔN for
		D	83																		function module
		E	DB			IN	PORT 1 B														(A) ← present
		F	05																		voltage
	8 24	0	C9			RET															Dispatch to
		1	00			NO P															function module
		2	00			NO P															
		3	00			NO P															
	4																				
	5																				
	6																				
	7																				
	8																				Figure 4-44b

FUNCTION GENERATOR - EXIT FROM INTERRUPT

	A	D	D	R	CODE										
CODING SHEET	8				0										
					1										
					2										
					3										
		824			4	D3	OUT	PORT	B						EXIT
					5	05									<i>Output new voltage</i>
					6	3E	MVI	A,	01						<i>Enable and clear</i>
					7	01									<i>Times 0 interrupt</i>
					8	D3	OUT	CNT	2						
					9	0F									
MICROCOMPUTER TRAINING SYSTEM				A	C1	POP	B								
				B	D1	POP	D								
				C	E1	POP	H								
				D	F1	POP	PSW								
				E	FB	EI									
				F	C9	RET									
		8			0										
					1										
					2										
					3										
				4											
				5											
				6											
				7											
				8											
INTEGRATED COMPUTER SYSTEMS				A											
				B											
				C											
				D											
				E											
				F											
		8			0										
					1										
					2										
					3										
				4											
				5											
				6											
				7											
				8											

Figure 4-44c

A D D R		CODE	FUNCTION TRIWV				
CODING SHEET	8 25	0 46	MOV	B	M		$(B) \leftarrow \Delta V$
	1	23	INX	H			Address target
	2	BE	CMP	M			Compare voltage
	3	D2	JNC	8	260		If voltage \geq target
	4	60					go to subtract
	5	82					
	6	80	ADD	B			$(A) \leftarrow V' - V + \Delta V$
	7	DA	JC	8	266		If beyond FF go
	8	66					to reverse
MICROCOMPUTER TRAINING SYSTEM	A	BE	CMP	M			Compare new
	B	D2	JNC	8	266		voltage to target
	C	66					If past target
	D	82					go to reverse
	E	C9	RET				Exit if not past
	F	00	NOP				
	8 26	0 90	SUB	B			$(A) \leftarrow V' = V - \Delta V$
	1	DA	JC	8	266		If below 00 go
	2	66					to reverse
INTEGRATED COMPUTER SYSTEMS	3	82					Compare new
	4	BE	CMP	M			voltage to target
	5	DD	RNC				Exit if not past
	8 26	6 7E	MOV	A	M		Reverse
	7	2F	CMA				Complement target
	8	77	MOV	M	A		voltage
	9	2F	CMA				Return with
	A	C9	RET				$V = \text{old target}$
	B						
C							
D			ENTER	WITH			
E			(A)	=	PRESENT VOLTAGE		
F			((HL))	=	INCREMENT, ΔV		
8	0		((HL) + 1)	=	TARGET VOLTAGE		
1							
2			RETURN				
3			(A)	=	NEW VOLTAGE		
4			(HL)	ADDRESSING	TARGET		
5			AT	OVERFLOW	RETURN		
6			(A)	=	OLD TARGET		
7			TARGET	COMPLEMENTED			
8						Figure 4-44d	

		A	D	D	R	CODE	FUNCTION - MAIN LOOP, DISPATCH											
CODING SHEET	8	28	0	FB		EI											Enable after key processing	
			1	2A		LHLD											(L) ← Δt	
			2	90													(H) ← ΔN	
			3	83														
			4	CD		CALL											DWORD	
			5	D1														
MICROCOMPUTER TRAINING SYSTEM		828	7	DB		IN											PORTIB	
			8	05													Read output voltage)	
			9	CD		CALL											DISPL	
			A	EB														Display voltages
			B	82														
			C	CD		CALL												SCAN
			D	57														Test keyboard)
			E	02														
			F	D2		JNC												8287
		8	29	0	87													Loop if no key
INTEGRATED COMPUTER SYSTEMS			1	82														
			2	CD		CALL												ENTBY
			3	36														(A) ← command
			4	03														(L) ← data
			5	5D		MOV												(D) ← key key count
			6	21		LXI												(E) ← data
			7	80														8280
			8	82														Push address of display function for return from process module.
			9	E5		PUSH												H
			A	C6		ADI												95
		B	95															
		C	6F		MOV													L, A
		D	6E		MOV													L, M
		E	E5		PUSH													H
		F	7B		MOV													A, E
	8	2A	0	21		LXI												H, 8391
			1	91														Address of voltage increment
			2	83														
			3	F3		DI												
			4	C9		RET												
			5															Jump to process module
			6															
			7															
			8															

Figure 4-44e

		A	D	D	R	CODE	FUNCTION - KEY PROCESSING														
CODING SHEET	8	0																			
		1																			
		2																			
		3																			
		4																			
MICROCOMPUTER TRAINING SYSTEM	82A	5	D	1	*	M	E	M												Set exponential	
		6	C	A		R	E	G													Set triangular
		7	A	4		A	D	D	R												
		8	B	0		S	T	E	P												Set Δv
		9	A	D		R	U	N													Set Δt
		A	C	0		N	E	X	T												Set v reverse target
		B	B	2		B	R	K													Set fixed voltage
		C	B	4		C	L	R													Set voltage
		82A	D	D	3	O	U	T	T	I	M	0									RUN Set Δt
		E	1	4																	Load timer 0
		F	2	B		D	C	X	H												Address Δt
		82B	0	7	7	M	O	V	M	,	A										STEP Set Δv
		1	C	9		R	E	T													
		82B	2	3	6	M	V	I	M	,	0	0									BRK set fixed v
		3	0	0																	Clear Δv
	82B	4	D	3	O	U	T	P	O	R	T	1	B							CLR Set v	
	5	0	5																		
	6	2	3	*	I	N	X	H												Address low byte	
	7	2	3	*	I	N	X	H												of v at 8392	
	8	2	3	*	I	N	X	H													
	9	3	6	*	M	V	I	M	,	0	0										
INTEGRATED COMPUTER SYSTEMS	A	0	0	*																	
	B	C	9	*	R	E	T														
	C																				
	D																				
	E				*	C	H	A	N	G	E	D	F	O	R	E	X	P	O	N	E
	F																				
	8	0																			
		1																			

Figure 4-44f

FUNCTION - KEY PROCESSING (continued)

		A	D	D	R	CODE																	
CODING SHEET	8	2C	0	2	3														NEXT set Target				
			1	1	4														Test for data entered				
			2	1	5																		
			3	C	2															If data entered			
			4	C	8															Jump to store as			
			5	8	2															target voltage			
			6	7	E															Elec complement			
			7	2	F															old target			
		8	2C	8	7	7														Store Target			
MICROCOMPUTER TRAINING SYSTEM	9		C	9																RET			
		8	2C	A	2	1	*													LXI			
				B	5	0															H, 8250		
				C	8	2															(HL) ← Address TRIWV		
			8	2C	D	2	2														SHLD		
					E	9	8															8398	
					F	8	3															Store function	
			8	2D	0	C	9															address for	
			8	2D	1	2	3	*															interrupt service
INTEGRATED COMPUTER SYSTEMS			2	2	3																		
			2	2	3																		
			2	7	7																		
			2	2	1																		
			2	2	0																		
			2	8	1																		
			2	C	3																		
			2	C	D																		
			2	8	2																		
		A																					
		B																					
		C				*																	
		D																					
		E																					
		F																					
	8	0																					
		1																					
		2																					
		3																					
		4																					
		5																					
		6																					
		7																					
		8																					

Figure 4-44a

		A	D	D	R	CODE	DISPLAY VOLTAGE SUBROUTINE															
CODING SHEET	8	0																				
		1																				
		2																				
		3																				
		4																				
		5																				
		6																				
		7																				
		8																				
		9																				
MICROCOMPUTER TRAINING SYSTEM	A																					
	82E	B	CD			CALL														Display voltage		
		C	95																			
		D	02																			
		E	79			MOV	A,	C												(A) ← voltage		
		F	21			LXI	H,	00FF												To shift ones		
	82F	0	FF																	into (H)		
		1	00																		Divide by 29 to	
	82F	2	D6			SUI		D													divide voltage range	
		3	1D																		into 9 segments	
	4	DA			JC		82FB													when remainder		
	5	FB																		less than 0		
	6	82																		go to display		
	7	29			DAD		H													Shift in ones		
	8	C3			JMP		82F2													Long		
	9	F2																				
INTEGRATED COMPUTER SYSTEMS	A	82																				
	82F	B	7C			MOV	A,	H													Display result	
		C	D3			OUT		PORT	1A												in LED's	
		D	04																			
		E	C9			RET																
		F	00			NOP																
	8	0																				
		1																				
		2																				
		3																				
	4																					
	5																					
	6																					
	7																					
	8																				Figure 4-44h	

		A	D	D	R	CODE	FUNCTION	SUBROUTINE	EXPV	
CODING SHEET	8	12	0	5	7	MOV	D, A			(D) ← n (high)
			1	4	6	MOV	B, M			(B) ← Δt/RC
			2	2	3	INX	H			Address switch
			3	7	E	MOV	A, M			Set CY if 280
			4	1	7	PAL				To increase voltage
			5	3	E	MUL	A, 00			
			6	0	0					
			7	2	3	INX	H			Address V _s
			8	4	E	MOV	C, M			(C) ← V _s
			9	D	A	JC	8' 2 D			Jump if voltage increasing
		A		2	D					
	MICROCOMPUTER TRAINING SYSTEM		B		8	1				
		C		4	F	MOV	C, A			clear V _s
		8	12	D	2	3	INX	H		Address n (low)
			E		5	E	MOV	E, M		(E) ← n (low)
			F		D	5	PUSH	H, D		save n
		8	13	0	E	5	PUSH	H		save address n
			1		E	B	XCHG			(HL) ← n
			2		9	5	SUB	L		
			3		5	F	MOV	E, A		
			4		7	9	MOV	A, C		(DE) ← V _s - n
			5		9	C	SBB	H		
INTEGRATED COMPUTER SYSTEMS				6		5	7	MOV	D, A	
			7		0	E	MVI	C, 000		(BC) ← Δt/RC
			8		0	0				as two bytes
			9		C	D	CALL	B MULT		(HL) ← (HL) + (DE)(BC)
		A		0	7					n' = n + (V _s - n) Δt/RC
			B		8	1				
			C		E	B	XCHG			(DE) ← n'
			D		E	1	POP	H		Address n (low)
			E		7	3	MOV	M, E		Store n' (low)
			F		C	1	POP	B		(BC) ← old n
		8		0			CONTINUED			
				1			ENTER	(A) =		n (high bytes)
			2			RETURN	(A) =		n' (high bytes)	
			3			AT ENTRY				
			4			((HL))			= Δt/RC	
			5			((HL) + 1)			=	
			6			((HL) + 2)			= Source voltage	
			7			((HL) + 3)			= n (low bytes)	
			8							

FIGURE 4-46a

EXPV - SWITCH WHEN V UNCHANGED

		A	D	D	R	CODE													
CODING SHEET	8	14	0	2	B		D	C	X	H							Address switched		
			1	2	B		D	C	X	H									
			2	7	B		M	O	V	A	,	E					Compare v' (low)		
			3	B	9		C	M	P	C	,							with old v (low)	
			4	7	A		M	O	V	A	,	D						$(A) \leftarrow v'$ (high)	
			5	C	0		R	N	Z									Exit if v changed	
			6	B	8		C	M	P	B								Compare v (high)	
			7	C	0		R	N	Z									Exit if v changed	
			8	7	E		M	O	V	A	,	M						Complement	
			9	2	F		C	M	A									switched variable	
MICROCOMPUTER TRAINING SYSTEM	A	7	7			M	O	V	M	,	A								
	B	7	A			M	O	V	A	,	D						$(A) \leftarrow v'$ (high)		
	C	C	9			R	E	T											
	D																		
	E																		
	F																		
	8	0																	
		1																	
		2																	
		3																	
INTEGRATED COMPUTER SYSTEMS		4																	
		5																	
		6																	
		7																	
		8																	
		0																	
		1																	
		2																	

Figure 4-46b

This page intentionally left blank

MICROCOMPUTER INTERFACING WORKBOOK

CHAPTER 5

ANALOG TO DIGITAL INPUT

This page intentionally left blank.

5. ANALOG TO DIGITAL INPUT

Microprocessors in instruments and control systems generally require input of analog signals, which must be converted to digital form for processing. Variables generated by sensors are likely to be any of the following:

Frequency or Pulse Interval (tachometers, flow meters and other motion sensors, and instrumentation electronics used for conversion of other variables)

Pulse Width (instrumentation electronics - not common)

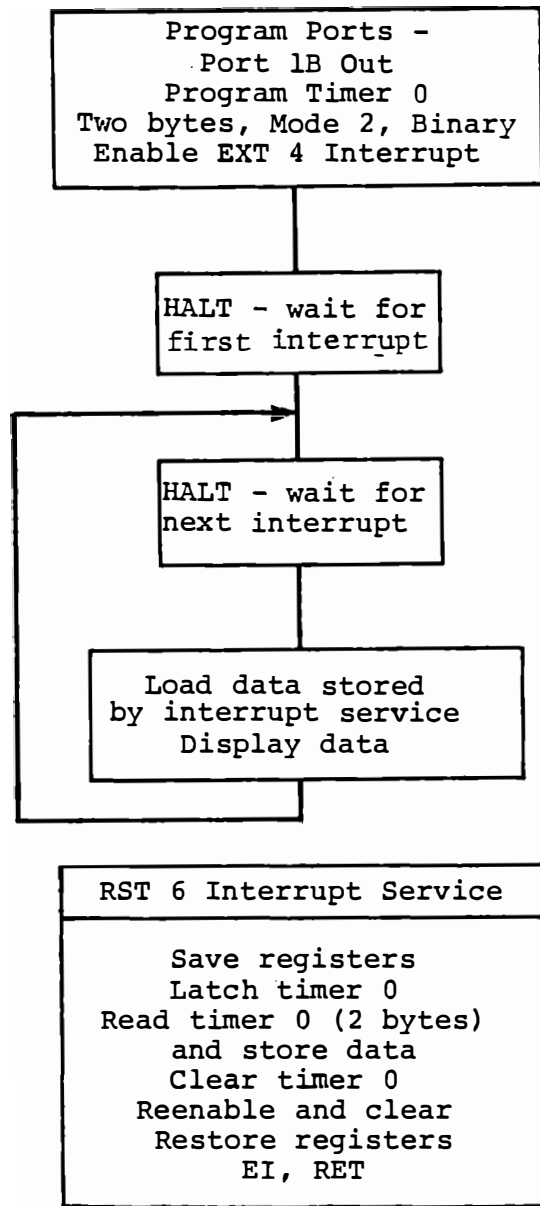
Resistance (thermistors, strain gauges, position sensors)

Capacitance or Inductance (position sensors - usually converted to frequency)

Voltage or Current (thermocouples, photovoltaic cells, or conversions of variable resistance.)

In this chapter we will consider digital conversion of pulse interval, frequency, voltage, and resistance. In Section 3.6, we measured a pulse width. That section should be reviewed, and especially Section 3.6.3 which discussed reading a timer while it is running.

ANALOG TO DIGITAL INPUT



Pulse Interval Measurement

Figure 5-1

5.1 PULSE INTERVAL MEASUREMENT

An external input of the ITS (either EXT4 or EXT5) with its edge triggered latch makes it easy to measure a pulse interval. To avoid noise triggering of the flip flop the signal should be coupled through an optical coupler (see page 1-18). Connect the AUDIO OUT test point near the upper right corner of the AMTS to OPTO IN, and connect OPTO OUT to EXT4 IN.

5.1.1 Measuring a Steady Signal

EXERCISE

When a rising edge occurs at the EXT 4 input, it will set the latch and generate an RST6 interrupt. An interrupt service routine will read and restart a timer, to measure pulse interval.

The main program displays the data read from the timer by interrupt service, and loops to a Halt instruction. This stops program execution until another interrupt has occurred. Execution of the main program resumes at the location following HLT after the interrupt has been serviced. We ignore the first measurement because it is meaningless. The counter was started when power came on, not at an edge of the signal. Thereafter, we load and display the measured data after each interrupt. Since the 8253 counts down from zero this value is the twos complement of the number of clocks in one modem cycle (see page 3-10). A flow chart is shown in Figure 5-1, and a coding solution in Figure 5-2.

PULSE INTERVAL-MAIN

		A	D	D	R	CODE				
CODING SHEET	8	20	0	3E		MVI	A,	80		Program Ports
			1	80						Port 1B Out
			2	D3		OUT	CNT	1		
			3	07						
			4	3E		MVI	A,	92		
			5	92						
			6	D3		OUT	CNT	2		
			7	0F						
			8	3E		MVI	A,	30		Program Timer 0
			9	30						Two bytes
MICROCOMPUTER TRAINING SYSTEM	A		D3		OUT	TIM	CT			Mode, Binary
	B		17							
	C		3E		MVI	A,	09		Enable EXT4	
	D		09						interrupt	
	E		D3		OUT	CNT	2			
	F		0F							
	8	21	0	76		HLT				Ignore first inter.
		821	1	76		HLT				LOOP
			2	2A		LHLD	D	8300		Load measured
			3	00						interval
INTEGRATED COMPUTER SYSTEMS			4	83						
			5	CD		CALL	DWORD	D		Display interval
			6	D1						
			7	02						
			8	C3		JMP	8211			Loop
			9	11						
		A		82						
		B								
		C								
		D								
	E									
	F									
	8		0							
			1							
			2							
			3							
			4							
			5							
			6							
			7							
			8							

Figure 5-2a

PULSE INTERVAL - INTERRUPT SERVICE

A D D R		CODE										
CODING SHEET	8	23	0	F5	PUSH	H	PSW					
			1	E5	PUSH	H	H					
			2	21	LXI	H	8300				Address data	
			3	00							storage location	
			4	83								
			5	3E	MVI	A	00				Latch Timer 0	
			6	00								
			7	D3	OUT	TIMCT						
			8	17								
			9	DB	IN	TIM0					Read Timer 0	
MICROCOMPUTER TRAINING SYSTEM	A	14									and store data	
	B	77		MOV	M	A						
	C	23		INX	H							
	D	DB		IN	TIM0							
	E	14										
	F	77		MOV	M	A						
	INTEGRATED COMPUTER SYSTEMS	8	24	0	AF	XRA	A					Clear Timer 0
				1	D3	OUT	TIM0					
				2	14							
				3	D3	OUT	TIM0					
			4	14								
			5	3E	MVI	A	09				Reenable and	
			6	09							clear EXT 4	
			7	D3	OUT	CNT2						
			8	0F								
			9	E1	POP	H						
	A	F1		POP	PSW							
	B	FB		EI								
	C	C9		RET								
	D											
	E											
	F											
	8	0										
		1										
		2										
		3										
		4										
		5										
		6										
		7										
		8										

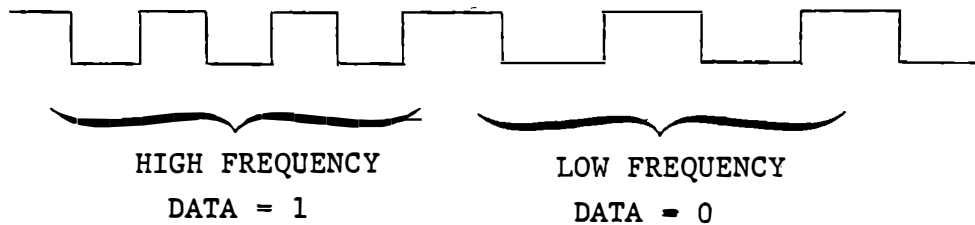
Figure 5-2b

ANALOG TO DIGITAL INPUT

5.1.2. Measuring a Multi-Valued Interval

EXERCISE

When the cassette modem is actually in use its pulse interval is switched between two values, depending on the data being recorded.

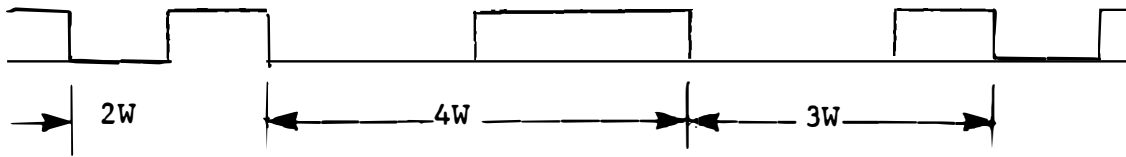


We can compare the measured interval to some threshold value to decide which frequency is present. We will use the monitor's tape recording program SEROT to generate a data train to the cassette modem, and develop an interrupt service routine to measure the intervals, decide which frequency is present, and measure an average interval for one frequency.

The low frequency is half of the high frequency, and the signal is a square wave, so if we define W as the width of a high frequency pulse, the interval for the high frequency is $2W$ and the low frequency interval is $4W$. If the bit time is constant the number of $2W$ intervals for a data bit equal to one should be twice the number of $4W$ intervals for a zero. This ratio is severely distorted however, by the interrupt service routine which interferes with the bit timing loop in SEROT. Therefore the number of cycles of each frequency is meaningless during this test. The measurement of pulse intervals is valid, however, because the recording frequency is

generated by the modem hardware, not by the program.

If we use a single threshold, as suggested above, an uncertainty arises in the measurements because an intermediate pulse interval can also occur. The modem changes its interval after its input data from the processor changes, at the moment when its output changes from high to low or from low to high.

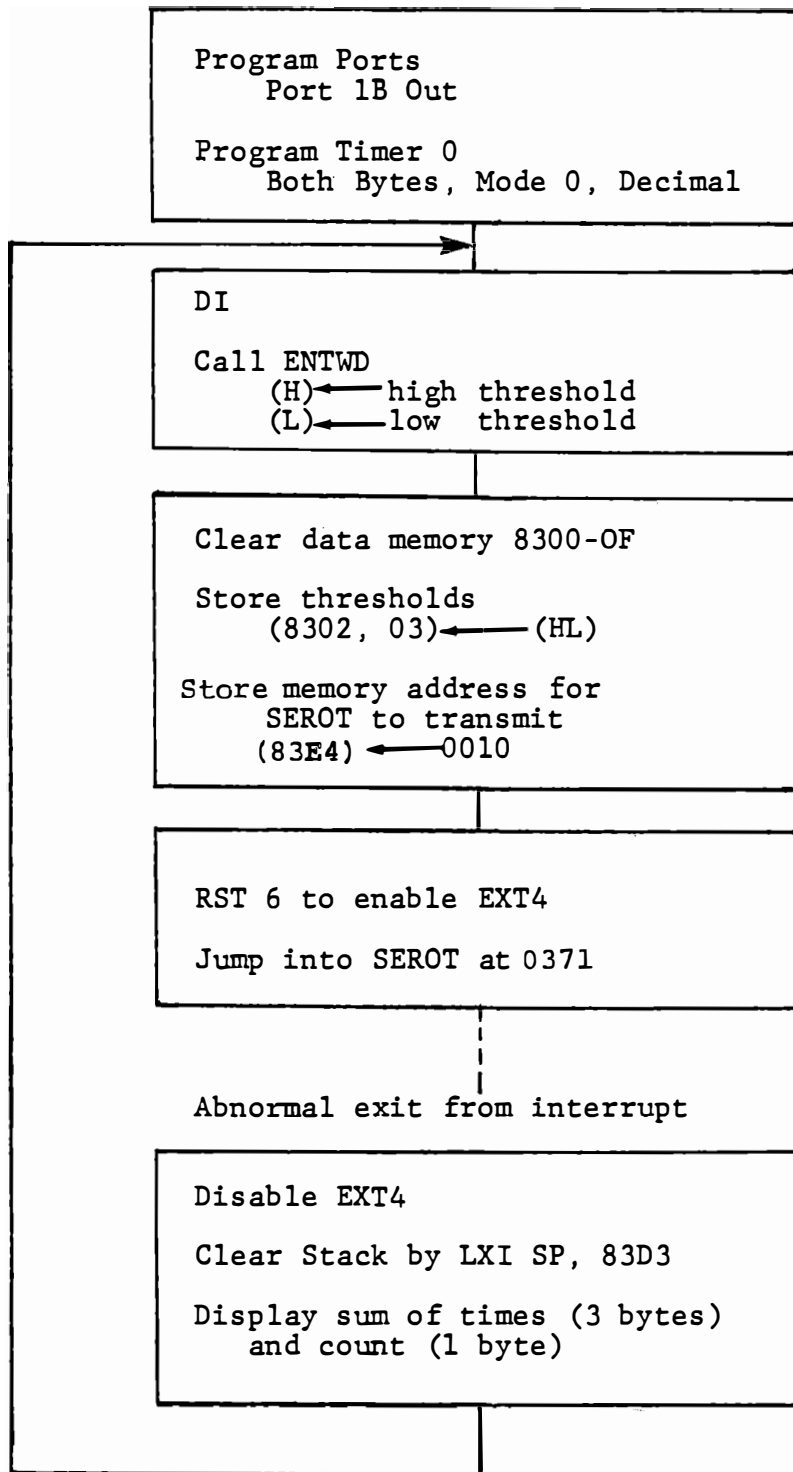


The intermediate value $3W$ occurs much less frequently than the other values because it can only occur at a bit boundary, and even there it has only a 25% probability. It will be detected, however, and our program should provide for it.

5.1.2.1 Program Design

The program will accept (through keyboard entry) two different thresholds. The modem output will be sensed as before by the EXT4 interrupt. At each interrupt the preceding time interval will be measured and compared with the thresholds. If it lies between them the interval will be added into a sum, and counted. If the interval is less than the lower threshold or more than the higher, the time will be discarded. All intervals will be counted (separately from the count of those between thresholds).

ANALOG TO DIGITAL INPUT



Multi-Valued Interval-Main

Figure 5-3a

The program will be stopped either when 100 values have been summed or when 65,536 interrupts have occurred. The latter stop will indicate few or no measurements have occurred between the thresholds. If the two thresholds have been selected to include one of the 2W, 3W, or 4W intervals the program will be stopped after 100 occurrences. By varying the thresholds we can measure the average interval for each nominal value. For ease of interpretation the interval measurement and summing will be in decimal.

5.1.2.2 Main Program

The main program is depicted in Figure 5-3a. Timer 0 is used again to measure the interval, but now it is programmed for decimal counting. After the two thresholds have been entered the main program clears the data memory and stores the thresholds. RST6 is used to call the interrupt service, which will start the timer and enable the EXT4 interrupt. (This leads to some invalid measurements which will be discarded by interrupt service). Now the reserved memory locations 83E4,E5 are loaded with a convenient address for the start of data transmission by SEROT. Any address can be used, except that unless quite frequent alternations of ones and zeros are present there will be very few 3W intervals. A starting address of 0010 works nicely.

A jump into SEROT causes the monitor to start transmitting data to the modem. SEROT starts at 0371.

ANALOG TO DIGITAL INPUT

This page intentionally left blank

5.1.2.3 Abnormal Exit

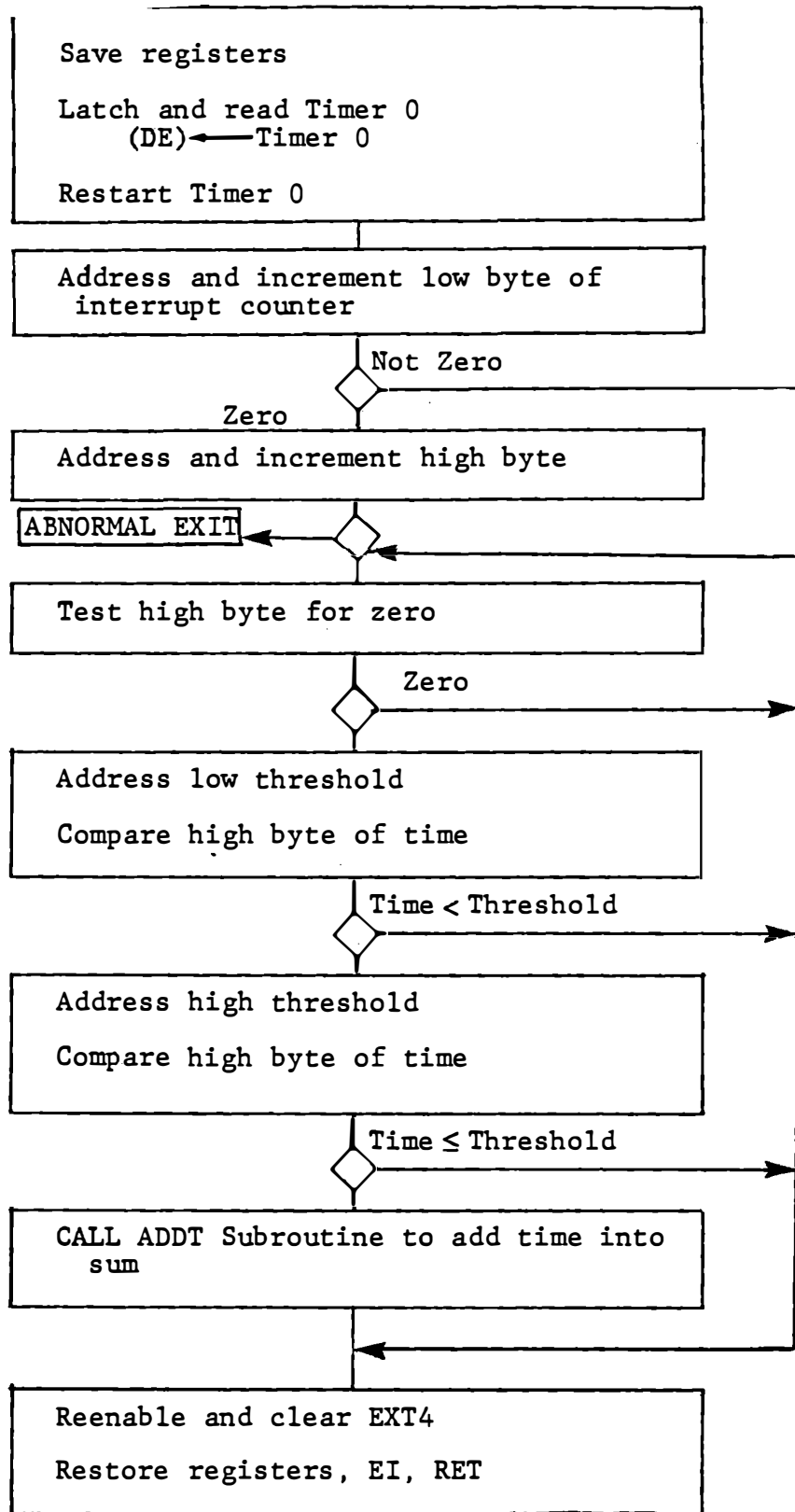
When interrupt service for EXT4 reaches a stopping point, after 100 measured intervals or 65,536 total intervals, it makes an abnormal exit. Instead of restoring registers and returning to SEROT, the abnormal exit clears the stack and displays the measured data.

Previously we have sometimes made abnormal exits from subroutines, clearing the stack by popping the return address into a register pair. Here we cannot use that method because SEROT uses several nested subroutines and also uses the stack to save registers, and we do not know how many levels of the stack may be in use. In this program we clear the stack by reinitializing the stack pointer:

```
31      LXI      SP,83D3
      D3
      83
```

This loads the stack pointer with the same address normally loaded by the monitor. Although the stack contents have not been erased, this effectively discards the past history and gives a fresh start.

ANALOG TO DIGITAL INPUT



Multi-Valued Interval - Interrupt Service

Figure 5-3b

5.1.2.4 Interrupt Service

Figure 5-3b shows the interrupt service routine. After saving the registers we read and restart Timer 0. The two byte interrupt count is incremented. At 65536 interrupts the counter reaches zero and the abnormal exit is taken. To avoid recording invalid data caused by initialization both in the main program and in SEROT, we ignore the first 256 measurements, by testing the high byte of the count. The high byte of the timer data is compared to the two thresholds. Since the timer returns the hundreds complement of the interval time, the comparison is made not by CMP but by:

```

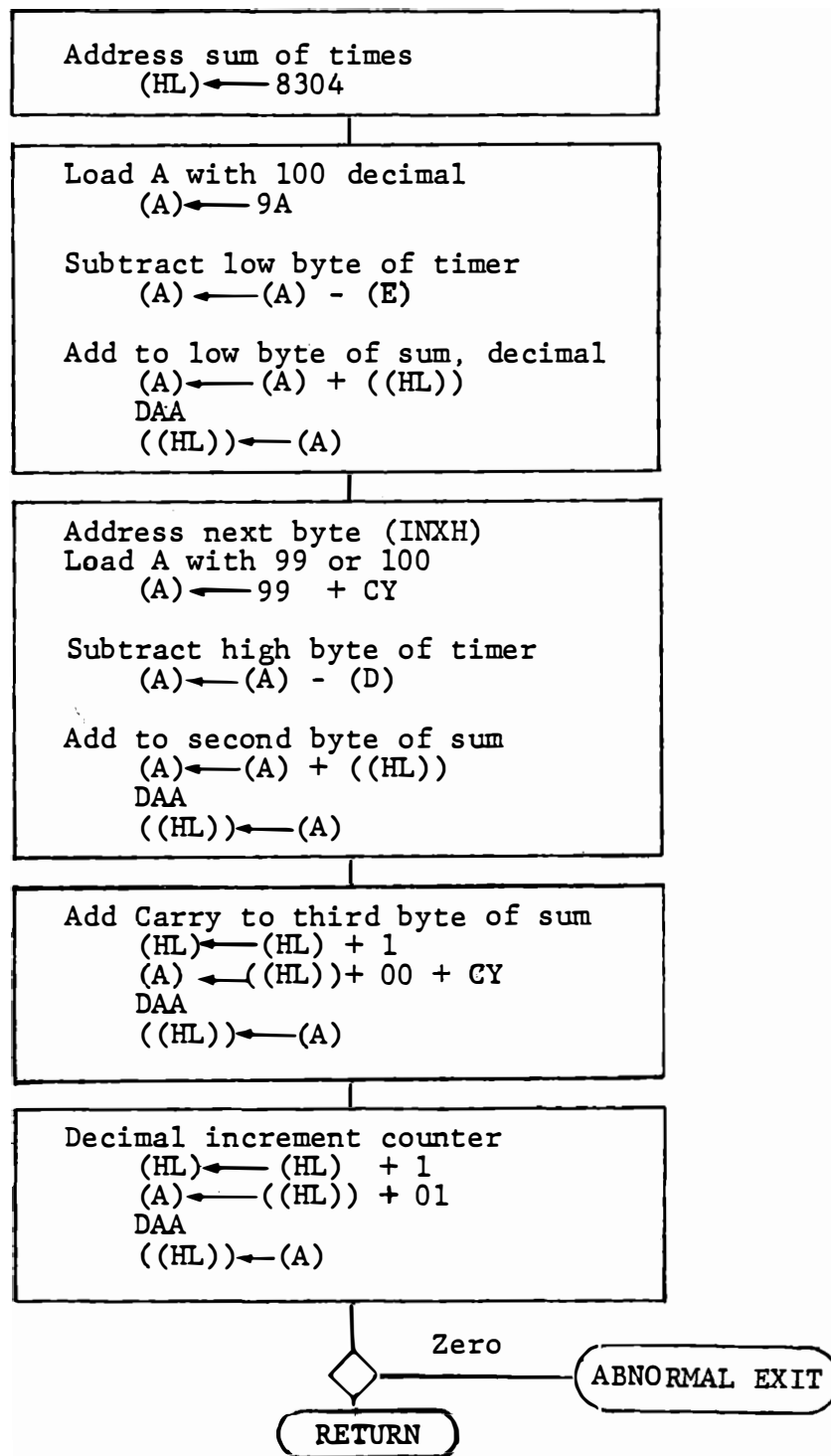
MOV    A,D    (A)←---high byte of time
ADD    M      Add threshold
DAA

```

This sets CY if the time is less than the threshold.

If the time lies between the two thresholds a subroutine is called to add the hundreds complement of the timer data into the sum, and to increment the decimal counter.

ANALOG TO DIGITAL INPUT



Add Decimal Time to Memory

Figure 5-3c

5.1.2.5 Decimal Addition Subroutine

Figure 5-3c shows the addition subroutine. Since the timer data represents the hundreds complement of the time interval we must complement it before adding it into the sum. Here we combine the two functions. The Intel 8080 (or the NEC 8080AF) does not allow DAA after subtraction, but only after addition.

```

MVI    A,9A    (A)<---100
SUB    E        Subtract low bytes
ADD    M        Add to low sum
DAA
MOV    M,A

```

Subtracting a decimal value from 9A or 99 cannot generate any carry. After adding the old sum, DAA will make the proper adjustment to a decimal value, generating a carry if the result exceeds 99. For the second byte we load A with 99 and add the carry from the first byte, so it contains either 99 or 9A (= 100 decimal).

```

INX    H        Address second byte
MVI    A,99    Load A with 99 or 100
ACI    00
SUB    D        Subtract timer
ADD    M        Add into old sum
DAA
MOV    M,A

```

If this generates a carry it must be added into the third byte.

ANALOG TO DIGITAL INPUT

Now the decimal counter is incremented and we return to increment the binary count unless the decimal counter reaches 100. Then the abnormal exit is taken. Note that the use of LXI SP permits the abnormal exit from any subroutine level without regard to the stack.

Write your program and test it. Section 5.1.2.6 describes the use of the program. Memory assignments in the solution given in Figure 5-4 are:

8300,01	Binary count
8302	Low threshold
8303	High threshold
8304,05,06	Sum of times
8307	Decimal count
83E4,E5	Starting address for transmission

MULTI-VALUED INTERVAL MEASUREMENT

	A	D	D	R	CODE						
CODING SHEET	8	20	0		3E	MVI	A,	80			Program Ports
			1		80						Port B Out
			2		D3	OUT		CNT1			
			3		07						
			4		3E	MVI	A,	92			
			5		92						
			6		D3	OUT		CNT2			
			7		0F						
			8		3E	MVI	A,	31			Program Timer 0
			9		31						Both buses
MICROCOMPUTER TRAINING SYSTEM		A			D3	OUT		TIMCT			Mode 0 ^d
			B		17						Decimal
			C		F3	DI					
			D		CD	CALL		ENTWD			Enter thresholds
			E		46						(H) ← high
			F		03						(L) ← low
		B	21	0		AF	XRA	A			Clear data memory
				1		11	LXI	D,	8310		8300-830F
				2		10					
				3		83					
INTEGRATED COMPUTER SYSTEMS		821	4		1D	DCR	E				
			5		12	STAX	D				
			6		C2	JNZ		8214			
			7		14						
			8		82						
			9		22	SHLD		8302			Store thresholds
			A		02						
			B		83						
			C		21	LXI	H,	0010			Address for start of transmission
			D		10						
		E			00						
			F		22	SHLD		83E4			Store address for SEROT
		B	22	0		E4					
				1		83					
				2		00	NOP				Use FI during delay
				3		F7	RST6				To enable EXTH
		822	4			C3	JMP		0371		Jump into SEROT
				5		71					
			6		03						
			7		00						
			8								Figure 5-4a

INTERRUPT SERVICE

		A	D	D	R	CODE							
CODING SHEET	8	23	0	F	5	PUSH	PSW						
			1	E	5	PUSH	H						
			2	D	5	PUSH	D						
			3	C	5	PUSH	B						
			4	3E	MVI	A, 00						Latch and read	
			5	00								Time 0	
			6	D3	OUT	TIMCT							
			7	17									
			8	DB	IN	TIM0							
			9	14									
MICROCOMPUTER TRAINING SYSTEM	A	5F		MOV	E, A								
	B	DB		IN	TIM0								
	C	14											
	D	57		MOV	D, A								
	E	AF		XRA	A							Restart timer 0	
	F	D3		OUT	TIM0								
	8	24	0	14									
			1	D3	OUT	TIM0							
			2	14									
			3	21	LXI	H, 8300							Address binary
		4	00									counter for	
		5	83									interrupts	
		6	34	INR	M							Increment low byte	
		7	23	INX	H							Address high byte	
		8	C2	JNZ	824F								
		9	4F										
INTEGRATED COMPUTER SYSTEMS	A	82											
	B	34		INR	M							At zero increment	
	C	CA		JZ	8290							high byte	
	D	90										At 0000 go to	
	E	82										abnormal exit	
	8	24	F	B6	ORA	M							
			0										Test for high byte
			1										greater than zero
			2										
			3										
		4											
		5											
		6											
		7											
		8											

Figure 5-4b

		A	D	D	R	CODE							
CODING SHEET	8 25	0	CA		JZ				8	2	6	1	Until high byte > 0 go to exit
		1	61										
		2	82										
		3	23		INX	H							Address low threshold
		4	7A		MOV	A, D							Add to complement of time interval
		5	86		ADD	M							
		6	27		DAA								
		7	DA		JC				8	2	6	1	Exit if interval less than lower threshold
		8	61										
		9	82										
MICROCOMPUTER TRAINING SYSTEM	A	23		INX	H								Test higher threshold
	B	7A		MOV	A, D								
	C	86		ADD	M								
	D	27		DAA									If time between thresholds add to sum of times
	E	DC		CC				8	2	7	0		
	F	70											
	8 26	0	82										
	8 26	1	3E		MVI	A, 09							Normal exit
		2	09										
		3	D3		OUT	CNT 2							
INTEGRATED COMPUTER SYSTEMS		4	0F										
		5	C1		POP	B							
		6	D1		POP	D							
		7	E1		POP	H							
		8	F1		POP	PSW							
		9	FB		EI								
		A	C9		RET								
		B											
		C											
		D											
	E												
	F												
	8	0											
	1												
	2												
	3												
	4												
	5												
	6												
	7												
	8												

Figure 5-4c

BETWEEN THRESHOLDS

		A	D	D	R	CODE												
CODING SHEET	8	27	0	21		L	X	I	H	,	8	3	0	4	Address sum of intervals			
			1	04														
			2	83														
			3	3E		M	V	I	A	,	9	A			Add hundred's complement of timer data to sum			
			4	9A														
			5	93		S	U	B	E									
			6	86		A	D	D	M									
			7	27		D	A	A										
			8	77		M	O	V	M	,	A							
			9	23		I	N	X	H									
MICROCOMPUTER TRAINING SYSTEM	A			3E		M	V	I	A	,	9	9						
	B			99														
	C			CE		A	C	I	0	0								
	D			00														
	E			92		S	U	B	D									
	F			86		A	D	D	M									
	8	28	0	27		D	A	A										
			1	77		M	O	V	M	,	A							
			2	23		I	N	X	H									
			3	7E		M	O	V	A	,	M							
		4	CE		A	C	I	0	0									
		5	00															
		6	27		D	A	A											
		7	77		M	O	V	M	,	A								
		8	23		I	N	X	H										
		9	7E		M	O	V	A	,	M								
INTEGRATED COMPUTER SYSTEMS	A			C6		A	D	I	0	1								
	B			01														
	C			27		D	A	A										
	D			77		M	O	V	M	,	A							
	E			C0		R	N	Z										
	F			00		N	O	P										
	8		0															
			1															
			2															
			3															
		4																
		5																
		6																
		7																
		8																

Figure 5-4d

ABNORMAL CALL

	A	D	D	R	CODE																
CODING SHEET	8	29	0		3E		M	V	I	A,	08									Disable EXT 4	
			1		08																
			2		D3		O	U	T	C	N	T	2								
			3		0F																
			4		31		L	X	I	S	P,	83	D	3							
			5		D3																Clear stack
			6		83																
			7		21		L	X	I	H,	83	0	4								Address sum of times
			8		04																
			9		83																
	A			11		L	X	I	D,	83	F	F								Address right hand digit	
	B			FF																	
	C			83																	
MICROCOMPUTER TRAINING SYSTEM	8	29	D		7E		M	O	V	A,	M									Display 4 bytes	
			E		CD		C	A	L	L	D	B	Y	2							
			F		98																
	8	2A	0		02																
		1		23		I	N	X	H												
		2		7B		M	O	V	A,	E										After four bytes DBY 2 returns	
		3		FE		C	P	I	F	8										(DE) = 83F7	
		4		F8																	
		5		D2		J	N	C	8	29	D									Loop until four bytes displayed	
		6		9D																	
		7		82																	
		8		C3		J	M	P	8	20	C									Jump to get new thresholds	
		9		0C																	
	A			82																	
INTEGRATED COMPUTER SYSTEMS	B																				
	C																				
	D																				
	E																				
	F																				
	8		0																		
			1																		
			2																		
		3																			
		4																			
		5																			
		6																			
		7																			
		8																			

Figure 5-4e

ANALOG TO DIGITAL INPUT

5.1.2.6 Program Debugging and Operation

The use of RST6 to call the interrupt service routine not only provides an easy way to start the timer and enable the EXT4 interrupt, but also allows you to step through interrupt service for debugging. You can step through with a normal return or force the abnormal exit by preloading the decimal counter with 99. If the thresholds are set at 99 and 00 then interrupt service will call the decimal addition subroutine for any timer data.

For meaningful results the program must be run in AUTO mode. Enter thresholds of 99 and 00 (9900,NEXT). Almost immediately a decimal value for the time is displayed as three bytes, with a count of 00.

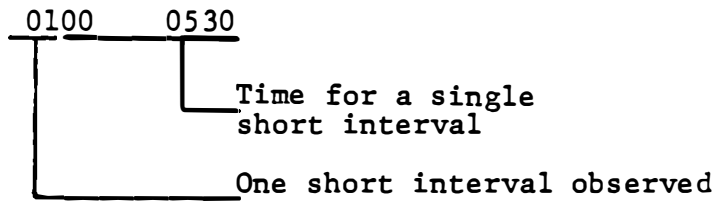
For instance:

```
0007 1332
┌───┬───┐
│   │   │
│   │   │
│   │   │
└───┴───┘
      TOTAL TIME FOR 100
      INTERVALS
      DECIMAL COUNT (=100)
```

If equal numbers of wide and narrow intervals were received this would be a central or average value. In fact SEROT generates enough leading high frequency intervals that this measurement includes no 3W or 4W intervals. Prove this by entering thresholds of 09 and 00 (0900,NEXT). A similar average will be obtained.

Try thresholds of 06 and 00. No intervals of fewer than 600 clocks should occur, so the program will run for 65,536 interrupts, and then display 000 000. Possibly one or a few short intervals will occur.

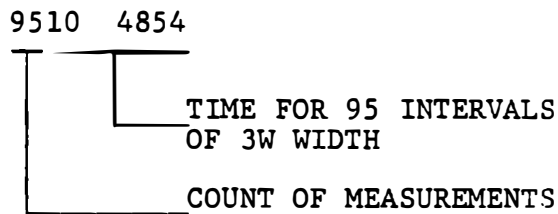
The display might show:



Try thresholds of 07 and 00. Depending on the time constant of the cassette modem oscillator there may be no intervals, a few, or many intervals in this range.

Now exclude the short intervals by entering thresholds of 99 and 09. The 2W intervals will be excluded and the sum of times will include mostly 4W intervals and possibly a few 3W intervals. The average will be close to the 4W interval. The lower threshold can be raised to exclude the 3W intervals. Try 99 and 13, which should include only the 4W intervals.

Finally, set thresholds to include only 3W intervals. Try 13 and 09 (1309, NEXT). There may, or may not, be 100 measurements made before the 65536 interrupts have been counted. In one experiment the result was:



The average time is $104854/95$ or 1104 clock times.

ANALOG TO DIGITAL INPUT

5.1.3 Measuring Received Pulse Intervals

EXERCISE

The program of 5.1.2 can also be used to measure the pulse intervals returned by the cassette recorder. Here we do not need (or want) the SEROT program running, so change the JMP 0371 instruction to a jump to itself. In the solution given in Figure 5-4:

```
8224    JMP    8224
```

Create a tape with a leader (all ones) of several seconds, or use one you already have. Connect the EXT 4 input to OPTO OUT, and connect OPTO IN to the AMTS test point AUDIO IN. Connect the cassette, start it in playback mode, and wait until you hear the steady tone from the leader. Now start the program as before; when it has received 356 one bits it will display the average pulse intervals. Compare these with those observed for recording. This gives a measure of the speed stability of your recorder.

5.2 FREQUENCY MEASUREMENT

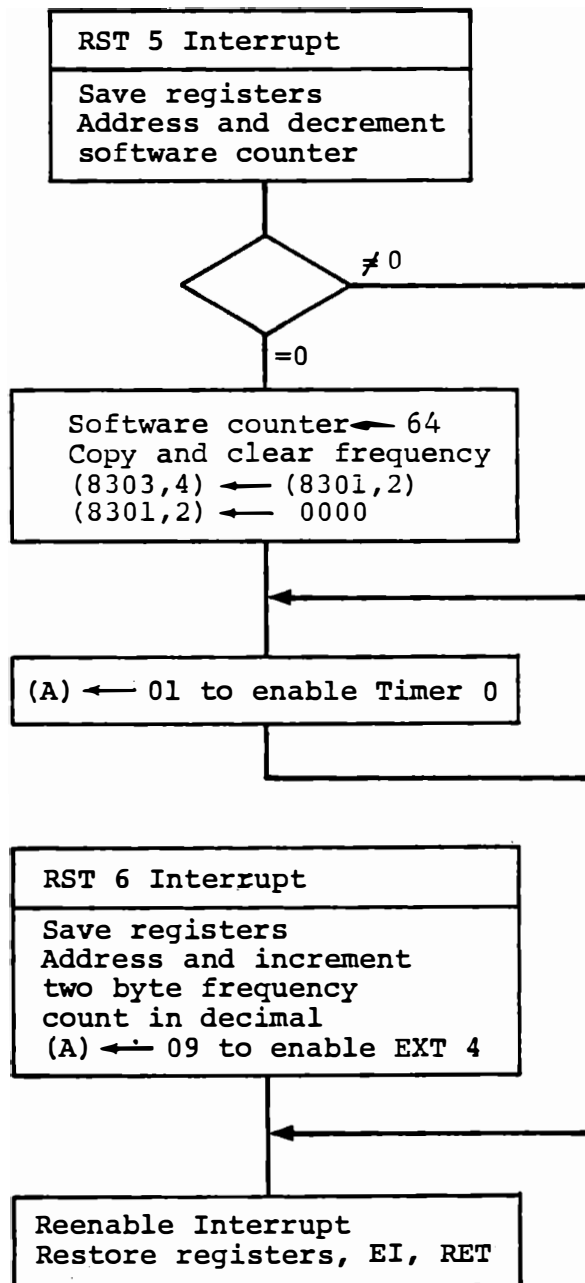
Clearly a frequency can be obtained from a pulse interval measurement by inverting the measured data. This gives the instantaneous frequency, which may be needed in some instances, especially when the rate of change is important. Often the rate of change is small compared to the frequency, and we can measure frequency by counting pulses over some period of time such as one or ten seconds. This method is used in the two following exercises.

5.2.1 Logic Level Frequency Measurements

EXERCISE

Measure the frequency of a logic level signal. Use EXT 4 (as in the preceding exercise) to detect the rising edge of the signal, and count the occurrences (in decimal). Use timer 0 with a software counter to measure one second intervals. The Timer 0 interrupt decrements a software counter, which starts at 64 to count 100D intervals of 10 milliseconds each. At zero, the counter is reloaded. The frequency count is copied to another pair of memory locations and the counter locations are cleared.

The main program does only initialization and display. Program port 2 and timer 0. Load timer 0 with 5000 for a 10 millisecond interrupt interval and enable EXT 4 and Timer 0 interrupts. Then repetitively load the copy of the frequency count and display it, as suggested in Figures 5-5 and 5-6.



Frequency Measurement - Interrupt

Figure 5-5

FREQUENCY MEASUREMENT

A D D R		CODE					
CODING SHEET	8 20	0 3E	MVI	A,	92		Program Port 2
		1 92					Asn Bin Count
		2 D3	OUT	CNT	2		
		3 0F					
		4 3E	MVI	A,	24		Program Timer 0
		5 94					high byte, Mode 2
		6 D3	OUT	TIMCT			
		7 17					
		8 3E	MVI	A,	50		Load timer 0
		9 50					for 10 millisecond
MICROCOMPUTER TRAINING SYSTEM	A	D3	OUT	TIMO			interrupt
	B	14					
	C	3E	MVI	A,	11		Enable Timer 0
	D	11					and EXT 4
	E	D3	OUT	PORT	2C		
	F	0E					
	8 21	0 2A	LHLD	8303			
		1 03					
		2 83					
		3 CD	CALL	DWORD			
	4 D1						
	5 02						
	6 C3	JMP	8210				
	7 10						
	8 82						
	9						
INTEGRATED COMPUTER SYSTEMS	A						
	B						
	C						
	D						
	E						
	F						
	8	0					
		1					
		2					
		3					
	4						
	5						
	6						
	7						
	8					Figure 5-6a	

FREQUENCY MEASUREMENT - INTERRUPTS

		A	D	D	R	CODE														
CODING SHEET	8	0																		
		1																		
		2																		
		3																		
		4																		
		5																		
		6																		
		7																		
MICROCOMPUTER TRAINING SYSTEM	8 22	8	F5			PUSH		PSW											RST5 - timer 0	
		9	E5			PUSH		H												
		A	21			LXI		H, 8300												
		B	00																	
		C	83																	
		D	C3			JMP		8250												
		E	50																	
		F	82																	
	INTEGRATED COMPUTER SYSTEMS	8 23	0	F5			PUSH		PSW											RST6 - EXT 4
			1	E5			PUSH		H											
		2	21			LXI		H, 8301											Address frequency	
		3	01																count	
		4	83																	
		5	37			STC													To add 1	
8 23		6	7E			MOV		A, M											(A) ← count	
		7	CE			ACI		00											Add carry into	
		8	00																count	
		9	27			DAA													Decimal	

Figure 5-6b

		A	D	D	R	CODE									
CODING SHEET	8	25	0	3	5		D	C	R	M			Decrement counter		
			1	3	E		M	V	I	A	,	0	1	to readable timer 0	
			2	0	1										
			3	C	2		J	N	Z	8	2	4	1	Exit unless	
			4	4	1									counter reached 00	
			5	8	2										
			6	D	5		P	U	S	H	D				
			7	3	6		M	V	I	M	,	6	4		
			8	6	4									Reload counter	
			9	2	3		I	N	X	H				for 100,0 interval	
MICROCOMPUTER TRAINING SYSTEM	A		5	E		M	O	V	E	,	M			Copy and clear	
	B		3	6		M	V	I	M	,	0	0		count of EXT 4	
	C		0	0										interrupts	
	D		2	3		I	N	X	H						
	E		5	6		M	O	V	D	,	M				
	F		3	6		M	V	I	M	,	0	0			
	8	26	0	0	0	0									
			1	2	3		I	N	X	H					
			2	7	3		M	O	V	M	,	E			Store count of EXT 4
			3	2	3		I	N	X	H					interrupts for
		4	7	2		M	O	V	M	,	D			display as frequency	
		5	D	1		P	O	P	D						
		6	C	3		J	M	P	8	2	4	1		Exit	
		7	4	1											
		8	8	2											
INTEGRATED COMPUTER SYSTEMS	A														
	B														
	C														
	D														
	E														
	F														
	8		0												
			1												
			2												
		3													
		4													
		5													
		6													
		7													
		8												Figure 5-6c	

ANALOG TO DIGITAL INPUT

5.2.2 AC Input Signal

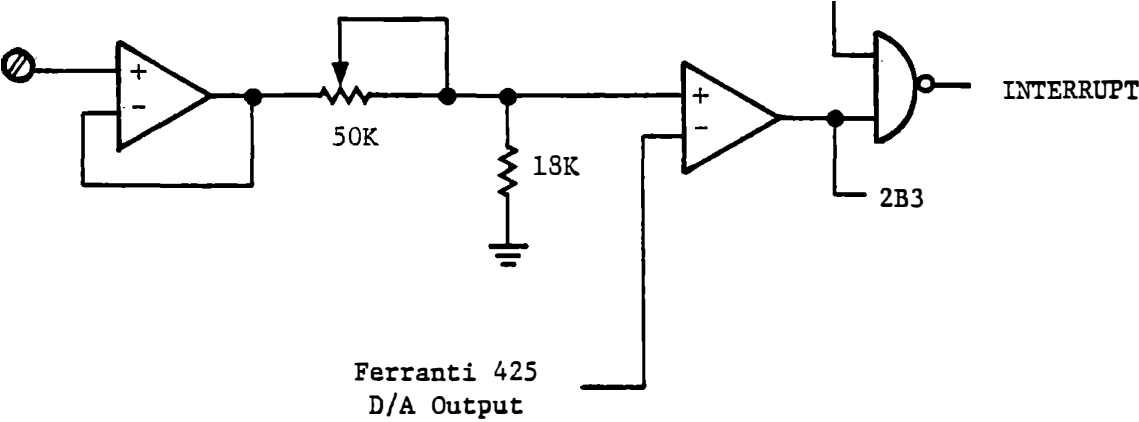
EXERCISE

In the preceding section, we measured the frequency of a logic level signal. Often the variable input may be an ac signal, without sharply defined edges. For accurate results, the input signal must be squared. A sinusoidal input may not be detected at the same point in its cycle every time. Squaring can be accomplished with an integrated circuit comparator or an op-amp connected as a comparator. The interface board includes a comparator in the analog input circuit which can be used in this way. Figure 5-7a shows the circuit.

CAUTION: The input to the op-amp must not go more negative than -0.3 volts. If the signal is alternating above and below ground, a protection circuit must be provided as indicated in Figure 5-7b or else the signal must be attenuated to swing within ± 0.25 volts.

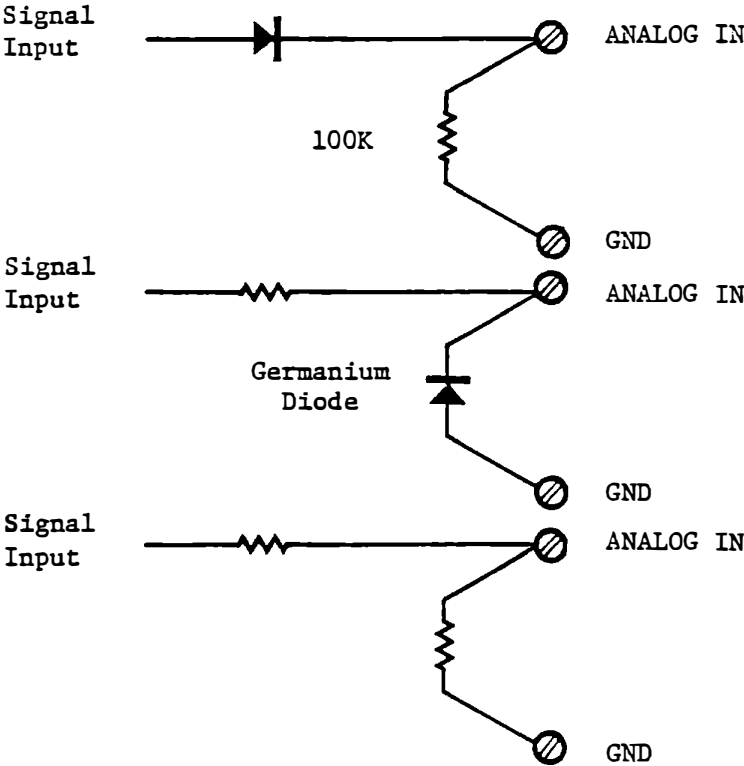
CAUTION: The input to the op-amp must not go more negative than -0.3 volts. If the signal is alternating above and below ground, a protection circuit must be provided as indicated in Figure 5-7b or else the signal must be attenuated to swing within ± 0.25 volts.

ANALOG TO DIGITAL INPUT



COMPARATOR CIRCUIT

Figure 5-7a



Protection Circuits for AC Signals

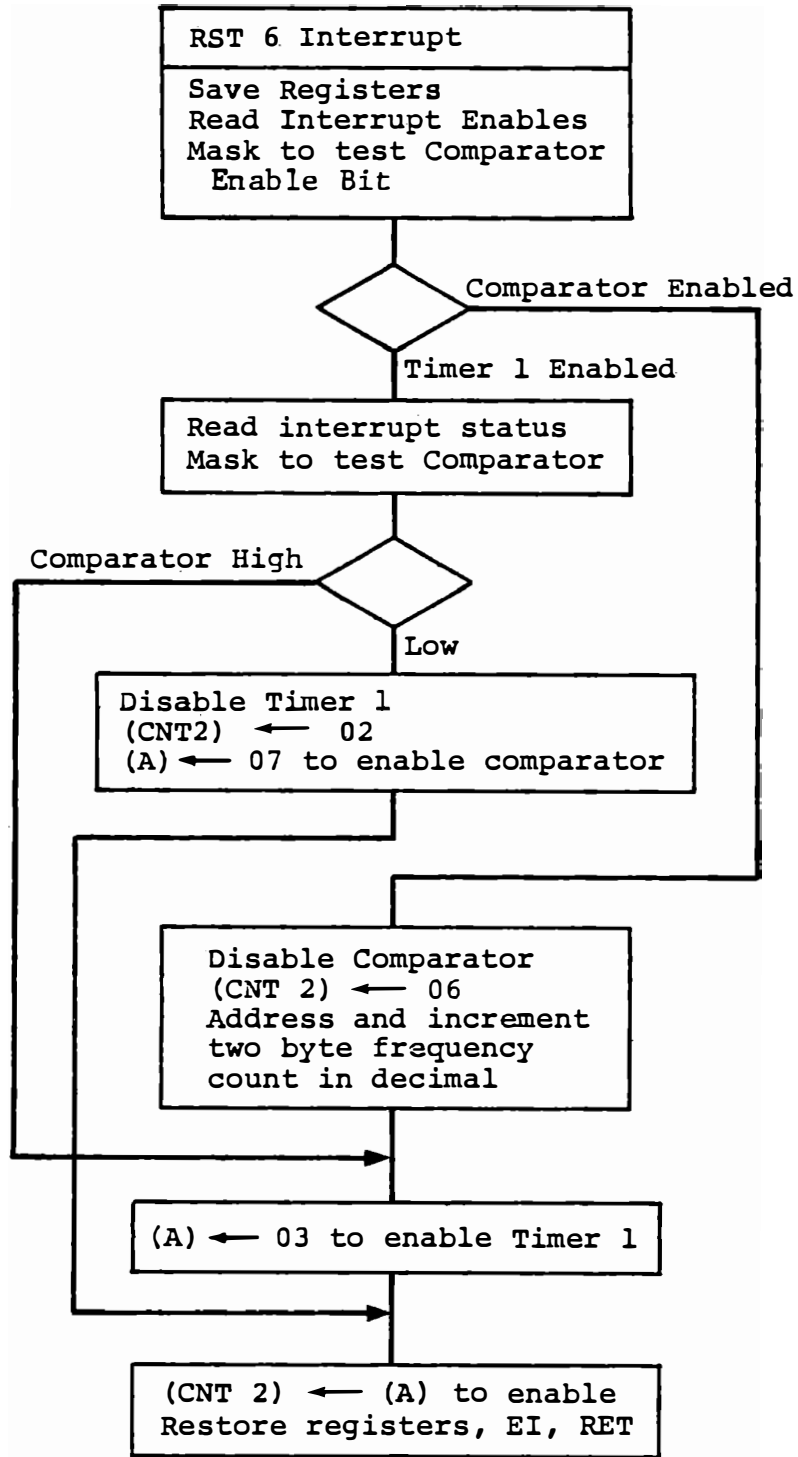
Figure 5-7

ANALOG TO DIGITAL INPUT

The input signal to ANALOG IN is amplified (with unity voltage gain) by the first op-amp, attenuated if necessary by the pot, and compared with the output signal from the D/A converter. A threshold signal is provided by the converter. This can be set very close to 0 volts, or to some more positive value. When the input signal is greater than the threshold, the output of the second op-amp is a low logic level. When the input signal is less than the threshold, the logic signal goes high and can generate an interrupt or be sensed at port 2B3.

The cassette modem output provides a suitable signal to test this program. It has an amplitude of about ± 0.3 volts with a very high source impedance. Connect a 100K resistor from analog input to ground to ensure that the signal stays within the safe range for the op-amp. Set the ANALOG IN pot to the far right, and connect the CASSETTE AUX output to ANALOG IN.

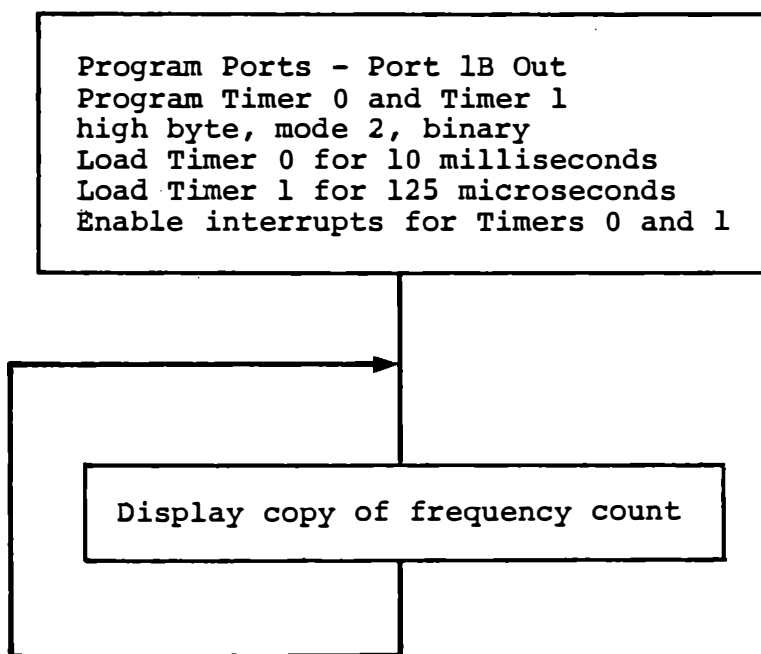
With this arrangement, a RST 6 interrupt will occur whenever the external signal goes below the threshold. Since there is no latch for this interrupt, the program must monitor port 2B3, and not enable the interrupt again until this signal has become low. In the program of Figures 5-8 and 5-9, timer 1 is used to interrupt the main program often enough to detect when the comparator output goes low and then enable the A/D comparator interrupt. Timer 0 again counts time. The frequency is counted in response to comparator interrupts instead of EXT 4 interrupts.



Sinusoidal Measurement -- RST 6 Interrupt

Figure 5-8a

ANALOG TO DIGITAL INPUT



AC Signal Frequency - Main

Figure 5-8b

Interrupt service for this program (Figure 5-8a) introduces a primitive interrupt manager. The occurrence of RST6 does not by itself tell the program which interrupt source must be serviced. We read the interrupt enable byte (port 2C) and test whether the comparator or timer 1 was enabled to create the interrupt. (We know that only one has been enabled.) If the comparator interrupt was enabled, we know that the comparator caused the interrupt and now must be disabled and timer 1 enabled, and the frequency count should be incremented. If timer 1 was enabled, we read the interrupt status byte (port 2B) to decide whether it is now time to enable the comparator (and disable timer 1) or whether timer 1 should be reenabled and the comparator remain disabled.

Clearly, the function of testing the comparator signal could be relegated to the main program, which has very little to do. We used the two RST 6 interrupts in order to demonstrate one means of distinguishing the source.

AC SIGNAL FREQUENCY MEASUREMENT

		A	D	D	R	CODE				
CODING SHEET	8	20	0	3E		MVI	A,	80		Program Ports Port B out
		1		80						
		2		D3		OUT	CNT	1		
		3		07						
		4		3E		MVI	A,	92		
		5		92						
		6		D3		OUT	CNT	2		
		7		0F						
MICROCOMPUTER TRAINING SYSTEM	8			3E		MVI	A,	24		Program Timers High byte Mode 2 Binary
		9		24						
		A		D3		OUT	TIM	CT		
		B		17						
		C		3E		MVI	A,	64		
		D		64						
		E		D3		OUT	TIM	CT		
		F		17						
INTEGRATED COMPUTER SYSTEMS	8	21	0	3E		MOV	A,	50		10 millisecond interval to Timer 0
		1		50						
		2		D3		OUT	TIM	0		
		3		14						
		4		3E		MVI	A,	01		125 millisecond interval to Timer 1
		5		01						
		6		D3		OUT	TIM	1		
		7		15						
	8		3E		MVI	A,	03		Enable Timers 0 and Timers 1	
	9		03							
	A		D3		OUT	PORT	2C			
	B		0E							
	821C		2A		LHLD			8303		Loop-Load Copy of program Displays
	D		03							
	E		83							
	F		CD		CALL			DWORD		
	8	22	0	D1						
	1		02							
	2		C3		JMP			821C		
	3		1C							
	4		82							
	5									
	6									
	7									
	8									

Figure 5-9a

AC FREQUENCY MEASUREMENT - INTERRUPTS

		A	D	D	R	CODE																
CODING SHEET	8	0																				
		1				*	S	A	M	E	A	S	F	I	G	U	R	E	5-6b			
		2					E	X	C	E	P	T	F	O	R	P	A	T	C	H	E	
		3					M	A	R	K	E	D	*									
		4																				
		5																				
		6																				
		7																				
MICROCOMPUTER TRAINING SYSTEM	822	8	F	5			P	U	S	H	P	S	W							same as 5-6b		
		9	E	5			P	U	S	H	H											
		A	2	1			L	X	I	H,	8	3	0	0								
		B	0	0																		
		C	8	3																		
		D	C	3			J	M	P	8	2	5	0									
		E	5	0																		
		F	8	2																		
		823	0	F	5			P	U	S	H	P	S	W							RST6 - COMP. TIM1	
		1	E	5			P	U	S	H	H											
		2	C	3	*		J	M	P	8	2	7	0								PATCH - Jump to	
		3	7	0	*																test which interrupt	
		4	8	2	*																	
	823	5	3	7			S	T	C												COMPARATOR	
	823	6	7	E			M	O	V	A,	M										INTERRUPT	
		7	C	E			A	C	I	0	0										same as 5-6b	
		8	0	0																		
		9	2	7			D	A	A													
		A	7	7			M	O	V	M,	A											
		B	2	3			I	N	X	H,												
		C	D	A			J	C	8	2	3	6										
		D	3	6																		
		E	8	2																		
	823	F	3	E			M	V	I	A,	0	3									To enable TIM1	
	824	0	0	3	*																	
	824	1	D	3			O	U	T	C	N	T	2								same as 5-6b	
		2	O	F																		
		3	E	1			P	O	P	H												
		4	F	1			P	O	P	P	S	W										
		5	F	B			E	I														
		6	C	9			R	E	T													
		7																				
		8																				

Figure 5-9b

FREQUENCY - TIMER 0 INTERRUPT (continued)

		A	D	D	R	CODE						
CODING SHEET	8 25	0	3	5		D	C	R	M		Decrement counter	
		1	3	E		M	V	I	A, 01		To reload timer 0	
		2	0	1								
		3	C	2		J	N	Z	8241		Exit unless	
		4	4	1							counter reaches 00	
		5	8	2								
		6	D	5		P	U	S	H	D		
		7	3	6		M	V	I	M, 64		Reload counter	
		8	6	4							for 100.0 intervals	
		9	2	3		I	N	X	H			
MICROCOMPUTER TRAINING SYSTEM	A	5	E		M	O	V	E, M		Copy and clear		
	B	3	6		M	V	I	M, 00		count of EXT 4		
	C	0	0							interrupts		
	D	2	3		I	N	X	H				
	E	5	6		M	O	V	D, M				
	F	3	6		M	V	I	M, 00				
	8 26	0	0	0								
		1	2	3		I	N	X	H			
		2	7	3		M	O	V	M, E		Store count of EXT 4	
		3	2	3		I	N	X	H		interrupts for	
	4	7	2		M	O	V	M, D		display as frequency		
	5	D	1		P	O	P	D,				
	6	C	3		J	M	P	8241		Exit		
	7	4	1									
	8	8	2									
	9											
INTEGRATED COMPUTER SYSTEMS	A											
	B					S	A	M	E	A	S	
	C									F	I	
	D									G		
	E											
	F											
	8	0										
		1										
	2											
	3											
	4											
	5											
	6											
	7											
	8											

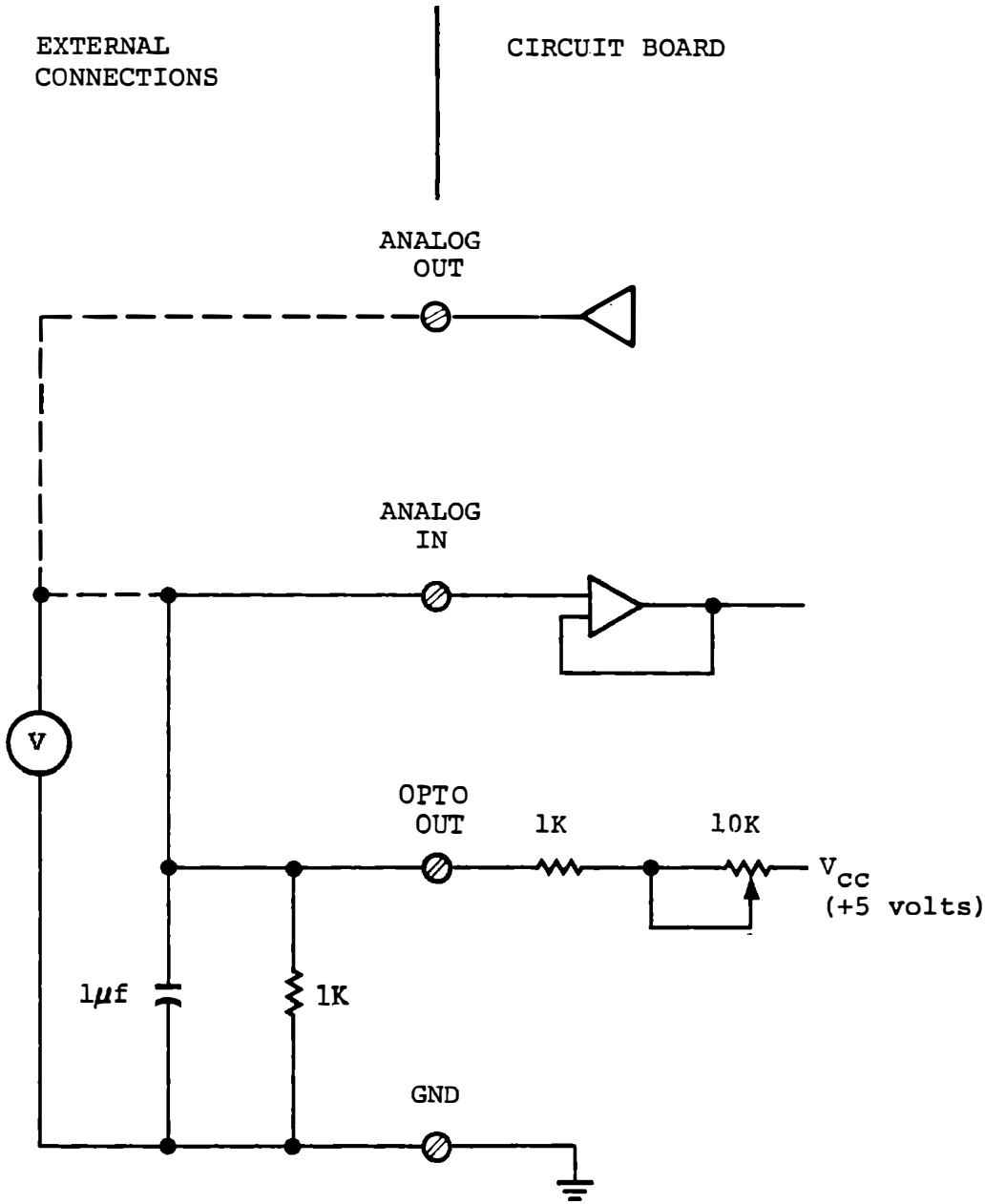
Figure 5-9c

INTERLUPT SERVICE FRICH FOR IN DEGREE

		A	D	D	R	CODE					
CODING SHEET	8	27	0	DB		IN	PORT	2C			Read interrupt enabled
			1	0E							
			2	E6		ANI	08				Test for comparator interrupt enabled
			3	08							
			4	CA		JZ		8281			Jump if not
			5	81							
			6	82							COMPARATOR
			7	3E		MVI	A,	06			Disable comparator interrupt
			8	06							
			9	D3		OUT	CNT	2			
MICROCOMPUTER TRAINING SYSTEM	A			0F							
	B			21		LXI	H,	8301			Address frequency count
	C			01							
	D			83							
	E			C3		JMP		8235			Jump to increment frequency count
	F			35							
	8	28	0	82							
		828	1	DB		IN	PORT	2B			TIMER 1
			2	0D							Read interrupt status
			3	E6		ANI	08				Test comparator
INTEGRATED COMPUTER SYSTEMS			4	08							
			5	C2		JNZ		823F			If still high go to reenable
			6	3F							Timer 1
			7	82							
			8	3E		MVI	A,	02			If low disable Timer 1
			9	02							
	A			D3		OUT	CNT	2			
	B			0F							
	C			3E		MVI	A,	07			and enable comparator interrupt
	D			07							
E			C3		JMP		8241				
F			41								
8	29	0	82								
		1									
		2									
		3									
		4									
		5									
		6									
		7									
		8									

Figure 5-9d

ANALOG TO DIGITAL INPUT



Connections for Voltmeter Experiments

Figure 5-10

5.3 A/D INPUT - VOLTAGE

Conversion of a voltage input to a digital value is generally performed by comparing the input signal with a voltage generated by digital to analog conversion. The result of the comparison is used to adjust the digital value until the two voltages are alike. A/D converters differ in the adjustment procedure, three principal methods being repetitive ramp, tracking, and successive approximation. We will experiment with each of these.

The comparison between the D/A output and the analog input is the primary purpose of the comparator circuit and gating that were introduced in Section 5.2.2. Refer again to Figure 5-7a, which shows the circuit. For direct measurement of a voltage that is within the 0 to +2.55 volt range of the D/A converter, the ANALOG IN pot can be set for no attenuation of the input signal. For a signal between 2.5 and 5.0 volts, the pot can be set to attenuate the signal by a known amount. Signals greater than 5.0 volts must be attenuated externally, because the op-amp cannot handle a signal greater than its supply voltage. (It will not be damaged by any signal up to +30 volts, but remember that signals lower than -0.3 volts will damage the op-amp.) Since our OPTO OUT will be less than 2.5 volts, we can set the ANALOG IN pot for no attenuation (rotate fully to the left).

A variable DC voltage is needed for these experiments. The OPTO OUT of the interface board is connected through a pot to 5 volts (see Figure 5-10). With an external 1K resistor to ground, a voltage between 0.4 and 2.4 volts can be obtained. A capacitor from the output to ground is needed to remove noise from the signal. The

ANALOG TO DIGITAL INPUT

signal is to be connected to ANALOG IN, and your voltmeter will be connected to either ANALOG IN or ANALOG OUT.

Note: If you are familiar with A/D conversion, you may want to skip the exercises of this section and proceed to Section 5.4, where the use of the automatic A/D input feature is described.

5.3.1 Output, Input and Display Subroutine

EXERCISE

All of the voltmeter programs involve changing the digital value repetitively, comparing its analog conversion with the input signal, and making a decision on that comparison. This operation involves the following steps (with the digital value kept in register L):

```
MOV A,L      (A) <--- digital value
```

```
OUT PORT 1B  To D/A converter
```

(about 40 micro-seconds delay is required between OUT and IN)

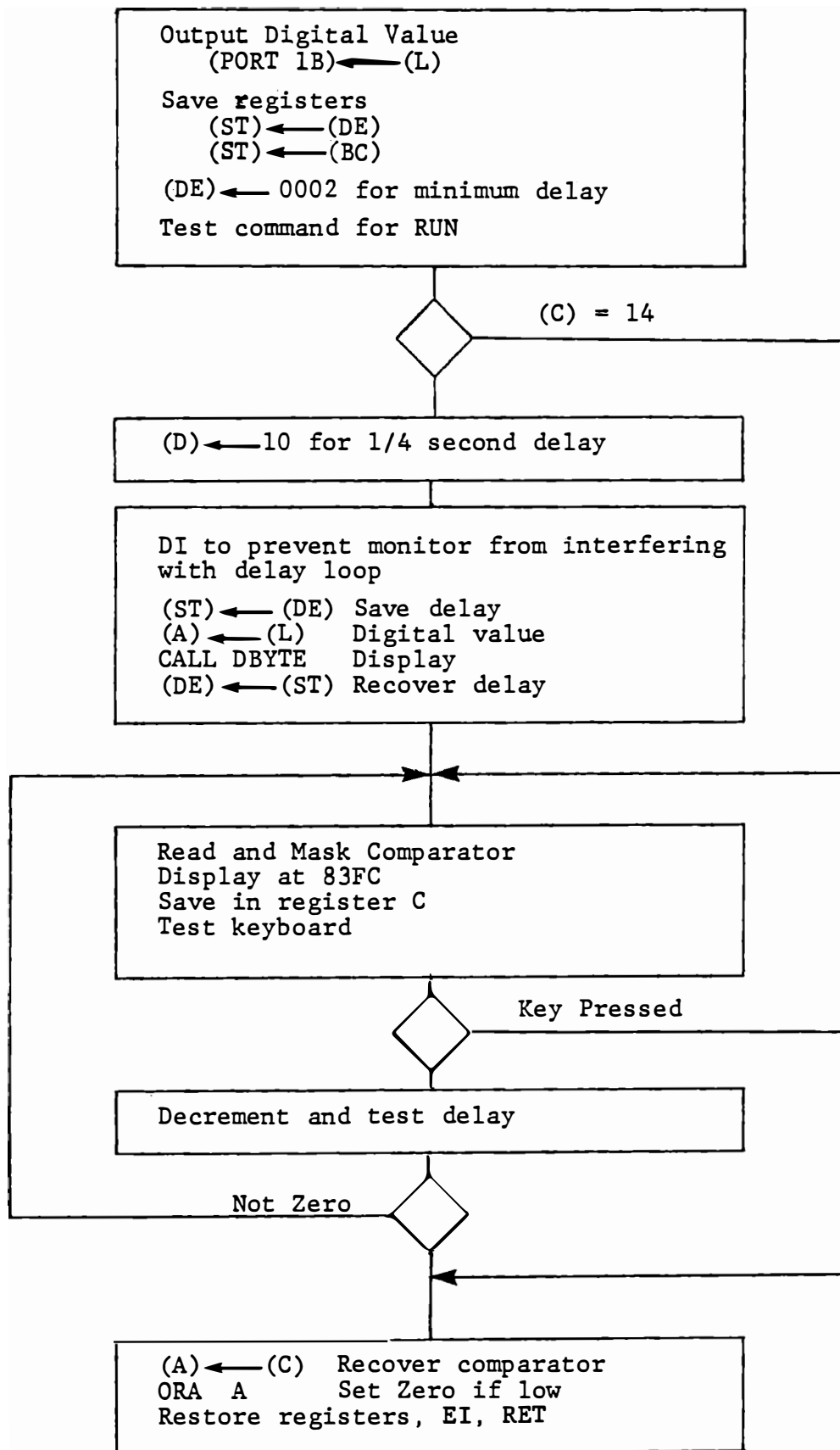
```
IN PORT 2B   Read interrupt status
```

```
ANI 08      Mask for comparator
```

Both the digital to analog converter and the comparator require some settling time before the comparison of D/A output voltage to input voltage is valid. This delay should be at least 40 microseconds. Usually some other function can usefully be accomplished, but otherwise a delay loop can be used.

These steps will usually be followed by jump if zero, or jump if not zero for the decision.

ANALOG TO DIGITAL INPUT

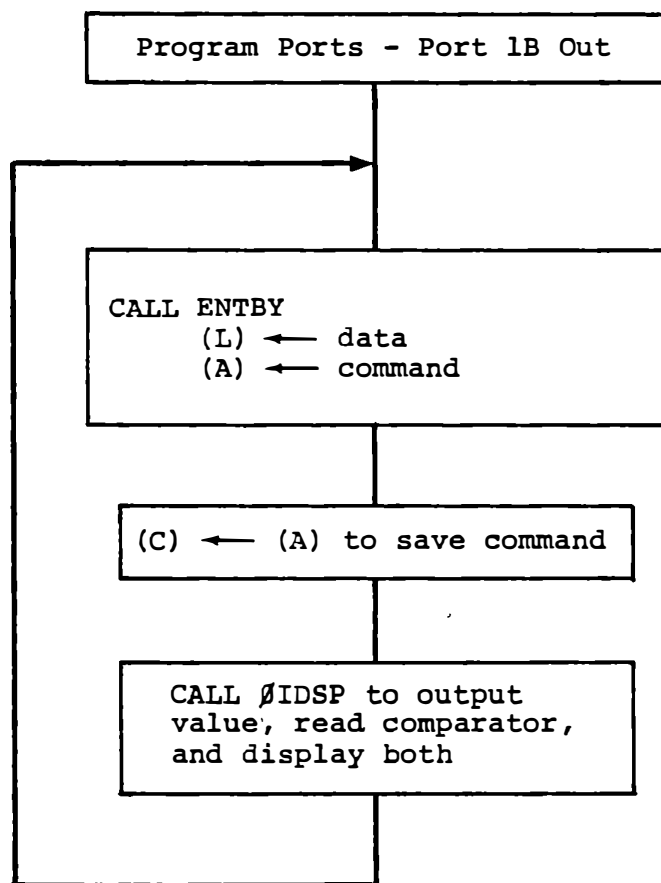


Output, Input and Display Subroutine

Figure 5-11

For convenience in debugging several voltmeter programs, we will create a subroutine that will perform output and input, and also display the digital value and the result of the comparison for some fixed length of time before allowing the main program to proceed with its operations. It will save registers so that it will have exactly the effect of the above process when it returns. In addition, it will make two tests to defeat the delay, as shown in Figure 5-11. If a key input command previously entered and saved in register C is RUN (=14) a minimum delay is set and the display is bypassed. Otherwise a 1/4 second delay is entered and the digital value is displayed. Within the delay loop the keyboard is tested, and if any key is pressed the delay is abandoned.

Note that saving registers, loading the delay and testing the command provide marginally enough time for the comparator to settle after the digital output. To guarantee enough time when RUN is used, a delay count of 0002 is used in this case.



Test Program for OIDSF

Figure 5-12

The comparator is read, displayed and saved within the delay loop, by:

IN	PORT2B	Read interrupt status
ANI	08	Mask comparator
STA	83FC	Display
MOV	C,A	Save comparator bit

At exit from the loop, either when the delay count reaches zero or when a key is pressed, the comparator bit is recovered from (C). Since the flag set by masking has been lost by the keyboard test and delay count, ORA A is executed to set or clear the zero flag according to the content of (A). At exit the zero flag is set if the digital value is less than the input voltage.

A trivial test program, shown in Figure 5-12, is suitable for debugging your Output/Input/Display subroutine (OIDSP), and also for calibration of the potentiometers. Connect the voltmeter to ANALOG OUT initially. When you key in a numeric value, with any command, it will be output to the D/A converter. Key in FA, STEP, and adjust the ANALOG OUT pot to obtain 2.50 volts on the voltmeter. Key in other values, and find the value at which the comparator output changes from low to high, as shown on the display. When the digital value is greater than the input signal, the comparator bit is set, and will be displayed as a bottom horizontal bar, indicating that the digital value must be reduced.

This page intentionally left blank.

TEST PROGRAM FOR VOLTAGE DISPLAYS

	A	D	D	R	CODE						
CODING SHEET	8	20	0	3E	MVI	A,	80				Program Ports
			1	80							Port 1B Out
			2	D3	OUT	CNT	1				
			3	07							
			4	3E	MVI	A,	92				
			5	92							
			6	D3	OUT	CNT	2				
			7	0F							
MICROCOMPUTER TRAINING SYSTEM	820	8	00	NOP							
		9	CD	CALL	ENTBY						
		A	36								(L) ← digital value
		B	03								(A) ← command
		C	00	NOP							
		D	4F	MOV	C,	A					(C) ← command
		E	CD	CALL	OIDS	P					
		F	C0								
INTEGRATED COMPUTER SYSTEMS	8	21	0	82							
			1	C3	JMP	8208					
			2	08							
			3	82							
			4								
			5								
			6								
			7								
		8									

Figure 5-13a

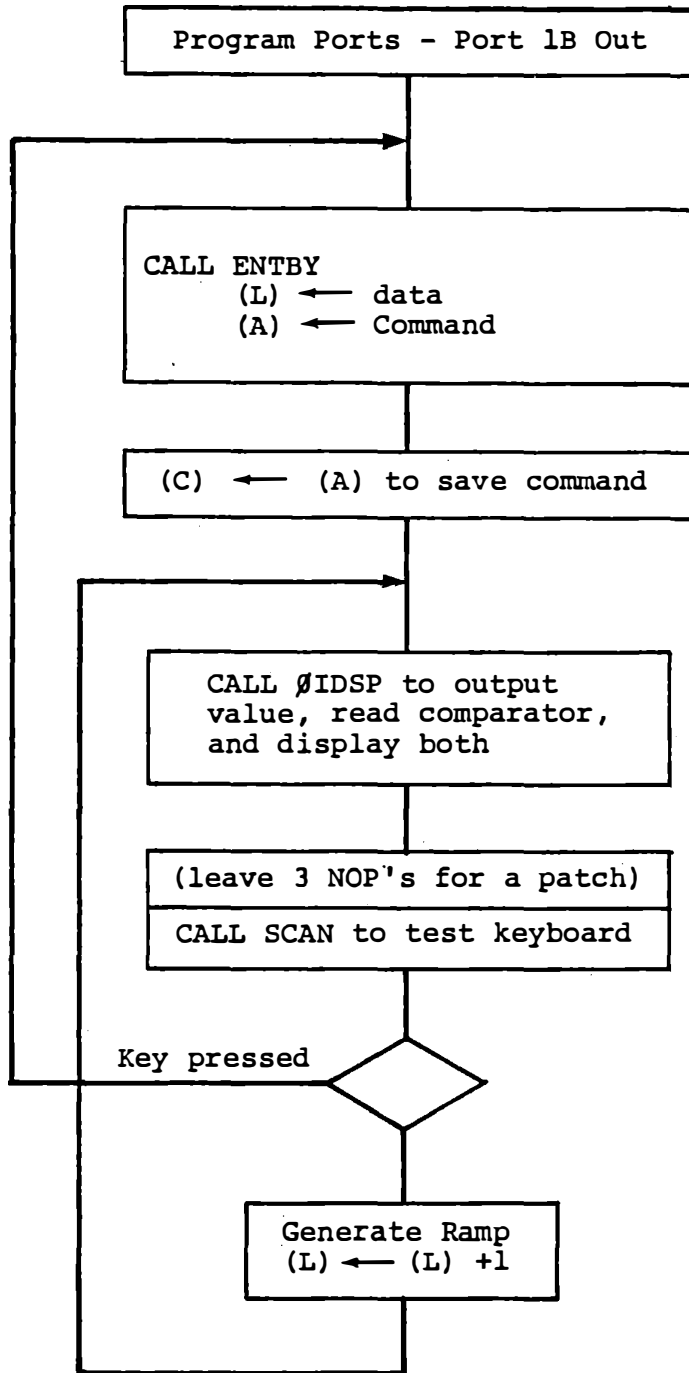
OIDSP - OUTPUT, INPUTS, DISPLAY

A	D	D	B	CODE									
8	2C	0	7D	MOV	A, L								Output digital value
		1	D3	OUT	PORT 1 B								to DA converter
		2	05										
		3	D5	PUSH	D								save register
		4	C5	PUSH	B								
		5	11	LXI	D, 0002								Minimum delay
		6	02										for RUN command
		7	00										
		8	79	MOV	A, C								Test for RUN
		9	FE	CPI	14								
		A	14										
		B	CA	JZ	82D7								Jump next display
		C	D7										if RUN
		D	82										
		E	16	MVI	D, 10								for 1/4 second
		F	10										delay in next RUN
8	2D	0	F3	DI									Disable monitor
		1	D5	PUSH	D								during long delay
		2	7D	MOV	A, L								
		3	CD	CALL	DBYTE								
		4	95										
		5	02										
		6	D1	POP	D								
	82D	7	DB	IN	PORT 2 B								Read interrupt
		8	0D										status byte
		9	E6	ANI	08								Mask for
		A	08										A/D Comparator
		B	32	STA	83FC								Display
		C	FC										comparator lit
		D	83										
		E	4F	MOV	C, A								save comparator
		F	00	NOP									
8		0											
		1		ENTER	(L) =	DIGITAL VALUE							
		2		RETURN									
		3		(A) = 00	AND ZERO FLAG SET								
		4		IF (L) <	INPUT VOLTAGE								
		5											
		6		(A) = 08	AND NOT ZERO								
		7		IF (L) >	INPUT VOLTAGE								
		8											Figure 5-13b

OIDSP (continued)

A D D R		CODE							
CODING SHEET	8 2E	0	DB		IN		PORT 0 A	Test keyboard	
		1	00					(FF if no key)	
		2	3C		INR	A			
		3	C2		JNZ		8 2 EC	Exit if key pressed	
		4	EC						
		5	82						
		6	1B		DCX	D		Decrement and	
		7	7B		MOV	A, E		test delay count	
		8	B2		ORA	D			
		9	C2		JNZ		8 2 D7	Loop until	
MICROCOMPUTER TRAINING SYSTEM		A	D7					delay finished	
		B	82						
		C	79		MOV	A, C		(A) ← comparator bit	
		D	B7		ORA	A		set zero if low	
		E	C1		POP	B		Exit	
		F	D1		POP	D			
	INTEGRATED COMPUTER SYSTEMS	8 2F	0	FB		EI			
			1	C9		RET			
			2						
			3						
		4							
		5							
		6							
		7							
		A							
		B							
	C								
	D								
	E								
	F								
	8	0							
		1							
		2							
		3							
		4							
		5							
		6							
		7							
		8							

Figure 5-13c



Voltage Ramp Generator

Figure 5-14

5.3.2 Ramping Voltmeter

EXERCISE

Modify the test program as shown in Figure 5-14, to generate an output voltage ramp. After output and delay, test the keyboard and go to CALL ENTBY only if a key is pressed, otherwise increment the digital value and go again to OIDSP. (Remember that OIDSP exits immediately when a key is pressed, but it does not indicate whether a key was pressed. Therefore, the main program must test the keyboard independently.)

Now the program will cycle the digital value in register L, counting from 00 through FF and back to 00. This will generate a voltage ramp at the D/A output, as shown in Figure 5-15. When the output is less than the input, the comparator bit will be low. When the output becomes greater, the comparator bit will be high. Your voltmeter on ANALOG OUT will show the ramp. Suggested code is shown in Figure 5-16.

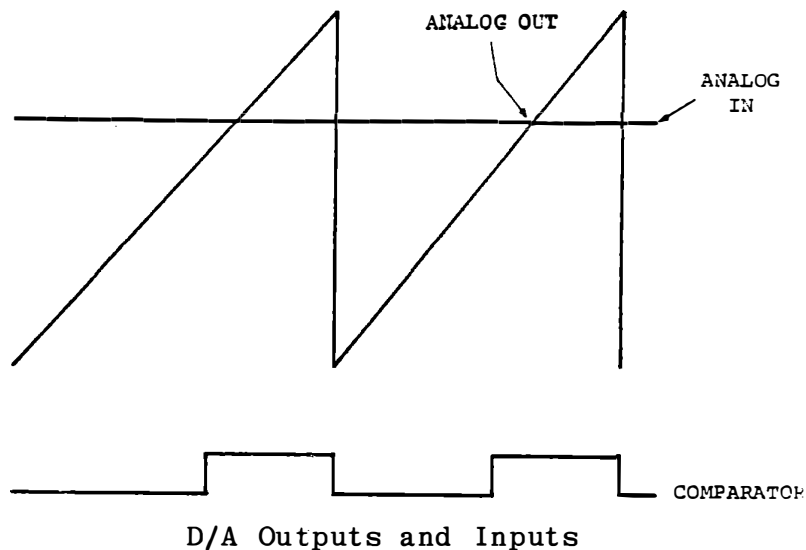


Figure 5-15

		A	D	D	R	CODE	VOLTAGE RAMP GENERATOR								
CODING SHEET	8	20	0	3	E		M	V	I	A, 80	Program Ports Port B out				
			1	8	0										
				2	D	3		O	U	T	CNT1				
				3	0	7									
				4	3	E		M	V	I	A, 92				
				5	9	2									
				6	D	3		O	U	T	CNT2				
				7	0	F									
MICROCOMPUTER TRAINING SYSTEM	820	8	0	0			W	O	P						
			9	C	D		C	A	L	L	E	N	T	B	Y
			A	3	6										
			B	0	3										
			C	0	0			N	O	P					
			D	4	F		M	O	V	C, A					
		820	E	C	D		C	A	L	L	O	I	D	S	P
			F	C	0										
		8	21	0	8	2									
				1	0	0		N	O	P					
			2	0	0		N	O	P						
			3	0	0		N	O	P						
			4	C	D		C	A	L	L	S	C	A	N	
			5	5	7										
			6	0	2										
			7	D	A		J	C		8208					
			8	0	8										
			9	8	2										
INTEGRATED COMPUTER SYSTEMS			A	2	C		I	N	R	L					
			B	C	3		J	M	P	820E					
			C	0	E										
			D	8	2										
			E												
			F												
		8	0												
			1												

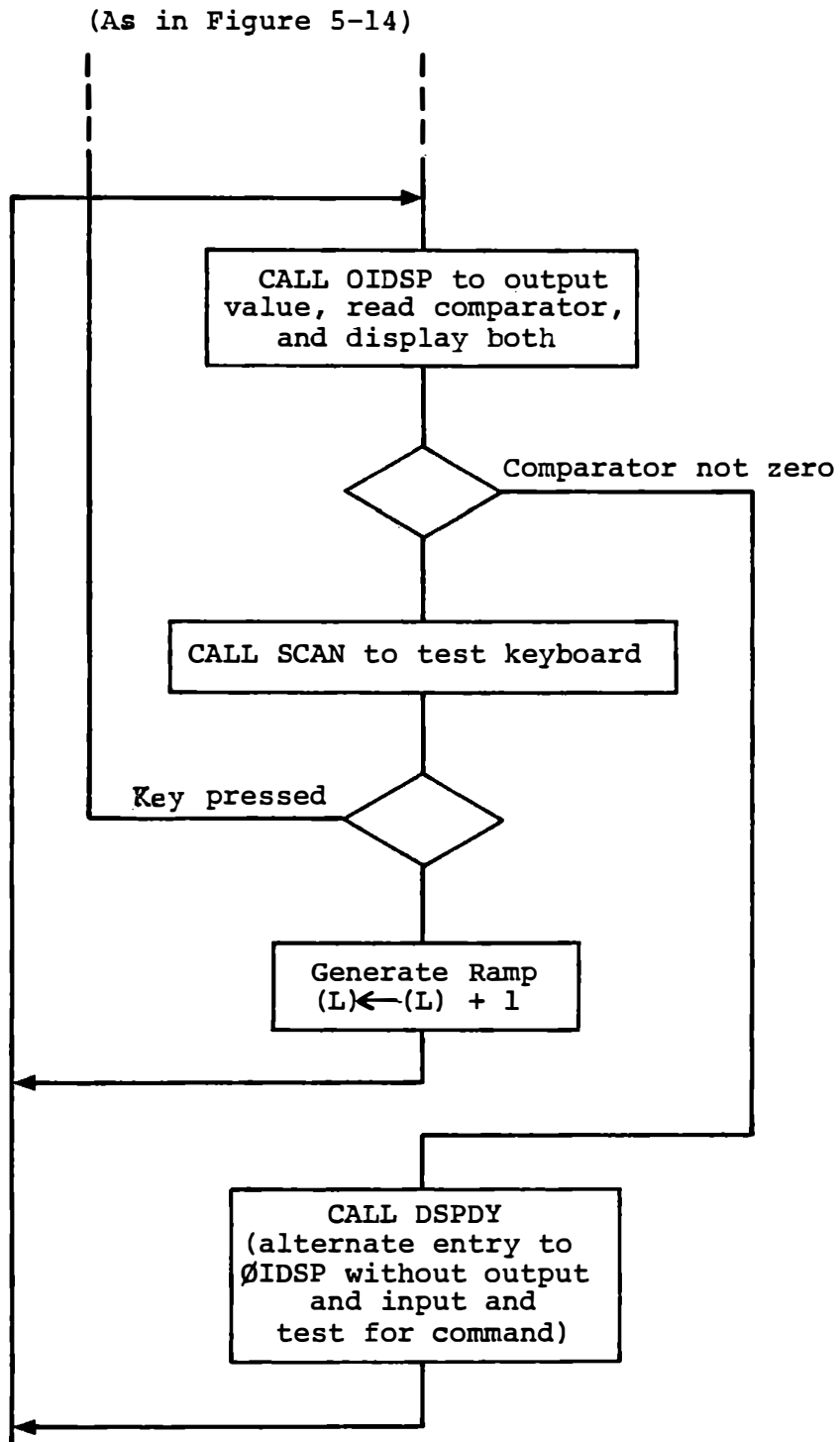
Figure 5-16

You may want to shorten the delay for each step by reducing the value loaded for the delay loop in OIDSF. With a value of 1000 the full cycle of the ramp will take about 60 seconds.

Now watch the display of the comparator bit. It will be blank until the D/A output exceeds the input voltage. As soon as it appears, press NEXT and hold it down. This will stop the program (ENTBY will wait for release of the key) and the output voltage will be displayed. When you release the key ENTBY will return with 00 in register L to start a new ramp.

Obviously this function need not depend on your finger, since the processor has the decision bit available. OIDSF returns with the zero flag set when the comparator is low, and cleared when it is high. Insert JNZ after the return from OIDSF, to a patch that will display the result and restart the ramp. The program of Figure 5-17 calls an alternate entry to OIDSF that bypasses the digital value output and the check on the stored command, and loads a different delay time.

ANALOG TO DIGITAL INPUT



Ramping Voltmeter

Figure 5-17

After the result display, the digital value is set to zero to start a new ramp. You can see the ramp on your voltmeter. If you press RUN, the display and delay for intermediate results will be inhibited by OIDSF, and only the final value will be displayed.

Now move the voltmeter to ANALOG IN to observe the input signal. Compare the value displayed by the program with the measured voltage. They should agree closely, but if some error exists, you can adjust the ANALOG IN pot to compensate for it. Adjust the OPTO SENSE pot to change the input value and observe the value measured by the program, comparing it with the voltmeter.

RAMPING VOLTMETER

A D D R		CODE					
CODING SHEET	8 20	0 3E	MVI	A,	80		Program Ports Port 1B Out
		1 80					
		2 D3	OUT	CNT1			
		3 07					
		4 3E	MVI	A,	92		
		5 92					
		6 D3	OUT	CNT2			
		7 0F					
MICROCOMPUTER TRAINING SYSTEM	820	8 00	NOP				
		9 CD	CALL	ENTBY			
		A 36					(L) ← digital value
		B 03					(A) ← command
		C 00	NOP				
		D 4F	MOV	C,	A		(C) ← command
	820	E CD	CALL	OIDS	P		Output, Input Display, Delay
		F C0					
	821	0 82					
		1 C2	JNZ	8220			Jump to display output
	2 20						
	3 82						
	4 CD	CALL	SCAN			Test keyboard	
	5 57						
	6 02						
	7 DA	JC	8208			If key pressed go to CALL ENTBY	
	8 08						
	9 82						
INTEGRATED COMPUTER SYSTEMS		A 2C	INR	L			Else increment the digital value and go to output, input and display.
		B C3	JMP	820E			
		C 0E					
		D 82					
		E					
		F					
	8	0					
		1					
	2						
	3						
	4						
	5						
	6						
	7						
	8						

Figure 5-18a

A D D R		CODE							
CODING SHEET	8	22	0	CD		CALL		DISPLAY	Display result
			1	F2					
			2	82					
			3	2E		MVI		L, 00	Load digital
			4	00					value
			5	C3		JMP		820E	start new
			6	DE					conversion
			7	82					
MICROCOMPUTER TRAINING SYSTEM			8						
			9						
			A						
			B						
			C						
			D						
			E						
			F						
INTEGRATED COMPUTER SYSTEMS	8		0						
			1						
			2						
			3						
			4						
			5						
			6						
			7						
		8							

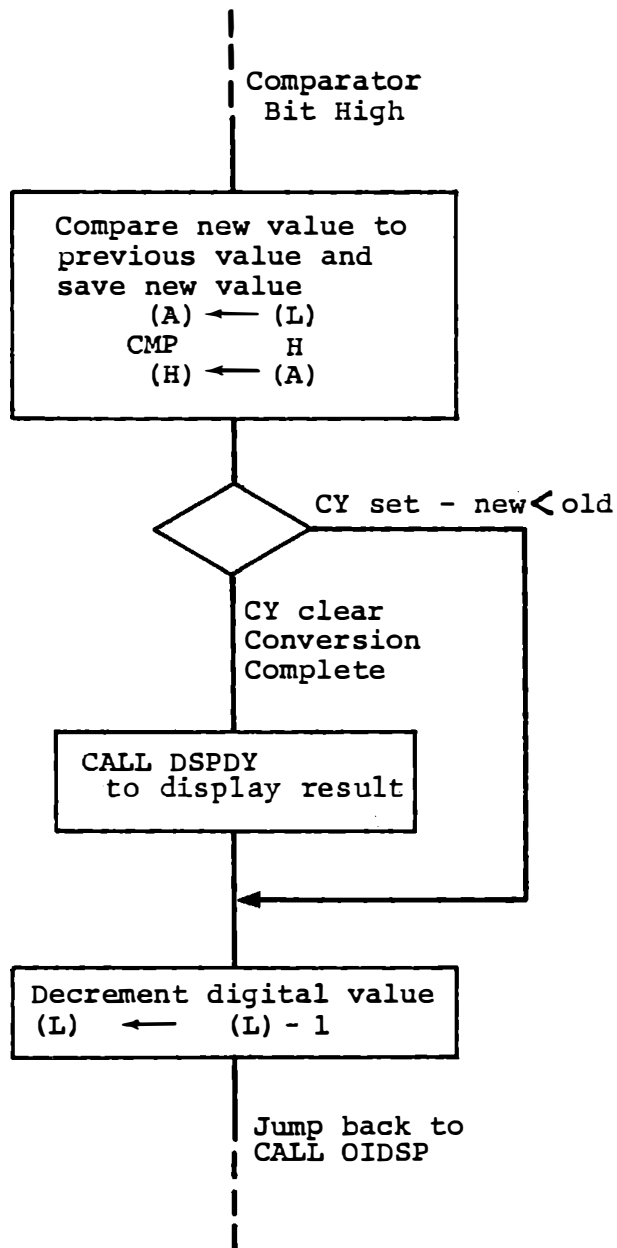
Figure 5-18b

OIDSP - OUTPUT, INPUTS, DISPLAY

	A	D	D	R	CODE																		
CODING SHEET	8	2C	0	7D	MOV	A, L														Output digital value			
			1	D3	OUT	PORT 1B															to D/A converter		
			2	05																			
			3	D5	PUSH	D																save registers	
			4	C5	PUSH	B																	
			5	11	LXI	D, 0002																Minimum delay	
			6	02																		for RUN command	
			7	00																			
			8	79	MOV	A, C																sent for RUN	
			9	FE	CPI	14																	
MICROCOMPUTER TRAINING SYSTEM	A			14																			
	B			CA	JZ	82D7																Jump past display	
	C			D7																		if RUN	
	D			82																			
	E			16	MVI	D, 10																for 1/4 second	
	F			10																		delay if not RUN	
	8	2D	0	F3	DI																	Disable monitor	
			1	D5	PUSH	D																	during long delay
			2	7D	MOV	A, L																	
			3	CD	CALL	D BYTE																	
INTEGRATED COMPUTER SYSTEMS				95																			
				02																			
				D1	POP	D																	
	82D	7		DB	IN	PORT 2B																Read interrupt	
				0D																			status byte
				E6	ANI	08																	Mask for
	A			08																			A/D comparator
	B			32	STA	83FC																	Display
	C			FC																			comparator bit
	D			83																			
			4F	MOV	C, A																	save comparator	
			00	NOI																			
	8	0																					
			1		ENTER	(L) =																DIGITAL VALUE	
			2		RETURN																		
			3		(A) = 00	AND																ZERO FLAG SET	
			4		IF (L) <																	INPUT VOLTAGE	
			5																				
			6		(A) = 08	AND																NOT ZERO	
			7		IF (L) >																	INPUT VOLTAGE	
			8																			Figure 5-18c	

A D D R		CODE										
CODING SHEET	8	2E	0	DB		IN		PORT	0A		Test keyboard (FF if no key)	
			1	00								
			2	3C		INR	A					
			3	C2		JNZ	82EC				Exit if key pressed	
			4	EC								
			5	82								
			6	1B		DCX	D				Decrement and test delay counter	
			7	7B		MOV	A, E					
			8	B2		ORA	D					
			9	02		JNZ	82D7				Loop until delay finished	
MICROCOMPUTER TRAINING SYSTEM	A			D7								
	B			82								
	82E	C		79		MOV	A, C				(A) ← comparator bit set zero if low	
	D			B7		ORA	A, C				Exit	
	E			C1		POP	B					
	F			D1		POP	D					
	8	2F	0	FB		EI						
			1	C9		RET						
		82F	2	D5	*	PUSH	D				DSPDY -	
			3	C5		PUSH	B				Display final result for	
		4	11		LXI	D, 8000						
		5	00									
		6	80									
		7	C3		JMP	82D0						
		8	D0									
		9	82									
INTEGRATED COMPUTER SYSTEMS	A											
	B				*	ALTERNATE	ENTRY	DSPDY				
	C					DISPLAYS	FINAL	RESULT				
	D					FOR	2	SECONDS				
	E											
	F											
	8		0									
			1									
		2										
		3										
		4										
		5										
		6										
		7										
		8									Figure 5-18d	

ANALOG TO DIGITAL INPUT



Tracking Voltmeter

Figure 5-19

5.3.3 Tracking Voltmeter

EXERCISE

If an analog to digital conversion is being performed on only one input signal, after the first conversion is complete, it is not necessary to generate repetitive ramps. Instead, the digital value can be decreased when the comparator indicates that it is greater than the input, and increased when it is less. Now the D/A voltage will track the input signal. Modify the ramping voltmeter program so that it enters a tracking mode when a conversion is complete. When a key is pressed, it should start a new conversion. As before, the RUN command will cause display of completed conversion only, while NEXT will call for display of intermediate results. When the program is in tracking mode, the conversion is complete when the comparator bit becomes high after an increase in the digital value.

ANALOG TO DIGITAL INPUT

Figure 5-19 shows the modification to the ramping voltmeter. The program is identical except for the action taken when the comparator bit is high. Now after each test by OIDSF, if the comparator is low the digital value is incremented as before, but if it is high the digital value is decremented. If the comparator remains high for successively lower digital values the decrementing continues, so as to track a decreasing voltage. When the comparator is high at a digital value equal to or greater than the previous value, the conversion is complete and the long display is made. Now the digital value output will alternately increase and decrease by one bit. A complete new conversion, starting at zero will be started only when a command key is entered.

Connect your voltmeter to the ANALOG OUT signal, and watch it track the input as you adjust the SENSE pot.

TRACKING VOLTMETER

A D D R		CODE										
CODING SHEET	8	22	0	7D	MOV	A	L					
			1	BC	CMP	H						
			2	67	MOV	H	A					
			3	D4	CNC	D	S	P	D	Y		
			4	F2								
			5	82								
			6	2D	DCR	L						
			7	C3	JMP	P	2	0	E			
			8	0E								
			9	F2								
MICROCOMPUTER TRAINING SYSTEM	A				PROGRAM	AT				8200-821F		
	B				AND	O	I	D	S	P	82C0-82FF	
	C				I	D	E	N	T	I	C	
	D				R	A	M	P	I	N	G	
	E										V	
	F										O	
												L
												T
												M
												E
INTEGRATED COMPUTER SYSTEMS	8	0										
			1									
			2									
			3									
			4									
			5									
			6									
			7									
			8									
			9									
		A										
		B										
		C										
		D										
		E										
		F										
		8	0									
			1									
			2									
			3									
			4									
			5									
			6									
			7									
			8									

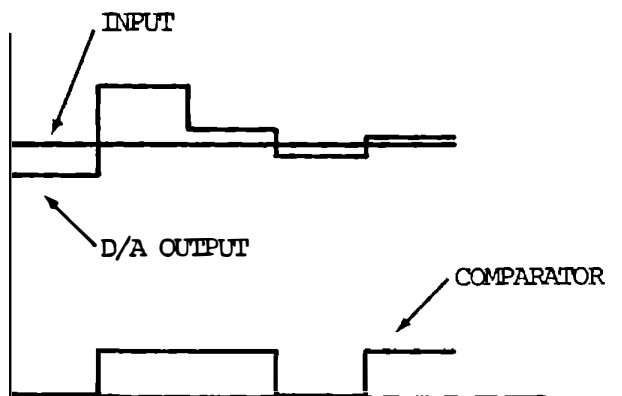
Figure 5-20

ANALOG TO DIGITAL INPUT

5.3.4 Successive Approximation Voltmeter

EXERCISE

The ramping voltmeter is relatively slow in reaching equality of output and input, and the time required for a conversion varies according to the input voltage. The successive approximation method overcomes both of these drawbacks. Instead of starting at zero and ramping upward, the D/A converter output is started at one half of full scale and increased or decreased according to the comparator signal by successively smaller amounts ($1/2$, $1/4$, $1/8$ - - -) until the increment or decrement of one least significant bit has been processed. Figure 5-21, below, shows the signals.



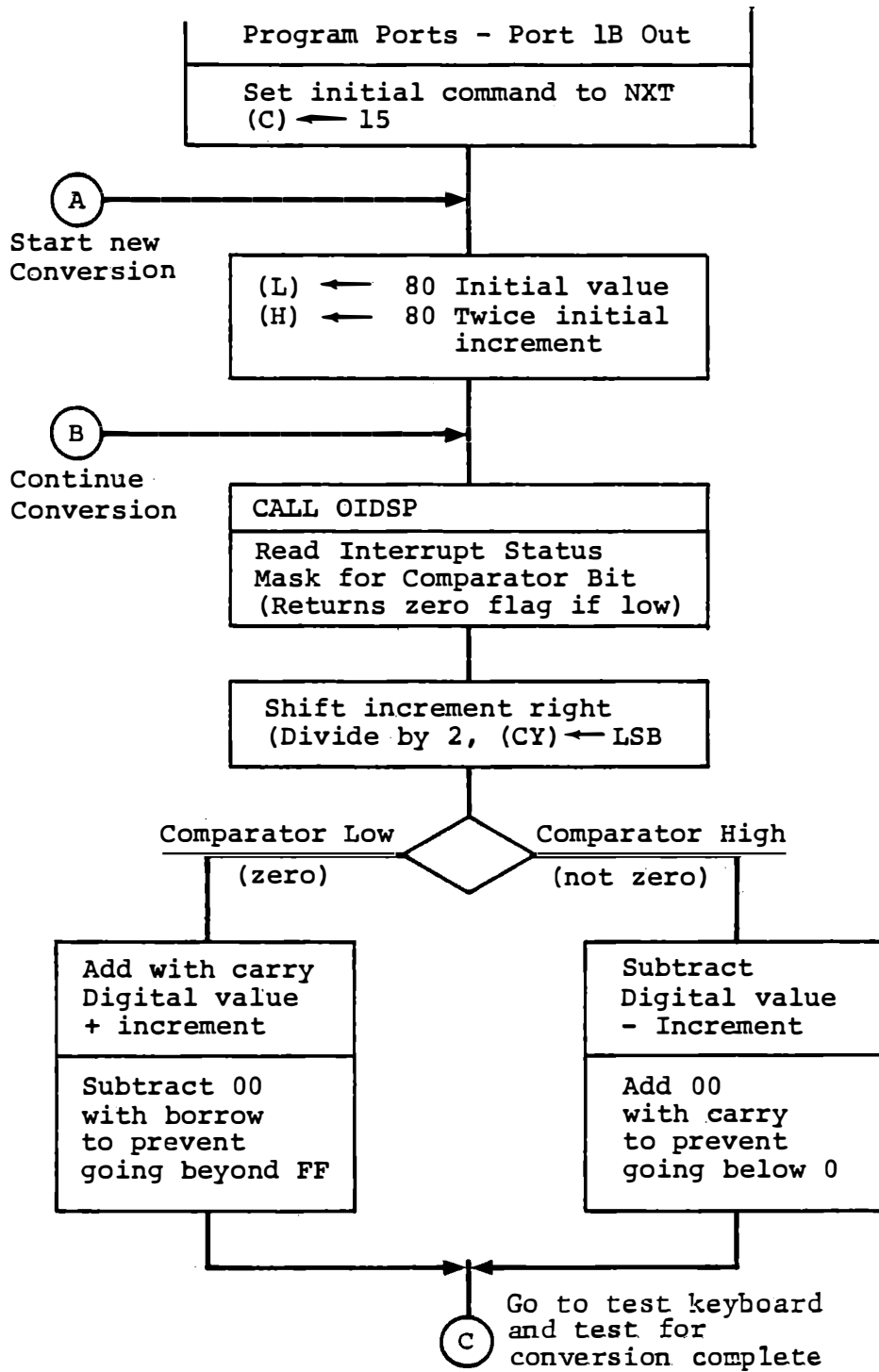
Successive Approximation Signals

Figure 5-21

After all of the successively smaller increments or decrements down to one least significant bit have been processed, the digital value is within + or - 1 bit of the analog input. If the process were stopped here, the result would always be an odd number, since in the last step the digital value, up to here an even number, was either increased or decreased by one. The procedure of Figures 5-22 and 5-23 avoids this problem by shifting the increment right just before adding or subtracting, and doing an ADC if the comparator is low, but SUB if it is high. Thus when the increment reaches zero, the digital value may still be increased, but will not be decreased. The result is therefore within $\pm 1/2$ bit of the input value.

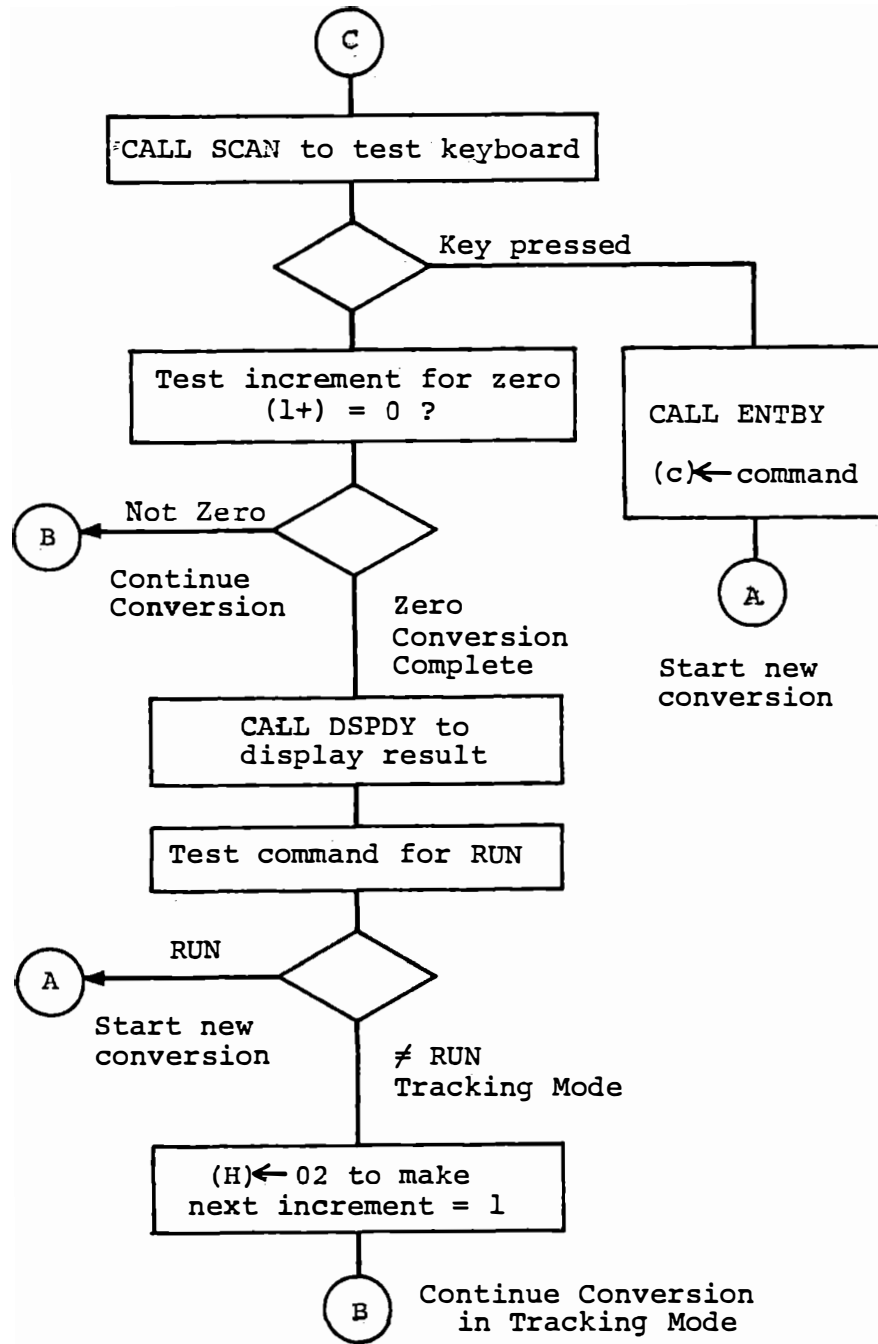
As in the preceding program, we will enter a tracking mode when the conversion is complete, and start a new conversion when NEXT is pressed. With the RUN key, however, we will start a new conversion as soon as the old conversion has been completed.

ANALOG TO DIGITAL INPUT



Successive Approximation Voltmeter

Figure 5-22a



Successive Approximation Voltmeter (continued)

Figure 5-22b

SUCCESSIVE APPROXIMATION VOLTMETER

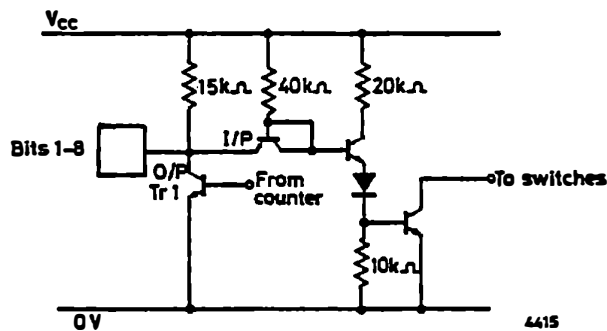
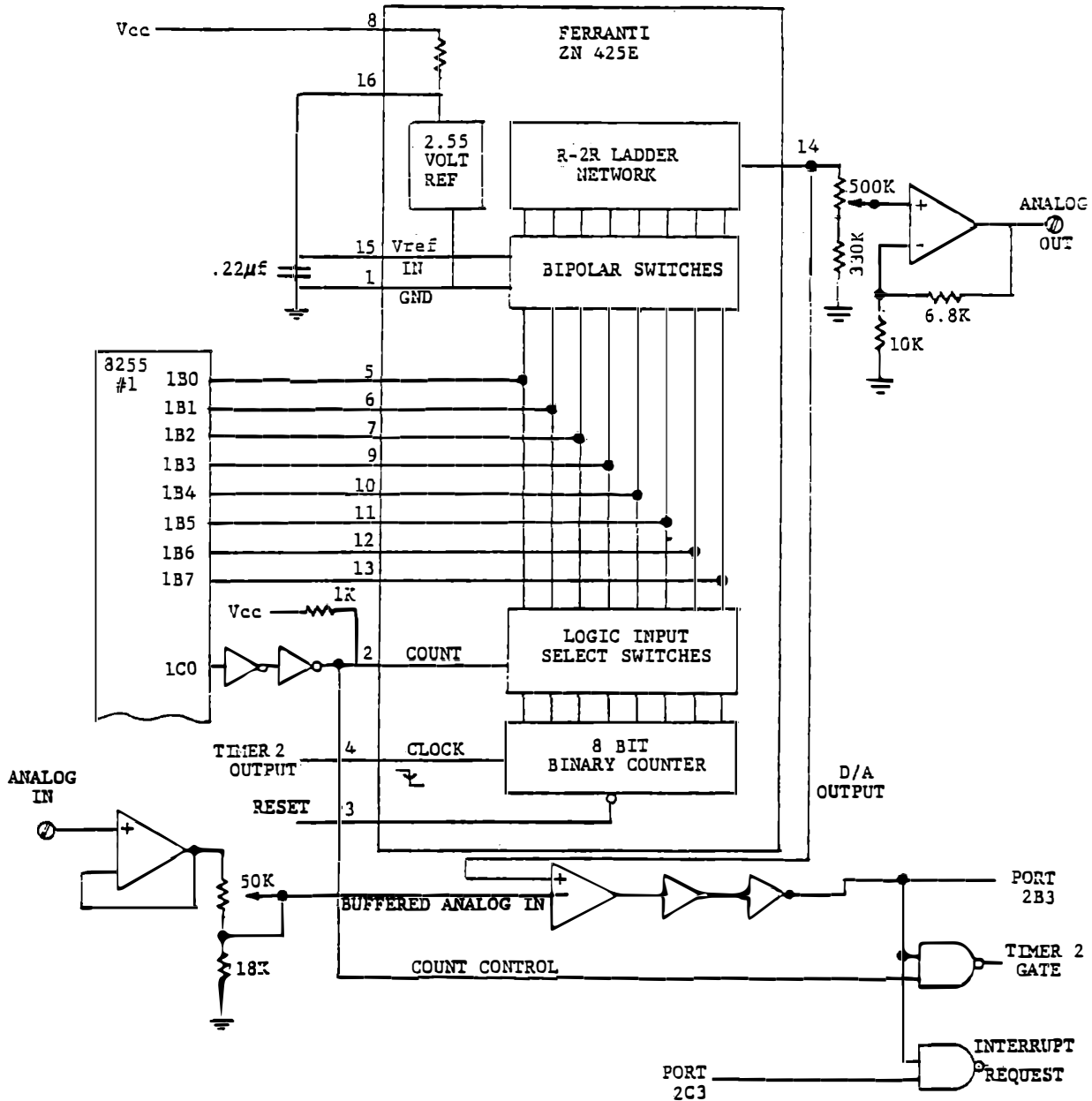
A D D R		CODE										
CODING SHEET	8 20	0	3F	MVI	A,	80						
		1	80									
		2	D3	OUT	CNT1							
		3	07									
		4	3E	MVI	A,	92						
		5	92									
		6	D3	OUT	CNT2							
		7	0F									
MICROCOMPUTER TRAINING SYSTEM	820	8	3E	MVI	A,	15					Make command NXT	
		9	15								for initial conversion	
	820	A	4F	MOV	C,	A					(C) ← command	
	820	B	21	LXI	H,	8080					(L) ← initial value	
		C	80								(H) ← 2x initial	
		D	80								increment	
	820	E	CD	CALL	OIDS	SP						
		F	CO									
	INTEGRATED COMPUTER SYSTEMS	821	0	82								
			1	7C	MOV	A,	H					
		2	1F	RAR								
		3	67	MOV	H,	A						
		4	7D	MOV	A,	L						
		5	C2	JNZ	821E							
		6	1E									
		7	82									
		8	8C	ADC	H							
		9	DE	SBI	00							
	A	00										
	B	C3	JMP	8221								
	C	21										
	D	82										
	821	E	94	SUB	H							
		F	CE	ACI	00							
	822	0	00									
	822	1	6F	MOV	L,	A						
		2										
		3										
		4										
		5										
		6										
		7										
		8										
											NOTE: OIDS P (FIG 5-18, c, d)	
											IS REQUIRED	
											Figure 5-23a	

SUCCESSIVE APPROXIMATION VM (continued)

A D D R		CODE													
8		0													
1															
CODING SHEET	822	2	CD	CALL	SCAN	Test keyboard									
		3	57												
		4	02												
		5	DA	JC	823B	Jumps to CALL ENTBY									
		6	3B			if key pressed									
		7	82												
		8	7C	MOV	A, H	test increment									
		9	B7	ORA	A	low zero									
		A	C2	JNZ	820E	Jumps to CALL ODISP									
		B	0E			if conversion									
MICROCOMPUTER TRAINING SYSTEM		C	82			not complete									
		D	CD	CALL	DSPDY	Display result									
		E	F2			when conversion									
		F	82			is complete									
	823	0	79	MOV	A, C	test command									
		1	FE	CPI	14	low RUN									
		2	14												
		3	CA	JZ	820B	if RUN as to									
		4	0B			start new									
		5	82			conversion									
INTEGRATED COMPUTER SYSTEMS		6	26	MVI	H, 02	To force next									
		7	02			increment to 01									
		8	C3	JMP	820E	continue in									
		9	0E			tracking mode.									
		A	82												
	823	B	00	NOP											
		C	CD	CALL	ENTBY										
		D	36												
		E	03												
		F	00	NOP											
824	0	C3	JMP	820A											
824	1	0A													
824	2	82													
	3														
	4														
	5														
	6														
	7														
	8														

Figure 5-23b

ANALOG TO DIGITAL INPUT



Logic Input Circuitry
Ferranti A/D Logic

Figure 5-24

5.4 AUTOMATIC A/D INPUT

In Section 5.3, we have been using the microprocessor's arithmetic capability to control the D/A output for comparison with an input. Generating a voltage ramp requires such simple logic that our Ferranti ZN 425E D/A converter includes the necessary counter and switches.

Two control inputs of the 425 must be operated for automatic A/D input, and a clock signal must be provided for the internal counter. Figure 5-24 shows the logic of the 425. The internal counter is driven by its clock signal and cleared by its reset signal. If the "count" control signal is low, as we have been using it, the counter outputs are isolated by the open collector transistor switches. Then the R-2R ladder is controlled by the data output from Port 1B. If the "count" control signal is high, internal gating allows the internal counter to control the data on the I/O port and the switches of the R-2R ladder. Port 1B must be programmed for input, and the data from the A/D converter can be read there.

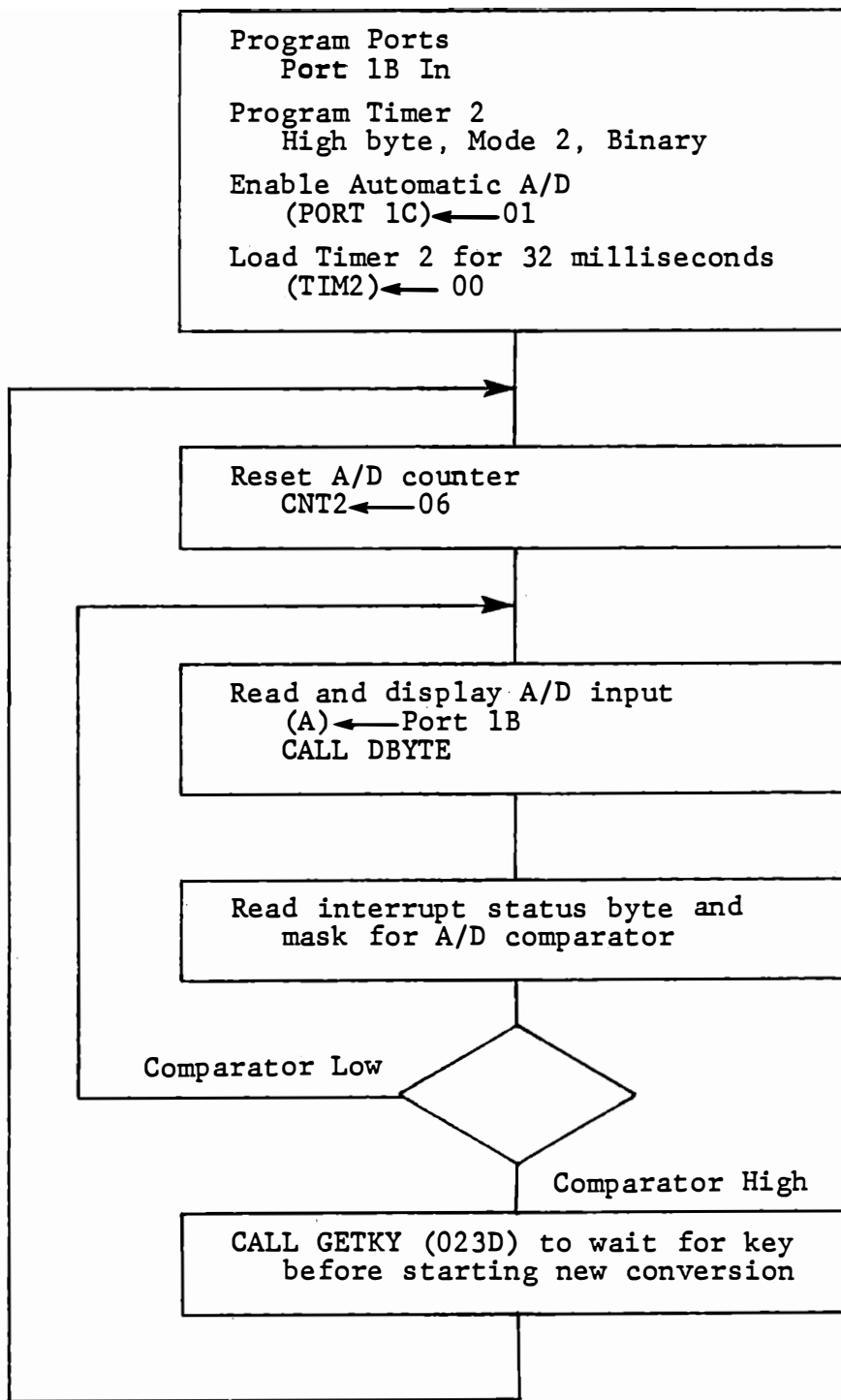
ANALOG TO DIGITAL INPUT

This page intentionally left blank.

The clock for the internal counter is taken from timer 2. When it generates clock pulses, the Ferranti counts up, generating a voltage ramp at the D/A output. If it exceeds the analog input signal, the comparator output goes high. This signal is NAND gated with count control, so that when both signals are high the timer 2 gate input becomes low, inhibiting further counting. The Ferranti's counter now contains the digital value corresponding to the analog input voltage. This value, available at port 1B, is held until the counter is reset.

The reset input to the 425 counter is generated when the A/D comparator interrupt is enabled or disabled. The D/A output becomes zero, so the comparator output goes low and clears the interrupt. Therefore this interrupt behaves as though it had a latch cleared by the enable or disable interrupt command. There is an important constraint on treating this as a latch, as we shall see in the following exercise. First we will demonstrate the automatic ramp generation.

ANALOG TO DIGITAL INPUT



Automatic A/D Input

Figure 5-25

5.4.1 Reading A/D Input

EXERCISE

Figures 5-25 and 5-26 show a program to operate the Ferranti 425 in its automatic A/D input mode. Port 1B is programmed for input and Port 1C0 is set high to enable the counter. Timer 2 provides the clock to the Ferranti: here it is set to a maximum delay to make the ramp visible.

A new conversion is started by resetting the A/D counter. Since we are not using an interrupt system, the disable command is given by writing 06 to CNT2. The content of the A/D counter is read and displayed, and then the comparator input is tested by:

IN	PORT2B	Read interrupt status
ANI	08	Mask for A/D comparator

The input, display and test procedure is repeated until the comparator is high. The increasing ramp is readily observed in the display, and can also be seen by connecting your voltmeter to ANALOG OUT. When the comparator becomes high the program waits for a key to be pressed, and then starts a new conversion.

AUTOMATIC A/D INPUT

A D D R		CODE					
CODING SHEET	8 20	0	3E	MVI	A, 82		Program Ports
		1	82				Port B In
		2	D3	OUT	CNT1		for A/D
		3	07				
		4	3E	MVI	A, 92		
		5	92				
		6	D3	OUT	CNT2		
		7	0F				
		8	3E	MVI	A, A4		Program Timer 2
		9	A4				4.1818 use
MICROCOMPUTER TRAINING SYSTEM	A	D3	OUT	TIMCT			Mode 2
	B	17					Binary
	C	3E	MVI	A, 01			set Port PC0 high
	D	01					to enable
	E	D3	OUT	PORT1C			automatic A/D
	F	06					
	8 21	0	AF	XRA	A		Load Timer 2
		1	D3	OUT	TIM2		for 32 milliseconds
		2	16				
	8 21	3	3E	MVI	A, 06		Reset A/D Counter
	4	06				Disable interrupt	
	5	D3	OUT	CNT2			
	6	0F					
8 21	7	DB	IN	PORT1B		Read A/D input	
	8	05					
	9	CD	CALL	DBYTE		Display A/D	
A	95						
B	02						
C	DB	IN	PORT2B			Restart interrupt	
D	0D					status byte	
E	E6	ANI	08			Mask for	
F	08					A/D comparator	
8 22	0	CA	JZ	8217		Loop to read	
	1	17				and display input	
	2	82				unless comparator	
	3	CD	CALL	GETKEY		high. Then wait	
	4	3D				for key, and	
	5	02				start a new	
	6	C3	JMP	8213		conversion.	
	7	13					
	8	82					

Figure 5-26

Now interchange the test of the comparator and the input and display of the A/D value.

```

CNVRT      MVI      A,06          Reset A/D counter
           OUT      CNT2
TEST       IN       PORT2B       Wait for comparator
           ANI      08           to become high
           JZ       TEST
           IN       PORT1B       Read A/D
           CALL     DBYTE        Display voltage
           CALL     GETKY        Wait for key
           JMP      CNVRT

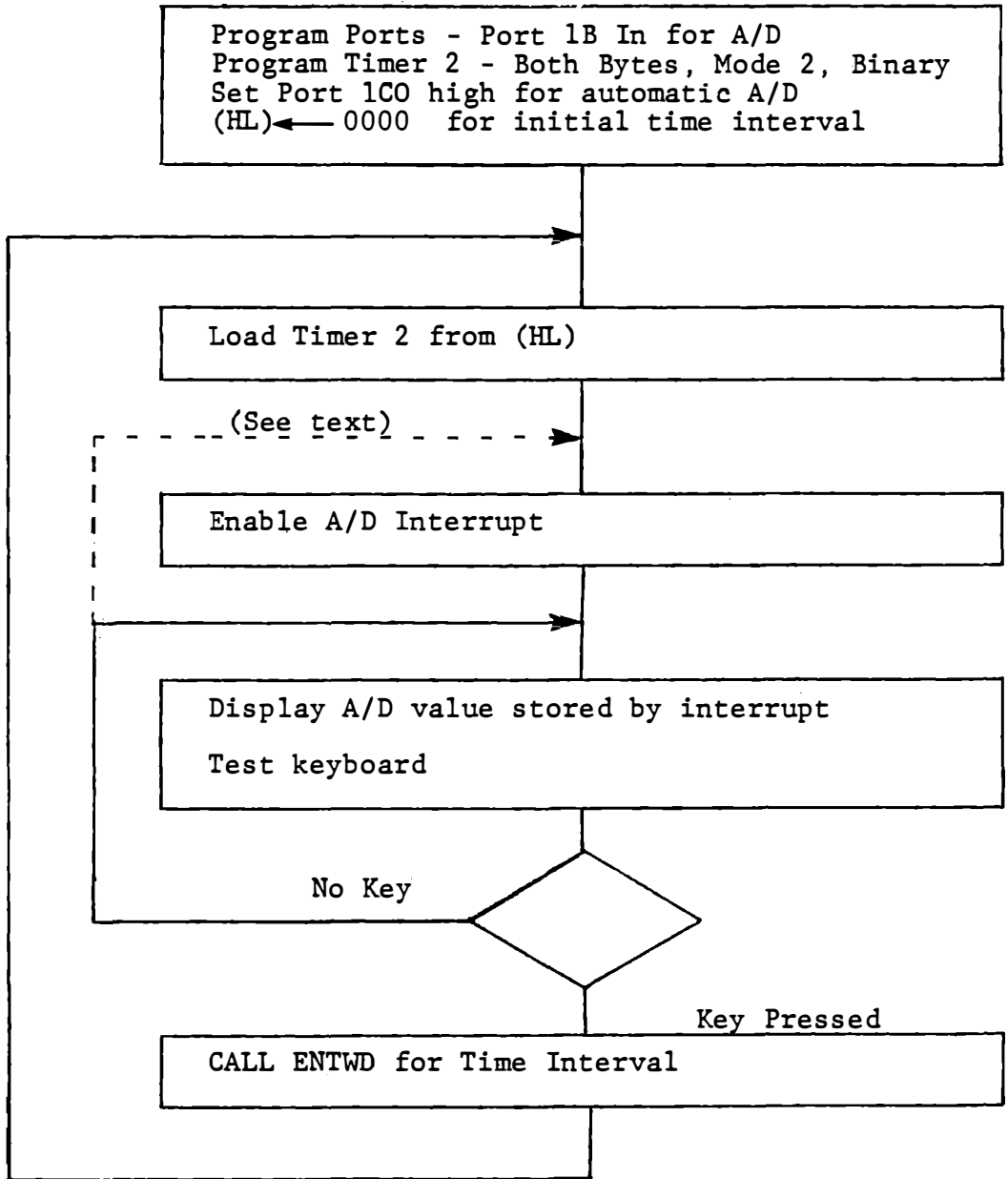
```

It appears that this should display only the final voltage, since we read it when the comparator is high. Instead, it sometimes displays 00! This demonstrates the constraint previously mentioned. The A/D comparator does not reset instantly, as does a latch. After the reset, the D/A output goes low very quickly, but the comparator takes several microseconds to respond. Therefore reading the comparator immediately after a reset may find it still high, and the program above falsely supposes that the conversion is finished. If you press a key quickly twice in succession, or press it again as soon as 00 is displayed, then the program will make the conversion properly.

This page intentionally left blank.

A better scheme is to rearrange the program again, placing the reset before CALL GETKY. Then, if you like, replace CALL GETKY with CALL DELAY (0236) for a fully automatic voltmeter. You can speed it up by loading Timer 2 with 02 instead of 00, to give a clock interval of 125 microseconds instead of 32 milliseconds. In the solution given in Figure 5-26, this can be done by deleting the XRA A instruction before loading Timer 2. We shall investigate the effect of the clock interval in the following exercise.

ANALOG TO DIGITAL INPUT



RST6 Interrupt Service

```

PUSH PSW only
Read and Store A/D Input
Reset A/D Counter
POP PSW, EI, RET
  
```

A/D Input with Interrupt

Figure 5-27

5.4.2 A/D Input with Interrupt

EXERCISE

Since the A/D comparator generates an interrupt signal, the input can be accepted with an interrupt service routine. In this exercise we shall demonstrate two risks with the interrupt service and a technique that protects against both.

The main program (Figure 5-27) programs Timer 2 for two bytes and initially loads it for 32 milliseconds. The A/D interrupt is enabled. Now when a conversion is complete the interrupt service routine reads the A/D input, stores the value in memory, and resets the A/D counter. The main program displays the stored value. Now if a key is pressed ENTWD accepts a new interval for the clock to the Ferranti and loads Timer 2. Otherwise it merely displays the voltage repeatedly.

Do you see the danger in this program? Write and test the program and try to identify the problem that may occur in the solution from Figure 5-28.

Interrupt service only provides the time for POP PSW and EI after resetting the A/D counter. If the comparator does not respond in that time, the interrupt is still present when RET is executed. The main program is never allowed to execute an instruction after enabling the A/D interrupt. (This problem is not certain to occur, however.)

A/D INPUT WITH INTERRUPT

A D D R		CODE										
CODING SHEET	8	20	0	3E		MVI	A,	82			Program Ports	
			1	82							Port 1B In	
			2	D3		OUT	CNT	1				
			3	07								
			4	3E		MVI	A,	92				
			5	92								
			6	D3		OUT	CNT	2				
			7	0F								
			8	3E		MVI	A,	B4			Program Timer 2	
			9	B4							Both latches	
MICROCOMPUTER TRAINING SYSTEM		A	D3		OUT	TIM	CT				Mode 2	
			B	17							Binary	
			C	3E		MVI	A,	01				
			D	01								
			E	D3		OUT	PORT	1C				
			F	06								
		8	21	0	21		LXI	H,	0000			Initial value
				1	00							for clock interval
				2	00							
			821	3	7D		MOV	A,	L			Load timer 2
			4	D3		OUT	TIM	2			with clock	
			5	16							interval for	
			6	7C		MOV	A,	H			A/D counter	
			7	D3		OUT	TIM	2				
INTEGRATED COMPUTER SYSTEMS			8	16								
			9	C3		JMP	8240				Jump past	
			A	40							interrupt service	
			B	82								
			C									
			D									
			E									
			F									
		8	0									
			1									
		2										
		3										
		4										
		5										
		6										
		7										
		8										

Figure 5-28a

		A	D	D	R	CODE	A/D INPUT - INTERRUPT, MAIN LOOP				
CODING SHEET	8 23	0	F5		PUSH	PSW					
		1	DB		IN	PORT 1B					Read A/D input
		2	05								
		3	32		STA	8300					Store A/D value
		4	00								
		5	83								
		6	3E		MVI	A, 07					Reset A/D counter
		7	07 *								Change to 06
		8	D3		OUT	CNT 2					
		9	0F								
MICROCOMPUTER TRAINING SYSTEM	A		F1		POP	PSW					Insert CALL DELAY
	B		FB		EI						Resume POP PSW
	C		C9		RET						according to text
	D										
	E										
	F										MAIN LOOP
	8 24	0	3E		MVI	A, 08					Enable A/D
		1	08								interrupt
		2	D3		OUT	PORT 2C					
		3	0E								
	4	3A		LDA	8300					Disable voltage	
	5	00								stored bus	
	6	83								interrupt service	
	7	CD		CALL	DBYTE						
	8	95									
	9	02									
INTEGRATED COMPUTER SYSTEMS	A		DB		IN	PORT 0A					Test bus boards
	B		00								(FF if no bus)
	C		3C		INR	A					
	D		CA		JZ	8244					Use initial test
	E		44 *								Change to 8240
	F		82								according to text
	8 25	0	CD		CALL	ENTWD					
		1	46								(HL) ← new time
		2	03								interval
		3	C3		JMP	8213					Go to load timer
	4	13									
	5	82									
	6										
	7										
	8										

Figure 5-28b

ANALOG TO DIGITAL INPUT

This problem is easily solved. Before POP PSW, EI, RET insert CALL DELAY (0236) in the interrupt service routine. This monitor subroutine generates a delay of about one millisecond, using only the A register. Now there is plenty of time for the A/D comparator to settle. The voltmeter program should now operate correctly, although it is very slow because of the long clock interval. Try shorter intervals: 4000, 1000, 0400, 0100, 0050. Adjust the OPTO SENSE pot and see the display follow it.

If you make the clock interval short enough, and the voltage low enough, the main loop will not be able to operate because a correct A/D conversion is completed before DELAY returns. Again the interrupt is already there before EI, RET is executed.

To solve both problems, reset the A/D counter with a disable command instead of enable:

```
        MVI    A,06
        OUT    CNT2
```

In the main program, enable the A/D interrupt by writing 08 to Port 2C, and include this in the short loop. Now an interrupt can occur only once for each pass through the main loop, so it is guaranteed to run, with or without the delay in interrupt service.

The slow response of the comparator introduces another problem which can be investigated with this program. With a fast clock, several counts may occur after the D/A output actually exceeds the input voltage, but before the comparator responds and inhibits the clock. This leads to slightly different results, depending on the clock

rate. For any given rate, however, the error is fixed and can be compensated for. In our experiments it is small enough to be ignored. We shall generally use a time interval of 20 (hex), which gives a 16 microsecond clock.

The LM324N op-amp was selected here because it is able to operate on a single +5 volt supply and still work with signals down to zero volts. Faster op-amps cannot handle signals as low as the negative supply voltage, so would require an additional power supply. A specialized voltage comparator circuit such as National Semiconductor LM339N would provide fast response with the single supply voltage, but would not serve for the other op-amp functions needed here.

5.5 DIGITAL NOISE FILTER

If the voltage at the analog input is very close to a particular D/A output value, it is likely that the least significant bit of the measured voltage will change from one reading to another. If noise is present on the analog signal, several bits may change. A filter is needed to reduce the noise. We connected a capacitor at the input (Figure 5-10) for this purpose.

Filtering can also be done by digital processing. An excellent technique for estimating the present value of a signal that is changing with time, but also includes noise is to calculate a running average that gives less and less weight to older data measurements.

$$E_i = f_0 V_i + f_1 V_{i-1} + f_2 V_{i-2} + \dots$$

Where the $V(i)$ are successive measurements and the $f(j)$ are weights applied to them. Obviously, the weights must be selected so that if V does not change $E(i)$ will equal V . This is achieved by the following expression, which also minimizes the data to be stored.

$$E_i = fV_i + (1-f) E_{i-1}$$

Storage is required only for the present estimate, $E(i)$, as a variable and a single value of f as a constant, yet is exactly equivalent to the infinite series of the first expression with:

$$\begin{aligned}
 f_0 &= f \\
 f_1 &= f(1-f) \\
 f_2 &= f(1-f)^2 \\
 f_3 &= f(1-f)^3 \quad \text{etc.}
 \end{aligned}$$

If we were to use $f = 0.25$, these would be:

$$\begin{aligned}
 f_0 &= 0.25 \\
 f_1 &= 0.1875 \\
 f_2 &= 0.140625 \\
 f_3 &= 0.10546875 \quad \text{etc.}
 \end{aligned}$$

Greater values of f lead to faster response of the estimate to new data, while smaller values give more noise filtering.

5.5.1 Filter Program Algorithm

The filter calculations will be performed by a subroutine FILTR. To simplify the calculations, we will restrict the value of f to $1/2$, $1/4$, $1/8$ or $1/16$. With such power of 2 fractions, the multiplication and divisions required become simple shifts by n bits, where $f = 1/2^n$. The expression to be calculated then becomes:

$$E_i = \frac{V_i + (2^n - 1) E_{i-1}}{2^n}$$

Data read from the A/D input have single byte precision, but to avoid the loss of the less significant bits of each measurement, we will carry out calculations and store results with two byte precision. Rather than storing $E(i)$ after each new input and calculation, we will store $2^n E_i$, which will be used at the next calculation.

ANALOG TO DIGITAL INPUT

Suppose that we choose $f = 1/4$, or $n=2$. Then the stored value is $4E_{i-1}$, which represents $4E_{i-1}$ when a new measured value $V(i)$ is obtained. The algorithm is shown below. (For convenience in notation let $m = 2^n$).

STEP	RESULT	
	$m=4$	general
Recover stored data	$4E_{i-1}$	mE_{i-1}
Multiply by m	$16E_{i-1}$	$m^2E_{i-1} \quad 1)$
Subtract stored data	$12E_{i-1}$	$m(m-1)E_{i-1} \quad -1)$
Divide by m	$3E_{i-1}$	$(m-1)E_{i-1} \quad -1)$
Add new measurement	$4E_i$	mE_i
Store result		
Divide by m	E_i	E_i

The multiplications and divisions by m are simple shifts of n bits, so the constant to be stored is n. To use this for single byte precision for the measured value and double byte precision for the arithmetic, n must be no greater than 4 ($m=16$), since one intermediate result has the data shifted left by 2n bits from the measured value. In fact, $n=4$, making $f = 1/16$, gives rather slower response than we will generally want.

5.5.2 Program Definitions

Subroutine FILTR and a local subroutine SHFTN are defined below and depicted in Figure 5-29.

The program in Figure 5-30 displays both filtered voltage, $E(i)$, and inputted voltage, $V(i)$. The leftmost two hex digits are $E(i)$ and the next two digits are $V(i)$. The value for n (1,2,3 or 4) is entered via the keyboard and displayed in the rightmost two display digits.

5.5.2.1 Subroutine FILTR

Calculates estimated value of a variable with repetitive measurements containing noise.

Enter with new data in register A and (HL) addressing a four byte memory area:

((HL)) = n
 ((HL) + 1,2) = $2^n E_i - 1$ (used in calculation)
 ((HL) + 3) = E_i (previous result)

Calculates and stores (in the same two locations)

$$2^n E_i = (2^n - 1) E_i + V_i$$

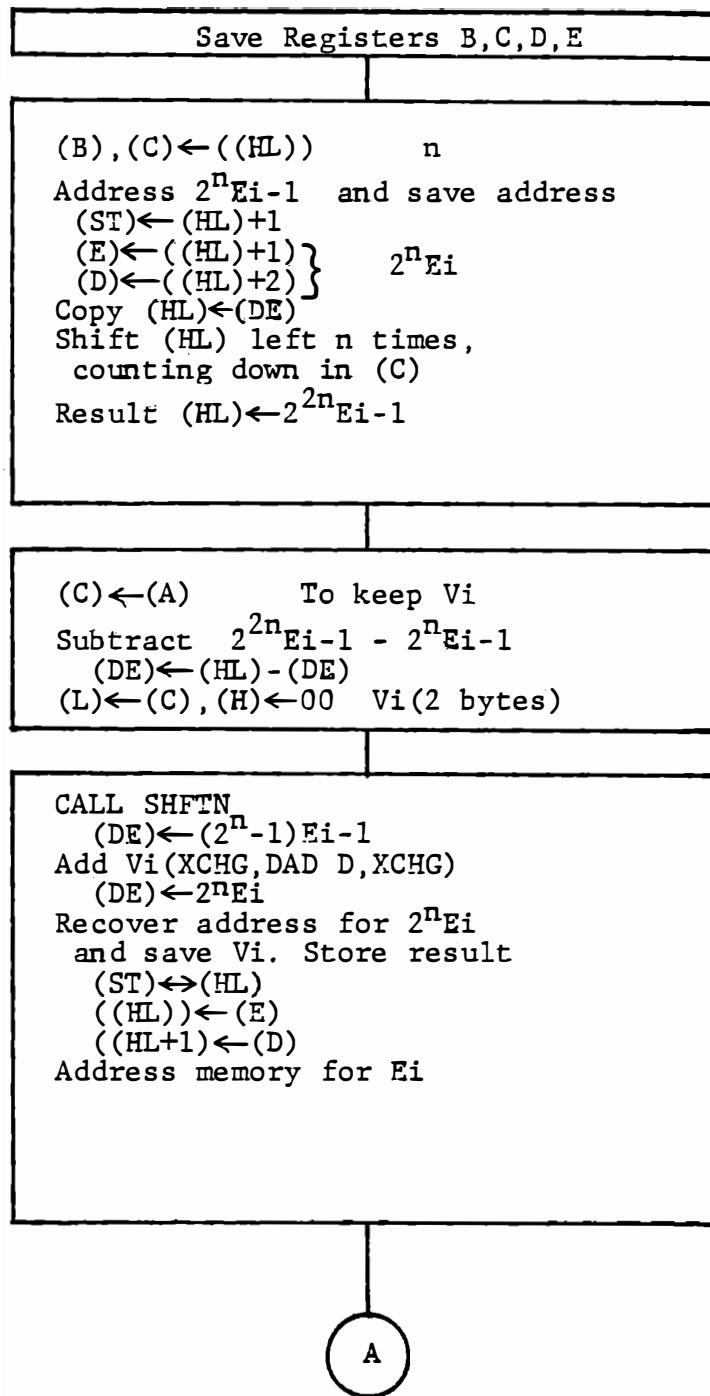
Returns (A) = $E_i = \frac{2^n - 1) E_i + V_i}{2^n}$

(H) = E_i

(L) = V_i

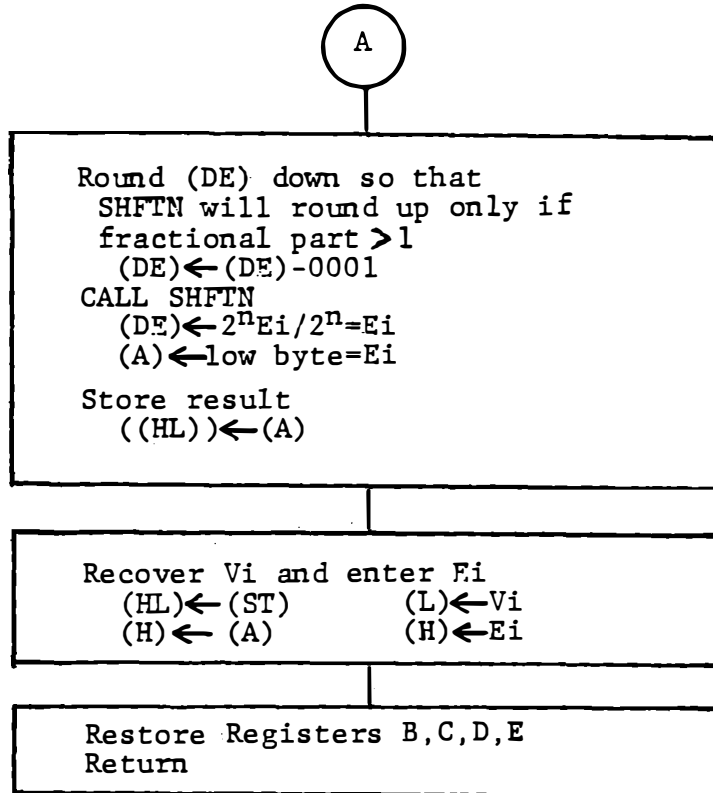
Registers B,C,D and E are preserved.

ANALOG TO DIGITAL INPUT



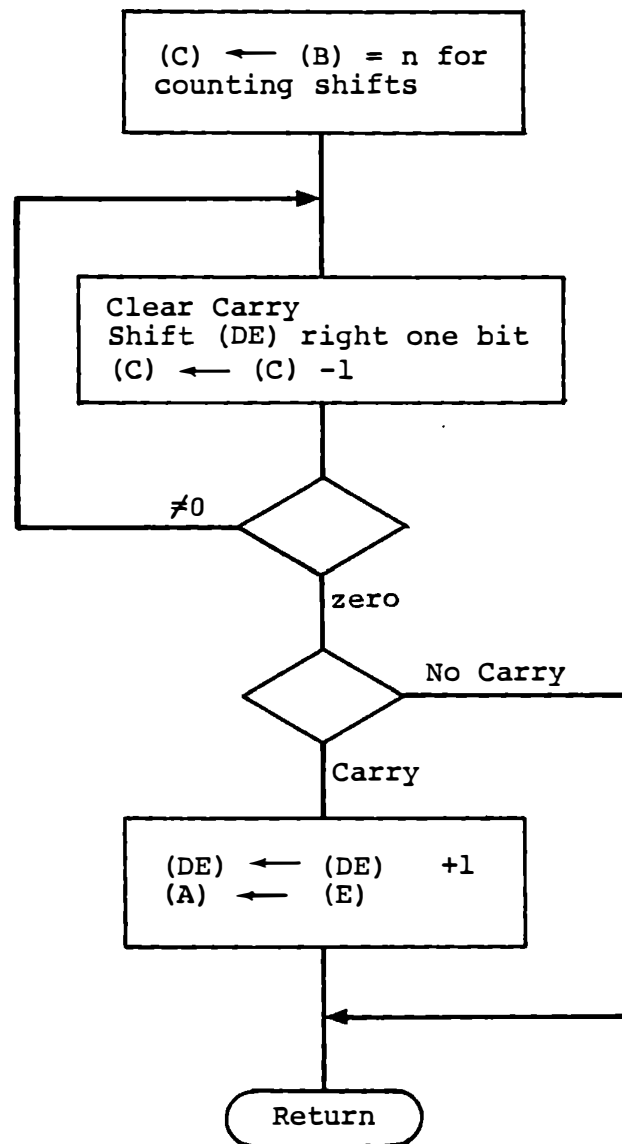
Subroutine FILTR

Figure 5-29a



Subroutine FILTR (continued)

Figure 5-29b



Subroutine SHFTN

Figure 5-29c

5.5.2.2 Subroutine SHFTN

Shifts a data word right n bits with rounding.

Enter with (B) = n

(DE) = data word

Return with (B) = n (unchanged);

(DE) = data word/ 2^n ;

(A) = less significant byte of data word.

Register C is cleared. Registers H and L are preserved. The carry flag is set if roundup has occurred.

Roundup occurs if the highest bit shifted out is 1. That is, if the fractional part of $(\text{data}/2^n)$ is greater than or equal to one half.

ANALOG TO DIGITAL INPUT

A comment on rounding of the calculations is necessary, because the requirement is not at all obvious. When the division of $m(m-1)E_{i-1}$ is done, round up should occur if the highest bit shifted out is 1; that is, if the fractional part of the result is equal to or greater than $1/2$. If this roundup is not done, the final result, with constant input, is always one less than the measured value. When the division of m^{E_i} is done to obtain $E(i)$ for display, the roundup must occur only if the fractional part is greater than $1/2$. Otherwise the filter can never reach a value of zero. This is handled by a DCX D before calling SHFTN the second time.

Modify the A/D input with interrupt program (Figure 5-28) to use FILTR. Display both the measured value and the filtered value, when an interrupt occurs. Use the keyboard input to accept a new value of n for the filter. When a new value is entered, clear the existing $mE(i)$ from memory, to ensure that the high bits (beyond the precision being used) will not be left with data.

Test the program with the existing connections (Figure 5-10). Then remove the filter capacitor and test it again.

A/D INPUT WITH FILTER - INITIALIZE

A D D R		CODE							
CODING SHEET	8	20	0	3E		MVI	A, 82	Initialization identical to Figure 5-28a except as marked	
			1	82					
			2	D3		OUT	CNT1		
			3	07					
			4	3E		MVI	A, 92		
			5	92					
			6	D3		OUT	CNT2		
			7	0F					
			8	3E		MVI	A, B4		
			9	B4					
MICROCOMPUTER TRAINING SYSTEM		A		D3		OUT	TIMCT		
			B						
			C		3E		MVI	A, 01	
			D		01				
			E		D3		OUT	PORTIC	
			F		06				
	INTEGRATED COMPUTER SYSTEMS	8	21	0	21		LXI	H, 0020	Fixed value for clock interval - 16 μ sec clock
				1	20	*			
				2	00				
				3	7D		MOV	A, L	
			4	D3		OUT	TIM2		
			5	16					
			6	7C		MOV	A, H		
			7	D3		OUT	TIM2		
			8	16					
			9	C3		JMP	8250	Jump past interrupt service	
	A			50	*				
		B		82					
	C								
	D								
	E								
	F				*	CHANGED FROM FIGURE 5-28a			
	8		0						
			1						
			2						
			3						
			4						
			5						
			6						
			7						
			8						

Figure 5-30a

A/D INTERRUPT WITH FILTR

		A	D	D	R	CODE						
CODING SHEET	8	23	0	F5		PUSH	PSW					
			1	E5		PUSH	H					
			2	D5		PUSH	D					
			3	C5		PUSH	B					
			4	DB		IN	PORT	1B				Read A/D value
			5	05								(A) ← Vi
			6	21		LXI	H,	8300				Address memory
			7	00								locations for FILTR
			8	83								
			9	CD		CALL	FILTR					(A) ← Ei
MICROCOMPUTER TRAINING SYSTEM	A		70								(H) ← Ei	
	B		82								(L) ← Ei	
	C		CD		CALL	DWORD					Display Ei at left	
	D		D1								and Vi	
	E		02									
	F		3E		MVI	A,	06					Reset and disable
	8	24	0	06								A/D interrupt
			1	D3		OUT	CNT	2				
			2	0F								
			3	C1		POP	B					
INTEGRATED COMPUTER SYSTEMS			4	D1		POP	D					
			5	E1		POP	H					
			6	F1		POP	PSW					
			7	FB		EI						
			8	C9		RET						
			9									
		A										
		B										
		C										
		D										
	E											
	F											
	8		0									
			1									
			2									
			3									
			4									
			5									
			6									
			7									
			8									

Figure 5-30b

A/D INPUT WITH FILTR - MAIN LOOP

		A	D	D	R	CODE							
CODING SHEET	8 25	0	3E			MVI	A,	01					Initial value for n in FILTR
		1	01										
	8 25	2	F3			DI							No interrupts while
		3	21			LXI	H,	8300	0				changing data
		4	00										and FILTR
		5	83										
		6	77			MOV	M,	A					(8300) ← n
		7	AF			XRA	A						} (Clear 2 nd E _{i-1} (8301, 8302)
		8	23			INX	H						
		9	77			MOV	M,	A					
	A	23			INX	H							
	B	77			MOV	M,	A						
	C	FB			EI								Enable A/D
MICROCOMPUTER TRAINING SYSTEM	8 25	D	3E			MVI	A,	08					interrupt without
		E	08										resetting counter
		F	D3			OUT	PORT	2C					
MICROCOMPUTER TRAINING SYSTEM	8 26	0	0E										
		1	DB			IN	PORT	0A					Test keyboard
		2	00										(FF if no key)
		3	3C			INR	A						
		4	CA			JZ	825D						Reenable A/D
		5	5D										until key pressed
		6	82										
		7	CD			CALL	ENTBY						Accept new n
		8	36										(must be
		9	03										1, 2, 3 or 4)
INTEGRATED COMPUTER SYSTEMS	A	7D				MOV	A,	L					(A) ← n
	B	C3				JMP	8252						Go to store n
	C	52											and clear 2 nd E _i
	D	82											
	E												
	F												
	8	0											
		1											
	2												
	3												
	4												
	5												
	6												
	7												
	8												

Figure 5-30c

SUBROUTINE FILTR

	A	D	D	R	CODE									
CODING SHEET	8	27	0		D5	PUSH	D						Save (DE)	
			1		C5	PUSH	B						Save (BC)	
			2		46	MOV	B, M						(B) ← n	
			3		48	MOV	C, B						(C) ← n	
			4		23	INX	H						Address $2^m E_{i-1}$	
			5		E5	PUSH	H						Save address	
			6		5E	MOV	E, M						}	
			7		23	INX	H'						{(DE) ← $2^m E_{i-1}$	
			8		56	MOV	D, M						}	
			9		6B	MOV	L, E							(HL) ← $2^m E_{i-1}$
MICROCOMPUTER TRAINING SYSTEM	A	62			MOV	H, D								
	827	B	29		DAD	H								
		C	0D		DCR	C								
		D	C2		JNZ	P 27 B							{(HL) ← $2^{2^m} E_{i-1}$	
		E	7B										}	
		F	82										}	
	8	28	0		4F	MOV	C, A							(C) ← V_i
			1		7D	MOV	A, L							}
			2		93	SUB	E'							{(DE) ← (HL) - (DE)
			3		5F	MOV	E, A							} = $2^m (2^m - 1) E_{i-1}$
		4		7C	MOV	A, H							}	
		5		9A	SBB	D'							}	
		6		57	MOV	D, A							}	
		7		69	MOV	L, C							}	
		8		26	MVI	H, 00							{(HL) ← V_i	
		9		00									}	
INTEGRATED COMPUTER SYSTEMS	A	CD			CALL	SHFTN							Divide by 2^m	
	B	AD											(DE) ← $(2^m - 1) E_{i-1}$	
	C	P2											}	
	D	EB			XCHG								}	
	E	19			DAD	D							{(DE) ← $V_i + (2^m - 1) E_{i-1}$	
	F	EB			XCHG								} = $2^m E_i$	
	8	0				CONTINUED							AT 8298	
		1												
	2				ENTER	(A)							= V_i (new value)	
	3				(HL)								= n (filter constant)	
	4				(HL) + 1								} $2^m E_{i-1}$	
	5				(HL) + 2								}	
	6				RETURN									
	7				(HL) + 3								= (A) = (H) = E_i	
	8												(L) = V_i	

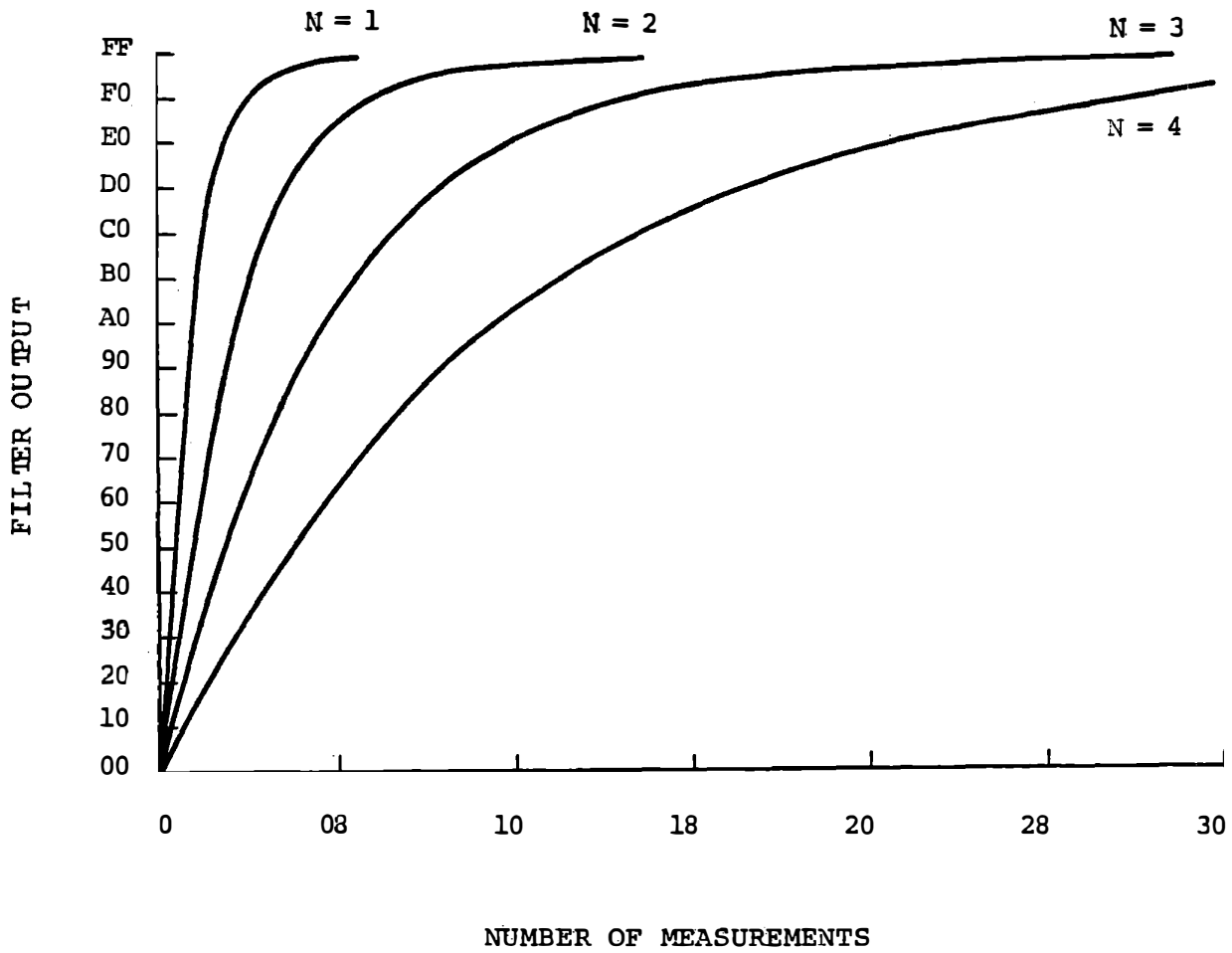
Figure 5-30d

FILTR (continued) AND SHFTN

	A	D	D	R	CODE														
CODING SHEET	8	29	0		E3		X	T	H	L								(HL) ← Address $2^m E_i$	
			1		73		M	O	V		M	,	E						
			2		23		I	N	X		H							{ Store $2^m E_i$	
			3		72		M	O	V		M	,	D						
			4		23		I	N	X		H							Address E_i	
			5		1B		D	C	X		D							For round up only if	
			6		CD		C	A	L	L		S	H	F	T	N		fractional part $> 1/2$	
			7		A0														
			8		82														
			9		77		M	O	V		M	,	A						Store E_i
MICROCOMPUTER TRAINING SYSTEM	A				E1		P	O	P		H							(L) ← V_i	
	B				67		M	O	V		H	,	A					(H) ← E_i	
	C				C1		P	O	P		B							Reset, BCDE	
	D				D1		P	O	P		D								
	E				C9		R	E	T									Exit	
	F				00		N	O	P										
	8	2A	0		48		M	O	V		C	,	B					SHFTN (C) ← n	
			1		AF		X	R	A		A							Loop - clear carry	
			2		7A		M	O	V		A	,	D						
			3		1F		R	A	R										
		4		57		M	O	V		D	,	A					} Shift (DE) right to divide by 2		
		5		7B		M	O	V		A	,	E							
		6		1F		R	A	R											
		7		5F		M	O	V		E	,	A							
		8		0D		D	C	R		C									
		9		C2		J	N	Z		82A1									
INTEGRATED COMPUTER SYSTEMS	A				A1														
	B				82														
	C				D0		R	N	C									Exit if LSB = 0	
	D				13		I	N	X		D							Else round up	
	E				7B		M	O	V		A	,	E					(A) ← less	
	F				C9		R	E	T									significant byte	
	8		0																
			1																
			2																
			3																
		4																	
		5																	
		6																	
		7																	
		8																	

Figure 5-30e

ANALOG TO DIGITAL INPUT



Filter Response for Various N

Figure 5-31

5.5.3 Filter Response

The behavior of a filter may be described in terms of frequency response, or as response to a step function. Each contains the same information, mathematically speaking, but one or the other is more convenient depending on the purpose or the structure of the filter. For a programmed digital filter, it is far easier, in general, to obtain the step function response, since only a two valued input is required. The response of FILTR to a full scale step input is shown in Figure 5-31. You can obtain similar data by a simple process of programmed calls to FILTR. Start with the memory locations for 2^{nE_i} cleared. Load register A with FF (or some other value), call FILTR, and display the result. Wait for a key, then repeat the load and call.

ANALOG TO DIGITAL INPUT

5.6 TEMPERATURE MEASUREMENT

Two important devices are used for temperature measurement in conjunction with microprocessors: thermocouples and thermistors. A thermocouple is a junction of two dissimilar metals. When heated the junction develops a voltage which can be measured and converted to temperature. The thermocouple is highly precise, requires no calibration, and is extremely rugged. It has the disadvantage that the voltage generated is small, and no current may be allowed to flow in its circuit, because resistive voltage drop in the wire would mask the thermal voltage.

5.6.1 Thermistor Characteristics

A thermistor is a semiconductor device that appears as a variable resistance dependent on temperature. Figure 5-32 is a plot of resistance versus temperature for the thermistor supplied with your interface board. The manufacturer's data for this device is

Keystone Part No. RL2012-5506-120-D1

Resistance	10000 ohms at 25 degrees C	Temp. Coefficient
	5506 ohms at 37.8 degrees C	4.84% per degree C
	251 ohms at 125 degrees C	at 25 degrees

The plots of Figure 5-32 were obtained by fitting these data with a curve generated numerically from:

$$R_{i+1} = R_i (1 - C_i \Delta T)$$

$$C_{i+1} = f^{\Delta T} C_i$$

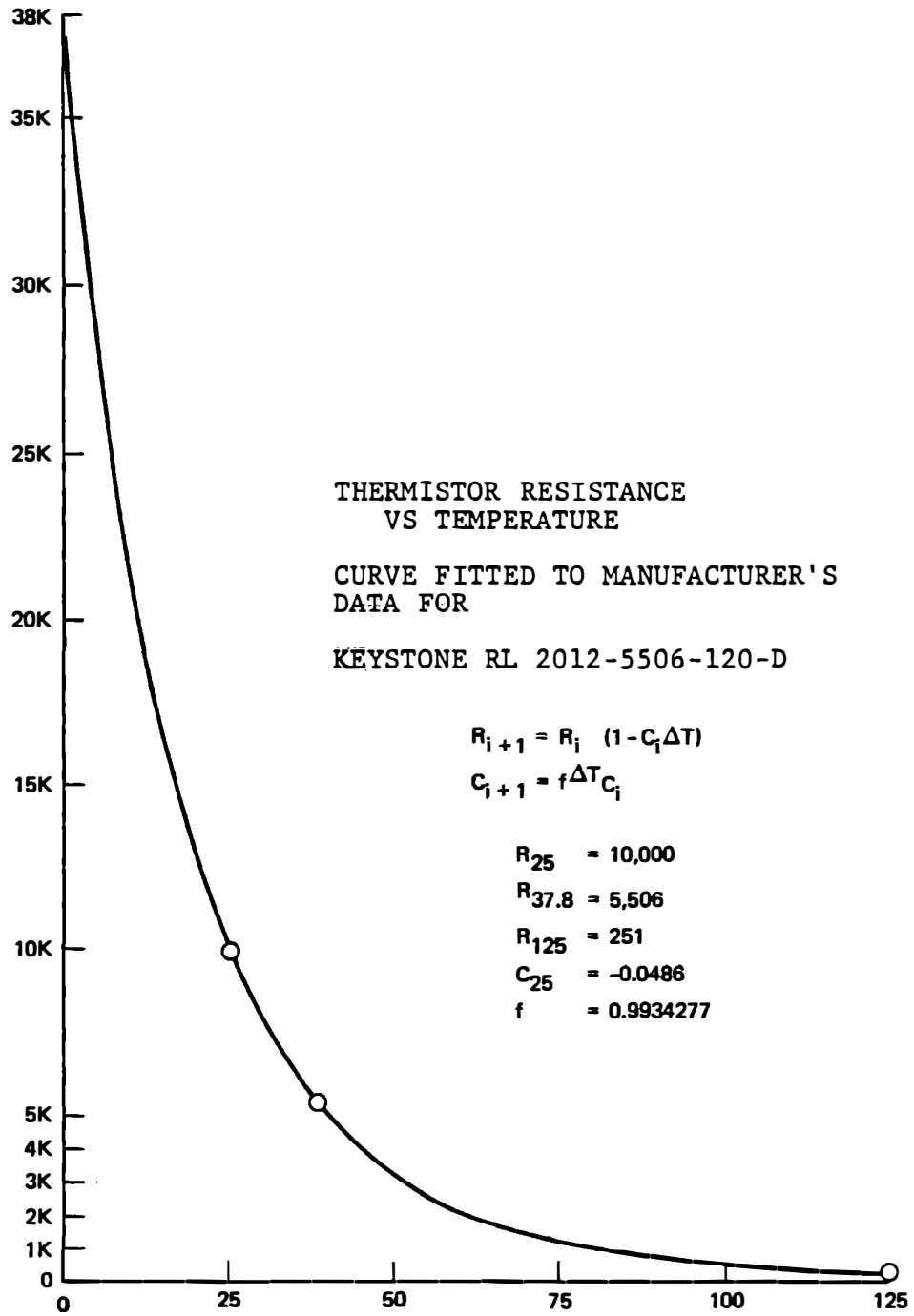


Figure 5-32a

ANALOG TO DIGITAL INPUT

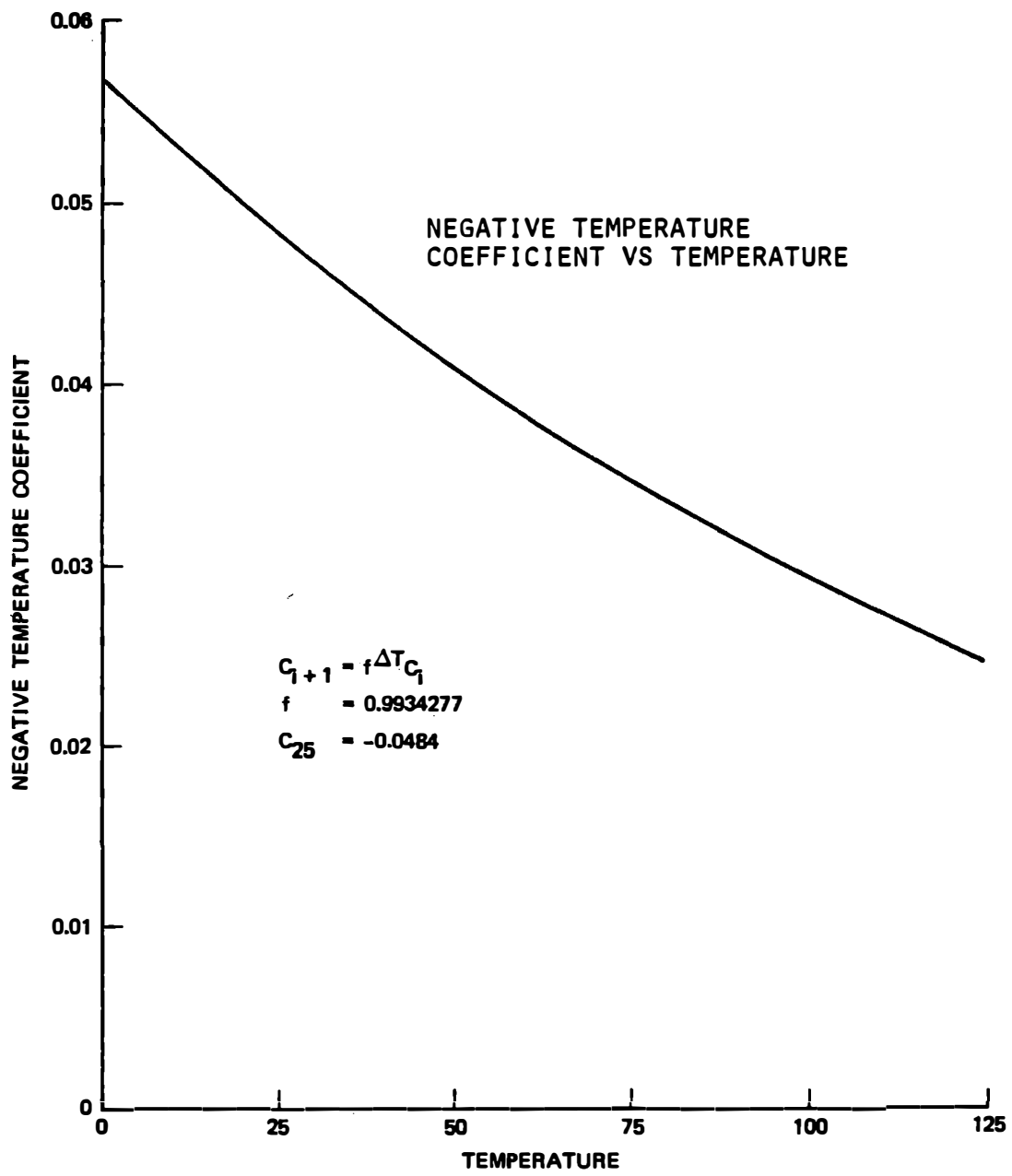
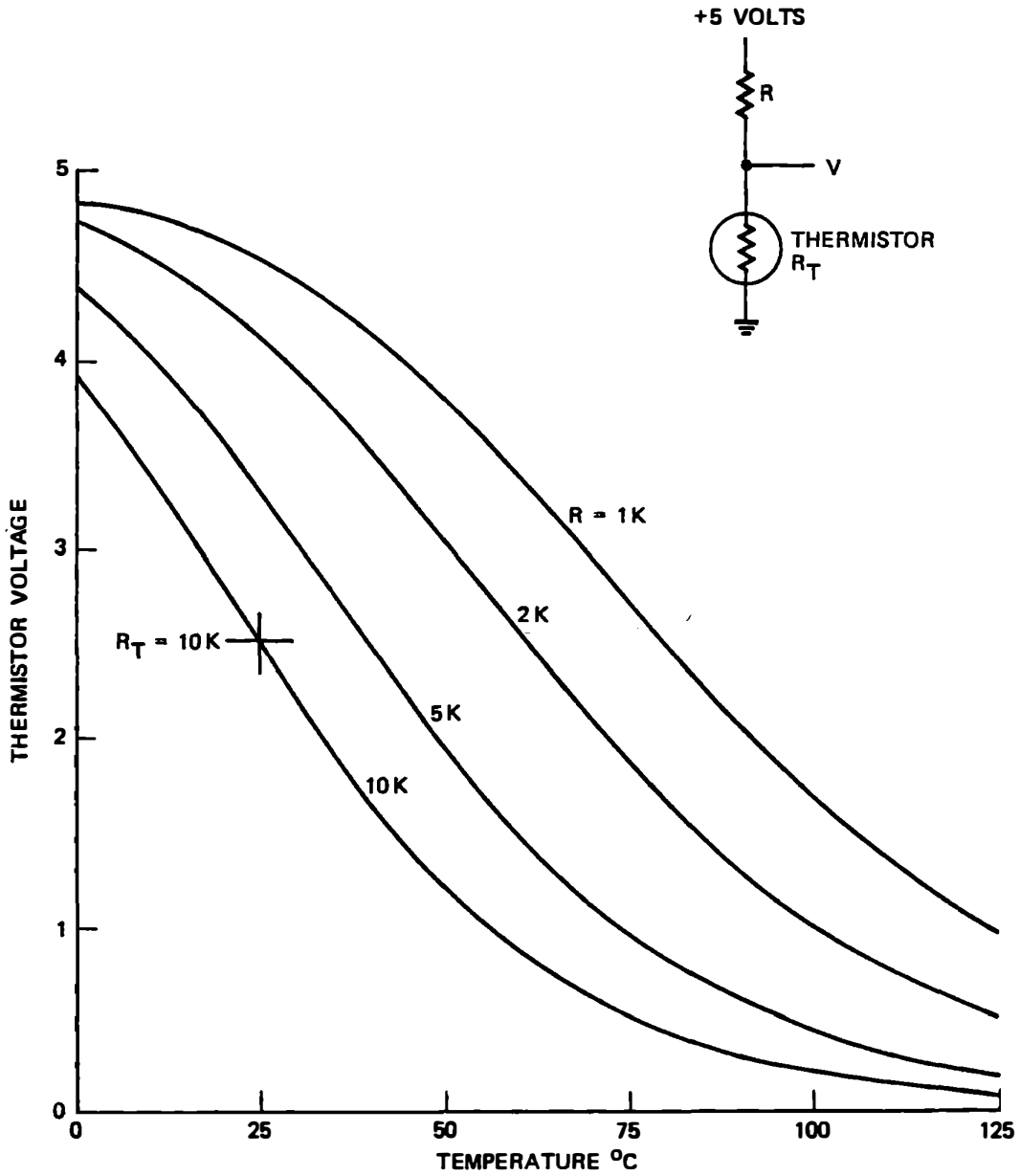


Figure 5-32b

This page intentionally left blank.



Thermistor Connection and Voltage Plot

Figure 5-33

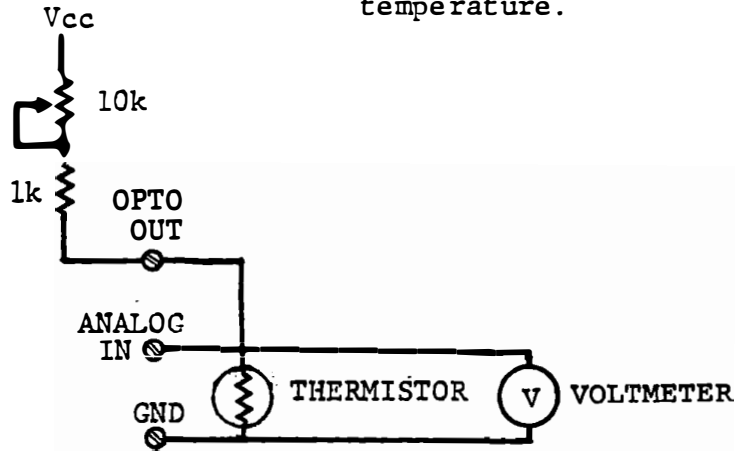
5.6.2 Thermistor Operation

When the thermistor is connected in a circuit such as shown in Figure 5-33, it gives a voltage which is close to linear over modest temperature ranges. The data can be linearized, and scaled from voltage to temperature, by a table lookup with linear interpolation. For the experiments in this section we will use the 10K ohm resistance data, but for high temperatures you might choose a lower resistance.

Because of its non-linearity and because it is subject to the uncertainties of semiconductor manufacturing, a thermistor must be calibrated. To do this adequately requires a laboratory thermometer and thermistor. This can be done by heating the thermistor and thermometer in a water bath. If you do not have these facilities available, it is reasonably satisfactory to measure the resistance at a known room temperature and scale the manufacturer's data appropriately. The following procedure does the necessary scaling by a pot adjustment, assuming that you will use the fitted curve data.

ANALOG TO DIGITAL INPUT

Adjust OPTO SENSE pot to obtain expected voltage at room temperature.



Temperature		Expected Voltage	A/D Input with 2:1 Attenuation
°F	°C		
65	18.33	2.910	92
66	18.89	2.876	90
67	19.44	2.842	8E
68	20.00	2.808	8C
69	20.56	2.773	8B
70	21.11	2.739	89
71	21.67	2.705	87
72	22.22	2.671	86
73	22.78	2.637	84
74	23.33	2.603	82
75	23.89	2.568	80
76	24.44	2.534	7E
77	25.00	2.500	7D

Expected Voltage at Room Temperature

Figure 5-34

5.6.3 Thermistor Input adjustment

Connect the thermistor as shown in Figure 5-34. Find the room temperature. (A household thermometer is sufficiently accurate for this.) Find the expected voltage from:

$$V_t = 2.50 + 0.0615 (25-T) \text{ (Celsius)}$$

or
$$V_t = 2.50 + 0.0342 (77-T) \text{ (Fahrenheit)}$$

or use the table of Figure 5-34. Note that for any temperature below 25° C the expected voltage is beyond the 2.55 volt range of the D/A converter. We will adjust the ANALOG IN pot to divide the voltage by 2. With the OPTO SENSE pot fully to the right for maximum resistance, measure the actual input voltage. Using a digital voltmeter program such as Figure 5-28 or 5-30, display the A/D input. Adjust ANALOG IN to make the A/D value half of the actual input. Now adjust the OPTO SENSE to obtain the expected voltage for the actual temperature. This procedure sets the pot to match the thermistor resistance at 25 degrees C, thereby removing the principal uncontrolled variable of the thermistor. You can now heat or cool the thermistor and observe the voltage changing. If you have a thermometer, you can calibrate the thermistor, taking a series of voltage and temperature measurements.

ANALOG TO DIGITAL INPUT

5.6.4 Table Lookup and Interpolation

To convert a measured voltage to a temperature requires a table lookup and possibly some form of interpolation. Four approaches are available:

5.6.4.1 Method A

Store a complete table of temperature versus voltage. This is not unreasonable, since the 8 bit A/D input only requires 256 table entries. The temperature can be stored in binary and converted to decimal for display. or with two byte entries, it can be stored in decimal. This approach minimizes program complexity, is very fast, but is extravagant of memory.

5.6.4.2 Method B

Store a partial table, listing voltage and temperature at appropriate intervals. Find two (adjacent) points in the table, above and below the measured voltage, and do a linear interpolation between them. This requires the least storage, but requires multiplication and division for the interpolation, and since that would probably be done in binary, it also needs binary to decimal conversion for the display.

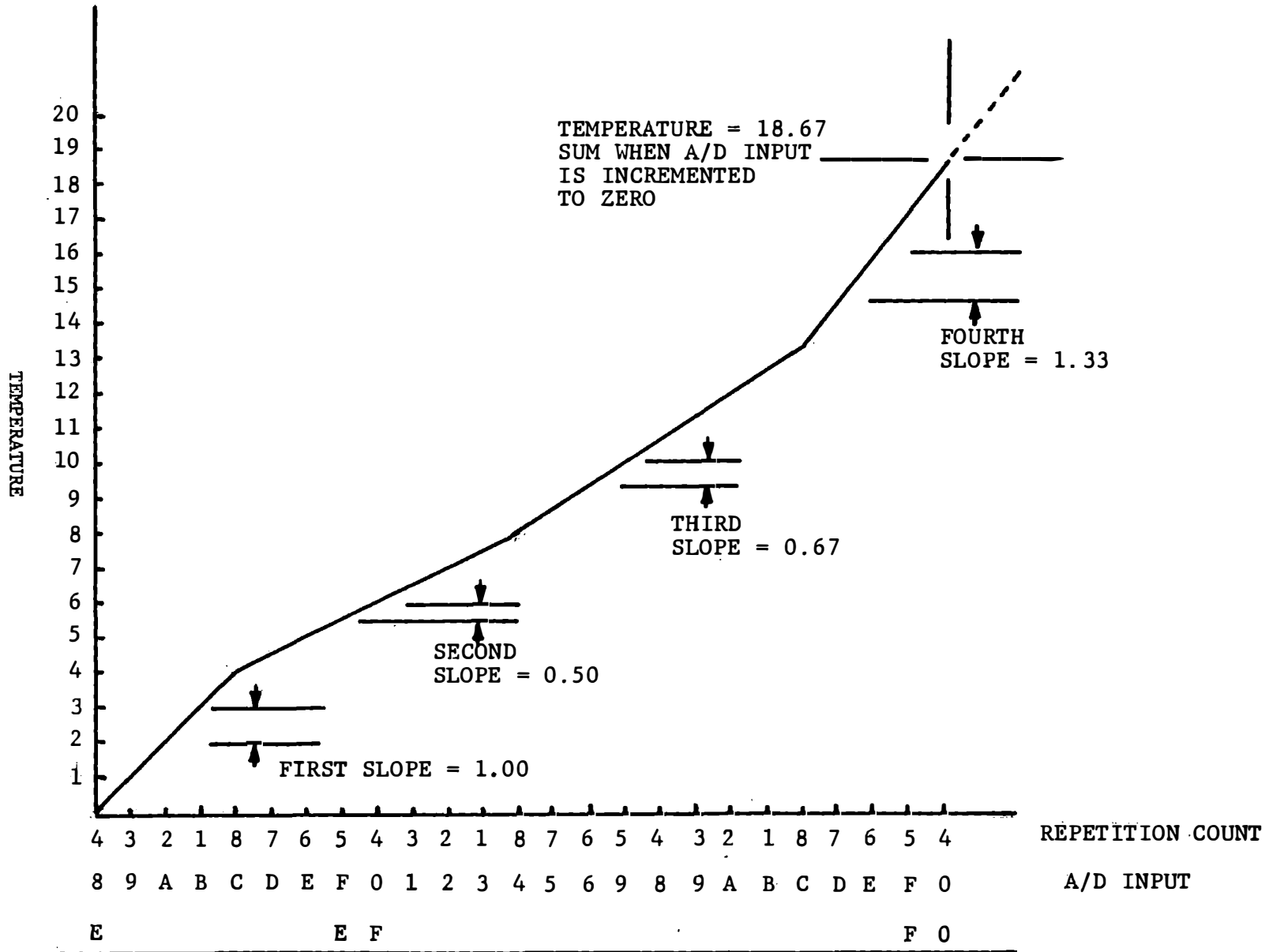
5.6.4.3 Method C

Store a table of voltage, temperature, and slope at appropriate intervals. Find the lowest table entry whose voltage is greater (hence temperature lower) than the measured value and multiply the

slope by the difference between tabulated voltage and measured voltage. Add this to the tabulated temperature. This avoids division, and can reasonably be done in decimal to avoid binary to decimal conversion. The multiplication can readily be done by successive addition, since the multiplier (the voltage difference) will always be a small number.

5.6.4.4 Method D

Store a table of slopes with ranges over which each slope applies, and perform a numerical integration from the start of the table to the measured voltage. The stored slopes need greater precision than with an interpolation, since errors accumulate, but the storage requirement is still less than any method except that of 5.6.4.2. This method is used in the following exercise.



Temperature Conversion by Integration

Figure 5-35

5.6.5 Voltage to Temperature Conversion

Develop a subroutine and a data table to convert voltage to temperature and display both the input and the result. The integration technique discussed in Section 5.6.4.4 is to be used.

5.6.5.1 Data Table

Each table entry includes a slope (in decimal) and a count. The slope is repeatedly summed into the temperature while its count is decremented and the A/D input is incremented. When the A/D input reaches zero, the integration is complete. If the count is decremented to zero before the A/D input reaches zero, the next table entry is accessed and the process continues. The process is portrayed in Figure 5-35, for a hypothetical A/D input of E8. The slopes and temperatures shown are illustrative and not at all realistic. Figure 5-36 lists actual data for an ideal thermistor with the previously specified characteristics. A subroutine for the conversion by integration is defined in Section 5.6.5.2, and shown in Figures 5-37 and 5-38.

ANALOG TO DIGITAL INPUT

TEST DATA		TABLE DATA	
A/D Input	Temperature	Repetitions (hex)	Slope (decimal)
C4	0.600	4	0.405
C0	2.219	10	0.373
B0	8.188	10	0.341
A0	13.644	20	0.324
80	24.012	10	0.335
70	29.372	10	0.355
60	35.052	10	0.394
50	41.356	10	0.458
40	48.684	10	0.532
38	52.940	8	0.606
30	57.788	4	0.681
2C	60.512	4	0.743
28	63.484	4	0.820
24	66.764	4	0.918
20	70.436	4	1.046
1C	74.620	4	1.214
18	79.476	4	1.452
14	85.284	4	1.803
10	92.496	2	2.190
0E	96.876	2	2.524
0C	101.924	2	3.089
0A	108.102	1	3.573
09	111.675	1	4.065
08	115.740	1	4.697
07	120.437	1	5.537
06	125.974	1	6.700

Thermistor Calibration Data

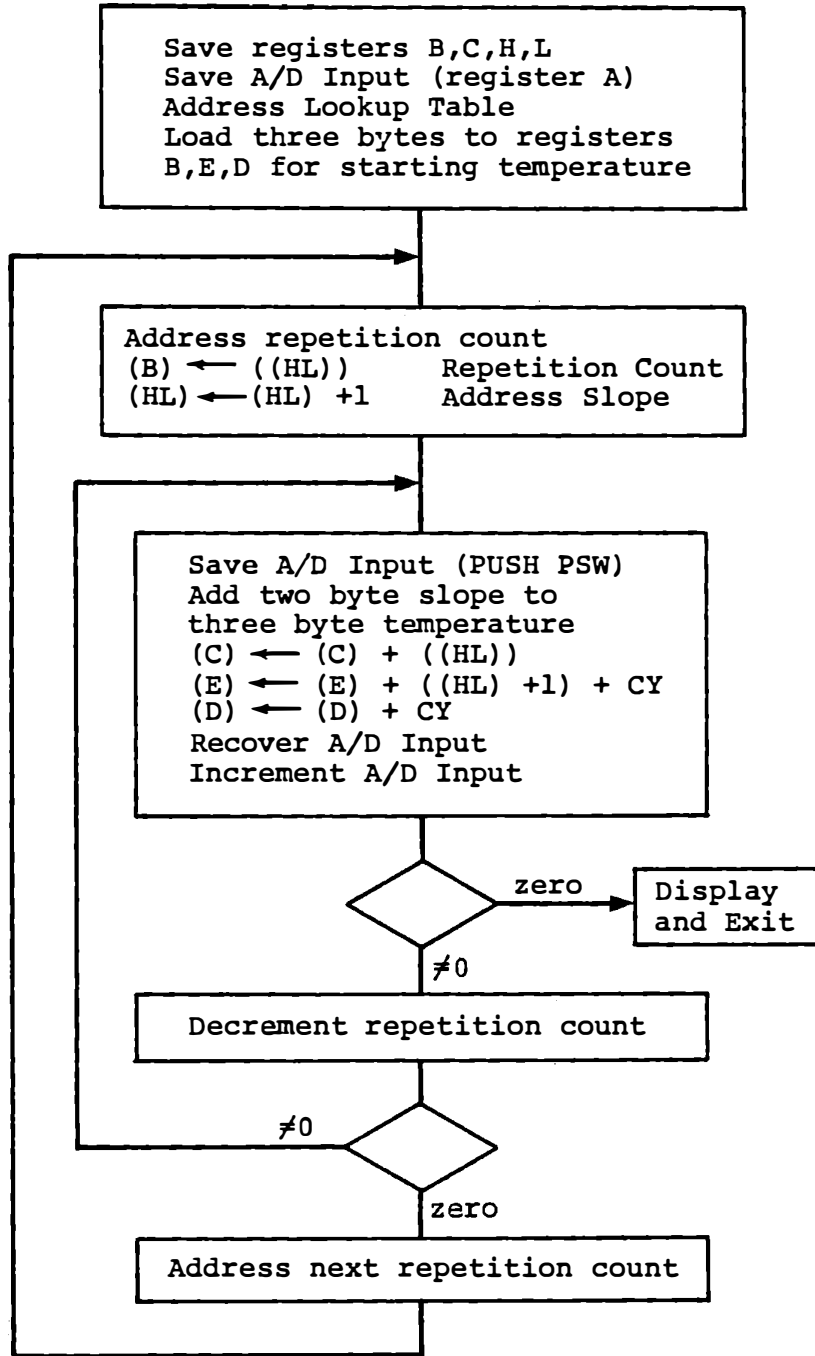
Figure 5-36

ANALOG TO DIGITAL INPUT

Our calibration data do not extend to 2.55 volts (FF). The first meaningful point occurs at 0.60° C at 3.92 volts (C4). The integration procedure to be used demands that data be provided for all possible values, so we will start the process with a linear integration from -23.701° C at a slope of 0.405 for 64 repetitions. This generates 0.600° C at 3.92 volts and gives the correct slope from there to 2.219° C at 3.84 volts. Results down to slightly negative temperatures will be approximately correct. The first table entry is the starting temperature (in hundreds complement form) and the next entry provides the 0.405° C per 20 mv slope with 40H repetitions.

8310	99	} -23.701
11	62	
12	97	
13	40	64 repetitions
14	05	} 0.405° C/20 mv
15	04	
16	10	16 repetitions
17	73	} 0.373° C/20 mv
18	03	
19	10	16 repetitions
1A	41	} 0.341° C/20 mv
1B	03	
etc.		

ANALOG TO DIGITAL INPUT



Temperature Lookup by Integration

Figure 5-37

5.6.5.2 Subroutine Temp

Enter with, (A) = measured voltage
 Return with (A) = measured voltage

Registers B,C,H,L preserved.

Display A/D input in hexadecimal, temperature in decimal either as three bytes (xxx.xxx) or as two bytes rounded (xxx.x).

Data table, located at 8310-836F,

8310-02	
8313	Repetition count for first slope
8314-15	Slope, decimal, as x.xxx
8316	Next repetition count
8317-18	Next slope
etcetera	

Note that the flow diagram does not detail the display function, which is left to the student.

5.6.5.3 Test for TEMP

A simple test program is given in Figure 5-38a. Key in a hex value representing a voltage and observe the result displayed by TEMP. Try values from Figure 5-36 to be sure that the corresponding temperature is displayed. It is suggested that subroutines FILTR and TEMP and the temperature table be saved on tape, because they will be used in later exercises.

TEST PROGRAM FOR TEMPERATURE LOOKUP

		A	D	D	R	CODE														
CODING SHEET	8	2	0			F	3			D	I									
			1			C	D			C	A	L	L		E	N	T	B	Y	
			2			3	6													
			3			0	3													
			4			F	B			E	I									
			5			7	D			M	O	V			A	.	L			
			6			C	D			C	A	L	L		T	E	M	P		
			7			B	0													
			8			8	2													
			9			C	3			J	M	P			8	2	0	0		
MICROCOMPUTER TRAINING SYSTEM	A		0			0	0													
	B					8	2													
	C																			
	D																			
	E																			
	F																			
	8		0																	
			1																	
			2																	
			3																	
			4																	
			5																	
			6																	
			7																	
			8																	
	INTEGRATED COMPUTER SYSTEMS	A																		
B																				
C																				
D																				
E																				
F																				
8			0																	
			1																	

Figure 5-38a

TEMPERATURE LOOKUP AND DISPLAY

A D D R		CODE							
CODING SHEET	8 2B	0 E5	PUSH	H					
		1 C5	PUSH	B					
		2 F5	PUSH	PSW					
		3 21	LXI	H, 8310					
		4 10							
		5 83							
		6 46	MOV	B, M				} Copy starting value into B, E, D	
		7 23	INX	A					
		8 5E	MOV	E, M					
		9 23	INX	H					
MICROCOMPUTER TRAINING SYSTEM	A	56	MOV	D, M					
	8 2B	B 23	INX	H				} Loop - address count (C) - repetitions	
		C 4E	MOV	C, M					
		D 23	INX	H					
		8 2B	E F5	PUSH	PSW				} Loop - save A/D
			F 7E	MOV	A, M				
		8 2C	0 80	ADD	B				} Add low byte of slope to low byte of temperature
			1 27	DAA					
			2 47	MOV	B, A				} Address high byte of slope and add to second byte of temperature
			3 23	INX	H				
		4 7E	MOV	A, M					
		5 8B	ADC	E					
		6 27	DAA					} Address low byte	
		7 5F	MOV	E, A					
		8 2B	DCX	H				} Add carry to high byte of temperature	
		9 3E	MVI	A, 00					
INTEGRATED COMPUTER SYSTEMS	A	00							
		B 8A	ADC	D					
		C 27	DAA						
		D 57	MOV	D, A					
		E F1	POP	PSW				} (A) ← A/D input Increment input	
		F 3C	INR	A					
		8	0						
			1						
			2						
			3						
		4							
		5							
		6							
		7							
		8							

Figure 5-38b

TEMPERATURE LOOKUP AND DISPLAY (continued)

A D D R		CODE								
CODING SHEET	8 2 D	0	CA	JZ				8 2 D B		Exit when A/D
		1	DB							input is incremented
		2	8 2							to zero
		3	0 D	DCR		C				Decrement repetitions
		4	C 2	JNZ			8 2 B E			Loop to add slope
		5	BE							until all repetitions
		6	8 2							of this slope done
		7	2 3	INX		H				(Address) high byte to
		8	C 3	JMP			8 2 B B			Loop to address
		9	BB							next slope
MICROCOMPUTER TRAINING SYSTEM	A	8 2								
	8 2 D	B	EB	XCHG						Exit and display
		C	F 1	POP		P SW				(ST) ← low byte of
		D	4 F	MOV		C, A				temperature and
		E	C 5	PUSH		B				original A/D input
		F	C D	CALL		DWORD				Display high bytes
		8 2 E	0	D 1						of temperature
INTEGRATED COMPUTER SYSTEMS		1	0 2							at left
		2	F 3	XTHL						(HL) ← low byte, voltage
		3	1 1	LXI		D, 8 3 F F				
		4	F F							
		5	8 3							
		6	C D	CALL		DWORD				Display voltage at
		7	D 4							right, and low byte
		8	0 2							of temperature
		9	1 B	DCX		D				(Address) whole degrees
		A	1 A	LDA		X D				and enter decimal
	B	F 6	ORI		8 0				point in display.	
	C	8 0								
	D	1 2	STAX		D					
	E	7 D	MOV		A, L				(A) ← Voltage	
	F	D 1	POP		D				(DE) ← Temperature	
	8	0	C 1	POP	B				Restore other	
		1	E 1	POP	H				registers	
		2	C 9	RET						
		3								
		4								
		5								
		6								
		7								
		8								

Figure 5-38c

TABLE OF REPETITIONS AND SLOPES

		A	D	D	R	CODE					
CODING SHEET	8	31	0	99						} Starting temperature -0.23, 401°C	
			1	62							
				2	97					} as 100's complement FF - CD	
				3	40						
				4	05					} 0.405°C/20mv	
				5	04						
				6	10					BF - BD	
				7	73					} 0.373°C/20mv	
				8	03						
				9	10					AF - AD	
MICROCOMPUTER TRAINING SYSTEM		A		41						} 0.341°C/20mv	
			B	03							
				C	20					9E - 80	
				D	24					} 0.324°C/20mv	
				E	03						
				F	10					7F - 70	
		8	32	0	35					0.335°C/20mv	
				1	03						
				2	10					6F - 60	
				3	55					0.355	
			4	03							
			5	10					5F - 50		
			6	94					0.394		
			7	03							
			8	10					4F - 40		
			9	58					0.458		
INTEGRATED COMPUTER SYSTEMS			A	04							
				B	08					3F - 38	
				C	32					0.532	
				D	05						
				E	08					37 - 30	
				F	06					0.1006	
		8	33	0	06						
				1							
				2							
				3							
			4								
			5								
			6								
			7								
			8								

Figure 5-38d

REPETITIONS AND SLOPES (CONTINUED)

	A	D	D	R	CODE																	
CODING SHEET	8				0																	
					1																	
		835			2	01														09		
					3	73															3.573	
					4	35																
					5	01															08	
					6	65															4.065	
					7	40																
					8	01															07	
					9	97															4.697	
MICROCOMPUTER TRAINING SYSTEM					A	46																
					B	01															08	
					C	37															5.537	
					D	55																
					E	01															07	
					F	00															6.700	
		8	36		0	67																
					1	FF															06	125.974°
					2	00															upper limit for	
					3	00															thermistors	
INTEGRATED COMPUTER SYSTEMS					4																	
					5																	
					6																	
					7																	
					8																	
					A																	
					B																	
					C																	
					D																	
					E																	
				F																		
	8			0																		
				1																		
				2																		
				3																		
				4																		
				5																		
				6																		
				7																		
				8																		

Figure 5-38f

5.6.6 Thermometer Program

EXERCISE

Develop a program to read the thermistor voltage and convert the measurement to decimal degrees by table lookup with interpolation.

In many systems it is more appropriate to take measurements at regular intervals than as rapidly as possible. This is particularly true with temperatures which typically change slowly and where rate of change may be of interest. In this program a timed interrupt will decrement a time counter and at one second intervals, it will increment a seconds counter. At each interrupt (20 milliseconds) it will reset the A/D converter and enable the A/D interrupt. Thus, a measurement of temperature will be made every 20 milliseconds, and a timer will be available to the main program.

RST 6 services the A/D interrupt. It will read the input from port 1B and call FILTR, the subroutine of Section 5.5.2, to obtain a filtered value for the input voltage. Since a measurement is wanted at 20 millisecond intervals, RST 6 service disables the A/D interrupt.

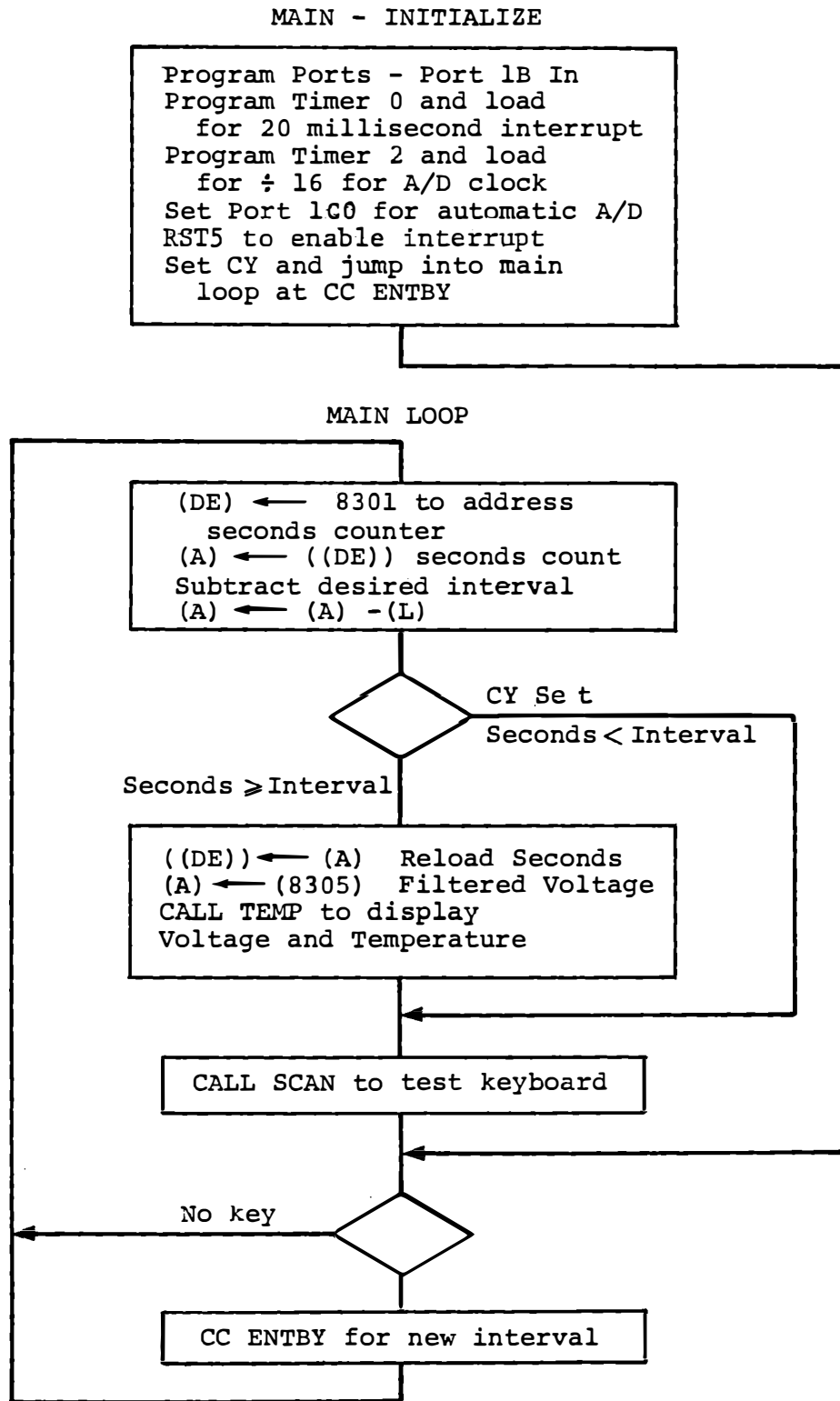
The main program loop compares the interval counter with an interval obtained by keyboard input. When the count has reached the desired interval, it restarts the counter, loads the current estimate of voltage and calls TEMP (the subroutine of Section 5.6.5.2) to display the temperature. It also tests the keyboard and calls ENTBY if a key is pressed to enter a new interval.

The subroutine developed in Section 5.3.5 will convert the measured voltage to temperature by table lookup and interpolation. Both temperature and voltage are displayed. The low byte of temperature is not significant, since the absolute accuracy is not better than half a degree. The display function should be modified to display only the two higher bytes. Rounding of the three byte result is easily achieved by making the initial value -23.651 instead of -23.701 (976.349 in hundreds complement).

Memory assignment for the program are:

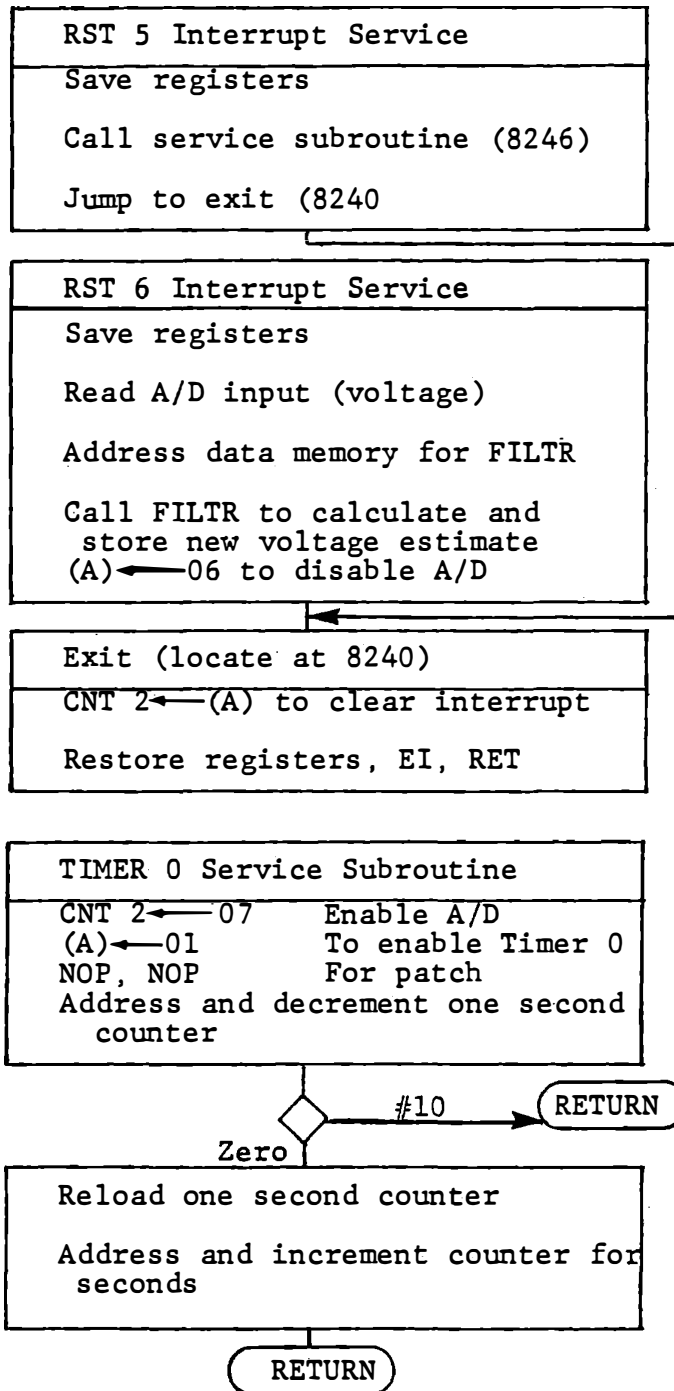
8200 - 8227	Main - Initialize
8228 - 8257	Interrupt Service
8258 - 826F	Main Loop
8270 - 82AF	Subroutine FILTR
82B0 - 82FF	Subroutine TEMP
8300	One second counter
8301	Seconds counter
8302	n for FILTR
8303-4	$2^n E_i$ for FILTR
8305	E_i
8310 - 836F	Table for TEMP

Note that 8300 - 8305 must be initialized at program loading, along with the table for TEMP, even though they contain variables. A solution is shown in Figures 5-39 and 5-40.



Thermometer - Main

Figure 5-39a



Thermometer - Interrupt Service

Figure 5-39b

THERMOMETER - INITIALIZE

A D D R		CODE						
CODING SHEET	8	20	0	3E	MVI	A,	82	Program Ports Port B In for A/D
			1	82				
			2	D3	OUT	CNT1		
			3	07				
			4	3E	MVI	A,	92	
			5	92				
			6	D3	OUT	CNT2		
			7	0F				
MICROCOMPUTER TRAINING SYSTEM			8	3E	MVI	A,	24	Program Timer 0 High byte Mode 2 Binary
			9	24				
		A		D3	OUT	TIMCT		
		B		17				
		C		3E	MVI	A,	94	
		D		94				
		E		D3	OUT	TIMCT		
		F		17				
MICROCOMPUTER TRAINING SYSTEM	8	21	0	3E	MVI	A,	A0	Load Timer 0 for 20 millisecond interrupt
			1	A0				
			2	D3	OUT	TIM0		
			3	14				
			4	3E	MVI	A,	20	
			5	20				
			6	D3	OUT	TIM2		
			7	16				
INTEGRATED COMPUTER SYSTEMS			8	3E	MVI	A,	01	Set Port 1CD high for automatic A/D conversion
			9	01				
		A		D3	OUT	CNT1		
		B		07				
		C		EF	RST	5		
		D		37	STC			
		E		C3	JMP	826A		
		F		6A				
INTEGRATED COMPUTER SYSTEMS	8	22	0	82				Jump to CC ENTBY in main loop to accept time interval
			1					
			2					
			3					
			4					
			5					
			6					
			7					
		8						

Figure 5-40a

THERMOMETER - INTERRUPT SERVICE

		A	D	D	R	CODE													
CODING SHEET	8	0																	
		1																	
		2																	
		3																	
		4																	
		5																	
		6																	
		7																	
MICROCOMPUTER TRAINING SYSTEM	822	8	F5			PUSH		PSW										RST5 - timer 0	
		9	E5			PUSH		H											
		A	CD			CALL		8246										Service timer 0	
		B	46																
		C	82																
		D	C3			JMP		EXIT											Exit
		E	40																
		F	82																
MICROCOMPUTER TRAINING SYSTEM	823	0	F5			PUSH		PSW										RST6 - A/D	
		1	E5			PUSH		H											
		2	DB			IN		PORT1B											Read voltage
		3	05																
		4	21			LXI		H, 8302											Data address
		5	02																for FILTR
		6	83																
		7	CD			CALL		FILTR											Calculate and store new voltage estimate
INTEGRATED COMPUTER SYSTEMS		8	70																
		9	82																
		A	3E			MVI		A, 06											To disable A/D interrupt
		B	06																
		C	C3			JMP		EXIT											
		D	40																
		E	82																
		F	00																
INTEGRATED COMPUTER SYSTEMS	8	0																	
		1																	
		2																	
		3																	
		4																	
		5																	
		6																	
		7																	
	8																	Figure 5-40b	

INTERRUPT SERVICE - EXIT, TIMER 0

CODING SHEET

MICROCOMPUTER TRAINING SYSTEM

INTEGRATED COMPUTER SYSTEMS

A	D	D	R	CODE						
8	24	0		D3		OUT		CNT	2	EXIT
		1		0F						
		2		E1		POP		H		
		3		F1		POP		PSW		
		4		FB		EI				
		5		C9		RET				
	824	6		3E		MVI		A,	07	TIMER 0 SERVICE
		7		07						
		8		D3		OUT		CNT	2	Reset and enable
		9		0F						A/D comparator
		A		3E		MVI		A,	01	To reenable
		B		01						Timer 0 interrupt
		C		00		NOP				Room for patch
		D		00		NOP				
		E		21		LXI		H,	8300	Address and
		F		00						decrement one
8	25	0		83						second counter
		1		35		DCR		M		
		2		C0		RNZ				
		3		36		MVI		M,	32	Reload one second
		4		32						
		5		23		INX		H		Address and
		6		34		INR		M		increment seconds
		7		C9		RET				
		8								
		9								
		A								
		B								
		C								
		D								
		E								
		F								
8		0								
		1								
		2								
		3								
		4								
		5								
		6								
		7								
		8								

Figure 5-40e

THERMOMETER - MAIN LOOP

	A	D	D	R	CODE													
CODING SHEET	8	0																
		1																
		2																
		3																
		4																
		5																
		6																
		7																
MICROCOMPUTER TRAINING SYSTEM	825	8	11		LXI	D	8301										(Address) seconds counter	
		9	01															
		A	83															
		B	1A		LDA	X	D										(A) ← seconds count	
		C	95		SUB	L											Compare interval	
		D	DA		JC		8267										If count less than interval, go to test keyboard	
		E	67															
		F	82															
	826	0	12		STAX	D												Restart seconds count
		1	3A		LDA		8305											(A) ← filtered input voltage stored by FILTR
	2	05																
	3	83																
	4	CD		CALL	TEMP												Convert to temperature and display	
	5	B0																
	6	82																
826	7	CD		CALL	SCAN													Test keyboard
	8	57																
	9	02																
826	A	DC		CC	ENTBY													If key pressed Accept time interval (one byte, seconds) for temperature update
	B	36																
	C	03																
	D	CB		JMP		8258												
	E	58																
	F	82																
INTEGRATED COMPUTER SYSTEMS	8	0																
		1			REQUIRES	S	SUBROTINES											
		2			FILTR	AT	8270 - 82AF											
		3			TEMP	AT	82BD - 82FF											
		4			INITIAL	DATA	8300 - 8305											
		5			TEMP	TABLE	8310 - 836F											
		6																
		7																
	8																Figure 5-40d	

SUBROUTINE FILTR

	A	D	D	R	CODE									
CODING SHEET	8	27	0		D5	PUSH	D						Save (DE)	
			1		C5	PUSH	B						Save (BC)	
			2		46	MOV	B	M					(B) ← n	
			3		48	MOV	C	B					(C) ← n	
			4		23	INX	H						Address 2 ⁿ E _{i-1}	
			5		E5	PUSH	H						Save address	
			6		5E	MOV	E	M					} (DE) ← 2 ⁿ E _{i-1}	
			7		23	INX	H							
			8		56	MOV	D	M					} (HL) ← 2 ⁿ E _{i-1}	
			9		6B	MOV	L	E						
MICROCOMPUTER TRAINING SYSTEM	A	62			MOV	H	D							
	827	B	29		DAD	H							} (HL) ← 2 ⁿ E _{i-1}	
		C	0D		DCR	C								
		D	C2		JNZ			827B						
		E	7B											
		F	82											
	8	28	0		4F	MOV	C	A					(C) ← V _i	
			1		7D	MOV	A	L					} (DE) ← (HL) - (DE)	
			2		93	SUB	E							
			3		5F	MOV	E	A					} = 2 ⁿ (2 ⁿ -1)E _{i-1}	
		4		7C	MOV	A	H							
		5		9A	SBB	D								
		6		57	MOV	D	A							
		7		69	MOV	L	C							
		8		26	MVI	H	00					} (HL) ← V _i		
		9		00										
INTEGRATED COMPUTER SYSTEMS	A	CD			CALL								Divide by 2 ⁿ	
	B	AD											(DE) ← (2 ⁿ -1)E _{i-1}	
	C	82												
	D	EB			XCHG									
	E	19			DAD	D							} (DE) ← V _i + (2 ⁿ -1)E _{i-1}	
	F	EB			XCHG									= 2 ⁿ E _i
	8	0											CONTINUED AT 8290	
		1												
		2				ENTER	(A)							= V _i (new value)
		3				((HL))								= n (filter constant)
	4				((HL) + 1)								} 2 ⁿ E _{i-1}	
	5				((HL) + 2)									
	6				RETURN									
	7				((HL) + 3)								= (A) = (H) = E _i	
	8												(L) = V _i	

FILTR (CONTINUED) and SHFTN

		A	D	D	R	CODE													
CODING SHEET	8	29	0	E	3		X	T	H	L								(HL) ← Address 2 ⁿ E _i	
			1	7	3		M	O	V		M	,	E						
			2	2	3		I	N	X		H	,						} Store 2 ⁿ E _i	
			3	7	2		M	O	V		M	,	D						
			4	2	3		I	N	X		H	,						Address E _i	
			5	1	B		D	C	X		D								To round up only if
			6	C	D		C	A	L	L		S	H	F	T	N			fractional part > 1/2
			7	A	0														
			8	8	2														
			9	7	7		M	O	V		M	,	A						Store E _i
MICROCOMPUTER TRAINING SYSTEM	A	E	1			P	O	P		H	,							(L) ← V _i	
	B	6	7			M	O	V		H	,	A						(H) ← E _i	
	C	C	1			P	O	P		B	,							Restore BCDE	
	D	D	1			P	O	P		D									
	E	C	9			R	E	T										Exit	
	F	0	0			N	O	P											
	8	2A	0	4	P		M	O	V		C	,	B					SHFTN (C) ← n	
			1	A	F		X	R	A		A	,						Loop - clear carry	
			2	7	A		M	O	V		A	,	D						
			3	1	F		R	A	R										Shift (DE) right
		4	5	7		M	O	V		D	,	A						to divide by 2	
		5	7	B		M	O	V		A	,	E							
		6	1	F		R	A	R											
		7	5	F		M	O	V		E	,	A							
		8	0	D		D	C	R		C	,							Loop n times	
		9	C	2		J	N	Z		8	2	A	1					to divide by 2 ⁿ	
INTEGRATED COMPUTER SYSTEMS	A	A	1																
	B	8	2																
	C	D	0			R	N	C										Exit if LSB = 0	
	D	1	3			I	N	X		D								Else round up	
	E	7	B			M	O	V		A	,	E						(A) ← less	
	F	C	9			R	E	T										significant byte	
	8		0																
			1																
			2																
			3																
		4																	
		5																	
		6																	
		7																	
		8																	

Figure 5-40f

TEMPERATURE LOOKUP AND DISPLAY

	A	D	D	R	CODE							
CODING SHEET	8	2B	0		E5		PUSH	H				
			1		C5		PUSH	H	B			
			2		F5		PUSH	H	PSW			
			3		21		LXI	H		8310		
			4		10							
			5		P3							
			6		46		MOV	B		M		
			7		23		INX	H				
			8		5E		MOV	E		M		} Copy starting values into B, E, D
			9		23		INX	H				
		A		56		MOV	D		M			
MICROCOMPUTER TRAINING SYSTEM		82B	B		23		INX	H				} Loop - address count (C) ← repetitions
			C		4E		MOV	C		M		
			D		23		INX	H				} Address slope Loop - save A/D
		82B	E		F5		PUSH	H		PSW		
			F		7E		MOV	A		M		} Add low byte of slope to low byte of temperature
	8	2C	0		80		ADD	B				
			1		27		DAA					} Address high byte of slope and add to second byte of temperature
			2		47		MOV	B		A		
			3		23		INX	H				} Address low byte
			4		7E		MOV	A		M		
		5		8B		ADC	E				} Add carry to high byte of Temperature	
		6		27		DAA						
		7		5F		MOV	E		A		} (A) ← A/D input increment input	
		8		2B		DCX	H					
		9		3E		MVI	A		00			
INTEGRATED COMPUTER SYSTEMS			A		00							} Add carry to high byte of Temperature
			B		8A		ADC	D				
			C		27		DAA					} (A) ← A/D input increment input
			D		57		MOV	D		A		
			E		F1		POP	PSW				
			F		3C		INR	A				
	8		0									
			1									
			2									
			3									
		4										
		5										
		6										
		7										
		8										

Figure 5-40g

TEMPERATURE WITH TWO BYTE DISPLAY (continued)

A	D	D	R	CODE													
CODING SHEET	8	2D	0	C	A		J	Z			8	2D	B	Exit when A/D			
			1	D	B									input is incremented			
			2	8	2									to zero			
			3	0	D		D	C	R	C				Decrement repetitions			
			4	C	2		J	N	Z		8	2	B	E	Loop to add slope		
			5	B	E									until all repetitions			
			6	8	2									of this slope done			
			7	2	3		I	N	X	H				Address high byte			
			8	C	3		J	M	P		8	2	B	B	Loop to address		
			9	B	B										next slope		
MICROCOMPUTER TRAINING SYSTEM	A	8	2														
		8	2D	B	E	B		X	C	H	G			Exit and display			
			C	C	D		C	A	L	L	D	W	O	R	D	Display temperature	
			D	D	I												
			E	0	2												
			F	F	I		P	O	P	P	S	W					
	INTEGRATED COMPUTER SYSTEMS	8	2E	0	C	D		C	A	L	L	D	B	Y	T	E	Display voltage
				1	9	5											(C) ← Voltage
				2	0	2											
				3	E	B		X	C	H	G						(DE) ← Temperature
			4	2	E		M	V	I	L	F	A				Address (whole	
			5	F	A											degrees)	
			6	7	E		M	O	V	A	M						
			7	F	6		O	R	I	8	0					Insert decimal point	
			8	8	0												
			9	7	7		M	O	V	M	A						
		A	7	9		M	O	V	A	C					(A) ← Voltage		
		B	C	I		P	O	P	B								
		C	E	I		P	O	P	H								
		D	C	9		R	E	T									
		E															
		F															
INTEGRATED COMPUTER SYSTEMS	8		0														
			1														
			2														
			3														
			4														
			5														
			6														
			7														
		8															

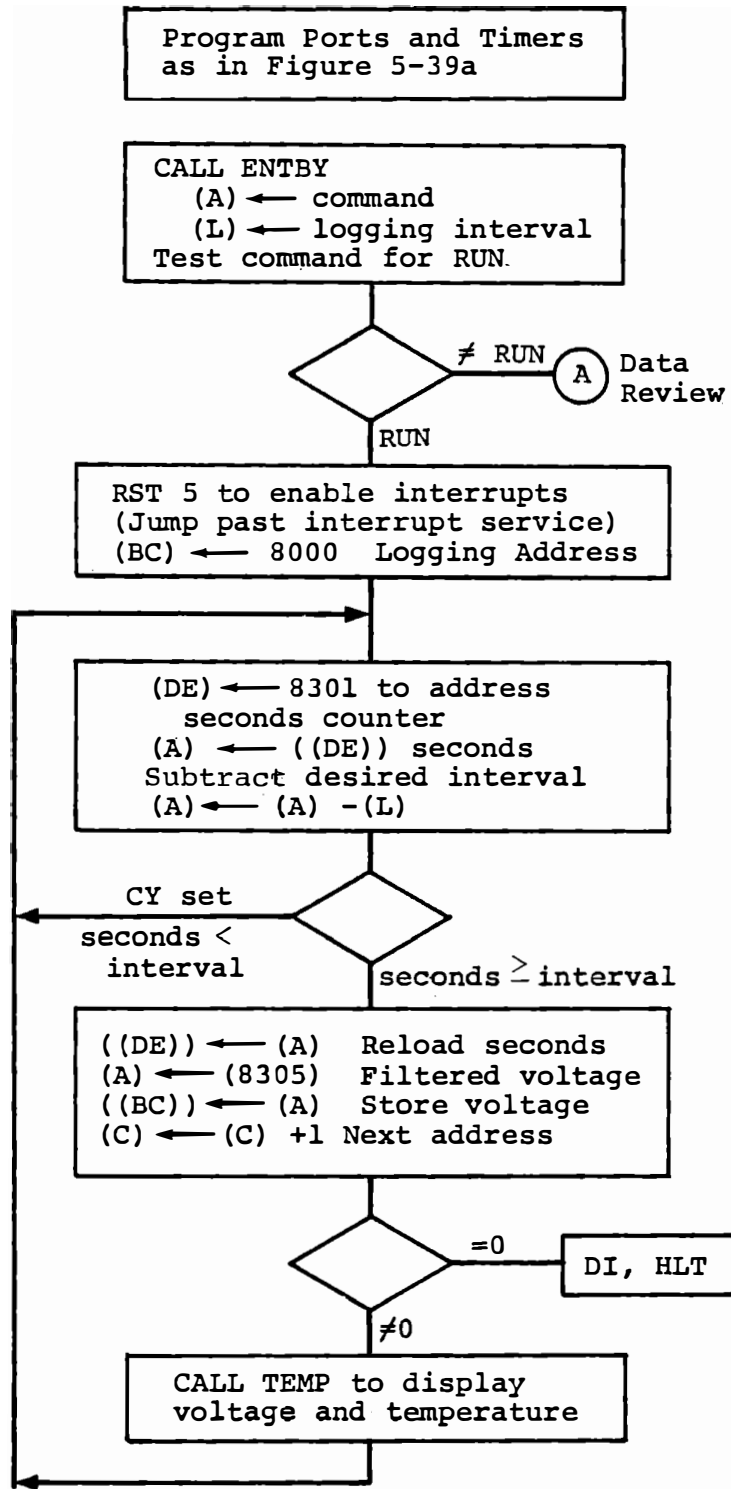
Figure 5-40h

5.6.7 Data Logging

OPTIONAL EXERCISE

Modify the thermometer program to make a data logger. This will record in memory a series of measurements for later review. It will also provide for subsequently reviewing the data. The design of subroutines FILTR and TEMP makes the revision very simple. Before CALL TEMP in the main loop, increment a data address and copy the filtered voltage to that location. Figure 5-41 shows the data logging program. To fit the program into the same space, we abandon the keyboard test while running, and simply call ENTBY once as part of initialization to obtain an interval. Register pair BC is available for the data logging address. Note the virtue of having subroutines save registers.

ANALOG TO DIGITAL INPUT



Logging Thermometer - Main

Figure 5-41

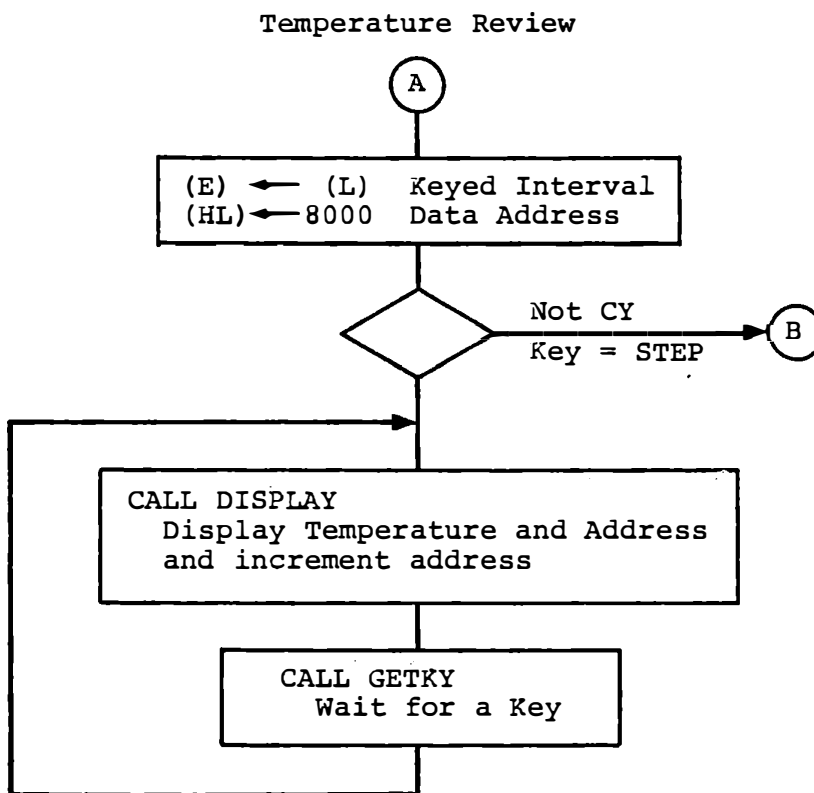
After CALL ENTBY in the initialization module, we will test the command for any of three keys:

- RUN Start data logging, at intervals given
 (in seconds)

- NEXT Review the stored data, showing the
 temperature at each point in succession
 when NEXT is pressed

- STEP Replay the stored data as an output to the
 D/A converter, using the time interval
 entered with the command.

NEXT and STEP jump to the modules shown in Figures 5-42 and 5-43. Replay is interesting because it allows several hours of data to be displayed on a scope or voltmeter in a much shorter time. For instance, if the interval entered for data logging was 3C (decimal 60), the recording interval is one minute, allowing four hours of data to be recorded. Then a replay of the logged data with an interval of 01 will play the data back in four minutes. Greater speedup can be obtained by altering the program constants used for loading timer 0 and the "one second" counter. The table of Figure 5-44 shows the constants needed for various recording and playback intervals, and Figure 5-45 presents a program solution.

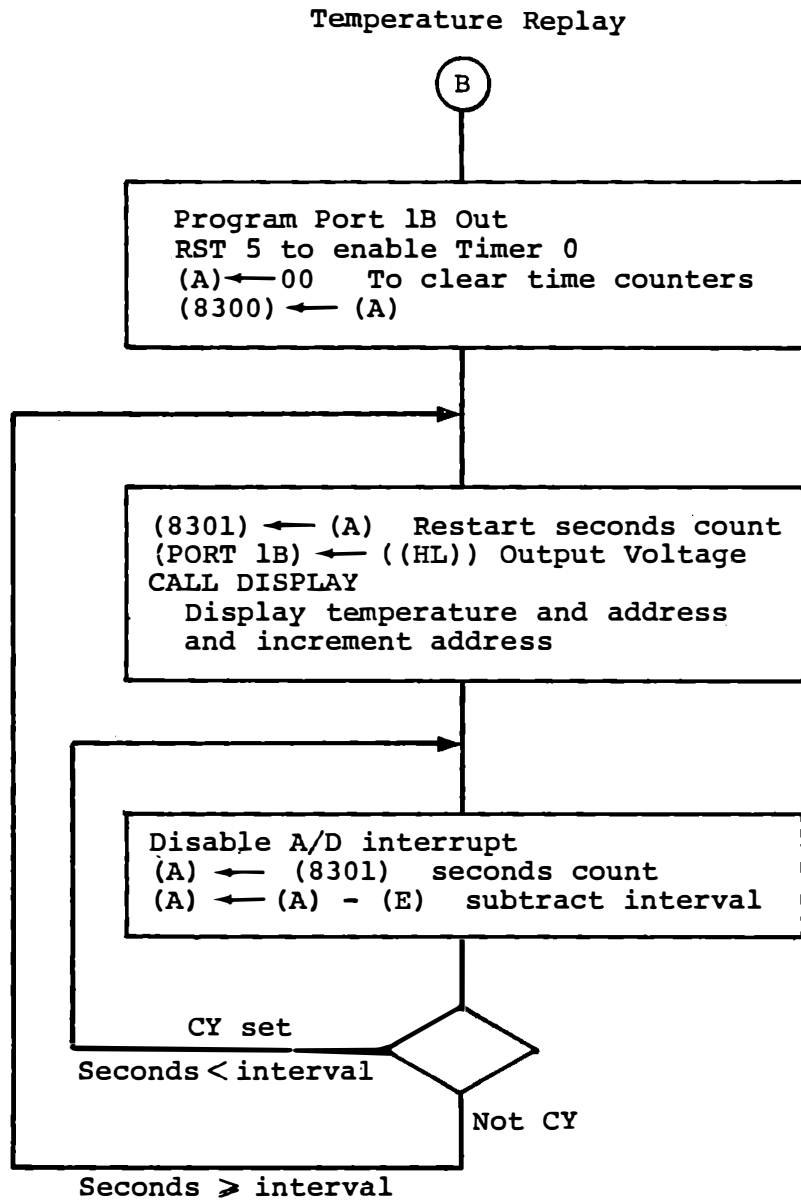


Display Subroutine

(ST) ← (DE)	Save (DE)
(A) ← ((HL))	Voltage
CALL TEMP	Display Temperature
(A) ← (L)	Display Address
CALL DBYTE	Next Address
(L) ← (L) +1	Restore (DE)
(DE) ← (ST)	
Return	

Logging Thermometer - Review Data

Figure 5-42



Logging Thermometer - Replay

Figure 5-43

Timer 0 Preload (high byte)	Interrupt Time	"One Second" Preload	Interval Count Time	Interval Entered	Recording Time	Max Run Time (approx)
08	1 ms	01	1 ms	01	1 ms	.25 sec
A0	20 ms	32	1 sec	01 0A 1E 3C 78	1 sec 10 sec 30 sec 1 min 2 min	4 min 40 min 12 hrs 4 hrs 8 hrs
F0	20 ms	C8	6 sec	0A 3C 64 96	1 min 6 min 10 min 15 min	4 hrs 24 hrs 40 hrs 64 hrs
F0	30 ms	F0	7.2 sec	FA	30 min	5 days

Logging Thermometer - Timing Constants

Figure 5-44

LOGGING THERMOMETER - INITIALIZE

		A	D	D	R	CODE			
CODING SHEET	8	20	0	3E		MVI	A,	82	Program 8255 #1
			1	82					Port B In
			2	D3		OUT	CNT	1	
			3	07					
			4	3E		MVI	A,	92	Program 8255 #2
			5	92					
			6	D3		OUT	CNT	2	
			7	0F					
MICROCOMPUTER TRAINING SYSTEM			8	3E		MVI	A,	24	Program Timer 0
			9	24					High byte
			A	D3		OUT	TIM	CT	Mode 2, Binary
			B	17					
			C	3E		MVI	A,	94	Program Timer 2
			D	94					Low byte
			E	D3		OUT	TIM	CT	Mode 2, Binary
			F	17					
MICROCOMPUTER TRAINING SYSTEM	8	21	0	3E		MVI	A,	A0	Load Timer 0
			1	A0					for 20 millisecond
			2	D3		OUT	TIM	0	interrupt
			3	14					
			4	3E		MVI	A,	20	Load Timer 2
			5	20					for 16 microsecond
			6	D3		OUT	TIM	2	clock to A/D
			7	16					
INTEGRATED COMPUTER SYSTEMS			8	3E		MVI	A,	01	Set Port 1C0
			9	01					In automatic A/D
			A	D3		OUT	CNT	1	0
			B	07					
			C	CD		CALL	ENT	BY	
			D	36					(L) ← Interval
			E	03					(A) ← Command
			F	FE		CPI	RUN		
INTEGRATED COMPUTER SYSTEMS	8	22	0	14					
			1	C2		JNZ	8100		Jump to review
			2	00					if not RUN
			3	81					
			4	EF		RST	5		
			5	C3		JMP	8258		
			6	58					
			7	82					
		8							

Figure 5-45a

LOGGING THERMOMETER - RUN LOOP

		A	D	D	R	CODE					
CODING SHEET	8	0									
		1									
		2									
		3									
		4									
		5									
		6									
		7									
MICROCOMPUTER TRAINING SYSTEM	825	8	01			LXI	B	8000			Initial data storage address
		9	00								
	A		80								
	825	B	11			LXI	D	8301			Address seconds
		C	01								
		D	83								
	825	E	1A			LDAX	D				(A) ← seconds
		F	95			SUB	L				Subtract interval
	826	0	DA			JC		825E			Loop until seconds ≥ interval
		1	5E								
		2	82								
		3	12			STAX	D				Restart seconds
	4	3A			LDA		8305			(A) ← filtered voltage	
	5	05									
	6	83									
	7	02			STAX	B				Store voltage	
	8	0C			INR	C				Next address	
	9	CD			CALL		TEMP			Display voltage and temperature	
INTEGRATED COMPUTER SYSTEMS	A		B0								
		B		82							
		C		C3		JMP		825B			Loop
		D		5B							
		E		82							
		F									
	3	0				REQUIRES		SUBROUTINES			
		1				FILTR		8270-82AF			
	2				TEMP		82B0-82FF				
	3				INITIAL		DATA 8300-8305				
	4				TEMP		TABLE 8310-				
	5										
	6										
	7										
	8										

Figure 5-45b

TEMPERATURE REVIEW WITH NXT KEY

		A	D	D	R	CODE														
CODING SHEET	8	10	0	EB		X	C	H	G									(E) ← voltage		
		1		21		L	X	I		H	,	8	0	0	0				(HL) ← address	
		2		00																
		3		80																
		4		DA		J	C			8	1	2	0							
		5		20																
		6		81																
		8	10	7	CD		C	A	L	L	8	1	1	0						Display temperature and address
		8		10																
		9		81																
M CROCOMPUTER TRAINING SYSTEM	A			CD		C	A	L	L	G	E	T	K	Y					wait for key	
	B			3D																
	C			02																
	D			C3		J	M	P		8	1	0	7						loop	
	E			07																
	F			81																
	8	11	0	DK		P	U	S	H	D									Display subroutine	
		1		7E		M	O	V	A	M									(A) ← voltage	
		2		CD		C	A	L	L	T	E	M	P						Display temperature	
		3		B0																
	4		82																	
	5		7D		M	O	V	A	L									Display address		
	6		CD		C	A	L	L	D	B	Y	T	E							
	7		95																	
	8		02																	
I NTEGRATED COMPUTER SYSTEMS	9		2C		I	N	R	L											Next address	
	A		D1		P	O	P	D												
	B		C9		R	E	T													
	C																			
	D																			
	E																			
	F																			
	8		0																	
		1																		
		2																		
	3																			
	4																			
	5																			
	6																			
	7																			
	8																			

Figure 5-45c

TEMPERATURE REPLAY

		A	D	D	R	CODE				
CODING SHEET	8	12	0	3E		MVI	A,	80		Program Port 1B Out
			1	80						for D/A conversion
			2	D3		OUT		CNT1		
			3	07						
			4	EF		RST	5			Enable Timer 0
			5	AF		XRA	A			interrupt
			6	32		STA		8300		Clear one second counter
			7	00						
MICROCOMPUTER TRAINING SYSTEM	8	12	9	32		STA		8301		Clear seconds counter
			A	01						
			B	83						
			C	7E		MOV	A,	M		(A) ← voltage
			D	D3		OUT		PORT 1B		Output to
			E	05						D/A converter
			F	CD		CALL		8110		Display temperature
		8	13	0	10					and address and
				1	81					increment address
		8	13	2	3E		MVI	A,	06	Disable A/D
				3	06					interrupt
				4	D3		OUT		CNT2	
			5	0F						
			6	3A		LDA		8301		
			7	01						
			8	83						
			9	93		SUB		E		
INTEGRATED COMPUTER SYSTEMS			A	DA		JC		8132		
			B	32						
			C	81						
			D	C3		JMP		8129		
			E	29						
			F	81						
		8		0						
				1						
			2							
			3							
			4							
			5							
			6							
			7							
			8							

Figure 5-45d

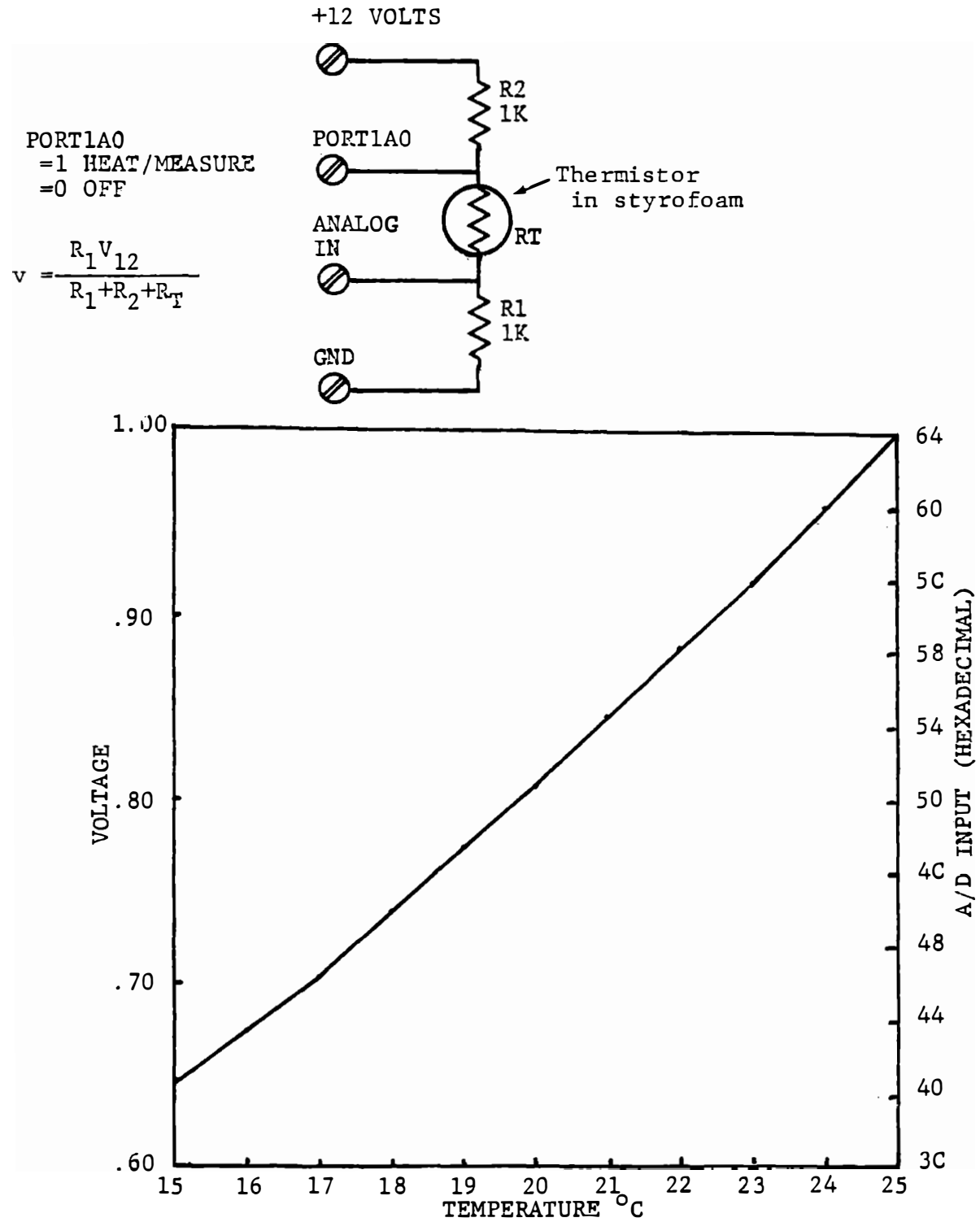
5.6.8 Thermistor Self Heating

When a temperature measuring device is carrying current, as the thermistor does, it generates heat internally. If the thermal resistance between the sensor and its environment is very low, as it will be in a liquid, this means that the sensor affects the experiment. If the thermal resistance is high, as in still air, the measured temperature will be higher than the real temperature.

This effect is usually negligible. In the circuit we have been using, the maximum self heating is about 0.6 milliwatt, occurring near 25^o C. Thus, the self heating would be less than 0.1 degree, not detectable with our A/D converter. For an experiment, the effect can be increased in two ways. Supply the thermistor from +12 volts with smaller series resistance to increase the heat generated, and bury the thermistor in a piece of styrofoam to decrease its dissipation. The connection of Figure 5-46 accomplishes this, and also allows switching the power to the thermistor on and off. We will show that the self heating error can be eliminated by applying power only during brief measurement intervals.

Since the circuit is changed, our previous calibration data are not valid for this experiment. Figure 5-46 shows a plot of voltage vs temperature for this connection, near room temperature. The input is in the neighborhood of one volt, so the ANALOG IN pot should be adjusted to no attenuation for maximum sensitivity.

ANALOG TO DIGITAL INPUT



Thermistor Connection and Calibration
for Self Heating Experiment

Figure 5-46

Note that with this connection, the slope is inverted from that of the normal connection, so the temperature conversion of the previous exercises must be modified. Interrupt service must be modified to set port 1A0 high at RST 5 to enable the measurement. To detect self heating, leave port 1A0 high, but to demonstrate its elimination, set port 1A0 low at RST 6.

In the interrupt service routine given in Figure 5-40b and 5-40c room was left for two patches to control power to the thermistor. In the subroutine that services Timer 0, we had:

```

MVI    A,01           To reenable Timer 0
NOP
NOP

```

Replace the NOP instructions with:

```

OUT    PORT1A        Turn on Thermistor Power

```

Now self heating should be measurable. To eliminate self heating we will turn thermistor power off after reading and processing the input. After the call to FILTR insert:

```

MVI    A,00           Turn off thermistor power
OUT    PORT1A
MVI    A,06           To disable A/D

```

ANALOG TO DIGITAL INPUT

Now power will be applied to the thermistor only while the A/D conversion is being performed. The MVI A,00 can be changed to MVI A,01 to again keep power on. The data log should demonstrate the difference. The revised program is shown in Figure 5-47. One instruction in TEMP is changed from INR A to DCR A, and a very short calibration table is entered.

THERMISTOR SELF HEATING - INTERRUPT SERVICE

		A	D	D	R	CODE						
CODING SHEET	8	0										
		1										
		2										
		3										
		4										
		5										
		6										
		7										
MICROCOMPUTER TRAINING SYSTEM	8 22	8	F5			PUSH	PSW				Timer 0	
		9	E5			PUSH	H					
		A	CD			CALL	8246				Call for service	
		B	46								of Timer 0	
		C	82									
		D	CB			JMP	8240				Jump to exit	
		E	40								with (A) = 01	
		F	82								to enable timer 0	
		8 23	0	F5			PUSH	PSW				A/D CONVERTER
			1	E5			PUSH	H				
INTEGRATED COMPUTER SYSTEMS		2	DB			IN	PORT 1B				Read A/D input	
		3	05									
		4	21			LXI	H, 8302				Data address	
		5	02								for FILTR	
		6	83									
		7	CD			CALL	FILTR				Calculate and store	
		8	70								new voltage estimate	
		9	82									
		A	3E	*		MVI	A, 00				To stop heating	
		B	00	*							01 to leave heat on	
	C	D3	*		OUT	PORT 1A						
	D	04	*									
	E	3E	*		MVI	A, 06				To disable and		
	F	06	*							reset A/D		
	8	0										
		1			*	CHANGED	FROM	5-40b				
		2										
		3										
		4										
		5										
		6										
		7										
		8									Figure 5-47a	

INTERRUPT SERVICE - EXIT, TIMER 0

A D D R		CODE									
CODING SHEET	8	24	0	D3		OUT	CNT2				EXIT
			1	0F							
			2	E1		POP	H				
			3	F1		POP	PSW				
			4	FB		EI					
			5	C9		RET					
MICROCOMPUTER TRAINING SYSTEM	8	24	6	3E		MVI	A, 07				TIMER 0 SERVICE
			7	07							
			8	D3		OUT	CNT2				Reset and enable
			9	0F							A/D converter
			A	3E		MVI	A, 01				Apply power to
			B	01							thermistors
			C	D3	*	OUT	PORT1A				
			D	04	*						
			E	21		LXI	H, 8300				Address and
			F	00							decrement
INTEGRATED COMPUTER SYSTEMS	8	25	0	83							one second counter
			1	35		DCR	M				Unless zero
			2	C8		RNZ					exit with (A) = 01
			3	36		MVI	M, 32				Reload one
			4	32							second counter
			5	23		INX	H				Address and
			6	34		INR	M				increment seconds
			7	C9		RET					Exit with (A) = 01
		8								to reenable Timer 0	
		A			*	CHANGED	FROM	5-40c			
		B									
		C									
		D									
		E									
		F									
		8	0								
			1								
			2								
			3								
			4								
			5								
			6								
			7								
			8								

Figure 5-47b

5.6.9 Other Temperature Logging Experiments

The thermistor is encapsulated so that it can be used in water, making several interesting experiments possible. Plot temperature versus time as you bring a pot of water from room temperature to boiling. Determine the temperature difference from a very gentle boil to a full boil. Determine the effect of a lid on the pot.

There is an old wives' tale that hot water will freeze more rapidly than cold water. Test it.

5.6.10 Abbreviated Temperature Lookup

For most measurement and control purposes, the extensive temperature table used up to here is not necessary, since it gives a precision greater than the absolute accuracy of the thermistor. In the program of Figure 5-48, a much shorter table is provided, fitting in the memory space 82EC to 82FF. Each slope is used for 32 additions, so the repetition count is not needed. This table gives the same results within $\pm 0.1^{\circ}$ C over the range of 10° C to 40° C and $\pm 1.0^{\circ}$ C from 0° C to 90° C.

A D D R CODE ABTMP - ABBREVIATED TEMPERATURE LOOKUP

A	D	D	R	CODE	ABTMP - ABBREVIATED TEMPERATURE LOOKUP					
CODING SHEET	8	2B	0	E5	PUSH	H				Save registers
			1	C5	PUSH	B				
			2	F5	PUSH	PSW				Save voltage
			3	21	LXI	H, 82	ED			Address initial
			4	ED	*					temperature in
			5	82	*					table
			6	46	MOV	B, M				
			7	23	INX	H				Copy starting value into B, E, D
			8	5E	MOV	E, M				
			9	23	INX	H				
MICROCOMPUTER TRAINING SYSTEM	A	56		MOV	D, M					
	82B	B	23	INX	H					Loop - address slope
		C	0E	*	MVI	C, 20				Set counter
		D	20	*						
	82B	E	F5		PUSH	PSW				Loop - Save A/D
		F	7E		MOV	A, M				
	8	2C	0	80	ADD	B				
			1	27	DA A					
			2	47	MOV	B, A				
			3	23	INX	H				Add slope to temperature
		4	7E	MOV	A, M					
		5	8B	ADC	E					
		6	27	DA A						
		7	5F	MOV	E, A					
		8	2B	DCX	H					
		9	3E	MVI	A, 00					
INTEGRATED COMPUTER SYSTEMS	A	00								
		B	8A		ADC	D				
		C	27		DA A					
		D	57		MOV	D, A				
		E	F1		POP	PSW				(A) ← A/D input
		F	3C		INR	A				Increment input
	3	0								
		1								
		2			*	CHANGED				FROM FIGURE 5-40a
		3								
	4									
	5									
	6									
	7									
	8									

Figure 5-48a

ABTMP (continued)

		A	D	D	R	CODE							
CODING SHEET	8 2D	0	CA		JZ			82DB				Exit when A/D	
		1	DB									input is incremented	
		2	82									to zero	
		3	OD		DCR		C					Decrement repetitions	
		4	C2		JNZ			82BE				Loop to add slope	
		5	BE									until 32 repetitions	
		6	82									of this slope done	
		7	23		INX		H					Address) and then to	
		8	C3		JMP			82BB				Loop to address	
		9	BB									next slope	
MICROCOMPUTER TRAINING SYSTEM	A		82										
	B		EB		XCHG							Exit and display	
	C		CD		CALL		DWORD					Display temperature	
	D		DI										
	E		O2										
	F		IE		MVI		E, FA					Address) whole degree	
	8 2E	0	FA										
		1	IA		LDA		X D						
		2	F6		ORI		80						Insert decimal point
		3	80										
	4	I2		STAX		D							
	5	F1		POP		PSW						(A) ← Voltage	
	6	CD		CALL		DBYTE						Display voltage	
	7	95											
	8	O2											
	9	EB		XCHG								(DE) ← Temperature	
INTEGRATED COMPUTER SYSTEMS	A		C1		POP		B						
	B		E1		POP		H						
	C		C9		RET								
	8 2E	D	99									Starting temperature -23.701°C as 100's complement	
		E	62										
		F	97										
		3	0										
		1											
		2											
		3											
	4												
	5												
	6												
	7												
	8												

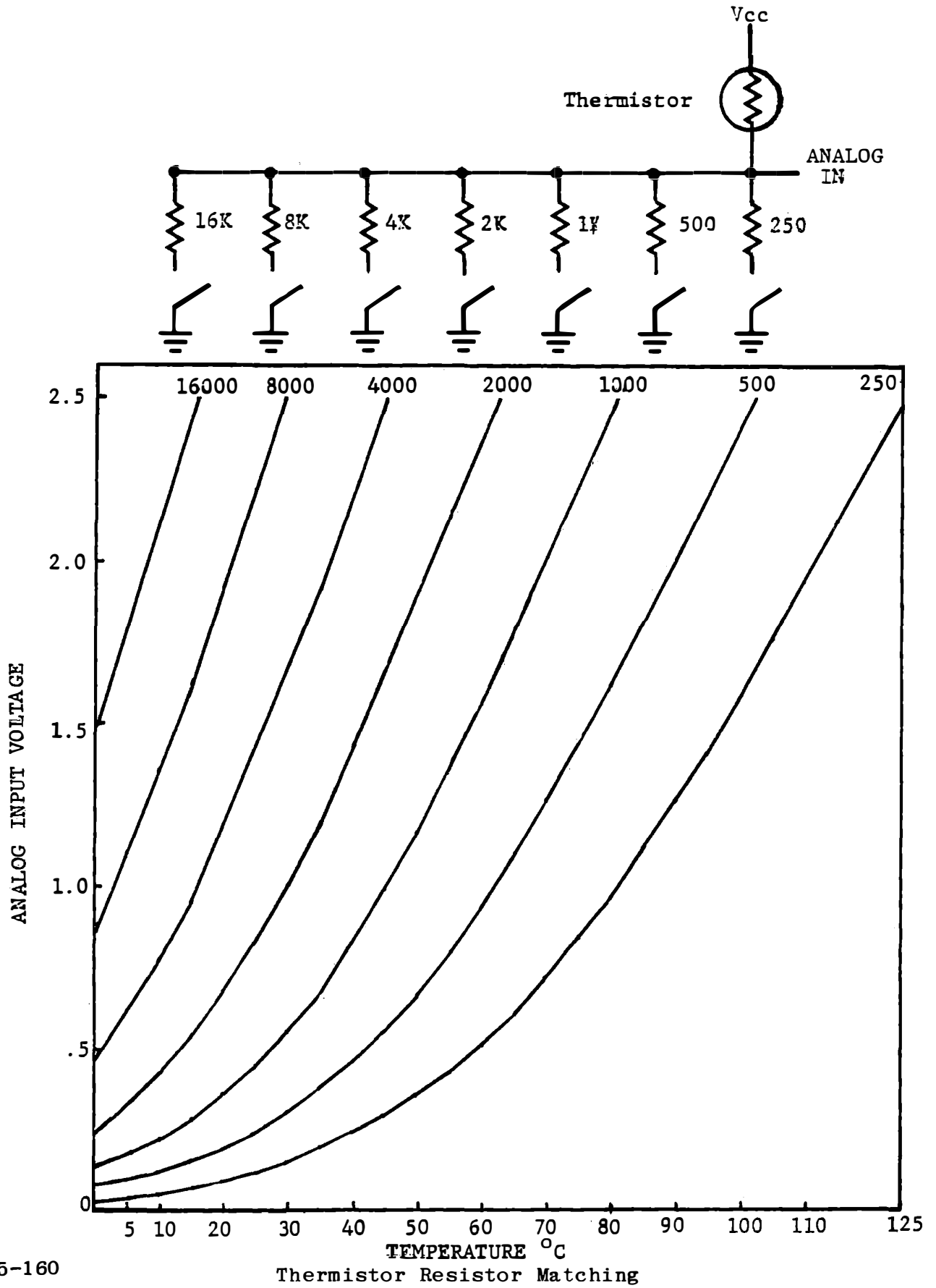
Figure 5-48b

ABTMP (continued)

	A	D	D	R	CODE																
CODING SHEET	8	2	F	0	05															FF-EO	
				1	04																0.405 °C/20mv
				2	05																DF-EO
				3	04																0.405 °C/20mv
				4	57																BF-AO
				5	03																0.357 °C/20mv
				6	24																9F-80
				7	03																0.324 °C/20mv
				8	42																7F-60
				9	03																0.342 °C/20mv
MICROCOMPUTER TRAINING SYSTEM	A				10															5F-40	
	B				04															0.410 °C/20mv	
	C				66															3F-20	
	D				06															0.666 °C/20mv	
	E				88															1F-00	
	F				13															1.388 °C/20mv	
	8				0																
					1																
				2																	
				3																	
				4																	
				5																	
				6																	
				7																	
				8																	
INTEGRATED COMPUTER SYSTEMS	A																				
	B																				
	C																				
	D																				
	E																				
	F																				
	8				0																
					1																
				2																	
				3																	
				4																	
				5																	
				6																	
				7																	
				8																	

Figure 5-48c

ANALOG TO DIGITAL INPUT



5-160

Figure 5-49

5.6.11 Thermistor Resistance Matching

OPTIONAL EXERCISE

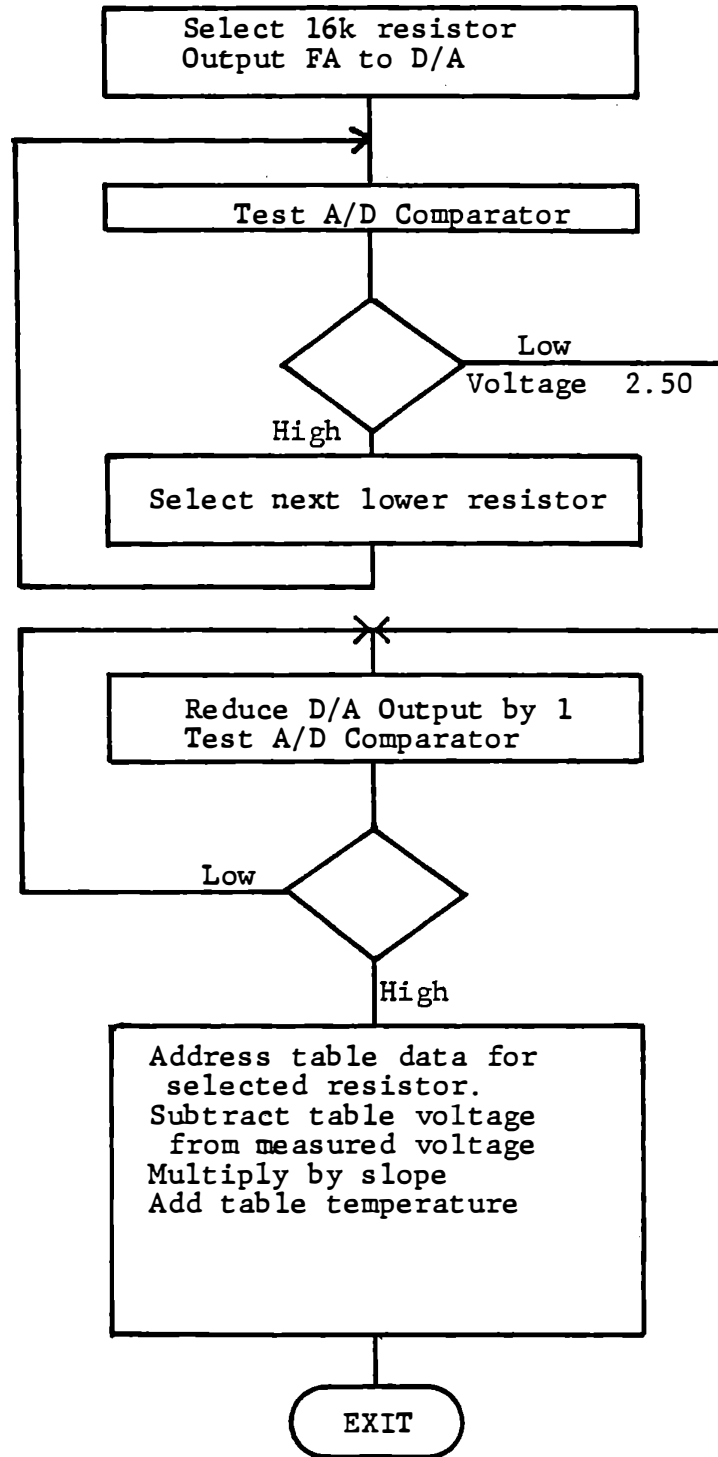
The chief difficulties in using the thermistor come from the non-linearity and high slope (degrees/volt) at higher temperatures. These problems can be avoided if the series resistance is well matched to the thermistor resistance at the temperature being measured. By switching discrete outputs the computer can accomplish this matching automatically. Figure 5-49 shows a circuit in which the thermistor is connected between Vcc and a resistor ladder which can be switched to provide different series resistance. Calibration curves for each resistance are shown. The advantage is that only a linear portion of each calibration curve is used, and a moderate slope (temperature/voltage) is retained at all temperatures.

Switching of the network can be done by the Port 1A outputs. These introduce an offset voltage because their outputs are not pulled perfectly to ground but only to about + 0.4 volts. To use this scheme effectively each curve should be calibrated independently, with two temperatures for each.

ANALOG TO DIGITAL INPUT

The processing algorithm finds the highest single resistor which brings the input voltage on-scale (less than 2.56 volts). Thus at 20^o C the 16K resistor produces an off-scale voltage, but the 8K resistor produces 1.92 volts. A table stores, for each resistor, the lowest temperature for which that resistor will be used, the corresponding voltage, and the slope. The measured temperature is the low temperature plus the slope times the voltage difference. Figure 5-50 shows the program flow.

The resistors required for this experiment are not supplied with the course.



Thermistor Resistor Matching Flow

Figure 5-50

This page intentionally left blank



INTEGRATED COMPUTER SYSTEMS

EDUCATION IS OUR BUSINESS™

NORTH AMERICAN HEADQUARTERS

Integrated Computer Systems
3304 Pico Boulevard
P.O. Box 5339
Santa Monica, California 90405 USA
Telephone: (213) 450-2060
TWX: 910-343-6965

NORTH AMERICA - EASTERN REGION

Integrated Computer Systems
300 North Washington Street
Suite 103
Alexandria, Virginia 22314 USA
Telephone: (703) 548-1333
TWX: 710-832-0045

EUROPEAN HEADQUARTERS

ICSP - U.K.
Pebblecoombe, Tadworth
Surrey KT20 7PA
England
Telephone: Leatherhead (03723) 79211
Telex: 915133

FRANCE

ICS France
90 Ave Albert 1er
92500 Rueil-Malmaison
France
Telephone: (01) 749 40 37
Telex: 204593

GERMANY

ICS D GmbH
Leonrodstrabe 54
8000 Munich 19
West Germany
Telephone: (089) 19 80 66
Telex: 5215508

SCANDINAVIA

ICSP Inc. - Scandinavia
Utbildningshuset AB
Box 1719
S-221 01 Lund, Sweden
Telephone: (046) 30 70 70
Telex: 33345