

MICROPROCESSOR REAL-TIME INTERFACING

Workbook/Text

Volume 2



Self-Study Course

Course 536A:

MICROPROCESSOR REAL-TIME INTERFACING

Workbook/Text

Volume 2

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.

© 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 O-89438-003-6
Volume I
ISBN O-89438-004-4
Volume II
ISBN O-89438-005-2

VOLUME I

	TABLE OF CONT	TENTS	i
	LIST OF ILLUS	STRATIONS	vii
1	HARDWARE INTI	ERFACING 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

	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	INTERVA	L 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

	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 T	O 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		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

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

VOLUME II

6	CLOSED I	OOP 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 CO	ONTROL	
	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

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

VOLUME I LIST OF ILLUSTRATIONS

FIGURE	TITLE	PAGE
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	<pre>Interrupt Service - RST 5, RST 6 (Flowchart)</pre>	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

FIGURE	TITLE	PAGE
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

FIGURE	TITLE	PAGE
4 7	Dulga Width Madulation Duranes	4-18
4-7	Pulse Width Modulation Program	
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 Regording 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	Keyborad 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

FIGURE	TITLE	PAGE
4-39	Successive Charge/Discharge Cycles	4-114
4-40	Key Selection of Waveform	4-117
4-41	EXPV	4-117
4-42	Subroutine BMULT	4-121
4-42	Test Program for EXPV	4-126
4-44	Function Generator	4-131
4-44		4-132
4-45	Multiplication - Subroutine BMULT	
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 OIDSP	5-46
5-13	OIDSP - 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

FIGURE	TITLE	PAGE
5-23	Successive Approximation Voltmeter	
3-23	(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

VOLUME_II

FIGURE	TITLE	PAGE
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

FIGURE	TITLE	PAGE
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

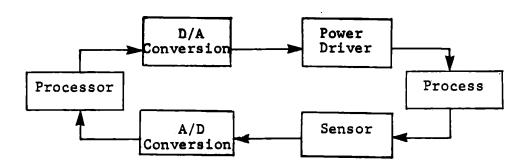
CHAPTER 6

CLOSED LOOP CONTROL

This page intentionally left blank.

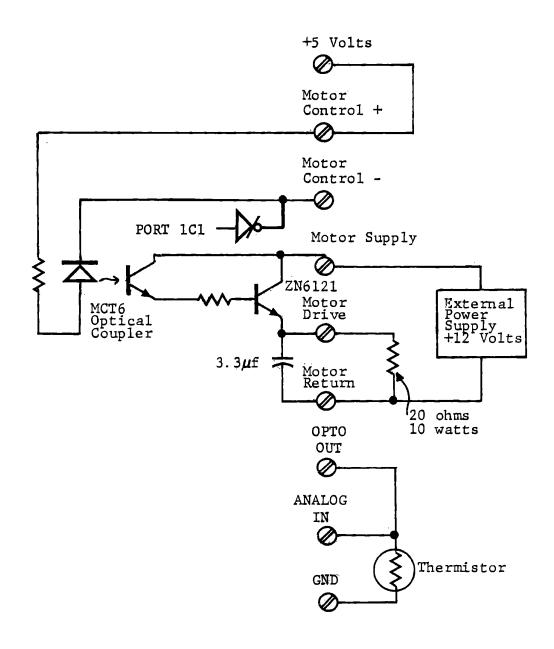
6. CLOSED LOOP CONTROL

As we have seen, a computer can generate control signals and receive sensed inputs. When the input is used to determine the control output, we have "closed the loop".



In this chapter we will deal with three closed loop problems: on-off (thermostat) control, proportional voltage or temperature control, and speed control.

Voltage and temperature control are essentially similar problems and will use the same forms of analog to digital input and digital to analog output. The automatic A/D input function of the Ferranti 425 will be used to sense the condition of the external "process", and output switching or pulse width modulation will be used for control. In speed control we will determine the speed by a frequency measurement, and we will try both PWM and the Ferranti D/A conversion for control.



Connections for Thermostat Exercise
Figure 6-1

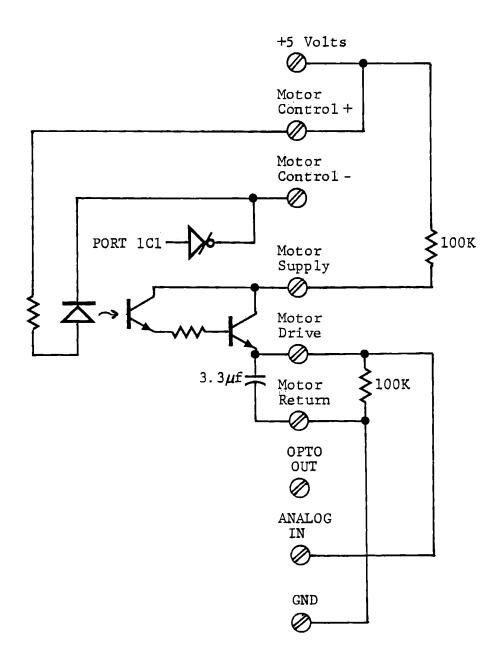
6.1 ON-OFF CONTROL

The simplest closed loop control system is exemplified by the familiar household thermostat. The temperature is measured, and if it is too low (i.e., below some preset value) the furnace is turned on. When the temperature reaches the desired value the furnace is turned off.

An on-off control system usually has a "dead band" in which the present condition of the output is not changed - if the furnace is on it stays on until an upper temperature limit is reached; if it is off it stays off until a low limit is reached.

Without the dead band the output will switch on and off too frequently, resulting in inefficient operation (and annoying noise in the household situation). In a simple thermostat, the dead band is provided by mechanical hysteresis in the bimetal switch. In more sophisticated systems the upper and lower limits can be programmed independently. It is also possible to let system time constants provide the dead band, as we shall see in the first exercise.

The interface board includes a Fairchild 2N6121 power transistor driven by a Monsanto MCT6 optical coupler. This circuit can be used directly to heat a power resistor, as shown in Figure 6-1. The thermistor must be coupled to the heater as closely as possible, since the heating available is not very great. One way of achieving close coupling is to wrap the resisitor and thermistor together with electrical tape.



Connections for On-Off Voltage Control Figure 6-2

Alternately, the power transistor can drive a relay controlling power to an AC heater, with the thermistor in water that is being heated. (Note: Exercise 6.1.1 should not be done with a relay, because the fast switching would damage the relay contacts).

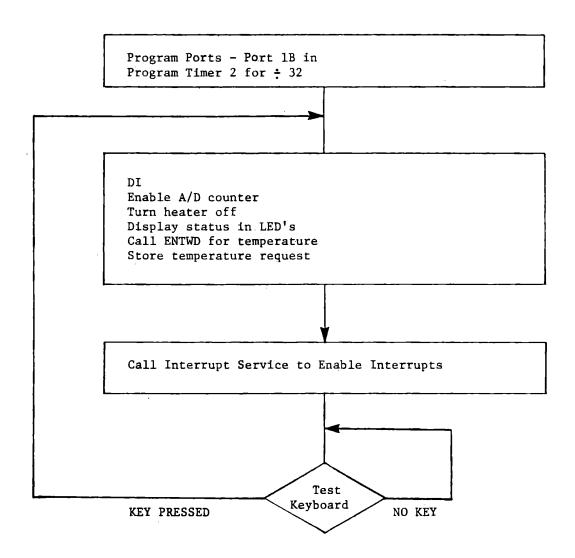
The experiments can be performed without heating anything, but simply simulating heat by the charge on a capacitor and measuring its voltage, as shown in Figure 6-2.

6.1.1 On-Off Control Without Deadband

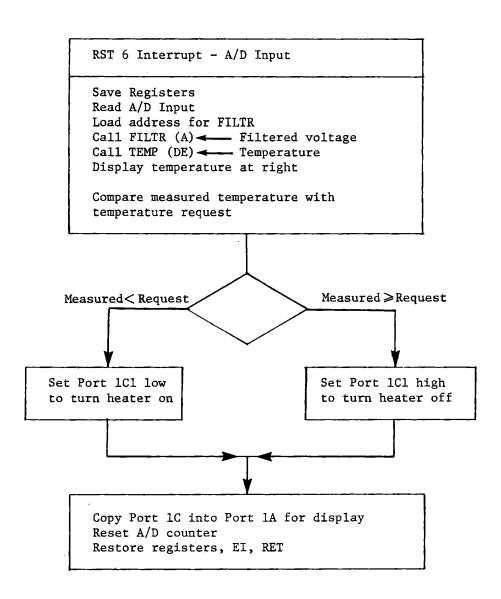
EXERCISE

The on-off control experiment will use subroutines TEMP and FILTR, Leveloped in Chapter 5. The program of Figure 6-3 accepts a temperature request (by keyboard entry) and attempts to heat the load to that temperature as measured by the thermistor. Whenever the temperature is less than that requested it turns the heater on by setting Port 1C1 low; when the temperature is greater it sets Port 1C1 high to turn the heater off. The temperature is calculated and displayed by subroutine TEMP. The measurement and control functions are contained entirely in the RST6 interrupt service, with the main program performing initialization and then calling ENTWD for a temperature request. The desired temperature must be entered as degrees and tenths, to permit comparison with the temperature returned by TEMP. For instance, enter 315 to request a temperature of 31.5° C.

Clearly this program does nothing very exciting. Its main purpose is to demonstrate the effect of having no deadband. Observe the high frequency at which it switches the power on and off when the desired temperature is reached. This is no problem to the power transistor, nor to an SCR, but would damage the contacts of a relay. Moreover, many heating devices are less efficient when being switched on and off repeatedly, and some may be damaged.



Initialization
On-Off Control - No Deadband
Figure 6-3a



Interrupt Service
On-Off Control - No Deadband
Figure 6-3b

THERMOSTAT WITHOUT DEADBAND

	A D D	R	cor	DE												-71
	8 20	0	3	E		M		I		A		8	2			Program Porto
		1	8	2	_											Post 1B Un
		2	\supset	3		0	U	T		C	N	\mathcal{T}	/			
_		3	0	7												
SHEET		4	3	E		M	V	\mathcal{I}_{-}		A		9	2			
		5	9	2		- 1)		_			
S		6	5	3		0	V	7		C	N	7	2			
CODING		7	0			\mathcal{O}	U	1			70	/	مد			
ŏ		8	3	7		Μ	V					a	1/			11. 2
		9	_	E		141	V	<u></u>		A	,	7	7			Program Jine 2
			7	₩			/.				_	1.4				Low legte
		<u> </u>	1	-		0	ν	_/_		1	1	M	C	\mathcal{L}		Thorse 2
		В	Z	7			,			_	_					Genary
_	<u></u>	С	3	E		M	V	I		1		2	0		_4	Load Times 2
ΓEΜ		D	2	0							Ĺ_					for ÷ 32
SYST		E	D	3		0	U	\mathcal{T}		T	I	M	2			J
		F	/	6												
TRAINING	82/	0	F	3			I									Reinterrupto while
Ā		1	3	E		M	V	I		A		0	3			in ENTWD
		2	0	3							/					PICO Sigh Joy A ID
E E E		3	5	3		0	U	T		P	0	R	τ	1	C	PICI Risk
P		4	0	6	-		_	_	_	_			Ė	Ť		to sound all
MICROCOMPUTER		5	5	3		0	1)	厂	_	P	0	R	7	1	A	De la Lette
õ	<u> </u>	6	0	1/		۲		_		Ė		-	_	 	 	Captay in El Vs
2		7	0	カ			A	/	L		E	N	T	14/	D	0+ + +
≥	-	8	4			\subseteq	//	-			_	10	 	VV	1	Enter temperature
		9	0	3			-	1	-		-	-	-	-		limit in degrela
			1			0	H	,	<u> </u>	-	8	3	0	_	├	and lentes
SI	 -	A	0	2		<u>ی</u>	/7	1	D	-	8	3	0	6	-	(e.g., 273 for 27.3°C)
SYSTEMS		В	0	6		ļ	-	ļ	_	-	ļ		ļ	 	_	Store temperature
ΥS		С	8	3		2	_	_	_	-	<u> </u>	_	_	_		limit
		D	1	7		\mathcal{R}	5	7	6	ļ.,	<u> </u>	<u> </u>	<u> </u>	_	_	Enable A/D interrupt
COMPUTER	821	E	(B		I	N	<u> </u>	L.	P	0	R	\mathcal{T}_{-}	0	A	Test Seyboard
MP		F	0	0		<u> </u>		_	_	L					ļ	(FF if no key)
Ö	8 22	0	3	C		I	N	R		A						
		1	C	A		2	7			8	2	/	E		_	Loopuntil
A		2	1	E								-				Ley pressed
GB		3	8	2												
INTEGRATED		4	0	3		5	M	P		8	2	1	0			Gato DI and
=	<u> </u>	5	7	0		Ť	Ϊ,	Ė				ŕ		 		
		6	8		<u> </u>	Т			-		T		1		1	and accept data
		7	۲	<u> </u>			\vdash	\vdash	-		\vdash		t	一	<u> </u>	una accepi aura
		8	╁	\vdash	 -	\vdash	-	\vdash	-	+-	+-	\vdash	+	+-	+	Figure 6-4a
			1	1	L	Щ			L	1			1	Щ		rigure 6-4a

	<u> </u>	со	DF	THE	RMO	ST	ΑT	- 3	RST	6	IN	TE	RRU	PT	
	8 23 0	F	5		P	U	5	14		P	2	W			
	1	E	5		P	U	2	<i>H</i>		Н		u			
ĺ	2	כל	3		P	U	S	\mathcal{H}		$\sqrt{2}$					
 	3	C	5	-	P	U	5	 H		\mathcal{B}					
EET	4	2	/		L	X	I		14		8	3	0	0	Data address
SH	5	0	0		-	_	_	-		,			_		for FILTR
N.	6	8	3												The state of the s
CODING	7	D	B		I	N			P	0	R	T	1	$\boldsymbol{\beta}$	Lead A/D
	8	0	5												
	9	C	D		C	A	L	1		F	I	L	7	R	(A) - filtered
,	A	7	0												voltage
į	В	8	2												
	С	C	\mathcal{N}		C	A	L	L		1	E	Μ	ρ		Calculate and
'STEM	. D	β	0												display temperature
SYST	E	8	2												(DE) - lemperature (HL) - requested
	F	2	A		L	H	L	\supset		g	3	0	6		(HL) - requested
TRAINING	8 24 0	0	6												temperature
ΑAI	1	8	3											ļ. 	/
	2	7.	0		M	0	V		A		L				Compare temperature
JTER	3	9	3		S	V	\mathcal{B}	·	E						to request
MICROCOMPUT	4	7	\mathcal{C}		M	0	V		A	اــرــا	#				Sets CY if
Ö	5	9	A		S	\mathcal{B}	\mathcal{B}		D						temperature high
SRO	6	3	E		M	V	I		17		0	/	L		temperature high (A) ← 03 if high
M	7	0	/_						<u> </u>	Ĺ		ļ			€ 62 if low
	8	1/	7		R	A	L	<u></u>						<u> </u>	
	. 9		3		0	U	T		C	N	I	1			Set Port /C/ low (power on) if temperature low Read and display power control
<u>S</u>	A	0	7			ļ.,					_				(power on) if
ΓEΝ	В	1	\mathcal{B}		\mathcal{I}	N			P	0	R	7	/	C	temperature low
SYSTEMS	С	0	6	-	_			ļ		_		_	<u></u>	_	
	D	1	3		0	V	7	-	P	0	R	1	/	A	Keal and display
- 15	E	0	4		L.,	_		<u> </u>	-		_	-			power control
MΡ	F	10	0		N	0	P								
20	8 0	+	_			_						-		-	
핕	1	1	_		-	_	-	-			-			-	
ìRA	2	+	-		<u> </u>	-	-		-	<u> </u>	_	_		ļ.	
INTEGRATED COMPUTER	3	+	-				-			-	-	\vdash	-	\vdash	· · · · · · · · · · · · · · · · · · ·
Z	4	+-	 	<u></u>	-	-	 			\vdash		\vdash			
	5 6		-		<u> </u>	-			-		-	-			
	7	+	-			-			_	-	-				
	8	+	-		-	ļ			\vdash		\vdash	\vdash			Figure 6-4b

_	A D D R	co	DE	THE	RMO	ST	ΑT	– J	ΞXΙ	T I	RO	M]	INT	ERE	RUPI	r
[8 25 0	3	E		M	V	I		A	,	0	7			En	able A/D and
	1	0	E 7	,						. 1					מדל.	able A/D and eset counter
Ì	2	D			0	0	7		C	N	7	2				
	3	0	F													
SHEET	4	C	1		P	0	P		\mathcal{B}							
	5 .	D	1		P	0	P		B							
S	6	E	/		P	0	P		H							
CODING	7	E	/		$\overline{\rho}$	0	P		P	3	W					
ျ	8	F	B		É	I			-							
	9	C	9		R	_	7									
	A	1														
	В	1														
	С	1		_		<u> </u>		-								
Σ	D	1														
SYSTEM	E	†						ļ							-	
	F	\top														
TRAINING	8 0	1			3	U	B	R	D	1)	T	エ	N	F	2	REQUIRED
Ā	1				Ť	F			T	R		8	2	7	0	- AF
	2					7	E	М	ρ	- !		8	2	B	0	-EF
MICROCOMPUTER	3				D	A	7	M A		R	E	0	U	I	RI	- AF -EF ED 8310 - 836F
IPU.	. 4	1			Ť	7	E	M	ρ		T	A	B	۷	Ë	8310 - 836F
ő	5							ઝ જ ત								
RO	6						8	3	0	0		0	4		N	i)
2	7						8	3	0	/		0	0			Preload for
-	8						8	3	0	2		0	0			FILTR
	9						8		0	3		0	0)
	A						8	3	0	6						emousture request
SYSTEMS	В						8	3	0	7					م	toredly MAIN
/ST	С															
	D		<u> </u>													
JTE	E															
MPL	F								_				<u> </u>	<u> </u>		
CO	8 0	\perp			<u> </u>	<u> </u>	<u> </u>			<u> </u>	Ŀ					· · · · · · · · · · · · · · · · · · ·
ED	1	\perp	<u> </u>	-		<u> </u>	_									
٦AT	2	\perp	<u> </u>		<u> </u>						<u> </u>					· · · · · · · · · · · · · · · · · · ·
INTEGRATED COMPUTER	3															
IN	4															
	5															
	6	1_	igsqcup		_	_	L	<u></u>				<u></u>				
	7	\perp	<u> </u>					<u>_</u>	<u> </u>	<u> </u>					ļ	
	8						L			<u> </u>					F	igure 6-4c

	A D <u>D</u> R	C	ODE	SUB	ROU	TI	1E	FI	LTR						
	8270	Z	\top	<u> </u>	P	U)	S	H		Ŋ					Save (DE)
	1	C	5		P	V	ک	H		B					Save (BC)
	2	4	6		M	0	V		B		Μ				$(B)\leftarrow n$
L	3	4	18		Μ	0	V	·	C		\mathcal{B}				(c)←n
SHEET	4	2	- 1		工	N	X		#	,					address 2 mEi-1
	5	E			P	V	3	H		\mathcal{H}					Save address
CODING	6	5	L		M	0	V		E	,	M)
Ö	7	2			I	N	X		14	1					(DE) - 2 mEi-1
	8	3			Μ	D	V		D	_	M				
	. 9	6			M	0	V		L	,	EID				(HL) = 2 m Ei-1
	Α	6	. 1		M	0	V		14	,	8				
	827 B	2			D	A	D		Н	, 					
	С	C	D			C	R		C						
ĒM	D	0	2		2	N	Z		8	2	7	\mathcal{B}			>(HL) = 22m Ei-1
SYSTEM	E	17	_												
	F	8	1												
TRAINING	8 280	4	F		M		V	_	C	,	A				(C) F-Vi
\AII	1	7	10		M	0	V		A	,	L				
	2	9	3		S	U	\mathcal{B}		E	,					(DE) (HL) - (DE)
TEF	3	<u></u>	F		M	0	ν		E		A				$=2^{n}(2^{n}-1)E_{i-1}$
ΛΡU	4	7	10		M	0	V		A	,	H				
SO	5	9	A		S	B	B		D	,					
RO	6	3	7		M	0	V		\mathcal{D})	A				
MICROCOMPUTER	7	6	9		M	0	V		L	,	C)
	8	á	16		M	V	I		\mathcal{H}	,	0	0			> (HL) = Vi
	9	0								,)
(0	А	C	D		C	A	L	L		\mathcal{S}	14	F	7	N	Divide ly 2 m (DE) - (2 m-1) E;-1
SYSTEMS	В	A	40												(DE) - (2h-1) E:-1
/ST	С	8	2												
	D	E	B		X	C	1+	G							
TEI	Е		9		Ð	A	\mathcal{D}		\mathcal{D}						(DE) = Vi + (2m-1) Ei-,
MPU	F	E	B		\times	C	H	G							$=2^{m}E_{i}$
S	8 0					د	0	N	7	ェ	N	U	£	٦	AT 8290
ED	1														
NTEGRATED COMPUTER	2				E	N	7	E	R		1	A)	=	Vi (new value)
EGF	3				I	(Н	1)				=	n (filter constant)
IN I	4				((H	4)	+	1	ر		?	2nsEi-1
_	5				1	1	#	1)	+	2)		5	•
	6				R	E	T	U	R	N				Ĺ	
	7	\bot			\coprod	1	1+	1)_	+	3)	=_	(A) = (H) = Ei
	8		-			ŀ									_ (L) = Vi

6-12 Figure 6-4d

	A D D	R	COE	DΕ	FILT	ľR	(00	ont	in	ıed) <i>P</i>	ND	SF	IFT	N	
[8 29	0	E	3		X	ァ	/+	L		\Box					(HL) = address 2nEi
ļ		1	7			M	0	V		M		E)
		2	2	3		I	N			H	7	-				Store 2 mEi
		3	7	2		M	0	V		M		D				
SHEET	 	4	2	3		T	N	X		H	1					address Ei
		5	7	B		7	C	×		D						Jaround up muid
CODING		6	C	D		C	A	L	L		S]+	F	7	N	Factional part > 1/2
릵		7	A	0												
Ĭ		8	8	2												
		9	7	7		M	0	V		M		A				Stone Ei
		A	E	/		P	0	P		H	7					(L) - Vi
		В	6	7		Μ	0	V		1+	,	A				(H) ← Ei
		С	\mathcal{C}	/		P	0	P		\mathcal{B}						Restore B, C, D, E
EM.		D	D	1		ρ	0	P		D						
SYSTEM		E	C	9		R	E	T								Exit
		F	0	0		N	0	P								
TRAINING	8 2 A	0	4	8		M	0	V		C		\mathcal{B}				SHFTN (c)-n
٦¥		1	A	F		X	R	A		A						Loop - clear carry
		2	7	A		M	0	V		A		\mathcal{D}				
JTE	· · · · · · · · · · · · · · · · · · ·	3	/	F		R	A	R			_					(Shift (DE) right
MICROCOMPUTER		4	5	1		Μ	0	V		D		A				Shift (DE) right to divide by 2
Ö		5	7	\mathcal{B}		M	DA	V	_	A		E				
SRO		6		F		R	A			ļ_	_				Ŀ	
Σ		7	5	F	 	M	0	V		E		A		_		
		8	0	D		2	C N	R		EC 8	Ĺ	L			\ <u>_</u> _	to divide lu 2m
		9	C	2		2	N	Z		8	2	A	1			to divide by 2"
ဟု		Α	A	1				_		_						<i>C</i>
	<u> </u>	В	8	2		_	L.	_		<u> </u>	_			_	<u> </u>	
≺S		С	1	0		R	N	C		<u> </u>				_	<u> </u>	Exit if LSB = 0
R. S		D	/	3		I	N	X		D		_		<u> </u>	<u> </u>	Exit if LSB = 0 Else round up (A) - less
7		E	7	\mathcal{B}		M	0	V	_	A	_	E		_	<u> </u>	(A) - less
ΜP		F	C	9		R	E	7		<u> </u>					-	significant byte
S	8	0		<u> </u>		<u> </u>			<u> </u>	<u> </u>		_		_	<u> </u>	
<u> </u>		1	<u> </u>	<u> </u>	<u> </u>	ļ	<u> </u>	_	ļ	<u> </u>		<u> </u>	_	_	<u> </u>	
RAT		2			<u> </u>	_			_	<u> </u>				_	$oxed{oxed}$	
INTEGRATED COMPUTER SYSTEMS		3	<u> </u>	_	L	_	_	<u> </u>	<u> </u>	 		_	_		<u> </u>	
IN		4	<u> </u>	<u> </u>	L				_	↓	<u> </u>		_		<u> </u>	
		5	<u> </u>	_		_	_	<u> </u>	_	<u> </u>	_	_	$ldsymbol{f eta}$	_		
		6	 		<u> </u>	_	_				<u> </u>		<u> </u>		<u> </u>	
		7	 	_		<u> </u>	_		_	<u> </u>	<u> </u>	_	<u> </u>		_	
		8							<u>l</u> _	1						Figure 6-4e

TEMPERATURE LOOKUP AND DISPLAY

8 2 B 0		A D D	R	col	DE	,		,									
1		18 2B	0	E	5		$ \rho $	U	2	4		\mathcal{H}					
2			1	C.	5		P	U	S	Н	,	\mathcal{B}					
3 2 / L X I H, 8 3 / 0 4 1 0 S			2	F	S		P	U	S	14		P	2	ω			
1	—		3	2	/		L		I		14		8		1	0	
	EE		4	/	0		·					,					
BOOD STE MO V B A Copy starting value			5	8	જ				_								
S	N		6	_	6		M	0	ν		B		A		-		7
S	00		7				エ		_			 -	•	-			Constanting walne
9 23 INX H A 56 MOV D, M P2BB 23 INX H C 4 E MOV CI, M (C) - repetitions B 2B E F 5 PV 5 H P 5 W Loop - save A/D SS 50 MOV B, A Stope - save A/D B 2C 0 80 ADD B' stope to low byte of B 2C 0 80 ADD B' stope to low byte 1 27 DAA B A Stope to low byte 2 47 MOV B, A Stope and add 4 7 E MOV A, M Stope and add 4 7 E MOV B, A Stope and add 5 B B ADC E Stope and add 7 5 F MOV E, A Stope and add 8 2B DCX H ADC C Stope and add 9 3 E MVI A, O O Address low byte A O O A A D C C Stope and add SS B P A A D C C Stope and add 6 27 DAA B A C C Stope and add 8 2 B DCX H ADC C Stope and add 9 3 E MVI A, O O Address low byte A O O A A D C C Stope and address low byte A O O A A D A A A D C C Stope and address low byte 5 B P A A D C C Stope and address low byte A O O A A D A A A D A A A D A A A D A A D A A D A A D A D A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A A D A D A A	U		8	1	E		M		V		E		M				(BAD
BY SE B 23 INX H' Log-address count C' repetitions O 23 INX H' Radress slope BY BE F 5 PD 5 H PS W Loop-save A/D BY C 0 PD ADD B Loop slope to low leyte of slope to low leyte of slope to low leyte of slope to low leyte 1 27 DAA C 1 247 MOV B, A Coperan add slope to low leyte of slope to low leyte of slope to low leyte 2 47 MOV B, A Coperan legte 3 23 INX H' Action of slope and add 4 7E MOV A, M ACCION Legte of the second legte 6 27 DAA C F C C C C C C C C C C C C C C C C C			9	2			I		X			7-					()
B2BB 23 INX H' Soop-address count C 4E MDV C/M (C) - repetitions D 23 INX H' BASIN Slope B2BE F5 PD 5H P5W Scop-save A/D B2C B MDV A M Sala low lyte of alone to low lyte of alone to low lyte of alone to low lyte C A7 DAA MOV B, A C C C C C C C C C C C C C C C C C C			Α				M		_		\mathcal{D}	,	М)
WHEN SOUND IN SHEET STORY STOR		828	В				I	-	X			-					Lan- address count
BLOWDOOD WHITE B 2 B E F 5 P U S H P S W Loop - save A/D F 7 E M D V A, M Add low byte of slope to low byte of slope and add to slope to low byte of slope and add to		0.55.5	С	4	_		M	0	V	-		1.	M				
SOUNDED TO SERVICE STORY OF SERVICE STOR	Ē		D				エ		Y		_	7					
SWINDLY S 2 C 0	/ST	82B	E				ρ	_	5	H		P	5	W		-	Long - save A/D
SWELLS STREET OF			F	7			<u> </u>	D	V		A	٠	_	_			Add bar but and
THE HELD IN THE STATE OF TEMPORAL STATE OF TEMPO	ž	82C	0				À	7	D			1		_			slave to law te
BE HELDAMOODON B. A.	¥		1		7		D				\sim						of ton Our Tine
3 2 3 INX H 4 7 E MOV H M 5 PB PDC E 6 2 7 DAPP 8 2 B DCX H 8 2 B DCX H A OO A OO A OO A DC C B PH A DC C Chigh lighter of temperature Chigh lighter of the second lighter 9 3 E MVI A, OO A OO A OO A OO A DAPP B PH A DC C Chigh lighter of temperature C 2 7 DAPP E F / POP PS W 8 0 1 NX H 5 6 6 7			2				_	_	V		B		A) & congression
8 2 B D C X H A A A A A A A A A A A A A A A A A A	LER		3		3	-	ュ	N	X)					address lightente
8 2 B D C X H A A A A A A A A A A A A A A A A A A	PU		4		_		M		7				M	-			
8 2 B D C X H A A A A A A A A A A A A A A A A A A	ΝÖ		5				A		$\overline{}$		F.	/-					the second lute
8 2 B D C X H A A A A A A A A A A A A A A A A A A	ROC		6		7			A	A								of temporation
8 2 B D C X H A A A A A A A A A A A A A A A A A A	/IC		7		F			_	-	_	E	-	A) y sempseum
SWELLS A OO A	~		8	-	\mathcal{B}		D	C	X			- /-	, <u>, , , , , , , , , , , , , , , , , , </u>				address law linter
B P H A D C C C Shigh luyte of temperature of tempe			9	3	Ē	_		ν					0	0		-	
E P / P O P P S W (A) La AD input F 3 C INR A Increment input 2 3 3 4 5 6 7			A				ľ-		_		<u> </u>	1		-			add carry to
E P / P O P P S W (A) La AD input F 3 C INR A Increment input 2 3 3 4 5 6 7	SM		В		\Box		A	D	C		C	_	_				Eliah Protocol
E P / P O P P S W (A) La AD input F 3 C INR A Increment input 2 3 3 4 5 6 7	STE				7			Ã	A						-		temper turk
5 6 7	S			1	7		-				D	Ι,	A				
5 6 7	TER		E		1		$\overline{\rho}$		-			5	W				(A) - AD input
5 6 7	PU			<u> </u>	C	 	I	+	R		A	Ī				-	havement inout
5 6 7	Ŏ	8	0		_			İ	<u> </u>		٦		-				
5 6 7	Ö							<u> </u>		-					-	_	
5 6 7	ATE		2														
5 6 7	GR												_				
5 6 7	Z		4	Γ													
7	=		5	 													
			6														·
8 Figure 6-4f			7														
			8														Figure 6-4f

TWO	BYTE	DISPLAY	(continued)
3	OWT I	I TWO BYTE	I TWO BYTE DISPLAY

ſ	A D D	R	col	DE	1 11111		***									
	8 2D	0	C	A		5	Z				8	2	D	B		Exit when A/D
		1	\mathcal{D}	\mathcal{B}												input is incremented
		2	8	2		ا										te gro
ET		3	0	D		<u>ろ</u> 5	C	R		<u>C</u>						Decrement repetitions
w		4	C	2		\mathcal{I}	N	Z			8	2	\mathcal{B}	E		Loop to add slope
SSH		5	B	E												until all repetitions
Ž		6	8	2				·								of this slape done
CODING		7	2	3		I	N	X P		H						Address high byte
	_	8	C	3		٦	M	P			8	2	\mathcal{B}	\mathcal{B}		Loop to address
		9	-	\mathcal{B}						_						next slape
		Α	8	2					_							
	820	В	E	\mathcal{B}		X	C	H	G L						_	Exit and display Display temperature
		С	C	D		C	A	L	L		D	W	0	R	\mathcal{D}	Display temperature
TEM		D	D	4										-		V .
SYS	-	E	0	2		_			ļ		_		_		 	
و		F	F	4		ρ	0	ρ	L	P	2	W	У	_	_	
Ž	8 2 E	0	C	D		\mathcal{C}	/+	L	L		\mathcal{D}	B	<u>y</u>	T	E	Display voltage
Ä		1	9	5		_	_			-		-	ļ.,	_		C) + voltage
H.		2	0	2						ļ	_	<u> </u>		_		<u> </u>
5		3	E	8		X	C	H	G			_	_			(DE) - temperature
MP	<u>.</u>	4	2			M	V	I		L	,	F	A	_		(DE) - temperature address whole
MICROCOMP		5	1	A	·		_			_		-	_		-	degrees in display
CR		6	7	E		<u> </u>	0	V I	_	A 8	1	M	-		_	
Σ		7	8	6		0	K	1		8	0			-	-	Insert decimal point
		9		0		MA	_	1/	-	10	-	0				<u> </u>
		A	7			M	2	<u> </u>		M A B	+	A C				(0)
ΝS		В	C	7		0	0	0	-	A	,	<u>ر</u>	-			(A) - voltage
SYSTEMS		С	E			0	5	0	_	H	-			_	_	
SYS	·	D	C	9		R	E	$_{ au}^{\prime}$		//	<u> </u>			_		
		E	۲	\vdash		<u>'\</u>	_	<u>'</u>				<u> </u>		-		
5		F					-				-				-	
INTEGRATED COMPUTER	8	0	-		 	 	-	-	-			-		-	\vdash	
O O	-	1	\vdash	\vdash	-					-		-				
ATE		2				\vdash	_		-	-	-		-	-	-	
GR/		3	-	\vdash	· ·	 			-		-	-	-	-	-	*
TE		4	-			-	_	-	-	-	<u> </u>	-	-	-	-	
=		5		\vdash		┢		<u> </u>		-	-	-	-	-	<u> </u>	
		6				-	-		-	-			-	-	-	
ļ		7	-			-	-		-	\vdash	\vdash		-		-	
		8				\vdash	-	-	-	-	 		-	-	 	Figure 6-4g
	L			لـــا		Ь		Ь		<u> </u>	Ь	Ь		Ь	Ь	1 3

8 3 / 0	99				11	
1	- . 	 - 				Starting temperature -023, 761°C as 100 is complement FF - CO 0.405°C/20mo
 	62	 - 	- - - 		+ + -	>-023,76/°C
2	97	 - 				as 100 is complement
3	140					FF -CO
	05	 -				(0.405°C/20mv
	04	 - 				}
6	10		- - -			BF-BO 20.373°C/20mw
	73	 - 				(0.373°C/20mw
8	03					· ·
9	10	 - 				AF-AO 20.341°C/20mv
AA_	41					(0.34/°C/20mv
В	03					
C	20			<u> </u>		9F-80
D E	24					QF-80 20.324°C/20min
E	03					l V
I	10					7F-70
8 3 2 0	35	1				7F-70 0.335 °C/20mm
1	03	<u> </u>				
2	10					6F-60
3	مرح سی	1 _				0.355
3 4 5 6 7	03					
5	10					5F-50
6	94					5F-50 0.394
7	03					
8	10					4F-40
9	58					0.458
A						
B C	08					3F -38
C	32					3F -38 0.532
	05					
E F 8 3 3 0 1 2 3 4	08					37-30 0.606
F	06	- -	11-			0.606
8 33 0	06					
1		- -			7	
2	1	1 - 1 - 1				
3		 	++-	- -		
4		1-1-1	++-		11	
5	1 1	1 - 1 - 1	1 1		11	
6		1 - 1 - 1	_			
7			-			
8		 			1-1-	Figure 6-4h

TABLE OF REPETITIONS AND SLOPES (COntinued)

	A D D R	COL				
	8 0					
	8331	0	4			2F-2C_
	2	8	/			2F -2C 0.681
-	3	1				
SHEET	4	0	4			2B-28
S. S.	5	4	6437			2B-28 0.743
CODING	6	0	7			· ·
200	7	0	4			27-24 0.820
Ŭ	8	2	0			0.820
	9	020	8			
	Α		48			23-20 0.918
	В	/				0.918
_	С	0	9			
MICBOCOMPUTER TRAINING SYSTEM	D	0	4			IF - 1C 1.046
ΥS	E	4	6			1.046
S 5	F	/	0			
Z	8 34 0	0	4			1B-18 1.214
RAI	1	1				1.214
α. -	2	/	2			
JT.	3	0	4			17-14
MP	4	حى			 	1.452
S	5	Z	4 4			
CBC	6	0	4		- - - - - - - - - - - - - - - - - - - 	13-10 1.803
Ž		0	3			1.803
	8	/	8			
	9	0	20			OF - OE 2. 190 00-00 2.524
<u>v</u>	A	9	0			2. 190
TFN	В	2	/		 - - - 	
). >	С	0				00-00
<u>a</u>	D	2	_		 - - - 	2.524
Ė	E .	2	5			
INTEGRATED COMPLITER SYSTEMS	F_	0	2	- 	 	0B-0A 3.089
5	8 2000	8	9			3.089
_ T # F	11	3	0	- 		
P.	2	-	-	 		· · · · · · · · · · · · · · · · · · ·
TEG	3	\vdash	-	 		
ž		├-	-	 - - - -	+	
	5	\vdash	-	├─┤┤ ┼┼		
	7	\vdash		├──┤ ├ ─┤	+	
		-		┞┈╶╏┈┝╸ ┟┈┼┈┤	╌┼╌┼╌┼╌┼╌┤	Figure 6-4i
	8	Ц.	<u> </u>			rigure 6-41

This page intentionally left blank.

	A D D	R	COI						101		<u> </u>			
	R	0			_			Ĭ						
		1												
	835	2	0	/										09
	835	3	07	3										09 3.573
EET		4	1 %	حي										
SH		5	0	1										08
NG		6	6	5						1				08 4.065
CODING SHEET	<u> </u>	7	4	0										,,,,,
ပ		8	40	1			,						П	07
		9	9	7										4.697
	·	A	-	6										
		В	0	1										08 5.537
		C	3	7		H	-			\dashv				5 627
Σ	-	D		5	 	\vdash								0.557
TRAINING SYSTEM		E	5	7			Н	,						17
SΥ		F	0	1		H					_	-		07 6.700
NG.	8 36		0	07					_	-				6.700
Z	000	1	6	7		H		ī						0/ 105071/0
TR/		2	0	0			-	\vdash		\dashv				upper limit for thermistor
ER		3	-		<u>-</u>					-				upper limet for
7		4	0	0		\vdash		-						thermialor
MICROCOMPUTER		5	\vdash				-		-	-				
ö		6								\dashv			Н	
ICR		7	\vdash			-				-				
Σ	ļ	8	 											· · · · · · · · · · · · · · · · · · ·
		9	_			_								
			_		ļ									
JS.		Α			<u> </u>				-	\dashv				
SYSTEMS		В	-		[-	 					
SYS		C						-						
E.		D			<u> </u>		_							
Ę		E F	-						 	-				
MP			_						 -					
NTEGRATED COMPUTER	8	0							$\vdash \vdash$					
TEC		1	<u> </u>			-								
RA		2		ļ										
7EG	ļ	3	_	<u> </u>	ļ				 					
Z		4	_			ļ								
		5				<u> </u>								
		6			ļ	-								
		7			ļ	<u> </u>								
		8			<u> </u>						,		l	Figure 6-4j

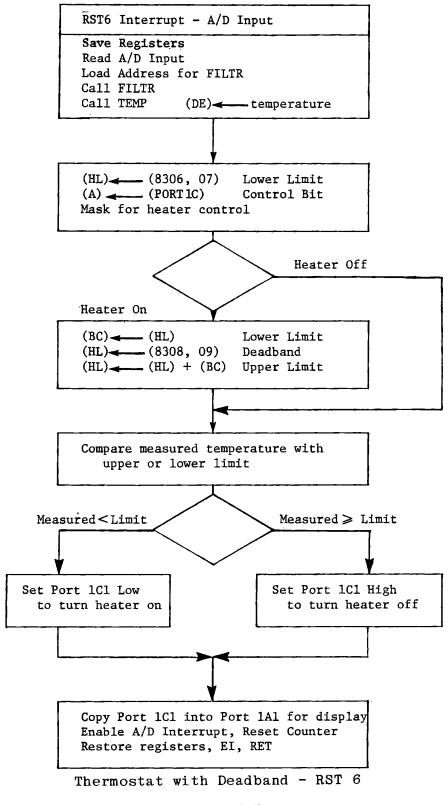


Figure 6-5

6.1.2 On-Off Control With Deadband

EXERCISE

Modify the program of the preceding Section (6.1.1) to provide a deadband. You can do this either by entering upper and lower temperature limits or by entering a value for the deadband which is always added to the lower temperature limit to obtain the upper limit. (The latter approach is used in the given solution.)

The main program is identical to that of figure 6-4a except that instead of storing data entered through ENTWD it calls a subroutine (STORE, at 82FO) which stores a deadband value at 8308,09 if the STEP key was used, or a lower temperature limit at 8306, 07 if the RUN key was used. NEXT stores the data entered as a lower temperature limit and also enters a default deadband of 2 degrees.

Figures 6-5 and 6-6 show interrupt service for the deadband thermostat. The present state of the control is tested to determine whether power is on or off. If it is off, the temperature is compared with the lower limit. If the power is on, the deadband is added to the lower limit before the comparison is made. Now power is turned on if the temperature is less than the selected limit, or off if the temperature is greater.

	A D D	R	со	DE_	THEF	u-1O:	3 1 F.	Y T .	M T T	.п.	JEA	יסמי	ANL		MA.	
	8 2 O	0	3	E		3	٨	I		A		8	\mathcal{Z}			Initialization
		1	8	2							7					same as
		2	$ \mathcal{C} $	3		0	U	\mathcal{T}		C	N	7	/			Initialization Some as Figure 6-4a
_		3	0	7												
EET		4	3	E		M	٧	I		A	•	9	2			
SH		5	9	2	-						,					
ING	 	6	D	3		0	U	\mathcal{T}		C	N	7	2			
CODING		7.	0	7		_	•					,	-			
0	_ · _ .	8	3	E		M	V	I		A		9	4			
		9	9	4		7.1	<u> </u>	_		,,	,		•			
		Α .	2	3		0	υ	\mathcal{T}		7	I	M	C	T		
		В	/	7			U	•				,		•		
		С	3	E		M	V	エ		A		2	0	_		
Σ		D		0		1,1	-	Ļ		17)	Ø	٠	_		
SYSTEM		E	3	3		0	U	$\overline{\mathcal{T}}$		7	I	M	2			
	· · · · · · · · · · · · · · · · · · ·	F	1	_		U		1		/	<u></u>	7.1	<u>a</u>			
TRAINING	821	0	1	3		7	7		-					-		V 7 . 7
Z	0 26 /	1	-	-		DM	エソ	7		0		۸	2			Data entry same as Figure 6-4a
TR/		2		E	_	18/	V	ユ		A	1	0	3			as Figure 6-4a
			0	3						1		\	_	,	0	
Į		3	D	3		0	U	T	_	12	0	R	7	1	١	
MICROCOMPUTER		4	0	6		_						6)	_		
ည်		5	D	3		0	U.	1_	_	ρ	0	R	T	/	A	
CR(6	0	4			_	_								
Σ		7	C	D		\mathcal{C}	A	<u>L</u>	1		E	N	7	W	D	
		8	4	6				_								
		9	0	3												+
S		Α	0	Ŋ		C	17	2	4		S	7	0	R	E	Stretemperature
SYSTEMS		В	/=	0						_						limet at 8306,07
YST		С	8	a						_						ore deadband at 08,09 Enable A/Dinterrupt Just Keyboard
		D	F	7		R	5	T	6		_		_			Enable A/Dinterrupt
린		E	2	\mathcal{B}_{\perp}		I	N			P	0	R	T	0	A	Jest Keyboard
MPL		F	0	0		<u> </u>			<u> </u>						 	0
INTEGRATED COMPUTER	822	0	3	C		I	N	R		A						
ED		1	C	A		2	Z		<u></u>	8	2	1	E		_	Loop until
₹AT		2	/	E		Ŀ	<u>_</u>			<u>_</u>						Lug pressed
EGF		3	8	2												0.
IN		4	C	3		2	M	P		8	2	/	0			Loto DI and
_		5	/	0												1
		6	8	2												and accept data
		7														
		8														Figure 6-6a

	A D D R	COL	DE												
	8 23 0	F	5		P	U	5	H		P	S	W			,
	1	E	3	,	P	U	S	\mathcal{H}		/+		•			
	2	2	5		ρ	U	S	H		D					
.	3	0	5	,	P	U	3	14		B					
SHEET	4	2	7		/	X	I		14		8	α	1	4	1000 Jule 71 TP
R	5	0	0		_		_			7	3	Ç	_		address for FILTR
S	6	8	3								_			-	
CODING	7	Ď	\mathcal{B}		I	N			ρ	0	R	\mathcal{T}	,	\mathcal{B}	Read A/D
ŏ	8	1			<u>+</u>	<i>N</i> _	_	-	-		<u> </u>	1	/		read n/D
	9	0	5 D		C	Λ	,	۷	-	F	7	,	7	ρ	(0), (1)
	A	7	$\frac{\sigma}{\lambda}$			A		_		_	I	_	工	R	(A) - filtered A/D voltage
		7	2								-		_	-	A/D voltage
	В	8			1	_	,	,		_	_				
5	C	_	Δ		<u>c</u>	A	L	4		7	ے	M	P	-	Calculate temperatur Display
TE	D	B	0			-					_	ļ	_	ļ	
SYSTEM	E	8	2	_							_	_	,		(DE) - temperature
	F	2	A		<u>L</u>	1+	4	D		8	3	0	6	ļ	(HL) - low limit
N	8 240	0	6			-							_	-	for temperature
TRAINING	1	8	3			ļ					_	_	<u> </u>		
ER T	2	D	B		I	N			P	0	R	7	1	C	(A) - controls
JTE	3	0	6											ļ ·	
MP	4	E	6		A	N	I		0	2	_		<u> </u>	L.	Lest for heater
MICROCOMPUT	5	0	2								_		_	ļ	on orlott
RO	6	C	2		7	N	Z		8	2	5	0			Jump if of
Σ	7	8	0										_	L.	(Port ICI high)
	8	8	a												
	9	4	Δ		M	0	V		C		Ľ				Leater on
	Α	4	4		M	0	V V		CB	7	H	0			(BC) - temp request
EMS	В	2	A		7	H	L	D		8	3	0	8		(BC) - temp request
ST	С	0	8								-				
S.	D	8	3		,			,							
正	E	0	9		D	A	D		B						(HL) - upper limit
Ē	F	0	0		N	A	P				-		T		and the same
NTEGRATED COMPUTER SYSTEMS	3 0	1	_		<u> </u>	_									
۵	1	†						†			1		-		
ATE	2										 - 		T		
GR,	3	+				+-	\vdash		\vdash	-	-		\dagger	+	
J F	4	+			 	<u> </u>	-	<u> </u>	 	-	_	<u> </u>	├-		
=	5	+	<u> </u>	<u> </u>	╁╌	\vdash		-	-	\vdash	-	-	\vdash	1	
	6	+	-		\vdash	\vdash		 	-	-	 	 	-	-	
	7	+			\vdash	-	-			\vdash	 	-	-	-	
		+	-	-	┝	-	-	 	-	-	├	-	├	+-	Figure 6-6b
	8	1	1	L	<u>L_</u>	<u> </u>	<u> </u>		<u> </u>	L	1	<u></u>	<u>L</u>		trante e-en

	A D D	R	coi	DE	THE	1/1/1/	JOI	ΛI		1(01	. 0		(CO	11 6 3	Lnu	cu,
		0	7	D		M	0	\vee		A		4				Compare temperature
		1	9	3		S	U	3		Ē)					Compare temperature with limit
	· · · · · · · · · · · · · · · · · · ·	2	7	C		Μ	0	ν		A		14				Sets CY if
		3	9	17		S	B	B		AD)	, ,				temperature high
SHEET		4	3	E		М		ゴ		A		0	/			(A) = 03 id high
		5	0	/							,					€ 62 il low
N		6		7		R	A	L)
CODING		7	D	3		0	U	$\overline{\mathcal{T}}$		C	N	7	/			Set Port 101 low
	-	8	0	7	_											(power on) it
	- !	9	2	B		I	N			P	0	R	7	/	C	temperature low
		Δ.	0	6												——
		В	D	3		0	V	\mathcal{T}		P	0	R	1	/	A	Display power control
	(C .	0	4												
		D	3	E		Μ	V	I		A		0	7			Leenable and
SYSTEM		E	0	7							,					reset A/D
G S		F		3		0	V	$\overline{\mathcal{T}}$		C	N	7	ટ્સ			,
Z	8 26	0	0	F												
TRAINING		1	C	/		P	0	ρ		\mathcal{B}						
	`	2	D	/		P	0	$\overline{\rho}$		D						
TEF	;	3	E	/		P	0	P		14						
MICROCOMPUTER		4	F	/		ρ	0	P		P	S	W				
201		5	F	B		E	ユ	,								
RO((6	C	9		R	E	T								
MIC		7														
	1	8				6	U	В	R	0	υ	Т	エ	Ν	E	S REQUIRED
		9					F	I	L	T	R		8	a	7	O-AF
		A					7	E	Μ				8	2	B	S REQUIRED 0-AF 0-EF RED E 8310-836F
EMS	ı	В				D	Á		A	_	R	E	0	U	I	RED
/ST	(C					T		M			7	A	B	L	E 8310-836F
3 S	Γ)														
TEF	I	E					8	3	0	0		0	4			n
ΛΡU		F ,					8	3	_	/		0	0			Theload for
COL	3	0					8	3	0	2		0	0			FILTR
ED (1					8	3	0	જ		0	0)
\AT		2														
EGF		3					8	3	0	6	_,	0	7			Temperature limit
INTEGRATED COMPUTER SYSTEMS		4					8	3	0	8		0	9			Sengerature limit Deadband Stored by STORE (82FO-82FF)
_		5														Stored by STORE
	(6														(82FO-82FF)
		7														<u> </u>
		8														Figure 6-6c

	A D D R	_ co	DE					OB	ROU	LT.T.	NE	ST	ORE	· ·	
	8 2F 0	F	E		C	ρ	\mathcal{I}		R	U	N				Lest command
	1	1	4												
	2	2	A		J	C			8	2	F	C			4 STEP sumpto
_[3	F	C												of STEP jump to
SHEET	4	8	2												
	5	2	Q		S	H	L	D		8	3	0	6		Else store lower
CODING	6	0	6												temperature
읽	7	8	3						-						
7	8	C	8		R	Z									Eit it RUN
	9	2	1		L	X	\mathcal{I}		#		0	0	2	0	Exit if RUN If NEXT enter
Ī	A	2	0							,	<u>, </u>				default deadband
ı	В	0	0												61 20°C
ı	С	2	2		C	H	4	D		8	3	0	8		SPEP (WNEXT)
2	D	0	8			,				<u> </u>	_				Store deadband
SYSTEM	E	8	3												and in minutes of
Ś	F	C	9		R	E	T							_	
ĭ	8 0	1	'	- 1			' -				-	_	 		
≨l	1	1												<u> </u>	
	2	†				-									
MICROCOMPUTER TRAINING	3	 		-	E	N/	T	E	R		W	I	τ	Н	
∑l	4	T			<i></i>	7	#		1	=	τ		M		ERATURE °C
Š	5	\dagger				1	1				7		0	W	ER LIMIT)
ğ	6	1-			_	-					0	R		D	
	7	†	-			7	A	5		=	2	0	M		AND
2	8	+				7	M	_	A	N	D	S	_	A	
ł	9	 	-				R	1)	N	HV_	_		9	T	ARE TEMP ANIV
t	Α	†				†	3	T	E	P	_		2	+	ORE DEADBAND ORE TEMPERATURE D SET DEFAULT ADBAND=2.0°C
Σŀ	В	+-				 	1	E		`		-	0	+	OPE TEMPEDATUR
띩	c	+	 			 	17	<u></u>	^	_	-	_	<u>Δ</u>	,	D CET DEFAULT
S	D	+	_			<u> </u>	-	-		_	-	-	7	7	D-3617- 10°C
띪		+	+-			+-		-	-			\vdash	٢	_	110011100-2.00
5	F	+	-	 	-	+-	+-	-	\vdash	\vdash	-	-	-	\vdash	
ĕ	3 0	+				 	-	-	-	-	-	-	-	-	
INTEGRATED COMPUTER SYSTEMS	1	+	_			+	\vdash	-	-	-	-	-	-	\vdash	
밁	2	+	-	-	_	+-	 	-	-	-	-		-	-	
GR/	3	+-	┼-	 	_	+	+-	-	-	\vdash	-	-	-	-	
ĪĒ	4	+-	-			-	-	-	-	-	-	-		-	
=	5	+	+-	-	\vdash	+	-	-	\vdash	-	-	-	-	\vdash	
	6	+	 -	 	├	+	+	\vdash	\vdash		-	-	-		
		+	├-	1	\vdash	₩	+-	├-	-	-	├-	-	├	\vdash	
	7		1			1			1	•					

6.1.3 Thermostat with Alarm Limits

OPTIONAL EXERCISE:

A very common requirement in process control systems is to test a temperature for certain limits and give an alarm if the limits are exceeded. This is likely to be used in conjunction with a temperature control, either thermostatic or proportional, so there may be as many as five limits:

Highest - Danger to equipment or personnel.

Probably indicates an equipment failure.

Next Highest - Process is out of control. May result in degraded product.

Normal High - Heater should be turned off to maintain normal process temperature.

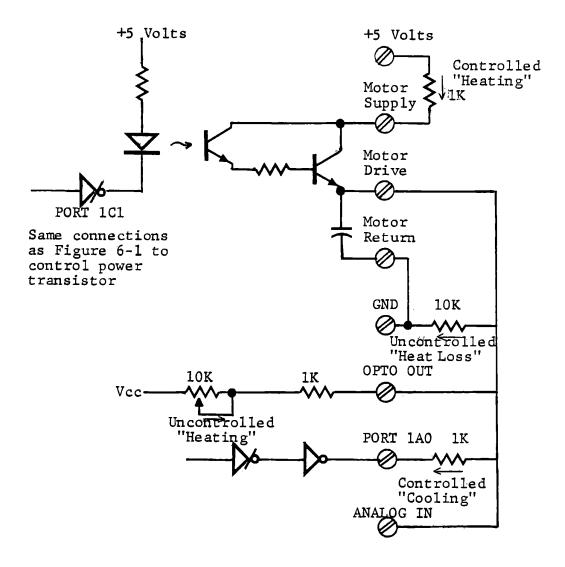
Normal Low - Heater should be turned on to maintain normal process temperature.

Lowest - Process is out of control. May result in degraded product.

In a batch process, of course, the initial and final temperatures are likely to be lower than the lowest control temperature, so the alarm corresponding to that temperature should not be given until after the normal low temperature has been reached, nor after the process has been turned off.

There may also be time limits imposed on the process. In Section 4.2.9 an optional exercise was suggested in which a heater was controlled according to a schedule of times, with an upper limit for off-time and a lower limit for on-time. We will impose such limits here. In addition, we will place an upper limit for on-time, because if the heater stays on too long it may indicate a failure in the temperature sensor such that the highest temperature limit could be reached without being detected.

Develop your own flow charts and program for this problem. Select limits that can be reached by the heater you are using, and force the alarm conditions to occur by connecting MOTOR CONTROL - to ground so that the heater will stay on.

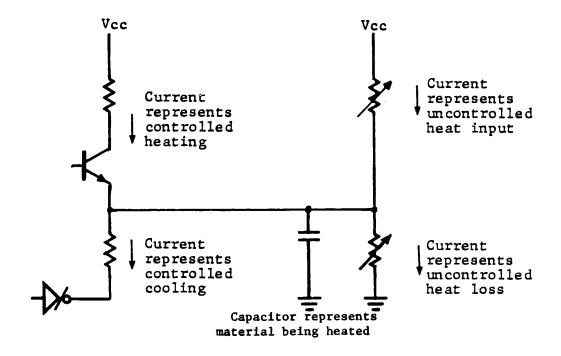


Circuit Connections for Simulation
Figure 6-7

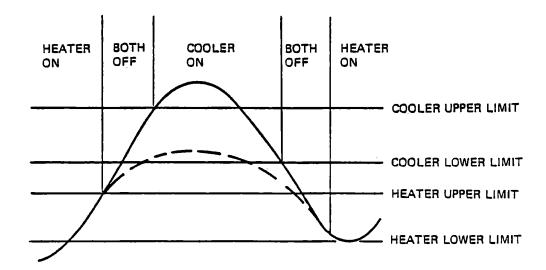
6.1.4 Two-Way Control

OPTIONAL EXERCISE:

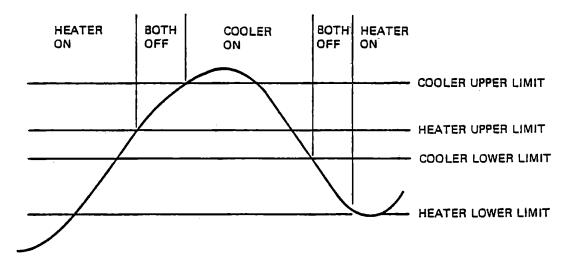
In the simple thermostat problem it was assumed that only heating would be needed to maintain a required temperature; incidental heat loss would provide for cooling. In many temperature control problems that is not the case - both heating and cooling are required. The building with both furnace and air conditioner is the obvious example, but chemical processes also may require both. We cannot readily provide a realistic exercise with a heater and cooler, but the functions are easily simulated by charging and discharging a capacitor. Figure 6-7 shows connections to the interface board, and Figure 6-8 shows a circuit diagram for such a simulation.



Heating and Cooling Simulation
Figure 6-8



Normal Inner Limits



Inverted Inner Limits

Heating and Cooling Limits
Figure 6-9

We use the SENSE pot to simulate a variable heat input that cannot be controlled by the program and a 10K resistor to simulate heat loss. (An external 10K pot can be used instead, but is not necessary). The power transistor and Port 1AO, each with a series 1K resistor, represent the controlled heating and cooling.

Figure 6-9 depicts the control process. In Figure 6-9a a normal situation is shown. As the process temperature rises with the heater on, it crosses the heater lower limit and reaches the heater upper limit, where the heater is turned off. If either thermal inertia or uncontrolled heating causes the temperature to reach the cooler upper limit the cooler is turned on to return the process toward the desired temperature, and it runs until the temperature drops to the cooler lower limit. if the temperature does not reach the cooler pper limit the cooler remains off (dashed line). Obviously in the case of temperature control of a building (rather than a chemical process) there are many days when only the heater is turned on or only the cooler is turned on.

In process control however, it is often the case that the cooler involves greater thermal inertia than the heater. Heating can usually begin very quickly after a control signal and also stop very quickly. Cooling is likely to involve significant delay, since cold water must be pumped through pipes that are full of water heated by the process, and when the cooler is turned off the remaining cold water in the pipes will still absorb heat. In such a case the heater upper limit may be above the cooler lower limit, as shown in Figure 6-9b. In an extreme case, both cooler limits might be between the

two heater limits. This implies that sometimes both the heater and cooler would be in operation at the same time. The conclusion is that independent upper and lower limits should be provided for both the heater and the cooler. (These may of course be expressed as lower limit and deadband).

Clearly the control algorithm is essentially the same as for the case of a heater (or cooler) only, but must be processed twice for each temperature measurement. Once again we leave program development to the student.

6.2 PROPORTIONAL Vs. INTEGRAL CONTROL

On - off control is unsuitable for many purposes. Imagine steering a car with a three position switch allowing only left turn, straight ahead, or right turn. You might make your way along a sufficiently broad highway, but the turn radius suitable for high speed driving would made parking very difficult. Proportional control is a method of applying a varying control force depending on the magnitude of the adjustment required.

In steering a car along a straight road you may be able to take your hands off the steering wheel for several seconds, and continue in a straight line. Soon, however, the car will drift off the track and a gentle touch on the wheel will be needed to return to the straight line. You have observed an "error signal" and applied a "control force" to correct the error. When the error becomes zero you again relax the control force.

If a bump in the road caused the car to swing sharply you would apply a more vigorous control force through the steering wheel. This is the essence of proportional control: a larger error results in a stronger restoring force.

When you reach an intersection and want to turn, you have changed the "setpoint". Suddenly a large error signal appears, because now the intended direction (i.e. setpoint) is pointed 90 degrees away from your car's current direction. You apply a large control force to correct for this large error signal, until once again the error signal reaches zero. Thus the control force is in direct relation or

proportional to the error signal. A proportional control system can be described by:

Control Force = Gain (Desired Value - Measured Value).

The difference between the desired value and the measured value is called the error signal. It may be temperature, distance, angle, speed, voltage, or any of a host of other continuous variables. The control force might be electric current or gas flow to a heater; voltage, current or frequency to an electric motor; hydraulic pressure, etcetera. Note that gain is usually not a dimensionless ratio in this abstract form. For instance if a measurement in degrees is to give a control force in pounds, gain would have the dimensions "pounds per degree". In many cases with computer control both the measured value and the control force may appear as voltage analogs of the real variables.

In some processes it is not enough to provide only a correcting force. It may be necessary to provide some driving force all the time. For instance, we might operate a heater at some steady current, but increase or decrease that current in response to a temperature measurement. Then the control force would be:

(a) F = G(E) + S

where F = control force

G = gain

E = error signal as above

S = steady state force

Now if the system is well understood and conditions are constant, corrective changes are made in the control force only when some disturbance causes an error signal.

If you were driving a camper or truck on a windy highway you would not take your hands off the steering wheel. Some force is needed all the time to keep the truck going in a straight line. This force must be increased or decreased momentarily to compensate for gusts or bumps. It is not a constant force, however, but must be adjusted if the wind force changes. Systems of this kind, that require a continuous control force to be adjusted for both momentary and long term changes, are more common than those where the control force is usually zero or constant.

The first kind of control system, with no steady state force, is called "proportional control" or "pure proportional control". It is well suited to processes that are subject to temporary disturbances or changes in setpoint, but will otherwise do what is intended by themselves. A simple autopilot in an airplane or boat detects any error between the compass and the setpoint and applies a control force to reduce the error to zero. Until a wind gust or a wave disturbs it, the vessel will go in a straight line.

When some control force is required all the time, as on our windswept highway, pure proportional control is not satisfactory because it will deliver a control force only when there is an error signal - as our camper goes off the road. To provide the continuously adjustable control, we would use "integral control". This system is named from its control equation, which is described below. Integral control can provide a steady state control force that will maintain a zero error signal when there are no disturbances. It will correct for disturbances such as the wind gusts, and it will adjust the steady state force in response to changing conditions.

Pure integral control also has a weakness. It is inclined to overcorrect for momentary disturbances because it cannot immediately distinguish between (for instance) a gust of wind and a change in the wind force. When both momentary and long term changes in the conditions will occur it is best to use a combination of proportional and integal control.

In the remainder of this chapter we will develop a "proportional plus integral" control system, working up by steps from an open loop system of pulse width modulation, to a pure integral control system, and finally to the combined system. Much of the description, the program, and the experiments will be devoted to observing the behavior of the system. First we will develop the control equation for an integral control system to see why it is so named.

In pure integral control we adjust the control force whenever an error signal occurs, and maintain that new control force until another error signal is measured.

(b)
$$F = GE_1 + GE_2 + GE_3 ----$$

As long as repeated measurements indicate a positive error (desired value greater than measured value) the control force will be increased. Eventually it will be enough to make the error negative, and the force will be reduced. After a time, if the control system is stable, the force will reach just the right value to maintain a Lero error under the existing conditions. The sum of a series of measurements of a variable is the integral of that variable, so if equation (b) is expressed as:

$$F_{i} = GE_{i} + F_{i-1}$$

or

(c)
$$F_i = G \int E$$

the derivation of the name "integral control" is apparent.

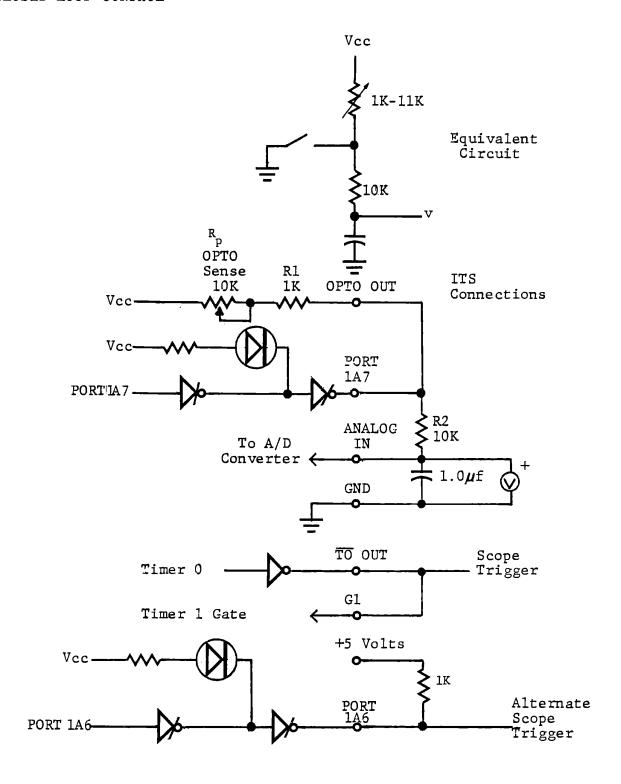
In the following experiments we require both analog output for the Control Force and analog input to determine the error signal. Since the interface board uses the D/A converter for A/D conversion we will use pulse width modulation to generate a voltage on a capacitor for the digital to analog output and measure that voltage with the automatic A/D input. Clearly, in any of these exercises the output voltage or the pulses could be driving some other load such as a heater or a motor, and the voltage input could be obtained from a

thermistor for heat, a linear potentiometer for position, or some other sensor with a voltage output. It is important here that control is exercised so as to force a measurable end result to be equal to a desired value.

In the first exercise we will develop some of the program modules necessary for the Pulse Width Modulation Control System: Initialization, Data Entry via Keyboard, Digital Data Filter, Digital Voltmeter Display, Interrupt Service for PWM and the Main Loop. However, we will not yet develop the modules responsible for Error Signal generation until section 6.2.4. This will allow the opportunity to experiment with an Open Loop PWM Control System since the system cannot perform error detection and correction. We can thus demonstrate how the system output is dependent on external factors (in this case, the setting of the OPTO SENSE pot). In the second exercise (section 6.2.3) we will observe the system's response time characteristics via data logging, and an oscilloscope if available.

In the third exercise (section 6.2.4) we will develop the program modules necessary to close the loop: Error Signal Calculation and Display, Control Force Calculation and Modification. We will also enable additional command keys to select either Open Loop or Closed Loop control. We can thus demonstrate how Closed Loop Control eliminates the system output's dependence on external factors.

Then, in section 6.2.6, we will experiment with various aspects of Closed Loop Control. Access to an inexpensive oscilloscope will be helpful.



Connections for PWM Experiment

Figure 6-10

6.2.1 Voltage Control Circuit

The circuit of Figure 6-10 is similar to that used in Section 4.2 for pulse width modulation, except that a capacitor is included to average the PWM signal. It is charged through the OPTO SENSE pot to introduce an external variable not controllable by the program. If this were to drive a load, an external amplifier would be needed, but we will not be concerned with that here. The capacitor voltage is measured both by the A/D converter and the voltmeter.

When Port 1A7 is high (output turned off) the capacitor is charged through the SENSE pot (Rp) the internal resistor R1, and the external resistor R2. It charges toward the Vcc (5 volt) supply, with a time constant of RsC,

where
$$Rs = Rp + R1 + R2$$
.

When Port 1A7 is set low, the capacitor discharges through R2, with a time constant (R2)C. With the resistances shown the charging time constant ranges from 11 to 21 milliseconds, according to the pot setting; the discharge time constant is ten milliseconds. If we operate pulse width modulation with a cycle time of a few hundred microseconds the capacitor voltage will change only slightly during each cycle.

Using a linear approximation, the voltage increases during charging by:

(a)
$$\Delta v_c = \frac{t_c}{R_s C} (v_c - v)$$

where tc = charging time

RsC = charging time constant

Vc = supply voltage (5 volts)

v = capacitor voltage

When Port 1A7 is low the capacitor discharges by:

(b)
$$\Delta v_d = \frac{t_d}{R_2C}$$
 (v-V_g)

where td = discharging time

R2C = discharging time constant

Vg = voltage drop across the open collector buffer
(0.2 volts)

Provided the pulse sequence is maintained for a time greater than several time constants, a steady state will be reached where the charging and discharging are equal in each cycle. Then:

(c)
$$\frac{t_c}{R_c^C}$$
 $(v_c - v) = \frac{t_d}{R_2^C}$ $(v-v_g)$

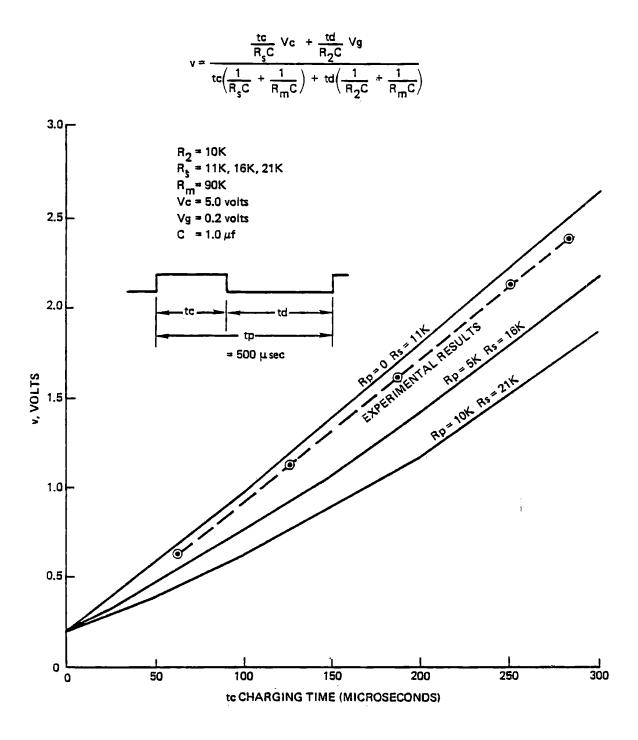
This equation can be solved for the steady state average voltage:

(d)
$$v = \frac{\frac{t_c}{R_s C} v_c + \frac{t_d}{R_2 C} v_g}{\frac{t_c}{R_s C} + \frac{t_d}{R_2 C}}$$

The value of the capacitor factors out of equation (d), showing that the steady state average voltage is independent of the capacitor. The equations above neglect the voltmeter, which drains a little current to ground. Accounting for the voltmeter resistance (Rm) leads to equation (e):

(e)
$$\frac{\frac{t_c V_c}{R_s}}{t_c \frac{1}{R_s} + \frac{1}{R_m} + t_d \frac{1}{R_2} + \frac{1}{R_m}}$$

This equation is plotted in Figure 6-11 for a fixed total period of 500 microseconds and varying charging time, for three different values of potentiometer resistance. Note that with zero resistance in the potentiometer the voltage is nearly linear with charging time. The voltage is only moderately sensitive to the pot setting because of the fairly large value (10K) of R2, but the pot does cause a substantial departure from linearity. Figure 6-11 also shows experimental results generated with the program to be developed.



PWM Voltage - Fixed Period
Figure 6-11

With constant total period tp = tc + td we can solve equation (e) for the ratio of charging time to total period required for any desired voltage. R_{m}

$$\frac{t_{c}}{(f)} = \frac{v - \frac{R_{m}}{R_{2} + R_{m}} v_{g}}{\left(\frac{R_{m}}{R_{2}R_{m}}\right) \left[v_{c} \frac{R_{2}}{R_{s}} - v_{g} + \left(1 - \frac{R_{2}}{R_{s}}\right) v\right]}$$

If the pot is set to zero resistance we have the following values:

$$\frac{R_{m}}{R_{2} + R_{m}} = \frac{90K}{10K + 90K} = 0.90$$

$$\frac{R_{2}}{R_{s}} = \frac{10K}{10K + 1K} = 0.91$$

$$V_{c} = 5.0$$

$$V_{g} = 0.2$$

Then:

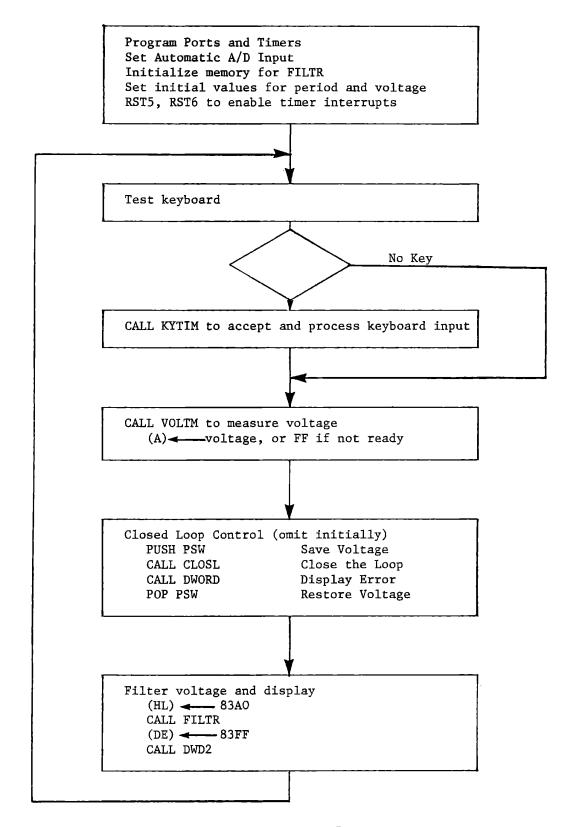
(g)
$$\frac{t_c}{t_p} = \frac{v - 0.18}{3.91 + 0.08v}$$

Since the .08v term in the denominator is quite small the relationship is nearly linear when the pot is set to zero, as shown in Figure 6-11. We will use this linear relationship in our pulse width modulation scheme.

This page intentionally left blank.

Timer 0 will be loaded with a count for the total period; at each interrupt from this counter Port 1A7 will be set high to start charging. Timer 1 will be loaded with a count for charging time, derived from the desired voltage. Timer 1 will be started when charging starts, and its interrupt will set Port 1A7 low to stop charging.

We will enter a desired voltage in hexadecimal with the least significant bit representing 10 millivolts, just as the A/D converter represents a voltage. Subtract 12 (representing 0.18 volts). This value will be used to determine the charging time in microseconds, so double it to account for two system clocks per microsecond, and load the result to Timer 1. The total period, loaded to Timer 0, should correspond to 3.91 volts or 391 microseconds. This is given by a hexadecimal value of 30E (= decimal 782). This value must be adjusted to compensate for resistor and voltage tolerances and the error caused by neglecting .08 v. The program provides for keyboard input of the total period so that the adjustment can be made while the program is running.



PWM Voltage Control - Main Loop

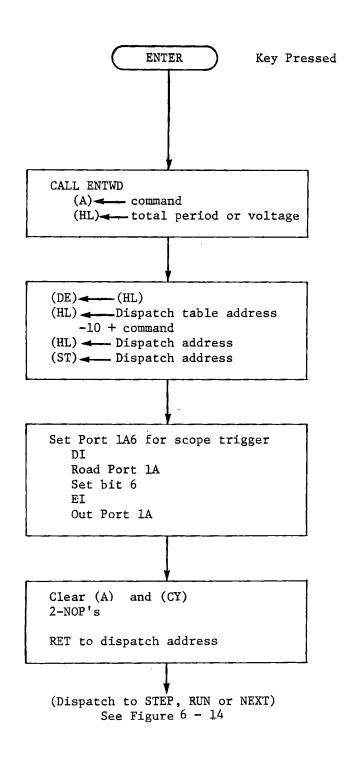
Figure 6-12

6.2.2 Voltage Control by PWM

EXERCISE:

In the program of Figure 6-12 we repetitively measure and display the voltage at the analog input, and test the keyboard for data entry. The voltage is determined by pulse width modulation under interrupt control; this is discussed in Section 6.2.2.4. The main loop in Figure 6-12 includes a call to a closed loop control subroutine that will be developed in Section 6.2.4. For developing the open loop program you can enter eight NOP instructions. The following sections discuss the three subroutines KYTIM, VOLTM, and FILTR.

As always, it is recommended that you study the problem and develop your own solution for each program module. Follow the flow charts given here closely, because a number of revisions will be made as we develop the closed loop system. With various modifications and additions we will use this program through the remainder of Chapter 6.



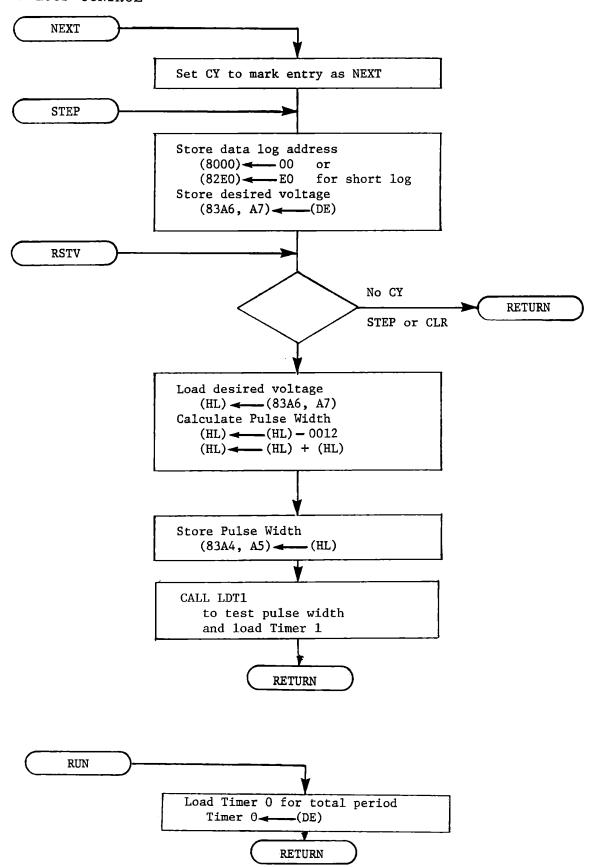
PWM Voltage - Subroutine KYTIM
Figure 6-13

6.2.2.1 Data Entry Subroutine KYTIM

Subroutine KYTIM (Figure 6-13) is called for data entry when the main loop detects a key depression. KYTIM calls ENTWD (0346) to accept up to four hex keys (returned in HL) and a command (returned in A).

A dispatch table is used for distinguishing among the keys. Although it is not necessary to treat all keys differently at present, we will add some functions later. One of the main purposes of this exercise is to observe the effects of different control algorithms. The program includes a data logging facility, and also provides for oscilliscope observations. Immediately after looking up the dispatch address and before jumping to the process, Port 1A6 is switched high to give a scope trigger for use in observing the response time after a new voltage is keyed in. This output is available at a tie block, but must be pulled up through a resistor since it is an open collector buffer. Using any undefined key will generate the scope trigger without changing the pulse width or total period; the ADDR key will be reserved for this function.

Since both KYTIM and interrupt service alter data at Port 1A, interrupts should be disabled while KYTIM is manipulating the port. To maintain good control, however, the time during which interrupts are disabled should be as short as possible.



PWM KYTIM - Set Pulse Widths

During debugging you may want to place a breakpoint at the RET instruction that dispatches to the command processing module. Recall that the monitor interrupt cannot occur until an instruction has been excuted with two or more machine cycles after EI. To overcome this limitation, place the EI before OUT PORTIA, which requires three machine cycles. No interrupt can occur until the OUT instruction has been executed; then either of the hardware interupts or the monitor interrupt can occur.

The key processing modules are described below and shown in Figure 6-14.

If the command is NEXT or STEP the input data represents a desired voltage. Before storing and processing this value, memory location 8000 is cleared to initiate a new data log. The desired voltage is then stored at (83A6, A7). For the data logging function it is important that these steps occur in the sequence indicated.

The STEP key calls for the same processing as NEXT up to this point, but the following steps are omitted by a conditional return if carry is clear, indicating that the command was STEP. The STEP key is useless in the open loop system but is needed for closed loop control. The RSTV ("restore voltage") entry also reaches this conditional return. In a later modification of the program the BRK and CLR keys will jump to this entry after performing their other functions, and the conditional return will be executed for CLR but not for BRK.

Now the required pulse width is calculated. This is done in response to NEXT, and it will be done in response to BRK. Because of this alternate entry the desired voltage is loaded into (HL) from the location (83A6, A7) where NEXT or STEP has stored it. Subtract 0012 to allow for the zero offset voltage Vg (0.18 volt) and double the result to give two clocks (one microsecond) for each ten millivolts. Then the calculated pulse width is stored at (83A4, A5). Finally, subroutine LDT1 is called to load Timer 1 with the new pulse width.

6 2.2.2 Subroutine LDT1

Timer 1 is to be loaded with a calculated pulse width both in response to the NEXT key and later under closed loop control. It is entered with a pulse width as two bytes in (HL).

Before actually loading Timer 1 we check that a legitimate pulse width has been entered. The system will go out of control if a very long time is entered to Timer 1. This could occur in the closed loop system if a negative width were calculated, or in the open loop system if you request a voltage less than 12. The timer treats 0000 as a maximum delay, so we also test for this value. One way of making the test would be:

PUSH H	Save original value
DCX H	Force 0000 to FFFF
DAD H	High bit to carry
РОР Н	Restore original value
RC	Exit if zero or negative

The above method has the virtue of changing only the carry flag.

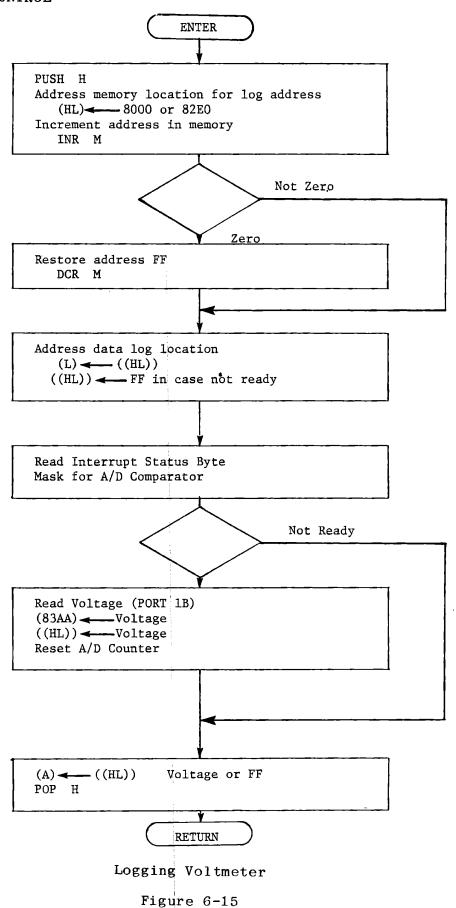
Another method is:

DCX H	Force 0000 to FFFF
MOV A, H	Set all flags according
ORA A	to content of H.
INX H	Restore original value
RM	Exit if zero or negative

This method, which the author has used, is faster and avoids using any stack area.

Provided that the test indicates a legitimate pulse width, Timer 1 is loaded with the data. This sets the charging pulse width. The RUN key copies the input data from ENTWD to Timer 0, to set the total period.

This page intentionally left blank.



6.2.2.3 Voltmeter Subroutine VOLTM

The automatic A/D converter is used to measure the capacitor voltage. To permit observing response times without additional instruments, the voltmeter subroutine will log the voltage each time it is called. The following initialization steps are necessary:

Program Port 1B for input.

Set Port 1C1 high to enable counting.

Program and load Timer 2 to divide the system clock by 8. Store an initial address for the data log.

Although in this program VOLTM is called from the main loop, and returns the measured voltage in register A, it is written to permit its use as an interrupt service routine. It stores the measured voltage in a fixed memory location and also logs the voltage at an address obtained from memory. Figure 6-15 shows the subroutine.

Memory page 80xx is allocated to the data log. Address 8000 stores the low byte of the last address used in the log. This location is loaded with 00 each time a new log is to be started. VOLTM increments the content of location 8000 to count the number of times it has been called, and uses the new value as the low byte of the log address. We are interested in the behavior of the system for a relatively brief period ater a change in the system setpoint (the voltage request), and do not want to destroy those data with later measurements. Therefore, the logging address is incremented only up to FF, and thereafter remains fixed.

Since VOLTM is called by the main program, not by an A/D interrupt, it is possible that a voltage conversion will not be ready. If the input voltage is greater than 2.55 volts it cannot be measured by the A/D converter because the comparator will always see the input greater than the D/A output. The comparator signal appears in bit 3 of the interrupt status byte. This is tested by IN PORT2B, ANI 08; if the bit is low the subroutine returns a value of FF with the zero flag set. If the comparator signal is high the voltage is read by IN PORT1B, and will be returned with the zero flag not set. Before return, however, the A/D counter must be reset by disabling the A/D interrupt, to start a new conversion.

Note that this subroutine will also return FF and Zero set if the voltage is less than 2.55 volts but insufficient time for the conversion is allowed between calls. In the program being developed, however, enough time is taken by the main loop and subroutines to ensure a valid conversion if the voltage is less than 2.55 volts.

6.2.2.4 Using FILTR

The voltage generated by pulse width modulation is averaged, or filtered, by the capacitor. It fluctuates by about 50 millivolts, so the voltage measured will vary in the less significant bits. To obtain a valid eight bit measurement we will use subroutine FILTR to provide additional filtering for display of the output.

FILTR is called with the raw (unfiltered) voltage in (A) and a memory address (83A0) in (HL). That memory location must be loaded with 1,2,3 or 4 during initialization, and the following two bytes must be cleared initially.

FILTR returns the input value (the raw voltage) in L, and the filtered voltage in both A and H. For display, load DE with 83FF and call DWD2 (02D4). This will display the data at the right hand side of the display, leaving your last keyed in data displayed at the left.

FILTR and DWD2 take most of the time needed for the A/D conversion. If you should choose not to use them you must provide about a one millisecond delay by other means, such as a call to the monitor subroutine DELAY (0236).

6.2.2.5 Timer Operation

Timer 0 is used to define the total period. It operates in mode 2 so that RST5 interrupts occur at precisely repeated intervals according to the value keyed in with RUN, and this period is repeated until a new value is entered. Interrupt service for RST5 sets Port 1A7 high

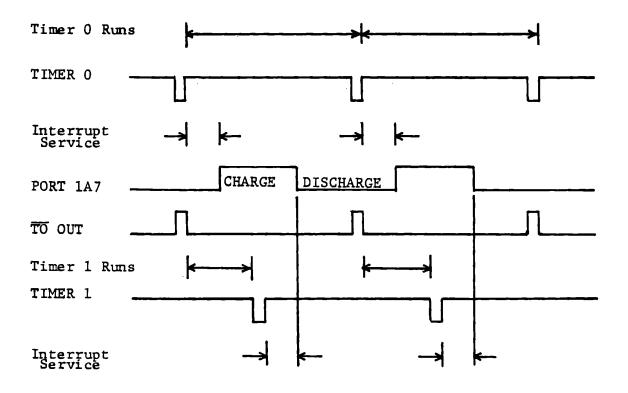
to start charging the capacitor. It also sets 1A5 low to clear the scope trigger.

Timer 1 controls the charging time. While it is counting, Port 1A7 will be high so that charging can occur. At its terminal count, RST6 interrupt occurs and Port 1A7 is set low to start discharging the capacitor.

Timer 1 is used in mode 5, "Hardware Triggered Strobe". This mode of the 8253 interval timer was briefly described in Chapter 3 but has not been used in any previous exercise. In this mode the timer can be preloaded with a count value, and when a rising edge signal occurs at its gate input it starts counting. The timer output is normally high; it goes low for one clock period at the zero count and rises again to create an interrupt one clock time later. The timer then waits for another rising edge at its gate input, obtained from Timer 0 output.

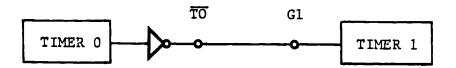
Figure 6-16 shows the timing relationship of Timer 0, Timer 1, and Port 1A7. Figure 6-17 shows the interrupt service routines. Note that both service routines should take the same length of time from the interrupt to the switching of Port 1A7.

Both mode 2 and mode 5 of the interval timer allow the timer to be laoded at any time. If the timer is counting, the present period is completed before the new time value is loaded. Therefore we can load these timers from the KYTIM subroutine without regard for their present states.



Timer 0 in Mode 2

Timer 1 in Mode 5, triggered by Timer 0



PWM Timer Operation
Figure 6-16

This page intentionally left blank.

RST 5 Interrupt - Timer 0

Set Port 1A7 high to start charging Set Port 1A6 low to clear scope trigger Reenable Timer O Interrupt

Exit

RST6 Interrupt - Timer 1

Set port 1A7 low to start discharging

Reenable Timer 1 Interrupt

Exit

Note Ports 1AO - 1A5 are not used in the system, so these may be set high or low as convenient.

PWM Interrupt Service
Figure 6-17

6.2.2.6 PWM Memory Allocation

You should develop a detailed flow diagram for the main program and write your own programs for MAIN, KYTIM, VOLTM, and the interrupt service routines. Subroutine FILTR was developed in Section 5.5. Remember to provide its initialization and to load (HL) with the required memory address. The following memory assignments are used in the given solution:

```
8200 - 8227
               Main - Initialize
8228 - 824F
                Interrupt Service
               Subroutine VOLTM
8250 - 826F
8270 - 82AF
               Subroutine FILTR
82B0 - 82CF
               Finish Initialization
82D0 - 82FF
               Main Loop
8100 - 815F
                Subroutine KYTIM
8160 - 817F
                Subroutine LDT1
8180 - 81FF
               Subroutine CLOSL and its local subroutines
```

Data memory assignments are:

83A0	value of N for FILTR (must be initialized to 1,2,3, or 4)
83A1,A2,A3	Used by FILTR
83A4,A5	Pulse width
83A6,A7	Desired voltage
83A8´	Measured voltage
8000	Data Log Address
8001 - 80FF	Data Log

6.2.2.7 Debugging

Check that you have provided all of the proper initialization by comparing your program with the solution given in Figure 6-18. Then step through all of the initialization procedure, including calls to special entries of KYTIM and the RST6 and RST5 programmed calls to interrupt service routines. (This is an added advantage to using those calls for enabling the interrupts, since it allows the monitor to operate through the service routine).

Since FILTR was developed in an earlier exercise it should need no debugging, except for checking that it has been loaded correctly.

To check the voltmeter subroutine (VOLTM) you should omit the RST5 and RST6 in the initialization procedure, so that interrupts will not be enabled. Connect a 10K resistor in parallel with the capacitor (from ANALOG IN to ground) to obtain a voltage within the A/D range. Enter a breakpoint at the start of the voltmeter subroutine, and press RUN. Step through the voltmeter section to test the program flow, also observing the A register when the interrupt status byte is read and when the A/D input is read.

KYTIM should be checked with RST5 and RST6 still omitted. Enter an RST4 before the RET that jumps to the processing module. Run the program in STEP mode. Press a command key, and after the RST4 command is executed step through the KYTIM process for that command. When all commands have been tested, remove the RST4, restore the RST5 and RST6 instructions, and run the program in AUTO mode.

6.2.2.8 Program Operation

Start the program with an initial value of 0400 for period and C8 for voltage (2.00 volts). Turn the OPTO SENSE pot fully to the left for no resistance (highest voltage) and observe the average voltage with the voltmeter and on the display. The A/D input value varies in the less significant bits, because it senses the voltage at random points in the charge, discharge cycle. You can observe any single measurement by pressing an undefined key (e.g., ADDR). While the key is held down the measurements are stopped. Do this repeatedly and observe the range of voltage.

The voltage measured will be less than the requested 2.00 volts because the total period (500 microseconds) is too long. Gradually reduce the total period by keying in values less than 0400, followed by RUN. You should be able to obtain an accurate output of 2.00 volts, or C8 in the hexadecimal display. (If the display and voltmeter do not agree, adjust the ANALOG IN pot to make the A/D measured voltage agree with the voltmeter). The total period needed will generally be less than the nominal value of 30E, principally because the supply voltage Vc at the terminal blocks will be less than 5.0 volts. Enter different voltage requests and record the resulting output.

TOTAL PERIOD

VOLTAGE REQUEST	RESULT

DECIMAL	HEX	VOLTMETER	A/D
2.40	FO		
2.00	C8		
1.50	96		
1.00	64		

Find the lowest voltage request that reduces the output voltage. There is a lower limit to the charging pulse width, set by the time taken by Timer 0 interrupt service. Lower voltages can only be obtained by extending the total period. Now request 2.00 volts again (C8) and alter the OPTO SENSE pot setting to reduce the output voltage to 1.50 volts (observed as 96 hex). Reduce the total period (entering values with the RUN key) to raise the output to 2.00 volts. With this setting again record the results for different voltage requests, and find the lowest value that can be achieved.

TOTAL PERIOD

VOLTAGE REQUEST RESULT

DECIMAL	HEX	VOLTMETER	A/D
2.40	FO		
2.00	C8		
1.50	96		
1.00	64		

The departure from linearity should be obvious. Any request lower than C8 will produce too low an output, and any request greater than C8 will produce too high an output. When we close the loop in Section 6.2.3 we will overcome this sensitivity to external conditions.

PWM VOLTAGE CONTROL - INITIALIZE

	A D D	R	COL	DE												
	820	0	3	E		Μ	V	I		A		8	2			Program 8255#/
		1	8	2								_	_			Port 1B in you A/D
		2	\mathcal{D}	3		0	\mathcal{U}	T		C	\mathcal{U}	7	/			
_		3	0	7												
IEET		4	3	E		M	V	工		A		9	a			Program 8255#2
3 SH	-	5	9	2												0
CODING		6	D	3		0	U	\mathcal{T}		C	N	7	2			
Ö		7	0	F												
		8	3	E		M	٧	I		A		3	4			Trogram Timer O
		9	3	4												Both lustes
		A	D	3		0	U	7		7	I	M	C	7		Mode 2
		В	/	7				-								Binaren
	· · · · · ·	С	3	E		M	V	エ		A	,	7	A			Program Timer 1
STEM		D	7	A							_	-				Both bytes
/ST		E	D	3		0	U	T		7	T	Μ	C	T		Mode 5
3 SY		F	1	7			_									Binary
TRAINING	821	0 .	3	E		M	V	I		A		9	4			Program Time 2
AIP		1	9	4							•					Low byte
		2	D	3		0	U	T		\mathcal{T}	\mathcal{I}	Μ	C	7		Mode 2
TER		3	/	7												Binary
MICROCOMPUT		4	3	E		M	ν	I		A		0	8			Load Timer 2
SO		5	0	8												for +8
RÕ		6	D	3		0	V	7		7	I	Μ	2			
N C		7	/	6												
		8	2	/		L	X	I		14		8	3	A	0	Initialize memory
		9	A	0							,					SOV FILTR
		A	8	3												7
SYSTEMS		В	3	6		M	V	ユ		M	,	0	4			n = 4
STI,		С	0	4							-		-			
	·	D	A			X	R	A		A						Clear 2"E:
TEF		E	2	1		I	N			H						Clear 2ª E; address 83A/
COMPUTER		F	7	7		M	0	V		M	Ţ,	A				
SON	3 22	0	2	3		I	N	X		H	,	,				address 83A2
		1	7	7		M	0	V		M		A				
ATE		2	3	C]	I	N	R		A						Set Port 1CO
NTEGRATED		3	2			0	U	T		P	0	R	T	1	C	Sa 2 automatia
NTE		4	0	6	<u> </u>	Ĭ					_		Ė	Ė		A/D in Out
Ξ		5	C	+		7.	M	P	<u> </u>	8	2	B	0			Jump post
	,	6	B			Ĭ		İ		_	-		_			interrupt service
		7	8	2					İ							A/D input Jump past interrupt services and FILTR
		8	<u> </u>						1							Figure 6-18a
			•			_		•	• -							C 71

A D D R CODE PWM VOLTAGE CONTROL

AD	D R	co	DE			_									
8	0				I	N	7	E	R	R	U	9	7		SERVICE
	1														
	2	1							-		-			_	
	3	+-			<u> </u>									-	
-	4	+			-	-	-							-	
	5					 	<u> </u>			L				 -	
	6	+			-	\vdash	-		-				_	-	
	7	+-			-	_					_	-			
		+-	4	-	1	,,	-	11	_	ก	-	4.3		-	257 7 3
82		1	5		2	V	<u>ら</u> エ	H		P	2	W			RST5-Jenev O
<u> </u>	9	3	E		M	V	1		A		8	0			Set Port 1A7 high
	A	8	0		_	<u> </u>	_	-			_	_		_	to start charging all other lits low
	В	2	3		0	U	\mathcal{T}		P	0	R	7	/	A	all other lits low
	С	0	4		_	<u> </u>	L								
	D	C	3		2	M	p		8	2	4	0			Jump past RST6
	E	4	0		Ĺ.					L .					0
	F	8	2												
8 2	30	F	5		\mathcal{P}	V	<u>S</u>	14		P	S	W			RST6 - Times/
8 2	1	3	E		M		I		A		0	0			Set Post 1A7 low
	2	0	0				_			- / -					to start discharge
	<u>.</u>	D	3		0	V	\mathcal{T}		P	0	R	τ	1	A	(all other lits also)
	4	10	1/		1		-				/(<u> </u>	/	 -	Note identical times
<u> </u>	5	3	E		M	V	7		A		0	3		-	as in RST5
	6	10	3		1/4/	<u> </u>	<u> </u>	-	//)				-	as in 165/6
<u> </u>	7	D	<u>ე</u>			1, 1	T		C	11	7	2		 	10 10 10 1
		_	٦		0	V		<u> </u>	_	N	-	~		-	Reenable and clear
	8	0	1		1	_	0		P	_	(1)	_		-	Times interrupt
 	9	15	/		P	_	P	-	1	5	W			-	Exit
<u>.</u>	A	F	B		E	1	_		-	-			_		
<u> </u>	В	C	9		K	E	7								
	С	0	0		IV	0	12						_		
	D	0	0		N	0	P							ļ	
<u> </u>	E	0	0		N	0	ρ					ļ		_	
<u></u>	F	0	0		N	0	ρ						L		
3 2	40	3	E		M	V	I		A		0	1		<u>_</u>	Reenable and clear
3	1	0	1							,					Lenable and clear Liner Ointerrupt
	2	D	3		0	U	7		C	W	7	2			
	3	0	F	· · ·	Ĭ					Ė					
;	4	E	1		P	0	P		P	7	W	-	-	1	Exit
<u> </u>		F	B		/=	T	'	-	' -	_			-	\vdash	- Cycl
	6	C	9	-	R	E	T		-	-		 	-	 	
<u> </u>	7	10	7	<u> </u>	$+^{\prime}$	<u> </u>	 	-	-			-	-	 -	
<u> </u>		+	-		-		-	-	-			 		-	Figure 6-18b
L	8		L	<u> </u>	<u>L</u>		<u> </u>	<u> </u>		<u> </u>		<u></u>	L		trante o-lop

	A D D R	co	DE	PWM		٥٥	סתנ	JUT.	TNE	, V	נידט	IAI	M T.7	ın	LOG
	8 25 0	E	5		P	U	٤	H		H					Save (HL)
	1	2	/		上	X	I		14	_	8	0	0	0	location for data log address
	2	0	0	*						_					location for
ET	3	8	0	*											data log address
ш	4	3	4		エ	N			M						Accepted address
3 SH	5	C	2		2	Ν	7		8	<u>a</u>	બ	9			but not beyond FF
CODING	6	3	9												
00	7	8	2												
	8	3	5		\mathcal{D}	C	R		M			_			Restore FF
	8259	6	E		M	0	V				Μ				(HL) - data log address
	Α	3	6		M	V	I		M	<i>,</i>	F	F			Stre FF in case
	В	F	F										ļ		A/D not ready
_	С	2	\mathcal{B}		工	N			P	0	R	T	2	B	Read interrupt
SYSTEM	D	0	D		L									_	status byte
ΥS	E	E	6		A	N	I		0	8					mark for
	F	0	8												A/D comparata
RAINING	8 260	C	A		2	て			8	2	6	\mathcal{D}			Jump if
RAI	1	6	D	<u> </u>		ļ									Jump if A/D not ready
H.	2	8	2		L								_		
JTE	3	12	\mathcal{B}		I	Ν			P	0	R	T	1	\mathcal{B}	Read A/D voltage
WICROCOMPUT	4	0	5						_		-		_	<u> </u>	
00	5	3	2		8	T	A		8	3	A	A	_		Store voltage
SHO	6	A	A			<u> </u>					_	_			in fixed location
ž	7	8	3		ļ.,	ļ		ļ			_	_			and log in
	8	12	7		M	0	V		Ŵ	ļ.,_	A			_	variable location
	9	3	E		M	V	I		A	,	0	6	_		Reset A/D Counter
S	Α	10	6		<u> </u>			ļ;		_		_	_		Spare interrupt disabled
INTEGRATED COMPUTER SYSTEMS	В	2	3		0	U	T		C	N	T	J			disabled
YS	C	0	F				ļ.,	<u> </u>	_		<u> </u>		-	-	
8	8260	7	E		M	0	V	<u> </u>	A	ļ, -	M	-	ļ		(A) - voltage
1	E	E	/		Ď	0	P		H		_	<u> </u>	_	_	Restore (HL)
Μ	F	C	9		R	E	7	<u> </u>		_	-	_	-	_	
8	3 0	↓_							_	Ļ		<u> </u>	_		
TED	1	+	-	*	1	8		0	0	را	-	6	0	1	TAINS
ΡĀ	2	4	ļ			<u> </u>	D	A	1	A	<u> </u>	Ļ	0	Ģ	
ĒĞ	3	1	<u> </u>	.	IR	E	T		R	N		(A	T	= VOLTAGE
Z		1	ļ		<u> </u>	_	Z	F	R	0	-	F	L	A	
	5	╁_	-		ユ	F	ļ.,			$\overline{}$	 	1	$\overline{}$	Γ	READY
	6	+	↓		_	-	μ	A)	=	F	F	<u> </u>	_	ZERO FLAGSET
	7	╁	-		-	_	_		<u> </u>	_	_	 	\vdash		The same of the
	8		<u> </u>	<u> </u>		<u></u>	L.	<u> </u>	<u> </u>		<u> </u>	<u>L</u>		<u> </u>	Figure 6-18c

	A D D R	COL	Ε	SUB	ROU	TI.	NE	FI	LTF	t					
	8 270	D	5		\mathcal{P}	ν	5	H		D					Save (DE)
	1	C	5		ρ	\mathcal{U}	2	H		B					Save (BC)
	2	4	6		M	0	V		B		Μ				(B)←n
_	3	4	8		M	0	V		C		B				(C) = ~
SHĒET	4	2	3		ユ	N	X 2		H						address 2 Ei-1
	5	E	5		ρ	V	2	1	•	H					Save address
CODING	6		E		M	D	V		E	,	M				1
100	7	2	3		工	N	X		\mathcal{H}	_					(DE) = 2 " Ei-1
	8	5	6		Μ	0	V		\mathcal{D}		Μ)
	9		\mathcal{S}	-	M	0	V		L		ED				(HL) - 2 m Ei-1
	A		2	_	M	0	V		H	<i>)</i> _	D			ļ	
	827B		9		D	A	\mathcal{D}		H						
V	С		D		D	C	R	_	<u>C</u>						
SYSTEM	D		2		J	N	Z		8	2	7	\mathcal{B}			7(HL) - 22 Ei-1
SYS	E	11	\mathcal{B}		_					_		_			
	F	8	2			- ()	. ,				Δ	_			
TRAINING	8 28 0	4			M			-	2		A				(C) - Vi
-RA	1	7	\mathcal{D}		M		7		A	-	<u>L</u>				
	2	9	3		S		B		E		1				(DE) (HL) - (DE) = 2 ~ (2 ~ 1) E;-1
MICROCOMPUTER	3	0			M	0	V	_	E	-	A				$=2^{-1}(2^{-1})E_{i-1}$
MP	4	1/	C		M	0	V		A		\mathcal{H}				
00	5	9	H		<u>ک</u>	B	\mathcal{B}		D						
CR	7	5	7	_	Μ	0	V	_	-	1	A	_		-	
≥		6	9	-	M	0	V	_	<u> </u>) -	C	~	-		
1	8	2	6		Μ	V	\mathcal{I}		Н	,	0	U			(HL) - Vi
	· A		D		0	A	,	1		S	11	_		A /	10 · · · 1
MS	В	0			C	77	_	_		J	M	F	1	/V	Divide by 2m (DE) = (2m1) Ei-1
SYSTEMS	<u>с</u>	8	00					_			_				(DE)= (& -1) Li-1
SYS	D	E	B		7	\mathcal{C}	H	G		_					<u> </u>
ËR	E	\rightarrow	9		^	A	D	3	D						((DE) - V:+(2m) E.
COMPUTER	F	E	B		X	\mathcal{C}	<u> </u>	G	1						(DE) = Vi+(2~1)E:-1 = 2 m Ei
∑ 0	3 0	1			\ \			7	.T	N	U	£	,		AT 8290
	1	1 -				0	~	1	£	~	U,		ם		1 12/0
ATE	2	1 1			E	1 /	$\overline{\tau}$	Ε	R		1	A	1	=	Vi (new value)
INTEGRATED	3	1 1			7	/	\mathcal{H}	7)	7		7 1		=	n (filterconstant)
N N	4	1			1	1	H	۷	1	+	1)		7	2m/t
=	5				7	(#	1	Ď	+	2	\int		3	
	6				$\tilde{\mathcal{R}}$	È	7	V	R	N					
	7				1	(Н	L)	+	3)	=	(A)=(H)=Ei
	8														(L) = Vi

6-74

Figure 6-18d

	A D D	R	COL	DE	FIL.	ΙK	· ·	COI	161	nue	:a)	ar ——	ıa	oni	TN	
	8 2 9	0	E	3		χ	7	/+	L							(HL) + address 2 "Ei
		1	7	3		M	0	V		M		E)
	1	2	2	3		I	N	X		1+	,					Stone 2 nE
		3	7	2		M	0	V		M		0		<u> </u>		1
EET	-	4	2	3		I	\mathcal{N}	X		Н	,					address E
SHI		5	$\overline{}$	\mathcal{B}		$\overline{\mathcal{D}}$	(X		P						
NG		6		2		C	A	,	L		S	4	F	7	1/	Fractional part > 12
CODING		7	A	7		L	-		_)	//	,	-	/ -	fractional face 12
Ö		8	8	2										\vdash		
		9	7	77		Μ	0	V		М		A				Store Ei
		A	É			$\frac{\mathcal{P}}{\mathcal{P}}$	0	$\overline{\rho}$	-	H		' '				(L) ← Vi
		В	6	7		M	0	1/		//		A		ļ		(1)
		C	9			ויון D	0	P		\mathcal{B}	1	//		 		(H)←E; Restore B, C, D, E
Σ		D	2	$\overline{\mathcal{I}}$		P	0	P		D	ļ	-	_	-	 	restore B, E, D, E
SYSTEM		E				_	E	•				<u> </u>		-	<u> </u>	0 .: 4
S≺S	<u> </u>	F	2	9		R		TP	-		 			├-		Exit
92	0.00		0	0		N M	0	<u> </u>		1	<u> </u>	0	- :	-	-	
RAINING	8 2 A	0	4 A	8		_	0	V		CA	,_	B		 	-	SHFTN (c) -N
ΓRΑ		1		F		X	R	A	<u> </u>					 	 	Loop - cless carry
ER 1	-	2	7	A		M	0	V		17		\mathcal{D}		ļ		·
⊢		3	1	F	-	R	A	R	<u> </u>	-	<u> </u>			ļ		Shift (DE) right
MICROCOMPU		4	5	7		M	0	V	_	\mathcal{D}		A		├		Shift (DE) right to divide by 2
8		5	7	\mathcal{B}		M	0	V		A	 ,	E	_	 	ļ	
CRC		6	/	F	-	R	A	R		L_	<u> </u>			<u> </u>	ļ	
ž		7	5	F		M	0	V		E	,	A			<u> </u>	
		8	0	D		\mathcal{D}	C	R		C		_		<u> </u>	ļ .	Loop on times to divide by 2"
		9	C	2		7	N	Z		8	2	A	/			to divide by 2"
S		Α	A	/	_				_		ļ	_		<u> </u>	ļ	<i>U</i> .
ĒΖ		В	8	2						ļ			ļ			
YST		С	D	0		R	N	C	ļ	<u> </u>				<u> </u>		Exit if LSB=0 Else round up (A) = less significant lyte
R S		D	/	3		I	\mathcal{N}	X		D						Else round up
F		E	7	\mathcal{B}		Μ	0	ν		A		E				(A) - less
MPL	<u>. </u>	F	C	9		R	E	7							<u> </u>	significant legte
INTEGRATED COMPUTER SYSTEMS	3	0										<u> </u>				
Ē		1														
ATI		2								[_			
EG.R		3														
N		4`														
=		5				İ				1						
		6	Π		<u> </u>											
		7									1 -				1	
		8			, ·											Figure 6-18e
								щ.				-			4	· · · · · · · · · · · · · · · · · · ·

	A D D	R	co	DΕ	PWM	VC	LT.	AGI	E –	SE	T	INI	TI.	AL	VA	LUES
	8 2B	0	/	/		ム	X	I		D	/	0	4	0	0	Let total period
		1	0	0							7					Let total period to 500 u sec
		2	0	4												
⊢		3	C	D		C	A	L	L		R	U	Ŋ			Lew key Sunction of
SHEET		4	3	8												KNTIM Sonds Times O
		5	8	/												for total period
CODING		6	/	/		L	X	I		D	,	0	0	C	8	for total period Let voltage to 2.00 volts
00		7	0	8							_					to 2.00 volts
		8	0	0												
		9	C	D		C	A	4	L		N	E	X	T		Rest key Sunction of
		Α	2	0												KYTIM Calculatta
		В	8	/		_									_	KYTIM Calculation of width, loads Times!
5	-	C	E	F		R	2	7		5						
SYSTEM		D	F	1		R				6			7	2	_	
SYS	<u> </u>	E	C	3		2	Μ	ρ			8	2	ν	0		
	- 0.0	F	D	0												
TRAINING	8 2 C		8	2			<u></u>	_							_	
ΓRΑ		2	-												-	
			-					-								
ΤŲ		4		-	-	-		-							-	
JMC		5	<u> </u>			-								ļ		
ÖC	<u> </u>	6				-							_		-	
MICROCOMPUTER		7	ļ —												_	
≥.		8	-			-									-	
		9	 												 	
		A						_		<u> </u>	-			_		
SYSTEMS		В									_					
'STE		С														
		D														
COMPUTER	· -	E														
MPU		F														
Ö	3	0	<u> </u>												ļ	
-ED		1	<u> </u>			_			_							
RAT		2	igspace			_				_					_	
NTEGRATED		3					_	_		<u> </u>	·		_	_	_	
Z		4	igapha	_			-		_	<u> </u>					-	
		5	\vdash	<u> </u>		-		-	-	-			-		-	
		7	┼-	_		_				<u> </u>			-	-	_	
1				-		-		_		ļ				_	-	Figure 6-18f
	1	8	1 .	1	I	L	Γ,	<u> </u>		L						TIGULE OF TOT

	A D D	R	_ CO1	DE	PWM	vo	LT	AGI	c C	гио	RO	L -	- M	IIA	1 L	ООР
	8 2 D	0	D	B	-	工	Ν			P	0	R	τ	\bigcirc	A	Test keyboard
		1	0	0												(FF if no key)
		2	3	\mathcal{C}		H	7	R		A						
⊢		3	C	4		C	N	7			K	У	7	I	Μ	Call for keyboard
SHEET		4	0	0												entry and process
		5	8	/												· ·
CODING		6	C	\mathcal{D}		C	A	L	L		V	0	L	T	7	Measure voltage (A) - A/D voltage
00		7	5	0		<u> </u>										(A) - AD voltage
		8	8	2												(FF if not ready)
		9	0	0		N	0	ρ								0
		Α	0	0												Save & lytes
		В	0	0												for closed loop
_		С	0	0												control.
LEN		D	0	0												
SYSTEM		E	0	0												
		F	0	0			Ц.									
NING	8 2 E	0	0	0			↓					<u> </u>	ļ	ļ	ļ	
RAI	82E	1	2	1		4	X	I		14	٠,	8	3	A	0	Themory address
H H		2	A	0		<u> </u>				1				_	_	Nemory address
Ę		3	8	3		ļ										
MICROCOMPU		4	C	Þ		C	A	4	L		F	I	<u>L</u>	7	R	
000		5	7	0										_	ļ	(L) = raw voltage
SRC		6	8	2				<u> </u>				ļ		_		(H) & filtered voltage
ž		7	/	_		<u>L</u>	X	I		D		8	3	F	F	Display raw voltage
		8	F	F		ļ					Ĺ			_		(at right)
		9	8	3		ļ		ļ	<u> </u>					_		and filtered reoltage
ώ		Α	C	D		C	A	7	L		D	W	\mathcal{D}	2		
Ē		В	D	4		ļ						_				
ΥS		С	0	2		ļ		_		<u>. </u>					_	
:R S		D	C	3		7	M	P			8	2	\mathcal{D}	0	_	Loop
UTE		<u>E</u>	2	0		ļ				_		<u> </u>				
INTEGRATED COMPUTER SYSTEMS	<u> </u>	F	8	2		_										
8	3	0				_		_	<u> </u>						ļ	
臣	ļ	1	ļ			_	<u> </u>					_	_	ļ	ļ	
RA.		2	_	<u> </u>	 		<u> </u>	<u> </u>		ļ.,	-				-	
reg		3			ļ 	_	<u> </u>	_		_		_			ļ	
Ξ		4	<u> </u>		<u> </u>				_	<u> </u>	_				<u> </u>	
		5		_		-					ļ	<u> </u>		<u> </u>		_
		6	<u> </u>	ļ	ļ	 	_	<u> </u>	_		_		_	ļ		
		7							_	ļ		<u> </u>		<u> </u>	-	7.10
		8.	<u>L</u>	<u> </u>	<u> </u>			L_	ļ		L_	<u></u>	L_{-}			Figure 6-18g

	A D D	R	со	DE	PWM	<u>-</u> 8	SUB	ROI	JTI	NE	KY	TIN	1			
	8 10	0	0	0		\mathcal{N}	0	ρ								for DI during debug
		1	C	D		\bigcirc	A	L	L		E	Ν	7	W	P	(A) - Command
		2	4	6												(1+L) = total period
⊢		3	0	3												(I+L) = total period ov voltage (DE) = data
tEET		4	E	B		X	C	/+	ტ							(DE) La data
3 SH		5	2	/		L	X	I		14		8	/	0	8	address dispatch
NI		6	0	8	 						,					table -10H
CODING		7	8	/												
)		8	8	5		7	\mathcal{D}	D		L						add command
		9	6	F		M	0	V		L		A				
		A	6	E		M	0			L	,	M			_	(5T) - dispatch address
		В	E	5		P	U	S	H		14					
		С	F	3		D	エ)
EM.		D	D	\mathcal{B}		I	N			P	0	R	1	1	A	\ .
SYSTEM		E	0	4												Set Port 1A6 high
		F	F	6		0	R	I		4	0					for rease trigger
Z	8 //	0	4	0												without changing
RAINING		1	F	B		E	I									for seope trigger without changing Port 1A7.
RITE		2	D	3		0	U	T		P	0	R	T	/	A	
ш		3	0	4)
₽U		4	A	F	_	X	R	A		A						Clear A and CY
SO		5	0	0		N	0	D								
MICROCOMPUT		6	0	0		N	0	P								During Deling RST4
MIC		7	C	9		R	E	T								or BKPT here
_	811	8	/	7		M	Ш	Μ								Underlined
		9	/	7		$ \alpha $	E	G						-		Underened
۲۵		A	/	7		A	D		R							Triager scope
SYSTEMS		В	2	/		S	$\overline{\tau}$	E	P							Store desired voltage
/ST		С	3	8		R	U	N								Set total seriod
		D	2	0		Ν	E	X	$\bar{\tau}$							Undefined Trigger scope Store desired voltage Set total period Store voltage, set width Undefined Undefined
TEF		E	/	7	_	B	R	K								Underined
NTEGRATED COMPUTER		F .	/	7		C	Ĺ	R								Undersine d
SON	8	0		_												
ED (1														
ATI		2														
EG.		3														
NT		4														
_		5														
		6														
		7														
		8														Figure 6-18h

8 / 2 0 3 7 STC NEXT (markinist CV) 1 2 / L XI H 8 0 0 0 SEP (CV class) 2 0 0		A D D	R	cor	DE	KYT	IM	- :	NEX	ΥТ,	Sī	ΈP	, F	RUN			
2	ſ		- T	\neg			S	7	C								NEXT (markswith CY)
South Sout			1	\neg	/		L		I]4		8	0	0	Q	
			2	0	0												
	_[3	8	0												address with its
6 22 SHLD 83H6 Structure woltage 7 A6 8 83 8/29 DO RNC A // LXI D, FFEE Josephant 12H BEE (0.18 wolt) from CEF DAD H CONTROLL FOO NOP 1 AH 3 0 22 SHLD 83A 4 Structure woltage (HL) - v - Vg (HL) - v - Vg (HL) - pulse wolt) 1 AH 2 83 CONTROLL 3 0 D CALL LDT / Lond Line / with Calculated pulse width 1 AH 2 83 CONTROLL 3 0 D CALL LDT / Lond Line / with Calculated pulse width 7 00 NOP 8 /3 8 7B MOV A E RUN A / H B 7 A MOV A D SON TOTAL FOO NOP FOO NOP FOO NOP 1 H FOO NOP FOO NOP 1 H FOO NOP 1 H FOO NOP 1 H FOO NOP 1 H FOO NOP 1 H FOO NOP 1 H FOO NOP 1 H FOO NOP 1 H FOO NOP 1 H FOO NOP FOO NOP 1 H FOO NOP FOO NOTE FOO NOP FOO NOTE FO	빌		4	7	5		M	0			M		L				own location
R			5	E	\mathcal{B}		X	C	H	G							1/ 1
S	Ĭ		6	2	2		S	H	L	\mathcal{D}		8	3	A	6		Store desired voltage
S	힝	· · ·	7														
A / / L X I D , F F E E to subtract 12H B E F C F F D / 9 D AD D C HL) - v - lg C HL - v - lg E 29 D AD H (HL) - pulse width 1 A H 2 8 3 CD C A L L D T / Lord Line / with calculated 2 8 3 C D R E T 7 00 NOP 9 D 3 0 U T T I M O 8 7 A / H B TA MOV A D F OO NOP F OO NOP P OO NOP 1 A / H 1 A			8		3												
B E E		8/2	9	D	0		R		C							ļ	Exit if STEP
B E E	Į		Α	/	/		上	X	I		D		F	F	E	E	To subtract 12H
C F			В	E	E												(O. 18 volts) from
	_		С	F	E												desired voltage
SASSESSE S N D P S 3 A 4 Store Calculated Pular Unidth Pu			D	/			_				-						(HL) = v - Vg
SASSESSE S N D P S 3 A 4 Store Calculated Pular Unidth Pu	.XS		E	2	9				_		H			_	ļ	-	(HL) = pulse width
3 CD CALL LDT / Soad Timen / with calculated 4 60				0			N		ρ			_	_		ļ.,		
3 CD CALL LDT / Soad Timen / with calculated 4 60	Ē	8 /3			2		3	H	L	\mathcal{D}		8	3	P	4	<u> </u>	Stone calculated
3 CD CALL LDT / Soad Timen / with calculated 4 60	ΗĀ				4									_	_		pulse width
MODOWN	<u>د</u>		2				_	_		<u> </u>		_			_	<u>_</u>	
S	빍			0	2		C	A	L	L		<u>L</u>	D	7	/		F, , —— 7
S	MP			0	0				_				_	_	_		
S	ပ္က				/			_	_	_		ļ		<u> </u>		<u> </u>	pulse wist
S	CR(_	\vdash	7			上	I						-	<u> </u>	
9 D 3 O U T T I M O Load Liner O for total period B TA MOV A D C D 3 O U T T I MO E C 9 R E T F O O MOP 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Ξ	<i>a</i>			0			0	1		_		_	_		<u> </u>	71
A		813	_	_	\vdash			0	V	_	<i> </i> 	<u> </u>	E		-	<u> </u>	
B 7/A MOV /4 D / D / D / D / D / D / D / D / D /				0	- ,	<u> </u>	0	0	/		/	1	M	0			
5 6 7	AS.	<u> </u>	_	1	_	<u> </u>	4.4	_			0	-	7			-	for total period
5 6 7	T.		_	1/2		·			 		-	4	_	_		├-	/
5 6 7	SYS			1	,	<u> </u>	0	0	/	-	/	1	/*1	0	-	\vdash	
5 6 7	ER			/			n	_	7	-	-	-		-			
5 6 7	Ž			-	_		\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	5	-			-	-		-		
5 6 7	JMC	а		0			עון		ν -			-	-		\vdash	\vdash	
5 6 7	Ö	-		-	-		-					-			<u> </u>	 	
5 6 7	λΤΕ		_	\vdash	\vdash		 								-	\vdash	
5 6 7	GR/		_	\vdash		 	-	-	+-	-	-	-			\vdash	-	
5 6 7	JTE		_	-	-	 	\vdash	 		-		-	-		-	+	
6 7	2			\vdash	-	 	\vdash		-	-		-	\vdash	+-	+-	\vdash	
7						<u> </u>	\vdash	\vdash	\vdash	-		-	\vdash	 			
				-	-	_		\vdash		-		_	 	 	\vdash	\vdash	
			8					-				-	\vdash				Figure 6-18i

	A D D	R	со	DE			LDT			LC	AD	TI	ME	R 1	l	
	8 16	0	2			D	C 0 R 2 X	X		\mathcal{H}						LDT/ Lest pulse width for zero or negative Exit if \(\beta \) Load Lines / with charging pulse width
		1	7	BC		M	0	$\overline{\nu}$		A		Н				Last pulses width
		2	B	7		0	R	A		HAA	,					Hard acra mi
ᅵ		3	2	3		エ	Ν	X		14						meastine
SHEET		4	2 1	38		エR	Μ	I	N	U	5		_			Exit it = 0
		5	7	D		M	0	V	·	A		L				Lord Temer /
CODING		6	2	3		0	U	7		7	エ	Μ	7			with charging
릵		7	/	کی												pulse width
		8	7	C		M	0	V		A		Н				
		9	D	3		0	U	\mathcal{T}		7	立	М	1			
		Α	/	S												
		В	C	9		R	E	7								
		С														
SYSTEM		D														
\ S		E														
		F														
TRAINING	8	0	Щ													
RAI		1	<u> </u>				_									
		2														
픠		3														
MICROCOMPUTER		4														
000		5														
SRC		6				_										
Ž		7													ļ	
		8				_					_					
		9				_										
S		Α				_	_	-	-							
SYSTEMS		В														
YS	<u> </u>	С	<u> </u>				_	_		-			_	-		
		D	<u> </u>			ļ					_					
COMPUTER		E	_	<u> -</u>		<u> </u>										
MP		F				_	-	_		-			_			
ပ္ပ	3	0	-	_		_				_				-	<u> </u>	
TEC		1		-	<u> </u>	-	-	_	ı	<u> </u>	-	_	_		-	
ЯĄ		2	-			-	<u> </u>	_	_		_					
INTEGRATED		3	-	-		-		-		-	_	-		-	-	
Ż		4	-	\vdash		-				_		<u> </u>		-	-	
		6	\vdash	-		-		-	\vdash	-	_			-	-	
		7	+-	-		\vdash		-		\vdash	-	<u> </u>			-	
			-	-		-	\vdash	_	-	\vdash		-	-		-	Figure 6-18j
		8	1		<u> </u>	<u> </u>	<u> </u>		L			L			L	TAUTE 0-10]

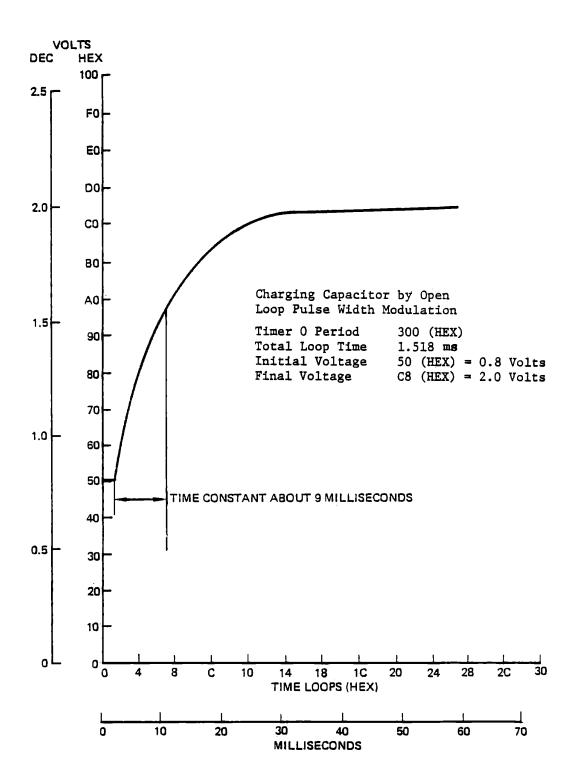
This page inentionally left blank

6.2.3 Observing Response Time

If an oscilloscope is available, observe the response of the capacitor voltage to a new input. Trigger the oscilloscope from Port 1A6, which is set high when a key is pressed and set low when the next interrupt occurs. If you do not have an oscilloscope you can observe the response by reviewing the data log created by VOLTM. Recall that whenever a new voltage request is made through KYTIM we start a new log, storing the voltage each time it is measured at the next available location in memory area 8001 - 80FF.

To use the log, set the desired total period (with RUN), enter a voltage (with NEXT) and then enter another voltage (with NEXT). Now press RST and review the data stored at 8000 - 80FF by entering ADDR 8000 and NEXT. Figure 6-19 is a plot of such data with a total period of 300 (hex), initial voltage 50 (hex) and a final voltage of C8. The curve shows the exponential charging of the capacitor toward 2.0 volts with an effective time constant of about 6 loop times or about 9 milliseconds.

A voltage has been recorded for each repetition of the main loop. These are not exactly equal intervals of time, principally because the number of interrupts during the main loop varies. With a Timer 0 period of 300 (hex) there will usually be four interrupts from each timer during the main loop, and occasionally one more, giving a total loop time of 1.49 to 1.59 milliseconds, and an average of 1.518. This value was used for the millisecond time scale in Figure 6-19.



PWM - Open Loop Response
Figure 6-19

The number of interrupts and the total loop time can be calculated from:

$$n = \frac{t_{\hat{k}}}{t_p - t_i}$$

$$t_t + nt_p$$

where n = number of interrupts per loop

t time for one pass through the main loop
and subroutines with no interrupts

For the author's solution the values are given below.

$$t_{i} = 2092.3 \quad \text{clocks}$$

$$t_{i} = 251.1 \quad \text{clocks}$$

$$t_{p} = 768 \quad \text{clocks}$$

$$n = 4.048 \quad \text{average interrupts/loop}$$

$$t_{t} = 3108.7 \quad \text{clocks per loop}$$

$$= 1.518 \quad \text{milliseconds per loop}$$

6.2.4 Closing the Loop

EXERCISE:

preceding section we can generate a PWM With the program of the voltage whose value is predictable if the circuit conditions are known. Open loop control is satisfactory in such a case Changing the SENSE pot setting alters the resulting voltage and introduces an error, which can be corrected in either of two ways. We can change the mathematical model that relates pulse width to voltage, or we can simply adjust the pulse width to achieve the desired value. Note that in this system the total period (nominally 030E) represents the "mathematical model"; a correction to this value can approximate the intended relationship of two counts of pulse width equal to one count of voltage, although it cannot remove the non-linearity. Alternately, we can adjust the pulse width to some value different Either of these methods can be than twice the desired voltage. applied by a computer program in response to an observed difference between the measured voltage and the desired voltage. We used the first method by manual entry of new periods in the preceding exercise. Now we will apply the second method automatically.

6.2.4.1 Error Signal Calculation

The error signal is the difference between the desired value and the measured value of the controlled variable. We obtain a measured value by reading the A/D input. Subroutine KYTIM has stored the desired value in memory at 83A6. (Only single byte values are meaningful now). The error signal is the desired value minus the measured value, which is positive when the measured value is too low, negative when it is too high. To adjust the driving force to correct the output we will add a positive error signal to the present charging pulse width, or subtract the magnitude of a negative error.

It turns out to be more convenient to subtract the desired voltage from the measured voltage, giving the complement of the error signal. Moreover, this seems to be a more meaningful value to display, being positive when the output is too high. Then we will take its twos complement for the correctly signed error signal to be added to the pulse width.

The error signal could range from -FF to +FF if the full voltage range of the A/D converter were available. Actually the range is somewhat less, but it is certainly greater than an eight bit value. The subtraction of measured voltage minus desired voltage gives a nine bit result in A and CY, with CY representing the sign. We will display only the eight bit value, since most of the time the sign will be obvious. For the calculation, however, we will convert it to its two byte twos complement.

This can be done by:

CMA complement magnitude

MOV C, A (C) <--- magnitude byte

CMC complement sign

SBB A (A) <--- 00 or FF

MOV B, A (B) <--- sign byte

INX B increment for two's complement

Now the properly signed error signal can be added to the pulse width (loaded into HL) by DAD B, giving a new pulse width.

When no error exists the pulse width will be constant; a positive error will increase the width and therefore the voltage; a negative error will decrease the width. Since the error signal is added into the steady state force, we have <u>integral</u> control. In Section 6.2.7 we will discuss the relationship between this simple control system and the integral control equation. First, however, we will develop subroutine CLOSL to perform the calculation and control the pulse width, and we will observe the results. CLOSL is to be located at 8180 - 81BF, and will be called after the return from VOLTM with (A) = measured voltage. You should now complete the main loop according to Figure 6-12. Note that CLOSL is to return data in (HL) for display by DWORD. FILTR needs the voltage returned by VOLTM, so the main loop saves that value by PUSH PSW before the call to CLOSL and recovers it before calling FILTR.

CLOSL is specified in Section 6.2.4.2 and shown in Figure 6-20. CLOSL itself calculates the error signal and loads the old pulse width. It calls another subroutine, INTEG (located at 81CO), to calculate the new pulse width. Then CLOSL calls LDT1 to load Timer 1, provided that the pulse width is positive and greater than zero.

INTEG calculates the new pulse width, in this version, simply by adding the error signal to the old width. It tests for a negative result and stores the pulse width (which is the integral of errors) if it is positive. (A zero integral is permitted, although zero is forbidden to be loaded to the timer).

The specification for INTEG in Section 6.2.4.3, Figure 6-21, states requirements that are to be met by both versions of INTEG that will be developed.

.

This page intentionally left blank

6.2.4.2 Subroutine CLOSL Specification (Call at 8180)

Function Calculate the error signal and set a new pulse width

Return negative error signal and desired voltage

ready for display by DWORD.

Enter (A) = Measured Voltage

(83A6) = Desired Voltage

(83A4,A5) = Old Pulse Width

All registers are used

<u>Calls</u> INTEG for pulse width calculation (at address 81CO)

LDT1 to load Timer 1 (at address 8160)

CALL LDTI
Load Timer 1 unless pulse width is
zero or negative

(HL) (ST) Display Data

RETURN
PWM Subroutine CLOSL

Figure 6-20

6.2.4.3 Subroutine INTEG Specification (call at 81C0)

Function Calculate new pulse width.

If result is positive (greater than zero) store result.

Enter (BC) = Error Signal

(HL) = Old Pulse Width

Return (HL) = New Pulse Width

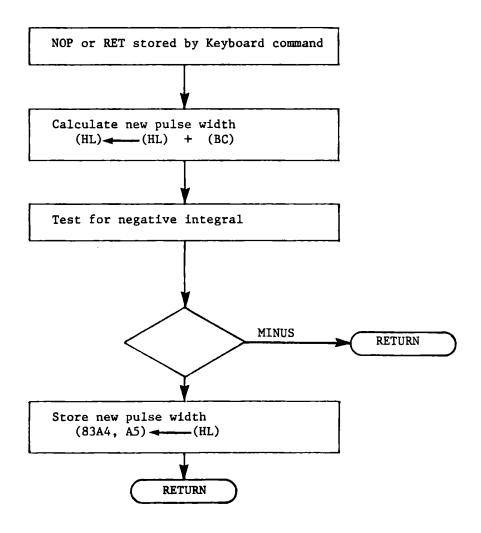
(83A4,A5) = New Pulse Width

Alternate Returns

If new pulse width is less than, or equal to zero, return without storing result.

Provision is made for insertion by keyboard control of a NOP or RET instruction at the entry to INTEG. The RET will disable closed loop control.

INTEG Called by CLOSL
(BC) = Error Signal
(HL) = Old Pulse Width



PWM Subroutine INTEG
Figure 6-21

6.2.4.4 Additional Command Keys

We will now define four additional command keys to be processed by KYTIM:

REG - Force output low temporarily for observing response.

(Dispatch to 8150).

MEM - Store data to be used by a subsequent version of CLOSL.

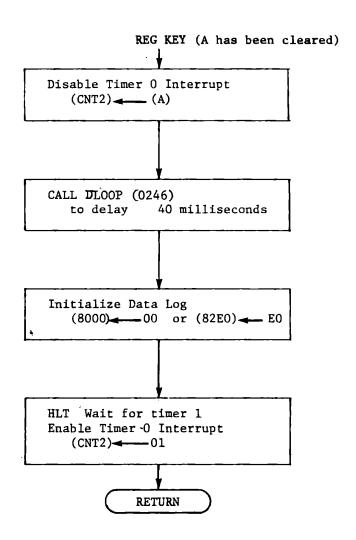
Store the data returned by ENTWD into memory locations 83A8,A9. (Dispatch to 8140).

BRK - Set open loop operation. (Dispatch to 8145).

CLR - Set closed loop operation. (Dispatch to 8147).

The change between open and closed loop operation will be accomplished by modifying subroutine INTEG. In response to CLR, store a NOP instruction at 81CO to allow INTEG to complete its functions. In response to BRK, store a RET instruction at 81CO to cause an immediate exit from INTEG.

(All of the addresses above refer to the author's solution. Your program may require different addresses).



REG Module of KYTIM
Figure 6-22

We will be interested in observing the response of the closed loop control system to a disturbance. This can be done by oscilloscope observation or by logging data. For convenience in using the oscilloscope the REG key will force a low output for long enough to discharge the capacitor. This is done by disabling the timer 0 interrupt and calling a monitor subroutine to generate the delay. At return from the delay start a new data log (as in NEXT and STEP), wait for a Timer 1 interrupt, and then re-enable and clear the Timer 0 interrupt. (See Figure 6-22).

Monitor subroutine DLOOP (located at 0246) is the delay function in GETKY. It repeatedly scans the keyboard, and returns after 20 milliseconds provided no key has been pressed. (This delay time is extended to about 40 milliseconds here by a different initial value).

Note that Timer 0, operating in mode 2, continues to run and reload itself even though its interrupt is disabled. Its output repeatedly triggers Timer 1 as in normal operation. The first interrupt form Timer 1 sets Port 1A7 low; it is not set high again until Timer 0 is enabled after the delay. A HLT instruction before enabling Timer 0 causes the first charge/discharge cycle after the delay to have its normal timing.

The main loop and subroutines KYTIM, CLOSL, and INTEG are given in Figure 6-23. Locations 8200 through 82CF are unchanged. Changes in the Main Loop and the KYTIM dispatch table are marked with an asterisk. The additions to KYTIM and the new subroutines CLOSL and INTEG are located at 8140.

	A D D I	۹	cor	DE	PWM	<u> </u>	JLT.	HGE		ONI	NU.	<u> </u>	- PL	U T 1/	- 11	
	8 22	0	D	B		I	N			P	0	R	T	0	A	Test keyboard
		1	0	0												(FF if no key)
	;	2	3	\mathcal{C}		I	2	R		A						
-	;	3	C	4		\cup	7	7		K	У	$ \mathcal{T} $	I	Μ		Call for keyboard
EET		4	0	0												
SH!		5	8	/												ently process
ING		6	C	D		C	A	L	L		V	0	L	7	M	Neasure voltage
CODING		,	5	0												(A) - A/D voltage
O		В	8	2												(FF idnot ready)
		•	F	5	*	P	U	ک	1+		P	5	ω			(FF ifnot ready) Save input Colculate error,
	,	\	C	D	*	C	A	L	L		C	L	0	S	4	Coloniate expose
	E	3	8	0	*)					adjust ouls excepted
	(5	8	1	*											adjust pulse width
ĒΜ		D	\mathcal{C}	D	*	C	A	L	L		D	W	0	R	D	Displayerror
/ST	1	E	$\widecheck{\mathcal{D}}$	1	*		-	Ů				-	-	•		and walter as request
3.5)		F	0	2	*								-			and voltage request
Ž	8 2 E	0	F	7	*	P	0	P		P	S	W				(A) = input voltage
Ā	1	1	2			1	X	I		4	_	P	3	A	0	(A) - input voltage Memory address for FILTR
T.		2	A	0							,					de) ETLTP
LER		3	8	3					_	-					,	7
5		1	C	2		0	A	L	1	\vdash	F	I	L	7	R	
MICROCOM	!	 5	7	0			 		_		Ė	+			-	(L)-raw voltage
800		6	8	2				_								(H)4 Littered entrace
Ş		7	/	/		上	X	エ		D		8	3	F	F	(H) - f. thered voltage
~		В	F	F			· `			_	/		Ĭ			Desplay
		9	8	3									 			Display row
		Α	C	D		0	A	1	7		ת	W	D	Q		and district
ΣWS		 B	D	4			,,									and grante
STE			0	2												water
INTEGRATED COMPUTER SYSTEMS	()	C	3		7	M	P		-	8	2	カ	0		woltage at right and filtered woltage
TER		Ε	D	0		Ť	Ť	Ė		<u> </u>	_ ب			Ť		
ľΡ		F .	8	2		<u> </u>						\vdash				
Ŏ	8	0					-									
<u></u>		1			*	C	H	A	N	G	F	D		F	R	oM
ATE		2		 		F	1	G		R			6	$\overline{}$	7	0M 8g
GR,		3	 		 	ľ	=			1,0	<u> </u>		1			7
AE AE		4			l			-								
=		<u> </u>				T	-	-	-						-	
		6			 	T	-				-					
		7	\vdash		<u> </u>	\vdash		<u> </u>		-				<u> </u>	Ė	
		В	\vdash	-	<u> </u>	f^{-}	\vdash	 		+-		\vdash			\vdash	Figure 6-23a
	L			Ц_	L			L							Щ.	1 -

<u>A</u>	D D R	_ co	DE	PWM			יינם	JU 1	, T141	<i>7</i> 1 C	.11.	L 141			
8	100	0	0		N	0	P								
	1	C	D		C	0 A	L	L		E	N	\mathcal{T}	W	\mathcal{D}	(A) - command
	2	4	6												(HL) - total paried
	3	0	3											_	on voltage
	4	E	\mathcal{B}		X	C	Ή	G							or voltage (DE) - data
SHE	5	2	/		L	X	I		H		8	/	0	8	address dispatch
CODING	6	0	8							-					table -10H
	7	8	7												70,7
	8	8	5		A	Þ	D		L						add command
	9	6	F		M	0	V		7		A)
	Α	6	E		M	0	V		L	,	M				> (57) - dispatch address
	В	E	5		P	V	5	H		H					() suppose
	С	F	3		$\overline{\mathcal{D}}$	\mathcal{I}		/ -		•					K
<u>≅</u>	D	D	\mathcal{B}	_	I	N			P	0	R	7	1	A	
SYSTEM	E	0	4			•			<u> </u>		T T		_		Let Part 1A/a licht
	F	F	6		0	R	$\overline{\mathcal{I}}$		4	0					Set Port 1A6 high for scope trigger without changing Port 1A7
Ž 8	// 0	4	0	_			_		1	_					without obancing
TRAINING 8	1	F	B		E	\mathcal{I}		-							Put 1A7
	2	7)	3		1	ν	T		P	0	R	T	/	A	1000 777
<u> </u>	3	0	4				<u> </u>		'		_		_		/
MICROCOMPUTER	4	A	F		X	R	A		A						Clear A and CY
ĕ├─	5	0	0		Ñ	7	P	-	,,		-				Comment of and of
	6	0	0		N	0	P	_				_			
<u> </u>	7	C	9		R	E	7	 		_			_		To dispatch address
≥	8	$\overline{}$	0		M	E	M	_			-				To disparch address
-	9	5	5		R	F	G	 					-		
-	Α	1/	7		A	D	D	R	<u> </u>			_			4. ')
≨ —	В	2	/		//	ファ	E	P		_	-			-	rigger scope
SYSTEMS	С	3	8		<u>3</u>	<u> </u>	7	-						_	Trigger scope Store desired voltage Set total period Store voltage, set width
} 	D	$\overline{}$	0		7	E	X	7			-				lto 1111
≝├ ─	E	2			0	R	K	1	-	-	-		-		sou voltage, ser wide
⋚├─	F	4	5		B	1	R	_			-		-		
	<u>.</u> 0	14	/			_	^	_			├		_	-	
္ကိုိ	1	+			-			_		_	-		_		
ᆰᅳ		+		<u> </u>	 	-		-	\vdash	-	-	_	-		
3R/-	2 	+			-		-	-	\vdash		-		-	-	
INTEGRATED COMPUTER	4	+	-		-	\vdash						<u> </u>	-		
≥	4 5	+	-		-		-	-		_	-		\vdash	-	
-		+	-	<u> </u>	-		-	-		-	├-	 	-	-	
-	<u>6</u> 7	+	-		-		\vdash	-	-	-	-		\vdash	 	
-		+	-	 	-		-	-	\vdash		\vdash	-			Figure 6-23b
$ldsymbol{ldsymbol{ldsymbol{ldsymbol{eta}}}$	8			<u> </u>		<u> </u>		<u> </u>	<u> </u>		<u> </u>	L		<u> </u>	

	A D D	R	col	DE	VII.	111		.41.	· <i>~</i> ,							
	8 /2	0	3	7		S	T	C	·							NEXT (mark with CY)
	8/2	1	2	4		L	X	I]	,	P	0	0	0	
		2	0	0												address data log
_		3	8	0	,											and load with its
SHEET		4	7	5		M	0	V		M		L				own address
		5	E	B		X	C	#			_					
CODING		6	2	2		S	Н	L	D		8	3	A	6		
100		7	A	6												
		8	8	3												
		9	D	0		R		C							-	Exit if STEP
		Α	/	_		L	X	I		D	,	F	F	E	E	To subtract 12H
		В		E												(O. 18 volts) from
_		С	F	F				ᆜ								desired vollage
SYSTEM		D	1	9		\mathcal{D}	A A O	\mathcal{D}		\mathcal{D}						(HL) = N - Vg (HL) = pulse width
SYS		E	2	9		D	A	D		H						(HL) - pulse width
		F	0	0		N	0	P	_)	4	. ,	_	
RAINING	8 / 3	0	2	2		Š	A	ᆫ	\mathcal{D}		8	૩	A	4		Stone calculated
FA		1	A	4				_								as integral Lond Times!
R		2	8	3			_	_								as integral
UTE		3	C	D		C	A	L	L		L		\mathcal{D}	1		
MICROCOMPUT		4	6	0				_								with calculated
200		5	8	/			1	_								pulse width
CR(6	C	9		R	E									· · ·
Ξ		7	0	0		N	Ŏ	P		_						
	813	9	7	B		\check{M}	0	¥		/ }		E	_			RUN
			D	3		0	V				1	W	0			Load Timer O
NS.	ļ	A	1	4			_				٠.	7				for total period
TE		С	7	A		M		¥		<u> </u>	工	D D	Δ			,
SYS		D	2	3		0	U	1		T	<u></u>	1~1	0			
ER	-		1 C	<i>4 9</i>		R	_	7								
INTEGRATED COMPUTER SYSTEMS		E F	0	0		N	EO	P								
OMF	3	0	U			, v										
Ö		1				I	D	F	A /	7	エ	\sim	A	<u>ل</u>		TO
λTE		2				F	I	EG	7	R			6	_	/	TO 8i
GR/		3				,	+	J	U	17	드		9		<u> </u>	
JE	-	4	<u> </u>		_											
=		5		\vdash											_	
		6	\vdash		-							<u> </u>				
	_	7					-									
		8	\vdash		- ,					-						Figure 6-23c
	L	-				Щ.	<u> </u>			L	Ц		<u> </u>		Щ.	. 19410 0 200

KYTIM - MEM, BRK, CLR, REG CODE EB MEM Storegains for CLOSE 22 3 2 8 3 3 CODING SHEET $\overline{\mathcal{C}}$ 9 RE 4 7 BRK - Open loop Enter RET in INTEG CLR - close loop Enter NOP in INTEG 3 M V I C 9 A 9 C 32 CO CO P 9 Start a new log. 000 Α X 0 0 В MICROCOMPUTER TRAINING SYSTEM MO V M 7 D E7 9 Ε OP F REG - Force output low Disable Finer Ointerrupt P Monitor function Delays Momilliseconds 3 CNTA 0 15 1 00 D 2 6 3 2 4 8000 Start a new log LXI 5 6 7 0 MOV M 8 Wait for Timer 1 7 7 9 MV Enable and clear 3 I A Α 0 NTEGRATED COMPUTER SYSTEMS Timer Ointerrupt В 0 3 CNT |U|TС D 0 9 RE7 Ε NOP 00 F 3 0 1 2 3 4 5 6 7 Figure 6-23d 8

R O R CODE LDT LDAD THERE	tl ve
9 D3 OUT TIM I	td ve
9 D3 OUT TIM I	ive
9 D3 OUT TIM I	
A / 5	
B $ O Q $ $ Q E T $ $ $	
С	
D E E	
S E	
9 F	
8 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
5 3	
3 4 4 5 5 5 5 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	
8 5	-
	-
8 9	
A	
B C C C C C C C C C C C C C C C C C C C	
Eg 3	
E E B B B B B B B B B B B B B B B B B B	
5 5	
6	
7	
Figure 6-23e	

	A D D	R	co	DE	PWM	_	SU	BRO	TUC	INE	C	LOS	SL	- 7	ÆR	SION 1
	8 /8	0	2	A		L	14	L	D		8	3	A	6		CLOSL
Ì		1	A	6									-	_		(L) - desired voltage
		2	8	3												
L		3	9	5		S	V	B		L						at entry (A)=measured (A)← -Error Signal
EET		4	6	7		M		V		\mathcal{H}		A				(H) - Error Signal
SHE		5		8		R	7									Exit if error = 0
CODING	_	6	E	ری		p	U	S	H		H		_			
O		7	2	F		C	M	A								Save display data
٦		8	4	F		M	0	V	_	C	_	A				
		9	3	F		C	M	C					-			(BC) - + Error
		A	9	F		S	B	_		A						as two by tes
j		В	4	7		M	0	V		B		A	-	-		
		С	0	3		\mathcal{I}	N			\mathcal{B}	-					J
ES		D	2	A		L	H	L	D		8	3	A	4		(HL) - Old pulse width
SYSTEM		E	A	4			-									
		F	8	3												
RAINING	8 / 9	0	C	D		C	A	4	L		\mathcal{I}	N	T	E	G	Calculate and store
\AII		1	C	0												Calculate and store new pulse width
⊢	·	2	8	1												7
TER	819	3	0	0		N	0	P								Save Blytes Sor
1PU		4	0	0		N	0	ρ							,	Save Blytes for proportional control
SON		5	0	0		N	0	P								
MICROCOMPU		6	C	D		C	A	7	L		L	D	7	1		Load Lines / with
MIC		7	6	0												new pulse width
		8	8	1												
		9	E	/		P	0	P		H						Lecover display data
(0		Α	C	9		R	E	T								1 0
EM		В														
SYSTEMS		С														
		D														
INTEGRATED COMPUTER		E														
MPU		F														
ဝိ	8	0				E	N	T	E	R		W	I	T	H	
ED		1	$oxed{oxed}$	_		L_	(A)	=	Μ	E	A	S	U	RED
٦AT		2				R	E	T	U	R	N		W	I	T	H
EGF		3	<u> </u>			<u> </u>	(L)	=	D	E	5	I	R	
Ϋ́		4		_	ļ	L_	1	H	2	Ξ	N R	E	G	A		IVE ERROR
		5		_	<u> </u>	工	N	T	E	G	R	A		_	A	ND TIMER!
	ļ <u></u>	6	<u> </u>	L_	<u> </u>	_	U	P	D	A	T	E	$\overline{\mathcal{D}}$	_		
		7	_	<u> </u>	<u> </u>		_		_				_		-	
		8	1			1] _	1 _								Figure 6-23f

	A D D	R	со	DE	PWM		ان د	DK	701	T 1/1	, т	M T T				510N 1
	8 / C	0	0	0		N	0	0		R	E	τ				Set by keyboard entry (HL) - new integral Sest for negative integral Exit if negative Store new integral
		1	0	9		<i>N</i> D	A	\mathcal{D}		\mathcal{B}						(HL) - new integral
		2	7	C		M	A O R	7		A	1.	H				Lest for negative
L		3	B	7		0	R	A		7)					integral
E		4	F	8		R	R M	I	N	V	S					Eit if negative
3.54		5	2	2		RS	\mathcal{H}	L	\mathcal{Q}		8	3	A	4		Store new integral
N		6	A 8	4												
CODING SHEET		7	8	3												
		8	C	9		R	E	7								
		9	L													
		Α														
		В			L				l							
		С				Ε	N	7	E	R		W	I	\mathcal{T}	H	
ΓEΜ		D					(B	C	7	=	+		E	8	ROR
MICROCOMPUTER TRAINING SYSTEM		E						H	1	7	=	0	<u>L</u>	\mathcal{D}		INTEGRAL
IG S		F									_					
Z Z	8	0		,	ļ <u>. </u>	R	E	Т	U	R	N	<u> </u>	ļ			
RAI		1	1_				(\mathcal{B}	C E)	_	P	R	E	S	ERVED ERVED
RT		2	↓				(D	E	3		9	R	E	S	ERVED
JTE		3	_	_		<u> </u>		H	L)	_	N	E	W		INTEGRAL
MPL		4	_			ļ		_					_		_	
000		5	_			N_{-}	EN	W	_	I	/V	T	E	G	R	AL STORED TIVE
CRC		6	1			U	N	L	E	2	<u> </u>	N	E	G	A	TIVE
Ž		7	_			ļ	_		_	ļ			ļ			
		8	_			L	<u> </u>		<u> </u>		_					
		9	╀-	<u> </u>		<u> </u>	_		_	-	ļ <u>-</u>		_	-		
S			↓_			<u> </u>				<u> </u>			 —			
TEN		В	-			<u> </u>			_	_	_		<u> </u>			
ΥS		C .	+						<u> </u>	_	<u> </u>					
æ.		D	╀	-		ļ	_	_	<u> </u>	<u> </u>	_		_	<u> </u>		
INTEGRATED COMPUTER SYSTEMS			\vdash			<u> </u>	_			ļ	<u> </u>	_	<u> </u>	_		
MP			-			-		_	-		-		1			
S	8	0	}_		 	-	_	_	-	-	_		-			
TEL	<u> </u>	1	╁┈	<u> </u>							-		l I		<u>.</u>	
,RA		2	╀		 		-		-		-			-		
TEG		3	┼	_	ļ	-		_	-	-	-	-	-		_	
Z		4	╀	╁	 				-		-	-	<u> </u>		-	
		5	+	1		-	-	-	-	-	-		 		-	
		6	+	-	 	-	-	\vdash	-	_	-	_	-			
		7	+	\vdash	 	-	+	\vdash	├-	├-	├-	├	\vdash	-	-	Riguma 6-22g
		8			<u> </u>	<u>L</u> .	l	L_		<u></u>	ل	<u> </u>	<u> </u>	<u></u>	L	Figure 6-23g

6.2.5 Closed Loop Operation

With closed loop control the program will force the output voltage to equal the requested voltage. You can enter a voltage with the NEXT key, which calculates a new pulse width, or with STEP, which does not. In either case the program will adjust the duty cycle to generate the requested voltage. The voltage will now be independent of the total period and the OPTO SENSE pot setting. It will fluctuate over a range of about six counts (60 millivolts), from C5 to CB. fluctuation is displayed in the measured voltage (at the right) and the error signal (at the left). These will be changing so rapidly as to be unreadable. Since the measurement and display are handled in the main loop, pressing a key will stop the measurements and allow you to read the voltage. By doing this repeatedly you can observe the range of measurements. The ADDR key will do this without changing any stored data or controls. The fluctuation of the voltage is an inherent and undesirable effect of closed loop control Fortunately it can be reduced by several means that we will investigate later.

Perform the experiments described in the following sections. Results of these experiments can be observed with your voltmeter. The experiments of Section 6.2.6 require either an oscilloscope or the laborious task of plotting data from the log made by the VOLTM subroutine.

6.2.5.1 Seeking Desired Voltage

Enter the following data and commands:

CLR	Set closed loop control
400, RUN	Set 0.5 millisecond period
C8,STEP	Set 2.0 volts

Observe the filtered voltage in the second pair of digits from the right, and observe the voltmeter reading. They should agree closely. Adjust the ANALOG IN pot, and observe that this now changes the actual output as observed on the voltmeter, because the closed loop control forces its measured input voltage to equal the requested voltage. Now enter the following voltages and observe the resulting output.

Result

Total Period 0.5 ms (400 hex)

Voltage Request

Decimal	Hex		Voltmeter	A/D
2.40	FO	STEP		
2.00	C8	STEP		
1.50	96	STEP		
1.00	64	STEP		

6.2.5.2 External Resistance Variation

Enter the following data and commands:

CLR	Set closed loop control
400, RUN	Set 0.5 millisecond period
C8,STEP	Set 2.0 volts

Adjust the OPTO SENSE pot over its full range from left to right and back to the left again. There should be no appreciable change in the voltmeter reading.

BRK Set open loop control

Adjust the SENSE pot to reduce the voltage to 1.50 volts (96 hex).

CLR Set closed loop control

Result

Request the several voltages again and observe the results.

Total Period 0.5 ms (400 hex)

Voltage Request

Decimal	liex		Voltmeter	A/D
2.40	FO	STEP		
2.00	C8	STEP		
1.50	96	STEP		
1.00	64	STEP		

The results should be essentially the same as before.

6.2.5.3 Total Period Variation

Return the SENSE pot to the full left position. Enter:

CLR	Set	clos	sed	loop	cont	rol
400, RUN	Set	0.5	mi l	lisec	ond	period
C8.STEP	Set	2.0	vol	lts		

Now enter different total periods and observe the results.

Period			Voltage					
Milliseconds	Hex		Voltmeter	A/D				
0.375	0300	RUN						
0.50	0400	RUN						
0.75	0600	RUN						
1.00	0800	RUN	-					
2.00	1000	RUN						
32.00	0000	RUN						

With the final value (32 milliseconds) a greater difference between the voltmeter and the A/D converter may occur because of the wide variation in the voltage within a charge/discharge cycle. With the two millisecond period the capacitor charges and discharges by 300 millivolts in each cycle. At 32 milliseconds the voltage is actually swinging from 0.6 to 3.6 volts, but the average is held to 2.1 volts by closed loop control.

CLOSED LOOP CONTROL

After the 32 millisecond test enter 400,RUN. The voltage will rise very nearly to 5.0 volts, because the charging time will have been set to about 15 milliseconds, much longer than the newly entered total period. Since the A/D converter cannot measure the off-scale voltage it returns FF. CLOSL calculates an error signal of C8 - FF = -37 and repeatedly reduces the pulse width by this amount until the voltage returns to a value within range, and thereafter adjusts the pulse width appropriately.

6.2.5.4 Response to Disturbance

Enter the following data and commands:

CLR Set closed loop control

400, RUN Set 0.5 millisecond period

C8,STEP Set 2.0 volts

REG Force voltage low

The output voltage will momentarily drop to about 0.2 volt and rise again to the requested voltage. We will observe this response in detail in section 6.2.6. The voltmeter will show the drop, but it will not be fast enough to go down more than a few tenths of a volt before closed loop control resumes and restores the desired voltage.

6.2.5.5 Open Loop Operation

The open loop control program is able to maintain a voltage very well, but is unable to compensate for changes in external conditions, as we observed earlier. Once an appropriate pulse width has been set by the closed loop system we can open the loop and the voltage will remain essentially constant. Enter these data and commands:

CLR Set close loop control

400, RUN Set 0.5 millisecond period

C8,STEP Set 2.0 volts

REG Force voltage low

BRK Set open loop control

REG Force voltage low

Observe that the open loop system returns to the requested voltage once an appropriate pulse width has been set by closed loop control. Now with open loop operation, adjust the SENSE pot to obtain 1.50 volts. Now press:

CLR Set closed loop control

BRK Set open loop control

REG Force output low

CLOSED LOOP CONTROL

Again closed loop control has set the pulse width and open loop control can restore the voltage after a disturbance. This is sometimes an acceptable mode of operation for a control system where external variables change slowly. Closed loop control can be invoked periodically, or when conditions are known to have changed. After the necessary adjustments have been made to the control force an open loop system can maintain the operation.

Students who are not especially interested in closed loop control may want to skip the remainder of Chapter 6. It is concerned with methods of improving the performance of a closed loop control system.

6.2.6 Closed Loop Response

Observing the response to a disturbance under various conditions demonstrates very important features of closed loop control systems. This can be done most conveniently with an oscilloscope, or data can be logged and plotted. Section 6.2.6.1 describes the use of the oscilloscope and shows open and closed loop waveforms. Section 6.2.6.2 presents closed loop results obtained by the data log, and discusses the waveforms. The effect of changing the total period is shown in 6.2.6.3.

6.2.6.1 Oscilloscope Observation

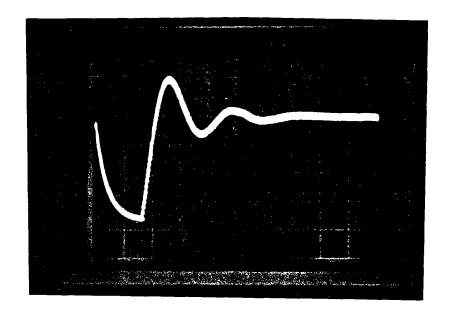
The oscilloscope is to be triggered by Port 1A6, which is set when a command key has been pressed and released. The capacitor voltage is to be observed.

Use a time scale of 20 milliseconds per division and a voltage scale of 0.5 volts/division. Enter the following:

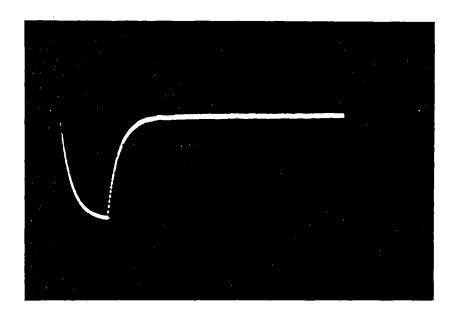
CLR Set closed loop control

400, RUN Set 0.5 millisecond period

C8,STEP Set 2.0 volts



CLOSED LOOP RESPONSE



OPEN LOOP RESPONSE

SCALES: 0.5 VOLT/DIV 20 MS/DIV
Open and Closed Loop Waveforms
Figure 6-24

Test the scope triggering by repeatedly pressing ADDR. A single sweep should occur each time you release the key. This may take some adjustment of the oscilloscope trigger controls.

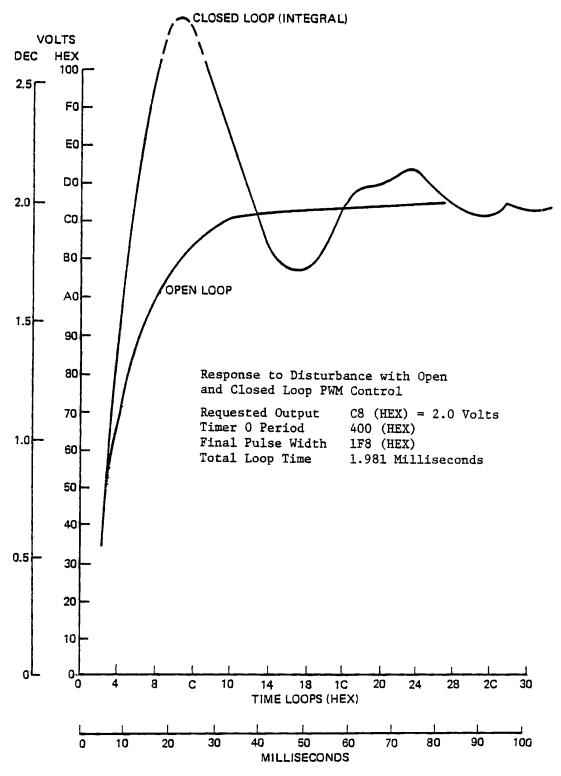
When you press and release REG the output will be forced low for about 40 milliseconds and then closed loop control will be resumed. The oscilloscope should display a waveform similar to the upper photograph in Figure 6-24. Now:

BRK Set open loop control

REG Force output low

A waveform similar to the lower photograph in Figure 6-24 should be seen.

On repeated operations of REG with open loop control the waveform should be very consistent. It merely shows the charging of the capacitor in response to a constant duty cycle PWM voltage. In closed loop control there will be substantial variations in the waveform, principally because of the random relationship between the time that the A/D conversion is completed and the time that CLOSL acts on the result. The reasons for the waveshape are discussed in the next section.



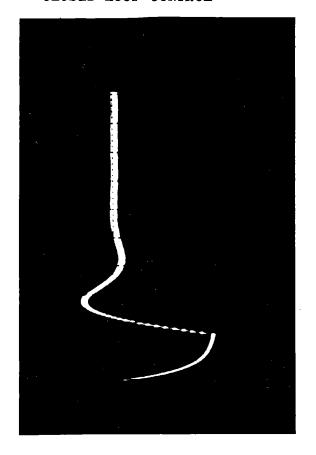
Closed Loop Response Waveform

Figure 6-25

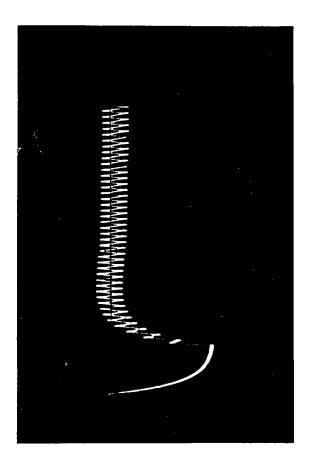
6.2.6.2 Closed Loop Response Waveform

Waveforms observed as the closed loop control system restores the desired voltage after a disturbance are shown in Figures 6-24 and The large overshoot as the desired voltage as approached, and the continuing oscillation above and below the desired voltage are inherent and undesirable results of Integral Control. When the low output voltage is measured the computer calculates a large error and adjusts the pulse width accordingly. At the next signal measurement a smaller but still substantial error is observed and a further adjustment is made to the pulse width. This continues until the actual voltage has reached and passed the desired voltage. At this point the pulse width is set too wide for the desired voltage. A negative error is detected and the pulse width is reduced. The capacitor is still charging however, so the voltage continues to Moreover, the error detected is small and the adjustment made rise. is not yet sufficient to bring the voltage back down to the desired The result is that the voltage rises substantially above the value. desired value before it starts down. This is called "overshoot". A number of measurements and adjustments are made before the voltage again reaches the desired value. By now the closed loop control system has reduced the pulse width too far, and undershoot occurs. The process continues indefinitely, reaching a steady state of oscillation above and below the desired value. The amount of overshoot and undershoot, the amplitude of the oscillation, and the time before a steady state is reached will be seen to depend on the total period of the pulse width modulation.

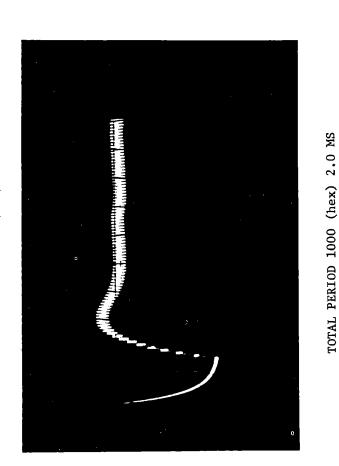
CLOSED LOOP CONTROL



TOTAL PERIOD 800 (hex) 1.0 MS



TOTAL PERIOD 400 (hex) 0.5 MS



SCALES: 0.5 VOLTS/DIV, 20 MS/DIV

TOTAL PERIOD 2000 (hex) 4.0 MS

EFFECT OF TOTAL PERIOD FIGURE 6-26

6.2.6.3 Effect of Total Period

Figure 6-26 shows oscilloscope traces for four different values of total period. It is apparent that by using a long period we can greatly reduce the overshoot and oscillation, but at the cost of introducing a large voltage fluctuation in each charge/discharge cycle. With a total period of 2,000 (hex) the overshoot is small but the voltage varies by about 0.4 volt during each cycle. This is clearly not a desirable scheme. When we develop a more sophisticated closed loop control system in Section 6.3 we will overcome this problem.

6.2.6.4 Gain of Integral Control

We have exercised integral control by adding the error signal to the steady state pulse width. Let use examine the meaning of this procedure in terms of the integral control equation:

$$(a) F = G \int E$$

Since the Integral represents the sum of all past error signals plus the new measurement, we can say:

(b)
$$F = GE + F'$$

Where F' is the previous value of the control force.

The relation tc/tp represents the control force, since we can scale these two values without changing the result, but changing either alone does affect the output. Therefore:

(c)
$$F = \frac{t_c}{t_p}$$

(d)
$$F' = \frac{t_{C'}}{t_{p}}$$
 for constant total period

(e)
$$\frac{t_c}{t_p} = GE + \frac{t_{c'}}{t_p} \text{ from (b) and (d)}$$

The procedure we have used added the error signal E to the old pulse width tc' to obtain a new tc.

$$(f) \qquad t_{c} = E + t_{c}$$

Dividing by tp:

$$(g) \qquad \frac{t_{c}}{t_{p}} = \frac{E}{t_{p}} + \frac{t_{c'}}{t_{p}}$$

From inspection of equations (e) and (g) it is apparent that G = 1/tp. When we increased the total period to reduce the overshoot we were reducing the gain of the control system. It is much more effective to do this by arithmetic in the control calculation, and this will be done in Section 6.3.

6.2.6.5 Pure Proportional Control

At the beginning of Section 6.2 we presented the control equation for proportional control with a steady state force:

(a)
$$F = GE + S$$

The program of Section 6.3 will introduce a proportional term separate from the integral term we have been using. To see the effect of proportional control alone, change the RM instruction in INTEG to RET. In the program of Figure 6-23g, this change is:

81C4 C9 RET (was RM)

The pulse width will be calculated as before but the integral will never be stored. This gives proportional control with a steady state term set by the open loop system. Do this experiment:

CLR Set closed loop (proportional) control

C8, STEP Request 2.0 volts

03E8, RUN Enter total period

Adjust the total period to obtain 2.0 volts.

REG Force output low

Observe that proportional control restores the output voltage much as open loop control would. Now open the loop and restore the 500 microsecond total period.

BRK

0400, RUN

The voltage will be lower than requested, because of the excessive total period. Close the loop with proportional conrol.

CLR

Set proportional control

Proportional control will increase the output voltage, but will not reach the desired value. This is because the nominal pulse width calculated in response to NEXT remains as the steady state value. The pulse width is increased by the proportional control system, but only by the amount of the error measured at that instant with no cumulative correction. In the system we have here, pure proportional control is little more effective than open loop control. It is useful in systems where no steady state control force is needed, or in combination with integral control.

6.3 PROPORTIONAL PLUS INTEGRAL CONTROL

A control system that requires a steady state force to be adjusted for variable external conditions demands integral control. Random disturbances are better overcome by proportional control. Therefore a combination of both forms of control is very commonly used; it is called Proportional Plus Integral control. The control equation becomes:

$$F = G_{p}E + G_{i} \int E$$

We determined at the end of section 6.2 that $G_i = 1/\text{tp}$ in our present system, where the error signal is added into the integral and the result sets the charging pulse width.

If we then add the error term again before loading the timer but do not change the integral in response to this addition, we will have a proportional plus integral system with equal gains.

We recognized in Section 6.2.6.4 a need to reduce the integral gain to avoid large overshoot. We also observed that with pure proportional control a large error signal was needed to affect the output significantly. The proportional control would have been more effective with a higher gain, since then the correction applied would have been greater for any given error signal. It is very common in proportional plus integral control systems for the proportional gain to be much greater than the integral gain.

6.3.1 Applying Gain to Error Signal

In our new system we will provide for dividing the error signal by some value before adding it into the integral term, and multiplying it by some other value before adding it as the proportional term.

For ease of computation the integral gain divisor will be of the form $1/2^n$, so that the division is merely a shift of n bits to the right. Typical values for n will be 0 to 4, giving division by 1, 2, 4, 8 or 16. The multiplication for the proportional term will be done by repeated addition, so the number used (again typically 0 to 4) will now actually be the multiplier. Our control equation will now be

$$F = \frac{KE}{t_p} + \frac{1}{2^n t_p} \int E$$

For convenience in further discussion we will ignore the total period here and speak of K and $1/2^n$ as the proportional and integral gains. These are the data elements stored by the MEM key, at (83A8,A9). Both values are to be entered: K first, followed by n, followed by MEM. For instance, 203 MEM will set k=2 and n=3, giving a proportional gain of 2 and integral gain of 1/8. Note that the data entry procedure stores k at 83A9 and n at 83A8.

The calculation procedure is described briefly below, with detailed flow charts in Figure 6-27 and the program in Figure 6-28.

CLOSL: Calculate error signal

INTEG: Divide error by 2ⁿ

Add to old integral

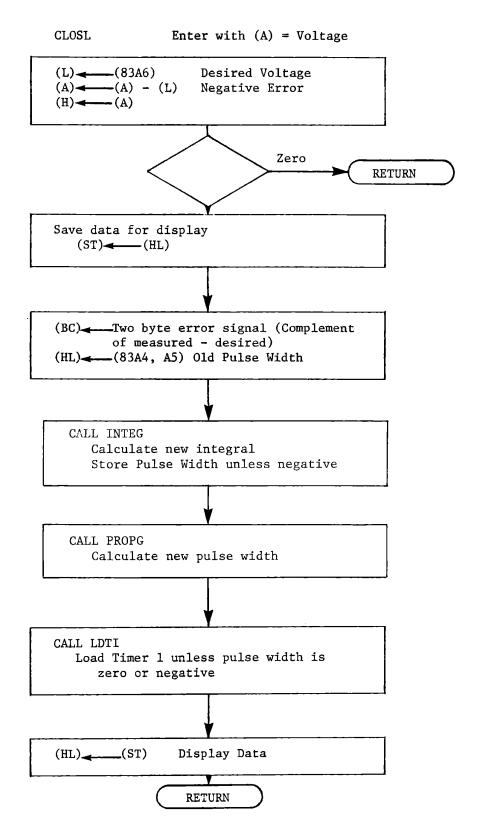
Store new integral unless negative

PROPG: Multiply error by k

Add to new integral

LDT1: Load Timer 1 with new pulse width

unless zero or negative



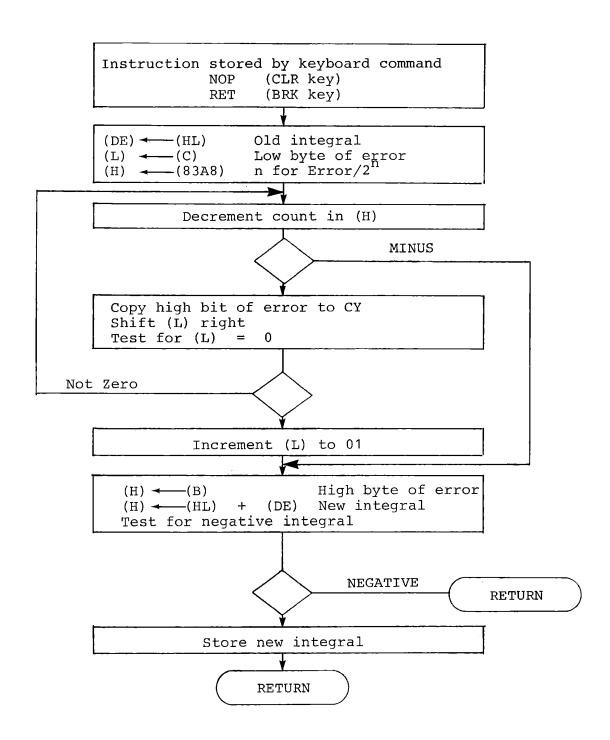
PWM Subroutine CLOSL Figure 6-27a

6.3.2 Subroutine CLOSL Version 2

CLOSL is unchanged except that after calling INTEG to generate and store the new integral it now calls another subroutine PROPG to apply the proportional error term.

A revised version of INTEG divides the error by 2^n (shifts the error right by n bits) and adds this to the old integral. The new integral is stored, unless it is negative; it is returned in (HL) even if it is negative.

The new subroutine PROPG repeatedly adds the error into the integral, according to the value entered as K and stored at 83A9. Thus PROPG returns the sum of the integral and proportional terms.

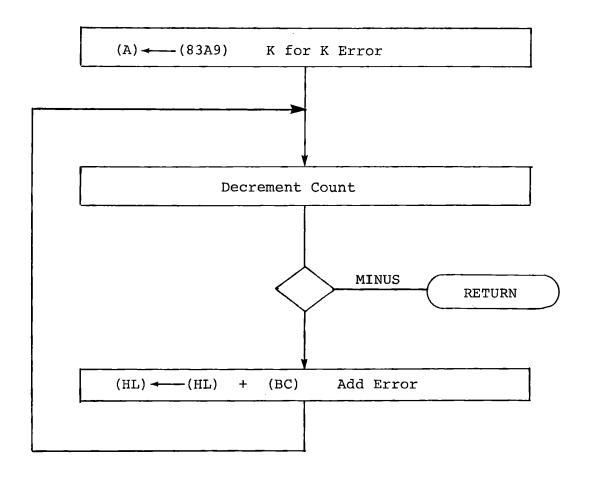


PWM - Subroutine INTEG Version 2
Figure 6-27b

6.3.2.1 Subroutine INTEG Version 2

Again provision is made for modifying the program by BRK and CLR, storing NOP or RET at the start of INTEG. CLR will set integral control (by entering NOP). BRK will disable integral control, leaving pure proportional control. If the proportional gain multiplier is set to zero the process is then identical to open loop operation. To divide the error signal by 2 we shift the error right by n bits. Since the high byte of the error is always either 00 or FF we need not shift the high byte, but merely shift its low bit into the low byte as the low bit is shifted right. Note from Figure 6-27b that n is decremented before shifting, so that if n is zero initially no shifting is done and the gain is unity.

When n is greater than zero and the error signal is a small positive value, the result may be zero even if there was an error. It is desirable to increase the integral in this case, especially since an equally small negative error will reduce the integral. (FFFF shifted right remains FFFF). Therefore, we test the shifted error for zero, and if it is zero increment it to 01. Now a positive error will always increase the integral and a negative error will always decrease it, no matter how small the gain. A zero error does not affect the pulse width because of the Return If Zero in CLOSL after the error calculation. As before we must test for a negative integral in subroutine INTEG before storing the result. Zero is not forbidden here, and negative integrals would be acceptable but, as we will demonstrate, there is a possibility of losing control.



PWM - Subroutine PROPG
Figure 6-27c

6.3.2.2 Subroutine PROPG (Figure 6-27c)

The error signal is to be multiplied by K (stored at 83A9) and added to the integral, to obtain the new pulse width. Subroutine INTEG preserves the error in (BC) and returns the new integral in (HL). Since K will always be a small integer value it is efficient to simply add (BC) into (HL) while counting down. Once again the count should be decremented before the addition, so that a zero value for K will give a zero again.

6.3.3 Revised Program

The revised program with the new INTEG (version 2) and PROPG is given in Figure 6-28. For convenient reference, the main loop and subroutines KYTIM and LDT1 are repeated although no changes have been made. CLOSL now includes the call to PROPG after the call to INTEG. The new subroutines appear in Figures 6-28g and 6-28h.

A D D R CODE PWM VOLTAGE CONTROL - MAIN LOOP

	A D D	R	COI	DE	7 1177					7111						
	8 2D	0	7	\mathcal{B}		I	N			ρ	0	R	\mathcal{I}	0	A	Lest keyboard
		1	0	0	_											(FF-ifno key)
		2	3	C		工	N	R		A						
-		3	\mathcal{C}	4		C	N	Z			K	Y	\mathcal{T}	I	M	Call for keyboard
SHEET		4	0	0							_					entry and process
		5	8	/												
CODING		6	0	Ð		0	A	L	L		V	0	乚	T	M	Measurevoltage
00		7	ی	0							_					(A) - A/D voltage
		8	8	2												(FF if not ready)
		9	F	<u>ح</u>		P	\mathcal{U}	S	H		P	5	W			Save input
		A	\mathcal{C}	Δ		C	A	L	<u>L</u>		C	L	0	2	L	Calculate error
		В	8	0								_				adjust pulse width
_		С	8	/								_				0 /
STEM		D	2	\mathcal{D}		C	A	L	L		\mathcal{D}	ω	0	R	\mathcal{Q}	Displayerror
SYS.		E	D	_												and voltage request
		F	0	2												
AINING	8 2E	0	F			ρ	0	P		P	5	W				(A) - input voltage Memory address for FILTR
RAI		1	2	/		L	X	I		H	ار	8	3	A	0	Memory address
щ Н		2	A	0							_					YOU FILTR
ш		3	8	3												0
MICROCOMPUT		4	2	D		C	A	L	<u>L</u>		F	I	L	T	R	
000		5	7	0												(L) - raw voltage (H) - filter voltage
SRO	- 	6	8	2												(H) - filter voltage
<u> </u>		7	/			<u>L</u>	Χ	I		D		8	3	F	F	
		8	F	F												
		9	8	3								_				
S		Α	C	<u>D</u>		<u>C</u>	A	<u>L</u>	<u>L</u>		\mathcal{D}	W	\mathcal{D}	2		
Ē		В	1	4										-		
YST		С	0	23		_		_								
RS		D				7	M	P	<u>. </u>	ļ	8	2	D	0		
JTE		E	D	0												
INTEGRATED COMPUTER SYSTEMS		F .	8	3					 	ļ						
8	3	0				_			-			_	<u> </u>	_		
FED		1	_			エ	D	E		T	エ	C		$\overline{}$		TO
RAT		2	_			F	I	G	U	R	E		6	_	2	3a
Ē	ļ	3	_			_	_			_						
N		4	ļ		ļ		_	_	<u> </u>	<u> </u>		_	_	_	_	
		5	-		ļ	<u> </u>		_	_	<u> </u>			-		-	
	<u> </u>	6			 	_	-	_	_		_	_	_		_	
		7	-				ļ		-	-	-	<u> </u>	_	_	-	7
	. 130	8			<u> </u>	<u>L</u>	<u> </u>	L_	<u> </u>	<u>L</u>	<u> </u>		<u></u>		L	Figure 6-28a

	Α_	D D	R	со	DE	PWM		_									
	8	10	0	0	0		\mathcal{N}	0	P								
			1	C	D		\mathcal{C}	A	L	7		E	N	7	W	\mathcal{D}	(A) - Command
			2	4	6												(HL) - total period
_		,	3	0	3												ov voltage
SHEET			4	E	\mathcal{B}		X	C	Н	G							(WE) - Glata
			5	2	/		<u>L</u>	χ	I		\mathcal{H}		8	/	0	8	address dispatch
N			6	0	8)					table -10H
CODING			7	8	/												
O		_	8	8	5		A	$\overline{\mathcal{D}}$	\mathcal{D}		L						add command
			9	6	F		M	0	٧		L		A				
			Α	6	E		M	0	V		L	7	M				(57) - dispatch ashress
			В	E	5		$\overrightarrow{\mathcal{P}}$	V	S	#		Ħ					
	_		С	F	3	,	$\overline{\mathcal{D}}$	I	_				·		-		K
Ε̈́		•	D	ת	\mathcal{B}		Ī	N			P	0	R	T	1	A	
SYSTEM			E	0	4			_			Ť					_	Set Port 1A6 high
	-		F	E	6		0	R	エ		4	0			<u> </u>		for scope trigger
N	8	//	0	4	0				_		-		ļ	-			to the taken of
TRAINING			1	F	\mathcal{B}		E	I					-				Fort 1A7
H			2	7	3	-	0	U	T		P	0	R	T	/	A	100
ER			3	0	//			U	-			U	`	<u>'</u>	 	<u> </u>	
PUT			4	Δ	F		X	R	A.		A		-				Clear Aand CY
MICROCOMPUTER			5	1/2	2		N	0	P		<i>,,</i>		\vdash				Geac Harle C.
OC	_		6	0	0		A/	0	φ				<u> </u>	-			
ICF.			7	0	9		R	E	<u>-</u>		ļ		\vdash		-		1 1 to ladde
≥			8	1/	0		M	E	M								To dispatch address
			9	5	0		R	L	G						_		
			Α	1	7		A			D							4:001
MS			В	2	/		S	T	E	RAP			-	_	1		Trigger scope Store desired voltage Set total period Store voltage set width
STE			c	3	8		$\frac{3}{R}$	ΰ	N	1							Store descred vollage
SYS			D	2	0		N	E	X	T					-		Secrorar sperior
ER			 E	f –	5		\mathcal{B}	$\frac{C}{R}$	रि				-				store vollage ser wises
TU			- F	4	7		7	1	R				-				
MO	3		0	7	 		1	<u></u>				_	-				
Ö	Ë		1	╁			┢						-	-	1		
Ţ		,	2	-				-		-					-	\vdash	
3RA	_			\vdash	-		┢				<u> </u>				-		
INTEGRATED COMPUTER SYSTEMS			3	\vdash	-		\vdash		_				-	ļ <u>.</u>	-		
Z			4	-	<u> </u>		┼—							<u> </u>			
	<u> </u>		5		-		_			<u> </u>			\vdash				
	-		6	-	-	<u> </u>	\vdash			<u> </u>		<u> </u>	\vdash		-		
	\vdash		7	-	-	ļ	\vdash						-		-	_	Rigues 6 22
	<u> </u>		8	<u> </u>	<u>L</u>	}			<u> </u>	L	L	<u></u>	<u> </u>	<u> </u>	<u> </u>	<u> </u>	Figure 6-28b

	A D D	R	co	DE	KYT	IM	-	NE	KT,	S	EP	, I	RUN			
	8 /2	0	3	7		g	7	C								NEXT (mark with CY)
	812	1	2	/		L	Χ	I		14		8	0	0	0	STEP/ CY coas)
		2	0	0			_				-	_		_		address data log
⊢		3	8	0												and load with its
SHEET		4	7	5		M	0	V		M	,	L				own address
		5	E	B		X	C	14	G							
CODING		6	2	2		S	1+	L	A		8	3	A	6		
300		7	A	6												
		8	8	3												
		9	D	0		R	N	\mathcal{C}								Exit if STEP
		Α	_	/		L	X	I		\mathcal{D}		F	F	E	E	To subtract 12H
		В	E	E												(0.18 volts) from
_		С	F	F			_									desired voltage
LEM		D	/	9	I	\mathcal{D}	A	\mathcal{D}		\mathcal{D}						(HL) = N - Va
SYSTEM		E	2	9		D	A	D		H						(HL) - v - Vg (HL) - pulse width
		F	0	0		NS	0	P								/
TRAINING	8 /3	0	2	2		S	14	L	D		8	3	A	4		Store calculated
RAI		1	A	4												pulse width
		2	8	3	_											as integral Lond Times!
프		3	C	\supset		C	A	L	L		L	\mathcal{D}	7	/		Load Times!
MICROCOMPUTER		4	6	0												with calculated
Ö		5	8	/										_		pulse width
SRO		6	C	9		R	E	7								/
Σ		7	0	0		N	0	P							<u>.</u>	
	813	8	7	\mathcal{B}		M	0	V		A		E				RUN
		9	2	3	L	0	\mathcal{U}	7		7	I	M	0		_	Load Liner O for total period
S		Α	1	4			_									for total period
SYSTEMS		В	7	A		M	0	V		A	,	\mathcal{D} \mathcal{M}				
YST		С	2	3		0	\mathcal{U}	1		T	I	M	0			
R S		D	1	4		L					_	ļ			ļ	
JTE		E	C	9		R	\mathcal{E}	7	<u>.</u> .	_					ļ	
MΡ		F	0	0		N	0	P		_	ļ			ļ		
ပ္ပ	3	0					<u> </u>	-		<u> </u>				_		
red		1	_	-		$ \mathcal{I} $	D		N	T	I	C	4	1	<u> </u>	TO
RA		2	_			F	I	G	U	R	E	<u> </u>	6	_	/	2i
INTEGRATED COMPUTER		3	_			_	_		_	_	-					
Z		4	_	_		<u> </u>	_	_	_	_	<u> </u>	ļ	ļ	<u> </u>	-	
		5	-			<u> </u>	-		-			-	_	<u> </u>	-	
	<u></u>	6	-		 	-			_	-			ļ		-	, , , , , , , , , , , , , , , , , , ,
	<u> </u>	7				-	-		-	-	_	-	_		-	
		8	<u> </u>	<u> </u>	[Ľ.	L_				<u> </u>	L		<u> </u>	Figure 6-28c

KYTIM - MEM, BRK, CLR, REG CODE MEM Store gains for CLOSE XCHG SHLD EB 83A8 22 A8 2 83 3 CODING SHEET C9 RE BRK - Open loop Enter RET in INTEG CLR - Close loop Enter NOP in INTEG 3 9 $\overline{\mathcal{C}}$ 2 C0 3 A C 8 Startanew log 000 2 Α 0 0 В С MICROCOMPUTER TRAINING SYSTEM L 0 1 M D 5 ET 9 R C Ε 0 P F 0 0 REG-Ince output low Doorba Timer Dinterrupt Monitor function Delay ~ Dmillisecons CNT2 3 UT D ۵ سی/ 1 CALL 2 C D 3 2 4 8000 Start a new log. 2 I 17 6 8 7 0 M MOV 8 Enable and clear 6 9 3 E 01 Α SYSTEMS Line Ointerrupt В 3 D CNTA OUT С NTEGRATED COMPUTER 9 RET C Ε NOP 00 F 0 1 2 3 4 6 7 Figure 6-28d 8

	A D D R CODE									T)	ME	R 1				
	8 /6	0	2	\mathcal{B}		D	C	Х		1+						LDTI Sest pulse width for gers or negative Exit if = 0 Load Timer! with charging pulse width
		1	7	C		M	_	_		A	,	Н			,	Lest pulse width
		2	B	7		0	R	H		A						Low ners or negative
_		3	2 F	3		T	0822	X		Н						1 8
SHEET		4				IR			2	υ	S					Exit if = 0
		5	7	2		M	٥	V		A	<u>.</u>	L				Load Times!
CODING		6	0	3		0	U	\mathcal{T}		\mathcal{T}	I	M	1			with charging
00		7	1	5						j						pulse wiath
		8	7	C		M	0	V		17	ـرا	Н				
		9	(0	U	7		\mathcal{T}	I	M	_			
		Α	1	5							_		_		_	
		В	C	9		R	E	7								
5		C		\vdash				_								
SYSTEM	 	D	-	_		_					_					
SYS	- :	E	<u> </u>	ш	1	_		_				-			_	
		F	-				-	_		_					_	
TRAINING	8	0					_						_			
TRA		1		\vdash		_	-	_				_	-	-		
		2	-	1	_											
MICROCOMPUTER		4		-			-									
JMC		5	-			_				-						
OC		6					-									
ICF		7														
[≤		8				-	-								_	-
		9				-										
		A		Н			-		-						_	
SYSTEMS		В	_													
STE		С					-									
		D														
COMPUTER		E														
ΛΡU		F														
CO	3	0														
		1														
NTEGRATED		2														
EGF		3														
Z		4				_										
		5					<u> </u>							_		
	ļ	6						_								
		7	ļ	_		_	_	_						-		
		8	<u> </u>		<u> </u>	<u> </u>	ŀ	<u> </u>		<u></u>						Figure 6-28e

	A D D R	_ CODE	PWM -	50	אמ	JUT.	TNE	, C.	<u> </u>	ъ.		/ER	510N Z
	8 18 0	2 A	L	H	L	Θ		8	3	A	6		CLOSL
	1	A 6											(L) - desired voltage
	2	83				·							
_	3	95	3	\mathcal{U}	B		<u>L</u>						atentry (A)= measured (A) = - Error
IEET	4	67	M	0	V		H		A				(H)← - Error
3 SHI	5	08	R	Z				,					Exit if error = 0
CODING	6	E 5	P	V	S	H		H					Save display data
COL	7	2F	C	M	A								
)	8	4 F	M	0	V		C	1	A				
	9	3 F	C	M	C								(BC) - + Error
	A	9 F	S	B	B		A						(as two by tea
	В	47	M	0	V		B	_	A				
_	С	03	エ	N	X		В						<i>J</i>
SYSTEM	. D	2 A	<u> </u>	H	L	D		8	3	A	4		(HL) - old integral
ΥS	E	A4				ļ	<u> </u>						0
	F	83											
RAINING	8 /90	CD	C	A	L	L		I	N	T	E	G	Calculate and
RAI	1	100		↓									store new integral
R	2	8/		<u> </u>						ļ			store new integral (HL) ← (HL)+ (BC)/2m
JTE	3	CD	C	<u> A</u>	L	L		P	R	0	P	G	add proportional term
MICROCOMPU	4	E O		<u> </u>	<u> </u>	ļ	ļ						(HL) - (HL) + K*(BC)
000	. 5	8/	 	<u> </u>	ļ_		<u> </u>			ļ			
CRC	6	CD	C	A	L	L		<u>L</u>	\mathcal{D}	T	1		Load Times I with
ž	7	60	 -	 _	ļ		 		_			_	new pulse with
	8	81	1		<u>_</u>	_		ļ					
	9	E 1	$\frac{P}{P}$	0 E	1	ļ	H						Lecover display data
S	A	C9		E	T	ļ	ļ			<u> </u>	-		
TER	В			<u> </u>	ļ	-	ļ			—			
INTEGRATED COMPUTER SYSTEMS	C		+ +		-	-	ļ .	-	-	-	-		
 	D	+	+ +	-	-	-	-	_	_	_			
5	E		 	-	-	-	-			-	-	-	·
MP	F	++	+ +	_	\vdash			-			-		
S	8 0		+	-	-	-	-	_				-	
TEC	1		++	-	\vdash	_	-	-	-	-		-	
RA	2		+ +	-	-	-		-	_	-	-	-	
TEG	3		+ +	+	-	-	-			ļ	ļ	 	
Z	4		 	+-	<u> </u>	 	╁	-		ļ	ļ	├	
	5	++-	+	+-	\vdash	\vdash		-	-	-	-	 	
	6			┼-	-	-	┼-	-	<u> </u>	-	_		
	7	-	 	-	ļ	_	├	-	<u> </u>	 	_	<u> </u>	
	8				1			<u>L</u>	<u>L</u>	<u> </u>	<u> </u>	Щ.	Figure 6-28f

	A D D	R	_co	DΕ	PWM	_	SU	BR	TUC	INE	E I	NTE	EG	- 7	ÆR	SION 2
	8 / C	0	0	0		N	0	P	/	R	E	\mathcal{T}				Stored by keyboard entre
	<u></u>	1	E	\mathcal{B}		X	C	Н	G							(DE) - old integral
		2	6	9		Μ	0	V		L	_ 1	C				(L) Low liste error
⊢		3	3	A		L	D	A			8	3	A	8		(A) - n sov
SHEET		4	A	8												Erron/2/w
		5	8	3												
CODING		6	6	7			0	V		14		A				(H) -n
00	810	7	2	5		<u>ک</u>	2	R		H						Decrement count
		8	F	A		フ	M	エ	N		8	/	D	5		Exit loop after O
		9	D	5			_									,
		Α	8	4												
		В	7	8		M	0	V		A		\mathcal{B}	_			Set CY if error
_		С	/	F		R	A	R				_				Set CY if error negative
TEN		D	7	\mathcal{D}		M	0	V		A	,	L	_	_		Shift low by to
SYSTEM	·	E	/	F		R	A	R								error right
		F	6	F		M	0	V		L		A				0
Z	8 /D	0	B	7		0	R	A		A						Test for error = 0
TRAINING	_	1	\mathcal{C}	2		7	Ν	Z	_		8	/	C	7		Loopunless
ER T		2	C	7												shifted error = 0
JTE		3	8													If shifted error=0
MICROCOMPUT		4	2	C		I	N	R		上						make it 01
00	810	5	6	0		М	0	V	_	\mathcal{H}	,_	В				(H) = high lugte error
SRC		6	/	9		\mathcal{D}	A	\mathcal{D}		\mathfrak{D}						(HL) - new integral Test for negative
ME	i	7	7	C		Μ	0	V		A	,	H				Test for negative
•		8	B	7		0	R	A	_	A						integral
		9	F	8		R	Μ			U	S					Exit if negative Store new integral
S		<u> </u>	2	2		S	H	<u>L</u>	D		8	3	A	4		Store new integral
SYSTEMS		В	A	4	_						_					<i>b</i>
YS		C	8	3	-	_			_							
		D	C	9		R	E	7	_						_	
INTEGRATED COMPUTER		E	0	0		N	-	P	_							
MP		F	0	0		N	0	P								
8	8	0	-	<u> </u>	L	_	ļ.,	_	_		_		_		, .	
LED	<u> </u>	1	_			E	N		E	R		ω	I	\mathcal{I}	14	
RA		2	_			_	(13	C	$\overline{)}$		=	+		E	PROR (B=00 OR FF) D INTEGRAL
reg		3	-				(14	4)		=		0	<u></u>	D INTEGRAL
Z	_	4		_		R	E	I	0	R	N	1	_	_	_	50.158
		5	-		<u>.</u>		Ļ	B		7		P	R		6	ERVED
		6		_		-	(D	E)	=	0	<u>L</u>	D	_	INTEGRAL
		7	-	-		<u> </u>	L	1/1	1-	7	=	N	E_	W	_	INTEGRAL
		8	1	L					L	}		L_				Figure 6-28g

PWM - SUBROUTINE PROPG 83A9 3 A for Kerron 9 328 CR MIN AD MP A CODING SHEET U S 93 B CE Ē 3 8 6 3 7 8 9 Α В T E R B C) # L) R WITH С MICROCOMPUTER TRAINING SYSTEM ERROR D PULSE WIDTH UPDATED BY INTEG E F 0 TURN 1 PRESERVED PRESERVED NEW PULSE WIDTH 2 Ш 3 4 NTEG + K ERROR 5 6 7 8 9 A INTEGRATED COMPUTER SYSTEMS В С D Ε F 0 1 2 3 4 5 6 7 Figure 6-28h 8

6.3.4 Experiments with PI Control

The effect of proportional plus integral control is to achieve rapid response to a disturbance or to a change in the desired output value without the objectionable overshoot and oscillation associated with pure integral control. If an oscilloscope is available the observations are easily made; lacking an oscilloscope the data can be logged and plotted. Waveform photographs are presented here for several of the experiments.

6.3.4.1 Open Loop Control

To demonstrate that open loop control is still available in the system with version 2 of CLOSL and INTEG, enter the following data and commands:

400, RUN Set 0.5 ms total period

MEM Set zero proportional gain

BRK Disable integral control

40.NEXT Set minimum output

C8, NEXT Request 2.0 volts

The voltage will rise to its initial value of about 1.5 volts.

Adjust the SENSE pot and observe that the voltage changes.

Restore the SENSE pot to the full left position.

6.3.4.2 Pure Proportional Control

Set pure proportional control by:

400, RUN	Set 0.5 ms total period
MEM	Set zero proportional gain
BRK	Disable integral control
64, NEXT	Request 1.0 volts
100, MEM	Set proportional gain = 1

The output voltage will rise somewhat as proportional control increases the charging time above the fixed value calculated by NEXT.

Observe the voltage generated with different proportional gains.

COMMAND	GAIN	OUTPUT VOLTAGE
MEM	0	
100,MEM	1	
200, MEM	2	
300, MEM	3	
400, MEM	4	
600,MEM	6	
800, MEM	8	
1000, MEM	16	

At a sufficiently high gain the multiplied error signal will lead to an unacceptable pulse width which will be rejected by LDT1.

RESPONSE WITH PROPORTIONAL CONTROL

Gain = 4

Proportional

Gain = 8

Proportional

FIGURE 6-29

If an oscilloscope is available, do the following experiment. Set the oscilloscope to 0.2 volts/division, 5 milliseconds per division, to observe the rise time.

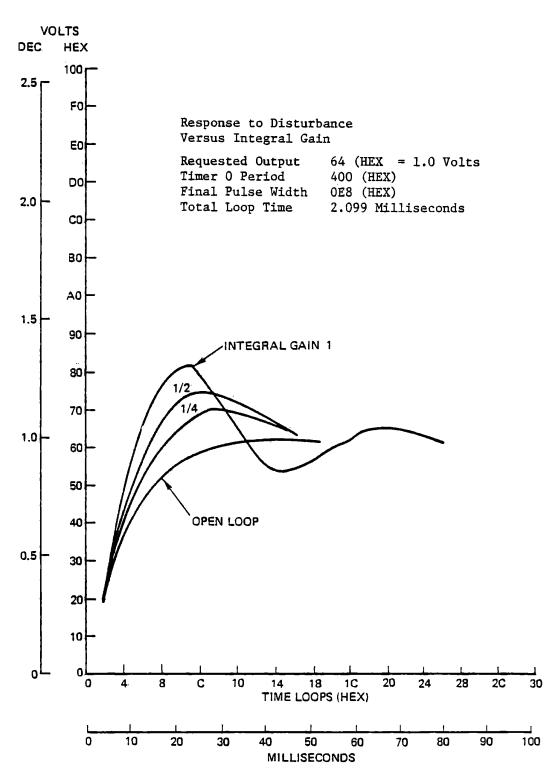
64, NEXT	Request 1.0 volt
MEM	Set integral gain = 1
	and proportional gain = 0
CLR	Set closed loop to adjust the pulse width
BRK	Set open loop
REG	Force output low.

Observe the response to the disturbance.

100, MEM Set proportional gain = 1

REG Force output low

Repeat with successively higher proportional gain values to observe the response. Figure 6-29 shows several response waveforms. The speed of response increases as the proportional gain is increased. With high gain substantial overshoot and oscillation appear.



Response Versus Integral Gain
Figure 6-30

6.3.4.3 Pure Integral Control

The effect of integral control on the response waveform has been observed previously. The following experiments shows the effect of reduced integral gain. If you have an oscilloscope make all of these observations. Otherwise, plot the response from the data log for gain = 1 and gain = 1/4. Enter these commands:

MEM Set proportional gain = 0

and n = 0 for integral gain = 1

CLR Enable integral control

64, NEXT Request 1.0 volt

REG Force output low

Observe or plot the response.

1, MEM Set integral gain = 1/2

REG Observe response

2, MEM Set integral gain = 1/4

REG Observe or plot response

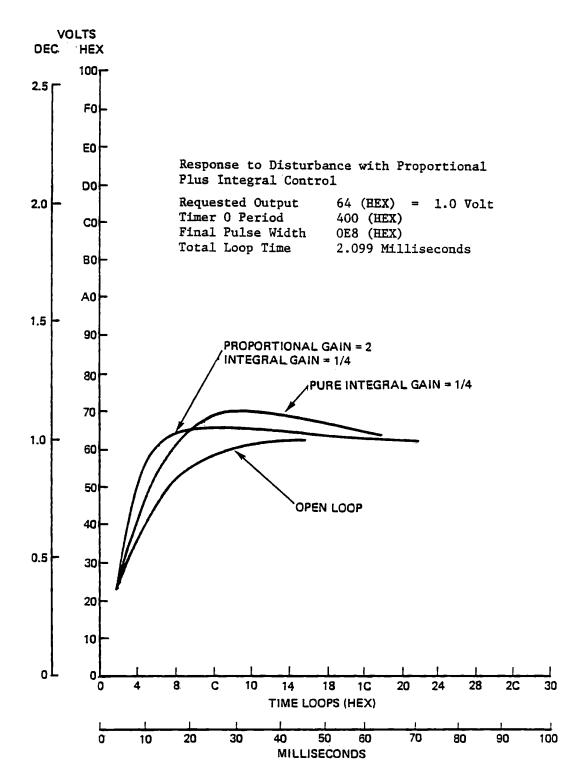
3, MEM Set integral gain = 1/8

REG Observe response

BRK Set open loop

REG Observe response

Continue to decrease the gain and observe the response. Figure 6-30 shows responses for selected gains, and for open loop operation.



Proportional Plus Integral Response
Figure 6-31

6.3.4.4 Proportional Plus Integral Control

Directly after the preceding experiment observe or plot the response with proportional plus integral control.

202, MEM Set proportional gain = 2

and integral gain = 1/4

REG Observe or plot response

This demonstrates an effective control system. Less overshoot occurs than with pure integral control using the same gain because the higher gain proportional control promptly corrects the overshoot. A fast response to the disturbance is observed. Figure 6-31 compares the response obtained here with some of our earlier results.

6.3.4.5 Response to Voltage Request

The preceding observations of response time have all maintained a constant desired voltage, forcing the output low under open loop control. Observe the response to a change in desired voltage:

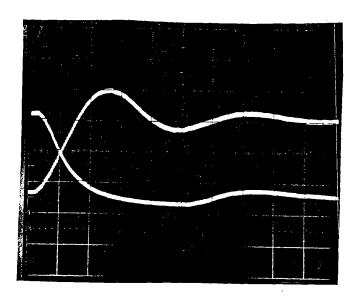
BRK, MEM	Set open loop operation
C8, NEXT	Request 2.0 volts
xxx,RUN	Set total period to obtain requested voltage
40, NEXT	Request minimum output and observe response
C8, NEXT	Request 2.0 volts and observe output

INCREASING PROPORTIONAL GAIN

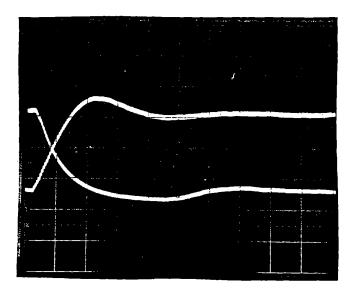
Proportional Gain	=	0
Integral Gain	=	1

Voltage	was	C8
Voltage	requested	40

Voltage	was	40
Voltage	romineted	C8



Proportional Gain = 1
Integral Gain = 1

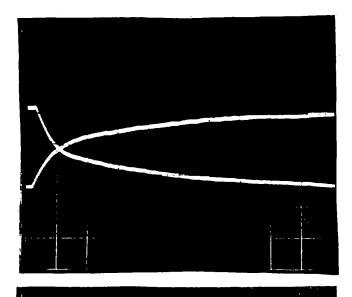


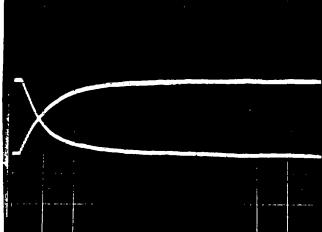
Response to Voltage Request
Figure 6-32a

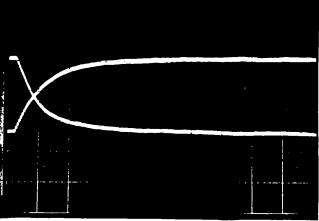
INCREASING INTEGRAL GAIN

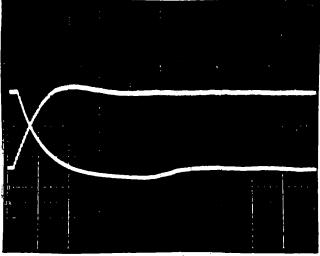
Proportional Gain = 2

Integral Gain









Proportional Gain = 2

Integral Gain

Proportional Gain = 2

Integral Gain

Response to Voltage Request (continued)

Figure 6-32b

CLOSED LOOP CONTROL

Now set proportional plus integral control by:

202, MEM	Set proportional gain =2
	and integral gain = $1/4$
CLR	Enable integral control
50,STEP	Request minimum output
	and observe response
C8,STEP	Request 2.0 volts
	and observe response

Results of this test are shown in Figure 6-32. Note that with open loop control the rise and fall are similar, but with closed loop control they are distinctly different.

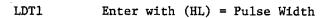
6.3.5 Full Scale Control and Overflow

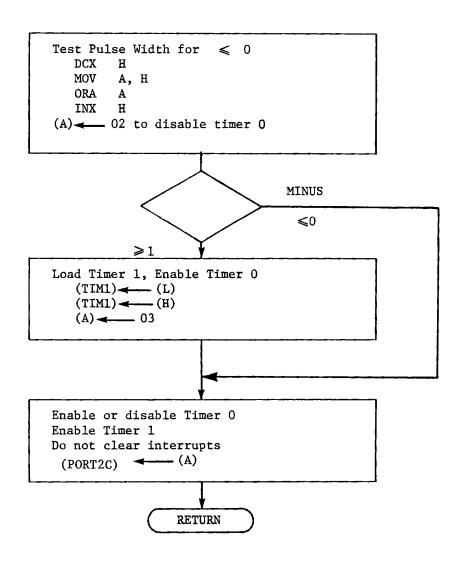
In subroutine LDT1 we have protected against loading Timer 1 with an unintended long time period when the calculated pulse width is zero or negative. The reaction when this occurs is to leave the preceding value of the pulse width unchanged. This has been acceptable but is not the best arrangement. It would be better to recognize the condition and disable charging altogether if the measured voltage is so much greater than the desired value that proportional control cannot operate correctly. This would allow the voltage to drop more quickly, and also remove the limitation imposed by the time taken in the interrupt service routines.

(Recall that Timer 0 interrupt starts charging the capacitor by setting Port 1A7 high, and no matter how small a value is loaded to Timer 1 the output will remain high until the Timer 0 interrupt is

finished and the Timer 2 interrupt sets the output low. This takes 66 clock periods after OUT PORT1A in Timer 0 service, plus 50 clock periods for the Timer 1 interrupt and processing through its OUT PORT1A. This gives a minimum pulse of 57 microseconds and an output of 0.8 volts if the total period is 300 hex).

When we abandon proportional control and set a minimum (or maximum) control force we are employing "full scale control". We can do this here by changing subroutine LDT1.





LDT1 with Full Scale Control
Figure 6-33

6.3.5.1 LDT1 with Full Scale Control

Revise subroutine LDT1 according to Figure 6-33. With this program the calculated pulse width is tested as before for a negative or zero value. If the width is greater than zero it is loaded into Timer 1 and the interrupt from Timer 0 is enabled. If the width is zero or negative the timer is not loaded, and Timer 0 is disabled so that no charging occurs. Note that the enable or disable is not to clear the interrupt, so it is done by writing to Port 2C, not to CNT2. If the bit set function were used to enable the interrupt it would occasionally occur just as the timer generated an interrupt, thereby inhibiting a charging pulse. With the method of writing to Port 2C, normal operation will be totally unaffected, since this does not reset the interrupt source flip flops. After a time of full scale control with Timer O disabled, the voltage will decrease, the calculated pulse width will again become acceptable and Port 2CO will be set high. An interrupt will occur immediately. A charging pulse will start, but its duration will be random, depending on when the next Timer 1 interrupt occurs.

	A D D	R_	со	DE	PWM	_	SU	BRO	TUC	INE	L	DT1	i –	VI	ERS	ION 2
	8 /6	0	2	\mathcal{B}		D	C	X		Н						Lest calculated,
HEET		1	7	C		M	0	V		A		H				pulse width for
		2	B	7		0	R	A		A						zero or negative
		3	2	3		\perp	N	X		\mathcal{H}						
		4	3	E		M	V	I		A		0	8			Load (A) to disable
3 SH		5	0	2												Timer O interrupt
SODING	·	6	F	A		J	Μ	I	N		8	/	7	/		
100		7	7	1												and jump if pulse width = 0
		8	8	7												
		9	7	2		Μ	0	V		A		L				Pulse width >0
	· · · · · · · · · · · · · · · · · · ·	A	0	3		0	U	T		7	I	M	1			Load Times I for
		В	7	5												new pulse width
Į		С	7	C		M	-	V		A	_	H				
SYSTEM	·	D	1	3		0	U	T		T	I	Μ	1			
.XS		E	1	5												
		F	3	E		M	V	I		A	,	0	3			Load (A) to enable
Z	8 /7	0	0	3												Timer Ointerrupt Enable or disable
TRAINING	817	1	D	3		0	V	7		P	0	R	T	2	C	
Œ		2	0	E												Times O interrupt
JTE	-	3	C	9		R	E	7						_		and enable
MICROCOMPUT		4									i					Timer / interrupt
000		5														
CRC		6	<u> </u>											_		
Z		7				R			エ	S	E	\mathcal{D}		L		T
		8	-			F	0	R		F	U	L	L		3	CALE CONTROL
		9	-			_	. ,		1	1			_		-	
S		Α	├			E	N	I		R		ω	I	-	H	
TEN.		В						H	L	ر		i.		2		LCULATED
.XS		C	-	<u> </u>		1					_		-	P	$\hat{\Omega}$	LSE WIDTH
E		D	—			7	上		(7	L)	_		<u> </u>	ZERO
5		E	_			_		0		\mathcal{D}		T	I	W	E	RIAND
MP		F	!				E	N	H	\mathcal{B}	_	E		13	0	TH INTERRUPTS
NTEGRATED COMPUTER SYSTEMS	8	0	-	_					_	11	_	_	_	_	7	7.50
TEC		1	⊢	-	-	ㅗ	7 1	_	-	1+)	_	-		ZERO
RA		2	ļ .		-		D	エ	5	A	B		E		T	IMER O TO
TEG		3	\vdash	-			I	N	}+	I	B	7	T	-	C	HARGING
Z		4	\vdash	\vdash		\vdash		ļ	-		_	<u> </u>	_	-	_	
		5	\vdash			<u> </u>	-				-		-	-	-	
	ļ _ _	7	-		 	<u> </u>	 		-	-		-	-	-		
	<u> </u>		\vdash	-	-	<u> </u>	-	-	_		-	_	-	-	-	Figure 6-24
		8	<u> </u>	Ц	<u> </u>	<u> </u>	Ľ	l				L.	<u> </u>	<u> </u>		Figure 6-34

6.3.5.2 Full Scale Control Experiments

To test the ability of full scale control to generate a low output signal enter:

400, RUN Set 0.5 ms total period

MEM Set proportional gain = 0

and integral gain = 1

CLR Enable integral control

STEP (or NEXT) Request zero output

Timer 0 interrupt will be disabled, so Port 1A7 will be continuously low. (Observe this in the LED). The output voltage will be determined by the division of Vg across the voltmeter resistance and the 10K resistor R2.

$$v = \frac{v_{g}R_{m}}{R_{m} + R_{2}}$$

Note that the voltage can be changed by switching scales on the voltmeter (which changes the voltmeter's resistance). If you disconnect the voltmeter the voltage measured by the A/D converter will be almost exactly equal to Vg, the zero offset voltage of the open collector driver of the output port. The voltmeter will reduce that voltage slightly if a 3 volt scale is used and significantly on more sensitive scales. (If you are using an electronic voltmeter with a high impedance input the voltmeter will not affect the output voltage at all).

CLOSED LOOP CONTROL

Now, reset the computer and check the integral stored at (83A4,A5). It will be some value between zero and the output voltage that was achieved. After this value was stored, INTEG attempted to reduce it by the observed error. The result was negative so it was not stored. A lower integral gain would allow it to be reduced further. The negative (or zero) pulse width passed to LDT1 gave full scale control and a minimum output.

Run the program again and enter:

MEM Set proportional gain = 0

and integral gain = 1

CLR Enable integral control

STEP Request zero output

BRK Disable integral control

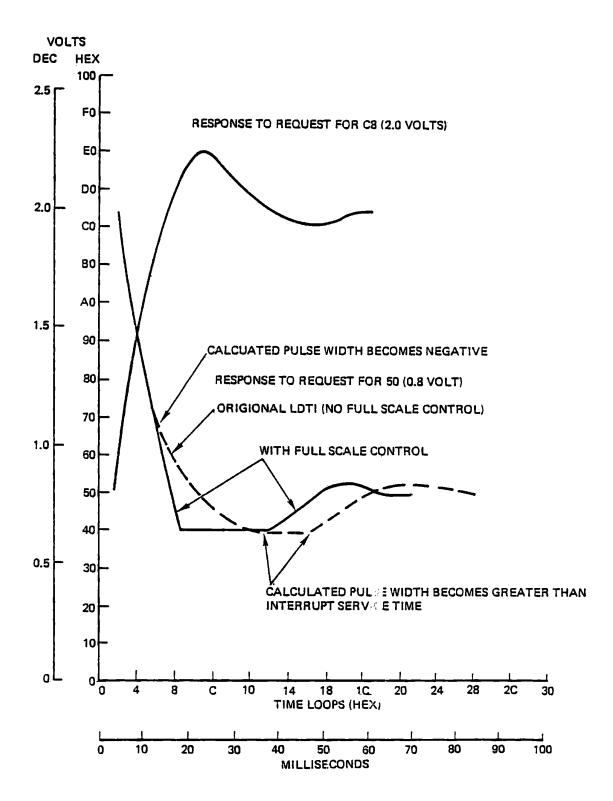
The voltage will rise to the minimum determined by the interrupt service processing. Without closed loop control no attempt is made to reduce the pulse width below zero, so LDT1 will enable Timer 0 interrupt. Enter:

100, MEM Set proportional gain = 1

Now the proportional control will generate negative pulse widths and full scale control will be invoked.

Observe the response to voltage requests with various proportional and integral gains. Figure 6-35 compares the response to requests for 50 (hex) and C8, with the original version of LDT1 and the new version with full scale control. For this plot both gains are set to unity.

100,MEM	Set proportional gain = 1					
	and integral gain = 1					
CLR	Enable integral control					
C8,STEP	Request 2.0 volts					
50,STEP	Request 0.8 volt and observe response					
C8,STEP	Request 2.0 volts and observe response					



Full Scale Response to Voltage Request
Figure 6-35

6.3.5.3 Maximum Output Full Scale Control

Clearly full scale control could also be exercised in the other direction. If proportional control could not achieve a desired voltage or demanded too great a pulse width, Timer 1 could be disabled to leave Port 1A7 on continuously. In fact, the present integral control system has full scale control, since the charging pulse width can be increased up to 7FFF, about 16 milliseconds or 32 times the 0.5 millisecond "total period". If the Timer 1 period is greater than the Timer 0 period, each new output pulse from Timer 0 retriggers Timer 1, whose count is then reloaded automatically and never reaches zero. No Timer 1 interrupts are generated and Port 1A7 stays high.

Full scale control is very commonly employed in proportional control systems where the designer recognizes a need for signals outside of the proportional range. Especially in pure proportional systems (i.e., no integral control) it is often desirable to use very high proportional gain to obtain fast response to small error signals. This usually limits the proportional range to fairly small error signals and demands full scale control for large errors.

6.3.5.4 Integral Overflow

We have limited the integral of error signals stored by INTEG to the range 0000 to 7FFF. This limitation tends to distort the results, since the lower limit is often encountered, but it is almost impossible to reach the upper limit. We are also wasting half of the range of a two byte variable. If this were important (which it is

CLOSED LOOP CONTROL

not) we could extend the range by allowing and correctly processing negative values of the integral.

At present INTEG refrains from storing the integral if it is negative. This applies the limits of 0000 to 7FFF. If this test is not applied the integral will temporarily go negative when the desired voltage is switched from a high value too a low value, giving a large negative error signal. This invokes full scale control. Provided that the measured voltage eventually becomes less than the desired voltage, giving a positive error signal, the integral will (after some time) become positive again and settle at a value that gives the correct pulse width.

To experiment with this, remove the following instructions from CLOSL and INTEG. (Replace each of them with NOP).

8186	E5	PUSH H	Save display data
8199	E1	POP H	Recover display data
81D9	F8	RM	Exit if integral <0

Removing PUSH H and POP H will cause CLOSL to return the calculated pulse width for display instead of the error signal and desired voltage. Removing RM will cause INTEG to store the integral regardless of its value. Now run the program with the following data and commands. (We will use pure integral control so that the displayed pulse width will be the integral value).

MEM Set proportional gain = 0

and integral gain = 1

CLR Enable integral control

FA, STEP Request 2.5 volts

40,STEP Request 0.64 volt

The two voltage requests may be repeated as often as necessary. After 40,STEP you will see an F appear momentarily in the left hand display digit, showing that the calculated pulse width has become negative. Thereafter the integral will become and remain positive.

20,STEP Request 0.32 volt

To reach this low a voltage full scale control must be invoked a large part of the time. The integral displayed at the left will show FFxx with an occasional appearance of OOxx.

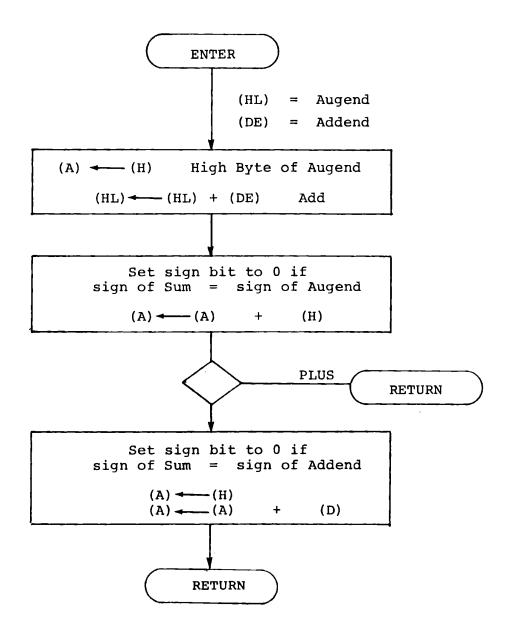
STEP Request zero output

Since a zero volt output cannot be achieved even with full scale control the error signal will always be negative. The integral will be repeatedly reduced from FFxx down to 80xx, then to 7Fxx where it suddenly appears to be a large positive value. Until this point, full scale control has kept Timer 0 interrupt disabled and Port 1A7 low. Now Timer 1 is loaded with a long pulse width (about 16 milliseconds) and Timer 0 is enabled. Port 1A7 is set high and the output rises above 2.55 volts. The large negative error signal is calculated, rapidly reducing the integral until it goes negative again. You can observe the peculiar behavior on the voltmeter or at

the LED for Port 1A7. A computer, like a person, may go crazy when presented with an impossible task and no escape mechanism. The program works well as long as it is able to achieve its objective, but it needs the protection of the RM instruction for the impossible request.

Restore the three instructions that were deleted for this experiment.

Recognize from this experiment that a negative integral is perfectly acceptable; the fault occurs when a negative error added to a negative integral generates a positive result. This is referred to as arithmetic overflow, and can be detected. The result of an addition should always have the same sign as either the augend or the addend. If both are positive but their sum is negative, or vice versa, overflow has occurred. Figure 6-36 shows a simple procedure for testing the result of a double precision add. Replace DAD D and the instructions that test for negative results in INTEG with CALL ADTOV (81FO). Enter subroutine ADTOV (Figure 6-37). Now either positive or negative integrals will be stored unless overflow has occurred. There will be a difference in the response to large changes in the requested voltage, and the program is more satisfying aesthetically, but you will see that the difference is not important.



Subroutine ADTOV

Double Precision Add and Test for Overflow

Figure 6-36

	<u> A D D</u>	R_	со	DE	PWM	-	SU	BRO	TUC	INE	I	NTE	G	- V	/ER	SION 3
	8 /C	0	0	0		N	0	P	/	\mathcal{R}	E	7				
		1	E	\mathcal{B}		X	C	1+	G							(DE) - old integral
		2	6	9		Μ	0	V		L		C				(L) + low luste error
-		3	3	A		L	D	Ā			8	3	A	8		(A) - n Low
SHEE		4	A	8												(L) + low lugte error (A) + n for Error/2"
		5	8	3												
CODING		6	6	7		M	0	V		1+		A				(H) - N
300	81C	7	2	رح		$\overline{\mathcal{D}}$	C	R		Н						Decrement count
Ū		8	F	A		7	M	I	7		8	/	D	3		Exit loop after O
		9	2	5			_									
		Α	8	/												
		В	7	8		M	0	V		A		B				Set CY if error
_		c	1	F		R	M	R			, 			_		negative
SYSTEM		D	7	D		M	0	٧		17	,	L				Set CY if error negative Shift low byte of error right
.XS		E	/	E		\mathcal{R}	A	R			_					error right
		F	6	F		M	0	<u>/</u>		L	,	A				0
TRAINING	8 1 2	0	B	7		0	_	A		A						Lest for error = 0
RAI		1	C	2		7	N	Z			8	/	C	7		Loop unless O
œ		2	C	7												, , , , , , , , , , , , , , , , , , ,
JTE		3	8												-	Hshifted error=0 Make error = 01
MICROCOMPUT		4	2	C	L	I	N	_		<u>L</u>		_				
000	810	5	6	0		M		V		14	<u>, </u>	B	_			(H)←high byte error add (HL)←(HL)+(DE)
CRC		6	C	Σ		C	A	L	L		À	D	T	0	٧	
⋛	ļ	7	F	0												Return MINUS if
		8	8	6		1		_			(_			arithmetic overflow
	ļ		F	8		K			1/2	U_)]	.1	,,		0
JS.		A	2	2		S	1+	L	\mathcal{D}		8	3	H	4		store new integral
SYSTEMS		В	A	4								-				
SYS		C	8	39		D	_	T					_			
		D	\vdash			$\frac{R}{M}$		7	-	-						
Ť۷		E F	0	0		$\frac{N}{N}$	0	P								
M	3	· 0	0	0		10		Γ		-	-		_			
Ö		1		-		E	2	T	=	R			+	Τ	7	
NTEGRATED COMPUTER		2				_	10	1 B	E	1		=	1	+	1.	ERROR (B=00/FF)
GR/		3	-				-	<u>n</u>	_	7	-	=	-	0	1	D INTEGRAL
Ē		4		-		R	E	17	$\frac{2}{0}$	R	N	_			-	D TIMER KAP
=		5			-	 	7	B	C	7	, *	P	R	E	5	ERVED
		6					(D	E	5	-	=	'``		٦	D INTEGRAL
		7			 		7	H	_	5	_	=				W INTEGRAL
		8	T.	-				 							_	Figure 6-37a
	C 162				<u> </u>											· · · · · · · · · · · · · · · · · · ·

	A D D	R	CO		ADT			ועה	, A.	.,	<u> </u>				T 11.	
	8 /F	0	7	09		M	0	V		A		H				
		1	/	9		\mathcal{D}	A	D		\mathcal{D}	_					
		2	A	\cdot C		X R	12 12 D	Α		H S						
		3	F	0		R	Ω	L	U	S						
E		4	7	C		M	0	V		A		\mathcal{H}				
Ϋ́		5	A	A		X	R	A		\mathcal{D}	7-					
N		6	C	9		X	E	7								
CODING SHEET		7										_				
		8														
		9				E	Ŋ	T	E	R						
		A					(D	ഥ	7	=		0	T	a	INTEGR AL
		В					(1+	L)	=		E	R	α	INTEGRAL OR /2"
		С														
EM		D				Ŋ	Ш	T	S	R	N					
MICROCOMPUTER TRAINING SYSTEM		E					1	H	٦)	=		N	E	W	INTEGRAL
G S		F				_							(D	E	INTEGRAL) + (HL)
Ž	8	0					M	I	Ν	U	S		F	L	A	G SET IF
\AII	1	1						R	I	T	4	M	E	T	H	C OVERFLOW
₹ TF		2														
TEF		3														
ΛΡU		4														
SON		5														
RO		6														
MIC		7														
		8														
		9														
(0		Α														
SYSTEMS		В														
/ST		С														
R S		D														
INTEGRATED COMPUTER		E														
MPL		F														
S	8	0														
ED		1														
3AT		2														
EGF		3														
INT		4														
		5														
		6	_													
	<u> </u>	7														
		8	_	_												Figure 6-37b

6.4 PROPORTIONAL - INTEGRAL - DIFFERENTIAL CONTROL

Consider a system with substantial inertia, such as steering a ship aircraft. When the aiming point is changed or a sudden disturbance such as a wind gust or wave causes a large error signal, the proportional term in the control equation generates a large (possibly full scale) control force. The inertia prevents an instantaneous response, so after a brief time this force is further increased by the integral term. Now as the ship starts to respond the proportional term GpE decreases but the integral term Gi \int E is still increasing. The ship now swings toward the aiming point, and its inertia will carry it beyond the desired point. An error in the opposite direction appears, the proportional term in the control equation becomes negative, and eventually the ship settles on its new course, provided the control system is stable. There may be significant and undesirable oscillations before the new course is achieved if high gains are used in the control system, and it is possible to have unstable operation where the oscillations are maintained indefinitely.

Differential or rate control can be applied to detect and respond to the fact that the ship is approaching the desired aiming point. As it begins to turn, even though the error signal may still be substantial, it is observed that the error is decreasing. This implies that a smaller control force should now be applied, or even that the control force should be reversed to overcome the ship's inertia. The control equation becomes:

$$F = G_p E + G_i \int E + G_d \Delta E$$

The differential term is not limited to reducing the control force as the error decreases, but also adds to the control force when the error is observed to increase. In a system subjected to disturbances the differential term may dominate the result, maintaining such small errors that the proportional term is generally very small and the integral term is almost constant.

Applying the differential control to a system having as little inertia as the capacitance in our PWM voltage control problem has very little effect. For the student interested in pursuing Proportional-Integral-Differential Control (PID) it is suggested that the filtered voltage returned by FILTR be used as the input to the closed loop control equations. Here FILTR gives the effect of a system with large inertia.

6.5 SUMMARY

In the preceding sections the most important concepts of feedback control systems have been demonstrated. Although controlling the voltage on a capacitor is a trivial and perhaps unexciting example, it gave us an easy way to observe and measure the behavior of the system. We have seen the response to a disturbance with proportional control; the need for integral control to provide a steady state force when external conditions are variable; and some of the problems such as overshoot, oscillation, and arithmetic overflow. We have also seen the use of the microcomputer to monitor its own performance. Chapter 7 will apply the same principles to control of a motor.

MICROCOMPUTER INTERFACING WORKBOOK

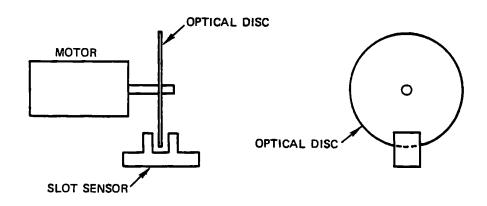
CHAPTER 7

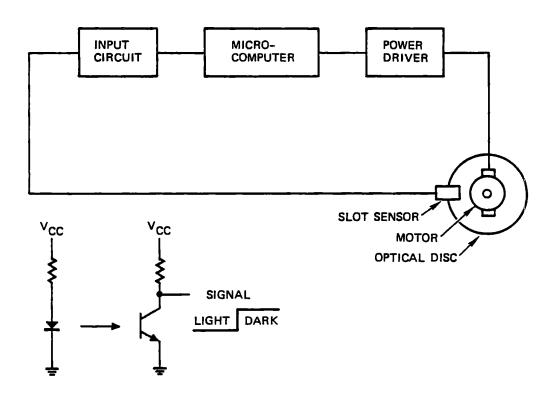
MOTOR CONTROL

This page intentionally left blank.

7. MOTOR CONTROL

Power to a dc motor is to be controlled to set its speed. We will develop a technique to measure speed, using open loop control of power by pulse width modulation; then we will close the loop using proportional plus integral control. Both the program itself and our development of it will resemble the pulse width modulated voltage control of Chapter 6.





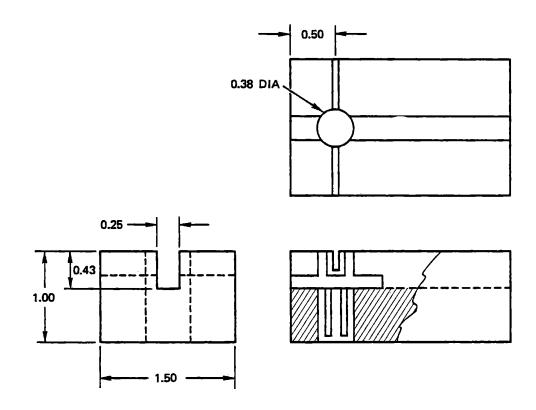
Motor and Slot Sensor
Figure 7-1

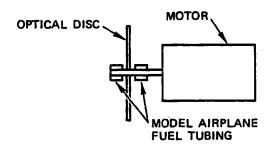
7.1 OPTICAL DISC AND SLOT SENSOR

Motor speed will be measured by observing an optical disc with transparent and opaque segments rotating between a light emitting divide and a phototransistor. Figure 7-1 suggests the physical arrangement and the system block diagram.

The "slot sensor" contains an infrared light emitting diode aimed at a phototransistor across a gap of 0.1 inch. When power is applied to the LED and the phototransistor as shown in Figure 7-1, the infrared light falling on the base of the phototransistor turns it on just as base current would in a normal transistor. The transistor current then drives the output signal low. When the light is blocked the phototransistor is turned off and the output signal becomes high.

If your configuration includes the Integrated Experiment Assembly, the motor and slot sensor are mounted and connected, with an amplifier for the slot sensor signal. In this case the connections and tests described in Sections 7.1.1 and 7.1.2 are not required. Read the material of Section 7.1.3, but refer to the "Selected Experiments" manual for making connections.





Motor, Sensor and Disc Mounting
Figure 7-2

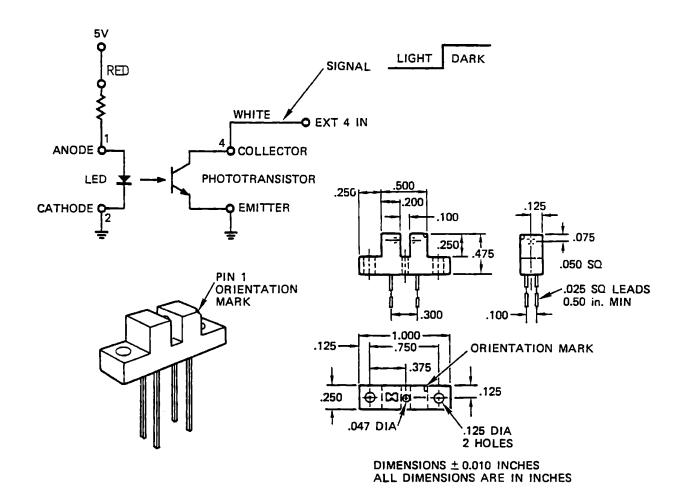
7.1.1 Motor, Sensor and Disc Mounting

The motor and slot sensor can be mounted on a block of styrofoam or balsa wood (Figure 7-2). Drill a 3/8 diameter hole through the block for the slot sensor leads. Cut a slot wide enough to grip the sensor (.25 inch) and deep enough to make the top of the sensor flush with the top of the block. This need not be at all precise.

Now cut a narrow slot across the width of the block to accept and guide the optical disc through the slot sensor opening.

Mount the optical disc on the motor in either of two ways. You can cut a small X at the center of the disc and force the motor shaft through that hole. The springy mylar will grip the shaft. If you find that the disc slips on the shaft, a small piece of tubing can be placed on the shaft on each side of the disc and squeezed together to grip the disc.

If you have cut the slot to fit the sensor closely it will hold it with no other mounting. If it is too loose, build up the projecting ends of the sensor with Scotch tape. The motor can be held in place on top of the block with tape or rubber bands.



		CABLE	
	PIN FUNCTION	COLOR	CONNECTION
1	LED ANODE	RED	+5 VOLTS
2	LED CATHODE	BLACK	GROUND
3	EMITTER	ORANGE	GROUND
4	COLLECTOR	WHITE	EXT 4 IN

Optical Slot Sensor
Figure 7-3a

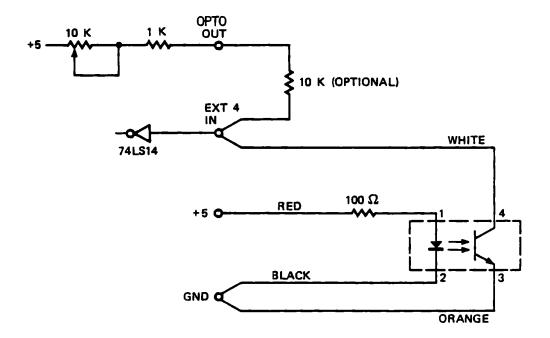


Figure 7-3b

7.1.2 Slot Sensor Connection and Adjustment

The light emitting diode and phototransistor of the slot sensor are shown in Figure 7-3a. The pin numbers and color codes of the cable are shown. Note that a 100 ohm resistor is built into the red lead of the cable, so no external resistor is needed. Check that the cable is wired correctly, and plug the cable ends into the tie blocks as indicated in Figure 7-3b. Be very careful about these connections, because the slot sensor can be damaged or destroyed by reverse voltages.

Connect a 10K ohm resistor between EXT 4 and OPTO OUT. Together with the OPTO SENSE pot this provides a pullup resistance from EXT 4 to ground.

Now when a clear segment of the disc lies in the light path of the slot sensor, infrared light from the LED falls on the phototransistor and pulls the output signal close to ground. When a dark segment interrupts the light the phototransistor turns off and the voltage is pulled up close to 5 volts. Observe this with the voltmeter.

The 74LS14 Schmitt trigger circuit at the EXT 4 input requires that the signal switch below 0.6 volts to guarantee correct operation. The slot sensor may not have enough gain to achieve this with the pullup resistance. If it does not, remove the connection to OPTO OUT. Now the phototransistor will pull the input down when a clear segment is observed. When a dark segment is present the internal pullup resistance of the Schmitt trigger will pull the input voltage up to about 1.2 volts and the dark segment will be recognized. The circuit

will operate correctly without the external pullup resistor, but will be more sensitive to noise pickup.

A test program is given in Figure 7-4. Connect EXT 4 to ANALOG IN in addition to the pullup resistor, and remove the voltmeter. The test program displays the state of the EXT 4 input in LED number 6, and uses the automatic A/D input to measure and display the voltage. With this program you can observe the switching and the actual voltage at the input as you rotate the optical disc and adjust the OPTO SENSE pot. Be sure that the voltage goes below 0.60 (3C hex) when a clear segment is observed. If it does not, remove the connection to OPTO OUT or substitute a higher valued resistor.

When a dark segment appears in the slot, the voltage must switch above 2.0 volts if an external pullup resistor is used. Without any pullup the Schmitt trigger should clamp the voltage at about 1.2 volts.

Be sure that the slot sensor operates correctly before going on with the program development. Unreliable operation of the detector will result in a useless control system.

	A D D	R	со	DF	TES?	r F	OR	SL	TO	SE	NSC	OR				
	8 20		3	E		M	V	I		A		8	2			Program Ports
		1	8	2												Port 1B In
		2	0	3		0	U	T	_	C	N	\mathcal{T}	1			
		3	0	7	_											
冒		4	3	Ë	_	M	V	I		A		9	2			
SHE		5	9	2							,					
CODING		6	0	3		1)	IJ	\mathcal{T}		C	N	7	2			
		7	0	I												
		8	3	E		М	1/	\mathcal{I}		A		9	4			Program and loan
		9	9	4			_			, ,		-				Grogram and load Jimes 2 for A/D input
		A	2	3		0	U	7		T	I	M	C	T		In A/D input
		В	1	7						,						Jac. 1
		С	3	E		М	V	I		A		2	0		_	
E		D	2	0				-			/		_			25 16.5
SYSTEM		E	1	3		0	U	T		7	I	M	2			
		F	1	6				Ė		,						
RAINING	82/	0	3	E		М	V.	ュ		A	,	0	3		-	Let Port 1C/ high
A		1	0	3				-			1	_				for auto A/D
\vdash		2	2	3		0	U	7		P	0	R	7	1	C	1 + Pa+ 100 1.
TER		3	0	6			Ť									to tube could
J.	821	4	3	E		М	V	I		A		0	8			to turn power transistor off
MICROCOMPUT		5	0	8			-				/					2
ROG		6	7	3		0	V	T		P	0	R	Т	2	C	Enable A/D interrupt without reset Read interrupt status byte
A C		7	0	E												without reset
	-	8	7	${\mathcal B}$		工	N			P	0	R	T	2	B	Pead interpret
		9	0	Ď								<u> </u>		1		status but
		Α	E	6		A	N	I		4	0					and display EXT 4 Input
MS		В	4	0												EXT 4 Spart
SYSTEMS		C	2	3		0	U	T		P	0	R	7	1	A	1940
3 SY		D	0	4												
TEF		E	C	3		5	M	P		8	2	/	4			Loop
IPU		F	1	4												
INTEGRATED COMPUTER	8	0	8	2												
) Q3		1														
AT		2														
EGR.		3	1							_						
N		4														
_		5														
		6														
		7														
		8														Figure 7-4a

Že ro i		D D R	co	DE	TES!	r F	OR	SL	TO	SE	NSC	OR	- 1	A/D	IN	NTERRUPT
	8	230	تح	5		P	U		H		_		W			
		1		\mathcal{B}		I	N			P	0	S R	$\overline{\mathcal{T}}$	/	\mathcal{B}	
	-	. 2		ریی	25.0									·		(L) = valtage
		3	6	F		M	0	V		7	-	A) =) /20
EET		4	3	E		M	V	$\overline{\mathcal{I}}$		A	,		6			(L) - voltage Reset
SH	-	5	0	6	3						1					
S	_	6	7	3		0	U	$\overline{\mathcal{T}}$		C	N	\mathcal{T}	2			
CODING SHEET	-	7 /	0	F												
Ö	l i	8	7	D		M	0	V		A		L				
		9	C	D		C	A	L	4		D	\mathcal{B}	У	T	E	
′	T.	A	9	5												
	┢╌	В	0	2			-			-						
		С	Ψ	/		P	0	P		P	5	ϑ				
≅		D	Ų	\mathcal{B}		E	I									
STI	<u> </u>	Ε	C	9		R	E	7								
SY	一	F														
MICROCOMPUTER TRAINING SYSTEM	В	0			_											
¥		1			_											
H.	Г	2				<u> </u>					_					
LER	┢	3														<u> </u>
P	-	4	L						_			_		_		
Ş.	┢	5					_				- 1					
မ္တ	Г	6 												Ŀ		
딜		7	_									.:	,	_		
=		- 8							L.,					_		
	Г	9				<u> </u>	<u> </u>		_					-		
		A	_	-		<u> </u>	_		L				_			
SIN:		- В	_			<u> </u>			ļ					_	<u> </u>	
STE	T	- с				_								_		
S	Т	- D	_			<u> </u>	_		ļ.,		<u> </u>	_				
TEH	Г	- <u>E</u>	-	-		 	ļ	_	_				_	_	_	
∑.	Г	- F	-	<u> </u>		├-		_	_		_	_		_		
Š	8	- 0	-	ļ.,		-	_		<u> </u>	ļ	<u> </u>	_		_		
9	厂	- 1	-	-		-			-	-	-			<u> </u>		
ATE	一	2	-	-		┡	<u> </u>	<u> </u>	ļ	<u> </u>		_		_	_	
GR	H	- 3	_			<u> </u>	<u> </u>	-	-				_	_	-	
INTEGRATED COMPUTER SYSTEMS	1	- 4	-	-	<u> </u>		-	<u> </u>	-	-	-	-	<u> </u>	-		
=	\vdash	5	-	-	-	-	-	\vdash	-	-	-	-		-		
		6		-		-	-	-			-	-	-	 	<u> </u>	
	T	7	_			-	_	ļ	-		-	-	-	-		Pigure 7-45
	H	- <u> 8 </u>	<u>L</u>	L_		L		L_	<u> </u>		<u> </u>	<u>L_</u>	<u> </u>	Ц.,		Figure 7-4b

7-11

9200

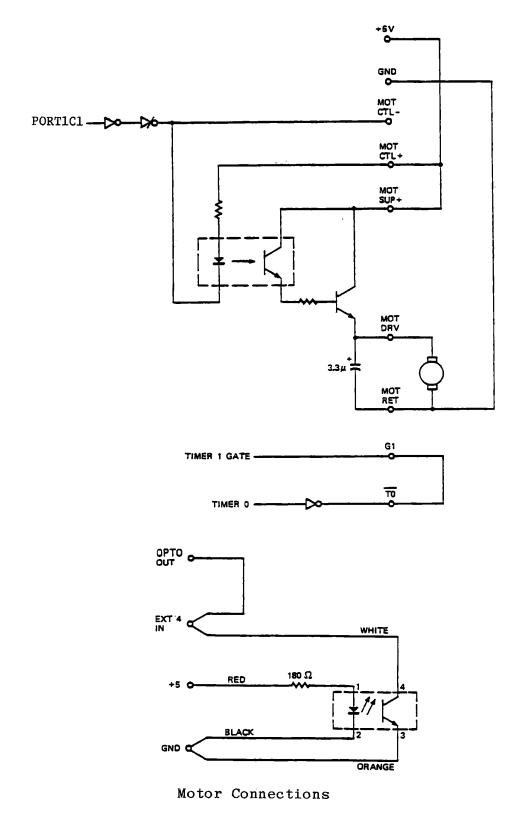


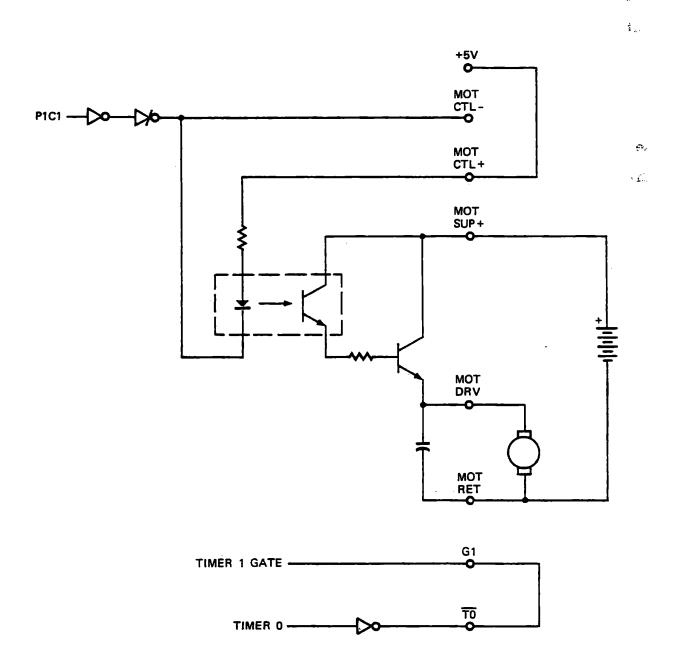
Figure 7-5

7.1.3 Motor Connection

The motor will be driven from the five volt supply using the power transistor as a switch for pulse width modulation control. Figure 7-5 shows the circuit connections for the motor and the slot sensor.

The power transistor and the optical coupler that drives it are isolated from the system power supplies to allow use of an external supply. For this experiment you can use the five volt system supply, although in general it is very poor design practice to place a noise generating load such as a motor on the computer's regulated supply. Figure 7-6 shows how the connections would be made with an external power source such as a lantern battery. Note that here there is no electrical connection between the motor and the computer.

Output PORTICI drives an inverter and an open collector inverter to control the optical coupler. When PICI is set low the open collector inverter draws current through the LED of the optical coupler. Infrared light from the LED turns on the phototransistor, which in turn provides base drive to the power transistor. This allows current to flow in the motor circuit either from the system power supply (as in Figure 7-5) or from the external supply (Figure 7-6).

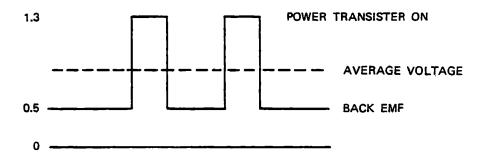


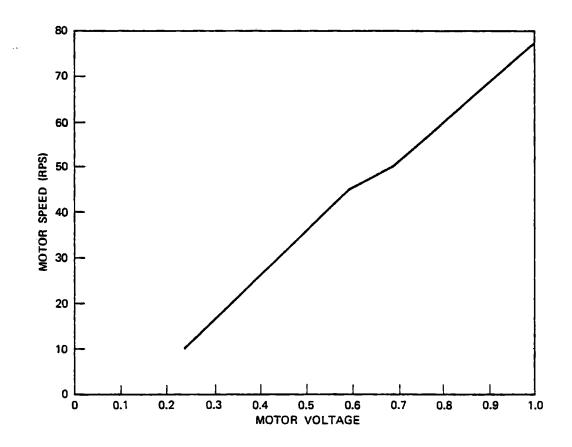
Ŋ

Motor Connections with External Power
Figure 7-6

Port 1C1 is set to input mode by system reset. In this condition it appears as a high input to the first inverter, so the open collector output is in the high impedance state and the optical coupler is switched off, so the motor does not run. When Port 1C is programmed for output during initialization its outputs become low, and power is applied to the motor. In general the initialization procedure should to the motor will not be turned on until intended.

Figures 7-5 and 7-6 also indicate that a connection is to be made from Timer 0 output to Timer 1 gate, for PWM control.



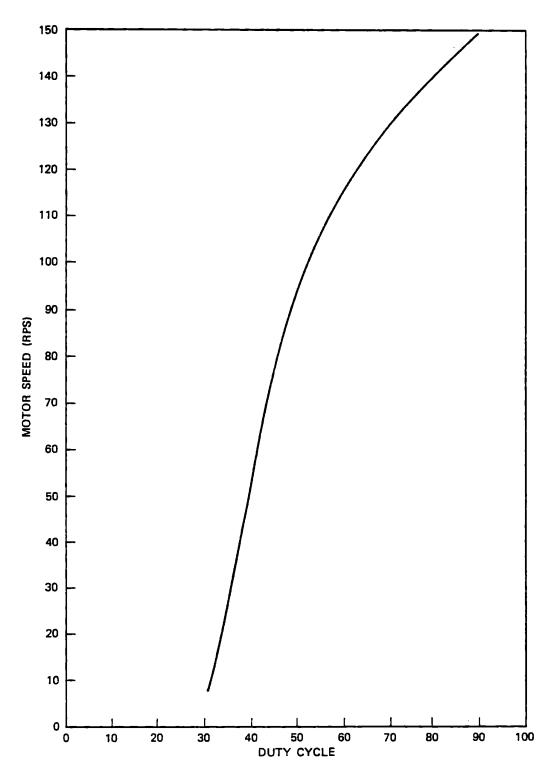


Motor Speed vs. Voltage
(Measured with closed loop control)
Figure 7-7

7.1.4 Motor Characteristics

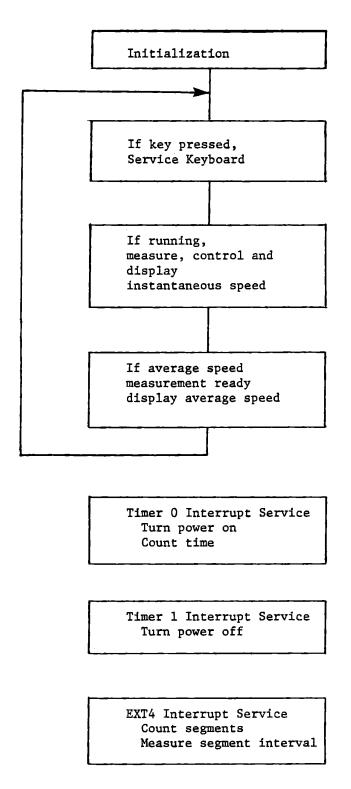
The motor is a three pole commutated dc motor. Its speed with constant load is approximately proportional to the voltage across the motor, as indicated in Figure 7-7. The anomaly in Figure 7-7 in the vicinity of 0.6 to 0.7 volts is related to effects of synchronism between the driving pulses and the motor commutation. The data for Figure 7-7 was taken with the closed loop control program that is developed in this section. Other control schemes would show a linear relationship.

The average voltage measured at the motor is not linear with pulse width (or duty cycle) as the capacitor voltage was in Chapter 6. The motor itself generates a voltage, which depends on its speed. This is called "Back EMF". (EMF stands for ElectroMotive Force, which means voltage). This voltage is present whenever the motor is running, even though the power transistor is turned off part of the time. The voltage observed at the motor is shown at the top of Figure 7-7.



Motor Speed vs. Duty Cycle Open Loop
Figure 7-8

In Figure 7-8 the motor speed is shown as a function of pulse duty cycle. This was measured in an open loop control system. A linear relationship exists from 30% to 50% only. Below 30% duty cycle the motor will run only sporadically, while above 50% the slope decreases. No external load was placed on the motor for these tests, but the motor bearing friction represents a substantial load at the higher speeds. With a load on the motor the speed versus duty cycle would be linear over a larger range. Because this toy motor has plastic bearings different motors will behave very differently, and a single motor will change its behavior from time to time. Therefore you cannot expect to duplicate these results with any precision. The relationship shown here, with a small change in pulse width giving a large change in motor speed, will be typical.



Motor Control Program Structure
Figure 7-9

7.2 CONTROL SYSTEM DEVELOPMENT

We will develop a closed loop control system following the general design structure and development approach that were used in the preceding chapter for voltage control. In this program keyboard commands permit setting a pulse duty cycle for open loop control or setting a desired speed for closed loop control. There is provision for setting gains in the proportional plus integral control equation. The motor can be started by RUN and stopped by STEP.

As in the voltage control system the program comprises initialization, interrupt service, and a main loop which calls various subroutines as needed (see next page). A keyboard service module is called when the main loop detects a key being pressed. If the motor is running a speed control subroutine is called once during each pass through the main loop, and the instantaneous speed is displayed. Each time a new average speed measurement is completed the main loop calls DWORD to display the average speed.

Pulse width modulation is used for power control. Timer 0 runs continuously in mode 2 to define the pulse frequency, or total period. Power to the motor is turned on by Timer 0 interrupt. Timer 1 operates in mode 5 and is triggered by the output of Timer 0. The interval loaded to Timer 1 sets the pulse width (on-time) and its interrupt turns power off. Both instantaneous and average speed are measured and displayed. The control loop acts on the instantaneous speed measurement. The only new programming problem is the calculation of instantaneous speed.

MOTOR CONTROL

Program Memory Assignments

8200 - 27	Initialization
8228 - 3F	Interrupt Manager
8240 - 5F	Timer Interrupt Service
8260 - 7F	EXT4 Interrupt Service
8280 - AF	Main Loop
82BO - BF	Not used
82C0 - DF	Subroutine SPEED
82EO - FF	Subroutine WIDTH
8100 - IF	KYTIM - Entry and Dispatch
8120 - 3F	Key Processing Modules
8160 - 7F	Subroutine LDT1
8180 - 5F	Subroutine DECBI
8190 - 9F	Subroutines SMULT, SCUML
81AO - BF	Subroutine DIVID

Data Memory Assignments

83A0	Binary Time Count
83A1	Motor Control Byte
83A2	Binary Segment Count
83A3,A4	Decimal Segment Count
83A5,A6	Average Speed
83A7,A8	Timer 2 Data
83A9	Desired Speed
83AA	Integral Gain
83AB	Proportional Gain
83AC.AD	Error Integral

MOTOR CONTROL

7.2.1 Speed Measurement

Speed is measured by observing the EXT4 interrupts generated by the optical disc. Each time a clear segment of the disc appears between the LED and the phototransistor of the slot sensor, the phototransistor is turned on, its output signal goes low, and an EXT4 interrupt occurs. In Chapter 5 we measured pulse interval time (Section 5.1) and frequency (Section 5.2). Both techniques are used in this program.

Average speed is measured as a frequency, by counting interrupts over a fixed period of time. Since the optical disc has 16 clear segments we will receive 16 EXT4 interrupts per revolution. If we were to count EXT4 interrupts during 1/16 of a second the count would represent revolutions per second. For average speed we would like better resolution, so we will count for 10/16 second, obtaining the average speed in tenths of a revolution per second. Thus a count of 0506 would represent 50.6 rps. For convenience the counting and display are in decimal.

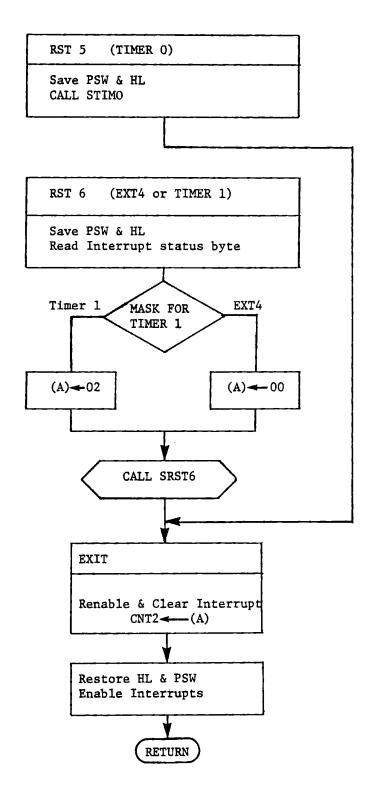
To control the speed well under variable load conditions we need more frequent measurements. Even the 1/16 second interval would be too infrequent for good speed control. Therefore the closed loop control is based on pulse interval measurement, giving "instantaneous speed" by division.

Speed = 16/(time per segment)

Time per segment = (clocks per segment)/2048000

Speed = $16 \times 2048000/(clocks per segment)$

The subject of binary division has not been treated previously in this course nor in Course 525. A subroutine, DIVID, is given in the program solution.



Motor Control Interrupt Manager
Figure 7-10

7.2.2 Interrupt Service

Three interrupts are used in the motor control system: Timer 0, Timer 1, and EXT4. Timer 0 is distinguished by its vector, RST5. Timer 1 and EXT4 both generate RST6 and must be distinguished by reading and masking the interrupt status byte.

7.2.2.1 Interrupt Manager

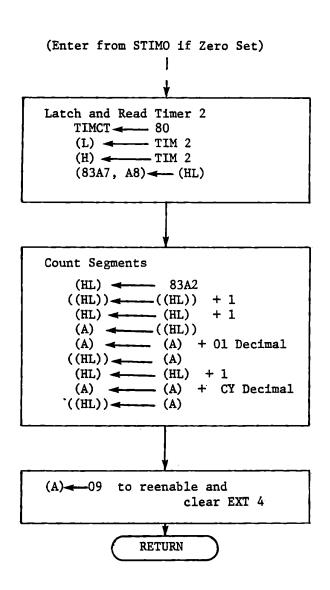
Figure 7-10 shows the interrupt manager. Each entry saves appropriate registers (PSW and HL only). A service subroutine STIMO is called at RST5, and then a jump is made to the exit module. At RST6 the same registers are saved, the interrupt status byte is read and masked by:

IN PORT 2B

ANI 02

Then a service subroutine is called. At entry to this subroutine register A contains 02 if Timer 1 generated the interrupt. Otherwise (A) = 00 and the zero flag is set. The subroutine executes a jump if zero to service EXT4, or proceeds directly to service Timer 1.

Each service subroutine must return in register A the necessary control byte to clear and reenable its interrupt flip flop. The exit module writes this byte to CNT2, restores the environment, and returns to the interrupted instruction. Note that the service subroutines are restricted to using registers H, L, A, and the flags, or must preserve other registers.



EXT 4 Interrupt Service
Figure 7-11

7.2.2.2 EXT4 Service

EXT4 interrupt occurs each time a clear segment of the optical disc appears between the LED and phototransistor of the slot sensor. EXT4 service performs two functions in response. It latches and reads Timer 2, which is running continuously, and stores the data for use by the speed measurement subroutine. In that subroutine (called by the main program loop) we subtract the latest measurement from the preceding measurement to obtain the time difference. We could clear and restart Timer 2 at each EXT4 interrupt, thereby making each measurement complete in itself. We will see later that this would require more rather than fewer instructions in the speed measurement subroutine, as well as requiring additional instructions here.

Note an interesting point with regard to the mode selected for Timer 2. We usually think of mode 2 for a timer which is to run continuously. When no output signal or interrupt is needed we can use mode 0 instead, because the timer continues to count down after it reaches zero, although its output will remain high after the first time it reaches zero. Mode 0 is used in this program to demonstrate the point.

MOTOR CONTROL

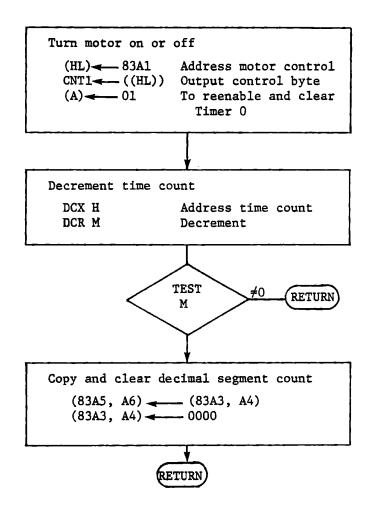
This page intentionally left blank.

7.2.2.3 Timer Service Subroutines

Pulse width modulation is controlled by Timer 0 and Timer 1 interrupts. Their service routines turn the power transistor on or off by writing a byte to Port 1C. If the byte is 00 it sets Port 1C1 low and turns the transistor on, while 02 sets Port 1C1 high and turns the transistor off. As in the previous program the Timer 0 interval is constant and defines the total period. If the motor is running, Timer 0 turns the power transistor on. The Timer 0 output also triggers Timer 1, whose interval sets the on-time. The Timer 1 interrupt turns the transistor off. Timer 1 has no other function.

Timer 1 Service

JZ	SEXT4	If zero set service EXT4
OUT	PORT 1 C	Output 02 to turn motor off
MVI	A,03	To reenable Timer 1
RET		Exit



Timer 0 Service
Figure 7-12

Timer 0 has several functions. Since we often will want the motor to be stopped, the control byte to be written to Port 1C is stored in memory (at 83A1) by keyboard command. RUN stores 00 and STEP stores 02. This byte is loaded from memory and output at each Timer 0 interrupt. In addition Timer 0 decrements a time counter (at 83A0) and at zero it copies and clears the decimal segment count accumulated by EXT4 interrupts. The timing is arranged so that this count directly represents average speed. The completed count is stored at 83A5.A6 for display by the main loop.

There is a fortuitous time interval that is suitable for the PWM total period and also for measuring the 10/16 second interval for average speed. 256 counts in the binary timer counter equals 10/16 second if the PWM total period is set at:

$$10/(16 \times 256) = .00244140625$$
 second

Although this may appear to be an awkward time interval to obtain, in fact the system clock generates it very easily:

$$5000/2048000 = .00244140626$$
 second

Timer 0 is programmed in mode 2, decimal, high byte only, and loaded with 50. The interval, slightly less than 2.5 milliseconds, is quite suitable for pulse width modulation.

7.2.3 Initialization

Ports and Timers are to be programmed as follows:

8	3255	#1	A out B out C out
8	3255	#2	Ain Bin Cout
]	Cimer	0	High byte, mode 2, decimal
7	limer	1	Both bytes, mode 5, binary
7	limer	2	Both bytes, mode 0, binary

Timer 0 is to be loaded with 50 (high byte) to generate the 2.44 millisecond total period. Timer 2 must be loaded (in both bytes) to start it; the value does not matter since it will always count down from zero after it first reaches zero.

Recall that the motor is off when Port 1C1 is high, and running when that output is low. After system reset all ports are programmed for input, which gives an apparent high output and turns the motor off. As soon as Port 1C is programmed for output its output signals go low, the power transistor is turned on and power is applied to the motor. (You can demonstrate this by stepping through your initialization procedure). Since none of the control data have been entered we want to stop the motor during initialization; this can be done by a call to the CLR key processing module in KYTIM. The command keys are defined in Section 7.2.5.

We have used RST5 and RST6 commands in the past to enable interrupts. This would work for RST5 in this program, but two calls to RST6 service would be required to enable both EXT4 and Timer 1. Even with two calls there is no certainty that Timer 1 would be enabled because

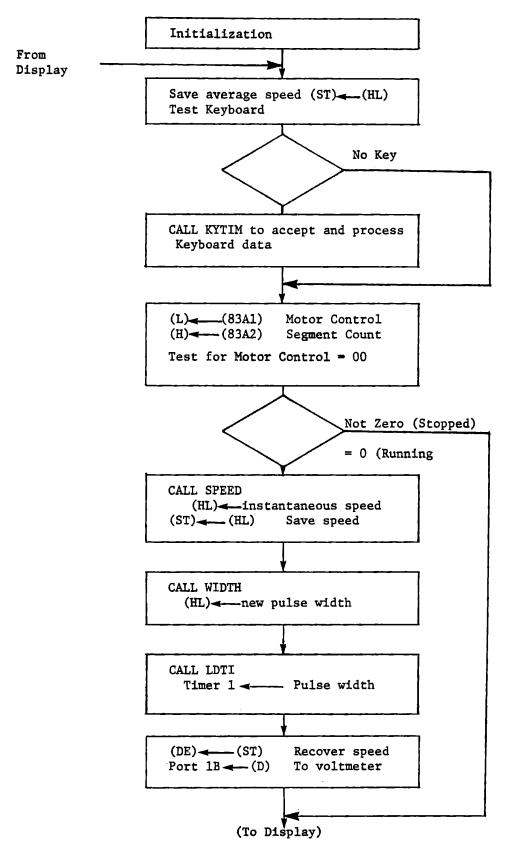
it is only serviced if its interrupt is present. Therefore we enable the interrupts by writing 13 to Port 2C. This usually results in all three interrupts being serviced immediately, since writing to Port 2C does not clear the interrupt flip flops. During debugging it is desirable to replace this process with the RST instructions, which will permit stepping though the interrupt service routines.

	Fi na l		Debug	
3E	MVI	A,13	E F	RST5
13			F7	RST6
D3	OUT	PORT 2C	F7	RST6
OE			00	NOP

While you step through the initialization the motor will run at full speed until the CLR function stops it. It is advisable to unplug the motor during this task.

You may have to force the service of Timer 1 by replacing the interrupt status byte after reading it. (Use REG, A, O2 after the status byte has been read by the IN PORT2B instruction).

When debugging of the initialization and interrupt service routines has been finished, replace the RST instructions with the load and output instructions. These are followed by a jump to the main loop.



Motor Control - Main Loop Figure 7-13a

7.2.4 Main Program Loop

Figure 7-13 shows the main program loop. This is a more detailed version of Figure 7-9.

At the start of the loop register pair HL normally contains the average speed. (After initialization this value is meaningless). This value is saved, and the keyboard is tested by:

IN PORTOA

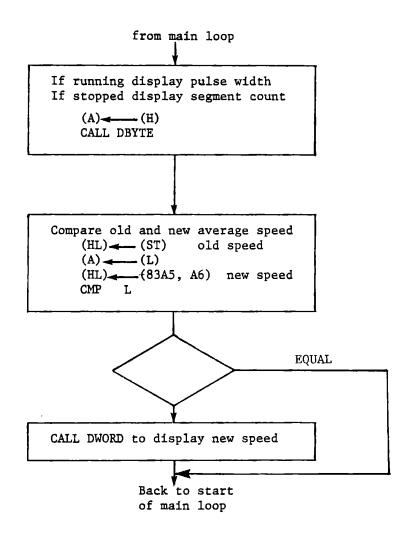
INR A

CNZ KYTIM

Note that while Figure 7-13a indicates a conditional jump, a conditional call is actually used here.

As in the voltage control program the process operates in open loop mode while KYTIM waits for keyboard entry and processes the data entered.

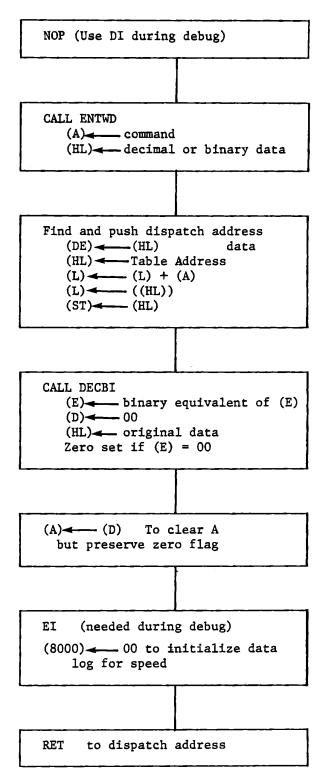
The motor control byte stored for Timer 0 interrupt service is also used here to determine whether the motor is running. If it is stopped (control byte = 02) we skip all of the control functions and go to display the binary segment count. If the motor is running three subroutines are called. SPEED finds the instantaneous speed from a segment interval; WIDTH calculates a new pulse width; LDT1 loads Timer 1 with the new width.



Motor Control - Display
Figure 7-13b

The high byte of the speed is output to the D/A converter. This allows observation of the speed with a voltmeter, which is interesting to watch when a load is placed on the motor. Either the high byte of the speed, or optionally the high byte of the pulse width, is displayed at the right, as shown in Figure 7-13b.

After displaying the instantaneous speed or the segment count the main loop may display the average speed. The average speed is stored by Timer 0 interrupt once every 5/8 second. Since the optical disc has 16 light segments, the decimal segment count during the 5/8 second directly represents speed in tenths of a revolution per second. Thus a display of 0506 means 50.6 rps. To avoid wasting time in the control loop the program tests for a change in the average speed before displaying it. To permit this test the old value is stored at the beginning of the main loop and recovered before the (possibly) new value is loaded. The result is that DWORD is called only once in about 250 passes through the main loop. (It is sufficient to compare the low bytes of the old and new speeds, since it is unlikely that successive measurements will be alike in the low byte).



KYTIM - Input and Dispatch
Figure 7-14

7.2.5 Keyboard Input Subroutine KYTIM

This version of KYTIM uses the same keyboard entry and dispatch techniques used in previous programs. A decimal to binary conversion is required, since some data are entered as decimal values but are needed as binary values. This is done by subroutine DECBI (Section 7.2.6) which is entered with keyboard data in (HL), and returns with the input data preserved and the binary equivalent of the low byte in (E) and also in (A). Register D is cleared. The zero flag is set if (E) = 00. After return register A is cleared by MOV A,D, so the zero flag is preserved and passed to the command processing module.

We will provide for logging the motor speed for subsequent review, as we logged the voltage in Chapter 6. Clear the content of memory location 8000 to initiate a new log at each keyboard entry.

This page intentionally left blank.

The following command key processing modules are needed:

CLR Clear the binary segment count and stop the motor.

This permits measuring the stopping distance.

STEP Stop the motor (turn off the transistor and store 02 at 83A1)

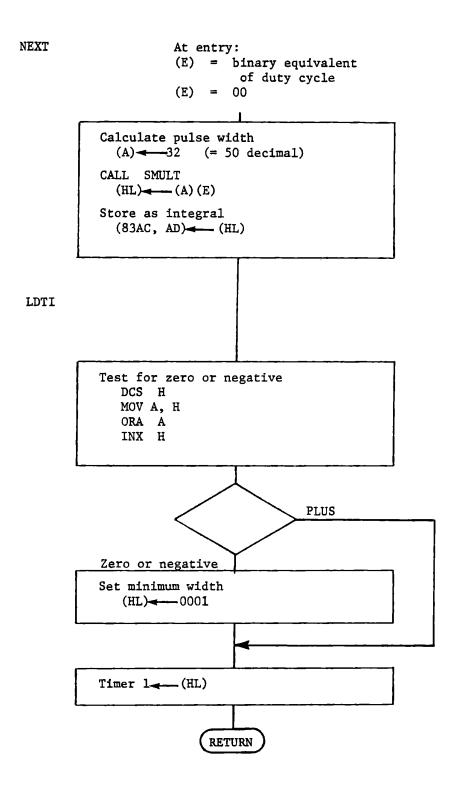
RUN Start the motor (store 00 at 83A1).

If a desired speed is entered, store its binary equivalent at 83A9.

MEM Store integral and proportional gain. These are entered in binary, and are to be stored at 83AA and 83AB. (83AA,AB) <--- (HL)

NEXT Given a decimal duty cycle (converted to binary by DECBI) calculate pulse width. Store the results as

The total period for PWM has been set to 5000 (decimal) clocks. We can calculate a pulse width by multiplying the duty cycle (treated as an integer) by 50 (decimal). For instance, if a duty cycle of 40% is requested, $40 \times 50 = 2000$, which is 40% of 5000.



Load Timer 1 Modules
Figure 7-15

In fact the calculation will be done in binary, and the binary result is used. A multiplication subroutine SMULT (Section 7.2.7) returns (HL) = (A)*(E). Since DECBI has given (E) = binary equivalent of the input data, the processing for NEXT is:

MVI	A,32	Binary equivalent of 50
CALL	SMULT	(HL) < pulse width
SHLD	83AC	Store as integral
CALL	LDT1	Load Timer 1

As in the voltage control program the Timer 1 loading subroutine is also used in closed loop control. It tests for a zero or negative value in (HL) and replaces such a value with 0001 for minimum pulse width. Timer 1 is loaded with the contents of L and H.

7.2.6 Subroutine DECBI

A two digit decimal value can be converted to an equivalent binary value by:

Binary value = Low digit + 5/8 (High digit)

A convenient algorithm for this calculation is expressed as:

Binary Value = Decimal Value
+ 1/8 High digit
- 1/2 High digit

For convenience in the motor control program subroutine DECBI accepts and returns data as follows:

ENTER

(HL) = Keyboard data

RETURN

(HL) = Keyboard data preserved

(A) = (E) = Binary equivalent of low byte

(D) = 00

Zero set if low byte is zero

CY clear

(BC) is preserved

A program solution is given in Figure 7-22d.

7.2.7 Subroutines SMULT, SCUML

SMULT multiplies two single byte values and returns the two byte product. SMULT clears the product at entry. SCUML is an alternate entry at which the product is not cleared, allowing cumulative multiplication.

ENTER

- (A) = Multiplier
- (E) = Multiplicand
- (D) = 00 if multiplicand is positive
- (D) = FF if multiplicand is negative
- (HL) = Previous product (SCUML only)

RETURN

- (HL) = Product (A)*(E) for SMULT
- (HL) = (HL) + (A)*(E) for SCUML
- (DE) destroyed
- (BC) preserved
- (A) = 00

Zero set, Carry clear

The multiplication is carried out by repetitively shifting (A) right, and adding the multiplicand to the product if the bit shifted out is a one. After each shift of (A), and after the addition if it is performed, the multiplicand is shifted left. The subroutine returns when the content of A is zero. A program solution is given in Figure 7-22e.

7.2.8 Open Loop Operation

With the functions described so far you can operate the system in open loop mode. (Use an unconditional JMP around the control functions instead of JNZ. This permits debugging of the main loop, KYTIM, and interrupt service.) Write all of these program modules. Check the program flow through interrupt service, using RST5 and RST6 instructions. Check the program flow through KYTIM and see that the calculations and data storage are correct.

To run the motor enter a high duty cycle (60% or more) with NEXT and then press RUN. See that the motor can be speeded by higher duty cycles and slowed by lower duty cycles. Find the lowest duty cycle that will keep the motor running once it is started, and the lowest that will cause it to start. Record and plot speed versus duty cycle, and compare your result with Figure 7-8.

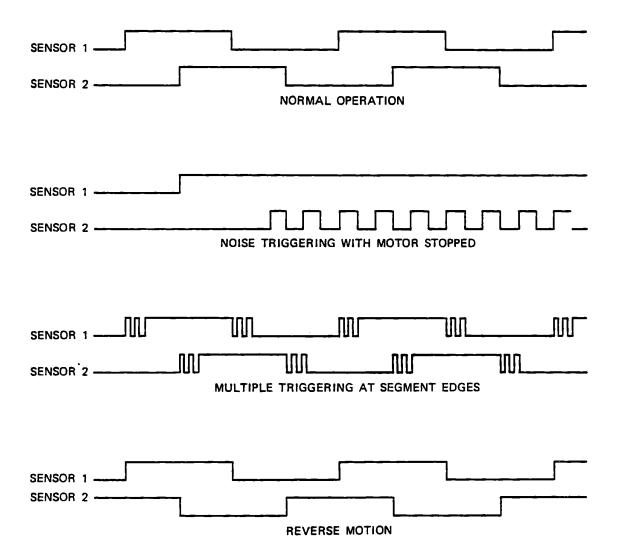
7.2.9 False Speed Indications

When the motor is stopped with an edge of a segment in the optical sensor light path, the phototransistor will be in an active state, neither fully on nor fully off. In this state the circuit is very susceptible to small signals, and noise pickup is likely to cause the EXT4 input to switch. The Timer 0 output, which is connected to Timer 1 gate in this experiment, is a source of such noise. If this source does cause switching, every Timer 0 cycle will result in an EXT4 interrupt. Then the binary and decimal segment counts will constantly be incremented. After 256 counts Timer 0 service will copy and clear the decimal count, so an average speed of 2.56

revolutions per second will be displayed. It is fairly difficult to find the exact sensor position necessary to obtain the continuous counting. If you turn the shaft slowly by hand it is easy to observe multiple counts for each segment, demonstrating that the false switching occurs.

Another false speed indication can occur when power is applied to the motor but the pulse width is insufficient to start it. At each pulse the motor shaft will turn slightly, but will be pulled back by the motor magnets when the pulse ends. If this occurs with a segment edge in the optical sensor light path the phototransistor will switch in time with the motor pulses.

There is пo good solution to these problems with a single optical sensor. In any application where such false indications are intolerable it is necessary to use two sensors to observe the disc. They must be so located that the two sensors cannot both see segment edges at the same time. Now the program can test for a sequence of switching in the two sensors to ensure that only valid segment Figure 7-16 shows the sequences under transitions are observed. various conditions. A program could be designed so that an interrupt from sensor 1 disables itself and enables the sensor 2 interrupt, and sensor 2 interrupt disables itself and enables sensor 1. Now interrupts can occur only when both sensors switch in sequence. Noise and multiple triggering are thereby excluded.



Motion Detection with Dual Sensors
Figure 7-16

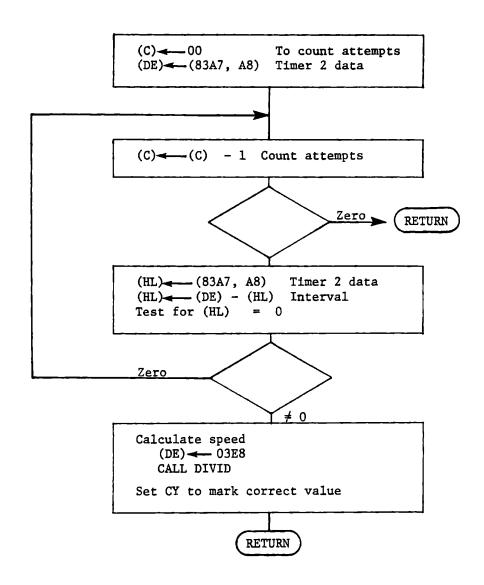
Dual sensors can also be used to detect direction of motion. The difference between the top and bottom diagrams in Figure 7-16 is readily analyzed to determine which way a shaft is turning. Since we have only a single sensor available, we cannot determine direction nor eliminate the false speed indication. This is a problem only when the motor is stopped or running very slowly. At speeds above 5 to 10 revolutions per second the single sensor is accurate. Although some schemes are available to reduce the probability of false speed indications with a single sensor, we will not attempt their implementation.

7.3 CLOSED LOOP MOTOR CONTROL

To close the loop we will calculate the instantaneous speed from the measurement of segment time, and apply the proportional plus integral control equation:

$$F = G_{p}E + \int G_{1}E$$

Five subroutines are used. SPEED obtains the interval time and calls DIVID to calculate the instantaneous speed. WIDTH subtracts the instantaneous speed from the desired speed to give the error signal, calls a cumulative multiplication subroutine SCUML to calculate a new error integral, and calls SCUML again to calculate a new pulse width. Finally LDT1 loads Timer 1 with the pulse width.



Subroutine SPEED

Figure 7-17

7.3.1 Subroutine SPEED

EXT4 service records the content of Timer 2 when the optical disc generates an interrupt. SPEED (shown in Figure 7-17) repeatedly loads this value (from 83A7,A8) and subtracts it from the previous value. If the result is zero no EXT4 interrupt has occurred, so the reading and subtraction are repeated. After 256 attempts it is assumed that the motor is not moving and SPEED returns with (HL) = 0000.

If successive values of Timer 2 data are different the subtraction of the later value from the earlier gives the time interval, since the timer counts down.

We will obtain the instantaneous speed by division. The speed in revolutions per second is given by:

(16 segments/rev.X2048000 clocks/sec.)/clocks per segment

The division subroutine DIVID (Section 7.3.3) is designed for use with left justified floating point numbers, although it does not handle the exponents. It returns a 16 bit result in (HL) with the most significant bit representing the integer part and 15 bits representing a fraction. It can handle numbers that are not left justified provided that the dividend is not greater than twice the divisor.

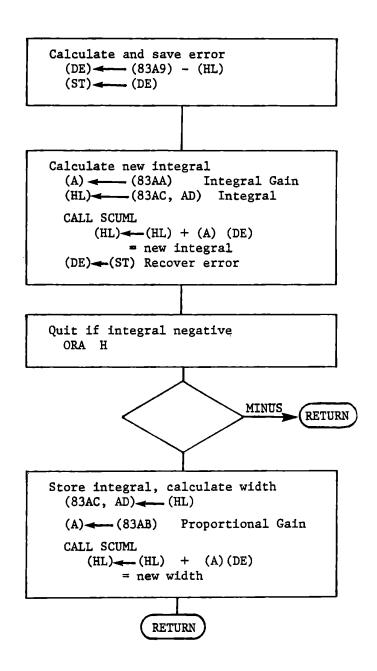
This page intentionally left blank.

It turns out that if we enter DIVID with a dividend of 03E8 (in DE) and the segment interval (in HL) as divisor, it returns a quotient (in HL) representing speed as XX.XX. That is, H contains the integer part and L contains the fractional part. The table below shows the relationship between speed and clocks per second. For all speeds that this motor can achieve the number of clocks per segment (the divisor) is more than half of 03E8, so DIVID will return a correct result.

Carry is set before return from SPEED to indicate that a valid measurement has been made. This is in fact ignored in the present main program, but could be used to invoke full scale control.

Speed and Clocks per Segment

Motor Speed rps	Segments per Second	Seconds per Segment	•	n Clocks Segment Hex		
2	32	.031250	64000	FAOO		
5	80	.012500	25600	6400		
10	160	.006250	12800	3200		
20	320	.003125	64000	1900		
50	800	.001250	2560	0A00		
100	1600	.000625	1280	0500		
150	2400	.000417	853	0355		



Subroutine WIDTH
Figure 7-18

7.3.2 Subroutine WIDTH

At entry to WIDTH the instantaneous speed is in (HL) as XX.XX. The calculations are limited to single byte values with sign, so we calculate error from the high byte, rounding from the low byte of speed.

MOV	A,L	Set CY if low byte
RAL		Greater than 1/2
LDA	83A9	Desired speed
SBB	H	Subtract speed
MOV	E , A	(E) < error
SBB	A	(D) < FF if negative
MOV	D,A	(D) < 00 if positive

The multiplication subroutine SCUML demands that register D contains 00 if (E) is positive, FF if negative. The error is saved in the stack because it is needed for both the integral and proportional calculations, and SCUML destroys the contents of DE. SCUML returns (HL) = (HL) + (A)*(E). To calculate a new integral we load the integral gain to A and the old integral to HL and call SCUML. At return HL contains the new integral and A contains zero. The error is recovered by POP D, and the integral is tested by ORA H, which sets the minus flag if the integral is negative. RM (Return if Minus) after the test avoids storing a negative integral.

This page intentionally left blank.

Provided that the integral is positive, a new pulse width is calculated by loading A with the proportional gain and calling SCUML again.

At this call we have:

(A) = Proportional gain

(E) = Error

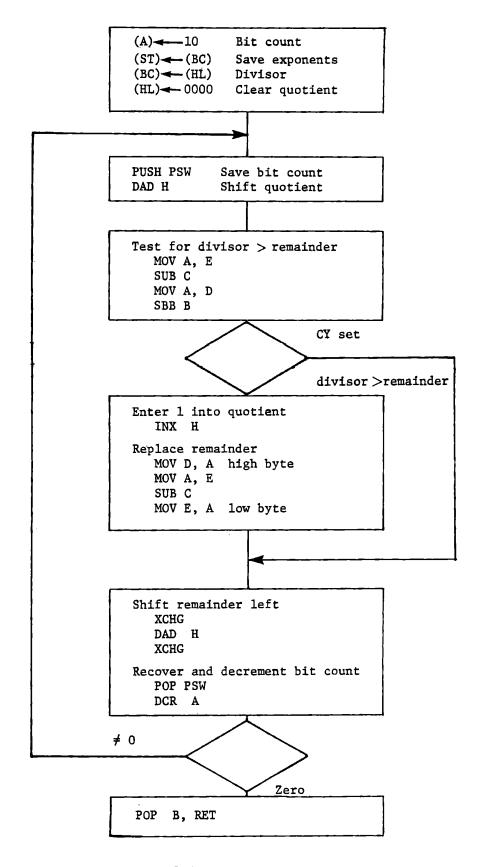
(IIL) = Integral

SCUML returns (HL) = (HL) + (A)*(E) which is the new pulse width:

$$F = G_{p}E + \mathbf{f}G_{i}E$$

The main loop will call LDT1 to load Timer 1 with the new pulse width.

You may want to elaborate this program to permit a negative integral and test for arithmetic overflow. If a positive error changes the integral from positive to negative, the motor is running too slowly (or is stalled), so return a large pulse width, such as 1200H (92% duty cycle). If a negative error changes the integral from negative to positive, return a minimum pulse width (0001) to slow the motor. The erroneous integral should not be stored. The same test and correction should be made after adding in the proportional term.



Subroutine DIVID

7.3.3 Subroutine DIVID

Binary division is performed by repeatedly comparing the divisor to the dividend or remainder. If the divisor is less than the dividend or remainder, it is subtracted to form a new remainder and a one is entered into the quotient. Otherwise the old remainder is retained and a zero is entered into the quotient. Now both remainder and quotient are shifted left if more bits are to be processed, and the process is repeated.

Figure 7-19 shows subroutine DIVID. Registers B and C are saved in the stack; in a floating point division program which uses DIVID these registers hold the exponents. Here we do not really need to save them.

Register A is loaded with a bit count; the divisor is copied to (BC) and the quotient (HL) is cleared. The subroutine could be shortened here, because 16 left shifts will clear the old data from (HL). In some fixed point applications, however, a different bit count might be used.

Since all registers are used the bit count is saved in the stack during each loop. The quotient is shifted left at the beginning of the loop rather than at the end because it should not be shifted after the final bit has been processed. The left shift enters a zero into the quotient.

This page intentionally left blank.

A two byte subtraction sets carry if the divisor is greater than the remainder, in which case the old remainder and the zero bit shifted into the quotient are retained. If the subtraction does not generate carry the low bit of the quotient is made one by INX H, and the remainder is replaced. The high byte of the new remainder is available in A, but the low byte must be generated.

Now the remainder is shifted left and the bit count is recovered and decremented. When the bit count reaches zero the quotient is compete in (HL).

Because this subroutine was developed for floating point arithmetic it expects left justified values—the highest bit of the dividend and of the divisor should be one. This implies that the dividend is less than twice the divisor. The 16 bit quotient represents a single bit to the left of the binary point and 15 bits to the right. The algorithm is valid for numbers which are not left justified only if the dividend is less than twice the divisor. As we have seen, this relationship does hold in the motor control program.

	JIOK (COIN	IKO																
н	×	Keyboard Data except if NEXT	×	Instantaneous speed	Instant speed	ruse widtn	Pulse Width	See Notes	Keyboard Data	Keyboard Data Preserved	×	(A) *(E)		Previous Value	(HL) + (A) * (E)		Divisor	Quotient	
D B	×	×	×	×	×	×	*	Saved	×	00 Binary of low byte	00 or FF M'cand	×	(See Notes)	00 or FE M'cand	×	(See Notes)	Dividend	×	(See Notes)
ВС	×	00 Command	×	Saved 00	×	Saved	×	Saved	×	Saved	×	Saved		×	Saved		×	Saved	
A	×	×	×	00	×	00	×	= (H)	×	Binary of low byte	M'plier	00		M'plier	00		×	00	
ZERO	×	×	×	Set	×	Set	×	×	×	See Notes	×	Set		×	Set		×	Set	
CZ.	×	×	×	See Notes	×	Clear	×	Clear	×	Clear	×	Clear		×	Clear		×	Saved	
	KYTIM - Entry	Return	SPEED - Entry	Return	WIDTH - Entry	Return	LDTI - Entry	Return	DECBI - Entry	Return	SMULT - Entry	Return		SCUML - Entry	Return		DIVID - Entry	Return	

Subroutine Register Usage

Figure 7-20

7.3.4 Summary of Subroutines

The subroutines used in the motor control program are briefly summarized below. Figure 7-20 shows the register usage for each subroutine. In the figure the data required at entry and returned at exit are listed. An X for entry indicates that the register content does not matter; an X for return indicates that the register content is destroyed.

KYTIM accepts (via ENTWD) optional decimal or hexadecimal data and a command. Data entered are stored in assigned memory locations as described in Section 7.2.5. Valid commands are:

NEXT - Set duty cycle (enter in decimal)

STEP - Stop motor (optional - set speed)

RUN - Start motor (optional - set speed)

CLR - Stop motor, clear segment count

MEM - Store proportional and integral gains

SPEED calculates instantaneous speed. Obtains interval time from Timer 2 data stored in memory by EXT4 interrupt. Returns carry set except if no interval time is observed after 256 attempts. In that case speed is reported as 0000.

WIDTH calculates and stores integral of error, and returns new pulse width.

LDT1 loads pulse width to Timer 1. If entry value is zero or negative LDT1 replaces entry value with 0001 before loading timer, and returns MINUS flag set.

<u>DECBI</u> converts a packed decimal input byte in (L) and returns the binary equivalent of the low byte in (DE). The zero flag is set if the low byte was zero.

SMULT calculates (A) times (E) and returns the two byte product in HL. At entry register D must contain 00 if (E) is positive or FF if (E) is negative. Thus (DE) is a two byte twos complement value.

SCUML calculates (HL) + (A) (E). It is an alternate entry to SMULT, and omits clearing the product before multiplying.

 $\overline{ ext{DIVID}}$ calculates (DE)/(HL). The dividend must be no greater than twice the divisor. The quotient is represented as one bit left of the binary point and a 15 bit fraction.

7.3.5 Speed Logging

The solution for motor speed control in Figures 7-21 and 7-22 does not include speed logging. You can add this feature, using a technique similar to that in the logging voltmeter. To do this you will want a subroutine that records only once in ten or sixteen cycles through the main loop, because of the fairly long reaction times of the motor. Section 7.3.6, following the program solution, discusses some of the results you will see with the program given.

	Δ	ם ם	R	со	DF	MOTO	OR	CO	NTR	OL	-	INI	ΙΤΙ	AL.	IZE	;	
		20	0	3	E		M	V	I		A		8	0			Program Ports
	l		1 _	8	0					,							Port 1B Out
			2	(બ		0	U	\mathcal{T}		C	N	\mathcal{I}	1			
			3	0	7		_	_									
Ë			4	3	E		Μ	ν	T		A		9	2			
SHEET	-		5		<u></u>		741	V	L		//	/	7	Χ.			
	<u> </u>			14	3 2		_			\vdash			_	_			
CODING	ļ		6	$\underline{\mathscr{D}}$	3		0	U	7		C	N	\mathcal{T}	2			
္ပ			7	0	F												
			8	3	E		M	V	\mathcal{I}		A		2	5			Frogram Times O
			9	2	5												Ligh leyte
			Α	D	3		0	U	T		T	I	M	C	$ \mathcal{T} $		High lute Mode 2
		_	В		7												decimal
			С	3	E		M	V	T		A		5	0		,	Load Times O
Σ			D	5	0		4	<u> </u>	-4-			/	- _	<u> </u>		 	Son 2. 4414 ms
SYSTEM	 		E	3	3		0	.)	7	-	7	エ	М	0	-	-	
			F	1	4				/	<u> </u>	-	_	7-1			-	= 10/16 pecond
TRAINING	-	4 /	0	1/2	_		M	レ	7	-			7	1			= 1/16 second)
Ž	8 _	2/		3	E		141	V	ュ	├	A	1		A	\vdash	-	Program Timer /
RA			1	7	A	<u> </u>	┝	- -			-	_			-	_	Both leytes
			2	2	3	 	0	U	1	<u> </u>	7	I	M	C	T		Mode 5
JE			3	1	7		ļ.,	ļ.,		ļ			_				Bixary
MP	<u> </u>		4	3	E	ļ	M	V	I	<u> </u>	A	,	B	0			Frogram Times 2
MICROCOMPUTER			5	B	0		<u> </u>										Both leytes
,RO			6	D	3		0	υ	7	Ĺ	T	I	M	C	7		Mode O
Σ			7	/	7												Bixary Start Times 2
			8	A	F		X	R	A		A						Start Times 2
			9	7	3		0	U	7		\mathcal{T}	I	М	2			
			A	7	6		_	<u> </u>									
MS	1		В	D	3	<u> </u>	0	U	T		7	I	M	2			
STE	-		С	1	,		-		 	-	 	-	/ //	~	-		
INTEGRATED COMPUTER SYSTEMS	-		D	c	Ø D	<u> </u>	0	A	7	7	 	\mathcal{C}	7	R	ļ	<u> </u>	(1)
EB	<u> </u>			 -	· .	<u> </u>	۲	17	-	_		ب	ے	_	_	_	Clear segment
5	<u> </u>		E	2	0		├	-		_	-			-	<u> </u>	<u> </u>	Clear segment count and stop motor Enable interrupts Jimes O, Times 1,
ΜĀ	<u> </u>		F	8	/_		ļ.,	_	<u> </u>		-			-	<u> </u>		stop molor
8	8	22		3	E		M	V	I	<u> </u>	A	,		3	_		Enable interrupts
Œ.	<u> </u>		1	1	3		<u> </u>	<u> </u>	<u> </u>					ļ.,	<u> </u>		Jimes O, Times 1,
1AT			2	1	3		0	U	T	<u> </u>	P	0	R	$ \mathcal{T} $	2	C	EXT 4
EGE			3	0	E		L										
Z			4	C	3		5	M	P		8	2	8	0	L		
_			5	8	0												
			6	8	2												
			7	İ													
		<u> </u>	8			<u> </u>											Figure 7-21a
				—		-	L	Ц.	Щ.								

8 0 1 1 2 2 3 3 4 4 4 5 5 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	
2 3 4 4 5 5 6 7 8228 F5 PUSH PSW RST5—	
3 4 5 5 6 7 8228 F5 PUSH PSW RST5—	
4	
8228 F5 PUSH PSW RST5—	
8228 F5 PUSH PSW RST5—	
8228 F5 PUSH PSW RST5—	
8228 F5 PUSH PSW R5T5-	
8228 F5 PUSH PSW R5T5-	
9 E5 PUSH H Jiner O inte	
A CD CALL STIMO	·
	rrupe_
	· -
<u> </u>	
	·
8 23 0 F 5 P U S H P S W R5T 6 T E 5 P U S H H Jinew / ov E	
1 E5 PUSH H June 1 ov E 2 DB IN PORT 2 B Read interse 3 OD status byte 4 E6 ANI O2 Mask for 5 O2 June 1	=XT4
	ept
3 OD status byte 4 E 6 A N I O2 Mask for 5 O2 Jener / 6 CD CALL STIMI Call service A 7 58	ē
a E 6 A N I O2 Mark for	
8 5 02 Jinev/	
6 CD CHLL STIM Call service	routine
8 82	
8 82 8239 D3 OUT CNT2 EXIT - Leen A OF and cleavin	abl
o A OF and cleavin	terrupt
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	
5 C F/ POP PSW	
S D F B E I E C 9 P E 7 F O O 8 0 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	
F O O	
S 8 0	
	·
<u>Б</u>	
4	
5	
6	
7	
8 Figure 7-21b	

- -

	A D D R	со	DE	MOTO	OR	COI	NTR	OL	-	SEF	RVI	CE	TI	ME	RS
Ī	8 240	2	1		L	X	I		\mathcal{H}	,	8	3	A	7	STIMO
	1	A	/												address, load
	2	8	3												and output
⊢	3	7	E		M	0	V		A	,	M				motor control byte
SHEET	4	<u></u>	3		0	U	\mathcal{T}		P	0	R	7	/	0	to set PICI low
	5	0	6			<u>.</u>									if RUN, high if STOP
CODING	6	3	E		M	V	I		A		0	/			To renable
힝	7	0	/			<u> </u>									Lines O at exit
	8	2	B		D	C	X		H						address and
	9	3	5		D	C	R		M						decrement
	Α	<u>C</u>	0		R	N	Z				_			<u> </u>	time courter
	В	2	A		4	#	L	\mathcal{D}		8	3	A	3		at 00 = 5/8 second
_	С	A	3		<u> </u>	<u> </u>								_	copy decimal
SYSTEM	D	8	3		_	_			L						signest count
3YS	E	2	2		S	14	4	D	_	8	3	A	5	<u>_</u>	to display area
	F	A	2	_	_	Ĺ									, ,
TRAINING	8 25 0	8	3		ļ.,	<u>_</u>	_						_		
ВA	1	2	/		4	X	I	<u> </u>	#	,	0	0	0	0	
ER T	2	0	0			ļ		<u> </u>						<u> </u>	segment counter
UTE	3	0	0			<u> </u>	_	<u> </u>	<u> </u>				_	<u> </u>	0.
MICROCOMPUT	4	2	2		S	1	۷	\mathcal{D}		8	3	A	3		
2	5	A	3	_	-			<u> </u>	<u> </u>					<u> </u>	
CR(6	8	+=-			_	_				_		-		
≅	7	C	9		R	E	7	_							V
	8258	C	_		J	군		_	S	E	X	\mathcal{T}	4		STIM/
	9	6	0		-	_		<u> </u>	-						Jump if EXT4 Else (A) = 02 Turn motor off
۱S	A	8			<u> </u>	_	<u> </u>					_	_		Else (A) = 02
SYSTEMS	В	\mathcal{D}	3		0	U	T		P	0	R	T	/	0	Turn motor off
SYS	С	0	6	ļ	100		-	ļ					_		To reenable Time O
	D	3	E	<u></u>	M	V	I		A	<u>ر</u>	0	3	 	<u> </u>	To reenable Jemes O
Ų	E F	-10	+	<u> </u>		_	7		-	<u></u>					
OMF	8 0	C	17	<u> </u>	R	E	T		<u> </u>		-				-
Ö	1	+	-	 	-			-			_	-	-		
TEI	2	+	-	 	 	\vdash	<u> </u>	\vdash	 		-		-	\vdash	
NTEGRATED COMPUTER	3:	+	-	 	-	-	-		-	-	-	_	-		<u> </u>
JTE(4	+	 	-	-	-	 	-	_		<u> </u>		-	-	
2		-+	 	 -	\vdash	-	\vdash	-				<u> </u>	-		
	6		1	-	\vdash	 	-	 		_	\vdash	-	-	-	
	7		+		\vdash	-	-					\vdash	-	 	
	8		T	 		-	-					<u> </u>		-	Figure 7-21c
	L					١		Щ.							

	A D D	R	со	DE	EXT	4	TIN.	LEK	KUI	?T	SEI	KAT	CE			
	8 26	0	જ	E		M	V	\mathcal{L}		A		8	0			Latch read,
		1	8	0												and store
		2	D	3		0	U	\mathcal{T}		\mathcal{T}	I	M	C	T		Liner 2 data
⊢		3	/	7												
SHEET		4		\mathcal{B}		エ	N			7	I	M	Q			
		5	/	6									٠			
CODING		6	0	F		M	0	V		7		Ā				
S		7	D	\mathcal{B}		\mathcal{I}	N			T	Ί	Μ	2			
		8	/	6								2		·		
		9	6	7		M	0	V		\mathcal{H}	,	A				
		A	2	2		S	H	2	D		8	3	A	7		
		В	A	7												
		С	8	3												Ų.
-EM		D	2	/		L	X	I		H	1	8	B	A	8	address and
SYSTEM		E	A	2							,					segment count
		F	8	3												segment count
Z	8 27	0	B	4		\mathcal{I}	N.	R		M						
TRAINING		1	V	3		工	\mathcal{N}	×		\mathcal{H}						address and
		2	7	E		Μ	0	V		A		Μ				increment decimal
TE		3	C	6		A	D	I		0	1					segment count
MICROCOMPUTER		4	0	/												1
CO	_	5	2	7		D	A	A								
RÕ		6	7	7		M	0	V		Z		A				
MIC		7	2	3		\mathcal{I}	N'	X		Н	7					
		8	7	E		Μ		V		A		M				
		9	C	E		A	0	I		0	6					
(0		Α	0	0												
EMS		В	7	7		D	A	A								
/ST	-	С	7	7		M	A O	V		M	,	A				↓.
1 S)		D	3	E		M	V	Z		A	,	0	9			To reenable and clear EXT & To exit
TEI		E	0	E 9							,					clear EXT &
ΛΡU		F	C	9		R	Ε	T								To exit
CO	8	0														
INTEGRATED COMPUTER SYSTEMS		1														
3AT		2														
EGF		3														
L		4														
_		5														
		6														
		7														
		8														Figure 7-21d

	4 D D	D	coi) F	1101	OK	CC)N I	KOI	, –	IVIE	7 T IN	Τι	JOP		
	8 28	0		5		P	u	5	\mathcal{H}		H					Save and speed
		1	E	\mathcal{B}		\mathcal{I}	N			P	0	R	7	0	A	Test keyboard
		2	0	0			•			,	_	•				(FF if no key)
_		3	3	C		I	N	R		A						
SHEET		4	C	4		C	N	R		K	У	7	\mathcal{I}	7		If key pressed
		5	0	0												accept and
CODING		6	8	/												processingut
200		7	2	A		L	Н	L	\mathcal{O}		8	3	A	/		
		8	A	/												(H)- Segment Count (L) ← Notor Control
		9	8	3												02 = Stop 00 = Run
		A	7	\mathcal{D}		Μ	0	V		A		L				,
		В	\mathcal{B}	7		0	R	A		A	Ĺ					
_		С	C	2		J	N	7			8	2	A	0		If stopped go to
STEM		D	A	0												display segment
SYS		E	8	2							ļ <u></u>					Count
		F	0	0		N		ρ						ļ		
TRAINING	8 29	0	C	D		C	A.	1	4		S	P	E	E	D	(HL) - instantaneous
RAI		1	\mathcal{C}	0												speed
ER T		2	8	2			<u></u>									,
JTE		3	E	5		P	٠.	2	14		#					(ST) - speed
MICROCOMPUT		4	C	A	<u> </u>	C	A	4	4		W	I	D	T	H	(HL) - pulse width
000		5	E	0	ļ				<u> </u>		<u> </u>					/
CRC		6	8	2	<u> </u>		_	_			<u> </u>	_		_		
Ξ		7	C	\mathcal{D}		C	A	1	4		4	D	7	/		Load Timer /
		8	6	0												for pulse width
		9	8	/			_			<u></u>	-			-	_	
JS.		Α		1	 	P	0	1		$\frac{\mathcal{L}}{\Delta}$	_	~		<u> </u>		(DE) - speed Output speed
TEN		В	1/	A	ļ 	M	0	<u>V</u>		/+	,	\mathcal{D}		<u> </u>		Output speed
SYS		<u>c</u>	D	3		0	u	L	_	P	0	R	7	1	\mathcal{B}	to D/A for
E R		D 	0	<u>5</u>				P		.,		,,			-	voltmeter
5		E F	0	0		N		7	/	×	C	1	G	-		XCHG to display
JMP			0	0		N	0	P		<u> </u>	├		 	 		instantaneous speed
ö	8	1	-		<u> </u>	1	_			7	4 1		_	7		
TEL	_		 		_	C		N	/	I	<i>\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\</i>	<u>ч</u>	E	<u>ب</u>	-	
3RA	-	3			<u> </u>	_					_					
INTEGRATED COMPUTER SYSTEMS		<u> </u>			 	-		-		-	_					
Z	li .	4 5	-		-		-		-	-						-
	-	6	\vdash		-	-	\vdash	-	-		-	-	-		-	
	-	7 .	\vdash		 	-		\vdash				-				
		8	-			\vdash			-		-					Figure 7-21ë
	L			<u> </u>	Щ	<u> </u>	<u> </u>	1	L	<u>L</u> _	<u></u>	Ц	<u> </u>			7-71

	A D D	R	со	DE	MOT	OR	COI	NTR	OL	_	MA]	ľΝ	LO	OΡ	– 1	DISPLAY
,	8 2A		7	0		M	0	V	_	A		14				In running display
i		1	C	D		1	A		7	//_	カ	B	y	7	E	Sprunning display
		2	9	س			-		-						_	
		3	1	2									-			Segment Count
SHEET		4	E			P	0	P		14	. –					(H2) told and speed
		5	7			М	0	V		1		7				() July of the second of the
CODING		6	2	A		1	H	_	D	//_	8	3	A	5	_	(HL) - latest
OD		7	A	مح			,				-	_				
O		8	8	3											_	land interpreta
		9	R	D		C	M	P	_	1						from interrupts Test for equal
		A	C	4		1	N	Z		D	W	0	R	D		
		В	D	ノ												Display Everage
		С	0	2												again of another
ΕĞ		D	C	3		7	M	P		8	2	8	0			Pack to start
SYSTEM		E	8	0											_	
		F		2												
TRAINING	8 2 B	0					-									82BO-BF not used
ΑĀ		1													-	
		2							_							
MICROCOMPUTER		3														
۱PU		4			-											
CO		5												-		
RO		6														
MIC		7														
		8														
		9														
(A)		Α														
SYSTEMS		В														
YST		С														
		D														
NTEGRATED COMPUTER		E														
MPL		F .	_			_					L.				_	
8	8	0	_			L	L									<u>.</u>
9		1	<u>L</u>	<u> </u>		L	_									
RAT		2	_							<u> </u>						
EGF		3	_					L_								
IN		4				_		_							_	
		5													_	
	 	6	_			_									_	
		7	_				_	_								
		8					ŀ									Figure 7-21f

SUBROUTINE SPEED

,	A D D	R	COI	DE						J 1 1						
	8 2C	0	0	E		M	V	I		C		0	0			Loop Counter
l		1	0	0							_					
	_	2	2	A		4	\mathcal{H}	L	\mathcal{D}		8	3	A	7)
<u>_</u> [3	A	7												(DE) - Lines 2 Data
EET		4	8	3												
R		5	E	B		X	C	H	G)
Z	82C	6	0	9		$\overline{\mathcal{D}}$	\mathcal{C}	R		C						Count tries and
CODING		7	C	8		R	Z									oxit it not moving
	··· <u>-</u> ·	8	2	A		L	\mathcal{H}	L	\mathcal{D}		8	3	A	7		exit if not moving (HL) - Jimer 2 Data
		9	A	7												Constant of Files
1		A	R	3												
ı		В	7	B		M	0	1		A		F				N
		С	9	رح		S	_	\mathcal{B}		Ź	1	_				(HL) - initial time
Σ		D	6	F		M	'n	<u>~</u>		7		D				- new time
STEM		 E	7	A		M	7	1		1	-	\mathcal{P}		-		= segment interval
S.		F	9	C		S	R	\mathcal{B}		H	1	4	_			- segren interva-
AINING	8 2 D	0	6	7		M	000B0RZ	1		#		A				 }
A	· ac v	1	B	3		0	P	A	-	7	1	71			-	1 + - 1 1
T.B.		2	C	A		2	7	//		8	2	C	6			Test and loop if time not changed
ER	·	3	0	6		٥	٦		-	0	α	<u> </u>		<u> </u>		ume not changed
5		4	8	2			!			-					-	
MICROCOMPU		5	0	1		1	X	I	†	$\overline{\mathcal{D}}$		1	2	E	8	0250/4:
8		6	É	8		=	^	-	 	2	/			_	0	= speed as XX.XX
길		7	0	3		-	-	<u> </u>	ļ	 	_	-		-		- speed as xx.xx
≥		8	C	D	<u> </u>	C	A	/	/		\mathcal{D}	I	V	I	\mathcal{D}	(41)4-(20)/(41)
í		9	A			1	' /	-	-		$\overline{\mathcal{O}}$	_	V	1		(HL)←(DE)/(HL)
		A	8	1		\vdash			-	-		_	_		-	-speed
MS		В	3	7	l 	S	7	C								M 11- 24 11 4
3TE		C	C	9		1	E	T	ļ —	-			-	 		Mark ascorrect result
SYS		D	10	7	 	R	1	 	\vdash	\dagger	-	-	-		-	result
ER		E	 		<u> </u>	\vdash	-		-	-		_	-	-		
INTEGRATED COMPUTER SYSTEMS	-	F	\vdash			-	-	-	-	 		<u> </u>	-	-	-	
JMC	8	0	╁		<u> </u>		+	-	\vdash	├						
ŏ	-	1	-		<u> </u>	-	\vdash	-	+-	\vdash	-			+	-	·
TEI			\vdash	-	 	-	-	-	 	├	_		-	-	-	-
3RA		2	+			\vdash	\vdash	\vdash	\vdash	-	-	-	-	\vdash	-	
TEC		3	 		 	1	┼─	-		1-			-			
Z		4	\vdash			-	├-	-		-	-		-	\vdash	-	
		5.	-	-	 	\vdash	┢	├	\vdash	├-	-		-	-	-	
1		7	┼		├	-	\vdash	\vdash	\vdash	├—	_		-	-		
			1-		 	\vdash	├	<u> </u>	\vdash	\vdash			-	_	-	7 21-
		8	1	<u> </u>	<u> </u>	Ц.	<u>_</u>	Щ	<u> </u>	1	L			<u></u>		Figure 7-21g

	A D D	R_	col	DE	MOTO	OR	COI	1TR	OL	-	SUE	BRO	UT	INE	W]	IDTH
	8 2 E	0	7	D		M	0	1		A		L				(HL) = Speed
		1	/	7		R	A	L			7					
		2	3	A		L	D	A		8	3	A	9			MSB of low byte to CY for rounding (A) desired speed
_		3	A	9									-			(A) Legized speed
SHEET		4	8	3												
- 1		5	9	C		S	\mathcal{B}	B		H						
CODING		6	5	F		M	D	V		E	,	A				(DE) - speed error
톓		7	9	F		S	B	\mathcal{B}		A	7					as two lytes
		8	5	7		M	D	1		\mathcal{D}	/	A				
		9	0	5		P	V	S	H		D					Save error
- 1		A	3	A		۷	\mathcal{Q}	A		8	\mathcal{S}	A	A			(A)← G;
		В	A	A							,					(integral gain)
		С	8	3												
EM		D	d	A		L	H	4	\mathcal{D}		8	B	A	\mathcal{C}		(HL) ← old integral
SYSTEM		E	A	\mathcal{C}												
		F	8	3												•
AINING	8 2 F	0	C	D		C	A	L	۷		S	ک	ט	M	L	$(HL) \leftarrow (HL) + (A)(DE)$
RAII		1	9	3												= S G; E
1-		2	8	/								-				
TER		3	D	/		P	0	P		D						(DE) - Error
MICROCOMPUT		4	B	4		0	R	A		H						
CO	•	5	۲	8		R	M	I	\mathcal{N}	U	S					negative
RO		6	2	2		S	1	L	D		8	3	A	C		Quit if integral negative Store integral
MIC		7	A	\mathcal{C}												
		8	8	3												
		9	3	A		L	D	A		8	3	A	\mathcal{B}			(A) ← Gp (proportional gain) (HL) ← (HL) + (A)(DE) = SG; E + Gp E
(0		Α	A	\mathcal{B}												(proportional gain)
SYSTEMS		В	8	3									l			
YST		С	C	\mathcal{D}		C	A	1	L		S	C	U	M	L	(HL) - (HL) + (A) (DE)
R S		D	9	3									_			= SG; E + GpE
INTEGRATED COMPUTER		E	8	1												,
MPL		F	C	9		R	E	7								
Ö	8	0							<u>_</u>							·
ED		1		_				L							_	
3AT		2														
EGF		3							L_	L.	<u> </u>	<u> </u>				
Ν		4										L				
_		5	<u> </u>	<u> </u>		_	_	L	_	1	<u> </u>					
		6	ļ.	L.,		_	<u> </u>	_	_	_	<u> </u>	_	_		_	
		7	-	_		_	_	_	<u> </u>		_		_	_		
		8]	L.,			Ŀ	<u> </u>	1							Figure 7-21h

<u> A D D</u>	R	ÇOI	DE	MOTO	R	CON	1.T.K	ΟĽ		<u> </u>	TM.				
8 10	0	0	0		N	0 A	P								for DI during debug VA)← command (HL)← beyboard data
	1	C	\triangleright		C	A	L	۷		E	N	7	W	D	VA) - command
	2	4	6							_					(HL) + Senboard data
-	3	0	3												
SHEET	4	E	\mathcal{B}		X	\mathcal{C}	\mathcal{H}	G							(DE) - keyboard data
	5	2	/			X			14	_	8		0	8	address dispatch table
SNIGOOD NO.	6	0	8												
	7	8	/												
	8	8	5		A	D	$ \mathcal{Q} $		7						
	9	6	F		M	0	ν		1	ì	A				Dispatch address
	A	6	E		M	0	V		7	,	M				Dispetch address to stack
	В	E	5		P	U	S	Н		H					
	С	E	\mathcal{B}		X	C	H	G						•	(HL) - buyboard data
EM	D	C	2		C	CA	L	7		$\overline{\mathcal{D}}$	E	C	B	I	(E) + binary of (L)
SYSTEM	E	8	0												(D) =00
	F	8	/												Bern cet id (E) = 00
8 //	0	7	A		M	0	V		A		D				Zero set if (E) = 00 Clar A, presure flag
Y Y	1	F	B		E	I				1			-		Reeded during deline
H -	2	3	2		S	エフ	A			8	0	0	0		Reeded during deling Clear log address
E E	3	0	0				-			-					to start a new log
5	4	8	0		_										7
MICROCOMP	5	0	0		N	0	P								
8	6	0	0		N	0	P								
8 //	7	C	9		R	E	ァ								
8/1	8	3	0		M	E	M					_			Story agrica)
0-7-	9	7			R	E									Store gains Undefined
	A	1	7		À	7	G	R							Yla Redinal
SW:	В	2	3		S	T	E	P							It is metal
STE	С	2	7		R	U	N							_	Start motor
XS.	D	3	4		N	E	X	7							Let duty cueles
INTEGRATED COMPUTER SYSTEMS 8	Ε	1	7		\mathcal{B}	R	K								Undefined Stop motor Start motor Set duty cycle Undefined Clear count, stop
2	F	2	0		\mathcal{C}	1	R								Clear count stop
∑ 0 8	0				_									ļ —	1
	1	\vdash												-	
ATE	2	1													
GR	3	1			ļ										· · · · · · · · · · · · · · · · · · ·
N N	4	 				Ι-									
=	5				-						<u> </u>		\vdash		
	6					<u> </u>				_					
<u> </u>		1	-			†				\vdash	\vdash				
Į	7	1	1 .				ı				i	ı	ı	1	

	A D D	R	co	DE_	KYT	EM	-	CL	R,	ST	EP,	, R	UN	, M	EM,	, NEXT
	8 /2	0	3	2		S	T	A			8	B	A	2		CLR-Clear binary
		1	A	2												segment count and
		2	8	3												stop motor
ET	812	3	3	E		M	V	\mathcal{I}		A		0	2			STEP - Stop motor
ш		4	0	2							,				_	,
3 SH		5	0	3		0	U	\mathcal{T}		P	0	R	\mathcal{T}	/	\mathcal{C}	
SODING	L	6	0	6												
00	8/2	7	3	2		S	\mathcal{T}	A			8	3	A	1		RUN-Store motor
		8	A	/					_							Control legte
		9	8	3												DO = RUN 02 = STOP
		Α	0	8		R	Z									Exit if no data
		В	7	B		M	0	V		A		E				Store binary
_		С	3	2		S	7	A			8	3	A	9		equivalent of
SYSTEM	 	D	A	9		_										desired speed
.XS.	·	E	8	3		_										,
		F	C	9		RS	E	7								
RAINING	8 /3	0	2	2		S	/+	L	\mathcal{D}		8	3	A	A		MEM-Store gains
RAI		1	A	A												
Я		2	8	3												
щ		3	C	9		R	E	7								
MICROCOMPUT	8/3	4	3	E		M	V	I		A	,	3	2			NEXT - Duty cycle Multiply benory duty cycle (DE)
000	ļ	5	3	2			_				_		_ :			Multiply binary duty
SRC		6	C	D		C	A	L	L		S	M	U	<u>L</u>	7	cycle (DE)
ž		7	9	0								Ŀ.				× 32 for pulse width
		8	8	4		_					<u> </u>		_			
		9	2			S	#	<u>L</u>	D		8	3	H	\mathcal{C}		Store as integral of error Load Limer! for pulse width
ភ		<u> </u>	A	C											<u> </u>	of error
SYSTEMS		В	8	3									_		<u> </u>	
.XS	<u> </u>	C	C	2)		C	A	L	2		4	D	7	/	ļ [,]	Load Lines
8		D	6	0		-								-	ļ	for pulse width
15	ļ	<u> </u>	8			0	_							_	-	
MP		F	C	9		R	E	1	<u> </u>	_				<u> </u>		
NTEGRATED COMPUTER	8	0	-				<u> </u>	ļ				_				
TEC		1	-	H				<u> </u>						_	-	
ìRA		2	-			-	_				-			_		
TEG		3	-	\vdash		\vdash	-	_		-	_	_			-	
Z		4	-	\vdash		-	-	_			-			-	_	
		5	\vdash	\vdash	<u> </u>	-	-			_		<u> </u>			-	
		7	┼─	\vdash		-	-		-	\vdash		\vdash		\vdash	-	
			-	\vdash		-	-		-	-	-	-	-	-	-	Figure 7-22b
		8	Ш.	L.	L	L_	Γ	<u> </u>	[L		L	<u> </u>	<u> </u>	F19016 1-44D

5 6 7

8

Figure 7-22c

	A_ D D	R	co	DE	MOTO	OR	COI	NTR	OL	-	DE	сві				
	8 / 8	0	7	D		M	0	V		A		/				Law lester of input
		1	E	6		A	N	τ		F	0					(A) to his Production
		2	Z	0			,,	<u>-4-</u>		Ť	_					(H): sugh orga
		3		F		R	A	R			_					Low leyte of input (A) ← high digit (A) ← 1/2 high digit (E) ← 1/2 high
닯		4	7	7		M	0	1/		E		A				(A) - 12 night diges
SHEET		5		7		P	A	P		=	7	//	-		-	(A) (1/1 1 1)
S		6	/			R	A	R			_	-				(A) ← 1/4 high (A) ← 1/8 high (A) ← dec + 1/8
CODING		7	0			A	\mathcal{D}	$\frac{\lambda}{D}$		L					_	(A) high
ŏ		8	9	3		<u>''</u>	U	$\frac{\mathcal{D}}{\mathcal{B}}$		E				_		(A), 1 3/2
1		9	5	-	<u> </u>	M	0	1				A				(A)← dec - 3/8 = binary (D)← 00
}						<u>141</u> M		I		12 12	1	-	0	_		- wary
- 1		В	/	6		181	V	<u></u>	_	ν	1	0	$\frac{\partial}{\partial x}$			$(v) \leftarrow \delta \delta$
}		C	0	9		0		· -						\vdash		
⋝		D	C	7		R	E	_		-	_		-	-		
SYSTEM		Ë	+-													
	· · · · · · · · · · · · · · · · · · ·		-	-											_	
Ŋ								7			_	4.1	_	7	11	
TRAINING	8	0				E	N		E	R	-	ω	I	7	$\overline{}$	T 70+0
TRA		2	-				(<i>Н</i> S	2	<u>ال</u>	=	エ	N	P	1	T DATA
			-			6	A	7		0	M	Ē		(1) = PACKED DECIMAL
5		3	 	-		R	E	1	5		7	_	$\overline{}$			
MICROCOMPUTER		4	├				(A)	=		Ε)	=		BINARY EQUIVALENT
읽		5	├ -				7	_	_			_	_			OF (L)
CR	ļ	6	[—				(D	<u>ر</u>	=	0	0		ļ		7075
Ξ		7	├				_	B	1	C	1	#	1	4	! 	PRESERVED
		8	 							_		_			ļ	:
		9	-	-		_	C			C	L	E			<u> </u>	
JS	<u> </u>		-				Z	E	R	0		3	E	T		IF (E)=00
TEN		В	<u> </u>													
χλS		<u>C</u>	\vdash					_		ļ		-				
INTEGRATED COMPUTER SYSTEMS		D	┼				_							_		
UTE		E	-	<u> </u>			<u> </u>	 	<u> </u>			ļ. <u>. </u>				ļ
MP		F	-	_		-	-		-	-		-	_		_	-
00	8	0	—	<u> </u>	<u>.</u>	<u> </u>			_	-				_		
ΓED		1	-	-	<u> </u>	<u> </u>	_	_		-				-	_	
RA			-	_			_	_	_	_				_	<u> </u>	
7EG		3	↓_	_	ļ	<u> </u>	-	-	_	-		_		_	_	
Z		4	\vdash		ļ	_		<u> </u>	_	_		_		-	_	
		5	<u> </u>	-		ļ			_				<u> </u>	-		
		6		-	<u> </u>	<u> </u>	_	_			ļ	_		-	_	
		7	.	ļ.,	<u> </u>	<u> </u>	_	<u> </u>	<u> </u>		_	_	_	_	_	
		8	┸_				<u> </u>							L		Figure 7-22d

	A D D	R .	CO	DE	MOT	OR	C	тис	ROI	ւ –	SN	I UL	Т,	sc	UMI	7 - 79
	8 19	0	2	/			X	エ		14		0	0	0	0	SMULT
	-1-1-	1	0	0			_				•					Clear product
		2	0	0									•			7
_	819	3	B	ク		0	R	A		A						SCUML Clear CY
EET		4	C	8		R	7									Eit when multiplier = 0
SHE		5	1	F		R	A	R								(CY) = LSB of multiplier
CODING		6	D	2		2	۸J	\overline{c}			8	7	9	A		This add it LSB=D
0		7	9	A				Ī			i	Ť				UISR = 1 add
٥		8	8	1		-										Exit when multiplier = 0 (CY) = LSB of multiplier Skip add if LSB = 0 If LSB = 1 add multiplicand
		9	7	9		D	A	\mathcal{D}	,	A						inter or reduct
	819	A	Ē	B		X		H	G							Shift multiplicand
	7	В	2	9	_				Ť	H						suft mucipulant
		С	E	B		X	d C	1	G	<u>'</u> •					-	
Σ		D	C	3			<u>)</u>				र	C	u	~	1	8 4 4 4
SYSTEM		E	9	3)	-			<u> </u>	3			1-1	-	Loop to test multiplier
		F	8	_						-					-	multipul
ING	8	0	0	/ -						-		-		-	-	,
TRAINING	-	1	 -						-	<u> </u>		-				
		2	\vdash			E	N	T	E	R	-	S	М	11	,	T OR SCUML
MICROCOMPUTER		3	1	-		<u>_</u>	/	A	5	=		M	U			I PLIER
PU		4	 				1	E	7	=		M	U	<u>ا</u>		IPLICAND
OM		5	 	-			7	5	5	=		0	0	<u></u>		R FF(NEGATIVE)
300		6					L	U	,	 -			o	-		K PP(NEGR) IVE
ic.		7	┢			$ \mathcal{R} $	E	T	()	R	11	-		_		
≥		8	 	-		7			υ		T	-	7	Ш	1)←(A)*(E)
	_	9	╂─	-			6	(M	-			# 1	7) = (HL)+(A)*(E)
		A	 			_	٦	<u> </u>	U	/* 1	<u> </u>		-	<u> </u>	_	$\mathcal{L}=(\mathcal{H}\mathcal{L})+(\mathcal{H}\mathcal{L})\times(\mathcal{L}\mathcal{L})$
MS		В	-				7	A	1	\vdash	=		0		_	
STE		C	 				己	E	R		Ë	0	E	7	-	CARRY CLEAR
SΥS		 D	╁	-			Z		7		-	၂	_	├	+	CHRK) CLEAR
INTEGRATED COMPUTER SYSTEMS		 E	-	-	-		7	B	C	7		P	R	E	S	ERVED
PUT		<u>-</u> -	-	-				ري		را		1	13	2	٦	LAVED
Ψ O	8	0	\vdash	\vdash					_	-		-		-	-	
D C		1	╁	-					_		-	-				
٨TE		2	┢							-		-			-	
GR/		3	╁─	\vdash							-			_	-	· · · · · · · · · · · · · · · · · · ·
ITE(4		-	[_		-		-		 		
2		5	-	<u> </u>	<u></u>					-	\vdash	-				
		6	 	-	ļ		_	-		-	-	-			-	
		7	 		ļ	<u> </u>				-					-	
1			+-				<u> </u>	ļ	<u> </u>	-	-	-	-	-	-	Figure 7-220
	L	8	1		L	<u> </u>	Ц.	<u> </u>	L	L		L		L	<u> </u>	Figure 7-22e

	A D D	R	со	DE	DIV	ID				(H	L) <	<	(DI	Ξ)/	(H1	.)
	8 / A	0	.3	E	_	M	V	I		A		1	0			(A) - bit element
		1	1	0				ŕ			-/-					The second
		2	C	5		P	U	S	Н		\mathcal{B}					Save (BC)
		3	4	4		M	0	V	•	B		H				(BC) ← divisor
EET		4	4	D		М	0	V		0	/	Ĺ				(ice) amas
SH		5	2	1		<u></u>	X	Ī		H	-	0	0	0	0	Class quatient
CODING		6	0	0			· `	_)					Clear quotient
OD		7	0	0				-						- 1		
ပ	8/A	8	E	رحی	DVI	P	U	S	Н	_	P	5	W			Save bit count
	-/	9	2	9		D	A			14		_				Shilt a street
		A	7	\mathcal{B}		M	0	V		A		E				Shift quotiest Test for
		В	9	1		5	U	B		Ċ	-) -	<u></u>				division State of the
		С	7	A		М	0	V		A		D				divisor > remainder
EM	\	D	9	8		S	B	<u> </u>	-	B	- /-	<u> </u>				
ST		E	7	A		2	C	Ť		Ž	D	V	a			Our of it
3 SY		F	B	6		Ť	_	-			U	_	-			Jump if
AINING	8 /B	0	8	1	•	-			-			-		_	-	divisor > remainder
A		1	2	3	-	I	N	X		Н		_		_		Entre / into quotient
ΗR		2	5	1		M	0	V		D		A				Enter / into quotient
TE R		3	7	B		М	0	1		A	1	E				(DE) - new remainder
P		4	9	1		5	υ	B		(+	_				(DE) - MED NEMBERGEO
MICROCOMP		5	5	F		M	0	V		E		A		-)
300	81B	6	E	B	DV2	X		H	G	=	+	1,4				
IIC.	0.0	7 .	2	9	275	D	A	D		Ĥ		-		-		(It It have in day)
~		8	E	B		X	C	4	G	Ė						Shift remainder
		9	F	/		P	0			P	5	W			-	(A) ← bit count
		A	3	D		D	6	R		A		-				Company Comments
SYSTEMS		В	0	2			N	Z		<u> </u>	D	V	1		-	
STE		С	A	8	,	<u> </u>	-				_	Ė	<u> </u>			
		D	8					-								
TEF		E	Č	1		P	0	P		B						
Ē		F	C	9		R	E	7		_		_				
Š	8	0														
0		1			<u> </u>	E	N	T	E	R	-	1	0	Ē		= DIVIDEND
ATE		2										1	#	1)	= DIVISOR
GR		3					a	工	V	I	D	E	N			<2*DIVISOR
INTEGRATED COMPUTER		4				R		_	Ú	R			(H	L) = QUOTIENT
=		5					A				X	X	X	×		
		6											(D	E)= REMATNDER
		7											(B	$\overline{}$)- PRESERVED
		8					1	A	$\left(\right)$	=	0	0		Z		RO SET. CY PRESERVE
7_	_ 80						-									Figure 7-22f

7.3.6 Motor Control Program Operation

To operate closed loop control you must enter three data bytes; proportional gain, integral gain, and desired speed. Gains are entered as two consecutive bytes with MEM.

0801 MEM Set proportional gain = 8

Set integral gain = 1

50 RUN Set desired speed 50 rps

The system allows speed requests from one to 99 rps, but will not operate successfully at speeds much less than 10 rps. Experiment to see the average speed respond to various speed requests. Connect the voltmeter from the D/A output to ground to see an analog indication of instantaneous speed. Although the original program design intent was to display instantaneous speed, this varies so much and so quickly that it is not very readable, so a display of the pulse width is generally more interesting. In the given solution this is obtained simply by omitting an XCHG instruction at 829D. An oscilloscope display of the pulse width is especially interesting.

Observe the response of average speed as you place a load on the motor. This is most easily done by squeezing the motor shaft between your fingers. Do not attempt it by dragging on the optical disc, because if you start it slipping it will wear the hole that mounts it on the shaft.

When you apply the load, the motor will slow down but the closed loop control will apply more power to restore the speed. The range of loads over which speed can be maintained is fairly impressive, although unfortunately we have no means of measuring load.

When you release the load the motor will speed up rapidly, and the control system will hunt for the desired speed just as the voltage control system hunted for a desired voltage. The hunting is easily observed from the sound of the motor. Try different gain values and observe their effect.

If you stall the motor by holding the shaft it will not restart. This is because the integral will increase to a large positive value while the motor is stalled. In following cycles of the main loop a value greater than 7FFF will be calculated for the integral; this is taken to be negative and will not be stored. LDT1 will substitute a minimum pulse width which will not start the motor. You can restart the motor by pressing NEXT, which clears the error integral, allowing closed loop control to function again. SPEED returns to the main loop with carry clear if the motor is stalled. Develop a program modification to enter a 50% duty cycle if the motor is stalled.

The CLR key was defined to stop the motor and clear the binary segment count. This allows observation of the coasting distance after power is removed. Measure the relationship between speed and coasting distance.

7.4 MOTOR CONTROL BY VARIABLE VOLTAGE

We can control the motor by varying the voltage amplitude instead of using pulse width modulation. The output voltage from the power transistor can be controlled by varying the drive to its optical coupler. Remove the connection from MOT CTL+ to +5 volts, and connect ANALOG OUT to MOT CTL+. Now the program can vary the voltage. Only two trivial changes to the program are required:

- a) Timer 1 has no function, since power is to be turned on whenever the motor is running. Disable the Timer 1 inerrupt by changing the initialization step that originally enabled it.
- b) Output the high byte of pulse width to the D/A converter instead of the high byte of speed.

These two changes are shown in Figure 7-23.

MOTOR CONTROL

Now run the program as before.

0801 MEM Set gain

50 RUN Request speed

Before the motor will start the integral must build up to a much higher value than for pulse width modulation. This is principally because the optical coupler that drives the power transistor must receive a voltage input above about 1.5 volts before it will start to turn the power transistor on. The control will not be as smooth, because the optical coupler makes a much larger change in the motor voltage. Try different speed requests and see how small a change in the control voltage is required. Connect your voltmeter across the motor and observe that the very small change in control voltage displayed by the computer makes a much larger change in the motor voltage.

You can also switch between PWM control and voltage control by Keyboard command. Define REG to enter a speed for variable voltage control; RUN to enter speed for PWM operation. REG must disable Timer 1 interrupt; RUN must enable it. During PWM operation output FF to the D/A converter, allowing PORT1C1 to control the motor. One possible solution is given in Figure 7-24.

	A D D	R	со	DE								_					
I	8 2 2	0	3	E		M	V	\mathcal{I}		A			/			Enable EXT4	
		1	/	/	*											and Timer O	
		2	D	3		0	u	7		P	0	R	7	2	<u></u>	(not Times 1)	
⊢ [3	0	E												(not Timer 1)	
SHEET		4															
		5															
S		6															
CODING		7															
		8	-														
		9			_					`							\neg
		A														,	╗
	829	В	7	C	*	М	Δ	V		A		\mathcal{H}				Voltage Control	\neg
	021	С	Ź	3		0		7		P	7	P	7		B	("Pulse Width"	7
M		D	0	5		-	0,			/				/	,		
SYSTEM		E				 										(instead of speed to D/A output	
SΥ		F	-			-										is 4/11 outpur	
TRAINING	8	0			*	6	4	A	٨/	G	E	5		F	R	OM PUM	-
AIN		1	_		~	P	D	<u>//</u>	6) Q	<u>_</u>	N		/	7	OM PWM IGURE 7-21)	
TR		2	_			/_		D		1/	α	0	7	7	<u>ر</u>	LE VOLTAGE	
ER.		3	-						0			0	1	7	7	ROL	
PUT		4	_	\vdash	 _	/* 1	0	/	\mathcal{O}				U	/		X 0 2	一
OM		5				6		Δ /	N	E	0	7	_	М		T CTL+	
300		6				τ	0	70	A	1	<u> </u>	/		(+	ر	OUT	
MICROCOMPUTER		7				I	N	<	7	F	A	7)			<i>F</i> =	0UT +5V	
2		8					/-)	•	<i></i>							
		9	-														
		Α				\vdash	-	-	_				_	_			
MS		В	<u> </u>			 	1								-	<u> </u>	
SYSTEMS		С	-	\vdash													
SΥ	-	D			_ _	\vdash											
TEA		E			_					-							
PU		F										_					
ΝO	8	0				\Box											
0.0		1				\vdash		<u> </u>									
ATE		2	-			T	<u> </u>	—						-			
GR		3	-			\vdash	\dagger		-	-							
INTEGRATED COMPUTER		4	-			\vdash	<u> </u>		<u> </u>								
=		5	 		_	\vdash			-	-							
		6	<u> </u>				T	 -	_	-							
		7					<u> </u>	-									
		8				1				-		_			_	Figure 7-23	
	<u> </u>					٠.		Ь—	Ь—			Ь	-	Ь		705	

	A D D	R	co	DE	,								U 1. F.			
	8 /4	0	3	E		Μ	V	I		A		8	0			RUN = PWM
		1	8	0												_
	814	2	3	2		S	7	A		8	3	A	/			(Store OO or FC)
Ţ.		3	A	/												(Store OO or FC)
SHEET		4	8	3		_										
G S		5	/	7		R	A	۷								(A) ← 03 if RUN (A) ← 02 if REG
CODING		6	3	E		M	ν	I		Д	_	0	/			(A) - 02 if REG
ပ္ပ		7	0	/		_	_			_			_	_		
	<u> </u>	8	/	7		R	A	<i>L</i>								
		9	5	3	-	D	I					_	_			Enable or Disable
		A	2	3		0	u	1		C	N	7	Q		_	Times / Interrupt
		B	0	1-		_		_								0
Σ		<u>c</u>	C	0	ļ	ER	IZ								-	Exit if no speed entered
SYSTEM		D E	-	8		M	_	./		0		_	Н		_	entered
SY	·	F	7	B 2	l	3	0	V		А	8	<u>E</u> 3	9	9		Store bixary equivalent of blesired speed
N.	0 / 4-		3 A	9		7	/_	77			8	ح	A	7		equivalent of
Ž	8 /5	1	8	3	<u> </u>	_	<u>-</u>							_	-	desired splea
TRAINING		2	C	9	<u> </u>	R	E	7	_							· · · · · · · · · · · · · · · · · · ·
		3		7	<u> </u>		<u>L</u>	/	_							
ΤŢ		4	-			-										
MICROCOMPUTER		5				\vdash									_	
300		6	[-		P	A	T	1	Н	F	.0		T	0	DISPATCH
10.	<u> </u>	7	-	-		 	7	A			E)	I	N	_	KYTIM
2		8	-	-		-	-	<u>''</u>	U	_	<u> </u>	_			-	17.72.
	811	9	4	2		R	Ε	G								
		A	Ť			1	_	-								
EMS		В														
/STI	811	С	4	0		R	u	N								
S.		D														
TEI		E														
MPU		F	L_			_										
INTEGRATED COMPUTER SYSTEMS	3	0	igspace			Μ	0	7	0	R		C	0	N	7	ROL BYTE FINED AS
ËD	ļ	1	<u> </u>	ļ	ļ	_	I	S	_	Ν	0	W	-	D	E	FINED AS
RAT		2	<u> </u>	_	ļ		ļ									
ĒĞ		3			ļ		0	2	_	=		S		_	P	(CLR or STEP) (RUN)
Z		4	-	-		-	8	0		=	_	P	W	M	_	(RUN)
	<u> </u>	5	-	-			0	0	-	=		V	0	1	T	AGE (REG)
		6	-	-		-		-	-			_	-	-	_	
	<u> </u>	7	-	-	 	1-	-	-			\vdash	-	-	-	-	Figure 7-24a
	L	8	<u> </u>		<u> </u>	<u> </u>	L	1	<u> </u>	<u></u>			L_	<u></u>	<u>L</u>	1

8 28 0 E 5 PUSH H H Sare arg Speed 1 DB TN PORTOR 2 00 RT OA 3 3C TNR A R TM 4 CH CNZ KYTIM 5 00 C R Z KYTIM 6 8 1 L HLD P3A 1 (We Signest Count (We flator Control 9 83 MOV A L 80 = PWM C C2 JNZ R DD A Set CY if PWM C C2 JNZ R DD A Set CY if PWM C C2 JNZ R PSW D A0 R PSW E P2 F F S * PUSH PSW S P E E D (HL) instantaneous aped 1 C0 CALL SPEED (HL) instantaneous aped 1 C0 CALL WIDTH (H) - Pulse Width 5 E0 6 P 2 CALL LDT / Soad Lines / 8 G0 F * SBB A H JODE R OF MAIN LOOF 1 C S * POR A D F ROM FIGURE 7-212 8 C CAMP A C S E T CH DE TAIL		A D D	R_	COL	DE.							· <i>,</i>					
DB ZN PORTOR	Į	8 28	0	E	5		P	U	5	\mathcal{H}		H					Save avg Speed
3 3 C	l		1	0	B		I	N			P	0	R	7	0	A	, ,
4 C H C N Z K Y T I M 5 0 0 0 6 8 1	- [ı	2	0	0												
5 00 0			3	3	C		I	7	R		A						
6 8 /	끨		4	C	4		C	N	3			K	Y	7	I	M	
8 A / O			5	0	0												
8 A / O	Z		6	8	/												
8 A / OP HOV A, L 80 = Pwm 8 B 7 * A D D A Jet CY if Pwm C C2 JN Z B 2 A D Zero unlies stop D A D Sero unlies stop E 82 P S * PUSH PSW Save CY of Pwm 1 CD CALL SPEED (HL) = instantaneous speed 1 CD CALL WIDTH (HL) = Pushe Wistle 2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2	Ö		7		A		L	1+	۷	\mathcal{D}		8	3	A	/		(H) - Segment Count
A 7D MOV A, L 80 = RWM B 87 * ADD A Lt CY if RWM C C2 JN Z P2 AD Zerr incline stop D AD JO Serr incline stop D AD JO Serr incline stop D AD JO SER DO JONE STOP B 29 0 CD CALL SPEED (HL) = instantaneous speed 1 CO CALL SPEED (HL) = instantaneous speed 2 P2 3 OO * NOP 4 CD CALL WIDTH (HL) = Pulse Width 5 EO OC ALL DT / Load Inner 6 P2 CALL LD T / Load Inner 9 P / LOAD CALL LD T / Load Inner 9 P / LOAD CALL LD T / Load Inner 9 P / LOAD CALL LD T / Load Inner 9 P / LOAD CALL LD T / Load Inner 9 P / LOAD CALL LD T / Load Inner 9 P / LOAD CALL LD T / Load Inner 9 P / LOAD CALL LD T / Load Inner 9 P / LOAD CALL LD T / Load Inner 9 P / LOAD CALL LD T / LOAD CALL LD T / LOAD CALL 1 CO CALL LD T / LOAD CALL 1			8	A	1	_											(L) - Motor Control
B P 7 * A D D A Set CY if PWM C C 2 JN Z 8 2 A O 3err unlies stop D AO E P2 F F 5 * PUSH PSW Swe CY if PWM 1 CO C A L L SPEED (HL) = instantaneous speed 1 CO A C D C A L L WIDTH (HL) - Pulse Width pro 6 P 2 7 C D C A L L L D T / Load Inner 8 6 0 9 P / Load Inner 9 P /			9	8	3												02= Stop 00= Voltage
B 8 7 * ADD A Set CY if PWM C C2 JN 2 P2AO Zero unlise stop D AO D AO D AO SE P2 F F5 * PUSH PSW Save CY if PWM 8 29 0 CD CALL SPEED (HL) = instantaneous speed 1 CO SECURITY SPEED (HL) = instantaneous speed 1 CO SECURITY SPEED (HL) = instantaneous speed 4 CD CALL WIDTH (HL) = fulse Width or or or or or or or or or or or or or			Α	7	\mathcal{D}		M	0	٧		A		L				80 = PWM
C C2 JNZ P2AO Zero unlise stop B AO B AO C C2 JNZ PAO B AO B AO C AC L SPEED (HL) = instantaneous speed C AC L WIDTH (HL) = fulse width C AC D CALL DT / Soad Lines / C AC D CALL LDT / SOAD / C AC D CALL LDT / SOAD /			В	8	7	*	A		D		A	1					Let CY if PWM
D AO E 92 F F 5 * PUSH PSW Save CY of PWM B 290 CD CALL SPEED (HL) = instantaneous speed 1 CO A CD CALL WIDTH (HL) = fulse Width or 6 P2 7 CD CALL LD71 Soad Since / 8 60 7 CD CALL LD71 Soad Since / 8 60 A OO * NOP B F / * POP PSW Yolfage Control 9 P/ A OO * NOP B F / * SBB A H J NDER OF MAIN LOOF 1 REMAIN DER OF MAIN LOOF			С		2		1	N	-			8	2	A	0		
SWELLEY F S * PUSH PSW Save CY if PWM 8 29 0 CD CALL SPEED (HL) = instantaneous speed 2 P2 3 00 * NOP 4 CD CALL WIDTH (HL) = Pulse Width 5 E0 6 P2 7 CD CALL LDT/ Soad Inver 8 60 9 P/ A 00 * NOP B F/ * POP PSW Yorkage if rollage if rollage control D B4 * ORA H JODA autput E D3 * OUT PORT/B for MOT CTL+ F 05 * CHANGED FROM FIGURE 7-21e 1 REMAINDER OF MAIN LOOF 5 (P2 AO - P2AF)	ΕĞ		D	+-													0
SWEET S * PUSH PSW Save CY if PWM speed 8 29 0 CD CALL SPEED (HL) = instantaneous speed speed 2 P2 SPEED (HL) = instantaneous speed speed 2 P2 SPEED (HL) = instantaneous speed speed 2 P2 SPEED (HL) = fulse (Width or Speed 2 P2 SPEED (HL) = fulse	YST		E	8	2				_						_		
1 CO		· · · · · · · · · · · · · · · · · · ·	F	F	5	*	P	u	S	Н		P	S	V			Some CY of PWM
1	Z	8 29	0	C	D		C	A	1	4		S	ρ	E	E	D	
	3AII		1	C	0											,	speed
3 0 0 * N 0 P 4 CD CALL WIDTH (HL) = Pulse Width 5 E0 6 8 2 7 CD CALL LD7 / Load Linew / 8 60 9 P / A 00 * N 0 P B F / * P 0 P P SW Voltage if C 9 F * SBB A D B4 * ORA H E D3 * OUT PORT / B for MOT CTL+ F 05 * 8 0 8 0 1 REMAIN DER OF MAIN LOOF 5 (8 2 A0 - 8 2 A F)	μ,		2	8	2	_			-								
8 6 0	TEF		3		0	*	N		ρ								
8 6 0	ΛΡU		4	C	D	_		A	L	_		W	I	D	7.	Н	(HL) = Pulse Width
8 6 0	CO		5	E	0												כזא
8 6 0	RO		6	8	2												(H) - Variable Voltage
8 6 0 9 9 / A 00 ** NOP B F / * POP PSW Voltage if C 9 F ** SBB A voltage control D B4 ** ORA H E D3 ** OUT PORT / B for MOT CTL+ F 05 ** 2 ** CHANGED FROM FIGURE 7-2/2 3 1 REMAIN DER OF MAIN LOOF 5 (82 A0 - 82 AF)	MIC		7	C	\mathcal{D}		C	A	L	L		L	\mathcal{D}	7	/		Load Times!
9 8/ A 00 ** NOP B F/ ** POP PSW 7 Voltage if C 9F ** SBB A Voltage control D B4 ** ORA H JoD/A output E D3 ** OUT PORT / B for MOT CTL+ F 05 ** 2 ** CHANGED FROM FIGURE 7-2/2 3 4 ** REMAINDER OF MAIN LOOF 5 (82 AO - 82 A F)			8	6	0												Low on time
A OO * NOP PSW PLAGE IF B F / * POP PSW Poltage if C 9 F * SBB A POP PORT I B for MOT CTL+ F O 5 * OUT PORT I B for MOT CTL+ F O 5 * C HANGED FROM FIGURE 7-2/2 3 REMAIN DER OF MAIN LOOF 5 (82 AO - 82 AF)			9	8	/												
B F / * POP PSW 7 Voltage if C 9F * SBB A SD/A output B D B4 * ORA H JoD/A output E D3 * OUT PORT / B for mot CTL+ F 05 * OF MAIN LOOF A REMAINDER OF MAIN LOOF	"		A		1 _ 1	*	N	0	P) FF is PWM
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	EMS		В	1	/	*	P	1	P		9	S	\mathbb{V}				1/1/
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	/ST		С	9	F	*	S	B	B		A						V voltage control
F 05 ** CHANGED FROM FIGURE 7-212 A REMAINDER OF MAIN LOOF 5 (82AO-82AF)			D	\mathcal{B}	4	1					H						To DIA output
2 * CHANGED FROM FIGURE 7-2/2 3 REMAINDER OF MAIN LOOF 5 (82AO-82AF)	핕		E	1	3	*	0		T		P	0	R	7	1	\mathcal{B}	for mot CTL+
2 * CHANGED FROM FIGURE 7-2/2 3 REMAINDER OF MAIN LOOF 5 (82AO-82AF)	MP∪		F	0	3	*											
2 * CHANGED FROM FIGURE 7-2/2 3 REMAINDER OF MAIN LOOF 5 (82AO-82AF)	S	8	0														
5 (82AO-82AF)	ED		1														
5 (82AO-82AF)	₹AT		2			-*	C]+	A	N	G	E	\mathcal{D}		F	R	OM FIGURE 7-2/2
5 (82AO-82AF)	EGF		3														
5 (82AO-82AF)	N		4				R	E	M	A			D	E			OF MAIN LOOP
	_		5						1	2	A		_	8		A	F)
OMME PAIS			6			ļ	S	A	Μ			A	S	<u> </u>	F	I	GURE 7-2/4
7			7				<u> </u>										/
8 Figure 7-24b			8								_			L			

7.5 POWER TRANSISTOR DISSIPATION

A transistor dissipates (wastes) much more power when operating in its linear region than when it is used as a switch. You can measure this effect by attaching the thermistor to the power transistor and measuring its temperature. You must time share the digital/analog converter between the driving and measuring functions. Each time an completed, make a temperature average speed measurement is measurement. Use a tracking voltmeter approach, storing the D/A value in memory and trying only one value on each attempt. This gives the least interference with the motor drive. After the temperature measurement, restore the motor drive value to the D/A converter. Figure 7-25 shows a program solution for the Main Loop and a new subroutine to display speed and measure temperature. In the given solution the thermistor voltage is displayed in hexadecimal, with no conversion to degrees.

	A D D R		COL	DΕ												·
	8 2 A 0		4	7		M	0	V		B		A				(B) - Motor Drive
	1		7	D		M	0	V		A	,	<u>L</u>	_		, _	(A) - Old and speed
	2		2	A		L	/+	L	\mathcal{D}		8	3	A	3		(A) - Old and speed (HL) - Latist
H	3		A	5												average speed
SHEET	4		8	3												0,
	5		\mathcal{S}	<u> 2</u>]		\mathcal{C}	Μ	P		上					_	
CODING	6		0	4		\mathcal{C}	N	Z			S	P	7	M	P	Lynew average
. <u></u> 00	7		2	0						Ш						speed, display
	8		8	/		_										Speed, display speed of temperature
	9		2	3		J	M	ρ			8	2	8	0		,
	A		8	0			_									
	В		8	2												
_	c															
SYSTEM	D						_								_	
YS.	E									_						
	F						_								<u> </u>	
TRAINING	8 0			_		R	E	ρ	۷	A	C	E		7	A	ST PART OF ROL MAIN LOOP -21/2)
RAI	1					M	0	τ	0	R		C	0	\mathcal{V}	T	ROL MAIN LOOP
	2						1	F	I	G	u	R	E		7	-21/2)
JTE	3					<u> </u>	_									
MICROCOMPUTER	4					S		7	M	P		(F	I	G	URE 7-25/2)
000	5					_	D	\mathcal{I}_{μ}	2	ρ	_	A	У	3	<u> </u>	AVERAGE SPEED
CRC	6						A	N	1	_	Н	7	\mathcal{T}	E.	M	PTS A/D NT OF
Ξ	7					_	M	E	A	3_	U	R	E	M	E	NT OF
	8		_			<u> </u>	<u>T</u>	E	M	P	E	R	A	7	4	RE
	9									_				_		
JS.	A					_	_	_								
SYSTEMS	B		_		-					_					-	· · · · · · · · · · · · · · · · · · ·
3YS	<u> </u>	-					-			<u> </u>						
	D					_	_				_		_			
COMPUTER	E		_	•		_	-	_		-						<u> </u>
MP	F						-								-	
S	3 0			_		-	-	_			<u> </u>	_	_			
TEC	1		_			├-	-		_			_			-	
3RA	2	-				<u> </u>	-	ļ	-	-		_				·
NTEGRATED	3					-	-	_	_			-		_	-	
Z	4					<u> </u>	-	-		-				_	-	
	5	+					-	-		-	_	_	 -	-		
	6	$\overline{}$				-	-	-	-	<u> </u>	<u> </u>	<u> </u>	-	-	-	
	7	-				-	-	-	-	-	-	_	-	-	-	Figure 7-25a
	8				L		<u> </u>	L			<u> </u>	<u>_</u> _				7.00

r	A D D	R_	со	DE												
	8 / C	0	3	A		1	\mathcal{D}	A			8	3	A	E		(A) - Previous
		1	A	E												temperature
		2	8	3												measurement
⊢ [3	D	3		0	4	\mathcal{T}		P	0	\mathcal{R}	7	/	(S	to DIA
IEET		4	0	S	_											
SH		5	4	A		M	0	V		C		A				Save value
Ž.		6	3	C		I	N	R		A	7					Increase for
CODING		7	3	2		S	7	A			8	3	A	Ε		next trial
Ĭ		8	A	E						`						
		9	8	3												
Ī		Α	C	5		P	U	S	Н		\mathcal{B}					
Ī		В	C	D		C	A	4	7		7)	W	0	R	\mathcal{L}	Dielay average
Ī		С	D	/							~			, \	_	and and
Ξ		D	0	2												a partie
SYSTEM		E	C	1		P	0	P		\mathcal{B}						
	<u> </u>	F	_	\mathcal{B}		I	N			P	0	R	7	2	B	Lest D/A
TRAINING	8 / 🔿	0	0	D										•		
¥		1	E	6		A	N	\mathcal{I}		0	8					latinera ix DA
		2	0	8	-		_									Set yero if D/A too low
띮		3	7	8		M	0	ν		A		$\overline{\mathcal{B}}$				Restore motor
MICROCOMPUTER		4	D	3		0	U	7		P	0	\mathcal{R}	τ	/	B	drive to D/A
S		5	0	<u>ر</u> ک	,		-	•								Carrie Sca Spirit
õ		6	C	8		R	Z									Exit id DA Paux
5		7	7	9		M	0	V		4		C				Display and
-		8	C	D		C	A	L	7		<u>.</u> 5	B	У	7	E	Display good
Ī		9	1 _ 1	5								,	·			temperature
		Α	0	2												Surger Care
MS		В	3	D		D	c	R		A						De areases Vor
SYSTEMS		С	3	2		S	C	RA			8	3	A	E		Decrease for next treat
		D	A	E								_				7
비		E	8	3											_	
		F	C	9		R	E	7								
힍	3	0				<u> </u>										
		1				7	工	ς	P	L	A	У	S		1	HL) = SPEED
ATE		2	Ţ			M	A	K	E	S		0	N	E	_	TRIAL
GR	 	3				<u> </u>	M	E	A		u		E		E	
NTEGRATED COMPUTER		4				Γ	w	I	T	H		7		A		KING VOLTMETER
-		5					D	I	S	P	上	À	У	S		VOLTAGE IF
1		6						G	R	E	A	T	E	R		THAN INPUT
		7				3	E	N	D	S		(\mathcal{B}			TO D/A BEFORE
		8					R		T	u	R	N				Figure 7-25b

7.6 REVIEW

In the Motor Control exercise of Chapter 7 we have used many of the interfacing techniques covered in this course:

- A/D Input by interval measurement (for instantaneous speed).
- A/D Input by frequency counting (for average speed).
- A/D Input by voltage conversion (for temperature).
- D/A Output by pulse width modulation.
- D/A Output by parallel conversion to voltage.
- Use of a counter to measure a time interval.
- Use of a counter to generate a timed-interrupt.
- Use of an optical coupler (the slot sensor) for input, generating an interrupt in response to a mechanical input.
- Use of an optical coupler to drive an external device (the motor) and exclude noise from the computer.

MOTOR CONTROL

We have also used all of the Input/Output techniques:

- Programmed Input Keyboard data and temperature measurement.
- Programmed Output Variable voltage control.
- DMA Output Displays
- Interrupt Driven Input Speed measurement.
- Interrupt Driven Output PWM control.

We have used the microcomputer to operate a closed loop control system and simultaneously monitor its performance by displaying average speed and temperature. We have provided for modifying the system operation by changing gains in the control equation and by switching between PWM control and variable voltage control. Finally, we time-shared a single device (the D/A - A/D converter) for input and output operations. The fact that all of these functions are accomplished within 512 bytes of program memory (8100 - 82FF) demonstrates the capability of a very small microcomputer in a control system.

MICROGOMPUTER INTERFACING WORKBOOK

APPENDIX A

REFERENCE FIGURES

APPENDIX A

This Appendix contains the following figures for reference:

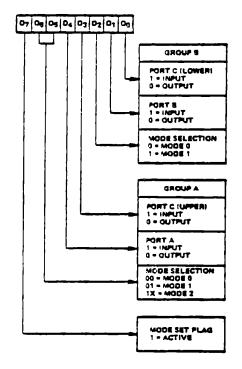
Figure A-1	Port Addresses
Figure A-2	Status and Command Bytes
Figure A-3	Standard Programming for 8255's
Figure A-4	Timer Conrol Byte Structure
Figure A-5	Timer Control Bytes
Figure A-6	8253 Timer Modes
Figure A-7	Count to Time Conversion

PORT ADDRESSES AND ASSIGNMENTS

ADDRESS	PORT NAME	FUNCTION	SPECIA	AL ASSIGNMENTS FOR OC AND 1C
00 01 02 03	PORT OA PORT OB PORT OC CNT O	MTS Keyboard Input Unessigned except 080 See column at right Control Port for MTS 8255 LED and Driver Outputs	0C7 0C6 0C5 0C4 0C3	Display Control (1 = On) Enable Command Keys (0 = On) Enable Keys 8-F (0 = On) Enable Keys 0-7 (0 = On) Zero Indicator
05 06 07	PORT 1B PORT 1C CNT 1	D/A Output or A/D Input See column at right Control Port for 8255 # 1	0C2 0C1 0C0	Carry Indicator Monitor Enable Cassette Modem Out
0C 0D 0E 0F	PORT 2A PORT 2B PORT 2C CNT 2	Unassigned Interrupt Status Input Interrupt Enable Output Control Port for 8255 # 2	080 1C7-4 1C3 1C2	Cassette Modem In Unassigned Interrupt (If Enabled by 2C6)
14 15 16 17	TIM 0 TIM 1 TIM 2 TIM CT	Timer 0 Timer 1 Timer 2 Cantral Port for 8253	1C1 1C0	Unassigned Motor Drive Buffer (1 = On) D/A Control (1 = Automatic A/D)

8255 PROGRAMMING CONTROL BYTES (WRITE TO 8255 CONTROL PORT)

					_	SE WI 8255	
CONTROL BYTE	PORT A	PORT B	PORT CO-CE	PORT C4-C7	0	1	2
80	Out	Out	Out	Out		D/A	
81	Out	Out	ln	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	ln	ln .		•	
8A	Out	ín	Out	ln		A/D	
88	Out	ln	ln .	În		A/D	
90	ln	Out	Out	Out	•	D/A	
91	In	Out	ln	Out	•	•	
92	ln .	În	Out	Out	•	A/D	•
93	In	in	In	Out	•	A/D	
98	in	Out	Out	ln		D/A	
99	ln	Out	ln	ln			
9A	in	ln	Out	ln_		A/D	
9B	In	In	ln	ln		A/D	



[.] Generally only these central bytes should be used for normal operation.

Forbidden configurations

Made Definition Formet

A-3

STATUS AND COMMAND BYTES

		ST	ATUS E		OBTAII		Y IN PO	ORT 29	3	WRITT OUT	
INTERRUPT SOURCE				BIN	HEX	DISABLE	ENABLE				
Timer 0	0	Х	X	Х	Х	Х	Х	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	QS
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	08
Port 1C3				(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

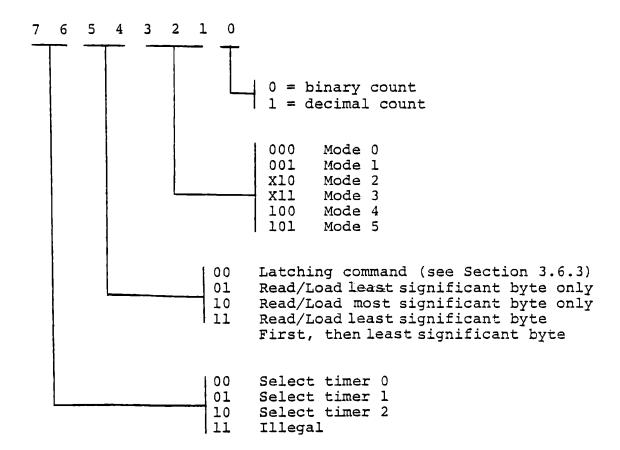
Figure A-2

Program	8255's	- 1B out
3E	MVI	A,80
80		
	OTTE	CAYOU I
D3	OUT	CNTl
07		
3E	MVI	A,92
92		,
		a
D3	OUT	CNT2
OF		

Program	8255's	- lB in
3E 82	MVI	A,82
D3 07	OUT	CNTl
3E 92	MVI	A,92
D3 OF	OUT	CNT2

Standard Programming for 8255's
Figure A-3

Timer Control Byte Structure



Timer Control Byte Structure
Figure A-4

APPENDIX A

This page intentionally left blank.

A-10

Timer 0

М	O	a	e

	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	73	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	9 A
Read/Load MSB	A0	A2	A4	A6	A8	AA
Read/Load Both (LSB first)	В0	В2	34	B6	В8	ВА

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 A-5

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

8253 Timer Modes Figure A-6

Binary Count (Hex representation)	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
OFA0	4000	1/512
07D0	2000	1/1024

Count to Time Conversion
Figure A-7

MICROCOMPUTER INTERFACE WORKBOOK

APPENDIX B

CASSETTE INTERFACE INSTRUCTIONS AND PROGRAM CASSETTE LIBRARY

$\underline{I} \ \underline{M} \ \underline{P} \ \underline{O} \ \underline{R} \ \underline{T} \ \underline{A} \ \underline{N} \ \underline{T}$

THE TAPE PROVIDED IS A MASTER TAPE. IT SHOULD BE COPIED TO A WORKING TAPE WHICH SHOULD BE USED FOR LOADING OF PROGRAMS. KEEP THE MASTER TAPE IN A SAFE PLACE SO THAT IF YOUR WORKING TAPE BECOMES DAMAGED OR WORN OUT, YOU CAN MAKE ANOTHER COPY. THE WORKING TAPE SHOULD BE A HIGH-QUALITY C-60 OR C-30 CASSETTE.

B.1 CASSETTE INTERFACE INSTRUCTIONS

The MTS monitor program contains the software routines for loading and storing binary data using a tape cassette unit. The input routine, SERIN, resides at location O3AE and the output routine, SEROT, resides at 0371.

The MTS includes a tape cassette modem for recording programs on tape and loading programs from tape. Full instructions are included in Course 525A, and are briefly summarized here. Your ITS circuit board may include a duplicate modem, provided for use with an earlier version of the MTS, but this is not connected to the microcomputer input and output ports.

B.1.1 Instructions for Reading from Tape

- (1) Find the program (or data) on the tape and listen for the solid tone which PRECEDES the program.
- (2) Connect the cable to EAR on the tape unit and the connector labelled with an inward pointing arrow on the MTS.
- (3) Set the AUTO/STEP toggle switch to AUTO.
- (4) Enter the beginning load address as the memory address e.g., if the beginning address is 8100, press ADDR, then 8100, then MEM.

- (5) Enter the start address of SERIN into the Program Counter: ADDR 03AE.
- (6) With about 80% of full volume, press the PLAY button on the tape unit. Wait until the DATA LED emits a constant glow.

(7) Press RUN

The MTS displays will blank out and the DATA LED will flicker as data are received.

- (8) A display of O3CF in the left MTS display indicates a successful load.
- (9) A display of Err indicates an unsuccessful load, so repeat the procedure.
- (10) Check the LRC by REG A. Register A should contain 00.

B.1.2 Instructions for Writing to Tape

- (1) Advance the tape to the point where you want the program (data) stored. If using the beginning of the tape, make sure you have advanced the tape beyond the leader.
- (2) Connect the cable to MIC or AUX on the tape unit and to the connector labelled with an outward arrow on the MTS.
- (3) Set the AUTO/STEP toggle switch to AUTO.

- (4) Enter the starting address of the program to be recorded as the memory address: ADDR 8100 MEM.
- (5) Enter the stopping address (the next address after the end of the program) as a breakpoint: ADDR 82A0 BRK.
- (6) Enter the start address of SEROT into the program counter: ADDR 0371.
- (7) Press the RECORD and PLAY buttons on the tape unit simultaneously. Wait 5-10 seconds.
- (8) Press RUN.

The MTS displays will blank out.

(9) A display of 0382 indicates a successful transfer.

CASSETTE LIBRARY PROGRAM DIRECTORY

Side 1 of Tape Cassette

- 1. Display Keyboard Input Figure 2-3
- 2. Time of Day Figure 3-17
- 3. Pulse Width Modulation Figure 4-7
- 4. Recorded Music Player Figures 4-5 Figures 4-15, 4-16, 4-17, and 4-19
- 5. Function Generators Figures 4-44, 4-45, and 4-46
- 6. A/D Input with FILTR Figure 5-30
- 7. Thermometer with Data Log Figures 5-38, 5-40, and 5-45

Side 2 of Tape Cassette

- 8. Thermostat with Deadband Figure 6-6
- 9. PWM Voltage Control Figures 6-10, 6-18, 6-28, 6-34, and 6-37
- 10. Motor Speed Control Figure 7-21
- 11. PONG
 Appendix B, Figure B-4
- 12. RS232C Message Handler Appendix C

Program Cassette Directory Figure B-1

B.2 PROGRAM CASSETTE LIBRARY

Your program cassette library is a cassette tape that contains solutions for the longer 535 course exercises. The cassette is included only to free the student from the time-consuming key-in procedure. It is recommended that you use the cassette only after you have attempted your own solutions to the exercises. The tape also contains a game program and a communications program.

The listings of the programs are available in the appropriate course section. These are identified in the following pages, and a list of library programs is shown in Figure B-1.

B.2.1 Display Keyboard Input: Figure 2-3

This program provides a trivial test to determine whether the ITS and MTS are properly connected. It also gives the user experience in reading a tape.

Play the tape until a steady tone is heard.

When you hear the tone, promptly stop the tape and do the following:

Press RESET

Press ADDR 8200 MEM

Press ADDR 03AE

Connect the tape player earphone output to the connector labelled with an inward pointing arrow at the top right of the MTS circuit board.

Start the tape and observe a steady light in the LED labelled DATA, near the input connector. Immediately press RUN.

When the computer starts to receive data, the AUDIO and DATA lights will flicker. This will continue while the program is being loaded. At the end of the program these lights will become steady, and then the display will show O3CF C5.

Stop the tape.

Check that the program has been read correctly by pressing REG A.

The display will show 03CF A-00. This implies that no errors have been detected.

To run the program, press RESET, RUN. The LED's on the ITS circuit board will all be lighted. Pressing any key on the MTS will extinguish one of the LED's while the key is held down.

B.2.2 Time of Day Program - Figure 3-17

This program uses the system's crystal clock and interval timer 0 to keep the time of day.

To load the program:

Set the red toggle switch to AUTO

Press RESET

Press ADDR 8200 MEM

Press ADDR 03AE

When the steady tone is heard, stop the tape, connect the input, and start playing the tape. Observe the DATA light and press RUN.

When the program has been loaded, check for errors by pressing REG A. The display should show O3CF A-OO.

Before running the program, press RESET. Enter the time of day as follows:

Press REG H and enter hours.

Press NEXT and enter minutes into register L.

Press RUN. The display will show hours, minutes, and seconds.

The following programs are loaded by the same procedure as described previously.

B.2.3 Pulse Width Modulation - Figure 4-7

This program is described in Section 4.2 of Course 536. The recorded program is identical to that given in Figure 4-7. To use the program, make connections as shown in Figure 4-2. Then press RUN and observe the voltmeter output, which should show about 2.5 volts. Enter any two digit decimal number followed by NEXT, and observe that the output is proportional to the decimal value entered. The program starts at address 8200.

B.2.4 Recorded Music Player - Figures 4-15, 4-16, 4-17, and 4-19

This program plays a tune from memory. It is described in Section 4.3.3 The program is given in Figure 4-15, with a patch for a visual display from Figure 4-19. The table of timer intervals for the chromatic scale and two American folk tunes are given in Figures 4-16 and 4-17.

To use the program, press RESET, RUN. To play the first tune, or to repeat a tune, press NEXT. To start a different tune, enter its address and press NEXT. Two tunes are included in the tape, at addresses 8300 and 8330.

For students who want to develop other music programs, it is suggested that the interrupt service and timer table from this program be used, and other program segments be developed in the 8000 to 81FF section of memory.

B.2.5 Function Generator - Figures 4-44, 4-45, and 4-46

This program generates triangular and exponential waveforms, output ting the signal through the Digital to Analog converter. Its development is described in Section 4.7.

The taped program is shown in Figures 4-44, 4-45, and 4-46. This is the final program including both waveforms.

Observe the output with a voltmeter at ANALOG OUT. To operate the program, press RESET, RUN. The exponential waveform will appear. Press REG to obtain a triangular waveform, or MEM to obtain the exponential. Variable data may be entered as described in Section 4.7.2.8 and Section 4.7.3.7.

Load the program at address 8100.

B.2.6 Analog to Digital Input with Digital Noise Filter - Figure 5-30

This program measures an input voltage and uses a digital filter to reduce noise present at the input. It is described in Sections 5.4 and 5.5

Connect a voltage source (not exceeding 2.5 volts) to ANALOG IN. Be careful that the input is positive with respect to ground -- negative inputs will damage the A/D converter. The ANALOG IN pot should be set fully to the left. The program will display the filtered voltage at the left and the input voltage next to it. Both displays are in hexadecimal. Adjust the ANALOG IN pot if necessary. Load the program at 8200.

B.2.7 Thermometer with Data Log - Figures 5-38, 5-40, and 5-45

This program measures the input voltage from a thermistor, applies the noise filter, converts the voltage to decimal degrees Celsius, and displays the temperature. It also records the filtered voltage at preset intervals in a data log, and permits subsequent review of the temperature history.

The voltage to temperature conversion is described in Section 5.6. The data log and review are described in Section 5.6.7.

Operate the program by pressing RESET, RUN. Then enter a timing interval in hexadecimal seconds, followed by RUN. This will display and record the temperature.

To review the temperatures one at a time, press RESET, RUN, NEXT. Each time you press NEXT, the next temperature will be displayed. To play the temperatures back continuously, press RESET and RUN. Then enter a timing interval and press STEP.

The program is loaded at address 8100.

B.2.8 Thermostat with Deadband - Figure 6-6

This program controls a heater to maintain the thermistor temperature within set limits. It is described in Section 6.1.2. Make the connections shown in Figure 6-1.

To run the program, press RESET, RUN. Enter a temperature limit followed by NEXT. This limit is expressed as decimal degrees and tenths, so to make the limit 25.0 degrees enter 250 NEXT.

The heater will be turned on if the measured temperature is below the limit. It will be turned off when the temperature rises two degrees above the limit.

Load the program at address 8200.

B.2.9 Closed Loop Voltage Control with Proportional Plus Integral Feedback - Figures 6-10, 6-18, 6-28, 6-34, and 6-37

This program controls an output voltage by pulse width modulation, and adjusts the output with proportional plus integral feedback.

Connections for testing the program are shown in Figure 6-10.

Program development and operation are described in Sections 6.2 and 6.3. The recorded program is the final program developed in Chapter 6 and includes segments from various figures listed above. The complete program listing is given in this appendix.

All of the experiments described in Sections 6.2.2.8, 6.2.5, and 6.3.4 can be performed with this program. Some results will be slightly different than shown in the text, because of the revised subroutines.

The program is to be loaded at address 8100.

B.2.10 Motor Speed Control - Figure 7-21

This program controls the speed of a motor with proportional plus integral closed loop control, sensing motor speed with a rotating disc and an optional coupler.

Electrical connections are shown in Figure 7-5.

All of Chapter 7 is devoted to development of this program. Operating instructions are given in Section 7.3.6. The program is shown in Figure 7-21.

Load the program at address 8100.

B.2.11 PONG

This is a game for two players. Operating instructions and program listing are shown in this appendix. Load the program at address 8200.

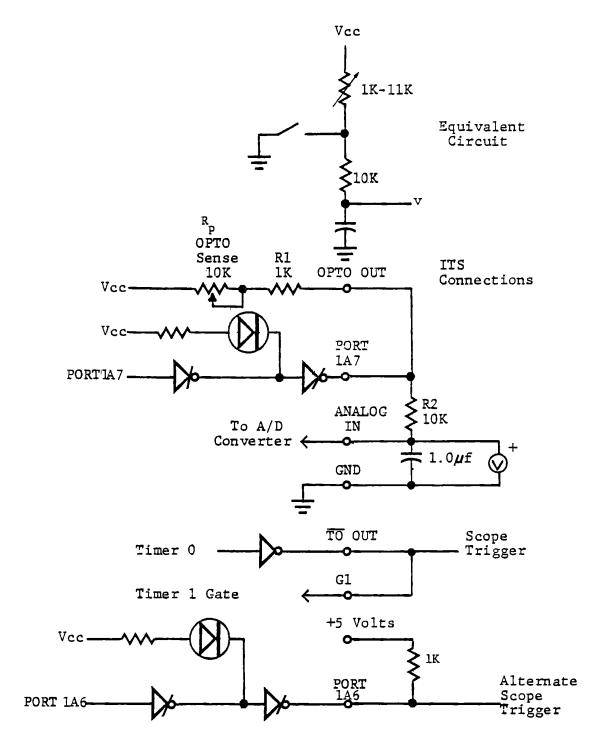
B.2.12 RS232C Message Handler - Appendix C

This program and required hardware are described in Appendix C. Load the program at address 8200.

Program Segments for Closed Loop Voltage Control

6-10	Connections
6-18a	Initialization
6-18b	Interrupt Service
6-18c	VOLTM
6-18d	FILTR
6-18e	FILTR and SHFTN
6-18f	Set Initial Values
6-28a	Main Loop
6 - 28b	KYTIM
6-28c	KYTIM
6-28d	KYTIM
6-34	LDT1, Version 2
6-28f	CLOSL, Version 2
6-37a	INTEG, Version 3
6-28h	PROPG
6-37b	ADTOV

Figure B-2



Connections for PWM Experiment (Figure 6-10)

Figure B-2a

	A D D	R	co	DE					 		_	_			
	8 20	0	3	E		M	V	<u>I</u>	A		8	2			Program 8255#1
		1	8	2						_					fort 1B in for A/D
		2	\mathcal{D}	3		0	U	\mathcal{T}	<u>C</u>	N	7	/			/
ابر		3	0	7										,	
SHEET		4	3	E		M	V	工	A		2	a			Program 8255#2
		5	9	2											
CODING		6	B	3		0	U	\mathcal{T}	C	N	T	2			
200		7	0	F											
		8	3	E		M	V	1	A		3	4			Trogram Timer O
		9	3	4						,					Both lustes
		A	$\overline{\Diamond}$	3		0	U	7	7	\mathcal{I}	M	C	7		Mode 2
		В	/	7											Binary
		С	3	E		X	1	Н	A		7	A			Program Times 1
STEM		D	7	A						,					Both lustes
YST		E	D	3		0	$\overline{\mathcal{U}}$	\mathcal{T}	7	工	M	C	T		Mode 5
G SY	-	F	1	7											Binary
RAINING	821	0	3	E		M	V	I	A	-	9	4		,	Program Times 2
¥.		1	9	4											Low byte
-		2	D	3		0	U	τ	T	I	M	C	7		Mode 2
TER		3	/	7						_					Binary
P		4	3	E		Μ	ν	I	A		0	8			Load Timer 2
MICROCOMPUT		5	0	8	-					,					for ÷8
RÕ		6	7	3		0	U	7	7	I	M	2			
N N		7	/	6											
		8	2	/		L	X	I	4		8	ദ	A	0	Initialize memory
		9	A	0		i				,				-	for FILTR
		A	8	3											7
EMS		В	3	6		M	V	エ	M	-	0	4			m = 4
'STI		С	0	4						-				: -	
S		D	A			X	R	A	A						Clear 2"E:
ŢĒ		E	2	3		I	N		H						Clear 2 "E: address 83A1
JPU.		F	7	7		M	0	1	M		A				
Š	3 2 2	0	2	3		I	_	X	H						address 83A2
<u> </u>		1	7	7		M	_	V	M		A				
ATE		2	3	C		$\overline{\mathcal{T}}$	N	R	A						Set Port 1CO
INTEGRATED COMPUTER SYSTEMS		3	2	3		0		T	P	0	R	T	1	C	In) outrontia)
NTE		4	0	6		Ť	-							_	for automatic A/D input Jump past
=		5	C	3		7	M	P	8	2	В	0			Jump , part
	-	6	B	0			Ė	İ	=			_			interrept service
		7	8	2				Г						Γ-	interrupt services
		8	Ť	Ť	ļ — —									<u> </u>	Figure 6-18a
				<u> </u>					 		ــــــ				D 15

	A D D R	c	DDE	PWM	ı V	DLT	AG!	E C	eno:	ro	L				
	8 0				I	N	7	E	R	R	U	P	7		SERVICE
	1				1		<u> </u>		-		Ū				
	2														
L	3										<u> </u>			Г	
SHEET	4														
	5		1												
N.	6														
CODING	7														
U	8228	F	5		P	V	S	H		P	5	W			RST5-Times O
	9	3	E		M	•	I		A	,	8	0			
	Α	8	0							,					Set Port 1A7 high to start charging all other lits low
	В	12			0	U	\mathcal{T}		P	0	R	7	1	A	all other lists law
	С	0	· .	1											
Ē	D	C	3		2	M	P		8	8	4	0			Jump past RST6
SYSTEM	E	4	0												
	F	8	2												
TRAINING	8 230	F	5		P	V	S	H		P	S	W			RST6 - Times/
ξ	1	3	E		M	V	エ		A	,	0	0			Set Port 1A7 low
	2	0	0							7					to start discharge
MICROCOMPUTER	3	D	3		0	U	$\overline{\mathcal{T}}$		P	0	R	τ	/	A	(all other lite also)
P	4	0	4												Note identical times
S	5	نح	E		M	V	I		A		0	3			as in RST5
Ã	6	0	3												
ΜC	7	Z	3		0	U	T		C	N	7	Q			Reenable and clear
	8	0	1												Lines 1 interrupt
	9	F	1		P	0	P		P	S	W				Exit
۲۵	А	F	B		E	I									
SYSTEMS	В	C	9		R	E	\mathcal{T}								
/ST	С	0	0		N	0	P								
	D	0	0		N	0	P								
TEI	E	0	0		N	0	P								
ΔPO	F	0	0		N	0	P								
S	3 24 0	3	E		M	V	I		A		0	1			Reenable and clear
ED	1	0	1							_					Liner Ointerrupt
INTEGRATED COMPUTER	2	D	3		0	U	7		C	\mathcal{V}	\dashv	2			
EGF	3	0	F												
N	4	F	1/		P	0	P		P	\mathcal{C}	ω				Efit
	5	F			Ē	I									,
	6	_ C	9		R	E	\mathcal{T}								
	7		<u> </u>												
	. 8														Figure 6-18b

	A D D R	СО	DE_	PWM	- 1	SU	BR	TUC	'INI	e v	OL:	ΓM	WI'	TH	LOG B - 17
	8 25 0	E	3		P	U	5	14		H					Save (HL)
	1	2	1	-	上	X	I		14	-	8	0	0	0	addressstrage
	2	0	0	*						1					location for
	3	8	0	*				_					1	†	data log address
EET	4	3	4		エ	N	R		M	-			1		Increment address
S	5	C	2		┿	N	ァ		8	a	5	9	 	T	but not beyon & FF
CODING	6	5	9		~	10	٦	┢─	0			 	 	\vdash	and not sugon at -
	7	5	2		-		-		-	\vdash	┢	-	+-	-	· · · · · · · · · · · · · · · · · · ·
5	8	3	7		D	6	R		M	-	-	\vdash	 	-	Restore FF
		1	E		岩	0		-	7	-	M	-	-	╁	
	825°9	6	6		M	V	I	 	M	-	E	F	 	-	(HL) - data log addres
	В	3	_		1/*1	V	+		10-1	1	-	_	 	-	stre FF in case
i		15	F		+				P	_	P	-	-	0	A D not ready
Σ	C	2	\mathcal{B}		1-	N			<u> </u>	0	R	-	α	B	Kead interrupt
SYSTEM	D	0	D		1	_	_		_	~	-	-	-		status byte
2	E	15	6		A	N	I		0	8	<u> </u>	<u> </u>	-		Mark for
	F	0	8			<u> </u>	<u> </u>		_	<u> </u>	<u> </u>		<u> </u>	<u> </u>	A/D comparato
INING	8 260	C	A		2	Z	ļ		8	2	6	\mathcal{D}	 	_	Jump if
TRA	1	6	D		 	-	_		_			<u> </u>	<u> </u>	_	A/D not ready
ER I	2	8	2			ļ.,			<u> </u>				ļ	Ļ	
7	3	2	\mathcal{B}		I	N			P	0	R	T	1	$ \mathcal{B} $	Read A/Dwoltage
MICROCOMPUT	4	0	5		_		_		L						
٥ ا	5	3	2	_	S	\mathcal{T}	A		8	3	A	A			Store voltage
2	6	A	A		<u> </u>										in fixed location
∑ ∑	7	8	3												and log in
į	8	7	7		M	0	V		M		A				variable beating
	9	3	$\boldsymbol{arepsilon}$		M	V	I		A		0	6			Reset A/D Counter
	A	0	6							,					Leave interrupt
Ĭ	В	(3		0	U	7		C	7	7	7			disabled
S	С	0	F									_			
S.~	8260	7	E		M	0	7		A		M				(A) - voltage
TEF	E	E	/		P	0	4		+	,					Restore (HL)
3	F	C	9		R	_	7								
Š S	3 0														
<u>ی</u>	1			*	7	8	0	0	0	7		6	0	N	TAINS
A	2				<u> </u>	-	D	A	<u> </u>	A		1	0	G	
בו פאל	3	 			न्न	E	T	v	dρ	7		7	A		= VOLTAGE
NIEGRATED COMPUTER SYSTEMS	4			<u> </u>	广	_	己	E			\vdash	1		Á	
=	5	1	\vdash		ュ	F	۳	4	1	99	_	N	0	F	READY
	6	1	\vdash		-	-	7	A		11	E	F	5	'-	ZERO FLAGSET
	7	1-	-	 -	\vdash		-	П	1			厂	-	\vdash	EERU FLITG SEI
İ		+-	\vdash		 	\vdash	-				_	 	-	<u> </u>	Figure 6-18c
	88	<u> </u>		<u> </u>	<u></u>	L				L		Ļ	L	L	B-17

	A D D R	co	DE	SUB	ROU	JTI	ΝE	FI	LT	₹					
	8 270	2	5		$\overline{\mathcal{P}}$	V	5	H		D					Lave (DE)
	1	C	5		P	U	S	#		B					Save (BC)
	2	4	6		M	0	V		B		M				(B) - n
-	3	4	8		M	0	V		C		β				(C) = N
SHEET	4	2	3		I	N	Х		H	,					address 2 Ei-1
	5	E	5		ρ	V	S	1-1		H					Save address
CODING	6	3	E		M	D	V		E	1.	M				()
00	7	2	3		工	\mathcal{N}	X		H						(DE) = 2" Ei-1
_	8	5	6		M	0	V		D		M				 }
	9	6	B		M	0	V		<u>L</u>	<u>,</u>	E		<u>L</u>	<u> </u>	(HL) - 2 m Ein
	Α	6	2		M	0	V		#		D				
	827B	2	9		D	A	\mathcal{D}		H					<u> </u>	<u> </u>
_	С	0	D		D	<u>C</u>	R		C						
SYSTEM	D	C	2		J	N	Z		8	2	7	\mathcal{B}			7(HL) = 22 Ei-1
ΥS	E	7	\mathcal{B}											L	
	F	8	2												V
TRAINING	8 28 0	4	F		Μ	0	1		C		A				(C) = Vi
RAI	1	7	D		M	0	V	•	A		L				<u> </u>
	2	9	3		S	U	β		E						(DE) (HL) - (DE) = 2 - (2 - 1) E:-1
JTE	3	5	F		M	0	V		E		A				} = 2 ~ (2 ~ 1) E :- 1
MICROCOMPUTER	4	7	C		M	0	V		A	, ,	\mathcal{H}				
CO	5	9	A		S	\mathcal{B}	\mathcal{B}		D					<u> </u>	
SRO	6	7	7		Μ	0	V		D	,	A)
MIC	7	6	9		M	_	1		L	<u>,</u>	\mathcal{C})
	8	2	6		M	V	\mathcal{I}		Н	<u></u>	0	0			(HL) = Vi
	9	0	_												J
S	Α	0	D		C	A	<u>L</u>	<u>L</u>		S	H	F	I	N	DE) = (2 n) Ei-
SYSTEMS	В	A	0								_				(DE) = (2 1) Ein
YST	С	8	2							_					
	D	E	B		X	C	H	G	_)
JTE	E	/	9		\mathcal{D}	A	D		Δ						(DE) = Vi+(2-1)Ei-
INTEGRATED COMPUTER	F	E	B		X	C	H	G							= am Fi
S	3 0				ے	0	N	7	Z	N	U	E	2		AT 8290
ED.	1				_						_	_			
RAT	2				E	N	\mathcal{I}	E	2			A)	=	Vi (new value)
EG	3						H	7)	7		_		=	n (filterconstant)
Z	4				4		\mathcal{H}	4	Ų,	+	4	7		2	2nt i-1
	5		\square	·	4		#	4	۲	+	2	<u>)</u>		5	
	6				R,	Ę	7	U	R	M	_				
	7		\square		4		H	<u>_</u>)	+	3)	=		$A)=(H)=E_{\mu}$
,	8													· _	(L) = Vx

B-18

Figure 6-18d

	A D D	R_	ÇO	DE	L T T		•			mue			<u> </u>		. 114	B ~ 19
	8 2 9	0	E	3		X	7	#	L							(HL) - address 2 "E:
		1	7	3		N	0	V		Μ		E)
		2	2	3		工	N	X		1	,					Stone 2 nE
		3	7	2		M	0	V		M		C C				1
EET		4	2	3		I	\mathcal{N}	X		Н	-					address E:
SH		5	7	B		D	/	X		٦						
NG	-	6	C	D		2	A	,	7		5	H	F	7	1/	To round up only if
CODING SHEET	-	7	A	2		ب ا	/1	J	-		٠	//	_	-	//	factional part > 1/2
ၓ		8	8	2								-		-	\vdash	· · · · · · · · · · · · · · · · · · ·
		9	7	7		М	~	V	\dashv	М	_	A		-		1+ 5.
		A		/		P	00	$\frac{v}{\rho}$		H	1	//				Store Ei
			E	1		<u> </u>		7		_						(L) ← Vi
		В	6	/		M	0	7		14	1	A		_		(H)-E;
5		C	0	/		P	0	P		BA					_	Restore B, C, D, E
SYSTEM		D	7			P	0	1		D					<u></u>	
SYS		E	C	9		R	E	7						-		Exit
		F	0	0		N	0	Ρ						_		
N	8 2 A	0		8		M	0	V		2		\mathcal{B}				SHFTN (c) N
MICROCOMPUTER TRAINING		1	A	F		X	R	A		A						Loop - Clear carry
RT		2	7	A		M	0	V		A	,_	\mathcal{D}				1
JTE		3	\angle	F		R	A	R								Shift (DE) right
MPL		4	5	7		M	0	V		\mathcal{D}	,	A				Shift (DE) right to divide by 2
COI		5	7	B		M	0	V		A		E				
RO		6		7		R	A	R				-			.	
MIC		7	5	F		Μ	0	V		E		A)
		8	0	D		\mathcal{D}	C	R		C	,					Loop of times
		9	C	2		4	N	Z		8	8	A	/			Toop on times to divide by 2"
4.0		A	A	/		i										
EMS		В	8	2												
STI		С	D	0		R	N	\mathcal{C}								Exit id LSB=0
SY		D	1	3		工	N	X		D		1				Flow round up
TEF		E	7	B		M	0	ν		A		E				(A) = less
INTEGRATED COMPUTER SYSTEMS		F	C	9		R	Ē	7			-					Exit if LSB=0 Else round up (A) = less significant lyte
Ŏ	3	0				Ť										
٥		1	\vdash													
ATE		2														
GR,		3	\vdash	\vdash												
ŢĒ		4	\vdash					-						-		
=		5	一			┢┈	\vdash	-						-		
		6	-		<u> </u>		+-		\vdash	-	-		-			
	-	7	-	-		\vdash	\vdash	-		<u> </u>	-	\vdash		-		
		8	-	 	 	\vdash	-	-		-	-	\vdash	\vdash	-		Figure 6-18e
		0		<u> </u>	┖	Щ.	L	L_	<u> </u>	<u> </u>	L	L		Ц	Ц	rigule 6-18e

	A D D	R	со	DE.	PWM	VC	LT	AGI	Ξ –	SE	T_	IN	TI.	AL	VA	LUES
	8 2B	0	1	/		7	X	I		D	,	0	4	0	0	Let total seriod
		1	0	0			_									Let total period to 500 u sec
		2	0	4												
_		3	C	D		C	A	L	2		R	U	N			Lun ken Synction of
SHEET		4	3	8	-		-									KYTIM John Gener O
		5	8	1												La) total assint
ING		6	7	7		L	X	I		D		0	0	C	8	Let sultage
CODING		7	C	8							/		7			for total period Set voltage to 2.00 volta
O		8	0	0				_								
		9	C	D		C	A	L	L		\sim	E	X	T		KYTIM Calculates Width, loads Jimes!
		A	2	0									-			KYTIM Calculate
		В	8	1												width basilines!
)	С	E	F		R	5	7		5			_			
EM		D	F	1		R	<u>S</u>	\mathcal{T}		6						
TRAINING SYSTEM		E	C	3		7	M	ρ			8	2	\mathcal{D}	0		
G.S.		F	D	0									_	_		
Ž	8 2 C	0	8	2				-								
IAI		1											-			
		2										-				
MICROCOMPUTER		3														
/PU		4														
SON		5														
RO		6												-		
MIC		7														
		8														
		9														
(0		Α														
SYSTEMS		В														
/ST		С														
R S		D														
INTEGRATED COMPUTER		E														
MPL		F				_		Ш				`				
CO	3	0	L			<u> </u>										
ED.		1		_		ļ			_							
PAT		2														
EG!	- -	3				<u> </u>										
INT		4	_		 											
		5														
		6				<u> </u>										
1		7				_										
		8				<u> </u>										Figure 6-18f

,	A D D	R	co	DE	FWM		111	AGE		ONI		_				301 R = 21
	8 2D	0	\supset	\mathcal{B}		I	N			p	0	R	\mathcal{T}	0	A	Test keyboard
		1	0	0				_								(FF if no key)
	·	2	3	C		エ	N	R		A				_		
		3	C	4		C	N	Z			K	Y	T	I	M	Call for keyboard
SHEET		4,	0	0										_		entry and process
5		5	8	/												
CODING		6	2	D		C	A	L	4		V	0	<u></u>	T	M	Measurevoltage
힝		7	ک	0												(A) - A/D voltage
		8	8	2									`			(FF if not ready)
		9	F	5		P	\mathcal{U}	S	H		P	2	W			Savelingut
		A	C	D		C	A	L	<u>L</u>		C	L		S	1	Calculate error
		В	8	0												adjust pulse width
		С	8	1												
EN		D	C	V		C	A	L	L		D	ω	0	R	A	Displayerror
SYSTEM		E	D	1												and voltage request
		F	0	1												
RAINING	8 2E	0	F	/		ρ	0	p		7	S	W				(A) - input voltage
₹[1	2	1		L	X	I		H		W 8	3	A	0	(A) - input voltage Memory address
\vdash		2	A	0												Now FILTR
TER		3	8	3												
₹[4	2	D	_	C	A	L	L		F	I	L	T	R	
MICROCOMPU		5	7	0	_											(L) - raw voltage
2		6	8	2												(H) - filter voltage
[≥		7	/	/			Χ	\mathcal{I}		Θ		8	က	F	7	
		8	H	F												
		9	8	3												
<u>"</u> [A	C	D		\mathcal{C}	A	L	7		\mathcal{D}	W	\mathcal{D}	2		
E		В	D	4												
S		С	0	2												
R S		D	C	3		2	Μ	ρ			8	2	\mathcal{D}	0		
		E	D	0												
INTEGRATED COMPUTER SYSTEMS		F	8	3												
9	8	0														
		1				I	Q	E	N	T	I	\cup	A	L		TO
M		2				F	I	EG	U	R	E		6	-	2	ΤΟ 3ω
		3														
틸		4														
-		5														
Ī		6														
1		7														
j		8					.									Figure 6-28a
																B-21

	Α.	D D R	со	DE	PWM		50	BK	001	.TM1	۲ K	YT.	ΓM			
	8	100	0	0		N	0	P								
		1	C	D		C	A	L	1		E	N	7	W	D	(A) 4- Command
		2	4	6												(111) + 4-1.
⊢ -		3	0	3												ov voltage
SHEET		4	E	B		X	C	H	G							(DE) ← data
S S		5	2	/		L	X	I		14		8	/	0	8	address dispatch
N		6	0	8	-			-			7					table -10H
CODING		7	8	/												
		8	8	5		A	\mathcal{D}	\mathcal{D}		L						add command
		9	6	F		M	0	V		1		A				
		Α	6	E		M	0	<u>V</u>		L		Μ				(57) - dispatch address
		В	E	5		P	V)	S	#		Ĥ					,
_		С	F	3		D	I									
SYSTEM		D	D	$ \mathcal{B} $		工	N			P	0	\mathcal{R}	T	1	A	
YS		E	0	4												Set Port 1A6 high
		F	F	6		0	R	工		4	0	_				I for scope trigger
TRAINING	8	// 0	4	0												without changing
RAI		1	F	\mathcal{B}		E	\mathcal{I}					_				Port 1A7
		2	0	3		0	U	T		P	0	R		1	A	
JTE		3	0	4)
MICROCOMPUTER		4	A	F		X	R	A.		A				_		Clear Hand CY
000		5	0	0		N	0	p								
CRC		6	0	0		N	0	9				_				
Ž		7	C			R	Ε	<i>T</i>				_				To dispatch address
		8	4	0		$\overline{\mathbb{W}}$	田田	M		_				_		
	_	9	5	0		Ř		6				_				
JS		A	-	7		A	\mathcal{D}	Ī	2		_	_				Trigger scope Store desired voltage Set total period Store voltage, set width
τEΝ		В	2	/) 	T	E	P							Store desired voltage
3YS		<u> </u>		8		$ \mathcal{R} $	U	N	7							Set total period
ER S		D	2			N	E	쏫	\mathcal{T}							Store voltage set width
UTE		E		5		B	R	K			_			_		
MP		F .	4	7		Ċ	<u>L</u>	R			_	_				<u> </u>
INTEGRATED COMPUTER SYSTEMS	8	0	-	$\left \cdot \cdot \right $		_					\dashv	\dashv				
TEC		1	\vdash						٠		_	-				
3RA		2	\vdash	$\vdash \dashv$						-	-				_	
TEG	ļ	3	-	\vdash								-				
Z		4 5	 								-	\dashv				
		6	-	\vdash								\dashv				
		7					\vdash				-	\dashv				
		8	-	\vdash				-	-			-				Figure 6-28b
		0						لــــــا								

	A D D	R	ÇO	DE	KYT	TM		NE)	CT,	51	EP	, !	KUN			B - 23
	8 /2	0	3	7		g	7	\mathcal{C}								NEXT (mark with CY)
	812	1	2	/		7	X	I		14		8	0	0	0	STEP/ CY clear
		2	0	0												address data log
L		3	8	0											,	and load with its
SHEET		4	7	5		M	0	V		M		L				sun address
		5	E	В		X	$\overline{\mathcal{C}}$	1+	G							
NG		6	2	2		S	14		Ā		8	3	A	6		
CODING		7	A	6			•					<u> </u>	,			
ပ		8	8	3												
		9	9	2		\overline{R}	~	C								Exit if STEP
		Α	7	1		7	X	I		$\overline{\mathcal{D}}$		F	F	F	E	12 2 1 til to 12 14
		В	Ė	E			_	_		U	}	·		-	<u></u>	(0,18 volts) from
		c	<u></u>	F			_			-				-	-	
Σ		D	/	9		$\overline{\mathcal{D}}$	٦	$\overline{\mathcal{D}}$		D				_		desired voltage
SYSTEM			1	9		$\frac{\mathcal{D}}{\mathcal{D}}$	AA	D								(HL) - v - Vg
		 F	2					9	_	H		_		_		(HL) = pulse with
MICROCOMPUTER TRAINING		_		0		N S	0	7	7		0	0	Λ	11	<u> </u>	
Z	8 /3	0	2	2		<u> </u>	1+	_	D		8	3	A	4		Store colculated
ΓRΑ		1	8	3								-			-	as integral
Я		2				_	_	,	,		,	7		_		as integral
UTE		3	0	D		C	A	4	<u>L</u>		L	\mathcal{D}	7	_		Lond Times!
ΜP		4	6	0								_			<u> </u>	with calculated
သင	·	5	8	/			_		_						<u> </u>	pulse width
CRC		6	C	9	:	R	E	7				<u>.</u> .			<u> </u>	
Ξ		7		0		N	0	P						 		
	813	8	7	B		M	0	V		A	,	E		L		RUN
		9	2	3		0	U	7		7	I	M	0			Load Timer O
S		Α	1	4								_			L	for total period
Ë		В	2	A		M	0	1		A	,	\mathcal{D}		L		
YST		С	2	3		0	U	T	_	I	I	M	0	_		
R S		D .	1	4				L								
JTE		E	C	9		R	<u>E</u>							<u> </u>		
INTEGRATED COMPUTER SYSTEMS		F	0	0		N	0	P						<u>L</u>		•
õ	8	0														
ED		1			<u> </u>	I	D	E	N	T	工	C	A	<u>L</u>		TO
١		2	L^{-}			L	I	G		R	Ш		6	_	/	TO Bi
EGR		3														
Ĭ		4														
_		5														
		6														
		7	Ī													
		8									_					Figure 6-28c
			<u> </u>											_		B-23

KYTIM - MEM, BRK, CLR, REG CODE MEM Store gains for CLOSL E B HG 0 S 22 $\overline{\mathcal{D}}$ Ī 3 A8 1 A8 2 3 8 CODING SHEET 4 0 RE 8 3 $\overline{\mathcal{C}}$ BRK - Open loop Enter RET in INTEG 9 $\overline{\mathcal{C}}$ CLR - Close loop 2 3 A 0 0 0 Enter NOP in INTEG 8 8 9 Startanew log, 2 8 14 XII 0 0 Α 0 0 В C MICROCOMPUTER TRAINING SYSTEM MOV Μ D 5 RET 9 E OP F 0 0 2 3 0 N T REG-Force output low 0 F 1 CAL RST C D D 2 3 30 millisecond 2 0 4 2 00 17 5 I 6 8 7 M MO V 8 6 9 Wait for Times E I A 3 0 Α INTEGRATED COMPUTER SYSTEMS Line Ointerrupt В Ď 3 CN 2 7 O|U|TC D 9 C RIEIT Ε 0 NO F 0 1 2 3 4 5 6 7 8 Figure 6-28d

	A D D	R	co	DE	PWM	_	50	BRC	JUT	TIVE	, L	יים		VI	CAL	10N 2 B = 25
	8 16	0	2	\mathcal{B}		D	C	X		H						Test calculated
		1	7	C		M	0	V		A		\mathcal{H}				pulse width for
		2	B	7		0	R	A		A						zero or negative
_		3	2	3		H	>	X		\mathcal{H}						
SHEET	1	4	3	E		N	7	H		A		0	8			Load (A) to disable
		5	0	2												Times O interrupt
N		6	F	A		J	Μ	エ	N		8	/	7	/		and summ it
CODING		7	7	7												and jump if pulse width = 0
)		8	8													/
		9	7	0		8	0	1		A	,	L				Pulse widel >0
		Α	2	3		0		T		7	I	M	1			Load Times I for
		В	/	5												new pulse width
		С	7	C		M	0	V		A		Н				
EM		D	D	3		0	U	T		T	I	Μ	1			
SYSTEM		E	1	5		-										
		F	3	E		M	V	I		A		0	3			Load (A) to enable
RAINING	8 /7	0	0	3							,					
.¥	817	1	D	3		0	V	ア		P	0	R	\mathcal{T}	V	C	Times Ointerrupt Enable or disable
-		2	0	E												Times O interrupt
TEF		3	C	9		R	E	7								and enable
MICROCOMPUTER		4														Timer / interrupt
SON		5														
RO		6														
MIC		7				R	E	V	エ	S	E	\mathcal{D}	-	L	D	TI
_		8				F		R		1	U	L	L		S	TI CALE CONTROL
	`	9														
		Α				E	N	T	Ε	R		W	I	τ	H	
EMS		В					1	7)		=		C	A	LCULATED
ST		C														LSE WIDTH
A S		D				\mathcal{H}	F		(7	L)			j	ZERO
TEI		E					L	0	A	$\dot{\mathcal{U}}$		\top	I	X	Ш	RIAND
APU		F					F	N	A	\mathcal{G}	L	Е		$\bar{\boldsymbol{w}}$	0	TH INTERRUPTS
S	8	0														
ED		1				I	Ш		(1	Ţ.)			1	ZERO
IAT		2					\mathcal{Q}	I	S	A	B		Ε			IMER O TO
EGF		3						7	#	I	B	\mathcal{I}	T			HARGING
INTEGRATED COMPUTER SYSTEMS		4														
-		5														
		6														
		7														
		8														Figure 6-34

	A D D	R	со	DE	PWM		<u>5</u> 0	BK)O.I.	TNE	5 C	TO:	<u>ъ</u>		VER	SION 2
	8 18	0	2	A		L	H	L	D		8	3	A	6		CLOSL
		1	A	6						_						(L) - desired voltage
		2	8	3												atentry (A)= measured
Ы		3	9	5		3	U	B		L						(A) = -Error
SHEE		4	6	7		M	0	V		\mathcal{H}		A				(H) = - Error
	-	5	C	8		R	Z									Exit if error = 0
CODING		6	E	5		P	V	S	H		H					Save display data
3		7	2	F		C	M	A				_				
			4	F		M	0	<u>V</u>		2	,	A				
		9	3	E		Č	W	C		_						(BC) - + Error
		Α	9	F		S	\mathcal{B}	B		A	_	1		_		as two by tea
		В	4	7		M	0	V		B		A		_	_	
5	<u>_</u>		0	3		エ	ν	X	7	Ħ	_	_	>	.,		J
SYSTEM			2	A		L	H	L	\mathcal{D}		8	3	A	4		(HL) - old integral
SYS	· · ·		A	4						_	_					
		F.	8	3		4	_		•		_	• 1	1	_		
TRAINING	8 / 9		C	\sum_{i}		C	A.	L	レ		<u></u>	N		<u> </u>	G	
F. A			C	0												store new integral
		2	8							_)	2	,	7	_	(HL) ← (HL)+ (BC)/2"
5	<u></u> .	3	C	D		C	A	L	L		٦	R	\mathcal{O}	$rac{2}{2}$	G	add proportional term
MICROCOMPUTER			E	0											_	(HL) = (HL) + K*(BC)
8		5 6	8	7		C	A		L	-	L	7	$\overline{\tau}$	1		<i>P</i> 1 1: 1 11
S		7	1			<u> </u>	Π	_	_		_	D	-			Load Times / with
Σ		 8	8	7								-				new pulse wisth
ł		9	E	5	-	D	0	P		H		_				0 1 1 1 1 7
ŀ			C	9		R		$\frac{\prime}{T}$	_	′′						Lecover display data
SE SE		В					<u></u>	<u> </u>							_	
STE		C														
S		D	-													
빌		E							_							
<u>P</u>		F														
Š	3	0														
		1														
ĭĀŢ		2														
EGR		3														
INTEGRATED COMPUTER SYSTEMS		4														
_[5								_						
										_						
		7								_						
į		8														Figure 6-28f

•	A D D	R	CO	DE												
l	8 /C	0	0	0		\mathcal{N}	0	P		R	Ш	$ \mathcal{T} $				
		1	E	${\mathcal B}$		X	C	H	G							(DE) - old integral
		2	6	9	-	M	0	V		7		C				(L) + low luste error
		3	B	A		L	00	A			8	3	A	8		(L) ← low byte error (A) ← n for Error/2 n
SHEET		4	A	8												Erron/2m
		5	8	3			- 1									
N.		6	6	7		Μ	0	V		1+		A				(H) - m
CODING	810	7	2	حی		$\overline{\mathcal{D}}$	C	R		Н						Decrement count
		8	I	A			M				8	1	D	Z		Exit loop after O
		9		5												
		Α	8	/												
ľ		В	7	8		M	0	V		A		B				Set CY if error
		С	/	F		R	A	R			,					set CY if error
E		D	7	\mathcal{D}		M	0	۷		A		L				shift low byte of error right
SYSTEM		E	7	F		$\overline{\mathcal{R}}$	A	R			,					error right
		F	6	F		M	0	\checkmark		L		A				0
INING	8 / 2	0	B	7		0	R	A		A	,					Lest for error = 0
RAII		1	C	2		$\overline{\mathcal{L}}$	N	Ŕ			8	1	U	7		Loop unless O
<u>⊢</u>		2	C	7												
TER		3	8	/												If shifted error = 0
₽ P		4	2	C		\mathcal{I}	2	R		L						Makeerron = 01
MICROCOMPU	810	5	6	0		M	0	V		1		\square				
Ŝ.		6	C	D		C	A	L	L		À	D	T	0	V	(H) ← high byte error add (HL) ← (HL) + (DE)
<u>₹</u>		7	F	0												Return MINUS if
		8	8	/												arithmetic over flow
		9	F	8		R	M	I	7	Ü	S					0
()		A	2	2		S	H	L	\mathcal{D}		8	3	A	4		store new integral
EĞ		В	A	4												
/ST		С	8	39												
RS		D	C			R		<u>T</u>								
1		E	0	0		N	0	ρ								
퇼		F	0	0		N	0	P								
INTEGRATED COMPUTER SYSTEMS	8	0														
ED		1				E	2	T	E	R		W	エ	T	$ \mathcal{I} $	
₹Ā	· · · · · · · · · · · · · · · · · · ·	2						B	C	\mathcal{L}		=		+		ERROR (B=00/FF)
EG		3	L					H	L)		Ξ		0	L	D INTEGRAL
R		4			_	R	E		U	Ŗ	N					
		5						B	C)		P	R		2	ERVED
		6					۲	D	Ε)		=		0	_	D INTEGRAL
		7						4	L)		=		N	E	W INTEGRAL
		8	<u> </u>												L.	Figure 6-37a

PWM - SUBROUTINE PROPG CODE DA A CODING SHEET Exit after yero Add error into pulse width from INTEG and I N U 3 MP 8/E3 6 E B 7 8 9 Α В NTER (BC) WITH C MICROCOMPUTER TRAINING SYSTEM ERROR D HL PULSE WIDTH E PD AT ED BY INTEG F 0 ETURN 1 PRESERVED
PRESERVED
NEW PULSE WIDTH
INTEG + K ERROR 2 3 4 5 6 7 8 9 Α INTEGRATED COMPUTER SYSTEMS В С D E F 0 1 2 3 4 5 6 7

Figure 6-28h

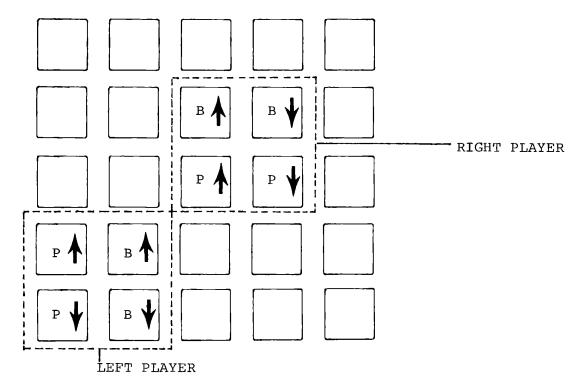
8

Figure 6-37b

STORE INTO MEMORY STARTING AT 8200.

DEPRESS. RST RUN

CONTROL KEYS ARE AS FOLLOWS:



P MOVE PADDLE UP

P↓ MOVE PADDLE DOWN

B MOVE BALL UP

B **♦** MOVE BALL DOWN

LOCATION 830C CONTROLLS BALL SPEED.

OC = NORMAL SPEED; 08 = Fast.

FIRST PLAYER TO SCORE 15 POINTS WINS.

TO START NEW GAME, DEPRESS ANY KEY.

Pong Game

Figure B-4

_	A D D	R	COL	DE	PON	J, 		 T T T	.AD	126		J 1 F	71/1			
	820	0	2	/	STAT	L	X	I		H		L	E	F	7	Clear score - Load
L		1	F	E												HL with score location
		2	8	2												
_[3	A	F		X	R	A		A						
SHEET		4	4	F		M	0	V		C		A				Clear Irall Control
		5	7	7		M	0	V		M	,	A				
왕	-	6	2	3		T	N	X		Η		•				
CODING		7	7	7	1	M	0	V		Μ	_	A				
악	820	8	3	E	INIT	M	V	Ī		A	,	4	0	Н		let faddle anition
ŀ	000	9	4	0				_			,			<u> </u>		Let Paddle positions
Ì		Α	3	2		S	τ	A		${\mathcal D}$	/					(1= top, 40 = middle
ŀ		В	F	8		<u>)</u>	 '	,,,		2	_	_	 		-	(1=10p, 40 = miaace
ł	·	c	8	3			-	-							-	8= bottom)
۶		D	3	8		S	7	A		7	8	_	<u> </u>	-	-	
빉			_	-		٦	1	/7		D	8				-	
SYSTEM		E	F	F		<u> </u>	-	_				_		_	-	
9		F	8	3		1.0	ļ.,	_					_		-	
RAINING	8 2 /	0	3	E		M	ν_	I		A	,	8	F	H	-	Set all SCAN lines
A.		1	8	F		Ļ	<u> </u>		-			_		ļ		on keyboard to O
ER T	_	2	D	3		0	U	Z	<u> </u>	ρ	0	R	7	0	C	
削		3	0	2		L	L.	<u> </u>	<u> </u>		_	L.,		_		
MICROCOMPUT		4	0	6		M	V	I		B	_د	4	0	H	_	Set initial ball
8		5	4	9	<u></u>	<u> </u>	<u> </u>		_	_	<u> </u>					position to middle
욄		6	2	/		1	X	工		14		D	4	L	<u> </u>	Set ball position in HL
影		7	F	\mathcal{B}				L	_		Ĺ					
		8	8	3												
	821	9	7	8	LOOP	M	0	V		A		\mathcal{B}				Load ball into display
,,		A	B	6		0	R	V		M						
EM	-	В	7	1		M		V		M	Ţ, <u> </u>	A				
INTEGRATED COMPUTER SYSTEMS		С	C	D		C	A	L	L		D	E	L	A	У	Delay for I frame
3.5		D	0	B												00
且		E	8	3												
J.		F	D	\mathcal{B}		I	N			P	0	R	T	0	A	Het lett planar
Š	3 22	0	0	0		Γ				Ţ <u>. </u>					1	Set left player
0		1	T				T			I^-		Г				7
ATE		2														
GR		3	\vdash	†		T				t —		1	T		 	
YTE.		4	†	1	 	 	\top	T	<u> </u>	 			 		\vdash	
=		5	\dagger	†	1	f^{-}	\dagger	<u> </u>	\vdash	 	 		†-	 	T	
		6	†-	†		T	†	T	\vdash	-	\vdash		T	+	\vdash	
		7	1	†	 -	1-	+-	T	 	 	\vdash		†-	+-	\vdash	
		<u>.</u> 8	\vdash	1	\vdash	\vdash	+	╁╴	+	\vdash			\vdash	+-	\vdash	Figure B-4a
	L			ـــــ				1		<u></u>	<u> </u>	<u> </u>	┸	1	<u> </u>	1-19410 10 44

A D D R CODE PONG - TEST FOR MOVE PADDLE

	A D D	R	co	DE	FONG								עעג			
	8	0														·
	822	1	E	6		A	N	I	·	/	/	/+		·		Bit 4= up, Bit 0= down
		2	1	/												''
-		3	E	A		7	P	E		N	\mathcal{C}	14	L			If both the same,
SHEET		4	2	F												ianore (no change
		5	8	2												left
SODING		6	E	6		A	N	I		0	1					See if down
100		7	0	/												0
_		8	/	/		1	X	エ		\mathcal{D}		D	1			Load paddle location
		9	F	8										_		/
	<u> </u>	Α	8	3												
		В	C	\mathcal{D}		C	A	L	L		C	H	P			Clarge paddle position
_	 	С	١	0												
STEM		D	8	3												
SYS		E	/	2		S	T	A	χ		D					Save
	822	F	\mathcal{D}	\mathcal{B}	NCHL	I	N			P	0	R	T	0	A	Get right player
INING	8 23	0	0	0										_		paddle info
RAI		1	E	6		A	N	I		0	C	H				Let right player paddle info Bit 2=up, Bit 3=down
RT		2	0	C												
ш	 	3	E	A		7	P	E		Ν	C	H	R			If both the same, ignore (no change) right) Lee if down
MICROCOMPUT		4	3	F												ignore (no change)
Ö		5	8	2												right)
SRO		6	E	6		A	N	I		0	8	H		Ĺ.,		See ifdown
ž		7	0	8										ļ 		
	ļ	8	/	1		<u>L</u>	X_	I		D		D	8			Load paddle position
	l	9	F	F												, ,
S		Α	8	3							_	_			<u>.</u>	
SYSTEMS		В	C	D		C	A	L	L_		C	H	9		_	Change paddle position
YST		С	0	0												, , ,
R S		D	8	3		_	_	_	-					_		
JTE	-	E	/	2		S	T	A	X_	_	D	_			_	Save
INTEGRATED COMPUTER	823	F	7	D	NCHR	M	0	V	-	A	-, -	1	<u> </u>			Save See if end of field
္ပ	3	0		ļ.			<u> </u>		_			<u> </u>	<u> </u>	_		0
ED		1	<u> </u>	-	ļ — — —	<u> </u>	_	_				_	_	_	_	
RAJ		2	_	_			-	_		_			ļ	_		
-EG	ļ	3	<u> </u>		ļ	_	<u> </u>	_		_		_		-		
N		4	_		ļ	<u> </u>	-	-		_		-		-		
		5	 	-	 	-		-	<u> </u>	-	ļ	-	-	ļ	ļ	
		6	-			-	 	-	_	-		-	-	 	-	
		7				-	_	-	<u> </u>			-	-	_	-	Figure B-4b
	L	8	<u></u>		<u> </u>	<u> </u>	<u> </u>	<u> </u>			L			<u> </u>	1_	righte p-40

			DE												
8 0	24/ 0	سر	E		U	ρ	H		F	8	H				If left of net
	1	Y	8												
	2	0	A		7	Z			L	$\bar{\omega}$	C				then -> LBC;
	3	7	C												then -> LBC;
	4	8	2												
Y	5	F	E		C	ρ	エ		F	F	Н				Elsa : I right sides
N N	6	F	F			•			İ	•					and the
CODING	7	0	A		2	Z			R	В	C				Else if right side of net then -> RBC;
	8	8	6		Ť										2000 3 100
	9	8	2		\vdash										
<u> </u>	Α	A	F		X	R	A	-	A						Elia alean) ald anitime
	В	7	7		M	0	1		M		A				Else clear old position
		B	9		1	M	ρ	\dashv	C	+	1				1.1.1.1.1.1.1.1.2
<u> </u>		C	A		2	7	H		N	B	C				Who has the ball? No one - no ball change
SYSTEM	E	-	8	-	۲	2		-	N	D	<u> </u>	_		_	110 one -> no ball change
	F	7	2		 	 				-		-		<u> </u>	
AINING 8 G		8	D		<u></u>	_	n		C				_		
<u> </u>	0 مى <u>2</u> 1	0	9		D M	C	R		7	ļ	C		-		Court (less) frame
THE STATE OF THE S		1	-			0	<u>۷</u>	-	/+	,	۷				
~ <u> </u>		3	F	<u> </u>	1	M	C				-			-	
5	3	1	7		R	A	<u>_</u>	-	_	_	_	~		-	
MICROCOMPUTE		7	A		2	C			<u> </u>	F	T	<u>C</u>			If Cy=1 then → LFTC
잃	5	6	3		-			Щ		 	<u> </u>				
š	6	8			_		 		_		_	<u> </u>	_	_	else → RTC
≅	7	D	B	RTC	I	N			P	0	K	T	0	A	
	8	0	0	ļ	ļ_					_	_			<u> </u>	
	9	E	6		14	N	$ \mathcal{I} $		\mathcal{C}	0	H			<u> </u>	Bit 7=down, Bit 6=up
ω	A	C		ļ	_	L				L	Ļ.,			_	
<u> </u>	B	E	A		2	P	E		N	C	H	<u> </u>			TY An= AL
\ 	С	7	/		_	<u> </u>	<u> </u>			<u> </u>	<u> </u>			_	then -> no change
R S	D	8	2		_	L_			L_:	L					
<u> </u>	E	E	6	ļ	A	N	I		8	0	H				Ty An = Ab then -> no change If down then
8 8°	250 F	8	0	<u></u>	<u> </u>				<u> </u>	_					then
<u> </u>	0		L_		L	_			_						
	1	L				_			L.	_		L	_		
¥	2	L											<u> </u>		
9	3														, , , , , , , , , , , , , , , , , , ,
INTEGRATED COMPUTER SYSTEMS	4				\prod										
	5														
	6														
	7						Ĭ								
	8														Figure B-4c

PONG - MOVE BALL (continued)

_	A D D	R	co		PONG	- r	40V	E 1	3AL						, 	
	8 26	0	\mathcal{C}	3		7	M	ρ		S	A	V	\mathcal{B}			→ go change
		1	6	C												
	_	2	8	2												
-[826	3	D		LFTC	I	N			P	0	R	T	0	A	Check for lett player
SHEET		4	0	0												change
		5	E	6		A	N	I		2	2	H				Chick for left player change Bit5-up; Bit 1= down
CODING		6	Z	8												7,
Š		7	E	A		J	P	E		N	C	H				JA Ax = A.
1		8	7	/												If As = A, then -> no change
Γ		9	8	2							-					
Γ		A	E	6		A	N	I		0	2	H				If down
		В	0	2		_	-	_								then
r		С	7	8	SAVB	М	0	V		A		\mathcal{B}				Load old ball position
בוֹב		D	0	D		C	A	Ĺ	L			Н	B			Change ball position
STEM		E	0	1	÷						_	, <u>, -</u>		-		Sary sar garage
S -		F	8	3												
ž l	8 27	0	4	7		Μ	0	V		B		A		-		Save in ball's reg.
	827		7	9	NCH			_		A	-	0				It to be said the
뛰	0 4 1	2	E	6	7.07.		N	エ		0	F) }		-		Is a + + B
		3	0	C		 ''	10	=	 -	0	-	//				Het change reg. If count + 8 then - noball change
51-		4	C	2		7.	N	7		N	B	C		-		Anen - Mobaci Change
MICROCOMP	·	5	7	8			//	<u>_</u>		/ /	U		-	-		
<u>S</u>	·	6	8	2										-		
핡		7	0 4	F		M	7	V	-	0		A				Else set count = 0
	827				NAC					H	,	1 1		-	-	
\vdash	0 & 7	9	X N	<u>رم</u>	NBC	=	C	0		L		0	ρ	-		Change ball position -
ŀ		A	<u></u>	9		۲	1	' -		_			-	-	-	eune up or aour
S -		В	8	2		 			-					1	-	
Ӹ		 C	M	E	LBC	M	0	V		A		M				Was Alley assitions +
5		D	B	8	-30	6	M	P		В	-,-	1.1			-	If paddle position & ball position then - left lose
		<u>-</u> E	C	2		7	N	己		1	1	0	S	E	┢	vall position
5		F	9	8	ļ	٦	10	_		-	<u> </u>		3	-		then - life lose
٦	s 28	0	8	2		<u> </u>	-			-			-			
ŌĹ	0 00 8	1	8	a		-	-	\vdash		_			-			
맑		2	-	-		-	-	<u> </u>			-			 	-	
왕		3				-	-			-	<u> </u>	-	-		-	
INTEGRATED COMPUTER SYSTEMS		4	-		<u> </u>	_	-		-	-	-		 	-		
╧├		5	-			-		-				 			-	
+	 	6	-			-	-		-			-	-	-	-	
+		7	-			-	-		-						-	
L	-	8	-	<u> </u>			-	-			-		+-	† –	+-	Figure B-4d

	A D D	R	co	DE .	PONG	- 1	Œν	EKS		BAL	، رُ بار	50	JUK.	C		В - 35
	8	0														
	828	1	0	E		M	V	I		C		8	3	H		Elegand by the contra
	<u></u>	2	8	3	_	,					_					Else new ball counter
		3	C	3		7	М	Ρ		R	E	V			_	Goreverse direction
핆		4	8	Ď												Survivorse destriction
팖		5	8	2		_		1		-		-				
읽	828	6	77	E	RBC	M	0	V		A		Μ				11 110
CODING	U ac v	7	B	8		7	M	P		B	. 1	1.1				of peddle position #
ŏ		8	C	2		し 力		Z		R	1	0	~	E		ball position
ŀ		9	9	E		<u> </u>	//	_		-	_)	_		then - right lose
ŀ		A				-	┝		\dashv			-				
ŀ			8	2		M	 	7		C		_	2	11		
}		B C	0	E		111	V	I	-	\vdash	,	0	3	\mathcal{H}		Set new ball counter
Σ	000		0	3		_	7			. 1	0		_			
STEM	828	D	3	A	REV	<u>_</u>	D	A		N	<u>B</u>	C	<u> </u>		\vdash	Liverse ball direction
SYS		E	7	•			}	\vdash				<u> </u>				INX H/DCX H, bit 3 nia XOR (Classical
9		F	8	2						_		.,	ļ.,	ļ		INX H/DCX H, lit 3
Z	8 29		E	E		X	R	I		0	8	Н	<u> </u>			nia XOR (Classical
ΑA		1		8		•						<u> </u>	<u> </u>			Kludge)
۳		2	3	2		S	Т	A		N	B	C	<u> </u>			
JE		3	7	8			l									
MP	_	4	8	2								L	_			
MICROCOM		5	C	3		2	M	P		N	В	<u>C</u>	L.	<u> </u>		So change tall position
읪	<u> </u>	6	7	8								L	<u> </u>			
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\		7	8	a												
	829	8	2	1	LLOSE	L	X	I		11	_	L	E	F	T	add to right socone
		9	F	E							Ĺ		L.,			0
,,		Α	8	a												
EM		В	C	3		2	M	5		S	C	0	R			_
/ST		С	A	1												
3.5)		D	8	2								L				
INTEGRATED COMPUTER SYSTEMS		E	2	1	RLOSE	L	X	I		H		R	I	G	H	Tadd 1 to left's some
Ð		F	F	F							,					
Š	3 2 A	0	8	2												
0		1	Ī													
ATE		2														
GR		3	\vdash			T		1	<u> </u>			\vdash		T	-	
TE E		4	 			\vdash		 	T			<u> </u>				
ے		5	 				+	 			H	T	T	t	 	
		6	†	\vdash	 	-	T	 	t^-		\vdash	\vdash		\dagger	 	
		7	\vdash	-	 -	\vdash		 			\vdash	\vdash	\vdash	+-	\vdash	
		8	+-	\vdash	 	\vdash	\vdash	+-		-	\vdash	+	\vdash	\vdash	-	Figure B-4e
	L	<u> </u>				<u> </u>	_[Ц_	<u> </u>	L	<u> </u>	Ц_	Ц	1	<u> </u>	rigure B-4e

PONT - SCORE (continued)

	A D D	R	coi	DE	PONT			<u>π</u> Ε	(0	Ont	. III	uec	ı) 			
	8	0											_ [
	82A	1	F	E	SCOR	2	0	V		A		\overline{M}				Load score and
		2	C	6		A	\mathfrak{D}	Н		0	/	Н				add!
[3	0	7												
SHEET		4	2	7		(A)	A	A								with decimal adjust
	 	5	7	7		M	0	٧		M		A				Save
S S		6	F	E		C	P	I		1	5					JY (A) \$15
CODING		7		3			-	_			-					
0		8	1	2		7	2	Z		N	W	I	N			dans as winder
Ì		9	D	5			<u> </u>			•	70	-				then -> no winner yet
l		A	8	2							_					
Ì		В	2	\overline{c}		T	Ν	R		L						£0.0
		c	2	커		<u> </u>	X	エ		<i>H</i>		D	8			Else Jx L+1=0
돍		D	F	É			^				1	U	U			9
SYSTEM		E	8	3		-				-						
		F	0	A		7	マ			R	T	L				4 11 +
TRAINING	8 2 B	0	B	<i>//</i>		V	_	_		17	1	J			-	then right lost
2	· & D		8	2												
H H		2		_		M	V	エ	_	L		_	B	Н		1 1 1 1
		3	2	E		1-1	V	<u>_</u>		<u>_</u>	1	F	<i>i</i> S	11		else left lost
Ž	BAR	4	F	B	RTL	P		۲			H		-			
MICROCOMPUTER	82B	5	E	5 A	KIL	<u>C</u>	UA	7	H		<u>C</u>	1	1	6		Save position Clear display
ö		6		7		\vdash	117	_	١		L		E	1	14	Clear display
임		7	8	2		_	-				_				-	
≥			0	-		0	_	2		11					-	
		9	E 3	/		P	0	PI	-	7 7	_	л.	-	11		
		Α				Μ	V	1		141	+	<u>'</u>	E	-		
S			-	9		7	_	. ,		11					-	
圁		В		B		D	<u></u>	X		H			_	11		
SYSTEMS		C	3	6		M	V	工	L	M	-	11	S	-		
E.S		D -	6	5 6		_	_		-	1.1				<u> </u>	-	
INTEGRATED COMPUTER		E	280	B		D	C	X		H		"	_	,,	-	
MP		F .	<i>S</i>			Μ	V	工		M	1	<u> "</u>	0	<u> </u>	-	
00	3 2 C	0	3	F				<u> </u>		<u> </u>	Ļ.		-	-		
TED		1				<u> </u>			-	<u> </u>		_			 	
RA.		2	_	_		<u> </u>		<u> </u>	_			<u> </u>		_	<u> </u>	
EG		3				_		<u> </u>	_					_	<u> </u>	
IN		4	_			_	_	_	_			_		_	<u> </u>	
		5	_	L.		<u> </u>	<u> </u>	_	_			_	_	_	 	
		6	<u> </u>	ļ			 	<u> </u>	_	_		_		_	 	
		7	_			_	ļ					_		_	<u> </u>	
		8		<u> </u>			<u> -</u>	_	<u></u>				<u> </u>		<u> </u>	Figure B-4f

- -

	A B B	В	co	D.E.	PONG	– S	STA	RT	NE	w c	AM	E				в - 37
[8 D D	0 0		٦٤												
	82C	1	2	B		D	C	X		H						
		2	3	6		Μ	V	I		M		. //	L	"	i	
		3	3	8							7					
SHEET		4	3			M	V	I		A		F	F	H		Hold display
		5	F	F							7					
CODING		6	C	۵		C	A	L	L		D	I	S			
릵		7	0	D												
Ĭ		8	8	3						,						
		9	D	\mathcal{B}	NKY	Н	N			P	0	R	٢	0	A	Wait for key to begin new game
		A	0	0												beain new game
		В	B	7		0	R	A		A						
Ì		С	E	A		5	Ρ	E		N	K	У				
-EM		D	C	9												
SYSTEM		E	8	Q												
G S		F	C	D		C	A	L	L		C	Γ	E	A	R	Clear display
TRAINING	8 20	0	8	7												
١Ą٢		1	0	2												
ЯŢ		2	C	3		2	M	Ρ		2	7	R	7		_	Startnew game
MICROCOMPUTER		3	0	0			_									
MPU		4	8	2												
CO	, <u>.</u>	5	C	D	NWIN	C	A	L	L		C	L	E	A	R	
;RO		6	8	7												
MIC		7	0	2		L										
		8	2	1	<u> </u>	4	X	I		14	<u> </u>	L	E	F	7	Load left losing score
		9	E	E		L_				<u> </u>	Ĺ					
S		A	8	2												
EM.		В	/	/		1_	X	I		D		D	6	_		Display on right
YST) 	С	F	0		_	_			<u> </u>	Ĺ			<u> </u>	<u> </u>	0.
S		D	8	3		<u> </u>	L.	<u> </u>	<u> </u>	_	<u> </u>	L_		_	_	
JTE	ļ	E	7	E		M	0	V	_	A	با	M				
MP		F	C	D		\mathcal{C}	A	L	L		D	В	У	2		via monitor
ဗ	3 2 E	0	9	8		_		<u> </u>	_				_		<u> </u>	
Ë	82E	1	0	2		<u> </u>		_		_	-	 		<u> </u>	L.	
RAT		2		_		_		<u>L</u> .		_	_	_		_		
INTEGRATED COMPUTER SYSTEMS		3	<u> </u>			_		_	_	<u> </u>	ļ		<u> </u>	_	_	`
INT		4	L	_		<u> </u>	_	<u> </u>		_	_		<u> </u>	L		
	ļ	5	<u> </u>	_		_	<u> </u>		_					_	_	
		6	L	<u> </u>		<u>L</u>	L	<u> </u>				_	L_	L		
	l	7	1	1	1			l	i	1	i		l	1	1	

Figure B-4g

	<u> A D</u>	D	R	_co	DE	PONG	- 5	TA	RT	NE	w c	AM	E	(co	nti	inu	ed)
	8		0														
			1														
	80	?E	2	2	3		I	N	メ		\mathcal{H}						Let right losing score
—			3	7	E		M	0	V		A	,	M				Get right losing score
EET			4	C	8		C	A	L	L		D	B	Y	2		Display on left
S SH			5	9	8												
CODING			6	0	2												
ΩOΩ			7	3	E		M	V	I		A		4	0	H		Hold for a while
		-	8	4	0							7		_			
			9	C	D		C	A	L	L		A	I	2			
			Α	0	D												
			В	8	3												
			С	C	Σ		C	A	L	L		C	L	E	A	R	
STEM			D	8													
/ST			E	0	2							-					
GSY			F	A	F		X	R	A		A						
TRAINING	8 2	F	0	4	F		M	0	V		C		A	ļ			clear ball counter
AIN			1	C	3		_	M	ρ		I	N	工	T			I back
			2	0	8				_								
rea			3	8	2									-			
MICROCOMPUT	80	F	4	E	E	DOWN	C	P	ュ		0	8	H				then do not alonge
NO:			5	0	8			<u>`</u>			_		<u> </u>				along aleas and my
300			6	0	8		R	7	-		-						and alaman anaition
/IC			7	B	7	!	7	Z	A	-	A			-			then do not change else clear carry and change position (A, -> AN-3)
~			8	1	F		R	A	R					-			
			9	1	F		R	A	R		 	-	\vdash		 -	-	
			Α	1	F		R	A	R	-	<u> </u>	-		-	-	-	
MS			В	C	9	<u> </u>	R	E	T	 - -	 	-		_			
SYSTEMS	_		С		/	-	-		-	-	-	-	-	1			
S			D				_		<u> </u>	-	-				-	-	
FER	82	=	E	\vdash		LEFT		-				-	<u> </u>	-	†		file mine (inter pane)
2	82		F			RIGHT	-			-			<u> </u>	-	-	-	Lefts misses (right's score) Right's misses (eift's score
Ö	3	<u>'</u> –	0	 				-			\vdash			-			regard (light & North
C	<u> </u>		1	1	-							-			T		
ATE			2	1							\vdash		 			-	
GR,	<u> </u>		3	\dagger	 	 		-	 	1		-	-	 	-	 	
INTEGRATED COMPUTER	ļ		4	+	† –				\vdash	-	\vdash				1	-	
=			5	T	\vdash							-		<u> </u>			
	 		6	1	†-	 	-		<u> </u>				 				
	<u> </u>		7	1				1	\vdash	-	\vdash		T			-	
	 -		8	1		-		-	 	-			\vdash	† –	<u> </u>		Figure B-4h

	A D D	R	çoı	DE	PONG	- 8	UB			NES						B - 39
	8 30	0	/	A	CHP	L	\mathcal{D}	A	X		D					Change paddle
		1	\mathcal{C}	A	CHB	2	2			\mathcal{D}	0	W	N			Classime we have
		2	F	4				_								position for ball
ı-		3	8	Q												
SHEET		4	F	E		0	Ъ	I		0	/	\mathcal{H}				Lyup and on top
		5	0	/												
CODING		6	\mathcal{C}	8		R	Z									then do not change
30		7	/	7		3	A	L								
	L	8	1	7		R	A	L							_	
		9	/	7		R	A									
		Α	C	9		R	E	7								
		В	3	E	DELAY	M	V	I		A	_	0	C	H		Delay for I frame
	830	С	0	C	**						_					Delay for / frame * * BALL SPEED**
SYSTEM	830	D	1	1	DIS	L	X	I		Ø		0	7	B	4	Main delay
YSI		E	\mathcal{B}	4							_					1'
1G S		F	0	7			<u> </u>					_				
MICROCOMPUTER TRAINING	8 3 /	0	/	D	DEL	D	C	R		E	_	_				Delay by decrement
RAI		1	C	2		2	N	7		D	E	L		_		
R.		2	1	0			_	_	<u> </u>	ļ _		_				
JTE		3	8	3	L	<u></u>	L									
MPL		4	1	5	L	\mathcal{D}	-	R		D	_	_	_		_	
Ö		5	C	2	<u> </u> 	2	N	7	L_	D	E	C	<u> </u>	ļ_		
SRO		6	1	0		_	<u> </u>		L		_	_	_	_		
Σ		7	8	3					_	ļ_				_		
		8	3	D		D	C	R		A	_	_	_	_		
		9	C	2		2	N	7	_	D	I	5	_	_		
$\overline{\mathbf{s}}$		Α	0	D		L	<u> </u>	_	_	<u> </u>	_	_	_			
Ē		В	8	3		_	ļ		_	ļ.,	_		<u> </u>	_		
YS.		С	C	9		R	E	7	ļ _	_	_		_			
INTEGRATED COMPUTER SYSTEMS		D	igspace	_	<u> </u>	_	-	<u> </u>	_	-	-	ļ	_	_	-	
UTE		E	-	_	<u> </u>	_	ļ.,	<u> </u>	Ļ.	-	_	ļ	-	_	_	
Ā		F	-	-	**		Ĭ	C	H	A					13	ALL SPEED
8	3	0	<u> </u>			_	B	Ä	ļ	Š	4	Ð	N	G	I	NG LOCATION
TED		1	\vdash	-	 	-	8	3	0	C	-	F	R	0	M	OC TO 08)
PA.			-	 -		-	-	-	 		-	├	-		 	
reg		3	-	-	ļ	┞	-	-	<u> </u>	-	 	-		-	-	
Z	-	4	↓_	├-	ļ	1	+	-		ŀ	 	-	-	 	<u> </u>	
	 	5	<u> </u>	 -	-		-	-	-	\vdash	-	+	-	+	}_	
	 	6	↓_	-	 -	-	-	┼	-	-	\vdash	-	-	├-	\vdash	
		7	↓_	-	ļ	₽-	╀-	<u> </u>	ـ	-	-	╁	 	-	↓ _	
		8	上		l		<u></u>	<u> </u>		<u> </u>	L_	<u>L</u>				Figure B-4i

This page intentionally left blank.

MICROCOMPUTER INTERFACING WORKBOOK

APPENDIX C

RS 232c INTERFACE SYSTEM

This page intentionally left blank.

RS232C INTERFACE SYSTEM

The RS232C Interface System consists of the following I/O driver subroutines and Interface Training System board connections.

Software is used to convert parallel data, transmitted by the MTS, into a serial stream of bits consisting of one low start bit, eight bits of data, and one high stop bit. A Software delay subroutine is used to produce a baud rate of 300 bits/second or 30 characters/second. This delay program may be modified to allow transmission rates of up to 4800 baud (480 characters/second). The ITS hardware option and connections of Section C.1 are required to convert the digital data up to the standard +/- 12 volt EIA-RS232 connector in order to communicate with a terminal.

C.1 ICS RS232 SOFTWARE

The RS232 Software includes the following subroutines:

- * IMSG Input a line of text from the terminal, terminated by a Carraige Return, via ICHR.
 - Entry Point 8297
- * ICHR Input a character from the terminal into register E via

 Port 1A0.
 - Entry Point 82B0
- * OCHR Output a character from register E to the terminal via

 Port 1C1.
 - Entry Point 82D0.
- * OMSG Output a block of characters, terminated with a CNTL-C (O3H), to the terminal via OCHR.
 - Entry Point 82F6.

Specifications, flowcharts and source code for the above programs are given in Section C.3.

C.2 REQUIRED CONNECTIONS FOR RS 232 INTERFACE

C.2.1 Parts List

QTY	Description
1	16-pin DIP plug
7	12" lengths of ©22 gauge wire jumpers
1	25 pin CANNON DB-25s-5 connector
1	1 K ohm resistor - 1/4 watt

5 spade clips (optional)

* -12 volt power supply (most terminals)

C.2.2 Fabrication Procedure

DIP plug connector

- 1. Solder a jumper wire to pin 2 of DIP plug. (Fasten spade clip to other end of wire.)
- 2. Solder a jumper wire to pin 16 of DIP plug. (Fasten spade clip to other end of wire.)

CANNON connector

- 3. Connect pins 5, 6 and 8 together.
- 4. Solder one end of a jumper wire to this connection (pin 5, 6 or 8).
- 5. Solder other end of jumper to the 1K resistor (insulate solder joint with heat shrink tubing or electrical tape).
- 6. Connect pins 1 and 7 together.
- 7. Solder a jumper wire to pin 1 (or 7). (Fasten spade clip to other end of jumper.)

- 8. Solder a jumper wire to pin 2. (Fasten spade clip to other end of jumper.)
- 9. Solder a jumper wire to pin 3. (Fasten a spade clip to other end of jumper.)

C.2.3 ITS Board Connections

- 10. CANNON pin 1 (or 7) to ITS ground (TIE Block GND or CASSETTE GND).
- 11. CANNON pin 2 to ITS terminal strip pin 4 (RS 232 REC).
- 12. CANNON pin 3 to terminal strip pins 2 and 3 (TTY SEND and TTY RET).
- 13. CANNON pin 5 (or 6 or 8) jumper with 1K resistor to TIE Block +12v

C.2.4 -12 Volt Power Supply

- 14. Connect the -12 volt power supply lead and its ground to the ITS TIE Block -12 and GND tie points, respectively.
- * NOTE While the RS 232 standards specify a high (or 1) signal level of +12 volts and a low (or 0) signal of -12 volts, some terminals (such as the Lear-Seigler ADM-3) will recognize a signal level of 0 volts or less as a low. For these devices the -12v supply is not necessary.

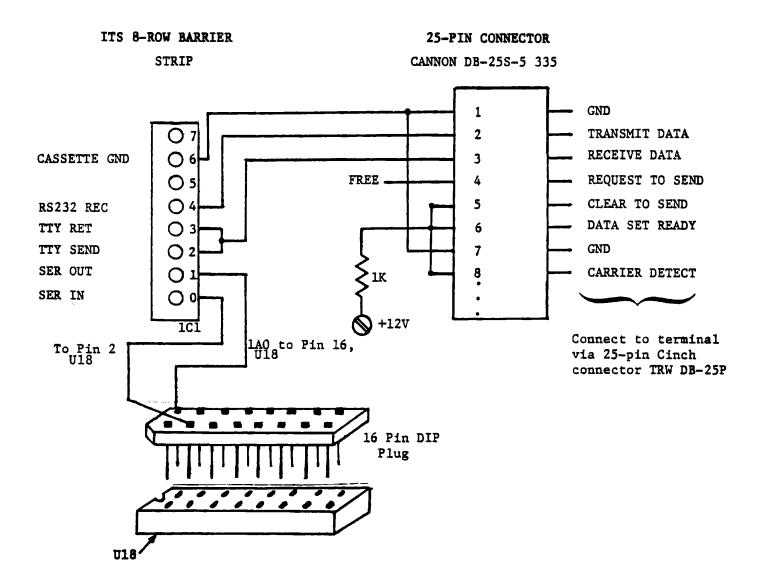


Figure C-1

TERMINAL SIGNALS	25-PIN CANNON CONNECTOR	ITS CONNECTIONS	1	NOTES	
GND	I	GND	Screw	terminal	<i>‡</i> 7
TRANSMIT DATA	2	RS232 REC	Screw	Terminal	<i></i> #4
RECEIVE DATA	3	"TTY SEND"/ "TTY RET"	Screw	Terminal	#2&3
REQUEST TO SEND	4	unconnected			
CLEAR TO SEND	5	Pulled up to +12V through 1K resistor	1		
DATA SET READY	6	Pulled up to +12V through 1K resistor	ı		
GND	7	GND			
CARRIER DETECT	8	Pulled up to +12V through 1K resistor	1		

SUMMARY OF SIGNAL CONNECTIONS

Figure C-2a

BIT 8 = 0

Parity = INH

STOP = 1

Data = 8

Parity = Don't Care

RS232

FDX

BAUD RATE = 300

ADM3A SWITCH SETTINGS

Figure C-2b

C.3 ICS RS232 SOFTWARE SPECIFICATIONS

The following pages contain the formal specifications, flowcharts, and source code listings for the RS232 Software. The student is advised to note the format used in this documentation for his/her own efforts.

C.3.1 Subroutine IMSG

This subroutine inputs a string of characters from the terminal and echos the characters back to the terminal until it encounters a Carraige Return character (ODH). The routine stores the input string beginning at the address specified in the HL register pair. The parity bit (high order bit 7) is masked out before the ASCII character is stored. The last character in the buffer is always ETX (O3H or CNTL-C). Register C contains the maximum input string length acceptable (input buffer size).

The PMPT entry point outputs a question mark ('?') as a prompt character prior to invoking IMSG. All arguments are the same.

* Entry Point 8297

* Arguments

- Upon entry
 - H,L register pair contains the input buffer start address
 - C contains the maximum input buffer length minus 1

* Upon return

- C contains the number of input buffer bytes remaining

- A contains 03H
- B and D contain 00H
- E contains last input character
- H,L register pair contain the address of the last input buffer entry (the CNTL-C, 03H, or ETX)
- The Input Buffer contains the inputted character string where the last character in the buffer is an ETX (O3H or CNTL-C) character. The Carraige Return character is not stored.
- The Stack has been used and restored.
- * Subroutine used
 - OCHR at 82D0H
 - ICHR at 82BOH
- * Other entry points
 - PMPT Output a prompt character ('?') before invoking IMSG. Entry point 8290H
- * Sample usage

21	00 83	LXI	H, INBF	Load input buffer address	nput buffer addres
OE	1 F	MVI	C,BFLEN	Load input buffer length	nput buffer length
CD	90 82	CALL	PMPT	Prompt with '?' and input string	with '?' and inpu

Figure C-3

IMSG Source Code

A D D	R	со	DE											
8 29	0	C	5	PMPT		U	S	H		B				Prompt! Output '?' as a prompt character
	1	/	E		M	V	\mathcal{I}		E	,_	//	Ś	"	· Output '?' as a
	2	3	F											prompt character
<u>-</u> [3	C	D		C	A	7	۷		0	\mathcal{C}	H	R	, ,
SHEET	4	0	0											
	5	8	2											
N.	6	C	/		P	0	P		\mathcal{B}					
9 8 29	7	C	5	IMSG	P	V	5	H		\mathcal{B}				TMSG Entre
	8	C	D		C		7	7		BI	C	Н	R	IMSG Entry Yet a character from the keyboard
	9	B	0			-							_	the best hand de
	Α	8	2				_							Service -
	В	E	6		A	N	$\overline{\mathcal{I}}$		7	F	-	-		
	С	7	F		' 		-	-	'	•				Mask-out the Parity Bit, but 7 (MSB)
<u>N</u>	D	7	7		M	0	1/		M		A			- Serve ((7-13D)
SYSTEM	E	0	Ź		C	A	7	L	1.1	6	C	H	R	Ent the in the
	F	5	0		\vdash	//	ب				ب	//		Echo the inputted character back to CRT
8 2 A		8	2		-	-	-	-, -	-					Characle back to CRI
Z ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	1	0	1	-	P	0	P		B	_				
품	2	77	0	-	M	0	1)		A		E			
<u></u>	3	/	7	<u> </u>		0	V	_		Ŕ	L		_	0-3
5	4	1	E		\vdash	10	<u> </u>	-		K			_	Is it a Carriage Cetural
MICROCOMPUTER		0	2		-	7			_		1 I			
8	6	C	7		2	Z			D	0	N	E		Yea; finish up.
S		A	D	-	-	-			_					
≅	7		2	<u> </u>	 -	_	_			-				
	8	0	2		D	C	R		C	_				No;
	9	2	3			N		-	H			<u> </u>		Update buffer pointer
<u>s</u>	A	C	2		7	N	7	_	I	M	S	G		and character counter
릴	В	9	7		_	!		-						Is Suffer full?
\s\ \	С	8	2		<u> </u>	-								No; Stetnertcharacter Yes; Put EXT character
5 82A	D	3	6	DONE	M	V	I		M	_	0	3		Yes; Put EXT Character
<u> </u>	E	0	<i>3</i>			_								into buffer and
NTEGRATED COMPUTER SYSTEMS 8 7 8 7 8	F	C	9	<u> </u>	R	E	7			_				return.
S 8	0			ļ			_							
	1	_	_	 		-		_			_		_	
¥	2					_					_	_		
<u> </u>	3					L_			_	_				
닐	4	_	_		L.	_			L				L_	
	5	_					<u> </u>			<u> </u>	ļ	<u> </u>		
	6	_	_		<u>_</u>	_						<u> </u>		
	7	L	_			_								
1	8	1	1	I	1	1	1	1	1	1	l	1	1	Figure C-4

C.3.2 Subroutine ICHR

This subroutine inputs a character from the terminal keyboard through Port 1AO, with no translation, into the A and E registers.

- * Entry point 82B0
- * Arguments
 - No entry arguments
- Upon return
 - A contains the last character inputted
 - E contains a copy of A (input character)
 - B and C are OOH
 - D, H and L are unaffected
- * Subroutines used
 - DLY of OCHR (at 82E9)
 - DLY1 of OCHR (at 82EB)
- * Sample usage

CD BO 82 CALL ICHR Input a character

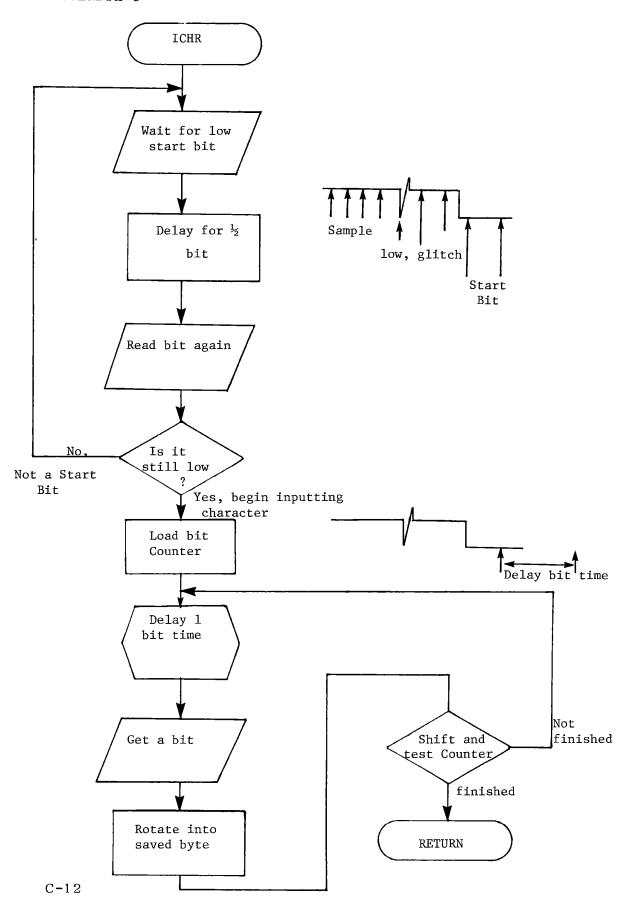


Figure C-5

_	A D D	R_	col	DE _		G	E	7		K	/	B		C	Н	AR INTO REGA
	8 2B	0	D	$ \mathcal{B} $	ICHR	\mathcal{I}	2		P	0	R	\mathcal{T}	1	A		Get a list
[1	0	4												
ſ		2	/	F		R	A	R								
_[3	0	A		ナ	C			\mathcal{I}	C	H	R			Wait for law
SHEET		4	\mathcal{B}	0	_											start bit
		5	8	2												
CODING		6	0	E		M	V	I		7		2				Delay La 1/2 bit
릵		7	0	2							7					Delay for 1/2 bit
		8	C	D		C	A	1	7		D	4	У	1		
Ī		9	E	B												
ĺ		A	8	2												
Ì		В	2	B		I	N	-	P	0	R	T	7	A		Lead lit again
Ì		С	0	4									_			· · · · · · · · · · · · · · · · · · ·
E		D	1	F		R	A	R				-				Joit still low?
SYSTEM		E	D	A		J	C			I	C	H	R			False alarm
		F	B	0												
NING	8 2 C	0	8	2												
RAII		1	1	E		M	V	I		E		8	0			Bit Counter
F		2	8	0	_						7					
凹	82C	3	C	D	ILP	C	A	۷	2		0	7	У			Delay Lell lit
JP.		4	E	9												Delay full lit
S		5	8	2												
MICROCOMPU		6	D	B		I	N		P	0	R	T	/	A		Let a bit
<u>8</u>		7	0	4												
ļ		8	1	F		R	A	R								$CY \leftarrow (A_0)$
		9	7	B		RM	0	V		A		E				Let previous lists
ζ,		A	/	F		R	A	R								$A_2 \leftarrow (cy)$
EM		В	5	F		M	0	V		E		A				Save new laste
/ST		С	D	2		J	N	C		I	4	P				Get next list unless CY shows
R S		D	C	3		L	Ľ.		<u> </u>					<u> </u>		unless CY shows
TE		E	8	2		L			L	Ĺ		Ŀ	L			we've got all 8
INTEGRATED COMPUTER SYSTEMS		.F	C	9		R	E	7								
Ö	8 2 D	0														
ED		1					<u> </u>					_				Returns with last
₹AT		2														
EGF		3														in Rega A and E
N		4														
		5							L							
	· <u> </u>	6	_			L								L		
		7														
		8					<u> </u>	L	<u></u>						L	Figure C-6

C.3.3 Subroutine OCHR

This subroutine outputs the character in the E register to the terminal via Port 1C1 at 300 baud (30 chars/sec). The baud rate may be changed by changing the programmed delay in the DLY subroutine at address 82ECH.

Other entry points are DLY and DLY1 for the appropriate delays for the 300 baud (or higher) data rates.

* Entry point(s)

- 82D0 for OCHR
- 82E9 for DLY
- 82ED for DLY1

* Arguments

- Upon entry
 - E contains the character to be outputted
- Upon return
 - E contains the outputted character
 - A contains 03H (or ETX)
 - B, C and D contain 00H
 - H and L are unaffected

* Sample usage

O6 21 MVI E,"A" Output an 'A' to the CRT CD DO 82 CALL OCHR terminal.

* Other entry points

- DLY at 82E9
- Causes the appropriate delay for a 300 baud data rate. The baud

rate may be changed up to 4800 baud by recoding address 82EC as follows:

82 EC	baud r	ate	chars/sec
70H	300	30	
38H	600	60	
ОВН	1200	1 20	
0 4 H	4800	480	

- DLY1 at 82EB

- Alternate delay entry allows 1/2 bit time delay (used by ICHR). Also allows 110 baud rate delay. User must load register C with appropriate counter (see source code for OCHR).

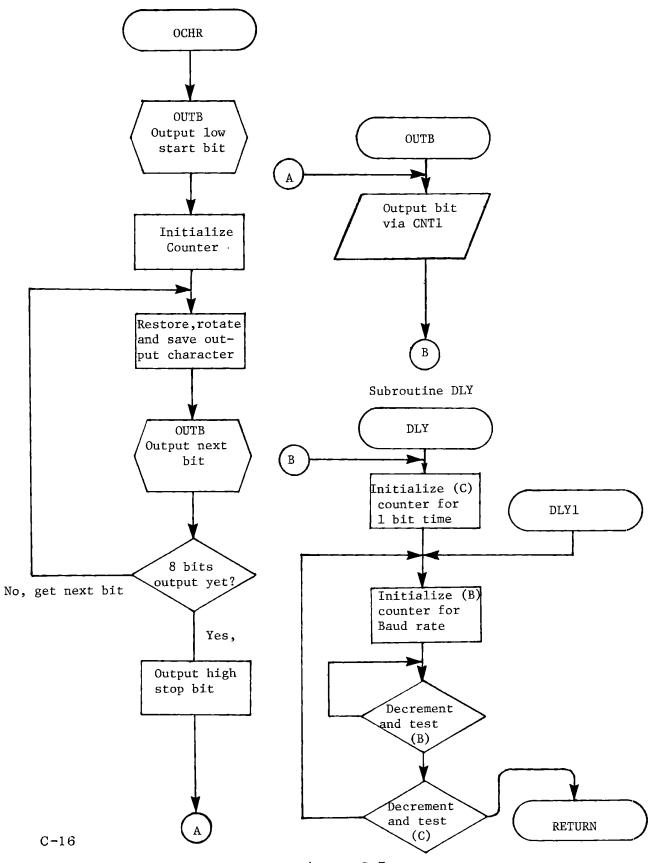


Figure C-7

	A D D	R	cor	DE_											
	8 2D	0	3	E	OCHR	M	V	I		A	_	0	2		Output a low start bit
		1	0	2											start bit
		2	$ \mathcal{C} $	\mathcal{D}		C	A	4	L		0	U	T	$ \mathcal{B} $	
_		3	E	1											
EET		4	8	2											
SH		5	A	F		X	R	A		A					Clear carry
S		6	/	6		М		\mathcal{T}		D		8			Suitialine bit Counter
CODIN		7	0	8			Ť				1				Jan Charles
O	820	8	7	B	OLP	M	0	V		A		E			Het with to character
		9	0	7	_	R		C	-						Het output character
		Α	5	F		M	0	V		E		A	-		Save rotated characte
		В	3	E		M	_	I		A	7	0			
		С)	7		,.,	<u> </u>			Ė	,				YCY then CNT/ ← 03 else CNT/ ← 02
E		D	8	F		A	D	6		A				-	
/STE		E	0	<u> </u>			A		ム	-	0	()	T	\mathcal{B}	D. T. + 1:+
SS		F	E	7			π		_		<u> </u>		 •	_	Output bit
N	8 2 E	0	8	2		 			-				<u> </u>	-	<u> </u>
A	- 22	1	1	5		D	C	R		D		-			Count lits until
H		2	0	2		<u>J</u>		7	-	0	L	P		-	all 8 are done
FR	-	3	D	8		Ť	-	٦				Ė		 	de o are more
P.		4	8	2	-		-	ļ						-	
MICROCOMF		5	3	E		M	V	\mathcal{I}		A		0	3		Tiched To thick
Š		6	0	3		/	-	_		,	-			┢	stop but Output the bit
1CF	82E	7	D	_	ουτβ	0	U	$\overline{\tau}$	 		N	7	1		1 7 + + + 1 + 1 +
2	092	8	0	7			-		_		//	<u> </u>		-	tulput the bit
	82E	9		Ė	DLY	M	1/	I	-	0	_	0	3		
	0 & L	A	0		<u> </u>	, , ,	<u> </u>	-	-		,	٢		-	list time
MS	82E	В	0		DLY I	M	1/	I		В		9	4	ļ	Load times for band
STE	042	С	7	0	<i>J L j</i> .	 ` `	_	-	-		,	-	**	 	rate: 94 = 300 band
S	82F	D	0	5	DLYA	D	C	R		B			_		49 = 600 band
ER	1002	E		2		2		7		0	1	Y	2	<u> </u>	23 = 1200 band
P.		F	E	D		ا	,,	-	_	2	-	 	2		10 = 2400 band
COMPUTER SYSTEMS	82F	0	8	2				 - -	-			<u> </u>			06 = 4800 band
0	-4/	1	0	2	-	D	C	R		C			-		00 - 4100 Dans
NTEGRATED	-	2	C	2		7		Z	_	D	1	У	1		1 12:10 1
GR/		3		8 B		٦	10	2	_	ט		/_	-	\vdash	Loop 'till done.
ITE(<u> </u>	4	8	2		\vdash	<u> </u>	-		-		-	-	-	+-+
2		5	C	9		D	E	7	-			_		-	+
		6	۲	7		R	E	1				-	-	-	
		7	\vdash			\vdash		_				-		-	+
			-					-	-			-			Figure C. 9
	L	8	L					<u></u>			L	<u></u>		<u> </u>	Figure C-8

C.3.4 Subroutine OMSG

This subroutine will output a block of characters, whose starting address is specified in the HL register pair, via OCHR until an ETX (O3H or CNTL-C) character is encountered. (O2H, O1H and OOH also terminate transmission.)

- * Entry point 82F6
- * Arguments
 - Upon entry
 - HL register pair contain the output buffer start address
 - Upon return
 - A contains 03H (ETX character)
 - B, C and D contain 00H
 - E contains the last character output
 - HL contain the address of the last character output +1 (The address of the ETX byte.)
- * Subroutines used
 - OCHR at 82D0
- * Sample usage

21 00 83	LXI	H,OBUF	Load ouput buffer
			address
CD F6 82	CALL	OMSG	and output it

C.3.4.2 Subroutine OMSG Flowchart

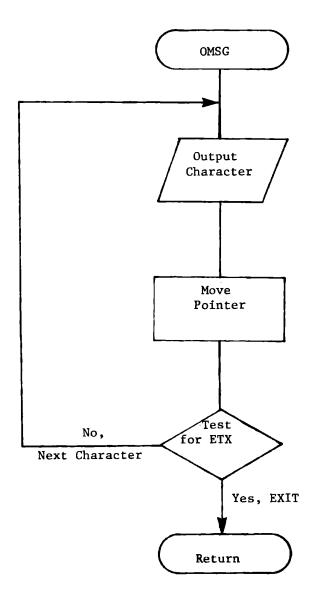


Figure C-9

	A D D	R	со	DE					0 1	M S	G					
	8	0														Output the buffer
		1													_	whose address is
		2													-	in H L min to
		3				Ì	-								_	in H, L pair to terminal until an
SHEET		4					-									ETX Character is
R		5			 -		-									encountered
CODING	82F	6	5	E	OMEG	м	0	V	-	Ε		М			-	Let next claracter
		7	/	5	OMSG	6	A	1	,	_	7	<u>/-1</u>	Н	R		see next craracte
ပ		8	ク	0		_	/.	_								
		9	8	2									_			
		Α	2	3	- -	\mathcal{I}	N	x		H						Marie Dais Tax
		В	9	3		S	1)	R	-	H		_				Consoli to 13
		С	F	0		R	P	B	-	-		-			_	Citil 100 0102 min3
Σ		D	C	3		5	М	P	<u> </u>	1	M	5	G		-	More pointer Compare to 03 Exit if 00, 01, 02, or 03 else go for next character
TRAINING SYSTEM		E	F	6		_	11-1	'	-			-				else go for nege
λS	82F		8	2			-							\vdash	-	character
N N		0	0	a		 	-			-			-			
N N		1	-			┝	 	<u></u>		-				-		
H.		2					-	-								
		3		-		-		-	-			_				
MICROCOMPUTER		4				-		-			-	-				
	-	5			<u> </u>		-	-	-							
ö		6		-				-	-							
CR		7				-	-	_	-	-					_	
Σ			-			├	-	-		_	_					
		9			<u> </u>	├	-	-	-							
		 А			<u> </u>		-	-	-	-			_			
ΝS			-		<u> </u>	-	-	-	 	-			-			
SYSTEMS		B 	-	-		_	-	_	-	-	_	_		-		
SΥS		<u>C</u>	-	-	 	<u> </u>	-	-		-				 -		
		D -	-		_	<u> </u>		-	-	-			_	H		
NTEGRATED COMPUTER		E F	-		<u> </u>	-		-	-	ļ						
)MP			1-	-		<u> </u>	-	-	<u> </u>						_	
\mathcal{C}		0	-			├-	-	-	-	-	-	-			_	
TEC		1	 			╢	-					-		-		
RA		2	<u> </u>		 	 	_	-		-	-	-	-		ļ	
ΓEG		3	-		<u> </u>	-		-	ļ	-	-		_		-	
Z		4	├ -	ļ	 	<u> </u>	\vdash	<u> </u>	-	_		<u> </u>	_	-		
		5	-	-		\vdash	 	-	<u> </u>	-	ļ	-	\vdash		<u> </u>	
		6	-	-	 	-	-	-	<u> </u>	-		<u> </u>		_	<u> </u>	
		7	├ -	-		 	├-	ļ	<u> </u>	-	_	-		_	<u> </u>	
	l	8	1 _		<u> </u>		<u> </u>	1_	<u> </u>							

C.3.5 Main Calling Program

This routine exercises the RS 232 system by calling the appopriate subroutine modules to output a message to the terminal, accept an input message (with echo), and output the inputted string. The program then repeats the procedure.

- * Entry point 8200H
- * Arguments, none
- * Subroutines used
 - IMSG at 8290 to prompt and then input a message
 - OMSG at 82F6 to output a character string message
- * Sample usage procedure
 - 1. Load the RS232 System from the Cassette Library
 - 2. Verify memory using the enclosed listings
 - 3. Press RST, RUN
 - 4. The display should go blank, and the terminal should display:

What is your name?

5. The system is now awaiting keyboard input. Type in your response followed by a Carraige Return character (Press the <u>RETURN</u> key).

APPENDIX C

6. The system will respond with

Hello (your response)

Hi There. I am ... etc.

7. The system will repeat the sequence.

* Data Tables

- An output buffer is provided at addresses 8300H-8362H with the "Hi There ..." message
- An output buffer is provided at addresses 83AOH 83A8H with the "Hello" character string.
- An input buffer is provided at addresses 8370H-838FH for the response character string.

NOTE: You may wish to experiment with your own messages and input and output sequences. There is ample space in the 512 bytes of RAM to code your own MAIN Calling program with your own messages. However, note that OMSG expects an ETX (03H) character as the terminating character in the output buffer.

If you have the 1K RAM option, you may wish to use the alternate IMSG routine provided at address 8000H with basic editing capabilities (Underscore for delete character, CTRL-X for delete line). The alternate IMSG is provided in Section C.4.

C.3.5.2 Main Flowchart

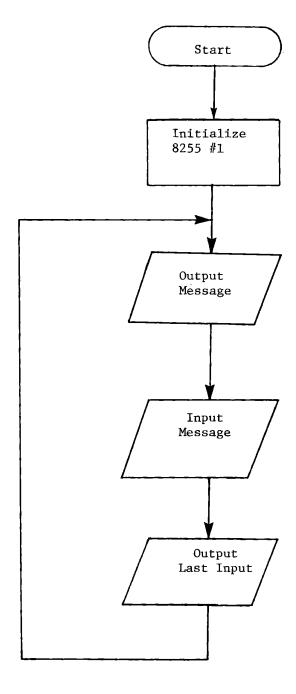


Figure C-11

RS232 INTERFACE EXERCISE PROGRAM - MAIN CALLING PROGRAM

-	A D D	R	со	DE												· · · · · · · · · · · · · · · · · · ·
	820	0	3	E	MAIN	M	V	I		A		9	2	4		Initialize 8255 #1
		1	9	2												8
		2	2	3		0	U	\mathcal{T}		C	N	7	/			
		3	0	7										L.		
SHEET	820	4	2		LOOP	7	X	I		\mathcal{H}		8	3	0	0	Output huffer
		5	0	0												Output buffer address
CODING		6	8	3												_
		7	C	\supset		C	A	L	L		0	Μ	S	G		
		8	F	6		Į.										
		9	8	2											L	
		Α	2	E		M	V	I		4		7	0	H		Input buffer
		В	7	0							,					address
		С	0	E		M	V	I		C		/	F			Input buffer
SYSTEM		D		F							,					martineme length -1
\Z		E	C	\mathcal{D}		C	A	L	L		P	M	P	\mathcal{T}		Prompt with " ?" and
		F	9	0					•							input a message
	821	0	8	Q												
TRAINING		1	A	E		M	V	I		1		A	0	H		"Dello"address
		2	A	0												
16		3	C	V		C	A	L	L		0	M	2	G		
MICROCOMPUTER		4	F	6												
S		5	8	2												
8		6	2	E		M	ν	I		L	,	7	0	Н	-	Name buffer address
$\frac{1}{2}$		7	7	0												//
		8	C	Ø		C	A	L	L		0	M	2	G		
		9	F	6												
[Α	8	2												
EMS		В	C	3		J	M	P		L	0	0	P			
ST		С	0	4												
S		D [']	8	2												
門		E														
INTEGRATED COMPUTER SYSTEMS		F														
8	8	0														
		1											_			
¥		2		L												
EGF		3	Ľ													
Z		4														
_		5														
		6										_			<u> </u>	
		7														
		8													<u> </u>	Figure C-12

	A D D	R	col	DE										
	8 30	0	0	A	OBUF	D	3		7	F				Line Feed
		1	0	D		D	B		C	\mathcal{R}				Line Feed Carriage Peturs
		2	4	8		D	B		H					
L		3	6	9		D	β		I					
SHEET		4	2	0		1.	:							
SH		5	5	4		1			T				7	
N.G		6	6	8		4			\mathcal{H}					
CODING		7	6	5		ľ	>		E		- 1	$\neg \dagger$		
ပ		8		2		 	Y		R			+	-	
		9	6	3		-	-		É	<u> </u>		- -	\top	
		A	2	E						<u> </u>			+	
		В	2	0					•	 				
		С	2	0		-	-						+-	
Σ		D	4	9					T	-		\dashv	+	
STE		 E	2	0		-			_	-		\dashv		
S	· · · · · · · · · · · · · · · · · · ·			1			-	-	A			-+	+	
N.G	83/		6	Ď		┢	-		M			+		
Z	05/	1	62	0		╁	 	-	ויין			\dashv	_	
MICROCOMPUTER TRAINING SYSTEM		2	7	9		-	-		V			\dashv		
ER.				ĭ		├			_			$\overline{}$	-	
Þ		4	6	F		⊢			0	ļ		-+	+	
Ä			7	5		├	-		U		\vdash	-	+	
ö		.6	7	2		├			R		\vdash	+	+	
ICR	<u> </u>	7	2	0		-		-	_			+		
Σ	<u> </u>		6	6		├			FR		\vdash	+		
		8		2		-	_		_	-			-	· · · · · · · · · · · · · · · · · · ·
		9	6	9	.	-			I	<u> </u>		\dashv	-	
SI			666	5		-			E	_		-+		
TEN		В	6	E	<u> </u>	 	_		N		\vdash		_	
.XS		С	6	4					D					
8		D	6		<u> </u>	-		ļ	L		\vdash			
T	601	<u>E</u>	7			D	В В		У			_	_	
ΜĀ	831	F	2	0		D	B			-				
INTEGRATED COMPUTER SYSTEMS	8	0	<u> </u>	<u> </u>		 	_		_		\sqcup			
TED	<u> </u>	1			<u> </u>	1					\vdash	_	_	
RA.		2				↓_	<u> </u>					_	_	
EG.		3					_				$\sqcup \downarrow$			
Z		4	<u>.</u>		<u> </u>	lacksquare				<u> </u>		_	_	
		5	<u> </u>	<u> </u>		lacksquare	_		_		\sqcup			
		6								<u> </u>	\sqcup			
		7				$oxed{oxed}$							\perp	
		8				<u>L</u>	ŀ	<u> </u>						Figure C-13a

	A D D R	co	DE	OUT	'PU	T E	UFI	ER	(0	con	tir	ıue	d)		
	8 32 0	6	3		D	В		1	CI						
	1	6	F		D	В			5						
	2	6	D			Ī			4		T				
L	3	7	0		-	-;			Ö	T	\dashv				
SHEET	4	7	5		¢				ク			_			
SH	5	7	4			¥			7	1	7	_		7	
ING	6	6	5				1	1	=						
CODING	7	7	2						F						
	8	2	0						1						
	9	7	2				1	1	R					Ĭ	
	Α	7	5					1	וני						
	В		E						V						
	С	6	EE9					-	V						
-EM	D	6	9					•	\mathbf{I}						
YST	E	6	E					1	/						
MICROCOMPUTER TRAINING SYSTEM	F	6	E 7						G						
Z	8 3 3 0	2	0]	
AAI	1	6							7	\perp					
F	2	6	E						V						
TEI	3	2													
MPU	4	2	0							\perp					
S	5	5	2						R			_			
8	6	5	3					;	5	_			_		
Σ	7	3	2						2	_	_		_		
	8	ઉ	3				\perp		3	\perp	\perp		_		
	9	3	2						3	\dashv	\downarrow				
S	Α							0	듸	\perp	4	_	_		
EM	В	2	0						_		_	_	_		
YST	С	4	9					_ -		\dashv	4		_		
RS	D	6	E					1	リア	_	_	_	_		
JTE		7	4						-+	\dashv	\dashv				
MP	833 F	6	52					- 4	R	_	\dashv	_	_		
္ပ	8 34 0		2				_	<u> </u> 1	2	_	_	_	_	_	
	1	6	6							\dashv	+		\dashv		
INTEGRATED COMPUTER SYSTEMS	2	6	7	ļ					A	\dashv		_			
TEG	3	6	3				\square		C	-	\dashv				
Z	4	6	5	<u> </u>	D	В			E	+	\dashv	\dashv			
	5	2	0			ļ	\vdash		\dashv	\dashv	-				
	6		-			-		\dashv	\dashv	+	\dashv				
	7		-			-				\dashv	\dashv	_			Figure C-13b
	8		1	Li			لبل								rigure C-13D

OUTPUT BUFFER (continued)

	A D D	R	co	DE				_	DOI						-u,	
	8	0														
		1										.				
		2														
		3									-					
EE		4														
CODING SHEET		5	\vdash			-	1									
Ŋ	834		_	_		7	D			P		_				
O	8 34	7	5	2		D	B			\mathcal{R}					_	
\mathcal{S}		8	'. -	E		-	1			R O		_			-	
			6				1		_							
		9		7		-	+		H	G			-	<u> </u>	-	
		Α	7	2			v		<u> </u>	2			_			
		В	6	/		!				A						
_		С	6	\triangleright						M						
TEN		D_	2	E A												
MICROCOMPUTER TRAINING SYSTEM		E		A						L	F			_	 	Line Feed Carriage Return
S. S.	×	F	0	D						C	8				İ	Carriage Leturn
Z	8 35	0	5	7						8						
₹		1	6	8						H						
Ë		2	6	/						A						
TEF		3		4						7						
Ē	· · · · · · · · · · · · · · · · · · ·	4	2	0												
Š		5	6	9				_	_	7						
õ		6		3						S						
딜		7	2	0		-	Ì	_	-		_					
~		8	2	9						V			<u> </u>			
		9	6	۶				_		0				_		
	 -	Α	I –		ļ —	-			-	Γ.				<u> </u>	-	
MS		В	7	2					_	U R					 	-
TE		C		_	 		\vdash			K			-			
SYS		D	2						-	11	_		-	-		
E P			6	E					-	N	_	_	-	-		
5		<u>E</u>	6	/					<u> </u>	A		_	<u> </u>	-		
MΡ		F	6	D	_				<u> </u>	M		_	├-	├	<u> </u>	
8	836		6	5		<u> </u>			<u> </u>	E		_		<u> </u>	<u> </u>	
	<u> </u>	1	2	0		<u> </u>			ļ				ļ_			
INTEGRATED COMPUTER SYSTEMS		2	0	3	<u> </u>	D	\mathcal{B}			E	\mathcal{T}	X	<u> </u>	_	_	End of buffer character
Ē		3	L.			<u>_</u> _					_		<u> </u>	_		character
F		4	L			L									L.	
_		5														
		6														
		7														
		8														Figure C-13c
																C-27

	A D D	R	co	DE	HELI	20	נטס	PU	T E	BUF:	FEF	ι,	INE	UT	В	FFER
[8 3 A	0	0	A	HELLO	D	B			L	F					HELLO message
ı		1	0	D		D	\mathcal{B}			C	R					HELLO message Linefeed, carriage return
ı		2	4	8		7)	B		//	H	"					
		3	6	3		Ď	B		11	E	"					
SHEET		4	6	\mathcal{C}		D	B		"	L	"					
		5	6	\mathcal{C}		ת	B		"	L	"					
S		6	6	K		7	B		11	0	11					
CODING		7	2	0		D	B			S	Δ	A	C	E		
		8	0	3		7	B			E	T	X		_		End and builder
		9	Ť				7									End of buffer
		A									7					
		В	7		-	7				7				7		
ł		C							/					/		
EN		D /							1				7			
SYSTEM		E											/			
S S		F			/			7				/				
TRAINING	8 37	0			IBUF	D	S		3	2	/+					Reserve 32 leyter of RAM for 3/ input characters + EXT > 32 leyter
Ā	839	0														of RAM day 3/ input
		2														Charactery + EXT → 32
削		3														britis
MICROCOMPUTER		4														
Ś		5														
Š		6														
N N		7														
		8														
		9														
,,		A														
SYSTEMS		В														
/ST		С														
		D														
TEI		E														
ΛPU		F														
CO	8	0														
ED		1														
3AT		2									,					
INTEGRATED COMPUTER		3														
N		4						L					<u></u>			
		5						_		L	<u> </u>		<u> </u>	_	<u> </u>	
		6				<u> </u>			_	_	_		L			
		7	L				_	_	L_					_	_	
		8							L_							Figure C-13d

C.4 ALTERNATE IMSG PROGRAM

This subroutine is identical to the IMSG routine with the following exceptions:

- 1. Entry point is 8000H
- 2. There is no PMPT entry

character entered

4. A CTRL-X (18H) character will delete the entire input line. The flowchart is shown in Figure C-14 and the code is in Figure C-15.

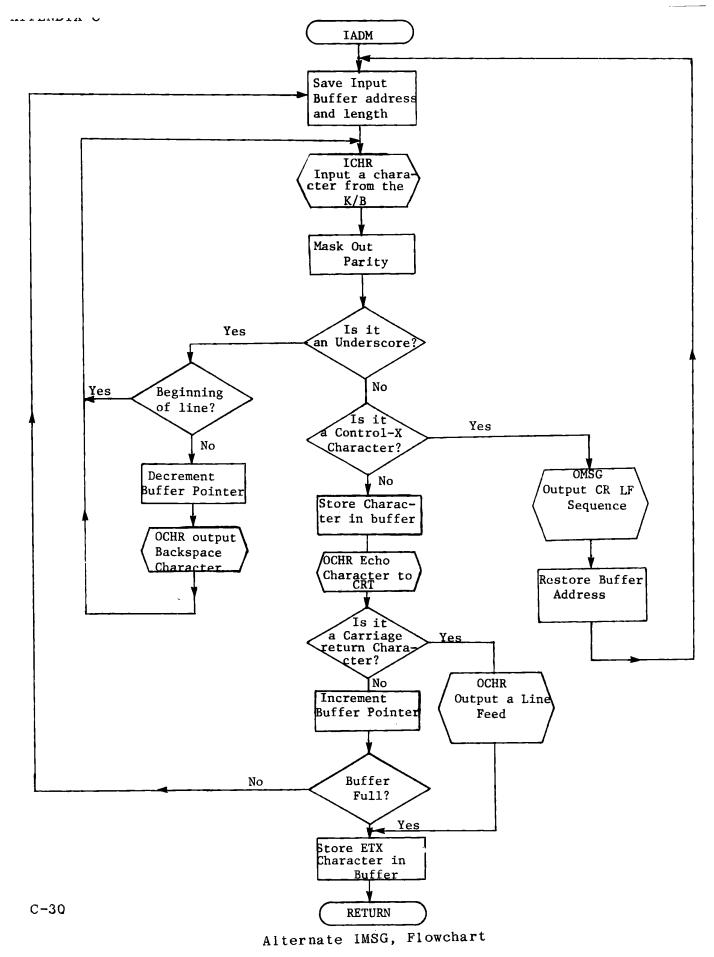


Figure C-14

ALTERNATE IMSG SUBROUTINE

Be o I I I I I I I I I I I I I I I I I I		A D D R	coi	DE	i	ALT	ERI	rAI.	'E	IMS	G	SUE	RO	UTI	NE
ADM termihable boroning undersore as a blockspace character and CT2-X as a delete line character. I gent be character. I gent be character. I gent be character. I gent be character. I gent be character. B character. I gent be character. B character. I gent be character. B charac	ſ											•			Input a) string from
A CALL CALLED BY SEE COLUMN TO THE STATE OF	ţ	1				-									
A B D O O E S TADM P US H H Law Inflorables length in C. B C D Les C A L L T C H R Shit a character B B C C A D Les C A L L T C H R Shit a character B B C C A D Les C A L L T C H R Shit a character B B C C A D Les C A L L T C H R Shit a character B B C C A D D Les C A L L T C H R Shit a character B B C C A D D Les C A L L T C H R Shit a character B B C C C C C C C C C C C C C C C C C	ĺ	2					Ī								
Character and CTL-X as a delete line character and CTL-X as a delete line character and CTL-X as a delete line character and CTL-X as a delete line character and CTL-X as a delete line character and CTL-X as a delete line character and CTL-X buffer address H pair and maximum buffer length in C. buffer address H pair and maximum buffer length in C. buffer length a C. buffer length a C. buffer length a C. buffer length a C. buffer length a C. buffer length a C. buffer length a C. buffer length character and CTL-X as a delete line maximum buffer length in C. buffer length a C. buffer length buffer length buffer length a C. buffer length buffer		3					1								
STATE OF THE PROPERTY OF THE STATE OF THE PROPERTY OF THE PROP	ωl	4													
Character. Input buffer address B A B C D A B C D A B C D D		5													
NOT THE COLOR OF T	욁	6													
NOT THE COLOR OF T	밁	7													
NATIONAL BEAUTION OF THE PORT OF THE PROPERTY OF THE PROPERTY OF THE PORT OF T		8			·										4 1/ 2011 2.1
STAND STAN		9													maximum buddes
STAND STAN		Α													length in C.
BUNNEY B 00 0 E 5 IADM P US H H Save buffwaddress 1 4 1 M 0 V B, C Save buffw length 800 2 C 5 Le1 P US H B 800 3 C D Le2 C A L L T C H R Star character 4 B 0 5 8 2 N T 7 F Nask out parity bit 7 7 F Nask out parity bit 8 F E C P I " - " Seat for underscore 9 5 F A C A J Z B K S P So Save buffwaddress A C A J Z B K S P So Save buffwaddress 8 F E C P I C A N ^X: Seat for CN71-X E / P J D F L N Yes, go debte line 3 0 / 0 44 J D Z D F L N Yes, go debte line 5 6 7 8 Figure C-15a		В													
SWELL SWEL		. С													
SWELL SWEL	EM	D													
SWELL SWEL	YST	E													
		F													
	Ž	8 00 0	E	5	TADM	\mathcal{P}	\mathcal{U}	S	H		H				Save buffer address
	Ž		4	7		M				B		C			
		8002	C	5	LPI	P	U				\mathcal{B}				11
A	TEF		C	D	LP2	C	A	L	L		I	C	H	R	It a character
8 F E C P I " - " Yest for underscore 9 5 F	IPU.		_	0											
8 F E C P I " - " Yest for underscore 9 5 F	Sol	5	8	2											
8 F E C P I " - " Yest for underscore 9 5 F	RÕ	6	E	6		A	N	I		7	F				Mask out mrity lit
8 F E C P I " - " Yest for underscore 9 5 F	MIC	7	7	F											7
9 5 F A C A TZ B K S P		8	F	E		C	0	I		"	_	11			Yest Sorunderscore
B 2 C C P O C P I C A N ^X: Sest for CN72-X E / B		9	5	F											
B 2 C C P O C P I C A N ^X: Sest for CN72-X E / B		Α	C	A		2	Ŋ				B	K	S	ρ	lo backspace
E / 8	EMS	В	2	C											
E / 8	/ST	С	8	0											
1 80 2 3 3 4 5 6 7 7 8 Figure C-15a		D	F	E		C	ρ	I		C	A	N			"X: Lest for CNTL-X
1 80 2 3 3 4 5 6 7 7 8 Figure C-15a	旦	E	/	8											
1 80 2 3 3 4 5 6 7 7 8 Figure C-15a	AP.	F	\mathcal{C}			3	N	Z			0	E	1	N	Yes; go delete line
5 6 7 7 Figure C-15a	00	30/0	4	4											<i>y.</i> • <i>y</i>
5 6 7 7 Figure C-15a	B	1	8	0											
5 6 7 7 Figure C-15a	3AT	2				L									
5 6 7 7 Figure C-15a	EGF	3													
5 6 7 7 Figure C-15a	IN	4													
7 8 Figure C-15a	_	5													
8 Figure C-15a		6		_					_						·
		7													
C-31		. 8	L												

	A D D	R	со	DE		AL'	ĽER	NA'.	ľΕ	IMS	3G	(C	ont	ını	ued) C - 33
	8	0											-			
		1		_			_				<u> </u>					
	801	2	7	7		М	0	٧/		Μ		A			\vdash	Store the character
	001	3	3	V			0			E	,	Á			-	
Ë		4)(\mathcal{L}		(A		L	٢	0	c	Ш	R		Echo character
SHE	· · ·	5	1	7		_	Т.	<u> </u>	<u>ب</u>		\mathcal{O}	۷	П	K	<u> </u>	
		6	3	5						-	-			-	-	to CRT
CODING			8	2		1]		0		-			 	
윙		7		/		P	0	P	_	B			-		<u> </u>	
	301	8	Ž	23	L		0			4		E				
	801	9		E			P	エ		C	Ŕ	_				Lecharacter a
		A	0	S			-			_			_		├	carr ge return?
j		В	0	6		M	V	工		E	,	L	F	<u> </u>		
-		<u>C</u>	0	H			L.					_	_		<u> </u>	
SYSTEM		D	<u>C</u>	C		C	Z				0	C	1	R	L_	Cheracter and
ΥS		E	D	0										_		character and
		F	8	2												
TRAINING	802	0	C	A		2	Z					E	X	工	T	then exit
Α		1	2	8												
		2	8	0												
MICROCOMPUTER		3	2	જ	•	I	N	X		H						March Idas mitter
		4	0	7		D	(R		C						Move buffer pointer Decrement character country
NO.		5	0	2		_	N	Z		-	1	P	1			and an aid most
00		6	0	2		-	 '`			-	-	•	-			character, if buffer
힏		7	8	0		<u> </u>							-	-	-	character is truffer
Σ	000			_	277	N/1	1	—		M		^	2	-		not full. Store EXT (03H)
	802	9	3	93	EXIT	M	V	I	-	141	,	0	3	-	-	
		A	3	<u> </u>		P	_			D	-	-		-	-	character at end of
JIS	· 		19	2	 -	⊢ '	0	P	-	Ψ	┝	<u> </u>				mag. and return
SYSTEMS		В	2	9		R	E	1		2		-		-		* * *
3YS	802	С	<u>C</u>	<u>/</u>	BKSP	P	0			B		-	-	├	-	Backspace / Character
		D	14	1	<u> </u>	P	0	P		D	_	-		_		Ketrieveluffer address
COMPUTER		E	2	5	 	P	U	S	H		D	_	_	<u> </u>	ļ	
MPI		F-	7	B		M	0	V	ļ	A		E	 	-	-	
00	3 03	0	B	D		C	M	P		1_	1_	_	_		_	already front line,
ED		1	C	A		2	Z				L	P	1			ignore backspace
₹AT		2	0	2												,
EGF		3	8	0	,					_						
INTEGRATED		4														
_		5														
ı		6														
	-	7														
		8	1													Figure C-15b

					A	LTE	!RN	አ ጥፑ	а т	MSC		(cc	n+	inı	ıed	1
	A D D	R	co	DE.			1		_		, 	, –	,			,
	8	0	<u> </u>			<u> </u>				<u> </u>	<u> </u>	_			<u> </u>	
			 _	_		<u>L</u>			_		<u> </u>				<u> </u>	
		2	_						<u> </u>							
-		3														
SHEET	803	4	2	\mathcal{B}		D	C	X		H						Nove buffer printer
		5	0	C		I	N	R		C						Nove buffer pointer back one character
Z		6	C	5		P	U	S	H		B					
CODING		7	1	E		M	ν	I		E	T.	"	^	"		Dutant a "mare
O		8	0	8			Ť				/					Output a "more
	803	9	C	D		C	A	L	1		0	C	H	R		cursor left" ADM-3 character
		Α	Ď	0									-		†	Harris Catalansee
		В	8	2				_					-		 	
		С	0	0		N	0	P	 		-		-	-	† -	These NOP's leave room
Σ		D	0	0	-	N	0	P	<u> </u>	1	-	-			 	
SYSTEM			0	6		N	0	0		\vdash	├	-	-		╁╌	to include code to
		 F	0	5		Ť.	0	D	-		-	-	-	-	1	output a backslash
TRAINING	804	0	0	0	<u></u>	N	-	P	-	-		-	-	-	-	deleti character
Z	004	<u> </u>	0	3		2	M	10 10			L	D	2	_	-	sequence (like D.E.C.)
TR/		<u>'</u>	 ~			ا-،	1	1-	-		<u> </u>	10	~		ļ .	
E.R.			0	3		-	-		-		-	-	-		ļ .	Soget next character.
5	(2 x · /	3	8	0	 	ļ. :	.,	_	-	 	ļ .	_	_	_	<u> </u>	
MICROCOMPUT	804		2	/	DETN	<u> </u>	X	I		14	, , .	C	रि	L	۲	Delete entire line;
00		5	5	0		-			-	-		-				
CR	ļ	6	8	0			_		ļ.—					_	-	
⋛		7	C	D		C	A	<u>_</u>	1		0	M	5	G	ľ	Output Carrige Return - Line Feed sequence.
		8	F	6						<u> </u>					_	Line Leed sequence.
		. 9	8	S		L				L						Line Feed sequence.
S		Α	\mathcal{C}	/		P	0	ρ		\mathcal{B}					L	Restore buffer address
EN.		В	4	8		M	0	V		C		B				and character
SYSTEMS		С	E	/	· .	P	0	P		\mathcal{H}						Counter.
		D	C	3		2	M	ρ		I	A	D	M			Lo back to beginning
旦上		E	0	0												The state of the s
AP.		F	8	0												
8	8 05	0	0		CRLF	D	S				C	2				CR, LF buffer
<u>.</u>		1	0	A		$\overline{\mathcal{D}}$	S	_	-		L	F				The state of the s
AT!		2	0	3		D	S		_		E	T	X		_	
GR		3					_	_	-		_	<u> </u>		-	-	
NTEGRATED COMPUTER	·	4				$\vdash \dashv$		_				-			-	
Ξ		5	\vdash		_	\vdash			-							
		6			 -						-				-	
		7	\vdash			\vdash		-		 	-	-			-	

C-33

Figure C-15c

ALEBNUIA C

This page intentionally left blank.

MICROCOMPUTER INTERFACING WORKBOOK

APPENDIX D

TELETYPE INTERFACE SYSTEM

This page intentionally left blank.

APPENDIX D

TELETYPE INTERFACE SYSTEM

This Appendix describes a procedure for interfacing a teletype to the ITS utilizing an 11 bit 110 baud code.

D.1 SPECIAL EQUIPMENT REQUIRED:

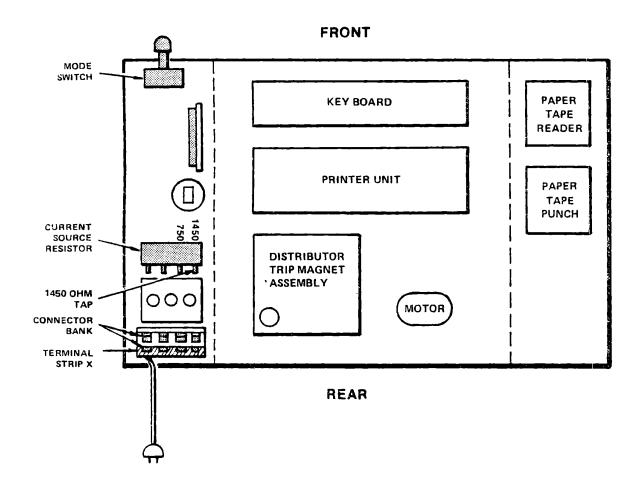
A source of -12 volts 1 volt @ 50 mA. (power supply or batteries)

D.2 MODIFICATIONS TO TELETYPE:

Inerfacing the teletype requires the following modifications to the teletype unit itself:

- 1. 20 milliamp current loop option
- 2. Half duplex option

The 20 milliamp current loop and the half-duplex connections are options available on most teletype units. Check your unit before proceeding.



Top View - Teletype with Housing Removed
Figure D-1

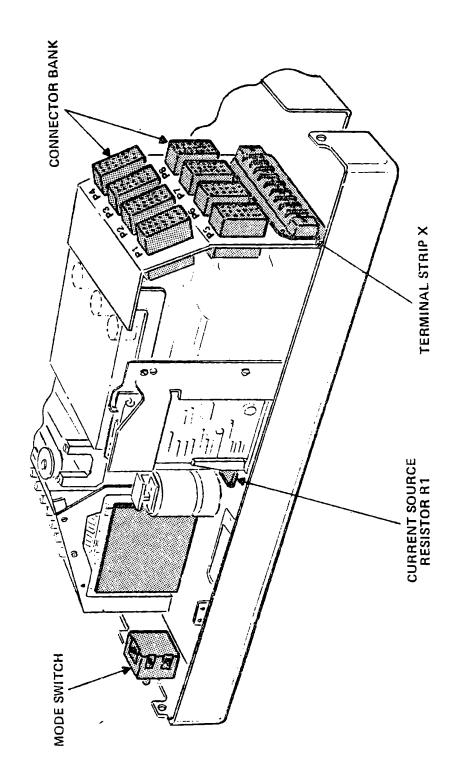
D.2.1 Removing the TTY Housing

It is necessary to remove the TTY housing to inspect or modify the TTY options.

- 1. Unplug the TTY from the power source.
- 2. Remove the roll of TTY printer paper from its cradle.
- 3. Remove the manual paper feed knob by pulling firmly.
- 4. Remove the mode select knob located on the right front by pulling firmly.
- 5. Remove the metal trim panel behind the mode select knob by prying downward.
- 6. Remove the 4 screws under the metal trim panel.
- 7. Remove the screw on the left side of the paper tape reader housing.
- 8. Remove the four knurled knobs along the lower rear edge of the housing.
- 9. Lift upward on the housing to remove, being careful of the controls on the paper tape reader as they clear their openings in the housing.

D.2.2 Locations of Modifications

See Figures D-1 and D-2 to locate the terminal strip X. Terminal strip X is located at the bottom of the rear of the teletype.



Side View - Teletype with Housing Removed

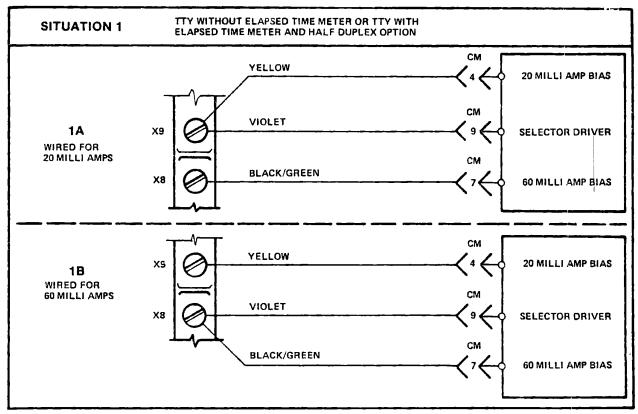
Figure D-2

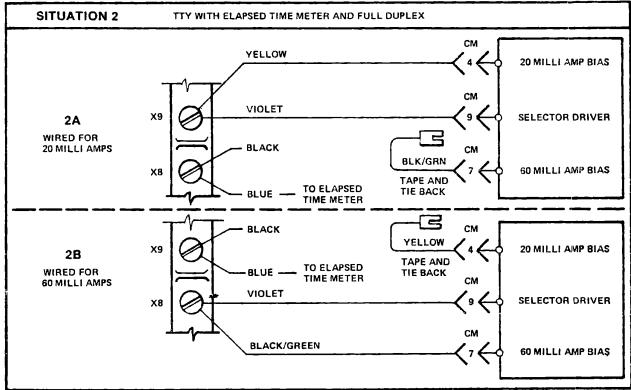
D.2.3 Current Loop Option

The TTY send and receive current loop can be optionally selected to work from either 20 milliamp or 60 milliamp. When the selection is made both the internal current source and the selector drive current bias must be modified to be compatible.

D.2.3.1 Internal Current Source

The internal current source is set to 20 milliams by putting the blue wire on the 1450 ohm tap of power resistor R1 located on the right side of the TTY.





Current Loop Option

Figure D-3

D.2.3.2 Selector Drive Current Bias

The selector drive current bias is set to 20 milliams by optional wiring on terminal strip X located below the connector bank in the right rear corner of the TTY. In making this change various wiring configurations may be encountered as shown in Figure D-3, depending on whether the unit has an elapsed time meter.

TTY Without Elapsed Time Meter

A TTY without an elapsed time meter may be wired either as 1A or 1B of Figure D-3. To modify for 20 milliamp:

If wired as 1A:

Do nothing; this is the correct connection for 20 milliamp without an elapsed time meter.

If wired as 1B:

Remove the violet wire from terminal X8 and move it to terminal X9 with the yellow wire.

APPENDIX D

TTY with Elapsed Time Meter

A TTY with an elapsed time meter may be wired as 1A, 1B, 2A, or 2B To modify for 20 milliamp:

If wired as 1A:

Remove the black/green wire from X8, tape the exposed end and tie-back into the wire bundle. Locate a black wire and a blue wire on teminal X5. Move both wires from X5 to terminal X8.

If wired as 1B:

Remove the violet wire from X8 and move it to X9. Remove the black/green wire from X8; tape the exposed end and tie-back into the wire bundle. Locate a black wire and a blue wire connected on terminal X5. Move both wires from X5 to X8.

If wired as 2A:

Do nothing; this is correct connection for 20 milliamp with an elapsed time meter.

If wired as 2B:

Remove the black wire and blue wire from X9. Remove the violet wire and black/green wire from X8. Connect the black wire and blue wire to X8. Connect the violet wire to X9. Locate the yellow wire taped back in the wire bundle. Connect the yellow wire to X9. Tape the exposed end of the black/green wire and tie-back into wire bundle.

APPENDIX D

D.2.4 Full Duplex Option

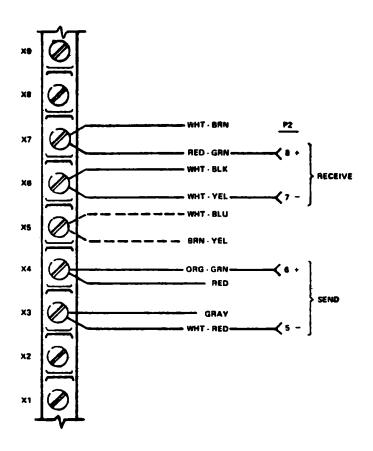
The full duplex option is wired into the TTY on terminal strip X located below the connector bank in the right rear corner of the unit.

If the TTY is wired for full-duplex, terminal strip X should appear as in Figure D-4.

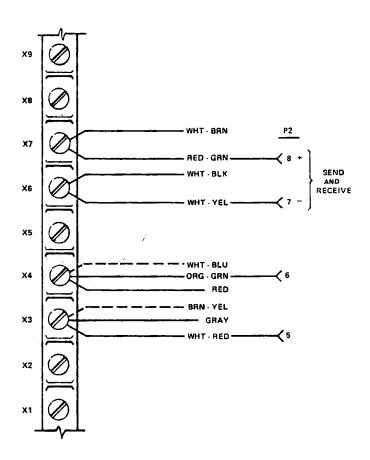
If the TTY is wired for half-duplex, terminal strip X should appear as in Figure D-5.

D.2.4.1 To Convert from Full-Duplex to Half-Duplex:

- 1. Confirm that screw lug X5 has a white/blue wire and a brown-yellow wire connected. If there is a BLACK wire and a BLUE wire on X5 an elapsed time meter is installed. Refer to the current LOOP option for instructions on moving the black wire and blue wire from X5 to X8.
- 2. Move the white-blue wire from screw lug X5 to X4.
- 3. Move the brown-yellow wire from screw lug X5 to X3.

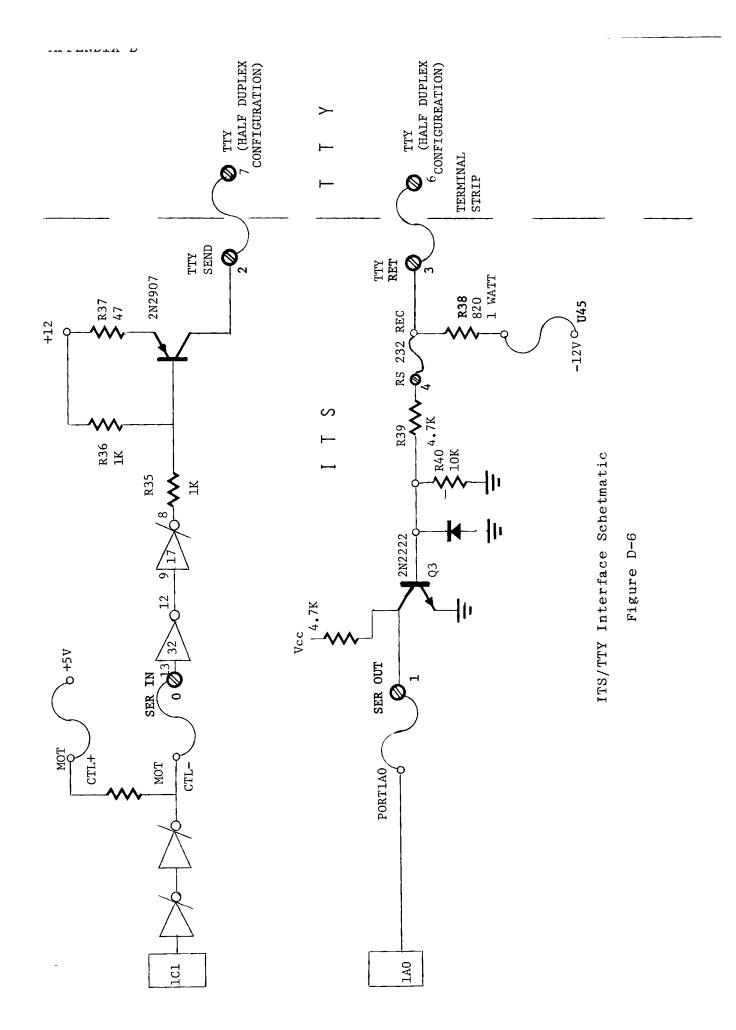


TTY Full-Duplex Option
Figure D-4



TTY Half-Duplex Option
Figure D-5

This page intentionally left blank.



D.3 CONNECTING THE HALF-DUPLEX TELETYPE TO THE ITS

Figure D-6 shows the schematic of the connection between the 8255 ports on the ITS and the teletype. The following wiring will implement this schematic:

- 1. Connect a wire between terminal 7 on the TTY rear terminal strip and "TTY SEND" (screw #2).
- 2. Connect a wire between terminal 6 on the TTY rear terminal strip and "TTY RET" (screw #3).
- 3. Connect a jumper wire between "TTY RET" (screw #3) and "RS 232 REC" (screw #4).
- 4. Connect a wire between "SER OUT" (screw #1) and Port
 1A0 (pin 16 on location U18).
- 5. Connect a wire between "SER IN" (screw #0) and "MOT CTL-" on the left side of the board.
- 6. Connect a wire between "MOT CTL+" and 5V on the left side of the board.
- 7. Finally, connect -12 volts (from pin 45 of the ribbon cable connection) to R38.

APPENDIX D

D.4 POWERING UP THE SYSTEM

Power up the MTS/ITS first. Depress "RST" on the MTS. Turn on the teletype by switching the front swich to "Line".

D.5 RUNNING THE TELETYPE PROGRAM

Enter the program of Figure D-7. Start the program by typing "RST", "RUN". A message should be typed on the teletype printer. Respond to the question by typing your name followed by the "RETURN" key. It will respond by typing a second mesage followed by the response you typed in.

D.5.1 Description of Program Modules

The program listing contans the following modules:

START: Main routine (8200 - 822A)

OMSG: TTY output message routine (8250 - 825B)

ICHR: TTY character input routine (82B0 - 82CF)

OCHR: TTY character output routine (82D0 - 82FF)

MESSAGES: Buffers for output and input (8300 - 837A)

TELETYPE PROGRAM - MAIN ROUTINE

А	. D D	R	co	DE	TELI	ETY	PE	PR	OG1	RAM	_	MA	IN	RO	UT:	INE
8	20	0	3	E	START	M	V	I		A	,	9	2			Initializa 8255#1
		1	9	2												Initialize 8255#1 Ain Cout
		2	\mathcal{D}	3		0	υ	7		0	7					
		3	0	7			-	•								
		4	2	7	LOOPI	1	X	I	_	H		8	3	0	2	Int in the same
SHEET		5		$\frac{1}{2}$		_	^	_		77	/	0)		0	1º message pointer
			0	2					_	-		\vdash				
SODING		6	8	3	· ·	4						4.0	-	(
8 _		7	C	∇		C	A	<u></u>	L	<u> </u>	0	M	2	G		
_		8	5	0												
<u> </u>		9	8	2												
		Α	2	/		<u>L</u>	X	\mathcal{I}		14		8	3	5	4	Input buffer pointer
L		В	5	4												
		С	8	3	_											
EN _		D	C	D	LOOP	C	A	L	L		I	C	H	रि		Input character
SYSTEM		E	B	0												
		F	8	2												
TRAINING	21	0	F	E		\overline{c}	P	I	_	8	D	Н	-	\vdash		Lait CR?
3	<u> </u>		8	D			-	-	 - -	0	ب	/ <u>'</u>	-		 	Some CR:
治		2	C	A		7	7	-	S	K	I	p	-		 	· , ,
			1	-		2	Z	├	9	_	<u> </u>	1-			-	yes; goon.
5 -		3	6	A		-	<u> </u>	 -	<u> </u>	<u> </u>		-	-		-	
MICROCOMPUTER		4	8	2	-	_	 	H-	 -	. ,	Ĺ	_	ļ. <u> </u>	-	<u> </u>	
잃느		<u> </u>	7	7		M	_	V		M	7	A	<u> </u>	-	-	To; stre character
띩		6	2	3		H	N	X		H	Ĺ		_		_	Increment buffer pointer
ĕĻ		7	C	3		<u>J</u>	M	P		L	0	0	P			Let next character
L		8	0	D												
		9	8	2												
.,		A	A	F	SKIP	X	R	A		A						Luffer 2 message pointer
XI.		В	7	7		M		V		M		A				ludder
TS		C	2	1		1	V	T		H	_	8	3	4	6	2nd Junay sainter
<u></u>		D	4	6							,	_	-			The state of the s
		E	8	3				1.								
닭		F	C	D		C	A	7	1		0	M	7	G	\vdash	0 + + 224
INTEGRATED COMPUTER SYSTEMS	22	0	5	0	 	۲	' 		-	 	<u>ں</u>	/''	٦	٧	\vdash	Output 2nd message
바	07 0	1	8	2	 	<u> </u>	-	-		\vdash	-	-	-	 	-	
ᆰ	-	2	10	13	 	-	\vdash	_	-	-	-	\vdash	 	-	\vdash	
AR-			-		<u> </u>	├	-	\vdash	-	-	-		-	<u> </u>	<u> </u>	
빤		3	-	<u> </u>	ļ	<u> </u>	ļ	-	-	-	_	-	-	-	-	
Ż _		4	<u> </u>	<u> </u>		 					į.	<u> </u>		ļ	ļ	
L		5	┞	<u> </u>	L		-			$oxed{ightarrow}$	<u> </u>	_	<u> </u>	<u> </u>		
		6	$oxed{oxed}$	L_	ļ		_		1_	_	L_		_			
		7	_			_		_	_	L	L				L_	
		8	L													Figure D-7a

	A D D	_ R	со	DE	TTY	ou	TP	UT	ME	SSA	GE.	RC	UT.	INE	Ē	
	8	0														
		1				_									-	
-	822	2	2	1		L	X	I		H		8	3	7	0	3 messagespointer
		3	7	0		_=			-	•	•	U)	-		- missagegiocala
SHEET		4	8	3								_		<u> </u>	_	
		5	C	D		0	A	2	4	-	\	Μ	0	G		D + + 2nd
CODING		6	5	5		_	//		-	-		/ <u>-</u>	Ç	<u> </u>	 	Output 3rd message
		7	-	2							_					message
ျ		8	C	3		7	Μ	a	-		L	0		P	1	
ı		9	0	4		J	170	_	 			0	Ų	1-	-	
1			8	2					├─		_	-	_		-	
1			0	$\tilde{}$			-			-			_		<u> </u>	
ł			\vdash			_	-	-		-		_	-		-	
Σ				-				-		-			-	-		
SYSTEM						\wedge	()	7	P	15	7		M		-	5065 5055
	<u> </u>		\vdash	-	-	O			1		1	_	<u>/*\</u>	C	13	SAGE SUBROUTINE
2	8 25	0	7	E		4.4	7	17				14				W / . / -
TRAINING	• & U	1	 			MC	<i>Ο</i>	V I		A	6	M	_		-	Let character?
T.		2	1	E		L	7	<u>+</u>	-	0	U			-		Soit yero!
			0	0		0	-7								-	
5		3		8		R	Z	1.7				^				Exit if end No, send character
MICROCOMPUTER		4	5	<u> </u>	-		2	ν	 	E	1	A				To, send character
읽		5	0	\mathcal{D}		C	A	L	1		Ó	Ċ	Н	\mathcal{K}		
CR		6	S	9											-	
Ξ		7	8	2										_		
		8	C	3	_	7	Μ	P			0	Μ	5	G		Loop
			5											_	ļ	
2		Α	8	2					ļ							
SYSTEMS		В						L	ļ							
ΥS		С														
RS		D	_											_		
븨		E														
MP		F													ļ	
NTEGRATED COMPUTER	8	0							<u> </u>							
밁	 	1														
Ϋ́		2						_								
EGF		3														
Ρ		4														
_[5														
Į		6														
		7														
		8														Figure D-7b

TELETYPE CHARACTER INPUT ROUTINE, BYTE IS PLACED

8 2B 0 CD CAR CALL TBIT Input Port 180 1 FB 2 P2 3 DA 5 C TCHR AND Hope the second of	_	A D D	R	co	DE]	IN REC	SIS	TEI	RS	AC. A 1	AND	E,	R	EQ	JIR	ES	OCHR ROUTINE
A B C D C A L L T B / T Start bit detected the start by t	Γ	8 2B	0	C											\mathcal{F}		
Start bit detected Start b				F													
A B C D C A L L T B I T Start bit still there B C D C A L L T B I T Start bit still there C F B D P 2 E D A J C T C H R No; keep trying No; keep trying F F B No J C J C A L L D L Y I S F F B D A J C T C H R No; keep trying I feet of the sep trying No J C J C J C J C J C J C J C J C J C J			2	P													Start lit detected?
4 80 5 82 6 0 E M VI C 0 4 Cles; wait 1/2 bit 7 0 4 D L Y / 9 F 0 A 82 B CD CALL TB / T Start bit still there C FB D A J C T C H R Project again F FB 1 / E M VI E 80 Initialize 1 / E M VI E 80 Initialize 2 80 B CD CALL D L Y White for next bit 1 / E M VI E 80 Initialize 2 80 G C D C A L L D L Y White for next bit 4 E E D C C A L L TB / T Input Port 1A0 F B M D V A F E Stet previous bits A / E R A R			3		_		7	C			T		H	P			
S	딺		4					,	_		_		<u> </u>				110, scipsiging
B CD CALL TBIT Start for next list of the counter o	돲		5								-						
B CD CALL DLY / 9 FO A 82 B CD CALL TB / T Start but still there C FB D P2 E DA JC TCHR MVT E, 80 Initialize 1 / E MVT E, 80 Initialize 2 80 6 CD CALL DLY What for next but 8 8 2 9 7 B MOV A, E Way carry into lit. 7	왕			0			1	12	T		0			11			71
B CD CALL DLY / 9 FO A 82 B CD CALL TB / T Start but still there C FB D P2 E DA JC TCHR MVT E, 80 Initialize 1 / E MVT E, 80 Initialize 2 80 6 CD CALL DLY What for next but 8 8 2 9 7 B MOV A, E Way carry into lit. 7	핡			10			/*/	ν,	<u></u>		_	,		7	Н	\vdash	ges, wait 12 bet
B CD CALL TBIT Start Let still there C FB D P 2 E DA J C T CHR P So; try again F FB B 2 C O P 2 1 I E MVI E, 80 Initialize 2 80 ILP CALL D LY Wait for next bit 4 EE 5 8 2 6 CD CALL TBIT There some still 8 8 2 C 9 7 B MOV A F Let previous bits A I E RAR	ᇬ						~	1	,	\ <u>\</u>		7	,	₩	7		<u> </u>
B CD CALL TBIT Start bit still there C FB D P2 E DA JC TCHR To; try again F FB 1 IE MVIE, 80 Initialize 2 80 Initialize 3 CD ILP CALL DLY Whit for next bit 4 EE 5 P2 6 CD CALL TBIT JAput Port 1A0 7 FB 8 8 2 9 7B MOV A, F Very case into lite	ŀ			ے	-		ب	77	<u>_</u>	_		D	4	 	-		333
B C D C A L L T B / T Start bit still there C F B D P 2 E D A J C T C H R To; try again F F B 8 2 C 0 P 2 1 / E M V T E, 8 D Initialize 2 8 D LLP C A L L D L Y Whit for next list 4 E E 5 P 2 6 C D C A L L T B / T Input Port IAD 7 F B 8 P 2 9 7 B M D V A, F Set previous bits A / E P A R	- }			10			-		-	-	-	-		_			
C FB D P2 E DA JC ICHR No; try again F FB 8 2 C 0 P2 1 / E MVI E, 80 Initialize 2 80 Initialize 2 80 Initialize 4 EE 5 P2 6 CD CALL TB/T Japet Port 1A0 7 FB 8 P2 9 7B MOV A, F Set previous bits A / E PAR	-							_	ŀ.	<u> </u>	-	-	0	-	-		1
E DA JC ICHR Morrison Into Part 120 B 2 C O P 2 1 / E MVI E, 80 Initialize 2 80 Initialize 3 CD ILP CALL DLY Wait for next list 4 EE 6 CD CALL TB/T Input Port 1A0 7 FB 8 8 2 9 7 B MOV A, E Set previous lists A / E P AR	-			_	_		C.	/+	4	4	-	_	0	_	<u> </u>	\vdash	Start bit still there?
BY B 2 C 0 B 2	-			<u>-</u>	<u> </u>		 		<u> </u>				<u> </u>	_	<u> </u>		
BY B 2 C 0 B 2	削			8			_			L_	_		_		<u> </u>	Ш	
BY B 2 C 0 B 2	S.		E	0	_		J	C		工	C	H	R		_	Щ	Y/o; try again
BE S O SILP CALL DLY Whit for next list 4 EE 5 8 2 6 CD CALL TBIT Input Port IAO 8 8 2 9 7 B MOV A E Set previous lists A LE RAR			F	F	_				_	<u> </u>	<u> </u>				_	Ш	
BE S O SILP CALL DLY Whit for next list 4 EE 5 8 2 6 CD CALL TBIT Input Port IAO 8 8 2 9 7 B MOV A E Set previous lists A LE RAR	ź	8 2 C	0	8		_			L								
BE S O SILP CALL DLY Whit for next list 4 EE 5 8 2 6 CD CALL TBIT Input Port IAO 8 8 2 9 7 B MOV A E Set previous lists A LE RAR	Ž.		1	/	E		M	V	I		E		8	0			Initialize
5 82 6 CD CALL TBIT Input Port IAO 7 FB 8 82 9 7B MOVA F BUT Servious lists A LE RAR MOV CARRY into 12	-		2	8	0												bit counter
5 82 6 CD CALL TBIT Input Port IAO 7 FB 8 82 9 7B MOVA F BUT Servious lists A LE RAR MOV CARRY into 12	핃	_	3	C	D	ILP	C	A	7	<u> </u>		\mathcal{D}	2	У			Wait for next list
5 82 6 CD CALL TBIT Input Port IAO 7 FB 8 82 9 7B MOVA F BUT Servious lists A LE RAR MOV CARRY into 12	₹[4	E	E												
8 8 2 9 MOVA E Set previous lists A LE RAR MOVACANA int. 7	ģ		5	8	2												
8 8 2 9 MOVA E Set previous lists A LE RAR MOVACANA int. 7	Š		6				C	A	7	L		ア	B	1	7		Junet Port 1AD
8 8 2 9 MOV A E Set previous lists A LE RAR MOV CANALIST LIST. 7	읡		7		_	-											<u> </u>
9 7B MOVA E Get previous lite			8	R	_							_			1		
A /F RAR May carry int hit?	ŀ		9		_		M	1	1/	 	A	 	E	 			Mat alassiana) lital
B 5 F MOV E, A Save present bits C D 2 JNC ILP Set next bit until CV shows we got E 9 2 all 8	ŀ		A	/			D	A	P	<u> </u>	1)	_		\vdash		May say it 1 t 7
SHE C D 2 JNC ILP Het next bit until CV shows we got F C 9 RET	SIS			1	E	<u> </u>			11		E	 	0	-			Nov Carry into but
BE P2 CY shows we got all 8	밝			片	2				1	-		1		-	\vdash	-	Nave present buls
E P 2 all 8 F C 9 R E 7	χ			2			۲	<u>'' </u>	٢		-	 - -	-	\vdash	 		set next but until
F C9 RET	띪			_	_		-	┝	 	_		-	-	┝	╁─		CY shows we got
	찱					 -	0	_	7		-	├		-	├	\vdash	all 8
	봙	0 4 5		۲	7_		1	E	/_		-	<u> </u>	\vdash	-	├-	-	
8 2 0	낅	* 2 D		├	-	 -	⊢	-	-	<u> </u>		-	-	_	-		0.4
H 1 Returns with	비	·		-	-	 		-	\vdash	\vdash	├—	<u> </u>	\vdash	-	-	\vdash	Keturno with
Zast keyboard Sast keyboard Charactel in regs A and E	AA A			$oxed{igwedge}$	_		<u> </u>	_	\vdash	_	 	<u> </u>	_	_	<u> </u>		Last keyboard
Charactel in	띮			_	_	<u> </u>	_		<u> </u>	_	-		_	_	<u> </u>		character in
	되			_	<u> </u>		igspace				1_	<u> </u>		_	<u> </u>	\square	regs A and E
5	1		5	$oxed{oxed}$	_	ļ	<u> </u>		<u> </u>	_	<u> </u>	<u> </u>	_	_	_		
6					_	L	$ldsymbol{f eta}$	_	_		$oldsymbol{ol}oldsymbol{ol}ol{ol}}}}}}}}}}}}}}}}}$			<u></u>	<u> </u>	igsqcut	
7			7	$oldsymbol{f igl }$		ļ	<u> </u>	L	L	L			L.	L			
			8									٠					Figure D-7c

	A D D	R	co	DE	OUTI	PUT	C	ГИC	EN'	rs	OF	RE	G I	E T	0 2	ASR 33 TELETYPE
	8 2	0	3	E	OCHR	M	V	I		A		0	3			Output start bit
		1	0	3							7					
		2	0	Ŋ		0	A	7	L		0	U	\mathcal{T}	B		
. 1		3	E	<u>C</u>			-		_		J					
		4		2												
SKEI		5	A	$\frac{\infty}{E}$		X	R	A		A						00
		6	77	/		S				$\frac{7}{3}$			Ø			Clear carry
CODING			/	00		/*(V	I	-) 	0	8	_		Tritialize Counter
싱		7	0	8					-]				
		8	Z	0	OLP	M	0	V		A	,	E				Set character
		9	2	F	<u> </u>	Ç	M	H	ļ							Invertit
		Α	0	F		K	R	2								Shift into carry
		В	2	F	l	C	M	A								Invertit back
_		С	5	F		M	0	V		E	-/-	A				Saveit
		D	3	E		M	V	\mathcal{I}		A		0	1			SYCY = 0, then A -> 02
SYSTEM		E	0	/					1		,					Secy = 1 then A -> 03
		F	8	F		A	0	C		A						
AINING	8 2E	0	0	\overline{C}		0	A	2	4		0	U	7	B		Output to Port ICI
¥		1	E									_		~		cargue se i o o ·
H		2	8	2			-	-	-		-			-		
ER		3	1	5		D	C	R	-	7	-				-	Done 8 lits ?
MICROCOMPUT		4	1	2		5	-	7	-	0	<u>ر</u>	P		-		None obils
Š		5	5	8		۲	I/V	E	-	0	_			-		Not yet
ö			0	_			-		-			-		-		
CR		6	8	2	 	-	1	.	ļ.		Ö				0	
≅	<u> </u>	7	C	2		C	H	1	4	_	Š	T	0	P	\mathcal{D}	Yes; output 1stop list
		8	E	H		<u> </u>	 									! /
		9	8	2		_	ļ.,									Hall into 2nd top bit Stop bit routine Bit output routine
S		Α	3	E	STOPB	M	V	I		A	_	0	2			Stop but routine
SYSTEMS		В	0	2			Ĺ				_					
/ST		С	7		OUTB	0	U	7		C	N	$ \mathcal{T} $	上			Bit output routine
		D	0	7									,			
TEF		E	0	E	DLY	M	V	I		C		0	9			Delay I bound routine
INTEGRATED COMPUTER		F	0	9						T _	1					The state of the s
Θ	8	0					ļ		_		-	-				
)]		1	\vdash		 	<u> </u>	<u> </u>	-	<u> </u>		ļ	<u> </u>		-	\vdash	
/TE	<u> </u>	2	+-	-		-	-	-	-	+-	-	-		-		
3R/			\vdash	<u> </u>	 	-	\vdash	-	-	1		-	\vdash	 		
TE(3	├-	-	 	 -	-	-	\vdash	-	-	\vdash		-		
Z		4	├		 	┼─	-	-	-			-	-	+-	-	
		5	┼-	-	 	-	 	-	-			-	-	-		
		6	1	-	 	 	┼-	-	-	 	-	-		\vdash		
		7	↓_	-		<u> </u>	<u> </u>	_	-	ļ		_	-	_	 	
		8	1	l _	<u> </u>	<u> </u>	ŀ									Figure D-7d

TELETYPE PROGRAM INPUT AND OUTPUT ROUTINES (continued) Delay / milline cond B 2F0 DEY 8 1 DCR B 2 5 DLYA JNZ $\overline{\mathcal{C}}$ Z \mathcal{D} 1/2 3 **CODING SHEET** 2 4 S 5 82F D 6 R JNZ 2 7 0 1 8 0 2 9 9 C RET Α B TBIT Let bit PORT 82FB \mathcal{I} N C 0 MICROCOMPUTER TRAINING SYSTEM Invert but Bit -> CY 2 CMA D E C 9 F 0 1 2 3 4 5 6 7 8 9 Α INTEGRATED COMPUTER SYSTEMS В C D E F 8 0 1 . 2 3 4 5 6 7. Figure D-7e 8

D-21

	A D D R	co	DE .	TELETY	PΕ	PROG	RAM	MES	SSAGI	ES	
	8 30 0	8	D								CR
	1	18	D		-						CR
	2	8	A	_							CR CR LF
_	3	C	8								#
EE	4		5								E
3 S	5	C	\mathcal{C}								7
CODING SHEET	6	C	C								L
200	7	C	F								0
	8	A	C								5
	9	A	0								
	Α	C	D								M
	В	D	9								У
_	С	A	0								
MICROCOMPUTER TRAINING SYSTEM	D	C	E								N
YS.	E	C	1								A
S S	F	C	2					_			M
N	83/0	C	5				_				E
RAI	1	A	0					_			
ВT	2	C	9				_	\perp			I
JTE	3	D	3		_					_	5
MP	4	A	0							_	
20	5	0	4					_ _	-		7
CRC	6	C	8			- -	-	_		_	H
Ĭ,	7	C	5					_		_	E
	8	A	0				-	_		_	
	9	C	-					_			M
SI	A	Ž	4			-			-		T
TEN	В	7	3				-		-	\perp	S
ΥS	C	A	F	 			-	_			
ER S	D	12	4			-	-	_			T
UTE	E	2	4				-		-		T
MP	F	D	9		_	- -	-			_	У
CC	8320	A	0					-			
TEC	1	2	4					-+			T
ЯY	2	<u>C</u>	5				-	-		-	E
INTEGRATED COMPUTER SYSTEMS	3	7	3		-		-	-	+		2
Z	4		4	-	-				+		T
	5	A			-		+-1	+	+ 1		
	7	+	Н	 - 	-		 - 				
,	8	+-	\vdash	 -	_		-	-	+		Figure D-7f
	L	ــــــــــــــــــــــــــــــــــــ		<u> </u>	L	Щ.		ــــــــــــــــــــــــــــــــــــــ			19

TELETYPE PROGRAM MESSAGES (continued)

А	D D	R	со	DE	TELI	ETY	PΕ	PR	OGF	MAS	ME	ESS	AGI	ES	(cc	ontinued)
8	32	0														
	<u> </u>	1	ł													
		2														
		3														
SHEE		4	1							_						
FS -		5	<u> </u>	1		_										
CODING		6	D	0												P
		7	7	2	-					_						R
اد		8	C	F									_			0
		9	C	7												G
		A	2	2												R
		В	C	7			_								\vdash	Ä
-		С	C	カ	_							-		-		M
≅├─		D	A				_									
<u>-</u>		E	8	E			_	-		_	-			 - -		CR
MICROCOMPUTER TRAINING SYSTEM	<u>. </u>	F	8	D		-	_		- 1			-				CR
	33	0	0	A										<u> </u>		CR LF
<u> </u>	<u> </u>	1	8	7												W
<u>~</u>		2	C	8				 								
# <u> </u>		3	C	/	<u> </u>	-	·			-i				 	-	A
5-		4	2	4		-		-					_	<u> </u>		T
<u> </u>		5	A		_									<u> </u>	-	
<u>Ş</u> —		6	C	9		┝	<u> </u>						_			T
╧├─		7	3	3			\vdash					_				I S
≥ ├─		8	A	-			<u> </u>	-				_		\vdash		
-		9	7	9		\vdash								-		У
-		A	1	_	-	<u> </u>	-	 				_	<u> </u>	-	_	0
~	-	В	C	F		-	<u> </u>						<u> </u>	\vdash		
;;}—	_	c	7	2		_	_							_	-	U R
-\ <u>}</u>		D	A			<u> </u>	-	ļ		_						
<u> </u>		E	C	E		<u> </u>	┢							\vdash		AJ
5-		F	0	[]		-	 	-	$\vdash \vdash$				<u> </u>	-	-	A
INTEGRATED COMPUTER SYSTEMS	34		C	D		-	-	\vdash	Н			\vdash	-	\vdash		M
	<u>0 7</u>	1	1 <u>C</u>	5		-		-	$\vdash \dashv$			-	_			
屵		2	\mathcal{B}			-	-		H				-	-	-	E
% ├─		3	A	FO		-	-	-	-			<u> </u>		_	-	
┋├─		4	0	0		-	-	-	-			ļ	\vdash	-	_	505
≤├─	<u> </u>	5	10	1		-	-	-	-				_		\vdash	EOF
<u> </u>		6	+	$\vdash \vdash$									\vdash	-	 -	
 	:	7	+	$\vdash \dashv$		\vdash	_	\vdash	\vdash	-	-	-	 		-	
<u> </u>		8	+	$\vdash \dashv$		-	-	+	-		-	-	-	-	\vdash	Figure D-7g
Ц				Ц.,		L	<u>L</u>	1		L	L			1	<u> </u>	D-23

	_A D_D R	<u> </u>	<u>co</u> 1	DΕ	TEL	ETY	PE	PR	OGI	RAM	ME	SS	AGI	ES	(cc	onti	nued	.)
	8 0																	
	1								_									
	2																	
L	3		T	\neg							\exists							
SHEET	4	. 1				-						┪		_				
SH	5																	
CODING	8346		P	D												CF	>	
00	7			D		-					_	1				CF		
O	8		8	Ā	-						7					LF		
	9		$\overline{}$	4							7	_		_		<u></u>		
	A			8							_	\dashv				H		
	В		0	7				-				_		_		A	-	
	С		-+	Ē				_		\vdash	_	7		_		N		
ΕĞ	D		$\overline{}$	$\overline{\mathcal{B}}$								╗			-	K		
/STI	E			0		\vdash		_				_						
SS	F			9							7	_	_			У		
Ž	0 سي جي 8		7	F		-					_					Ó	-	
TRAINING SYSTEM	1		_	5		_	H					1				U		
	2		_	C					_		7		_					
MICROCOMPUTER	3		\neg	0							_					-		
P	4			0		-		-	_			1				Ec) F	
Š	5		0	0												1		
ROG	6		$\overline{}$	0									_				-	
S	7	-	0	0													 · · ·	
_	8			0								1						
	9	7	<i>)</i>	0								1						
	A		0	0														
ΞMS	В		0	0	-	-			_			7						
'STI	С		0	0														
₹S.Y	D	-	0	0														
TEF	Е		0	0														
1PU	F		0	0														
800	836 0	-		0	,													_
ED (1		0	0	- _													
ATE	2		0	0														
EGR.	3		0	0					_									
INTEGRATED COMPUTER SYSTEMS	4		0	0														
_	5		0	0														
	6		2	0	1													
	7	$ I_{\alpha}$	0	0												J		
	8		0	0			ļ									Ec	F	Figure D-7h

	•				TELI	ETY	PΕ	PR	OGI	RAM	ME	ESS	AGI	ΞS	(cc	ontinued)
(8 D D	0 0	СО	DE												
		1				Н			\vdash							
		2	<u> </u>		-	Н								_		
Ì		3	_						-						-	
E	-		_			_				\vdash		$\overline{}$			<u> </u>	
H		4														
CODING SHEET		5	_			_										
N		6	<u> </u>													
Ö		7	<u> </u>		<u> </u>											
		8														
	836	9	0	0												<i>EOF</i> ↑
		A	0	0												<u> </u>
		В	0	0												·
		С	0	0												
EM		D	0	0												
/ST		E	0	0												J
MICROCOMPUTER TRAINING SYSTEM		F	-													EOF CR CR LF LF LF LF
N	837	0	8	0												CR
AIN		1	8	000												CR
TR		2	8											_		1 F
FE	-	3	8	A A A	<u> </u>			_								1 F
PUJ		4	8	A											-	I F
OM		5	8	A		\vdash								<u> </u>		IF
SOC		6	8	A												15
CF	_	7	8	7				-						<u> </u>		BELL
2		8	8	7										 		DELL
		9	8	7			H					-		_		BELL BELL BELL
		A	T -			\vdash		_					_			
NS			0	0						-				<u> </u>		EOF
TE		В	├ —						_					<u> </u>		
SYS		<u>c</u>	-	H		_	<u> </u>							<u> </u>		
ER S		D	<u> </u>						_					ļ		
INTEGRATED COMPUTER SYSTEMS		E	_						ļ	Щ			-	<u> </u>		
Σ	<u> </u>	F	_			_	ļ						-	 		
S	8	0	<u> </u>			<u> </u>					. —		ļ		L	
<u> </u>		1	 			<u> </u>		_	_					<u> </u>	L	
RA		2	<u> </u>						ļ					ļ		
S		3	L	_			L	_					<u></u>	<u> </u>		
Ĭ		4										L	L			
	L	5	L				<u>L</u> .	_	<u> </u>			_		_	<u> </u>	
		6		1					_							
		7	L					L							L	
		8														Figure D-7i

This page intentionally left blank.

	•	

		4

	•		
•			

	-	

	/	
		/

		~	
1			



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

FRANCE

ICS France 90 Ave Albert 1er 92500 Rueil-Malmaison France

Telephone: (01) 749 40 37 Telex: 204593

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

GERMANY

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

EUROPEAN HEADQUARTERS

ICSP – U.K. Pebblecoombe, Tadworth Surrey KT2O 7PA England Telephone: Leatherhead (O3723) 79211 Telex: 915133

SCANDINAVIA

ICSP Inc. – Scandinavia Utbildningshuset AB Box 1719 S-221 OT Lund, Sweden Telephone: (O46) 3O 7O 7O Telex: 33345