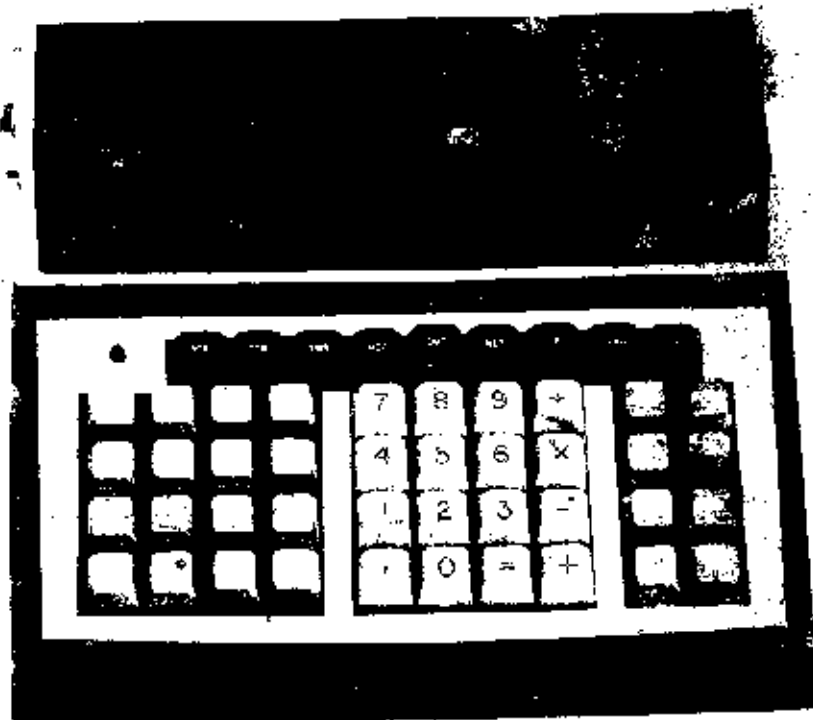


# TEXAS INSTRUMENTS

## PROGRAM MASTER 550

### INTELLIGENT PROGRAMMABLE CONTROL SYSTEM PROGRAMMING MANUAL



5342  
3042

# **PROGRAM MASTER 550 INTELLIGENT PROGRAMMABLE CONTROL SYSTEM**

## **PROGRAMMING MANUAL**

Copyright © 1979  
by  
Texas Instruments Incorporated

# TABLE OF CONTENTS

## Chapter 1

Paragraph	Title	Page
	<b>INTRODUCTION</b>	1 1
1 0	General	1-1
1 1	Purpose of the <i>PM550 Read/Write Programmer Manual</i>	1-1
1 2	Features of the PM550 300 Programmer	1 1
1 3	User Memory	1 2

## Chapter 2

	<b>SYSTEM OPERATION</b>	2 1
2 0	System Theory of Operation	2-1
2 1	Central Control Unit Theory of Operation	2-4
2 2	Scan Time	2-4

## Chapter 3

	<b>PREPARATIONS FOR USING THE PROGRAMMER</b>	3 1
3 0	General	3-1
3 1	Assigning Input/Output Identifiers to Ladder Diagram	3 1
3 2	Redrawing of Ladder Diagram	3 2
3 3	Documentation of Ladder Logic Programs	3-2
3 4	Preparation for Loop Entry	3-2

## Chapter 4

	<b>PROGRAMMER OPERATING INSTRUCTIONS</b>	4-1
4 0	General	4-1
4 1	Ladder Entry Keys	4-1
4 2	Loop and Auxiliary keys	4 4
4 3	Display	4 4
4 4	Power Flow	4 4
4 5	Beeper	4 4

## Chapter 5

	<b>HOW TO ENTER LADDER DIAGRAMS</b>	5 1
5 0	General	5-1
5 1	Clearing Memory	5-3
5 2	Single Contact — Programming Exercise No. 1	5 4

Paragraph	Title	Page
	Ladder Element (L Memory) Storage Record	5-6
5.3	Simple Series and Contacts - Programming Exercise No. 2	5-6
5.4	Use of <b>RLAD</b> Key to Check Contents of Memory	5-9
5.5	Use of Backstep - <b>BSTEP</b>	5-10
5.6	Use of the Power Flow Lamp	5-10
5.7	Use of the <b>FIND</b> Function	5-11
5.8	Changing a Ladder Diagram After Entry Into the CCU	5-11
5.9	Simple Parallel <b>OR</b> Contacts-Programming Exercise No. 3	5-16
5.10	Multiple Outputs-Programming Exercise No. 4	5-18
5.11	Parallel-Series <b>OR</b> - <b>STR</b> Contacts Using <b>OR</b> - <b>STR</b> Function	5-20
5.12	Programming Exercise No. 5	5-22
5.13	Use of <b>AND</b> - <b>STR</b> Function-Programming Exercise No. 6	5-24
5.14	Using <b>OR</b> - <b>STR</b> and <b>AND</b> - <b>STR</b> Functions - Programming Exercise No. 7	5-29
5.15	Complex Ladder Diagram - Programming Exercise No. 8	5-33
5.16	Redrawing Complex Ladder Diagrams	5-34
5.17	Programming <b>NOT</b> Function	5-39
5.18	Programming Normally Closed Inputs	5-41
5.19	Programming <b>MCR</b> Function	5-45
5.20	Programming Jump Function	5-49
5.21	Programming Shift Registers	5-51
5.22	Holding Circuits and Latch Relays	5-51
5.23	Programming End of Scan <b>END</b> Function	5-55
5.24	Defining Integer Values	5-58
5.25	Ladder Logic Multi-Word Functions	5-85
5.26	Reading and Writing the Image Register	

## Chapter 6

	ENTERING SPECIAL FUNCTIONS	6-1
6.0	Principle of Operation	6-1
6.1	Entering Non-integer Values	6-4
6.2	SF1 - Binary to BCD Conversion	6-8
6.3	SF2 - BCD to Binary Conversion	6-9
6.4	SF3 - Scale	6-11
6.5	SF4 - Unscale	6-13
6.6	SF5 - Square	6-15
6.7	SF6 - Square Root	6-16
6.8	SF7 - Compare	6-18
6.9	SF8 - Math	6-21
6.10	SF9 - Sequential Data Table	6-24
6.11	SF10 - Correlated Data Table	6-28
6.12	SF11 - ASCII String	6-30

<i>Paragraph</i>	<i>Title</i>	<i>Page</i>
6.13	SF12 -- Synchronous Shift Registers . . . . .	6-34
6.14	SF13 -- Asynchronous Shift Registers -- Input . . . . .	6-35
6.15	SF14 -- Asynchronous Shift Registers -- Output . . . . .	6-36
6.16	Chaining Special Functions . . . . .	6-37
6.17	SF0 -- The Entry Point Special Function . . . . .	6-40

## Chapter 7

	<b>PROGRAMMING PROCESS CONTROL EQUATIONS -- LOOPS . . . . .</b>	<b>7-1</b>
7.1	Basic Loop Features of the PM550 . . . . .	7-2
7.2	Specifying Loop Configurations . . . . .	7-27
7.3	Loop Errors . . . . .	7-40

## Chapter 8

	<b>AUXILIARY FUNCTIONS . . . . .</b>	<b>8-1</b>
8.1	AUX 1 -- Clear Memory . . . . .	8-1
8.2	AUX 2 -- Program Mode/Single Scan . . . . .	8-2
8.3	AUX 3 -- Programmer Self-Test . . . . .	8-3
8.4	AUX 4 -- Central Control Unit (CCU) Self-Test . . . . .	8-6
8.5	AUX 5 -- Read/Clear Error Table . . . . .	8-7
8.6	AUX 6 -- Tape Control . . . . .	8-11
8.7	AUX 7 -- Entering ASCII Message . . . . .	8-15
	<b>Appendix A . . . . .</b>	<b>A-1</b>

# LIST OF ILLUSTRATIONS




Figure	Title	Page
2.0A	Typical System Block Diagram	2-2
2.1A	Central Control Unit Block Diagram	2-5
3.0A	Sample Ladder Diagram	3-3
3.1A	I/O Wiring for Sample Ladder Diagram	3-4
3.1B	Input/Output Assignment Record for Sample Diagram	3-5
3.1C	Auxiliary Input/Output Record Form for Sample Diagram	3-6
3.1D	Constant (K) Memory Assignment Record	3-7
3.1E	Variable (V) Memory Assignment Record	3-7
3.2A	Redrawn Sample Ladder Diagram	3-8
3.3A	Ladder Memory Storage Record for Sample Diagram	3-9
4.1A	Read/Write Programmer Key Display	4-5
4.3A	Typical Ladder Location Display	4-5
5.0A	System Block Diagram	5-2
5.2A	Ladder Diagram for One Output Controlled by One Input	5-5
5.3A	Ladder Element (L Memory) Storage Record Form (Exercise No. 1)	5-6
5.4A	Ladder Diagram for Two Input Elements in Series	5-8
5.4B	Ladder Element (L Memory) Storage Record (Exercise No. 2)	5-8
5.9.1A	Use of  Insert Function	5-1
5.9.2A	Use of  Delete Function	5-1
5.9.3A	Changing an Element in a Ladder Diagram	5-1
5.9.4A	Changing an Element in Ladder Diagram Using 	5-1
5.10A	Ladder Diagram for Programming Exercise No. 3	5-1
5.10B	Ladder Element (L Memory) Storage Record (Exercise No. 3)	5-1
5.11A	Ladder Diagram for Programming (Exercise No. 4)	5-1
5.11B	Ladder Element Storage Record (Exercise No. 4)	5-1
5.12A	Ladder Diagram for Programming (Exercise No. 5)	5-1
5.12B	Ladder Element (L Memory) Storage Record (Exercise No. 5)	5-1
5.13A	Ladder Element (L Memory) Storage Record (Exercise No. 6)	5-1
5.13B	Ladder Diagram for Programming Exercise No. 6	5-1
5.14A	Ladder Diagram for Programming Exercise No. 7	5-1
5.14B	Ladder Element (L Memory) Storage Record (Exercise No. 7)	5-1
5.15A	Ladder Diagram for Programming Exercise No. 8	5-1
5.15B	Ladder Element (L Memory) Storage Record (Exercise No. 8)	5-1

Figure	Title	Page
5.16A	Original Ladder Diagram	5-33
5.16B	Redrawn Ladder Diagram	5-34
5.17A	Status of Each I/O Device is Applied to Mounting Base	5-35
5.17B	Using NOT Function to Invert Input Switch	5-35
5.17C	Ladder Element (L Memory) Storage Record (NOT Function)	5-36
5.17D	Using NOT Function to Invert Control Relay (CR)	5-37
5.17E	Ladder Element (L Memory) Storage Record (Inverting Control Relay)	5-38
5.19A	Master Control Relay Function Using Relay Terminology	5-42
5.19B	Master Control Relay Function Using the PM550 CCU	5-43
5.19C	Ladder Element (L Memory) Storage Record for MCR Function	5-44
5.20A	Ladder Diagram of 2-Line JUMP Function	5-46
5.20B	Ladder Element (L Memory) Storage Record for the JUMP Function	5-47
5.21A	Ladder Diagram and Program for Shift Register	5-50
5.22A	Programming Latch Outputs	5-51
5.23A	PM550 Timing Diagram without FOS	5-52
5.23B	PM550 Timing Diagram with EOS at Location No. 500	5-52
5.23C	Scan Time Formula	5-53
5.23D	Ladder Diagram and Storage Record for FOS	5-53
5.24A	User Memory Storage Record	5-57
5.25.1A	Ladder Diagram and Storage Records for the Counter Function	5-59
5.25.2A	Ladder Diagram and Storage Records for Delay on Timer	5-63
5.25.2B	Delay on Timer for Two Elements, Ladder Element and User Memory Storage Records	5-64
5.25.2C	Ladder Element and User Memory Storage Records for Timer Function	5-66
5.25.3A	Simple One-Input Delay Off Timer	5-68
5.25.4A	Extended Timer Using Cascade Method	5-70
5.25.4B	Extended Timer Using Counter Method	5-71
5.25.5A	Ladder Diagram and Storage Record for Ladder Logic Integer MOVE Function	5-72
5.25.6A	Ladder Diagram and Storage Record for Compare Function	5-76
5.25.6B	Power Flow Used to Select Type of Compare	5-77
5.25.7A	Ladder Diagram and Storage Records for Integer Addition	5-79
5.25.8A	Ladder Diagram and Storage Records for Integer Subtract	5-81
5.25.9A	Ladder Diagram and Storage Record for Timer Function	5-83
5.25.9B	Delete of Multi-Word Function	5-84
5.25.9C	Insert of Multi-Word Function	5-85
6.3A	Special Function Description Table	6-2
6.0B	Special Function Operation	6-3
6.0C	Ladder Diagram and Storage Record for Special Function Request Line	6-4

<i>Figure</i>	<i>Title</i>	<i>Page</i>
6 1A	Example of "F" Notation . . . . .	6-5
6 10A	Cam Curve for Sequential Data Table Example . . . . .	6-24
6 11A	Sample Correlated Data Table . . . . .	6-28
7 1A	Process Unit With Three Conventional Feedback Control Loops . . . . .	7-3
7 1B	LAM Loop Access Module (LAM) . . . . .	7-4
7 1C	A Computed Variable, the Heat Transfer in an Exchanger . . . . .	7-5
7 1D	Current Loop Input on the PM550 . . . . .	7-6
7 1E	Role of the Auto Manual Mode Selection . . . . .	7-11
7 1F	Example Requiring a Direct Acting Controller . . . . .	7-14
7 1G	Example Requiring a Reverse Acting Controller . . . . .	7-14
7 1H	Error Deadband Relationship . . . . .	7-17
7 1J	Cascade Switch . . . . .	7-18
7 1K	Cascade Control System . . . . .	7-20
7 1L	Cascade Control Configuration in the PM550 . . . . .	7-21
7 1M	Process Variable Alarms . . . . .	7-24
7 1N	Deviation Alarms . . . . .	7-26
7 2A	Data Entry Conversions . . . . .	7-28
7 2B	Typical Loop Specification Sheet . . . . .	7-29
7 2C	Length of V and C Tables . . . . .	7-32
7 2D	Typical Sequence for Entering Loop Table Memory Allocations . . . . .	7-32
7 2E	Typical Sequences for Entering the Process Variable Specifications . . . . .	7-34
7 2F	Example of a CCU of Commands for Entering Control Applications . . . . .	7-37
7 2G	Typical <span style="border: 1px solid black; padding: 0 2px;">ENTER</span> Sequence for Entering Alarm Specifications . . . . .	7-39
7 2H	C Table Structure . . . . .	7-41
7 2I	V Table Structure . . . . .	7-44
7 2K	IR Bits Associated with Loops . . . . .	7-47
7 3A	Loop Error Table . . . . .	7-48
8 3A	Read/Write Programmer Self Test Program Flow . . . . .	8-5
8 5A	System Error Table by Number . . . . .	8-9
8 6A	STR LINK II Hookup . . . . .	8-11
8 7A	ASCII Character To Decimal Code Chart For Aux 7 . . . . .	8-16
8 7B	Sample Message Coding . . . . .	8-17
8 7C	Memory Locations for AUX 7 Example . . . . .	8-17



## CHAPTER 1

### INTRODUCTION

#### 1.0 GENERAL

The PM550-300 programmer provides the user with a means of interactively programming ladder diagrams, boolean equations, process control loops, and special functions such as user math and look up tables, into the PM550 central control unit (CCU). The CCU uses state of the art microprocessor technology to perform the programmed functions replacing relays, card logic, analog controllers, and minicomputers with one integrated, easy to program system. The interactive programming language prompts the user for the necessary data to program any function. Ladder diagrams and boolean logic can be programmed directly — no special programming language need be learned. The power flow indicator along with the programmer display facilitate analyzing the system operation.

#### 1.1 PURPOSE OF THE PM550 READ/WRITE PROGRAMMER MANUAL

The purpose of this manual is to provide operating instructions for the PM550-300 programmer and sufficient programming examples to enable the user to program the PM550 with confidence and efficiency. The user progresses gradually from programming simple ladder logic expressions to setting up complex process loop equations. The user is urged to program each example and verify it using the 511-4100 simulator. Familiarity with ladder diagrams and basic process control are assumed.

#### 1.2 FEATURES OF THE PM550-300 PROGRAMMER

The following features enhance the usability of the PM550-300 programmer:

**Direct Entry using relay ladder symbology:** ladder diagrams can be programmed without translation.

**Direct Entry using boolean equations:** logic sequences written in boolean form can be programmed without modifying the sequence.

**Interactive programming** display prompting requests necessary data to program advanced functions and loops

**Powerful Editor:** find, back step, delete and insert functions make program documentation verification, modification, or error correction easy

**Twenty character alpha numeric display:** display of logic and data is in alphanumeric format.

**Error diagnostics:** display provides error messages or additional prompting for incorrect programming entries

**Fault isolation:** power flow indicator shows logic states for any point in the ladder diagram facilitating rapid troubleshooting of the control logic.

**Multiple References:** any input (X), output (Y), control relay (CR) or user memory location (V, C or A) may be referenced as many times as required.

### 1.3 USER MEMORY

The PM550 user accessible memory includes:

L - Logic programming 2048/4096 words

C - Constant Data 1024/2048 words.

V - Variable Data 1024 words

A - 64 Analog and/or Parallel words - The 7MT modules are the memory storage locations.

In the CCU's image register are the status of:

X - Up to 256 discrete inputs.

Y - Up to 256 discrete outputs

CR - Up to 512 control relays or internal flags (CR0 CR255 are retentive).

## CHAPTER 2

### SYSTEM OPERATION

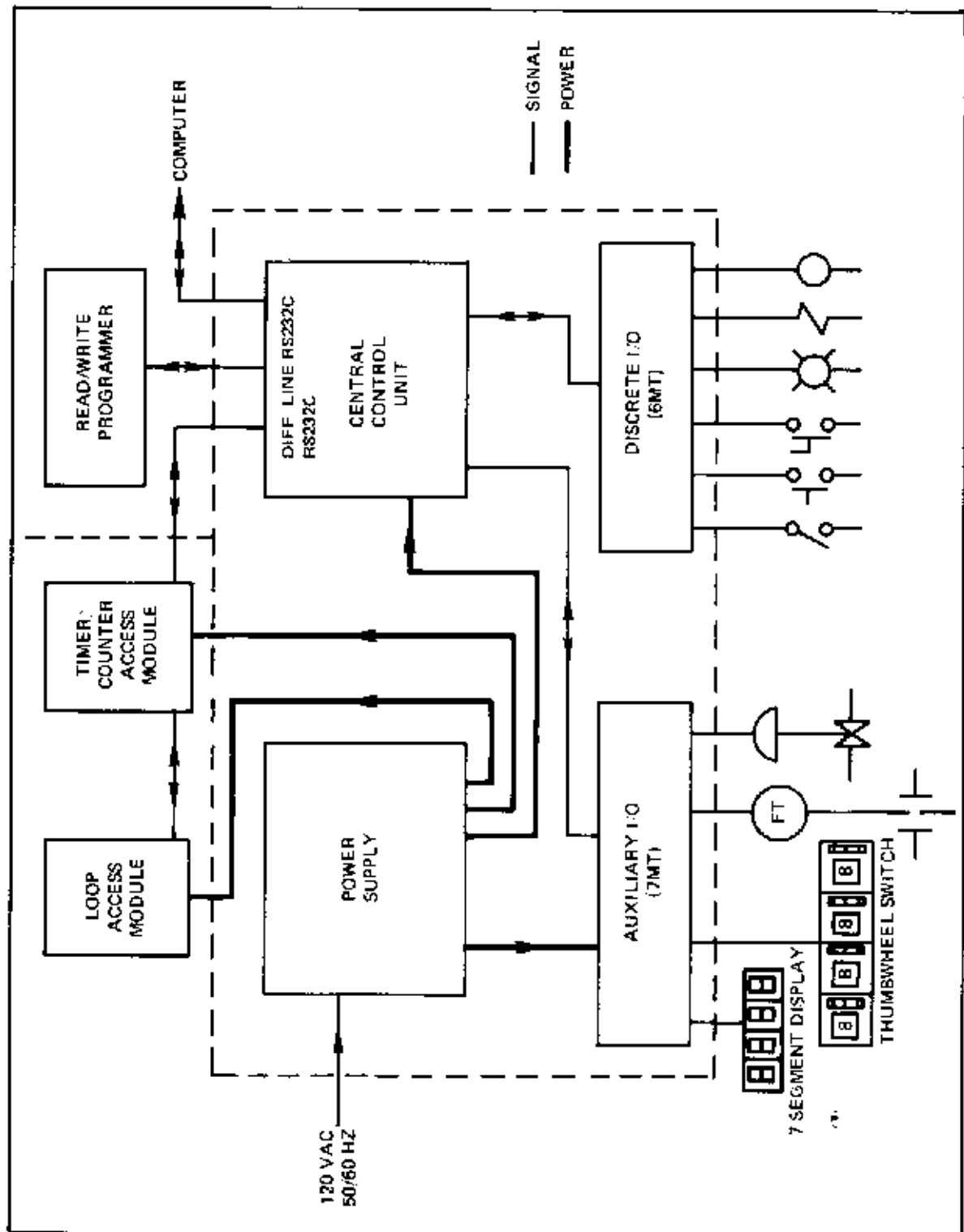
#### 2.0 SYSTEM THEORY OF OPERATION

Figure 2.0A illustrates a typical PM550 system consisting of a central control unit, power supply, 6MT discrete I/O, 7MT auxiliary I/O, a timer/counter access module, a loop access module, and a read/write programmer. A read/write programmer is not a part of the final system configuration, it is only used to program the users application. Up to 512 points of 6MT discrete I/O can be added to the system through the use of the 511-5500 I/O expander (see *Installation and Operation Manual* for details of requirements). Two racks of 7MT auxiliary I/O can be driven from the central control unit. Eight timer/counter access modules and eight loop access modules can be connected to the differential line; however, only two modules can be powered from the system power supply. Each is thoroughly covered in the *Installation and Operation Manual*.

The read/write programmer communicates with a central control unit over one of the two RS232C ports. Many RS232C compatible devices may interface through either of the RS232C ports on the central control unit, such as printers, modems, host computers, and the EPI STR LINK II cassette loader.

The power supply requires 120 volts AC 50/60 Hz input (*Installation and Operation Manual* section 3) which is converted to the necessary operating voltages for the rest of the system. The power supply employs advanced switching regulator design to operate over voltage fluctuations. High noise immunity to transients is insured by an input line filter and other advanced design techniques. The power supply also provides a power down signal to the central control unit so the CCU can complete its scan and complete a data save routine before the DC voltages from the power supply fall below the normal operating range. A battery pack attached to the CCU provides battery backup for all system memories for at least 72 hours. Provisions are provided for an external battery input to extend this time.

The central unit contains all the intelligence for the PM550 system consisting of a dual 16 bit microprocessor architecture. One microprocessor executes the sequential logic in the PM550, the other handles loop processing, special functions, and communications. The CCU communicates to RS232C compatible peripherals through two identical ports at 300 or 1200 baud. It also communicates



with operator interface peripherals (timer/counter access modules and loop access modules) via a 9600 baud differential line port. Analog and parallel data to user input/output is handled through the 7MT auxiliary I/O. Single bit or discrete I/O is handled through the 6MT input/output system.

The 6MT discrete I/O system interfaces to on/off devices at industry standard voltage levels. Inputs and outputs are optically isolated from the CCU by the 6MT modules with breakdown voltages of up to 3800 volts. AC and DC input and output modules are available with voltage ranges from 5 to 220 volts with current ranges up to 3 amps. The 6MT is part of the widely accepted Model 5TI programmable controller system and is compatible with the PM550.

The 7MT auxiliary I/O provides a TTL level parallel interface to items like thumbwheel switches and 7 segment displays. It also provides analog to digital and digital to analog conversion to interface signals found in the process control environment. Standard input ranges are 0 - 5 volts full differential and 0 - 20 milliamperes. The analog outputs range from 0 - 10 volts and 0 - 20 ma. This allows interface to standard temperature, pressure, flow, or other transmitters as inputs and is capable of driving control valves, current to pneumatic convertors, SCR controllers and other similar output devices.

The timer/counter access module (TCAM) is an operator interface to the PM550. It allows access to the first 99 timers and 99 counters programmed in the PM550. Both the current value and the preset value can be read by the TCAM and depending on how the timer or counter is programmed, the preset and current values may be modified. Up to 8 TCAMS can be connected to the differential line in a series or daisy chain fashion.

The loop access module (LAM) is the second operator interface to the PM550. It is the digital analog of the front panel of a 3 mode process controller. It continuously displays the process variable of any of the 8 loops controlled by the PM550. It will selectively display on its second digital display any of the following - setpoint, bias, deviation, output, gain, rate, or reset (all in engineering units). All these parameters can be modified by the LAM depending on loop mode and loop programming. Also included are loop alarm lamps and a simulated deviation display.

The read/write programmer, an independent peripheral to the PM550 system, provides the user with an entry and troubleshooting ability far more powerful than has ever been offered on a programmable controller. With its English language prompting and free format ladder diagrams, the programmer allows the user to quickly and efficiently program his application in the PM550. A read/write programmer communicates through one of the RS232C ports on the CCU and is capable of remote operation over standard telephone lines using external modems.

Other peripherals may be connected to the RS232C ports to communicate with the PM550. A printer such as the TI Silent 700 model 745, may be connected directly to a RS232C port and print out ASCII messages. The EPI STR-LINK II program loader may be also connected to an RS232C port.

It can be used to record, load, or verify PM550 programs. Also the PM550 can be connected to a host computer via RS232C ports, allowing the PM550 to become part of an advanced distributed control system.

## 2.1 CENTRAL CONTROL UNIT THEORY OF OPERATION

The PM550 central control unit contains two independent 16 bit microprocessors (Figure 2.1A). A custom microprocessor, the MP9514, is the logic processor which handles the ladder logic, timers, counters, move, compare, integer add, and integer subtract. It also provides the logic necessary to initiate a special function process. The second microprocessor, a TMS9900, is the master controller in the PM550. It handles all the communications through the two RS232C ports and the differential line, it also performs the loop processing and special function as requested by the logic processor. The logic processor resides on the common memory unit bus and has access to the 7MT auxiliary I/O, the constant data memory area, the variable data memory area, and the ladder logic program memory. It also communicates with the 6MT discrete I/O through the image register and can pass data to the control processor by the common memory bus. The control processor also resides on the common memory unit bus and has access to the same memory areas as the logic processor. In addition it has control memory areas that contain the executive operating system for the PM550.

In normal operation the logic processor continuously executes the user's logic program from the ladder logic program memory. It accesses the variable and constant data memory areas, the auxiliary I/O (7MT) and the image register as necessary. If the logic processor encounters a special function it halts and interrupts the control processor. The control processor recognizes the special function and places it in a first in first out stack (a queue) to be processed as time becomes available. The logic processor is then restarted and continues with the ladder logic scan. In the meantime the control processor is handling the data communications, loop processing, and special functions on a time shared basis. When the control processor completes a special function it communicates this to the logic processor by setting the output power flow of the special function to a true (ON) condition.

## 2.2 SCAN TIME

Scan time is a nominal 17 ms for a 2k logic program and 31 ms for a 4K logic program. Exact scan rate formula is given in section 5.23.

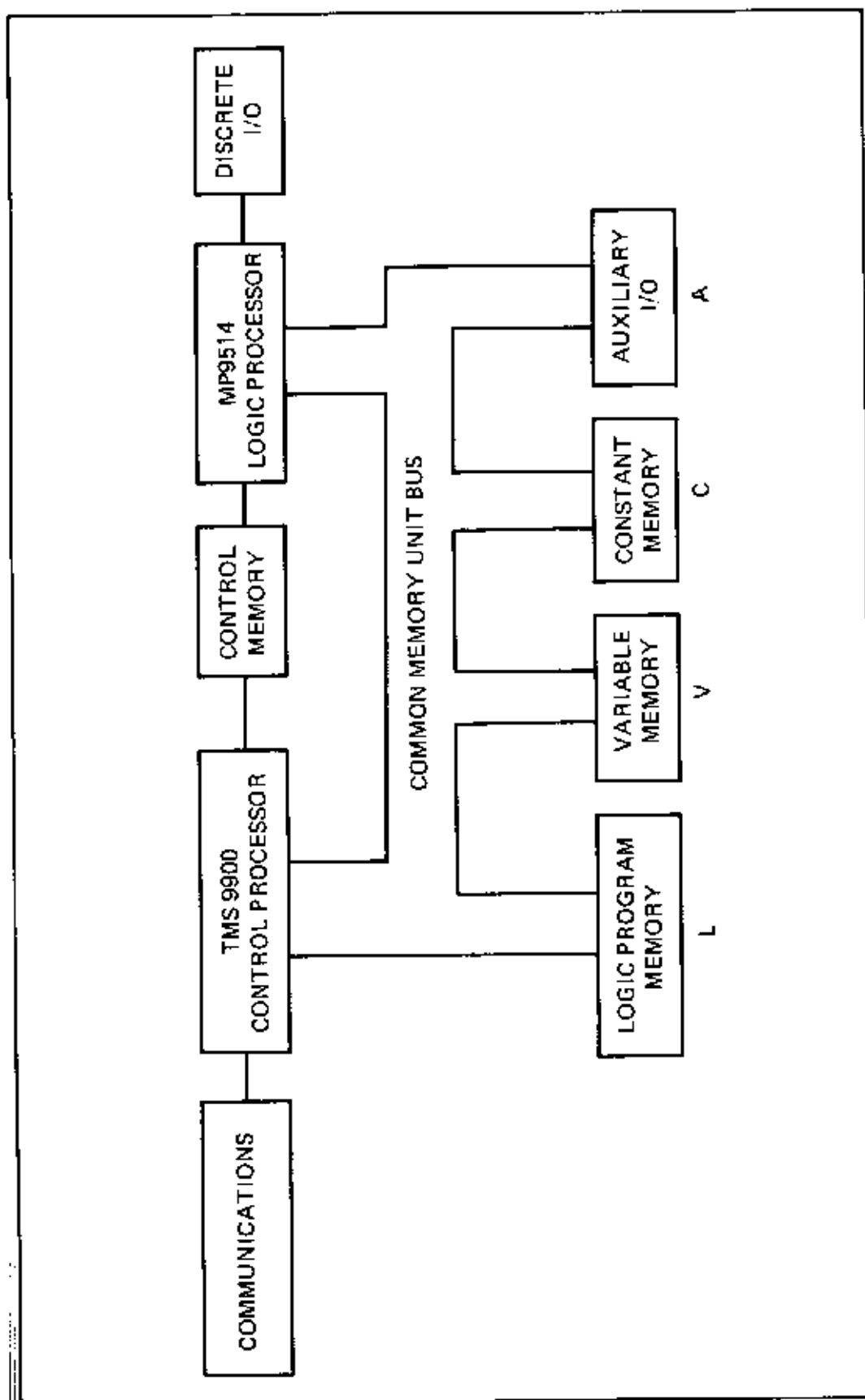


Figure 2 1A: Central Control Unit Block Diagram

## CHAPTER 3

### PREPARATIONS FOR USING THE PROGRAMMER

#### 3.0 GENERAL

Before a control sequence can be entered into the PM550 sequencer a ladder diagram similar to that shown in Figure 3.0A must be drawn.

#### 3.1 ASSIGNING INPUT/OUTPUT IDENTIFIERS TO LADDER DIAGRAM

To prepare the ladder diagram of Figure 3.0A for entry into the PM550 CPU, each of the elements should be given an internal PM550 identifier. The internal PM550 identifier is dependent upon how the external devices are wired to the 6MT and 7MT I/O interfaces. Referring to Figure 3.1A the limit switch LS1 is wired to the 6MT I/O Mounting base X0; therefore, the internal designation for LS1 will become X0. Correspondingly, the internal designation for LS2 will be X1. The internal identifier for pilot light No. 1 will be Y0, solenoid 3 will be Y1, etc. Refer to the 6MT I/O installation guide in the *Installation and Operation Manual* for additional information on terminal numbering. On the 7MT rack, the BCD thumbwheels are connected to channel 0 of the first module thereby giving the thumbwheel an input address of A0. The seven segment display is wired up to the 5th module of the 7MT I/O rack. It is connected to channel 0 giving it an address of A500. Refer to the *PM550 Installation and Operation Manual* for detailed numbering and addressing of the 7MT I/O. The assignments of inputs and outputs to terminals is completely arbitrary depending upon channel availability and user preference. Once an input is assigned, the element retains that identifier unless the input is moved to another terminal on the mounting base. A record of the terminal assignments is recommended as shown in Figure 3.1B and Figure 3.1C. These are memory allocations for ladder diagrams. There are two user data storage areas in the PM550, the variable data storage area (V) and the constant data storage area (C). In the sample program in Figure 3.0A the counter requires 1 constant memory location for its preset word and one variable memory location for its current



count. A preset word is assigned to the constant memory location C20 as shown in Figure 3.1D. The current word of the counter is assigned to the variable data memory area location V20 as documented on Figure 3.1E. Special function 2, the BCD TO BINARY conversion function, is stored in the constant memory area addresses C100 through C103 as shown in Figure 3.1D. The BINARY TO BCD conversion function SF1 also stored in the constant memory area is in location C104 through C107. The remaining variables from the add function are assigned to locations, since inadvertent changes to a special function or a loop table through indiscriminate use of data moves will result in unpredictable program behavior (Chapter 7).

### 3.2 REDRAWING OF LADDER DIAGRAM

Once the I/O assignments and internal memory storage locations are determined, the ladder diagram of Figure 3.0A may be redrawn as shown in Figure 3.2A. Note that the limit switch and pushbutton designations have been changed to the internal PM550 identifiers. Also the variable and constant storage locations have been documented on the ladder logic blocks. This form of the ladder logic diagram is often a more convenient way of documenting a PM550 program. Another noteworthy point is that once an input or output is assigned an internal identifier, that input or output may be used as many times as necessary in the ladder logic diagram. For example, in Figure 3.2A the control relay CR1 could be eliminated. All its contacts in a ladder logic diagram could be replaced with a designator Y1 and the logic would function identically.

### 3.3 DOCUMENTATION OF LADDER LOGIC PROGRAMS

The use of a ladder element storage record form as shown in Figure 3.3A is another form of program documentation that has proven to be invaluable. Since there is a one to one correspondence of ladder logic memory locations (L) and the elements in the ladder logic diagram, this form can be filled out before programming and used as a program entry aid.

### 3.4 PREPARATION FOR LOOP ENTRY

The steps prepared for loop entry into the PM550 are similar to those required for the ladder logic diagram. First, prepare a process flow diagram of the intended process. Then assign I/O identifiers to the input and output elements. Next generate the user memory storage records for the V and C areas. For further information and examples of this procedure see Chapter 7 - Programming Process Control Equations - Loops.

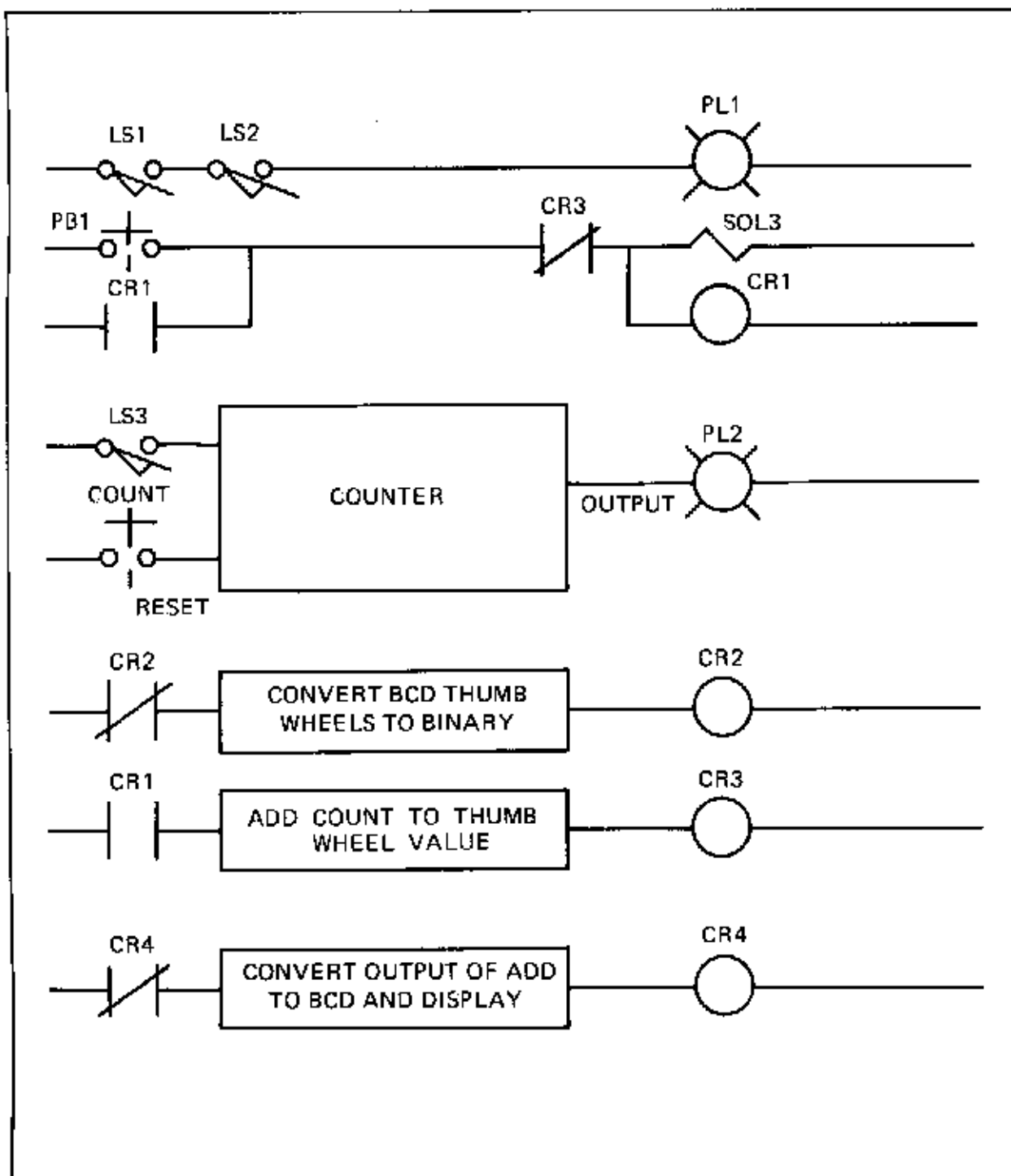
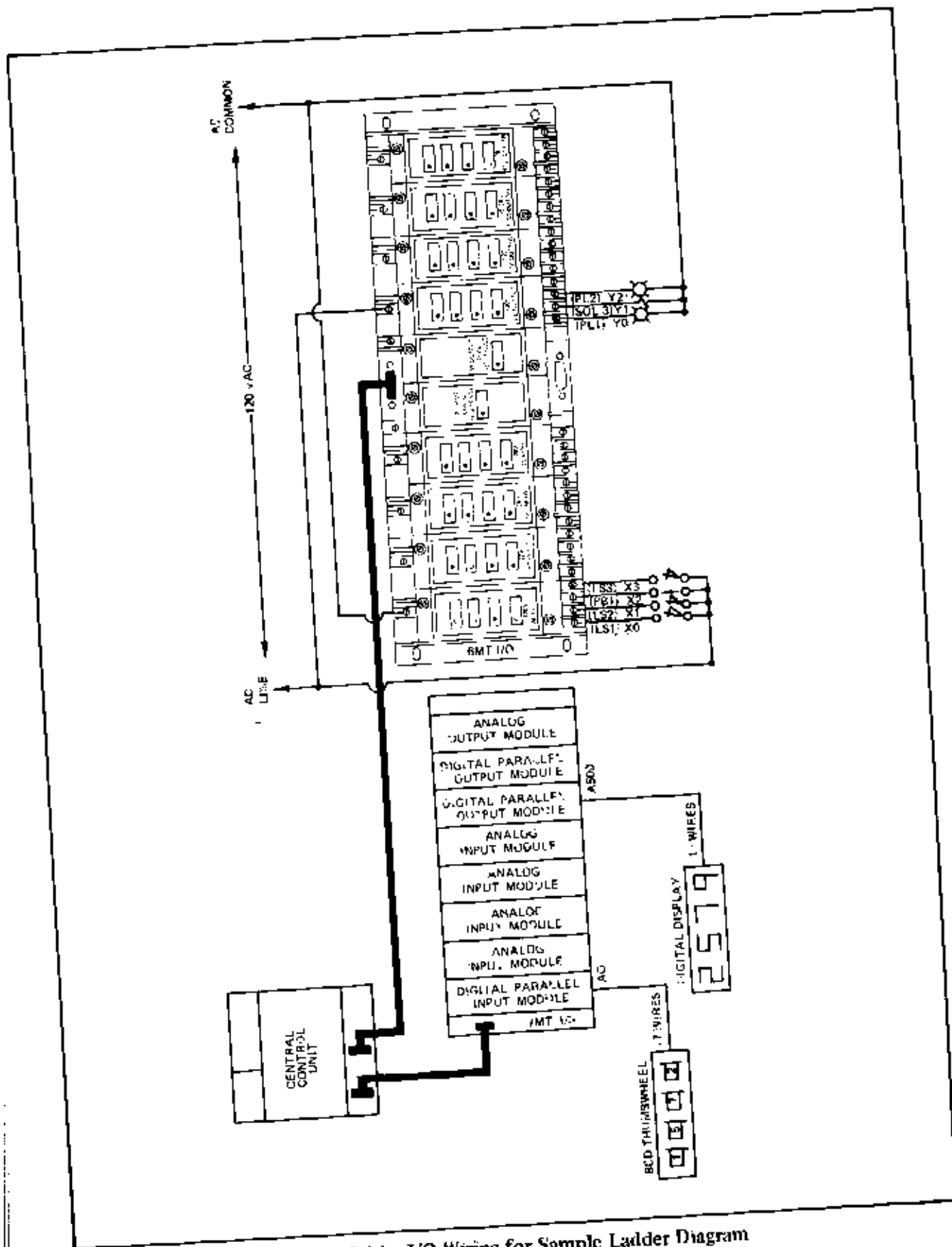


Figure 3.0A. Sample Ladder Diagram



# DISCRETE INPUT-OUTPUT RECORD FORM

BASE ASSEMBLY NO. 1

*TERMINAL DESIGNATION	TERMINAL NAME
X0	LS1 - PART DETECT
X1	LS2 - PART IN PLACE
X2	PB1 - CYCLE START
X3	LS3 - PART COUNT
X4	
X5	
X6	
X7	
X8	
X9	
X10	
X11	
X12	
X13	
X14	
X15	

BASE ASSEMBLY NO. 1

*TERMINAL DESIGNATION	TERMINAL NAME
Y0	PL1 - PART IN PLACE
Y1	SOL3 - CLAMP SOLFNOID
Y2	PL2 - END OF CYCLE
Y3	
Y4	
Y5	
Y6	
Y7	
Y8	
Y9	
Y10	
Y11	
Y12	
Y13	
Y14	
Y15	

\*X is used to designate discrete inputs and Y for discrete outputs. See Section 3.C.7 in the *Installation and Operation Manual*.

Figure 3.1B Input/Output Assignment Record for Sample Diagram

AUXILIARY INPUT/OUTPUT RECORD FORM				
MODULE	CHANNEL	ADDRESS	DESIGNATION	COMMENTS
0	0	A0	TW1	THUMBWHEEL INPUT NO. 1
0	1	A1		
0	2	A2		
0	3	A3		
5	0	A500	DS1	7 SEGMENT DISPLAY A1
5	1	A501		
5	2	A502		
5	3	A503		
6	0	A600		
6	1	A601		
6	2	A602		
6	3	A603		
7	0	A700		
7	1	A701		
7	2	A702		
7	3	A703		

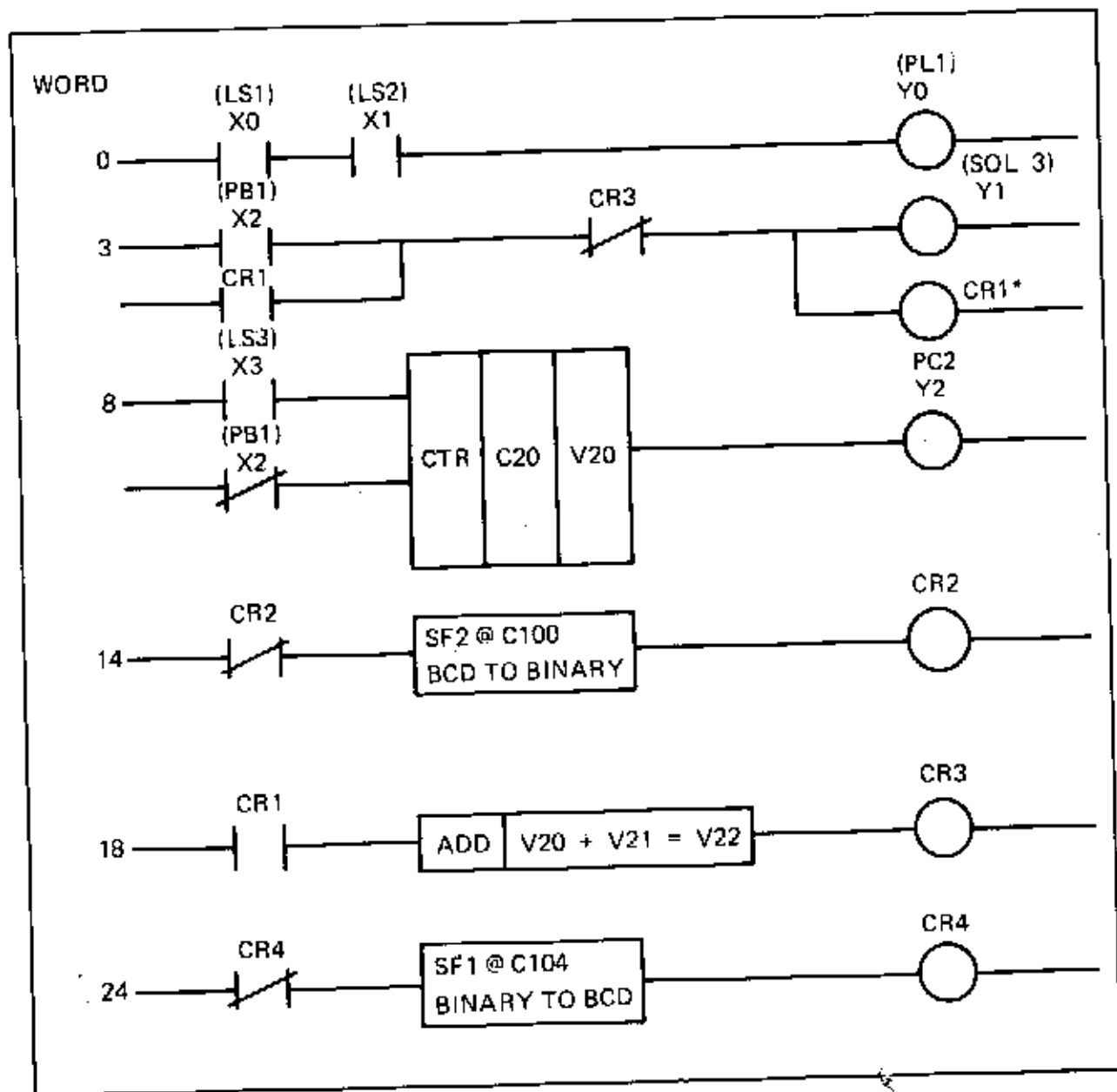
Figure 3.1C. Auxiliary Input/Output Record Form for Sample Diagram

USER MEMORY STORAGE RECORD		
AREA	LOCATION NUMBER	COMMENTS
C	20	COUNTER NO 1 PRESET WORD 157
C	21	
C	22	
C	23	
C	100	SF2 - BCD TO BINARY CONVERSION
C	101	
C	102	
C	103	
C	104	
C	105	SF1 - BINARY TO BCD CONVERSION
C	106	
C	107	
C	108	
C	109	

Figure 3.1D Constant (C) Memory Assignment Record

USER MEMORY STORAGE RECORD		
AREA	LOCATION NUMBER	COMMENTS
V	20	COUNTER NO 1 CURRENT WORD
V	21	BINARY OUTPUT OF SF2 @ C100
V	22	OUTPUT OF ADD AND BINARY INPUT TO SF1 @ C104
V	23	
V	24	
V	25	
V	26	
V	27	

Figure 3.1E Variable (V) Memory Assignment Record



\*Section 3.2 pointed out that this output could be eliminated since Y1 could be used as often as needed, in this case to replace CR1.

Figure 3 2A Redrawn Sample Ladder Diagram

# LADDER MEMORY STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENT.	I/O NAME	COMMENTS
0	STR	X	0	LS1	
1	AND	X	1	LS2	
2	OUT	Y	0	PL1	PART IN PLACE P.I.
3	STR	X	2	PB1	
4	OR	CR	1		
5	ADD NOT	CR	3		
6	OUT	Y	1	SOL1	CLAMP SOLLENOID
7	OUT	CR	1		
8	STR	X	3	LS3	COUNTER COUNT CONTACT
9	STR NOT	X	2	PB1	COUNTER PRESET CONTACT
10	CTR				PARTS COUNTER
11	C70				PRESET LOCATION
12	V20				CURRENT LOCATION
13	OUT	Y	2	PL2	END OF CYCLE P.L.
14	STR NOT	CR	2		
15	NSI				ST2 @ C100 BCD TO BINARY
16	C100				
17	OUT	CR	2		
18	STR	CR	1		
19	ADD				ADD V20 + V21 = V22
20	V20				
21	V21				
22	V22				
23	OUT	CR	3		OVERRIDE TERMINATE CYCLE
24	STR NOT	CR	4		

Figure 3.3A Ladder Memory Storage Record for Sample Diagram



## CHAPTER 4







### PROGRAMMER OPERATING INSTRUCTIONS

#### 4.0 GENERAL

Plug the line cord of the read/write programmer into a source of 120 volts 50/60-Hz. Plug one end of a PM550-911 96 read/write programmer cable into the RS232C connector on the back of the read/write programmer. Plug the other end into either one of the RS232C ports on the CCU (Port 1 or Port 2). Turn the power switch on the Read/Write programmer to the ON position and observe "READY" on the display of the read/write programmer. If READY is not displayed within 5 seconds, please refer to the Installation and Maintenance Manual.

#### 4.1 LADDER ENTRY KEYS

Refer to Figure 4.1A for the key location as described below.

- |   |   |
|---|---|
|  | used to enter the first ladder element of each new line of control sequence   |
|  | -- selects the counter function.  |
|  | - selects the timer function. <u>Timing increments are in tenths of a second.</u>                                       |
|  | selects the master control relay function. This function enables or disables all or any portion of the control sequence |
|  | - signifies that the element being entered is to be an output signal.   |
|  | - inverts (negates) the logical status of the selected input, output, or control relay contact.                         |

- ☐ OR — permits the operator to enter parallel contact combinations
- ☐ AND — permits the operator to enter series contact combinations
- ☐ JUMP — selects the JUMP function. Causes the specified number of outputs to be skipped when not activated
- ☐ COMP — selects the compare function. Compare two memory locations for equality or greater than or equal to
- ☐ MOVE — programs the data move function. Transfers the contents of one memory location to another.
- ☐ EOS — programs the conditional or unconditional end of scan. EOS when active causes the logic scan to return to location 0 and restart.
- ☐ SF — programs special function instruction in ladder logic. Also allows programming of special function data in user memory.
- ☐ FIND — searches ladder logic memory for desired function.
- ☐ V — selects the variable data area of user memory
- ☐ C — selects the constant data area of user memory.
- ☐ A — selects auxiliary I/O and change I/O to input output module location
- ☐ X — signifies that the element being entered is an input signal from a device connected to the 6M150 mounting base.
- ☐ Y — signifies that the element being entered is an output signal which will be applied to a device connected to the 6M150 mounting base
- ☐ CR — there are 512 control relays inside the CCU. These are internal storage elements that are similar to control relays. Each CR can be referenced as many times as desired. The first 256 CR's are non latching. The last 256 CR's latch and retain their status on power failure as long as battery backup is available.
- ☐ NO — answer prompts ending with a question mark ~~affirmatively~~ **NEGATIVELY**
- ☐ YES — answer prompts ending with a question mark ~~negatively~~ **AFFIRMATIVELY**

- 0 - 9 – the digit keys are used to select input/output or control relay identification numbers, select user memory locations, or to enter numerical data into storage locations, loops, or special functions. A decimal point . used with a user memory location signifies a noninteger (floating point) storage location. Also to be used in entering non integer numbers.
  
- ÷ – used in the user math special function to enter the mathematical divide operation.
  
- X – used in the user math special function to enter in mathematical multiply operation. Also used to enter the exponent in the scientific notation entry mode.
  
- – used to program the ladder logic subtract operation or to enter the mathematical subtract operation in the math special function or change the sign of a numeric value.
  
- + – used to program the ladder logic addition operation or to enter the mathematical add operation in the user math special function.
  
- = – used to assign memory address for the results in the ladder logic add and subtract functions. Also used to assign the memory address for the results in the math special function.
  
- INSRT – causes the instruction keyed in to be inserted in the memory address displayed and all subsequent instructions to be moved down in memory.
  
- ENTER – causes the instruction or instructions programmed to be entered into memory.
  
- STEP – causes the memory address display to increment by one. Also used to 'step' through multiply word instructions – special functions, loops, or auxiliary functions.
  
- BSTEP – causes the display memory address to be decremented by one.
  
- READ – causes the read/write programmer to enter the READ mode. In READ mode the programmer reads the displayed memory address continuously. READ mode is exited by clear entry or clear.
  
- ENTER – causes the currently read instruction to be deleted and subsequent instructions to be moved up to fill the gap. NOOPS (NO Operation) is placed at end of memory.
  
- CLR – clears the introduction entry leaving just the memory location displayed. Used to modify a location that has just been read.

- CLR** — causes the current procedure to be aborted and returns the programmer display to READY.

## 4.2 LOOP AND AUXILIARY KEYS

- LOOP** — used to enter the loop entry mode on read/write programmer.

- AUX** — used to access auxiliary functions.

## 4.3 DISPLAY

The twenty character alpha-numeric display of the read/write programmer allows verification of keys pressed and read back of programs using the same mnemonics as the keys. It also allows full English prompting in the math, loops, special functions, and auxiliary functions (see Figure 4.3A).

## 4.4 POWER FLOW

The power flow lamp shows the status of the logic flow at the current displayed memory location. If the power flow lamp is on, it is the equivalent of current flow at that point in the ladder logic diagram. If the power flow lamp is off, there is no current flow at that point and there will be no output at the end of that logic line. The power flow lamp is meaningless when displaying user memory data storage locations, special functions, loops or auxiliary functions.

### NOTE

The power flow lamp on the Programmer is updated approximately every 1/2 second. Therefore, it may not follow rapidly occurring events. A "Real Time" power flow lamp is provided on the CCU that is capable of following any event.

## 4.5 BEEPER

The beeper will sound when an invalid key is pressed or when a memory write error is detected. After the **ENVR** key is depressed, the CCU writes the data to memory and then reads it back to verify data integrity. If the data written does not match the data read back, the beeper will sound.

POWER FLOW		STR	CTR	TMR	MCR	OUT	NOT	OR	AND	JUMP	
COMP	FIND	V	X		7	8	9	-		NSRT	DLTE
MOVE	LOOP	C	Y		4	5	6	X		ENTR	CE
EOS	L	A	CR		1	2	3	_		STEP	BSTP
SF	AUX	YES	NO			0	=	+		READ	CLR

Figure 4 1 A Read/Write Programmer Key Display

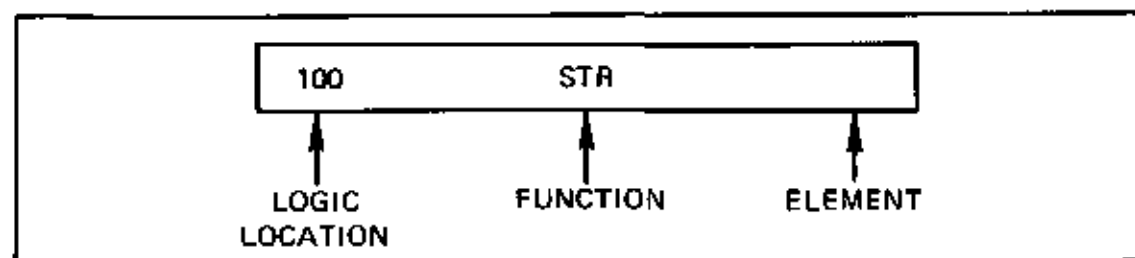


Figure 4 3 A Typical Ladder Location Display

## CHAPTER 5

### HOW TO ENTER LADDER DIAGRAMS

#### 5.0 GENERAL

The steps to the successful application of the PM550 control system are:

- Define Input/Output assignments — as described in Chapter 3
- Develop Ladder Logic or Boolean Equations
- Define PM550 Program Listing
- Enter Program into PM550 Read/Write memory using programmer

This chapter provides a detailed description of each PM550 Ladder Diagram function along with examples of corresponding key entry sequences.

With the PM550 CCU, Power Supply, programmer and simulator connected as shown in Figure 5.0A each example should be entered and verified. Understanding the PM550 functions and key entry sequences will then aid you in developing the program listing and entry procedure for your application.

To aid you in the correct entry of each function, the programmer will de-activate keys which could cause the function to be entered incorrectly.

If the programmer display does not respond to a key which has been pressed, check the function description in this manual to verify the correct key sequence.

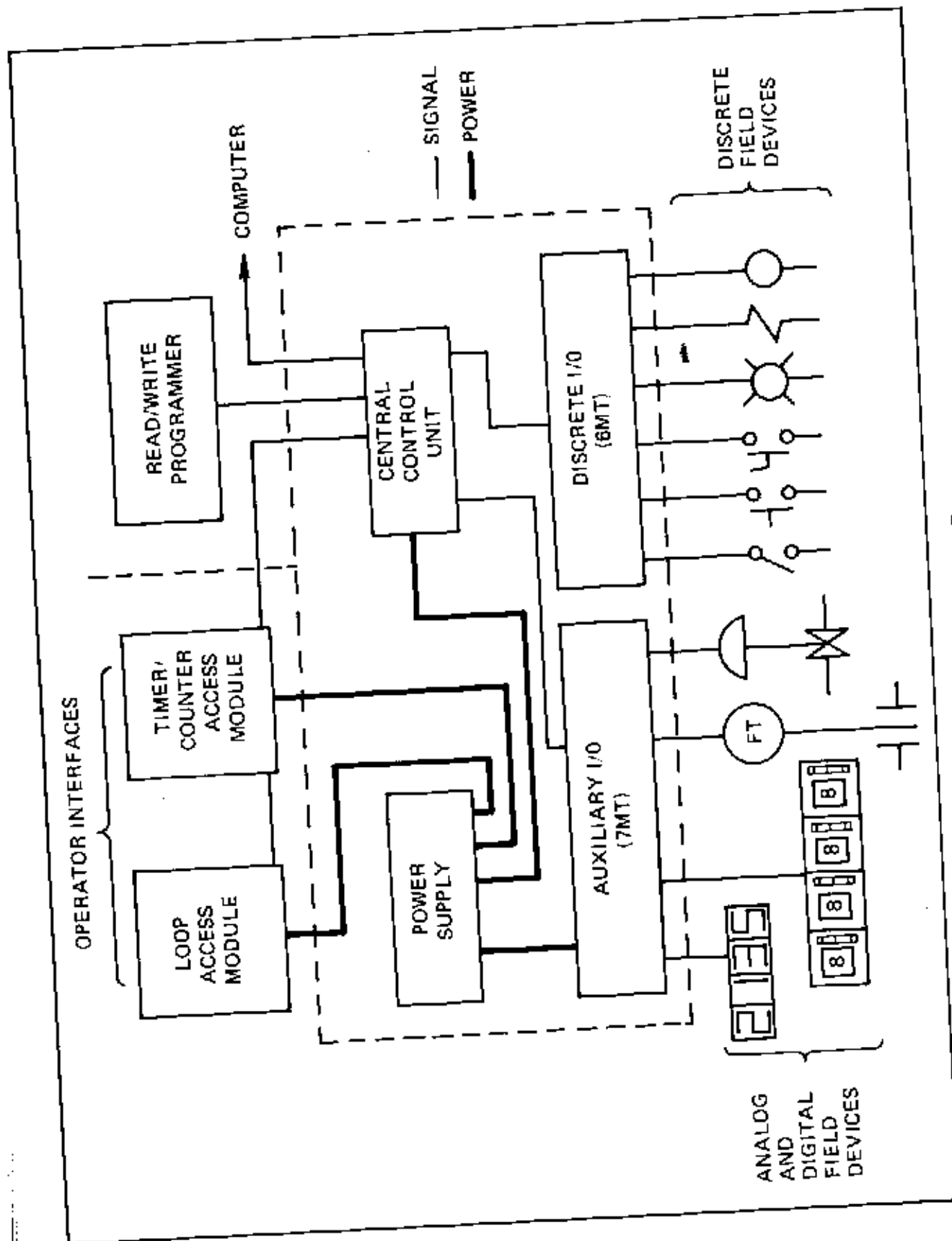


Figure 5 0A. System Block Diagram

## 5.1 CLEARING MEMORY

It is a good practice to clear (erase) the read/write memory of the CCU prior to entering ladder logic through the programmer.

Chapter 2 of this manual describes 3 areas of memory which are available for program entries:

L Ladder Logic Memory (2048 or 4096 Locations)









V Area (1024 Locations)

C Area (1024 or 2048 Locations)

In addition the X, Y, and CR image registers are memory areas which hold the most recent state (on or off) of the 256 inputs, 256 outputs, and 512 control relays. Auxiliary Function #1 (AUX = 1, see Section 8.1) can be used to clear these memory areas. To initiate a memory clear press the following keys in the order given. Note that the programmer display provides prompting messages to aid you in clearing the correct memory area.

### NOTE

The Start-up/Run Switch on the CCU must be in Start-up memory to clear.

KEY SEQUENCE	DISPLAY	NOTE
	READY (START)	
 	AUX = 1	
	CLEAR L ?	
	PRESS ENTR TO CLEAR	Pressing FNTR will clear the 2048 or 4096 L locations, plus the 1024 Image Register Locations.
	*WAIT*	
	CLEAR V ?	
	PRESS ENTR TO CLEAR	
	*WAIT*	
	CLEAR C ?	



KEY SEQUENCE	DISPLAY	NOTE
<input type="checkbox"/> YES	PRESS ENTR TO CLEAR	Pressing ENTR also will clear all loop tables.
<input type="checkbox"/> ENTR	*WAIT*	ALL AREAS of memory are now cleared.
	READY	

Note that the display requests a yes or no answer on each memory area. To leave a memory area intact press ☐ NO and the programmer will step to the next question. Pressing ☐ CLR at any time causes the programmer to leave the AUX 1 Function and return to the READY state.

## 5.2 SINGLE CONTACT – PROGRAMMING EXERCISE NO. 1

Figure 5.2A shows a Ladder Diagram in which one output is controlled by one input. Limit switch LS2 is represented by X1 because it is connected to mounting base input terminal X1. Solenoid Sol 3 is represented by Y1 because it is an output and is connected to mounting base output terminal Y1. Limit switch LS2 and solenoid Sol 3 are examples of a discrete (on/off) input and output which would be field wired to the GMT50 I/O mounting base. To enter the ladder diagram of Figure 5-1 push the following keys in the order given.

- ☐ CLR – Display indicates READY. This key clears a previous entry from the programmer display this key does not clear the CCU.
- ☐ 0 – Display indicates 0, the first location available for storing ladder diagram elements.
- ☐ STR – This key is used to enter the first element of every new line in a ladder diagram, note that the display now indicates:

0 STR

- ☐ X – This key signifies that the element to be entered refers to an input device connected to a mounting base. The display now indicates:

0 STR X

- 1 Selects input terminal No. 1 to which LS2 is connected. Display indicates:

0 S1R X1

- ENTR - Enters the above information (S1R X1) into CCU ladder element storage location 0. If an error occurs, the beeper will sound, otherwise, the logic address will increment to 1.
- dir - Signifies that element about to be entered is an output
- Y - Signifies that output is to be made to a mounting base. (Remember that a CR is also an output, but is internal to the CCU).
- 1 - This represents the specific terminal to which the output signal will be applied. The display now indicates:

1 OUT Y1

- ENTR - Enters all information shown on the display into location 1. Verify that when X1 closes, an output occurs at Y1 by closing switch X1 on 5TI4100 Simulator. Lamp Y1 will light. This verifies that all entries are correct and that the control sequence performs as planned. The Start-up/Run Switch must be in Run to verify the program.

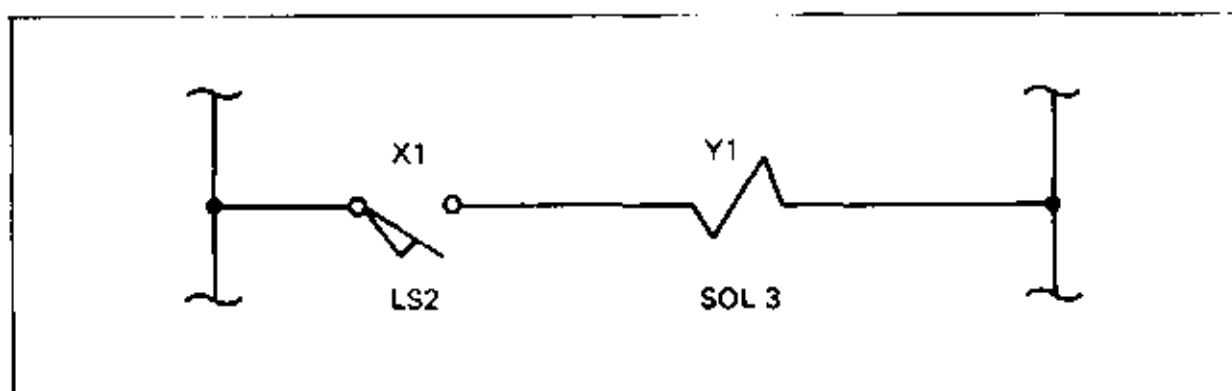


Figure 5-2A Ladder Diagram for One Output Controlled by One Input

### 5.3 LADDER ELEMENT (L MEMORY) STORAGE RECORD

Assignments of CCU Ladder Element (L Memory) Storage Location numbers should be recorded on the Ladder Element (L Memory) Storage Record form shown in Figure 5.3A. There are 2048 or 4096 (including zero) Ladder Element (L Memory) Storage Location numbers available, and attempting to find an entry error in a large control sequence would be prohibitive without some record of what was entered into the CCU. Note that the elements of the ladder diagram of Figure 5.2A have been recorded in the form of Figure 5.3A. (see Appendix A-1)

LADDER ELEMENT (MEMORY) STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFIER	I/O NAME	COMMENTS
0	STR	X	1	LS2	
1	OUT	Y	1	SOL3	
2					
23					
24					

Figure 5.3A Ladder Element (L Memory) Storage Record Form  
(Exercise No. 1)

### 5.4 SIMPLE SERIES AND CONTACTS - PROGRAMMING EXERCISE NO. 2

Two elements represented by X1 and X2 are shown connected in series in the ladder diagram of Figure 5.4A. Both input devices must close before an output will occur. To enter the ladder diagram of Figure 5.4A press the following programmer keys:

CLR ACN 1 STOP YES

FNTR CLR - Pressing the **FNTR** key executes the clear ladder element (memory) storage command and all 2048 or 4096 locations are cleared. This operation also clears the discrete (Y and CR) image register. Display now indicates

READY (START)

**3** - Selects ladder element location No. 3 to receive element about to be entered. (Storage locations 0, 1, and 2 have arbitrarily been left unassigned).

**STR** - Pressed to define the first element of each new ladder line

**X** - Signifies that the element to be entered is an external input.

**1** - Selects specific input connected to input terminal No. 1 of mounting base

Display now indicates

3 STR X1

**ENTER** - Enters STR X1 into location L3. Until the **ENTER** key is pressed, nothing is entered into the CCU. The storage location is automatically incremented to 4 after the data is written.

**AND** - This key is used to join elements in series, and can be employed as many times as required in a single ladder line.

**X** - Signifies that the element to be entered is an external input

**2** - Selects specific terminal to which input will be made. The device connected to the third mounting base input terminal will receive an input signal. (The first input and output terminals are X0 and Y0).

Display indicates

4 AND X2

**ENTER** - Enters **AND** **X** **2** into location L4.

**OUT** - Signifies that next element is an output.

**Y** - Signifies that the output is external and will be made to the GMT50 mounting base

**2** - Selects specific terminal to which output will be made; the device connected to the third mounting base output terminal will receive the output signal.

**ENTER** - Enters OUT Y2 into the CCU

Verify that the ladder diagram of Figure 5 4A has been entered correctly by switching CCU to RUN; set switches X1 and X2 on the simulator to CLOSED. Lamp Y2 will light to indicate the control sequence is valid. Figure 5 4B shows how the ladder diagram of Figure 5 3A would be listed on the ladder element (L Memory) storage record form.

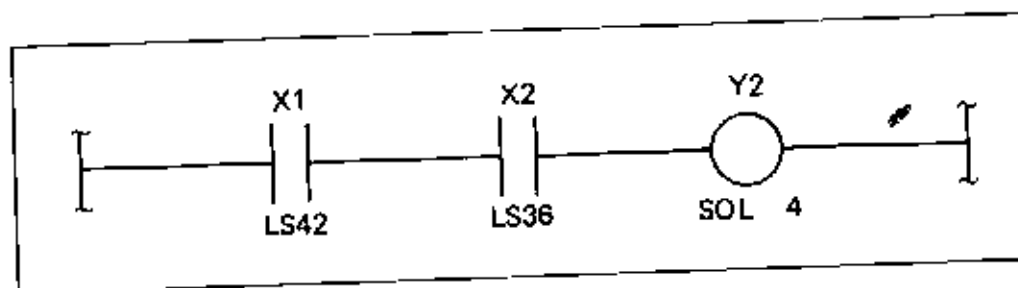


Figure 5 4A. Ladder Diagram for Two Input Elements in Series

LADDER ELEMENT (L MEMORY) STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFIER	I/O NAME	COMMENTS
0					
1					
2					
3	STR	X	1	LS42	
4	AND	X	2	LS36	
5	OUT	Y	2	SOL4	
6					
7					

Figure 5 4B. Ladder Element (L Memory) Storage Record  
(Exercise No. 2)

## 5.5 USE OF KEY TO CHECK CONTENTS OF MEMORY

The  key is used to display the content of any CCU ladder element storage location to verify that the control sequence has been correctly entered. Check the content of each CCU ladder element storage location against that information listed on the ladder element storage record. Contents are checked by reading the information indication on the programmer display. To observe the  operation press the following programmer keys:

- Pressing  key returns the programmer to the ready mode, the first location to be checked can now be defined.
- Selects ladder element storage location L3.
- Press this key and observe that the display indicates

3 STR X1

We see that in ladder element storage location L3 we have entered STR X1. Refer to Figure 5.4B and observe that this is the correct entry for location number 3. Now press the  key once, the display now indicates







4 AND X2

Check this information against the entry in ladder element location L4 as shown in Figure 5.4B. Once the  key has been depressed, the programmer continues to read the contents of the ladder element location defined on the display. The  key can then be used to increase the location number while the programmer reads and displays each location's contents. Pressing any key on the programmer except the , , or  keys will discontinue the read mode.





Referring to Figure 5.4B: Press  once, then press  until location No. 6 is displayed. Note that the display contains a NOOP. This is because no information was entered into this ladder element storage location. It must be pointed out that checking the CCU contents does not have to begin at location 0, it can begin at any location; for example, if you wanted to check the contents of CCU ladder element location No. 107 (logic memory) you would first press the  key.


Then you would press the digit keys ,  and . Next you would press  and read the display to determine what kind of element is contained in that memory location.

## 5.6 USE OF BACKSTEP —


Just as the  key is used to increase the location number, the  Backstep key is used to decrease the location number of logic, constant or variable memory or X's, Y's, or CR's. Press the  key followed by the  key, selecting location No. 5. Press the  key and observe the contents of location No. 5 on the display. Press the  key and observe the location number changes from 5 to 4, and the contents of location No. 4 are displayed.



## 5.7 USE OF THE POWER FLOW LAMP


The power flow (PF) lamp is used as a troubleshooting aid. By observing its status when the  key is pressed, the accumulative status of the PF at a particular location can be observed. To illustrate its use in the previous example, press the  ,  , and  keys. The display will show STR X1, in location No. 3. Observe the power flow lamp as you close the simulator switch X1. The power flow lamp will come on.\* This indicates that the CCU has sensed the switch closure, stored the condition of X1 in the image register, and placed the status of X1 in the power flow in accordance with the instruction executed.

Press the  key and observe the display. We are now looking at location 4 which contains AND X2. The power flow lamp shows the result of combining X2 and X1 in an AND fashion in the PF. The lamp will come on if both X2 and X1 are on. Thus, the power flow lamp can be used much as a volt meter is used in relay control to determine that contacts are closed to produce an output.

Note that because the power flow is being read in the middle of a ladder line, whenever X1 is off, the true status of X2 cannot be determined. To determine if X2 is open or closed independent of X1, press the following keys:

 — Display indicates READY (RUN)

  — Display indicates X2 with no ladder element location number

 — Display indicates X2 = 0. Close X2 and observe that the display changes to X2 = 1.

This approach can be used to read the independent status of any input (X) or output (Y, CR).

### \*NOTE

The power flow lamp on the Programmer is updated approximately every 1/2 second. Therefore, it may not follow rapidly occurring events. A "Real Time" power flow lamp is provided on the CCU that is capable of following any event.

## 5.8 USE OF THE **[FIND]** FUNCTION

The **[FIND]** key is used to enlist the aid of the programmer in locating a specific ladder element within the 2048 (or 4096) ladder element storage locations. To illustrate its use in the previous example, press the **[CLR]**, **[0]**, **[X]**, **[2]**, and **[FIND]** keys. The programmer will begin to search through memory starting at location 0 and proceeding until a location is identified with X2. The display then indicates the complete contents and location number in which X2 is found.

4 AND X2

To search for additional locations which contain X2 continue to press the **[FIND]** key until all memory has been searched.

Each ladder element consists of an opcode and an identifier. For example AND X2

- (1) AND    OP CODE
- (2) X      ELEMENT TYPE
- (3) 2      ELEMENT IDENTIFIED

A search may be conducted for a complete word (AND X2), the opcode (AND), or the identified element (X2).

When a search is conducted for an element which is not contained in the ladder element locations, the display will indicate NOT FOUND.

The storage location preceding the element to be FOUND is the starting location for the search. Start with location 0 to search all logic memory.

The following elements are forbidden arguments for a FIND: NOT, X, Y, CR, V, C, AND A. A location number must be present to execute the FIND which searches the rest of memory starting with the location keyed in.

The Read/Write Programmer remembers the FIND argument until **[CLR]** or another FIND Sequence is entered. Therefore, a location can be found and modified and when **[FIND]** is pressed again, the next location found.

## 5.9 CHANGING A LADDER DIAGRAM AFTER ENTRY INTO THE CCU

Once a ladder diagram is entered into the CCU, adding an element, deleting an element, or changing an element OP CODE may be required to effect a ladder diagram change.



### 5.9.1 Inserting New Elements

The programmer **NSRP** key is used to insert a new ladder element into a ladder line that has been previously entered. Figure 5.9.1A shows how the ladder element location assignments for the series contact example will change when switch X10 is added in series to the ladder line. To add X10 the following keys are pressed:

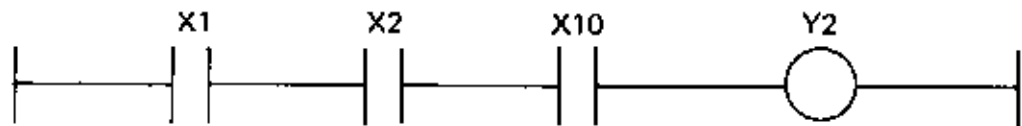
- CLR** — Display indicates READY (RUN)
- 5** — Defines the ladder element location in which X10 will be stored
- AND** **X** **1** **0** — X10 is to be placed in series with X1 and X2. Display now indicates  
5 AND X10

- NSRP** — The element AND X10 is entered into location 5. The previous contents of location No. 5 (OUT Y2) are saved and written into location 6. This upward shifting of locations continues through location 2047 (or 4095). The element in location 2047 (or 4095) is lost.

### 5.9.2 Deleting Elements from a Ladder Line

The programmer **DL-L** key is used to remove an element from a ladder line that has been previously entered. Referring to Figure 5.9.2A element X10 can be removed from the line by pressing the following keys:


- CLR** — Display indicates READY (RUN)
- 5** — Defines the location of the element to be removed.
- READ** — Display indicates 5 AND X10
- DLTE** — Element AND X10 erased, and the contents of location 6 are downward shifted into location 5. The display now indicates 5 OUT Y2. This downward shifting continues through location 2047 (or 4095). Location 2047 (or 4095) becomes a NOOP

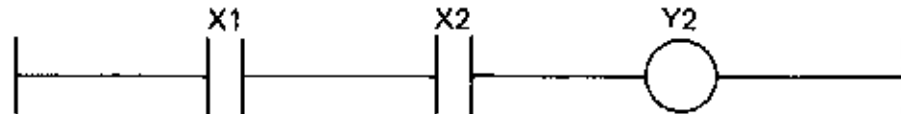


### LADDER ELEMENT STORAGE RECORD

#### LADDER ELEMENT STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFICATION	I/O NAME	COMMENTS
0					ELEMENT X 10 ADDED BY USE OF INSERT FUNCTION
1					
2					
3	STR	X	1	LS42	
4	AND	X	2	LS36	
5	AND	X	10		
6	OUT	Y	2	SOL4	


Figure 5.9.1A Use of  Insert Function



### LADDER ELEMENT STORAGE RECORD

#### LADDER ELEMENT STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFICATION	I/O NAME	COMMENTS
3	STR	X	1	LS42	ELEMENT X10 IS ERASED AND ELEMENT OUT Y2 IS DOWNWARD SHIFTED TO LOCATION NO. 5
4	AND	X	2	LS36	
5	OUT	Y	2	SOL4	

Figure 5.9.2A Use of  Delete Function

### 5.9.3 Changing an Element

Any previously entered element can be changed by addressing the ladder element location, defining the new element, and entering the new element using the **ENTR** key. Refer to Figure 5.9.3A and press the following keys in order given:

- CLR** – Display indicates READY (RUN)
- 4** – Defines location to be changed
- AND** **X** **1** **2** – Defines the new element to be entered into location 4
- ENTR** – Automatically erases the previous statement (AND X2) and enters the new statement (AND X12) into location 4

Verify that the new ladder line is functional by closing switches X1 and X12 on the simulator; lamp Y1 will light

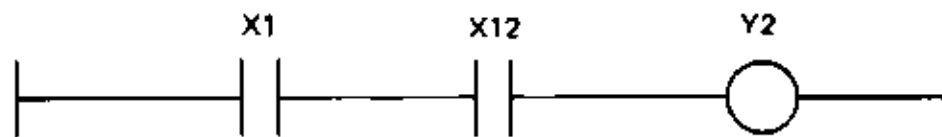
This method can also be used to erase an element from a line. Press the following keys

- CLR** – Display indicates READY (RUN)
- 4** – Defines location to be changed
- ENTR** – Automatically erases the previous statement (AND X12). Since no new element was defined, a NOOP is entered in location 4.

### 5.9.4 Use of Clear Entry **CE** Key

The **CE** key can be used at any time to clear the programmer display without clearing the displayed location number. This can be useful in correcting an error during definition of an element or in editing an existing program.

Refer to Figure 5.9.4A and press the **CLR**, **5**, and **READ** keys. The display now indicates 5 OUT Y2. Pressing the **CE** key clears OUT Y2 from the display (NOT FROM UCU MEMORY) so that a new entry can be made. Press **OUT**, **Y**, **3**, **ENTR**. Location 5 now contain element OUT Y3 which can be verified by closing switch 1 on the simulator and noting that element Y3 lights.

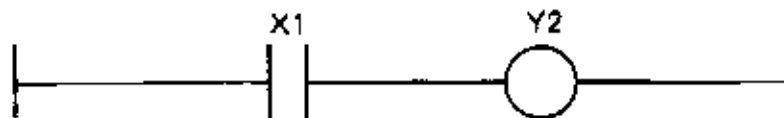


LADDER ELEMENT STORAGE RECORD

LADDER ELEMENT STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFICATION	I/O NAME	COMMENTS
3	STR	X	1		
4	AND	X	12		AND X12 FRASD. AND X12 WRITEN IN
5	OUT	Y	2		
6					

Figure 5.9 3A Changing an Element in a Ladder Diagram



LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFICATION	I/O NAME	COMMENTS
3	STR	X	1		
4					AND X12 IS FRASD FROM THIS MEMORY LOCATION
5	OUT	Y	2		
6					

Figure 5.9 4A Changing an Element in Ladder Diagram Using 

## 5.10 SIMPLE PARALLEL OR CONTACTS PROGRAMMING EXERCISE NO. 3

In the ladder diagram shown in Figure 5.10A, 3 input devices are connected in parallel. Figure 5.10B shows how the control sequence would be listed prior to entry into the sequencer.

To enter this ladder diagram into the CCU, press the following keys in the order given (CCU must be in START-UP mode.)

CLR ALX 1 STEP YES

FNTR -- Erases all ladder elements locations.

LOC 5 -- Selects CCU ladder element storage location 5.

STR -- Used to enter the first element of a ladder line

X Signifies that the element to be entered refers to an input line connected to the 6MT50 Mounting Base.

3 Digit key selects specific input terminal of mounting base

Display now indicates STR X3 in location No. 5

FNTR -- Enters the element STR X3 into location 5 and increments the storage location to 6.

PR This element type key signifies that the next element entered will be connected in parallel with the element previously entered. Any number of elements may be entered in parallel.

X 4 -- (Press X key first and then digit key 4). Signifies that element to be entered refers to a signal at input terminal No. 4 on the mounting base

FNTR -- The FNTR key is used to enter the element into the CCU; and to select the next ladder element storage location

OR X 9 FNTR Pressing each of these keys separately enters elements OR X9 and selects the next ladder element storage location number.

OUT Y 6 FNTR Pressing these keys separately enters element Y6, and signifies that element Y6 is an output signal to be applied to output terminal No. 6 of the mounting base.

Verify that the ladder diagram of Figure 5.10A has been entered correctly by closing and opening simulator switches X3, X4, and X9 one at a time; as each switch is closed lamp Y6 will light. To verify CCU must be in RUN mode.

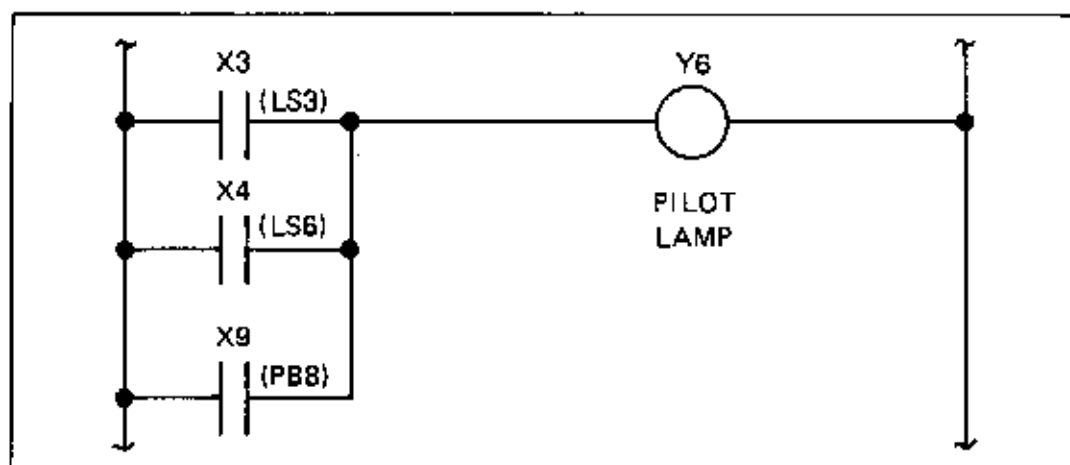


Figure 5.10A. Ladder Diagram for Programming Exercise No. 3

LADDER ELEMENT (L MEMORY) STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFIER	I/O NAME	COMMENTS
0					
1					
2					
3					
4					
5	STR	X	3	LS3	
6	OR	X	4	LS6	
7	OR	X	9	PB8	
8	OUT	Y	6	PILOT LAMP	CLOSURE OF X3, X4, OR X9 WILL ENERGIZE Y6 TURNING ON PILOT LAMP
9					

Figure 5.10B. Ladder Element (L Memory) Storage Record  
(Exercise No. 3)

## 5.11 MULTIPLE OUTPUTS – PROGRAMMING EXERCISE NO. 4

Figure 5.11A shows a ladder diagram in which the first two elements are connected in parallel to control output Y7; then the parallel combination is connected in series with a single element. Three outputs are shown. Either X6 or X7 must close to energize Y7; Either X6 or X7 must close and X10 must close before the dual outputs Y5 and Y11 will energize. To prepare for entering the ladder diagram of Figure 5.11A press the following keys.

**CLR** **AUX** **1** **STEP** **END**

**ENTR** – We have cleared out all information contained in the CCU Ladder Element (L Memory) Storage Location area. New information can now be assigned to those areas we have used before. Note that we could have elected to enter new information into the CCU without clearing the Ladder Element (L Memory) Storage Location area. By entering into an area already used the new entry replaces the previous entry.

Figure 5.11B shows how the ladder diagram of Figure 5.11A appears when its elements are listed on the Ladder Element (L Memory) Storage Record. To enter the ladder diagram into the CCU, which is now clear of all previous information, press the following keys (explanation of the key action follows below).

**CLR** **0** **STR** **X** **7** **ENTR**  
**OR** **X** **6** **ENTR**  
**OUT** **Y** **7** **ENTR**  
**AND** **X** **1** **0** **ENTR**  
**OUT** **Y** **5** **ENTR**  
**OUT** **Y** **1** **1** **ENTR**

With CCU in RUN, now close switches X7 and X10 on the Simulator, lamps Y5, Y7 and Y11 will light. Open switch X7 and close switch X6; again lamps Y5, Y7 and Y11 light. Open X10 and Y7 will remain on.

Look again at the key sequence; note that element X7 was connected in parallel with element X6 using the **OR** element key. Output Y7 was generated using the **OUT** element key. The combination of X6 OR X7 was connected in series with element X10 using the **AND** element type key. Both outputs were connected in parallel by repeated use of the **OUT** element type key. As many outputs as are required may be connected in this manner. Outputs may be placed at any point in the line.

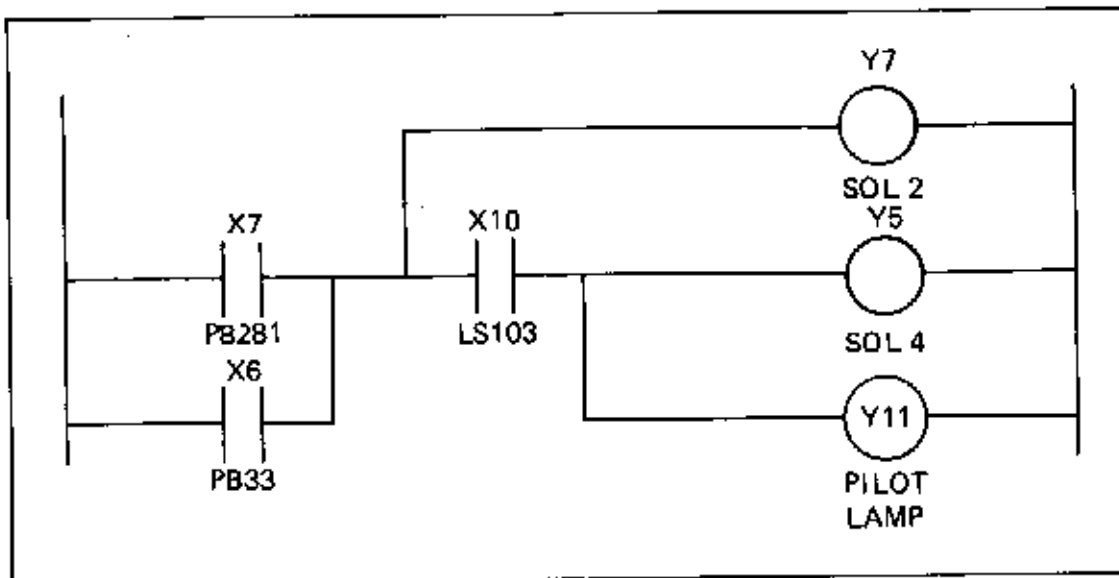


Figure 5.11A. Ladder Diagram for Programming  
(Exercise No. 4)

LADDER ELEMENT (L MEMORY) STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFIER	I/O NAME	COMMENTS
0	STR	X	7	PB281	
1	OR	X	6	PB33	
2	OUT	Y	7	SOL 2	
3	AND	X	10	LS103	
4	OUT	Y	5	SOL 4	
5	OUT	Y	11	PILOT LAMP	
6					

Figure 5.11B. Ladder Element Storage Record  
(Exercise No. 4)



## 5.12 PARALLEL SERIES ☐ OR ☐ STR CONTACTS USING ☐ OR ☐ STR FUNCTION — PROGRAMMING EXERCISE NO. 5

The control sequence for the ladder diagram shown on Figure 5.12A is given in Figure 5.12B. In Ladder Element (L Memory) Storage Location 4 an OR STR function (made by pressing two keys) is entered.

To understand the use of the OR STR function, enter the ladder diagram of Figure 5.12A as follows:

☐ CLR ☐ AUX ☐ 1 ☐ STEP ☐ YES

☐ ENTER — To clear programmer and CCU

☐ CLR ☐ 0 — Ladder Element (L Memory) Storage Location is ready for assignment, as shown by the display.

☐ STR ☐ X ☐ 3 ☐ ENTER — These keys enter element and OP code STR X3 and select next storage location.

☐ AND ☐ X ☐ 4 ☐ ENTER — These keys connect elements X4 and X3 in series and select the next storage location; the result of combining element X3 and X4 in series is placed in the PF.

☐ STR ☐ X ☐ 2 ☐ ENTER — These keys store X3 ANDed (put in series) with X4 in Push Down Stack 1, enters element X2; and selects the next storage location.

☐ AND ☐ X ☐ 5 ☐ ENTER — These keys connect elements X2 AND X5 in series within the PF and select the next storage location.

☐ OR ☐ STR ☐ ENTER — These keys, when pressed, provide the instruction that connects the contents of PDS1 in parallel with the contents of PF, i.e., to generate.

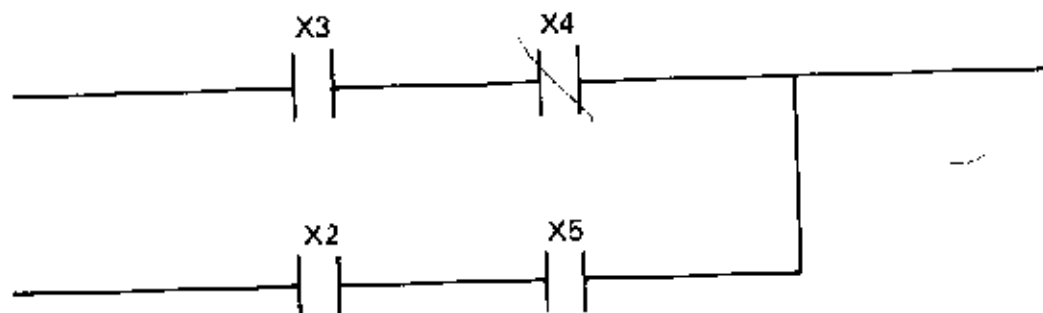


Figure 5.12A. Ladder Diagram for Programming  
(Exercise No. 5)

# LADDER ELEMENT (L MEMORY) STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFIER	I/O NAME	COMMENTS
0	STR	X	3	LS33	
1	AND	X	4	PB48	
2	STR	X	2	LS98	
3	AND	X	5	LS10	
4	OR- STR	-	-	-	
5	AND	X	1	LS6	
6	OUT	Y	5	PILOT LAMP	
7					
8					
9					
10					
11					
12					
23					
24					

Figure 5.12B. Ladder Element (L Memory) Storage Record  
(Exercise No 5)

### 5.13 USE OF AND STR FUNCTION-PROGRAMMING EXERCISE NO. 6

The control sequence listed in Figure 5.13A shows how the ladder diagram of Figure 5.13B will be entered into the CCU. Enter the ladder diagram by pressing the following keys:

CLR AUX 1 STEP YES

ENTER

To clear CCU logic memory with CCU in Startup.

STR 1

- Digit 1 selects logic memory location 1

STR X 1 ENTER

- Places STR X1 in location 1

STR

This STR term will cause element X1 to be placed in PDS1 (Push Down Stack 1) of pushdown stack as the program is executed.

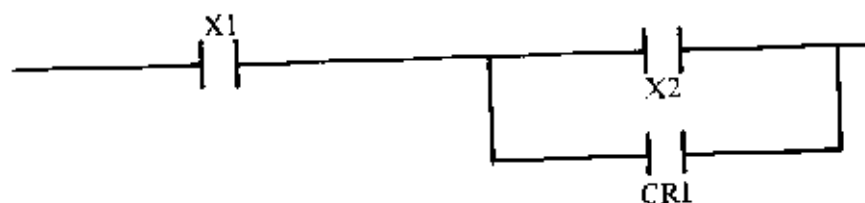
X 2 ENTER

OR CR 1 ENTER

- NOTE. This is a case where an internal output CR1, is used as an input

AND STR ENTER

- The AND-STR entry causes the contents of the PF, power flow, (X2 OR CR1) to be connected in series with the contents of PDS1 (X1) - to obtain the following:



OUT Y 1 ENTER

(At this point the ladder diagram of Figure 5-13B has been entered into the CCU; however, to obtain a visual display on the simulator and verify that the entry is correct, we have added output element Y1)

OUT Y 1 ENTER

LADDER ELEMENT (MEMORY) STORAGE RECORD					
LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFIER	I/O NAME	COMMENTS
0					
1	STR	X	1	LS12	STATUS LS PLT IN PF
2	STR	X	2	PB42	STATUS OF X2 PLT IN PF X1 TO PDS1
3	OR	CR	1	CONTROL RELAY 1	PARALLEL STATUS OF X2 OR CR1 IN PF
4	AND STR				STATUS OF PDS1 (X1) COMBINED WITH PF STATUS (X2 OR CR1) IN SERIES AND
5	OUT	CR	1	CONTROL RELAY 1	
6	OUT	Y	1	FOR VERIFY ONLY	
7					
8					
9					
10					
11					
12					
23					
24					

Figure S 13A Ladder Element (L Memory) Storage Record  
(Exercise No. 6)

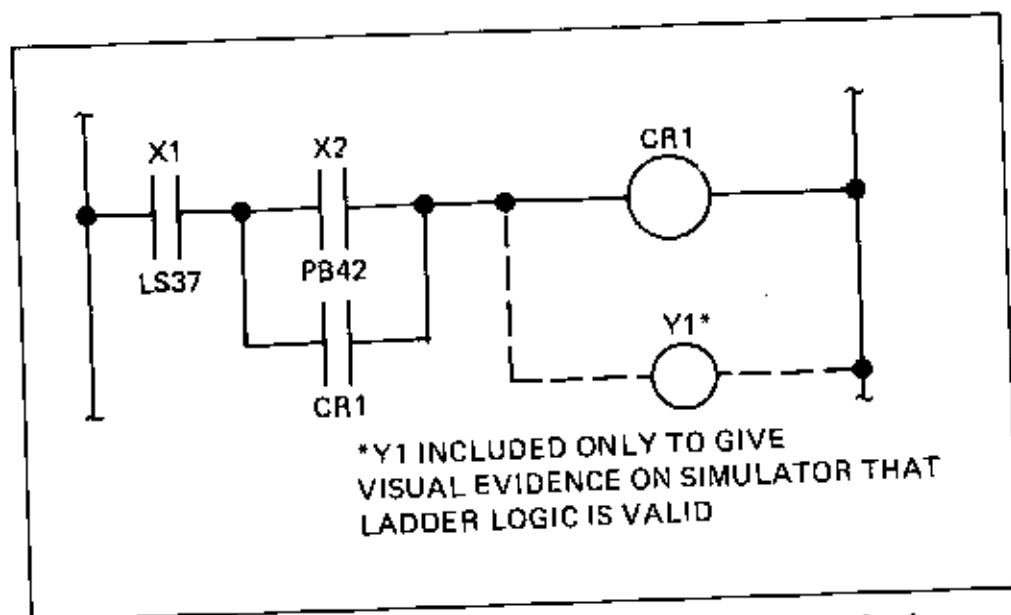


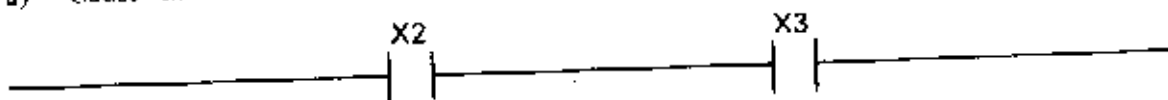
Figure 5.13B. Ladder Diagram for Programming Exercise No. 6

Verify that the control sequence has been entered correctly by closing simulator switches X1 and X2; lamp Y1 lights. Now open switch X2 and note that lamp Y1 remains lighted, indicating that CR1 completes the circuit across X2. Open switch X1 and lamp Y1 will extinguish.

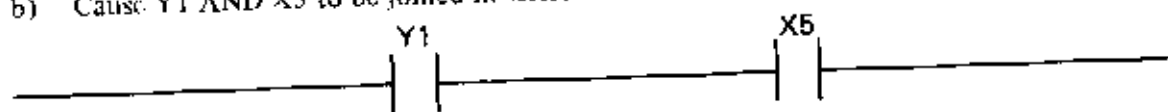
#### 5.14 USING ☐ OR ☐ AND ☐ AND ☐ FUNCTIONS - PROGRAMMING EXERCISE NO. 7

To enter the ladder diagram in Figure 5.14A we must:

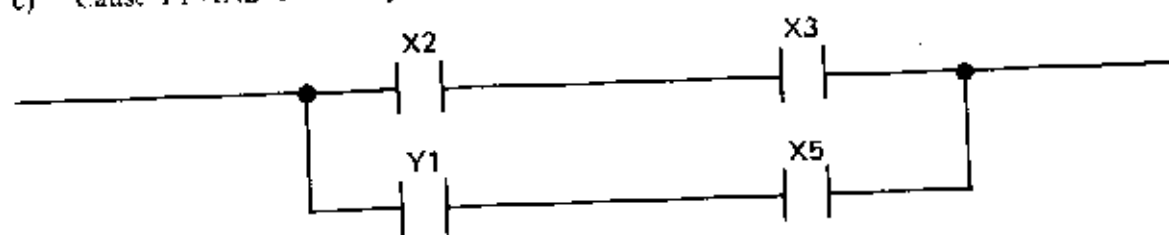
- a) Cause X2 AND X3 to be joined in series



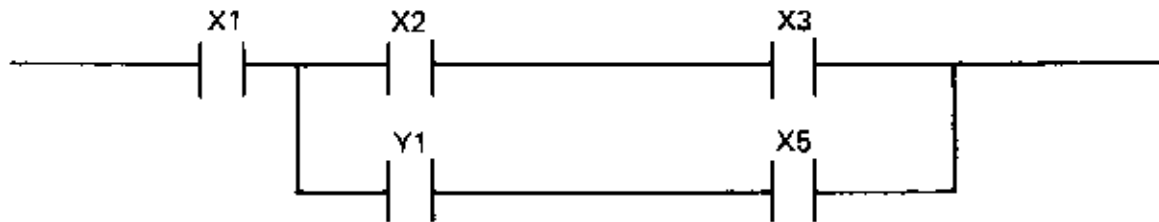
- b) Cause Y1 AND X5 to be joined in series



- c) Cause Y1 AND Y5 to be joined in parallel with X2 AND X3



d) Cause X1 to be joined in series with the parallel combination



To accomplish this, we will enter the control sequence into the CCU in the order shown below. The control sequence for Figure 5.14A is shown in Figure 5.14B.

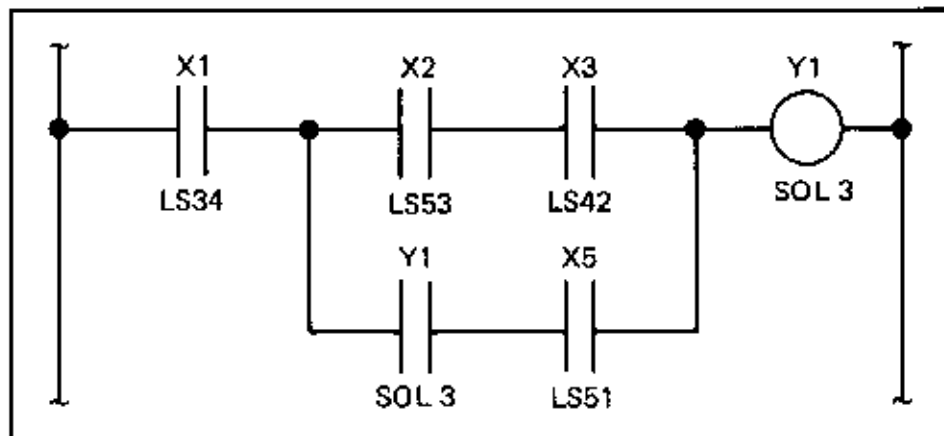


Figure 5.14A Ladder Diagram for Programming Exercise No. 7

LADDER ELEMENT (I. MEMORY) STORAGE RECORD					
LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFIER	I/O NAME	COMMENTS
0	STR	X	1	LS34	NOTE: BEGIN ENTRY BY CLEARING BOTH PROGRAMMER AND CCU WITH AUX 1
1	STR	X	2	LS53	THIS STR FUNCTION PLACES X1 INTO PDS1. X2 AND X3 WILL BE COMBINED IN SERIES AND PLACED IN P1.
2	AND	X	3	LS42	
3	STR	Y	1	SOL3	THIS STR FUNCTION MOVES X1 FROM PDS1 TO PDS2. PLACES X2 AND X3 INTO PDS1. Y1 AND X5 WILL BE COM- BINED IN SERIES AND PLACED IN P1.
4	AND	Y	5	LS51	
5	OR STR				CAUSES (X2 AND X3) TO BE JOINED IN PARALLEL WITH (Y1 AND X5)
6	AND STR				CAUSES PARALLEL COMBINATION TO BE JOINED IN SERIES WITH X1
7	OUT	Y	1	SOL3	
8					
9					
10					
11					
12					
23					
24					

Figure 5 14B Ladder Element (L Memory) Storage Record  
(Exercise No. 7)

**LADDER ELEMENT  
STORAGE LOCATION**

**KEYS PRESSED**

**EXPLANATION**

READY

CLR

—

AUX 1 STEP

PF5 ENTER

Clears all ladder element locations.

CLR 0

First entry to be in ladder element location 0.

0

STR X 1 ENTER

X1 status is placed in PF

1

STR X 2 ENTER

X1 status placed in PDS1.

2

AND X 3 ENTER

AND function causes X2 to be connected in series with X3 and this combination is placed in PF

3

STR Y 1 ENTER

Third STR function places Y1 into PF, causes X1 to be pushed down to PDS2 (see Figure 5.14B), and moves X2 AND X3 from PF to PDS1.

4

AND X 5 ENTER

Second AND function causes Y1 in PF to be connected in series with X5, and this combination is placed in PF.



# LADDER ELEMENT STORAGE LOCATION

## KEYS PRESSED

## EXPLANATION

5



OR -STR function causes the contents of PDS1 (X2 AND X1) to be connected in PARALLEL with contents of PF (Y1 AND X5) to obtain:

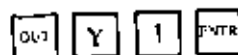
At the same time the contents of PDS2 (X1) pop up into PDS1

6



AND-STR function causes contents of PDS1 (X1) to be connected in SERIES with contents of PF (Y1 AND X5) OR (X2 AND X3) to obtain.

7



OUT function causes an output to occur at terminal 1 of output Mounting Base.

It can be seen that the OR-STR and AND -STR functions are used to either OR the contents of PDS1 with PF or to AND them together. In addition, these two functions bring the contents of PDS2 into PDS1, PDS3 into PDS2, and PDS4 into PDS3

Clear CCU memory using the AUX1 Function before proceeding to the next example.

### 5.15 COMPLEX LADDER DIAGRAM – PROGRAMMING EXERCISE NO 8

In the ladder diagram of Figure 5.15A, a single element (X1) is connected in series with a parallel combination of five elements. Within the parallel combination there are two series combinations. The control sequence for the ladder diagram in Figure 5.15A is shown listed in Figure 5.15B. Once the control sequence is entered into the CCU the following action will take place inside the CCU as it executes the program.

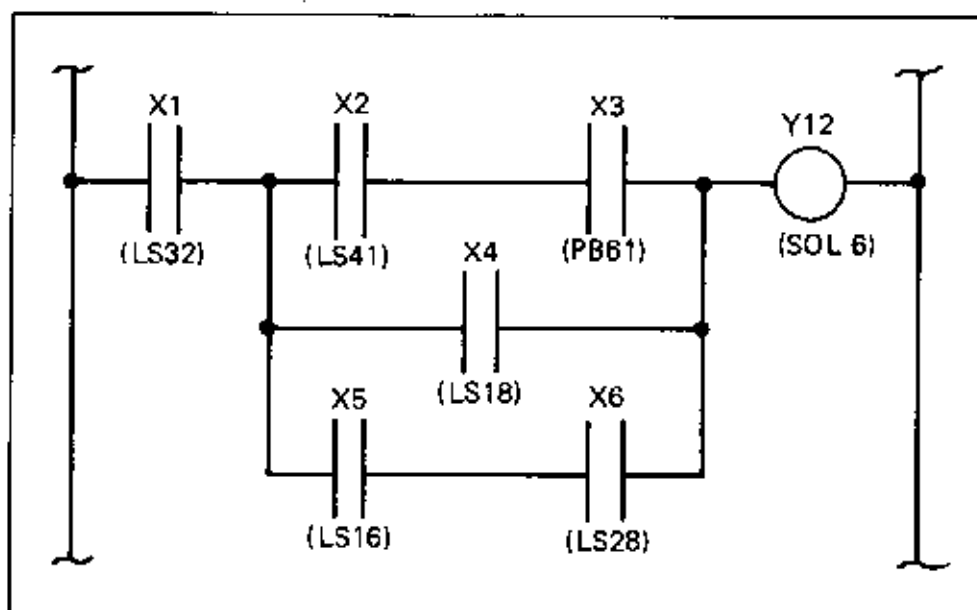


Figure 5.15A. Ladder Diagram for Programming Exercise No. 8

# LADDER ELEMENT (L MEMORY) STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFIER	I/O NAME	COMMENTS
0	STR	X	1	LS32	
1	STR	X	2	LS41	
2	AND	X	3	PB61	
3	OR	X	4	LS18	
4	STR	X	5	LS16	
5	AND	X	6	LS28	
6	OR STR				
7	AND STR				THIS JOINS X1 FROM PDS1 IN SERIES WITH ((X2 AND 1 x 1) OR X4) OR (X2 AND X3)) IN PF.
8	OUT	Y	12	SOL6	
9					
10					
11					NOTE: STEP KEY IS PRESSED FOLLOWING EACH ENTRY TO ADVANCE TO NEXT STORAGE LOCATION.
12					
23					
24					

Figure 5.15B. Ladder Element (L Memory) Storage Record  
(Exercise No. 8)

# LADDER ELEMENT STORAGE LOCATION

## KEYS PRESSED

## EXPLANATION

READY

0

Selects Location 0

0

STR X 1 ENTER

Status of element X1 will be placed in PF

1

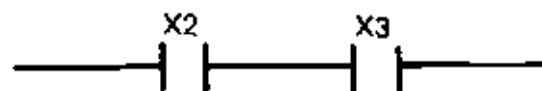
STR X 2 ENTER

Contents of PF (X1) are placed into PDS1, and status of element X2 is placed into PF

-

AND X 3 ENTER

Contents of PF are ANDed with element X3 to obtain:

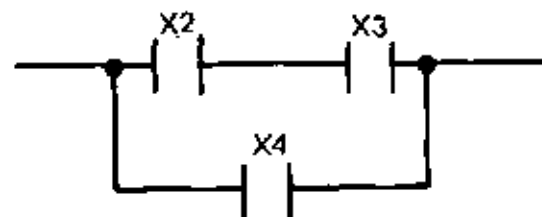


And this combination is placed into PF.

2

OR X 4 ENTER

Contents of PF is ORed with X4 to obtain



Contents of PDS1 (X1) is pushed down into PDS2, content of PF (X2 AND X3) OR X4 is pushed down into PDS1, and element X5 is placed into PF

STR X 5 ENTER

5

AND X 6 ENTER

Content of PF (X5) is ANDed with element X6 and this combination placed into the PF

6

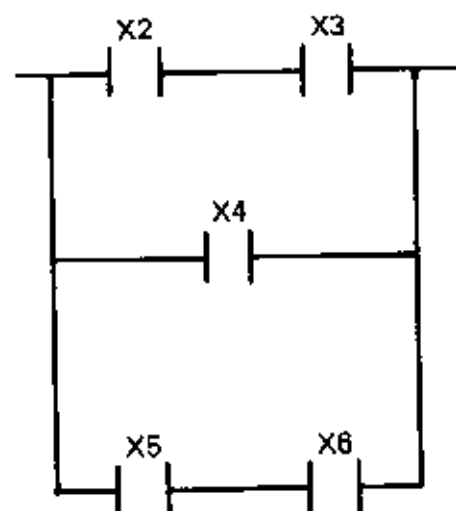
OR STR ENTER

The content of PDS1 (X2 AND X3) OR X4 is ORed with content of PF (X5 AND X6) to obtain

LADDER ELEMENT  
STORAGE LOCATION

KEYS PRESSED

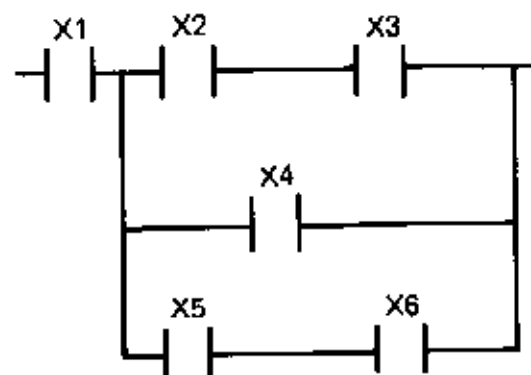
EXPLANATION



and this combination is placed into the PF. at the same time, content of PDS2 pop up to PDS1.

AND STR ENTR

Content of PF (X2 AND X3) OR X4 OR (X5 AND X6) is ANDed with content of PDS1 (X1) to obtain



and this combination is placed in PF

OUT Y12 ENTR

OUT Y12 places the contents of PF into the IR for output to output terminal 12 of the mounting base.

## 5.16 REDRAWING COMPLEX LADDER DIAGRAMS

A ladder diagram having dual outputs and parallel paths to energize those dual outputs is shown in Figure 5.16A. To attempt an entry of the elements in that configuration would prove extremely difficult. The ladder diagram of Figure 5.16A is redrawn in 5.16B and, in this configuration, entry of the elements can be handled using procedures already covered in this manual. A final check of the input power paths required to produce the outputs is the only check required to insure that the control sequence of the redrawn version is the same as that for the more complex ladder diagram.

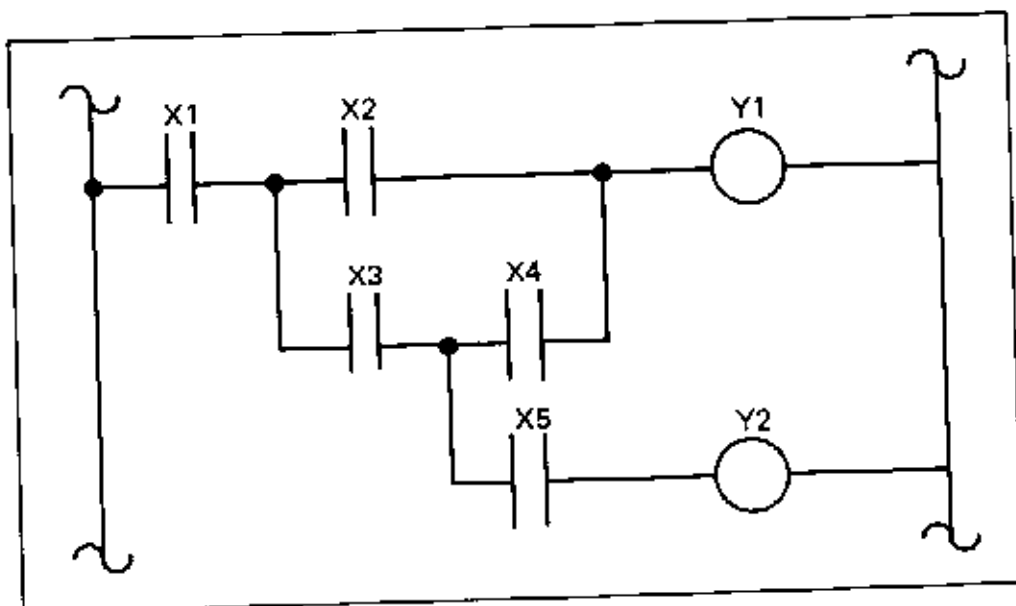


Figure 5.16A Original Ladder Diagram

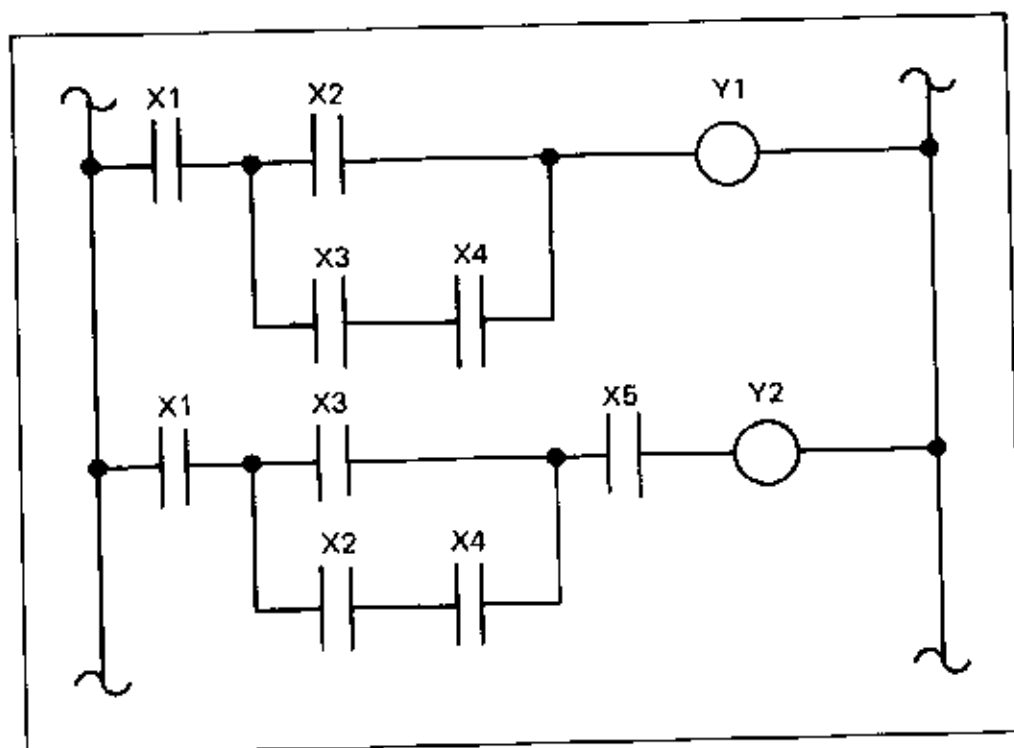


Figure 5.16B. Redrawn Ladder Diagram

## 5.17 PROGRAMMING NOT FUNCTION

The NOT function inverts the status of the selected input, output or control relay. Refer to Figure 5.17A and note that limit switch LS1 is connected to mounting base input terminal X0; also remember that X0 becomes the I/O identifier for LS1. Note also that an output is connected to output mounting base terminal Y1 and terminal Y2. Figure 5.17B shows a ladder diagram in which LS1 and outputs Y1 and Y2 might be combined for entry into the CCU; closing LS1 will, as shown by the ladder diagram, energize Y1 and deenergize Y2. Enter the ladder diagram of Figure 5.17B pressing the following programmer keys in the order given:

CLR AUX 1 STOP YES ENTER

CLR 0 STR X 0 ENTER

OUT Y 1 ENTER

CLR NOT X 0 ENTER

OUT Y 2 ENTER

Set switch X0 on the simulator to closed and observe that lamp Y1 lights and Y2 goes out. Now, set switch X0 to open, lamp Y1 will extinguish and Y2 will light. Using the NOT function inverted the logical status of switch X0 as that status existed at the mounting base input terminal. The NOT function inverts the logical status of any element. Figure 5.17C shows the listing of element location for Figure 5.17B ladder diagram. Turn out lamp Y2 before proceeding by entering the following sequence:

CLP    LCA    1    STOP    YES    ENTER

Note that X0 is referenced in two separate lines but only connects to mounting base at one point.

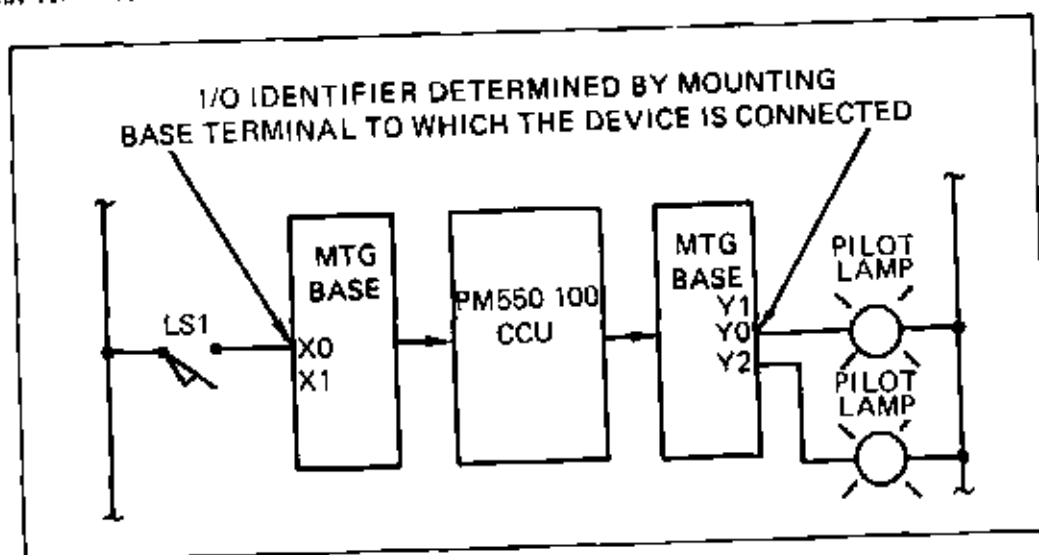


Figure 5.17A Status of Each I/O Device is Applied to Mounting Base

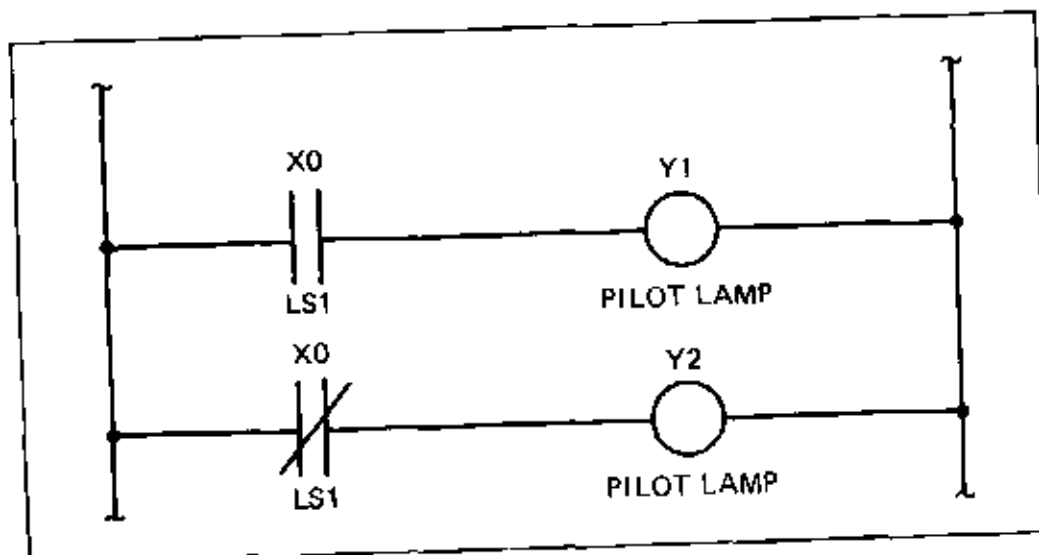


Figure 5.17B Using NOT Function to Invert Input Switch



Figure 5.17D shows a two-line ladder diagram in which the NOT function is used in another way. Output Y1 will be prevented from energizing if the two switches, X1 and X2, are closed at the same time. To see how this operates, enter the ladder diagram of Figure 5.17D by pressing the following programmer keys in the order given. Figure 5.17E shows the Ladder Element (L Memory) Storage Location record.

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFIER	I/O NAME	COMMENTS
	STR	X	0	LIMIT SW 1	
1	OUT	Y	1	PILOT LAMP 1	
2	STR NOT	X	0		WHEN ENTERED INTO THE SEQUENCE, THE NOT FUNCTION CAUSES THE LOGICAL STATUS OF SWITCH X0 TO BE INVERTED. IN THIS CASE A NORMALLY OPEN SWITCH WILL FUNCTION AS A CLOSED SWITCH. CLOSE X40 AND LAMP Y2 IS EXTINGUISHED
3	OUT	Y	2	PILOT LAMP 2	
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
23					
24					

Figure 5.17C. Ladder Element (L Memory) Storage Record (NOT Function)

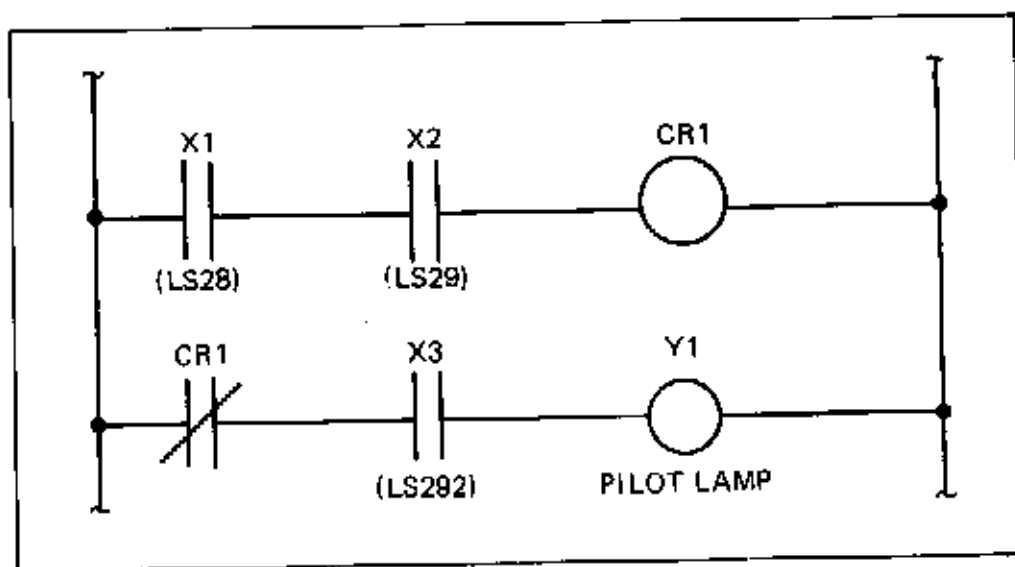


Figure 5 17D. Using NOT Function to Invert Control Relay (CR)

```

CR 0 STR X 1 ENTR
AND X 2 ENTR
OUT CR 1 ENTR
STR NOT ILK 1 ENTR
AND X 3 ENTR
OUT Y 1 ENTR
  
```

Close switch X3 on the Simulator and lamp Y1 will light, close switches X1 and X2 and lamp Y1 will extinguish. Using the NOT function to enter CR1 in the second line inverts the status of CR1; and until CR1 is energized in the first line, Y1 will light each time that switch X3 is closed. Thus, when X1 and X2 are closed, CR1 is energized, and lamp Y1 extinguished. Open either switch X1 or X2 and lamp Y1 will light again.

# NOTE

The NOT key is a legal suffix for the following keys: STR OUT OR AND

However, the combinations "NOT AND STR" and "NOT OR STR" are forbidden

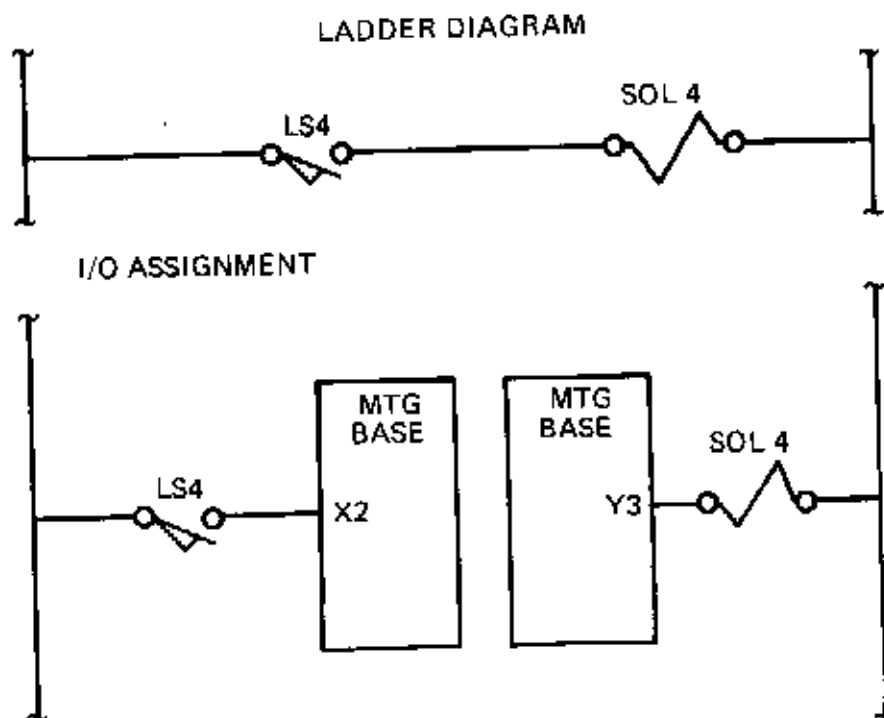
## LADDER ELEMENT (MEMORY) STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFIER	I/O NAME	COMMENTS
0	STR	X	1	LIMIT SW 28	
1	AND	X	2	LIMIT SW 29	
2	OUT	CR	1	CONTROL RELAY 1	
3	STR NOT	CR	1	CONTROL RELAY 1	WITH LS28 AND LS29 CLOSED AT THE SAME TIME THIS PREVENTS THE PILOT LAMP CONTROLLED BY Y1 FROM BEING TURNED ON
4	AND	X	3	LIMIT SW 292	
5	OUT	Y	1	PILOT LAMP	
6					
7					
8					
9					
10					
11					
12					
13					
14					
23					
24					

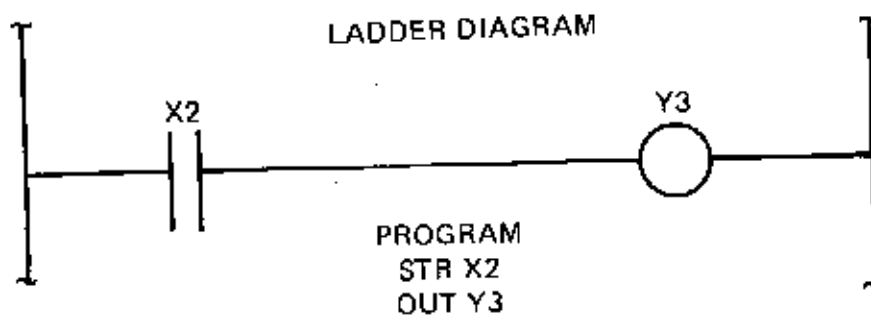
Figure 5.17E Ladder Element (L Memory) Storage Record (Inverting Control Relay)

## 5.18 PROGRAMMING NORMALLY CLOSED INPUTS

The CCU "sees" only that a switch is closed (voltage at an input terminal) or open, not if it is normally closed or open. Therefore, if a function is to occur when an input switch is closed, a normally open contact must be programmed. In the following example, SOL4 is to be energized when LS4 is closed.

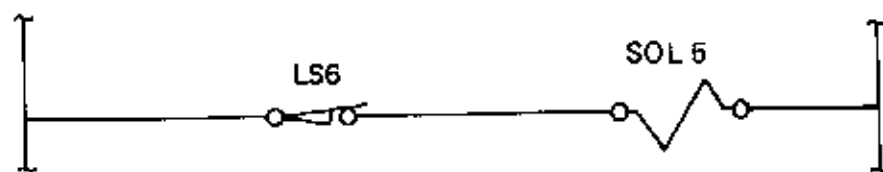


Here LS4 (normally open) is assigned to X2 and SOL4 is assigned to Y3. The diagram and program would be:

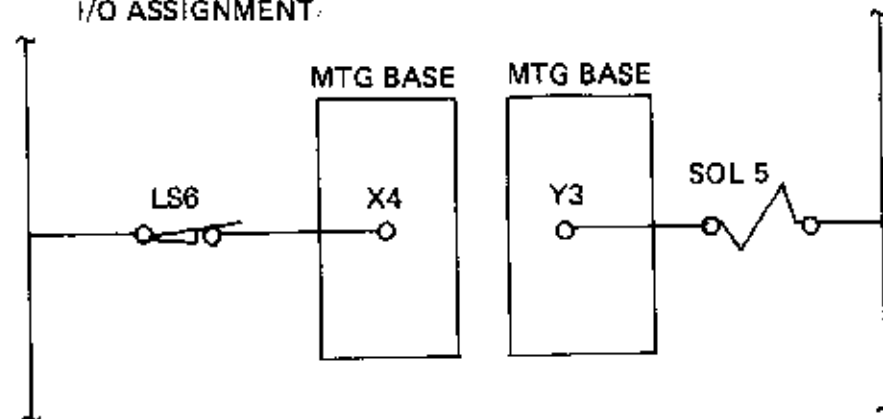


In the next sample SOL5 will be energized when LS6 is closed. A normally closed contact of LS6 is used and SOL5 is energized until LS6 has been actuated.

LADDER DIAGRAM

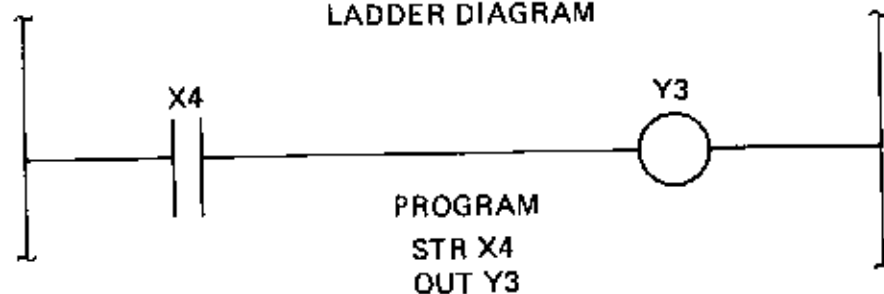


I/O ASSIGNMENT



The diagram and program would be:

LADDER DIAGRAM



PROGRAM  
STR X4  
OUT Y3

Note that in both of these examples, a normally open contact is programmed since an output is to be energized when a contact is closed. If the user wanted Y3 to turn on when LS6 opens, he would have programmed it as follows:

VER NOT X 4 ENTER

OUT Y 3 ENTER

## 5.19 PROGRAMMING MCR FUNCTION

The MCR is used to enable/disable the output(s) in its field of control. Figure 5.19A shows a ladder diagram in which a master control relay (MCR) function is constructed using relay circuitry. Outputs SOL K and SOL L cannot occur until this MCR relay is energized. The same MCR function is shown in Figure 5.19B as it would be diagrammed for entry into the CCU

Outputs Y1 and Y2 are controlled by the MCR function and cannot be energized by their respective actuating switches until the logic combination of ((X3 AND X4) OR X5) is true. When X3 and X4 are closed (lines being controlled are active) outputs Y1 and Y2 can be energized, for output Y1 to energize, X6 and X7 must be closed. If X3 or X4 should open while Y1 is energized, Y1 will be de-energized regardless of whether X6 AND X7 is logically true. Thus the MCR function controls the logical status of the outputs, both Y and CR, that comprise the MCR field of control. Figure 5.19B shows that the MCR field of control is two outputs. To enter the MCR function as diagrammed in Figure 5.19B press the following keys in the order given:

CLR AUX 1 STEP YES ENTER

CLR O STR X 3 ENTER

AND X 4 ENTER

OR X 5 ENTER

MCR 2 ENTER

STR X 6 ENTER

AND X 7 ENTER

OUT Y 1 ENTER

STR X 8 ENTER

OR X 9 ENTER

OUT Y 2 ENTER

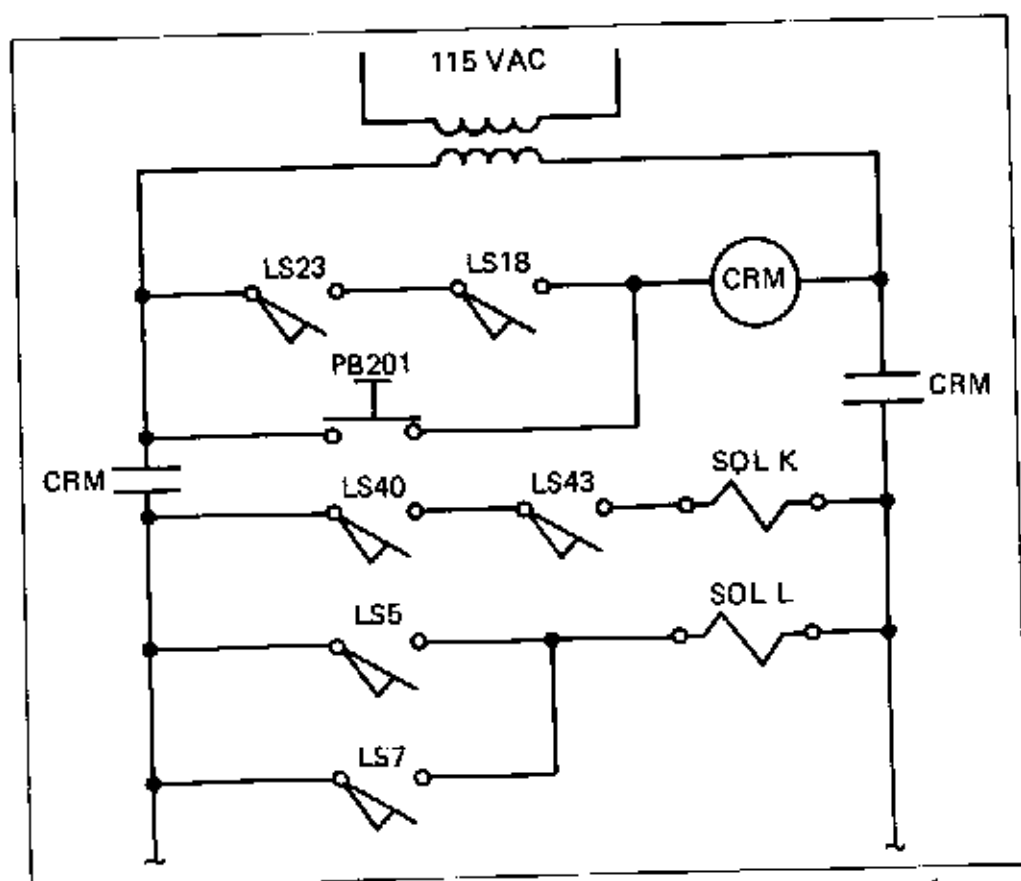


Figure 5.19A Master Control Relay Function Using Relay Terminology

Insure that all Simulator switches are open and CCU is in RUN then close switch X6 and X7. Lamp Y1 does not light because the logic input for MCR is not true. Close switch X5 and lamp Y1 will light. Open switch X5 and lamp Y1 extinguishes. Output Y1 is controlled by the MCR function. Close switch X8. Output Y2 does not light because the logical status of the MCR inputs is not true. Close switches X3 and X4 and the MCR logical input becomes true. Output lamps Y1 and Y2 light. Open switch X3, and output lamps Y1 and Y2 extinguish. Close switches X8 and X9. No output lamp lights. When an output is within the MCR field of control two conditions must be obtained before that output can energize:

- 1) MCR logical input must be true
- 2) Logic required for energizing the specific output must be true

Figure 5.19C shows how the MCR function is recorded before entry in the CCU

A MCR can control up to 1023 outputs.

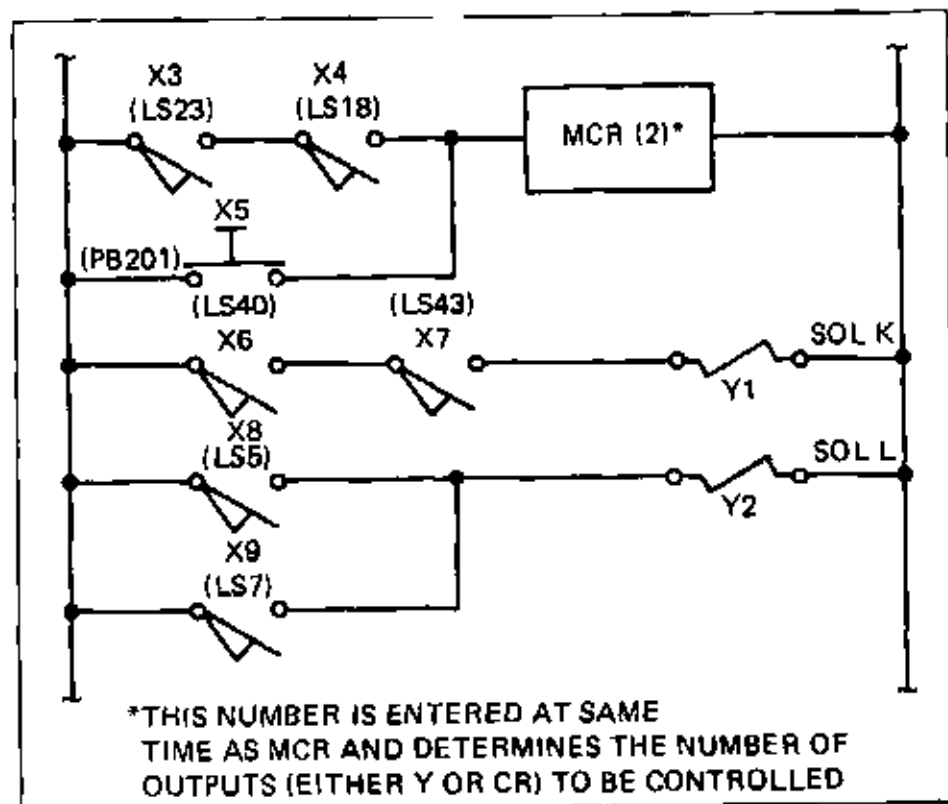


Figure 5 19B Master Control Relay Function Using the PM550 CCU



# LADDER ELEMENT (L MEMORY) STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFIER	I/O NAME	COMMENTS
0	STR	X	3	LS23	
1	AND	X	4	LS18	
2	OR	X	5	PE201	
3	MCR	-	2	-	NUMBER OF OUTPUTS COMPRISING MCR Y OR CN OUTPUT FIELD OF CONTROL ENTERED WITH MCR
4	STR	X	6	LS40	
5	AND	X	7	LS43	
6	OUT	Y	1	SOL K	
7	STR	X	8	LS5	
8	OR	X	9	LS1	
9	OUT	Y	2	SOL I	
10					
23					
24					

Figure 5 19C Ladder Element (L Memory) Storage Record for MCR Function

## 5.20 PROGRAMMING JUMP FUNCTION

The JUMP function with its input off causes a specified number (its field of control) of outputs to be skipped, or "frozen" logically in their last state. When a sequence is required which must be performed only at a specified times and the outputs must remain in their state at all other times, a jump function can be used. The ladder diagram shown in Figure 5.20A contains a 2 output jump function. The assignment of Ladder Element (L Memory) Storage Location numbers for the ladder diagram is shown in Figure 5.20B. The "field of control" for a jump function is defined as the number of subsequent outputs, both Y and CR, controlled by the jump function. For example, JUMP (2) would indicate that the next two outputs would be controlled by the JUMP function. A JUMP can have a field of 4095 outputs maximum. As shown in the ladder diagram, in Figure 5.20A, the JUMP function field-of-control consists of outputs Y15 and Y7. An output to energize Y15 can occur only if X8 is closed at the same time that X26 is closed. (The same is true for Y7 and X19). However, if during the time that Y15 is energized X8 opens, Y15 will remain energized, should X26 open after X8 opens, Y15 will still remain energized until X8 closes again. (This is true for all outputs within the JUMP function field of control). To enter the ladder diagram of Figure 5.20A containing the JUMP function, press the following keys in the order given:

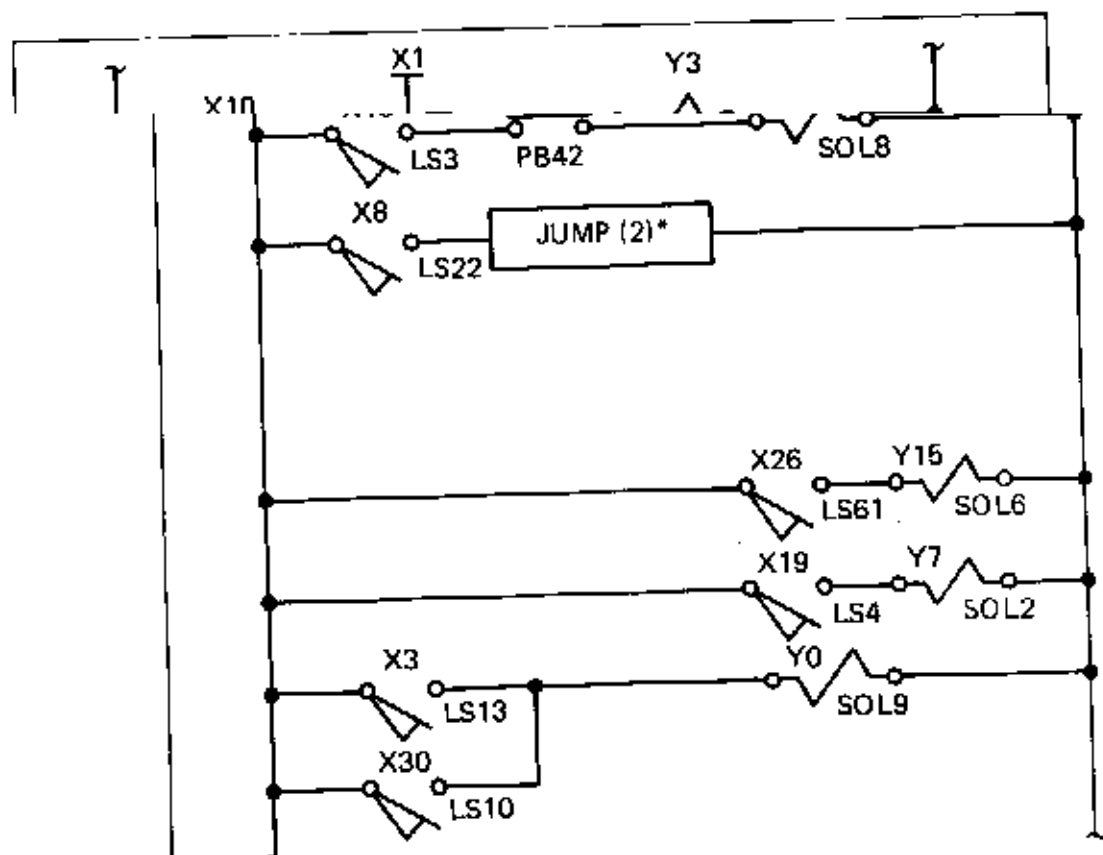
CLR AUX 1 STEP YES ENTR

CLR 0 STR X 1 0 ENTR

AND X 1 ENTR

OUT Y 3 ENTR

STR X 8 ENTR



\*WHEN SWITCH X8 IS OPEN, OUTPUTS Y15 AND Y7 ARE JUMPED OVER DURING SEQUENCE AND NO CHANGE IN STATUS OF OUTPUTS CAN OCCUR. WHEN X8 CLOSES (LINES BEING CONTROLLED ARE ACTIVE), OUTPUTS FROM Y15 AND Y7 CAN OCCUR IF X26 AND X19 CLOSE. WHEN X8 OPENS, Y15 AND Y7 RETAIN THEIR OUTPUT CONDITION (ON OR OFF) AND CANNOT CHANGE REGARDLESS OF STATUS OF X26 OR X19 UNTIL X8 CLOSES AGAIN

Figure 5 20A Ladder Diagram of 2-Line JUMP Function

# LADDER ELEMENT (L MEMORY) STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFIER	I/O NAME	COMMENTS
0	STR	X	10	LS3	
1	AND	X	1	PB42	
2	OUT	Y	3	SOL 8	
3	STR	X	8	LS22	OUTPUTS Y3 AND Y0 ARE NOT PART OF THE JUMP FUNCTION; TO VERIFY THIS, OPEN X8 AND CLOSE SWITCHES X10, X1, AND X3 ON SIMULATOR. LAMPS Y3 AND Y0 RESPECTIVELY WILL LIGHT. CLOSE SWITCH X26 AND NOTE THAT Y15 DOES NOT LIGHT. CLOSE SWITCH X8, AND Y15 WILL LIGHT WITH X26 STILL CLOSED. OPEN X8, Y15 REMAINS LIGHTED. OPEN X26 AND LAMP Y15 WILL NOT EXTINGUISH - PROVING THAT OUT PU IS UNDER CONTROL OF THE JUMP FUNCTION CANNOT CHANGE THEIR ON/OFF STATUS EXCEPT WHEN THE CONTROL LING SWITCH (IN THIS CASE X8) IS CLOSED. THIS IS TRUE FOR ALL OUTPUTS IN THE JUMP FIELD OF CONTROL
4	IMP	-	2		
5	STR	X	26	LS61	
6	OUT	Y	15	SOL 6	
7	STR	X	19	LS4	
8	OUT	Y	7	SOL 2	
9	STR	X	3	LS13	
10	OR	X	30	LS10	
11	OUT	Y	0	SOL 9	
12					
13					
23					
24					

Figure 5.20B Ladder Element (L Memory) Storage Record for the JUMP Function

At this point the jump function must be selected; and, as shown in Figure 5.20A, this will be assigned to Jadder Element (L Memory) Storage Location 4.

JUMP 2 ENTER - (This number is the number of outputs that will be contained in the JUMP field of control)

MCR X 2 6 ENTER

OUT Y 1 5 ENTER

STR X 1 9 ENTER

OUT Y 7 ENTER

STR X 3 ENTER

OR X 3 0 ENTER

OUT Y 0 ENTER

The difference between the MCR and JUMP function is that while the MCR turns outputs off if its input condition is false (PF = 0), the JUMP function simply jumps over them (does not change their logical status) if its input condition is false.

#### NOTE

A jump or MCR within the control range of another active jump or MCR is treated as a "No Operation", i.e., it has no effect.

## 5.21 PROGRAMMING SHIFT REGISTERS

A shift register is used to store and transfer information from one output or stage to another in conjunction with the movement of parts on a machine (see 5.12 and 13 also). The shift register stages correspond to physical locations on the machine. Shift registers are commonly used with index tables or transfer lines to keep track of part status. The information is shifted from stage to stage by a switch on the indexing mechanism. Figure 5.21A shows an example of a 4-stage shift register. To enter the shift register, press the following keys in the order given:

IR ALX 1 STEP YES ENTER  
IR 0 STR NOT X 3 ENTER  
MEP 5 ENTER  
STR X 2 ENTER  
AND NOT CR 0 ENTER  
JUMP 4 ENTER  
STR Y 2 ENTER  
OUT Y 3 ENTER  
STR Y 1 ENTER  
OUT Y 2 ENTER  
STR Y 0 ENTER  
OUT Y 1 ENTER  
STR X 1 ENTER  
OUT Y 0 ENTER  
STR X 2 ENTER  
OUT X 0 ENTER

# LADDER ELEMENT (L MEMORY) STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFIER	I/O NAME	COMMENTS
0	STR NOT	X	3	PB1	<p>X1 DEFINES STATE SHIFTED IN X2 SERVES AS SHIFT PULSE X3 IS RESET</p> <p>ADDITIONAL STAGES CAN BE ADDED HERE BOTH THE MCR AND JUMP FIELDS OF CONTROL MUST BE INCRE- MENTED BY ONE FOR EACH ADDED STAGE.</p> <p>CLOSE X1 AND X2. OUTPUT Y0 WILL LIGHT. OPEN X1 AND CLOSE X2 SEVERAL TIMES. OBSERVE THAT THE OUTPUT STEPS ONCE FOR EACH SWITCH CLOSURE. CLOSING X3 TURNS ALL OUTPUTS OFF. X3 MUST BE OPEN TO MAKE THE SHIFT REGISTER OPERATIVE.</p>
1	MCR	-	5		
2	STR	X	2	LS4	
3	AND NOT	CR	0		
4	IMP	-	4		
5	STR	Y	2		
6	OUT	Y	3		
7	STR	Y	1		
8	OUT	Y	2		
9	STR	Y	0		
10	OUT	Y	1		
11	STR	X	1		
12	OUT	Y	0		
13	STR	X	2		
14	OUT	CR	0		
23					
24					

Figure 5-21 A. Ladder Diagram and Program for Shift Register

## 5.22 HOLDING CIRCUITS AND LATCH RELAYS

The image register in the CCU is divided so that CR0 through CR255 and Y0 through Y255 are nonretentive in nature and can be used in nonretentive sealing (holding) circuits. They will be cleared at power turn on. CR256 through CR511 are retentive in nature and therefore, perform as latch relays during POWER OFF. Latch outputs (Y) may be programmed by using retentive CRs as shown in Figure 5.22A.

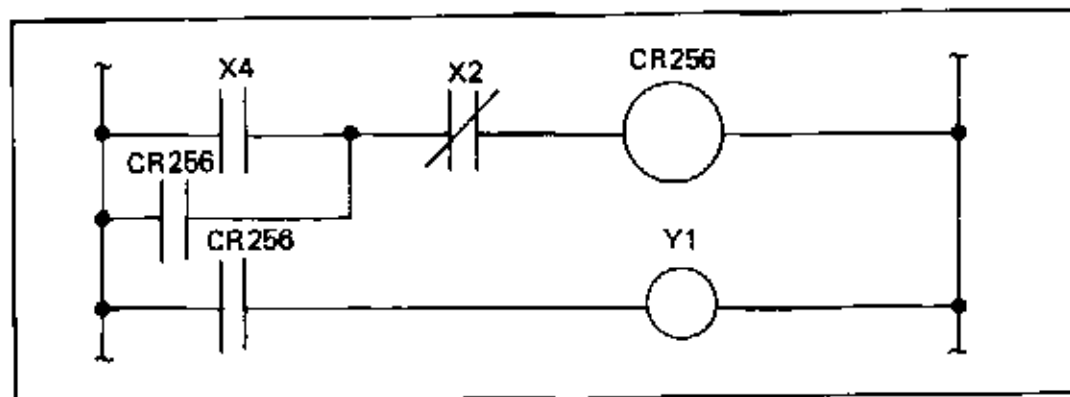


Figure 5.22A Programming Latch Outputs

## 5.23 PROGRAMMING END OF SCAN FUNCTION

As described in Chapter 2 (System Operation) the PM550 operates with a nominal 17 msec scan of its 2048 words of ladder logic memory (31 msec nominal for 4096 word), followed by a 2 msec I/O cycle. The END OF SCAN is a conditional operation which can be implemented in any ladder logic (I) location to reduce the memory scan time.

When a program is entered that requires only a portion of the 2048 (or 4096) word capacity, the scan time can be reduced by using an END OF SCAN at the end of the program. Figures 5.23A and 5.23B display the PM550 ladder logic timing diagram with and without EOS. The equation in Figure 5.23C can be used to determine memory scan time based on the L location in which the EOS is used.

EOS can be implemented in the middle of a LOGIC program. That portion of the program after the EOS will not be scanned, causing all outputs to remain in their last state; counters and timers are inoperative. Figure 5.23D shows a ladder diagram which used EOS in the middle of ladder logic.



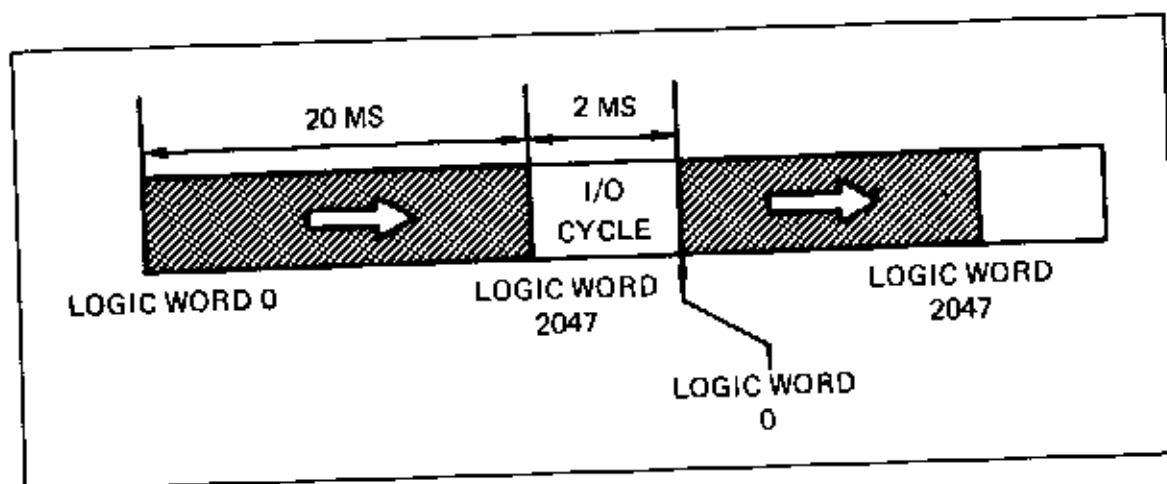


Figure 5.23A. PM550 Timing Diagram without EOS

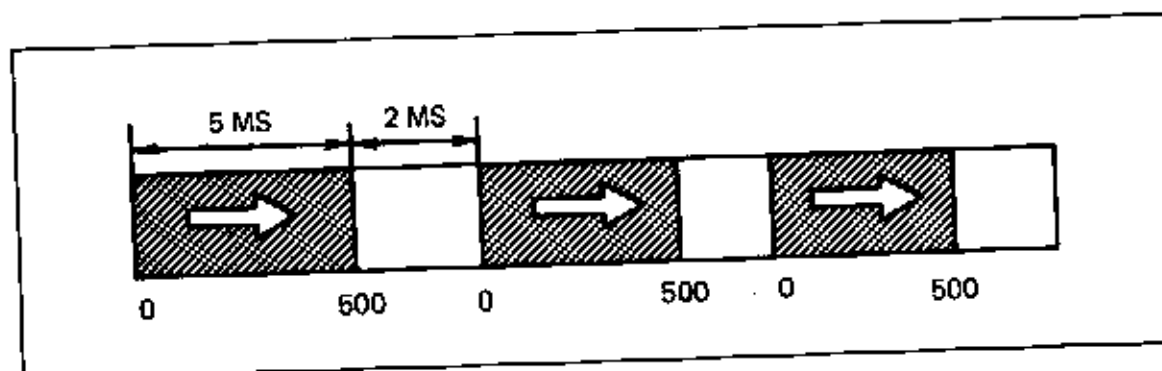


Figure 5.23B PM550 Timing Diagram with EOS at Location No 500

Memory Scan Time (Milliseconds) – Worst Case

$$= 3.3 + 0.00518 \times M + 0.6 \times F$$

M = Length of Ladder Logic Program in Words

F = Number of Special Functions

Example:

3 Special Functions

678 Words to EOS

$$\text{Scan Time} = 3.3 + 0.00518 \times 678 + 0.6 \times 3$$

$$= 8.61 \text{ Milliseconds -- Worst Case}$$

Figure 5.23C. Scan Time Formula

# LADDER ELEMENT STORAGE RECORD

0	STR X1
1	OUT Y1
2	STR X2
3	EOS (C)
4	STR X3
5	OUT Y3

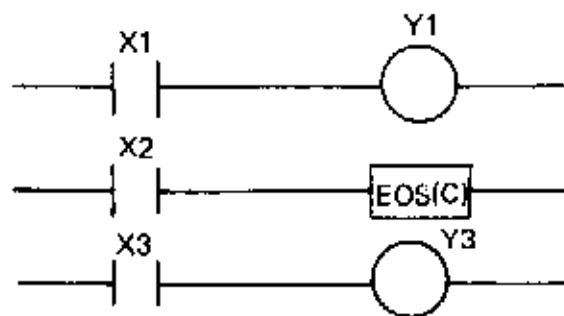


Figure 5.23D Ladder Diagram and Storage Record for EOS

### 5.23.1 PROGRAMMING A CONDITIONAL END OF SCAN

To enter the ladder diagram in Figure 5.23D press the following keys in the order given:

#### NOTE

When the  key is pressed, the prompt "EOS COND?" is displayed. If  is answered, the EOS is conditional upon input power flow.

With X2 closed power flow is true (on) and EOS does not effect the logic. With X2 open, power flow is false (off) and EOS takes effect.

To verify EOS operation open switches X1, X2, and X3. With Switch X2 open the End of Scan takes effect. Closing X3 does not effect output Y3. Close X1 and Y1 lights. Now close X2, the EOS is no longer in effect and X3 will now operate Y3 correctly. With Y3 energized, open X2 and verify that X3 no longer operates Y3 due to the EOS.

### 5.23.2 PROGRAMMING AN UNCONDITIONAL END OF SCAN

The unconditional END OF SCAN (EOS) causes the scan to terminate regardless of the power flow in, the I/O cycle occurs, and then execution begins again at location 0. It is programmed identically to a conditional END OF SCAN, except  is answered to the prompt "EOS COND?"



## 5.24 DEFINING INTEGER VALUES

In order to implement timers, counters, math equations, special functions and process control equations, numeric values, positive or negative, used in these functions must be defined and stored in the USER STORAGE AREA.



There are 2048 (or 3072) Locations (C0 through C1023 (or C2047), V0 through V1023) which are available for storage of numeric values.



Because ladder logic is restricted to the use of Integer values (no fractions or decimals), the following examples illustrate the entry of integers. Chapter 6 describes the entry of non-integer values and their use in special functions.



As an example, the value of 29 can be assigned to data storage location C22 by pressing the following keys.




 As indicated in Section 5.2, the  key returns the programmer to the READY state. In all the previous ladder diagram programming examples a ladder element location No. (0-4095) was then selected for entry of the first ladder element. The READY state allows entry into any of several available storage areas.



 — User "Constant" Data Storage Area has been selected.

  — User storage area location No. 22.  
Display indicates: C22

 — The  key has been pressed so that a value can be assigned to data storage location No. C22.

  — Display indicates: C22 = 29

 NOTE: The displayed information does not transfer into the CCU until the  key has been pressed. Pressing  automatically steps to the next storage location.

Once a location in the data storage area has been selected, the  and  keys can be used to increase or decrease the location number.

### WARNING

Do not use user area C0 → C15 for user program data. These 16 locations are reserved for loop table pointers. Appropriate warnings will appear on the Read/Write Programmer display.

The following example illustrates the use of the **READ** key in the data storage area:

**CLR** - Display indicates READY (RUN)

**C** **2** **2** - Selects location C22 to be read

**READ** - Display indicates: C22 = 29

Assignments of data storage area location numbers should be recorded on the data storage record form shown in Figure 5.24A. As a program is developed, a record should be kept of the location numbers that have been assigned a value along with the value. It can be employed as many times as required in the ladder diagram program. To change the value assigned to location C22 press the following keys:

**CLR** - Programmer indicates READY (RUN)

**C** **2** **2** - Selects location C22

**=** **9** - Selects new value

**ENTR** - The old value is automatically erased and the value 9 entered in its place.

If an error is made in defining a value for a particular data storage location, the **CE** key can be used to clear the value from the display so that a correct entry can be made.

KEY SEQUENCE	DISPLAY	NOTE
<b>CLR</b>	READY	
<b>C</b> <b>2</b> <b>2</b> <b>=</b> <b>2</b> <b>9</b>	C22 = 29	The <b>CE</b> affects the programmer display only, with no effect on CCU memory
<b>CE</b>	C22	
<b>=</b> <b>4</b> <b>0</b>	C22 = 40	
<b>ENTR</b>	C23	








Storing and recalling "variable" data in the V area uses the same key sequences with V substituted for C.


USER MEMORY STORAGE RECORD			
C AREA	LOCATION NO.	VALUE	COMMENTS
C	1		
C	2		
C	3		
C	21		
C	22	29	

Figure 5.24A. User Memory Storage Record

## 5.25 LADDER LOGIC MULTI-WORD FUNCTIONS

Each of the functions which have been described up to this point have been single word functions. The PM550 has 7 multi-word functions which can be programmed into ladder logic memory:

	TIMER	3 Words
	COUNTER	3 Words
	INTEGER ADD	4 Words
	INTEGER SUBTRACT	4 Words
	INTEGER COMPARE	3 Words
	INTEGER MOVE	3 Words
	SPECIAL FUNCTION REQUEST	2 Words

Single word functions are defined and entered into Logic memory one word at a time. Multi-word functions are different in that  is pressed only after the complete function (2, 3 or 4 words) is defined. Paragraphs 5.25.1 through 5.25.8 describe the correct entry procedure for these functions. Paragraph 5.25.10 describes how to change or edit a multi word function after it has been entered.

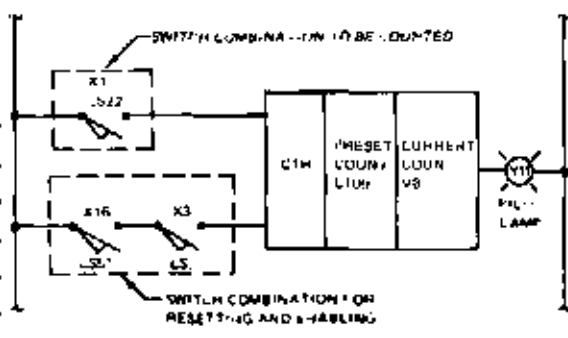
Special Function (SF) entry and editing procedures are described in detail in Chapter 6.

With the exception of SF, all other ladder logic multi-word functions are performed on every scan of memory - without affecting the memory scan time described in paragraph 5.23. As mentioned earlier, these are ladder logic functions and are restricted to the use of integers as data values.

### 5.25.1 ENTERING COUNTERS

A ladder diagram for a typical counter is shown in Figure 5.21.1A along with the Ladder Element (L Memory) Storage Location assignments. The number of closures of X1 will be counted, and this switch will be entered into the CCU first. Any combination of switches can be entered, in which case the number of times that the combination is active will be counted. Next, the combination of switches constituting the reset/enable condition is entered, in this case reset is performed when X16 or X3 is opened. The counter is enabled when both are closed. Constant C100 has been selected to define the PRESFT count. This is followed by V8 which is used as a blank space where the current count is developed. To enter the ladder diagram of Figure 5.21.1A press the following keys in the order given.

# LADDER ELEMENT (MEMORY) STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	UD TYPE	UD IDENTIFIER	UD NAME	COMMENTS
0	STR	X	1	LS2A	 <p>SWITCH COMBINATION - ON - TO BE COUNTED</p> <p>SWITCH COMBINATION - ON - RESETTING AND HABILITING</p>
1	STR	X	16	LS31	
2	AND	X		LS	
3	CTR				
4	C100				
5	V8				
6	OUT	Y	11	PILOT LAMP	<p>TO VERIFY THAT COUNTER FUNCTIONS AS ENTERED: CLOSE SWITCHES X16 AND X3 ON SIMULTANEOUSLY. THEN, CLOSE SWITCH X1 SEVENTEEN TIMES. LAMP Y11 WILL LIGHT, INDICATING PRESET COUNT HAS BEEN OBTAINED. OPEN SWITCHES X1 AND X3 (OR X16); OPENING EITHER SWITCH X16 OR X3 RESETS THE COUNTER. CLOSE X3, NOW PRESS PROGRAMMER [CLR] KEY AND THEN PRESS [H] [H] PRESS [READ] KEY. CLOSE X1 SEVENTEEN TIMES AND NOTE THAT THE DISPLAY INCREMENTS IN DECIMAL THROUGH 9 SHOWING THAT COUNTER IS ACTIVE INCREMENTING IN THIS MANNER IS A GOOD CHECK THAT THE COUNTER IS FUNCTIONING CORRECTLY. TURN OFF Y11 BEFORE PROCEEDING THROUGH NEXT EXAMPLE.</p>
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					

COUNTER MEMORY STORAGE RECORD			
AREA	LOCATION NO	VALUE	COMMENTS
✓	1		
✓	2		
✓	3		
✓	4		
✓	5		
✓	6		
✓	7		
✓	8	*	CURRENT VALUE

Figure 5-25.1A Ladder Diagram and Storage Records for the Counter Function



KEY SEQUENCE	DISPLAY	NOTE
CLR AUX 1 STEP	CLEAR L?	CCU must be in Start up to clear memory.
YES	PRESS ENTR TO CLEAR	
ENTR	CLEAR V?	
YES	PRESS ENTR TO CLEAR	
ENTR	CLEAR C?	
YES	PRESS ENTR TO CLEAR	
ENTR	READY (START)	Clears all memory (L, V, and C)
0 STR X 1	0 STR X1	Closures of X1 to be counted if Reset/Enable (X16 and X3) combination true
ENTR STR X 1 6	1 STR X16	Reset/Enable input in series with X3 input.
ENTR AND X 3	2 AND X3	
ENTR CTR	3 CTR PROTECT?	Counters and Timers can be protected from modification by the TCAM (Timer Counter Access Module) by answering YES to the prompt. Protected counters read as "COUNTER (P)".
NO	COUNTER	
STEP C 1 0 0	4 PRE: C100	Preset stored in C100
STEP V 8	5 CUR: V8	Current value stored in V8

Note: Because a counter requires three locations to be complete, the **ENTR** key is pressed only AFTER all three have been defined

## KEY SEQUENCE

## DISPLAY

## NOTE

ENTR [OUT] Y 1 1

6 OUT Y 11

Backstep (BSTP) is not allowed during definition of the preset and current words.

ENTR CLR

READY (START)

C 1 0 0 = 7

C100 = 7

ENTR

C101

Verify counter operation as indicated in Figure 5.25.1A. The current count can be observed by viewing storage area location V8. The maximum count capability is 32,767.

## 5.25.2 DELAY-ON TIMERS

A delay-on timer is used where it is desired to actuate a device at some time after the actuating switch has closed. Figure 5.25.2A shows one form of delay-on timer in which a single switch starts and resets the timer (X1 closed = timer start, X1 open = timer reset). The dynamics of this delay-on timer is as follows. Three seconds after switch X1 is closed, the pilot lamp Y8 lights. To enter the delay-on timer shown in Figure 5.25.2A press the following keys in the order given:

## KEY SEQUENCE

## DISPLAY

## NOTE

CLR AUX 1 STOP

CLEAR L?

YLG

PRESS ENTR TO CLEAR

ENTR

CLEAR V?

YES

PRESS ENTR TO CLEAR

ENTR

CLEAR C?

YES

PRESS ENTR TO CLEAR

ENTR

READY (START)

All Memory Cleared.

C STR X 1

0 STR X1

ENTR STR X 1

1 STR X1

# KEY SEQUENCE

# DISPLAY

ENTR <sup>TMR</sup> <del>PRM</del>	2 TIMER PROTECT
VIS	2 TIMER (P)
STEP V 1	3 PRE V1
STEP V 1 0	4 CUR V10
ENTR OUT Y 8	5 OUT Y8
ENTR TMR	READY (START)
V 1 - 3 0	V1 = 30
ENTER CLR	READY (START)

## NOTE

Because a timer requires three words to be complete the enter key is pressed only after all three have been defined. Also, BSTP is not allowed during definition of the Preset and Current words.

To enter a delay-on timer that will accumulate increments of time in accordance with the duration of input switch closure requires a timer configuration like that shown in the ladder diagram of Figure 5-25-2B. Note that two different input elements are required, one element enables/resets the timer and the second element activates the timer each time it closes. The dynamics of the delay-on timer contained in the ladder diagram shown in Figure 5-25-2B follows. Both switches X3 and X4 must close before the timer can set and be activated. Closing X4 enables the timer and closing X3 activates the timer (if X4 is closed). Each time X3 closes (while X4 is closed) the timer decrements.

# LADDER ELEMENT (MEMORY) STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	ELEMENT TYPE	I/O TYPE	I/O IDENTIFIER	I/O NAME	COMMENTS
0	STR	X	1	1023	<p>SW X1 IS IN SERIES WITH TMR. TMR IS IN SERIES WITH Y1. Y1 IS IN SERIES WITH PILOT LAMP.</p>
1	STR	X	1	1023	
2	TMR	(P)			<p>TIME DELAY ON TMR. TMR IS HIGH WHEN X1 IS HIGH. Y1 IS HIGH WHEN TMR IS HIGH.</p>
3	VI				
4	VO				<p>TIME DELAY ON TMR. TMR IS HIGH WHEN X1 IS HIGH. Y1 IS HIGH WHEN TMR IS HIGH.</p>
5	OUT	Y	X	PILOT LAMP	
6					<p>NOTE: TO VERIFY THAT LADDER DIAGRAM FUNCTIONS AS ENTERED, CLOSE SWITCH X1 ON SIMULATOR. THREE SECONDS LATER LAMP Y1 WILL LIGHT, INDICATING THAT TIMER HAS TIMED OUT. NOW OPEN X1 AND PRESS PROGRAMMER [CLR] KEY. PRESS [0] [0] NEXT PRESS [READ] KEY AND HOLD KEY DOWN. CLOSE SWITCH X1 AND OBSERVE THAT THE HAND DISPLAY FIRST SHOWS THE NUMBER 30 (REPRESENTING 30.0 SECONDS). THEN AFTER X1 IS CLOSED, THE DISPLAY CHANGES FROM 30 TO 0, DECREMENTING, TIMING DOWN, LAST 1.0 SECOND. THIS IS A VISUAL LOOK AT THE TIMER DYNAMICS AND PROVIDES A GOOD CHECK OF TIMER OPERATION.</p>
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					

# USER MEMORY STORAGE RECORD

V AREA	LOCATION NO	VALUE	COMMENTS
V	1	30	PRESET TIME
V	2		
V	3		
V	4		
V	5		
V	6		
V	7		
V	8		
V	9		
V	10	*	CURRENT TIME

Figure 5-25-2A Ladder Diagram and Storage Records for Delay On Timer

# LADDER ELEMENT 1 MEMORY STORAGE RECORD

LADDER ELEMENT STORAGE LOCATION	EL. NAME	DO TYPE	DO	DO TYPE	DO	DO NAME	COMMENTS
0	STR	X	1	1	1	1.521	
1	STR	X	4	4	4	1.531	
2	TMR						
3	CIOO						
4	VI						
5	DEI	Y	8	8	8	1.541	
6							
7							
8							
9							
10							<p>NOTE: TO VERIFY THAT LADDER DIAGRAM FUNCTIONS AS ENTERED, CLOSE SWITCH X4 ON SIMULATOR. NEXT, CLOSE SWITCH X2 FOR APPROXIMATELY 10 SECONDS, AND THEN OPEN X2. CONTINUE TO CLOSE AND OPEN X2. TIME WILL BE ACCUMULATED EACH TIME X2 CLOSES UNTIL 4 SECONDS ACCUMULATE AND LAMP Y1 LIGHTS. NO FURTHER TIME IS ACCUMULATED AFTER TIMER TIMES OUT. RE-ARMED. ON NUMBER OF TIMES X2 IS CLOSED. OPEN X2 AND X4. PRESS PROGRAMMER [CLR] KEY, AND PRESS [V]1. PRESS PROGRAMMER [READ] KEY, HOLD DOWN, AND CLOSE SWITCH X4. OBSERVE PROGRAMMER LEFT HAND DISPLAY AND COUNT, AND OPEN X2, BEGINNING WITH NUMBER 40 (40 X .01 = 4 SECONDS). RELAYS PLAYS WILL DECREMENT TO ZERO STOPPING EACH TIME X2 IS OPEN. THUS, IT IS SEEN THAT TIME ACCUMULATES ONLY WHEN X2 IS CLOSED, AND TIMER IS ENABLED.</p>
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							

1-SEC MEMORY STORAGE RECORD			
C AREA	LOCATION NO	VALUE	COMMENTS
C	100	40	
C	101		

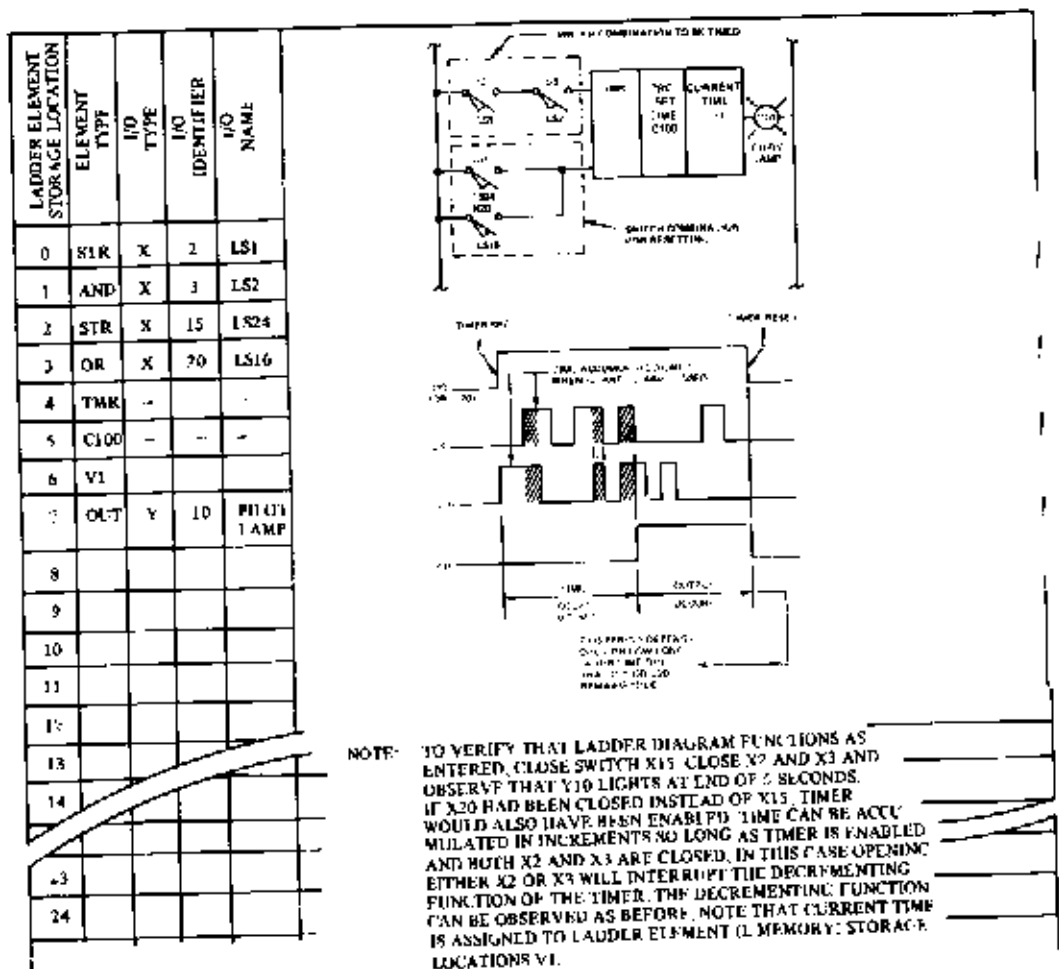
USER MEMORY STORAGE RECORD			
V AREA	LOCATION NO	VALUE	COMMENTS
V	2	*	CURRENT TIME
V	2		

Figure 5-25-2B Delay On Timer For Two Elements, Ladder Element and User Memory Storage Records

To enter the timer configuration shown in the ladder diagram of Figure 5-25-2B, press the following keys in the order given.

CLR    ATX    1    STEP    YES    ENTER  
 YES    ENTER  
 YES    ENTER  
 0    YR    X    3    ENTER  
 STR    X    4    ENTER  
 ENR    NO    STEP    C    1    0    0    STOP    V    1    ENTER  
 GUL    V    8    ENTER  
 CLR    C    1    0    0    =    4    0    ENTER

Verify this timer function as described in Figure 5-25-2B. Delay-on timers can be set and activated by almost any combination of input devices; the combination is dependent only upon the task to be performed. Figure 5-25-2C shows a ladder diagram in which the duration of closure of two series-connected elements is timed; the timer set/reset function is provided by two parallel-connected elements. The dynamics of the delay-on timer contained in the ladder diagram shown in Figure 5-25-2C is as follows: Timer is enabled when switch X15 (or X20) is closed and reset when X15 and X20 are opened. After timer is enabled, it is activated when both X2 and X3 are closed. Opening X2 or X3 will interrupt timer active time, but so long as timer is set, time is accumulated each time both X2 and X3 are closed. To enter the ladder diagram shown in Figure 5-25-2C, press the following keys in the order given:



USER MEMORY STORAGE RECORD			
C AREA	LOCATION NO	VALUE	COMMENTS
C	100	50	PRESEMI TIME
C	101		

USER MEMORY STORAGE RECORD			
V AREA	LOCATION NO.	VALUE	COMMENTS
✓	1	*	CURRENT TIME
✓	2		

Figure 5 25 2C. Ladder Element and User Memory Storage Records For Timer Function

CLR AUX 1 STOP YES ENTER

YES ENTER

YES ENTER

0 STR X 2 ENTER

AND X 3 ENTER

STR X 1 5 ENTER

JR. X 2 0 ENTER

TMR OPT STEP C 1 0 0 STEP V 1 ENTER

OUT Y 1 0 ENTER

LR C 1 0 0 = 5 0 ENTER

### 5 25 3 DELAY OFF TIMERS

A delay off timer is used where it is desired to turn a device on immediately after a switch is closed and off at a delayed time after the switch is opened. Figure 5 25 3A shows a diagram of a delay-off timer in which the same input switch starts and resets the timer. In this timer configuration the output Y1 is complemented with the NOT function; this is true also of the input switch X0. To implement the delay-off timer of Figure 5 25 3A press the following keys in the order given:

CLR V 1 STEP YES ENTER

YES ENTER

YES ENTER

0 STR NOT X 0 ENTER

STR NOT X 0 ENTER

TMR NO STEP C 1 0 0 = 5 0 ENTER

OUT NOT Y 1 ENTER

CLR C 1 0 0 = 5 0 ENTER

Set switch X0 on the simulator first to CLOSED and Y1 will come on; then set X0 to OPEN and output lamp Y1 will go out at the end of the 5 seconds.



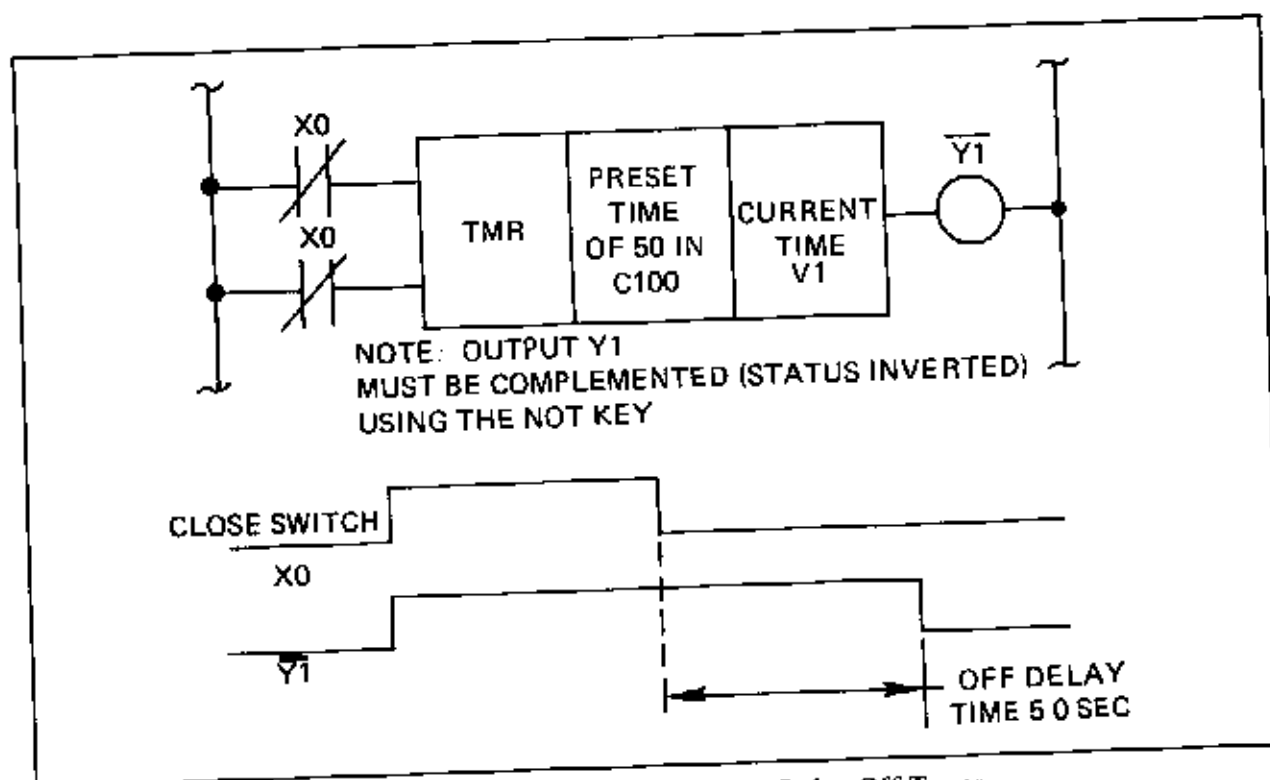


Figure 5.25.3A Simple One-Input Delay-Off Timer

## 5.25.4 HOW TO EXTEND TIMERS

**Extending Timer Time By Addition (Cascade Method).** The maximum time span for a single timer is 54.6 minutes with a tolerance of  $\pm 0.1$  second. One method of creating longer time spans is to cascade timers as shown by the ladder diagram in Figure 5.25.4A. In this case, closure of switch X11 is the event to be timed, and switch X14 functions as the timer enable/reset event. To enter the two timers in cascade as shown by the ladder diagram in Figure 5.25.4A, press the following keys in order given:

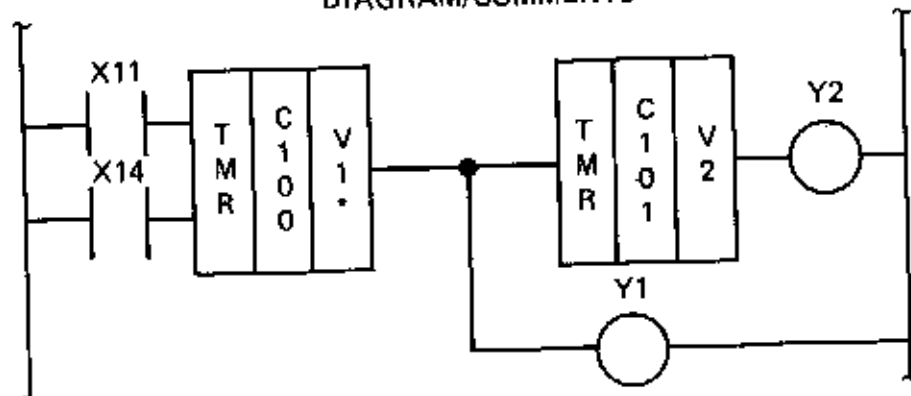
CLR	APX	1	STFP	YES	FNTR
YES	FNTR				
YES	FNTR				
0	STR	X	1	1	FNTR
STR	X	1	4	FNTR	
TMR	NO	STFP	C	1	0 0
OUT	Y	1	FNTR		

TMR MC STOP  
 C 1 0 1 STOP  
 V 2 ENTR  
 DIT X 2 ENTR  
 CLR C 1 0 0 = 3 0 ENTR  
 = 4 0 ENTR

Verify timer operation as described in Figure 5.25.4A.

**Extending Timer Time By Multiplication (Counter Method).** A second method of extending timer time is to use a counter in tandem with a timer to effectively multiply the timer output. Figure 5.25.4B shows a diagram of the counter method of extending timer. Note that the inverted status of CR1 (an internal FLAG inverted with the NOT function in this case) is utilized as a reset input to the timer. CR1 output is also an input to the counter and represents the item to be counted. The inverted output of the counter Y0 is also one of the timer reset inputs; Y0 causes the counter/timer to stop and reset after it has timed out. This arrangement will cycle at a period determined by multiplying the present time figure (in tenths of seconds) by the preset count figure. In this case the total time equals 50 tenths of seconds x 4 to obtain 20 seconds. To enter the extended timer shown in Figure 5.25.4B press the following keys in the order given:

# DIAGRAM/COMMENTS



Closing X14 enables (activates) both times. Y1 will be energized when X11 has been closed a total 3 seconds. Y2 at 7 seconds.

## L MEMORY

LOCATION	PROGRAM
0	STR X11
1	STR X14
2	TMR
3	C100 (Preset Location)
4	V1 (Current Location)
5	OUT Y1
6	TMR
7	C101 (Preset Location)
8	V2 (Current Location)
9	OUT Y2

## C MEMORY

LOCATION	VALUE	
C100	30	First timer's 3 seconds preset value
C101	40	Second timer's 4 seconds preset

## V MEMORY

LOCATION	VALUE	
V1	*	First timer's current value. Equal to 30 at preset, times down to 0 the Y1 energized
V2	*	Second timer's current value equal to 40 (4 seconds) when reset or preset and equals 0 when timed out and Y2 energized

Figure 5.25.4A. Extended Timer Using Cascade Method

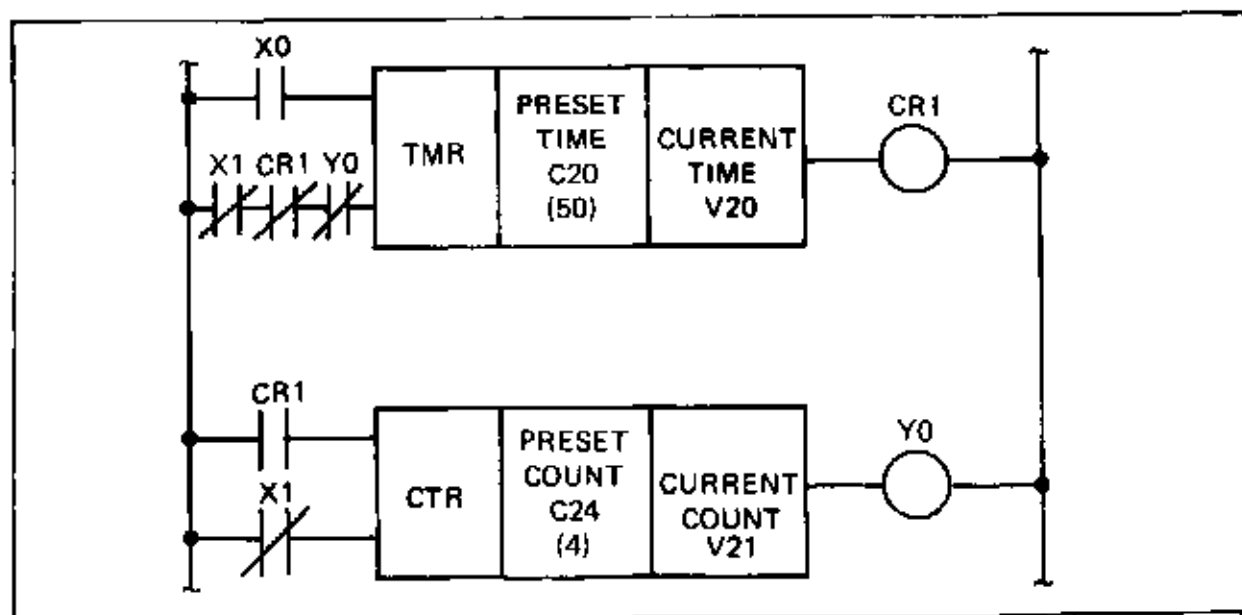


Figure 5.25 4B Extended Timer Using Counter Method

```

[CLR] [X0] [1] [STEP] [V20] [50] [0] [STR] [X] [0]
[CR] [ENTR] [STR] [NOT] [X] [1] [PSTR] [AND] [NO] [CR] [1] [ENTR]
[AND] [NOT] [Y] [0] [PSTR] [TMR] [C] [2] [0] [STEP]
[V] [2] [0] [ENTR] [OUT] [CR] [1] [ENTR] [STR] [CR] [1] [ENTR] [STR]
[NOT] [X] [1] [ENTR] [CTR] [NO] [STEP] [C] [2] [4] [STEP] [V] [2] [1]
[ENTR] [OUT] [Y] [0] [ENTR] [CLR] [C] [2] [0] [=] [5] [0] [ENTR]
[=] [4] [ENTR]
  
```

To verify the counter method of extending time, set switch X0 on the simulator to CLOSED. In 20 seconds the output lamp Y0 will light. Closing X1 will reset the timer and counter.

### 5.25.5 PROGRAMMING INTEGER MOVE FUNCTION

There are often conditions in machine or process control when different modes of operation require a change in parameters in an equation or a new value in a Timer or Counter. The MOVE key provides the ability to MOVE an integer from one memory location to another.

Figure 5.25 5A describes an example where the MOVE function is utilized to externally select the preset value for a delay on timer. MOVE is a conditional operation; that is, it occurs as a result of continuity through a line of logic. When power flow is on, a MOVE occurs.

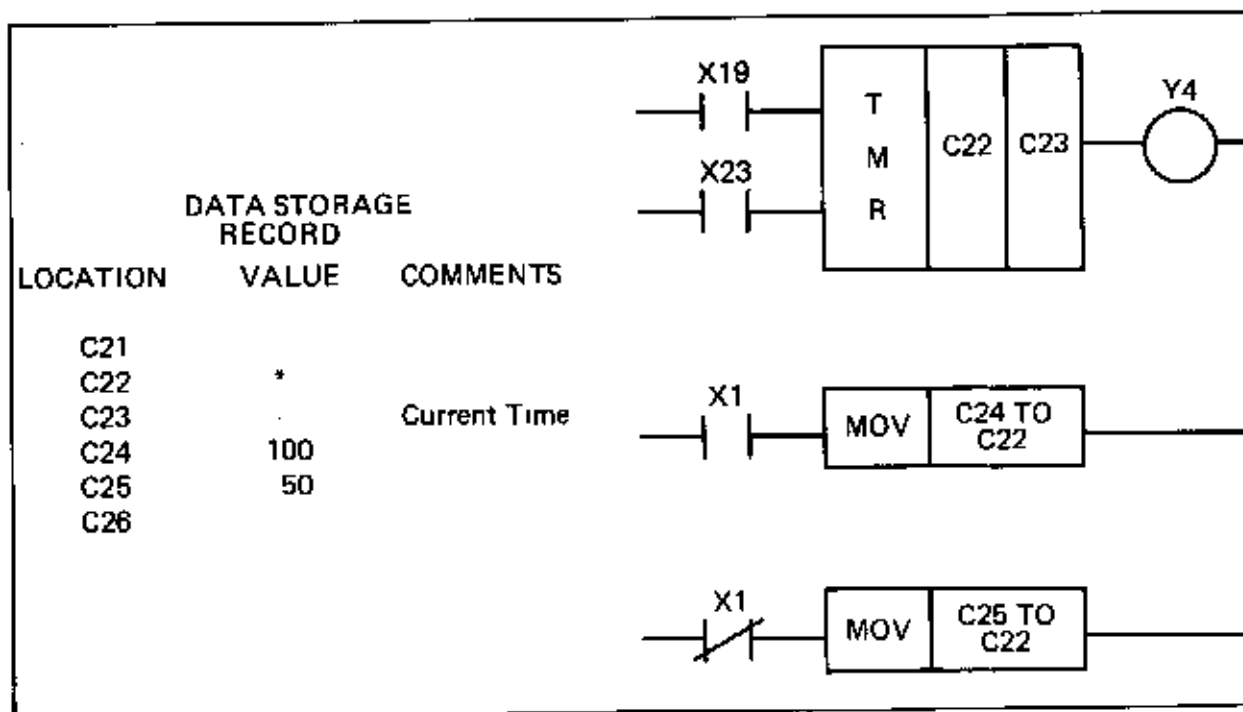












Figure 5 25.5A. Ladder Diagram and Storage Record for Ladder Logic Integer  Function

Input X1 has been selected to control the MOVE in Figure 5 25.5A. When X1 is open, a MOVE occurs from C25 to C22. When X1 is closed, the value in C24 is moved into C22. Note that the values are always maintained in C25 and C24.

To enter the example of Figure 5 25.5A press the following keys in the order given

KEY SEQUENCE	DISPLAY	NOTE
   	CLEAR L?	CCU in Start-up.
	PRESS ENTR TO CLEAR	
	CLEAR V?	Logic memory and image register cleared.
	PRESS ENTR TO CLEAR	
	CLEAR C?	Variable data storage cleared.
	PRESS ENTR TO CLEAR	

KEY SEQUENCE	DISPLAY	NOTE
<b>ENTR</b>	READY (START)	Constant memory cleared.
<b>5</b> <b>STR</b> <b>X</b> <b>1</b> <b>9</b>	5 STR X19	
<b>ENTR</b> <b>STR</b> <b>X</b> <b>2</b> <b>3</b>	6 STR X23	
<b>ENTR</b> <b>TMR</b>	7 TIMER PROTECT?	
<b>NO</b>	TIMER	
<b>STEP</b> <b>C</b> <b>2</b> <b>2</b>	8 PRE C22	Preset value address.
<b>SLIP</b> <b>C</b> <b>2</b> <b>3</b>	9 CUR: C23	Current value address.
<b>ENTR</b> <b>OUT</b> <b>Y</b> <b>4</b>	10 OUT Y4	
<b>ENTR</b> <b>STR</b> <b>X</b> <b>1</b>	11 STR X1	
<b>ENTR</b> <b>MOVE</b>	12 MOVE	Because a MOVE requires three instructions to be complete, the <b>ENTR</b> key is pressed only after all three have been defined. As the displayed prompting message indicates, the MOVE FROM address is defined first, followed by the MOVE to address.
<b>STEP</b> <b>C</b> <b>2</b> <b>4</b>	13 FROM: C24	
<b>SLIP</b> <b>C</b> <b>2</b> <b>2</b>	14 TO: C22	
<b>ENTR</b> <b>STR</b> <b>NOT</b> <b>X</b> <b>1</b>	15 STR NOT X1	
<b>ENTR</b> <b>MOVE</b>	16 MOVE	
<b>C</b> <b>2</b> <b>5</b>	17 FROM: C25	
<b>SLIP</b> <b>C</b> <b>2</b> <b>?</b>	18 TO: C22	
<b>ENTR</b> <b>CLR</b> <b>C</b> <b>2</b> <b>4</b> <b>=</b> <b>1</b> <b>0</b>	C24 = 100	
<b>0</b>		
<b>ENTR</b> <b>=</b> <b>5</b> <b>0</b> <b>ENTR</b>	C25 = 50	
<b>=</b> <b>5</b> <b>0</b> <b>ENTR</b>	C26 = 50	

To verify the MOVE function open switches X19, X23 and X1. To read the TMR current value press the **CLR** **C** **2** **3** and **READ** keys. Close switches X23 and X19. TMR operates from a 5.0 sec. preset because X1 is open and the 5.0 sec. value in location C25 is moved into the preset location C22. Open switch X19, X23, and close switch X1. The preset time is now moved from C24 to C22 and the TMR operates with a 10.0 second preset. Close X19 and X23 and verify correct TMR operation.

## 5.25.6 PROGRAMMING INTEGER COMPARE FUNCTION

There are many occasions in automatic control when it is desirable to know if a value is equal to, less than, or greater than a certain specified limit. The PM550 compare function uses the power flow to determine its mode (equal to or greater than or equal to), then sets a new power flow based on the result of its comparison of two specified values.

Figure 5.25.6A illustrates how the current value of two counters can be compared. In this case CR40 was arbitrarily selected to set the power flow and determine the mode of comparison. The compare function requires 3 locations to be defined.

COMP  
FIRST ADDRESS  
SECOND ADDRESS

In all cases, the second value is compared to the first. Note that just as in TMR's and CTR's the address (C, V, or A) is defined, not a real numeric value. That address stores the value to be compared. To enter the example of Figure 5.25.6A press the following keys in the order given:

KEY SEQUENCE	DISPLAY	NOTE
<b>TR</b>	READY (RUN)	
<b>0</b> <b>ATR</b> <b>X</b> <b>1</b>	0 STR X1	
<b>ENTER</b> <b>CTR</b> <b>X</b> <b>2</b>	1 STR X2	
<b>ENTER</b> <b>CTR</b> <b>PROTECT</b>	2 CTR PROTECT	
<b>NO</b>	COUNTER	
<b>STOP</b> <b>NO</b> <b>C</b> <b>2</b> <b>1</b>	3 PRE C21	Preset value address.
<b>STOP</b> <b>C</b> <b>2</b> <b>2</b>	4 CTR C22	Current value address
<b>ENTER</b> <b>OUT</b> <b>Y</b> <b>1</b>	5 OUT Y1	Energized when value in C22 = 21

KEY SEQUENCE	DISPLAY	NOTE
ENTER STR X 3	6 STR X3	
ENTER STR X 4	7 STR X4	
ENTER	8 CTR PROTECT	
NO	COUNTER	
STEP C 2 3	9 PRF C23	
STEP C 2 4	10 CUR C34	
ENTER OUT Y 2	11 OUT Y2	
ENTER STR C R 4 0	12 STR CR40	With CR40 off compare is for equality
ENTER COMP	13 COMPARE	
STEP C 2 2	14 COMP C22	
STEP C 2 4	15 WITH C24	
ENTER OUT Y 3	16 OUT Y3	Y3 is on if value C22 = C24
ENTER STR NOT C R 4 0	17 STR NOT CR40	With CR40 on compare is for $\neq$
ENTER COMP	18 COMPARE	
STEP C 2 2	19 COMP C22	
STEP C 2 4	20 WITH C24	
ENTER OUT Y 4	21 OUT Y4	Y4 is on if C22 $\neq$ C24
ENTER CLA ( 2 1 = 1 0	C21 = 100	
0		
ENTER STEP C 2 <del>3</del> = 2 0	C22 = 200	
0	C23	
ENTER		

Verify compare operation as described in Figure 5.25 6A.



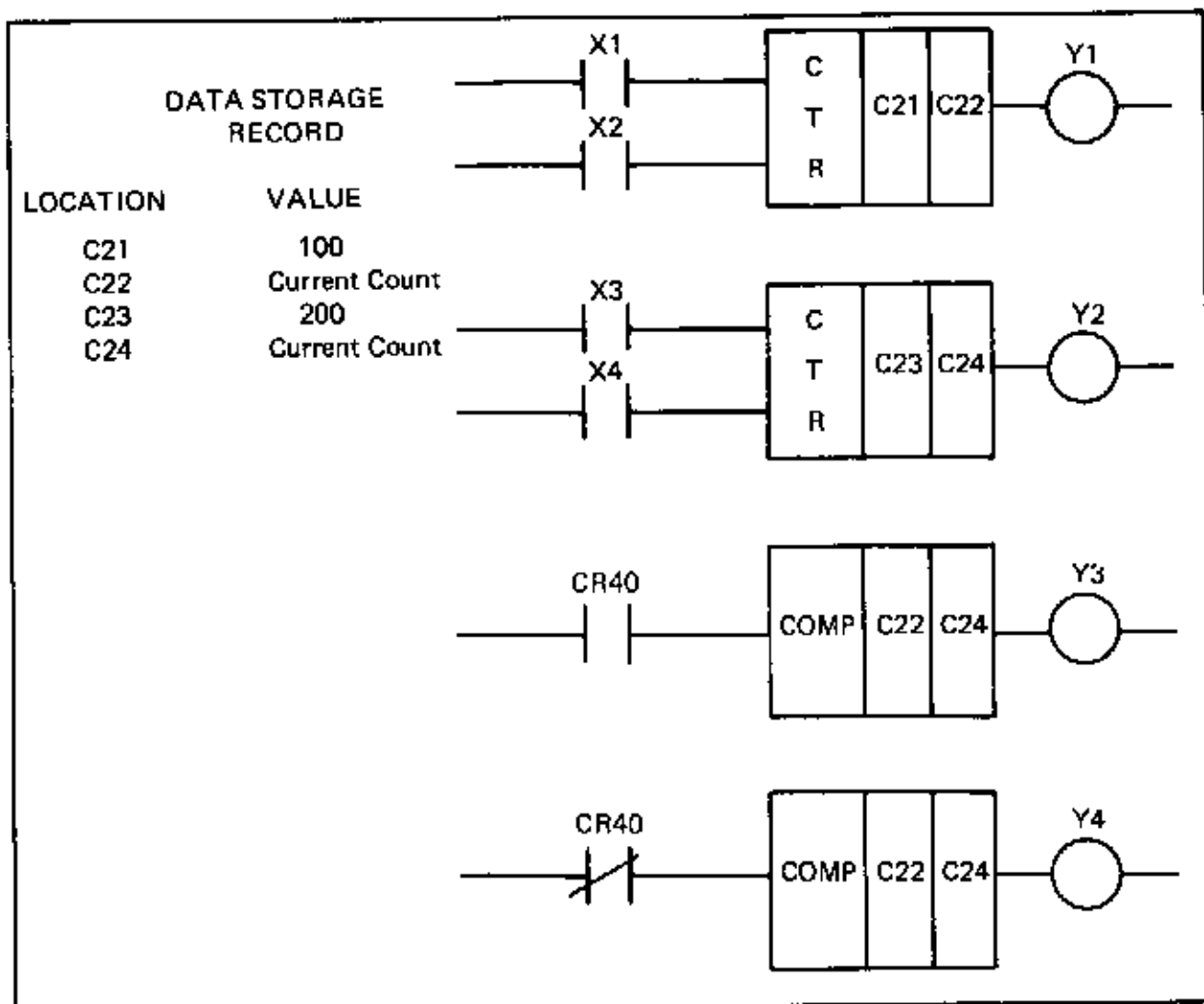


Figure 5.25.6A. Ladder Diagram and Storage Record for Compare Function

Because CR40 is not used as an output in this program, when CR40 is programmed as a normally open input its power flow is always 'off'. When programmed as a normally closed input, its power flow is always 'on'. This power flow determines the mode of comparison (Figure 5.25.6A)

To verify the compare function open switches X1 through X4. The counters are reset and  $C22 = C24$  which energizes outputs Y3 and Y4. Closing switches X3 and X4 increments C24 to a value of 1, while  $C22 = 0$ . Y4 is then energized because  $C24 > C22$  while Y3 is turned off. Close switch X2, then toggle X1 twice. C24 is now less than C22 and Y4 turns off.

Figure 5.25.6B, displays how Power Flow can be utilized to select the type of comparison to be made. The power flow after the comparison is used to indicate the result of the compare. Unlike the move function which operates conditionally, the compare function is updated on every memory scan of the PM550. The compare can be placed in the field of an MCR so that its output can be externally controlled.

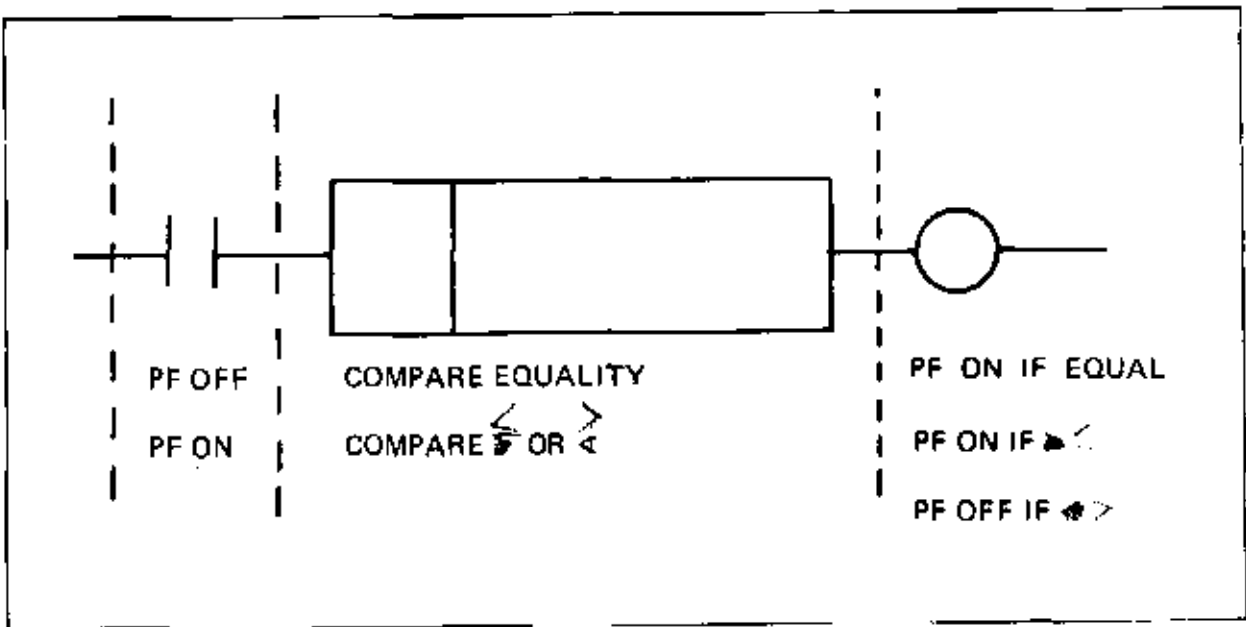


Figure 5-25-6B Power Flow Used to Select Type of Compare

### 5-25-7 PROGRAMMING INTEGER ADDITION (+)

Addition of two integer values can be programmed as shown in Figure 5-25-7A. To enter the ladder diagram press the following keys in the order given.

KEY SEQUENCE	DISPLAY	NOTE
<b>TR</b> <b>LOC</b> <b>1</b> <b>STP</b> <b>15</b> <b>ENTER</b> <b>CLR</b>	READY (START)	
<b>0</b> <b>STP</b> <b>X</b> <b>1</b>	0 STR X1	
<b>ENTER</b> <b>STP</b> <b>X</b> <b>2</b>	1 STR X2	
<b>ENTER</b> <b>CTR</b>	2 CTR PROTECTION	
<b>NO</b>	COUNTER	

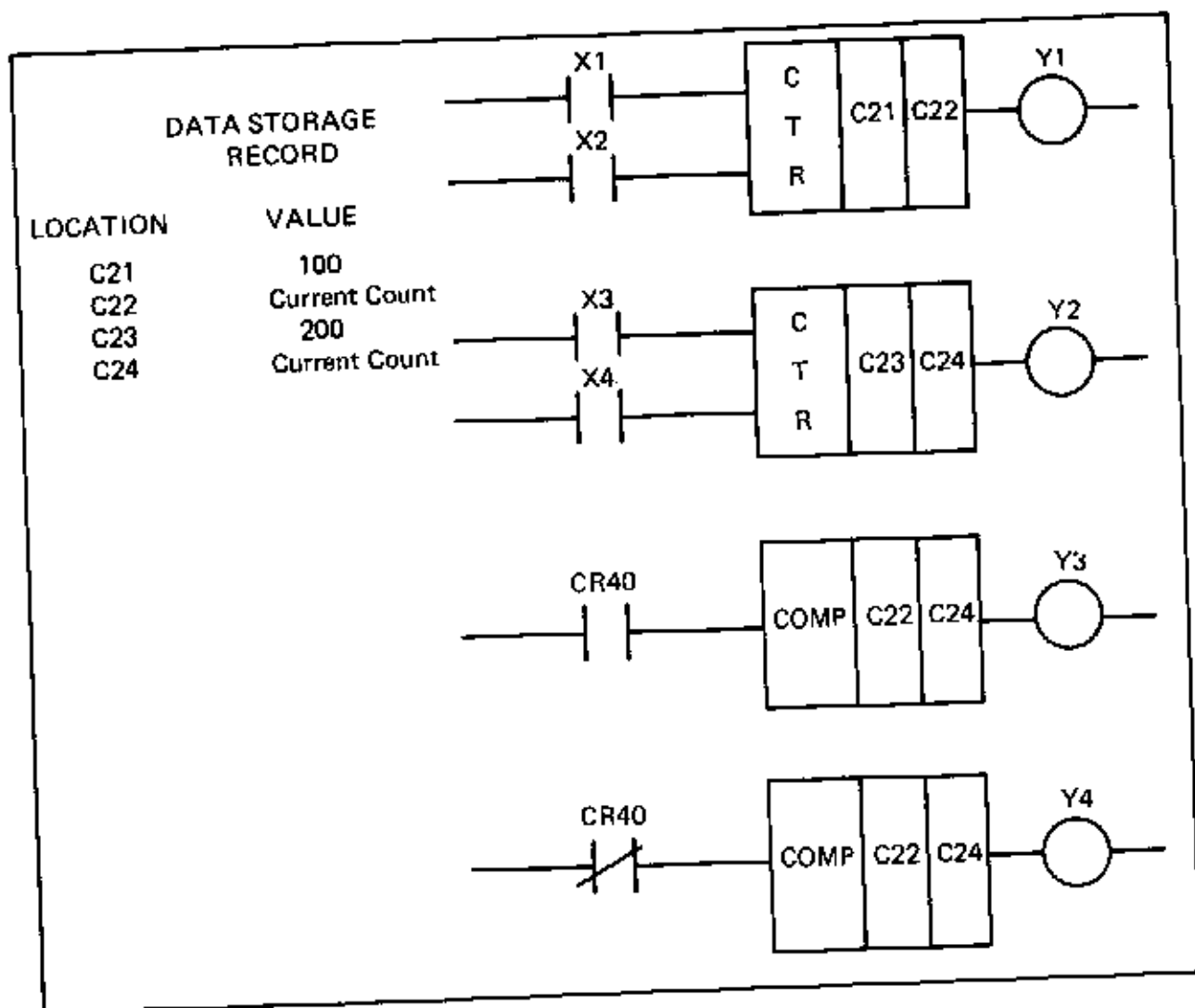


Figure 5.25.6A. Ladder Diagram and Storage Record for Compare Function

Because CR40 is not used as an output in this program, when CR40 is programmed as a normally open input its power flow is always 'off'. When programmed as a normally closed input, its power flow is always 'on'. This power flow determines the mode of comparison (Figure 5.25.6A).

To verify the compare function open switches X1 through X4. The counters are reset and C22 = C24 which energizes outputs Y3 and Y4. Closing switches X3 and X4 increments C24 to a value of 1, while C22 = 0. Y4 is then energized because  $C24 > C22$  while Y3 is turned off. Close switch X2, then toggle X1 twice. C24 is now less than C22 and Y4 turns off.

Figure 5.25.6B displays how Power Flow can be utilized to select the type of comparison to be made. The power flow after the comparison is used to indicate the result of the compare. Unlike the move function which operates conditionally, the compare function is updated on every memory scan of the PM550. The compare can be placed in the field of an MCR so that its output can be externally controlled.

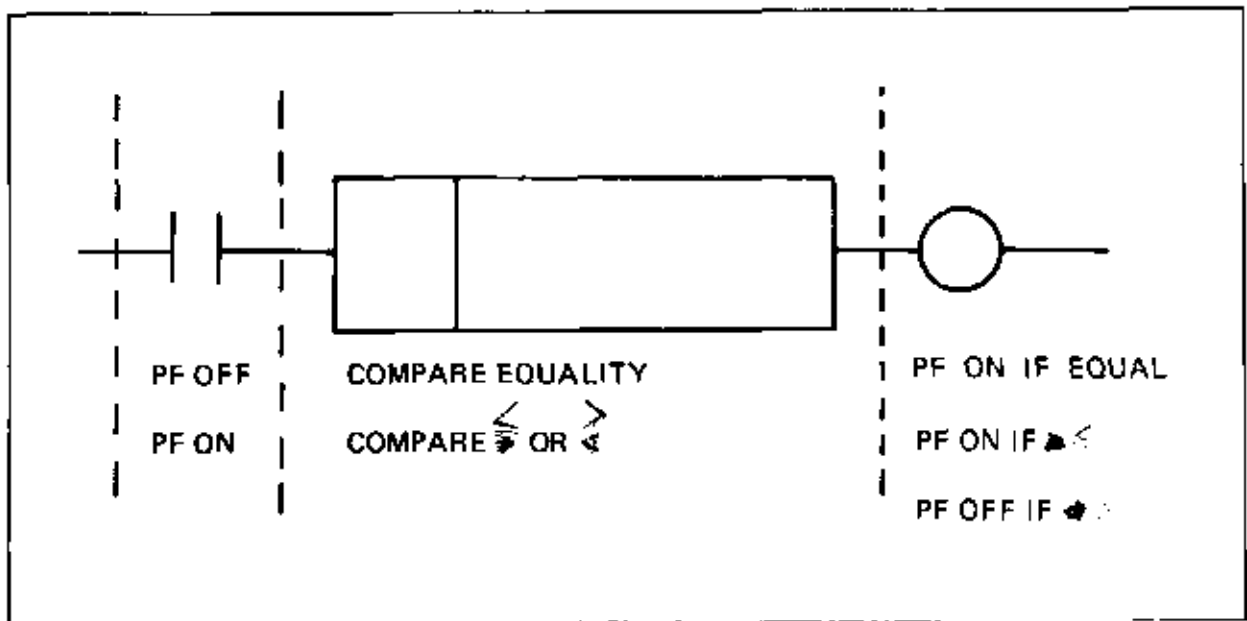


Figure 5 25 6B. Power Flow Used to Select Type of Compare

#### 5 25 7 PROGRAMMING, INTEGER ADDITION (+)

Addition of two integer values can be programmed as shown in Figure 5 25 7A. To enter the ladder diagram press the following keys in the order given

KEY SEQUENCE	DISPLAY	NOTE
CLR ALN 1 STEP YES ENTR CLR	READY (START)	
0 STR X1	0 STR X1	
ENTR STR X 2	1 STR X2	
ENTR CLR	2 CTR PROTECT <sup>9</sup>	
NU	(COUNTER	

## KEY SEQUENCE

## DISPLAY

## NOTE

STEP C 1 0 0

3 PRE: C100

STEP C 1 0 1

4 CUR: C101

ENTR OUT Y 1

5 OUT Y1

ENTR STR X 3

6 STR X3

ENTR STR X 4

7 STR X4

ENTR STR

8 CTR PROTECT?

NO

COUNTER

STEP C 1 0 2

9 PRE: C102

STEP C 1 0 3

10 CUR: C103

ENTR OUT Y 2

11 OUT Y2

ENTR STR X 5

12 STR X5

ENTR +

13 ADD

STEP C 1 0 1

14 LET: C101

STEP C 1 0 3

15 +: C103

Contents of C101 is added to  
contents of C103

STEP C 1 0 4

16 =: C104

END

CLR C 1 0 0 - 1 0

C100 = 10

The result is stored in C104

ENTR C 1 0 2 = 1 5

C102 = 15

ENTR

C103

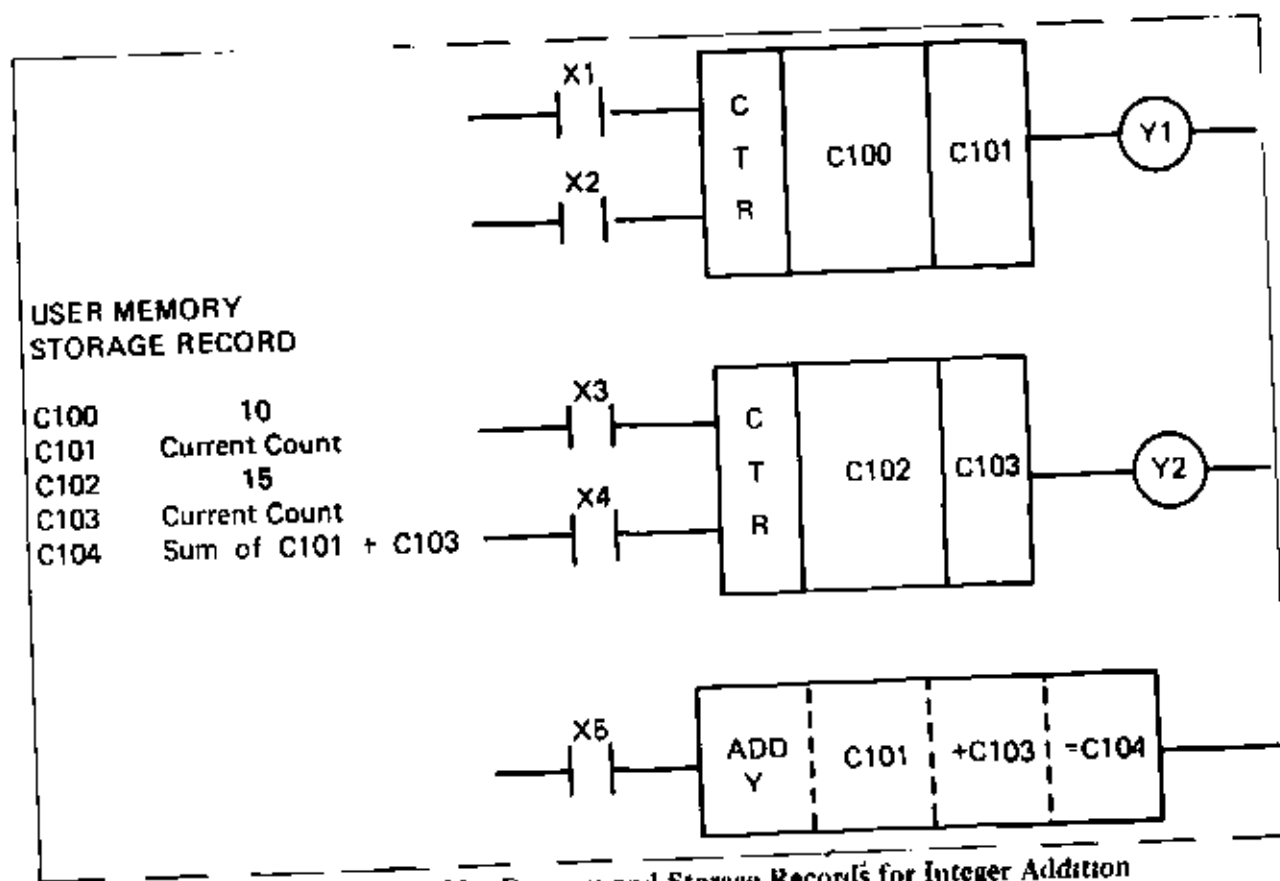


Figure 5 25 7A Ladder Diagram and Storage Records for Integer Addition

Integer addition requires 4 locations to be complete. **ENTR** key is pressed to enter the equation into Ladder Logic locations 13–16 after all 4 items are defined. Like **IMR**, **CTR**, **MOVE** and **COMPARE** functions, integer addition uses indirect addressing to define the values of its parameters. An address (**V** or **A**) must be defined when entering an equation, never a numeric value.

The integer addition function provides the ability to find the sum of 2 integer values (no decimal). It operates conditionally, that is it is calculated dependent on input power flow.

The output power flow of an add indicates an overflow if it is equal to  $\frac{0}{\text{zero}}$  and normal if it is equal to  $\frac{1}{\text{normal}}$ . Results are invalid if an overflow is indicated.

Special function 8 (user math) provides for the generation of non integer math equations which are calculated on a conditional basis. User math is described in Chapter 6.

To verify Integer addition open switches X1 through X4. Close X5, press **ENTR**, **C**, **1**, **0**, **4** and **RIAD** keys on programmer. Close X2 and X4. Each closure of X1 or X3 will now increment C104 by one count. Note that data is read from data storage area, not ladder logic storage where equation was entered. Open X5 and observe that C104 holds its last value, even if C101 and C103 are incremented.

## 5.25.8 ENTERING INTEGER SUBTRACTION (-)

Like integer ADD, this function is a conditional operation programmed in ladder logic. It provides the ability to find the difference between two signed integer values.

To enter the ladder diagram of Figure 5.25.8A, press the following keys in the order given:

KEY SEQUENCE	DISPLAY	NOTE
[RUN]	READY (RUN)	
[0] [STR] [X] [1]	0 STR X1	
[ENTR] [STR] [X] [2]	1 STR X2	
[ENTR] [CTR]	2 CTR PROTECT?	
[NO]	COUNTER	
[STEP] [C] [1] [9]	3 PRE: C19	
[STEP] [C] [2] [0]	4 CUR: C20	
[ENTR] [OUT] [Y] [1]	5 OUT Y1	
[ENTR] [STR] [X] [3]	6 STR X3	
[ENTR] [STR] [X] [4]	7 STR X4	
[ENTR] [CTR]	8 CTR PROTECT?	
[NO]	COUNTER	
[STEP] [C] [3] [0]	9 PRE: C30	
[STEP] [C] [4] [0]	10 CUR: C40	
[ENTR] [OUT] [Y] [2]	11 OUT Y2	
[ENTR] [STR] [X] [5]	12 STR X5	
[ENTR] [-]	13 SUBTRACT	

KEY SEQUENCE	DISPLAY	NOTE
STEP C 2 0	14 LET: C20	$C20 - C40 = C50$
STEP C 4 0	15 =: C40	
STEP C 5 0	16 =: C50	
ENTR CLR C 1 9 = 1 5	C19 = 15	
ENTR CLR C 3 0 = 1 0	C30 = 10	
PNTR	C31	

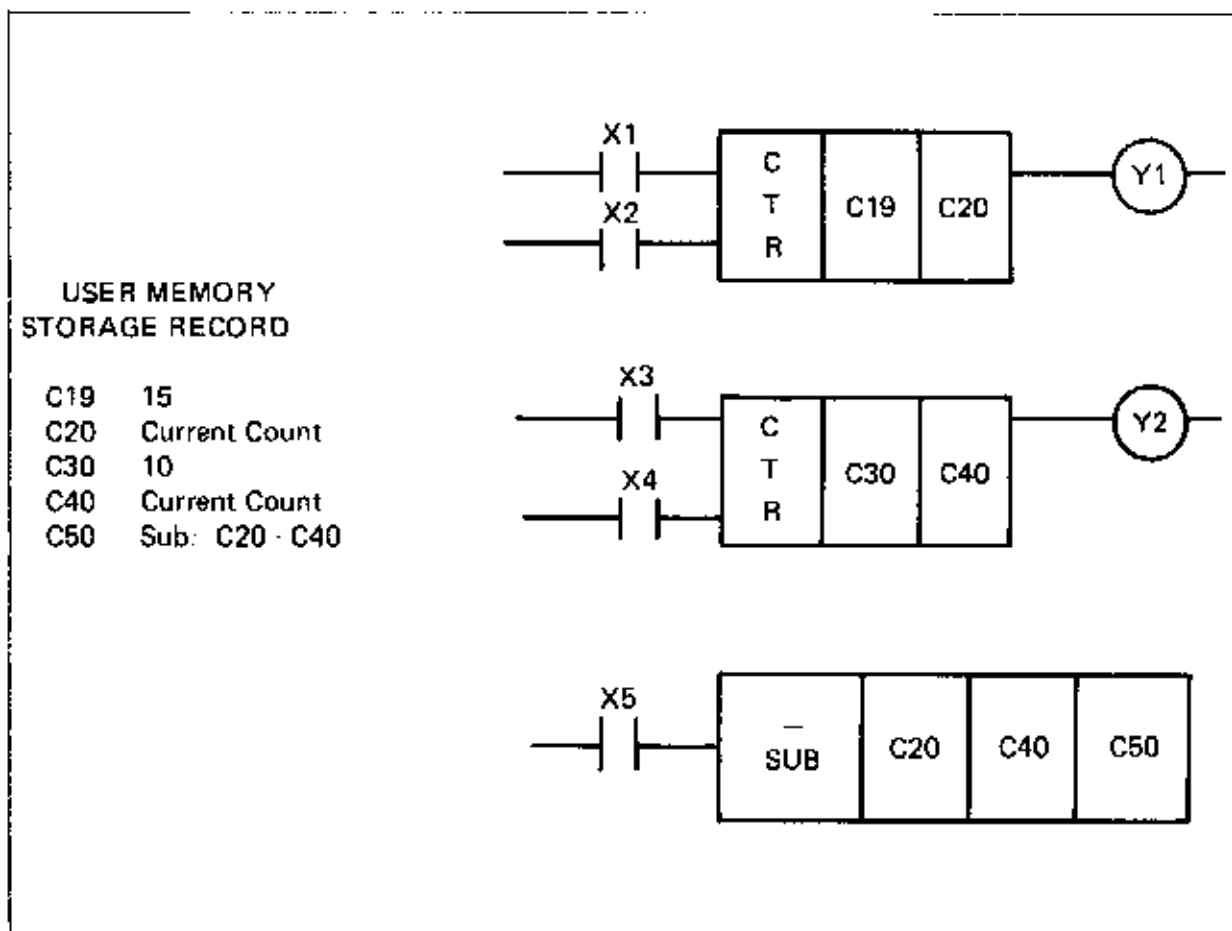


Figure 5-25-8A. Ladder Diagram and Storage Records for Integer Subtract



To verify Integer subtract, open switches X1 through X4. Close X5. Press **CLR**, **C**, **5**, **0**, and **READ** keys on programmer. Close X2 and X4 to enable counters. Close X3 five times noting that C50 increments from 0 to 5. Each closure of X1 will now decrement C50 by 1 count.

### 5.25.9 CHANGING MULTI-WORD FUNCTIONS AFTER ENTRY

Once a multi-word function has been entered into ladder logic it may be necessary to read a word, change a word, delete or erase the entire function.

Figure 5.25.9A displays the ladder logic and storage data for a timer function. Press the following keys to enter the timer into CCU memory:

**CLR** **0** **STR** **X** **1** **ENTR**  
**STR** **X** **2** **ENTR**  
**TMR** **NO** **STR** **C** **1** **9** **STEP** **V** **2** **ENTR**  
**DN** **Y** **1** **ENTR**  
**LR** **C** **1** **9** **=** **1** **0** **0** **ENTR**

While entering the timer function (As in all multi-word functions) the programmer displays prompting messages as an aid in the definition of terms. These prompting messages appear only during entry and are not displayed during a memory READ. To read Location No. 4 press

**CLR** **4** **READ**

The display indicates 4 V2

To change the timer current word location from V2 to C20 press **CLR** **C** **2** **0** **ENTR**

The display indicates 4 C20

Close X1 and X2 while reading location C20 to verify timer operation.

Ladder Diagram Storage Record	User Memory Storage Record	User Memory Storage Record
0 STR X1		
1 STR X2		
2 TIMER		
3 C19		
4 V2	C 19 = 100	V1
5 OUT Y1	C 20	V2 Current Word

**Figure 5.25.9A. Ladder Diagram and Storage Record for Timer Function**

Figure 5.25.9B shows the ladder diagram storage record before and after the delete operation. Note that all three words of the timer function were deleted.

A DLTE may occur under the following conditions:

- 1) Any ladder logic address of a multi-word function can be selected.
- 2) All locations of a multi-word function will be deleted simultaneously.

Ladder Diagram Storage Record Before DLTE	Ladder Diagram Storage Record After DLTE
0 STR X1	0 STR X1
1 STR X2	1 STR X2
2 TIMER	2 OUT Y1
3 C19	3
4 C20	4
5 OUT Y1	5

Figure 5.25.9B. Delete of Multi-Word Function

To insert a counter function into the ladder logic memory press the following keys:

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	
<input type="button" value="2"/>	2	Selects location for 1st word of counter function
<input type="button" value="CTR"/>	2 COUNTER	All three words are defined before insert is pressed.
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="3"/>	3 PRE: V3	
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="4"/>	4 CUR: V4	
<input type="button" value="INSERT"/>	4 CUR: V4	All three elements are inserted into memory.

Press  ,  ,  and  through memory to verify that the insert was performed correctly. Refer to Figure 5.25.9C for the correct Ladder Diagram Storage Record. Note that all other words in L memory were pushed down by 3 memory locations.

Ladder Diagram Storage Record Before Insert	Ladder Diagram Storage Record After Insert
0 STR X1	0 STR X1
1 STR X2	1 STR X2
2 OUT Y1	2 COUNTER
3	3 V3
4	4 V4
5	5 OUT Y1

Figure 5.25.9C Insert of Multi-Word Function

## 5.26 READING AND WRITING THE IMAGE REGISTER

Individual locations in the image register can be written or read from the programmer. This feature is useful for both program and I/O check out.

To write a one to the image register (Y or CR), press the following keys:

To write a zero to Y2, press the following

To read the image register press the following keys

The value of CR27 will be displayed with 1 = on and 0 = off.

Both  and  keys may be used when reading or writing the image register.

## CHAPTER 6

### ENTERING SPECIAL FUNCTIONS

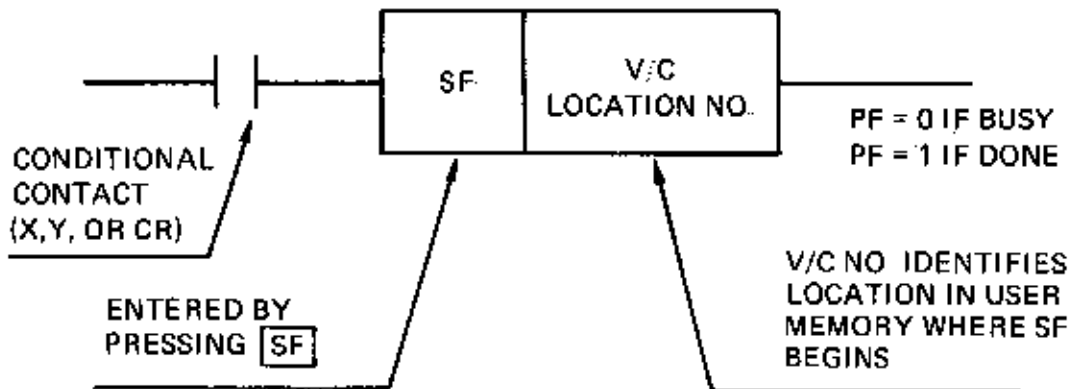
#### 6.0 PRINCIPLE OF OPERATION

As previously indicated, all operations in ladder logic are limited to using integer (no fraction or decimal) values. This applies to ladder logic addition and subtraction as well as move and compare. The executive processor must be called upon to perform any operation in which a non integer is used. In addition, the executive processor can perform a broad variety of subroutines designated as **SPECIAL FUNCTIONS**. The table in Figure 6.0A lists the available Special Functions (SF).

All SF's follow a basic format for implementation. Figure 6.0B describes SF operation.

##### Ladder Logic Entry

The Special Function Request Line mentioned in Figure 6.0B has the following format:



This line in ladder logic identifies

- 1) When it is necessary to perform the SF – by the conditional contact power flow. This contact must remain closed until the SF has been completed.
- 2) Where the particular SF is located in user memory – by the V/C location number.
- 3) Status of SF via output power flow.

SF NO	TITLE	DESCRIPTION
SF0	ENTRY POINT	PROVIDES ALTERNATE ENTRY POINT FOR A SPECIAL FUNCTION
SF1	BINARY TO BCD	CONVERTS A POSITIVE BINARY NUMBER LESS THAN 10000 TO A POSITIVE 4 DIGIT BCD CODED NUMBER
SF2	BCD TO BINARY	CONVERTS A POSITIVE 4 DIGIT BCD CODED NUMBER TO A POSITIVE BINARY NUMBER.
SF3	SCALE	CONVERTS A SCALED BINARY NUMBER * TO A FLOATING POINT NUMBER IN ENGINEERING UNITS
SF4	UNSCALE	CONVERTS A FLOATING POINT NUMBER IN ENGINEERING UNITS TO A SCALED BINARY NUMBER. *
SF5	SQUARE	SQUARES AN INTEGER OR FLOATING POINT NUMBER
SF6	SQUARE ROOT	TAKES THE SQUARE ROOT OF A POSITIVE INTEGER OR FLOATING POINT NUMBER.
SF7	COMPARE	COMPARES TWO INTEGERS OR FLOATING POINT NUMBERS FOR < ≤, =, ≥, >, AND ≠
SF8	MATH	PERFORMS CHAIN CALCULATIONS (+, X, AND -) ON INTEGER OR FLOATING POINT NUMBERS
SF9	SEQUENTIAL DATA TABLE	OUTPUTS SUCCESSIVE ENTRIES OF A TABLE EACH TIME IT IS EXECUTED
SF10	CORRELATED DATA TABLE	LOCATES ENTRY IN TABLE 1 THAT IS ≥ INPUT AND OUTPUTS CORRESPONDING ENTRY FROM TABLE 2
SF11	ASCII STRING	OUTPUTS ASCII CODED MESSAGE STORED IN USER MEMORY TO RS 232 PORTS 1 OR 2
SF12	SYNCHRONOUS SHIFT REGISTERS	SPECIFY A REGISTER OF ANY LENGTH SHIFT OCCURS EVERY TIME SF12 CALLED
SF13	ASYNCHRONOUS SHIFT REGISTERS INPUT	INPUTS TO ASYNCHRONOUS SHIFT REGISTER
SF14	ASYNCHRONOUS SHIFT REGISTERS-OUTPUT	OUTPUTS FROM ASYNCHRONOUS SHIFT REGISTER

\* A SCALED BINARY NUMBER HAS A RANGE OF 0 TO 32000 WHERE 0 CORRESPONDS TO THE LOWER ENGINEERING UNITS LIMIT AND 32000 CORRESPONDS TO THE UPPER ENGINEERING UNITS LIMIT. I.E. IF A FLOW RANGES FROM 2 TO 30 GPM THE SCALED BINARY VALUE OF 0 = 2 GPM THE SCALED VALUE OF 16000 = 18000/32000 (30-2)+2 = 16 GPM and 32000 = 30 GPM

Figure 6 0A Special Function Description Table

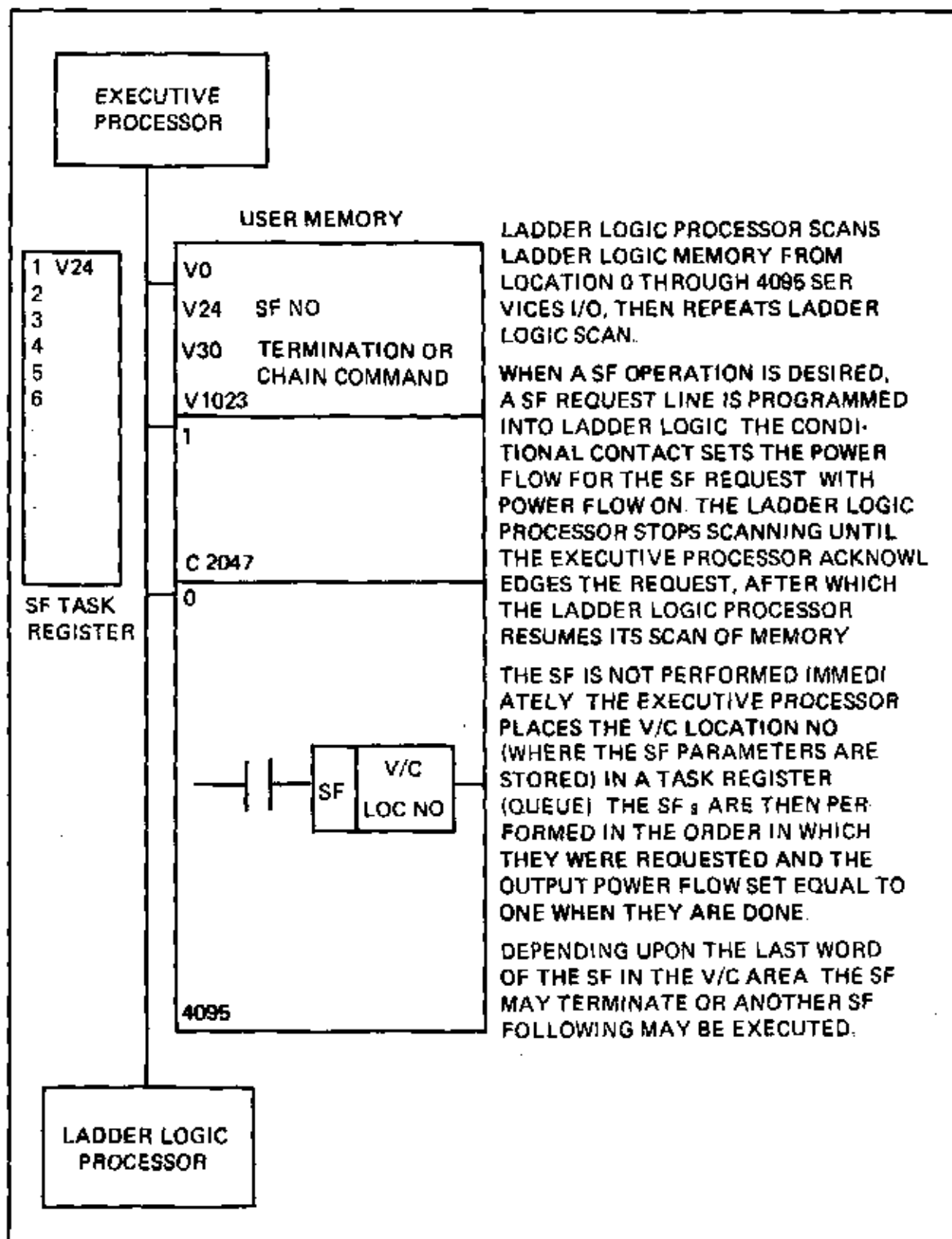


Figure 6.0B Special Function Operation

To enter the ladder logic request line of Figure 6.0C, press the following keys:

KEY SEQUENCE	DISPLAY	NOTE
<b>CLR</b>	READY (RUN)	
<b>1 0 STR X 1 2</b>	10 STR X12	Conditional contact
<b>ENTR SF</b>	11 SF PRIORITY?	
<b>YES</b>	11 SF (P)*	
<b>STEP V 1 0 0</b>	12 AT: V100	Location of SF in data storage memory
<b>ENTR</b>	13	

\*There are two queues (or stacks) for pending SF's. One is the priority (P) queue which is accessed by answering YES. The other is the non-priority queue accessed by answering NO. Both queues are treated identically! The queue with the fewest number of special functions will run the fastest. Each queue can have a maximum of 32 SF's.

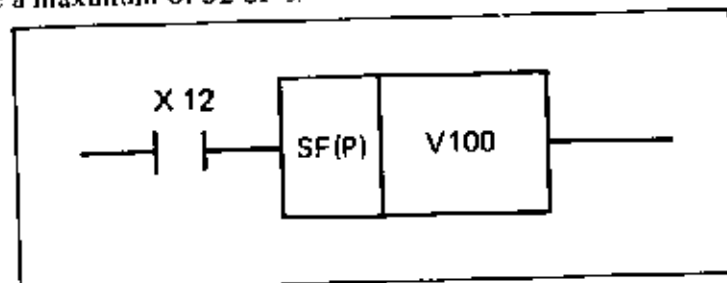


Figure 6.0C Ladder Diagram and Storage Record for Special Function Request Line

## 6.1 ENTERING NON-INTEGER VALUES

All data values used in ladder logic are integer values with a range of +32767 to -32768. Special Functions can utilize non-integer data values with a range of  $\pm 0.5397605 \times 10^{-78}$  to  $\pm 0.7237005 \times 10^{+77}$ .

All non-integer numbers require two consecutive memory locations for storage. To enter the value 29.32 into user memory press the following keys:



KEY SEQUENCE	DISPLAY	NOTE
	READY (RUN)	
	V1 = 29.32	Pressing the  key following the user memory location number (V1) identifies the value to be a non-integer. The value 29.32 is automatically stored in location V1 and V2.

V3

If the number exceeds the 10 digits plus decimal point allowed in the above example, the number can be entered in scientific notation or "E" notation. In "E" notation the first number before the "E" called the mantissa, is multiplied by ten to the second number following the "E" called the exponent (power). For example, 1.257E3 is equal to  $1.257 \times 10^3$  or 1257. See Figure 6.1A for examples. The "E" is entered using the key. i.e., 1.27E3 is entered in C20, as follows:

To read press:

"E" Notation	Equivalent Value
1.257E3	1,257
.3546E7	3,546,000
-.543E0	-.543
123 F-2	.00123
-.98E-8	-.0000000098
1.5E15	1,500,000,000,000,000

Figure 6.1A. Example of "E" Notation

READ — To read the non-integer which has been entered, press the following keys:

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	Failure to press the <input type="button" value="◀"/> key will cause the programmer to read an integer value in location V1, and the correct value of $2.9320E+01$ will not appear in the display. The non-integer requires 2 consecutive memory locations for storage. A read of the second location (V2 in the above example) will also cause invalid data to appear in the display.
<input type="button" value="V"/> <input type="button" value="1"/> <input type="button" value="◀"/>	V1	
<input type="button" value="READ"/>	V1 = 2.9320E+01	

#### STEP - BSTEP

As described in Paragraph 5.5, pressing  initiates a continuous read mode. Pressing  (or ) after  will cause the contents of the next higher (or lower) location to be displayed. When using the , , and  functions in user memory (V/C) caution must be exercised in interpreting the displayed information. When reading non-integer numbers the  key must be depressed to read each memory location.

Enter the value 96.5 into V5, and 12.5 into V7, by pressing the following keys:

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	
<input type="button" value="V"/> <input type="button" value="5"/> <input type="button" value="◀"/> <input type="button" value="="/> <input type="button" value="9"/> <input type="button" value="6"/> <input type="button" value="◀"/> <input type="button" value="5"/>	V5 = 96.5	Pressing <input type="button" value="ENTER"/> will automatically put V7 on the display.
<input type="button" value="ENTER"/> <input type="button" value="."/> <input type="button" value="="/> <input type="button" value="1"/> <input type="button" value="2"/> <input type="button" value="."/> <input type="button" value="5"/>	V7 = 12.5	
<input type="button" value="ENTER"/>	END	

To read the values press **CLR** **V** **5** **\*** **READ** Display indicates: V5 = +9.6500E+01  
 Press **STEP** **\*** **READ** Display indicates: V7 = +1.2500E+01

ENTER the value 100 into V10 and 5 into V11 by pressing the following keys

KEY SEQUENCE	DISPLAY	NOTE
<b>CLR</b>	READY (RUN)	
<b>V</b> <b>1</b> <b>0</b> <b>=</b> <b>1</b> <b>0</b> <b>0</b>	V10 = 100	
<b>ENTER</b> <b>=</b> <b>5</b>	V11 = 5	
<b>READ</b>	V12	
To read the values press <b>CLR</b> <b>V</b> <b>1</b> <b>0</b> <b>READ</b> Display indicates: V10 = 100. Press <b>STEP</b> , display indicates: V11 = 5		

#### NOTE

Because the first value read was an integer, stepping causes the programmer to read an integer in the next available location.

Because intermixing integers and non-integers can create some difficulty when using the **READ** , **STEP** , and **BNTP** keys, it is recommended that when assigning V/C location numbers to data values, integers and non integers should be separated, not intermixed

## 6 2 SF1 - BINARY TO BCD CONVERSION

When an output device requires data in BCD format (such as a digital display), SF1 must be used to convert the PM550 binary value to BCD format. Wiring the output device to a specific parallel output module establishes the A address. SF1 takes the value in a V, C, or A address, converts it to 4 digit BCD format then stores the result in the designated V, C, or A address.

SF1 is programmed into user memory (V/C) as follows.

KFY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	
<input type="button" value="V"/> <input type="button" value="6"/> <input type="button" value="SF"/> <input type="button" value="1"/>	V6 SF = 1	
<input type="button" value="STEP"/>	BINARY TO BCD	
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ERR OUT? YES	If no was answered the prompt "ERR OUT" would be skipped. The error out allows the specification of an image register location which is turned on if an error occurs. An error will be indicated for SF1 if the binary number is negative or is greater than 9999.
<input type="button" value="STEP"/> <input type="button" value="CLR"/> <input type="button" value="2"/> <input type="button" value="8"/>	ERR OUT CR 28	
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="1"/>	BINARY V1	
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="5"/>	BCD V5	
<input type="button" value="STEP"/> <input type="button" value="NO"/>	CHAIN SF? NO	See Section 6 16 for details on how to chain SF's
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ENTER SF? YES	The SF is not written until <input type="button" value="ENTER"/> is pressed
<input type="button" value="STEP"/>	PRESS ENTR TO ENTER	
<input type="button" value="ENTER"/>	V20	V20 is the next blank location after the SF

To read the special function key in the following

,  ,  ,  ,  Then step thru the SF using the  key The  key is operative on SF's

To modify a special function, read it and step thru until the term to be modified is displayed. Use  to clear the user entry and key in the new data. Continue to step thru until the prompt "PRESS ENTER TO ENTER" appears. Press  and the special function will be modified.

### 6.3 SF2 - BCD TO BINARY CONVERSION

When an input device provides data values in BCD format (such as thumbwheel inputs), SF2 must be used to convert the value to binary format. As discussed in Chapter 3, each Parallel Input module can interface with four separate inputs by specifying four separate A addresses. Once a device has been wired to a parallel input module its address is defined by the module's position on the mounting base.

SF2 is programmed into user memory (V6) as follows.

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	
<input type="button" value="V"/> <input type="button" value="6"/> <input type="button" value="0"/> <input type="button" value="2"/>	V6 SF = 2	
<input type="button" value="STEP"/>	BCD TO BINARY	
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ERR OUT? YES	If no was answered the next prompt ERR OUT would be skipped. The error out allows the specification of an image register location which is turned on if an error occurs. An error will be indicated if a non-BCD coded number is input.

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="STEP"/> <input type="button" value="CR"/> <input type="button" value="5"/>	ERR OUT CR5	
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="2"/>	BCD: V2	Specify BCD address
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="4"/>	BINARY: V4	Specify Binary address
<input type="button" value="STEP"/> <input type="button" value="M"/>	CHAIN SF? NO	See Section 6.16 for details on how to chain SF's.
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ENTER SF? YES	The SF is not written until enter is pressed.
<input type="button" value="SLP"/>	PRESS ENTR TO ENTER	
<input type="button" value="ENTR"/>	V10	V10 is the next location after the SF.

To read the special function, key in the following

Then step thru the SF using the  key. The  key is inoperative on SF's.

To modify a special function, read it and step thru until the term to be modified is displayed. Use  to clear the user entry and key in the new data. Continue to step thru until the prompt "PRESS ENTR TO ENTER" appears. Press  and the special function will be modified.

## 6.4 SF3 -- SCALE

Internal integer data in loops is scaled from 0 to 32000 in binary. Zero corresponds to the lower engineering units limit and 32000 corresponds to the upper engineering units limit (i.e. if a flow ranges from 2 to 30 GPM, the scaled binary value of 0 = 2 GPM, the scaled value of 16000 = 16000 / 32000 (30-2) + 2 = 16 GPM and the scaled value of 32000 = 30 GPM).

Another area <sup>where</sup> scaling is found <sup>is</sup> on the analog input and output modules. If a 0 - 20 ma or 0 - 5 volt input signal is applied to an analog input module with a 0 - 20.48 ma or 0 - 5.12 volt range, a binary output of 0 to 32000 results. Similarly, if a binary output of 0 - 32000 is applied to an analog output module with a 0 - 20.48 ma or 0 - 10.24 volt range, an output of 0 - 20 ma or 0 - 10 volts results.

In many process applications, a range of 4 - 20 ma is used instead of 0 - 20 ma. This allows open circuit detection since a minimum of 4 ma must flow at all times. A 4 - 20 ma input yields a binary output range of 6400 - 32000. By answering the 20% offset question with yes, 6400 is now the lower limit and the upper limit remains at 32000.

To program SF3 the following key sequence is required:

KEY SEQUENCE	DISPLAY	NOTE
	READY (RUN)	
<input type="button" value="V"/> <input type="button" value="7"/> <input type="button" value="x"/> <input type="button" value="3"/>	V7 SF = 3	If no was answered the next prompt "ERR OUT:" would be skipped. The error out allows the specification of an image register location which is turned on if an error occurs. An error will be indicated if the binary input is not between 0 and 32000 inclusive.
<input type="button" value="STOP"/>	SCALE	
<input type="button" value="STOP"/> <input type="button" value="YES"/>	ERR OUT: YES	
<input type="button" value="STOP"/> <input type="button" value="Y"/> <input type="button" value="7"/>	ERR OUT: Y7	

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="STEP"/> <input type="button" value="A"/> <input type="button" value="1"/> <input type="button" value="0"/> <input type="button" value="0"/>	SCALE: A100	The scale number must be an integer
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="0"/> <input type="button" value="."/>	RESULT: V0	The result must be a floating point location.
<input type="button" value="STEP"/> <input type="button" value="YES"/>	20% OFFSET? YES	
<input type="button" value="STEP"/> <input type="button" value="3"/> <input type="button" value="."/>	LOW LIM = 3.2E1	The "E" is entered using the <input type="button" value="X"/> key. See paragraph 6.1
<input type="button" value="STEP"/> <input type="button" value="2"/> <input type="button" value="."/>	HI LIM = 2.12E1	
<input type="button" value="STEP"/> <input type="button" value="NO"/>	CHAIN SF? NO	See Section 6.16 for details on how to chain SF's.
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ENTER SF? YES	The SF is not written until enter is pressed
<input type="button" value="STEP"/>	PRESS ENTR TO ENTER	
<input type="button" value="ENTER"/>	V16	Next available V location after SF

For the above example the low limit of 32 (3.2E1) corresponds to 6400 binary which is 4 ma into an analog input module. The high limit of 212 (2.12E2) corresponds to 32000 binary which is 20 ma into an analog input. Therefore, if a temperature transmitter with a range of 32°F to 212°F and an output of 4-20 ma is connected to A100, V0 would contain the temperature measured in °F.

To read the special function, key in the following:

Then step thru the SF using  key. The  key is inoperative on SF.

To modify a special function, read it and step thru until the term to be modified is displayed. Use  to clear the user entry and key in the new data. Continue to step thru until the prompt "PRESS ENTR TO ENTER" appears. Press  and the special function will be modified.

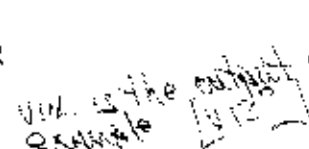


## 6.5 SF4 - UNSCALE

SF4 is basically the inverse of SF3. It is used to convert a quantity in engineering units probably from an operator interface, into a scaled binary number that the PM550 can use in loops or as an output to an analog module.

The key sequence for SF4 follows:

KEY SEQUENCE	DISPLAY	NOTE
[LTD]	READY (RUN)	
[V] [3] [SI] [4]	V3 SF - 4	
[UNSP]	UNSCALE	
[STEP] [YES]	ERR OUT? YES	If no was answered the next prompt ERR OUT? would be skipped. The error out allows the specification of an image register location which is turned on if an error occurs. An error will be indicated if the input is not between the High and Low limits inclusive.
[STEP] [4] [2] [5]	ERR OUT CR35	
[STEP] [V] [2] [0] [-]	UNSCALE V20	The number to be unscaled must be floating point.
[STEP] [V] [1] [2]	RESULT V12	The results is a scaled binary integer.
[STEP] [NO]	20% OFFSET? NO	
[STEP] [5] [0] [ ] [0]	Low Lim = 50.0	
[STEP] [2] [5] [0] [.] [0]	Hi Lim = 250.0	

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="STEP"/> <input type="button" value="NO"/>	CHAIN SF? NO	See Section 6.16 for details on how to chain SF's.
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ENTER SF? YES	The SF is not written until enter is pressed.
<input type="button" value="STEP"/>	PRESS ENTR TO ENTER	<i>Unit is the output of the SF in the example</i> 
<input type="button" value="ENTR"/>	V12	

For the above example an input of 50 would yield a scaled output of zero and an input of 250 would yield 32000. Therefore an input of 135 would yield  $\frac{135 - 50}{250 - 50} \times 32000 = 13600$ .

To read the special function, key in the following.

Then step thru the SF using the  key. The  key is inoperative on SF's

To modify a special function, read it and step thru until the term to be modified is displayed. Use  to clear the user entry and key in the new data. Continue to step thru until the prompt "PRESS ENTR TO ENTER" appears. Press  and the special function will be modified.

## 6.6 SF5 SQUARE

This function generates the square of an integer or floating point number. The input and output can be any combination of integer and floating point numbers and can be in V, C, or A. The key sequence follows:

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	
<input type="button" value="V"/> <input type="button" value="1"/> <input type="button" value="5"/>	V1 SF = 5	
<input type="button" value="STEP"/>	SQUARE	
<input type="button" value="STEP"/> <input type="button" value="YES"/>	FRR OUT? YES	If no was answered the next prompt "FRR OUT?" would be skipped. The error out allows the specification of an image register location which is turned on if an error occurs. An error will be indicated if overflow or underflow occurs.
<input type="button" value="STEP"/> <input type="button" value="CR"/> <input type="button" value="1"/> <input type="button" value="0"/> <input type="button" value="1"/>	ERR OUT: CR101	
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="2"/> <input type="button" value="0"/>	SQUARE OF V20	The input is an integer in this case and the output is a floating point number
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="2"/> <input type="button" value="1"/> <input type="button" value="."/>	EQUALS V21	
<input type="button" value="STEP"/> <input type="button" value="NO"/>	CHAIN SF? NO	See Section 6.16 for details on how to chain SF's
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ENTER SF? YES	The SF is not written until pressed.
<input type="button" value="STEP"/>	PRESS ENTER TO ENTER	
<input type="button" value="ENTER"/>	V5	V5 is the next location after the SF

To read the special function, key in the following:

**CLR** **V** **I** **GF** **READ**

Then step thru the SF using the **STEP** key. The **BSTEP** key is inoperative on SF's.

To modify a special function, read it and step thru until the term to be modified is displayed. Use **CE** to clear the user entry and key in the new data. Continue to step thru until the prompt "PRESS ENTR TO ENTER" appears. Press **ENTR** and the special function will be modified.

## 6.7 SF6 - SQUARE ROOT

SF6 takes the square root of a positive floating point or integer number. The input and output can be specified as integer or floating point independently. An attempt to take the square root of a negative number results in zero and an error flag. The entries required for SF6 are shown below:

KEY SEQUENCE	DISPLAY	NOTE
<b>CLR</b>	READY (RUN)	
<b>V</b> <b>9</b> <b>SF</b> <b>6</b>	V9 SF = 6	
<b>STEP</b>	SQUARE ROOT	
<b>STEP</b> <b>YLS</b>	ERR OUT? YES	If no was answered the next prompt "ERR OUT" would be skipped. The error out allows the specifications of an image register location which is turned on if an error occurs. An error will be indicated if the square root of a negative number is attempted or if an underflow occurs.

KEY SEQUENCE	DISPLAY	NOTE
<b>[STEP]</b> <b>[Y]</b> <b>[1]</b> <b>[1]</b>	FRR OUT: Y11	
<b>[STEP]</b> <b>[C]</b> <b>[2]</b> <b>[1]</b> <b>[ ]</b> $\{C21\}$ <i>C 2 2</i>	SQUARE ROOT OF C21	
<b>[STEP]</b> <b>[C]</b> <b>[2]</b> <b>[4]</b> <b>[ ]</b> $\{C22\}$ <i>C 2 2 4</i>	EQUALS C2223	
<b>[STEP]</b> <b>[NO]</b> $\{C22\}$	CHAIN SF? NO	See Section 6.16 for details on how to chain SF's
<b>[STEP]</b> <b>[YES]</b>	ENTER SF? <b>[YES]</b>	The SF is not written until enter is pressed
<b>[STEP]</b>	PRESS ENTR TO ENTER	
<b>[ENTER]</b>	V13	V13 is the next location after the SF

To read the special function, key in the following

**[CLR]** **[V]** **[9]** **[ ]** **[RE-D]**

Then step thru the SF using the **[STEP]** key. The **[RE-D]** key is inoperative on SF's.

To modify a special function, read it and step thru until the term to be modified is displayed. Use **[CE]** to clear the user entry and key in the new data. Continue to step thru until the prompt "PRESS ENTR TO ENTER" appears. Press **[ENTER]** and the special function will be modified.

## 6.8 SF7 COMPARE

SF7 provides the ability to compare two floating point or a floating point and an integer value. (Note. If both values to be compared are integers, ladder logic compare — paragraph 5.25.6 — can be used.) An image register location is turned on or off as the result of the compare. An example of the key sequence of SF7 in which V25 is to be compared against V30 with CR38 turned on when they are not equal is as follows:

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="ON"/>	READY (RUN)	
<input type="button" value="C"/> <input type="button" value="1"/> <input type="button" value="7"/> <input type="button" value="X"/> <input type="button" value="7"/>	CLF SF = 7	
<input type="button" value="STEP"/>	COMPARE	
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ERR OUT? YES	If no was answered the next prompt "ERR OUT?" would be skipped. The error out allows specification of an image register location which is turned on if an error occurs; an error will be indicated if any user supplied addresses are out of range.
<input type="button" value="STEP"/> <input type="button" value="X"/> <input type="button" value="3"/> <input type="button" value="9"/>	ERR OUT. CR39	
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="2"/> <input type="button" value="5"/> <input type="button" value="E"/>	COMPARE: V25.	
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="3"/> <input type="button" value="0"/>	WITH V30	
<input type="button" value="STEP"/> <input type="button" value="NO"/>	TEST FOR L.T.? V25 < V30 (less than)	
	NO	
<input type="button" value="STEP"/> <input type="button" value="NO"/>	TEST FOR L.E.? V25 ≤ V30 (less than or equal)	
	NO	
<input type="button" value="STEP"/> <input type="button" value="NO"/>	TEST FOR E.Q.? V25 = V30 (equal)	
	NO	

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="STOP"/> <input type="button" value="V"/>	TEST FOR GLE? NO	$V25 \geq V30$ (greater than or equal)
<input type="button" value="STOP"/> <input type="button" value="V"/>	TEST FOR GT? NO	$V25 > V30$ (greater than)
<input type="button" value="STOP"/> <input type="button" value="Y"/>	TEST FOR NE? YES	$V25 \neq V30$ (not equal to)
<input type="button" value="STOP"/> <input type="button" value="CR"/> <input type="button" value="3"/> <input type="button" value="8"/>	STATUS: CR38	The status location in the image register receives the result of the compare (On = True, Off = False)
<input type="button" value="STOP"/> <input type="button" value="NO"/>	CHAIN SF/ NO	See paragraph 6.16 for details on how to chain SF's.
<input type="button" value="STOP"/> <input type="button" value="Y"/>	ENTER SF/ YES	The SF is not written until enter is pressed.
<input type="button" value="STOP"/>	PRESS ENTR TO ENTER	
<input type="button" value="ENTR"/>	C22	C22 is the next location after SF

### CAUTION

Avoid the use of "EQ" (Equal to) compare unless both terms are guaranteed to be exactly equal. Small round off errors caused by floating point to integer or integer to floating point conversions can cause the "EQ" Compare to fail.

A YES answer to the appropriate question selects the type of comparison to be made. In this example, we selected to compare V25. with V30 and tested for inequality.

Here is another SF7 example where output Y12 is turned on when V45 is less than V41.

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	
<input type="button" value="C"/> <input type="button" value="1"/> <input type="button" value="7"/> <input type="button" value="SF"/> <input type="button" value="7"/>	C17 SF = 7	
<input type="button" value="STEP"/>	COMPARE	
<input type="button" value="STEP"/> <input type="button" value="NO"/>	ERR OUT? NO	The prompt asking for an error out location was skipped since no was answered.
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="4"/> <input type="button" value="5"/> <input type="button" value="V"/>	COMPARE V45	
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="4"/> <input type="button" value="1"/>	WITH: V41	
<input type="button" value="STEP"/> <input type="button" value="YES"/>	TEST FOR L.I.? YES	The other condition prompts are skipped after the first "YES" response.
<input type="button" value="STEP"/> <input type="button" value="Y"/> <input type="button" value="1"/> <input type="button" value="2"/>	STATUS Y12	
<input type="button" value="STEP"/> <input type="button" value="NO"/>	CHAIN SF? NO	See Paragraph 6.16 for details on how to chain SF's.
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ENTER SF? YES	The SF is not written until enter is pressed.
<input type="button" value="STEP"/>	PRESS ENTR TO ENTER	
<input type="button" value="ENTR"/>	C22	C22 is the next location after the SF.

To read the special function, key in the following

Then step thru the SF using the  key. The  key is inoperative on SF's.



To modify a special function, read it and step thru until the term to be modified is displayed. Use **CH** to clear the user entry and key in the new data. Continue to step thru until the prompt "PRESS ENTER TO ENTER" appears. Press **ENTER** and the special function will be modified.

## 6.9 SF8 - MATH

SF8 is used to develop math equations which include add (+) subtract (-) multiply (x) divide (/) and a result (=). These equations can include both integer and floating point variables. The equations are solved in sequence starting with the first two terms.

An example equation is

$$V1 + C33 \times C32 = V18$$

Where V1 is a scaled analog input value

C33 is a non-integer constant

C32 is an integer constant

V18 is a non-integer result

V1 will be added to C33, then that quantity will be multiplied by C32. The result will be stored in V18 as a non-integer number.

To enter that equation the following keys are pressed

KEY SEQUENCE	DISPLAY	NOTE
<b>CLR</b>	READY (RUN)	
<b>C</b> <b>10</b> <b>SE</b> <b>=</b> <b>8</b>	C10 SE = 8	
<b>MATH</b>	MATH	
<b>STOP</b> <b>YES</b>	ERROR? YES	If no was answered the next prompt ERROR? would be skipped. The error out allows the specification of an image register location which is turned on if an error occurs. An error will be indicated if overflow or underflow occurs or if a divide by zero is attempted.

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="STEP"/> <input type="button" value="Y"/> <input type="button" value="7"/>	ERR OUT. Y7	
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="1"/>	FIRST TERM: V1	
<input type="button" value="STEP"/> <input type="button" value="+"/> <input type="button" value="C"/> <input type="button" value="3"/> <input type="button" value="3"/> <input type="button" value="."/>	NEXT TERM: +C33.	
<input type="button" value="STEP"/> <input type="button" value="X"/> <input type="button" value="C"/> <input type="button" value="3"/> <input type="button" value="2"/>	NEXT TERM: XC32	
<input type="button" value="STEP"/> <input type="button" value="="/> <input type="button" value="V"/> <input type="button" value="1"/> <input type="button" value="8"/> <input type="button" value="."/>	RESUL1 =: V18.	
<input type="button" value="STEP"/> <input type="button" value="NO"/>	CHAIN SF? NO	See Section 6.16 for details on how to chain SF's.
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ENTER SF? YES	The SF is not written until enter is pressed
<input type="button" value="STEP"/>	PRESS ENTR TO ENTER	
<input type="button" value="ENTR"/>	<i>C17</i> <i>16?</i>	<i>16?</i> <i>C17</i> is the next location after the SF.

To determine the number of memory locations used count one location for SF8 header, one location for each term of the equation, and one location for a termination flag. The above example required 6 locations for its entry.

Note that the equation is entered from left to right, with each new term operating on the combination of all terms to its left. The result of the equation must always be entered last.

Addresses (V, C or A), not data values, are entered in user math equations.

There is no limit to the number of terms which can be used in SF8.

Because both integers and floating point numbers can be mixed within a single SF8 equation, you must select whether the result will be integer or floating point. If the calculated result of an equation contains a fraction, yet the defined storage location is for an integer value, the calculated result will automatically be truncated. For example:

$V1 \times V2 = V4$   
 where  $V1 = 9$   
 $V2 = 5.6$   
 then  $V4 = 50$

Because V4 is designated to be an integer, the actual result of 50.4 is truncated to 50 which is stored in V4.

This feature can be used to convert a non-integer value to an integer. If  $V1 = 39.2$ , it can be truncated to a whole number and placed in V29 by

KEY SEQUENCE	DISPLAY	NOTE
<b>CLR</b>	READY (RUN)	
<b>C</b> <b>↓</b> <b>1</b> <b>=</b> <b>8</b>	C 41 SF = 8	
<b>STOP</b>	MATH	
<b>STOP</b> <b>→</b>	ERR OUT? NO	Since no was answered, the prompt asking for an error location is skipped.
<b>STEP</b> <b>V</b> <b>1</b> <b>-</b>	FIRST TERM V1	
<b>STEP</b> <b>=</b> <b>V</b> <b>2</b> <b>9</b>	RESULT V29	
<b>STEP</b> <b>→</b>	CHAIN SF? NO	See Section 6.16 for details on how to chain SF's.
<b>YES</b> <b>→</b>	ENTER SF? YES	The SF is not written until enter is pressed.
<b>STEP</b>	PRESS ENTR TO ENTER	
<b>INTR</b>	C44	C44 is the next location after the SF.

To read the special function, key in the following:

**CLR** **C** **2** **0** **SY** **READ**. Then step thru the SF using the **STEP** key. The **RSTP** key is inoperative on SF's.

To modify a special function, read it and step thru until the term to be modified is displayed. Use **CLR** to clear the user entry and key in the new data. Continue to step thru until the prompt "PRESS ENTR TO ENTER" appears. Press **ENTR** and the special function will be modified.

#### 6 10 SF9 - SEQUENTIAL DATA TABLE

The sequential data table function has a pointer to a data table in user memory. Every time the special function is called the value at the pointer is moved to the output location and the pointer is advanced to the next location. When the pointer reaches the end of the table, it is reset to the start position and the "restart flag" is turned off. The "restart flag" is on if the pointer is at any other location other than the beginning of the table.

One excellent application is replacing a profile cam follower at the remote setpoint to a control loop. The halves corresponding to each segment of the "cam" are entered into the table. A timer activates the sequential data table function periodically which causes the simulated "cam to rotate". The output of the special function is then used as the setpoint in the loop which follows the profile programmed in the table.

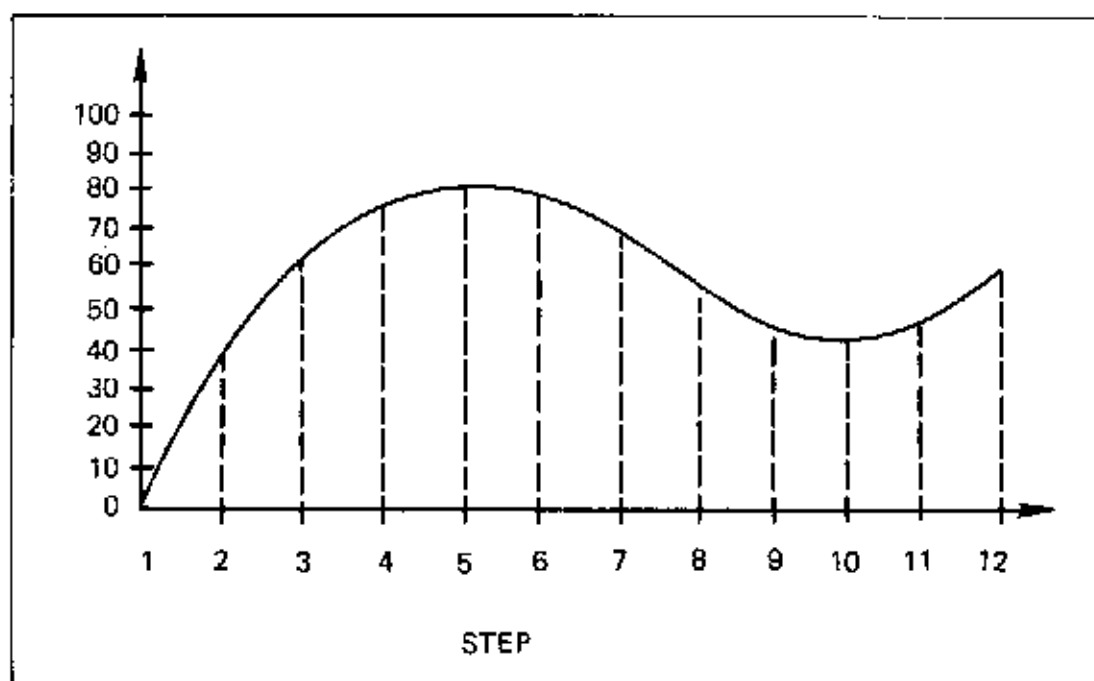






















































Figure 6 10A Cam Curve for Sequential Data Table Example

To enter SF9 follow the procedure below

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	
<input type="button" value="V"/> <input type="button" value="6"/> <input type="button" value="SF"/> <input type="button" value="9"/>	V6 SF - 9	
<input type="button" value="STEP"/>	SEQUENTIAL DATA TBL	
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ERR OUT? YES	
<input type="button" value="STEP"/> <input type="button" value="CR"/> <input type="button" value="1"/> <input type="button" value="0"/> <input type="button" value="9"/>	ERR OUT CR 109	If no was answered the next prompt "Err Out" would be skipped. The error out allows specification of an image register location which is turned on if an error occurs. An error will be indicated if the pointer goes out of line.
<input type="button" value="STEP"/> <input type="button" value="C"/> <input type="button" value="2"/> <input type="button" value="0"/>	TABLE C20	Starting address of table
<input type="button" value="STEP"/> <input type="button" value="1"/> <input type="button" value="2"/>	NO. OF ENTRIES 12	Specify number of table entries
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="2"/>	POINTER V2	Points to current table memory location. The user can modify the value in the pointer if desired.
<input type="button" value="STEP"/> <input type="button" value="A"/> <input type="button" value="1"/> <input type="button" value="0"/> <input type="button" value="1"/>	OUTPUT A101	Current value at pointer location is output to A101
<input type="button" value="STEP"/> <input type="button" value="R"/> <input type="button" value="2"/> <input type="button" value="8"/>	RLSTART FLAG CR28	CR28 off when pointer is pointing at first location in table
<input type="button" value="STEP"/> <input type="button" value="NO"/>	CHAIN SF? NO	See Section 6.16 for details on how to chain SF's
<input type="button" value="STEP"/> <input type="button" value="ENTER"/>	ENTER SF? YES	The SF is not written until enter is pressed

KEY SEQUENCE	DISPLAY	NOTE
	PRESS ENTR TO ENTER	
	V13	V13 is the next location after the SF.

To enter the data table that corresponds to the profile in Figure 6 10A use the following sequence:

KEY SEQUENCE	DISPLAY	NOTE
	READY (RUN)	
    	C20 = 0	
   	C21 = 35	
   	C22 = 55	
   	C23 = 65	
   	C24 = 72	
   	C25 = 70	
   	C26 = 63	
   	C27 = 50	
   	C28 = 40	
   	C29 = 37	
   	C30 = 38	
   	C31 = 48	

To read the special function, key in the following:

. Then step thru the SF using the  key. The  key is inoperative on SF's.

To modify a special function, read it and step thru until the term to be modified is displayed. Use  to clear the user entry and key in the new data. Continue to step thru until the prompt "PRESS ENTR TO ENTER" appears. Press  and the special function will be modified.

## 6.11 SF10 – CORRELATED DATA TABLE

The correlated data table function locates the entry in the input table that is greater than or equal to ( $\geq$ ) the input value. It then outputs the corresponding value from the output table. See Figure 6.11A for an example.

An application for the correlated data table is to look up the appropriate slope and constant to linearize a thermocouple using a segmental straight line approximation. The appropriate segment is selected by the non-linearized input.

INPUT TABLE	OUTPUT TABLE
C60 = 0	C80 = 7
C61 = 5	C81 = 15
C62 = 10	C82 = 3
C63 = 15	C83 = 27
C64 = 20	C84 = 48
C65 = 25	C85 = 23
C66 = 30	C86 = 62
C67 = 35	C87 = 98
C68 = 40	C88 = 72 V1 = 72
C69 = 45	C89 = 65
C70 = 50	C90 = 41
C71 = 55	C91 = 32
C72 = 60	C92 = 6
C73 = 65	C93 = 20
C74 = 70	C94 = 17
C75 = 75	C95 = 83
C76 = 80	C96 = 56
C77 = 85	C97 = 4
C78 = 90	C98 = 9
C79 = 95	C99 = 0
If V0 = 37 then V1 = 72 after SF10 is executed	

Figure 6.11A. Sample Correlated Data Table



To program the correlated data table, an input table and an output table similar to that generated for the sequential data table in Section 6 10 must be entered along with the following key sequence.

# KEY SEQUENCE

## DISPLAY

## NOTE

**CIR**

READY (RUN)

**C 1 6 SF 1 0**

C16 SF = 10

**STLI**

CORRELATED  
DATA TBL

**STEP YES**

ERR OUT? YES

If no was answered the next prompt "ERR OUT" would be skipped. The error out allows the specification of an image register location which is turned on if an error occurs. An error will be indicated if the input is out of range of input table.

**STEP CR 9 7**

ERR OUT: CR97

**STEP V 0**

INPUT: V0

**STEP C 6 0**

INPUT TBL: C60

It is extremely important that the input table is ordered in ascending order for this function to operate i.e., lowest value in lowest memory location; highest value in highest location, etc.

**STEP C 8 0**

OUTPUT  
TBL C80

**STEP 2 0**

NO. OF  
ENTRIES = 20

Both tables must have the same number of entries.

KEY SEQUENCE	DISPLAY	NOTE
<b>STEP</b> <b>V</b> <b>1</b>	OUTPUT: V1	
<b>STEP</b> <b>NO</b>	CHAIN SF? NO	See Section 6.16 for details on how to chain SF's
<b>STEP</b> <b>YES</b>	ENTER SF? YES	The SF is not written until enter is pressed.
<b>STEP</b>	PRESS ENTR TO ENTER	
<b>ENTR</b>	C23	C23 is the next location after the SF.

To read the special function, key in the following:

**CLR** **C** **1** **6** **SF** **READ** . Then step thru the SF using the **STEP** key. The **STEP** key is inoperative on SF's.

To modify a special function read it and step thru until the term to be modified is displayed. Use **CE** to clear the user entry and key in the new data. Continue to step thru until the prompt "PRESS ENTR TO ENTER" appears. Press **ENTR** and the special function will be modified.

## 6.12 SF11 - ASCII STRING

The PM550 has the capability of sending a pre-programmed message to a printer or computer and/or upon command from the ladder logic program. ASCII string outputs a group of character location values stored in user memory. The message can be directed to either of the two RS232 ports at 300 baud or 1200 baud.

KEY SEQUENCE	DISPLAY	NOTE
<b>CLR</b>	READY (RUN)	
<b>C</b> <b>3</b> <b>0</b> <b>SF</b> <b>1</b> <b>1</b>	C30 SF = 11	
<b>STEP</b>	OUTPUT ASCII MESSAGE	

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ERR OUT? YES	If no was answered the prompt "ERR OUT:" would be skipped. The error out allows the specification of an image register location which is turned on if an error occurs. An error will be indicated if <i>what?</i>
<input type="button" value="STEP"/> <input type="button" value="CR"/> <input type="button" value="2"/> <input type="button" value="4"/> <input type="button" value="7"/>	ERR OUT. CR247	
<input type="button" value="STEP"/> <input type="button" value="C"/> <input type="button" value="1"/> <input type="button" value="0"/> <input type="button" value="0"/>	MESSAGE START: C100	
<input type="button" value="STEP"/> <input type="button" value="1"/>	PORT NUMBER = 1	Enter RS232 port to which message is directed (1 or 2).
<input type="button" value="STEP"/> <input type="button" value="NO"/>	CHAIN SF? NO	See Section 6.16 for details on how to chain SF's
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ENTER SF? YES	The SF is not written until enter is pressed
<input type="button" value="STEP"/>	PRESS ENTR TO ENTER	
<input type="button" value="ENTER"/>	C37	C37 is the next location after SF.

Next the message to be sent must be entered in user memory. Entry is via the AUX 7 function. The key sequence required to send "Counter 1 = (Current Value)" is presented below. The message starts at C100 and the timer current value is at V18. See Section 8.7 for further details on AUX 7. Maximum length of an ASCII string is 32 characters. ASCII strings can be chained for messages greater than 32 characters.

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	
<input type="button" value="AUX"/> <input type="button" value="7"/>	AUX = 7	The message is entered using Aux 7. The message is printed using SF11.
<input type="button" value="STEP"/>	ASCII MESSAGE	
<input type="button" value="STEP"/> <input type="button" value="C"/> <input type="button" value="1"/> <input type="button" value="0"/> <input type="button" value="0"/>	START C100	Location of beginning of message.
<input type="button" value="STEP"/> <input type="button" value="6"/> <input type="button" value="7"/>	C100L = 67	See code chart in Section 8.7 for code number.
<input type="button" value="ENTR"/>	C100L = 67 (C)	Read/Write Programmer decodes and displays character if displayable. Character is written when <input type="button" value="ENTR"/> is pressed.
<input type="button" value="STEP"/> <input type="button" value="7"/> <input type="button" value="9"/>	C100R = 79	Two characters are stored in each memory location. Each character is 8 bits wide.
<input type="button" value="ENTR"/>	C100R = 79 (0)	
<input type="button" value="STEP"/> <input type="button" value="8"/> <input type="button" value="5"/>	C101L = 85	
<input type="button" value="ENTR"/>	C101L = 85 (U)	
<input type="button" value="STEP"/> <input type="button" value="7"/> <input type="button" value="8"/>	C101R = 78	
<input type="button" value="ENTR"/>	C101R = 78 (N)	
<input type="button" value="STEP"/> <input type="button" value="8"/> <input type="button" value="4"/>	C102L = 84	
<input type="button" value="ENTR"/>	C102L = 84 (T)	
<input type="button" value="STEP"/> <input type="button" value="6"/> <input type="button" value="9"/>	C102R = 69	
<input type="button" value="ENTR"/>	C102R = 69 (E)	

## KEY SEQUENCE

## DISPLAY

## NOTE

STEP 8 2

C103L = 82

ENTER

C103L = 82 (R)

STEP 3 2

C103R = 32

ENTER

C103R = 32 ( )

STEP 4 9

C104L = 49

ENTER

C104L = 49 (1)

STEP 6 1

C104R = 61

ENTER

C104R = 61 (=)

STEP V 1 8

C105L = V18

If a storage location is specified within a message, its value will be printed. The storage location takes one memory location (16 bits).

ENTER

C105L = V18

STEP 1 3

C106L = V13

ENTER

C106L = V13 ( )

STEP 1 0

C106R = 10

ENTER

C106R = 10 ( )

STEP 9 9 9

C107L = 999

Any code  $\geq 127$  will terminate AUX 7.

ENTER

END C108

STEP

READY (RUN)

To read the special function, key the following.

. Then step thru the SF using the  key. The  key is inoperative on SF's.

To modify a special function, read it and step thru until the term to be modified is displayed. Use  to clear the user entry and key in the new data. Continue to step thru until the prompt "PRESS ENTR TO ENTER" appears. Press  and the special function will be modified.

### 6.13 SF12 – SYNCHRONOUS SHIFT REGISTERS

The user can specify a table of any length to be used as a shift register. Each time SF12 is called, the data in the table will be shifted down one position. The first position in the table will be filled with zeros. The data shifted out of the last table position will be lost. The programmer must insert new data into the first table position with a MOVE instruction. Each piece of data is a full word (16 bits). Input could come from a 7MT 300 input module.

If the table is empty following a SF12 execution, then a status bit (IR bit) will be turned on indicating that there is no data in the register. If the table (register) contains any non-zero elements, then the status bit will be turned off.

SF12 can be programmed using the following sequence.

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	
<input type="button" value="V"/> <input type="button" value="3"/> <input type="button" value="SF"/> <input type="button" value="1"/> <input type="button" value="2"/>	V3 SF=12	
<input type="button" value="STEP"/>	SEQ SHIFT REG	
<input type="button" value="STEP"/> <input type="button" value="YES"/>	FRR OUT? YES	
<input type="button" value="CR"/> <input type="button" value="1"/> <input type="button" value="0"/>	ERR OUT CR10	CR10 will be on if an error is detected when executing SF12.
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="1"/> <input type="button" value="0"/>	REG START V10	Shift register start position.
<input type="button" value="STEP"/> <input type="button" value="1"/> <input type="button" value="0"/>	REG LENGTH = 10	Shift register length.
<input type="button" value="STEP"/> <input type="button" value="CR"/> <input type="button" value="2"/>	STATUS BIT CR2	CR2 on if register empty.
<input type="button" value="STEP"/> <input type="button" value="NO"/>	CHAIN SF? NO	

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ENTER SF? YES	
<input type="button" value="STEP"/>	PRESS ENTR TO ENTER	
<input type="button" value="ENTR"/>	V7	Next available data storage location

#### 6.14 SF13 – ASYNCHRONOUS SHIFT REGISTERS – INPUT

The asynchronous special function is accessed using two different special functions SF13 and SF14. SF13 is used to input to the shift register and SF14 is used to output from the register. Similar to the synchronous special function, a starting address is specified for the register along with the length of the register. In the case of the asynchronous shift register, the user is actually programming the maximum length the register will reach. If this length is exceeded, an error will occur.

The following sequence shows how to input to an asynchronous shift register.

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	
<input type="button" value="V"/> <input type="button" value="3"/> <input type="button" value="SF"/> <input type="button" value="1"/> <input type="button" value="3"/>	V3 SF=13	
<input type="button" value="STEP"/>	FAIL THRU SR INPUT	
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ERR OUT? YES	CR5 on indicates an error has occured – i.e. maximum length of SR has been exceeded.
<input type="button" value="STEP"/> <input type="button" value="CR"/> <input type="button" value="5"/>	ERR OUT: CR5	
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="1"/> <input type="button" value="0"/> <input type="button" value="0"/>	REG START V100	
<input type="button" value="STEP"/> <input type="button" value="1"/> <input type="button" value="0"/>	REG LENGTH = 10	

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="STEP"/> <input type="button" value="C R"/> <input type="button" value="9"/>	STATUS BIT CR9	CR9 is on if there is no information in the SR.
<input type="button" value="STEP"/> <input type="button" value="A"/> <input type="button" value="1"/> <input type="button" value="0"/> <input type="button" value="1"/>	SOURCE ADDR. A101	A101 is the address of the input information.
<input type="button" value="STOP"/> <input type="button" value="N O"/>	CHAIN SF? NO	
<input type="button" value="STEP"/> <input type="button" value="Y A S"/>	ENTER SF? YES	
<input type="button" value="STEP"/>	PRESS ENTR TO ENTER	
<input type="button" value="ENTR"/>	V7	

### 6.15 SF14 – ASYNCHRONOUS SHIFT REGISTERS – OUTPUT

SF14 is used to output information from an asynchronous shift register. This is done on a first in/first out basis – namely the information furthest down in the table is the first information to be output.

The following sequence demonstrates how to output information from the register.

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	
<input type="button" value="V"/> <input type="button" value="3"/> <input type="button" value="SF"/> <input type="button" value="1"/> <input type="button" value="4"/>	V3 SF = 14	
<input type="button" value="STOP"/>	FALL THRU SR OUTPUT	
<input type="button" value="STEP"/> <input type="button" value="Y L S"/>	ERR OUT? YES	
<input type="button" value="STEP"/> <input type="button" value="C R"/> <input type="button" value="5"/>	ERR OUT CR5	
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="1"/> <input type="button" value="0"/> <input type="button" value="0"/>	REG START: V100	
<input type="button" value="STEP"/> <input type="button" value="1"/> <input type="button" value="0"/>	REG LENGTH = 10	



KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="STEP"/> <input type="button" value="C R"/> <input type="button" value="9"/>	STATUS BIT. CR9	
<input type="button" value="STEP"/> <input type="button" value="A"/> <input type="button" value="5"/> <input type="button" value="0"/> <input type="button" value="1"/>	DEST REG: A501	A501 is AIM location where we will send SR output information.
<input type="button" value="STEP"/> <input type="button" value="N U"/>	CHAIN SF? NO	
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ENTER SF? YES	
<input type="button" value="STEP"/>	PRESS ENTR TO ENTER	
<input type="button" value="ENTER"/>	V7	Next available V location.

## 6.16 CHAINING SPECIAL FUNCTIONS

In many applications, more than one special functions may be required to solve a particular equation and the delay caused by sequential queuing of each function is unacceptable. To solve these applications, all special functions have been designed with a chaining capability. If Yes is answered to the prompt "CHAIN SF?", a special function immediately following (it must start in the next available location) will be executed without delay. An unlimited number of functions can be chained. An error in a special function will stop execution of the chain.

An example of an equation that requires chaining special functions is the pressure - temperature - gravity compensation of an orifice plate style gas flow meter. The equation for flow is given below

$$F = F_{MAX} \sqrt{\frac{P_0}{P}} \sqrt{\frac{G_0}{G}} \sqrt{\frac{T}{T_0}} \quad (1)$$

$$F = F_{MAX} \sqrt{\frac{P_0 G_0}{T_0}} \sqrt{\frac{T}{PG}} \quad (2)$$

Since these terms are constants let  $F_{MAX} \sqrt{P_0 G_0 / T_0} = M$ .

Then  $F = M \sqrt{\frac{T}{PG}} \quad (3)$

Equation 3 will be solved using three chained special functions. The first will employ "User Math" to solve  $1/PG$ . The second will use "Square Root" to solve for  $\sqrt{T/PG}$ . The last will multiply  $M$  BY  $\sqrt{T/PG}$  with user math.

Assume  $M$  is in location C18,  $T$  is inputted via A100,  $P$  via A101,  $G$  via A102, and store the results in V0. The following sequence will program equation 3.

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (-RUN)	
<input type="button" value="C"/> <input type="button" value="2"/> <input type="button" value="0"/> <input type="button" value="SF"/> <input type="button" value="8"/>	C20 SF = 8	
<input type="button" value="STEP"/>	MATH	
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ERR OUT? YES	
<input type="button" value="STEP"/> <input type="button" value="C"/> <input type="button" value="R"/> <input type="button" value="2"/> <input type="button" value="0"/>	ERR OUT: CR20	
<input type="button" value="STEP"/> <input type="button" value="A"/> <input type="button" value="1"/> <input type="button" value="0"/> <input type="button" value="0"/>	FIRST TERM A100	
<input type="button" value="STEP"/> <input type="button" value="+"/> <input type="button" value="A"/> <input type="button" value="1"/> <input type="button" value="0"/> <input type="button" value="1"/>	NEXT TERM + A101	
<input type="button" value="STEP"/> <input type="button" value="+"/> <input type="button" value="A"/> <input type="button" value="1"/> <input type="button" value="0"/> <input type="button" value="2"/>	NEXT TERM + A102	
<input type="button" value="STEP"/> <input type="button" value="="/> <input type="button" value="V"/> <input type="button" value="2"/> <input type="button" value=" "/>	RESULT V2	V2. is temporary storage
<input type="button" value="STEP"/> <input type="button" value="YES"/>	CHAIN SF? YES	
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ENTER SF? YES	$V2. = \frac{T}{PG}$
<input type="button" value="STEP"/> <input type="button" value="ENTER"/>	PRESS ENTR TO ENTER	
<input type="button" value="STEP"/> <input type="button" value="ENTR"/>	C26	
<input type="button" value="STEP"/> <input type="button" value="SF"/> <input type="button" value="6"/>	C26 SF = 6	
<input type="button" value="STEP"/>	SQUARE ROOT	
<input type="button" value="STEP"/> <input type="button" value="YES"/>	ERR OUT? YES	















KEY SEQUENCE	DISPLAY	NOTE
STEP L R 2 1	ERR OUT: CR21	
STEP V 2 .	SQRT OF: V2.	
STEP V 2 .	EQUALS: V2	The input and output can be the same location V2. gets replaced with its square root.
STEP YES	CHAIN SF? YES	
STEP YES	ENTER SF? YES	$V2. = \sqrt{\frac{T}{PG}}$
STEP	PRESS ENTR TO ENTR	
ENTR	C30	
S F 8	C30 SF = 8	
STEP	MATH	
STEP YES	ERR OUT? YES	
STEP C R 2 2	ERR OUT: CR22	
STEP V 2 .	FIRST TERM: V2.	
STEP X C 0 .	NEXT TERM: XC18	
STEP = V 0 .	RESULT: V0.	
STEP NO	CHAIN SF? NO	
STEP	ENTER SF? YES	$V0. = M \sqrt{\frac{T}{PG}}$
STEP	PRESS ENTR TO ENTER	
ENTR	C35	

## 6 17 SF0 – THE ENTRY POINT SPECIAL FUNCTION

In many applications, the same special function is used in many parts of the program simultaneously. To avoid having to duplicate the special function in C or V memory every time it is used, the entry point special function can be used to create an alternate entry address. This is necessary because the same SF cannot be queued more than once. SF0 avoids this problem.

Any number of entry point SF's may be placed before the desired special function and "chained" with it. Each additional simultaneous use of the function must have its own entry point SF.

To program the entry point SF, follow the steps below

KEY SEQUENCE	DISPLAY	NOTE
 	READY (RUN)	
   	C22 SF = 0	
	ENTRY POINT	
 	ERR OUT? NO	Always answer no since this function will never give an error
 	CHAIN SF? YES	Always answer yes or the function will do nothing.
	ENTER SF?	
	PRESS ENTR TO ENTER	
	C24	Continue here with the SF desired since SF0 by itself does nothing.

## CHAPTER 7

### PROGRAMMING PROCESS CONTROL EQUATIONS    LOOPS

The steps in the successful application of the PM550 control system for feedback control loops are as follows:

- 1    Develop a control configuration for the process of interest
- 2    Determine the specific parameters for each loop, such as input hardware addresses, spans of the field transmitters for each input, output hardware addresses, etc.
- 3    Develop a Loop Specification Sheet for each loop
- 4    Enter parameters into the PM550 memory using the read/write programmer

This chapter is divided into two sections.

- 1    The first section describes the basic concepts behind loop implementation in the PM550.
- 2    The second section describes the mechanisms by which the user enters loop specifications and parameters.

Coupled with the Loop Specification Sheet and the prompting of the read/write programmer, the application of the PM550 to process control is a straightforward procedure.

Examples of how the PM550 is actually applied to process control are discussed in the PM550 Loop Application Manual.

## 7.1 BASIC LOOP FEATURES OF THE PM550

The standard loop processing features of the PM550 provide for the specification of loop configurations such as the level, temperature, and flow loops of Figure 7.1A. The PM550 can support a maximum of eight such loops and also provides for such advanced control concepts as ratio control, cascade control, feedforward control, and others. Coupled with its sequence control features, the PM550 is ideally suited for control of such batch units as chemical reactors, product or raw material blenders, and others. In addition to control calculations, the PM550 also provides for alarms on the process variable and on the control deviation. In addition to the standard loop processing features preprogrammed into the CCU (PID loops), the PM550 system can solve control problems such as proportional-only control loops and time-proportioning control loops using the special function mathematics and look-up tables. The control loop capabilities, are therefore limited by the control requirements and analog input/output capacity (64 channels).

To specify the loops to the PM550, the engineer uses the read/write programmer to create the loop table. Part of this table resides in the V-area of memory, while the remainder resides in the C-area of memory. The objective of this first section is to acquaint the engineer with the basic concepts that must be appreciated in creating the loop tables.

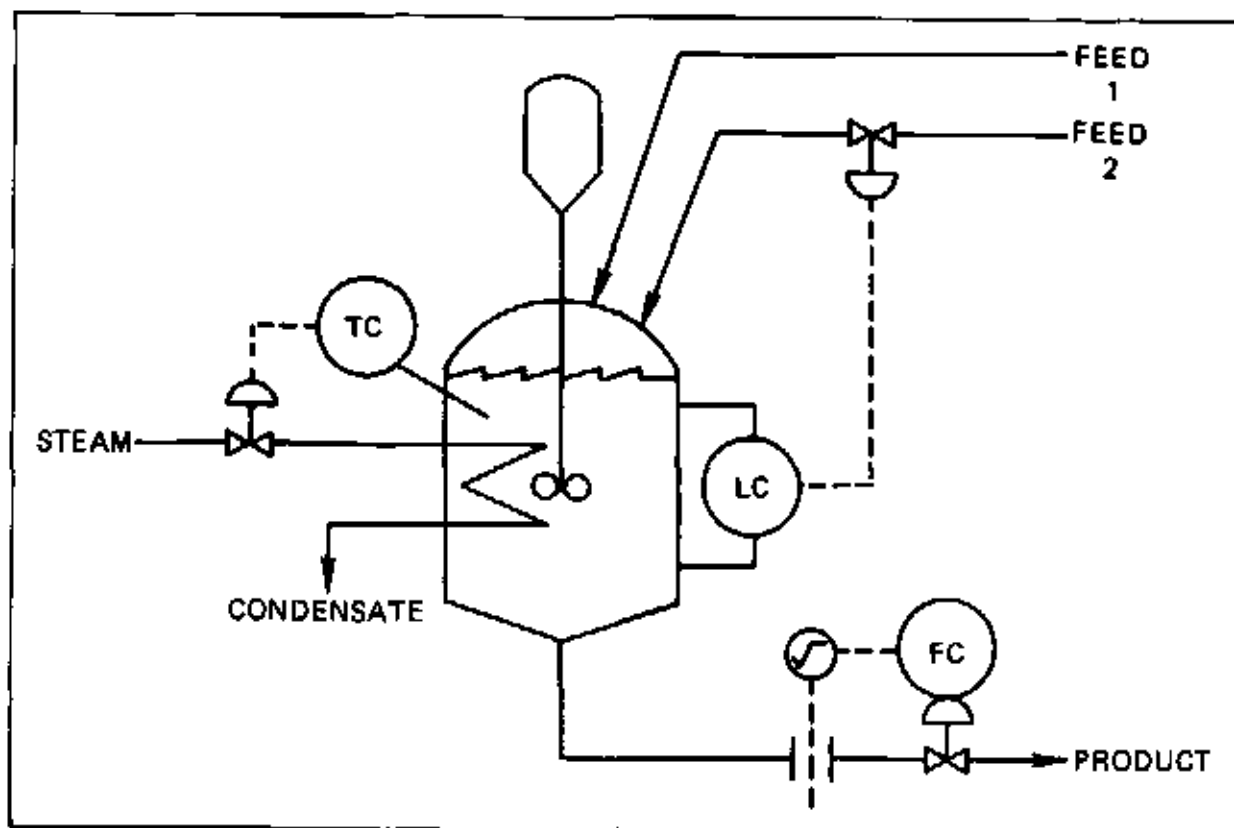
The operator interfaces with the loops via the Loop Access Module (LAM) illustrated in Figure 7.1B. For the selected loop, this station provides the following:

1. Display of the process variable.
2. Display of one other operator selected loop parameter: either the set point, deviation, integral residue (bias), or loop output.
3. Capability of changing set point or bias if loop is in automatic, or loop output if loop is in manual. If loop is in cascade, neither may be changed.
4. Display and selection of loop mode: either manual, auto or cascade.
5. Display of alarm on the process variable or warnings on the control deviation.
6. Display and enter loop tuning parameters (changing loop tuning parameters is not permitted without a key).

### 7.1.1 Process Variable

For each loop, one process variable must be specified. This process variable may be taken from either of the following:

1. From the analog input module, in which case the engineer will specify an AIM address.
2. From a location in the V-area. In this case, the process variable is frequently called a computed variable. Note: C memory locations may also be used if C is read/write memory.



**Figure 7 1A Process Unit With Three Conventional Feedback Control Loops**

An example of a computed variable is illustrated in Figure 7 1C. The heat transferred in the exchanger is the product of the fluid flow, the fluid heat capacity, and the temperature rise. The math capability of the PM550 can readily perform this computation, and the result would be stored in the V area from which it is retrieved and used as the process variable in the loop calculations. This example is given in detail in a later section.

The upper five-digit display in the operator station is dedicated at all times to the display of the process variable. This display is refreshed approximately every 200 msec. The display of the process variable is in engineering units, with the position of the decimal point specified by the engineer during loop specification.

#### 7 1 1 1. Input Span

The actual span of the PM550's analog inputs is 0 to 5.12 volts. The loop processing features provide for a linear conversion over either of the following process variable input spans:

1. A span of 0 to 5.0 volts, commonly referred to as a span of 0 to 100%.
2. A span of 1.0 to 5.0 volts, commonly referred to as a span of 20% to 100%.

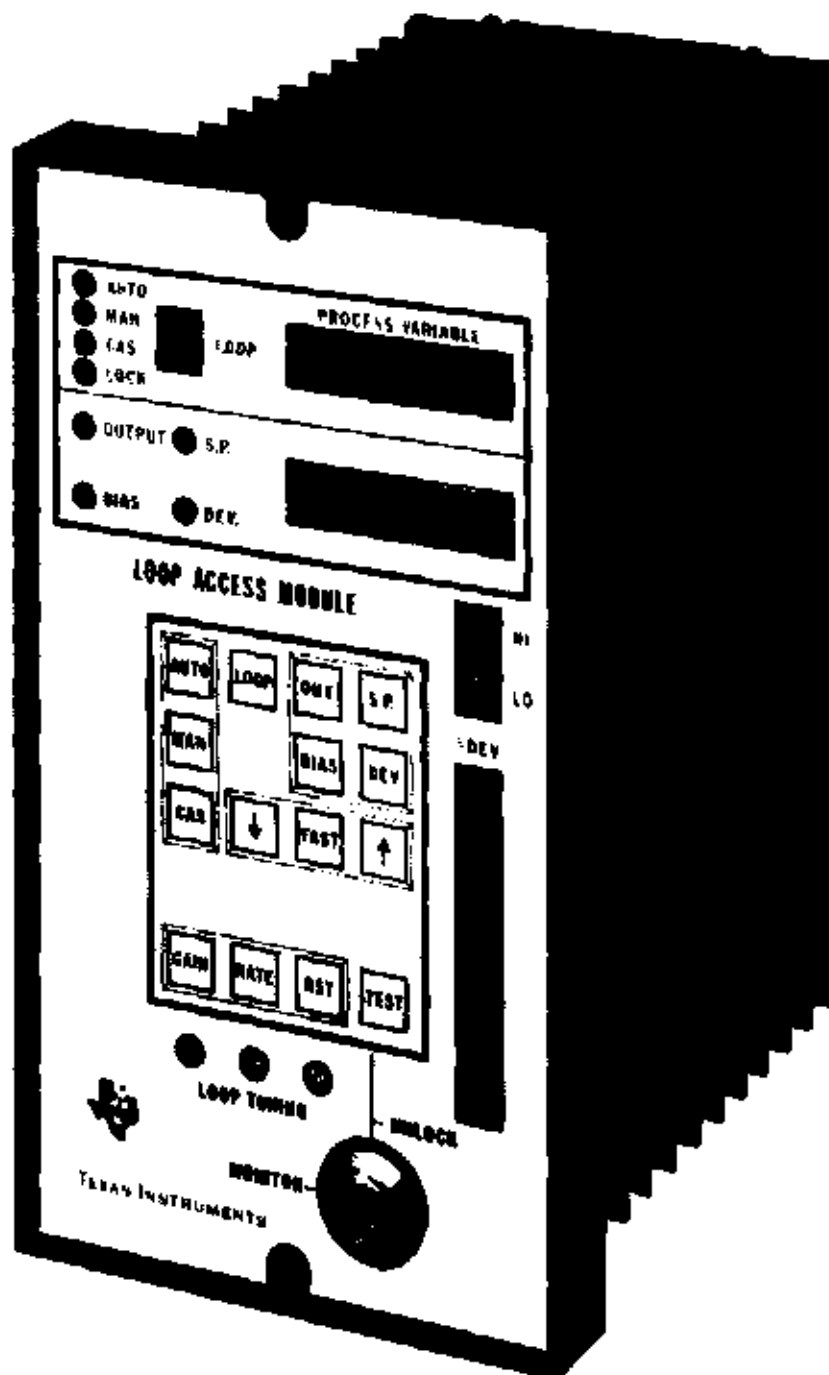


Figure 7 1B Loop Access Module (LAM)



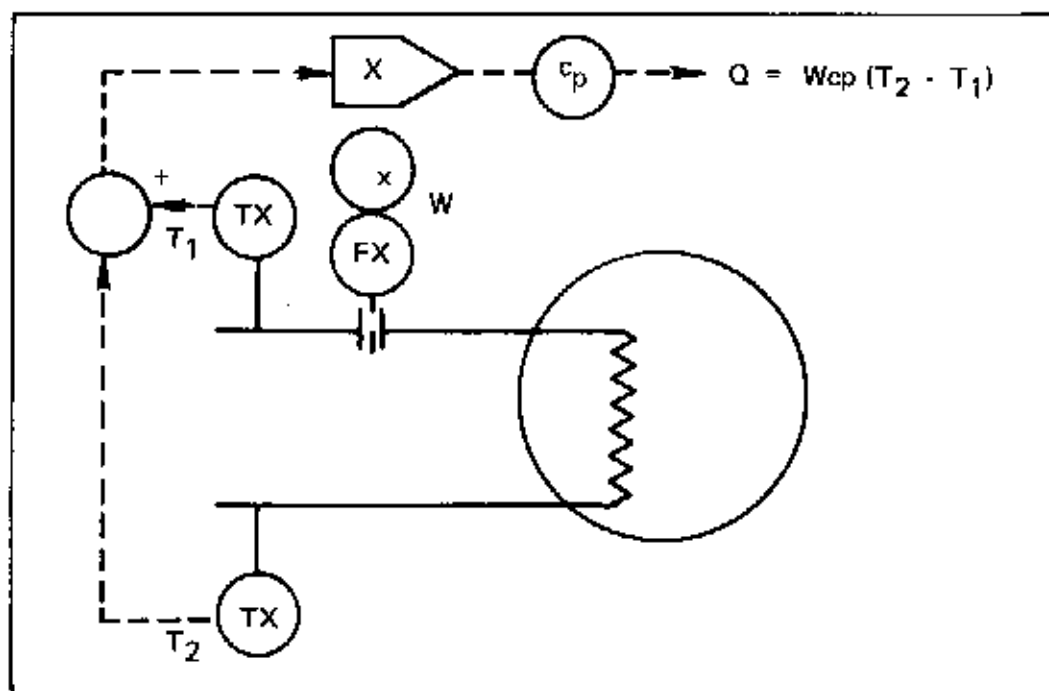


Figure 7.1C. A Computed Variable, the Heat Transfer in an Exchanger

When the loop tables are created, the engineer must specify which span is to be used.

The most common input of the latter type is the 4-20 ma current loop, where a range resistor is used to generate a 1.0 volt input differential for a current of 4 ma, and a 5.0 volt input differential for a current of 20 ma. This resistor is built into the 7MT analog input module.

In addition to specifying which of the above two spans will be used, the engineer must also specify the engineering values that correspond to the upper and lower ranges of the input span. If the span is 0 to 100%, the upper range is the engineering value corresponding to 100% and the lower range is the value corresponding to 0%. If the span is 20% to 100%, then the lower range is the engineering value corresponding to 20% (or 4 ma in the case of the current loop). Figure 7.1D illustrates the input of a temperature of 80°F when a 4-20 ma temperature transmitter is used with a span of 50 to 150°F.

The upper and lower ranges must be entered with the same precision that the process variable is to be displayed to the operator. For the example in Figure 7.1D, entries of 150.0 and 50.0 would cause the process variable to be displayed with a resolution of 0.1°F.

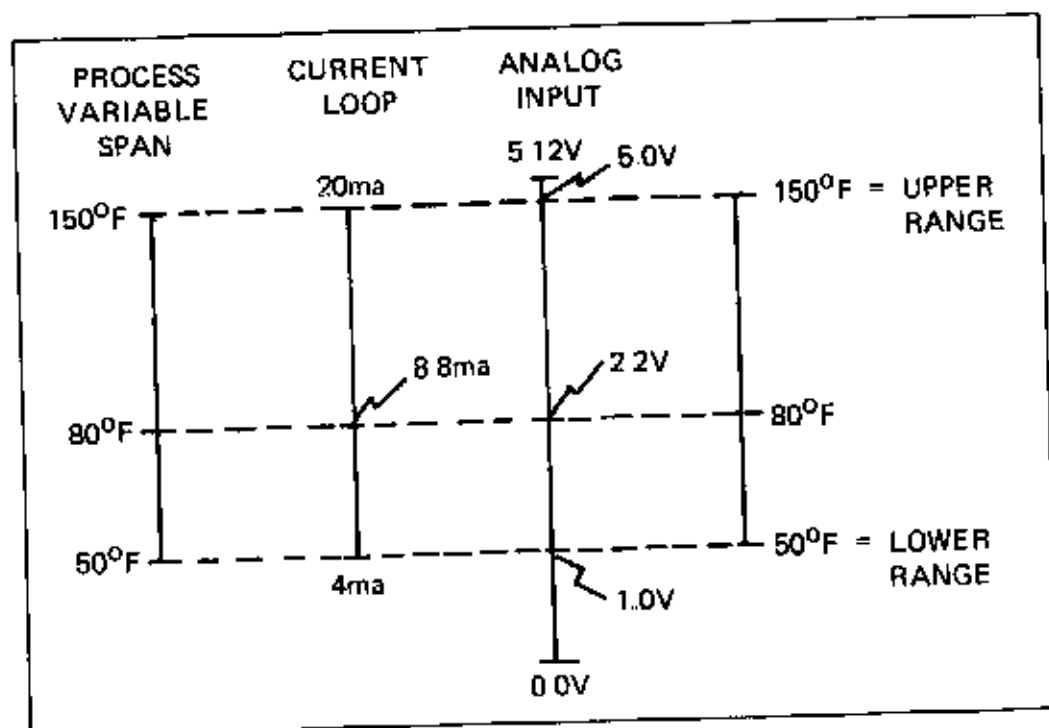


Figure 7 1D Current Loop Input on the PM550

### 7 1.1.2 Square Root

The orifice meter is a very common flow measurement device. Although variations exist for special applications, the device usually consists of a concentric orifice inserted into the pipe across which the differential pressure is measured. Thus, the actual analog input will be the output of the differential pressure transmitter. Clearly a zero differential pressure (or  $\Delta P$ ) corresponds to a zero flow. From the geometry of the pipe and orifice and from the properties of the fluid, the volumetric flow (in units such as GPM) that corresponds to the upper range (or 20 ma output from the P transmitter) can be calculated.

The relationship describing the orifice meter is

$$F = C_v \sqrt{\frac{\Delta P}{\rho}}$$

which involves a square root relationship. The lower range on the  $\Delta P$  transmitter is always zero. Let the upper range be  $\Delta P_{\max}$ . Then the above equation can be written as follows

$$F = \left( C_v \sqrt{\frac{\Delta P_{\max}}{\rho}} \right) \left( \frac{\sqrt{\Delta P}}{\sqrt{\Delta P_{\max}}} \right)$$

Note that  $C_v \sqrt{\Delta P_{\max}/\rho}$  must be the volumetric flow, say  $F_{\max}$ , that corresponds to a full scale  $\Delta P$  reading of  $\Delta P_{\max}$ . Thus, the equation becomes

$$F = F_{\max} \sqrt{\frac{\Delta P}{\Delta P_{\max}}}$$

Further, observe that  $\Delta P/\Delta P_{\max}$  must always be a number between 0.0 and 1.0. Therefore, we might consider  $\Delta P/\Delta P_{\max}$  to be a normalized  $\Delta P$  which we could represent as  $\Delta P'$ . The flow equation becomes

$$F = F_{\max} \sqrt{\Delta P'}$$

Normally the  $\Delta P$  transmitter will output a 4-20 ma signal. The PM550 can calculate  $\Delta P'$ , the normalized  $\Delta P$ , directly from the input voltage, namely,

$$\Delta P' = \frac{V}{5.0} - \frac{1.0}{1.0}$$

The span of the  $\Delta P$  transmitter is not needed for this calculation.

To calculate the flow, the PM550 needs only to take the square root of  $\Delta P'$  and then multiply by  $F_{\max}$ . Thus, in creating the loop table, the engineer must specify the square root option on the process variable, and provide the span range as 0.0 for lower range and  $F_{\max}$  for upper range. Specifically, note that the span of the differential pressure transmitter (in  $H_2O$ , psi, mm Hg, or other units) is never entered. The display to the operator will always be the flow in the units of  $F_{\max}$ , and the set point will also be specified in these units.

### 7.1.1.3 Special Functions on the Process Variable

The PM550 special functions can be used to perform calculations on the process variable. If necessary, these calculations can also involve other analog inputs and values stored in the V and C areas. The value of the process variable is always made available for these calculations. If the square root option is specified, the square root of the process variable is computed first and the result passed to the special function calculations.

One example of a special function calculation would be smoothing (or filtering) the process variable. The basic smoothing or filtering equation is the following differential equation

$$\tau \frac{dPV}{dt} + \overline{PV} = PV$$

where PV = raw input,  $\overline{PV}$  = smoothed value, and  $\tau$  = filter time constant. The corresponding difference equation is

$$\overline{PV} = kPV + (1-k)\overline{PV}$$

where

$$k = \exp(-T_s/\tau) \approx \frac{1}{1 + \frac{T_s}{\tau}}$$

$T_s$  = loop sampling time

This calculation requires one floating point location in the V-area for storing the value of  $\overline{PV}$ .

Another example where the special function feature would be necessary is in the pressure/temperature/gravity compensation of gas flows. In order to compute  $F_{max}$  for the flow equation, the gas temperature is assumed to be  $T_0$  (absolute), the gas pressure is assumed to be  $P_0$  (absolute), and the gravity is assumed to be  $G_0$ . The actual flow equation then becomes

$$F = F_{max} \sqrt{\frac{\Delta P}{P}} \sqrt{\frac{T}{T_0}} \sqrt{\frac{P_0}{P}} \sqrt{\frac{G_0}{G}}$$

The meter factor is defined to be

$$M = \sqrt{\frac{T}{T_0}} \sqrt{\frac{P_0}{P}} \sqrt{\frac{G_0}{G}} = \sqrt{\frac{P_0 G_0}{T_0}} \sqrt{\frac{T}{PG}}$$

where  $\sqrt{P_0 G_0 / T_0}$  is a pre-computed constant stored in either the V or C area. The special function calculations basically compute the meter factor M and multiply by the value supplied for the process variable to obtain the true flow. To provide protection against transmitter failures, upper and lower bounds can be placed on the meter factor.

The special function feature can be utilized in a variety of ways, including the computation of the heat transferred as illustrated in Figure 7.1C. This feature makes the PM550 capable of implementing truly powerful control systems.

### 7.1.2 Manipulated Variables

The results of all control calculations is a number between 0 and 1.0, which is always displayed to the operator in percent to a resolution of 0.1% (that is, the output range is 0.0% to 100.0% for the operator station). The output may be directed to either of two destinations:

1. As an analog output from AIM.
2. To an internal memory location in the V-area for cascading loops. Note: C area of memory can be used if it is maintained in Read/Write.

When the output is an analog signal to a TMT output module, a 0% output will produce a zero output voltage or current. A 100% output produces a 10 volt or 20 ma signal. If a 20% offset output is selected, 0% output produces a 1 volt or 4 ma signal, and 100% output produces a 10 volt or 20 ma signal.

### 7.1.3 Control Calculations

The basic purpose of the loop features in the PM550 is to enable the engineer to easily and quickly implement feedback control on his process. The basic control equation is the commonly used proportional-integral-derivation (PID) control law which has been used in conventional controllers for many years. In addition to the basic PID control equation, the PM550 provides the engineer with error squared and error deadband options and incorporates an anti windup algorithm or reset. The PM550 implementation of PID control inherently provides for bumpless transfer. In addition, cascade control configurations are easily implemented, again incorporating bumpless transfer.

The sections that follow describe the features available for implementing control loops

#### 7.1.3.1 The PID Control Equation

Although considerable effort has been directed toward superior control equations, the conventional PID equation continues to be the most popular. The basic form of the equation is as follows.

$$M - M_0 = K_c \left( e + \frac{1}{T_i} \int e dt + T_d \frac{de}{dt} \right)$$

where

- M = control output
- e = error signal = R - PV
- R = set point
- K<sub>c</sub> = proportional gain
- T<sub>i</sub> = reset time
- T<sub>d</sub> = derivative time
- M<sub>0</sub> = initial value of M

The coefficients K<sub>c</sub>, T<sub>i</sub>, and T<sub>d</sub> are called tuning constants. The value of M<sub>0</sub> is selected to provide for bumpless transfer.

A common modification to the above equation is to base the derivative on the feedback variable instead of the error signal. The result is:

$$M - M_O = K_c \left( e + \frac{1}{T_i} \int e dt - T_d \frac{dPV}{dt} \right)$$

One advantage of this is that set point changes no longer affect the derivative mode.

In the PM550, the following discrete version of the above equation is used:

$$M_n - M_O = K_c \left[ e_n + \frac{T_s}{T_i} \sum_{i=0}^n e_i - \frac{T_d}{T_s} (PV_n - PV_{n-1}) \right]$$

where

- $M_n$  = output at time  $t = nT_s$
- $e_n$  = error at time  $t = nT_s$
- $PV_n$  = process variable at time  $t = nT_s$
- $T_s$  = sampling time

The equation is commonly referred to as the position form of the PID control equation

In the PM550 implementation, certain modifications to the above equation have been made to provide for computational efficiency. In addition, protection against reset wind-up has been incorporated. Executing the above equation requires two memory locations, one for storing the integral sum and one for storing the previous process variable ( $PV_{n-1}$ ). These have been incorporated into the loop tables, so that the user need not be concerned with them.

The sampling time is specified for each loop on an individual basis. The basic loop processing interval for the PM550 is 0.5 second. The sampling time must be specified as an integer multiple of 0.5 seconds up to a maximum of 4095.5 seconds.

### 7.1.3.2 Auto-Manual Modes

The basic PID control loop can be in one of two modes:

- 1 Auto
- 2 Manual

A third mode, cascade, is concerned only with the source of the set point, but this will not be discussed in Section 7.1.3.7. The location of mode keys and lights on the LAM is illustrated in Figure 7.1B.

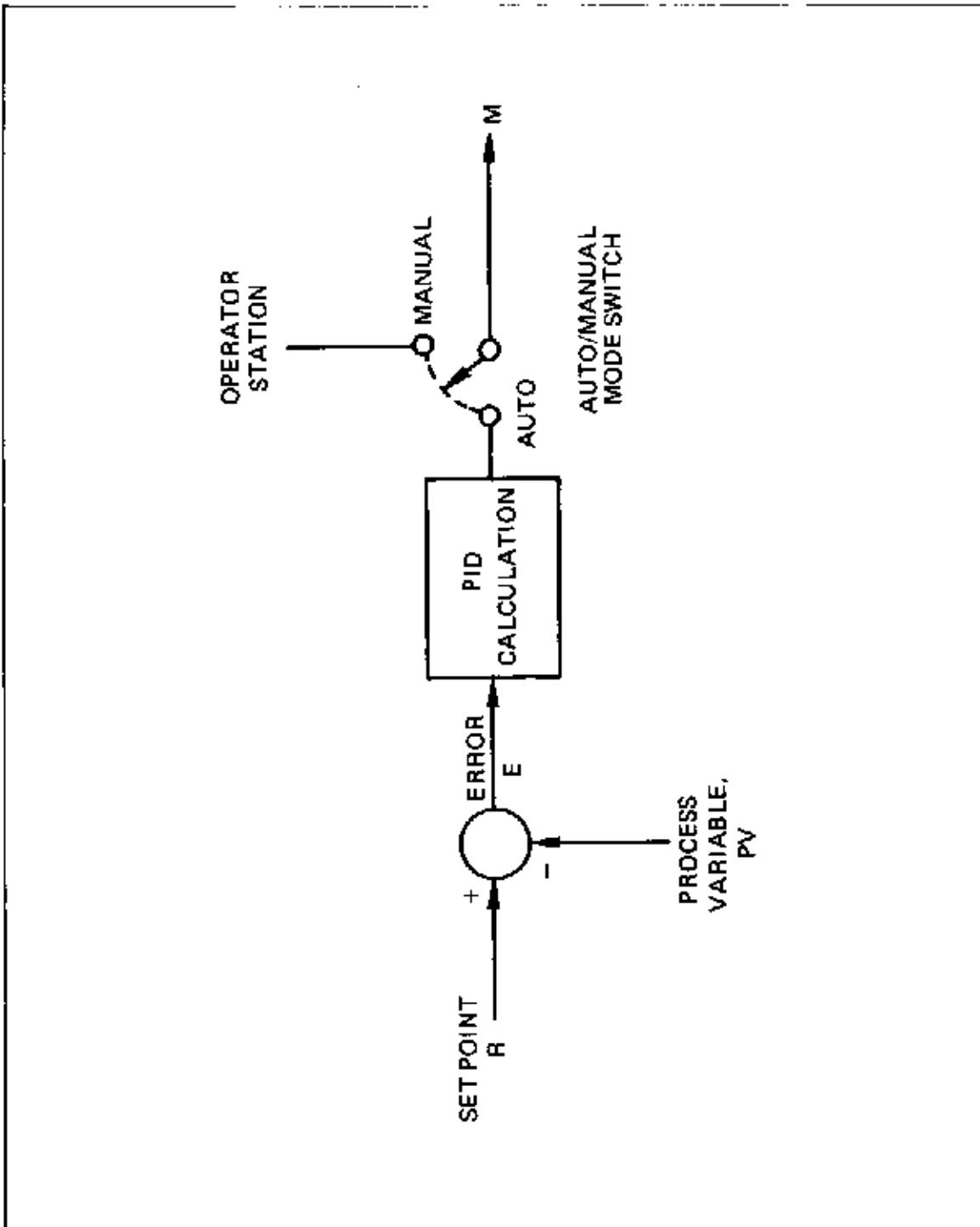


Figure 7.1E. Role of the Auto/Manual Mode Selection

Basically, no loop control calculations are performed in the manual mode. When the loop is in manual, the operator may specify the loop output through the LAM. Essentially, the auto manual mode selection acts like the switch in Figure 7.1E. When the switch is in the manual mode, the output is as specified through the LAM. When the switch is in the auto mode, the output is the result of the PID calculations. To conserve processor time, the PM550 executes the PID calculations only when the loop is in Auto.

When the loop is in Auto, the operator may select any of the following for display in the lower five-digit display on operator station:

1. Set point
2. Error (or deviation,  $\Delta$ )
3. Integral sum (or Bias)
4. Output
5. Loop tuning parameters

In the auto mode, the operation may only change the set point and bias. Although he can display the output to the process, he cannot change its value. The set point and error are displayed in the same engineering units as the process variable. The output and bias (integral sum) are both displayed in %.

In order to achieve bumpless transfer, it is necessary that the output not change when the loop is switched from manual to auto or vice versa. For the switch from auto to manual, the PM550 simply retains the last value computed by the PID equation until the operator specifies a new value. For the switch from manual to auto, the set point is set equal to the process variable and the value of  $M_0$  in the PID equation is set equal to the present manual output to the process. In the PM550, this latter step is accomplished by providing the proper initial condition to the bias.

As will be described in a subsequent section, the auto/manual mode status is available to the ladder logic portion of the PM550 via the selected bits in the image register. The CCU may either read or set the image register bit for the auto/manual mode.

### 7.1.3.3 Tuning Constants

As indicated in the previous section, the PID control equation involves three coefficients commonly referred to as tuning constants. The units and ranges on these three coefficients are as follows.

Coefficient	Units	Range
Proportional gain, $K_c$	%/%	0.00 - 99.99
Reset time, $T_i$	min	0.01 - 999.99
Derivative time, $T_d$	min	0.00 - 999.99



Since the reset time coefficient is in the denominator in the control equation, a zero value is not allowed. Furthermore, if the reset time is specified as 999.99, the coefficient  $1/T_i$  in the control equation is set equal to zero, or in other words, the control equation has no reset action. The special case of 0.00 for the proportional gain  $K_c$  will be discussed in the next section (the result is integral-only control).

In some conventional controllers, the proportional mode setting is the proportional band in %. To convert the proportional band (PB) in % to the proportional gain in %/% (actually % change in output/% change in input), the equation is

$$K_c = \frac{100}{PB}$$

In other words, simply change the PB from % to a number and then take the reciprocal.

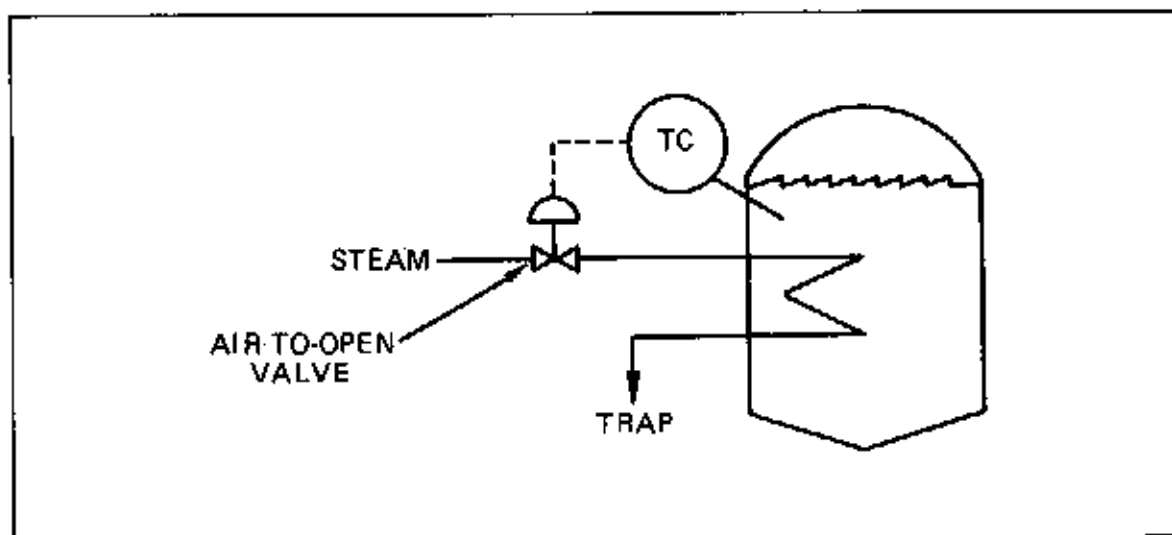
The proportional gain is always entered as a positive number. This is reasonable when a positive change in the output from the PM550 will result in a positive change in the loop's process variable. Figure 7.1F illustrates such a process. If the signal to the steam valve increases, the result would be an increase in the process temperature. Thus, the gain should be positive, or as is commonly said, the controller should be direct acting.

Figure 7.1G illustrates a case where a positive change in the PM550 output would result in a decrease in the process variable. If the signal to the valve increases, more cooling water is admitted, which would lower the process temperature. For this case, the controller's proportional gain should be negative, or as is commonly said, the controller should be reverse acting.

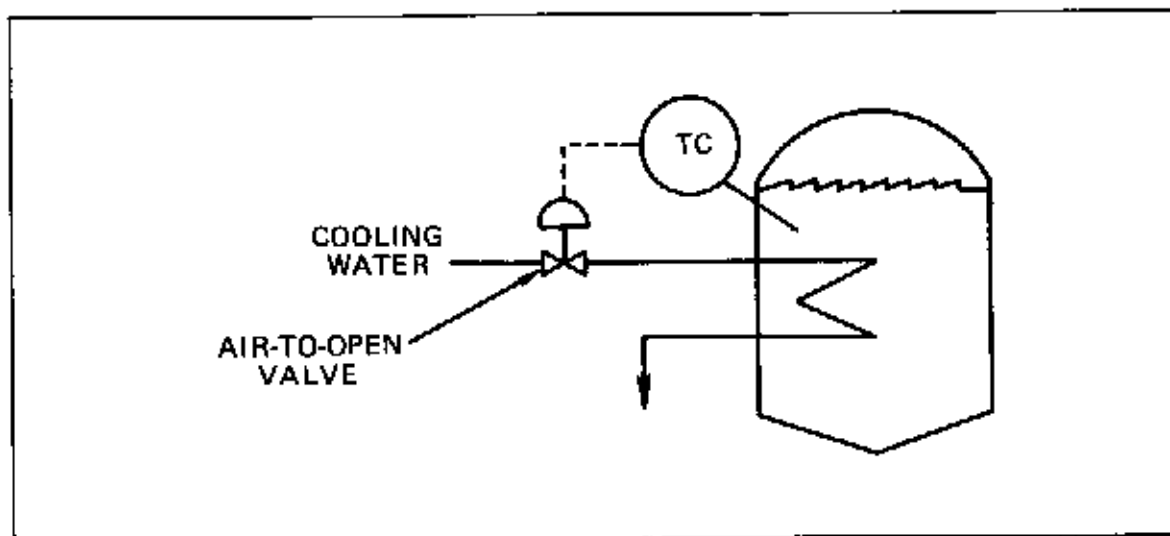
In specifying the loop table for the PM550, the proportional gain is always entered as a positive number. A separate entry to the read/write programmer specifies the controller to be direct or reverse acting, which is used internally to provide the proper sign for the controller gain.

#### 7.1.3.4 Integral-Only Control

The control equations as presented in Section 7.1.3.1 all had the proportional gain  $K_c$  multiplying all three modes. The PID equation was written this way to be consistent with the customary presentation of the equation, but the PM550 provides for a case where the proportional gain  $K_c$  is specified to be zero.



**Figure 7.1F. Example Requiring a Direct Acting Controller**



**Figure 7.1G. Example Requiring a Reverse Acting Controller**

In effect, a zero value for  $K_C$  eliminates the proportional and derivative modes but not the reset mode, the result being

$$M - M_0 = \frac{K_C}{T_i} \int e dt = K_R \int e dt$$

This is a reset only controller, which has some practical applications (one being in noisy flow loops). In this case, the integral mode gain  $K_R$  is taken to be  $1/T_i$ , or in other words, the gain  $K_i$  is taken to be 1.0 for the reset mode but zero for all other modes.

Actually, the integral-only option permits a loop to be used as an integrator instead of a controller. To do this, we must specify the set point to be zero and the controller to be reverse acting. The equation then becomes

$$M - M_0 = \frac{1}{T_i} \int (0 - PV) dt = -\frac{1}{T_i} \int PV dt$$

One application of this is as a flow totalizer. Usually  $M_0$  will be specified as zero (that is, the output will be specified to be zero before the "controller" is placed in auto to begin the integration). However, the reset time  $T_i$  must be set so as to keep  $M$  from reaching 1.0.

For example, suppose the span on the process variable is 0 to 20 GPM, and the largest flow total (output of the integral) will be 480 gallons. It would therefore take

$$\frac{480 \text{ Gal}}{20 \text{ GPM}} = 24 \text{ Minutes}$$

to span of the PV. To prevent  $M$  from exceeding 1.0, the value of  $T_i$  must be 24 or greater. Therefore, we might use 24 for  $T_i$  if we were highly certain that 480 gallons would be the maximum flow total. If we were not so sure about the 480 gallons being the maximum, we might set  $T_i$  at 50. This would yield 1000 gal. corresponding to an output of 1.0.

### 7.1.3.5 Error-Squared Control

In applications such as pH control and, occasionally, level control, the standard PID control equation is not very satisfactory. In these applications, the controller should have a low gain when the error (or deviation from set point) is small, but a much larger gain when the error is large.

One approach to obtaining these characteristics is to multiply the gain by the error, giving:

$$M - M_0 = K_C |e| \left( e + \frac{1}{T_i} \int e dt + T_d \frac{dPV}{dt} \right)$$

In effect, the control equation is now based on  $e|e|$  (or signed error squared) instead of the error alone.

This option is achieved simply by replying "yes" to a question "ERROR SQUARED?" from the read/write programmer

#### 7.1.3.6 Error Deadband

The previous section indicated that some applications required a controller with a low gain for small errors and a high gain for large errors. Another approach to implementing such a characteristic is to incorporate an error deadband into the controller.

In the PM550, the error deadband option causes the error to be "recomputed" as follows

$$\text{If } (|e| - e_{db}) < 0 \quad \text{then } e^* = 0$$

$$\text{If } (|e| - e_{db}) > 0 \quad \text{then } e^* = (|e| - e_{db}) \text{sgn}(e)$$

where:

$e_{db}$  = error deadband

$\text{sgn}(e)$  = sign of  $e$  (1 for  $e > 0$ , -1 for  $e < 0$ )

$e^*$  = error used in control calculations

Figure 7.1H illustrates the relationship of  $e^*$  vs.  $e$ .

This option is attained by simply responding "yes" to the read/write programmer question — "DEADBAND?" However, both error squared and error deadband may not both be used simultaneously for the same loop. The value used for  $e_{db}$  is that for the yellow deviation alarm as described in Section 7.1.4.2.

#### 7.1.3.7 Remote Set Point and Cascade Mode

The previous sections described procedures where the set point of a loop is entered via the operator station. The PM550 also permits the set point to be taken from an internal memory location in either the V-area or the C-area. This feature's counterpart in a conventional controller is the remote set point option, which must be purchased whenever the set point is provided by another device.

For example, suppose it is necessary to adjust the temperature according to a pre-defined manner while a batch reaction is proceeding. Using conventional equipment, a cam programmer could be used to drive the set point to a conventional controller. To accept the signal from the cam programmer, the conventional controller must have the remote set point option.

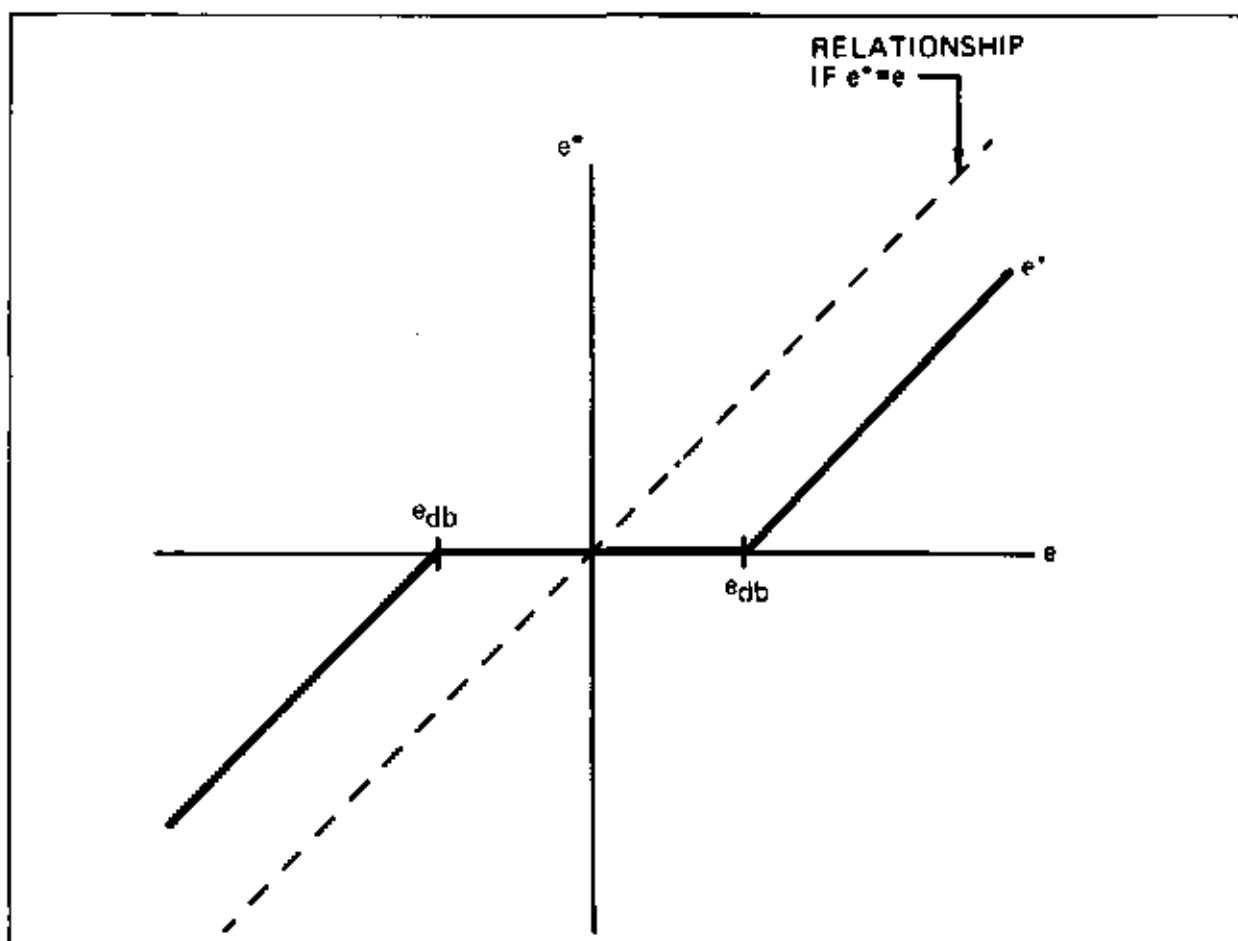


Figure 7.11 Error Deadband Relationship

Using the PM550, this capability can be obtained without the purchase of extra hardware. Using the look-up table special function, the proper temperature at any time can be obtained from the table and stored in a location in the V area. As will be described below, the loop can be instructed to retrieve its set point from this location.

Figure 7.11 expands upon the diagram in Figure 7.1E to incorporate the remote set point capability. An additional switch known as the cascade switch is provided on the set point input. When the switch is in the open cascade position, the set point is accepted only from the operator station (LAMP). When the switch is in the closed cascade position, the set point is taken from a location in the V or C area, and will not be accepted from the operator station.

On the operator station, there are no keys labeled "open cascade" and "closed cascade", but only a single key labeled "CASC ADD". This key is actually the closed cascade position as described in Figure 7.11. This position only makes sense if the loop is also in AUTO. Thus, there are only three reasonable combinations of the positions of the two switches in Figure 7.11.

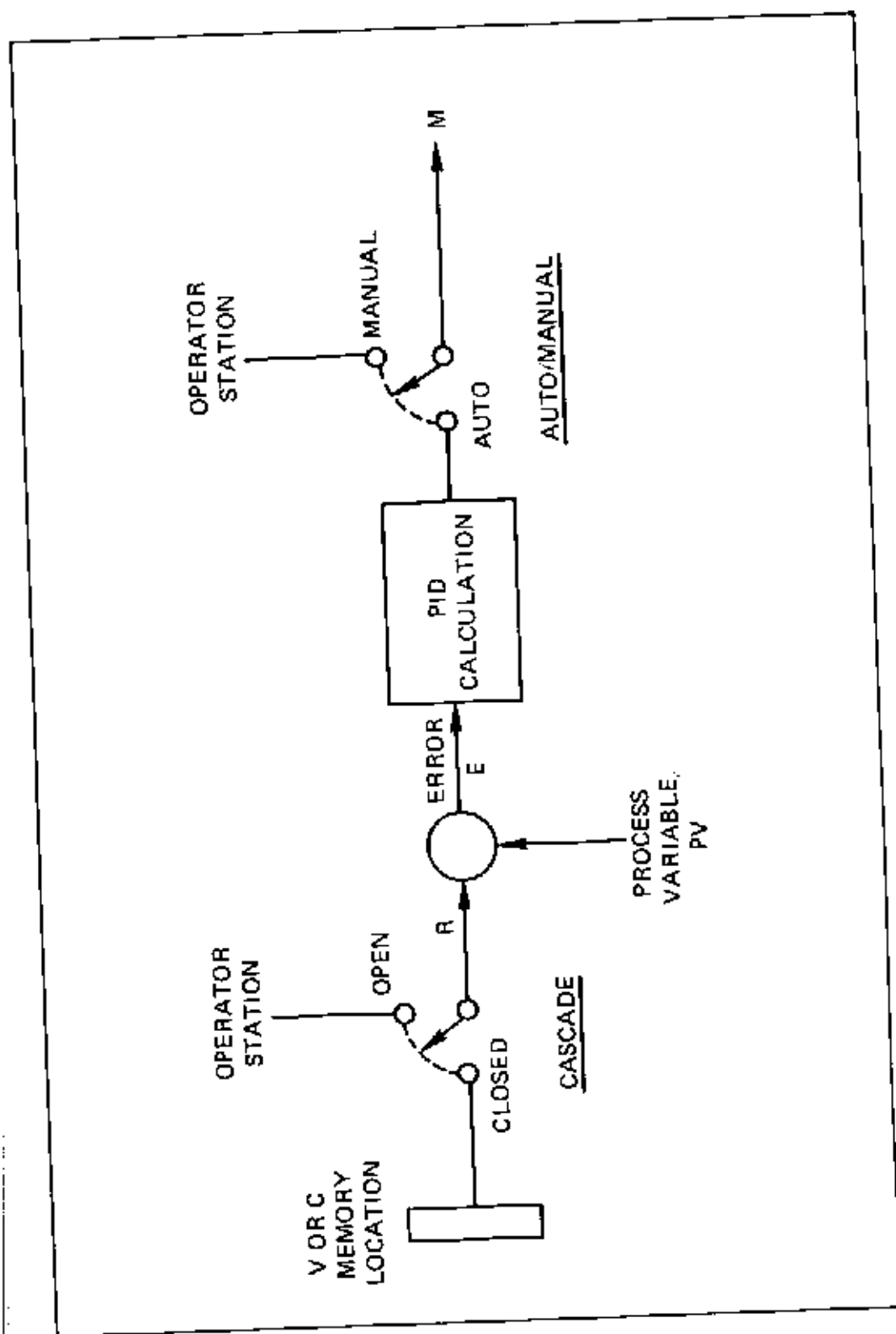


Figure 7.1J Cascade Switch

- 1 Auto/Manual switch in Manual. Actually, the position of the cascade switch is immaterial, but the PM550 will automatically force the switch to open cascade. This position is attained at any time by pressing the MANUAL key.
- 2 Auto/Manual switch in Auto, Cascade switch in Open Cascade. This position is attained at any time by pressing the AUTO key. By inference, the AUTO key really means "Auto Open Cascade".
- 3 Auto/Manual switch in Auto, Cascade switch in Closed Cascade. This position is attained by pressing the CASCADE key. If the loop is in manual, depressing the CASCADE will place the loop into AUTO and then into CASCADE. Note that the CASCADE key really means "AUTO Closed Cascade".

The position "Manual Closed Cascade" is meaningless to the PM550, and it will never occur. If the loop is in CASCADE (actually Auto, Closed Cascade), depressing the manual key switches the loop to "Manual Open Cascade".

The PM550 also provides for bumpless transfer to closed cascade provided the source of the set point is a read/write location in user memory. Whenever the CASCADE key is depressed, the PM550 first takes the current value of the set point and stores it in the user area location from which the set point value is to be taken. Of course, the value in this location can be changed at any time. Conversely, when the transition is made from CASCADE to AUTO, the last value of the set point is retained until a new value is entered through the operator station.

In making the loop configuration, the engineer may choose not to specify a memory location for the set point by answering "NO" to "Remote Setpoint". In this case, the PM550 will not permit the loop to be put into CASCADE.

As for the Auto/Manual switch, the status of the cascade switch is available to the ladder logic through the CR-bits in the image register. The CCU may read the status of the cascade switch or may specify its status. Note that if a LAM is not used and the setpoint for a loop is to be obtained from a V, C, or A memory location, the prompt "LOOP FLAGS?" must be answered YES and the loop flags in the image register requesting Auto and Closed Cascade must be set to 1. (See table 7.2K).

### 7.1.3.8 Cascade Control

Some process applications require that the output of one controller be used as the set point for another. Figure 7.1K illustrates a case where a temperature controller is providing the set point to a flow controller. The temperature controller is said to be the outer or master controller, the flow controller is said to be the inner or slave controller. The PM550 accommodates configurations such as this, and even permits higher levels of cascading.

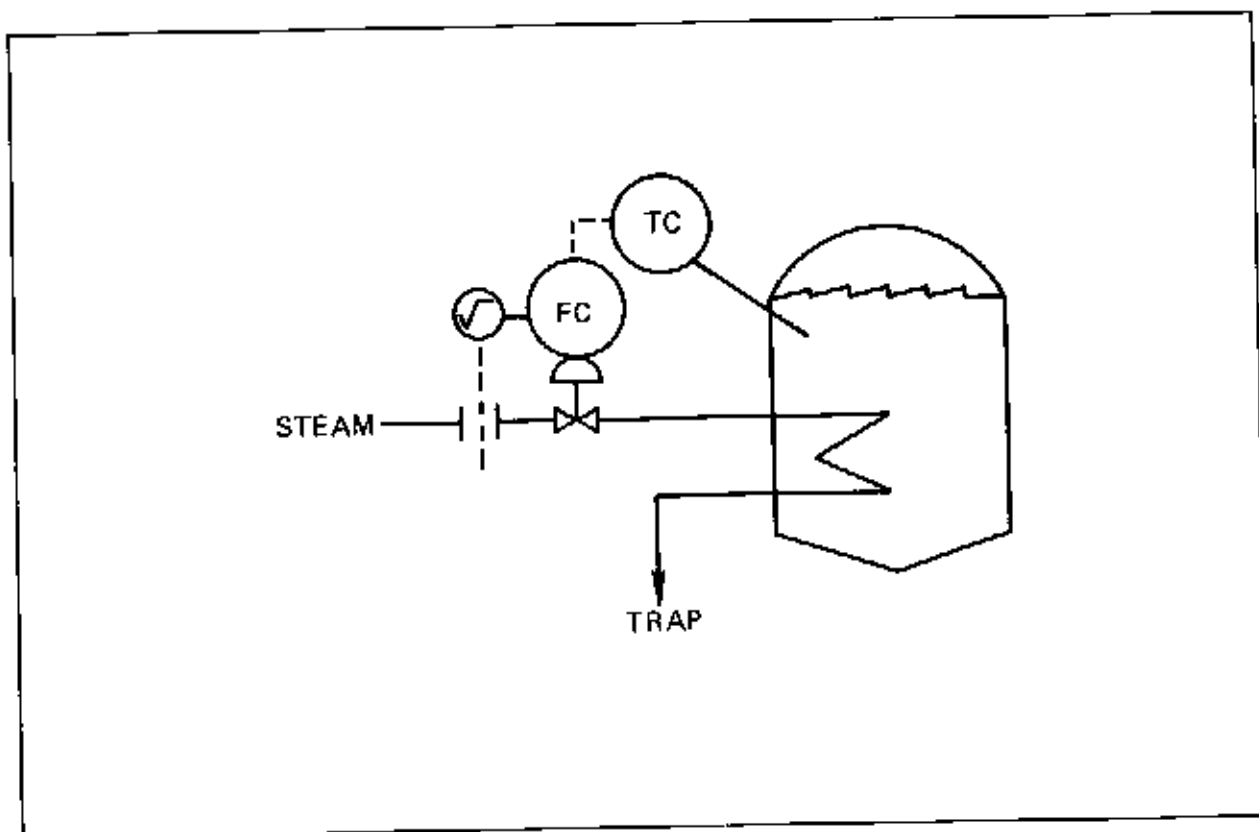


Figure 7.1K. Cascade Control System

Establishing such a configuration in the PM550 is easily accomplished, requiring only two specifications

1. For the inner controller (the flow controller in Figure 7.1K), a source for the set point must be specified as a location in the V-area
2. For the outer controller (the temperature controller in Figure 7.1K), the destination of the output must be the same location in the V-area.

In all other respects, these two controllers are specified as usual.

The loop configuration in the PM550 is illustrated in Figure 7.1L. For the outer control loop, it is not necessary that a source be provided for its set point, in which case the PM550 would not permit closed cascade. Of course, a source for this set point can be provided if appropriate

The PM550 provides for bumpless transfer throughout the possible number of states of the cascade configuration in Figure 7.1L. Starting with both loops in manual, the transitions would be as follows:



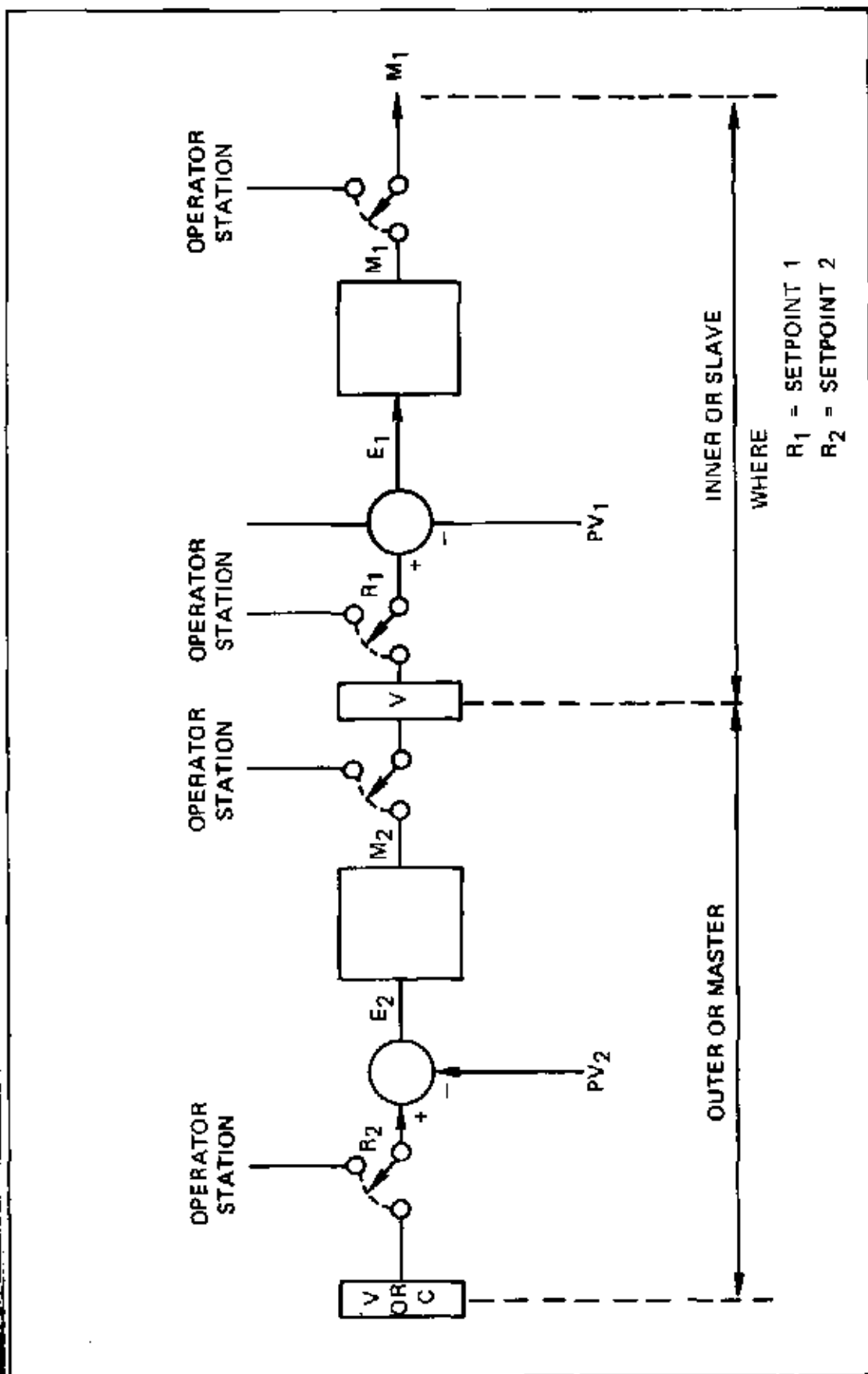


Figure 7 1 L Cascade Control Configuration in the PM550

1. Place inner loop in AUTO. The PM550 automatically sets  $R_1 = PV_1$  and appropriately initializes the integral sum (bias) for the inner loop.
2. Place the inner loop in CASCADE. The PM550 places the value of  $R_1$  in the V-area memory location.
3. Place the outer loop in AUTO. The PM550 automatically sets  $R_2 = PV_2$  and appropriately initializes the integral sum (bias).

If an internal source for  $R_2$  is provided, the outer loop can then be put in CASCADE.

The PM550 permits the mode transitions to occur in this sequence only. The PM550 will not permit the outer loop to be placed in AUTO unless the inner loop is in CASCADE. The PM550 permits any switch to be opened at any time, but the PM550 will automatically open any switches further out in the chain. For example, placing the inner loop on AUTO (which also means open cascade) causes the PM550 to place the outer loop in MANUAL (which also implies open cascade for the outer loop). Similarly, placing the inner loop in MANUAL (and thus open cascade) causes the PM550 to place the outer loop in MANUAL.

#### 7.1.4 Alarms

Process alarms comprise an integral part of the loop calculation features in the PM550. These alarms are of two types:

1. Alarms on the process variable.
2. Alarms on the control error or deviation (setpoint - process variable)

Each is described in detail in the following sections. Through bits in the image register, the status of all alarms is made available to the ladder logic program. If the prompt "LOOP FLAGS" is answered YES. The status of the alarms is also indicated by the alarm display lights on the operator station, as illustrated in Figure 7.1B.

#### 7.1.4.1 Process Variable Alarms

The PM550 provides for two process variable alarms:

1. High alarm occurs when the process variable exceeds a user-specified value known as the high alarm limit.
2. Low alarm occurs when the process variable becomes less than a user-specified value known as the low alarm limit.

Associated with each alarm is a deadband provided to prevent "chatter"

Figure 7.1M illustrates the process variable alarm limits and deadband. An alarm occurs when the process variable goes outside of either limit. The return-to-normal does not occur until the process variable returns to within the alarm limit with the associated deadband.

The operator station only displays the status of the alarms for the particular loop being displayed. Should a process variable alarm occur for a loop not being displayed, the loop display will begin to flash. The operator should then depress the loop increment button. The operator station will then display the loop for which the alarm condition exists. By this procedure, the operator acknowledges that he is aware of the existence of this alarm, and this alarm will no longer cause the loop display to flash. Acknowledging the alarm on any LAM will call all LAMS in the system to stop flashing. Should the process variable return to normal and then enter the alarm state again, the flashing will re-occur.

On some occasions, two or more unacknowledged alarms may exist at the same time. Depressing the loop increment button will cause the station to display the lowest numbered loop with an unacknowledged alarm, but the display will continue to flash. Depressing the loop increment button a second time will cause the station to display the next loop with an unacknowledged alarm. The operator should repeat this procedure until all alarms have been acknowledged.

During loop specification, the engineer must provide values for each process variable alarm limit. Should he not wish to specify alarms, the engineer should insert default values so that the alarms will never occur. The process variable alarms are always processed, regardless of whether the loop is in MANUAL, AUTO, or CASCADE.

#### NOTE

Loop alarms are stored as scaled integers, therefore, the value read back on the read/write programmer may differ slightly from the value programmed due to round off.

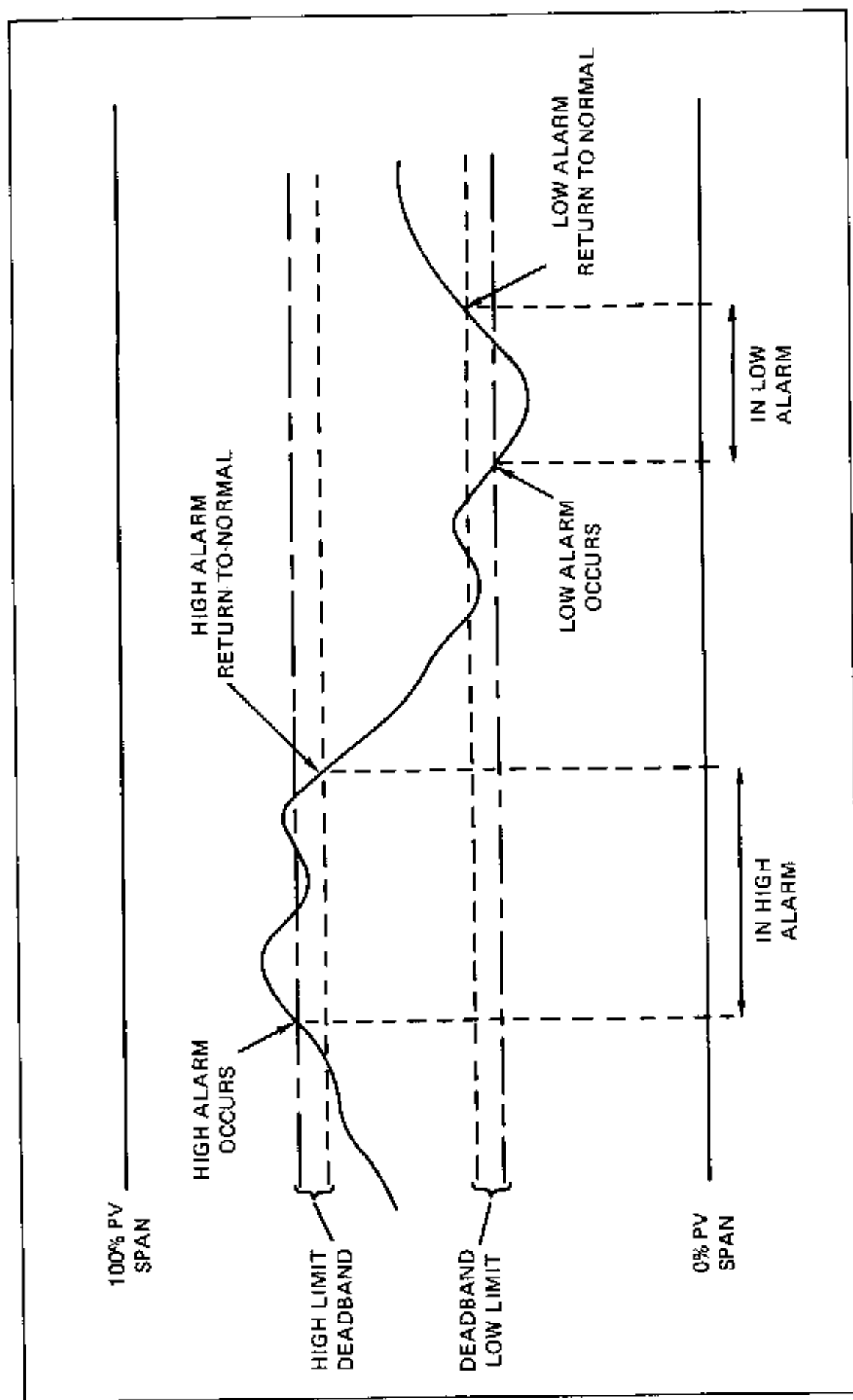


Figure 7.1M. Process Variable Alarms

### 7.1.4.2 Deviation Alarms

The PM550 provides for two levels of deviation alarms:

1. A yellow deviation limit.
2. An orange deviation limit.

The deviation alarm bands are always centered about the setpoint, or said another way, the deviation alarm test is actually on the control error. Since these alarms are on the control error, they are only processed while the loop is in AUTO or CASCADE.

Figure 7.1N illustrates the deviation alarm bands. In creating the loop tables, the user must enter values for the yellow deviation limit,  $\Delta_Y$ , and the orange limit,  $\Delta_O$ . If the absolute deviation is less than  $\Delta_Y$ , then the green deviation light will be illuminated on the LAM. If the absolute deviation is greater than  $\Delta_Y$  and less than  $\Delta_O$ , the high or low yellow deviation light will be illuminated, depending upon whether the error is above or below the setpoint. If the absolute deviation exceeds  $\Delta_O$ , then either the high or low orange deviation limit will be illuminated. As illustrated in Figure 7.1N, a deadband is incorporated at each limit to prevent chatter.

The loop display will flash only when the deviation enters the orange band, i.e., the absolute error exceeds  $\Delta_O$ . Thus, the operator must only acknowledge the orange deviation alarms. The loop flashing for the process variable alarms and the orange deviation alarms is identical, so the operator will not know which alarm has occurred nor in which loop until he acknowledges the alarm.

Through bits in the image register, the ladder logic program can ascertain the statuses of both the yellow and orange deviation alarms and can determine if the alarm is in the high or low band. However, the CCU cannot acknowledge an alarm.

#### NOTE

Loop alarms are stored as scaled integers; therefore, the value back on the read/write programmer may differ slightly from the value programmed due to round off.

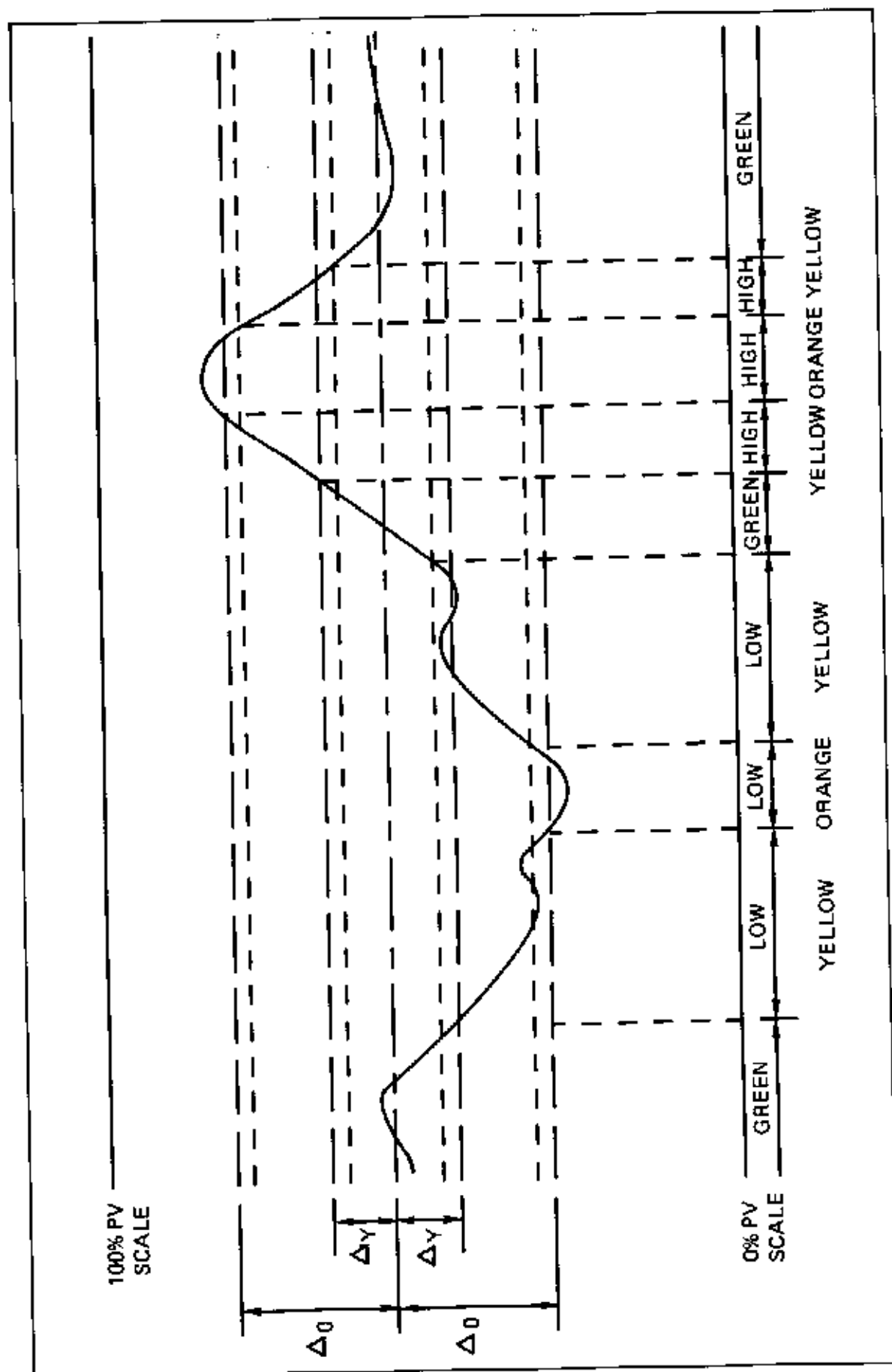


Figure 7.1N Deviation Alarms

## 7.2 SPECIFYING LOOP CONFIGURATIONS

With the aid of prompting commands from the read/write programmer, the loop configuration and associated constants are entered into tables. These commands are divided into four categories:

1. Loop table memory allocation
2. Process variable specifications
3. Control specifications
4. Alarm specifications

For ease of reference in this manual, each entry has been assigned a step number. However, the programmer does not display or accept a step number.

In response to a prompt from the programmer, one of the following types of information must be entered:

1. Numerical data
2. YLS, NO
3. Memory Address

The type of data to be entered can normally be inferred from the message in the prompt. Each prompt that elicits a response ends with either an equal sign (=), a question mark (?), or a colon (:). The use of these and some examples are given in Figure 7.2A.

It is also essential that a written record be maintained for each loop in the PM550 and that this record become an integral part of the documentation for the PM550 system. For this purpose, the Loop Specification Sheet is provided. Figure 7.2B illustrates a Loop Specification Sheet that has been completed for a typical loop. When all entries are completed, this sheet provides the information required to answer all prompts from the programmer. The Loop Specification Sheet requests some information (for example, engineering units for the process variable) that is not entered into the PM550. On the Loop Specification Sheet, such information is shown enclosed in quotation marks (" "). Although such information could certainly be omitted, good system documentation invariably proves useful.

Last Character in Prompt	Type of Data to be Entered	Examples
=	Numeric data	LOOP NUMBER= PV LOW RANGE= GAIN (%%)=
?	YES/NO	TUNE IN V? SQRT OF PV? REVERSE ACTION?
.	Address	CONSTANT TABLE: PV ADDR. SP ADDR

Figure 7 2A. Data Entry Conversions

A blank specification sheet is provided for your convenience in Appendix A. It may be duplicated without prior permission from Texas Instruments



## LOOP SPECIFICATION SHEET

Loop Description: "Process Hot Water Temperature Controller"

Tas: TC107

PM550 Loop Number = 5

### MEMORY ALLOCATION

Tuning Constants in V (C or V)

	Beginning Address	Tune C	Tune V	Ending Address
Constant Table	C61	21	15	<u>C75</u>
Variable Table	V41	12	18	<u>V58</u>

Are loop flags for alarms and mode switching allocated in the image register? YES If YES, give beginning address: CR29

### PROCESS VARIABLE

20% Offset? YES

Special calculation? NO

Low Range = 50

Engr. Units: DEGF

Address: A502

Square Root? NO

If YES, give address: \_\_\_\_\_

High Range = 250

Transmitter: TX 107

### CONTROL CALCULATIONS

Remote Set Point? NO

Special Calculation? NO

LOCKS—Set Point? YES

Error Squared? NO

Gain: 1.00%/%

Rate = 0.0 min

Output Address: V78

Sample Time = 15.0 sec

If YES, give address: \_\_\_\_\_

If YES, give address: \_\_\_\_\_

Auto/Manual? YES, Cascade? NO

Error Deadband? NO

Reset Time: 999.9 min

Reverse Acting? NO

20% Offset? NO

### ALARMS

Process Variable

Low = 120

Deviation

Yellow = 2

High = 160

Orange = 10

Figure 7.2B. Typical Loop Specification Sheet

## 7.2.1 Loop Table Memory Allocation

The loop tables contain all of the configuration information and data values required to completely define a loop. Part of this information will not change during the execution of the loop, but some information will change as the loop calculations progress. Thus, the table for storing the data for each loop is divided into two segments, one residing in the C-area and the other residing in the V-area.

It is essential that the user maintain an up-to-date memory allocation record for both the C-area and the V-area. The length of each table depends upon where the tuning constants are to be stored (either in V or in C). Figure 7.2C gives the length of each table for both options. The user must supply a starting address for each of these tables. The PM550 calculates the ending address of each table, and assumes that these locations are not being used for any other purpose.

As illustrated in Figure 7.2D, the entry of the loop table address data entails six steps.

**Step 1** To insure that the read/write programmer is ready to accept the loop specification data, the ☐ **CR** key should be depressed.

**Step 2** To inform the read/write programmer that a loop is to be defined, the ☐ **LOOP** key must be depressed. The read/write programmer requests the loop number by displaying "LOOP=". The loop number must be a number in the range of 1 to 8.

**Step 3** After the ☐ **STEP** key is depressed, the read/write programmer displays "CONSTANT TABLE:" to request the starting address of the loop table in the C-area. Any address in the C-area may be entered with the exception of C0 through C15 which are reserved to store loop table addresses.

**Step 4** After the ☐ **STEP** key is depressed, the read/write programmer displays "VARIABLE TABLE:" to request the starting address of the loop table in the V-area. Any address in the V-area may be entered.

**Step 5** Tuning constants may reside in either the C-area or in the V-area. When read/write memory is used for the C-area, there is no difference in the behavior of the system. Thus, the logical choice might be the area that has the most unused locations. When read only memory is used for the C-area, then the tuning constants can be changed only when they reside in the V-area. After depressing the ☐ **LOOP** key, the read/write programmer prompts VAR TUNE?. If the ☐ **YES** key is depressed, the tuning parameters will be key IN V memory. ☐ **NO** causes tuning parameter to be stored in C Memory.

Step 6. After the ☐ key is depressed, the read/write programmer displays the ending address (specifically, the last location used) of both tables.

#### **\*WARNING**

As the PM550 assumes that all locations between the start address and the end address, inclusive, are available for the loop tables, the memory allocation should be carefully checked to be sure that no intermediate location is being used for other purposes. At the time the read/write programmer displays these addresses, nothing has been written into memory. Consequently, the table entry can be aborted if necessary before any contents of memory are altered.

Step 7. After depressing ☐ , the message "LOOP FLAGS?" is displayed. The user may optionally specify that loop mode and alarm status information be passed to the ladder logic program via the image register. The use of the respective image register bits will be given in Section 7.2.5. If the response to "LOOP FLAGS?" is yes, then the address of the first CR or Y bit to be used for the loop flags must be entered in response to "LOOP FLAGS." If the response to "LOOP FLAGS?" is no, the message "LOOP FLAGS:" will not appear.

	Tuning Constants in C	Tuning Constants in V
C - Table Length	21	15
V - Table Length	12	18

Figure 7.2C. Length of V and C Tables

Step	Key	RWP Display
1	CLR	READY (RUN)
2	LOOP	LOOP NO =
	1	LOOP NO = 1
3	STEP	CONSTANT TABLE
	CR105	CONSTANT TABLE: C105
4	STEP	VARIABLE TABLE
	V27	VARIABLE TABLE: V27
5	STOP	VAR TUNE?
	Y..S	VAR TUNE? YES
6	STEP	C119/V44 END
7	STEP	LOOP FLAGS?
	Y..S	LOOP FLAGS? YES
7A	STEP	LOOP FLAGS
	CR105	LOOP FLAGS: CR105

Figure 7.2D. Typical Sequence for Entering Loop Table Memory Allocations

## 7.2.2 Process Variable Specifications

The next section of the data entry procedure pertains to the process variable specifications. The following information must be specified:

1. The address of the process variable
2. The 20% offset option (yes/no)
3. Square root of the process variable (yes/no)
4. Special calculation for the process variable (yes/no), and if so, the address of the special calculation
5. Low range for the process variable
6. High range for the process variable

Figure 7.2E illustrates a typical sequence of commands for entering the process variable specifications

**Step 8.** After  is depressed, the read/write programmer displays "PV ADR:". The process variable address can be either an AIM address, a C-area address, or an V-area address. Although the programmer will accept a C-area address for the PV, this makes sense only when the C-area is read/write memory.

**Step 9.** After  is depressed, the message "PV 20% OFFSET?" is displayed. If the range is 20% to 100%, respond . The  response indicates a 0% to 100% input range.

### NOTE

The 20% offset option applies only to the process variable; specifically it does not apply to the set point.

**Step 10.** After  is depressed, the read/write programmer displays "SQRT OF PV?". Respond  if the square root of the process variable is desired, if not, respond . A YES response causes the square root to be taken of the value specified for the process variable. The square root will be used in loop calculations.

**Step 11.** After  is depressed, the message "SPEC CALC ON PV?" is displayed. Enter  if a special function is to be executed using the value of the process variable. If  is entered, Step 11a is bypassed (as is illustrated in Figure 7.2E). If the square root option is chosen, the square root will be taken prior to the special function execution.

**Step 11a.** After  is depressed, the read/write programmer displays "PV CALC ADR:". Enter the V-area or C-area address where the special function is stored.

STEP	KEY SEQUENCE	DISPLAY
8	STEP A103	PV ADR: PV ADR: A103
9	STEP YES	PV 20% OFFSET? PV 20% OFFSET? YES
10	STEP NO	SQRT OF PV? SQRT OF PV? NO
11	STEP NO	SPEC CALC ON PV? NO SPEC CALC ON PV? NO
12	STEP 100.0	PV LOW RANGE = PV LOW RANGE = 100.0
13	STEP 500.0	PV HI RANGE = PV HI RANGE = 500.0

**Figure 7.2E. Typical Sequences for Entering the Process Variable Specifications**

**Step 12.** After  is depressed, the message "PV LOW RANGE=" is displayed. Enter the engineering value corresponding to the low range of the input (20% if "YES" response to Step 9; 0% if "NO" response to Step 9).

**Step 13.** After  is depressed, the read/write programmer displays "PV HI RANGE=". Enter the engineering value corresponding to 100% on the input range.

### 7.2.3 Control Specifications

The third section of the data entry procedure pertains to the control specifications. The following information must be specified:

1. Sample rate (in seconds)
2. Source of setpoint
3. Special calculation on setpoint
4. Locks (setpoint, auto/manual, and cascade)
5. Error calculations (error squared or error deadband)
6. Tuning constants (gain, reset time, and derivative time)

- 7. Direct/reverse acting
- 8. Address for output
- 9. Output 20% offset

Figure 7.2F illustrates a typical sequence of commands for specifying the control configuration for a loop.

**Step 14.** The sample time must be entered in seconds to the nearest half second. Since the loop processing cycle time of the PM550 is one-half second, the sampling time for the loop must be an integer multiple of 0.5 seconds. The shortest sampling time 0.5 seconds; the longest permissible sampling time is 4095.5 seconds.

**Step 15.** If the response to the prompt "REMOTE SP?" is NO, then the setpoint can only be entered from the LAM. The control mode "closed cascade" will not be permitted, even if the operator or the CCU requests cascade. If the response to the prompt "REMOTE SP?" is YES, an address for the setpoint must be entered, as illustrated in Step 15A in Figure 7.2F. The memory address may be either a C area or an V area. The value in this location is assumed to be a scaled integer value. The value may also be taken from A, but a special function will be required if the value has a 20% offset.

**Step 16.** If no PV special calculation was specified in Step 11, the prompt "SPEC CALC ON SP?" appears. If the response is YES, then the address of the special calculation must be entered in Step 16A. If the answer is NO, Step 16A is skipped. If a PV special calculation was specified, then Step 16 is bypassed.

**Step 17.** For each loop in the PM550, locks may be specified on each of the following:

- 1. Setpoint
- 2. Auto/manual mode
- 3. Cascade mode

When specified, these locks are in effect only when the keylock is in the "Operate" position. When in the "Tune/Unlock" position, the locks are over-riden. This feature permits the engineer to set up a particular loop and then prevent the operator from making subsequent alterations.

As a practical matter, the cascade lock is only effective when the auto/manual lock is set. Thus, when the response to Step 17B is NO, Step 17C is bypassed.

All locks pertain only to LAM, specifically, they do not pertain to the ladder logic. For example, the setpoint lock has no effect when a loop is in closed cascade, as the sequence or some other loop may change the setpoint regardless of the status of the setpoint lock.

**Step 18** The control calculations are normally based on the error (setpoint minus process variable), but may optionally be based on error squared or error deadband (but not both). When the error squared option is specified, Step 18b is bypassed.

**Step 19** An address must be supplied for the output from the control calculations. This address would normally be either an AIM address or an V area address. A C-area address should be provided only if the C-area is read/write memory.

**Step 19a** When the output to AIM is to be 4-20 ma or 2-10 volt signal, a 20% offset must be specified for the output.

**Step 20** Values must be entered for the PID control equation's three tuning parameters: gain (in %/%), reset time (in minutes), and derivative time (in minutes). The special cases of the PID control equation are obtained as follows:

**PI control (no derivative):** enter zero for the derivative time

**No reset action:** enter 999.9 for the reset time

**Integral-only control:** enter zero for the gain

**Step 21** A direct acting controller has a positive gain, i.e., a positive error results in a positive change in the output from the controller. A reverse acting controller has a negative gain, i.e., a positive error results in a negative change in the output from the controller. If a direct acting controller is desired, the prompt "REVERSE ACTING?" should be answered NO.

## 7.2.4 Alarm Specifications

The final section of the data entry procedure pertains to the alarm specifications. The following parameters must be specified:

1. Low alarm limit
2. High alarm limit
3. Yellow deviation limit
4. Orange deviation limit



STEP	KEY SEQUENCE	DISPLAY
14	<div>STEP</div> <div>1 . 0</div>	SAMPLE (SEC) = SAMPLE (SEC) = 1.0
15	<div>STEP</div> <div>YES</div>	REMOTE SP? REMOTE SP? YES
15A	<div>STEP</div> <div>V 9 1</div>	SP ADDR SP ADDR. V91
16	<div>STEP</div> <div>YES</div>	SPEC CALC ON SP? SPEC CALC ON SP? YES
16A	<div>STEP</div> <div>V 1 0</div>	SP CALC ADDR: SP CALC ADDR. V10
17A	<div>STEP</div> <div>NO</div>	LOCK SP? LOCK SP? NO
17B	<div>STEP</div> <div>YES</div>	LOCK AUTO/MAN? LOCK AUTO/MAN? YES
17C	<div>STEP</div> <div>NO</div>	LOCK CASCADE? LOCK CASCADE? NO

Figure 7.2F. Example of a CCU of Commands for Entering Control Applications

STEP	KEY SEQUENCE	DISPLAY
18A	<div>STEP</div> <div>NO</div>	ERROR SQ? ERROR SQ? NO
18E	<div>STEP</div> <div>NO</div>	DEADBAND? DEADBAND? NO
19	<div>STEP</div> <div>A</div> <div>2</div> <div>0</div> <div>0</div>	OUTPUT ADDR. OUTPUT ADDR: A200
19A	<div>STEP</div> <div>YES</div>	OUT 20% OFFSET? OUT 20% OFFSET? YES
20	<div>STEP</div> <div>3</div> <div>2</div> <div>2</div>	GAIN (%:%) = GAIN (%:%) = 3.2
21	<div>STEP</div> <div>YES</div>	REVERSE ACTING? REVERSE ACTING? YES
22	<div>STEP</div> <div>2</div> <div>5</div> <div>0</div>	RESET (MIN) = RESET (MIN) = 50
23	<div>STEP</div> <div>3</div> <div>5</div> <div>0</div>	RATE (MIN) = RATE (MIN) = 0

Figure 7.2F Example of a CCU of Commands for Entering Control Applications (Cont)

Figure 7.2G illustrates a typical response to commands for entering the alarm specifications. Note that all alarm parameters are specified in the engineering units of the process variable.

**Step 22.** The value of the low alarm is entered in the engineering units of the process variable. If no low alarm is desired, enter a value equal to the lower range of the process variable.

**Step 23.** Enter a value for the high alarm in the engineering units of the process variable. If no high alarm is desired, enter a value to the upper range of the process variable.

**Step 24.** Enter a value for the yellow deviation alarm in the engineering units of the process variable. If the error deadband option is the control action is selected, this value will also be used as the error deadband. If no yellow deviation alarm is desired, enter a value equal to the span (upper range minus lower range) of the process variable.

**Step 25.** Enter a value for the orange deviation alarm in the engineering units of the process variable. If no orange deviation alarm is desired, enter a value equal to the span (upper range minus lower range) of the process variable.

**Step 26.** The display of "ENTER LOOP?" signifies that all necessary specifications have been entered for the control loop. If the response is ☐ YES, "PRESS ENTR TO ENTER" will be displayed. Depressing the ☐ ENTR key then enters the loop into the CCU.

STEP	KEY	RWP DISPLAY
22	<input type="checkbox"/> STOP	LOW ALARM =
	<input type="checkbox"/> 2 <input type="checkbox"/> 5 <input type="checkbox"/> 0	LOW ALARM = 250
23	<input type="checkbox"/> STOP	HIGH ALARM =
	<input type="checkbox"/> 3 <input type="checkbox"/> 5 <input type="checkbox"/> 0	HIGH ALARM = 350
24	<input type="checkbox"/> STOP	YELLOW DEV =
	<input type="checkbox"/> 1 <input type="checkbox"/> <input type="checkbox"/> 5	YELLOW DEV = 15
25	<input type="checkbox"/> STOP	ORANGE DEV =
	<input type="checkbox"/> 5	ORANGE DEV = 5
26	<input type="checkbox"/> STOP	ENTER LOOP?
	<input type="checkbox"/> YES	PRESS ENTR TO ENTER

Figure 7.2G. Typical ☐ ENTR Sequence for Entering Alarm Specifications

## 7.2.4 Loop Tables

The data entered via the read/write programmer is used to build the loop tables, part of which resides in the C-area and the remainder in the V-area. The data in the C table is normally not changed after the loop has been configured, whereas the data in the V table will change.

Although users of the PM550 will rarely need to access the data in these tables other than via the loop specification features of the read/write programmer, the structure of these tables will be presented. The memory locations can be displayed and even altered by any method for accessing the PM550's user memory. However, it is not recommended that the loop tables be altered in this way.

Figures 7.2H and 7.2J provide the structure and definition of the entries in the loop tables.

## 7.2.5 Image Register to Loops

As stated earlier, the user has the option of specifying that ten image register bits be allocated for the purposes of communicating between a loop and the ladder logic program. These bits are used for the following purposes:

1. Indicate the status of alarms to the ladder logic
2. Indicate the status of the manual/auto/cascade modes to the ladder logic
3. Enable the ladder logic to request that the loop be placed in manual, auto, or cascade mode.

Figure 7.2K gives the specific use of each of the ten bits. When the loop configuration is entered, the user must specify the address of the first bit. The remaining bits are located sequentially.

## 7.3 Loop Errors

The following errors (Figure 7.3A) can occur when attempting to program a loop. These errors will be displayed on the programmer when the loop is entered into the CCU. If an error occurs, the loop will be entered into the CCU as is, but will not be executed until the error is corrected.

To correct the error, read the loop back via the programmer by depressing the  key after specifying the loop number. Step through the prompts and responses until the response in error is encountered. Depress the  key and re-enter the correct response. Step through the remaining prompts and enter the loop into the CCU.

0	LOOP NO.	SAMPLE RATE
1	C – Table Flag Word	
2	Special Function Address	
3	Process Variable Address	
4	Set Point Address	
5	Output Address	
6	Process Variable	
7	Span	
8	Process Variable	
9	Lower Range	
10	Low Alarm Limit	
11	Low Alarm Limit	
12	Yellow Deviation Limit	
13	Orange Deviation Limit	
14	Address of CR Bits for Loop	
15	Proportional Gain	
16	$K_C$	
17	Reset Coefficient	
18	$K_C \cdot T / T_i$	
19	Derivative Coefficient	
20	$T_D / T_S$	

Figure 7.2H C-Table Structure

WORD	NOTES
0	<p>Loop Number is stored as 0 - 7 in high order 3 bits instead of 1 - 8</p> <p>Sample rate is the number of half seconds between control calculations for the loop Sample rate is stored in low order 13 bits</p>
1	<p>Bits in the flag word are used as follows.</p> <ul style="list-style-type: none"> <li>0    0 = Tuning parameters in C       1 = Tuning parameters in V</li> <li>1    0 = Do not take square root of PV       1 = Take square root of process variable</li> <li>2    0 = Direct Acting       1 = Reverse Acting</li> <li>3    1 = Control based on error squared</li> <li>4    1 = Control based on error deadband</li> <li>5    0 = No auto lock       1 = Auto lock</li> <li>6    0 = No cascade lock       1 = Cascade lock</li> <li>7    0 = No setpoint lock       1 = Setpoint lock</li> <li>8    0 = PV Scale is 0% - 100%       1 = PV Scale is 20% - 100%</li> <li>9    0 = Output Scale is 0% - 100%       1 = Output Scale is 20% - 100%</li> <li>10   0 = No PV Special Function       1 = Special function on PV</li> <li>11   0 = No set PT Special Function       1 = Special Function on Set point</li> <li>12   0 = Bypass all Calculation for loop       1 = Process loop normally</li> <li>13 } 14 }        Decimal point location in LAM Display 15 }</li> </ul>
2	Special function memory address (zero if no special function)
3	Process variable memory address
4	Set point memory address (zero if closed cascade not permitted)

Figure 7.2 H C-Table Structure (Cont)

WORD	NOTES
5	Memory address for loop output
6,7	Span of process variable (floating point value)
8,9	Lower range of process variable (floating point value)
10	Low alarm limit (scaled integer value is fraction of full scale)
11	High alarm limit (scaled integer value is fraction of full scale)
12	Yellow deviation alarm (scaled integer value is fraction of full scale)
13	Orange deviation alarm (scaled integer value is fraction of full scale)
14	Address of first CRU bit for loop (-1 means no CRU bits are reserved for this loop)
15,16*	Proportional gain $K_c T_s / T_i$ (floating point value)
17,18*	Reset coefficient, $K_c T_s / T_i$ (floating point value)
19,20*	Derivative coefficient, $T_D / T_s$ (floating point value)

\*These words reserved only when bit zero of C-Table flag word 13 clear = 0, i.e. tune in C.

Figure 7.2.H C-Table Structure (Cont)

WORD	LOOP NO	SAMPLE COUNT
0		
1	V — Table Flag Word	
2	Process	
3	Variable	
4	Setpoint	
5		
6	Error	
7		
8	Output	
9		
10	Bias	
11		
12	Proportional Gain,	
13	$K_C$	
14	Reset Coefficient,	
15	$K_C T_S / I_1$	
16	Derivative Coefficient,	
17	$T_D / I_S$	

Figure 7 2J V-Table Structure



WORD	NOTES
0	<p>Loop number is stored in high order 3 bits as 0–7 instead of 1–8</p> <p>Sample rate is the number of half seconds until the next control calculations will be performed. Stored in low order 13 bits</p>
1	<p>Bits in the flag word are used as follows:</p> <ul style="list-style-type: none"> <li>0    0 = Not in yellow deviation alarm       1 = In yellow deviation alarm</li> <li>1    0 = Not in orange deviation alarm       1 = In orange deviation alarm</li> <li>2    0 = Error is negative       1 = Error is positive</li> <li>3    0 = Not in low alarm       1 = In low alarm</li> <li>4    0 = Not in high alarm       1 = In high alarm</li> <li>5    0 = Open cascade       1 = Closed cascade</li> <li>6    0 = Manual       1 = Auto</li> <li>7    0 = Deviation alarm not acknowledged       1 = Deviation alarm acknowledge</li> <li>8    0 = HI/LO alarm not acknowledged       1 = HI/LO alarm acknowledge</li> <li>9       Not Used</li> <li>10   1 = LAM requesting cascade</li> <li>11   1 = LAM requesting auto</li> <li>12   1 = LAM requesting manual</li> <li>13   1 = CCU requesting cascade</li> <li>14   1 = CCU requesting auto</li> <li>15   1 = CCU requesting manual</li> </ul>
2,3	Process variable (floating point value stored as a fraction of full scale)
4,5	Setpoint (floating point value stored as a fraction of full scale)

Figure 7.2J. V-Table Structure (Cont)

WORD	NOTES
6,7	Error (floating point value stored as a fraction of full scale)
8,9	Output (floating point value stored as a fraction of output range)
10,11	Bias (floating point value stored as a fraction of output range)
12,13*	Proportional gain, $K_C$ (floating point value)
14,15*	Reset coefficient, $K_C T_S / T_I$ (floating point value)
16,17*	Derivative coefficient, $T_D / T_S$ (floating point value)

\*These words reserved only when bit 0 of C -- Table flag word is set (= 1).

Figure 7.21. V -Table Structure (Cont)

BIT	SET BY	MEANING OF 0	MEANING OF 1
1	CCU	Not requesting manual mode	Requesting manual mode
2	CCU	Not requesting auto mode	Requesting auto mode
3	CCU	Not requesting closed cascade	Requesting closed cascade
4	Loops	Loop is in manual	Loop is in auto
5	Loops	Loop is in open cascade	Loop is in closed cascade
6	Loops	PV is not in high alarm	PV is in high alarm
7	Loops	PV is not in low alarm	PV is in low alarm
8	Loops	Negative error deviation	Positive error deviation
9	Loops	Deviation is not in orange band	Deviation is in orange band
10	Loops	Deviation is not in yellow band	Deviation is in yellow band

**Figure 7-2K IR Bits Associated with Loops**

ERROR #	DISPLAY	MEANING
1	ER 1 OUT OF BOUNDS	User tried to access non-existent C or L
2	ER 2 COMMUNICATIONS	CCU found a non-ASCII-REX character in the communications stream (system error).
3	ER 3 WRONG MODE	CCU is in the wrong mode (state) to do the requested function.
4	ER 4	CCU found unexpected data in communication stream (system error).
5	ER 5 NUMBER RANGE	CCU found a number in the communication stream out of the expected range, i.e. clear error in system table can be no larger than 46.
6	ER 6 CCU FATAL ERROR	CCU found an error that would indicate a CCU hardware failure while attempting to carry out user request.
7	ER 7	Invalid task received by CCU (system error)
8	ER 8	Spare
9	ER 9 REMOVE CONTROL	Tape deck is either not in remote mode or tape was not rewound back to start
10	ER 10 PVSPAN = 0	Values for PV high = PVLOW or user has set internal value for PVSPAN to zero
11	ER 11 PV RANGE	Decimal point for PVHIGH out of range
12	ER 12 LOOP NUMBER	System encounter a loop number less than 1 or greater than 8.
13	ER 13 FLOATING PT	General floating point math error.
14	ER 14 REWIND TIMEOUT	Tape rewind did not occur within 40 seconds.
15	ER 15 SAMPLE = 0	Sample rate has become zero in loop table
16	ER 16	
17	ER 17 RESET = 0	Reset has become zero in loop table
18	ER 18 NO LOOPS	No loop table exist in the system (C0 — C15 = 0).

Figure 7 3A Loop Error Table

ERROR #	DISPLAY	MEANING
19	ER 19 LOOP TABLE	Invalid address found for this particular loop's C TABLE found in C0→C15.
20	ER 20 PV LOW	Error encountered writing/updating PV low
21	SPARE	
22	ER 22 PV HI	Error encountered writing/updating PV high.
23	ER 23 PVHI < PVLOW	Process variable high bound cannot be less than process variable low bound
24	ER 24 SAMPLE/RATE	Error encountered writing/updating rate while changing sample interval.
25	ER 25 SAMPLE/RESET	Error encountered writing/updating reset while changing sample interval
26	ER 26 GAIN	Error encountered writing/updating gain.
27	ER 27 GAIN/RESET	Error encountered writing/updating reset while changing gain
28	ER 28 RESET	Error encountered writing/updating reset.
29	ER 29 RATE	Error encountered writing/updating rate
30	ER 30 ALARM/DEV	Error updating orange, yellow, hi, low alarms
31	ER 31	Spare
32	ER 32	Spare
33	ER 33	Spare
34	ER 34 PV HI	Error reading PV high
35	ER 35 RESET	Error reading reset
36	ER 36 RATE	Error reading rate
37	ER 37 ALARM/DEV	Error reading orange, yellow, hi, low alarms
38	ER 38 TAPE TIMEOUT	Requested tape operation did not complete in 4 1/2 minutes.
39	ER 39	Spare

Figure 7 3A Loop Error Table (Cont)








## CHAPTER 8

### AUXILIARY FUNCTIONS

8.0 Auxiliary functions are used in an off-line mode of operation. These functions perform a number of tasks such as clearing memory, self-diagnostics, reading error table, etc. Each function will be covered in detail.

#### 8.1 AUX 1 - CLEAR MEMORY

The clear memory auxiliary function allows the three user memory areas to be cleared selectively. The required procedure is detailed below. The startup run switch must be placed in startup to clear memory.

KEY SEQUENCE	DISPLAY	NOTE
	READY (START)	
 	AUX = 1	No location address is required for an AUX function.
	CLEAR L ?	Image register is cleared when the L-area is cleared.
	PRESS ENTER TO CLEAR	Memory is cleared only if enter is pressed.
 	CLEAR V ? NO	If no is answered, the next memory area prompts appears after step.

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="STEP"/>	CLFAR C	Loops are cleared when the C area is cleared.
<input type="button" value="YES"/>	PRESS ENTR TO CLEAR	
<input type="button" value="ENTR"/>	READY (START)	

In the above example, the ladder logic (L) and the constant data (C) areas of user memory are cleared. The variable data (V) memory is not cleared. A "WRONG MODE" message indicates an attempt to clear memory with the startup/run switch in the run position.

## 8.2 AUX 2 - PROGRAM MODE/SINGLE SCAN

The program mode/single scan auxiliary function allows start/stop control of the ladder logic processor while loops continue to be processed. When program mode is active, the ladder logic processor is stopped. This provides a safe mode where extensive logic program modification can be made without affecting the state of the auxiliary I/O and loop procedures. Run mode restarts the ladder logic processor. In the program mode, a single logic scan can be initiated by the single scan command. This feature allows static debugging of rapid events like single scanning a one-shot, etc.

To activate program mode and/or the single scan functions, follow the steps below

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="AUX"/> <input type="button" value="2"/>	AUX - 2	No location address is required for any AUX function.
<input type="button" value="STOP"/>	PROGRAM MODE ?	
<input type="button" value="YES"/>	PRESS ENTR TO PGM	CCU enters program mode when ENTR is pressed.
<input type="button" value="ENTR"/>	SINGLE SCAN ?	Single logic scan occurs when ENTR is pressed.

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="YES"/>	PRESS ENTER TO SCAN	
<input type="button" value="ENTER"/> <input type="button" value="NO"/>	SINGLE SCAN ? NO	The single scan can be repeated as required.
<input type="button" value="STOP"/> <input type="button" value="NO"/>	RUN MODE ? NO	Leaves CCU in program mode
<input type="button" value="STOP"/>	READY (PGM RUN)	

To return to run mode from the program mode press the following keys:

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="ENTER"/>	READY (PGM RUN)	
<input type="button" value="AUX"/> <input type="button" value="2"/>	AUX = 2	
<input type="button" value="STOP"/> <input type="button" value="NO"/>	PROGRAM MODE ? NO	
<input type="button" value="STOP"/> <input type="button" value="YES"/>	RUN MODE ?	
<input type="button" value="YES"/>	PRESS ENTER TO RUN	
<input type="button" value="ENTER"/>	READY (RUN)	

### 8.3 AUX 3 – PROGRAMMER SELF-TEST

The Read Write programmer has extensive self test programs built in. These programs are initiated using AUX 3. This function must be performed with the Read/Write programmer disconnected from the CCU. AUX 3 is a stand alone test which can be run to test the programmer at any time. An RS232C loop back connector should be connected to the programmer prior to starting the test.



Follow the instructions below for operation of AUX 3. Once the self test is started it will continue until  is pressed. If the operator does not answer the prompts on the keyboard and display tests, they are assumed to have passed. The program flow of the test sequence is given in Figure 8.3A.

1. Insure the unit has a RWP loopback connector plugged into the back. If not, then the RS232C test will fail.
2. Insure the power switch is in the off position.
3. Plug the RWP into AC power
4. Turn on the RWP
5. Verify that "READY" is displayed.
6. Press  and verify that "READY" is displayed.
7. Press
8. "RWP SELF TEST" is displayed. Press
9. "REV" is displayed followed by 3 integers for each ROM. Each REV ID is displayed for 2.5 seconds. The numbers are used by TI to track ROM revisions for the Read/Write programmer.
10. After 5 seconds, verify the display shows:  
"ROM O K B O"
11. After 5 seconds, verify the display shows:  
"DSP O RS232C O"
12. After 5 seconds, verify the display shows:  
"RAM O"
13. After 15 seconds, verify the display shows:  
"KEYBOARD TEST"
14. Starting with the  key, press keys from left to right, down, right to left, down, etc. If the message:  
"ERROR TRY AGAIN"  
is displayed, a key was pushed out of sequence. Push the correct key and continue. If the display does not remain blank after pushing the correct key, then continue with step DO NOT push the  key as the last key in the sequence.

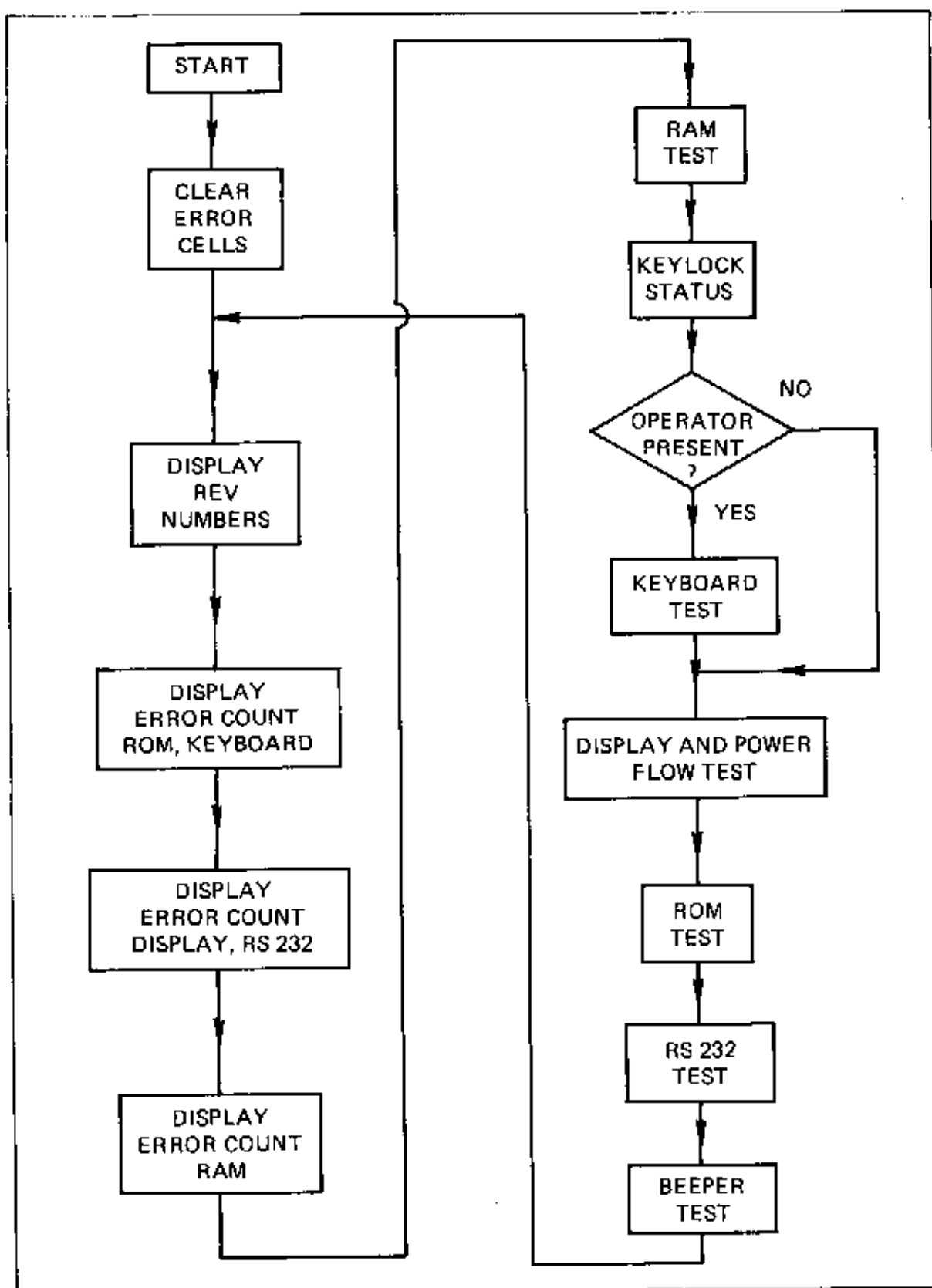


Figure 8.3A. Read/Write Programmer Self-Test Program Flow

15. The RWP displays all printable ASCII characters and all matrix dots. Check that the characters are displayed correctly and that all the dots are turned on at the end of the test. Power flow indicator will turn on and off during the display test.
16. The RWP displays: "DISPLAY GOOD ? Y OR N"
17. Press ☐ YES if the display operated correctly ☐ NO if any characters did not display correctly or if any of the dots did not turn on. If no key is pressed, the RWP defaults to YES after 5 seconds
18. Beeper test will be displayed for 25 seconds and the beeper will beep twice

#### 8.4 AUX 4 - CENTRAL CONTROL UNIT (CCU) SELF TEST

The central control unit (CCU) also contains extensive self-test programs. AUX 4 is used to start these programs. See Section 8.5 (AUX 5 - Read/Clear error table) to interpret the results of the CCU self-test if it fails.

To run the CCU self test use the following key sequence. Insure the CCU has a loopback connector plugged into the differential port or else the diagnostic will record an error.

KEY SEQUENCE	DISPLAY	NOTE
<input type="checkbox"/> TR	READY (RUN)	
<input type="checkbox"/> AUX <input type="checkbox"/> 4	AUX = 4	No location address is required for an AUX function
<input type="checkbox"/> STEP	CCU SELF-TEST	
<input type="checkbox"/> STEP	PRESS ENTR TO START	
<input type="checkbox"/> ENTR	CCU TEST RUNNING	
	CCU TEST PASSED	OR CCU TEST FAILED displayed upon completion
<input type="checkbox"/> TR	READY (RUN)	

## 8.5 AUX 5 – READ/CLEAR ERROR TABLE

The PM550 contains a table where any detectable CCU errors are logged. The AUX 5 function allows reading or clearing of any or all errors in the table. See Figure 8.5A for the meaning of each error number. The first 16 errors (0-15) are fatal and will halt the CCU following completion of the current scan. The fault relay will drop out and all I/O (6MT + 7MT) will freeze in its last state until the error is cleared.

Neither ladder logic nor loop processing is executed. The system will not proceed to the RUN state with an outstanding fatal error. Non-fatal errors are considered warnings and do not directly effect system execution. No external signal is given to the user that a non-fatal error has occurred.

Use the following procedure to read and clear single errors in the error table

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	
<input type="button" value="AVA"/> <input type="button" value="5"/>	AUX = 5	No location address is required for any AUX function.
<input type="button" value="STEP"/>	READ OR CLR ERR TBL	
<input type="button" value="STEP"/> <input type="button" value="NO"/>	CLEAR ALL ERRORS ? NO	
<input type="button" value="STEP"/> <input type="button" value="6"/>	ERROR NO = 6	The number in parenthesis is the number of errors that have occurred. The maximum count is 255.
<input type="button" value="READ"/>	ERROR NO = 6 (27)	
<input type="button" value="STEP"/>	ERROR NO = 7 (5)	
<input type="button" value="ENTER"/>	ERROR NO = 7 (0)	Pressing <input type="button" value="ENTER"/> clears the displayed error number.
<input type="button" value="CLR"/>	READY (RUN)	<input type="button" value="STEP"/> and <input type="button" value="STEP"/> allowed

Use the following sequence to clear all errors.

KEY SEQUENCE	DISPLAY
<input type="button" value="CLR"/>	READY (RUN)
<input type="button" value="AUX"/> <input type="button" value="5"/>	AUX = 5
<input type="button" value="STEP"/>	READ OR CLR ERR TBL
<input type="button" value="SLP"/> <input type="button" value="YES"/>	CLEAR ALL ERRORS ?
<input type="button" value="STEP"/>	PRESS ENTR TO CLEAR
<input type="button" value="ENTR"/>	READY (RUN)

ERROR NUMBER	CLASS	DESCRIPTION
0	F	USER REQUESTED CCU DIAGNOSTIC FAILED. (SEE ERRORS 32-46 FOR SPECIFIC ERROR)
1	F	OPERATING SYSTEM ROM ERROR
2	F	OPERATING SYSTEM RAM ERROR
3	F	USER MEMORY L, V, C PARITY ERROR
4	F	OPERATING SYSTEM ERROR
5	F	OPERATING SYSTEM ERROR (MULTI PROCESSOR COMMUNICATION)
6	F	BATTERY LOW ERROR (MAY BE OVERRIDDEN WITH OPTIONAL STRAP)
7	F	POWER FAIL RECOVERY ERROR. SYSTEM DID NOT POWER DOWN CORRECTLY.
8-15	—	RESERVED
16	F	INVALID LOOP FLAGS DETECTED IN C LOOP TABLE.
17	F	LOOP TABLE ERROR. LOOP STRUCTURES AT C0-C15 ARE NOT PROPERLY INITIALIZED. IF LOOP PROCESSING IS NOT IN USE, C0-C15 MUST BE SET TO ZERO.
18-19	NF	RESERVED
20	NF	INVALID INPUT VIA PORT ZERO (0), ONE (1), OR TWO (2), INVALID ASCII CHARACTER
21	NF	RESERVED
22	NF	COMMUNICATION FAILURE ON PORT 0 (LAM/ TCAM, DIFF PORT) SYSTEM TRIED TO COM- MUNICATE 3 TIMES WITH THE LAM OR TCAM AND FAILED.*
23	NF	COMMUNICATION FAILURE ON PORT 1 (RWP OR DUMB TERMINAL) SYSTEM TRIED TO COM- MUNICATE 3 TIMES WITH RWP AND FAILED.*

\*The system continues to attempt to communicate with the terminals even though 3 failures were encountered.

F = Fatal Error

NF = Non Fatal Error

Figure 8.5A System Error Table by Number

ERROR NUMBER	CLASS	DESCRIPTION
22	NF	COMMUNICATION FAILURE ON PORT 2 (RWP OR DUMB TERMINAL) SYSTEM TRIED TO COMMUNICATE 3 TIMES WITH THE RWP AND FAILED.*
23	NF	OPERATING SYSTEM ERROR
24-29	NF	RESERVED
30	NF	LOOP CALCULATIONS ARE TAKING LONGER THAN 0.5 SECONDS
31	NF	LOOP MATH ERROR. FLOATING POINT UNDERFLOW OR SQUARE ROOT ERROR (VIA SPECIAL FUNCTION LOOP VARIABLE PREPROCESSING).
32	F	PARITY ERROR L
33	F	MEMORY FAILURE L
34	F	PARITY ERROR V
35	F	MEMORY FAILURE V
36	F	PARITY ERROR C
37	F	MEMORY FAILURE C
38	F	PARITY ERROR C' (C' IS EXPANDED C MEMORY)
39	F	MEMORY FAILURE C'
40	F	LOW LEVEL INTERRUPT FAILURE
41	F	IR MEMORY FAILURE
42	F	PORT 0 (DIFFERENTIAL LINE) LOOPBACK DIAGNOSTIC ERROR
43	F	CLOCK INTERRUPT FAILURE
44	F	SPECIAL FUNCTION INTERRUPT ERROR
45	F	LADDER LOGIC TEST PROGRAM FAILURE
46	F	6MT I/O CYCLE FAILURE

\*The system continues to attempt to communicate with the terminals even though 3 failures were encountered.

F = Fatal Error  
NF = Non Fatal Error

Figure 8 5A. System Error Table by Number (Cont )

## 8.6 AUX 6 - TAPE CONTROL

The PM550 is compatible with the STR-LINK II digital cassette recorder manufactured by Electronic Processors, Inc. (EPI), Englewood, Colorado. AUX 6 allows control of loading, verifying or dumping user memory to/from cassette tape

PHONE 303-761-8540

The appropriate connection of the STR-LINK II to the PM550 system is shown in Figure 8.6A. Read the STR-LINK II Operating Manual before proceeding to the key sequences below.

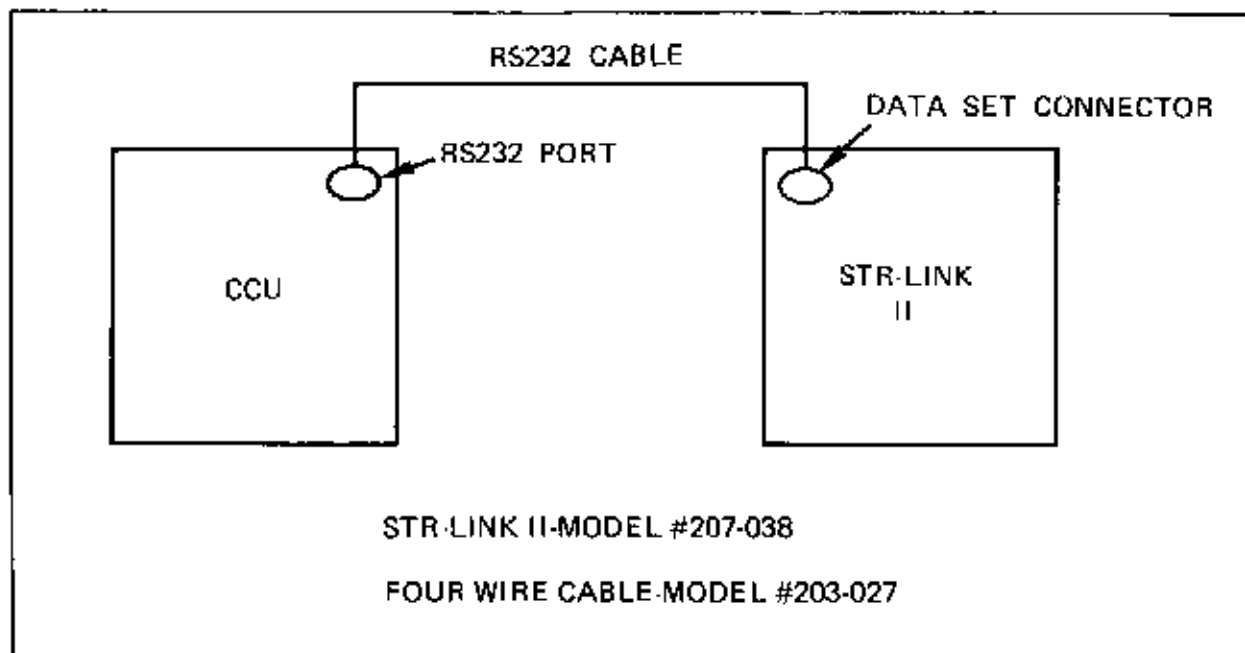


Figure 8.6A. STR-LINK II Hookup



The dump (record) user memory on tape uses the following procedures.

KFY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	
<input type="button" value="AUX"/> <input type="button" value="6"/>	AUX = 6	No location address is required for an AUX function
<input type="button" value="STOP"/>	TAPE CONTROL	
<input type="button" value="STEP"/> <input type="button" value="NO"/>	LOAD FROM TAPE ? NO	
<input type="button" value="STEP"/> <input type="button" value="YES"/>	DUMP TO TAPE ? YES	
<input type="button" value="STEP"/> <input type="button" value="YES"/>	DUMP L ? YES	
<input type="button" value="STEP"/> <input type="button" value="YES"/>	DUMP V ? YES	
<input type="button" value="STEP"/> <input type="button" value="YES"/>	DUMP C ? YES	Record V, C, and L user memory on tape.
<input type="button" value="STEP"/>	PRESS ENTER TO DUMP	Press stop on EPI, then push remote control button. Wait for tape to record and run.
<input type="button" value="ENTER"/>	*WAIT*	then press stop on EPI
	VERIFY ?	"VERIFY?" prompt or an error message will appear.
<input type="button" value="NO"/>	VERIFY / NO	
<input type="button" value="STEP"/>	READY (RUN)	

The information on the tape can be compared with that in memory using the verify feature. To verify, use the following sequence:

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	
<input type="button" value="AUX"/> <input type="button" value="6"/>	AUX = 6	
<input type="button" value="STEP"/>	TAPE CONTROL	
<input type="button" value="STOP"/> <input type="button" value="NO"/>	LOAD FROM TAPE ? NO	
<input type="button" value="STEP"/> <input type="button" value="NO"/>	DUMP TO TAPE ? NO	
<input type="button" value="STEP"/>	VERIFY ? YES	
<input type="button" value="YES"/>	PRESS ENTR TO VERIFY	
<input type="button" value="ENTR"/>	*WAIT*	Press stop on EPI, then remote control pushbutton. Wait for tape to stop, press stop again on EPI.
	READY (RUN)	

To load user memory from tape use the following

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	
<input type="button" value="AUX"/> <input type="button" value="6"/>	AUX = 6	
<input type="button" value="STEP"/>	TAPE CONTROL	
<input type="button" value="STEP"/>	LOAD FROM TAPE ?	
<input type="button" value="YES"/>	PRESS ENTR TO LOAD	
<input type="button" value="ENTR"/>	*WAIT*	Press stop on EPI, then remote control push button. Wait for tape to stop, then press stop on EPI. An error message will be displayed if the load fails.
	DUMP TO TAPE ?	
<input type="button" value="NO"/>	DUMP TO TAPE ? NO	
<input type="button" value="STOP"/> <input type="button" value="NO"/>	VERIFY ? NO	
<input type="button" value="STEP"/>	READY (RUN)	

## AUX 6 ERROR MESSAGE DISPLAY

If an error is encountered during a load, dump or verify the following is displayed

ERR STAT = LL VV CC

where LL represents L memory, VV V memory and CC C memory. The range of values for LL, VV and CC are as follows:

- 01 — Verify error, check sum error, invalid ASCII character or invalid start address.
- 02 — Read after write error
- 04 — Parity or framing error
- 08 — Invalid record length

The errors are cumulative and displayed in hex, e.g.

ERR STAT = 01 0C 02 represents

- L — error 1
- V — errors 4 and 8
- C — error 2

## 8.7 AUX 7 - ENTERING ASCII MESSAGE

The AUX 7 function allows the entry of ASCII messages into user memory. ASCII characters are entered via a decimal code given in Figure 8.7A. Special Function 11 commands the ASCII message in ladder logic.

The first step in entering an ASCII message is to look up the code for each character in Figure 8.7A. The message for the programming example below is "COUNTER 1 = (V18) CR, LF" ((V18) means the value of V18). Figure 8.7A shows how this message is coded.

ASCII characters are stored with two characters per memory location. Addresses for values to be printed take one location and one location is required for the terminator. Therefore, the memory required is:

$$\text{Memory} = \frac{\text{number of chars.} + 1}{2} + \text{number of values rounded up to the nearest whole number}$$

See Figure 8.7C for an example

ASCII Character	Decimal Code	ASCII Character	Decimal Code
NULL	0	@	64
SOH	1	A	65
STX	2	B	66
ETX	3	C	67
END	4	D	68
ACK	5	E	69
BEL	6	F	70
BS	7	G	71
HT	8	H	72
LF	9	I	73
VT	10	J	74
FF	11	K	75
CR	12	L	76
SO	13	M	77
SI	14	N	78
OLE	15	O	79
DC1	16	P	80
DC2	17	Q	81
DC3	18	R	82
DC4	19	S	83
NAK	20	T	84
SYN	21	U	85
ETB	22	V	86
CAN	23	W	87
EM	24	X	88
SUB	25	Y	89
ESC	26	Z	90
FS	27		91
GS	28	Back Slash	92
RS	29		93
US	30		94
SPACE	31		95
!	32		96
"	33		97
#	34		98
\$	35		99
%	36		100
&	37		101
'	38		102
(	39		103
)	40		104
*	41		105
+	42		106
,	43		107
-	44		108
.	45		109
/	46		110
0	47		111
1	48		112
2	49		113
3	50		114
4	51		115
5	52		116
6	53		117
7	54		118
8	55		119
9	56		120
:	57		121
;	58		122
<	59		123
=	60		124
>	61		125
?	62		126
	63		127

Figure 8.7A. ASCII Character To Decimal Code Chart For Aux 7

<u>CHARACTER</u>	<u>CODE</u>	
C	67	
O	79	
U	85	
N	78	
T	84	
E	69	
R	82	
SP	32	(SP) is one space
1	49	
=	61	
(V18)	V18	This prints the value of V18
CR	13	A CR causes the carriage (or print read) to return.
LF	10	A LF causes the paper to advance one line preparation for the next line of print

Figure 8 7B. Sample Message Coding

C100	C	O
C101	U	N
C102	T	E
C103	R	SP
C104	1	=
C105		V18
C106	CR	CF
C107	TERMINATOR	

Figure 8 7C. Memory Locations for AUX 7 Example

The required steps for the message in Figure 8 7B follow

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="CLR"/>	READY (RUN)	
<input type="button" value="AUX"/> <input type="button" value="7"/>	AUX = 7	
<input type="button" value="STEP"/>	ASCII MESSAGE	
<input type="button" value="STEP"/> <input type="button" value="C"/> <input type="button" value="1"/> <input type="button" value="0"/> <input type="button" value="0"/>	START: C100	
<input type="button" value="STEP"/> <input type="button" value="6"/> <input type="button" value="7"/>	C100L = 67	We are programming the left 8 bits of C100.
<input type="button" value="ENTR"/>	C100L = 67 (C)	
<input type="button" value="STEP"/> <input type="button" value="7"/> <input type="button" value="9"/>	C100R = 79	We are programming the right 8 bits of C100.
<input type="button" value="ENTR"/>	C100R = 79 (0)	
<input type="button" value="STEP"/> <input type="button" value="8"/> <input type="button" value="5"/>	C101L = 85	
<input type="button" value="ENTR"/>	C101L = 85 (U)	
<input type="button" value="STEP"/> <input type="button" value="7"/> <input type="button" value="8"/>	C101R = 78	
<input type="button" value="ENTR"/>	C101R = 78 (N)	
<input type="button" value="STEP"/> <input type="button" value="8"/> <input type="button" value="4"/>	C102L = 84	
<input type="button" value="ENTR"/>	C102L = 84 (1)	
<input type="button" value="STEP"/> <input type="button" value="6"/> <input type="button" value="9"/>	C102R = 69	
<input type="button" value="ENTR"/>	C102R = 69 (F)	
<input type="button" value="STEP"/> <input type="button" value="8"/> <input type="button" value="2"/>	C103L = 82	
<input type="button" value="ENTR"/>	C103L = 82 (R)	
<input type="button" value="STEP"/> <input type="button" value="3"/> <input type="button" value="2"/>	C103R = 32	

KEY SEQUENCE	DISPLAY	NOTE
<input type="button" value="ENTR"/>	C103R = 32 ( )	
<input type="button" value="STEP"/> <input type="button" value="4"/> <input type="button" value="9"/>	C104L = 49	
<input type="button" value="ENTR"/>	C104L = 49 (1)	
<input type="button" value="STEP"/> <input type="button" value="6"/> <input type="button" value="1"/>	C104R = 61	
<input type="button" value="ENTR"/>	C104R = 61 (=)	
<input type="button" value="STEP"/> <input type="button" value="V"/> <input type="button" value="1"/> <input type="button" value="8"/>	C105L = V18	If a storage location is specified within a message its value will be printed. The storage location takes one memory location
<input type="button" value="ENTR"/>	C105L = V18	
<input type="button" value="STEP"/> <input type="button" value="1"/> <input type="button" value="3"/>	C106L = 13	
<input type="button" value="ENTR"/>	C106L = 13 ( )	Any unprintable character will be shown as a blank.
<input type="button" value="STEP"/> <input type="button" value="1"/> <input type="button" value="0"/>	C106R = 10	
<input type="button" value="ENTR"/>	C106R = 10 ( )	
<input type="button" value="STEP"/> <input type="button" value="9"/> <input type="button" value="7"/> <input type="button" value="9"/>	C107L = 999	Any code > 128 will terminate AUX ?
<input type="button" value="ENTR"/>	END: C108	
<input type="button" value="STOP"/>	READY (RUN)	

To read back an ASCII message follow the same sequence as required to write the message but press  instead of  after entering the START location. Then use  to examine the message.



## APPENDIX A

### DOCUMENTATION

- A-I LADDER ELEMENT (L MEMORY) STORAGE RECORD
- A-II DISCRETE INPUT/OUTPUT (6MT) RECORD FORMS
- A-III AUXILIARY (ANALOG/PARALLEL) INPUT/OUTPUT (7MT) RECORD
- A-IV DATA MEMORY STORAGE RECORD
- A-V LOOP SPECIFICATION SHEET

LADDER ELEMENT (L MEMORY) STORAGE RECORD

PROGRAM TITLE \_\_\_\_\_

PROGRAMMER \_\_\_\_\_

DATE \_\_\_\_\_ PAGE \_\_\_\_ OF \_\_\_\_

[illegible]

CONTROL PRODUCTS DIVISION  
JOHNSON CITY, TENNESSEE 37601 • TELEPHONE: 615 926 1167

[illegible]

10

\_\_\_\_\_

**AUXILIARY INPUT/OUTPUT RECORD FORM**  
**(7MT) RECORD**

[illegible]

[illegible]

# LOOP SPECIFICATION SHEET

Loop Description:

PM550 Loop Number\_\_\_\_\_

## MEMORY ALLOCATION

Tuning Constants in\_\_\_\_(C or V)

	Beginning Address	Tune C	Tune V	Ending Address
Constant Table	_____	_____	_____	_____
Variable Table	_____	_____	_____	_____

Are loop flags for alarms and mode switching allocated in the image register?\_\_\_\_\_ If yes, give beginning address: \_\_\_\_\_

## PROCESS VARIABLE

20% Offset?\_\_\_\_\_

Special calculation?\_\_\_\_\_

Low Range = \_\_\_\_\_

Engr Units\_\_\_\_\_

Address: \_\_\_\_\_

Square root?\_\_\_\_\_

If yes, give address: \_\_\_\_\_

High Range = \_\_\_\_\_

Transmitter\_\_\_\_\_

## CONTROL CALCULATIONS

Remote Set Point?\_\_\_\_\_

Special Calculation?\_\_\_\_\_

LOCK - Set Point?\_\_\_\_\_

Error Squared?\_\_\_\_\_

Gain = \_\_\_\_\_

Rate = \_\_\_\_\_

Output Address: \_\_\_\_\_

Sample Time = \_\_\_\_\_

If yes, give address: \_\_\_\_\_

If yes, give address: \_\_\_\_\_

Auto/Manual?\_\_\_\_\_ Cascade?\_\_\_\_\_

Error Deadband?\_\_\_\_\_

Reset Time = \_\_\_\_\_

Reverse Acting?\_\_\_\_\_

20% Offset?\_\_\_\_\_

## ALARMS

Process Variable

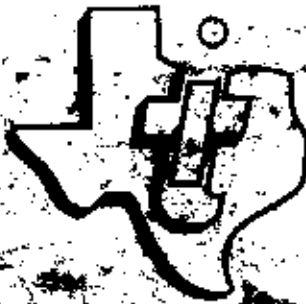
Low = \_\_\_\_\_

Deviation

Yellow = \_\_\_\_\_

High = \_\_\_\_\_

Orange = \_\_\_\_\_



For information write to:  
**INSTRUMENTS INCORPORATED**  
Industrial Controls Marketing  
Memphis, Tennessee 37203

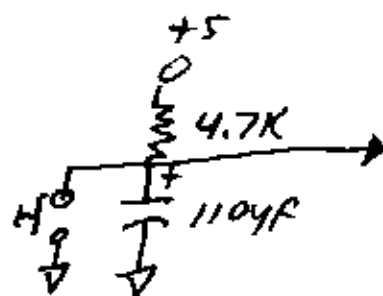


## PM550 NOTES

Reset Line →

Jumper

to ground  
For CGV good  
Light



Battery disconnected gives a fault solder Datten to terminal

Jumper 9 pin 1 to 5 & 7 to 8 else will fault

TURN ON +5 volt LAST

9.1V is for I/O RACK ONLY

U13 74128 causes Flaky I/O on I/R card