GE Fanuc Automation

Programmable Control Products

Series Six™ Programmable Controllers

Programming Manual

GEK-25362

May 1982

Warnings, Cautions, and Notes as Used in this Publication

Warning

Warning notices are used in this publication to emphasize that hazardous voltages, currents, temperatures, or other conditions that could cause personal injury exist in this equipment or may be associated with its use.

In situations where inattention could cause either personal injury or damage to equipment, a Warning notice is used.

Caution

Caution notices are used where equipment might be damaged if care is not taken.

Note

Notes merely call attention to information that is especially significant to understanding and operating the equipment.

This document is based on information available at the time of its publication. While efforts have been made to be accurate, the information contained herein does not purport to cover all details or variations in hardware or software, nor to provide for every possible contingency in connection with installation, operation, or maintenance. Features may be described herein which are not present in all hardware and software systems. GE Fanuc Automation assumes no obligation of notice to holders of this document with respect to changes subsequently made.

GE Fanuc Automation makes no representation or warranty, expressed, implied, or statutory with respect to, and assumes no responsibility for the accuracy, completeness, sufficiency, or usefulness of the information contained herein. No warranties of merchantability or fitness for purpose shall apply.

The following are trademarks of GE Fanuc Automation North America, Inc.

Alarm Master	GEnet	PowerMotion	Series One
CIMPLICITY	Genius	ProLoop	Series Six
CIMPLICITY PowerTRAC	Genius PowerTRAC	PROMACRO	Series Three
CIMPLICITY 90–ADS	Helpmate	Series Five	VuMaster
CIMSTAR	Logicmaster	Series 90	Workmaster
Field Control	Modelmaster		

©Copyright 1996 GE Fanuc Automation North America, Inc.
All Rights Reserved

Preface

The intent of this manual is to provide the reader with the information needed to program a Series Six Programmable Controller. A general description of the Program Development Terminal (PDT) is given in Chapter 1. Chapter 2 is a guide to the installation and initial check of the PDT prior to entering any programs.

Chapter 3 introduces the reader to each of the keys on the keyboard, describing their function in detail. In this chapter each of the Basic and Extended mnemonic functions are described and tables are provided listing valid ranges for references and displayed data.

Chapter 4 describes ladder diagram programming. The Supervisor and Scratch Pad displays are described along with the Status and Register tables. Entry of each of the relay functions is presented in a step-by-step sequence of keystroke entries.

How to enter each of the Basic mnemonic functions is shown in a series of step-by-step keystroke sequences in Chapter 5. At the end of this chapter is a sample program using several of the Basic functions.

Chapter 6 provides an example of how to enter each of the Extended mnemonic functions, again using the step-by-step sequence. For additional information and samples of programs using the Extended functions, refer to the SERIES SIX APPLICATIONS GUIDE, GEK-25365.

System Check

After a program has been entered into the Program Development Terminal and transferred to the CPU, it is recommended that the program be thoroughly checked before beginning on-line operation to control a machine or process. There are several ways this can be done. One way is to put the CPU in the RUN mode with outputs disabled. All inputs to the system can then be cycled and the program monitored on the PDT for proper operation of each line of logic. Another method is to use the override function. By overriding individual output coils, the logic controlling those coils can be observed on the PDT while in the monitor mode, thereby ensuring proper control of each output.

Further, it is recommended that the system be thoroughly exercised through all possible modes of operation in order to insure proper and expected response of the program to all configurations of inputs.

It is extremely important to observe all possible precautions before beginning operation of a system in order to preclude the possibility of damage to equipment or personal injury. Use of each of the programming functions should be throughly understood. Since it is impossible to anticipate all applications in which a Series Six programmable controller may be used, the user must assume responsibility for proper use of the programming capabilities of the system.

Table of Contents

CHAPTER 1—General Description

Program Development Terminal Description 1.1
Enclosure 1.2
Environmental and Electrical Characteristics 1.3
Cable Storage 1.4
External Device Connection 1.5
Cable Connections 1.6
Keyboard 1.7
CRT Display 1.9
Internal Tape Unit 1.9
Cooling Fan 1.10
Switches 1.11
System Connection 1.12

CHAPTER 2—Installation

Unpacking 2.1
Inspection 2.1
Cable Connection 2.1
Printer Cable 2.3
External Tape Unit Cable 2.4
Composite Video Monitor Cable 2.4
PDT-To-CPU Cable 2.4
AC Line Cord 2.4
Start-Up Instructions 2.5

CHAPTER 3—Program Development Terminal Controls and Functions

Introduction 3.1
AC Power Switch 3.1
Mode Selection Keyswitch 3.1
  On-Line Mode 3.2
  Monitor Mode 3.3
  Off-Line Mode 3.3
Keyboard Operation 3.3
  Mode/Control Group 3.4
  Numeric Group 3.14
  Operand Group 3.15
  Relay Group 3.17
  Basic Mnemonic Group 3.19
  Power Flow 3.20
  Extended Mnemonic Group 3.25

CHAPTER 4—Ladder Diagram Programming

Introduction to Programming 4.1
Ladder Diagram Display 4.4
Supervisor Display 4.10
Display Requested Status or Register Table 4.12
Scratch Pad Display 4.19
Building Ladder Diagrams 4.29

CHAPTER 5—Entering Basic Mnemonic Functions

Introduction 5.1
Shift 5.1
I/O R 5.3
R I/O 5.4
BIBCD 5.5
BCDBI 5.6
ADD 5.7
SUB 5.8
COMPR 5.9
MCR 5.10
SKIP 5.11
ENDSW 5.12
NOOP 5.12
SCREQ 5.13
DPREQ 5.14
Entering a Program 5.15

CHAPTER 6—Entering Extended Mnemonic Functions

Introduction 6.1
Data Move Group 6.1
Move 6.2
Move Right 8 ..... 6.3
Move Left 8 ..... 6.4
Block Move 6.5
Signed Arithmetic Group 6.6
DP Add 6.7
DP Sub 6.8
ADDX 6.9
SUBX 6.10
MPY 6.11
DVD 6.12
Greater Than 6.13
Table Move Group 6.13
Table-to-Dest 6.14
SRC-to-Table 6.15
Move Table 6.16
List Group 6.18
Add-to-Top 6.18
Rem-fm-Bot 6.20
Rem-fm-Top 6.21
Sort 6.22
Matrix Group 6.24
AND 6.25
IOR 6.26
EOR 6.27

INV 6.28
Compare 6.29
Bit Matrix Group 6.31
Bit Set 6.32
Bit Clear 6.33
Shift Rt 6.34
Shift Left 6.35
Control Group 6.36
Do Sub 6.36
Return 6.38
Suspend I/O 6.39
Do I/O 6.40
Status 6.41

Appendix A—Glossary of Terms

CHAPTER I

General Description

The Program Development Terminal (PDT) provides the user of a Series Six Programmable Controller with the means of creating, modifying and monitoring the programs necessary for control of a machine or process.

This chapter provides a description of the hardware features of the Program Development Terminal and a description of optional devices which may be used with the PDT. Guidelines for connection to a system are also provided.

PROGRAM DEVELOPMENT TERMINAL DESCRIPTION

The PDT (Figure 1) is the physical device used for entering, editing (modifying), and monitoring programs while the programmable controller is operational.

Programs are entered in a free-format ladder diagram form. Individual inputs or outputs may be monitored by observation of the status tables. Data stored in registers for manipulation may also be monitored by observation of the register display.

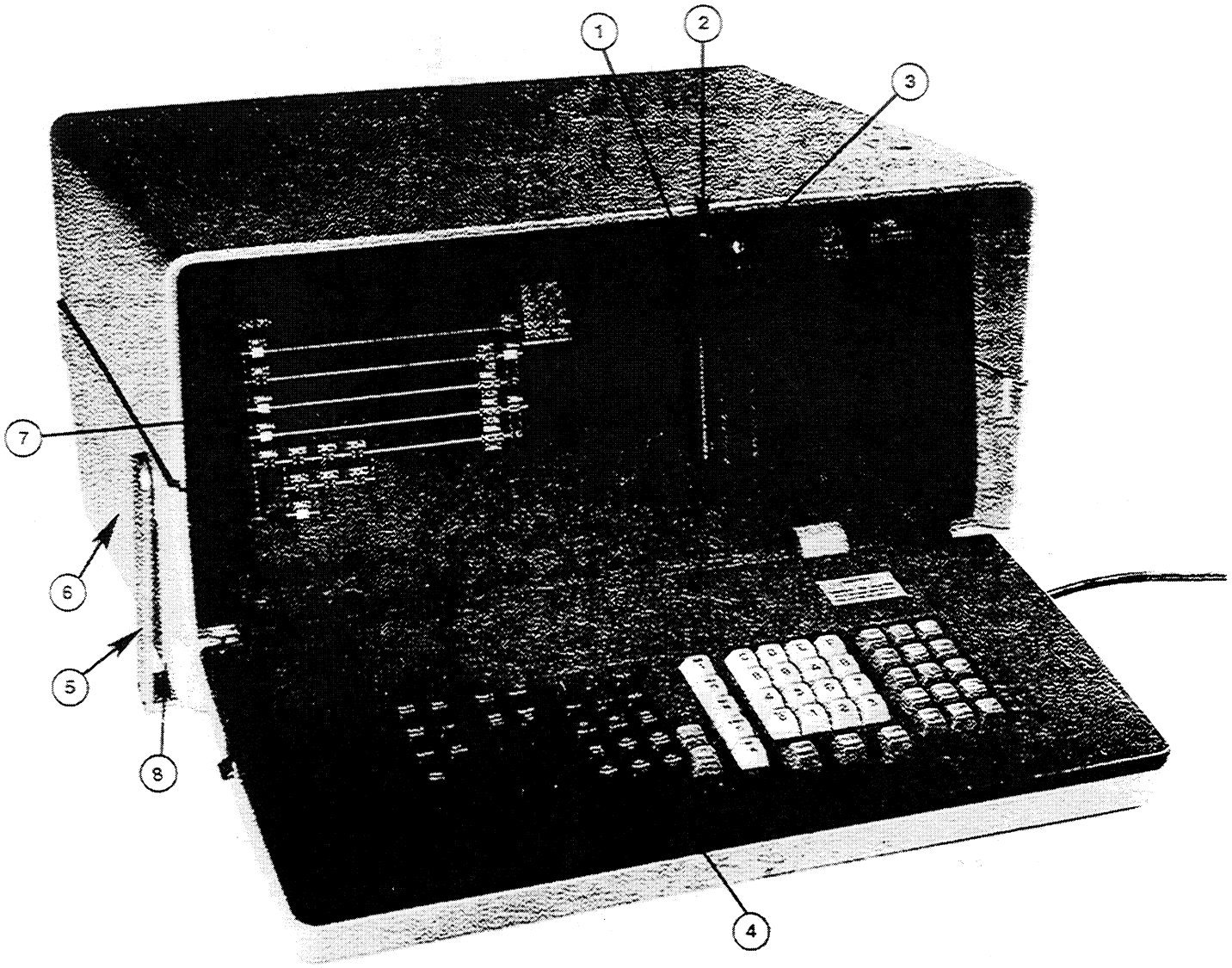
The ON-LINE mode allows a real-time observation of the power flow of any rung of relay logic. While in this mode certain one-word changes can be made to the existing program without affecting the operation of the scan. Multi-word changes, up to an entire 32K program, can also be accomplished with outputs suspended.

The MONITOR mode allows the PDT to display any logic rung and its power flow status. The content of any register, the status of all inputs, or outputs can also be monitored. However, no alteration of the logic, override state, register content, or other changes to the memory content of the CPU can be made in this mode. MONITOR mode is invaluable in the troubleshooting of both the Programmable Controller as well as the entire control system.

The OFF-LINE mode is a convenient feature in that it allows the PDT to be programmed without being connected to the CPU. This allows a program to be developed in a location other than where the CPU is located. When the program has been verified as being correct, it can be copied to tape and shipped or transmitted to the location of the CPU for loading into the CPU memory.

The PDT is a powerful tool to be used as a part of a Series Six system. Use of the PDT can be learned easily by reading the introductory material and following the step-by-step guide to programming provided in this manual.

The PDT is a transportable microprocessor-based terminal utilizing a keyboard for entry of data, a CRT (Cathode Ray Tube) which is a 12 inch (305 mm) black and white display device and an optional tape unit for recording (storing) and loading programs.



1. Tape Unit (Optional)
2. Power Switch
3. Mode Selection Keyswitch
4. Keyboard
5. Handle
6. Foam Enclosure
7. CRT Display
8. Cable storage and serial device connectors

FIGURE 1
Program Development Terminal

ENCLOSURE

The enclosure (Figure 2) for the Program Development Terminal is made of structural foam which provides strength and durability. Dimensions of the unit with the keyboard in the closed position are 12.5 × 21.5 × 16.5 inches (317.5 × 546.1 × 419.1 mm). Weight of the unit is 57 pounds (26 Kg).

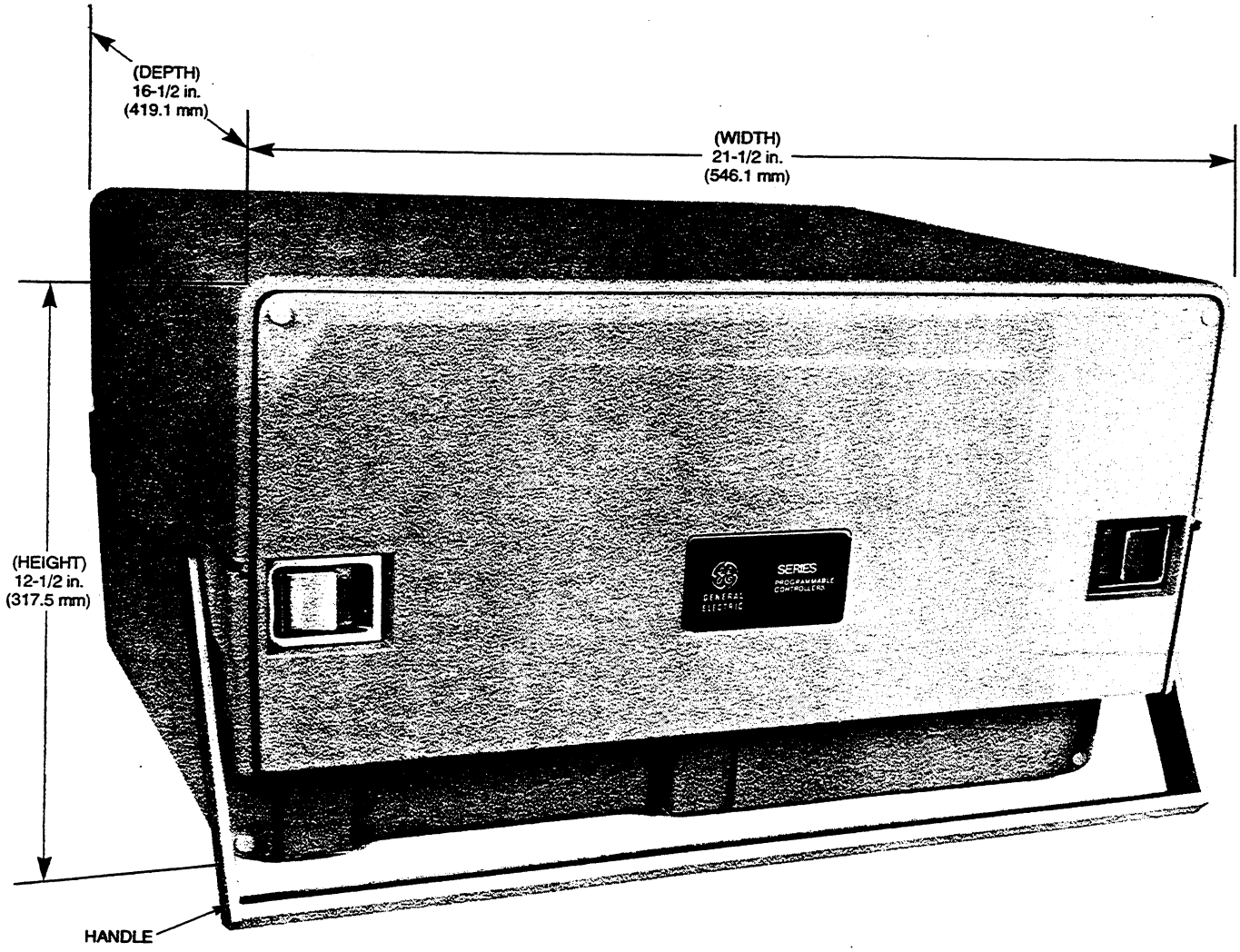


FIGURE 2
Program Development Terminal Enclosure

ENVIRONMENTAL AND ELECTRICAL CHARACTERISTICS

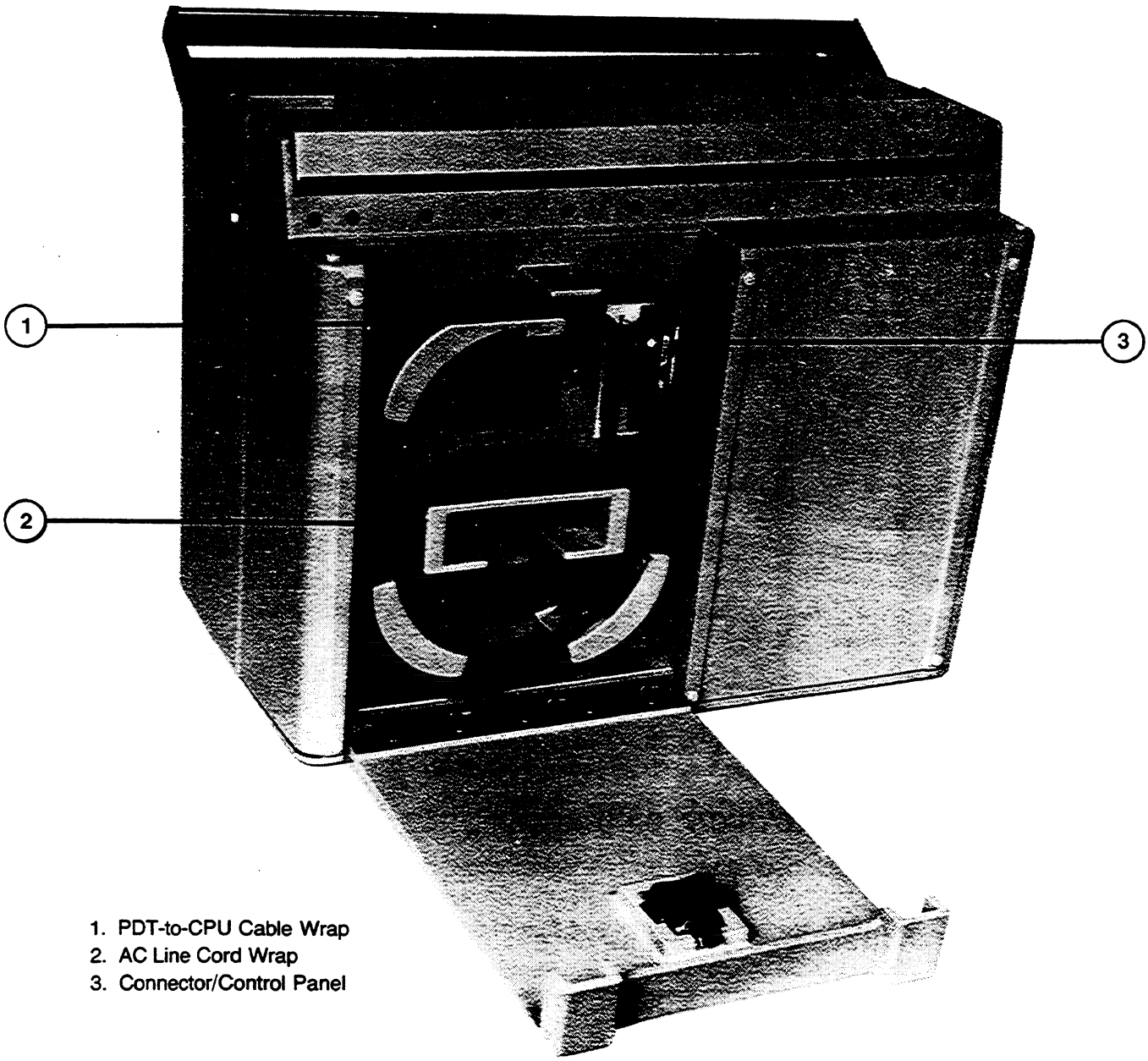
A 3-wire grounding plug is provided at the end of a 6 foot line cord which terminates inside of the PDT.

TABLE 1—Environmental and Electrical Characteristics

Temperature:	
Operation	0 to 45°C (32° to 113°F)
Storage	−20° to +65°C (−4° to +149°F)
Humidity	20 to 80% non-condensing
Required Power:	
Voltage	115V AC (Range of 95-130V AC)
Frequency	47-63 Hz
Power Consumption	180 VA maximum

CABLE STORAGE

A cable storage compartment is located underneath the PDT on the left side as viewed from the front of the PDT. (See Figure 3.) The compartment is accessed by lowering a hinged panel. There are 2 cable wraps molded into the compartment which provide convenient storage for the AC power cord and the PDT-to-CPU cable. The AC power cord is wrapped around the inner rectangular form and the PDT-to-CPU cable is wrapped on the outer form.

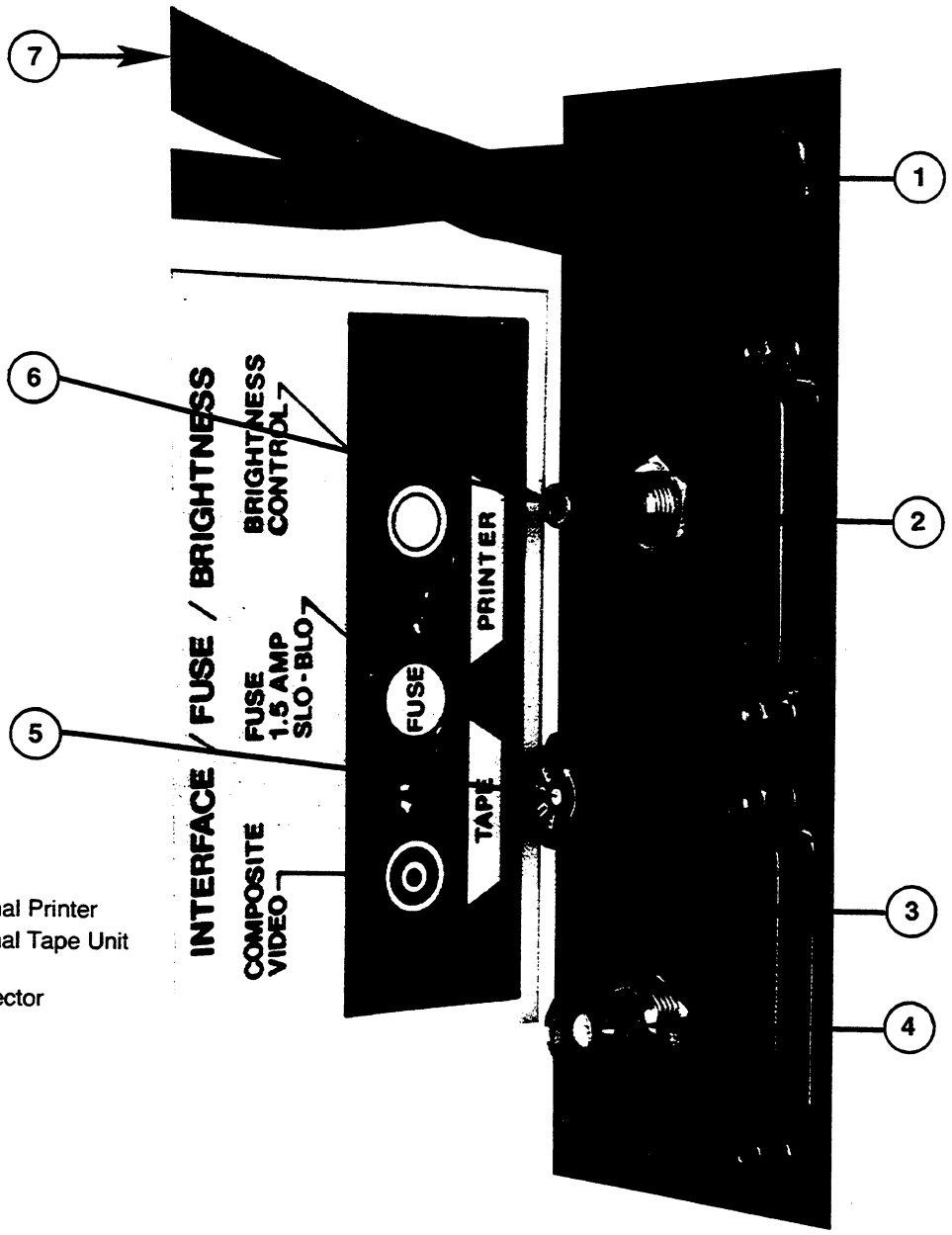


1. PDT-to-CPU Cable Wrap
2. AC Line Cord Wrap
3. Connector/Control Panel

FIGURE 3
Program Development Terminal
Cable Storage Compartment

EXTERNAL DEVICE CONNECTION

In addition to cable storage the compartment contains a panel with connectors which allow the connection of peripheral devices for use with the PDT. Figure 4 shows the location of these connectors along with other hardware items described in this section.



1. AC Power Cord
2. Connector to External Printer
3. Connector to External Tape Unit
4. Composite Video Monitor BNC Connector
5. Fuse, 1.5A, Slo-Blo
6. Brightness Control
7. PDT-to-CPU Cable

FIGURE 4
PDT Connector Panel

EXTERNAL TAPE UNITS

The bottom connector is a 25 pin D-type female connector allowing connection to an external magnetic tape unit for recording from, or loading programs into, the PDT. This connection is RS-232C compatible. Tape units recommended for use with the PDT are the STR LINK IIA which uses a Philips Data Cassette or the STR LINK III which uses the 3M DC100A Data Cartridge. Tapes generated on the STR LINK III are completely compatible with those generated on the optional built-in internal tape unit.

PRINTER

The top connector is a 25 pin D-type female connector which allows connection to an RS-232C compatible printer such as the Terminet® family. The use of a Terminet printer gives the capability of obtaining a listing of a complete ladder diagram program, selected rungs of a ladder diagram or a cross reference table printout.

When connecting a printer to this port the baud rate must be set the same as the baud rate selected by the item BAUD RATE on the Supervisor display. Baud rates selectable on the PDT are 110, 300, 1200, 2400, 4800, or 9600 baud.

EXTERNAL VIDEO CONNECTOR

A composite video output connector is provided for connection to an external digital video monitor. Connection to a video monitor requires the use of BNC connectors and 75 ohm cable. The composite video output provides a video signal with the required sync pulses to allow a display on a monitor specifically designed for the display of underscan digital data.

BRIGHTNESS CONTROL

The brightness control for the CRT display is located adjacent to the top (printer) connector (refer to Figure 4). This control provides an adjustment for the brightness of the CRT display to allow a clearly viewable display under a variety of ambient lighting conditions.

The brightness control is factory set for average lighting conditions and normally does not require adjustment. If adjustment is required, turning the control clockwise increases the brightness and turning the control counter-clockwise will decrease the brightness of the CRT display.

FUSE HOLDER

A fuse for the PDT is located in the fuse holder which is panel mounted adjacent to the connector for the external tape unit. The fuse is rated at 1.5 amps and is a slo-blo type.

WARNING

Before checking or replacing the fuse, unplug the PDT line cord from the AC power source. With the line cord plugged in, 115V AC is present on the fuse which could cause personal injury.

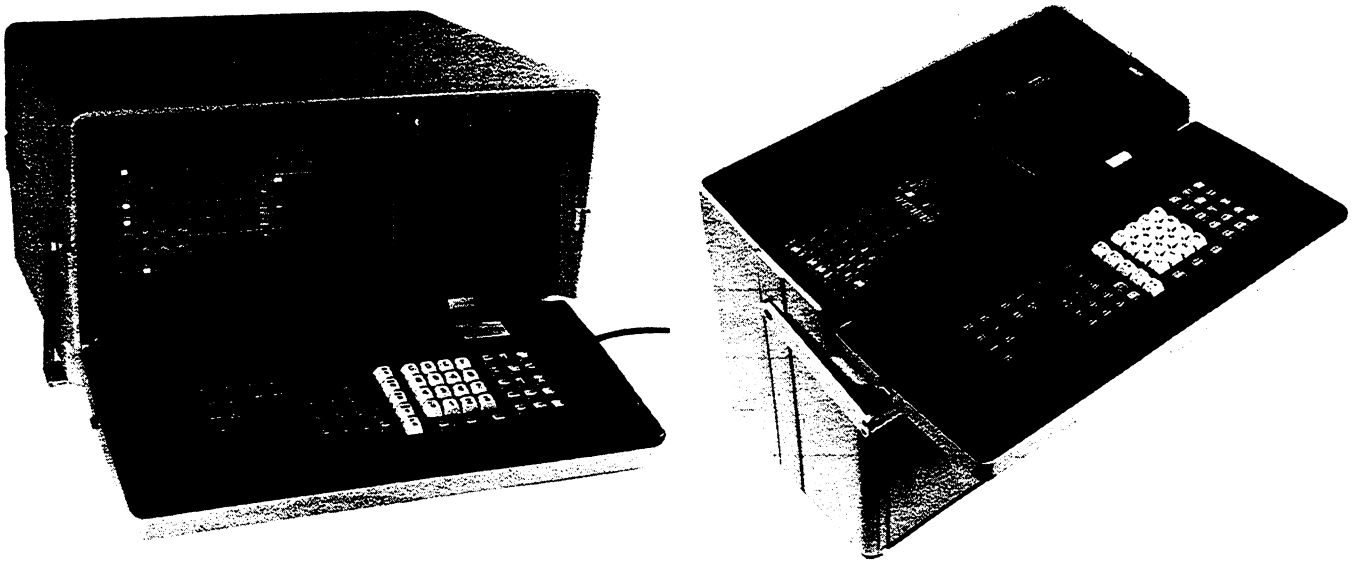
CABLE CONNECTIONS

Two cables are accessed and stored in the cable storage compartment. The AC power cable is 6 feet long and is routed through the panel to the power supply. For storage purposes the cable is wrapped around the inner rectangular molded cable form.

The PDT-to-CPU cable is 10 feet long. Inside the PDT the cable is routed through the panel and connected to the PDT electronics. The opposite end has a 37 pin D-type male connector. This is connected to the top connector of the I/O Control module in a CPU or to the bottom connector of an I/O Receiver in a CPU station or a Local I/O station. For storage purposes this cable is wrapped around the outer molded form in the storage compartment.

KEYBOARD

The Program Development Terminal uses a keyboard assembly which folds out from the enclosure. The keyboard can be used in 2 positions as shown in Figures 5A and 5B



A
PDT, Horizontal With Keyboard in a normal operation position.

B
PDT, Vertical With Keyboard folded out.

FIGURE 5
Program Development Terminal Keyboard Positions

Figure 5A shows the Program Development Terminal resting on its base in a horizontal position. This is the position normally used when programming from a sitting position and is the position most comfortable for data entry.

With the Program Development Terminal resting on its back, the keyboard opens to a position which allows programming from an upright position. This position is useful when troubleshooting an operating system or when the PDT is connected to an I/O rack in a system for implementing on-line changes.

In both positions the keyboard extends 8.25 inches (209.55 mm) from the PDT. The keyboard folds up into the terminal for storage and transportation, thus protecting it in a secure, rugged enclosure.

A molded ridge around the edge of the keyboard protects against damage to the keys if the handle should be in the way as the keyboard is being closed.

The pushbuttons are reliable solid-state modules with good tactile feel and travel which helps greatly in speed and ease of data entry and in reduction of operator fatigue.

KEYBOARD ORGANIZATION

The keys are grouped and color coded by function. Figure 6 shows a layout of the keyboard which illustrates the key groups and color codes. The color-coded keys provide a quick reference for the operator when entering programs.

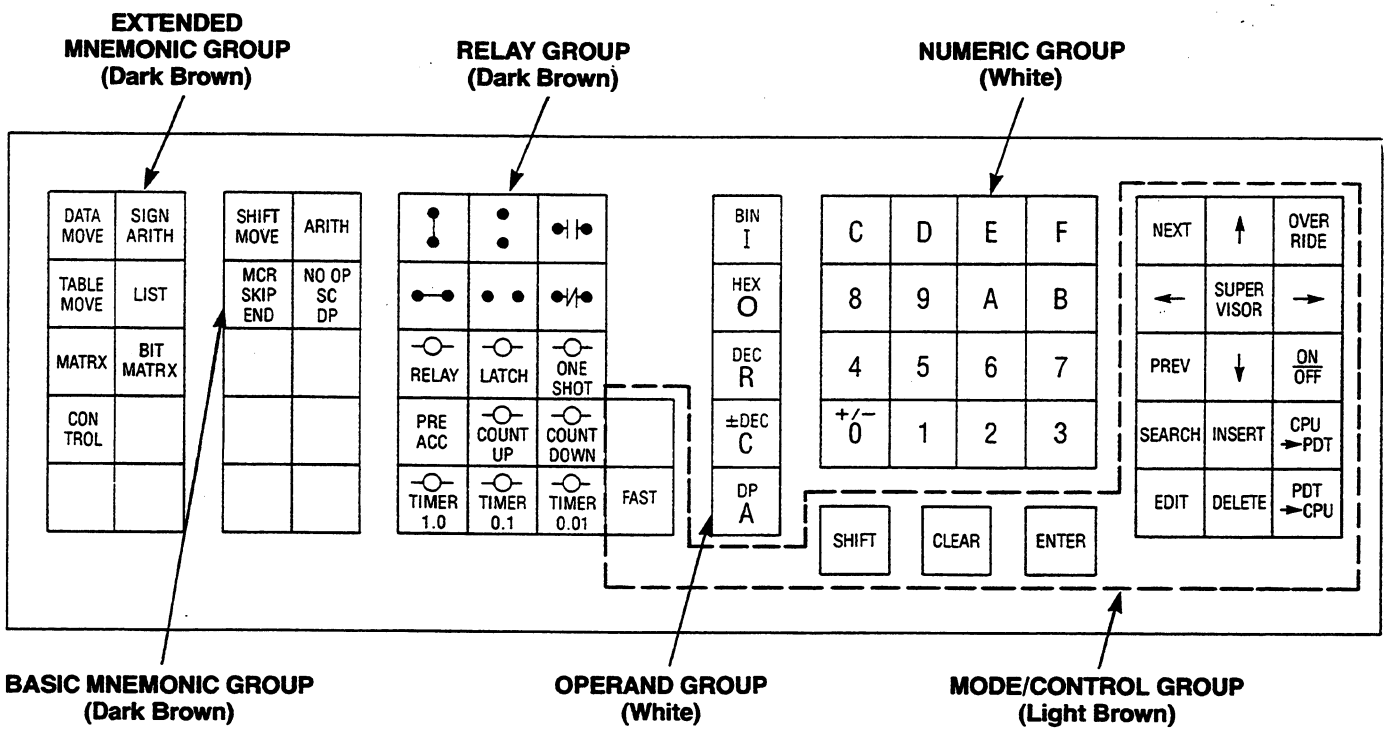


FIGURE 6
Keyboard Layout

The key tops are two-shot injection molded. By use of this method of molding the legend color goes all the way through the key and will not rub off.

The keyboard groups and a brief description of their functions are shown below in Table 2.

TABLE 2—Keyboard Groups

GROUP	FUNCTION
Extended Mnemonic	Enters mnemonic functions from the extended function set.
Basic Mnemonic	Enters mnemonic functions from the basic function set.
Relay	Enters contacts, relays, counters, timers, latches and one-shots.
Operand	Used to specify input and output status table and register memory references, constants and auxiliary references and base changes.
Numeric	Hexadecimal keypad used for entry of numeric data.
Mode/Control	Used for data transfers, screen control and mode control functions.

CRT DISPLAY

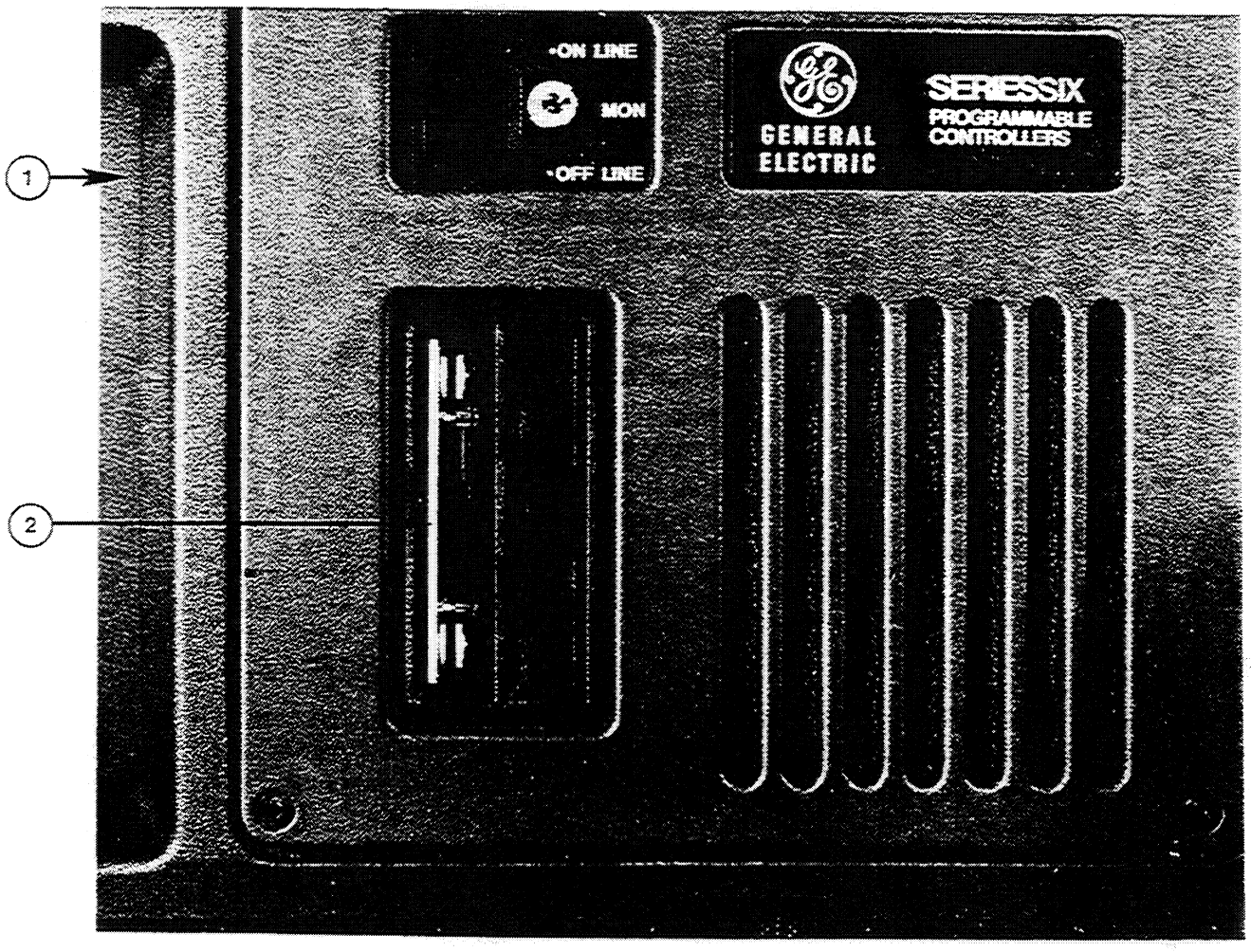
The Cathode Ray Tube (CRT) provides a 12 inch (305 mm) diagonal measure screen for displaying the user's program as well as various system operating parameters. The CRT has an anti-glare faceplate which allows good visibility under adverse lighting conditions. Characters are displayed on the screen in a 5 × 7 dot matrix.

The brightness control located in the cable storage area allows the brightness of the display on the CRT to be adjusted to provide a clearly visible display under various ambient lighting conditions.

INTERNAL TAPE UNIT

The Program Development Terminal has an optional built-in tape unit which allows the user to load programs from tape into the PDT or to record programs entered into the PDT onto tape for storage of those programs. The user can also compare the information on tape to the program contained in the PDT memory.

If provided, the tape unit is physically located to the right of the CRT as shown in Figure 7.



1. CRT
2. Built-In-Tape Unit (Optional)

FIGURE 7
Optional Built-In Tape Unit

The tape used with the built-in tape unit is the 3M DC100A minicartridge. The minicartridge snaps into place for use and is removed by depressing a push button located to the right of the tape unit.

Data is transferred between the PDT and the tape unit in a parallel mode of operation. This mode of operation allows rapid data transfer; a 32K (maximum user memory) program can be transferred to tape or from tape in less than 3 minutes using a single cartridge.

COOLING FAN

A grill is located to the right of the internal tape unit. (Refer to Figure 7.) A fan is located behind the grill to provide a flow of air which helps to cool the internal electronics of the Program Development Terminal. The PDT is designed to operate in ambient air temperatures of 0°C to 45°C.

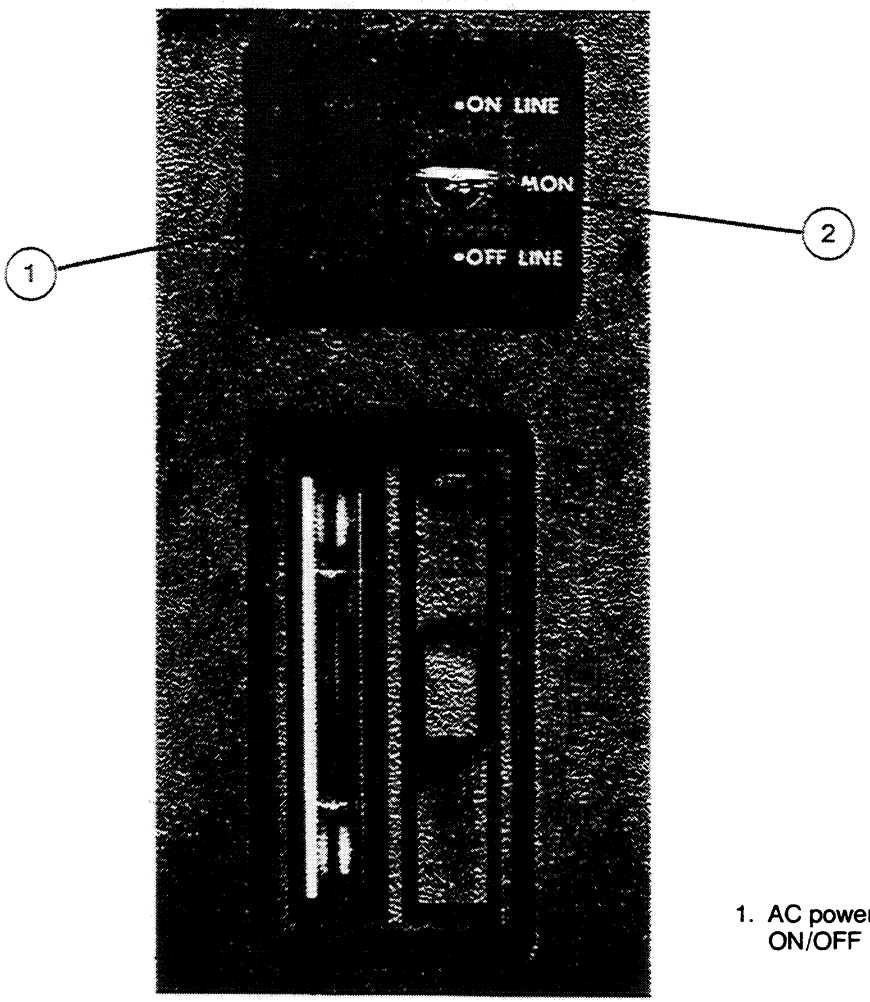
CAUTION

Do not close the keyboard with the PDT running. The airflow will be blocked and could possibly cause the PDT to operate improperly.

The air filter located in front of the fan should be checked for dust build-up and cleaned periodically. For more detailed instructions on cleaning the filter refer to Chapter 4 of GEK-25361, INSTALLATION AND MAINTENANCE MANUAL FOR THE SERIES SIX PROGRAMMABLE CONTROLLER.

SWITCHES

The Program Development Terminal has 2 switches located on the front panel as shown below in Figure 8.



1. AC power ON/OFF Switch
2. Mode Selection Keyswitch

FIGURE 8
PDT Switches

The ON/OFF switch is an illuminated rocker switch that controls the 115V AC input to the PDT. When the top of the switch is depressed, AC power is applied to the unit and the light in the switch turns on to indicate the presence of power.

The other switch is a 3 position, key-operated switch for selection of 1 of 3 operating modes. The 3 modes are On-Line, Monitor and Off-Line. The key is removable when turned to the Monitor mode.

The On-Line mode allows the user to read data from the CPU and to write data to the CPU. All tables and registers are updated automatically by the CPU to reflect the current operating state of the CPU. While in the On-Line mode, single word changes in the operating program in the CPU can be made to the CPU while it is running. Single word changes are changes such as a normally-open contact to a normally-closed contact, a relay to a one-shot, or overriding a contact. A detailed list of single word changes that are allowed can be found in Chapter 4.

In the Monitor mode the Program Development Terminal can read data from the CPU but may not transfer data to the CPU. All tables and registers are updated automatically to reflect the current operating state of the CPU.

In the Off-Line mode tables and registers are updated by command (not automatically) from the keyboard and single word changes while the CPU is running are not permitted. The Off-Line mode allows a convenient method for creating programs without the necessity of having the Program Development Terminal connected to the CPU. Programs can be created in a location distant from the CPU and transferred to tape. The tapes can then be used for entering the program into the CPU as required.

SYSTEM CONNECTION

For operation with the Program Development Terminal connected directly to the CPU, the PDT-to-CPU interface cable should be plugged into the upper connector on the I/O Control module in a CPU. The length of this cable is 10 feet.

The PDT-to-CPU cable can also be plugged into the lower connector on the last I/O Receiver module in a CPU station or a Local I/O station in the primary or auxiliary I/O chain. This provides a method of connecting the Program Development Terminal to an I/O rack in an I/O system that is distant from the CPU.

WARNING

Only 1 operating PDT may be connected to a system at a time. An attempt to operate a system with more than 1 Program Development Terminal may result in improper and possibly hazardous operation of the control system.

When connected to an I/O rack, the Program Development Terminal can be used to debug programs or diagnose hardware faults and allows the user to be close to the device(s) controlled.

The only noticeable effect in having the Program Development Terminal distant from the CPU is that communication between devices becomes slightly slower with increasing distance due to propagation delays.

CHAPTER II
Installation

The Program Development Terminal requires minimal installation procedures. It has been thoroughly tested at the factory and is ready for operation upon connecting the AC line cord. This chapter provides the installation instructions for your Program Development Terminal.

UNPACKING

The Program Development Terminal is shipped from the factory in a rugged cardboard container. Open the container. Inside, the Program Development Terminal has been packed in a urethane foam molded two-piece package. This enclosure prevents the Program Development Terminal from moving around during shipment.

Remove the top cap. The unit can now be lifted out of the bottom container. It is recommended that the shipping material be saved in the event that the Program Development Terminal must be shipped or transported to a different location or if it should need to be returned to the factory for any reason.

INSPECTION

Inspect the Program Development Terminal at this time to ensure that there has been no damage during shipment. If any shipping damage is noted, the carrier should be notified immediately. If the shipping container shows evidence of damage be sure to save it for inspection by the carrier.

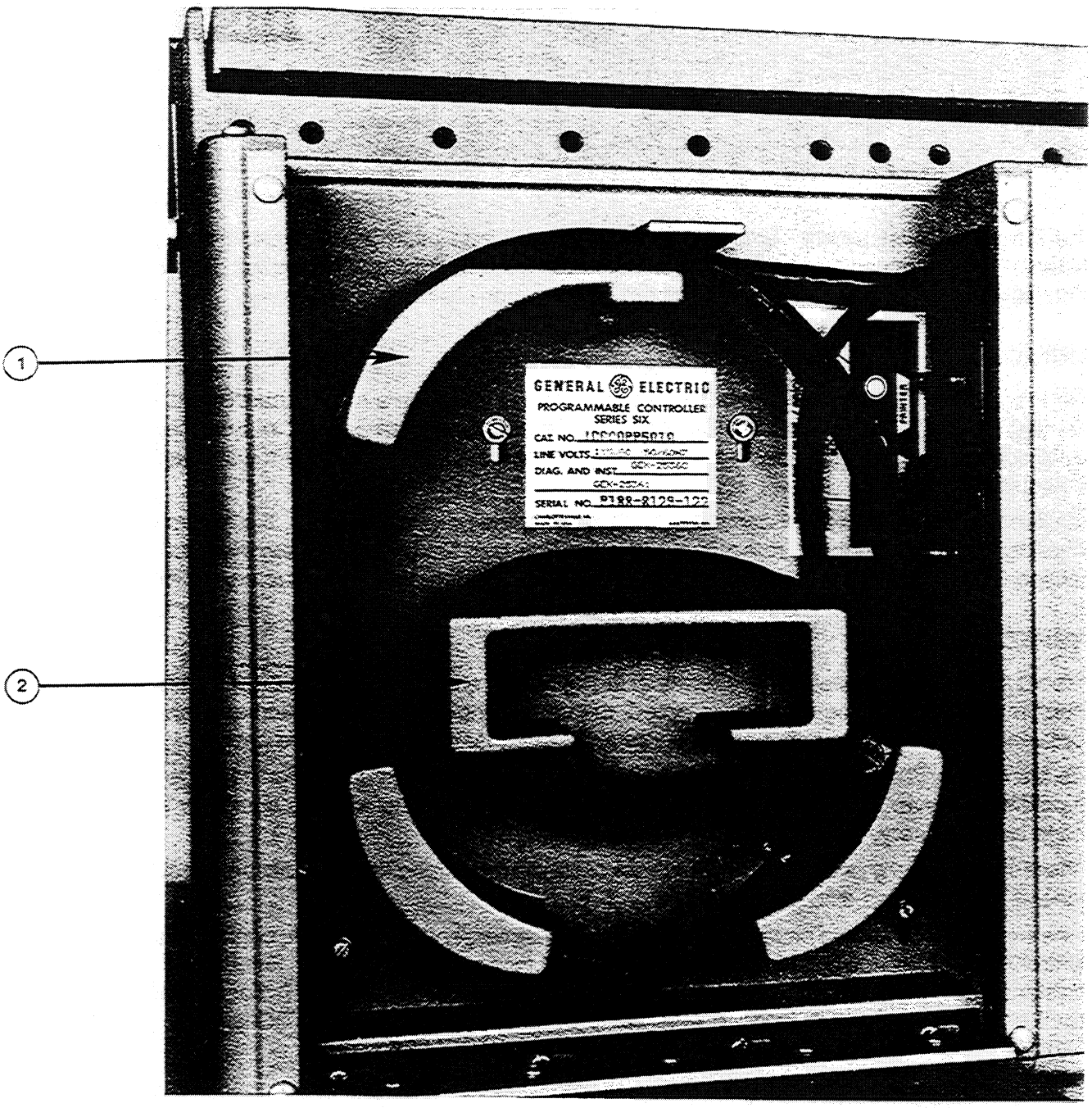
It is the responsibility of the consignee to register a claim with the carrier for damage incurred during shipment. However, General Electric Company will fully cooperate with the customer should any such action be necessary.

If there should be any problem with your order or for service during the warranty period contact your local General Electric sales office.

In the event service should be required because of an emergency breakdown, contact the Programmable Control Product Service Specialist at (804) 978-5624 for general information. For technical information or assistance call the Programmable Control Service Center at (804) 978-5747. For further information on troubleshooting and repair refer to Chapter 4 of GEK-25361, INSTALLATION AND MAINTENANCE MANUAL FOR THE SERIES SIX PROGRAMMABLE CONTROLLER.

CABLE CONNECTION

Set the Program Development Terminal on its bottom to allow access to the cable storage compartment which is located in the bottom of the unit. Lower the hinged panel. This allows access to the cables, connectors and controls located inside the compartment. Unwrap the PDT-to-CPU cable and the AC line cord from the cable wrap area as shown in Figure 1.



1. Unwrap PDT-to-CPU Cable From Here.
2. Unwrap AC Line Cord From Here.

FIGURE 1
Access to Cables and Connectors

Notice that a label in the rear of the cable storage area shows the location and indicates the function of each connector and control on the panel.

PRINTER CABLE

If a Terminet printer is to be connected to the Program Development Terminal it should be connected to the top connector using a cable with 25 pin D-type connector on both ends. The part number for the cable is IC600WF010A (Table 1) and its length is 10 feet. The cable is intended for use with a GE printer such as the Terminet 30, 300 or 1200. Other printers with an RS-232C interface can be used; however the signal and pin configuration should be verified as being correct before doing so.

NOTE

The standard cable supplied with a Terminet printer for use with a modem or computer is not compatible with the PDT.

TABLE 1
PDT to Terminet Cable

CONNECTOR 1 PIN (PDT)	RS-232 CIRCUIT	DESCRIPTION	CONNECTOR 2 PIN (PRINTER)	RS-232 CIRCUIT
2	BA	PDT Received Data	3	BB
3	BB	PDT Transmitted Data	2	BA
4	CA	Clear to Send	5	CB
5	CB	Request to Send	4	CA
7	AB	Signal Ground	7	AB

The printer port in the PDT is configured as shown below. This information should be used to verify that the signal and pin configuration of printers other than the Terminet are compatible for use with the PDT. It is also useful for those who wish to build their own cables.

PIN	SIGNAL	DESCRIPTION
2	TxD	Data transmitted by the PDT.
3	RxD	Data received by the PDT.
4	RTS	Request to Send (PDT Output).
5	CTS	Clear to Send (PDT Input).
7	GND	Signal Ground.
20	DTR	Data Terminal Ready (PDT Output).

EXTERNAL TAPE UNIT CABLE

If a STR LINK tape unit is to be connected to the Program Development Terminal, connection should be made to the lower 25 pin D-type connector. This cable is also 10 feet long and is part number IC600WG010A (Table 2). A STR LINK tape unit can be used directly connected to a Program Development Terminal at a location distant from a CPU, such as in a CPU I/O station or a Local I/O station. If a tape unit other than the STR LINK is to be used with the PDT, the signal and pin configuration should be verified as being compatible with the PDT.

TABLE 2
PDT to STR LINK Tape Unit Cable

CONNECTOR 1 PIN (PDT)	RS-232 CIRCUIT	DESCRIPTION	CONNECTOR 2 PIN (TAPE)	RS-232 CIRCUIT
2	BA	PDT Received Data	2	BA
3	BB	PDT Transmitted Data	3	BB
4	CA	Request to Send	4*	CA
5	CB	Clear to Send	5*	CB
6	CC	Data Set Ready	6	CC
7	AB	Signal Ground	7	AB
20	CD	Data Terminal Ready	20	CD

*Jumper pins 4 and 5 at tape end.

COMPOSITE VIDEO MONITOR CABLE

Connection of a video monitor to the Program Development Terminal requires a 75 ohm cable terminating at each end with a BNC connector. Connect 1 end of the cable to the composite video BNC connector on the panel in the cable storage compartment. Connect the other end to the BNC connector on the video monitor. The monitor must be capable of displaying digital data.

PDT-TO-CPU CABLE

If the Program Development Terminal is to be connected directly to the CPU, connect the interface cable to the top connector of the I/O Control module in the CPU rack. Tighten the mounting screws on the connector to ensure a secure connection.

If the Program Development Terminal is to be connected to an I/O rack in a CPU I/O station or a Local I/O station, the connector at the end of the cable should be connected to the bottom connector of the last I/O Receiver in that I/O station.

AC LINE CORD

Plug the 3-prong (grounding) line cord into an AC outlet which will provide the correct voltage, current, and frequency as specified in the power requirements listed in Table 1, Chapter 1. The PDT should be plugged into the same AC power source as the CPU or I/O rack that the PDT is connected to. The AC power source should be grounded properly in accordance with good safety practices. The power source should also be near enough to the unit to permit direct connection of the line cord. An AC power source not grounded properly may cause personal injury or may cause erroneous operation or damage to the Program Development Terminal.

After connecting all of the cables, close the hinged cover. The cables will feed out of the opening towards the top of the unit.

Set the Program Development Terminal in its horizontal position for normal operation. Lower the front hinged cover which allows access to the keyboard, CRT, and switches.

You are now ready to perform an initial start-up sequence which includes the CPU as well as the Program Development Terminal. This sequence will clear the CPU memory of any extraneous data which could cause improper operation of the system. The procedure also ensures that the proper required parameters are entered into the CPU.

START-UP INSTRUCTIONS

NOTE

The following steps are appropriate if the CPU has not previously been programmed. If the CPU contains a program, go to Chapter 3 after completing the PROM check.

- Turn the CPU key switches to the STOP and WRITE positions.
- Turn the Program Development Terminal key switch to OFF-LINE.
- Connect either the cable from an I/O rack or the terminator connector supplied with your CPU to the bottom connector on the I/O Control module.
- Connect the PDT-to-CPU cable to the top connector on the I/O Control module in the CPU or to the lower connector of an I/O Receiver in a CPU station or a Local I/O station.
- Turn on AC power to the CPU and any I/O racks in the system.
- Turn on AC power to the Program Development Terminal by depressing the rocker switch towards the top of the unit. The light in the rocker switch will come on to indicate the presence of power.
- The Program Development Terminal initially performs a PROM integrity check on power-up. At this time a display will be seen on the screen as shown below in Figure 2.
  To abort the PROM integrity check, depress {**CLEAR**}. This will cause the Supervisor menu to be displayed without going through the complete PROM integrity check sequence.

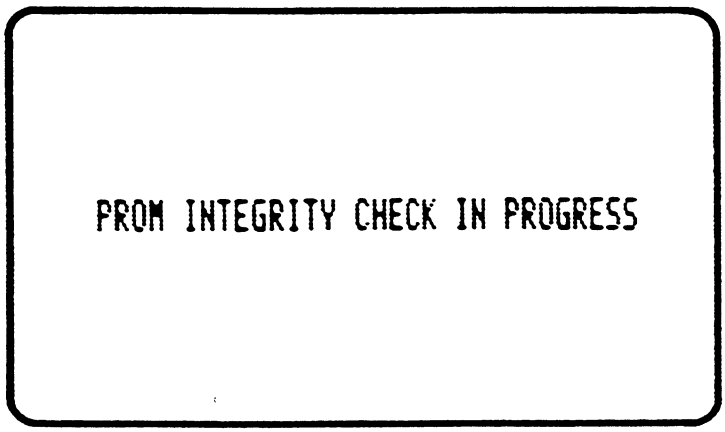


FIGURE 2
PROM Integrity Check Display

Allow the check to finish. This will take about 1 minute. When the check is complete and has passed, the SUPERVISOR DISPLAY will appear on the screen.

If the PROM integrity check does not pass, power down and back up to initiate the test again. If the test fails and an error message, RAM ERROR or PROM ERROR appears on the screen refer to the paragraph, INSPECTION on page 2.1 for information on assistance.

- When the SUPERVISOR DISPLAY appears on the screen move the cursor down to the item, DISPLAY SCRATCH PAD, by using the ↓ key located in the mode/control key group.
- Depress the {**ENTER**} key (also located in the mode/control group), the Scratch Pad display will now be displayed on the screen.
- Move the cursor next to each item displayed in reverse video. Reverse video is displayed as a white block with black characters inside the block. These items should be configured with the proper operating parameters for your system. Chapter 4 describes fully the required entries for the Scratch Pad. When powering up On-Line, this step is not necessary since the CPU automatically performs the configuration.
- Return to the SUPERVISOR DISPLAY by depressing the key sequence {**SUPERVISOR**}, {**ENTER**}. Both keys are located in the mode/control group.
- When the SUPERVISOR DISPLAY appears on the screen, move the cursor down to the last line of the display, CLEAR SCRATCH PAD AND TRANSITION PARITY ERROR.
- Depress the {**ENTER**} key, the message TO EXECUTE TYPE SHIFT-ENTER will appear in the work area in the lower right part of the screen. Depress {**SHIFT**} and {**ENTER**} simultaneously.
- The message BUSY will appear in the work area. When the BUSY status message disappears move the cursor up 1 line to the item, CLEAR LOGIC MEMORY PARITY ERROR.
- Depress the {**ENTER**} key, the message TO EXECUTE TYPE SHIFT-ENTER will again appear in the work area. Depress {**SHIFT**} and {**ENTER**} simultaneously.
- The BUSY status message will appear again. When it disappears move the cursor to the top line of the display, DISPLAY PROGRAM.
- Depress the following keys in the order given {**ENTER**}, {**PDT → CPU**}, {**ENTER**}. The message TO EXECUTE TYPE SHIFT-ENTER will be displayed in the work area, depress {**SHIFT**} and {**ENTER**} simultaneously.
- The memory clear data previously entered is now transferred to the CPU and the CPU memories are cleared.
- To verify that the PDT → CPU transfer has occurred the message for the CPU and Program Development Terminal Compare status should change from CPU NOT EQ PDT to CPU EQ PDT. This message is located on the fourth line in the upper part of the work area on the screen.

NOTE

The sequence of steps listed above assumes that the PDT is connected to a CPU. A CPU is not necessary for entering programs into a PDT in the OFF-LINE mode. If an attempt is made to clear any of the CPU memories without being connected to a CPU, the error message BAD DATA will be displayed on the screen.

The Program Development Terminal is now ready for you to become familiar with the keys and switch functions and then to begin entering ladder diagrams.

Proceed to Chapter 3 which will describe each of the keys. The key switch modes are also described in that chapter.

NOTE

Initialization of CPU status memory, normally accomplished at the factory or after battery disconnect/failure, can cause a 'BAD DATA' message to be displayed on the PDT. This message occurs after performing a CLEAR SCRATCHPAD AND TRANSITION PARITY ERROR function from the PDT's Supervisor menu and can be ignored. It was based on previous incorrect data that will be corrected as soon as the parity is reset.

CHAPTER III

Program Development Terminal Controls and Functions

INTRODUCTION

This chapter will introduce the user of a Series Six Program Development Terminal to the controls located on the unit. The controls consist of 2 switches and a keyboard consisting of 76 keys. Six of the keys have dual functions, and 12 of the keys generate menus for programming of mnemonic functions.

The first step in understanding the operation of the Program Development Terminal is to become familiar with the controls. Before attempting to enter a program, take the time to look at the mode switch and the keyboard. Familiarize yourself with the location of each key group and each key in each group.

As you read through this chapter try each key and observe the results of the key stroke. Notice particularly the work area display. Menus for each mnemonic key and other key entries will be displayed in this area.

Only when you become thoroughly familiar with the controls should you begin entering programs into the system. This manual will take you through the entry of a simple relay rung up to and including programs using the basic and extended mnemonic functions.

AC POWER SWITCH

The AC power switch is a 2 position rocker switch with a built-in indicator light. The Program Development Terminal requires an AC source of power. The switch is depressed towards the top of the unit to turn power on. To turn power off the switch is depressed towards the bottom of the unit. When AC power has been applied the indicator light will turn on.

MODE SELECTION KEYSWITCH

The mode selection keyswitch is located to the right of the AC power switch and is used to select the mode of operation for the Program Development Terminal. Three modes of operation are selectable; ON-LINE, MONITOR, and OFF-LINE. A legend indicating the keyswitch position for each mode is printed adjacent to each position on the switch.

The operation of the Program Development Terminal in each mode is described in the following paragraphs. Certain transfers of data between the Program Development Terminal and a CPU are performed while in the 3 modes of operation. A summary of the allowable transfers is shown below in Table 1.

TABLE 1—PDT Keyswitch Modes

TRANSFERS / PDT MODE	TABLE MEMORIES REGISTER MEMORY		USER PROGRAM (LOGIC MEMORY)	
	TO CPU	FROM CPU	TO CPU	FROM CPU
On-Line	M	A	M	M
Monitor	X	A	X	M
Off-Line	M	M	M	M

M: Transferred Manually
X: Not Permitted
A: Transferred Automatically

ON-LINE MODE

In the ON-LINE mode the CPU periodically reads an updated input and output status table, register memory, override table and Scratch Pad to the Program Development Terminal.

Transfers of information from the Program Development Terminal to the CPU and from the CPU to the Program Development Terminal are permitted by use of the PDT → CPU and CPU → PDT keys, respectively.

A limited number of single word changes may be made to the operating program in the CPU from the Program Development Terminal. The message "CPU EQ PDT" must appear in the work area before these changes can be made. Table 2 is a summary of single word program changes which may be made. These changes are entered directly into the CPU and will change the operation of the program.

TABLE 2—Summary of Single Word Program Changes

CONTACT CHANGES		OUTPUT CHANGES		MNEMONIC CHANGES	
From	To	From	To	From	To
Normally open	Normally closed	One-shot	Relay	Present register reference number	New register reference number
Normally closed	Normally open	Timer	Counter	Present status table reference number within table	New status table reference number within table
Input reference	Output reference	Counter	Timer		
Output reference	Input reference	Present output reference number	New output reference number		
Present status table reference number	New status table reference number	Present accumulate reference	New accumulate reference		

TABLE AND REGISTER CHANGES

Input Table — Turn a bit in the input table on or off. Change a word in the input table (on a byte boundary).

Output Table — Turn a bit in the output table on or off. Change a word in the output table (on a byte boundary).

Override Table — Override a bit in the input or output table.

Register Memory — Change the contents of a register.

Auxiliary Input Table — Turn a bit in the auxiliary input table on or off. Change a word in the auxiliary input table (on word boundary).

Auxiliary Output Table — Turn a bit in the auxiliary output table on or off. Change a word in the auxiliary output table (on word boundary).

WARNING

When making on-line changes to an operating program care must be taken since improper modification could cause hazardous conditions to exist which could damage equipment or cause personal injury.

The procedure for making single word changes will be described when discussing the PDT → CPU key. Multi-word changes are also possible with the scan temporarily halted and outputs suspended.

MONITOR MODE

In this mode of operation the operator can monitor the system operation by displaying powerflow, register content, and I/O status. As in the ON-LINE mode the status tables, register memory, override table and Scratch Pad are routinely updated. The user program in the CPU memory can be transferred from the CPU to the PDT by use of the CPU → PDT key. No transfers from the PDT to the CPU can be made while in the MONITOR mode. This is the only mode that allows the key to be removed.

OFF-LINE MODE

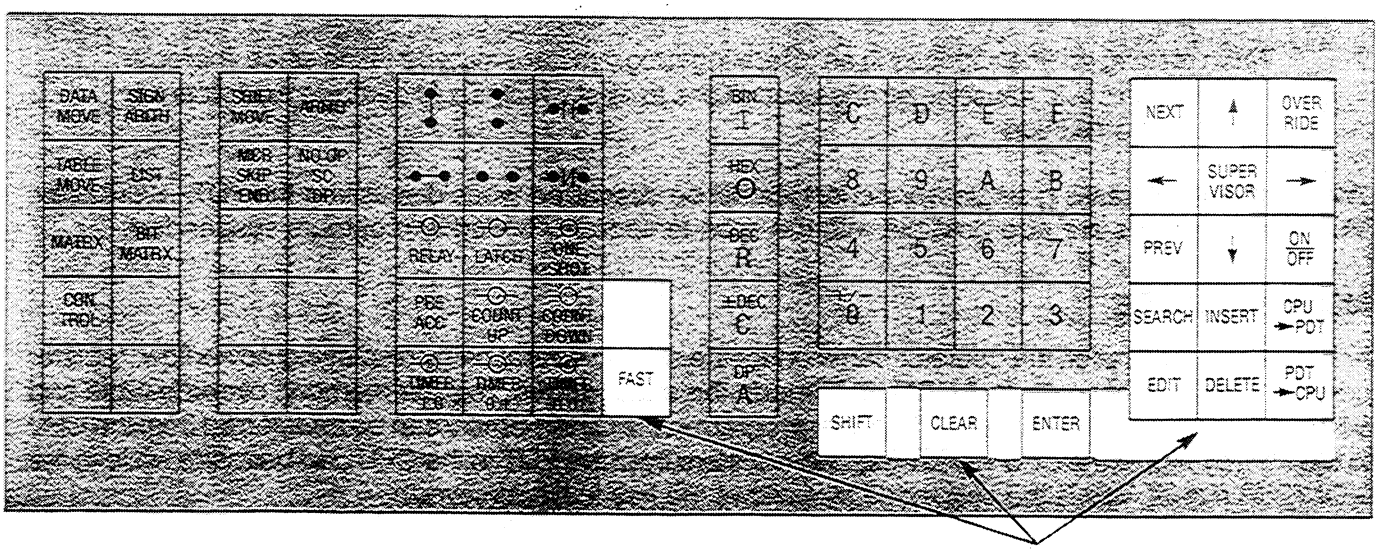
The OFF-LINE mode allows transfers from the PDT to the CPU and from the CPU to the PDT only by use of the PDT → CPU and CPU → PDT keys, respectively. No automatic transfers of data are made to the PDT from the CPU. The status tables, register memory, Scratch Pad and user memory can be transferred manually by use of the CPU → PDT key.

The OFF-LINE mode allows the Program Development Terminal to be used without being connected to a CPU. This feature is convenient in that the Program Development Terminal can be used at a location such as an office for entry of user programs. A program can be entered into the Program Development Terminal, edited and when correct recorded on tape. Recording the program on tape provides a safe storage for the program since the Program Development Terminal has no battery back-up for its RAM memory.

KEYBOARD OPERATION

Each of the keys on the keyboard are described in the following paragraphs. The keys will be described by keygroup. The keygroup will be highlighted and each key will be broken out of the group and described individually.

MODE/CONTROL GROUP



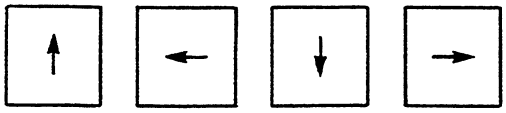
Mode Control Key Group

The mode/control keys are used to cause the Program Development Terminal to enter a requested mode or to provide various control functions required for normal operation of the Program Development Terminal. The color of these keys is light brown.

NOTE

In the following description of key operation, braces, { } are used to indicate key depression on the keyboard.

CURSOR CONTROL KEYS



The cursor is a position indicator viewed on the CRT display which indicates the position of data entry. When the program or a table is being displayed the cursor appears as an underline. In the EDIT mode the cursor is displayed as a reverse video block. The block covers 8 horizontal and 3 vertical characters on a ladder diagram. Figure 1 illustrates the various PDT cursors and describes their use.

	(1) Used to Select options from menu (2) Appears as a reverse video rectangle.
	(1) Indicates the position where a data entry is to be made. (2) Appears as a flashing underline.
	(1) Used when PDT is in EDIT mode. (2) The horizontal section indicates the position where elements can be entered. (3) The vertical section indicates where a vertical line can be entered. (4) Appears as a reverse video block.
	(1) Used by PDT when input or output table is displayed and is superimposed over a specific input or output. (2) Appears as a flashing reverse video rectangle.
	(1) Used by PDT when the register table is displayed and is superimposed over a specific register or a word in the input or output table. (2) Appears as a flashing reverse video block.

FIGURE 1
PDT Cursors

The movement of the cursor is controlled by the keys ↑, ←, ↓, and → which move the cursor in the indicated direction. If a key is depressed and held down, the cursor continues to move in the indicated direction until the key is let up.

Next Key NEXT

The NEXT key scrolls (advances) the ladder diagram display to the next rung in the user program or displays a selected rung as the top line of the display on the screen. In order to use this key the user program must be displayed on the screen by selecting the item DISPLAY PROGRAM from the Supervisor menu.

To scroll the display on the screen depress the {**NEXT**} Key. The second or next rung in the program will then move up or scroll to the top position of the display. The remaining rungs will move up and will fill the screen with as many rungs as will fit on the display (maximum of 8 lines). The top rung will be moved off of the screen.

Previous Key PREV

The PREV Key has 2 functions, the first function is, to scroll back 1 rung to the previous rung of logic and the second function is to return a rung of logic to its original configuration before an editing function has been completed.

In order to scroll the display back to the previous rung the user program must be displayed on the screen. If it is desired to observe or edit a rung in the program previous to the one displayed at the top of the screen depress the {**PREV**} Key. This will cause the display to scroll down and the rung previous to the one that had been at the top will now be displayed as the top rung. Each time the key is depressed the display will scroll back 1 rung.

When in the EDIT mode if it is desired to delete all changes and return to the logic as it was prior to the edit, the {**PREV**} key can be depressed to cause this action to take place. After depressing the {**PREV**} key, the message REMOVE CHANGES? TYPE SHIFT-PREV will appear in the work area. If the change is to be removed depress {**SHIFT**} and {**PREV**} simultaneously. Depressing {**CLEAR**} will cancel the request to remove changes. This procedure is particularly helpful when multiple errors have been made or for some other reason you should decide to abort the editing of that rung.

Override Key OVER RIDE

The OVERRIDE Key provides a means of overriding inputs and outputs in the user program. Inputs or outputs which are overridden are not changed by relay logic in the user's program. An overridden input or coil will remain in its overridden state regardless of the physical input device or solved relay logic of the coil. This key provides a useful tool when debugging a program or troubleshooting a hardware problem.

CAUTION

Mnemonic functions that operate on the status tables ignore overrides.

The OVERRIDE Key can be used in the ON-LINE mode when the compare message in the work area indicates that the CPU and PDT memories are equal.

To override an input, display the input status table with the cursor positioned on the input to be overridden. This can be accomplished by selecting the specific input directly by key sequence or by moving the cursor to the input. As an example, if in a table display, to select I0024 depress the key sequence {I}, {24}, {**ENTER**}. The input status table will be displayed with the cursor on I0024. The alternate method of selecting the desired input is to select the SUPERVISOR display, position the cursor next to the item DISPLAY REQUESTED STATUS OR REGISTER TABLE, depress {I}, {**reference number**}, {**ENTER**}. The input table is displayed. In both instances the input status table is selected from the SUPERVISOR display.

When the input to be overridden is selected depress the key sequence {**OVERRIDE**}, {**ENTER**}. It should

be noted that a word must be in the binary format before overriding. The input will be overridden and a visual indication will be provided by the cursor flashing on that input. Also, the input reference adjacent to INPUT at the top of the screen will flash. If the input is observed in the program while in the DISPLAY PROGRAM mode the input reference will also be flashing. The override condition will remain until it is removed by again depressing the {**OVERRIDE**} key followed by {**ENTER**} while the cursor is positioned on the input.

An output coil can be overridden in the output table in the same manner as in overriding an input. Select the OUTPUT status table and the specific output, depress {**OVERRIDE**} and {**ENTER**} to override the output. An output coil can also be overridden when the user program is displayed. Position the cursor under the output coil (position 10), depress {**OVERRIDE**} and {**ENTER**}. The output coil reference will flash indicating that it is overridden. If the OUTPUT status table is displayed the overridden output will be flashing in the table. The override condition will remain until the key sequence {**OVERRIDE**}, {**ENTER**} is again depressed. The flashing of the status bit and coil will stop verifying that the override condition has been removed. If an attempt is made to override a contact while in the Display Program mode with the user program displayed, the message NOT A COIL will be displayed in the work area.

When overriding an input or coil the state of that input or coil remains unchanged while it is overridden. The status of a coil or input can be changed by use of the ON/OFF Key (See ON/OFF Key description).

When an input contact or output coil is overridden it will be overridden at every point in the program where it is referenced. All overridden statuses are retentitive upon power failure.

NOTE

The OVERRIDE Key can be used only with the main I/O system. Coils and inputs in the auxiliary I/O system cannot be overridden.

On/Off Key

The ON/OFF Key, when depressed, turns an input or output on or off for 1 sweep of the CPU. This key may be used with the OVERRIDE Key to change the state of an overridden contact or coil.

When used with the OVERRIDE Key, the status of an input or output can be set on or off for the duration of the override condition.

When used without the OVERRIDE Key, the ON/OFF Key can set an input or output on or off for 1 sweep of the CPU when either the user program or the input or output status table is displayed. After 1 sweep, the I/O scan portion of the CPU cycle will set the status to the condition of the input or set the output to the result of the solution of the rung(s) of logic containing the coil. When in the DISPLAY PROGRAM mode the ON/OFF key can be used to turn an output coil on or off.

Input references that have no associated module in the I/O section and coils that are not used in the ladder diagram, will remain ON once activated by the ON/OFF key. Only additional action by this key can turn them off. The override status has no effect on the state of these inputs and coils.

Supervisor Key

The SUPERVISOR Key allows the SUPERVISOR menu to be selected and displayed on the screen. Depressing the key sequence {**SUPERVISOR**}, {**ENTER**} will cause the SUPERVISOR menu to be displayed in the middle of the screen.

The SUPERVISOR menu provides the user with a method of selecting various operational features of the Program Development Terminal. The cursor (which is a reverse video rectangle) should be positioned next to the desired line by using the ↑ and ↓ cursor control keys.

A detailed discussion of each of the items on the SUPERVISOR display can be found in Chapter 4.

Search Key

All references to a specified input, output, or register which have been entered into a ladder diagram program may be located by use of the SEARCH key.

The format is { } { } {SEARCH}

- { } → I, O, AI, AO, R
- { } → reference number 1-1024 (1-256 in a model 60), for register references; 1-1024 for I, O, AI, AO
- {SEARCH}

After entering the above format and depressing the {SEARCH} key, the program is searched from the beginning for a reference to the specified input or output. If a match is found, the rung containing the specified input or output is displayed at the top of the screen followed by as many rungs as will fit. Depressing {SEARCH} again causes program memory to be searched from the point of the last reference to the next reference. This process can continue until the end of the program has been reached.

A search can be initiated for any mnemonic. To search for a mnemonic depress the key in either the basic or extended function group which will call up the menu containing the desired mnemonic. Select the desired mnemonic from the menu, then depress the {SEARCH} Key. As an example, to search for an ENDSW the sequence would be as follows.

—This causes a menu to be displayed in the work area which is:

1 MCR
2 SKIP
3 END

, depress {3}, {SEARCH}.

Whenever a search is initiated, the status message 'BUSY' will be displayed in the lower part of the work area. This indicates that a search is in progress.

Searches can be intermingled with other types of display and programming activity. Any time that {SEARCH} is depressed, the search continues from the point where it previously stopped. If the search had stopped at the last word in memory, it starts again at the first.

The display of 2 ENDSW's at the top of the screen indicates that the search has reached the end of the program. To terminate a search at any time depress the {CLEAR} Key.

Searching for Auxiliary References

Any reference to the auxiliary I/O tables may be searched for in the same manner as a reference to the main I/O tables. When searching for a particular I/O reference in the auxiliary I/O table, the key A (Auxiliary) must be included in the key sequence for that I/O reference.

As an example, to search for the I/O reference Auxiliary Input 124 enter the following key sequence:
{A}, {I}, {124}, {SEARCH}

A search will be initiated and stopped when AI0124 is found. The rung containing the reference is displayed at the top of the screen.

Insert Key INSERT

This key is used to insert a new ladder diagram rung between 2 existing rungs in a program or to begin entry of a new program at the start of memory. When the key is depressed it puts the PDT in the EDIT mode.

If no program is entered into the PDT (initial power-up), use of the INSERT Key will cause the first rung to be inserted before the 2 End-of-Sweeps at the top of the CRT. Once a line of logic has been entered, the INSERT Key will cause a new rung to be inserted ***after*** the rung displayed as the top line of the CRT.

The first rung of logic entered will be at the start of memory. This will prevent the adding of rungs before the first rung during future editing. To prevent this from happening the mnemonic NO OP should always be entered as the very first rung of logic. The NO OP will in effect reserve the first memory location. Lines of logic may be entered after the NO OP.

Each depression of the {**INSERT**} Key allows 1 rung of logic to be entered. When the new rung of logic has been entered, depress the {**EDIT**} Key. This will check the validity of the rung of logic and cause it to be entered into the memory if it is correct. The PDT will leave the EDIT mode and return to the DISPLAY PROGRAM mode. If the new rung is to be deleted for any reason before entering it into the PDT memory, depress {**SHIFT**} {**PREV**}.

Edit Key EDIT

The EDIT Key initiates entry into the EDIT mode when the PDT is ON-LINE or OFF-LINE and is in the DISPLAY PROGRAM mode. By depressing {**EDIT**}, all rungs except the top one on the screen are removed from the display. The remaining rung is eligible for structural modifications (editing). If, while in the EDIT mode it is desired to return the rung to the condition it was in when the EDIT mode was entered, depressing {**SHIFT**} {**PREV**} will restore the rung to its initial condition. This is a useful procedure when making changes and a wrong entry has been made.

A rung is edited by positioning the cursor in the desired position on the screen and inserting horizontal and vertical line segments, contact, mnemonic and output symbols until the rung is complete. A rung can consist of 1 or more lines of logic. When the rung is complete, depressing {**EDIT**} causes the rung to be checked for validity and entered into the PDT memory if correct. It may then be stored on tape or transferred to a CPU for storage in the CPU memory.

Notice that when {**EDIT**} is depressed the display will flash. During this time the PDT is processing the rung as entered and will add lines and symbols to complete the display.

When a function is first entered, a block of asterisks will appear in certain positions in the display. These asterisks are prompts for the entry of operands. The operand blocks can be entered in any order.

When a function is selected from the menu and displayed on the screen the cursor will be positioned on the leftmost operand field in the function.

As each operand is specified the {**ENTER**} Key must be depressed and the operand is checked for validity by the PDT. If the entry is correct, it replaces the block of asterisks above the cursor. The operand type (I, O, R, AI or AO) and reference number must be entered. After operands have been entered they may be changed (while in the EDIT mode) without reentering the complete function.

All asterisks must be replaced by operands. If an attempt is made to leave the EDIT mode with any operands unspecified, the remaining blocks of asterisks will flash and the error message INCOMPLETE ELEMENT will be displayed in the work area. Table 3 lists several error messages and illegal conditions causing them. These messages will appear while in the EDIT mode. A complete list of error messages can be found in Chapter 4, Ladder Diagram Programming.

TABLE 3—EDIT Mode Error Messages

CONDITION	ERROR MESSAGE
Incorrect operand type	INVALID REFERENCE TYPE
Incorrect operand address	INVALID REFERENCE NUMBER
Unspecified operand(s) exist	MISSING REFERENCE
Unspecified constant value(s) exist	MISSING CONSTANT VALUE or INVALID ENTRY
Incorrect operand value	ILLEGAL NUMBER
Not enough room left in the line for display of function	NOT ENOUGH ROOM IN LINE

An example of a display with asterisk blocks is shown below in Figure 2. Two displays are shown. The top is an add function requiring the entry of register references. The bottom display is an Inclusive OR (extended function set). Note that the operand type for the length is specified as a constant since this is the only permissable operand type for a length operand (LEN). LEN operands do not have a default value, therefore, if an attempt is made to leave the EDIT mode without specifying a value for LEN, the word CONST will flash as a prompt for entry of LEN. The error message INCOMPLETE ELEMENT will be displayed in the work area.

```
  *****    *****     *****
+[  A   +   B   =     C  ]+

  *****    *****     *****     CONST
+[  A  IOR  B   =     C         LEN ]+
```

FIGURE 2
Function Display in Edit Mode
Before Entering of Operands

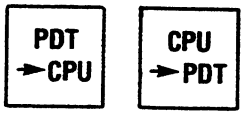
If a function is entered and the cursor is positioned so that the function being entered partially overlaps any other functions, the overlaid functions will be completely removed from the display.

Delete Key DELETE

The DELETE Key is used for removing individual rungs of logic from the user program. With the PDT in the DISPLAY PROGRAM mode, scroll the rung to be deleted so that it starts at the top line of the display. The

rungs may be scrolled by use of the PREV and NEXT keys, or an output coil may be moved to the top of the display by using the SEARCH function.

When the desired rung is at the top of the screen it may be deleted from the program by the key sequence {**DELETE**}, {**ENTER**}. The rung is then deleted from the PDT memory.

Transfer Keys 

The PDT → CPU and CPU → PDT Keys are used to initiate the transfer of information from the PDT to the CPU and from the CPU to the PDT respectively. The action taken when the transfer keys are depressed is dependent on the position of the mode selection keyswitch (as shown in Table 1, PDT Keyswitch modes).

When a program is displayed, the transfer keys if depressed will cause the contents of the logic memory, register memory, and all tables (input, output, override) to be transferred, again dependent on the position of the mode switch. To initiate a transfer depress the desired key, either {**PDT → CPU**} or {**CPU → PDT**} followed by {**ENTER**}. The message TO EXECUTE TYPE SHIFT ENTER is displayed in the work area of the PDT. To complete the transfer the {**SHIFT**} and {**ENTER**} keys must be depressed. To abort the transfer sequence, type {**CLEAR**}.

With the CPU running and the PDT key switch set to on-line or off-line depressing {**PDT → CPU**}, {**ENTER**} will cause a message to be displayed in the work area as shown below.

TRANSFER WILL
SUSPEND OUTPUTS
FOR XXXX MS

TO EXECUTE
TYPE SHIFT-ENTER

The transfer time XXXX is automatically calculated by the PDT and that worst case time in milliseconds (100 msec per K) will appear in the message. To complete the transfer hold down {**SHIFT**} and depress {**ENTER**}. The PDT will then write a SUSPEND I/O function followed by an ENDSW function into the start of logic memory. This procedure allows on-line changes to be made without affecting the I/O's for the period of time when the transfer is taking place. At the end of the transfer the I/O scan will be restored to normal operation by replacing them with the beginning of the user's program.

NOTE

Basic CPU's ignore the SUSPEND I/O function and will continue the I/O scan, but not solve logic, thus producing the same effect.

To transfer data from the PDT to the CPU with the CPU stopped, depress {**PDT → CPU**} followed by {**ENTER**}. The message displayed in the work area will be:

TO EXECUTE
TYPE SHIFT-ENTER

To complete the transfer depress {**SHIFT**} and {**ENTER**}. This procedure is the same for a system with the Basic function set only.

Shift Key SHIFT

The SHIFT key is used in conjunction with an operand key (see Figure 2) to perform a function associated with the legend on the upper part of the operand keys. The key sequences and the functions they initiate are listed below. Note that {SHIFT} is depressed and held down, then the selected key is depressed followed by {ENTER}.

{SHIFT} {BIN / I} —changes base to binary

{SHIFT} {HEX / O} —changes base to hexidecimal

{SHIFT} {DEC / R} —changes base to decimal

{SHIFT} {±DEC / C} —changes the base of the word that the cursor is on (register or status table) to signed decimal

{SHIFT} {DP / A} —changes the base of 2 registers to a signed, 10 digit decimal, double precision quantity when the register table is being displayed. The register the cursor is on and the next higher register are converted and the result is displayed right justified.

The SHIFT key is also used with the numeral 0 in the numeric key group.

{SHIFT} {+/− / 0} —changes the sign of a single or double precision decimal number. The default sign for signed numbers is +.

In addition the SHIFT key is used in the key sequence {SHIFT} {ENTER} when doing transfers or when clearing soft parity errors in the logic memory or scratch pad and transition tables.

When a transfer is to be made depress either the {CPU → PDT} or the {PDT → CPU} key. The message TO EXECUTE TYPE SHIFT-ENTER will be displayed in the work area. To complete the transfer, the {SHIFT} key must be depressed and held down then depress the {ENTER} key.

The sequence for clearing soft parity errors is to position the cursor on one of the 2 bottom items on the Supervisor display and depress {ENTER}. A message will then be displayed in the work area, TO EXECUTE TYPE SHIFT-ENTER. To complete the sequence depress {SHIFT} and {ENTER}.

WARNING

After a soft parity error is corrected, thus proving the source of the error, the entire program should be reloaded to insure no changes in the logic have occurred. Memory changes detected by the parity error checking can result in unreliable control system operation.

When editing a rung or entering a new rung depressing {SHIFT} (hold key down) then {PREV} before leaving the EDIT mode will return the rung to its previous state before making the new entries. For further detail on using the {SHIFT} key in conjunction with the {PREV} key, refer to the paragraph, Previous Key on page 3.6.

Clear Key

The CLEAR key is used to remove key entries displayed in the work area before they are entered into memory if a typing error has been made or for some other reason the data is to be changed. Depressing {**CLEAR**} will recover the PDT from an error message and will abort any operation with BUSY displayed such as a search.

Enter Key

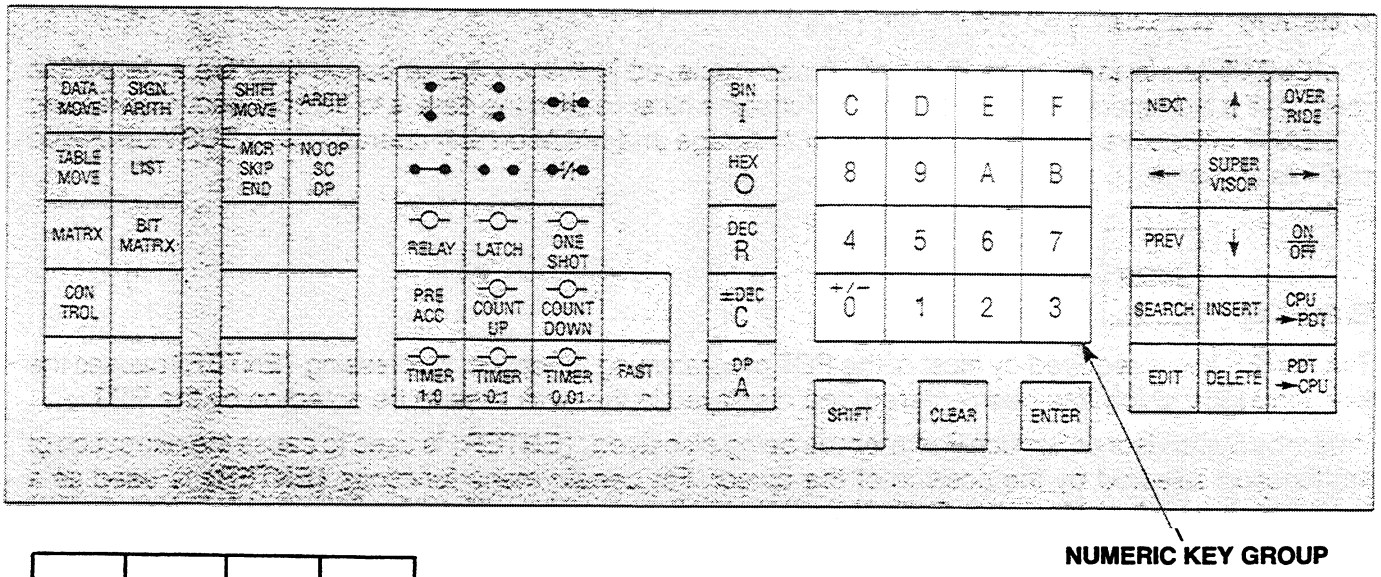
The ENTER key is required by most of the PDT programming functions. Depressing {**ENTER**}causes the key sequence which has been entered and displayed in the work area to be acted on by the PDT.

When the Supervisor or Scratch Pad menu is being displayed, {**ENTER**} is used to cause the execution of the function selected by the position of the cursor. For certain menu functions {**ENTER**} is used as a toggle, e.g. to select either the BASIC or EXTENDED functions when off-line and in the Scratch Pad display, or to cause a scroll through a list, e.g. BAUD RATE.

Fast

When in the program or table display mode this key allows the fast updating of the reference specified by the cursor. No other references on the screen will be updated. Moving the cursor or depressing {**CLEAR**} will cancel the fast update.

NUMERIC GROUP



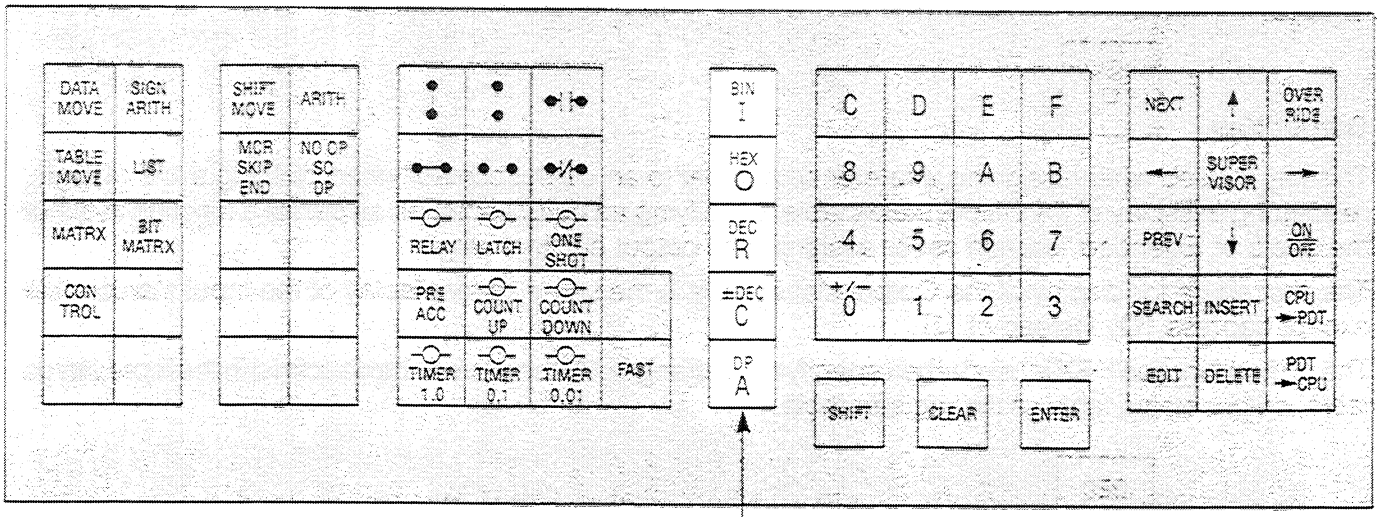
C	D	E	F
8	9	A	B
4	5	6	7
+/− 0	1	2	3

The numeric keys are located on a hexadecimal keypad and are used for entering numerical data. Typical uses of the numeric keys are for entering references for input contacts and output relays, as well as data for tables and responses to prompts.

The 0 key has a second function, +/− which is the shifted function. When used with the SHIFT key, +/− will cause the sign of a signed decimal number to be changed.

The color of the keys in the numeric key group is white.

OPERAND GROUP



OPERAND KEY GROUP

The operand keys all perform dual functions. They are used to specify input table, output table, register memory, constant, and auxiliary operands when in the EDIT mode and at other times when specific operands are to be selected.

When used in conjunction with the SHIFT key, the operand keys allow base changes of numeric data. The base changes specified by the shifted keys are binary (BIN), hexadecimal (HEX), decimal (DEC), signed decimal (±DEC), and signed decimal double precision (DP).

The color of the keys in the Operand key group is white.

Input Key

This key is used when assigning a reference to an input contact when building a line of logic, requesting a display of the Input status table or specifying an Input operand as part of a function in either the Basic or Extended function set.

To assign an input reference {I} precedes the input contact number which can be 1-1024.

To display the Input status table the item DISPLAY REQUESTED STATUS OR REGISTER TABLE is selected by positioning the cursor next to it on the Supervisor display. Depress {I} then the desired input number then {**ENTER**}. The Input status table will be displayed with the cursor positioned over the requested input.

NOTE

The base of an entire table can be set as follows:
{ }, { }, { }, and { }
- { } — base (optional), if other than the default base must be preceded by SHIFT. DP valid only in a Register table, BIN is not valid in a Register table
- { } — SHIFT (optional) held while selecting base
- { } — reference for cursor position (optional)
- { } — desired table, either I, O, AI, AO or R

The shifted function on the key, BIN is used for specifying a binary change of base value to binary in the input or output status table. Procedures for changing the base of a value in a table are fully described in Chapter 4.

Output Key

This key is used when assigning a reference number to an output contact when building a line of logic, requesting a display of the Output status table, specifying an output operand as part of a function in either the Basic or Extended function set or assigning an output coil in a rung.

The procedure for display of the Output status table is the same as for display of the Input status table except depress {O} instead of {I}.

The shifted function HEX is used when specifying a change of base value to hexidecimal in the input status table, output status table or the register table.

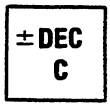
Register Key

The R key is used for assigning a register reference and for selecting the Register table for display on the screen. When a register reference is required as an operand with a function, {R} is depressed followed by the number of the selected register. Specific registers are assigned by assigning a number from 1 to 1024 in a model 600 or 6000 or 1 to 256 in a model 60 CPU.

The Register table is displayed by selecting the item DISPLAY REQUESTED STATUS OR REGISTER TABLE from the Supervisor display by moving the cursor next to it. Depress the {R} key followed by the register number followed by {ENTER}. The selected register will be displayed with the cursor positioned on the selected register. The register memory is not displayed on the screen in its entirety. A block of 128 registers is displayed at a time, e.g. 1-128, 129-256, etc. If an attempt is made to request a register number greater than 256 in a model 60 CPU, an error message will be displayed.

The contents of a register can be changed while displaying the register table by positioning the cursor under the word to be changed and entering the desired value. Display of registers content may be changed by use of the change of base keys which is described in detail in Chapter 4.

The shifted function DEC is used for specifying a change of base value to unsigned decimal in a status or register table.

Constant Key 

The C Key is used to specify constant operands when in the EDIT mode. Constant operands are used with many of the functions in the extended function set.

The shifted function, ±DEC is used to change the base of a word in the register or status tables to signed decimal.

Auxiliary Key

The A Key is used to enter a reference to an auxiliary input or output when in the EDIT mode or to select an auxiliary input or output table from the Supervisor display menu.

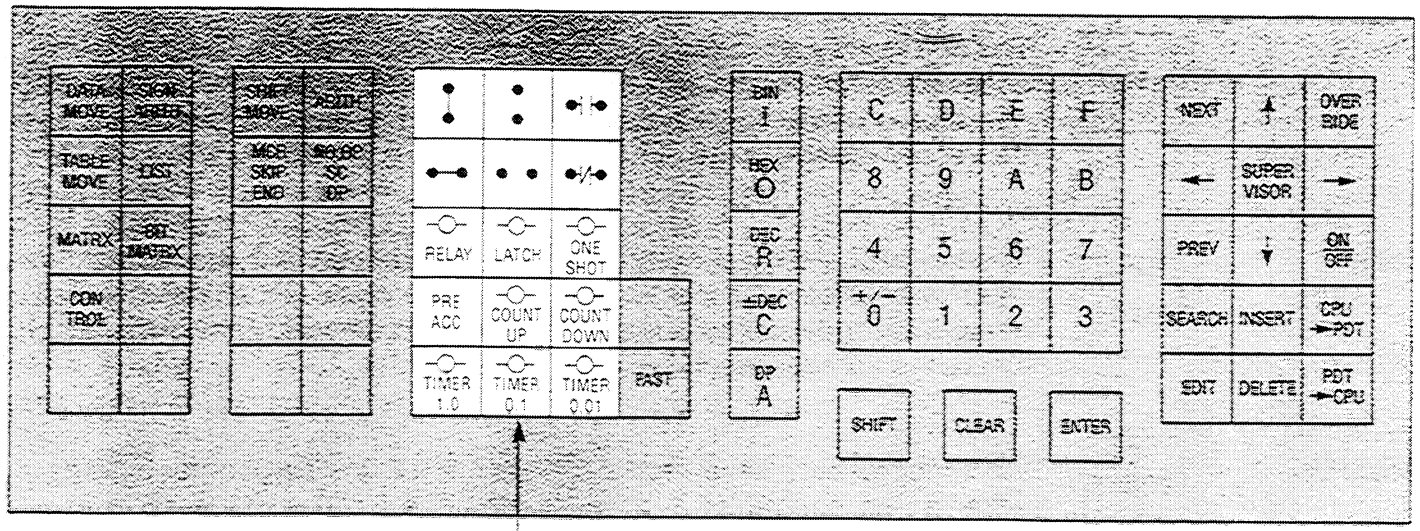
To specify an auxiliary reference or select an auxiliary status table, the key stroke {A} is included in the key sequence.

By use of the auxiliary key with a model 6000 CPU's; Auxiliary I/O module, an additional 1000 inputs and 1000 outputs are accessed. This comprises the auxiliary I/O Chain. The auxiliary key also can be used when specifying an auxiliary reference for internal storage in any model CPU (only the model 6000 can have physical auxiliary inputs and outputs). If an attempt is made to assign an auxiliary reference to a contact in a CPU with the Basic function set selected, the following error message will be displayed:

REQUIRES
EXTENDED SET

The shifted function, DP is used with the register tables to change the base of 2 registers to a signed, 10 digit decimal, double precision quantity.

RELAY GROUP



RELAY KEY GROUP

The keys in this group are used when building relay and contact networks. Together they form the basic contact connections and functions in a program as it is entered into the PDT. The color of the keys in the Relay key group is dark brown.

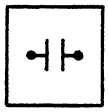
Horizontal Connector — This key allows two elements to be connected within a line of logic.

Vertical Connector — This key allows two lines of logic to be connected within a rung.

Horizontal Open — Allows the deletion of elements from a horizontal line of logic.

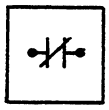
Vertical Open — Allows the removal of a vertical connection between two lines of logic.

Normally-Open Relay Contact



Allows entry of normally-open contacts in lines of logic.

Normally-Closed Relay Contact



Allows entry of normally-closed contacts in lines of logic.

Relay

This key allows the assignment of a relay coil in a rung of logic. When entering a relay coil O or AO (AO valid only with the extended set) must be specified and a reference number from 1 to 1024 in a model 600 or 6000 CPU or 1 to 256 in a model 60. An example of a required entry is {**O**}, {**24**}, {**RELAY**}. The cursor must be in position 10 when entering a relay.

One Shot

This key allows entry of a coil which functions as a one-shot. The output coil specified as a one-shot is activated for 1 sweep of the CPU. Entry of a one-shot is similar to that of a relay *except* that the ONE SHOT key is depressed instead of the RELAY key.

Latch

This key allows entry of a latch function in a program. A 2 line network is used when a latch function is specified. The top line is the latch contact network and the bottom line is the unlatch contact network. Entry of the latch function is fully described in Chapter 4.

Count Up

Count Down

When depressed, these keys allow entry of a 2 line network for building either an up counter or a down counter. The top line is the count line and the bottom line is the reset line. The count line requires entry of a constant preset or a preset value in a register while the reset line requires the specifying of a register to contain the accumulated value of the counter. Entry of both types of counters is fully described in Chapter 4.

Timers

0.01 Sec.

0.1 Sec.

1.0 Sec.

These keys allow building of timing functions with minimum time intervals of either .01, .1 or 1.0 seconds. The top line is the run line while the bottom line is the reset line. A constant preset or a preset value contained in a register is required in the run line. A register to contain the accumulate value must be specified in the reset line. Entry of timers is described in Chapter 4.

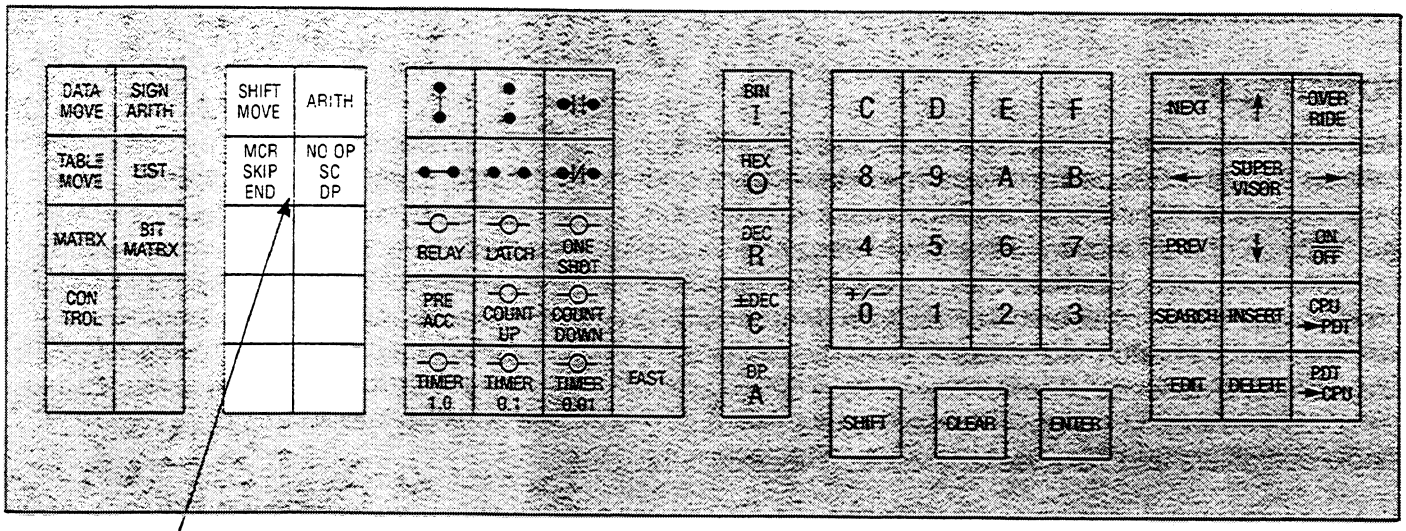
Preset Accumulate

When this key is depressed it causes a menu to be displayed in the work area for selection of the required mnemonic function, either PRESET CONSTANT, PRESET REGISTER or ACCUMULATE REGISTER. The display is as shown below.

1 PRESC
2 PRERG
3 ACCRG

To select the required mnemonic, depress the number preceding it and ENTER. The cursor must be in position 9 of the top or bottom line of a counter or timer network.

BASIC MNEMONIC GROUP



BASIC MNEMONIC KEY GROUP

The keys in this group when depressed cause a menu to be generated which is displayed in the lower part of the work area. A menu consists of from 3 to 7 items.

To select an item from each menu, depress the number on the numeric keypad corresponding to the number displayed in front of the desired menu item followed by {**ENTER**}.

These keys along with the keys in the relay group comprise the Basic function set. The color of the keys in this group is dark brown.

Each chart on the following pages describes the functions in a menu generated by depressing one of the keys in the Basic mnemonic key group. The headings for each column and their content is described below.

MENU DISPLAY

The name or abbreviation for each of the Basic functions in a menu generated by depressing the key shown at the top left of the chart is displayed in this column. The number preceding a function (followed by ENTER) is depressed after the menu is displayed in order to select that particular function.

MNEMONIC FORMAT

The display for each function as it will be viewed on the screen is shown here. The asterisks indicate that an operand reference is required in that position. The operand reference can be R, I, or O and a number. If a constant value is required, CONST will be displayed.

The allowable displayed contents for the selected table or constant are shown beneath the mnemonic. The letter d indicates an unsigned decimal value and the letter h indicates a hexadecimal digit.

VALID REFERENCE ENTRIES

The valid range of addresses for each operand reference is shown in this column. R indicates a register, I or O indicate the status tables. For register references, valid ranges for model 600 and 6000 will be shown. When programming the model 60, the range should be reduced by 768 to reflect the model 60's capacity of 256 registers

VALID RANGES FOR DISPLAYED DATA

This column shows the valid ranges (minimum to maximum value) for the displayed data.

A brief description of each Basic mnemonic function is given following each chart. Detailed programming steps for the Basic functions can be found in Chapter 5.

A chart for the PRE/ACC key is included in the following descriptions since it generates a menu listing mnemonic functions.

POWER FLOW

Many of the mnemonic functions are controlled by symbolic power flow operating from the left vertical rail towards the right. This power flow can be controlled by a single relay contact, a group of contacts in series as well as parallel, or other mnemonic functions. In addition, some mnemonic functions pass power flow to the "output" of the function. Such power flow can be used to energize coils, activate other mnemonic functions, or ignored. Limited only by a total of nine series elements, relay contacts can be added to condition the output of mnemonic functions prior to supplying it to other logic elements. Some mnemonic functions are not performed unless activated by receiving power flow, others are performed unconditionally even in the absence of power flow. Unless otherwise indicated, all mnemonic functions are performed (executed) only upon receipt of power flow (activated). Power flow is an important feature of programmable controllers and very useful to control mnemonic functions.

MENU DISPLAY	MNEMONIC FORMAT	VALID REFERENCE ENTRIES	VALID RANGES FOR DISPLAYED DATA
1 PRESC	CONST +[PRESC]+ ddd	None	ddd = 0 to 999
2 PRERG	***** +[PRERG]+ ddddd	*****(R) = 1 to 1024	ddddd = 0 to 65,535
3 ACCRG	***** +[ACCRG]+ ddddd	*****(R) = 1 to 1024	ddddd = 0 to 65,535

PRESC PRESet Constant

The constant value specified in the function is used as a preset value. This function may only be used in conjunction with timers and counters. Execution is unconditional.

PRERG PREset from ReGister

The contents of the specified register are used as a preset value. This function is used only with timers and counters. Execution is unconditional.

ACCRG ACCumulate from ReGister

The register specified in the function is used as a storage location for the accumulated value of a timer or counter. Execution is unconditional.

ARITH **Arithmetic Key**

MENU DISPLAY	MNEMONIC FORMAT	VALID REFERENCE ENTRIES	VALID RANGES FOR DISPLAYED DATA
1 ADD	***** ***** ***** +[A + B = C]+ ddddd ddddd ddddd	*****(R) = 1 to 1024	ddddd = 0 to 65,535
2 SUB	***** ***** ***** +[A - B = C]+ ddddd ddddd ddddd	*****(R) = 1 to 1024	ddddd = 0 to 65,535
3 COMPR	***** ***** +[A : B]+ ddddd ddddd	*****(R) = 1 to 1024	ddddd = 0 to 65,535

ADD ADD Binary

The contents of register A (first operand) and register B (second operand) are added together (binary add). The result is placed in register C (third operand). If the operation results in a carry out of the most significant bit (result too large to store in one register), power flow is outputted; otherwise no power flow is generated.

SUB SUBtract Binary

The contents of register B (second operand) are subtracted from the contents of register A (first operand). The absolute value of the result is placed in register C (third word). If the result of the operation is 0 or negative, power flow is outputted; if the result is greater than zero, no power flow is generated.

COMPR COMPaRe

The contents of register A (first operand) are compared to the contents of register B (second operand). If they are equal, power flow is outputted, otherwise no power flow is generated.

SHIFT MOVE **Move and Convert Key**

MENU DISPLAY	MNEMONIC FORMAT	VALID REFERENCE ENTRIES	VALID RANGES FOR DISPLAYED DATA
1 SHIFT	***** +[SHIFT]+ hhhh	*****(I or O) = 1 - 1009	hhhh = 0 to FFFF
2 I/OR	***** ***** +[I/O TO REG]+ hhhh hhhh	*****(I or O) = 1 - 1009 *****(R) = 1 to 1024	hhhh = 0 to FFFF
3 RI/O	***** ***** +[REG TO I/O]+ hhhh hhhh	*****(R) = 1 to 1024 *****(I or O) = 1 - 1009	hhhh = 0 to FFFF
4 BIBCD	***** ***** +[BIN TO BCD]+ ddddd hhhh	*****(R) = 1 to 1024 *****(I or O) = 1 - 1009	ddddd = 0 to 65,535 hhhh = 0 to FFFF
5 BCDBI	***** ***** +[BCD TO BIN]+ hhhh ddddd	*****(I or O) = 1 - 1009 *****(R) = 1 to 1024	hhhh = 0 to FFFF ddddd = 0 to 65,535

SHIFT SHIFT

The 16 bits of the status table specified by the operand are shifted 1 bit to the left. A zero is loaded into the least significant bit of the word. If the most significant bit is a one, power flow is outputted; if it is a zero, no power flow is generated.

I/OR Move I/O table to Register

The 16 bits of the status table specified by the first operand are loaded into the register specified by the second operand.

RI/O Move Register to I/O table

The 16 bit contents of the register specified by the first operand are loaded into the status table at the address specified by the second operand.

BIBCD BInary to BCD convert

The contents of the register specified by the first operand are converted from binary to 4 digit BCD and are stored in the status table at the address specified by the second operand. If the contents of the register are greater than 9999 the four least significant BCD digits of the result are stored in the status table, and power flow is outputted, otherwise no power flow is generated.

BCDBI BCD to BInary convert

The 16 bit contents of the status table specified by the first operand are converted from 4 digit BCD to binary and are stored in the register specified by the second operand. If there are any illegal BCD digits in the first operand the result will be computed as the binary sum of the BCD weights of the bits and power flow is outputted, otherwise no power flow is generated.

MCR SKIP END **Control Key**

MENU DISPLAY	MNEMONIC FORMAT	REQUIRED ENTRIES
1 MCR	CONST +[MCR]+ ddd	ddd = 0 to 255
2 SKIP	CONST +[SKIP]+ ddd	ddd = 0 to 255
3 ENDSW	+[ENDSW]+	NONE

MCR Master Control Relay

This function implements a master control relay function. The number of rungs within the scope of the master control relay is determined by the specified constant. Rungs following the MCR function that end in a timer, counter, one-shot, latch or relay are counted to determine the scope of the master control relay. A constant value of 0 has special meaning, and indicates that the scope of the MCR extends to the first ENDSW or RETURN function encountered. The scope of an active MCR is terminated if an ENDSW or RETURN function is encountered.

When the MCR is active, all outputs associated with relays, that are within the scope of the function are turned off if they are not overridden, and the execution of all other functions within this scope is skipped. Power flow is not outputted by this function.

SKIP SKIP

This function allows the execution of all or part of the program to be conditionally skipped. The number of rungs within the scope of the SKIP is determined by the specified constant. Rungs following the SKIP function that end in a timer, counter, one-shot, latch or relay are counted to determine the scope of the SKIP. A constant value of 0 has special meaning, and indicates that the scope of the SKIP extends to the first ENDSW or RETURN function encountered. The scope of an active SKIP is terminated if an ENDSW or RETURN function is encountered. Power flow is not outputted by this function.

When the SKIP is active, the execution of all functions that are within the scope of the function is skipped.

ENDSW END of SWeep

This function tells the CPU to enter its executive routine. ENDSW must be the last function in a ladder diagram program. Two consecutive ENDSW functions are interpreted by the PDT to be the absolute end of the program. When programming subroutines with the Extended function set, one ENDSW indicates the end of the main user program. The function following the single ENDSW is the first word of the first user subroutine. Power flow is not outputted by this function.

Communications Request/No Operation Key

MENU DISPLAY	MNEMONIC FORMAT	VALID REFERENCE ENTRIES	VALID RANGES FOR DISPLAYED DATA
1 NO OP	+[NO OP]+	None	
2 SCREQ	***** +[SCREQ]+ hhhh	*****(R) = 1 to 1018	hhhh = 0 to FFFF
3 DPREQ	***** +[DPREQ]+ hhhh	*****(R) = 1 to 1018	hhhh = 0 to FFFF

NO OP NO OPeration

No operation is performed by this function. It occupies 1 word of user memory. This function should be the first word of a program. The NO OP function can also be used to separate sections of the program so they can be located easily by use of the search function. Another use of the NO OP function is to set a line of logic permanently on, e.g., the run line of a timer. Execution is unconditional.

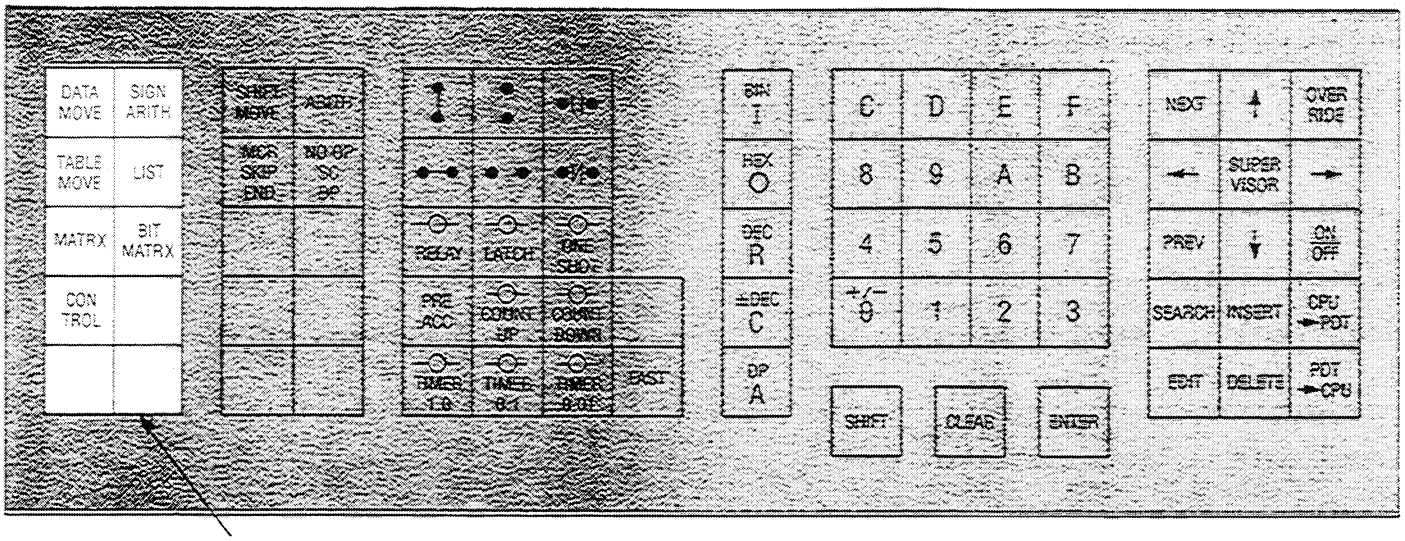
SCREQ Serial Communication REQuest

This function initiates communication with the Communication Control Module. The register referenced in the function is the first of a block of registers containing all parameters necessary to the Communication Control module. If a previous SCREQ was not honored, power flow is outputted, otherwise no power flow is generated.

DPREQ Data Processor REQuest

This function initiates communication with the Data Processor. The register referenced in the function is the first of a block of registers containing all parameters necessary to the Data Processor. If a previous DPREQ was not honored, power flow is outputted, otherwise no power flow is generated.

EXTENDED MNEMONIC GROUP



EXTENDED MNEMONIC KEY GROUP

The keys in this group when depressed cause menus to be generated and displayed for selection of the Extended mnemonic functions. The Extended mnemonic option available for all models of the Series Six CPUs (standard in the model 6000) provides capabilities beyond those in the basic group. The Extended mnemonic functions provide control for applications requiring more than relay, timing and counting functions.

Each key when depressed causes a menu to be displayed in the work area listing several functions. To select the desired function from the menu depress the number on the numerical keypad corresponding to the number displayed next to the item.

Each chart on the following pages is relevant to a particular key in the Extended Mnemonic key group. The headings for each column and their content is as follows.

MENU DISPLAY

The name or abbreviation for each Extended function relating to the key shown at the top of the chart is listed in this column. The menu in its entirety is displayed in the lower right section of the work area. The number displayed in front of each function is depressed to select that particular function.

MNEMONIC FORMAT

The display for each function as it will be viewed on the screen is depicted here. The asterisks at the top of a word in the function indicate that an operand reference is required, either R, I, O, AI, AO or CONST and a reference number. If a constant is required, CONST will be displayed.

The contents of each reference are shown beneath the mnemonic. As a guide to the allowable contents, the types of valid numbers are indicated. The letter d means an unsigned decimal number, −d means a signed decimal number (2's complement), −D means a signed decimal, double precision, number (2's complement) and h indicates a hexadecimal number.

VALID REFERENCE ENTRIES

The valid range for each operand reference is listed in this column. If a register reference is indicated, its valid range for a model 600 or 6000 is shown. When programming a model 60, the range should be reduced by 768 to reflect the model 60's capacity of 256 registers.

VALID RANGES FOR DISPLAYED DATA

The valid ranges for the displayed numerical data are indicated in this column. The numerical ranges for all displayed data are given as minimum to maximum numbers. Any constraints or values unique to a particular reference are noted.

A brief description of each Extended mnemonic function follows each chart. Detailed programming steps for the Extended mnemonic functions can be found in Chapter 6.

DATA MOVE Data Move Key

MENU DISPLAY	MNEMONIC FORMAT	VALID REFERENCE ENTRIES	VALID RANGES FOR DISPLAYED DATA
1 MOVE	***** ***** +[A MOVE B]+ -ddddd -ddddd	*****(I,O) = 1 to 1009 *****(AI,AO) = 1 to 1009 *****(R) = 1-1024	If I,O,AI,AO or R -ddddd = -32,768 to +32,767 If a Constant ddddd for A = -16,384 to +16,383
2 MOVE RIGHT 8	***** ***** +[MOVE RIGHT 8 BITS]+ hhhh hhhh	*****(I,O) = 1 to 1016 *****(AI,AO) = 1 to 1016 *****(R) = 1-1024	hhhh = 0 to FFFF Hexadecimal
3 MOVE LEFT 8	***** ***** +[MOVE LEFT 8 BITS]+ hhhh hhhh	*****(I,O) = 1 to 1009 *****(AI,AO) = 1 to 1009 *****(R) = 1-1024	hhhh = 0 to FFFF Hexadecimal
4 BLOCK MOVE	***** +[BLOCK MOVE]+ Data Word #1-Data Word #7 -ddddd -ddddd	*****(I,O) = 1 to 904 *****(AI,AO) = 1 to 912 *****(R) = 1-1018 Word #1-#7 if a Constant = 16 bit two's complement number	-ddddd = -32,768 to +32,767

NOTE

All signed numbers are stored in 2's complement form. Single precision constants are 15 bits, and double precision constants are 31 bits. The CPU extends (copies) the sign bit one place to the left, so constants can be handled as word or double word quantities, respectively.

MOVE MOVE A to B

If active, the contents of reference A are copied into reference B, and the power flow is outputted.

MOVE RIGHT 8 MOVE RIGHT 8 bits

If active, the right half of the contents of the first reference are copied into the right half of the second reference; the left half of the second reference remains unchanged. Power flow is outputted whenever this function is active.

MOVE LEFT 8 MOVE LEFT 8 bits

If active the left half of the contents of the first reference are copied into the left half of the second reference; the right half of the second reference remains unchanged. If a constant is used as the first reference, its value is limited from −16,384 to +16,383. Power flow is outputted whenever this function is active.

BLOCK MOVE BLOCK MOVE

If active, the 7 constant operands contained in the function are copied into 7 locations in a table (R, I, O, AI, AO), starting at the address specified by the reference. The constants in this function are true 16 bit numbers, and are not sign extended by the CPU. Power flow is outputted whenever this function is active.

SIGN ARITH

Signed Arithmetic Key

MENU DISPLAY	MNEMONIC FORMAT	VALID REFERENCE ENTRIES	VALID RANGES FOR DISPLAYED DATA
1 DPADD	***** ***** ***** +[A DPADD B = C]+ D D D	*****(I,O) = 1 to 993 *****(AO,AO) = 1 to 993 *****(R) = 1–1023	D = −2,147,483,648 to +2,147,483,647 If A or B Constant then D = −1,073,741,824 to +1,073,741,823
2 DPSUB	***** ***** ***** +[A DPSUB B = C]+ D D D	*****(I,O) = 1 to 993 *****(AI,AO) = 1 to 993 *****(R) = 1–1023	D = −2,147,483,648 to +2,147,483,647 If A or B Constant then D = −1,073,741,824 to +1,073,741,823
3 ADDX	***** ***** ***** +[A ADDX B = C]+ -ddddd -ddddd -ddddd	*****(I,O) = 1 to 1009 *****(AI,AO) = 1 to 1009 *****(R) = 1–1024	−ddddd = −32,768 to +32,767 If A or B Constant then −ddddd = −16,384 to +16,383
4 SUBX	***** ***** ***** +[A SUBX B = C]+ -ddddd -ddddd -ddddd	*****(I,O) = 1 to 1009 *****(AI,AO) = 1 to 1009 *****(R) = 1–1024	−ddddd = −32,768 to +32,767 If A or B Constant then −ddddd = −16,384 to +16,383

SIGN ARITH

Sign Arithmetic Key (Contd)

MENU DISPLAY	MNEMONIC FORMAT	VALID REFERENCE ENTRIES	VALID RANGES FOR DISPLAYED DATA
5 MPY	***** ***** ***** +[A MPY B = C]+ -ddddd -ddddd D	*****(A,B) = 1–1009 if I,O *****(A,B) = 1–1009 if AI, AO *****(A,B) If R = 1–1024 *****(C) = 1–993 if I,O *****(C) = 1–993 if AI,AO *****(C If R) = 1–1023	–ddddd (A&B) = –32,768 to +32,767 –ddddd (If A and B a 15 bit Constant) = –16,384 to +16,383 D(C) = –2,147,483,648 to 2,147,483,647
6 DVD	***** ***** ***** ***** +[A DVD B = QUO REM]+ D -ddddd -ddddd -ddddd	*****(A) = 1–993 if I,O *****(A) = 1–993 if AI,AO *****(A) If R = 1–1023 *****(B, QUO, REM) = 1–1009 if I,O *****(B, QUO, REM) = 1–1009 if AI,AO *****(B, QUO, REM) if R = 1–1024	D(A) = –2,147,483,648 to +2,147,483,647 D(If A is a 31 bit Constant) –1,073,741,824 +1,073,741,823 –ddddd (B,QUO,REM) = –32,768 to +32,767 –ddddd (If B is a 15 bit Constant) = –16,384 = +16,383
7 GREATER THAN	***** ***** +[A GREATER THAN B]+ D D	*****(I/O) = 1–993 *****(AI,AO) = 1–993 *****(R) = 1–1023	D = –2,147,483,648 to D = +2,147,483,647 D(A,B Constant) = –1,073,741,824 to +1,073,741,823

NOTE

All signed numbers are handled in 2's complement form.

DPADD Double Precision ADD

References A, B, and C are signed, double precision numbers. If active, the contents of reference A are added to the contents of reference B and the result is placed in reference C. If a 2's complement overflow occurred power flow is outputted; otherwise there is no power flow generated. A positive overflow will cause reference C to be set to +2,147,483,647, and a negative overflow will cause reference C to be set to −2,147,483,648.

DPSUB Double Precision SUBtract

References A, B, and C are signed, double precision numbers. If active, the contents of reference B are subtracted from the contents of reference A and the result is placed in reference C. If a 2's complement overflow occurred power flow is outputted; otherwise there is no power flow generated. A positive overflow will cause reference C to be set to +2,147,483,647, and a negative overflow will cause reference C to be set to −2,147,483,648.

ADDX Extended ADD

References A, B, and C are signed, single precision numbers. If active, the contents of reference A are added to the contents of reference B and the result is placed in reference C. If a 2's complement overflow occurred power flow is outputted; otherwise there is no power flow generated. A positive overflow will cause reference C to be set to +32,767, and a negative overflow will cause reference C to be set to −32,768.

SUBX Extended SUBtract

References A, B, and C are signed, single precision numbers. If active, the contents of reference B are subtracted from the contents of reference A and the result is placed in reference C. If the result is zero or negative, power flow is outputted; otherwise no power flow is generated. If a positive overflow occurs, reference C will be set to +32,767, and a negative overflow will cause reference C to be set to −32,768.

MPY Multiply

References A and B are signed, single precision numbers. Reference C is a signed, double precision number. If active, the contents of reference A are multiplied by the contents of reference B, and the result is placed in reference C. Power flow is always outputted when this function is active.

DVD Divide

Reference A is a signed, double precision number, References B, QUO (quotient), and REM (remainder) are signed, single precision numbers. If active, a test is performed to see if the division of the contents of reference A by the contents of reference B would result in an overflow (including divide by zero). If an overflow would result, the division is not performed, and power flow is outputted. If an overflow will not result, the division is performed; the quotient is placed in the operand QUO, the whole number remainder is placed in the operand REM, and no power flow is generated.

GREATER THAN A Greater Than B

References A and B are double precision numbers. If active, the contents of reference A are 2's complement compared to the contents of reference B. If reference A is greater than reference B, power flow is outputted; otherwise no power flow is generated.

TABLE MOVE **Table Move Key**

MENU DISPLAY	MNEMONIC FORMAT	VALID REFERENCE ENTRIES	VALID RANGES FOR DISPLAYED DATA
1 TABLE-TO-DEST	***** ***** CONST +[TABLE -TO-DEST LEN]+ ddddd -ddddd ddd	*****(TABLE) = 1-993 if I,O 1-993 if AI,AO If R 1-1023 *****(DEST) = 1-1009 if I,O 1-1009 if AI,AO If R 1-1024	ddddd (TABLE) = 0 to 65,535 -ddddd (DEST) = -32,768 to +32,767 ddd (LEN) = 255 maximum
2 SRC-TO-TABLE	***** ***** CONST +[SRC-TO-TABLE LEN]+ -ddddd ddddd ddd	*****(SRC) = 1-1009 if I,O 1-1009 if AI,AO If R 1-1024 *****(TABLE) = 1-993 if I,O 1-993 if AI,AO If R 1-1023	-ddddd (SRC) = -32,768 to +32,767 ddddd (TABLE) = 0 to 65,535 ddd (LEN) = 255 maximum
3 MOVE TABLE	***** ***** CONST +[TBL A MOVE TBL B LEN]+ hhhh hhhh ddd	*****(TBL A, TBL B) 1 - 1009 if I,O 1-1009 if AI,AO If R 1-1024	hhhh (TBL A, TBL B) = 0 to FFFF Hexadecimal ddd (LEN) = 255 maximum

TABLE-TO-DEST Move TABLE TO DESTination

The least significant half of the contents of the reference TABLE is used as a pointer (8-bit unsigned binary) to one of the words in the table. The most significant half of the reference TABLE is unaffected by the function. The number of words in the table is specified by the reference LEN; the word containing the pointer is not counted as a word of the table.

If active, the pointer is incremented by one. If the pointer is equal to the reference LEN, power flow is outputted, otherwise no power flow is generated. If the pointer is greater than the reference LEN, the pointer is set to one. If the reference TABLE is a register, the contents of the register whose address is equal to the address of TABLE plus the pointer is copied to the reference DEST. If the reference TABLE is I, O, AI, or AO, the contents of the word whose address is equal to the address of the reference TABLE plus 16 times the value of the pointer is copied to the reference DEST.

SRC-TO-TABLE Move SouRCe TO TABLE

The least significant half of the contents of the reference TABLE is a pointer (8-bit unsigned binary) to one of the words in the table. The most significant half of the reference TABLE is unaffected by the function. The number of words in the table is specified by the reference LEN; the word containing the pointer is not counted as a word of the table.

If active, the pointer is incremented by one. If the pointer is equal to the reference LEN, power flow is outputted, otherwise no power flow is generated. If the pointer is greater than the reference LEN, the pointer is set to one. If the reference TABLE is a register, the contents of the reference SRC are copied into the register whose address is equal to the address of TABLE plus the pointer. If the reference TABLE is I, O, AI, or AO, the contents of the reference SRC are copied to the word whose address is equal to the address of the reference TABLE plus 16 times the value of the pointer.

MOVE TABLE MOVE TABLE A to TABLE B

The reference TBL A and TBL B specify the starting addresses of 2 different tables. The number of words in each table is specified by the reference LEN.

If active, the entire contents of TBL A are copied into TBL B and the output is turned on. The function will copy TBL A into TBL B properly even if the tables overlap.

LIST List Key

MENU DISPLAY	MNEMONIC FORMAT	VALID REFERENCE ENTRIES	VALID RANGES FOR DISPLAYED DATA
1 ADD-TO-TOP	***** ***** CONST +[SRC ADD-TO-TOP LIST LEN]+ -ddddd ddddd ddd	*****(SRC) 1-1009 if I,O 1-1009 if AI,AO If R 1-1024 *****(LIST) 1-993 if I,O 1-993 if AI, AO If R = 1-1023	-ddddd (SRC) = -32,768 to +32,767 ddddd (LIST) = 0 to 65,535 ddd (LEN) = 255 maximum
2 REM-FM-BOT	***** ***** CONST +[LIST REM-FM-BOT DEST LEN]+ ddddd -ddddd ddd	*****(LIST) = 1-993 if I,O 1-993 if AI,AO If R 1-1023 *****(DEST) = 1-1009 if I,O 1-1009 if AI, AO If R 1-1024	ddddd (LIST) = 0 to 65,535 -ddddd (DEST) = -32,768 to +32,767 ddd (LEN) = 255 maximum
3 REM-FM-TOP	***** ***** CONST +[LIST REM-FM-TOP DEST LEN]+ ddddd -ddddd ddd	*****(LIST) 1-993 if I,O 1-993 if AI,AO If R 1-1023 *****(DEST) 1-1009 if I,O 1-1009 if AI, AO If R 1-1024	ddddd = 0 to 65,535 -ddddd = -32,768 to +32,767 ddd (LEN) = 255 maximum
4 SORT	***** ***** CONST +[LIST A SORT LIST B LEN]+ -ddddd -ddddd ddd	*****(R) 1-1024	-ddddd (LIST A, LIST B) -32,768 to +32,767 ddd (LEN) = 63 maximum

ADD-TO-TOP ADD TO TOP of List

The least significant half of the contents of the reference LIST is a pointer (8-bit unsigned binary) to the current bottom of the list. The most significant half of the reference LIST is unaffected by the function. The maximum length of the list is specified by the reference LEN; the word containing the pointer is not counted as a word of the list.

If active and the pointer is less than LEN, then each item in the list is moved to the next higher word address in the list, and the contents of the reference SRC are copied into the first word (top) of the list (the top of the list is the first word after the reference LIST), and the pointer is incremented by one. If the increment resulted in the pointer becoming equal to LEN power flow is outputted, otherwise no power flow is generated.

If active and the pointer is greater than or equal to LEN, then the list is full. Power flow is outputted, the pointer is set equal to LEN, and nothing is added to the list.

REM-FM-BOT REMove FroM BOTtom of List

The least significant half of the contents of the reference LIST is a pointer (8-bit unsigned binary) to the current bottom of the list. The most significant half of the reference LIST is unaffected by the function. The maximum length of the list is specified by the reference LEN; the word containing the pointer is not counted as a word of the list.

If active and the pointer is less than LEN, then the contents of the word at the bottom of the list (the bottom of the list is pointer value plus one word after the reference LIST) are copied into the reference DEST, and the pointer is decremented by one. If the decrement resulted in the pointer becoming equal to zero, power flow is outputted, otherwise no power flow is generated.

If active and the pointer is zero, then the list is empty; power flow is outputted, and the contents of reference DEST remain unchanged.

If active and the pointer is greater than LEN, then an illegal condition exists; power flow is outputted, and the contents of the reference DEST remain unchanged.

REM-FM-TOP REMove FroM TOP of List

The least significant half of the contents of the reference LIST is a pointer (8-bit unsigned binary) to the current bottom of the list. The most significant half of the reference LIST is unaffected by the function. The maximum length of the list is specified by the reference LEN; the word containing the pointer is not counted as a word of the list.

If active and the pointer is less than LEN, then the contents of the word at the top of the list (the top of the list is the word after the reference LIST) are copied into the reference DEST, each item in the list is moved up to the next lower word address in the list, and the pointer is decremented by one. If the decrement resulted in the pointer becoming equal to zero, power flow is outputted, otherwise no power flow is generated.

If active and the pointer is zero, then the list is empty; power flow is outputted, and the contents of reference DEST remain unchanged.

If active and the pointer is greater than LEN, then an illegal condition exists; power flow is outputted, and the contents of the reference DEST remain unchanged.

SORT SORT Ascending

LIST A and LIST B are references specifying the starting addresses of two lists of length specified by reference LEN.

If active, the contents of LIST A are rearranged by sorting the words of the list in ascending order (2's complement). LIST B is created with numerical values (1-LEN) indicating where the corresponding word of LIST A was located before the sort took place. Power flow is outputted whenever this function is active.

If inactive, no action takes place, all references remain unchanged, and no power flow is generated.

MATRX Matrix Key

MENU DISPLAY	MNEMONIC FORMAT	VALID REFERENCE ENTRIES	VALID RANGES FOR DISPLAYED DATA
1 AND	***** ***** ***** CONST +[A AND B = C LEN]+ hhhh hhhh hhhh ddd	***** = 1-1009 if I,O 1-1009 if AI,AO If R 1-1024	hhhh (A,B,C) = 0 to FFFF Hexadecimal ddd (LEN) = 255 maximum
2 IOR	***** ***** ***** CONST +[A IOR B = C LEN]+ hhhh hhhh hhhh ddd	***** = 1-1009 if I,O 1-1009 if AI,AO If R 1-1024	hhhh (A,B,C) = 0 to FFFF Hexadecimal ddd (LEN) = 255 maximum
3 EOR	***** ***** ***** CONST +[A EOR B = C LEN]+ hhhh hhhh hhhh ddd	***** = 1-1009 if I,O 1-1009 if AI,AO If R 1-1024	hhhh (A,B,C) = 0 to FFFF Hexadecimal ddd (LEN) = 255 maximum
4 INV	***** ***** CONST +[A INV B LEN]+ hhhh hhhh ddd	***** = 1-1009 if I,O 1-1009 if AI,AO If R 1-1024	hhhh (A,B) = 0 to FFFF Hexadecimal ddd (LEN) = 255 maximum
5 COMPARE	***** ***** ***** ***** CONST +[COMPARE INPUT REF MASK FAULT LEN]+ hhhh hhhh hhhh ddddd ddd	***** (INPUT,REF,MASK) = 1-1009 if I,O 1-1009 if AI,AO If R 1-1024 ***** (FAULT) = 1-1009 if I,O 1-1009 if AI,AO 1-1024 if R	hhhh (A,B,C) = 0 to FFFF Hexadecimal ddddd (D) = 0 to 65,535 ddd (LEN) = 255 maximum

AND Matrix AND

References A, B, and C specify the starting addresses of three matrices of length specified by reference LEN. The function works from low addresses to high addresses, and care must be exercised when using overlapping matrices.

If active, the contents of corresponding words of matrices A and B are logically AND-ed, and the result is placed in the corresponding word of matrix C. If the resulting matrix C consists of all zeros, power flow is outputted, otherwise no power flow is generated. Matrices A and B are unchanged.

IOR Matrix Inclusive OR

References A, B, and C specify the starting addresses of three matrices of length specified by reference LEN. The function works from low addresses to high addresses, and care must be exercised when using overlapping matrices.

If active, the contents of corresponding words of matrices A and B are logically inclusive OR-ed, and the result is placed in matrix C; power flow is outputted. Matrices A and B are unchanged.

EOR Matrix Exclusive OR

References A, B, and C specify the starting addresses of three matrices of length specified by reference LEN. The function works from low addresses to high addresses, and care must be exercised when using overlapping matrices.

If active, the contents of corresponding words of matrices A and B are logically exclusive OR-ed, and the result is placed in the corresponding word of matrix C. If the resulting matrix C consists of all zeros, power flow is outputted, otherwise no power flow is generated. Matrices A and B are unchanged.

INV Matrix INVert

References A and B specify the starting addresses of two matrices of length specified by reference LEN.

If active, the contents of each word of matrix A is logically inverted and stored in the corresponding word of matrix B, and power flow is outputted. The function will copy matrix A into matrix B properly even if the matrices overlap.

COMPARE Masked COMPARE

The references INPUT, REF (reference), and MASK are bit matrices of length specified by reference LEN X 16. The contents of reference FAULT are used as a pointer into the bit matrices, and to store the location of an unmasked miscompare.

If active, the contents of reference FAULT are incremented. If the resulting contents of FAULT are greater than the number specified by LEN X 16, then FAULT is set to 1. The function starts with the bit specified by the contents of reference FAULT and compares corresponding bits in the matrices INPUT and REF. After each compare the reference FAULT is incremented. If a miscompare occurs, the corresponding bit in the matrix MASK is tested to see if the miscompare has been masked (mask bit equal to one).

If the mask bit is set the comparison continues until an unmasked miscompare is found, or the end of the matrix is reached (contents of FAULT equal to (LEN X 16) + 1). If the end of the matrix is reached and no miscompare is found, no power flow is generated.

If the mask bit was not set (unmasked miscompare), power flow is outputted, the corresponding mask bit in the matrix MASK is set, and the execution of the function terminates. The contents of FAULT indicate the location of the unmasked miscompare.

BIT MATRX **Bit Matrix Key**

MENU DISPLAY	MNEMONIC FORMAT	VALID REFERENCE ENTRES	VALID RANGES FOR DISPLAYED DATA
1 BIT SET	***** ***** CONST +[BIT SET MATRIX LEN]+ ddddd hhhh ddd	***** = 1–1009 if I,O ***** = 1–1009 if AI, AO ***** = 1–1024 if R If BIT a Constant = 15 bits	ddddd = 0 to 65,535 ddddd (Constant) = 0 to 32,767 hhhh = 0 to FFFF Hexadecimal ddd = 255 maximum
2 BIT CLEAR	***** ***** CONST +[BIT CLEAR MATRIX LEN]+ ddddd hhhh ddd	***** = 1–1009 if I,O ***** = 1–1009 if AI, AO ***** = 1–1024 if R If BIT a Constant = 15 bits	ddddd = 0 to 65,535 ddddd (Constant) = 0 to 32,767 hhhh = 0 to FFFF Hexadecimal ddd = 255 maximum
3 SHIFT RT	***** ***** CONST +[SHIFT RT N MATRIX LEN]+ ddddd hhhh ddd	***** = 1–1009 if I,O ***** = 1–1009 if AI,AO ***** = 1–1024 if R If N a Constant = 15 bits	ddddd = 0 to 65,535 ddddd (Constant) = 0 to 32, 767 hhhh = 0 to FFFF Hexadecimal ddd = 255 maximum
4 SHIFT LEFT	***** ***** CONST +[SHIFT LEFT N MATRIX LEN]+ ddddd hhhh ddd	***** = 1–1009 if I,O ***** = 1–1009 if AI,AO ***** = 1–1024 if R If N a Constant = 15 bits	dddd = 0 to 65,535 ddddd (Constant) = 0 to 32,767 hhhh = 0 FFFF Hexadecimal ddd = 255 maximum

BIT SET Set/Sense Bit

If the function receives power flow, the bit set function will be performed; if not, the sense function will be performed. In this function MATRIX references a matrix consisting of LEN times 16 bits. The contents of the reference BIT point to a bit in the matrix.

If the function receives power flow and the reference BIT is within the matrix (1 to LEN × 16), then the bit in the matrix specified by BIT is set to a one, and the power flow is outputted. If the reference BIT is not within the matrix, the matrix does not change and no power flow is generated.

If no power flow is received, and the reference BIT is within the matrix, then the bit in the matrix specified by BIT is sensed. If the bit is a one, power flow will be outputted, if it is a zero, no power flow will be generated.

BIT CLEAR Clear/Sense Bit

If the function receives power flow, the bit clear function will be performed; if not, the sense function will be performed. In this function MATRIX references a matrix consisting of LEN times 16 bits. The contents of the reference BIT point to a bit in the matrix.

If the function receives power flow, and the reference BIT is within the matrix (1 to LEN × 16), then the bit in the matrix specified by BIT is cleared (set to a zero), and no power flow is generated. If the reference BIT is not within the matrix, the matrix does not change and no power flow is generated.

If no power flow is received, and the reference BIT is within the matrix, then the bit in the matrix specified by BIT is sensed. If the bit is a one, power flow will be outputted; if it is a zero, no power flow will be generated.

SHIFT RT Shift Right N Bits

This function implements a shift register of length (16 × LEN) bits. The contents of reference N indicate the number of shifts to be executed.

If active, and N is greater than zero and less than or equal to (16 × LEN), then all bits within the matrix are shifted the specified number of bits to the right (towards the least significant bit) within each word, and in the direction of decreasing addresses between words (i.e., the least significant bit of the word at address X is shifted into the most significant bit of the word at the address X-1). Zeros are shifted into the most significant bit of the highest addressed word of the matrix. The last bit shifted out of the least significant bit of the lowest addressed word in the matrix controls power flow. If it is a one, power flow is outputted; otherwise no power flow is generated.

If N is zero or greater than (16 × LEN), then the shift is not performed, and no power flow is generated.

SHIFT LEFT Shift Left N Bits

This function implements a shift register of length (16 × LEN) bits. The contents of reference N indicate the number of shifts to be executed.

If active, and N is greater than zero and less than or equal to (16 × LEN), then all bits within the matrix are shifted the specified number of bits to the left (towards the most significant bit) within each word, and in the direction of increasing addresses between words (i.e., the most significant bit of the word at address X is shifted into the least significant bit of the word at address X + 1). Zeros are shifted into the least significant bit of the lowest address word in the matrix. The last bit shifted out of the most significant bit of the highest addressed word controls power flow. If it is a one, power flow is outputted; otherwise no power flow is generated.

If N is zero or greater than (16 × LEN), then the shift is not performed, and no power flow is generated.

CON TROL **Control Key**

MENU DISPLAY	MNEMONIC FORMAT	VALID REFERENCE ENTRIES	VALID RANGES FOR DISPLAYED DATA
1 DO SUB	+[DO SUB CONST ***** N REPS]+ ddd ddddd	***** = 1–1009 if I,O 1–1009 if AI, AO 1–1024 if R	ddd = 1 to 16 ddddd = 0 to 65,535
2 RETURN	+[RETURN]+	None Required	Not Applicable
3 SUSPEND I/O	+[SUSPEND I/O]+	None Required	Not Applicable
4 DO I/O	+[DO I/O ***** ***** START END]+ ddddd ddddd	***** = 1 to 1024 if R	ddddd = 0 to 65,535 If START or END a Constant then ddddd = 1 to 1000
5 STATUS	***** +[STATUS]+ hhhh	***** = 1 to 1009 if I,O 1 to 1009 if AI, AO 1 to 1024 if R	hhhh = 0 to FFFF Hexadecimal

DO SUB DO SUBroutine

If active, the subroutine indicated by reference N is executed as many consecutive times as is indicated by the 8 least significant bits of the reference REPS (repetitions). The most significant half of the reference REPS is used by the function to count the number of executions. This is set to zero before the first execution and incremented by 1 at the end of each execution and then tested for equality with the least significant half. When the least significant byte and most significant byte of the reference REPS are equal, the most significant byte of REPS is set to zero and control is returned to the main program (refer to the RETURN function). Execution commences at the point following the last function executed prior to the transfer of control to the subroutine. User interrupts are disabled while this function is being executed. Subroutines must not contain imbedded DO SUB functions.

NOTE

Normal subroutines should not modify the contents of the REPS reference. Unexpectedly long scan times may result that exceed the hardware watchdog timer if this reference is modified during a subroutine.

RETURN

This function is executed unconditionally. When encountered, it returns control to the main program. Execution commences at the point following the last function executed prior to the transfer of control to the subroutine. RETURN functions must not be embedded in the main program.

SUSPEND I/O

If active, I/O scanning will not occur. Each physical output will retain the state it was in when the SUSPEND I/O function was encountered, and the input table will not be updated with data from physical input devices, unless the program contains active DO I/O functions. The input and output tables will be updated by the results of program operation.

If inactive, normal I/O scanning will occur.

DO I/O

If active, and the contents of START and END are valid, the range of input and output points specified by the contents of the references START (rounded to the nearest lower byte boundary) and END (rounded to the nearest upper byte boundary) are serviced immediately and power flow outputted.

If active, and the contents of START and/or END are invalid, no power flow is generated. All of the following conditions must be met:

1. The contents of START must be in the range of 1 to 1000.
2. The contents of END must be in the range of 1 to 1000.
3. The contents of START must be less than or equal to the contents of END.

STATUS

This function may be used to enable or disable DPU (Data Processor Unit) and/or CCM (Communication Control Module) windows (by writing to a status indication byte in the scratch pad), or to determine the status of the DPU and/or CCM module (by reading the status indication byte). The content of the fifth bit of the reference is used to indicate DPU status (10H), and the content of the sixth bit of the reference is used to indicate CCM status (20H).

Reference Contents (after AND operation if writing to scratch pad)	DPU Window	CCM Window	Alarm #2 (Minor Alarms)
0 H	Enabled	Enabled	Off
10 H	Disabled	Enabled	On
20 H	Enabled	Disabled	On
30 H	Disabled	Disabled	On

If the function receives power flow, the contents of the reference are logically ANDed with 30 H, and the result is written to the status byte of the scratch pad. The DPU and CCM windows and ALARM #2 are affected as shown in the chart.

If the function does not receive power flow, the contents of the upper byte of the reference is zeroed, and the contents of the status byte of the scratch pad is written to the lower byte of the reference. The contents of the lower byte of the reference can be used to determine the status of the DPU, CCM, and ALARM #2 as shown (disabled indicates an error condition).

CHAPTER IV

Ladder Diagram Programming

This chapter describes the tables and displays used when programming a Series Six system. The basic information is provided which will allow you to enter ladder diagrams by use of the relay functions. A detailed step-by-step procedure is used when describing the entering of each function. This method allows the user with little or no programming experience to understand the sequence of steps required for the entry of a program.

INTRODUCTION TO PROGRAMMING

The Series Six family of programmable controllers uses the ladder diagram approach to programming. A visual display allows the programmer to observe each step of the program while entering the rungs of logic required by the program. The program can be entered, corrected, and verified before transferring the program to a CPU. When the program is verified to be correct it can then be transferred to the CPU for storage in the user memory. Figure 1 is a typical CRT display of a section of a ladder diagram.

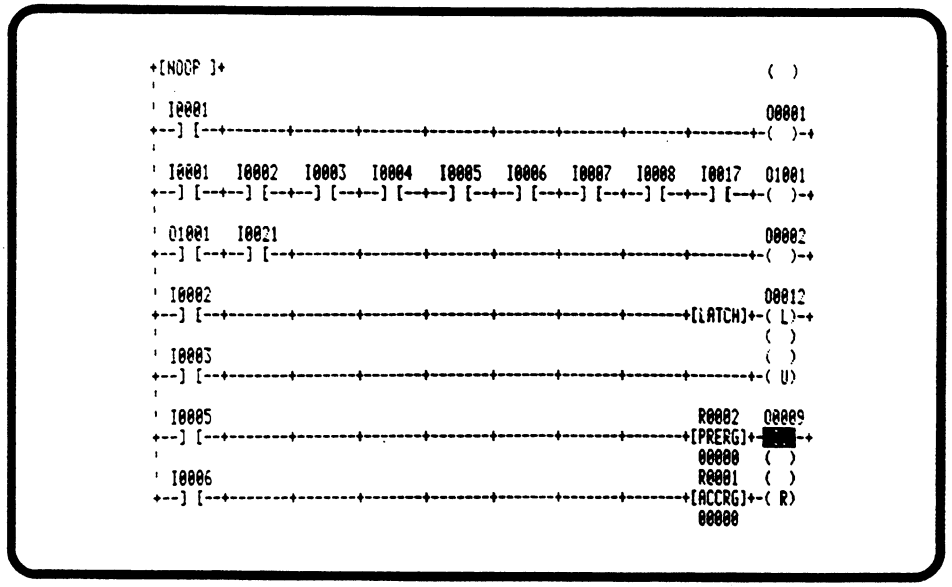


FIGURE 1
Ladder Diagram Display

NOTE

Programming is accomplished by entering rungs of logic made up of horizontal lines using normally-open and normally-closed contacts, relay coils, timers, counters, latches, and one-shots. The capabilities of this function set are enhanced by mnemonic functions available as a basic set and an extended set. Mnemonic programming allows data manipulation for more flexible and precise control of an application. A mnemonic is a function using letters which represent a command or series of commands to perform a particular operation.

The various functions can be grouped together as shown in Table 1.

TABLE 1—Basic and Extended Functional Groups

GROUP	BASIC	EXTENDED*
Relay	Relay Ladder Logic Timing (1.0, 0.1, 0.01 sec.) Counting (up, down) Latch, One-shot	Auxiliary I/O References
Arithmetic	Addition Subtraction Compare Shift } Unsigned Binary	Double Precision Addition, Subtraction Multiply, Divide Greater Than } signed 2's complement
Control	Master Control Relay Skip	Do Subroutine, Do I/O, Suspend I/O, Return, Status
Move/ Convert	Data Moves I/O Table to Register Table Register Table to I/O Table Binary to BCD BCD to Binary	Table to Destination Source to Table, Move Table Move Right 8 Bits Move Left 8 Bits Block Move, Move A to B
Communications Request	Serial Communication Request Data Processor Request	
Matrix		AND Inclusive & Exclusive OR Invert Masked Compare Set/Sense Clear/Sense Shift Right Shift Left } Bit
List		Add to Top Remove from Bottom Remove from Top Sort
Miscellaneous	End of Sweep NO OP	

*Extended Set includes Basic Functions.

Many of the functions in the program can reference the Input and Output status tables and register memory in the CPU. Certain hardware (CPU) tables are also used by the functions. These tables and the various memories are referred to as CPU resources and are described below. All tables are inherently retentative upon power failure.

INPUT TABLE

There are 1024 inputs available numbered for reference from 1 to 1024. The Input table is physically located on the Internal Memory module and is a group of consecutive memory (storage) locations. Inputs 1009 to 1024 cannot be connected to physical devices; however, they are available for internal storage and can be used for program references.

Inputs are used to represent contacts activated by devices such as pushbuttons, limit switches, analog sensors or relay contacts. Inputs also represent contacts operated by output coils of program functions such as relays, counters, timers, latches and one-shots. Many of the mnemonic functions also operate on the Input table.

OUTPUT TABLE

There are 1024 outputs available numbered for reference from 1 to 1024. The Output table is a group of consecutive storage locations located on the Internal Memory module. Outputs 1001 to 1024 cannot be connected to physical devices; however, they are available for internal storage and are used for program references. Outputs are used to represent the coils of relays, counters, timers, latches, and one-shots. Various mnemonic functions operate on the Output table.

TRANSITION TABLE

The Transition table is a hardware table divided into 2 parts. The 1024 bits in half of the table correspond to the output bits in the Output status table. Counter and one-shot functions use these bits to store the state of the enabling contact string. With this information, the functions sense the off-to-on transition which either enables counting or fires a one-shot.

The other half of the Transition table is used by the Auxiliary counter and one-shot functions with the extended functions. These 1024 bits correspond to the lowest 64 words (× 16 bits) of the Register memory (which is the auxiliary output table).

The only condition under which the Transition table can be written into or read by the Program Development Terminal is when the Transition table is cleared from the Program Development Terminal. This action will clear parity errors.

OVERRIDE TABLE

The Override table is divided into 2 parts, the Input Override table and the Output Override table. The Override tables are used by the following functions: relays, counters, timers, latches and one-shots. Mnemonic functions ignore the Override table when operating on the Input and Output tables. When a bit in the Input or Output table is overridden, it is maintained in the state (on or off) it was in when the override was applied until the override is removed or its state is changed by toggling ON/OFF followed by ENTER (overridden bits can be changed by mnemonic functions).

Inputs and outputs may be overridden from the Program Development Terminal. By use of overrides in conjunction with the ON/OFF key, bits in the Input or Output table can be forced to a desired state. This is a very useful tool during testing and debugging of programs.

NOTE

Auxiliary inputs and outputs cannot be overridden.

REGISTER MEMORY

The Register memory consists of 1024 consecutive 16-bit storage locations in the model 600 and 6000 CPUs or 256 consecutive 16-bit locations in the model 60 CPU. Certain mnemonic functions refer to locations in the Register memory and use those locations for data manipulation when specifying lists, storage for results of arithmetic operations, data tables, data moves and matrices.

USER PROGRAM MEMORY (LOGIC MEMORY)

The Logic memory is used for storage of the User program. The maximum Logic memory available is 32,767 (32K) words in a model 6000 CPU. A model 600 CPU can have a maximum of 8,192 (8K) words and the maximum Logic memory available in a model 60 CPU is 2048 (2K) words. A word is 2 eight bit bytes (16 bits total) plus 2 parity bits.

Programs are entered into the Program Development Terminal and transferred to a CPU, where they are entered into the Logic memory in consecutive locations beginning with the first location.

Figure 2 illustrates the CPU resources as described in the previous paragraph.

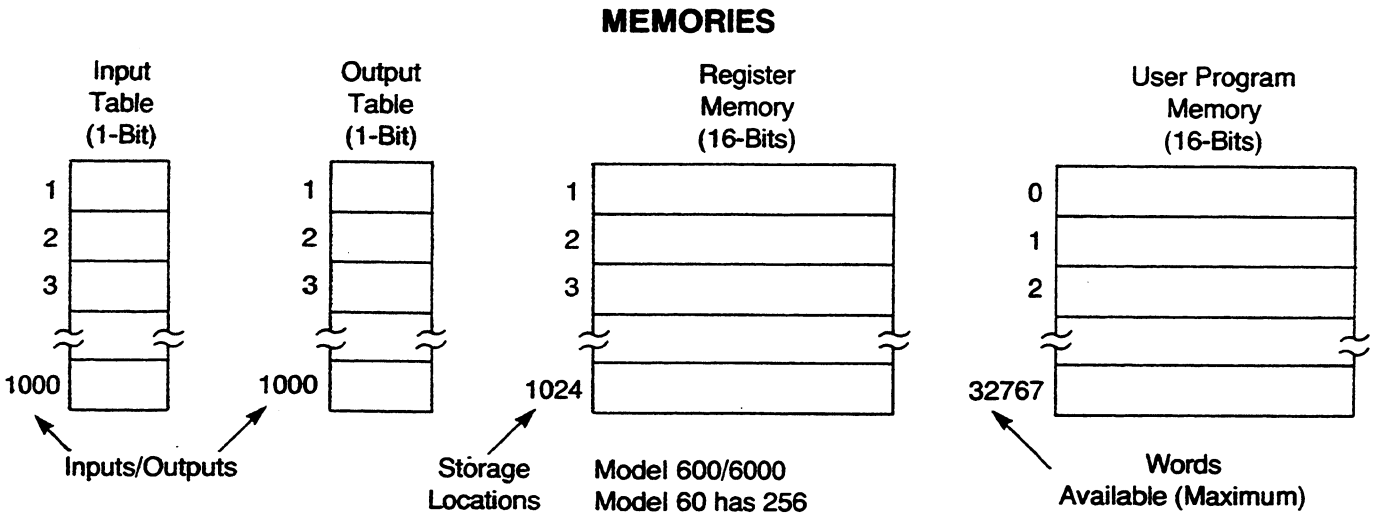


FIGURE 2
CPU Resources

CRT DISPLAYS

There are several displays that are viewed on the CRT which the user should become familiar with. These displays are the Ladder Diagram (Program) display, the work area on the CRT, the Supervisor display and the Scratch Pad display.

LADDER DIAGRAM DISPLAY

The Ladder Diagram display provides a visual representation of the User program. This display is accessed by selecting the item DISPLAY PROGRAM from the Supervisor display. This display is useful when entering, editing or monitoring programs. The Ladder Diagram display is built from symbols and mnemonic functions entered by the Operator from the keyboard.

The layout of the CRT screen for building a ladder diagram can be thought of as a 10 × 8 grid (10 squares horizontal, 8 vertical). Position 1 of the first grid square is on the left with position 10 reserved for an output coil. Figure 3 is an illustration of the layout of the screen for a ladder diagram display.

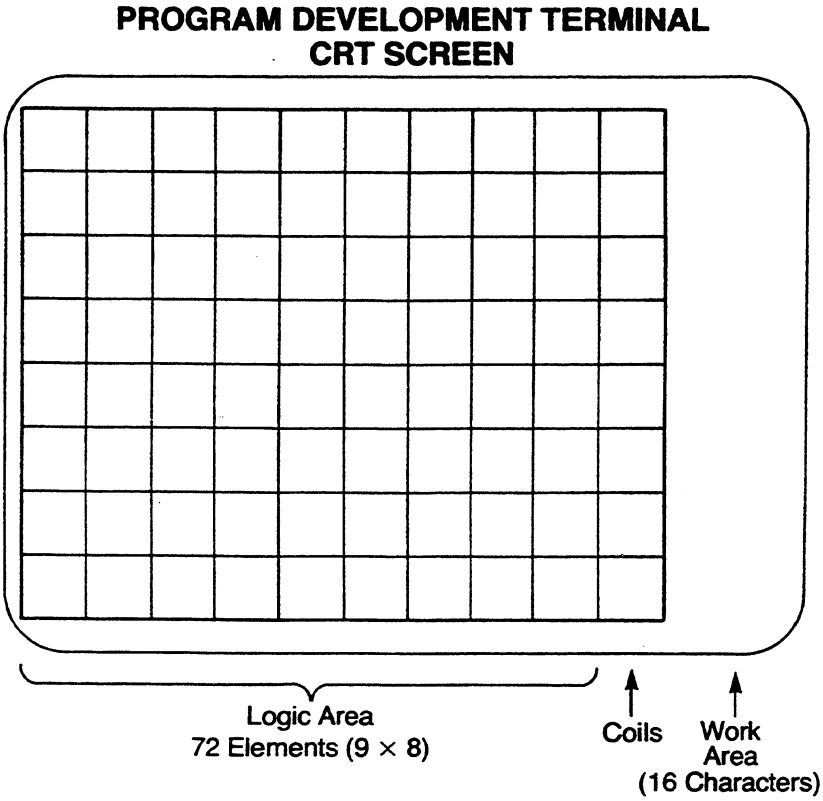


FIGURE 3

CRT Ladder Diagram Display Area

Each grid square can contain 1 horizontal and 1 vertical segment. By combining these segments into a ladder network a rung of relay logic is formed. One or more lines form a rung of logic and a series of completed rungs form a complete ladder diagram or program.

Ladder diagrams are created and modified in the EDIT mode. When the program is displayed the EDIT mode is entered by depressing {**EDIT**} or {**INSERT**}. The cursor for the EDIT mode is a reverse video block consisting of 8 horizontal and 3 vertical characters which outline the contact structure. When entering the EDIT mode the cursor is on the top left grid position. The cursor can be moved around the screen by using the cursor control keys.

Any of the 80 positions on the edit grid may have its contents changed by positioning the cursor on the grid and depressing the desired line segment and reference replacement followed by {**ENTER**}. Any vertical or horizontal line segment entered replaces the former segment. If only one segment is entered, only one is replaced.

When entering contacts or mnemonics as a part of the ladder diagram, the result of most of the keystrokes (menu entries, references, etc.) is displayed in the work area on the right side of the screen. This allows the operator to verify that the key entry is correct before completing the contact or mnemonic being entered.

WORK AREA

The work area is a 16 character wide area located on the right side of the CRT display. The first 6 lines are always displayed and contain status information about the CPU and Program Development Terminal. Figure 4 shows a typical CRT display with the work area highlighted. The status information lines are in the upper right portion of the display. Table 2 describes the status information.

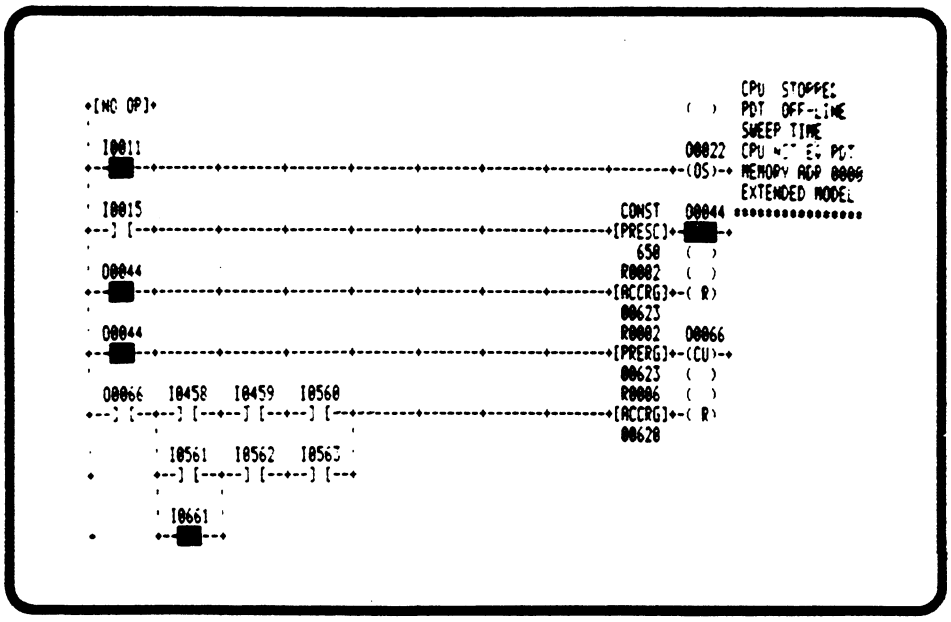


FIGURE 4
Work Area Location

TABLE 2—Work Area Status Display Description

STATUS DISPLAY	DESCRIPTION
CPU:	Operational status of the CPU as contained in the Scratch Pad. Status can be RUN ENABLED, RUN DISABLED OR STOPPED.
PDT:	Visual indication of the Program Development Terminal mode as selected by keyswitch position. Mode will be MONITOR, OFF-LINE or ON-LINE.
SWEEP TIME:	The sweep time of the CPU in milliseconds. (Sweep time not displayed when Off-Line.)
CPU EQ PDT or CPU NOT EQ PDT	Indicates whether the program in CPU memory is the same as in the PDT memory. Status will be either EQ (equal) or NOT EQ (not equal). The CPU EQ PDT status must be displayed before on-line one word changes can be made to an operating program.
MEMORY ADR:	The starting address of the top rung of the displayed program is shown in hexadecimal.
BASIC MODEL or EXTENDED MODEL	Information contained in the Scratch Pad memory defining the function set present in the CPU is displayed here. Will be either BASIC MODEL (Basic Function Set) or EXTENDED MODEL (Extended Function Set) When On-Line or Monitor mode. User defined when Off-Line. Default is BASIC MODEL.

In addition to the status display information described in Table 2, the work area is used for the display of data including key entry, menus for mnemonic functions, error messages, busy status and operator prompts.

A BUSY message indicates that the previously requested operation is being performed by the PDT. An example of this is when a SEARCH is in progress for a reference in a program. In this case the BUSY status message is displayed until the SEARCH is complete.

As described in Chapter 3, key depression of each key in the Basic mnemonic or Extended mnemonic key group will display a menu of mnemonic functions which allows selection of the required function by the Programmer. These menus are displayed in the lower portion of the work area.

The error messages appear in the lower portion of the work area and indicate that an error has been made in programming or operating of the Program Development Terminal. Table 3 summarizes most of the Series Six Error Messages. An error message can be removed from the work area by depressing {CLEAR}.

TABLE 3—Error Messages

CATEGORY	DISPLAY MESSAGE	CAUSE OF MESSAGE
Utility	COMMUNICATION ERROR	Data error.
	TAPE NOT EQ MEMORY	Tape contents do not match PDT.
	DEVICE NOT READY	Printer/Cassette not Ready.
	ILLEGAL RUNG	Illegal Rung.
	NOT OFF-LINE	Must be off-line for cassette and printer functions.
Scratch Pad	ILLEGAL NUMBER	1. Illegal No. for CPU ID. 2. Illegal No. for memory size.
	EXTENDED INSTRUCTION	Extended functions found in memory when not allowed.
Display	NOT A COIL	Can't program bit when cursor not at output coil.
	ILLEGAL FUNCTION	Invalid output function found in memory.
	START OF MEMORY	First rung in memory is displayed, can't display a previous rung.
	END OF MEMORY	Last rung in memory is displayed, can't display the next rung.
	INVALID REFERENCE	Have extra operand or not enough operands for mnemonic.
	INVALID OP CODE	Unrecognizable mnemonic found in memory.
	ILLEGAL RUNG	Incorrect function sequence found in memory.
Single Word Changes	CPU NOT EQ PDT	Can't make single word change while CPU is not equal to PDT in on-line mode.
	ILLEGAL REPLACE	1. To change output function can only interchange relay and one-shot or timers and counters. 2. Line segment entered on position with no line segment.
	NOT A COIL	Cannot enter I and number for output coil.
	INVALID REFERENCE	1. Invalid reference for line segment input or mnemonic. 2. Illegal number for reference.
	NOT EDIT MODE	Cannot change mnemonic while in display mode.
Supervisor Menu	ILLEGAL NUMBER	1. Illegal number for "Number of Rungs." 2. Illegal number for "Start Output No." 3. Illegal number for Cursor. 4. Illegal number for status word.
	NO AUX. TABLE	Cannot display Auxiliary I/O tables with no extended functions.
	CURSOR NOT ASSIGNED	1. No cursor, so cannot change base. 2. Cannot move cursor if no cursor is displayed.
	ILLEGAL FUNCTION	Cannot change data in Register table to binary.
	NOT BINARY	Cannot program bit if base is not binary.
	NOT DECIMAL OR HEX	Cannot program word if base is binary.

CATEGORY	DISPLAY MESSAGE	CAUSE OF MESSAGE
Edit Mode	NO COIL	Output coil function not complete before leaving EDIT Mode.
	ILLEGAL RUNG	1. Nothing entered but output function. 2. Illegal Rung—cannot leave EDIT mode (most cases have the line segment in error blinking. 3. LATCH or OS with PRESET or ACC. 4. Timer or counter with LATCH Mnemonic.
	INVALID POSITION	1. Cannot enter mnemonic in position 10. 2. Cannot enter line segment in position 10.
	INVALID NUMBER	Number entered for mnemonic menu is too large.
	NOT ENOUGH ROOM IN LINE	A mnemonic takes more than one screen position and not enough room on the line with current cursor position.
	EXCEEDS MEM SIZE	Not enough room in memory to insert rung.
	INCOMPLETE ELEMENT	1. Cannot move cursor until contact is complete. 2. Entering an output coil number with no output function assiged. 3. Rung not complete—cannot leave EDIT. 4. Preset without ACC or ACC without Preset entered in rung, cannot leave EDIT.
	ONE COIL PER RUNG	Output function has already been entered at different screen location.
	REQUIRES DOUBLE LINE	Output function in either LATCH, ONE-SHOT, TIMER, or COUNTER rung is not double line.
	ILLEGAL FUNCTION	Double line of logic with output function of relay or one-shot.
	INVALID REFERENCE	Number entered for output coil is too large.
	MUST BE IN POSITION 9	Cannot enter PRESET or ACC in position other than 9.
Miscellaneous	RAM ERROR	Bad RAM found on power-up.
	PROM ERROR	PROM CRC does not match calculated CRC-Bad PROM.
	ILLEGAL IN MONITOR MODE	No editing or programming allowed in monitor mode.
	INVALID ENTRY	Illegal key sequence entered.
	MUST BE IN POSITION 10	Displayed if user attempts to enter an output in a location other than 10.

SUPERVISOR DISPLAY

When the Program Development Terminal is initially turned on and the PROM INTEGRITY CHECK has been completed successfully, the Supervisor display will appear in the center of the screen. The Supervisor display provides the operator with a means of selecting a particular operation or action.

The Supervisor display can also be called up by depressing the key sequence {**SUPERVISOR**}, {**ENTER**}. Figure 5 shows the Supervisor display as it appears on the CRT. The cursor is a reverse video rectangle which can be positioned next to the desired action by use of the ↑ and ↓ keys. Unless stated otherwise in the following description of each item in the display, depressing {**ENTER**} causes the execution of the item selected.

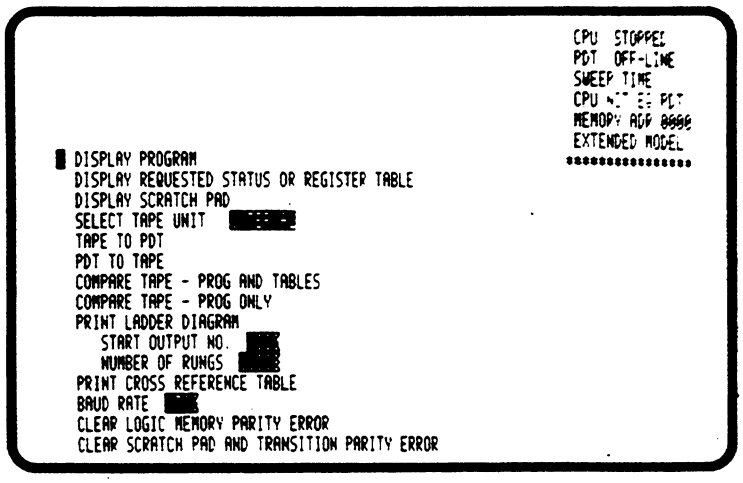


FIGURE 5
Supervisor Display

DISPLAY PROGRAM

The item DISPLAY PROGRAM must be selected whenever a program is to be entered, modified or monitored. When DISPLAY PROGRAM is requested for the first time (with no programs previously entered) 2 End-of-Sweeps are displayed at the top of the screen. On power-up, the PDT memory is filled with End-of-Sweep op-codes; however, only the first 2 are displayed.

The PDT remembers which rungs were displayed the last time the program display was on the screen and will return to that point when DISPLAY PROGRAM is invoked. When a tape is read or a CPU to PDT transfer is made, the beginning of the program will be displayed.

The ladder diagram display on the screen contains as many rungs as can fit on the screen. There will be no incomplete rungs displayed.

While a ladder diagram is displayed on the screen, information can be transferred between the Program Development Terminal and a CPU by using the key sequence {**PDT → CPU**}, {**ENTER**} or {**CPU → PDT**}, {**ENTER**}. In either case after ENTER is depressed, a message is displayed in the work area, TO EXECUTE TYPE SHIFT-ENTER. {**SHIFT**} and {**ENTER**} must be depressed simultaneously to complete the requested transfer.

Once a ladder diagram display is on the screen any portion of the program in memory can be requested and displayed. A specific coil and its associated logic can be requested by depressing the key sequence:

{**0**}, { }, {**SEARCH**}
└ Desired output (coil) number

The requested rung is displayed at the top of the screen followed by as many complete rungs as will fit. If the requested output is an auxiliary output, {**A**} must precede the {**0**} key.

Depressing the keys {**PREV**} or {**NEXT**} will scroll the display one rung backward or forward, respectively. To modify a rung it must be at the top of the screen. When the rung is scrolled to the top of the display, depressing {**INSERT**} will put the PDT in the EDIT mode, thereby allowing entry of a new rung, which will be inserted below the rung at the top of the screen or allowing modification of an existing rung. Depressing {**EDIT**} will put the PDT in the EDIT mode, allowing the rung at the top of the screen to be modified.

All references to a specific input, output or mnemonic can be located and displayed by use of the SEARCH key. Refer to the discussion of the SEARCH key in Chapter 3 for a detailed description of the use of this key.

If the requested rung cannot be displayed in its entirety on one display on the screen, an error message will appear in the work area. This condition cannot occur with programs created by the Program Development Terminal as rules for creation and modification of programs limits them to logic which is displayable on the screen.

Page 4.29 discusses the construction of ladder diagrams.

DISPLAY REQUESTED STATUS OR REGISTER TABLE

When selected, this function allows the Operator to specify an input status, output status or a register table for display on the screen. The format for selection of one of these tables is:

{ }, { }, { }, { }, {ENTER}
- { } — desired display base (optional)
- { } — number between 1-1024 (optional)
- { } — I, O or R
- { } — A (if an auxiliary I or O table)

The first parameter is required only if an auxiliary input or output table is to be displayed. The second parameter (I, O or R) specifies selection of the desired table. If a reference number is not specified the requested table will begin with input or output 0001 or register 0001 and no cursor will be displayed. When a specific input, output or register is to be displayed and highlighted by the cursor a reference number must be specified as the third parameter.

The desired display base for the entire table can be selected by depressing the {**SHIFT**} key (hold down) and either {**BIN**}, {**HEX**}, {**DEC**}, {±**DEC**}, or {**DP**} (BIN not valid in Register table, DP not valid in Input or Output table).

When the input or output table is displayed, the entire table is displayed in binary groups (default display is binary) of 8 bits (1 byte). Each bit is displayed as a 1 (on) or a 0 (off) (binary format). If a specific input or output number is requested the cursor (a reverse video block) will be displayed positioned on that bit. Figure 6 is an example of a status table display. In the example Output 0001 was specified and the cursor is displayed on that bit. Output 0001 is on the upper right side of the display. The address of the bit that the cursor is on is displayed at the top of the screen next to the display heading. The base of the word in the Input or Output table that the cursor is on (cursor is on the low byte of the word) may be changed.

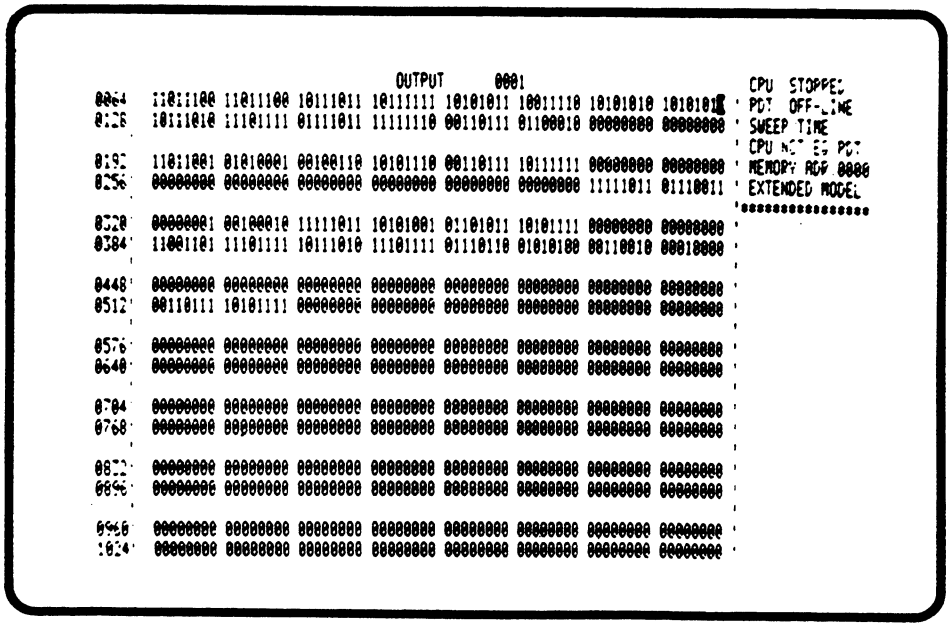


FIGURE 6
Output Status Table Display

The default display for registers is unsigned decimal. Register 0001 is in the upper right side of the display. When a specific register is requested, the register has the cursor (a reverse video block) positioned on that register. Figure 7 is an example of a register table display. The register memory is not displayed in its entirety, a group of 128 registers is displayed at a time.

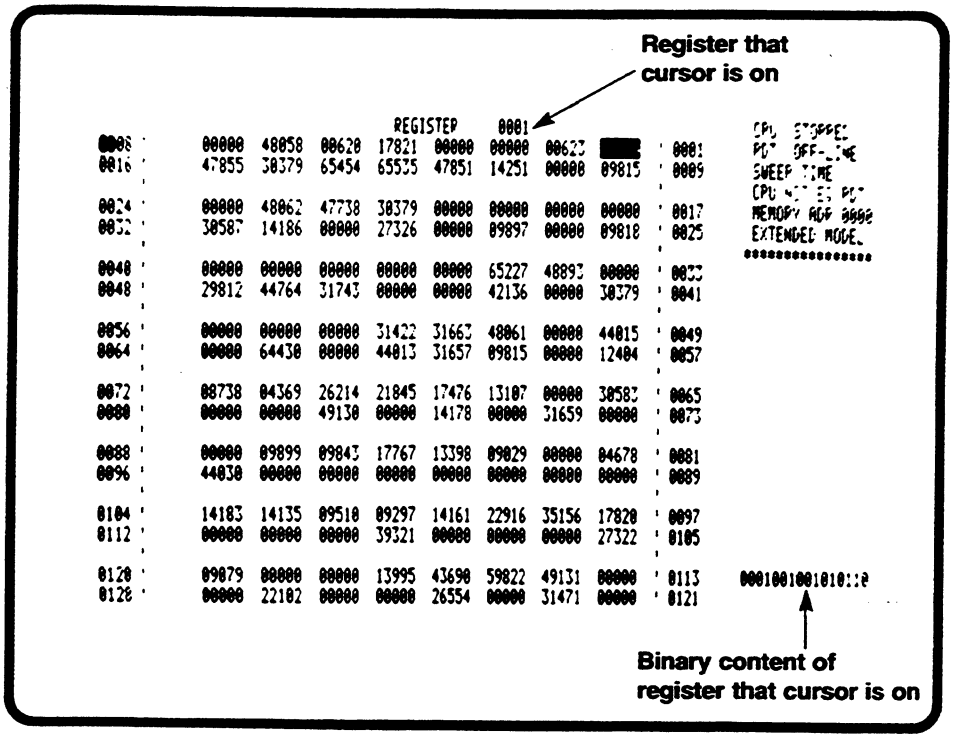


FIGURE 7
Register Table Display

The position of the cursor allows the value of the byte or word that the cursor is on to be changed to a base value other than the default value. Table formats and base changes are described more fully in the following paragraphs.

Table Display Format

A status or register table can be displayed with a number format other than the default format by specifying the desired format when selecting a table from the Supervisor menu or from another table. A table can be displayed in any of its legal number formats.

To display a table in the desired format add the appropriate shifted key to the sequence for selecting that table. As an example, if the Input status table is to be displayed in its entirety in the hexadecimal format rather than the binary default format, the following key sequence would be entered.

{I}, **{reference number}**, {SHIFT} and {HEX}, {ENTER}

The display would then appear on the screen in the format shown in Figure 8.

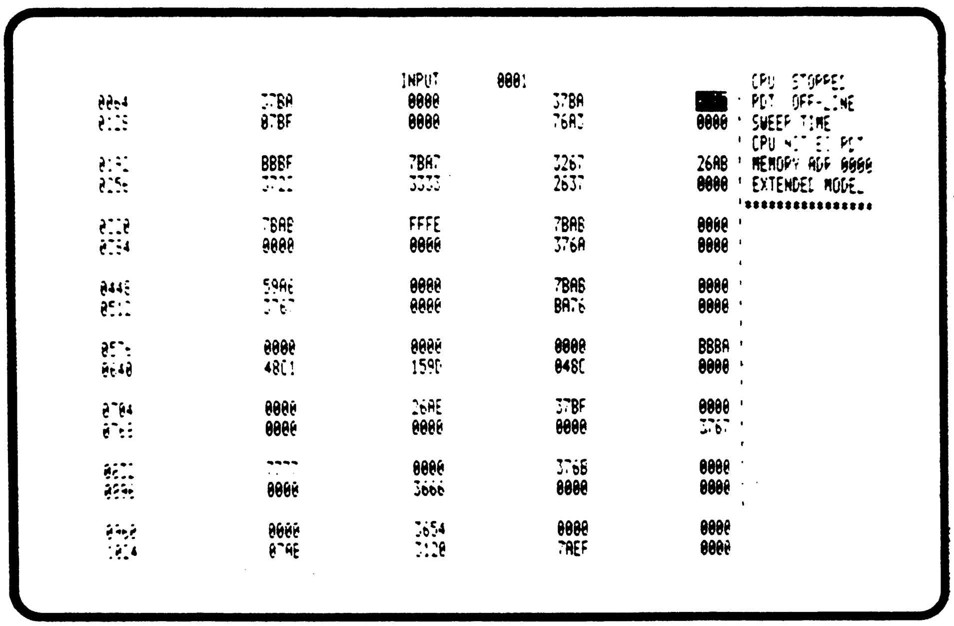


FIGURE 8
Input Status Table Hexadecimal Format

Auxiliary Status Table

The auxiliary inputs and outputs provide the capability of having an additional 1000 inputs and 1000 outputs in a model 6000 CPU. The auxiliary I/O status tables are located in registers 1 to 128. Registers 1 to 64 are the auxiliary output tables and registers 65 to 128 are the auxiliary input tables. Register 1 low order (right end) bit controls auxiliary output number 1 (AO0001); register 65 low order (right end) bit controls auxiliary input number 1 (AI0001).

To use the auxiliary status tables the item INSTRUCTION SET in the Scratch Pad display must be set to EXTENDED.

An attempt to program an auxiliary reference, search for an auxiliary reference or display an auxiliary status table with INSTRUCTION SET toggled to BASIC will cause an error message:

INVALID
REFERENCE TYPE or INVALID ENTRY or NO AUX TABLE

to be displayed in the work area.

Programming of Auxiliary References

References to the auxiliary I/O tables are programmed in the same manner as references to the main I/O tables, except that the A (Auxiliary) key must be included in the key sequence for the I/O location referenced. Any CPU with Extended functions can use auxiliary references.

Displaying the Auxiliary Status Tables

The auxiliary input and output status tables are selected from the Supervisor display menu, DISPLAY REQUESTED STATUS OR REGISTER TABLE or from another table display in the same manner as the main I/O tables, except that the A (Auxiliary) key must be included in the key sequence.

The format of the display for the auxiliary I/O tables is the same as the display of the main I/O tables, except that the table headings are AUXILIARY INPUT or AUXILIARY OUTPUT as shown in Figure 9.

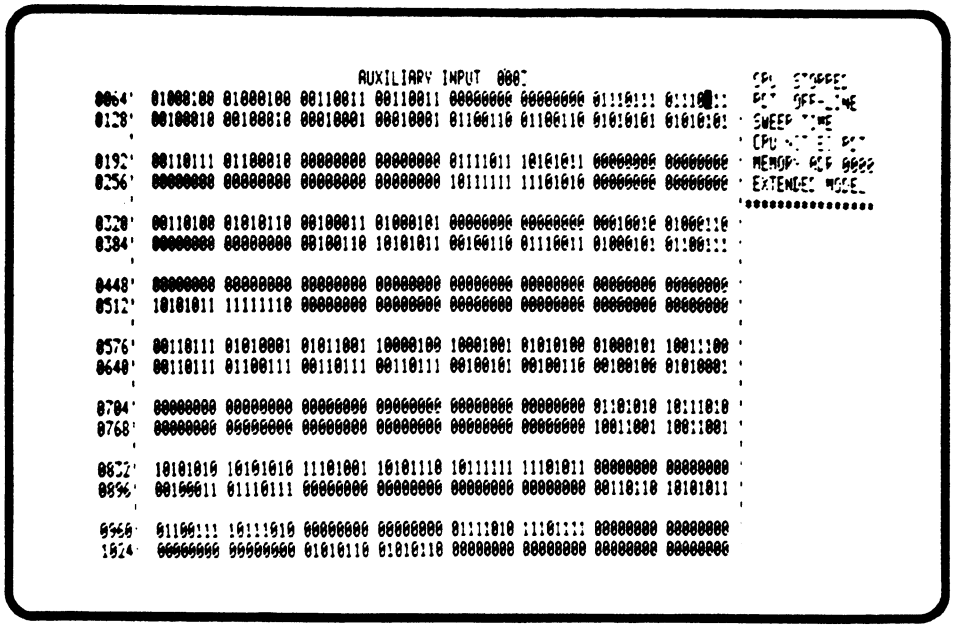


FIGURE 9
Auxiliary Table Display

Changing Base of Values—Status Tables

The default value of the status and auxiliary status tables is binary. The format of the display can be changed by using the ±DEC, DEC and HEX keys to signed decimal, decimal or hexadecimal, respectively. The base of an entire table can be changed on even or odd byte boundaries depending on how the cursor is specified.

NOTE

If an attempt is made to change the display to double precision in a status table an error message, ILLEGAL FUNCTION will be displayed in the work area.

The procedure for changing the base of an individual word in a table is as follows:

1. Request a bit of the byte to be changed.
2. The cursor will appear as a blinking reverse video display and will be on the bit requested.
3. The word containing that bit is eligible for change.
4. Depress the key sequence {SHIFT} and {DEC}, {±DEC} or {HEX}.
5. The requested base change will now appear on the display for the word containing the cursor. The cursor is always assumed to be on the low byte of a word.

In a system with the Extended function set, use of the ±DEC, DEC or HEX keys for a base change will change the display of the word (16 bits) located by the cursor to the desired base.

The bytes of the status table affected by the change of base keys are the byte the cursor is on and the byte with the next higher address. The byte the cursor is on is the least significant byte of the word to be changed. The word is displayed right justified in the location of the byte that the cursor is on.

To change the base of a value in a status or auxiliary status table that is displayed on the screen in a system with the Extended function set, proceed exactly as for the main status tables.

NOTE

Auxiliary table values can only be changed on even word boundaries instead of on byte boundaries, as in the main status tables, since the auxiliary tables are actually located in 16-bit registers.

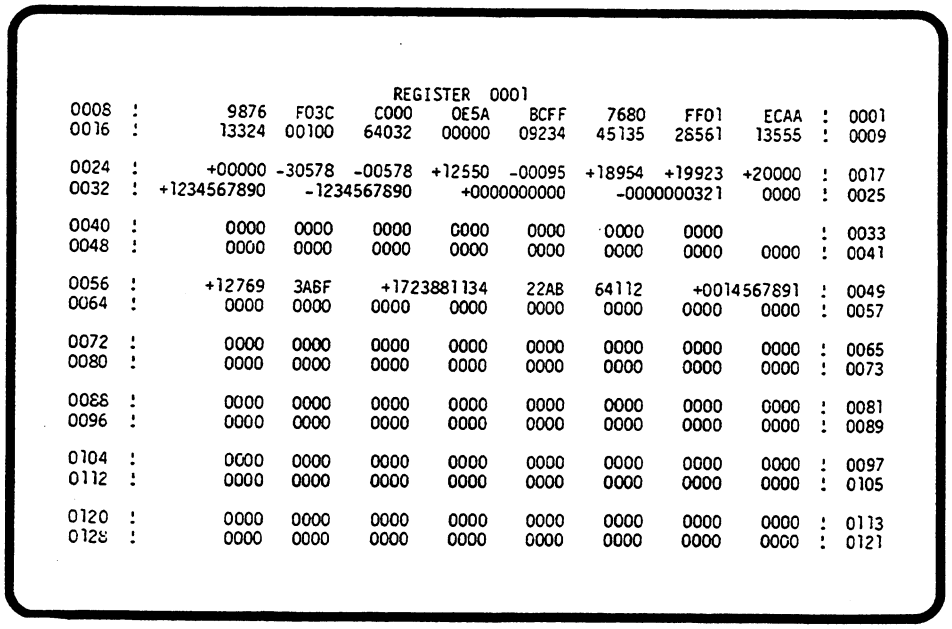
Register Display Format

The format used to display the registers is illustrated in Figure 10. The range and format of each number system is shown in the table below. The default display for the register display is unsigned decimal.

TABLE 4—Number Formats

TYPE OF NUMBER	RANGE OF VALUES
Unsigned Decimal	0000 to 65535
Hexadecimal	0000 to FFFF
Signed decimal Word (2's complement)	−32768 to +32767
Signed Decimal, Double Precision (2's complement)	−2147483648 to +2147483647

The illustration in figure 10 shows each of the 4 possible register table display formats for the Series Six. Formats can be mixed in the display. They are shown in the same format in each row for illustration purposes.



REGISTER 0001

0008	:	9876	F03C	C000	0E5A	BCFF	7680	FF01	ECAA	: 0001
0016	:	13324	00100	64032	00000	09234	45135	28561	13555	: 0009
0024	:	+00000	-30578	-00578	+12550	-00095	+18954	+19923	+20000	: 0017
0032	:	+1234567890		-1234567890		+0000000000		-0000000321	0000	: 0025
0040	:	0000	0000	0000	0000	0000	0000	0000		: 0033
0048	:	0000	0000	0000	0000	0000	0000	0000	0000	: 0041
0056	:	+12769	3ABF	+1723881134		22AB	64112	+0014567891		: 0049
0064	:	0000	0000	0000	0000	0000	0000	0000	0000	: 0057
0072	:	0000	0000	0000	0000	0000	0000	0000	0000	: 0065
0080	:	0000	0000	0000	0000	0000	0000	0000	0000	: 0073
0088	:	0000	0000	0000	0000	0000	0000	0000	0000	: 0081
0096	:	0000	0000	0000	0000	0000	0000	0000	0000	: 0089
0104	:	0000	0000	0000	0000	0000	0000	0000	0000	: 0097
0112	:	0000	0000	0000	0000	0000	0000	0000	0000	: 0105
0120	:	0000	0000	0000	0000	0000	0000	0000	0000	: 0113
0128	:	0000	0000	0000	0000	0000	0000	0000	0000	: 0121

FIGURE 10
Register Table Display

Formats illustrated in the above display are as shown in table 5.

TABLE 5—Format Display for Figure 10

REGISTER NUMBER	NUMBER FORMAT
0001 to 0008	Hexadecimal
0009 to 0016	Unsigned Decimal
0017 to 0024	Signed Decimal
0025 to 0032	Signed Decimal Double Precision
0049 to 0056	Mixture of Formats

Changing Base of Values—Register Tables

The default value of the register display is unsigned decimal. The base of the values displayed can be changed by use of the HEX, ±DEC and DP keys which will change the register display of the selected register to hexadecimal, signed decimal or signed decimal double precision, respectively.

The hexadecimal, unsigned decimal and signed decimal displays each use 1 word in the register display. A signed decimal, double precision value uses 2 words in the register display. Note that 1 word is 1 register.

Double Precision Display

When using the DP key to change a register display to a signed decimal, double precision display 2 registers are affected by the change; the register that the cursor is on and the register with the next higher address. Figure 11 shows a double precision display.

The register that the cursor is on is the least significant word of the double precision number. The double precision number is displayed right justified in the location of the register on which the cursor is positioned. On row wrap-arounds (see example, Figure 10, R0032) the double precision number is displayed right justified in the location of the register (0032) that the cursor is on and the display is blank in the location of the register (0033) containing the most significant word.

To change a signed decimal, double precision display to another base, the cursor must be positioned on the least significant word when the change of base key is depressed. Both words (least and most significant words) are converted to the selected base.

NOTE

If the cursor is on register 1024 and an attempt is made to change the base to double precision, the error message INVALID ENTRY will be displayed in the work area.

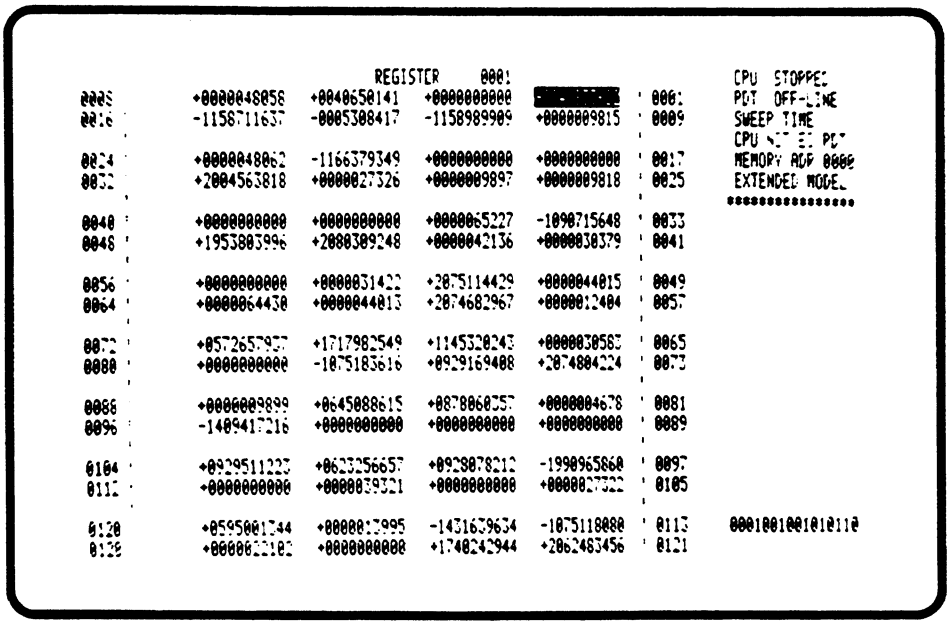


FIGURE 11
Double Precision Display

DISPLAY SCRATCH PAD

The scratch pad is a section of CPU memory containing information about system parameters and operating conditions of the CPU. The information provided by the scratch pad is then displayed on the screen as shown in Figure 12. Parameters which may be programmed by the Operator are displayed in reverse video.

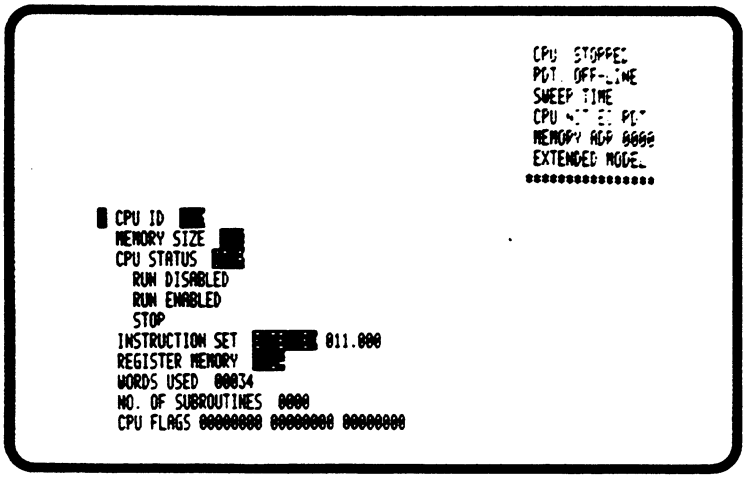


FIGURE 12
Scratch Pad Display

SCRATCH PAD DISPLAY

The scratch pad display is accessed by selecting it as an item on the Supervisor display. When the PDT keyswitch is in ON-LINE or MONITOR the PDT automatically reads the CPU scratch pad memory and updates the PDT memory with the current information.

When the PDT is in the OFF-LINE mode certain information can be entered by the operator to ensure that the PDT is configured with the same characteristics that the CPU will have during normal program execution.

Items on the scratch pad display are accessed by positioning the cursor next to the item by moving the cursor with the ↑ and ↓ keys. The selected parameter can then be updated or changed as required. The CPU can be stopped or started from this display.

When the keyswitch position is changed from the off-line mode to the on-line or monitor modes the parameters for MEMORY SIZE, INSTRUCTION SET and REGISTER MEMORY that are entered in the scratch pad display are compared with the corresponding parameters in the CPU. If an entry is incorrect an error message will be displayed in the work area. The current CPU status is displayed in the work area as well as in the scratch pad display and is updated whenever a change in the CPU status occurs.

When a parity error occurs in the CPU a bit is set in the scratch pad and the PDT displays a parity error message in the work area.

CPU Identification 001

Selection of this item allows the user to assign a CPU identification number. This is normally used if there is more than one CPU in a system. The CPU number is displayed in a reverse video block. The format for this entry is:

{ }, {ENTER}

Decimal number 0 to 255

The default CPU ID number on power up is 000.

Memory Size 32K

This item displays the size of the user memory in the CPU. If the display does not agree with the actual memory size an entry can be made to correct the display. The format for this entry is:

{ }, {ENTER}

An even decimal number from 2 to 32K, except 30K which is an illegal memory size.

CPU Status RUN ENABLED

RUN DISABLED
RUN ENABLED
STOP

The current operating status of the CPU is displayed here when in the on-line or monitor modes. The CPU can be stopped or started from this display when the PDT is in the on-line mode. To change the CPU status, position the cursor next to the desired choice with the cursor control keys ↑ and ↓, then depress {ENTER}.

If the CPU is stopped or started by using the keyswitch on the CPU, the new operating status will immediately be updated in the scratch pad display on the PDT if it is in the on-line or monitor modes.

The CPU status can be:

RUN DISABLED—Starts CPU with all outputs off.
RUN ENABLED —Starts CPU with all outputs enabled.
STOP —Stops CPU, all outputs are turned off.

Instruction Set EXTENDED XXX.YYY

When the PDT is in the off-line mode, the operator can select the desired function set by positioning the cursor next to **INSTRUCTION SET** and depressing the {ENTER} key. This will cause the display to toggle between BASIC and EXTENDED.

In order to enter a program using the extended function set, the display must read EXTENDED. If an attempt is made to program with the extended functions and the display reads BASIC, the following message will be displayed in the work area:

REQUIRES
EXTENDED SET

If the ladder diagram program contains extended functions and an attempt is made to change the entry in INSTRUCTION SET to BASIC, the entry will remain EXTENDED and the following message will be displayed in the work area:

REQUIRES
EXTENDED SET

The display also contains 2 sets of numbers. The first 3 numbers (XXX) indicate the version of the PDT software. The second 3 numbers (YYY) indicate the version of the CPU software (when the PDT is in the on-line or monitor mode).

Register Memory 1024

The number present in this display indicates the amount of register memory present in the CPU. The default value on power-up is 1024.

Depressing {ENTER} causes a toggle between 1024 (model 600 or 6000) and 256 (model 60).

Words Used 00000

This display indicates the number of words of user memory in the program. The display is updated automatically and maintains a current status of the number of words used. This display is not changeable by the operator. The number format of the display is decimal.

Number of Subroutines 00000

This display shows the number of subroutines in the program. The maximum number of subroutines available in a program is 16. This display is updated automatically and cannot be changed by the operator.

CPU Flags XXXXXXXX XXXXXXXX XXXXXXXX

This display is provided by data from the CPU (when the PDT is in the on-line or monitor mode) and indicates the location and type of error that has occurred in a system. Three bytes of binary data are displayed. Figure 13 is a representation of the 3 bytes and the error information available in each byte.

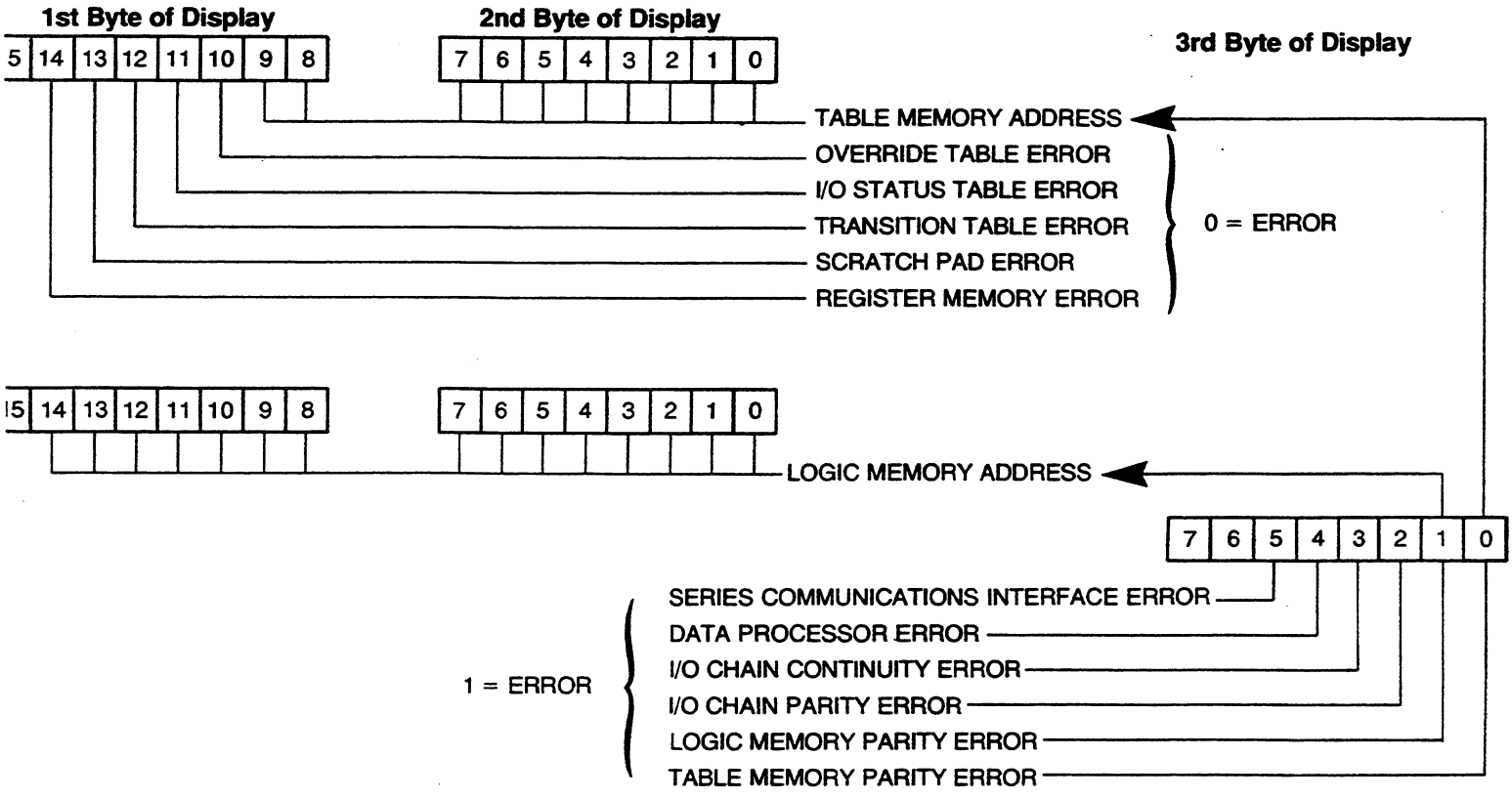


FIGURE 13
CPU Flags, Byte Data

SELECT TAPE UNIT

Selection of this item from the Supervisor menu provides a method of configuring the Program Development Terminal tape I/O interface to accept either an internal or 1 of 2 external tape units.

Position the cursor (reverse video block) next to the menu item SELECT TAPE UNIT and depress the {**ENTER**} key. The display will show the next item in the sequence: INTERNAL, EXTERNAL IIA, and EXTERNAL III. The default display on power-up is INTERNAL.

Each time the ENTER key is depressed the next item in the sequence will be displayed. The item in the sequence that is displayed is the selected configuration for the tape interface.

The tape units and media which may be used in conjunction with the selected configuration are shown in Table 6.

TABLE 6—Tape Unit Selection

DISPLAY	TAPE UNIT	TAPE MEDIA
INTERNAL		3M DC 100A Data Cartridge
EXTERNAL IIA	STR LINK IIA Compatible Device	Philips Data Casette
EXTERNAL III	STR LINK III Compatible Device	3M DC 100A Data Cartridge

Tapes generated on the internal tape unit and STR LINK III compatible devices are interchangeable.

TAPE TO PDT

Selection of this item from the menu causes a tape to be read into the Program Development Terminal memory. The tape will be read from the internal tape unit if the entry SELECT TAPE UNIT is set to INTERNAL, otherwise the appropriate tape unit should be connected to the TAPE port located inside the cable access door on the bottom of the PDT. If an external tape unit is to be used, the entry BAUD RATE must be set to the appropriate value. The internal tape unit can read or write a tape for a 32K CPU in less than 3 minutes.

The Program Development Terminal must be off-line to use this function. If an attempt is made to transfer a tape to PDT memory and the PDT is not off-line an error message NOT OFF-LINE will be displayed in the work area.

If an attempt is made to use this function without a cassette in the tape unit (internal or external) or if the external Minicartridge Tape Unit is not ready in some other way (no power, etc.), the error message UNIT NOT READY will be displayed in the work area.

PDT TO TAPE

This function causes the program, tables and scratch pad to be written to tape. The format, error messages, etc. are the same as for the TAPE TO PDT function. The entire program memory can be transferred to one cassette or minicartridge. If the tape is write protected and this operation is attempted, the message WRITE PROTECT will appear in the work area after the tape is rewounded (Max. 1 min.). The PDT must be off-line.

COMPARE TAPE-PROGRAM AND TABLES

Selection of this item causes a tape to be transferred to the Program Development Terminal; however, the contents of the tape are compared with the Program Development Terminal memory rather than stored in it. The User program, Input table, Output table and Register tables are compared.

If there are any differences between the contents of the tape and the contents of memory, the part of memory (e.g. table, register memory or program memory) and the address of the part of memory that did not compare will be displayed in the work area. The first 10 addresses not comparing will be displayed. The message TAPE NOT EQ PDT will be displayed at the bottom of the work area. The PDT must be off-line.

COMPARE TAPE-PROGRAM ONLY

Selection of this item causes only the User program on tape to be read into and compared with the program in the Program Development Terminal. Any parts of the program that do not compare are handled in the same manner as in the COMPARE TAPE-PROGRAM AND TABLES function in that the first 10 addresses of program memory not comparing will be displayed in the work area. If there are miscompares, the message TAPE NOT EQ PDT will also be displayed at the bottom of the work area. The PDT must be off-line.

PRINT LADDER DIAGRAM

This function when selected causes all or portions of the User program stored in the Program Development Terminal memory to be sent through the external printer port in order to obtain a hard copy listing of the program.

The printer interface is serial, RS-232C, half duplex with a selectable baud rate. The baud rates are 110, 300, 1200, 2400, 4800 or 9600 baud. The printer configuration including the baud rate should be the same as the printer interface in the Program Development Terminal. The PDT mode selection keyswitch must be in the OFF-LINE position.

A rung will be shown in its entirety on a page and each page will contain a header. The header will contain the address of the first word on the page and the page number. Figure 14 is a typical page of a ladder diagram listing.

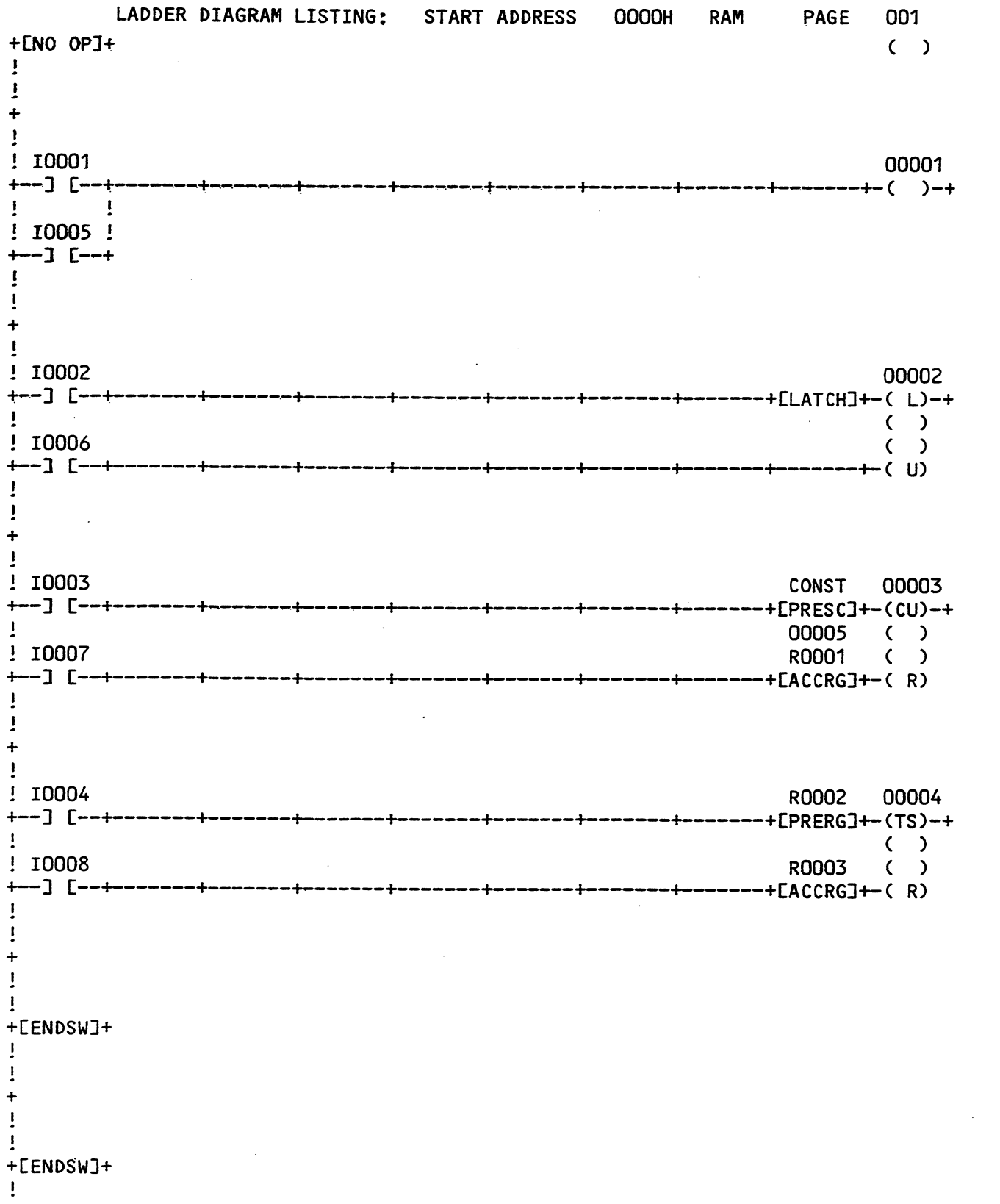


FIGURE 14
Typical Ladder Diagram Printout

There are 2 selectable features which should be configured before initiating printing. They are START OUTPUT NO. and NUMBER OF RUNGS. The format and description of these features is described below.

{ }, { }, {**ENTER**}
- { } — A if an auxiliary output is to be specified
- { } — Output number from 1 to 1024

This allows the operator to specify the starting rung for the ladder diagram listing (printout). If a 0 is entered the listing starts with the first rung in the program, which is also the default value if a number is not specified. If a number greater than 1024 is entered an error message ILLEGAL NUMBER will be displayed in the work area. The number of rungs defaults to the quantity in the ladder diagram.

{**NUMBER OF RUNGS**}, {**ENTER**}
- {**NUMBER OF RUNGS**} — Number of rungs to be printed (optional). 0 is an illegal number

This feature gives the operator the option of specifying the number of rungs for the ladder diagram to be printed. If a number is not specified, all rungs will be printed.

After selecting the starting output number and the number of rungs to be printed position the cursor next to PRINT LADDER DIAGRAM and depress {**ENTER**}. Printing of the ladder diagram listing will then begin.

While the ladder diagram is being printed the status message BUSY is displayed in the work area. If it is desired to stop the printing before completion of the specified number of rungs, depressing {**CLEAR**} will cause printing to stop and control will return to the Supervisor display.

If a ladder diagram listing is requested, but the printer is not ready or not connected an error message DEVICE NOT READY will be displayed in the work area. If the PDT mode selection switch is not in the OFF-LINE position, the error message NOT OFF-LINE will be displayed. When the problem specified by the error message has been corrected, positioning the cursor next to PRINT LADDER DIAGRAM and depressing {**ENTER**} will cause the printing of the ladder diagram to begin.

PRINT CROSS REFERENCE TABLE

Selection of this function allows the operator to obtain a listing of a cross reference table or tables on an external printer. Printer parameters must be configured to match those of the serial interface of the PDT. The tables that may be printed are the Input, Output, Auxiliary Input, Auxiliary Output and Register. If no table is specified all cross reference tables are printed.

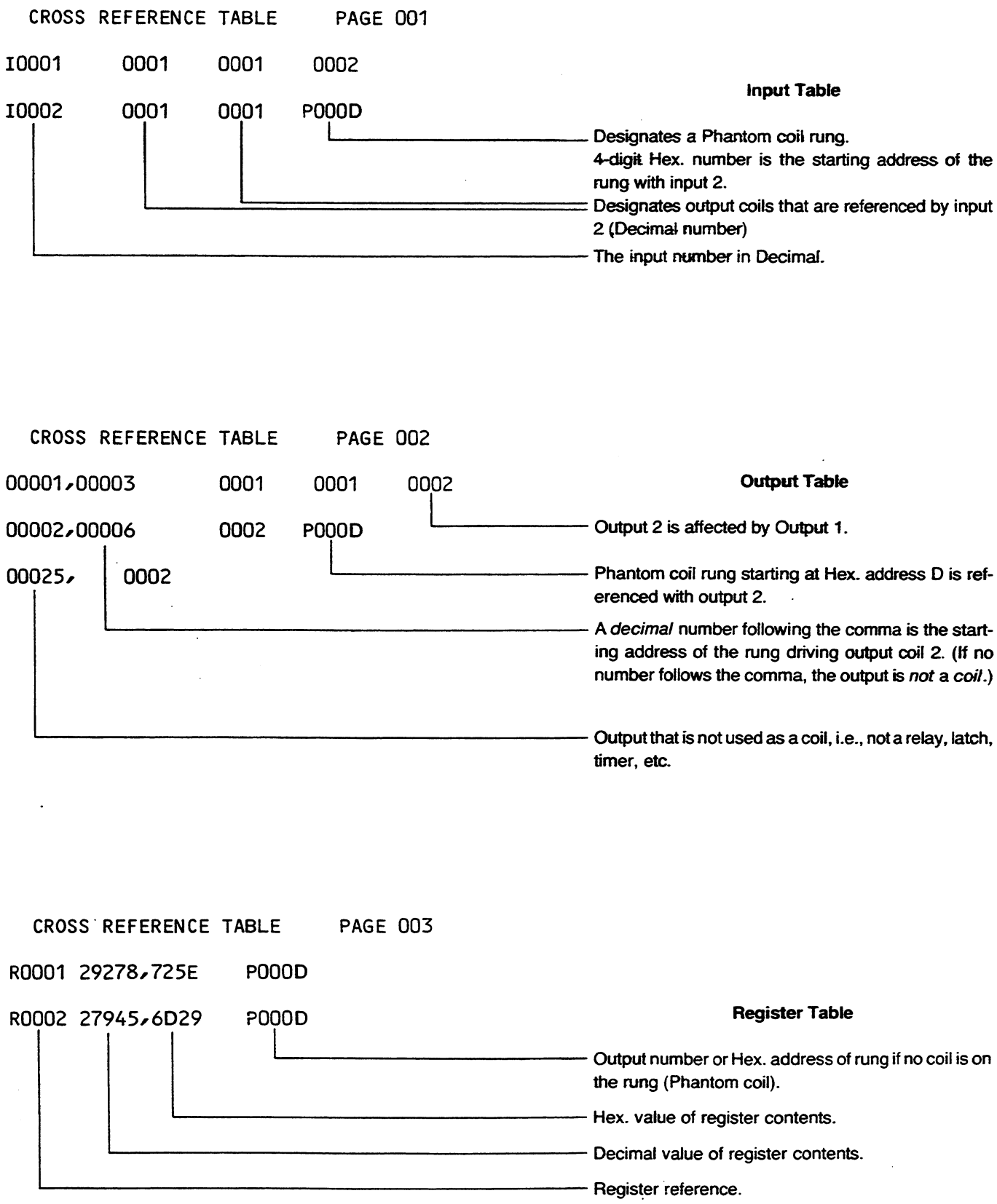
To initiate printing of the selected cross reference table or tables position the cursor next to the function and proceed with the following format.

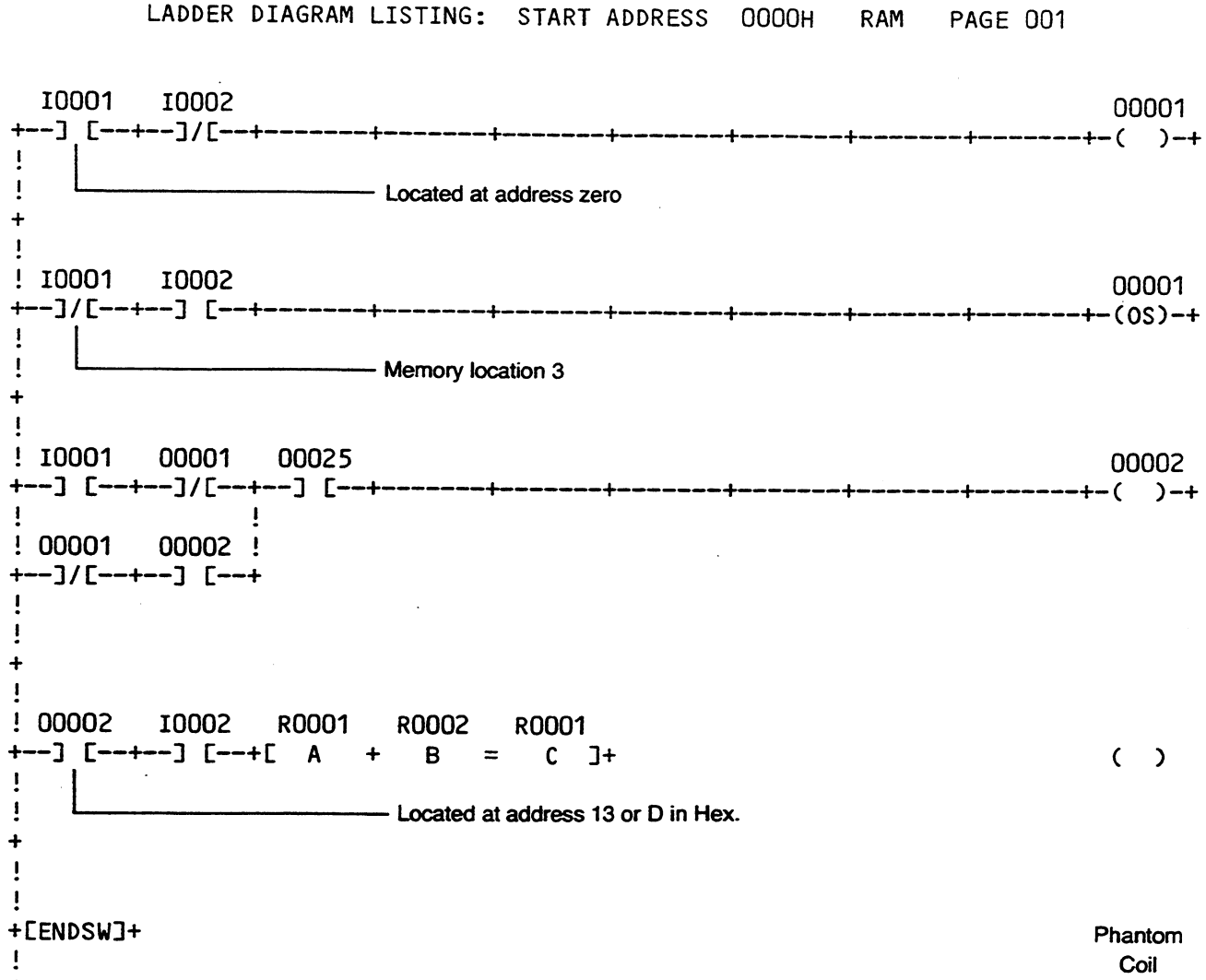
{ } {**ENTER**}
- { } — I, O, AI, AO, R Selects the desired table.

NOTE

Printing a cross reference table causes memory to be initialized to all End of Sweeps. Therefore, it is recommended that the program in the PDT memory be transferred to tape or stored in a CPU before printing a cross reference table.

Figure 15 is a sample of a typical printout of a cross reference table.





Example: Suppose you wanted to see where input 2 is located on your program. You would go to the input table (page 1) and see output coils 1, 1, and phantom coil at D. You then look at the output table to see where outputs 1 and 1 are. You can only find the last rung output coil 1 is in. The previous rung you must look for in the Program (Usually you won't have two coils with the same output number). In this case, address 3 and D are your rung starting addresses.

FIGURE 15
Cross Reference Table Description

BAUD RATE

This function allows the operator to select the baud rate for the serial ports in the PDT. To select a baud rate other than the one displayed, position the cursor next to the menu item BAUD RATE. Notice that a baud rate is displayed in a reverse video block. Depressing the {**ENTER**} key causes a scroll to the next allowable baud rate. The baud rate displayed is the baud rate selected.

The selectable baud rates are 110, 300, 1200, 2400, 4800 and 9600 baud.The default baud rate on power-up is 300 baud.

CLEAR LOGIC MEMORY PARITY ERROR

Selection of this function allows logic memory parity errors in the CPU to be cleared. The contents of the logic memory are written to the Program Development Terminal and back to the CPU which will clear any soft parity errors in the logic memory.

To use this function the CPU must be stopped. Position the cursor next to the item on the Supervisor menu and depress {**ENTER**}. A message will appear in the work area, TO EXECUTE TYPE SHIFT-ENTER. To complete the initiation of the function the {**SHIFT**} and {**ENTER**} keys must be depressed simultaneously. A BUSY status message appearing in the work area indicates that the function is in process. The BUSY message will be removed when the function is complete.

If an attempt is made to initiate this function when a CPU is running, the error message CPU MUST BE STOPPED will be displayed in the work area.

CLEAR SCRATCH PAD AND TRANSITION PARITY ERROR

Selection of this function allows scratch pad and transition table parity errors in the CPU to be cleared. The contents of the scratch pad and transition table memories are written to the Program Development Terminal and back to the CPU, thereby clearing any soft parity errors in those memories.

The procedure for completing this function is the same as for the CLEAR LOGIC MEMORY PARITY ERROR function except that the cursor is positioned next to the item CLEAR SCRATCH PAD AND TRANSITION PARITY ERROR.

BUILDING LADDER DIAGRAMS

The following section describes the formats and shows how to build ladder diagrams using the basic relay functions including relays, one-shots, latches, counters, and timers. The symbols used for building ladder diagrams are shown in Figure 16.

SYMBOL	REPRESENTS
+-------+	Horizontal Connector
+ ! ! +	Vertical Connector
+ +	Horizontal Open
+ +	Vertical Open
+--] [--+	Normally Open Contact
+--]/[--+	Normally Closed Contact
-[XXXXX]-	Mnemonic Functions
+-( )-	Coils

FIGURE 16
Ladder Diagram Symbols

The method for describing how to enter each function will consist of 3 steps:

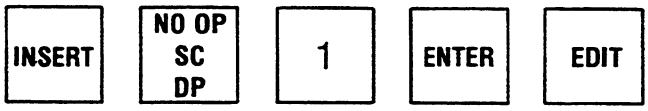
- The format for entering the function is described. Required cursor positions and any other considerations for that function are detailed.
- An illustration of the completed rung as it will appear on the PDT display is shown.
- A step-by-step entry of the function with an illustration of each key is shown.

Since the PDT accepts a free format for entry of most functions there is more than one order in which the keys can be depressed when entering a function.

When the keys are depressed in the order given, the resulting display will be as shown in the illustration. As each key is depressed note the entry displayed in the work area. Use this display to verify your entry before completing each step. If an illegal key entry is attempted an error message will be displayed at the bottom of the work area. A chart listing the error messages and their meanings can be found on page 4.8. The error message and the key entry display can be removed from the screen by depressing {**CLEAR**}.

Following are several rules and guidelines which should be followed when entering new rungs or lines of logic in a program or when modifying an existing program.

1. To enter lines of logic and build rungs in a ladder diagram, the user program section of memory must be displayed. This is initiated from the Supervisor menu. Depressing the {**SUPERVISOR**} and {**ENTER**} keys will cause the Supervisor menu to be displayed. The cursor will be on the first entry, DISPLAY PROGRAM. Depressing {**ENTER**} again will cause the first part of program memory to be displayed or the part that was last displayed.
2. With no program in the PDT or the CPU a NO OP should be entered as the first line of a program. The PDT display with no program entered will be 2 End-of-Sweeps (ENDSW). Enter the NO OP by the following sequence.



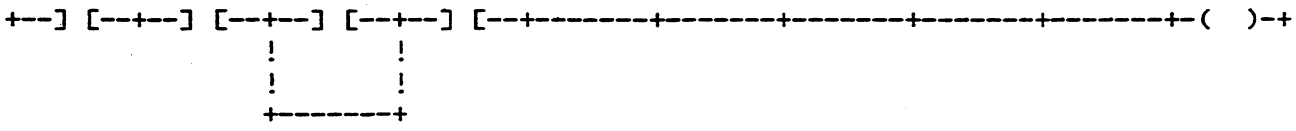
   The first line will now be NO OP, the PDT will be out of the Edit mode and in DISPLAY PROGRAM control. The PDT is now ready to accept the entry of rungs of logic.
3. In order to enter any line of logic or change a line of logic the PDT must be in the EDIT mode. The EDIT mode is entered by depressing {**EDIT**} or {**INSERT**}. The Edit cursor will be displayed in the first position (top left) on the screen when entering the EDIT mode. Depressing {**EDIT**} allows you to change an existing rung. {**INSERT**} allows you to create a new rung.
4. The rung of logic displayed at the top of the screen precedes the next rung to be entered in sequence (e.g. rung #1, rung #2, etc.). When entering a program with rungs in sequential order {**INSERT**} puts the next rung at the top of the screen and in the proper sequence.
5. If a rung is to be entered in an existing program, scroll the program to where the new line is to be inserted using the {**NEXT**} or {**PREV**} key. When the line that will precede the new line is displayed as the first line of the display depress {**INSERT**}.
6. After entering a new rung of logic or modifying an existing rung depress {**EDIT**}. This will cause the rung to be checked for correctness and if correct it will be entered into PDT memory. If the rung is incorrect an error message will be displayed. The PDT is now out of the EDIT mode. The underline cursor is at the top left of the screen.
7. The new program or rung of logic is not in the CPU memory at this time, it must be transferred from the PDT memory to the CPU memory. Depress {**PDT → CPU**}, {**ENTER**}. A message will appear in the work area (see Chapter 3 for a detailed description of the transfer keys). To complete the transfer depress {**SHIFT**} and {**ENTER**}.

NOTE

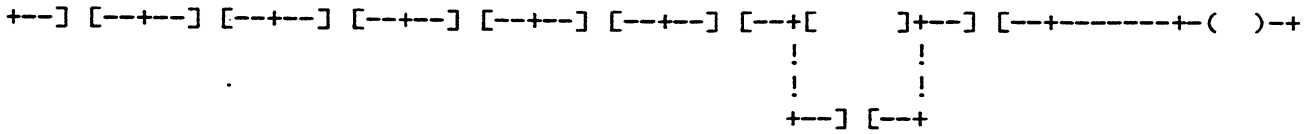
The CPU keyswitch must be in the WRITE position in order to enter data into the CPU memory.

8. Any rung entered must be of a legal configuration, otherwise the PDT will not accept the rung until the errors are corrected. Following are several examples of illegal rungs.

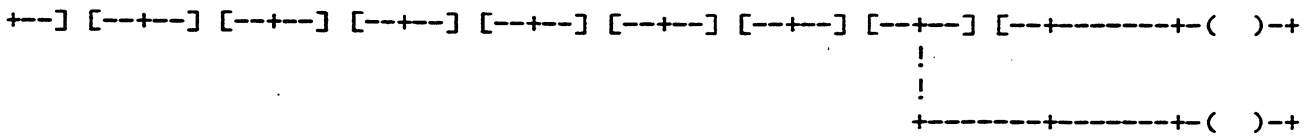
Illegal Rung: Short circuit



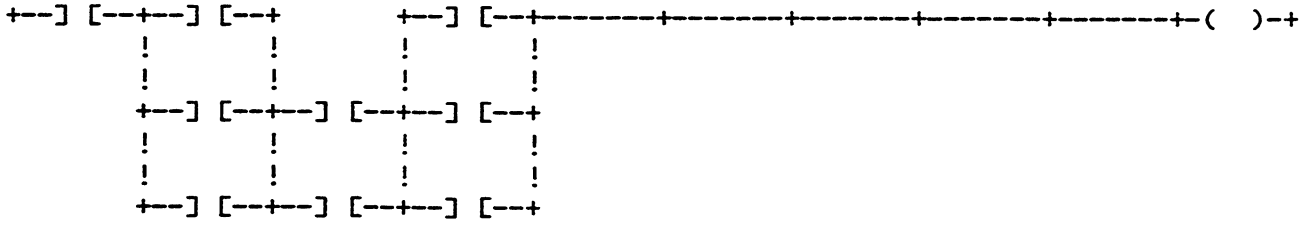
Illegal Rung: Nothing allowed in parallel with a mnemonic



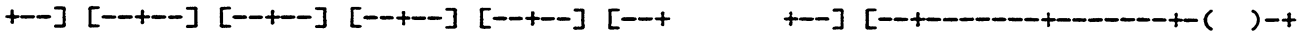
Illegal Rung: No more than one output per rung



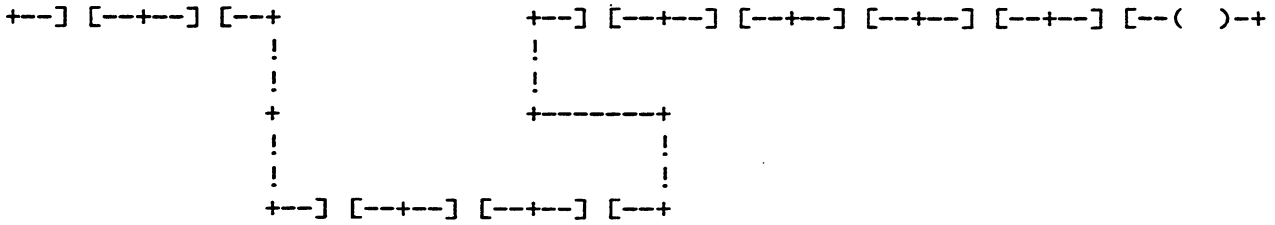
Illegal Rung: Connect contact to topmost available junction



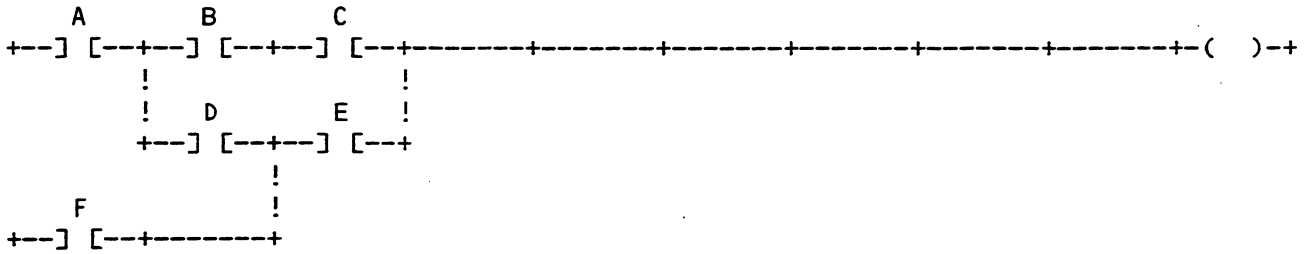
Illegal Rung: Open Circuit



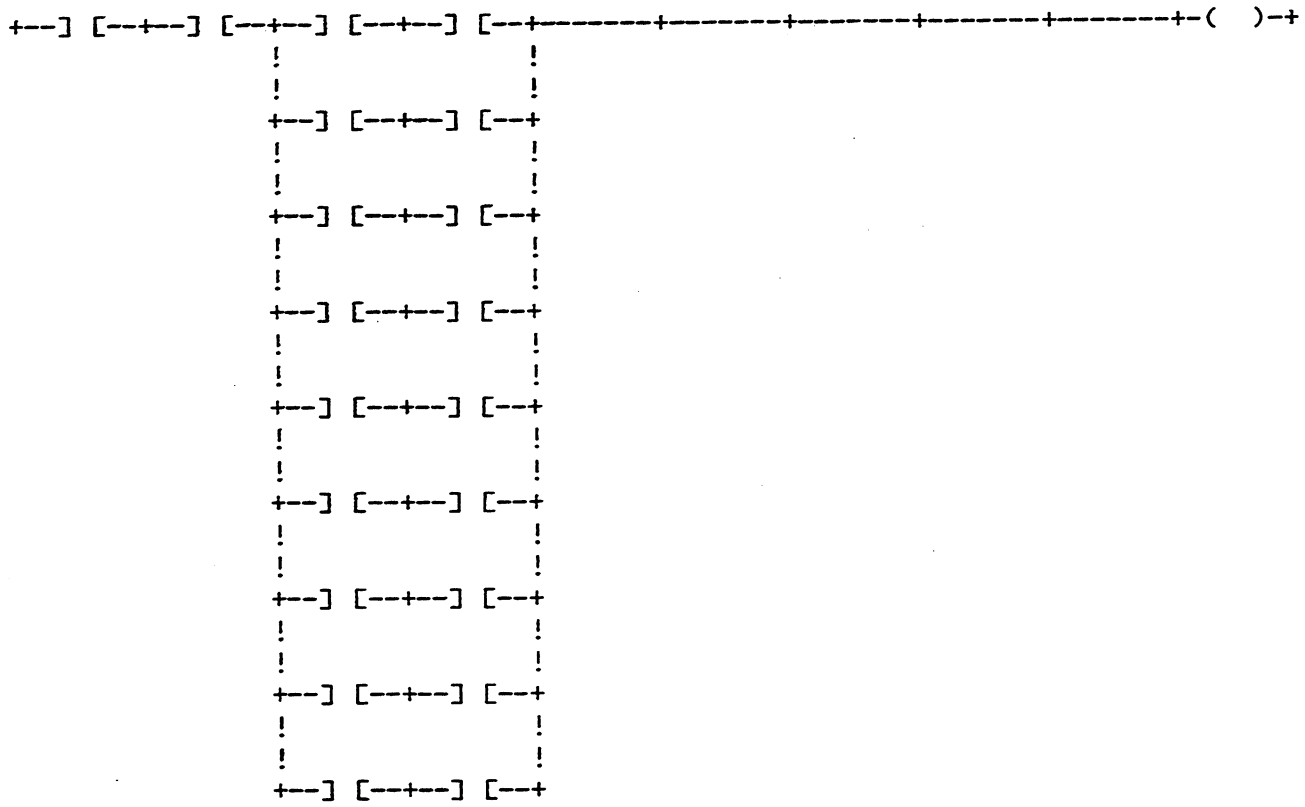
Illegal Rung: More than nine series contacts



Illegal Rung: Starting a branch within a branch



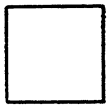
Illegal Rungs: More than eight lines of logic



In the following step-by-step instructions for entering rungs the symbols shown below will be used to denote the programming steps used in the examples.

Indicates the line number in a rung of logic or in a program

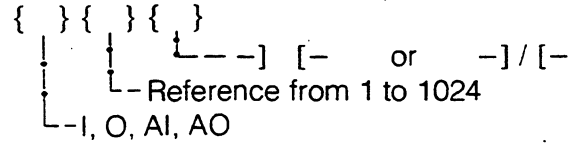
Indicates position of the cursor in a line of logic



Indicates a key to be depressed

ENTERING CONTACTS

The format for entering a contact is:



A contact is entered in a line of logic in the position specified by the position of the cursor. There are 10 positions horizontally in a line; however, contacts can only be entered in position 1 to 9. The tenth position is reserved for an output coil. In the example below the contacts are shown in position 1 and 2. The contacts as they will appear on the PDT display are shown first, then the keystrokes required for the entering those contacts. The sequence of keystrokes enters a normally-open and a normally-closed contact.

DISPLAY — CONTACTS	
I0005 AI0015 +--] [--+--]/[--+	
CURSOR POSITION	**KEYSTROKES**
	INSERT
L1 1	I 5 ⊣⊢ →
L1 2	DP A I 1 5 ⊣/⊢

ENTERING RELAYS AND ONE-SHOTS

To enter a relay or a one-shot rung, first enter the contacts in positions 1 to 9. If more than 9 contacts are required in series, enter the 9 contacts and assign an output coil reference to a coil in position 10. Use a contact from that coil as the first contact in the next line of logic. In this manner as many contacts as are required can be put in series.

Contacts in a rung can be entered in parallel with as many as 8 contacts in parallel.

The tenth position is reserved for the output coil. Only 1 coil per rung is permissable. A relay and a one-shot rung are identical except for the coil assignment in position 10 which will be either RELAY or ONE SHOT as applicable.

{ } { } { } {EDIT}
- └ Reference from 1 to 1024
- └ O or AO (defaults to O if not specified)
- └ RELAY or ONE SHOT

DISPLAY — RELAY

```
  I0025                                                          00100
+--] [--+-------+-------+-------+-------+-------+-------+-------+-------+-( )-+
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I, 2, 5, ⊣⊢
L1 10	-0- RELAY, 1, +/- 0, +/- 0, ENTER, EDIT

DISPLAY — ONE SHOT

```
  I0145                                                          00325
+--] [--+-------+-------+-------+-------+-------+-------+-------+-------+-(OS)-+
```

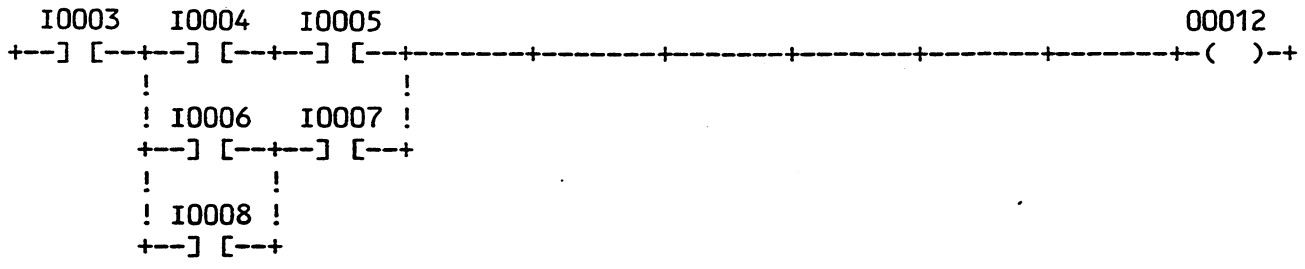
CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I, 1, 4, 5, ⊣⊢
L1 10	-0- ONE SHOT, 3, 2, 5, ENTER, EDIT

The above sequence enters a one-shot and transfers the rung of logic to the PDT memory.

ENTERING PARALLEL CONTACTS

The following sequence of keystrokes will cause a relay rung to be entered with parallel contacts. The sequence of keystrokes was selected to use the least number of cursor movements when entering the parallel contacts.

DISPLAY — RELAY RUNG, PARALLEL CONTACTS



CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 3 ⊣⊢ →
L1 2	I 4 ⊣⊢ ↓ ↓
L2 2	I 6 ⊣⊢ ↓ ↓
L3 2	I 8 ⊣⊢ →
L3 3	↑
L2 3	↓ I 7 ⊣⊢ ↑
L1 3	I 5 ⊣⊢ →
L1 4	↓
L1 10	-0- RELAY 1 2 ENTER EDIT

ENTERING LATCHES

A latch function contains 2 lines of logic. The top line is the latch contact network and the bottom line is the unlatch contact network.

To enter a latch, first build the latch contact network. Contacts or mnemonics can be entered in the first eight positions in the line. No contacts or mnemonics are allowed in position nine. After the required contacts or mnemonics have been entered, move the cursor to the tenth position and specify the latch function, then assign an output reference number to the associated coil. The unlatch line is predominant.

When the latch network is complete build the unlatch network. When the last contact or mnemonic has been entered, depressing the {EDIT} key will complete the building of the latch.

DISPLAY — LATCH

```
  I0035   AI0163                                                    00130
+--] [--+--]/[--+-------+-------+-------+-------+-------+-------+[LATCH]+-( L)-+
!                                                                          (  )
! I0642                                                                    (  )
+--] [--+-------+-------+-------+-------+-------+-------+-------+-------+-( U)
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 3 5 ⊣⊢ →
L1 2	DP A I 1 6 3 ⊣/⊢
L1 10	-0- LATCH 1 3 +/- 0 ENTER
L2 1	I 6 4 2 ⊣⊢ EDIT

The example of a latch shows 2 contacts, one normally-open and one normally-closed in the latch line. Note that the normally-closed contact has an auxiliary reference. Auxiliary contacts can only be used with a system configured with the Extended function set. The unlatch line has one contact (in the example).

Depressing the {EDIT} key completes the keystrokes for the latch function and causes the PDT to complete building the latch. The PDT returns control to the DISPLAY PROGRAM mode. Depressing {INSERT} puts the PDT into the EDIT mode and the next rung of logic can be entered.

ENTERING COUNTERS

A counter function contains 2 lines of logic. The top line is the count string and the bottom line is the reset string. Each positive transition on the count line causes the accumulated value to increment (up-counter) or decrement (down-counter) one count if there is no continuity in the reset line. Down counters turn on when the accumulate value = 0. Up counters turn on when the accumulate value is equal to or greater than the preset value. Down counters reset the accumulate value to the preset value. Up counters reset the accumulate value to zero.

To enter a counter, first build the count string. Contacts or mnemonics can be entered in the first 8 positions. The ninth position is reserved for the count preset value. When the PRE/ACC key is depressed a menu is displayed in the work area as shown below.

1. PRESC (Preset with a constant)
2. PRERG (Preset from a specified register)
3. ACCRG (Accumulate the count value in a specified register)

The desired preset is selected by depressing {1} or {2} and {**ENTER**}. PRESC requires entering a constant operand value. PRERG requires entering a register reference.

After entering the preset, move the cursor to position 10 and assign a number to the coil, then specify either an up counter or a down counter.

The next step is to build the reset string. The ninth position of the second line is reserved for the counter accumulate value. Again depress the {**PRE/ACC**} key and select 3 ACCRG. A register must be assigned for storage of the accumulate value. Depressing the {**EDIT**} key will complete building the counter and return control to DISPLAY PROGRAM.

DISPLAY — COUNT UP

```
  I0036                                                    CONST  00055
+--] [--+-------+-------+-------+-------+-------+-------+-------+[PRESC]+-(CU)-+
!                                                                    060    (  )
! I0432                                                             R0005  (  )
+--] [--+-------+-------+-------+-------+-------+-------+-------+[ACCRG]+-( R)
                                                                   00000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 3 6 ⊣⊢
L1 9	PRE ACC 1 ENTER 6 −/+ 0 ENTER →
L1 10	-0- COUNT UP 5 5 ENTER
L2 1	I 4 3 2 ⊣⊢
L2 9	PRE ACC 3 ENTER R 5 ENTER EDIT

The function entered above is the count-up function with a constant preset specified. The example below will enter a count-down function using a preset value from a specified register.

DISPLAY — COUNT DOWN

```
  I0222                                                          R0015  00144
+--] [--+-------+-------+-------+-------+-------+-------+-------+[PRERG]+-(CD)-+
!                                                                01035   (  )
! I0119                                                         R0009   (  )
+--] [--+-------+-------+-------+-------+-------+-------+-------+[ACCRG]+-( R)
                                                                 00000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 2 2 2 ⊣⊢
L1 9	PRE ACC 2 ENTER R 1 5 ENTER
	1 +/− 0 3 5 ENTER →
L1 10	-0- COUNT DOWN 1 4 4 ENTER
L2 1	I 1 1 9 ⊣⊢
L2 9	PRE ACC 3 ENTER R 9 ENTER EDIT

ENTERING TIMERS

The programming structure of a timer is similar to a counter in that 2 strings of logic are required. The top string starts the timer when power flow is provided to the timer through contacts in the string. A timer can be one of three types, either 1.0 second, 0.1 second or 0.01 second. A preset value for the duration of the timer is specified in a register or as a constant.

The bottom line contains the reset string. The current (accumulated) time is stored in the specified register. When the current time in the accumulate register equals or exceeds the preset value, the coil specified by a reference in position 10 will turn on. Continuity in the bottom line of contacts will reset the accumulated time to zero.

To enter a timer follow the procedure used for entering counters. Build the timer string by entering contacts, entering a preset value (constant or from a specified register), assigning a coil reference number and specifying the type of timer. Then build the reset string by entering contacts and specifying a

register for storing the accumulation of time. Depressing the {EDIT} key will complete building of the timer rung. Control will then return to the DISPLAY PROGRAM mode.

NOTE

The example of a timer rung below is a 1.0 second timer. To enter a 0.1 or 0.01 timer depress the proper key in place of the 1.0 second key.

DISPLAY — TIMER (TS, TT, TH)

```
  I0225                                                          R0032  00012
+--] [--+-------+-------+-------+-------+-------+-------+-------+[PRERG]+-(TS)-+
!                                                                00060   (  )
! I0042                                                         R0005   (  )
+--] [--+-------+-------+-------+-------+-------+-------+-------+[ACCRG]+-( R)
                                                                 00000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 2 2 5 ⊣⊢
L1 9	PRE ACC 2 ENTER R 3 2 ENTER
	6 +/− 0 ENTER →
L1 10	-0- TIMER 1.0 1 2 ENTER
L2 1	I 4 2 ⊣⊢
L2 9	PRE ACC 3 ENTER R 5 ENTER EDIT

To enter a constant value in the preset position of the top string select PRESC from the menu after depressing the {PRE/ACC} key. A constant value is then entered to complete the preset entry. When you have completed entering the 3 basic timers into the PDT you should have a good understanding of the basic relay functions. Chapter 5 will provide a step-by-step entry of the mnemonic functions in the Basic Function Set.

CHAPTER V
Entering Basic Mnemonic Functions

INTRODUCTION

The intent of this chapter is to allow the user to become familiar with the Basic mnemonic functions used when programming a Series Six system. Three of the mnemonic functions (PRESC, PRERG and ACCRG) were described in Chapter 4 with the relay functions since they are used only when entering timers and counters.

The method that will be used for presenting each function is to:

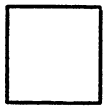
- Describe what the function does.
- Show an illustration of a rung of logic using the function.
- Show a step-by-step sequence of keystrokes required for entering the illustrated rung.

NOTE

In the following step-by-step sequences for entering functions, the symbols shown below are used to denote the programming steps used in the examples.

Indicates the line number in a rung of logic or in a program

Indicates position of the cursor in a line of logic



Indicates a key to be depressed

SHIFT SHIFT Status Table Left One Bit

This function will cause the 16 contiguous bits of the status table specified by the Input or Output reference to be shifted one bit (position) to the left (towards a higher reference number). The most significant bit controls the power flow and a 0 is loaded into the least significant bit. If the most significant bit prior to the shift is a one, power flow is outputted; otherwise no power flow is generated. The data displayed is the current contents of the referenced table. The function can be placed in any of the first 9 positions in a line of logic.

Each scan will cause the shift function to execute until all of the bits are shifted or until there is no continuity (power flow) to the function. The status table address (reference) must be on a byte boundary for the last significant bit, e.g. I0001, I0009, I0017, etc.

DISPLAY — SHIFT

```
  I0215   I0009
+--] [--+[SHIFT]+                                    (  )
          0000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 2 1 5 ⊣⊢ →
L1 2	SHIFT MOVE 1 ENTER I 9 ENTER EDIT

I/O R Move I/O Table To Register

This function allows a word of data to be moved from an I/O status table word to a register. An Input or Output status table is specified by the reference in the first word of the function. The status table reference must be on a byte boundary for the least significant bit. The 16 contiguous bits of the status table referenced in the first word are moved to the register specified by the second word of the function.

The function can be placed in any of the first 8 positions in a line of logic. If the cursor is in position nine and an attempt is made to enter the I/O R function, the error message NOT ENOUGH ROOM IN LINE will be displayed. The function is conditional in that it will operate only when power flow is provided to it in the line of logic. Power flow is outputted whenever the function is active.

DISPLAY — I/O R

```
  I0472    I0025     R0018
+--] [--+[ I/O  TO  REG ]+                                    ( )
           0000      0000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 4 7 2 ⊣⊢ →
L1 2	SHIFT MOVE 2 ENTER I 2 5 ENTER
	→
L1 3	R 1 8 ENTER EDIT

R I/O Move Register To I/O Table

This function when executed allows a word of data to be moved from a register to an I/O status table word. The 16 bits in the register specified by the first reference in the function are moved to the Input or Output status table specified by the second reference. The address of the status table reference is that of the least significant bit and must be on a byte boundary. If an I/O table reference is entered, but is not on a byte boundary the PDT will automatically enter the address of the nearest lower byte boundary. The reference on the byte boundary will be displayed.

Execution of the function is conditional upon receipt of power flow. Entry of the function can be in any of the first 8 positions in the line of logic. Power flow is outputted whenever the function is active.

DISPLAY — R I/O

```
  I0333    R0067     00065
+--] [---+[ REG  TO  I/O ]+                                    ( )
           0000      0000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 3 3 3 ⊣⊢ →
L1 2	SHIFT MOVE 3 ENTER R 6 7 ENTER
	→
L1 3	HEX O 6 5 ENTER EDIT

BIBCD BInary To BCD Convert

This function converts a binary value contained in a register to 4-digit BCD and stores the BCD number in a status table. This provides a convenient method for allowing the I/O structure to output a number for visual display.

The contents of the register specified by the BIN reference are converted to 4-digit BCD and stored in the status table at the address specified by the BCD reference. The BCD reference is the least significant bit of the status table and must be on a byte boundary. The register contents remain the same.

Execution of this function is conditional upon receipt of power flow. If the value of the register is greater than 9999, the 4 least significant digits are stored in the status table and power flow outputted, otherwise no power flow is generated.

DISPLAY — BIBCD

```
 AI0045   R0214    I0033
+--] [--+[ BIN  TO  BCD ]+                                    ( )
         00000     0000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	DP A, I, 4, 5, ⊣⊢, →
L1 2	SHIFT MOVE, 4, ENTER, R, 2, 1, 4
	ENTER, →
L1 3	I, 3, 3, ENTER, EDIT

BCDBI BCD To BInary Convert

This function converts the 16-bit contents of the status table specified by the BCD reference from 4-digit BCD to binary and stores them in the register specified in the BIN reference. The BCD status table reference is the least significant bit of the specified reference and must be on a byte boundary. The contents of the status table remain the same.

Execution of this function is conditional upon receipt of power flow. If any illegal BCD digits are in the status table the result will be computed as the binary sum of the BCD weights of the bits in the status table word and power flow is outputted, otherwise no power flow is generated.

DISPLAY — BCDBI

```
  I0054    I0097     R0129
+--] [--+[ BCD  TO  BIN ]+                                    ( )
          0000      00000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 5 4 •⊣⊢• →
L1 2	SHIFT MOVE 5 ENTER I 9 7 ENTER
	→
L1 3	R 1 2 9 ENTER EDIT

ADD ADD Binary

This function performs the addition of two 16-bit values such as those generated by counters, timers and numerical I/O devices. The contents of the register specified by the first reference are added to the contents of the register specified by the second reference (binary addition). The result (sum) is placed in the register specified by the third reference.

The function will pass power flow if the result of the addition is greater than 16 bits (65535). When the sum of the addition does exceed 16 bits, which causes a carry out of the most significant bit, power flow will be outputted, otherwise no power flow will be generated.

The ADD function can be inserted in position 1 through 7 in a line of logic in a ladder diagram. Position 7 is the last position in which the function can be inserted since it requires 3 positions in a line of logic. If an attempt is made to insert the function in position 8 or 9, the following error message will be displayed in the work area:

NOT ENOUGH
ROOM IN LINE

Operation of the function is conditional upon receipt of power flow.

DISPLAY — ADD

```
  I0162    R0245    R0246  R0247                                        00516
+--] [--+[  A    +   B  =   C  ]+-------+-------+-------+-------+-------+-(  )-+
          00000    00000  00000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 1 6 2 ⊣⊢ →
L1 2	ARITH 1 ENTER R 2 4 5
	ENTER →
L1 3	R 2 4 6 ENTER →
L1 4	R 2 4 7 ENTER
L1 10	-0- RELAY 5 1 6 ENTER EDIT

SUB SUBtract Binary

The subtract function when executed will subtract from the contents of the register specified by the first reference, the content of the second reference (binary subtract). The result of the subtraction which is an absolute value will be placed in the register specified by the third reference.

The function will output power flow if the result of the subtraction is a negative value (register 1 is less than register 2), otherwise no power flow is generated.

Rules for inserting the function in a ladder diagram are the same as for the ADD function. Operation of the function is conditional upon receipt of power flow.

DISPLAY — SUBTRACT

```
  I0215   R0175   R0180  R0185                                    00438
+--] [--+[  A   -   B  =   C  ]+-------+-------+-------+-------+-------+-( )-+
          00000   00000  00000
```

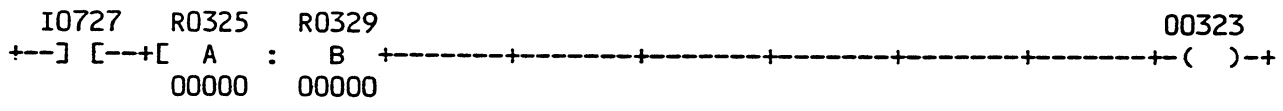
CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 2 1 5 ⊣⊢ →
L1 2	ARITH 2 ENTER R 1 7 5
	ENTER →
L1 3	R 1 8 +/− 0 ENTER →
L1 4	R 1 8 5 ENTER
L1 10	-0- RELAY 4 3 8 ENTER EDIT

COMPR COMPaRe

The compare function examines the 16-bit contents of two registers and if they are exactly equal will output power flow. Specifically, the contents of the register specified by the first reference are compared to the contents of the register specified by the second reference. If they are equal power flow is outputted, otherwise no power flow is generated.

Operation of the function is conditional upon receipt of power flow.

DISPLAY — COMPARE



CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 7 2 7 ⊣⊢ →
L1 2	ARITH 3 ENTER R 3 2 5
	ENTER →
L1 3	R 3 2 9 ENTER
L1 10	-0- RELAY 3 2 3 ENTER EDIT

MCR Master Control Relay

The master control relay function allows you to specify a quantity of coils and their associated logic to be skipped when the function is active. All coils within the scope of the function will be de-energized. The number of rungs within the scope of the function is determined by the constant operand specified which can be a decimal number from 0 to 255.

The rungs counted (following the function) to determine the scope of the MCR are those that end in a timer, counter, one-shot, latch or relay. If the constant operand entered is a zero it indicates that the scope of the MCR extends to the first ENDSW or RETURN function encountered. An active MCR function is terminated by an ENDSW or RETURN function.

When the MCR is active all of the outputs associated with relays within the scope of the function are turned off (if they are not overridden). The execution of all other functions is skipped. Execution of the MCR function is conditional upon receipt of power flow.

DISPLAY — MASTER CONTROL RELAY

```
  I0637    CONST
+--] [--+[ MCR ]+                                    (  )
           012
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 6 3 7 •⊣⊢• →
L1 2	MCR SKIP MOVE 1 ENTER 1 2 ENTER EDIT

SKIP SKIP

When implemented, this function allows you to specify all or part of a program to be skipped. When the SKIP is active, the execution of all functions that are within the scope of the function are skipped. The number of rungs in a program that are within the scope of the function are determined by the specified constant which can be a decimal number from 0 to 255. Any rung following the SKIP function ending in a timer, counter, one-shot, latch or relay is counted to determine the scope.

The SKIP will cause the logic associated with each coil within the scope of the function not to be solved, thereby leaving the coil in its ON or OFF state prior to implementing the function.

If a value of zero is entered as the constant, it indicates that the scope of the SKIP extends to the first ENDSW or RETURN function encountered after the SKIP. Execution of the SKIP function is conditional upon receipt of power flow.

DISPLAY — SKIP

```
 AI0422     CONST
+--] [--+[ SKIP ]+                                          ( )
           045
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	DP A, I, 4, 2, 2, ⊣⊢, →
L1 2	MCR SKIP END, 2, ENTER, 4, 5, ENTER, EDIT

ENDSW END of SWeep

When the End of Sweep function is encountered, it tells the CPU to enter its executive routine. An ENDSW must be the last function in a ladder diagram program. Two consecutive ENDSW functions indicate the absolute end of the program.

When programming subroutines with the extended function set, one ENDSW indicates the end of the main program. The function immediately following one ENDSW is the first word of the first user subroutine. No other functions can be entered in a line of logic with an ENDSW function. Execution of the ENDSW function is unconditional.

DISPLAY — END OF SWEEP	
+[ENDSW]+	
CURSOR POSITION	**KEYSTROKES[1]**
L1 1	INSERT MCR SKIP END, 3, ENTER, EDIT

NO OP NO OPeration

As the mnemonic of this function implies, no operation is performed by the function. A NO OP should be the first word of a program. It occupies one word of user program memory. The NO OP can be used as a filler to separate sections of a program so that they can be easily located by use of the search function. The NO OP function can be used to set a line of logic permanently on, for example, the run line of a timer. Execution of the NO OP function is unconditional.

DISPLAY — NO OPERATION	
+[NO OP]+	( )
CURSOR POSITION	**KEYSTROKES**
L1 1	INSERT NO OP SC DP, 1, ENTER, EDIT

SCREQ Serial Communication REQuest

This function is used to initiate communication with the Communication Control Module. The function requires a register reference which is the first of a block of registers containing the parameters required by the Communication Control Module. The contents of the specified register are displayed in hexadecimal beneath the function.

If a previous SCREQ was not honored, power flow is outputted, otherwise no power flow is generated. Execution of this function is conditional upon receipt of power flow.

DISPLAY — SERIAL COMMUNICATION REQUEST

```
  I0037    R0234
+--] [--+[SCREQ]+                                    ( )
          0000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 3 7 ⊣⊢ →
L1 2	NO OP SC DP 2 ENTER R 2 3 4
	ENTER EDIT

DPREQ Data Processor REQuest

This function is used to initiate communication with the Data Processor. The function requires a register reference which is the first of a block of registers containing all parameters necessary to the Data Processor. The contents of the specified register are displayed in hexadecimal.

If a previous DPREQ was not honored, power flow is outputted, otherwise no power flow is generated. Execution of this function is conditional upon receipt of power flow. See the Data Processor User's Guide for detailed use of this function.

DISPLAY — DATA PROCESSOR REQUEST

```
  I1023   R0314
+--] [--+[DPREQ]+                                    (  )
          0000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 1 +/- 0 2 3 ⊣⊢ →
L1 2	NO OP SC DP 3 ENTER R 3 1 4
	ENTER EDIT

ENTERING A PROGRAM

A simple program using the Basic function set is described below. The entry of the program into the PDT is shown in a step-by-step sequence of keystrokes. This will provide practice in entering a program using several of the functions in the Basic function set. By starting with simple programs and progressing to more complex programs you can easily increase your speed and efficiency in using the PDT.

The program you will enter is a time-of-day clock using the 24 hour system. The clock uses a 1.0 second interval timer preset to 60 with the count of the time accumulated in Register 1. The timer is started by a switch (I0101), counts to 60 and is self resetting, using its output O0111 to cause the reset.

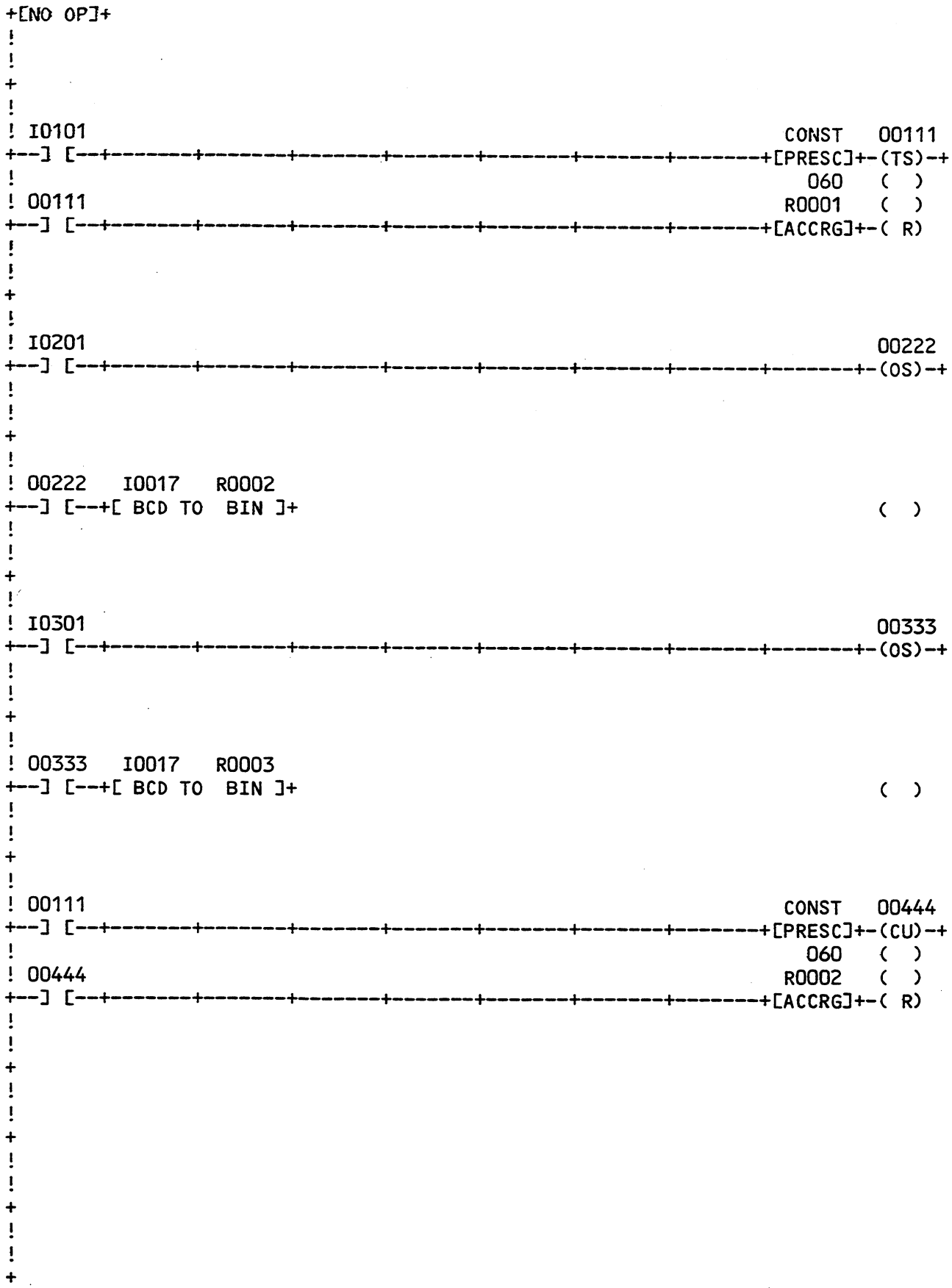
Two up-counters are used, the first one counts the minutes (0 to 60) while the second one counts the hours (0 to 24). The output of the timer (O0111) is used as the input to the minutes counter. Each time the timer accumulates a count of 60, its output is turned on which causes the minutes counter to advance by one count. This counter is preset to 60 and accumulates the number of minutes in Register 2. When the count reaches 60, the counter is reset by its output O0444.

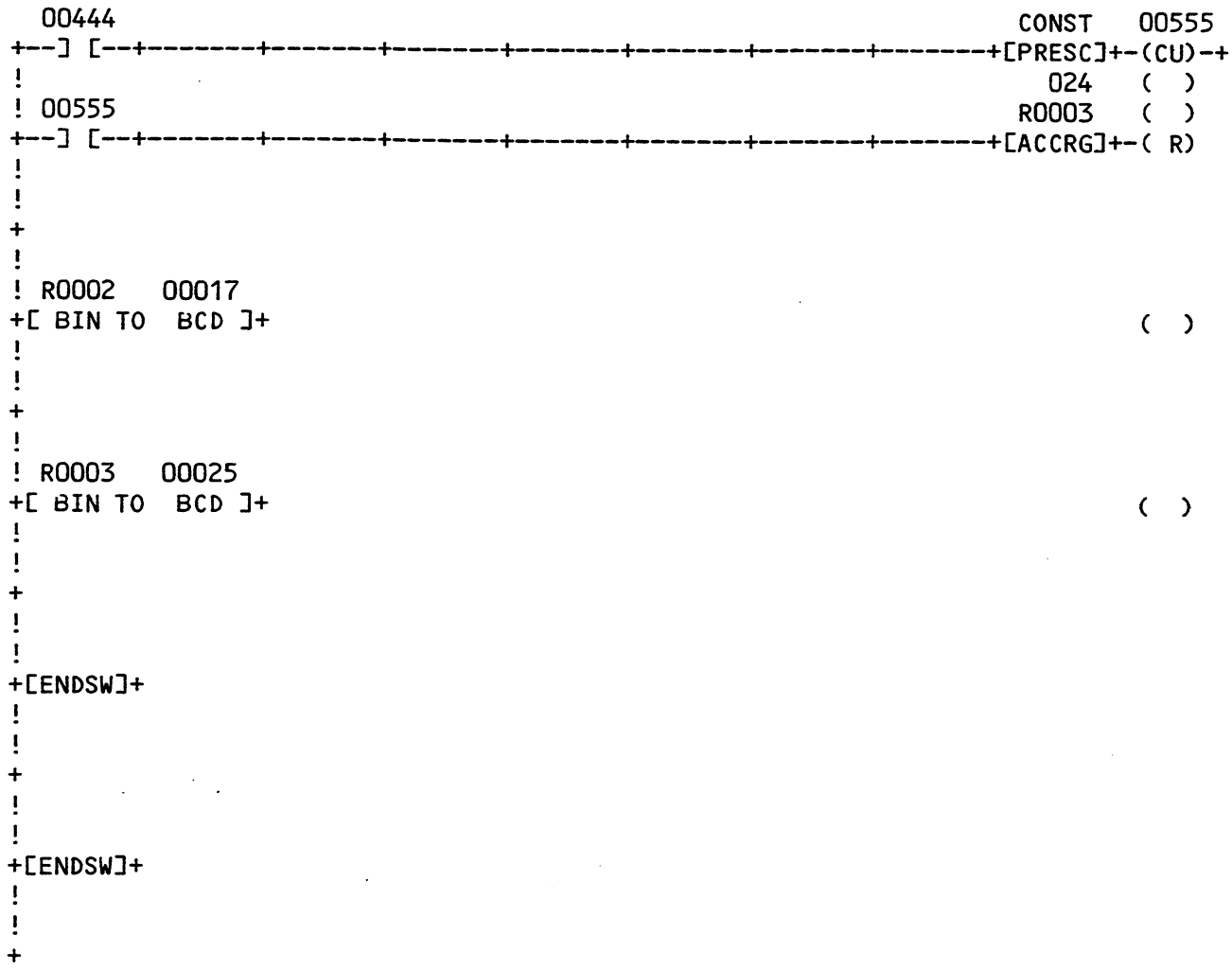
The output of the first up-counter (O0444) is the input to the second up-counter. This counter is preset to 24 and accumulates the number of hours in Register 3. Each time the minutes up-counter counts up to 60, its output turns on and causes the hours up-counter to advance by one count. When this counter counts up to 24, its output will turn on. The output (O0555) is used as a contact in the reset line which will reset the counter when the output is turned on.

The minutes and hours can be preset by entering a value (time) from a thumbwheel switch into the register (R2 and R3). The one-shots are used to cause the value to be entered into its respective register when one of the switches (I0201 or I0301) is closed. Switch I0201 will enter the minutes value into R2, while switch I0301 will enter the hours into R3. The BCD to BIN (BCDBI) function is used to convert the BCD value entered into the thumbwheel to Binary for entry into a register.

The last two functions, BIN to BCD (BIBCD), convert the binary value in the registers (accumulated minutes and hours) to BCD in order to be displayed through the outputs (O0017 to O0032). Outputs O0017 through O0024 will display the minutes, and outputs O0025 through O0032 will display the hours.

A printout of the program is provided, followed by the step-by-step keystroke sequence for entering the program.


[NO OP]
I0101
CONST
00111
[PRESC]
(TS)
060
( )
00111
R0001
( )
[ACCRG]
(R)
I0201
00222
(OS)
00222
I0017
R0002
[BCD TO BIN]
( )
I0301
00333
(OS)
00333
I0017
R0003
[BCD TO BIN]
( )
00111
CONST
00444
[PRESC]
(CU)
060
( )
00444
R0002
( )
[ACCRG]
(R)



PRINT-OUT FOR TIME-OF-DAY CLOCK PROGRAM

DISPLAY — TIME OF DAY CLOCK	
See the previous page for this display	
CURSOR POSITION	**KEYSTROKES**
	INSERT
L1 1	NO OP SC DP · 1 · ENTER · EDIT
	INSERT
L1 1	I · 1 · −/+ 0 · 1 · ⊣⊢
L1 9	PRE ACC · 1 · ENTER · 6 · +/− 0 · ENTER · →
L1 10	-0- TIMER 1.0 · 1 · 1 · 1 · ENTER
L2 1	HEX O · 1 · 1 · 1 · ⊣⊢
L2 9	PRE ACC · 3 · ENTER · R · 1 · ENTER · EDIT
	INSERT
L1 1	I · 2 · +/− 0 · 1 · ⊣⊢

CURSOR POSITION	KEYSTROKES
L1 10	-0- ONE SHOT, 2, 2, 2, ENTER, EDIT
	INSERT
L1 1	HEX O, 2, 2, 2, ⊣⊢, →
L1 2	SHIFT MOVE, 5, ENTER, I, 1, 7, ENTER
	→
L1 3	R, 2, ENTER, EDIT
	INSERT
L1 1	I, 3, +/− 0, 1, ⊣⊢
L1 10	-0- ONE SHOT, 3, 3, 3, ENTER, EDIT
	INSERT

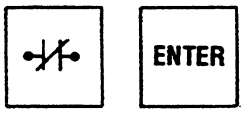
CURSOR POSITION	KEYSTROKES
L1 1	HEX O, 3, 3, 3, ⊣⊢, →
L1 2	SHIFT MOVE, 5, ENTER, I, 1, 7, ENTER, →
L1 3	R, 3, ENTER, EDIT, INSERT
L1 1	HEX O, 1, 1, 1, ⊣⊢
L1 9	PRE ACC, 1, ENTER, 6, +/− 0, ENTER, →
L1 10	-0- COUNT UP, 4, 4, 4, ENTER
L2 1	HEX O, 4, 4, 4, ⊣⊢
L2 9	PRE ACC, 3, ENTER, R, 2, ENTER, EDIT

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 4, 4, 4, ⊣⊢
L1 9	PRE ACC, 1, ENTER, 2, 4, ENTER, →
L1 10	-0- COUNT UP, 5, 5, 5, ENTER
L2 1	HEX O, 5, 5, 5, ⊣⊢
L2 9	PRE ACC, 3, ENTER, R, 3, ENTER, EDIT
	INSERT
L1 1	SHIFT MOVE, 4, ENTER, R, 2, ENTER, →
L1 2	HEX O, 1, 7, ENTER, EDIT
	INSERT
L1 1	SHIFT MOVE, 4, ENTER, R, 3, ENTER, →
L2 2	HEX O, 2, 5, ENTER, EDIT
	End of Program

The next step is to transfer the program which has now been entered into the PDT to the CPU. Complete the transfer by using the following key sequence.

To verify that the program has been entered correctly and is operating properly, the outputs can be connected to a visual display or the PDT can be used for verification. In order to observe the program while it is operating, the PDT must be on-line, the CPU in the RUN mode and the status message CPU EQUAL PDT must be displayed in the work area of the PDT.

If a switch is not physically connected to I0101, position the cursor under it and change the contact to a normally closed contact.



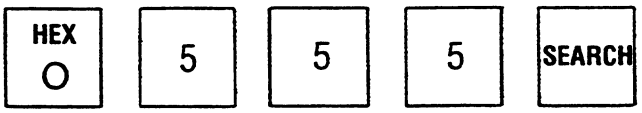
This key sequence changes I0101 to a normally closed contact.

With I0101 now a closed contact, the timer will start. To enter the current time the following sequence is shown as an example.

Enter the key sequence:

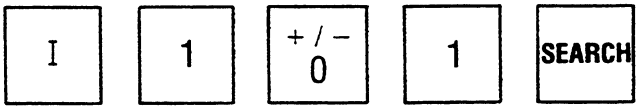
The minutes counter rung will now be displayed at the top of the screen. Move the cursor to line 2, position 9 and enter the current minute.

Enter the key sequence:



The hours counter rung will now be displayed at the top of the screen. Move the cursor to line 2, position 9 and enter the current hour.

To enter the current seconds, display the timer rung at the top of the screen by depressing the key sequence:



Move the cursor to line 2, position 9 and enter the current seconds. The time of day clock will now contain the current hours, minutes and seconds.

CHAPTER VI

Entering Extended Mnemonic Functions

INTRODUCTION

The intent of this chapter is to allow the user to become familiar with entering each of the Extended mnemonic functions. The functions were defined in Chapter 3.

The procedure used for presenting each function is as follows:

- General description of each keygroup (each key allows selection of a group of functions).
- Brief description of each function.
- Illustration of a rung of logic using the function. Functions requiring a permissive contact (conditional operation) will show the contact. A one-shot can be used with the functions to ensure one execution.
- Show a step-by-step sequence of keystrokes required for entering the illustrated rung of logic.

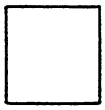
NOTE

For a more detailed description of the Extended function set, refer to the Series Six Application guide, GEK-25365.

In the following step-by-step sequences for entering functions, the symbols shown below will be used to denote the programming steps used in the examples.

Indicates the line number in a rung of logic or in a program.

Indicates position of the cursor in a line of logic.



Indicates a key to be depressed.

Most of the Extended functions are conditional in their operation; that is, they do not operate unless they receive power flow. This power flow can be directly from the left power rail, or via one relay contact, or a group of series and parallel contacts, or other mnemonic functions. The exact logic configuration is limited only by the ten by eight size of a single rung.

DATA MOVE GROUP

The Data Move functions provide a convenient means of moving data from one location to another. The source data is not altered since it is copied to the specific reference(s). The MOVE function moves the contents of a reference, MOVE RIGHT 8 and MOVE LEFT 8 operate on either the right half (least significant 8 bits) or left half (most significant 8 bits) of the references and the BLOCK move copies a group of 7 user memory words into a table beginning with the reference specified in the function.

MOVE

If active, the contents of reference A are copied into reference B and power flow is outputted.

DISPLAY — MOVE A to B

```
  I0335    R0426       R0609
+--] [--+[  A    MOVE    B  ]+                    (  )
         +00000      +00000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 3 3 5 ⊣⊢ →
L1 2	DATA MOVE 1 ENTER R 4 2 6
	ENTER →
L1 3	→
L1 4	R 6 +/- 0 9 ENTER EDIT

MOVE RIGHT 8

If the function is active, the right half (least significant 8 bits) of the contents of the first reference is copied into the right half (least significant 8 bits) of the second reference and power flow is outputted. The left half (most significant 8 bits) of the second reference is unchanged.

DISPLAY — MOVE RIGHT 8 BITS

```
  I0715   00449            00465
+--] [--+[  MOVE RIGHT 8 BITS  ]+                    ( )
          0000             0000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 7 1 5 ⊣⊢ →
L1 2	DATA MOVE 2 ENTER HEX O 4 4 9
	ENTER →
L1 3	→
L1 4	HEX O 4 6 5 ENTER EDIT

MOVE LEFT 8

If the function is active, the left half (most significant 8 bits) of the contents of the first reference is copied into the left half (most significant 8 bits) of the second reference and power flow outputted. The right half (least significant 8 bits) of the second reference is unchanged.

DISPLAY — MOVE LEFT 8 BITS

```
 AI0215   R0330          R0333
+--] [--+[  MOVE LEFT 8 BITS   ]+                              ( )
         0000            0000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	DP A, I, 2, 1, 5, ⊣⊢, →
L1 2	DATA MOVE, 3, ENTER, R, 3, 3, +/− 0
	ENTER, →
L1 3	→
L1 4	R, 3, 3, 3, ENTER, EDIT

BLOCK MOVE

If the function is active, the 7 constant operands (16 bit numbers) are copied into 7 locations in a table starting at the specified reference and power flow is outputted.

DISPLAY — BLOCK MOVE

```
  I0157   R0008
+--] [--+[                        BLOCK MOVE                                ] (  )
        +00000 -32768  -14569  -01123  +00089  +13924  +25183  +32767
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 1 5 7 ⊣⊢ →
L1 2	DATA MOVE 4 ENTER R 8 ENTER →
L1 3	SHIFT and +/− 0 3 2 7 6
	8 ENTER →
L1 4	SHIFT and +/− 0 1 4 5 6
	9 ENTER →
L1 5	SHIFT and +/− 0 1 1 2 3
	ENTER →
L1 6	8 9 ENTER →

CURSOR POSITION	BLOCK MOVE KEYSTROKES (Cont.)						
L1 7	1	3	9	2	4	ENTER	→
L1 8	2	5	1	8	3	ENTER	→
L1 9	3	2	7	6	7	ENTER	EDIT

SIGNED ARITHMETIC GROUP

The Signed Arithmetic group allows the use of functions which are performed based upon signed values, either positive or negative. This is very useful when performing operations using analog values. Zero is considered a positive value in the system.

The extended add and subtract (ADDX, SUBX) operate similar to the basic add and subtract except that the values being operated on are signed values (maximum −32,768 to +32,767). The double precision add and subtract (DPADD, DPSUB) functions use 2 consecutive words for each value allowing a maximum value of −2,147,483,648 to +2,147,483,647.

The multiply function (MPY) takes 2 single precision signed values and multiplies them with the result being a double precision signed value. The divide (DVD) function divides a double precision signed value by a single precision signed value with the result being a signed single precision quotient and remainder.

The last function in this group (GREATER THAN) will determine if one double precision signed value (reference A) is greater than the second double precision signed value (reference B).

DPADD

If active, the contents of reference A are added to the contents of reference B and the result is placed in reference C (A, B and C are signed double precision numbers). Each reference (A, B, C) requires two consecutive words. A 2's complement overflow will cause power flow to be outputted, otherwise no power flow is generated. A positive overflow will set reference C to +2,147,483,647 while a negative overflow will set reference C to −2,147,483,648.

DISPLAY — DOUBLE PRECISION ADD

```
  I0538   R0009            R0011            R0013
+--] [--+[  A      DPADD     B        =       C          ]+                ( )
         +0000000000      +0000000000      +0000000000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 5 3 8 ⊣⊢ →
L1 2	SIGN ARITH 1 ENTER R 9 ENTER →
L1 3	→
L1 4	R 1 1 ENTER →
L1 5	→
L1 6	R 1 3 ENTER EDIT

DPSUB

References A, B and C operate on signed double precision values with each reference using 2 words each. The contents of reference B are subtracted from the contents of reference A and the result is placed in reference C. References A or B can be a 31 bit constant. The range of values of a constant are −1,073,741,824 to +1,073,741,823. If the values are variable, the range can be −2,147,483,648 to +2,147,483,647. A 2's complement overflow will cause power flow to be outputted, otherwise no power flow is generated. A positive overflow will set reference C to +2,147,483,647 while a negative overflow will set reference C to −2,147,483,648.

DISPLAY — DOUBLE PRECISION SUBTRACT

```
  I1011    R0161            R0129            R0193
+--] [--+[  A      DPSUB     .B       =       C          ]+                ( )
         +0000000000       +0000000000      +0000000000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 1 +/− 0 1 1 ⊣⊢ →
L1 2	SIGN ARITH 2 ENTER R 1 6 1
	ENTER →
L1 3	→
L1 4	R 1 2 9 ENTER →
L1 5	→
L1 6	R 1 9 3 ENTER EDIT

ADDX

The extended ADD function operates on signed values. Each reference (A,B,C) uses one word of storage. If reference A or B is specified as a constant the maximum range of values is −16,384 to +16,383. If the values are variable, the range can be −32,768 to +32,767. If active, the contents of reference A are added to the contents of reference B and the result is placed in reference C. A 2's complement overflow will cause power flow to be outputted, otherwise no power flow is generated.

DISPLAY — EXTENDED ADD

```
  I0863    CONST    R0546    R0547
+--] [---+[  A   ADDX B    =    C  ]+                                    (  )
         +00555   +00000   +00000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 8 6 3 ⊣⊢ →
L1 2	SIGN ARITH 3 ENTER ±DEC C 5 5 5
	ENTER →
L1 3	R 5 4 6 ENTER →
L1 4	R 5 4 7 ENTER EDIT

SUBX

If active, the contents of reference B are subtracted from the contents of reference A and the result is placed in reference C. The extended SUBTRACT function operates on signed, single precision numbers. Each of the references (A,B,C) uses one word of storage. A 2's complement overflow will cause power flow to be outputted, otherwise no power flow is generated. Reference A or B can be a 15 bit constant (maximum range of values −16,384 to +16,383).

DISPLAY — EXTENDED SUBTRACT

```
 AI0324   R0124   R0125   R0126
+--] [--+[  A  SUBX B   =   C  ]+                    (  )
         +00000  +00000  +00000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	DP A, I, 3, 2, 4, ⊣⊢, →
L1 2	SIGN ARITH, 4, ENTER, R, 1, 2, 4
	ENTER, →
L1 3	R, 1, 2, 5, ENTER, →
L1 4	R, 1, 2, 6, ENTER, EDIT

MPY

If active, the contents of reference A (signed, single precision number) are multiplied by the contents of reference B (signed, single precision number) and the result which is a signed, double precision number is placed in reference C. If reference A or B is specified as a constant, their range of values is −16,384 to +16,383. Power flow is always outputted when this function is active.

DISPLAY — MULIIPLY

```
  00935    R0913    R0914    R0915
+--] [--+[  A  MPY  B   =   C          ]+                              (  )
        +00000   +00000   +0000000000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 9, 3, 5, ⊣⊢, →
L1 2	SIGN ARITH, 5, ENTER, R, 9, 1, 3
	ENTER, →
L1 3	R, 9, 1, 4, ENTER, →
L1 4	R, 9, 1, 5, ENTER, EDIT

DVD

If active, the contents of reference A (signed, double precision number) are divided by the contents of reference B (signed, single precision number). The quotient is placed in the reference specified by QUO and the whole number remainder is placed in the reference specified by REM. Power flow is outputted and no values altered if the quotient is too large to be stored in the QUO reference, including division by zero. Otherwise, no power flow is generated.

DISPLAY — DIVIDE

```
  I0045   R0401            R0403   R0404  R0405
+--] [--+[  A      DVD      B   =  QUO    REM ]+                    (  )
         +0000000000      +00000  +00000 +00000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 4 5 ⊣⊢ →
L1 2	SIGN ARITH 6 ENTER R 4 +/- 0 1
	ENTER →
L1 3	→
L1 4	R 4 +/- 0 3 ENTER →
L1 5	R 4 +/- 0 4 ENTER →
L1 6	R 4 +/- 0 5 ENTER EDIT

GREATER THAN

References A and B are both signed, double precision numbers and require 2 consecutive words of storage. If active, the contents of reference A are compared to the contents of reference B. If reference A is greater than reference B, power flow is outputted. If reference A is equal to or less than reference B, no power flow is generated.

DISPLAY — A GREATER THAN B

```
  I0099   R0655            R0665
+--] [--+[  A  GREATER THAN  B         ]+                          (  )
        +0000000000      +0000000000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 9 9 9 ⊣⊢ →
L1 2	SIGN ARITH 7 ENTER R 6 5 5
	ENTER →
L1 3	→
L1 4	R 6 6 5 ENTER EDIT

TABLE MOVE GROUP

The TABLE MOVE group of functions provides the capability of moving data into or out of a table or copying the entire contents of a table into another table. A table as used for data manipulation is defined as a group of consecutive storage locations. A table length (LEN) can consist of from 1 to 255 locations (register reference) or 1 to 63 locations (I, O, AI, AO) as specified when programming a function. If a table is specified as a group of registers, the table length begins at the starting reference plus 1. If the table is to be a group of discrete references (I, O, AI, AO), the table length begins at the starting reference plus 16 since each table location consists of a group of 16 contiguous discrete references.

The starting reference (TABLE) for a table contains a pointer which is the least significant half of the contents of the reference and is used as an index to the next location to be operated upon. When the function is active, each scan will cause the location specified by the pointer to be acted upon. The pointer will increment by one and be compared to the table length. The pointer is not counted as part of the table when calculating its length.

TABLE-TO-DEST

The word pointed to in the specified table is copied to the reference specified by DEST when the function is active. If the reference TABLE is a register, the contents of the register with an address equal to the address of TABLE plus the pointer is copied. If the reference TABLE is I, O, AI, or AO, the contents of the word whose address is equal to the address of the reference TABLE plus 16 times the value of the pointer is copied. Power flow is outputted when the last value of the table is accessed, otherwise no power flow is generated.

DISPLAY — MOVE TABLE TO DESTINATION

```
  00112    R0121    R0225  CONST
+--] [--+[TABLE-TO-DEST    LEN]+                    (  )
        00000  +00000    075
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 1, 1, 2, ⊣⊢, →
L1 2	TABLE MOVE, 1, ENTER, R, 1, 2, 1
	ENTER, →
L1 3	R, 2, 2, 5, ENTER, →
L1 4	7, 5, ENTER, EDIT

SRC-TO-TABLE

This function, when active, causes the 16 bit contents of the reference SRC to be copied to a location in a table. The least significant half of the contents of the reference TABLE points to one of the words in the table. The number of words in the table is specified by the reference LEN (the word containing the pointer is not counted as a word of the table). If the function is active, the pointer is incremented by one. If TABLE is a register, the contents of SRC are copied into the register with an address equal to the address of TABLE plus the pointer. If the reference TABLE is I, O, AI, or AO, the contents of SRC are copied to the location with an address equal to the address of TABLE plus 16 times the value of the pointer. Power flow is outputted when the last value of the table is loaded, otherwise no power flow is generated.

DISPLAY — MOVE SOURCE TO TABLE

```
  00333    08441    00713     CONST
+--] [--+[ SRC-TO-TABLE       LEN]+                    (  )
        +00000    00000       015
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 3, 3, 3, ⊣⊢, →
L1 2	TABLE MOVE, 2, ENTER, HEX O, 4, 4, 1
	ENTER, →
L1 3	HEX O, 7, 1, 3, ENTER, →
L1 4	1, 5, ENTER, EDIT

MOVE TABLE

References TBL A and TBL B each specify a starting location for two different tables. The reference LEN specifies the number of words in each table. If TBL A (source) and TBL B (destination) are I, O, AI or AO, the maximum LEN can be 63. If TBL A and TBL B are register references, the maximum LEN can be 255. When this function is active, the contents of TBL A are copied into TBL B and power flow is outputted.

DISPLAY — MOVE TABLE A TO TABLE B

```
 A00259   R0615          R0715  CONST
+--] [--+[TBL A    MOVE  TBL B    LEN ]+                          ( )
         0000            0000    064
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	DP A, HEX O, 2, 5, 9, ⊣⊢, →
L1 2	TABLE MOVE, 3, ENTER, R, 6, 1, 5
	ENTER, →
L1 3	→
L1 4	R, 7, 1, 5, ENTER, →
L1 5	6, 4, ENTER, EDIT

FOR FUTURE EXPANSION

LIST GROUP

The LIST group of functions provide a means of building a first in/first out (FIFO), last in/first out (LIFO) operation or sorting a list. The ADD-TO-TOP function copies a value into a list at the top location in that list and pushes any existing entries down.

The REM-FM-BOT function causes the last entry in a list to be copied to a specified location (FIFO). The REM-FM-TOP function copies the most recent entry in a list to a specified location (LIFO).

The SORT function examines the contents of a list (LIST A) and sorts the values algebraically with the smallest value at the top of the list. LIST B will contain a number associated with each value in LIST A when the sort is executed. The numbers in the second list will be in the same position in the lists as the value it is associated with. The number will correspond to the position of its associated value before the sort took place.

ADD-TO-TOP

The reference specified by LIST contains a pointer (least significant 8 bits) to the current bottom of the list. Maximum length of the list is specified by the reference LEN (the word containing the pointer does not count as a word in the list).

When the function is active and the pointer is less than LEN each item in the list is moved down to the next higher word address in the list and the contents of the reference SRC are copied into the top (first word) of the list (pointer is incremented by one).

When the pointer increment results in the pointer becoming equal to LEN power flow is outputted, otherwise no power flow is generated.

The example below shows a one-shot entered, then a normally-open contact of the one-shot being used as the permissive contact for the ADD-TO-TOP function. In this example since the permissive contact is O0444, it indicates that the contact is derived from a one-shot.

The example also shows that the pointer is at the 25th location (R0749) in a list that has a length of 64 locations (words).

DISPLAY — ADD-TO-TOP OF LIST

```
  I0313                                                                  00444
+--] [--+-------+-------+-------+-------+-------+-------+-------+-------+-(OS)-+
!
!
+
!
! 00444    R0625            R0724    CONST
+--] [--+[ SRC ADD-TO-TOP   LIST      LEN]+                                ( )
          00000            00025     064
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 3 1 3 ⊣⊢
L1 10	-0- ONE SHOT 4 4 4 ENTER EDIT
	INSERT
L1 1	HEX O 4 4 4 ⊣⊢ →
L1 2	LIST 1 ENTER R 6 2 5
	ENTER →
L1 3	→
L1 4	R 7 2 4 ENTER 2 5
	ENTER →
L1 5	6 4 ENTER EDIT

REM-FM-BOT

The least significant half of the contents of the reference LIST is a pointer to the current bottom of the list. The maximum length of the list is specified by the reference LEN. The word containing the pointer is not counted as a word of the list.

If the function is active and the pointer is less than LEN, the contents of the word at the bottom of the list are copied into the reference DEST, and the pointer is decremented by one. When the decrement results in the pointer being equal to zero, power flow is outputted, otherwise no power flow is generated.

If active and the pointer value is greater than LEN, an illegal condition exists for this function. The power flow is outputted, and the contents of the reference DEST remain unchanged.

The example shows O0420 as the permissive contact that allows the function to execute. The list length is 24 as specified by the constant reference LEN. Register R0225 contains the number 10 indicating that the current bottom of the list is the 10th location in the list (R0235). The next time the function is executed, the contents of R0235 will be copied to R0515 (DEST).

DISPLAY — REMOVE FROM BOTTOM OF LIST

```
  00420    R0225              R0515    CONST
+--] [--+[ LIST REM-FM-BOT DEST       LEN]+                    (  )
          00010            +00000    00024
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 4, 2, +/- 0, ⊣⊢, →
L1 2	LIST, 2, ENTER, R, 2, 2, 5
	ENTER, 1, +/- 0, ENTER, →
L1 3	→
L1 4	R, 5, 1, 5, ENTER, →
L1 5	2, 4, ENTER, EDIT

REM-FM-TOP

The least significant half of the reference LIST is a pointer to the current bottom of the list. The list length (maximum) is specified by the reference LEN (the word containing the pointer is not counted as a word of the list).

When the function is executed and the pointer value is less than the value specified by LEN, the contents of the word at the top of the list are copied into the location specified by the reference DEST. The top of the list is the word immediately following the reference LIST. At the same time each item in the list is moved up to the next lower word address in the list (R0050 to R0049, R0049 to R0048, etc.) and the pointer is decremented by one. When the decrement results in the pointer becoming equal to zero, it indicates that the list is empty and power flow is outputted, otherwise no power flow is generated.

In the following example a normally-open contact (O0044) is used as the permissive contact for the function. The reference for LIST is an Auxiliary Input (AI0049). Each location in the list contains 16 consecutive discrete elements (1 word), i.e. the first location in the table consists of AI0049 through AI0064. The number 21 indicates that the current bottom of the list is location 21. The list length specified by the reference LEN is 21. Each time the function is executed the contents of the first location in the list is copied to O0257 (DEST), each word in the list moves towards the top of the list and the pointer value decrements by one.

DISPLAY — REMOVE FROM TOP OF LIST

```
  00044   AI0049            00257    CONST
+--] [--+[ LIST REM-FM-TOP DEST      LEN]+                    ( )
        +00021             +00000     021
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 4, 4, ⊣⊢, →
L1 2	LIST, 3, ENTER, DP A, I, 4, 9
	ENTER, 2, 1, ENTER, →
L1 3	→
L1 4	HEX O, 2, 5, 7, ENTER, →
L1 5	2, 1, ENTER, EDIT

SORT

The reference LIST A and LIST B specify the starting location of two lists. The length of both of the lists is specified by the reference LEN. The maximum list length specified can be 63. Only register references can be used with this function.

When the function is executed, the contents of LIST A are sorted algebraically with the lowest value in the first location. After the sort LIST B contains a number that indicates the position (from 1 to LEN) where the corresponding word of LIST A was located before the sort took place (See the following example). Upon execution, power flow is outputted.

LIST A		LIST B	
R0129	+02178	R0145	XXXXX
R0130	+31642	R0146	XXXXX
R0131	−24567	R0147	XXXXX
R0132	+18921	R0148	XXXXX
R0133	+24113	R0149	XXXXX

Before Execution of SORT

LIST A		LIST B	
R0129	−24567	R0145	00003
R0130	+02178	R0146	00001
R0131	+18921	R0147	00004
R0132	+24113	R0148	00005
R0133	+31642	R0149	00002

After Execution of SORT

DISPLAY — SORT ASCENDING

```
  00055    R0129              R0145   CONST
+--] [--+[LIST A    SORT    LIST B     LEN]+                    (  )
        +00000              +00000     005
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 5, 5, •⊣⊢•, →
L1 2	LIST, 4, ENTER, R, 1, 2, 9
	ENTER, →
L1 3	→
L1 4	R, 1, 4, 5, ENTER, →
L1 5	5, ENTER, EDIT

MATRIX GROUP

The matrix group of functions consists of 5 items; logical AND, logical Inclusive OR, logical Exclusive OR, logical Invert and Masked Compare. The AND, IOR and EOR operate logically on 2 separate matrices and store the result in a third matrix. All matrix operations are completed in their entirety prior to examining the next function in the ladder diagram.

The matrix functions operate on the individual bits in a word and not on the numerical value stored in a location. A matrix length (LEN) can specify up to 255 locations, thus allowing a matrix to consist of 4080 bits (255 × 16). If I, O, AI, or AO tables are used, maximum length is 64 words (1024 bits).

The AND, IOR and EOR functions operate logically on each comparable group of 16 bits starting at the references (A,B) specified by the function. The bit patterns resulting from the operation are stored in the locations starting at reference C. The truth table for these operations is shown below.

TABLE 1—Truth Table-Matrix Logical Operations

REFERENCE		RESULT IN C		
A	B	AND	IOR	EOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

The Invert function inverts all 16 bits (0 to 1, 1 to 0) in each location in a matrix and places the result in a comparable location in a second matrix. All of the above functions examine the referenced matrices non-destructively with the resultant bits stored in the result matrix.

The Compare (Masked Compare) function compares one matrix (INPUT) to another (REF) and identifies miscompares. Individual bits can be skipped by setting a pattern in (MASK). The contents of the reference FAULT will indicate the location of an unmasked miscompare.

AND

References A, B, and C specify the starting locations for each matrix. The 16 bits of each word in matrix A are logically AND-ed with the 16 bits of each corresponding word in matrix B and the resultant bit pattern is placed in the corresponding word in matrix C.

If the resultant bit pattern in matrix C consists of all zeros, power flow is outputted. Any other bit pattern in matrix C will cause no power flow to be generated. The contents of matrix A and matrix B are unchanged by the operation. The contents of A, B and C are displayed in hexadecimal format.

DISPLAY — MATRIX AND

```
  00035    R0385    R0449    R0513    CONST
+--] [--+[  A  AND  B    =    C        LEN  ]+                    (  )
          0000     0000     0000     064
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 3, 5, ⊣⊢, →
L1 2	MATRX, 1, ENTER, R, 3, 8, 5
	ENTER, →
L1 3	R, 4, 4, 9, ENTER, →
L1 4	R, 5, 1, 3, ENTER, →
L1 5	6, 4, ENTER, EDIT

IOR

References A, B, and C specify the starting locations for 3 matrices with the length of each matrix specified by the reference LEN.

When the function is executed, the contents of the corresponding words in matrices A and B are logically inclusive OR-ed. The result is placed in matrix C, and power flow is outputted. The contents of matrix A and matrix B are unchanged by the operation. The contents of references A, B, and C are displayed in hexadecimal.

DISPLAY — MATRIX INCLUSIVE OR

```
  A00629  R0641   R0769   R0897   CONST
+--] [--+[  A  IOR  B   =   C      LEN]+                    (  )
   0000   0000    0000    0000     128
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	DP A, HEX O, 6, 2, 9, ⊣⊢, →
L1 2	MATRX, 2, ENTER, R, 6, 4, 1
	ENTER, →
L1 3	R, 7, 6, 9, ENTER, →
L1 4	R, 8, 9, 7, ENTER, →
L1 5	1, 2, 8, ENTER, EDIT

EOR

References A, B, and C specify the starting locations for 3 matrices with the length of each matrix specified by the value entered in the reference LEN.

When the function is active, the contents of the corresponding words in matrices A and B are logically exclusive OR-ed. The result is placed in matrix C. If the resulting bit pattern in matrix C consists of all zeros, power flow is outputted, otherwise no power flow is generated. The contents of matrix A and matrix B are unchanged by this function.

DISPLAY — MATRIX EXCLUSIVE OR

```
  00416    AI0001   AI0257   AI0513   CONST
+--] [--+[   A  EOR  B    =    C       LEN]+                    (  )
          0000     0000     0000     016
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 4, 1, 6, ⊣⊢, →
L1 2	MATRX, 3, ENTER, DP A, I, 1, ENTER
	→
L1 3	DP A, I, 2, 5, 7, ENTER, →
L1 4	DP A, I, 5, 1, 3, ENTER, →
L1 5	1, 6, ENTER, EDIT

INV

References A and B specify the starting locations for 2 matrices with the length of each matrix specified by the value entered in the reference LEN.

When the function is active, the contents of each word (16 bits) in matrix A are logically inverted and stored in the corresponding word of matrix B. Whenever this function is active, power flow is outputted. The contents of matrix A are unchanged.

DISPLAY — MATRIX INVERT

```
  00038   R0129   R0177   CONST
+--] [--+[  A  INV  B      LEN]+                    (  )
          0000    0000     032
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 3, 8, ⊣⊢, →
L1 2	MATRX, 4, ENTER, R, 1, 2, 9
	ENTER, →
L1 3	R, 1, 7, 7, ENTER, →
L1 4	3, 2, ENTER, EDIT

COMPARE

The references INPUT, REF (Reference) and MASK are each a bit matrix with a length specified by the reference LEN. If INPUT, REF, and MASK are register references, each matrix can contain up to 4080 bits (255 × 16).

The contents of the reference INPUT are compared to the contents of the reference REF. Any bits to be skipped when not necessary to the compare operation are masked by setting the corresponding bit in the matrix MASK to a one. The content of the reference FAULT are used as a pointer into the bit matrices and to store the location of an unmasked compare.

When the function is active, the contents of FAULT are incremented. FAULT is set to 1 if the resulting contents are greater than the number specified by LEN × 16. The function starts with the bit specified by the content of FAULT and compares the corresponding bits in INPUT and REF. FAULT increments after each compare.

When a miscompare is identified, the corresponding bit is tested to see if it has been masked. If it is masked, the comparison continues until an unmasked miscompare is identified, or the end of the matrix is reached.

When an unmasked miscompare is found power flow is outputted and the corresponding mask bit in the matrix MASK are set to a one and execution terminates for the current scan. The location of the unmasked miscompare is indicated by the contents of FAULT.

An example of the masked compare function is shown below. For simplicity, only the first location in the matrices specified by the INPUT, REF, and MASK references is shown. The references and length shown in the step-by-step entry of the function are used in the example.

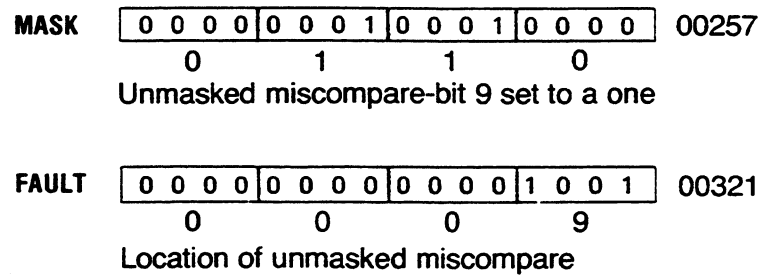
Before execution of masked COMPARE

16 bits of the word starting at 00129 will be compared to the 16 bits of the word starting at 00193.

Bit 5 masked (set to 1)—will be skipped.

Contents of FAULT word are 0.

After execution of masked COMPARE



Upon execution, a miscompare occurs at bit 5; however, since bit 5 is masked, the compare continues until an unmasked miscompare is found in bit 9. Bit 9 in MASK is set to a one. The location of the unmasked miscompare is set in FAULT. During the next execution FAULT will increment until the end of the matrix is reached. (Contents of FAULT will be 65 (LEN × 16) +1) or another miscompare is found.

DISPLAY — MASKED COMPARE

```
  00088              00129   00193   00257   00321     CONST
+--] [--+[COMPARE  INPUT     REF    MASK    FAULT       LEN]+          ( )
                     0F00    0E10    0010    00000       004
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 8, 8, ⊣⊢, →
L1 2	MATRX, 5, ENTER, →
L1 3	HEX O, 1, 2, 9, ENTER, F, +/− 0
	+/− 0, ENTER, →
L1 4	HEX O, 1, 9, 3, ENTER, E, 1
	+/− 0, ENTER, →
L1 5	HEX O, 2, 5, 7, ENTER, 1, +/− 0
	ENTER, →
L1 6	HEX O, 3, 2, 1, ENTER, →
L1 7	4, ENTER, EDIT

BIT MATRIX GROUP

The bit matrix group contains 4 functions: Bit Set/Sense, Bit Clear/Sense, Shift Right N Bits and Shift Left N Bits. The Bit Set and Bit Clear functions can operate on a single bit in a specified matrix. A bit can be turned ON, turned OFF, or sensed. Sensing a bit does not alter the bit state.

The Shift Right and Shift Left N Bits functions provide a means of shifting a matrix of bits to the left or to the right a specified number of bits.

BIT SET

The SET/SENSE BIT function can operate upon any specified single bit in a matrix. If there is power flow coming into the function, the bit set function will be performed; if not, the sense function will be performed. The reference MATRIX specifies the start of a matrix consisting of a length of LEN times 16 bits. The contents of the reference BIT point to a bit in the matrix.

The bit in the matrix specified by BIT is set to a one (turned on) when power flow is received and the reference bit is greater than zero and less than or equal to the reference LEN × 16. Power flow is outputted. If the referenced BIT is zero or greater than the reference LEN, the matrix does not change and no power flow is generated.

If there is no power flow when the function is executed, and the reference BIT is greater than zero and less than or equal to the reference LEN × 16, the bit in the matrix specified by BIT is sensed. A one bit will cause power flow to be outputted and a zero bit results in no power flow generated.

DISPLAY — SET/SENSE BIT

```
  00367    R0125            R0327    CONST
+--] [--+[ BIT    SET      MATRIX      LEN]+                    ( )
          00632             0000       048
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 3, 6, 7, •⊣⊢•, →
L1 2	BIT MATRX, 1, ENTER, R, 1, 2, 5
	ENTER, 6, 3, 2, ENTER, →
L1 3	→
L1 4	R, 3, 2, 7, ENTER, →
L1 5	2, 8, ENTER, EDIT

BIT CLEAR

The CLEAR/SENSE BIT function operates on a specific bit in a matrix in a manner similar to the BIT SET/SENSE function except that the specified bit is cleared (turned off) rather than turned on.

When the function is executed and power flow is received and the reference BIT is greater than zero and less than or equal to the reference LEN × 16, then the specified bit in the matrix is cleared and no power flow is generated.

If there is no power flow when the function is executed and the reference BIT is greater than zero and less than or equal to the reference LEN × 16, then the bit in the matrix specified by BIT is sensed (the bit state is not altered). A one bit causes power flow to be outputted, and a zero bit results in no power flow generated.

DISPLAY — CLEAR/SENSE BIT

```
  00212   R0015          R0025   CONST
+--] [--+[ BIT  CLEAR   MATRIX     LEN]+                    (  )
          00328           0164     255
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 2, 1, 2, •|•, →
L1 2	BIT MATRX, 2, ENTER, R, 1, 5, ENTER
	3, 2, 8, ENTER, →
L1 3	→
L1 4	R, 2, 5, ENTER, →
L1 5	2, 5, 5, ENTER, EDIT

SHIFT RT

This function uses a shift register with a length specified by the reference LEN (× 16). If LEN is 4, the shift register would be 64 bits in length. The content of the reference N indicate the number of one bit shifts to be executed each scan. A binary 5 in the reference N indicates that each bit in the matrix starting with the reference MATRIX will be shifted 5 bits to the right each time the function is executed.

When the function is active and N is greater than zero and less than or equal to LEN (× 16), then all bits in the matrix are shifted the specified number of bits to the right (toward the least significant bit). Zeros are shifted into the most significant bit of the last word in the matrix. The last bit shifted out of the matrix (least significant bit of the lowest addressed word) controls the power flow. A one bit causes power flow and a zero bit results in no power flow generated.

The shift does not take place if N is zero or greater than LEN (× 16) and no power flow is generated.

DISPLAY — SHIFT RIGHT N BITS

```
  00035              I0129   I0257   CONST
+--] [--+[ SHIFT RT  N      MATRIX    LEN]+                    (  )
                    00016    0000     012
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 3, 5, •||•, →
L1 2	BIT MATRX, 3, ENTER, →
L1 3	I, 1, 2, 9, ENTER, 1, 6
	ENTER, →
L1 4	I, 2, 5, 7, ENTER, →
L1 5	1, 2, ENTER, EDIT

SHIFT LEFT

This function operates similar to the SHIFT RIGHT N BITS function, except that all of the bits (LEN × 16) in a matrix starting at the reference MATRIX are shifted N (number specified by the contents of reference N) bits to the left.

When the function is active and N is greater than zero and less than or equal to LEN (× 16), all bits in the matrix are shifted to the left (towards the most significant bit) the specified number of bits. In the example each bit is shifted 32 bits to the left. The matrix in the example starts at Register R0705 and is 64 Registers in length.

Zeroes are shifted into the least significant bit of the first word (lowest reference) in the matrix. The last bit shifted out of the most significant bit of the highest referenced word controls the power flow. A one bit causes power flow and a zero bit results in no power flow generated.

The shift does not take place if N is zero or greater than LEN (× 16) and no power flow is generated.

DISPLAY — SHIFT LEFT N BITS

```
 01012            R0113   R0705   CONST
+--] [--+[SHIFT LEFT N   MATRIX     LEN]+                    (  )
                 00032    0000     064
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 1, +/− 0, 1, 2, ⊣⊢, →
L1 2	BIT MATRX, 4, ENTER, →
L1 3	R, 1, 1, 3, ENTER, 3, 2
	ENTER, →
L1 4	R, 7, +/− 0, 5, ENTER, →
L1 5	6, 4, ENTER, EDIT

CONTROL GROUP

The control group of functions provides a means of altering or controlling the normal sequence of logic operations in a Programmable Controller. The functions included in this group of functions are: Do Subroutine, Return, Suspend I/O, Do I/O and Status.

DO SUB

A subroutine is a group of functions that are located or stored in the user memory immediately after the main ladder diagram program. Subroutines are separated from the main program by a single END-OF-SWEEP function. A subroutine is terminated by a RETURN function. A program can contain up to 16 subroutines. Each subsequent subroutine in a sequence of subroutines must be terminated by a RETURN function.

Example of Subroutine sequence:

Main Ladder Diagram Program
END-OF-SWEEP
Subroutine 1
RETURN
Subroutine 2
RETURN

Subroutine 16
RETURN
END-OF-SWEEP
END-OF-SWEEP

A subroutine can be initiated in two ways. A subroutine is executed each time an active DO SUB function is encountered in the main body of a ladder diagram program. The DO SUB function is described herein. The second method of initiating a subroutine is by the generation of an interrupt. There are 8 user interrupts available in the main I/O chain and 8 user interrupts available in the auxiliary I/O chain. The interrupts are triggered through an Interrupt Input module which is described in the Series Six Installation and Maintenance Manual, GEK-25361.

The Do Subroutine function when encountered in the main program (and active) will cause the subroutine specified by the reference N (1 to 16) to be executed the number of consecutive times as is indicated by the 8 least significant bits of the reference REPS (repetitions).

NOTE

Normal subroutines should not modify the contents of the REPS reference. Unexpectedly long scan times may result that exceed the hardware watchdog timer if this reference is modified during a subroutine.

The 8 most significant bits of the reference REPS is used to count the number of executions. This byte is set to zero before the first execution, then increments by one after each execution and when the count is equal to the least significant byte, the most significant byte is reset to zero and control returns to the main program.

NOTE

A DO SUB function cannot be embedded in a subroutine.

DISPLAY — DO SUBROUTINE

```
 00670            CONST  R0259
+--] [--+[DO SUB    N     REPS]+                    (  )
                   012   0000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 6, 7, +/− 0, ⊣⊢, →
L1 2	CON TROL, 1, ENTER, →
L1 3	1, 2, ENTER, →
L1 4	R, 2, 5, 9, ENTER, EDIT

RETURN

The RETURN function is executed unconditionally. When it is encountered at the end of a subroutine, it returns control to the main program if the required repetitions have been performed. Execution of the main program then continues at the point following the last function that was executed prior to entering the subroutine.

RETURN functions cannot be embedded in the main program, they can only be used in subroutines.

DISPLAY — RETURN	
+[RETURN]+ ( )	
CURSOR POSITION	**KEYSTROKES**
L1 1	INSERT CON TROL 2 ENTER EDIT

SUSPEND I/O

When this function is active, the CPU will skip servicing of the I/O's. All physical outputs will be held in their state previous to the execution of the function and the input table will not be updated with data from physical input devices (unless the program contains active DO I/O functions). When active, power flow is outputted.

When inactive, normal I/O scanning occurs and no power flow is generated. Any active suspend I/O, in the scan will cause I/O to be suspended. To continue to be effective, suspend I/O's must continue to receive power flow.

DISPLAY — SUSPEND I/O

```
  00851
+--] [--+[ SUSPEND I/O ]+                                    (  )
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 8, 5, 1, ⊣⊢, →
L1 2	CON TROL, 3, ENTER, EDIT

DO I/O

When this function is active, the contents of START and END (if valid) specify the beginning and ending I/O points which are to be serviced immediately. The reference START will be rounded to the nearest lower byte boundary and END will be rounded to the nearest upper byte boundary.

When active and the contents of START and END are valid, power flow is outputted. In order to be valid the contents of START and END must be in the range of 1 to 1000 and the contents of START must be less than or equal to the contents of END.

This function allows new input data to be obtained upon command and new data to be provided to outputs without depending upon the end of scan I/O servicing. DO I/O is also useful to allow the I/O scan to be completed in a shorter duration by scanning only those I/O points pertinent to the user program. The time savings can be significant on small systems requiring rapid scanning.

DISPLAY — DO I/O

```
  00015            R0025   R0026
+--] [--+[ DO I/O START     END]+                    ( )
                   00356   00872
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	HEX O, 1, 5, ⊣⊢, →
L1 2	CON TROL, 4, ENTER, →
L1 3	R, 2, 5, ENTER, 3, 5, 6
	ENTER, →
L1 4	R, 2, 6, ENTER, 8, 7, 2
	ENTER, EDIT

STATUS

When active, the function is used to enable (open) or disable (close) DPU (Data Processor Unit) and/or CCM (Communication Control Module) windows by moving the contents of the reference to the CPU scratch pad (to a status indication byte). The contents (word) of the reference are logically AND-ed with 30H (Hexadecimal) and the results written to the scratch pad. The DPU and CCM windows and Alarm #2 (Advisory alarm) are affected as shown below.

REFERENCE CONTENTS (WORD) AFTER WRITE TO SCRATCH PAD	DPU WINDOW	CCM WINDOW	ALARM #2
0 Hexadecimal	open	open	off
10 Hexadecimal	closed	open	on
20 Hexadecimal	open	closed	on
30 Hexadecimal	closed	closed	on

When the function is inactive, the contents of the upper byte of the reference are set to zero and the contents of the status indication byte are moved to the lower byte of the reference. The contents of the lower byte provide a status indication for the DPU, CCM and Alarm #2. The status indications are as shown above (closed = disabled = error condition).

DISPLAY — STATUS

```
  I0099   R0256
+--] [--+[    STATUS    ]+                                    (  )
         0000
```

CURSOR POSITION	KEYSTROKES
	INSERT
L1 1	I 9 9 ⊣⊢ →
L1 2	CON TROL 5 ENTER R 2 5 6
	ENTER EDIT

NOTE

For additional information and samples of programs using the Extended functions, refer to the SERIES SIX APPLICATIONS GUIDE, GEK-25365.

APPENDIX A
Glossary of Terms

Address—A number or combination of letters and numbers assigned to a specific memory location within a PC used to access that location.

Analog—A numerical expression of physical variables such as rotation and distance to represent a quantity.

AND (Logical)—A mathematical operation between bits whereby all bits must be a 1 in order for the result to be a 1.

ASCII—An eight-level (7 bits plus 1 parity bit) code commonly used for exchange of data which forms the American Standard Code for Information Interchange.

Basic Mnemonic Function Set—A group of functions primarily used for programming basic relay logic in the Series Six.

Baud—A unit of data transmission speed equal to the number of code elements (bits) per second.

BCD (Binary Coded Decimal)—A 4-bit system in which individual decimal digits (0 through 9) are represented by 4-bit binary numerals; e.g., the number 43 is represented by 0100(4) 0011(3) in the BCD notation.

Binary—A numbering system using only the digits 0 and 1 which can represent (on-off, open-closed, etc.) physical conditions in a PC system.

Bit—An acronym for Binary DigIT. The smallest unit of information in the binary numbering system, represented by either a 0 or 1. The smallest division of a PC word.

BUSY—A message generated by the PDT and displayed in the work area indicating that a previously requested function is being processed. When the BUSY message is removed from the work area, the next requested function can be processed.

Byte—A group of binary digits operated on as a single unit. In the Series Six a byte is 8 bits.

CPU (Central Processing Unit)—The central device or controller that interprets user instructions, makes decisions and executes the instructions based on the decisions.

Composite Video—Circuitry within the PDT which allows connection to an external digital video monitor.

Constant—A predetermined value stored in a register which when acted upon by an event does not change.

Counter—A circuit internal to a PC which when controlled by a user programmed function controls other devices according to a preset number of on/off transitions.

Cross Reference Table—A table within the PDT which keeps track of the usage of I, O, AI, AO and R references within a user program.

CRT—An acronym for Cathode Ray Tube. Commonly used when referring to the screen in the PDT which displays the ladder diagrams, tables, etc. associated with the PC system.

Cursor—A position indicator viewed on the screen of the PDT that shows the position of the next data entry.

Data Processor—A peripheral unit which when used with a CPU in a Series Six system provides the ability to perform complex functions, generate messages and access large quantities of data without interrupting or slowing down the logic processing in the PC.

Discrete—Consisting of individual, distinct things such as bits, characters, or circuit components. Also refers to "ON-OFF" type I/O modules.

Double Precision Number—A twos complement value which when used in a Series Six system consists of two words (31 bits plus sign), thereby allowing a variable value from −2,147,483,648 to +2,147,483,647.

Edit Mode—A mode of operation in the PDT whereby the existing user program in PC memory may be modified or new rungs of logic may be added.

EOR (Logical Exclusive OR)—A mathematical operation between bits which when two bits are operated on will result in a 1 if either, but not both bits are a 1. If the bits are the same the result will be 0.

Extended Mnemonic Function Set—A group of functions used in programming the Series Six which increase its flexibility and allow its adaptation to a greater number of control and process functions.

Firmware—A series of software instructions contained in ROM (Read Only Memory) which are used for internal processing. These instructions are transparent to the user.

Full Duplex—A method of data transmission which allows data to be simultaneously transmitted and received in both directions.

Half Duplex—A method of data transmission which allows data to be communicated in only one direction at a time.

Hard Copy—A printed document listing the contents of a portion of memory such as a ladder diagram listing.

Hardware—All of the mechanical, electrical and electronic devices that make up a programmable controller and its application.

Hexadecimal—A numbering system having 16 as a base, represented by the digits 0 through 9, then A through F.

IOR (Logical Inclusive OR)—A mathematical operation between bits whereby any bit being a 1 will cause the result to be a 1.

Input Devices—Devices that as a result of their mechanical or electrical action supply data to a Programmable Controller. Typical devices are limit switches, push buttons, pressure switches, digital encoders and analog devices.

INV (Logical Invert)—A mathematical operation on bits in a matrix which will cause all ones to be replaced by zeros and all zeros to be replaced by ones. The result is stored in a second matrix.

I/O—Commonly used abbreviation for Input/Output.

I/O Scan—A method by which the CPU monitors all inputs and controls all outputs within a prescribed time.

Ladder Diagram—A representation of control logic relay systems. The user programmed logic is expressed in relay equivalent symbology.

Latch—A PC function that causes an output to stay on and remain on even if power or the input is removed. Referred to as a retentive function.

Line—A group of horizontal logic elements connected in series; up to nine can be accommodated in the Series Six format, plus a coil. Lines can be connected by vertically short circuits to form a rung of logic.

List—A group of consecutive storage locations in PC memory with a specified length and used for data manipulation. Data is accessed from either the top or bottom of a list.

Logic—A method of solving complex problems through the repeated use of simple functions that define or represent basic concepts. OR and AND are commonly used basic logic functions.

Mask—The process of setting internal program controls to prevent certain events from occurring.

Matrix—A group of consecutive 16 bit storage locations with a specified length in which the individual bits are operated on by various functions.

Memory—A grouping of circuit elements that have data entry, storage and retrieval capability.

Memory Protect—A hardware capability that prevents memory from being altered by an external device. This capability in the Series Six system is controlled by a keyswitch on the CPU.

Microsecond (μs)—One millionth of a second. 1×10^{-6} or 0.000001 second.

Millisecond (ms)—One thousandth of a second. 1×10^{-3} or 0.001 second.

Mnemonic—An abbreviation given to a function, usually an acronym formed by combining initial letters or parts of a series of words.

Monitor Mode—A mode of operation in the PDT which allows an operating program to be monitored. No changes in the program can be made from this mode.

Nanosecond (ns)—One billionth of a second. 1×10^{-9} or 0.000000001 second.

Off-Line Mode—A PDT mode of operation which allows entry of programs and changes to programs prior to insertion into the CPU memory. Allows programs to be developed and entered at a location remote from the CPU.

On-Line Mode—A PDT mode of operation in which the operating program can be observed and changes may be made to the program while it is operating.

One-Shot—A discrete reference to a ladder diagram element, usually a coil in a line of logic which is energized (turned on) for one scan of the Programmable Controller, when there is an off to on transition of the input.

Operand—A quantity on which a mathematical operation is performed.

Output—Information transferred from the CPU, through a module for level conversion, for control of an external device.

Output Devices—Physical devices such as motor starters, solenoids, indicators, etc. that receive data from the Programmable Controller.

Override—A method for causing inputs or outputs not to be changed by contact logic in a user program.

PC—Commonly used abbreviation for Programmable Controller.

Peripheral Equipment—A unit that can communicate with a Programmable Controller; e.g., Program Development Terminal, Minicartridge Tape Unit or printer.

Preset—A numerical value entered into a register which establishes a limit for a counter or timer network. A coil will energize when this value is reached.

Program—A sequence of functions entered into a Programmable Controller to be executed by the CPU for the purpose of controlling a machine or process.

Program Development Terminal (PDT)—A peripheral device used for creating and modifying programs or for monitoring programs. Entered programs can be transferred to the CPU or stored on tape. The PDT consists of a CRT, keyboard, and an optional internal tape unit.

Programmable Controller—A solid-state control system which receives inputs from user supplied control devices such as switches and sensors, implements them in a precise pattern determined by instructions stored in the user memory, and provides outputs for control of user supplied devices such as relays and motor starters.

PROM—An acronym for Programmable Read Only Memory. A retentive digital storage device programmed at the factory and not readily alterable in the field.

RAM—An acronym for Random Access Memory. A solid-state memory that allows individual bits to be stored and accessed. This type of memory is volatile; i.e., stored data is lost under no power conditions, therefore a battery backup is required.

Register—In the Series Six, a portion of RAM memory used for temporary storage of numerical values and for bit manipulation.

Relay Line—A line of logic in a ladder diagram used to simulate the effect of mechanical relays. The coil in a relay line will be energized when continuity is complete from the left to the right vertical rail of a ladder diagram.

Retentive Output—An output that will remain in its last state, even though power has been removed.

Reverse Video—An area on a CRT display whereby the background is light and displayed characters are dark. Some of the PDT cursors are displayed in this manner.

RS-232C—A standard specified by the Electronic Industries Association (EIA) for the mechanical and electrical characteristics of the interface for connecting Data Terminal Equipment (DTE) and Data Communications Equipment (DCE).

Rung—A sequence or grouping of PC functions that control one output. One or more rungs form a ladder diagram.

Scan—The technique of examining or solving all logic steps specified by the program in a sequential order from the first step to the last.

Scratch Pad Display—A display viewed on the PDT screen which shows information about certain system parameters and operating conditions of the CPU.

Significant Bit—A bit that contributes to the precision of a number. The number of significant bits is counted beginning with the bit contributing the most value referred to as the Most Significant Bit (MSB) and ending with the bit contributing the least value referred to as the Least Significant Bit (LSB).

Subroutine—A sequence of functions which provide a means of altering the order in which logic is solved or scanned within the PC. In the Series Six, a subroutine is stored in user memory immediately after the main program. Subroutines are initiated by a function or by an associated interrupt input.

Supervisor Display—A display viewed on the PDT screen which provides an operator with a means of selecting a particular operation or action.

Table—A group of consecutive storage locations in PC memory with a specified length and used for data manipulation. Data may be accessed randomly in a table.

Thumbwheel Switch—A rotating numeric switch which can be used for inputting numeric data to a PC.

Timer—An internal PC function which when controlled by a user programmed function will control the operating cycle of other devices by a preset and accumulated time interval.

Toggle—A circuit or device being capable of assuming either one of two stable states when activated by a switching device.

Twos Complement—A form of binary arithmetic used to perform binary subtractions with addition techniques. The twos complement negative of a binary number is formed by inverting each bit in the number and adding 1 to the result. For example: the twos complement of 0011 is 1100 + 1 or 1101.

Unlatch—A PC function that causes an output to turn off no matter how briefly the function is enabled. In the Series Six the unlatch contacts are located in the second line of the Latch function.

Watchdog Timer—A timer internal to the CPU used to insure that a scan is completed at least once each 200 milliseconds.

Word—A group of binary digits in sequence operated on as a single unit. In the Series Six a word is 16 bits.

Work Area—A 16 character wide area on the right side of the PDT display. Contains CPU and PDT status information and is used for display of keystroke data, error messages, and operator prompts.

Series Six™ Programmable Controllers Advanced Functions

Supplement

GE Fanuc Automation

SEPTEMBER 1984 GEK-25399

This document is based on information available at the time of its publication. While efforts have been made to be accurate, the information contained herein does not purport to cover all details or variations in hardware and software, nor to provide for every possible contingency in connection with installation, operation, and maintenance. Features may be described herein which are not present in all hardware and software systems. GE Fanuc Automation assumes no obligation of notice to holders of this document with respect to changes subsequently made.

GE Fanuc Automation makes no representation or warranty, expressed, implied, or statutory with respect to, and assumes no responsibility for the accuracy, completeness, sufficiency, or usefulness of the information contained herein. No warranties of merchantability of fitness for purpose shall apply.

Copyright© 1987 GE Fanuc Automation North America, Inc.

Warning, Caution, and Notes As Used In This Publication

WARNING

Warning notices are used in this publication to emphasize that hazardous voltages, currents, temperatures, or other conditions that could cause personal injury exist in this equipment.

CAUTION

Caution notices are used where equipment might be damaged if care is not taken.

In situations where inattention could cause either personal injury or damage to equipment, a Warning notice is used.

NOTE

Notes merely call attention to information that is especially significant in understanding and operating the equipment.

ADVANCED FUNCTIONS

INTRODUCTION

This supplement to the Series Six Programming Manual, GEK-25362, describes the Advanced Functions required for programming and using the 8K of Registers available in a Series Six programmable controller.

DESCRIPTION

The Series Six Programmable Controllers, have been enhanced by the addition of 8K of user storage locations (registers) when using CPU Software Level 104 or higher. This increase in storage is designed for large data requirements typically associated with recipes, standards, and reports. The expanded storage is *not* obtained at the expense of logic memory. The Series Six family can provide up to 32K for your program *plus* 8K for numerical storage, *plus* 320 words for I/O status (total 40.3K of 16-bit words). To utilize this expanded storage, the following equipment must be available.

1. Advanced Functions - CPU software version 104 or higher, available on Logic Control module IC600CB502**C** or higher revision.

2. 8K Register module - IC600CB507**A** or higher revision.

3. Updated PDT - software revision 16 or higher, available on PDT PROM Memory module IC600PB552**B** or higher revision.

All of these prerequisites were in production as of July, 1984. The Advanced software (and 8K Register module) will be the standard offering in lieu of the Extended level on Models 600 and 6000 starting in the fourth quarter of 1984. Update kits for previously shipped units are also available. One kit is required for each CPU (Kit No. 44A286314-G01) and another for each PDT (Kit No. 44A286315-G01). The new software will provide the capabilities within the Series Six family as shown in Table 1 on the following page.

The RPU, DPU and CCM1 module will only be able to access the first 1024 registers. The CCM2 module (IC600CB516**C** or higher revision) will be able to access all 8192 registers. When making a tape of a program (or reloading the program) through the STR™-LINK Tape Loader , and it is desired to record the content of registers above R1024 (e.g., R1025-R8192), the CCM2 module must be used. Tapes made or loaded by an updated PDT (version 16 or higher) will be able to access all 8192 registers and do not require any CCM module. The ASCII/BASIC module can also address all 8192 registers.

All application programs developed on previous CPUs (basic or extended) or with previous PDTs are fully compatible with this new level of functionality. These previous programs can be directly loaded into an Advanced CPU without any change required by the operator, and the CPU will operate as it previously did. Programs developed with an Advanced CPU can also be directly placed in Basic or Extended CPUs as long as all of the functions used are availble with those CPUs.

™Trademark of Electronic Processors Inc.

Table 1.
CPU CAPABILITIES

MODEL	PART NUMBER	FUNCTIONS	LOGIC MEMORY	CAPACITY		REGISTER STORAGE
				INPUT	OUTPUT	
60 (Includes Logic Memory)	IC600CP201	Basic	2K	256	256	256
	IC600CP211	Extended	2K	256	256	256
	IC600CP231	Basic	4K	1000	1000	1024
	IC600CP241	Extended	4K	1000	1000	1024
600 (Excludes Logic Memory, maximum shown)	IC600CP301	Basic	8K	1000	1000	1024
	IC600CP321	Advanced	8K	1000	1000	8192
6000 (Excludes Logic Memory and Auxiliary I/O Control, maximum shown)	IC600CP421	Advanced	32K	2000	2000	8192

The Advanced level CPU adds one Table MOVE function that has the unique ability to address all 8192 registers. Other functions such as timers, counters, adds, subtracts, matrices or moves are unchanged and operate only upon the first 1K of registers (R0001-R1024). This new fun tion is selected as part of the TABLE MOVE group. When this key is depressed on the PDT, the following selection is presented to the operator:

1. TABLE-TO-DEST.
2. SRC-TO-TABLE
3. MOVE TABLE
4. MOVE TABLE EX.

The fourth function is the one added to create the Advanced level CPU.

This Extended Table Move typically allows groups of registers to be moved into and out of the added storage (see Figure 1). A single function can move up to 255 registers in one direction (into or out of added registers). Multiple functions can be programmed in any CPU equipped with level 104 or higher software. When active, each execution of the MOVE TABLE EX function will require 16 μsec plus 9.35 μsec per register moved. The maximum single function time is 2.4 μsec to move 255 registers.

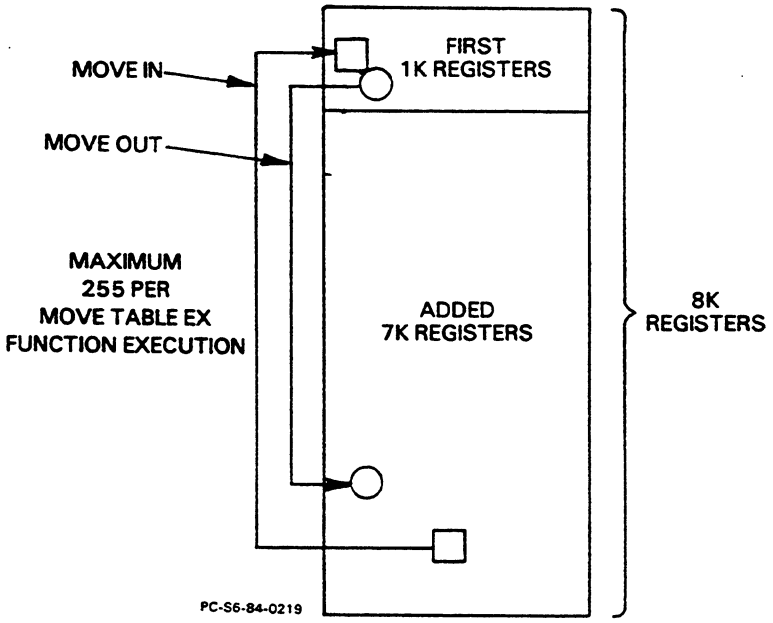


Figure 1.
BLOCK DIAGRAM OF EXTENDED MOVE FUNCTION

The function itself can be programmed similar to other Table Moves with access to more than just registers (see Figure 2). It requires four columns in the PDT display and can be located anywhere on the screen where sufficient space is available. It requires five words of logic memory per function programmed. Power flow must be received by the function before data will be moved. Without power flow no data is moved and the scan time impact is reduced to 2.5 μsec per function. When power flow is received, the entire table (1 to 255 registers or words as specified by the length value) will be moved. Both the FROM and TO elements can be programmed with any valid reference such as registers (R001-R8192), Inputs (I0001-I1017), Outputs (O0001-O1017), Auxiliary Inputs (AI0001-AI1009) and Auxiliary Outputs (AO0001-AO1009).

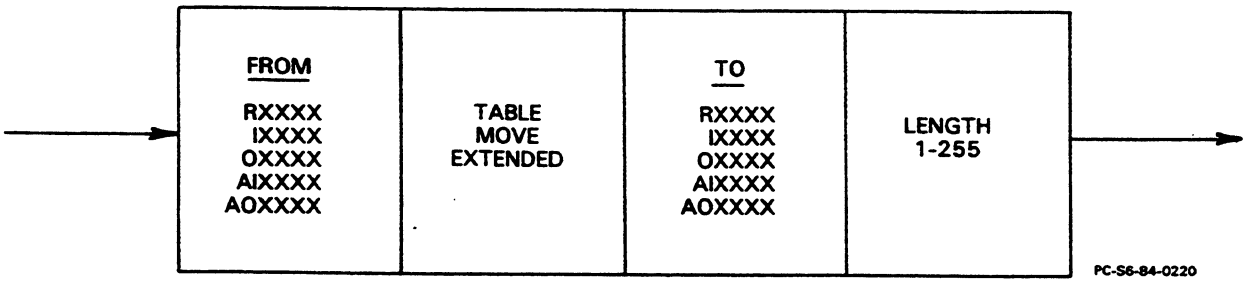


Figure 2.
FORMAT OF TABLE MOVE EXTENDED

See LADDER LOGIC EXAMPLES, page 6, for programming examples.

The length value is a fixed number between 1 and 255, entered at the time of programming . The PDT will verify the proper selection of references and length to ensure a valid program at the time the rung is programmed. The PDT will automatically correct discrete references to the next *lowest* valid number during editting. Regardless of reference selected, the move function will always operate in groups of 16 bits; thus the length defines exactly the number of registers or, when multiplied by 16, the number of discretes moved. The PDT will ensure that there is sufficient memory within the CPU to support the move function. For example, if the length is 10, the following references will be invalid: R8188, I0945 and AO1009. To correct this invalid situation, the PDT will require the operator to either change the reference or shorten the length. The PDT does *not* check that one table overlaps another previously programmed by the operator.

Although the main use of the Extended Table Move is to access the additional registers, it can be programmed to move data within the first 1K registers (i.e., replaces Table Move) or within or between I/O status storage. By selecting proper references, the following moves can be selected:

FROM	TO
First 1K Registers	Added Registers
Added Registers	First 1K Registers
Added Registers	Added Registers
First 1K Registers	First 1K Registers
I/O	All Registers
All Registers	I/O
I/O	I/O
Aux I/O	All Registers
All Registers	Aux I/O
Aux I/O	Aux I/O
I/O	Aux I/O
Aux I/O	I/O

In order to program this new function, the scratchpad must be properly configured. When ON-line and a CPU to PDT transfer has been specified, the scratchpad will be updated by the connected CPU. If this CPU has both an 8K Register Memory module installed *and* version 104 or higher software, the PDT will allow programming the Extended Move function. Otherwise, the PDT will not allow entry of this function.

When in the OFF-Line mode, the PDT will assume whatever value the last CPU or internal program had contained. If it is powered up in the OFF-LINE mode, the scratchpad will default to 8K of registers and Extended version 104 software. In the PDT, the Advanced software is indicated by the name Extended and level 104. To change the register quantity, the cursor is placed on the register size line and the Enter key depressed. For each depression, the display will change once from 8192 to 16,384 then to 256, to 1024 and back to 8192. The value 16,384 is for future expansion only; there is currently no hardware available to support 16K registers.

If the software level selected was Basic, the scratchpad register selection would be limited to 256 or 1024. Programs that are developed with a scratchpad setting of 8192 registers and/or Advance functionality, must be loaded into a CPU with these parameters. The PDT will inhibit loading of such a program into a lessor CPU. If a program within the PDT uses the Extended Move or register values greater than 1024, selecting smaller scratchpad register size will be inhibited by the PDT until the program is changed. Registers used implies tables (e.g., R1021 with a table length of 10 refers to R1021-R1030) and double precision references. The Extended Move will also allow tables to overlap from the first 1K register into the added 7K registers. Similar verification of register use is made with other software levels of the CPU; however, changing to the 256 register size will not check for double precision references to R0256, implying use of R0257.

In addition to the ability to program the Extended Move function, the PDT version 16 software also changes its operation in two other areas, Printer Port and Tape Loader Diagnostics. When a ladder diagram printout or portion thereof is requested, the messages "BUSY" and "PRESS ON/OFF TO SUSPEND" will be placed on the screen. If the printer (or similar device) does not respond within 1 second, the "BUSY" will be replaced with "DEVICE NOT READY". Whenever the printer responds, the "BUSY"" message will be restored. If the operator wants to suspend the printout while it is in progress, the ON/OFF key should be depressed. Suspending the printout may be desirable if new paper needs installing, the paper must be rethreaded, or the ribbon should be changed. The ENTER key will restart the printout after it is suspended.

To abort a printout at any time, the CLEAR key can be selected. The printout is then terminated and the CRT display reverts to that of the Supervisor level. An operation similar to the SUSPEND I/O function can be generated at the printer device. The PDT will recognize the XOFF character (HEX 13) received through the RS-232 port as a Suspend command and stop outputting characters. The XON character (HEX 11) will restore normal printout; if the XON is not received within 1 second of the XOFF, the CRT will display "DEVICE NOT READY." If this message appears for extended time periods, the connection between the PDT and the printer should be examined for a hardware fault.

The verification of the internal tape loader operation has been expanded to detect additional errors. If a fault is detected, the operator will be alerted by an error message and a diagnostic code. This code will be useful as an aid to GE Programmable Control Service personnel in identifying the cause of the error and recommending corrective action. The following telephone number should be used on a 24-hour a day basis, if such problems occur: (804) 978-5747.

LADDER LOGIC EXAMPLES

Following are ladder logic examples using the Extended Table Move function to move data into or out of the added Register storage area of memory.

1. The first example, when executed would move the contents of 255 Registers, starting with Register 0001 to 255 consecutive Registers, starting with Register 7000.

```
  I0152     REG                        REG    CONST
+--] [--+[  A    MOVE  TBL  EXT   B       LEN   ]+          ( )
        00001                       07000   255
```

2. The second example will cause a group of Inputs from the Input Status Table starting with I0097 to be moved to a group of Registers beginning with Register 2000. Since the Extended Table Move function operates on groups of 16 consecutive bits, the number of Inputs moved is 56 x 16 Or 896.

```
  00039   I0097                      REG    CONST
+--] [--+[  A    MOVE  TBL  EXT   B       LEN   ]+          ( )
        00000                       02000   056
```

3 This example will cause the data in a group of 23 Registers, starting with Register 5002, to be moved to the Auxiliary Output Table Beginning with Auxiliary Output AO0497. Upon execution, the data will occupy 368 consecutive locations in the Auxiliary Output Table.

```
 AI0599     REG                      A00497 CONST
+--] [--+[  A    MOVE  TBL  EXT   B       LEN   ]+          ( )
        05002                       00021   023
```