

2000-OIL

Operator Interface Language

P/N 96327-001E

Xycom Revision Record

<i>Revision</i>	<i>Description</i>	<i>Date</i>
A	Manual Released	6/90
B	Manual Updated	8/91
C	Manual Updated (incorporated PCN #160)	7/93
D	Not Released	
E	Updated Revision Level	10/97

Trademark Information

Brand or product names are registered trademarks of their respective owners.

Copyright Information

This document is copyrighted by Xycom Incorporated (Xycom) and shall not be reproduced or copied without expressed written authorization from Xycom.

The information contained within this document is subject to change without notice.



xycom

Technical Publications Department
750 North Maple Road
Saline, MI 48176-1292
734-429-4971 (phone)
734-429-1010 (fax)

WARNING

Dangerous voltages are present within all Xycom Industrial Terminals. These voltages will linger after all electrical power is turned off. Use caution whenever the unit is opened. Avoid touching high-voltage areas within the terminal. Do not work alone.

WARNING

The FRAGILE Cathode Ray Tube (CRT) is exposed when the front panel is opened. Wear safety glasses to protect eyes in case of accidental breakage of the CRT. Internal coating of CRT is extremely TOXIC. If exposed RINSE IMMEDIATELY and consult a physician.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
1	INTRODUCTION	
1.1	Introduction	1-1
1.2	About OIL	1-2
1.3	Manual Structure	1-3
1.4	Firmware Specifications	1-4
2	INSTALLATION	
2.1	Introduction	2-1
2.2	Back Panel	2-1
2.3	Firmware Installation	2-2
2.4	Controller Board	2-5
2.5	EPR0M Installation	2-6
2.6	Verifying Installation	2-8
2.7	Power-Up or Reset	2-8
3	BASIC CONCEPTS	
3.1	Introduction	3-1
3.2	Menu Hierarchy	3-1
3.3	Main Menu	3-3
3.4	Configuration Menus	3-4
3.4.1	Primary Serial Port Configuration Menu	3-4
3.4.2	Secondary Serial Port Configuration Menu	3-7
3.4.3	Miscellaneous Configuration Menu	3-8
3.4.4	Real Time Clock Configuration Menu	3-11
3.5	Program Utilities Menu	3-12
3.5.1	Using the Editor	3-13
3.5.2	Program Execution	3-16
3.5.3	Copying Program Blocks	3-17
3.5.4	Backup/Restore Programs	3-18
3.5.4.1	Backup Programs	3-18
3.5.4.2	Restore Programs	3-19
3.5.4.3	Verify Programs	3-19
3.5.4.4	Print Programs	3-20
3.5.5	Using the Directory	3-21
3.5.6	Erase	3-22
3.5.7	Search	3-22
3.5.8	Move	3-22
3.6	Backup/Restore Menu	3-23
3.7	Diagnostics Menu	3-24
3.7.1	Complete Test	3-25
3.7.2	Continuous Test	3-25

CHAPTER	TITLE	PAGE
3	BASIC CONCEPTS	
3.7.3	RAM Test	3-26
3.7.4	ROM Checksum	3-27
3.7.5	EEPROM Test	3-27
3.7.6	Real Time Clock Test	3-27
3.7.7	RS-232 Serial Loopback Test	3-27
3.7.8	RS-485/Multidrop Serial Loopback Test	3-28
3.7.9	Printer Port Test	3-29
3.7.10	Parallel Input Port Test	3-29
3.7.11	Matrix Keyboard Loopback Test	3-30
3.7.12	Beeper Test	3-30
3.7.13	Battery Test	3-30
3.7.14	Dipswitch Test	3-31
3.7.15	Character Attributes	3-31
3.7.16	CRT Crosshatch Pattern	3-31
3.7.17	CRT Brightness Pattern	3-31
3.7.18	Touch Screen Test	3-32
3.7.19	.CM Command	3-32
3.8	Terminal Mode Menu	3-33
3.9	Set Password	3-33
4	OPERATOR INTERFACE LANGUAGE (OIL)	
4.1	Introduction	4-1
4.2	Command List	4-3
4.3	Overview of the Operator Interface Language (OIL)	4-9
4.4	Transferring Data to/from the Workstation	4-11
4.4.1	Data Registers	4-12
4.4.2	Transferring Data to the Workstation	4-12
4.4.3	Requesting Data from the Workstation	4-13
4.4.4	Transmitting Data from the Workstation	4-13
4.5	Programming Rules	4-15
4.6	Commands	4-17
4.6.1	User Interface Considerations	4-17
4.6.2	Screen Text	4-17
4.6.3	Register Value Display	4-18
4.6.4	Position Cursor	4-19
4.6.5	Label	4-20
4.6.6	Add to Register	4-21
4.6.7	AND	4-21
4.6.8	Beep	4-22
4.6.9	Draw Box	4-22
4.6.10	Clear Buffer	4-23
4.6.11	Clear Line	4-23
4.6.12	Clear Bit	4-23

CHAPTER	TITLE	PAGE
4	OPERATOR INTERFACE LANGUAGE (OIL) (cont.)	
4.6.13	Clear Screen	4-24
4.6.14	Clear Window	4-24
4.6.15	Down	4-24
4.6.16	Decrement Data Register	4-25
4.6.17	Define Zone or Zones	4-25
4.6.18	Divide Register	4-26
4.6.19	Display String	4-26
4.6.20	Execute Program and Reset Subroutine Counter	4-27
4.6.21	Execute Subprogram Block	4-27
4.6.22	Execute Program Block	4-28
4.6.23	Exit OIL Program	4-29
4.6.24	Draw Filled Box	4-29
4.6.25	Go	4-30
4.6.26	Horizontal Bar Left	4-31
4.6.27	Horizontal Bar Right	4-32
4.6.28	Draw Horizontal Line	4-33
4.6.29	Conditional	4-34
4.6.30	If Bit	4-36
4.6.31	If String	4-37
4.6.32	Increment Data Register	4-39
4.6.33	Exit ONKEY Subroutine Program	4-39
4.6.34	Keypad Status	4-40
4.6.35	Left	4-40
4.6.36	Remainder	4-41
4.6.37	Multiply Register	4-42
4.6.38	New Line	4-42
4.6.39	NOT	4-43
4.6.40	ONKEY	4-43
4.6.41	ONTOUCH	4-44
4.6.42	OR	4-44
4.6.43	Plot	4-45
4.6.44	Pause	4-46
4.6.45	Right	4-47
4.6.46	Reset Attributes	4-47
4.6.47	Restore Attributes	4-47
4.6.48	Restore Cursor	4-48
4.6.49	Save Attributes	4-48
4.6.50	Save Cursor Position	4-48
4.6.51	Set Attributes	4-49
4.6.52	Set Bit	4-51
4.6.53	Stop Program Execution	4-52
4.6.54	Store String	4-52
4.6.55	Subtract from Register	4-53
4.6.56	Transfer System Characteristics	4-54
4.6.57	Exit ONTOUCH Subroutine Program	4-55
4.6.58	Transfer Data	4-55

CHAPTER	TITLE	PAGE
4	OPERATOR INTERFACE LANGUAGE (OIL) (cont.)	
4.6.58.1	Pictures	4-56
4.6.58.2	Data Types	4-57
4.6.58.3	Transferring Text Characters	4-58
4.6.58.4	Numeric Data Types	4-61
4.6.59	Transfer Text	4-69
4.6.60	Transfer Touch Screen Data	4-69
4.6.61	Up	4-70
4.6.62	UNPLOT	4-70
4.6.63	Vertical Bar Up	4-71
4.6.64	Vertical Bar Down	4-72
4.6.65	Draw Vertical Line	4-73
4.6.66	XOR	4-74
4.7	Sample Program	4-75
4.7.1	Executing the Program Block	4-77
5	PROGRAMMING	
5.1	Introduction	5-1
5.2	Data Registers	5-1
5.3	Reading the Time and Date	5-3
5.4	Reading Data from the Keyboard or Serial Port	5-4
5.4.1	Input Data Queues	5-4
5.4.2	Queue Status Registers	5-7
5.4.3	Dummy Destination NO	5-8
5.4.4	Menu Program	5-8
5.4.5	I/O to the Serial Port	5-9
5.5	Nested Screens	5-10
5.6	Incrementing Data Registers	5-14
5.7	Keyboard Translation	5-14
5.8	Performance Hints	5-14

CHAPTER	TITLE	PAGE
6	REMOTE COMMANDS	
6.1	Introduction	6-1
6.2	Serial Remote Commands	6-2
6.2.1	Reset	6-4
6.2.2	Execute Program	6-4
6.2.3	Transfer to Data Register	6-5
6.2.4	Transfer from Data Register	6-6
6.2.5	Load Time and Date	6-7
6.2.6	Load All Registers	6-7
6.2.7	Send Password	6-8
6.2.8	Execute Subprogram Block	6-8
6.2.9	Receive Program Block	6-9
6.2.10	Send a Number of Registers	6-9
6.2.11	Receive a Number of Registers	6-10
6.2.12	Receive Extra Registers	6-10
6.2.13	Backup Programs	6-11
6.2.14	Verify Program	6-12
6.3	How Remote Commands are Processed	6-13

Table of Contents

CHAPTER	TITLE	PAGE
7	SAVING/LOADING/VERIFYING OIL PROGRAMS	
7.1	Saving OIL Programs	7-1
7.2	Restoring Screen Programs	7-3
7.3	Verifying Screen Programs	7-4

APPENDICES

2000-XX

*Only the appendix that discusses the shipped firmware will be included in the documentation package.

B	OIL COMMANDS
C	TOUCH SCREEN

LIST OF FIGURES

FIGURE	TITLE	PAGE
2-1	2000 Back Panel	2-2
2-2	Removing the Cover Plate	2-3
2-3	Installing the Firmware Chip	2-4
2-4	2000 Controller Board	2-5
3-1	Menu Hierarchy	3-2
3-2	Main Menu	3-3
3-3	Serial Port Configuration Menu	3-4
3-4	Miscellaneous Configuration Menu	3-8
3-5	Real Time Clock Configuration Menu	3-11
3-6	Program Utilities Menu	3-12
3-7	Diagnostics Menu	3-24
3-8	Serial Port Test Plugs, RS-232	3-28
3-9	Serial Port Test Plugs, RS-485	3-29
4-1	The Block Command Transfer	4-13
4-2	Screen Created by Sample Program	4-76
5-1	Keyboard Input Queue Example (1)	5-5
5-2	Keyboard Input Queue Example (2)	5-6
5-3	Nested Blocks	5-10
5-4	Nested Screen Example	5-12
7-1	PC/AT to 2000 Connection	7-1
7-2	PC/XT to 2000 Connection	7-2

LIST OF TABLES

TABLE	TITLE	PAGE
3-1	Cursor Movement Keys	3-14
4-1	OIL Commands by Section	4-1
4-2	OIL Commands by Function	4-3
4-3	Registers #1 - #11	4-12
5-1	Registers #8 - #11	5-7
6-1	Serial Remote Commands	6-2

1.1 INTRODUCTION

The 2000 Series OIL firmware are software upgrades that are installed into Xycom's 2000 Industrial Workstation. These firmware upgrades bring to the 2000 Industrial Workstation Operator Interface Language (OIL). OIL is an easy-to-use programming language, designed to simplify the creation and enhancement of screen displays for industrial applications. OIL allows you to set character attributes, position the cursor, draw boxes and lines, manipulate data registers, clear a line or the screen, execute other program blocks from within a program, make decisions, repeat an operation, and much more. OIL also allows the workstation to communicate with a wide range of Programmable Logic Controllers (PLCs), provides a time-of-day clock, and 126 Kbytes of program memory.

Most of the firmware options give access to the data table area on a specific PLC via OIL. The firmware interfaces with the individual PLC or network of PLCs via a direct connect. Protocols and commands for these firmware options are discussed in the appendices.

The Xycom 2000 Series has many firmware options available*, such as:

- Operator Interface Language (OIL)
- OIL/Allen Bradley Data Highway
- OIL/Modicon MODBUS
- OIL/Westinghouse
- OIL/General Electric Series 1/Series 6 CCM Port
- OIL/Square D SY/MAX
- OIL/Texas Instruments 305
- OIL/Siemens
- OIL/A-B T3
- OIL/TI 405
- OIL/Omron
- OIL/Mitsubishi
- OIL/GE Series 90
- OIL/TI 500/505

*Firmware options are always being updated. Ask your Xycom representative for details.

These firmware upgrades provide the 2000 Workstation with memory, communications, and programming capabilities. Each workstation is shipped with 126 Kbytes of battery-backed CMOS program memory.

1.2 ABOUT OIL

OIL programs allow easy formatting of the screen displays for a specific application. A single OIL program could be used to format several complete screen displays, or a single screen display could have portions formatted by several different OIL programs. There need not be a one-to-one relationship between screen displays and OIL programs. The simplest OIL program consists of a sequence of displayable characters. When such an OIL program is executed, the characters stored in the program are just displayed on the screen.

The program memory feature allows full screen editing, program execution, program copying, and OIL program transmission capabilities. The program memory provides room for up to 255 program blocks. A program consists of sequences of displayable text and graphic characters combined with OIL commands.

A 2000 Workstation with OIL is very versatile because a program block can do much more than display previously entered data and figures on the screen. In the course of execution, a program block can read data directly from the serial ports, data registers, the keypad, or the keyboard. It can then arrange this data in any way and display it on the screen. This allows a screen display to be constantly updated based on data being received by the workstation.

A host computer can have its data displayed simply by sending new data to the workstation, which can read the new data, send it, format it, and flash it on the screen. Since the workstation now handles all the logic of interfacing with the operator, the host computer can focus on what it does best -- handling the logic of its process or machine control.

In addition, execution of a program block can write data to the serial port. Execution of such a program block can be initiated from the keyboard, from the optional touch screen or keypad, from the host computer, or from another program block.

500 sixteen-bit data registers are provided (up to 9999 extra registers can be selected from the configuration menu) to hold any data, such as current process variable values, set points, valve position, and/or motor speed, etc. These additional registers can hold the same kind of information that other annunciator systems require, such as panel lights, meters, and simple message display units, for presentation to an operator. Likewise, registers can hold operator input data that otherwise would have required thumb wheel switches and push buttons.

Included in the data register architecture are several dedicated registers which provide time and date data from the on-board clock/calendar, and status of all the keypad, keyboard, touch screen, and serial port data buffers.

The PLC interface occurs through the primary serial port. This leaves the secondary serial port available to connect to a printer or computer.

1.3 MANUAL STRUCTURE

A brief overview of the content of the chapters in this manual is shown below.

<u>Chapter 1</u>	Introduction: an introduction to Xycom's 2000 OIL firmware, including functional and environmental specifications
<u>Chapter 2</u>	Installation: instructions for installing OIL firmware onto the 2000 Industrial Workstation
<u>Chapter 3</u>	Basic Concepts: an overview of the OIL menu structure, including backup/restore
<u>Chapter 4</u>	Operator Interface Language (OIL): a descriptive list of the OIL commands, including examples for each command and sample programs
<u>Chapter 5</u>	Programming: examples of how OIL commands can be used together in typical applications
<u>Chapter 6</u>	Remote Commands: a descriptive list of serial remote OIL commands
<u>Chapter 7</u>	Saving/Loading/Verifying Oil Programs: instructions describing how to save, reload and verify stored screens using the IBM PC
<u>Appendix A*</u>	2000-XX: information specific to a particular PLC interface firmware option, including commands and pinouts
<u>Appendix B</u>	OIL Commands: lists OIL commands by section and function and defines syntax terms
<u>Appendix C</u>	Touch Screen: lists codes returned by touch screen zones when OIL firmware is installed and touch screen OIL commands

* Only the appendix that discusses the shipped firmware will be included in the documentation package.

1.4 FIRMWARE SPECIFICATIONS

The following specifications are for the 2000-XX firmware option (OIL). Communication specifications for other firmware options are given in the appendices.

Serial Ports (2)

- RS-232C or RS-485 interface
- Baud rates 300, 600, 1200, 2400, 4800, 9600, 19.2K
- Full or half duplex
- Signals supported:
 - TD, RD, RTS, CTS (both ports)
 - DTR, DCD (primary port only)
- RTS/CTS or XON/XOFF handshaking

Program Memory

- 126 Kbytes (RAM) or 96 Kbytes (ROM) and 32 Kbytes (RAM)
- Battery backed
- 255 program blocks

Time-of-Day Clock/Calendar

- Separate hour, minute, second, year, month, date, and day of week registers
- Accuracy: $\pm 0.001\%$ @ 25°C
- Drift: $+0.001\%$, -0.012%
0°C - 50°C
- Automatic leap year adjustment
- Battery-backed

Operator Interface Language

- Built-in full-screen editor
- Immediate program execution -- no compiling
- Data transfer among:
 - host computer
 - keypad/keyboard/touch screen
 - data registers
 - serial port
- Storage of all message text and attributes in battery-backed memory
- Arithmetic operations (add, subtract, multiply, divide)
- Decision-making (IF) and looping (GO) commands
- Sub-program nesting
- Screen formatting commands
- Logical functions (AND, OR, XOR, NOT)
- Time, date, day of week data registers
- Programmable pause for timed displays
- Program execution controllable from workstation or host computer
- Built-in diagnostic self-tests
- Program back-up and loading
- Automatic start-up of user-specified program upon power-up or reset
- Help screens

Data Registers

- 500 16-bit registers, (11 dedicated)
- Selectable placement in battery-backed memory
- Internal and external access
- Up to 9999 extra data registers selectable

Security

- Selectable lockout of configuration
- Optional password protection

Diagnostics

- Power-up confidence test
- Individually executable tests

Status Lines

- Menu selectable from 0 to 24 status lines

2.1 INTRODUCTION

The standard version of the Xycom 2000 Industrial Workstation consists of a CRT assembly, a power supply board, a controller board, and a 2000-01 firmware chip (for base terminal emulation). Installation of OIL firmware is simply a matter of removing the 2000-01 firmware and replacing it with the OIL firmware. This chapter provides instructions for installing the OIL firmware onto the 2000 Industrial Workstation.

When operating the 2000 Industrial Workstation with OIL firmware, screen programs may be placed in ROM. To store screens in ROM, a 128 Kbyte EPROM chip must be installed on the controller board. Instructions for accessing the controller board and installing an EPROM chip are also included in this chapter.

NOTE

These instructions are for installing the OIL firmware and EPROM chip **only**. Be sure to follow the complete hardware and firmware installation procedures included in Chapter 2 of the 2000 Industrial Workstation Manual.

2.2 BACK PANEL

The back panel of the 2000 offers access to the ports, the power assembly, jumpers, switches, and the firmware socket. Figure 2-1 on the next page shows the back panel of the 2000 Industrial Workstation, with the cover plate removed.

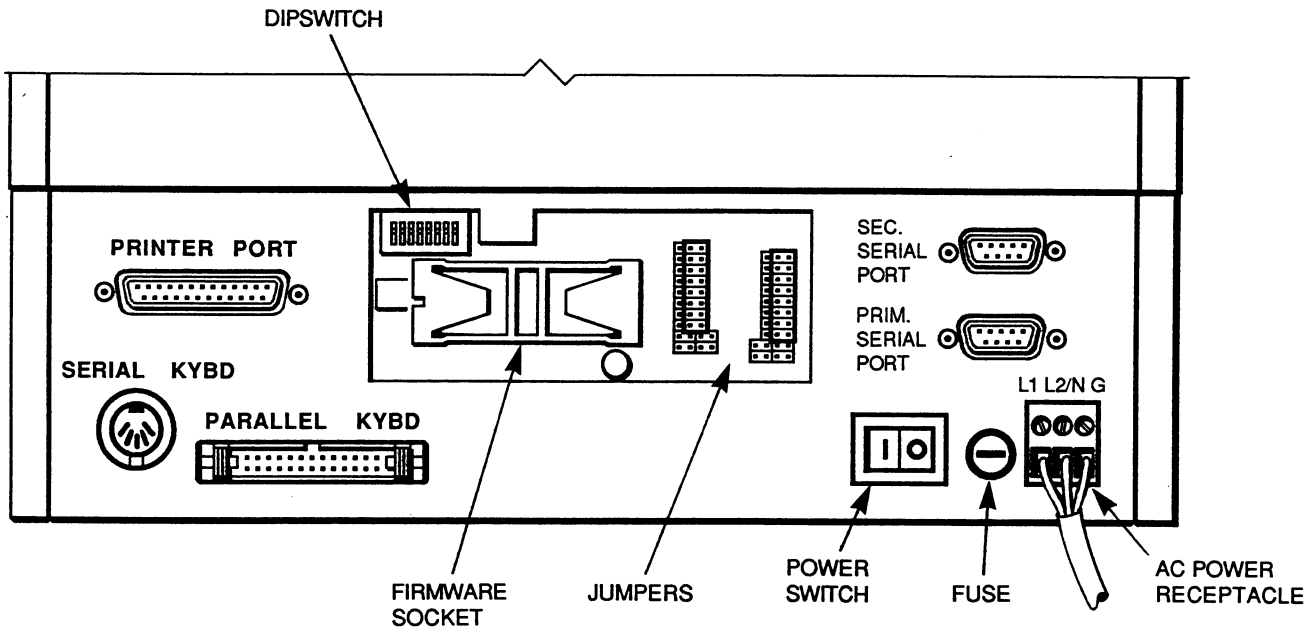


Figure 2-1. 2000 Back Panel

2.3 FIRMWARE INSTALLATION

All 2000 Industrial Workstations are shipped with the 2000-01 (base terminal) firmware installed. To install an OIL firmware chip, follow the instructions below.

WARNING
Never attempt to open any piece of equipment without disconnecting all external power sources.

CAUTION

Back up screen programs prior to changing the firmware. Existing screen programs are cleared upon power up whenever upgraded or different firmware is installed.

1. Turn off power to the terminal and remove the power cord.
2. Remove the cover plate from the back panel of the 2000 by loosening the two screws as shown in Figure 2-2 below. Save all screws and washers.

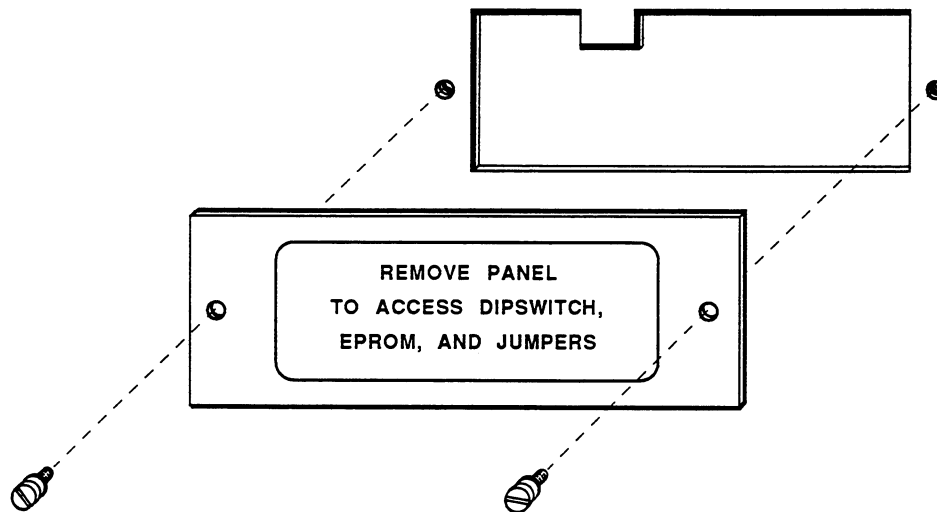


Figure 2-2. Removing the Cover Plate

3. Locate the firmware socket (see Figure 2-1). Remove the 2000-01 firmware chip.
4. Install the replacement firmware chip into the socket, orienting pin one of the chip holder to pin one of the socket, as shown in Figure 2-3, below. Apply gentle, even pressure, and be sure that no pins are bent or out of alignment in the sockets.
5. Replace the cover plate. (If jumpers or switches need to be changed, leave the plate off.)
6. To ensure proper function of the new chip, invoke a .CM command (see Section 3.7.19 for further details).

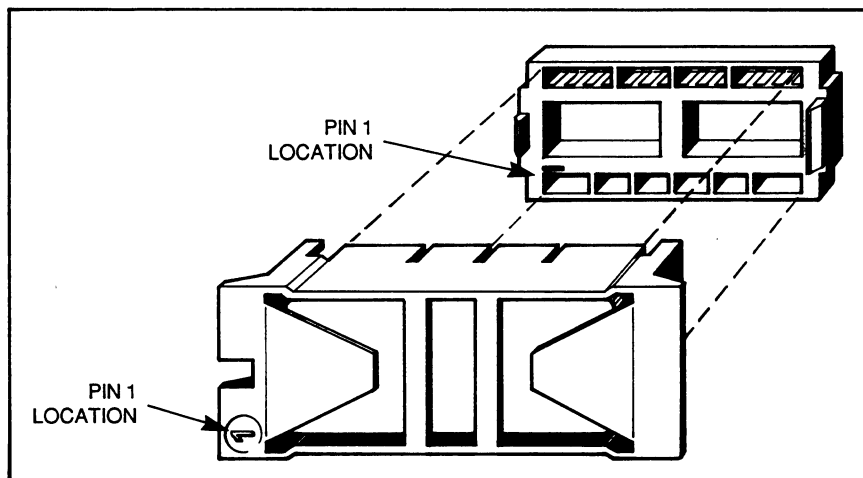


Figure 2-3. Installing the Firmware Chip

2.4 CONTROLLER BOARD

Normally the 2000 Industrial Workstation user should not need to remove the back panel of the unit to access the controller board. Access to the controller board is necessary if installing a 128 Kbyte EPROM chip for storing screen programs in ROM. See the next section for chip installation instructions.

WARNING
Never attempt to open any piece of equipment without disconnecting all external power sources.

Figure 2-4, below, shows the location of the jumpers, connectors, and sockets on the controller board.

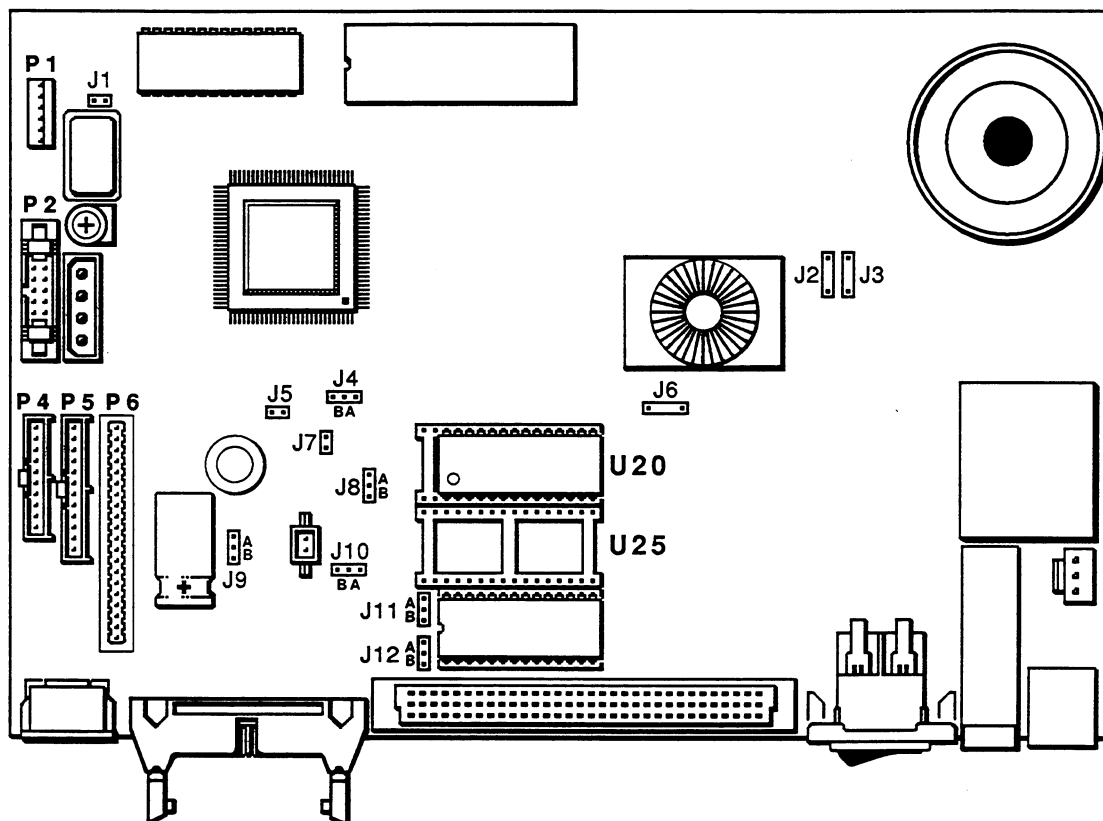


Figure 2-4. 2000 Controller Board

2.5 EPROM INSTALLATION

If you wish to place OIL programs in ROM, an Intel 27C010 128 Kbyte EPROM chip, rated 250 ns or faster, must be installed in socket U25 of the controller board. When installed, this chip supplies 96 Kbytes of usable EPROM for program storage.

Installing the EPROM chip requires access to the controller board. See Figure 2-4 for a diagram of the controller board.

WARNING

Never attempt to open any piece of equipment without disconnecting all external power supplies.

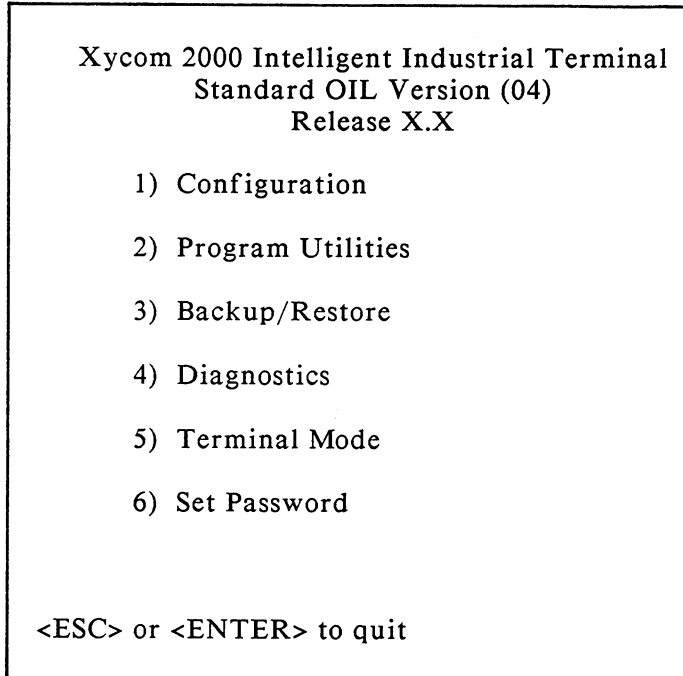
1. Remove and set aside the 10 screws that secure the back/bottom panel to the unit.
2. The controller board is located on the bottom panel. To gain full access to the board, disconnect the video cable from the P2 connector, and the touch screen cable (if touch screen is installed) from the P6 connector. If you have the 2005 version of the workstation, also disconnect the cables from P1, P4, or P5.
3. Install the EPROM just programmed into socket U25 (see Figure 2-4 for location of U25).
4. Position jumper J8 to B for EPROM.
5. Position jumper J10 to B to select 32K RAM.

6. Connect all cables removed in step 2.
7. Secure the bottom/side panel by replacing the screws that were removed in step 1.
8. Remove the cover plate from the back panel of the 2000 by loosening two screws.
9. Position switch 3 to ON (or closed) to select EPROM. The system configuration switches are beneath the cover plate on the back panel of the unit (see Figure 2-1).
10. Replace the cover plate by attaching the two screws that were removed in step 8.

2.6 VERIFYING INSTALLATION

Once the power-up diagnostics are complete, a blank screen will appear. To bring up the Main Menu, press the <F10> key of the keyboard twice.

The Main Menu should now appear, as shown below.



2.7 POWER-UP or RESET

When the terminal is powered-up it goes through a set sequence, which consists of:

- Clearing all command and data queues.
- Performing a diagnostic RAM and ROM test.

When the terminal is reset through a remote command, all command and data queues are cleared, but diagnostics are not run.

3.1 INTRODUCTION

This chapter introduces the basic programming and application concepts of the Xycom 2000 Industrial Workstation with the 2000-04 firmware option (OIL).

3.2 MENU HIERARCHY

The 2000-04 firmware allows the 2000 Workstation to operate in any of three modes: Operating Mode, Set-Up Mode, and Terminal Mode. OIL programs are executed in Operating Mode. When in Operating Mode the workstation can execute program blocks and process remote commands. When in Terminal Mode the 2000 emulates a teletype or dumb terminal. When in Set-Up Mode, the 2000 displays menus on the screen which allow you to change its configuration, type programs into a program block, or edit existing programs.

Upon power-up, the 2000 is automatically in Operating Mode. To put the workstation in Set-Up Mode do the following:

Press the <F10> key twice on the keyboard.

In order for the workstation to receive and execute commands from the control system, it must be returned to Operating Mode. To return to Operating Mode from the Main Menu, simply press the <Esc> or <Enter> key once. If you are in any of the secondary menus you will have to press <Esc> or <Enter> once to return to the Main Menu. Then press the key again to go to Operating Mode.

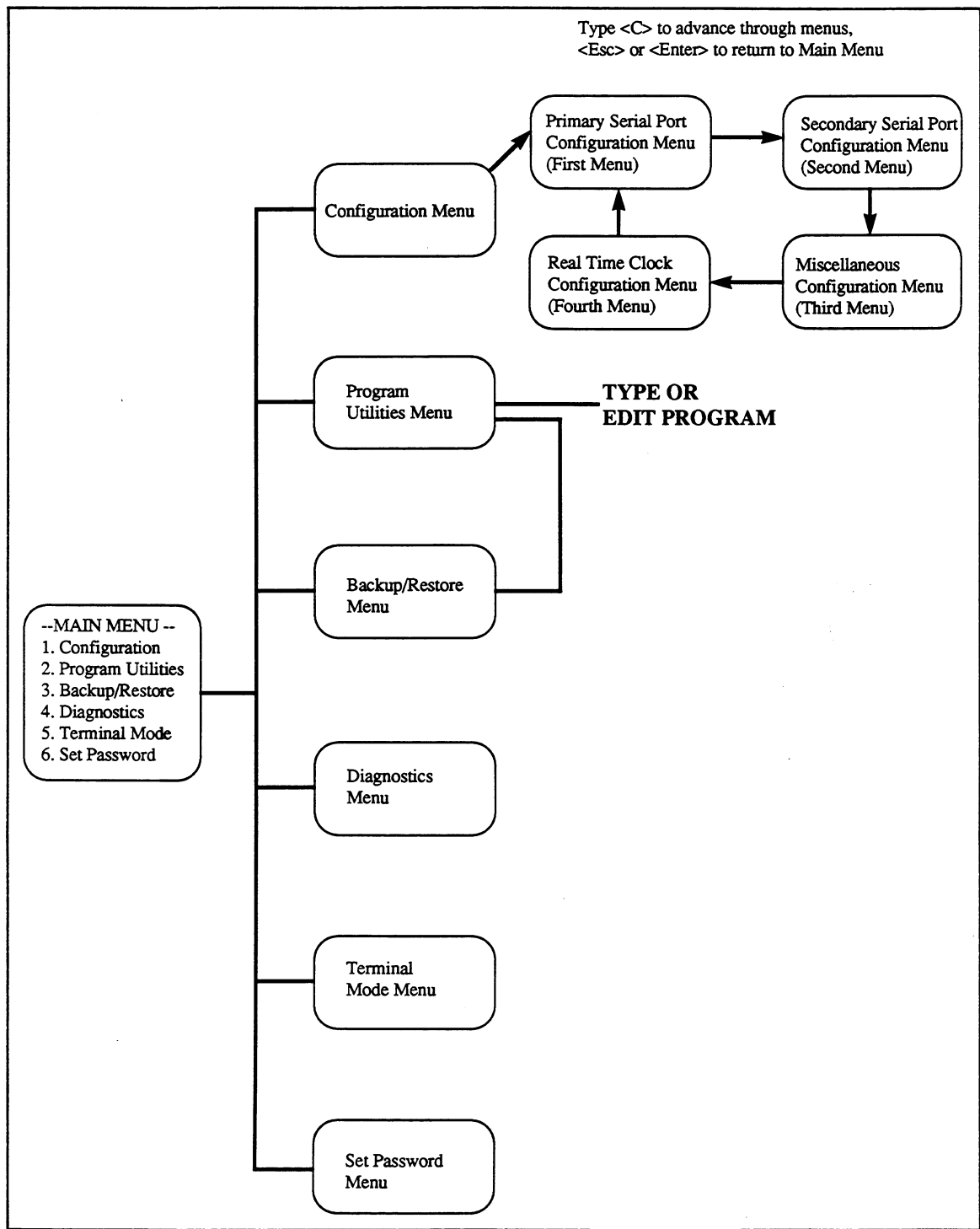


Figure 3-1. Menu Hierarchy

3.3 MAIN MENU

When the 2000 enters Set-Up Mode, the Main Menu will appear with the following options:

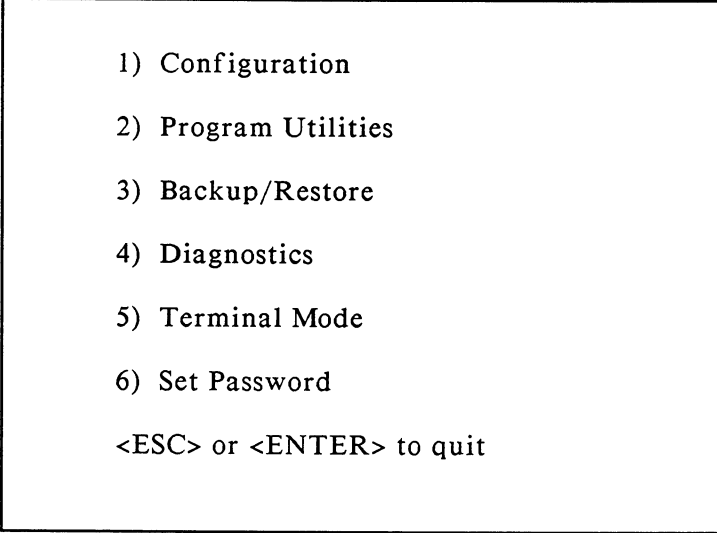
- 
- 1) Configuration
 - 2) Program Utilities
 - 3) Backup/Restore
 - 4) Diagnostics
 - 5) Terminal Mode
 - 6) Set Password
- <ESC> or <ENTER> to quit

Figure 3-2. Main Menu

Selecting any of the options (option number) will bring up secondary menus with additional options. Figure 3-1 on the previous page shows the tree structure of the 2000 Workstation option menus and the relationship of each option when it is in Set-Up Mode.

The following sections provide an overview of each option from the Set-Up Mode Main Menu.

3.4 CONFIGURATION MENUS

There are four configuration menus. The first configuration menu is for PLC interface configuration when firmware 2000-06 or higher is installed. To display the first configuration menu, press "1" while in the Main Menu. To get to the second, third or fourth menu, press "C" while in the menu preceding it (e.g., press "C" in the first menu to get to the second menu). Pressing "C" in the fourth menu invokes the first menu. See Appendix A for the configuration menu specific to your PLC.

3.4.1 Primary Serial Port Configuration Menu (First Configuration Menu)

The first configuration menu, titled Primary Serial Port Configuration, comes up when the Configuration Menu option is entered. If "C" is typed from this menu, the second menu is displayed.

```

-- Primary Serial Port (P) Configuration Menu --

6      Baud - 1=300 2=600 3=1200 4=2400 5=4800 6=9600 7=19.2K
0      Parity - 0=zero 1=one 2=even 3=odd
0      1=Parity Enabled          0=Disabled
1      1=8 Data Bits            0=7 Data Bits
1      1=Full Duplex            0=Half Duplex
0      1=Handshaking Enabled    0=Disabled
0      1=RTS/CTS Handshaking    0=XON/XOFF
0      1=Echo Input              0=Don't Echo Input
0      1=RS-485                  0=RS-232

Use <UP-ARROW>, <DOWN-ARROW>, <LEFT-ARROW>, <RIGHT-ARROW>.
Use values 0 through 9.
"C" for next configuration menu, <ESC> or <ENTER> to quit
```

Figure 3-3. Serial Port Configuration Menu

The first column of the menu lists the current settings of all the configuration options. The available options and their corresponding settings are listed to the right.

To change a configuration option, first move the cursor to the row containing the value which is to be changed. This is done by pressing the up-arrow and down-arrow keys. When the cursor is properly positioned, press a number to select the desired option. After all changes are made, press <Esc> or <Enter>.

Baud Rate. The baud rate of the serial channel should be set to match the serial device connected to the workstation.

Parity. If parity is enabled (see the next item), the type of parity used by the workstation must be set to match that used by the serial device. The types of parity that can be selected are:

Primary Port

- 0 = parity bit always 0
- 1 = parity bit always 1
- 2 = even parity
- 3 = odd parity

Disable or Enable Parity. Parity bit insertion and checking can be enabled or disabled. Parity should be enabled if the connected device uses parity bit insertion and checking; otherwise, parity should be disabled. If parity is enabled, the type of parity must also be selected (see previous item).

Data Bits. Number of data bits per byte (7 or 8) should be set to match the serial device.

Full/Half Duplex. If the connected device is capable of simultaneous two-way communications and is set up for echoing, the workstation should be used in full-duplex mode. If echoing is not used or the host is not capable of simultaneous two-way communications, select half-duplex mode.

NOTE

When the unit is configured for half-duplex, the RTS line takes on a special function.

When a character is transmitted from the workstation, RTS will go high and remain high until one of the terminating characters are transmitted:

- <CR> Carriage Return ASCII 13 (decimal)
- <ETX> End of Text ASCII 3 (decimal)
- <EOT> End of Transmission ASCII 4 (decimal)

When the termination character is transmitted, RTS will go low and remain there until the next non-termination character is transmitted.

Handshaking Enabled. Must be set to 1 to enable either RTS/CTS or XON/XOFF handshaking. If handshaking is enabled with half-duplex selected, the workstation will ignore handshaking and disable it when you leave the configuration menu. In addition, if full-duplex is selected and handshaking disabled, RTS will always be high.

NOTE

Handshaking and half-duplex cannot be enabled simultaneously.

RTS/CTS Handshaking, XON/XOFF.

- 1 = RTS/CTS handshaking
- 0 = XON/XOFF generation

RTS/CTS Handshaking enabled. Handshaking is accomplished through hardware in the following manner:

RTS is an output from the workstation. It will be asserted (High) when it is time for an external device to send data to the workstation. When RTS is inactive (Low), the sending unit should not attempt to send data. This protects the workstation from input buffer overflow.

CTS is an input to the workstation. If this line is asserted (High) the workstation assumes that it is time to transmit data to an external device. When CTS is inactive (Low) the workstation will stop transmitting data to an external device. This keeps the workstation from overflowing the input buffers on an external device.

XON/XOFF Handshaking enabled. Handshaking is accomplished through software in the following manner:

An XON (DC1 ASCII 17 decimal) will be sent by the workstation when it is time for the external device to send data.

If an XON is received by the workstation, it will assume that it is time to send data to an external device.

An XOFF (DC3 ASCII 19 decimal) will be sent by the workstation when the sending device should stop sending data.

If an XOFF is received by the workstation, it will stop sending data until an XON is received.

NOTE

Care should be taken when using XON/OFF handshaking. If the data stream being transmitted contains the XOFF (ASCII 19 or DC3) character, an operator could inadvertently disable communications.

Echo Serial Input. If this item is set to 1, input from the serial input port SI (where SI is the serial input on the controller board) read by a Transfer command will automatically be output to the serial output port (SO). For example, if the Transfer command "TR SI(d),#80" is executed with echoing enabled, the character in SI will be echoed to SO if it is a decimal digit, and its decimal value will be stored in register #80.

RS-485/RS-232. This option selects whether the port is transmitting RS-232 or RS-485 signals. It informs the workstation firmware of serial port jumper settings. For proper serial port jumper settings, consult the 2000 Industrial Workstation Manual.

3.4.2 Secondary Serial Port Configuration Menu (Second Configuration Menu)

The second configuration menu, titled Secondary Serial Port Configuration, is displayed when "C" is typed from the first configuration menu. To advance to the third configuration menu, enter another "C" at the second configuration menu.

This menu is identical to the Primary Serial Port Configuration Menu (Figure 3-3), with two exceptions; the RS-485/RS-232 option is not available, and the only parity options are even and odd.

3.4.3 Miscellaneous Configuration Menu (Third Configuration Menu)

The third configuration menu, Miscellaneous Configuration, is displayed if "C" is typed from the second configuration menu. (From this menu, typing another "C" brings up the fourth configuration menu.)

```

-- Miscellaneous Configuration Menu --

0      1=Block Cursor                0=Underline Cursor
1      1=60 Hz.                      0=50 Hz.
1      1=Add printer linefeeds       0=Do not add linefeeds
1      1=Process all serial data     0=Ignore non-remote commands
0      1=Lock Keypad Menu Entry      0=Unlock Keypad Menu Entry
1      1=Parallel Printer            0=Serial Printer
1      Parallel Input Port Strobe:   0=Low True  1=High True
0      Serial Port for Backup/Restore: 0=Primary   1=Secondary
000    Start-up Program Block Number
0000   Number of Extra Data Registers
1      Number of Status Lines (0 - 24)

Use <UP-ARROW>, <DOWN-ARROW>, <LEFT-ARROW>, <RIGHT-ARROW>.
Use values 0 through 9.
"C" for next configuration menu, <ESC> or <ENTER> to quit
```

Figure 3-4. Miscellaneous Configuration Menu

Block/Underline Cursor. Determines how the cursor will be displayed on the screen.

60 or 50-Hz Refresh. This option should be set to match the frequency of the AC power source: usually 60 Hz in the United States, 50 Hz in Europe.

Add Printer Linefeeds. Determines whether printer linefeeds will occur after carriage returns.

Process all Serial Data/Ignore non-remote commands. When process all serial data is selected, all serial data, including remote commands, will be processed. If ignore non-remote commands is chosen, all non-remote data will be thrown away. This option should only be used when operating the 2000 Workstation remotely.

Lock Keypad Menu Entry. If this field contains the number 1, then a user will not be allowed to enter Setup Mode from the keypad or external sealed keyboard. Therefore, an external full-stroke keyboard would be required to make any configuration or OIL program changes.

Parallel/Serial Printer. If parallel printer is selected, printing will occur through the Centronics parallel port. If serial printer is selected, printing will occur through the port selected for backup/restore in the third configuration menu when base OIL (2000-04) firmware is installed. If 2000-06 or higher firmware is installed, printing will occur through the secondary serial port, as the primary serial port will be connected to the PLC. When serial printing is selected, the parallel port will accept data input.

NOTE

To use the parallel port for data input, printing must occur through the serial port.

Parallel Input Port Strobe. If the parallel port is being used for input, this menu option selects the voltage level of the strobe. When an OIL program is strobed into the port, it will be executed unconditionally. To use the parallel port for data input, serial printing must be selected.

Serial Port for Backup/Restore. This option selects which serial port is used for backup/restore and printing, and only appears on the menu when base OIL (2000-04) firmware is installed. When 2000-06 or higher firmware is installed for use with a direct connect, the secondary port is always used for backup/restore, because the primary serial port is used for PLC connections.

Start-up Program Block Number. If this field contains a number other than 00, that particular program block will be executed whenever the workstation is powered-up or reset. The start-up program block is executed after successful completion of the power-up diagnostics. If this field is 00, no block will be executed at power-up or reset, and, the unit will enter operating mode.

The start-up program block is also executed when switching from Set-up Mode to Operating Mode. This allows you to debug the start-up program block without having to turn the workstation off and on.

NOTE

Be sure the start-up program block is error free. A program started with autostart will terminate on an error.

In order for the workstation to receive and execute commands from the control system it must be returned to Operating Mode.

Number of Extra Data Registers. This selection allows from 0 to 9999 extra data registers to be selected. These registers always reside in CMOS and are in addition to the 500 standard registers. Register allocation has been designed such that if 200 extra registers (501-700) have been defined previously and the menu selection is changed to only 50 extra registers (501-550), the content of the 50 remaining extra registers is not changed. Two bytes of program space are used for each additional register selected.

If there is not enough room in CMOS for the extra registers the number of extra registers is set to 0 and you are notified, except on a power-up.

Number of Status Lines (0-24). If this menu selection is chosen, you will have from 0 to 24 non-scrolling status lines to be maintained at the bottom of the display. The full area of the screen, including the 25th line, is addressable.

Graphics commands (BX, FBX, VBU, VBD, HBR, and HBL) may span the border between the scrolling and non-scrolling areas. The (0,0) coordinate for the PLOT and UNPLOT commands is the lower left corner of the scrolled area.

3.4.4 Real Time Clock Configuration Menu (Fourth Configuration Menu)

The fourth configuration menu, "Real Time Clock Configuration," is displayed if "C" is typed from the third configuration menu. To return to the first configuration menu, type "C" at this menu.

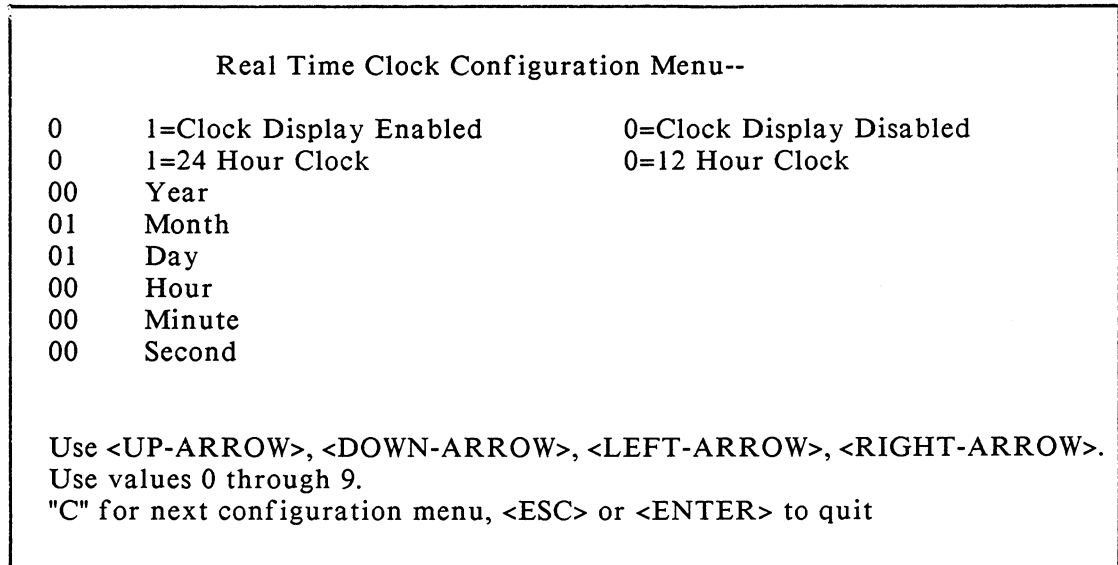


Figure 3-5. Real Time Clock Configuration Menu

Clock Display. When the Clock Display is enabled, a date will appear in the lower left corner of the screen, and the time will appear in the lower right corner of the screen during program execution or Operating Mode.

24 Hour Clock/12 Hour Clock. This option selects whether the clock display will use a 24 hour military clock or a 12 hour clock with AM and PM designations. To set the hour, use 0 - 23, even if a 12 hour clock is selected.

Year, month, day, hour, minute, seconds. Whenever the workstation is powered up after the CMOS RAM has been powered down or the battery removed, the clock/calendar will be initialized to 01/01/00, 00:00:00. Data registers #1 - #7 are continuously updated with the current year, month, day, hour, minute, second, and day-of-week, respectively. (The year, month, and day are automatically adjusted for leap years for any date from 1950 to 2050.) These seven registers cannot be altered by an OIL program running in the workstation -- they can only be read. Their values can be changed only through the Miscellaneous Configuration Menu or through the "Set Time" remote command.

3.5 PROGRAM UTILITIES MENU

This option from the Main Menu is used to type new program blocks into CMOS RAM, modify existing blocks, execute program blocks, and backup/restore program blocks. When Option #2 is selected from the Main Menu the secondary menu in Figure 3-6 below will appear.

The amount of free memory available for OIL programs or registers will appear in the lower right corner of the menu.

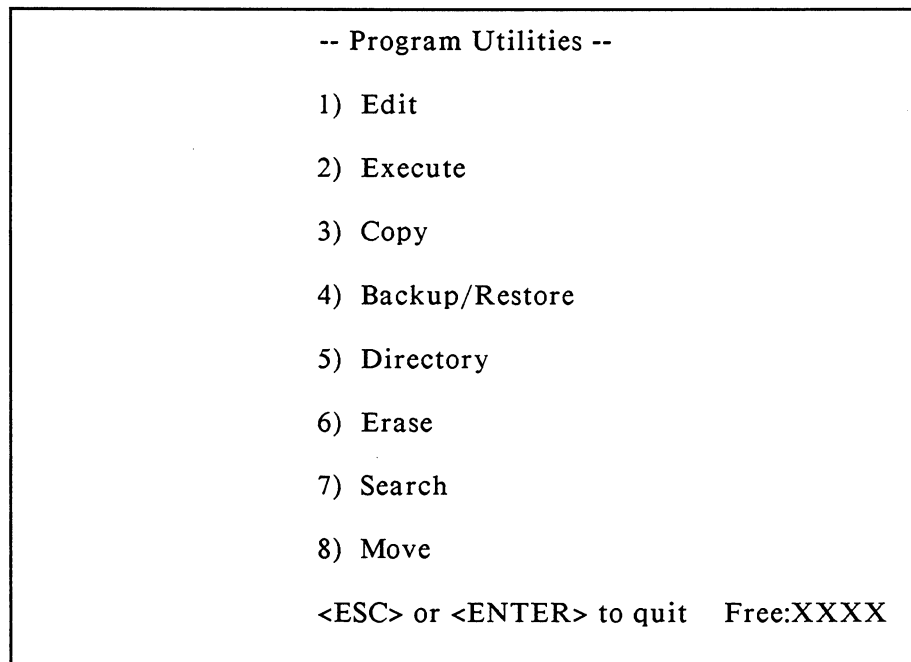


Figure 3-6. Program Utilities Menu

Selecting any of the options (option number) will bring up secondary menus with additional options. Figure 3-1 shows the tree structure of the 2000-OIL Workstation option menus and the relationship of each option when it is in Set-Up Mode.

3.5.1 Using the Editor

NOTE

Entering and editing programs requires the use of an external full-stroke or membrane AT- or XT-style keyboard. See 2000 Industrial Workstation Manual for dipswitch settings.

Selecting option #1 from the Program Utilities Menu puts the workstation in the edit mode. It is in this mode that new programs are typed and existing programs are modified. Before describing the operation of the editor, it is necessary to define a few terms:

Program Block — A program block is defined as the memory space occupied by a single OIL program. There are 255 program blocks in memory, and each of them can store a separate program. On a new firmware chip, all of the program blocks are empty.

Program — A program consists of a sequence of instructions written in Xycom's easy-to-use Operator Interface Language (OIL). Each command in the language will give the workstation specific instructions concerning exactly what characters or symbols to display on the screen, and where on the screen these characters will be displayed. You can also select the size of the characters (single, double, or quad size) and also their attributes (blinking, reverse video, etc.). Programs can also perform other functions, such as reading data or commands from the control system, or reading and writing to the serial ports.

Program Execution — Merely entering a program into a program block does not create a screen display. The program will remain in battery-backed CMOS memory for an indefinite period of time (or until it is replaced by a different program typed or copied into the same block or erased). However, the screen display specified by the program is created only when the program is executed. It will remain on the screen until another program block is executed or until you enter Set-Up Mode to reconfigure the workstation.

Upon entering the Editor, the following prompt appears at the bottom of the screen:

Edit which program? (1-255 or <ESC>):

Enter the number of the program you wish to create or edit, press <ENTER>, and the program block appears on the screen. The status line at the bottom of the edit screen looks like this:

Char <ALT>Line <F6>Run <F5>Quit <F1>Help Pgm #1 Free:28609

To enter a program into a program block, simply type the program. The <Enter> key will move the cursor to the beginning of the next line. The built-in screen editor allows for correction by deleting and inserting characters.

The text of a program can only be altered at the cursor position. Keys which control cursor movement are defined in Table 3-1, below.

Cursor Movement Keys	
→	Moves cursor one position to the right
←	Moves cursor one position to the left
↓	Moves cursor one line down
↑	Moves cursor one line up
<END>	Moves cursor to the end of the line
<HOME>	Moves cursor to the beginning of the line
<PG UP>	Moves cursor to the top of the page
<PG DN>	Moves cursor to the bottom of the page

Table 3-1. Cursor Movement Keys

Control characters are displayed with a two-character representation of these ASCII names. Thus the carriage return at the end of each line is displayed as a "CR" symbol.

Inserting Characters. The editor is always in "insert" mode. This means that any characters typed will be inserted into the program block at the cursor position. Because of this, a user can never inadvertently overwrite anything. A specific action must be made to replace or delete any text.

Deleting Characters. Characters are deleted at the cursor position with the key; characters are deleted to the left of the cursor with <BACKSPACE>.

Replacing Characters. To replace a character, delete the character to be replaced and insert the new character. For example, to replace the "O" in

WORNING_

with an "A", first move the cursor to the "O",

WORNING

then press the key once to delete the "O",

WRNING

and finally type (uppercase) "A" to insert the "A"

WARNING

Inserting a Line. To insert a new line or break a single line into two lines, move the cursor to that point and press the <Enter> key. The character at the cursor position will now be the first character of the next line, and the rest of the lines in the program block (if any) will be moved down one line. A carriage return symbol will appear at the end of each line.

Deleting a Line. To delete an entire line, press the <ALT> and keys simultaneously. This deletes the line where the cursor is currently positioned.

Help Screens. Help Screens are available to you in editing mode. These help screens contain all the commands discussed in detail in Chapter 4 of this manual.

Press <F1> for the first of a series of Help Screens. The status line now reads:

Press <ESC> or <ENTER> to quit, any other key for next screen.

Testing a Program. A newly-entered program can be tested immediately. To execute the program block displayed on the screen, press <F6>. (Before executing the program, the workstation automatically clears the screen, puts the cursor in the upper left hand corner, and resets all display attributes to their default condition.)

The screen will display an error message if any commands have been typed incorrectly or if a value is out of range. The line on which the first error occurred will be near the top of the screen, with the cursor positioned where the program was executing when the error was detected. (The error may not be precisely at the cursor location, but it will be close.) To display the entire program block again, move the cursor upward to scroll the screen.

If <F10> is pressed twice during execution, the workstation will stop the program and return to the editor. If <F10> is pressed twice while the workstation is transferring data (TR command) into or out of the workstation, the workstation returns to the Main Menu.

3.5.2 Program Execution

A program block can be executed in three ways:

- From within the editor, by pressing the <F6> key;
or
- By selecting option #2 - "Execute" from the Program Utilities Menu;
or
- By entering operation mode with an autostart program specified in the Miscellaneous Configuration Menu.

NOTE

If the program has an error and was started with Program Utilities <F6> or "Execute," an error message will be displayed. If the program was started with an autostart program, the program will terminate on the error.

If option #2 is selected from the Program Utilities Menu, you are asked for the number of the program block to be executed:

Execute which program? (1-255 or <ESC>):

Consult the directory (option #5 of the Program Utilities Menu) if the program block number is not known. Press <ESC> to exit the "Execute" menu without executing a program.

Before executing the program, the workstation automatically clears the screen, puts the cursor in the upper left hand corner, and resets all display attributes to their default condition.

To exit from the executing program, press <F10> twice.

3.5.3 Copying Program Blocks

Option #3 - "Copying" from the Program Utility Menu allows you to copy any program to any of the 255 program blocks in memory. This utility provides a means for arranging programs in a specific numerical order or for assigning certain types of programs to certain blocks of numbers for indexing purposes. When option #3 is selected from the Program Utilities Menu the following prompts will be displayed:

Copy from which program? (1-255 or <ESC>):
and then
Copy into which program? (1-255 or <ESC>):
and then
Number of programs to copy? (1-255 or <ESC>):

Example:

Copy from which program? 10
Copy into which program? 50
Number of programs to copy? 2

2 programs will be copied, 10 to 50 and 11 to 51.

NOTE

All existing data in the destination block is lost during a copy.

To prevent the loss of existing programs, the directory should be consulted prior to making a copy.

3.5.4 Backup/Restore Programs

Option #4 from the Program Utilities Menu, "Backup/Restore," is provided as a means of saving and loading screen programs via the serial port and a computer system. Section 7.1 of this manual deals with the procedure for saving and restoring programs on a PC compatible computer. This option can also be selected via item #3 on the Main Menu.

When this option is selected the following choices are displayed:

- 1) Backup Programs
 - 2) Restore Programs
 - 3) Verify Programs
 - 4) Print Programs
- <ESC> or <ENTER> to quit

3.5.4.1 Backup Programs

Option #1 - "Backup Programs" allows you to save program blocks from the screen memory to a connected back-up unit. When option #1 is selected the following prompts will be displayed:

Backup <P>rograms, <R>egisters, <A>ll or <ESC>?

If <P>rograms are selected:

Backup which programs? (1-255 or <ESC>):
and then
Save it as which program? (1-255 or <ESC>):
and then
Number of programs to backup? (1-255 or <ESC>):
The programs selected will be transmitted out the backup/restore port,
and then
Program Backup Done.

If <R>egisters are selected:

Transmitting.....
All registers will be transmitted out the backup/restore port, and then
Program Backup Done.

If <A>ll is selected:

Transmitting.....

All programs and registers will be transmitted out the backup/restore port, and then
Program Backup Done.

Refer to Section 6.4 for more information on backing-up programs.

3.5.4.2 Restore Programs

The restore option loads previously saved programs into screen memory.

Selecting (2) Restore Programs displays the following prompts:

Receiving Programs <ESC> to Quit Pgm XXX
Either "Regs" or the number of the program being received is displayed in the lower right-hand corner of the screen, and then
Program Receiving Done.

Refer to Section 6.4 for more information on restoring programs.

3.5.4.3 Verify Programs

The verify option will check data just backed-up against the original program in memory (in order to verify that the transfer was successful).

Selecting (3) Verify Programs displays the following prompts:

Verifying Programs <ESC> to Quit Pgm XXX
The number of the program being verified is displayed in the lower right-hand corner of the screen, and then
Program Verification Passed.
or
Program Verification Failed.

3.5.4.4 Print Programs

The print programs option will go through the same sequence of prompts for program numbers as a backup function. When program (register) selection is complete, the programs/registers selected will be output to the serial port or Centronics port formatted for a printer. Only non-empty programs will be printed.

Selecting (4) Print Programs displays the following prompts:

Print <P>rograms, <R>egisters, <A>ll or <ESC>?

If <P>rograms are selected:

Print which programs? (1-255 or <ESC>):
and then

Number of programs to print (1-255 or <ESC>):

The programs selected will be transmitted out the printer port, and then
Print Programs Done.

If <R>egisters are selected:

Transmitting.....

All registers will be transmitted out the printer port, and then
Print Programs Done.

If <A>ll is selected:

Transmitting.....

All programs and registers will be transmitted out the printer port, and
then
Print Programs Done.

3.5.5 Using the Directory

Option #5 "Directory" from the Program Utilities Menu allows you to see the first line of every program block in screen memory. Thus, it is recommended that the first line of every program be a program name or title (any text preceded by a semicolon will be treated by OIL as a comment or remark and will be ignored). The directory option displays the program blocks in groups of 20. The next group of 20 blocks can be accessed by pressing any key other than <Esc>. Exit the directory by pressing <Esc> key.

3.5.6 Erase

Option #6 "Erase" from the Program Utility Menu allows you to erase any program or range of programs of the 255 program blocks in memory. This utility provides a means to delete unwanted programs. When this option is selected from the Program Utilities Menu, the following prompt will be displayed:

```
Erase starting at which program? (1-255 or <ESC>):
                        and then
Number of programs to erase? (1-255 or <ESC>):
                        and then
OK to erase programs x to y (Y or N):
```

Example:

```
Erase starting at which program? 10
Number of programs to erase? 2
OK to erase programs 10 to 11 (Y or N): Y
```

2 programs will be erased, 10 and 11.

3.5.7 Search

Option #7 "Search" from the Program Utility Menu allows you to search for a string of characters. You may specify a string up to twenty characters in length. All program blocks, starting at block #1, are searched. If a match is found, the screen displays the string, the program block number, and the line number in which the string was found. You may either quit or continue searching.

3.5.8 Move

Selecting Option #8, "Move," from the Program Utility Menu allows you to bring up the move programs into the EPROM menu. This option will transfer the OIL programs in RAM out the secondary port using Extended Intel Hex format to an EPROM programmer. The OIL programs can then be programmed into a 128Kx8 EPROM. This EPROM can then be installed into the 2000 Workstation. Refer to Chapter 2 of the 2000 Industrial Workstation Manual for dipswitch setting and jumper configurations.

The EPROM programmer should be set as follows:

```
9600 baud
No parity
8 data bits
1 stop bit
```


When Option #8 is selected from the Program Utility Menu, the following prompts will be displayed:

--- Move Programs into EPROM ---

Enter an Autostart Program Number if you would like to ROM your Autostart Program <ESC> to skip this option.

(1-255 or <ESC>):

Verify cable connection between Secondary Port and EPROM Programmer.
You will be downloading Extended Intel Hex Records for one (128Kx8) EPROM.

Press <Space Bar> when Programmer is ready to receive data.....
<ESC> to exit.

Downloading data for EPROM.....Please Wait

NOTE

To store programs in ROM, system configuration switch position #3 needs to be set to ON (on = closed, off = open) (see 2000 Industrial Workstation Manual for switch settings).

3.6 BACKUP/RESTORE MENU

This option is provided as a means for saving and loading program blocks via the serial port. For information on using this option refer to Sections 3.5.4 and 7.1.

3.7 DIAGNOSTICS MENU

Selecting option #4 from the Main Menu brings up a Diagnostics Menu for general purpose testing of RAM, ROM, ports, character attributes, the CRT, and the time-of-day clock/calendar. Figure 3-7 shows the options available from the Diagnostics Menu. When a selected test is completed, status information about the completed test and the Diagnostics Menu will be displayed.

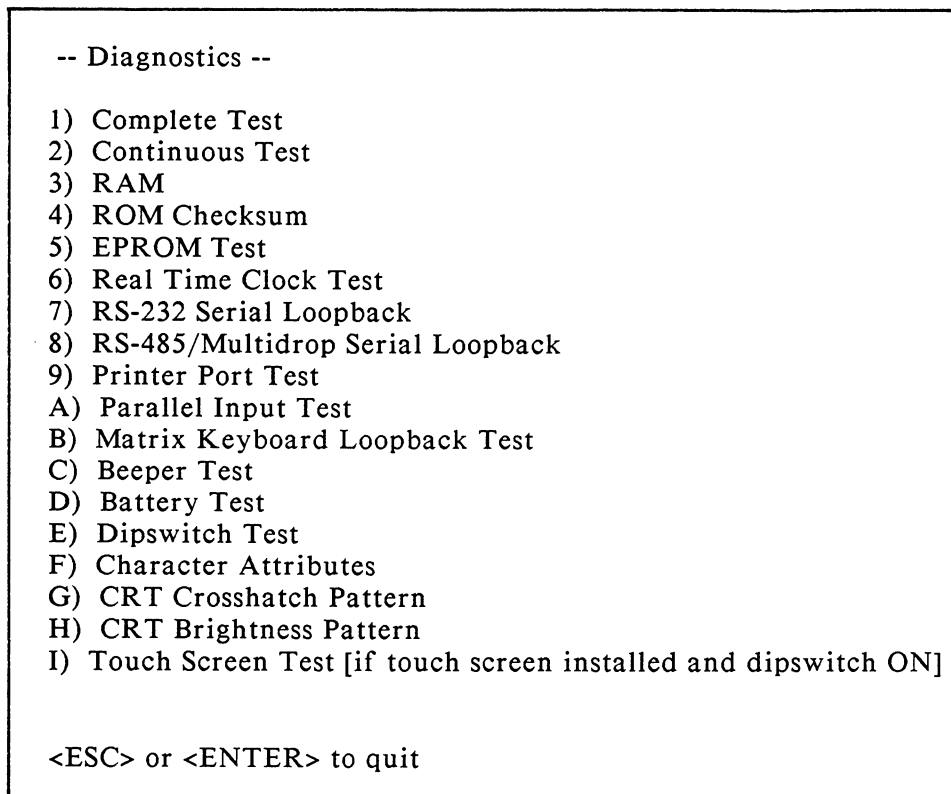


Figure 3-7. Diagnostics Menu

NOTE

Depending on the firmware option installed, some diagnostics selections may not be available.

3.7.1 Complete Test

The complete test exercises all the tests listed on the Diagnostics Menu, one at a time. You are prompted to install serial loopback connector(s) before beginning the tests. After each test is run, you are prompted to hit any key to continue. See the following sections for descriptions of the specific diagnostic tests.

3.7.2 Continuous Test

CAUTION

Turning off power while the continuous test is in progress could destroy all clock data and data in CMOS RAM.

In this mode, the workstation continuously cycles through the RAM, serial port, parallel input port, ROM, and real time clock tests. If an error is found, the workstation stops testing and displays an appropriate error message along with the prompt:

Press any key to continue.

If a key is then pressed, testing will continue. Press any key twice to discontinue testing.

After all testing is complete, the program will continue indefinitely. To exit the continuous test mode, press any key.

3.7.3 RAM Test

If the RAM test is selected, the workstation will check the CPU RAM, the CMOS RAM, the display RAM, the attribute RAM, and the character generator RAM. After checking the CPU RAM the workstation will display one of the following messages:

CPU RAM OK
or
CPU RAM failure

<p style="text-align: center;">CAUTION Turning off power while the CMOS RAM test is in progress will destroy all data in CMOS RAM.</p>

The next test checks the CMOS RAM. After testing, the workstation displays one of these messages:

CMOS RAM OK
CMOS RAM failure Page #nn

The workstation will then test the display RAM, during which a pattern will be flashed on the video display followed by one of these messages:

Display RAM OK
or
Display RAM failure Page #nn

The workstation will then test the attribute RAM, again flashing a pattern on the video display followed by one of these messages:

Attribute RAM OK
or
Attribute RAM failure Page #nn

The workstation will then test the character generator RAM. A pattern will flash on the video display, followed by one of these messages:

Character Generator RAM OK
or
Character Generator RAM failure

3.7.4 ROM Checksum

This test "ROM checksum is: nnnn Should be: nnnn" on the status line. The two checksums listed (nnnn) should match.

3.7.5 EEPROM Test

When this test is selected, a non-destructive write/read test of all registers in the EEPROM is performed, and the message:

EEPROM test in process

appears on the screen. When the test is complete, one of the following messages will appear:

EEPROM device OK
or
EEPROM failure Register #nn

3.7.6 Real Time Clock Test

CAUTION

Turning off power while the real time clock test is in progress will destroy all clock data.

A non-destructive storage test, counter roll-over test, and control signal test is performed on the clock calendar. If no errors are found, the workstation will display the message "Real Time Clock OK." If an error is found, the subtest which failed and error information will be displayed instead.

3.7.7 RS-232 Serial Loop Back

This test checks the primary and secondary serial ports for the RS-232 configuration. Before these ports are tested, a serial loopback connector must be installed, and jumpers must be set to configure the ports as RS-232 (see 2000 Industrial Workstation Manual for jumper settings). The test plugs should be constructed of a DE-9S connector and jumper wires. The configuration of the test plugs is shown in Figure 3.8 on the next page.

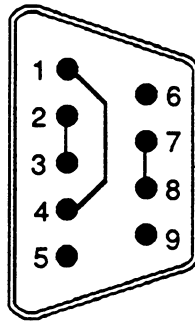


Figure 3-8. Serial Port Test Plugs, RS-232

If the serial ports are operating correctly, the workstation will display the message:

Prim. port: OK. Sec. port: OK.

If an error is found, the workstation will display one of the following messages:

Time out err.
Data err.
CTS-RTS err.
DTR-DSR err.

3.7.8 RS-485/Multidrop Serial Loop Back

This test checks the primary and secondary serial ports for the RS-485 configuration. Before these ports are tested, serial loopback connectors must be installed, and jumpers must be set to configure the ports as RS-485 (see 2000 Industrial Workstation Manual for jumper settings). The test plugs should be constructed of a DE-9S connector and jumper wires. The configuration of the test plugs is shown in Figure 3-9 on the next page.

If the serial ports are operating correctly, the workstation will display the message:

Prim. port: OK. Sec. port: OK.

If an error is found, the workstation will display one of the following messages:

<u>If RS-485:</u>	<u>If Multidrop:</u>
Time out err (RS-485).	Data err (MULTI).
Data err (RS-485).	Time out err (MULTI).

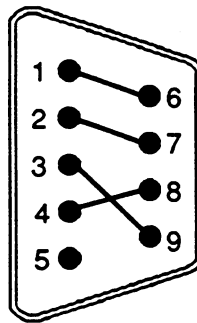


Figure 3-9. Serial Port Test Plugs, RS-485

3.7.9 Printer Port Test

To run this test, a printer cable must be attached to the parallel port at the rear of the unit, and the printer must be on-line.

The test pattern that is sent to the printer should be the same as the message that appears on-screen. If the printer port is operating correctly, the workstation will display the message:

Printer Port test passed.

If an error is found, the workstation will display the message:

Printer Error: <ESC> to Abort, Any other key to continue

3.7.10 Parallel Input Port Test

This test checks the parallel port for input capability. If the port passes the diagnostics, the workstation will display the message:

Parallel Input Port test passed.

If an error is found, the workstation will display the message:

Parallel Input Port test failed.

3.7.11 Matrix Keyboard Loopback Test

This test checks the matrix keyboard port. Before the port is tested, a column driver loopback connector must be installed.

If the matrix keyboard port is operating correctly, the workstation will display the message:

Matrix Test Passed. - Remove Loopback.

If an error is found, the workstation will display the following message:

Matrix Test Failed. - Remove Loopback.

3.7.12 Beeper Test

This test checks the beeper. If the beeper is functioning, the workstation will emit a continuous beep, and the following message will appear on the screen:

----- Beeper should be sounding. Press any key to exit test. -----

3.7.13 Battery Test

This test checks current condition of the battery. If the battery is functioning properly, the screen will display the message:

Battery Test passed.

If an error is found, the screen will display the message:

Battery Test Failed.

3.7.14 Dipswitch Test

To test the dipswitches, bring up the dipswitch test screen and toggle the switches. Closed settings are represented by "C"; open settings are represented by "O." When all the switches are closed (on), the display should look like this:

```
Switch:      8 7 6 5 4 3 2 1
             -----
             C C C C C C C
             -----
```

When all the switches are open (off), the display should look like this:

```
Switch:      8 7 6 5 4 3 2 1
             -----
             O O O O O O O
             -----
```

NOTE

Return all switches to their original settings after testing.

3.7.15 Character Attributes

When this test is run a status line displays reverse video, high-intensity, underlining, blinking, and double-wide characters.

3.7.16 CRT Crosshatch Pattern

When this test is run, a crosshatch pattern and the message "48.5 Hz Grid" will display on the screen. Pressing any key will display another crosshatch pattern and the message "61.5 Hz Grid." This test verifies the capability of the CRT monitor to hold vertical sync in 50 and 60 Hz vertical scan rate mode.

3.7.17 CRT Brightness Pattern

When this test is run, reverse video highlighted spaces are displayed, producing a standard "white level" screen.

3.7.18 Touch Screen Test

When this test is run on workstations with touch screen, an 8 x 10 grid appears on the screen, dividing the screen into the 80 touch screen zones. When a touch screen zone is pressed, a block character should be displayed in that zone. When a zone is released, an R should be displayed in that zone.

The following message is displayed on the screen:

Touch each zone and "R"elease each zone -- Press any key to exit test

3.7.19 .CM Command

A hidden command, .CM, is available from the Diagnostic Menu for clearing the contents of the battery-backed CMOS RAM. This command clears the RAM, performs a resize of the RAM space, and initializes the real time clock to its default setting. To invoke this routine, simply type <.CM> at the Diagnostic Menu.

CAUTION

The .CM command will destroy any programs residing in CMOS RAM.

If a keyboard is not available to perform the .CM routine, it can be invoked by disabling the battery. The battery is enabled and disabled via Jumper J9 of the workstation controller board (see the 2000 Industrial Workstation Manual for location of J9 and instructions on accessing the controller board). Simply power down the workstation, access the controller board, set Jumper J9 to B for 5 seconds, and return Jumper J9 to A. When the workstation is powered up, the .CM routine will be invoked.

3.8 TERMINAL MODE MENU

When this option is selected the workstation will act like a teletype, or dumb terminal. Characters typed on the keyboard or keypad are sent out the serial port and echoed to the screen (if the workstation is configured for half-duplex). Characters received at the serial port are echoed to the screen. All characters except four are interpreted literally. The exceptions are:

- 1) A carriage return will cause a new line to be performed.
- 2) A line feed will be ignored.
- 3) A backspace will cause a backspace to be performed.
- 4) A bell will cause the beeper to sound momentarily.

The terminal mode can be exited by pressing the "F10" key twice.

3.9 SET PASSWORD

A password may be used to tamper-proof the workstation's configuration.

To create, disable, or change a password, select #6, Set Password, from the Main Menu. The following screen will be displayed:

```
Enter new password (3 characters)
<ESC> to disable password
<ENTER> only for no change
```

Once a password has been established, it is required to access the Main Menu after power-up. After striking F10 twice, the following prompt will be displayed:

```
Enter password (3 characters), <RET> or <ENTER> to quit:
```

Type the correct 3-character sequence and the Main Menu of the Set-up Mode will be displayed. When an incorrect password is entered the workstation remains in Operating Mode.

If you forget the password, the remote command Send Password will send the password out the serial port to the host computer (see Chapter 6).

The workstation configuration may also be tamper-proofed via keypad menu lockout. See Section 3.4.3.

Chapter 4 - OPERATOR INTERFACE LANGUAGE (OIL)

4.1 INTRODUCTION

This chapter discusses the Operator Interface Language (OIL) commands. Each PLC has its own specific commands for reading (GET) and writing (PUT) data. These are discussed in the appendixes for each firmware upgrade. The following tables list the OIL commands common to all of the modules, their parameters, and the sections where they may be found.

Table 4-1. OIL Commands by Section

COMMAND	SYNTAX*	SECTION
Screen Text	"{character}"	4.6.2
Register Value Display	reg	4.6.3
Position	@[xcoord],[ycoord]	4.6.4
Label	%label	4.6.5
Add to Register	ADD num,reg	4.6.6
AND	AND reg,num	4.6.7
Beep	BP	4.6.8
Draw Box	BX bval,[xstart],[ystart],[xend],[yend]	4.6.9
Clear Buffer	CB device	4.6.10
Clear Line	CL	4.6.11
Clear Bit	CLRB reg,bit	4.6.12
Clear Screen	CS	4.6.13
Clear Window	CW	4.6.14
Down	D [num]	4.6.15
Decrement Register	DEC reg	4.6.16
Define Zone**	DFZ upper left zone,lower right zone, num	4.6.17
Divide Register	DIV reg,num	4.6.18
Display String	DS reg	4.6.19
Execute Program and Reset Subroutine Counter	ER num	4.6.20
Execute Sub-Program Block	ES num	4.6.21
Execute Program Block	EX num	4.6.22
Exit from a Sub-Program	EXIT	4.6.23
Draw Filled Box	FBX bval,[xstart],[ystart],[xend],[yend]	4.6.24
Go	GO label	4.6.25
Horizontal Bar Left	HBL [xstart],[ystart],lngth,maxlngth	4.6.26
Horizontal Bar Right	HBR [xstart],[ystart],lngth,maxlngth	4.6.27
Draw Horizontal Line	HL lval,[xstart],[ystart],hlngth	4.6.28
If	IF cond,[:command]	4.6.29
If Bit	IFB reg,bit,[:command]	4.6.30
If String	IFS cond,[:command]	4.6.31
Increment Register	INC reg	4.6.32
Exit ONKEY Subroutine Program	KEXIT	4.6.33

Table 4-1. OIL Commands by Section (cont.)

COMMAND	SYNTAX*	SECTION
Keypad Status	KS reg	4.6.34
Left	L [,num]	4.6.35
Remainder (Modulus)	MOD reg,num	4.6.36
Multiply Register	MUL reg,num	4.6.37
New Line	NL [,num]	4.6.38
NOT	NOT reg	4.6.39
ONKEY	ONKEY num	4.6.40
ONTOUCH**	ONTOUCH num	4.6.41
OR	OR reg,num	4.6.42
Plot	PLOT xpoint,ypoint	4.6.43
Pause	PS num	4.6.44
Right	R [,num]	4.6.45
Reset Attributes	RE {attr}	4.6.46
Restore Attributes	RSTA reg	4.6.47
Restore Cursor Position	RSTC reg	4.6.48
Save Attributes	SAVA reg	4.6.49
Save Cursor Position	SAVC reg	4.6.50
Set Attributes	SE {attr}	4.6.51
Set Bit	SETB reg,bit	4.6.52
Stop	STOP	4.6.53
Store String	SS label,reg	4.6.54
Subtract from Register	SUB num,reg	4.6.55
Transfer System Characteristics	SYS reg	4.6.56
Exit ONTOUCH Subroutine Program**	TEXIT	4.6.57
Transfer Data	TR source,dest	4.6.58
Transfer Text	TS source,dest	4.6.59
Transfer Touch Screen Data**	TT dest	4.6.60
Up	U [,num]	4.6.61
Unplot	UNPLOT xpoint,ypoint	4.6.62
Vertical Bar Up	VB VBU [xstart],[ystart],hght, maxhght	4.6.63
Vertical Bar Down	VBD [xstart],[ystart],hght,maxhght	4.6.64
Draw Vertical Line	VL lval,[xstart],[ystart],vlength	4.6.65
XOR	XOR reg,num	4.6.66

*See the key on page 4-6 for syntax definitions.

**Touch screen option only.

4.2 COMMAND LIST

Table 4-2 lists all the available OIL commands by function. Section numbers are listed in the far right column.

Table 4-2. OIL Commands by Function

COMMAND	SYNTAX*	SECTION
Screen Text	"{character}"	4.6.2
Register Value Display	reg	4.6.3
select character size and attributes (i.e., blinking or reverse video)		
Set Attributes	SE {attr}	4.6.51
Reset Attributes	RE {attr}	4.6.46
Save Attributes	SAVA reg	4.6.49
Restore Attributes	RSTA reg	4.6.47
move the cursor		
Restore Cursor Position	RSTC reg	4.6.48
Save Cursor Position	SAVC reg	4.6.50
Up	U [,num]	4.6.61
Down	D [,num]	4.6.15
Left	L [,num]	4.6.35
Right	R [,num]	4.6.45
Position	@[xcoord],[ycoord]	4.6.4
New Line	NL [,num]	4.6.38
draw figures and lines		
Draw Box	BX bval,[xstart],[ystart],[xend],[yend]	4.6.9
Draw Filled Box	FBX bval,[xstart],[ystart],[xend],[yend]	4.6.24
Draw Vertical Line	VL lval,[xstart],[ystart],vlength	4.6.65
Draw Horizontal Line	HL lval,[xstart],[ystart],hlength	4.6.28
Vertical Bar Up	VB VBU [xstart],[ystart],hght, maxhght	4.6.63
Vertical Bar Down	VBD [xstart],[ystart],hght,maxhght	4.6.64
Horizontal Bar Right	HBR [xstart],[ystart],length,maxlength	4.6.27
Horizontal Bar Left	HBL [xstart],[ystart],length,maxlength	4.6.26
Plot	PLOT xpoint,ypoint	4.6.43
Unplot	UNPLOT xpoint,ypoint	4.6.62

Table 4-2. OIL Commands by Function (cont.)

COMMAND	SYNTAX*	SECTION
display data or move between the keyboard, data registers, and host computer		
Transfer Data	TR source,dest	4.6.58
Transfer Text	TS source,dest	4.6.59
Transfer Touch Screen Data**	TT dest	4.6.60
Store String	SS label,reg	4.6.54
Display String	DS reg	4.6.19
Transfer System Characteristics	SYS reg	4.6.56
conditional and looping commands		
If	IF cond,[:command]	4.6.29
If Bit	IFB reg,bit:command	4.6.30
If String	IFS cond,[:command]	4.6.31
Go	GO label	4.6.25
Label	%label	4.6.5
set and clear bits of a data register		
Set Bit	SETB reg,bit	4.6.52
Clear Bit	CLRB reg,bit	4.6.12
arithmetic operations		
Increment Register	INC reg	4.6.32
Decrement Register	DEC reg	4.6.16
Add to Register	ADD num,reg	4.6.6
Subtract from Register	SUB num,reg	4.6.55
Multiply Register	MUL reg,num	4.6.37
Divide Register	DIV reg,num	4.6.18
Remainder (Modulus)	MOD reg,num	4.6.36
logical operations		
AND	AND reg,num	4.6.7
XOR	XOR reg,num	4.6.66
OR	OR reg,num	4.6.42
NOT	NOT reg	4.6.39

Table 4-2. OIL Commands by Function (cont.)

COMMAND	SYNTAX*	SECTION
program nesting		
Execute Program and Reset Subroutine Counter	ER num	4.6.20
Execute Program Block	EX num	4.6.22
Execute Sub-Program Block	ES num	4.6.21
Exit	EXIT	4.6.23
Stop	STOP	4.6.53
touch screen option		
ONTOUCH**	ONTOUCH num	4.6.41
Exit ONTOUCH Subroutine Program**	TEXIT	4.6.57
miscellaneous		
Beep	BP	4.6.8
Clear Buffer	CB device	4.6.10
Clear Line	CL	4.6.11
Clear Screen	CS	4.6.13
Clear Window	CW	4.6.14
Define Zone**	DFZ upper left zone, lower right zone, num	4.6.17
Keypad Status	KS reg	4.6.34
Pause	PS num	4.6.44
ONKEY	ONKEY num	4.6.40
Exit ONKEY Subroutine Program	ONKEY	4.6.33

*See the key on page 4-6 for syntax definitions.

**Touch screen option only.

KEY

<u>SYMBOL</u>	<u>MEANING</u>
[]	Optional argument
{}	One or more occurrences of this argument can be included
	Choice of either argument
char	Any single character (e.g., N, m, \$)
reg	#n OR \$n, where n=1 through maximum number of data registers possible (10,499) #n = displays register value in decimal \$n = indirect register
bit	A specific bit within a register (0-15)
num	A decimal number (0-65535), a hexadecimal number (&0-&FFFF), or a data register
text	One or two characters enclosed in double quotes (e.g., "A" or "ra")
xcoord xstart xend	Column coordinate (0 - 79)
ycoord ystart yend	Row coordinate (0 - 23)
vlength	Vertical length of graphic (0 - 24)
hlength	Horizontal length of graphic (0 - 79)
hght maxhght	Height of graphic (0 - 255)
lngth maxlngth	Length of graphic (0 - 255)
xpoint	Column coordinate (0 - 159)

KEY (cont.)

<u>SYMBOL</u>	<u>MEANING</u>
ypoint	Row coordinate (0 - 79)
bval	1 = thick line, 2 = thin line, any other character = figure composed of that character any other number = figure composed of characters corresponding to the number
lval	1 = thick line, 2 = thin line, 3-6 (see VL and HL commands) any other number = figure composed of characters corresponding to the number
source	Source for data in a transfer. Any of the following: KB (keyboard and keypad) SI (serial input) (PI for 2000-04) #n (data register) \$n (numerical argument) (text argument)
dest	Destination for data in a transfer. Any of the following: SC (screen output) SO (serial output) NO (dummy output) (PO for 2000-04) \$n (data register)
attr	Character attribute. Any of the following: QS (quad-size) DS (double size) RV (reverse video) HI (highlight) UN (underline) BL (blinking) DW (double-wide) DH (double-high) SC (screen output) PR (printer output) SS (screen scrolling) CU (cursor on/off) G1 (process graphics) G2 (thin-line and block graphics) G3 (process control graphic connectors) G4 (mini Process graphics)

KEY (cont.)

<u>SYMBOL</u>	<u>MEANING</u>
cond	————— A logical function inserted between two nums. It can be: <, >, <>, =, =>, >=, <=, or =<
label	————— A sequence of alphanumeric characters
device	————— A device used to handle data. It can be: KB (keyboard or keypad) PI, SI (primary and secondary serial input) PO, SO (primary and secondary serial output) TS (touch screen)

4.3 OVERVIEW OF THE OPERATOR INTERFACE LANGUAGE (OIL)

A program consists of sequences of displayable characters and OIL commands. The simplest program consists of a sequence of characters enclosed in double quotes:

```
"Warning"
```

If the above sequence of characters is typed into a program block and then the program block is executed, the following characters will be displayed in the top left corner of the screen:

```
Warning
```

In addition to sequences of characters, a program block can contain any number of OIL commands. OIL is an easy-to-use programming language, especially designed to simplify the creation of screen displays and operator interaction for industrial applications. For example, the following complete program block will enclose the letters "Warning" in a box and center it on the screen (try entering it in a program block and executing the block):

```
BX,1,30,5,50,15      ;draw a box in the center of the screen  
@35,10              ;move the cursor inside the box  
"Warning"           ;display a message
```

The first two lines of the above program are the Draw Box and the Position Cursor commands. They simplify the drawing of a box and cursor movement. A box does not have to be created character by character, moving the cursor to a new position after each character comprising the box is displayed. Instead, the entire box can be created by one command.

To further enhance the screen, the message "Warning" can be printed in double-size characters and blinking the video. This is done by changing the attributes of a character. A character's attributes determine its size (regular, double-high, double-wide, double-size, quad-size) and how it is displayed (normal, blinking, underline, etc.). Attributes also control other features of the display; for example, screen scrolling and the selection of special process graphics characters. For a complete list of all the possible attributes, refer to the Set Attributes command.

By adding one command to the above program, "Warning" will be displayed in double-size, blinking characters:

```
BX,1,30,5,50,15      ;draw a box in the center of the screen  
@35,10              ;move the cursor inside the box  
SE,DS,BL            ;set attributes to "double-size" and "blinking"  
"Warning"           ;display a message
```

Note that after the Set Attributes command (SE) has been executed, all characters subsequently displayed on the screen will be double-sized and blinking. To return to normal characters, the following command would have to be added to the end of the program:

```
RE,DS,BL          ;reset attributes to normal, i.e., negate double-size
                  ;and blinking
```

The above examples include commands which draw figures (Draw Box), move the cursor (Cursor Position), and set character attributes (Set Attributes, Reset Attributes).

The workstation can create more complicated screen displays, including:

- nested displays in which one program block executes others (ES, ER, and EX commands)
- displays that loop or re-execute themselves
- programs with conditional commands (IF command)
- timed displays (PAUSE command)
- displays that include the date and time

4.4 TRANSFERRING DATA TO/FROM THE WORKSTATION

In the course of execution, a program block can read data directly from the serial ports, data registers, PLC data registers, touch screen, keypad or keyboard. It can then arrange this data in any way and display it on the screen. This allows a screen display to be constantly updated based on data being received by the workstation.

In addition, execution of a program block can write data to the serial ports. Execution of such a program block can be initiated from the serial ports by a simple command. This program block can perform functions such as transmitting data out the serial ports. In this way you can transmit data from the keyboard to another device. Or any device can use the workstation to output data serially. Since the workstation is intelligent and has memory to accommodate large program blocks, it relieves the device of the burden of transmitting serial data. In response to a command from a device the workstation can transmit specific data and/or character strings over the serial ports.

With the Transfer command (TR) data can be moved from any of the following sources:

- Serial input ports
- Keyboard and Keypad
- Data Registers

to any of the following destinations:

- Workstation Screen
- Serial output ports
- Data Registers

The Transfer command can also restrict the type and length of data that it will accept from a user. For example, expected data can be specified as all numeric. Then any data not conforming to the restrictions will simply not be accepted.

When the workstation is in operating mode it constantly monitors the serial ports for data, even while it is executing a program block. The workstation stores received data in large queues which are associated with the serial ports. This data will be stored in the queue until a program block reads the serial input port queues.

The workstation can act as a display and communications center for any device, displaying messages on the screen while it handles the operator input and serial communications.

4.4.1 Data Registers

Data registers are 16-bit memory locations which are used for communication between a device and a workstation. Both the device and the workstation can read/write to data registers. There are at least 500 such registers on the workstation, addressed as #1 through #500. Up to 9999 additional registers may be added via the configuration menu (address #501 to #10,499).

The workstation can perform mathematical operations on values stored in data registers. It can also use data register values as parameters for most OIL commands (e.g., specifying the height of a bar graph). Registers can hold values or they can contain pointers to other registers (indirect addressing).

Since data registers are kept in battery-backed CMOS RAM, their contents will remain unchanged when the workstation is reset, goes through a power-down/power-up cycle, or a program block is executed from the editor or the Program Utilities menu.

The contents of registers #1 through #11 are shown below.

Table 4-3. Registers #1 - #11

Register	Contents
#1 - #7	Time and Date
#8	Keyboard Queue
#9	Secondary Serial
#10	Primary Serial
#11	Touch Screen

Registers #1 through #11 are special-purpose registers. Registers #1 through #7 contain the time and date, and are automatically updated by the workstation. Registers #8 through #11 hold the number of characters currently in the keyboard queue, serial input queues, and the touch screen queue (if the workstation has the touch screen option). Registers #1 through #11 cannot be altered by an OIL program running on the workstation. Transferring data into these registers will not alter their values.

4.4.2 Transferring Data to the Workstation

Data in the data queue can be transferred to the screen or to a data register at any time by the Transfer command (TR).

There are two queues for the serial ports: the command queue and the data queue. All information coming into the serial ports will go into the data queue until a command sequence is detected (ESC). Once this is detected, information goes into the command queue, until the command terminator is detected.

The remote command Transfer To Data Register will cause the data imbedded in the command to be written in a specific data register, #12 - maximum register. This command is described in detail in Chapter 6.

4.4.3 Requesting Data from the Workstation

Any device can read data from a data register, independent of any currently executing OIL program, by transmitting the remote command "Transfer From Data Register." In response to this command, the workstation will transmit the contents of the specified data register to the port. (See Chapter 6 for details.)

4.4.4 Transmitting Data from the Workstation

The block command Transfer (TR) can transfer data from a variety of sources to a variety of destinations. Figure 4-1 shows the capabilities of the transfer command.

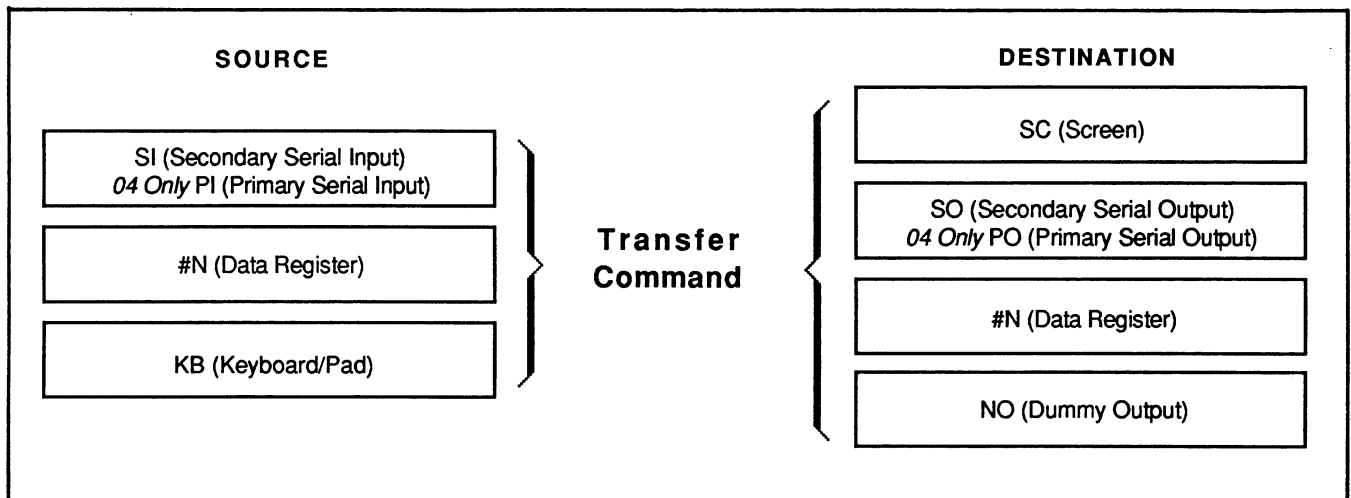


Figure 4-1. The Block Command Transfer (TR)

Transfer can move data from any of the above sources to any destination. For example, if the destination is specified as SO, the data will be transmitted out the secondary serial port. Below are some examples of the Transfer command:

TR KB(picture), SO(picture)

Whatever is typed on the keyboard will be transmitted out the serial port.

NOTE

Pictures are used to control data formatting. They are precise ways of indicating what data should look like when it is input or output. Pictures are described in detail in Section 4.6.58.1.

TR SI(picture) SC(picture)

Whatever is received at the serial port will be displayed on the workstation screen at the current cursor position.

TR SI(picture), #50

Whatever is received at the serial port will be stored in data register #50.

TR #50, SC(picture)
-- or simply --
#50

The contents of data register #50 are displayed on the screen at the current cursor position. There are many more transfer possibilities than these. The transfer command is fully explained in Section 4.6.58.

4.5 PROGRAMMING RULES

Commands can be typed in upper or lower case.

Command parameters can be separated by:

- a single comma (,)
- one or more blanks ()

For example, an SE command could be typed as:

SE BL RV

- or -

SE,bI,RV

- or -

SE BL,RV

Any text appearing between double quotes ("...") is stored as ASCII characters, and is used literally. For a double quote to be used literally, it must be preceded by another double quote:

"Press the button marked ""Emergency Stop"""

Any text following a semicolon is a comment, which is ignored when the program block is executed. (If a semicolon is inside double quotes, it is used literally.) Comments extend to the end of a line:

SE BL RV ;Set attributes to blinking, reverse video

Since the Program Utilities item "Directory" displays the first line of any program, it is helpful if the first line is a comment which gives the name and function of the program:

;WARNING MESSAGE -- print "Warning" in a box

Commands are usually entered on separate lines. However, more than one command can be included on a line if the commands are separated by a colon (:). For example:

```
@25,25 : "Hello"      ;move cursor and write message
```

is equivalent to:

```
@25,25  
"Hello"
```

Note that a line technically ends whenever a <Return> character is pressed. This means that a line can wrap around to the next line on the workstation screen. However, a single command cannot contain more than 255 characters. If a long message seems to have been cut short, it has probably exceeded this limit.

In the following commands and their examples, variables m and n are interchangeable as are the direct register (#) and indirect register (\$). For more information please refer to Section 5.2, Data Registers.

4.6 COMMANDS

The commands are listed in the same order as found in Table 4.1, which may be used as an index to the commands.

4.6.1 User Interface Considerations

The user interface gives you the ability to monitor or change areas of the network devices via the OIL command language. The way you "get at" the device is different for each PLC interface. The protocols are specified in the appendices for each PLC-supporting firmware chip.

4.6.2 Screen Text ("...")

Text surrounded by double quotes (") is displayed on the screen at the current cursor position. Note that the attributes of the text (e.g. its size and whether it is underlined, blinking, etc.) can be specified by the Set Attributes (SE) command.

Example: "Warning"

"The line is DOWN!!"

If the quotes appear in the screen text, they must be preceded by another double quote:

"Press the ""Alarm Reset"" switch" ;will cause the following to be displayed:

Press the "Alarm Reset" switch

The maximum number of characters within a single pair of double quotes is 255.

4.6.3 Register Value Display (#, \$ or &)

Syntax: #reg
 \$reg

where: reg is a register number (1 - maximum register)

Typing "#" followed by the data register number will cause the decimal value of the register to be displayed on the screen.

Typing "\$" followed by a register number will display the decimal value of the register pointed to by the specified register.

For example, if register 20 contains the decimal value 30, "\$20" will display the contents of register 30.

Registers printed by this method are:

- Leading Zero Blanked
- Left Justified
- Space Padded Right to 5 Characters

NOTE

The upper limit of the register is determined by the number of extra data registers selected in the configuration menu. If all 9999 are selected, the maximum register number is 10499.

Example 1:

```
#73           ;display decimal value of register 73  
$89           ;display decimal value of register pointed to by register 89  
&91           ;display hexadecimal value of register 91
```

Example 2:

Register	Displayed
05000	5000 —
00050	50 ——
00000	0 ——

where: __ = space

4.6.4 Position Cursor (@)

Syntax: @x,y

This command moves the cursor to a specified position on the screen.

where: x is the column coordinate, ranging from 0 through 79
y is the row coordinate, ranging from 0 through 24

Example 1:

@10,20

This example moves the cursor to column 10, row 20

Example 2:

@#21,#22

This example moves the cursor to the column specified in register #21 and the row specified in register #22

Example 3:

@15,

This example moves the cursor to column 15 of the current row

Example 4:

#,23

This example positions the cursor on row 23 of the current column

4.6.5 Label (%)

Syntax: GO label

where: label can be any sequence of alphanumeric characters (of any length of up to 255 characters) beginning with a letter.

A label can appear anywhere in a line.

NOTE

A label appearing by itself in a label statement must begin with %. However, the label parameter included within the GO command must omit the % character. For example:

```
%label:"ABC"  
GO label
```

NOTE

Both upper and lower case letters may be used, and the two are regarded as different, e.g. the following labels are all different: start1, START1, Start1.

Example 1:

```
"This is a test":NL:%start:"Hello":NL:go,start.
```

Example 2:

This example will print the message "Hello" on every line of the screen:

```
%start  
"HELLO"  
NL ; new line  
GO start
```


4.6.6 Add to Register (ADD)

Syntax: ADD num,reg

where: num is a decimal value (0 - 65535), a hexadecimal value (&0 - &FFFF), or a register
 reg is a register (12 - maximum register up to 10499)

The ADD command adds a register value to another value. If the sum is greater than 65535, an overflow will occur. The result is stored in the specified register.

Example 1:

If register #100 contains the number 5 and register #12 the value 21, after this command:

```
ADD #100,#12
```

Register #12 will contain the value 26.

Example 2:

```
ADD 100,#46
```

This example will add 100 to register #46.

4.6.7 AND

Syntax: AND reg,num

where: reg is a register (12 - maximum register)
 num is a decimal value (0 - 65535), a hexadecimal value (&0 - &FFFF), or a register

The AND command performs a bit-wise logical AND of the data in the register and the numeric value. The result is stored in the specified register.

Example 1:

```
AND #20,31
```

This example will logically AND the contents of register #20 with the number 31 and place the result in register #20.

Example 2:

```
AND #20,1
```

This example will change bits 2 - 15 to zero and leave bit zero unchanged.

4.6.8 Beep (BP)

Syntax: BP

This command causes the workstation to emit a short beep.

Example:

```
BP
```

4.6.9 Draw Box (BX)

Syntax: BX bval,[xstart],[ystart],xend,yend

where: bval specifies the box outline in the BX command:
1 = thick-line box
2 = thin-line box
any other character enclosed in double quotes = a box composed of that character

xstart is the column coordinate of the top left corner

ystart is the row coordinate of the top left corner

xend is the column coordinate of the bottom right corner

yend is the row coordinate of the bottom right corner

BX draws the outline of a box with the top left corner at (xstart,ystart) and the bottom right corner at (xend,yend). (Note that x indicates the column, y the row.)

Example 1: BX 1,20,10,30,20

This example will draw a thick-line box with top left corner at col. 20, row 10 and bottom right at col. 30, row 20.

Example 2: BX "*",20,10,30,20

This example will create a box outlined with asterisks.

Example 3: BX 1,,,30,20

This example will create a box whose upper left corner is at the current cursor position.

4.6.10 Clear Buffer (CB)

Syntax: CB device

where: device is the data handler:

KB is the keyboard or keypad (Register #8)

PI is primary serial input (Register #9)

PO is primary serial output

SI is secondary serial input (Register #10)

SO is secondary serial output

TS is touch screen input (Register #11)

The Clear Buffer command (CB) empties one of the device buffers.

Example: CB,KB

This example clears the keyboard queue.

4.6.11 Clear Line (CL)

Syntax: CL

This command clears to the end of the line in which the cursor is located.

Example: CL

4.6.12 Clear Bit (CLRB)

Syntax: CLRB reg,bit

where: reg is the register in which the bit is located (12 - maximum register)

bit is a bit in the register (0 - 15, 0 = LSB)

The CLRB command clears a specified bit in a specified data register to 0, leaving the other bits unchanged.

Example: CLRB #44,7

This example will clear bit 7 of register #44 to 0.

4.6.13 Clear Screen (CS)

Syntax: CS

This command clears the entire screen and puts the cursor in the upper left hand corner of the screen.

Example: CS

4.6.14 Clear Window (CW)

Syntax: CW

This command clears the portion of the screen where the cursor is positioned -- either the scrolled or non-scrolled (i.e. status lines) portion.

Example: CW

4.6.15 Down (D)

Syntax: D[num]

where: num is a decimal value (1 - 24), a hexadecimal value (&0 - &FFFF), or a ~~register~~

This command moves the cursor the number of positions specified in the argument. If the argument is omitted, the cursor is moved one position.

If the cursor is on the bottom line and the D command is executed, the screen will scroll, unless Screen Scrolling attribute has been reset by a Reset Attributes command (RE SS). The cursor will wrap around to the top instead of scrolling when scroll is disabled.

If quad-size or double-high attributes are selected by the SE command, the number of lines moved down by a "D" or "D,n" command will be greater than n. See your workstation manual for a description of cursor movement with large characters.

Example: D #12

This example moves the cursor down the number of lines specified in register #12.

4.6.16 Decrement Data Register (DEC)

Syntax: DEC reg

where: reg is the register to be decremented

The DEC command decrements the specified register, i.e. subtracts 1 from its contents. Note that 0 decrements to 65535.

Example 1:

If register #24 contains the value 45, after the DEC command it would contain 44.

Example 2: DEC #47

This example will decrement register 47 by one.

4.6.17 Define Zone or Zones (DFZ) (Touch Screen Only)

Syntax: DFZ upper left zone,lower right zone,num

where: upper left zone is the upper left boundary of the zone to be defined
lower right zone is the lower right boundary of the zone to be defined
num is a decimal (1-255) or hexadecimal (&1-&FF) number

With OIL, a touch screen zone will return its zone number when pressed (ie., zone 1 returns a 1, zone 8 returns an 8). The DFZ command sets a zone or block of contiguous zones to return a common code when pressed.

Example 1: DFZ 1,18,&81

This example causes touch screen zones 1-8 and 11-18 to return the value 81H when pressed. Other zones remain unaffected.

Example 2: DFZ 12,12,&45

This example causes the single touch screen zone 12 to return the value 45H when pressed. Other zones remain unaffected.

NOTE
On power-up, touch screen zones are reset to return their zone numbers.

4.6.18 Divide Register (DIV)

Syntax: DIV reg,num

where: reg is a register (12 - maximum register)
num is a decimal value (1 - 255), hexadecimal value (&1 - &FF), or register

The DIV command divides a register value by another value, dropping the remainder.

In DIV the num value cannot be zero.

The result is stored in the specified register.

Example: TR #20,#80
DIV #80,10

If register #20 contained the value 36, after the above program is executed registers #80 will contain the value 3.

4.6.19 Display String (DS)

Syntax: DS reg

where: reg is the beginning register

The DS command displays a string of characters contained in multiple registers. The display will begin with a string of characters starting in the specified register, and will end when the zero terminating character is encountered.

This command is used in conjunction with the SS (Store String) command.

Example: SS "Caution",#20 ;store the string in reg. 20-23
DS #20 ;display the string in reg. 20-23

This program will store the word "Caution" in registers 20-23 and display it on the screen.

4.6.20 Execute Program and Reset Subroutine Counter (ER)

Syntax: ER num

where: num is the number of the program block to be executed

The ER command executes the program block specified and resets any subroutine nesting. Unlike the ES command, control does not return to any statements following the ER command.

Example 1:

ER #68

This example executes the program block whose number is contained in register #68.

Example 2:

ER 3

This example executes program block 3.

4.6.21 Execute Subprogram Block (ES)

Syntax: ES num

where: num is the number of the next subprogram block to be executed (1 - 255)

The ES command causes the workstation to execute another block, then return to finish execution of the current one. (Note that the EX or ER command does not return to the current program block.)

For example, suppose an ES command is located in the middle of program block 20:

program block 20

command 1
command 2
ES 34
command 3
command 4
command 5

After command 2 has been executed, execution of program block 34 begins. After execution of program block 34 is completed, execution of program block 20 resumes with command 3.

Program blocks containing ES commands are called "nested" programs. Sub-programs may be nested ten programs deep. Nested programs are described in the next chapter.

Example 1: ES 230

This example executes sub-program block 230.

Example 2: ES #68

This example executes the sub-program block whose number is contained in register #68.

4.6.22 Execute Program Block (EX)

Syntax: EX num

where: num is the number of the next program block to be executed (1 - 255)

The EX command causes a program block to be executed.

For example, suppose that the last command in program block 21 is: EX 22. Then after execution of program block 21 has been completed, execution of program block 22 can begin. In this way several program blocks can be chained together.

Program blocks can also execute themselves. For example, if program block 121 contained "EX 121," it would be re-executed continuously.

Example 1: EX 230

This example executes program block 230.

Example 2: EX #68

This example executes the program block whose number is contained in register #68.

4.6.23 Exit OIL Program (EXIT)

Syntax: EXIT

The EXIT command is used inside a sub-program block (which will be called by another via the ES command). It exits the sub-program block and returns to the calling program block. If the program block containing the EXIT command was not called by another block, EXIT acts just like STOP: it terminates the program.

Example: EXIT

4.6.24 Draw Filled Box (FBX)

Syntax: FBX bval,xstart,ystart,xend,yend

where: bval specifies the character which will comprise the filled box in the FBX command. It can be any character (enclosed in double quotes, e.g. ">") or a number corresponding to a character, which will draw a box composed of that character.

xstart is the column coordinate of the top left corner

ystart is the row coordinate of the top left corner

xend is the column coordinate of the bottom right corner

yend is the row coordinate of the bottom right corner

FBX draws a filled box composed of the bval character. For example, to draw a solid box, use 221 as the bval character. Experiment with other characters for different fill patterns.

NOTE

With the FBX command, unlike the BX command, if the xstart = xend and ystart = yend, the end result will be a box that is one character cell wide or character cell high.

Example 1: FBX 221,20,10,#40 #41

This example draws a shaded box with bottom coordinates found in registers #40 (column) and #41 (row).

Example 2: FBX "*" ,20,10,30,20

This example draws a box filled with asterisks.

4.6.25 Go (GO)

Syntax: GO label

where: label can be any sequence of alphanumeric characters (of any length of up to 255 characters) beginning with a letter. Both upper and lower case letters may be used, and the two are regarded as different, i.e. the following labels are all different: start1, START1, Start1.

The GO command allows non-sequential execution of commands. In other words, instead of commands being executed in the order in which they appear in the block, a GO statement can be used to jump to any label in the program block. Execution will continue with the command following the label.

NOTE

A label appearing by itself in a label statement must begin with %. However, the label parameter included within the GO command must omit the % character. For example, %label:"ABC", but GO label.

GO commands are often combined with IF statements to provide conditional branching.

Example: %start
"HELLO"
NL ; new line
GO start

This example will print the message "Hello" on every line of the screen.

4.6.26 Horizontal Bar Left (HBL)

Syntax: HBL xstart,ystart,lngth,maxlngth

where: xstart	is the column coordinate of the left end
ystart	is the row coordinate of the top left end
lngth	is the length of the bar, in units of a character cell*
maxlngth	is the maximum length of the bar, in units of a character cell*

This command functions just like the Horizontal Bar Right command, except that the bar extends to the left from the (xstart,ystart) coordinates (see HBR command in the next section). The background will be erased before a bar is drawn.

Example: HBL 79,0,200,201

This example will draw a bar leftwards from right edge of the screen.

*Each character cell is 5 units wide (1 unit = 1/5 character cell). The maximum value for lngth and maxlngth is 255.

4.6.27 Horizontal Bar Right (HBR)

Syntax: HBR xstart,ystart,lngth,maxlngth

where: xstart	is the column coordinate of the left end
ystart	is the row coordinate of the top left end
lngth	is the length of the bar, in units of a character cell*
maxlngth	is the maximum length of the bar, in units of a character cell*

This command draws a bar of length lngth towards the right from (xstart,ystart). If either coordinate is absent, the current cursor coordinates will be used. Before the new bar is drawn, a bar length maxlngth is erased. The parameters lngth and maxlngth must specify the length in units of a character cell. For example, to draw a bar $6 \frac{3}{5}$ character cells wide on a monochrome workstation, lngth would be 33. The parameter maxlngth specifies how long a bar to erase before the bar of lngth is drawn. Therefore, maxlngth should be no less than lngth.

The background will be erased before a bar is drawn.

Example 1: HBR 10,20,60,150

This example will draw a horizontal bar rightwards 60 units long from the cursor position (10,20).

Example 2: HBR #51,#52,200,200

This example will draw a horizontal bar with column and row coordinates found in registers #51 and #52 respectively, length of bar is 200 units.

Example 3: HBR,,100,100

This example will draw a horizontal bar extending rightwards from current cursor position.

*Each character cell is 5 units wide (1 unit = $\frac{1}{5}$ character cell). The maximum length for lngth and maxlngth is 255.

4.6.28 Draw Horizontal Line (HL)

Syntax: HL lval,xstart,ystart,hlngh

where: lval can have the following arguments:
1 = thick line
2 = thin line
3 = GR3 character !
4 = GR3 character #
5 = GR3 character)
6 = GR3 character +
any other character (enclosed in double quotes, e.g. ">") or number
corresponding to a character = a line composed of that character

xstart is the column coordinate of the left end

ystart is the row coordinate of the top left end

hlngh is the number of characters the line extends to the right

Draws a line of length hlngh to the right from (xstart,ystart). (Note that x indicates the column, y the row.) The lval parameter specifies what the line will be composed of.

Example 1: HL 1,10,20,15

This example will draw a thick line 15 characters long whose left end is at col. 10, row 20.

Example 2: HL 2,#40,#41,#20

This example will draw a thin line with left end coordinates found in registers #40 (column) and #41 (row), and with length in register #20.

Example 3: HL ">",10,20,10

This example will draw a line composed of ">."

4.6.29 CONDITIONAL (IF)

Syntax: IF num,cond,num,[:command]

where: num and num are two registers or numerical values (0 - 65535) that are being compared

cond can be any of the following relations:

<u>Relation</u>	<u>Meaning</u>
=	equals
<	less than
>	greater than
<= or =<	less than or equal to
>= or =>	greater than or equal to
<>	not equal to

command is any OIL command or string separated by colons

NOTE

Spaces cannot be included between "=" and the adjacent parameters. In other words, the following commands are not valid.

```
IF #45> #50:.....  
IF #45<= #50:.....  
IF #45 =#50:.....
```

The IF command compares the value of one register with the value of another register or a numerical value. If the condition specified is true, the commands on the same line as the IF command will be executed. If the condition is false, the following commands on that line will be skipped, and execution will continue at the beginning of the next line.

Example 1: IF #45=#63: "Too much solvent has been added"

In this example, the message will be displayed only if the contents of registers #45 and #63 are equal.

IF #50>0: "WARNING": TR 1,#51: ES 20

If Register #50 is greater than 0, all commands on the same line will be executed

IF #43<=230: GO label1

Go to label1 if the contents of register #43 is less than or equal to 230.

IF #17=0:ES 45

If the content of register #17 is zero, execute sub-program block #45.

Example 2: "OIL PRESSURE IS TOO"

IF #54=0: " LOW"

IF #54=1: " HIGH"

This example prints the message "OIL PRESSURE IS TOO LOW" if the value of register #54 is 0, "OIL PRESSURE IS TOO HIGH" if the value of register #54 is 1.

Example 3: IF #200>5: IF,#200<8: GO RETRY

The third example jumps to the label RETRY if the value of register #200 is 6 or 7.

4.6.30 IF Bit (IFB)

Syntax: IFB reg,bit

where: reg specifies a register (1 - maximum register)
bit is a bit in a register (0 - 15, 0 = LSB)

The IFB command checks to see if the specified bit is 1. If it is, all subsequent OIL commands in the same line will be executed. If the bit value is 0, all subsequent OIL commands on the same line will be skipped.

Example 1: IFB #64,0: ES 100: go label
ES 101
%label

If bit of register 640 is 1, execute program block 100, otherwise execute program block 101.

Example 2: IF #59=#70: GO goodbye
BX 1,20,5,63,19 ; Draw thick-line box
@36,12 ; Move cursor inside box
"Hello"
STOP
%goodbye
BX 2,20,5,63,19 ; Draw thin-line box
@36,12 ; Move cursor inside box
"Goodbye"
STOP

If the contents of registers #59 and #70 are equal, this program will print the message "Goodbye" in a thin-line box. If they are not equal, it will print the message "Hello" in a thick-line box.

4.6.31 If String (IFS)

Syntax: IFS reg,cond,reg[:command]

where: reg and reg specify two strings to be compared

cond can be any of the following relations:

<u>Relation</u>	<u>Meaning</u>
=	equals
<	less than
>	greater than
<= or =<	less than or equal to
>= or =>	greater than or equal to
<>	not equal to

command is any OIL command or string separated by colons

NOTE
Spaces cannot be included between "=" and the adjacent parameters.
In other words, the following commands **are not** valid.

IFS #45> #50:.....
IFS #45<= #50:.....
IFS #45=#50:.....

The IFS command compares the contents of one register or text string with another register or text string. If the condition specified is true, the commands on the same line as the IFS command will be executed. If the condition is false, the following commands on that line will be skipped, and execution will continue at the beginning of the next line.

Example 1: SS "ABCDEFGG",#20 ;store the string
SS "1234567",#50 ;store the string
IFS #20=#50:"Strings are equal":stop ;compare, message
"strings are not equal" ;other message

When the above program is run, the message "strings are not equal" will be displayed.

Example 2: SS "ABCDE",#21 ;store the string
SS "ABCDG",#51 ;store the string
DS #21:NL ;display string
DS #51:NL ;display the string
IFS #21=#51:"strings are equal":stop ;compare equal
IFS #21<#51:"#21 < #51":stop ;compare less
IFS #21>#51:"#21 > #51":stop ;compare greater

When the above program is run, the following message will appear on the screen:

```
ABCDE
ABCDG
#21 < #51
```

Example 3: SS "1234567",#25 ;store the string
IFS "1234567"=#25:"strings are equal":stop ;compare equal
"strings are not equal" ;new message

When the above program is run, the following message will appear on the screen:

```
strings are equal
```

4.6.32 Increment Data Register (INC)

Syntax: INC reg

where: reg is the register to be modified (12 - maximum register)

The INC command increments the specified register, i.e., adds 1 to the contents of the specified register.

<p style="text-align: center;">NOTE 65535 will increment to 0.</p>

For example, if register #24 contains the value 45, after an INC command it would contain 46.

Example: INC #45

4.6.33 Exit ONKEY Subroutine Program (KEXIT)

Syntax: KEXIT

The OIL command KEXIT terminates the subroutine program block that the OIL command ONKEY specified.

<p style="text-align: center;">NOTE This command MUST terminate the subprogram block specified by the ONKEY command or the workstation will not re-enable ONKEY trapping upon completion of the routine (see ONKEY).</p>

Example: TR,KB(X) #30
IF #30="y":TR,"Y",#30:KEXIT
IF #30="n":TR,"N",#30:KEXIT
IF #30<>"Y":IF,#30<>"N":BP:TR,0,#30
KEXIT

Transfer character out of keyboard buffer, change to upper case (if "y" or "n" is selected), put the value in register #30, and terminate. It will sound the beeper and zero response if invalid, then terminate.

This example is constructed to complement the ONKEY example (refer to Section 4.6.40).

4.6.34 Keypad Status (KS)

Syntax: KS reg

where: reg is a register (12 - maximum register)

The Keypad Status command allows you to find out if a key is pressed on the Keypad, and if so, which key it is. This status is placed into a data register. If the register value is zero, no key is pressed, if the register is a non-zero, the value is the code for the key being pressed. Note that this status is only for the keypad on the workstation, not for the external serial or matrix keyboard.

Example: KS #20

This example puts the current keypad status into register #20.

4.6.35 Left (L)

Syntax: L[num]

where: num is a decimal value (0 - 80), a hexadecimal value (&0 - &FFFF), or a register

This command moves the cursor the number of positions specified in the argument. If the argument is omitted, the cursor is moved one position.

If quad-size or double-high attributes are selected by the SE command, the number of lines moved over by a "L" or "L,n" command will be greater than n. The same is true of the "R" command with quad-size or double-high characters. See the 2000 Industrial Workstation Manual for a description of cursor movement with large characters.

L causes the cursor to wrap around to the end of the line above if it encounters the beginning of a line. L has no effect if the cursor hits the top left of the screen.

Example: L 65

This moves the cursor left 65 spaces.

4.6.36 Remainder (MOD)

Syntax: MOD reg,num

where: reg is a register (12 - maximum register)
num is a decimal value (0 - 65535), a hexadecimal value (&0 - &FFFF), or a register

The MOD command divides a register value by another value, but drops the result and keeps the remainder. In MOD the m value cannot be zero and must be less than 256.

The result is stored in the specified register.

Example 1: TR #20,#80
TR #20,#90
DIV #80,10
MOD #90,10

If register #20 contained the value 36, after the above program is executed registers #80 and #90 will contain the following values:

#80 -- 3
#90 -- 6

This arithmetic command changes the value of the specified register.

Example 2: tr,100,#80
tr,10,#90
tr,#80,sc(ddddd)
NL:tr,#90,sc(ddddd)
NL:div,#80,100
tr,#80,sc(ddddd)
NL:MOD,#90,100
tr,#90,sc(ddddd)
STOP

If you execute the above example, the following will be displayed:

00100 ; contents of Register 80
00010 ; contents of Register 90
00001 ; the result of the DIV command
00010 ; the remainder of the MOD command

4.6.37 Multiply Register (MUL)

Syntax: MUL reg,num

where: reg is a register (12 - maximum register)
num is a decimal value (0 - 65535), a hexadecimal value (&0 - &FFFF), or a register

The MUL command multiplies a register value by another value. The m value must be less than 256. If the product is greater than 65535, an overflow will occur.

The result is stored in the specified register.

Example: MUL #24,2

If register #24 contained the value 12, it will contain the value 24 after the MUL command is executed.

4.6.38 New Line (NL)

Syntax: NL num

where: num is a decimal value (0 - 65535), a hexadecimal value (&0 - &FFFF), or a register

If no number is specified, this command moves the cursor to the beginning of the next line. If num is specified, this command moves the cursor down num lines and then to the beginning of the line.

If the cursor is on the bottom line and the NL command is executed, the screen will scroll, unless Screen Scrolling attribute has been reset by a Reset Attributes command (RE SS). If Screen Scrolling is Reset, the cursor will wrap around to the top instead of scrolling when scroll is disabled.

If quad-size or double-high attributes are selected by the SE command, the number of lines moved down by the "NL" or "NL,n" command will be greater than n. See the 2000 Industrial Workstation Manual for a description of cursor movement with large characters.

Example 1: NL

This moves the cursor to the beginning of the next line.

Example 2: NL 3

This moves the cursor to the beginning of the line, 3 lines down.

Example 3: NL #23

This moves the cursor down the number of lines specified in register #23, (to the beginning of the line).

4.6.39 NOT

Syntax: NOT reg

where: reg is a register (12 - maximum register)

NOT performs a one's complement of the data in the specified register. I.E., all ones are changed to zero and all zeroes are changed to ones.

Example: NOT #20

This example of NOT performs a one's complement on the contents of Register 20 and the result is placed in Register 20.

4.6.40 ONKEY

Syntax: ONKEY num

where: num is program block to be executed (1-255).

This command is used within a program block and will be executed whenever any keyboard key, function key, or keypad key is pressed. ONKEY 0 disables the ONKEY feature.

Because ONKEY uses a key in the keyboard buffer, care must be taken to assure that the character is removed by the subprogram block specified in the ONKEY command. If the character is not removed, the specified subprogram block will be executed continuously (see KEXIT).

Example 1: ;ONKEY example - Wait for Y or N
TR,0,#30 ;Zero response register
ONKEY 2 ;Execute subprogram block 2 when key is pressed
"Press Y or N" ;Print prompt
%Wait ;Wait loop
If,#30=0:GO,Wait ;Loop until #30 is non-zero
"O.K." ;Print final message

This example will zero register #30, execute subprogram block 2 when a key is pressed, print the prompt, go into a wait-loop until register #30 is non-zero, and print a final message. This example is constructed to complement the KEXIT example (refer to Section 4.6.33).

4.6.41 ONTOUCH (Touch Screen Only)

Syntax: ONTOUCH num

where: num is program block to be executed (1-255).

This command is used within a program block and will be executed whenever any zone of the touch screen is pressed. ONTOUCH 0 disables the ONTOUCH feature.

Because ONTOUCH uses a zone in the touch screen buffer, care must be taken to assure that the character is removed by the subprogram block specified in the ONTOUCH command. If the character is not removed, the specified subprogram block will be executed continuously (see TEXIT).

Example 1: ;ONTOUCH example - Display zone number on screen
TR,0,#30 ;Zero response register
ONTOUCH 2 ;Execute subprogram block 2 when a zone is pressed
"Touch Zone Pressed:" ;Print prompt
%Disp ;Display loop
@20,0 ;Position cursor
TR,#30,SC(DDD) ;Display zone pressed
GO,Disp ;Loop

This example will zero register #30, execute subprogram block 2 when a touch screen zone is pressed, and continuously print the touch screen zone number that is pressed.

4.6.42 OR

Syntax: OR reg,num

where: reg is a register (12 - maximum register)
num is a decimal value (0 - 65535), a hexadecimal value (&0 - &FFFF), or a register

The OR command performs a bit-wise logical OR of the data in the register and the numeric value. The result is stored in the specified register.

This command can be used to set particular data bits in a register.

Example: OR #20,31

This ORs contents of register 20 with the number 31 and the result is placed in register 20.

4.6.43 Plot

Syntax: PLOT xpoint,ypoint

where: xpoint is the column coordinate of the point (0-159)

ypoint is the row coordinate of the point (0-71).

NOTE

The 0,0 coordinate for the Plot command is not the upper left corner (as in the other commands), but the lower left corner (as you would expect when plotting points on a graph).

Each Plot command will draw a single square 1/6 the size of a character cell.

Example: PLOT 0,0

This displays a point in the lower left corner of the screen.

4.6.44 Pause (PS)

Syntax: PS time

where: time is the number of time units to pause (each unit is 1/10 sec.)

The PS command will cause an executing program block to pause for a specified length of time (in 1/10 second units) before continuing execution. This command is most useful in showing sequences of displays. For example, three different screens can be displayed, each screen being displayed for 10 seconds before being replaced by the next screen in the sequence. It can also be used to create "animated" displays in which the entire screen does not appear at once, but certain areas appear before others.

Example 1:

```
%loop          ;program will continuously loop
CS             ;clear screen
SE DS         ;select double-size attributes
@25,10        ;position cursor
"LINE IS DOWN"
PS 100        ;pause ten seconds
CS             ;clear screen
@25,10        ;position cursor
"SEE SUPERVISOR"
PS 100        ;pause ten seconds
GO loop
```

This example displays two different messages on the screen, each for 10 seconds.

Example 2:

```
BX 1,10,10,50,20 ;draw a box
PS 20            ;pause for two seconds
@25,15          ;move cursor inside box
"HELLO"         ;display message
```

In this example, the box is drawn two seconds before the text in the box.

4.6.45 Right (R)

Syntax: R[num]

where: num is a decimal value (0 - 80), a hexadecimal value (&0 - &FFFF), or a register

These commands move the cursor the number of positions specified in the argument. If the argument is omitted, the cursor is moved one position.

R causes the cursor to wrap around to the beginning of the line below. R has no effect if the cursor hits the bottom right of the screen.

Example: R 20

This will move the cursor right 20 spaces.

4.6.46 Reset Attributes (RE)

Syntax: RE attr,attr,...
or RE,ALL

Reset Attributes turns off the specified attributes. The command RE ALL turns off all attributes except CU and SC. See Set Attributes Command (SE), Section 4.6.51.

Example: RE,QS,BL

This will turn quad-size and blinking to OFF.

4.6.47 Restore Attributes (RSTA)

Syntax: RSTA reg

where: reg is a register number (12 - maximum register)

This command restores the attribute values saved in the specified register by an earlier Save Attributes (SAVA) command. See Save Attributes Command (SAVA) in Section 4.6.49.

4.6.48 Restore Cursor (RSTC)

Syntax: RSTC reg

where: reg is a register number (12 - maximum register)

This command restores the cursor position saved in the specified register by an earlier Save Cursor (SAVC) command (see Section 4.6.50).

4.6.49 Save Attributes (SAVA)

Syntax: SAVA reg

where: reg is a register number (12 - maximum register)

This command saves the current settings (on or off) of all the attributes in the specified data register. To restore these settings at some later time, issue the Restore Attributes command (RSTA) (see Section 4.6.47).

4.6.50 Save Cursor Position (SAVC)

Syntax: SAVC reg

where: reg is a register number (12 - maximum register)

This command saves the current cursor position in the specified register. To restore the cursor position, use the Restore Cursor Position (RSTC) command (see Section 4.6.48).

4.6.51 Set Attributes (SE)

Syntax: SE attr,attr,...

where: attr is any of the following values:

QS	Quad-Size characters (four times as high, four times as wide)
DS	Double-Size characters (double high and double wide)
RV	Reverse Video characters
HI	High Intensity characters
UN	Underlined Characters
BL	Blinking Characters
DH	Double-High characters
DW	Double-Wide characters
CU	Cursor on or off
SC	Screen output
PR	Printer output
SS	Screen Scrolling
G1	Process graphics
G2	Thin-line and block graphics
G3	Process control graphic connectors
G4	Mini Process graphics

SE,ALL is ignored

The Set Attributes command determines how text will be displayed on the screen. SE turns an attribute on.

If more than one attribute is specified in a command, they are set (or reset) from right to left. For example, "SE QS DS" will leave the text display in quad-size mode.

Once an attribute has been selected, it remains in effect for all characters subsequently displayed on the screen. Therefore, if you only want some of the screen text to have a specific attribute, you must reset the attribute (turn it off) with the Reset Attributes (RE) command. Some of the attributes are described in greater detail on the next page.

QS, DS, DH, DW - Refer to the 2000 Industrial Workstation Manual.

UN - Underline. When this attribute is set, underline will be enabled. Cannot be used with QS, DS, DH, G1, G2, G3, G4.

SC - Screen Output. When the workstation is powered up, this attribute is automatically set (turned on). This means that all screen text (characters surrounded by double quotes, e.g. "Warning") generated by an executing program block is displayed to the screen.

If this attribute is reset (turned off) by the Reset Attributes (RE) command, screen text will no longer be displayed on the screen.

PR - Printer output. In addition to being displayed on the screen, screen text (characters surrounded by double quotes) will also be transmitted out the secondary port. (A printer can be connected to the secondary port). However, bars, lines, boxes, and other graphics will not be transmitted out the secondary port. Only the character and not its attributes will be transmitted. For example, if QS and UN are in effect, the message "Warning" will be transmitted to the printer in regular-sized, non-underlined letters. Also, if G1 is in effect, the letters, not the corresponding graphics symbols, will be transmitted to the printer.

This attribute is set to off when the workstation is powered up or reset.

SS - Screen Scrolling. If SS is reset the cursor will not move beyond the last boundaries of the screen (i.e., bottom and right), even if ordered to by the cursor movement commands. Normally, moving the cursor below the last line of the screen will cause the screen to scroll upwards.

G1 - Process Graphics characters. If this attribute is set, text in quotes will not be displayed as letters. Instead, the text will be displayed as Process Graphics characters (see the 2000 Industrial Workstation Manual).

G2 - Thick and Thin Line Graphics characters. If this attribute is set, text in quotes will not be displayed as letters, the text will be displayed as Thick and Thin Line Graphics characters (see the 2000 Industrial Workstation Manual).

G3 - Process Control Graphics Connectors. If this attribute is set, text in quotes will not be displayed as letters, the text will be displayed as Process Control Graphics Connectors (see the 2000 Industrial Workstation Manual).

G4 - Mini Process Graphics. If this attribute is set, text in quotes will not be displayed as letters, the text will be displayed as Mini Process Graphics characters (see the 2000 Industrial Workstation Manual).

Example 1: SE QS,BL
"WARNING"
RE QS,BL
NL 5
"See supervisor" ;printed in standard letters

This example sets the character attributes to quad-size, blinking characters, then the word "WARNING" is displayed with those attributes. Quad-size and blinking are turned to OFF, and the cursor is moved down 5 lines. Then the message "See supervisor" is printed in normal letters.

Example 2: SE G1 ;Set attributes to graphics characters
"IJ" ;"I" = left half of tank, "J" = right half
RE G1 ;Reset attributes to cancel graphics characters
NL 5 ;Move cursor down 5 lines
"IJ" ;Displays the two letters "I" and "J"

This program will display the left and right half of a tank bottom using graphics characters, and the letters "I" and "J."

4.6.52 Set Bit (SETB)

Syntax: SETB reg,bit

where: reg is the register in which the bit is located (12 - maximum register)
bit# is a particular bit in the register (0-15), with 0 being the LSB

The SETB command sets a specified bit in a specified data register to 1, leaving the other bits unchanged.

Example: SETB #44,7

This will set bit 7 of register #44 to 1.

4.6.53 Stop Program Execution (STOP)

Syntax: STOP

The STOP command terminates execution of the currently executing program.

Example: STOP

4.6.54 Store String (SS)

Syntax: SS text,reg

where: text is a string of data to be stored, offset by double quotes

reg is the beginning register to store the consecutive characters in, preceded by the # sign

This command stores a string of data into consecutive registers, and terminates with a zero. The first character goes in the Least Significant Bit (LSB) of the register specified; the second character goes in the Most Significant Bit (MSB) of the register specified; the third character goes in the LSB of the next consecutive register, and so on.

NOTE

Registers can only hold two characters. Because of the zero terminator, a string of two characters will take two registers.

Example: SS "Hello",#100 ;store string in consecutive registers
DS #100 ;display contents of register

The above program will store the string "Hello" in registers 100, 101, and 102, and will display the message on the screen.

4.6.55 Subtract from Register (SUB)

Syntax: SUB num,reg

where: num is a numeric value to be subtracted

reg is a register (12 - maximum register)

The SUB command subtracts a value from the specified register.

The sub command subtracts a value from the specified register. If the product is less than 0, an overflow will occur.

The result is stored in the specified register.

Example 1: SUB #81,#100

This will subtract the contents of register #81 from register #100.

Example 2: SUB "A",#100

This will subtract 65 (ASCII value of "A") from register #100.

4.6.56 Transfer System Characteristics (SYS)

Syntax: SYS reg

where: reg is the register where data bits will be sent

This command transfers bits defining the 2000 hardware characteristics to a register. Bits indicating monitor type, keyboard type, and whether touch screen is installed will be sent.

The layout for the system configuration bits is as follows:

X X X X X X M M X P P B B R T K

Key: X = reserved bits	BB = matrix keyboard type 00 = QWERTY 01 = ABC
MM = monitor type 00 = monochrome 01 = color 10 = flat panel 11 = EGA	R = location of OIL programs 0 = RAM 1 = ROM
PP = keypad type 00 = type 0 01 = type 1 10 = type 2 11 = type 3	T = touch screen option 0 = no touch screen 1 = touch screen
	K = keyboard type 0 = AT 1 = XT

NOTE

The last eight system configuration bits (XPPB BRTK) correspond to the 8 DIP switches on the back panel of the workstation. See the 2000 Industrial Workstation Manual for more information on these switches.

Example: SYS #56 ;send characteristics to register
TR #56,SC(iiii iiii iiii iiii) ;display bit characteristics on screen

If the workstation is configured for a monochrome monitor, keypad type 1, a QWERTY-type matrix keyboard, OIL programs in RAM, touch screen, and an AT-style keyboard, the following message will appear on the screen:

0000 0000 0000 0010

4.6.57 Exit ONTOUCH Subroutine Program (TEXT) (Touch Screen Only)

Syntax: TEXT

The command TEXT terminates the subroutine program block requested by the OIL command ONTOUCH.

NOTE

This command MUST terminate the subprogram block specified by the ONTOUCH command or the workstation will not re-enable ONTOUCH trapping upon completion of the routine (see ONTOUCH).

Example: TT #30 ;Touch screen zone to register 30
TEXT

The above program will transfer a touch screen zone out of the touch screen buffer, put the value in register #30, and terminate.

4.6.58 Transfer Data (TR)

Syntax: TR source,dest

The workstation has several data input and output devices:

Input Devices

KB (keyboard and keypad)
SI (secondary serial input)
PI (primary serial input - 04 only)
\$n (data register pointer)
Literal (number or string)
&n (hexadecimal number)
#n (data register)

Output Devices

SO (secondary serial output)
PO (primary serial output - 04 only)
\$n (data register pointer)
NO (dummy output -- see
Section 5.4.2)
SC (screen display)
#n (data register)

In addition, a constant can be transferred to any of the above output devices.

The Transfer Data command allows the workstation to transfer data between any source and destination.

4.6.58.1 Pictures

Pictures are used to control data formatting. A picture is a concise way of indicating what data should look like when it is input or output. A picture is a string of characters enclosed in parentheses. Each picture character is treated either as a placeholder or as a literal constant. Placeholder characters indicate the type of data that is allowed in that position in the formatted data. The placeholders are:

a	Alphabetic character
c	Alphanumeric character (a-z, A-Z, 0-9)
d	Decimal digit (0123456789)
h	Hexadecimal digit (a-f, A-F, 0-9)
i	Binary digit (0, 1)
l	Any ASCII character
n	Carriage return or enter
x	Binary byte (not echoed to the screen)
z	Decimal digit with leading zero suppression

Picture elements that are capitalized will not echo data.

Picture elements n and N may appear only once in a picture. If one of these elements appears in a picture, a carriage return will terminate the transfer.

The following are examples of pictures:

(aaaaaa)	Six alphabetic characters filling the field
(ddd)	Two blanks followed by three digits
(dd.dd)	Four digits with decimal point and one blank
(dd:dd:dd)	Six digits with colons
(cccn)	Up to three alphanumerics followed by a return

Note that pictures are not required for data registers.

4.6.58.2 Data Types

There are basically two data types, text and numeric. The pictures used for these two types are grouped as follows:

<u>DATA TYPE</u>	<u>PICTURE ELEMENTS</u>
text	a any alphabetic character (upper/lower case letters) l any ASCII character c any alphanumeric character (letters or numbers) n Carriage Return
numeric	d decimal digit (0 through 9) h hexadecimal digit (0 through 9, A through F, a through f) i binary digit x binary byte z zero decimal digit with suppression, zero's will be replaced by blanks until a non-zero digit is encountered

4.6.58.3 Transferring Text Characters

To transfer text characters, simply specify one text picture element (a,l,c,n,) for every character being transferred. A picture must be included in both the source and destination. Examples follow:

Picture element a

Any alphabetic character (a-z, A-Z).

This picture element should be used whenever only the alphabetic characters are wanted.

Example 1: All characters valid.

```
TR "Hello",dest(aaaaa)
```

The literal string "Hello" is transferred to the destination.

Example 2: Some characters are valid.

```
TR "A12$a",dest(aaaaaa)
```

The literal string "A12\$a" is filtered and "Aa" is transferred to the destination. Note that numeric and special symbols were not transferred.

Example 3: Register transfer (#30 = "Hi").

```
TR #30,dest(aa)
```

The characters "Hi" are transferred to the destination. Note that a register will contain exactly 2 characters, one in the high byte and one in the low byte.

Example 4: Keyboard to destination.

```
TR, KB(aaa),dest(aaa)
```

This example will take the first 3 alpha characters from the keyboard buffer and transfer them to the destination as they are received.

Picture Element c Any alphanumeric character (a-z, A-Z, 0-9).

This picture element should be used whenever only the alpha-numeric characters are wanted.

Example 1: All characters are valid.

```
TR "10KVA,dest(ccccc)
```

The literal string "10KVA" is transferred to the destination.

Example 2: Some characters are valid.

```
TR "A12$#a",dest(cccccc)
```

The literal string "A12\$#a" is filtered and "A12a" is transferred to the destination. Note that the special symbols were not transferred.

Example 3: Register Transfer (#30="A1").

```
TR #30,dest(cc)
```

The characters "A1" are transferred to the destination. Note that a register will contain exactly 2 characters, one in the high byte and one in the low byte.

Example 4: Keyboard to destination.

```
TR,KB(ccc),dest(ccc)
```

This example will take the first 3 alphanumeric characters from the keyboard buffer and transfer them to the destination as they are received.

Picture element l (lower case L), any character.

This picture element should be used whenever filtering is not required.

Example 1: All characters valid.

```
TR "A12$#a",dest(lllll)
```

The literal string "A12\$#a" is transferred to the destination. Note that filtering was not done.

Example 2: Register Transfer (#30="\$1").

```
TR #30,dest(ll)
```

The characters "\$1" are transferred to the destination. Note that a register will contain exactly 2 characters, one is in the high byte and one in the low byte.

Example 3: Keyboard to destination.

TR,KB(III),dest(III)

This example will take the first 3 characters from the keyboard buffer and transfer them to the destination as they are received.

Picture element n Carriage return.

This picture element should be used in the source part of a transfer when you want to terminate the transfer by receiving a carriage return from the source. When used in this way, the transfer is terminated when the carriage return is received, even if the rest of the input has not been completed. Only one can be used. When used in the destination portion of a transfer the n can be used to send a carriage return to the destination. Only one n is allowed per picture.

Example 1: Transfer characters from the source to destination with carriage return terminator.

TR source(IIIIIIIIIn),dest(IIIIIIII)

This will allow up to 10 characters of any kind. After 10 characters are received, a carriage return must be received to terminate the command. If a carriage return was received before the 10 characters are input, the command terminates and the transfer is completed.

Example 2: Carriage return as a constant.

TR source(III),dest(III n)

This will cause the first 3 characters received from the source to be transferred to the destination followed by a carriage return. Note that the n must be at the end of the picture.

4.6.58.4 Numeric Data Types

There are three numeric data types:

actual value (e.g. 65 represented as:	0000 0000	0100 0001)
	00H	41H
	NULL	"A"
ASCII format (e.g. 65 represented as:	0011 0110	0011 0101)
	36H	35H
	"6"	"5"
Hex format (e.g. 65 represented as:	0000 0000	0110 0101)
	00H	65H
	NULL	"e"

When data is being displayed or entered, the ASCII format is desirable because that is what the operator recognizes and what a keyboard device sends. However, if math is going to be done on these numbers or the PLC will be looking at these numbers, the actual values are required. OIL allows great flexibility in the transferring of numerical data and automatically takes care of conversion from ASCII to actual and vice versa. There are five picture elements which can be used to specify how data is to be converted. They are d, z, h, i, x.

Picture element d Decimal digit.

This picture element should be used whenever only the ASCII digits 0-9 are wanted. When used in the source portion of a transfer command, only ASCII digits 0-9 will be allowed.

Example 1: Source to a data register.

```
TR Source(dddd),#30
```

The first 4 valid ASCII decimal digits received are converted from ASCII decimal to actual and placed in register #30. If the source received "1"0"2"4", register #30 would look like this: 0000 0100 0000 0000, which is the actual value 1024.

Example 2: Source to destination other than register (also see picture element i).

```
TR Source(d),dest(iiii)
```

The first ASCII decimal digit received will be converted to actual and then converted back to ASCII Binary digits 1 and 0 accordingly. If the source received "5" it would be converted to 0101 actual and then it would be transferred to the destination as 4 ASCII digits "0"1"0"1".

When used in the destination portion of a transfer command, actual values will be transferred as ASCII decimal digits.

Example 3: Register to destination.

TR #30,dest(dddd)

The actual value in register #30 is converted to ASCII decimal digits and transferred to the destination as such. If #30 has an actual value of 5280, the ASCII decimal digits "5""2""8""0" are transferred to the destination. Note that if register #30 had an actual value of 1 that the ASCII decimal digits "0""0""0""1" would be transferred. This satisfies the requirements for all 4 d picture elements.

Picture element z Zero blanked decimal digit.

The picture element is identical to the d picture element with one exception. When used in the destination portion of a transfer command, all leading zeroes are replaced with spaces (ASCII 20H).

Example 1: Register to destination.

TR #30,dest(zzzz)

If register #30 contains the actual value 520, then it would be converted to ASCII decimal digits and transferred to the destination as " ""5""2""0". If #30 had an actual value of 0, the destination would receive all spaces, " "" "" "" ". Note that the final zero is blanked. You should use the d picture element in the last position so the zero is not blanked, as shown in the next example.

Example 2: Register to destination.

TR #30,dest(zzzd)

If register #30 contains the actual value of 0, the destination would receive " "" "" ""0".

Picture element h Hexadecimal digit.

This picture element should be used whenever only the ASCII hexadecimal digits 0-9, A-F, a-f are wanted.

When used in the source portion of a transfer command, only ASCII hexadecimal digits 0-9, A-F, a-f will be allowed.

Example 1: Source to data register.

TR source(hhhh),#30

The first 4 valid ASCII hexadecimal digits received are converted from ASCII hexadecimal to actual and placed in register #30. If the source received "2""a""3""F", register #30 would look

like this: 0010 1010 0011 1111 which is the value 2A3FH or 10,815.

Example 2: Source to destination other than a register (also see picture element i).

TR source(h),dest(iiii)

The first ASCII hexadecimal digit received will be converted to actual and then converted back to ASCII Binary digits 1 and 0 accordingly. If the source received "C" it would be converted to 1100 actual and then be transferred to the destination as 4 ASCII binary digits "1" "1" "0" "0".

When used in the destination portion of a transfer command, actual values will be transferred as ASCII hexadecimal digits.

Example 3: Register to destination.

TR #30,dest(hhhh)

The actual value in register #30 is converted to ASCII hexadecimal digits and transferred to the destination as such. If #30 has an actual value of 10815, the ASCII hexadecimal digits "2" "A" "3" "F" are transferred to the destination. Note that if register #30 had an actual value of 10 that the ASCII hexadecimal digits "0" "0" "0" "A" would be transferred. This satisfies the requirement for all 4 h picture elements.

Picture element i Binary digit.

This picture element should be used whenever only the ASCII Binary digits 1 and 0 are wanted.

When used in the source portion of a transfer command, only ASCII Binary digits 1 and 0 will be allowed.

Example 1: Source to data register.

TR source(iiii),#30

The first 4 valid ASCII Binary digits received are converted from ASCII Binary to actual and placed in register #30. If the source received "1" "0" "0" "1", register #30 would look like this 0000 0000 0000 1001, which is the value 1001 Binary or 9.

Example 2: Source to destination other than register (also see picture element h).

TR source(iiii),dest(h)

The first 4 ASCII Binary digits received will be converted to actual and then converted back to an ASCII hexadecimal digit accordingly. If the source received "1" "0" "1" "1" it would be converted to 1011 (binary) actual and then be transferred to the destination as an ASCII hexadecimal digit "B".

When used in the destination portion of a transfer command, actual values will be transferred as ASCII binary digits.

Example 3: Register to destination.

TR #30,dest(iiii)

The actual value in register #30 is converted to ASCII binary digits and transferred to the destination as such. If #30 has an actual value of 12, the ASCII binary digits "1"1"0"0" are transferred to the destination. Note that if register #30 had an actual value of 2 that the ASCII binary digits "0"0"1"0" would be transferred. This satisfies the requirement for all 4 i picture elements.

Picture element x Binary byte

This picture element should be used whenever the "raw" actual value of ASCII data is wanted. When used in the source portion of a transfer command, any character is allowed.

Example 1: Source to a data register.

TR source(x),#30

The first character received is placed in the lower byte of register #30.

TR source(xx),#30

The first character received is placed in the lower byte of register #30 and the second character received is placed in the upper byte.

Example 2: Source to destination other than a register (also see picture element d).

TR source(xx),dest(ddddd)

The first character received is the low byte of an actual value and the second character received will be the high byte of the actual value. This actual value is then converted to ASCII decimal digits and transferred to the destination. If the source receives "A" "B" the "A" is placed in the lower byte and "B" is placed in the upper byte to get an actual value that looks like this:

0100 0010	0100 0001
42H	41H
"B"	"A"

This is then converted to ASCII decimal digits "1"6"9"6"1" and transferred to the destination as such.

When used in the destination portion of a transfer command, actual value will be transferred

as 1 or 2 bytes.

Example 3: Register to destination.

```
TR #30,dest(x)
```

The low byte of register #30 is transferred to the destination without modification.

```
TR #30,dest(xx)
```

The low byte of register #30 is transferred to the destination and then the high byte is transferred.

Upper case picture elements

Upper case picture elements operate identically to their lower case counter-parts except that they do not echo data.

When used in the source portion of a transfer command, data will not be echoed.

Example 1: (see picture element d)

```
TR,KB(DD),#30
```

This will allow 2 ASCII decimal digits to be entered, converted to actual and then transferred to register #30. The two digits entered will NOT be echoed to the screen. Note: When the keyboard is the source, the destination for echoed data will always be the screen.

Example 2: (see picture element l)

```
TR,SI(LLLL),SC(IIII)
```

This will allow 4 ASCII characters to be transferred from the serial port to the screen. Because the picture element "L" is used, the data will not be echoed to SO even if that option is enabled in the configuration menu (see Echo Input). Note: When the serial input port is the source, the destination for echoed data will always be the serial output port.

When used in the source portion of a transfer command, data will not be transferred to that device.

Example 3: (see picture element l)

```
TR,SI(LLLL),SO(IIII)
```

This will take the first four characters out of the serial input port and transfer the first 2 of these to the serial output port. If the characters "A""B""C""D" were in the serial input buffer, "A""B" would be sent to the serial output port and "C""D" will be thrown away.

Care should be taken when transferring from the keyboard to the screen since the screen is the keyboard's default device.

Example 4: (see picture element 1)

TR,KB(IILL),SC(ILIL)

The first character will be echoed to the screen, because the source is the keyboard, and is then transferred to the destination, which is also the screen. Therefore, the first character appears twice. The second character will be echoed to the screen and not transferred to the screen. The third character will not be echoed but will be transferred. The fourth character will not be echoed or transferred.

One final exception should be noted. The X picture element will not echo when it is used on the source side of a transfer. When upper case X is used on the destination side of a transfer, it will echo just like x. This means that x and X are interchangeable and identical in function.

Other considerations

Text vs Numeric

Two different types of picture elements have been mentioned, TEXT and NUMERIC. It is important to understand the difference between the two when a transfer command is executed.

When the source and destination are using text picture elements, each character is transferred AS IT IS RECEIVED.

When a numeric picture element is specified, data will not be sent until the transfer condition is satisfied. When the condition is met, data is converted (see numeric picture elements) and is then transferred.

Picture Elements: How Many?

When using text pictures, the only limitation is the 255 character command line limit in OIL. However, you should be aware of the following guidelines when using the numeric picture elements:

d,z picture elements

Since the largest decimal value in 16 bits is 65535, the maximum number of d picture elements required will be 5.

h picture element

Since the largest hexadecimal value in 16 bits is FFFF, the maximum number of h picture elements required will be 4.

i picture element

Since the largest binary value in 16 bits is 1111 1111 1111 1111, the maximum number of d picture elements required will be 16.

x picture element

Since there are two bytes in 16 bits, a maximum of two x picture elements will be required.

If fewer than the maximum number of picture elements are used for any numeric transfer, the least significant portion of that number will be transferred. For example, if register #30 has an actual value of 52397 and TR,#30,SC(ddd) is executed, the ASCII decimal digits "3"9"7" will be transferred.

Data Registers and Picture Elements

Data registers do not require pictures. Keep in mind that a data register is 16 bits and can contain a maximum of 2 bytes of character information.

Other Characters in Picture Elements

Any character which is not a valid picture element can be used as a constant in a picture.

When in the input portion of a transfer command, the character received must match the one specified in the picture element before the transfer can continue.

When used in the output portion of a transfer command the constant will be sent.

Combinations of Picture Elements

- 1) Text and numeric picture elements can not be combined in the same picture.
- 2) Different numeric picture elements can not be combined in the same picture.
- 3) Upper and lower case picture elements can be combined if rules 1 and 2 above are followed.
- 4) Text picture elements can be blended in any manner within a picture.
- 5) Text and numeric picture elements can be combined in different pictures (e.g.: TR,source(text),dest(numeric)).
- 6) Different numeric picture elements can be combined in different pictures (e.g.: TR,source(ddddd),dest(hhhh)).

What OIL does while waiting on a transfer

OIL continually goes through these major operations:

- 1) Execute OIL command
- 2) Execute remote commands (if any)
- 3) House keeping

When a transfer command is executed OIL essentially waits for that command to complete before going on with the other operations it does. This is important to understand because remote commands may be received during a transfer. If that transfer takes long enough, (e.g.: TR,KB(III),SC(LL)), remote commands will pile up in the command buffer. At some point, the command buffer is full. If this condition occurs, data will be lost. This can be avoided by implementing handshaking between the host and the workstation.

A transfer command is complete when one of the pictures is satisfied.

Example 1: TR,KB(AAAN),SC(aaan)

In this example the transfer will not wait for the carriage return because the destination will have received its quota of characters. To keep this from happening, place another picture element in the destination picture.

TR,KB(AAAN),SC(aaaan)

This way a carriage return is required to meet the source picture.

Example 2: TR,KB(aaa),SC(AAAA)

This transfer will terminate after 3 valid characters are entered because the source has received its quota of characters.

The key point to remember is that when one of the pictures is satisfied, the transfer will end.

4.6.59 Transfer Text (TS)

Syntax: TS source,reg

where: source is the data source for the transfer; only keyboard pictures are valid

reg is the first of consecutive registers the data is transferred to

This command transfers text from the keyboard to consecutive registers. To display the text, use the Display String command (DS). Remember that each register will hold up to two characters. The string is stored in the same manner as the Store String command (SS).

a,l,c, or n pictures may be used; descriptions are listed with the Transfer Data command (TR), Section 4.6.58.

Example: TS,KB(LLLLL) #55 ;transfer text to register
DS #55 ;display string

The above program will transfer the five characters input from the keyboard to consecutive registers 55, 56, and 57. When the program is run, the text string will appear on the screen.

4.6.60 Transfer Touch Screen Data (TT) (Touch Screen Only)

Syntax: TT reg

where: reg is the register to transfer the zone number to

This command transfers the number of the currently pressed touch screen zone (1 - 80), or the value selected for that zone via the DFZ command, to a specified register. The released state is zero.

Example: TT #35 ;transfer touch screen data to register

The above program will transfer the number of the currently pressed touch screen zone to register #35. For example, if zone 1 (upper left hand corner of touch screen) is pressed, that zone number will be transferred to register #35.

4.6.61 Up (U)

Syntax: U[num]

where: num is the number of positions the cursor will be moved.

This command moves the cursor the number of positions specified in the argument. If the argument is omitted, the cursor is moved one position. If the cursor hits the top of the screen, the command is ignored.

NOTE

The cursor will stop when it reaches the top of the screen.

Example: U 20

This moves the cursor up 20 spaces or to the top of the screen, whichever is less.

4.6.62 UNPLOT

Syntax: UNPLOT xpoint,ypoint

where: xpoint is the column coordinate of the point (0-159)

ypoint is the row coordinate of the point (0-71)

This command removes a plotted point from the screen.

NOTE

The 0,0 point is the lower left corner of the scrolled area, so the maximum y value is dependent on the number of status lines chosen in the configuration menu.

Example: UNPLOT 0,0

This example removes the previously plotted point from the lower left hand corner of the screen.

4.6.63 Vertical Bar Up (VB or VBU)

Syntax: VBU xstart,ystart,hght,maxhght
 or
 VB xstart,ystart,hght,maxhght

where: xstart and ystart are the column and row coordinates of the bottom of the bar

 hght is the height of the bar, in units of a character cell*

 maxhght is the maximum height of the bar, in units of a character cell,
 which will be erased before a bar of new height is drawn:

Draws a bar of height hght upward from (xstart,ystart). If either coordinate is absent, the current cursor coordinates will be used. Before the new bar is drawn, a bar of height maxhght is erased. The parameters hght and maxhght must specify the height in units of a character cell. For example, to draw a bar 6 1/2 character cells high on a monochrome workstation, hght would be 78. The parameter maxhght specifies how high a bar to erase before the bar of hght is drawn. Therefore, maxhght should be no less than hght.

Example 1: VBU 10,20,120,180

This draws a vertical bar 120 units high after erasing an area 180 units high.

Example 2: VBU #40,#41,20,240

This draws a vertical bar whose column and row coordinates are found in registers #40 and #41 respectively; the height of the bar is 20 characters.

*Each character cell is 12 units high (1 unit = 1/12 character cell). The maximum size for hght and maxhght is 255.

4.6.64 Vertical Bar Down (VBD)

Syntax: VBD xstart,ystart,hght,maxhght

where: xstart and ystart are the column and row coordinates of the top of the bar

hght is the height of the bar, in units of a character cell*

maxhght is the maximum height of the bar, in units of a character cell, which will be erased before a bar of new-depth is drawn:

This command functions just like the Vertical Bar Up command, except that the bar extends downward from the (xstart,ystart) coordinate.

Example: VBD 1,1,120,121

This draws a vertical bar down 120 units after erasing an area 121 units.

*Each character cell is 12 units high (1 unit = 1/12 character cell). The maximum size for hght and maxhght is 255.

4.6.65 Draw Vertical Line (VL)

Syntax: VL lval,xstart,ystart,vlength

where: lval can have the following arguments:
1 = thick line
2 = thin line
3 = GR3 character "
4 = GR3 character \$
5 = GR3 character *
6 = GR3 character ,
any other character (enclosed in double quotes, e.g. "*") or
number corresponding to a character = a line composed of
that character

xstart and ystart are the column and row coordinates of the bottom of the line

vlength is the number of characters the line extends upwards

Draws a line of length vlength upward from (xstart, ystart). (Note that x indicates the column, y the row.) The lval parameter specifies what the line will be composed of.

Example 1: VL 1,10,20,15

This draws a thick line 15 characters high whose bottom is at column 10, row 20.

Example 2: VL 2,#40,#41,#20

This draws a thin line with bottom coordinates found in registers #40 (column) and #41 (row), and with height in register #20.

Example 3: VL "s",10,20,10

This draws a line composed of s's.

Example 4: VL 1,,12

This draws a line whose bottom coordinates are at the current cursor position.

4.6.66 XOR

Syntax: XOR #n,m

where: #n is a register (12 - maximum register)
m is a numeric value

An exclusive OR performs a bit-wise logical XOR of the data in the register and the numeric value. The result is stored in the specified register.

This command can be used to "flip" particular data bits in a register.

Example: XOR #20,31

This XORs the contents of register #20 with the number 31 and places the result in register #20.

4.7 SAMPLE PROGRAM

This section shows how to create a simple screen display on a workstation. To create this display, it is not required that the workstation be connected to a host or a secondary device. When the workstation is plugged into a standard power socket it is ready to implement the instructions presented in this chapter.

NOTE

A program can only be typed into the workstation if the remote keyboard is attached to the workstation. Programs cannot be entered on the workstation keypad. Remote keyboards are available separately from XYCOM.

There are some terms which will make it easier to understand what is happening when typing a program into the workstation. A screen display is generated whenever a program stored in a program block is executed. There are 255 program blocks on the workstation, and each of them can store a separate program. When the 2000-10 option is first installed, all the program blocks are empty. This chapter will illustrate how to type a program into one of the program blocks.

A program consists of a sequence of instructions written in Operator Interface Language (OIL). Each command in OIL will give the workstation specific instructions concerning exactly what characters or symbols to display on the screen and where on the screen these characters will be displayed. For example, the cursor can be moved anywhere on the screen and write a word or a figure such as a box at that position. The size of the characters (single, double, or quad-size) and their attributes (blinking, underline, etc.) can be selected. Programs can also perform other functions, such as reading data or commands from any device, writing data to a device, or reading and writing to a serial port.

Entering a program into a program block does not create a screen display. The program will remain in the workstation's battery-backed memory for an indefinite period of time (until it is replaced by a different program typed into the same block). The screen display specified by the program is created only when the program is executed. It will remain on the screen until another program block is executed or until the Set-up mode is called up to reconfigure the workstation.

This sample program (Figure 4-23) illustrates how easy it is to create a screen display on the workstation. However, it reveals only a tiny fraction of all the capabilities of the workstation. This program is ready to be typed into a program block (see Chapter 3 for use of the workstation's editor).

Sample Program

```
;VESSEL #3 OVERTEMP
CS                      ;clear screen
BX 2,15,0,62,7         ;draw outer box
BX 2,17,1,60,6         ;draw inner box
SE QS,RV               ;set for quad-size, reverse video
@19,2                  ;position the cursor
"WARNING"
SE DS                  ;set for double-size characters
RE RV                  ;reset attributes
@15,9                  ;position the cursor
"VESSEL No. 3 HAS EXCEEDED"
@11,11                 ;position the cursor
"MAXIMUM TEMPERATURE SETPOINT"
@7,16                  ;position the cursor
"CLOSE STEAM VALVE No. 12 MANUALLY"
@11,21                 ;position the cursor
"CONTACT FOREMAN - PAGER #763"
HL 2,15,14,47          ;draw a horizontal line
HL 2,15,19,47          ;draw a horizontal line
```

Screen Display

The sample program above will create the screen display shown in Figure 4-2.

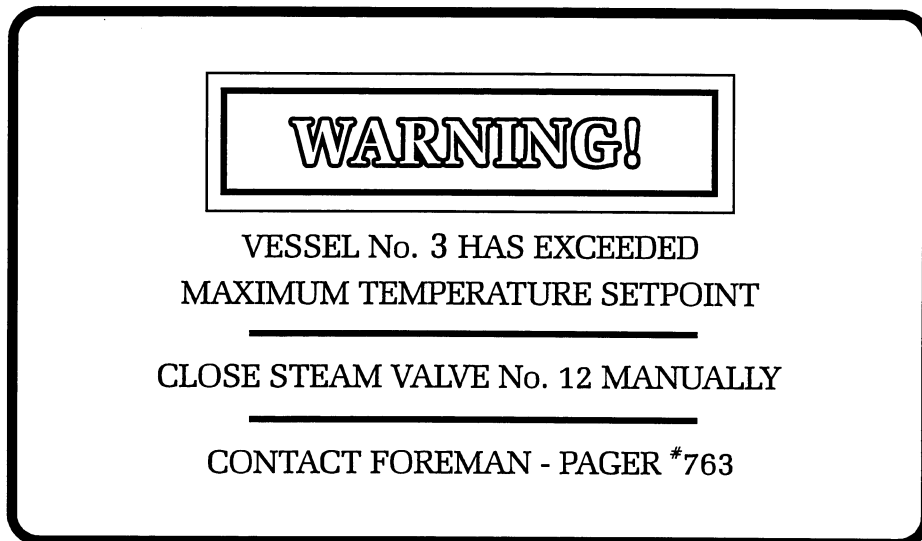


Figure 4-2. Screen Created By Sample Program

In any line, everything following a semicolon is regarded by the workstation as a comment and ignored when the program block is executed. The comments do not have to begin at any particular space on a line, so the exact number of spaces between the end of a command and the semicolon is unimportant (although they do use up program memory). Nor do the comments have to line up vertically.

All the individual commands contained in the program are described in detail earlier in this chapter.

4.7.1 Executing the Program Block

The program can be tested by executing it. The program block being displayed on the screen will execute by pressing the <F6> key. If the appearance of the screen matches that of Figure 4-2, the program has been correctly entered. If any commands were incorrectly typed, the workstation will display an error message. The line on which the first error occurs will be near the top of the screen. To display the entire program block again, move the cursor upward to scroll the screen.

If an error message is displayed on the screen, check the program against Figure 4-2. Press any key to be returned to program block 1 and edit the program. Insert or delete the appropriate characters to correct the error. Execute the program again to see if the problem has been corrected.

A program block can also be executed directly from the "Program Utilities" menu. Instead of selecting item 1, "Edit", specify item 2, "Execute". A prompt will request the number of the program block to be executed:

Execute what program block number? (1-255)

Press the "1" key and then <Enter>. Program block #1 (the sample program) will be executed.

5.1 INTRODUCTION

This chapter provides detailed examples of how the OIL commands can be used together in typical workstation applications. The following applications are described in this chapter.

- Setting and reading the time of day
- Reading data from the keyboard or serial port
- Nesting screens
- Adding/subtracting and incrementing/decrementing data registers

Tips on increasing the speed of OIL programs are also given.

5.2 DATA REGISTERS

Data registers are 16-bit memory locations which are used for communication between a device and the workstation. Both the device and the workstation can read/write to data registers. There are at least 500 such registers on the workstation, addressed as #1 through #500. Up to 9999 additional registers may be added via the configuration menu (addresses #501 to #10,499).

The workstation can perform mathematical operations on values stored in data registers. It can also use data register values as parameters for most OIL commands (e.g., specifying the height of a bar graph). Registers can hold values or they can contain pointers to other registers (indirect addressing).

Registers #1 through #11 are special-purpose registers. Registers #1 through #7 contain the time and date, and are automatically updated by the workstation. Registers #8 through #11 hold the number of characters currently in the keyboard queue, serial port input queue, and touch screen queue. Registers #1 through #11 are read-only registers. Transferring data to them will not change them.

Indirect Addressing. When a register number preceded by # is used in an OIL command, the value stored in the register is used as an argument in the command. For example, suppose register 68 contains the value "15". Then the command

EX #68

will cause the program block whose number is contained in register 68, namely program block 15, to be executed. However, if the register number is preceded not by # but by \$, indirect register addressing is used.

In indirect register addressing, the value in the specified register is not regarded as the value to be used as an argument in the command, but is interpreted as a pointer to another register which stores the value. For example, suppose register 68 contains the value "15" as before, and register 15 contains the value "80". Then the command

EX \$68

will cause program block 80 to be executed. (The value "15" in register 68 is regarded as a pointer to register 15, which contains the value to be used in the EX command.)

Example 1:

If #20 = 100
#100 = 123

After executing the OIL instruction:

PLOT,\$20,#20

a point would be plotted at plot location 123,100.

Example 2:

If #20 = 100
#100 = 123

After executing the OIL instruction:

INC,\$20

register 100 would contain the number "124".

5.3 READING THE TIME AND DATE

Data Registers #1 through #7 are reserved for the time and date, limited to the range January 1, 1950 to December 31, 2049.

#1 -- Year	(holds last two digits of the year, i.e., 87 for 1987)
#2 -- Month	(1-12)
#3 -- Date	(1-31)
#4 -- Hour	(0-23)
#5 -- Minute	(0-59)
#6 -- Second	(0-59)
#7 -- Day of Week	(0=Sunday, 1=Monday, ...)

Registers #1 through #6 are updated every second to provide the current time and date. They can be read at any time and their values displayed on the screen or transferred to another data register or output ports. They can only be changed through the Configuration Menu (or the remote command that loads them -- see Chapter 6). Register #7 is calculated and set automatically by the clock/calendar. Unlike registers #12 through #500, these registers cannot be written to with the OIL Transfer command.

Example Program 1:

The following short program displays the hour, minute, and seconds as they are being incremented (for a total of 10 seconds).

```
;clock, with day and date names
re cu      ;to prevent cursor movement between display fields
se qs      ;quad size
tr #6,#190:ADD 10,#190  ;Reg. #190 holds upper bound for loop
MOD #190,60
%Loop
TR #4,#191  ;read hours
TR "AM",#192
IF #191>11: TR "PM",#192  ;Set AM or PM
IF #191>12:SUB 12,#191    ;Convert military time to 12 hour time
@14,15:TR #191,SC(dd):":":TR #5,SC(dd):":":TR #6,sc(dd): " ":TR #192,SC(11)
IF #6<>#190:GO Loop
```

Example Program 2:

This program displays the day, date, and time as they are being updated.

```
;DAY, DATE, AND TIME
re,cu           ;Turn off cursor
SE DH DW       ;Set double-high, double-wide
%loop          ;Label
@12,2          ;Position cursor
if,#7=0:"Sunday   ":go,bottom ;Test for day of week
if,#7=1:"Monday   ":go,bottom
if,#7=2:"Tuesday  ":go,bottom
if,#7=3:"Wednesday":go,bottom
if,#7=4:"Thursday ":go,bottom
if,#7=5:"Friday   ":go,bottom
if,#7=6:"Saturday ":go,bottom
%bottom
tr,#2,sc(dd):"/" ;month
tr,#3,sc(dd):"/" ;day
tr,#1,sc(dd):"  " ;year
tr,#4,sc(dd):":" ;hour
tr#5,sc(dd):":" ;minute
tr#6,sc(dd)      ;second
if,#8=0:go loop  ;repeat until key pressed
```

5.4 READING DATA FROM THE KEYBOARD OR SERIAL PORT

The keyboard and serial port can all be sources in a transfer command. Some examples:

```
TR KB(a),#80      ; ASCII value of character typed is put in Reg. #80
TR SI(aaaa),SC(aaaa) ; Four ASCII characters read from serial port and
                   ; displayed on the screen
```

The following sections describe how data from the keyboard/keypad and serial port is handled.

5.4.1 Input Data Queues

The workstation provides four queues, the Keyboard Input Queue, the two Serial Port Queues, and the Touch Screen Queue. While the workstation is executing a program block, it is constantly monitoring the keyboard, keypad, touch screen, and the serial port for incoming remote commands. If any data is received, it is put into the corresponding queue.

Each queue is a first-in, first-out (FIFO) buffer. Each key pressed on either the keypad or keyboard is placed in the keyboard (KB) input queue; each touch screen zone pressed is put into the touch screen buffer. Everything that is received over the serial port is first examined by the workstation to see if it could be a remote command. If it has the correct command format, a character sequence is placed in the remote command queue, where it will wait until the workstation completes the current OIL command, and is then executed (possibly after other remote commands ahead of it in the buffer). If it does not appear to be a remote command, each byte of data is placed in the appropriate input queue. If a command in the remote command queue is, upon execution, found to be illegal (such as trying to access too large a register number), that command is ignored. Illegal commands are not placed into any data input queue.

Data from the queues is read and removed from the queue when the appropriate Transfer command is executed (commands from the command queue are executed when the current OIL command is finished). For example, If, while a program block was being executed, you typed the three characters "ABC" on the keyboard. These three characters would be placed in the 16-character Keyboard Input Queue (see Figure 5-1).

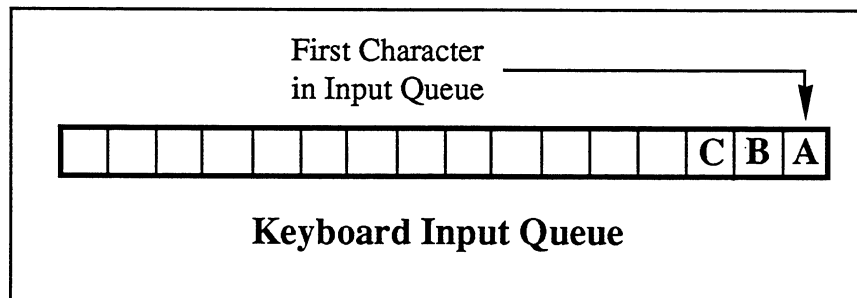


Figure 5-1. Keyboard Input Queue Example (1)

The next time that a Transfer command was executed to read the keyboard, what the command would actually read is the first character in the Keyboard Input Queue. For example, suppose the following command was executed:

```
TR KB(aa),SC(aa) ;read two chars. from keyboard and display on screen
```

The first two characters from the Keyboard Input Queue, namely "A" and "B", would be read from the queue and thereby removed from it. The screen would display the characters "AB", and the Keyboard Input Queue would now look like this:

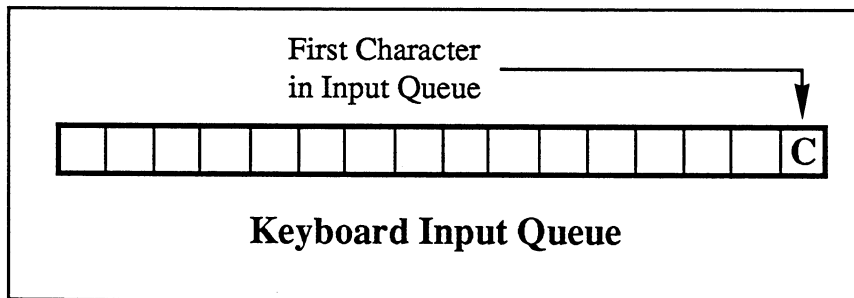


Figure 5-2. Keyboard Input Queue Example (2)

5.4.2 Queue Status Registers

Associated with each input queue is a dedicated data register that contains the number of characters in that input queue. For example, if the Keyboard Input Queue is empty, Register #8 contains the value 0. If the Keyboard Input Queue received the characters "ABC" in that order, Register #8 would contain the value 3 (see Figure 5-1).

Table 5-1. Registers #8 - #11

Queue Status Register	Associated Queue
#8	Keyboard Input Queue
#9	Secondary Input Data Queue
#10	Primary Input Data Queue
#11	Touch Screen Queue

The main use for Queue Status Registers is to check the "readiness" of an input. In the case of the keyboard, its Queue Status Register (#8) will indicate if any keys on the keyboard have been pressed. If the value of register #8 is 0, no keys have been pressed; if non-zero, there are characters in the queue. For example, if the following Transfer command were issued:

```
TR KB(a),#100
```

and no key had been pressed, the TR command would wait (possibly forever) until a key was pressed before being executed. Register #8 provides a way of checking for keyboard input without having to wait for a key to be pressed:

```
IF #8<>0: ES 20 ;If #8 does not equal 0 (i.e., a key was pressed), execute
;subprogram
```

You should check Register #8 before using the Transfer command to read keyboard data.

Each time a character is read from a queue, the associated Queue Status Register is decremented by 1. It is not set to zero until there are no more characters left in the queue.

The following program asks whether or not to display the temperature. Program 50, which displays the temperature, is executed only if "Y" is typed. While it is waiting for the operator to press a key, it keeps doing other useful work. Since the pressed key is not echoed, invalid keystrokes do not show on the display.

```
"Display Temperature? (Y or N): "  
%Loop1  
IF #8=0: ES 40: go,Loop1 ;Update display while waiting for key  
TR KB(x),#20  
IF #20="Y": EX 50 ;"Y" = yes  
IF #20="N": EXIT ;"N" = no  
GO Loop1
```

5.4.3 Dummy Destination NO

The dummy destination NO is provided to let you remove data from a queue (and therefore the associated Queue Status Register) without outputting the data to a particular data register, serial port, or the screen. The following example will empty the Keyboard Input Buffer:

```
%mt:if #8>0:TR KB(x),NO(x):Go,mt ; = CB, KB
```

5.4.4 Menu Program

The following will display a menu on the screen, and depending on which key is pressed, execute a specific subprogram. A program like this can be used to generate any menu required.

Program Block 25

```
;demo menu execution  
cs:se ds:re cu  
@10,: "Select which program to execute:"  
@16,4:"1. Program 1"  
@16,6:"2. Program 2"  
@16,8:"3. Program 3"  
se cu  
CB,KB  
%wait ;wait for keystroke  
tr kb(x),#30  
if #30="1":es 27:exit  
if #30="2":es 28:exit  
if #30="3":es 29:exit  
go wait ;ignore other characters
```

Program Block 27

```
@16,10:"Program 1 would be executed here"
```

Program Block 28

```
@16,10:"Program 2 would be executed here"
```

Program Block 29

```
@16,10:"Program 3 would be executed here"
```

5.4.5 I/O to the Serial Port

The following example reads the temperature from the secondary port, displays it, and writes the temperature to the control system. In addition, if the temperature exceeds 500, the workstation will execute program 100, which displays a warning, then transfers the character "T" and temperature to the control system via the secondary port.

Program Block 90

```
;Read temp from SI, display, and write to SO
CS:RE CU
%LOOP
@10,10
SE DS
"TEMPERATURE: "
SE BL
TR SI(dddd),#50
TR,#50,SC(dddd)
IF #50>500:ES 100
GO LOOP
```

Program Block 100

```
@20,20
SE QS BL
"WARNING: TEMPERATURE TOO HIGH"
TR "T" SO(a)
TR #50 SO(ddddd)
```

5.5 NESTED SCREENS

The block command Execute Subprogram Block (ES) can be included within a program block. For example, suppose Program Block mm contains an Execute Subprogram Block command to execute block nn. When Program Block mm is executed, all the block commands will be executed in the order in which they occur. When the Execute Subprogram Block command is executed, control will be transferred to program block nn, which will be executed in its entirety. When Screen nn is finished, control will be returned to program block mm, and execution of subsequent block commands in Screen mm will continue.

Figure 5-3 illustrates nested program blocks.

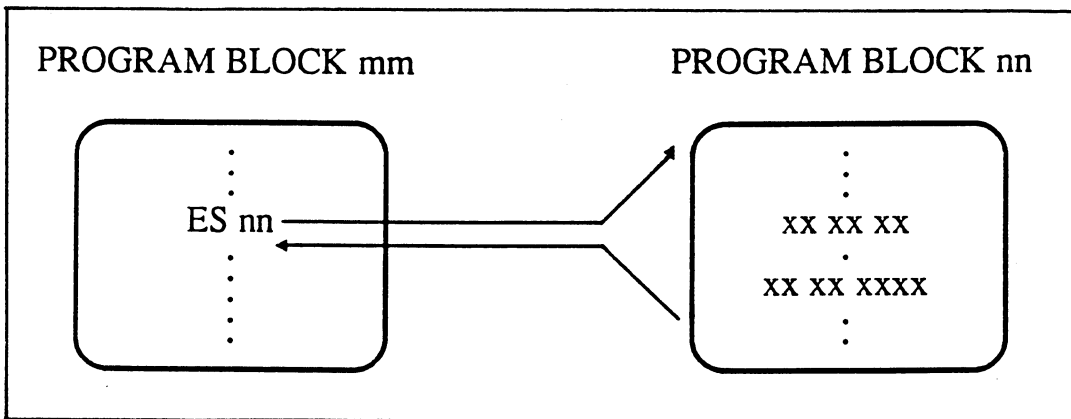


Figure 5-3. Nested Blocks

Any number of Execute Program Block commands can be included in a program block, and may be included anywhere within the block. Up to ten levels of nesting are permitted for the ES command. A program block may also call itself, however the ten level limit must be obeyed.

Program blocks are especially useful in designing complicated displays in which some portions of the display vary while others remain the same. The constant areas of a display can be put in one program block, while each of the possible variations may be placed in separate program blocks. These variations can be called from other program blocks as needed. Then instead of redesigning each program block from scratch, you may design a program block in parts and reuse these parts in other program blocks.

A simple example: An application is monitoring a vat for two alarm conditions, "pressure too low" and "temperature too high". When it detects either condition, the application should flash "WARNING" at the top of the screen. However, it would also be desirable to identify the type of danger condition by displaying the phrase "Temperature Too High" or "Pressure Too Low" following the "WARNING" message. One way of designing the screens for this application is to use three separate program blocks (see Figure 5-4).

Program block 24: Creates the display "WARNING"
Program block 20: Creates the display "TEMPERATURE TOO HIGH"
Program block 21: Creates the display "PRESSURE TOO LOW"

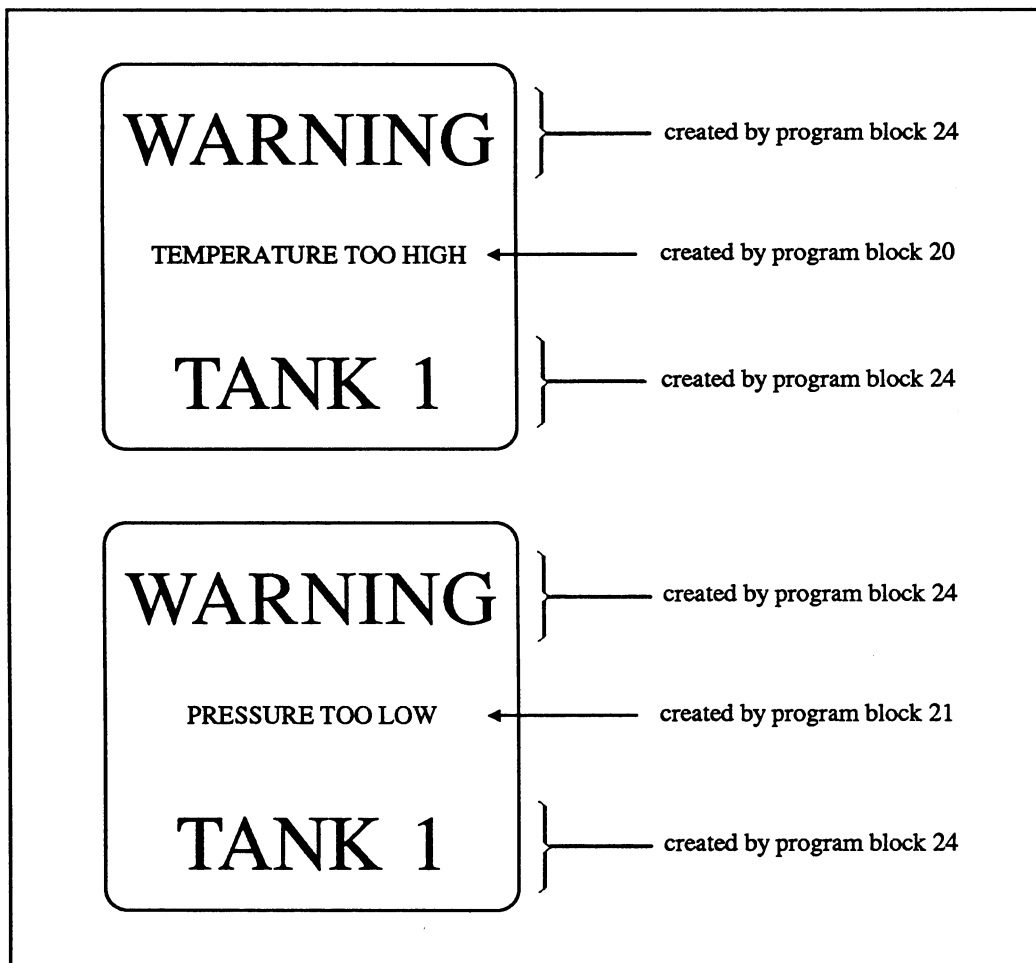


Figure 5-4. Nested Screen Example

To display the "TEMPERATURE TOO HIGH" warning, program block 7 would be executed. To display the "PRESSURE TOO LOW" warning, program block 8 would be executed. Program block 7 contains an Execute Subprogram Block command to execute subprogram blocks 24 and 20, while program block 8 executes subprogram blocks 24 and 21.

The control system selects which program block will be executed (7 or 8) by writing either 7 or 8 to Register #50. Program block 1 uses the value of #50 to branch to either block 7 or block 8.

Program Block 1

TR SI(x),#50
ES #50

Program Block 7 (For "TEMPERATURE TOO HIGH" Warning on Tank 1)

ES,24 ;Execute Program Block 24
ES,20 ;Execute Program Block 20

Program Block 8 (For "PRESSURE TOO LOW" Warning on Tank 1)

ES,24 ;Execute Program Block 24
ES,21 ;Execute Program Block 21

Program Block 24

@10,1 ;Position command
SE BL DS ;Set Attributes to blinking, double-size
"WARNING"
@12,17 ;Position command
RE BL ;Turn off blinking
"TANK 1"
@10,10 ;Position command
RE DS ;Reset attributes to default values

Program Block 20

"TEMPERATURE TOO HIGH"

Program Block 21

"PRESSURE TOO LOW"

5.6 INCREMENTING DATA REGISTERS

The following example uses the increment command within a loop to execute consecutive subprogram blocks.

```
;Display Zones' Status on a Round-Robin Cycle
%DZLOOP
TR 41,#40 ;Use Register 40 for the Subprogram Block Number
%NEXTZONE
ES #40: PS 35 ;Display Zone Status and pause 3.5 seconds
INC #40: IF #40<46: GO NEXTZONE ;Increment Block # and do next one
GO DZLOOP ;Repeat the whole display
```

5.7 KEYBOARD TRANSLATION

The following short program demonstrates how keys pressed on the keypad can be translated into other keys or even sequences of keys.

```
;key translation
se qs
%loop
tr kb(x),#100
if #100="A":"Run":NL
if #100="B":"Stop":NL
go loop
```

5.8 PERFORMANCE HINTS

Commands execute faster if the parameters are separated by commas rather than blanks.

Incrementing/decrementing a data register is faster than adding/subtracting one to/from the register.

Shorter label names are found more quickly than long names.

Construct loops to minimize the number of commands within the loops. For example, if the display attributes do not change within a label/GO loop, then you should execute any Set Attribute command before starting the loop rather than within it.

The workstation always searches for labels starting at the beginning of the program block. The farther a label is from the beginning of the block, the longer it will take to execute a GO to that label.

Construct decision-making sequences to minimize GOing to labels. For example, this:

```
;check for Pressure Value between 837 and 1042  
IF #100<=837:GO SHUTDOWN  
IF #100>=1042:GO SHUTDOWN  
;Pressure is OK
```

rarely executes the GO commands, and so executes faster than this:

```
;check for Pressure Value between 837 and 1042  
IF #100>837:GO CHECKHI  
GO SHUTDOWN  
%CHECKHI:IF #100<1042:GO OK  
GO SHUTDOWN  
%OK ;Pressure is OK
```


6.1 INTRODUCTION

A host computer can control the display on the 2000 Industrial Workstation by sending a set of remote commands over the workstation serial port. There are only a few remote commands:

- Reset
- Execute Program Block
- Execute Subprogram Block
- Transfer from Data Register
- Transfer to Data Register
- Load Time and Date
- Load All Registers
- Send Password
- Receive Program Block (Serial Port only)
- Send a Number of Registers
- Receive a Number of Registers
- Receive Extra Registers
- Backup Programs
- Verify Program

The interface protocol to your control system can be kept simple, because the workstation handles complex operations by using OIL commands.

Note, however, that a host computer cannot send OIL commands to be executed directly by the workstation. OIL commands can be entered into a program block either by typing them at the workstation or by downloading them over the serial port. All of the commands which create the displays, such as Draw Box or cursor movements, are command lines within a program block. Chapter 4 describes all the available program commands.

Each of these remote commands is executed "between" the execution of OIL program commands. Before executing any OIL commands, the workstation checks to see if any remote commands have been received, and executes these first. After executing all pending remote commands, the next OIL command is executed. Once execution of an OIL command has begun, all remote commands received are put in a queue, and will be executed only after execution of the current OIL command has been completed. (An exception is the remote command Reset, which will reset the workstation immediately.)

A host computer can control workstation displays in these three ways:

- 1) By initiating the execution of an already existing program block (Execute Program Block command) or subprogram block (Execute Sub-program Block).
- 2) Transferring data directly to the workstation data registers. The workstation can then read this data with the Transfer command and use it in the screen display. For example, a program block command such as NL,#32 will move the cursor down by the number of lines specified in data register #32. By writing values in data register #32, a host can indirectly control the screen display. Similarly, a program block can display the contents of any register.
- 3) Transmitting data to the workstation serial input port. The workstation can read this data with the Transfer command and use it just like data written in a data register.

6.2 SERIAL REMOTE COMMANDS

Table 6-1 lists all the available serial remote commands.

Table 6-1. Serial Remote Commands

COMMAND	FORMAT
Reset	<ESC>[0p
Execute Program Block	<ESC>[12;<block#>p
Transfer To Data Register	<ESC>[30;<reg#>;<data>p
Transfer From Data Register	<ESC>[31;<reg#>p
Load Time and Date	<ESC>[32;YY;MM;DD;hh:mm:ssp
Load All Registers	<ESC>[14;000p<#11val><#12val>...<#500val>
Send Password	<ESC>[20p
Execute Subprogram Block	<ESC>[33;<block#>p
Receive Program Block	<ESC>[14;<block#>p<text>
Send a Number of Registers	<ESC>[35;REG;NUMp
Receive a Number of Registers	<ESC>[34;REG;NUMp<MSB><LSB><LSB>...
Receive Extra Registers	<ESC>[14;000rxxxx<#501val><#502val>...
Backup Programs	<ESC>[36;PROG;CONp
Verify Program	<ESC>[37;<block#>p<text>

KEY:

<block#>	a string of up to three ASCII decimal digits specifying a block in the range 1 through 255 (e.g., "25" for program block #25)
<reg#>	a string of up to three ASCII decimal digits specifying a data register in the range 1 through maximum register (e.g., "25" for register #25)
<data>	a string of up to five ASCII decimal digits specifying a value in the range 0 through 65535 (e.g., the three digits "1", "2", "3" for the value 123)
#nval	two bytes (MSB,LSB) specifying the value to be written in data register #n (max. value 65635)
	the ASCII delete character (7FH)
<text>	the text of the program block
CON	number of consecutive programs to return
PROG	beginning program number for the workstation to return
REG	beginning register number
NUM	the number of registers in ASCII
MSB	Most Significant Byte
LSB	Least Significant Byte

6.2.1 Reset

Description: This command causes the workstation to initialize itself. All data input and output queues are cleared, and the start-up program block (if non-zero) is executed.

Syntax: <ESC>[0p

where: <ESC> = 1BH
[= 5BH
0 = 30H (the digit 0)
p = 70H

The Reset command will be performed whenever it is received over the serial port, even when there are other commands ahead of it in the queue. However, it will not be performed if the command queue is full.

6.2.2 Execute Program

Description: This command causes the workstation to terminate any program currently executing, and to begin executing the specified program.

Syntax: <ESC>[12;dddp

where: <ESC> = 1BH
[= 5BH
1 = 31H
2 = 32H
; = 3BH
ddd = string of up to 3 ASCII decimal digits specifying a program block in the range 1 through 255 (e.g., "25" for program block #25).
p = 70H

6.2.3 Transfer To Data Register

Description: This command causes the specified value to be placed into the specified data register.

Syntax: <ESC>[30;ddd;nnnnnp

where:

- <ESC> = 1BH
- [= 5BH
- 3 = 33H
- 0 = 30H
- ; = 3BH
- ddd = string of up to 3 ASCII decimal digits specifying a data register in the range 11 through maximum register (e.g., "40" for data register #40).
- nnnnn = a string of up to five ASCII decimal digits specifying a value in the range 0 through 65535.
- p = 70H

6.2.4 Transfer From Data Register

Description This command causes the workstation to return a message via the serial port indicating the current value of the specified register. The response is terminated by a carriage return to simplify programming in a high-level language (such as BASIC) in the host computer or programmable controller system.

Syntax: <ESC>[31;dddp

where: <ESC> = 1BH
[= 5BH
3 = 33H
1 = 31H
; = 3BH
ddd = string of up to 3 ASCII decimal digits specifying a data register in the range 1 through maximum register (e.g., "30" for data register #30).
p = 70H

Returns:

In response to this command, the workstation will transmit the following:

<ESC>[nnnnnr<CR>

where: r = 72H
nnnnn = current value of the specified data register
<CR> = 0DH

The return format does not pad with spaces or zeroes to 5 digits. The return will not always return 5 characters.

6.2.5 Load Time and Date

Description: This command sends a time and date to the workstation to set the time-of-day clock. (The clock can also be set from the keyboard through the Real Time Clock Configuration Menu.)

Syntax: <ESC>[32;YY;MM;DD;hh;mm;ssp

where: <ESC> = 1BH
[= 5BH
3 = 33H
2 = 32H
; = 3BH
YY = two ASCII digits specifying the year; for example, "8" and "6" (i.e., 38 35) for 1986.
MM = two ASCII digits specifying the month (01 through 12).
DD = two ASCII digits specifying the day of the month (01 through 31).
hh = two ASCII digits specifying the hour (00 through 23).
mm = two ASCII digits specifying the minute (00 through 59).
ss = two ASCII digits specifying the second (00 through 59).
p = 70H

6.2.6 Load All Registers

Description This command causes values specified in the command to be written to successive registers, beginning with register #11.

Syntax: <ESC>[14;000p<reg #11 value><reg #12 value>...

where: <ESC> = 1BH
[= 5BH
; = 3BH
1 = 31H
4 = 34H
p = 70H
<reg #n value> = <low-byte><high-byte>
 = 7FH

The command must contain all 490 register values (from #11 through maximum register). For example, suppose the command contained the following two bytes as its first register value:

00 (NUL)
01 (SOH)

The value 256 (00000001 00000000) would then be stored in register 11.

6.2.7 Send Password

Description This command transmits the currently active password out the serial port. The password is set through the Set Password menu.

Syntax: <ESC>[20p

where: <ESC> = 1BH
[= 5BH
2 = 32H
0 = 30H
p = 70H

Returns:

In response to this command, the workstation will transmit the following:

<pas><CR>

where: <pas> is the currently active password (up to 3 alphanumeric characters)
<CR> = 0DH

If no password is currently active, just <CR> will be returned by the workstation.

6.2.8 Execute Subprogram Block

Description This command causes the workstation to suspend any program currently executing, execute the specified program block, and resume execution where it was suspended.

Syntax: <ESC>[33;<block#>p

where: <ESC> = 1BH
[= 5BH
3 = 33H
; = 3BH
<block#> = string of up to 3 ASCII decimal digits specifying a program block in the range 1 through 255
p = 70H

6.2.9 Receive Program Block

Description This command causes the workstation to accept and store a screen program which is imbedded within the command. The screen must be terminated by a DEL character (7FH).

This command also restores programs to the workstation.

Syntax: <ESC>[14;<block#>p<text of stored screen>

where: <ESC> = 1BH
[= 5BH
1 = 31H
4 = 34H
; = 3BH
<block#> = string of up to 3 ASCII decimal digits specifying a program block in the range 1 through 255
p = 70H
<text of stored screen> = text of stored screen as ASCII characters
 = 7FH

Programs that are being restored could potentially overwrite currently executing programs. To eliminate this problem, when the remote command is received, the screen clears, "Restore In Process" appears on the screen, and the workstation goes into operational mode. After the programs are restored no OIL programs will be executing. Things will begin running again once an Execute Screen command is transmitted. The Clear Screen (CS) command should be part of the screen being executed; this will get rid of the "Restore in Process" message.

6.2.10 Send a Number of Registers

Description This command generates a Receive a Number of Registers command.

Syntax: <ESC>[35;REG;NUMp

where: <ESC> = 1BH
[= 5BH
3 = 33H
5 = 35H
; = 3BH
REG = is the beginning register number
NUM = is the number of registers (ASCII)

6.2.11 Receive a Number of Registers

NOTE
This remote command should not be used with the XON/XOFF software handshaking, as the MSB and LSB data may equal the ASCII value of the XON/XOFF characters.

Description This command takes a specified number of 16-bit values and places them consecutively, starting at a specified register.

Syntax: <ESC>[34;REG;NUMp<MSB><LSB><MSB><LSB>...

where: <ESC> = 1BH
 [= 5BH
 3 = 33H
 4 = 34H
 ; = 3BH
 REG = is the beginning register number
 ; = 3BH
 NUM = is the number of registers in ASCII decimal representation
 MSB = a binary byte
 LSB = a binary byte
 = 7FH

6.2.12 Receive Extra Registers

Description This command writes specified values to a chosen number of consecutive extra data registers beginning with register 501.

Syntax: <ESC>[14;000rxxxx<#501val><#502val>...

where: <ESC> = 1BH
 [= 5BH
 1 = 31H
 4 = 34H
 ; = 3BH
 0 = 30H
 r = 72H
 xxxx = number of extra registers to receive (ASCII 0001 - 9999)
 <#501val> = first register to receive values
 = 7FH

6.2.13 Backup Programs

Description This command backs up a specified number of consecutive programs.

Syntax: <ESC>[36;PROG;CONp

where: <ESC> = 1BH
[= 5BH
3 = 33H
6 = 36H
; = 3BH
PROG = number of the beginning program for the workstation to return (in ASCII)
CON = number of consecutive programs to return

Returns:

In response to this command, the workstation will transmit the following:

```
<ESC>[14;ddd<text of stored screen PROG><DEL>
<ESC>[14;ddd+1p<text of store screen PROG+1><DEL>
.
.
.
<ESC>[14;ddd+CONp<text of stored screen (PROG+(NUM-1))<DEL>
<ESC>[14;0q
```

where: ddd = string of up to 3 ASCII decimal digits specifying a program block in the range 1 through 255
p = 70H

Following these returns the backup is terminated.

Because the workstation could be executing an OIL program when the Backup Programs command is transmitted, it is necessary to halt the operation of the currently executing program. Execution of the OIL program will resume after the backup is terminated.

6.2.14 Verify Program

Description This command causes the workstation to verify the contents of the restored program with the current program contents. Following each program verification, a response will be sent over the serial port indicating whether the verification passed or failed. This allows OIL programs to be verified while OIL programs are executing.

Syntax: <ESC>[37;<block#>p<text>

where: <ESC> = 1BH
[= 5BH
3 = 33H
7 = 37H
; = 3BH
<block#> = Program number for the workstation to verify (up to 3 ASCII characters, range 1 through 255)
p = 70H
<text of stored screen> = text of stored screen as ASCII characters
 = 7FH

When the workstation receives this command, OIL programs stop executing. When the sequence is completed, program execution resumes.

Returns:

In response to this command, the workstation will transmit one of the following commands:

<ESC>[38;0p

where: 3 = 33H
8 = 38H
0 = 30H

This indicates that the verification passed.

or

<ESC>[38;<block#>p

where: <block#> is the program number that failed

6.3 HOW REMOTE COMMANDS ARE PROCESSED

If the command is a valid command, it is placed into an internal input command queue from which it is executed. A valid command is one which follows the command format. A valid but incorrect command is one which follows the command format but which is not an available command. Some examples for serial remote commands follow:

Valid serial remote commands -

```
<ESC>[12;7p -- Execute program 7
<ESC>[12;009p -- Execute program 9
<ESC>[30;200;12345p -- Set data register #200 to the value 12345
<ESC>[31;10p -- Return the value of data register #10
```

Valid but incorrect serial remote commands (will be ignored and will not be transferred to any input queue) -

```
<ESC>[2p
<ESC>[;;;;;p
<ESC>[123456789p
```

Invalid serial remote commands (will be transferred to the input data queue) -

```
X
<ESC>p
<ESC>[x
<ESC>[30;200;12345q
<ESC>[31:10p
```


The following instructions describe how to save, reload and verify stored screens using the IBM PC.

7.1 Saving OIL Programs

1. Connect the workstation serial port chosen for backup/restore (via the Miscellaneous Configuration Menu) to the IBM COM1 serial port with a cable. The pin numbers used depend on the type of computer used. The IBM PC/XT uses a 25-pin connector as COM1, while the PC/AT uses a 9-pin connector. The pinouts of the cable are shown below:

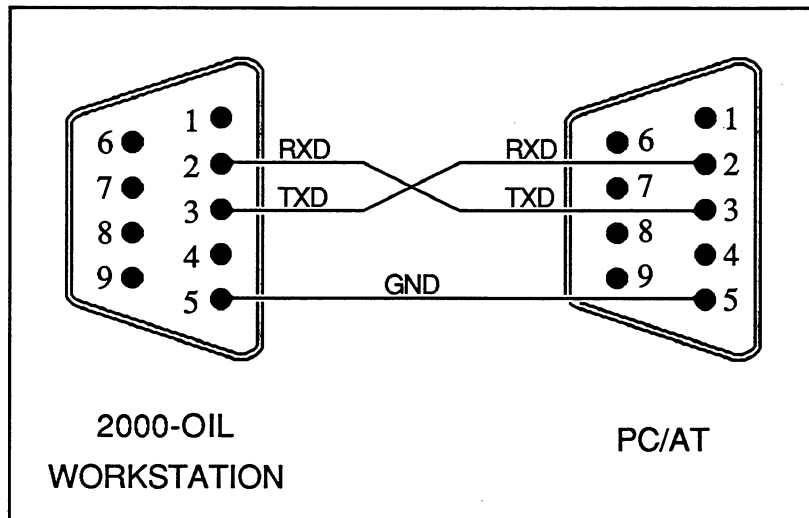


Figure 7-1. PC/AT to 2000 Connection

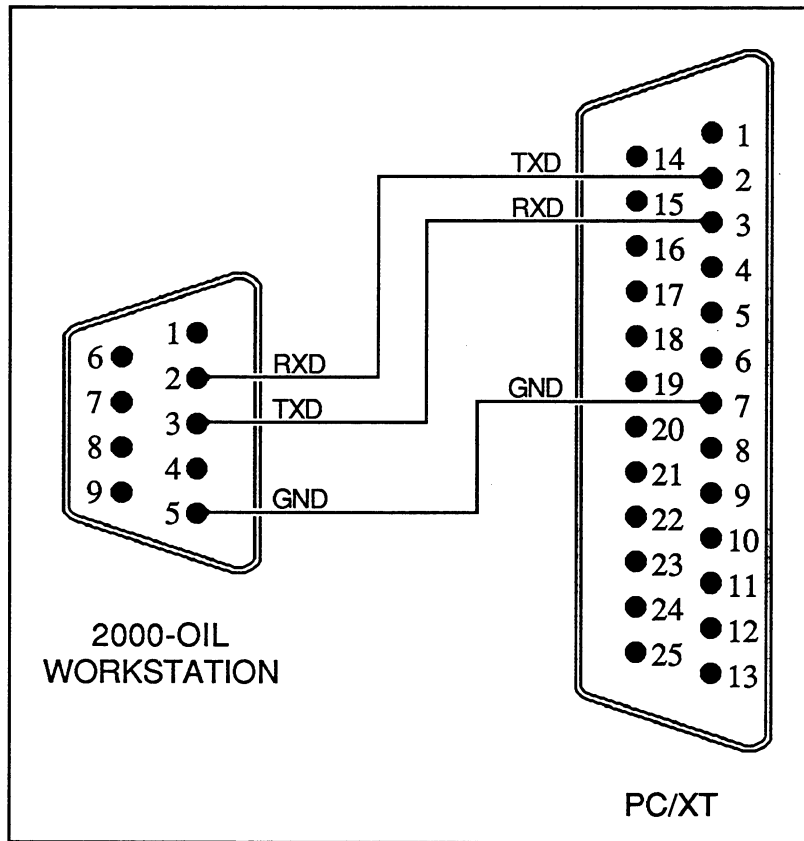


Figure 7-2. PC/XT to 2000 Connection

On the IBM PC, load and configure your particular ASCII communication program for the proper communication parameters.

2. Configure the workstation for the following parameters:

- Data Bits (to match communications package)
- Full Duplex
- Handshaking (XON, XOFF)
- Baud Rate (to match communications package)
- Parity (to match communications package)

3. From the workstation, choose the Backup/Restore function (Main Menu item #3), then choose the Backup function (Backup/Restore Menu item #1).

4. The screen prompts for a backup of programs, data registers, or both programs and data registers (all). Select one of the options.

If you selected backup of programs in the previous step, the workstation asks which of the 255 programs you wish to backup (a backup of all programs is also an option). When data registers are backed-up, they are all automatically selected. When the actual backup begins, the program numbers are displayed at the lower right of the screen.

7.2 Restoring Screen Programs

Perform Steps 1 and 2 from above.

3. From the workstation, choose the Backup/Restore function (Main Menu item #3), then choose the Restore function (Backup/Restore menu item #2).
4. On the PC, select to send the file that contains the backed-up program.

The screen(s) will be sent to the workstation and stored as the screen number you gave them when they were saved. This completes the program restoration.

7.3 Verifying Screen Programs

Perform Steps 1 and 2 from Section 6.4.1.

3. From the workstation, choose the Backup/Restore function (Main Menu item #3), then choose the Verify function (Backup/Restore menu item #3).
4. On the PC, select to send the file that contains the backed-up program.

The screen(s) will be sent to the workstation and verified. If an error occurs during verification, the workstation will immediately display a "Verification failed" message.

Appendices A-05 through A-19 - Firmware Upgrades

[Only the appendix that discusses the shipped firmware will be included in the documentation package. Remove this page and replace it with the appropriate appendix.]

Table B-1. OIL Commands by Section

COMMAND	SYNTAX*	SECTION
Screen Text	"{character}"	4.6.2
Register Value Display	reg	4.6.3
Position	@[xcoord],[ycoord]	4.6.4
Label	%label	4.6.5
Add to Register	ADD num,reg	4.6.6
AND	AND reg,num	4.6.7
Beep	BP	4.6.8
Draw Box	BX bval,[xstart],[ystart],[xend],[yend]	4.6.9
Clear Buffer	CB device	4.6.10
Clear Line	CL	4.6.11
Clear Bit	CLRB reg,bit	4.6.12
Clear Screen	CS	4.6.13
Clear Window	CW	4.6.14
Down	D [num]	4.6.15
Decrement Register	DEC reg	4.6.16
Define Zone**	DFZ upper left zone,lower right zone, num	4.6.17
Divide Register	DIV reg,num	4.6.18
Display String	DS reg	4.6.19
Execute Program and Reset Subroutine Counter	ER num	4.6.20
Execute Sub-Program Block	ES num	4.6.21
Execute Program Block	EX num	4.6.22
Exit from a Sub-Program	EXIT	4.6.23
Draw Filled Box	FBX bval,[xstart],[ystart],[xend],[yend]	4.6.24
Go	GO label	4.6.25
Horizontal Bar Left	HBL [xstart],[ystart],lngth,maxlngth	4.6.26
Horizontal Bar Right	HBR [xstart],[ystart],lngth,maxlngth	4.6.27
Draw Horizontal Line	HL lval,[xstart],[ystart],hlngth	4.6.28
If	IF cond[:command]	4.6.29
If Bit	IFB reg,bit[:command]	4.6.30
If String	IFS cond[:command]	4.6.31
Increment Register	INC reg	4.6.32
Exit ONKEY Subroutine Program	KEXIT	4.6.33

Table B-1. OIL Commands by Section (cont.)

COMMAND	SYNTAX*	SECTION
Keypad Status	KS reg	4.6.34
Left	L [,num]	4.6.35
Remainder (Modulus)	MOD reg,num	4.6.36
Multiply Register	MUL reg,num	4.6.37
New Line	NL [,num]	4.6.38
NOT	NOT reg	4.6.39
ONKEY	ONKEY num	4.6.40
ONTOUCH**	ONTOUCH num	4.6.41
OR	OR reg,num	4.6.42
Plot	PLOT xpoint,ypoint	4.6.43
Pause	PS num	4.6.44
Right	R [,num]	4.6.45
Reset Attributes	RE {attr}	4.6.46
Restore Attributes	RSTA reg	4.6.47
Restore Cursor Position	RSTC reg	4.6.48
Save Attributes	SAVA reg	4.6.49
Save Cursor Position	SAVC reg	4.6.50
Set Attributes	SE {attr}	4.6.51
Set Bit	SETB reg,bit	4.6.52
Stop	STOP	4.6.53
Store String	SS label,reg	4.6.54
Subtract from Register	SUB num,reg	4.6.55
Transfer System Characteristics	SYS reg	4.6.56
Exit ONTOUCH Subroutine Program**	TEXTIT	4.6.57
Transfer Data	TR source,dest	4.6.58
Transfer Text	TS source,dest	4.6.59
Transfer Touch Screen Data**	TT dest	4.6.60
Up	U [,num]	4.6.61
Unplot	UNPLOT xpoint,ypoint	4.6.62
Vertical Bar Up	VB VBU [xstart],[ystart],hght, maxhght	4.6.63
Vertical Bar Down	VBD [xstart],[ystart],hght,maxhght	4.6.64
Draw Vertical Line	VL lval,[xstart],[ystart],vlength	4.6.65
XOR	XOR reg,num	4.6.66

*See the key on page B-6 for syntax definitions.

**Touch screen option only.

Table B-2. OIL Commands by Function

COMMAND	SYNTAX*	SECTION
Screen Text	"{character}"	4.6.2
Register Value Display	reg	4.6.3
select character size and attributes (i.e., blinking or reverse video)		
Set Attributes	SE {attr}	4.6.51
Reset Attributes	RE {attr}	4.6.46
Save Attributes	SAVA reg	4.6.49
Restore Attributes	RSTA reg	4.6.47
Restore Cursor Position	RSTC reg	4.6.48
Save Cursor Position	SAVC reg	4.6.50
move the cursor		
Up	U [,num]	4.6.61
Down	D [,num]	4.6.15
Left	L [,num]	4.6.35
Right	R [,num]	4.6.45
Position	@[xcoord],[ycoord]	4.6.4
New Line	NL [,num]	4.6.38
draw figures and lines		
Draw Box	BX bval,[xstart],[ystart],[xend],[yend]	4.6.9
Draw Filled Box	FBX bval,[xstart],[ystart],[xend],[yend]	4.6.24
Draw Vertical Line	VL lval,[xstart],[ystart],vlength	4.6.65
Draw Horizontal Line	HL lval,[xstart],[ystart],hlength	4.6.28
Vertical Bar Up	VB VBU [xstart],[ystart],hght, maxhght	4.6.63
Vertical Bar Down	VBD [xstart],[ystart],hght,maxhght	4.6.64
Horizontal Bar Right	HBR [xstart],[ystart],length,maxlength	4.6.27
Horizontal Bar Left	HBL [xstart],[ystart],length,maxlength	4.6.26
Plot	PLOT xpoint,ypoint	4.6.43
Unplot	UNPLOT xpoint,ypoint	4.6.62

Table B-2. OIL Commands by Function (cont.)

COMMAND	SYNTAX*	SECTION
display data or move between the keyboard, data registers, and host computer		
Transfer Data	TR source,dest	4.6.58
Transfer Text	TS source,dest	4.6.59
Transfer Touch Screen Data**	TT dest	4.6.60
Store String	SS label,reg	4.6.54
Display String	DS reg	4.6.19
Transfer System Characteristics	SYS reg	4.6.56
conditional and looping commands		
If	IF cond[:command]	4.6.29
If Bit	IFB reg,bit:command	4.6.30
If String	IFS cond,[:command]	4.6.31
Go	GO label	4.6.25
Label	%label	4.6.5
set and clear bits of a data register		
Set Bit	SETB reg,bit	4.6.52
Clear Bit	CLRB reg,bit	4.6.12
arithmetic operations		
Increment Register	INC reg	4.6.32
Decrement Register	DEC reg	4.6.16
Add to Register	ADD num,reg	4.6.6
Subtract from Register	SUB num,reg	4.6.55
Multiply Register	MUL reg,num	4.6.37
Divide Register	DIV reg,num	4.6.18
Remainder (Modulus)	MOD reg,num	4.6.36
logical operations		
AND	AND reg,num	4.6.7
XOR	XOR reg,num	4.6.66
OR	OR reg,num	4.6.42
NOT	NOT reg	4.6.39

Table B-2. OIL Commands by Function (cont.)

COMMAND	SYNTAX*	SECTION
program nesting		
Execute Program and Reset Subroutine Counter	ER num	4.6.20
Execute Program Block	EX num	4.6.22
Execute Sub-Program Block	ES num	4.6.21
Exit	EXIT	4.6.23
Stop	STOP	4.6.53
miscellaneous		
Beep	BP	4.6.8
Clear Buffer	CB device	4.6.10
Clear Line	CL	4.6.11
Clear Screen	CS	4.6.13
Clear Window	CW	4.6.14
Define Zone**	DFZ upper left zone,lower right zone,num	4.6.17
Keypad Status	KS reg	4.6.34
Pause	PS num	4.6.44
ONKEY	ONKEY num	4.6.40
ONTOUCH**	ONTOUCH num	4.6.41
Exit ONKEY Subroutine Program	ONKEY	4.6.33
Exit ONTOUCH Subroutine Program**	TEXIT	4.6.57

*See the key on page B-6 for syntax definitions.

**Touch screen option only.

KEY

<u>SYMBOL</u>	<u>MEANING</u>
[]	Optional argument
{}	One or more occurrences of this argument can be included
	Choice of either argument
char	Any single character (e.g., N, m, \$)
reg	#n, \$n, or &n where n=1 through maximum number of data registers possible (10,499)
bit	A specific bit within a register (0-15)
num	A decimal value (0-65535), a hexadecimal value 9&0 - &FFFF), or a register
text	One or two characters enclosed in double quotes (e.g., "A" or "ra")
xcoord xstart xend	Column coordinate (0 - 79)
ycoord ystart yend	Row coordinate (0 - 23)
vlength	Vertical length of graphic (0 - 24)
hlength	Horizontal length of graphic (0 - 79)
hght maxhght	Height of graphic (0 - 255)
length maxlength	Length of graphic (0 - 255)
xpoint	Column coordinate (0 - 159)
ypoint	Row coordinate (0 - 79)

KEY (cont.)

<u>SYMBOL</u>	<u>MEANING</u>
bval	1 = thick line, 2 = thin line, any other character = figure composed of that character any other number = figure composed of characters corresponding to the number
lval	1 = thick line, 2 = thin line, 3-6 (see VL and HL commands) any other number = figure composed of characters corresponding to the number
source	Source for data in a transfer. Any of the following: KB (keyboard and keypad) SI (serial input) #n (data register) \$n (numerical argument) (text argument)
dest	Destination for data in a transfer. Any of the following: SC (screen output) SO (serial output) NO (dummy output) \$n (data register)
attr	Character attribute. Any of the following: QS (quad-size) DS (double size) RV (reverse video) HI (highlight) UN (underline) BL (blinking) DW (double-wide) DH (double-high) SC (screen output) PR (printer output) SS (screen scrolling) CU (cursor on/off) G1 (process graphics) G2 (thin-line and block graphics) G3 (process control graphic connectors) G4 (mini Process graphics)
cond	A logical function inserted between two numerical arguments. It can be: <, >, <>, =, =>, >=, <=, or =<

KEY (cont.)

<u>SYMBOL</u>	<u>MEANING</u>
label	A sequence of alphanumeric characters
device	A device used to handle data. It can be: KB (keyboard or keypad) PI, SI (primary and secondary serial input) PO, SO (primary and secondary serial output) TS (touch screen)

C.1 TOUCH SCREEN

The 2000T touch screen option available for the 2000 Industrial Workstation functions differently when the Workstation is equipped with OIL. With OIL, a touch screen zone will transmit its zone number, as shown in Figure C-1, below.

Figure C-1. Default Codes Transmitted by Touch Screen Zones
(OIL Firmware Installed)

Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9	Z10
1H	2H	3H	4H	5H	6H	7H	8H	9H	10H
Z11	Z12	Z13	Z14	Z15	Z16	Z17	Z18	Z19	Z20
11H	12H	13H	14H	15H	16H	17H	18H	19H	20H
Z21	Z22	Z23	Z24	Z25	Z26	Z27	Z28	Z29	Z30
21H	22H	23H	24H	25H	26H	27H	28H	29H	30H
Z31	Z32	Z33	Z34	Z35	Z36	Z37	Z38	Z39	Z40
31H	32H	33H	34H	35H	36H	37H	38H	39H	40H
Z41	Z42	Z43	Z44	Z45	Z46	Z47	Z48	Z49	Z50
41H	42H	43H	44H	45H	46H	47H	48H	49H	50H
Z51	Z52	Z53	Z54	Z55	Z56	Z57	Z58	Z59	Z60
51H	52H	53H	54H	55H	56H	57H	58H	59H	60H
Z61	Z62	Z63	Z64	Z65	Z66	Z67	Z68	Z69	Z70
61H	62H	63H	64H	65H	66H	67H	68H	69H	70H
Z71	Z72	Z73	Z74	Z75	Z76	Z77	Z78	Z79	Z80
71H	72H	73H	74H	75H	76H	77H	78H	79H	80H

KEY:

Z = Zone (e.g., Z1 = touch screen zone 1)

C.2 TOUCH SCREEN OIL COMMANDS

There are four OIL commands available on the 2000 Industrial Workstation for use with the touch screen option. These commands are used to define a touch screen zone, execute a subroutine program block when a zone is pressed, terminate a subroutine program block, and transfer touch screen data to a register. These OIL commands are described below. (They are also described in Chapter 4.)

C.2.1 Define Zone (DFZ)

Syntax: DFZ upper left zone,lower right zone,num

where:

upper left zone	is the upper left boundary of the zone to be defined
lower right zone	is the lower right boundary of the zone to be defined
num	is a decimal (1-255) or hexadecimal (1-FF) number

With OIL, a touch screen zone will return it's zone number when pressed (ie., zone 1 returns a 1, zone 8 returns an 8). The DFZ command sets a zone or block of contiguous zones to return a common code when pressed. Upon power up zones are reset to return their zone numbers.

Example 1: DFZ 1,18,&81

This example causes touch screen zones 1-8 and 11-18 to return the value 81H when pressed. Other zones remain unaffected.

Example 2: DFZ 12,12,&45

This example causes the single touch screen zone 12 to return the value 45H when pressed. Other zones remain unaffected.

C.2.2 ONTOUCH

Syntax: ONTOUCH num

where: num is the program block to be executed (1-255).

This command is used within a program block and will be executed whenever any zone of the touch screen is pressed. ONTOUCH 0 disables the ONTOUCH feature.

Because ONTOUCH uses a zone in the touch screen buffer, care must be taken to assure that the character is not removed by the subprogram block specified in the ONTOUCH command. If the character is not removed, the specified subprogram block will be executed continuously (see TEXTIT).

Example 1: ;ONTOUCH example - Display zone number on screen
TR,0,#30 ;Zero response register
ONTOUCH,2 ;Execute subprogram block 2 when a zone is pressed
"Touch Zone Pressed:" ;Print prompt
%Disp ;Display loop
@20,0 ;Position cursor
TR,#30,SC(DDD) ;Display zone pressed
GO,Disp ;Loop

This example will zero register #30, execute subprogram block 2 when a touch screen zone is pressed, and continuously print the touch screen zone number that is pressed.

C.2.3 Exit ONTOUCH Subroutine Program (TEXTIT)

Syntax: TEXTIT

The command TEXTIT terminates the subroutine program block requested by the OIL command ONTOUCH.

NOTE

This command **MUST** terminate the subprogram block specified by the ONTOUCH command or the workstation will not re-enable ONTOUCH trapping upon completion of the routine (see ONTOUCH).

Example: TT #30 ;Touch screen zone to register 30

The above program will transfer a touch screen zone out of the touch screen buffer, put the value in register #30, and terminate.

C.2.4 Transfer Touch Screen Data (TT)

Syntax: TT reg

where: reg is the register to transfer zone number to

This command transfers the number of the currently pressed touch screen zone (1 - 80), or the value selected for that zone via the DFZ command, to a specified register. The released state is zero.

Example: TT #35 ;transfer touch screen data to register

The above program will transfer the number of the currently pressed touch screen zone to register #35. For example, if zone 1 (upper left hand corner of touch screen) is pressed, that zone number will be transferred to register #35.

NOTE

Codes specified via the "Define Touch Screen Zone" command take precedence over the default codes. If any zones have been assigned new codes, those codes are the codes that will be transferred with the "Transfer Touch Screen Data" command.